DYNAMIC SAFETY ASSESSMENT OF FPGA-BASED SAFETY CRITICAL SYSTEMS WITH APPLICATIONS IN NUCLEAR POWER GENERATION

By:

PHILLIP MCNELLES

A THESIS SUBMITTED TO THE FACULTY OF ENERGY SYSTEMS AND NUCLEAR SCIENCE AT THE UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILIPSOPHY IN NUCLEAR ENGINEERING

FACULTY OF ENERGY SYSTEMS AND NUCLEAR SCIENCE

UNVIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

2000 SIMCOE STREET NORTH, OSHAWA, ONTARIO, CANADA, L1H 7K4

December, 2016

DYNAMIC SAFETY ASSESSMENT OF FPGA-BASED SAFETY CRITICAL SYSTEMS WITH APPLICATIONS IN NUCLEAR POWER GENERATION

DR. LIXUAN LU, SUPERVISOR

DR. ANTHONY WAKER, COMMITTEE MEMBER

MR. JOHN FROATS, COMMITTEE MEMBER

DR. WALID MORSI IBRAHIM, UNIVERSITY EXAMINER

DR. ZHIGANG TIAN, EXTERNAL EXAMINER

PHILLIP MCNELLES, CANDIDATE

Abstract

Field Programmable Gate Arrays (FPGAS) are a type on integrated circuit that is configured by the end user to perform desired digital logic functions. FPGAs do not run any software or operating system, as the logic functions are configured as a hardware implementation on the FPGA chip. Documentation from the International Atomic Energy Agency (IAEA) states that FPGA implementations of I&C systems in Nuclear Power Plants (NPPs) is expected to increase significantly in the future. One issue facing FPGAs in the nuclear field is a lack of technical standards and design/review documentation. Therefore, the research program undertaken during this thesis considered the application of a new safety analysis methodology for the modelling and analysis of FPGA-based systems. The methodology chosen is a modern, dynamic (time-dependant) methodology known as the Dynamic Flowgraph Methodology (DFM), which is intended to be applied to digital I&C systems. Initially, a Failure Modes and Effects Analysis (FMEA) was performed to ascertain the potential failure modes that could affect FPGA-based systems, and that FMEA data was used to create and FPGA failure modes taxonomy. Using that FMEA data to provide information for fault injection, DFM was applied to analyze several FPGA-based test systems, and the results of the DFM analyses were compared and contrasted with results from Fault Tree Analysis (FTA), to determine the potential advantages and disadvantages of DFM. It was seen that DFM had several advantages when modelling clock delays, oscillating clock signals, and Multiple-Valued Logic, however for large systems DFM continues to experience the “state explosion” problem, limiting its effectiveness to small-medium sized systems. Potential avenues of future work are also presented.

ACKNOWLEDGEMENT

I would like to thank my supervisor, Dr. Lixuan Lu for taking me on as her student for PhD degree, and for all of her support and guidance throughout my degree. I am also greatly appreciative of her allowing me to intern at the Canadian Nuclear Safety Commission (CNSC) for the final two years of my doctorate, as that was another great experience.

I would also like to thank Dr. Anthony Waker and Professor John Froats for agreeing to be on my committee, as well as Dr. Walid Morsi Ibrahim and Dr. Zhigang Tian for being the examiners at my defence. All of these individuals provided me with excellent feedback and comments regarding improvements to my thesis document, and for suggestions for potential topics for the continuation of this research.

Lastly, I would like to thank Zhao Chang (Charles) Zeng and Guna Renganathan for supervising my research at the CNSC. They were always happy to help with the research projects and answer any of my questions, and I was able to learn a great deal from them. Additionally, I want to thank my directors at the CNSC, Greg Lamarre and Yolande Akl, for bringing me in to work with their divisions, as well as Sophie Gingras and Marc Leblanc, for allowing me to continue with the research program during my time in the Secretariat.

TABLE OF CONTENTS

List of Figures

List of Tables

Glossary

Acronym	Definition
AECL	Atomic Energy of Canada Ltd.
AES	Advanced Encryption Standard
AF	Address Fault
AIAA	American Institute of Aeronautics and Astronautics
ALS	Advanced Logic System
APRM	Average Power Range Monitor
ASIC	Application Specific Integrated Circuit
ASTS	Automatic Seismic Trip System
BC	Boundary Conditions
BDD	Binary Decision Diagrams
BER	Bit Error Rate
BI	Birnbaum Importance Measures (Risk Importance Measure)
BIST	Built-In Self-Test
BPA	Bent Pin Analysis
BTI	Bias Thermal Instability
CAFTA	Computer Aided Fault Tree Analysis
CB	Complete Base
CCF	Common Cause Failure
CCMT	Cell-To-Cell Mapping Technique
CCSF/SCCF	Common Cause Software Failure
CDC	Clock Domain Crossing
CDM	Charged Device Model
CFMA	Cable Failure Matrix Analysis
CIM	Component Interface Module
CNSC	Canadian Nuclear Safety Commission
CPLD	Complex Programmable Logic Device
CRC	Cyclic Redundancy Check
CSA	Canadian Standards Association (CSA Group)
CSNI	Committee on the Safety of Nuclear Installations
D3	Defence in Depth
DAS	Diverse Actuation System
DB/DBI	Dynamic Birnbaum Importance (Risk Importance Measure)
DCC	Digital Control Computers
DCM	Digital Clock Management
DFCS	Digital Feedwater Control System
DFM	Dynamic Flowgraph Methodology
DFV	Dynamic Fussel-Vesely (Risk Importance Measure)
DI	Dynamic Risk Increase Worth
DI&C	Digital Instrumentation and Control

Acronym	Definition
DICREL	Digital Instrumentation and Control Reliability Group
DICWG	Digital Instrumentation and Controls Working Group
DMR	Double Modular Redundancy
DPC	Direct Probability Calculator
DPS	Diverse Protection System
DR	Dynamic Risk Decrease Worth
DRF	Data Retention Fault
E/E/PE	Electrical/Electronic/Programmable Electronic
ECC	Error Correction Codes
EDAC	Error Detection and Correction
EDC	Error Detection Codes
EEPROM	Electrically Erasable Programmable Read-Only Memory
EM	Electromigtration
EOS	Electrical Overstress
ESA	European Space Agency
ESA SA	European Space Agency Sneak Analysis
ESFAS	Emergency Safety Features Actuation System
ESD	Electrostatic Discharge
EPRI	Electric Power Research Institute
EQ	Exact Quantification
ET	Event Tree
EXC	Extensive Conditions
FBD	Functional Block Diagrams
FIFO	First-In First-Out
FMEA	Failure Mode Effects and Analysis
FPAA	Field Programmable Analog Array
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
FTA	Fault Tree Analysis
FV	Fussel-Vesely (Risk Importance Measure)
FXP	Fixed Point Data Representation
HBM	Human Body Model
HCE/HCI/HCD	Hot Carrier Effects/Hot Carrier Injections/Hot Carrier Degradation
HDL	Hardware Description Language
HER	Hard Error Rate
HIL	Hardware in the Loop
HPD	HDL Programmed Device
HPS	Hard Processor System
HSI	Human-System Interface
I&C	Instrumentation and Control
IAEA	International Atomic Energy Agency
IB	Irredundant Base
IC	Integrated Circuit

Acronym	Definition
IEC	International Electrotechnical Commission
IEEE	International Institute of Electrical and Electronics Engineers
I/O	Input/Output
IP	Intellectual Property
ISO	International Organization for Standardization
JEDEC	Joint Electron Device Engineering Council
JEP	JEDEC Publication
JTAG	Joint Test Action Group
LPRM	Local Power Range Monitor
LSELS	Load Shedder and Emergency Load Sequencer
LUT	Look-Up Table
MATLAB	Matrix Laboratory
MBU	Multiple Bit Interrupt
MCS	Minimal Cut Set
MCSUB	Minimal Cut Set Upper Bound
MCU	Multiple Cell Interrupt
MEI	Mutually Exclusive Implicant
MFTE	Main Feedwater Turbine Electron-Hydraulic
MHD	Moving Head Disk
MFV	Main Feed Valve
MIU	Multiple Independent Upset
MTBF	Mean Time Between Failure
MTBMO	Mean Time Between Metastability Occurrence
MTTE	Mean Time To Event
MTTF	Mean Time To Failure
MUX	Multiplexer
MVL	Multi-Valued/Many-Valued Logic
MSFS	Main Steam and Feedwater Isolation System
NEA	Nuclear Energy Agency
NBTI	Negative Bias Thermal Instability
NPP	Nuclear Power Plant
NUREG	US NRC Technical Report Designation (Nuclear Regulatory Commission)
OECD	Organization for Economic Co-operation and Development
OPG	Ontario Power Generation
ORNL	Oak Ridge National Laboratory
OS	Operating System
OTP	One-Time Programmable
PAL	Programmable Logic Array
PAR	Place-and-Route
PBTI	Positive Bias Thermal Instability
PCB	Printed Circuit Board
PCM	Phase Change Memory

Acronym	Definition
PDD	Programmable Digital Device
PI	Prime Implicant
PID/PDI	Proportional Integral Derivative (Controller)
PLA	Programmable Logic Array
PLC	Programmable Logic Controller
PLD	Programmable Logic Device
PLL	Phase Locking Loop/Phase Locked Loop
PNPSF	Passive Neighbourhood Pattern Sensitive Fault
PPS	Process Protection System
PSA/PRA	Probabilistic Safety Assessment/Probabilistic Risk Assessment
PRBS	Pseudo-Random Binary Sequence
PRM	Power Range Neutron Monitor (BWR)
PRNM	Power Range Neutron Monitor (ABWR)
PRPS	Primary Reactor Protection System
PRWS	Pseudo-Random Word Sequence
PUF	Physically Uncloneable Function
REGDOC	Regulatory Document (CNSC)
RAM	Random Access Memory
RAW	Risk Achievement Worth
RCS	Rod Control System
RDR	Risk Decrease Ratio
Regt	Register
RIC	Reactor In-Core Measurement System
RIH	Reactor Inlet Header
RIR	Risk Increase Ratio
ROH	Reactor Outlet Header
ROM	Read Only Memory
RPCLS	Reactor Power Control and Limitation System
RPS	Reactor Protection System
RRW	Risk Reduction Worth
RTIS	Reactor Trip and Isolation System
RTL	Register Transfer Level
SART	Smart Alternative Routing Technique
SBU	Single Bit Interrupt
SCA	Sneak Circuit Analysis
SDS	Shutdown System
SEB	Single Event Burnout
SED	Single Event Disturb
SEDB	Single Event Dielectric Breakdown
SEE	Single Event Effect
SEFI	Single Event Functional Interrupt
SEGR	Single Event Gate Rupture
SEL	Single Event Latch-up

Acronym	Definition
SEMT	Single Event Multiple Transient
SEMU	Single Event Multiple Upset
SER	Soft Error Rate
SESB	Single Event Snapback
SET	Single Event Transient
SEU	Single Event Interrupt
SFBDD	Structure Function Binary Decision Diagram
SFT	Standard/Static Fault Tree
SHE	Single Hard Error
SM	Stress Migration
SMHA	State Machine Hazard Analysis
SNR	Signal to Noise Ratio
SOC	System on a Chip
SOF	Stuck-Open Fault
SPAR	Standardized Plant Analysis Risk
SRAM	Static Random Access Memory
SRNM	Start-Up Neutron Monitor
SSLC	Safety System Logic and Control
SSN	Simultaneous Switching Noise
SSPS	Solid State Protection System
SSWA	Sneak Software Analysis
STA	Static Timing Analysis
SUM	Rare Event Approximation
TC	Thermal Cycling
TDDB	Time-Dependant Dielectric Breakdown
TDRFP	Turbine Driven Feedwater Reactor Pumps
TG-FAN	Topical Group on Field Programmable Gate Array Applications in Nuclear Power Plants
TMR	Triple Modular Redundancy
TPI	Timed Prime Implicant
TSP	Trip Setpoint
TTL	Transistor-Transitory Logic
USNRC	United States Nuclear Regulatory Commission
V&V	Verification and Validation
VBMC	VHDL Bounded Model Checker
VHDL	Very High Speed Integrated Circuit HDL
VHSIC	Very High Speed Integrated Circuit
VLSI	Very Large Scale Integration
VTT	Valtion Teknillinen Tutkimuskeskus (Technical Research Centre of Finland)
WGRISK	Working Group on Risk Assessment
YADRAT	Yet Another Dynamic Reliability Analysis Tool

1 Introduction

Chapter 1 provides an introduction to the research program, as well as an introduction to the actual thesis document. Sub-section 1.1 provides a brief outline of thesis, including the topics discussed in each chapter. Sub-section 1.2 discusses the motivation for the topics considered in this research program: FPGAs and the dynamic reliability analysis methodologies. Sub-Section 1.3 presents the novelty of the research done in this thesis, as well as the contribution of this work with regards to scientific and technical knowledge. Sub-section 1.4 provides a summary of the topics discussed in this chapter.

1.1 Thesis Outline

This thesis document is organized in the following chapters/sections. Chapter 1 provides a basic outline of the chapters in the overall thesis document, as well as a discussion on the overall motivation regarding the selection on the research program. This includes the importance of FPGA-based systems research to the nuclear power industry, and the importance of employing modern dynamic reliability analysis techniques to those FPGA-based systems.

Chapter 2 provides the background information on two important topics regarding this thesis:, a description of FPGAs and FPGA-based systems, and a description of the reliability analysis methodologies that were applied during the research program. Regarding the information on FPGAs, the specific properties of FPGAs, types of FPGAs, individual sub-components, and the unique advantages and challenges posed by FPGAs with regards to safety-critical systems will be considered. A detailed literature review of the use of FPGAs in I&C systems in nuclear power plants is provided, highlighting international FPGA implementations and research projects. Afterwards, a detailed discussion of the reliability methodologies is presented. The two methods considered were Fault Tree Analysis (FTA), and the Dynamic Flowgraph Methodology (DFM). The discussion on FTA will consider coherent and non-coherent logic, and popular methods algorithms for analyzing Fault Trees, including cut-set methods and Binary Decision Diagram (BDD) methods. The information regarding DFM will include Multiple-Valued/Many Valued Logic (MVL), methods for solving DFM models, DFM models, tools and features, and advanced rules for solving DFM models.

Chapter 3 presents a failure mode taxonomy of FPGA failure modes. An extensive literature survey was performed that compiled and categorized the potential failure modes of FPGA-based systems. A failure modes taxonomy, for digital (software-based) systems was previously published by the OECD-NEA, and was used to re-categorize the FPGA failure modes. The OECD-NEA taxonomy was extended to include FPGAs, and an additional layer of abstraction was added to fully incorporate the hardware and "software" (HDL code) errors. The same example system in the original OECD-NEA taxonomy (Reactor Trip System/Engineered Safety Features Actuation System) was used, to showcase the FPGA taxonomy.

Chapter 4 provides the results of research on the application of DFM for modelling and analyzing FPGA-based systems. It will first focus on introductory modelling of an FPGA-Based Post Accident Monitoring Systems (PAMS), to demonstrate basic deductive, inductive, qualitative and quantitative DFM analyses. Secondly, DFM will be used to model four important aspects of FPGA-based system, and the DFM analysis results are compared to simulations from the ModelSim logic simulator software, to confirm the accuracy of the DFM results. Following that preliminary research, in-depth comparisons on the use of DFM and FTA for analyzing FPGA-based systems (which include the failure mode information from Chapter 3) were performed. A one-channel, one-parameter FPGA-based trip logic loop, based off of the "Over-Temperature" trip parameter for an AP1000 nuclear reactor, was used as a test system. The DFM models and FTA fault trees were created for "Missed Trip" and "Spurious Trip" Top Events. Comparisons of the results included comparisons of the Top Event probabilities, Prime Implicants/Minimal Cut Sets and the Birnbaum Structural Importance measure. Potential reasons for these differences are discussed, including a detailed analysis of the underlying theory and algorithms used by FTA and DFM. Finally, a modified test system was used to compare dynamic results for an FPGA-test system, including dynamic Top Event probabilities, dynamic Prime Implicants, and dynamic Fussel-Vesely importance measures.

Chapter 5 provides an overall discussion on the use of DFM for the modelling and analysis of FPGA-based system, based on the results of the research program. These discussions will include the advantages and disadvantages of the application of DFM to FPGA system, as well as comparisons to the other reliability analysis methodologies considered during this research program.

Chapter 6 with discuss the conclusions of the research program, and present potential future avenues of research.

The list of references is found following Chapter 6. Several appendices are found after, which list the presentations/publications, a glossary of terms and definitions found in this research, and permission letters for the use of certain figures included in this thesis.

1.2 Research Motivation

The overall research program revolved around the analysis of FPGA-based systems using the Dynamic Flowgraph Methodology. There were several factors behind the selection of these two elements. Sub-section 1.2.1 will explain the importance of FPGA research in the nuclear domain. Sub-section 1.2.2 will explain the rationale behind dynamic methods, and sub-section 1.2.3 discusses the motivation for the selection of DFM as the main reliability analysis methodology.

1.2.1 Motivation for FPGA Research

Information published in the technical literature from international organizations such as the International Atomic Energy Agency (IAEA) and the Electric Power Research Institute (EPRI) discusses the importance of FPGA-based systems with respect to the nuclear field. Sub-section 1.2.1.1 presents the international perspective on the importance of the potential for expanded use of FPGA-based NPP systems in the future. Sub-section 1.2.1.2 discusses the specific uses that FPGAs are likely to see regarding NPP I&C systems.

1.2.1.1 International Perspectives on the Importance of FPGA-Based Systems

According to documents from the Topical Group on Field Programmable Gate Array Applications in Nuclear Power Plants (TG-FAN) of the IAEA, "An increased number of FPGA based applications can be expected as nuclear operators and regulators become more familiar with the advantages of the technology" and that "…the technology is expected to be applicable to large scale replacement of I&C systems in modernization projects, as well as providing complete I&C systems (safety and non-safety) in new nuclear power plant designs" [1]. It was also stated that "The implementation of FPGA based safety and non-safety related applications in operating and new plants is expected to grow substantially" [1].

Therefore, the perspective from the international community is that there will be significantly more FPGA-based system implementation in the future, making the design, analysis and review of those systems an increasingly important field of work.

Furthermore, the effect of FPGAs and similar technologies has been listed as one of the seventeen "technical challenges" facing digital I&C systems in NPPs, according to the IAEA [2]. This is in part, because although FPGAs have seen increased implementations in NPP I&C functions, those are mainly recent implementations, so information regarding "lessons learned" and international technical standards are not prevalent. Briefly, these challenges are summarized as [2]:

1.) Limited information on operational experience and lessons learned in FPGA NPP applications
2.) Only one international standard, published by the International Electrotechnical Commission (IEC) exists, but has not been universally adopted
3.) Few suppliers of FPGAs, design tools, and FPGA-based I&C systems specific to NPPs
4.) FPGA-based system design/review is not always user friendly
5.) FPGA design tools may be less mature than equivalent design tools for software-based systems, and changes in those tools may affect the suitability of FPGAs in NPP systems

A more detailed discussion on the limits of FPGA-based systems with regards to NPPs is given in Section 3.0.

However, FPGAs are still expected to see expanded use in a variety of NPP I&C systems, and many implementation and research projects have/are taking place worldwide [3–7]. Therefore, the increased use of FPGA systems and the need for more technical information makes the modelling, safety and reliability analysis of FPA-based systems an important and practical endeavour.

1.2.1.2 Potential Uses for FPGAs in Nuclear Power Plants

Typically, an FPGA is intended to carry out relatively simple, well-defined and well-bounded digital logic functions [5]. These types of functions are found in safety function actuation logic, priority logic, component control logic, data communication, etc. IT has been stated that FPGAs are principally suited

for safety systems and other high-reliability applications, due to their fast response time, reduced complexity, and that safety-critical systems often utilize relative simple logic functions [5].

Out of all the expected applications, the implementation of primary reactor protection systems was said to be the most critical in the NPPs. Several other systems/applications of particular interest have been identified, including emergency diesel generators and load sequencers, diverse actuation systems and post-accident monitoring systems. Additionally, FPGAs have been considered for non-safety systems, and also for use in simple human-system interfaces [1].

In terms of replacement systems, FPGAs have seen use (or are being considered for) the replacement of obsolete systems in existing NPPs. In certain cases, complex logic is still implemented in analog systems, which requires a large number of circuit boards, wiring and cabinetry. The same logic functions could be implemented in a single (or a small number of FPGAs), significantly decreasing the amount of components, wiring and space that is required by the equivalent analog system [1]. Furthermore, FPGAs are considered for the replacement of digital systems as well, for obsolete components that are no longer supported, and replacement parts cannot be obtained [3].

FPGAs are also being considered for use alongside software-based systems (such as in diverse systems or back-up systems), for the purpose of increased diversity and defence-in-depth [1]. Diversity is seen as a method to mitigate common cause failures, and as such, the use of different technologies will increase the level of diversity, and therefore reduce the risk of a common cause failure. Examples of this include a system with a primary microprocessor and FPGA-backup, primary FPGA with microprocessor back-up, or primary FPGA with diverse FPGA back-up (such as different chip model, manufacturer or technology) [5]. Similarly, FPGAs are considered for use as dedicated communication links in complex I&C systems, as those links are thought to be another source of common cause failure, to defend against the propagation of failures through an I&C system. This increased level of defence-in-depth is also a recommended defence against cyber-security attacks, making the diverse system more resistant to tampering and other malicious acts [1,5].

An additional consideration is the reduced complexity of FPGAs and the resulting FPGA-based systems. As the final logic in an FPGA-based system will be a pure hardware implementation, and there is no actual software or operating system running on the FPGA chip, it is believed that the Verification and Validation (V&V) process for FPGA-based systems would be much simpler than for traditional software-

based systems, such as PLCs [1,4]. This, in turn would simplify the licensing process for the FPGA-based systems, potentially allowing for a shorter, less expensive licensing process for the system vendors, as well as the operators of the nuclear power plants, when compared to the licensing process for software-based systems. A more detailed discussion on the potential advantages of FPGA-based systems is provided in sub-section 2.1.7.

Overall, the replacement of aging, obsolete digital and analog I&C systems, the construction of brand new systems based on FPGA technology, and the use of FPGAs as a diverse back-up/primary system are areas that are likely to see increased implementations in the future [1].

1.2.2 Motivation for the Selection of a Dynamic Reliability Analysis Methodology

A "Dynamic Methodology" is defined as "those that can account for the coupling between systems through explicit consideration of the time element in system evolution"[8]. With the increased use of digital technology in NPPs, dynamic methodologies have garnered more attention in recent years. The Committee on the Safety of Nuclear Installations (CSNI), as part of the Organization for Economic Co-operation and Development Nuclear Energy Agency (OECD-NEA), published a document in 2015 entitled "*Failure Modes Taxonomy for Reliability Assessment of Digital I&C Systems for PRA"* [9]. The work showcased in that taxonomy report represents an extensive research project, where initial results were published in a previous document (NEA/CSNI/R(2009)/18) entitled "*Recommendations on Assessing Digital System Reliability in Probabilistic Risk Assessments of Nuclear Power Plants*" [10]. Although this second document is older than the recently-published taxonomy report, that taxonomy report states that "many of the recommendations in given in the previous digital I&C report (NEA/CSNI/R(2009)/18) are still valid" [9].

The OECD-NEA taxonomy document states the most of the participants used FTA for the modelling, however it is also stated that it is not clear if FTA can "capture all dependencies, fault tolerant features and software hardware interactions", with regards to digital I&C systems [9]. For this purpose, the United States Nuclear Regulatory Commission (USNRC) sponsored several recent studies on the use of dynamic methodologies for modelling/analyzing digital I&C systems. In the NEA/CSNI/R(2009)/18 document, it was stated that "dynamic modelling" would be a topic of future/continued research. Also

regarding dynamic methods, it stated was that"…several participants indicated that such methods might be warranted when modelling software-based control systems. Several organisations are carrying out research projects in this area and some pointed out that the benefits of the research include evaluating the added value of dynamic methods and helping to identify weaknesses of a system" [10].

A similar sentiment is presented in the NUREG/CR-6901 report, which states "While the static event-tree/fault-tree (ET/FT) approach has been used in the reliability modeling of digital I&C systems in nuclear power plants, numerous concerns have been raised in the reliability literature in the past about the capability of the ET/FT approach to properly account for Type I interactions. Studies reported in the literature indicate that such interactions may lead to coupling between the triggered or stochastic logical events (e.g., valve openings, pump start-ups) during an accident with significant impacts on the predicted system failure probabilities. Similar arguments can be made for Type II interactions as well, based on the computational evidence for very simple situations. The lack of treatment of such dynamic interactions means that potentially significant dependencies between the failure events may not be identified or properly quantified." [8].

These two types of interactions, "Type I" and "Type 2" are defined as [8,11,12]:

Type 1 Interactions:

Dynamic Interactions between physical process variables (e.g. temperature, pressure, etc.) and the I&C systems that monitor and manage the process.

Type 2 Interactions:

Dynamic Interactions within the I&C system itself due to the presence of software/firmware (i.e. multi-tasking and multiplexing).

Furthermore, the issue of "Reliability", with regards to the Probabilistic Risk Assessment of digital systems, was also included as one of the "Technical Challenges" by the IAEA [2]. In that document, it was stated that "Digital systems present difficulties for traditional methods owing to their use of software for which systematic failure modes dominate the random modes of failure normally modelled in PRAs.

This introduces the potential for complex interdependencies as I&C systems influence most aspects of plant control, protection and monitoring" [2].

As FPGAs are a form of digital technologies, that FPGA-based systems would be digital systems, and would share some properties of other digital systems. Therefore, it was decided to apply a more modern, dynamic methodology, for the purpose of modelling and analyzing the FPGA-based I&C systems.

1.2.3 Motivation for the Selection of the Dynamic Flowgraph Methodology

After the choice to employ a dynamic methodology was made, the exact methodology had to be selected. Several of these dynamic methodologies exist, and have been reviewed in the literature [13]. In the end, the Dynamic Flowgraph Methodology (DFM) was selected [14]. This decision was based on information obtained from the literature, including the review and assessment of dynamic methodologies from NUREG/CR-6901 [8].

1.2.3.1 NUREG/CR-6901 Review and Assessment

A US NRC contractor report entitled "*Current State of Reliability Modeling Methodologies for Digital Systems and Their Acceptance Criteria for Nuclear Power Plant Assessments*" provided a detailed review and assessment of dynamic reliability analysis methodologies [8]. This report reviewed 13 potential dynamic methodologies (including DFM), against eleven assessment criteria. The results of that assessment are shown in Table 1, with the acceptance requirements discussed afterwards [8]. In Table 1, the "X" denotes that the methodology fulfills the requirement, the "O" denotes that the methodology does not fulfill the requirement and a value of "?" means that more research is needed to make determination of if the methodology will or will not meet the requirement.

Table 1: Dynamic Methodologies and Acceptance Requirements

Requirement/ Methodology	1	2	3	4	5	6	7	8	9	10	11
Continuous Event Tree	X	X	X	X	O	?	?	X	?	?	O
Dynamic Event Tree	X	X	X	?	X	?	?	?	X	X	O
Markov Models	X	X	X	X	O	?	X	X	?	?	O
Monte Carlo Simulation	X	X	X	X	?	?	?	?	?	?	O
Petri Net	X	X	X	X	O	?	?	?	?	?	O
DFM	X	X	X	?	X	?	?	?	X	X	X
Dynamic Fault Tree	X	?	?	?	X	?	X	?	X	?	X
ESD	X	X	X	X	O	?	?	?	X	X	O
Go-Flow	X	?	X	?	O	?	?	?	X	X	X
Bayesian Methods	X	?	?	?	O	O	?	?	?	?	X
Test Based Approaches	?	?	X	O	X	?	X	X	?	O	X
Software Metrics	O	?	O	O	?	?	X	X	O	O	X
Schneidewind Model	X	?	?	?	?	?	?	?	O	O	X

Acceptance Requirements:

1.) The model needs to accurately predict encountered failures and future failures

2.) The model needs to account for the important parameters of the system being analyzed/modelled

3.) The assumptions used in the model must be reasonable

4.) The model needs to give an accurate representation of the quantitative values of the dependencies between failure events

5.) The model must not be hard to understand and implement

6.) The quantitative data used in the model construction/analysis needs to be credible

7.) The model needs to differentiate states that fail one safety check from states that fail multiple stafety checks

8.) The model needs to differentiate between faults that cause intermittent failures and faults that cause function failures.

9.) The model needs to provide useful information to the users (such as cut sets, failure probabilities and uncertainty values)

10.) The methodology needs to model the digital components of the I&C system(s) under accident scenarios with a level of accuracy so that the non-digital components of the I&C can be properly analyzed

11.) The model does not need continuous state or strongly time-dependant plant state information

The assessment in the literature returned two methodologies with "… the most positive features and least negative or uncertain features when evaluated against the requirements for the reliability modeling of digital I&C systems", although it should be noted that none of the methodologies were able to meet all eleven acceptance criteria. [11,12]. These methods were DFM, and an extension of Markov Modelling, known as the Cell-to-Cell Mapping Technique (CCMT). Overall, it was stated in NUREG/CR-6901 that "…DFM ranks as the most preferable methodology"[8]. Research projects were carried out using DFM and Markov CCMT, for modelling a generic digital feedwater controller for an NPP, with results published in the literature [11,12,15].

1.2.3.2 Additional DFM Literature Review

On top of the NUREG reports cited in this section, DFM has been used and positively reviewed in scientific literature for its ability to model the hardware/software/firmware interactions in digital control systems in the nuclear field [13,15–18]. DFM has also seen use in the modelling and analysis of accident management [19,20], human factors [21], and the analysis of advanced reactors [22]. Outside

the nuclear field, DFM has been used for general modelling analysis of control systems/process control systems [23–26], and for the modelling of control systems software by the National Aeronautics and Space Administration (NASA) [27,28].

1.2.3.3 Final Discussion on the Selection of DFM

The information in sub-sections 1.2.2 and 1.2.3 provide the rationale for selecting DFM for use in this research program, as opposed to the other available dynamic methodologies. However, all of the references discussed in those sub-sections considered only software-based digital I&C systems, and did not consider programmable hardware technology, such as FPGAs. Therefore, expanding on the previous research on the DFM analysis of digital I&C systems to model/analyze FPGA-based systems represents a new and unique avenue of research.

1.3 Novelty and Contribution of this Thesis

There were two overall sub-topics in this thesis that made a large contribution to technical and engineering knowledge; the FPGA FMEA and Taxonomy, as well as the DFM modelling of FPGA-based systems. The specifics of both of these sub-topics are discussed in this sub-section.

1.) FPGA FMEA and Taxonomy

- Compiled a comprehensive list of FPGA failure modes data (failure modes, effects, causes, etc).
- Categorized these first by stage in the lifecycle (“Design” and “Operation”), then by “Cause”
- “Failure Sets” grouped based on similar causes and effects, in order to provide detailed information on avoidance and/or mitigation methods
- FMEA data used to construct a plug-in to interface with the OECD-NEA digital I&C failure modes taxonomy, creating the FPGA taxonomy and fulfilling an important topic of future work as stated by an international working group (WGRISK)

2.) DFM Modelling of FPGA-based systems

- DFM not previously applied to analyze FPGA-based systems (only generic SW-based systems)
- Confirmed the usefulness and accuracy of DFM for modelling FPGA-based systems using an industry standard simulator
- First detailed comparison of FTA/DFM model of FPGA-based system:
    - At the chip/board level
    - Failure mode “fault injection”
    - In-depth discussion on several reasons for the differences, including both theoretical (algorithms), to practical (model construction/analysis)
- Identified several advantages of DFM over general static analysis methods, with special consideration given to the advantages of DFM over FTA and simulation.
- Determined several potential avenues of future research regarding the reliability analysis of digital I&C/safety systems

1.4 Chapter Summary

In this chapter, the overall motivation for the research program undertaken as part of this thesis is presented and discussed. FPGAs are a relatively new technology in the nuclear domain, and documentation from the IAEA states that the use of FPGAs in various I&C systems and nuclear plants will increase greatly in the future. However, the IAEA documents also states that there is not a great deal of available standards and operating experience in the nuclear field, so any additional research workinto the reliability and safety of FPGA-based systems is of use to the international community. With regards to the reliability analysis methodology, information in the literature has stated that traditional, static methods may not fully capture the unique characteristics of digital systems, such as FPGAs, leading to the selection of a dynamic methodology for this research program. Furthermore, the Dynamic Flowgraph Methodology was the dynamic methodology of choice for this research program, based on the results of research performed in NRC NUREG reports, and from a survey of technical publications. The results from this research program will provide additional data and information that can be used to improve upon the design, modelling and review of FPGA-based systems in NPPs. The novelty and

contribution of this research work is also presented, and a brief outline of this thesis document is included at the beginning of this section.

2 Background

This chapter presents the background information relevant to the overall research work. It includes both the background information for FPGAs and FPGA-based systems, as well as the background information on the reliability analysis methods used during this thesis (FTA and DFM). Sub-section 2.1 provides the background information on FPGAs, and sub-section 2.2 presents a detailed literature review of FPGA-based systems in the nuclear field, which was subsequently published in the literature [6]. Subs-section 2.3 describes the reliability analysis methods, FTA and DFM. Sub-section 2.4 provides a summary of the information discussed in this chapter.

2.1 FPGA Background

As FPGAs are the focal point of this research work, so are more in-depth description of FPGAs is important and necessary, with this chapter providing that information. This sub-section provides a detailed description of FPGAs, how they relate to other electronic logic technologies, FPGA architecture and technology, advantages and disadvantages of FPGAs, as well as providing a brief discussion of FPGAs in other industries and related technologies.

2.1.1 FPGA Descriptions

FPGAs were first created in 1985 by Ross Freeman, a co-founder of the Xilinx Company, which is currently a top supplier of FPGAs [29]. The main suppliers of FPGAs in modern times are the aforementioned Xilinx, and Altera (now owned by Intel) [30]. Other chip manufacturers include Lattice [31] and Microsemi [32]. Several other companies provide development boards using Xilinx and Altera chips, such as Opal Kelly [33], Digilent [34] and Terasic [35].

FPGAs are a form of large scale integrated circuits programmed to perform digital logic functions. The internal hardware of the FPGA is “programmed” (configured) by the user after the chip is manufactured, to perform it’s desired logic function(s), as FPGA chips contain no logic after they are manufactured [1,3,4]. The end user will program logic functions onto the blank FPGA chip using Hardware Description

Languages (HDL). FPGAs can be reprogrammable or be One-Time Programmable (OTP), depending on the type of technology used. Like microprocessors (including Programmable Logic Controllers (PLCs)), FPGAs are considered to be a form of Programmable Digital Device (PDD) [36]. Unlike a microprocessor, the FPGA logic is synthesized directly into the hardware of the FPGA chip, so the FPGA does not run any software or operating system, it is purely a hardware implementation at that stage. The capacity of FPGAs has increased many times since their inception, and the current models are capable of carrying out parallel executions with very fast response times.

2.1.2 FPGAs in the Electronic Logic Family

The way that FPGAs fit into the overall family of electronic hardware is shown in Figure 1 [1,4], with some important definitions given afterwards.

HDL Programmed Devices (HPD)

An HDL Programmed Device (HPD) is defined as an "Integrated circuit configured (for NPP I&C systems) with Hardware Description Languages and related software tools" [37]. They contain arrays of logic elements that are connected by the end user to configure the device to perform the needed logic function [1].

Hardware Description Language (HDL):

"Language Used to formally describe the functions and/or structure of an electronic component for documentation, simulation or synthesis**"** [37]**.**

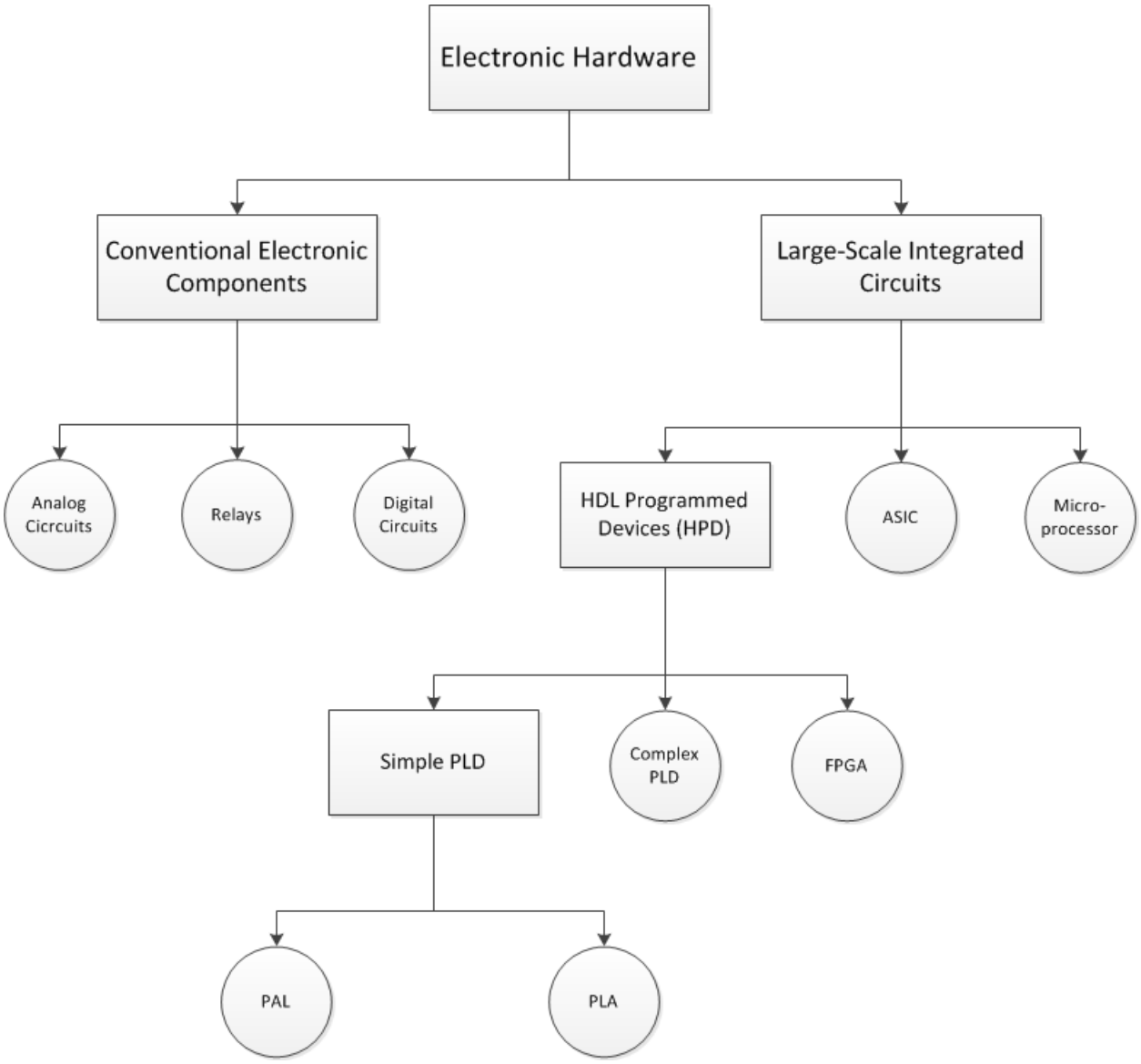


Figure 1: Electronic Logic Family Block Diagram

Application Specific Integrated Circuit (ASIC):

An Application Specific Integrated Circuit (ASIC) is defined as an "Integrated Circuit designed for specific applications" [37]. Unlike FPGAs, ASICs are not configurable/reconfigurable after they are manufactured, as their functionality is custom designed/fabricated by the manufacturer at the time of construction [1].

It should be noted that there is some disagreement as to whether an ASIC should be considered as an HPD. Documentation from the IAEA [1] and Electric Power Research Institute (EPRI) [4] does not consider ASICs as HPDs. However, documentation from the Multinational Design Evaluation Program (MDEP) does list ASICs as HPDs [38]. Therefore, there is still some discussion among the international community as to the exact categorization of the different electronic logic technologies.

Field Programmable Gate Array (FPGA):

An integrated circuit that can be programmed in the field by the instrumentation and control (I&C) manufacturer. It includes programmable logic blocks (combinatorial and sequential), programmable interconnections between them and programmable blocks for inputs and/or outputs. The function is then defined by the I&C designer, not by the circuit manufacturer [1,37]**.**

Programmable Logic Device (PLD):

A Programmable Logic Device (PLD) is defined as an "Integrated circuit that consists of logic elements with an interconnection pattern, parts of which are user programmable" [37]. HPDs began as simple Programmable Logic Devices (PLDs), with includes Programmable Logic Arrays (PLA) and Programmable Array Logic (PAL). Complex Programmable Logic Devices (CPLDs) are descended from PALs, and are basically combinations of multiple PALs onto a single chip with configurable interconnections [1]. FPGAs are not considered to be PLDs, as FPGAs are more complex, more powerful devices, however the exact determination between PLD and FPGA is not entirely defined [37].

In general, FPGAs differ in terms of the routing methods and their logic blocks. A brief comparison is shown in Table 2 [1,4].

Programmable array logic (PAL):

"A type of simple programmable logic device that consists of a programmable AND-plane followed by a fixed OR-plane" [1,4].

Programmable logic array (PLA):

"A type of simple programmable logic device that consists of two levels of logic, an AND-plane and an OR-plane, both of which are programmable" [1,4].

Table 2: A Comparison of Important Technology Attributes of FPGAs and CPLDs

Technology Attribute	FPGA	CPLD
Configurable Logic Block	Gate Array	Logic Array
Density	>500,000 gates	<500,000 gates
Speed	Design and Application Dependent	Fast and Predictable
Interconnect	Routing	Crossbar
Power Consumption	Low-Medium	High

The important differences between CPLD include [1,4]:

1.) CPLD logic cells have a larger granularity that FPGAs, so it takes less logic cells to implement a certain function on a CPLD than on an FPGA

2.) Intellectual Property (IP) cores can be embedded into FPGAs for the purpose of performing complex logic functions, however this is generally not possible with CPLDs.

3.) CPLDs possess a larger logic-to-interconnect ratio than FPGAs, so they are typically faster for very simple applications. Alternatively, this allows FPGAs to have greater flexibility and more design capacity.

4.) CPLDs have a continuous interconnecting structure, resulting in high performance and functional predictability, specifically considering signal propagation delays. On the other hand, FPGAs have a segmented interconnect structure, where the number of segments needed to route the signals is established by the software tools. Due to this, the exact timing of the signal propagation in FPGAs cannot be known until the full design is placed and routed.

5.) The "routing" interconnections used by FPGAs permits far more signal paths to be constructed, than is allowed by the crossbar routing of CPLDs.

2.1.3 FPGA Architecture

FPGA Logic Gate Composition

FPGAs, like many other forms of digital logic, are based on Complementary Metal Oxide Semiconductor (CMOS) technology. The building blocks of CMOS technology are referred to as Metal Oxide Semiconductor Field Effect Transistors (MOSFETS) [3]. CMOS technology makes use of two types of transistors; “P-Type” and “N-Type”. The “P-Type” transistors allow current to pass between the drain and source when the gate electrode voltage is negative. On the other hand, the “N-Type” transistors allow current to pass between the drain and source when the gate electrode voltage is positive. CMOS technology uses complementary pairs of these two types of transistors to form the logic gates, to perform the digital logic functions (such as “OR”, “AND”, etc.) [3].

During operation, if one of the transistors in the complementary pair is “On”, the other transistor will be “Off”. Due to this, the electric current, and therefore the power usage, is only needed for a short amount of time, when the logic gates change their states. The reduced power usage is an advantage of CMOS technology.

When considering CMOS technology, the “Complementary” term denotes the complementary pairs of those two transistor types. A visual representation of the basic components of a MOSFET transistor is seen in Figure 2 [3].

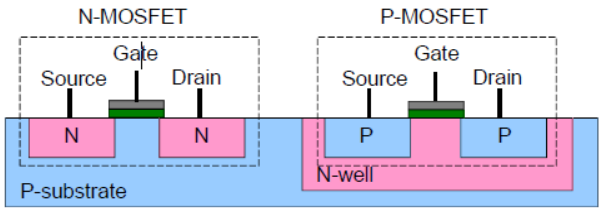


Figure 2: Diagram of “P-Type” and “N-Type” Transistors

Generic FPGA Chip Architecture

Regardless of the chip make, model or manufacturer, FPGAs are composed of several basic parts: I/O connections mounted on the edges, programmable/configurable logic components (logic blocks or CLB),

and the wiring between the logic blocks that is routed through switchboxes (sometimes referred to as a programmable interconnect) [39]. The CLBs are the logic units of the FPGAs, and are sometimes referred to as Logic Array Blocks (LABs). These are comprised of flip-flops and Look-Up Tables (LUT), but the exact packaging of these components can vary between FPGA series and manufacturers. In many cases, FPGAs will include additional application data memory [1,4]. More advanced FPGAs may also have additional components, but the ones listed above are common to all FPGAs. An outline of a basic FPGA is shown in Figure 3 and Figure 4. A more detailed description of the main components of the FPGA are given below [1,3,4].

Figure 3 is a simplified representation of an FPGA chip, and it outlines the placement of the three main components that were previously discussed [39]. The logic blocks are represented by squares, the I/O blocks are shown by the circles, and the lines connecting the blocks denote the programmable interconnects. The exact set-up can vary between makes and models of FPGAs. Figure 4 expands on the basic FPGA architecture, including application data memory, and highlighting the inner components of the CLBs [4].

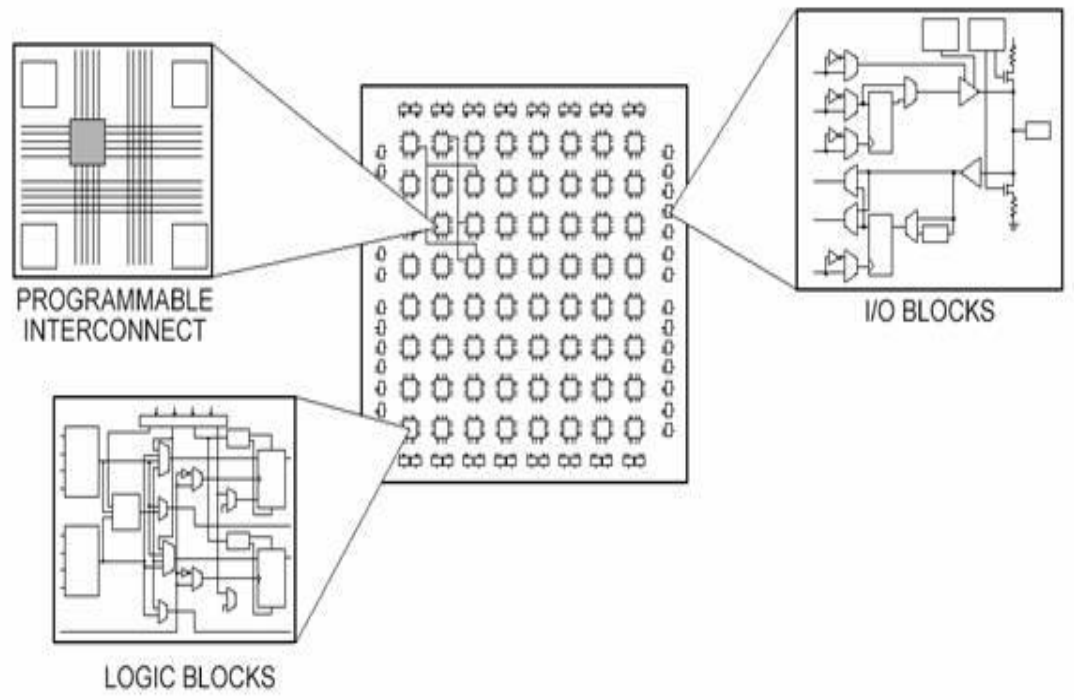


Figure 3: Outline of an FPGA chip showing the three main components

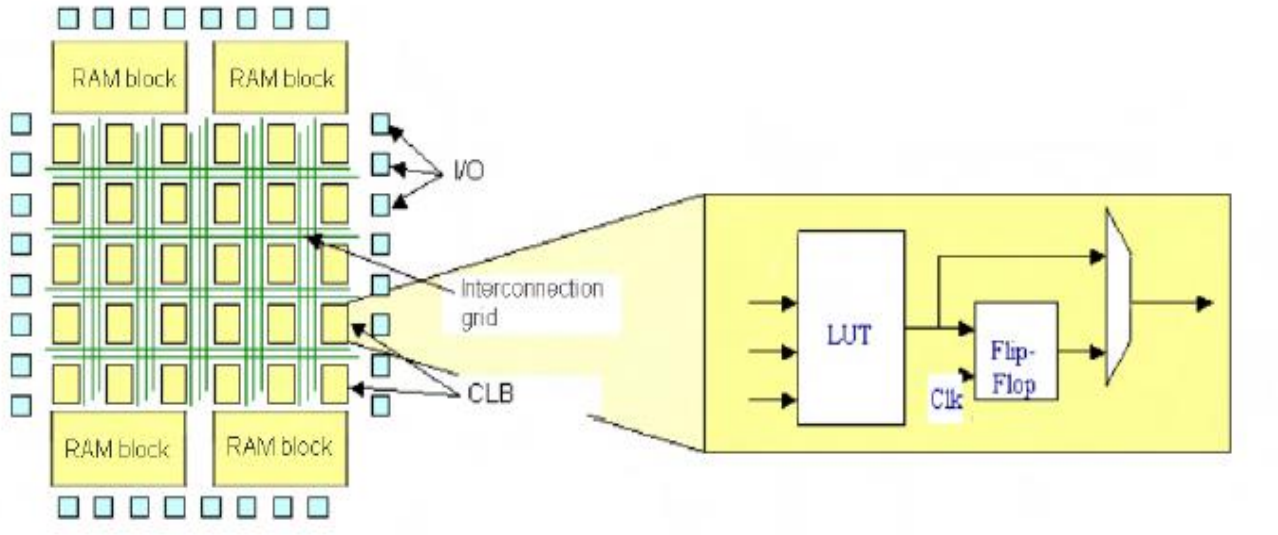


Figure 4: Basic FPGA Architecture with Block RAM and CLB Close-Up

Configurable Logic Blocks (CLB):

The CLBs are the FPGA component that will perform the digital logic functions. Based on the FPGA configuration (programming), the CLBs will configured to perform their specified function ("AND", "OR", etc. Therefore, each CLB will feature X Boolean inputs with Y Boolean outputs, and is configured to implement an X-to-Y Boolean operation. Classically, this implementation could be performed using logic gates, or through LUTs [4].

As seen in Figure 4, a generic logic block could be constructed using an LUT, a Flip-Flop/Register for data storage, and a Multiplexer (MUX), that can bypass the Flip-Flop/Register if desired/ Generally though, the CLB outputs are synchronized through that Flip-Flop/Register, to maintain a synchronous design in the FPGA. In order to perform more complex digital logic functions, multiple CLBs can be strung together. A more detailed view of an LUT-based CLB is seen in Figure 5, and an expanded view of the MUX-based CLB is given in Figure 6 [4].

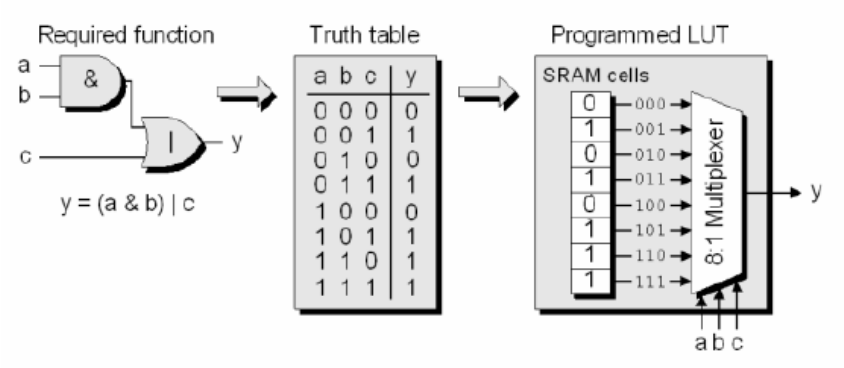


Figure 5: CLB Implemented with an LUT

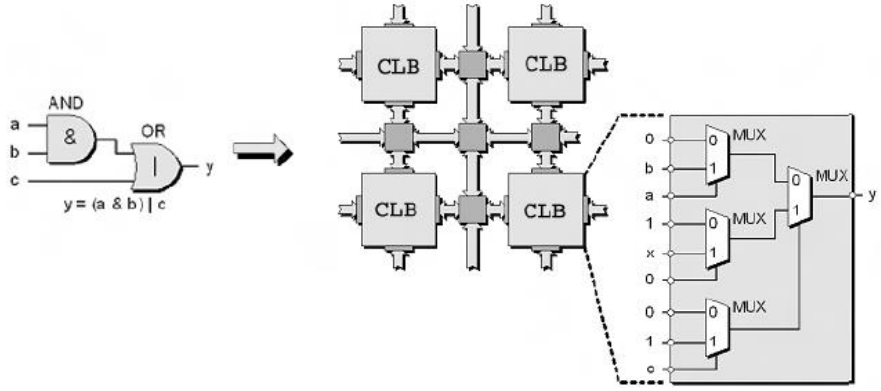


Figure 6: CLB Implemented with MUXs

Figure 5 shows an LUT-based implementation of a CLB. LUTs are composed of a MUX and a small amount of RAM, and can be used to implement basic logic functions. The example shown in Figure 5 represents the logic function "*y = (a AND b) OR c"*, denoted as *y = (a & b)* | *c* in the literature [4]. Here, *y* denotes the single output, whereas *a*, *b*¸ and *c* represent three separate inputs. The truth table is implemented in the FPGA using 8-bit RAM and an 8-to-1 MUX.

Conversely, Figure 6 shows the Mux-based implementation of the CLB. It would function in a similar way as a tree of simple, 2-to-1 MUXs, where each MUX performs one logic equation operand that has been configured into the CLB. Using the same logic function example ("*y = (a AND b) OR c"*,), it would be implemented in the FPGA using a set of 4 MUXs, with the final configuration being determined by the FPGA design/synthesis toolset.

Programmable Interconnects (Internal Connection Grid):

The interconnects are the internal wiring of the FPGA, and consist of sets of vertical and horizontal wires. These wires are originally unconnected, but contacts can be made at the intersection of the wires, with those contacts being controlled by the switchboxes. The interconnect contacts are used to connect the CLBs to each other, as well as the CLBs to the FPGA chip I/O.

Input/Output (I/O) Connectors (Blocks):

The I/O connections are used to propagate signals into, and out of, the FPGA chip. They are electrical boundaries between the higher voltages/currents used by external electrical/electronic components that the FPGA is connected to, and the low voltage/current signals used inside the FPGA. The I/O ports connect to CLBs inside the FPGA chip, and can be configured to be either inputs or outputs. In the case of some more advanced FPGAs, the I/O connectors are also able to act as analog-to-digital converters (ADCs).

Application Data Memory:

While this is not strictly a base component of FPGAs, it has become very common in FPGAs, especially in modern times. This additional memory is used to make up for the small amount of memory available in the CLBs. Typically, they involve blocks of SRAM (often referred to as “Block RAM”), but could also be blocks of Flash memory, if greater radiation tolerance and/or configuration data retention is required.

2.1.4 FPGA Technologies

There are three main forms of FPGA technologies; Static Random Access Memories (SRAM), Flash, and Antifuse [1,3,4]. By these technologies, it is meant how the FPGA stores logic on the chip, and how the chip retains its configuration logic (programming). A fourth type can be considered, if Electically Erasable

Programmable Read Only Memory (EEPROM) is considered a distinct type, however it is similar to Flash memory, which is a more modern technology. A discussion of these technologies is given below [1,3,4]. It should be noted that regardless of the choice of FPGA technology, the logic gates are all based on CMOS technology.

SRAM:

SRAM is the most common technology used in commercial FPGAs, and is the same as the working memory found in computers. SRAM FPGAs are all based on Complementary Metal Oxide Semiconductor (CMOS) technology. Typically, SRAM technology allows for the most powerful/highest performing FPGAs (fastest and largest logic programs). They are reprogrammable, and are generally the fastest to program/reprogram. SRAM FPGAs work by keeping the configuration (program) as the state of groups of cross-coupled transistors. Each transistor group forms a single memory bit, and decides if a switch/interconnect is closed or open. When power is supplied to the FPGA, the transistors will hold the system in a state of either “0” or “1”. Conversely, if power is lost, the state of the transistor will also be lost.

As SRAM FPGAs will only retain their programming when powered on, and if they lose power, then the FPGA must be reconfigured. In some cases, an external configuration system will be used to reconfigure the FPGA on start-up. Therefore, SRAM FPGAs are said to have “volatile” memory. This also makes SRAM FPGAs the most susceptible to power glitches, and to suffer memory upsets or configuration errors from radiation strikes known as Single Event Effects (SEE). Therefore, extra care must be taken when using SRAM FPGAs in operating environments where radiation interactions are expected.

Flash and EEPROM:

Flash memory is considered to be a more modern version of EEPROM, and has often replaced that technology in FPGAs. Flash FPGAs work by using “floating gate” transistor, where that gate determines the transistor behaviour. Flash FPGAs require only one transistor, as opposed to the six transistors used in an SRAM FPGA. As the floating gate is electrically insulated, a Flash FPGA will retain its configuration when powered off, and as such is said to be “non-volatile”. Flash FPGAs are reprogrammable, but will only handle a certain number of Program/Erase (P/E) cycles, usually on the orders of 100,000-1,000,000. Similarly, the Flash FPGA will retain its configuration for a limited time, usually on the order of 10-20 years.

The “Flash” term in this context is due to the erasure of large block of date at once during the P/E cycles. The main difference between Flash and EEPROM FPGAs is the way the configuration memory is written. In Flash FPGAs, large blocks of memory can be written at once, while in the case of EEPROM, the memory bits must be written individually, making it in slower process. In general, Flash and EEPROM FPGAs will be slower and hold less data than SRAM FPGAs. However, Flash FPGAs have also been seen to have a lower sensitivity to SEEs than SRAM FPGAs.

Antifuse:

Antifuse FPGAs are configured by sending a high current through the wires of the interconnection grid, to create a contact between those two wires. Since the connections (between CLBs and I/Os) are created by links made heated conduction, which is the opposite method of how a fuse works, these FPGAs earned the name of “antifuse”. Like Flash FPGAs, antifuse FPGAs are “non-volatile”, however unlike SRAM and Flash, antifuse FPGAs cannot be reprogrammed (OTP). In order to change the logic on an antifuse FPGA, the entire chip would have to be replaced. Antifuse FPGAs are less common than other SRAM or Flash FPGAs. They tend to be the lowest in terms of performance (speed and chip density), however they also have the highest resistance to SEEs, and from tampering with the system, as the FPGA logic cannot be changed.

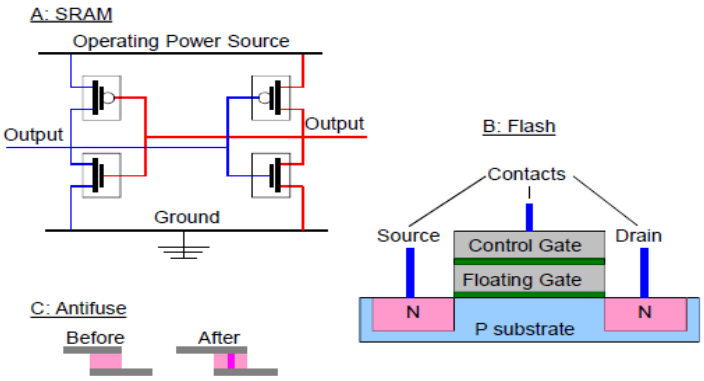


Figure 7: FPGA Configuration Storage Technologies

A visual overview of the FPGA technologies is given in Figure 7, with the structure of the connecting wires being seen in Figure 8 [3].

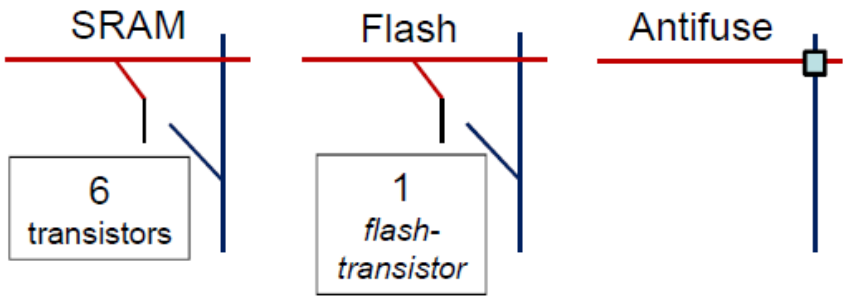


Figure 8: FPGA Switchbox/Interconnect Structure for Different Technologies

A simple comparison of the three main FPGA technologies is given in Table 3 [4]

Table 3: Comparison of FPGA Technologies

Attribute	Antifuse	Flash	EEPROM	SRAM
Speed	Best	Worst	Medium	Worst
Power	Near Best	Best	Worst	Varies
Density	Second	Best	Worst	Medium
SEE Tolerance	Best	Medium	Medium	Worst
Reprogrammable	No	Yes	Yes	Yes

2.1.5 FPGA Programming

As mentioned previously, FPGAs utilize a form of low-level programming language known as a Hardware Description Language (HDL). Popular versions include Verilog (IEE standard 1364) [40], VHDL (Very High Speed Integrated Circuit HDL, IEEE standard 1076) [41], and SystemVerilog (IEEE standard 1800) [42]. HDLs are used to provide a description of a hardware system, for the purpose of system programming, modelling and analysis, as their name implies. These languages use a Register Transfer Level (RTL) design abstraction, in order to model digital circuits, such as those used in FPGAs [1]. This design abstraction is independent of the physical hardware implementation, and shows the necessary logic operations and signal propagation for the desired application, such as a circuit on an FPGA. The gate-level description (also referred to as a “netlist”), for a certain FPGA logic model is created by the logic synthesis of the RTL design abstraction. The output of that logic synthesis is transferred to a physical implementation for the actual FPGA chip through the synthesis tools, using a method called “Place and Route (PAR)”. Once this

configuration has been loaded onto the desired FPGA chip, the FPGA is said to be configured, or "programmed" [1]. The definitions of several important terms are given below:

Register Transfer Level (RTL):

"Synchronous parallel model of an electronic circuit, describing its behaviour by means of signals processed according to combinatorial logic and transferred between registers and clock pulses. The RTL model is typically written in HDL or generated out of HDL source code**"** [37]**.**

Netlist (Gate Level):

"Description of an electronic component in terms of interconnections between its terminal elements" [37].

Place and Route (PAR):

"The step in integrated circuit or printed circuit board design that determines the physical locations of components, circuitry and logic elements, and the wiring paths required to connect the components" [1].

Bitstream:

"A contiguous sequence of bits (binary digits), representing a stream of data, serially transmitted continuously over a communications path. It is frequently used to describe the configuration data to be loaded into an FPGA" [1].

When working in one of the HDL Languages, there are several software packages available to program the FPGA. Xilinx provides the ISE line of products, including the free ISE Webpack (later replaced by Vivado), which includes the simulator program ISim [43]. Xilinx also supplies more expensive packages, such as ISE Design Suite, needed to program higher grade FPGA chips, and includes logic analyzers. Altera provides the Quartus II set of software packages, which include the free Quartus II Web Edition (now Quartus Prime), along with the ModelSim simulator and the SignalTap logic analyzer [44].

Traditionally, many people did not possess the skills and expertise needed to program the FPGA using HDLs due to the multiple steps required for this process. In more recent years, FPGA suppliers have developed more user friendly programming software that is available to users. Graphical programing languages, such as LabVIEW be used to program National Instruments (NI) FPGA-based devices directly, avoiding the need to master an HDL language [39]. Other program languages such as MATLAB/SIMULINK [45] and Python [46] have modules to convert their respective code into synthesizable HDL code.

2.1.5.1 Comparison of HDLs and Common Programming Languages

There are several notable differences between HDLs and common high level programming languages (like C, C++, etc.) that were discussed in the literature. These include [1]:

- Software source code is written in a high level language, compiled and generates binary code which is executed sequentially on the selected microprocessor. HDL code is also considered a high-level language, however in that case the RTL abstraction is translated through the synthesis and PAR methods into a bitstream, to physically configure the FPGA chip. This results in a parallel execution on the FPGA, as opposed to the sequential execution of traditional software-based systems

- The HDL code for the FPGA is only used to configure the FPGA hardware, and is not actually executed like on a microprocessor

- FPGAs do not run any Operating System (OS), which introduces an additional level of complexity. As the FPGA implementation is purely hardware, no OS is used

- If a simple FPGA configuration is used (i.e. no soft-processor cores are implemented), software-based run-time problems such as task scheduling and interruptions/exceptions are not seen in HDL programmed devices

Overall, one of the main differences is that the HDL implementation will be executed in parallel, while traditional microprocessor software executes sequentially. It should be noted that there are also notable similarities between HDLs and traditional programming languages, including [1]:

- The use of formal methods for logic verification

- Logic and/or syntax errors in software and HDL can manifest into implementation errors when the logic is downloaded onto the respective device

- Overall development process for software-based systems and FPGAs are treated the same by some regulators, such as the United States Nuclear Regulatory Commission (USNRC) [7], as is the current recommendation from other international organizations [2]

2.1.5.2 FPGA Mathematics

Mathematical calculations performed by FPGAs are most frequently performed using the Fixed-Point Data Type (FXP), as the calculations are performed faster, with less chip resources than calculations with equivalent floating point numbers [47]. A fixed point number is composed of two parts: the integer (containing the sign if needed) and the fraction. The integer portion represents the whole number and sign (the part of the number before the decimal point), while the fraction portion represents the decimal (section of the number after the decimal point) [47]. When using fixed point math, the decimal point is often referred to as the radix point. The term “fixed-point” means that the number has a fixed number of digits after the decimal point, as opposed to floating-point numbers, where the decimal point can be moved (“floats”). To synthesize a fixed point number, one must specify two parameters: a “Word Length” and an “Integer Length”. The word length gives the total number of bits for the number, while the integer length gives the number of bits for the integer portion. Any leftover bits then become the bits used by the fraction (decimal) [47]. The most common word lengths are 16 and 32 bit, however there other sizes can be used, and the sizes are limited by the FPGA development tool.

When using the fixed point data type, there is always the possibility for some loss of precision with the decimal portion of the number. If the calculated value does not land exactly on certain numbers, then

there will be some shifting of the decimal numbers. This due to the resolution of the fixed point number, as it will only be able to exactly represent certain values within a certain range. If the floating point data is above/below the range that can be represented by the FXP numbers, then there is an "Overflow/Underflow" Error. If the floating point data is inside the FXP range, but does not exactly match the FXP values, then rounding must occur, and one has a "Resolution Error". This is visualized in Figure 9, where grey bars denote the values that FXP numbers can take, and the orange circles represent the actual floating point data.

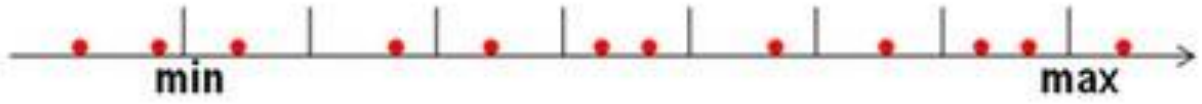


Figure 9: Comparison of FXP and Floating Point Representation

The methods for dealing with these problems are discussed below, as are good FXP FPGA design practices (general guidelines) for avoiding these issues.

Overflow/Underflow [47]**:**

There are two main ways for dealing with the overflow/underflow conditions: wrap and saturation. In the "wrap" mode, the maximum value is wrapped around to the minimum value once the maximum is exceeded, and then minimum value is wrapped around to the maximum value when the value drops below the minimum range. When considering the "Saturation" mode, floating point values outside of the fixed point range will be capped at the minimum/maximum values, regardless of how far from the maximum/minimum values they drift

Rounding (Resolution Issue) [47]**:**

There are three separate methods that can be applied to round off the floating point numbers to match the fixed point numbers: truncate, round half up, and round half even. With truncate, the left neighbouring gird (digit), will always be selected, as this method would eliminate the Least Significant

Bits (LSB). Round half up will select the closest neighbour digit, but will always round up if it is exactly between the two numbers. Round half even will also select the closest neighbour digit, but can round up or down (depending on which number has an LSB of zero) when it is exactly between two numbers. Round half even is considered to be the most accurate rounding mode. A visual representation of these methods is seen in Figure 10 [47].

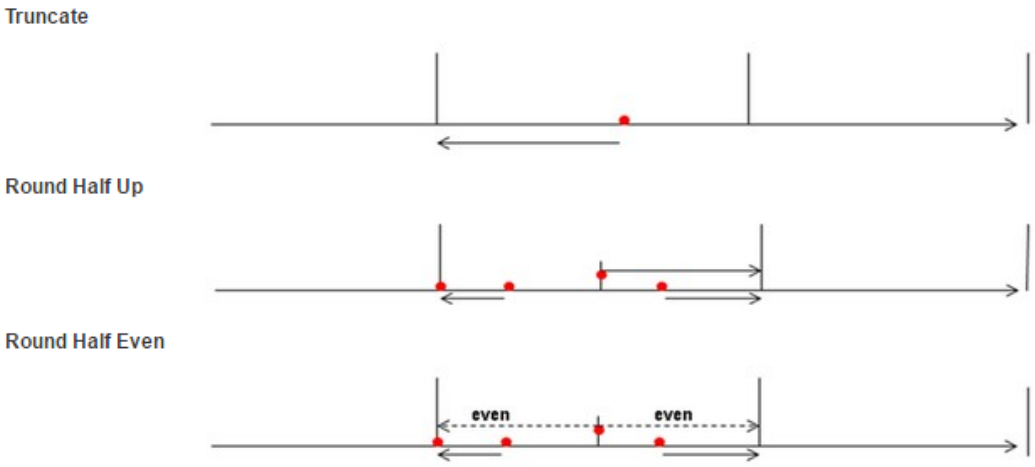


Figure 10: Methods for Solving FXP Round OFF/Resolution Errors

FXP FPGA Design Practices

As previously discussed, the use of FXP numbers/mathematics has the potential to introduce certain errors into the FPGA calculations. While methods exist to mitigate these errors to a certain extent, all of those mitigation methods will still lead to some loss in accuracy. In order to use FXP representations/mathematics with the greatest accuracy, one should pre-calculate the ranges of all input values [48]. If the total range of the floating point numbers is known, then an appropriate FXP range can be determined, to eliminate overflow. With regards to the issue of FXP resolution, it can be mitigated to an extent. An acceptable loss of resolution should be determined in the specification stage, which will allow for the proper integer and word lengths to be selected, minimizing the round off/resolution errors [48].

In more recent times, advanced packages for FXP [49] and floating point numbers [46] have been developed, and have been incorporated into the most recent VHDL standard [41]. However, not all FPGA synthesis/development tools fully support these packages, so additional care must be taken when applying those advanced packages, especially when considering a safety-critical system.

2.1.5.3 FPGA Programming Process

The overall programming process for FPGAs can be broken down into four main categories; “Component Requirement Specification”, “Preliminary Design”, “Design”, and “Implementation”, with verification occurring at each one of those stages [1,4,37,38]. The final two categorises, “System Integration” and “System Validation” occur after the FPGA has been programmed/configured. These consider the FPGA aspects of system integration and validation, after the total system has been assembled. This is often represented as a “V-Shaped” programming model, as seen in Figure 11 [1,4,37,38]. The first four stages in that process are discussed in this sub-section. The final two stages represent the FPGA-specific aspects of their specific phases. Those overall phases are discussed in more detail in sub-section 2.1.6.

Component Requirements Specification:

Define and state all requirements applicable to the final, configured FPGA. Those requirements generally stem from the overall I&C system architecture [1,4].

Preliminary Design:

Determine the important design choices such as the amount of sequential vs combinatorial logic, intended modules, IP cores and pre-developed code, application-specific code, fault tolerant design techniques, library functions, and the intended device/device family [1,4].

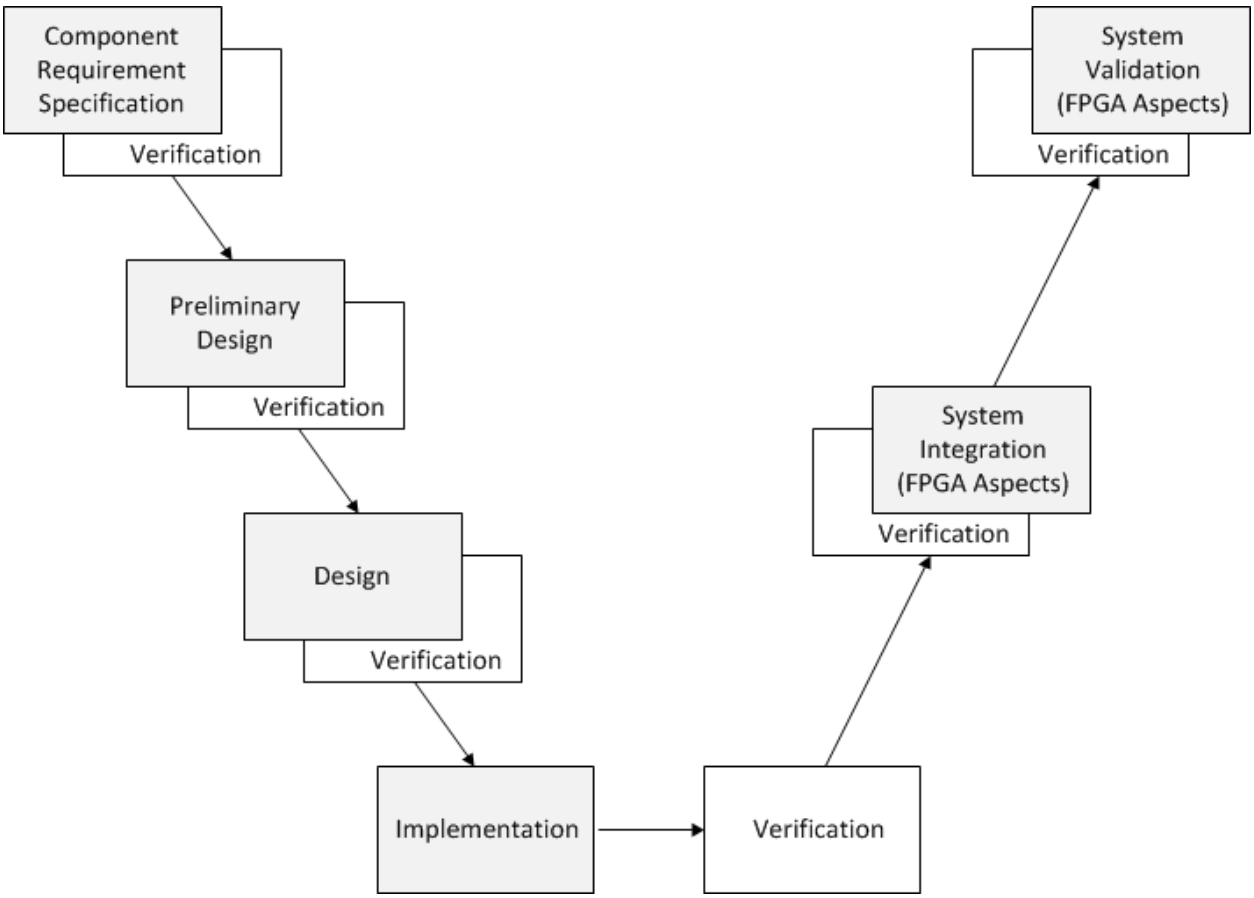


Figure 11: FPGA Programming Process (V-Shape)

Design:

Develop the detailed description of the intended logic functions needed to be performed by the FPGA. It is often device independent, and the design is generally done using one form of HDL. The HDL code in turn describes the RTL, which can then undergo simulation and/or formal verification to test for logical/behaviour errors [1,4].

Implementation:

This is the stage where the actual FPGA chip is configured. It generally considers two steps, synthesis, and then PAR. The goal of the synthesis phase is to convert the (circuit independent) RTL into a description that can be applied to the desired FPGA. The resulting circuit-dependent description

becomes the netlist. Additional simulations and formal verification can be performed at this point. Additionally, timing simulations are performed more accurately, due to the timing files generated during synthesis.

After the synthesis step, the PAR occurs. The design tool will compute the most efficient configuration for the FPGA. Additional timing files are generated, to allow for more accurate timing simulations than in the synthesis phase, as the PAR timing files consider the actual hardware routing. At the end of the "Implementation" stage the bitstream is generated, to program/configure the FPGA chip [1,4]. A block diagram of the total "Implementation" step is seen in Figure 12 [4].

.

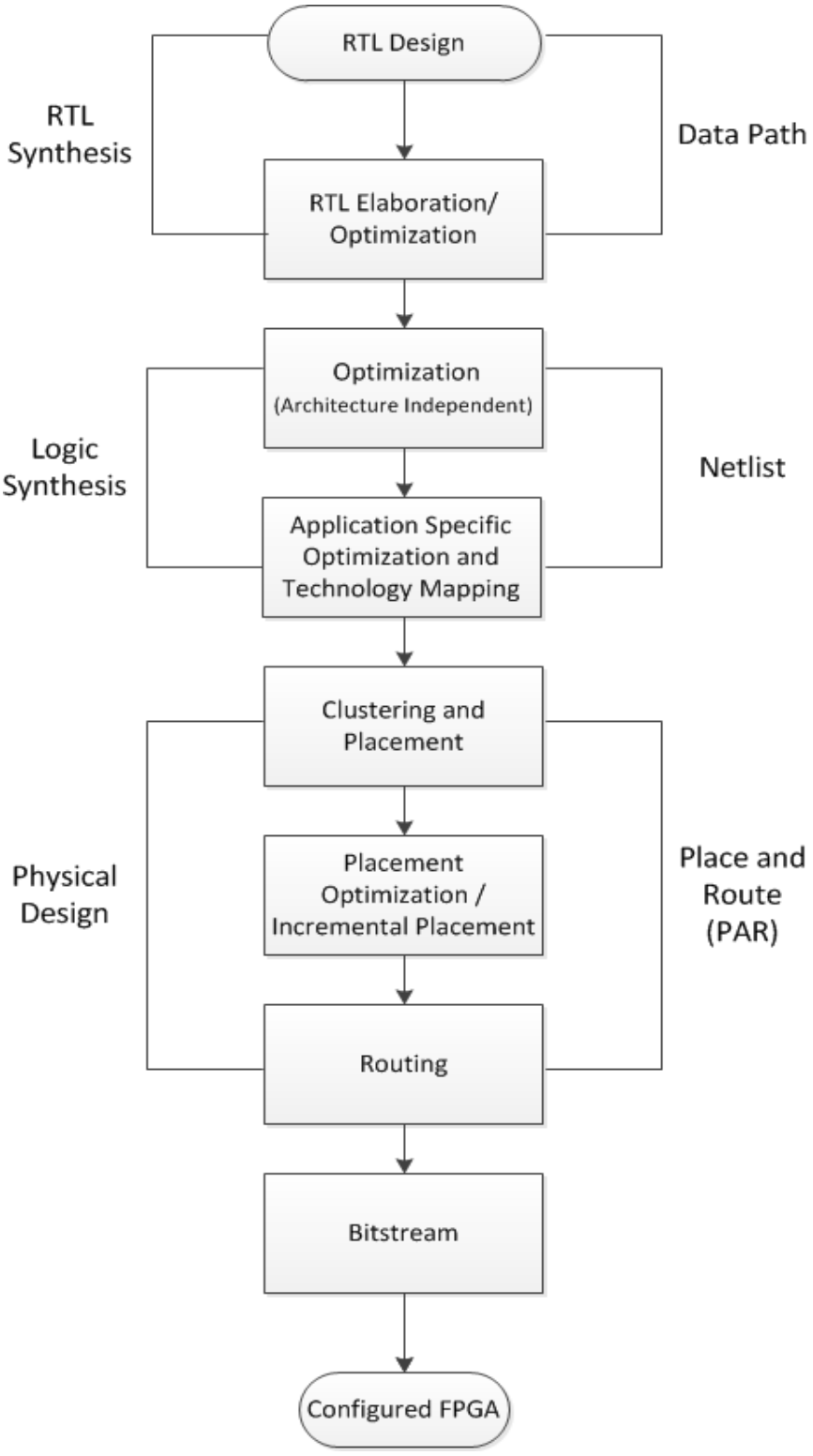


Figure 12: Block Diagram of the "Implementation" Stage of FPGA-Based Systems Programming

2.1.6 FPGA-Based I&C System Lifecycle

This section focuses on the lifecyle of the total FPGA-based system, not just the FPGA chip itself. It would include all the necessary processes, documentation and organization needed to develop an FPGA-based system. The system could use either a pre-developed platform (components combined into a system architecture and configured for implementing certain tasks), or be designed completely from scratch. A block diagram of the FPGA-based I&C system life cycle is seen in Figure 13 [1]. Several important stages in that lifecycle are discussed proceeding that figure.

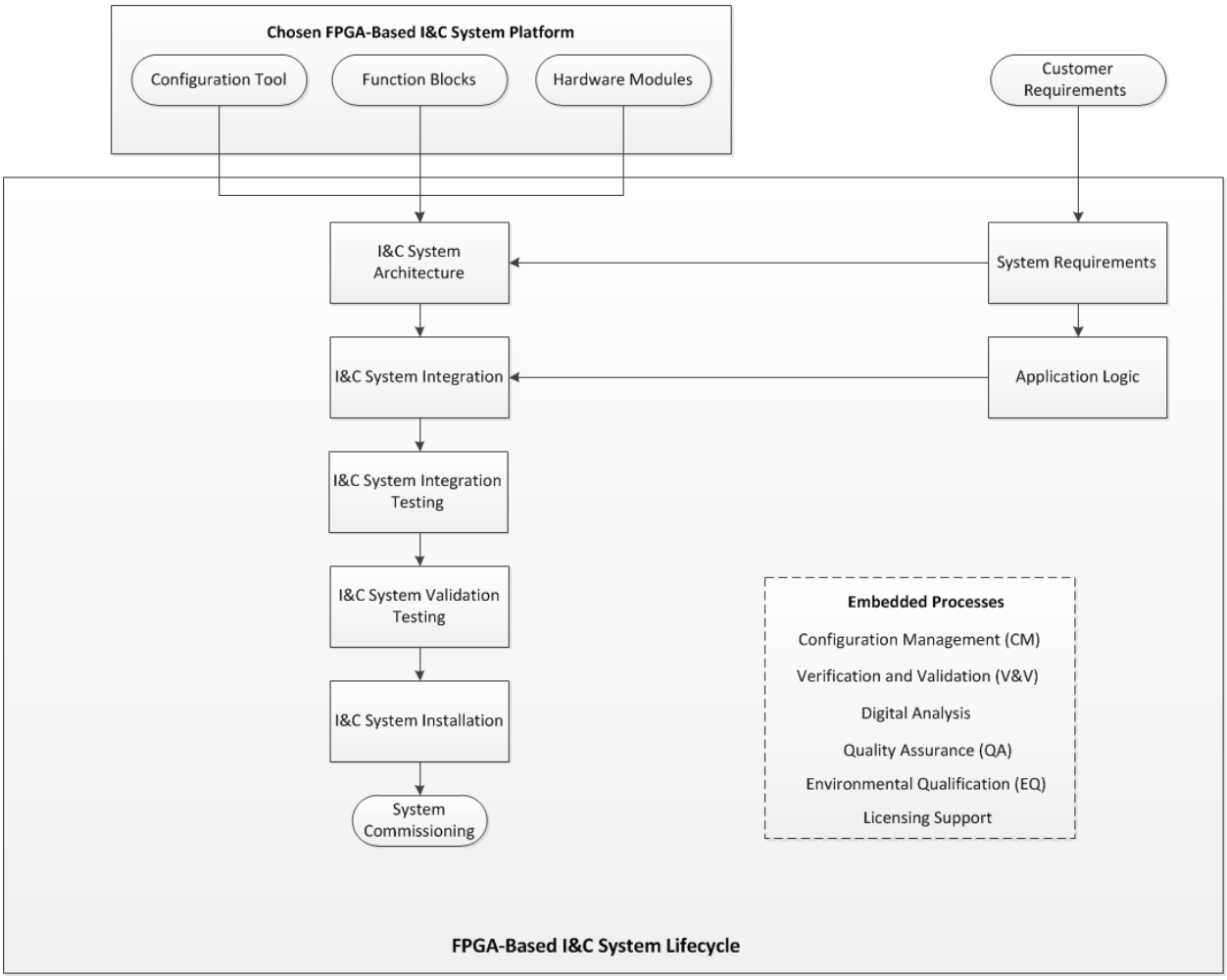


Figure 13: Overall Lifecycle of FPGA-Based NPP I&C Systems

System Requirements Specification:

In this step, important requirements for the I&C system are specified. These include the system boundaries and interfaces with other systems and human operators, the environmental operating conditions, the design constraints, and the actual function(s) that the system is intended to perform [1].

System Architectural Design:

The system architecture will be based on the system requirements (including the use of any pre-developed items, if needed). The architectural design phase will determine the architecture of the total system, including the organization, functions, requirements and interfaces between modules/subsystems (if any are included). The design documentation for those aforementioned system aspects will also be produced at this stage [1].

