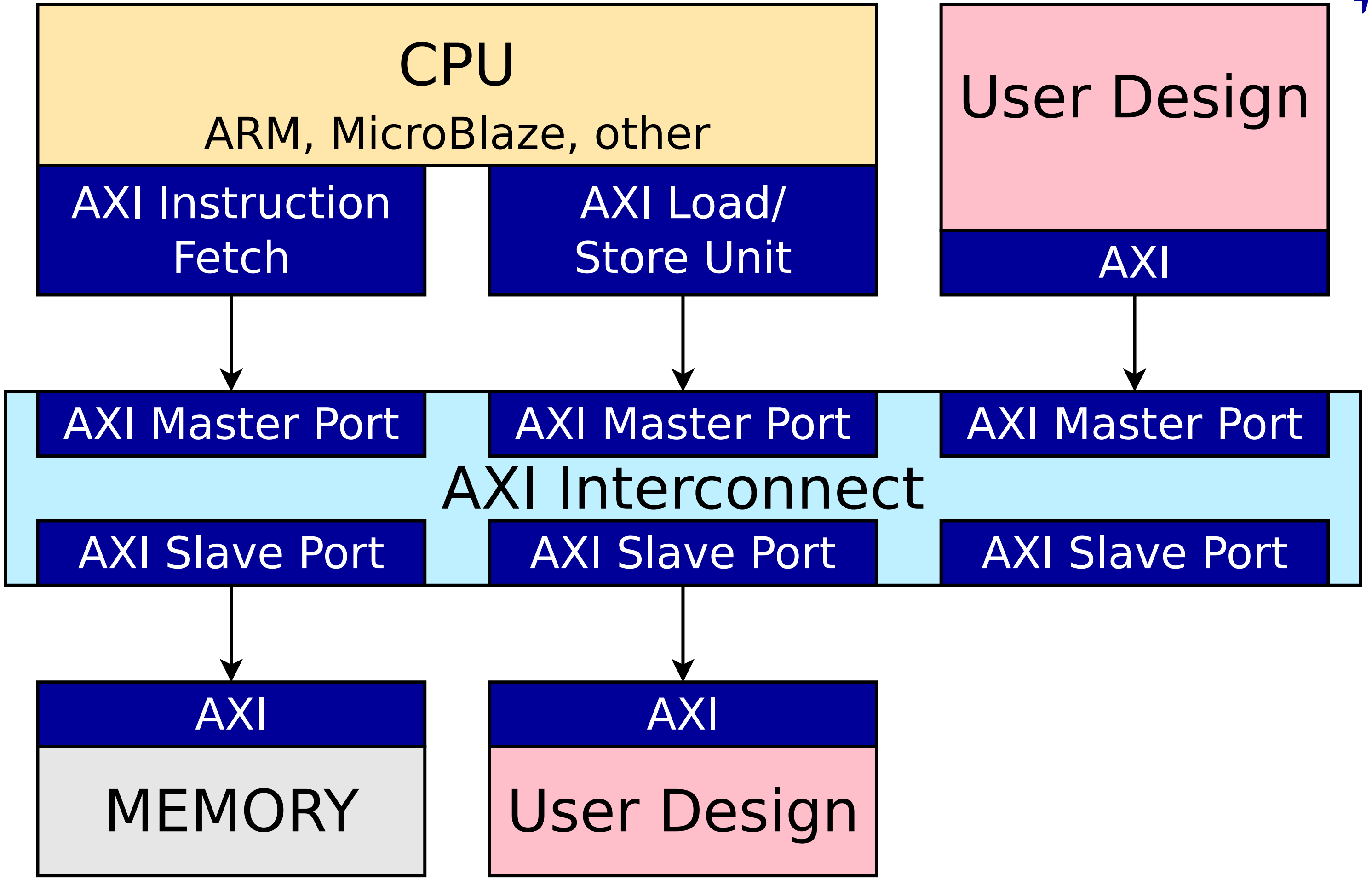
Gisselquist
Technology, LLC

Formally Verifying AXI interfaces

Daniel E. Gisselquist, Ph.D.
September, 2019

AXI is everywhere



The Challenge of AXI

AXI is the high speed bus in ARM's AMBA protocol suite

- You need to speak AXI to work with ARM microprocessors
  This includes ARM+FPGA SOC devices, such as Zynq
- It's high speed, cache aware, and very capable
- It's also difficult to get right
  One dropped transaction request or response, and your design will hang until the next reset
- It's even harder to do well

