



Static Analysis

Adam@adiuvoengineering.com

www.adiuvoengineering.com



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

Lessons Learned from FPGA Developments - 2002 – S Habinc

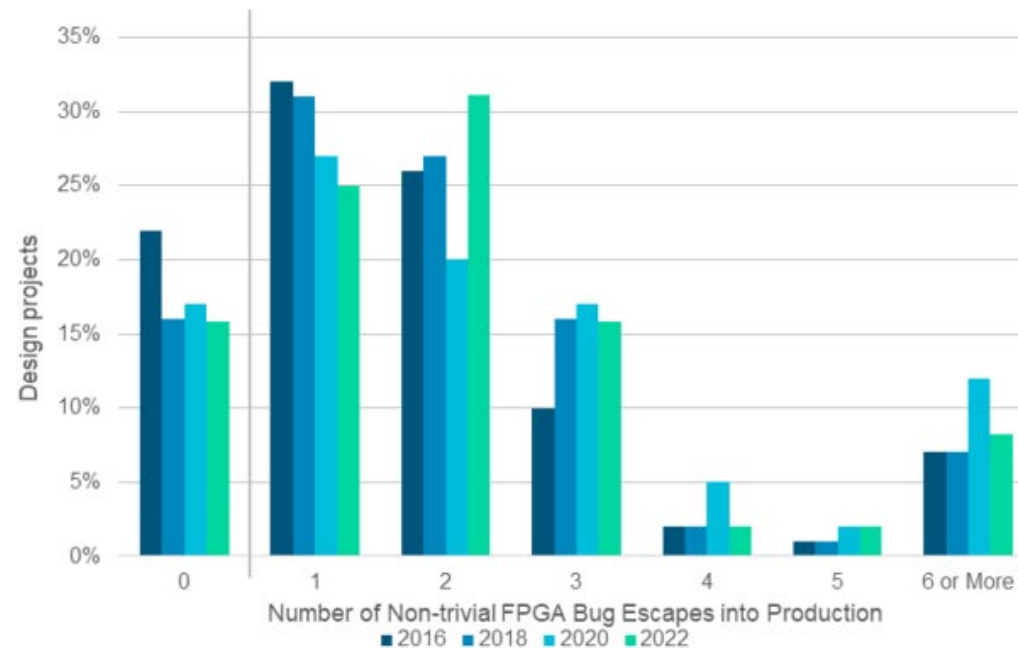


ADIUVO
ENGINEERING AND TRAINING, LTD.

We are Pretty Bad at Designing FPGAs

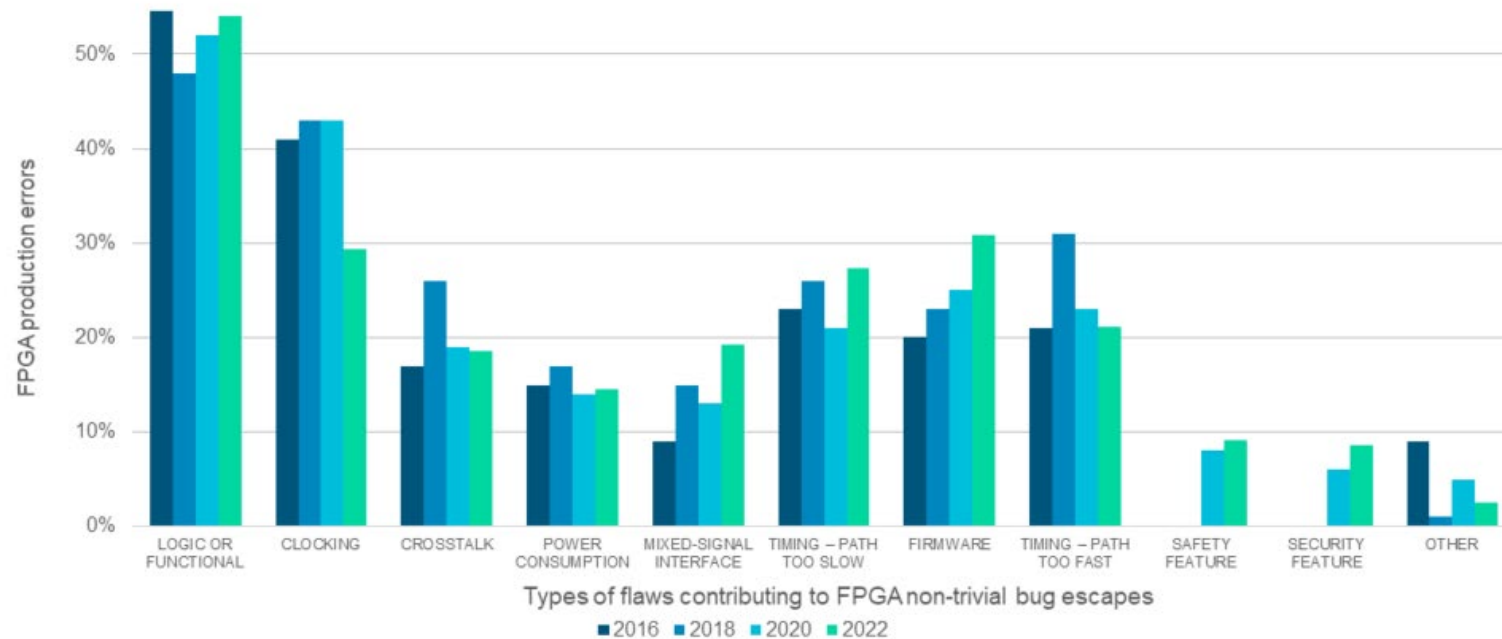
FPGA projects are performing no better than ASIC