FPGA Requirements Specification:

This step matches up with the "Component Requirement Specification" phase seen in Figure 11. The required for the FPGA(s) will be based on the system architecture and the overall system requirements. It includes requirements for the function(s) each FPGA will perform, interfaces and I/O, operation parameters and operating environment, performance/timing requirements, and fault tolerance features [1].

System Integration and System Integration Testing:

In this stage, the overall system is constructed from new and pre-developed components. System verification Is generally done through testing, to check the overall system function and system interfaces [1].

System Validation:

This represents the final step in the process. The compliance of the entire system is checked against the system requirements, using a validation test plan and test specification [1].

2.1.7 Advantages of FPGAs

There are certain potential advantages to using FPGAs in the I&C systems in NPPs. Several of those advantages are listed and discussed in brief in this sub-section.

Reduced Complexity and Easier V&V Process [1,3–5,51]**:**

FPGAs are seem to be less complex than traditional software-based systems. The use of OS and/or application software would make even specifically designed safety PLCs, microprocessors or Distributed Control Systems (DCS) more complex than standard FPGAs. This, in turn helps reduce the testing and qualification process. A comparison of Complexity vs Capability is shown in Figure 14 [1]. More advanced FPGAs can include soft processor IP cores, or Hard Processor Systems (HPS). This will increase their functionality, but also their complexity. The drawbacks of FPGA capabilities are discussed in sub-section 2.1.8.

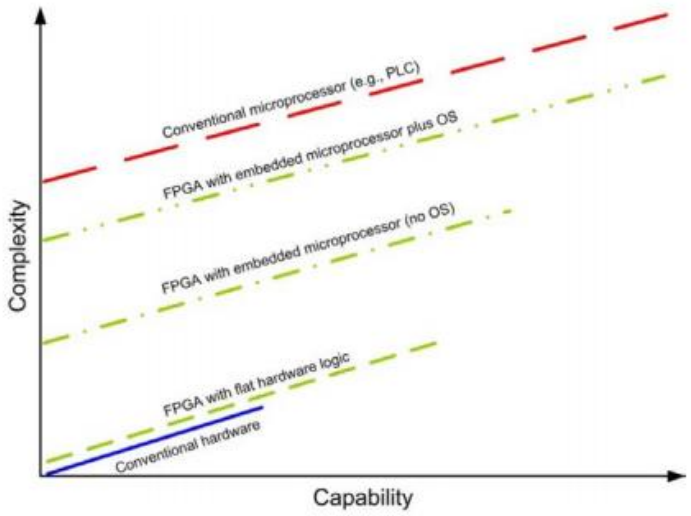


Figure 14: Complexity and Capability of Selected Digital Logic Devices

Faster Processing Speed (Reduced Response Time) [1,3,52]**:**

FPGAs have faster processing speeds, and in turn a lower execution time than software-based systems. This is because FPGAs do not need to run any software (the logic functions are executed as pure

hardware), and that the FPGAs are capable of true parallel processing, if hardware resources are not shared.

Logical Separation (Separation of Safety Functions) [1,4,5]**:**

As FPGAs can perform logical operations in true parallel execution, logical functions can be separated on the FPGA chip. For example, monitoring functions can be separated from safety functions, or independent safety functions can be separated from each other on the same chip, so that failures in one function will not affect other safety functions. This is in contrast with a microprocessor, where sharing an OS or other software services makes the independence of functions more difficult to achieve.

Useful for a Wide Range of Applications [1,4,5] **:**

FPGAs have the ability carry out a large variety of logic functions, making them applicable to most NPP systems. They are especially well-suited to safety systems, as those systems tend to rely on simple logic functions (making FPGA programming easier), and require has response times.

Code Portability and Reduced Obsolescence (Longer Lasting Technical Support) [1,3–5,51]**:**

If only HDL code is used to program the FPGA chip (i.e. vendor specific IP cores are not used), the FPGA code becomes very portable, and can be ported to other FPGAs, FPGA families, and even FPGAs from other manufacturers. Additionally, as portions of the FPGA programming cycle are circuit independent, so they will not have to be repeated on replacement chips, reducing the obsolesce of the HDL code. A block diagram of this aspect is given in Figure 15 [4].

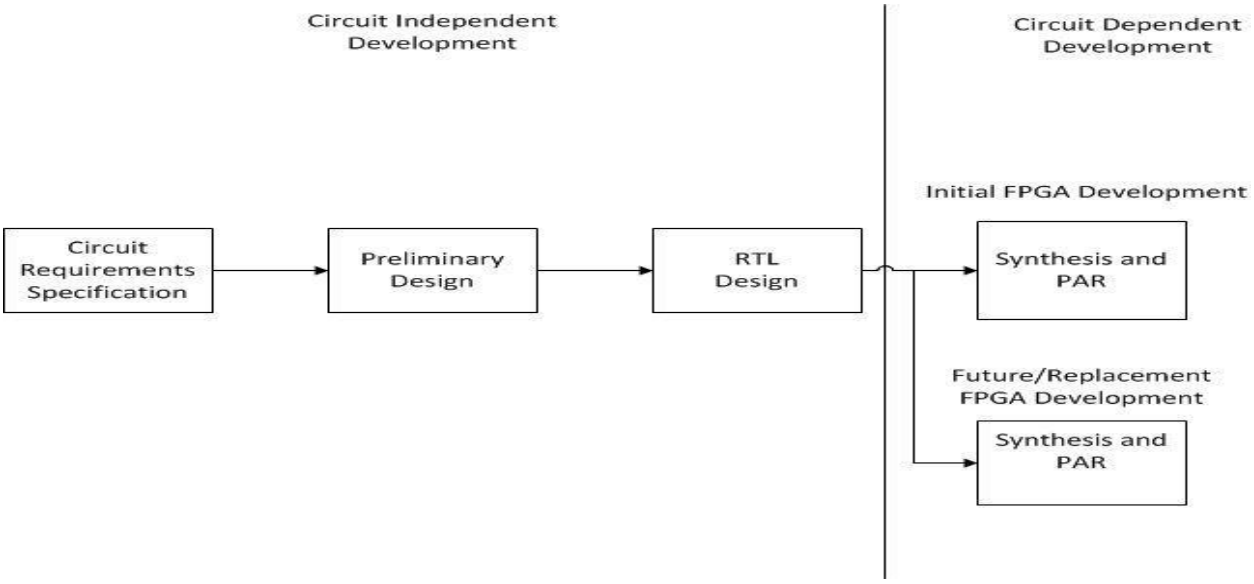


Figure 15: HDL Code Portability

Increased Diversity [1,3,4]**:**

Including FPGAs along with other forms of digital/analog logic in a system will increase the diversity of components in that system, thus reducing the likelihood of Common Cause Failures (CCF). The use of FPGAs is techniques to help meet diversity requirements, such as those set out in NUREG/CR-6303 [53], NURREG/CR-7007 [54] and EPRI-1002835 [55].

Reduced Vulnerability to Cyber security Attacks [1,3,4]**:**

FPGAs are generally more resistant to cyber security attacks than software-based systems. FPGA-based systems can be implemented without high-level general purpose components that are more easily altered/tampered with. The system itself can be designed such that physical access to the FPGA is needed to alter the program, and in the case of OTP FPGAs, the configured FPGA logic cannot be altered.

Increased Cost Effectives [1,4]**:**

The potential factors of reduced complexity, reduced V&V process, and the use of pre-developed tools are believed to lead to more cost-effective implementations of FPGA-based systems.

FPGA Prototyping [1]**:**

Reprogrammable FPGAs allow for quicker and more cost-effective prototyping for final systems using other technologies.

2.1.8 Disadvantages of FPGAs

FPGAs are not without certain drawbacks and limitations, however. Several of the potential issues regarding FPGA-based systems are discussed in this sub-section.

Lack of Experience with FPGAs in the Nuclear Industry [1–4]**:**

The implementation of FPGA-based systems is relatively new in the nuclear industry. This means that there is not a wealth of knowledge from which to draw design experience, operating experience, lessons learned and best practices, with regards to FPGA-based NPP I&C systems. In addition, there is little technical documentation for FPGAs, with only one international standard published [37] at the time of this thesis, however additional design documentation is said to be in the works.

Use of Vendor Specific Design Products [1,4]**:**

The use of vendor-specific design aspects, such as hard macros, synthesis/timing directives and IP cores, will reduce the portability of the FPGA logic, as the code cannot be directly copied into a different FPGA make/model from a different vendor. The transparency of the IP cores is a second issue, and can complicate the regulatory review process (negating advantages of FPGAs in the V&V process), as the internal works of the IP core would not be known to the end-user, and the vendor may change the IP core over time.

Availability of FPGA Design Tools/Products [1,4,7]**:**

There are not a large number of vendors for FPGA chips and design/tools, which limits the options regarding the choice of FPGA technology. Similarly, there are not many FPGA-based I&C systems available for NPPs, that have gone through the approval/qualification process.

A separate issue involves the transparency of the design tools. As the PAR process is performed automatically using the synthesis/PAR tools there is the question of how well the end design will reflect the original HDL code.

Need for Specialized Experts (FPGA Hardware/Software) [1,4]**:**

The development, analysis and review of FPGA-based systems require personnel with expertise specific to those FPGA-based systems. The differences between FPGA hardware/software from analog devices and traditional software-based devices (i.e. HDL coding, RTL, timing analysis, design tool usage) means that FPGAs would need their own specific design/review process, performed with FPGA technical experts. The lack of FPGA experience in the nuclear field adds to this problem, as fewer qualified personnel would be employed in the industry, with regards to FPGA-based systems.

Decreased Access to Internal Signals [1,4]**:**

In general, there will be reduced access to internal signals of FPGAs, when compared to a micro-processor or conventional electronics. This can create issues with regards to monitoring/testing the FPGA system and system logic. Implementing test access ports can help mitigate this issue, but it cannot be guaranteed that all signals will be connected to that test access port. Additionally, any code optimization algorithms used by the design tools may inadvertently eliminate signal outputs, making testing/monitoring of the internal signals more difficult.

Issues with Graphics and Human System Interfaces (HSI) [1,4]**:**

The hardware-based solutions of FPGAs do not lend itself to complex graphical and HSI applications as well as a software-based solution. Additionally, HSIs generally need to be modified more often than safety or control algorithms, and those modifications may be easier to implement in traditional software (especially if OTP FPGAs are used).

2.1.9 Comparison of FPGAs and Other Electronic Control Technologies

Previous sub-sections in Section 2 have provided some discussion on the advantages/disadvantages of FPGA based I&C systems in NPPs, and comparisons between FPGAs and other hardware or software-based electronic logic technologies. In this sub-section, a brief summary of the characteristics of various FPGA utilizations and other forms of electronic logic will be considered, from information in the literature [4].

There are several different ways FPGAs can be utilized as an electronic design technology. This ranges from the pure hardware implementation ("flat hardware logic") in the simplest form, to FPGAs with embedded microprocessors (soft processor IP cores). In between those two, there are FPGAs utilizing IP cores to ease in the application of repeated and/or more complex functions, and FPGAs that serve as emulators, to emulate legacy software functions. While some of these implementations increase the functionality of the FPGA, and in turn, the applications the FPGA can be applied to, they will also increase the complexity of the device, potentially negating some of the advantages of FPGAs.

Table 4: Comparison of FPGAs and Other Electronic Control Technologies

Technology (Examples/ Comments)	Design Process	End Product	Safety Justification	Technology Support Length	Level of Functional Capability
Conventional Hardware	Hardware Design	Hardware	Design Review (HW)	Long	Low
Analog circuits, relays and discrete digital electronics (e.g. TTL)	Conventional HW Design and HW QA process	Discrete HW Components (e.g. TTL)	Functional Testing Conventional HW design, design controls and QA	Components Obsolete (Used to be long)	Difficult to implement self-tests and complex algorithms
FPGA **(Flat HW Logic)**	HW Design (SW Assisted)	Hardware	Software-similar HW design process review	Long	Moderate-High
FPGA HW logic only configured in IC	IC Design/V&V, assisted by software tools Formal Methods	IC	Review design and V&V processes (software) Review HW Design Functional Testing	Similar to conventional HW (code portability may extend support length further)	Parallel, independent logic Self-Test features Difficulty with complex HSI
FPGA **(Flat HW and IP Cores)**	HW Design (SW Assisted) IP Core Integration	Hardware	Software-similar HW design process review Evaluation of IP Cores	Moderate-Long	Moderate-High
Flat HW logic used alongside IP core(s)	HW design plan evaluation (SW assisted), with integration/ evaluation of IP core(s)	IC	HW Design Process Review Assessment of IP Core(s) (COTS review process)	Similar to Flat HW Logic FPGA, but is partially dependant on support of IP core(s)	Same as Flat HW Logic FPGAs IP core(s) may simplify new/ complex functions
FPGA Emulator	HW Design (SW Assisted) SW Re-Use	Hardware and existing Software (qualified)	Software-similar HW design process review SW evaluation	Long	Existing SW Functionality
FPGA emulates an existing processor (legacy SW)	HW design (SW assisted) for emulator	IC Emulator Legacy Software	HW design process review Assessment of emulator	Typically the same as for flat HW logic FPGAa	Same functionality as the original (legacy) software

SW Functionality	Re-use of legacy SW		functionality		
Technology (Examples/ Comments)	**Design Process**	**End Product**	**Safety Justification**	**Technology Support Length**	**Level of Functional Capability**
FPGA with Microprocessor (SoC)	HW Design (SW Assisted) SW Design	Hardware Application-Specific SW	Software-similar HW design process review SW Design process/ design review	Moderate-Long	High
Flat HW logic with an Embedded Microprocessor	HW Design (SW Assisted) for IC SW Design/QA for SW (Processor)	IC Application SW and/or PDS Components	HW Design/ Process Review SW Development and V&V Process Review	Dependent on PDS components and OS (if used)	Self-tests More complex functions (than flat HW) Limited HSI
Conventional SW-Based System	SW Design	Software and Hardware	SW Design process/ design review	Short	High-Very High
Microcontrollers, DCS, PLCs, etc.	SW Design and QA, assistance of SW tools	Conventional Microprocessor Application and basic system SW	SW Development and V&V process Review based on SW engineering standards	Microprocessors and their basic SW have shorter lifecycles than conventional HW	Complex algorithms Self-Tests Information display and complex HSI

A summary of the information from the literature comparing the various ways FPGAs can be deployed, as well as comparisons with conventional hardware and microprocessors is given in Table 4 [4]. That table is ordered in increasing order of complexity/functionality.

Notes:

SW: Software

HW: Hardware

Safety Justification:

It should be noted that the safety justification discussed in Table considers only the freedom from design errors and the assurance of design correctness. Other facets of safety justification, such as HW qualification, QA of the manufacturing process, etc., were not considered by the literature source.

Pre-Developed Software (PDS):

"Software part that already exists, is available as a commercial or proprietary product, and is being considered for use" [37,56]

2.1.10 Additional Uses For FPGAs

While FPGAs have received only recent consideration in the nuclear industry, they have seen more use in a variety of other fields, for a much longer time. Additionally, technologies similar to FPGAs have sprouted off in more recent years, each with their own benefits and drawbacks. These additional facets of FPGA-related technology is briefly discussed in this section.

2.1.10.1 FPGAs in Similar Industries

The aerospace field is sometimes seen as similar to the nuclear field, because both require a high level of reliability and quality. In addition, in some high altitude and space applications, radiation can be encountered, which could affect unshielded hardware components. Although shielding is present in nuclear power plants, special consideration should be given to the effects of different forms of radiation on the FPGAs in case of an accident or leak. Several examples and solutions have previously been investigated and published by the aerospace industry. In the end, the use of FPGAs in aerospace applications could be used as a guide for implementing FPGA based systems in the nuclear field [3,57].

2.1.10.2 System on a Chip

A System-On-A-Chip (SOC) combine an FPGA and a HPS onto a single chip, with interconnections between the two parts of the chip [3]. Additional components, such as extra RAM, ROM, analog-to digital and/or digital-to-analog converters, etc. This allows for the implementation of FPGA and traditional software-based functions (either with or without an OS), on the same chip, utilizing the benefits of each system. As an example, an HSI function could be implemented on the HPS, with the high-speed digital logic functions performed on the FPGA, and data shared between the two, using the interconnections. While this increases the functionality of the chip, it also increases the complexity of those systems [4].

2.1.10.3 Field Programmable Analog Array

The Field Programmable Analog Array (FPGAA), is thought of as the analog cousin of the FPGA. The idea behind FPAAs is similar to that of FPGAs, however the FPAA contains Configurable Analog Blocks (CAB)/ Configurable Analog Modules (CAM), instead of the CLBs used by FPGAs. The FPAA is then configured to carry out the desired analog function(s). FPAA technology is newer and less developed than FPGA technology, and at the time of this thesis, had not achieved the level of mainstream use as FPGAs [3,58].

2.2 FPGA Literature Review

An extensive literature survey was undertaken at the start of the research work, in order to understand the latest developments regarding FPGA systems in nuclear power plant I&C. This survey included journal publications, conference papers and reports form industry and engineering groups. The results were categorized an included in a review paper entitled "A Review of the Current State of FPGA Systems in Nuclear Instrumentation and Control", that was published in the ICONE-21 conference proceedings [6]. The results published in that review paper are shown in sub-sections 2.2.1 - 2.2.4. Sub-section 2.2.5 will briefly discussion developments in FPGA-based I&C systems that were published after the review paper was accepted. Sub-section 2.2.6 discusses the potential gaps in the (published) knowledge with regards to FPGA-based systems, and how the literature review results provided guidance for the research in this thesis.

2.2.1 FPGA Developments in North America

In Canada, there are certain projects underway (some just for test and research purposes) by a variety of organizations studying the usage of FPGAs for safety systems in CANDU nuclear plants. A few research cases involving the SDS-1 Safety Systems are presented here, as are the experiences of Ontario Power Generation (OPG) with FPGAs at their Darlington facility. So far in the USA, only one NPP has begun using FPGA based systems during its operations, and it is not for a safety critical system. This NPP is the Wolf Creek plant located in Kansas, and it was granted a license for an FPGA based system implementation in 2009. There are plans to introduce more of them in the near future. The Diablo Canyon Plant is planning to perform similar system upgrades

System Design: Canada (SDS-1)

A recent paper by Jiang and She from 2011 studied the use of FPGAs in Shutdown System No. 1 (SDS-1, the use of shutdown rods dropping into the reactor core) in the CANDU power plants [59]. The goal of this project was to compare the response time of an FPGA based shutdown system, with the response time of an operating system (software) based system used in current CANDU plants. To carry out this

project, a Hardware in the Loop (HIL) nuclear power plant training simulator was used to simulate real-life operation of a CANDU NPP, where a trip parameter of the SDS-1 trip logic is used for the study.

The design and implementation of this FPGA system involved the use of some basic operations, such as "range checking" and "sorting" that were carried out in parallel. Range checking involves comparing a measured value against the upper and lower bounds simultaneously (due to parallel circuits). The sorting function creates a sorting network that allows pipeline implementation that will increase the processing speed and enhance the efficiency of the system.

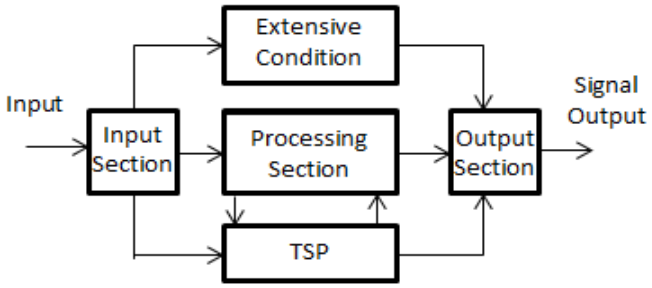


Figure 16: System Architecture of the FPGA-based SDS-1

Figure 16 provides a basic outline of the overall digital system architecture of the FPGA based SDS-1 [59]. This figure shows the relationships between the different functionalities and subsystems. The aforementioned functions were programmed using a HDL variant known as Very High Speed Integrated Circuits (VHSIC) HDL. The design of the FPGA chip was completed, and the designed system was downloaded onto the FPGA chip to be tested.

Figure 17 is a more detailed system description of the FPGA based SDS-1 Trip Channel [59]. It demonstrates the circuits and logic blocks involved with this system. After the trip thresholds have been determined, the processing logic will determine if the trip conditions have been met. If that occurs, a signal is sent to the output circuit to trip the system. The "Extensive Conditions (EXC)" logic block is used to determine if an extensive condition would be an effective trip condition. This is done because the reactor can have some special operating status. This block also controls the

output circuit and can overwrite the trip signal. The output circuit will then send out the trip signal under the right conditions.

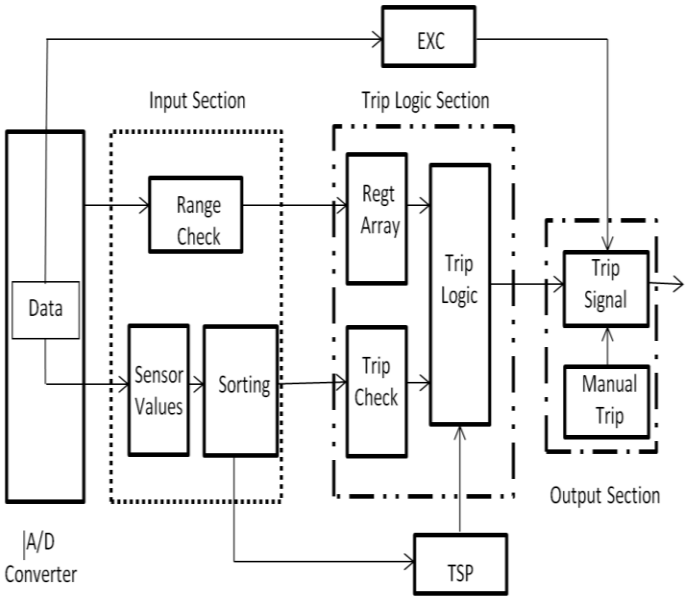


Figure 17: System Description of the FPGA based Trip Channel for SDS-1

System Testing and Simulation (SDS-1): During the test, the response time and the functionalities of the FPGA-based system were analyzed. The experimental set-up consisted of connecting this FPGA trip channel to the HIL simulation through the use of the NPP simulator. In this experiment, "Channel E" of the SDS-1 simulator was replaced by the FPGA based SDS-1 [59].

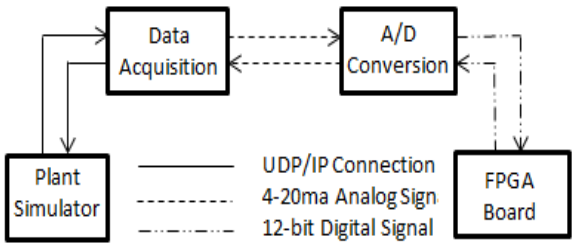


Figure 18: HIL Simulation for Functionalization Test

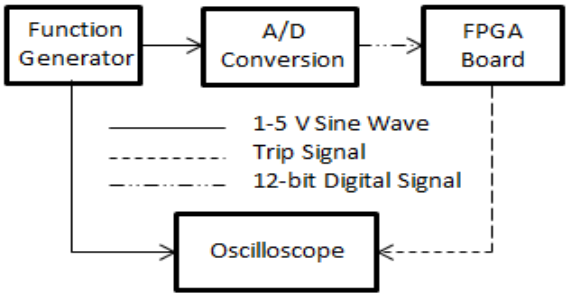


Figure 19: Set-Up of Response Time Measurement

Figure 18 shows the block diagram for the experimental test set-up of the HIL simulation, used to test the functionality of the FPGA based SDS-1. It consists of the NPP training simulator, an A/D converter, data acquisition system and the FPGA system. The simulation data (from the NPP simulator) is sent to the FPGA board via a data acquisition system and a UDP/IP Ethernet connection. Data is passed back to the simulator the same way [59].

Using the set-up shown in Figure 19, the response time of the aforementioned HIL setup was determined. This was done separately since the time step of the simulator is longer than the clock frequency of the FPGA. This new setup applied a sinusoidal signal to the FPGA, and the output responses were recorded by an oscilloscope [59].

The functionality chosen for this test was known as "Steam Generator (SG) Low Level". The measurements of the response time of the FPGA trip channel were done using the set-up in Figure 19. The simulations were performed over a variety of operating conditions.

System Results (SDS-1): The functionality tests showed that the FPGA based SDS-1 trip functions properly under the "SG low level" operating condition. It was seen that the proper trip signals were sent by the FPGA trip channel. The results recorded during the simulation showed a very fast response time for the FPGA based system—faster than that of software based PLCs used in NPP applications [60] . After analysis of the timing simulation, it was seen that the FPGA trip channel had an average response time of 10.50 ms, whereas the aforementioned software based PLC system had a response time of 78.69 ms [60]. These results show that the FPGA trip channel has an average response time that is about 86.66% lower (faster) than the response time of the software trip channel. By this value, the authors meant that

the response time of the FPGA-based system accounted for 13.34% of the response time of the PLC system. It should be noted that this test was done solely to evaluate the speed of the FPGA response, and did not take into account other advantages that these PLCs possess such as triple redundancy for the PLC, as opposed to no redundancy with this FPGA channel. The author's (Jiang and She) concluded that there were four advantages to using FPGA based shutdown systems, including faster response time, fewer failure modes, faster licensing process, and a better resistance to obsolescence [59].

Further Work on SDS-1: In 2012 Jiang and She performed more research into improving SDS-1 response times through the use of FPGA systems [61]. Again, the HIL and the NPP simulator were used. However, this test was expanded to also include an offline, industry-standard thermal hydraulic simulator known as CATHENA. In this situation, a Large Loss of Coolant Accident (LOCA) emergency event is studied, where the response time will have an impact on safety.

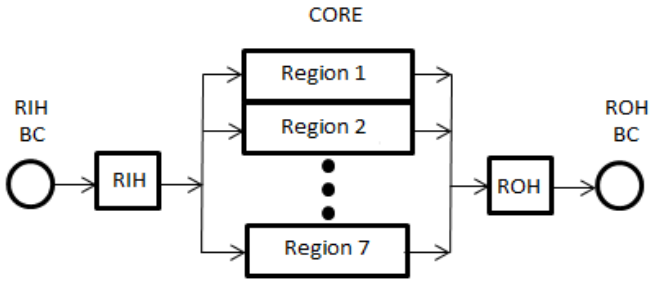


Figure 20: CATHENA Simulation Model for LOCA Study

Figure 20 demonstrates the simulation model used in that paper [61]. In this model, the core is broken up into seven regions. These regions are then connected to Feeder Pipes, which are in turn connected to the Reactor Inlet Header (RIH) and the Reactor Outlet Header (ROH). Boundary Conditions (BC) are then applied on either end. In reality there would be two independent loops, but this model is simplified to consider one combined loop. The arrows on either side of the core denote Feeder Pipes.

The CATHENA software was used to show how the behaviour of the critical parameters of the reactor during the accident are dependent on the shutdown time. The Neutron Overpower measurement was used as the trip signal in the FPGA trip logic implementation, as it is the most important parameter in

this situation. The HIL simulator was used to model a LOCA event in real-time. The FPGA trip logic was again used to replace the standard logic in one of the channels of the trip system. The experiment followed a similar set-up with similar components as in the 2011 paper, with a key difference being that the response time of the FPGA was not being directly measured. Instead, they measured the speed at which the neutron power in the (simulated) reactor dropped. The FPGA system used was a NI PXI-7811R system from National Instruments with a Virtex-II 1M gate FPGA chip, programmed using LabVIEW.

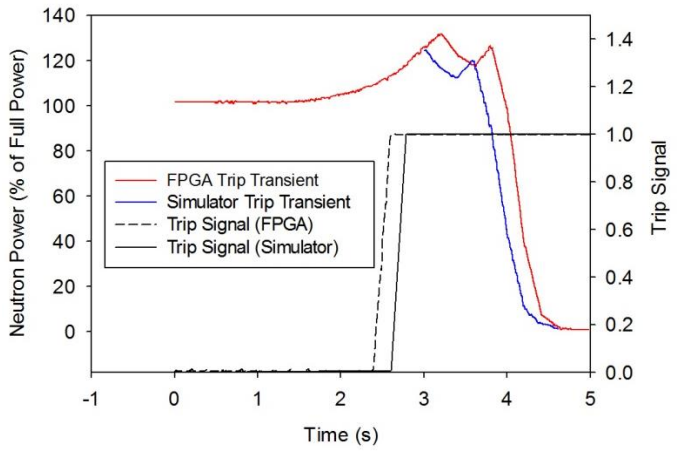


Figure 21: Comparison of Neutronic Power Between the FPGA Trip and Simulator Trip Channels

In Figure 21, the results of the simulations are presented [61]. It was found that the FPGA Trip channel had a lower response time (faster response) than the simulator software trip channel. This in turn means that the peak transient of the power is lower in the instance of the FPGA trip system than in the case of the simulator software trip channel, due the fact that the transient response is a function of the response time of the SDS. From the results discussed above, Jiang and She again found that the FPGA-based system reduced the response time of the SDS-1, leading to increased safety, due to a faster reactor shutdown. These authors also feel that this could lead to the reactors operating at a higher power, since the reactors could be safely shut down more quickly than before.

Current Project: Canada (Radiy and AECL)

In a 2010 project by Xing et al., Radiy and AECL collaborated to create an application development process based on the Radiy FPGA platform, for the purpose of a pilot project for implementing shutdown systems in CANDU plants. The goal of this would ultimately be to create a FPGA development

process that will meet regulatory and safety requirements [3,62]. Research into the SDS for CANDU plants is ongoing.

Current and Future Project: Canada (OPG and Darlington)

In the 1990s, FPGA technology began to see some use in CANDU plants located in Canada. The first application was a FPGA-based emulator for the once popular, but now defunct PDP-11 computer. They were used for several non-safety systems such as the Fuel Handling System for about 10 years. Recent plans include using a more modern version of the emulator in Digital Control Computers (DCCs) that are used in the control of the reactors. Peripheral devices that are used alongside the DCCs are going to be replaced by FPGA-based technology. In some cases this has already been done. These systems included the Moving Head Disks (MHDs) and magnetic tape drives, and were part of Stage 1 of the replacement. Stage 2 is ongoing, and will encompass more complex components, such as the CPU and Operator Console Panel. OPG is also considering implementing an FPGA-based reactor protection system, potentially in the near future [4,5].

Current Project: USA (Wolf Creek Upgrade)

Main Steam and Feedwater Isolation System: The old Main Steam and Feedwater Isolation System (MSFIS) was replaced with a new FPGA-based system in 2009. The system was based on the Advanced Logic System (ALS), by CS Innovations, and includes flash technology based FPGAs. The purpose of the MSFIS is to receive the signals from the Solid State Protection System (SSPS), (or alternatively manual signals), and then send signals to individual valves [3].

The MSFIS is considered a simple system, as it does not receive any measurements from sensors or detectors. Due to its simplicity, and also that it was a system that needed to be replaced, it was easier to get NRC approval for the FPGA technology than in most cases involving software. In this case, it took about 2 years for the NRC to grant approval for this project, which is much faster than the approval for traditional software that can last up to 10 years in certain cases [63,64].

The ALS is a scalable, modular system that can be modified after installation if desired. It provides dedicated and redundant control logic to maximize safety. It is believed, (by Wolf Creek and CS Innovations), that this new system will lead to increased integrity and reliability, reduce the cost and effort required for modifications, and reduce the impact of obsolescence in the future. It would also allow for more advanced testing and diagnostics [63,64].

Since the MSFIS was the first NRC approved FPGA-based system, it meant that there were no regulatory documents available on the issue of qualifying FPGA systems. The Safety Evaluation Report (SER) for the replacement MSFIS at Wolf Creek serves as one of the best guides for getting NRC approval of FPGA-based systems (more recently, NRC guidelines have been developed, as discussed in a later section) [5]. There has not been much data published regarding the performance of the FPGA based systems at Wolf Creek, however there have been no publically reported problems with the system.

Current Project: USA (NRC Guidelines)

Due to the recent interest in FPGAs for Nuclear I&C, it would be pertinent to discuss the guidelines set out by nuclear regulators. The US NRC published a report in 2009 entitled “Review Guidelines for Field Programmable Gate Arrays in Nuclear Power Plant Safety Systems” [65].

The report compiled design practices for FPGAs that would be utilized by NRC members to review any and all FPGA-based safety systems in NPPs. It is comprised of previous regulatory documents as well as new information to cover issues not previously discussed in past reports. It breaks down the FPGA design practices into three main groups; FPGA design entry methods, FPGA design practices, and FPGA design methodologies.

The report further breaks down these design practices into four important attributes; reliability, robustness, traceability and maintainability, according to existing frameworks. It states that the FPGA should be treated as a device composed of hardware and software, and is similar to other Complex Programmable Logic Devices (CPLDs). A recommended design flow (including design path and verification path) is illustrated Figure 22 [65].

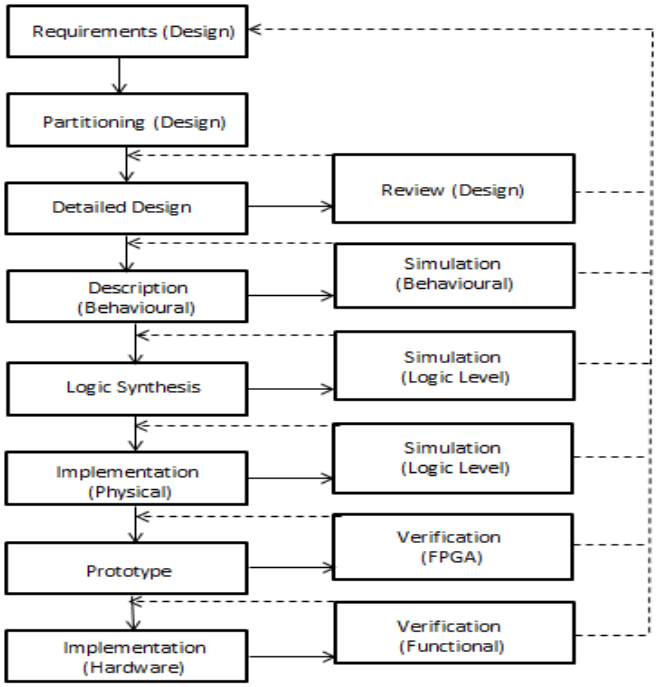


Figure 22: US NRC FPGA Design Flow

The flow chart found in Figure 22 showcases the design flow that is recommended by the US NRC. This includes the design path and the verification path, where all the design path steps correspond to a step in the verification path. The design steps might need to be revised based on the results of the verification steps. These verification steps may also need to be changed if the initial requirements of the FPGA change, and this could, in turn, require changes to be made to previous design steps. Overall, the goal of the report was to describe and list the FPGA design practices that were considered to be acceptable, as well as listing the design practices considered to be unsafe. The report was compiled at the Oak Ridge National Laboratory by the Office of Nuclear Regulatory Research of the US NRC.

The NRC is also currently reviewing five new reactor designs that plan to utilize FPGA technology for I&C systems. This includes the AP1000 by Westinghouse, the US-APWR by Mitsubishi, the US EPR by Areva, the ESBWR from GE-Hitachi, and the ABWR from the South Texas Project [5].

Future Project: USA (Wolf Creek Continued Upgrade Plan)

Figure 9 is a representation of the planned Safety I&C system upgrades based on ALS (FPGA) systems. It was outlined in 2008/2009, and is meant to take place over 5 years [63,64]. This plan includes milestones such as the Load Shedder and Emergency Load Sequencer (LSELS) to be installed in 2012. It consists of other components, such as the SSPS, Balance of Plant Emergency Safety Features Actuation System (BOP ESFAS), etc. and shows the overall architecture of the proposed safety systems. The dashed box represents the FPGA-based MSFIS system, which has been installed. To date, there has not been any information published publically regarding the state of the project.

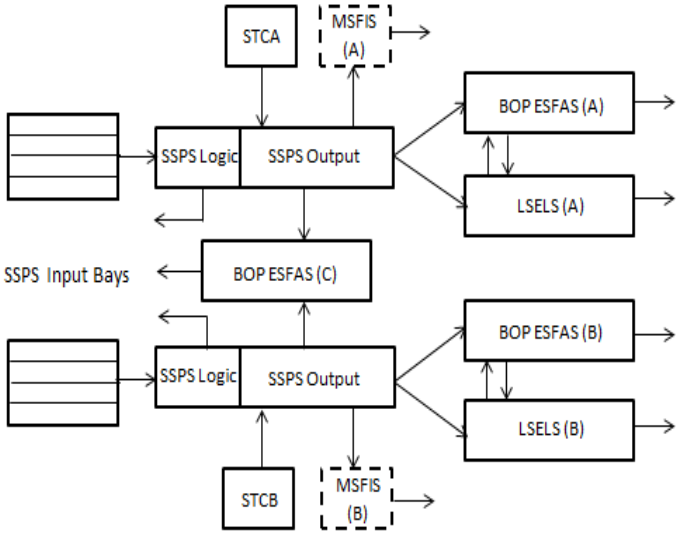


Figure 23: Proposed Safety I&C System for Wolf Creek

Future Project: USA (Diablo Canyon Upgrade Plan)

Officials at The Diablo Canyon nuclear plant, located in California, are considering replacing some of its systems with FPGA technology. The plan is to replace the Process Protection System (PPS), which currently employs a microprocessor, and was originally implemented in 1994. This system performs and processes bistable functions on the inputs from the SSPS, and then sends activation signals to the ESFAS and RTS. The new PPS will also use microprocessor technology, and will utilize FPGAs with internal diversity for tasks that would normally be done by the Diverse Actuation System (DAS), or be done manually as a way to mitigate common cause failures. The FPGA systems would be based on the platform used by Wolf Creek (but with added internal diversity), which is already approved by the NRC

[5]. The Defense-in-Depth (D3) assessment was approved by the NRC in 2011, and the plant applied for a license amendment request later in October 2011 [66]. The NRC review is ongoing, as it came up with a number of items that it wants addressed. The NRC requested that additional information be included for their review by October 31st, 2013 [67] .

Regional Results

From the research presented in the work regarding SDS-1, it was seen that the FPGA technology can perform all of the necessary functions of a CANDU shutdown system. It was also seen that the FPGA-based systems had a faster response time, and thus the shutdown system had a faster response time, than a software based shutdown system. When considering the existing FPGA-based system at Wolf Creek, due to a lack of publicly available data, it is difficult to draw conclusions about the effectiveness of that system. However, there still remains interest in FPGA upgrades for Wolf Creek and Diablo Canyon, as well as from the NRC, and some procedures for implementing FPGAs are now in place. It has been demonstrated that FPGA-based system are fast, functional, and have had an easier time clearing certain regulatory hurdles than other control technologies.

2.2.2 FPGA Developments in Asia

Overview

In Asia, there is work being done involving FPGAs for the Lungmen NPP in Taiwan, as well as by Toshiba for use in different types of NPPs. At Lungmen, a Reactor Protection System (RPS) is being designed based on FPGAs, as is a Feedwater Controller. At Toshiba, they are working on several FPGA-based safety systems for Boling Water and Advanced Boiling Water Reactors (BWR and ABWR, respectively). In South Korea, some projects have begun for systems in the new APR-1400 reactors.

System Design: Taiwan (Lungmen NPP RPS)

As of 2010, an FPGA based RPS is in the design phase, as the Lungmen NPP has not been completely constructed yet. The plan is to test it using a full-scale simulator (similar to the work done for the CANDU plants) against very detailed reactor core and NPP system models. These simulations will be run with a variety of operating conditions, as well as abnormal and accident scenarios.

The instrumentation and control systems at the upcoming Lungmen plant are to be completely digital, utilizing microprocessor and software technologies. However, on occasion software problems can go unnoticed and cause failures in these systems. To counteract this, the researchers investigated the use of quadruple redundant FPGA based technology for the RPS. The high reliability of the FPGA based systems would serve as a backup and safeguard against software faults [68].

The RPS is part of a larger system, known as the Safety System Logic and Control (SSLC), and is used in both power generation and safety applications. The new FPGA based system contains an advanced, flash-based Actel SmartFusion chip that includes its own microprocessor, internal memory, internal monitors, signal processors, and A/D and D/A converters. The RPS is also designed to have four circuit boards on it, with two separate sets of hardware logic to maximize reliability. This is accomplished by using different logic inputs, such as Verilog or VHDL [3].

System Design: Taiwan (Lungmen ABWR Feedwater Controller)

In an ABWR, the water level is an important parameter for reactor operation, and must be maintained within certain levels. If the water level in the reactor is too low or too high, it could trip a safety system, so precise control of the feedwater system is necessary. For systems at the Lungmen NPP, a possible control method is a Fuzzy Logic Controller (FLC). Here, the FLC was implemented using a modern FPGA. The full-scope simulator at the Lungmen NPP would then be used to test this new FPGA based system [69].

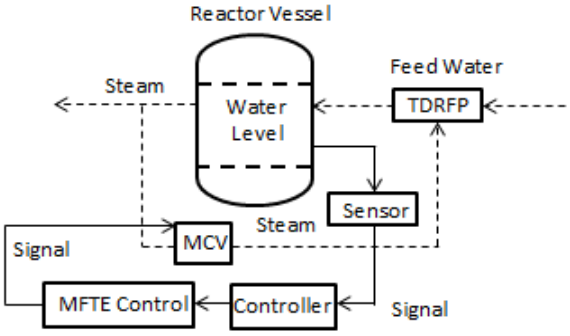


Figure 24: Schematic of an ABWR Feedwater Controller

The schematic shown in Figure 24 is that of a Feedwater controller for an ABWR [69]. A sensor measures the water level in the tank, and sends a signal to the three element controller, which in turn sends a signal to the Main Feedwater Turbine Electron-hydraulic controller (MFTE) in order to change its rotation rate. The MFTE then transforms the signal and controls the MCV in order to change the rotation speed of the Turbine Driven Feedwater Reactor Pumps (TDRFP), therefore adjusting the flow rate. The dashed arrows represent steam or water, and the solid arrows denote the signals.

Software and Hardware: The simulation software used is known as the "3KEYMASTER" software, a powerful software tool based on Microsoft Windows. It can be used to develop, model and run simulation software. For the simulations done here, the MASTER Graphical Engineering Station was used to verify and validate the FLC. These tests were run at the full power state, with four transitions (changes in the water level). The hardware component of the test included the FPGA model "Spartan-3E XC3S1600E", from the Xilinx corporation. The design software used was the ISE 11.1, also from Xilinx, and the FPGA itself was programmed by the embedded supporting software. The two Look-Up Tables of the FPGA were calculated and used for offline FLC.

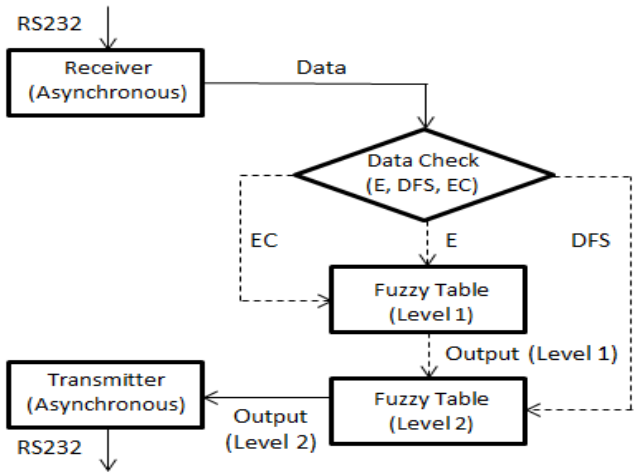


Figure 25: Block Diagram of the FLC in the FPGA

Figure25 1 is the block diagram for the FLC in the FPGA [69]. It consists of a receiver and a transmitter to receive and send data, respectively, using the RS232 protocol. The program will check if the data is E (water level error), EC (water level error change) or DFS (difference between feed water flow and steam flow), and pass the data accordingly. The two offline Look-Up Tables are also seen in the above figure,

and they are checked using a C program to ensure the values in those tables are correct. The “Level 2” Output is sent to the computer (PC) to control the pump speed.

Feedwater Controller Simulation Results: Several simulations were run to determine the system response with respect to the water level and feed water flow rate. This was done by changing the set point through a 15 cm increase (425 cm to 440 cm) and then a 15cm decrease (440 cm to 425 cm). The simulation result for the water level is shown in the figure below. Additional tests were done, with the thermal power being reduced from 100% to 90%, and also from 100% to 80%.

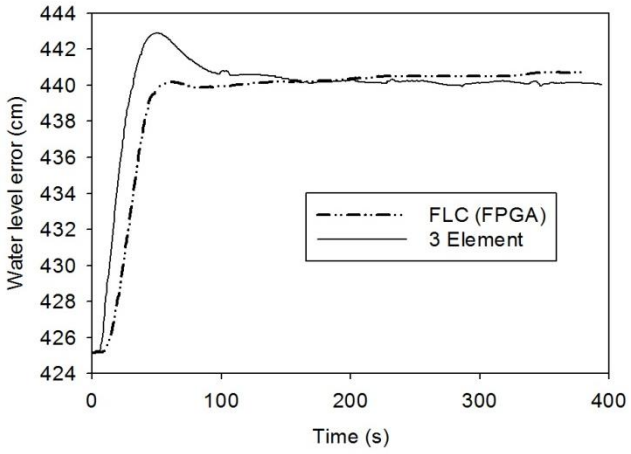


Figure 26: Performance Comparison for the Water Level after a 15cm increase in Set Point

Figure 26 compares the performance of the FPGA-based FLC and the Element (3E) control system, which is made up of two PI controllers [69]. In this test, the set point was raised from 425 cm to 440 cm (15 cm difference), and the resulting changes in water level were graphed. It was found that the FLC performed better than the 3E controller. The FLC had a lower overshoot (of water level) and a lower settling time than the PI controller. However, the rise time (of water) of the FLC was longer than the PI controller, and it did have some undershoot. The authors believe that this can be overcome by refining the rules tables (rule matrices used in the calculations) of the FLC. In the future, the authors plan on including other parameters (such as core pressure and temperature), as well as integrating the FPGA based FLC with other systems in the ABWR.

System Design: Japan (Toshiba ABWRs and BWRs)

Toshiba has been developing several FPGA based systems for safety applications for ABWRs and BWRs [3]. Some of these systems include signal generation (isolation system and reactor trip), and monitoring

systems, like the Power Range Neutron Monitor (PRM). Toshiba utilized Non-Rewritable (NRW) FPGAs for this purpose. These NRWs have added security over standard FPGAs, since they cannot be re-programmed (although it will make them more vulnerable to obsolescence). The PRM for the BWR was the first to be developed, and the design process was then used for future modules. Later Toshiba moved onto other safety systems, such as the RPS, the Power Range Neutron Monitor for the ABWR (PRNM), Start-Up Neutron Monitor (SRNM), and the Reactor Trip and Isolation System (RTIS).

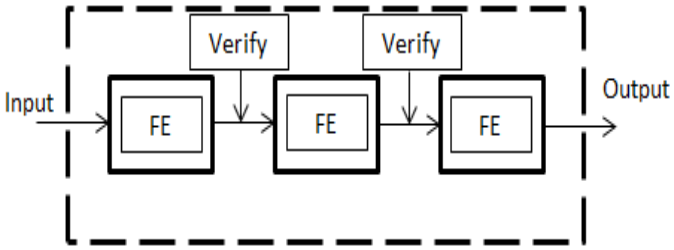


Figure 27: Toshiba FPGA Structure

Figure 27 displays the basic structure of the FPGAs used by Toshiba [70]. The functional element (FE) is said to be the minimum logical function element of an FPGA. The FEs encompass a basic logic function that (in this case) is verified using full pattern testing. The FPGA would be programmed using combinations of already verified FEs, and Figure 27 shows a conceptual FPGA made up of a small number of FEs (including verification). There would be some input, there would be logical operations applied to it, and then the output would be passed on.

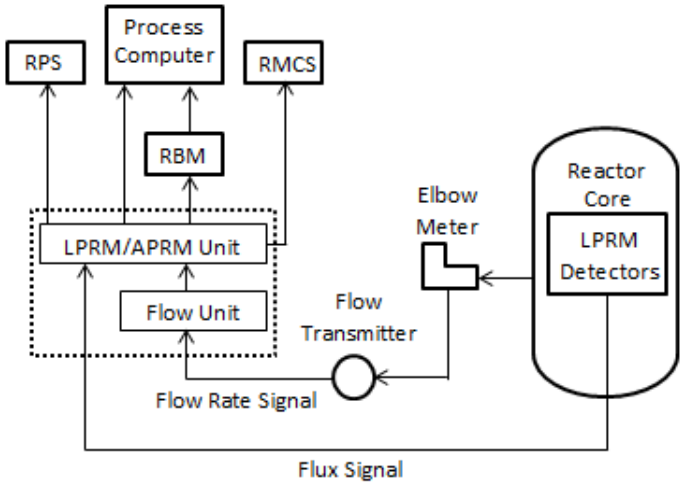


Figure 28: PRM for a BWR Plant

The PRM (shown in Figure 28) measures the neutron level in the BWR core from 10-125% of full power [70]. In that figure, PCV is the Primary Containment Vessel, PLR is the Primary Loop Recirculation, RBM stands for Rod Block Monitor, and the RMCS is the Reactor Manual Control System. It will acquire the electrical signals from the differential pressure transmitters located at the recirculation loops, as well as the neutron detectors placed inside the core. The location of the PRM in the overall reactor schematic is shown in the above figure. In the new system, each module would be composed of at least one Printed Circuit Board (PCB), with FPGAs for signal processing and Human Machine Interface (HMI). The PRM also contains the Flow Unit, the Average Power Range Monitors (APRM), the Local Power Range Monitor (LPRM) modules, where FPGAs are used to filter an electric signal that is input from a detector [70].

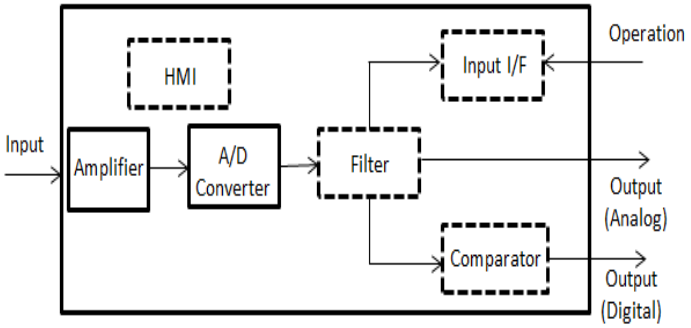


Figure 29: LPRM Module with FPGAs

The LPRM is shown in more detail in Figure 15 [70]. There is an LPRM for each neutron detector. It will acquire an electrical signal from one of the neutron detectors, then amplifies it and converts it to a digital signal. A digital filter is used to reduce the noise. After a gain value is applied to the filtered signal, it produces the LPRM level and sends that to the APRM, as well as other analog devices. The LPRM also compares the LPRM level to a setpoint, and will trigger an alarm if the level is higher than that setpoint. Most of the electronics in this module are FPGAs, which are represented by the dashed boxes in Figure 29.

For the ABWRs, there is a larger selection of FPGA-based technologies that are available. Similar to the PRM of the BWR, the PRNM of the ABWR has also been redesigned to include FPGA technology. The PRNM is one of the plant safety systems, consisting of four independent divisions that contain antifuse type FPGAs to ensure safe and reliable operation. In Figure 30, the layout of one of these divisions is illustrated [71]. In the PRNM, the LPRM detectors are connected to four APRM channels (which would correspond to four divisions), so each APRM channel acquires 52 LPRM detector signals. Each PRNM

division contains four LPRMs, and each one will acquire 13 LPRM detector signals. It is a similar set-up to the PRM in the BWR plants. The FPGAs are used for signal processing in both the LPRM and APRM [71].

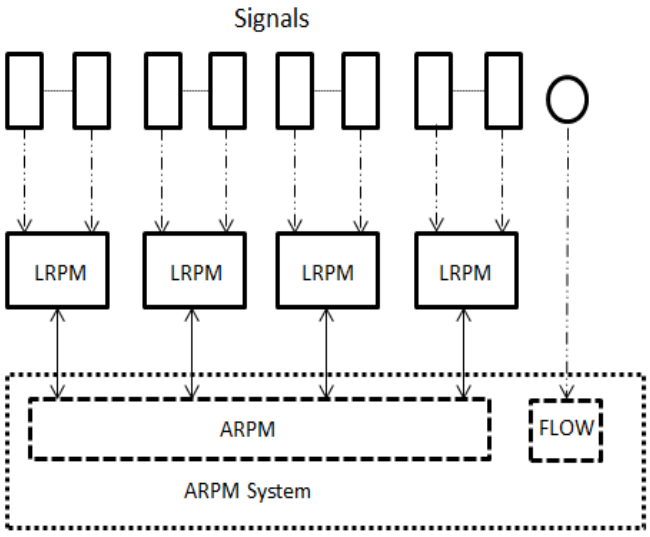


Figure 30: One Division of the PRNM for ABWR

The signal is then sent into the Reactor Trip and Isolation system (RTIS), an important part of the RPS and main steam isolation system. The RTIS has also been adapted to utilize FPGA technology.

Current Project: Japan (Toshiba Implementation)

The Toshiba modules described in the papers in the above sections have seen actual service. In 2004 the Tokyo Electric Power Company (TEPCO) began installing the SRNM, PRNM and RTIS in existing BWRs. In 2007, the first PRNM system was installed in an ABWR, and was considered to be the first safety related FPGA-based system used in these plants. The installation of non-safety critical radiation monitors began in 2003, and by 2008 TEPCO announced that it had over 200 FPGA-based neutron monitoring and radiation monitoring systems, encompassing over 3500 FPGA ICs [4,5].

Current Project: South Korea (APR-1400)

In South Korea, FPGAs have been utilized for self-diagnostic functions, as well as component control functions as part of the engineered safety features in the new APR-1400 power plants that are currently under construction [5].

Regional Results

A large amount of research into and installation of, FPGA-based systems has occurred in Asia. Toshiba especially has put in a large amount of work in these systems, installing a variety of both safety and non-safety neutron monitors, radiation detectors, and reactor trip systems, in reactors operated by TEPCO. Regarding the Lungmen plant in Taiwan, both an RPS and a Feedwater controller have been developed using FPGAs, and both systems are very important for reactor operations. Simulations have shown the effectiveness of these systems, and in some cases have performed better than previously used control technology. FPGAs can be used effectively for a variety of safety and non-safety control, monitoring and detection systems, as well as serve as an important part of technology diversification.

2.2.3 FPGA Developments in Europe

Overview

Over in Europe, Radiy RPC (Research and Production Corporation) is a Ukrainian based company that has become a major name in FPGA systems. However, they are not the only organization in Europe that is doing a large amount of work with FPGA technology. Electricite de France and Rolls Royce are working together to upgrade certain components of France's large nuclear fleet. In addition, system upgrades have occurred in the Czech Republic and Sweden.

Current Project: Ukraine and Bulgaria (Radiy)

There are a few nuclear plants in Eastern Europe that have been using FPGA based technology for a variety of systems, including safety systems for several years now. In Ukraine there are four nuclear power stations (Zaporozhye, South Ukraine, Rovno, and Khmelnitskiy) and one Bulgarian nuclear plant (Kozloduy) that employ FPGA based systems. In all, over 60 FPGA systems have been implemented by Radiy. These systems include Reactor Trip Systems, Engineering Safety Features Actuation System, Reactor Power Control and Limitation System (RPCLS), Rod Control System and Regulation, and a Control and Protection System for a research reactor [72–74].

The RPS functions by measuring the neutron power, and comparing it to a Trip Set Point (TSP). If the neutron power reaches the TSP, a signal is sent to activate the shutdown systems. The ESFAS has

multiple safety functions, including automatic process control, protection, blocking and operation of actuators, as well as manual remote control of those actuators. The RPCLS is used to regulate the neutron power of the reactor, as well as the steam pressure of the main line of the turbine. The RCS accounts for the position of the control and safety rods, as well as the drive control that will move them. Lastly, the Control and Protection System was installed in the VVR-M research reactor in 2008. Radiy has been working on these modern I&C systems for nuclear plants since 2000, with the first installation occurring in 2003. These systems were designed for the VVER-440 and VVER-1000 Model Reactors (Russian Water-Cooled Water-Moderated Energy Reactor) [72–74]. In the time these systems have been functioning, there have been no reported problems [75]. Currently, there are 28 RTS, 18, ESFAS, 9 RPCLS, and 1 each of the RCS and research reactor I&C systems installed in the aforementioned plants.

Current and Future Project: France (Electricite de France (EDF) and Rolls Royce Modernization)

FPGAs have seen some use in French reactors. EDF contracted the instrumentation and control department of Rolls Royce to perform a large modernization project at the 34 900 MWe reactors [3]. All of the Reactor In-Core measurement systems (RIC) and the Rod Control Systems (RCS) were going to be replaced with FPGA based systems. The RCS is designed to generate control signals that will move the control rods, verify the rod position; interface with the PLC based control and diagnostic unit, as well as allowing for the human system interface. The speed of FPGAs allows them to attain the required millisecond time resolution, which would be difficult to achieve with a microprocessor. These FPGAs are flash based, so they can be reprogrammed, but only with special tools and at the Rolls Royce facilities. This is done to minimize the chance of FPGAs being tampered with. The RCS is not strictly considered a safety system; however it does require a high degree of reliability. The project was started in 2005, with the first replacements coming in 2010. As of 2011, The Tricastin Unit 1 power plant had the RIC and RCS systems installed, and the second and third installations at Bugey Unit 2 and Fessenheim Unit 1 plants are ongoing. The plan is to have this refurbishment completed by 2020 [10,52].

EDF is carrying out additional research for FPGA-based technology in its 1300 MW series of plants. EDF recently developed and verified a replacement for the obsolete Motorola 6800 microprocessor that is based on an FPGA-based emulator. These systems would be used for a variety I&C tasks, including safety critical functions like reactor protection systems. This upgrade would allow for the replacement of the Motorola 6800, without losing the current software that is already qualified for safety applications [5].

Current Project: Czech Republic (Temelin 1 and 2)

The Temelin 1 and 2 plants in the Czech Republic have begun using FPGAs for Non-Programmable Logic (NPL). The implementation program began in the 1990s and was completed in 2000 when the Temelin 1 plant was commissioned. This NPL performs multiple functions, including the safety load interface. At that interface, the priority logic arbitration amongst the microprocessor-based Diverse Protection System (DPS), the Primary Reactor Protection System (PRPS), and the fixed-wire component control signals transpires. The NPL is also used to make sure the loads reach a safety state in case of a variance between the DPS and PRPS occurs. A third task of the NPL is the implementation of the Safety Diesel Load Sequencer, which also encompasses the sequencer automation tests [5].

Current Project: Sweden (Ringhals 2)

The Ringhals 2 plant in Sweden is utilizing FPGA based technology due to a replacement of some of the plants I&C systems. This project had been in the works since 1990s, and was completed in 2010. The FPGAs are used in a component of the replacement safety system known as the Component Interface Module (CIM). This module is used as an interface connecting the plant equipment to the microprocessor based safety actuation system. The CIM also responds to the signal from the independent Diverse Actuation System (DAS) and from operator commands. This module contains priority logic, to make sure that the correct signal is sent to each component, and in case of a conflict the component will be put into a safe state [5].

Regional Results

Radiy has become one of the largest suppliers of FPGA-based systems, including a number of safety systems, which has led to a large number of installations in reactors in Eastern Europe. Smaller projects have also occurred recently in Sweden and the Czech Republic, encompassing several systems. A large implementation project in France is aimed to replace older, high-reliability systems with FPGAs, and additional research is being carried out to introduce safety-critical FPGA-based systems. It was seen in this section that FPGAs can be used effectively for safety systems and other high reliability systems without issue and that are a viable replacement for obsolete technology in nuclear plants.

2.2.4 Other FPGA Developments

Overview

It is worth mentioning other uses of FPGA-based systems in this field. These include seismic sensors in case of an earthquake, measurement of nuclear pulses, as well as detectors for health physics purposes. Furthermore, other developments have occurred since the publication date, and are also briefly discussed here.

Automatic Seismic Trip System: There has been some work using a Petri Net, or PN (mathematical modelling language used to describe distributed systems) design of a FPGA based controller for Automatic Seismic Trip System (ASTS). This system is applied to the RPS, and contains earthquake sensors to measure seismic activity. It was found through simulations that this sort of FPGA based system would be feasible, and more work would be put into it in the future [76].

Measurement and Detectors: There has also been interest in using FPGAs for applications such as measurement of nuclear pulses, and for various radiation detectors. The speed and reliability of FPGAs make them good candidates to measure the pulse heights and speeds for nuclear applications (such as nuclear pulse detectors) [77]. FPGA systems are also being looked into for the electronics in radiation detectors and other health physics applications [78]. Due to many of these systems not being considered safety critical, some of them are already in use.

2.2.5 Recent Developments

Overview

Information was published regarding several other developments/applications regarding FPGA- based systems, after the literature review was published. This sub-section briefly considers those aspects, to provide a more complete picture of recent FPGA applications in NPP I&C systems.

ALS Approval: It was announced in September 2013 through a press release from Westinghouse that the full ALS has received final approval from the US NRC in a SER. This would make the installation of FPGA-based safety systems in US nuclear plants a possibility [79].

PLC to FPGA Platform Change: As mentioned previously, FPGAs have been thought of as a replacement for PLCs in safety systems. However, a lot of knowledge and experience has been built up over the years, and completing discarding that information in order to implement a new development process would be a long and risky endeavour for these safety systems. The purpose of this work was to implement the software development process for an RPS, but in this case the platform would be an FPGA, not the PLC, but would keep all the outputs from the original design. The Functional Block Diagram (FBD) designs from the PLC were transformed into the equivalent HDL (Verilog) process, using a prototype RPS (based on a real-word system) in South Korea [80].

Development Systems: In India, there has been a good deal of work done regarding newer I&C systems, for use in newer and future nuclear power plants. Many of these newer systems are digital, and include the use of software, making software reliability an important facet of the design and qualification program. In addition, a good deal of that work was focused on FPGA-based systems, due to increased reliability, reduced component counts, and reduced obsolescence. Another benefit sited was the re-configurability of the FPGA, allowing it to be programmed in the field, to meet different needs. In order to ensure the reliability of the FPGA-based systems, the same verification techniques that are applied to software systems were applied to the FPGA-based systems. Additionally, a hardware verification tool known as VBMC (VHDL Bounded Model checker) was developed to verify the design logic used in FPGAs [81].

2.2.6 Research Directions Based On Literature Review

The literature review provided a comprehensive overview of the recent uses and developments of FPGA-based systems in the international nuclear community. This literature review surveyed numerous implementations and planned future implementations of FPGA-based systems, as well as the research performed on FPGA-based systems and published in the scientific literature. On the results of that literature review with respect to the research plan in this thesis, two important points were noted:

1. The majority of the implementations and research programs surrounding FPGA-based systems considered either safety-critical systems, or systems for which a high degree of safety/reliability is required. Systems with lower reliability requirements (relative to safety-critical systems), received less consideration regarding FPGA-based systems

2. The bulk of the information in the literature focused on the design and implementation of FPGA-based systems, but it was seen that there was less focus on the reliability and safety analysis of the FPGAs and the overall FPGA-based systems.

From the literature review, it is seen that focusing this research on safety-critical FPGA-based systems is warranted, as those are the systems that would see the most use in actual NPPs. Additionally, it was seen that there is a gap in the knowledge with regards to the (published) information on the reliability and safety analysis of those FPGA-based systems. Therefore, the performance of the FPGA FMEA/taxonomy construction, and subsequent DFM analysis, will provide important research and information to fill that knowledge gap.

2.3 Reliability Analysis Techniques

This research work considered the use of two reliability analysis methodologies, Fault Tree Analysis (FTA), and DFM. While the overall goal of the research program was to model/analyze FPGA-based systems using DFM, part of that research included a comparison of DFM and FTA for analyzing those systems. As discussed in sub-sections 1.2.1 and 1.2.3, there have been concerns about the ability of FTA (and other traditional reliability analysis methods) to accurately analyze modern digital I&C systems. However, at the time of writing this thesis, there had been little information published on the topic of direct comparisons between the two methods. Therefore, a direct comparison of FTA and DFM was carried out, and is discussed in detail in sub-section 4.4. In this section, the theory and background information for FTA and DFM is presented. The discussion on FTA is given in sub-section 2.3.1, and the information regarding DFM will be presented in sub-section 2.3.2.

2.3.1 Fault Tree Analysis

FTA is one of the most common and well-established methods of reliability analysis, has seen extensive use in the nuclear industry, as stated in the literature [8,11,12]. This sub-section will provide background information and calculation methods used during FTA.

Fault Tree Analysis [82]**:** A deductive technique that starts by hypothesizing and defining failure events and systematically deduces the events or combinations of events that caused the failure events to occur.

2.3.1.1 Introduction to Fault Trees

Fault Trees are deductive methods that consist of directed, acyclic graphical models that are constructed using properties called "gates" and "events" [83]. Gates show how failure states can travel through the system being analyzed, and common gates include "OR", "AND", "INHIBIT, and "Voting" gates, among others. Events are referred to as "occurrences" within the system being analyzed, and generally represent failures of a component or subsystem. The original, and most basic form of fault trees are Static (or Standard) Fault Trees (SFT) [83]. SFTs originate from Bell Labs in the 1960s, where they were used for the safety analysis of ballistic missiles [84].

Due to the prevalence of FTA in the nuclear field, the US NRC published their own "*Fault Tree Handbook*", as means of providing training material to NRC personnel, contractors, and others that work on the construction and evaluation of fault trees [85]. This handbook was in response to recommendations given in a report by the Risk Assessment Review Group, which stated that "the fault tree/event tree methodology both can and should be used more widely by the NRC" [86]. FTA has also seen extensive use in other fields, where reliability and safety is an important concern. For example, NASA produced their own handbook on FTA, entitled "*Fault Tree Handbook with Aerospace Applications*" [87], to provide training and guidance for FTA practitioners in the aerospace field. Additionally, the International Electrotechnical Commission (IEC) published a standard on FTA [88].

Fault Trees are used to find Cut Sets, which are considered to be a “set of basic events, such that, if all events in a Cut Set occur, the Top Event will occur” [83,85]. A type of Cut Set, known as a Minimal Cut Set (MCS), is defined as “A Cut Set that does not contain other Cut Sets as a subset” [83]. Generally, the MCSs are what are desired when using FTA, as well as the probability of each MCS and the Top Event. An example of a generic Fault Tree is given in Figure 31 [83].

Figure 31 provides an example of a Fault Tree of a computer system. It contains the following components; Power Supply (PS), non-redundant Bus (B), redundant CPUs (C1 and C2) and redundant memory units (M1, M2, M3). These components would contribute to the Top Event, which in this case is designated as a “System Failure”. G6 represents a “2-out-of-3” (2oo3) “VOTING: gate, where 2 of the gate inputs must occur to propagate a failure. G4, G5, and G2 denote “OR” gates, while G3 represents an “AND” gate. Lastly, G1 denotes an “INHIBIT” gate. This type of fault tree gate requires the input and the enabling event to occur (“In Use (U)”), before the output will occur.

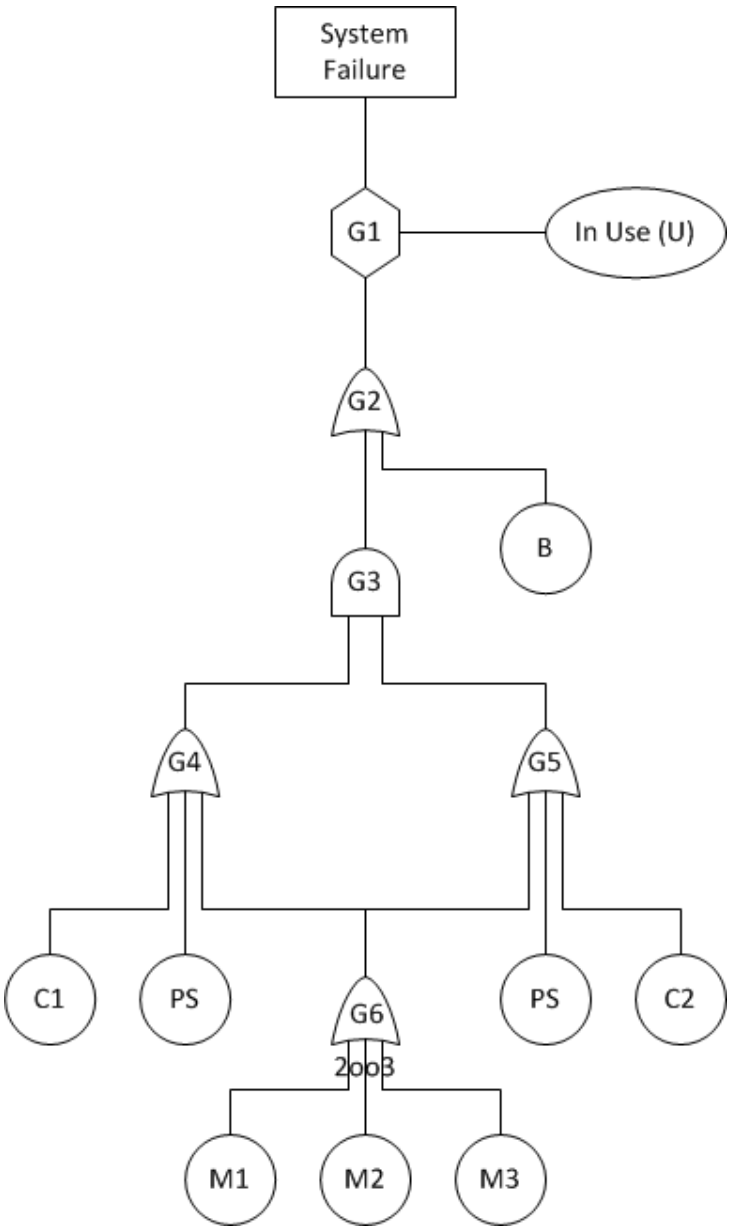


Figure 31: Generic Fault Tree Example of a Computer System

2.3.1.2 Structure Functions

In a binary, it can be considered to be in one of two possible states. These states are the “Working” and the “Failed” state, so a binary system must take on one of those states [89]. Fault trees are considered represent binary systems, and as such are confined to these two states as well. If one takes component *I*

and associates it with the binary indicator variable x_i, it is said that if x_i =1 the component fails, and if x_i = 0, the component works.

The overall system state can then be defined in terms of the state of its components. Therefore, one can define a function, known as the "Structure Function", that represents the system state [89]. Denoting the structure function as $\varphi(\underline{x})$, the system will be in a failed state when $\varphi(\underline{x}) = 1$, and the system will be in a working state when $\varphi(\underline{x}) = 0$. Here, $\underline{x} = (x_1, x_2, \ldots, x_n)$, and is a vector of all *n* component states [89].

2.3.1.3 Coherent and Non-Coherent Logic

Fault trees can be said to be either "coherent" or "non-coherent". In theoretical terms, the issue of coherence/non-coherence can be determined from the structure function of the fault tree. A fault tree is considered coherent if it satisfies two conditions [89,90]:

1.) Each component is relevant. Each component listed in the fault tree contributes to the system state. This is seen is Equation 1 [89]:

$$\boldsymbol{\varphi}(\mathbf{1}_i, \underline{\boldsymbol{x}}) \neq \boldsymbol{\varphi}(\mathbf{0}_i, \underline{\boldsymbol{x}}) \quad \textbf{(1)}$$

In other words, the structure function will be different, based on if component *I* is in the "1" or "0" state.

2.) The fault tree structure function is non-decreasing (also called monotonically increasing), as given in Equation 2 [89]:

$$\boldsymbol{\varphi}(\mathbf{1}_i, \underline{\boldsymbol{x}}) \geq \boldsymbol{\varphi}(\mathbf{0}_i, \underline{\boldsymbol{x}}) \quad \textbf{(2)}$$

This is seen visually in Figure 32 (a)-(c) [89]. In that figure, the X-axis represents the component state, and the y-axis represents the state of the structure function. In Figure 32 (a), it is seen that when the component state changes from "0" to "1" (representing a component failure), the structure function remains at "0" (system does not fail), hence non-decreasing. In Figure 32 (b), the component state changes from "0" to "1", and the structure function remains at "1" (system remains in a failed state), and is also non-decreasing. In Figure 32 (c), it is seen that when the component state changes from "0"

to “1”, the structure function also changes from “0” to “1” (component failure causes system failure). In this case, the structure function is said to be “increasing”, but as that is still “non-decreasing”, the system itself is still said to be coherent.

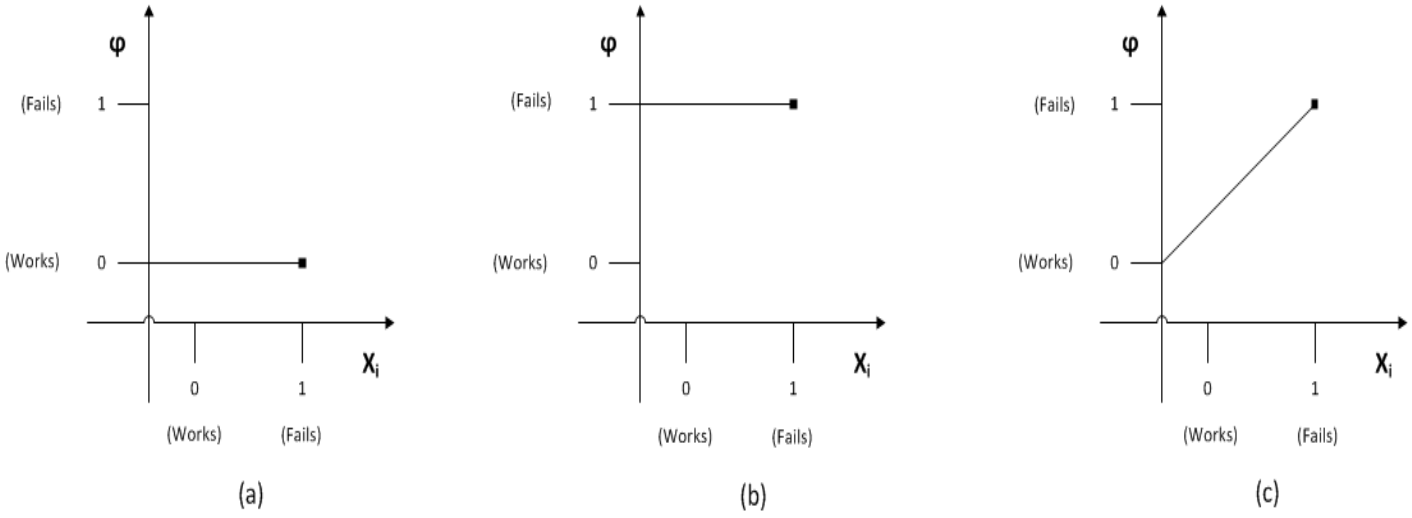


Figure 32: Three Cases of Non-Decreasing Structure Functions

On the other hand, Figure 33 shows a “decreasing” structure function. In that case, as the component fails (transitions from “0” to “1”), the system structure function transitions from “0” to “1”. As this represents a “decreasing” structure function, the system is said to be non-coherent. This would entail that the component failure would result in a failed system returning to its working state.

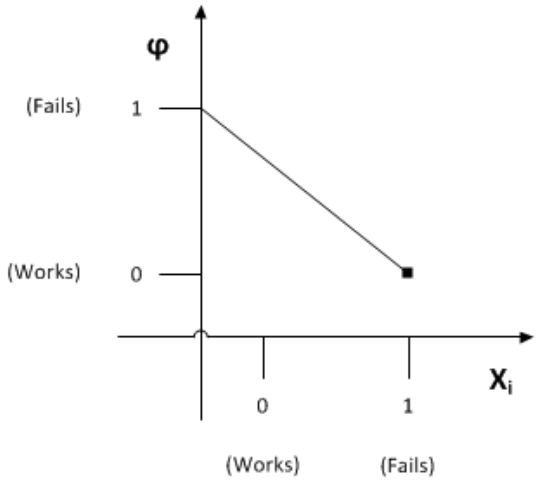


Figure 33: Decreasing Structure Function

In more applied terms, a fault tree is considered to be "coherent" if it only contains "AND" and "OR" logic (as well as combinations of those gates, such as voting gates) [89,91,92]. In a coherent fault tree, if any of the basic events had their value change from "f" to "T" (i.e. from the "unfaulted" to the "faulted" state), the Top Event could not go from "T" to "F" [93]. In other words, if the system is considered to be in a failed state, it could not return to the normal/working state, due to the failure of another component [93]. When considering FTA, much of the work performed and calculation methods was focused on coherent fault trees [85].

However, not all systems are best described using only "AND" or "OR" gates [94]. The use of "NOT" logic [95], and/or logic such as "XOR" gates that directly reference "NOT" logic [89,96], cause the resulting fault tree to be non-coherent. In the case of non-coherent fault trees, the analysis will return "Prime Implicants (PIs), which are defined as "minimal combinations of component states (working or failed) which cause the fault tree Top Event" [92,96]. PIs are thought of as the non-coherent logic equivalent of Minimal Cut Sets (MCS) [93]. The analysis of non-coherent fault trees requires different methods than coherent fault trees, however certain algorithms and software packages have been extended to incorporate non-coherent FTA [83].

2.3.1.4 Many-Valued/Multi-Valued Logic

In both the coherent and non-coherent cases discussed in sub-section 4.1.3, the system considered only binary logic, as the components, and the system itself can only take on one of two states. However, in reality, many systems and system components could exist in more than just the "Works" or "Fails" state. In a Many-Valued/Multi-Valued Logic (MVL) system, one is not restricted to using only binary logic, as every variable (component and system state) can be discretized into an arbitrary number of states. Take for example a pressure tank. The tank pressure could be discretized into many states such as "Very Low", "Low", "Normal", "High" and "Dangerously High" [14]. These states could correspond to different numerical ranges for the tank pressure, for which the system may react differently. FTA would not be able to represent the five states of the pressure tank in the same way. This extra flexibility of MVL may allow for more detailed modelling than the standard binary logic employed by FTA. In the case of DFM, it has been developed to incorporate MVL, and binary logic may also be included if desired (i.e. to represent a component like a switch, which is "ON" or "OFF"). MVL is also said to be non-coherent logic, and as such utilizes PIs instead of MCS. A list of terms/definitions pertaining to MVL is given in Table 5 [93]:

Table 5: MVL Terms and Definitions

Cut Set	A set of events that will cause a top event if all those events occur
Minimal Cut Set	A cut set that does not contain other cut sets as a subset.
Binary Logic	Two states described by True or False
Multi Valued Logic	States are discretized into an arbitrary number of states
Literal	A primary variable taking on one state (e.g. A_1)
Monomial	A conjunction of literals (e.g. $A_1B_2C_3D_4$)
Boolean Function Top	A disjunction of monomials
Implicant	A monomial of C of the disjunctive form of the Boolean TOP such that TOP $\cap$ X=X
Subsume	A monomial X subsumes the monomial Y if every literal of Y is contained in X.
Prime Implicant	X is an Implicant of TOP, and any other monomial Y subsumed by X is not an Implicant of TOP
Base of Boolean Function TOP	Any disjunction of prime Implicants which is equivalent to the function TOP
Irredundant Base	A base which ceases to be a base if one of its Prime Implicants is removed
Complete Base	The disjunction of all Prime Implicants

2.3.1.5 Qualitative Fault Tree Analysis (Coherent Fault Trees)

There are several methods to find them the MCS from coherent SFTs. These include traditional methods for finding the MCS, as well as methods based on Binary Decision Diagrams (BDDs). In the case of incoherent fault trees, several modifications and or/approximations to these methods need to be applied in order to obtain the correct MCSs.

Top Down Method (MOCUS)

A very common method for obtaining the MCS of a coherent SFT is a "Top Down" methodology [89,97], through the use of the "MOCUS" algorithm [85,97]. The MOCUS algorithm uses the property of fault trees, where the "AND" gate increases the size of individual cut sets, while the "OR" gates will increase the number cut sets, and then works via the following procedure [97]:

1.1 Name each logic gate
2.1 Number all basic events
3.1 Construct a matrix, with the top-most gate in the first column and row of that matrix
4.1 Repeat the following permutations from the top to the bottom of the fault tree:
    a. Substitute each "OR" gate with a vertical array of all the gate inputs, and then increase the number of cut sets
    b. Substitute each "AND" gate with the a horizontal array of all the gate inputs, and then increase the cut set size
5.1 Once this procedure has been completed for all gates (all cut sets have been found), removal all supersets of cut sets, to determine the MCS.

It should be noted that since the MOCUS algorithm only considers "AND" and "OR" gates, it cannot be directly applied to non-coherent fault trees without a loss of some accuracy.

An example of the MOCUS algorithm is performed, using the fault tree shown in Figure 34 [97]. To find all of the MCS of that fault tree, the MOCUS algorithm would be performed as such [97]:

Step 1:

The top gate in that fault tree is “G0”, so that would go in the first row/column of the matrix. It is an “OR” gate, to it is replaced by its individual input events (“1”, “2”, “G1”), each placed in their own row.

1

2

G1

Step 2:

As the “G1” gate is an “OR” gate as well, it is again replaced by its individual input events in separate rows, yielding:

1

2

G2

G3

Step 3:

As the “G2” gate is an “AND” gate, it is replaced by its inputs in the same row, but separate column, giving:

1

2

G4, G5

G3

Step 4:

“G3” is another “OR” gate, and is thus expanded as follows:

1

2

G4, G5

3

G6

Step 5:

The “G4” “OR” gate must now be expanded, to include its two basic events, “4” and “5”. Similarly, the “G6” “AND” gate is expanded using basic events “5” and “6”. This yields:

1

2

4, G5

5, G5

3

5, 6

Step 6:

In the final step, the “AND” gate G5 is expanded using basic events “6” and “7”, resulting in:

1

2

4, 6, 7

5, 6, 7

3

5, 6

The seven rows seen in “Step 6” represent the six cut sets of the example fault tree. It should be note Cut Set {5, 6,7} is not a MCS, as {5, 6} is also a Cut Set. Therefore, {5, 6, 7} is actual a superset of Cut Set {5, 6}, and would have to be removed, resulting in the following five Cut Sets:

{1}, {2}, {4, 6, 7}, {5, 6}, {3}

The steps for this example can be summarized in Table 6 [97]:

Table 6: Steps in MOCUS Example

Step	1	2	3	4	5	6
	1	1	1	1	1	1
	2	2	2	2	2	2
	G1	G2	G4, G5	G4, G5	4, G5	4, 6, 7
		G3	G3	4	5, G5	5, 6, 7
				G6	3	3
					5, 6	5, 6

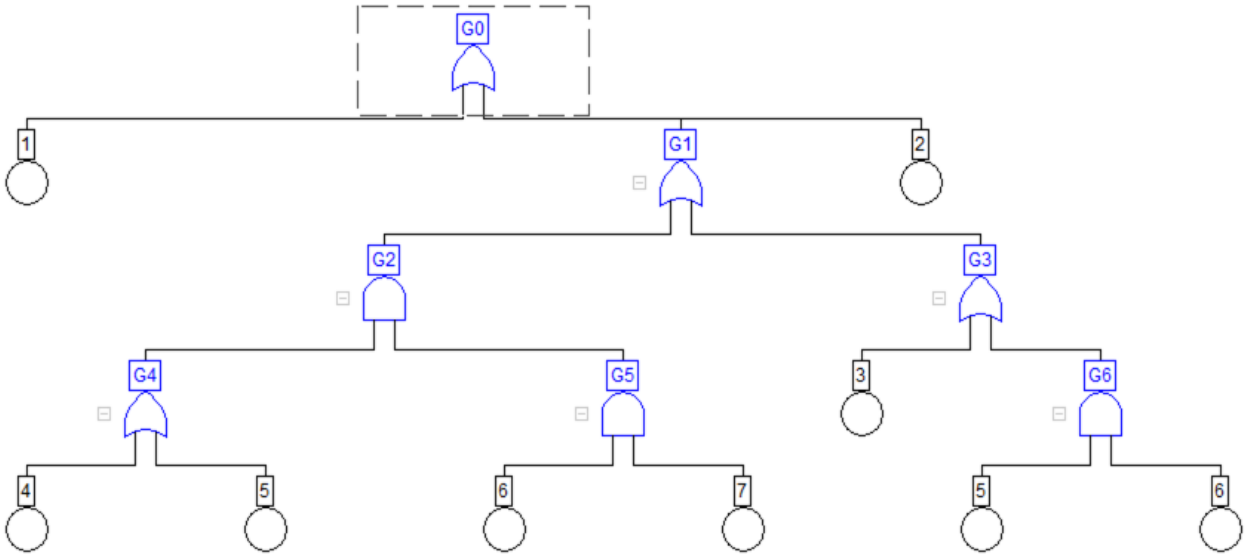


Figure 34: Example Fault Tree for FTA Demonstration

Bottom-Up Method

Performed in the opposite fashion to the MOCUS method, this algorithm starts from the bottom-most gate, and uses the following procedure [34]:

1. Take the bottom-most gate
    a. Set “OR” gates equal to the union of input basic
    b. Set “AND” gates equal to the intersection of input events
2. After all gates at the bottom level have been replaced, Boolean logic rules are used to reduce the formulas [30,34]
3. Repeat steps “1” and “2” in a bottom-to-top fashion, until the Top Event formula is found.