The Challenge of AXI

AXI is the high speed bus in ARM's AMBA protocol suite

- You need to speak AXI to work with ARM microprocessors
  This includes ARM+FPGA SOC devices, such as Zynq
- It's high speed, cache aware, and very capable
- It's also difficult to get right
  One dropped transaction request or response, and your design will hang until the next reset
- It's even harder to do well

AXI-Lite

- Simplified version of AXI
- Doesn't support burst transactions
- Responses must be returned in order
- No exclusive bus access support

How complex is it?

The Challenge of AXI

AXI is the high speed bus in ARM's AMBA protocol suite

- You need to speak AXI to work with ARM microprocessors
  This includes ARM+FPGA SOC devices, such as Zynq
- It's high speed, cache aware, and very capable
- It's also difficult to get right
  One dropped transaction request or response, and your design will hang until the next reset
- It's even harder to do well

AXI-Lite

- Simplified version of AXI
- Doesn't support burst transactions
- Responses must be returned in order
- No exclusive bus access support

How complex is it? *Neither Intel nor Xilinx got it right*

Formal Verification

Formal Verification is means of exhaustively searching for bugs

- □ Search over *all* possible design inputs

SymbiYosys supports three basic modes

1. Bounded Model Checks: Search the first N timesteps
   - □ Success is declared when the first fault is found
   - □ Black-Box, no internal design knowledge required
2. Induction Checks: Search N timesteps, starting anywhere
   - □ Success is declared when no fault can be found
   - □ White-Box testing, requires detailed internal knowledge
3. Cover
   - □ Success is finding a trace to match each goal
   - □ The tool will find the first trace that matches each

Overview

Everywhere
Challenge
Formal
▷ Overview
Properties
Test Set
Bugs Found
Xilinx's AXI-lite demo
Personal Examples
AXI-lite is powerful
The Problem
Conclusions
Backup Slides
How Long?
Initial
Full AXI Properties
Intel Bugs
Xilinx Bugs
Xilinx's AXI demo
Better AXI demo
Stories
Bus Fault Isolator

AXI-Lite

- 4 Basic AXI-Lite properties
- Examples of bugs found
- Traces from demonstration designs

Backups

- How long does a proof take?
- Initial statements and resets
- AXI (full) properties
- Examples of bugs found
- Traces from demonstration designs
- Customer experiences, from forums.xilinx.com
- Bus Fault Isolator

