How Formal Methodology Shrank the Verification Schedule of a Complex Statistics Block by 6x

Sandesh Borgaonkar, Oski Technology
Chirag Agarwal, Oski Technology
Rajesh Kothari, Cisco Systems
Darrow Chu, Cadence Design Systems
Introduction

- Design Port Statistics Module (PSM) needs to
  - react to hundreds of packets stat updates in real time
  - compute parity and perform ECC detection/correction
  - allow simultaneous, asynchronous CPU updates
- Simulation would have unacceptably increased quality and schedule risk
- End-to-end formal verification completed about 6x faster
  - And resulted in difficult corner-case bugs almost impossible to find with simulation
**Design overview**

**DUT interfaces**

- **ID**
- **CPUIn**
- **RxStats**
- **RxTxStats**
- **Clk**
- **Interrupt**
- **PSM**
- **TxStats**
- **PSM**
- **CPUOut**

**DUT hierarchy**

- **PSM**
- **Rx**
- **RMW**
- **PSM**
- **CPU**
- **Tx**
- **RMW**

**Design overview**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of PSM Inputs</td>
<td>353</td>
</tr>
<tr>
<td>No. of PSM outputs</td>
<td>103</td>
</tr>
<tr>
<td>No. of Rx RMWs</td>
<td>57</td>
</tr>
<tr>
<td>No. of Tx RMWs</td>
<td>25</td>
</tr>
<tr>
<td>No. of Lines in RTL</td>
<td>211,255</td>
</tr>
<tr>
<td>No. of Flops</td>
<td>82,602</td>
</tr>
</tbody>
</table>

**PSM top level block diagram**

- **57 Instances**
- **Even Bank**
- **Odd Bank**
- **25 Instances**
- **Even Bank**
- **Odd Bank**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Type1 RMW</td>
<td>37</td>
</tr>
<tr>
<td>No. of Type2 RMW</td>
<td>42</td>
</tr>
<tr>
<td>No. of Type3 RMW</td>
<td>2</td>
</tr>
<tr>
<td>No. of Type4 RMW</td>
<td>1</td>
</tr>
<tr>
<td>No. of Memories</td>
<td>164</td>
</tr>
</tbody>
</table>
Why Formal?

• About 250 different QOS, operational, diagnostic statistics for 28 channels, in over 150 memories
  • billions of unique, legal states

• Parallelized controllers for each statistic
  • Each can be interrupted by asynchronous CPU updates

• Plenty of corner cases
  • cross-combinations of counter saturation with simultaneous updates from CPU Reads and Writes

• Getting full coverage with simulation is challenging
  • Hard-to-hit corner cases
  • Requires thousands of cycles to reach interesting counter saturation states
How in Formal?

**Phased Process**: applied Formal in several discrete, successively more complex phases eventually covering complete design

1. Initially, reduced design to have limited RMWs
2. Restricted to fewer channels, disallowing CPU access
3. Controlled statistics traffic to have pure CPU read/write
4. Eventually full traffic allowed to check complete functionality, on original design

**Formal Challenges**: Abstraction methods were used to surmount the challenges for formal verification methodology caused by:

- Numerous statistics spread over in multiple memories, for many channels
- Deep counters (28, 32 bits)
Memory abstraction

- Abstracted to single address memory model
- Tracked address is controlled by symbolic random variable
- Memory depth is reduced by $28^*n$ times
  - $n$ is number of statistics maintained in the corresponding memory
- Abstracted to have symbolic random reset value

Symbolic Random

##1 \text{select} == \$\text{past}(\text{select})$

Original Memory IP

Abstract Memory Model
Counter Abstraction (1/2)

- Tracks one counter at a time
- All counters are tracked one-by-one
- Single run is sufficient to check all stats in all memories

Multiple RMWs

82 RMW (250 Stats) X 2 Banks X 28 Ports

Selected RMW

1 to 7 Stats X 2 Banks X 28 Ports

Selected Bank

1 to 7 Stats X 28 Ports

Selected Stat

1 Stat X 28 Ports

Selected Port

1 Stat

Counter Selection Abstraction
Counter Abstraction (2/2)

- Reduced sequential depth from state space search

- Reset Value == 0

- Random Reset Value

- Takes N cycles

- Value == Max

- Takes 1 cycle

- Value == Max

Similar abstractions are used for PSM registers to verify individual bits
Formal setup

Constraints

- Checkers: 64
- Covers: 20

Project Duration

- Time to find First Bug: 1 Week
- Total # of Bugs: 26
- Total Run Time: 12 Hrs

Formal setup block diagram
Checkers

- RMW related checkers
  - 9 checkers for each type of RMW (9x4 = 36 Checkers)
  - No statistics loss
    - CPU is able to read latest and correct statistic value
  - Parity errors generate corresponding error flag
- Configuration/Status register related checkers
  - 20 checkers for configuration/status registers
  - Register read/write under various traffic scenarios
- Central CPU interface related checkers
  - 8 checkers for Central CPU interface
  - CPU interface generates grant for all valid CPU requests
  - CPU bypass
Bugs and covers

• Bugs
  • Unexpected data loss in memory
  • False/double generation of ack
  • Incorrect statistic reported on CPU read
  • Parity error flag/interrupt

• Covers
  • Worst case traffic scenarios
  • Concurrent CPU and statistics access
  • CPU timeout and interrupts
  • No over constraints
Bugs found

- Incorrect Memory Update

1\textsuperscript{st} StatUpd Req to 'h0F even bank

2\textsuperscript{nd} StatUpd Req to 'h0F even bank

CPU Read Req to 'h0F even bank

Stat updates to multiple addresses and bank of this RMW

Memory gets updated to 0x4003

Memory reset to 0x0000

Memory reset to 0x4001

CPU read returns value of 0x4001
## Comparison with simulation

<table>
<thead>
<tr>
<th>Metric</th>
<th>Formal</th>
<th>Simulation (estimated)</th>
</tr>
</thead>
<tbody>
<tr>
<td># of lines in testbench</td>
<td>9,814</td>
<td>~10,000</td>
</tr>
<tr>
<td># of checkers (tests)</td>
<td>64</td>
<td>159</td>
</tr>
<tr>
<td># of lines in tests</td>
<td>0</td>
<td>~5000</td>
</tr>
<tr>
<td>Trace Length</td>
<td>&lt; 27 cycles</td>
<td>1,000 – 100,000 cycles</td>
</tr>
<tr>
<td>Run Time</td>
<td>~ 12 hrs</td>
<td>&gt; 12 hrs</td>
</tr>
<tr>
<td># of Bugs Found</td>
<td>26</td>
<td>&lt; 26</td>
</tr>
<tr>
<td>Verification time</td>
<td>3 Months</td>
<td>18 Months</td>
</tr>
</tbody>
</table>