For SFTs, the “MOCUS” and “Bottom-Up” Methods should produce the same results for the MCS. As in the case with “MOCUS”, the “Bottom-Up” method can only be directly applied to coherent fault trees. The use of the “Bottom-Up” method employs additional logical reduction rules, to simplify the logical expressions(s) that are returned after each step in the algorithm. These logic reduction rules are [97]

Binary Fault Tree Logical Reduction Rules:

1.) A + A = A

2.) A*A = A

3.) A+B = B+A

4.) A*B = B*A

5.) A*(B+C) = A*B+A*C

6.) A+A*B = A

7.) A+B*C = (A+B)*(A*C)

In the case of these logical reduction rules, the "+" refers to the union of the input events (logical "OR"), and the "*" denotes the intersection of the input events (logical "AND").

An example using the "Bottom-Up" method, also using the fault tree in Figure 34, is given below [97].

From Figure 34, it is seen that lowest gates (bottom-most) on the fault tree are gates "G4" ("OR" gate), "G5" and "G6" ("AND" gates). These gates are expanded as:

$G4 = X_4 + X_5$

$G5 = X_6 * X_7$

$G6 = X_5 * X_6$

The next level of gates (one level above the three aforementioned gates) are "G2" ("AND" gate), and "G3" ("OR" gate). Thus, these gates are expanded as:

$G2 = G4 * G5 = (X_4 + X_5)*(X_6*X_7) = X_4*X_6*X_7 + X_5*X_6*X_7$ (Using Rule 5)

$G3 = X_3 + G6 = X_3 + X_5*X_6$

The next step of logic gates consists only of the “OR” gate “G1, which is expanded as:

$G1 = G2 + G3$

$G1 = X_4*X_6*X_7 + X_5*X_6*X_7\ X_3+X_5*X_6$

Using Rule 6:

$X_5*X_6 + X_5*X_6*X_7 = X_5*X_6$

For “G1”, this yields:

$G1 = X_4*X_6*X_7 + X_3 + X_5*X_6$

Lastly, reaching the top gate of “G0”, another “OR” gate returns:

$G0 = X_4*X_6*X_7 + X_3 + X_5*X_6 + X_1 + X_2$

Each of the above terms corresponds to one of the five MCS as shown below:

{1}, {2}, {4, 6, 7}, {5, 6}, {3}

These are the same MCS determined using the MOCUS method in sub-section 4.1.4.1. The only difference being that the MOCUS algorithm required deleting (non-minimal) Cut Sets at the end of the algorithm, however in the “Bottom-Up” method, logical reduction is performed after every level, so only the MCS were produced. As the MOCUS and “Bottom-Up” method will yield the same results, only the MOCUS algorithm was considered further

Binary Decision Diagram (BDD) Method

BDDs have become a popular method for performing FTA, due to the speed at which the analysis can be performed [83,98]. A BDD is a form acyclic, directed graph. All BDD paths will terminate at either a "0" state (system works), or a "1" state (system failure), referred to as "terminal vertices" [92]. Non-terminal branches in the vertices represent the basic events of the fault tree, represented as "nodes" in the BDD. If a path ends at a "1" state, then it is a cut set of that Top Event. However, if the path ends at a "0" vertex, then the system is said to be in the "working" state.

Before the BDD is constructed, the "variable ordering" must be determined. As an example, a fault tree with basic events "X1", "X2" and "X3" could be ordered as X1<X2<X3, X3<X2<x1, etc. The variable ordering can affect the computational efficiency of the model analysis. Different methods have been explored in the literature to determine variable ordering [83,99], however that is beyond the scope of this paper. Computationally, BDDs are usually solved using the "If-Then-Else (ITE)" method [89], based on "Shannon's Theorem" or "Shannon's Decomposition" [100,101]. In common terms:

$$\text{ITE(X, f1, f2)} = Xf_1 + \bar{X}f_2 \tag{3}$$

Implies [89,98]:

If X1 fails:

Consider f1

Else:

Consider f2

In terms of BDDs, this is given as X = ITE(X, 1, 0). This process will continue until all variables have been converted, in order of the variable ordering, and will end with all the "1" and "0" vertices being determined. This conversions are determined by the use of either "AND" or "OR" gates, as described in the literature [98,102] :

Consider two arbitrary basic events, "G" and "H":

G = ite(x, g1, g2)

H = ite (y, h1, h2)

The conversions of the fault tree to the BDD occurs based on the variable ordering:

$$\text{If } (x<y) \rightarrow \text{ite}(x, g1\hat{}H, g2\hat{}H) \quad (4)$$

$$\text{If } (x = y) \rightarrow \text{ite}(x, g1\hat{}h1, g2\hat{}h2) \quad (5)$$

Where, "^" is taken as either the "OR" or "AND" operator. Both cases are considered below:

When "^" = "OR":

0^G = G, 1^G = 1

When "^" = "AND":

0^G = 0, 1^G = G

The MCS are the failure events on path(s) that end at a "1" terminal. The final BDD is said to represent the structure function of the fault tree, and therefore is often referred to as the "Structure Function BDD" (SFBDD) . An example of what a generic SFBDD looks like is seen in Figure 35 [102].A second example, showing the BDD representation of "OR", "AND" and a "COMBINATION" (2oo3 Voting) gates is given in Figure 36 [103].

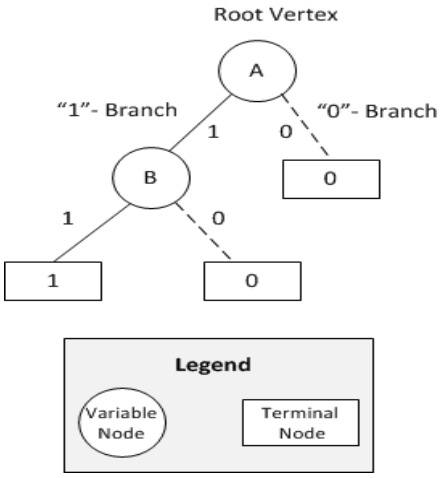


Figure 35: Example of a Generic SFBDD

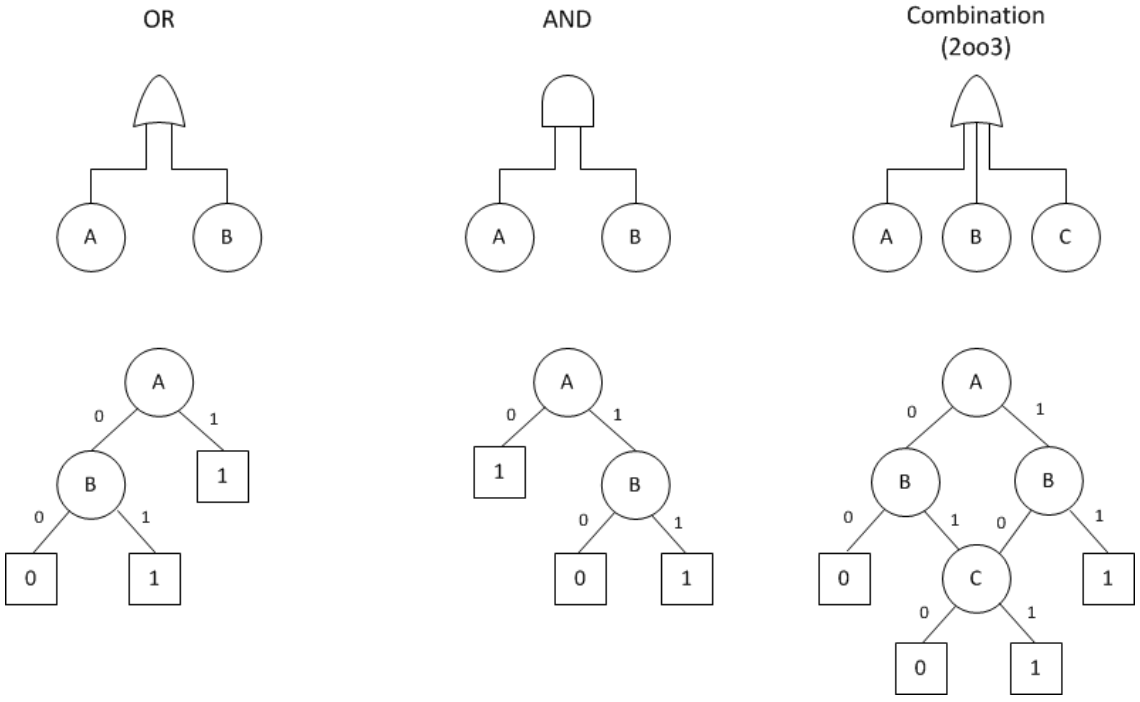


Figure 36: BDD Representations of Common Fault Tree Logic Gates

An example of the FTA → BDD conversion, using the fault tree shown in Figure 37, is given below [104].

.

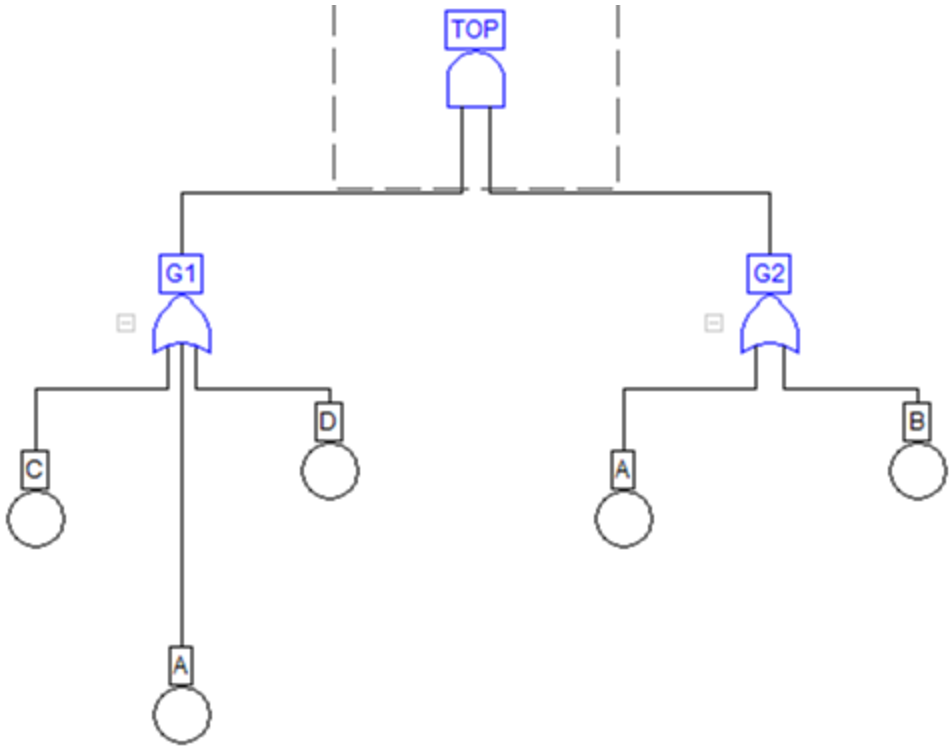


Figure 37: Example Fault Tree for BDD Demonstration

Employing the "ite" method, a "top-down left-right" path through the fault tree was used to determine the variable ordering, resulting in an ordering of c < a < d < b. Employing Equations 3-5 for the gates "G1" and "G2" (both "OR" gates") yields:

G1 = c + a + d

G1 = ite(c,1,0) + ite (a,1,0) + ite(d,1,0)

G1 = ite(c,1,ite(a,1, ite(d,1,0)))

G2 = a + b

G2 = ite (a,1,0) + ite(b,1,0)

G2 = ite (a,1,ite(b,1,0))

The top gate, denoted as “TOP” is an “AND” gate, with gates “G1” and “G2” as its inputs. The top gate is represented as:

TOP = G1*G2

TOP = ite(c,1,ite(a,1, ite(d,1,0))) * ite (a,1,ite(b,1,0))

TOP = ite(c, ite(a, 1, ite(b,1,0)), ite(a,1,ite(d,ite(b,1,0),0),0)))

The function for the “TOP” gate seen above determines the construction of the corresponding BDD for the fault tree seen in Figure 37. The resulting BDD is shown in Figure 38 [104].

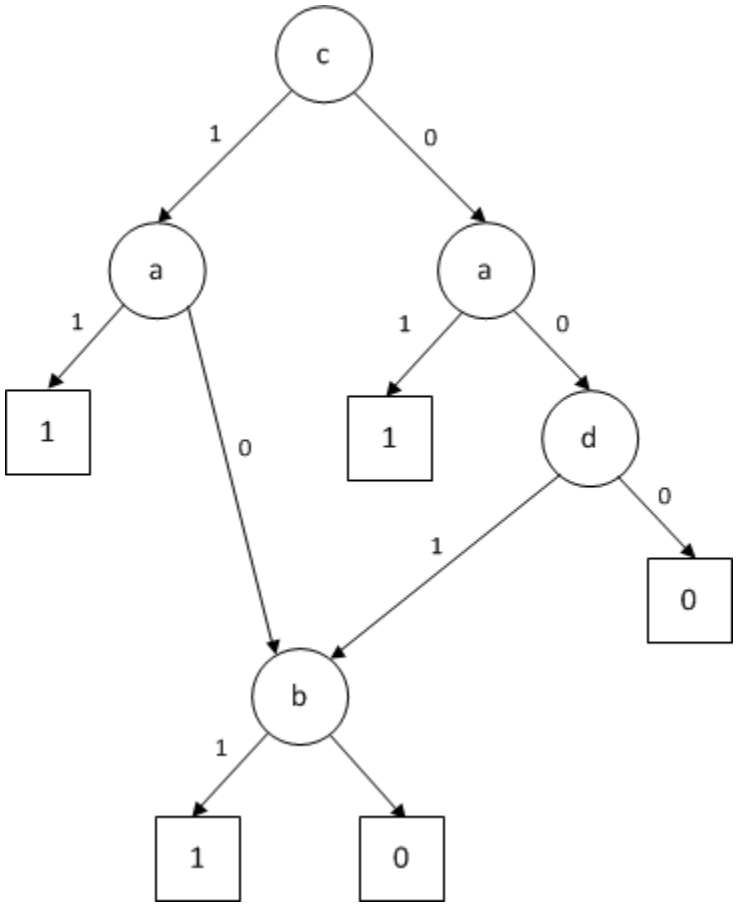


Figure 38: Resulting BDD for the Fault Tree from Figure 37

2.3.1.6 Qualitative Fault Tree Analysis (Non-Coherent Fault Trees)

The analysis methods discussed in sub-section 2.3.1.5 considered only coherent fault trees, and modifications to those methods need to be made, in order to apply them to the analysis of non-coherent fault trees. The reasoning for those modifications, and the methods for performing qualitative analyses of non-coherent fault trees are explored in this sub-section.

Consider the fault tree shown in Figure 39 [92]. This fault tree represents the ignition of a gas release in a gas transport system.

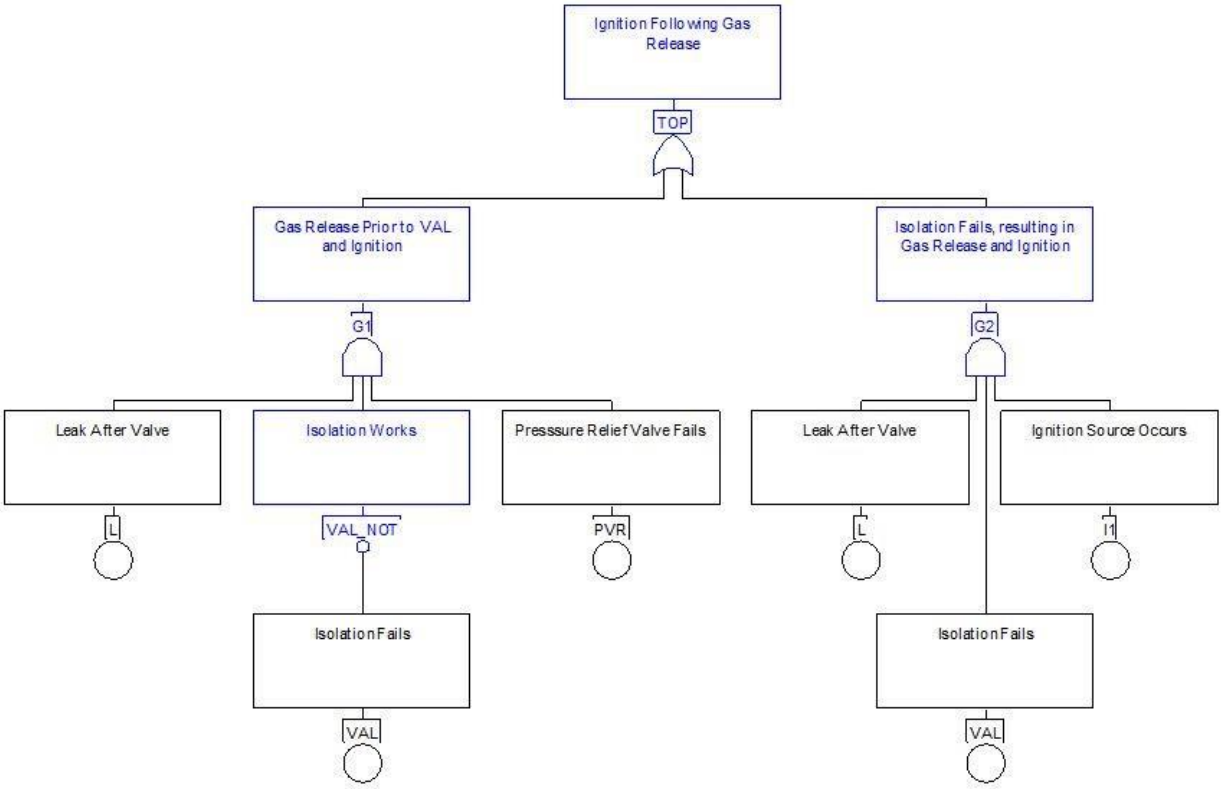


Figure 39: Example of a Non-Coherent Fault Tree

If a gas leak happens in the part of the gas transport system after the Isolation Valve (VAL), a gas detection system would close the isolation valve to prevent a gas build-up in an area where an ignition source (I1) could occur. If the valve works correctly and the gas is isolated, the pressure due to the gas would burst the pipe, and a separate leak will occur, before the isolation valve. It is assumed that there

is a second, permanent ignition source (I2, not shown in the fault tree), gas ignition would be prevented by shunting the gas flow elsewhere, using a Pressure Relief Valve (PRV). If the gas leak happens before the Isolation Valve, it was assumed that the ignition of the gas would always occur.

MOCUS Method for Non-Coherent Fault Trees

Applying the MOCUS algorithm from subsection 2.3.1.5 would result in Table 7. As this is a simplistic fault tree, the MOCUS methods returns two MCS, one from the expansion of the "AND" gate "G1", and the other from the expansion of the "AND" gate "G2".

Table 7: MOCUS Algorithm for Non-Coherent Fault Tree

Step	1	2	3
	G1	L, $\overline{\text{VAL}}$, PVR	L, $\overline{\text{VAL}}$, PVR
	G2	G2	L, I1, VAL

According to the MOCUS algorithm, there are two PIs (Prime Implicants, now that the fault tree is non-coherent). They are {L, $\overline{\text{VAL}}$, PVR} and {L, I1, VAL} However, there is actually a third PI, which is {L, I1, PVR}. In relation to the gas transport system, that third PI represents the case where a leak occurs, the Pressure Relief Valve fails, and an Ignition Source is present in the section after the Isolation Valve. If those three events occur, then the state of the Isolation Valve (Works/Fails) does not matter. In order to identify these additional PI(s), the "Consensus Law" must be applied.

Consensus Law

In a non-coherent fault tree, the Consensus Law is applied to the MCS identified using conventional methods (such as the MOCUS algorithm). The Consensus Law is given in Equation 6 [89,92,96]:

$$AX + \bar{A}Y = AX + \bar{A}Y + XY \quad (6)$$

The Consensus Law would be applied to all sets of PIs that contain both a normal and negated literal. This would result in the complete set of PIs. In the case of simple fault trees, this method is relatively easy to apply, however it can become computationally demanding for more complex fault trees [89]. In the case of the example for the fault tree in Figure 39, the Consensus Law would be applied to the two returned PIs, since those are PI pairs with a normal and negated literal. The application of the Consensus to this case is performed as follows [92]:

$$TOP = L*\overline{\mathbf{VAL}}*PVR + L*I1*VAL \quad (7)$$

$$TOP = L*(\overline{\mathbf{VAL}}*PVR + I1*VAL) \quad (8)$$

The Consensus Law is applied to Equation 8:

$$\overline{\mathbf{VAL}}*PVR + I1*VAL = \overline{\mathbf{VAL}}*PVR + I1*VAL+I1*PVR \quad (9)$$

Plugging the results of the Consensus Law back into the "TOP" function yields:

$$TOP = L*(\overline{\mathbf{VAL}}*PVR + I1*VAL+ I1*PVR) \quad (10)$$

$$TOP = L*\overline{\mathbf{VAL}}*PVR + L* I1*VAL + L* I1*PVR \quad (11)$$

This includes all three PIs. In general PIs will be of higher order (more basic events) than MCS. Additionally, due to the Consensus Law, there are often far more PIs than there would be MCS [92].

Additional Qualitative Methods for Non-Coherent FTA

Additional methods may be applied to non-coherent FTA, if needed. In non-coherent fault trees where gates such as "NOT", "NOR", "NAND" and "XOR" are used, they can be "Pushed-Down" the fault tree, so that more traditional FTA methods may be applied. Alternatively, another method is to simply assume an approximate version of the fault tree without the "NOT"-type logic, assuming the loss of accuracy is not significant.

Pushing Down

If “NOT” gates (including “NAND”, “NOR”, “XOR” etc.) are present in the fault tree, they must be removed in some fashion. This can be done by “pushing down” the “NOT” logic, through the use of De Morgan’s Laws [89]:

$$\overline{(\boldsymbol{A} + \boldsymbol{B})} = \overline{\boldsymbol{A}} \cdot \overline{\boldsymbol{B}} \quad (12)$$

$$\overline{(\boldsymbol{A} \cdot \boldsymbol{B})} = \overline{\boldsymbol{A}} + \overline{\boldsymbol{B}} \quad (13)$$

The “NOT” logic is “pushed down” the fault tree from top to bottom, until all “NOT” gates are removed, leaving only “AND” and “OR” gates, as well as negated Basic Events. The same process is applied to “NOR”, “NAND” and “XOR” gates by transforming them into their composite “NOT”, “OR” and/or “AND” logic, and then pushing the equivalent “NOT” logic down the fault tree. These transformations are seen in Figure 40 [94].

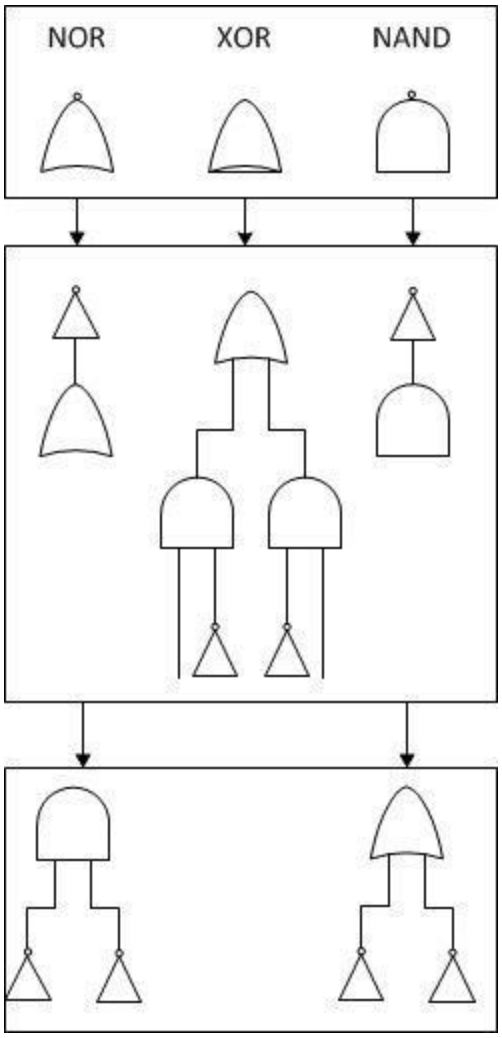


Figure 40: Equivalence Library for the transformation of "NOR", "XOR" and "NAND" Gates

Once all "NOT"-type gates have been removed, the complete set of PIs can be determined using "Top-Down" or "Bottom-Up" methods, including the application of the "Consensus Law".

Coherent Approximation

The methods discussed earlier in this sub-section will produce the full set of PIs, however it can be a very computationally intensive procedure, especially when the fault trees become large or complex [89]. To avoid this issue, the "Coherent Approximation" can be applied to non-coherent fault trees [89,92,96]. In this approximation, it assumes that all of the component "working" states are "True", and therefore allows for the removal of any negated logic in the Basic Events. This permits traditional fault tree

analysis methods to be applied. In this case, only the positive section of PI sets will be identified, which are sometimes referred to as “minimal p-cuts” [89]. In order to apply this approximation, “NOT”-type logic gates would have to first be pushed down the fault tree, leaving only the negated Basic Events. As an example, if we set logic Gate “G” as $G = a \cdot b \cdot \bar{c}$, then the coherent approximation would become $G = a \cdot b$, as the negated term is discounted. This method can reduce the computational burden of determining the PIs/minimal p-cuts, however depending on the probability that was assigned to the negated Basic Event, the accuracy of the TE calculation may be reduced [89].

BDD Analysis of Non-Coherent Fault Trees

If BDDs are selected for the analysis of a non-coherent fault tree, then there are several methods that can be applied, as discussed briefly in this sub-section. It should be noted that the “Coherent Approximation” can be applied to BDDs as well, so it is not discussed further in this sub-section [89]. For a simple fault tree/SFBDD, the PIs may be determined manually using the following method. The complement of a variable is given using the ITE method as [92]:

$$\bar{\boldsymbol{X}} = \boldsymbol{ITE}(\boldsymbol{X}, \mathbf{0}, \mathbf{1}) \tag{14}$$

The BDD would then be constructed as normal. To produce the PIs, the nodes are searches individually, in a bottom-up method, considering three possibilities;

a.) PIs containing “*X*”
b.) PIs containing “$\bar{X}$”
c.) PIS without the “*X*” literal

A full description and example using this method is found in the literature [92]. Referring to the non-coherent fault tree seen in Figure 39, it can be represented as a BDD, using the following process [92]:

Normally, each of the basic events in a fault tree would be represented in this manner:

$$\text{VAL} = \text{ite}(\text{VAL}, 1, 0) \tag{15}$$

However, in the case of the negated logic, the compelement of that node can be represented by:

$$\overline{\mathbf{VAL}} = \text{ite}(\text{VAL}, 0, 1) \tag{16}$$

The rest of the BDD construction process then proceeds as normal. The two gates, “G1” and “G2” are both “AND” gates, so applying the rules set out in sub-section 2.1.3.5 will yield the following transformations:

G1 = ite(L, 1, 0)*ite(VAL, 0, 1)*ite(PRV, 1, 0)

G1 = ite(L, ite(PRV, ite(VAL, 0, 1), 0), 0)

G2 = ite(L, 1, 0)*ite(VAL, 1, 0)*ite(I1, 1, 0)

G2 = ite(L, ite(I1, ite(VAL, 1, 0), 0), 0)

The top gate, “TOP”, is an “OR” gate, so it is represented as:

TOP = G1 + G2

TOP = ite(L, ite(PRV, ite(VAL, 0, 1), 0), 0) + ite(L, ite(I1, ite(VAL, 1, 0), 0), 0)

TOP = ite(L, ite(PRV, ite(I1, 1, ite(VAL, 0, 1)), ite(I1, ite(VAL, 1, 0,) 0), 0)

From the above function for the “TOP” gate, the corresponding BDD is created:

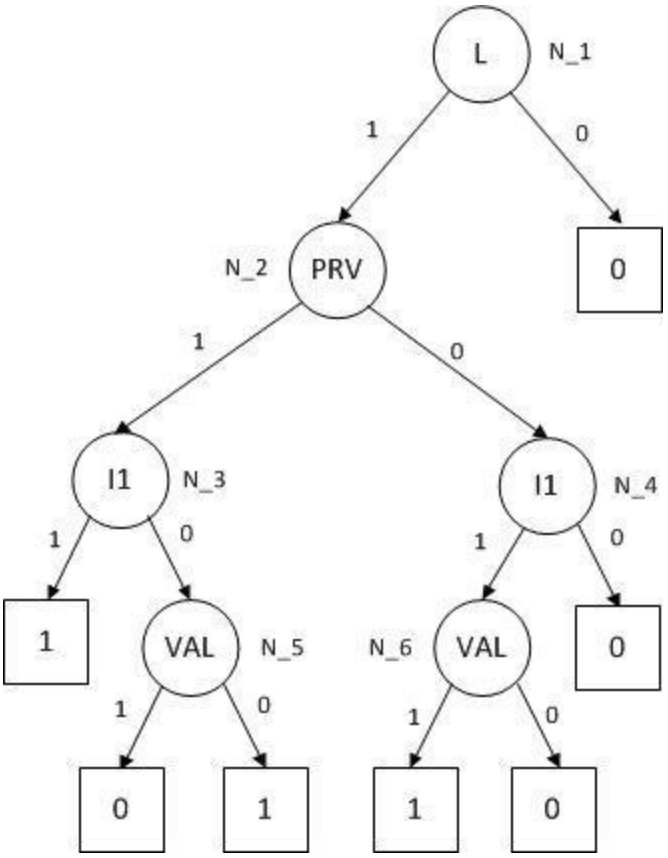


Figure 41: BDD Representation of the Example Non-Coherent Fault Tree

To ascertain the MCS from the BDD shown in Figure 41, that BDD will be traversed in a bottom-up manner, using the three rules described at the beginning of this sub-section. The results of that method is shown in Table 8 [92]. In this case, each node (*N*) is labelled in Figure 4, to make the process easier to follow.

Table 8: PIs Determined from BDD

Node (N)	**Prime Implicants**
N_1	{L*$\overline{\text{VAL}}$*PVR}, {L* I1*VAL}, {L* I1*PVR}
N_2	{I1*PRV}, {I1*VAL}, {$\overline{\text{VAL}}$*PRV}, {VAL*I1*$\overline{\text{PRV}}$,} (Not a PI)
N_4	{VAL, I1}
N_3	{I1}, {$\overline{\text{VAL}}$}, {$\overline{\text{I1}}$*$\overline{\text{VAL}}$} (Not a PI)
N_5	{$\overline{\text{VAL}}$}
N_6	{VAL}

At the end of that process, the same three PIs are returned, as seen in the second row of Table 8.

Additional Methods for Non-Coherent BDD FTA

Additional methods were considered, with each of these methods containing their own benefits and drawbacks, as discussed in the literature [96].

Ternary BDDs (TDD)

The TDD method includes a third branch from every node (on top of the traditional "1" and "0" branch). This third branch is referred to as the "consensus branch", and denotes the additional PI sets that are uncovered by applying the "consensus law" discussed in Sub-section 2.3.1.6. After the BDD is constructed, non-minimal paths are removed using minimization algorithms [96].

Meta-Products BDDs

In this method, every component ("X"), is assigned two variables, $P_X$and S_X. $P_X$represents relevancy, and S_X represents the form of relevancy (repair relevant or failure relevant).The "Meta-Product" MP(π), is then the "intersection of all the system components according to their relevancy in the system state", with π denoting the PI set within meta-product MP(π) [89,96]. Once solved, the Meta-Products BDD will always be minimal, therefore it will solve for the exact PIs.

Zero-Suppressed BDDs (ZDDs)

ZDDs represent a more computationally efficient method for applying BDDs. ZDDs use reduction rules to simplify BDD calculations. These rules involve eliminating all nodes which have their "1" branch connecting to a terminal vertex in state "0". That branch is then connected to the BDD structure beneath the "0" branch of the eliminated node [96]. In terms of non-coherent FTA, the ZDD will include labelling of the nodes with the working/failed states of the Basic Events, and then decompose the PIS sets based on the occurrence of certain states of each Basic Event. ZDDs represent a computationally efficient method of determining the PIs from non-coherent fault trees [96,105].

Labelled BDD (L-BDD)

L-BDDs involve labelling each node based on one of three possible variable types. These types are "Single Form Positive" (SP), where the variable only appears in its normal form, "Single Form Negative" (SFN), where the variable only appears in the negated form, and "Double Form" (DF), where the variable

appears in both normal and negated forms [96,105]. The nodes belonging to different variable types involve different algorithms for analysis, so correct variable labelling is important. The DF nodes are of special note, which add the most to computationally intensity of the analysis, making it prudent to minimize the number of DF nodes in the L-BDD [96,105].

2.3.1.7 Quantitative Methods for Fault Tree Analysis

The methods discussed in sub-sections 2.3.1.5 and 2.3.1.7 deal with the qualitative analysis of fault trees, however they do not directly consider any quantitative behaviour. Once the PI/MCS have been determined, the probabilities of each PI/MCS, and the probability of the Top Event (TE), may also need to be calculated. Several methods have been formulated to perform the quantitative calculations used in FTA and DFM, with the various methods incorporated into the software tools.

MCS/PI and Top Event Probabilities

Regardless of the methodology used to determine the MCS or PIs, (i.e. DFM or FTA), there are similarities in the quantitative calculations. The probability of the individual PI or MCS is found by [93]:

$$\boldsymbol{MCS_j = \cap_{i=1}^{n} X_i^{(j)} = \prod_{i=1}^{n} X_{i=1}^{(j)}} \quad (17)$$

Where $X_i^{(j)}$ is the Boolean variable for the i^{th} primary event in the j^{th} minimal MCS/PI, and n is the number of primary events in that MCS/PI. Once the PI/MCS and their probabilities has been determined, there are several ways to calculate the Top Event probability that are found in the literature. The simplest method is to take a straight sum of all MCS/PI probabilities, sometimes referred to as the "Rare Event", and is given by [93,98,106]:

$$\boldsymbol{TE_{SUM} = \cup_{j=1}^{m} MCS_j} \quad (18)$$

A second common method is referred to as the "Upper Bound Approximation" or "Minimal Cut Set Upper Bound" (MCSUB), performed using the formulation [93,98,106] :

$$\boldsymbol{TE_{MCSUB} = \left(1 - \prod_{j=1}^{n} \left(1 - P\left(MCS_j\right)\right)\right)} \quad (19)$$

Here $P(MCS_j)$ represents the probability of the *j*th MCS/PI, while *n* signifies the total number of MCS/PI.

While the methods shown in Equation (6) and (7) are computationally efficient, they have the potential to overestimate the Top Event probability, as they do not account for the mutually exclusiveness of the MCS/PI. The exact TE probability is found using the Inclusion-Exclusion principle [98,106]:

$$\boldsymbol{TE_{Exact}} = \cup_{\boldsymbol{j=1}}^{\boldsymbol{m}} \boldsymbol{MCS_j} - \sum_{\boldsymbol{J=2}}^{\boldsymbol{N}} \sum_{\boldsymbol{I=1}}^{\boldsymbol{)-1}} \boldsymbol{P}(\boldsymbol{C_j} \cap \boldsymbol{C_J}) + \cdots + (-\boldsymbol{1})^{\boldsymbol{N_C}} \boldsymbol{P}(\boldsymbol{C_1} \cap \boldsymbol{C_2} \cap \ldots \cap \boldsymbol{C_{N_C}}) \quad (20)$$

where N_C represents the number of MCS/PI. Although the method shown in Equation 12 will account for the mutual exclusive behaviour of MCS/PI, it can be computationally intensive to perform. Therefore, the "Sum" or "MCSUB" methods are often implemented to give approximations of the TE probability. In terms of system reliability, overestimating the TE probability would be less serious than underestimating the TE probability. In general, the TE calculations between these methods compare as [98]:

$$TE_{SUM} \geq TE_{MCSUB} \geq TE_{Exact} \geq TE_{Lower}$$

where, TE_{Lower} represents the lower bound on the TE probability. As this method could underestimate the TE probability, it is not directly used in this thesis, although it is considered during the TE_{Exact} computation procedure. It should be noted that for systems with basic events that have small probabilities of failure (i.e. Probability of Failure << 1), the difference between the results from the "SUM" approximation and "Rare Event" approximation is negligible [107].

2.3.2 Dynamic Flowgraph Methodology

The reliability assessment is performed using the Dynamic Flowgraph Methodology (DFM), and it has seen use in both the nuclear and aerospace industries. This sub-section will provide for detailed information into the background, theory, calculations and software tools for DFM analysis.

2.3.2.1 Introduction and Background for DFM and Dymonda

This methodology represents the system being analyzed with a directed graph (digraph) model. This model is similar to that of a signal flow graph, which is used in control systems engineering. This model will explicitly show the timing and the cause-and-effect relationships between states and parameters that will best describe the system. After the model is built, it can be analyzed by automated inductive and deductive algorithms built into the methodology [8,11,12,15]. The inductive procedures can be applied to the model to analyze how a certain combination of basic component states can result in different possible event sequences and the ensuing system-level states. The deductive process works in the opposite fashion, where it is applied to identify how system states (possibly representing certain failure or success conditions) can be produced through sequences and combinations of the basic component states. DFM is also able to incorporate time dependency into the system, allowing for both static and dynamic (time dependant) models.

The DFM deductive analysis will return what are known as Prime Implicants (PI), which are sets of occurrences that would cause the Top Event (possibly a failure event). Alternatively, the results produced by the inductive analysis are referred to as Sequences. When discussing software/programming failures, any Prime Implicant that does not include some form of component failure would indicate a programming issue, and can therefore uncover hidden faults in the system [24,93].

One other facet of DFM/Dymonda is the ability to include probabilities and uncertainty calculations. The probability of each state of each node in the model can be entered (by the user or calculated by the program), to determine the probability of each Implicant or Sequence occurring. The program will automatically calculate the probabilities, and display the total probability of each Implicant or Sequence in descending order. Afterwards, the results can be pruned, to eliminate events with a probability of

occurring below a certain user-defined threshold. The inclusion of probabilities allows for exact quantification calculations, to determine the exact probability of the Top Event occurring.

2.3.2.2 DFM Theory

DFM shares certain similarities, as well as several differences with traditional methods (i.e. static, binary methods) used in reliability modelling/analysis. DFM has been touched on briefly earlier in this thesis, however this sub-section will provide more detailed information behind the theory of DFM, as well as ways to perform a DFM analysis on a model of an MVL system. In the case of DFM theory, the following definitions are important:

Disjunction:

The logical "OR" (union) operator, generally represented as "+", "∪", or "∨".

Conjunction:

The logical "AND" (intersection) operator, generally represented by "*", "·", "∩" or "∧".

DFM Prime Implicants

DFM is based on the use of Multi Valued Logic (MVL, also referred to as Many Valued Logic), as opposed to the binary logic of methods such as fault trees. The deductive analysis is performed similarly to a standard fault tree analysis, as both methods start with a top event, and work backwards in time to uncover root conditions. For standard Fault Tree analysis, the official notation for a Minimal Cut Set (MCS) is given (as the conjunction of primary events) by [14,93]:

$$\mathbf{PI_j} = \bigcap_{i=1}^{n} \boldsymbol{X_i^{(j)}} \tag{21}$$

Where $X_i^{(j)}$ is the Boolean variable for the i^{th} primary event in the j^{th} PI, and n is the number of primary events in that cut set. Setting TOP to be the Boolean variable for the top event, which takes the values of either True (T) or False (F) that the top event can be shown in a disjunctive form by [14,93]:

$$\boldsymbol{TOP} = \cup_{\boldsymbol{j=1}}^{\boldsymbol{m}} \boldsymbol{PI_j} = \mathbf{1} - \cap_{\boldsymbol{j=1}}^{\boldsymbol{m}} (\mathbf{1} - \boldsymbol{PI_j}) \quad (22)$$

Where m is the number of PI. The use of nodes by DFM to represent variables (voltage, current, temperature, pressure, etc), and software states means that binary logic is inadequate, and multi-valued logic is implemented instead. Alternatively, 1 and 0 could be used in place of True and False, respectively. Using MVL, each node can be discretized into an arbitrary number of states. During the analysis, the intermediate transition tables are generated (comparable to a MVL fault tree), which contain non-binary primary events. However, these events can be expressed in a comparable binary form by using "Binary Algebra with Restrictions on Variables"[108] .

Additionally, the disjunctive form found through the Boolean reduction does not have to contain all Prime Implicants. The Prime Implicants are said to be unique and finite. Moreover, unlike with coherent Fault Trees, the MVL tree is said to be "non-coherent", as variables can change from higher to lower states, as well as lower to higher states, and higher states are not always considered to be increasingly faulted when compared to lower states

More generally, DFM Prime Implicants are considered to be the MVL variant of a minimal cut set used by fault trees, but are more difficult to find. In a DFM analysis, the goal is to determine the PIs that will cause the Top Event to occur. A set of Prime Implicants that are the logical analog of the TOP function (i.e. the set of PIs that result in the Top Event) are referred to as a base, of which two distinct types exist. The "Complete Base (CB)" being the set of all of the PIs that will result in the Top Event, and the "Irredundant Base (IB)", which would not be a base if any PI was removed [93]. It should be noted that the IB and CB are generally different, and only the CB will be unique [93]. General definitions for Implicants and Prime Implicants are given below, with the more technical definitions being seen in Table 5.

Implicant/Implicant Set:

A combination of basic events (success or failure) which produces the top event. An Implicant/Implicant set can be considered as the MVL equivalent of a "Cut Set" [98].

Prime Implicant/Prime Implicant Set:

A combination of basic events (success or failure) which is both necessary and sufficient to cause the Top Event. A Prime Implicant/Prime Implicant set can be thought of as the MVL equivalent of a "Minimal Cut Set" [98].

Introduction to DFM Analysis Procedures

In the literature, there have been several methods considered for determining the CB. These include the "Tabular Method" [93,109], the "Nelson Method"[93,109,110] , and the "Method of Generalized Consensus" [93,109,111], based on the Quine method [112,113]. Of these methods, the "Tabular Method" is performed graphically, making it a realistic choice only for small, simple systems [93,109]. However, both the "Nelson Method" (especially when factoring methods are involved [114]) and "Method of Generalized Consensus" can be implemented into software codes, allowing them to solve much more complex systems [93,115]. An example from the literature, regarding the process for applying the "Tabular Method" and "Nelson Method" will be provided in this sub-section. As the "Method of Generalized Consensus" was the method employed during this research program, it will be discussed in more detail later on in this sub-section. In the examples given in this sub-section, it should be noted that the "·" refers to the "AND" operator, and the "+" refers to the "OR" operator.

Tabular Method

Consider the MVL tree, shown in Figure 42 [109]. It contains three variables; "X", "Y", and "Z". Variables "X" and "Y" are both discretized into three states; "0", "1" and "2", while "Z" is discretized into four states; "0", "1", "2", and "3". This tree contains two Operators, designated as "Op4" and "Op5", seen in Figure 43 [109]. The output of "Op5" is denoted as "Q" on the MVL tree. The Top Event ("TOP"), comes directly from "Op4", and as such is also discretized into three states; "0", "1" and "2", i.e. TOP = {0, 1, 2}.

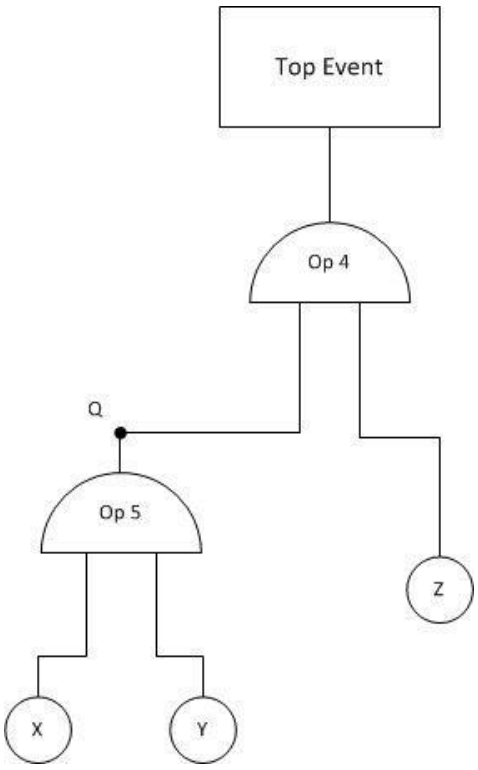
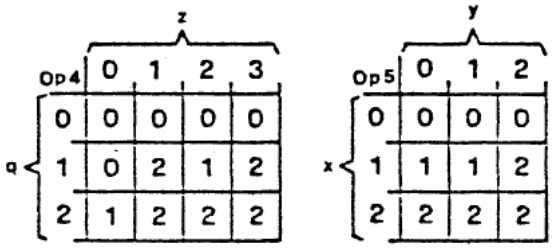


Figure 42: Example of a Simple MVL Tree



Op4	z=0	z=1	z=2	z=3
q=0	0	0	0	0
q=1	0	2	1	2
q=2	1	2	2	2

Op5	y=0	y=1	y=2
x=0	0	0	0
x=1	1	1	2
x=2	2	2	2

Figure 43: Operators (Op4 and Op5) For the Example MVL Tree (© 1985 IEEE)

Determining the PIs using the "Tabular Method" involves breaking down the MVL "Top" function into a graphical representation in Cartesian space, from its disjunctive form. That Cartesian representation is seen in Figure 45 [109], and is similar to that of a Karnaugh Map [116]. Also seen in that figure, are the corresponding PIs to the different values of the "TOP" function. As the total TOP function contains three states, each of those states (γ), will have its own truth function (f_γ), determined by the PIs. In the case of the "Tabular Method", it is seen that with a number of variables, *m*, the PIs for each truth function, f_γ, will be corresponding to Maximum Rectangular Coverings (MRCs), of dimension *m* [109].

Maximum Rectangular Covering (MRC):

A Maximum Rectangular Covering is defined as consisting "…of the largest rectangular arrangement of cells of the operator, where cells contain values belonging to I_i" [109]. In this case, an analyst would start with the TOP function, and from that function, select the subset, I_{Top}, corresponding to the truth function $f_{I,TOP}$. These sets, denoted as I_i, must be covered by at least one of the MRCs. Each MRC is obtained from a conjunction of two literals. The rectangular arrangement of the MRCs must be kept intact, however the MRCs can be composed of arrangements of non-adjacent cells.

As an example, consider "Op 4" in Figure 43. The overall truth function, "TOP", contains states "0", "1" and "2". If a set $I_A = \{0\}$ is selected, the MRCs corresponding to that set are Q^0 and $Q^{01} \cdot Z^0$. Alternatively, if the set $I_B = \{1, 2\}$ is selected, then the corresponding MRCs are Q^2 and $Q^{12} \cdot Z^{123}$. An example of an MRC composed of non-adjacent cells occurs when the set $I_c = \{2\}$ is selected. One of the resulting MRCs is $Q^{12} \cdot Z^{13}$, composed of the arrangement of the non-adjacent cells $Q^{12} \cdot Z^1$ and $Q^{12} \cdot Z^3$. The graphical representation of these MRCs is seen in Figure 44 [109].

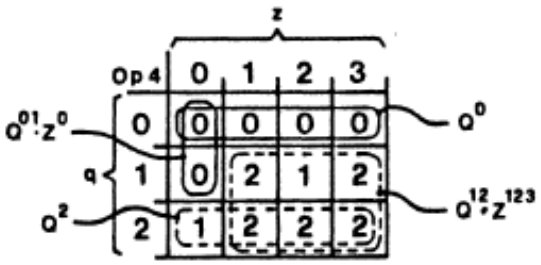


Figure 44: Graphical Example of Select MRCs (© 1985 IEEE)

Karnaugh Map (K-Map):

A Karnaugh Map, or K-Map is a graphical method for simplifying Boolean logic expressions. The necessary Boolean logic information is taken from a truth table, and ordered into a two-dimensional grid. Inside that grid, the position of each cell represents one input combination, while each cell value denotes the corresponding output value. Optimal groupings of the Boolean logic values ("0s" and "1s") are found, to write the reduced (simplest) Boolean logic function that represents the overall truth table [116].

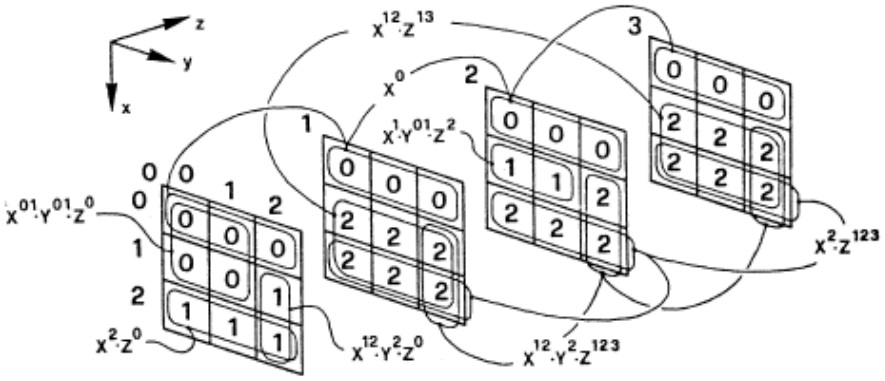


Figure 45: Graphical (Cartesian) PI Determination Using the "Tabular Method" (© 1985 IEEE)

The truth functions for the three values of TOP ("0", "1", and "2"), are denoted as (f_0), (f_1) and (f_2), respectively. These truth functions are a disjunction of the PIs, with the PIs obtained from Figure 44. These truth functions and corresponding PIs are found to be [109]:

$$\boldsymbol{f_0 = X^{01} \cdot Y^{01} \cdot Z^0 + X^0} \quad (23)$$

$$\boldsymbol{f_1 = X^1 \cdot Y^{01} \cdot Z^2 + X^2 \cdot Z^0 + X^{12} \cdot Y^2 \cdot Z^0} \quad (24)$$

$$\boldsymbol{f_2 = X^2 \cdot Z^{123} + X^{12} \cdot Y^2 \cdot Z^{123} + X^{12} \cdot Z^{13}} \quad (25)$$

In Equations 23-25, the superscript values denote the states of the variable (e.g. "X^2" denotes state "*2*" of variable "*X*").

It is noted from the above example, this method would not be a realistic way to solve for the PIs of a large or complex system [109], and a such will not be considered further in this thesis.

Nelson Method

The "Nelson Method" (or "Nelson Algorithm") was originally intended to be used for Boolean logic [110], however it has been shown in the literature that it is easily extended to be applicable to MVL logic as well [109]. The process for the "Nelson Method" starts with the disjunctive form of the selected truth function. That truth function is negated, and the resulting conjunctive form is expanded. The expanded conjunctive form is then simplified using logic reduction rules, such as the "Absorption Rule", which removes subsuming values. Additional logic simplifications, such as removing zero products ($X \cdot \bar{X} = 0$)

and repeated literals ($X \cdot X = X$) occur if needed. Factoring the expanded conjunctive form may also simplify the process [114]. The next step is to negate the simplified conjunctive function, to turn it back into its disjunctive form. Further simplification will then leave the original truth function, defined by the disjunction of the PIs. An example of this is seen in the literature, and presented below [109]:

A truth function, equivalent to the $\boldsymbol{f_2}$ truth function of Equation 25 is used as the example. However, in this case, the truth function is given as a disjunction of Implicants only, not PIs, so the actual PIs must be determined. In this example, the $\boldsymbol{f_2}$ truth function is shown in Equation 26.

$$\boldsymbol{f_2 = X^{12} \cdot Z^{13} + X^{2} \cdot Z^{123} + X^{1} \cdot Y^{2} \cdot Z^{2}} \tag{26}$$

Negating f_2 yields:

$$\boldsymbol{\overline{f_2} = \left(X^{0} + Z^{02}\right) \cdot \left(X^{01} + Z^{0}\right) \cdot \left(X^{02} + Y^{01} + Z^{013}\right)} \tag{27}$$

Expanding and then simplifying the conjunctive form produces the simplified disjunctive form gives:

$$\boldsymbol{\overline{f_2} = X^{0} + Z^{0} + X^{01} \cdot Y^{01} \cdot Z^{02}} \tag{28}$$

Negating Equation 28 returns the truth function back into a conjunctive form:

$$\boldsymbol{\overline{\overline{f_2}} = f_2 = X^{12} \cdot Z^{123} \cdot \left(X^{2} + Y^{2} + Z^{13}\right)} \tag{29}$$

Expansion and simplification of Equation 29 produces a disjunctive form, corresponding to the disjunction of PIs:

$$\boldsymbol{f_2 = X^{2} \cdot Z^{123} + X^{12} \cdot Y^{2} \cdot Z^{123} + X^{12} \cdot Z^{13}} \tag{30}$$

It is seen that Equation 30 corresponds to the same truth function and Prime Implicants as seen in Equation 25. During the negation process, De Morgan's Laws (Equations 12 and 13) are applied. Additionally, the use of "NOT" logic in MVL can be slightly different than in the binary case. For example, consider the literal X^0. As "X" has been discretized into three states ("0", "1" "2"), then $\overline{X^0} = X^{12}$.

Applying one of De Morgan's Laws, namely Equation 13 to the first term of Equation 26 and negating it results in $\overline{\boldsymbol{X^{12} \cdot Z^{13}}} = \boldsymbol{X^{0} + Z^{02}}$.

While the "Nelson Method" does lend itself to a computational implementation of DFM, it was not the method used in this research program, and as such will not be discussed further.

Method of Generalized Consensus (Theory)

The original Quine implementation of this method considered binary logic; however it has been expanded to consider MVL systems too. The determination of the PIs is performed iteratively using two steps, "reduction" and "development" [109]. In the "reduction" step, the terms in the "critical transition table" are reduced using logic reduction rules. The rules that are considered are "absorption", "reduction", "merging" and "reduction-merging", with the binary and MVL formulations discussed in the literature [21,117].

In the "development" step, monomials are added to the (now reduced) set of implicants. These monomials are found through the merging of other couples of implicants, if merging is allowed by logic rules. These new monomials are referred to as the "consensus terms", and could be implicants or PIs. These two steps are repeated, until no more new consensus terms are generated, and at this point all consensus terms are also PIs. The total set of PIs will include the reduced implicants (now PIs) from the original "critical transition table", as well as the new PIs generated to make the consensus term(s) [109].

Decision Tables

The concept of a truth table is well-known and well-defined. These are mathematical tables used to calculate the output values of Boolean logic functions, based on all combinations of the input variables [118]. The truth table for a generic "AND" gate (Figure 46), is given in table 9.

Figure 46: Generic "AND" Gate

Table 9: Truth Table for a Generic "AND" Gate

Input		Output
A	B	A*B
0	0	0
0	1	0
1	0	0
1	1	1

A related concept to a truth table, is that of a "Decision Table" [117,119]. A decision able can be thought of as a more advanced truth table, and is a tabular representation of certain sets of "Conditions", "Actions", "Rules" and "Entries". These are defined as [119]:

Conditions:

Variables that are considered in order to obtain a decision

Actions:

The operations that must occur when a certain set of conditions is present.

Rules:

Particular sets of conditions and the resulting actions required by those conditions.

Entries:

Any additional information regarding an action or condition relevant to a certain rule.

These aspects are represented in a decision tables as [119]:

1.) Condition statement

2.) Condition Entry

3.) Action Statement

4.) Action Entry

5.) Rule

Those above aspects make up a theoretical decision table, separated by double line, as shown in Table 10, with the example of a decision table for a credit approval process shown in Table 11 [119]. In that example, if “Credit is OK”, then “Approve Order”. Alternatively, if “Credit is OK” is not true, but “Pay Experience Favourable” is true, then “Approve Order”. If neither is true, then the order is “Return to Sales”. It should be noted that with decision tables, the “-“ is generally used to denote a “Don’t Care” value. It should be noted that in the case of decision tables, one is not limited to binary logic, as in the case of decision tables. The entries in decision tables can take on any terms that the analyst requires.

Table 10: Aspects of Decision Tables

	Rule(s)
Condition Statement	Conditions Entry
Action Statement	Action Entry

Table 11: Example Decision Table for Credit Approval

	Rule 1	Rule 2	Rule 3
Credit “OK”	Y	N	N
Pay Experience “Favourable”	-	Y	N
Approve Order	X	X	-
Return to Sales	-	-	X

In the case of DFM, decision tables are used to represent the interactions between the process variables of the system. The credit check example shown in Table 11 was used to explain the design and

implementation of decision tables in a more accessible way. An example of how decision tables are used in DFM is seen later on in this sub-section, in Table 12.

Critical Transition Table

In the literature, methods have been developed to compute the PIs of a system, using the decision tables for individual components [21,117]. During the analysis in DFM, those component decision tables (represented by the decision tables that connect the nodes together) are merged into one "critical transition table" [21,117,120]. This merging is based on the structure of the DFM model, as well as the selected Top Event(s). Once this table is constructed, logical merging and absorption operations are used on the "critical transition table" to determine the complete base of PIs, based on the "Method of Generalized Consensus" [111–113].

5.1.1.1. Logic Reduction Operations

The logical reduction of the critical transition table is performed using four logical reduction operations, "Absorption", "Merging", "Reduction" and "Reduction-Merging". Most of these have a binary and MVL equivalent, which are discussed below.

Type-1 Absorption Rule:

A term, denoted as $\Phi_1$will subsume (absorb) a second term, denoted as Φ_2, in a decision table if both terms have the same outputs, and id every input event in Φ_2 is also seen in Φ_1. It this case, the longer Φ_1 term is cut from the table, and is said to be "absorbed" by the shorter Φ_2 term [113]. This logic operation is applicable to the binary and MVL case, without modification.

$$\begin{array}{cc|c} \boldsymbol{A} & \boldsymbol{B} & \\ \hline \boldsymbol{T} & - & \equiv \\ \boldsymbol{T} & \boldsymbol{T} & \end{array} \quad \begin{array}{cc} \boldsymbol{A} & \boldsymbol{B} \\ \hline \boldsymbol{T} & - \end{array} \tag{31}$$

Type-2 Merging Rule:

If two terms in a decision table are identical, except for just one input entry, and that entry in the two terms has opposite (logic) values, then the terms cab be merged [112]. For example, in the binary case $A\bar{B} + AB = A$, shown in Equation 32.

$$\begin{array}{cc|c} \boldsymbol{A} & \boldsymbol{B} & \\ \hline \boldsymbol{T} & \boldsymbol{F} & \equiv \\ \boldsymbol{T} & \boldsymbol{T} & \end{array} \begin{array}{cc} \boldsymbol{A} & \boldsymbol{B} \\ \hline \boldsymbol{T} & - \end{array} \tag{32}$$

This is easily extended to the MVL, with a small modification. If one considers *m* terms in a decision table that are identical, save for one *m*-input entry, and all possible states of *m* exist in there terms, then the *m* terms can be "merged". The MVL case, assuming *B* can have states {-1, 0, 1},a and if it is assumed *A* is in an arbitrary state *W*, is seen in Equation 33 [117].

$$\begin{array}{cc} \boldsymbol{A} & \boldsymbol{B} \\ \hline \boldsymbol{W} & \boldsymbol{-1} \\ \boldsymbol{W} & \boldsymbol{0} \\ \boldsymbol{W} & \boldsymbol{1} \end{array} \equiv \begin{array}{cc} \boldsymbol{A} & \boldsymbol{B} \\ \hline \boldsymbol{W} & - \end{array} \tag{33}$$

Type-3 Reduction Rule:

When considering binary logic, If two of the terms in a truth/decision table are the same, except just one input entry, then the larger of those two terms is obtained using that input variable, when it has opposite values (normal/complement) in those two terms. This rule is visualized in Equation 34 [117].

$$\begin{array}{ccc} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{C} \\ \hline \boldsymbol{T} & \boldsymbol{T} & \boldsymbol{T} \\ \boldsymbol{T} & \boldsymbol{F} & - \end{array} \equiv \begin{array}{ccc} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{C} \\ \hline \boldsymbol{T} & - & \boldsymbol{T} \\ \boldsymbol{T} & \boldsymbol{F} & - \end{array} \tag{34}$$

When considering the MVL, there are some modifications made to rule three in order for it to be applicable. In that case, if we consider the variable *B* to be a variable with *m* events (states), and assume that those states are {N, R, F}, then for the largest term to be reduced, all of the *m* states in variable *B* must be present in other comparable terms. The example for the MVL case is seen in Equation 35 [117].

$$\begin{array}{cccc} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{C} & \boldsymbol{D} \\ \hline \boldsymbol{W} & \boldsymbol{N} & \boldsymbol{F} & \boldsymbol{W} \\ - & \boldsymbol{F} & - & \boldsymbol{W} \\ \boldsymbol{W} & \boldsymbol{R} & - & - \end{array} \equiv \begin{array}{cccc} \boldsymbol{A} & \boldsymbol{B} & \boldsymbol{C} & \boldsymbol{D} \\ \hline \boldsymbol{W} & - & \boldsymbol{F} & \boldsymbol{W} \\ - & \boldsymbol{F} & - & \boldsymbol{W} \\ \boldsymbol{W} & \boldsymbol{R} & - & - \end{array} \tag{35}$$

Type-4 Reduction-Merging Rule:

Unlike the previous three rules, this fourth rule is a new rule introduced in the reference, and is only applicable to MVL systems (there is no direct binary equivalent). It is used to reduce an MVL decision table into its irredundant (most reduced) form. Originally, "Reduction" and "Merging" operations would be performed in separate steps. The example of this is shown in Equations 36-38 [117]. First, it is assumed that the variable *B* is composed of states {-1, 0, 1}, and the starting table is given in Equation 36.

$\boldsymbol{Row}$	$\boldsymbol{A}$	$\boldsymbol{B}$	$\boldsymbol{C}$
$\mathbf{1}$	$\mathbf{2}$	$\mathbf{0}$	$\mathbf{1}$
$\mathbf{2}$	$\mathbf{2}$	$\mathbf{1}$	$\mathbf{1}$
$\mathbf{3}$	$-$	$\mathbf{-1}$	$\mathbf{1}$

(36)

Applying the Type-3 operation ("Reduction") to the starting table, will reduce it to the decision table seen in Equation 37.

$\boldsymbol{Row}$	$\boldsymbol{A}$	$\boldsymbol{B}$	$\boldsymbol{C}$
$\mathbf{1}$	$\mathbf{2}$	$-$	$\mathbf{1}$
$\mathbf{2}$	$\mathbf{2}$	$\mathbf{1}$	$\mathbf{1}$
$\mathbf{3}$	$-$	$\mathbf{-1}$	$\mathbf{1}$

(37)

After the "reduction" operation is utilized, it is seen that the "absorption" rule can be applied, allowing Row 2 of Equation 37 to be absorbed by Row 1, resulting in the simplified decision table shown in Equation 38.

$\boldsymbol{Row}$	$\boldsymbol{A}$	$\boldsymbol{B}$	$\boldsymbol{C}$
$\mathbf{1}$	$\mathbf{2}$	$-$	$\mathbf{1}$
$\mathbf{2}$	$-$	$\mathbf{-1}$	$\mathbf{1}$

(38)

The implementation of a "Reduction-Merging" rule allows for the above process to occur in one step, instead of two. It is given the following definition in the literature: If a decision table includes terms where all of the *m* possible states of the output variable exits, and if none of the other variables have opposite entries, then those terms can be simplified. The larger of those terms is reduced by that input variable, and all other terms with identical input entries are also removed from the table [117].

The above four logical reduction rules are applied to the “Critical Transition Table”, to determine the PIs for that system/Top Event.

Method of Generalized Consensus (Literature Example)

This sub-section presents an example of a generic MVL application, as seen in the literature. The following arbitrary variables and their states are given in Table 12 [21,117]:

Table 12: Variables and States for the Literature Method of Generalized Consensus Example

Variable	States			
A	-1	0	1	N/A
B	N	R	F	N/A
C	-2	-1	0	1
D	H	N	L	N/A

From the variables/states in Table 12, a decision table for a top function, denoted as “TOP” (assumed to have a value of “1”), was constructed and is shown in Table 13. This decision table is not in its irredundant form, and as such must be reduced, using the logical reduction operations. The initial decision table was given as [21,117]:

Table 13: Initial Decision Table for the Example "TOP" Function

Row	A	B	C	D	TOP
1	-	R	-1	N	1
2	0	-	1	H	1
3	-	R	0	-	1
4	-	-	-1	L	1
5	0	R	-1	H	1

6	-	N	-2	-	1
7	-1	R	-1	H	1
8	0	R	-2	H	1
9	1	R	-1	H	1
10	0	F	-	H	1

First, Row 7 and Row 9 will undergo a “Merging” operation with Row 5. This returns a “Don’t Care” (“-“) value in Column 1 of that row. It also produces a new decision table, given in Table 14 [21,117].

Table 14: Decision Table for the Example “TOP” Function after the “Merging” Operation

Row	A	B	C	D	TOP
1	-	R	-1	N	1
2	0	-	1	H	1
3	-	R	0	-	1
4	-	-	-1	L	1
5	-	R	-1	H	1
6	-	N	-2	-	1
7	0	R	-2	H	1
8	0	F	-	H	1

The next step is to apply a “Reduction” operation to Rows 6-8 in Table 14. This returns a “-“ value in Column 2, Row 7. Additionally, Rows 1, 4 and 5 in Table 14 experience a “Reduction-Merging” operation, resulting in Table 15 [21,117].

Table 15: Irredundant Decision Table for the Example “TOP” Function

Row	A	B	C	D	TOP
1	-	R	-1	-	1
2	0	-	1	H	1
3	-	R	0	-	1
4	-	-	-1	L	1

5	-	N	-2	-	1
6	0	-	-2	H	1
7	0	F	-	H	1

At this point, table 15 is in its irredundant form, as no more logical reduction operations are applicable to any of the terms in that table, so those seven entries all represent PIs of that Top Event. However, there is one additional PI, which is generated from the consensus term using Rows 1-3 and 6. This final PI (consensus term) is shown in Row 8 of Table 16, along with the other seven PIs [21,117].

Table 16: Consensus Term and all PIs for the Example "TOP" Function

Row	A	B	C	D	TOP
1	-	R	-1	-	1
2	0	-	1	H	1
3	-	R	0	-	1
4	-	-	-1	L	1
5	-	N	-2	-	1
6	0	-	-2	H	1
7	0	F	-	H	1
8	0	R	-	H	1

From the example presented in this section, it becomes clearer that the "Variables" represent process variables in a DFM model, and the "States" are the different values which the process variables are allowed to take on. The example presented in this sub-section was a more generic, theoretical example, to demonstrate the theory behind the "Method of Generalized Consensus". A more practical example from the literature, is discussed in sub-section 2.3.2.9.

2.3.2.3 Consistency Rules (Time Dependence)

The previous sub-sections of sub-section 2.3 have mainly focused on the theory and rules behind the MVL aspect of DFM. However, the dynamic (time-dependant) behaviour of DFM is another important aspect of this methodology. In order to allow for time-dependant modelling, certain consistency rules are included. These can be divided into two categories, "physical consistency rules" and "dynamic consistency rules". Physical consistency rules are inherent rules, used to eliminate PIs that are physically impossible to occur, while dynamic consistency rules are set by the user, and will eliminate PIs based on the constraints on the dynamic behaviour of the system being modelled/analyzed [14,93]. The logic behind the physical consistency rules is derived from "Binary Algebra with Restrictions on Variables" [108]. If one assumes that a primary variable can be in any one of *n* states $\{1,2,\dots,n\}$, then a variable in state k would be considered as A_k. There are two restrictions forced upon the variable [14,93]:

$$\cup_{i=1}^{n} \boldsymbol{A_k} = \boldsymbol{T} \tag{39}$$

$$\boldsymbol{A_i} \cap \boldsymbol{A_j} = \boldsymbol{F}, \boldsymbol{for\ i \neq j} \tag{40}$$

In Equations 39 and 40, the overall meaning is that the variable must take on a state, but it cannot take on two different states at once. The use of these physical consistency rules are applied to prune out impossible PIs that are returned when the logical reduction operations are applied. As an example of physical consistency, the following is considered:

<variable A=1 AND variable A=2> (41)

Equation 41 would violate the physical consistency rule given in Equation 40. If time-dependency is considered, a PI in the form:

<A = 1 @ time t = T1 AND A = 2 @ time t = T2> (42)

would be allowed, as it no longer violates Equation 40. Here, the Prime Implicant is referred to as a Timed Prime Implicant (TPI).

In the case of dynamic consistency rules, they are optional, and can be set by the user, based on their knowledge of the system under analysis. They are applied to allow certain variations of parameter values across the time steps. Consistency Rules are discussed in more detail in sub-section 2.3.2.3

2.3.2.4 DFM Quantitative Analysis

The quantitative analysis for DFM is actually very similar to the quantitative process for FTA, once the PIs have been determined. Despite the significant differences in the algorithms used to determine the respective MCS/PIs, the methods for calculating the MCS/PI and Top Event probabilities almost the same. The Pi probabilities are a product of all of the events inside of that PI, analogous to the calculation of MCS probabilities in FTA, as stated in Equations 17 and 21. Regarding the Top Event probability, the same overall methods as discussed in sub-section 2.3.1.7 The "SUM", "MCSUB" and "EQ" methods are all applicable to DFM, replacing the MCS with PIs.

There is one additional way in which the EQ value can be calculated using DFM. The set of Prime Implicants is converted into a set of *m* Mutually Exclusive Implicants (MEI), denoted as MEI_1 – MEI_m:

$$\boldsymbol{Top\ Event} = \boldsymbol{MEI_1} \vee \ldots \vee \boldsymbol{MEI_m} \tag{43}$$

Where $MEI_i \wedge MEI_j = \phi,\ for\ any\ i \neq j$

As the MEI are basically the multi-valued logic version of cut sets that do not return a cross product term, the sum of the probabilities of these MEI gives the probability of the top event [14,93]

$$\boldsymbol{P(Top\ Event)} = \boldsymbol{P(MEI_1)} + \cdots + \boldsymbol{P(MEI_m)} \tag{44}$$

2.3.2.5 DFM Tools

In this research program, the actual modeling and analysis, when applying DFM, was be performed using a software package known as Dymonda, from ASCA Inc. [14]. Dymonda allows for the graphical construction of the flowgraph that represents the model being analyzed. The Dymonda software package utilizes the "Method of Generalized Consensus", which is why that method was the focus during this thesis [14,93].

In more modern versions of the software used optimized methods to reduce the section of the system being analyzed into a single critical transition table to produce the complete set of Prime Implicants. The software is also capable of calculating the complete base, from an irredundant base.

It should be noted that there exists an alternate implementation of DFM that has been performed by VTT in Finland, using their "YADRAT" (Yet Another Dynamic Reliability Analysis Technique) tool. This approach utilizes BDDs, and more specifically ZBBDs to determine the PIs for a system under analysis [121–124]. However, as YADRAT is not a commercially-available tool, it could not be directly applied to the analysis in this thesis [125].

2.3.2.6 Advantages and Disadvantages of DFM

DFM is a large improvement upon the stand static fault tree analysis. Normal fault trees create cut sets for only one, static, binary top event. DFM will generate time-dependant and multi-valued logic Prime Implicants for a large variety of possible top events for the system. The top event can be selected from any of the possible states and variables in the model, including using multiple states and variables for the same analysis. This means that one the DFM model has been created, it can be used to model a large variety of possible top events [14,93]. However, DFM is not without any drawbacks, with the main issues being the amount of information and computational resources required. In order to create an accurate model, the user must have access to a large amount of data for the system in order to properly discretize every node and transition table. In many cases, this information is obtained through a Failure Mode and Effects Analysis (FMEA), and other available data. Additionally, for large systems, running the analysis can require a lot of time (in the example shown in sub-section 2.3.2.9, it took more than 24 hours to run the analyses for each top event), making detailed analyses of a system a very time consuming process.

2.3.2.7 DFM Modelling

The actual DFM model (constructed using Dymonda, during this thesis) relies on a series of process variable nodes, connections and transitions/transfers between these nodes to show the relationship between the parameters of the system [25,26]. The nodes and transition boxes (as of version 7.0) are seen in Figure 47.

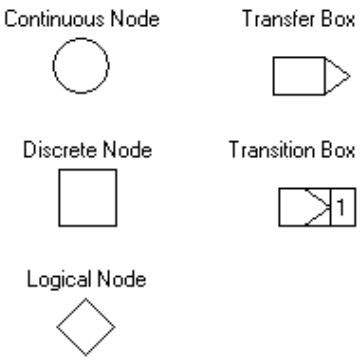


Figure 47: DFM Nodes and Transfer Boxes

The nodes are used to represent the parameters, variables or components of the system in question. Continuous nodes represent a continuous behaviour, discrete nodes represent a discrete behaviour, and logical nodes are there to show logic tests on the current state of the system. Under analysis, all 3 nodes function the same way, but look different in the model to make it easier to understand what is being represented. A process variably node could represent an output value, such as a current or voltage output. A transfer box represents functional relationships between the components of the model (the transfer function of the system). The transition box works in mostly the same way, but accounts for time delay, allowing one to model the time dependence of the system and/or the system components. The default time delay is 1 unit of time, but it can be altered. The connections between the various nodes and boxes are done one of two ways [25,26]:

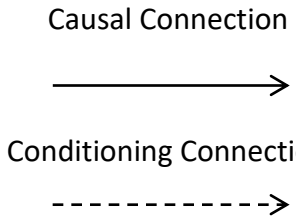


Figure 48: DFM Connectors

The causal connection shows the input and output of the transfer/transition boxes (cause-and-effect behaviour). The conditioning connectors illustrate the connections between input and output of functions and will determine what function is being used. Once the model has been constructed using these tools, the analyses can be run on the system.

2.3.2.8 Additional Rules and Functionalities

DFM contains additional rules and functionalities to perform the requested analysis. Some of these rules are applied automatically, to filter our impossible scenarios, while others are defined by the user, to simplify or streamline the analysis. A discussion of some of these added rules and functionalities is included here [14].

Physical Consistency Rules: Rules applied automatically by the software during analysis to eliminate and impossible conditions found when creating the intermediate/timed fault trees. An example of this is shown previously in Equations 40 and 41, where a system variable cannot be in two separate states in the same time step.

Dynamic Consistency Rules: Similar to physical consistency, as they are applied to filter out impossible scenarios. However, these rules are set by the user, and are used to constrain the system's dynamic behaviour. They are applied to allow certain variations of parameter values across the time steps.

State Dynamic Consistency Rules: Dynamic Consistency Rules that are applied to certain states of nodes. There are three separate forms of these rules:

> **Sink:** This rule creates a sink state, indicating that once a node enters into that sink state, it is unable to transfer to another state for the rest of the analysis. Sink states can be used to model unrecoverable failures.
>
> **Min/Max Duration:** This rule will decide how long a node will remain in a certain state, once it has entered that state, before it can transfer to a different state. An example would be for a minimum duration of 3 would mean that the node must wait for 3 time steps before it can transition.
>
> **Min/Max Intermittence:** This rule controls how long a node needs to remain in different state(s) after transitioning out of the state this rule was applied to. An example would be if a state had an intermittence value of 3, then there would have to be a 3 time step wait period after transitioning out, before transitioning back in.

Classification Nodes: Similar to logic nodes, classification nodes are used to group together similar states into one node. These nodes are typically used when a distinction between similar states in early

points of the mode model are important, but not in the latter parts. They are not essential in the model, and can be replaced using regular process variable nodes, but are sometimes used to simplify and clarify the model for the use. In Dymonda, classification nodes are represented by a triangle shape.

Event Nodes: These nodes represent events (something that will either happen or not happen). The main difference between these nodes and standard nodes is that with a standard node, the probability assigned to it represents the probability of the node being in certain state(s). However, with an event node, the probability characterizes the probability of transitioning into a particular state. An example would be a control valve failure: the valve failure is the event, it fails in a particular moment and then stays failed. With event nodes, the transition probabilities are given by a state transition matrix. Both continuous and discrete nodes can be converted into event nodes, which are displayed differently in the model viewer.

Condition Nodes: These are not a separate type of node, but rather represent a node that will affect the outcome of the process, but is not itself affected by the process. Condition nodes characterize parameters which will affect the overall logic structure of the system (often a digital system) by changing the causal relations connecting the different process variable nodes. Condition nodes often represent parameters like states for component failure, changes of process operation modes, and software changes.

Boundary Conditions: Typically applied during inductive analyses, are defined to reflect the failure profile of certain elements in order to determine the effects of that various combinations of components failures will have on the overall system. In essence, the boundary can freeze a component in a certain state for a certain number of time steps, so see the effect that it, or other components can have on the evolution of the system.

2.3.2.9 DFM Example of a Digital I&C System (Digital Feedwater Controller)

A recent, well known example of applying DFM to nuclear I&C is the analysis of a Digital Feedwater Control System (DFCS) for a Pressurized Water Reactor (PWR). The model was constructed using DYMONDA, and then was first analyzed deductively for two separate top events: SG low level and SG

high level. Afterwards, the same model was analyzed inductively, for the initiating events “Main Feed Valve (MFV) Stuck During Ramp Up” and “Main Feed Valve Stuck during Ramp Down”. The DFWCS was chosen because it had been a major contributor of plant trips instances, although it is not strictly defined as a safety system. These instances make use of the safety systems, and therefore are some of the initiating events for accident scenarios in standard plant PRAs. The report states that these instances are more likely to occur when the reactor power ramps up or ramps down, so those cases were chosen for the analysis. A screenshot of the model is shown in Figure 49 [12,15].

The nodes were all discretized to include the relevant state information, and then the decision tables in transfer/transition boxes were completed. Afterwards, the analyses were performed using the aforementioned top events/initiating events, with a selection of the results being included here. In these analyses, the probabilities were given as an hourly failure rate. It should also be noted that the time step (t) is not actually negative, but in the case of the deductive analyses, the top event is generally said to be at time zero (t = 0). Additionally, in the following tables, the bold entries illustrate the main failure states, while the italicized entry shows the boundary condition(s) [12,15].

In Table 17, the Prime Implicant with the highest probability is shown. “Mode = 1” denotes that the reactor is operating normally (70% power in steady state), “PDI-T = OP” means that the transition of the PDI controller is operating normally, “MFV-T = Stuck” shows that the main feed valve transitions to being stuck, while the “MFV-P = Comm” signifies that the previous main feed valve state was communicating/operating normally. “MFVA-P = 2” is the size of the main feed valve aperture in the previous state, given as 70-74%, and “LP = 0” corresponds to the previous SG level being in the range of -0.17 to 0.17 feet. From this analysis, the most likely Prime Implicant involved the main feed valve getting stuck in a largely open position such that the increase in steam flow cannot be counteracted by the increase in feedwater which in turn leads to the SG level to decrease until it reaches and unacceptably low level. The probability of this Prime Implicant occurring was found to be $3.33*10^{-4}$, out of a total probability of $4.19*10^{-4}$ for the top event. The Prime Implicant given in Table 17 is one of 1197 Prime Implicants.

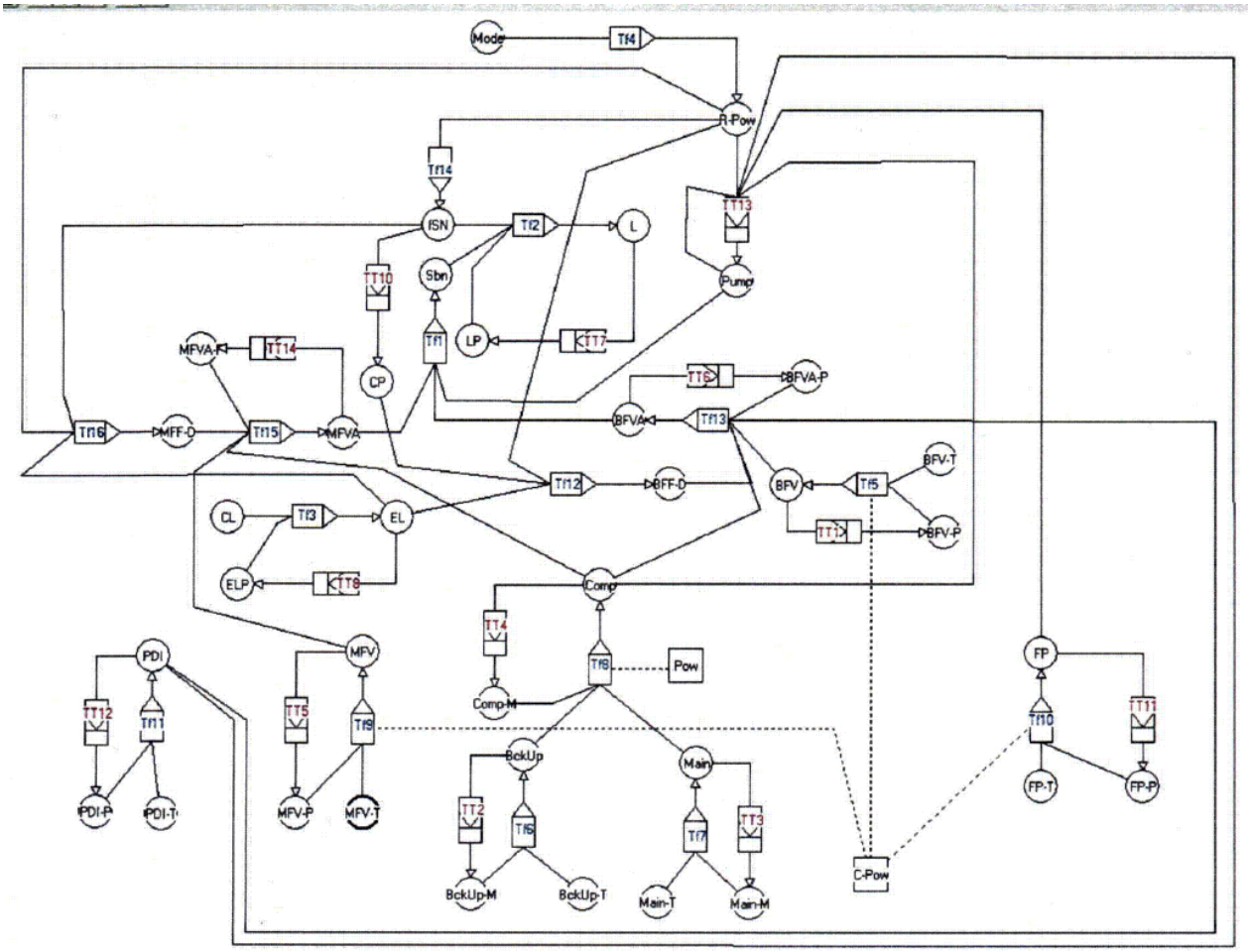


Figure 49: DFM Model for DFCS Benchmark Example in NRC Report (NUREG/CR-6985)

Deductive: SG Low Level

Table 17: SG Low Level Prime Implicant No. 1

No.	Prime Implicant	Probability (Failures/hour)
1	Mode = 1 @ t = 0 *PDI-T = OP @ t= -1* **MFV-T = Stuck @ t = -1** **MFV-P = Comm @ t = -1** MFVA-P = 2 @ t = -1 LP = 0 @ t = -1 Mode =1 @ t =-1	$3.33*10^{-4}$

Deductive: SG High Level

Table 18: SG High Level Prime Implicant No. 1

No.	Prime Implicant	Probability (Failures/hour)
1	Mode = 1 @ t = 0 **MFV-T = Stuck @ t = -1** **MFV-P = Comm @ t = -1** MFVA-P = 4 @ t = -1 Main-T = OP @ t =-1 LP = 0 @ t = -1 Mode =1 @ t =-1	$3.33*10^{-4}$

In Table 18, the "Main-T = OP" represents the transition of the main computer to an operational state. Another difference in this analysis is that the previous state of the main flow valve (MFVA-P) is in state 4, not state 2, corresponding to a 78% position. In this instance, the MFV is again stuck, this time in the 78% position. The decrease in the steam flow cannot be offset by reducing the feed flow, which will lead to the SG level to increase until it reaches an unacceptably high level. The probability was calculated to be $3.33*10^{-4}$ failures/hour, with a total probability of this top event occurring being $3.3374*10^{-4}$. The Prime Implicant given in Table 18 is one of 138 Prime Implicants.

Inductive: MFV Stuck (Ramp Up)

For this analysis, a number of initial and boundary conditions were selected, including the top Prime Implicant from the SG Low Level deductive analysis. The analysis is then performed forwards in time, to determine the effects of those initial conditions on the system. When the analysis was complete, it was seen that the Main Feedflow Demand (MFF-D) attempts to follow along with the power increase, the MFV gets stuck (fails), forcing the aperture (MFVA) to remain in same position, causing the SG level to drop. The total probability was calculated to be $3.34*10^{-4}$.

Inductive: MFV Stuck (Ramp Down)

'The inductive analysis was repeated for the Ramp Up case, using the top Prime Implicant from the SG High Level deductive analysis as some of the initial conditions. After the analysis was finished, it was found that the MFF-D attempts to keep up with the decrease in power, however the valve failure (valve stuck) again forces the aperture to remain in its previous state, causing the SG level to increase. The total probability was calculated to be $3.34*10^{-4}$.

The results from the NRC reports showcased several important factors regarding DFM. First, it is capable of modelling a complex digital control system that is an important part of plant operation. Second, it showed that by constructing one model allows for many difference analyses to be run on the system. Third, that due to the dynamic capabilities of DFM, it can reveal how certain component failures can lead to very different top events (i.e. the top Prime Implicant for both SG low and SG high were very similar, with essentially the same major error), and fourth, that the inductive analyses can confirm both the cause of the top event, but also the probability of it happening.

