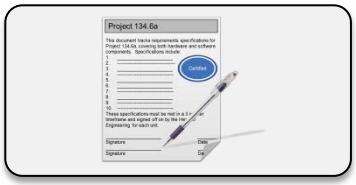
SYNOPSYS®
Predictable Success

# Achieving Safety Critical Designs With FPGAs

Bob Efram/Tom West
Synopsys

# 5 Challenges



## Complete and Accurate Design Specification

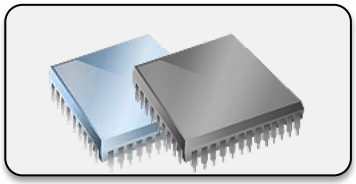
Early synthesis reports.
Constraint checking,
Clock domain cross checks

## Built in Design Reliability

Preserving critical design logic.
Safe Finite State machines
Triple Module Redundancy

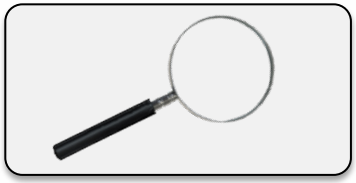
## Verifying the Design Meets Specification.

Design visualization.
Debug with RTL/Gate level Simulation.
Debug on-chip implementation.



## Reproducible, Documented Design Process

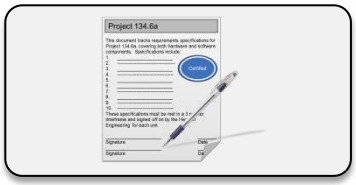
Documentation
Revision Control



## Proof of Concept and Hardware-Based Validation.

Prototyping boards.
High Speed Interfaces.

# 5 Challenges



## Complete and Accurate Design Specification

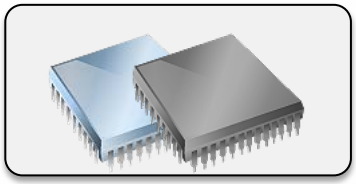
Early synthesis reports.
Constraint checking,
Clock domain cross checks

## Built in Design Reliability

Preserving critical design logic.
Safe Finite State machines
Triple Module Redundancy

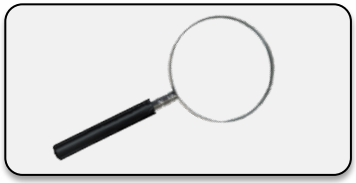
## Verifying the Design Meets Specification.

Design visualization.
Debug with RTL/Gate level Simulation.
Debug on-chip implementation.



## Reproducible, Documented Design Process

Documentation
Revision Control



## Proof of Concept and Hardware-Based Validation.

Prototyping boards.
High Speed Interfaces.

# Early Synthesis Reports to Verify Specification Accuracy.

- A complete specification defines the design:
  - Functionality and performance
  - Physical requirements: Device, package size, power.
- Without RTL, correlating the physical requirements with the functional requirements is difficult.
- Simulation tools verify functionality of RTL.
- Early synthesis reports provide:
  - Early information on size, performance, and area.
  - Early information on constraints.
  - Early information on clock domain synchronization.

# Area and Timing Reports

## Verify Area and Timing of RTL blocks as they are completed.

Block Area Report

I/O ports: 51

Register bits: 99 (0%)

RAM/ROM usage summary
Single Port Rams (RAM256X1S): 16

Global Clock Buffers: 2 of 32 (6%)

Total LUTs: 151 (0%)

| Starting Clock | Frequency | Frequency | Period | Period | Slack |
|---|---|---|---|---|---|
| dll_clk_1x_derived_clock | 100.0 MHz | 243.7 MHz | 10.000 | 4.104 | 5.896 |
| pin_clk | 200.0 MHz | 200.0 MHz | 5.000 | 5.000 | 0.000 |

# Constraints Checker

Checks constraint syntax and for timing constraints applied to non existent or invalid types of arguments/objects

##### SUMMARY ##################################################################

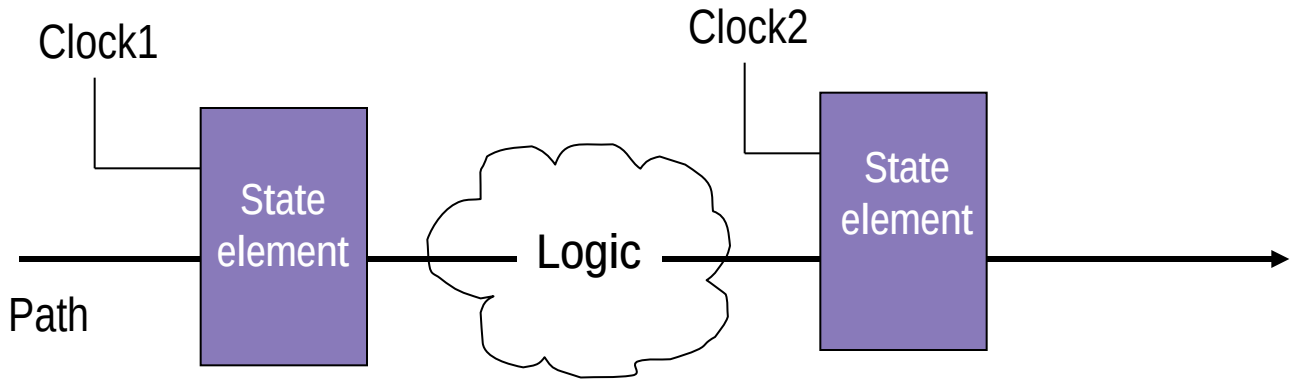
Found 1 issues in 1 out of 37 constraints

Inapplicable constraints
**********

define_clock { clk2 } -name { clk2 } -freq { 100 } -clockgroup { default_clkgroup_3 }
    @E:"C:\bus_demo\bus_demo_haps_lx330_identify.sdc":1:0:1:0|object "clk2" does not exist

# Accurate Design Specification
# Clock domain synchronization assurance

Find combinatorial paths that cross clock domains without synchronization

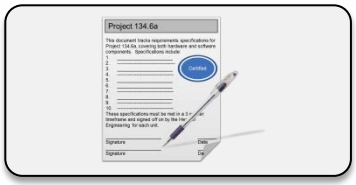



Clock1 and Clock2 controlled by clocks in <u>different</u> clock domains
i.e. are in different clock groups

- Generates **report** for all paths that:
  - Start at state element in one clock domain (clock group)
  - End at state element in different clock domain (different clock group)

  *Reports longest path between these 2 points*
- User can then appropriately synchronize the path if synchronization was intended

# 5 Challenges



## Complete and Accurate Design Specification

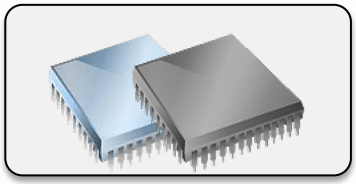
Early synthesis reports.
Constraint checking,
Clock domain cross checks

## Built in Design Reliability

Preserving critical design logic.
Safe Finite State machines
Triple Module Redundancy

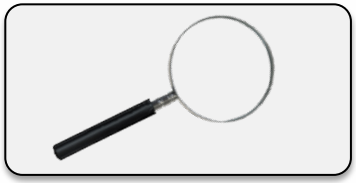
## Verifying the Design Meets Specification.

