ADIUVO
ENGINEERING AND TRAINING, LTD.

Introduction to Designing for High-Reliability

14.10.2020

PREPARED BY ADIUVO ENGINEERING AND TRAINING, LTD.

Section 1	Section 2	Section 3	Session 4
The Challenge	The Standards	The System	The FPGA

The Challenges

Depending upon the use case of the embedded system it may have to address challenges such as

- » Temperature
- » Vibration
- » Shock
- » EMC / ESD
- » Radiation – Terrestrial
- » Radiation – Space

Temperature

Wide variety of impacts on our system

- » Effect timing – Faster at lower temperatures, slower at high temperatures
- » Effect Power Consumption – increased power demands when temperature is low
- » May cause start up issues at low temperature e.g. oscillators
- » Higher temperatures will effect reliability
  - High Dissipation devices (processors, FPGA etc.) could exceed rated junction temperature

Temperature

How might we mitigate?

- » Simplest select parts which are appropriated rated e.g. industrial or Mil
- » Follow a derating standard to ensure Junction Temperature is appropriately derated
- » Use Margin – Qualify design at higher limits than operational e.g. +/- 10C to operating temperatures.
- » Work with mechanical & Thermal team to produce a design which uses Conduction, Radiation and Convection effectively to meet enable the derating to be achieved.
- » Ensure timing analysis, includes worse case performances – not just the best or nominal.

ADIUVO
ENGINEERING AND TRAINING, LTD.

Environment Vibration & Shock

Random Vibration – many causes
- Vehicles – random vibration caused by interaction with road surface
- Aircraft – Wind Buffeting, Turbine / Jet Engines

Sine Vibration
- Any reciprocal or regular cycling input

Acoustic
- Caused by reflected sound waves – Jet Take off 100 dB(A)
- Experienced by electronic boxes as random vibration

Shock - Short Duration High Acceleration Event
- Mechanical Release or Latching
- Equipment being dropped

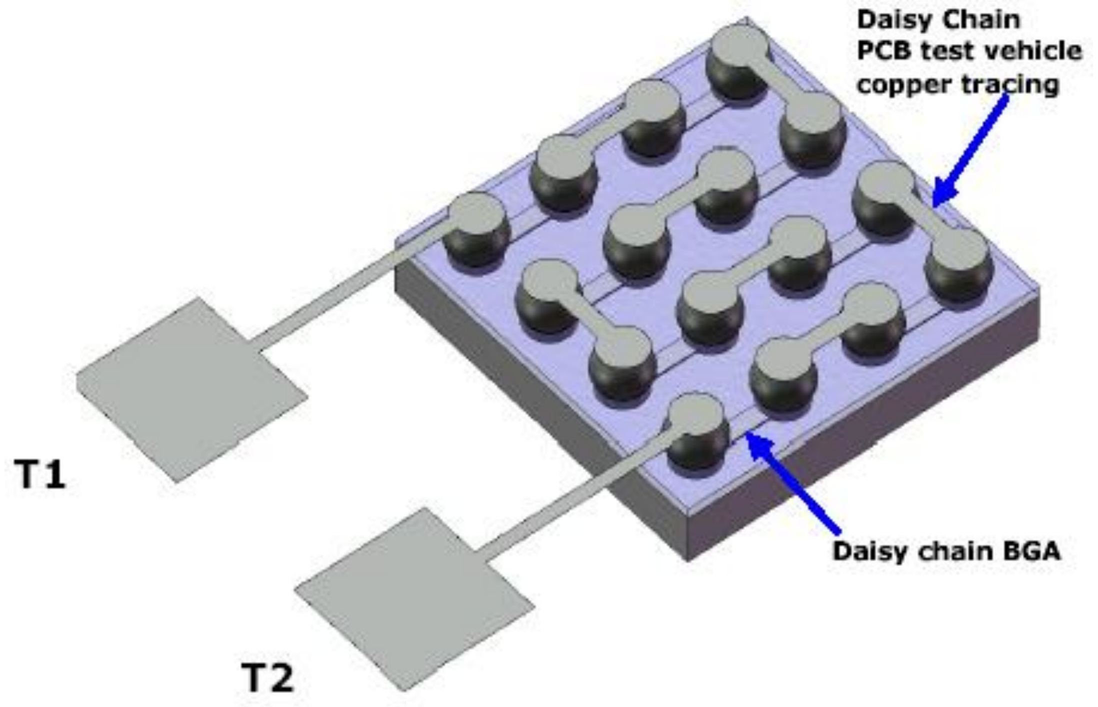
Can result in Failure of equipment due to physical failure – But can result in momentary noise and glitches

Daisy Chain Devices

If necessary we can chain multiple devices together

Used to ensure the mounting does not fail under dynamic loads

Bring out to external connectors. Test equipment can monitor the continuity



Environment – EMC / ESD

The ability of an Equipment, Sub-System or System to share the electromagnetic spectrum and perform their desired functions without unacceptable degradation from or to the environment in which they exist.

Needs to be able to accept and function correctly when subject to Radiated and Conducted Susceptibility

- » Complete corruption of digital words
- » Offsets in analogue signals
- » Crosstalk in communication systems
- » Equipment switch off/reset

Radiation Effects Environment

Radiation falls into four types

- » Total Ionizing Dose (TID)
- » Total Non Ionizing Dose (TNID)
- » Single Event Effect (SEE)
- » Deep Dielectric Discharge (DDD)

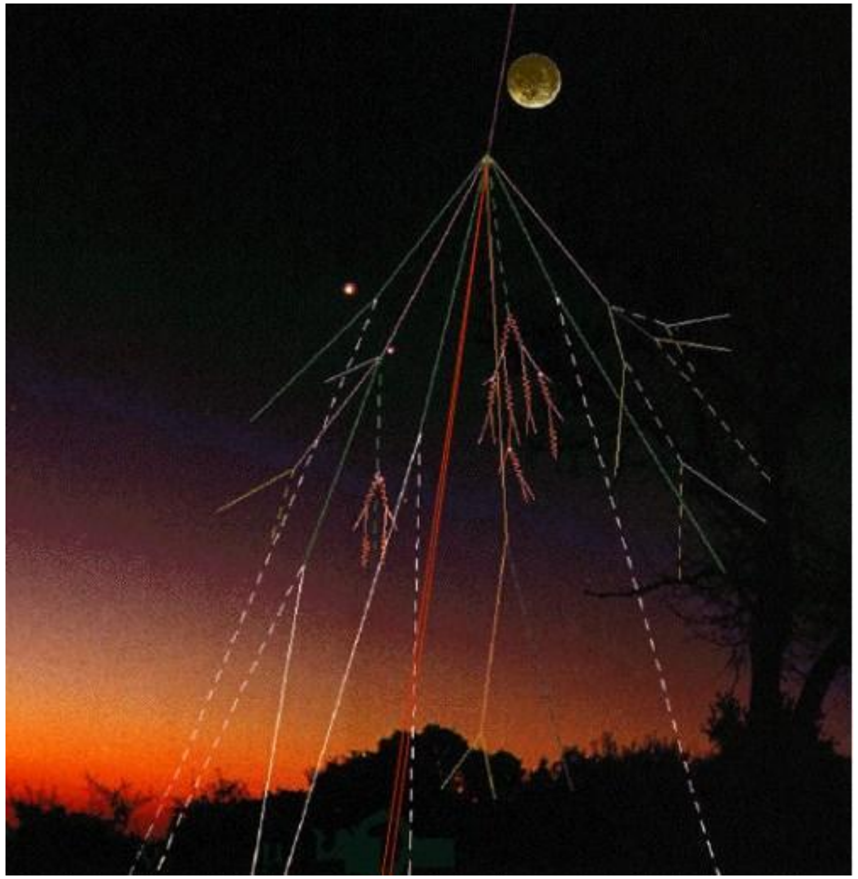
The first three of these categories have the potential to effect electronic components while the fourth DDD is a more systematic effect.

DDD allows a build up of charge in materials leading to an electrostatic discharge potentially leading to damage, for this reason all floating metal must be connected to ground

Environment – Atmospheric Neutrons

Atmospheric neutrons result from cosmic rays strike the earth atmosphere

They collide with other atoms present in the atmosphere creating neutron and proton showers



Radiation Effects Space Environment

Both TID and TNID are long term effects upon electronic components which result in the degradation of component parameters over time, for example Timing or Power parameters.

One problematic aspect of TID is Enhanced Low Dose Rate Sensitivity (ELDRS) .

ELDRS – Is when a device experiences worse performance degradation when subjected to a lower dose rate. This normally effects Bipolar or BiCMOS technologies.

ADIUVO
ENGINEERING AND TRAINING, LTD.

Radiation Effects

Single Event Effects are different to total dose radiation being instantaneous and therefore occurring at any point within a mission

SEE can be either destructive or non-destructive

When designing engineers tend to focus upon the non-destructive effects. This is due to having worked with part and radiation engineers to ensure the parts selected are radiation hard in line with the radiation hardness assurance requirements

ADIUVO
ENGINEERING AND TRAINING, LTD.