2.4 Chapter Summary

This chapter presented the relevant background information required for this thesis. It was divided into two sections; background information on FPGAs, and the background information on the two reliability analysis methodologies (DFM and FTA). The section on FPGA background included the descriptions of FPGAs, how they relate to other forms of control logic, architectures, FPGA technologies, the programming and lifecyle process, as well as potential advantages and disadvantages of FPGAs. A detailed literature review on the use of FPGAs in NPP I&C systems was also presented. The section on reliability analysis considered both FTA and DFM, covering topics such as coherent, non-coherent and many-valued logic and qualitative and quantitative analysis methods. Additional DFM-specific information considered consistency rules, potential advantages of DFM, as well as an example system from research performed by the US NRC.

3. FPGA Failure Modes Taxonomy

An important component of this research program considered a detailed analysis of an FPGA-based system, with realistic failure mode data. Therefore, an extensive literature review was undertaken, to determine the potential failure modes, both hardware and software (HDL code), that could affect an FPGA-based system. Furthermore, it was seen that this failure mode data could be categorized, to present it in useful fashion to those designing and reviewing FPGA-based systems in NPPs.

While this research work into FPGA failure modes was occurring, the OECD-NEA published their failure mode taxonomy for digital systems [9]. This taxonomy presented a great deal of information regarding the failure modes of generic software-based I&C systems, however it did not specifically include FPGAs. Therefore, the expansion of the collected FPGA failure mode data into an FPGA Taxonomy, became the next step in the research work. The preliminary work from the FPGA failure mode work was presented at the "*7th International Conference on Modelling and Simulation in Nuclear Science and Engineering*" [126]. The full FPGA Taxonomy was submitted to the journal "Reliability Engineering and System Safety", and the revised version is currently under review [127]. The information obtained from the FPGA FMEA and the resulting FPGA Taxonomy were used as a form of fault injection for the DFM and FTA models used in sub-sections 4.3. and 4.4

Sub-section 3.1 details the results from the FPGA failure modes research, including the literature review, failure sets/failure mode categorization, and failure mode mitigation methods. Sub-section 3.2 discusses the OECD-NEA digital failure modes taxonomy, and the important aspects of it. Sub-section 3.3 presents the new FPGA Taxonomy, including how it can interface with the original OECD-NEA taxonomy. Sub-section 3.4 provides a summary of the information discussed in this chapter.

3.1. FPGA Failure Modes Research

The first step in creating the failure modes taxonomy for FPGA-based systems was to compile information on all available failure modes that could apply to FPGAs. Not all of these may be specific to FPGAs or FPGA-based systems, but they still apply to FPGAs. Over the course of that part of the research

program, the FPGA failure mode data was compiled, processed, and categorized to present clear, useful information for the purpose of designing, analyzing and reviewing FPGA-based I&C systems. The following sub-sections, detail the work on FPGA failure modes.

3.1.1. Failure Mode and Effects Analysis (FMEA)

Failure Modes and Effects Analysis (FMEA) is a commonly used method in reliability and safety analysis, and are often performed at the start of a reliability/safety analysis program. FMEAs become an important part of that program. The FMEA was performed on the FPGAs to identify data regarding the failure modes. This includes identifying the potential failure modes, their causes, their potential effects on FPGA-based systems, as well as providing information on how to eliminate or mitigate/control those failure modes. Additionally, the FMEA can identify the effects of latent design errors, and/or determine if the Single Fault Criteria is established [128]. To obtain the data required for this FMEA, an extensive literature review was performed, taking into account a wide variety of information from the international community. This documentation included the aforementioned reports from the US NRC and ORNL, reports from VTT, EPRI and the OECD-NEA, and standards from IEC, IEEE and CSA. Documentation, such as reports and white papers from FPGA manufacturers (Xilinx, Altera and Microsemi) were considered, as was research published in scientific journals and conferences. The data from the literature review was further processed during this research, to provide a more detailed analysis of the failure mode information

The FMEA resulted in a list of potential failure modes, based on the literature review. These failure modes were then broken down in categories based on the point in the FPGA-based system lifecycle where the failures occur, as well as the Failure Types. The FMEA was also used to document the potential cause(s) of each failure mode, its potential effect(s) on an overall FPGA-based system, and identify the methods to eliminate or mitigate those failure modes. Recommendations are noted from the study of the literature available for these failure modes. These include recommendations for regulatory review of the FPGA-based system(s). The FMEA results from this work provides additional information for use in future modelling and reliability analysis of FPGA-based systems, to further analyze the failures and the methods to mitigate those failures and to develop regulatory review guidelines to implement FPGA-based safety systems.

Some important definitions with regards to FMEA are given below [129]:

FMEA: A systematic procedure for the analysis of a system to identify the potential failure modes, their causes and effects on system performance. In the IEC 60812 standard, "system" is used as representation of hardware, software (with their interactions) or a process.

Failure Mode: Manner in which in item

Failure Effect: Consequence of a failure mode in terms of the operation, function or status of the item

Item: Any part, component, device, sub-system, functional unit, equipment or system that can be individually considered

The results for the FPGA FMEA include a breakdown of the different failure categories and the sets of general failure modes (as discussed in Section 5.2.3 "*Failure Mode Determination"* of IEC 60812) [129]. Additionally, the FMEA presents information on the failure modes, effects of those failure modes, as well as potential mitigation methods. It should be noted that this research did not consider a specific system. Instead, the FMEA focused on general FPGA-based systems, and only considered the effects of failures at the board level. The effects mentioned in this paper are the effects that the failure modes would have on the system elements that are under consideration. These are referred to as "Local Effects" in Section 5.2.5.2 "*System Initiation, Operation, Control and Maintenance"* of IEC 60812 [129].

3.1.2. FPGA Failure Modes Categorization

The results of the FMEA were modelled using two main categories; "Design" and "Operation". The "Design" category consists of two groups "Design Defects" and "Manufacturer Defects", while "Operation" is broken down into "Environmental", "Stress/Aging", and "Maintenance Induced (Human Factors)". These terms are explained in more detail below, and a visual representation is given Figure 50.

These groups are again divided into smaller subsections, based on their effects on the system and the remedy actions to eliminate, mitigate and control these potential failure modes.

Design: Modelled failures with the logic or the chip itself that occur in the design or fabrication stage.

Design Defects: Failures due to problems with the FPGA logic and/or the system hardware.

Manufacturer Defects: Defects in the physical FPGA chip from the manufacturing process.

Operation: Modelled failures that occur while the FPGA is in operation inside the nuclear power plant.

Environmental: Failures that occur due to the environment that the FPGA is operating in. Possible failures would include radiation-induced failures such as SEEs.

Aging/Stress: Failures that occur due to the aging effects experienced by semiconductors, as well as thermal and/or mechanical stress on the FPGA.

Human Factors: Failures that occur during maintenance, such as tampering with the FPGA, that is either intentional or unintentional.

This categorization was done in this research for the ease of discussing the remedy actions, and was not present in the literature that was surveyed. For example, for the failure modes considered as “Design Defects”, efforts should be made to eliminate/mitigate those failure modes during the design state. In the case of residual failure modes, Built-In Self-Test (BIST) should be incorporated into the design to detect errors that occur during operation.

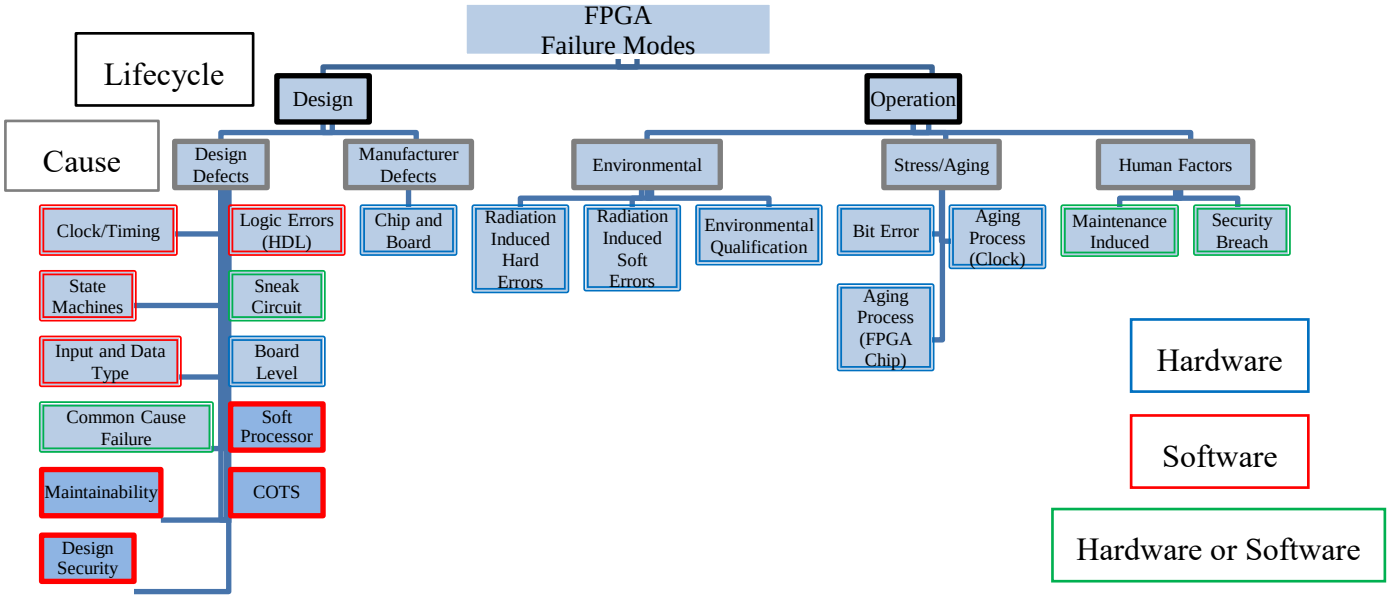


Figure 50: FPGA Failure Mode Categories ("Failure Sets")

A detailed description of all of the failure sets shown in Figure 50 is presented in sub-section 3.1.3.

3.1.3. Sets of Failure Modes

IEC 60812 describes the different failure groups as a "Failure Effects Summary", so that terminology was used in this report. IEC 60812 defines "failure effect" to be the "consequence of a failure mode in terms of the operation, function or status of the item" [129]. The categories mentioned in sub-section 3.1.2 were then broken down into several "Sets of Failure Modes", based on their similar causes and/or failure effects. Each set that was identified in the FMEA includes a description of the set, and methods to eliminate or mitigate those failures. This section is organized with the description of each set of failure modes, followed by the discussion of the avoidance and/or mitigation methods.

3.1.3.1. Design Defects

Design defects considered in this section focus on logic failures, as well as potential hardware "Faults and Failures to be assumed when quantifying the effect of random hardware failures or to be taken into

account in the derivation of safe failure fraction", as specified in IEC 61508-2, Table A.1 that were deemed to be relevant to FPGA-based systems [130].

Clock/Timing Failure Modes

Proper timing is critical for FPGA systems to function as intended, so any design or logic errors that affect the clock, upset the system timing, or in general create timing errors can cause the system to behave incorrectly. Therefore, all potential failures due to the timing/clock behavior must be controlled.

Many of the failure modes in this category were due to the use of asynchronous (not tied to the clock) signals, and could be remedied by using synchronous (tied to the clock) signals instead. If the input signal is asynchronous, synchronizer chains (double or triple registers) or FIFOs (First In First Out) can be used to synchronize that signal. Timing errors can be identified and eliminated during the design phase through the inclusion of proper timing constraints, Static Timing Analysis (STA), and by performing timing (gate-level) simulations. The gate level simulations are used to expand upon STA (limited when analyzing asynchronous signals and multi-cycle clock paths), verify reset sequence and initialization, power estimation, detect metastability or glitches, and verify the proper timing execution [38].

Logic Error (HDL Programming) Failure Modes

These failure modes refer to failures due to errors in the actual HDL code used to program the FPGA, not an error in the design specifications. The FPGA is programmed by the end user, meaning any programming errors or deficiencies could lead to unforeseen logic errors in the final system(s) [65,131]. Such issues include synthesizer problems, calculation errors, difficulties with simulation, and additional asynchronous behaviour.

The failure modes in this section were due to programming errors, and can be avoided by following HDL programming standards and industry best practices [37,132]. Additionally Section 3.1 "*Reliability*" of NUREG-7006 provides a summary of the methods to avoid these programming errors [65].

State Machine Failure Modes

State Machines, or Finite State Machines (FSM) are a mathematical computational model that is used to design software programs and sequential logic circuits. FSMs have seen extensive use in FPGA-based I&C systems, meaning any failures in the state machine could cause the system to hang, suppressing the outputs and/or cause the erratic activation of other system elements. The FMEA uncovered 8 potential failure modes for (FPGA) state machines.

IEC 61508-7, Section B.2.3.2 "*Finite State Machines/State Transition Diagrams"* provides general guidance for state machines, while IEC 612566, Section 8.4.6 "*Finite State Machines"* provides requirements for FPGA-based state machines. State machines should be analyzed and tested to ensure that they conform to those standards. Reset signals can also be used, to force the state machine into a state that is already analyzed (such as the starting state), in case errors/hang-ups/deadlock occurs [37,133].

Sneak Circuit Failure Modes

When considering the FPGA logic, the sneak circuits are a design error that could either cause the intended output not to be generated, or cause an unintended output to be generated. Due to the possible system failure, sneak circuits must be analyzed. In this case, it is the only failure mode considered. It should be noted, however, that sneak circuits could be caused by either hardware design errors or software design errors. Sneak circuits are a form of latent design error.

Sneak circuits should be eliminated where possible. Even for small, combinational FPGA-based system designs with a limited number of I/O, and where 100% testability can be obtained, that testing may not guarantee that the system is sneak circuit free [65,131]. To remedy this, Sneak Circuit Analysis (SCA) should be performed to locate and eliminate sneak circuits. Guidelines for general sneak circuit analysis [134–136], and FPGA-specific sneak circuit analysis are applicable [137].

Input and Data Type Failure Modes

This section includes information about possible input or data type errors. The inputs can overflow and/or become stuck, causing inaccurate data to be propagated through the system. The use of different data types (such as fixed-point or floating-point packages), which currently are not well-supported by many vendor tools (e.g. VHDL-2008 is not as universally supported as VHDL-1993). At the time of this thesis, the current standards for HDL code include IEEE-1076 (VHDL), IEEE-1364 (Verilog), and IEEE-1800 (SystemVerilog).

In order to mitigate these failure modes, the input range should be properly defined before implementation, to avoid overflow altogether [48]. However, in case overflow does occur, it must be detected, and there should be alerts that are sent to the operators to warn of errors [65]. Resolution and Resize errors can occur when using Fixed-Point mathematics (a data type that has a fixed number of digits before and after the decimal point, as opposed to the Floating-Point math that is commonly used in modern computers), which is common in FPGAs. The proper resolution for all values in the system (especially critical parameters such as setpoints) should be carefully calculated beforehand. The potential failures that could be caused by the newer fixed-point and floating-point data type packages (such as those in VHDL-2008) can be avoided by using the better supported standards (such as VHDL-1993) instead of the newer packages [38].

Board Level Failure Modes

These represent general, high-level failure modes (potentially representing fault tree top events), which require consideration under IEC 61508-2 [130]. Complying with that standard will assist in achieving a system where the hardware will function reliably. This section includes a subset of the IEC 61508-2, Table A.1 items that are relevant to FPGA-based systems, which accounts for 9 assumed failure modes.

These failure modes were taken from the standard IEC 61508-2, Table A.1. Additional information on those failure modes is presented in Tables A.2 to A.14 of that standard. These represent general failure modes, that any electrical/electronic safety system could encounter, and must be eliminated (or controlled). The methods for testing, detection, eliminating and controlling these failures can be found in IEC 61508 [130].

Commercial-Off-The-Shelf “Software” (COTS)

Represents dedication of any commercial grade software (HDL code, IP cores) and software tools used in the configuration of the FPGA-based system [138]. This failure set would also include Pre-Developed Software (PDS), used by the FPGA.

Issues due to COTS software are eliminated through the (commercial-grade) dedication process. In terms of the nuclear industry, guidance on this dedication exists specific to FPGAs [138], as well as general guidance for software-based systems that is applicable [139,140]. Additionally, the requirements by some regulators states that PDS needs to undergo some form of failure modes analysis, such as FMEA) according to certain standards [141].

Maintainability

Attributes included during the “Design” phase, which will assist with the maintenance of the “software” (HDL code) during the “Operation” phase. Attributes that may impede maintainability include the use of vendor-specific IP Cores, vendor-specific hard macros, synthesis attributes and constraints, as well as place and route directives [65,131]. An overreliance on vendor-specific features also reduces the portability of the HDL code.

In general, the “Maintainability” issues are avoided by avoiding (or at least minimizing) the use of vendor-specific features in the design of the FPGA-based system [65,131]. With regards to certain regulators, there are requirements put in place regarding “Design for Maintainability”, which must be met [142].

Design Security

This failure sets consider malicious logic (such as HDL code or IP cores) functions and/or timing constraints inserted into the FPGA during the design stage. These malicious functions could lay dormant, waiting to be activated upon some triggering condition (logic or timing bombs) [143]. This failure set is different from the “Security Breach” failure set, as it is strictly software considers malicious logic inserted into FPGA chip during the “Design” stage in the lifecycle, whereas the “Security Breach” failure set considers hardware and software failure modes, and only considers malicious acts during the “Operation” part of the lifecycle.

Mitigation of this failure set requires the use of a secure lifecycle, for the entire lifecycle of an FPGA-based system in the NPP. All of the design/qualification of the system must be performed using trusted, tools, personnel and IP cores (if included) [3,144,145].

Common Cause Failures (CCF)

Common Cause Failures (CCF) are a serious issue in reliability engineering, as seemingly redundant systems can fail due to a single initiating event, which removes the underlying assumption that all failure modes are independent. Therefore, one cannot only consider independent and random failures, as CCFs must be mitigated, in order to have a reliable design. There is only 1 failure mode in this set (CCF itself), and it is a form of latent design error that is assumed to exist in the design and cannot be found through testing. CCF was included in the "Design Defect" section, as the system logic has the potential to be to cause of a CCF, however hardware or environmental factors may also cause CCFs, as stated below.

Conventional CCF causes are [129]:

- Design (Software/Logic)
- Manufacturing (Component Flaws)
- Environmental (Temperature, Electrical Interference)
- Human Factors (Maintenance Actions)

FMEAs have limited use when analyzing CCFs, however it can be used to study all the possible causes that could trigger a CCF [129]. Traditionally, a combination of different methods is used to mitigate CCFs. These include using functional diversity and defense-in-depth, system modelling, component analysis, and the physical separation of components. If the system is a very simple, asynchronous design, it is possible to obtain 100% test coverage, which would eliminate any latent design (logic) error. However, that would not remove the vulnerability to CCF due to other causes. Due to the importance of CCF to the nuclear industry, it has been widely studied, with discussions on coping with CCF published in the recent literature [146–149]. Additional standards and documents from IEC [37,150,151], IEEE [36,152], the IAEA [153,154], and regulators such as the CNSC [142] should be considered when designing systems that are resilient to common cause failures.

3.1.3.2. Manufacturer Defects

FPGA Chip and Board Failure Modes

These are failures that can occur in the manufacturing/fabrication process of the physical FPGA chip (hardware), and could result in damaged pins, or impurities that cause corrosion. The failure modes of the chips itself must be known, to ensure that the chips used in the FPGA system are reliable.

To mitigate failures due to the FPGA chip and board, the chips should be inspected and tested for any defects and/or impurities that would cause failures and/or accelerate the aging process. The chips should be inspected for cleanliness (such as water or foreign particles), that could lead to corrosion or ion mobility failures. The chips should be tested to check for any damaged pins, and the supply voltage/current should be tested to eliminate power-up or power pin decoupling errors [65,131,155]. Additionally, Bent Pin Analysis (BPA) (also called Cable Failure Matrix Analysis (CFMA)) should be performed, to determine the potential hazards of bent/damaged pins, and the mitigation required to control those hazards [156,157].

3.1.3.3. Stress/Aging Failures

These failures are due to the aging process. As with any technology, aging effects will eventually render the chip unusable. The failures due to the aging process must be mitigated, as these failures modes cannot be completely eliminated during the design phase of the FPGA-based system.

Bit Error Failure Modes

These failures are almost exclusively due to the continuous reconfiguration of SRAM FPGAs. Eventually, the multiple reconfiguration cycles will result in bit failures. Bit errors, coupling faults, stuck bits, and data degradation were all modeled under the umbrella term of “Bit Error Failures”, as they all affect the state of the bits (“0” or “1”) [158]. However, configuration errors are still possible in Antifuse and Flash FPGAs, and non-volatile memory can degrade over time, so 2 of those failure modes could still apply to non-volatile FPGAs.

Bit Error Failure Modes can largely be avoided by using Antifuse (OTP) FPGAs. The Antifuse FPGAs will eventually lose data, however the Mean Time To Failure (MTTF) can be calculated to predict when that failure will happen [65,131]. Regardless of the FPGA technology (SRAM, Flash or Antifuse), the configured system should then be tested thoroughly, to verify that it performs correctly and matches the simulations, to ensure no bit errors have occurred, or detect the failure(s) if any did occur. If a bit error occurs with an OTP FPGA, then the FPGA would have to be replaced, however the Flash or SRAM FPGA could be reconfigured in the event of a configuration or routing error. However, the SRAM chip must be re-configured every time there is a power cycle, meaning that it must be tested thoroughly every time, whereas Flash and OTP chips are only configured once and retain their configuration through power on/off cycles [65,131]. The constant re-configuring that occurs with SRAM FPGAs is what causes it to be susceptible to configuration failures, while OTP and Flash FPGAs are more resilient.

Aging Process (Clock) and Aging Process (FPGA Chip)

Failure modes due to aging cannot be eliminated during the design and implementation part of the life-cycle, and as such must be mitigated. Many of these failures have dependencies on temperature, electric properties (electric field, voltage, or current) or the material properties. The two Failure Sets are discussed jointly, as they have similar causes, although the effects are different.

Aging process failures, of either case, cannot be eliminated from the FPGA. In both cases, statistical methods, such as thermal aging calculations/test and MTTF calculations can be used to estimate when failures will occur, and the MTTF data should be reflected in the maintenance program [155,159]. The use of (cold) system redundancy could be included, to ensure the system will function properly if one component/function fails. Built-In Self-Test (BIST) should be implemented, to test for signs of aging process failures, and to indicate to maintenance personnel that the aging process has become hazardous, and the FPGA is starting to fail. However, aging failures could also affect the BIST, so periodic testing should be performed at scheduled maintenance intervals. The BIST features should be isolated (separate from the safety features), so that the BIST functions do not interfere with the primary safety functions of the system.

With regards to the Aging Process (Clock) specific failures, the timing constraints/requirements of the FPGA, and total system should account for the eventual clock slowdown of the FPGA, due to aging failures that will deteriorate the clock frequency [160].

The periodic tests should include tests for clock period (Clock failures), and temperature, power use/dissipation and short-circuit currents (Chip failures), as those parameters can indicate that the chip is starting to degrade and may need to be replaced. The aging process failures will eventually require the FPGA to be replaced [65,131].

3.1.3.4. Environmental Induced Failures

The failures in this sub-section are all due to interactions with the FPGA and the surrounding environment. In general, this includes failures due to inappropriate environmental qualification, and in the case of digital electronics, interactions with the FPGA and ionizing radiation.

With regards to radiation-induced errors, they are caused by radiation interactions with the semiconductor materials of the FPGA. Many of these failures are part of a group known as Single Event Effects (SEE), however there are multiple events as well. Radiation failures can either cause destructive (hard) or non-destructive (soft) errors.

Environmental Qualification

This failure category consist of failures that should be accounted for during environmental qualification, such as failures caused by high temperatures, humidity, electrical noise, electromagnetic interference (EMI), seismic/vibration, etc., As an example, high temperatures may cause damage to the FPGA-based system at the device, package and board levels. This includes damage to the soldering, die, connectors, substrate and bonds. It should be noted that temperature is an important factor in many aging-related failure modes, however those are considered under the "Stress-Aging" cause in Figure 6 [1,161,162].

In order to mitigate these failures, the appropriate environmental qualification procedures/guides/standards must be followed, for the duration of the FPGA-based system lifecycle [1,163,164].

Radiation Induced Hard Errors

Hard errors include SEEs such as Single Event Burnout (SEB) and Single Event Dielectric Breakdowns), which will cause permanent damage to the FPGA chip. Over time, this could cause severe damage or complete destruction of the chip. These interactions include destruction of logic gate and LUTs, as well as short-circuit currents that will burnout the FPGA chip [165–167].

Mitigation methods for Hard errors include the use of radiation resistant/tolerant FPGAs (protective circuits), power de-rating, and sensitivity testing to determine the sensitivity of the FPGA chip to these destructive errors [167–169].

Radiation Induced Soft Errors

The Soft Errors failure set considers events such as Single Event Upsets (SEU) and Single Event Transients (SET). This are temporary (transient) events that may affect the memory/logic of an FPGA, however they will not cause permanent damage to the chip [165–167].

Soft Error failure modes can be eliminated by using Antifuse or Flash-based FPGAs, as these materials are resistant to radiation interference [65,131]. In the case of SEUs, the use of redundancy, such as Double Modular Redundancy (DMR) or Triple Modular Redundancy (TMR) is often implemented [170–172], while in the case of SETs, circuit freezes could be used to stop the transient from affecting the FPGA logic states [173]. Various error detection and correction codes have also been developed, to check for and correct errors such as those caused by these Soft Errors [174].

3.1.3.5. Human Factors

Failure Modes due to Human Factors are due to human (personnel) issues, and are the result of either unintended actions, such as accidents or neglect, or intended actions, such as the purposeful attacks that cause damage to, or reprogramming of, the FPGA. In this research program, the “Maintenance Induced” failures refers to the unintentional introduction of failure modes, while the “Security Breach” denotes the intentional introduction of failure modes.

Maintenance Induced Failures

In terms of the FPGA hardware, the "Maintenance-Induced" failure modes revolve around the accidental, damaging/destructive effects known as Electrostatic Discharge (ESD), and Electrical Overstress (EOS). These events refer to electrical damage to the FPGA chip and/or board, due to electrical discharges, excessive current/voltage, improper testing/protection of the system, etc. It is a concern that these events could damage the FPGA chip during operation, especially during maintenance and testing activities [175–177].

The issue of ESD and EOS can be mitigated by ensuring that the system can withstand ESD according to the industry standards (the ESD test requirements can be found in IEC 62003 [178], with guidance for performing these tests found in IEC 61000-4-1) [179], by having proper protections plans/programs for EOS/EOD [175,176].

The "Maintenance-Induced" failure could also include software (HDL code) failures, if changes to the HDL code are made during the maintenance periods. One such issue is known as the "Composition Problem", where the interactions of multiple IP cores cause them to interact in a problematic way. As an example, and IP core could interfere with the workings of a separate IP core, resulting in the system working improperly [3,145]. This failure mode could be intentional, if one attempts to sabotage the FPGA-based system via changes to logic in the maintenance period. However, it could be non-deliberate too, as new/updated IP cores may not function correctly together, once changes have been made to the workings of those IP Cores.

In general, failures such as those due to the "Composition Problem" are avoided through the use of a secure FPGA lifecycle, secure system architecture/communications, and trusted/verified FPGA tools, personnel and IP cores [3,144,145].

Security Breach

Security Breaches include attacks on both the hardware and software (HDL code) of the FPGA-based system. Physical attacks include Differential Power Analysis (DPA), where attackers are able to ascertain important system security keys by hooking up to the system [180,181]. HDL code breaches include FPGA

virus, at least one of which was created, that caused physical damage to the chip, and eventually destroyed it [144,182].

The unauthorized access to, and reprogramming of the FPGA is mitigated through the use of a secure lifecycle for the FPGA-based system [3,144,145].. Additionally, the use of OTP FPGAs (or disable reprogramming capability on FLASH FPGAs), is recommended, as the program cannot be altered. Additional methods to control access include countermeasures such as encryption, such as the Advanced Encryption Standard (AES), restricting/disabling the Joint Test Action Group (JTAG) access port, and restricting access of personnel to the FPGA itself [181]. The FPGA-based system should also inform control room personnel if attempts are made to access the FPGA bitstream, attempt to reprogram the FPGA, or access any data on the FPGA, while the system is in operation. Additionally, genera cyber security guidelines are applicable to FPGA-based systems [183].

3.1.4. Failure Set Mapping

A generic fault taxonomy from seen in the literature presents, an abstract, high-level taxonomy that could be applied to a wide variety of systems [143], while the FPGA FMEA categorization focused on the FPGA-specific faults. There are some similarities though, as both classifications include a “Design/Development” and “Operation” lifecycle stages. As the FMEA categorization is a lower-level classification, it can be mapped to the fault taxonomy in Ref [143].

As seen in Table 19, all of the FPGA FMEA categories can be mapped to the elementary fault classes. All of the software/HDL failures in that fall under the “Design Defect” cause would be considered as “Software Faults”. “Manufacturer Defects” would map to “Production Faults” and “Hardware Errata”, with the exception being that “Hardware Errata” only considers human-made causes, while “Production Faults” also include natural causes. The “Environmental Failures” map to “Physical Interference” (specifically “Natural, Hardware”, and the “Radiation-Induced” failures could be “Permanent (Perm)” or “Transient (Trans)”).

Table 19: FMEA Fault Category Mapping

FMEA Category	Elementary Fault Example
All Software/HDL Failures (Except "Design Security")	Software Faults
Manufacturer Defects Board Level (Design)	Production Defects and/or Hardware Errata
Environmental (Environmental Qualification)	Physical Interference *(Natural, Hardware (HW))*
Environmental (Radiation Induced Hard Errors) (Radiation Induced Soft Errors)	Physical Interference *(Nat., HW., Perm)* *((Nat., HW., Trans.)*
Stress/Aging	Physical Deterioration
Human Factors (Maintenance Induced)	Physical Interference *(Hardware, Non-Mal)* Input Mistakes *(Software, Non-Mal)*
Human Factors (Security Breach)	Intrusion Attempts *(Hardware, Mal)* Virus/Worms *(Software, Mal, Int)*
Design Security	Logic/Timing Bombs *(Software, Mal, Dev)*

Similarly, the "Stress/Aging" category would map to the "Physical Deterioration" example. The "Human Factors" category has broken up into the two failure sets, to cover five possible mappings, based on the "Dimension" (Hardware or Software), "Objective" (Mal or Non-Mal), and fault grouping (Interaction (Int) or Development (Dev). The "Maintenance Induced" failure sets would map to "Physical Interference" (hardware) and "Input Mistakes" (software), as they are both non-malicious. The "Security-Induced" failure set would map to "Intrusion Attempts" for hardware faults, "Virus/Worms" for Interaction software faults, and "Logic Bombs" for Development software faults, as all three of those faults are considered malicious.

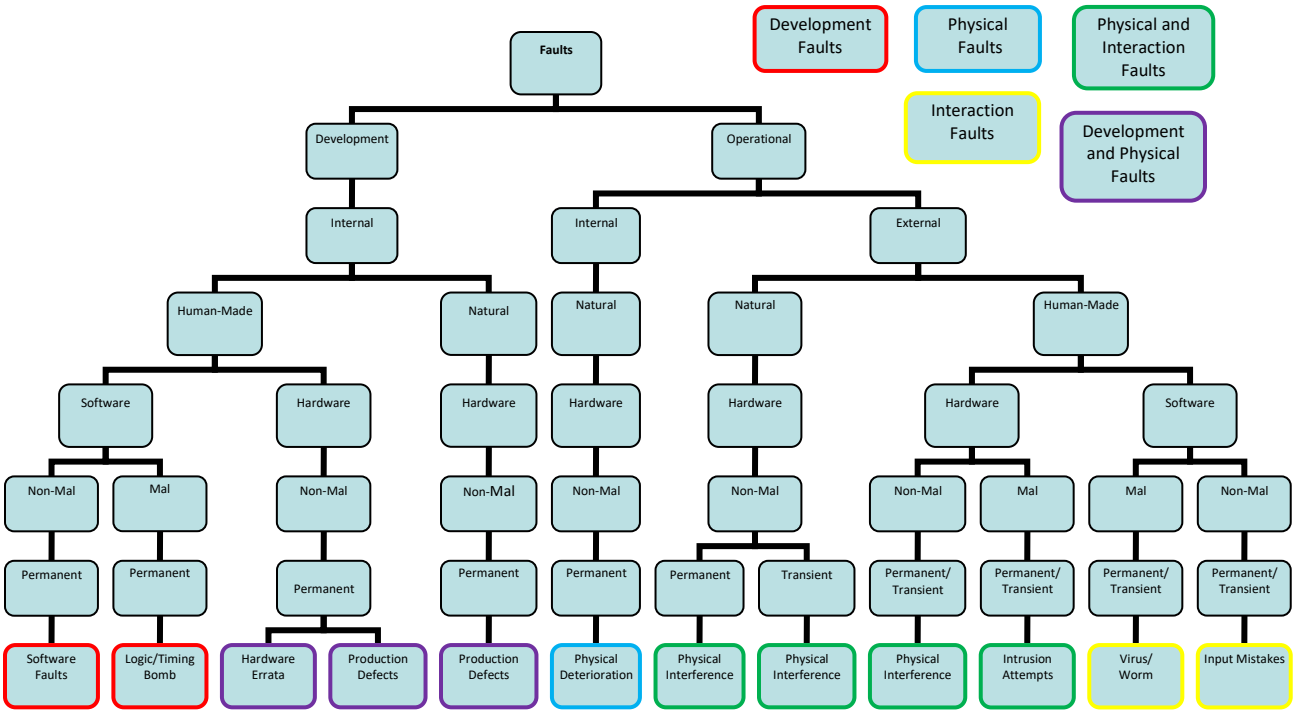


Figure 51: Elementary Fault Classes

This mapping allows for the low-level FPGA faults to be related to the high-level elementary fault classifications, providing more information on the possible cause(s) of each failure modes. These two categorizations (seen in Figures 64 and 65) would be useful in the design/development stage of an FPGA-based system, to identify potential failures for FPGA-based systems, allowing for avoidance or mitigation methods to be utilized.

3.2. OECD-NEA Digital Failure Modes Taxonomy

The Committee for the Safety of Nuclear Installations (CSNI) of the Organization for Economic Cooperation and Development Nuclear Energy Agency (OECD-NEA) published a document (NEA/CSNI//R(2014)16) entitled "*Failure Modes Taxonomy for Reliability Assessment of Digital Instrumentation and Control Systems for Probabilistic Risk Assessment*" [9]. The purpose of this taxonomy was to aid in the performance of Probabilistic Risk Assessment (PSA) of digital I&C systems. It represents the culmination of an international research project, done by the "Working Group on Risk Assessment", completed over the course of several years. The FPGA failure mode data can be re-

categorized using the OECD-NEA taxonomy guidance, creating a plug-in that allows the FPGA taxonomy to interface with the OECD-NEA Taxonomy.

3.2.1. OECD-NEA Taxonomy Introduction

This document sought to provide a failure modes taxonomy for digital I&C systems, at several levels of abstraction. These levels of abstraction (from the top down) are: System level → Division Level →I&C Unit Level → I&C Module Level → Basic Component level. It breaks down failure modes for these levels of abstraction, and then provides an example of modelling a generic digital (software-based) Reactor Protection System (RPS) using FTA, based on this taxonomy. This taxonomy includes both hardware and software failure modes, and is intended to support PRA analysis and modelling for digital systems.

This taxonomy focused on a generic digital I&C Reactor Protection System (RPS), consisting of a platform that performed the Reactor Trip System (RTS) and Engineered Safety Feature Actuation System (ESFAS) functions. This example system was a software-based system, meaning the taxonomy didn't directly consider FPGAs. It was stated in the report that future research could include "Complementation of the failure modes taxonomy with issues that were left out of the scope, e.g., control systems, networks, PLD technology (FPGA/ASIC)" [63]. As such, with a large amount of data from the FMEA on FPGAs, the OECD-NEA taxonomy could be applied.

3.2.2. Levels of Abstraction and Failure Effects

The OECD-NEA taxonomy considered five levels of abstraction. From the highest to lowest level, these were: System level, Division level, I&C Unit Level, I&C Module Level and Basic Component Level. These levels are defined below [9]:

System Level: The complete I&C system (the complete RPS, as seen in the taxonomy document).

Division Level: The physical separation of the I&C system, where each division is comprised of the I&C units.

I&C Unit Level: The elements that execute the specific functions that are necessary for the I&C system to carry out its specified purpose. These units are defined by the general system functions they perform, and consist of I&C modules.

Module Level: Hardware and software elements that support the specific tasks needed for the system to function. Examples would include I/O cards (hardware) or operating systems (software), and is comprised of the basic components.

Basic Component Level: The individual hardware components, such as CPUs, memory, etc., as well as the software used in those hardware components.

A graphical model of those levels of abstraction, when considering the digital (software-based) RPS from the OECD-NEA taxonomy is seen in Figure 52 [9]:

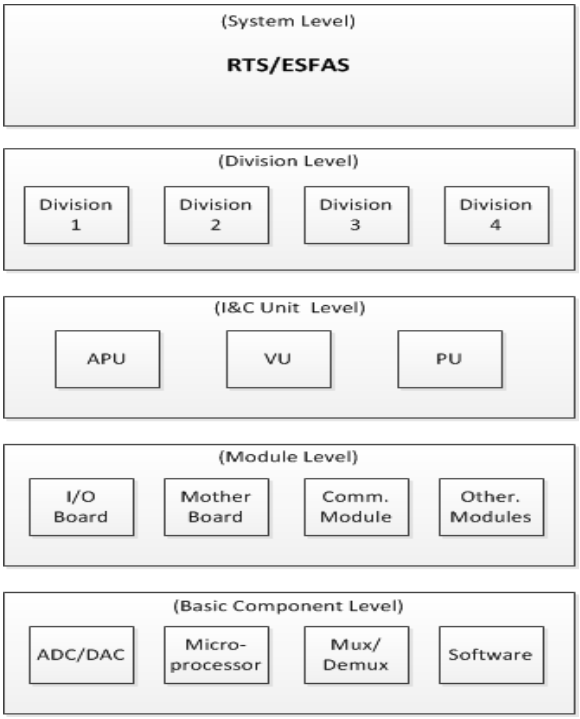


Figure 52: Simplified RTS/ESFAS Test System

In order to deal with complex systems, the failure modes for the “System”, “Division” and “I&C unit” levels are only considered from a functional point of view. This entails that there is no real distinction made between the hardware and software components, for those levels of abstraction.

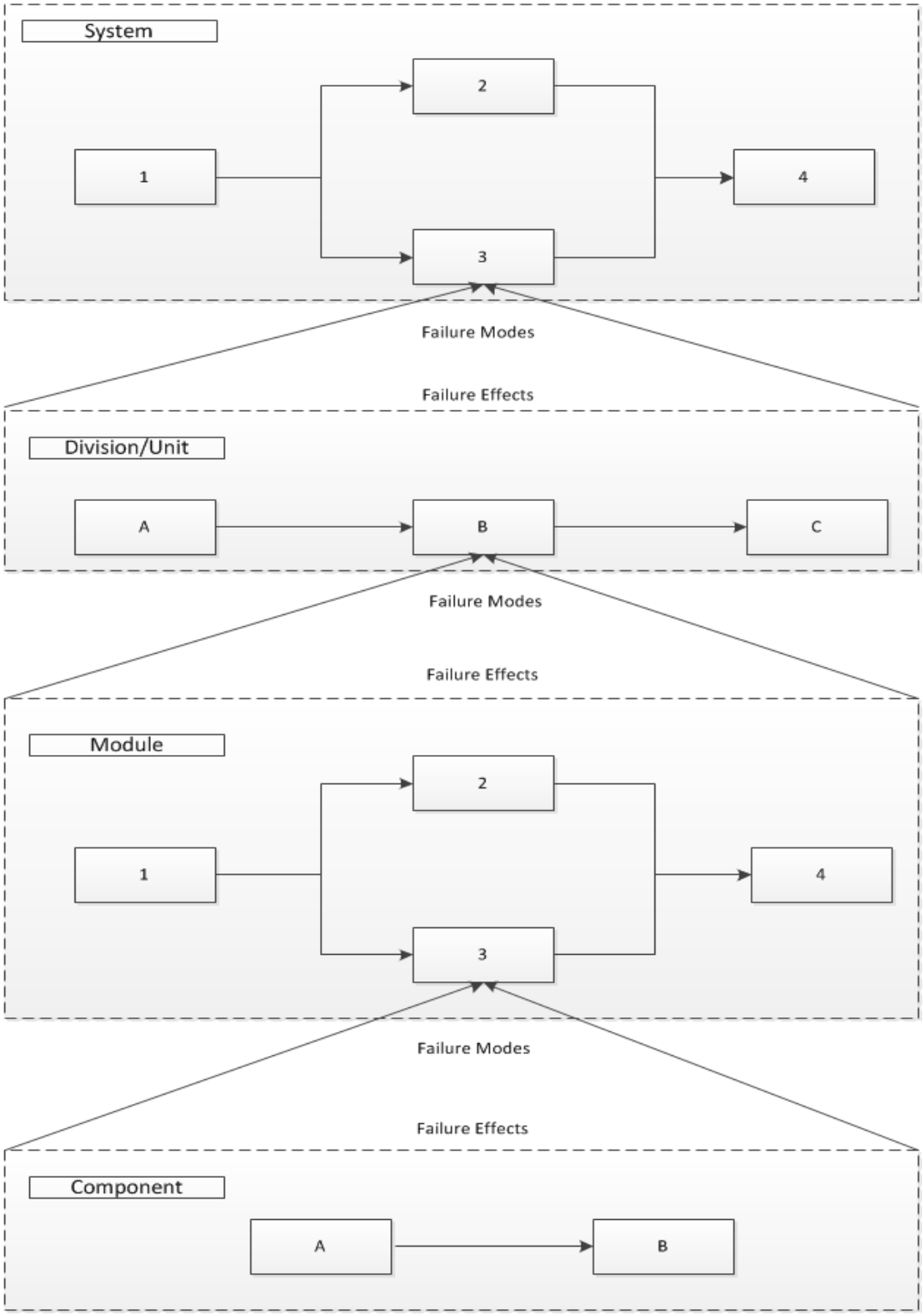


Figure 53: Relationship between Failure Effects and Failure Modes Between Levels of Abstraction

When the “Module” and “Basic Component” levels are considered, the failure modes are taken from both the functional and structural points of view. This means that there can be a distinction between the software and hardware components, for those levels of abstraction.

It was seen that the failure effects at a lower level of abstraction, will become the failure modes at the higher level of abstraction, as seen in Figure 53 [9,129].

3.2.3. Failure Propagation

According to the OECD-NEA taxonomy, there were two ways for failures to propagate through the I&C system, through cascade failures and common cause failures, as described below [9].

Cascade Failure Propagation: A failure (such as a systematic software failure or random hardware failure) in one component results in an incorrect output, which then becomes the wrong input for another component of the system. This error can cascade through the system, causing the overall system to fail.

Common Cause Failure Propagation: Multiple failures happen simultaneously in the I&C system due to the same event, as the same fault exists in multiple locations. This could include systematic software failures and hardware failures due to environmental factors.

3.2.4. Failure Effects Categories

The OECD-NEA Taxonomy considers two overall categories of Failure Effects; “Fatal”, and “Non-Fatal”. These two categories are each further broken down in two more classifications, to give a total of four categories, as discussed below [9].

Fatal: The unit stops functioning completely, and no longer provides an output.

Ordered Fatal: When the failure occurs, the unit outputs are forced into pre-set values.

Haphazard Fatal: When the failure occurs, the unit outputs are not forces into pre-set values, so the unit is in an unpredictable state.

Non-Fatal: The unit fails, but still performs computations and passes along incorrect output data.

Plausible Behavior: The incorrect outputs cannot be easily identified, given the current plant condition.

Implausible Behavior: The outputs from the unit are obviously incorrect.

3.2.5. Fault Uncovering

There are certain situations where the fault(s) of the digital I&C system will be uncovered. Two specific uncovering cases were considered:

1.) Uncovered Without Demand
- Failure detected with detection mechanisms
- Failure causes a Spurious Action

2.) Uncovered due to an Actual Demand
- The system failure occurs when the intended action is demanded

Detection methods can be broken down into “Online” and “Offline” detection. Online detection methods include self-monitoring (such as Built-In Self-Test (BIST)), and external monitoring. Offline detection includes periodic testing, during maintenance intervals. Overall, this leads to four possible uncovering situations, as seen in Figure 54 [9]:

- Spurious Action
- Demand
- Online Detection
- Offline Detection.

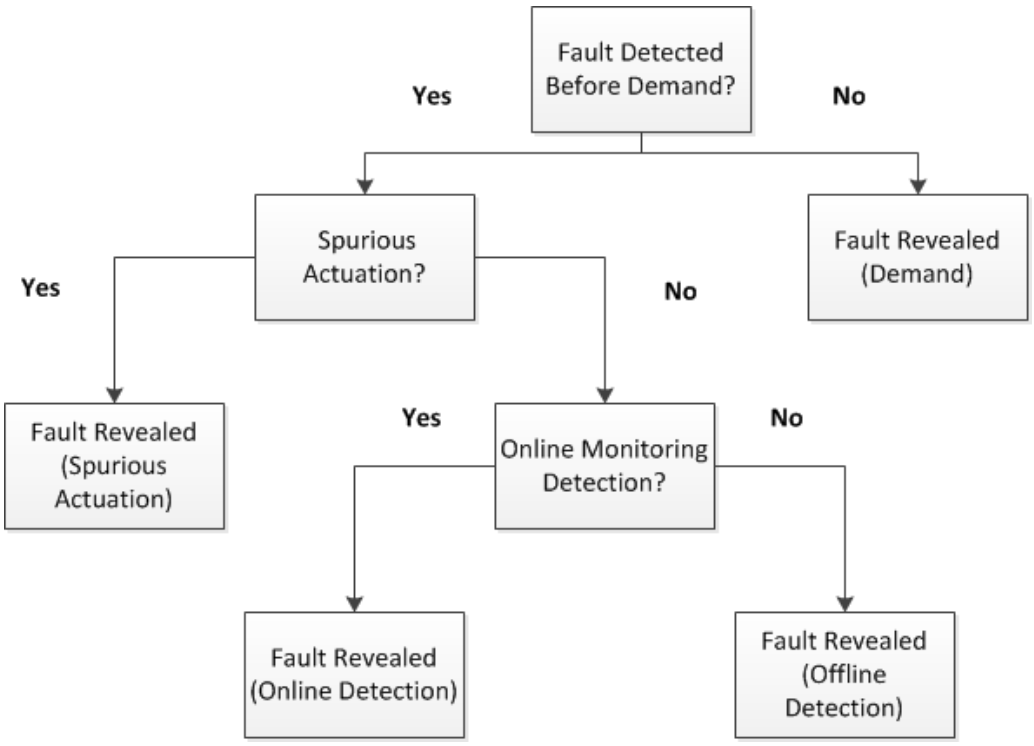


Figure 54: Fault Uncovering Situations for Digital I&C Systems

3.2.6. OECD-NEA Taxonomy Basis

Overall, the OECD-NEA Taxonomy considers four main elements:

1.) Fault Location
2.) Failure Effect
3.) Uncovering Situation
4.) End Effect (Maximum and most likely)

When considering the "End Effect", it would be identified during a specific analysis. In order to perform that analysis, three additional aspects can be included:

5.) Failure Origin
6.) Maximum possible end effect (assuming FTD features are not used or do not work)
7.) The most likely end effect (assuming FTD features are included and are effective).

The above 7 elements will be applied to the FPGA FMEA data and future FPGA-based system modelling.

3.2.7. OECD-NEA Categorization and the FPGA FMEA

The categorization used by the OECD-NEA taxonomy, discussed in earlier in this sub-section, provides a good way to categorize the failure modes based on the end effects of the failure mode, its uncovering situation and the level of abstraction in which that failure occurs. The basis for the OECD-NEA taxonomy is applied to create the modelling example shown in that taxonomy based on the failure mode data. However, one potential shortcoming of that taxonomy is that it does not provide any categorization for the cause of those failure modes, as was done in Ref [126].

Therefore, the framework from the OECD-NEA taxonomy is applied to the FPGA FMEA results and Failure Sets. The Failure Effects and uncovering situations are defined for the hardware and software (HDL) FPGA failure modes, as well as their potential end effects on the "I&C Module" and "System" levels of abstraction. The development of the FPGA taxonomy is laid out in sub-section 3.3.

3.3. FPGA Failure Mode Taxonomy

The development of the FPGA taxonomy followed the framework laid out in the OECD-NEA taxonomy, for the hardware and software (HDL) components. The FPGA taxonomy expands on the work done by the WGRISK by creating a "Sub-Component" level of abstraction, and creating a plug-in to interface the FPGA Taxonomy with the OECD-NEA taxonomy, allowing the FPGA Taxonomy to be useful to those involved with international working groups such as WGRISK. The development of the FPGA Taxonomy, as well as an example of it, is provided through sub-section 3.3.

3.3.1. Purpose of Developing the FPGA Taxonomy

The reasoning for performing hazard analysis is said to be to *"explore and identify conditions that are not identified by the normal design review and testing process"* [36]. This includes the identification, avoidance, evaluation and resolution of hazards in all phases of the system lifecycle. Theses hazards are

caused by failure modes, which must be identified and evaluated. Therefore, the FPGA taxonomy presented in this paper provides a means of identifying, categorizing and modelling the failure modes for use in hazard analysis, during the design and review of FPGA-based I&C systems. This would provide a basis for the decisions on engineering and safety based on system review criteria [184].

Regarding the OECD-NEA taxonomy, it was stated that in that document that “An activity focused on the development of a common taxonomy of failure modes is seen as an important step towards standardised digital instrumentation and control (I&C) reliability assessment techniques” and “The taxonomy will be the basis of future modelling and quantification efforts” [9]. These statements from the OECD-NEA underscore the importance of having failure modes taxonomy for the analysis and assessment of digital systems. As stated previously, the OECD-NEA taxonomy considered a software-based system, and stated that the development of an FPGA taxonomy would be a source of future work on this topic.

Furthermore, the OECD-NEA taxonomy document laid out certain criteria, in which the taxonomy was intended to meet. The only criteria that was designated as “Not Met”, was entitled “*Should capture defensive measures against fault propagation (detection, isolation and correction) and other essential design features of digital I&C*”, and was again left as a topic of future work [9]. In this FPGA Taxonomy, potential mitigation methods were also included, for the example failure modes/failure categories. Therefore, this FPGA taxonomy fulfills two important areas of future work, as described by the OECD-NEA taxonomy.

In terms of the recent scientific/technical literature It has been seen that there has been a great deal of work put into the design, verification and validation (V&V) and safety analysis of FPGA-based control systems in general [185,186], and specifically in the case of the nuclear industry [187–189]. The unique properties of FPGAs present certain challenges during the safety analysis process, which may be different challenges than with analog systems or software-based digital systems. Future V&V and safety analysis could be improved upon, if the FPGA failure mode data was properly compiled, categorized and analyzed, such as was done in the OECD-NEA taxonomy.

3.3.2. Taxonomy Integration

The OECD-NEA taxonomy was developed with software-based systems in mind, but the same framework can be extended to FPGA-based systems. This section discusses the shortcomings of the OECD-NEA taxonomies as it pertains to FPGA-based systems, and provides a detailed discussion of the proposed “Logic Process”.

3.3.2.1. Application to FPGA-Based Systems

Upon inspection of the test system, it is seen that an FPGA-based system and taxonomy would not be significantly different from the software-based system and taxonomy, using the current levels of abstraction. In an actual system, the failure modes at the “System”, ‘Division” and “I&C Unit” levels would not differ from a software-based to an FPGA-based system. For example, for the system level failure modes for an RTS, the system could either have a ”Missed Trip”, “Spurious Trip”, or a “Partial/Delayed Trip”, according to the OECD-NEA taxonomy [9], and these would not change with an FPGA-based system. However, there would be two main changes in an FPGA Taxonomy, at the “Module” Level and “Basic Component” Level, respectively.

Module Level: At this level, the OECD-NEA Taxonomy considers separate hardware and software failure modes. However, as the FPGA does not run actual software, and does not have an Operating System (OS), failure modes like “OS Freeze” or “OS Crash” would not apply.

Basic Component Level: In an FPGA-based system, there would be no “Microprocessor” or accompanying “Software”, so these two entries would be replaced by “FPGA” and “HDL Code”, respectively.

However, that requires swapping out parts of the OECD-NEA taxonomy for different systems, as the original taxonomy does not include devices such as FPGAs. This issue could be rectified, by modifying the “Basic Component” level of the taxonomy, to work with all forms of digital technology. The use of the OECD-NEA taxonomy for the creation of an FPGA taxonomy is important, as the OECD-NEA methodology is internationally recognized, and is used by working groups and researchers in this field. Following the protocol/methodology of the OECD-NEA taxonomy allows the FPGA taxonomy to retain its

importance to those working groups, and ensures that the quality of the FPGA taxonomy is up to that international standard.

3.3.2.2. Application to FPGA-Based Systems

Modelling FPGA-based systems with the original OECD-NEA taxonomy would not be possible, as FPGAs would not fit into its framework .The method proposed in this paper to extend the OECD-NEA taxonomy to incorporate FPGA-based systems is through the use of the “Logic Process”. This block would replace all digital logic hardware and software/HDL at the “Basic Component” level. For example, both “microprocessor” and “software”, as well as “FPGA” and “HDL” code could be represented by a single “Logic Process” block, as shown in Figure 55. Components like the ADC/DAC, MUX/DEMUX, and any additional components at the “Basic Component” Level identified in the OECD-NEA taxonomy remain unchanged.

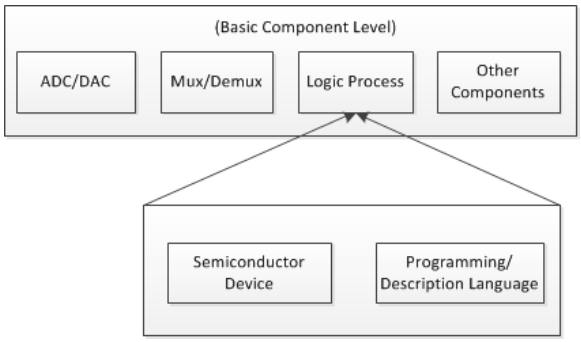


Figure 55: Extended Taxonomy Using “Logic Process”

In Figure 55, the “Logic Process” block represents all potential digital hardware technologies (microprocessor, FPGA, etc.), as well as all software/HDL components. This extends the OECD-NEA taxonomy to incorporate FPGA and other forms of control technologies, for all five levels of abstraction. Using this extended taxonomy creates a plug-in for other forms of digital technology, and allows for the modelling of the FPGA failure modes to be performed within the context of the OECD-NEA framework. This plug-in allows for the FPGA failure mode data to be used within the OECD-NEA taxonomy framework, as presented in this paper, or by itself with the failure mode data given in Reference [126].

3.3.3. Sub-Component Level of Abstraction

A “Sub-Component” (SC) level of abstraction was created to account for the failure modes of FPGAs, including the Failure Set data from Figure 6. The FPGA failure data collected during the FMEA affects the FPGA chip/board or the actual HDL code. However, these failures would not be included in the OECD-NEA Taxonomy, as it stopped at the “Basic Component” level, and did not consider failures beyond that level in detail. To remedy this, a “Sub-Component” (SC) level of abstraction was proposed, in order to account for the effects of the different FPGA and HDL code failures on the example system. The FPGA Taxonomy focuses on the “Sub-Component” and “Basic Component” levels, to tie together the FPGA FMEA data and the OECD-NEA Taxonomy example system, by plugging the “Sub-Component” level into the “Logic Process”, developed in sub-section 3.3.2.

The “Sub-Component” level of abstraction is shown in Figure 56. The “Sub-Component” Level is a potential way of demonstrating the effects of the failure modes of the FPGA system. This will breakdown the “FPGA” and “HDL” entries at the “Basic Component” level, to their most basic hardware and “software” components, respectively. This allows for the “Failure Categories”, presented in Figure 50, to be used to construct the FPGA failure mode taxonomy, based on the OECD-NEA template. As seen in Figure 56, the FPGA Chip can be broken down into many categories based on the hardware (blue) and “software” (red) modules that make up a (configured) FPGA.

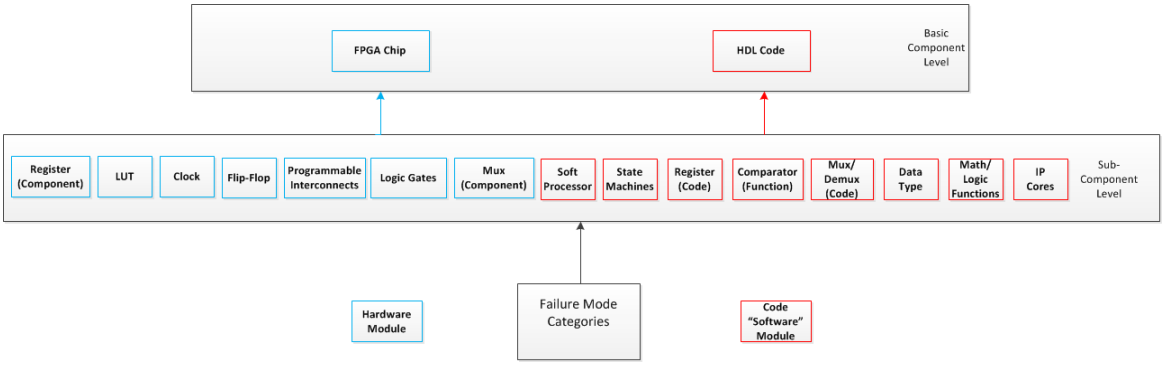


Figure 56: Relationship Between “Basic Component”, “Sub-Component”, and “Failure Categories”

This will allow for the application of the failure categories, shown in Figure 50, to the taxonomy, to provide detailed information about the failure modes and failure effects regarding the hardware and software components of the FPGAs and FPGA-based systems. It should be noted that the “Soft

Processor" is a form of "IP Core"; however it was included here separately, due to the inclusion of the "Processor" in the OECD-NEA Taxonomy.

This additional level of abstraction will allow us to re-construct the failure mode data in this paper, and to provide information on classifications, fault locations and uncovering situations that are given in the taxonomy. Furthermore, the lifecycle information of the failure modes is included in the new FPGA taxonomy. Lastly, the inclusion of the sub-component level allows for the demonstration of how the low-level failure modes obtained through the FPGA FMEA could cascade up through all the levels of abstraction, potentially causing a failure at the division or system level.

3.3.4. Sub-Component Hardware Taxonomy

The "Sub-Component" taxonomy for the hardware sub-components is discussed here. Figure 57 gives a representation of the FPGA Chip and Board, along with the hardware sub-components, and an example of failures that affect those components. In Figure 57, the FPGA chip is divided into the three underlying components; FPGA I/O, Configurable Logic Blocks (CLB), and the Programmable Interconnects. The CLB is further subdivided by its sub-components; the Look-Up Tables (LUT) Register/Flip Flops, Mux's (discussed in the OECD-NEA taxonomy).

The effects of failures of the inputs from the clock, the FPGA board (which the FPGA chip itself would reside), as well as inputs into the FPGA board itself, are also considered. All of these components (except for the Mux) were then assigned an example failure mode, taken from the FPGA FMEA research. The sub-components and failure data shown in Figure 56 are reconstructed in Table 21, which also includes information tying the failure modes back to the fault classes shown in Figure 51, and the failure mode categories in Figure 50, along with the potential effects of those failures on the "Basic Component" and "Sub-Component" levels of abstraction.

Table 21 also considers the elementary fault classes from Ref. [143], listed in parenthesis below the information taken from Figure 50. The column "Failure Set" includes the elementary fault example mapping from Table 19. The "Cause" column denotes if the failure from the "Development", "Physical"

or “Interaction” groups. Lastly, the “Lifecycle” column states if the failures is in the “Development” or “Operational” portion of the lifecycle.

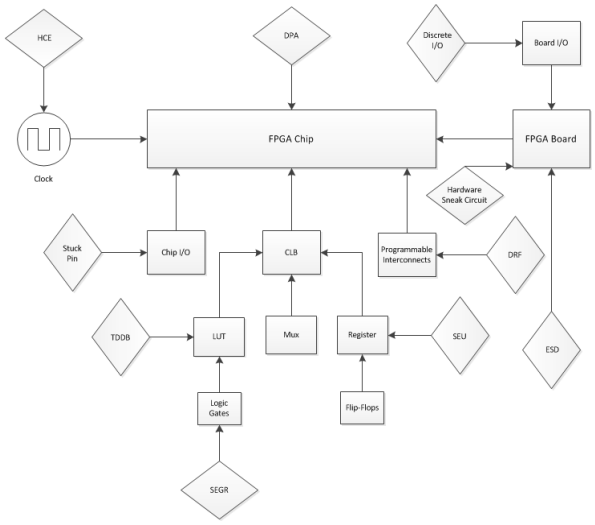


Figure 57: FPGA Chip/Board Hardware Failures

The hardware failures in Table 21 include Hot Carrier Effects (HCE), which will slow down the clock period, and Single Event Upsets (SEU), which will invert a data bit that is stored in a memory element, such as a register. A full list of all acronyms and definitions for the failures can be found in Appendix A. Failures that affect the interconnects, such as Electromigration (EM), could result in either fatal or non-fatal errors, depending on the location where the failure occurred. If the EM failure occurs in an interconnect carrying only part of the data (i.e. it is one of several inputs that will be summed and then output) denoted “EM 1”, the failure will be non-fatal. If the EM failure occurs in an interconnect that is the only input or output path for the signal, then the failure would be fatal, as seen with “EM 2”. Similarly, an error in a CLB (such as Time Dependent Dielectric Breakdown (TDDB)) could be non-fatal, if there are many logic blocks performing computations in parallel, as in the case of “TDDB 1”. However, if it is the only logic block leading to an output, then it could be considered as a fatal error, denoted by “TDDB 2”. For Non-Fatal failures in both cases, the effects could be either “Plausible” or “Implausible”, depending on the failure location, logic process, and combination of inputs. These situations are seen in Figure 58.

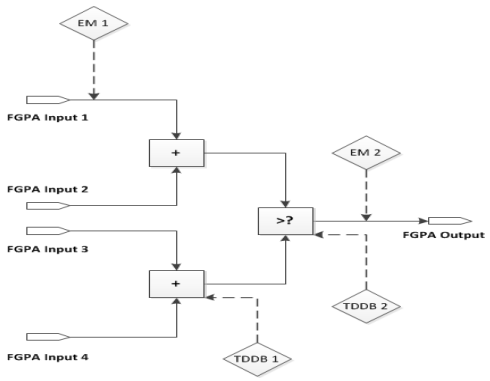


Figure 58: Effects of failures of CLBs and Programmable Interconnects

In the case of registers (storage elements), there is the vulnerability to SEU. These failure could invert a stored memory value (0 →1 or 1→ 0), and that could affect the output of the system or sub-system. The effect that this inverted bit would have on output of the values would depend on which bit in memory is inverted. For example, for an 8-bit signal, shown in Table 20, if the Most Significant Bit (MSB), typically the leftmost bit is flipped, the difference is much greater than if the Least Significant Bit (LSB) is flipped, typically the rightmost bit. In Table 20, the first example is a binary input of "10101010", corresponding to a value of "170" in decimal notation (base 10). If the MSB is flipped (1 → 0), then the resulting binary signal is "00101010", or "42" in decimal notation.

Table 20: Effects of SEU on Register Storage Values

Intended Numeric Value		**Bit Flipped (MSB/LSB)**	**Erroneous Numeric Value**	
Base 2	**Base 10**		**Base 2**	**Base 10**
10101010	170	MSB	00101010	42
10101010	170	LSB	10101011	171
00101010	42	MSB	10101010	170
00101010	42	LSB	00101011	43

Table 21: Sub-Component Level Failure Modes and Failure Effects (Hardware)

Failure	Fault Location	SC Level Effect	BC Level Effect	Failure Type	Failure Set	Cause	Lifecycle	Mitigation
TDDB	LUT	Destruction of FPGA LUT	No Output/Incorrect Output	Fatal or Non-Fatal	Aging Process (FPGA Chip) (Physical Deterioration)	Stress/Aging (Physical)	Operation (Operational)	Design techniques to improve oxide lifetime[39] MTBF for TDDB[40]
SEGR	Logic Gate	Destruction of FPGA logic gate	No Output/Incorrect Output	Fatal or Non-Fatal	Radiation-Induced Hard Errors (Physical Interference)	Environmental (Physical/Interaction)	Operation (Operational)	SEGR Sensitivity Testing[41] Protective Circuits[43] Power De-Rating[42]
SEU	Register (Storage Element)	Temporary Bit Upset in Memory Element	Incorrect Output	Non-Fatal	Radiation-Induced Soft Errors (Physical Interference)	Environmental (Physical/Interaction)	Operation (Operational)	TMR[33] Error Detection and Correction (EDAC)[36]
SET	Register and/or Logic Gates	Transient pulse through logic/registers/ output	Incorrect Output	Non-Fatal	Radiation-Induced Soft Errors (Physical Interference)	Environmental (Physical/Interaction)	Operation (Operational)	Spatial or Temporal Redundancy[43,44] Circuit Freezing[45]
Substrate Breakdown (High Temp.)	FPGA Chip/Board	Destruction of FPGA Device, Package or Board	Board/Chip destroyed, no output	Fatal	Environmental Qualification (Physical Interference)	Environmental (Physical/Interaction)	Operation (Operational)	Environmental Qualification Procedures[1,46]
Stuck-Pin	FPGA I/O	FPGA Pin Stuck ("1" or "0")	Incorrect Output	Non-Fatal	Chip and Board Production Defects/ Hardware Errata)	Manufacturer Defects (Development/Physical)	Design (Development)	Detect/control damaged/disconnected pins[22,23]
HCE	Clock	Reduction in Clock Frequency	Delayed Output	Non-Fatal	Aging Process (Clock) (Physical Deterioration)	Stress/Aging (Physical)	Operation (Operational)	Monitor Clock Skew[22,23] MTBF for HCE[40]
ESD	FPGA Chip/ Board	ES damage to the FPGA Chip/Board	Board/Chip destroyed	Fatal	Maintenance Induced (Physical Interference)	Human Factors (Physical/Interaction)	Operation (Operational)	Electrostatic Protection Program[26,27]
Differential Power Analysis (DPA)	FPGA Chip/Board (FPGA Logic)	Secret Cryptographic keys are recovered (unauthorized)	System security and/or IP is compromised	Non-Fatal	Security Breach (Intrusion Attempts)	Human Factors (Physical/Interaction)	Operation (Operational)	Secure Lifecycle[29] Side Channel Attack[31,32,47] Countermeasures Cyber Security Guide[48]
Hardware Sneak Circuit	FPGA Chip/Board	Spurious or Missed Actuation	Incorrect Output or No Output	Fatal or Non-Fatal	Sneak Circuit (Production Defect)	Design Defect (Physical)	Design (Development)	General and FPGA-Specific Sneak Circuit Analysis[49-52]
Data Retention Failure (DRF)	Programmable Interconnect	Interconnect Self-Healing	Incorrect Output	Non-Fatal	Bit Error (Physical Deterioration)	Stress/Aging (Physical)	Operation (Operational)	Copies of Programming Data[22,23] MTBF for Interconnects[22,23]
Discrete (Digital) Input	Board I/O	Failure of Input on the FPGA Board	No output (resulting from no input)	Fatal	Board Level (Production Defects/ Hardware Errata)	Design Defects (Development/Physical)	Design (Development)	Standards for fault models, diagnostic coverage and mitigation[53,54]
Hardware CCF	FPGA Chip/Board	Common Cause Failure (Hardware)	No output	Fatal	Common Cause Failure (Production Defects)	Design Defects (Development/Physical)	Design (Development)	Diversity and Defence in Depth[8] Requirements from technical standards[11,55,60] CCF Analysis[56,57]

Table 22: Uncovering Situation Examples for Sub-Component Level (Hardware)

Uncovering Situation	Fault Tolerance Feature
Online detection mechanisms	**HCE**[159,160]**:** Revealed by monitoring clock skew
Offline detection mechanisms	**EM**[159,160]**:** Revealed by periodic testing of the FPGA
Latent revealed by demand	**TDDB**[159,160]: Damage to the LUT causes incorrect calculations, not detected until actuation required
Triggered by demand	**ESD**[175,176]**:** FPGA fails due to Electro-Static Discharge from another component (due to Maintenance errors) Failure is not detected
Spurious actuation	**SEU**[166,167]: Memory Upset causes values to read above a setpoint, causing a spurious trip Failure is detected (EDAC Methods)
Undetectable	**Hardware CCF** [9]**:** Undetectable

Following in the same vein as the OECD-NEA taxonomy, Table 22 gives examples of the uncovering situations for the failure modes seen in Figure 50 and Table 21. It should be noted that (hardware) CCF was not explicitly shown in the example figure; however it is included in the corresponding tables (Table 21 and Table 22, respectively). Additionally, CCF was stated to be "undetectable" in the OECD-NEA taxonomy [9].

3.3.5. Sub-Component HDL Code Taxonomy

The same principle in sub-section 3.3.4 is applied to the HDL Code failure modes (the "software" component of the FPGA). The example in this case is loosely developed with a reference to the Overtemperature and Overpressure trip parameters in the Westinghouse AP1000 documentation, and is presented in Figure 59 [190]. Several software/logic errors are seen, including the use of a latch instead of a register in the synchronization chain, mathematical error due to "Arithmetic Overflow", and a "Stuck Output" (output will not update, even with changing inputs) from the soft processor core that

calculates the TSP. The Software Sneak circuit reveals that a sneak circuit is created that could bypass certain important functions, resulting in a “Missed Trip” or “Spurious Trip”. A second IP core, this time a COTS IP Core (considered to perform some generic signal processing function) may contain failure, or the specifications of that IP core may not be appropriate. Composition Errors (either malicious or non-malicious) could cause the IP cores the IP cores to alter one another, or interfere with each other’s functions. Design Tool Subversion could input code that effects the clock signal (such as if a dynamic clock frequency is used), distorting proper chip timing. Finally, an FPGA virus could cause damage to the FPGA chip (such as a short circuit), resulting in no output being sent. All of these failures could affect the output of the FPGA logic.

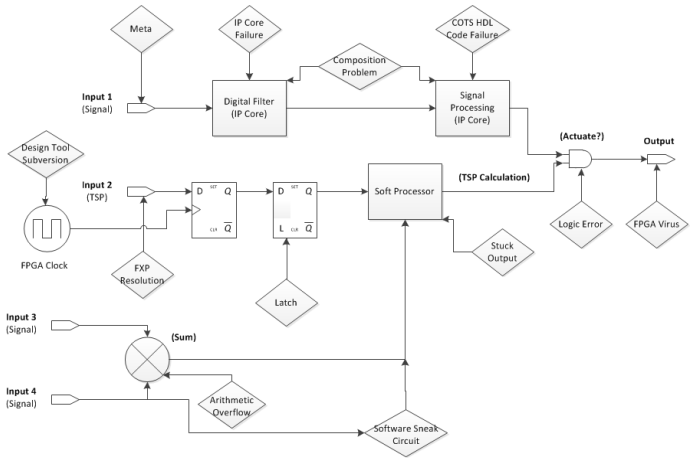


Figure 59: FPGA “Software” Failures (Parameter Trip)

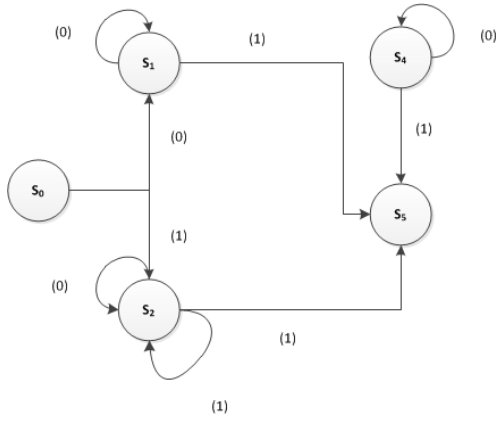


Figure 60: FPGA “Software” Failures (State Machine)

A smaller example shown in Figure 60 shows certain potential errors in the FPGA state machine. As can be seen, the state machine could get caught in an Endless Loop (Infinite Loop), as the state S_2 is encoded

in such a way that a value of "1" has two possible paths. In this case, the state machine could continuously loop back into itself, causing it to hang. A second fault in the state machine is seen with S_4, where the state is unreachable. State Machines often see use in FPGA-based systems, and as such was given consideration in the FPGA taxonomy. The information regarding the software sub-components, failure modes, and failure categories was compiled and displayed in Table 23.

It should be noted that the failure mode(s) and uncovering situations of individual IP cores are dependent on the functionality of each individual IP Core, as stated in the OECD-NEA taxonomy. In Figure 59, it was assumed that the filter (such as a lead-lag filter) was implemented (digitally) on the FPGA, using an IP Core. A failure in that core could affect the filtered output, with failures such as "Stuck Output", "No Output", "Delayed Output", etc.

The IP Core in this case was used as an example. As there are potentially numerous IP Cores available by different vendors, it is not practical to discuss failure modes for all the individual IP Cores (with the aforementioned exception of the Soft Processor). As in the case of the hardware sub-components, the information regarding uncovering situations is given in Table 24, with software CCF being "undetectable".

Table 23: Sub-Component Level Failure Modes and Failure Effects (Software)

Failure	Fault Location	SC Level Effect	BC Level Effect	Failure Type	Failure Set	Cause	Lifecycle	Mitigation
Endless Loop	State Machine	State Machine caught in an endless loop	No Output/Stuck Output	Fatal	State Machine (Software Fault)	Design Defects (Development)	Design (Development)	WDT[54, 60] Return to pre-defined state[22,23]
Unreachable States	State Machine	State(s) cannot be reached as intended	No Output or Incorrect Output	Fatal or Non-Fatal	State Machine (Software Fault)	Design Defects (Development)	Design (Development)	State Machine Hazard Analysis[54] Return to pre-defined state[22,23]
COTS HDL Code Failure	FPGA Logic	Function Dependent HDL Error	No Output or Incorrect Output	Fatal or Non-Fatal	COTS (Software Fault)	Design Defects (Development)	Design (Development)	FPGA Dedication[21] Software Dedication[63,64]
Stuck Output	Soft Processor	SP core stops updating	No Output/Stuck Output	Fatal	Soft Processor (Software Fault)	Design Defects (Development)	Design (Development)	FPGA V&V[1,60] Software V&V[11,65,66]
Math Error	EF (Math)	Arithmetic	Incorrect (Math)	Non-	Logic Errors (HDL)	Design Defects	Design	HDL Code

(Arithmetic Overflow)		overflow leads to calculation error	Output	Fatal	(Software Fault)	(Development)	(Development)	V&V[1,22,23] International Standard[60]
Logic Error (Comparator Error)	EF (Logic)	Error in comparator leads to logic error	Incorrect (Logic) Output	Non-Fatal	Logic Errors (HDL) (Software Fault)	Design Defects (Development)	Design (Development)	HDL Code V&V[1,22,23] International Standard[60]
Fixed Point Resolution Error	Data Type	Low resolution of FXP value	Incorrect setpoint	Non-Fatal	Input and Data Type (Software Fault)	Design Defects (Development)	Design (Development)	Ensure proper calculation/ verification of all FXP values[63]
Software Sneak Circuit	FPGA Chip/Board	Spurious or Missed Actuation	Incorrect Output or No Output	Fatal or Non-Fatal	Sneak Circuit (Production Defect)	Design Defect (Development)	Design (Development)	General and FPGA-Specific Sneak Circuit Analysis[49-52]
Latch	Registers	Unintended asynchronous signals	Unknown/Random Output	Non-Fatal	Clock/Timing (Software Fault)	Design Defects (Development)	Design (Development)	Implement Registers[22,23,60] Complete logic statements and sensitivity lists (accidental latch)[22,23] Static Timing Analysis[22,23] Timing Simulations[1,60]
Design Tool Subversion	FPGA Logic/Timing (FPGA Synthesis Tool)	Unauthorized HDL code synthesized	Unauthorized access, incorrect outputs or device damage	Fatal or Non-Fatal	Design Security (Logic/Timing Bomb)	Design Defects (Development)	Design (Development)	Secure Lifecycle[29] Trusted Tools and IP Cores[6,29]
FPGA Virus	FPGA Logic/ Circuitry	Internal signal conflict in FPGA	Device damage/ destruction	Fatal or Non-Fatal	Human Factors (Virus/Worm)	Security Breach (Interaction)	Operation (Operational)	Secure Lifecycle[29] Bit stream Protection[6,29] Secure Architecture and comms[6,29] Trusted Tools and IP Cores[6,29]
Composition Problem	IP Core	Updated IP Cores alter or interfere with each other	Incorrect outputs or device damage	Fatal or Non-Fatal	Human Factors (Input Mistakes)	Maintenance-Induced (Interaction)	Operation (Operational)	Secure Architecture and comms[6,29] Trusted Tools and IP Cores[6,29]
IP Core Failure	IP Core	Function Dependent	Function Dependent	Fatal or Non-Fatal	Maintainability (Software Fault)	Design Defects (Development)	Design (Development)	FPGA V&V[1,60] Software V&V[11,65,66]
Software CCF	FPGA Chip/Board	CCF due to Software	No Output	Fatal	Common Cause Failure (Software Fault)	Design Defects (Development)	Design (Development)	Diversity and Defence in Depth[8] Requirements from technical standards[11,55,60] CCF Analysis/ specification[56,57,67]

Table 24: Uncovering Situation Examples for Sub-Component Level (Software)

Uncovering Situation	Fault Tolerance Feature
Online detection mechanisms	**Endless Loop** [37,133]**:** State Machine Endless Loop caught by WDT. State Machine returned to pre-defined state
Offline detection mechanisms	**Unreachable States** [133]: Unreachable states found and corrected by using State Machine Hazard Analysis
Latent revealed by demand	**FXP Resolution Error** [38,48]**:** Low Resolution for the FXP data type leads to an inaccurate TSP, causing the demand to fail when it should actuate
Triggered by demand	**Metastability** [65,131]**:** Lack of synchronizing registers causes metastable signals to propagate through the system. Failure is not detected
Spurious actuation	**Logic Error (Comparator)** [65,146]**:** Incorrect comparator logic (i.e. ">" instead of "<") causes a spurious trip. Failure is detected
Undetectable	**Software CCF** [9]**:** Undetectable

3.3.6. FPGA Taxonomy Demonstration

The OECD-NEA taxonomy was demonstrated using the aforementioned RTS/ESFAS test system. For the FPGA taxonomy demonstration, the FMEAs were used to create the FPGA taxonomy, for hardware and "software", at the "Basic Component" and "Sub-Component" levels of abstraction. Filling out these levels of abstraction links the FMEA and the PRA modelling information given in the example in the OECD-NEA document, to demonstrate how the failures of the configured FPGA and HDL code would affect the overall system. Sub-section 3.3.6.2 will cover the "Basic Component" level, while Sub-section 3.3.6.3 presents the demonstration for the "Sub-Component" level hardware and software FMEA. Lastly, a modelling example using fault trees is provided in Section 3.3.6.4

3.3.6.1. Example System and Failure Events

An example of the assumed modules of a generic digital RTS/ESFAS, that was used as the example system in the OECD-NEA taxonomy, and in this research work, is seen in Figure 61 [9].

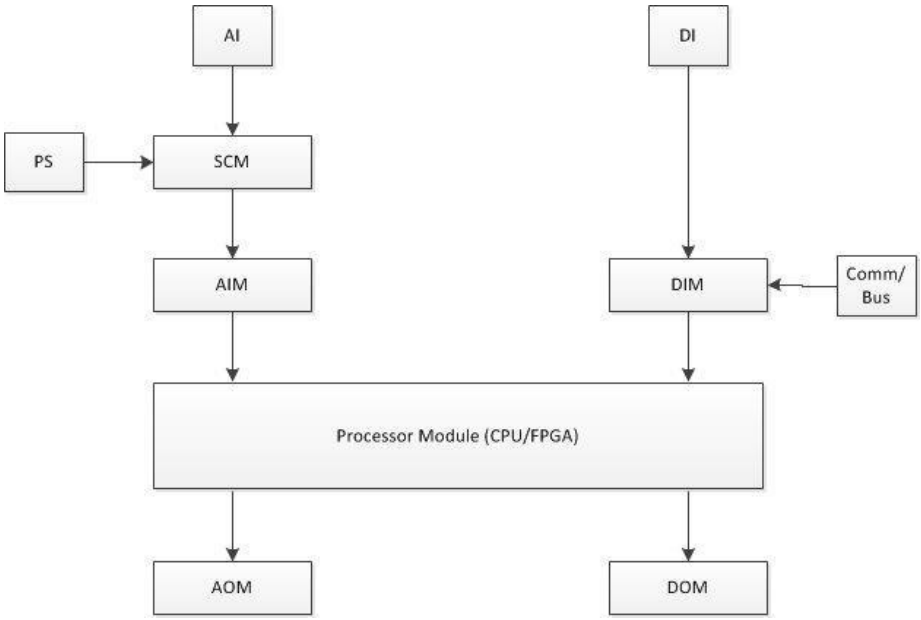


Figure 61: Modules Included in the Example RTS/ESFAS System

This example system includes Analog/Digital Inputs (AI/DI), Power Supply (PS), Analog and Digital Input Modules (AIM/DIM), Analog and Digital Output Modules (AOM/DOM), Signal Conditioning Modules (SCM), and Communication and Bus links (Comm/Bus). The central module, would represent the processing module, and could utilize software-based and/or FPGA-based solutions. Other modules, such as the AIM, DIM, AOM and DOM could also include FPGAs or software-based control technologies. In the OECD-NEA taxonomy, the AIM was expanded during the "Basic Component" Level demonstration, so the same module was used in this FPGA Taxonomy demonstration.

3.3.6.2. Basic Component Level Demonstration

It was determined to start with the "Basic Component Level" for the FPGA taxonomy, and work from there. At this level, the example failure modes for the components other than the FPGA/microprocessor and HDL/software would again be the same, so they were not considered here. Examples relevant to the FPGA and/or HDL code based on the OECD-NEA examples are shown in Table 7. The "Sub-Component"

taxonomy demonstrations, will build up to the "Basic Component" levels, and then up to the total "System" level, to demonstrate the effect of the FPGA and HDL failures on the overall system. The Analog Input Module (AIM) was used in Table 7, as it was the module used in the "Basic Component" example in the OECD-NEA Taxonomy. The effects of FPGA failures on the AIM can then be related to the digital RTS/ESFAS.

Table 25: Basic Component Level FPGA FMEA for the OECD-NEA AIM

Failure Mode	Failure Mode Detection		Failure Effects on AIM	Comments
	Application HDL	WDT		
Hardware/Software CCF	(Undetectable)	(Undetectable)	Incorrect AIM Output (All Channels Fail)	CCF would cause all the AIM channels to fail
Hardware/Software Sneak Circuit	No	No	Incorrect AIM Output	Spurious or Missed Output (SCA)
Random or Incorrect FPGA Outputs	No	No	Incorrect AIM Output	May not be detected (if plausible failure)
FPGA logic stuck internally, no outputs sent (No Output)	No	Yes	No AIM Output FPGA Reset or Predefined state (Detection)	Logic issues may be detected with WDT, resetting the FPGA, or sending it to a predefined state
FPGA stops updating outputs (Stuck Output)	No	Yes	Incorrect AIM Output FPGA Reset or Predefined state (Detection)	Logic issues may be detected with WDT, resetting the FPGA, or sending it to a predefined state
Delayed Outputs	No	Yes	Delayed AIM Outputs FPGA Reset (Detection)	Timing errors cause FPGA to output data too slowly, delaying AIM outputs
Timing Error (Fast or Slow)	No	Yes	Missed AIM Outputs FPGA Reset (Detection)	Timing errors cause FPGA to output data too quickly, AIM outputs are missed

3.3.6.3. Hardware FMEA and Sub-Component Demonstration

The OECD-NEA Taxonomy gives a four-step process to compress the hardware failure modes into their functional effects on the system. The steps are as follows [9]:

1.) The failure effects are assigned to different failure modes of the RPS test system, based on the failure modes taxonomy at the module level. This allows for the uncovering situations and the functional impacts to be described for the test system.
2.) Failure mode categories are defined based on the failure effect(s) and uncovering situation(s) for the different failure modes, at the I&C unit level. Categories for the detection of failures are created based on the information on the location of, and the detection of these faults.
3.) The end effect of the fault is described based on fault tolerance coverage, location of the detection, and the functional impact on the I&C unit level.
4.) Group all of the basic failure modes for each I&C module that have the same generic attributes, detection method, and end effects.

The full explanation, along with the example system is given in the literature [9]. The example was discussed from the "Module" level up to the end effect on the RTS/ESFAS. However, since the overall system consists of separate divisions that contain basically the same I&C units, the end effect of each I&C is the effect that is generally considered. The same basic process will be applied to the "Sub-Component" level, with the end effects being given for the "Basic Component" level, and then up to the full system level (RTS/ESFAS) at the end. The (hardware sub-components) process is seen in Tables 41-43. In the case of this taxonomy, those end effects are assumed to occur if the mitigation methods are not implemented, fail, or are implemented incorrectly.

