2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Using Static RTL Analysis to Accelerate Satellite FPGA Verification

Adam Taylor

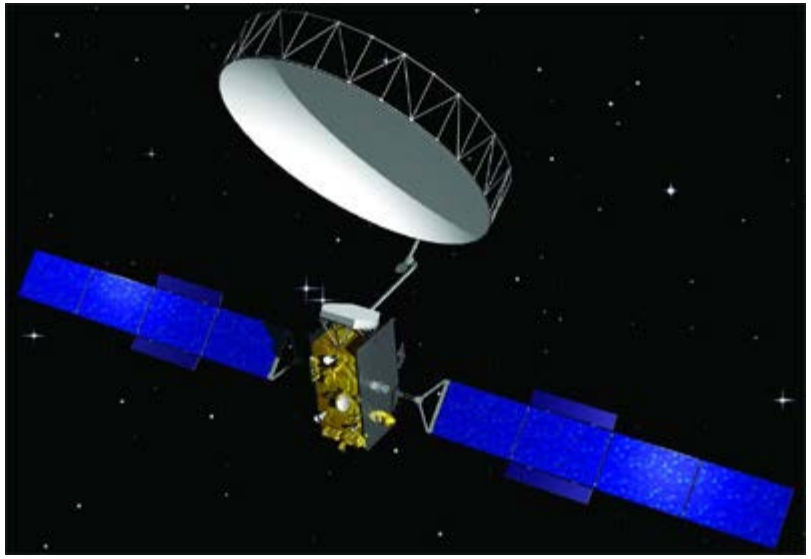
Speaker: Dave Wallace

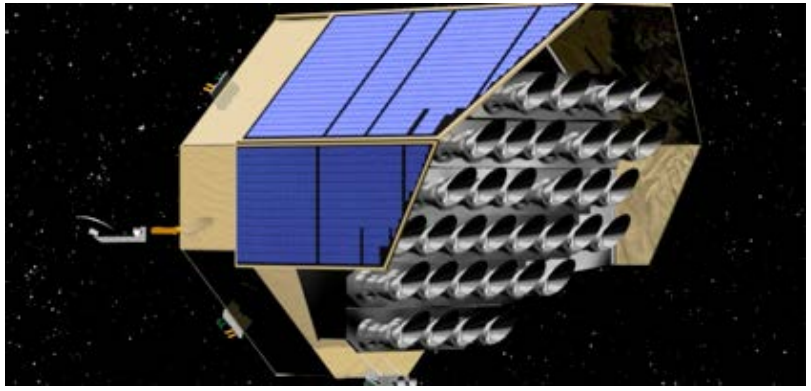
Visual Verification™ Suite
The next generation of RTL verification

accellera SYSTEMS INITIATIVE

FPGA In Satellite Applications

- FPGAs, flexibility and performance means significant use in satellite applications. Examples Include
  - Payload Control & Housekeeping
    - Used to implement control algorithms such as PID for temperature control
  - Imaging e.g. Multi Spectral Earth observation, or deep space science missions
    - Sensor Interfacing and image processing and compression
  - Telecommunications e.g. Inmarsat and Satellite Military Communications
    - Beam Forming, signal processing and channelization





FPGA Challenges

- Several Challenges faced in Orbit
  - Single Event Effects (SEE) – These can flip the state of bits in FPGA Configuration, BRAM and Flip Flops – potentially leading to functional or performance issues
  - Total Ionising Dose (TID) – Affects parametric performance of the device, reduces timing performance, increases power consumption
  - Thermal management – Achieving junction temperature targets to comply with de-rating standards.

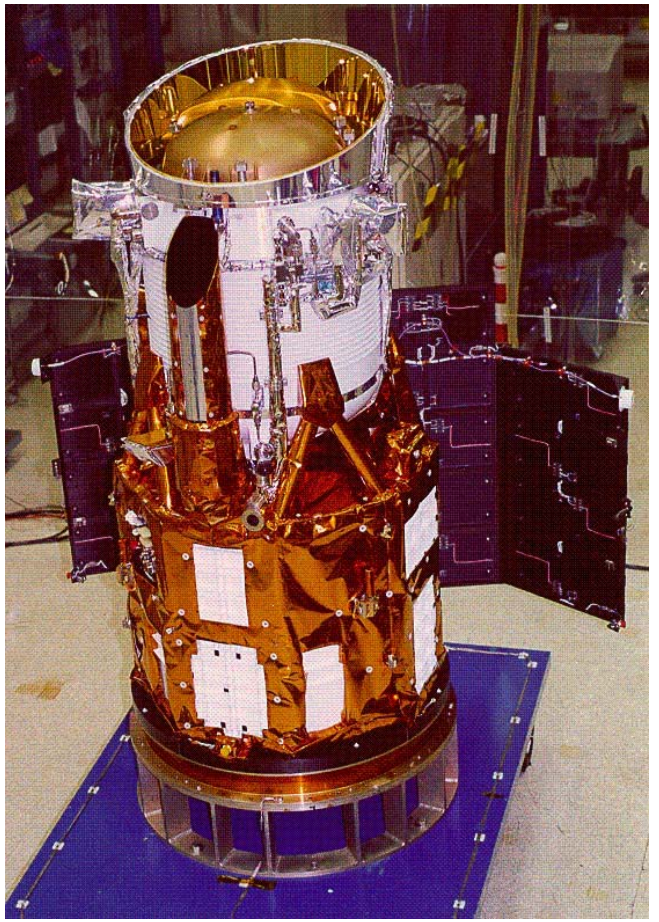
FPGA Challenges – Development

- Not limited to space significant challenges on earth
  - SEE Mitigation
    - Error Correcting Codes, FSM Analysis – remove unused states / deadlock etc
  - Ensuring Code Quality & Auditing
    - Typical approach design standards, simulation, code reviews
  - Clock Domain Crossing
    - Hard to find in Simulation, often rely on implementation tools

In Orbit Failures – FPGA Issues

- Failures in orbit are both expensive and nationally / internationally embarrassing
- 1999 NASA WideField InfraRed Explorer
  - Relied upon default state on power on
- ESA FPGA Task force review – upon reviewing ESA and NASA FPGA developments

  "FPGAs should be forbidden in space applications until the designers learn how to design with them."