Formal Properties

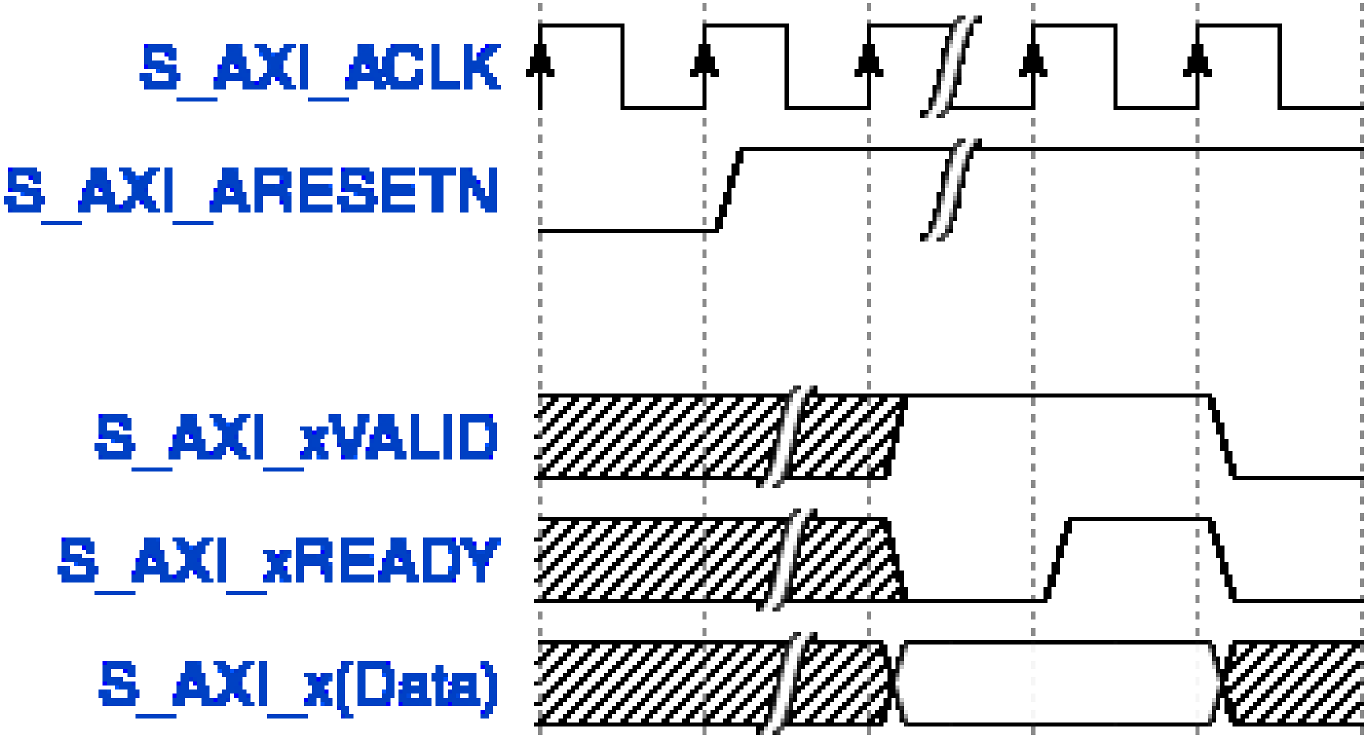
1. Reset clears all requests

```
always @(*)
if (!f_past_valid || !$past(S_AXI_ARESETN))
begin
        // Make assumptions about inputs
        assume(!S_AXI_ARVALID);
        // Make assertions about outputs
        assert(!S_AXI_BVALID);
end
```

I have a similar set of properties that reverses the assumptions and assertions–useful for verifying bus masters

