Failure Mode Modular De-Composition

R.P. Clark

A thesis submitted in partial fulfilment of the requirements for the University of Brighton for the degree of Doctor of Philosophy.

September 2013

Colophon

In short “Thanks every body”!

It has been a great privilege to spend several years visiting the Mathematics and Engineering departments of the University of Brighton, pushing me forward in clarity of self-expression, precision through mathematics, critical assessment and carefully crafted English: its members will always remain dear to me. My first debt of gratitude must go to my supervisors, Dr. A. Fish, Dr. C Garret and Professor J. Howse. They patiently provided the guidance, encouragement and advice necessary for me to proceed through the research, consolidation and write-up phases of the PhD program, to prepare and present three papers to conferences [19, 20, 21] and to complete and submit this thesis.

I owe a debt of thanks to Dr J. flower, my MSc project supervisor, who explained that the chapter in my project documentation postulating a modular form of FMEA—which had potential for making the process more efficient—was a concept worthy of being developed for a PhD and assisting me to present the chapter as a conference paper [22]. Further I thank her for encouraging me to apply for the PhD. I also wish to thank Alan Jones of Brighton College of Technology for taking a chance on someone with no ‘A’ levels and letting him start an HND in software Engineering in 1986. That more than anything changed my life and gave me fantastic opportunities.

I am deeply thankful to the directors of Energy Technology Control Ltd not only for funding this course, but providing training and work experience in the field of safety critical engineering and giving me Friday afternoons to pursue my studies. At Energy Technology Control, the following people gave encouragement, and validated the concepts for the ‘modular FMEA’ that I was developing, Martin Thirsk, Colin Talmay, Darren Legge and Hazel Anderson. These Engineers, whose whole careers have been focused on the safety critical electronic/computing area, gave valuable time to look at and comment on my FMMD proposals. Their comments gave me confidence that the methodology I was developing had potential practical applications and benefits. The environment and context of the work at Energy Technology Control Ltd was very useful for clarifying concepts relating to FMEA and safety; at least once a week there is a new practical case study arising and being discussed, be it, say, the observability of the effect of failures in an traditional amplifier configuration, or how a particular sensor could fail. The field of industrial burner control, is highly regulated and is rich with practical examples of safety measures built into hybrid digital/electronic systems. This has given me many

opportunities to apply the new methodology against 'real world' problems. These real world failure scenarios and their proposed solutions, were often detailed in requirements and design documentation, submitted in support of safety accreditation. I was glad to be tasked to produce many of these documents. Again I thank Energy Technology Control Ltd, for giving me these parallel tasks, which aided my studies.

I wish to thank my parents, Jennifer and Richard Clark. I hope that this work makes you proud.

Typeset in LaTeXFebruary 6, 2014.

Declaration

I declare that the research contained in this thesis, unless otherwise formally indicated within the text, is the original work of the author. The thesis has not been previously submitted to this or any other university for a degree, and does not incorporate any material already submitted for a degree.

Signed

Dated

Abstract

Failure Mode Modular De-Composition

The certification process of safety critical products for European and other international standards typically demand environmental stress, endurance and electro magnetic compatibility testing. Theoretical, or 'static testing' also a requirement. Failure Mode Effects Analysis (FMEA) is a tool used for static testing. FMEA is a bottom-up technique that aims to assess the effects of all component failure modes in a system. Its use is traditionally limited to hardware systems. With the growing complexity of modern electronics traditional FMEA is suffering from state explosion and re-use of analysis problems. Also with the now ubiquitous use of microcontrollers in smart instruments and control systems, software is increasingly being seen as a 'missing factor' for FMEA.

This thesis presents a new modular variant of FMEA, Failure Mode Modular Decomposition (FMMD). FMMD has been designed to integrate mechanical/electronic and software failure models, by treating them all as components in terms of their failure modes. For instance, software functions, electronic and mechanical components can all be assigned sets of failure modes. FMMD builds failure mode models from the bottom-up by incrementally analysing functional groupings of components, using the results of analysis to create higher level derived components, which in turn can be used to build functional groupings. In this way a hierarchical failure mode model is built. Software functions are treated as components by FMMD and can thus be incorporated seamlessly into the failure mode hierarchical model. A selection of examples, electronic circuits and hardware/software hybrids are analysed using this new methodology. The results of these analyses are then discussed from the perspective of safety critical application. Performance in terms of test efficiency is greatly improved by FMMD and the examples analysed and theoretical models are used to demonstrate this.

This thesis presents a methodology that mitigates the state explosion problems of FMEA; provides integrated hardware and software failure mode models; facilitates multiple failure mode analysis; encourages re-use of analysis work and can be used to produce traditional format FMEA reports.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Increasingly society relies on automation in everyday life. Many automated systems have the potential to cause harm or even death should they fail. Safety assessment and certification is now required for almost all potentially dangerous equipment. As part of the assessment/certification process, typically a battery of tests is applied, examining features such as resistance to extremes of environment, Electro Magnetic Compatibility (EMC), endurance regimes and static testing. Static testing is at the theoretical, or design level, and involves looking at failure scenarios and trying to predict how systems would react. This thesis deals with one area of static testing, that of Failure Mode Effects Analysis (FMEA) [16], a commonly used technique that is a legal requirement for a wide range of equipment certification.

The ability to assess the safety of machinery has been a concern since the dawn of the industrial age [31, 52]. The philosophy behind safety measures has progressed over time and by World War Two concepts such as 'no single component failure should cause a dangerous system failure' [12] emerged [55][Ch.13]. Concepts such as these allow objective criteria of safety assessment. The 'no single failure' concept can be extended to double or even multiple failures being unacceptable as the cause of dangerous states. The concept of a double failure causing a dangerous condition being forbidden can be found in the legally binding European standard EN298[1] which came into force in 2006 [14]. More sophisticated statistically based standards, i.e EN61508 [95] and variants thereof, are based on statistical thresholds for the frequency of dangerous failures. For instance, acceptable maximum numbers of dangerous failures per billion hours of operation could be stated. Orders of failure rates can then be broadly categorised into Safety Integrity Levels (SIL) [89]. So for a maximum of 10 potentially dangerous failures per billion hours of operation a SIL level of 4 is assigned, for 100 a SIL level of 3, and so on in powers of ten. If SIL ratings can be determined, they can be matched against given risks. The more dangerous the consequences of failure the higher the SIL rating. A band-saw with one operative may require a SIL rating of 1, but systems such as nuclear power-stations or air-liners, with far greater consequences on dangerous failure, may require a SIL ratings of 4.

All of these risk assessment techniques are based on variations of Failure Mode Effect Analysis (FMEA), which has its roots in the 1940's mass production industry and was designed to save large companies money by

[1] EN298:2003 became a legal requirement for all new forced draft industrial burner controllers in 2006 within the European Union.

prioritising the most financially draining problems in a product. The FMEA of the 1940's has been refined and extended into four main variants. This thesis describes the refinements and additions made to FMEA to tailor them for military or statistically biased use. It then reveals common flaws which make them unsuitable for the higher safety requirements of the 21st century. Problems with state explosion in failure mode reasoning and the current difficulties of integrating software and hardware failure mode models [42] are the most obvious of these. These four current methodologies are described in chapter 2 and critically assessed in chapter 3. In chapter 4, a new methodology is proposed which addresses the state explosion problem and using contract programmed software, allows the modelling of integrated software/electrical systems. This is followed by two chapters showing examples of the new modular FMEA analysis technique (Failure Mode Modular De-Composition, FMMD) firstly looking at a variety of common electronic circuits and then at electronic/software hybrid systems.

1.1 Motivation

The motivation for this study came from two sources, one academic (the author's Software Engineering MSc project) and the other practical (the author is a practising embedded software engineer working with FMEA on safety critical burner systems).

MSc Project: Euler/Spider diagram Editor. The author had recently completed an MSc and the project was to create an Euler/Spider Diagram [47] editor in Java. This editor allowed the user to draw Euler/Spider diagrams, and could then represent these as abstract—i.e. mathematical—definitions. The primary motive for writing the Spider diagram editor was to provide an alternative to formal languages for software specification. An added attraction for using spider diagrams was that they could be used in proving logic and theorems [33, 32] in an intuitive way. Because of the author's daily work exposure to FMEA, it was natural to think of ways to apply formal languages and spider diagrams to failure mode analysis.

European Safety Requirements increase in scope and complexity. At work—which consisted of designing, testing, building and writing embedded 'C' and assembly language code for safety critical industrial burners—the design team was faced with a new and daunting requirement. Conformance to the latest European standard, EN298 [14]. It appeared to ask for the impossible: not only did it require the usual safety measures (self-checking of ROM and RAM, watchdog processors with separate clock sources, EMC testing and the triple fail safe control of valves), it had one new clause in it that had far reaching consequences. It stated that in the event of a failure, where the controller had gone into a 'lockout state'— a state where the controller applies all possible safety measures to stop fuel entering the burner—it was not permitted to become dangerous should another fault occur. In short this meant dealing with double failures. Any of the components that could, in failing, create a dangerous state were already documented and approved using failure mode effects analysis (FMEA). This new requirement effectively meant that single and double component failures were now required to be analysed [14][9.1.5]. This, from a state explosion problem alone, meant that it was going to be virtually impossible to perform. To compound the problem, FMEA has a deficiency of repeated work, as each component failure is typically represented by one line or entry in a spreadsheet [62]; analysis on repeated sections of

circuitry (for instance repeated $4 \rightarrow 20mA$ outputs on a PCB) meant that analysis of identical circuitry was performed many times.

1.1.1 Modularising/De-Composing FMEA: Initial concepts.

In the field of digital signal processing there is an algorithm that revolutionised access to frequency analysis of digital samples called the Fast Fourier Transform (FFT) [23]. This took the Discrete Fourier Transform (DFT), and applied de-composition to its mesh of (often repeated) complex number calculations [99][Ch.8]. By doing this it broke the computing order of complexity down from having a polynomial to logarithmic order [83][pp.401-3]. The author wondered if this thinking could be applied to the state explosion problems encountered in FMEA. The authors reasoning was that if the problem were analysed in small modules, from the bottom-up following the FFT example, checking for all double failure scenarios could have been applied. Once these first modules were analysed—now called functional groupings—the symptoms of failure could be determined for them. Using the symptoms of failure, these modules could be treated as components in their own right—or derived components—and used to build higher level functional groupings. Higher and higher levels of functional groupings could be built until a hierarchy representing a failure mode model for the complete system had been created. Double simultaneous failure mode checking can be applied as the number of components in each functional grouping is typically small; state explosion problems are thus avoided. If double checking is applied all the way up the hierarchy, all possible double simultaneous failures in a system can be guaranteed to have been considered. This means, as a fortunate by-product, that many multiple as well as double failures would be analysed, but because failure modes are traceable from the base components to the top level—or system—failure modes, these relationships can be held in a traversable data structure. If held in a traversable data structure automated methods can be applied to search for all the combinations of multiple failure modes throughout the model being analysed. Because of this, it will not always be necessary to apply double checking at all higher levels in the analysis hierarchy, to achieve complete double failure coverage. The points at which it is possible to relax double failure checking can be verified automatically by traversing the failure mode model.

1.1.2 Initial direction: Application of Spider diagrams to FMEA.

Because, Euler/Spider Diagrams [47] could be used to model failure modes in components it was thought that a diagrammatic notation would be more user friendly than using formal logic. For an FMEA Spider diagram, contours represent failure modes, and the Spider diagram 'existential points' represent instances of failure modes. Overlapping contours represent multiple failure modes. By drawing a spider collecting existential points, a common failure symptom could be determined and from this a new diagram generated automatically to represent the derived component. Each spider represented a derived failure mode. The act of collecting common symptoms by drawing spiders meant that the analyst was forced to associate one component failure mode with one symptom/derived failure mode of failure. These concepts were presented at the "Euler 2004" [22] conference held at the University of Brighton. This defined the concepts for modularising FMEA using the formal visual notations from Spider diagrams. This lead to work on rapidly calculating available zones in Euler diagrams [21, 85]. The spider diagram notation was useful in defining the concepts and initial ideas, but a

more traditional 'spreadsheet' format has been used for the analysis stages of the new methodology. Euler diagrams have been used later in the thesis to describe the containment relationships of derived components when building hierarchical analysis models with the modularised variant of FMEA that this thesis proposes and defends.

1.2 Objectives of the thesis.

The primary objective of the work performed for this thesis is to present a new modularised variant of FMEA which solves the problems of:

- State Explosion,
- Multiple failure mode modelling,
- Re-usability of pre-analysed modules,
- Inclusion of software in failure mode modelling.

To support this, worked examples using the new methodology were created and the work published and presented to IET safety conferences. The development of FMMD, starting with a critique of FMEA and a "wish-list" for a better methodology, was presented to the IET System safety conference in 2011, [19]. FMEA, currently cannot integrate software models into its hardware failure mode models [91, 90, 40, 74], but FMMD can use the existing structure of functional software, in conjunction with contract programming to model software; this concept was presented to the IET System safety conference in 2012 [20].

Overview of the thesis. Chapter 2 examines the current state of FMEA based methodologies, Chapter 3 examines the benefits and drawbacks of these methodologies and proposes a detailed wish list for an ideal FMEA technique. Chapter 4 proposes Failure Mode Modular de-composition (FMMD)—a modularised variant of FMEA designed to address the points in the detailed wish list. Chapter 5 provides worked examples using selected electronic circuits. Chapter 6 gives two examples of integrated software and electronic systems analysed using FMMD. Metrics and evaluation, along with an example showing double simultaneous failure analysis, are provided in Chapter 7, with a conclusion and further work in Chapter 8.

Chapter 2

Failure Mode Effect Analysis

The generic and statistical European Safety Standard, EN61508:6[95][B.6.6] describes Failure Mode Effect Analysis (FMEA) as:

> "To analyse a system design, by examining all possible sources of failure of a system's components and determining the effects of these failures on the behaviour and safety of the system."

Introduction

This chapter introduces Failure Mode Effect Analysis (FMEA). It starts with a generic conceptual overview of the process. It then looks at the stages of the FMEA process in greater detail, starting with how to determine the failure modes associated with components. Two common electrical components, the resistor and the operational amplifier are examined in the context of two sources of information that define failure modes. To introduce the concept of FMEA, a simple example is given, using a hypothetical four to twenty milli-amp ($4 \rightarrow 20mA$) reader. The four main current FMEA variants are described along with the concepts that underlie the usage and philosophy of FMEA. The overall process of FMEA is then reviewed and modelled using UML. By using UML the entities needed to implement FMEA are defined. The act of defining relationships between the data objects in FMEA raises questions about the nature of the process and allows analysis of its strengths and weaknesses.

2.1 FMEA Basic concept.

FMEA [53][pp.341-344] is widely used, and proof of its use is a legal requirement for a large proportion of safety critical products sold in the European Union. The acronym FMEA can be expanded as follows:

- **F - Failures of given component,** Consider a particular component in a system;
- **M - Failure Mode,** Choose a particular failure mode of this component;
- **E - Effects,** Determine the effects this failure mode will cause;
- **A - Analysis,** Analyse how much impact this symptom will have on the environment/operators/the system itself.

FMEA is a broad term; it could mean anything from an informal check on how failures could affect some equipment in a brain-storming session to formal submission as part of safety critical certification. FMEA is a manual, time intensive process. To reduce the amount of manual work performed, software packages [101, 100] and analysis strategies have been developed [81, 76]. FMEA is always performed in context. That is, the equipment is always analysed for a particular purpose and in a given environment. An 'O' ring for instance can fail by leaking but if fitted to a water seal on a garden hose, the system level failure would be a slight leak at the tap. Applied to the rocket engine on a space shuttle an 'O' ring failure could cause a catastrophic fire and destruction of the spacecraft and occupants [105]. At a lower level, consider a resistor and capacitor forming a potential divider to ground. This could be considered a low pass filter in some electrical environments [78], but for fixed frequencies the same circuit could be used as a phase changer [97][p.114]. The failure modes of the latter, could be 'no signal' and 'all pass', but when used as a phase changer, would be 'no signal' and 'no phase' change. The actual failure modes for a 'group of components', are therefore defined by the function that they perform.

2.2 FMEA Process

The initial stage of the FMEA process is with the basic, or starting components. These components are the sort bought in or considered as pre-assembled modules. These are termed 'base components'; they are considered "atomic" i.e. they are not broken down further. The first requirement for a base component is to define the ways in which it can fail, this relationship is shown, using UML, in figure 2.1.

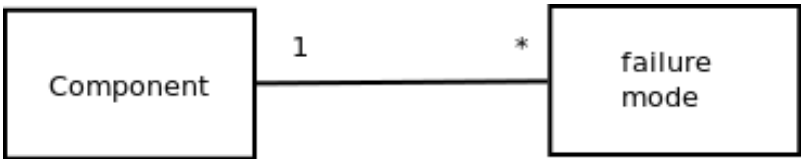


Figure 2.1: Base Component to Failure Modes relationship UML diagram

The next stage is analysis, that is reasoning applied to the system in the event of a given failure mode. To analyse how a failure mode, after considering its effect on other components in the system, will translate to a system level symptom/failure. The result of FMEA is to determine system level failures, or symptoms for each given component failure mode. In practise, each entry of an FMEA analysis of a base component failure mode would typically be one line in a spreadsheet. The analysis to symptom relationship is generally one-to-one, however here (see figure 2.2), allowance is made for the possibility of more than one failure symptom.

Figure 2.2 defines the data relationships for FMEA. This model is later extended in the conclusion of this chapter.

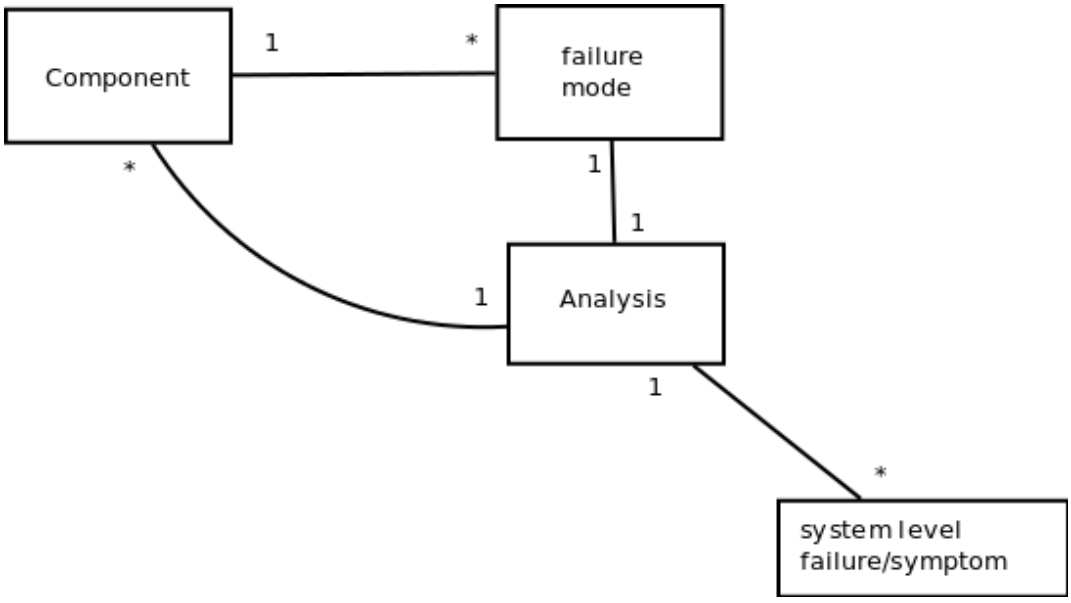


Figure 2.2: FMEA analyis entry data relationships

2.3 Determining the failure modes of base components

In order to apply any form of FMEA the ways in which the base components[1] can fail must be clearly defined. In practice, this part of the process is guided by the particular standard which is being conformed to. Standards may differ in their definitions for the failure modes of base components. The reasons for these differences are examined below using two example components. Typically, when choosing components for a design, engineers will look at manufacturers' data sheets which describe functionality, physical dimensions, environmental ranges and tolerances etc. . It is rare for a data sheet to list failure modes. Data sheets after all are a sales tool as well as being a usage guide and technical description. However, 'reading between the lines' or noting what is not stated, can in some cases indicate how a component could fail/misbehave.

How components could fail internally is not of interest to an FMEA investigation. The FMEA investigator needs to know what failure behaviour a component could exhibit. A large body of literature exists giving guidance for the determination of component failure modes. An interesting discussion on semi-conductor failure modes may be found in [108][Ch.44]. For this study FMD-91 [24] and the gas burner standard EN298 [14] are examined. In EN298 failure modes for most generic component types are listed, or if not listed, are determined using a procedure: typically of the form of examining scenarios such as 'all pins open' and then 'all adjacent pins shorted' [14][A.1 note e].

FMD-91 [24] is a reference document released into the public domain by the United States DOD and describes 'failures' of common electronic components, with percentage statistics for each failure. FMD-91 entries include general descriptions of internal failures alongside failure modes of use to an FMEA investigation. FMD-91 entries need, in some cases, some interpretation to be mapped to a clear set of component failure modes suitable for use in FMEA. A third document, MIL-1991 [26] provides overall reliability statistics for component types, but does not detail specific failure modes. Using MIL1991 in conjunction with FMD-91 statistics can be determined

[1] A good introduction to hardware and software failure modes may be found in [96][pp.114-124].

for the failure modes of component types. As these documents are now a little old, the results from them can be on the conservative side. A FIT[2] value for a micro-processor may be determined at around 100 using these documents for instance, but FIT claims for modern integrated micro-controllers are typically less than five [65]. The FMEA variant[3] used for European standard EN61508 [95] requires statistics for Mean Time to Failure (MTTF) for all base component failure modes.

2.4 Determining the failure modes of Components.

The starting points in the FMEA process are the failure modes of the base components. In order to define FMEA, a discussion on how these failure modes are defined and their relationship to particular standards is presented below. Two common electrical components are used as examples, and examined against two sources of failure mode information. Failure mode definitions for a given generic component may not always agree. The reasons why, some failure modes can be found in one source, but not in the others and vice versa, are discussed. Finally, the failure modes determined from the FMD-91 [24] reference source and from the guidelines of the European burner standard EN298 [14], are compared and contrasted.

[2]Failure rates measured per 10^9 hours of operation are known as Failure in Time (FIT) values.

[3]EN61508 (and related standards) are based on the FMEA variant Failure Mode Effects and Diagnostic Analysis (FMEDA)

2.4.1 Failure mode determination for generic resistor.

Resistor failure modes according to FMD-91. FMD-91[24][3-178] lists many types of resistor and lists many possible failure causes, for instance for **Resistor, Fixed, Film** the following failure causes are given:

- Opened 52% ,
- Drift 31.8% ,
- Film Imperfections 5.1% ,
- Substrate defects 5.1% ,
- Shorted 3.9% ,
- Lead damage 1.9% .

To make this useful for FMEA each failure cause must be mapped to a symptomatic failure mode descriptor [4] as listed below:

- Opened 52% $\mapsto$ OPENED,
- Drift 31.8% $\mapsto$ DRIFT,
- Film Imperfections 5.1% $\mapsto$ OPEN,
- Substrate defects 5.1% $\mapsto$ OPEN,
- Shorted 3.9% $\mapsto$ SHORT,
- Lead damage 1.9% $\mapsto$ OPEN.

Note, that the main cause of resistor value drift is overloading. This is borne out in the FMD-91 [24] entry for a resistor network where the failure modes do not include drift. If it is ensured that resistors will not be exposed to overload conditions, the probability of drift (sometimes called parameter change) is significantly reduced, enough for some standards to exclude it [14, 15].

Resistor failure modes according to EN298. EN298, the European gas burner safety standard, tends to give failure modes that are more directly usable for performing FMEA than FMD-91. The certification process for EN298 requires that a full FMEA be undertaken, examining all failure modes of all electronic components [14][11.2 5]. Annex A of EN298, prescribes failure modes for common components and guidance on determining sets of failure modes for complex components (i.e. integrated circuits). EN298 [14][Annex A] (for most types of resistor) only requires that the failure mode OPEN be considered for FMEA analysis. For resistor types not specifically listed in EN298, the failure modes are considered to be either OPEN or SHORT. The reason that parameter change is not considered for resistors chosen for an EN298 compliant system, is that they must be *downrated* during the design process. That is to say the power and voltage ratings of components must be calculated for maximum possible exposure, with a 40% margin of error. This drastically reduces

[4] The symptomatic descriptors chosen are based on experience and are not unique.

the probability that the resistors will be overloaded, and thus subject to drift/parameter change. Clearly the assumed failure modes of base components represent a fundamental limit of resolution in any failure analysis methodology.

2.4.1.1 Resistor Failure Modes

The difference in resistor failure modes between FMD-91 and EN298 is that FMD-91 would include the failure mode DRIFT. EN298 does not include this, mainly because it imposes circuit design constraints that effectively side step that problem. For this study the conservative view from EN298, but restrictive view from FMD-91 (i.e. no DRIFT) is taken, and the failure modes for a generic resistor taken to be both OPEN and SHORT. The function fm is used to return a set of failure modes, i.e.

$$fm(R) = \{OPEN, SHORT\}.$$

2.4.2 Failure modes determination for a generic operational amplifier

The operational amplifier (op-amp) is very widely used in nearly all fields of modern analogue electronics. Only one of two sources of information on base component failure modes being compared has an entry specific to operational amplifiers (FMD-91). EN298 does not specifically define the failure modes of op-amps but instead has a procedure for determining the failure modes of components types not specifically listed. Operational amplifiers are typically packaged in dual or quad configurations—meaning that a chip will typically contain two or four amplifiers. The failure modes determined from the FMD-91 entries are presented and then the failure mode determination procedure of EN298 is applied to a typical op-amp designed for instrumentation and measurement, the dual packaged version of the LM358 [66] (see figure 2.3). The results from both sources of failure mode definition are then compared.

Failure Modes of an Op-Amp according to FMD-91. For Op-Amp failures modes, FMD-91[24]3-116] states,

- Degraded Output 50% Low Slew rate - poor die attach
- No Operation - overstress 31.3%
- Shorted inputs (labelled V_+ to V_-), overstress, resistive short in amplifier 12.5%
- Opened input (labelled V_+) open 6.3%

These are mostly internal causes of failure, more of interest to the component manufacturer than a test engineer looking for symptoms of failure. These failure causes within the Op-Amp need to be translated to symptomatic failure modes. Each failure cause is examined in turn, and mapped to potential failure modes suitable for use in FMEA investigations.

Op-Amp failure cause: Poor Die attach. The symptom for this is given as a low slew rate. Slew rate for a circuit/component is the maximum rate at which it can change an output voltage level (i.e. $\frac{\delta V}{\delta t}$). A low

slew rate will mean that the op-amp will not react quickly to changes on its input terminals. This is a failure symptom that may not be of concern in a slow responding system like an instrumentation amplifier. However, where higher frequencies are being processed, a signal may be lost entirely. This failure cause can be mapped to a symptomatic failure mode called LOW_SLEW.

No Operation - over stress. Here the OP-Amp has been damaged, and the output may be held HIGH or LOW, or may be effectively tri-stated, i.e. not able to drive circuitry along the next stages of the signal path: this failure mode is termed NOOP (no Operation). This failure cause thus maps to three failure modes, LOW, $HIGH$, $NOOP$.

Shorted inputs: V_+ to V_-. Due to the high intrinsic gain of an op-amp, and the effect of offset currents, this will force the output HIGH or LOW. This failure cause maps to $HIGH$ or LOW.

Open input: V_+. This failure cause will mean that the minus input will have the very high gain of the Op-Amp applied to it, and the output will be forced HIGH or LOW. This failure cause maps to $HIGH$ or LOW [5].

Collecting Op-Amp failure modes from FMD-91. An Op-Amp's failure mode behaviour, under FMD-91 definitions will have the following failure modes:

$$fm(OpAmp) = \{HIGH, LOW, NOOP, LOW\_SLEW\}. \tag{2.1}$$

Failure Modes of an Op-Amp according to EN298. EN298 does not specifically define op-amp failure modes; these can be determined by following a procedure for 'integrated circuits' outlined in annex A [14][A.1 note e]. This demands that all open connections, and shorts between adjacent pins be considered as failure scenarios. In table 2.1 these failure scenarios on the dual packaged $LM358$ [66] are examined and from this its failure modes are determined. Collating the op-amp failure modes from table 2.1, the same failure modes from FMD-91 are obtained—listed in equation 2.1—except for LOW_SLEW.

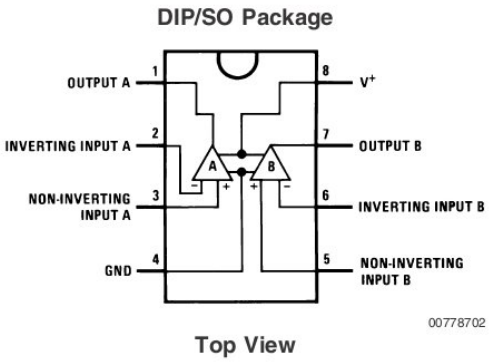


Figure 2.3: Pinout for an LM358 dual Op-Amp

[5] No failure mode for open input V_- was listed in this FMD-91 entry [24][3-116].

Table 2.1: LM358: EN298 Open and shorted pin failure symptom determination technique

Failure cause	**Amplifier Effect**	**FMEA component Failure Mode**
FS1: PIN 1 OPEN	A output open	$NOOP_A$
FS2: PIN 2 OPEN	A-input disconnected, infinite gain on A+input	LOW_A or $HIGH_A$
FS3: PIN 3 OPEN	A+input disconnected, infinite gain on A-input	LOW_A or $HIGH_A$
FS4: PIN 4 OPEN	power to chip (ground) disconnected	$NOOP_A$ and $NOOP_B$
FS5: PIN 5 OPEN	B+input disconnected, infinite gain on B-input	LOW_B or $HIGH_B$
FS6: PIN 6 OPEN FS6:	B-input disconnected, infinite gain on B+input	LOW_B or $HIGH_B$
FS7: PIN 7 OPEN	B output open	$NOOP_B$
FS8: PIN 8 OPEN FS8:	power to chip (V+ supply) disconnected	$NOOP_A$ and $NOOP_B$
FS9: PIN 1 $\stackrel{short}{\longrightarrow}$ PIN 2	A -ve 100% Feed back, unity gain	LOW_A
FS10: PIN 2 $\stackrel{short}{\longrightarrow}$ PIN 3	A inputs shorted, output controlled by internal offset	LOW_A or $HIGH_A$
FS11: PIN 3 $\stackrel{short}{\longrightarrow}$ PIN 4	A + input held to ground	LOW_A or $HIGH_A$
FS12: PIN 5 $\stackrel{short}{\longrightarrow}$ PIN 6	B inputs shorted, output controlled by internal offset	LOW_B or $HIGH_B$
FS13: PIN 6 $\stackrel{short}{\longrightarrow}$ PIN 7	B -ve 100% Feed back, low gain	LOW_B
FS14: PIN 7 $\stackrel{short}{\longrightarrow}$ PIN 8	B output held high	$HIGH_B$

2.4.2.1 Failure modes of an Op-Amp

For the purpose of the examples to follow in this document, op-amp's are assigned the following failure modes:

$$fm(OPAMP) = \{LOW, HIGH, NOOP, LOW\_SLEW\}.$$

2.4.3 Comparing the component failure mode sources: EN298 vs FMD-91

The EN298 pinouts failure mode technique cannot reveal failure modes due to internal failures, and that is why it misses LOW_SLEW. The FMD-91 entries for op-amps are not directly usable as component failure modes in FMEA and require interpretation. However, once a failure mode determination has been carried out, the model can be re-used throughout the FMEA process.

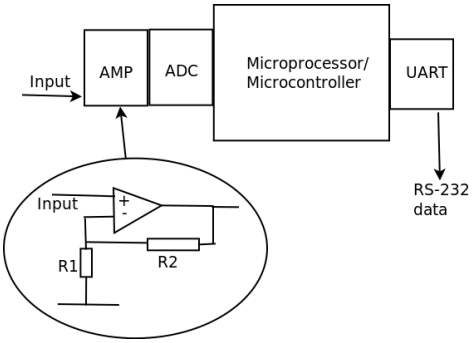


Figure 2.4: System diagram of a milli-volt reader, showing an expanded circuit diagram for the component of interest.

2.5 FMEA worked example: milli-volt reader.

FMEA is a bottom-up procedure which starts with the failure modes of the low level components of a system. An example analysis will serve to demonstrate it in practice. Consider a system of a simple milli-volt reader, consisting of instrumentation amplifiers connected to a micro-processor that reports its readings via RS-232.

2.5.1 FMEA Example: Milli-volt reader

Undertaking an FMEA on the milli-volt reader to consider how one of its resistors failing could affect it and choosing the resistor R1 in the OP-AMP gain circuitry:

- **F - Failures of given component** The resistor (R1) could fail by going OPEN or SHORT (EN298 definition),
- **M - Failure Mode** Consider the component failure mode SHORT,
- **E - Effects** This will drive the minus input LOW causing a HIGH OUTPUT/READING,
- **A - Analysis** The reading will be out of the normal range, i.e. will have an erroneous milli-volt reading.

The analysis above has given a result for one single component failure mode. A complete FMEA report, would have to contain an entry for each failure mode of all the components in the system under investigation. In theory it would be necessary to look at the failure mode in relation to the entire circuit. Intuition has been used to determine the probable effect of this failure mode. For instance it has been assumed that the resistor R1 going SHORT will not affect the ADC, the Microprocessor or the UART. The base component failure mode R1 SHORT has been examined and failure reasoning applied, along a heuristically determined signal path, to find a putative system level symptom. That is R1 going SHORT is expected to just give an out of range value that can be read by the ADC and reported correctly by the software. Potential side effects of this failure mode may not have been factored. To put this in more general terms, this failure mode has not been examined against all other components in the system, only those expected on the signal path. Examining the failure mode R1 SHORT against all component in this system, would be a more rigorous and complete approach in looking for system failures. FMEA where each failure mode is compared against all other components is termed exhaustive

FMEA (XFMEA). An indicator of the vagueness of not performing XFMEA, in terms of failure outcome, is shown in the UML relationship in figure 2.2 giving a one to many mapping for a failure mode and its system level symptom.

2.6 Theoretical Concepts in FMEA

In this section some fundamental concepts and underlying philosophies of FMEA are examined.

Failure modes of a component and mutual exclusivity. It is desirable that the failure modes for a component are mutually exclusive, were a component able to fail in several ways at the same time, this would complicate analysis. It would mean having to consider combinations of internal component failures as separate failure modes. This concept is discussed in sections 4.3.1.1 and 7.3. In general, failure modes for simple components are mutually exclusive, but large and complex components (such as integrated circuits), especially where they contain separate modules, could have non mutually exclusive failure modes and these need special handling, see section 7.7.

The signal path. Most electronic systems are used to process a signal: with signal processing there is usually a clear path from the signal coming into the system, it being processed in some way, and a resultant effect on an output or control signal. That is, there is an input, some processing and an output. In electronics this could be termed a sensor, processing and actuator model. In software this would be termed afferent, transform and efferent data flow. For the purpose of FMEA, the signal path is defined by the components and connections used to process the signal. Some circuits have feedback loops or even circular signal paths, but it is normal for a signal path to exist. An FMEA investigation will often take the component failure mode and examine its effect along this path, in the direction of the signal, echoing diagnostic/fault finding methods [36, 58]. When fault finding, the signal path is followed, checking for correct behaviour along it: when something out of place is found, the circuit behaviour is measured in finer granularity, until a faulty component or module [36] is identified. With this style of fault finding, because it is based on experiment, hopping from module to module eliminating working ones, until a failure is found [58], is efficient in terms of concentrating effort. The rationale and work-culture of those tasked to perform FMEA are generally personnel who have performed fault finding [80][p.97].

FMEA is a theoretical discipline. It would be very unusual to build a circuit and then simulate component failure modes. This would be time consuming as it would involve altering/building a circuit for each component failure mode in the system[6]. It is not possible, as with fault finding, to verify modules along the signal path for correct behaviour and eliminate them from the investigation. FMEA is a 'thought experiment', not actual experiment. With FMEA there is a need to be more thorough in the consideration of the effects a failure mode may have on the other components in a system, than with fault finding. The question is by how much. Too much and the task becomes impossible due to time/labour constraints. Too little and the analysis could become meaningless, because it could miss potential system failures. For a more complete analysis, the strategy of examining each component failure mode along the complete signal path, forwards and backwards from the

[6]Building circuit simulations and simulating component failure modes would be a very time consuming process and might only be performed as a final-stage of accident investigation, where the cause is required to be proven.

placement of the component exhibiting the failure mode under investigation, could be applied. Is following the effects of a failure mode *only* through the components along the signal path acceptable? This could easily ignore side effects; this leads onto the idea of looking at a failure mode's effects on all other components in the system. In practise, a compromise is made between the amount of time/money that can be spent on analysis relative to the criticality of the project. Metrics for measuring the amount of work to undertake for FMEA are examined in section 2.6.2.

Failure Modes and the signal path. In general a component failure mode in an electronic circuit will change the circuit topology. For a single failure this effect may cause additional complications for the analyst. For multiple failures this means that the analyst will have to deal with altered—or changed circuit topologies—of the electronic circuit for each analysis.

Single component failure mode to system failure relation. FMEA, due to its inductive bottom-up approach, is good at mapping potential single component failures to system level faults/events. The concept of the unacceptability of a single component failure causing a system failure is an important and easily understood measurement of safety. Statistics for single failures are easy to calculate because Mean Time to Failure (MTTF) statistics [24, 26] for commonly used components can be found. Also, used in the design phase of a project, FMEA is a useful tool for discovering potential failure scenarios [100]. From a large system perspective, it may be found that base component failure modes may have more than one possible system event associated with them. Often there will be a clear one to one mapping, but probabilities to failure (as used in FMECA, see section 2.8) could mean one (failure mode) too many (system level symptoms).

Use of Markov chains to model failure modes. We could represent a failure mode and its possible outcomes using a Markov chain [41]. Where multiple simultaneous failure modes are considered this complicates the statistical nature of the Markov chain cause and effect model. What we in fact get is the merging, or local interaction of two Markov chains for the cause and effect model.

Subjective and Objective thinking in relation to FMEA. FMEA is always performed in the context of the use of the equipment. In terms of philosophy the context is in the domain of the subjective and the logic and reasoning behind failure causation, the objective. By using objective reasoning a component level failure to a system level event can be traced, but only in the subjective sense its meaning and/or severity be determined. It is worth remembering that failure mode analysis performed on the leaks possible from the O ring on the space shuttle did not link this failure to the catastrophic failure of the spacecraft [105, 1]. This was not a failure in the objective reasoning, but more of the subjective, or the context in which the leak occurred. What this means is that for an objectively calculated failure mode outcome, there may have more than one subjective outcome.

This means that objective reasoning can be applied to determine objective effects, but the criticality —or the seriousness/consequences— of those failures depends upon the Equipment Under Control (EUC) and its environment. For instance a leak of nuclear material aboard a spacecraft could have the consequences of loss of mission, but a leak on earth could have serious health and environmental consequences. This means one line of FMECA describing a system risk is an over simplification (consider that the same nuclear material will

be present during transport and launch, and when outside earth's environment). Subjective appraisal of the outcome of a system failure mode can also be subject to management and/or political pressure. The two most recent variants of FMEA, FMEDA and FMECA have dipped a metaphorical toe into the subjective realm, FMECA with its 'criticality factor' and FMEDA with its definition of 'dangerous'. However, while starting to address the subjective side of failure analysis, these methodologies do not separate the final subjective stage from the objective. A subjective assessment is made during the analysis of each base component failure mode regardless of the fact that most base component failure modes cause shared system level failures. This means that work at the subjective level is repeated. Detailed work on subjective analysis is beyond the scope of this study.

Multiple Simultaneous Failure Modes. FMEA is less useful for determining events for multiple simultaneous failures[7]. Multiple failures may cause the same system level failure (i.e. two separate failures could cause the same system failure, and in combination still cause the same failure), this can be termed a common failure result. Work has been performed using component failure statistics and logic to offer selected—by virtue of statistical likelihood and common failure result reduction—multiple failures for analysis and consideration by an investigating engineer [82]. A complication for multiple failure analysis is that failure modes may cause a change in circuit topology meaning the additional failures might have to be analysed with respect to the changed topology. Because multiple failures mean dealing with changed topologies the objective criteria is additionally complicated with the subjective adding yet another layer of complication. Traditional FMEA has the translation from an objective to subjective failure modes as an intrinsic part of its process, which can be considered a weakness.

Failure modes and their observability criterion: detectable and undetectable. Often the effects of a failure mode may be easy to detect, and equipment can react by raising an alarm or compensating for the resulting fault. Some failure modes may cause undetectable failures, for instance a component that causes a measured reading to change could have adverse consequences yet not be flagged as a failure. This type of failure can not be dealt with by passing error indication to higher level modules because it simply cannot be detected. The system therefore has no way of knowing the reading is invalid. The term observable has a specific meaning in the field of control engineering [43, 17]; systems submitted for FMEA are generally related to control systems, and so to avoid confusion the terms 'detectable' and 'undetectable' (as defined in EN61508[95]) will be used for describing the observability of failure modes in this document.

Impracticality of Field Data for Modern Systems. Modern electronic components, are generally very reliable, and the systems built from them are thus very reliable too. Reliable field data on failures will, therefore, be sparse. Should it be wished to prove a continuous demand system for say 10^{-7} failures[8] per hour of operation, even with 1000 correctly monitored units in the field there could only be one failure per ten thousand hours

[7] Multiple simultaneous failures are taken to mean failures that occur within the same detection period. Detection periods are typically determined for the process under control. For instance, for a flame detector in an industrial burner this is typically one second. [14]

[8] 10^{-7} failures per hour of operation is the threshold for S.I.L. 3 reliability [95]. Failure rates are normally measured per 10^9 hours of operation and are known as Failure in Time (FIT) values. The maximum FIT values for a SIL 3 system is therefore 100.

expected (i.e. a little over one a year) . It would be utterly impractical to get statistically significant data for equipment at these reliability levels. However, FMEA can be used (more specifically the FMEDA variant, see section 2.9), working from known component failure rates, to obtain statistical estimates of the equipment reliability.

Forward and Backward Searches. A forward search starts with possible failure causes and uses logic and reasoning to determine system level outcomes. Forward search types of fault analysis are said to be 'inductive'. A backward search starts with (undesirable) system level events and works back down to potential causes using de-composition of the system and logic. FMEA based methodologies are forward searches[56] and top down methodologies such as FTA [84, 70] are backward searches. Backward (or bottom-up) searches are said to be deductive (i.e. the results of failure are deduced).

2.6.1 Reasoning distance.

Reasoning distance, is the number of stages of logic and reasoning used in failure mode analysis to map a failure cause to its potential outcomes; counted by the number of failure mode to component checks made. The basic FMEA example in section 2.1 considered one failure mode against some of the components in the milli-volt reader. To create an exhaustive FMEA report on the milli-volt reader, every known failure mode of every component within it would have to be examined against all its other components. 'Reasoning distance', for one failure mode, is defined as the number of components checked against it to determine its system level symptom(s). No current FMEA variant gives guidelines for the components that should be included to analyse a failure mode in a system. Were a failure mode examined against all the other components in a system this would give us the maximum reasoning distance. This is termed the exhaustive FMEA case for a single failure mode. Thus the exhaustive reasoning distance for a particular component would be to multiply the number of failure modes it has by the number of remaining components in the system. The exhaustive reasoning distance for a system would be the the sum of these multiplications for all the components it contains. If the milli-volt reader had say 100 components, with three failure modes each, this would give an exhaustive reasoning distance—for single failure analysis—of $3 \times 100 \times 99$. The discussion on reasoning distance provides a metric to examine the state explosion problems associated with forward search failure investigation methodologies. It is apparent that the shorter the reasoning distance, the more precisely theoretical examination can determine failure symptoms. For instance for a very simple small circuit, a better understanding of failure effects is expected, than for a very large system where there are more variables and potential failure mode interactions.

2.6.2 FMEA and the State Explosion Problem

Problem of which components to check for a given base component failure mode. FMEA for safety critical certification (i.e. for EN298 and EN61508) [14, 95] has to be applied to all known failure modes of all components within a system. Each one of these, in a typical report, would be one line of a spreadsheet entry. FMEA does not define or specify the scope of the investigation for each component failure mode. For instance should the signal path be followed, with all components encountered along that, or should the scope be wider?

Exhaustive Single Failure FMEA. To perform exhaustive FMEA (XFMEA), every possible interaction of a failure mode with all other components in a system must be examined. Or in other words, all possible failure scenarios considered. This is represented in the equation below, where N is the total number of components in the system, RD_{single} is the reasoning distance and f is the number of failure modes per component:

$$RD_{single} = N.(N-1).f. \tag{2.2}$$

This means an order of $O(N^2)$ checks to perform to undertake XFMEA for single failures. Even small systems have typically 100 components, and they typically have 3 or more failure modes each, which would give $100 \times 99 \times 3 = 29,700$ as a reasoning distance.

Exhaustive FMEA and double failure scenarios. For looking at potential double failure scenarios[9] (two components failing within a given time frame) and the order becomes $O(N^3)$. Where RD_{double} is the reasoning distance for double failure scenarios:

$$RD_{double} = N.(N-1).(N-2).f. \tag{2.3}$$

For a theoretical system with 100 components and a fixed 3 failure modes each, this gives reasoning distance of $100 \times 99 \times 98 \times 3 = 2,910,600$. In practise there is an additional complication here, that of the circuit topology changes that failure modes can cause.