Addressing the Challenges

- **Enforcement of the RTL Coding rules** to a company / organizational standard e.g. NASA, ESA
- **Analysis of the RTL modules for structural RTL issues** e.g. Missing Case, Reset Synchronization, Missing Reset, Using Initial Value etc.
- **State Machine Analysis,** the ability to analyze state machines within the design. This is crucial to prevent terminal states and ensure unused states are correctly addressed.
- **Clock Domain Analysis**, the ability to analyze clock domain crossings within the design and identify if the crossing is safe or not. As FPGA development increasingly uses Intellectual Property (IP) blocks from several vendors this tool must also be able to identify clock domains associated with IP blocks including those encrypted with IEEE 1735.
- **Metrics** which enable the design maturity and status of coding rules e.g. where they correctly run in the analysis.

Static Analysis

- Enables us to identify structural issues in the code and compliance with naming standards
- Can leverage formal design techniques – but not wholly reliant on them
- Different to simulation, simulation requires the correct question is asked – very challenging for CDC.
- Many issues found in simulation are not functional issues but simple user issues [DVCon Panel 2018]

RTL Standards and Analysis – 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 can 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

RTL Standards & Analysis

- A good design starts with compliance against an agreed coding standards / Design Rules
- Design Rules will include
  - Naming conventions - (Architectures, Packages, Test Benches, Signals) etc.
    - Signal should indicate bi-directional, active low and synchronisation signals
  - Asynchronous assertion, synchronous de assertion of resets
  - Ability to reset all flip flops to a known state
  - State machines designed to prevent deadlock / lock up due to single event effects
  - Use of Error Detection and Correction codes on stored data
  - Safe Clock Domain Crossing between asynchronous domains

Why is this bad practice ?

```
11  architecture rtl of fsm is
12
13  type state_type is (state0, state1);
14  signal current_state : state_type;
15
16  begin
17
18  process(clk,reset)
19  begin
20      if reset = '1' then
21          current_state <= state_type'left;
22      elsif rising_edge(clk) then
23        y <= '0';
24        case current_state is
25          when state0 =>
26              if x = '0' then
27                  current_state<= state1;
28                  y <= '1';
29              end if;
30          when state1 =>
31              if x = '1' then
32                  current_state <= state_type'left;
33              end if;
34         end case;
35      end if;
36  end process;
37
38  end architecture;
```

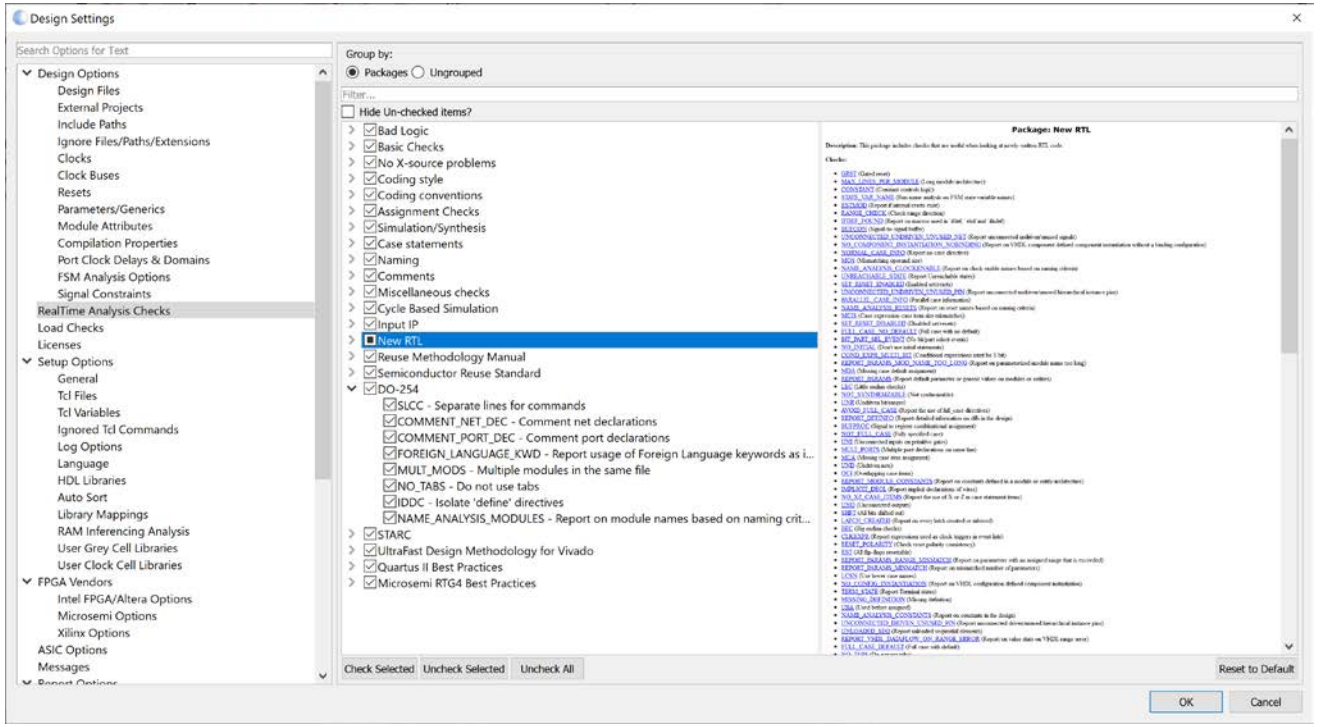
Reset State is defined purely by position in the declaration

How big is the code quality issue ?