Radiation Effects

Description	Effect	Technology Affected	Destructive
Latchup – SEL	High Current	CMOS, BiCMOS	Yes
SnapBack –SESB	High Current	N Channel MOSFET, SOI Devices	Yes
Gate Rupture – SECR	Rupture of gate dielectric	Power MOSFET	Yes
Hard Error – SHE	Unalterable change of memory element	Memories, Latches, Registers	Yes
Burn Out – SEB	Destructive Burn Out	BJT, N Channel Power MOSFET	Yes
Upset – SEU	Register or Memory Corruption	Memories, Registers and Laches	No
Multi Bit Upset – MBU	As SEU but multiple bits affected	Memories, Registers and Laches	No
Functional Interrupt - SEFI	Loss of normal operation	Complex device e.g. FPGA	No
Transient – SET	Impulse Response	Analog & Mixed Circuitry	No
Disturb – SED	Momentary Corruption of information stored in a bit	Combinatorial Logic	No

Section 1	Section 2	Section 3	Session 4
The Challenge	The Standards	The System	The FPGA

Standards

Numerous standards depending upon the end application

IEC61508 – Functional Safety of Electrical, Electronic, Programmable Electronic Safety Systems. Introduces the famous SIL level, lead to many variants :-

- IEC62279 – Railway Implementation
- IEC61511 – Process Industry Implementation
- IEC61513 – Nuclear Industry Implementation
- IEC62061 – Machinery Implementation
- ISO26262 – Automotive Electrical / Electronic Systems

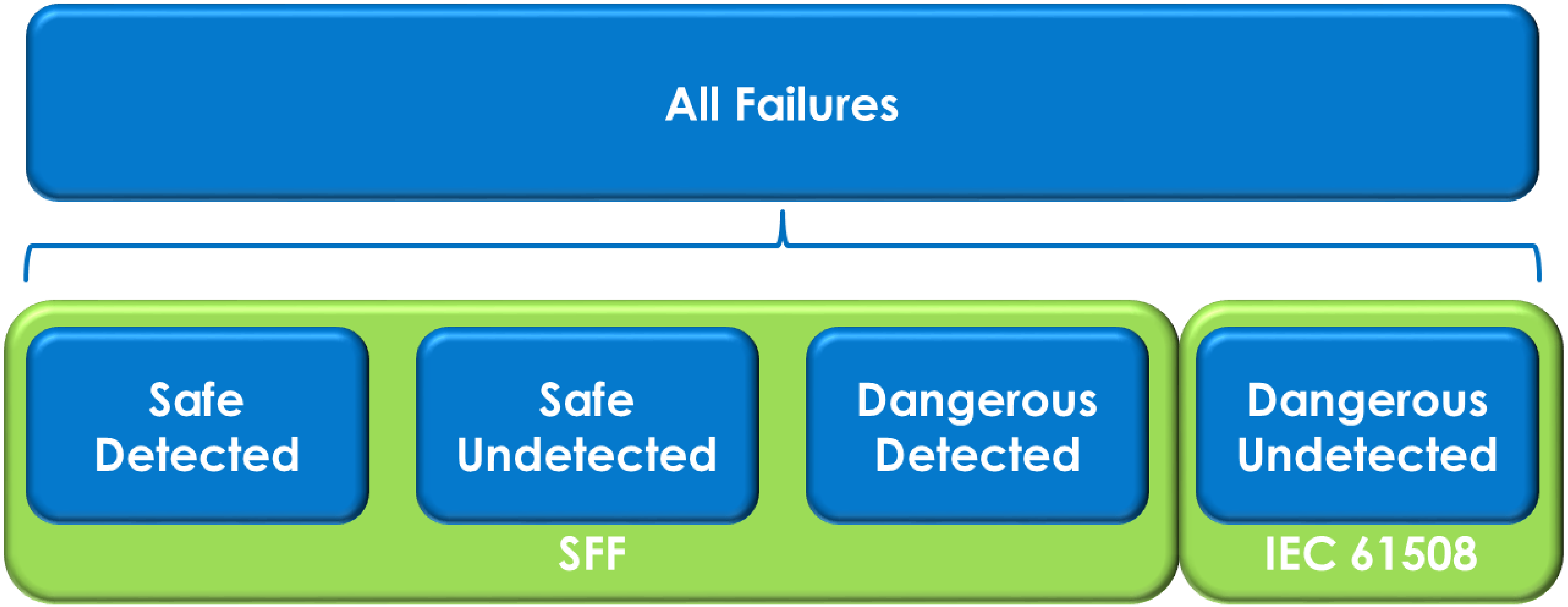
DO254 – Design Assurance Guidance for Airborne Electronic hardware

ADIUVO
ENGINEERING AND TRAINING, LTD.

SIL: Continuous Operation

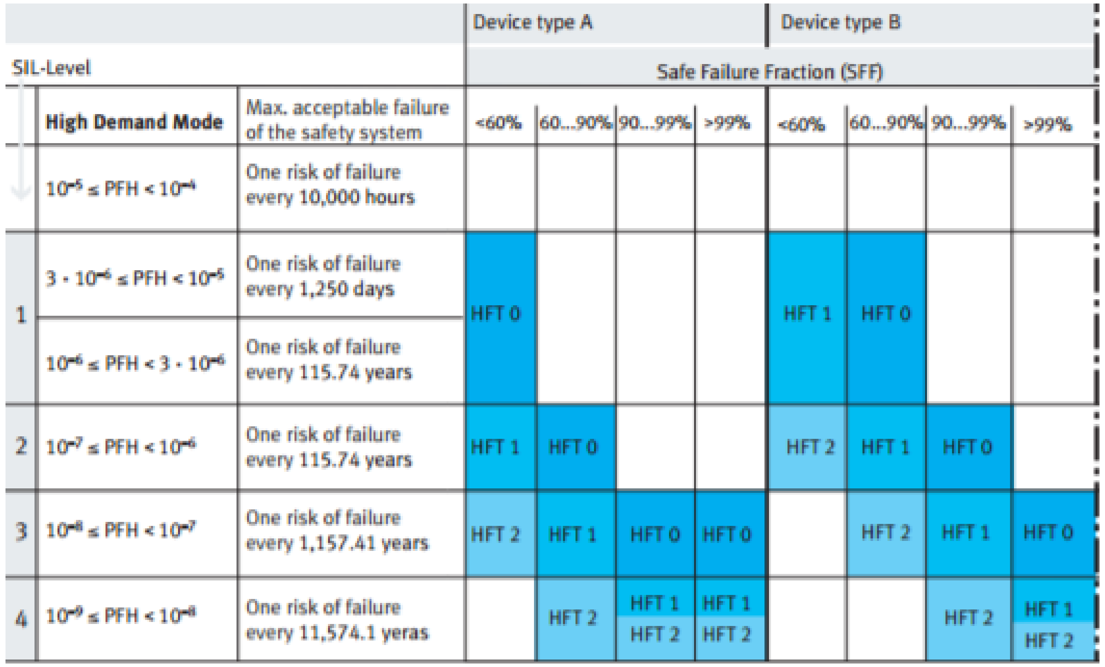
SIL	Time To Failure (hours)	Failure in Time (FIT)
4	100,000,000	10
3	10,000,000	100
2	1,000,000	1,000
1	100,000	10,000

Failure Space



Safe Failure Function

- Proportion of all safe and detected failures based on the total amount of failure
- FME(D)A (Failure Modes, Effects and Diagnostics Analysis) Methods of analysis for quantitative determination of types of failure and failure rates