Design visualization.
Debug with RTL/Gate level Simulation.
Debug on-chip implementation.



## Reproducible, Documented Design Process

Documentation
Revision Control



## Proof of Concept and Hardware-Based Validation.

Prototyping boards.
High Speed Interfaces.

# Preserving Parts of the Design from Optimization

## Maintain critical logic

- Synthesis will, by default, optimize the design to meet timing and then reduce area by
  - Collapsing nets, Dissolving Hierarchies
  - Removing duplicate registers and instances with unused outputs
- Use synthesis attributes to preserve
  - Redundant logic that you want to maintain for reliability purposes
  - Specific signals that you wish to be able to probe
  - FSM Error mitigation logic

| Attribute | Value | Description |
|---|---|---|
| `syn_keep` | 1/0 | Preserve a net |
| `syn_preserve` | 1/0 | Preserve a cell / sequential component |
| `syn_hier` | firm, hard, macro, flatten | Preserve a block |
| `syn_noprune` | 1/0 | Preserve an instantiated component (Instance) |

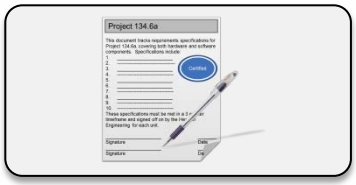
# Safe State Machines.

- Synthesis tools are very good at optimizing FSMs for performance (FSM Compiler).
  - Re-encoding state-bits (ie one-hot)
  - Removing unreachable states. (ie the default/others clause)
- Solutions:
  - Use the attributes to preserve all logic and manually determine the optimal FSM encoding.
  - Current attributes for automatic  safe FSM optimization.
    - syn_encoding=safe – Drives FSM to reset state for illegal states. Best for FSM performance but limited for customization.  Used for 10 years for space based designs.
  - New automatic safe FSM  attributes
    - Automatic preservation of  the default/others clause. Allows customization of error detecting/correcting.  May affect performance depending on others clause logic.
    - Hamming error detection and correction.  Automatically detect and correct for illegal states and transitions.  Used for slow FSMs due to correction logic overhead.

# TMR - Triple Mode Redundancy

- Synthesis tools are very good at optimizing away redundant logic.
  - Replicated logic in the RTL may be removed by synthesis.
- Solutions:
  - Use the attributes to preserve all logic and manually determine the optimal FSM encoding. Syn_preserve, syn_keep, syn_noprune.
  - Current attributes for automatic TMR optimization.
    - Syn_Radhardlevel = tmr – Local TMR . Replicates sequential logic and inserts voters. MicroSemi today.
  - New automatic TMR.
    - Local TMR for Xilinx. Triplicates sequential logic and inserts voters.
    - Distributed TMR for Xilinx. Triplicates combinational and sequential logic and inserts voters.
    - Block based TMR for Xilinx. Triplicates RTL blocks and inserts voters.

# 5 Challenges



## Complete and Accurate Design Specification

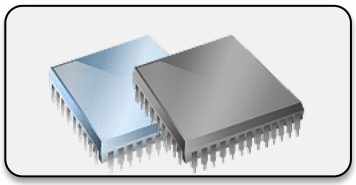
Early synthesis reports.
Constraint checking,
Clock domain cross checks

## Built in Design Reliability

Preserving critical design logic.
Safe Finite State machines
Triple Module Redundancy

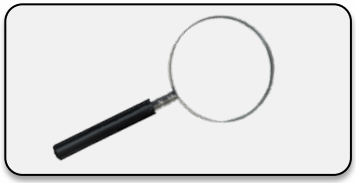
## Verifying the Design Meets Specification.

Design visualization.
Debug with RTL/Gate level Simulation.
Debug on-chip implementation.



## Reproducible, Documented Design Process

Documentation
Revision Control



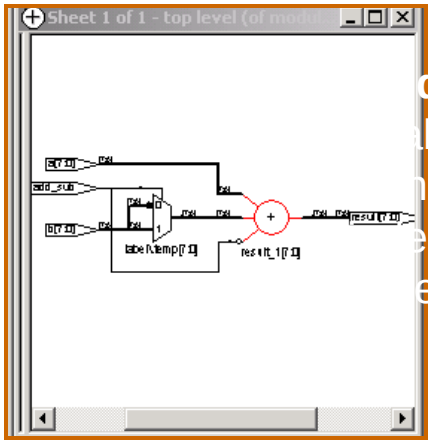
## Proof of Concept and Hardware-Based Validation.

Prototyping boards.
High Speed Interfaces.

# Integrated Crossprobing Between HDLAnalyst Views and Source

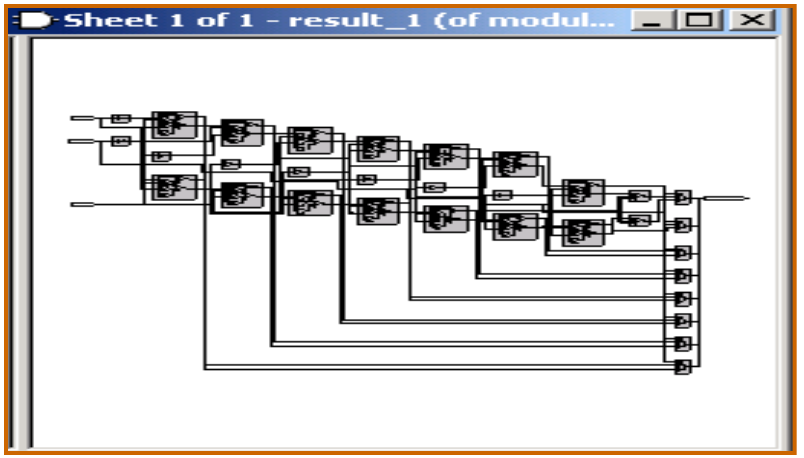
misc\resrcshr.v (verilog)

```
00031     // declarations. I declared temp.
00032     reg [7:0] temp;
00033     if (add_sub)
00034         temp = b;
00035     else
00036         temp = ~b;
00037     result = a + temp + !add_sub;
00038 end
00039
00040 endmodule
00041
```