84%
 FPGA design projects have non-trivial bug escapes into production



Types of Errors we see

Causes of FPGA non-trivial bug escapes into production
 Logic/functional failures consistently the top cause of FPGA non-trivial bug escapes



Source: Wilson Research Group and Siemens EDA, 2022 Functional Verification Study

Page 6 Unrestricted | © Siemens 2022 | Functional Verification Study

* Multiple replies possible



Why is this

Several Reasons

- Incomplete / Poor / Fluid Requirement specifications
- Project time scales - We are expected to do more faster
- Poor Process – Ill defined process to go from beginning to end
- Investment in tools – Reliance on free versions of tools, simulators etc.
- We are human bugs happen – The key is to find them early, and efficiently

What is Static Analysis

Analyses RTL, to find issues in minutes which might take hours in Synthesis and Simulation.

Done correctly it enables a better quality of code for simulation and synthesis.

Adiuvo uses Blue Pearls – Visual Verification Suite

What is Blue Pearl's Visual Verification Suite™

- Analyze RTL™ linting and debug –
 - This enables us to ensure coding standards are followed, analyse the design for several structural issues, verify the design complies with manufacturer coding standards e.g. Xilinx Ultrafast Methodology, run path analysis and explore finite state machines.
- Clock Domain Crossing –
 - Analyze design and clocking structures to ensure CDC is implemented correctly in the design.
- Automated SDC generation –
 - Identified false paths and multi-cycle paths in your design and generated the appropriate SDC file.