Reliance on experts for meaningful FMEA Analysis. Current FMEA methodologies cannot consider—for the reason of state explosion—an exhaustive approach. Because for practical reasons, XFMEA cannot be performed for anything other than a trivial system, reliance is placed upon experts on the system under investigation to perform a meaningful analysis. These experts must use their judgement and experience to choose sub-sets of the components in the system to check against each failure mode. Also, these experts have to select the areas they see as most critical for detailed FMEA analysis: it is usually impossible, for reasons of time to perform the work, to action a detailed level of analysis on all component failure modes on anything but a small hypothetical system.

2.6.3 Component Tolerance

Component tolerances may need considering when determining if a component has failed. Calculations for acceptable ranges to determine failure or acceptable conditions must be made where appropriate. An example of component tolerance considered for FMEA is given in section 5.6.2.

2.7 FMEA in current usage: Four variants

Four main Variants of FMEA

- **FMECA - Criticality** Emphasis on minimising the effect of critical systems failing;

[9] Certain double failure scenarios are already legal requirements—The European Gas burner standard (EN298:2003)—demands the checking of double failure scenarios (for burner lock-out scenarios).

- **FMEDA - Statistical Safety** Statistical analysis giving Safety Integrity Levels;
- **DFMEA - Design or Static/Theoretical** Approval of safety critical systems using FMEA and single or double failure prevention;
- **SFMEA - Software FMEA** — Usage not enforced by most current standards [14, 15, 95].

2.8 FMECA - Failure Modes Effects and Criticality Analysis

FMECA places emphasis on determining criticality rather than the cost of system failures. It applies Bayesian statistics within the FMEA process (i.e. using probabilities of component failures and the probability of those failures causing given system level failures) to determine the risk of system level events/symptoms. The results of these risk probabilities, i.e. for system level failures, are then multiplied by the estimated operational time of the system. For instance a military or emergency system may be typically operational for a given number of hours. The risk against time value, in conjunction with the severity of the system level event gives a 'criticality level'. Bayes' theorem can be seen as a theory on the 'probability of causes' [94][p.9]. A given component failure may for instance, be associated with a particular system failure to a calculated, or measured from field data, statistical probability. Applying Bayesian statistics to failure analysis, suffers the problem that correlation does not imply causation [57]. However, correlation is evidence for causation, and maybe the only evidence to hand and this is the justification behind its use. This implies a weakness in the FMECA philosophy. It means that failure causes can be inferred, rather than analytically determined, to become part of the failure mode model. A history of the usage and development of FMECA may be found in [18].

FMECA - Statistical variables. FMECA refines FMEA, but instead of a simple top level failure as a result, a criticality or seriousness factor is also ascribed. FMECA has three probability factors for component failures, a system operational time and a severity factor.

FMECA λ_p value. This is the overall failure rate of a base component. This will typically be the failure rate per million (10^6) or billion (10^9) hours of operation [26].

FMECA α value. The failure mode probability, usually denoted by α is the probability of a particular failure mode occurring within a component [24].

FMECA β value. The second probability factor β, is the probability that the failure mode will cause a given system failure. This corresponds to 'Bayesian' probability, i.e. given a particular component failure mode, the probability of a given system level failure [84][VI-19].

FMECA 't' Value. The time that a system will be operating for, or the working life time of the product is represented by the variable t.

Severity 's' value. A weighting factor to indicate the seriousness of the putative system level error.

The statistical formula to calculate the criticallity factor for one component failure mode is given below:

$$C_m = \beta.\alpha.\lambda_p.t.s. \tag{2.4}$$

The highest C_m values would represent the most dangerous or serious system level failures. The highest C_m values would be at the top of a 'to fix' list for a project manager, and some levels of risk may be considered

unacceptable and require re-design.

2.9 FMEDA - Failure Modes Effects and Diagnostic Analysis

SIL	**Low Demand** Prob of failing on demand	**Continuous Demand** Prob of failure per hour
4	10^{-5} to $< 10^{-4}$	10^{-9} to $< 10^{-8}$
3	10^{-4} to $< 10^{-3}$	10^{-8} to $< 10^{-7}$
2	10^{-3} to $< 10^{-2}$	10^{-7} to $< 10^{-6}$
1	10^{-2} to $< 10^{-1}$	10^{-6} to $< 10^{-5}$

Table 2.2: Table adapted from EN61508-1:2001 [7.6.2.9 p33], showing statistical tolerance of 'dangerous failures' to comply with a given SIL level

FMEDA is a modern extension of FMEA, in that it recognises the effect of self checking features on safety, and provides detailed recommendations for computer/software architecture. FMEDA is the fundamental methodology of the statistical (safety integrity level) type standards (EN61508/IOC5108). The end result of an EN61508 analysis is an overall 'level of safety' known as a Safety Integrity level (SIL) assigned to an installed system. It has a simple final result, a Safety Integrity Level (SIL) from 1 to 4 (where 4 is safest). These SIL levels are broadly linked to the concept of an acceptance of given probabilities of dangerous failures against time, as shown in table 2.2. The philosophy behind this is that it is recognised that no system can have a perfect safety integrity, but that risk and criticality can be matched to acceptable, or realistic levels of risk. SIL levels are intended to classify the statistical safety of installed plant: sales terms such as a 'SIL 3 sensor' or other 'device' given a SIL level, are meaningless. SIL analysis is concerned with 'safety loops', not individual modules, sensors, computing devices or actuators. In control engineering terms, the safety loop is the complete path from sensors to signal processing to actuators for a given function in the plant. This entire loop must be designed to detect and deal with any hazards and have measures in place to reduce their affects. In EN61508 terminology, a safety loop is known as a Safety Instrumented Function (SIF). FMEDA requires the analyst to consider all hardware components in a system and requires that an MTTF value is assigned for each base component failure mode; the MTTF may be statistically mitigated (improved) if it can be shown that self-checking measures will not only detect it within the SIF, but also react in a safe way. That is that the SIF can recognise that it has a fault condition and can take appropriate action. The MTTF value for each component failure mode is denoted using the symbol 'λ'.

SIL and Software. EN61508 regulation in relation to software provides procedural quality guidelines and constraints (such as forbidding certain programming languages and/or features): it does not provide a means to trace failure mode effects in software or across the software/hardware interface. While procedural guidelines and constraints can improve software reliability, ensuring that reliability targets, for software, are actually met for given SIL levels is currently almost impossible [5].

Failure Mode Classifications and metrics in FMEDA.

- **Safe or Dangerous.** Failure modes are classified SAFE or DANGEROUS.
- **Detectable failure modes.** Failure modes are given the attribute DETECTABLE or UNDETECTABLE.
- **Four attributes for FMEDA Failure Modes.** All failure modes may thus be Safe Detected(SD), Safe Undetected(SU), Dangerous Detected(DD), Dangerous Undetected(DU)
- **Four statistical properties of a system.** The statistics for the four classifications of system failures are summed:
  $\sum \lambda_{SD}$, $\sum \lambda_{SU}$, $\sum \lambda_{DD}$, $\sum \lambda_{DU}$.

Diagnostic Coverage. The diagnostic coverage is simply the ratio of the dangerous detected probabilities against the probability of all dangerous failures, and is normally expressed as a percentage [95][2-Annex C]. $\Sigma\lambda_{DD}$ represents the percentage of dangerous detected base component failure modes, and $\Sigma\lambda_D$ the total number of dangerous base component failure modes,

$$DiagnosticCoverage = \Sigma\lambda_{DD}/\Sigma\lambda_D.$$

The **diagnostic coverage** for safe failures, where $\Sigma\lambda_{SD}$ represents the percentage of safe detected base component failure modes, and $\Sigma\lambda_S$ the total number of safe base component failure modes, is given as

$$SF = \frac{\Sigma\lambda_{SD}}{\Sigma\lambda_S}.$$

Safe Failure Fraction. A key concept in FMEDA is Safe Failure Fraction (SFF). This is the ratio of safe and dangerous detected failures against all safe and dangerous failure probabilities. Again this is usually expressed as a percentage,

$$SFF = \big(\Sigma\lambda_S + \Sigma\lambda_{DD}\big)/\big(\Sigma\lambda_S + \Sigma\lambda_D\big).$$

SFF determines how proportionately fail-safe a system is, not how reliable it is. A weakness in this philosophy is that by adding extra safe failures (even unused ones) the apparent SFF would be improved[10].

2.9.1 FMEDA - Failure Modes Effects and Diagnostic Analysis

To achieve SIL levels, diagnostic coverage and SFF levels are prescribed along with hardware architectures and software techniques. The overall aim of SIL is to classify the safety of a system, by statistically determining how frequently it can fail dangerously. That is to say the PFH or FIT value assigned to a component in a SIF is its dangerous failure rate. Individual dangerous component failure modes may be mitigated by their detection percentage and their diagnostic interval.

2.9.2 Automotive Safety Integrity Levels

The EN61508 variant for automotive use, as defined in standard ISO 26262, is known as Automotive SIL (ASIL) [48]. Safety instrumented functions (SIFs) for vehicles are assigned ASIL ratings. ASIL classifications

[10]The artificial inflation of SFF, by including unnecessary safe functions or unused components (i.e. a loophole) is closed in the 2010 edition of the standard.

are rated from A to D, where D is the most safety critical. For instance very critical functions such as the brakes and steering will have the highest ASIL rating of D. The automotive industry generally uses bought in modules typically built by specialist companies. These modules themselves must have been tested and approved so, for a car manufacturer designing from scratch is not generally financially feasible. This means that to implement an ASIL SIF designers will usually have to rely on bought in modules. However, these bought in modules may not be rated to the ASIL level required by the SIF. Because of the modular paradigm forced on the designers by having to buy in components a process has been developed called 'ASIL de-composition' [107]. This allows a highly safety critical function to be implemented with lower ASIL rated components, as long as it can be shown that they have independent failure causes and implement redundancy. This is in effect a top down de-composition of safety requirements. This is rather like the demand for multiple engines on aircraft that must make long journeys over the sea to statistically limit the likelihood of one failure cause — i.e. one engine failure — causing a serious incident. The drawback to this redundancy concept is an unexpected common failure mode [103]. The ASIL philosophy does represent a modular approach to safety analysis. This makes it of interest to this study, which later proposes a modular failure mode analysis methodology.

2.10 FMEA used for Safety Critical Approvals

2.10.1 DESIGN FMEA: Safety Critical Approvals FMEA

Experts from Approval House and Equipment Manufacturer discuss selected component failure modes judged to be in critical sections of the product. This could be considered as a design check method, deliberately looking for weaknesses at a theoretical level. Because design FMEA meetings can have the format of a meeting and discussion they can have the following drawbacks:

- Impossible to look at all component failures let alone apply FMEA exhaustively/rigorously,
- In practice, failure scenarios for critical sections are contested, and either justified or extra safety measures implemented,
- Often meeting notes or minutes only: it is unusual for detailed technical arguments to be documented.

2.11 Conclusion

Returning to the FMEA model, the data relationships shown in figure 2.2 hold for the variants of FMEA discussed. This could be extended, if it is considered that the system level symptoms have subjective interpretations. With the addition of subjective failure mode symptoms, the UML model for FMEA gains an attribute (see figure 2.5). The UML data model reveals some undefined qualities of FMEA. These raise questions and are discussed below.

Which, or how many components should be checked for each failure mode entry? For instance a given failure mode will have its effect measured in relation to some of the components in the system. These

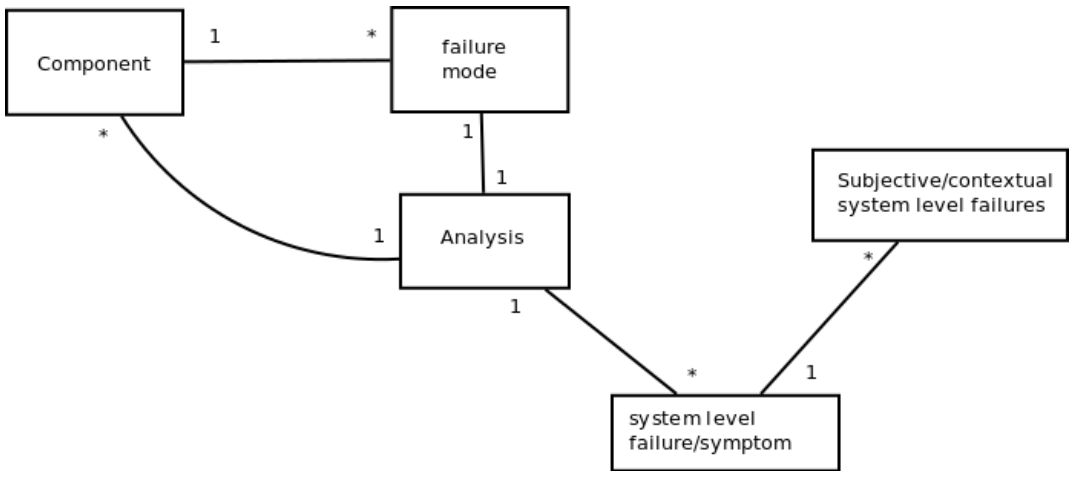


Figure 2.5: FMEA UML data representation with subjective system level failure modes.

components could be chosen by stipulating criteria relating to the signal path or adjacency in the electronic circuit, potential strategies are listed below:

- Look at all components electronically adjacent (i.e. connected to the affected component),
- Look at all components connected (as above) and those once removed (those connected to those connected to the affected component),
- Look at components forward of the failure mode in the signal path,
- Look at all components in the signal path,
- Look at all components in the signal path including those one connection removed,
- Look at all components within pre-determined dependency models [80][Ch.5],
- Look at all components in the system (i.e. XFMEA).

No current variant of FMEA gives any guidelines for which, or how many components to check for a given failure mode.

FMEA gives us objective system level failures/symptoms. The two more modern variants of FMEA, FMECA and FMEDA start to address the problem of subjective/contextual failure symptoms of a system. FMEDA classifies them as dangerous or safe failures. FMECA gives us a statistically biased criticality level. In both of these methodologies however, there is no formal stage where objective to subjective system failures are mapped, this processes seems to be intertwined with the basic analysis itself.

Re-use potential of an FMEA report. Each failure mode entry in an FMEA report should have a reasoning or comments field. This should provide a guide to someone re-examining, or trying to re-use results on a similar project. However, the depth of description for reasoning stages in FMEA entries is in practise variable. Ideally each FMEA entry would contain a clear reasoning description for each failure mode, so that the entry can be more easily reviewed or revisited/audited. Because FMEA is traditionally performed with one entry per

component failure mode, full reasoning descriptions are rare. Another effect on a one entry per failure mode model, is that the terminology may be inconsistent. Failure symptoms, although being the same at a system level, may be given different names in the same project. These factors mean that re-use, review and checking of traditional analysis can often be started from 'cold'. Work has been performed to assist in incremental FMEA production by use of a software tool which in conjunction with circuit simulation and a database of component failure modes (providing consistency in terminology) speeds up the FMEA process and aids re-use [81].

Chapter 3

FMEA Criticism

Introduction

This chapter examines current FMEA practise in a critical light. Chapter 2 introduced concepts underlying FMEA, and this chapter seeks to use these concepts to determine the drawbacks and advantages in its current usage. Legally mandatory FMEA, for a large proportion of safety critical systems in Europe and the USA, at the very least means that experienced engineers have to discuss a system at a level of detail starting at base component failure modes. This undoubtedly reveals dangers inherent in designs and makes our lives safer. This chapter aims to look for the deficiencies in current FMEA processes, to probe for weaknesses and look for ways in which it could be performed better and more efficiently.

A major problem is with the scope of examination—i.e. which/how many components should be checked against a particular failure mode—to apply FMEA analysis. Checking all possible combinations of failure modes against all components quickly leads to a state explosion problem. The difficulties of integrating software and hardware in FMEA failure models mean that FMEA is showing its age: designed in an era of simple electro-mechanical systems, the modern world with ubiquitous cheap micro-controllers and processors mean that most of todays systems are now software/hardware hybrids.

Even analogue electronics, with the advent of surface mount and miniature components, means that modern electronic circuits are typically far more complex and have far higher component counts, than those of the era when FMEA methodologies were invented.

With FMEA it is very difficult to perform meaningful multiple failure analysis [82, 58]. The main reasons for this are that in electronics, each failure can introduce a circuit topology change and state explosion means there can be extremely large numbers of double failures to check. In software, in a similar vein, one failure can influence the programmatic behaviour and decisions made, complicating the analysis of additional failures. Dual failure analysis is required by some recent European standards [14, 15] and with increasing demands on safety, additional multiple failure FMEA requirements are likely.

Other problems such as the inability to easily re-use, and validate/audit (through traceable reasoning) FMEA models are presented. Finally a list of deficiencies in current FMEA methodologies and a wish list for an improved methodology are presented.

3.1 Historical Origins of FMEA and the base component failure mode to system level failure/symptom paradigm

3.1.1 FMEA: base component failure mode to system level failure modelling

FMEA traces it roots to the 1940s when it was used to identify the most costly failures arising from car mass-production [62]. It was later modified slightly to identify/compare severity levels of the system level failures (FMECA [25]). In the 1980s FMEA was extended again (FMEDA [39]) to provide statistics for predicting safety levels/failure rates. However a typical entry in each of the above methodologies, starts with a particular component failure mode and associates it with a system—or top level—failure symptom. This means that there is one analysis case per component failure mode for all the components in the system under investigation. This analysis philosophy has not changed since FMEA was first used.

3.1.2 FMEA does not encourage Traceable Reasoning

An FMEA report normally assigns one line of a spreadsheet to each base component failure mode. This means that the reasoning involved in determining the system level failure/symptom is described (if at all) very briefly. Ideally supporting documentation would give the reasoning and calculations behind each analysis case, but the structure of current FMEA reports does not encourage this.

Re-use of FMEA analysis. Given the base component failure mode to system level failure mode paradigm it is difficult to re-use FMEA analysis. Several strategies to aid re-use have been proposed [86, 54, 101], but the fundamental problem remains, that, with any changes to the component base in a system, it is very difficult to determine which FMEA test scenarios must be re-worked. With component failure mode databases, in conjunction with circuit simulation, work has been performed to address this [81]. It is common in safety critical systems to have repeated circuit topologies. For instance there may be several signal input and output structures that are repeated. The failure mode behaviour of these repeated structures will be the same. However due to the base component failure mode to system level failure mode mapping paradigm of FMEA, work is likely to be repeated.

3.1.3 FMEA does not support modularity.

It is a common practise in the process control industry to buy in sub-systems, typically sensors and actuators connected to an industrially hardened computer bus, i.e. CANbus [79, 11], modbus [68] etc. With traditional FMEA it is difficult to deal with a 'plug and play' paradigm. The design philosophy of FMEA is to trace base component failures through to system failures. This is incompatible with a modular approach where the architecture of a system may be different for implementation sites. The modularity problem is exacerbated by FMEA's problems modelling software/hardware hybrids, a problem examined in section 3.3.2.

3.1.4 FMEA one to many mapping for component failure to system level failure modes

Traditional FMEA allows for the possibility of base component failure modes causing more than one potential system failure mode. This can be seen as an indicator of the lack of cause to effect precision possible when analysing large systems using FMEA. Ideally this relationship would be many (base component failure modes) to one (system level symptoms). This would be beneficial in terms of validating precision of analysis, and for by-products of the process such as developing diagnostic fault trees [80][Ch 6.2] from FMEA results.

3.2 Comparison Complexity

Traditional FMEA cannot ensure that each failure mode of all its components are checked against any other components in the system which it may affect, due to state explosion. FMEA is therefore performed using heuristics to decide on which components to check the effect of a component failure mode. Typically FMEA will be performed by following the signal path of the component failure mode to its system level effect, echoing fault finding/diagnostic techniques [36]. This is less than ideal and it can easily miss interactions with adjacent components, that could cause other system level symptoms. If a reasoning distance used is compared with the theoretical maximum, i.e. as defined in equation 2.2, comparison complexity figures can be produced. Complexity comparison here, means the maximum number of checks (i.e. exhaustive analysis) compared to the number actually performed. In effect a yard stick for the amount of work performed for a particular FMEA analysis technique/strategy.

The ideal of exhaustive FMEA (XFMEA). Obviously, exhaustively checking every component failure mode in a system, against all other components is the ideal for finding all possible system level failures. While this is impossible for all but trivial systems, it should be possible for small groups of components that work together to provide a well defined function. A small group of components performing a well defined function is termed a 'functional grouping'. Potentially, using functional groupings, is a way of de-composing the problem and reducing the $O(N^2)$—see equation 2.2—state explosion effect associated with XFMEA. That is if the analysis problem can be broken into smaller steps, involving small groups of components, XFMEA could be applied within those, without causing a debilitating state explosion effect. This property is examined in section 7.1.3. A comparison complexity order, or reasoning distance, of $O(N^2)$ could be seen as desirable in an automated process such as a search algorithm, but here it is a time consuming manual process which demands experienced and highly qualified personnel [76]. It is therefore desirable to reduce this order further.

3.3 Software and FMEA

Traditional FMEA deals only with electrical and mechanical components, i.e. it does not have provision for software. Modern control systems nearly always have a significant software/firmware element, and not being able to model software with current FMEA methodologies is a cause for criticism [53][Ch.12]. Some techniques apply blanket estimates for a given software implementation [53][pp.156-9], based on the verification techniques

applied in its testing, to aid calculation of system level reliability statistics [6]. Similar difficulties in integrating mechanical and electronic/software failure models are discussed in [2, 10].

Current work on Software FMEA. SFMEA usually does not seek to integrate hardware and software models, but to perform FMEA on the software in isolation [73]. Work has been performed using databases to track the relationships between variables and system failure modes [44], to introduce automation into the FMEA process [106] and to provide code analysis automation [90]. Although the SFMEA and hardware FMEAs are performed separately, some schools of thought aim for Fault Tree Analysis (FTA) [70, 84] (top down - deductive) and FMEA (bottom-up inductive) to be performed on the same system to provide insight into the software hardware/interface [40]. Subtle problems in embedded software are often due to interrupt contention causing unintended corruption of variables: automated tools to aid the detection of this are becoming available [88]. Work has been performed to parse software, and to map source code statements as edges and variables as nodes, to form directed acyclic graphs [91], where failure mode propagation can be traced. Although current software FMEA techniques should give a better picture of the failure mode behaviour, they are by no means a rigorous approach to tracing errors that may occur in hardware being followed through to the top (and therefore ultimately controlling) layer of software. That is they do not offer an integrated software hardware failure mode model. With the increasing use of micro-controllers in place of much analogue electronics for most new designs of electronic product, the poor software integration capabilities of FMEA are now being seen as deficiencies.

This is becoming apparent in a dilemma now faced by organisations dealing with highly safety critical systems and having to rely on 'smart instruments' [7] that can no longer be validated using FMEA. Smart instruments are discussed in the section below. Distributed real time systems, which rely on micro-controllers connected in a network using a communications protocol, similarly are difficult to meaningfully analyse using FMEA (see section 3.3.2).

3.3.1 The rise of the smart instrument

A smart instrument is defined as one that uses a micro-processor and software in conjunction with its sensing electronics, rather than analogue electronics only [72]. It is termed 'smart' because it has some software, or intelligence incorporated into it. For instance, an AVO-8 multi-meter circa 1970, uses only analogue electronics and it can therefore be determined using FMEA how component failures within it could affect readings. A modern multi-meter will have a small dedicated micro-processor and sensing electronics, all on the same chip, with firmware to read the user controls and display results. For quality control, many safety critical processes require regular inspections and measurements of physical characteristics of materials and machinery. For highly critical systems e.g. the nuclear industry [77], the instruments used to perform these measurements, must be analysed using traditional assessment (which entails FMEA), to ensure that failure modes within the instrument cannot lead to invalid measurements. Some work has been performed to offer black box—or functional testing—of these instruments instead of static analysis [8]. However, black box testing of smart instruments is yet to be an approved method of validation.

Most modern instruments now use highly integrated electronics coupled to micro-controllers, which read and filter the measurements, and interface to an LCD readout. For the highly critical systems, that means they

cannot use traditional FMEA to validate the design of instruments. While noting that being more modern, these instruments are likely to be more reliable and accurate than the analogue instruments in use some twenty years ago but this cannot be validated to a high level of reliability. This remains an unsolved problem for the industries dealing with highly safety critical systems. Currently the only way that some smart instruments have been permitted for use in highly critical systems is to have them extensively functionally tested [9].

3.3.2 Distributed real time systems

Distributed real time systems are control systems where smart sensors/actuators communicate over a communications bus to a master controller. Most modern cars follow this information technology pattern and use CANbus [11, 79]. For instance, in a modern car there will be no mechanical linkage from the throttle pedal to the engine, instead the pedal will be linked to a sensor to determine how far down it is pressed. This sensor will be read by a micro-controller, and values passed via CANbus, to the Engine Control Unit (ECU) which will use that information (along with information from other sensors) to adjust the power required from the engine. This adjustment could be direct, or could be another CANbus message passed to a micro-controller regulating engine function. In terms of FMEA, see figure 3.1, our reasoning path spans (at least) four interface layers of electronics to software. Traditional FMEA does not cater for the software hardware interface and using a distributed system means the signal path will cross several hardware/software interfaces[1].

The failure reasoning paths for a distributed real time system, with its multiple passes of the hardware/software interface, mean traditional FMEA, for these systems, is impossible to perform. The base component failure mode to system failure paradigm is thus utterly anachronistic in the distributed real time system environment.

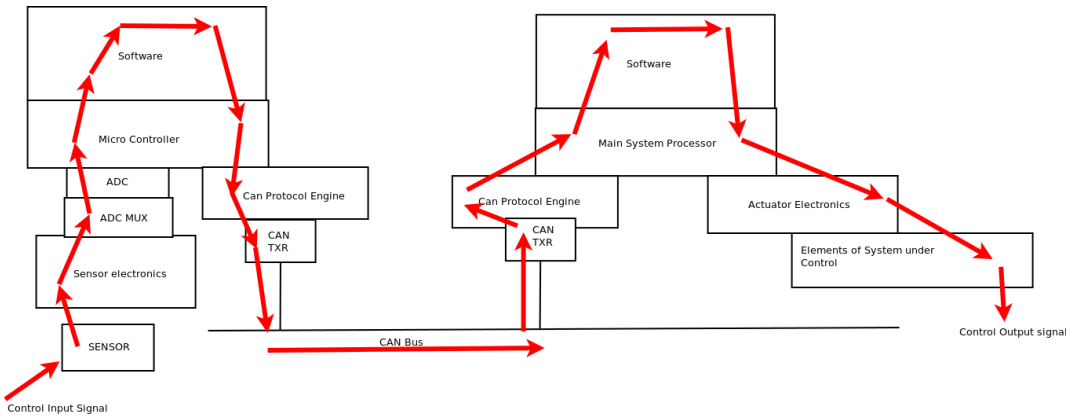


Figure 3.1: Distributed Control System FMEA signal path for a single input.

3.4 FMEA —- general criticism — conclusion

A summary of deficiencies in current FMEA methodologies is listed below:

- State explosion - very difficult/time consuming to perform FMEA exhaustively,

[1]The complications of introducing a communications protocol and the failure mode characteristics of the communications physical layer must also be considered for a distributed system.

- Difficult to re-use previous analysis work,
- Very difficult to model simultaneous/multiple failures,
- Software and hardware models are separate (if the software is modelled at all) meaning the software interface may not be correctly modelled,
- FMEA methodologies are undefined in regard to which components to check against given failure modes,
- Distributed real time systems are very difficult to analyse with FMEA because they typically involve many hardware/software interfaces.

Traditional forms of FMEA are no longer of meaningful use for complex modern systems especially those incorporating programmatic elements. They were designed to analyse simple electro-mechanical systems and even common place high component count analogue circuits (that are usually surface mount and therefore physically small), are getting too complicated for meaningful analysis using FMEA.

3.4.1 FMEA Criticism: Conclusions.

FMEA is a useful tool for basic safety — it provides statistics on safety where field data is impractical — and is good with single failure modes linked to top level events. FMEA has become part of the safety critical and safety certification industries. SFMEA is in its infancy, and there are corresponding gaps in certification for software, EN61508 [95] a modern standard based on a modern variant of FMEA, FMEDA, recommends hardware redundancy architectures in conjunction with FMEDA for hardware: for software it recommends language constraints, software life cycle control, testing regimes and quality procedures but no inductive fault finding technique. FMEA has adapted from a cost saving exercise for mass produced items [62, 69], to incorporating statistical techniques (FMECA) to allowing for self diagnostic mitigation (FMEDA). However, it is still based on the concept of single component failures mapped to top level/system failures, with a one step analysis stage.

3.4.2 FMEA - Better Methodology - Wish List

A wish list is presented, stating the features that should exist in an improved FMEA methodology,

- Must be able to analyse hybrid software/hardware systems,
- avoid state explosion (i.e. XFMEA is impractical by hand [80]),
- exhaustive checking at a modular level,
- traceable reasoning inherent in system failure models,
- re-usable i.e. it should be possible to re-use analysis,
- possibility to analyse simultaneous/multiple failures,
- one to one mapping from base component failure modes to system level failures (see section 3.1.4),
- modular — i.e. usable in a distributed system.

Chapter 4

Failure Mode Modular Decomposition

4.1 Introduction

This chapter starts with a worked example to introduce a new methodology, Failure Mode Modular Decomposition (FMMD). This is followed by a discussion on the design of FMMD, a description of the FMMD process and finally the data structures required using UML class models.

FMMD is in essence a modularised variant of traditional FMEA [96][pp.34-38]. In order to analyse from the bottom-up and apply a modular methodology, small groups of components that naturally work together to perform simple functions are chosen: these groups are termed 'functional groupings'. The components to include in a functional grouping are chosen by hand. With a functional grouping the failure modes of all the components that belong to it can be determined. All the failure modes of all the components within a functional grouping are collected. Each component failure mode can be considered as a 'failure scenario' or 'test case' to be applied to the functional grouping. Each of these failure modes, and optionally combinations of them, are formed into test cases which are analysed for their effect on the failure mode behaviour of the functional grouping. Once the failure mode behaviour of the functional grouping is obtained, its symptoms of failure can be determined. These symptoms are then treated as failure modes of the functional grouping. That is, how the functional grouping can fail has been determined. As a set of failure modes has been defined for the functional grouping it can be treated as a component in its own right. The functional grouping can be considered as a 'derived component' with its own set of failure modes. Because a derived component has a set of failure modes it can be used in higher level functional groupings which in turn produce higher level derived components. These derived components can be used to build further functional groupings until a hierarchy of functional groupings and derived components has been built, converging to a final derived component at the top of the hierarchy. The failure modes of the final or top derived component are the failure modes of the system under investigation. That is, the traditional FMEA process has been taken and modularised from the bottom-up. In this way FMEA is applied incrementally to an entire system. This has advantages of concentrating effort where modules interact (interfaces), of being able to re-use work and savings in the complexity of performing FMEA (because the analysis is typically performed in several small stages thus avoiding state explosion).

4.2 Worked Example: Non-Inverting Amplifier

The principles of FMMD are demonstrated, by using it to analyse a common circuit, the non-inverting amplifier built from an op amp [78][p.234] and two resistors; a circuit schematic for this is shown in figure 4.1. The

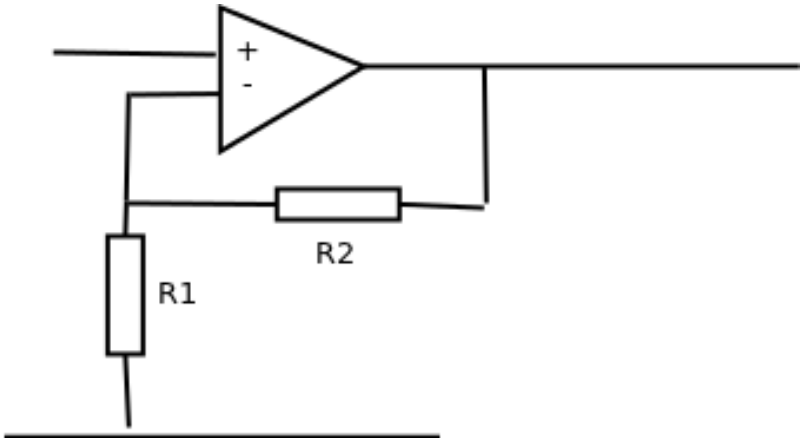


Figure 4.1: Standard non inverting amplifier configuration

function of the resistors in this circuit is to set the amplifier gain. The resistors act as a potential divider—assuming the op-amp has high impedance—and program the inverting input on the op-amp to balance them against the positive input, giving the voltage gain (G_v) defined by $G_v = 1 + \frac{R2}{R1}$ at the output.

Analysing the failure modes of the Potential Divider. Since the resistors work to provide a clearly defined function, that of a potential divider, they can be treated as a collection of components with a specific functionality—i.e. a 'functional grouping'. This functional grouping has two members, $R1$ and $R2$. The potential divider circuit can be considered as a component that provides the function of splitting two voltages into three, the third voltage being a ratio defined by the values of the resistors. Using the EN298 specification for resistor failure [14][App.A], we can assign failure modes of $OPEN$ and $SHORT$ to the resistors individually (assignment of failure modes is discussed in more detail in section 2.4.1). A resistor and its failure modes are represented as a directed acyclic graph (DAG) in figure 4.2.

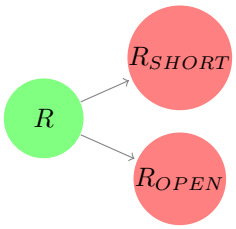


Figure 4.2: DAG representing a resistor and its failure modes.

Thus $R1$ has failure modes $\{R1_{OPEN}, R1_{SHORT}\}$ and $R2$ has failure modes $\{R2_{OPEN}, R2_{SHORT}\}$. Each of these base component failure modes are examined to determine how they affect the operation of the potential divider. Each resistor failure mode is a potential failure cause in the potential divider. For each failure mode in this functional grouping—potential divider—a failure cause number is assigned (see table 4.1). Each

failure cause is analysed to determine a failure in the potential dividers' operation. For instance if resistor R_1 were to go open, then the potential divider would not be grounded and the voltage output from it would float high (+ve). This would mean the resulting failure of the potential divider would be voltage high output. The failure mode of a high potential divider output is termed 'HighPD', and for it outputting a low voltage 'LowPD'. From table 4.1 it can be seen that the resistor failure modes lead to some common symptoms of failure from the

Table 4.1: Potential Divider: FMEA for single failures

Failure Cause	**Pot.Div Effect**	**Derived Component Failure modes**
FC1: R_1 SHORT	LOW	LowPD
FC2: R_1 OPEN	HIGH	HighPD
FC3: R_2 SHORT	HIGH	HighPD
FC4: R_2 OPEN	LOW	LowPD

perspective of the functional grouping. Notice the many to one mapping from base component failure modes to derived component failure mode; this is a typical effect of an FMMD analysis stage, and means that with each analysis stage the number of failure modes to consider has been reduced. The FMMD analysis task is therefore simplified for further stages. By drawing vertices for failure modes, and edges for the relationships between them analysis is represented by the DAG in figure 4.3. A derived component to represent this potential

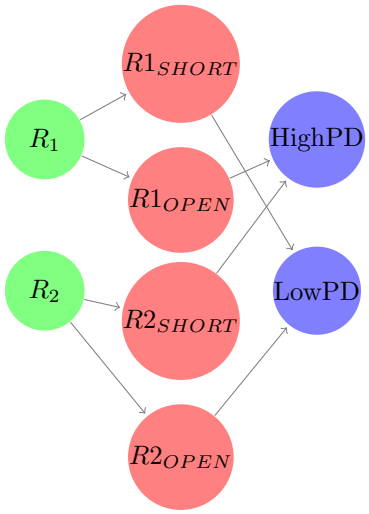


Figure 4.3: Failure mode graph of the Potential Divider

divider has been created : this is named **PD**. This derived component will have two failure modes, $HighPD$ and $LowPD$. This derived component model for a generic potential divider can be used as a building block for other functional groupings in the same way that the base components $R1$ and $R2$ were.

Failure Mode Analysis of a generic op-amp. Consider the op-amp as a base component. According to FMD-91 [24][3-116] an op amp may have the following failure modes latch-up (l_up), where the output voltage is stuck at high , latch-down (l_dn), where the output voltage is stuck low, no-operation (noop), where

the op-amp cannot drive the output, and low slew rate (lowslew) where the op-amp cannot react quickly to changes on its inputs. These op-amp failure modes are represented on the DAG in figure 4.4. The op-

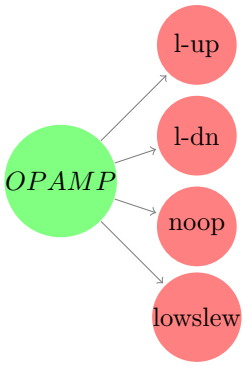


Figure 4.4: DAG representing failure modes of an Op-amp

amp and the derived component *PD* are now formed into a functional grouping to model the failure mode behaviour of the non-inverting amplifier. The two components in this new functional grouping, the op-amp and the derived component *PD* have failure modes which are used as failure causes in table 4.2. For this

Table 4.2: Non Inverting Amplifier: Failure Mode Effects Analysis: Single Faults

Failure Cause	**Amplifier Effect**	**Derived component Failure Mode**
FC1: *OPAMP* LatchUP	Output High	AMPHigh
FC2: *OPAMP* LatchDown	Output Low Low gain	AMPLow
FC3: *OPAMP* No Operation	Output Low	AMPLow
FC4: *OPAMP* Low Slew	Low pass filtering	LowPass
FC5: *PD* LowPD	Output High	AMPHigh
FC6: *PD* HighPD	Output Low Low Gain	AMPLow

amplifier configuration there are three derived component failure modes; *AMP_High, AMP_Low, LowPass.* This model now has two stages of analysis. From the analysis in table 4.2 the derived component *NONINVAMP* can be created, which represents the failure mode behaviour of the non-inverting amplifier. The analysis stages of INVAMP are presented as an Euler diagram, showing the choice of de-composition of the system into functional groupings in figure 4.6. The failure mode relationships in the derived component *INVAMP* can be traced through the DAG. It is possible to traverse this DAG, tracing the top level failure modes down to the base component failure modes, and thus determine all possible causes for the three high level symptoms, i.e.

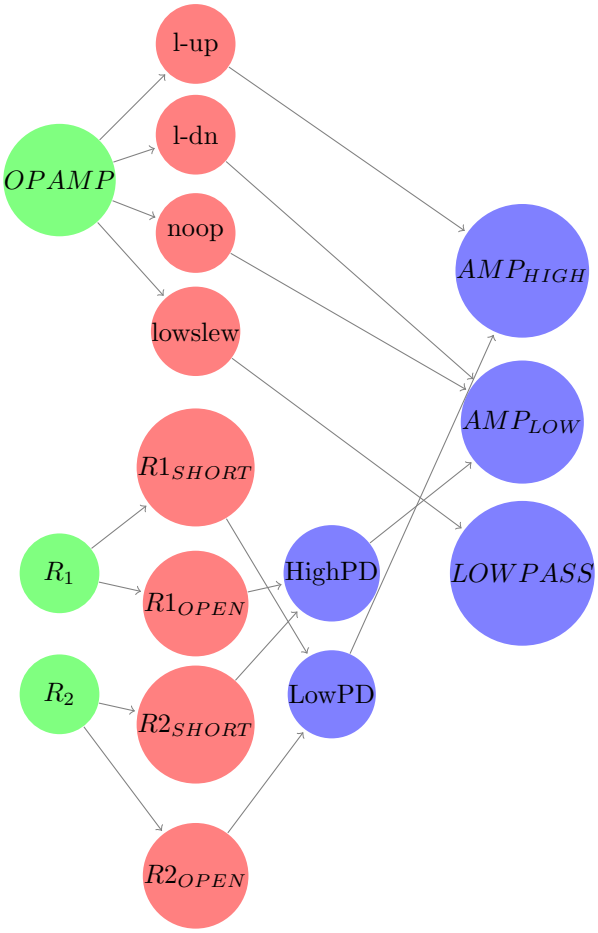


Figure 4.5: Full DAG representing failure modes and base components of the Non Inverting Op-amp Circuit

the base component failure modes of the non-inverting amplifier derived component *INVAMP*. Knowing all possible causes for a top level event/failure mode is extremely useful; if a particular top level/system failure was classified as catastrophic for instance, this information could be used to strengthen components that could cause that particular top level event/system failure. Figure 4.5 shows a DAG, where top level failure modes can be traced to the base component failure modes that can cause them. That is, failure mode effects can be traced from base component level to the top and vice versa.

4.3 Defining terms

A discussion on the terms Parts, Components and Base Components. A component is anything used to build a system. It could be something quite complicated like an micro-controller/servo motor, or quite simple like a resistor. A component is usually identified by its name, a manufacturer's part number and perhaps a vendor's reference number. Geoffrey Hall, writing in Spacecraft Systems Engineering [34][p.619] defines a 'part' thus "Part(definition)—The lowest level of assembly, beyond which further disassembly irrevocably destroys the item". This definition is useful, but consider parts, such as quad packaged op-amps: in this case we have four op-amps on one chip. Using traditional FMEA methods [96][p.34] each op-amp in the package would be considered as a separate building block for a circuit. For FMMD each of these four op-amps in the chip would

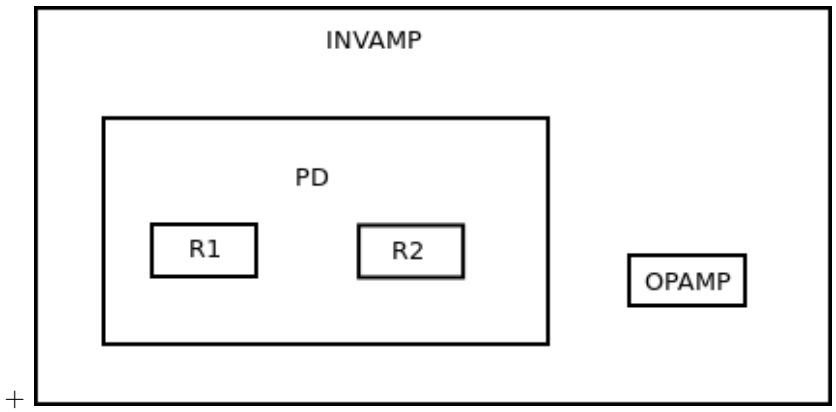


Figure 4.6: FMMD analysis of the INVAMP represented as an Euler diagram, showing how the components have been collected into functional groupings and then used as derived components to build the analysis hierarchy.

be considered to be a separate base component.

The above definition of a part, needs further refinement, i.e. to be defined as an atomic entity. Base component is defined as the lowest level entity — an entity with which to begin analysis — a component used as a starting bottom-up building block. Both op-amps and transistors have published statistical failure rates and yet an op-amp is constructed from transistors. However, a circuit designer would usually consider individual transistors and individual op-amps as lowest level building blocks. In fact any lowest level building block with published failure modes could be considered to be a base component, but this determination is the choice of the analyst, which may be influenced by the particular standard [14] [95] to which the system is being approved/analysed. To summarise, the terms, part, component, module and sub-system may have subtly different interpretations for different methodologies. FMMD considers two types of components, these are:

- A base component — A starting or building block entity with given failure modes,
- A derived component — An entity determined from failure mode analysis.

4.3.1 Definition of terms: sound system example.

A system, is any coherent piece of equipment that performs a given task. A component can be viewed as a sub-system that is a part of some larger system. A modular system common to many homes is the sound separates audio system or stereo hi-fi. This is used as an example to describe the concepts of functional grouping and derived component used by FMMD. For instance a stereo amplifier separate/slave is a component. A whole sound system consists perhaps of the following components: CD-player, tuner, amplifier separate, loudspeakers and ipod interface.

Functional Groupings and Components. Components can be composed of components, recursively on down to the base components. However each component will have a fault/failure behaviour and it should always be possible to obtain a set of failure modes for each component. Looking at the sound system example, the CD player could fail in several distinct ways, and this could have been caused by a number of the CD players internal component failure modes. Using the reasoning that working from the bottom up forces the

consideration of all possible component failures (which can be missed in a top down approach [30][Ch.9]), a problem is encountered: which initial collections of base components should we choose? For instance in the CD player example, if we start at the bottom, a massive list of base components will be found, resistors, motors, user switches, laser diodes, etc. Working from the bottom up, it is necessary to pick small collections of components that work together in some way. These collections are termed 'functional groupings'. For instance, the circuitry that powers the laser diode to illuminate the CD might contain a handful of components, and as such would make a good candidate as one of the base level functional groupings. It is a good candidate because it performs a well defined function and it could be considered a design module.

4.3.1.1 Functional grouping to derived component process outline.

Functional groupings have been defined as a set of components that interact to perform a specific function. After analysis of the fault behaviour of a functional grouping, it can be treated as a 'black box'. The functional groupings fault behaviour will consist of a set of failure modes caused by combinations of its component's failure modes. A new component can be derived from analysing the functional grouping where the symptoms of failure of the functional grouping are the failure modes of this new 'derived component'. An outline of the FMMD process is itemised below:

- Collect components to form a functional grouping,
- Create 'test cases' for all failure modes of the components within the functional grouping,
- Analyse the effect of all the test cases on the operation of the functional grouping,
- Determine the common failure modes of the functional grouping,
- Create and name a derived component for the functional grouping,
- Assign the common failure modes from the functional grouping as the failure modes of the derived component.

