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

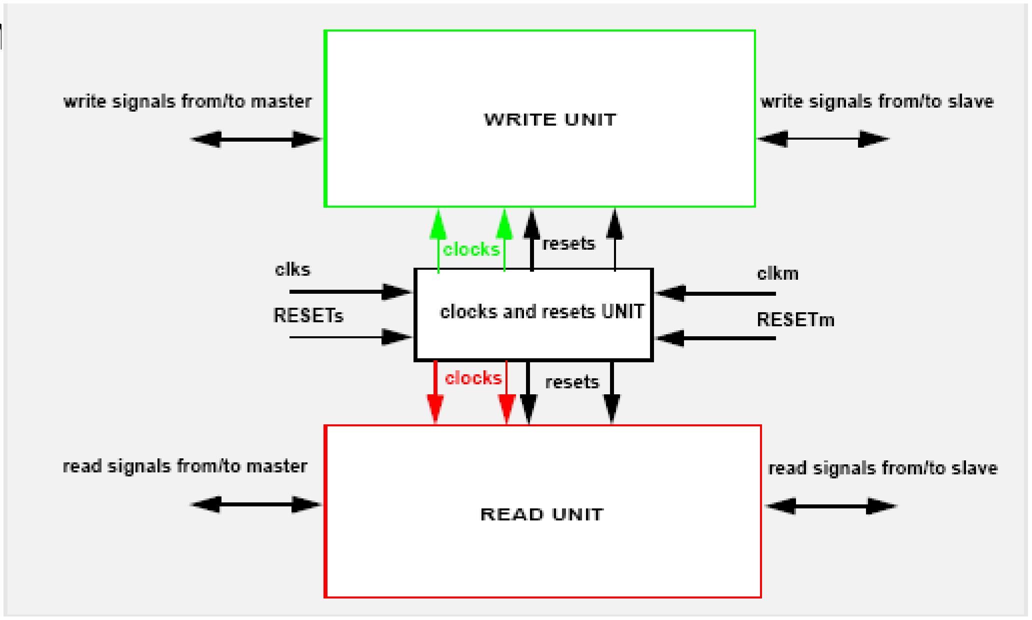
- Jörg Müller
  - Solutions Engineer at Cadence Design Systems Munich
  - 15 years experience in ASIC Design and Verification
  - Expert in Formal Verification
  - Supported ST-Ericsson for advanced verification

Agenda

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

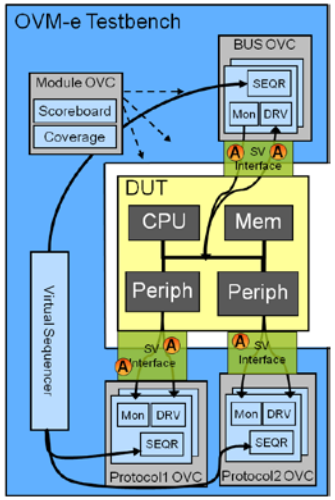
Overview GALS Design Verification

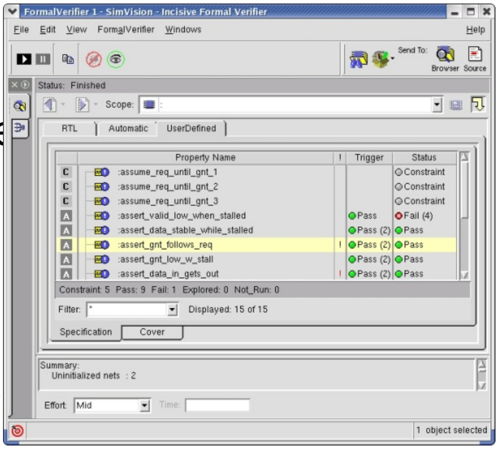
- Definition: GALS
  - Globally Asynchronous Locally Synchronous design techniques used for SoC
  - Solve physical implementation problems (power, timing, etc)
  - Requires synchronization between clock domains with different frequencies
- Synchronizer between dom
  - Example: AXI2AXI bridge
  - 2 clock domains
    - AXI Master
    - AXI Slave
- Verification Challenges
  - Protocol Compliance
  - Datapath Integrity


write signals from/to master
WRITE UNIT
write signals from/to slave
clocks
resets
clks
RESETs
clocks and resets UNIT
clkm
RESETm
clocks
resets
read signals from/to master
READ UNIT
read signals from/to slave

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 scenarioes only, missed bug
- First Formal
  - Many inconclusive results due to complexity of design
  - Debbugging 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