VVS Workspace

Design
Analysis
Settings

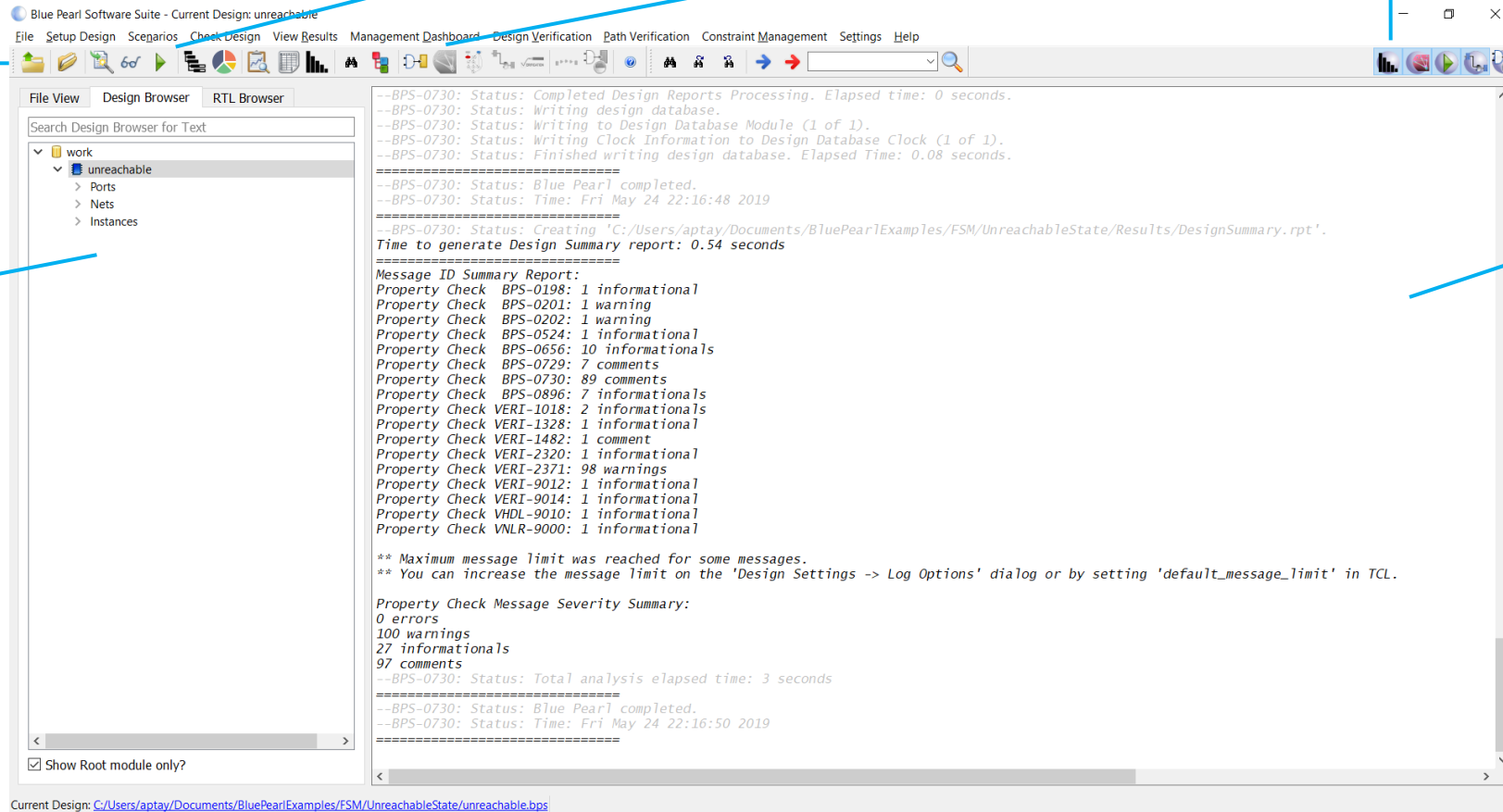
Visual
Analysis
Environments

Checked
Out
Licenses

Tool
Menu

Messages

Design
Browser



The screenshot displays the Blue Pearl Software Suite interface. The top menu bar includes File, Setup Design, Scenarios, Check Design, View Results, Management Dashboard, Design Verification, Path Verification, Constraint Management, Settings, and Help. The Design Browser on the left shows a tree view with 'work' and 'unreachable' folders, containing sub-items like Ports, Nets, and Instances. The Messages panel on the right displays a log of design verification results, including status messages and a detailed Message ID Summary Report. The summary report lists various checks with their counts and severities, such as 'Property Check BPS-0198: 1 informational' and 'Property Check VERI-2371: 98 warnings'. The status at the bottom indicates 'Current Design: C:/Users/aptay/Documents/BluePearlExamples/FSM/UnreachableState/unreachable.bps'.

Adiuvo / ESA Project

Multi Year project to analyse IP cores using static analysis tool Blue Pearl Visual Verification Suite.

- 2020 – Creation of coding rules and analysis circa 50% of IP cores, ESA

Training

- 2021 – Analysis of remaining blocks
- 2022 – Reflect changes on identified block, rerun verification – training of ESA Engineers in basic and advanced BPS.

ESA BPS Coding Rules

Activity Started in 2020 to identify Set of coding rules which can be applied by ESA to its IP cores

Some challenges – ESA is unique several organizations delivering RTL code

Diverse presentation standards e.g camelCase, PascalCase, Snake_Case etc.

Presentation standards – Not as important

Coding Standards are however important - These ensure quality of the actual design

ESA BPS Coding Rules

Creation of coding rules pulled from

- ECSS-Q-ST-60-02C
- CNES - DESIGN AND VHDL HANDBOOK FOR VLSI DEVELOPMENT
- ESA IP - ESA IP Core Technical Requirements
- ESA Model - VHDL Modelling Guidelines
- Realtra Coding Rules

Coding Rules covering

ESA BPS Coding rules cover

- FSM Unreachable States
- No Initialization statements
- Ensuring all Flip Flops are Set / Reset
- Input and Output Registers being used
- Unconnected and undriven pins and nets
- Identifying use of Hard Coded Constraints
- Incomplete Sensitivity Lists
- CDC Incorrectly performed
- Unconstrained INT
- Latch Creation
- Using RE and FE edges of the clock