Table 26: Sub-Component Level FPGA Taxonomy PSA Demonstration (Step 1) - Hardware

Failure Set	Hardware Module	Failure Mode	Failure Effect	Uncovering Situation	Functional Impact on "BC"	Functional Impact on AIM	Mitigation Methods
Aging Process (Clock)	Clock	HCE	Non-Fatal (Plausible)	Online Detection Revealed by Demand (Latent)	Delayed Output	Delayed AIM Outputs FPGA Reset (Detection)	Monitor Clock Skew MTBF for HC Periodic Testing
		NBTI	Non-Fatal (Plausible)				
Aging Process (FPGA Chip)	Programmable Interconnect	Electromigration	Fatal (Ordered)	Revealed by Demand (Latent)	No Output (Fatal) Incorrect Output (Non-Fatal)	Incorrect AIM Output	MTBF for Aging Failures Periodic Testing
			Non-Fatal (Implausible)				
		Stress Migration	Fatal (Ordered)				
			Non-Fatal (Implausible)				
Radiation Induced Hard Errors	CLB Logic	SEDB	Fatal (Ordered)	Revealed by Demand (Latent)	No Output (Fatal) Incorrect Output (Non-Fatal)	Incorrect AIM Output	Sensitivity Testing Protective Circuits Power De-Rating Periodic Testing
			Non-Fatal (Implausible)				
		SEGR	Fatal (Ordered)				
			Non-Fatal (Implausible)				
Radiation Induced Soft Errors	Registers	SEU	Non-Fatal (Plausible)	Online Detection Spurious Actuation	Incorrect Output	Incorrect AIM Output	TMR (EDAC)
			Non-Fatal (Implausible)				
		SET	Non-Fatal (Implausible)				Spatial or Temporal Redundancy Circuit Freezing
Chip and Board	FPGA Chip I/O	Stuck Pin	Non-Fatal (Plausible)	Revealed by Demand (Latent) Spurious Actuation	Incorrect Output	Incorrect AIM Output	Detect/control damaged/ disconnected pins Periodic Testing
Board Level	FPGA Board I/O	Discrete (Digital) Input	Fatal (Ordered)	Online Detection	No output (resulting from no input)	No AIM Output	Standards for fault models, diagnostic coverage and mitigation
Maintenance Induced	FPGA Chip/Board	ESD	Fatal (Haphazard)	Revealed by Demand (Triggered)	No Output	No AIM Output	Electrostatic Protection Program
		EOS					
Bit Error	Programmable Interconnect	DRF	Non-Fatal (Plausible or Implausible)	Revealed by Demand (Latent)	Incorrect Output	Incorrect AIM Output	Copies of Programming Data[2] Data for Configuration memory and P/E cycles MTBF for Interconnects
Environmental Qualification	FPGA Chip/Board	Substrate Breakdown (High Temp.)	Fatal (Haphazard)	Revealed by Demand (Triggered)	No Output	No AIM Output	Environmental Qualification Procedures

Security Breach	FPGA Chip/Board (FPGA Logic)	DPA	Non-Fatal (Plausible)	Online Detection	No Functional Impact (System Security Compromised)	No Functional Impact (System Security Compromised)	Secure Lifecycle Attack Countermeasures Cyber Security Guides
Sneak Circuit	FPGA Chip/Board	Hardware Sneak Circuit	Non-Fatal (Plausible or Implausible)	Revealed by Demand (Latent)	No Output	Incorrect AIM Output	General and FPGA-Specific Sneak Circuit Analysis
				Spurious Actuation	Incorrect Output		
CCF	FPGA Chip/Board	Hardware CCF	Fatal (Haphazard)	Undetectable	Board level Failure	No AIM Output	Diversity and Defence in Depth Requirements from technical standards CCF Analysis

Table 27: Sub-Component Level FPGA Taxonomy PSA Demonstration (Step 2-3) - Hardware

Failure Set	Hardware Module	Uncovering Situation	Functional Impact on "BC"	Compressed Failure Mode	Failure Detection	Failure End Effect (AIM)	Mitigation Methods
Aging Process (Clock)	Clock	Online Detection	Delayed Output	Latent Loss of Function	Online Monitoring	Delayed AIM Outputs	Monitor Clock Skew Periodic Testing MTBF for Clock-related aging
		Revealed by Demand (Latent)			Offline Detection		
Aging Process (FPGA Chip)	Interconnects	Revealed by Demand (Latent)	No Output (Fatal)/Incorrect Output (Non-Fatal)	Latent Loss of Function/Loss of Function	Offline Detection	Incorrect AIM Output	MTBF for Chip Aging Failures Periodic Testing
		Offline Detection			Offline Detection		
Radiation Induced Hard Errors	CLB Logic	Revealed by Demand (Latent)	No Output (Fatal)/Incorrect Output (Non-Fatal)	Latent Loss of Function/Loss of Function	Offline Detection	Incorrect AIM Output	Sensitivity Testing Protective Circuits Power De-Rating Periodic Testing
		Offline Detection			Offline Detection		
Radiation Induced Soft Errors	Registers	Online Detection	Incorrect Output	Loss of Function	Online Monitoring	Incorrect AIM Output	TMR EDAC Spatial or Temporal Redundancy Circuit Freezing
		Spurious Actuation		Spurious Function	Self-Revealing		
Chip and Board	FPGA Chip I/O	Revealed by Demand (Latent)	Incorrect Output	Loss of Function	Offline Detection	Incorrect AIM Output	Detect/control damaged/ disconnected pins Periodic Testing
		Spurious Actuation		Spurious Function	Self-Revealing		
		Offline Detection		Loss of Function	Offline Detection		
Board Level	FPGA Board I/O	Online Detection	No Output/Incorrec	Loss of Function	Online Detection	No Output/Incorrect AIM Output	Standards for fault models, diagnostic

			t Output		Offline Detection		coverage and mitigation
Maintenance Induced	FPGA Chip/Board	Revealed by Demand (Triggered)	No Output	Loss of Function	Undetectable	No AIM Output	Electrostatic Protection Program
Bit Error	Programmable Interconnect	Revealed by Demand (Latent)	Incorrect Output	Latent Loss of Function	Online Detection	Incorrect AIM Output	Copies of Programming Data Data for Configuration memory and P/E cycles MTBF for Interconnects
Environmental Qualification	FPGA Chip/Board	Revealed by Demand (Triggered)	No Output	Loss of Function	Undetectable	No AIM Output	Environmental Qualification Procedures
Security Breach	FPGA Chip/Board (FPGA Logic)	Revealed by Demand (Triggered)	No Functional Impact (System Security Compromised)	No Functional Impact (System Security Compromised)	Online Monitoring	Normal AIM Output (System Security Compromised)	Secure Lifecycle Attack Countermeasures Cyber Security Guides
Sneak Circuit	FPGA Chip/Board	Revealed by Demand (Latent)	No Output	Latent Loss of Function	Offline Detection	Incorrect AIM Output	General and FPGA-Specific Sneak Circuit Analysis
		Spurious Actuation	Incorrect Output	Spurious Function	Self-Revealing		
CCF	FPGA Chip/Board	Undetectable	Board level Failure	Loss of Function	Undetectable	No AIM Output	Diversity and Defence in Depth Requirements from technical standards CCF Analysis

Table 28: Sub-Component Level FPGA Taxonomy PSA Demonstration (Step 4) - Hardware

Failure Set	Hardware Module	Compressed Failure Mode	Failure Detection	Failure End Effect (AIM)	Failure End Effect (RTS/ESFAS)	Mitigation Method
Aging Process (Clock)	Clock	Latent Loss of Function	Monitoring Periodic Test	Delayed AIM Outputs (Latent loss of Function)	Loss of 1oo4 conditions of specific APU/VU outputs	Monitor Clock Skew Periodic Testing MTBF for Clock-related aging
Aging Process (FPGA Chip)	Interconnects	Latent Loss of Function/Loss of Function	Periodic Test	Incorrect AIM Output (Loss of Function)	Loss of 1oo4 conditions of specific APU/VU outputs	MTBF for Chip Aging Failures Periodic Testing
Radiation Induced Hard Errors	CLB Logic	Latent Loss of Function/Loss of Function	Periodic Test	Incorrect AIM Output (Loss of Function)	Loss of 1oo4 conditions of specific APU/VU outputs	Sensitivity Testing Protective Circuits Power De-Rating Periodic Testing

Radiation Induced Soft Errors	Registers	Loss of Function	Monitoring	Incorrect AIM Output (Loss of Function Or Spurious Function)	1oo4 conditions of specific APU/VU outputs according to FTD	TMR EDAC Spatial or Temporal Redundancy Circuit Freezing
		Spurious Function	Self-Revealing			
Chip and Board	FPGA Chip I/O	Loss of Function	Periodic Test	Incorrect AIM Output (Loss of Function or Spurious Function)	1oo4 conditions of specific APU/VU outputs according to FTD	Detect/control damaged/ disconnected pins Periodic Testing
		Spurious Function	Self-Revealing			
Board Level	FPGA Board I/O	Loss of Function	Periodic Test	No Output/Incorrect AIM Output (Loss of Function)	1oo4 conditions of specific APU/VU outputs according to FTD	Standards for fault models, diagnostic coverage and mitigation
Maintenance Induced	FPGA Chip/Board	Loss of Function	Undetectable	No AIM Output (Loss of Function)	1oo4 conditions of specific APU/VU outputs according to FTD	Electrostatic Protection Program
Bit Error	Programmable Interconnect	Latent Loss of Function	Monitoring	Incorrect AIM Output	Loss of 1oo4 conditions of specific APU/VU outputs	Copies of Programming Data Data for Configuration memory and P/E cycles MTBF for Interconnects
Environmental Qualification	FPGA Chip/Board	Loss of Function	Undetectable	No AIM Output (Loss of Function)	1oo4 conditions of specific APU/VU outputs according to FTD	Environmental Qualification Procedures
Security Breach	FPGA Chip/Board (FPGA Logic)	Revealed by Demand (Triggered)	Monitoring	Normal AIM Output (System Security Compromised)	Normal RTS/ESFAS Operation (System Security Compromised)	Secure Lifecycle Attack Countermeasures Cyber Security Guides
Sneak Circuit	FPGA Chip/Board	Latent Loss of Function	Periodic Test	Incorrect AIM Output	Loss of 1oo4 conditions of specific APU/VU outputs	General and FPGA-Specific Sneak Circuit Analysis
		Spurious Function	Self-Revealing			
CCF	FPGA Chip/Board	Loss of All Functions	Undetectable	Loss of multiple Channels (AIM)	Loss of Multiple Channel functions	Diversity and Defence in Depth Requirements from technical standards CCF Analysis

Table 26 covers Step 1 of the process. Examples of potential failures modes for the different (sub-component) hardware modules are presented, along with the Failure Effect and Uncovering Situations

(as defined by the OECD-NEA taxonomy). Finally, the functional impact of the failure modes on the Basic Component level (overall FPGA chip) is listed. Table 27 shows Step 2 and Step 3 of the process, including the "Compressed Failure Mode" (high level failure mode at the I&C unit level, based on similar uncovering situations and failure effects), examples of failure detection methods, and the potential effect(s) on the AIM. The AIM was chosen, as it was used in the "Basic Component" level exampled in the OECD-NEA taxonomy, and was used in Table 25 to create the "Basic Component" level FPGA FMEA example. This allows Tables 40-43 to be tied together, and for the RTS/ESFAS example system used in the OECD-NEA taxonomy to be applied to the FPGA taxonomy. Finally, Table 28 gives the final step, and shows the impact of failures of the hardware components at the "Sub-Component" level through to the total system level.

An example of a Hardware failure is a Single Event Upset (SEU) that occurred in one of the Registers. This would temporarily invert a memory bit, which would lead to either a "Plausible" or "Implausible" Non-Fatal failure. The incorrect data would be passed through the chip, causing an "Incorrect Output" at the "Basic Component" Level (Table 27). It is possible to detect these types of errors using Error Detection Codes/Error Detection and Correction Codes (EDC/EDAC), so if those methods are included, it may reveal that an SEU has occurred. If these methods fail or were not included, then the error may not be seen until there is a "Spurious Actuation", or it is "Revealed by Demand (Latent)". This will lead to an "Incorrect Output" from the AIM (Table 27). Finally, this AIM failure will cause a failure at the "System" level that is dependent on the exact RTS/ESFAS function where the failure occurred.

3.3.6.4. Software FMEA and Sub-Component Demonstration

In the OECD-NEA Taxonomy, separate treatments for the Hardware and Software FMEA/demonstrations were given. In the case of software, the OECD-NEA Taxonomy is based on the FMEA of the following list [9]:

1.) Software CCF for all subsystems
2.) CCF for one subsystems
3.) Software fault causing a failure at the level of redundant systems
4.) Software causing a fault in application functions.

While this is sufficient for the levels of abstraction discussed in the OECD-Taxonomy, it would not be applicable to the "Sub-Component" level that has been developed in this paper. The HDL Code FMEA and Demonstration followed the same process as the Hardware case, outlined in sub-section 3.3.6.3. This is reasonable in an FPGA, as any software faults will manifest themselves as hardware logic errors, once the HDL code is synthesized and the FPGA is configured. The results are shown in Table 29 and Table 30. In order to reduce the space taken up by the data tables, the intermediate step for the HDL Code FMEA is not shown.

Table 29 and Table 30 demonstrated the "Sub-Component" level of abstraction, as it pertains to the RTS/ESFAS test system. This was all based on the structure and template of the OECD-NEA Taxonomy. It was seen that both hardware failures (Tables 41-43) and HDL Code failures (Tables 44-45) could cause a failure or failures in the AIM, and those AIM failure modes would then have an effect on the total system. The "Sub-Component" level failures would cause a failure at the "Basic Component" level (the FPGA or HDL Code), which would in turn cause a failure in the "Module" Level, then the "Unit" Level, and finally the "System Level". This is the "Cascading Failure Propagation" discussed in the OECD-NEA Taxonomy. It should be noted that the end effects the AIM failure modes had on the RTS/ESFAS system were taken from the OECD-NEA Taxonomy [9]. As in the case of the hardware failure modes, those end effects are assumed to occur if the mitigation methods are not implemented, fail, or are implemented incorrectly.

Table 29: Sub-Component Level FPGA Taxonomy PSA Demonstration (Step 1) – Software

Failure Set	Software Module	Failure Mode	Failure Effect	Uncovering Situation	Functional Impact on "BC"	Mitigation
Soft Processor	Soft Processor	Hang/Deadlock	Fatal (Ordered)	Online Detection	No Output (Loss of FPGA Function)	FPGA V&V[1,60] Software V&V[11,65,66]
		Signal Delay	Non-Fatal (Plausible)	Revealed by Demand (Latent)	Delayed FPGA Signal Output	
		Random/Unknown Signal	Non-Fatal (Plausible)	Revealed by Demand (Latent)	Incorrect Output	
			Non-Fatal (Implausible)	Online Detection	Incorrect Output	
				Spurious Actuation	Incorrect Output	
State Machine	State Machine	Hang/Deadlock	Fatal (Ordered)	Online Detection	No Output or	WDT State Machine Hazard Analysis Return to pre-
		Endless Loop	Fatal (Ordered)	Online Detection		

		No Exit	Fatal (Ordered)	Online Detection	Stuck Output (Fatal, loss of FPGA function) Incorrect Output (Non-Fatal)	defined state
				Revealed by Demand (Latent)		
		Unreachable	Non-Fatal (Plausible)	Revealed by Demand (Latent)		
				Offline Detection		
Logic Errors (HDL)	HDL EFs	Math Errors	Non-Fatal (Implausible)	Online Detection	Incorrect Output	HDL Code V&V International Standard
			Non-Fatal (Plausible)	Revealed by Demand (Latent)		
				Spurious Actuation		
		Logic Errors	Non-Fatal (Implausible)	Online Detection		
			Non-Fatal (Plausible)	Revealed by Demand (Latent)		
				Spurious Actuation		
Clock/Timing	Registers	Latch	Non-Fatal (Plausible)	Revealed by Demand (Latent)	Incorrect Output (Timing Error)	Implement Registers Complete logic statements and sensitivity lists (accidental latch) Static Timing Analysis Timing Simulations[1,60]
				Spurious Actuation		
		Set-up/Hold Violation	Non-Fatal (Implausible)	Revealed by Demand (Latent)		
				Spurious Actuation		
Input and Data Type	Data Type	FXP Resolution	Non-Fatal (Plausible)	Online Detection	Incorrect Output	Ensure proper calculation/ verification of all FXP values
			Non-Fatal (Implausible)	Revealed by Demand (Latent)		
		Input Overflow	Non-Fatal (Plausible)	Online Detection		
			Non-Fatal (Implausible)	Revealed by Demand (Latent)		
Maintainability	IP Core	IP Core Failure	Fatal or Non-Fatal (Function Dependent)	Function Dependent	Function Dependent	FPGA V&V Software V&V
Design Security	FPGA Logic/Timing (FPGA Synthesis Tool)	Design Tool Subversion	Fatal or Non-Fatal	Triggered by Demand	Unauthorized access, incorrect outputs or device damage	Secure Lifecycle Trusted Tools and IP Cores
COTS	FPGA Logic	COTS HDL Code Failure	Fatal or Non-Fatal (Function Dependent)	Function Dependent	Function Dependent	FPGA Dedication Software Dedication
Maintenance Induced	IP Core	Composition Problem	Fatal or Non-Fatal (Function Dependent)	Function Dependent	Function Dependent	Secure Architecture and comms Trusted Tools and IP Cores

Security Breach	FPGA Logic/Circuitry	FPGA Virus	Fatal (Haphazard)	Triggered by Demand	No Output (Chip Damaged/ Destroyed)	Secure Lifecycle Bit stream Protection Secure Architecture and comms. Trusted Tools and IP Cores
Sneak Circuit	FPGA Chip Logic	Software Sneak Circuit	Non-Fatal (Plausible or Implausible)	Periodic Test Spurious Actuation	Incorrect Output	General and FPGA-Specific Sneak Circuit Analysis
CCF	FPGA Logic (Multiple Chips)	Software CCF	Fatal	Undetectable	Board level Failure	Diversity and Defence in Depth Requirements from technical standards CCF Analysis/ specification

Table 30: Sub-Component Level FPGA Taxonomy PSA Demonstration (Steps 2-4) - Software

Failure Set	Software Module	Compressed Failure Mode	Failure Detection	Failure End Effect (AIM)	Failure End Effect (RTS/ESFAS)	Mitigation Method
Soft Processor	Soft Processor	Loss of Function	Monitoring (WDT)	Delayed Output or No Output (Latent loss of Function)	Loss of 1oo4 conditions of specific APU/VU outputs	FPGA V&V[1,60] Software V&V[11]
		Latent Loss of Function	Periodic Test			
		Spurious Function	Self-Revealing			
State Machine	State Machine	Loss of Function	Monitoring (WDT)	No Output or Incorrect Output (Loss of Function)	Loss of 1oo4 conditions of specific APU/VU outputs	WDT State Machine Hazard Analysis Return to pre-defined state
		Latent Loss of Function	Periodic Test			
		Spurious Function	Self-Revealing			
Logic Errors (HDL)	HDL EFs	Latent Loss of Function	Periodic Test	Incorrect Output (Loss of Function)	Loss of 1oo4 conditions of specific APU/VU outputs	HDL Code V&V International Standard
		Spurious Function	Self-Revealing			
Clock/Timing	Registers	Latent Loss of Function	Periodic Test	Delayed or Missed Output (Timing Error)	Loss of 1oo4 conditions of specific APU/VU outputs	Implement Registers Complete logic statements/ sensitivity list Static Timing Analysis Timing Simulations
		Spurious Function	Self-Revealing			
Input and Data Type	Data Type	Loss of Function	Monitoring (Sanity Check)	Incorrect Output (Loss of Function)	Loss of 1oo4 conditions of specific APU/VU outputs	Ensure proper calculation/ verification of all FXP values
		Latent Loss of Function	Periodic Test			
Maintainability	IP Core	Function Dependent	Function Dependent	Function Dependent	1oo4 conditions of specific APU/VU outputs according to FTD	FPGA V&V Software V&V

Design Security	FPGA Logic/Timing (FPGA Synthesis Tool)	Latent Loss of Function	Undetectable	Incorrect Output or No Output (Once triggered)	Loss of 1oo4 conditions of specific APU/VU outputs	Secure Lifecycle Trusted Tools and IP Cores
COTS	FPGA Logic	Function Dependent	Function Dependent	Function Dependent	1oo4 conditions of specific APU/VU outputs according to FTD	FPGA Dedication Software Dedication
Maintenance Induced	IP Core	Function Dependent	Function Dependent	Function Dependent	1oo4 conditions of specific APU/VU outputs according to FTD	Secure Architecture and comms Trusted Tools and IP Cores
Security Breach	FPGA Logic/Circuitry	Loss of Function Latent Loss of Function	Undetectable	Damage/ Destruction of AIM	Loss of 1oo4 conditions of specific APU/VU outputs	Secure Lifecycle Bit stream Protection Secure Architecture and comms. Trusted Tools and IP Cores
Sneak Circuit	Software Sneak Circuit	Latent Loss of Function Spurious Function	Periodic Test Self-Revealing	Incorrect Output	Loss of 1oo4 conditions of specific APU/VU outputs	General and FPGA-Specific Sneak Circuit Analysis
CCF	Software CCF	Latent Loss of Function	Undetectable	Complete AIM failure	Loss of Multiple Division Functions	Diversity and Defence in Depth Requirements from technical standards CCF Analysis/ specification

An example of an HDL Code failure is an error in a state machine, causing a “No Exit” failure. This would disallow the state machine to transition out of that state, and could cause the state machine to become “stuck”. This could be a fatal error, causing “No Output” or “Loss of FPGA Function” at the “Basic Component” level (Table 29). This error might be detected by diagnostics measures like Watchdog Timers (WDT), or through offline measures such as Periodic Testing. This would likely cause a “Loss of Function” in the AIM, which would then cause a failure of the RTS/ESFAS system (Table 30).

3.3.7. FPGA Taxonomy PSA Demonstration

The new FPGA taxonomy based on the original OECD-NEA taxonomy will now be demonstrated on an FPGA-based test system. The OECD-NEA taxonomy provided some examples of fault trees, one of which

was recreated and shown in Figure 62. It considers a “Spurious Actuation” of one division (Division “X”) in the Emergency Feed Water system (EFV), due to a failure in the Voting Unit (VU) in that division. At that level of abstraction, there is little difference between the fault tree for the FPGA and software-based system taxonomies, so the fault tree must be expanded to include the lower levels of abstraction. Of specific interest is the basic event entitled “HW Module Failure #6”, highlighted in Grey. According to the OECD-NEA taxonomy, one of the potential causes of “HW Module Failure #6” is a failure in the AIM. That basic event is then expanded on (with a specific focus on the AIM), in sub-trees that are shown in Figure 63 and Figure 64.

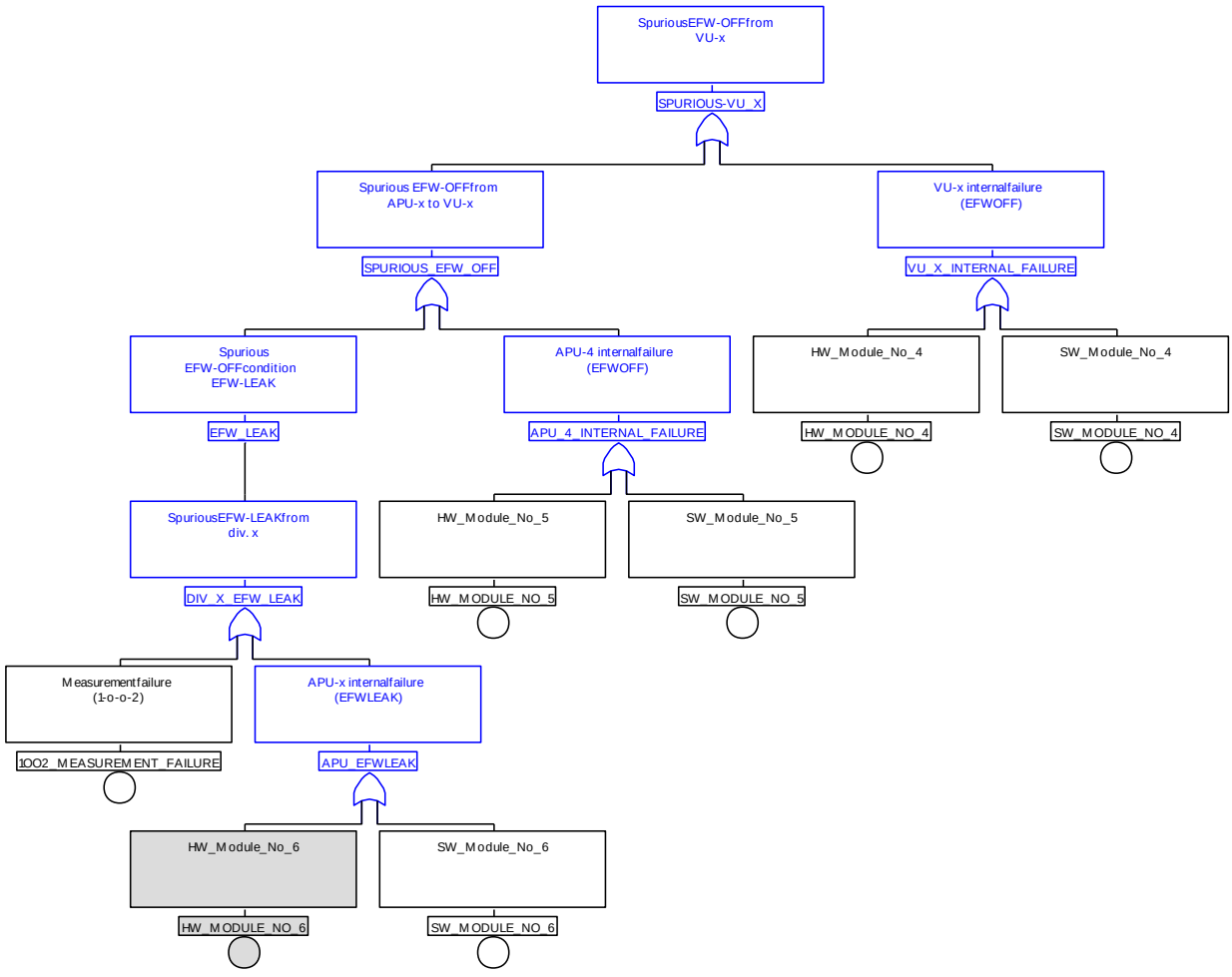


Figure 62: OECD-NEA Taxonomy Fault Tree for a spurious division-X “EFW-OFF” Event

Figure 63 sets the “HW Module Failure #6” as the Top Event, and then proceeds down the levels of abstraction, through the AIM (Unit Level), FPGA Chip and HDL Code (Component Level), ending at the

individual hardware and software modules (Sub-Component Level). This allows for the individual failures for the hardware and software sub-components, as well as their effects on the total system (in this case the Spurious Trip of the EFW), to be modelled. The fault tree shown in Figure 64 is similar to the fault tree in Figure 63, except that the basic events are set as the Failure Categories from Figure 50. This allows for the effects of the different failure categories to be modelled, as opposed to the failures of the sub-components. The mitigation methods for those failure modes are also listed in the fault trees in Figure 63 and Figure 64, to show the effect of the failure of mitigation methods, or if the mitigation methods are not employed.

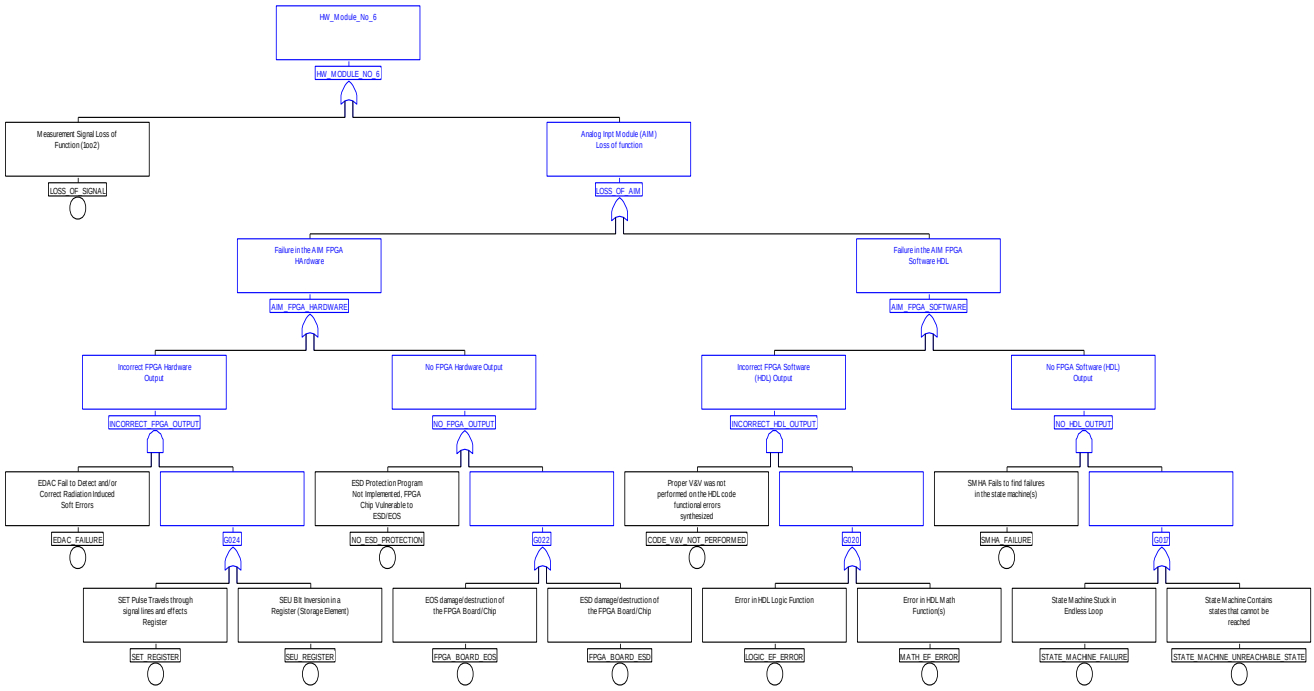


Figure 63: Fault Tree For "HW Module #6" (Sub-Component Level)

The fault trees in Figure 63 and Figure 64 are similar. The main difference is that some of the basic events from the fault tree in Figure 63 were from the same Failure Set, which allowed them to be combined into one event in the Fault Tree in Figure 64. The mitigation methods (or lack thereof) are retained in Figure 64. Including the overall failure sets and general mitigation methods in those fault trees gives a wide overview of the potential failure modes that could be included in the reliability analysis.

The fault trees shown in Figure 62, Figure 63, and Figure 64 could be too simplistic for an actual RTS/ESFAS. However, they serve the purpose for showing how the FPGA taxonomy down to the sub-component level will affect the overall system. Additionally, for a PSA to be performed, quantitative reliability data would need to be included in the fault tree (or other methodology). However, that is beyond the scope of this paper. Furthermore, the FPGA Taxonomy is not limited being used only with fault trees, and could be applied to both static and dynamic reliability assessment methodologies.

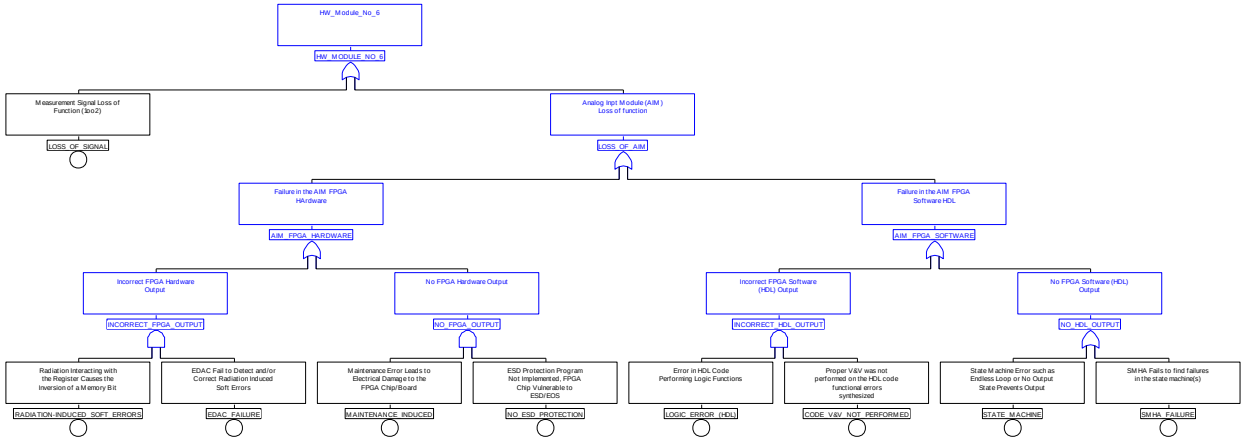


Figure 64: Fault Tree For "HW Module #6" (Sub-Component Level) Using Failure Categories

3.3.8. Conclusions from the FPGA Taxonomy

This FPGA taxonomy discussed several methods that were used to categorize/model failure modes for digital systems. The FPGA taxonomy is intended for use in the design and analysis of FPGA-based I&C systems for NPPs. It would serve as a benchmark for evaluating the failure mode analysis/hazard analysis of FPGA-based systems. This new taxonomy provides information regarding the stage in the lifecycle the failure occurs, if the failure effects are fatal or non-fatal, the uncovering/detection methods of the failure, as well as how failures at the FPGA/HDL level will affect the overall I&C system. This allows for the identification of the failure modes to be avoided in the "Design/Development" stage, and for residual failure modes to be mitigated in the "Operation" stage of the system lifecycle. The use of

the FPGA taxonomy in the hazard analysis of FPGA-based systems presents a basis for the safety and engineering decisions during the design and review of those systems.

The FPGA FMEA and corresponding FPGA Taxonomy provided a great deal of information on the failure modes for FPGAs, and the effects of those failure modes. This failure mode data then became part of the more advanced DFM modelling of FPGA-based systems, as well as the comparisons between DFM and FTA, as seen in sub-sections 4.3 and 4.4.

3.4. Chapter Summary

This chapter presented the results of the FPGA FMEA, as well as the corresponding FPGA Failure Modes Taxonomy. A detailed FMEA was performed to identify the potential failure modes, faults, etc. that could affect FPGAs and FPGA-based systems. This FMEA data was categorized first by the "Lifecycle", then "Cause", and finally into "Failure Sets", based on the effects of the failures, and the mitigation methods for those failures. Afterwards, the FPGA FMEA data was restructured based on the OECD-NEA digital failure mode taxonomy. The OECD-NEA taxonomy provided information on categorization failure modes in digital system due to their effects, fault location and uncovering situations. However, that document left FPGAs, as well as mitigation methods, as a topic of future work. The FPGA taxonomy created a plug-in to interface with the OECD-NEA taxonomy, completing an important topic of future work that was identified in the OECD-NEA taxonomy, as well as making the FPGA taxonomy useful for international working groups. In the context of this thesis, the FMEA/FPGA Taxonomy results were used a form of "fault injection" during the DFM/FTA comparisons shown in subsections 4.3 and 4.4.

4. Application of DFM to FPGA-Based System Analysis

Chapter four details the results of the DFM modelling and analysis research program for FPGA-Based systems. Previously, DFM had been applied to model and analyze digital I&C systems in NPPs, however these were always software-based systems. The first part of this research program consisted of preliminary research into the use of DFM for modelling and analyzing FPGA systems. Therefore, this preliminary research (sub-sections 4.1 and 4.2) focused on demonstrating the effectiveness and suitability of DFM for the modelling and analysis of FPGA-based systems. The second part focused on the comparison of DFM and FTA (sub-sections 4.3 and 4.4), for the purpose of modelling FPGA-based systems, which included the failure mode data ascertained in Chapter 3. As FTA is a commonly-used methodology in the nuclear field, comparisons of DFM and FTA will help showcase potential advantages and disadvantages of applying DFM to analyze FPGA-based systems. Sub-section 4.1 will discuss the first DFM modelling work, encompassing qualitative and quantitative, inductive and deductive analyses, using an FPGA-based Post-Accident Monitoring System (PAMS) as a test system. Sub-section 4.2 presents the results for the comparison of DFM with the FPGA/HDL simulation program "Modelsim", for the purpose of simulating FPGA logic, and to prove the effectiveness of DFM for analyzing FPGA-based systems. Sub-section 4.3 discusses the preliminary DFM/FTA comparisons and results. Sub-section 4.4 describes the advanced DFM/FTA comparisons, delving deeper into the theoretical reasons for differences, and exploring DFM analyses with multiple time steps. Sub-section 4.5 presents a summary of the information discussed in this Chapter.

4.1. FPGA PAMS

The first DFM modelling considered a simplified PAMS system. The actual DFM models were static at this point (they did not contain any control loops or feedback), and as such did not utilize any time-dependency. The use of MVL values was also limited in this preliminary work. The results from the system design, testing and modelling were published in the "*Proceedings of the 22nd International Conference on Nuclear Engineering*" [191] Afterwards, some small modifications to the conference paper were made, and the sections focusing on DFM modelling were published as a "Technical Note/Technical Brief", in the "*Journal of Nuclear Engineering and Radiation Science*" [192].

4.1.1. System Description

The PAMS is an electronic system that monitors systems and variables over the anticipated ranges for accident conditions in order to improve safety in the AP 1000. The system is intended to meet the requirements of US NRC (Nuclear Regulatory Commission) Regulatory Guide (RG) 1.97 [193]. Also referenced are documents from IEEE [194], and from the Canadian Standards Association (CSA) (endorsed by the CNSC) regarding accident monitoring [195]. In the AP-1000, the PAMS is a safety system that has been included as part of a larger, FPGA-based system known as the Advanced Logic System (ALS) [79]. It includes both Analog and Digital I/O, with displays and alarms to inform operators of changes in the variables being monitored, and if those variables exceed setpoints. According to the documentation [193–195], the ALS PAMS is required to, and capable of, monitoring multiple variables, such as neutron power, temperature, flow rates, fluid pressure, fluid level, radiation levels and valve positions.

4.1.2. System Design

A National Instruments (NI) cRIO-9076 (Compact Reconfigurable Input/Output) chassis was used for the lab-scale demonstration. It contains a Xilinx Spartan-6 LX45 FPGA chip, and slots for four I/O modules. The LabVIEW FPGA module was then used to program the FPGA [196]. This set-up includes the chassis, and 4 of the c-series modules: 9207, 9219, 9375 and 9365, as seen from left to right in Figure 65 [191]. The remaining module (9363) performs a similar function to the 9365, and can be swapped in/out when needed. The functionality of each I/O module is given in Table 31.

The functionality of the system follows the following basic pattern; using I/O modules the input signal from the sensors is sent to the FPGA to be processed, which returns a value of the variable being monitored. This value is then compared to user defined setpoints, and if the values are too high, then an alarm is triggered. A picture of the test system is shown in Figure 65.

Figure 65: Lab-Scale PAMS Set-Up with NI Equipment

In order to test the system using real input, a number of sensors were procured. Table 32 displays the sensors used for each measurement. All of the measurements were taken using real industry sensors, with the exception of neutron flux, radiation level and valve position, which were simulated. As the focus of this research program was the DFM analysis of FPGA-based systems, a more detailed discussion of the system design and testing is left to the literature [191].

Table 31: FPGA PAMS C-Series Module Description

Module	Description
9207	AI Voltage and Current
9219	Universal AI
9375	Digital I/O
9363	Analog Voltage Output
9365	Analog Current Output

Table 32: FPGA PAMS Sensor Description

Variable	Sensor
Temp	NI Type-K Thermocouple
Temp	NI 3-Wire RTD
Level	Sharp Optical Sensor
Level	Kobold Differential Pressure Sensor
Flow	Kobold Paddlewheel Flow Sensor
Pressure	Kobold Pressure Transducer
Valve	(Simulation)
Neutron Flux	(Simulation)
Radiation	(Simulation)

In terms of the simulated values, the Labview software was set-up to output values used in the simulations. In the case of the "Valve" position, it was simulated to be in either an "Open" or "Closed" state. With regards to the "Neutron Flux" and "Radiation" values, Labview was set-up to output these respective values based on random values inside of reasonable ranges, based on the author's technical opinion and literature information.

4.1.3. FPGA PAMS DFM Models

The DFM models were constructed to model the reliability of the system. The DFM model for a general subsystem in the PAMS is showed in Figure 66. The model was analyzed both deductively and inductively, to obtain a complete picture of the system. At this point, the DFM analysis is intended to show the usefulness and effectiveness of using DFM to model an FPGA-based safety system. In order to properly quantify the failure probabilities and obtain exact reliability measures, other factors such as software faults and CCF modes would need to be included.

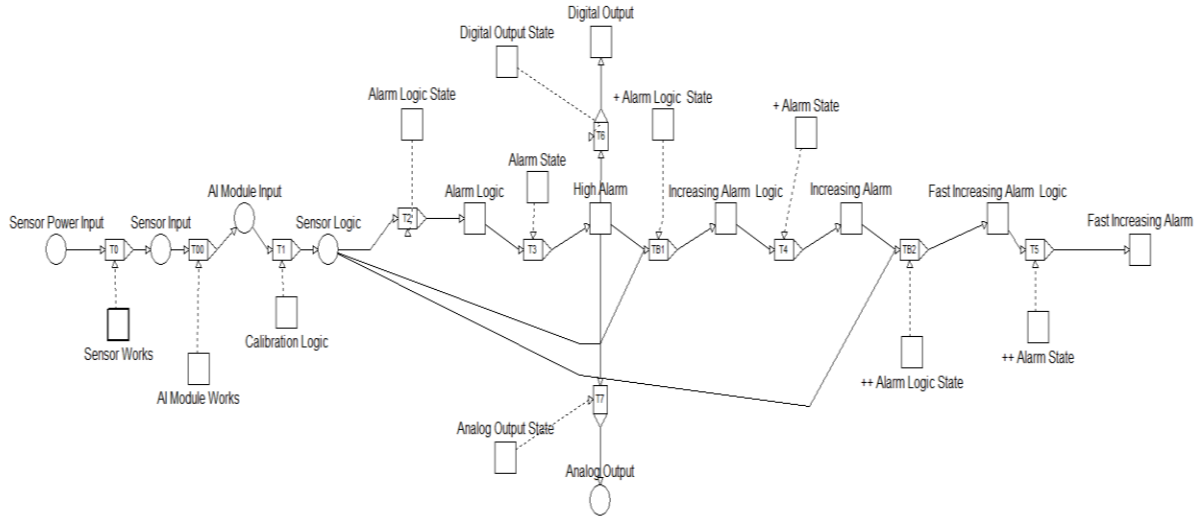


Figure 66: General PAMS Subsystem DFM Model

4.1.3.1. Deductive Analysis

False Alarm:

A false alarm (the alarm tripping when it should not) is less of a safety risk than a missed trip, but it is still a case of a system malfunction and is to be avoided. Knowing that the false alarm can happen, the "Alarm Logic" node is used in this deductive analysis, with a "False Alarm" state set as the top event. A sample of the results of this analysis is shown in Table 33. Fourteen total implicants were produced. There are both hardware and logic failures that can cause the error. Implicant 2 involves only hardware issues (sensor); however Implicant 1 involves a failure with the logic, and the incorrect trip logic leads to a false alarm. Knowing this, the operator can examine the sensors and input modules, as well as the logic to correct any faults. Although there is no software on the FPGA chip itself, the FPGA is programmed using Hardware Description Language (HDL), so logic errors would carry over to the FPGA. In this example, the logic improperly determines the trip condition. In this example, the logic improperly determines the trip condition. This analysis would allow the developers to focus attention on those sections of the FPGA logic, in order to avoid any logic errors.

Table 33: Implicants for "False Alarm" Top Event (FPGA PAMS)

Implicant 1	
AI Module State	AI Module Works
Alarm Logic State	Logic Fails (High)
Calibration Logic	Logic Works
Sensor Power Input	Correct Input
Sensor State	Sensor Works
Implicant 2	
AI Module State	AI Module Works
Alarm Logic State	Logic Works
Calibration Logic	Logic Works
Sensor Power Input	High Input
Sensor State	Sensor Works

4.1.3.2. Inductive Analysis

Calibration Logic High:

It was seen when performing the deductive analyses that errors in the logic, such as the calibration logic, can have adverse effects on certain functions of the system, such as a false alarm. Often, an inductive analysis is performed after the deductive analyses, using the discovered fault(s) as starting event(s). Performing this analysis can help confirm the effect of the starting event (the inductive analysis should return to the original top event), as well as detect other possible issues that that fault could cause. A sample of the results is shown in Table 34. In the inductive analysis, the implicants are referred to as "sequences", and the analysis returned 84 total sequences, with Sequence 12 being shown in the table, and the initiating event is shown in italics. The underlined event shows that the "High Alarm" is in the "ON" state, as predicted in the deductive analysis, which confirms that this particular logic error could cause a False Alarm. Furthermore, it uncovers other issues, such as the "Increasing Alarm" being tripped and output signals being sent, when those events should not occur. Performing these analyses as well can show if a certain fault has been fixed entirely. Here, the "+" refers to the "Increasing Alarm" and the "++" represents the "Fast Increasing Alarm".

Table 34: Sequences for "Calibration Logic Fails (High)" Initiating Event (FPGA PAMS)

Sequence 12	
Calibration Logic	*Logic Fails (High)*
AI Module Input	Correct Input
Alarm Logic	False Alarm
Alarm Logic State	Logic Works
Alarm State	Alarm Works
+Alarm State	Alarm Works
++Alarm State	Alarm Works
Analog Output	Output
+ Alarm Logic State	Logic Works
++ Alarm Logic State	Logic Works
Digital Output	Output
Fast Increasing Alarm	ON
Increasing Alarm	ON
<u>High Alarm</u>	<u>ON</u>
Analog Output State	Works
Digital Output State	Works
Sensor Logic	High

4.1.3.3. Probability Calculations

Probabilities were added to the model using Mean Time Between Failure (MTBF) data from NI documentation, based on the Bellcore model, as well as datasheets from similar sensors [197]. The failures were converted from failures/10^6 hours to failures/year for easier calculations. Logic failure calculations were not completely performed at this stage, so values were assumed (taken as 10% of the failure rate of an input module) [17,198]. The deductive analysis was run for both "False Alarm" and "Missed Trip" top events.

Table 35 shows a section of the analyses, specifically the most likely PI for each Top Event. It was seen that a sensor failure is most likely to cause a missed trip, and a logic failure is most likely to cause a False Alarm. Also seen, is that the top Missed Trip implicants have a higher likelihood of occurring, which is expected as it is more likely that the hardware fails low or fails completely, than failing high. The probability analyses will illustrate what are the most likely failures and their causes.

Table 35: DFM Probability Calculations (FPGA PAMS)

Missed Trip	**Implicant 1** (1.9260E-03) Sensor Fails (Low)
False Alarm	**Implicant 1** (1.1022E-03) Calibration Logic Fails (High)

4.1.3.4. Logic Analysis

DFM was used to examine the logic sections of the system to uncover logic errors. Figure 67 shows a basic model for the logic flow involving one of the pressure sensors. The sensor output is scaled to a linear range, making it simple to determine the calibration equation, which was synthesized on the FPGA. Each logic operation performed by the FPGA for this section is checked using DFM. A deductive analysis is again run, with "Missed Trip" as the Top Event. In total, nine implicants were found, with Implicant 1 shown in Table 36. It is seen that there is an issue with the trip logic, causing it not to generate a trip result, leading to the alarm not being tripped. This analysis could uncover issues, such as

improper comparison logic in the trip section. This analysis would ensure that any and all logic issues could be identified and repaired.

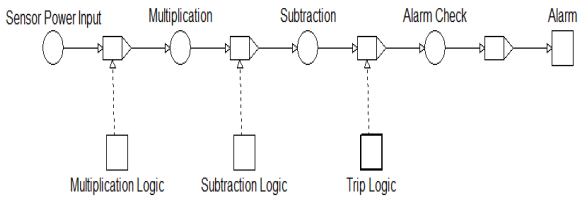


Figure 67: General Logic DFM Model (FPGA PAMS)

Table 36: Implicants for Code Section "False Alarm" Top Event (FPGA PAMS)

Implicant 1	
Multiplication Code	Code Works
Sensor Power Input	Correct Input
Subtraction Code	Code Works
Trip Code	No Value
Implicant 2	
Multiplication Code	Code Fails (Low)
Sensor Power Input	Correct Input
Subtraction Code	Code Works
Trip Code	Low Value

4.1.4. Conclusions from FPGA PAMS DFM Modelling

A lab-scale demonstration of an FPGA-based Post Accident Monitoring System was designed and analyzed. DFM has been shown to work effectively when analyzing FPGA-based systems, by uncovering possible faults in the system, and the potential effects of those faults. Moreover, it was able to calculate the probabilities of certain system failures, and illustrated that improvements in the hardware and logic could be made. DFM can then be incorporated into analyzing more complex systems, with more components and more advanced control logic. However, this work was somewhat limited, by a small test system, lack of dynamic behaviour, and limited application of MVL states. The next phase of the DFM/FPGA research program included dynamic behaviour as well as MVL states, and was used to verify the effectiveness of DFM modelling for FPGA-based systems.

4.2. Comparisons Between DFM and FPGA/HDL Simulations

In order to prove that DFM is accurately modelling the FPGA logic and system properties, the DFM results must be compared to results from an established source. To accomplish this, VHDL code was used to create test systems, as well as testbenches. The testbenches would provide input stimuli in order to test the VHDL test programs. The test programs used synthesizable VHDL code where possible, which were also used to create the VHDL netlists. The VHDL code was written and synthesized (where possible) using the Quartus II Web Edition software package [44]. The Modelsim simulator program was used to simulate the VHDL code, using the test benches [199]. The results of the Modelsim simulations ("waveforms") were then compared to the results of the DFM analyses, to determine the accuracy of the DFM models. Modelsim is a widely used industry software package, making it a suitable choice for verifying the ability of DFM to model FPGA-based systems. This work was published in the journal "*Nuclear Engineering and Technology*" [200].

4.2.1. FPGA Aspects

The three principle aspects of FPGAs were considered for this comparison and verification portion of the research work, and are presented in this sub-section. These include IEEE Standard 1164 (applies to VHDL), a Register/Flip-Flop for storing information, the CLBs that perform that logical computations in the FPGA, and finally a case study using an FPGA-based signal compensator test system. These important FPGA aspects were modelled using DFM, and simulated using Modelsim, to compare how well the DFM results would predict the behaviour of the FPGA logic.. Additionally, a simplified FMEA that was used to obtain the failure data (Top Events) for the DFM analysis of the aforementioned FPGA aspects is provided in this subsection.

4.2.1.1. IEEE Standard 1164

IEEE 1164 is an important standard when using VHDL code to program the FPGA. It is a package that is compiled into a library, and is often imported into VHDL files. This standard defines important features,

including nine-state logic, resolution function, and Boolean functions, such as AND, OR, XOR, and NOT. The Boolean functions AND, OR, XOR, and NOT are defined in tables, and the remaining functions (NOR, XNOR, NAND), are made using NOT function [201]. The states for the nine-state logic can be found in the literature [201]. DFM is used to model the logic and mathematical functions based on this standard. A DFM model consisting of the logic and mathematical functions is shown in Figure 68.

In Figure 68 the logic functions includes AND, OR, XOR, NOT, NOR, NAND and XNOR functions. The inputs are set to be the logic states given in [201] producing various combinations of outputs. A similar method is taken with the math functions, where two inputs are added (In_Add_1 and In_Add_2), and the sum is compared to the product of the other inputs (In_Prod_1 and In_Prod_2). The output "G_Out" is then the output from the "greater or equal to function" (shown as a "less than" function in the netlist). The Modelsim and DFM results are given in Sub-section 4.2.2.

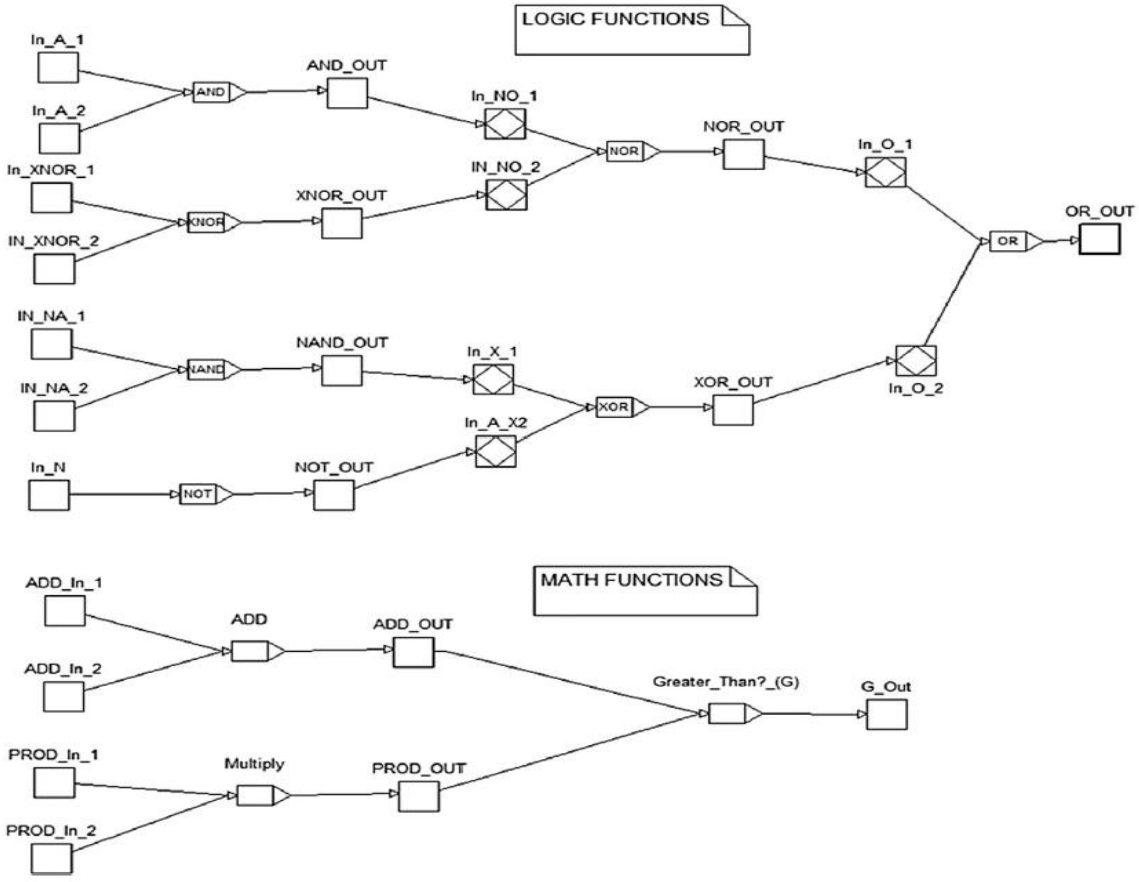


Figure 68: DFM Model for Logic and Mathematical Functions

4.2.1.2. Register

The register is an important component of digital logic and FPGA operation. It is commonly used in electronic systems to store a data value and output it at a certain clock edge. In FPGAs these registers are used in many kinds of sequential and recursive logic, when the algorithms require results or data from previous time steps [202]. The DFM model for the register is shown in Figure 69. Here, the “Reset” , “Preset”, “Input”, “Data” nodes all represent input signals into the Register, where the transition box (entitiled “Register”) performs as a register in an FPGA would. A time delay on “1” is included, in order to hold the data for a time step, similar to an FPGA register. The “Reset” signal will reset the register to “0”, while the “Clock Enable” signal will allow the register to store the new “Input” value, and then output the new value after a time delay (based on the “Clock” signal). If the “Clock” signal does not allow the register to update the value, then the previous input value (“Prev_Input”) is used instead.This model also includes the “Clock_Period” (if it is longer/shorter than specified), “Clock_Cycle” (if the clock has a duty cycle that is different than 50%), as well as a node for the previous input (“Prev_Input”), as that can have an effect on the output.

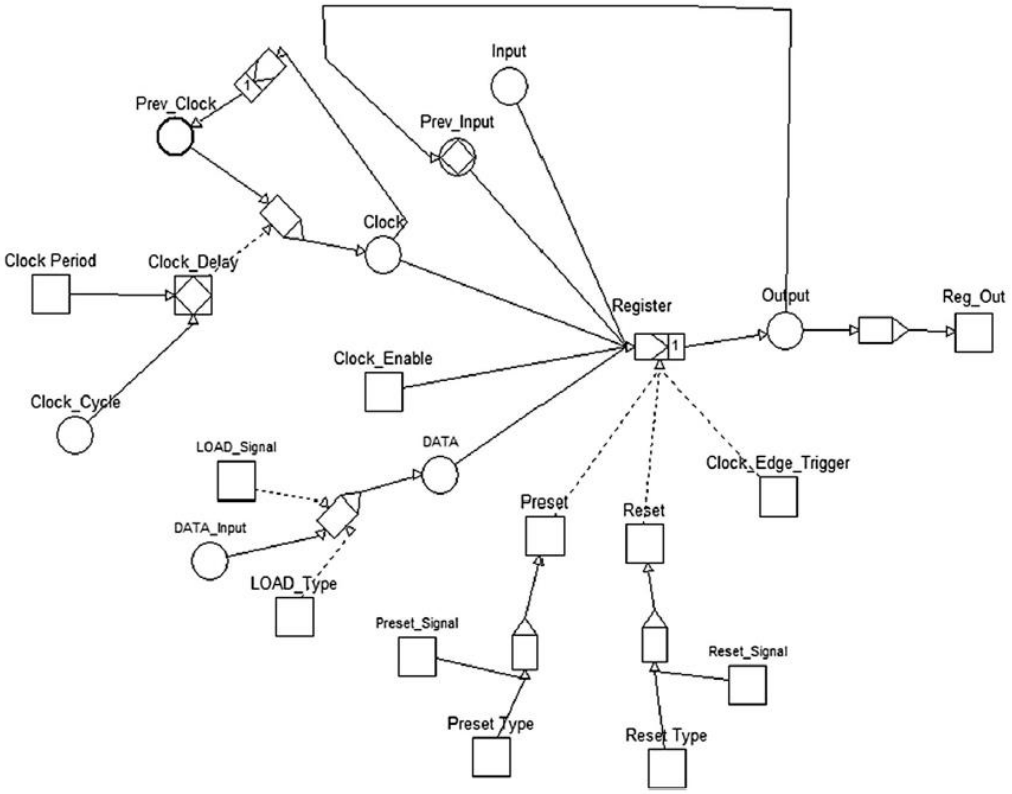


Figure 69: DFM Model of an FPGA Register

The value from the “Output” is fed back into the “Prev_Input”, after a delay. The “Clock” node was broken down into 4 states; 1, 0, +, -, to represent the clock transitions. To allow the “Clock” state to cycle, the “Prev_Clock” node was included, which will store the previous “Clock” signal (after a time delay), and then output the next state to the “Clock”. State “+” represents the rising edge, while state “-“ represents the falling edge. The clock will cycle through the clock states as the time step changes. The “Output” node includes states for “1” and “0”, as well as extra states for the outputs for Reset, Preset and Data. The results are then funnelled to the standard “1” and “0” for the “Reg_Out” node. Results for the register model and Modelsim simulations are given in sub-section 4.2.2 An example of a decision table, similar to the one used in the “Register” transition box is given in Table 37. To save space, it was assumed that the “Previous Input” had a value of “1”. It should be noted that the “-” represents a “Don’t Care” value.

Table 37: Sample Decision Table for Simplified Register (DFM/ModelSim Comparisons)

Inputs				Outputs
Reset	Enable	Clock	Input	
1	-	-	-	0
0	0	-	-	1
0	1	0	-	1
0	1	1	+	0

4.2.1.3. FPGA Logic Blocks (CLB)

All FPGAs share certain basic components, namely Input/Output (I/O) ports, Programmable Interconnects, and Configurable Logic Blocks (CLBs). I/O ports are used to carry data signals to and from the FPGA, while programmable interconnects are used to connect the CLBs together. CLBs are of particular interest, as those logic blocks contain the logic elements needed to perform the desired logic functions. In the most basic form, each CLB will contain a Look-Up Table (LUT), Register, and possibly a Multiplexer (MUX) that can be used to bypass the Register if desired [1]. DFM was used to create two separate Logic Blocks, one with an “AND” gate LUT, and the other with an “OR” gate LUT. This will show how DFM can model the basic logic elements of an FPGA, encompassing the components discussed in sub-sections 4.2.1.1 and 4.2.1.2. A block diagram is presented in Figure 70.

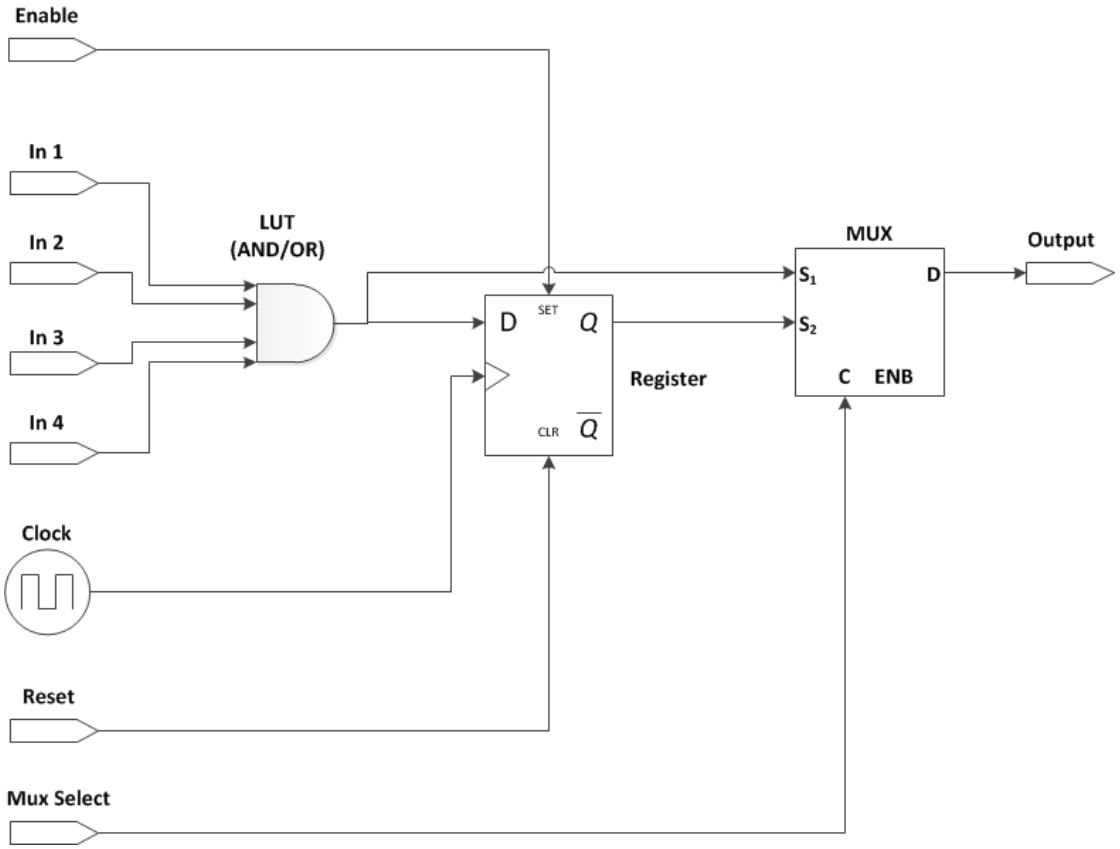


Figure 70: CLB Flowgraph with Either "AND" Gate or "OR" Gate LUT

4.2.1.4. Platinum Dynamic Compensator

Neutron detectors are used in nuclear power plants to monitor neutron power (flux) inside the reactor core. Several different materials are used, such as platinum, rhodium, vanadium and cobalt. In a CANDU (Canada Deuterium Uranium) reactor, platinum detectors are used in safety systems due to their fast (prompt) response. Platinum detectors are composed of multiple isotopes, and therefore have multiple decay chains [203,204]. Dynamic signal compensation is used to compensate for the delayed response of the detector, in order to obtain an accurate reading of the current neutron flux. The details of the platinum detectors and signal compensation can be found in [205,206].

$$\boldsymbol{\Phi_c = 0.89\phi + \frac{0.045\phi}{1+3.9s} + \frac{0.017\phi}{1+30s} + \frac{0.021\phi}{1+250s} + \frac{0.045\phi}{1+2500s}} \quad (45)$$

where Φ_c refers to the calculated neutron flux, and ϕ is the actual neutron flux. Equation 46 is a simplified transfer function for the dynamic signal compensator [207]:

$$\boldsymbol{\Phi_c} = \left[\boldsymbol{K_1} - \frac{\boldsymbol{K_2}}{\boldsymbol{1+T_1 s}} + \frac{\boldsymbol{K_3}}{\boldsymbol{1+T_2 s}}\right]\boldsymbol{I} \quad \textbf{(46)}$$

where I is the current from the detector, K_1, K_2, K_3 are coefficients, T_1, T_2 are corresponding time constants, and s is the Laplace transform variable [203,204]. Appending the values for K_1, K_2, K_3, T_1 and T_2, Equation 46 becomes:

$$\boldsymbol{\Phi_c} = \left[\boldsymbol{1.066} - \frac{\boldsymbol{0.028}}{\boldsymbol{1+30s}} + \frac{\boldsymbol{0.038}}{\boldsymbol{1+2500s}}\right]\boldsymbol{I} \quad (47)$$

Converting Eqn. 47 to a state space representation yields the following equations:

$$\dot{\boldsymbol{\Phi}}(\boldsymbol{t}) = \begin{bmatrix}\boldsymbol{-0.0337} & \boldsymbol{0}\\ \boldsymbol{1} & \boldsymbol{0}\end{bmatrix}\boldsymbol{\Phi}(\boldsymbol{t}) + \begin{bmatrix}\boldsymbol{1}\\ \boldsymbol{0}\end{bmatrix}\boldsymbol{I}(\boldsymbol{t}) \quad (48)$$

$$\boldsymbol{\Phi_c}(\boldsymbol{t}) = [\boldsymbol{-0.0142} \quad \boldsymbol{0}\,]\boldsymbol{\Phi}(\boldsymbol{t}) + [\boldsymbol{15.99}]\,\boldsymbol{I}(\boldsymbol{t}) \quad \textbf{(49)}$$

The state space representation also included the scaling factor, so that an input range of 0-10 V would produce an output range of 0-150% FP. The state space model was then discretized, using the Zero-Order Hold method, and a sampling time of 0.5 seconds, resulting in the following representation:

$$\dot{\boldsymbol{\Phi}}(\boldsymbol{t}) = \begin{bmatrix}\boldsymbol{0.9833} & \boldsymbol{0}\\ \boldsymbol{0.4958} & \boldsymbol{1}\end{bmatrix}\boldsymbol{\Phi}(\boldsymbol{t}) + \begin{bmatrix}\boldsymbol{0.4958}\\ \boldsymbol{0.1243}\end{bmatrix}\boldsymbol{I}(\boldsymbol{t}) \quad (50)$$

$$\boldsymbol{\Phi_c} = [\boldsymbol{-0.0142} \quad \boldsymbol{0}\,]\boldsymbol{\Phi}(\boldsymbol{t}) + [\boldsymbol{15.00}]\,\boldsymbol{I}(\boldsymbol{t}) \quad (51)$$

The block diagram for the platinum compensator is given in Figure 71.

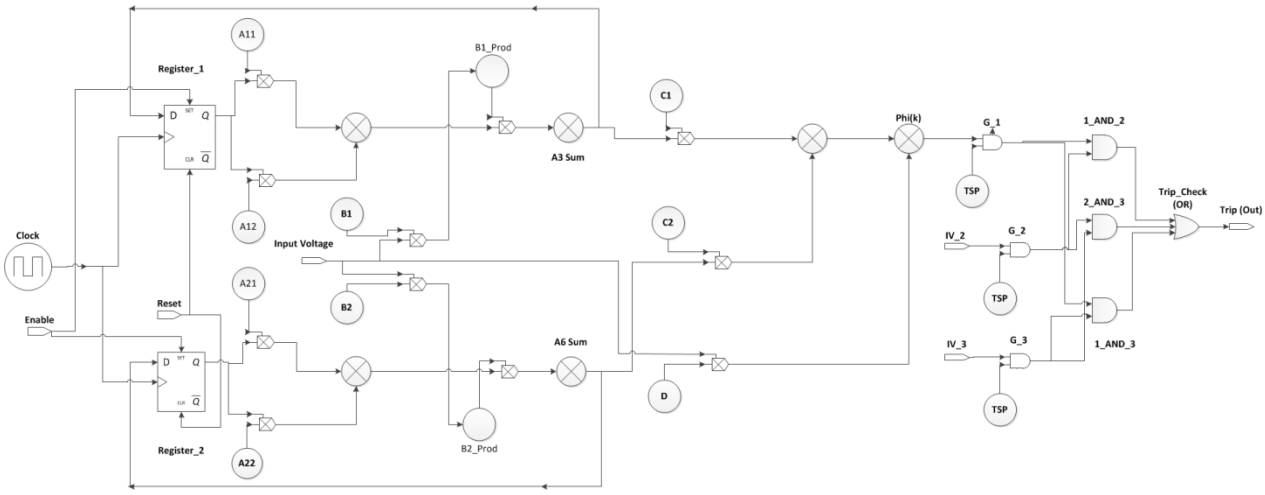


Figure 71: Block Diagram for the FPGA-based Platinum Signal Compensator

In Figure 71, the nodes “G_1”, “G_2”, G_3”, are the logic tests for the input being greater or equal to the TSP and “IV_2” and “IV_3” represent the additional inputs. The “Phi(k)” node refers to the total flux calculation, and the nodes named “A”, “B”, “C” or “D” which also contain numbers (eg. “A11”, “B1”, “C1”, D”) represent constants in the state space equations in Equations 50 and 51. The registers (“Register_1” and “Register_2”) store the outputs from Eqn. 5 that are needed for the calculation at the next time step. The output of the state space equations is a neutron power, based on the detector current, which is based on the neutron flux. As the neutron flux is very high (generally on the order of 10^{14} n/cm^{2}s), it is often expressed as a percentage of Full Power (FP). To simplify the calculations, the voltage input was taken with the range of 0-10 V, and the neutron power was scaled for 0-15%, with a TSP set at 12.0%. In reality, the neutron power would have a range of 0-150% FP, and the TSP being set at around 125% FP.

It can be seen that the previous values for $\Phi(t)$ are required to calculate the correct flux. The are used to store this information, so the FPGA code (and netlist) contains two registers. Additional code was added, for trip logic. The flux value was compared to a Trip Setpoint (TSP), along with two other inputs, to create the 2 out of 3 logic. The analysis results for the Platinum Comparator and trip logic is given in sub-section 4.2.2.

4.2.1.5. Simplified FMEA Example

This sub-section will provide an example of an FMEA for the different FPGA aspects discussed in sub-sections 4.2.1.1-4.2.1.4. These FMEAs produce the Top Events (TE) that are used in some of the DFM analysis results, seen in sub-section 4.2.2. It should be noted that not every DFM analysis considered actual failures, as the aim of the paper was to show the applicability of DFM to modelling of FPGA-based system logic, and as such the models included both correct and erroneous behaviours.

Table 38: Sample FMEA for FPGA Aspects

Aspect	Failure Mode	Cause	Effect
IEEE 1164	Or = 0	"X" Logic "H" Logic	Math Error
	G_Out = 0	"U" Logic	Logic Error
Register	Output "X"	"X" Logic	Memory Error
CLB	Output "1"	Logical Errors	Incorrect CLB Value
DSC	Spurious Trip	Constant Error	System Failure
	Missed Trip	Input Error ("X")	System Failure

Table 38 presents a simplified FMEA for failures that were considered for the different aspects of FPGA logic. The first is the IEEE Logic standard, where it is seen that there could be math or logic errors (respectively), due to the presence of certain logic states. The "U" (Uninitialized), "H" (High), or "X" (Unknown) logic states are generally used to represent errors in simulation. These logic states can produce errors in mathematical/logical functions, resulting to the incorrect (in this case) value of "0", due to how the functions are defined in the IEEE 1164 standard [201]. A similar issue is seen with the register. If an "X" logic value is input into the register, it would be stored, and the "X" would be output on the next clock cycle, which could cause a failure at future time steps.

The effects of the failure modes of the IEEE 1164 standard and register are manifested when the larger FPGA aspects are considered. Logic failures on math/logic functions or registers (such as "U" or "X" logic) can affect the entire CLB. In this thesis, an erroneous Output of "1" (in this example) could be

produced, due to the “U” or “X” logic states. This may not occur with the “OR” logic, depending on the other inputs, but for other logic, such as “AND” logic, the output of the CLB could be affected. This all builds up to the whole test system level, where it is seen that a “Missed Trip” occurs due to an “Input Error”. This could be due to an error with the input signal itself, or it could be due to errors passed along from the other components (i.e. failure modes at the logic, register or CLB level cause failures at the whole system level). Lastly, it was seen that “Spurious Trip” could occur not only through errors in the sub-components, but through an incorrectly specified constant, used in multiplication.

4.2.2. Results of DFM/ModelSim Comparisons

This sub-section shows the results for the DFM models and ModelSim simulations created in sub-section 5.2.1. In each case, the entries in the PI tables that are of the most notable are bolded and underlined. It should be noted that in certain Modelsim waveforms, the signals “Reset”, “Clock Enable”, “Clock”, and “Preset” (if needed) were shortened to “CLR” (Clear), “CE” or “Enable”, “CLK”, and “PRE”, respectively, to save space in the specific waveform graph.

4.2.2.1. IEEE 1164 Standard Results

The results for the DFM analysis and the Modelsim Simulations for the IEEE 1164 standard are presented in this sub-section. Table 39 shows sample implicants for the IEEE 1164 DFM model, with the Modelsim results given in Figure 72. The Top Event for both the logic and math functions were set to be “0”, and the DFM model was run deductively, to find the prime implicants. In total, there are 192 prime implicants for the “OR_OUT” = 0 node and, 104 prime implicants for the “G_OUT” = 0 node. It was seen that the inputs and outputs of the logic model matched up with the inputs and outputs in the Modelsim simulation. The “H” Logic in the “In_NA_1” input produced an “X” value at the output of the “XNOR” logic gate. This in turn caused the overall output of the “OR” logic to read “0”. It was also seen with the math model, that an unknown state (“XX”) in the “In_Add_2” input would cause the “Add_Out” output to also read “XX”, forcing the “G_OUT” node to read “0” (False), indicating that an error with one of the inputs could cause the trip signal not to actuate. This is the same as discussed in the FMEA is sub-section 4.2.1.5.

Table 39: Sample Implicants for "OR_OUT = 0" and "G_OUT = 0" Top Events

Implicant 1 (Node)	Implicant 1 (State)	Implicant 117 (Node)	Implicant 117 (State)
AND_OUT	1	**ADD_In_2**	**XX**
IN_NA_1	**H**	ADD_OUT	XX
IN_NA_2	1		
In_A_1	1		
In_A_2	1		
In_N	1		
NAND_OUT	0		
NOR_OUT	0		
NOT_OUT	0		
XOR_OUT	0		

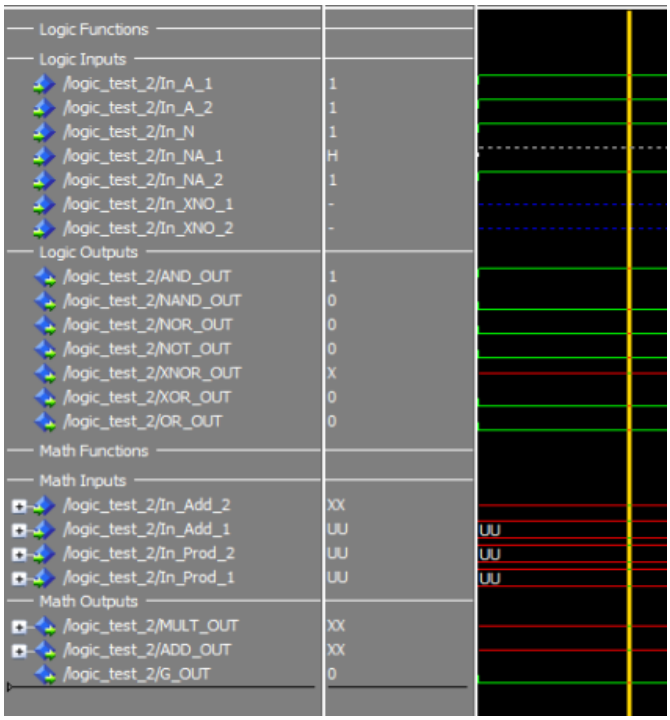


Figure 72: ModelSim Results for "OR_OUT" and "G_OUT" Top Event

4.2.2.2. Register Results

The results for the DFM analysis and Modelsim Simulations for the register are given here. In the Modelsim waveforms, the "Input" node (signal) is denoted as "D", while the "Output" is denoted as "Q". Table 40 and Figure 73 show the results for a register where the Output = "1". The simulation in Figure 73 had the top event "Reg_Out = 1", for one time step. The state "R_E" refers to a rising edge clock trigger, "Normal" clock period means that there is no clock delay, and the "Synch/Asynch" refers to synchronous and asynchronous processes, respectively. The time steps in this DFM model were taken as one time step being equal to one half of the clock cycle. In Fig. 7 (Implicant 26), it is seen that having an Input of "1", with the Clock_Enable signal of "1" and a rising edge clock and trigger will produce the "Output" of "1", when it is not pre-empted by other inputs such as the Reset, Preset or Load signals. The Modelsim results will not explicitly show values for the "Clock_Edge_Trigger", and will not explicitly state if the "Clock_Period" is correct, it will just show what the value is.

Table 40: Prime Implicant for DFM FPGA Register Analysis (Top Event "Output = 1")

Implicant 26 (Node)	Implicant 26 (State)	Implicant 26 (Time Step)
Clock	**+**	-1
Clock_Period	Normal	-1
Clock_Edge_Trigger	**R_E**	-1
Clock_Enable	**1**	-1
DATA_Input	No_DATA	-1
Input	**1**	-1
Preset_Signal	0	-1
Prev_Clock	0	-1
Prev_Input	0	-1
Preset_Signal	0	-1
Reset_Signal	0	-1

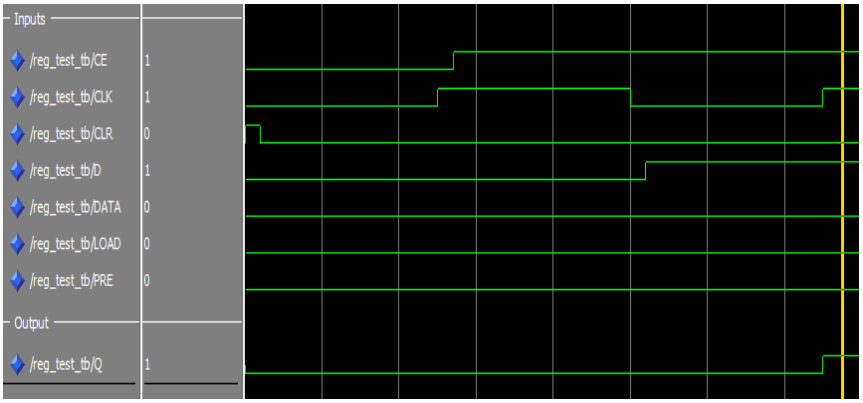


Figure 73: ModelSim Results for FPGA Register Analysis (Top Event "Output =1".

DFM can also be used to identify possible failure and/or undesired outputs. This could include unknown values ("X"), incorrect outputs, or incorrect clock transitions, as seen in Table 41 and Figure 74. In Table 41, the top event was set to "Output = X", to simulate an error/failure state with the register. In this case, an "Input" of "X", along with the "+" "Clock" transition on a rising edge clock, caused the "X" state to be passed to the output. The "Input" is not overruled by a "DATA", "Preset" or "Reset" signal (both "Reset" and "Preset" signals are in the error state of "X", so that do not pre-emp the "Input"). The Modelsim results in Figure 8 confirm this, where the inputs of "Reset" = "X", "Preset" = "X", "CE" ="1", and "Input" = "X", produce an "Output" of "X" (as discussed in sub-section 4.1.2.5) when the clock transitions on a rising edge.

Table 41: Prime Implicants for "Top Event = X"

Implicant 16 (Node)	Implicant 16 (State)	Implicant 16 (Time Step)
Clock	**+**	-1
Clock_Period	Normal	-1
Clock_Edge	**R_E**	-1
Clock_Enable	1	-1
DATA	0	-1
DATA_Input	No_DATA	-1
Input	**X**	-1
Preset	X	-1
Prev_Clock	0	-1
Reset_Signal	X	-1

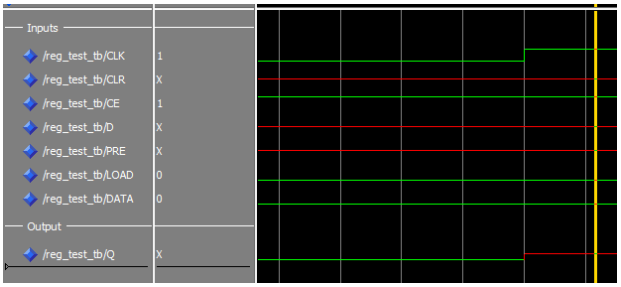


Figure 74: ModelSim results for FPGA register analysis (Top Event "Output = X")

4.2.2.3. Logic Block Results

The individual logic block models ("AND" and "OR") were analyzed, with the results shown in this sub-section. The analysis for the "AND" gate CLB was run for two time steps, with the Top Event set to "Logic_Block_Out ("LB_OUT") = 1" at time steps "0" and "-1". This produced 23 PIs, with two of them shown here. In Table 42, it is seen that the 4 inputs to the LUT ("In_1" … "In_4), are all in state "1", which could produce a value of "1" from an "AND" gate. When the clock transitions ("+"), and the Enable signal is "1", the "AND" value of "1" is loaded into the register. The register value is then selected by the Mux Select signal ("Mux_S"), which is then output from the logic block at TS = -1 and TS = 0, respectively. The corresponding Modelsim results are seen in Figure 75. It is seen that all four input signals are at a value of "1", the "Enable" is at "1", the "Reset" is at "0", so when the "Clock" transitions to "1", the "LB_Out" transitions to "1" (due to "AND" logic), and stays there for the next clock cycle. As the "Mux_S" signal is "0", the register is not bypassed.

Table 42: Prime Implicant for DFM FPGA Logic Block Analysis (Top Event "Logic Block Out = 1")

Implicant 9 (Node)	Implicant 9 (State)	Implicant 9 (Time Step)
In_1	**1**	-2
In_2	**1**	-2
In_3	**1**	-2
In_4	**1**	-2
AND_Out	1	-2
Clock	**+**	-2
Prev_Clock	0	-2
Clock_Enable	**1**	-2
Reset_Signal	**0**	-2
Clock	1	-1
Mux_S	**0**	-1
Prev_Clock	+	-1

Reset_Signal	0	0
In_1	**1**	0
In_2	**1**	0
In_3	**1**	0
In_4	**1**	0

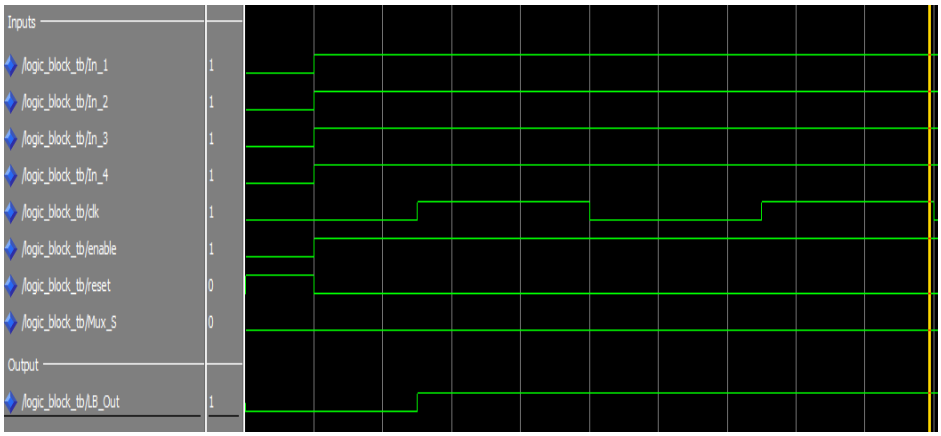


Figure 75: ModelSim results for "AND" logic block "Top Event = 1 at TS = 0 and TS =-1"

When discussing the "OR" gate CLB, an inductive analysis of one time step was chosen, with the results seen in Table 43. It was seen that each input was assigned a different logic state. Due to the "OR" gate and the 1164 definition, the output of the "OR" gate is a "1". The value is then stored in the register, as the "Enable" signal is "1", "Reset" signal is "0" and the clock transitions on the rising edge ("+"). This value is stored in the register for one time step (stored at TS = 0), due to the "MUX_S" value being "0". The signal stored in the register is then output at the next time step (TS = 1), making the CLB output value equal to "1". The Modelsim results for this simulation are shown in Figure 76. The "Reset" signal is "0", "Enable" signal is "1", and the "Mux_S" signal is "0", the register is used again. The inputs this time include several potential error states ("U" and "X"), which eventually resolve to "1" due to "OR" logic when the clock transitions (as discussed in sub-section 4.1.2.5).