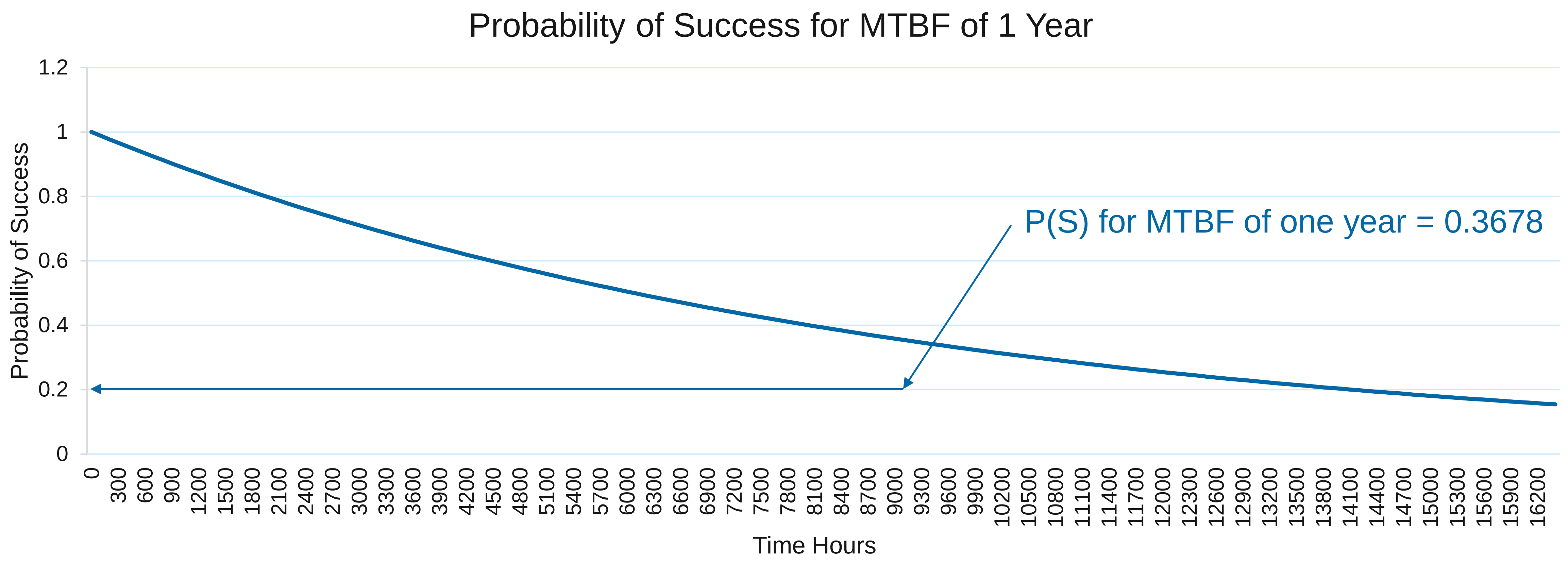


			Device type A				Device type B			
SIL-Level			Safe Failure Fraction (SFF)							
	High Demand Mode	Max. acceptable failure of the safety system	<60%	60...90%	90...99%	>99%	<60%	60...90%	90...99%	>99%
	$10^{-5} \leq PFH < 10^{-4}$	One risk of failure every 10,000 hours								
1	$3 \cdot 10^{-6} \leq PFH < 10^{-5}$	One risk of failure every 1,250 days	HFT 0				HFT 1	HFT 0		
	$10^{-6} \leq PFH < 3 \cdot 10^{-6}$	One risk of failure every 115.74 years								
2	$10^{-7} \leq PFH < 10^{-6}$	One risk of failure every 115.74 years	HFT 1	HFT 0			HFT 2	HFT 1	HFT 0	
3	$10^{-8} \leq PFH < 10^{-7}$	One risk of failure every 1,157.41 years	HFT 2	HFT 1	HFT 0	HFT 0		HFT 2	HFT 1	HFT 0
4	$10^{-9} \leq PFH < 10^{-8}$	One risk of failure every 11,574.1 yeras		HFT 2	HFT 1 HFT 2	HFT 1 HFT 2			HFT 2	HFT 1 HFT 2

MTBF

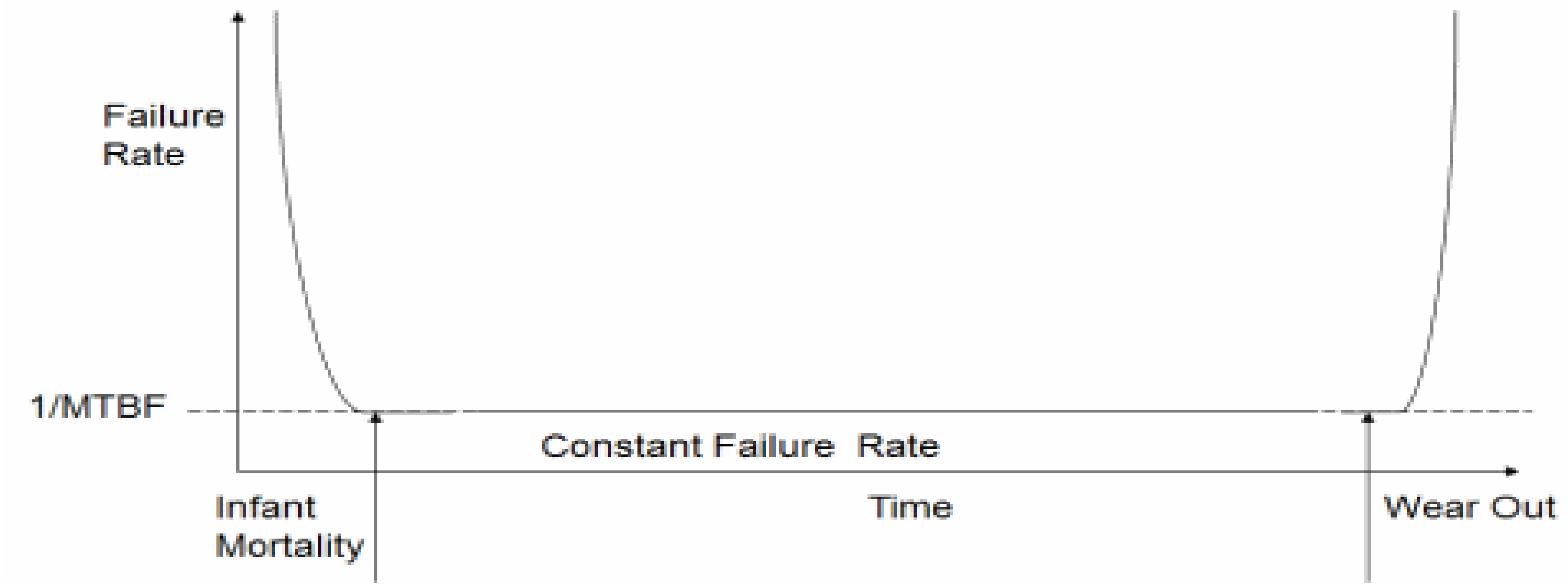
If a system has a MTBF of 8760 Hours (1 year) does that guarantee the system will operate without error for a year?

Probability of Success



MTBF

The MTBF is also only valid for the constant failure rate period of the equipment and does not include infant mortality or wear out phase.



For this reason units are normally burnt in to weed out infant mortality

Component Quality Level

Standard Commercial, Extended and Industrial

Differing standards for components

Type	Standard	Military	Space
Integrated Circuits	MIL-PRF-38535	QML Q (Class B)	QML V (Class S)
Hybrids	MIL-PRF-38536	Class H	Class K
Discrete	MIL-PRF-19500	JAN TXV	JAN S

Introduction of non-hermetic is leading to development of

Class L – Hybrids

QMV Y – Integrated Circuits

What are we protecting against?

Systematic – Faults arising from the design or manufacturing phase

Random – Occur due to stresses, aging or the environment

» Can be permanent faults e.g. an input stuck at a level
» Temporary e.g. memory corruption due to SEU

Avoidance

» How to AVOID (or to reduce the Probability of) the failure occurrence? No single point failures, best Hi Rel products

Tolerance

» How to TOLERATE failures? Recognize Avoidance is not to achieved and include Redundancy scheme in mitigation