ESA Coding Rules

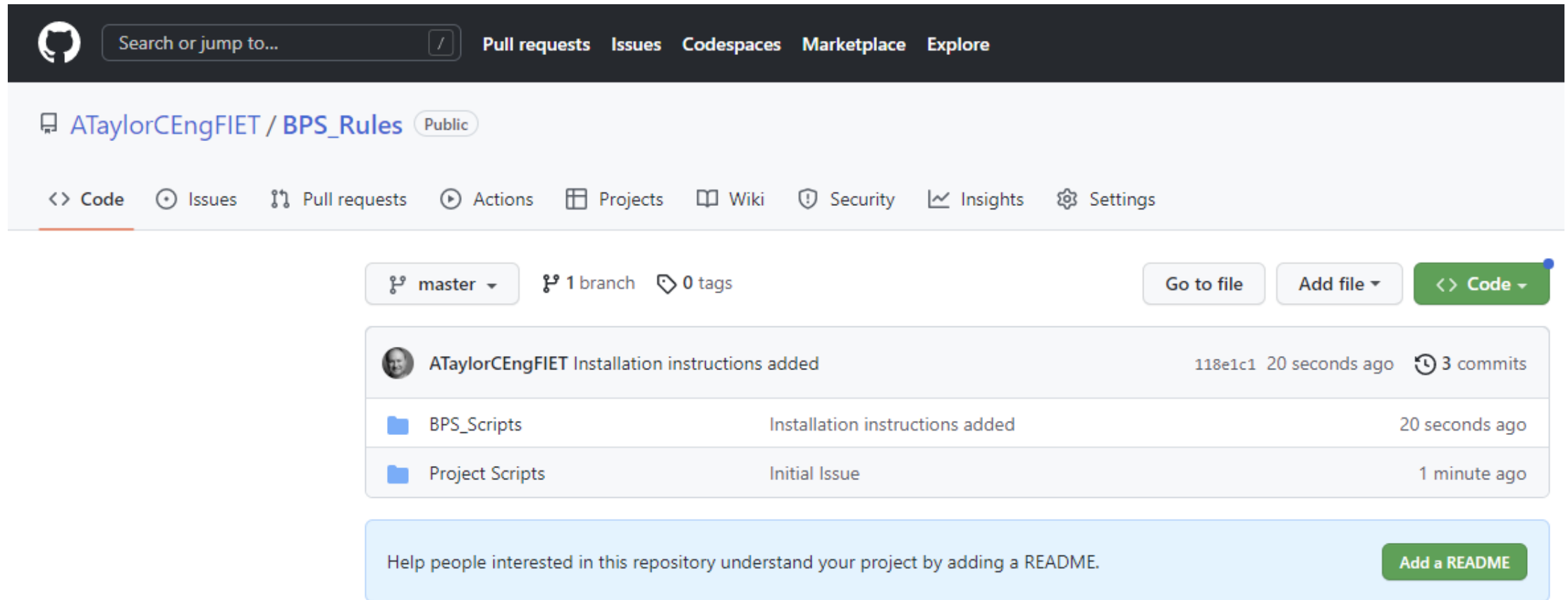
Collated into a TCL script for use with BluePearl Visual Verification Studio

Available here https://github.com/ATaylorCEngFIET/BPS_Rules

Additional information includes

- How to install the BPS Rules
- How to use the Scripts and Batch files to analyse a Project

BPS Rules






The screenshot shows the GitHub interface for the repository 'ATaylorCEngFIET / BPS_Rules'. At the top, there is a search bar and navigation links for Pull requests, Issues, Codespaces, Marketplace, and Explore. Below the repository name, there are icons for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The main content area shows the current branch 'master', 1 branch, and 0 tags. There are buttons for 'Go to file', 'Add file', and 'Code'. A commit history table is visible, showing a recent commit by ATaylorCEngFIET with the message 'Installation instructions added'. Below the commit history, there is a blue box with the text 'Help people interested in this repository understand your project by adding a README.' and a button 'Add a README'.

Search or jump to... / Pull requests Issues Codespaces Marketplace Explore

ATaylorCEngFIET / BPS_Rules Public

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags Go to file Add file <> Code

 ATaylorCEngFIET	Installation instructions added	118e1c1 20 seconds ago	🕒 3 commits
 BPS_Scripts	Installation instructions added	20 seconds ago	
 Project Scripts	Initial Issue	1 minute ago	

Help people interested in this repository understand your project by adding a README. Add a README

Modules Analyzed

2020 Batch Components

- EDAC
- FTADR
- HurriCAN
- Leon
- PTME
- SHYLOC 121
- SHYLOC 123
- SPACE FIBRE
- SPW

2021 Batch Components

- AUIP
- EDAC-07
- NoCIP
- Pdec
- PTCD
- SCCC
- SPW RMPA
- SPW Node



As would be expected with IP cores already verified no MAJOR issues were found however there were some minor issues.