Table 43: Sequence for "OR = 1" Inductive Analysis

Sequence 3 (Node)	Sequence 3 (State)	Sequence 3 (Time Step)
In_1	**1**	0
In_2	**0**	0
In_3	**U**	0
In_4	**X**	0
Mux_S	**0**	0

Clock_Enable	**1**	0
Clock	+	0
Reset_Signal	0	0
Logic_Block_Out	**0**	0
In_1	**1**	1
In_2	**0**	1
In_3	**U**	1
In_4	**X**	1
Mux_S	**0**	1
Clock_Enable	**1**	1
Clock	**1**	1
Reset_Signal	**0**	1
Logic_Block_Out	**1**	1

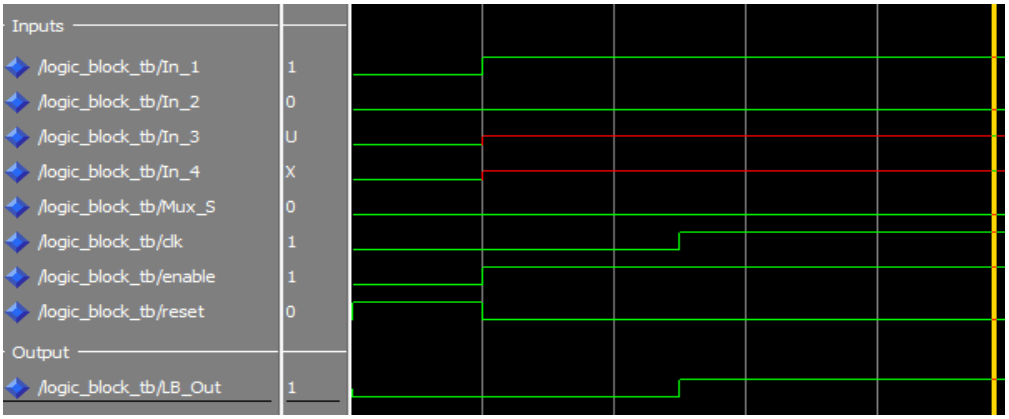


Figure 76: ModelSim Results for "OR" Logic Block Inductive Analysis

4.2.2.4. Platinum Signal Compensator Results

The results for the DFM analysis and Modelsim Simulations for the platinum signal compensator are presented here. The models were run for the Top Events of "Trip" and "Total Flux High", and "No Trip". In Table 44, it is seen that all of the system components are functioning correctly, except in one instance. The clock transitions "+", the "Reset" signal is "0", and the "Clock Enable" signal is "1", allowing the new (correct) values to be output from the registers. However, it was seen that the value of "D" in the state space model was higher than it should be, causing the value of "Phi" to be above the setpoint. A second input also read high (IV_3), causing the system to trip.

A similar implicant is shown in Table 45, however in this case the "D" value does not affect the top event. In this case, the "Trip" signal reads "0", in part due to an error with the "Input_Voltage", which

has a value of “X”. The other nodes/states were the same in the run shown in Table 45 as the run shown in Table 44, so only the differing states were included in Table 45.

Table 44: Implicant for “Trip” and “Total Flux High”

Implicant 11 (Node)	Implicant 11 (State)	Implicant 11 (Time Step)
Clock	**+**	-1
Clock_En_State	1	-1
Clock_En_Sig	**En_1**	-1
Clock_Period	Normal	-1
D	**D_High**	-1
I(k)_2	I(k)_2_Correct	-1
I(k)_3	I(k)_3_Correct	-1
Input_Voltage	Correct Voltage	-1
Prev_Clock	0	-1
Prev_Input_1	Correct_Input	-1
Prev_Input_2	Correct_Input	-1
Register_1_Out	Reg_Input_Correct	-1
Register_2_Out	Reg_Input_Correct	-1
Clock_Period	Normal	0
D	**D_High**	0
Phi	**Total Flux High**	0
I(k)_2	**I(k)_2_Correct**	0
I(k)_3	**I(k)_3_High**	0
IV_3	High_Voltage_3	0
Input_Voltage	Correct_Voltage	0
Register_1_Out	Reg_Input_Correct	0
Register_2_Out	Reg_Input_Correct	

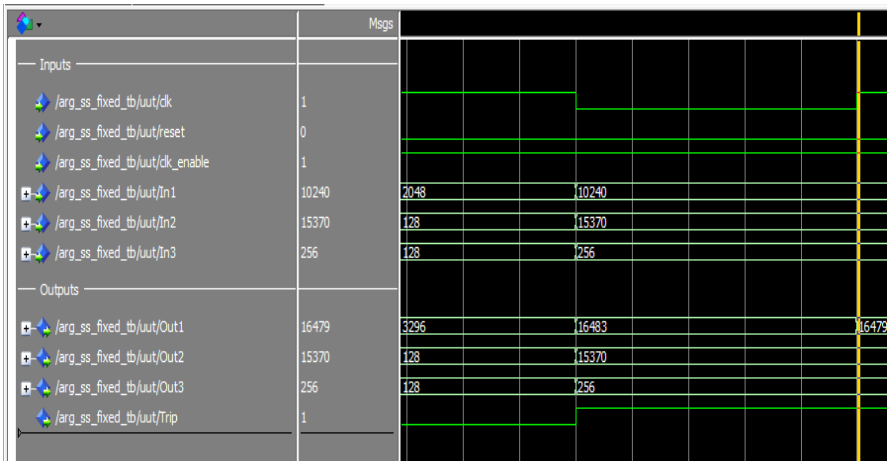


Figure 77: ModelSim results for “Trip” and “Total Flux High"

Figure 77 shows the Modelsim results for Table 44. The high value for "D" causes the flux to read "High", and assist in causing a trip (false alarm), when one of the other channels also denotes a trip. In this case, the Modelsim results do not specifically show the value for "D" (internal signals are not always monitored). However, it is seen in the Modelsim results that the Flux value ("Out1") is higher than it should be, for the given inputs. Figure 78 gives the simulation results for Table 45, where it shows "No Trip", due to some failure with the input. The "Input_Voltage" node ("In1" in Figure 71) has an input of "X", which points to some form of input failure, including those from other parts of the system (as discussed in sub-section 4.2.1.5). This leads to the value of "Phi" to also be "X", as seen in "Out1". This causes the 2oo3 logic to return a value of "0", and the "Trip" value is seen as "0" in the figure. This would have the potential to miss a trip, if the value for that input was supposed to be high enough.

Table 45: Implicant for "No Trip"

Implicant 3 (Node)	Implicant 3 (State)	Implicant 3 (Time Step)
Input Voltage	**Input Voltage_X**	-1
D	D_Correct	-1
I(k)_1	**X**	0
Phi	**Total_Flux_X**	0
D	D_Correct	0

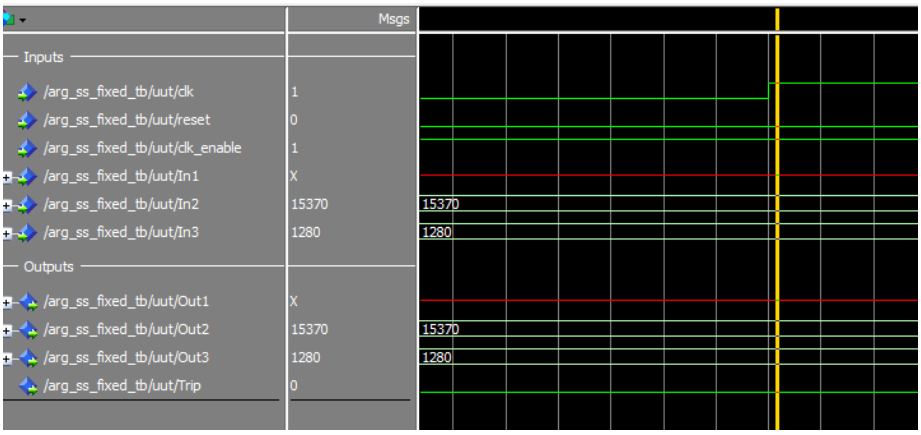


Figure 78: ModelSim results for "No Trip"

It should be note that the "X" value is considered to be a "metalogical" value, and is not a true logical value like "0" and "1" [41]. It is intended for use in simulation only, and would

not occur during actual hardware synthesis. In the case of an actual NPP safety system, if the value of the neutron power/flux was actually unknown, safe design principles would state that the system would actuate a trip, even if it later turned out to be a spurious actuation.

4.2.3. Conclusions of the DFM and Modelsim Comparisons

In sub-section 4.2.3, DFM was applied to model and analyze important aspects of FPGA-based systems that could find use in nuclear plant safety and control systems. These aspects included the underlying IEEE 1164 standard for VHDL, the use of Registers (D Flip-Flops), Logic Blocks and an implementation of a FPGA-based dynamic signal compensator and trip logic system. The analysis results were compared to the Modelsim simulations of the synthesized (where possible) VHDL code, which confirmed that DFM was able to correctly model the FPGA logic and properties, making DFM a potential option for the modelling and simulation of FPGAs and FPGA-based systems. With the effectiveness and suitability of DFM confirmed for modelling basic FPGA logic, components and test systems for generic failure modes, the next phase of the research work involved applying DFM to a more complex test system, using specific failure modes for FPGA-based systems. However, first detailed failure mode information for FPGA-based systems must be realized.

4.3. Preliminary DFM and FTA Comparisons

With effectiveness and suitability of DFM for FPGA-based systems determined, and the requisite FMEA data composed and categorised in a useful fashion, the next phase of the research work was to compare DFM and FTA, for a more complex and realistic FPGA-based test system. The information detailing the construction of the models, and the results from the analyses will be detailed in this section. This work was accepted for publication by the journal "*Reliability Engineering and System Safety"* [208].

4.3.1. Reliability Analysis Methods and DFM/FTA Comparisons

The reliability analysis methods chosen for this research were Fault Tree Analysis (FTA), and the Dynamic Flowgraph Methodology (DFM). In order to perform the comparisons, the appropriate software had to be used. There are a number of software packages that can perform FTA. The FTA software selected for this paper is known as the CAFTA (Computer Aided Fault Tree Analysis) software package that is available from the Electric Power Research Institute (EPRI). The CAFTA software tool has seen extensive use in the nuclear industry [209]. Additionally, CAFTA has been used by the aerospace and chemical process industries, including being used by NASA [209]. With regards to DFM modelling, there are few choices when it comes to analysis software, so the Dymonda software from ASCA Inc. was selected.

4.3.2. Software Calculation Methods

In terms of FTA, the analysis done by CAFTA is relies on the "MCSUB" approximation method to determine Top Event probabilities [98]. A second calculation method, known as the Direct Probability Calculator (DPC), is available with CAFTA. DPC is used to calculate exact Top Event probabilities [98]. While the CAFTA manual does not explicitly detail the quantification used by DPC, it does state that it calculates the upper and lower bounds during the TE calculation, making it appear to function similarly to the Inclusion-Exclusion principle. Tests by the authors of this paper on simple fault trees also support that conclusion.

Regarding DFM, the Dymonda software tool uses the “SUM” approximation [14], while the YADRAT tool utilizes the “MCSUB” approximation [123]. However, once the individual PIs have been determined, it is relatively straightforward to apply the “SUM” and “MCSUB” methods to either DFM implementation. Dymonda also supports an exact quantification procedure, by converting all of the PIs to Mutually Exclusive Implicants (MEIs), and taking a sum of those MEIs [14].

4.3.3. DFM vs FTA Literature Comparisons

FTA is a well-established reliability method. On the other hand, DFM is a comparatively new and has not been used as extensively as FTA. Although more research into the use of DFM has occurred in recent years, there has been little direct comparison between DFM and FTA methods for digital I&C systems analysis. However, there has been some information published in reports from the US NRC. One comparison was performed using a digital Feedwater Controller, similar to the actual control used in certain US NPPs. The results for the traditional methods were published in NUREG/CR-6997 [27], while the DFM results were published in NUREG/CR-6985 [12]. It was seen in NUREG/CR-6997 and NUREG/CR-6985 that DFM and FT were both able to determine the MCS/PI that had the highest probability, and both reports state that there were additional PIs found using DFM[12,210]. NUREG/CR-6985 refers to these PIs as “Risk Relevant” (as opposed to “Risk Significant)”, however it does not define what constitutes “Risk Relevant”, or how it differentiates from “Risk Significant” [12]. Additional statements given in NUREG/CR-6901 suggest that using Fault Trees and Event Trees overestimate the predicted Top Event frequencies, however there are not specific examples given in that report [8].

4.3.4. FPGA-Based Test System for DFM/FTA Comparisons

The test system for this paper was a generic one-trip parameter, one channel FPGA-based logic loop for reactor trips, developed using a single FPGA chip. It was developed with a reference to the EPRI TR-109390 document, and Westinghouse AP1000 Chapter 7 documentation, both of which are publically available [190,211]. A description of the test system is given in this sub-section.

4.3.5. Fault Tolerant Design

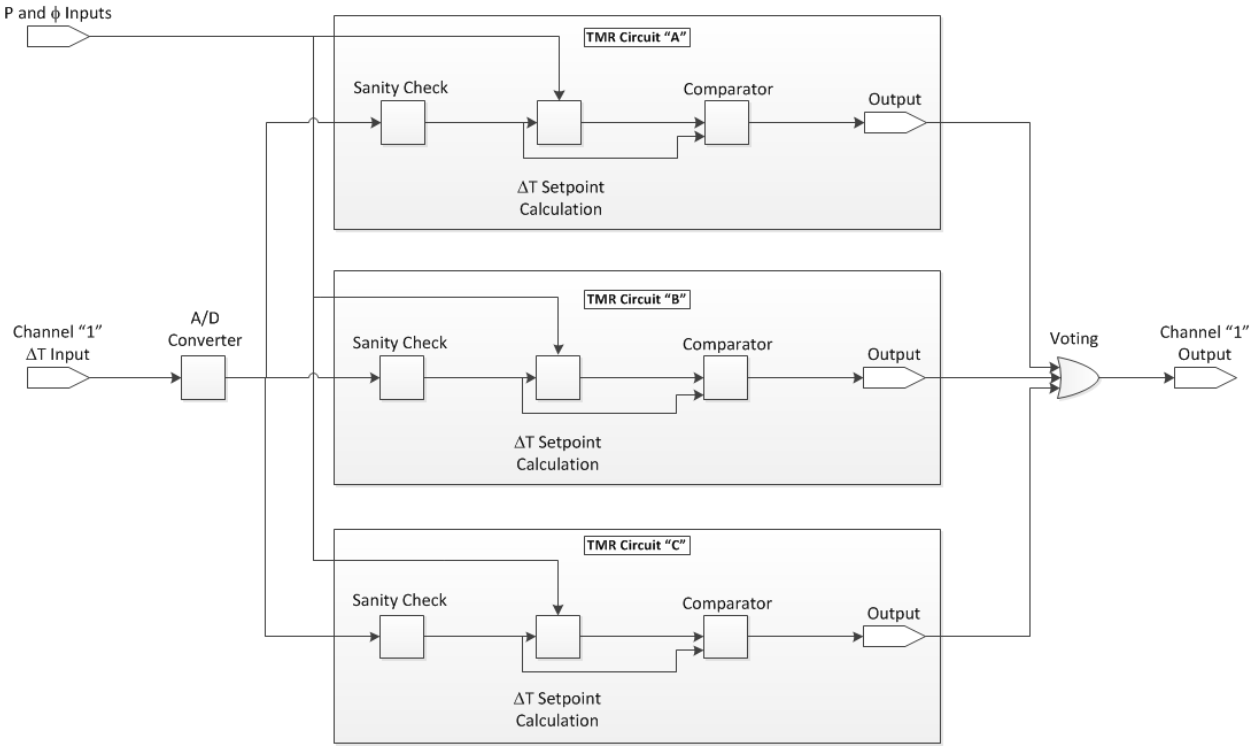


Figure 79: High level block diagram for the one-channel FPGA-based test system

The block diagram for the total system model was created using Microsoft Visio, and is shown in Figure 79. As seen in Figure 79, the test system consists of one channel, using a single trip parameter, utilizing Triple Modular Redundancy (TMR). The use of this configuration was done for the purpose of reliability, as it has often seen use in the fault tolerant design of FPGA-based systems in aerospace applications to protect against radiation-induced soft errors [171]. This fault tolerance method is also suggested in Section A.1.4 of IEC 61508-7 [133]. The three circuits are referred to as TMR Circuit "A", TMR Circuit "B" and TMR Circuit "C", respectively. It should be noted that additional fault tolerant methods for FPGAs were discussed in sub-section 4.3.7, and generic fault tolerant methods are widely discussed in the literature [212,213].

4.3.6. Subsystem Descriptions

In the test system, each of the TMR circuits contains three subsystems; Sanity Check (SC), Parameter Calculation (ΔT Setpoint), and the Comparator (COMP). A fourth subsystem, the Analog-to-Digital Converter (ADC), is considered outside of the TMR configuration, as seen in Figure 1. This gives a total of four unique subsystems in the test system. Figures 80-83 present a close-up view of the ADC and each of the sub-systems inside one of the parallel circuits of the TMR configuration.

The Parameter Calculation could represent several different trip parameters, such as temperature, pressure, neutron power, etc. For this paper, the Parameter Calculation was taken as an Overtemperature (OT) trip parameter, denoted as "ΔT Setpoint Calculation" in Figure 79. The calculation for the Overtemperature ΔT Trip Setpoint (TSP) requires information about the average temperature (T_{avg}), and also requires information about pressure and neutron power, so those measurements were included for the purpose of the TSP calculation. This would amount to 3 unique input measurements. However, as only the ΔT measurement is used to determine if a trip occurs (it is the only measurement used in the "Comparator" sub-system), there is said to be 1-trip parameter (ΔT measurement). The single parameter was used for demonstration purposes, due to the computational intensity of DFM, to ensure the system simulation would resolve in a reasonable time frame. In Figure 79, the pressure (*P*) and neutron power (Φ), were represented by a single input, to simplify the figure. A discussion of each sub-system is given in this section.

This test system differs significantly from the digital Feedwater System discussed in sub-section 2.3.2.9. That system took a more macroscopic approach, and analyzed the complete system (computers, pumps, valves, power supplies, etc.), but did not go into the details at the component level. The research in this paper focuses on the design of FPGA-based systems, so it considers the individual registers, muxs, decoders, comparators, interconnects, logic gates, and the failure modes that would affect those components. The DFM and Fault Tree models were built to include every component and every state of those required to produce the desired Top Events.

4.3.6.1. Analog-To-Digital Conversion (ADC) and Sanity Check

These two sub-systems were the smallest of the four, so they were included in one flowgraph model and accompanying block diagram. ADC is important to the FPGA system, as the signals that are input from the NPP sensors (in this case temperature readings) would be analog signals, and require conversion to digital signals for FPGA processing. After the signal conversion, the digital signal would be sent to the Sanity Check (SC). The Sanity Check includes a check against minimum and maximum values (as was done in this paper), and can include additional checks, such as rate checks. This check is done to ensure that only realistic values are passed on to the Comparator sub-system. Sanity checks would be applied to all input parameters in a real system. However in the FTA and DFM analyses in this paper, the modelling of the SC was restricted to the Temperature parameter, as it was the actual parameter used to trip the system. It was assumed for this test system the Sanity Check for the other input parameters would be applied by their own, separate Sanity Check circuits, so the inputs to the ΔT Setpoint Calculation is the output from the Sanity Check from those parameters. This was done to reduce the computational intensity, by removing duplicate SC sub-components, as the SC model would be identical for each parameter it was applied to.

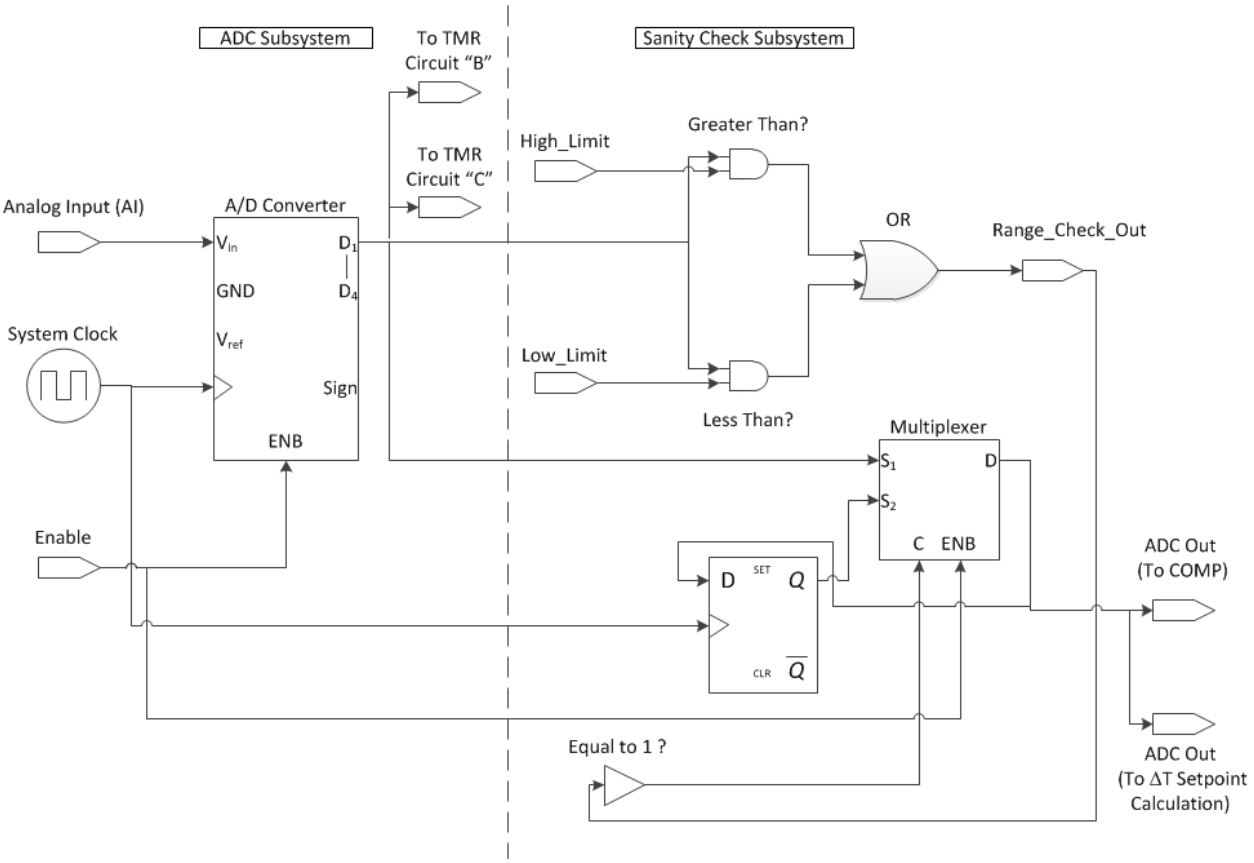


Figure 80: ADC and Sanity Check Block Diagram

There are high and low limits, and if either check is failed, then the "Range Check" is failed, and a value of "1" is passed to the Multiplexer (Mux). The Mux then decides on which output value(s) will be sent to the Comparator and the ΔT Setpoint Calculation. If the "Range Check" is passed (output from the OR gate is "0"), then the current signal is sent to the "P Register" input of the Comparator and ΔT Setpoint Calculation. If the "Range Check" fails, then a previous, correct value stored in the register will be passed to the Comparator and ΔT Setpoint Calculation instead. A block diagram for the ADC and sanity check models is seen In Figure 80. In the model the ADC and Sanity are not part of the same block, however they were included in the same block diagram in Figure 80 to simplify the test system diagram.

4.3.6.2. Trip Parameter (Over Temperature) Calculation

The OT trip parameter (ΔT) exists to protect the reactor core from the departure from nucleate boiling, based on coolant temperature, power, pressure and axial power distribution. This trip will occur if the transient is slow in comparison with the transient piping delays between the temperature detectors and the core, and if the pressure value is between the high and low pressure trip setpoints. The ΔT setpoint is calculated as follows [190]:

$$\Delta T_{Setpoint} = \Delta T_0 \left[K_1 - K_2 \left(\frac{1+\tau_1 s}{1+\tau_2 s} \right) \left(T_{Avg} - T_{Avg}^0 \right) + K_3 (P - P_0) - f_1(\Delta\Phi) \right] \quad (52)$$

The reactor is tripped if:

$$\frac{\Delta T(1+\tau_4 s)}{(1+\tau_5 s)} \geq \Delta T_{setpoint} \quad (53)$$

Here, K_1, K_2 K_3 are pre-set constants,τ_1, τ_2, τ_4, τ_5 are time constants to compensate for instrument, piping delays and lead-lag filters (lead-lag compensators). The terms P and P_0 represent the Pressurizer pressure and normal operating pressure, respectively. With regards to the temperature parameters, ΔT_0 is the indicated ΔT at rated thermal power, T_{Avg} is the average temperature of the reactor coolant, and T_{Avg}^0represents the nominal T_{Avg} at the rated thermal power. The final term ($f_1(\Delta\Phi)$), is a function based on the difference in the neutron flux between the flux signals of the upper and lower ionization chambers. The full definition of each parameter and constant can be found in Reference [190]. The output of the setpoint calculation is then sent to the "Q Register" input of the Comparator.

It should be noted that the average value of the ΔT measurement (T_{Avg}) is needed to calculate the ΔT setpoint. This T_{Avg} value would be calculated using four separate loops (with each loop containing 2 hot leg measurements and 1 cold leg measurement) [29]. As the test system in this paper was designed to be a simplified version of a realistic test system, the average value was not presented, and the system was assumed to use one loop (one ΔT measurement). Therefore, the ΔT measurement is passed from the Sanity Check to the Comparator and the ΔT Setpoint Calculation. This simplification was employed for testing purposes, to ease the computational burden when using DFM.

The lead-lag filter is shown in a separate block diagram model for clarity. In order for the filter to be used on an FPGA, the analog transfer function must be digitized (transformed to a discrete-time transfer function). The terms C_1, C_2 and C_3 are the coefficients of the z transform variable in the discrete time transfer function. This compensator is included to compensate for piping and instrument delays in the

system. The block diagram model for the filter is the representation of a digital lead-lag filter Register Transfer Level (RTL) netlist. The block diagrams for the OT parameter calculation and lead-lag filter are shown in Figures 81 and 82, respectively.

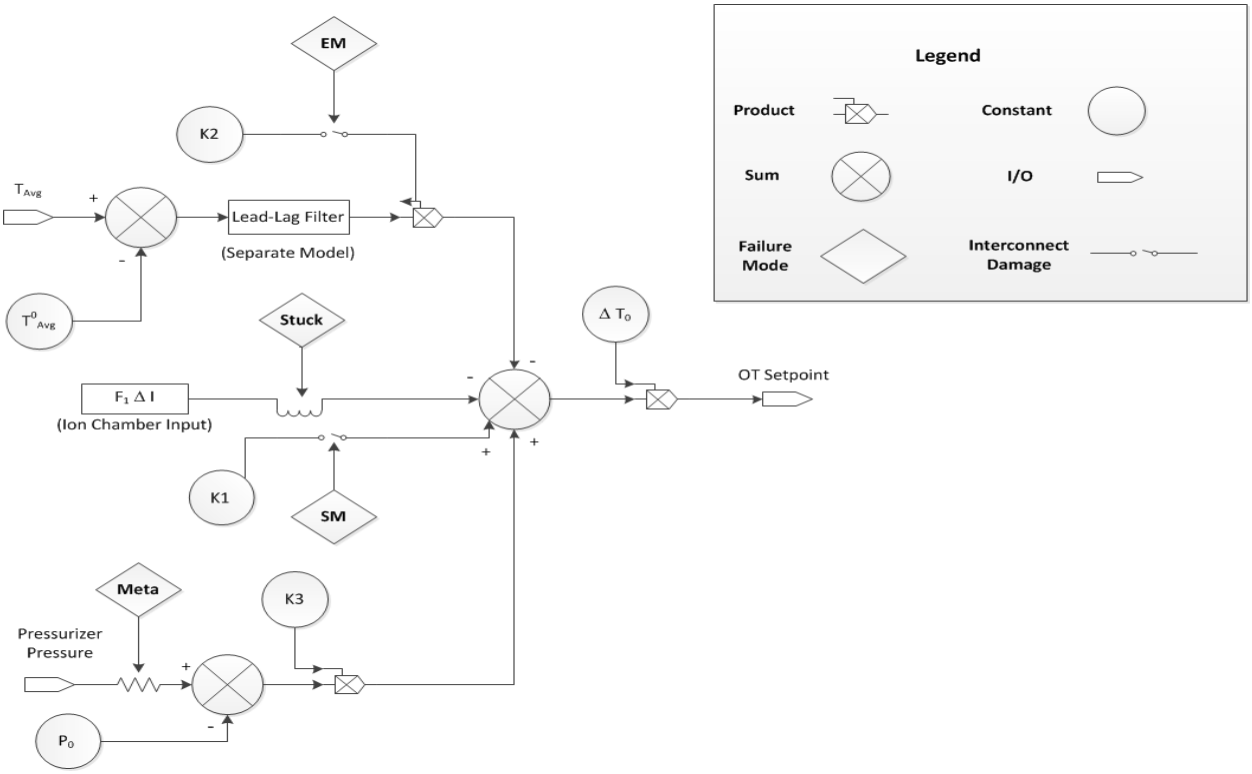


Figure 81: Overtemperature Calculation Block Diagram

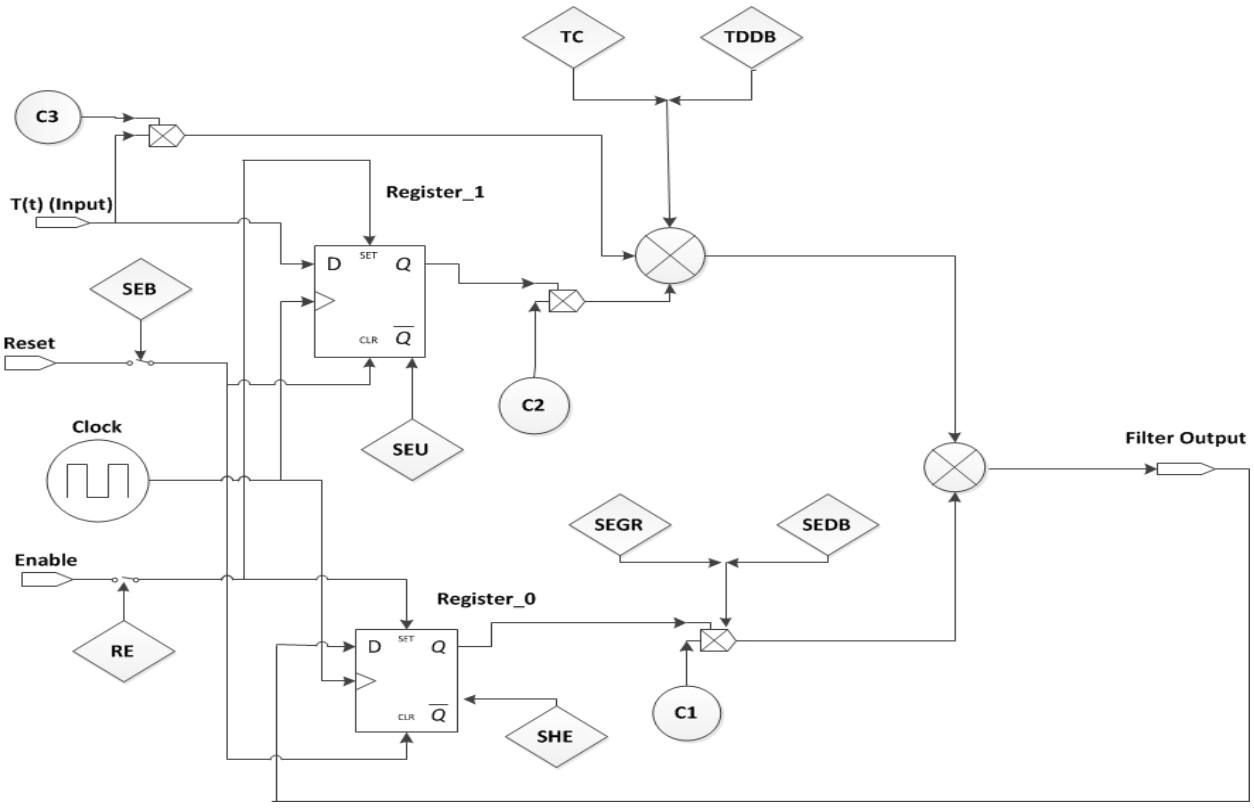


Figure 82: Lead-lag filter block diagram (part of OT calculation)

4.3.6.3. Comparator

The final component in the test system is the Comparator. The Comparator consists of two input registers: "P" and "Q". The "P" register receives the input signal from the ADC and Sanity Check block, which represents the measured signal from a temperature sensor. The "Q" register stores the $\Delta T_{setpoint}$, which was calculated in the OT parameter block. The value of the "P" register is compared against the setpoint stored in the "Q" register. If the "P" value is greater than or equal to the "Q" value, a trip signal will be sent to the "Trip Register" to be stored, and then output to the voting logic. The block diagram for the Comparator model is shown in Figure 83.

The Decode block in the Comparator diagram allows for the generic Comparator to be used for multiple trip conditions. The setpoint in this paper used a "greater than or equal to condition", however other trip conditions could require a "less than or equal to" condition. The decoder block allows for different trip conditions to be used with the same generic Comparator component. For the purpose of this research, only the trip condition in Equation 52 is considered.

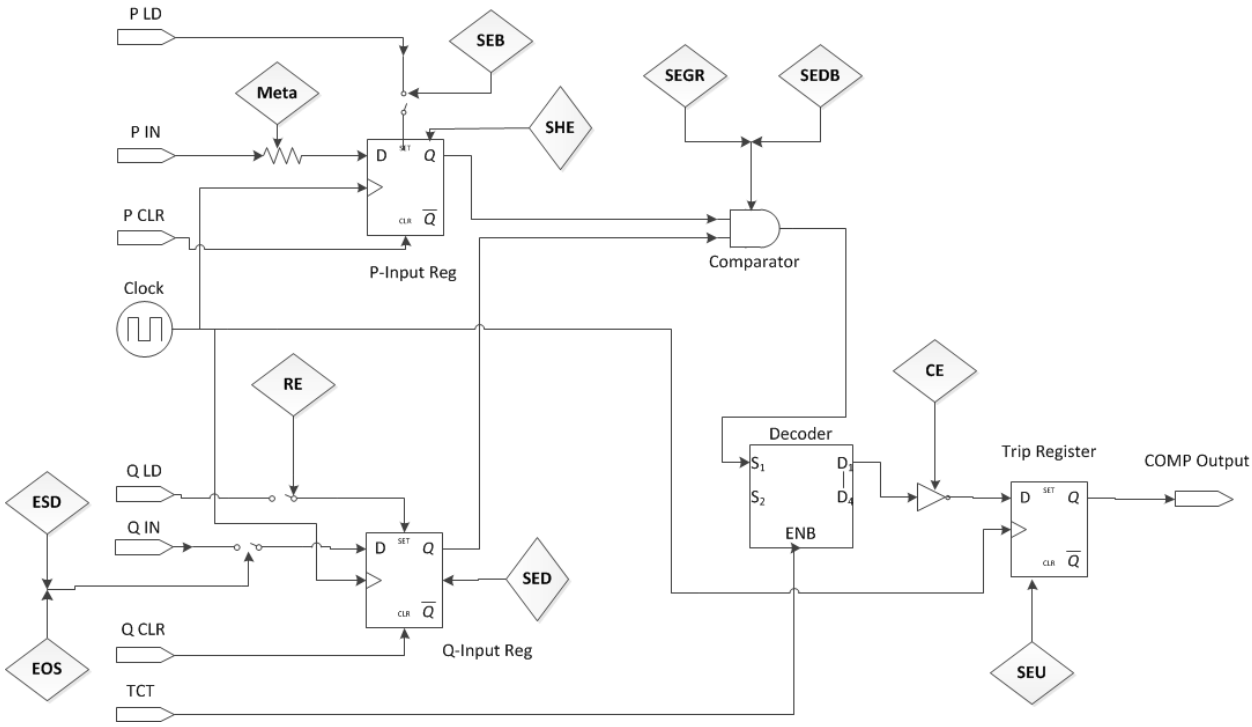


Figure 83: Comparator Block Diagram

4.3.7. Failure Modes

There are a number of potential failure modes that could affect the FPGA-based safety system, although not all of them are unique to FPGAs [126]. It should be noted that only failures that occur during the "Operation" stage of the lifecycle were considered here, as it was assumed that all failure modes that could be introduced in the design stage were identified and eliminated. The failure modes that were considered largely came from two categories. The first were radiation-induced failure modes, commonly referred to as Single Event Effects (SEEs). The interaction of ionizing radiation with the semiconductor material of FPGAs can produce both soft (temporary) or hard (permanent) errors. The other main failure category was failures due to the aging process.

Aging (wear) effects will eventually damage the hardware in the FPGA, such as destroying the logic gates or programmable interconnects that the FPGA logic requires. Values for the probabilities of the selected failure modes were included based on data in the literature for metastability [214], aging failures [160], SEEs [215,216], ADC failure/stuck input [217] and Electrostatic Discharge (ESD) [176]. Electric Overstress (EOS) was assigned the same probability as ESD. These references were then used to calculate the

Mean Time To Event (MTTE). To incorporate these values into the DFM and FTA models, the MTTE was transferred into a failure rate (1/MTTE), and the resulting values were used as probabilities in the model [218]. These numbers are taken as an average from a range of values to be realistic for the purpose of the failure analysis in this paper. However, as the research is focused on the comparison of DFM and FTA, the individual probabilities of the failure modes were not the primary concern, as long as those values were in a realistic range.

The failure modes in this research were restricted to the Comparator and OT subsystems. This was done to ensure there were a large variety of potential failure modes, however without going to the extent of making the individual failure modes negligible. Certain failure modes were not considered in this paper. These include design errors such as logic (programming) errors, as it was assumed that those errors would be eliminated in the design phase. Errors that affect the clock, such as Hot Carrier Effects (HCE) or Bias Thermal Instability (BTI) were not discussed at this stage of the research. Common Cause Failures (CCF) also was not considered, as discussed in sub-section 4.3.8.

A list of all SEE failure modes is given in Table 46, while Table 2 lists the remaining failure modes [141]. The first four rows in Table 47 correspond to aging process failures; while the fifth and sixth rows represent maintenance (human factor) related failures. Row seven would correspond to a clock/timing issue, and row eight represents an input failure, due to a malfunctioning ADC. More information on these failure modes is found in the literature [126].

Table 46: Selected SEE FPGA failure modes

Single Event Upset (SEU)	Temporary information corruption in a memory element (Logic bit flip)
Single Event Disturb (SED)	Temporary information corruption in a memory element (Bit in unknown logic state)
Single Event Functional Interrupt (SEFI)	Corruption of entire data path and loss of system operation
Single Hard Error (SHE)	Permanent state change in memory element
Configuration Error (CE)	Permanent logic inversion
Routing Error (RE)	SEE-induced interconnect damage
Single Event Burnout (SEB)	High-current induced destructive burnout
Single Event Gate	Rupture of the dielectric

Rupture (SEGR)	material in gates
Single Event Dielectric Breakdown (SEDB)	Rupture of the dielectric material in gates

Table 47: Additional FPGA failure modes

Time Dependent Dielectric Breakdown (TDDB)	Destruction of logic gates and look-up tables due to dielectric breakdown
Thermal Cycling (TC)	Destruction of logic gates and look-up tables due temperature cycling
Electromigration (EM)	Electron flow resulting in the destruction of interconnects
Stress Migration (SM)	Thermal-mechanical stress resulting in the destruction of interconnects
Electrostatic Discharge (ESD)	Damage to the FPGA chip or board due to discharge of static electricity
Electrical Overstress (EOS)	Damage to the FPGA chip or board due to inadequate electrical protection
Stuck Input (Stuck)	Values from ADC input stuck at “0” or “1”
Metastability (Meta)	Oscillations between “0” and “1”, cause “Unknown” value

4.3.8. Common Cause Failure (CCF)

For the purpose of this paper, Common Cause Failures (or Common Mode Failures) were not considered. As indicated by Figure 79, this is a one-channel, one-trip parameter, TMR configuration. In the case of CCF, it is used when analyzing failure at the system level, such as the simultaneous failure of multiple channels. An example would be the failure of multiple channels in a 2oo4 system, where a single CCF would cause two, three or four channels to fail at once. Therefore, CCF was not considered at this time to allow for the focus of the research to be on the DFM/FTA comparisons of modelling failure modes. CCF in digital I&C systems has been identified in the literature as an important area of future research, and will be specifically considered in future work [127].

4.3.9. DFM and FTA Model Construction

While DFM and FTA can be used for similar purposes, the construction of the models used during the analysis is different for both methods. Additionally, the implementation of the failure modes will be different for both methodologies, as discussed in this sub-section.

General Model Construction

In the case of DFM, only one model is constructed to represent the entire system. The Top Event(s) for the analysis are then selected using a state, or combination of states, for the discretized nodes. On the other hand, with FTA every Top Event will require a separate fault tree to be constructed. As an example, the two Top Events considered in sub-section 4.3.10 required one DFM model, and two fault trees (one for each Top Event).

In the actual DFM model, there were 132 total nodes in each circuit, so 396 nodes in the total system (3 circuits). This includes 59 nodes (per circuit) that were source nodes. Each node was discretized into 2-3 states (for the failure modes), and 4 states for the constants, and 5 states for the inputs (as well as the outputs of the registers). In the case of FTA, each individual node/state combination requires its own Basic Event. This led to a total of 172 Basic Events in each circuit, or 516 Basic Events for all three circuits.

The different methodologies also handle the failure modes differently. With DFM, additional nodes were added, and then discretized to include the relevant failure states, similar to what is shown in Figures 82-84. These nodes were connected to transfer boxes, to include the effects of the failure modes on the FPGA components. On the other hand, to include the failure modes in the two fault trees, additional basic events had to be added.

DFM SHE Failure Mode Implementation

To properly model the FPGA system/failure modes, the DFM model included registers, which would store and output data on clock cycles. A section of the DFM model that focused on the "P" register is seen in Figure 84.

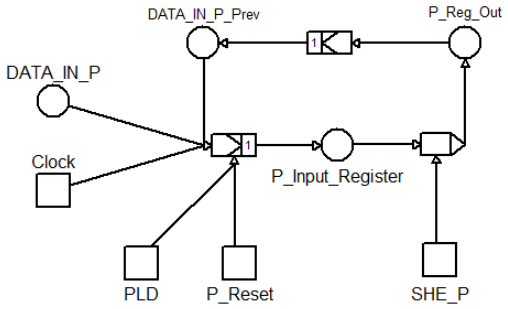


Figure 84: DFM Model Section for "P" Register

The inputs include the reset ("P_Reset"), load/enable ("PLD"), clock, input signal ("Data_In_P"), and the signal from the previous time step ("Data_In_P_Prev"). The previous signal is needed, as the register will only output the new value at the correct clock cycle (it was assumed in this paper that all registers used a "rising edge" trigger). After the register value is derived, the SHE node (failure mode) was applied. An example section of the decision table for the register ("P_Input_Register") is given in Table 48, while an example section of the decision table for the "SHE" failure ("P_Reg_Out") is seen in Table 49.

Table 48 provides an example of each of the five outputs ("Out" column) for the "P" Register: "0", "High", "Low", "OK", and "X" (Unknown). The "Data" column refers to the register input, with the "Data_P" column referring to the previous register data. The column "PLD" represents the "Load" ("Enable") signal. Lastly, the "-" denotes the "Don't Care" value.

Table 48: Sample of "P Register" Decision Table

Reset	Data	PLD	Clock	Data_P	Out
1	-	-	-	-	0
0	High	-	-	High	High
0	Low	1	1	-	Low
0	OK	-	-	OK	OK
0	-	-	0	X	X

Table 49: Sample of "SHE" Failure Decision Table

P_In	SHE_P	P_Reg_Out
Ok	SHE_P_OK	P_OK
*	SHE_P_1	P_High
*	SHE_P_0	P_0
X	SHE_P_OK	P_X
0	SHE_P_OK	P_Low

In Table 49, the "SHE" failure was applied to the output from the "P" Register, which also produces the five possible output states. It is seen that regardless of what the input value is ("P_In"), the SHE failure will force the output into either a "High" or "0" state, depending on which bit the failure occurs. It should be noted that Tables 48 and 49 could have been combined into one larger table, however due to the number of nodes/state combinations using one table was impractical. Therefore, two separate transfer/transition boxes were included, one for the register, and one for the SHE failure mode.

FTA SHE Failure Mode Implementation

To represent the same SHE failure as seen in this sub-section with a fault tree requires a much different procedure. An example of a section of the overall fault tree is given in Figure 85.

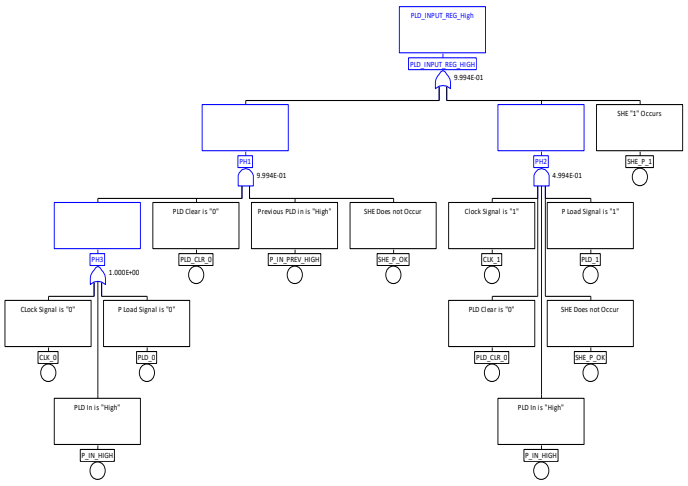


Figure 85: Fault Tree for the "High" Output of the "P" Register

Figure 85 shows the different basic events required to cause a "High" output value from the "P" register. As with DFM, the "SHE_1" state (SHE forcing a significant bit to a value of "1") will always cause a high

value. In the case of the fault tree, the derivation of the register value and the application of the SHE failure mode were done in one tree. This is because the FTA model used several smaller fault trees, as opposed to one/two larger decision tables, making it a more reasonable alternative. The values for the previous inputs were kept in the fault tree, and may work for one time step, however this model runs into difficulties when multiple time steps are concerned, as discussed in sub-sections 4.3.11.3

DFM and FTA Model Differences

There were two significant differences between the fault tree and DFM implementations. The first difference is that the each of the possible five output states requires its own fault tree, so five fault trees were created to represent the "P" register output (as well as outputs from other registers), while this could be done in DFM in using one transition box. Overall, the two main fault trees that represented the two Top Events were constructed from sets of smaller fault trees, such as the one seen in Figure 85. Secondly, each of the fault trees may not contain the same failure mode information (states). This may cause incorrect MCSs, as the different states would be mutually exclusive.

For example, in Figures 84 and 85, the "SHE" failure mode was pictured. The node representing the "SHE" was discretized into 3 states, "SHE_1", "SHE_0", and "SHE_OK", with the last one indicating that no "SHE" failure occurred. In DFM, the analysis will not allow the node to be in two different states during the same time step, for the same PI. This would violate one of the "physical consistency rules", which underpins the DFM calculation. In the case of FTA, there is no such "physical consistency rule", which could lead to mutually exclusive Basic Events existing in the same MCS. Taking the "SHE" example, it could be possible to have more than one of the three states (Basic Events) in the same MCS, depending on how the fault tree was constructed, and the failure modes that were included. This potential issue is not limited only to the failure modes, and could affect other nodes as well, such as the clock, reset or load/enable signals. This potential issue is further discussed in sub-sections 4.3.11.3

4.3.10. Test System Results for DFM/FTA Comparisons

There were two separate test runs carried out for both the DFM and FTA methods. The first run was for a Top Event of Comparator Output "1", to simulate a "Spurious Trip". The second run for a Top Event of Comparator Output "0", to simulate a "Missed Trip" or "Fail to Trip". Both of these Top Events would be considered as system failures, so both were analyzed during this research. Initial conditions for the DFM and FT models were used, to ensure that only the system failures were being modelled. For example, a high input from the "P" register should cause a system trip, so a Comparator Output of "1" would not be a failure in that instance. For this research, the initial conditions were used to see when a high "P" value did not result in a system trip, which would count as a "Missed Trip".

Sub-section 4.3.10.1 presents the results for a simple test case using a dynamic component known as a Register (sometimes referred to as a D Flip-Flop). In this case, the register output was looped back into the register input. The results from the CAFTA analysis are given in Sub-section 4.3.10.2, including a discussion of a possible shortcoming with FTA. Sub-section then presents the results from the Dymonda models. The Top Event probabilities are on a per demand basis, as reactor trip systems are actuated on demand. The probabilities shown for the literals in sub-section 4.3.10.3 represent the failure probability at that time step. However, as the analysis was assumed to run over a short time span (i.e. on the order of FPGA clock cycles, such as the time taken for the system to react to an input signal), any time-dependant changes in the failure probabilities would be negligible. Therefore, the probabilities were set as a constant value. Additionally, as one time step was used in the analysis, the probability for each state did not have the ability to change. It should be noted that for all test runs, the truncation limit was 1.0E-12.

4.3.10.1. Register Results

As previously discussed, the register is the basic storage element of an FPGA, and is required for logic that contains control loops and feedback [202]. This component is largely responsible for causing the time dependant behaviour in the test system examined here. Before comparing the results from the larger test system, it is prudent to compare the results from a simple register, with an injected failure mode. An SEU was used, to compare how DFM and FTA would handle the combination of data inputs, "Clock", "Reset", and "Enable" signals, and the SEU. The Top Event that was considered was the Register in a "High" state, and was run for one time step and two time steps (in the DFM model). This Top Event

could be due to inputs to the register also being in the "High" state, or it could also be due to the SEU. The results are shown in Table 50 and Table 51, respectively. With regards to Table 51, the row labelled "Time Steps" denotes that the model was run twice. The first run was performed for one time step, and the second run was performed for two time steps.

Table 50: DFM Results for Register with SEU

Time Steps	1		
Method	SUM	MCSUB	EQ
Probability	2.610E-01	2.377E-01	1.890E-01
PI #	13	N/A	N/A
Time Steps	2		
Method	SUM	MCSUB	EQ
Probability	1.00	7.631E-01	7.201E-01
PI #	25	N/A	N/A

Table 51: FTA Results for Register with SEU

Method	CSG	DPC
Probability	2.38E-01	1.799E-01
MCS #	13	N/A

In the case of the run with two time steps, it is seen that the "SUM" calculations returns a value of "1.00", which would suggest a 100% chance of failure. However, this is a known issue with the "SUM" approximation, as it can result in a Top Event probability that is greater than or equal to 1 (the Dymonda tool will automatically truncate the result to "1.00"). It is seen that the "MCSUB" and "EQ" methods return values that are significantly less than a probability of 1. For this reason, the "MCSUB" method is often preferred to the direct summation. The full DFM and FTA results for this Register model are listed in Appendix I. The FTA and DFM results for the larger models, such as those in sub-sections 4.3 and 4.4 were too large to be shown in this document.

4.3.10.2. CAFTA Results

The results from the CAFTA analysis for the "Missed Trip" and "Spurious Trip" Top Event are presented here. The data collected including the number of MCS, the Top Event Probabilities from the "CSG" and

"DPC" calculations, and the analysis time for the DPC method. Table 52 shows the results for the "Missed Trip" Top Event, and Table 53 presents the results for the "Spurious Trip" Top Event.

Table 52: FTA Results for "Missed Trip" Top Event

CSG	DPC	MCS #	Analysis Time (DPC)
4.20E-04	1.09E-05	2535	7.5s

Table 53: FTA Results for "Spurious Trip" Top Event

CSG	DPC	MCS	Analysis Time (DPC)
3.33E-04	1.60E-06	2457	13.0s

An issue with FTA was seen when inspecting the CAFTA results. The registers, such as those in the Comparator subsystem rely on clock signals to capture and store the new input data. The clock signal has an obvious time dependency, and will cycle between the "0" and "1" states. Depending on the other conditions (such as "Reset", "Enable" or input signals), the output value comes when the clock is at "0" (old value), or a new value when the clock is transitioning from "0" to "1" (known as a "rising edge trigger") [202]. This entails that the register output could occur with the clock in either a "0" or "1" state, but not with the clock in both states at once. CAFTA is not fully capable of discriminating between clock cycles, and is such, certain MCS include the clock in state "0" and "1", which is not possible. The information pertaining to this issue is seen in Table 9. The "% Change" column refers to the percentage of the total number of MCS that is represented by those impossible MCS.

Table 54: Impossible CAFTA Minimal Cut Sets

Model	Probability	MCS #	% Change
Missed Trip	1.67-04	1260	39.86
Spurious Trip	1.31E-04	1138	39.27

As seen in Table 54, the overlap of clock states represents a significant percentage of the total Top Event probability and a large number of MCS. This makes using FTA for FPGA-based systems (at this level of extrapolation) problematic, as it will overestimate the Cut Sets and the Top Event probability. A

potential way to alleviate this issue is to use two runs, one for each clock state. The results from those analyses are seen in Table 55

When comparing the results from Tables 54 and 55, it is seen that considering only one clock state has a sizeable effect on both the number of MCS, and the "CSG" Top Event probability, however the "DPC" probability is very similar to the clock state "1" run, when compared with Tables 52 and 53. As expected, the runs with clock state "1" have a much larger representation, as the clock transition from "0" to "1" is when the new data will be stored in the register. This overlap of clock states is not a concern when using DFM, as the clock can be forced to cycle ("0" -> "1" -> "0" -> "1"), and will not result in multiple clock states in the same time step.

Table 55: FTA Results for Individual Clock States

Model	Clock State	CSG	DPC	MCS #
Missed Trip	1	6.67E-05	1.09E-05	1079
Missed Trip	0	1.27E-08	4.25E-09	9
Spurious Trip	1	1.59E-06	1.35E-06	663
Spurious Trip	0	2.03E-06	7.10E-07	69

4.3.10.3. DFM Results

The test system was then run using the Dymonda software, to obtain the DFM results. The "Missed Trip" and "Spurious Trip" Top Events were considered again, with the results given in Table 56 and Table 57, respectively. Each table contains the total number of PIs, the Top Event probabilities calculated from the "SUM" "MCSUB", and "EQ" methods, and the analysis time. In both cases, the models were run for one time step. In Tables 56 and 57, it was seen that the "Sum" and "MCSUB" methods produced almost identical results, so they were grouped together in the same column.

Table 56: DFM Results for "Missed Trip" Top Event with one Time Step

SUM/MCSUB	EQ	PI #	Analysis Time
1.556E-05	1.554E-05	53	1.6s

Table 57: DFM Results for "Spurious Trip" Top Event with one Time Step

SUM/MCSUB	EQ	PI #	Analysis Time
3.117E-05	3.116-05	63	1.7s

It was seen that in both cases, the "SUM" and "MCSUB" values were almost identical, and were the same past 3 decimal places, and as such were grouped together. As expected, the "EQ" result is slightly lower, as it only considers MEI.

4.3.11. Discussion of Test System Results for DFM/FTA Comparisons

This section presents a detailed discussion of the differences between the DFM and FTA analysis results. Sub-section 4.3.11.1 compares and contrasts individual MCSs and PIs; sub-section 4.3.11.2 presents a comparison of the Birnbaum Structural Importance (BSI) method; sub-section 4.3.11.3 discusses potential reasons for these differences.

4.3.11.1. Test System Results for DFM/FTA Comparison

The comparison between the actual MCS and PIs is an important consideration when comparing DFM and FTA methods. It was seen that there were some similarities and some differences in the PIs and MCS that were considered. A selection of those is presented in this section. Here, TMR Circuit "B" and TMR Circuit "C" refer to the redundant circuits in the TMR configuration. The circuits were simplified for "B" and "C" in the data tables for brevity.

Dymonda and CAFTA return an identical PI/MCS to the PI shown in Table 58. In this case, a SEGR event happens, causing damage to an important logic gate in the FPGA, resulting in the system failing to trip.

The CAFTA result contains a higher probability, 5.93E-06, due to the way the system probability is calculated. A differing PI/MCS is seen in Table 59.

Table 58: Similar DFM PI and CAFTA MCS for "Missed Trip"

Node	State	Time Step	Prob
Clock	1	-1	N/A
SEGR	SEGR_Fail	-1	5.95E-04
SEU (T)	No SEU	0	9.993E-01
TMR Circuit "B"	0	0	2.285E-03
PI Probability:			1.359E-06

In Table 59, one of the PI that occurs when the clock is in state "0" is seen. The DFM analysis returns the simple PI, with only 4 entries, with the Missed Trip due to an SEU in the "Trip Register", which is the final register in the system that stores the trip signal. The node "Trip_Reg (Prev)" holds the previous value of the "Trip Register", delayed by one time step. The CAFTA analysis however, retains many more entries in the MCS that contain both "Clock 0" and "SEU T" states, which drastically increases the size of the MCS, and as well as the probability. In that case, the MCS probability is 2.18E-09, close to two orders of magnitude smaller than the DFM value.

Table 59: Different DFM PI and CAFTA MCS for "Missed Trip"

Node	State	Time Step	Prob
Clock	0	-1	N/A
Trip_Reg (Prev)	1	-1	N/A
SEU (T)	SEU	0	6.53E-05
TMR Circuit "C"	0	0	2.285E-03
PI Probability:			1.492E-07

Considering the results from the "Spurious Trip" Top Event, it was seen that there were similarities in the prevalence of the CE failure mode in the Decoder, designated as "CE (D)". Both methods produced multiple PI/MCS similar to the one shown in Table 60, containing this failure mode, and it was consistently ranked near the top of the results. Similarly, both methods ranked the PI/MCS with SHE (P)

value of “1”, as the next failure mode most likely to cause the Top Event. This failure mode pertains to a SHE in the “P” register in the Comparator, causing a bit to be permanently stuck in the “1” state. A difference in the MCS/PI for this Top Event is shown in Table 61. The DFM results ranked the PI with the SED failure mode in the “Q” register comparatively high to the CAFTA results. In CAFTA, the SED MCS are ranked below MCS with other failure modes, such as EOD and EOS, which is not the case in DFM.

Table 60: Similar DFM PI and CAFTA MCS for “Spurious Trip”

Node	State	Time Step	Prob
Clock	1	-1	N/A
Trip Type	00	-1	0.5
CE (D)	CE Error	-1	5.95E-04
SEDB	No SEGR	-1	9.94E-01
SEGR	No SEDB	-1	9.94E-01
SED	No SED	-1	9.999E-01
SEU (T)	No SEU	0	9.993E-01
Circuit “C”	0	0	2.285E-03
PI Probability:			1.514E-07

Table 61: Different DFM PI and CAFTA MCS for “Spurious Trip”

Node	State	Time Step	Prob
Clock	1	-1	N/A
Trip Type	01	-1	0.5
CE (D)	No CE	-1	9.94E-01
SEDB	No SEGR	-1	9.94E-01
SEGR	No SEDB	-1	9.94E-01
SED	SED (Q)	-1	6.61E-06
SEU (T)	No SEU	0	9.993E-01
Circuit “C”	0	0	2.285E-03
PI Probability:			3.36E-09

4.3.11.2. Birnbaum Structural Importance Comparison

The BSI measure, is a structural importance measure (meaning that it does not require probabilistic information), used to compare the relative importance of components. It was selected for use in this paper as it is the fundamental importance measure based on which important system information can

be revealed. It is the oldest and one of the most well-known importance measures [219]. Additionally, it is relatively simple to derive a DFM BSI that is analogous to the BSI used in FTA, which has also seen use with the YADRAT tool at VTT [219]. The BSI measure was used to probe further differences in the results of the DFM and FTA results, due to the differences in model construction and analysis methods. The differences in the BSI results are also discussed in this sub-section.

In DFM, the BSI value is calculated as [42]:

$$\boldsymbol{BSI_{DFM}} = \frac{\cup_{j=1}^{m} \boldsymbol{PI_i^j}}{\boldsymbol{PI}} \tag{54}$$

Equation 54 entails that the BSI for DFM is the number of PI containing component *i* divided by the total number of PI. This works the same way with FTA, except with MCS replacing PI. BSI can be applied to both the nodes and states, where the BSI for the node is the sum of the BSI for all states in that node. A comparison of the top 5 nodes and states by BSI for the DFM and FTA are shown in Tables 62-64 ("Missed Trip") and Tables 65-67 ("Spurious Trip").

Table 62: BSI Comparison for "Missed Trip" Top Event

Node	DFM	FTA (Clock 0)	FTA (Clock 1)
SEU (T)	0.981	0.889	0.999
CE (D)	0.868	0.889	0.995
SHE (P)	0.830	0.889	0.997
SEGR	0.491	0.889	0.997
SEDB	0.491	0.889	0.997

Table 63: DFM State BSI Comparison for "Missed Trip" Top Event

State	BI
SEU (T)	0.491
SHE_1 (P)	0.339
CE (D)	0.301
SHE_0 (P)	0.113
SEGR/SEDB	0.0377

Table 64: FTA State BSI Comparison for "Missed Trip" Top Event

State	Clock 0	State	Clock 1
SEU (T)	0.889	CE (D)	0.437
N/A	N/A	SEU (T)	0.311
N/A	N/A	SHE_0 (P)	0.168
N/A	N/A	SM	0.152
N/A	N/A	RE	0.111

Inspecting Table 62 shows a large degree of similarity for the Node BSI. The DFM and CAFTA results had the same 5 nodes with the highest BSI. It was seen in the CAFTA results that the BSI were consistently very high, while with DFM, the values dropped off significantly after the first 2 nodes. When considering the BSI of the states shown in Table 63 and Table 64, there are some larger differences that are seen. The DFM results rank the SEU (T) state the highest, and the SHE_1 (P) state the second highest, while the order was reversed with the FTA results.

Both methods include the SHE (P) node, however DFM includes both failure states ("1" and "0"), while the FTA results contain only the "0" state. It may seem unexpected that the SHE "1" state is so heavily represented, however BSI does not take into account probability, and in actuality that state does not contribute as much to the Top Event probability. The SEGR and SEDB failure states are shown in fifth place, with equal BSI for DFM, however CAFTA ranks the SM and RE failure modes as fourth and fifth, respectively. The "N/A" values in the table are there as there was only one failure state present in the "Clock 0" run, due to the small number of MCS.

Considering the "Spurious Trip" Top Event, more differences are seen with the Node BSI, than in the previous case. Both methods have the failure modes "SEU T", "CE (D)" and "SHE (P)", ranked as the top three, respectively. The difference is seen looking at the next node, where "SED" is ranked next by DFM, however with FTA the nodes "SEGR" and "SEDB" are ranked next. As in the case of the "Missed Trip", the BSI values drop off more quickly with DFM.

Examining the state BSI values, it is again seen that DFM ranks "SEU (T)" above "CE (D)", while it is the opposite in FTA. Both methods have states from the "SHE (P)" node, however DFM shows more of the "0" state (by BSI only), while FTA shows more of the "1" state. The last two states by DFM are "SED", and a second "SHE" (P) state (whereas only one states was ranked in CAFTA), while FTA ranked SEB and RE as fourth and fifth. As evidenced by the information in this section, there are several similarities and in the BSI measures between the two methods.

Table 65: Node BSI Comparison for "Spurious Trip" Top Event

Node	DFM	FTA (Clock 0)	FTA (Clock 1)
SEU (T)	0.981	0.996	0.999
CE (D)	0.825	0.996	0.999
SHE (P)	0.698	0.996	0.998
SED (Q) (DFM)	0.667	0.98	0.997
SEGR (FTA)	0.475	0.996	0.998
SEDB (FTA)	0.475	0.996	0.998

Table 66: DFM State BSI Comparison for "Spurious Trip" Top Event

State	BI
SEU (T)	0.603
SHE_0 (P)	0.286
CE (D)	0.222
SED	0.095
SHE_1 (P)	0.095

Table 67: DFM State BSI Comparison for "Spurious Trip" Top Event

State	Clock 0	State	Clock 1
SHE_1 (P)	0.579	CE (D)	0.479
CE (D)	0.318	SEU (T)	0.378
RE	0.115	SHE_1 (P)	0.135
SHE_0 (P)	0.115	SEB	0.116
SED	0.115	RE	0.114

4.3.11.3. Discussion on Possible Reasons for DFM/FTA Differences

From the results in sub-sections 4.3.10 and 4.3.11, it is evident that there are some differences between the DFM and FTA methods. This sub-section will discuss some potential reasons for those differences, which could include the choice of initial conditions, number of set time steps, and the truncation cut-off value.

Initial Conditions

DFM models are often run using initial conditions, for reasons such as reducing computational time, avoiding impossible PI, or to place the system in a normal operating state before the analysis. The latter was done in this analysis, as it was assumed that the system was operating normally, when it is interrupted by the effects of the failure mode(s). The initial conditions (or initial probabilities) would force the analysis to accept/reject certain nodes and states, reducing the total number of PIs, and eliminating certain states from the analysis.

Additionally, any nodes that are the outputs of transition boxes need to have nodes defined at the initial time step (either through initial conditions or set probabilities), or else the Dymonda analysis will consider all possible states in that node. This is because of the time delay in the transition box, so it is not able to pass on the value (state) to the node at the initial time, unlike with normal transfer boxes. In order to ensure that the system was operating normally, the output nodes of the transition boxes (registers) were set using initial conditions. As only one time step was considered, the calculated values (states) may not have been passed through the transition box, leaving only the states set by the initial conditions to be considered in the final analysis. Additional time steps may allow for the calculated states to be passed through the system, however attempts to do that faced other issues, as discussed below. Therefore, the use of different initial conditions could affect the number of PIs, as well as which nodes and states are included or excluded.

Time Steps

Dymonda allows the user to specify the number of time steps in the system. A greater number of time steps allow one to analyze how the system evolves through time, however the more time steps used can drastically increase the computational time (known as “state explosion”). State explosion (also referred to as the “combinatorial explosion of states”) occurs when the state space of a system grows exponentially with the number of variables (number of nodes and states per node in the case of DFM), and the interactions of the system components (amount and complexity of transition/transfer boxes in the DFM model). The use of multiple time steps can grow the state space even more, further exacerbating the “state explosion” issue, as the critical transition table will expand based on dynamic node/state/time step combinations.

To avoid that issue, the test system models were run with one time step. More time steps would generally produce more PIs, potentially matching the number of MCS returned by CAFTA. It is also possible that using one time step does not allow for a complete description of the test system, based on the way the current model is built. The system contains multiple registers and feedback loops, so it may take multiple time steps to completely describe the system, which may produce additional PI.

Attempts were made to run the test system with two time steps, however those runs were unsuccessful. The analysis would not finish (on the computer that Dymonda was installed on), even over a period of 48-72 hours, including cases when the program crashed. From these attempts, it appears that the test system may have been too large for Dymonda to analyze for more than one time step.

Retention

The CAFTA Cut Set Generator generally retains many more states than the Dymonda analysis. States such as the constants in the OT/Filter block were set to have a probability of 1, in order to ensure that only the failure modes contributed to the Top Event. CAFTA includes many more states in the MCS results than Dymonda does in the PI results, which leads to a larger number of MCS, with lower probabilities. This was seen in Table 59 of Sub-section 4.3.11.1, where the Dymonda results only consist of 4 entries, whereas the CAFTA result contains many more entries, and breaks down that PI into several MCS, each with much lower probabilities. This could partly explain why CAFTA returns many more MCS than Dymonda does PI, and yet the “DPC” probabilities are lower than the DFM probabilities.

Circular Logic

Static FTA does not explicitly allow for circular logic to be created in the fault tree (CAFTA includes a “Circular Logic Check”, which will return an error and disallow the analysis from being performed), however DFM does allow for it (with time lags). Therefore, the logic loops had to be manually broken, in order for those events to be included in the fault tree. Besides the registers, which also create time dependency in the system, there are two large circular logic loops in the test system. The first is in the SC block, where a (correct) ADC/SC output is fed back into a register for possible use in future time steps

in case the Sanity check is failed. The second is in the filter part of the OT block, where the filter output is fed back into a register, and is then added to the new input value to produce the filter output in the next time step. Since static FTA cannot specifically model the logic loops, it is another possible reason for the differences

Truncation Value

The truncation value eliminates the MCS/PI with a probability below that value. This would have eliminated all the MCS/PI below the 1.0E-12 value. It was seen that the most represented nodes/states were those in the Comparator subsystem (the last block in the block diagram), so it is possible that failure modes in the Overtemperature and/or Filter sections were truncated out more that the failure modes in the Comparator block. It is also possible that, due to the different methods used to find and calculate the PI/MCS probabilities, the truncation limit eliminated more of the PI from the DFM results, than MCS were eliminated from the CAFTA results.

Computational Methods

A key issue may the algorithms used by the software tools. Dymonda uses the “method of generalized consensus” on a “critical transition” table, in order to determine the Prime Implicants [93]. On the other hand, there are several different algorithms that can be used to find the MCS from Fault Trees, including MOCUS, PREP, ELREFT, etc. [85]. In complex, dynamic systems, the algorithms used by the tools could return different results.

4.3.11.4. Overall Difference

Sub-sections 4.3.11.3 discussed some of the potential reasons for the differences in the results. At this stage of the research work, it is difficult to pinpoint the exact reasons for these differences. The test system was large (numerous nodes/states), which necessitated the use of initial conditions and

prevented the system from running for multiple time steps. From the preliminary analysis the, major contributors appear to be:

1.) Initial conditions: These will enforce restrictions on the system analysis.
2.) Time steps: FTA cannot include time steps, while DFM will propagate the analysis for time steps "-1" to "0".
3.) Circular Logic: FTA cannot include the control loops that DFM does.
4.) Algorithms: Overall algorithms and computational implementations are different.

Future work involving additional time steps and different initial conditions needs to be performed, in order to confirm the main contribution(s) to these differences.

4.3.12. Conclusions from the Preliminary DFM/FTA Comparisons

This part of the research program provided a preliminary comparison between DFM and FTA for the purpose of reliability modelling of FPGA-based I&C systems. It considered the Top Event probabilities, MCS/PI and Birnbaum Structural Importance measures. Examining the test system, it was found that there were some similarities in the probabilities, PI/MCS, and BSI measures, however several differences were seen as well. Several potential reasons for the differences are also discussed, such as the use of initial conditions, time steps, and truncation values. Certain issues with the different methodologies were demonstrated during this research. A potential issue with FTA is that static Fault Trees cannot handle the oscillating clock signals leading to impossible MCS being generated, however DFM is able to model the oscillating clock. With regards to DFM, it was seen that it may not be able to solve the large model in a reasonable time without the use of initial conditions, which could affect the accuracy of the results. FTA could compute the Top Event quickly, even for the full system model.

Overall, this leads to two important topics for advanced comparisons. The first, is a more in-depth look at the underlying algorithms behind DFM and FTA. The second, being DFM/FTA comparisons for a test system that is run more than one time step. These two aspects are discussed in sub-section 4.4.

4.4. Advanced DFM and FTA Comparisons

The comparisons between DFM and FTA made in sub-section 4.3 discussed some of the potential reasons for the differences in the analysis results returned by those two methodologies. This section delves deeper into one of those reasons, namely the way the DFM and FTA algorithms handle the MVL when computing the MCS/PIs. When considering a small system with only one time step, it was seen that their results were very similar, however there was a noticeable change when multiple time steps were used. Additional comparisons will include modelling a smaller FPGA-test system for multiple steps with DFM, to provide more insight into the effect of the dynamic behaviour on the differences between DFM and FTA results.

4.4.1. Theoretical DFM and FTA Comparisons

In order to properly assess the reasons for the differences in the DFM and FTA results obtained in in previous research, a detailed comparison of the underlying theory is discussed in this section. This involved applying DFM and FTA to simplified test systems, and comparing the analysis process and results.

4.4.1.1. Static Comparisons

In regards to the "static" system comparisons, these involve DFM analyses for one time step (TS = -1). Although they are not technically static, as some amount of time lag is included in the system, when a single time step is used in a simple system it was seen that DFM and FTA will produce very similar results [208]. Therefore, the test runs considered in this section can be approximated as static systems by FTA.

4.4.1.2. Simplified Feed Water System Model

Figure 86 represents a DFM model of a simplified section of the Feed Water system discussed in the literature [123,124]. In this example, the value of "Water Level Measurement" (WLM) at the current time step is a function of its value at the previous time step, as well as the value of the "Water Level" (WL) and "Measurement Failure" (MF) inputs. In this example, all of the time delays were set equal to 1

time step. The system seen in Figure 86 includes time delays, making it a dynamic system, and not a static one. However, an analysis of one time step for this simple system is straightforward to implement in FTA. The three nodes are discretized as seen in Table 68, with the corresponding decision table for the "WLM = 1" TE given in Table 69 [123].

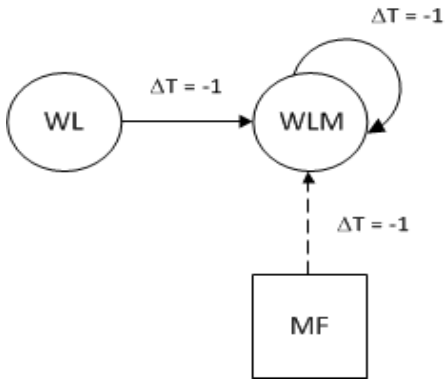


Figure 86: Simplified Water Level Measurement System

Table 68: Simplified Feed Water System Node Discretization

Node	State(s)		
WLM	-1 (Low)	0 (Normal)	1 (High)
WL	-1 (Low)	0 (Normal)	1 (High)
MF	N/A	0 (Works)	1 (Fails)

Table 69: Decision Table for "WLM = 1" TE

Row	**Inputs**			**Outputs**
	MF	WLM	WL	WLM
1	0	-1	1	1
2	0	0	1	1
3	0	1	1	1
4	1	1	-1	1
5	1	1	0	1
6	1	1	1	1

The six rows in this decision table would entail the Implicants, that cause the TE ("WLM" = "1"). However, those are not necessarily the Prime Implicants. The same six Implicants are implemented exactly in a fault tree using the CAFTA package, as seen in Figure 87.

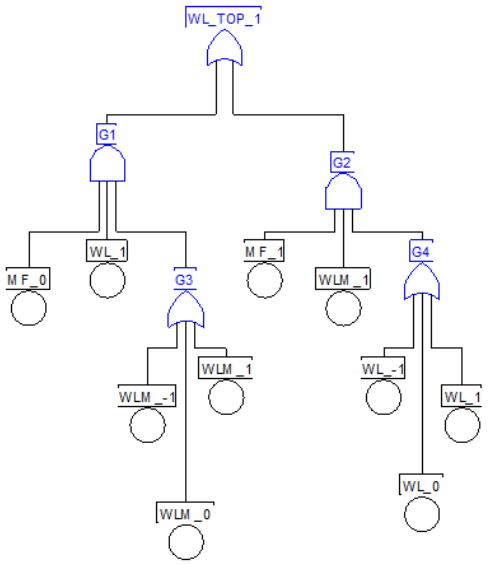


Figure 87: Fault Tree for Simplified Feed Water System

4.4.1.3. Simplified Feed Water System DFM PIs

In DFM, the PIs are obtained through the decision table in Table 68 using the logical reduction/absorption operations discussed in Refs [21,117]. As this is a simple system, the "critical transition table" is the same as the decision table seen in Table 69. The probabilities in Table 70 were applied to the test system, to allow for quantitative analysis using both DFM and FTA [123]. These probabilities are per time unit (time step), as per the reference.

Table 70: Feed Water Test System Probabilities

Node	State/Probability		
WLM	-1 (0.33)	0 (0.34)	1 (0.33)
WL	-1 (0.33)	0 (0.34)	1 (0.33)
MF	N/A	0 (0.95)	1 (0.05)

The process for the logical reduction needed to produce the PIs from the decision table is seen as follows.

1. Perform a “Merge” operation on Rows 4-6
2. Perform a “Merge” operation on Rows 1-3

These two operations will produce two PIs, as shown in Table 71, however there is a third PI, generated as the “consensus” term from the first two PIs. It can be seen that is term could be generated from the first two PIs. It should be noted that the “-“ represents the “Don’t Care” value.

Table 71: DFM PIs for the Simple Feed Water System

Row	Inputs			Outputs
	MF	WLM	WL	WLM
1	0	-	1	1
2	1	1	-	1
3	-	1	1	1

As this is a simple system, the IB and CB are the same, however that is generally not the case. While the complete base will be unique, it is possible that different logic reduction operations can be applied, and different consensus terms can be generated. A second, more complex approach, still starting from Table 68, would be:

5. Merge rows 3 and 6

Table 72: Critical Transition Table after Merging Rows 3 and 6

Row	Inputs			Outputs
	MF	WLM	WL	WLM
1	0	-1	1	1
2	0	0	1	1
3	-	1	1	1
4	1	1	-1	1
5	1	1	0	1

6. Reduction-Merge (new) Rows 1-3

Table 73: Critical Transition Table after Reduction-Merging (new) Rows 1-3

Row	Inputs			Outputs
	MF	WLM	WL	WLM
1	0	-	1	1
2	-	1	1	1
3	1	1	-1	1
4	1	1	0	1

7. Reduction-Merge (new) Rows 2-4

Table 74: Critical Transition Table after Reduction-Merging (new) Rows 2-4

Row	Inputs			Outputs
	MF	WLM	WL	WLM
1	0	-	1	1
2	-	1	1	1
3	1	1	-	1

In this case, the same three PIs are found. However, using this method it was not necessary to generate a separate consensus term, as all three PIs are identified using the logical reduction operations. This would not generally be the case though, as this was a simple test system.

Table 75: Simple Feed Water Tank DFM PI Probabilities

PI #	PI		PI Prob.
1	MF_0	WL_1	0.3135
2	WL_1	WLM_1	0.1089
3	MF_1	WLM_1	0.0165

Considering the quantitative values from Table 69, the PI and Top Event probabilities (per unit time) determined using DFM are given in Table 75 and Table 76, respectively. In terms of the "Exact" probability, it is the sum of PI "1" and PI "3", as those are the two MEIs in the results.

Table 76: Simple Feed Water Tank DFM Top Event Probabilities

Method	Sum	MCSUB	Exact
Probability	4.39E-01	3.98E-01	3.33E-01

4.4.1.4. Simplified Feed Water System FTA MCS/PI

In the case of FTA, the Fault Tree shown in Figure 87 consists only of "AND" and "OR" gates, allowing traditional methods to solve for the MCS (PIs) to be a used. Applying the "MOCUS" algorithm outlined in Sub-section 2.3.1.5 yields the calculation shown in Table 77:

Table 77: Simple Feedwater Tank MCS Determination via MOCUS Algorithm

Steps			
1	**2**	**3**	**4**
Top	G1	MF_0, WL_1, G3	MF_0, WL_1, WLM_-1 MF_0, WL_1, WLM_0 MF_0, WL_1, WLM_1
	G2	MF_1, WLM_1, G4	MF_1, WLM_1, WL_-1 MF_1, WLM_1, WL_0 MF_1, WLM_1, WL_1

In Table 77, "Step 4" showcases the MCS (PIs). When the probabilities in Table 76 are applied, the MCS (PIs) are seen in Table 78, with the corresponding Top Event probabilities seen in Table 79.

Table 78: Simple Feed Water Tank FTA PI Probabilities

PI #	PI/MCS			PI Prob.
1	MF_0	WL_1	WLM_-1	1.03E-01
2	MF_0	WL_1	WLM_0	1.07E-01
3	MF_0	WL_1	WLM_1	1.03E-01
4	MF_1	WLM_1	WL_-1	5.45E-03
5	MF_1	WLM_1	WL_0	5.61E-03
6	MF_1	WLM_1	WL_1	5.45E-03

Table 79: Simple Feed Water Tank FTA Top Event Probabilities

Method	Sum	MCSUB	Exact
Probability	3.30E-01	2.94E-01	2.271E-01

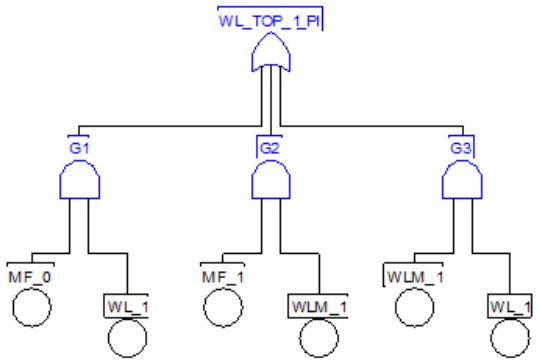


Figure 88: Fault Tree for Simplified Feed Water System (PIs Only)

For a simple system, such as the test system seen in Figure 86, one could easily identify two PIs through inspection, as the "WLM" state is not needed for PI 1, (Table 75) and the "WL" state is not needed for PI 2 (Table 75). However, that would still leave PI 3 (Table 75), which needs to be identified using the "Consensus Law". In the case of the simplified Feed Water system, it is relatively easy to determine the PIs by inspection/consensus law from the decision table seen in Table 69 and to create the appropriate fault tree. A fault tree showing only the 3 PIs is seen in Figure 88. Analyzing that fault tree results in the same PIs, PI Probabilities and Top Event Probabilities as the DFM analysis discussed in sub-section 4.4.1.3.

4.4.1.5. Simplified Feed Water System BDD MCS/PI

The fault tree from Figure 89 can be converted into a BDD, using the process outlined in Sub-section 4.2. The variable ordering was taken as WL_1 < MF_0 < WLM_1 < MF_1 < WLM_0 < WL_0 < WLM_-1 < WL < -1. This resulted in the same six MCS/PIs being returned as seen with the MOCUS/CAFTA method, as well as the same Top Event probability.

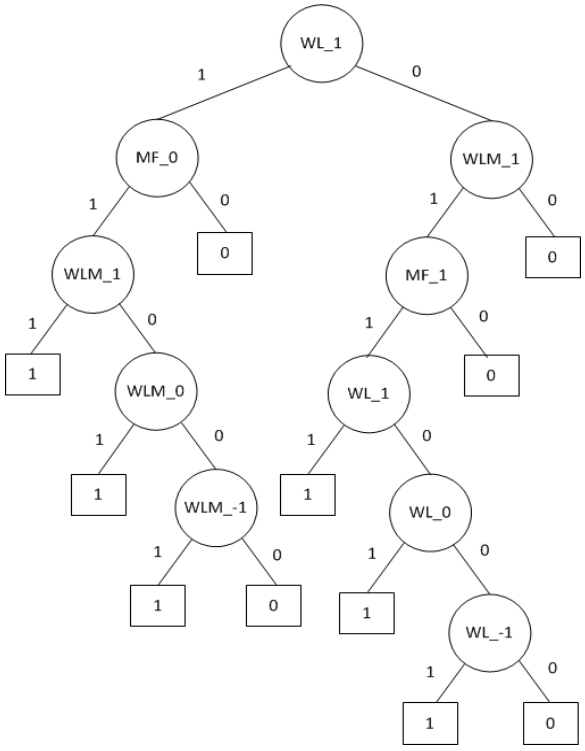


Figure 89: SFBDD for TS 1 Fault Tree

4.4.1.6. Non-Coherent Fault Tree Attempts

The fault trees seen in sub-sections 4.4.1.1-4.4.1.4 attempted to use MVL logic, as DFM would. However, those fault trees only used "AND" and "OR" gates, so the fault trees were constructed and analyzed using coherent methods. As these fault trees are intended to represent MVL (non-coherent) logic, they are re-constructed in this sub-section, using non-coherent logic ("NOT" gates).

"MF_1" Complement (MF_1 C)

Consider the fault tree shown in Figure 3. The Basic Event "MF_1" would be represented as $\overline{MF\_0}$, in binary logic. Therefore, the fault tree was amended to replace the "MF_1" with a "NOT" gate, and the "MF_0" Basic Event. It should be noted that the choice of making "MF_1" or "MF_0" the complement is arbitrary in this case, as in MVL neither term would represent the "normal" or "complement" variable.

To accomplish this using BDDs, a change is made to the BDD in Figure 90. Following the procedure discussed in sub-section 2.3.1.6, $\overline{MF\_0} = ITE(MF\_0, 0, 1)$. This leads to the following substitution for the "MF_1" node in Figure 90.

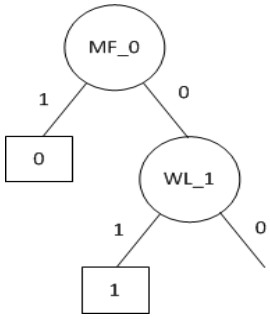


Figure 90: Switching "MF_1" with "Complement MF_1" in the Figure 89 BDD

Now, "0" branch from the "MF_0" node terminates at a "1" vertex. This represents the fact that the $\overline{MF\_0}$ term is included in the Prime Implicants. Additionally, when searching the BDD for PIS, the "0" branch for all nodes must be considered. For this example, the tree is searched using the "Bottom-Up" method for determining the PIs from non-coherent fault trees.

