A Paradigm Shift in Verification Methodology

Pranav Ashar
Real Intent, Inc.
FMCAD, October 2016
The New Paradigm

Generic Tools

- RTL & Netlist Simulators
- Formal Equivalence Checker
- Assertion-based Formal Tool
- Static Timing Analyzer

Targeted Solutions

- **Untimed Paths**
  - Async Signals & Xings
  - Timing Exceptions
  - GALS

- **Clock-Domain Crossing Checker**
  - Timing Constraint and Exception Manager and Checker
  - Reset-Failure Tool

- **Unknown Values**
  - Lazy Initialization
  - Sync Reset

- **Initialization Checker**
  - X-Safety Tool
  - Power-Ctrl Manager and Checker

- **SOC Integration**
  - HW-SW Interface
  - Realizability

- **DFT Verifier**
  - Connectivity Checker
  - Register Verification Tool

- **Functional Fails**
  - Language Error
  - RTL Error
  - Interconnect Fabric Issue

- **RTL Code Linter**
- **RTL Auto-Formal Bug Hunter**
- **Security Verification Tool**
- **Protocol Verifier**
- **Deadlock Checker**
- **FIFO Checker**...
New Failure Modes are Very Real

• High performance
  • Includes high speed ARM11 cores & caches and over 10 application acceleration engines
  • Needs thorough analysis to ensure correct functionality

• Complex
  • Complex clock domain interactions
  • Many reset domains
  • Several asynchronous interfaces: Processor, caches, application engines, low-latency integrated memory, system and networking interfaces

• Large: >250M gates
  • Needs massive capacity for the design analysis

• High risk for silicon failure
  • Insidious bugs found late in the process

Cavium CNS3420 SoC Processor, designed specifically for Networking Devices with full offloading and hardware support for network stack, IP/SSL Security, RAID and NAT.
The Left Shift

- Start verification earlier
- Compress the development cycle
- Sign-off level confidence
- Lower Cost

Cost of a bug increases exponentially with each stage of the design process

Numerous iterations

Old-Style Verification

RTL Coding

Implementation

Old Paradigm

Fewer iterations

Targeted Verif

RTL Coding

Implementation

RTL Sign-off

New Paradigm

Shorter Development
A Manifestation of the New Paradigm

- **High-Value** verification targets
  - CDC, Reset, Constraints, Exceptions, X-safety etc
  - Beyond & complement existing flows (Simulation + STA)
- **Systematic** convergence
  - Setup + Semantic Analysis + Formal Analysis
  - Execute -> Review -> Iterate
- **Use Model**
  - Accuracy, Capacity, Debug, Data Mgmt

Raw RTL

Pre-signoff Verification

High integrity RTL

RTL Signoff Verification

Signed-off RTL

Netlist Signoff Verification

Signed-off Netlist

©Copyright 2016 Real Intent Inc., Proprietary and Confidential
A Manifestation of the New Paradigm

- **High-Value** verification targets
  - CDC, Reset, Constraints, Exceptions, X-safety etc
  - Beyond & complement existing flows (Simulation + STA)

- **Systematic** convergence
  - Framing + Scoping + Sign-off
  - Execute -> Review -> Iterate

- **Use Model**
  - Accuracy, Capacity, Debug, Data Mgmt

**Diagram**

- Raw RTL
  - Pre-signoff Verification
    - High integrity RTL
      - RTL Signoff Verification
        - Signed-off RTL
          - Netlist Signoff Verification
            - Signed-off Netlist

**Circular Diagram**

- Lint
  - AutoFormal
    - RTL Compliance
      - Functional Integrity
        - High Quality RTL
          - XV
            - X Safety
              - Clock & Reset Verification
                - RDC
Hidden Cost Without the New Tools: Over-design

• Many examples:
  • Extra latency on async crossings
  • Paths that could be exceptions are timed in STA
  • Explicitly reset every FF
  • Synchronous reset where Async reset could’ve worked
Hidden Cost Without the New Tools: Over-design

• Many examples:
  • Extra latency on async crossings
  • Paths that could be exceptions are timed in STA
  • Explicitly reset every FF
  • Synchronous reset where Async reset could’ve worked

STA constrains the normal timing paths

**Problem:** Assertion of SOFT_RST_A creates an untimed path
Hidden Cost Without the New Tools: Over-design

• Many examples:
  • Extra latency on async crossings
  • Paths that could be exceptions are unnecessarily timed in STA
  • Explicitly reset every FF
  • **Synchronous reset where Async reset could’ve worked**
Debug in the New Paradigm

Debug is a Major Bottleneck in Verification

- Customers spending >50% verification effort in Debug

![Pie chart showing verification efforts]

- Verification complexity requires advanced class and macro debug
- Debug methodology shift from signals → Class / Transactions

©Copyright 2016 Real Intent Inc., Proprietary and Confidential
• Better tools => Designers take more risks

Multi-mode FIFO

Mode 1: Clk1 and Clk2 are synchronous
Mode 2: Clk1 and Clk2 are asynchronous

Sneaky path causes a glitch
Moral Hazard (2)

• Better tools => Methodology is irrelevant

Untimed path was correct in RTL but mangled during synthesis.