Starting Points with Max Worst Slack
**********

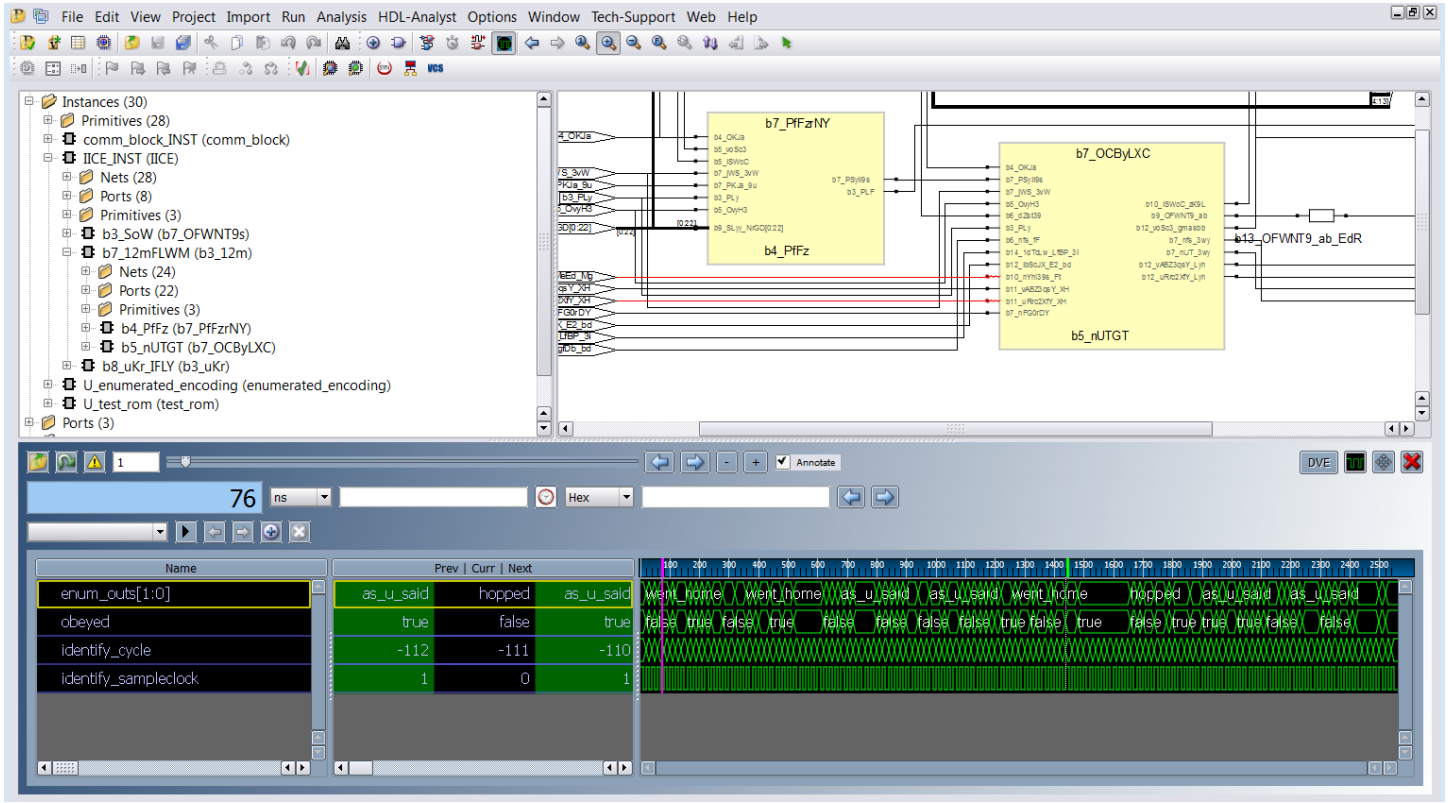
| Instance | Pin | Arrival Time | Slack | Starting Reference Clock |
|---|---|---|---|---|
| Type | | | | |
| crossbar.wb_conmax_top.s2.msel.arb0.state[1] | | | | sys_clk |
| SDFFRX2 | Q | 0.589 | 0.374 | |
| usb2.u4.csr[27] | | | | phy_clk_2 |
| SDFFQX2 | Q | 0.678 | 0.375 | |
| crossbar.wb_conmax_top.s3.msel.pri_out[0] | | | | sys_clk |
| SDFFQX1 | Q | 0.904 | 0.376 | |




annotated timing

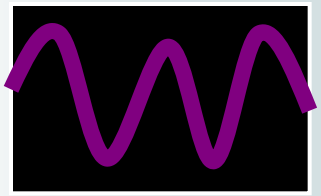
# RTL Simulation

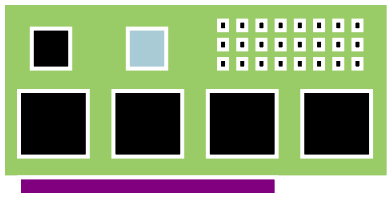
- Best visibility for correlation to the specification
- Slow, relative to FPGA speeds.
- Require comprehensive testbench and code coverage.