The FMMD process is described using formal definitions and algorithms in section B.3.

Functional grouping determination. Determining which components to include in a functional grouping is a decision made by the analyst. The analyst must look at the system schematics/design documentation and identify potential functional groupings. This would typically involve recognising configurations of components performing specific functions. To choose appropriate functional groupings involves a good understanding of the sub-system in hand and an initial top down perspective.

Failure modes used for base components. For common base components there is established literature for the failure modes for the system designer to consider (often with accompanying statistical failure rates) [26, 14, 24]. For instance, a simple resistor is generally considered to fail in two ways, it can go open circuit or it can short. Electrical components have data-sheets associated with them. Data sheets, supplied by the manufacturer, are a detailed source of information on the component. Because they are written for system designers, and to

an extent advertise the product, they rarely list failure modes. For FMEA purposes, ideally, failure modes along with with environmental factors and MTTF [96][p.165] statistics would be presented. Given the growing usage of FMEA/FMEDA and the emergence of SIL as a safety benchmark in industry, this may change. Currently, failure mode information is generally only available for generic component types [26, 24]. Thus we can associate a set of failure modes to types of component, for example $ResistorFaultModes = \{OPEN, SHORT\}$[1].

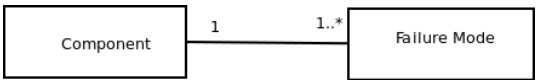


Figure 4.7: UML diagram of a component and its associated failure modes.

The UML class diagram in figure 4.7 shows a component as a data structure with its associated failure modes. From this diagram it can be seen that each component must have at least one failure mode. To clearly show that the failure modes are mutually exclusive states, or unitary states associated with one component, each failure mode is referenced back to only one component. This constraint is discussed in detail in section 7.3. By 'modularising a system' this means recursively breaking it into smaller sections for analysis. When modularising a system from the top down, as in Fault Tree Analysis (FTA) [70][84] , it is common to term the modules identified as sub-systems. When modularising failure mode behaviour from the bottom up, it is more meaningful to call them 'derived components' (i.e. they have been derived from the bottom-up according to functional criteria, rather than with the top down approach, de-composed from a system into 'sub-systems').

4.4 Failure Modes in depth

In order to perform FMEA a set of failure modes is required for each base component in the system under investigation. These are failure modes from the perspective of the user of the component. The FMEA analyst is not usually concerned with how the component has failed internally. What the analyst needs to know are the symptoms of failure. With these symptoms, their effects can be traced through the system under investigation and finally top-level failure events can be determined. Different approval agencies may list different failure mode sets for the same generic components. This apparent anomaly is discussed in section 2.3 using two common electronic components as examples.

4.5 Fault Mode Analysis, top down or bottom up?

Traditional static fault analysis methods, such as FTA [84, 70] work from the top down. They identify faults that can occur in a system, and then work down to see how they could be caused. The aim of FMMD analysis is to produce complete[2] failure models of safety critical systems from the bottom-up, starting where possible with known base component failure modes. An advantage of working from the bottom up is that it can be ensured that all component failure modes have been considered. A top down approach (such as FTA) can

[1] The failure modes of the resistor are discussed in section 2.4.1.

[2] Completeness dependent upon the completeness/correctness of the failure modes supplied by the germane standard for the base components.

miss [30][Ch. 9] individual failure modes of components, especially where there are non-obvious or unexpected top-level failures.

4.5.1 From functional group to newly derived component

The process for taking a functional grouping, analysing its failure mode behaviour, considering all the failure modes of all the components in the group and collecting symptoms of failure, is termed 'symptom abstraction'. This is examined using an algorithmic description, in appendix B.

In terms of the UML model (see figure 4.8), the symptom abstraction process takes a functional grouping and creates a new derived component from it. The newly created derived component requires a set of failure modes of its own. As a derived component inherits from component, the UML model shows that it inherits the property of a set of failure modes. A functional grouping must comprise of at least one component, and the UML diagram shows this with the one to many relationship. Under exceptional circumstances a component may need to be a member of more than one functional grouping (this is looked at in section 7.8.1.1). The relationship between the functional grouping and component is therefore—using UML notation—'$\star \leftrightarrow 1..\star$'. A functional grouping will only be associated with one derived component and is given a one to one relationship in the UML diagram. Each functional grouping will have one analysis report associated with it. The UML representation is shown in figure 4.8.

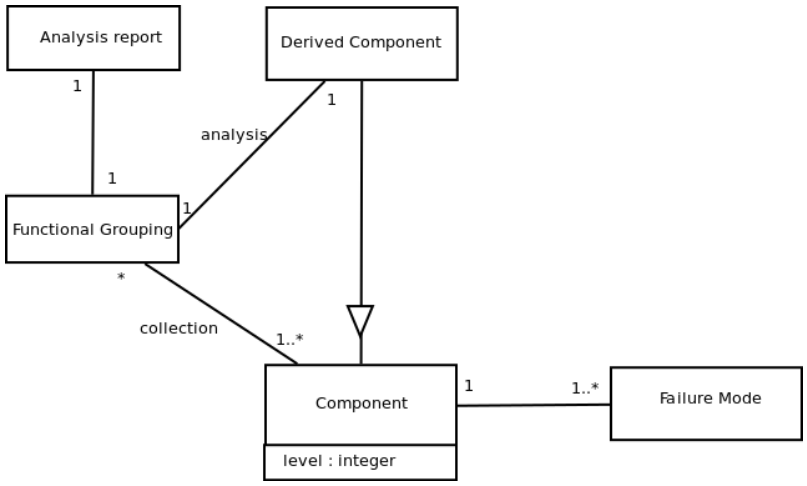


Figure 4.8: Basic UML Meta model for FMMD hierarchy

4.5.2 How the UML Meta Model maps to an FMMD Hierarchy

The UML meta model above (see figure 4.8) describes a hierarchical structure. This is because, as derived components inherit the properties of components, derived components may be used to form functional groupings. Consider the hierarchy from the example in figure 4.6. The lowest level in this hierarchy are the base components, the resistors and the op-amp. The resistors are collected into a functional grouping, and the *PD* derived component created from its analysis, is shown enclosing R1 and R2. As this derived component inherits the properties of a component, it may be used in a functional grouping higher in the hierarchy. The *PD* derived component is now placed into a functional grouping with the op-amp. This functional grouping is analysed

and a derived component created to represent the failure mode behaviour of the *INVAMP*[3]. The *INVAMP* derived component may now be used in even higher level functional groupings. An analysis report is generated for each stage in the FMMD process. The UML model in figure 4.8 describes a hierarchical structure analogous to that of a file system with directories, but instead of directory and file nodes, there are closely linked functional grouping and derived component pairs, that perform a similar structural function. To demonstrate the hierarchical nature of the UML model for FMMD, the NONINVAMP example is presented as an instance diagram below (see figure 4.9). By tracing the component failure modes to symptoms (which would defined in the analysis reports) the failure causation logic can be followed and thus the DAG's derived (see figure 4.5).

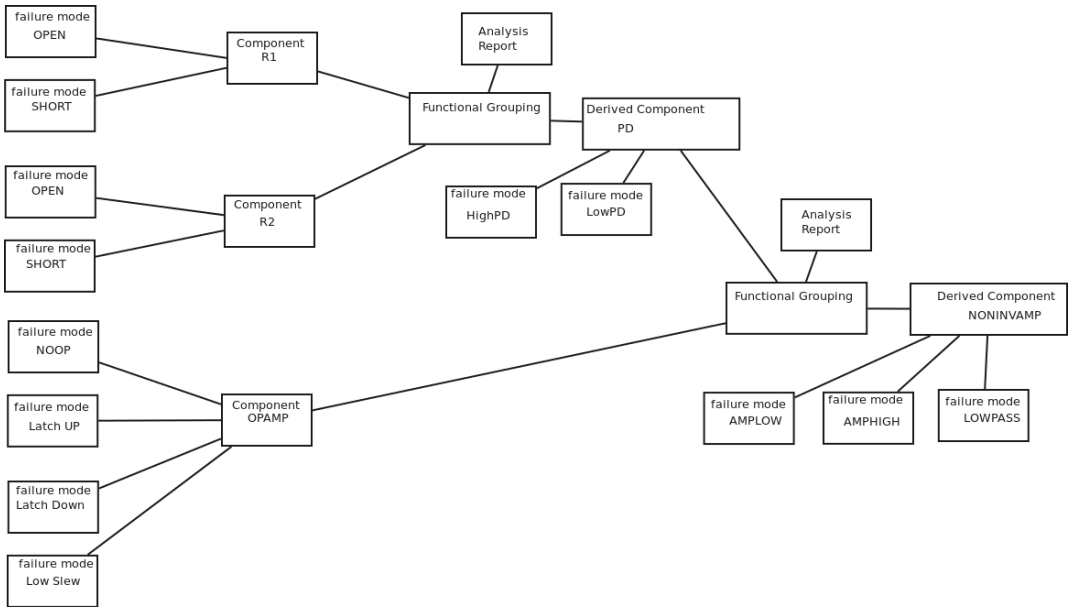


Figure 4.9: Instance diagram for the NONINVAMP example.

Traceability and quality of FMMD analysis. By having an analysis report for each analysis stage, we add traceability to the reasoning applied to the FMMD process. Consider that traditional FMEA has one large reasoning stage, that of component failure mode directly to system level failure. The reasoning given is typically a one line comment on a spreadsheet entry [96][p.38]. FMMD typically has several reasoning stages (i.e. from each functional grouping to derived component) up to the final system level derived component. Thus, each possible cause for a system failure will have a collection of FMMD analysis reports associated with it. These collections of analysis reports will provide a cause and effect story for each possible scenario that could lead to the system level failure. Traceability of design processes are considered necessary for safety critical product [95] and is an important concept in quality systems [13]. Having analysis reports increases the traceability—or documented paper trail—aiding understanding and maintainability for failure mode models. Also a detailed cause and effect model is useful for creating diagnostic schemas [60, 80].

[3]The results of this analysis are placed into the analysis report. This will contain mapping relationships between the component failure modes and the derived component failure modes and ideally, descriptions that would aid auditors to understand the reasoning behind each analysis test case.

Keeping track of the derived components position in the hierarchy. The UML meta model in figure 4.8, shows the relationships between the entities used in FMMD. To keep track of the level in the hierarchy (i.e. how many stages of component derivation have led to the current derived component) we can add an attribute to the component data type. This can be a natural number called the level variable $\alpha \in \mathbb{N}_0$. The α level variable in each component, indicates the position in the hierarchy. Base components have a 'level' of $\alpha = 0$. Derived components take a level based on the highest level component used to build the functional group it was derived from plus 1. So a derived component built from base level components would have an α value of 1. In this example the resistors and op-amp are level zero (base components, $\alpha = 0$), the *PD* a level 1 derived component ($\alpha = 1$) and the *INVAMP* a level 2 derived component ($\alpha = 2$). Because functional groupings may include components at varying levels of α, having it quickly available as an attribute will be required in practical implementations to order the tree, and assist in preventing recursion in the hierarchy (i.e. where a functional grouping could erroneously include a component above its self in the hierarchy). The abstraction level concept is formally defined in appendix B.3.5.

4.6 Conclusion

Failure model Completeness. It is undesirable to miss any component failure mode in the analysis process; were this to happen the failure model would be incomplete. Given the starting conditions of base component failure modes from the literature, it can be ensured that all these failure modes are traceable to subsequent derived component failure modes in the model. With the above condition true, this is termed a 'complete' FMMD failure model. Ensuring this condition is described in section B.3.2.

Mutual exclusivity of derived component failure modes. It is a desirable feature of a component that its failure modes are naturally mutually exclusive. This also applies to derived components produced in the FMMD process. In the FMMD process common symptoms are collected, i.e no component failure modes may be linked to more than one symptom and therefore the failure modes of a derived component are mutually exclusive. Thus FMMD naturally produces derived components with failure modes that are mutually exclusive. This property forces the FMMD analyst to create failure modes models that have a many to one mapping from base component failure mode to system level failure, or symptom (see section 3.1.4). This property, termed a 'unitary state failure mode', is examined formally in section 7.3.

Objective and contextual/subjective failure symptoms. Because the top level failure symptoms of an FMMD analysis are objective, or the result of reasoning, we can have a final stage where we consider the subjective or contextual effects of these symptoms. With traditional FMEA methodologies this decision (the contextual effects) has to be made for each component failure mode in the system.

State explosion problem of FMEA mitigated by FMMD. Because FMMD considers failure modes within functional groups; the traditional state explosion problem in FMEA–which lead to the ideal of XFMEA—disappears. With FMMD, because the functional groupings have small numbers of components in them, XFMEA can be easily applied within the functional groupings. In broad terms, FMMD mitigates state explosion by

reducing the number of checks—failure modes against components—to perform. This issue addressed formally in section 7.1.

Uses of the FMMD failure mode model. Having a failure mode graph/model, where base component failure modes are traceable to top level/system events, provides a forward search derived failure mode model. This means that for every system level failure we can traverse back to possible failure causes in the base components. Coupled with MTTF statistics for the base components this allows prediction of statistical failure rates for system level failures (this is described in greater detail in section 2.3). The FMMD model can also be used to derive information to assist in creating related models such as FTA [84, 70], traditional FMEA, FMECA [53][p.344], FMEDA [89], diagnostics schemas [80, 60] and other failure mode analysis methodologies.

Chapter 5

FMMD Examples

This chapter demonstrates FMMD applied to a variety of typical electronic circuits including analogue and digital hybrids.

- The first example applies FMMD to an operational-amplifier inverting amplifier (see section 5.1); this examines re-use of the potential divider derived component from section 4.2. This amplifier is analysed twice, using different compositions of functional groupings. The two approaches, i.e. effects of choice of membership for functional groupings are then discussed.
- Section 5.2 analyses a circuit where two op-amps are used to create a differencing amplifier. Building on the two approaches from section 5.1, re-use of the non-inverting amplifier derived component from section 4.2 is examined, where re-use is appropriate in the first stage and not in the second.
- Section 5.3 analyses a Sallen-Key based five pole low pass filter. It demonstrates re-use of the first Sallen-Key analysis, increasing test efficiency. This example also serves to show a deeper hierarchy of derived components.
- Section 5.4 shows FMMD applied to a loop topology—using a 'Bubba' oscillator—demonstrating how FMMD differs from fault diagnosis techniques. Two analysis strategies are employed, one using initially identified functional groupings and the second using a more complex hierarchy of derived components showing that a finer grained/more decomposed approach offers greater efficiency and re-use possibilities in future analysis tasks.
- Section 5.5 demonstrates that FMMD can be applied to mixed analogue and digital circuitry by analysing a sigma delta ADC.
- Section 5.6 demonstrates FMMD being applied to a commonly used Pt100 safety critical temperature sensor circuit, analysed for single and double failure mode scenarios.

5.1 Example Analysis: Inverting OPAMP

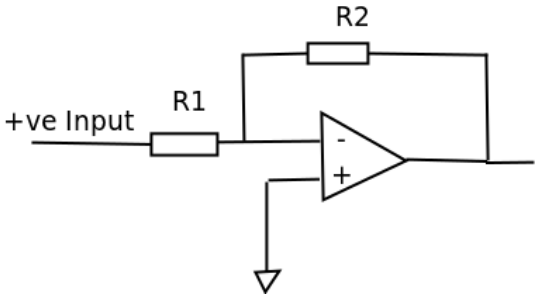


Figure 5.1: Inverting Amplifier Configuration

Figure 5.1 shows a standard configuration inverting amplifier. A valid range for the output value of this circuit is assumed. Because the amplifier inverts and the input is guaranteed positive any output voltage above or equal to zero would be erroneous i.e. an 'AMP_{HIGH}' failure symptom. A threshold would be determined for an 'AMP_{LOW}' failure symptom (i.e. the output voltage more negative than expected). Following the guidelines for the FMMD process (see section 4.3.1.1), initial potential functional groupings are identified. There are two obvious ways in which this circuit can be modelled. One is to do this in two stages, firstly by considering the gain resistors to be a potential divider and then combining it with the OPAMP failure mode model. Secondly to place all three components in one functional grouping. Both approaches are followed in the next two sub-sections.

5.1.1 First Approach: Inverting OPAMP using a Potential Divider derived component

Ideally the derived components from the PD from section 4.2 would be re-used; on initial inspection it looks a good candidate for this. However, PD cannot be directly re-used, and not just because the potential divider is floating i.e. that the polarity of the R2 side of the potential divider is determined by the output from the op-amp. The circuit schematic stipulates that the input is positive. In normal operation then, this is an inverted potential divider. It must therefore be viewed as an inverted potential divider and analysed as such; see table 5.1. A derived component can be formed from the analysis results in table 5.1 and called an inverted

Table 5.1: Inverted Potential divider: Single failure analysis

Failure Cause	**Inverted Pot Divider, IPD, Effect**	**Symptom**
FC1: R1 SHORT	$HIGH$	$IPDHigh$
FC2: R1 OPEN	LOW	$IPDLow$
FC3: R2 SHORT	LOW	$IPDLow$
FC4: R2 OPEN	$HIGH$	$IPDHigh$

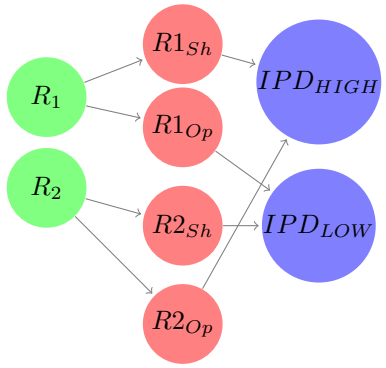


Figure 5.2: Failure symptoms of the 'Inverted Potential Divider' IPD

potential divider (IPD) with the following failure modes:

$$fm(IPD) = \{IPDHIGH, IPDLOW\}$$

The final stage of analysis for this amplifier, is made by by forming a functional grouping with the OpAmp and the new derived component IPD. Failure modes for the derived component $INVAMP$ can be expressed thus;

$$fm(INVAMP) = \{HIGH, LOW, LOWPASS\}.$$

A DAG is drawn representing the failure mode behaviour of this amplifier (see figure 5.3). Note that this allows failure symptoms to be traced back to causes, i.e. to traverse from system level or top failure modes to base component failure modes. For the one stage analysis, a DAG showing the failure mode behaviour is presented in figure 5.4.

Table 5.2: Inverting Amplifier: Single failure analysis using the IPD derived component

Failure cause	**Inverted Amp. Effect**	**Symptom**
FC1: IPD LOW	Negative on -input	$HIGH$
FC2: IPD HIGH	Positive on -input	LOW
FC5: AMP L_DN	$INVAMP_{low}$	LOW
FC6: AMP L_UP	$INVAMP_{high}$	$HIGH$
FC7: AMP NOOP	$INVAMP_{nogain}$	LOW
FC8: AMP LowSlew	$slowoutput\frac{\delta V}{\delta t}$	$LOWPASS$

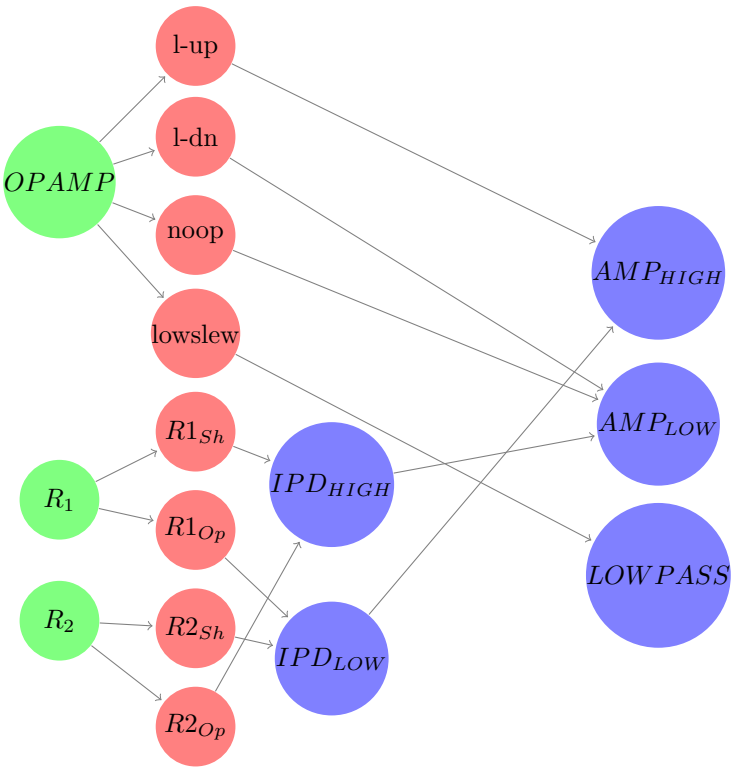


Figure 5.3: Full DAG representing failure modes and symptoms of the Inverting Op-amp Circuit

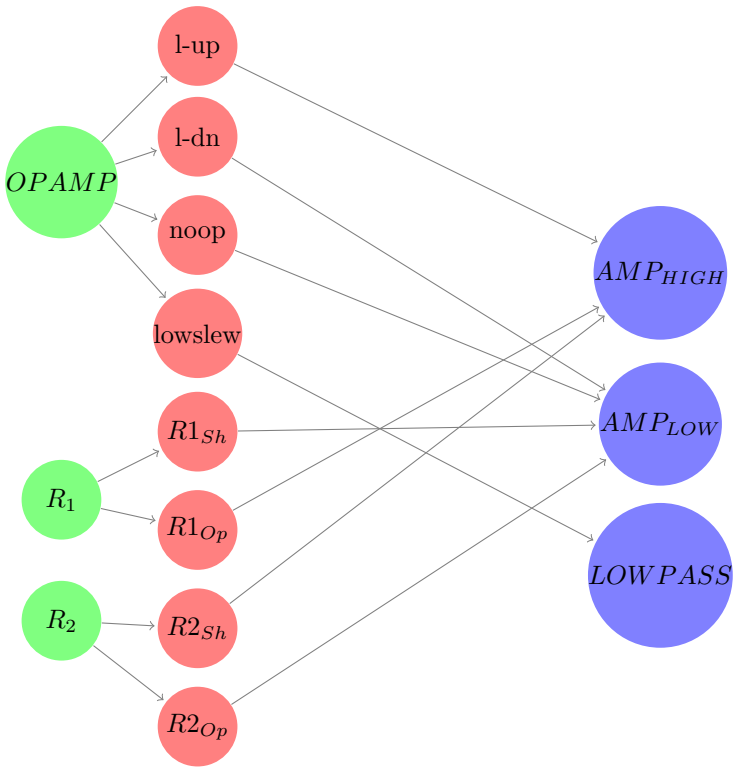


Figure 5.4: Full DAG representing failure modes and symptoms of the Inverting Op-amp Circuit analysed in one stage.

5.1.2 Second Approach: Inverting OpAmp analysing with three components in one larger functional grouping

In this second approach the inverting amplifier is analysed without using an intermediate IPD derived component. If the input voltage was not constrained to being positive this 'one stage' analysis would be necessary. This concern is re-visited in the differencing amplifier example in the next section.

Table 5.3: Inverting Amplifier: Single failure analysis: 3 components

Failure cause	**Inverting Amp. Effect**	**Symptom**
FS1: R1 SHORT	-ve in high gain	LOW
FS2: R1 OPEN	zero volt follower	$HIGH$
FS3: R2 SHORT	$INVAMP_{unitygain}$	$HIGH$
FS4: R2 OPEN	NEGATIVE out of range	LOW
FS5: AMP L_DN	$INVAMP_{low}$	LOW
FS6: AMP L_UP	$INVAMP_{high}$	$HIGH$
FS7: AMP NOOP	$INVAMP_{nogain}$	LOW
FS8: AMP LowSlew	$slowoutput\frac{\delta V}{\delta t}$	$LOWPASS$

Collecting the symptoms of failure from table 5.3 a derived component, $INVAMP$, is formed where:

$$fm(INVAMP) = \{LOW, HIGH, LOWPASS\}.$$

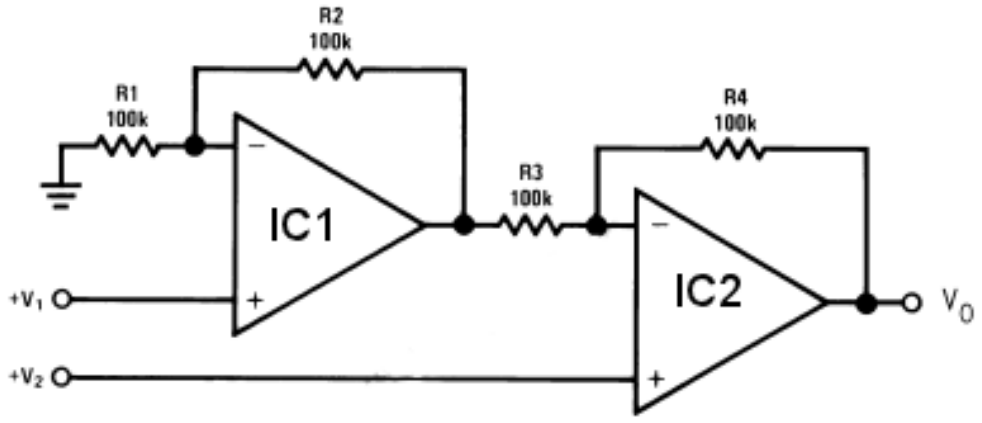
5.1.3 Comparison between the two approaches

The first analysis used two FMMD stages. The first stage analysed an inverted potential divider giving the derived component (IPD). The next stage analysed a functional grouping comprised of the IPD and an OpAmp. The second analysis (3 components) looked at the effects of each failure mode of each resistor and the op-amp. This meant more work for the analyst—that is an increase in the complexity of the analysis—compared to checking the two known failure modes from the pre-analysed inverted potential divider against the OpAmp. Both analysis strategies obtained the same failure modes for the inverting amplifier (i.e. the same failure modes for the derived component INVAMP).

5.1.4 Conclusion

All FMEA is performed in the context of the environment and functionality of the entity under analysis. This example shows that for the condition where the input voltage is constrained to being positive, two levels of decomposition can be applied. For the unconstrained case, i.e. where the input could be positive or negative, it is necessary to consider all three components as one larger functional grouping.

5.2 Differencing Amplifier using two op-amps



$$V_0 = \left(1 + \frac{R4}{R3}\right)(V_2 - V_1), \text{ where R1 = R4 and R2 = R3}$$

$$\text{As shown: } V_0 = 2(V_2 - V_1)$$

Figure 5.5: Differencing Amplifier using two op-amps.

The circuit in figure 5.5 amplifies the difference between the input voltages $+V1$ and $+V2$. The circuit is configured so that both inputs use the non-inverting (high impedance inputs) ensuring that they will not electrically load the previous stage. Because this differencing amplifier presents high impedance to both inputs, and only uses two amplifiers, this is a useful circuit wherever a high impedance differencing amplifier is required. This is a configuration that is commonly used in electronic circuits, it would therefore, be desirable to represent this circuit as a derived component called say $DiffAMP$. Identifying functional groupings from the components in the circuit is the starting point for analysis. Looking first at the components in the signal path, it can be noticed that a non-inverting amplifier is formed by R1,R2 and IC1. In fact, apart from being inverted visually on the schematic, it is identical to the example used in section 4.2 (the first practical example used to demonstrate FMMD). It is therefore possible to re-use the derived component NI_AMP and the failure modes for it, thus:

$$fm(NI\_AMP) = \{AMPHigh, AMPLow, LowPass\}.$$

5.2.1 The second stage of the amplifier

The second stage of this amplifier, following the signal path, is the amplifier consisting of $R3, R4$ and $IC2$. This is in exactly the same configuration as the first amplifier, but it is being fed by the first amplifier. The first amplifier was connected to ground via a resistor on its minus input and received as input '+V1' (explicitly a positive voltage from the schematic). This means the junction of R2 R3 is always +ve. This means the input

voltage '+V2' could be lower than this. This means R3 R4 is not a fixed potential divider, with R4 being on the positive side. It could be at either polarity. Here, even though R3 and R4 are used as a potential divider, it could be either inverted or non-inverted according to the voltages on the inputs. Therefore the resistors cannot be modelled as a potential divider, but must be placed in the functional grouping with the OpAmp and analysed.

Table 5.4: Second Amplifier SEC_AMP: Failure Mode Effects Analysis: Single Faults

Failure cause	SEC_AMP **Amplifier Effect**	**Symptom**
TC1: $OPAMP$ LatchUP	Output High	AMPHigh
TC2: $OPAMP$ LatchDown	Output Low : Low gain	AMPLow
TC3: $OPAMP$ No Operation	Output Low	AMPLow
TC4: $OPAMP$ Low Slew	Low pass filtering	LowPass
TC5: *R3_open*	+V2 follower	AMPIncorrectOutput
TC6: *R3_short*	Undefined (impedance of IC1 vs +V2)	AMPIncorrectOutput
TC5: *R4_open*	High or Low output +V2>+V1 $\mapsto$ High +V1>+V2 $\mapsto$ Low	AMPIncorrectOutput
TC6: *R4_short*	+V2 follower	AMPIncorrectOutput

Collecting the symptoms it can be seen that this amplifier fails in four ways. A derived component, SEC_AMP, is created with failure modes described by:

$$fm(SEC\_AMP) = \{AMPHigh, AMPLow, LowPass, AMPIncorrectOutput\}.$$

5.2.2 Final stage of the $DiffAmp$ Analysis

For the final stage a functional grouping consisting of two derived components of the type NI_AMP and SEC_AMP is created. FMMD analysis is applied to this functional grouping in table 5.5. Common symptoms

Table 5.5: Difference Amplifier $DiffAMP$: Failure Mode Effects Analysis: Single Faults

Failure cause	$DiffAMP$ **Effect**	**Derived Component Failure Mode**
TC1: NI_AMP AMPHigh	IC2 output driven high	DiffAMPLow
TC2: NI_AMP AMPLow	IC2 output driven low	DiffAMPHigh
TC3: NI_AMP LowPass	IC2 output with lag	DiffAMP_LP
TC4: SEC_AMP AMPHigh	Diff amplifier high	DiffAMPHigh
TC5: SEC_AMP AMPLow	Diff amplifier low	DiffAMPLow
TC6: SEC_AMP LowPass	Diff amplifier lag/lowpass	DiffAMP_LP
TC7: SEC_AMP IncorrectOutput	Output voltage is not proportional to $(V2 - V1)$	DiffAMPIncorrect

of failure are collected. A derived component to represent the failure mode behaviour of the differencing amplifier circuit (see figure 5.5) is created:

$$fm(DiffAMP) = \{DiffAMPLow, DiffAMPHigh, DiffAMP\_LP, DiffAMPIncorrect\}.$$

The failure analysis performed is represented as a directed graph in figure 5.6. Using this any top level fault can be traced back to a component failure mode that could have caused it[1]. This circuit performs poorly from a safety point of view. Its failure modes could be undetectable, i.e. indistinguishable from valid readings (especially when it becomes a V2 follower).

The failure mode $DiffAMPIncorrect$ may seem like a vague failure mode—however, this failure mode is impossible to detect in this circuit— in fault finding terminology [36, 58] this failure mode is said to be unobservable, and in EN61508 [95] terminology is an 'undetectable fault'. Were this failure to have safety implications, this FMMD analysis will have revealed this undetectable condition; this would likely prompt re-design of this circuit. A typical way to solve an undetectable fault such as this is to periodically switch in test signals in place of the input signal. Alternatively, two amplifiers could be used with different gains, from the same outputs, and the results scaled and then compared (perhaps in software) to detect any failure.

5.2.3 Conclusion

This example shows three stages of hierarchy, and a graph tracing the base component failure modes to the top level event. It also re-visits the decisions about membership of functional groupings, due to the con-

[1] An FTA diagram can be constructed from the information in this graph. A top level event is chosen and the DAG worked down through its edges using XOR gates.

text of the circuit raised in section 5.1.2. This FMMD analysis also revealed an undetectable failure mode, $DiffAMPIncorrect$.

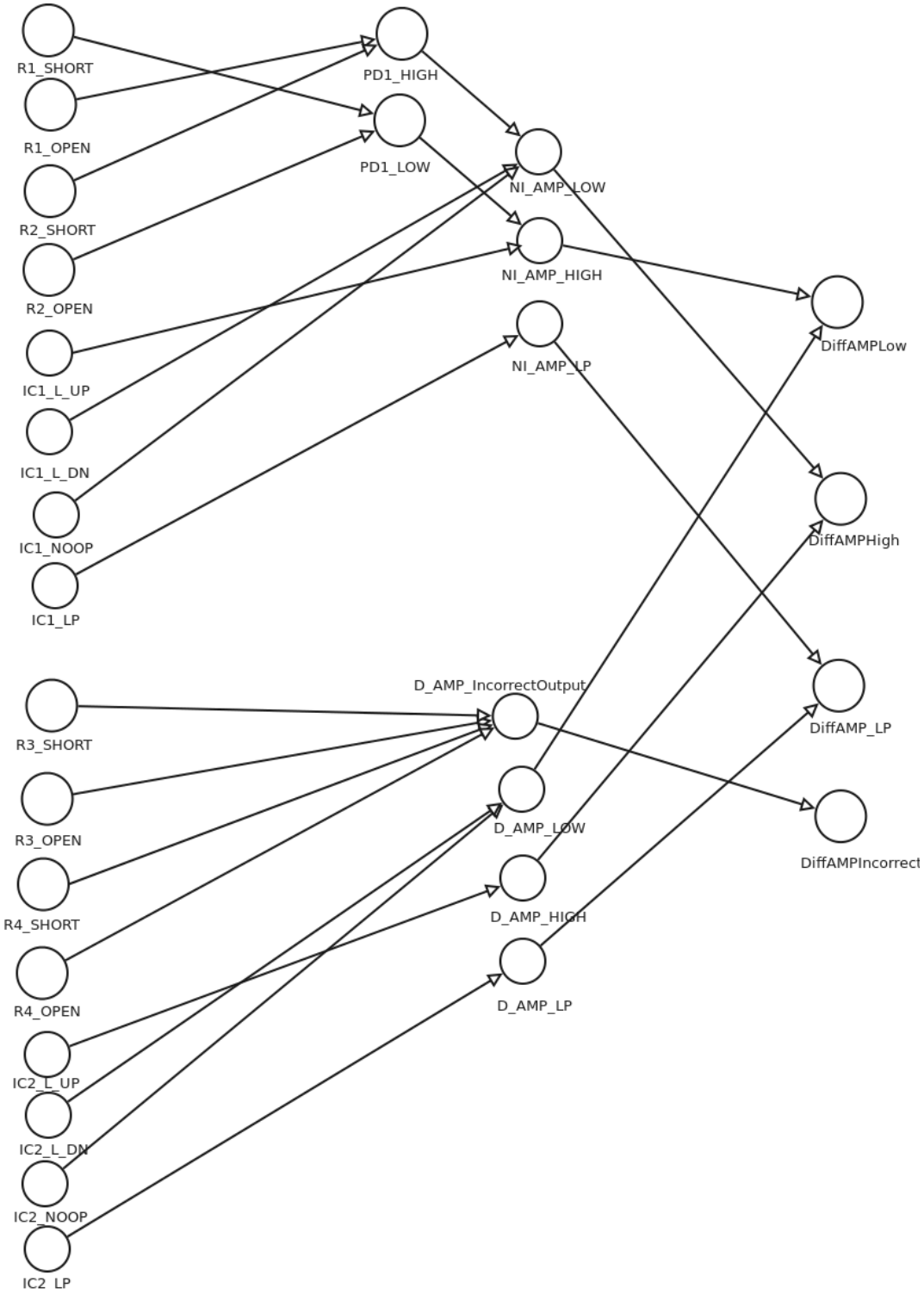


Figure 5.6: Directed Acyclic Graph of the two op-amp differencing amplifier failure modes

5.3 Five Pole Low Pass Filter, using two Sallen Key stages.

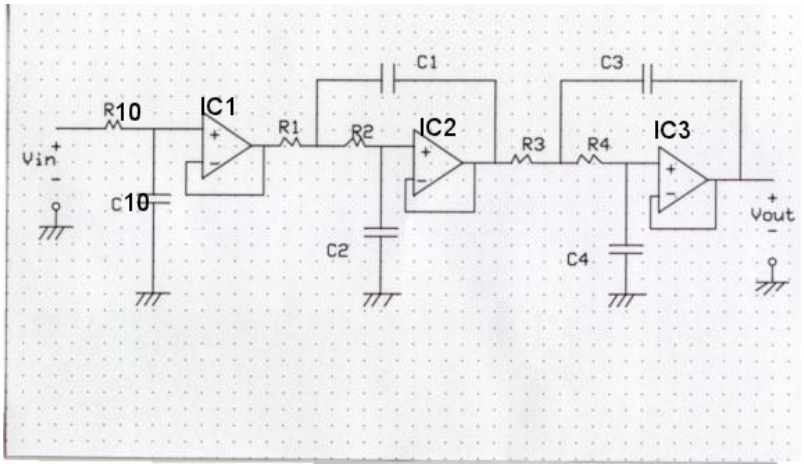


Figure 5.7: Five Pole Low Pass Filter, using two Sallen Key stages and three op-amps. An example of FMMD applied to a multi-stage but linear signal path topology.

The circuit in figure 5.7 shows a five pole low pass filter. Using the FMMD guidelines (see section 4.3.1.1), a top down view of the circuit is taken, and then groups of components performing specific tasks are identified. Starting at the input, there is a first order low pass filter buffered by an op-amp, the output of this is passed to a Sallen Key [78][p.267] [97][p.288] second order low-pass filter. The output of this is passed into another Sallen Key filter. The first Sallen Key low pass filter is analysed and then re-used for the second stage (avoiding repeat work that would have been performed using traditional FMEA).

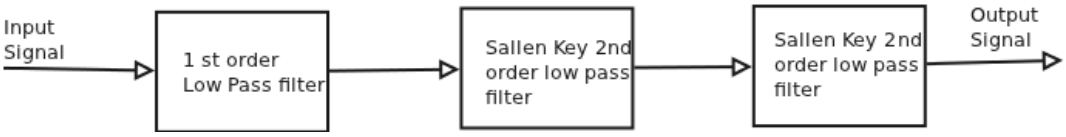


Figure 5.8: Signal Flow through the five pole low pass filter

5.3.1 First Order Low Pass Filter

Following the signal path from the input, the first order low pass filter formed by $R10$ and $C10$, is encountered. This configuration (or functional grouping) is very commonly used to remove unwanted high frequencies/noise from a signal. R10 and C10 act as a potential divider, with the crucial difference between a purely resistive potential divider being that the impedance of the capacitor is lower for higher frequencies. Thus higher frequencies are attenuated at the point its output signal is read/used. However, from a failure mode perspective it can be analysed in a very similar way to a potential divider (see section 4.2). Capacitors generally fail OPEN but some types fail OPEN and SHORT. The worst case for failure for capacitors is taken, i.e. OPEN and SHORT. The first order low pass filter is analysed in table 5.6.

The symptoms $\{LPnofilter, LPnosignal\}$ are collected and a derived component created called $FirstOrderLP$.

Table 5.6: FirstOrderLP: Failure Mode Effects Analysis: Single Faults

Failure cause	**First Order Low Pass Filter**	**Symptom**
FS1: R10 SHORT	$NoFiltering$	$LPnofilter$
FS2: R10 OPEN	$NoSignal$	$LPnosignal$
FS3: C10 SHORT	$NoSignal$	$LPnosignal$
FS4: C10 OPEN	$NoFiltering$	$LPnofilter$

Applying the fm function yields:

$$fm(FirstOrderLP) = \{LPnofilter, LPnosignal\}.$$

This simple filter is not robust to circuit loading, that is, in electronics terms it has a high output impedance. This means that were it to be overloaded by a subsequent stage of the circuit its signal processing properties could be altered.

5.3.2 Addition of Buffer Amplifier: First stage

The op-amp IC1 is being used simply as a buffer. By placing it between the stages on the signal path the possibility of unwanted signal feedback to the low-pass filter, formed by C10 and R10, is avoided. The buffer is one of the simplest op-amp configurations. It has no other components, and a functional grouping is formed from the $FirstOrderLP$ and the OpAmp component.

Table 5.7: First Stage LP1: Failure Mode Effects Analysis: Single Faults

Failure cause	**First stage LP1 Effect**	**Symptom**
TC1: $OPAMP$ LatchUP	Output High	LP1High
TC2: $OPAMP$ LatchDown	Output Low	LP1Low
TC3: $OPAMP$ No Operation	Output Low	LP1Low
TC4: $OPAMP$ Low Slew	Unwanted Low pass filtering	LP1filterincorrect
TC5: $LPnofilter$	No low pass filtering	LP1filterincorrect
TC6: $LPnosignal$	No input signal	LP1nosignal

From the table 5.7 three symptoms of failure of the first stage of this circuit (i.e. R10,C10,IC1) are observed. A derived component is created for it, $LP1$, where:

$$fm(LP1) = \{LP1High, LP1Low, LP1filterincorrect, LP1nosignal\}$$

In terms of the circuit, the functional groupings $FirstOrderLP$, and $LP1$ have been modelled. These can be represented on the circuit diagram by drawing contours around the components on the schematic in figure 5.9.

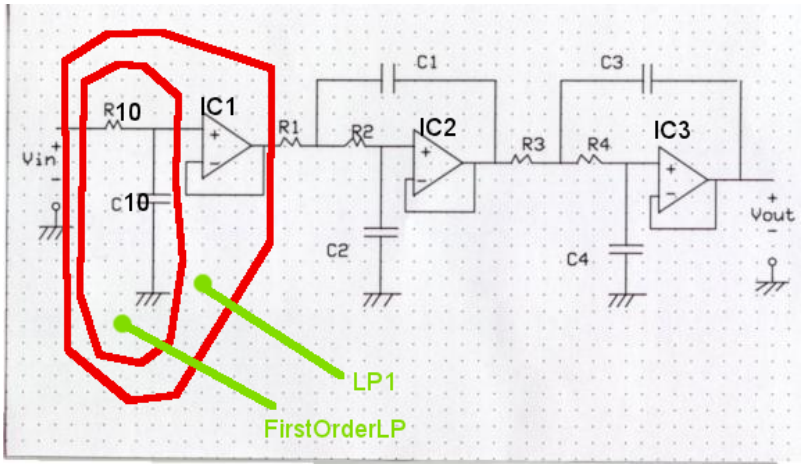


Figure 5.9: Five Pole Sallen Key Filter: Circuit showing the first two functional groupings modelled as an Euler diagram super-imposed onto the electrical schematic.

5.3.3 Second order Sallen Key Low Pass Filter

The next two filters in the signal path are the component groups R1,R2,C2,C1,IC2 and R3,R4,C4,C3,IC3. These are Sallen Key low pass filters [102]. From a failure mode perspective these are identical. The first one can be analysed (see table 5.8) and then these results re-used for the next stage of analysis (see figure 5.10).

Table 5.8: Sallen Key Low Pass Filter SKLP: Failure Mode Effects Analysis: Single Faults

Failure cause	**SKLP Effect**	**Symptom**
TC1: $OPAMP$ LatchUP	Output High	SKLPHigh
TC2: $OPAMP$ LatchDown	Output Low	SKLPLow
TC3: $OPAMP$ No Operation	Output Low	SKLPLow
TC4: $OPAMP$ Low Slew	Unwanted Low pass filtering	SKLPfilterIncorrect
TC5: R1 OPEN	No input signal	SKLPnosignal
TC6: R1 SHORT	incorrect low pass filtering	SKLPfilterIncorrect
TC7: R2 OPEN	No input signal	SKLPnosignal
TC8: R2 SHORT	incorrect low pass filtering	SKLPfilterIncorrect
TC9: C1 OPEN	reduced/incorrect low pass filtering	SKLPfilterIncorrect
TC10: C1 SHORT	reduced/incorrect low pass filtering	SKLPfilterIncorrect
TC11: C2 OPEN	reduced/incorrect low pass filtering	SKLPfilterIncorrect
TC12: C2 SHORT	No input signal, low signal	SKLPnosignal

A derived component is created to represent the Sallen Key low pass filter, called $SKLP$:

$$fm(SKLP) = \{SKLPHigh, SKLPLow, SKLPIncorrect, SKLPnosignal\}.$$

5.3.4 A failure mode model of the five pole Sallen Key filter

A derived component representing the three stages of this filter is created following the signal flow in the filter circuit (see figure 5.8). As the signal has to pass through each block/stage in order to be 'five pole' filtered, these three blocks are brought together to form a functional grouping. This will give a failure mode model for the whole circuit. The Sallen Key stages can be indexed, and these are marked on the circuit schematic in figure 5.10. So the final functional grouping will consist of the derived components $\{LP1, SKLP_1, SKLP_2\}$.