Need:
1. Tighter control of scripts
2. Verification at every level
Moral Hazard (2)

RTL ✓

NETLIST X

Transmit (Tx) Domain
Receive (Rx) Domain

Glitch Free Mux

Synthesis / Optimization

Untimed path was correct in RTL but mangled during synthesis. Need: 1. Tighter control of scripts 2. Verification at every level.
Overall Impact of the New Paradigm is Salutary

- Exhaustive - No test benches
- Quick start and minimal setup
- Early detection - Helps prepare the design for simulation
- Sign-off on failure modes that are hard for simulation
- Address simulation’s limited semantics e.g. x-prop
- Parallelizes verification: Reduced simulation
- Shorter debug cycle time

Narrows the Verification Gap
**Problem:** Synthesis optimizes logic without knowing that X-pessimism is introduced

*Observed in actual netlists*

**Intended Functionality**

- Initializes to 0
- (Sel == 1 and Din == 0)

**Synthesis-Optimized Version**

- Will not be able to initialize to 0
- (Sel == 1 and Din == 0)
**Problem:** Synthesis optimizes logic without knowing that X-pessimism is introduced

*Observed in actual netlists*

### Intended Functionality

- **Input:** Sel, D0, D1
- **Output:** Out

- Initializes to 0
  - \( Sel = 1 \) and \( Din = 0 \)

### Synthesis-Optimized Version

- **Input:** Sel, D0, D1
- **Output:** Out

- Will not be able to initialize to 0
  - \( Sel = 1 \) and \( Din = 0 \)
X-pessimism analysis is conceptually a QBF problem

Is there a combination of $V_{\neg X}$ such that the value of $V_{Out}$ is the same for all projections of $V_X$?

$V_{\neg X}$ and $V_X$ are dynamic subsets of $V_{In}$
RTL and Netlist Simulations are Inaccurate in the Presence of X

If (sel)
D=1;
else
D=0;

Dsel=1'bx
CLK
1'b0

Optimism

D=sel*1+ ~sel*1
X
1
1
Sel=x
CLK
1'bx

Pessimism
Simulation behavior inaccurate
  • X’s cause bugs to be missed at RTL
  • X’s cause unnecessary additional X’s at netlist
Difficult to verify initialization in the presence of X’s
Gate level simulation bring up times are impacted by X’s
  • Massive productivity loss

RTL Design & Verification
RTL Handoff
Netlist Verification

So much for the #!?@! schedule

#!?@! My RTL works but my netlist fails. What is going on? Where to start?? I’m behind schedule! This signal isn’t even in my RTL – what is it?
Focus in on the Problem and Develop a Complete and Systematic Solution

- X’s appear in netlist simulations that were not in RTL simulations due to pessimism and due to real X’s that were masked by optimism in RTL
- Must resolve the optimism at RTL and then correct the pessimism in netlist simulations to avoid simulation differences at netlist.
Context-Smart Reporting and Debug