Avoidance and Tolerance

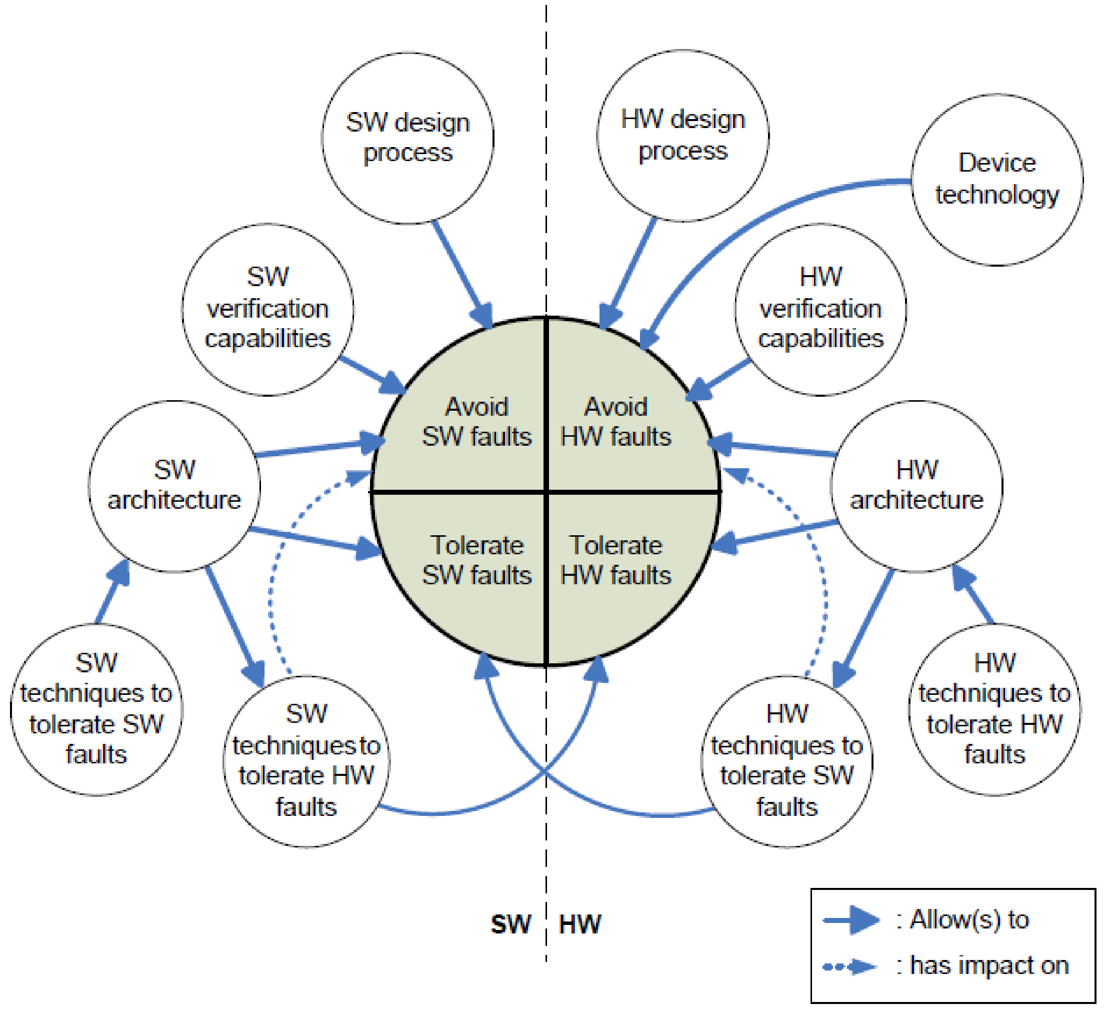
Examples of avoidance include

- » Can be permanent faults e.g. an input stuck at a level
- » Functional Analysis, Derating, Design Rules, Parts Quality Level, Worse case analysis, Quality Standards.

Example of Tolerance include

- » Detection – Identify a failure in the system
- » Localisation – Identify the failed element of the system
- » Isolation – Limit failure propagation
- » Reconfigure – Recover from the failure – e.g. redundant paths

Holistic Approach Required



SW design process
HW design process
Device technology
SW verification capabilities
HW verification capabilities
Avoid SW faults
Avoid HW faults
SW architecture
HW architecture
Tolerate SW faults
Tolerate HW faults
SW techniques to tolerate SW faults
SW techniques to tolerate HW faults
HW techniques to tolerate SW faults
HW techniques to tolerate HW faults
SW
HW
: Allow(s) to
: has impact on

Failure Rate Standards

There are a number of specifications for failure rate

- »Mil-Hnd Book 217F Notice 2
- »Telcordia SR332
- »Siemens Norm
- »FIDES
- »UTE 80-810

While Mil-Hnd BK 217 is the most well used it has not been updated for a number of years and as such is not accurate for modern advanced technology

Two Approaches FMECA vs FTA

Failure Mode Effect and Criticality Analysis:

- » Single event (-)
- » Exhaustivity (+)
- » Volume (-)
- » Easiness (+)
- » Extension to SW errors and operators errors (+)

FAULT TREE:

- » Events combination (+)
- » Based on analyst skill (-)
- » Optimised (+)
- » Difficult to perform (-)
- » Extension to SW errors and operators errors (+)

Failure Mode Effect and Criticality Analysis

Purpose:

- » To ASSESS the consequences of the elementary failures (parts, functional block...) on the system
- » To DESIGN (building) and DEMONSTRATE (verification) the efficiency of the FDIR policy.
- » To IDENTIFY Critical Items
- » To SUPPORT contingency analysis
- » To SUPPORT safety analysis

Phase:

- » Preliminary: functional approach
- » Detailed Design: detailed approach

Fault Tree Analysis

Purpose:

- To IDENTIFY the causes of hazardous events
- To IDENTIFY Critical Items
- To SUPPORT Safety analysis

Preliminary Design: functional approach (system level)

Hardware Level – Derating

All manufacturer component data sheets provide absolute maximum and operating electrical stresses for the device in question

If the engineer designed the system such that the device was operating with an electrical stress just below the absolute maximum the reliability of the design would be considerably lowered as it would be operating outside the recommended operating conditions

Reducing the electrical stress upon the design therefore enables the engineer to produce reliable equipment.

Hardware Level - Derating Standards

- Depending upon end user / application
  - » Mil-STD-975 – NASA
  - » Mil-STD-1547 – DoD
  - » AS4613 – US Navy
  - » Nav Sea TE000-AB-GTP-010 – US Navy
  - » ESCC-Q-30-11A – ESA
  - » MSFC-STD-3012 – NASA Marshall Space Centre

Section 1	Section 2	Section 3	Session 4
The Challenge	The Standards	The System	The FPGA

Redundancy

Depends on architecture

- » FAIL Safe - failure detected and goes to a safe state operation does not continue
- » Continue operation – continues to operate following a failure – e.g. satellite

Active Redundancy – transition is seamless no break in operation

Cold Spared – Redundant part powered down, takes time to get up and running

What can affect redundancy

- » Electrical PROPAGATION - Ensure electrical isolation depending on failure mode
- » Physical PROPAGATION - Ensure separation of redundant channels
- » COMMON CAUSE – Effects all redundant channels

Common Cause failures

Within a electronic system there exist a number of Common Cause Failures

- » Power Supplies and sequencing
- » Clocks
- » Resets
- » Memories and Configuration memories
- » IO interfacing – Share but need to be careful about fault propagation
- » Using Component from the same batch

Section 1	Section 2	Section 3	Session 4
The Challenge	The Standards	The System	The FPGA

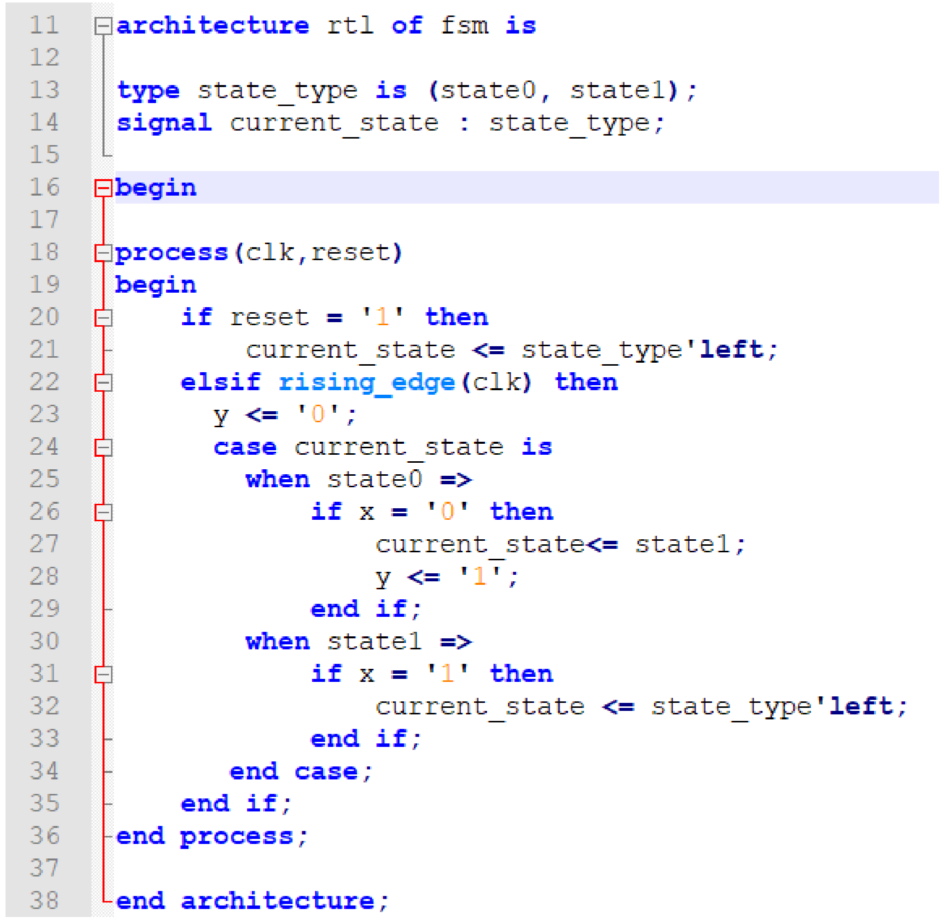
Code Quality

- What do we mean by code quality
  - » Readable and Maintainable
  - » Comply with coding standards – either internal or external
  - » Ensuring no simulation / synthesis mismatch
  - » Enforce RTL clarity and reduce complexity
  - » Enable Testability and Traceability of the code
  - » Specific checks include
    - FSM without terminal states, unmapped states
    - Long If Then Else (ITE) chains
    - Control Signal test
    - Ensuring Structural correctness – e.g. used, undriven nets.
    - Gated Clocks
    - Allowable Types - e.g. std logic, unsigned etc

How can we achieve code quality?

