Overcoming AXI Asynchronous Bridge Verification Challenges with AXI Assertion-Based Verification IP (ABVIP) and Formal Datapath Scoreboards

Bochra El-Meray, ST-Ericsson

Jörg Müller, Cadence
About the Authors

- **Bochra Elmeray**
  - Verification Engineer at ST-Ericsson Rabat
  - 5 years experience in IP verification
  - Expert in Formal Verification

- **Jörg Müller**
  - Solutions Engineer at Cadence Design Systems Munich
  - 15 years experience in ASIC Design and Verification
  - Expert in Formal Verification
  - Supported ST-Ericsson for advanced verification
Agenda

- Overview GALS Design Verification
- Environments
- Formal Protocol Checking
- Formal Functional Checking
- Technology
- Results
- Conclusion
Overview GALS Design Verification

• Definition: GALS
  • Globally Asynchronous Locally Synchronous design techniques used for SoC
  • Solve physical implementation problems (power, timing, etc)
  • Requires synchronization between clock domains with different frequencies

• Synchronizer between domains
  • Example: AXI2AXI bridge
  • 2 clock domains
    • AXI Master
    • AXI Slave

• Verification Challenges
  • Protocol Compliance
  • Datapath Integrity
Verification Environments

- **Traditional**
  - Constrained Random Simulation (SpecMan)
  - Metric Driven Analysis (Coverage and Fault)
  - Applied on sub system level only, not on IP level
  - Focus on known application scenarios only, missed bugs
- **First Formal**
  - Many inconclusive results due to complexity of design
  - Debugging failures on signal level is difficult
  - No functional checking, only protocol compliance
  - No verification plan or progress metrics
- **New Formal**
  - Adding methodology and technology to fill holes
  - Replace simulation efforts for IP level features
New Formal Verification Strategy

- Add 2 new components in the environment
  - New AXI3 Assertion Based Verification IP optimized for protocol checking
  - New methodology for verifying asynchronous datapaths for functional checking
- Embed it in formal-aware metric driven verification and regression environment
  - Orchestrating and distributing formal environments on server farms
  - Collect results and provide global view of overall verification state
  - Allows tracking of progress and assessment of completeness
- Take advantage of new debugging capabilities
  - Transaction level representation of AXI protocol activity
- Leverage latest formal technology available
  - Incisive Enterprise Verifier XL
Protocol Verification

- Goal: Guarantee protocol compliance against AXI specification
- Technology used: Formal and Assertion Based VIP (ABVIP)
- Optimized properties for formal validation of interface protocol
  - Instantiate and connect to DUT interface
  - Provides checks and constraints for protocol compliance checking
  - Provides constraints for functional checking
Functional Verification

- Goal: Guarantee core functionality of the bridge – data transport
- Methodology introduced: Verifying asynchronous datapaths with formal scoreboarding
  - Utilizing symbolic sequences (refers to Wolper, Stangier, Mueller)
  - Formally verifies data integrity
  - Implemented as formal scoreboard (provided by Cadence)
- Fills hole of previous formal verification environment

![Diagram showing DUT with inputs and outputs](image)
Symbols used in Formal Scoreboard

- Using Symbol in Formal Verification
  - Declare one symbol that represents all possible values, in all possible locations, at all possible times, under any possible condition
  - Symbol implemented as non-deterministic constant
    \[
    \text{wire } [31:0] \text{ symbol; } // \text{ uninitialized} \\
    \text{assert property($\text{stable(symbol)}$),}
    \]
  - Example sequence „one symbol only“
    \[
    \text{00...00100..00}
    \]
  - Symbol used to constrain unique value in input domain sequence
    \[
    \text{assume property @(posedge in_clk)} \\
    \text{in\_symbol\_seen } \&\& \text{ in\_dvalid } |\rightarrow \text{ in\_data } != \text{ symbol);}
    \]
  - Symbol used to check for matching same value in the output domain sequence
    \[
    \text{assert property @(posedge out\_clk)} \\
    \text{out\_symbol\_seen } \&\& \text{ out\_dvalid } |\rightarrow \text{ out\_data } != \text{ symbol);}
    \]
Sequences of Symbols in Formal Scoreboard

1. always symbol
   
2. never symbol
   
3. first symbol
   
4. one symbol only
   
5. two consecutive symbol
   
6. two symbol only
   
7. two consecutive symbol only*

* Stangier‘s approach

Definition:

- `s`: Symbol
- `.`: Anything but symbol
- `?`: Anything
Error Types Detected by Formal Scoreboard

1. loss (arbitrary item)  \( \text{XXXX} \rightarrow \text{XXXC} \)
2. loss all (items of particular value C)  \( \text{XXXX} \rightarrow \text{XXX} \)
3. creation (arbitrary value)  \( \text{XXXXX} \rightarrow \text{XXXX} \)
4. creation (illegal value only)  \( \text{XXXXX} \rightarrow \text{XXXX} \)
5. duplication (same value only)  \( \text{XXXX} \rightarrow \text{XXXX} \)
6. manipulation (to arbitrary value)  \( \text{XXXX} \rightarrow \text{XXXX} \)
7. manipulation (to illegal value only)  \( \text{XXXX} \rightarrow \text{XXXX} \)
8. reordering (arbitrary items)  \( \text{XBCXX} \rightarrow \text{XCBXX} \)