<table>
<thead>
<tr>
<th>Rows Per Page</th>
<th>Show/Hide Columns</th>
<th>Commit</th>
<th>Reset</th>
<th>Multi-Column Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>X SOURCE - List of X-sources in the design</td>
<td>XsrcType</td>
<td>Xin</td>
<td>Xout</td>
<td>EngineComments</td>
</tr>
<tr>
<td>Xsctype takes the values, listed in priority order:</td>
<td>User: Signal is treated as an X-source as directed by the signal BIT</td>
<td>Xin</td>
<td>Xout</td>
<td>EngineComments</td>
</tr>
<tr>
<td>Explicit: An explicit or unDriven: A signal that is used</td>
<td>Xin</td>
<td>Xout</td>
<td>EngineComments</td>
<td></td>
</tr>
<tr>
<td>BlackBox: Assignment</td>
<td>Xin</td>
<td>Xout</td>
<td>EngineComments</td>
<td></td>
</tr>
<tr>
<td>OutOfRange: Signal value is out of range</td>
<td>Xin</td>
<td>Xout</td>
<td>EngineComments</td>
<td></td>
</tr>
<tr>
<td>BusContent: A signal with potentially constrained or multiple drivers</td>
<td>Xin</td>
<td>Xout</td>
<td>EngineComments</td>
<td></td>
</tr>
<tr>
<td>UninitFlop: Flop that did not have a known Q1 value after reset</td>
<td>Xin</td>
<td>Xout</td>
<td>EngineComments</td>
<td></td>
</tr>
<tr>
<td>UninitLatch: Latch that did not have a known Q1 value after reset</td>
<td>Xin</td>
<td>Xout</td>
<td>EngineComments</td>
<td></td>
</tr>
<tr>
<td>NonRetained: Flop latch that was not set or a saturation flop by user and that was not an X-source</td>
<td>Xin</td>
<td>Xout</td>
<td>EngineComments</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RuleDataId</th>
<th>Command</th>
<th>Mode</th>
<th>Signal</th>
<th>XsrcType</th>
<th>Xin</th>
<th>Xout</th>
<th>EngineComments</th>
<th>RuleContentStatus</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>verify</td>
<td>Optimism</td>
<td>wb_adr_i[2:0]</td>
<td>User</td>
<td>Xin</td>
<td>48</td>
<td>16</td>
<td>Waived</td>
<td>waive, add monitor</td>
</tr>
<tr>
<td>2</td>
<td>verify</td>
<td>Optimism</td>
<td>wb_stb_i</td>
<td>User</td>
<td>Xin</td>
<td>40</td>
<td>40</td>
<td>New</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>verify</td>
<td>Optimism</td>
<td>wb_cyc_i</td>
<td>User</td>
<td>Xin</td>
<td>40</td>
<td>40</td>
<td>ToolWaived</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>verify</td>
<td>Optimism</td>
<td>wb_ack_o</td>
<td>User</td>
<td>Xin</td>
<td>40</td>
<td>40</td>
<td>New</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>verify</td>
<td>Optimism</td>
<td>wb_dat_i[7:0]</td>
<td>User</td>
<td>Xin</td>
<td>40</td>
<td>40</td>
<td>New</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>verify</td>
<td>Optimism</td>
<td>byte_controller</td>
<td>User</td>
<td>Xin</td>
<td>40</td>
<td>40</td>
<td>New</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>verify</td>
<td>Optimism</td>
<td>scl_pad_i</td>
<td>User</td>
<td>Xin</td>
<td>40</td>
<td>40</td>
<td>New</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>verify</td>
<td>Optimism</td>
<td>sda_pad_i</td>
<td>User</td>
<td>Xin</td>
<td>40</td>
<td>40</td>
<td>New</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>verify</td>
<td>Optimism</td>
<td>wb_dat_of_i[7:0]</td>
<td>UninitFlop</td>
<td>Xin</td>
<td>40</td>
<td>0</td>
<td>New</td>
<td></td>
</tr>
</tbody>
</table>

Sorting is by XIN risk factor

- **Type of X-source**
  - Number of data signals that it can propagate to
  - Number of control signals that it can propagate to

If (Xin) Xout <= X;

Status tracking and User Comments
Another New-Paradigm Example: CDC

- **The Metastability Problem**
  - When input changes within setup/hold window, the output of the flop becomes metastable, could settle into either 0 or 1

- **The Challenges**
  - Hard to detect and diagnose (with simulation or in the lab)
  - Very high number of CDC crossings
  - Variety of ways of implementing the crossings

- **Impact**
  - Chip failure in the field
  - Expensive to fix
The Metastability Problem
- When input changes within setup/hold window, the flop becomes metastable, settling into either 0 or 1.

The Challenges
- Hard to detect and diagnose (with simulation or in the lab)
- Very high number of CDC crossings
- Variety of ways of implementing the crossings

Impact
- Chip failure in the field
- Expensive to fix

Another New-Paradigm Example: CDC

Understand the problem at fundamental level
Establish that it is a high-value problem
Typical CDC Issues

Signal A must be held long enough to be captured by slow clock clk_B

Data loss in fast to slow transfer

Improper data enable sequence

Re-convergence of synced signals
Typical CDC Issues

- **Data loss in fast to slow transfer**
- **Data enable sequence**
- **Re-convergence of synced signals**

**Understanding the failure modes**

- Signal A must be held long enough to be captured by slow clock clk_B.
Important checks Setup stage
- Missing clocks and derived clocks
- Missing clock relationships
- Missing boundary conditions
- Missing resets
- Conflicts between env specs and/or design

Important Checks Structural analysis
- DATA and CNTL
- Glitch
- CNTL with multiple fanouts
- Reconvergence
- Resets crossing domains

Important Checks Formal analysis
- Data Stability
- Pulse Width
- Glitch Analysis
- GRAY CODE Checks
Systematic CDC Methodology