FormalVerifier 1 - SimVision - Incisive Formal Verifier
File Edit View FormalVerifier Windows Help
Send To: Browser Source
RTL
Automatic
UserDefined
Specification
Cover

Status: Finished

Scope: :

	Property Name	!	Trigger	Status
C	:assume_req_until_gnt_1			Constraint
C	:assume_req_until_gnt_2			Constraint
C	:assume_req_until_gnt_3			Constraint
A	:assert_valid_low_when_stalled		Pass	Fail (4)
A	:assert_data_stable_while_stalled		Pass (2)	Pass
A	:assert_gnt_follows_req	!	Pass (2)	Pass
A	:assert_gnt_low_w_stall		Pass (2)	Pass
A	:assert_data_in_gets_out	!	Pass (2)	Pass

Constraint: 5 Pass: 9 Fail: 1 Explored: 0 Not_Run: 0

Filter: * Displayed: 15 of 15

Summary:
Uninitialized nets : 2

Effort: Mid Time:

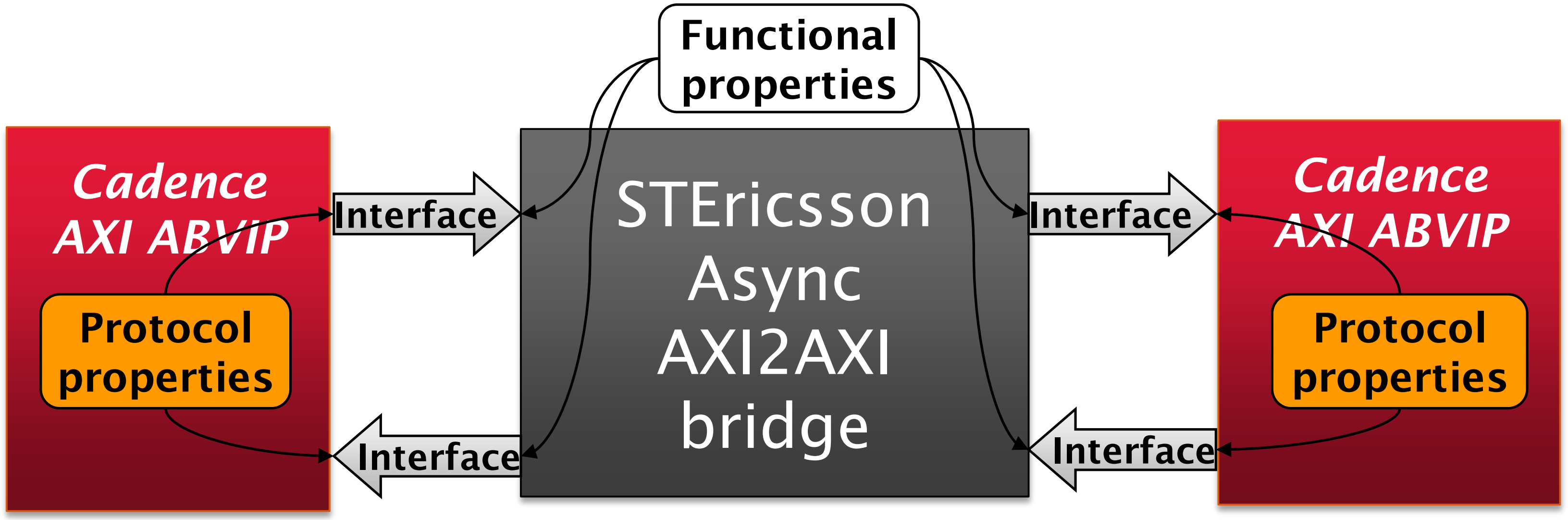
1 object selected

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
- Embedd 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 completenss
- Take advantage of new debugging capabilities
  - Transaction level representation of AXI protocol activity
- Leverage latest formal technology available
  - Incisive Enterprise Verifier XL