- Wilson Group Survey Respondents report 84% of designs make it to the field with a non-trivial bug
- Aspen Core Embedded Survey responses indicate the biggest issue to delivery is the development time scales expected and debugging.
- While not specific to the space industry they show an industry wide trend.
- **Pressure on time and resources leads invariably to quality issues as errors with the source code are not identified during peer review but later in the development cycle, extending the cycle**
- ***Structural analysis tools alleviate the time and resource issues by automating code quality enforcement***

How can we enforce code quality?

- Linting tools – able to check the code against language and other rules
- RTL Analysis, like Blue Pearl's Visual Verification Suite
  - Enables Linting checks against standard
  - Structural Design Checking
  - FSM Checking
  - Clock Domain Crossing Analysis
  - False and Multi Cycle Path Analysis



FSM Analysis

- State machines are the heart of the FPGA design
- Incorrect implementation in a space application could result in significant issues including deadlock
- Manual analysis can easily miss deadlocks and terminal states – especially in complex state machines
- Automated FSM Analysis
  - Enables analysis of complex state machines
  - Visual debugging environment to correct issues
  - Can save several hours in simulation of reputation if issue occurs in flight

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

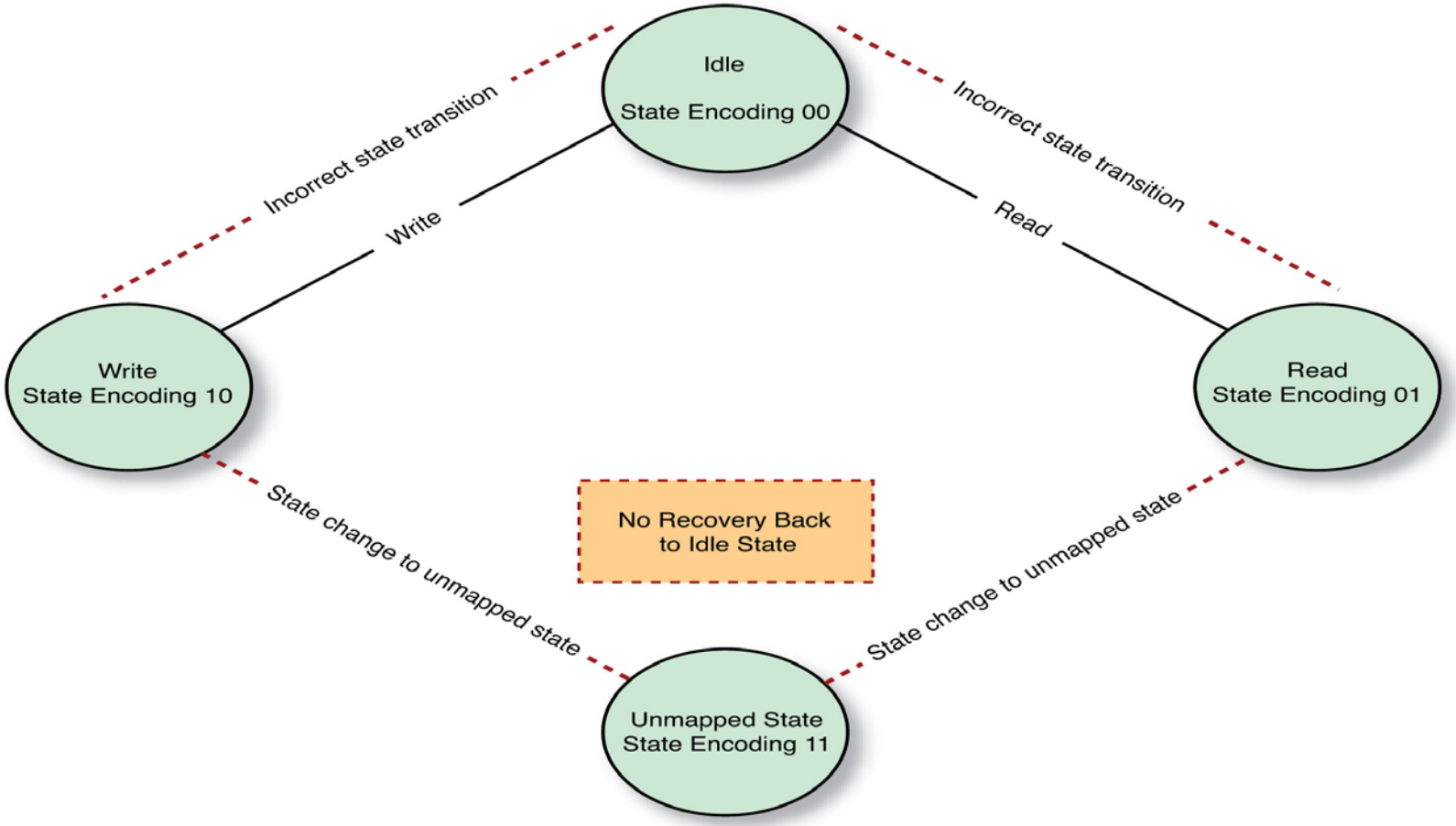
- Terminal states – State from which once entered there is no path to leave
- Unused states – State which are unused in the design, transition into these states may also result in a terminal state.