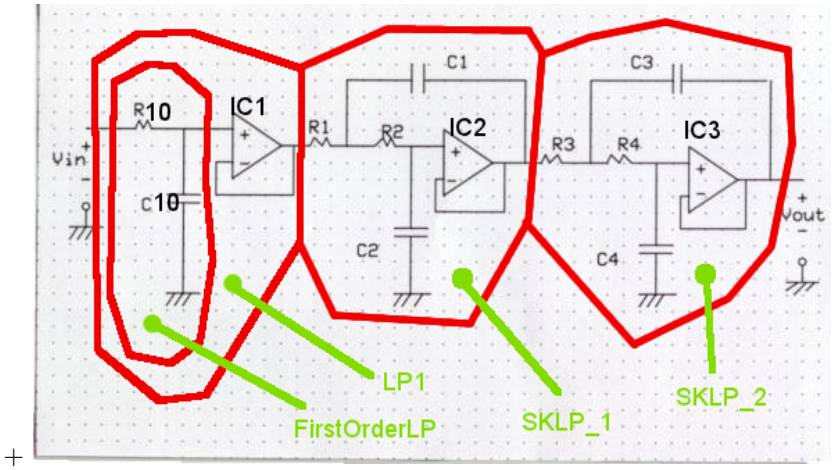


Figure 5.10: Functional Groupings in Five Pole Low Pass Filter. Shown as an Euler diagram super-imposed onto the electrical schematic.

This is analysed in table 5.9. The resulting FMMD hierarchy is shown in figure 5.11.

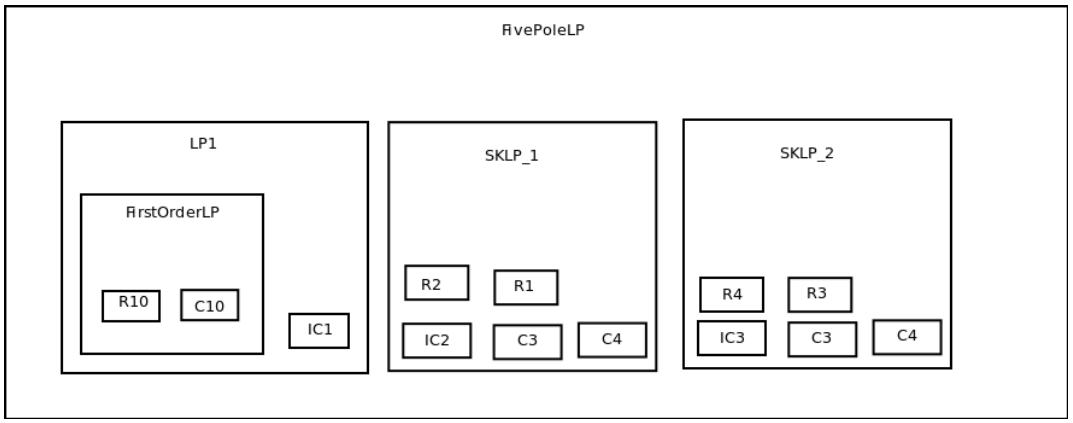


Figure 5.11: Euler diagram showing functional grouping/derived component relationships for the analysis of the Five Pole Sallen Key filter. This is an abstract version of figure 5.10

.

+

Table 5.9: Five Pole Low Pass Filter: Failure Mode Effects Analysis($FivePoleLP$): Single Faults

Failure cause	$FivePoleLP$ **Effect**	**Symptom**
TC1: $LP1$ LP1High	signal HIGH	HIGH
TC2: $LP1$ SKLPLow	signal LOW	LOW
TC3: $LP1$ LP1filterIncorrect	filtering incorrect	FilterIncorrect
TC4: $LP1$ LP1nosignal	no signal propagated	NO_SIGNAL
TC5: $SKLP_1$ High	signal HIGH	HIGH
TC6: $SKLP_1$ Low	signal LOW	LOW
TC7: $SKLP_1$ filterIncorrect	filtering incorrect	FilterIncorrect
TC8: $SKLP_1$ nosignal	no signal propagated	NO_SIGNAL
TC9: $SKLP_2$ High	signal HIGH	HIGH
TC10: $SKLP_2$ Low	signal LOW	LOW
TC11: $SKLP_2$ filterIncorrect	filtering incorrect	FilterIncorrect
TC12: $SKLP_2$ nosignal	no signal propagated	NO_SIGNAL

A derived component is created to represent the circuit in figure 5.7, called $FivePoleLP$: applying the fm function (see table 5.9) yields:

$$fm(FivePoleLP) = \{HIGH, LOW, FilterIncorrect, NO\_SIGNAL\}.$$

The failure modes for the low pass filters are very similar, and the propagation of the signal is simple (as it is never inverted). The circuit under analysis is – as shown in the block diagram (see figure 5.8) – three op-amp driven non-inverting low pass filter elements. It is not surprising therefore that they have very similar failure modes. From a safety point of view, the failure modes LOW, $HIGH$ and NO_SIGNAL could be easily detected; the failure symptom $FilterIncorrect$ is not detectable.

5.3.5 Conclusion

This example shows the analysis of a linear signal path circuit with three easily identifiable functional groupings and re-use of the Sallen-Key derived component.

5.4 Quad Op-Amp Oscillator

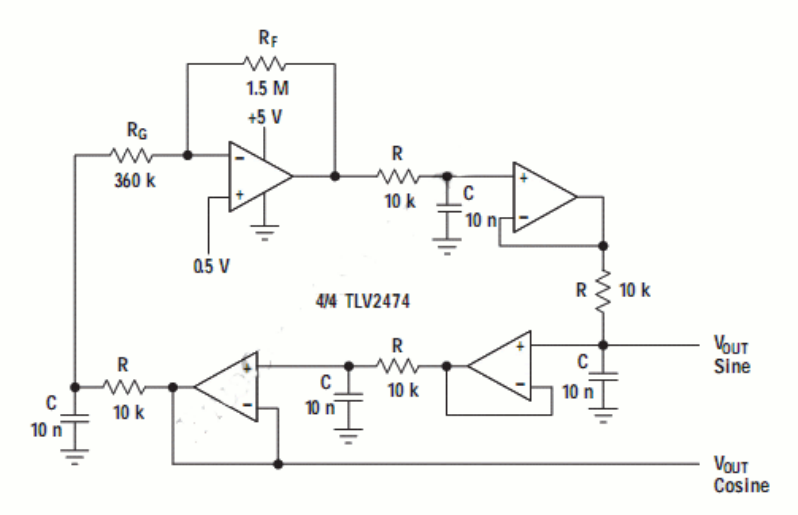


Figure 5.12: Circuit diagram for the Quad Op-Amp 'Bubba' Oscillator

This circuit is described in the Analog Applications Journal [59][p.37]. The circuit implements an oscillator using four 45 degree phase shifts, and an inverting amplifier to provide gain and the final 180 degrees of phase shift (making a total of 360). The circuit provides two outputs with a quadrature phase relationship. From a fault finding perspective this circuit cannot be decomposed, as the whole circuit is enclosed within a feedback loop, hence a fault anywhere in the loop is likely to affect all stages. However, this is not a problem for FMMD, as functional groupings are readily identifiable. Using the FMMD guidelines (see section 4.3.1.1), a top down view of the circuit is taken, and then groups of components performing specific tasks are identified. Initially three types of functional groupings are identified, an inverting amplifier (analysed in section 5.1), a 45 degree phase shifter (a $10k\Omega$ resistor and a $10nF$ capacitor) and a non-inverting buffer amplifier. These are named $INVAMP$, $PHS45$ and $NIBUFF$ respectively. These functional groupings are used to describe the circuit in block diagram form with arrows indicating the signal path, in figure 5.13.

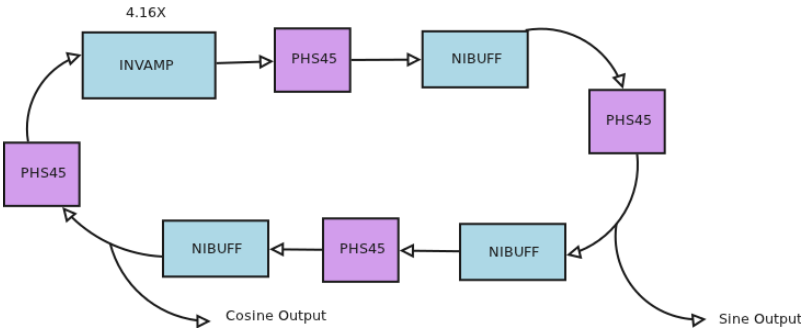


Figure 5.13: Circuit 3: Electrical signal path block diagram of the 'Bubba' oscillator, showing the circular circuit topology.

Each of these functional groupings are analysed to create failure mode models for them, and from these derived components determined.

5.4.1 Inverting Amplifier: INVAMP

The inverting amplifier was analysed in section 5.1 and can be re-used. i.e. the derived component $INVAMP$. This inverting amplifier, as a derived component, has the following failure modes:

$$fm(INVAMP) = \{AMP\_High, AMP\_Low, LowPass\}.$$

5.4.2 Phase shifter: PHS45

This consists of a resistor and a capacitor. Failure mode models exist for these components – $fm(R) = \{OPEN, SHORT\}$, $fm(C) = \{OPEN, SHORT\}$ – the question next is, how do these failure modes affect the phase shifter? Note that the circuit here is identical to the low pass filter in circuit topology (see section 5.3.1), but its intended use is different. Therefore this circuit is analysed from the perspective of it being a *phase shifter* not a *low pass filter*. The functional grouping for the phase shifter consists of a resistor and a capacitor, $G_0 = \{R, C\}$ (FMMD analysis details in appendix A.1.1),

$$fm(PHS45) = \{nosignal, 0\_phaseshift\}.$$

5.4.3 Non Inverting Buffer: NIBUFF.

The non-inverting buffer functional grouping is comprised of one component, an op-amp. The failure modes for an op-amp [24][p.3-116] are used to represent this group. The failure modes for the non-inverting buffer ($NIBUFF$) are expressed thus:

$$fm(NIBUFF) = fm(OPAMP) = \{L\_up, L\_dn, Noop, L\_slew\}.$$

5.4.4 Bringing the functional groupings Together: FMMD model of the 'Bubba' Oscillator.

At this point all the derived components could be collected into one large functional group (see figure 5.14) or merged in smaller stages, which would have the side-effect of creating intermediate derived components. Initially the first identified functional groupings are used to create the failure mode model without further stages of refinement/hierarchy.

5.4.5 FMMD Analysis using initially identified functional groupings

By indexing the re-used derived components the functional grouping for this analysis can be expressed thus:

$$G = \{PHS45_1, NIBUFF_1, PHS45_2, NIBUFF_2, PHS45_3, NIBUFF_3, PHS45_4, INVAMP\},$$

or in Euler diagram format in figure 5.14.

The detail of the FMMD analysis can be found in appendix A.1.2. Applying fm to the Bubba oscillator returns two failure modes,

$$fm(BubbaOscillator) = \{NO_{osc}, HI_{fosc}\}.$$

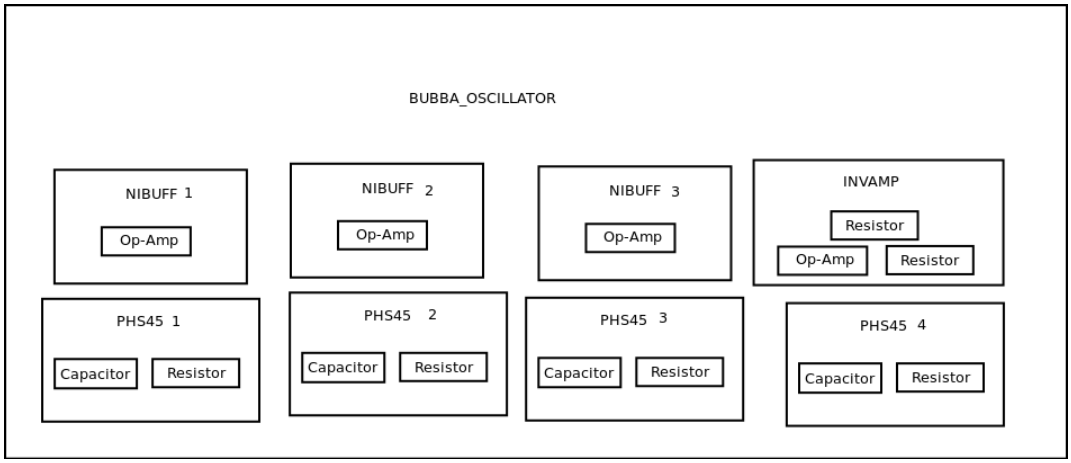


Figure 5.14: Euler diagram showing the hierarchy of the initial FMMD analysis performed on the Bubba Oscillator circuit.

The analysis here appears top-heavy; it should be possible to refine the model more and break this down into smaller functional groupings by allowing more stages of hierarchy. By decreasing the size of the modules with further refinement, new derived components may be discovered that could be of use for other analyses in the future.

5.4.6 FMMD Analysis of Bubba Oscillator using a finer grained modular approach (i.e. more hierarchical stages)

The example above—from the initial functional groupings—used one very large functional grouping to model the circuit. It should be possible to determine smaller functional groupings and refine the model further.

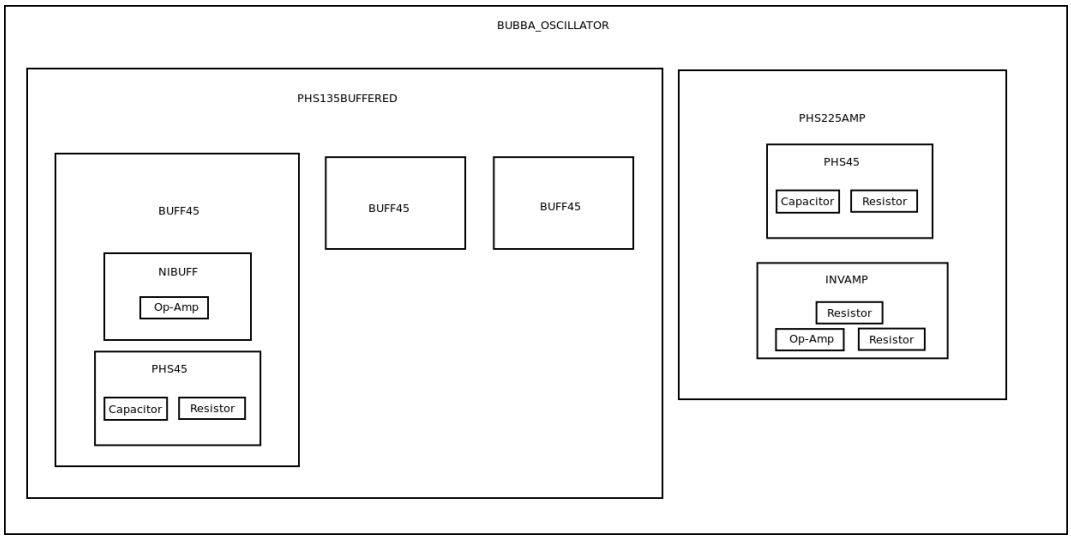


Figure 5.15: Euler diagram showing functional groupings for the Bubba oscillator using a more decomposed approach.

Outline of finer grained FMMD analysis of the Bubba oscillator. The pre-analysed $NIBUFF$ and $PHS45$ derived components are used to form a functional grouping, analysed in table A.3, giving the derived component $BUFF45$. $BUFF45$ is a derived component representing an actively buffered 45° phase shifter. From the block circuit diagram (figure 5.12), it is seen that there are three 45° phase shifter circuits in series. Together these apply a 135° phase shift to the signal. This property is used to model a higher level derived component, that of a 135° phase shifter. The three $BUFF45$ derived components form a functional grouping which is analysed in table A.4. The result of this analysis is the derived component $PHS135BUFFERED$ which represents an actively buffered 135° phase shifter. This is shown in the Euler diagram in figure 5.15.

Analysis details of the finer grained FMMD analysis of the Bubba oscillator. A PHS45 derived component and an inverting amplifier[2], form a functional grouping providing an amplified 225° phase shift, analysed in table A.5 resulting in the derived component $PHS225AMP$. Applying FMMD the derived component $PHS225AMP$ is created with the following failure modes:

$$fm(PHS225AMP) = \{180\_phaseshift, NO\_signal\}.$$

To complete the analysis we now bring the derived components $PHS135BUFFERED$ and $PHS225AMP$ together and perform FMEA with these (see appendix A.1.6), to obtain a model for the Bubba Oscillator.

$$fm(BUBBAOSC) = \{HI_{osc}, NO\_signal\}.$$

[2]Inverting amplifiers apply a 180° phase shift to a signal regardless of its frequency.

This more decomposed approach has identified five derived components, which could potentially be re-used in other projects.

5.4.7 Comparing both approaches

Smaller functional groupings signify less by-hand checks and a more finely grained model. This means that more derived components will be created and this increases the potential for re-use. A finer grained model—with potentially more hierarchy stages—also means that more reasoning stages, i.e. FMMD analysis stages with their associated analysis reports, have been created.

5.4.8 Conclusion

With FMMD there is always a choice for the membership of functional groupings. This example has shown that the simple approach, identifying initial functional groupings and using them to build a large functional grouping to model the circuit gives a valid result. However, it involves a large reasoning distance, the final stage having 24 failure modes to consider against each of the other seven derived components. A finer grained approach produces more potentially re-usable derived components and involved several stages with an overall lower reasoning distance. These reasoning distances, or complexity comparison figures are presented in the metrics chapter 7 in section 7.2.2. This example demonstrates that the finer grained models benefit from lower reasoning distances to determine the failure mode model.

5.5 Sigma Delta Analogue to Digital Converter ($\Sigma\Delta ADC$).

The following example is used to demonstrate FMMD analysis of a mixed analogue and digital circuit (see figure 5.16).

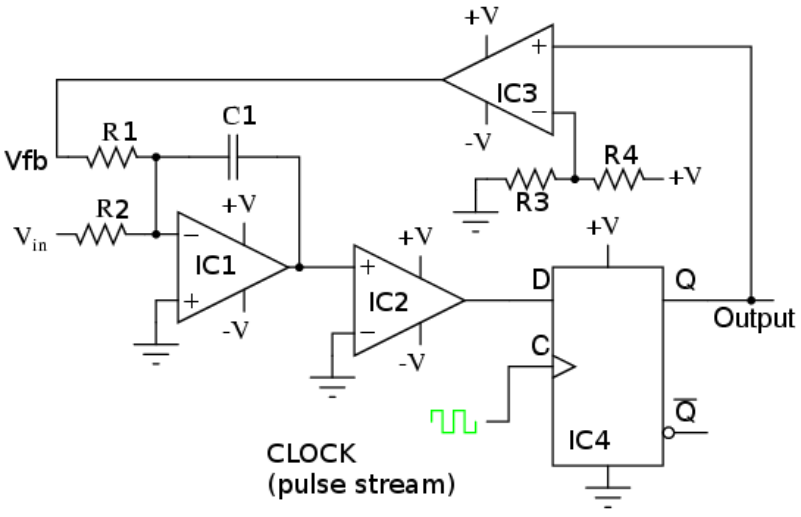


Figure 5.16: Sigma Delta Analogue to Digital Converter

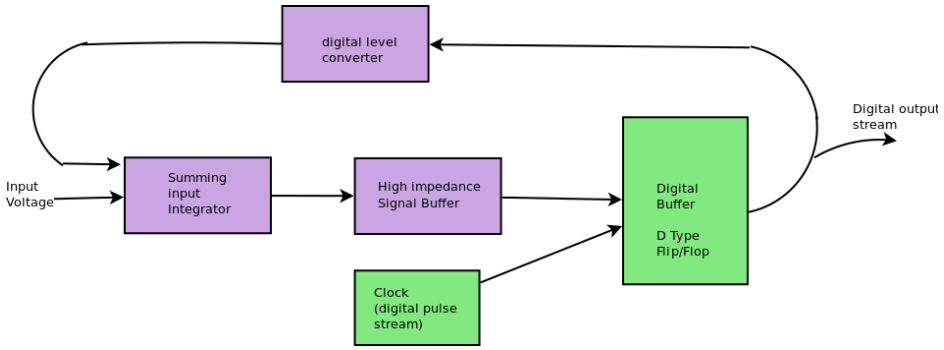


Figure 5.17: Electrical signal path Block diagram: $\Sigma\Delta ADC$

How the circuit works. A detailed description of $\Sigma\Delta ADC$ may be found in [50][pp.69-80]. The diagram in 5.17 shows the signal path used by this configuration for a $\Sigma\Delta ADC$. It works by placing the analogue voltage to be read into a mixed analogue and digital feedback circuit. A summing junction and integrator is used to compare the negative feedback signal with the input. The output of the integrator is converted to a digital level (by IC2) and fed to the D type flip flop. The output of the flip flop is routed to the digital output and to the feedback loop. It must be level converted, i.e. from digital logic voltage levels to analogue levels, before being fed to the analogue feedback/integrator. It is level converted to an analogue signal by IC3—i.e. a digital 0 becomes a -ve voltage and a digital 1 becomes a +ve voltage— and fed into the summing integrator completing the negative feedback loop. In essence this implements an over-sampling one bit analogue to digital converter [108][pp.729-730]. The output of the flip flop forms a bit pattern representing the value of the input voltage (i.e. the value of the sum of 1's and 0's is proportional to the voltage value at the input).

5.5.1 FMMD analysis of $\Sigma\Delta ADC$

The parts for the $\Sigma\Delta ADC$ are a mixture of analogue (resistors, capacitors, OpAmps) and digital (D type flip flop, and a digital clock). The failure modes of all components are examined in this circuit below. IC1,IC2 and IC3 are all OpAmps and have failure modes for this component type (i.e. from section 4.2):

$$fm(OPAMP) = \{HIGH, LOW, NOOP, LOW\_SLEW\}.$$

The literature was examined for a failure model for a D-type flip flop [24][3-105], and the CD4013B [87] chosen. Its failure modes are expressed using the fm function:

$$fm(CD4013B) = \{HIGH, LOW, NOOP\}.$$

The resistors and capacitor failure modes are taken from EN298 [14][An.A]. The failure modes for the resistors (R) and capacitors (C) are expressed thus:

$$fm(R) = \{OPEN, SHORT\},$$

$$fm(C) = \{OPEN, SHORT\}.$$

A CLOCK signal is required for the $\Sigma\Delta ADC$. For the purpose of example one failure mode is assigned to this, that it might stop. The failure mode of the CLOCK, is stated thus:

$$fm(CLOCK) = \{STOPPED\}.$$

5.5.2 Identifying initial functional groupings

5.5.2.1 Summing Junction Integrator (SUMJINT)

The next stage is to choose initial (base) functional groupings. The most obvious way to find initial functional groupings is to follow the signal path. The signal path is circular, but can be started with the input voltage, which is applied via $R2$, this voltage is labelled V_{in}. The feedback voltage for the ADC is supplied via $R1$, this voltage is called V_{fb}. $R2$ and $R1$ form a summing junction to IC1: they balance the integrator provided by the capacitor C1 and the opamp IC1. This can be the first functional grouping and it is analysed in appendix A.2.1:

$$FG = \{R1, R2, IC1, C1\}.$$

That is, the failure modes (see FMMD analysis at A.2.1) of the new derived component $SUMJINT$ are:

$$fm(SUMJINT) = \{V_{in}DOM, V_{fb}DOM, NO\_INTEGRATION, HIGH, LOW\}.$$

5.5.2.2 High Impedance Signal Buffer (HISB)

Next in the signal path (see figure 5.17) is a signal buffer. This presents a high impedance to the circuit driving it. This prevents electrical loading, and thus interference with, the SUMJINT stage. This is simply an op-amp with the input connected to the +ve input and the -ve input grounded. This is an OpAmp in a signal buffer configuration and therefore simply has the failure modes of an Op-amp. As it is performing one particular

function it can be considered as a derived component, a High Impedance Signal Buffer (HISB). This is analysed using FMMD in appendix A.2.2. The derived component $HISB$ is created and its failure modes stated as:

$$fm(HISB) = \{HIGH, LOW, NOOP, LOW_{SLEW}\}.$$

5.5.2.3 Digital level to analogue level conversion ($DL2AL$).

The integrator is implemented in analogue electronics, but the output from the D type flip flop is a digital signal. A conversion stage is required to interface these elements. Digital level to analogue level conversion is performed by IC3 in conjunction with a potential divider formed by R3,R4. The potential divider provides a mid rail reference voltage to the inverting input of IC3.

Potential divider formed by R3,R4. The analysis from table 4.1 is re-used, i.e. the derived component PD represents the potential divider formed by R3 and R4.

$$fm(PD) = \{HIGH, LOW\}.$$

IC3 is an op-amp and has the failure modes

$$fm(IC3) = \{HIGH, LOW, NOOP, LOW\_SLEW\}.$$

The digital signal is supplied to the non-inverting input. The output is a voltage level in the analogue domain $-V$ or $+V$. A functional grouping is formed from PD and $IC3$.

$$FG = \{PD, IC3\}.$$

This functional grouping is analysed (see appendix A.2.3) giving:

$$fm(DL2AL) = \{LOW, HIGH, LOW\_SLEW\}.$$

5.5.2.4 $DIGBUF$ — digital clocked memory (flip-flop).

The digital element of the $\Sigma\Delta ADC$, is a 'one bit memory', or D type flip flop. This buffers the feedback result and provides the output bit stream. A functional grouping is created from the CLOCK and IC4 derived components to model this digital buffer,

$$FG = \{IC4, CLOCK\}.$$

This functional grouping (see appendix A.2.4) is now analysed giving the derived component $DIGBUF$: where

$$fm(DIGBUF) = \{LOW, STOPPED\}.$$

5.5.3 First functional groupings analysed

The initial functional groupings have been analysed giving the first derived components. These are:

- SUMJINT — A summing junction and integrator,
- HISB — A high impedance buffer,

- DIGBUF — A digital one bit buffer/memory,
- DL2AL — A digital to analog level converter.

These derived components follow the signal path shown in figure 5.17. These derived components can now be used to create higher level functional groupings. These are represented in the Euler diagram in figure 5.18. They are later used to create functional groupings to make a complete failure mode for the $\Sigma\Delta ADC$.

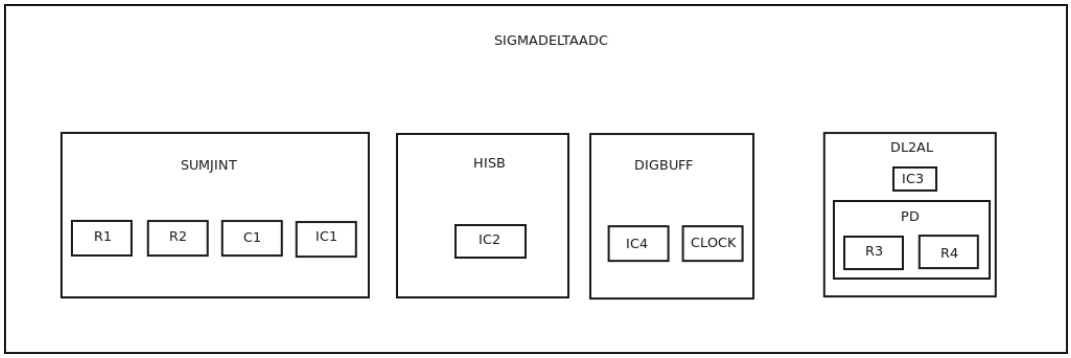


Figure 5.18: Euler diagram showing the initial derived components used to model the $\Sigma\Delta ADC$

5.5.3.1 Buffered Integrating Summing Junction (BISJ): functional grouping of $HISB$ and $SUMJINT$

A functional grouping with the two derived components $HISB$ and $SUMJINT$ is now created. This forms a buffered integrating summing junction functional grouping i.e. $FG = \{HISB, SUMJINT\}$. This is analysed using FMMD (see appendix A.2.5) giving the derived component $BISJ$:

$$fm(BISJ) = \{OUTPUTSTUCK, REDUCED\_INTEGRATION\}.$$

5.5.3.2 Flip Flop Buffer (FFB): functional grouping of $DL2AL$ and $DIGBUF$

The functional grouping formed by $DIGBUF$ and $DL2AL$ takes the flip flop clocked and buffered value, and outputs it at analogue voltage levels for the summing junction. $FG = \{DIGBUF, DL2AL\}$.

The buffered flip flop circuitry is analysed (see appendix A.2.6) and the derived component FFB created, where:

$$fm(FFB) = \{OUTPUTSTUCK, LOW\_SLEW\}.$$

5.5.4 Final, top level functional grouping for sigma delta Converter

The FMMD model now has just two derived components, FFB and $BISJ$. These together represent all base components within this circuit. A final functional grouping is formed with these:

$$FG = \{FFB, BISJ\}.$$

The buffered $\Sigma\Delta ADC$ circuit is analysed using FMMD (see appendix A.2.7) giving a derived component $SDADC$ which provides a failure mode model for the $\Sigma\Delta ADC$:

$$fm(SDADC) = \{OUTPUT\_OUT\_OF\_RANGE, OUTPUT\_INCORRECT\}.$$

The derived component hierarchy is shown in figure 5.19.

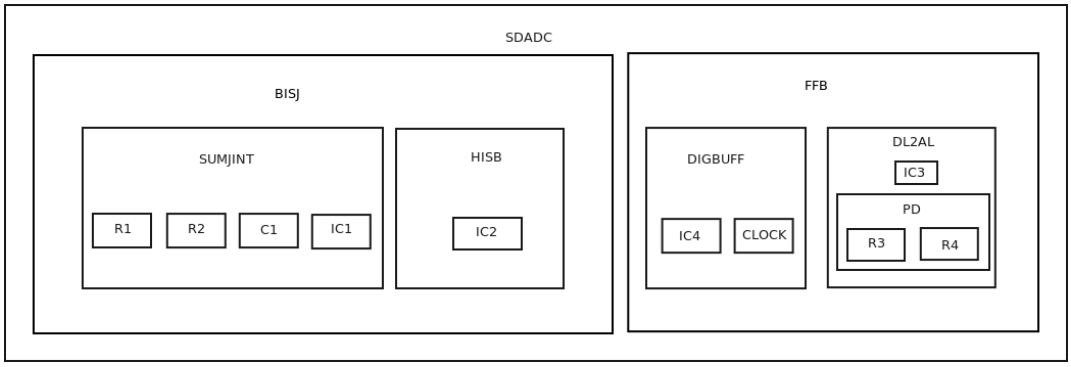


Figure 5.19: Euler diagram showing the final derived components used to model the $\Sigma\Delta ADC$

5.5.5 Conclusion

The $\Sigma\Delta ADC$ example, shows that FMMD can be applied to mixed digital and analogue circuitry: which means that analysis of the analogue/digital interface is achievable using FMMD. This leads onto interfacing to software and digital systems in the next chapter.

5.6 Pt100 Analysis: FMMD and Double Failure Mode Analysis

For this example an industry standard temperature measurement circuit, the 'four wire Pt100', is examined. The four wire Pt100 configuration is a commonly used and well known safety critical circuit. Applying FMMD provides a fresh look at this established circuit. It is analysed for both single and double failures, in addition it demonstrates FMMD coping with component parameter tolerances. The circuit is described from a conventional safety perspective and then analysed using the FMMD methodology. The Pt100, or platinum wire 100Ω sensor is a widely used industrial temperature sensor that is slowly replacing the use of thermocouples in many industrial applications below 600^oC, due to high accuracy[78].

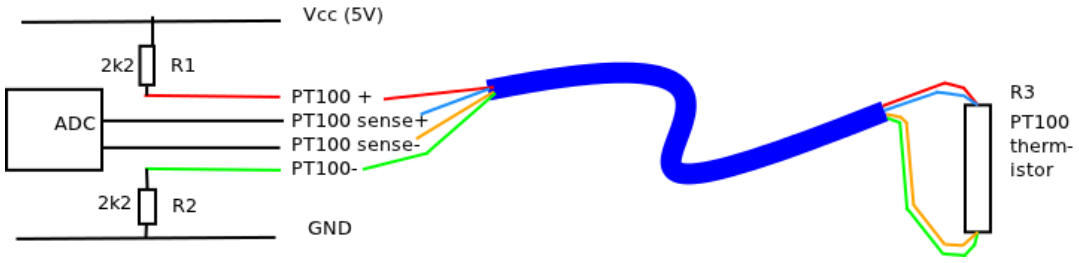


Figure 5.20: Pt100 four wire circuit

5.6.1 General Description of Pt100 four wire circuit

The Pt100 four wire circuit uses two wires to supply a small electrical current, and returns two sense voltages over the other two. By measuring voltages from sections of this circuit, which forms potential dividers, the resistance of the platinum wire sensor can be determined. The voltage ranges expected from this three stage potential divider[3] are shown in figure 5.21. Note that there is an expected range for each low and high reading, for a given temperature span. The low reading goes down as temperature increases, and the higher reading goes up. For this reason the low reading will be referred to as *sense-* and the higher as *sense+*.

Accuracy despite variable resistance in cables. Resistance from the supply cables causes a slight voltage drop in the supply to the $Pt100$. As no significant current is carried by the two 'sense' lines, the resistance back to the ADC causes only a negligible voltage drop, and thus a four wire configuration is more accurate[4].

Calculating Temperature from the sense line voltages. The current flowing though the whole circuit can be measured on the PCB by reading a third sense voltage from one of the load resistors. Knowing the current flowing through the circuit and knowing the voltage drop over the $Pt100$, its resistance is calculated by Ohms law $V = I.R$, $R = \frac{V}{I}$. The resistance to temperature conversion is achieved through published $Pt100$ tables[29]. Standard voltage divider equations (see figure 5.22 and equation 5.1) can be used to calculate expected voltages for failure mode and temperature reading purposes.

[3]Two stages are required for validation, a third stage is necessary to measure the current flowing through the circuit to obtain accurate temperature readings.

[4]The increased accuracy is because the voltage measured is the voltage across the thermistor only and not the voltage across the thermistor and current supply wire resistance.

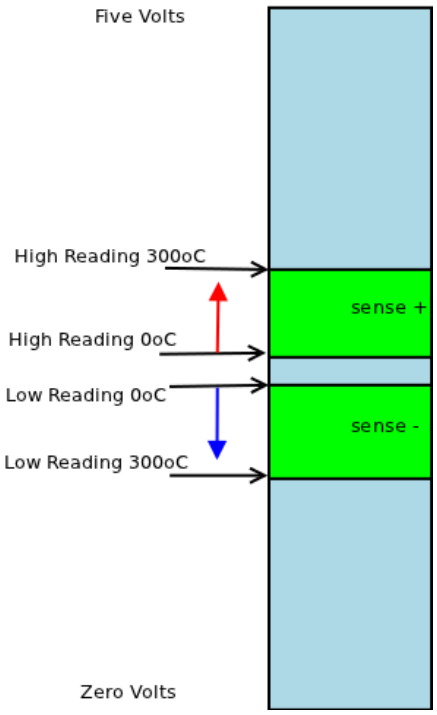


Figure 5.21: Pt100 expected voltage ranges for a temperature range of 0° to 300° centigrade

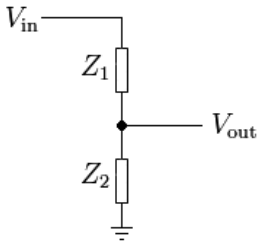


Figure 5.22: Voltage Divider

$$V_{out} = V_{in}.\frac{Z_2}{Z_2 + Z_1} \tag{5.1}$$

5.6.2 Safety case for 4 wire circuit: Detailed calculations

The following analysis of the Pt100 circuit firstly presents an FMEA analysis which is then supported by detail and calculations of the type that would be submitted to an approval agency. Detailed potential divider calculations and the effect of component tolerances are factored for germane test cases. The analysis is presented in the FMEA table 5.10. The next section 5.7, extends this analysis for double failure scenarios.

Single Fault FMEA Analysis of $Pt100$ Four wire circuit. The Pt100 circuit consists of three resistors, two 'current supply' wires and two 'sensor' wires. Resistors, are considered to fail by either going OPEN or SHORT. For the purpose of this analyis; R_1 is a $2k2\Omega$ from 5V to the thermistor, R_3 is the Pt100 thermistor and R_2, also $2k2\Omega$, connects the thermistor to ground.

The terms 'High Fault' and 'Low Fault' are defined here with reference to figure 5.21. Should a reading be outside the safe green zone in the diagram, it will be considered a fault. Should the reading be above its expected range, this is a 'High Fault' and if below a 'Low Fault'. Table 5.10 plays through the scenarios of each of the resistors failing in both SHORT and OPEN failure modes, and hypothesises error conditions in the readings. The temperature range 0^oC to 300^oC will be used to determine potential divider voltage outputs, and these later used to validate the FMEA in table 5.10.

Table 5.10: Pt100 FMEA Single Faults

Test Case	**Result sense +**	**Result sense -**	**Symptom**
R_1 SHORT	High Fault	-	Value Out of Range Value
R_1 OPEN	Low Fault	Low Fault	Both values out of range
R_3 SHORT	Low Fault	High Fault	Both values out of range
R_3 OPEN	High Fault	Low Fault	Both values out of range
R_2 SHORT	-	Low Fault	Value Out of Range Value
R_2 OPEN	High Fault	High Fault	Both values out of range

From table 5.10 it can be seen that any single component failure in the circuit will cause a common symptom, that of one or more of the values being 'out of range'.

Consideration of Resistor Tolerance. The Pt100 element is a precision part and will be chosen for a specified accuracy/tolerance range. One or other of the load resistors (the one that current is measured over) should also be of this accuracy.

The $2k2\Omega$ loading resistors may be ordinary, in that they would have a good temperature co-efficient (typically $\leq 50(ppm)\Delta R \propto \Delta^oC$), and typically be subjected to a narrow temperature range, being mounted on a PCB. To calculate the resistance of the Pt100 element the voltage over it, i.e. sense+ minus sense-, is read and with the current flowing through it, its resistance can be found. Let R_2 be used to measure the current flowing in the temperature sensor loop. As these calculations are performed by Ohms law, which is linear, the accuracy of the reading will be determined by the accuracy of R_2 and R_3[5].

Range and $Pt100$ Calculations. $Pt100$ resistors are designed to have a resistance of 100Ω at 0^oC [78],[29]. A suitable expected temperature range was considered to be 0^oC to 300^oC for a given application. According to the Eurotherm Pt100 tables [29], this corresponded to the resistances 100Ω and 212.02Ω respectively. From this the potential divider circuit can be analysed and the maximum and minimum acceptable voltages determined. These can be used as bounds results to validate the Pt100 FMEA analysis. As the Pt100 forms a potential divider with the $2k2\Omega$ load resistors, the upper and lower readings are calculated thus:

$$sense+ = 5V.\frac{2k2 + Pt100}{2k2 + 2k2 + pt100}$$

[5]To calculate the resistance of the Pt100 we need the current flowing though it. This can be determined via Ohms law applied to R_2, $V = IR_2$, $I = \frac{V}{R_2}$, and then using I, $R_3 = \frac{V_{R3}}{I}$.

and

$$sense- = 5V.\frac{2k2}{2k2 + 2k2 + Pt100} .$$

So by defining an acceptable measurement/temperature range, and ensuring the values are always within these bounds, there should be confidence that none of the resistors in this circuit have failed. To convert these to twelve bit ADC ($\mathcal{ADC}_{12}$)[6] counts:

$$sense+ = 2^{12}.\frac{2k2 + Pt100}{2k2 + 2k2 + pt100}$$

and

$$sense- = 2^{12}.\frac{2k2}{2k2 + 2k2 + Pt100}.$$

Table 5.11 gives ranges that determine correct operation. It will be shown that for any single error (shorting

Table 5.11: Pt100 Maximum and Minimum Values

Temperature	**Pt100 resistance**	**sense-**	**sense+**	**Description**
0 ^oC	100Ω	2.44V	2.56V	Boundary of
		2002$\mathcal{ADC}_{12}$	2094$\mathcal{ADC}_{12}$	out of range LOW
+300 ^oC	212.02Ω	2.38V	2.62V	Boundary of
		1954$\mathcal{ADC}_{12}$	2142$\mathcal{ADC}_{12}$	out of range HIGH

or opening of any resistor) this bounds check will detect it.

Single Fault FMEA Analysis of $Pt100$ Four wire circuit. This circuit supplies two results, the *sense+* and *sense-* voltage readings. To establish the valid voltage ranges for these, and knowing the valid temperature range for this example (0^oC .. 300^oC) valid voltage reading ranges have been calculated by using the standard voltage divider equation 5.1 for the circuit shown in figure 5.22.

Proof of Out of Range Values for Failures. Using the temperature ranges defined above the voltages calculated can be used to verify correct operation of the circuit; it is shown that the resistor failures, OPEN and SHORT, would cause 'out of range' voltages. There are six test cases and each will be examined in turn.

TC 1 : Voltages R_1 SHORT. Since sense+, because R_1 is shorted, is directly connected to the 5V rail this will be out of range. The sense- reading will be determined by the potential divider formed by $R2$ and R_3. This is calculated over the temperature range,
for 0°:

$$sense- = 5V.\frac{2k2}{2k2 + 100\Omega} = 4.78V,$$

and for 300°:

$$sense- = 5V.\frac{2k2}{2k2 + 212.02\Omega} = 4.56V.$$

Thus with R_1 shorted both readings are outside the proscribed range in table 5.11.

[6] An $\mathcal{ADC}_{12}$ with a 5V Vref is assumed for this example. Raw ADC counts would typically be used in software routines validating range/values in safety critical readings.

TC 2 : Voltages R_1 OPEN. In this case the 5V rail is disconnected. All voltages read are 0V, and therefore both readings are outside the proscribed range in table 5.11.

TC 3 : Voltages R_2 SHORT. This failure mode creates a potential divider formed by R1 and R3. This means that the sense+ and sense- lines will have voltages on them determined by this potential divider. Since with R_2 shorted the sense- is directly connected to the 0V rail, the sense- reading will be out of range. For sense+ voltages must be calculated over the extremes of the acceptable temperature range, and it must be ensured that these voltages could not lead to false readings. With Pt100 at 0^oC:

$$sense+ = 5V.\frac{100\Omega}{2k2 + 100\Omega} = 0.218V.$$

With Pt100 at the high end of the temperature range 300^oC,

$$sense+ = 5V.\frac{212.02\Omega}{2k2 + 212.02\Omega} = 0.44V.$$

Thus with R_2 shorted both readings are outside the proscribed range in table 5.11.

TC 4 : Voltages R_2 OPEN. Here there is no potential divider operating and both sense lines will read 5V, outside of the proscribed range.

TC 5 : Voltages R_3 SHORT. Here the potential divider is simply between the two 2k2 load resistors. Thus it will read a nominal 2.5V. Because the readings here depend on the values of resistors R_1 and R_2 resistor tolerance must be considered. Assuming the load resistors are fairly typical in terms of precision; taking a worst case of 1% either way:

$$5V.\frac{2k2 \times 0.99}{2k2 \times 1.01 + 2k2 \times 0.99} = 2.475V$$

and

$$5V.\frac{2k2 \times 1.01}{2k2 \times 1.01 + 2k2 \times 0.99} = 2.525V\ .$$

These readings both lie outside the proscribed ranges. Also the sense+ and sense- readings would have the same value.

TC 6 : Voltages R_3 OPEN. Here the potential divider is broken. The sense- will read 0V and the sense+ will read 5V. Both readings are outside the proscribed range.

5.6.3 Summary of Analysis

All six test cases have been examined and where necessary voltages calculated for the failure conditions. The results agree with the FMEA presented in table 5.10. For this circuit there is a common and easily detected symptom for all these single resistor faults—that of—'voltage out of range'. In practical use, by defining an acceptable measurement/temperature range, and ensuring the values are always within these bounds, there is confidence that none of the resistors in this circuit have failed.

5.6.4 Derived Component $Pt100$ analysed for single failure modes.

The Pt100 circuit can now be treated as a component in its own right, and has one failure mode, **O**UT_OF_RANGE i.e.:

$$fm(Pt100) = \{OUT\_OF\_RANGE\}.$$

This is a single, detectable failure mode. The detectability of fault conditions is very good with this circuit. This should not be a surprise, as the four wire $Pt100$ has been developed for safety critical temperature measurement.

5.7 Pt100 Double Simultaneous Fault Analysis

In this section the failure mode behaviour for the Pt100 is examined for double failures. Traditional FMEA methodologies do not provide double failure analysis [53][p.342] and double failure analysis for FMEA is a subject of current research [82, 76]. All the single failures have been analysed in the last section. Table 5.12 lists all possible combinations of double faults as FMMD test cases.

Table 5.12: Pt100 FMEA Double Faults

TC number	**Test Case**	**Result sense +**	**Result sense -**	**Symptom**
TC 7:	R_1 OPEN R_2 OPEN	Floating input Fault	Floating input Fault	Unknown value readings
TC 8:	R_1 OPEN R_2 SHORT	low	low	Both out of range
TC 9:	R_1 OPEN R_3 OPEN	Floating	low	Sense- out of range
TC 10:	R_1 OPEN R_3 SHORT	low	low	Both out of range
TC 11:	R_1 SHORT R_2 OPEN	high	high	Both out of range
TC 12:	R_1 SHORT R_2 SHORT	high	low	Both out of range
TC 13:	R_1 SHORT R_3 OPEN	high	low	Both out of Range
TC 14:	R_1 SHORT R_3 SHORT	high	high	Both out of range
TC 15:	R_2 OPEN R_3 OPEN	high	Floating input Fault	sense+ out of range
TC 16:	R_2 OPEN R_3 SHORT	high	high	Both out of Range
TC 17:	R_2 SHORT R_3 OPEN	high	low	Both out of Range
TC 18:	R_2 SHORT R_3 SHORT	low	low	Both out of Range

TC 7 : Voltages R_1 OPEN R_2 OPEN This double fault mode produces an interesting symptom. Both sense lines are floating. The $\mathcal{ADC}_{12}$ readings on them cannot be predicted. In practise these would probably float to low or high values but for the purpose of a safety critical analysis, all that can be stated is that the values are 'floating' and 'unknown'. This is an interesting case, because it is, at this stage an undetectable fault. Undetectable faults are generally to be avoided in a safety critical environment [17, 43].