# Evolution Of Hardware Debug at FPGA Speed

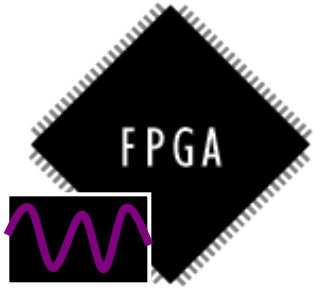
Logic Analyzer





ChipScope / SignalTap
*(Logic Analyzer-Like)*

**Embedded Logic Analyzer**

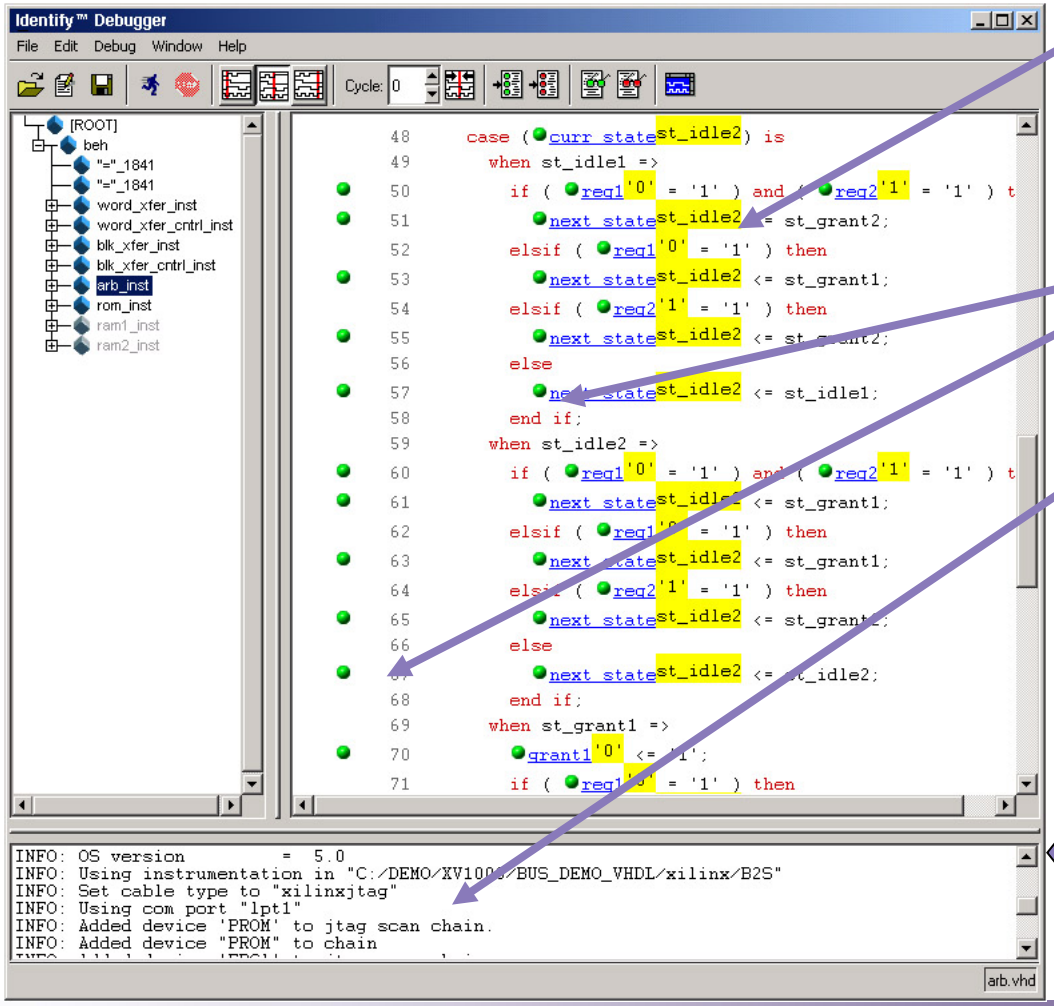



Identify Solution
*(Simulator-Like)*

**Embedded HDL Analyzer**

# Observe and Debug Data From the FPGA

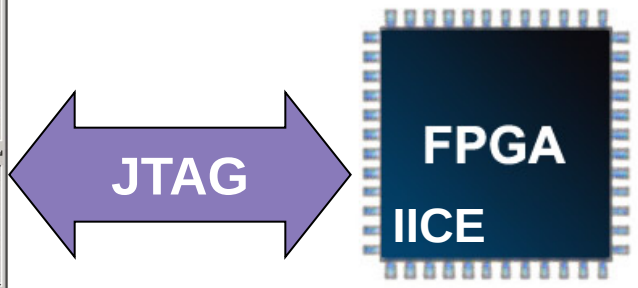
## Identify Debugger




Data values from FPGA tapped and annotated on top of your RTL to allow you to verify correspondence between RTL and the final implementation