- Safe state machine – Adds in additional logic, which could be affected and cause unexpected behaviour

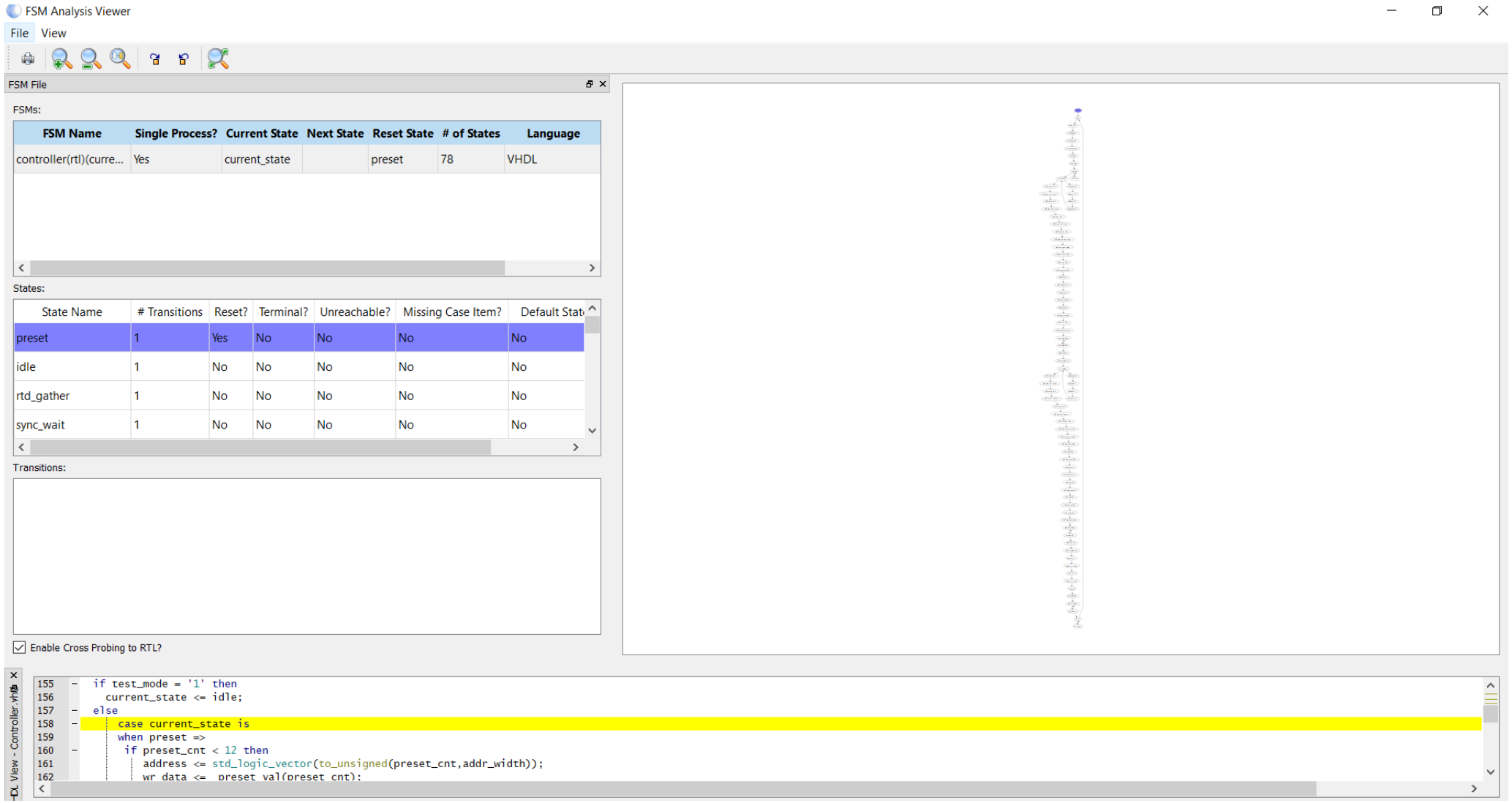
- Encoding – Is the state machine encoding correct for the environmental challenges?