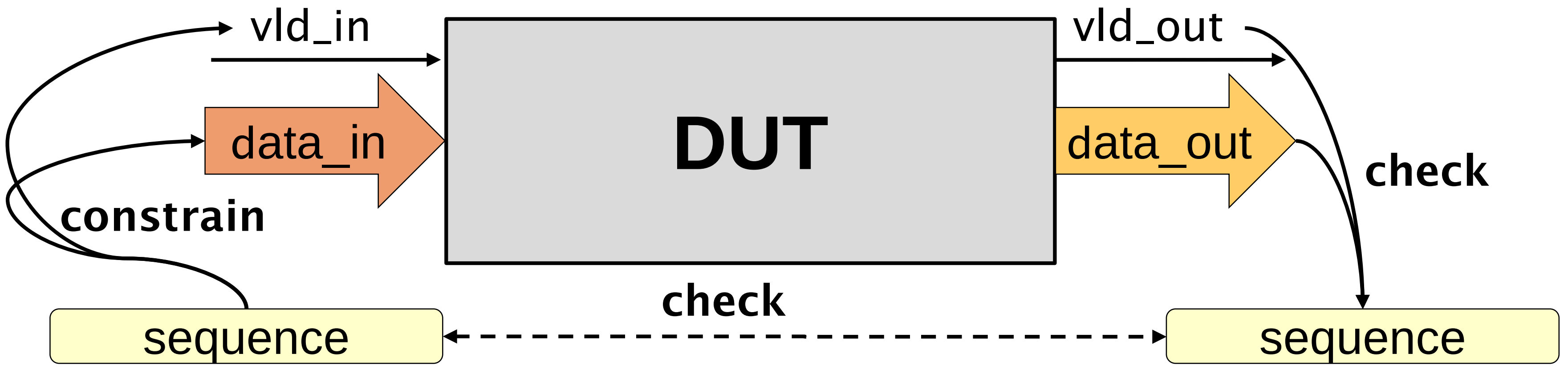
Protocol Verification

- 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



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

```
wire [31:0] symbol; // uninitialized
assert property($stable(symbol));
```

Value chosen by the formal tool to trigger failures!

- Example sequence „one symbol only“

00…00100..00

  - Symbol used to constrain unique value in input domain sequence

```
assume property (@(posedge in_clk)
    in_symbol_seen && in_dvalid |-> in_data != symbol);
```

  - Symbol used to check for matching same value in the output domain sequence

```
assert property (@(posedge out_clk)
    out_symbol_seen && out_dvalid |-> out_data != symbol);
```

Sequences of Symbols in Formal Scoreboard

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

* Stangier‘s approach

Definition:

S:	Symbol
.:	Anything but symbol
?:	Anything

cādence®

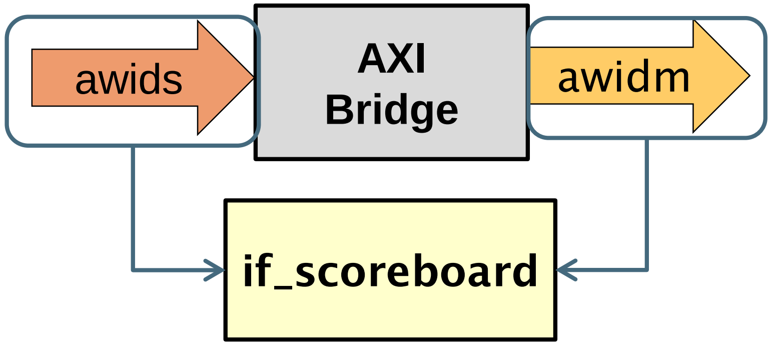
ST ERICSSON

Error Types Detected by Formal Scoreboard

1.	loss (arbitrary item)	XXCCX → XXCX
2.	loss all (items of particular value C)	XXCCX → XXX
3.	creation (arbitrary value)	XXXXX → XXXXEX
4.	creation (illegal value only)	XXXXX → XXXXYX
5.	duplication (same value only)	XXXCX → XXCCXX
6.	manipulation (to arbitrary value)	XXXCX → XXXEX
7.	manipulation (to illegal value only)	XXXCX → XXXYX
8.	reordering (arbitrary items)	XBCXX → XCBXX

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

Instantiating Formal Scoreboard



```
if_scoreboard #(
    // Parameters
    .DBUS_WIDTH   (ID_WIDTH),                // Size of the external datapath
  vector
    .CHECK_WIDTH  (CHECK_WIDTH)              // Size of the internal datapath
  check
) 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
);
```