TC 8 : Voltages R_1 OPEN R_2 SHORT This cuts the supply from Vcc. Both sense lines will be at zero. Thus both values will be out of range.

TC 9 : Voltages R_1 OPEN R_3 OPEN. Sense+ will be floating. Sense- will be tied to ground and will thus be out of range.

TC 10 : Voltages R_1 OPEN R_3 SHORT. This shorts ground to both of the sense lines. Both values will be out of range.

TC 11 : Voltages R_1 SHORT R_2 OPEN. This shorts both sense lines to Vcc. Both values will be out of range.

TC 12 : Voltages R_1 SHORT R_2 SHORT. This shorts the sense+ to Vcc and the sense- to ground. Both values will be out of range.

TC 13 : Voltages R_1 SHORT R_3 OPEN. This shorts the sense+ to Vcc and the sense- to ground. Both values will be out of range.

TC 14 : Voltages R_1 SHORT R_3 SHORT. This shorts the sense+ and sense- to Vcc. Both values will be out of range.

TC 15 : Voltages R_2 OPEN R_3 OPEN. This shorts the sense+ to Vcc and causes sense- to float. The sense+ value will be out of range.

TC 16 : Voltages R_2 OPEN R_3 SHORT. This shorts the sense+ and sense- to Vcc. Both values will be out of range.

TC 17 : Voltages R_2 SHORT R_3 OPEN. This shorts the sense- to ground, and sense+ to Vcc. Both values will be out of range.

TC 18 : Voltages R_2 SHORT R_3 SHORT. This shorts the sense+ and sense- to ground. Both values will be out of range.

Symptom Extraction, forming a derived component. The results of the test case analysis can now be examined and symptom abstraction applied. In all the test case results there is at least one out of range value, except for TC_7 which has two unknown values/floating readings. All the faults, except TC_7, are aggregated into the symptom OUT_OF_RANGE. As a symptom TC_7 could be described as $FLOATING$. The Pt100 circuit again, can now be treated as a component in its own right, and has two failure modes, **OUT_OF_RANGE** and **FLOATING**, i.e.

$$fm(Pt100) = \{OUT\_OF\_RANGE, FLOATING\}.$$

Chapter 6

Applying FMMD to Software and Hybrid Systems

6.1 Software and Hardware Failure Mode Concepts

In this chapter it is shown that FMMD can be applied to both software and electronics enabling us to build complete failure models of typical modern safety critical systems. With modular FMEA i.e. FMMD the concepts of failure modes of components, functional groupings and symptoms of failure have been defined. A programmatic function has similar attributes to an FMMD functional grouping. An FMMD functional grouping is placed into a hierarchy, likewise a software function is typically placed into the hierarchy of its call-tree. A software function calls other functions and uses data sources which could be viewed as its 'components': it has outputs, i.e. it can perform actions on data or hardware. It is shown below that a software function can be mapped to an FMMD functional grouping: its failure modes are the failure modes of the software components it calls and/or the hardware from which it reads values. Its outputs are the data it changes, or the hardware actions it performs. When a software function has been analysed—using failure conditions of its inputs as a source of failure modes—its symptoms of failure can be defined (i.e. how functions that call it will see its failure mode behaviour).

FMMD is applied to software functions by viewing them in terms of their failure mode behaviour. That is to say, using FMMD, software functions are treated like functional groupings of electronic components. As software already fits into a hierarchy, there one less analysis decision to make when compared to analysing electronics. For electrical and mechanical systems, although the original system designers concepts of modularity and sub-systems in design may provide guidance, applying FMMD means deciding on the members for functional groupings and the subsequent hierarchy. With software already written, the hierarchies are given. To apply FMMD to software, the elements used by a software function are collected along with the function itself to form a functional grouping. When the failure mode behaviour of this software functional grouping has been analysed and its failure mode symptoms collected, a derived component can be created. That derived component can be used by functions that call the function just analysed. This software analysis can be applied from the bottom-up on the software call tree, until a complete failure mode hierarchy of the system

under investigation has been built.

6.1.1 Software, a natural hierarchy

Software written for safety critical systems is usually constrained to be modular [95][3] and non recursive [61][15.2]. Because of this direct call trees can be assumed[1]. Functions call functions from the top down and eventually call the lowest level library or IO functions that interact with hardware.

What is potentially difficult with applying FMMD to a software function, is deciding how to map its component failure modes and its symptoms of failure in a manner compatible with the FMMD process. With electronic components, the literature points to suitable sets of failure modes [24] [26] [14]. With software only some library functions are well known and rigorously documented enough to have the equivalent of known failure modes, most software is 'bespoke'. A different strategy is required to describe the failure mode behaviour of software functions; concepts from contract programming can be used to assist in this.

6.1.2 Contract programming description

Contract programming [67] is a discipline for building software functions in a controlled and traceable way. Each function is subject to pre-conditions (constraints on its inputs), post-conditions (constraints on its outputs) and function wide invariants (rules).

Mapping contract 'pre-condition' violations to component failure modes. A precondition, or requirement for a contract software function defines the correct ranges of input conditions for the function to operate successfully. A software function is considered to be a collection of code, functions called and variables used. In this way it is similar to an electronic circuit, which is a collection of components connected in a specific way. Using this analogy for software, the connections are the functions code, and the called functions/variables/inputs are the components. Erroneous behaviour from called functions and variables/inputs has the same effect as component failure modes on an electronic functional grouping. If it is considered that called functions and variables/inputs are the components of a function, a modular and hierarchical failure mode model from existing software can be built. Thus for FMMD applied to software, a violation of a pre-condition is considered to be equivalent to a failure mode of 'one of its components'.

Mapping contract 'post-condition' violations to symptoms. A post-condition is a definition of correct behaviour of a function. A violated post-condition is a symptom of failure, or, in FMMD terms a derived failure mode, for a function. Post conditions could relate to either actions performed (i.e. the state of hardware changed) or an output value of a function. In pure contract programming, a violation of a pre-condition would cause the function to **not** be executed. In implementation code, a pre-condition violation should cause an error to be generated, and thus a post-condition to fail. A function can fail for reasons other than corruption of its input data (i.e. failure caused by variables it uses or return values from functions it calls). Variables can become corrupted, by radiation affecting RAM [75, 37] or by another software function erroneously overwriting variables [4]. Current work on software FMEA generally focuses on mapping variable corruption to failure

[1] A typical embedded system will have a run time call tree, and (possibly multiple) interrupt sourced call trees.

modes [73, 44, 28, 91]. However, errors other than variable corruption can occur. For instance a microprocessor may have subtle bugs in its instruction set, or incorrectly handled interrupt contention [88] which could cause side effects in software. For the failure mode model of any software function, it must be considered that all failure modes defined by post-condition violations could simply occur.

Mapping contract 'invariant' violations to symptoms and failure modes. Invariants are conditions that are considered to be relied on throughout the execution of a program. Here they are taken to mean invariants applying to data or conditions that the function under analysis deals with or could be affected by. Invariants in contract programming may apply to inputs to the function (where violations can be considered failure modes in FMMD terminology), and to outputs (where violations can be considered symptoms, or derived failure modes, in FMMD terminology).

6.1.3 Combined Hardware/Software FMMD

For the purpose of example, a simple common safety critical industrial circuit that is nearly always used in conjunction with a programmatic element has been chosen. A common method for delivering a quantitative value in analogue electronics is to supply a current signal to represent the value to be sent [78][p.934]. Commonly, $4mA$ represents a zero or starting value and $20mA$ represents the full scale, and this is referred to as $4 \rightarrow 20mA$ signalling[2]. Using current instead of voltage to transmit an analogue value has intrinsic electrical safety advantages mainly due to current being constant in a circuit (Kirchoff's current law [51][p.160]). What is sent as current is what will arrive at the receiving end.

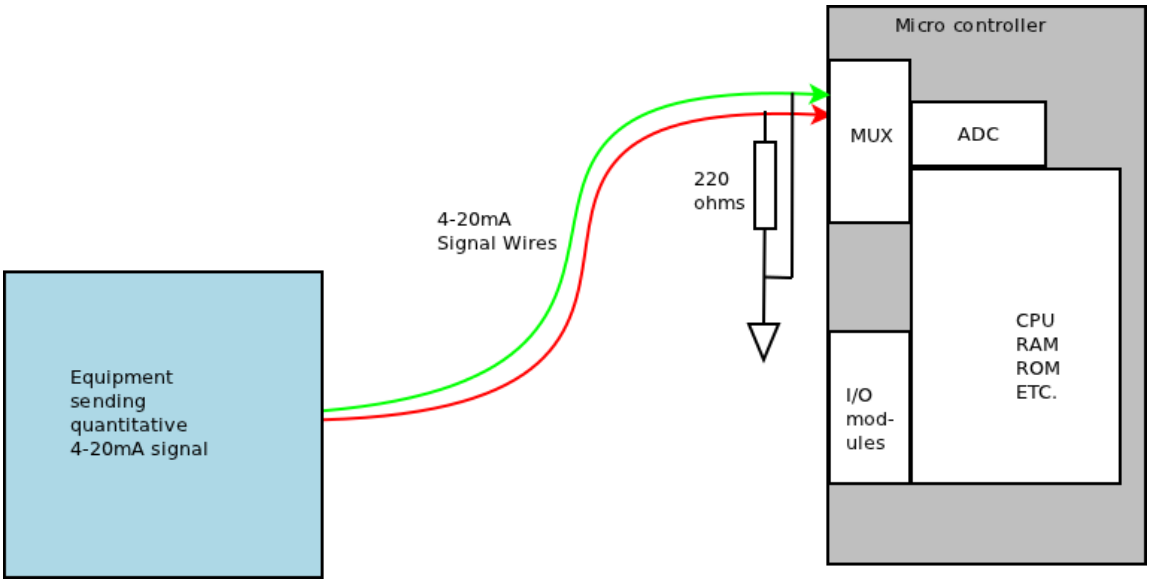


Figure 6.1: Context Diagram for $4 \rightarrow 20mA$ loop

The diagram in figure 6.1, shows some equipment which is sending a $4 \rightarrow 20mA$ signal to a micro-controller system. The signal is locally driven through a load resistor, and then read into the micro-controller via an

[2]Various current ranges have been used for value sending via electrical current, $10 \rightarrow 50mA$, being one other range used. However $4 \rightarrow 20mA$ signalling has emerged as the industry standard over the last few decades.

ADC and its multiplexer. With the voltage determined at the ADC, the intended quantitative value from the external equipment is read.

6.2 Simple Software Example: Reading a $4{\rightarrow}20mA$ input into software

Consider a software function that reads a $4{\rightarrow}20mA$ input, and returns a value between 0 and 999 (i.e. per mil $0/_{00}$) representing the current detected; plus an additional error indication flag.

The $4{\rightarrow}20mA$ input circuitry used in the example and its related software, are accepted practise and in common use, and therefore its failure mode behaviour is well known and understood. For this reason it is a good example to use for comparing the results from FMMD analysis with known failure mode behaviour from the field/direct experience of engineers.

From figure 6.1 the $4{\rightarrow}20mA$ detection is via a 220Ω resistor and the voltage is read from an ADC into the software. Because the signal is specified as $4{\rightarrow}20mA$ any value outside the 4mA to 20mA range can be defined as an error condition. As voltage (rather than current) is read by an ADC, Ohms law [78] is used to determine the mA current detected: $V = IR$, $0.004A \times 220\Omega = 0.88V$ and $0.020A \times 220\Omega = 4.4V$. The acceptable voltage range[3] is therefore

$$(V \geq 0.88) \wedge (V \leq 4.4) \ .$$

This voltage range forms an input requirement and can be considered as an invariant condition i.e. both a pre-condition and a postcondition; for the system to be operating correctly the voltage should be within the above bounds. For the purpose of example the 'C' programming language [49] is used. In 'C' a function is declared with parenthesis to differentiate it from other types of variables (data types or pointers). In this document this format is borrowed, hence the C language function called 'main' would be presented as **main()**. The software function that performs a conversion from the voltage read to a per mil representation of the $4{\rightarrow}20mA$ input is now discussed. The function **read_4_20_input()** takes a floating point value for the voltage read, checks that it is within bounds, and then applies a conversion to a per-mil value which it returns via a pointer. The source code is presented in figure 6.2. A function **read_ADC()** is assumed that returns a floating point value which represents the voltage read (see code sample in figure 6.3).

The function called by **read_4_20_input()**, **read_ADC()** is now examined; this returns a voltage for a given ADC channel. This function deals directly with the hardware in the micro-controller on which the software is running. The function **read_ADC()**s' job is to select the correct channel (ADC multiplexer) and then to initiate a conversion by setting an ADC 'go' bit (see code sample in figure 6.3). It takes the raw ADC reading and converts it into a floating point[4] voltage value.

[3]For the purpose of clarity resistor tolerance has been ignored. In a practical $4{\rightarrow}20mA$ reader resistor tolerance would be factored into the limits, or 'dead-bands' of $\approx \frac{1}{2}mA$ at either end of the range would be implemented.

[4]the type, 'double' or 'double precision', is a standard C language floating point type [49].

```
/*********************************************/
/* read_4_20_input()                         */
/*********************************************/
/* Software function to read 4mA to 20mA input */
/* returns a value from 0-999 proportional     */
/* to the current input.                       */
/*********************************************/
int  read_4_20_input ( int * value ) {
  double input_volts;
  int error_flag;

  /* require: input from ADC to be
              between 0.88 and 4.4 volts */


  input_volts = read_ADC(INPUT_4_20_mA);

  if ( input_volts < 0.88 || input_volts > 4.4 ) {
    error_flag = 1; /* Error flag set to TRUE */
  }
  else {
    *value = ((input_volts - 0.88) / ( 4.4 - 0.88 )) * 999.0;
    error_flag = 0; /* indicate current input in range */
  }

  /* ensure: value is proportional (0-999) to the
             4 to 20mA input                      */

  return error_flag;
}
```

Figure 6.2: Software Function: **read_4_20_input()**

```
/*********************************************/
/* read_ADC()                                */
/*********************************************/
/* Software function to read voltage from a  */
/* specified ADC MUX channel                 */
/* Assume 10 ADC MUX channels 0..9           */
/* ADC_CHAN_RANGE = 9                        */
/* Assume ADC is 12 bit and ADCRANGE = 4096  */
/* returns voltage read as double precision  */
/*********************************************/
double  read_ADC( int  channel ) {
  int timeout = 0;
  int dval = -3.0;
  /* require: a) input channel from ADC to be
              in valid ADC range
              b) voltage ref is 0.1% of 5V     */

  /* return out of range result  */
  /* if invalid channel selected */
  if ( channnel > ADC_CHAN_RANGE )
     return -2.0;
  /* set the multiplexer to the desired channel */
  ADCMUX = channel;
  ADCGO = 1; /* initiate ADC conversion hardware */
  /* wait for ADC conversion with timeout */
  while ( ADCGO == 1 || timeout < 120 )
     timeout++;
  if ( timeout < 100 )
       /* the following converts ADC12 counts to voltage */
       dval = (double) ADCOUT * 5.0 / ADCRANGE;
  else
       dval = -1.0; /* indicate invalid reading */
  /* return voltage as a floating point value */
  /* ensure: value is voltage input to within 0.1% */
  return dval;
}
```

Figure 6.3: Software Function: **read_ADC()**

A very simple software structure, a call tree, shown in figure 6.4 has been obtained. This software is above

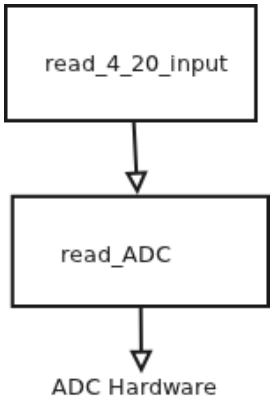


Figure 6.4: Call tree for software example

the ADC hardware in the conceptual call tree—from a programmatic perspective—the software is reading values from the 'lower level' electronics. The hardware is simply a load resistor, connected across an ADC input pin on the micro-controller and ground. The resistor and the ADC module of the micro-controller are identified as the base components in this design. FMMD is now applied, from the bottom-up, starting with the hardware.

6.2.1 FMMD Process

Hardware only Functional Grouping - Convert mA to Voltage - CMATV. This functional grouping, G_1, contains the load resistor and the physical Analogue to Digital Converter (ADC). G_1 is thus a set of base components: $G_1 = \{R, ADC\}$. It is a hardware only functional grouping. The failure modes of all the components in the functional grouping G_1 are now determined. For the resistor the failure mode set from the literature [14] is used. Where the function fm returns a set of failure modes for a given component:

$$fm(R) = \{OPEN, SHORT\}.$$

For the ADC the following failure modes are determined:

- STUCKAT — The ADC outputs a constant value,
- MUXFAIL — The ADC cannot select its input channel correctly,
- LOW — The ADC output is always LOW, or zero ADC counts,
- HIGH — The ADC output is always HIGH, or maximum ADC counts.

We can use the function fm to define the failure modes of an ADC thus:

$$fm(ADC) = \{STUCKAT, MUXFAIL, LOW, HIGH\}.$$

With these failure modes defined, analysis can begin on the functional grouping G_1, see table 6.1.

Common failure symptoms are now collected for G_1, these being $\{HIGH, LOW, V\_ERR\}$. Using the common failure symptoms a derived component is created, $CMATV$ (an an acronym for *Convert milli-amps to*

Table 6.1: functional grouping G_1: Failure Mode Effects Analysis

Failure cause	**Failure Effect**	**Symptom**
1: R_{OPEN}	resistor open, voltage on pin high	$HIGH$
2: R_{SHORT}	resistor shorted, voltage on pin low	LOW
3: $ADC_{STUCKAT}$	ADC reads out fixed value	V_ERR
4: $ADC_{MUXFAIL}$	ADC may read wrong channel	V_ERR
5: ADC_{LOW}	output low	LOW
6: ADC_{HIGH}	output high	$HIGH$

Voltage). As its failure modes are the collected symptoms of failure from the functional grouping G_1, the failure modes for the new derived component are:

$$fm(CMATV) = \{HIGH, LOW, V\_ERR\}.$$

Software and hardware hybrid functional grouping — RADC. The software function **Read_ADC()** uses the ADC hardware analysed as the derived component CMATV above. The code fragment in figure 6.3 states pre-conditions, as */* require: a) input channel from ADC to be in valid ADC range b) voltage ref is 0.1% of 5V */*. From the above contractual programming requirements, it is seen that the function must be sent the correct channel number. A violation of this can be considered a failure mode for the function, which is termed $CHAN\_NO$. The reference voltage for the ADC has a 0.1% accuracy requirement. If the reference value is outside this, it is also a failure mode of this function, which is termed V_REF[5]. Taken as a component for use in FMEA/FMMD the function has two failure modes. It can also fail its post condition, which is given the symptom VV_ERR. Therefore it can be treated as a generic component, **Read_ADC()**, by stating:

$$fm(\textbf{Read\_ADC()}) = \{CHAN\_NO, VREF, VV\_ERR\}$$

With the failure mode model for this function, it is used in conjunction with the ADC hardware derived component CMATV, to form a functional grouping G_2, where $G_2 = \{CMATV, \textbf{Read\_ADC()}\}$. This functional grouping is analysed in table 6.2. The common symptoms of failure from table 6.2 are collected giving $\{VV\_ERR, HIGH, LOW\}$. A derived component called $RADC$ is created with failure modes of:

$$fm(RADC) = \{VV\_ERR, HIGH, LOW\}.$$

This derived component is a hybrid of software and hardware, and is an example of a hardware interface modelled by FMMD.

[5]The failure mode V_REF is detectable only if a test input is used to measure a high precision voltage reference. This validates the supply voltage to the ADC. This is common practise for safety critical readings when using an ADC.

Table 6.2: functional grouping G_2: Failure Mode Effects Analysis

Failure cause	**Failure Effect**	**Symptom**
1: $CHAN\_NO$	wrong voltage read	VV_ERR
2: $VREF$	ADC volt-ref incorrect	VV_ERR
3: $CMATV_{V\_ERR}$	voltage value incorrect	VV_ERR
4: $CMATV_{HIGH}$	ADC may read wrong channel	$HIGH$
5: $CMATV_{LOW}$	output low	LOW
6: post-condition fails	software fails C function: **Read_ADC()**	VV_ERR

Functional Group - Software - voltage to per mil - VTPM. The next function higher in the call tree is **read_4_20_input()**: This function calls the function **Read_ADC()** which is a member of the functional grouping from which the derived component $RADC$ was derived. The pre-conditions for the function **read_4_20_input()** are examined to determine its failure modes. Its one pre-condition is, */* require: input from ADC to be between 0.88 and 4.4 volts */*. A violation of this pre-condition can become the failure mode VRNGE (an acronym for Voltage Range); we state, $fm(\textbf{read\_4\_20\_input()}) = \{RI_{VRNGE}\}$. To this we add the post-condition, *ensure: value is proportional (0-999) to the* $4{\rightarrow}20mA$ *input*, which can be termed VAL_ERR: the failure modes for **read_4_20_input()** are now defined as:

$$fm(\textbf{read\_4\_20\_input()}) = \{RI_{VRNGE}, RI_{VAL\_ERR}\}.$$

A functional grouping, G_3, is formed with the derived component $RADC$ and the software component **read_4_20_input()**, i.e. $G_3 = \{\textbf{read\_4\_20\_input()}, RADC\}$. The failure symptoms for the functional grouping are $\{OUT\_OF\_RANGE, VAL\_$ For single failures these are the two ways in which this function can fail. An OUT_OF_RANGE condition will be flagged by the error flag variable, a detectable failure mode. The VAL_ERR will simply mean that the value read is incorrect: an undetectable failure mode and therefore undesirable condition. Finally a derived component is created to represent a failure mode model for the combined hardware and software $4{\rightarrow}20mA$ input. This can be named $R420I$, for *read* $4{\rightarrow}20mA$ *input*. This derived component has the following failure modes:

$$fm(R420I) = \{OUT\_OF\_RANGE, VAL\_ERR\}.$$

This software/hardware FMMD analysis is represented as a hierarchical diagram, see figure 6.5.

6.2.2 Conclusion: $4{\rightarrow}20mA$ Reader Software/Hardware FMMD Model

The derived component representing the hybrid software and hardware $4{\rightarrow}20mA$ reader demonstrates that FMMD can integrate software and electrical FMMD models. With this analysis a complete 'reasoning path'

Table 6.3: G_3: **Read_4_20()**: Failure Mode Effects Analysis

Failure cause	**Failure Effect**	**Derived Component Failure Mode**
1: RI_{VRGE}	voltage outside range	OUT_OF_RANGE
2: RI_{VAL_ERR} post-condition fails	software fails	VAL_ERR
3: $RADC_{VV\_ERR}$	voltage incorrect	VAL_ERR
4: $RADC_{HIGH}$	voltage value incorrect	VAL_ERR
5: $RADC_{LOW}$	ADC low voltage so out of range i.e. $< 0.88V$	OUT_OF_RANGE

linking the failures modes from the electronics to those in the software has been created. Each functional grouping to derived component transition represents a reasoning stage[6]. Each reasoning stage will have an associated analysis report[7]. With traditional FMEA methods the reasoning distance is large, because it stretches from the component failure mode to the top or system level failure. For this reason applying traditional FMEA to software stretches the reasoning distance even further. This is exacerbated by the fact that traditional SFMEA is performed separately from Hardware FMEA (HFMEA) [91, 71], additionally even the software/hardware interfacing is usually treated as a separate FMEA task [74, 40, 73]

A derived component for a $4\rightarrow20mA$ input in software has now been defined. Typically, more than one such input could be present in a real-world system. This derived component could thus be re-used.

The unsolved symptoms, or undetectable errors, i.e. VAL_ERR could be addressed by another software function to read other known signals via the multiplexer (MUX) (i.e. voltage references). This strategy would detect ADC_STUCK_AT and MUX_FAIL failure modes. Where the integrity of the MUX is very demanding, separate pull down test lines may be implemented on the germane inputs as well. A software specification for a hardware interface will typically concentrate on data formats, how to interpret raw readings, or what digital signals to apply for actuators [74]. The FMMD process naturally determines failure mode models for the hardware/software interface.
The $4\rightarrow20mA$ example above is based on the paper presented to System Safety in 2012 [20].

[6]Each of these reasoning stages, will have a reasoning distance associated with it, and because functional groupings are generally small XFMEA can be applied within those stages without undue state explosion problems.

[7]Having an analysis report for each functional grouping in a system analysed under FMMD, automatically provides a context sensitive documentation trail, improving accessibility to anyone re-viewing or auditing the analysis.

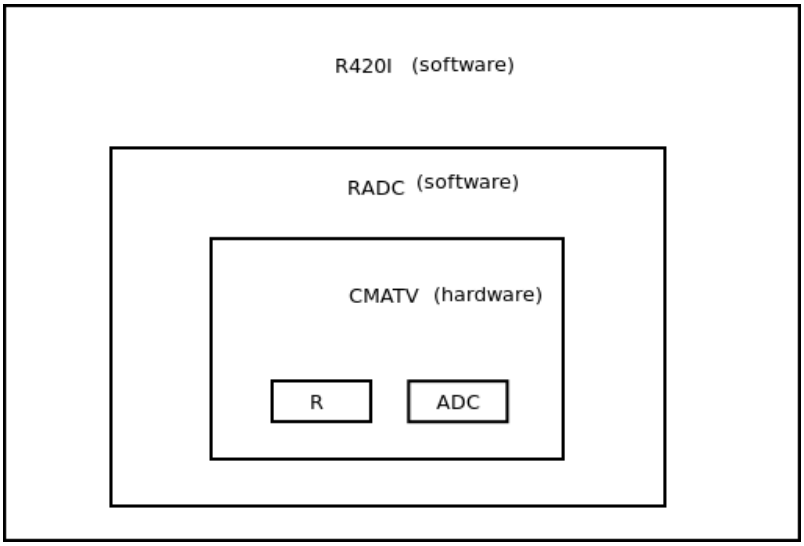


Figure 6.5: Electronics and Software shown in an integrated failure mode model—an Euler diagram showing relationship between derived components determined from electronics and software—the two outermost contours are software functions, and the inner two are electronic derived components.

6.3 Closed Loop Control Hardware/Software Hybrid Example

It is desirable to model a complete standalone system with FMMD, not only a standalone system, but ideally a hybrid software/hardware system. Temperature control is typically a first order differential problem, and is often addressed using the Proportional Integral Differential (PID) algorithm [35][p.66]. Traditionally this was performed in analogue electronics with trimmer potentiometers providing the P, I and D parameters. Since the introduction of digital computers, it has been possible to implement PID in software. A PID temperature controller is presented as a complete example of an electronic/hardware hybrid analysed using FMMD.

6.3.1 Design Stage: Implementation on a micro-controller.

When designing a computer program it is often useful to start with a system overview. A structured analysis 'Yourdon' context diagram [109] is presented below, see figure 6.6. Using figure 6.6 the system in terms of its data flow is reviewed, starting with the data sources (the Pt100 temperature sensor inputs) and the data sinks (the heater output and the LED indicators). There are two voltage inputs (see section 5.6) from the Pt100 temperature sensor. For the Pt100 sensor, the voltages it outputs are read and this requires an ADC and MUX. For the output, a Pulse Width Modulator (PWM) can be used (this is a common module found on micro-controllers facilitating variable power output [78][p.360]). PWM's ADC's and MUX's are commonly built into cheap micro-controllers [63][Ch.15].

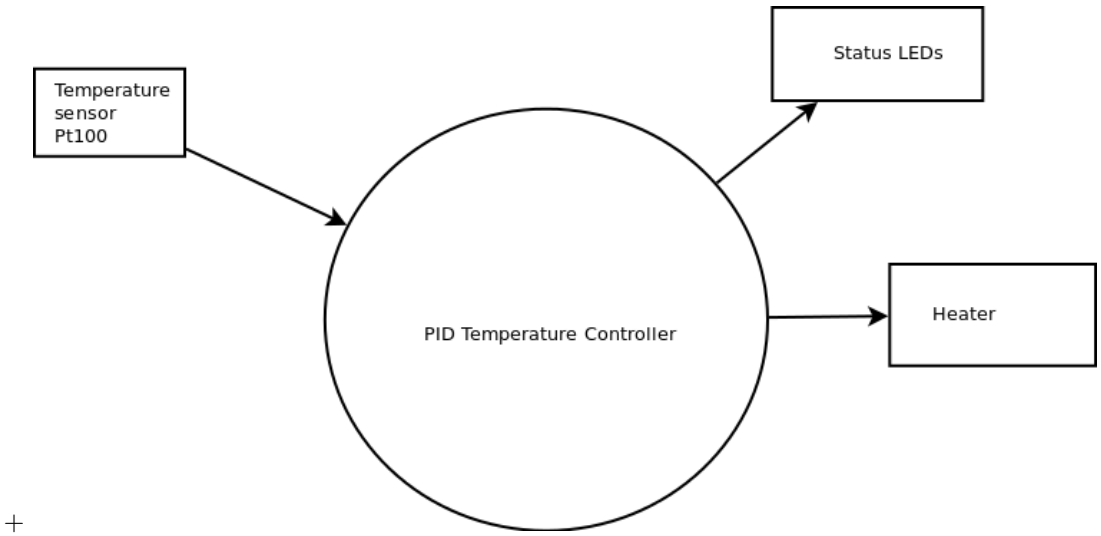


Figure 6.6: Yourdon Context Diagram for a standalone micro-processor implemented PID Temperature Controller.

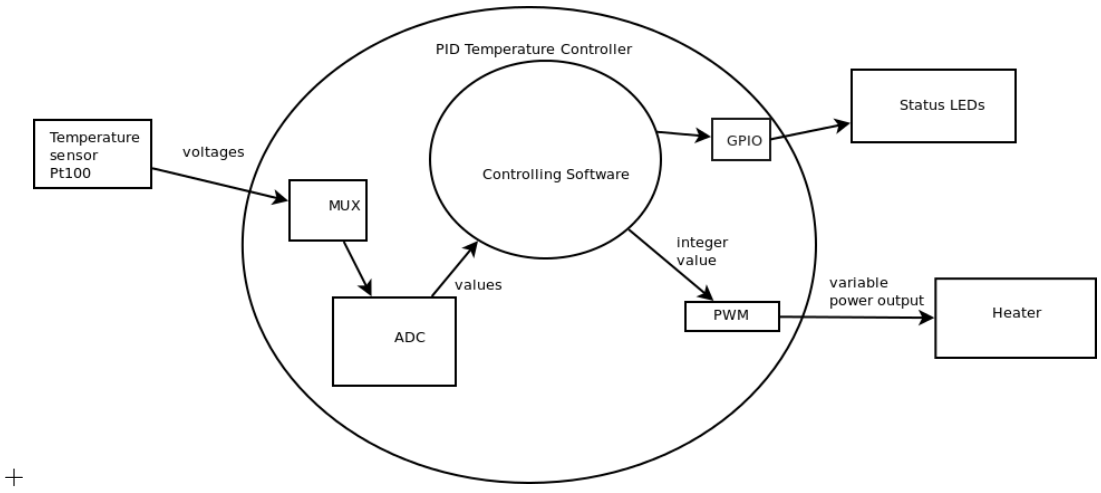


Figure 6.7: Yourdon data flow diagram for PID Temperature Controller identifying initial processing nodes.

The Yourdon methodology provides model refinement, by zooming into data transform bubbles, analysing them in more depth and creating more paths and transform bubbles which further define the data flow and processing. The Yourdon diagram is refined, by adding detail to both the afferent data flow coming through the MUX and ADC on the micro-controller and the efferent channelled through a PWM module. This next stage of model refinement is shown in figure 6.7. The controlling software is then further refined, by looking at or zooming into transform bubbles and adding more detail i.e. following the data streams through the process, additional transform bubbles are created as required. The lines connecting the 'transform bubbles' define the data passed between them. When the data flow analysis is finished, each transform bubble represents a software function. Because the connecting lines define the data passed between transform bubbles, the inputs and outputs of the associated software functions are also defined. The Yourdon methodology thus allows the refinement and modelling of a process from a data flow perspective defining software functions in its final stage (see figure 6.8). In all 'bare metal'[8] software architectures, a rudimentary operating system is required, often referred to as the 'monitor'. PID, because the algorithm depends heavily on integral calculus [35][Ch.3.3] is time sensitive and it is necessary to execute it at precise intervals determined by its proportional, integral and differential (PID) coefficients. Most micro-controllers feature several general purpose timers [63]. An internal timer can be used in conjunction with the monitor function to call the PID algorithm at a regular and precise time interval.

Data flow model to programmatic call tree. The Yourdon methodology also gives guidance as to which software functions should be called to control the process, or in 'C' terms be the main function. Using figure 6.8

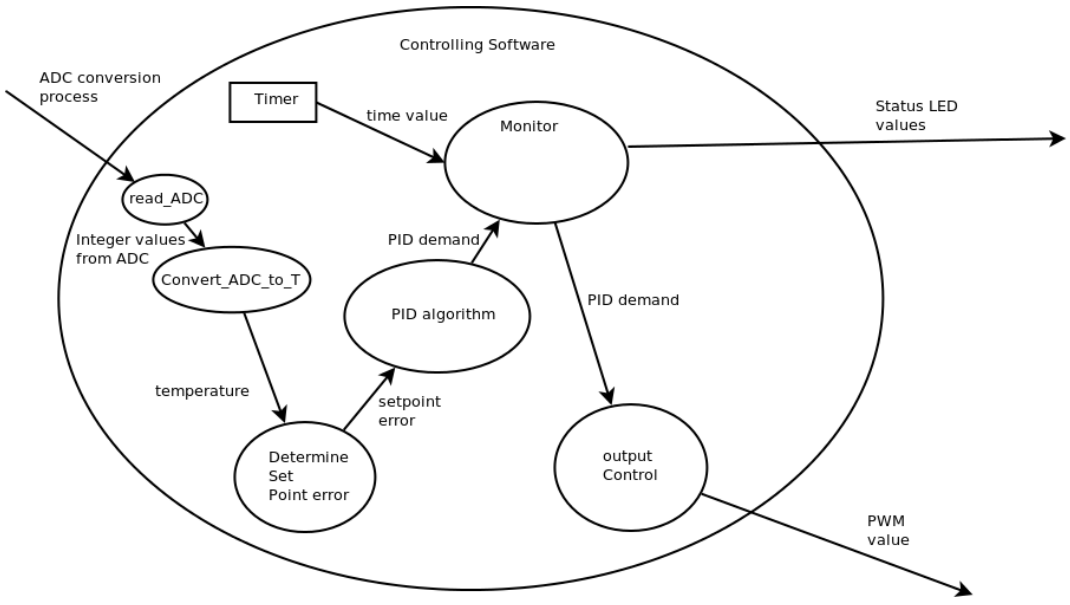


Figure 6.8: Final Yourdon data flow diagram which has defined the software functions for the PID temperature controller

the transform bubble to represent the 'main' or controlling function in the software must be chosen. This can be

[8]'Bare metal' is a term used to indicate a micro-processor controlled system that does not use a traditional operating system. These are generally coded in 'C' or assembly language and run immediately from power-up.

thought of as picking one bubble and holding it up. The other bubbles hang underneath forming the software call tree hierarchy, see figure 6.9. From examining the diagram, and in common with established embedded programming practise, this is clearly going to be the monitor function.

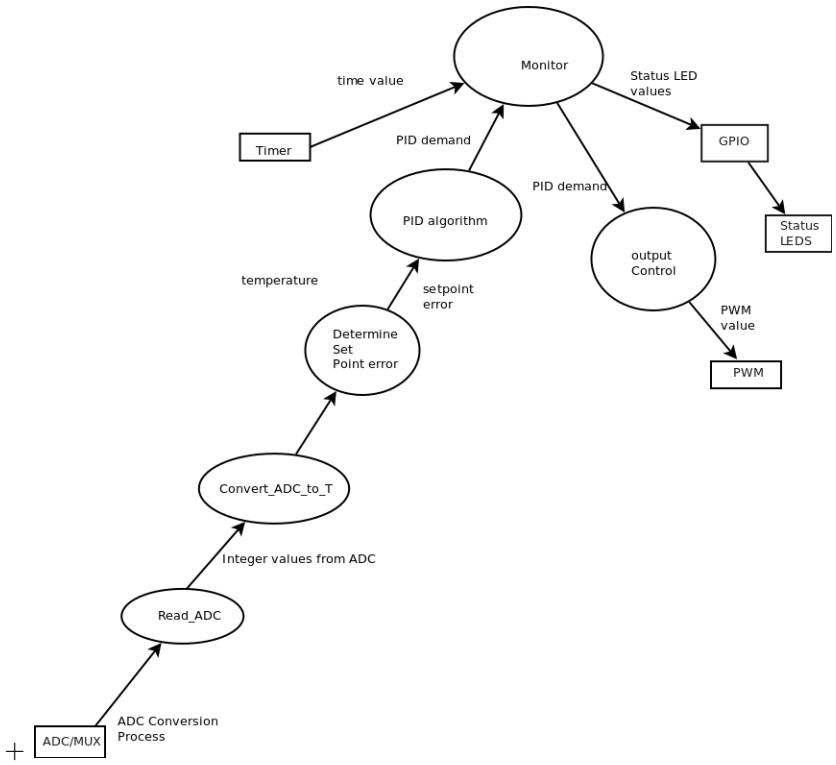


Figure 6.9: Software: Yourdon data flow diagram converted to programatic call tree.

Software Algorithm. The monitor function will orchestrate the control process. Firstly it will examine the timer value, and when appropriate, call the **PID()** function. The **PID()** function calls **determine_set_point_error()** which calls **convert_ADC_to_T()** which in turn calls **Read_ADC()** (the function developed in the earlier example) which reads from hardware. With the set point error value the **PID()** function will return an output control value to its calling function (i.e. the PID demand which will be returned to the monitor function). The PID demand value will be applied via the pulse width modulation (PWM) module. A rudimentary closed loop control system incorporating both hardware and software has been defined. By using the Yourdon methodology a programmatic design frame-work i.e. a call tree structure was obtained. All the components, i.e. hardware elements and software functions that will be used in the temperature controller are now defined. These are listed, and from the bottom-up, FMMD analysis is begun.

6.3.2 FMMD Analysis of PID temperature Controller

To summarise from the design stage, the electronic components identified thus far:

- ADCMUX — Electronics, analysed in previous example,
- TIMER — Internal micro controller timer,
- HEATER — Heating element, essentially a resistor,
- Pt100 — Pt100 Temperature sensor, as analysed in section 5.6,
- PWM — Internal micro controller pulse width modulation module,
- General Purpose I/O (GPIO) — I/O used to drive LEDS,
- LEDs — Indication LEDs via GPIO,
- micro-controller — the medium for running the software.

6.3.3 Temperature Controller Hardware Elements FMMD.

ADCMUX and Read_ADC. We re-use the derived component from section 6.2.1.

$$fm(RADC) = \{VV\_ERR, HIGH, LOW\}.$$

TIMER. The internal timer, from a programmer's perspective is a register, which when read returns an incremented time value. Essentially its a free running integer counter with an interfacing register. Using two's complement mathematics, by subtracting the time last read value, we can calculate the interval between readings (assuming the timer has not wrapped around more than once). A timer can fail by incrementing its value at an incorrect rate, or can stop incrementing. The failure modes of $TIMER$ are defined thus:

$$fm(TIMER) = \{STOPPED, INCORRECT\_INTERVAL\}.$$

HEATER. A heating element is typically some configuration of resistive wire. It therefore has the same failure modes as a resistor:

$$fm(HEATER) = \{OPEN, SHORT\}.$$

Pt100 Platinum Temperature Sensor. The Pt100 four wire configuration was analysed in section 5.6, the derived component is re-used here:

$$fm(Pt100) = \{OUT\_OF\_RANGE\}.$$

PWM. From a programmatic perspective a PWM output is a register to which software writes an unsigned magnitude value [63][Ch.15]. The PWM hardware module applies this using a mark space ratio proportional to that value, providing a means of varying the amount of power supplied. When the PWM action is halted, or fails, the digital output pin associated with it will typically be held in a high or low state. The PWM has the following failure modes:

$$fm(PWM) = \{HIGH, LOW\}.$$

Micro-Controller. The Micro controller is a complex piece of highly integrated electronics. At a minimum it would include a micro-processor with PROM and RAM general I/O and external interrupt lines. Typically there are many other I/O modules incorporated (e.g. TIMERS, UARTS, PWM, ADC, ADCMUX, CAN). In this project the ADCMUX, TIMER, PWM and general purpose computing facilities are used. Consider the general computing, CLOCK, PROM and RAM failure modes:

$$fm(micro{-}controller) = \{PROM\_FAULT, RAM\_FAULT, CPU\_FAULT, ALU\_FAULT, CLOCK\_STOPPED\}.$$

6.3.4 Temperature Controller Software Elements FMMD

Identified Software Components:

- — **Monitor()** (which calls **PID()**,**output_control()** and **setLEDS()**),
- — **PID()** (which calls **determine_set_point_error()**),
- — **determine_set_point_error()** (which calls **convert_ADC_to_T()**),
- — **convert_ADC_to_T()** (which calls **read_ADC()**),
- — **read_ADC()** (analysed in the previous section 6.2.1),
- — **output_control()** (which sets the PWM hardware according to the PID demand value).

With the call tree structure defined (see figure 6.9), a hierarchy compatible with FMMD for analysis has been obtained. However, it is only the top, i.e. the software, part of the hierarchy. FMMD is a bottom-up process, thus it starts with the lowest level, i.e. the electronics. The Yourdon context diagram (see figure 6.6) is useful here as its data sources and sinks are by definition the lowest levels in the system. The input, or origin of the afferent data flow can be followed to find system inputs, and the output, or efferent flow to find the bottom level for outputs/actuators etc. Starting with the afferent flow, the reading of the temperature and its conversion to a PID calculated heater output demand is examined.

6.3.4.1 Afferent flow FMMD analysis, Pt100, temperature, set point error, PID output demand.

Staring with the afferent data flow for the temperature readings, the lowest level in the hierarchy is found, the Pt100 sensor. Beginning at the bottom, a functional grouping is formed with the function **read_ADC()** and the Pt100. This gives a derived component, 'Read_Pt100' (see appendix A.3). The derived component Read_Pt100 has the following failure modes:

$$fm(Read\_Pt100) = \{VOLTAGE\_HIGH, VAL\_ERR, VOLTAGE\_LOW\}.$$

Moving along the afferent flow, the **convert_ADC_to_T()** function is next up the hierarchy. This will call **Read_ADC()** twice, once for the high Pt100 value, again for the lower. The resistance of the Pt100 element is then calculated, and with this—using a polynomial or a lookup table [29]—the temperature determined. The pre-conditions for the function are that:

- The lower Pt100 value is within an acceptable voltage range i.e. Pt100_lower_voltage,

- The higher Pt100 value is within an acceptable voltage range i.e. Pt100_higher_voltage,
- The lower and higher values agree to within a given tolerance i.e. Pt100_high_low_mismatch.

Any violation of these pre-conditions is equivalent to a failure mode[9]. The post-condition is that it returns a temperature within a given tolerance to the temperature at the sensor. A failure of this post-condition can be termed 'temp_incorrect'.

[9] An actual measured temperature outside the pre-defined range would be detected as an unacceptable voltage range failure.