- Stringent set of coding rules – Including
  - » Naming conventions (Architectures, Packages, Test Benches, Signals) etc.
    - Signal should indicate bi-directional, active low and synchronisation signals
  - » Using Constants in place of hard coded numbers – in common packages if shared
  - » Use of enumerated types for FSM
  - » Rules on Signal vs Variable use
  - » Rules on Synthesis Attributes

Bad coding practice example



```vhdl
architecture rtl of fsm is

type state_type is (state0, state1);
signal current_state : state_type;

begin

process(clk,reset)
begin
    if reset = '1' then
        current_state <= state_type'left;
    elsif rising_edge(clk) then
      y <= '0';
      case current_state is
        when state0 =>
            if x = '0' then
                current_state<= state1;
                y <= '1';
            end if;
        when state1 =>
            if x = '1' then
                current_state <= state_type'left;
            end if;
       end case;
    end if;
end process;

end architecture;
```

ADIUVO
ENGINEERING AND TRAINING, LTD.

How can we enforce code quality

- Linting tools – able to check the code against language and other rules

- RTL Analysis e.g. Blue Pearl – Visual Verification Suite
  - » Enables Linting checks against standard
  - » Structural Design Checking
  - » FSM Checking
  - » Clock Domain Crossing Analysis
  - » False and Multi Cycle Path Analysis

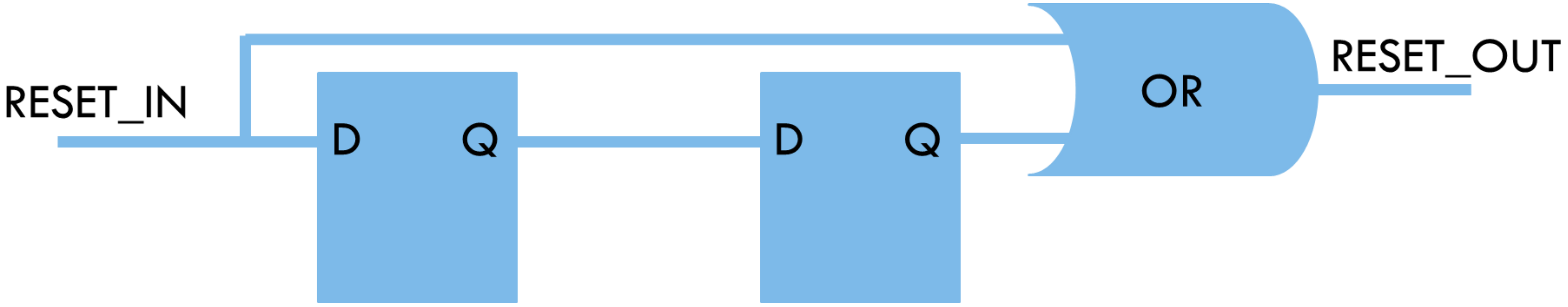
Clocks

Very simple approach which is easily often to get wrong

- Consider Oscillator start up time in the design analysis
- Carefully Plan the use of Clock Global and Local Clock resources
- Be very careful about crossing clock domains
- Ensure Asynchronous inputs are synchronised to the clock domain

Resets

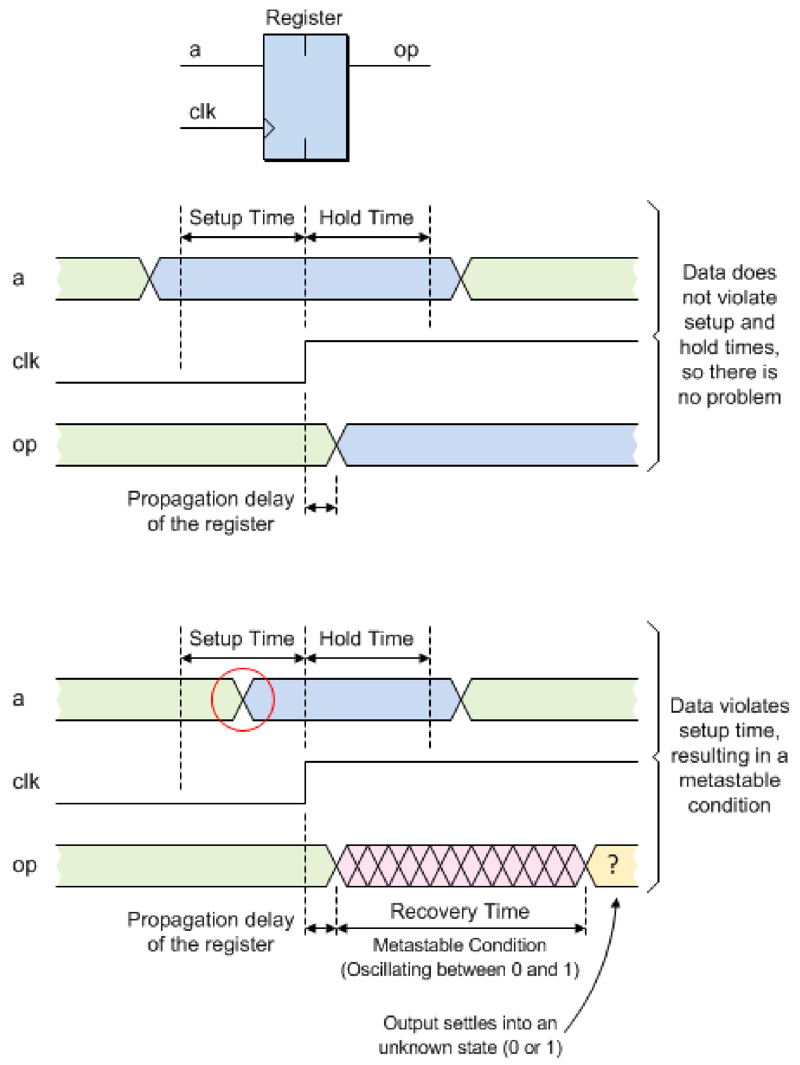
- Do not rely upon the default value at power up for critical aspects
- Reset needs to be Asynchronous Assertion – See clock start up time
- Synchronous De Assertion – Prevents the risk of meta-stability when cleared
- Ensure Outputs are reset to de assert peripherals especially bi directional busses



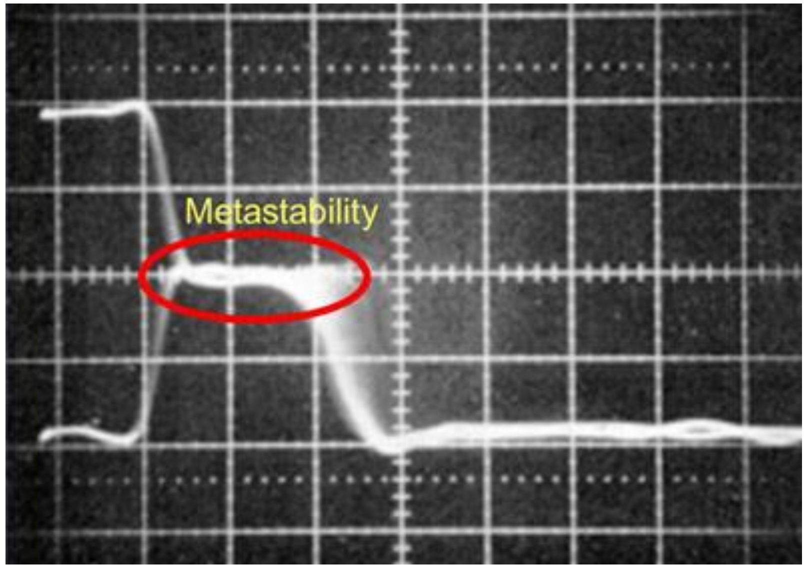
Metastability

- One issue which can arise with incorrect domain crossing is metastability
- This can lead to corruption of data or incorrect behaviour
- Occurs when a flip flops set up or hold time is violated

Metastability



Register
a
op
clk
Setup Time
Hold Time
a
clk
op
Data does not violate setup and hold times, so there is no problem
Propagation delay of the register
Setup Time
Hold Time
a
clk
op
?
Data violates setup time, resulting in a metastable condition
Propagation delay of the register
Recovery Time
Metastable Condition (Oscillating between 0 and 1)
Output settles into an unknown state (0 or 1)



Clock Domain Crossing

Several Techniques which can be used depending upon what needs to be transferred

- Two stage synchroniser – Ideal for single bit data
- Grey Code Synchroniser – Encodes data bus in grey code and transfer between domains – Ideal for counters as input to be converted to grey code can only decrement / increment by one from previous value
- Hand shake synchroniser – Transfers data bus between two clock domains using handshake signals
- Pulse synchroniser – transfer pulse from one clock domain to another
- Asynchronous FIFO – transfers data from one domain to another, useful for high throughput / burst transfers