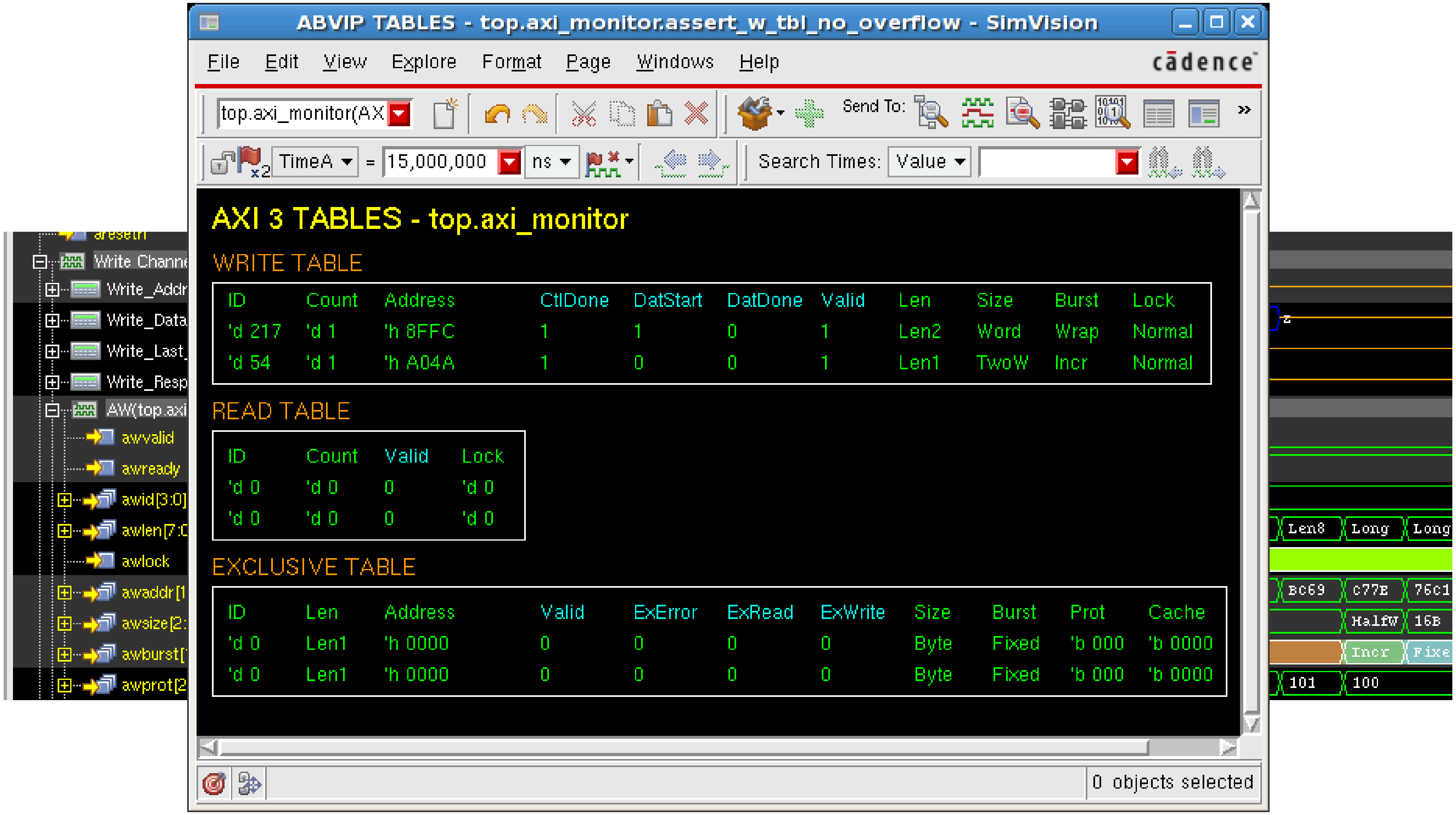
Datapaths in the Async AXI Bridge

- 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 pathes 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

ID	Count	Address	CtlDone	DatStart	DatDone	Valid	Len	Size	Burst	Lock
'd 217	'd 1	'h 8FFC	1	1	0	1	Len2	Word	Wrap	Normal
'd 54	'd 1	'h A04A	1	0	0	1	Len1	TwoW	Incr	Normal