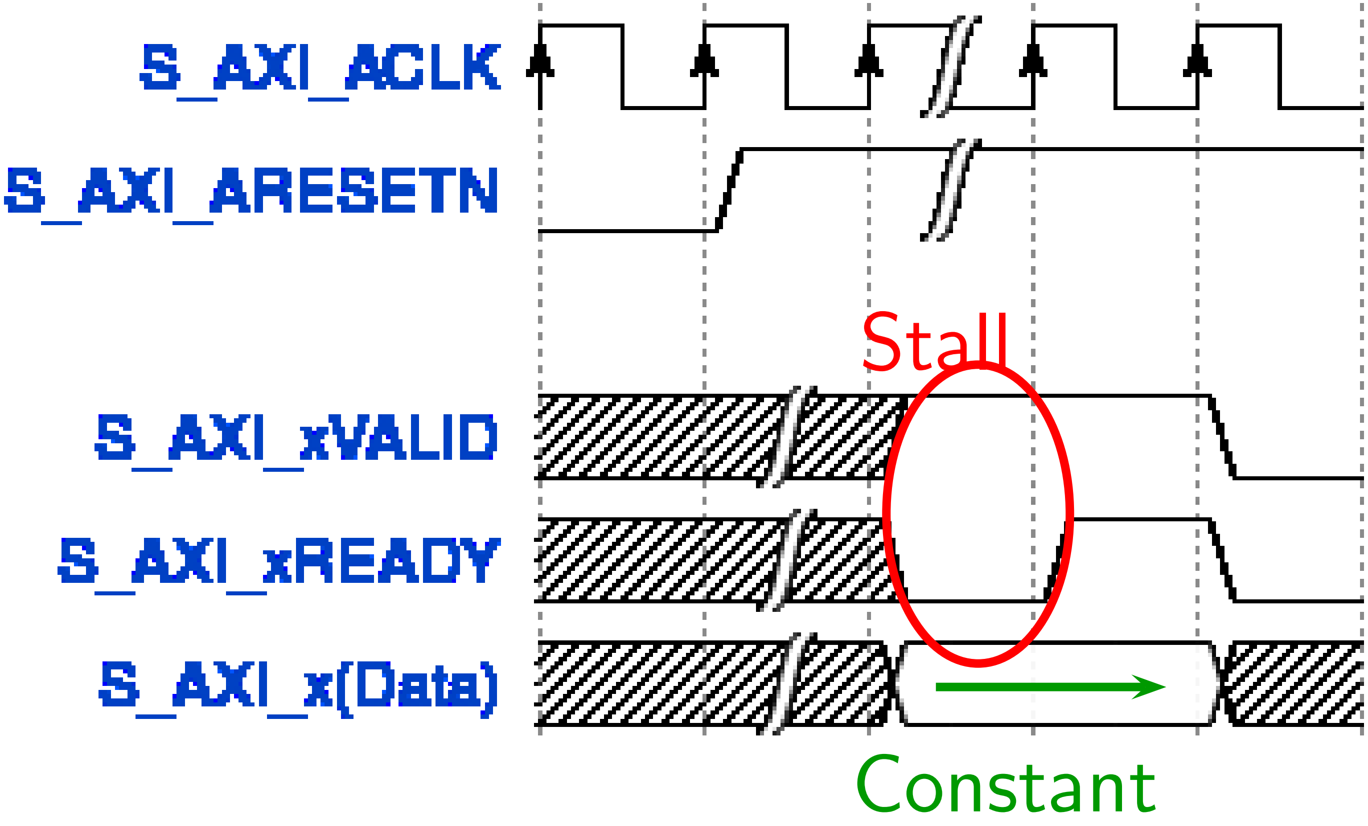
Formal Properties

1. Reset clears all requests
2. If a channel is stalled, nothing should change



Formal Properties

1. Reset clears all requests
2. If a channel is stalled, nothing should change



Formal Properties

1. Reset clears all requests
2. If a channel is stalled, nothing should change

```
always @(posedge S_AXI_ACLK)
if (S_AXI_ARESETN && $past(S_AXI_ARESETN
        && S_AXI_AWVALID && !S_AXI_AWREADY))
begin
        assert(S_AXI_AWVALID);
        assert($stable(S_AXI_AWADDR));
        // ...
```

Formal Properties

Everywhere
Challenge
Formal
Overview
▷ Properties
Test Set
Bugs Found
Xilinx's AXI-lite demo
Personal Examples
AXI-lite is powerful
The Problem
Conclusions
Backup Slides
How Long?
Initial
Full AXI Properties
Intel Bugs
Xilinx Bugs
Xilinx's AXI demo
Better AXI demo
Stories
Bus Fault Isolator

1. Reset clears all requests
2. If a channel is stalled, nothing should change
3. No responses without prior requests

- Count the outstanding requests
- Don't allow any responses unless the counter is non-zero

```
always @(*)
if (writes_outstanding == 0)
        assert(!S_AXI_BVALID);
```

- Watch out for overflow!