CDC Design Analysis tools

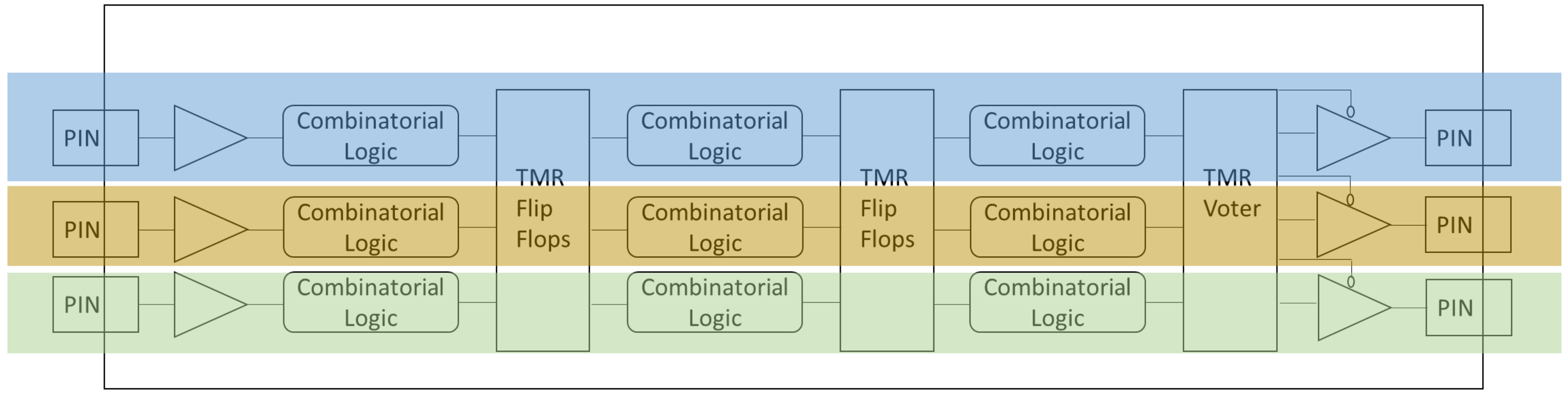
- Detecting all CDC issues can be a challenge in large designs
  - » Are all IP IO on the correct clock domain
  - » Very easy to associate signal with wrong domain e.g. FIFO empty and WR clock
- CDC issues can be very difficult to find changing on each start up and may be intermittent
- Can be hard to find in simulation – Timing simulation required, takes a long time simulate the system
- Static analysis tools are better suited to find the CDC issues.

Blue Pearl - Demo

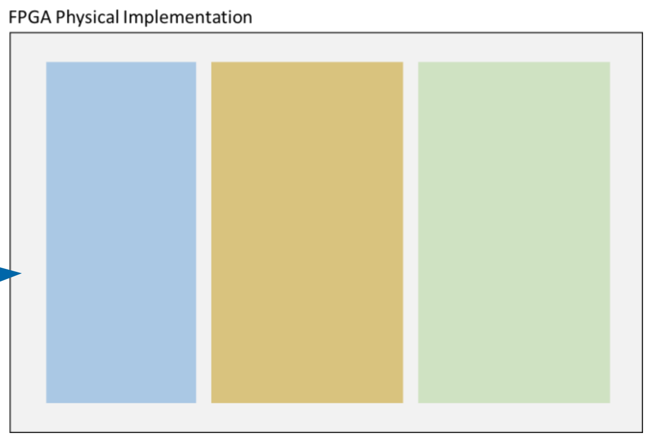
Global TMR

- Triplicates all resources in the design including IOB and Clocks, reset trees
- Protects entire design from SEU and SET
- Protects from errors in configuration memory – but it does not correct them
- More complicated to implement
  - Areas Penalty reduces size of design to be implemented
  - Validation is increased need to ensure final bit file is implemented as desired
  - Will have an impact upon power of the design
  - Need to manage clock skew carefully
- TMR needs to re synchronise

Global TMR



Need to ensure spatial separation in the implemented device

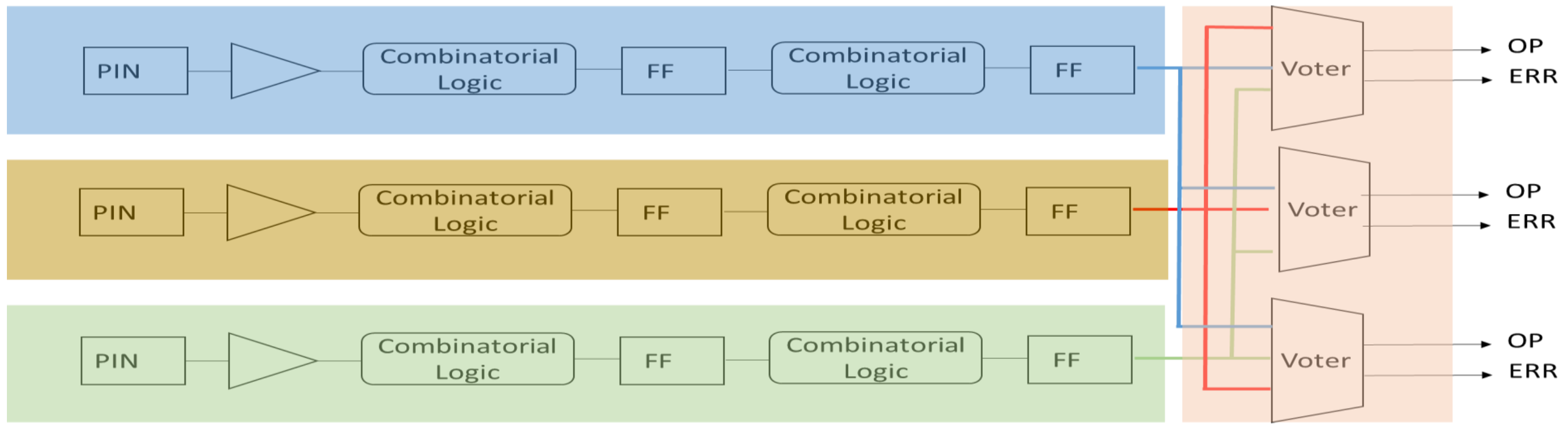


Large Grain TMR

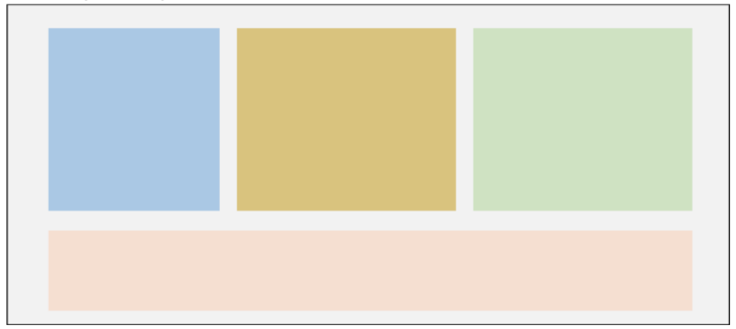
- Triplicates all resources in the design including IOB and Clocks, reset trees, HOWEVER Flip Flops are not voted upon
- Uses one voter prior to the output of the three modules.
- Unlike Global TMR the FF are not resynchronised
  - Can be used with partial reconfiguration to reconfigure a incorrect chain if required.
- It has minimal domain cross points unlike global TMR.
- Mitigates both configuration and user logic errors
- Like global TMR it has area and power penalties

ADIUVO
ENGINEERING AND TRAINING, LTD.

Large Grain TMR

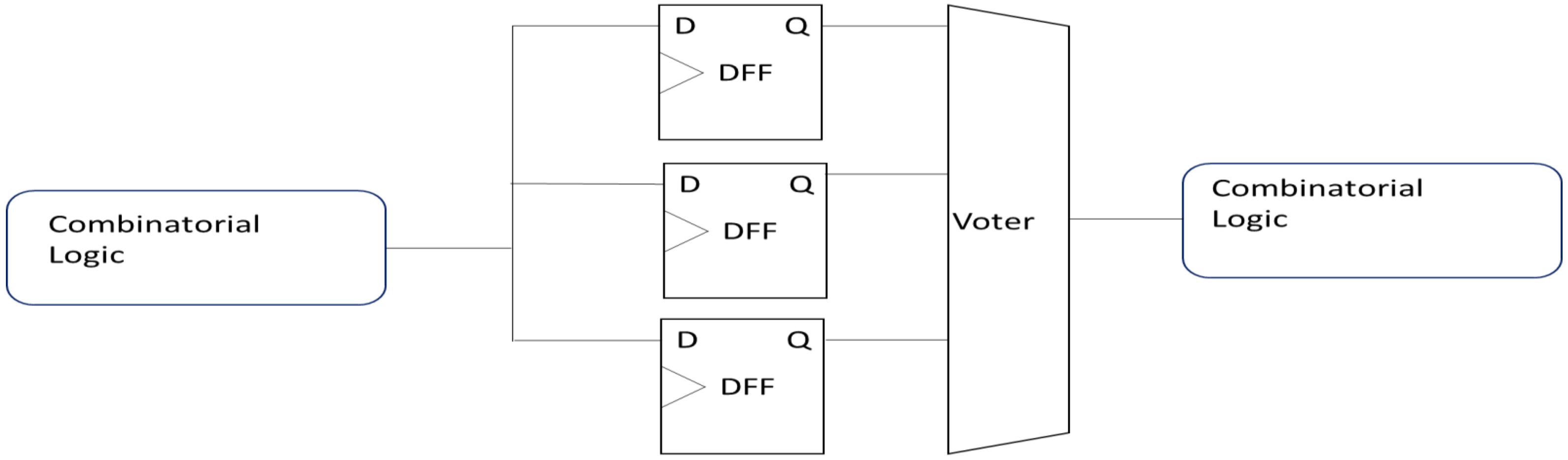


FPGA Physical Implementation



Local TMR

- Triplicates Flip Flops and votes on the output
- Best used in slow deigns to stop SET being clocked in
- Offers area advantage as combinatorial logic is not replicated
- Mitigates both user and configuration memory