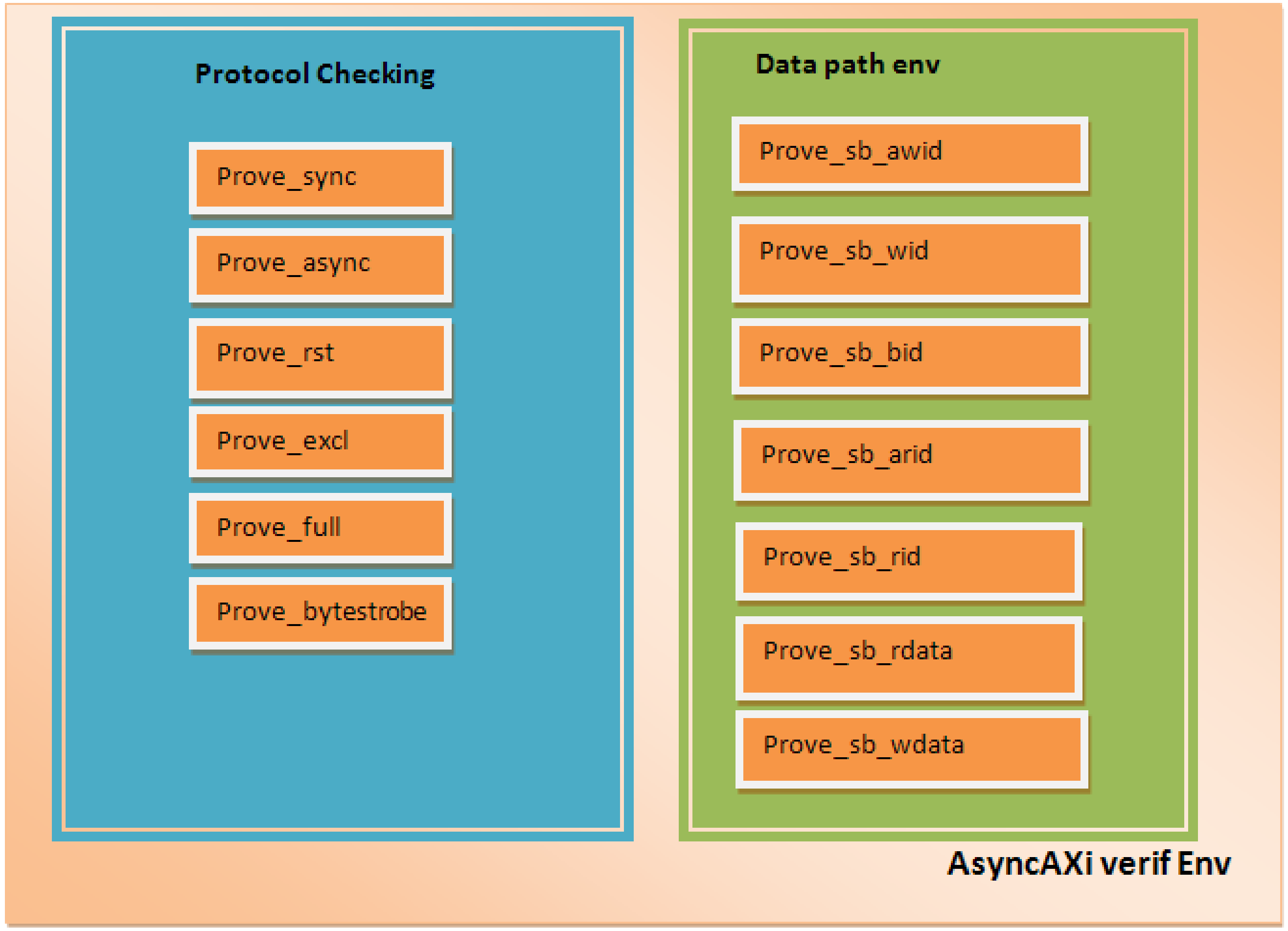
READ TABLE

ID	Count	Valid	Lock
'd 0	'd 0	0	'd 0
'd 0	'd 0	0	'd 0

EXCLUSIVE TABLE

ID	Len	Address	Valid	ExError	ExRead	ExWrite	Size	Burst	Prot	Cache
'd 0	Len1	'h 0000	0	0	0	0	Byte	Fixed	'b 000	'b 0000
'd 0	Len1	'h 0000	0	0	0	0	Byte	Fixed	'b 000	'b 0000