Applying FMMD to the functional grouping formed by **Read_Pt100()** and the function **convert_ADC_to_T()**. gives the derived component Get_Temperature. This analysis is presented in table A.15. Failure symptoms are collected and the derived component created with the following failure modes:

$$fm(Get\_Temperature) = \{Pt100\_out\_of\_range, temp\_incorrect\}.$$

Following the afferent flow further, the function to determine the control error value is examined. This is simply the target temperature subtracted from that measured by the sensor. A functional grouping is formed with the newly derived component Get_Temperature and the function **determine_set_point_error()**. The pre-condition for **determine_set_point_error()** is that the temperature read by it is accurate, and its post-condition is to return the correct control error value. The post-condition can fail, or the temperature read could be incorrect. This could be detectable (i.e. we detect a failure from the Pt100 $Pt100\_out\_of\_range$) or undetectable (i.e. the post condition for this function simply fails or the failure mode *temp_incorrect* occurs). This analysis is presented in table A.16. Failure mode symptoms are collected and a new derived component GetError created where:

$$fm(GetError) = \{KnownIncorrectErrorValue, IncorrectErrorValue\}.$$

Following the afferent path the PID algorithm is next in the software call tree. The *GetError* derived component and the **PID()** function form a functional grouping. The pre-condition for the **PID()** function is that it receives the correct error value. The post-condition is that it outputs correct control values. All digital signal processing algorithms are sensitive to calling frequency, and thus should be time invariant [99][p.58]. Were this function to be called at an incorrect rate, its output could be erroneous (the differential and integral parameters would effectively have been changed). However this problem is a failure mode for the consideration of the function calling it i.e. the context of use. That is, the **PID()** function is called, but its calling function is responsible for the timing, or in more general terms, it is the calling function that sets the context for the **PID()** function (i.e. what it is used for). The derived component PID is created, see table A.17, with the following failure modes:

$$fm(PID) = \{KnownControlValueErrorV, IncorrectControlErrorV\}.$$

The software call tree for the afferent flow has now been modelled using FMMD; this is represented as an

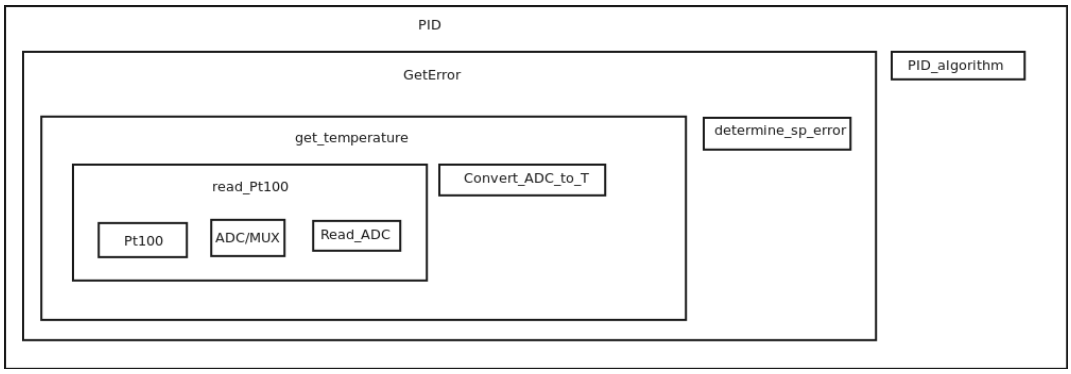


Figure 6.10: Euler diagram representing the hierarchy of FMMD analysis applied to the afferent branch of call tree for the PID temperature controller example.

Euler diagram in figure 6.10. Two call tree branches remain. The LED indication branch and the PWM/heater output.

6.3.4.2 Efferent flow, PID demand value to PWM output

The monitor function calls the **output_control()** function with the PID demand. The **output_control()** function then sets the PWM hardware register, which causes the mark space output of the PWM module to apply the demanded power. A functional grouping with the Heating element, a PWM module and the **output_control()** function is formed to model this branch of the efferent flow. This functional grouping is a hardware/software hybrid. FMMD analysis is applied to this functional grouping in table A.18. For the **output_control()** function, there is a pre-condition that the PWM module is configured and working, and has the correct clock frequency. A second pre-condition is that the heating element is connected and working. The post-condition is that it sets the correct value into the PWM register to implement the power output demand. A derived component is created called HeaterOutput, see table A.18, with the following failure modes:

$$fm(HeaterOutput) = \{HeaterOnFull, HeaterOff, HeaterOutputIncorrect\}.$$

As an aside: the *HeaterOnFull* failure should raise alarm bells for designers and upon its discovery, measures may be recommended to inhibit this (such as perhaps adding a safety relay to cut the power to the heater).

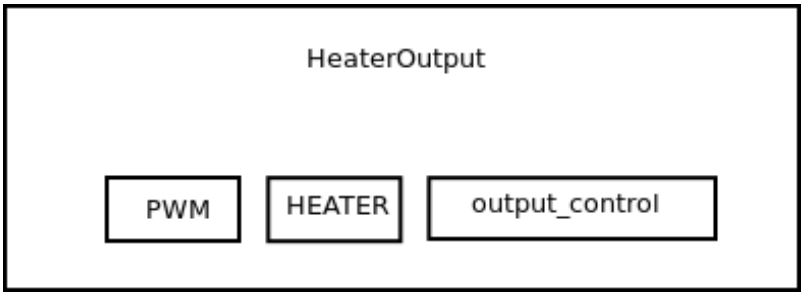


Figure 6.11: Euler diagram showing HeaterOutput with its two hardware components, PWM and HEATER, and its software component **output_control()**.

6.3.4.3 Efferent flow: LED status LEDs

The status LEDS will be controlled by general purpose (GPIO) I/O pins. Three LEDS could be used, one flashing with a human readable mark space ratio representing the heater output, one flashing at a regular interval to indicate the processor is alive and another flashing at an interval related to the temperature, (to indicate if the temperature readings are within expected ranges). Each LED should flash in normal operation, and any LED being permanently on or off would indicate to the operator that an error had occurred. The pre-condition for this function is that the GPIO is connected to working LEDS. The post-condition is that the function **setLEDS()** will supply correct indication by flashing the LEDs. A functional grouping is formed from the GPIO, the LEDs and the software function **setLEDs()**. FMMD analysis is applied to this functional grouping in table A.19. The derived component for the setLED function, GPIO and LEDs has the following failure modes:

$$fm(LEDoutput) = \{FailureIndicated, IndicationError\}$$

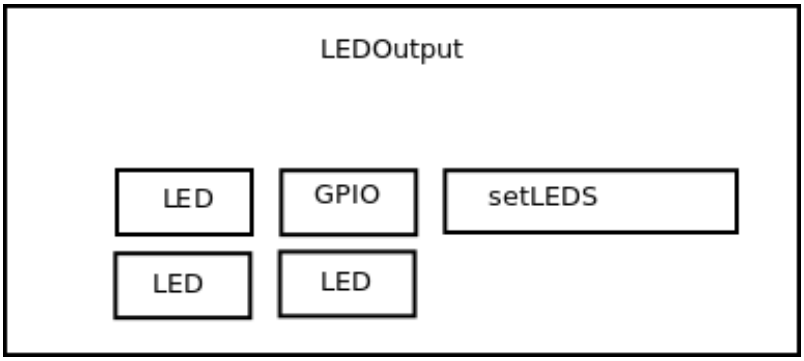


Figure 6.12: Euler diagram showing LEDOutput with its three LEDs and GPIO hardware elements, and its software component setLEDS.

6.3.4.4 Final Analysis Stage: PID Temperature Controller

The possibility of each software function failing its post-condition without a direct underlying cause from one of its components has been included in each analysis stage involving software. This is because software introduces the possibility of anything going wrong! The common causes for software failing are:

- Value/RAM corruption typically from interrupt contention problems [88] or accidental over writing [4], but can be from external sources such as radiation changing bits/values at runtime [37, 75];
- Address bus errors leading to program errors (program sequence);
- ROM memory failures;
- Unintended behaviour of software.
- Electro Magnetic Compatibility (EMC) interference.

Because the software is running on a medium, that of the processor or micro-controller, the FMMD analysis at the final or highest level (see table A.20), must include all possible failure modes of this medium i.e.

$$fm(micro{-}controller) = \{PROM\_FAULT, RAM\_FAULT, CPU\_FAULT, ALU\_FAULT, CLOCK\_STOPPED\}.$$

The final FMMD stage forms a functional grouping with the derived components determined previously:

- the micro-controller,
- PID,
- HeaterOutput,
- LEDoutput,
- the function **monitor()**.

The post-condition for the monitor function is that it implements the PID control task correctly. A derived component for the standalone temperature controller is now created, and given the name TempController. It will

have the following failure modes:

$$fm(TempController) = \{ControlFailureIndicated,$$
$$ControlFailure,$$
$$KnownIndicationError,$$
$$UnknownIndicationError\}.$$

The failure mode analysis of the complete PID controller is represented as an Euler diagram in figure 6.13.

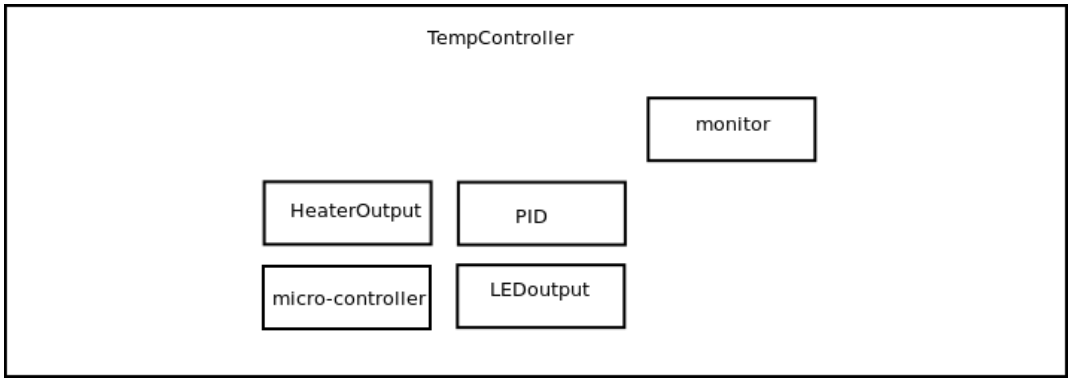


Figure 6.13: Euler diagram of the temperature controller final analysis stage, showing the hybrid software/hardware derived components and the function at the head of the call tree **monitor()**.

6.3.5 Conclusion: Standalone system, PID Temperature Controller

The PID temperature control example above, shows that complete hybrid software/electronic systems can be modelled using FMMD. This analysis has revealed system level failure modes that are un-handled and some that are undetectable. The FMMD model can be traversed from undesirable top level failures to the base component failure modes that are the causes. This means that by using FMMD, the sub-systems which require re-design to eliminate or reduce the likelihood of undetectable failure modes can be identified. The demands of EN61508 [95] for minimum safe failure fraction thresholds [89][p.52] associated with SIL levels, make this a desirable feature of any FMEA based methodology. For the failure modes caused by electronics, reliability statistics can be applied, and the possibilities of using higher rated components instead of potentially expensive re-design can be simulated/modelled. For software errors, it may be necessary to provide extra functions to provide self checking. EN61508 high reliability software measures such as duplication of functions with checking functions arbitrating them (diverse programming [95][C.3.5]) could be applied. For instance, measures may included to validate the processor clocking with an external watchdog and a simple communications protocol. For PROM and RAM faults measures such as run-time checksums and ram complement checking can be applied.

Chapter 7

FMMD Metrics Critiques Exceptions and Evaluation

Metrics

This chapter defines a metric for the complexity of an FMEA analysis task. This concept is called 'comparison complexity' and is a means to assess the performance of FMMD against current FMEA methodologies. This concept was introduced as reasoning distance in section 2.6.1. This metric is developed using set theory and then formulae are presented for calculating the complexity of applying FMEA to a group of components. These formulae are then used for a hypothetical example, which is analysed by both FMEA and FMMD. The hypothetical example leads to a general formula, which shows that the reasoning distance goes from a polynomial to a logarithmic order comparing XFMEA with FMMD. The reasoning distances obtained from the FMMD examples (see chapter 5) are compared against XFMEA. Following on from formal definitions, 'unitary state failure modes' are defined, i.e. ensuring that component failure modes are mutually exclusive. Standard formulae for combinations are then used to develop the concept of the cardinality constrained power-set. Using this in combination with unitary state failure modes an expression for calculating the number of failure scenarios to check for in double failure analysis is presented. This is followed by some critiques of FMMD.

7.1 Defining the concept of 'comparison complexity' in FMEA

When discussing safety critical systems they are usually thought of in terms of the physical plant—or in terms of their safety functionality. When performing FMEA the system under investigation is considered to be a collection of components which have associated failure modes. The object of FMEA is to determine cause and effect. FMEA can be viewed as a process, taking each component in the system and for each of its failure modes applying analysis with respect to the whole system. This however entails a problem: which other components in the system must be checked against each particular failure mode? Often a component failing will have obvious effects on functionally adjacent components. Sometimes side effects of failure may manifest due to interaction with other components not obviously functionally related. The temptation with FMEA can be to

follow direct lines of failure effect reasoning without considering side effects. To perform FMEA exhaustively, it could be stipulated that every failure mode must be checked for effects against all the components in the system. This would mean examining for all possible side effects that a base component failure could cause. This is termed 'exhaustive FMEA' (XFMEA). The number of checks to make to achieve this, gives an indication of the complexity of the analysis task. Comparison complexity (or reasoning distance) is defined as the count of paths (and thus reasoning checks applied) between failure modes and components necessary to achieve XFMEA for a given group of components G.

7.1.1 Formal definitions of entities used in FMEA

Using the language developed in the previous chapters, a system for analysis is considered as a collection of components. This set of components is termed G, and the number of components in it by $|G|$. G is simply a sub-set of all possible components. The set of all components is $\mathcal{C}$; it can be can stated that $G \subset \mathcal{C}$. Individual components are denoted as c with additional indexing where appropriate.

The function fm returns the failure modes of a component, its signature is $fm : \mathcal{C} \rightarrow \mathcal{F}$, where $\mathcal{F}$ is the set of all failures. The number of potential failure modes of a component, c, is $|fm(c)|$.

Indexing the components in the system under investigation $c_1, c_2 \ldots c_{|G|}$ allows expression of the number of checks required to exhaustively examine every failure mode against all the other components in a system (see equation 7.2). Comparison Complexity can be represented by a function CC, with its domain as G, and its range as the number of checks—or reasoning stages—to perform to satisfy an XFMEA inspection.

Let $\mathcal{G}$ represent the set of all functional groupings then CC is defined by,

$$CC : \mathcal{G} \rightarrow \mathbb{Z}. \tag{7.1}$$

Comparison complexity, CC, for a group of n components G, is given by

$$CC(G) = (n-1) \sum_{1 \leq i \leq n} |fm(c_i)|. \tag{7.2}$$

Equation 7.2 says that for every failure mode in the group G, it must be checked against all other components in the group (except itself). This gives a count of the number of reasoning paths to perform XFMEA. These reasoning distance concepts are discussed in section 3.2. Equation 7.2 can be simplified if the total number of failure modes in the system K can be determined, (i.e. $K = \sum_{n=1}^{|G|} |fm(c_n)|$); the equation becomes

$$CC(G) = K.(|G| - 1). \tag{7.3}$$

7.1.2 A general formula for counting Comparison Complexity in an FMMD hierarchy

An FMMD hierarchy consists of many functional groupings which are subsets of G. FMMD analysis creates a hierarchy $\hbar$ of functional groupings. Individual functional groupings can be defined using an index i for identification and a superscript for the α level i.e. FG_i^{α} (see section 4.5.2). For example the first functional grouping in a hierarchy containing base components only i.e. at the zeroth level of an FMMD hierarchy where $\alpha = 0$,

would have the superscript 0 and a subscript of 1: FG_1^0. The functional grouping representing the potential divider in section 4.2 has an α level of 0 (as it contains only base components). The functional grouping with the potential divider and the operational amplifier has an α level of 1.

An FMMD hierarchy will have reducing numbers of functional groupings as the hierarchy is traversed upwards. In order to calculate its comparison complexity, equation 7.2 must be applied to all functional groupings on each level. An FMMD hierarchy is defined as a set of functional groupings, $\hbar$. A helper function, g, is used that applies CC to all functional groupings at a particular level, ξ, in an FMMD hierarchy, $\hbar$, and returns the sum of the comparison complexities,

$$g : \hbar \times \mathbb{N} \rightarrow \mathbb{N}. \tag{7.4}$$

Let L represent the number of levels in the FMMD hierarchy $\hbar$ and $g(\hbar, \xi)$ represent the comparison complexity of functional groupings on the level ξ. The comparison complexity function CC is overloaded, to obtain the comparison complexity of an entire hierarchy thus:

$$CC(\hbar) = \sum_{\xi=0}^{L} g(\hbar, \xi). \tag{7.5}$$

7.1.3 Complexity Comparison Examples

The $NONINVAMP$ example from chapter 4, which has two analysis stages, the potential divider and then the amplifier, is chosen as an example for comparison complexity. The complexities are added from both these stages to determine how many reasoning paths there were to perform FMMD analysis on the non-inverting amplifier.

The potential divider discussed in section 4.2 has four failure modes and two components and therefore has CC of 4. This using equation 7.2 is calculated thus,

$$CC(potdiv) = \sum_{n=1}^{2} \big(|2| \times (|1|)\big) = 4.$$

The potential divider derived component is formed into a functional grouping with an op-amp which has four failure modes i.e. a functional grouping with two components, one with four failure modes and the other (the potential divider) with two,

$$CC(invamp) = 2 \times 1 + 4 \times 1 = 6.$$

The two calculated complexities are added to determine the number of reasoning paths to analyse the amplifier using FMMD. The potential divider has a comparison complexity of four and the amplifier section a comparison complexity of six. To analyse the inverting amplifier with FMMD a comparison complexity of 10 was required. Using traditional FMEA employing exhaustive checking (XFMEA) $2\times(3-1)+2\times(3-1)+4\times(3-1) = 16$ was obtained. Even with this very trivial example, benefits of taking a modular approach to FMEA are seen.

Complexity Comparison for a hypothetical 81 component system. A system, *example*, with just 81 components, with these components having 3 failure modes each would, using equation 7.3 have a CC of

$$CC(example) = \sum_{n=1}^{81} |3|.(|80|) = 19440.$$

The computational order for XFMEA would be polynomial ($O((N)(N-1)f) \approx O(N^2.f)$) (where f is the variable number of failure modes) as discussed in section 2.2. This order may be acceptable in a computational environment. However, the choosing of functional groupings and the analysis process are by-hand/human activities. It can be seen that it is practically impossible to achieve XFMEA for anything but trivial systems.

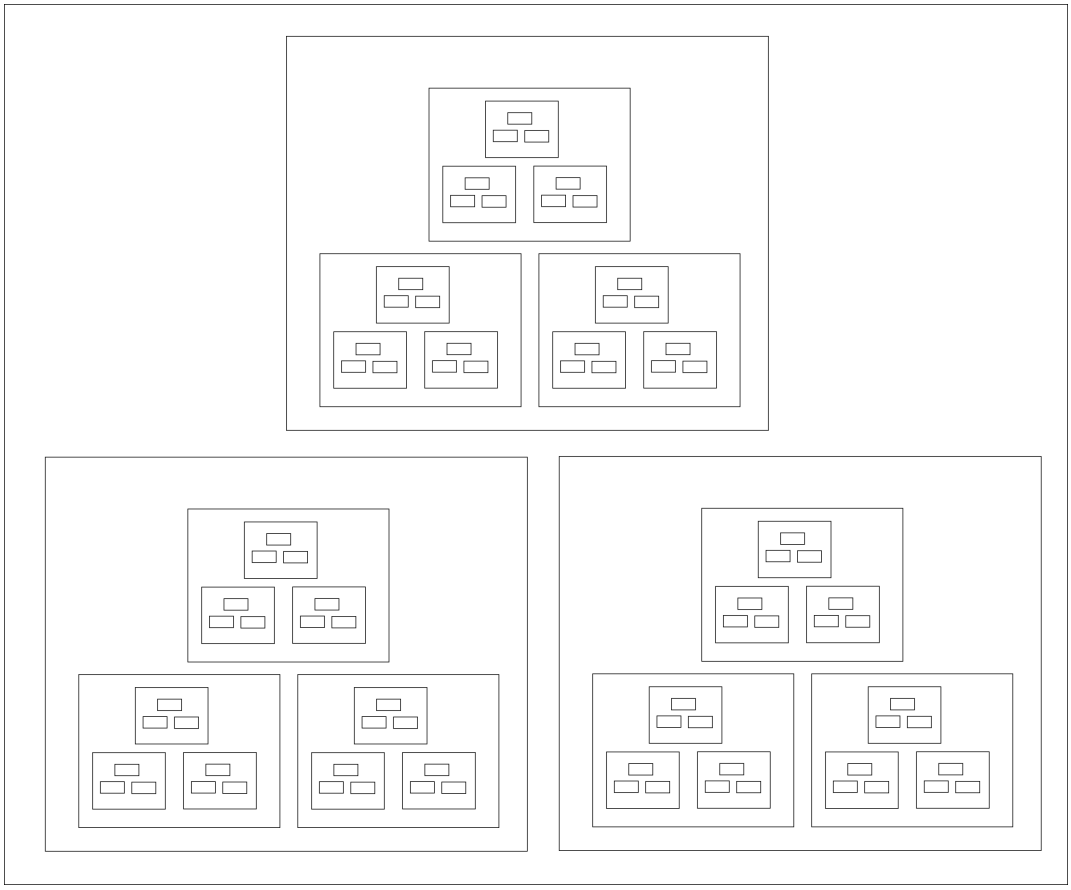


Figure 7.1: Euler diagram of a hypothetical FMMD Hierarchy with 81 base components with the number of components in each FG fixed to three ($|FG| = 3$)

7.1.4 Comparing FMMD and XFMEA Comparison Complexity

Because components have variable numbers of failure modes, and functional groupings have variable numbers of components, it is difficult to use the general formula for comparing the number of checks to make for XFMEA and FMMD. If an example is created by fixing the number of components in a functional grouping and the number of failure modes per component, formulae can be determined to compare the number of checks to make from an FMMD hierarchy to XFMEA. While real-world analysis models have variable numbers of failure modes per component type and different numbers of components in their functional groupings, a fixed model provides indicative estimates of complexity performance.

Consider k to be the number of components in a functional grouping (i.e. $k = |FG|$), f is the number of failure modes per component (i.e. $f = |fm(c)|$), and L to be the number of levels in the hierarchy of an FMMD analysis. The number of failure scenarios to check in a (fixed parameter for $|FG|$ and $|fm(c_i)|$) FMMD hierarchy is represented with equation 7.6.

$$\sum_{n=0}^{L} k^n.k.f.(k-1) \tag{7.6}$$

The thinking behind equation 7.6, is that for each level of analysis – counting down from the top – there

are k^n functional groupings within each level; XFMEA is applied to each functional grouping on the level. The number of checks to make for XFMEA, is the number of components k multiplied by the number of failure modes f checked against the remaining components in the functional grouping $(k-1)$. If, for the sake of example, the number of components in a functional grouping is fixed to three and the number of failure modes per component to three, an FMMD hierarchy would look like figure 7.1.

7.1.5 Comparing XFMEA and FMMD: an Example

Using the diagram in figure 7.1, there are three levels of analysis. Starting at the top, there is a functional grouping with three derived components, each of which has three failure modes. Thus the number of checks to make, or comparison complexity, in the top level is $3^0 \times 3 \times 2 \times 3 = 18$. On the level below that, there are three functional groupings each with an identical number of checks, $3^1 \times 3 \times 2 \times 3 = 56$. On the level below that there are nine functional groupings, $3^2 \times 3 \times 2 \times 3 = 168$. Adding these together gives 242 checks to make to perform FMMD (i.e. XFMEA *within the* functional groupings).

To take the system represented in figure 7.1, and apply XFMEA on it as a whole system, using equation 7.2, $CC(G) = \sum_{n=1}^{|G|} |fm(c_n)|.(|G|-1)$, where $|G|$ is 27, $fm(c_n)$ is 3 and $(|G|-1)$ is 26, this gives: $CC(G) = \sum_{n=1}^{27} |3|.(|27|-1) = 2106$.

In order to get general equations with which to compare XFMEA with FMMD, equation 7.2 can be re-written in terms of the number of levels in an FMMD hierarchy. The number of components in the system, is the number of components in a functional grouping raised to the power of the level plus one. The equation 7.2 is re-written as:

$$\sum_{n=1}^{k^{L+1}} (k^{L+1}-1).f \ , \tag{7.7}$$

or

$$k^{L+1}.(k^{L+1}-1).f \ . \tag{7.8}$$

Equation 7.6 (FMMD) and 7.2 can be used to compare (for fixed sizes of $|G|$ and $|fm(c)|$) the two approaches, for the work required to perform exhaustive checking.

For instance, having four levels of FMMD analysis, with these fixed numbers, will require 81 base level components. Applying equation 7.8, gives

$$3^4.(3^4-1).3 = 81.(81-1).3 = 19440. \tag{7.9}$$

Equation 7.8 shows that applying XFMEA where components all have three failure modes and there are 81 components, would involve 19,440 reasoning paths. Applying equation 7.7,

$$\sum_{n=0}^{3} 3^n.3.3.(2) = 720.$$

For FMMD (where within functional groupings the analysis **is exhaustive**) it only requires 720 reasoning paths.

7.1.5.1 Plotting XFMEA and FMMD reasoning distance

Using the gnuplot utility [92, 46] and implementing equation 7.8 for XFMEA and equation 7.6 for FMMD reasoning distances and using a logarithmic axis, the reasoning distance comparison is shown as a graph. The gnuplot script used to produce the comparison graph is listed in section A.3.9.

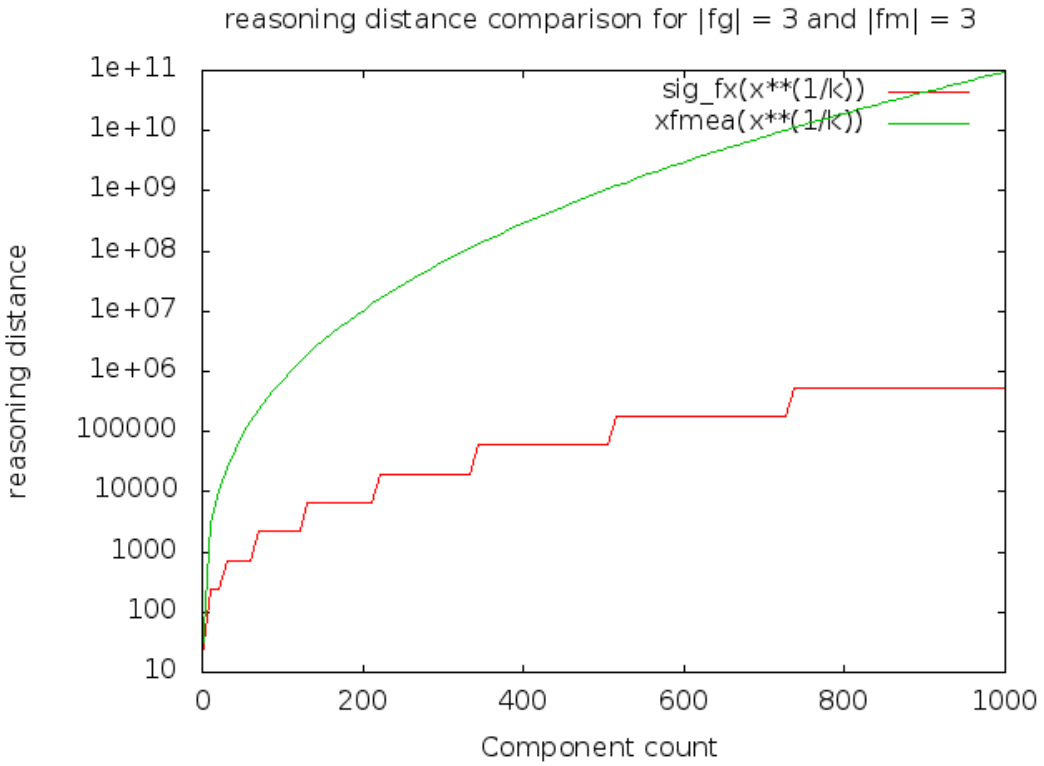


Figure 7.2: XFMEA and FMMD reasoning distance comparison graph.

Looking at the graph in figure 7.2 it is seen that acceptable reasoning distances for large numbers of components becomes extremely difficult to achieve for traditional FMEA. This shows that FMMD, by analysing a system in a modular and hierarchical way, has reduced the amount of analysis work significantly. It can be seen that the reasoning distance has gone from a polynomial to a logarithmic order. In mathematical terms this means the polynomial order has been converted to logarithmic by being able to take exponentiation values out to become instead constants of integration. This process can be viewed as similar to the order of processing that occurs in the decimation in time FFT [23] when compared to the DFT algorithm.

7.2 Complexity Comparison applied to FMMD electronic circuits analysed in chapter 5.

All the FMMD examples in chapter 5 showed a marked reduction in comparison complexity compared to XFMEA. To calculate XFMEA the comparison complexity equation 7.2 is used. Complexity comparison for FMMD vs. XFMEA for the first three examples in chapter 5 are presented in the following table 7.1. The complexity comparison figures for the example circuits in chapter 5 show that for the non-trivial examples, as

Hierarchy Level	**Derived Component**	**Complexity Comparison**	$\|fm(c)\|$: **number of derived failure modes**
Inverting Amplifier Two stage FMMD Hierarchy: section 5.1			
0	PD	4	2
1	INVAMP	8	3
2	Total for INVAMP:	10 (FMMD)	
0	Total for INVAMP:	16 (XFMEA)	
Inverting Amplifier One stage FMMD Hierarchy: section 5.1			
0	INVAMP	16	3
1	Total for INVAMP:	16 (FMMD)	
0	Total for INVAMP:	16 (XFMEA)	
Differencing Amplifier Three stage FMMD Hierarchy: section 5.2			
2	NonInvAMP reused [1]	10	3
0	SEC_AMP	16	4
3	DiffAMP	7	4
3	Total for DiffAMP	33 (FMMD)	
0	Total for DiffAMP:	80 (XFMEA)	
Five Pole Sallen Key Low Pass Filter: Three stage FMMD Hierarchy: section 5.3			
0	FirstOrderLP	4	2
1	LP1	10	4
2	SKLP	48	4
3	FivePoleLP	20	4
3	Total for FivePoleLP	82 (FMMD)	
0	Total for FivePoleLP	384 (XFMEA)	

Table 7.1: Comparison Complexity figures for the first three examples in Chapter 5.

more levels in the FMMD hierarchy are used, the performance gain over XFMEA is demonstrated.

Hierarchy Level	**Derived Component**	**Complexity Comparison**	$\|fm(c)\|$: **number of derived failure modes**
Bubba Oscillator one stage (naïve) FMMD Hierarchy: section 5.4.5			
1	PHS45	4	2
1	INVAMP	16	3
0	NIBUFF	0	4
2	BUBBA	308	2
2	Total for BUBBA:	328 (FMMD)	
0	Total for BUBBA:	468 (XFMEA)	
Inverting Amplifier Multiple stage FMMD Hierarchy: section 5.4.6			
1	PHS45	4	2
1	INVAMP	16	3
0	NIBUFF	0	4
2	BUFF45	6	2
3	PHS135BUFFERED	4	2
2	PHS225AMP	5	2
4	BUBBA	2	2
1	Total for BUBBA:	37 (FMMD)	
0	Total for BUBBA:	468 (XFMEA)	

Table 7.2: Complexity Comparison figures for the Bubba Oscillator FMMD example (see section 5.4).

7.2.1 Comparison Complexity for the Bubba Oscillator Example

The Bubba oscillator example (see section 5.4) was chosen because it had a circular signal path. It was also analysed twice, once by naïvely using the first functional groupings identified, and secondly by de-composing the circuit further. These two analyses are used to compare the effect on comparison complexity (see table 7.2) with that of XFMEA. The initial naïve FMMD analysis reduces the number of checks by around a third, the more de-composed analysis by more than a factor of ten.

7.2.2 Sigma Delta Example: Comparison Complexity Results

The complexity figures for this mixed analogue to digital circuit are not adversely affected by the digital to analogue level interfacing circuitry. This is where the modular approach aids understanding and analysis. When following this circuit through in a traditional way, following signal paths that are level shifted, adds to the complication of analysing it for failures. That is the signal path crosses from analogue to digital signalling and vice versa.

Hierarchy Level	**Derived Component**	**Complexity Comparison**	$\|fm(c)\|$: **number of derived failure modes**
$\Sigma\Delta ADC$ FMMD Hierarchy: section 5.5			
1	SUMJINT	30	4
0	HISB	0	4
2	BISJ	8	2
1	DIGBUF	2	4
1	PD	4	2
2	DL2AL	6	3
3	FFB	5	2
2	$\Sigma\Delta ADC$	4	2
2	Total for $\Sigma\Delta ADC$:	55 (FMMD)	
0	Total for $\Sigma\Delta ADC$:	225 (XFMEA)	

Table 7.3: Complexity Comparison figures for the $\Sigma\Delta ADC$ FMMD example (see section 5.5).

7.3 Unitary State Component Failure Mode Sets

Design Decision/Constraint. An important factor in defining a set of failure modes is that they should represent the failure modes as simply and minimally as possible. It should not be possible, for instance, for a component to have two or more failure modes active at once. Were this to be the case, additional combinations of failure modes would have to be considered within the component. Having a set of failure modes where N modes could be active simultaneously would mean having to consider an additional $2^N - 1$ failure mode scenarios. Should a component be analysed and simultaneous failure mode cases exist, the combinations could be represented by new failure modes, or the component should be considered from a fresh perspective, perhaps considering it as several smaller components within one package. This property, failure modes being mutually exclusive, is termed 'unitary state failure modes' in this study. This corresponds to the 'mutually exclusive' definition in probability theory [93].

What is required is to define a property for a set of failure modes F where only one failure mode can be active at a time; or borrowing from the terms of statistics, the failure mode being an event that is mutually exclusive within the set F. A set of failure mode sets called $\mathcal{U}$ is defined to represent this property.

7.3.1 Example of unitary state component failure modes

An example of a component with an obvious set of "unitary state" failure modes is the electrical resistor. The EN298 [14][Ann.A] failure mode definition for resistors: OPEN or SHORTED, is used. For a given resistor R the function fm can be applied to find its set of failure modes thus $fm(R) = \{R_{SHORTED}, R_{OPEN}\}$. A resistor cannot fail with the conditions open and short active at the same time, that would be physically impossible! The conditions OPEN and SHORT are thus mutually exclusive. Because of this, the failure mode set $F = fm(R)$

is 'unitary state'. These concepts are expanded in section 7.6.

A general case can be made by taking a set F (with $f_1, f_2 \in F$) representing a collection of component failure modes. A Boolean function $\mathcal{ACTIVE}$ is defined that returns whether a fault mode is active (true) or dormant (false). It can be said that if any pair of fault modes is active at the same time, then the failure mode set is not unitary state: formally;

$$\exists f_1, f_2 \in F \; where \; (f_1 \neq f_2 \wedge \mathcal{ACTIVE}(f_1) \wedge \mathcal{ACTIVE}(f_2)) \implies F \notin \mathcal{U}. \tag{7.10}$$

That is to say that it is impossible that any pair of failure modes can be active at the same time for the failure mode set F to exist in the family of sets $\mathcal{U}$. Note where there are more than two failure modes, by banning any pairs from being active at the same time, larger combinations are banned as well.

Design Rule: Unitary State All components must have unitary state failure modes to be used with the FMMD methodology and for base components this is usually the case. Most simple components fail in one clearly defined way and generally stay in that state. Traditional FMEA also has problems dealing with non unitary state failure modes. This is mainly because combinations of failure modes could cause effects very difficult to predict (as they are in effect new failure modes of the component). However, where a complex component is used, for instance a micro-controller with several modules that could all fail simultaneously, a process of reduction into smaller theoretical components will have to be made. This can be termed 'heuristic de-composition'. A modern micro-controller will typically have several modules which are configured to operate on pre-assigned pins on the device. Typically voltage inputs ($\mathcal{ADC}_{10}$/ $\mathcal{ADC}_{12}$), digital input and outputs, PWM (pulse width modulation), UARTs and other modules will be found on simple cheap micro-controllers [63]. For instance, the voltage reading functions which consist of a multiplexer and ADC—which must work together to channel readings— could be considered to be components inside the micro-controller package. The micro-controller thus becomes a collection of smaller components that can be analysed separately [2]. Were this constraint not to be applied, each component would not contribute N failure modes, but potentially 2^N. This would make the job of analysing the failure modes in a functional grouping impractical due to state explosion.

7.4 Handling Simultaneous Component Faults

For some integrity levels of static analysis, there is a need to consider not only single failure modes in isolation, but cases where more than one failure mode may occur simultaneously. Note that the 'unitary state' conditions apply to failure modes within a component. This does not preclude the possibility of two or more components failing simultaneously. It is an implied requirement of EN298 [14] for instance, to consider double simultaneous faults[3]. To generalise, it may be necessary to consider N simultaneous failure modes when analysing a functional group. This involves finding all combinations of failures modes of size N and less. The power-set, when applied

[2]It is common for the signal paths in a safety critical product to be traced, when examining a complex component like a micro-controller, the process of heuristic de-compostion is typically applied.

[3]Under the conditions of LOCKOUT [14] in an industrial burner controller that has detected one fault already. However, from the perspective of static failure mode analysis, this amounts to dealing with double simultaneous failure modes.

to a set S is the set of all subsets of S, including the empty set [4] and S itself. The power-set concept is augmented here to deal with counting the number of combinations of failures to consider under the conditions of simultaneous failures. In order to consider combinations for the set S where the number of elements in each subset of S is N or less, a concept of the 'cardinality constrained power-set' is proposed and described in the next section.

7.5 Cardinality Constrained Power-set

A Cardinality Constrained power-set is one where subsets of a cardinality greater than a threshold are not included. This threshold is called the cardinality constraint. To indicate this, the cardinality constraint $\leq cc$ is subscripted to the power-set symbol thus $\mathcal{P}_{\leq cc}$. Consider the set $S = \{a, b, c\}$.

The power-set of S:

$$\mathcal{P}S = \{\emptyset, \{a, b, c\}, \{a, b\}, \{b, c\}, \{c, a\}, \{a\}, \{b\}, \{c\}\}.$$

$\mathcal{P}_{\leq 2}S$ means all non-empty subsets of S where the cardinality of the subsets is less than or equal to 2.

$$\mathcal{P}_{\leq 2}S = \{\{a, b\}, \{b, c\}, \{c, a\}, \{a\}, \{b\}, \{c\}\}.$$

Note that $\mathcal{P}_{\leq 1}S$ (non-empty subsets where cardinality ≤ 1) for this example is:

$$\mathcal{P}_{\leq 1}S = \{\{a\}, \{b\}, \{c\}\}.$$

Calculating the number of elements in a Cardinality Constrained power-set A k combination is a subset with k elements. The number of k combinations (each of size k) from a set S with n elements (size n) is the binomial coefficient [93] shown in equation 7.11.

$$C_k^n = \binom{n}{k} = \frac{n!}{k!(n-k)!}. \tag{7.11}$$

To find the number of elements in a cardinality constrained subset S with up to cc elements in each combination sub-set, the sum of combinations must be added, from 1 to cc thus:

$$|\mathcal{P}_{\leq cc}S| = \sum_{k=1}^{cc} \frac{|S|!}{cc!(|S| - cc)!}. \tag{7.12}$$

7.5.1 Actual Number of combinations to check with Unitary State Fault mode sets

If all of the fault modes in S were independent, the cardinality constrained power-set calculation (in equation 7.12) would give the correct number of test case combinations to check. Because sets of failure modes in FMMD analysis are constrained to be unitary state, the actual number of test cases to check will usually be less than this. This is because certain combinations of faults within a components failure mode set are impossible

[4] The empty set ($\emptyset$) is a special case for FMMD analysis, it simply means there is no fault active in the functional group under analysis.

under the conditions of unitary state failure mode. To modify equation 7.12 for unitary state conditions, the number of component 'internal combinations' for each component must be subtracted from the total for the functional grouping under analysis. Note it is necessary to sequentially subtract using combinations above 1 up to the cardinality constraint. For example, say the cardinality constraint was 3, it would be necessary to subtract both $|\binom{n}{2}|$ and $|\binom{n}{3}|$ for each component in the functional grouping.

7.5.1.1 Example: Two Component functional grouping Cardinality Constraint of 2

For example: given a simple functional grouping with two components R and T, of which

$$fm(R) = \{R_o, R_s\}$$

and

$$fm(T) = \{T_o, T_s, T_h\}.$$

This means that the functional grouping $FG = \{R, T\}$ will have a component failure mode set of $fm(FG) = \{R_o, R_s, T_o, T_s, T_h\}$. Note this set of failure modes is as would be used for single failure analysis. For a cardinality constrained powerset of 2, because there are 5 error modes ($|fm(FG)| = 5$), applying equation 7.12 gives:

$$|P_{\leq 2}(fm(FG))| = \frac{5!}{1!(5-1)!} + \frac{5!}{2!(5-2)!} = 15.$$

This is composed of $\binom{5}{1}$, five single fault modes, and $\binom{5}{2}$, ten double fault modes. However the failure modes are mutually exclusive within a component. It is necessary then, to subtract the number of 'internal' component fault combinations for each component in the functional grouping. For component R there is only one internal component fault that cannot exist $R_o \wedge R_s$. As a combination $\binom{2}{2} = 1$. For the component T which has three fault modes $\binom{3}{2} = 3$. Thus for $cc = 2$, under the conditions of unitary state failure modes in the components R and T, it is necessary to subtract (3+1). The number of combinations to check is thus 11, $|\mathcal{P}_{\leq 2}(fm(FG))| = 11$, for this example, and this can be verified by listing all the required combinations:

$$\mathcal{P}_{\leq 2}(fm(FG)) = \{\{R_oT_o\}, \{R_oT_s\}, \{R_oT_h\}, \{R_sT_o\}, \{R_sT_s\}, \{R_sT_h\}, \{R_o\}, \{R_s\}, \{T_o\}, \{T_s\}, \{T_h\}\}$$

whose cardinality is indeed, 11.

7.5.1.2 Establishing Formulae for unitary state failure mode cardinality calculation

The cardinality constrained power-set in equation 7.12, can be modified for unitary state failure modes. Let C be a set of components (indexed by $j \in J$) that are members of the functional group FG i.e. $\forall j \in J, C_j \in FG$.

Let $|fm(C_j)|$ indicate the number of mutually exclusive fault modes of component C_j.

Let $fm(FG)$ be the collection of all failure modes from all the components in the functional group.

Let SU be the set of failure modes from the functional grouping where all FG is such that components C_j are in 'unitary state' i.e. $(SU = fm(FG)) \wedge (\forall j \in J, fm(C_j) \in \mathcal{U})$, then

$$|\mathcal{P}_{cc}SU| = \sum_{k=1}^{cc} \frac{|SU|!}{k!(|SU|-k)!} - \sum_{j \in J} \binom{|FM(C_j)|}{2}. \tag{7.13}$$

Expanding the combination in equation 7.13

$$|\mathcal{P}_{cc}SU| = \sum_{k=1}^{cc} \frac{|SU|!}{k!(|SU|-k)!} - \sum_{j \in J} \frac{|FM(C_j)|!}{2!(|FM(C_j)|-2)!}. \tag{7.14}$$

Equation 7.14 is useful for an automated tool that would verify that a single or double simultaneous failures model has complete failure mode coverage. By knowing how many test cases should be covered, and checking the cardinality associated with the test cases, complete coverage would be verified.

7.5.2 Example: Pt100 Verifying complete coverage for a cardinality constrained power-set of 2

The Pt100 example in 5.6 which performs double failure mode FMMD analysis is used as an example. It is important to check that all possible double fault combinations have been covered. Using the equation 7.14 to determine the number of failure scenarios, or checks, necessary for complete failure coverage.

$$|\mathcal{P}_{\leq cc}SU| = \sum_{k=1}^{cc} \frac{|SU|!}{k!(|SU|-k)!} - \sum_{j \in J} \frac{|FM(C_j)|!}{2!(|FM(C_j)|-2)!}. \tag{7.15}$$

$|FM(C_j)|$ will always be 2 here, as all the components are resistors and have two failure modes. Populating this equation with $|SU| = 6$ and $|FM(C_j)| = 2$.

$$|\mathcal{P}_{\leq 2}SU| = \sum_{1..2}^{k} \frac{6!}{k!(6-k)!} - \sum_{1..3} \frac{2!}{2!(2-2)!} \tag{7.16}$$

$|\mathcal{P}_2 SU|$ is the number of valid combinations of faults to check under the conditions of unitary state failure modes for the components (a resistor cannot fail by being shorted and open at the same time). Expanding the summations:

$$NoOfTestCasesToCheck = \frac{6!}{1!(6-1)!} + \frac{6!}{2!(6-2)!} - \Big(\frac{2!}{2!(2-2)!} + \frac{2!}{2!(2-2)!} + \frac{2!}{2!(2-2)!}\Big),$$

$$NoOfTestCasesToCheck = 6 + 15 - (1 + 1 + 1) = 18.$$

As the test cases are all different and are of the correct cardinalities (6 single faults and (15-3) double) there is confidence that all 'double combinations' of the possible faults have been checked in the Pt100 circuit (see section 5.7).

7.6 Component Failure Modes and Statistical Sample Space

A sample space is defined as the set of all possible outcomes. For a component in FMMD analysis, this set of all possible outcomes is its normal (or 'correct') operating state and all its failure modes. Failure modes can be considered as events in the sample space. When dealing with failure modes, the state where the component is working correctly or 'OK' (i.e. operating with no error) is not useful. For FMEA the analyst is interested only in ways in which it can fail. By definition, while all components in a system are 'working correctly', that system will not exhibit faulty behaviour. Thus the statistical sample space Ω for a component or derived component C is:

$$\Omega(C) = \{OK, failure\_mode_1, failure\_mode_2, failure\_mode_3, \ldots, failure\_mode_N\}.$$

The failure mode set F for a given component or derived component C is therefore $fm(C) = \Omega(C) \backslash \{OK\}$ (or expressed as $\Omega(C) = fm(C) \cup \{OK\}$).