Formal Properties

1. Reset clears all requests
2. If a channel is stalled, nothing should change
3. No responses without prior requests
4. Every request should get one (and only one) response
    - Requires implementing a timeout
    - All responses must be returned within a given number of clock cycles
        - Really three counters: Maximum number of stall cycles, maximum delay before response, maximum master response stall cycles
        - Can't be expected to respond or lower stall when the return channel is stalled
        - Stall = `xVALID && !xREADY;`

Formal Properties

1. Reset clears all requests
2. If a channel is stalled, nothing should change
3. No responses without prior requests
4. Every request should get one (and only one) response

Just four basic criteria

- . . . and a couple smaller ones

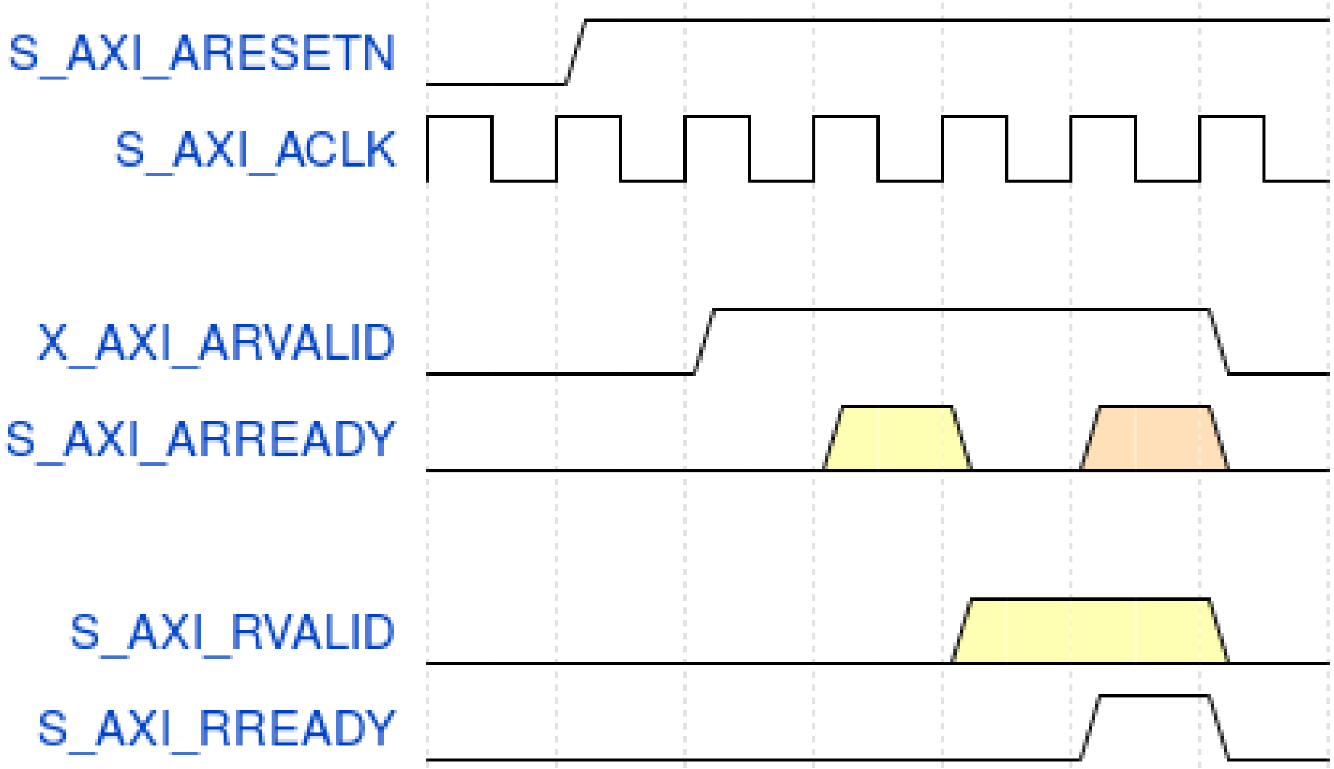
Link: AXI-lite properties

Test Set

- It doesn't work if it's never tested
  - Xilinx
    - Xilinx's demonstration cores
    - Xilinx's GPIO core
  - Cores from Github
    - Wishbone - AXI bridge
    - TinyTPU
    - (Major Vendor)
  - My own cores
- Full formal verification is *not* black box testing
  - Takes some work to set up
  - Xilinx's cores and their derivates are easy to set up