Regression Suite



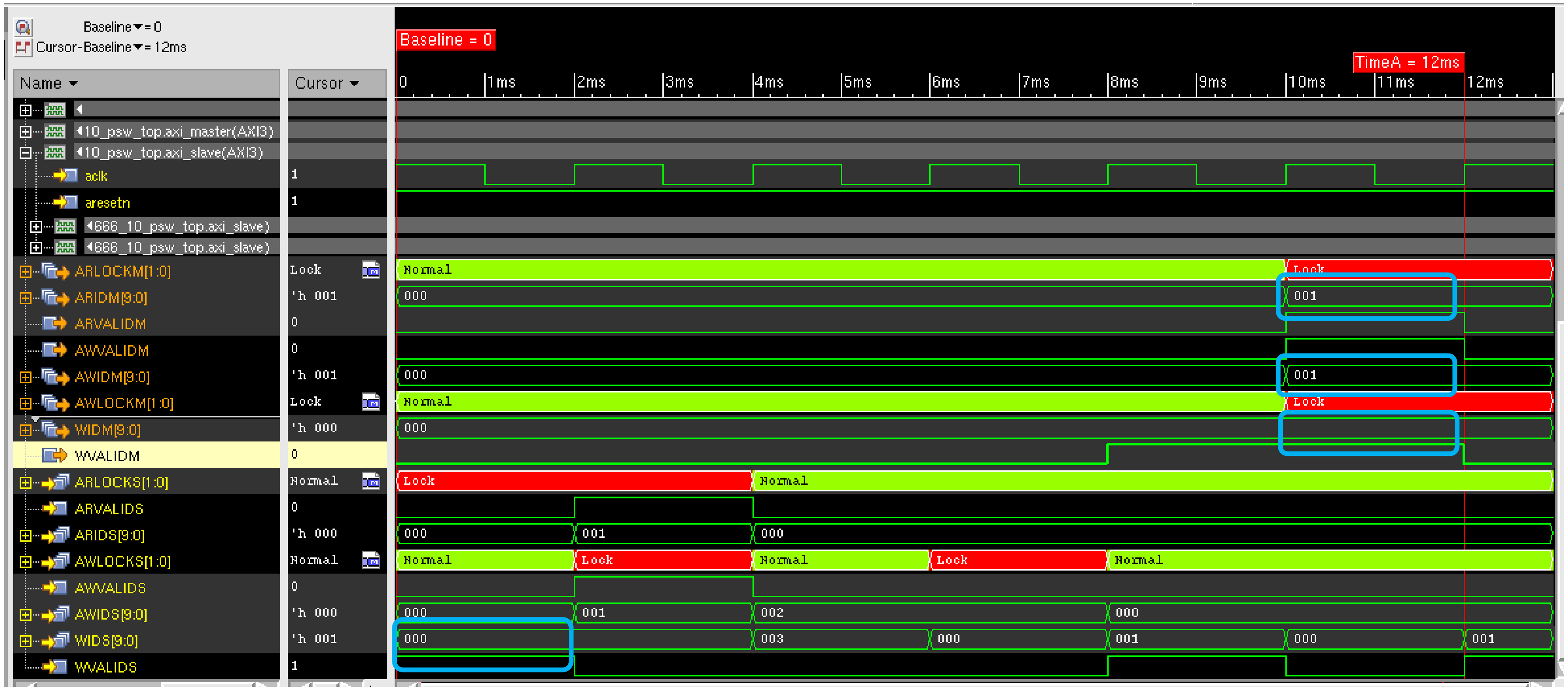
Comparing Formal Environments

	Config 1		Config2	
	Old	New	Old	New
Total	115	144	108	141
Pass	75 (65%)	108 (75%)	74 (68%)	109 (77%)
Fail	8 (7%)	9 (6%)	3 (3%)	9 (6%)
Explored*	32 (28%)	27 (19%)	31 (29%)	23 (16%)

* 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



Baseline = 0
Cursor-Baseline = 12ms
Name
Cursor
Baseline = 0
TimeA = 12ms
0 1ms 2ms 3ms 4ms 5ms 6ms 7ms 8ms 9ms 10ms 11ms 12ms
10_psw_top.axi_master(AXI3)
10_psw_top.axi_slave(AXI3)
aclk 1
aresetn 1
666_10_psw_top.axi_slave)
666_10_psw_top.axi_slave)
ARLOCKM[1:0] Lock — Normal, Lock
ARIDM[9:0] 'h 001 — 000, 001
ARVALIDM 0
AWVALIDM 0
AWIDM[9:0] 'h 001 — 000, 001
AWLOCKM[1:0] Lock — Normal, Lock
WIDM[9:0] 'h 000 — 000
WVALIDM 0
ARLOCKS[1:0] Normal — Lock, Normal
ARVALIDS 0
ARIDS[9:0] 'h 000 — 000, 001, 000
AWLOCKS[1:0] Normal — Normal, Lock, Normal, Lock, Normal
AWVALIDS 0
AWIDS[9:0] 'h 000 — 000, 001, 002, 000
WIDS[9:0] 'h 001 — 000, 003, 000, 001, 000, 001
WVALIDS 1

cadence®

ST ERICSSON

Summary

- Pro:
  - Positive experience with with formal verification, scoreboarding 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 ressources on verification

THANK YOU