Main Issues

- Unbound generics
- Reset not synchronously de-asserted
- Arithmetic Overflow Warnings
- Clock Synchronizer Warnings
- Combination feedback loop created
- Missing bit ranges
- Missing if then else, check no latches are created
- Update design to include all registers being capable of being reset – this prevents reliance on unknown register states at power up which could lead to undefined behaviour

Main Issues

- Update source code to remove hard coded constants from the source files to ensure the code is more readable.
- Correct FSM implementations to behave correctly in SEE environment.
- If-then-else statements too deep – Can impact timing
- Mixed edge clocking – check to ensure no timing issues
- Signals used but no driver – check how these signals are used to ensure this isn't an issue

Example of FSM Issue Found

FSM Analysis Viewer

File View

FSM File

FSMs:

FSM Name	gle Proc	irrent Sta	Next State	reset Stat	if of State	Language
dcache_default(rt)(r.dstate)	Yes	dstate			8	VHDL
dcom_default(struct)(r.state)	Yes	state			7	VHDL
dcom_uart_default(rt)(r.tstate)	Yes	tbstate			4	VHDL

States:

State Name	Transition	Reset?	Terminal?	reachable?	Missing Case Item?	Default State?
dtag	6	No	No	No	No	No
001	4	No	No	No	No	No
ddata	0	No	No	No	Yes	No

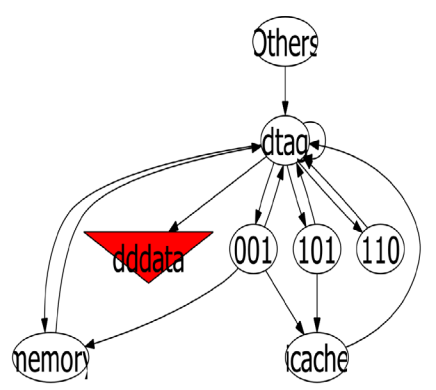
Transitions:

Idle -> pre

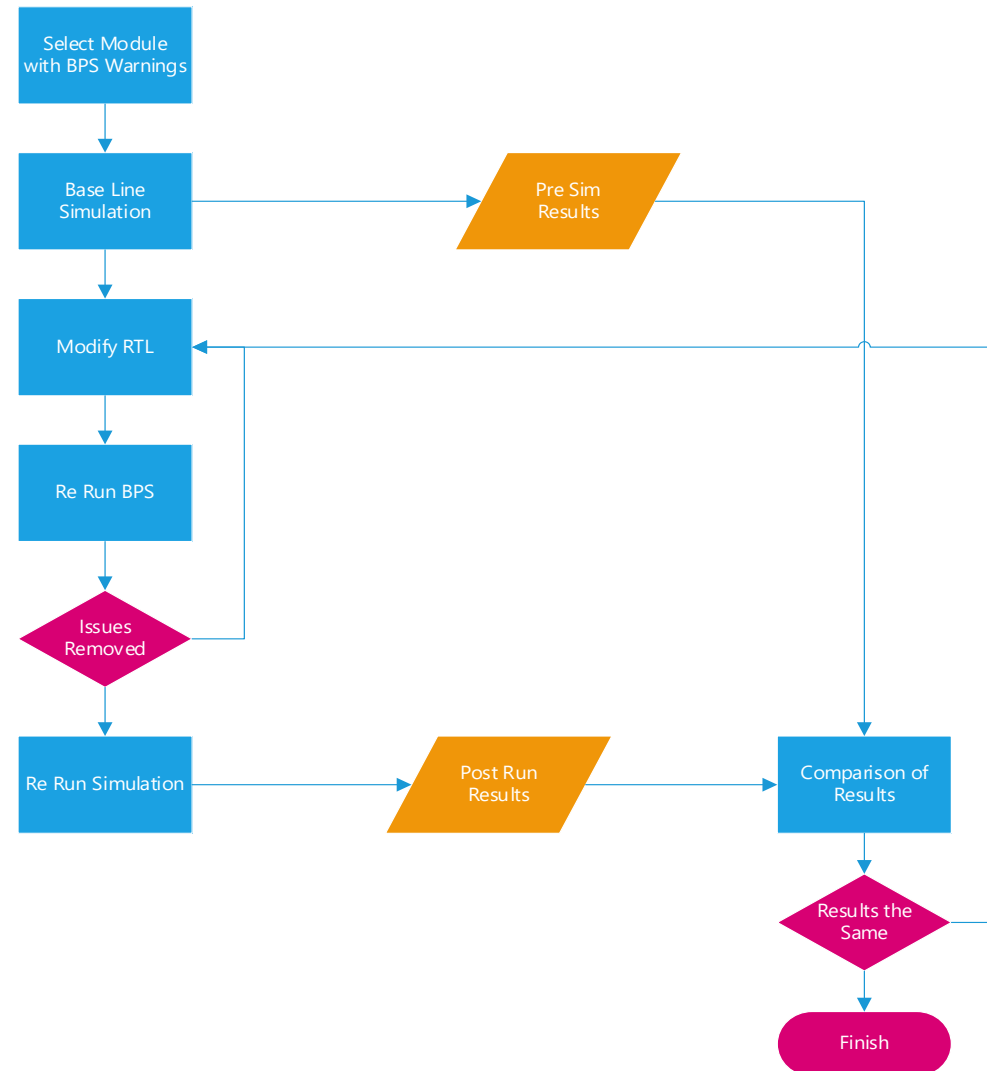
Enable Cross Probing to RTL?

```

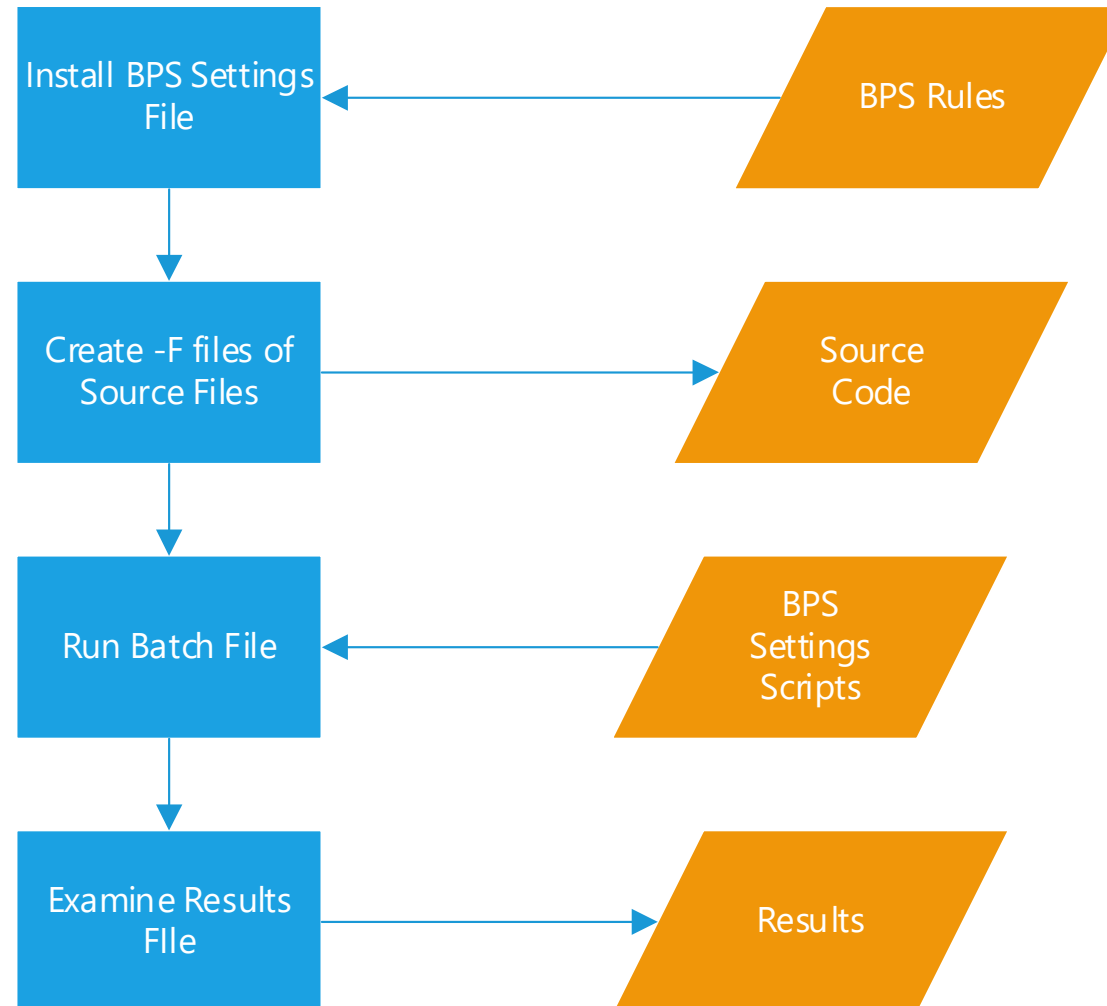
498   if r.stpend = '1' then
499     v.stpend := r.req; v.wb.data1 := r.wb.data2;
500     v.wb.lock := r.wb.lock and r.req;
501   end if;
502 end if;
503 if mcd0.grant = '1' then v.req := r.burst; v.burst := '0'; end if;
504
505 -- main Dcache state machine
506
507 case r.dstate is
508 when "000" => -- Idle state
509   v.nonds := r.nonds and not eholdn;
510   if (snoopval = '1') then v.valid := dcrmov.dramout(set).valid;
511   else v.valid := (others => '0'); end if;
512   if (r.stpend = '0') or ((mcd0.ready and not r.req) = '1') then -- wait for store queue
513     v.wb.addr := dci.maddress; v.wb.size := dci.size;
514     v.wb.read := dci.read; v.wb.data1 := dci.edata; v.wb.lock := dci.lock;
515     v.wb.as1 := dci.as1(3 downto 0);
516   end if;
  