All Complement (All C)

The "WL" and "WLM" nodes are represented by all three of their respective states, in the fault tree in Figure 3. In this fault tree, all of the MVL states have been replaced by binary logic states, which would make it easier to apply standard non-coherent FTA methods. The new fault tree is found in Figure 91. However, this raises other issues. First, in the case of MVL, the complement of "WLM_0" is actually "WLM_-1" + "WLM_1", as there are three states to that node. Therefore, $\overline{WLM\_0}$ actually represents two states, which is not fully captured in binary logic. The second issue is that elements of $\overline{WLM\_0}\ and\ \overline{WL\_0}$ appear elsewhere in the fault tree, in the form of "WLM_1" and "WL_1", respectively. This is not immediately apparent when using binary logic, and as such could present a problem when using standard non-coherent FTA methods. One point it its favour though, as seen in Figure 91, is the that it is clear that logic gates "G3" and "G4' are not involved in the PIs. For example, it is obvious from inspection that $WL\,M\_0\ "OR"\ \overline{WLM\_0} = 1$. This was not as obvious when MVL was considered, and may become less obvious in more complex fault trees.

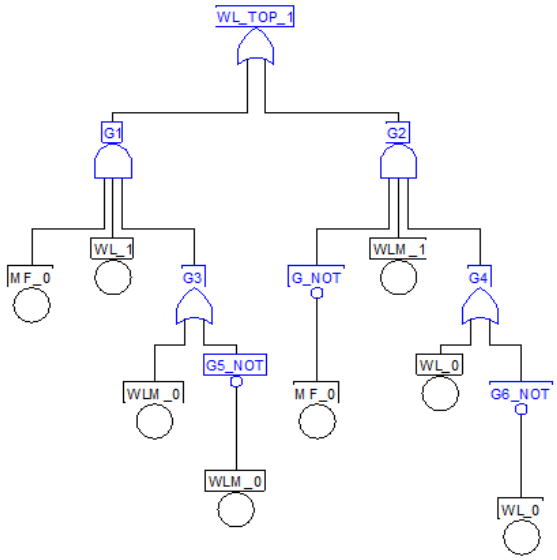


Figure 91: Feed Water Fault Tree with "MF", "WL" and "WLM" Complements

The corresponding BDD, using the variable ordering MF_0 < WL_1 < WLM_1 < WL_0 < WLM_0 is seen in Figure 92. In the BDD shown in Figure 92, it may seem strange to have nodes such as "WLM_0" connected to itself. This is due to the attempt to represent the MVL using non-coherent, binary logic. In this case, it represents that both "WLM_0" and its complement are in fault tree, and could be present in the PIs.

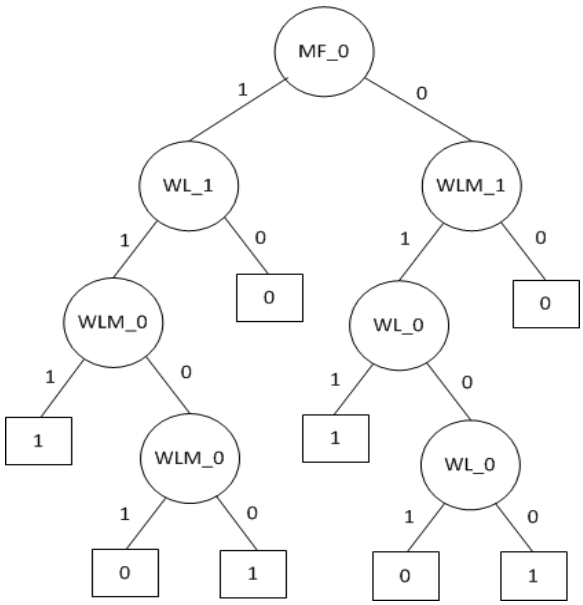


Figure 92: BDD for Fault Tree with "MF", "WL" and "WLM" Complements

Coherent Approximation (CA)

As discussed in sub-section 2.3.2.6, the coherent approximation can be applied to non-coherent fault trees. Considering an MVL system, this raises the question on which of the non-coherent Basic Events to delete from the fault tree. One could simply remove the "MF_1" Basic Event, as it can be considered the complement of the "MF_0" Basic Event. However, all of the "WL" and "WLM" states also represent MVL, so one must determine which Basic Event, if any, should be deleted. Considering, Figure 91, the $\overline{WL\_0}$ and $\overline{WLM\_0}$ terms could be removed as well, but that would significantly alter the Fault Tree. Therefore, only the coherent approximation with the "MF_1" Basic Event deleted was considered.

4.4.1.7. Non-Coherent Fault Tree Results

The results for the non-coherent FTA methods are presented here.

"MF_1" Complement

Applying the MOCUS method (and confirming the results with CAFTA), produced the same MCS/PIs seen in Table 78, except with "$\mathrm{MF\_1}$" being replaced by "$\overline{MF\_0}$". In this case, the non-coherent logic did not have to be "Pushed Down" the fault tree, as the negated logic was already at the bottom of the tree. Without further simplification, these overall results would be unchanged. However, consider the selected Implicants seen in Table 80:

Table 80: Select "MF_1 C" Implicants

Row	PI/MCS			PI Prob.
1	MF_0	WL_1	WLM_1	1.03E-01
2	$\overline{\mathrm{MF\_0}}$	WL_1	WLM_1	1.07E-01

As seen in Table 80, the same implicant will occur, regardless of the state of the "MF" node. This is more readily seen, when binary logic is used. Manually, this Implicant can be simplified, to produce the PI seen as "PI 3" in Table 75.This produces one of the PIs, and the remaining four Implicants are unchanged. The same effect was seen when analyzing the non-coherent fault tree using BDDs.

All Complement

Applying the same process to the fault tree from Figure 91 results in the MCS/PIs seen in Table 81

Table 81: "ALL C" Implicants

Row	PI/MCS			PI Prob.
1	MF_0	$\overline{\text{WLM\_0}}$	WL_1	2.07E-01
2	$\overline{\text{MF\_0}}$	$\overline{\text{WL\_0}}$	WLM_1	1.09E-01
3	$\overline{\text{MF\_0}}$	WL_0	WLM_1	5.63E-03
4	MF_0	WL_1	WLM_0	1.07E-01

It is seen that the FTA methods return fewer Implicants when MVL logic is replaced with binary logic. Here, it is seen that further simplifications can still be made. The Implicants in Rows 1 and 4 contain the states "WL_1" and "MF_0", along with "WLM" node in both of its binary states. Simplifying this results gives "PI 1" from Table 75. Similarly, Rows 2 and 3 in Table 81 contain the state "WLM_1", "$\overline{\text{MF\_0}}$", both states of the binary "WL" node. This simplifies to "PI 2" of Table 75, as "$\overline{\text{MF\_0}}$" = "MF_1". That simplification results in 2 PIs, and the "Consensus Law" is then applied, to locate the third PI ("PI 3" in Table 75). These PIs are also determined by searching the BDD in Figure 92 using the method discussed in sub-section 2.3.1, and discarding any Implicants that are not Prime Implicants.

Therefore, applying non-coherent logic to the fault tree can uncover some of the PIs when analyzing the system manually, or if the specific software tool is able to account for these binary logic reduction operations.

Coherent Approximation

Considering the coherent approximation, it is seen that it produces five total Implicants, the first one of which is a PI. However, Implicants "2" and "3" are not actually Implicants or PIs (the approximation results in incorrect MCS/PIs being returned), and the remaining two are just Implicants. In this case, the results from the "Coherent Approximation" appear to be significantly different from the actual results.

Table 82: Coherent Approximation Implicants

PI #	PI/MCS			PI Prob.
1	WLM_1	WL_1	N/A	1.039E-01
2	WLM_1	WL_0	N/A	1.12E-01
3	WLM_1	WL_-1	N/A	1.09E-01

4	MF_0	WLM_0	WL_1	1.07E-01
5	MF_0	WLM_-1	WL_1	1.03E-01

The overall Top Event probabilities (with and without additional simplification) for the non-coherent FTA methods, as well as the DFM results (for easier comparison) are seen in Table 83.

Table 83: Non-Coherent FTA Top Event Comparison

Non-Coherent Method	Quantitative Calculation Method (Top Event Probabilities)		
	Sum	**MCSUB**	**Exact**
MF_1 C	3.30E-01	2.94E-01	2.322E-01
MF_1 C (Simplified)	3.30E-01	2.94E-01	2.322E-01
All_C	3.31E-01	3.03E--01	3.300E-01
All_C (Simplified)	4.39E-01	3.98E-01	3.33E-01
CA	5.40E-01	4.35E-01	3.494E-01
DFM	4.39E-01	3.98E-01	3.33E-01

It was seen in all (un-simplified) cases, that more Implicants were produced than with DFM, however as in the case of the coherent fault trees, many of them were not Prime Implicants. Using the "All C" and "CA" methods produced fewer Implicants than coherent methods, and simplifying (reducing) the "MF_1 C" and "All C" results returned actual Prime Implicants.

Quantitatively, it is seen that the "CA" method overestimated the Top Event probabilities, while the other methods tend to underestimate the Top Event probabilities, as they are still identifying Implicants, and not just PIs. However, simplifying the "All C" method results in the exact PIs, and as such returns the same probabilities as DFM.

4.4.1.8. Static Results Comparisons

Qualitative Comparisons

There are several noticeable differences seen when comparing the DFM and FTA (MOCUS and BDD) results. The first is the lack of logical reduction performed by the fault tree software. This results in the six PI/MCS seen in Table 78. In reality, these are not all Prime Implicants, as it is clear by inspection that

PIs 1-3 and PIs 4-6 could be further simplified, to create 2 actual Prime Implicants. In fact, the six entries in Table 78 are actually Implicants, and not Prime Implicants. However, the fault tree analysis by itself cannot determine that three "WLM" or "WL" states make up the entirety of their prospective nodes, so conventional fault tree analysis methods cannot logically reduce those Implicants into Prime Implicants. Therefore, it was seen that FTA produces extra Implicants than DFM, but they are not Prime Implicants. In the case of the non-coherent FTA methods, a similar issue is seen, although not to the same degree. Generally, applying non-coherent binary methods to the MVL logic produced fewer Implicants than the coherent fault trees, and in some cases were able to produce the actual PIs. Additionally, it was seen that the non-coherent FTA results could be more easily reduced to locate the PIs.

A second difference is the "missed" PI, seen in Row 3 of Table 74. The DFM analysis produces that "Consensus Term", as it employs the "Method of Generalized Consensus". However, it is seen than traditional FTA methods, such as MOCUS will not apply the "Consensus Law" to generate the additional PI.

This becomes especially tricky, as in MVL variables with two states are not always defined by their "normal" and "complement" form, as they are in binary logic. As example, the "MF" node was discretized into two states, "0" (No Failure" and "1" (Failure). In binary logic used by fault trees, this would be represented as "MF" and "$\overline{\mathrm{MF}}$". Strictly speaking, if the "MF" node was defined as "0" and "1" as done with MVL/DFM, it would not register as "NOT" logic, so the "Consensus Law" may not be applied. Another issue is the software tool itself. Not every package supports negated logic, and even if the tool does, it may not identify all Prime Implicants. As an example, the CAFTA tool used in this paper supports the use of "NOT" logic, however it does not apply the "Consensus Law" to the MCS (PIs) returned in the analysis.

In the case where the "Consensus Law" is applied to the FTA results, there is the issue of the MVL nodes, in this case "WL" and "WLM". When there are three states in each node, the "NOT" logic becomes more complicated (i.e .$\overline{WL\_1} = WL\_ - 1 + WL\_0$). This makes applying the "Consensus Law" more difficult with binary fault tree logic, and could lead to additional "missed" PIs in larger, more complex systems.

A potential issue also arises from the use of the "All C" attempt, is that the same Basic Event is actually represented multiple times, in different ways. For example, in Figure 91, the states "WL_1" and "$\overline{\mathrm{WL\_0}}$" are both represented, however "WL_1" is realty a subset of ""$\overline{\mathrm{WL\_0}}$", (as is "WL_-1"). In this simple test case, it did not cause any issue, however thus may not be the case in a larger, more complex system.

Quantitative Comparisons

In terms of the quantitative comparisons, the differences are actually caused by the differences in the PIs found between the methods (qualitative differences), not the quantitative calculations themselves. The same general methods are applied to calculate the probability of the MCS/PIs, as well as the Top Event probabilities, in FTA and DFM, as discussed in sub-section 2.3.1. It was seen in sub-section 4.4.1.1 that when the fault tree was amended to include only the PIs, the MCS/PI and Top Event probabilities were the same.

Regarding the differences in the MCS/PI probabilities, it is seen that there is a large range in probabilities with the FTA results, as it only returns the Implicants, and not Prime Implicants. As these Implicants contain additional literals (three literals in each FTA PI as opposed to two literals in each DFM PI), the probability of each of the Implicants returned by FTA is lower than that of each PI returned by DFM. The differences in Top Event probabilities were in part caused by the differences in the PI probabilities, as well as the additional PI returned by DFM. This led to the FTA Top Event probability being noticeably lower than that of the DFM Top Event probability. Similar Issues were seen regarding the non-coherent FTA methods, as they generally underestimated the Top Event probability, (except in the case of the Coherent Approximation). The only other difference being that the "All C" non-coherent approach results could be reduced to determine the exact PIs, and therefore had the same Top Event probabilities.

4.4.1.9. Dynamic MVL Comparisons

In this sub-section, multiple time steps will be considered. This also allowed for the inclusion of the effects of dynamic consistency rules, as their effects may not be noticeable in static systems or systems with one time step. The same test system seen in Sub-section 4.4.1.1 was used, however this time it was run for two time steps. Additional runs were performed, to demonstrate the effects of dynamic consistency rules on the PIs and Top Events. In should be noted that the probabilities for these tests were the same as those given in Table 68.

4.4.1.10. Dynamic Test System DFM PIs

In DFM, once the model is built, the analysis can be run for an arbitrary number of time steps. Considering the test system from Figure 86, after the first time step, the "critical transition table" will be the same as the one shown in Table 70. After the second time step, the new "critical transition table" (after logical reductions) is given in Table 84. The corresponding PIs and PI probabilities are seen in Table 85. As seen in Table 84, the same node may appear several times in the results for different time steps, where the time step is shown in parentheses below the node name.

To produce Table 84, the "WLM" node at "TS = -1" is expanded in the analysis during the second time step, which is the reason that the node "WLM" does not appear in Table 10 at "TS = -1". This expansion involves substituting the 3 PIs from Table 71, into the WLM (-1) = "1" term which produces the three terms that appear at "TS = -2". The expansion and corresponding logic reduction produces the set of PIs and PI probabilities given in Table 85. The "critical transition table" quickly becomes too large to show all logical reduction steps, as fully expanded, Table 84 contains 108 rows. With larger, more complicated systems, this "state explosion" issue may limit the size of system that can be analyzed in a reasonable amount of time.

Table 84: DFM PIs for the Simple Feed Water System (TS = 2)

Row	**Inputs**					**Outputs**
	MF (-1)	MF (-2)	WL (-1)	WLM (-2)	WL (-2)	WLM
1	-	-	1	1	1	1
2	1	-	-	1	1	1
3	0	-	1	1	-	1
4	-	1	1	1	-	1
5	-	0	1	-	1	1
6	1	1	-	1	-	1
7	1	0	-	-	1	1

Table 85: Simple Feed Water Tank DFM PI Probabilities (TS = 2)

PI	**PI**			**PI Prob.**
1	MF_0 (-1)	WL_1 (-1)		3.14E-01
2	MF_0 (-2)	WL_1 (-2)	WL_1 (-1)	1.03E-01
3	WL_1 (-2)	WLM_1 (-2)	WL_1 (-1)	3.59E-02
4	MF_0 (-2)	WL_1 (-2)	MF_1 (-1)	1.57E-02
5	WL_1 (-2)	WLM_1 (-2)	MF_1 (-1)	5.45E-03
6	MF_1 (-2)	WLM_1 (-2)	WL_1 (-1)	5.45E-03
7	MF_1 (-2)	WLM_1 (-2)	MF_1(-1)	8.25E-04

The Top Event probabilities, based on the DFM analysis are present in Table 16. The "Exact" value is the sum of the two MEIs, namely "PI 1" and "PI 4".

Table 86: Simple Feed Water Tank DFM Top Event Probabilities (TS = 2)

Method	**Sum**	**MCSUB**	**Exact**
Model	**Probability**		
TS = 2	4.80E-01	4.23E-01	3.30E-01
Sink	4.59E-01	4.16E-01	3.27E-01
DCR	1.50E-02	1.49E-02	9.56E-03

Sink States

Different dynamic consistency rules are applicable to DFM analyses. In the model shown in Figure 86, the "MF = 1" failure state was considered to be a sink state in the original literature sources [123,124]. This was not enforced previously, however now that sink state is enforced, to determine what the effect on the analysis results will be. This sink state means that the "MF" node will not be able to transition from state "1" to state "0" (i.e. if the node transitions into state "1", it must remain there for the rest of the analysis). This will alter the PIs seen in Tables 84 and 85. The use of the sink state and other dynamic consistency rules would not have an effect on the system run for one time step, as that duration did not allow for the individual nodes to transition between different states. When the sink state is applied to the "MF = 1" node/state, and the model is run for two time steps, the new PIs are given in Table 87, with the Top Event probabilities seen in the row labelled "Sink" in Table 86.

Table 87: Simple Feed Water Tank DFM PI Probabilities (TS=2, Sink State MF = 1)

PI	**PI**			**PI Prob.**
1	MF_0 (-2)	MF_0 (-1)	WL_1 (-1)	2.98E-01
2	MF_0 (-2)	WL_1 (-2)	WL_1 (-1)	1.03E-01
3	WL_1 (-2)	WLM_1 (-2)	WL_1 (-1)	3.59E-02
4	MF_1 (-2)	WLM_1 (-2)		1.65E-02
5	MF_0 (-2)	WL_1 (-2)	MF_1 (-1)	1.57E-02
6	WL_1 (-2)	WLM_1 (-2)	MF_1 (-1)	5.45E-03

In this instance, there are 6 total PIs, as opposed to 7 that were produced without the sink state. The reason for this seen when comparing PI 4 from Table 87 with PI 6 and PI 7 of Table 85.When considering

PI 7 of Table 85, the "MF = 1" state appears in both time steps, indicating that that failure state must occur twice. However, once "MF = 1" is designated as a sink state, a failure at "TS = -2" would become permanent, so that node could not fail again. As the literal "MF = 1 at TS = -1" is not needed, PI 7 from Table 85 becomes PI 4 from Table 17. This has a secondary effect, when PI 6 (Table 15) is considered. It is seen that Pi 6 (Table 85) and PI 4 (Table 87) contain the same two literals, with PI 6 containing an additional literal. The inclusion of the sink state makes that third literal unnecessary, and in turn has turned PI 6 (Table 85) into an Implicant that is not a PI. Therefore, it is removed during the analysis.

A third difference is seen when inspecting PI 1 from Tables 85 and 87. In the analysis seen in Table 85, it did not matter what state the "MF" node was in at "TS = -2" (as denoted by the "-") entry in Table 14. However, once "MF = 1" Is designated a sink state, then the "MF" node must be in state "0" at "TS = -2" for that PI to occur, as a "0" → "1" transition is not allowed.

Dynamic Consistency Rules (DCR)

For this test, the "WL" node was assigned a DCR for "strictly decreasing", and the model was again run for two time steps. This entails that the state of the Water Level ("WL") node could only decrease as the analysis progressed through time. Any PI that included either an increase or no change in the "WL" value would be excluded. It should be noted that "MF = 1" state was not designated as a sink state, for this analysis. The PIs and PI probabilities are given in Table 88, while the Top Event probabilities are given in the row labelled "DCR" in Table 86.

Table 88: Simple Feed Water Tank DFM PI Probabilities (TS=2, WL = "Strictly Decreasing")

PI	PI			PI Prob.
1	MF_0 (-2) MF_1 (-1)	WL_1 (-2)	WL_0 (-1)	5.33E-03
2	MF_0 (-2) MF_1 (-1)	WL_1 (-2)	WL_-1 (-1)	5.17E-03
3	WL_1 (-2) WL_0 (-1)	WLM_1 (-2)	MF_1 (-1)	1.85E-03
4	WL_1 (-2) WL_-1 (-1)	WLM_1 (-2)	MF_1 (-1)	1.80E-03
5	MF_1 (-2)	WLM_1 (-2)	MF_1 (-1)	8.25E-04

In Table 98, it is seen that there are major differences between the results with and without the application of DCR. In the case of Table 85, PI 2 and PI 3 contain "WL = 1" at two separate time steps,

which is disallowed under the applied DCR. Any transitions that apply to an increase in "WL" (such as "-1" → "0" or "0" → "1") are similarly disallowed. This leads to extra literals in the PIs in Table 88, as the "WL" state must be in two different, decreasing states if that node appears in the PI. The only PI that is the same between the test runs is PI 7 (Table 85) and PI 5 (Table 88), since this PI does not include the "WL" node in any state, the application of DCR has no effect on it.

Overall, the effects of sink states and other DCRs on the PIs also explain the differences in certain PI probabilities, and Top Event probabilities, as seen in Tables 81-84.

4.4.1.11. Dynamic Test System FTA MCS/PI

In the case of FTA, it is clear that the static nature of the fault tree seen in Figure 87 will not allow for dynamic modelling (i.e. a change in time steps has no meaning to that specific fault tree). In order to model the Feed Water test system for more time steps, and additional fault tree would have to be created, for each additional time step. The fault tree corresponding to the system analysis for two time steps is seen in Figure 93.

In Figure 93, the two "WLM = 1" entries in the fault tree are expanded, as in the DFM model, to incorporate the additional time step. Only one expansion is shown to save space, however the other node expansion is denoted by the transfer gate. Applying the "MOCUS" algorithm to this fault returns the results seen in Table 89. It should be noted that the "transfer gate" is not directly considered by "MOCUS", as it only handles "AND" and "OR" gates. Therefore, one must include the full expansion of "WLM = 1" in both instances when finding the MCS using "MOCUS". The Top Event probabilities are seen in the "TS = 2" row of Table 89.

The full set of MCS, calculated using the CAFTA software tool, are found in Table 90. As in the case of one time step, when the PIs are known, the fault tree can be constructed in such a way that it will return only the PIs, with the same PI and Top Event probabilities as DFM. The fault tree for only the PIs for "TS = -2" is shown in Figure 94.

Table 89: Simple Feed Water Tank FTA Top Event Results (TS = 2)

Run	No. PI	Probability		
		SUM	MCSUB	Exact
TS = 2	26	3.30E-01	2.91E-01	2.11E-01
Sink	23	3.25E-01	2.87E-01	2.08E-01
DCR	11	2.21E-01	2.08E-01	1.80E-01

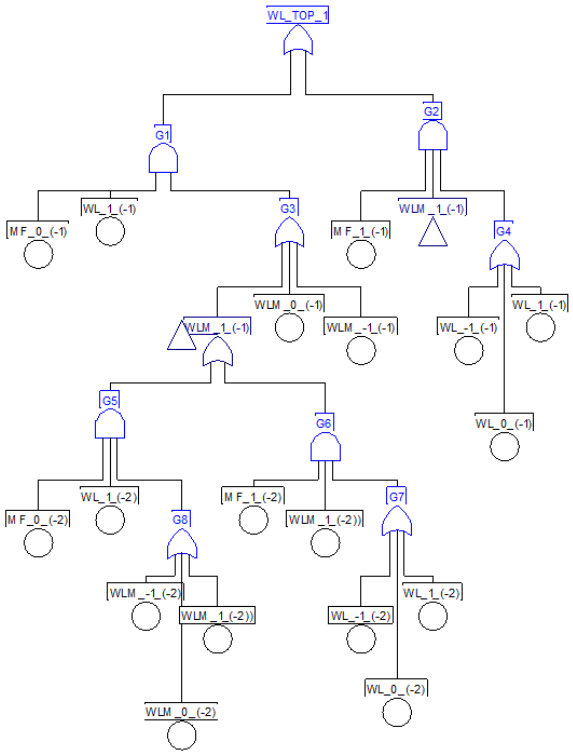


Figure 93: Simple Feed Water Tank Fault Tree (TS = 2)

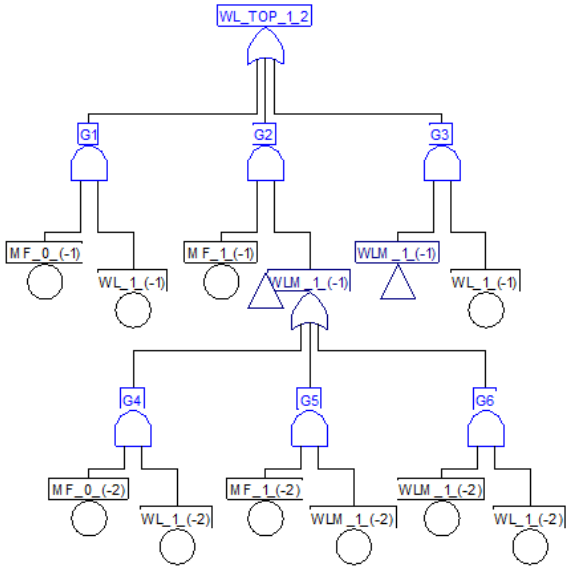


Figure 94: Fault Tree for Simplified Feed Water System (PIs Only, TS = -2)

Fault Tree Sink State

When considering the effect of a "sink state" on the FTA model, there are two issues that are readily apparent. The first issue, as seen in Table 90, 3 of the MCS/PIs, 7, 12, and 13, would violate that sink state condition, as they show a state transition from "MF = 1" → "MF = 0". The Basic Events that comprise these three MCS/PIs are shaded grey, and shown in Figure 95. FTA will not be able to directly apply a sink state during the analysis, as can be done with DFM, this leads to impossible MCS/PIS being produced during the analysis. Additionally, as those impossible MCS/PIs have a probability associated with them, it would affect the Top Event probability of the system. The Top Event probabilities with the disallowed MCS/PIs removed are seen in the "Sink" Row of Table 89.

Table 90: Simple Feed Water System Fault Tree MCS/PI for Two Time Steps

MCS/PI No.	MCS/PI Probability	Literals				
1	1.07E-01	MF_0_(-1)	WLM_0_(-1)	WL_1_(-1)		
2	1.03E-01	MF_0_(-1)	WLM_-1_(-1)	WL_1_(-1)		
3	3.34E-02	MF_0_(-1)	MF_0_(-2)	WLM_0_(-2)	WL_1_(-1)	WL_1_(-2)
4	3.24E-02	MF_0_(-1)	MF_0_(-2)	WLM_-1_(-2)	WL_1_(-1)	WL_1_(-2)
5	3.24E-02	MF_0_(-1)	MF_0_(-2)	WLM_1_(-2))	WL_1_(-1)	WL_1_(-2)
6	1.81E-03	MF_0_(-2)	MF_1_(-1)	WLM_0_(-2)	WL_0_(-1)	WL_1_(-2)
7	1.76E-03	MF_0_(-1)	MF_1_(-2)	WLM_1_(-2))	WL_0_(-2)	WL_1_(-1)
8	1.76E-03	MF_0_(-2)	MF_1_(-1)	WLM_-1_(-2)	WL_0_(-1)	WL_1_(-2)
9	1.76E-03	MF_0_(-2)	MF_1_(-1)	WLM_0_(-2)	WL_-1_(-1)	WL_1_(-2)
10	1.76E-03	MF_0_(-2)	MF_1_(-1)	WLM_0_(-2)	WL_1_(-1)	WL_1_(-2)
11	1.76E-03	MF_0_(-2)	MF_1_(-1)	WLM_1_(-2))	WL_0_(-1)	WL_1_(-2)
12	1.71E-03	MF_0_(-1)	MF_1_(-2)	WLM_1_(-2))	WL_-1_(-2)	WL_1_(-1)
13	1.71E-03	MF_0_(-1)	MF_1_(-2)	WLM_1_(-2))	WL_1_(-1)	WL_1_(-2)
14	1.71E-03	MF_0_(-2)	MF_1_(-1)	WLM_-1_(-2)	WL_-1_(-1)	WL_1_(-2)
15	1.71E-03	MF_0_(-2)	MF_1_(-1)	WLM_-1_(-2)	WL_1_(-1)	WL_1_(-2)
16	1.71E-03	MF_0_(-2)	MF_1_(-1)	WLM_1_(-2))	WL_-1_(-1)	WL_1_(-2)
17	1.71E-03	MF_0_(-2)	MF_1_(-1)	WLM_1_(-2))	WL_1_(-1)	WL_1_(-2)
18	9.54E-05	MF_1_(-1)	MF_1_(-2)	WLM_1_(-2))	WL_0_(-1)	WL_0_(-2)
19	9.26E-05	MF_1_(-1)	MF_1_(-2)	WLM_1_(-2))	WL_-1_(-1)	WL_0_(-2)
20	9.26E-05	MF_1_(-1)	MF_1_(-2)	WLM_1_(-2))	WL_-1_(-2)	WL_0_(-1)
21	9.26E-05	MF_1_(-1)	MF_1_(-2)	WLM_1_(-2))	WL_0_(-1)	WL_1_(-2)
22	9.26E-05	MF_1_(-1)	MF_1_(-2)	WLM_1_(-2))	WL_0_(-2)	WL_1_(-1)
23	8.98E-05	MF_1_(-1)	MF_1_(-2)	WLM_1_(-2))	WL_-1_(-1)	WL_-1_(-2)
24	8.98E-05	MF_1_(-1)	MF_1_(-2)	WLM_1_(-2))	WL_-1_(-1)	WL_1_(-2)
25	8.98E-05	MF_1_(-1)	MF_1_(-2)	WLM_1_(-2))	WL_-1_(-2)	WL_1_(-1)
26	8.98E-05	MF_1_(-1)	MF_1_(-2)	WLM_1_(-2))	WL_1_(-1)	WL_1_(-2)

The second issue is that the “MF = 1” state appears in both time steps (“TS = -1” and “TS = -2”), in nine of the MCS/PIs in Appendix B, MCS/PI 18-26. As the probability of “MF = 1” is low, the MCS/PIs that contain that literal twice end up being the MCS/PIs with the lowest probability of occurring. In sub-section 4.4.1.10, it was seen DFM would account for this sink state, so the same literal would not have to occur in each time step, as depicted in Table 87. In that case, the occurrence of “MF = 1” at “TS = -2” meant that “MF = 1” at “TS = -1” was not required for the PI (PI 4), as it is assumed that the “MF” node will be stuck in that state. The inability of FTA to account for this causes two issues. The first, is that the analysis will return PIs that are actually just Implicants, and the second being that the probability of those Implicants will be underestimated, as the inclusion of the second “MF = 1” literal will reduce the probability of that Implicant occurring. Both of these issues would affect the accuracy of the reliability analysis.

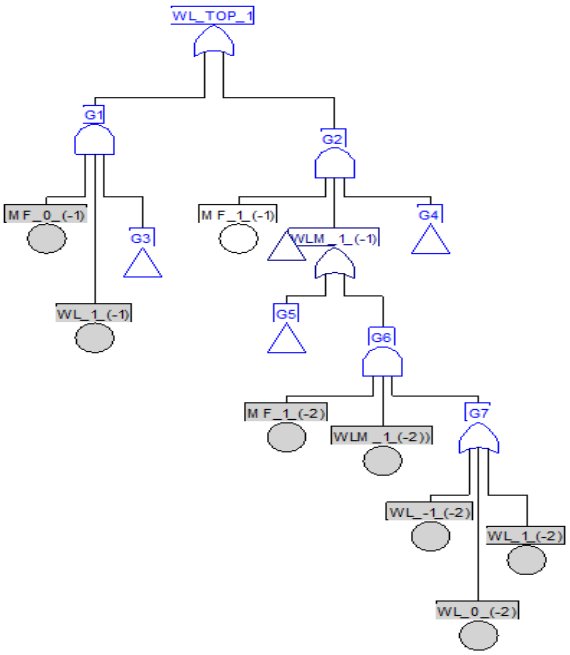


Figure 95: Disallowed Basic Event Combinations (Sink State)

The inclusion of sink states will make the model/analysis more realistic, however in the case of FTA, separate fault trees would have to be constructed, for all potential case(s) of sink states that would be considered. Simply deleting those Basic Events may not correct the problem, as they may be part of other MCS/PIs that are valid. This may be impractical, especially for complex systems.

Fault Tree DCR

Considering the application of the "WL = Strictly Decreasing" DCR to the model, issues similar to those of the sink state arise. In total, only 11 of the 26 MCS/PI from that table would be allowed, with the application of DCR. Those include MCS/PI 1 and 2 ("WL" appears only once), MCS/PI 6, 8, 11, 21, ("WL 1" → "WL 0"), MCS/PI 9, 14, 16, 24 ("WL 1" → "WL -1") and MCS/PI 19 ("WL 0" → "WL -1"). The remaining 15 MCS/PIs all include non-decreasing values for "WL". The Top Event probabilities are amended as such, and are shown in the "DCR" row of Table 89. In the case of DCR however, there is a much greater difference in the number of disallowed MCS/PIs (15), when compared to the sink state test run (3).

As in the case of sink states, DCRs may create a more realistic model, however separate fault trees would have to be created to model the system based on any/all combinations of dynamic consistency rules, which may be impractical for large/complex systems. Failing to do so would again lead to

impossible MCS/PI combinations, and therefore affecting the Top Event probabilities. In should be noted that in this test run, “MF = 1” was not designated as a sink state, so any “MF = 1” → “MF = 0” or “MF = 1” → “MF = 1” state transitions are allowed.

4.4.1.12. Dynamic Test Results Comparison

Qualitative Comparisons

Considering the dynamic results for the DFM and FTA methods, similar issues are seen as with the static comparisons. The FTA results again do not show the same level of logic reduction as the DFM results, leading to a large number of Implicants, which are not Prime Implicants. As an example, Rows 1 and 2 of the FTA results in Table 90, include an extra “WLM” literal, when compared to PI 1 of the DFM results. (Table 15). A second example considers Rows 3-5 in Table 90, where all three states of the “WLM” node are seen, creating three Implicants, where there should only be a single Prime Implicant. As the FTA results are not logically reduced to become PIs, it leads to a much larger number of MCS/PIs returned by FTA (26), when compared to the number of PIs from the DFM results (7).

When the “sink states” and DCR is considered, it is seen that FTA will not directly include these restrictions in the fault tree. A separate fault tree would have to be created for each sink state and/or DCR combination, and for every number of desired time steps, which is likely impractical for larger systems. It is also seen that the application of sink states and DCR will lead to impossible MCS/PIs being generated by the FTA, unless the fault trees are specifically constructed to consider these restrictions. In the case of sink states, it was seen that 3 of the MCS/PIs returned by the FTA would be violate the sink state constraints, and a further 9 MCS/PIs contain the “MF=1” literal in both time steps, which is again incorrect, as once that transition occurs at “TS = -2”, it does not transition again. In the case of DCR, the majority of MCS/PIs returned by the FTA would violate the DCR (15 out of 26 MCS/PIs violate the “strictly decreasing” DCR imposed on the “WL” node). These differences in the PIs returned by the two methodologies also create differences in the quantitative results.

Quantitative Comparisons

As in the case of one Time Step , the quantitative differences are strictly due to the different PIs returned by two methods, for the three different analyses ("TS = 2", "Sink", and "DCR"). When the fault tree was amended to only include the PIs, as seen in Figure 94, the PI and Top Event probabilities were same for both methods. It is seen that, in general, the PIs from DFM have a higher probability of occurring than the Implants returned by FTA, due to the higher number of literals in the FTA MCS/PIs. This, in turn is part of the reason why the Top Event probabilities for the DFM results (Table 86) are generally higher than the Top Event probabilities for the FTA results (Table 85). The notable exception being the "DCR" test run, where the large number of disallowed MCS/PIs in the FTA run, coupled with the additional literals in the DFM PIs, led to the FTA Top Event probability being higher.

4.4.2. Theoretical Reasons for Differences in Reactor Trip Logic Loop Results

The previous stage in the DFM/FTA comparison research considered a one-channel, one-parameter FPGA-based reactor trip logic loop, using a Triple Modular Redundancy (TMR) configuration. A comparison of the results from the FTA and DFM analyses revealed some differences in terms of PIs/MCS, Top Event Probabilities, and the Birnbaum Structural Importance (BSI) measures. Several reasons for those differences were postulated in that paper, with one of those differences being the "Computational Methods" employed by the two software packages. In this case, those methods are the "Method of Generalized Consensus" using DFM, and the "MOCUS" algorithm for FTA. It should be noted that the CAFTA user manual does not directly state which algorithm is used, the CAFTA results did return the same results as expected from applying the "MOCUS" algorithm by hand. Performing the comparisons on the simplified test system in sub-section 4.4.1 allowed for an in-depth theoretical comparison of those two methods, to better determine their effect on the results from the reactor trip logic loop test system. Those differences will be discussed in this section.

4.4.2.1. Prime Implicants vs Implicants

It was seen in the previous research paper that the FTA results produced noticeably more MCS/PIs than DFM, and that each FTA MCS (generally) contained more literals than similar DFM PIs. As discussed in sub-section 4.4.1, this is due to the inability of conventional fault tree methods/software to fully logically-reduce the MVL used in a DFM analysis. Therefore, it was seen that many of the MCS returned by FTA would not actually have been minimal, and instead were Cut Sets (Implicants). An example of was seen in Table 18 of the previous paper, shown again here in Table 91.

Table 91: PIs vs Implicants for "FPGA-Based Reactor Trip Logic Loop"

Node	State	Time Step	Prob
Clock	0	-1	N/A
Trip_Reg (Prev)	1	-1	N/A
SEU (T)	SEU	0	6.53E-05
TMR Circuit "C"	0	0	2.285E-03
PI Probability:			1.492E-07

As discussed in the previous paper, the FTA results contained many more literals than the DFM results, resulting in a series of similar MCS. However, as discussed in sub-section 4.4.1, many of those MCS in the original research paper would actually have been Cut Sets (Implicants), and not a Prime Implicant, as was returned by DFM. This partially explains the large discrepancy in the number of MCS/PIs returned by the two methods, as there would generally be many more Implicants than Prime Implicants. Unlike in the examples shown in the previous sub-section however, in the FPGA-based test system, the FTA did return some PIs, and not just Implicants, shown here in Table 92. It was seen in the original analysis, that both FTA and DFM returned the same PI and PI probability, meaning that FTA was able to identify at least some PIs, and not just the Implicants.

Table 92: Identical PIs with DFM and FTA for the "FPGA-Based Reactor Trip Logic Loop"

Node	State	Time Step	Prob
Clock	1	-1	N/A
SEGR	SEGR_Fail	-1	5.95E-04
SEU (T)	No SEU	0	9.993E-01
TMR Circuit "B"	0	0	2.285E-03
PI Probability:			1.359E-06

4.4.2.2. Missed PIs/Consensus Law

In terms of "Missed" PIs from FTA, an example of this is seen in Table 93. While this PI was produced in the original DFM analysis, it was not discussed in the original paper. This PI is produced when a "SHE" error forces the "P" register to produce a 'Low" value. At the same time, the value of the "Q" register (the Trip Setpoint (TSP)) is also low. If the "Trip Type" was set to "01" (a strictly "P>Q" condition), then the system will not trip. FTA did not return that combination of issues, potentially due to the MVL states of both the "SHE" and "Q" input nodes. This lead to a comparatively high probability PI/MCS being overlooked in the original analysis.

Table 93: "Missed" PI from "FPGA-Based Reactor Trip Logic Loop"

Node	State	Time Step	Prob
Clock	1	-1	N/A
SED_Q	No_SED	-1	9.999E-01
SHE_P	SHE_P_0	-1	2.975E-04
CE (D)	NO_CE (D)	-1	9.999E-01
Q_Register	Q_Low	-1	5.00E-01
Trip Type	01	-1	5.00E-01
SEU (T)	No SEU	-1	9.993E-01
TMR Circuit "C"	0	0	2.285E-03
PI Probability:			1.693E-07

4.4.2.3. Probabilistic Differences

While the computational methods may not solely be the reason for the differences in Top Event probabilities (other factors should as "Initial Conditions" and "Time Steps" were discussed in the original paper), the differences in MCS/PIs returned by the two methods, discussed in sub-section 4.4.1, did have some effect on the Top Event probability of the original test system. Also as seen in sub-section 4.4.1 of this thesis, it was seen that the FTA Top Event probabilities were generally lower than the DFM Top Event probabilities, for the simplified Feed Water System. However, it was seen in the original paper that the DFM Top Event probabilities were lower than that of the FTA Top Event probabilities, when comparing the "SUM" or "MCSUB" values. One reason for that was the much larger number of Implicants returned by FTA eventually catches up with the DFM analysis PIs, resulting in a higher Top Event probability. A second reason, is that in the FPGA-based reactor trip logic loop test system, some of

the high-probability PIs were found (unlike in the Feed Water system), further increasing the Top Event probability

4.4.3. Dynamic Comparisons with Applications to FPGAs

Sub-section 4.4.1 discussed theoretical differences between DFM and FTA methods, and sub-section 4.4.2 applied that information to better explain some of the differences between the DFM and FTA results ascertained in the previous research paper. In this section, a modified portion of the original test system is analyzed for multiple time steps, to determine the differences in the time-dependant results of the DFM and FTA methods.

4.4.3.1. Modified Test System

The test used for this paper was modified from the one in the original work. It was still developed with a reference to the publically available Westinghouse and EPRI documentation [190,211]. Sub-section 4.4.3.2 discusses the modified test system, and sub-section 4.4.3.3 presents additional HDL code ("software") failure modes that were added. It should be noted that the failure modes and corresponding probabilities are the same as those found in Ref. [208]. The exceptions being the inclusion on the HDL failure modes, as discussed in sub-section 4.4.3.2, and the ESD/EOS failure probabilities, which are now based on information from a different literature source [177]. The reliability analysis methods used in this paper are the same as those used in the previous research. Namely, FTA performed using the CAFTA software, and DFM, performed using the Dymonda software tool.

4.4.3.2. Updated FPGA Test System Architecture

One of the potential limitations of DFM is that it is limited to small or medium sized systems, due to the issue of "state explosion" [121,208].The size of the previous test system did not make running the

analysis for multiple time steps feasible. In this paper, the analysis focused on one part of the test system, namely the “Comparator” subsystem. The same TMR configuration described in the original test system is present here. The modified test system is shown in Figure 96.

The modified “Comparator” subsystem now more closely resembles that of the original reference [211]. It includes the “H” register, as well as additional MYXs for feedback and logic selection. The “H” register was intended in the literature to represent a “hysteresis” value, use to confirm or cancel a trip signal. In this case, the output of the “Trip Register” gets passed to the “Trip Check MUX”. In the literature reference, there were multiple “Trip Registers” in which the “Decode” value could be output too. All of these registers would output their values to the “Trip Check MUX”, to be output to the “MUX”. In the test system in this paper, only one “Trip Register” is used, so the “CS” input will always select the value from that register, unless a failure occurs.

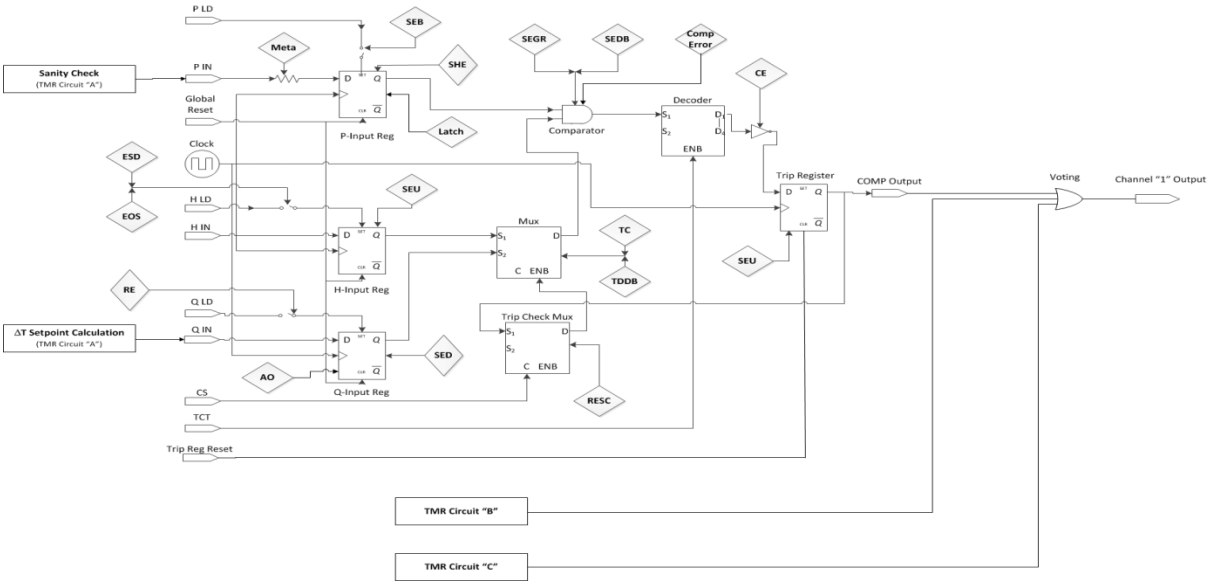


Figure 96: Modified Comparator (COMP) FPGA-Based Test System

Based on the “CS” input, the “Trip Check MUX” value will be passed on as the enable signal to the “MUX”. This “MUX” determines which input value, (“Q” or “H”), is used in the comparator. Under normal circumstances, the “Q” value would be selected and compared with “P”. However, if a “Trip” is found to occur, then the “MUX” will send the data from the “H” register to either confirm or cancel the trip. If the comparison of the “P” value with both the “Q” and “H” values returns a trip, then the trip is

confirmed, and the trip signal will be sent from that TMR circuit. Otherwise, the trip would be cancelled, and no trip signal would be sent.

4.4.3.3. HDL Code Failure Modes

The previous test system only considered failures that could affect the FPGA hardware, such as aging failures or Single Event Effects (SEEs) (the exception being the inclusion of the "Metastability" error). To expand on the failure modes included in the test system, the effects of several HDL code errors were included, using information in the literature [65,131]. These HDL code failures are presented in Table 94. Probabilities for the HDL code failure modes were based on software failure probabilities, found in the literature [220].

It should be noted that the reset signals are not always failures (e.g. a reset/clear can bring the system from an error state into a safe, analyzed state). However, for the purpose of the analysis in this paper, they were assumed to be erroneous reset signals.

Table 94: HDL Code FPGA Failure Modes

Arithmetic Overflow (AO)	Incorrect data type specification leads to lost input/calculation data (e.g. truncated data)
Latch	Accidental latch created instead of a register, produces asynchronous behavior
Comparator Error (Comp Error)	Error in comparator logic due to coding error (e.g. use of ">" instead of "<")
Global Reset	"P", "H" and "Q" registers incorrectly reset (e.g. asynchronous reset error)
Trip Register Reset (Trip Reg Reset)	"Trip" register incorrectly reset (e.g. asynchronous reset error)

4.4.3.4. FPGA Test System Dynamic Results

The fault trees and DFM models were constructed in a manner consistent with the methods described in the original paper [208]. One difference is that in the modified test system, hard errors were designated as “sink states”. Sink states do not affect the analysis when it is run for one time step, but can be a factor once it is run for multiple time steps. There are no “NOT” gates or similar negated logic. Therefore, the fault tree represents a coherent fault tree. The Top Event for these analyses was a “Missed Trip”, the same as in the original research. As this is a per-demand system, the Top Event probabilities on a per-demand basis, and represent the Probability of Failure on Demand (PFD). The DFM models were run for Time Steps of 1, 2, 3 and 4, respectively, with the Top Event Probabilities plotted in Figure 97. The reason for stopping at four time steps was due to the computational intensity of DFM. Using the CNSC laptop on which the Dymonda software tool was installed, four time steps was seen to be the highest number of time steps where the model could be analyzed in a reasonable amount of time (i.e. less than 24 hours). Attempting to run the model for five time steps did not produce any results, even over a 48-hour period. Additional information regarding the computational intensity/state explosion issue can be found in sub-section 5.2.

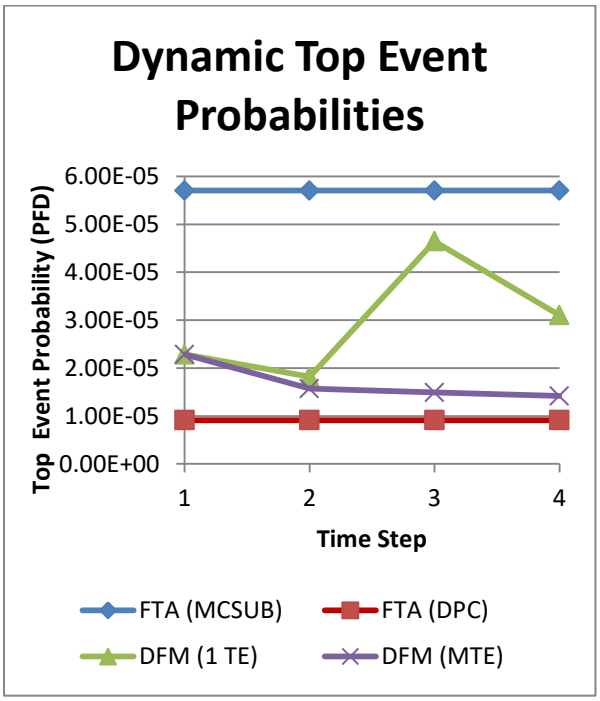


Figure 97: Dynamic Top Event Probabilities (PFD) for DFM and FTA Methods

The number of MCS/PIs for each test run is shown in Table 95. As with the original paper, the oscillating clock presents a problem for FTA, which returns MCS with the clock in both states (“1” and “0”). In the modified test system, the same issue was seen with the “Global Clear” signal. Originally, each register was assigned its own reset signal, however in the modified test system, a global reset was used for registers “P”, “Q” and “H”, to ensure they would be reset at the same time, as advised in the literature [65,131]. Using the global clear resulted in the “Global Reset” signal existing in states “1” and “0” for different registers in the same MCS. As in the case of the “Clock” state, these MCS were pruned after the analysis. The data in Figure 97 and Table 95 represents the data resulting from this pruning.

There were two distinct DFM analyses considered. The first, called “1 Top Event (1TE)”, was set for a “Missed Trip” (Trip Output = “0”), at Time Step “TS = 0”, for all four test runs. This meant that the value of the Trip Output could change through time, and that fluctuation helped produce the dynamic probability behaviour seen in Figure 97. While that is not a strictly realistic example (a positive trip signal could be “Sealed-In”, and not be allowed to switch back to the un-tripped state), it was done to show the effects of different Time Steps on different Top Events. Generally though, the probabilities trend upwards, as over time, there become more possible event combinations that could cause the Top Event. The second case was for “Multiple Top Events (MTE)”, which considered a “Missed Trip” at every time step, except for the initial time step. For example, the test run for three time steps would have the “Trip Output = 0” at “TS = 0”, “TS = -1”, and “TS = -2”, with the “TS=-3” entry left blank. This would represent a more realistic scenario, as the “Missed trip” must occur at multiple time steps, and not bounce between trip/no trip. The probability for this analysis steadily drops with each time step. This is due the “Missed Trip” condition existing in multiple time steps (e.g. Trip Output = 0 at ”TS = 0” and “TS = -1” …).

Table 95: Number of Returned PI/MCS

Test Run	**Time Step**			
	1	2	3	4
FTA	485	485	485	485
DFM (1TE)	69	95	297	335
DFM (MTE)	69	432	357	410

As the fault tree is static, there is no change in the Top Event probabilities or number of MCS with time. On the other hand, as DFM includes time dependant behaviour, the Top Event probabilities and number of PIs does change through time. As expected, the number of PIs increases over time, as failures at different time steps can combine to cause the Top Event ("Missed Trip").

It should be noted that the dynamic DFM Top Event probabilities may or may not converge as the analysis progresses through time. The system may reach a steady-state based on if the same input combinations are enforced and the permanent errors (sink states) accumulate. In the case of the "MTE" run, it does appear that the analysis results progresses towards a convergence (at a value similar to the DPC FTA results), however it is difficult to determine if a convergence will occur with the "1 TE" analysis results. It is also possible that convergence may occur at a time step that is not practical to calculate using DFM (or at least using the Dymonda tool and consumer grade laptops), and therefore it is hard to determine if/when convergence will occur in all cases.

4.4.3.5. Differences Between Dynamic MCS/PIs

The research paper in Ref. [208] compared and contrasted the resulting MCS/PIs, for a DFM model that went for one time step. Additional reasons for those differences are also discussed in sub-section 4.4.2 of this thesis. Here, several differences in MCS/PIs will be compared for the "MTE" run, over multiple time steps. Here, the PIs in the tables will be abbreviated to save space. These PIs are considered to cause the "Missed Trip" Top Event in TMR Circuit "A" only.

In Table 96, it is seen that an "SEU" error in the "H" register at Time Step "TS= -2" was enough to cause in failure in TMR Circuit "A". In the FTA results, the "SEU_H" state was not enough by itself to cause this failure, and was only seen in MCS that included additional failures. This greatly lowered the probability of MCS with that state occurring. Table 97 considers PIs for 3 time steps. It was seen that the "SHE "0"" error in the "P" register could occur at multiple time steps. As it is a "sink state" (hard error), the node cannot transition out of that state, so failures in earlier time steps are able to cause the Top Event. Static FTA analysis would not be able to differentiate between "SHE" failures at different time steps. Lastly, Table 98 considers four time steps. In DFM, it is seen that two separate erroneous reset/clear signals ("CLR_1") could supress the trip. The two reset signals combine to reset all the registers to "0" at their respective time steps, causing the overall trip signal to erroneously read "0". In FTA, these two failures

are independent of each other, and do not appear together in any of the resulting MCS. However, overall that PI is seen to be a very low probability event.

Table 96: "TS = 2" Prime Implicant

Node	State	Time Step	Prob
Clock	1	-2	N/A
Comp_Error	Ok	-2	9.9997E-01
SEU_T	No_SEU_T	-2	9.993E-01
SEU_H	SEU_H_0	-2	3.275E-05
SEU_T	No_SEU_T	-1	9.993E-01
PI Probability:			8.167E-06

Table 97: "TS = 3" Prime Implicant

Node	State	Time Step	Prob
SHE_P	SHE_P_0	-3	2.9750E-04
PI Probability:			3.7119E-05
OR			
SHE_P	SHE_P_0	-2	2.9750E-04
PI Probability:			3.7113E-05

Table 98: "TS = 4" Prime Implicant

Node	State	Time Step	Prob
Global_Reset	CLR_1	-4	2.95E-05
Trip_Reg_Rest	CLR_1	-2	2.95E-05
PI Probability:			5.41E-11

As seen in Figure 97, the Top Event probabilities for the DFM (SUM) results are lower than the FTA (MCSUB) results, and are generally closer to the DPC (exact) results. This was also observed in the preliminary research, due to reasons discussed in sub-section 4.4.2 of this thesis. Although the number of PIs increases through time, often these PIs are of very low probability (as seen in Table 98), due to the inclusion of multiple failures.

An additional point of note, when considering Table 97 and Table 98 is that DFM is able to differentiate, at least to some level, the differences between transient errors and permanent functional failures. Considering the "acceptance criteria" from NUREG/CR-6901 given in Table 1 from sub-section, 1.2.3.1, this provides some evidence that DFM can satisfy "Criteria 8" ("*The model needs to differentiate between faults that cause intermittent failures and faults that cause function failures*") [8]. As DFM models can include "sink states", that should allow for the differentiation of transient and permanent errors. In the case of Table 97, it is seen that the same (permanent) failure can occur in multiple time steps, and will have the same effect on the system (as it is a permanent error, so the system is unable to transfer out of that failed state). In Table 98, transient errors are considered, and exact combinations of error states/time steps are required, in order to produce the erroneous Top Event. More work should be performed into this area to prove that DFM is able to meet "Criteria 8" (as well as the other criteria set out in that report), as discussed in sub-section 6.3.

4.4.4. Risk Importance Measures

Risk Importance Measures (RIMs) are defined in CNSC RD-98 *Reliability Programs for Nuclear Power Plants* as "*A quantitative analysis to determine the importance of variations in equipment reliability to system risk and/or reliability*" [221]. RIMs have seen extensive use in the probabilistic analysis of NPP systems, and selected were considered during this research program. Sub-section 4.4.1 will consider the traditional RIMs and discuss dynamic extensions to those measures. The safety significance of these measures will be discussed in sub-section 4.4.2, with the results and comparisons seen in sub-section 4.4.3.

4.4.4.1. Traditional and Dynamic Risk Importance Measures

There are several RIMs of note that could be considered, that can be found in the literature, a summary of which is given in Table 99 [222,223]. Traditionally, these measures were applied to hardware components, however in digital systems, they are applicable to both hardware and software components [224].

Table 99: Risk Importance Measures for Nuclear Power Plants

Risk Measure	Acronym	Equation	Equation No.
Fussell-Vesely	FV	$\frac{R(base) - R(x_i = 0)}{R(base)}$	6.1
Risk Reduction	RR	$R(base) - R(x_1 = 0)$	6.2
Risk Achievement	RA	$R(x_i = 1) - R(base)$	6.3
Risk Reduction Worth	RRW	$\frac{R(base)}{R(x_i = 0)}$	6.4
Risk Achievement Worth	RAW	$\frac{R(x_i = 1)}{R(base)}$	6.5
Criticality Importance	CR	$\left(\frac{R(x_i = 1) - R(x_i = 0)}{R(base)}\right)(x_1(base))$	6.6
Partial Derivative	PD	$\frac{R(x_i + dx_i - R(x_i)}{dx_i}$	6.7
Birnbaum Importance	BI	$R(x_i = 1) - R(x_i = 0)$	6.8

Where

$R(x_i=1)$ is the increase in risk level where the basic event (x_i) is assumed to be failed

$R(x_i=0)$ is the decrease in risk level when the basic event is assumed to have perfect reliability

$R(base)$ represents the present risk level

$x_i(base)$ represents the present unviability of the x_i component.

The RIM specifically considered in this research work is the Fussell-Vesely (FV) method, defined in RD-98 as [221]:

Fussell-Vesely: "*For a specific basic event, the fractional contribution to PSA results for all accident sequences containing that basic event*."

In some cases, the FV value is calculated using the following equation [209]:

$$\boldsymbol{FV = \sum_{A\ in\ x_i} \frac{P(x_i)}{R(base)}} \quad \textbf{(55)}$$

Essentially, this formula sums all of the MCS that contain the component in a certain state, and then divide that value by the total top event probability, to produce the FV value, and is the method used to calculate FV in the CAFTA software [209]. Research has been performed at VTT, to allow their YADRAT

tool to calculate Dynamic Fussell-Vesely (DFV) values for DFM models. In the literature, the DFV for a random node is given by [122,123]:

$$\boldsymbol{DFV} = \frac{\boldsymbol{Q_{TOP}^{r(-t)=s}}}{\boldsymbol{Q_{TOP}}} \quad \textbf{(56)}$$

Where $Q_{TOP}^{r(-t)=s}$ is the probability that at least one PI, including a random node (*r*) that is in state (*s*) that is either at or before time step (–*t*).

4.4.4.2. Safety Significance of RIMs

The values for FV and RAW can be used to help categorize Structures, Systems and Components (SSC) based on the four Risk-Informed Safety Class (RISC) categories, as defined in Regulatory Guide 1.201 [225]. A further document reference in RG 1.201, designated as NEI-004, lays out guidelines for classifying SSCs into those RISC categories [49]. One possible method is to use the FV importance measures. For example, with an FV value that is greater than 0.005 then that component is a candidate for a "Safety Significant" categorization [226]. This would correspond to a RISC-1 or RISC-2 category for the affected component. Additionally, standards from the US Department of Energy (DOE) and the American Society of Mechanical Engineers (ASME) define a "Significant Basis Event" as "A basic event that contributes significantly to the computed risks for the total risk for risk for a specific hazard group. For internal events, this includes any basic event that has a Fussell-Vesely importance greater than 0.005 [227,228].

4.4.4.3. Risk Importance Measure Results

Table 100 lists the failure modes that have a FV value greater than the "0.005" cut-off, discussed in sub-section 4.4.4. Grouping SEGR and SEDB together (as those values are identical), FTA identified seven such failure modes, with DFM returning eight failure modes. The difference being the "SEU" failure mode in the "H" register. As seen in sub-section 4.4.3, that specific failure mode is only seen by itself in the DFM results, not the FTA results, causing it to have a much higher FV value in the DFM results. Regarding the DFM results, it is seen that there can be a large change in DFM values, as the system

progresses through time. For example, the "SHE_P (0)" failure has its FV values drop significantly over time, while the CE (D) FV values increase over time. Conversely, the SEGR/SEDB FV results were generally constant.

Table 100: DFM/FTA FV Comparison

Failure Mode	FV (FTA)	DFV (DFM)			
		Time Steps			
		1	2	3	4
SEDB/ SEGR	0.136	0.215	0.260	0.267	0.274
CE (D)	0.491	0.323	0.297	0.433	0.495
SHE_P (0)	0.122	0.161	0.187	0.067	1.1E-05
Trip Clear	0.0068	0.011	0.009	0.003	0.0001
Latch Error	0.012	0.016	0.013	0.015	0.019
Comp Error	0.024	0.016	0.015	0.005	1.4E-05
SEU H_1	N/A	0.012	0.007	7.9E-06	5.9E-06
SEU_T	0.069	0.029	0.016	0.0055	2.7E-05

In terms of the changing DFM FV values, it was seen that the failure modes that had FV values that dropped over time tended to do so because they were less likely to be the only failure mode in the PI, at higher time steps. As an example, when the model was run for one time step, the "SHE_P (0)" failure was the only required failure in more PIs, than when the system was run for four time steps (in many cases it was in a PI with an additional failure mode), which reduced its FV value. Alternatively, failures such as SEDB/SEGR and CE (D) that had steady or increasing FV values did so as they had a steady, or increasing number of PIs where they were the only failure mode present. As seen in Table 100, this has the added effect of dropping certain FV values below the cut-off value at higher (or potentially lower) time steps. More work could be performed in this area, to determine what should happen in this case.

4.4.5. Conclusions from Advanced DFM/FTA Comparisons

Overall, conventional FTA methods do not accurately model MVL like DFM does. FTA does not perform all necessary logic reduction operations, and as such will return mostly implicants, not the Prime

Implicants, and often misses certain Prime Implicants. This causes FTA results to have a higher Top Event probability, for larger, more complex systems. Considering dynamic behaviour, it was determined earlier that FTA cannot directly model control loops, however it is also seen that FTA cannot include sink states or dynamic consistency rules, which enforce realistic system behaviour, and have uses in modelling certain failure modes (hard errors). FPGA-based systems exhibit dynamic behaviour in their control logic, and MVL behaviour in certain failure modes and in the logic defined in common language standards. While DFM still suffers from the Issue of "state explosion", it does possess significant advantages over FTA when modelling small to medium FPGA-based systems.

4.5. Chapter Summary

Chapter 4 presented the research results for the DFM modelling of FPGA-based systems, which in turn, represents the majority of the work performed during this research program. Preliminary research using a simple Post Accident Monitoring system demonstrated that DFM could model a simple, static FPGA-based system using only binary logic. A second stage of preliminary research showed that DFM was able to model four important aspects of FPGA-based systems: the IEEE 1164 standard, registers/flip-flops, combinational logic blocks, and a dynamic signal compensator test system, while implementing dynamic behaviour and MVL. These DFM results were corroborated using Modelsim simulations, proving the effectiveness of DFM when modelling and analyzing FPGA-based systems. The next two sub-sections of the chapter focused on the comparison of DFM and FTA when modelling FPGA-based test systems, using the FPGA FMEA/Failure Modes Taxonomy data as a form of fault injection. Comparisons included the Top Event probabilities, MCS/PIs, and the Birnbaum Structural Importance measure, for "Missed Trip" and "Spurious Trip" Top Events. It was seen that there were some similarities, as well as some significant differences between the DFM and FTA results, such as the inability of FTA to model the oscillating clock signal. Potential reasons for the differences were discussed, with the differences in the FTA and DFM algorithms/computational methods analyzed in further detail. This advanced comparative work found that classical FTA algorithms will not properly account for the MVL used in DFM models, often do not intrinsically handle negated logic, and will not properly handle transient errors. Finally, a modified test system was constructed, to allow for multiple time steps to be run in the DFM model, with comparisons of dynamic probabilities, PIs and Risk Importance Measures being presented.

5. Discussion on the Use of DFM for FPGA-Based System Modelling and Analysis

The results of the research program that applied DFM to the analysis of FPGA-based systems has been presented in Sections 3-4. In this section, the discussion will focus on the potential advantages and disadvantages of applying DFM to FPGA-based systems, as well as additional comparisons of DFM to other methods. Sub-section 5.1 will present the potential advantages of DFM. Sub-section 5.2 will look at disadvantages of DFM. Sub-section 5.3 gives a comparison between DFM and Formal Methods. Sub-Section 5.4 provides a summary of the information presented in this chapter.

5.1. Advantages of DFM

This research program has identified several potential advantages of DFM over other commonly-used modelling and reliability methods. Therefore, the potential advantages of DFM are discussed in more detail in this sub-section.

5.1.1. Advantages of DFM Over Static Methods (General)

As seen from the analyses in Section 4, the "Clock" signal is a very important input for FPGA-based systems. The clock states and edge trigger will determine if the data is output through the register(s). However, the clock period and duty cycle could also affect the system. In general, static systems would not include methods to handle the dynamic clock.

An example of this (using the platinum Comparator from sub-section 4.2), was performed. A separate testbench was created, to introduce a sine wave into the system. It was seen that the 10 ns clock period will output the calculated neutron power much faster than a 25 ns clock period. The "Odd Cycle" response changes the standard 50% Duty cycle to a 75% "1", 25% "0" clock, with a period of 25ns. It was seen that there is no effect on the output signal. This is because the register transitions are on clock edges only, so the duty clock cycle does not affect the output value. In the DFM model, the

"Clock_Period" node could introduce a clock delay. The DFM model analysis then returned a clock delay for the "Clock_Period" node that was longer than the standard. This means that the DFM model can predict what could cause the clock delay, and the effect they could have on the system. While the clock periods are usually very fast (on the order of nanoseconds), long delays would slow down the calculations, and could lengthen the trip time.

An example of this is seen in Table 101. A clock delay causes the "Clock_Period" to be "Long", i.e. longer than it should be, meaning it will not transition when it is supposed to. Even though the "Input_voltage" changes to "High", the clock delay disallows the register to output the new value, causing the value for "Phi" to read as "Normal", and not as "High". In the end, this causes the voting logic to indicate a "No Trip". In real operation, this would lead to a delay in the trip signal, as the trip would occur once the next clock transition occurs. The modelling of the clock signals represents a potential advantage of DFM, over traditional (static) reliability analysis methods. The clock is dynamic, as it will change its state ("0" → "1") based on the clock period. In order to properly model the effects of the clock, including clock delays, the analysis must be able to capture these clock effects.

Table 101: PI for Missed Trip Due to Clock Delays

Implicant 1 (Node)	Implicant 1 (State)	Implicant 1 (Time Step)
Clock_En_State	1	-1
Clock_En_Sig	En_1	-1
Clock_Period	**Long**	-1
D	Normal	-1
I(k)_2	Correct Voltage	-1
I(k)_3	**High Voltage**	-1
Input Voltage	Correct Voltage	-1
Prev_Input_1	Correct_Input	-1
Prev_Input_2	Correct_Input	-1
Register_1_Out	Reg_Input_Correct	-1
Register_2_Out	Reg_Input_Correct	-1
Reset_State	X	0
Reset_Signal	R_0	0
Clock_Period	Long	0
I(k)_2	Correct	0
I(k)_3	**High**	0
Phi	**Total_Flux_Normal**	0
Input_Voltage	High Voltage	0
Register_1_Out	Reg_Input_Correct	0
Register_2_Out	Reg_Input_Correct	0

5.1.2. Advantages of DFM Over FTA

Due to the prevalence of the use of FTA for reliability analysis in the nuclear field, a large amount of this research work was focused on comparisons of DFM and FTA. Some final points on the limitations of FTA and potential advantages of DFM over FTA are discussed in this sub-section.

5.1.2.1. Potential Limitations of FTA

It was seen in this paper that there were several potential limitations with using FTA to model the FPGA-based system. The first, being the issue of circular logic. FTA cannot explicitly incorporate circular logic, as logic loops must be broken before the fault tree model can be analyzed, in order to avoid incorrect cut sets and/or to allow the FTA software tool to perform the analysis [229,230]. This makes modelling digital control properties such as control loops/feedback difficult. This creates further issues as digital systems (FPGA, software, etc) generally include control loops. The outputs of the loop may change over time, based on a change in inputs, the output from the previous time step, or from errors in the system.

Using FTA, it would be possible to construct fault tree models that would be analogous to the DFM model, if only fixed time step was considered. This would mean constructing separate fault trees for every desired time step and Top Event combination. The use of Basic Events to represent the previous values for certain states, as seen in Figure 85, could be used to mimic dynamic behaviour. However, each fault tree would only represent one Top Event/time step combination, and could not model any different Top Event/time step combinations. This was seen in sub-section 4.4, where the DFM/FTA results were the same for one time step, but changed significantly when two time steps were considered.

A secondary issue is the use of negated logic. Common fault trees utilize coherent logic, however the inclusion of negated logic (such as “NOT”) gates demands the use of non-coherent logic. While methods exist for analyzing non-coherent fault trees, not all fault tree software packages have that ability. The use of non-coherent logic may complicate FTA, however DFM was developed based on non-coherent logic, so the use of negation poses no problems to DFM [93]. Therefore, it is impractical to use FTA to model these dynamic systems, when compared to the capabilities of DFM. In contrast, it is comparatively much simpler to create and model control loops/feedback loops using DFM than with fault trees.

The second major issue was the way that FTA handles mutually exclusive states, as seen with the oscillating clock. In DFM, this was treated similar to a control loop, where the clock would oscillate from "0" → "1" → "0" → "1", as it would in real life. FTA cannot enforce this clock oscillation (cannot enforce the two mutually exclusive clock states), which lead to a large number of MCS (close to 40% of the total MCS found), that were impossible, due to the clock being in states "0" and "1", simultaneously. This can be partially remedied, by forcing the clock to be either in the "0" state or "1" state at the start of the analysis, or by using software tools to eliminate the impossible MCS after the analysis is performed. It is possible that other dynamic signals, such as the "Reset" and "Enable" signals, or in the individual failure modes, would have the same issue, which can be explored in the future. These issues could potentially limit the effectives of FTA, when modelling and analyzing FPGA-based systems, and other digital systems in general. By comparison, DFM will automatically account for the mutual exclusivity between states (such as the oscillating clock states).

5.1.2.2. Dynamic Behaviour of FPGA-Based Systems

As FPGAs are a form of digital logic, they implement signal processing and/or control algorithms using control loops, which rely on information calculated in a previous loop. In FPGAs, this information is stored in Registers, and then output on the next rising clock edge. DFM is able to model these logic loops, and how the logic states will change through time. It was seen that this is much more difficult to model using a fault tree, as a separate fault tree would have to be created for every time step, as opposed to one DFM model in total. Since static fault trees cannot directly include these control loops/feedback loops, it is not able to model the dynamic register behaviour as accurately as DFM.

Additionally, DFM allows for the inclusion of sink states and dynamic consistency rules in the model/analysis. These additional rules help ensure that the models exhibit realistic behaviour, by filtering out unrealistic PIs. It was seen that the use of these rules caused large differences between the DFM and FTA results. As fault trees are static, sink states and dynamic consistency rules are not considered, and the only way to include them would be to build the fault tree for every case of sink state, dynamic consistency rule, and time step combination, which is likely impractical. Sink states were especially useful for the model in Figure 96. The failures representing "hard errors" were designated sink states, as these errors could not be fixed during analysis. Other failures, such as "soft errors" or the reset

signals would be temporary (transient), and could occur multiple times. The difference between the hard errors/soft errors would not be fully realized with static fault trees.

5.1.2.3. Multiple Valued Logic in FPGA-Based Systems

The use of MVL to discretize parameters into an arbitrary number of states has occurred in the literature, in the case of a (complex) digital Feed Water controller [12,15] and Emergency Core Cooling system (ECC) [122,123]. Using only binary logic may not represent the system as accurately as using MVL, especially with digital systems, where there could be multiple different states for each system parameter. It was seen that even in the case of a very simple test system, as discussed in sub-section 8.1, FTA could not perform the necessary logic reduction operations on the MVL system. This lead to FTA results that were actually Implicants, and not Prime Implicants, which in turn lead to certain PIs not being revealed. With larger, more complex systems, this causes the Top Event probability to be higher than in actuality, while still not identifying all of the actual PIs.

In the case of FPGAs, MVL finds use in the definition of the FPGA logic states. While FPGAs operate on binary signals ("0" and "1"), standards for Verilog and VHDL include additional logic states (mainly useful during simulation). IEEE 1364 standard for Verilog defines four logic states (four-valued logic), including "0", "1", "Z" (High-Impedance), and "X" (Unknown) [40]. The IEEE 1164 standard for VHDL expands on this to include nine logic states [201]. As DFM is an MVL methodology, it would be better able to handle the behavioural modelling of FPGA logic, than the binary logic of fault trees. MVL was also extended to certain FPGA failures, such as the "SHE" and "SEU" failures, to represent "No Failure", "0 → 1 Bit Failure", and a "1→ 0 Bit Failure". FTA would have similar issues, when trying to properly model the failure modes that have more than one potential failure state.

5.1.2.4. Discussion on Relative Accuracy of DFM and FTA

During the comparisons of the DFM and FTA results, the question arises on which of the two methods is the more accurate methods, with regards to modelling and analyzing FPGA-based systems (and digital systems in general). It had been stated in the literature that DFM was able to identify "risk relevant"

sequences not found with traditional methods [12], and that traditional methods tended to overestimate the Top Event probabilities [8]. In terms of the results from this research project, it was determined that there were significant differences with regards to the individual MCSs and PIs returned by the respective methods, due to the different ways that DFM and FTA handle multi-valued logic and time dependant behaviour. Considering the Top Event probabilities in this research work, as seen in sub-section 4.3.10 and sub-section 4.4.3, the FTA results (MCSUB approximation) were larger, more conservative values than the DFM results. It was also seen that the DFM results were generally close to the values of the results returned by the DPC method using the CAFTA software tool, which does not use approximations as done in the MCSUB calculation, and is intended to return a more accurate Top Event probability [209]. By this logic, it appears that DFM produces more accurate probabilistic results from MVL FPGA-based systems that traditional FTA approximations, however more research should be performed on this topic, to directly prove that proposition. Additional comparisons using actual operation experience and/or accident data would also be beneficial, to see if one of the methodologies could better match up with the real-life data.

5.1.3. Advantages of DFM Over Simulation

It was seen in sub-section 4.2 that DFM is able to model the important logic and components of FPGAs, as well as the FPGA-based test system. The Modelsim simulations were intended to prove that DFM could be used accurately for that purpose. However DFM also possesses certain advantages over simulators. Modelsim would work inductively, giving outputs based on the inputs combinations. However, Modelsim will not identify exactly what combinations of events resulted in those outputs. It will just produce the outputs. DFM, on the other hand, will provide a list of PIs that will breakdown the potential cause of the certain events. Another advantage of DFM is modelling error states. With Modelsim, it is difficult to include the effect of hardware failures, whereas it is more easily accomplished in DFM using additional error nodes/states. Modelsim would provide evidence of logic errors. However DFM can be used for that purpose as well. Lastly, DFM can include probabilities in the model, for use in probabilistic analyses, while simulators such as Modelsim cannot.

5.2. Disadvantages of DFM

This research program also identified/confirmed several potential disadvantages of the application of DFM for modelling and analyzing FPGA-based systems. Therefore, the potential disadvantages of DFM are discussed in more detail in this sub-section.

5.2.1. Computational Intensity/State Explosion

At the time of this research work, the chief drawback of DFM is that it is limited to small-medium sized systems. This is due to what is known as the "combinatorial explosion of states" or "state explosion", which occurs with large DFM models and/or large decision tables [121]. This limits the model size, and number of time steps that can be run during the analysis. It is difficult to determine the exact limit on the size of the model for which DFM (in this case the Dymonda tool) can be applied, as this will depend greatly on the number of node/state combinations, number/complexity of the transfer and transition boxes, number of time steps that are required, and the number of specified Top Events. However, this issue was explored during this research, in order to determine a rough approximation on the maximum model size. Consider the DFM model for a simple register, as discussed in sub-section 4.3.10.1. That simple model was run for fifteen time steps, with the computational time plotted against the number of time steps and the number of PIs in Figure 98 and Figure 99, respectively.

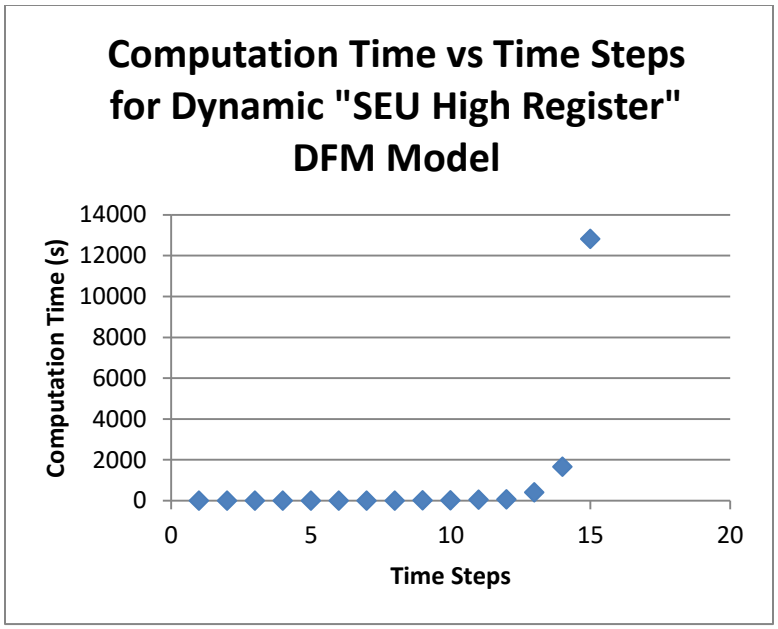


Figure 98: Computational Time vs The Number of Time Steps for the "SEU High Register" Model

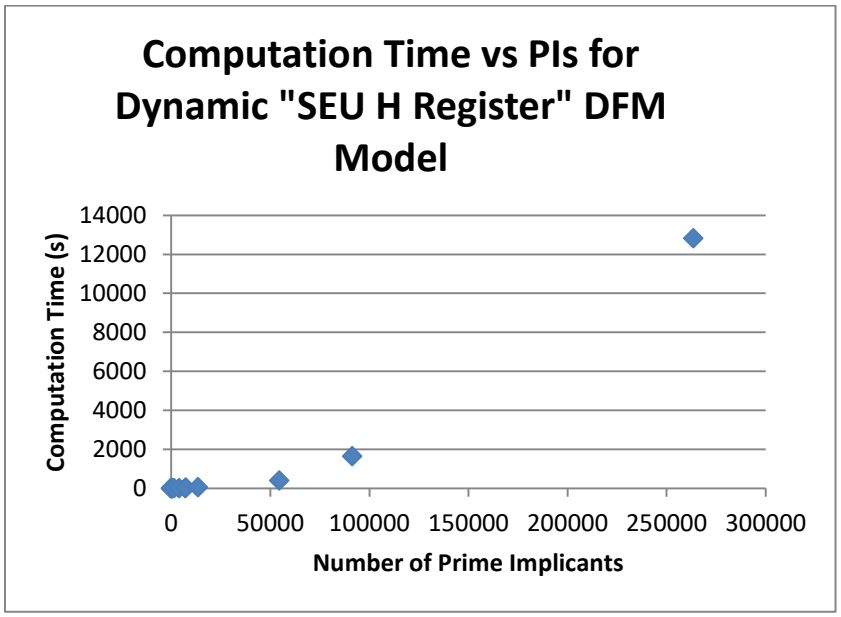


Figure 99: Computational Time vs Number of PIs for "SEU High Register" Model

The issue of "state explosion" becomes evident when inspecting Figure 98 and Figure 99. It was seen that an analysis of eight times steps or less would finish in under one second, and then starting from nine time steps the analysis time started to grow. The computational time increase began to become noticeable at twelve time steps, and then grew rapidly through time steps 13-15, due to the massive

increase in the number of PIs. As seen in Figure 98 and Figure 99, the computational time and number of PIs increases massively when going from fourteen to fifteen time steps, showcasing the issue of "state explosion".

In terms of how this relates to the maximum size of the system model, it was seen during the various DFM analyses performed through the course of this research, that the maximum size of the critical transition table that could be solved in a reasonable amount of time had approximately 100,000 rows, which in turn means approximately 100,000 PIs being identified. The results from Figure 98 and Figure 99 lend evidence to this postulation, as the explosion is seen between time steps 14 and 15, where the number of PIs (number of rows in the critical transition table) crossed the 100,000 "threshold" value. In this example, due to the simplicity of the model (small number of columns in the critical transition table) Dymonda was able to solve for greater than 100,000 PIs, however it was not able to do the same for the larger DFM models that were considered in sub-sections 4.3 and 4.4. It should be noted that the number of PIs returned was far less than 100,000 in those cases, due to the inclusion of truncation values and initial conditions. The use of these features will eliminate many of the PIs (rows from the critical transition table), however that critical transition table may still grow to be quite large before truncation occurs.

As a general rule, it is proposed that for a consumer grade computer (in the case of this research program a dual core 3.2 GHz processor with 8 GB of RAM), the model size should be limited so that the maximum size of the critical transition table should not surpass 100,000 rows, and therefore the maximum number of PIs should be 100,000. However, the use of initial conditions and truncation cut-off probabilities may greatly affect the number of PIs that are actually obtained. It should be noted that the issue of "State Explosion" is not limited to DFM, as it is an issue for other dynamic methods, such as Markov Models [10].

5.2.2. Dynamic Probabilities and Importance Measures

Another issue to consider is how to account for probabilities in models with multiple time steps. The probabilities may be kept static, if the analysis is run over a very short time span, such as the range of a few clock cycles in this paper, then the failure probability will not change over the selected time frame. However, the probabilities may change for each time step if the analysis is assumed to take place over a

large time span. Determining if the probabilities should change, and if so, what the rate of change should be is an important factor in obtaining an accurate quantitative analysis. These leads into a third issue, namely the calculation of dynamic risk importance measures, such as the dynamic FV measure discussed in sub-section 4.4.4. Traditional methods such as the RAW and RRW have use specified "Success" and "Failure" states, however when using MVL, many of the states may not be intrinsic failure states. Additionally, as the system progresses through time, it has to be determined when to calculate the importance measure value for reach node (i.e. is a value calculated at each time step, or at the end of the total analysis). In general, the traditional (static) importance measures must be modified before being applied to DFM models, and dynamic variants for the most common importance measures have been developed and published in the literature [122,123,219].

5.3. Comparison of DFM and Formal Methods

The research discussed in sub-section 4.2 focused on the use of DFM to model the logic of generic FPGA-based systems. In common practice, other methods are used for logic verification, such as Formal Methods. Formal Methods can be defined as "mathematically rigorous techniques and tools for the specification, design and verification of software and hardware systems" [231]. Formal methods utilize mathematical representations of the systems (software and/or hardware), to mathematically verify that the system functions as intended for all input combinations, and as such can uncover errors in the software/hardware. DFM and formal methods have some similarities, as they both can be used to analyze the software and hardware components of digital systems, and uncover errors in the system. Additionally, both methods run into issues for very complex, realistic systems, as it may not be feasible to fully analyze a complex system with either method. Formal methods have also seen use in the verification process of FPGA-based systems [232]. However, it should be noted that at the time of this paper, formal methods have been much more widely used than DFM.

There are some large differences with DFM and formal methods. DFM does not rely on the mathematically rigorous formulations that formal methods do, and instead employs user-input decision tables and PI/MVL functions to analyze the system. While DFM has seen some use in the direct validation of software logic (PIs that do not contain any hardware error states imply a software error) [93], DFM is generally used to analyze the causes of a Top Event, or the effects of an initiating event. For

example, in DFM if a Top Event is set to a “Trip”, then all possible scenarios, either correct or erroneous, would be solved for and stated in the PIs. With DFM, the probabilities of each PI is given, along with the probabilities of the Top Event, allowing for quantitative analyses, such as the calculation of risk importance measures, or the inclusion of DFM results in Probabilistic Safety Assessment (PSA). In contrast, formal methods will mathematically verify that the system works for given inputs, but it does not allow for specific top events to be set, in order to determine the MCS/PIs, and it will not perform probabilistic calculations.

Although DFM and formal methods share some connection for digital systems analysis, they generally serve different purposes. Formal methods are used to mathematically verify the system, while DFM performs a reliability analysis more similar to that of FTA or Markov methods, however direct comparisons on the use of DFM and formal methods were not seen in the literature, at the time of this thesis report.

5.4. Chapter Summary

This chapter provided a discussion on the advantages and the disadvantages of using DFM for the modelling and analysis of FPGA-based systems, resulting from the research presented in Chapter 4. It was seen that advantages of DFM over static methods included modelling of clock delays, proper handling of the oscillating clock signals and the ability to incorporate differences between transient and permanent errors. Furthermore, traditional methods such as FTA do not correctly perform logical reduction operations on MVL that can be incorporated into FPGA systems, and may not apply the “consensus principle” to negated logic, leading to inaccurate MCS/PI results. In comparison with simulations, DFM has the advantages in that it will provide the PIs, listing the possible events that could lead to the final event (Top Event), and do not just show the resulting event. Additionally, DFM can include quantitative analysis, while simulations do not. A discussion of DFM and formal methods was presented, detailing some similarities/differences between the two methodologies, however no direct comparisons were mad e during this research program. Lastly, it was seen that the main disadvantage of DFM was the issue of “state explosion”, limiting the size of model that can be realistically analyzed with DFM. A secondary issue was the proper interpretation of probabilities across multiple time steps.