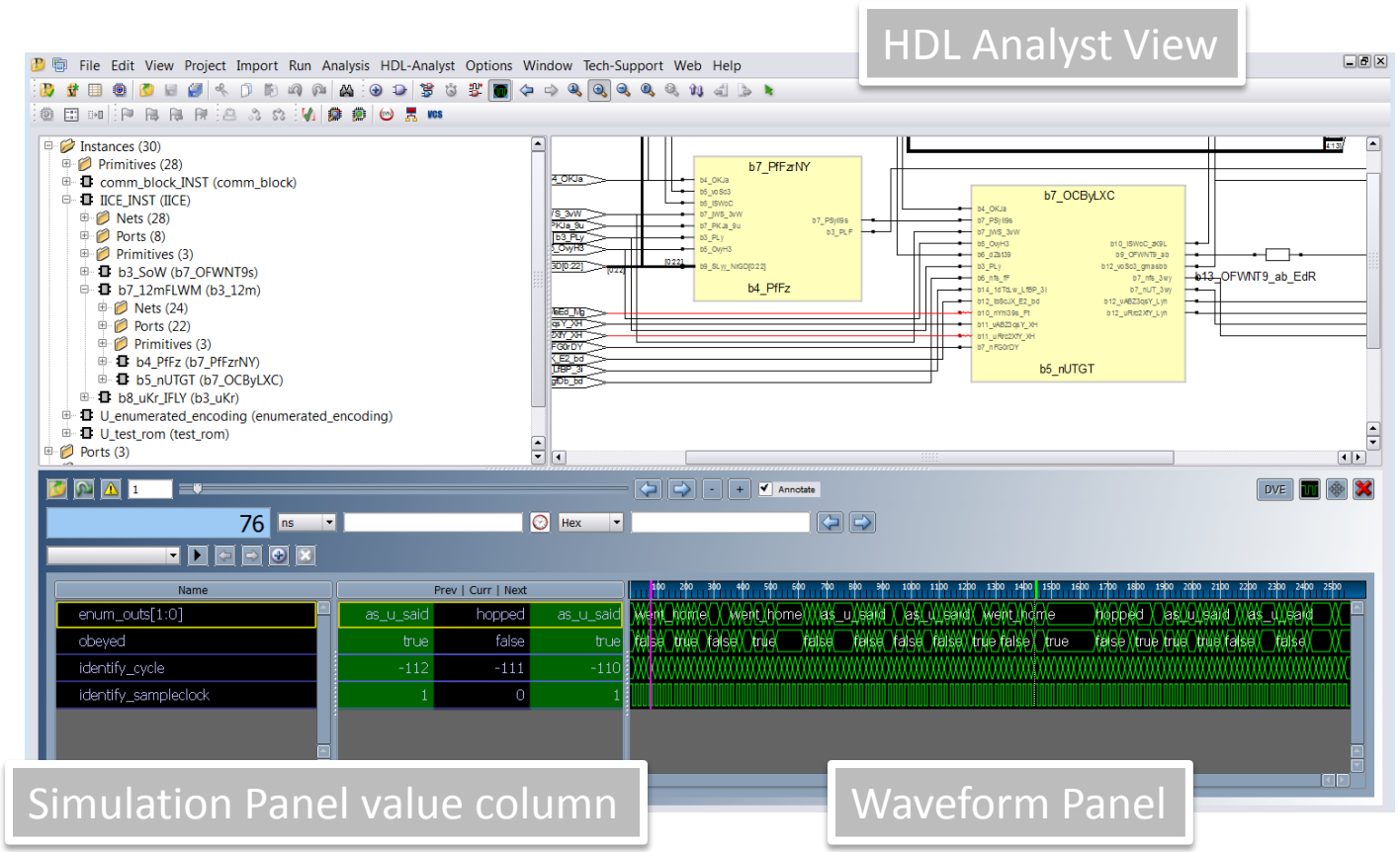
Click to enable triggers

Automate debugger with scripts



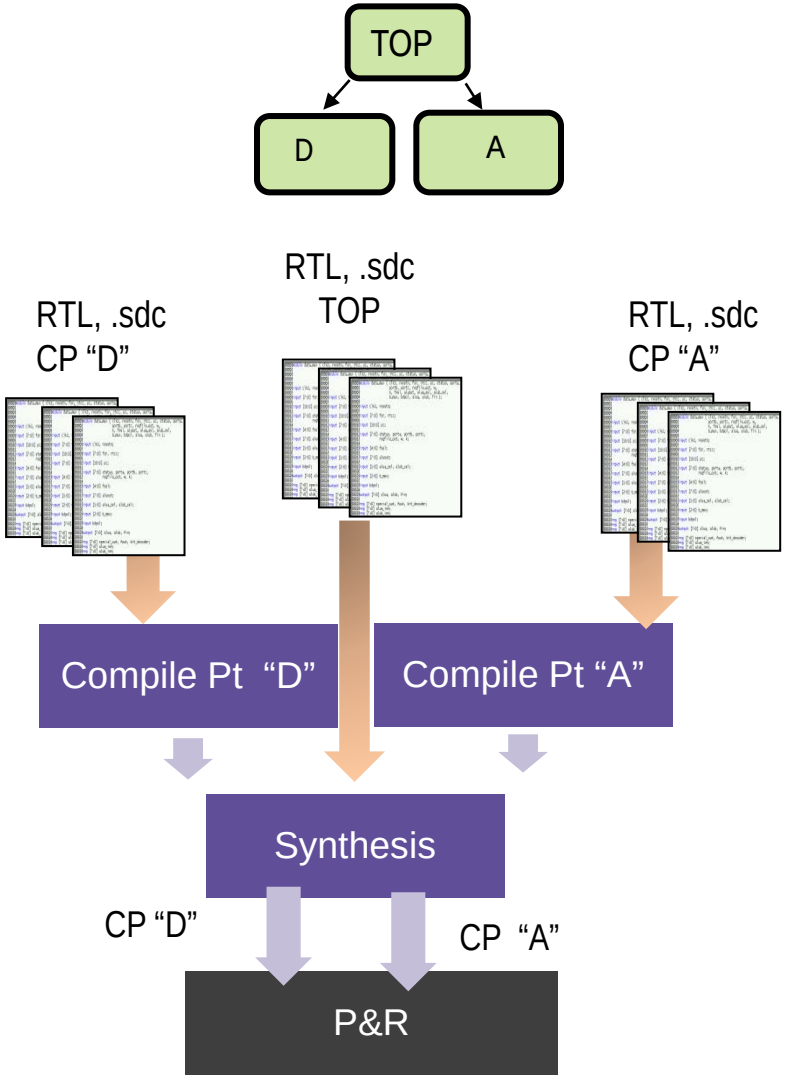
# Simulation and and On-Chip Debug Integration

## Visualization & Selection of VCD Data in HDL Analyst

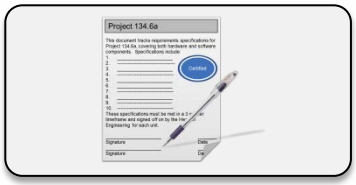


# Compile Point/Partition Block Based Flows

- "Divide and conquer" approach that saves time and ensures design repeatability.
- Isolate parts of the design
  - That already work
  - That comprise IP for which you wish to maintain port names for constraints application
- Partitions can be maintained throughout Synthesis and Place and Route



# 5 Challenges



## Complete and Accurate Design Specification

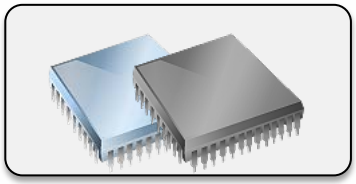
Early synthesis reports.
Constraint checking,
Clock domain cross checks

## Built in Design Reliability

Preserving critical design logic.
Safe Finite State machines
Triple Module Redundancy

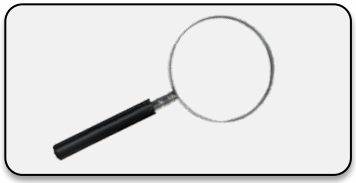
## Verifying the Design Meets Specification.

Design visualization.
Debug with RTL/Gate level Simulation.
Debug on-chip implementation.