State Machine

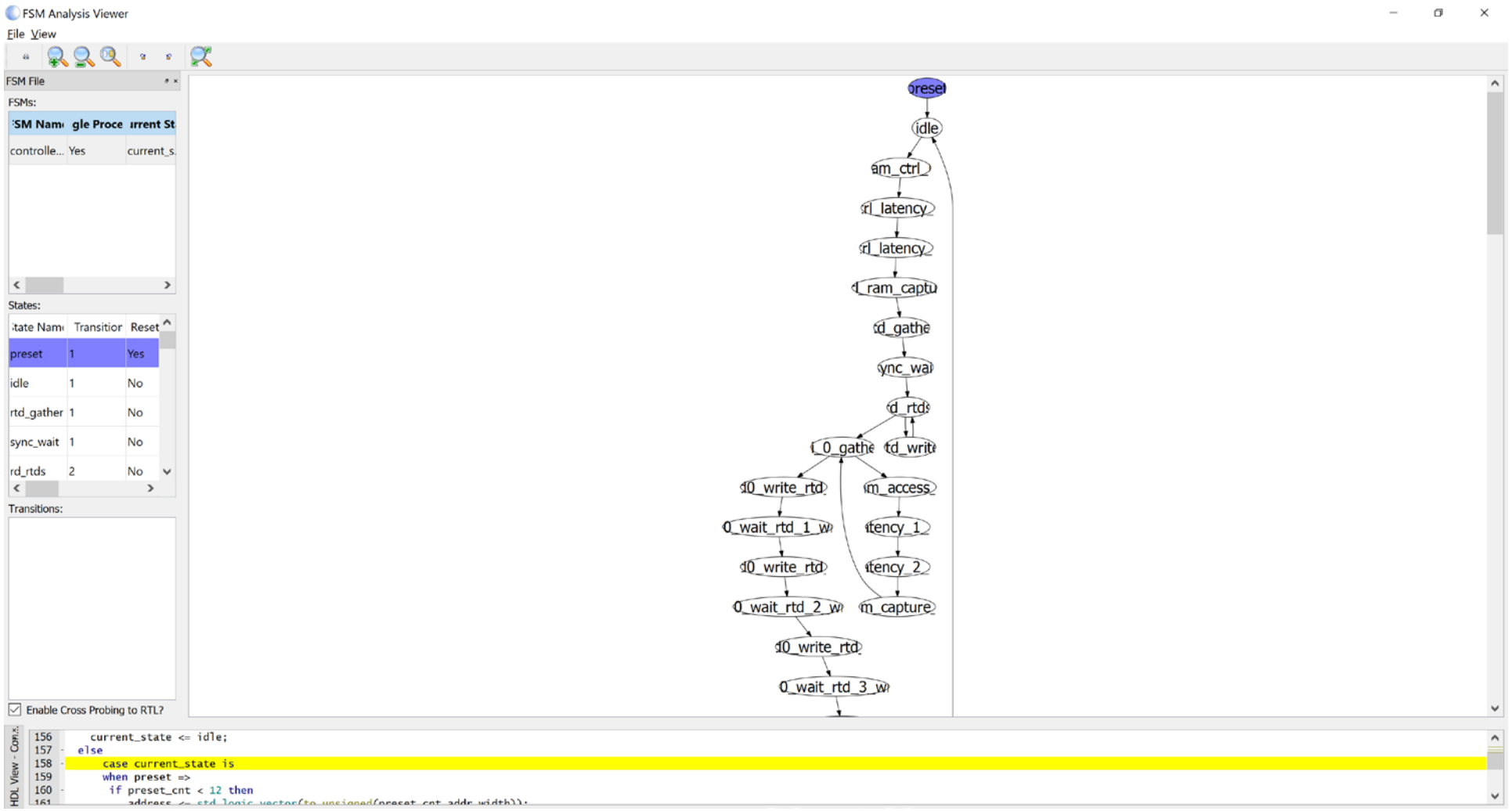
The Unmapped State Machine



FSM Analysis



FSM Analysis In Detail



FSM Analysis

How can we harden state machines?

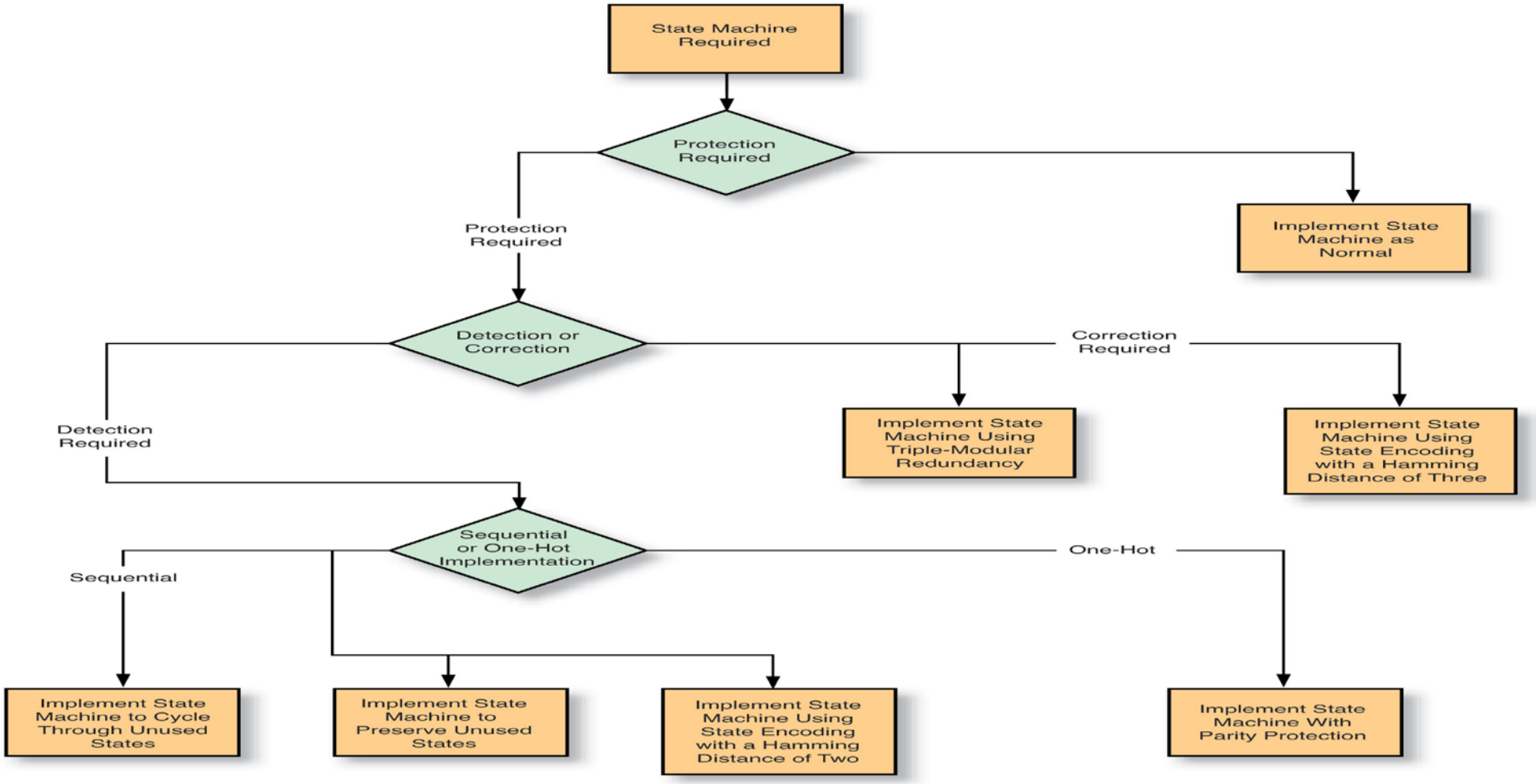
Triple Modular Redundancy – Instantiate three implementations of the state machine and majority vote on the output. This obviously has an area penalty and can reduce the operating frequency. There are tools like XMR from Xilinx which will assist with inserting TMR.

TMR also requires that the instantiations are separated physically from each other to ensure more than one instantiation cannot be effected by a SEU and create a MBU.

What other options are available ?