6. Conclusions and Future Work

The overall conclusions from this research work, as well as potential topics of future work are presented in this section. These conclusions are derived from the results of each section of the research work, and represent the overall conclusions from that research. Individual conclusions, specific to each section of the research work were discussed at the end of the individual sections (Sections 3-4). As a part of these overall conclusions, the limitations of the research work and the possible topics of future work, are discussed. Sub Section 6.1 presents the conclusions from this research work. Sub-section 6.2 provides the recommendations gleamed from the results of the DFM/FPGA research. Sub-section 6.3 discusses potential avenues of future work. Sub-section 6.4 provides a summary of the information discussed in this chapter.

6.1. Conclusions

The objective of this research work was to investigate the use of the Dynamic Flowgraph Methodology (DFM) for the purpose of modelling and analyzing FPGA-based systems. DFM is a modern, dynamic analysis methodology implementing MVL in order to model and analyze digital systems. In the literature, DFM has been applied to software-based digital systems, but not specifically FPGAs. Additionally, as FPGAs are a relatively new technology in the nuclear domain, new reliability and safety analysis data for FPGA-based systems will assist with designers and reviewers of those systems. Therefore, the application of DFM to FPGA-based systems, as well as the FPGA failure mode data, would provide a useful and logical extension to the research and knowledge in the field of digital control systems reliability, especially within the nuclear field.

Over the course of this research program, it was seen that DFM was able to correctly model the logic of several important FPGA components (IEEE 1164 standard, registers and CLBs), as well as the logic for the Dynamic Signal Compensator test system. The results of the Modelsim simulation confirmed the accuracy of the DFM analysis. Two separate comparisons of DFM with FTA showcased the ability of DFM to model complex FPGA-based systems based off of real case studies and industry documentation. Additionally, FPGA failure mode information was reviewed, compiled and categorized, and then

interfaced with an internationally-recognized methodology, to provide important failure mode data that were used in the models in this research.

It was seen that DFM has several advantages over traditional analysis methods. In comparison with simulations, it was seen that DFM will more easily determine the actual cause(s) of the Top Event behaviour, allows easier modelling of hardware and software components, and allows for probabilistic calculations. Considering FTA, DFM is able to incorporate time-dependant behaviour into the system models, which is an important property for digital control systems that rely on feedback/control loops. This effect was particularly evident when considering the oscillating clock behaviour, as FTA would return many impossible cut sets that included the clock in both states. A second dynamic issue was seen when considering transient errors, such as SEUs. These may appear and disappear over time, as was captured in the DFM analysis, but would not be correctly described by FTA It was also seen that FTA does not fully perform MVL logic reductions, and therefore returned many Implicants that were not Prime Implicants, when attempting to model FPGA logic states and failure modes that incorporated MVL states. This lead to an inflated number of MCS/PIs being returned, and affected the Top Event probabilities.

When considering the downsides of DFM, the main issue is still the computational intensity of the modelling process. Solving complex systems, or even simple systems for multiple time steps, requires significant time and computing resources. This limits the size of models that can be realistically analyzed by DFM, which will be smaller, less complex models than software packages such as CAFTA will allow. Overall, it was seen that DFM is an effective tool for modelling and analyzing FPGA-based systems, and possess certain advantages over traditional reliability analysis techniques.

However, there were still some limitations to this research work. In terms of the DFM/FTA comparison, the FPGA-based test system consisted of a one-parameter, one channel system. In reality, the system would need to include multiple trip parameters, and would include several channels (3 or 4 redundant channels). The use of multiple channels would introduce the issue of CCF, and as FPGAs are a digital technology, that would include both hardware and software CCF. Generally, other forms of mitigation methods, on top of TMR are also applied to FPGA-based systems, but were not considered with the test system modelling. In order to realize the true potential and ability of DFM to model FPGA-based systems, additional DFM models including multiple parameters, channels, mitigation methods and CCF would need to be analyzed.

A second point, is that the DFM modelling was largely focused on hardware failures (such as SEEs, aging process failures, etc.). In the preliminary DFM/FTA comparison work, it was assumed that software failures (HDL code failures) would be eliminated in the "Design" stage of the lifecycle, so they were not directly considered in that comparison. The advanced DFM/FTA comparisons included certain HDL code failures, although they were treated in a very similar way to the hardware failures. The DFM/Modelsim comparisons considered FPGA logic to a very general, abstract extent, but did not focus on specific HDL code failures. Overall, the effects of software (HDL code failures) should be modelled more extensively with DFM, in order to determine the overall effectiveness of DFM when modelling and analyzing FPGA-based systems.

6.2. Recommendations

From the results obtained through the work performed during this thesis, it is recommended that DFM be applied to model and analyze FPGA-based I&C systems/safety systems, up to a certain size. As it was shown that DFM has certain advantages over traditional methods (e.g. FTA, simulation), the application of DFM would lead to the improved reliability and safety analysis of safety critical FPGA-based systems. Therefore, it is recommended that DFM be used alongside simulation and in place of FTA when directly modelling small to medium sized FPGA-based systems, and to augment the results of Fault/Event Tree analysis when considering the total system (i.e. analog, digital, mechanical) and overall NPP PSA.

In terms of the exact size of the model that DFM can successfully model, it will strongly depend on the model's level of detail (node/state combinations) and the actual computer system that is running the DFM tool. However, as a general rule it was seen that the largest critical transition table (largest number of PIs for the selected top event) that could be handled by a consumer-grade laptop was roughly 100,000 rows (100,000 PIs).

In addition, it is recommended that the research into the application of DFM for the modelling and analysis of FPGA-based systems be continued, with potential topics of future research discussed in sub-section 6.3.

6.3. Potential Topics for Future Work

Through the work performed during this research program, several potential avenues of future research were identified. Therefore, eight potential future research topics are briefly discussed in this section, all of which could expand upon the DFM/FPGA research from this thesis.

1.) Failure Mode Mitigation

DFM could be used to model diagnostics and/or correction methods for mitigating failures such as SEEs. Several methods exist to correct and/or detect errors, including Parity Bits, Error Detection and Correction codes (EDC), Double or Triple Modular Redundancy (DMR/TMR) Cyclic Redundancy Checks (CRC), Reed-Solomon Codes (RS), etc, as discussed in the literature [174]. DFM could be used model the different methods, to show how they would be performed inside of an FPGA. DFM could be used determine the effectiveness of the different mitigating techniques, as well as how their effectiveness could change through time. Probabilistic effects could be included, as there are failure rates for many of these detection methods that are available in the literature [130,133]. This would allow for a comparison of how the different mitigation methods would affect the probability of success/failure through the different time steps, as well as allow us to see how the mitigation methods would affect the values of risk importance measures.

2.) Architecture Comparisons (Qualitative CCF):

The work involved in the DFM/FTA comparison paper consisted of one form of system architecture, essentially a one channel system with TMR. There are other architectures that could be considered for safety system design (such as those discussed in IEC 61131-6) [233], which include the different uses of redundancy and diagnostics. DFM could be used to evaluate and compare the different system architectures, to determine the strengths, weaknesses and possible improvements for each architecture. This architecture comparison could include a qualitative approach to CCF modelling, as one could probe the effects of CCF on different architectures, as well as evaluate mitigations for those CCF errors. This could allow for a more in-depth analysis of CCF, which was not considered in the previous DFM/FTA comparison.

3.) Application to Software-/Based Systems with Comparisons to FPGAs:

The study discussed in the this thesis report focused solely on FPGAs, however other forms of digital systems, such as software-based systems see use in NPPs. DFM could be used to analyze a software-based system such as a PLC-based system, in a similar way to the work performed in the studies done so far. This could include the hardware failure modes of the PLC, as well as software errors, which may be more prevalent in the PLC than in an FPGA. Alternatively, PLC-based safety system architecture (such as those shown in IEC 61131-6) [233] could be modelled, and comparisons could be made with an equivalent FPGA-based system to determine the potential advantages and disadvantages of both technologies.

4.) HDL Coding Analysis:

The DFM/FTA research work focused mainly on the hardware faults (such as aging failures or SEEs), with a cursory take on software failures, which were treated in a more similar way to the hardware failure modes. Therefore, research into software/coding faults could be an extension of that work. DFM could be used to investigate software/coding failures in the FPGA (or other digital systems), such as failures in state machines, causes and effects of software common cause failure, calculating software failure probabilities, or used in conjunction with formal verification methods to analyze errors in the HDL (or software) code. This would be a separate analysis into the actual programming of the system, allowing for both the hardware and software aspects of these systems to be analyzed. HDL (VHDL) code for the Comparator component has already been written, making it a potential test case (or part of a potential test case) for software analysis. As in previous cases, this could include FPGAs, but would be applicable to software-based systems as well.

5.) Additional Methodology Comparisons:

The comparisons in the DFM/FTA report were limited to DFM and FTA, however those are not the only methods of performing hazard analysis. Other methods, such as Markov analysis [234], Markov CCMT (Cell-to-Cell-Mapping Technique, a dynamic Markov variant) [15], and Dynamic Fault Trees [235] could be considered for comparison with DFM. These other methodologies have their own way of implementing time dependant behaviour, as well as their own benefits and drawbacks. Comparisons

with the VTT YADRAT DFM tool could be made, to determine the comparative accuracy of the two DFM tools when modelling FPGA (or other digital) systems. The comparison with the YADRAT tool could include comparisons of Top Event probabilities, Prime Implicants, importance measures and computation time.

A second potential topic would be the comparison of inductive analysis methods. The Dymonda tool is capable of performing an inductive analysis (the YADRAT was unable too, at the time of writing the report). White FTA is not capable of carrying out an inductive analysis, Event Trees are used for that purpose. Similar dynamic/static comparisons between DFM and Event Trees could be performed, so determine the similarities and differences, as well as the advantages and disadvantages of both methods.

6.) Dynamic CCF:

In the DFM/FTA comparisons, CCF was not specifically considered, as the test system consisted of one channel. A CCF analysis could be performed, as well as include Dynamic CCF, as it is done with the YADRAT tool [123]. The dynamic method can include components in a common cause group at multiple time steps (i.e. the components do not have to fail all at the same time). This would not be possible with standard FTA, since it lacks time-dependant properties. Dynamic CCF results could show how CCF events occur through time, and could also be compared to FTA CCF calculations, to give an in-depth comparison of CCF between the two methodologies.

7.) Additional Quantitative Measures:

In this research work, certain quantitative parameters were considered, such as Top Event probabilities, BSI and the Dynamic FV risk importance measures. There exist additional RIMs that are applicable to DFM [122,123], as discussed in sub-section 4.4.4. Comparisons of these additional importance measures could be made to the analogous FTA (traditional importance measures), as was done with the Dynamic and traditional FV RIM. Similarly, other quantitative analyses, such as uncertainty analysis or sensitivity analysis could be applied to DFM models, and could also be used to compare DFM results to FTA results.

8.) Additional Quantitative Measures:

As identified in this research work, and elsewhere in the literature, the main limitation of DFM is the computational intensity, which limits its applicability to small-medium sized systems. Therefore it has been suggested in the literature that DFM results be integrated with the results of traditional analysis methods, such as FTA and Event Trees [27,28]. In those documents it was advised that DFM should be applied to the software-based components of a system, to provide more detailed information in order to construct more accurate fault trees and/or event trees. The same concept could be applied to FPGA-based systems, where DFM is used to analyze in detail the FPGA hardware and/or software, and include those results as part of a larger, traditional system analysis.

9.) Completion of the NUREG/CR-6901 Criteria:

Considering the "acceptance criteria" from NUREG/CR-6901 given in Table 1 from sub-section, 1.2.3.1, it was seen that for four of those criteria (criteria 4, 6-8), it had not yet been shown that DFM would meet the requirements set out by the authors of that report [8]. Regarding criteria 8 ("*The model needs to differentiate between faults that cause intermittent failures and faults that cause function failures*"), there was some evidence from the DFM analyses in this report from sub-section 4.4.3 that DFM models can be constructed that will differentiate between transient errors and permanent errors. Therefore, another potential topic of future research would be to prove if DFM is or is not able to meet those additional criteria, and if it cannot, to determine ways in which the methodology could be improved.

6.4. Chapter Summary

General conclusions from the research program are discussed in this chapter. These include the potential advantages and disadvantages of DFM, as well as certain limitations of this research program. Several recommendations and potential future topics of research, which also consider FPGAs and/or DFM are outlined.

References

[1] IAEA. Application of Field Programmable Gate Arrays in Instrumentation and Control Systems of Nuclear Power Plants. Vienna: International Atomic Energy Agency; 2016.

[2] IAEA. Technical Challenges in the Application and Licensing of Digital Instrumentation and Control Systems in Nuclear Power Plants. Vienna: International Atomic Energy Agency; 2015.

[3] Ranta J. The Current State of FPGA Technology in the Nuclear Domain. Vuorimiehentie, Finland,: VTT Technical Research; 2012.

[4] Naser J. Guidelines on the Use of Field Programmable Gate Arrays (FPGAs) in Nuclear Power Plant I&C Systems. Palo Alto: EPRI; 2009.

[5] Naser J. "Recommended Approaches and Design Criteria for Application of Field Programmeable Gate Arrays in Nuclear Plant Instrumentation and Control. Palo Alto: EPRI; 2011.

[6] McNelles P, Lu L. A Review of the Current State of FPGA Systems in Nuclear Instrumentation and Control. Proceedings of the 2013 21st International Conference on Nuclear Engineering, Chengdu, China: ASME; 2013. doi:10.1115/ICONE21-16819.

[7] Menon C, Guerra, S. Field Programmable Gate Arrays in Safety Related Instrumentation and Control Applications. Sweden: Energiforsk; 2015.

[8] Aldemir T, Miller DW, Stovsky MP, et al. Current State of Reliability Modeling Methodologies for Digital Systems and Their Acceptance Criteria for Nuclear Power Plant Assessments. Washington DC: U.S. Nuclear Regulatory Commission; 2006.

[9] Committee on the Safety of Nuclear Installations. Failure Modes Taxonomy for Reliability Assessment of Digital Instrumentation and Control Systems for Probabilistic Risk Analysis. Paris: OECD-NEA; 2015.

[10] Committee on the Safety of Nuclear Installations. RECOMMENDATIONS ON ASSESSING DIGITAL SYSTEM RELIABILITY IN PROBABILISTIC RISK ASSESSMENTS OF NUCLEAR POWER PLANTS. Paris: OECD-NEA; 2009.

[11] Aldemir T, Stovsky MP, Miller DW, et al. Dynamic Reliability Modeling of Digital Instrumentation and Control Systems for Nuclear Reactor Probabilistic Risk Assessments. Washington DC: U.S. Nuclear Regulatory Commission; 2007.

[12] Aldemir T, Guarro S, Kirshenbaum J, et, al. A Benchmark Implementation of Two Dynamic Methodologies for the Reliability Modeling of Digital Instrumentation and Control Systems. Washington DC: U.S. Nuclear Regulatory Commission; 2009.

[13] Aldemir T. A survey of dynamic methodologies for probabilistic safety assessment of nuclear power plants. Annals of Nuclear Energy 2013;52:113–24. doi:10.1016/j.anucene.2012.08.001.

[14] ASCA Inc. Dymonda 7.0 Software Guide. Redondo Beach, California: ASCA Inc.; 2013.

[15] Aldemir T, Guarro S, Mandelli D, et al. Probabilistic Risk Assessment modeling of digitla instrumentation and control using two dynamic methodologies. Reliability Engineering and System Safety 2010:1011–39.

[16] Aldemir T, Miller DW, Stovsk M, et al. Methodologies For The Probabilistic Risk Assessment of Digital Reactor Protection and Control Systems. Nuclear Technology 2007;159:167–91.

[17] Authen S, Holmberg J-E. Reliability Analysis of Digital Systems in a Probabilistic Risk Analysis for Nuclear Power Plants. Nuclear Engineering and Technology 2012;44:471–82.

[18] Zio E. Integrated deterministic and probabilisitic safety assessment: Concepts, challenges, research directions. Nuclear Engineering and Design 2014;280:413–9. doi:10.1016/j.nucengdes.2014.09.004.

[19] Milici A, et al. Plant Management Advisor System (PMAS): An architecture for performing diagnostic reasoning and decision support in plant process management. Proceedings of the Probabilistic Safety Assessment and Management (PSAM) Conference, Crete: 1996.
[20] Milici A. Assisting emergency operating procedures execution with AMAS, an Accident Management Advisor System, Taipei: 1994.
[21] Milici A, et al. "Extending the Dynamic Flowgraph Methodology (DFM) to model human performance and team effects. Washington DC: U.S. Nuclear Regulatory Commission; 2001.
[22] Motamed M, et al. Development of tools for safety analysis of control software in advanced reactors," U.S. Nuclear Regulatory Commission Report. Washington DC: U.S. Nuclear Regulatory Commission; 1996.
[23] Garett C, Apostolakis G. Automated hazard analysis of digital control systems. Reliability Engineering and System Safety 2001:1–17.
[24] Garett C, Guarro S, Apostolakis G. The Dynamic Flowgraph Methodology for Assessing the Dependability of Embedded Software Systems. IEEE Transactions on Systems, Man and Cybernetics 1995;25:824–40.
[25] Al-Dabbagh A, Lu L. Dynamic flowgraph modeling of process and control systems of a nuclear-based hydrogen production plant. International Journal of Hydrogen Energy 2010:9569–80.
[26] Al-Dabbagh A, Lu L. Reliability modeling of networked control systems using dynamic flowgraph methodology. Reliability Engineering and System Safety 2010:1202–9.
[27] ASCA Inc. Risk-Informed Safety Assurance and Probilistic Risk Assemment of Mission Critical Software-Intensive Systems. Redondo Beach, California: ASCA Inc.; 2007.
[28] ASCA Inc. Context-Based Software Risk Model (CSRM) Application Guide. Redondo Beach, California: ASCA Inc.; 2013.
[29] Xilinx. Xilinx: All Programmable n.d.
[30] Altera 2016.
[31] Lattice Semiconductor. Lattice Semiconductor 2016.
[32] Microsemi Corporation. Microsemi 2016.
[33] Opal Kelly. Opal Kelly 2014.
[34] Digilent. Digilent: A National Instruments Company 2016.
[35] Terasic Inc. Terasic 2013.
[36] IEEE Power and Energy Society. IEEE Standard Criteria for Programmable Digital Devices in Safety Systems of Nuclear Power Generating Stations. New York, NY: IEEE; 2016.
[37] International Electrotechnical Commission. Development of HDL-programmed integrated circuits for systems performing category A functions. Geneva, Switzerland: IEC; 2012.
[38] Digital I&C Working Group. Common Position on the Treatment of Hardware Description Language (HDL) Programmed Devices for Use in Nuclear Safety Systems. OECD-NEA; 2013.
[39] National Instruments. FPGA Fundamentals 2012.
[40] IEEE. IEEE Standard for Verilog Hardware Description Language. New York, NY: IEEE; 2005.
[41] IEEE. IEEE Standard VHDL Language Reference Manual. New York, NY: IEEE; 2009.
[42] IEEE. IEEE Standard for SystemVerilog--Unified Hardware Design, Specification, and Verification Language. New York, NY: 2012.
[43] Xilinx All Programmable. Vivado Design Suite 2016.
[44] Altera Corporation. Quartus Prime Software 2016.
[45] Mathworks. HDL Coder 2016.
[46] MyHDL Community. MyHDL 2015.
[47] National Instruments. IP Corner: The LabVIEW Fixed-Point Data Type Part 1 – Fixed-Point 101 2011.
[48] Taylor A. The Basics of FPGA Mathematics. Xcell Journal 2012:44–9.

[49] Bishop D. Fixed Point Package User's Guide 2010.
[50] Bishop D. Floating Point Package User's Guide 2010.
[51] Fink R, Killian C, Nguyen T, Druilhe A, Daumas F, Naser J. Guidelines and a primer on application of field-programmable gate arrays in nuclear plant I&C systems. NPIC&HMIT 2010, Las Vegas, Nevada: ANS; 2010, p. 1305–55.
[52] Bach J, Tavolara, I. Use of FPGA Technology in Implementation of the Logic of the Modernized Rod Control System (RCS) of the 900 MW EDF Fleet. NPIC&HMIT 2010, Las Vegas, Nevada: ANS; 2010, p. 1326–36.
[53] Preckshot GG. Method for Performing Diversity and Defense-in-Depth Analyses of Reactor Protection Systems. Livermore, California: Lawrence Livermore National Laboratory; 1994.
[54] Wood RT, et al. Diversity Strategies for Nuclear Power Plant Instrumentation and Control Systems. Washington DC: U.S. Nuclear Regulatory Commission; 2008.
[55] Torok R. Guidelines for Performing Defense-In-Depth and Diversity Assessments for Digital Upgrades: Applying Risk Informed and Deterministic Method. Palo Alto: EPRI; 2004.
[56] IEC. Nuclear power plants - Instrumentation and control systems important to safety - Software aspects for computer-based systems performing category A functions. Geneva, Switzerland: IEC; 2006.
[57] Koga R, et al. Comparison of Xilinx Virtex-II FPGA SEE Sensitivities to Protons and Heavy Ions. IEEE Transactions on Nuclear Science 2004;51:2825–33. doi:10.1109/TNS.2004.835057.
[58] Hall TS. Field-programmable analog arrays: A floating-gate approach. PhD Dissertation. Georgia Institute of Technology, 2004.
[59] She J, Jiang J. On the Speed of Response of an FPGA-based Shutdown System in CANDU Nuclear Power Plants. Nuclear Engineering and Design n.d.;241:2280–7. doi:10.1016/j.nucengdes.2011.03.050.
[60] She J, Rankin DJ, Jiang J. Evaluation of Safety PLCs and FPGAs for Shutdown Systems in CANDU Nuclear Power Plants. ISSNP/CSEPC/ISOFIC, China: 2008.
[61] She J, Jiang J. Potential improvement of CANDU NPP safety margins by shortening the response time of shutdown systems using FPGA based implementation 2012;244:43–51. doi:10.1016/j.nucengdes.2012.01.003.
[62] Xing A, de Grosbois J, Archer, P, Awwal A, Sklyar V. FPGA-Based Controller in CANDU® Nuclear Safety-Reactor Applications. NPIC&HMIT 2010, Las Vegas, Nevada: ANS; 2010, p. 1337–44.
[63] Clarkson, G. FPGA Based Safety Related I&C Wolf Creek Generating Station, France: IAEA; 2008.
[64] CS Innovations, LLC. Licensing the ALS FPGA Based Safety Related I&C Platform 2009.
[65] Bobrek M, et al. Review Guidelines for Field Programmable Gate Arrays in Nuclear Power Plant Safety Systems. Washington DC: U.S. Nuclear Regulatory Commission; 2010.
[66] Becker JR. License Amendment Request 2011.
[67] Wang AB. Diablo Canyon Power Plant, Unit Nos. 1 And 2 - Acceptance R eview Of License Amendement Request For Digital Process Protection System Relacement 2012.
[68] Lu JJ, Chou HP, Wong KW. Conceptual Design of FPGA-based RPS for the Lungmen Nuclear Power Plant. NPIC&HMIT 2010, Las Vegas, Nevada: ANS; 2010, p. 944–53.
[69] Huang H, Chou H, Lin C. Design Of A FPGA Based ABWR FeedWater Controller. Nuclear Engineering and Technology 2012:363–8.
[70] Miyazaki T, Oda N, Goto Y, et al. Qualification of FPGA-based safety-related PRM system. Proceeding of NPIC&HMIT, Knoxville, Tennessee: 2009, p. 70.
[71] Kojima, et al. Qualification of Toshiba's FPGA-based safety-related systems. Proceeding of NPIC&HMIT, Las Vegas, Nevada: 2010, p. 935–43.
[72] Bahkmach E, et al. FPGA-based Technology and Systems for I&C of Existing and Advanced Reactors, Vienna, Austria: IAEA; 2009.

[73] Kharchenko V. Diversity-Oriented FPGA-Based NPP I&C Systems: Safety Assessment, Development, Implementation. Proceedings of the 18th International Conference on Nuclear Engineering, Xian, China: ICONE; 2010, p. 755–64.
[74] Kharchenko V, Siora O, Sklyar V. Multi-Version FPGA-Based Nuclear Power Plant I&C Systems: Evolution of Safety Ensuring, Nuclear Power - Control, Reliability and Human Factors 2011.
[75] FPGA-Based NPP Instrumentation and Control Systems: Development and Safety Assessment. n.d.
[76] Chen CK. A Petri Net Design of FPGA-based Controller for a Class of Nuclear I&C Systems. Nuclear Engineering and Design 2011;241:2597–2603. doi:10.1016/j.nucengdes.2011.04.004.
[77] Esposito B, Riva M, Marocco, D, Yaschuck, Y. A Digital Acquisition and Elaboration System for Nuclear Fast Pulse Detection. Nuclear Instruments and Methods in Physics Research A 2006;572:355–7. doi:10.1016/j.nima.2006.10.335.
[78] Schiffler R, Flaska M, Pozzi S, Carney S, Wentzloff D. A scalable FPGA-based digitizing platform for radiation data acquisition. Nuclear Instruments and Methods in Physics Research A 2011:491–3. doi:10.1016/j.nima.2010.10.039.
[79] Westinghouse Electric Company. Westinghouse Receives Final NRC Approval for Advanced Logic System® Safety System Solution. Westinghouse Press Releases 2013. http://www.westinghousenuclear.com/News_Room/PressReleases/pr20130918.shtm (accessed February 11, 2014).
[80] Yoo J, Lee HJ, Lee JS. A Research on Seamless Platform Change of Reactor Protection System from PLC to FPGA. Nuclear Engineering and Technology 2013;45:477–88.
[81] Srivastava GP. Electronics in nuclear power programme of India: An overview. Sadhana Academy Proceedings in Engineering Sciences 2013;38.
[82] International Atomic Energy Agency (IAEA). IAEA Safety Glossary: Terminology Used in Nuclear Safety and Radiation Protection. Vienna, Austria: IAEA; 2007.
[83] Stoelinga M, Ruijters, E. Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. Computer Science Review 2015;15–16:29–62. doi:10.1016/j.cosrev.2015.03.001.
[84] Ericson CA. Fault Tree Analysis - a history. Proceedings of the 17th International System Safety Conferenc, Orlando, Florida: n.d., p. 1–9.
[85] Vesely WE, Goldberg FF, Roberts NH, Haasl DF. Fault Tree Handbook (NUREG-0492). Washington DC: U.S. Nuclear Regulatory Commission; 1981.
[86] Lewis HW, Budnitz RJ, Kouts HJC, Loewenstein WB, Rowe WD, von Hippel F, et al. Risk Assessment Review Group report to the U. S. Nuclear Regulatory Commission. Washington DC: U.S. Nuclear Regulatory Commission; 1978.
[87] Stamatelatos M, Vesely W, et al. Fault Tree Handbook with Aerospace Applications. Virgina, USA: NASA; 2002.
[88] International Electrotechnical Commission. Fault tree analysis (FTA). Geneva, Switzerland: IEC; 2006.
[89] Beeson SC. Non coherent fault tree analysis. Loughborough University, 2002.
[90] Bendell A, Ansell J. The incoherency of multistate coherent systems. Reliability Engineering 1984;8:165–78. doi:10.1016/0143-8174(84)90022-2.
[91] Andrews JD. To not or not to not!!, Fort Worth, Texas: 2000.
[92] Andrews JD. The Use of Not Logic in Fault Tree Analysis. Quality and Reliability Engineering International 2001;17:143–50. doi:10.1002/qre.405.
[93] Yau M, Apostolakis G, Guarro S. The use of prime implicants in dependability of software controlled systems. Reliability Engineering and System Safety 1998;62:23–32. doi:Get rights and content.

[94] Kumamoto H, Henley EJ. Top-Down Algorithm for Obtaining Prime Implicant Sets of Non-Coherent Fault Trees. IEEE Transactions on Reliability 1978;R-27:242–9. doi:10.1109/TR.1978.5220351.
[95] Lu L, Jiang J. Joint Failure Importance for Noncoherent Fault Trees. IEEE Transactions on Reliability 2007;56:435–43. doi:10.1109/TR.2007.898574.
[96] Remenyte-Prescott R, Andrews JD. An Efficient Real-Time Method of Analysis for Non-coherent Fault Trees. Quality and Reliability Engineering International 2009;25:129–50. doi:10.1002/qre.955.
[97] Kececioglu DB. Reliability Engineering Handbook. vol. 2. Lancaster, USA: DEStech Publications; 2002.
[98] Andrews JD. Tutorial: Fault Tree Analysis. Proceedings of the 16th International System Safety Conference, Seattle, Washington: 1998.
[99] Bartlett LM, Andrews JD. Choosing a Heuristic for the 'Fault Tree to Binary Decision Diagram" Conversion, Using Neural Networks". IEEE Transactions on Reliability n.d.;51:344–9. doi:10.1109/TR.2002.802892.
[100] Schneewelss WG. Boolean Functions with Engineering Applications and Computer Programs. Berlin: Springer-Verlag,; 1989.
[101] Rauzy A. New algorithms for fault tree analysis. Reliability Engineering and System Safety 1993;40:203–11. doi:10.1016/0951-8320(93)90060-C.
[102] Deng Y, Wang H, Guo B. BDD algorithms based on modularization for fault tree analysis. Progress in Nuclear Energy n.d.:192–9. doi:http://dx.doi.org/10.1016/j.pnucene.2015.06.019.
[103] University of Maryland. Hybrid Causal Risk Methodology for Risk Assessment. College Park, Maryland: ProQuest; 2007.
[104] Andrews JD, Remenyte-Prescott R. Fault Tree Conversion to Binary Decision Diagrams. Proceedings of the 23rd International System Safety Conference, San Diego, USA: System Safety Society; 2005. doi:https://dspace.lboro.ac.uk/2134/3645.
[105] Matuzas V, Contini S. Dynamic Labelling of BDD and ZBDD for efficient non-coherent fault tree analysis. Reliability Engineering and System Safety 2015;144:183–92. doi:10.1016/j.ress.2015.07.012.
[106] Chu TL, Apostolakis G. Methods for Probabilistic Analysis of Noncoherent Fault Trees. IEEE Transactions on Reliability 1980;R-29:354–60. doi:10.1109/TR.1980.5220881.
[107] Collet J. Some Remarks on Rare-Event Approximation. IEEE Transactions on Reliability 1996;45:106–8. doi:10.1109/24.488924.
[108] Caldarola L. Fault tree analysis with multistate components. New York. Plenum Press; 1980.
[109] Garribba S, Guagnini E, Mussio P. Multiple-Valued Logic Trees: Meaning and Prime Implicants. IEEE Transactions on Reliability 1985;R-34:463–72.
[110] Nelson RJ. Simplest Normal Truth Function. Journal of Symbolic Logic 1954;20:105–8. doi:10.2307/2266893.
[111] Mott TH. Determination of the irredundant normal forms of a truth function by iterated consensus of the prime implicants. IEEE Transactions on Electronic Computers n.d.;9:245–52. doi:10.1109/TEC.1960.5219824.
[112] Quine WV. The problem of simplifying truth functions. American Mathematical Monthly 1952;59:521–31. doi:10.2307/2308219.
[113] Quine WV. A way to simplify truth functions. American Mathematical Monthly 1955;62:627–31. doi:10.2307/2307285.
[114] Hulme BL, Worrell, R. B. A Prime Implicant Algorithm with Factoring. IEEE Transactions on Computers 1975;C-24:1129–31. doi:10.1109/T-C.1975.224146.

[115] Lisnianski A, Levitin, S. Multi-state System Reliability: Assessment, Optimization and Applications. Singapore: World Scientific Publishing; 2003.
[116] The map method for synthesis of combinational logic circuits. Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics 1953;72:593–9. doi:10.1109/TCE.1953.6371932.
[117] Ogunbiyi EI, Henley EJ. Irredundant Forms and Prime Implicants of a Function with Multistate Variables. IEEE Transactions on Reliability 1981;30:39–42. doi:10.1109/TR.1981.5220957.
[118] Enderton H. A Mathematical Introduction to Logic. 2nd ed. New Yotk: Harcourt Academic Press; 2001.
[119] Dixon Pa. Decision Tables and Their Applications. Computers and Automation 1964:14–9.
[120] Ogunbiyi EI. Application of decision tables to risk analysis studies. University of Houston, 1980.
[121] Bjorkman K. Solving Dynamic Flowgraph Methodology Models Using Binary Decision Diagrams. Reliability Engineering and System Safety 2013:206–16.
[122] Tyrvainen T. Risk importance measures in the dynamic flowgraph methodology. Reliability Engineering and System Safety 2013:35–50.
[123] Tyrvainen T. Risk Importance Measures and Common Cause Failures in the Dynamic Flowgraph Methodology. Aalto University, 2011.
[124] Tyrvainen T. Prime Implicants in Dynamic Reliability Analysis. Reliability Engineering and System Safety 2016;146:39–46. doi:http://dx.doi.org/10.1016/j.ress.2015.10.007.
[125] Karanta I. Implementing dynamic flowgraph methodology models with logic programs. Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, 2013, p. 302–14.
[126] McNelles P, Zeng ZC, Renganathan G. Modelling of Field Programmable Gate Array Based Nuclear Power Plant Safety Systems Part I: Failure Mode and Effects Analysis. Proceedings Of the 7th International Conference on Modelling and Simulation in Nuclear Science and Engineering, Ottawa, Ontario, Canada: 2015.
[127] McNelles P, Zeng ZC, Renganathan G, Chirila M, Lu L. Failure Mode Taxonomy for Assessing the Reliability of Field Programmable Gate Array Based Instrumentation and Control Systems. Reliability Engineering and System Safety n.d.
[128] US NRC. Advanced Logic System Topical Report. Washington DC: U.S. Nuclear Regulatory Commission; 2010.
[129] International Electrotechnical Commission. Analysis techniques for system reliability - Procedure for failure mode and effects analysis (FMEA). Geneva, Switzerland: IEC; 2006.
[130] International Electrotechnical Commission. Functional Safety of electrical/electronic/programmable electronic safety related systems- Part 2: Requirements for electrical/electronic/programmable electronic safety-related systems. Geneva, Switzerland: IEC; 2010.
[131] Bobrek M, Bouldin, D. Review Guidelines for FPGAs in NPP Safety Systems. Oak Ridge, Tennessee: Oak Ridge National Labs (ORNL); 2010.
[132] Radio Technical Commission for Aeronautics (RTCA) Incorporated. Design Assurance Guidance For Airborne Electronic Hardware. Washington DC: RTCA; 2000.
[133] International Electrotechnical Commission. Functional Safety of electrical/electronic/programmable electronic safety related systems- Part 7: Overview of techniques and measures. Geneva, Switzerland: IEC; 2010.
[134] European Space Agency (ESA). Sneak Analysis- Part 1: Method and Procedure. Noordwijk, The Netherlands: ESA; 1997.
[135] European Space Agency (ESA). Sneak Analysis – Part 2: Clue list. Noordwijk, The Netherlands: ESA; 1997.

[136] Hahn HA, Blackman HS, Gertman, David I. Applying Sneak Circuit Analysis to the Identification of Human Errors of Commission. Reliability Engineering and System Safety 1991;33:289–300. doi:10.1016/0951-8320(91)90065-F.
[137] Remnant M. The Application Of Sneak Analysis To Safety Critical FPGAs. University of York, 2009.
[138] Sejin J, Kim E-S, Yoo J, Kim J-Y, Choi JG. An evaluation and acceptance of COTS software for FPGA-based controllers in NPPs. Annals of Nuclear Energy n.d.;94:338–49. doi:10.1016/j.anucene.2016.03.026.
[139] Preckshot GG. Proposed Acceptance Process for Commercial Off-The-Shelf (COTS) Software in Reactor Applications. Washington DC: U.S. Nuclear Regulatory Commission; 1996.
[140] Electric Power Research Institute (EPRI). Plant Engineering: Guideline for the Acceptance of Commercial Grade Design and Analysis Computer Programs Used in Nuclear Safety-Related Applications. Palo Alto, California: EPRI; 2013.
[141] CSA Group. Qualification of digital hardware and software for use in instrumentation and control applications for nuclear power plants. Toronto, Canada: CSA; 2015.
[142] Canadian Nuclear Safety Commission. Design of Reactor Facilities: Nuclear Power Plants. Ottawa, Ontario, Canada: CNSC; 2014.
[143] Avizienis, A, Laprie JC, Randall B, Landwehr C. Basic concepts and taxonomy of dependable and secure computing," in IEEE Transactions on Dependable and Secure Computing. IEEE Transactions on Dependable and Secure Computing 2004;1:11–33. doi:10.1109/TDSC.2004.2.
[144] Huffmire T. Handbook of FPGA Design Security. New York, NY: Springer; 2010.
[145] Huffmire T, et al. Managing Security in FPGA-Based Embedded Systems. IEEE Design and Test of Computers 2008;25:590–8. doi:10.1109/MDT.2008.166.
[146] Faller R. Specification of a Software Common Cause Analysis Method. Computer Safety, Reliability and Security, Nuremburg Germany: 2007, p. 162–71.
[147] O'Connor A, Mosleh A. A general cause based methodology for analysis of common cause and dependant failures in system risk and reliability assessments. Reliability Engineering and System Safety 2016;145:341–50. doi:10.1016/j.ress.2015.06.007.
[148] Kang HG, Kim HE. Unavailability and spurious operation probability of k-out-of-n reactor protection systems in consideration of CCF. Annals of Nuclear Energy 2012;49:102–8. doi:10.1016/j.anucene.2012.06.012.
[149] Hassija V, Kumar CS, Velusamy K. A pragmatic approach to estimate alpha factors for common cause failure analysis. Annals of Nuclear Energy 2014;63:317–25.
[150] International Electrotechnical Commission. Nuclear power plants – Instrumentation and control important to safety – General requirements for systems. Geneva, Switzerland: 2011.
[151] International Electrotechnical Commission. Nuclear Power Plants-Instrumentation and control systems important to safety-Requirements for coping with common cause failure (CCF). Geneva, Switzerland: IEC; 2007.
[152] IEEE Standard Criteria for Safety Systems for Nuclear Power Generating Stations. New York, NY: IEEE Power and Energy Society; 2009.
[153] International Atomic Energy Agency (IAEA). Protecting Against Common Cause Failures in Digital I&C Systems of Nuclear Power Plants. Vienna, Austria: IAEA; 2009.
[154] International Atomic Energy Agency (IAEA). Design of Instrumentation and Control Systems for Nuclear Power Plants. Vienna, Austria: IAEA; 2016.
[155] JEDEC Solid State Technology Association. Failure Mechanizms and Models for Semiconductor Devices. Arlington, Virginia: JEDEC; 2011.
[156] Ozarin N. What's wrong with bent pin analysis, and what to do about it. Proceedings of the Reliability and Maintainability Symposium (RAMS), Las Vegas, Nevada: IEEE; 2008, p. 386–92. doi:10.1109/RAMS.2008.4925827.

[157] IEEE. IEEE Standard for Boundary-Scan Testing of Advanced Digital Networks. New York, NY: IEEE; 2015.
[158] van der Goor AJ, Gaydadjiev GN, Yarmolik V N, Mikitjuk VG. March LR: A Test for Realistic Linked Faults. Proceedings of the 14th VLSI Test Symposium, Princeton, N.J.: IEEE; 1996, p. 272–80. doi:10.1109/VTEST.1996.510868.
[159] National Aeronautics and Space Administration (NASA). Microelectronics Reliability: Physics-of-Failure Based Modeling and Lifetime Evaluation. Pasadena, California: NASA; 2008.
[160] Srinivasan S, Krishnan R, Mangalagiri P, Xie Y, Narayanan V, Irwin MJ, et al. Toward Increasing FPGA Lifetime. IEEE Transactions on Dependable and Secure Computing 2008;5:115–27.
[161] Actel. Reliability Considerations for Automotive FPGAs. San Jose, California: Microsemi; 2003.
[162] Benfica J. Analysis of SRAM-Based FPGA SEU Sensitivity to Combined EMI and TID-Imprinted Effects. IEEE Transactions on Nuclear Science 2016;63:1294–300. doi:10.1109/TNS.2016.2523458.
[163] Korash K, Hassan M, Tanaka TJ, Wood RT. Technical Basis for Environmental Qualification of Microprocessor-Based Safety-Related Equipment in Nuclear Power Plants. Washington DC: U.S. Nuclear Regulatory Commission; n.d.
[164] CSA Group. Environmental Qualification of Equipment for CANDU Nuclear Power Plants. Toronto, Canada: CSA; 2015.
[165] Sturesson F. Single Event Effects (SEE) Mechanism and Effects 2009.
[166] Mutuel, LH. Appreciating the Effectiveness of Single Event Effect Mitigation Techniques. Proceedings of the 33rd Digital Avionics Systems Conference, Colorado Springs, USA: IEEE; 2014, p. 5B1-1-5B1-11. doi:10.1109/DASC.2014.6979481.
[167] Mutuel, LH. Single Event Effect Mitigation Techniques. New Jersey, USA: Federal Aviation Administration (FAA); 2016.
[168] Titus JL. An Updated Perspective of Single Event Gate Rupture and Single Event Burnout in Power MOSFETs,. IEEE Transactions on Nuclear Science 2013;60:1912–28. doi:10.1109/TNS.2013.2252194.
[169] Scheik L. Testing Guidelines for Single Event Gate Rupture (SEGR) of Power MOSFETs. Pasadena, California: NASA; 2008.
[170] Wang X, Holber KE, Clark, L. T. Single event upset mitigation techniques for FPGAs utilized in nuclear power plant digital instrumentation and control. Nuclear Engineering and Design 2011;341:3317–24. doi:10.1016/j.nucengdes.2011.06.033.
[171] Kastensmidt F, Rech P. FPGAs and parallel architectures for aerospace applications: soft errors and fault-tolerant design. Cham, Switzerland: Springer-Verlag; 2015.
[172] Kretzschmar U, Gomex-Cornejo J, Astarioa A, Bidarte U, Del Der J. Synchronization of faulty processors in coarse-grained TMR protected partially reconfigurable FPGA designs. Reliability Engineering and System Safety 2016;191:1–9. doi:10.1016/j.ress.2015.12.018.
[173] Smith F. Single event upset mitigation by means of a sequential circuit state freeze. Microelectronics Reliability 2012;52:1233–40. doi:10.1016/j.microrel.2011.11.019.
[174] Habinc S. Lessons Learned from FPGA Developments. Gloteborg, Sweden: Gaisler Research; 2002.
[175] Xilinx All Programmable. Understanding and Mitigating System-Level ESD and EOS Events in Xilinx 7 Series Device. San Jose, California: Xilinx; 2013.
[176] Actel Corporation. Electro-static discharge. Montain View, California: Microsemi; 2005.
[177] Khalaquzzaman M. A model for estimation of reactor spurious shutdown rate considering maintenance human errors in reactor protection system of nuclear power plan. Nuclear Engineering and Design n.d.;240:2693–971. doi:10.1016/j.nucengdes.2010.05.031.

[178] International Electrotechnical Commission. Nuclear power plants - Instrumentation and control important to safety - Requirements for electromagnetic compatibility testing. Geneva, Switzerland: IEC; 2009.
[179] International Electrotechnical Commission. Electromagnetic compatibility (EMC) - Part 4-1: Testing and measurement techniques - Overview of IEC 61000-4 series. Geneva, Switzerland: IEC; 2016.
[180] Mulder ED, Ors SB, Preneel B, Verbauwhede I. Differential power and electromagnetic attacks on an FPGA implementation of elliptic curve cryptosystems. Computers and Electrical Engineering 2007;33:367–82. doi:10.1016/j.compeleceng.2007.05.009.
[181] Trimberger SM, Moore JJ. FPGA Security: Motivations, Features and Applications. Proceedings of the IEEE 2014;102:1248–65. doi:10.1109/JPROC.2014.2331672.
[182] Hadzic I, Udani S, Smith JM. FPGA Viruses. 1999.
[183] US NRC. Cyber Security Programs for Nuclear Facilities. Washington DC: U.S. Nuclear Regulatory Commission; 2010.
[184] Mossman T. IDSRS Chapter 7, Appendix A: I&C Perspectives on Hazard Analysis (HA). Washington DC: U.S. Nuclear Regulatory Commission; 2013.
[185] Brombacher A, van Beurden IWR. RIFIT: analyzing hardware and software in safeguarding system. Reliability Engineering and System Safety n.d.;66:149–56. doi:10.1016/S0951-8320(99)00032-0.
[186] Monmasson E, Cirstea MN. FPGA Design Methodology for Industrial Control Systems – A Review. IEEE Transactions on Industrial Electronics 2007;54:1824–42. doi:10.1109/TIE.2007.898281.
[187] Lu J-J, Hsu T-C, Chou HP. System Assessment of an FPGA-Based RPS for ABWR nuclear power plant. Progress in Nuclear Energy 2015;85:44–55. doi:10.1016/j.pnucene.2015.05.010.
[188] Lu J-J, Huang H-H, Chou HP. Evaluation of an FPGA-based fuzzy logic control of feed-water ABWR under automatic power regulating. Progress in Nuclear Energy 2015;79:22–31. doi:10.1016/j.pnucene.2014.10.010.
[189] Yichan W, et al. Development, verification and validation of an FPGA-based core heat removal protection system for a PWR. Nuclear Engineering and Design 2016;301:311–9. doi:10.1016/j.nucengdes.2016.03.018.
[190] Westinghouse Electric Company. AP1000 Design Control Document (Revision 15), Chapter 7: Instrumentation and Controls 2015.
[191] McNelles P, Lu L. Lab-Scale Design, Demonstration and Safety Assessment of an FPGA-Based Post Accident Monitoring System for Westinghouse AP1000 Nuclear Power Plants. Proceedings of the 2014 22nd International Conference on Nuclear Engineering, Prague: ASME; 2014. doi:10.1115/ICONE22-30457.
[192] McNelles P, Lu L, Abi-Jaoude Ma-J. Dynamic Flowgraph Methodology Assessment of an FPGA-Based Postaccident Monitoring System for Westinghouse AP1000 Nuclear Power Plants. Journal of Nuclear Engineering and Radiation Science 2015;1:4 Pages. doi:10.1115/1.4029591.
[193] Criteria For Accident Monitoring Instrumentation For Nuclear Power Plants. Washington DC: U.S. Nuclear Regulatory Commission; 2006.
[194] IEEE Power and Energy Society. IEEE Standard Criteria for Accident Monitoring Instrumentation for Nuclear Generating Stations,. New York, NY: IEEE; 2016.
[195] Canadian Standards Association. Requirements for monitoring and display of nuclear power plant safety functions in the event of an accident,. Toronto, Canada: CSA; 2014.
[196] National Instruments. cRIO-9076 2016.
[197] Item Software. Bellcore / Telcordia - Reliability Prediction Procedure 2016.
[198] Holmberg J-E. Software Reliability Analysis in Probabilistic Risk Analysis. Nuclear Safety and Simulation 2012:281–91.
[199] Mentor Graphics. ModelSim 2016.

[200] McNelles P, Lu L. Field Programmable Gate Array Reliability Analysis Using the Dynamic Flowgraph Methodology. Nuclear Engineering and Technology 2016. doi:http://dx.doi.org/10.1016/j.net.2016.03.004.
[201] IEEE. IEEE Standard Multivalue Logic System for VHDL Interoperability (Std_logic_1164). New York, NY: IEEE; 1993.
[202] Synario Design Automation. VHDL Reference Manual. Redmond, Washington: Synario Design Automation; 1997.
[203] Mishra AK, Shimjith SR, Bhatt TU, Tiwari AP. Dynamic Compensation of Vanadium Self Powered Neutron Detectors for Use in Reactor Control. IEEE Transactions on Nuclear Science 2013;60:310–8. doi:10.1109/TNS.2012.2229719.
[204] Mishra AK, Shimjith SR, Bhatt TU, Tiwari AP. Kalman Filter-Based Dynamic Compensator for Vanadium Self Powered Neutron Detectors. IEEE Transactions on Nuclear Science 2014;61:1360–8. doi:10.1109/TNS.2014.2321340.
[205] Lynch GF, Shields RB, Coulter PG. Characterization of Platinum Self-Powered Detectors. IEEE Transactions on Nuclear Science 1977;24:692–5. doi:10.1109/TNS.1977.4328769.
[206] Todt WH. Characteristics of Self-Powered Neutron Detectors Used in Power Reactors. Horseheads, New York: Imaging and Sensing Technology Corporation.; n.d.
[207] Borairi M. Reactor Regulating System (Lecture Notes) 2014.
[208] McNelles P, Zeng ZC, Renganathan G, Lamarre G, Akl Y, Lu L. A Comparison of Fault Trees and the Dynamic Flowgraph Methodology for the Analysis of FPGA-based Safety Systems Part 1: Reactor Trip Logic Loop Reliability Analysis. Reliability Engineering and System Safety 2016;153:135–50. doi:http://dx.doi.org/10.1016/j.ress.2016.04.014.
[209] Electric Power Research Institute (EPRI). CAFTA Fault Tree Analysis. Palo Alto, California: EPRI; 2007.
[210] Chu TL, et al. Modeling a Digital Feedwater Control System Using Traditional Probabilistic Risk Assessment Methods. Washington DC: U.S. Nuclear Regulatory Commission; 2009.
[211] Electric Power Research Institute (EPRI). Design Description of a Prototype Implementation of Three Reactor Protection System Channels Using Field-Programmable Gate Arrays. Palo Alto, California: EPRI; 1997.
[212] Hwang I, Kim S, Kim Y, Seah CE. A Survey of Fault Detection, Isolation and Reconfiguration Methods. IEEE Transactions on Control Systems Technology 2010;18:636–53. doi:10.1109/TCST.2009.2026285.
[213] Salewski F, Taylor A. Fault Handling in FPGAs and Microcontrollers in Safety-Critical Embedded Applications: A Comparative Survey, Lubeck, Germany: IEEE; 2007, p. 124–31. doi:10.1109/DSD.2007.4341459.
[214] Altera Corporation. Understanding Metastability in FPGAs. San Jose, California: Altera; 2009.
[215] Todd B. Reliability Considerations for CPLD/FPGA Based Designs. Europe: CERN; n.d.
[216] Xilinx All Programmable. Device Reliability Report. San Jose, California: Xilinx; 2014.
[217] Singh, M, Koren I. Incorporating Fault Tolerance in Analog-To-Digital Converters (ADCs). Proceedings of the International Symposium on Quality Electronic Design, IEEE; 2002, p. 286–91. doi:10.1109/ISQED.2002.996753.
[218] McNelles P, Zeng ZC, Renganathan G. Modelling Radiation-Induced Failures in FPGAs Using the Dynamic Flowgraph Methodology. Transactions of the American Nuclear Society, vol. 113, Washington D.C: 2015, p. 415–8.
[219] Karanta I. Importance Measures for the Dynamic Flowgraph methodology. Finland: VTT Technical Research; 2011.

[220] Khalaquzzaman M, et al. Estimation of reactor protection system software failure probability considering undetected faults. Nuclear Engineering and Design 2014;280:201–9. doi:10.1016/j.nucengdes.2014.09.008.
[221] Canadian Nuclear Safety Commission. Reliability Programs for Nuclear Power Plants. Ottawa, Ontario, Canada: CNSC; 2012.
[222] Borst M van der, Schoonakker H. An Overview of PSA Importance Measures. Reliability Engineering and System Safety 2001;72:241–5. doi:10.1016/S0951-8320(01)00007-2.
[223] Lu L, Jiang J. Probabilistic Safety Assessment for Instrumentation and Control Systems in Nuclear Power Plants: An Overview. Nuclear Science and Technology 2004:323–30.
[224] Kamyab S, Nematollahi M, Shafiee G. Sensitivity analysis on the effect of software-induced common cause failure probability in the computer-based reactor trip system unavailability. Annals of Nuclear Energy 2013;57:294–303. doi:10.1016/j.anucene.2013.01.049.
[225] US NRC. Guidelines for Categorizing Structures, Systems and Components in Nuclear Power Plants According to their Safety Significance. Washington DC: U.S. Nuclear Regulatory Commission; n.d.
[226] Nuclear Energy Institute (NEI)`. SSC Categorization Guideline. Washington D.C: NEI; 2005.
[227] American Society of Mechanical Engineers (ASME). Addenda to ASME RA-S-2008: Standard for Probabilistic Risk Assessment for Nuclear Power Plant Applications. New York, NY: ASME; 2009.
[228] US Department of Energy (DOE). Development of Probabilistic Risk Assessments for Nuclear Safety Applications. Washington D.C: DOE; 2013.
[229] American Society of Mechanical Engineers (ASME). Standard for probabilistic risk assessment for nuclear power plant applications. New York, NY: ASME; 2002.
[230] US NRC. Interim Reliability Evaluation Program Procedures Guide. Washington D.C: U.S. Nuclear Regulatory Commission; 1983.
[231] Butler RW. What is Formal Methods 2001.
[232] Yalin H. Exploring Formal Verification Methodology for FPGA-based Digital Systems. Alburquerque, N.M: Sandhia National Laboratories; 2012.
[233] International Electrotechnical Commission. Programmable controllers - Part 6: Functional safety. Geneva, Switzerland: IEC; 2012.
[234] Kharchenko V, Butenko V, Odarushchecnko O, Sklyar V. Multifragmentation Markov Modeling of a Reactor Trip System. Journal of Nuclear Engineering and Radiation Science 2015;1. doi:10.1115/1.4029342.
[235] Cepin M, Mayko B. A dynamic fault-tree. Reliability Engineering and System Safety 2002;75:83–91. doi:10.1016/S0951-8320(01)00121-1.
[236] International Electrotechnical Commission. Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 4: Definitions and abbreviations. Geneva, Switzerland: IEC; 2010.
[237] IEEE Computer Society. Systems and software engineering -- Vocabulary. New York, NY: IEEE/ISO/IEC; 2010.
[238] JEDEC Solid State Technology Association. Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray Induced Soft Errors in Semiconductor Devices. Virgina, USA: JEDEC; 2006.
[239] Altera Corporation. Introduction to Single Event Upsets. San Jose, California: Altera; 2013.

Appendices

Appendix I: DFM and FTA Results for the "SEU High Register" Model

CAFTA MCS Results:

Cutset Report

HLD_INPUT_REG_HIGH = 2.38E-01 (Probability)

Prob.	%	Class	Inputs...				
9.00E-02		37.9%	CLK_0	HLD_CLR_0	H_IN_PREV_HIGH	NO_SEU_H	
9.00E-02		72.3%	HLD_0	HLD_CLR_0	H_IN_PREV_HIGH	NO_SEU_H	
4.50E-02		88.0%	CLK_1	HLD_1	HLD_CLR_0	H_IN_HIGH	NO_SEU_H
3.60E-02		100.0%	HLD_CLR_0	H_IN_HIGH	H_IN_PREV_HIGH	NO_SEU_H	
6.53E-06		100.0%	CLK_0	H_IN_0	SEU_H		
6.53E-06		100.0%	CLK_0	H_IN_LOW	SEU_H		
6.53E-06		100.0%	HLD_0	H_IN_0	SEU_H		
6.53E-06		100.0%	HLD_0	H_IN_LOW	SEU_H		
6.53E-06		100.0%	HLD_CLEAR_1	SEU_H			
3.27E-06		100.0%	CLK_1	HLD_1	H_IN_0	SEU_H	
3.27E-06		100.0%	CLK_1	HLD_1	H_IN_LOW	SEU_H	
2.61E-06		100.0%	H_IN_0	H_IN_PREV_0	SEU_H		
2.61E-06		100.0%	H_IN_LOW	H_IN_PREV_LOW	SEU_H		

Report Summary:
Filename: \\ot1pfp001\user$\data\mcnellesp\My Documents\HLD_INPUT_REG_HIGH.cut
Print date: 2016-07-14 1:11 PM
Not sorted
Printed in full

CAFTA DPC Results:

Gate Probability Report

Input Filename: \\ot1pfp001\user$\data\mcnellesp\My Documents\HLD_INPUT_REG_HIGH.DPC.CAF

Name	Min	Point Est.	Max	
..G$	8.000E-01	8.000E-01	8.000E-01	(0.0000E+00)
G001	1.440E-01	1.440E-01	1.440E-01	(0.0000E+00)
..G002	4.500E-02	4.500E-02	4.500E-02	(0.0000E+00)
..G012	4.000E-02	4.000E-02	4.000E-02	(0.0000E+00)
..G015	4.000E-02	4.000E-02	4.000E-02	(0.0000E+00)
..G026	3.600E-01	3.600E-01	3.600E-01	(0.0000E+00)
G018	1.800E-01	1.800E-01	1.800E-01	(0.0000E+00)
..G023	3.600E-01	3.600E-01	3.600E-01	(0.0000E+00)
G021	1.800E-01	1.800E-01	1.800E-01	(0.0000E+00)
..G031	3.600E-01	3.600E-01	3.600E-01	(0.0000E+00)
G028	9.000E-02	9.000E-02	9.000E-02	(0.0000E+00)
G010	3.765E-01	3.765E-01	3.765E-01	(0.0000E+00)
G008	2.458E-05	2.458E-05	2.458E-05	(0.0000E+00)
HLD_INPUT_REG_HIGH	1.800E-01	1.800E-01	1.800E-01	(0.0000E+00)

Execution Time : 0.3 secs

Input Filename : \\ot1pfp001\user$\data\mcnellesp\My Documents\HLD_INPUT_REG_HIGH.DPC.CAF

Database Name : \\ot1pfp001\user$\data\mcnellesp\My Documents\Comparator_Sub.rr

Date & Time : 07/14/16 13:11:57

Truncation Limit : 0.000e+00

Number Signals : 35

Maximum Table Width: 12

DPC Version 4.0

Updating database, source set to:DPC 07/14/16

Database successfully updated.

DFM Results:

Implicant 1 (8.9994E-02)
Clock 0 -1 (5.0000E-01)
DATA_IN_H_Prev H_In_High -1 (2.0000E-01)
HYSTD_Reset HYSTD_CLR_0 -1 (9.0000E-01)
SEU_H No_SEU_Reg_H 0 (9.9993E-01)

Implicant 2 (8.9994E-02)
DATA_IN_H_Prev H_In_High -1 (2.0000E-01)
HYSTD HYSTD_0 -1 (5.0000E-01)
HYSTD_Reset HYSTD_CLR_0 -1 (9.0000E-01)
SEU_H No_SEU_Reg_H 0 (9.9993E-01)

Implicant 3 (4.4997E-02)
Clock 1 -1 (5.0000E-01)
DATA_IN_H H_In_High -1 (2.0000E-01)
HYSTD HYSTD_1 -1 (5.0000E-01)
HYSTD_Reset HYSTD_CLR_0 -1 (9.0000E-01)
SEU_H No_SEU_Reg_H 0 (9.9993E-01)

Implicant 4 (3.5998E-02)
DATA_IN_H H_In_High -1 (2.0000E-01)
DATA_IN_H_Prev H_In_High -1 (2.0000E-01)
HYSTD_Reset HYSTD_CLR_0 -1 (9.0000E-01)
SEU_H No_SEU_Reg_H 0 (9.9993E-01)

Implicant 5 (6.5300E-06)
Clock 0 -1 (5.0000E-01)
DATA_IN_H_Prev H_In_0 -1 (2.0000E-01)
SEU_H SEU_Reg_H 0 (6.5300E-05)

Implicant 6 (6.5300E-06)
DATA_IN_H_Prev H_In_Low -1 (2.0000E-01)
HYSTD HYSTD_0 -1 (5.0000E-01)
SEU_H SEU_Reg_H 0 (6.5300E-05)

Implicant 7 (6.5300E-06)
HYSTD_Reset HYSTD_CLR_1 -1 (1.0000E-01)
SEU_H SEU_Reg_H 0 (6.5300E-05)

Implicant 8 (6.5300E-06)
Clock 0 -1 (5.0000E-01)
DATA_IN_H_Prev H_In_Low -1 (2.0000E-01)

SEU_H SEU_Reg_H 0 (6.5300E-05)

Implicant 9 (6.5300E-06)
DATA_IN_H_Prev H_In_0 -1 (2.0000E-01)
HYSTD HYSTD_0 -1 (5.0000E-01)
SEU_H SEU_Reg_H 0 (6.5300E-05)

Implicant 10 (3.2650E-06)
Clock 1 -1 (5.0000E-01)
DATA_IN_H H_In_Low -1 (2.0000E-01)
HYSTD HYSTD_1 -1 (5.0000E-01)
SEU_H SEU_Reg_H 0 (6.5300E-05)

Implicant 11 (3.2650E-06)
Clock 1 -1 (5.0000E-01)
DATA_IN_H H_In_0 -1 (2.0000E-01)
HYSTD HYSTD_1 -1 (5.0000E-01)
SEU_H SEU_Reg_H 0 (6.5300E-05)

Implicant 12 (2.6120E-06)
DATA_IN_H H_In_Low -1 (2.0000E-01)
DATA_IN_H_Prev H_In_Low -1 (2.0000E-01)
SEU_H SEU_Reg_H 0 (6.5300E-05)

Implicant 13 (2.6120E-06)
DATA_IN_H H_In_0 -1 (2.0000E-01)
DATA_IN_H_Prev H_In_0 -1 (2.0000E-01)
SEU_H SEU_Reg_H 0 (6.5300E-05)

Appendix II: List of Papers and Presentations

Conference Papers:

McNelles, P., and Lu, Comparison of Two Implementations of DFM for Analysing Reactor Control Systems. Proceedings of the SAI Computing Conference, London, UK, July 18-20, 2017; IEEE. (Submitted)

McNelles, P., and Lu, L. Design of a Tritium in Air Monitor Using Field Programmable Gate Arrays. Proceedings of the 2015 23rd International Conference on Nuclear Engineering, Chiba: ASME; 2015.

McNelles P, Zeng ZC, Renganathan G. Modelling of Field Programmable Gate Array Based Nuclear Power Plant Safety Systems Part I: Failure Mode and Effects Analysis. Proceedings Of the 7th International Conference on Modelling and Simulation in Nuclear Science and Engineering, Ottawa, Ontario, Canada: 2015.

McNelles P, Zeng ZC, Renganathan G. Modelling Radiation-Induced Failures in FPGAs Using the Dynamic Flowgraph Methodology. Transactions of the American Nuclear Society Winter Meeting, November 8-12, 2015. Washington, D.C.

McNelles P., and Lu, L. Dynamic Signal Compensation Using Field Programmable Gate Arrays, Proceedings of the 16th Pacific Basin Nuclear Conference, Vancouver, Canada, August 24-28, 2014.

McNelles P, Lu L. Lab-Scale Design, Demonstration and Safety Assessment of an FPGA-Based Post Accident Monitoring System for Westinghouse AP1000 Nuclear Power Plants. Proceedings of the 2014 22nd International Conference on Nuclear Engineering, Prague: ASME; 2014. doi:10.1115/ICONE22-30457.

McNelles P, Lu L. A Review of the Current State of FPGA Systems in Nuclear Instrumentation and Control. Proceedings of the 2013 21st International Conference on Nuclear Engineering, Chengdu, China: ASME; 2013. doi:10.1115/ICONE21-16819.

Journal Papers (Research Papers):

McNelles P, Zeng ZC, Renganathan G, Chirila M, Lu L. Failure Mode Taxonomy for Assessing the Reliability of Field Programmable Gate Array Based Instrumentation and Control Systems. Reliability Engineering and System Safety n.d. (Revision under Review)

McNelles P, Zeng ZC, Renganathan G, Lamarre G, Akl Y, Lu L. A Comparison of Fault Trees and the Dynamic Flowgraph Methodology for the Analysis of FPGA-based Safety Systems Part 1: Reactor Trip

Logic Loop Reliability Analysis. Reliability Engineering and System Safety 2016;153:135–50. Doi: http://dx.doi.org/10.1016/j.ress.2016.04.014

McNelles P, Lu L. Field Programmable Gate Array Reliability Analysis Using the Dynamic Flowgraph Methodology. Nuclear Engineering and Technology 2016.
doi: http://dx.doi.org/10.1016/j.net.2016.03.004.

Journal Papers (Technical Briefs):

McNelles P., and Lu, L. Design of a Tritium-In-Air-Monitor Using Field Programmable Gate Arrays. Journal of Nuclear Engineering and Radiation Science; 2016. doi: 10.1115/1.4033088

McNelles P, Lu L, Abi-Jaoude Ma-J. Dynamic Flowgraph Methodology Assessment of an FPGA-Based Postaccident Monitoring System for Westinghouse AP1000 Nuclear Power Plants. Journal of Nuclear Engineering and Radiation Science 2015;1:4 Pages. doi:10.1115/1.4029591.

Presentations:

The list represents additional presentations, not covered by the conference papers.

McNelles P, Zeng ZC, Renganathan G, Chirila M, and Lu L. Failure Mode Taxonomy for Assessing the Reliability of Field Programmable Gate Array Based Instrumentation and Control Systems. 9th International Workshop on the Applications of FPGAs in NPPs, Lyon, France, October 3-6, 2016.

McNelles P, Zeng Z C and Renganathan G. Dynamic Reliability Analysis of Radiation Induced Failure Modes in FPGA-Based Systems. 8th International Workshop on the Applications of FPGAs in NPPs, Shanghai, China, October 13-16, 2015.

McNelles P, Zeng Z C and Renganathan G. Failure Mode and Effects Analysis of FPGA-Based Nuclear Power Plant Safety Systems. 8th International Workshop on the Applications of FPGAs in NPPs, Shanghai, China, October 13-16, 2015.

Appendix III: Definitions

Many of the important definitions are given in the sections of the thesis report where the relevant topics are discussed. This appendix provides a compilation of all of those definitions, as well as several others that may aid the reader in understanding this thesis.

Application Specific Integrated Circuit (ASIC) [1,37]**:** An Application Specific Integrated Circuit (ASIC) is defined as an "Integrated Circuit designed for specific applications" [37]. Unlike FPGAs, ASICs are not configurable/reconfigurable after they are manufactured, as their functionality is custom designed/fabricated by the manufacturer at the time of construction [1].

Architecture [236]: Specific configuration of hardware and software elements in a system

Bitstream [1]**:** A contiguous sequence of bits (binary digits), representing a stream of data, serially transmitted continuously over a communications path. It is frequently used to describe the configuration data to be loaded into an FPGA

Channel [36]**:** An arrangement of components and modules as required to generate a single protective action signal when required by a generating station condition. A channel loses its identity where single protective action signals are combined

Combinatorial Logic [1]**:** In digital circuit theory, a concept in which two or more input states define one or more output states, where the resulting state or states are related by defined rules that are independent of previous states.

Common Cause Failure (CCF) [82]**:** Failure of two or more structures, systems and components due to a single specific event or cause.

Complexity [237]**:**

1. The degree to which a system's design or code is difficult to understand because of numerous components or relationships among components
2. The degree to which a system or component has a design or implementation that is difficult to understand and verify

Complex Programmable Logic Device (CPLD) [1]**:** A PLD that contains a number of 'macro cells' that are essentially the same as programmable array logic (PAL), and the means to interconnect them. A CPLD is sometimes referred to as a 'super-PAL'.

Component [82]**:** A discrete element of a system.

Note**:** A component may be hardware or software [36].

Crossbar [1]**:** A mechanism for connecting input wires and output wires in PLDs; for example, an $n \times m$ crossbar connects n different input wires to m output wires.

Cut Sets [98]**:** A list of failure events such that if they occur so does the Top Event.

Decision Table [119]: A related concept to a truth table. A decision able can be thought of as a more advanced truth table, and is a tabular representation of certain sets of "Conditions", "Actions", "Rules" and "Entries"

Division [36]**:** The designation applied to a given system or set of components that enables the establishment and maintenance of physical, electrical, and functional independence from other redundant sets of components. A division can have one or more channels

Dynamic Methodology [8]**:** A methodology that can account for the coupling between systems through explicit consideration of the time element in system evolution

Element [236]**:** Part of a subsystem comprising a single component or any group of components that performs one or more element safety functions

Emulation [237]**:** The use of a data processing system to imitate another data processing system, so that the imitating system accepts the same data, executes the same programs, and achieves the same results as the imitated system

Emulator [237]**:** A device, computer program, or system that accepts the same inputs and produces the same outputs as a given system

Failure Effect [129]**:** Consequence of a failure mode in terms of the operation, function or status of the item

Failure Mode [129]**:** Manner in which in item

Fault Tree Analysis [82]**:** A deductive technique that starts by hypothesizing and defining failure events and systematically deduces the events or combinations of events that caused the failure events to occur

Field Programmable Gate Array (FPGA) [1,37]**:** An integrated circuit that can be programmed in the field by the instrumentation and control (I&C) manufacturer. It includes programmable logic blocks (combinatorial and sequential), programmable interconnections between them and programmable blocks for inputs and/or outputs. The function is then defined by the I&C designer, not by the circuit manufacturer

Finite State Machine (FSM) [1]**:** An abstract model of a machine that has a primitive internal memory and a behaviour composed of a finite number of states and transitions between those states based on inputs, and output actions on other parts of the design. The behaviour of a finite state machine can be represented in a state transition diagram. Within a given state, for each input combination, there is only one possible transition from the present state to the new state. An application may contain multiple finite state machines interacting with each other through their inputs and outputs

Flip-Flop [1]**:** A bistable state circuit providing a single bit of memory. A flip-flop is usually controlled by one or two control signals and/or a gate or clock signal. The output often includes the complement as well as the normal output. As flip-flops are implemented electronically, they require power and ground connections.

FMEA [129]**:** A systematic procedure for the analysis of a system to identify the potential failure modes, their causes and effects on system performance. In the IEC 60812 standard, "system" is used as representation of hardware, software (with their interactions) or a process

Fussel-Vesely Importance Measure [221]**:** For a specific basic event, the fractional contribution to PSA results for all accident sequences containing that basic event. There exists both traditional and dynamic variants.

Hardware Description Language (HDL) [37]**:** Language Used to formally describe the functions and/or structure of an electronic component for documentation, simulation or synthesis

Hard Error[238]: An irreversible change in operation that is typically associated with permanent damage to one or more elements of a device or circuit

HDL Programmed Devices (HPD) [1,37]**:** An HDL Programmed Device (HPD) is defined as an "Integrated circuit configured (for NPP I&C systems) with Hardware Description Languages and related software tools" [37]. They contain arrays of logic elements that are connected by the end user to configure the device to perform the needed logic function [1]

Implicant/Implicant Set [98]**:** A combination of basic events (success or failure) which produces the top event. An Implicant/Implicant set can be considered as the MVL equivalent of a "Cut Set".

Integrated Circuit (IC) [237]**:** A small piece of semiconductive material that contains interconnected electronic elements

Intellectual Property (IP) Core [1]**:** A reusable unit of logic, cell design or chip layout design belonging to one party and licensed for use by another party. These are typically offered for ASIC and FPGA design components as netlists, but may be either soft or hard IP cores

Hard IP Core [1]**:** An IP core that is provided in the form of physical circuit layout; with a hard IP core, the end designer does not need to perform the synthesis and place and route process as would be required for a soft core

Soft IP Core [1]**:** An IP core that is in the form of a netlist or HDL. A soft IP core requires verification of a function following implementation (synthesis and/or place and route), unlike a hard IP core

Item [129]**:** Any part, component, device, sub-system, functional unit, equipment or system that can be individually considered

Logic Array [1]**:** An array made up of 'logic cells'. Interconnecting pathways are used to create the desired functionality

Logic Synthesis [1]**:** A process by which an abstract form of desired circuit behaviour, typically a register transfer level (RTL), is turned into a design implementation in terms of the resources (logic gates and other native blocks) of an actual hardware circuit. Common examples of this process include synthesis of HDLs, including very high speed integrated circuit hardware description language (VHDL) and Verilog. Some tools can generate bitstreams for PLDs such as PALs or FPGAs, while others target the creation of ASICs. Logic synthesis is one aspect of electronic design automation

Look-up Table (LUT) [1]: An electronic design block that replaces runtime computation with a simpler array indexing operation

Low Complexity E/E/PE Safety Related System [236]:
An Electrical/Electronic/Programmable Electronic (E/E/PE) safety-related system for which:

- The failure modes of each individual component are well-defined
- The behaviour of the system under fault conditions can be completely determined

Microprocessor [1]**:** A multipurpose, programmable device that accepts digital data as input, processes them according to instructions stored in its memory, and provides results as output. It incorporates the functions of a central processing unit (CPU) on a single chip. The semiconductor manufacturing process is now able to put multiple CPU cores onto a single chip

Minimal Cut Sets [98]**:** A list of minimal, necessary and sufficient conditions for the occurrence for the Top Event

Netlist (Gate Level) [37]**:** Description of an electronic component in terms of interconnections between its terminal elements

Place and Route (PAR) [1]**:** The step in integrated circuit or printed circuit board design that determines the physical locations of components, circuitry and logic elements, and the wiring paths required to connect the components

Pre-Developed Software (PDS) [37,56]**:** Software part that already exists, is available as a commercial or proprietary product, and is being considered for use

Prime Implicant/Prime Implicant Set [98]: A combination of basic events (success or failure) which is both necessary and sufficient to cause the Top Event. A Prime Implicant/Prime Implicant set can be thought of as the MVL equivalent of a "Minimal Cut Set".

Programmable array logic (PAL) [1,4]: A type of simple programmable logic device that consists of a programmable AND-plane followed by a fixed OR-plane

Programmable logic array (PLA) [1,4]: A type of simple programmable logic device that consists of two levels of logic, an AND-plane and an OR-plane, both of which are programmable

Programmable Logic Device (PLD) [1,37]**:**
A Programmable Logic Device (PLD) is defined as an "Integrated circuit that consists of logic elements with an interconnection pattern, parts of which are user programmable" [37]. HPDs began as simple Programmable Logic Devices (PLDs), with includes Programmable Logic Arrays (PLA) and Programmable Array Logic (PAL). Complex Programmable Logic Devices (CPLDs) are descended from PALs, and are basically combinations of multiple PALs onto a single chip with configurable interconnections [1]. FPGAs are not considered to be PLDs, as FPGAs are more complex, more powerful devices, however the exact determination between PLD and FPGA is not entirely defined [37].

Register Transfer Level (RTL) [37]**:** Synchronous parallel model of an electronic circuit, describing its behaviour by means of signals processed according to combinatorial logic and transferred between registers and clock pulses. The RTL model is typically written in HDL or generated out of HDL source code**.**

Risk Importance Measure [221]**:** A quantitative analysis to determine the importance of variations in equipment reliability to system risk and/or reliability

Safety Analysis [82]**:** Evaluation of the potential hazards associated with the conduct of an activity.

Sensitivity Analysis [82]**:** A quantitative examination of how the behaviour of a system varies with change, usually in the values of the governing parameters

Sequential Logic [1]**:** In digital circuit theory, this is a type of logic circuit whose output depends not only on the present value of its input signals but also on the past history of its inputs

Single Event Effect (SEE) [174]: Any measureable effect to a circuit due to an ion strike

Soft Error [239]**:** Storage element (memory cell, latch, or register) state change. No hardware damage and is correctable.

Structural Importance Measure [219]**:** A measure of the importance of a component from the system's topology (expressed by reliability block diagram, flow diagram etc.) and other structural information, without reference to actual probabilities.

System [82]**:** Comprised of several components, assembled in such a way to perform a specific (active) function

Synthesis [1] **:** A process by which an abstract expression of a digital circuit's behaviour at the RTL — for example, in an HDL — is translated into an equivalent description that is expressed in terms of the resources provided by the selected FPGA circuit. The circuit dependent description is called a netlist

Type 1 Interactions [8,11,12]**:** Dynamic Interactions between physical process variables (e.g. temperature, pressure, etc.) and the I&C systems that monitor and manage the process

Type 2 Interactions [8,11,12]: Dynamic Interactions within the I&C system itself due to the presence of software/firmware (i.e. multi-tasking and multiplexing)

Uncertainty Analysis [82]**:** An analysis to estimate the uncertainties and error bounds of the quantities involved in, and the results from, the solution of a problem

Verification and Validation (V&V) [237]**:** The process of determining whether the requirements for a system or component are complete and correct, the products of each development phase fulfill the requirements or conditions imposed by the previous phase, and the final system or component complies with specified requirements

Appendix IV: Permission Letters for Use of Copyright

Permission Letter From National Instruments

September 29, 2016

Phillip McNelles, B.Sc, M.Sc
University of Ontario Institution of Technology
2000 Simcoe Street North
Oshawa, Ontario, Canada
L1H 7K4

Re: Permission to Use Certain Copyrighted Materials of National Instruments Corporation ("NI")

Dear Mr. McNelles:

In response to your request to use certain copyrighted materials of NI, by this letter, NI grants you the limited right to reproduce:

1. White Paper "FPGA Fundamentals, Figure 2. The Different Parts of an FPGA", located at http://www.ni.com/white-paper/6983/en; and
2. Figure 10 and 11 of the newsletter "IP Corner: The LabVIEW Fixed-Point Data Type Part 1 Fixed Point 101", located at http://www.ni.com/newsletter/50303/en/

for inclusion in your thesis work.

1. You may use the NI Materials only for the purpose(s) described above (the "Purpose"). Please note that (unless expressly stated above in the Purpose) this is a one-time use right, and any additional uses by you will require the separate prior written authorization of NI.

2. Other than resizing the NI Materials as necessary for the Purpose, you may not revise, modify, or otherwise change the NI Materials in any manner without the prior written authorization of NI.

3. In no event may you use the NI Materials in a defamatory, libelous, or unlawful manner or in any manner that disparages NI.

4. You must retain all copyright and other proprietary notices of NI in the NI Materials. Without limiting the foregoing, you must include in the verification procedure, a copyright notice in prominent type and font indicating: (a) NI's copyright ownership of the NI Materials, and (b) that reproduction and redistribution of the NI Materials is courtesy of NI.

Please note that your use right is non exclusive and non transferable. NI reserves all rights and licenses not expressly granted you in this letter. **THE NI MATERIALS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, AND NO WARRANTIES (EITHER EXPRESS OR IMPLIED) ARE MADE WITH RESPECT TO THE NI MATERIALS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NON-INFRINGEMENT, OR ANY OTHER WARRANTIES THAT MAY ARISE FROM USAGE OF TRADE OR COURSE OF DEALING. YOU AGREE AND ACKNOWLEDGE THAT**

11500 N Mopac Expwy ⚲ Austin, TX 78759-3504 USA
Sales and Service: (512) 794-0100 ⚲ Other Calls: (512) 338-9119 ⚲ Fax: (512) 683-8411 ✉ info@ni.com www.ni.com

NATIONAL INSTRUMENTS

NI SHALL NOT BE LIABLE FOR ANY CLAIMS RESULTING FROM OR IN CONNECTION WITH ITS USE OF THE NI MATERIALS, AND YOU AGREE TO INDEMNIFY AND HOLD NI HARMLESS FROM ANY CLAIMS AGAINST OR EXPENSES OF NI ARISING OUT OF OR IN CONNECTION WITH ITS USE OF THE NI MATERIALS.

Sincerely,

Pete Smits
Sr. Intellectual Property Counsel
National Instruments Corporation

11500 N Mopac Expwy ▯ Austin, TX 78759-3504 USA
Sales and Service: (512) 794-0100 ▯ Other Calls: (512) 338-9119 ▯ Fax: (512) 683-8411 ▯ info@ni.com www.ni.com

Permission Letter From VTT

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Phillip McNelles <phillip.mcnelles@uoit.net>

Use of Copyright Question

info@vtt.fi <info@vtt.fi> Thu, Sep 29, 2016 at 8:11 AM
To: phillip.mcnelles@uoit.net

Hi

Thank you for getting in touch with VTT!

You have permission to include figures 1 (page 12), 3 and 4 (page 15) from the VTT report with title "The Current State of FPGA Technology in the Nuclear Domain" in your thesis document provided that references to original report is included and relevant.

Best regards

Jaana Nurmi
VTT Technical Research Centre of Finland Ltd.
Customer service
P.O. Box 1000, FI-02044 VTT, Finland
Fax. +358 20 722 7001
www.vtt.fi, info@vtt.fi

----Original Message----
From: Phillip McNelles <phillip.mcnelles@uoit.net>
Date: Tue, 27 Sep 2016 23:15:02 -0400
Subject: Use of Copyright Question
To: <info@vtt.fi>

Hi

My name is Phillip McNelles, and I am a graduate student at the University of Ontario Institute of Technology, in Oshawa, Ontario, Canada. I have been working on my PhD in the field of FPGA applications for nuclear power plant instrumentation and control systems. Over the course of my PhD research and publications, I have cited one of the VTT reports, entitled "The Current State of FPGA Technology in the Nuclear Domain", bu Jukka Ranta:

(http://www.vtt.fi/inf/pdf/technology/2012/T10.pdf).

Now that I am near the end of my PhD, I wanted to ask for permission to include a few of the figures in that report into my thesis document. Those figures include Figure 1 (page 12), and Figures 3 and 4 (page 15). I do not plan on altering these figures in any way (except for potentially re-sizing them), and would include the citation of the VTT report, as well as any additional information you would require. I feel these aforementioned details provide some excellent background information into FPGA technology, and would look more professional and presentanlr as is, then for me to try and re-draw them.

Please let me know if it is acceptable to you for me to include these figures in my thesis document.

[Quoted text hidden]

Permission Letter From IEEE

Copyright Clearance Center RightsLink®

Home | Create Account | Help | Live Chat

IEEE Requesting permission to reuse content from an IEEE publication

Title: Multiple-Valued Logic Trees: Meaning and Prime Implicants
Author: S. Garribba
Publication: Reliability, IEEE Transactions on
Publisher: IEEE
Date: Dec. 1985

LOGIN
If you're a copyright.com user, you can login to RightsLink using your copyright.com credentials. Already a RightsLink user or want to learn more?

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK | CLOSE WINDOW

Permission Letter From EPRI

Gmail Phillip McNelles <phillip.mcnelles@gmail.com>

Request for Copyright
1 message

copyright <copyright@epri.com> Wed, Dec 14, 2016 at 2:51 PM
To: "phillip.mcnelles@gmail.com" <phillip.mcnelles@gmail.com>

Dear Phillip McNells,

Thank you for contacting EPRI for use of our copyrighted material. Your request for copyright release for the material listed below is approved.

Best,

copyright@epri.com

From: Bulpitt, Carolyn
Sent: Wednesday, November 16, 2016 5:22 AM
To: copyright <copyright@epri.com>
Subject: FW: [External] Technical Information CRM:0800323

Good Morning!

Phil inquired about his copyright release, the tracking ID is #278. Is there any update?

Thanks,

Carolyn Bulpitt

EPRI Customer Assistance Center

800-313-3774 | direct: 972-556-6593

Connect with EPRI:

www.EPRI.com

www.EPRIJournal.com

Twitter | LinkedIn | Facebook

Together...Shaping the Future of Electricity

From: Phillip McNelles [mailto:phillip.mcnelles@gmail.com]
Sent: Tuesday, November 15, 2016 9:15 PM
To: Austin, Robert <raustin@epri.com>
Cc: Bulpitt, Carolyn <cbulpitt@epri.com>; Swezey, Rory <RSWEZEY@epri.com>
Subject: Re: [External] Technical Information CRM:0800323

Hi

Thank you for your response and the information you provided, I found it to be very useful.

There is one other matter you may be able to help me with. I put in a "Use of copyright" request earlier in the fall, to ask permission to use a few of the figures in this report in my thesis document, however I never received any response regarding it. I tried e-mailing the "copyright@epri.com" account, but I never received a response from there either.

The tracking ID was "#278".

Would you be able to help me out with this as well? I copied the original e-mail acknowledgement I received from EPRI below:

Thanks again, and have a good day.

-Phil

Dear Phillip McNelles,

Thank you for your request for Copyright Release. Your Tracking ID for this request is #278. Please keep this email for your records. Below is the detail of the completed form of the Copyright Release:

Obtaining Party Information
Company/Organization : University of Ontario Institute of Technology.
Contact Name : Phillip McNelles
Business Title : Graduate Student
Email : phillip.mcnelles@gmail.com
Phone : 4163713194
Address : 2000 Simcoe Street North
Oshawa, L1H 7K4
CA

Copyright Materials Details
Title/Description : Guidelines on the Use of Field Programmable Gate Arrays (FPGAs) in Nuclear Power Plant I&C Systems
Original Publish Year :2009
Intended Use: Hi I am a graduate student nearing the end of my PhD at the University of Ontario Institute of Technology (UOIT) located in Oshawa, Ontario, Canada. My PhD work has focused on certain aspects of the use of FPGA-based systems in Nuclear Power Plant systems, and I have cited the this EPRI report in some of my work. I wanted to ask for permission to use a few of the figures (listed in the box below) in my thesis document, as I think they help provide some important background information on FPGA
Material References: Figure 2-1, Page 2-2 Figure 2-4, Page 2-7 Figure 2-5, Page 2-8 Figure 7-1, Page 7-5

Authorized Signatory
Name : Phillip McNelles
Title : Graduate Student
Email : phillip.mcnelles@gmail.com

EPRI strives to respond to your request within 3-4 business days. You may contact Contact copyright@epri.com and reference you tracking number for any questions related to this request.

Regards,
EPRI