```



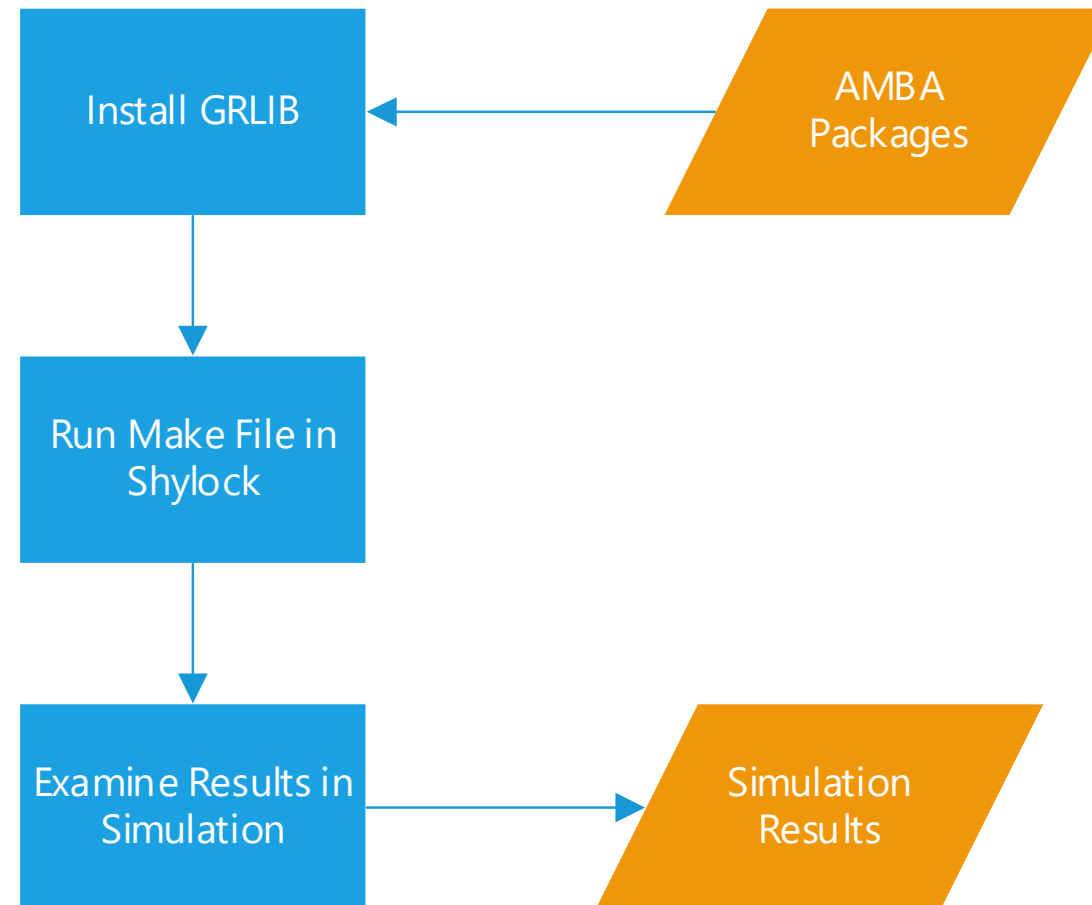
Module Correction Flow



Project Flow



Configuring the Simulation



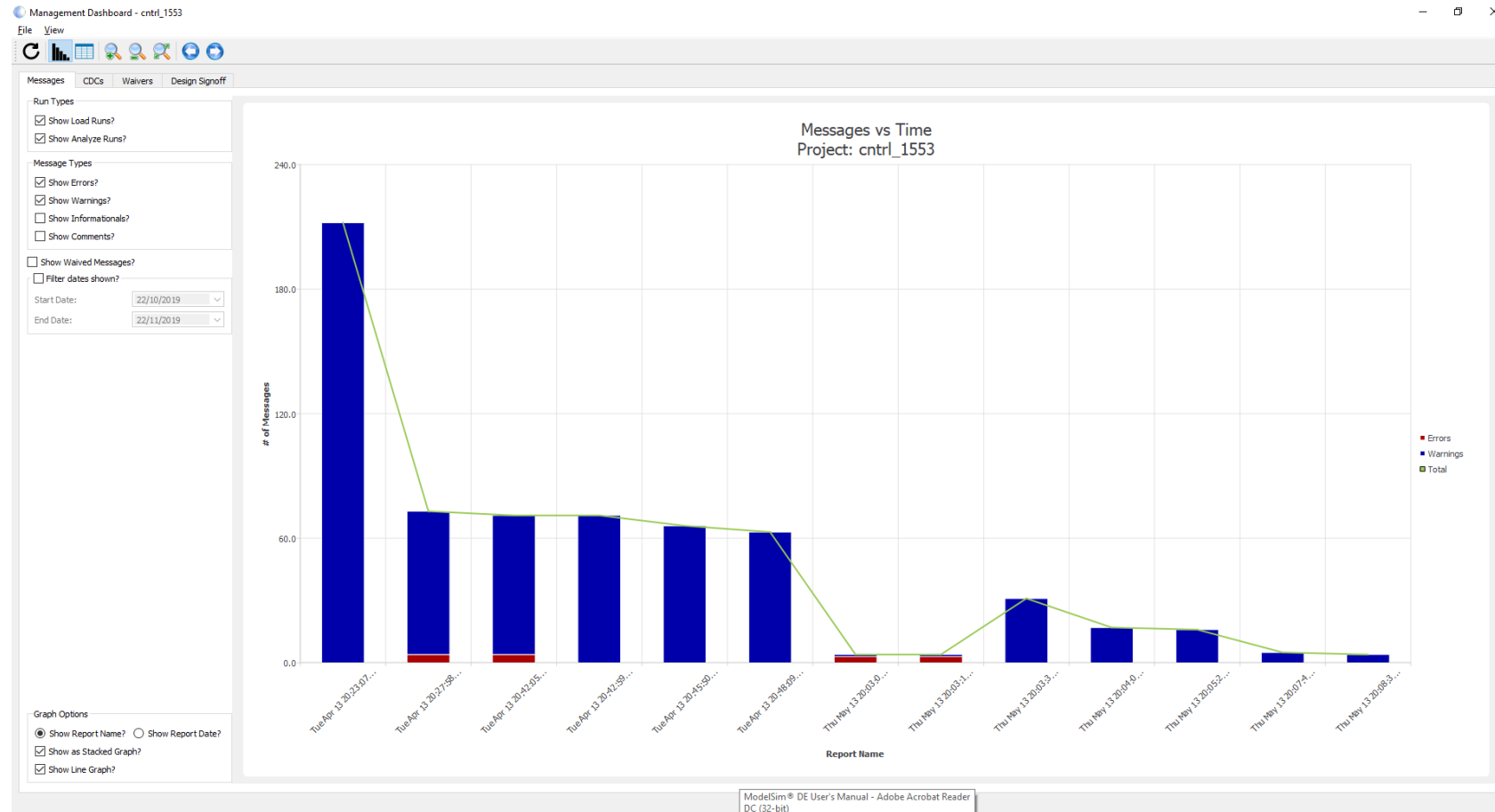
Adiuvo Plato

Adiuvo Design Three FPGA for ESA Plato Mission PLIU

- CTRL – Platform Controller - Manages 10,000 Registers and 54 PID Loops Circa 20K SLOC
- RTD – Ultra Low noise FPGA / ADC – Game Changing Accuracy & Stability! Circa 15K SLOC
- Heater – Low Noise High Power Heater drive

Static Analysis with agreed ESA rules used across all the modules prior to simulation – Full code coverage and synthesis.

Adiuvo Plato



Adiuvo Plato

All warning removed from code except for some which are waived

- If Then Else Depth - Lucky Running at 6 MHz
- Internal Clock Warning - None PLL divide by 2.5
- Tri State – Warning on MRAM Interface
- IO Types used - I2C

Demonstrates BPS is finding issues as the code is developed and the few warning not corrected are waived after analysis. And presented to ESA in verification plan with justification.

Adiuvo Gateway

Adiuvo Developing FPGA for ERSA CDHU – Circa 30K lines of RTL

Static Analysis was run on the modules against the ESA coding rules –
Integrated as part of the CI Suite

Each evening code in repo checked against results

Adiuvo Model Based

Adiuvo Recently Pioneered model-based FPGA design targeting MicroChip PolarFire using a newly developed tool.

Developed using model-based flow – SysML to RTL

- RTL Generated is vendor independent – Includes AXI Network instantiation
- Circa 100K lines of code
- Massively reduced development time & right first time – come see demo

Next challenge is to run the ESA Rule set against the output code – Optimistic as we wrote the tool it will work as expected but will report back next year!

Conclusion

Static Analysis great tool in the toolbox to help enforce good coding practice.

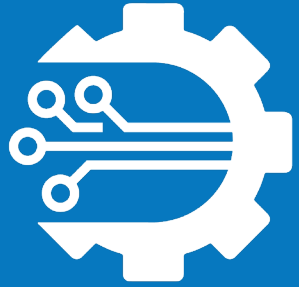
Enables entry into Simulation with a better code quality

Reduces development time (but hard to quantify) – When implemented in CI flow beneficial as just view reports no need to remember to do it



Jenkins





ADIUVO

ENGINEERING AND TRAINING, LTD.

www.adiuvoengineering.com



adam@adiuvoengineering.com