## Reproducible, Documented Design Process

Documentation
Revision Control

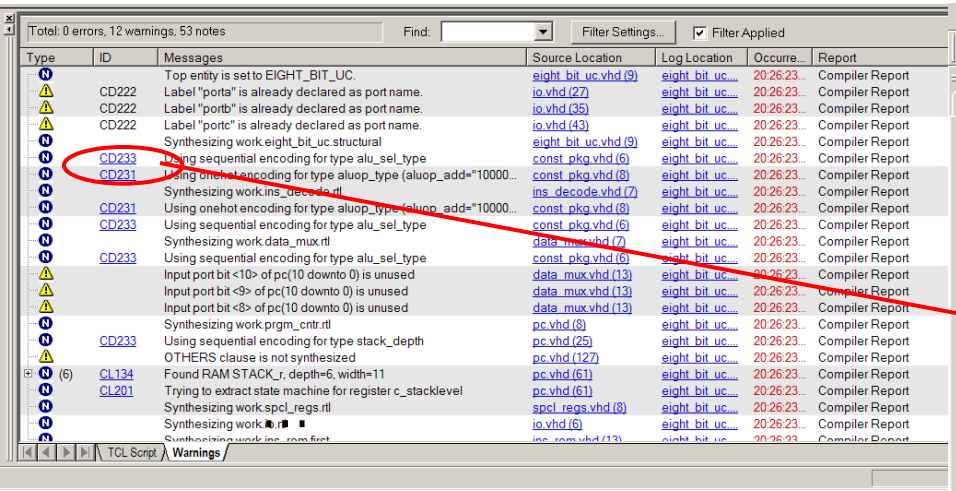


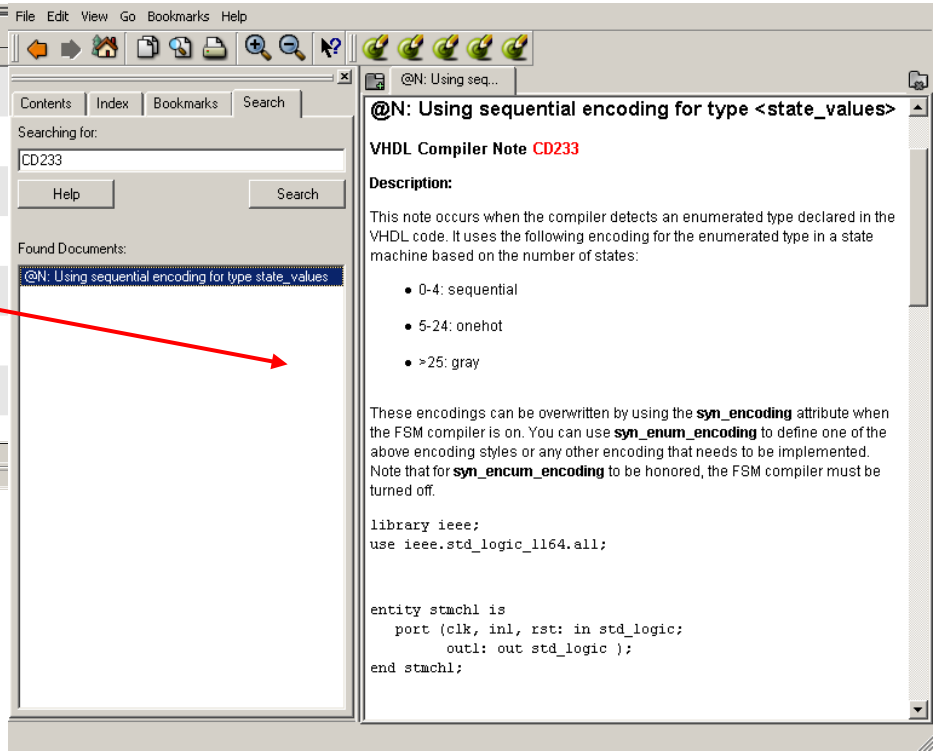
## Proof of Concept and Hardware-Based Validation.

Prototyping boards.
High Speed Interfaces.

# Reporting and Messaging
## Analyzing errors and warnings

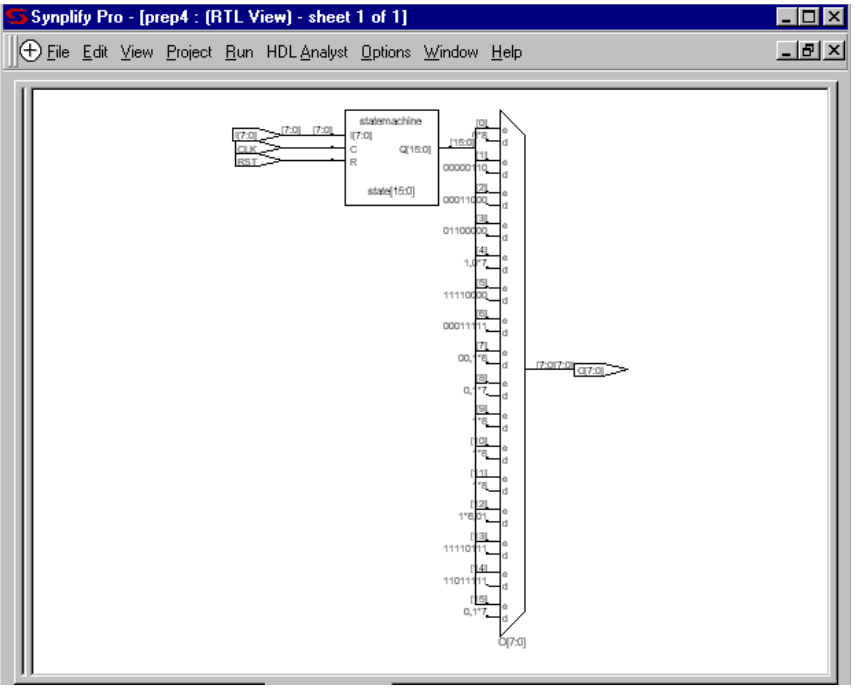
- Click on an errors or warning message.
- Generate text reports for sign-off.

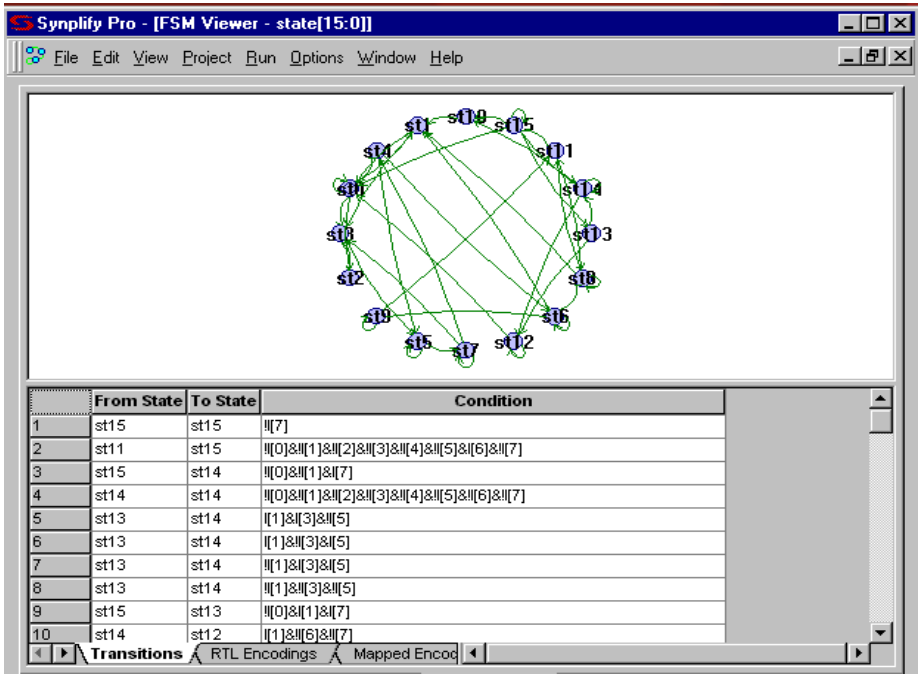

Total: 0 errors, 12 warnings, 53 notes
Find:
Filter Settings...
Filter Applied
Type
ID
Messages
Source Location
Log Location
Occurre...
Report
Top entity is set to EIGHT_BIT_UC.
Label "porta" is already declared as port name.
Label "portb" is already declared as port name.
Label "portc" is already declared as port name.
Synthesizing work.eight_bit_uc.structural
Using sequential encoding for type alu_sel_type
Using onehot encoding for type aluop_type (aluop_add="10000...
Synthesizing work.ins_decode.rtl
Using onehot encoding for type aluop_type (aluop_add="10000...
Using sequential encoding for type alu_sel_type
Synthesizing work.data_mux.rtl
Using sequential encoding for type alu_sel_type
Input port bit <10> of pc(10 downto 0) is unused
Input port bit <9> of pc(10 downto 0) is unused
Input port bit <8> of pc(10 downto 0) is unused
Synthesizing work.prgm_cntr.rtl
Using sequential encoding for type stack_depth
OTHERS clause is not synthesized
Found RAM STACK_r, depth=6, width=11
Trying to extract state machine for register c_stacklevel
Synthesizing work.spcl_regs.rtl
Synthesizing work.io.rtl
TCL Script
Warnings