- Different sequences can detect different type of errors
- All sequences overlay to full coverage of error types
Instantiating Formal Scoreboard

```vhdl
if_scoreboard #(
    // Parameters
    "DBUS_WIDTH" (ID_WIDTH), // Size of the external datapath
    "CHECK_WIDTH" (CHECK_WIDTH) // Size of the internal datapath
) sb_awid (
    // Ports
    .rst_n (rst), // active low reset
    .in_clk (aclks), // input clock
    .in_data (awids), // input data vector
    .in_dvalid (awvalids && awreadys), // input valid indicator
    .out_clk (aclkm), // output clock
    .out_data (awidm), // output data vector
    .out_dvalid (awvalidm && awreadym) // output valid indicator
);
```
Datapaths in the Async AXI Bridge

- For our AXI bridge we identified a total of 7 data transport paths
  1. Write Address ID
  2. Write Data ID
  3. Write Response ID
  4. Write Data
  5. Read Address ID
  6. Read Response ID
  7. Read Data
- Each receive a formal scoreboard instance
- Fully covering functional paths across the clock domain crossing
Technology in New Formal Environment

- Latest Engines in Incisive Enterprise Verifier
  - Addition and improvements of formal engines and running them all in parallel
  - Contributes to faster runtime and overall improved results

- Assertion Driven Simulation
  - ADS runs simulation using PSL/SVA constraints as testbench
  - Allows fast design exploration and provides instant feedback on constraints

- Replay
  - Using traces obtained by formal engine to guide ADS activating other properties
  - Contributed additional failures on previously explored properties

- Constraint Minimization
  - Patented algorithm to identify minimized set of constraint required for proof
  - Contributed additional Fail and Pass results on previously explored
Debugging

### AXI 3 TABLES - top.axi_monitor

#### WRITE TABLE

<table>
<thead>
<tr>
<th>ID</th>
<th>Count</th>
<th>Address</th>
<th>CtlDone</th>
<th>DatStart</th>
<th>DatDone</th>
<th>Valid</th>
<th>Len</th>
<th>Size</th>
<th>Burst</th>
<th>Lock</th>
</tr>
</thead>
<tbody>
<tr>
<td>'d 217</td>
<td>'d 1</td>
<td>'h 8FFC</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Len2</td>
<td>Word</td>
<td>Wrap</td>
<td>Normal</td>
</tr>
<tr>
<td>'d 54</td>
<td>'d 1</td>
<td>'h A04A</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Len1</td>
<td>TwoW</td>
<td>Incr</td>
<td>Normal</td>
</tr>
</tbody>
</table>

#### READ TABLE

<table>
<thead>
<tr>
<th>ID</th>
<th>Count</th>
<th>Valid</th>
<th>Lock</th>
</tr>
</thead>
<tbody>
<tr>
<td>'d 0</td>
<td>'d 0</td>
<td>0</td>
<td>'d 0</td>
</tr>
<tr>
<td>'d 0</td>
<td>'d 0</td>
<td>0</td>
<td>'d 0</td>
</tr>
</tbody>
</table>

#### EXCLUSIVE TABLE

<table>
<thead>
<tr>
<th>ID</th>
<th>Len</th>
<th>Address</th>
<th>Valid</th>
<th>ExError</th>
<th>ExRead</th>
<th>ExWrite</th>
<th>Size</th>
<th>Burst</th>
<th>Prot</th>
<th>Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>'d 0</td>
<td>Len1</td>
<td>'h 0000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Byte</td>
<td>Fixed</td>
<td>'b 000</td>
<td>'b 0000</td>
</tr>
<tr>
<td>'d 0</td>
<td>Len1</td>
<td>'h 0000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Byte</td>
<td>Fixed</td>
<td>'b 000</td>
<td>'b 0000</td>
</tr>
</tbody>
</table>
Regression Suite
Comparing Formal Environments

<table>
<thead>
<tr>
<th></th>
<th>Config 1</th>
<th></th>
<th>Config 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Old</td>
<td>New</td>
<td>Old</td>
<td>New</td>
</tr>
<tr>
<td>Total</td>
<td>115</td>
<td>144</td>
<td>108</td>
<td>141</td>
</tr>
<tr>
<td>Pass</td>
<td>75</td>
<td>108</td>
<td>74</td>
<td>109</td>
</tr>
<tr>
<td></td>
<td>(65%)</td>
<td>(75%)</td>
<td>(68%)</td>
<td>(77%)</td>
</tr>
<tr>
<td>Fail</td>
<td>8</td>
<td>9</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>(7%)</td>
<td>(6%)</td>
<td>(3%)</td>
<td>(6%)</td>
</tr>
<tr>
<td>Explored*</td>
<td>32</td>
<td>27</td>
<td>31</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>(28%)</td>
<td>(19%)</td>
<td>(29%)</td>
<td>(16%)</td>
</tr>
</tbody>
</table>

* The explored results were obtained with 1 hour tool effort per property
Finding Critical Bugs

- Failure Detected: ID values across locked access do not match!
- Scenario: Normal data without request enters bridge before lock
- Impact: Potentially blocking entire SoC
Summary

• Pro:
  • Positive experience with formal verification, scoreboard and ABVIP
  • Overall quality of results improved tremendously
  • Found corner case bug missed by simulation

• Con:
  • Some bounded proofs remained (although depth increased)
  • Not a push button flow (but that was not expected either)

• Conclusion
  • We count on mixed formal and simulation (ADS) in future projects of that type
  • Completeness of setup (protocol + functional) gives confidence to sign of IP without spending further resources on verification
THANK YOU