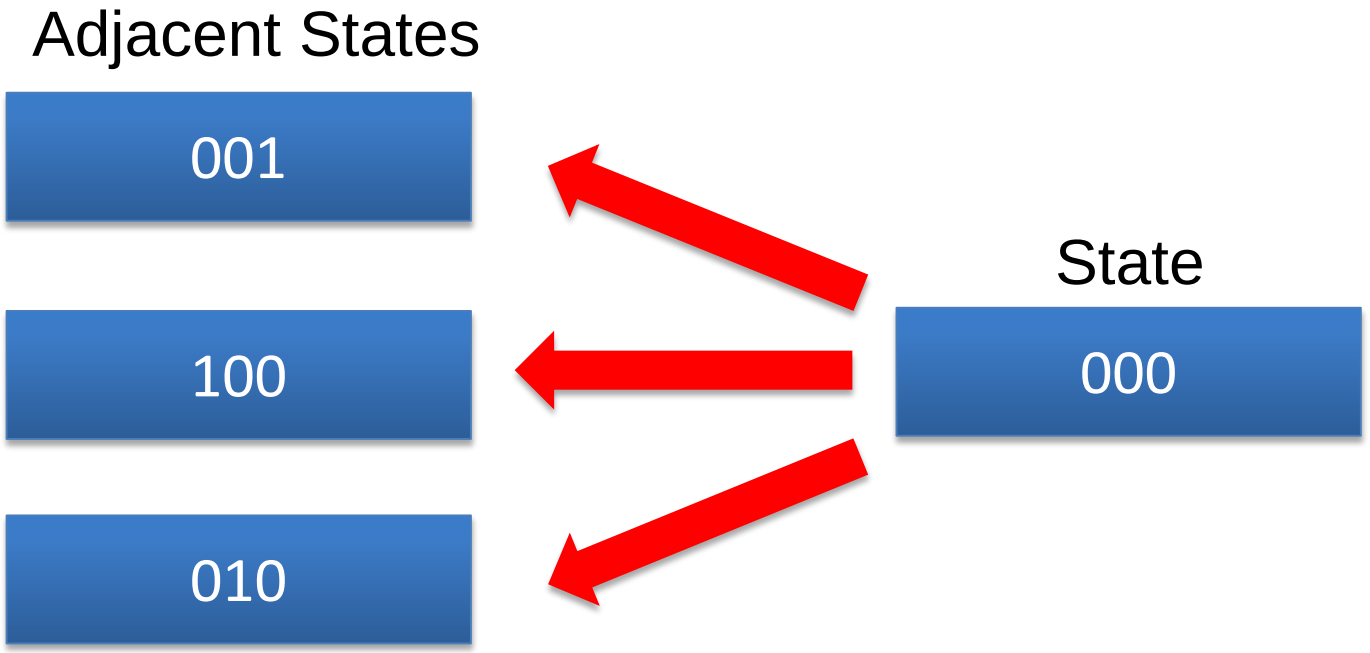
FSM Analysis

State machine choices other than TMR



FSM Analysis

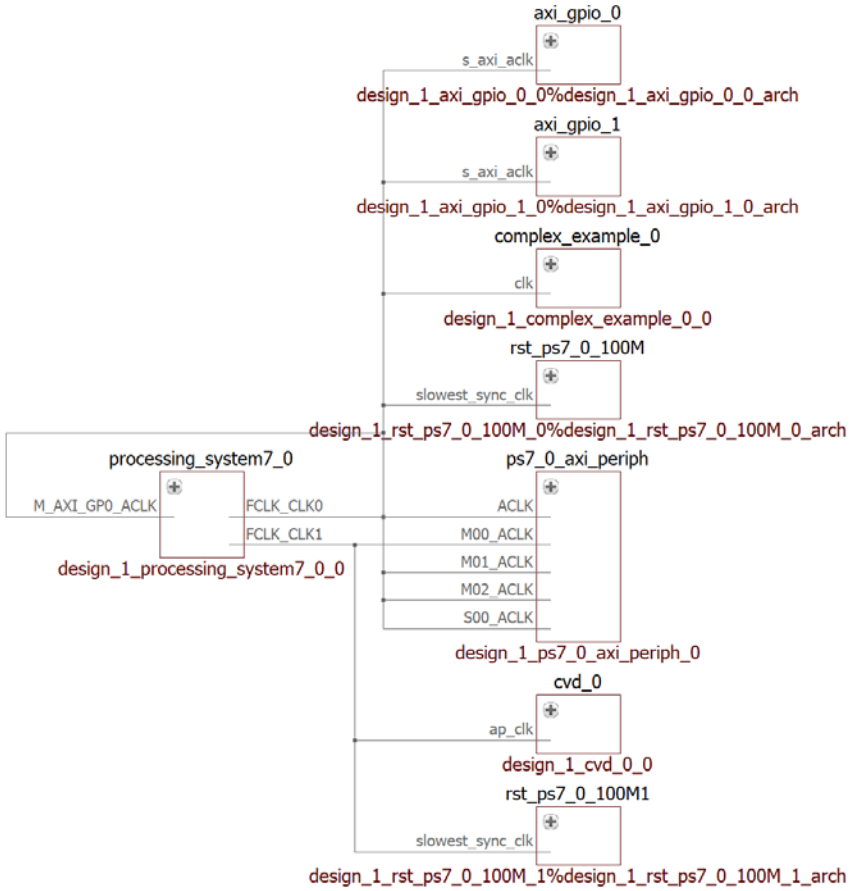
- What does a Hamming code of three mean?
- For each state with n number of bits there are also n adjacent states