File Edit View Go Bookmarks Help
Contents
Index
Bookmarks
Search
Searching for:
CD233
Help
Search
Found Documents:
@N: Using sequential encoding for type state_values
@N: Using seq...
@N: Using sequential encoding for type <state_values>
VHDL Compiler Note CD233
Description:
This note occurs when the compiler detects an enumerated type declared in the VHDL code. It uses the following encoding for the enumerated type in a state machine based on the number of states:
0-4: sequential
5-24: onehot
>25: gray
These encodings can be overwritten by using the syn_encoding attribute when the FSM compiler is on. You can use syn_enum_encoding to define one of the above encoding styles or any other encoding that needs to be implemented. Note that for syn_encum_encoding to be honored, the FSM compiler must be turned off.
library ieee;
use ieee.std_logic_1164.all;
entity stmchl is
port (clk, inl, rst: in std_logic;
outl: out std_logic );
end stmchl;

# HDLAnalyst Schematic View.

Copy and Paste graphics into documentation and reports




RTL View with state machine primitive [one hot representation]

FSM Viewer – State Transition Bubble Diagram and Transition Table

# Tcl Scripting To Generate Custom Reports

- In the command window
  - % source C:/report_dsp.tcl
  - % report_dsp
  - % report_rams
  - Generates custom reports for DSPs and RAMs.

DSP48 instances DSP48 instances

Technology View Instance Name: i:mult_1.un2_product_int[1:32]
Technology View Primitive Name: DSP48E1_14
RTL View Instance Name: mult_1.g1\.0\.product_int[31:0]

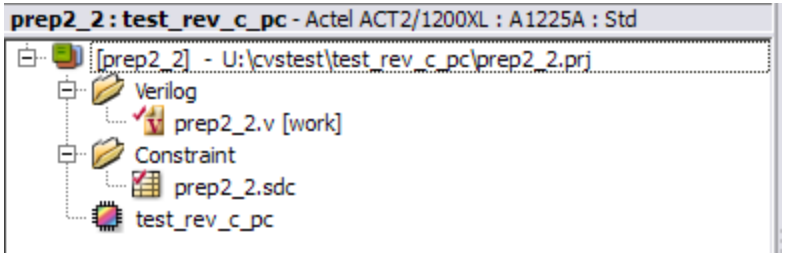
Distrbuted Ram instances

Technology View Instance Name: i:ram1_inst.ram_inst.mem_mem_0_0
Technology View Primitive Name: RAM256X1S
RTL View Instance Name: ram1_inst.ram_inst.mem[7:0]

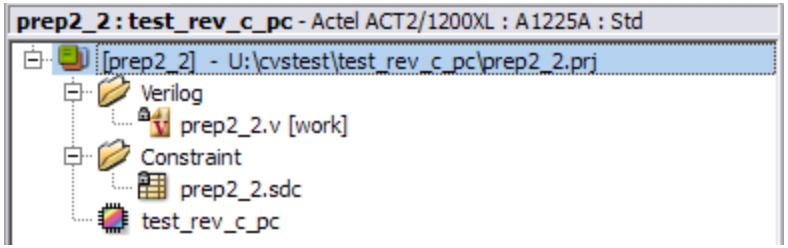
# Revision Control Systems

- Many different vendors, CVS, Subversion, ClearCase,….
- Required to prevent incorrect design files from becoming production!
- GUI based or command line.

- Red check mark = file checked out

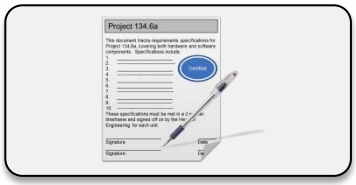



- Grey lock = file checked in




Note:
Use *Update Status* to refresh the icons!

# 5 Challenges



## Complete and Accurate Design Specification

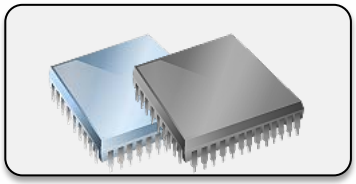
Early synthesis reports.
Constraint checking,
Clock domain cross checks

## Built in Design Reliability

Preserving critical design logic.
Safe Finite State machines
Triple Module Redundancy

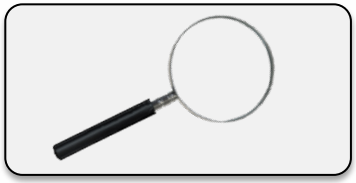
## Verifying the Design Meets Specification.

Design visualization.
Debug with RTL/Gate level Simulation.
Debug on-chip implementation.



## Reproducible, Documented Design Process

Documentation
Revision Control

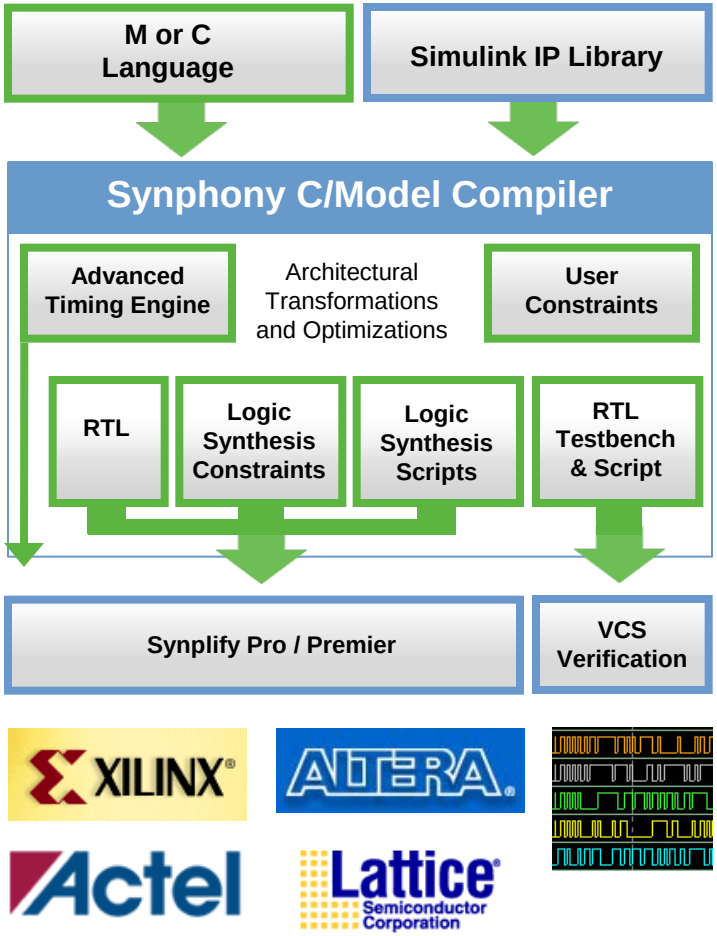


## Proof of Concept and Hardware-Based Validation.

Prototyping boards.
High Speed Interfaces.