The OK statistical case is usually the largest in probability, and is therefore of interest when analysing systems from a statistical perspective. For these examples, the OK state is not represented area proportionately, but is included in the diagrams. This type of diagram is germane to the application of conditional probability calculations such as Bayes theorem [93]. The current failure modelling methodologies (FMECA [25], FTA [84][70], FMEDA [95]) use Bayesian statistics to justify their methodologies. That is to say, a base component or a sub-system failure has a probability of causing given system level failures[5]. Another way to view this is to consider the failure modes of a component, with the OK state, as a universal set Ω, where all sets within Ω are partitioned. Figure 7.3 shows a partitioned set representing component failure modes $\{B_1 ... B_3, OK\}$: partitioned sets where the OK or empty set condition is included, obey unitary state conditions. Because the subsets of Ω are partitioned, it can be stated that these failure modes are unitary state.

7.7 Components with Independent failure modes

Suppose that a component that can fail simultaneously with more than one failure mode is included in an analysis. This would make it impossible to model as 'unitary state'.

De-composition of complex component. There are two ways in which this can be dealt with. The component could be considered a composite of two simpler components, and their interaction modelled to create a derived component (i.e. use FMMD). The second way would be to consider the combinations of non-mutually exclusive failure modes as new failure modes: this approach is discussed below.

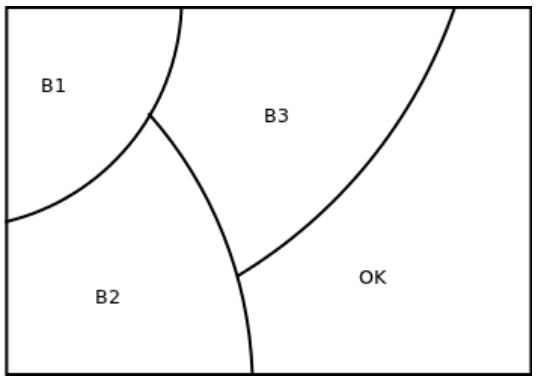


Figure 7.3: Component with three failure modes as partitioned sets

Combinations become new failure modes. The combinations of the non-mutually exclusive failure modes could be considered as new failure modes. An Euler diagram representation of an example component with three failure modes[6] $\{B_1, B_2, B_3, OK\}$ is presented in figure 7.3. For the purpose of example consider $\{B_2, B_3\}$

[5]FMECA has a β value that directly corresponds to the probability that a given part failure mode will cause a given system level failure/event.

[6]OK is really the empty set, but the term OK is more meaningful in the context of component failure modes

to be intrinsically mutually exclusive, but B_1 to be independent. This means there is the possibility of two new combinations $B_1 \cap B_2$ and $B_1 \cap B_3$. These are represented as shaded sections of figure 7.4.

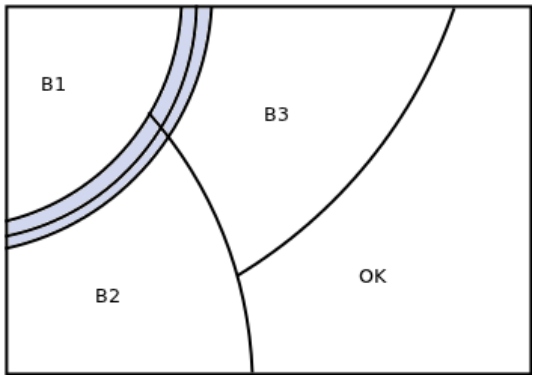


Figure 7.4: Component with three failure modes where B_1 is independent

The probabilities for the shaded areas can be calculated, assuming the failure modes are statistically independent, by multiplying the probabilities of the members of the intersection. The function P is used to return the probability of a failure mode, or combination thereof. Thus for $P(B_1 \cap B_2) = P(B_1)P(B_2)$ and $P(B_1 \cap B_3) = P(B_1)P(B_3)$.

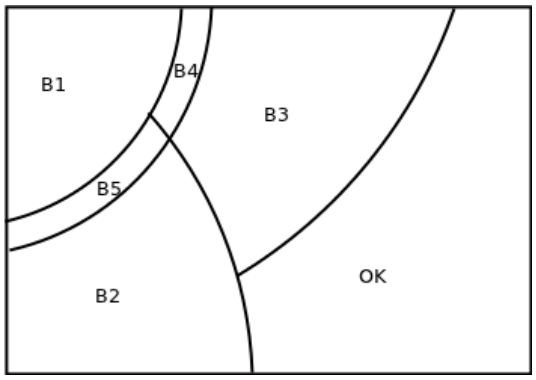


Figure 7.5: Component with two new failure modes

Consider the shaded areas as new failure modes of the component (see figure 7.5). Because of the combinations, the probabilities for the failure modes B_1, B_2 and B_3 will now reduce. The prime character ($'$), to represent the altered value for a failure mode, i.e. B_1' represents the altered value for B_1. Thus

$$P(B_1') = P(B_1) - P(B_1 \cap B_2) - P(B_1 \cap B_3) \ ,$$

$$P(B_2') = P(B_2) - P(B_1 \cap B_2) \ and$$

$$P(B_3') = P(B_3) - P(B_1 \cap B_3) \ .$$

Two new component failure modes B_4 and B_5 have been created as shown in figure 7.5. Their probabilities expressed as $P(B_4) = P(B_1 \cap B_3)$ and $P(B_5) = P(B_1 \cap B_2)$.

7.8 Critiques

7.8.1 Problems in choosing membership of functional groupings

The choice of components for functional groupings is one to be made by the analyst. The guiding principle it to choose components that are functionally adjacent and try to create the smallest groups possible. There are some mistakes that an analyst could make when choosing the members of functional groups. These are:

- Choosing components that are not functionally adjacent — i.e. components that do not work together to perform a specific function,
- Not including components that may have side effects on the functional grouping, but are not obviously connected.

If a deliberately 'bad' functional grouping were chosen it would be found that, on analysis, the component failure modes would not aggregate i.e. be collectable as common symptoms. This would be because, with non-functionally adjacent components, their failures will typically cause non-common failure symptoms. That is a well defined module will typically have a larger number of component failures than failure symptoms. With components that are not interacting, it is unlikely to see good aggregation of symptoms. This property could be of use in future automated FMMD tools to warn of potentially poorly chosen functional groupings.

7.8.1.1 Side Effects: A Problem for FMMD analysis

A problem with modularising according to functionality is that components that would intuitively be associated with one functional grouping could cause unintended side effects in other functional groupings. For instance to have a component that on failing $SHORT$ could bring down a voltage supply rail, could have drastic consequences for other functional groups in the system.

7.8.1.2 Example de-coupling capacitors in logic circuits

A good example of a component failure that can induce side effects in other components, are de-coupling capacitors, often used over the power supply pins of all chips in a digital logic circuit. Were any of these capacitors to fail $SHORT$, they could bring down the supply voltage to the other logic chips. To a power-supply, shorted capacitors on the supply rails are a potential source of the symptom, $SUPPLY\_SHORT$. In a logic chip/digital circuit functional grouping open capacitors are a potential source of symptoms caused by the failure mode $INTERFERENCE$. A possible solution to this is to include the de-coupling capacitors in the power-supply functional grouping.

A de-coupling capacitor going $OPEN$ might not be considered relevant to a power-supply module (even though there might be additional noise on its output rails). But in functional grouping terms, the power supply now has a new symptom, that of $INTERFERENCE$. Some logic chips are more susceptible to $INTERFERENCE$ than others. A logic chip with de-coupling capacitor failing, may operate correctly but interfere with other chips in the circuit. There is no reason why de-coupling capacitors cannot be included in each functional grouping that could be affected by $INTERFERENCE$, meaning that the same de-coupling

capacitors can be members of different functional groupings. This allows for the general principle of a component failure affecting more than one functional grouping in a circuit. This allows functional groups to share components where necessary. With poorly chosen functional groupings it would be possible to miss side effects in analysis.

Chapter 8

Conclusion

This study has examined the four main FMEA variants. It has exposed shortcomings in these methodologies, which can be summed up as an inability to model hybrid software and hardware systems, a problem with state explosion and difficulty of re-use of analysis. The FMECA and FMEDA variants also suffer from embedding subjective and objective assessments of failure modes. This thesis proposes modularised FMEA—Failure Mode Modular De-composition (FMMD)—to overcome some of these problems. This modularised version had been supported by the work already established by the definition of failure modes for base components in the literature [24, 26, 14, 15]. Specific electronic examples were analysed using FMMD to test circuit topologies with conventional and circular signal paths and mixed digital and analogue designs. For all these examples, the state explosion related performance was compared with that of traditional FMEA. In all cases there was a performance gain, that is to say that for all but trivial cases, the number of manual analysis operations to perform was significantly reduced. Not only this, but the analysis naturally provided modules which could be re-used, both in the same circuit and other circuits and potentially future projects as well.

Traditional FMEA methods have been applied to software, but analysis has always been performed separately from the HFMEA [71, 91]. Using established concepts from contract programming [67] FMMD was extended to analyse software, which facilitated a solution to the software/hardware interfacing problem [74]. Two examples of hybrid software/hardware systems were analysed as integrated FMMD models as proof of concept. The first example in chapter 6, was presented to the System Safety IET conference in 2012 [20]. Chapter 7 viewed FMMD from a formal perspective and examined problems and constraints necessary to perform FMEA and FMMD. Theoretical performance models were developed (see section 7.1.3) which showed that with increasing modularisation the number of manual checks to perform for analysis fell, which was validated by examining the reasoning distance performance of the examples from chapter 5. A unitary state failure mode concept was developed (see section 7.3), and it was shown that the FMMD process naturally enforced this throughout the hierarchy of a model. Finally the FMMD process was described algorithmically in appendix B.

In conclusion then, a new method of failure analysis has been devised which improves on established techniques in the following ways:

- FMMD provides the means to determine failure models that integrate software and hardware,
- the state explosion related to exhaustive FMEA reduced from a polynomial to logarithmic order,

- a modular approach to FMEA means that analysis work is re-usable,
- distributed systems, and smart instruments, can now be analysed and assessed,
- multiple failures can be analysed (without an undue state explosion cost).

These benefits require the following assumptions and constraints:

- Failure modes are available for all base components,
- Analysts are capable of finding suitable functional groupings from electronic schematics,
- Functional software and its elements (hardware interfaces, data and functions) can be modelled using contract programming.

Whilst investigating FMMD a number of further areas for research revealed themselves. These are presented below.

8.1 Further Work

8.1.1 How traditional FMEA reports can be derived from an FMMD model.

An FMMD model has a data structure (described by UML diagrams, see figure 4.8) and by traversing an FMMD hierarchy, system level failures can be mapped back to base component failure modes (or combinations thereof). Because these mappings can be determined, reports in the traditional FMEA format (i.e. base component failure mode $\mapsto$ system failure) can be produced. With the addition of base component failure mode statistics [26] reliability predictions for system level failures can be provided. The Pt100 example is revisited for this purpose and analysed for single and double failures, with statistics for base components taken from MIL1991 in section 8.1.2. With an FMMD failure mode model a top down perspective is possible. Each system level failure can have a causation tree produced for it, tracing back to all base component failure modes. This is very closely related to the structure of FTA (top down) failure causation graphs. The possibility of automatically producing FTA diagrams from FMMD models is examined in section 8.1.4.

8.1.2 Statistics: From base component failure modes to System level events/failures.

Knowing the statistical likelihoods of a components failing can give a good indication of the reliability of a system, or in the case of dangerous failures, the Safety Integrity Level of a system. EN61508 [95] requires that statistical data is available and used for all component failure modes analysed by FMEDA. FMMD, as a bottom up methodology can use component failure mode statistical data, and incorporate it into its hierarchical model. Because an FMMD model can be used to generate an FMEA report, with additional base component failure mode statistics an FMEDA report can be produced. FMMD has been applied with component failure statistics to the Pt100 example in appendix A.3.8. This demonstrates FIT values being obtained for single and doubly sourced system failure modes in a way that is compatible with FMEDA/EN61508.

8.1.3 Composition of functional groupings.

The members of a functional grouping are chosen to be components that work together to perform a specific function. The choice of functional grouping membership is made by the analyst. The act of choosing components to form a functional grouping raises questions about the circuit under investigation. Ideally functional groupings will be able to act as standalone modules. An inverting amplifier configuration, or a low pass filter are good examples of these: they have clear inputs and outputs, and are resilient to what they are connected to at the output (in electronics terms they have low output impedance). In defining members for functional groupings the analyst is forced to consider the interfaces between elements of circuitry to identify modules. The aim is to prevent undue influence on modules identified from circuitry they are/may be connected to. Consider the resistor capacitor low pass stage first looked at in example 5.3.1. This circuit element, while applying a filtering effect, has a high output impedance. With a simple OpAmp buffer amplifier on its output stage, it becomes an effective low impedance output standalone module[1]. The resistor/capacitor low pass stage and the OpAmp are good candidates therefore for being considered as a standalone module, and thus a functional grouping.

However, different analysts may choose different functional groupings when analysing the same circuit. This means that functional groupings are not guaranteed to be unique. This apparent anomaly is explored in the examples 5.1, 5.4 where different structures of the FMMD hierarchy were used to analyse the same circuitry. The same system level failure modes were obtained, but the more de-composed examples offered better performance in terms of comparison complexity. Potential problems of side effects and functional grouping membership choices are discussed in section 7.8.1. Further work may be required to apply justification for the choice of membership in functional groupings. For software already written this problem does not exist as the choice of membership has already been made by the programmer.

8.1.4 Deriving FTA diagrams from FMMD models

Fault Tree Analysis (FTA) [27] is a top down methodology that draws a fault tree—or top down fault causation diagram—for each given top-level failure. With an FMMD model, all the causes of system failures down can be traced to the base component level. The DAG produced from an FMMD analysis could be considered as a unification of all FTA trees of a system. This would in fact, be enough to create all fault causation trees, but FTA introduces concepts of operational and environmental states, and inhibit gates. The FMEA philosophy in relation to these three concepts are to assume that they are worst cases, that they *may* occur, and determine what system failures may arise. The FTA perspective is that some safety can be built in by preventing certain things happening (inhibit gates), and by considering different behaviour due to environmental or operational states [84, 70]. If FMMD is required to produce full FTA diagrams, these attributes must be added to the FMMD UML model[2].

Environment, operational states and inhibit gates: additions to the UML model. FTA, in addition to using symbols borrowed from digital logic introduces three new symbols to model environmental, operational

[1] A well behaved, or ideal electronics 'module' will have a high impedance input (i.e. it will not overload and affect any driving stages) and a low output impedance (i.e. it will drive an electrical load at the output without being affected its-self).

[2] Top down failure mode models, such as FTA, are additionally useful in guiding diagnostic analysis.

state and inhibit gates; how these can be incorporated into the FMMD model is discussed below. A system will be expected to perform in a given environment. Environment in the context of this study means external influences under which the system could be expected to work. A typical data sheet for an electrical component will give a working temperature range: mechanical components could be specified for stress and loading limits. It is unusual to have failure modes described in product literature, although for complicated components with firmware, errata documents [64] are sometimes produced.

Systems may have distinct operational states. For instance, a safety critical controller may have a LOCK-OUT state where it has detected a serious problem and will not continue to operate until authorised human intervention takes place. A safety critical circuit may have a self test mode which could be operated externally: a micro-processor may have a SLEEP mode etc. To make FMMD compatible with FTA operational states and environmental conditions should be factored into the UML model. An undesired condition may occur where it could be necessary to inhibit some action of the system. This is rather like a logical guard criterion. For instance in the gas burner standard EN298 it states that a flame detector must confirm that a pilot flame has been established before the main burner fuel can be applied. In FTA terms this would be an 'inhibit' condition on the main fuel, i.e. PILOT_NOT_CONFIRMED. The nature of these three attributes is examined and decisions are made as how they should fit into the UML model for FMMD developed in section 4.5.2.

Environmental Modelling. The external influences/environment could typically be temperature ranges, levels of electrical interference, high voltage contamination on supply lines, radiation levels etc. Environmental influences will affect specific components in specific ways[3]. Environmental analysis is thus applicable to components. Environmental influences, such as over-stress due to voltage can be eliminated by down-rating components as discussed in section 2.3. With given environmental constraints, it is therefore possible to eliminate some failure modes from the model.

Operational states. Within the field of safety critical engineering, elements are often encountered that include test or self-test facilities. Degraded performance (such as only performing certain functions in an emergency) and lockout/emergency conditions are also common conditions that are considered. These can be broadly termed operational states. The UML class most appropriate to hold a relationship to operational states must be chosen. Consider for instance an electrical circuit that has a TEST line. When the TEST line is activated, it supplies a test signal which will validate the circuit. This circuit will have two operational states, NORMAL and TEST mode. It seems more appropriate to apply the operational states to functional groupings which by definition implement functionality, or purpose. On this basis operational states are associated with functional groupings.

Inhibit Conditions. Inhibit conditions and the symbols used for them are described in [70][p.40]. Some failure modes may only be active given specific environmental conditions or when other failures are already active. To model this, an 'inhibit' class has been added. This is an optional attribute of a failure mode. This

[3]A good example of a part affected by environmental conditions, in this case temperature, is the opto-isolator [104] which typically starts having performance problems at 60 ^oC and above. Most electrical components are robust to temperature variations and would not normally require special environmental consideration/attributes.

inhibit class can be triggered on a combination of environmental or failure modes. In the UML diagram, this is therefore, linked with both environmental conditions and failure modes.

UML Diagram Additional Objects. The additional objects System, Environment, Inhibit and Operational States are added to UML diagram in figure 4.8 are represented in figure 8.1.

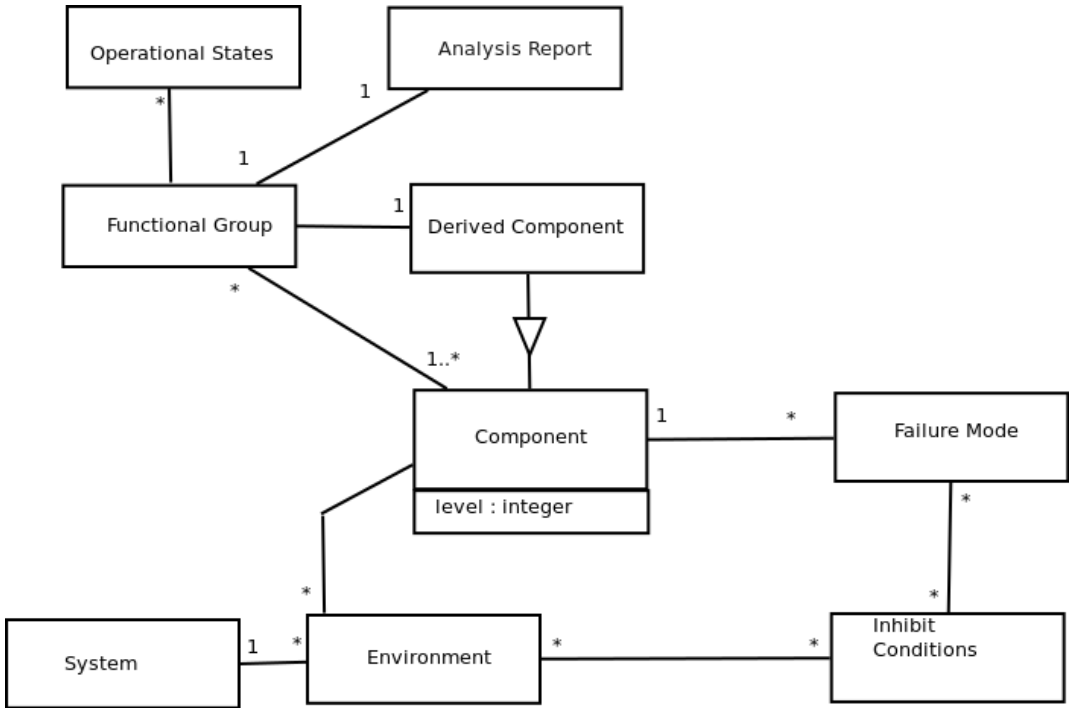


Figure 8.1: FMMD UML diagram extended for potential compatibility with FTA: incorporating Environmental, Operational State and Inhibit gates

8.1.5 Retrospective failure mode analysis and FMMD

The reasons for applying retrospective failure mode analysis could be:

- to re-visit a safety analysis after a small hardware or software change,
- upon discovery of a new base component failure mode—or in software—a new contract programming requirement,
- to determine the failure mode behaviour of an previously un-assessed sub-system/instrument used in safety critical verification.

FMMD can be applied retrospectively to a project, and because of its modular nature, coupled with its 'bottom-up work flow' it can reveal previously undetected system failure modes. This is because the analyst is forced to deal with all component failure modes when applying the FMMD process, and all failure modes of the resultant derived components as the hierarchy is built. FMMD requires that all failure modes of components in a functional grouping are resolved to a symptom in the resulting derived component. As 'complete' analysis can be enforced, FMMD can find failure modes which were missed by other FMEA processes; meaning that the FMMD process can expose un-handled failure modes.

Retrospective failure mode analysis and software. Retrospective FMMD can be applied to electronic and software hybrid systems. The electronic components failure modes are established in the literature [24, 26, 14, 15]. Each function in the software would have to be assigned a 'design contract' [67] (where violations of contract clauses will be treated as failure modes in FMMD).

Effect of newly discovered failure modes in components. Using traditional FMEA when discovering a new failure mode in a component or sub-system it could be difficult to know which parts of the FMEA analysis to re-visit. For instance, which components in the system should the newly discovered failure mode be checked against? This concern is linked to the concepts behind the need for failure mode coverage against all components in the system, that provoked discussions leading to idealised XFMEA requirements (see section 3.2). Using FMMD only those modules in the hierarchy above the component with the new failure mode need be re-visited. The failure mode DAGs (see chapter 4) can be traced to determine exactly which functional groupings exist in the hierarchy above the affected base components. This means that with FMMD the re-work task can be precisely defined. Also where a new base component failure mode is merged with an existing symptom in the analysis the re-work need not continue above it in the hierarchy. That is a new base component failure mode may cause a symptom already found in the analysis hierarchy. Finding these could be automated in a software tool that can traverse the failure mode trees. With the contracts in place for the software functions, they can be integrated into the FMMD model. FMMD models both software and hardware; thus it can be verified that all failure modes from the electronics module have been dealt with by the controlling software. If not, this would be an un-handled error condition relating to the software/hardware interface. This again can be flagged using an automated tool. By performing FMMD on a software electronic hybrid system, design deficiencies are revealed in both the software, the electronics and the software/electronics interface. FMEDA does not handle software —or—the software/hardware interface. It thus potentially misses many undetected failures (in EN61508 terms

undetected-dangerous and undetected safe failures). In Safety Integrity Level (SIL) [95] terms, by identifying undetectable faults and fixing them, the safe failure fraction (SFF) is raised.

8.1.6 Creation of a software FMMD tool.

A software tool could be created with an extendible library/database of base and derived components. This tool could guide the user through the analysis and hierarchy construction processes and use the constraints and algorithms defined in appendix B and the UML diagram developed (see figure 4.8) for verification of the process and models produced.

8.2 Objective and Subjective Reasoning stages

The act of applying failure mode effects analysis, is commonly performed from an 'engineering' oriented cause and effect perspective. This is the realm of the objective. The executive decisions about deploying systems are in the domain of management and politics. The dangers, or potential negative effects of a safety critical system depend not only on the system itself, but on the environment in which they are used and other human factors such as the training level of operatives, psychological and logical factors in the Human Machine Interface (HMI) [98].

Objective and Subjective Reasoning in FMEA: Three Mile Island nuclear accident example. An example of objective and subjective factors is demonstrated in the accident report on the 1979 Three Mile Island nuclear accident [53][App.D]. Here, a vent valve for the primary reactor coolant (pressurised water) became stuck open. This condition caused an objectively derived failure mode — 'leakage of coolant' — due to a stuck valve. This, if recognised correctly by the operators, would have lead quickly to a reactor shut-down and a maintenance procedure to replace the valve. The failure was not recognised in time and coolant was lost until a partial meltdown of the reactor fuel occurred, with a resulting leak of radioactive material into the environment. For the objective failure mode determined by FMEA, that of leakage of coolant, it would not be reasonable to expect this to go unchecked and unresolved for an extended period and cause such a critical failure. The criticality level of that failure mode was therefore subjective. It was not known how the operators would have reacted and deficiencies in the Human Machine Interface (HMI) were not a factor in the failure analysis.

Further Work: Objective and Subjective Reasoning in FMEA. Criticality prediction can be said to be in the domain of subjective reasoning. With an objectively defined system level failure it is often required to next determine its level of criticality, or how serious the risk posed would be. Two methodologies have started to consider this aspect, FMECA [25] with its criticality and probability factors, and FMEDA [95, 39] with its classification of dangerous and safe failures. It is the author's opinion that more work is required to clarify this area. Accurate models of objective failure modes, are seen by the author to be a pre-requisite for subjective assessment. The scope of FMMD is the objective level only, but offers significant benefits in terms of accuracy and labour savings. It also offers integrated modelling of software and hardware. Its failure mode model can also be used to assist in producing traditional FMEA formats.

Appendix A

Detailed FMMD analyses

For clarity the detailed workings of the FMMD analysis stages in many of the examples in chapters 5 and 6 have been moved here for reference.

A.1 Bubba Oscillator FMMD analyses

Detailed workings of the FMMD for the Bubba Oscillator are presented below.

A.1.1 PHS45 Detailed Analysis

FMEA study of a resistor and capacitor in use as a phase changer.

Table A.1: PhaseShift: Failure Mode Effects Analysis: Single Faults

Failure cause	$PHS45$ **Effect**	**Symptom**
FS1: R SHORT	0 degree's of phase shift	$0\_phaseshift$
FS2: R OPEN	No Signal	$nosignal$
FS3: C SHORT	Grounded,No Signal	$nosignal$
FS4: C OPEN	0 degree's of phase shift	$0\_phaseshift$

Collecting symptoms from table A.1, a derived component, $PHS45$ is created with the following failure modes:

$$fm(PHS45) = \{0\_phaseshift, nosignal\}.$$

A.1.2 Bubba Oscillator: One Large Functional Group: Detailed Analysis

Table A.2: Bubba Oscillator: Failure Mode Effects Analysis: One Large Functional Group

Failure cause	*BubbaOscillator* **Effect**	**Symptom**
FS1: $PHS45_1$ 0_*phaseshift*	osc frequency high	HI_{fosc}
FS2: $PHS45_1$ *no_signal*	signal lost	NO_{osc}
FS3: $NIBUFF_1$ L_{up}	output high No Oscillation	NO_{osc}
FS4: $NIBUFF_1$ L_{dn}	output low No Oscillation	NO_{osc}
FS5: $NIBUFF_1$ N_{oop}	output low No Oscillation	NO_{osc}
FS6: $NIBUFF_1$ L_{slew}	signal lost	NO_{osc}
FS7: $PHS45_2$ 0_*phaseshift*	osc frequency high	HI_{fosc}
FS8: $PHS45_2$ *no_signal*	signal lost	NO_{osc}
FS9: $NIBUFF_2$ L_{up}	output high No Oscillation	NO_{osc}
FS10: $NIBUFF_2$ L_{dn}	output low No Oscillation	NO_{osc}
FS11: $NIBUFF_2$ N_{oop}	output low No Oscillation	NO_{osc}
FS12: $NIBUFF_2$ L_{slew}	signal lost	NO_{osc}
FS13: $PHS45_3$ 0_*phaseshift*	osc frequency high	HI_{fosc}
FS14: $PHS45_3$ *no_signal*	signal lost	NO_{osc}
FS15: $NIBUFF_3$ L_{up}	output high No Oscillation	NO_{osc}
FS16: $NIBUFF_3$ L_{dn}	output low No Oscillation	NO_{osc}
FS17: $NIBUFF_3$ N_{oop}	output low No Oscillation	NO_{osc}
FS18: $NIBUFF_3$ L_{slew}	signal lost	NO_{osc}
FS19: $PHS45_4$ 0_*phaseshift*	osc frequency high	HI_{fosc}
FS20: $PHS45_4$ *no_signal*	signal lost	NO_{osc}
FS21: $INVAMP$ $OUTOFRANGE$	signal lost	NO_{osc}
FS22: $INVAMP$ $ZEROOUTPUT$	signal lost	NO_{osc}
FS23: $INVAMP$ $NOGAIN$	signal lost	NO_{osc}
FS24: $INVAMP$ $LOWPASS$	signal lost	NO_{osc}

Collecting symptoms from table A.2, the derived component *BubbaOscillator* is created with the following failure modes:

$$fm(BubbaOscillator) = \{NO_{osc}, HI_{fosc}\}.$$

A.1.3 BUFF45: Detailed Analysis

Table A.3: BUFF45: Failure Mode Effects Analysis

Failure cause	$BUFF45$ **Effect**	**Symptom**
FS1: $PHS45_1$ $0\_phaseshift$	no phase shift	$0\_phaseshift$
FS2: $PHS45_1$ no_signal	signal lost	NO_{signal}
FS3: $NIBUFF_1$ L_{up}	output high	NO_{signal}
FS4: $NIBUFF_1$ L_{dn}	output low	NO_{signal}
FS5: $NIBUFF_1$ N_{oop}	output low	NO_{signal}
FS6: $NIBUFF_1$ L_{slew}	signal lost	NO_{signal}

Collecting symptoms from table A.3, a derived component $BUFF45$ is created which has the following failure modes:

$$fm(BUFF45) = \{0\_phaseshift, NO\_signal\}.$$

A.1.4 PHS135BUFFERED: Failure Mode Effects Analysis

Table A.4: PHS135BUFFERED: Failure Mode Effects Analysis

Failure cause	$PHS135BUFFERED$ **Effect**	**Symptom**
FS1: $PHS45_1$ $0\_phaseshift$	phase shift low	$90\_phaseshift$
FS2: $PHS45_1$ no_signal	signal lost	NO_{signal}
FS3: $PHS45_2$ $0\_phaseshift$	phase shift low	$90\_phaseshift$
FS4: $PHS45_2$ no_signal	signal lost	NO_{signal}
FS5: $PHS45_3$ $0\_phaseshift$	phase shift low	$90\_phaseshift$
FS6: $PHS45_3$ no_signal	signal lost	NO_{signal}

Collecting symptoms from table A.4, a derived component $PHS135BUFFERED$ is created which has the following failure modes:

$$fm(PHS135BUFFERED) = \{90\_phaseshift, NO\_signal\}.$$

A.1.5 PHS225AMP: Failure Mode Effects Analysis

Table A.5: PHS225AMP: Failure Mode Effects Analysis

Failure cause	$PHS225AMP$ **Effect**	**Symptom**
FS1: $PHS45_1$ $0\_phaseshift$	phase shift low	$180\_phaseshift$
FS2: $PHS45_1$ no_signal	signal lost	NO_{signal}
FS3: $INVAMP$ L_{up}	output high	NO_{signal}
FS4: $INVAMP$ L_{dn}	output low	NO_{signal}
FS5: $INVAMP$ N_{oop}	output low	NO_{signal}
FS6: $INVAMP$ L_{slew}	signal lost	NO_{signal}

Collecting symptoms from table A.5, the derived component $PHS225AMP$ is created with the following failure modes:

$$fm(PHS225AMP) = \{180\_phaseshift, NO\_signal\}.$$

A.1.6 BUBBAOSC: Failure Mode Effects Analysis

Table A.6: BUBBAOSC: Failure Mode Effects Analysis

Failure cause		$BUBBAOSC$ **Effect**		**Symptom**
FS1: $PHS135BUFFERED$ *no_signal*		signal lost		NO_{osc}
FS2: $PHS135BUFFERED$ 90*_phaseshift*		phase shift low		HI_{osc}
FS4: $PHS225AMP$ 180*_phaseshift*		phase shift low		HI_{osc}
FS5: $PHS225AMP$ *NO_signal*		lost signal		NO_{signal}

Collecting symptoms from table A.6, a derived component $BUBBAOSC$ is created which has the following failure modes:

$$fm(BUBBAOSC) = \{HI_{osc}, NO\_signal\}.$$

A.2 Sigma Delta Detailed FMMD Analyses

This section of the appendix contains FMEA tables for the $\Sigma\Delta ADC$.

A.2.1 FMMD Analysis of Summing Junction Integrator: SUMJINT

Table A.7: Summing Junction Integrator($SUMJINT$): Failure Mode Effects Analysis

Failure cause	$SUMJINT$ **Effect**	**Symptom**
FS1: $R1\ OPEN$	V_{in} dominates input	$V_{in}DOM$
FS2: $R1\ SHORT$	V_{fb} dominates input	$V_{fb}DOM$
FS3: $R2\ OPEN$	V_{fb} dominates input	$V_{fb}DOM$
FS4: $R2\ SHORT$	V_{in} dominates input	$V_{in}DOM$
FS5: $IC1\ HIGH$	output perm. high	HIGH
FS6: $IC1\ LOW$	output perm. low	LOW
FS7: $IC1\ NOOP$	no current to drive C1	NO_INTEGRATION
FS8: $IC1\ LOW\_SLEW$	signal delay to C1	NO_INTEGRATION
FS9: $C1\ OPEN$	no capacitance	NO_INTEGRATION
FS10: $C1\ SHORT$	no capacitance	NO_INTEGRATION

Collecting symptoms from table A.7, the derived component $SUMJINT$ is created with the following failure modes:

$$fm() = \{V_{in}DOM, V_{fb}DOM, NO\_INTEGRATION, HIGH, LOW\}.$$

A.2.2 FMMD Analysis of High Impedance Signal Buffer : HISB

Table A.8: High Impedance Signal Buffer : Failure Mode Effects Analysis

Failure cause	$HISB$ **Effect**	**Symptom**
FS1: $IC2\ HIGH$	output perm. high	HIGH
FS2: $IC2\ LOW$	output perm. low	LOW
FS3: $IC2\ NOOP$	no current to output	$NOOP$
FS4: $IC2\ LOW\_SLEW$	delayed signal	LOW_SLEW

Collecting symptoms from table A.8, the derived component $HISB$ is created with the following failure modes:

$$fm(HISB) = \{HIGH, LOW, NOOP, LOW\_SLEW\}.$$

A.2.3 FMMD Analysis of Digital level to analogue level converter : DL2AL

Table A.9: $PD, IC3$ Digital level to analogue level converter: Failure Mode Effects Analysis

Failure cause	$DS2AL$ **Effect**	**Symptom**
FS1: $PD\ HIGH$	output perm. low	LOW
FS2: $PD\ LOW$	output perm. high	HIGH
FS3: $IC3\ HIGH$	output perm. high	HIGH
FS4: $IC3\ LOW$	output perm. low	LOW
FS5: $IC3\ NOOP$	no current drive	LOW
FS6: $IC3\ LOW\_SLEW$	delayed signal	LOW_SLEW

Collecting symptoms from table A.9, the derived component $DL2AL$ is created with the following failure modes:

$$fm(DL2AL) = \{LOW, HIGH, LOW\_SLEW\}.$$

A.2.4 FMMD Analysis of Digital Buffer : DIGBUF

Table A.10: $IC4, CLOCK$ Digital Buffer: Failure Mode Effects Analysis

Failure cause	$DIGBUF$ **Effect**	**Symptom**
FS1: $CLOCK\ STOPPED$	buffer stopped	STOPPED
FS2: $IC4\ HIGH$	buffer stopped	STOPPED
FS3: $IC4\ LOW$	buffer stopped	STOPPED
FS4: $IC4\ NOOP$	no current drive	LOW

Collecting symptoms from table A.10, the derived component $DIGBUF$ is created with the following failure modes:

$$fm(DIGBUF) = \{LOW, STOPPED\}.$$

A.2.5 FMMD Analysis of buffered integrating summing junction : BISJ

Table A.11: $HISB, SUMJINT$ buffered integrating summing junction($BISJ$): Failure Mode Effects Analysis

Failure cause	$BISJ$ **Effect**	**Symptom**
FS1: $SUMJINT\ V_{in}DOM$	output integral of V_{in}	$OUTPUTSTUCK$
FS2: $SUMJINT\ V_{fb}DOM$	output integral of V_{fb}	$OUTPUTSTUCK$
FS3: $SUMJINT\ NO\_INTEGRATION$	output stuck high or low	$OUTPUTSTUCK$
FS4: $SUMJINT\ HIGH$	output stuck high	$OUTPUTSTUCK$
FS5: $SUMJINT\ LOW$	output stuck low	$OUTPUTSTUCK$
FS6: $HISB\ HIGH$	output perm. high	$OUTPUTSTUCK$
FS7: $HISB\ LOW$	output perm. low	$OUTPUTSTUCK$
FS8: $HISB\ NO\_INTEGRATION$	no current drive	$OUTPUTSTUCK$
FS9: $HISB\ LOW\_SLEW$	delayed signal	$REDUCED\_INTEGRATION$

Collecting symptoms from table A.11, the derived component $BISJ$ is created with the following failure modes:

$$fm(BISJ) = \{OUTPUTSTUCK, REDUCED\_INTEGRATION\}.$$

A.2.6 FMMD Analysis of flip flop buffered : FFB

Table A.12: $DIGBUF, DL2AL$ flip flop buffered(FFB): Failure Mode Effects Analysis

Failure cause	$DIGBUF$ **Effect**	**Symptom**
FS1: $DIGBUF\ STOPPED$	output stuck	$OUTPUTSTUCK$
FS2: $DIGBUF\ LOW$	output stuck low	$OUTPUTSTUCK$
FS3: $DL2AL\ LOW$	output perm. high	$OUTPUTSTUCK$
FS4: $DL2AL\ HIGH$	output perm. low	$OUTPUTSTUCK$
FS5: $DL2AL\ LOW\_SLEW$	slow reaction to input	LOW_SLEW

Collecting symptoms from table A.12, the derived component FFB is created with the following failure modes:

$$fm(FFB) = \{OUTPUTSTUCK, LOW\_SLEW\}.$$

A.2.7 FMMD Analysis of $\Sigma\Delta ADC$: SDADC

Table A.13: $FFB, BISJ\ \Sigma\Delta ADC(SDADC)$: Failure Mode Effects Analysis

Failure cause	FFB **Effect**	**Symptom**
FS1: $FFB\ OUTPUTSTUCK$	value held high or low	$OUTPUT\_OUT\_OF\_RANGE$
FS2: $FFB\ LOW\_SLEW$	values will appear larger	$OUTPUT\_INCORRECT$
FS3: $BISJ\ OUTPUTSTUCK$	value held high or low	$OUTPUT\_OUT\_OF\_RANGE$
FS4: $BISJ\ REDUCED\_INTEGRATION$	values will appear larger	$OUTPUT\_INCORRECT$

Collecting symptoms from table A.13, the derived component $SDADC$ is created with the following failure modes:

$$fm(SDADC) = \{OUTPUT\_OUT\_OF\_RANGE, OUTPUT\_INCORRECT\}.$$

A.3 Standalone temperature controller

FMMD analysis tables from chapter 6.

A.3.1 Read_Pt100: Failure Mode Effects Analysis

Table A.14: Read_Pt100: Failure Mode Effects Analysis

Failure cause	**Failure Effect**	**Symptom**
FC1: RI_{VRGE}	voltage outside range	$VOLTAGE\_HIGH$
FC2: $RADC_{VV_ERR}$	voltage incorrect	VAL_ERR
FC3: $RADC_{HIGH}$	voltage value incorrect	$VOLTAGE\_HIGH$
FC4: $RADC_{LOW}$	voltage value from ADC value low	$VOLTAGE\_LOW$
FC5: post condition fails in function **read_ADC()**	software failure **read_ADC()**	VAL_ERR

Collecting symptoms from table A.14, the derived component *Read_Pt*100 is created with the following failure modes:

$$fm(Read\_Pt100) = \{VOLTAGE\_HIGH, VOLTAGE\_LOW, VAL\_ERR\}.$$

A.3.2 Get_Temperature: Failure Mode Effects Analysis

Table A.15: Get_Temperature: Failure Mode Effects Analysis

Failure cause	**Failure Effect**	**Symptom**
FC1: $Pt100 : Voltage\_High$	Pt100 voltage too high	Pt100_out_of_range
FC2: $Pt100 : Voltage\_Low$	Pt100 voltage too low	Pt100_out_of_range
FC3: $Pt100\_high\_low\_mismatch$	temperature can be calculated from either high or low reading, but should correlate	Pt100_out_of_range
FC4: $Pt100 : VAL\_ERR$	causes an incorrect temperature reading	temp_incorrect
FC5: post condition fails in function **convert_ADC_to_T()**	software failure **convert_ADC_to_T()**	temp_incorrect

Collecting symptoms from table A.15, the derived component $Get\_Temperature$ is created with the following failure modes:

$$fm(Get\_Temperature) = \{Pt100\_out\_of\_range, temp\_incorrect\}.$$

A.3.3 GetError: Failure Mode Effects Analysis

The error value being discussed here is an important concept in PID control. It represents how far from the control target the measured reading is. The lower the PID error value the closer to the controlled systems target/desired value.

Table A.16: GetError: Failure Mode Effects Analysis

Failure cause	**Failure Effect**	**Symptom**
FC1: $Pt100\_out\_of\_range$	pre-condition violated detectable failure mode	KnownIncorrectErrorValue
FC2: $temp\_incorrect$	pre-condition violated undetectable failure mode	IncorrectErrorValue
FC3: post condition fails in function **determine_set_point_error()**	software failure **determine_set_point_error()**	IncorrectErrorValue

Collecting symptoms from table A.16, the derived component $GetError$ is created with the following failure modes:

$$fm(GetError) = \{KnownIncorrectErrorValue, IncorrectErrorValue\}.$$

A.3.4 PID: Failure Mode Effects Analysis

Table A.17: PID: Failure Mode Effects Analysis

Failure cause	**Failure Effect**	**Symptom**
FC1: $KnownIncorrectErrorValue$	pre-condition violated detectable failure mode	KnownControlValueErrorV
FC2: $IncorrectErrorValue$	pre-condition violated undetectable failure mode	IncorrectControlErrorV
FC3: post condition fails in function **PID()**	software failure **PID()**	IncorrectControlErrorV

Collecting symptoms from table A.17, the derived component PID is created with the following failure modes:

$$fm(PID) = \{KnownControlValueErrorV, IncorrectControlErrorV\}.$$

A.3.5 HeaterOutput: Failure Mode Effects Analysis

Table A.18: HeaterOutput: Failure Mode Effects Analysis

Failure cause	**Failure Effect**	**Symptom**
FC1: $PWMstuckHIGH$	pre-condition violated PWM module not working	HeaterOnFull
FC2: $PWMstuckLOW$	pre-condition violated PWM module not working	HeaterOff
FC3: HEATER $SHORT$	heating element resistor SHORT no heating effect	HeaterOff
FC4: HEATER $OPEN$	heating element resistor OPEN no heating effect	HeaterOff
FC5: *output_control* post condition failure	The software supplies the wrong value to the PWM register	HeaterOutputIncorrect

Collecting symptoms from table A.18, the derived component $HeaterOutput$ is created with the following failure modes:

$$fm(HeaterOutput) = \{HeaterOnFull, HeaterOff, HeaterOutputIncorrect\}.$$

A.3.6 LEDOutput: Failure Mode Effects Analysis

Table A.19: LEDOutput: Failure Mode Effects Analysis

Failure cause	**Failure Effect**	**Symptom**
FC1: $TempLEDfails$	LED will not light	FailureIndicated
FC2: $ProcessorLEDfails$	LED will not light	FailureIndicated
FC3: $PWMLEDfails$	LED will not light	FailureIndicated
FC4: GPIO stuck HIGH	LED permanently OFF	FailureIndicated
FC5: GPIO stuck Low	LED permanently ON	FailureIndicated
FC6: Software **SetLEDs()** fails to set outputs correctly	Incorrect Indication Post condition failure	IndicationError

Collecting symptoms from table A.19, the derived component $LEDOutput$ is created with the following failure modes:

$$fm() = \{FailureIndicated, IndicationError\}.$$

A.3.7 Standalone temperature controller: Failure Mode Effects Analysis

Table A.20: Standalone temperature controller: Failure Mode Effects Analysis

Failure cause	**Failure Effect**	**Symptom**
FC1: PID KnownControlValueError	As error is detectable error can be indicated	ControlFailureIndicated
FC2: PID IncorrectControlerrorV	undetectable failure: PID will not control properly	ControlFailure
FC3: HeaterOutput HeaterOnFULL	Heater will constantly apply maximum power	ControlFailureIndicated
FC4: HeaterOutput HeaterOFF	no power supplied to heater	ControlFailureIndicated
FC5: HeaterOutput HeaterOutputIncorrect	incorrect power levels applied to heater	ControlFailure
FC6: LEDOutput FailureIndicated	failure of LED system where failure is detectable	KnownIndicationError
FC7: LEDOutput IndicationError	failure of LED system where failure is undetectable	UnknownIndicationError
FC8: micro-controller PROM_FAULT	un-defined behaviour	ControlFailure
FC9: micro-controller RAM_FAULT	un-defined behaviour	ControlFailure
FC10: micro-controller CPU_FAULT	un-defined behaviour	ControlFailure
FC11: micro-controller ALU_FAULT	incorrect arithmetic performed in processing	ControlFailure
FC12: micro-controller CLOCK_STOPPED	processor will not run indicator leds will not flash	ControlFailureIndicated
FC13: **monitor()**: software fails	postcondition fails	ControlFailure

Collecting symptoms from table A.20 the derived component $TempController$, is created with the following failure modes:

$$fm(TempController) = \{ControlFailureIndicated,$$
$$ControlFailure,$$
$$KnownIndicationError,$$
$$UnknownIndicationError\}.$$

A.3.8 Statistics and FMMD: Pt100 example for single and double failures

Pt100: Single Failures and statistical data. From an earlier example, the model for the failure mode behaviour of the Pt100 circuit, base component failure mode statistics are added to determine the probability of symptoms of failure. The DOD electronic reliability of components document MIL-HDBK-217F [26] gives formulae for calculating the $failures/10^6$ in hours for a wide range of generic components. These figures are based on components from the 1980's and MIL-HDBK-217F can give conservative reliability figures when applied to modern components. Using the MIL-HDBK-217F specifications for resistor and thermistor failure statistics, the reliability for the Pt100 example (see section 5.6) is calculated below.