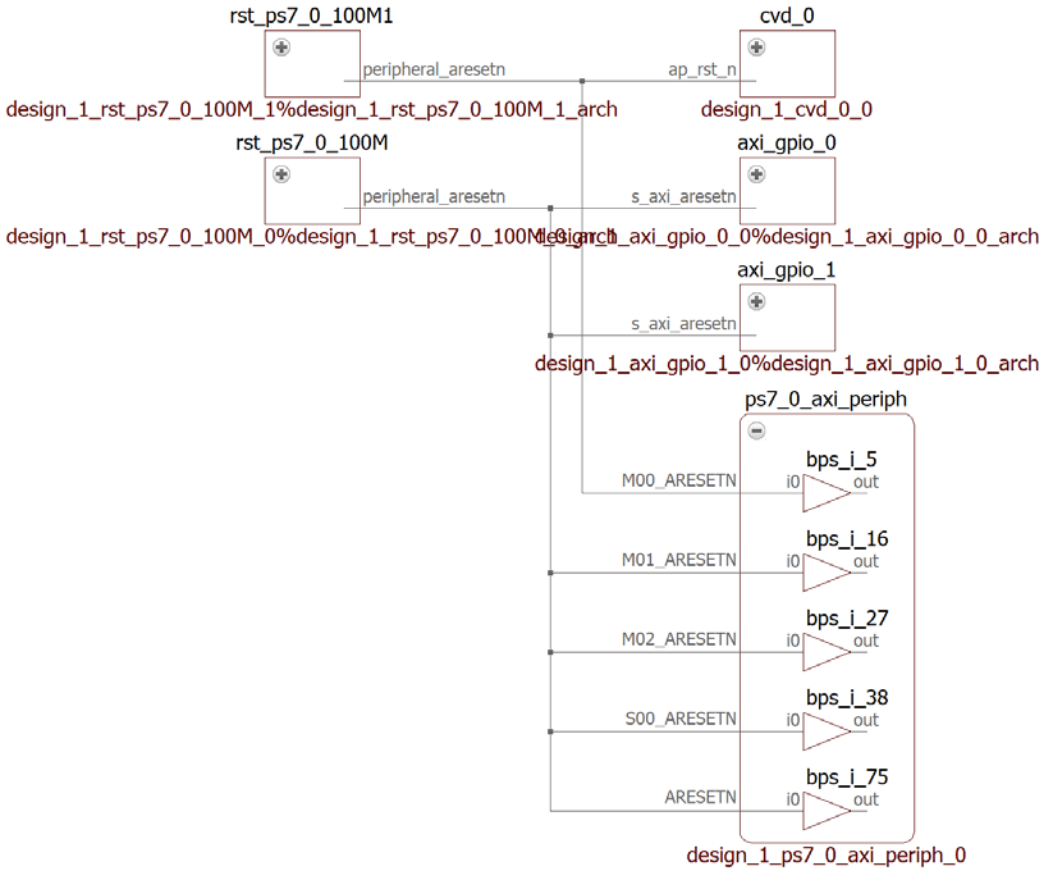


CDC Analysis

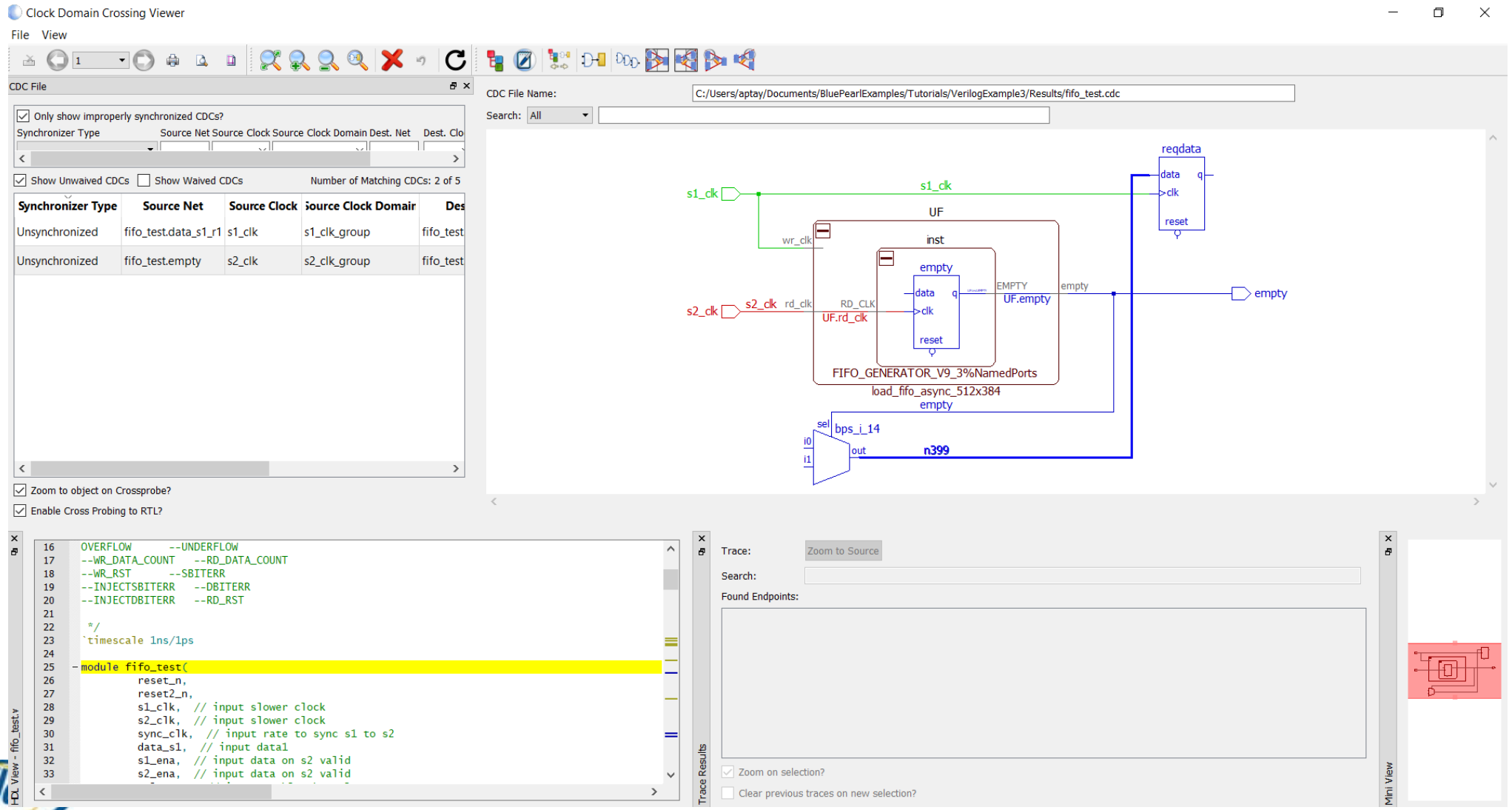
- According to the Wilson Group Survey [5] 90% of FPGA designs include more than one clock domain.
- In satellite applications it is easy to have 3 or 4 asynchronous clock domains as shown in the different satellite applications.
  - Transparent Telecommunications Processor – Four Clock Domains: ADC Clock Domain, DAC Domain, Communication Domain and Data Processing Domain. All of which will be asynchronous due to the often source synchronous nature of ADCs and DACs
  - Image Processing Processor – Three Clock Domains: Pixel Clock Domain, Processing Clock Do-main and Communication / Downstream link Domain.
- Manual design reviews which operate on module level reviews or interface level reviews for multiple blocks can often fail to spot CDC issues as identified by the ESA FPGA lessons learned task force.