State Machine

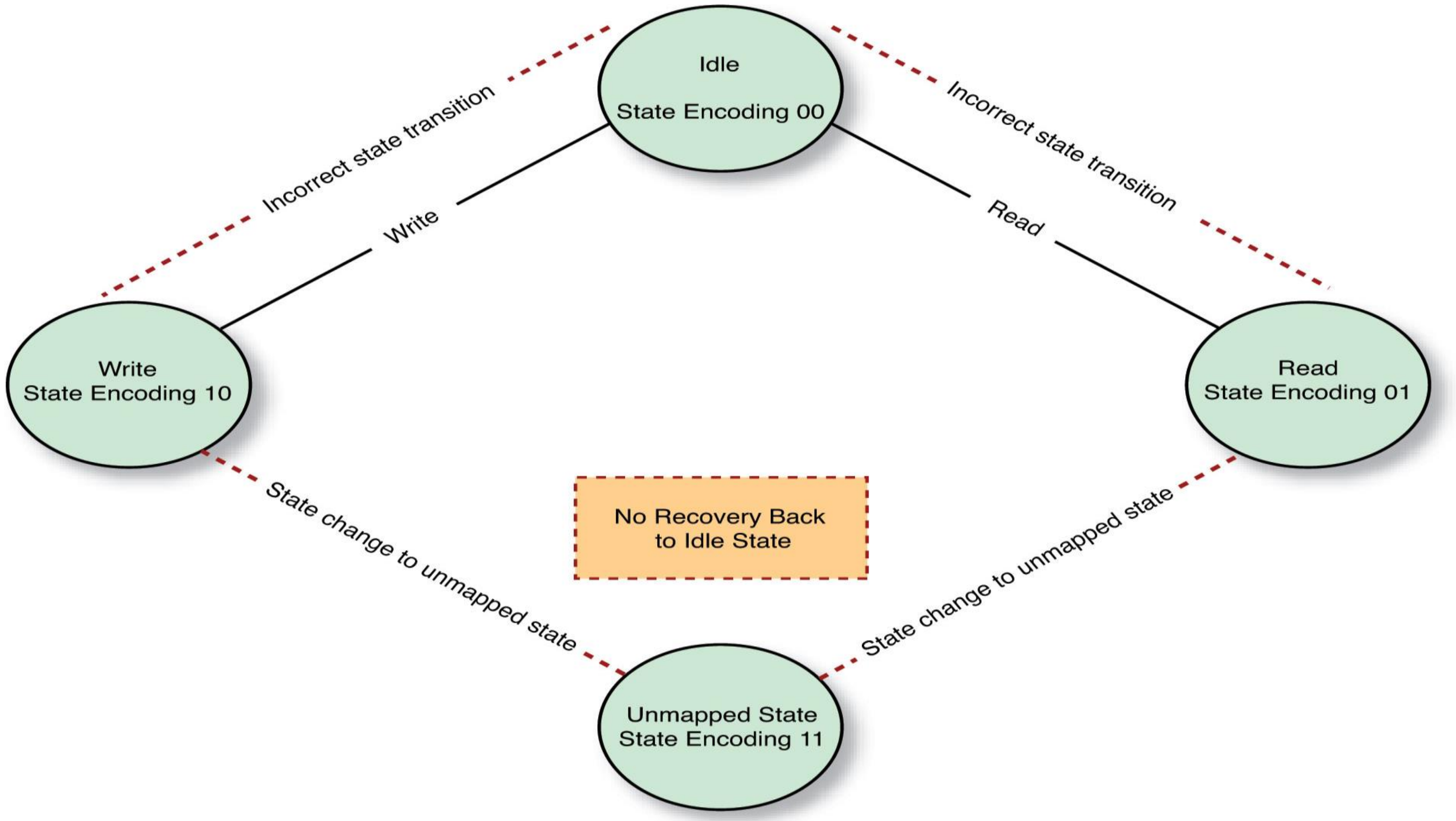
State machines are logical constructs that transition between a finite number of states. They are often at the heart of many of our FPGA designs.

However, there are several issues which can arise if not designed correctly

- » Terminal states – State from which once entered there is no path to leave
- » Unused states – State which are unused in the design, transition in these states also results in a terminal state.
- » Safe state machine – Adds in additional logic, which could be effected and cause un-expected behaviour
- » Encoding – Is the state machine encoding correct for the environmental challenges

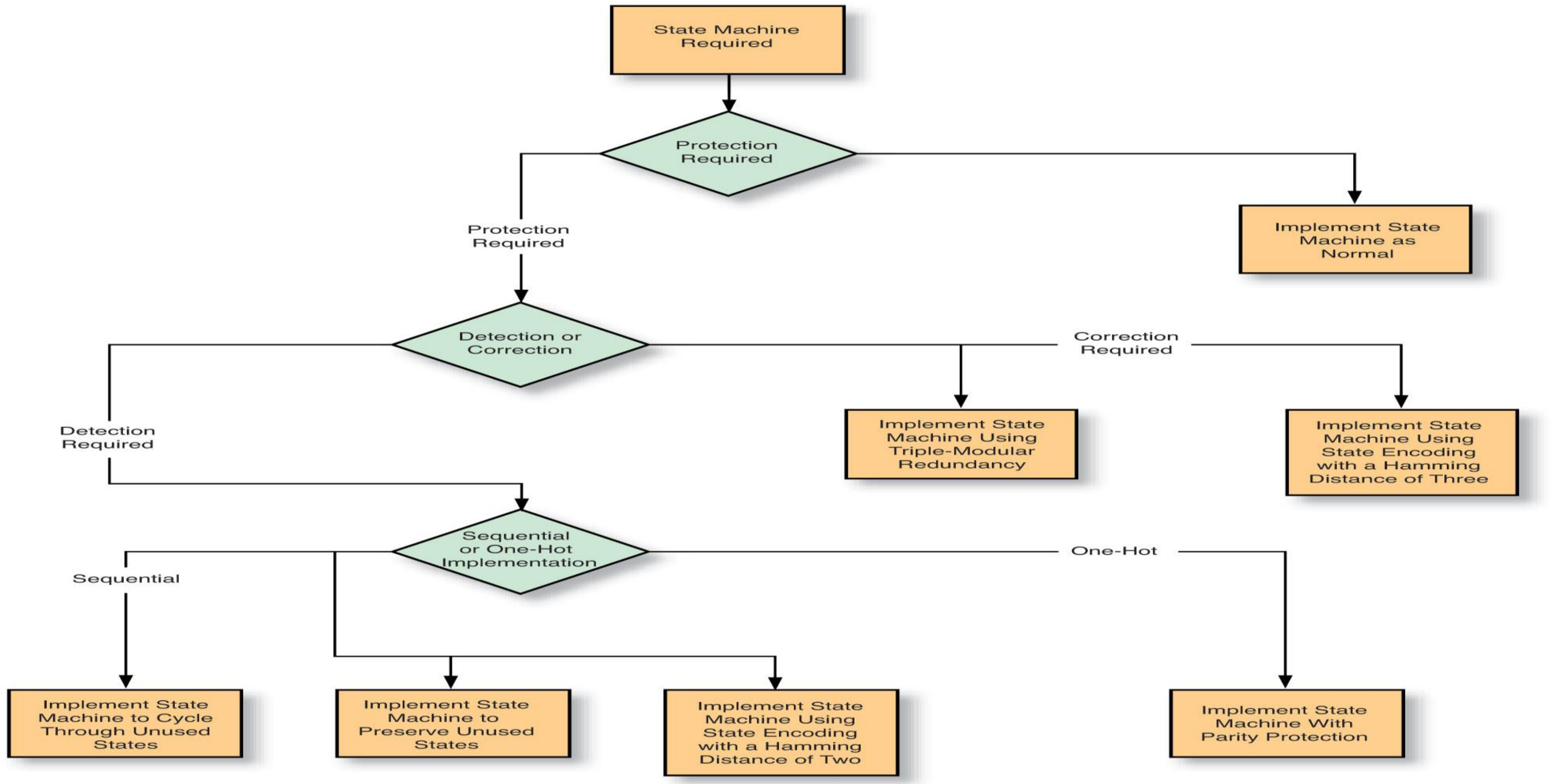
State Machine

The Unmapped State Machine


Idle
State Encoding 00
Incorrect state transition
Write
Incorrect state transition
Read
Write
State Encoding 10
Read
State Encoding 01
No Recovery Back to Idle State
State change to unmapped state
State change to unmapped state
Unmapped State
State Encoding 11