Resistor FIT Calculations. The formula given in MIL-HDBK-217F[26][9.2] for a generic fixed film non-power resistor is reproduced in equation A.1. The meanings and values assigned to its co-efficients are described in table A.21.

$$resistor\lambda_p = \lambda_b \pi_R \pi_Q \pi_E \tag{A.1}$$

Table A.21: Fixed film resistor Failure In Time (FIT) assessment.

Parameter	*Value*	*Comments*
λ_b	0.00092	stress/temp base failure rate 60^o C
π_R	1.0	Resistance range $< 0.1M\Omega$
π_Q	15.0	Non-Mil spec component
π_E	1.0	benign ground environment

Applying equation A.1 with the parameters from table A.21 give the following failures in 10^6 hours:

$$0.00092 \times 1.0 \times 15.0 \times 1.0 = 0.0138\ failures/10^6 Hours \tag{A.2}$$

While MIL-HDBK-217F gives MTTF for a wide range of common components, it does not specify how the components will fail (in this case OPEN or SHORT). Some standards, notably EN298 only consider most types of resistor as failing in OPEN mode. This example compromises and uses a 9:1 OPEN:SHORT ratio for resistor failure. Thus for this example resistors are expected to fail OPEN in 90% of cases and SHORTED in the other 10%. A standard fixed film resistor, for use in a benign environment, non military specification at temperatures up to 60^oC is given a probability of 13.8 failures per billion (10^9) hours of operation (see equation A.2). In EN61508 terminology, this figure is referred to as a Failure in Time (FIT)[1]. The formula given for a thermistor in MIL-HDBK-217F[26][9.8] is reproduced in equation A.3. The variable meanings and values are described in table A.22.

$$resistor\lambda_p = \lambda_b \pi_Q \pi_E \tag{A.3}$$

[1]FIT values are measured as the number of failures per Billion (10^9) hours of operation, (roughly 114,000 years). The smaller the FIT number the more reliable the component.

Table A.22: Bead type Thermistor Failure in time assessment

Parameter	*Value*	*Comments*
λ_b	0.021	stress/temp base failure rate bead thermistor
π_Q	15.0	Non-Mil spec component
π_E	1.0	benign ground environment

$$0.021 \times 1.0 \times 15.0 \times 1.0 = 0.315\ failures/10^6 Hours \tag{A.4}$$

Thus thermistor, bead type, 'non military spec' is given a FIT of 315.0. Using the above; table A.23 is presented which lists the FIT values for all single failure modes.

Table A.23: Pt100 FMEA Single Fault Statistics

Test Case	**Result sense +**	**Result sense -**	**MTTF per 10^9 hours of operation**
TC:1 R_1 SHORT	High Fault	-	1.38
TC:2 R_1 OPEN	Low Fault	Low Fault	12.42
TC:3 R_3 SHORT	Low Fault	High Fault	31.5
TC:4 R_3 OPEN	High Fault	Low Fault	283.5
TC:5 R_2 SHORT	-	Low Fault	1.38
TC:6 R_2 OPEN	High Fault	High Fault	12.42
TC:6 R_2 OPEN	High Fault	High Fault	12.42

The FIT for the circuit as a whole is the sum of MTTF values for all the test cases. The Pt100 circuit here has a FIT of 342.6. This is an MTTF of about ≈ 360 years per circuit. A probabilistic tree can now be drawn, see figure A.1, with a FIT value for the overall Pt100 circuit and FIT values for all its component fault modes. From this it can be seen that the most likely fault is the thermistor going OPEN. This circuit is around 10 times more likely to fail in this way than in any other. If a more reliable temperature sensor was required, this would probably be the fault mode scrutinised first.

Pt100 Example: Double Failures and statistical data. Because double simultaneous failure analysis can be performed under FMMD failure rate statistics for double failures can also be determined. Considering the failure modes to be statistically independent the FIT values for all the combinations of failures in the electronic examples from chapter 5 in table 5.12 can be calculated. The failure mode of most concern, the undetectable **FLOATING** condition, requires that resistors R_1 and R_2 both fail. Multiplying the MTTF probabilities for these types of resistor failing gives the MTTF for both failing simultaneously. The FIT value of 12.42 corresponds to 12.42×10^{-9} failures per hour. Squaring this gives 154.3×10^{-18}. This is an astronomically small MTTF, and so small that it would probably fall below a threshold to sensibly consider. However, it is interesting from a failure analysis perspective, because an undetectable fault (at least at this level in the FMMD hierarchy)

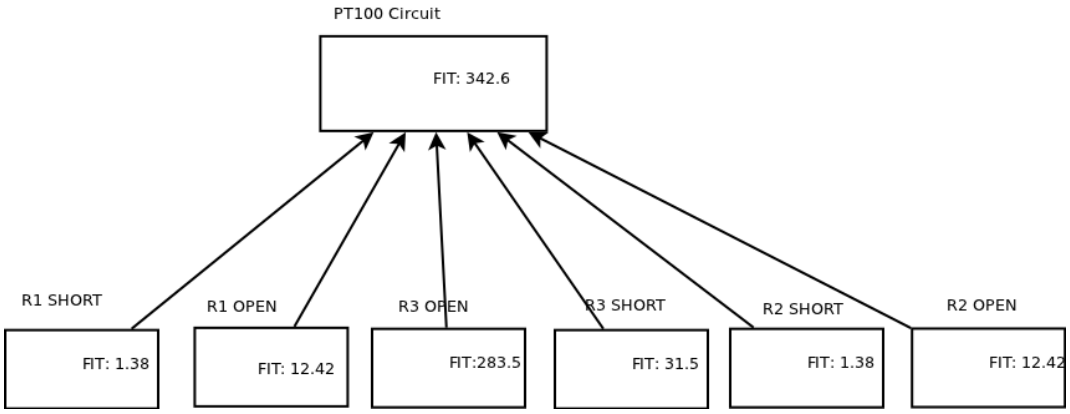


Figure A.1: Probablistic Fault Tree : Pt100 Single Faults

has been revealed. This means that should it be required to cope with this fault, a new way of detecting this condition must be engineered, perhaps in higher levels of the system/FMMD hierarchy.

MTTF statistics and FMMD hierarchies. In a large FMMD model, system/top level failures can be traced down to base component failure modes. To determine the MTTF probability for a system level failure, the MTTF statistics are added for all its possible causes. Thus even for large FMMD models accurate statistics for electronic sourced failures can be calculated.

A.3.9 Gnuplot script for hypothetical XFMEA FMMD reasoning distance comparison

```
##############################################################################################
# GNUPLOT SCRIPT to plot XFMEA FMMD reasoning distance
# comparisons.
#
#
# Always define floating point explicitly at initialisation, as in 'C',
# because otherwise gnuplot treats these as integers.
#
# number of failure modes per component
fm = 3.0
#
# number of components in each functional group
k = 3.0
#
# place the functional group size and failure mode per components
# size into a string to use as the graph title
#
tt = sprintf("reasoning distance comparison for |fg| = %d and |fm| = %d", k, fm)
set title tt
#
a = 0.0
b = 0.0
#
# formula for reasoning distance in one level of FMMD
# hierarchy (as given by ll)
#
fmmd(ll)=k**ll * k * fm * (k - 1)
#
# set up iterative sum in gnuplot syntax
# to iterate over FMMD levels
#
sum(a,b) = (a > b) ? 0 : fmmd(a) + sum(a+1, b)
sig_fx(c) = sum(a,c)
#
# reasoning distance for exhaustive case in FMEA
# where ll is the hierarchy level
```

```
xfmea(ll) = k**(ll+1) * ( k**(ll+1) -1 ) * fm
#
#
set xrange [0:1000]
set xlabel "Component count"
set ylabel "reasoning distance"
set logscale y
#
set terminal png
set output 'xfmea_fmmd_comp.png'
plot sig_fx(x**(1/k)), xfmea(x**(1/k))
#!sleep 20
##############################################################################################
```

Appendix B

Algorithmic Description of FMMD

This appendix formalises the process for performing one stage of the FMMD process from a given functional grouping to creating a derived component. The FMMD process is then examined in greater detail and described with set theory in an algorithmic context. The intention for defining FMMD in algorithmic and set theoretic terms is to provide a solid specification for the process that could guide a software implementation.

B.1 Overview of the FMMD analysis process

The FMMD process is described in chapter 4. To re-cap, FMMD has four main stages:

- collection of components to form functional groupings,
- applying FMEA to the functional groupings,
- collecting common symptoms from the FMEA results,
- creating a derived component modelling the failure mode behaviour of the functional grouping.

This process allows us to modularise and thus simplify FMEA analysis of systems.

FMEA applied to the functional grouping: choosing test cases. As a functional grouping is a collection of components, the failure modes to consider are the failure modes of its components. Single failure modes or combinations are used to create failure mode analysis scenarios, or test cases. The component failure modes in each test case are examined with respect to their effect on the functional grouping (in contrast, traditional FMEA examines each component failure modes effect on a whole system). The aim of FMMD analysis is to find out how the functional grouping fails given each test case.

Environmental Conditions or Operational States. Each test case must also be considered for all operational states and environmental conditions to which it may be exposed. In this way, all possible failure mode behaviour, due to all the conditions that can be applied for all the test cases will be examined.

Symptom Identification. When all test cases have been analysed, a set of FMEA results exists for the given functional grouping. These results can be viewed as symptoms of failure of the functional grouping.

Collection of Symptoms. Common symptoms of failure of the functional grouping are collected. A new component is created to represent the functional grouping and the aggregated symptoms considered as its failure modes. This new component is called a 'derived component'. Because the FMMD process is bottom up, it can be ensured that all failure modes from the components in a functional grouping have been considered. Were failure modes to be missed, the resulting failure mode model would be incomplete. It is possible here for an automated system to flag un-handled failure modes.

B.2 Expanding on a single stage of the FMMD process

The expanded FMMD process can now be described in five steps:

1. Choose a set of components to form a functional grouping, and collect the failure modes of each component in the functional grouping into a flat set.
2. Choose all single instances (and optional selected combinations[1]) of the failure modes to form 'test cases'.
3. Using the 'test cases' as scenarios to examine the effects of component failures, the failure mode behaviour of the functional grouping is determined. This is a human process, applying FMEA for each test case. Where specific environmental conditions, or operational states are germane to the functional grouping, these must also be examined for each test case.
4. Collect common symptoms by determining which test cases produce the same fault symptoms *from the perspective of the functional grouping.*
5. The common symptoms are now the fault mode behaviour of the functional grouping. i.e. given the functional grouping as a 'black box' the symptoms are the ways in which it can fail. A new 'derived component' can now be created where each common symptom, or lone symptom, is a failure mode of this new component.

B.2.1 Single stage of FMMD described as a 'symptom abstraction process'

A single stage of FMMD analysis can be described as symptom abstraction. This is because the failure modes of a derived component are at a higher—or meta/more abstract level—than the component failure modes it was derived from. The FMMD process is now described introducing set theoretical definitions that will be used in the algorithmic description of FMMD. Let the set of all possible components be $\mathcal{C}$ and let the set of all possible failure modes be $\mathcal{F}$ and $\mathcal{P}$ the powerset. The function fm is defined which returns the failure modes for a given component (see section 7.1.1):

$$fm : \mathcal{C} \rightarrow \mathcal{P}\mathcal{F}. \tag{B.1}$$

The notation for the function fm is overloaded and defined for the set of components within a functional grouping FG (i.e. where $FG \subset \mathcal{C}$) thus:

[1] Some specific combinations of failure modes might be included. For instance where a very reliable part is duplicated but very critical, like the 4 engines on a 747 aircraft.

$$fm : FG \rightarrow \mathcal{F}. \tag{B.2}$$

Where $\mathcal{FG}$ is the set of all sets of functional groupings, and $\mathcal{DC}$ is the set of all derived components, the symptom abstraction process is defined thus:

$$D : \mathcal{FG} \rightarrow \mathcal{DC}.$$

B.3 Algorithmic Description of Symptom Abstraction

The algorithm for *symptom abstraction* is described in this section using set theory and procedural descriptions. By defining the process and describing it using set theory, constraints and verification checks can be stated formally. The algorithm, represented by the symbol 'D', is described using five algorithmic steps below. The

Algorithm 1 Derive new 'Component' DC from a given functional grouping FG: $D(FG)$

1: F = fm (FG) ▷ collect all component failure modes
2: TC = dtc (F) ▷ determine all test cases giving a set of test cases TC
3: R = atc (TC) ▷ analyse the test cases giving a set of FMEA results R
4: SP = fcs (R) ▷ find common symptoms, aggregate results from R giving a set of symptoms SP
5: DC = cdc (SP) ▷ create a derived component
6: **return** DC

symptom abstraction process allows us to take a functional grouping of components, analyse the failure mode behaviour and create a new entity, a derived component that has its own set of failure modes. The checks and constraints applied in the algorithm ensure that all component failure modes are covered. This process provides the analysis 'step' to building a hierarchical failure mode model from the bottom-up.

B.3.1 Determine Failure Modes to Examine

The first stage is to find the failure modes to consider for analysis, using the earlier definition of the function fm applied to all components within the given functional grouping.

B.3.2 Determine Test Cases

From the failure modes associated with the functional group, test cases must next be determined. The test cases are collections of failure modes. These can be formed from single failure modes or failure modes in combination. Let $\mathcal{TC}$ be the set of all test cases, $\mathcal{F}$ be the set of all failure modes. The function dtc is defined thus:

$$dtc : \mathcal{F} \rightarrow \mathcal{TC},$$

given by

$$dtc(F) = TC.$$

In algorithm 2, the function **chosen** means that the failure modes for a particular test case have been chosen by a human operator and are additional to those chosen by the automated process (i.e they are special test cases involving multiple failure modes). The function **isunitarystate** takes as its argument a test case and returns true if no pairs of that test case's failure modes are sourced from the same component (see section 7.3). In other words, this function tests that failure modes in test cases only use failure modes that are mutually exclusive within components. The *dtc* function enforces completeness in the model by ensuring that all failure modes of the components in the functional grouping are included in at least one test case.

Algorithm 2 Determine Test Cases: dtc: (F)

Require: F is a non empty flat set of failure modes

All test cases are chosen by the investigating engineer(s). Typically all single component failures are investigated with some specially selected combination faults

Let TC be a set of test cases ▷ this set is used to collect the test cases

Let tc_j be a set of component failure modes where j is an index of J ▷ Each set tc_j is a 'test case' and $TC = \bigcup_{j \in J} \{tc_j\}$ where $J \subset \mathbb{N}_+$

$TC := \emptyset$ ▷ Initialise set of test cases

$j := 1$ ▷ Initialise index of test cases

for all $f \in F$ **do** ▷ Assign one test case per single fault mode

  $tc_j := f$

  $TC := TC \cup tc_j$ ▷ place this test case into the set TC

  $j := j + 1$

end for

if DoubleFaultChecking = TRUE **then** ▷ Assign one test case per valid double fault mode

  for all $f_1, f_2 \in F$ **do**

    $ptc := \{f_1, f_2\}$ ▷ Make ptc a provisional test case

    if $isunitarystate(ptc)$ **then**

      $j := j + 1$

      $tc_j := ptc$

      $TC := TC \cup tc_j$ ▷ place this test case into the set TC

    end if

  end for

end if

for all $ptc \in \mathcal{P}(F)$ **do** ▷ for all subsets of F

  if $chosen(ptc) \wedge ptc \notin TC \wedge isunitarystate(ptc)$ **then**

    $j := j + 1$

    $tc_j := ptc$

    $TC := TC \cup tc_j$

  end if

end for

Ensure: $\forall j_1, j_2 \in J \; such \; that \; j_1 \neq j_2 \big(tc_{j_1} \neq tc_{j_2}\big)$ ▷ Ensure test cases are distinct

Ensure: $\forall tc \in TC \big(tc \in \mathcal{P}(F)\big)$ ▷ Ensure each test case is a subset of F

28: let f represent a component failure mode

Ensure: $\forall f \; such \; that \; (f \in F)) \wedge (f \in \bigcup TC)$ ▷ Check that at each single failure mode in the functional grouping is included as a test case.

29: let $f1, f2$ represent component failure modes, and c any component in the functional group

Ensure: $\forall f1, f2 \;\; where \;\; (f1 \wedge f2) \notin (\forall c) \;\; such \;\; that \;\; (f1, f2 \in F)) \wedge (\{f1, f2\} \in \bigcup TC) \wedge (DoubleFaultChecking = TRUE)$ ▷ If $DoubleFaultChecking$ is required, check that all possible double failure modes in the functional grouping (see section 7.3) are included as a test cases.

30: **return** TC

Algorithm 2 has taken the set of failure modes $F = fm(FG)$ and returned a set of test cases TC. The next stage is to analyse the effect of each test case on the functional grouping. Double failure mode checking has been included in this algorithm specifically because of the double failure mode implications of European standard EN298 [14].

B.3.3 Analyse Test Cases

The test cases are now analysed for their impact on the behaviour of the functional group. Let $\mathcal{R}$ be the set of all test case analysis results, indexed by j (the same index used to identify the test cases tc_j). The function atc is defined thus:

$$atc : \mathcal{TC} \rightarrow \mathcal{R},$$

given by

$$atc(TC) = R.$$

Algorithm 3 Analyse Test Cases: atc(TC)

1: let r be a 'test case result'

2: define the function $Analyse : tc \rightarrow r$ ▷ This analysis is a human activity, FMEA, i.e. examining the component failure modes in the test case and determining how the functional group will fail under those conditions.

3: R is a set of test case results $r_j \in R$ where the index j corresponds to $tc_j \in TC$

4: **for all** $tc_j \in TC$ **do**

5: **for all** Environmental and Specific Applied States **do**

6: $rc_j = Analyse(tc_j)$ ▷ this is FMEA applied in the localised context of the functional grouping

7: $R := R \cup rc_j$ ▷ Add rc_j to the set R

8: **end for**

9: **end for**

10: **return** R

Algorithm 3 has built the set R, the sub-system/functional grouping results for each test case.

The analysis is primarily a human activity. Calculations or simulations are performed to determine how the failure modes in each test case will affect the functional group. Ideally field data and/or formal physical testing should be used in addition to static failure mode reasoning where possible. When all the test cases have been analysed, a 'result' will exist for each 'test case'. Each result will be described from the perspective of the functional grouping, not the members of it i.e. the components.

A set of results corresponding to our test cases is now available. These share a common index value (j in the algorithm description). These results are the failure modes of the functional grouping.

B.3.4 Find Common Symptoms

This stage collects results into 'symptom' sets. Each result from the preceding stage is examined and collected into common symptom sets. That is to say, results which have a common failure symptom from the perspective of the operation of the functional grouping. Let set $\mathcal{SP}$ be the set of all symptoms, and $\mathcal{R}$ be the set of all test case results. The function fcs is defined thus:

$$fcs : \mathcal{R} \to \mathcal{SP},$$

given by

$$fcs(R) = SP.$$

This raises the failure mode abstraction level, α (see section 4.5.2). The failures have now been considered not from the component level, but from the sub-system or functional group level. A set SP, the symptoms of failure, is obtained. Ensuring that no result is linked to more than one symptom enforces the 'unitary state failure mode constraint' for derived components (see section 7.3).

B.3.5 Create Derived Component

This final stage is the creation of the derived component. This derived component may now be used to build new functional groupings at higher levels of fault abstraction. Let DC be a derived component with its own set of failure modes. The function cdc is defined thus:

$$cdc : \mathcal{SP} \to \mathcal{DC},$$

given by

$$cdc(SP) = DC.$$

The new component will have a set of failure modes that correspond to the common symptoms collected from the FG. A derived component DC, which has its own set of failure modes has been created. This can now be used in with other components (or derived components) to form functional groups at higher levels of failure mode abstraction.

Enumerating abstraction levels. As described in section 4.5.2 the attribute of abstraction level α can be assigned to components, where α is a natural number, ($\alpha \in \mathbb{N}_0$). For a base component, let the abstraction level be zero. The symptom abstraction process gives a *derived component*, this derived component will have an α

value one higher than the highest α value of any of the components in the functional grouping used to derive it. Thus a derived component sourced from base components will have an α value of 1. The attribute α can be used to track the level of fault abstraction of components in an FMMD hierarchy. Because base and derived components are collected to form functional groupings, a hierarchy is naturally formed with the abstraction levels increasing with each tier.

B.3.6 Hierarchical Simplification

Since symptom abstraction aggregates fault modes, the number of faults to handle should decrease as the hierarchy progresses upwards. At the highest levels the number of faults is significantly less than the sum of its component failure modes. To go back to the sound system analogy (see section 4.3.1), it may have only four faults at its highest or system level,

$$SoundSystemFaults = \{TUNER\_FAULT, CD\_FAULT, SOUND\_OUT\_FAULT, IPOD\_FAULT\}$$

The number of causes for any of these faults is very large. It does not matter to the user, which combination of component failure modes caused the fault. But as the hierarchy goes up in abstraction level, the number of failure modes goes down for each level: as is found in practise in real world systems.

B.3.7 Traceable Fault Modes

Since the fault modes are determined from the bottom-up, the causes for all high level faults naturally form trees. That is to say from the bottom-up causes become symptoms, which in the next level become causes as the tree is traversed upwards. This is demonstrated in the examples chapter 5 where DAGS are drawn linking failure mode causes and symptoms in FMMD analysis hierarchies. These trees can be also traversed to produce minimal cut sets[70] or entire FTA trees[84], and by analysing the statistical likelihood of the component failures, the Mean Time to Failure (MTTF) and Failure in Time(FIT)[95] levels can be automatically calculated.

Glossary

base component Any bought in component, or lowest level module/or part, 8, 9, 29, 41, 42, 44

derived component A theoretical component, derived from a collection of components (which may be derived components themselves), 37, 38, 40, 43–45

functional grouping A collection of sub-systems and/or components that interact to perform a specific function, 37, 40, 42–45

ADC Analogue to digital converter, a digital device to read voltages into a computer/micro-controller, 16, 75, 99, 151, 152

backward search Failure analysis where the start points are system level failure/symptom and the results are lower level putative causes. Sometimes termed 'top down', 20

contract programming A software discipline whereby each function is assigned strict pre and post conditions which define a 'contract' formalising the function's behaviour, 4, 90, 91, 96, 97, 105, 109

DFMEA Design FMEA. FMEA applied in design stages of a product. Can be used as a discussion/brain storming method to reveal safety weakness and improve built in safety, 26

failure mode A component or sub-system may fail in a number of ways, and each of these is a failure mode of that particular component type, 9–12, 29, 39, 40, 44, 81, 124, 160, 168

failure rate The number of failures expected over a given time interval, 10, 131, 160–162

FIT Failure in Time (FIT). The number of times a particular failure is expected to occur within a 10^9 hour time period, 10, 20, 131, 160–162

FMEA Failure Mode and Effects analysis (FMEA) is a process where each base component failure mode in a given system is analysed to determine system level failures/symptoms, 7, 8, 16, 29, 45, 48, 129, 130

FMECA Failure Mode Effects and Criticality Analysis (FMECA). An extended FMEA technique, based on Bayesian statistics, which is used to order the severity or criticality of top level events/symptoms, 22, 23, 48, 136

FMEDA Failure Mode Effects and Diagnostic Analysis (FMEDA). An extended FMEA technique which provides for diagnostic mitigation and has a final statistical safety level as a result, 23–25, 48, 135, 136

FMMD Failure Mode Modular De-Composition (FMMD). A bottom-up methodology for incrementally building failure mode models, using a procedure taking functional groups of components and creating derived components representing them, and in turn using the derived components to create higher level functional groups, and so on, building a hierarchical failure mode model, 37, 40, 44, 48, 112, 130, 135, 165–167, 170–172

forward search Failure analysis where the start points are base component failure modes and the result is system level failure/symptom. Sometimes termed 'bottom up', 20

FTA Fault Tree Analysis (FTA). A top down failure analysis technique which starts with undesirable top level events, and using symbols from digital logic/electronics builds a tree, working downwards to putative causes, 44, 45, 48, 132, 133

HFMEA Hardware FMEA. FMEA applied to hardware i.e. mechanical or electrical equipment, 98, 129

inhibit A guard on a process such that if a condition is not met, the process may not continue, 132, 133

mutually exclusive Mutual exclusivity applied to component failure modes means that for each component it is ensured that only one of its failure modes may be active at any given time, 17, 44, 48, 119–121, 129, 168

observability If a failure mode cannot be detected it is termed unobservable or undetectable, 19, 109

Op-Amp An Operational Amplifier is a differential input high gain voltage amplifier typically implemented in an integrated circuit and is a commonly used element in analogue circuit design, 12, 13, 38, 39, 41, 46, 51, 53, 56, 58, 60, 64, 67, 69, 70, 76, 77

reasoning distance A reasoning distance is the number of components examined against failure scenarios, used to map a failure causes to potential outcomes for a given system, 20, 27, 111, 113, 115, 129, 130

SFMEA Software FMEA (SFMEA). FMEA techniques applied to software, 32, 135

signal path The components (software or hardware) and connections from which a particular signal or value is derived, 16–18, 31, 33

smart instrument A smart instrument is one that uses software in conjunction with its sensing electronics, rather than analogue electronics only [72], 33

State explosion State Explosion is the effect where very large numbers of combinations of conditions, or combinations of conditions and entities have to be processed. The number to be processed can quickly become too large for practical consideration, and when this happens 'state explosion' can be said to have occurred, 2, 3, 20–22, 29, 31, 35, 37, 48, 121, 129

sub-system A part of a system, sub-systems may contain sub-systems and so-on, 42, 44, 46

symptom A failure mode of a functional grouping, caused by a combination of its component failure modes, 37, 40, 45

Symptom Abstraction By applying failure mode analysis to a module the symptoms of failure for it are determined given the failure modes of its components, its topology and its expected behaviour, 45, 166, 167

system A product designed to work as a coherent entity, 42

XFMEA Exhaustive FMEA (XFMEA). Applying FMEA exhaustively means checking each failure mode for effects on all components in a given system, 21, 111, 116

Bibliography

[1] Sanjeev Kumar Appicharla. Analysis and modelling of space shuttle challenger accident using management oversight and risk tree(mort). *7th IET System Safety Conference 2012*, 2012.

[2] Volker Bachmann and Richard Messnarz. Improving safety and availability of complex systems by using an integrated design approach in development. *Journal of Software: Evolution and Process*, 2012.

[3] A. Balfour and D.H. Marwick. *Programming in Standard Fortran 77 ISBN 0-435-77486-7.* Heinemann Educational Books, 1979.

[4] Emery D Berger. Software needs seatbelts and airbags. In *Communications of the ACM*, pages 48 – 56, september 2012.

[5] P. Bishop. Sils and software. *Safety-Critical Systems Club Newsletter, 14(2)*, 14(2), 2005.

[6] P. Bishop, R. Bloomfield, Bev Littlewood, A. Povyakalo, and D. Wright. Toward a formalism for conservative claims about the dependability of software-based systems. *Software Engineering, IEEE Transactions on*, 37(5):708–717, 2011.

[7] Peter Bishop, Robin Bloomfield, Sofia Guerra, and Kostas Tourlas. Justification of smart sensors for nuclear applications. 3688:194–207, 2005.

[8] Peter Bishop and Lukasz Cyra. Overcoming non-determinism in testing smart devices: a case study. In *Proceedings of the 29th international conference on Computer safety, reliability, and security*, SAFECOMP'10, pages 237–250, Berlin, Heidelberg, 2010. Springer-Verlag.

[9] Peter Bishop and Lukasz Cyra. Testing smart devices: A case study. In Erwin Schoitsch, editor, *Computer Safety, Reliability, and Security*, volume 6351 of *Lecture Notes in Computer Science*, pages 237–250. Springer Berlin Heidelberg, 2010.

[10] Robin Bloomfield and Peter Bishop. Safety and assurance cases: Past, present and possible future an adelard perspective. In Chris Dale and Tom Anderson, editors, *Making Systems Safer*, pages 51–67. Springer London, 2010.

[11] Bosch. Can specification 2.0. *Bosch Technical Standard*, 1991.

[12] R Hanbury Brown. *Boffin: A Personal Story of the Early Days of Radar, Radio Astronomy and Quantum Optics.* Adam Hilger: ISBN 0-85274-317-3, 1991.

[13] BSI. Iso 9001 quality. *British Standards Institute*, 2012.

[14] EN Standard BSI. En298:2003 gas burner controllers with forced draft. British standards Institution http://www.bsigroup.com/, 2003.

[15] EN Standard BSI. En230:2005 automatic burner control systems for oil burners. British standards Institution http://www.bsigroup.com/, 2005.

[16] IEC Standard BSI. Iec 60812:1985 analysis techniques for system reliability - procedure for failure mode and effects analysis (fmea). British standards Institution http://www.bsigroup.com/, 1985.

[17] A. Chamseddine and H. Noura. Sensor network design for complex systems. *International Journal of Adaptive Control and Signal Processing*, 26(6):496–515, 2012.

[18] Ying Chen, Cui Ye, Bingdong Liu, and Rui Kang. Status of fmeca research and engineering application. In *Prognostics and System Health Management (PHM), 2012 IEEE Conference on*, pages 1 –9, may 2012.

[19] R Clark, A Fish, C Garrett, and J Howse. Developing a rigorous bottom-up modular static failure modelling methodology. *6th IET International Conference on System Safety, 2011*, 2011.

[20] R. Clark, A. Fish, C. Garrett, and J. Howse. Applying failure mode modular de-composition (fmmd) across the software/hardware interface. In *System Safety, incorporating the Cyber Security Conference 2012, 7th IET International Conference on*, pages 1–6, 2012.

[21] Robin Clark. Fast zone discrimination, an algorithm for quickly determining available zones in euler diagrams using the java area and shape classes. *Visual Languages and Computing 2007, Idaho, USA*, 2007.

[22] R.P. Clark. Failure mode modular de-composition using spider diagrams. *Electronic Notes in Theoretical Computer Science*, 134:19 – 31, 2005. Proceedings of the First International Workshop on Euler Diagrams (Euler 2004).

[23] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19:297–301, 1965.

[24] Reliability Analysis Center US DOC. Failure mode/mechanisms distributions 1991. *United States Department of Commerce: F30602-91-C-0002*, 1991.

[25] United States DOD. Mil-std-1629a: Procedure for performing a failure mode, effects and criticality analysis. *United States Department of Defence*, 1980.

[26] United States DOD. *MIL-1991: Reliability Prediction of Electronic Equipment.* United States Department of Defence, 1991.

[27] Clifton Ericsson. Fault tree analysis a history. *Proceedings of the 17th international safety conference*, 1999.

[28] Barbara J. Czerny et all. Delphi Corporation. Effective application of software safety techniques for automotive embedded control systems. *Occupant Safety, Safety-Critical Systems, and Crashworthiness (SP-1923)*, 2005.

[29] Eurotherm. Thermocouple emf tables and platinum 100 resistance thermometer tables. Technical report, Eurotherm Ltd, UK, 1973.

[30] Federal Aviation Administration FAA. *System Safety Handbook.* http://www.faa.gov/library/manuals/aviation/risk_management/ss_handbook/, 2008.

[31] William Fairbairn. *Useful Information for Engineers being a series of lectures delivered to the working engineers of Yorkshire and Lancashire : together with a series of appendices, containing the results of experimental inquiries into the strength of materials, the causes of boiler explosions.* Longman, 1864.

[32] Andrew Fish and Jean Flower. Investigating reasoning with constraint diagrams. *Electronic Notes in Theoretical Computer Science*, 127(4):53 – 69, 2005. Proceedings of the Workshop on Visual Languages and Formal Methods (VLFM 2004) Visual Languages and Formal Methods 2004.

[33] Jean Flower, Judith Masthoff, and Gem Stapleton. Generating readable proofs: A heuristic approach to theorem proving with spider diagrams. In AlanF. Blackwell, Kim Marriott, and Atsushi Shimojima, editors, *Diagrammatic Representation and Inference*, volume 2980 of *Lecture Notes in Computer Science*, pages 166–181. Springer Berlin Heidelberg, 2004.

[34] Stark Fortescue, Swinerd. *Spacecraft Systems Engineering ISBN:978-0-470-75012-4.* Wiley, 2011.

[35] Workman Franklin, Powell. *Digital Control of Dynamic Systems.* Addisson Wesley ISBN 0-201-33153-5, 1997.

[36] Chris Garrett. *Functional diagnosis strategies for analog systems using heuristic programming techniques.* PhD thesis, Brighton University, School of Electrical Engineering, 1989.

[37] M.M. Ghahroodi, M. Zwolinski, and E. Ozer. Radiation hardening by design: A novel gate level approach. In *Adaptive Hardware and Systems (AHS), 2011 NASA/ESA Conference on*, pages 74 –79, june 2011.

[38] Alan Gibbons. *Algorithmic Graph Theory ISBN:978-0521288811.* Cambridge University PressCambridge University Press, 1985.

[39] John C. Grebe Dr. William M. Goble. Fmeda accurate product failure metrics. *EXIDA publication. www.exida.com/articles/FMEDA%20Development.pdf*, 2007.

[40] Peter L. Goddard. Validating the safety of embedded real-time control systems using fmea. *Reliability and Maintainability Symposium (RAMS), 1993 Proceedings - Annual*, 1993.

[41] Lars Grunske, R. Colvin, and K. Winter. Probabilistic model-checking support for fmea. In *Quantitative Evaluation of Systems, 2007. QEST 2007. Fourth International Conference on the*, pages 119–128, Sept.

[42] S. Hartkopf. From a single discipline risk management approach to an interdisciplinary one: adaptation of fmea to software needs. In *Software Technology and Engineering Practice, 2003. Eleventh Annual International Workshop on*, pages 204 – 213, sept. 2003.

[43] N. Heraud, E. Hassan Mazzour, and C. Alberti. Observability and sensor positioning for processes described by linear-bilinear equations. In *Control Applications, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 2, pages 1276 –1280 vol.2, sep 1998.

[44] Baiqiao HUANG. Software fmea approach based on failure modes database. *Reliability, Maintainability and Safety, 2009. ICRMS 2009. 8th International Conference*, 2009.

[45] D. C. Ince. *In Introduction to discrete Mathematics, Formal System specification and Z.* Oxford University Press, 1992.

[46] Philipp K. Janert. *Gnuplot in Action: Understanding Data with Graphs.* Manning Publications Co., Greenwich, CT, USA, 2009.

[47] Gem Stapleton John Howse and John Taylor. Spider diagrams. *London Mathematical Society*, 8(0):145–194, December 2005.

[48] Peter Kafka. The automotive standard iso 26262, the innovative driver for enhanced safety assessment; technology for motor cars. *Procedia Engineering*, 45(0):2 – 10, 2012. 2012 International Symposium on Safety Science and Technology.

[49] Brian W. Kernighan and Dennis Ritchie. *The C Programming Language, Second Edition.* Prentice-Hall, 1988.

[50] Walt Kestler. *Mixed Signal and DSP Design Techniques ISBN 0750676116.* Newnes/Analog Devices, 2003.

[51] F Langford-Smith. *Radio designers Handbook: Fourth Edition.* ILIFFE, 1953.

[52] Nancy Leveson. High-pressure steam engines and computer software. Technical report, University of Washington, 1994.

[53] Nancy Leveson. *Safeware: System safety and Computers ISBN: 0-201-11972-2.* Addison-Wesley, 2005.

[54] I. Lopatkin, A. Iliasov, A. Romanovsky, Y. Prokhorova, and E. Troubitsyna. Patterns for representing fmea in formal specification of control systems. In *High-Assurance Systems Engineering (HASE), 2011 IEEE 13th International Symposium on*, pages 146–151, 2011.

[55] Bernard Lovell. *Echoes of War: The story of H2S RADAR.* Adam Hilger: ISBN 0750301309, 1991.

[56] Robyn R. Lutz and Robert M. Woodhouse. Requirements analysis using forward and backward search. *Ann. Softw. Eng.*, 3:459–475, January 1997.

[57] Loius. Lyons. In *Contemporary Physics: Bayes and Frequentism, A paticle physicists perspective*, volume 2, Feb 2013.

[58] Leo M Maikowski. *Tolereranced Multiple Fault Diagnosis of Analog Circuits.* PhD thesis, Brighton University, School of Electrical Engineering, 1995.

[59] Ron Mancini. Design of op-amp sine wave oscillators. *Analog Applications Journal: Texas Instruments: August*, 2000.

[60] Jakob Maus and Bernd Neumann. Diagnosis by algebraic modelling and fault tree induction. *Sixth International Workshop on principles of diagnosis*, 1995. Working papers of DX-95: Sixth International Workshop on principles of diagnosis.

[61] Gavin McCall. *MISRA:C:2004 Guidelines for the use of the C language in critical systems ISBN 978-0-9524156-4-0.* Hobbs, 2004.

[62] Robin McDermot. *The Basics of FMEA ISBN: 0-527-76320-9.* Productivity, 1996.

[63] Microchip. *PIC18F2523 Datasheet.* http://ww1.microchip.com/downloads/en/DeviceDoc/39755c.pdf, 2009.

[64] Microchip. *Datasheet Erratta: PIC18F66K80 Family Silicon Errata and Data Sheet Clarification DS805119D.* http://ww1.microchip.com/downloads/en/DeviceDoc/80519d.pdf, 2011.

[65] Microchip. Microchip: Reliability data search engine: Annual product reliability reports. Microchip Inc. http://www.microchip.com/reliabilityreport/Default.aspx, 2013.

[66] ST Microelecronics. *Datasheet: Low-Power dual operation amplifiers LM158,LM258,LM358: Doc ID 2163 Rev 10.* http://www.st.com/, 2012.

[67] R. Mitchel. *Design By Contract by Example ISBN-13: 978-0201634600.* Adisson-Wesley, 2002.

[68] MODBUS.ORG. Modbus over serial line: Specification and implementation guide v1.0. Technical report, MODBUS.ORG, 2002.

[69] C. Muller, M. Valle, E. Armengaud, and A. Tengg. A generic framework for failure modes and effects analysis of automotive networks. In *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*, pages 293 –298, july 2011.

[70] NASA. Fault tree handbook with aerospace applications. *NASA Handbook*, 2002.

[71] Shawulu Hunira Nggada. Software failure analysis at architecture level using fmea. *International Journal of Software and its applications Vol 6. 1*, 2012.

[72] T.S. Nobes. Functional safety of smart instruments - a user perspective. In *Is Your Product Safe? - IEE Seminar on (Ref. No. 2004/10724)*, pages 67–87, Sept.

[73] N. Ozarin. A process for failure modes and effects analysis of computer software. *Reliability and Maintainability Symposium, 2003. Annual*, 2003.

[74] N.W. Ozarin. Applying software failure modes and effects analysis to interfaces. *Reliability and Maintainability Symposium, 2009. RAMS 2009. Annual*, pages 533 – 538, jan. 2009.

[75] C. Pace, S. Libertino, I. Crupi, A. Marino, S. Lombardo, E.D. Sala, G. Capuano, M. Lisiansky, and Y. Roizin. Compact instrumentation for radiation tolerance test of flash memories in space environment. In *Instrumentation and Measurement Technology Conference (I2MTC), 2010 IEEE*, pages 652 –655, may 2010.

[76] Y. Papadopoulos, D. Parker, and C. Grante. Automating the failure modes and effects analysis of safety critical systems. In *High Assurance Systems Engineering, 2004. Proceedings. Eighth IEEE International Symposium on*, pages 310 – 311, march 2004.

[77] David Lorge Parnas, GJK Asmis, and Jan Madey. Assessment of safety-critical software in nuclear power plants. *Nuclear safety*, 32(2):189–198, 1991.

[78] Winfield Hill Paul Horowitz. *The Art of Electronics, 2nd Edition ISBN 0-521-37095-7.* Cambridge, 1989.

[79] Olaf Pfeiffer. *Embedded networking with CAN and CANopen.* RTC ISBN 0-929392-78-7, 2003.

[80] Chris Price. *Computer-Based Diagnostic Systems ISBN 3-540-76198-5.* Springer Practitioner series, 1999.

[81] C.J. Price. Effortless incremental design fmea. In *Reliability and Maintainability Symposium, 1996 Proceedings. International Symposium on Product Quality and Integrity., Annual*, pages 43 –47, jan 1996.

[82] C.J. Price and N.S. Taylor. Fmea for multiple failures. In *Reliability and Maintainability Symposium, 1998. Proceedings., Annual*, pages 43 –47, jan 1998.

[83] Gregory J.E. Rawlins. *Compared to What ? An introduction to the analysis of algorithms ISBN 0-7167-8243-x.* Computer Science Press, 1991.

[84] US Nuclear reg commission. Fault tree handbook. *Nuclear Safety Analysis Handbook*, 1981.

[85] Peter Rodgers. A survey of euler diagrams. *Journal of Visual Languages and Computing*, (0):–, 2013.

[86] SD Rudov-Clark and J Stecki. The language of fmea: on the effective use and reuse of fmea data. *submitted to HUMS2009, Melbourne*, pages 10–12, 2009.

[87] FAIRCHILD Semiconductor. *CD4013 Dual D-Type Flip Flop Datasheet.* http://www.fairchildsemi.com/ds/CD/CD4013BC.pdf, 2002.

[88] D. Sheridan. Simple concurrency analysis plugin for frama-c. Available from https://bitbucket.org/adelard/simple-concurrency, 2013.

[89] D. Smith. *Safety Critical Systems Handbook, 3rd Ed. ISBN 978-0-08-096781-3.* Butterworth HeinemannH, 2011.

[90] N Snooke. Model-driven automated software fmea. *Reliability and Maintainability Symposium (RAMS), 2011 Proceedings - Annual*, 2011.

[91] Neal Snooke and Chris Price. An automated software fmea. *International System Safety conference Singapore 2008*, 2008.

[92] Various Open source Project. Gnuplot: graph plotting utility. Available from http://www.gnuplot.info/, 2005.

[93] M R Spiegel. *Probability and Statistics Second edition : SHCAUM'S : ISBN 0-07-135004-7.* Oxford University Press, 1988.

[94] M R Spiegel, J Schiller, and A Srinivasan. *Probability and Statistics Crash Course : SHCAUM'S : ISBN 0-07-138341-7.* McGraw Hill, 2001.

[95] E N Standard. 61508:2002 functional safety of electrical/electronic/programmable electronic safety related systems. British standards Institution http://www.bsigroup.com/, 2002.

[96] Neil Storey. *Safety-Critical Computer Systems ISBN 0-201-42787-7.* Prentice Hall, 1996.

[97] Neil Storey. *Electronics - a systems approach (4. ed.). ISBN 978-0-13-129396-0.* Pearson / Prentice Hall, 2009.

[98] J. Stranks. *Human Factors and Behavioural Safety ISBN 978-0-7506-5510-1.* Butterworth-Heinmann, 2007.

[99] Robert D Strum and Donald E. Kirk. *First Principles of discrete Systems and Digital Signal Processing ISBN 0-201-09518-1.* Addison-Wesley, 1988.

[100] P. C. Teoh and Keith Case. An evaluation of failure modes and effects analysis generation method for conceptual design. *International Journal of Computer Integrated Manufacturing*, 18(4):279 – 293, 2005.

[101] D.R. Throop, J.T. Malin, and L.D. Fleming. Automated incremental design fmea. In *Aerospace Conference, 2001, IEEE Proceedings.*, volume 7, page 3458, 2001.

[102] Texas Instruments TI. Analysis of the sallen key architecture: Application report. Available from http://www.ti.com/lit/an/sloa024b/sloa024b.pdf, 2002.

[103] Betty Tootell. *All Four Engines Have Failed ISBN 0-233-97758-9.* Andre Deutsch, 1985.

[104] Toshiba. *TLP 181 Datasheet.* http://www.toshiba.com/taec/components2/Datasheet_Sync//206/4191.pdf, 2009.

[105] US Presidential Commission US-PCOM. Report of the spaceshuttle challanger accident. Available from http://science.ksc.nasa.gov/shuttle/missions/51-l/docs/rogers-commission/table-of-contents.html, 1986.

[106] Danhua Wang. An approach of automatically performing fault tree analysis and failure mode effects techniques to software. *Software Engineering and Data Mining (SEDM), 2010 2nd International Conference*, 2010.

[107] D.D. Ward and S.E. Crozier. The uses and abuses of asil decomposition in iso 26262. In *System Safety, incorporating the Cyber Security Conference 2012, 7th IET International Conference on*, pages 1–6, 2012.

[108] Jerry C whitaker. *The Electronics Handbook ISBN:0-8493-8345-5.* IEEE Press, 1996.

[109] Edward Yourdon. *Modern structured analysis.* Yourdon Press, Upper Saddle River, NJ, USA, 1989.