Clocking Domains





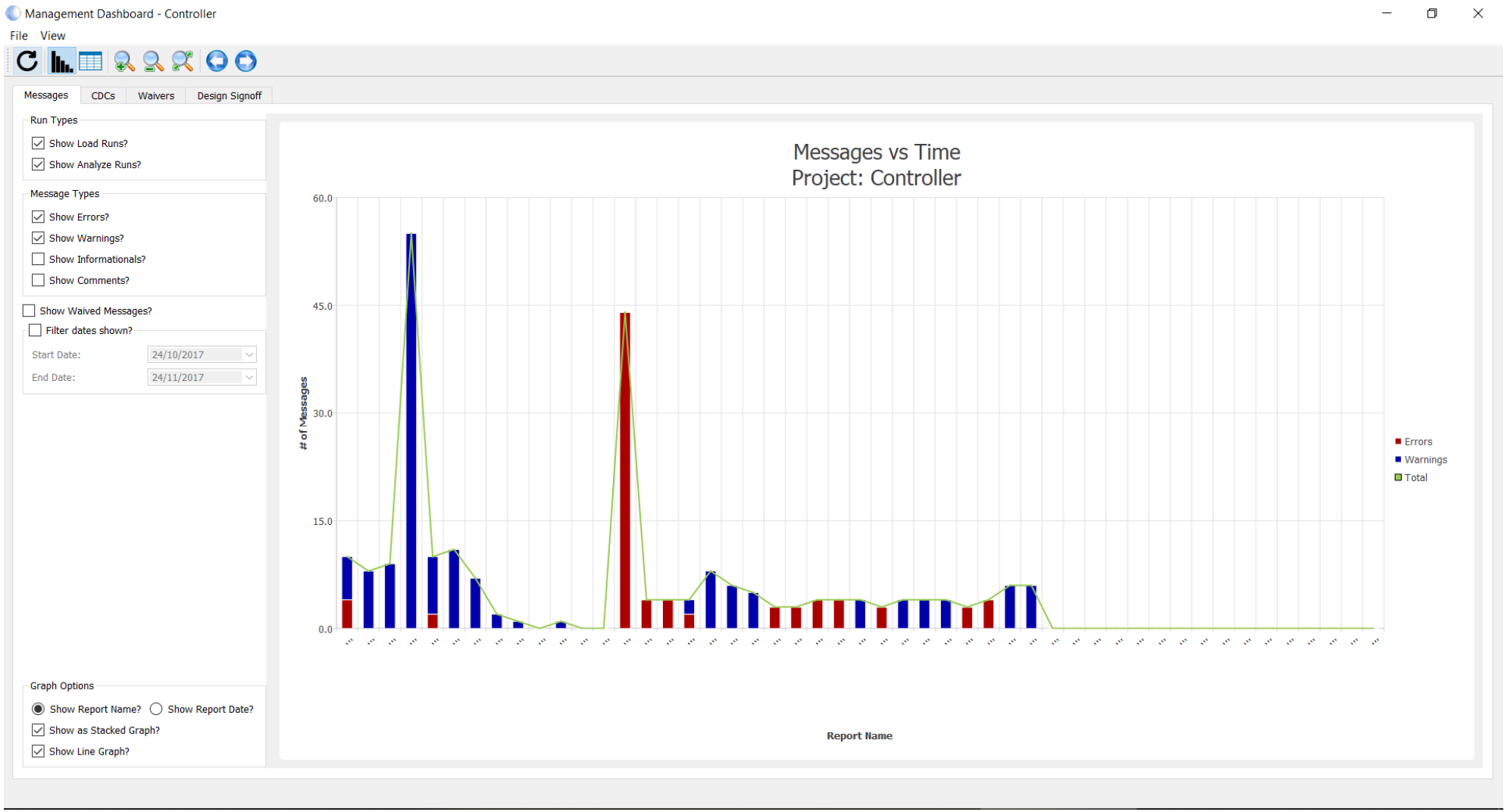
CDC Analysis



Metrics & Maturity

- Satellite development flow consists of many engineering reviews e.g PDR, CDR
- Each stage needs to demonstrate not only the maturity of the design but also that the design has been subjected to review.
- Collating this evidence can be a time-consuming task as individual review reports have to be collated

Metrics & Maturity



Wrapping it up

- There are several challenges which occur when developing a FPGA for satellite applications as demonstrated by ESA and NASA lessons learned on failures in orbit.
- Using a structural RTL design analysis tool such as Blue Pearl's VVS enables issues to be identified earlier in the design process, while also generating the necessary evidence of the design review completion.
- Using a structural RTL analysis enables a better quality of RTL code going forward for Simulation and Synthesis.