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%)

<table>
<thead>
<tr>
<th>Starting Clock</th>
<th>Frequency</th>
<th>Frequency</th>
<th>Period</th>
<th>Period</th>
<th>Slack</th>
</tr>
</thead>
<tbody>
<tr>
<td>dll_clk_1x_derived_clock</td>
<td>100.0 MHz</td>
<td>243.7 MHz</td>
<td>10.000</td>
<td>4.104</td>
<td>5.896</td>
</tr>
<tr>
<td>pin_clk</td>
<td>200.0 MHz</td>
<td>200.0 MHz</td>
<td>5.000</td>
<td>5.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>
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 different 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

<table>
<thead>
<tr>
<th><strong>Complete and Accurate Design Specification</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Early synthesis reports.</td>
<td></td>
</tr>
<tr>
<td>Constraint checking.</td>
<td></td>
</tr>
<tr>
<td>Clock domain cross checks</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Built in Design Reliability</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Preserving critical design logic.</td>
<td></td>
</tr>
<tr>
<td>Safe Finite State machines</td>
<td></td>
</tr>
<tr>
<td>Triple Module Redundancy</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Verifying the Design Meets Specification.</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Design visualization.</td>
<td></td>
</tr>
<tr>
<td>Debug with RTL/Gate level Simulation.</td>
<td></td>
</tr>
<tr>
<td>Debug on-chip implementation.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Reproducible, Documented Design Process</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation</td>
<td></td>
</tr>
<tr>
<td>Revision Control</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Proof of Concept and Hardware-Based Validation.</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Prototyping boards.</td>
<td></td>
</tr>
<tr>
<td>High Speed Interfaces.</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>syn_keep</td>
<td>1/0</td>
<td>Preserve a net</td>
</tr>
<tr>
<td>syn_preserve</td>
<td>1/0</td>
<td>Preserve a cell / sequential component</td>
</tr>
<tr>
<td>syn_hier</td>
<td>firm, hard, macro, flatten</td>
<td>Preserve a block</td>
</tr>
<tr>
<td>syn_noprune</td>
<td>1/0</td>
<td>Preserve an instantiated component (Instance)</td>
</tr>
</tbody>
</table>
Safe State Machines.

- Synthesis tools are very good at optimizing FSMs for performance (FSM Compiler).
  - Re-encoding state-bits (i.e., one-hot)
  - Removing unreachable states. (i.e., 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

Starting Points with Max Worst Slack
EMPLATE with Max Worst Slack

<table>
<thead>
<tr>
<th>Starting</th>
<th>Arrival</th>
<th>Instance</th>
<th>Type</th>
<th>Pin</th>
<th>Time</th>
<th>Slack</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_clk</td>
<td>0.589</td>
<td>crossbar.wb_conmax_top.s2.msel.arb0.state[1]</td>
<td>Q</td>
<td>0.374</td>
<td></td>
<td></td>
</tr>
<tr>
<td>phy_clk_2</td>
<td>0.678</td>
<td>usb2.u4.csr[27]</td>
<td>Q</td>
<td>0.375</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sys_clk</td>
<td>0.904</td>
<td>crossbar.wb_conmax_top.s3.msel.pri_out[0]</td>
<td>Q</td>
<td>0.376</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Integrated Crossprobing Between HDLAnalyst Views and Source**
- **Starting Points with Max Worst Slack**
- **Arrival**
- **Instance**
- **Type**
- **Pin**
- **Time**
- **Slack**

```
6.0031 // declarations. I declared temp.
6.0032 reg [7:0] temp;
6.0033 if (add_sub)
6.0034 temp = b;
6.0035 else
temp = -b;
6.0037 result = a + temp + !add_sub;
6.0038 end
6.0039
6.0040 endmodule
```

- **Technology View**
- **View post mapped schematic with annotated timing**
- **Timing Report**
- **Analyze critical path**

---

© Synopsys 2010
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)

Identify Solution (Simulator-Like)

Embedded Logic Analyzer

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

JTAG

FPGA

IICE

© Synopsys 2010
Simulation and On-Chip Debug Integration
Visualization & Selection of VCD Data in HDL Analyst

HDL Analyst View
Simulation Panel value column
Waveform Panel
“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 error or warning message.
- Generate text reports for sign-off.
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

- 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]

Distributed 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

© Synopsys 2010
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
  - 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
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?