# Early Algorithm Verification With Prototyping Boards.

- Verify Algorithm size and speed before production boards are available.
- Allows the capability to confidently move high level languages (C, Simulink)
- Drive the prototype with:
  - Testbenches from PC based tools (Simulation, Simulink)
  - Real world data from external sources.

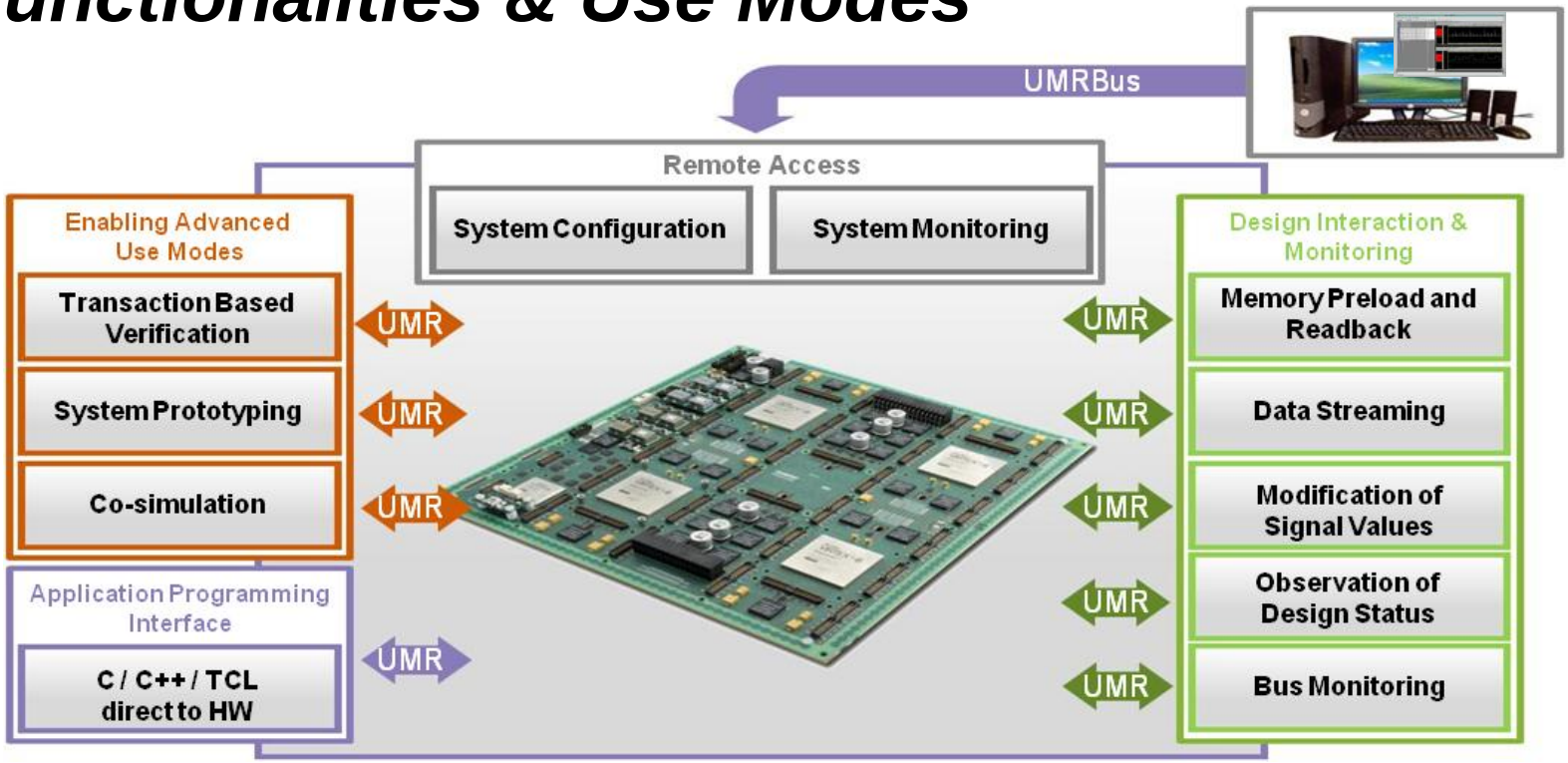


# Prototyping Boards.

- FPGA based prototyping boards available from many vendors
  - Smaller board: Xilinx, MicroSemi, Avnet, Altera, .
  - Largest boards: Synopsys HAPS
- Look for:
  - Board FPGA size and speed.
  - Board flexibility to interface with PC Software (Simulation, C, Matlab)
  - Board flexibility to interface with external  interfaces (PCIE, USB, Ethernet)
  - Board flexibility with clocks, voltages.

# Universal Multi-Resource Bus (UMRbus)

## *Functionalities & Use Modes*




### What It Is

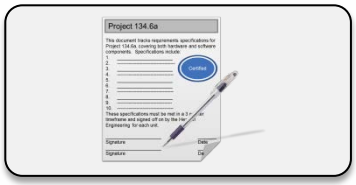
- High-performance, low-latency communication bus
- Connections to every FPGA, memories, registers, etc.

### Customer Benefits

- Remote prototype management
- Application-level programming
- Co-simulation
- Transaction-based verification

FAST UMRBus

# 5 Challenges



## Complete and Accurate Design Specification

Constraint checking,

Design Specification checking

## Built in Design Reliability

Triple Module Redundancy

Safe Finite State machines

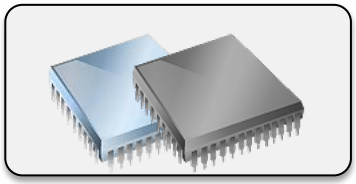
Maintaining  Debug and Test Logic.

## Verifying the Design Meets Specification.

Debug with RTL/Gate level Simulation.

Debug on-chip implementation at the gate level.

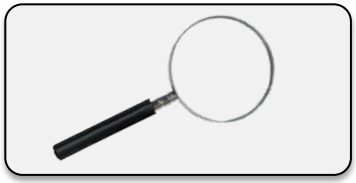
Debug on-chip implementation at the RTL Level



## Reproducible, Documented Design Process

Documentation

Revision Control



## Proof of Concept and Hardware-Based Validation.

Prototyping boards.

High Speed Interfaces.

# Questions?