State Machine

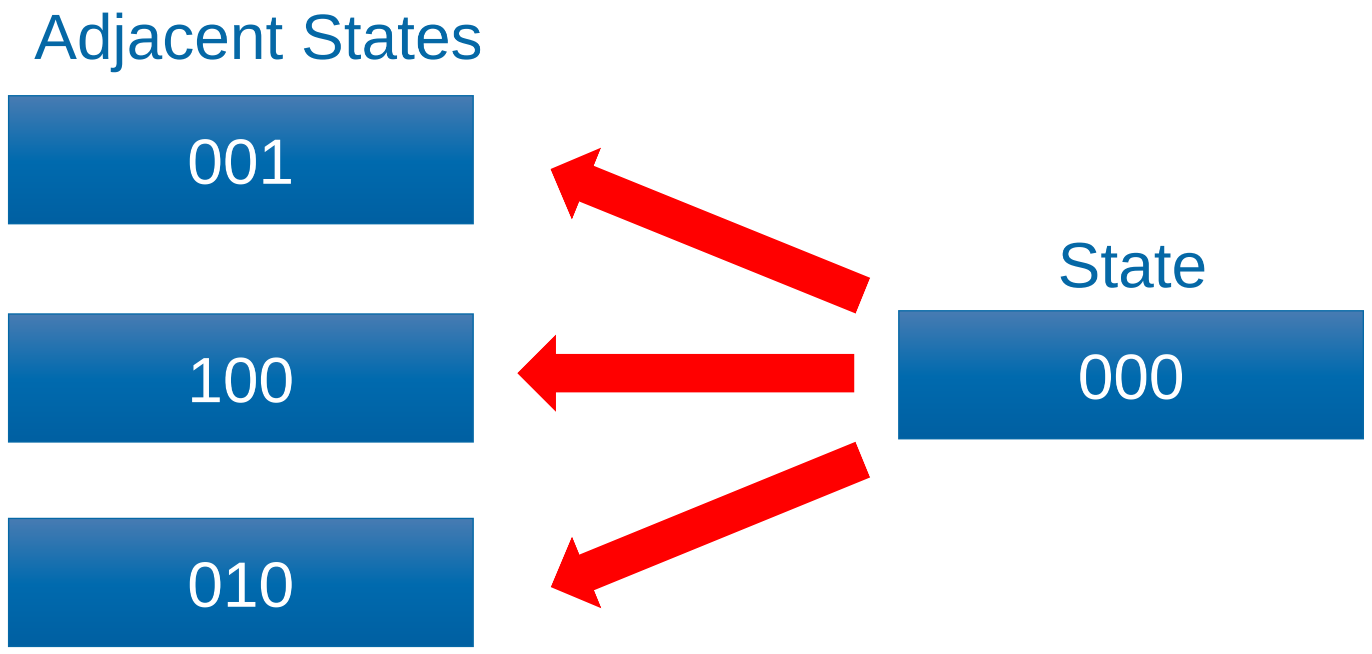
State machine choices other than TMR



State Machine Required
Protection Required
Implement State Machine as Normal
Protection Required
Detection or Correction
Correction Required
Implement State Machine Using Triple-Modular Redundancy
Implement State Machine Using State Encoding with a Hamming Distance of Three
Detection Required
Sequential or One-Hot Implementation
Sequential
One-Hot
Implement State Machine to Cycle Through Unused States
Implement State Machine to Preserve Unused States
Implement State Machine Using State Encoding with a Hamming Distance of Two
Implement State Machine With Parity Protection

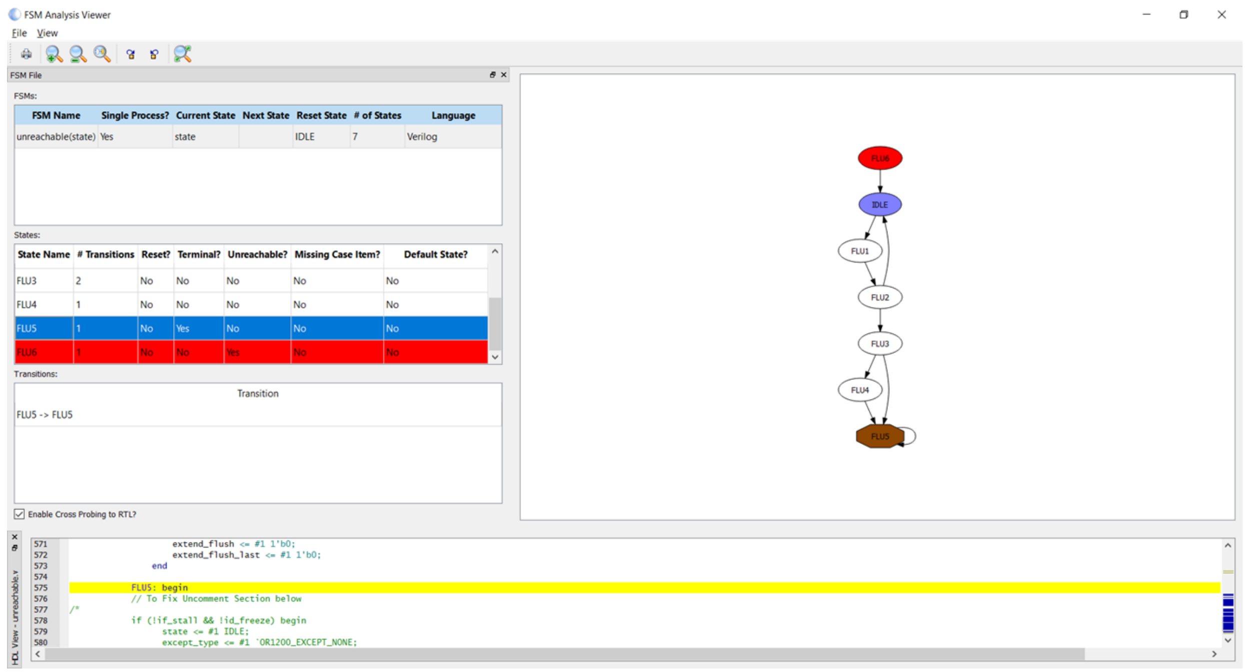
State Machines

- What does a Hamming code of three mean?
- For each state with n number of bits there are also n adjacent states



State Machine Analysis

- Complex state machines can be hard to analyse
- Are all states mapped
- Is there terminal states
- Static FSM Analysis is important in providing understanding & correct behaviour
- Can be used for documentation as well
- Blue Pearl VVS ideal



FSMs:

FSM Name	Single Process?	Current State	Next State	Reset State	# of States	Language
unreachable(state)	Yes	state		IDLE	7	Verilog

States:

State Name	# Transitions	Reset?	Terminal?	Unreachable?	Missing Case Item?	Default State?
FLU3	2	No	No	No	No	No
FLU4	1	No	No	No	No	No
FLU5	1	No	Yes	No	No	No
FLU6	1	No	No	Yes	No	No

Transitions:

Transition
FLU5 -> FLU5

[✓] Enable Cross Probing to RTL?

```
                    extend_flush <= #1 1'b0;
                    extend_flush_last <= #1 1'b0;
                end

            FLU5: begin
            // To Fix Uncomment Section below
/*
            if (!if_stall && !id_freeze) begin
                state <= #1 IDLE;
                except_type <= #1 `OR1200_EXCEPT_NONE;
```

Counters

Many Engineers look just for the terminal count of the counter for example

```
IF count = 9 THEN
```

However SEU can result in the value of the counter being >= 9

In such a case the counter would fail, and the effects may propagate

Instead use

```
IF count >= 9 THEN
```

In the worst case the timer will action early (which can be mitigated at FPGA level) rather than lock up the counter

Watchdog

- Commonly used in embedded systems
- Useful in FPGA systems as well to ensure a single event functional interrupt (SEFI) lock up has not occurred.
- If watchdog is not petted the device can be reset or reprogramming is triggered.
- Can be integrated in the FPGA if desired

Questions ?

ADIUVO
ENGINEERING AND TRAINING, LTD.

ADIUVO

ENGINEERING AND TRAINING, LTD.

www.adiuvoengineering.com

adam@adiuvoengineering.com