Important checks Setup stage
- Missing clocks and derived clocks
- Missing clock relationships
- Missing boundary conditions
- Missing resets
- Conflicts between env specs and/or design

Important Checks Structural analysis
- DATA and CNTL
- Glitch
- CNTL with multiple fanouts
- Reconvergence
- Resets crossing domains

Important Checks Formal analysis
- Data Stability
- Pulse Width
- Glitch Analysis
- GRAY CODE Checks
## Formal CDC Verification

### Formal Analysis Description

<table>
<thead>
<tr>
<th>Formal Analysis</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data stability</td>
<td>Check for safe data crossings across asynchronous clock domains</td>
</tr>
<tr>
<td>Gray code</td>
<td>Check that FIFO-related reconvergent control signals are Gray coded</td>
</tr>
<tr>
<td>Glitch analysis</td>
<td>Check that there is no glitch in the combinational circuit that can cause an incorrect value to be captured</td>
</tr>
<tr>
<td>Pulse width</td>
<td>Check that control crossings are held long enough to be sampled at the receiving domain</td>
</tr>
</tbody>
</table>
## Formal CDC Verification

### Formal Analysis Description

<table>
<thead>
<tr>
<th>Formal Analysis</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data stability</td>
<td>Check for data crossings across clock domains</td>
</tr>
<tr>
<td>Gray code</td>
<td>Check that FIFO-related reconvergent control signals are Gray coded</td>
</tr>
<tr>
<td>Glitch analysis</td>
<td>Check that there is no glitch in the combinational circuit that can cause an incorrect value to be captured</td>
</tr>
<tr>
<td>Pulse width</td>
<td>Check that control crossings are held long enough to be sampled at the receiving domain</td>
</tr>
</tbody>
</table>
Basic Data Stability Check

- Rising/Falling transition on Tx Flop lead to Rising/Falling transition on Rx Flop at next edge or Rx Clock.
Opposing transitions on TxFlops lead to a glitch on Rx Flop
Formal CDC Verification

- Parallel Formal for high throughput
  - Almost 100% coverage of failure trace, pass or deep-bounded pass

- Constraints support
  - Enable SVA/PSL constraints on the fly
  - Extract constraint dependence
  - Show in the debug

- Flexible tool control
  - Fast (re)start of formal analysis iterations
  - Inform users on formal run progress and completion status
Formal CDC Verification

- Parallel Formal for high throughput
  - Almost 100% coverage of failure trace, pass or deep-bounded pass

- Constraints support
  - Enable SVA/PSL constraints
  - Extract constraints
  - Show in the debugger

- Flexible tool control
  - Start of formal analysis iterations
  - Inform users on formal run progress and completion status

Enable throughput and deep-checking in formal analysis
Context-Smart Debug

Enable Systematic Closure
Scope Based Reporting: Simultaneous Chip Level and Block level results

- ModuleScope - the scope of the design violation is well contained in
- Available for all the rules
- Accessible through GUI or CLI for quick debug/SignOff

Speeds up CDC signoff by 3X
Scope Based Reporting: Simultaneous Chip Level and Block level results

- ModuleScope - the scope of the design violation is well contained in
- Available for all the rules
- Accessible through GUI or CLI for quick debug/SignOff

Enable Systematic Closure

Speeds up CDC signoff by 3X
The Constraints Problem

- Creating synthesis constraints takes a lot of effort
- Post-budgeting, partitioned SDC misses critical constraints - requires synthesis and P&R iterations
- Exceptions must be correct to ensure accurate sign-off analysis
- Constraints need to be managed across design partitions

The result – iterations and timing closure delays
The Constraints Problem

- Creating synthesis constraints takes a lot of effort.
- Post-budgeting, partitioned SDC misses critical constraints - requires synthesis and P&R iterations.
- Exceptions must be correct to ensure accurate sign-off analysis.
- Constraints need to be consistent across design flow.
- Constraints need to be managed across design partitions.

Product definition and design

RTL design

Functional verification

Planning/partitioning

Synthesis

Floorplanning, P&R

Layout

Functional verification

Planning/partitioning

RTL design

RTL design

The result – iterations and timing closure delays.

Understand the problem at fundamental level.
Establish that it is a high-value problem.
Then, set up solution to Frame, Scope and Analyze.
Apply semantic analysis to reframe the problem.
• Reset-Safety
  • Metastability & Correlation-loss based failure modes

• Auto-Formal
  • RTL functional implementation bugs
  • Challenge: Identify actionable failures quickly
  • Very high volume of implicit checks: Throughput vs. Depth
  • Root-cause analysis is key

• Code Quality (Lint)

• A Few More ...
Frame, Scope and Analyze Other Problems!

For example, Deadlock Verification