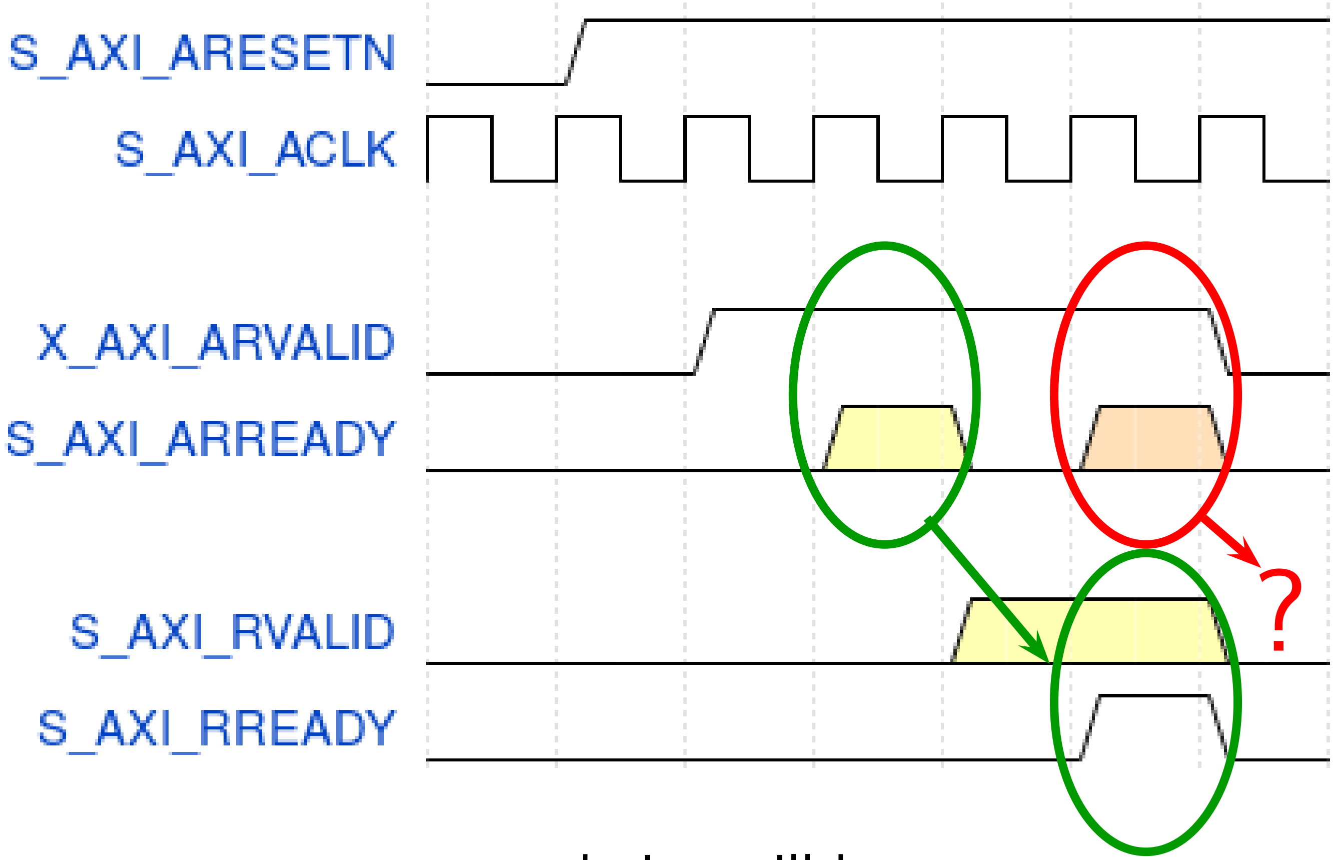
Bugs Found

Vivado 2016.1, AXI-lite READ



Bugs Found

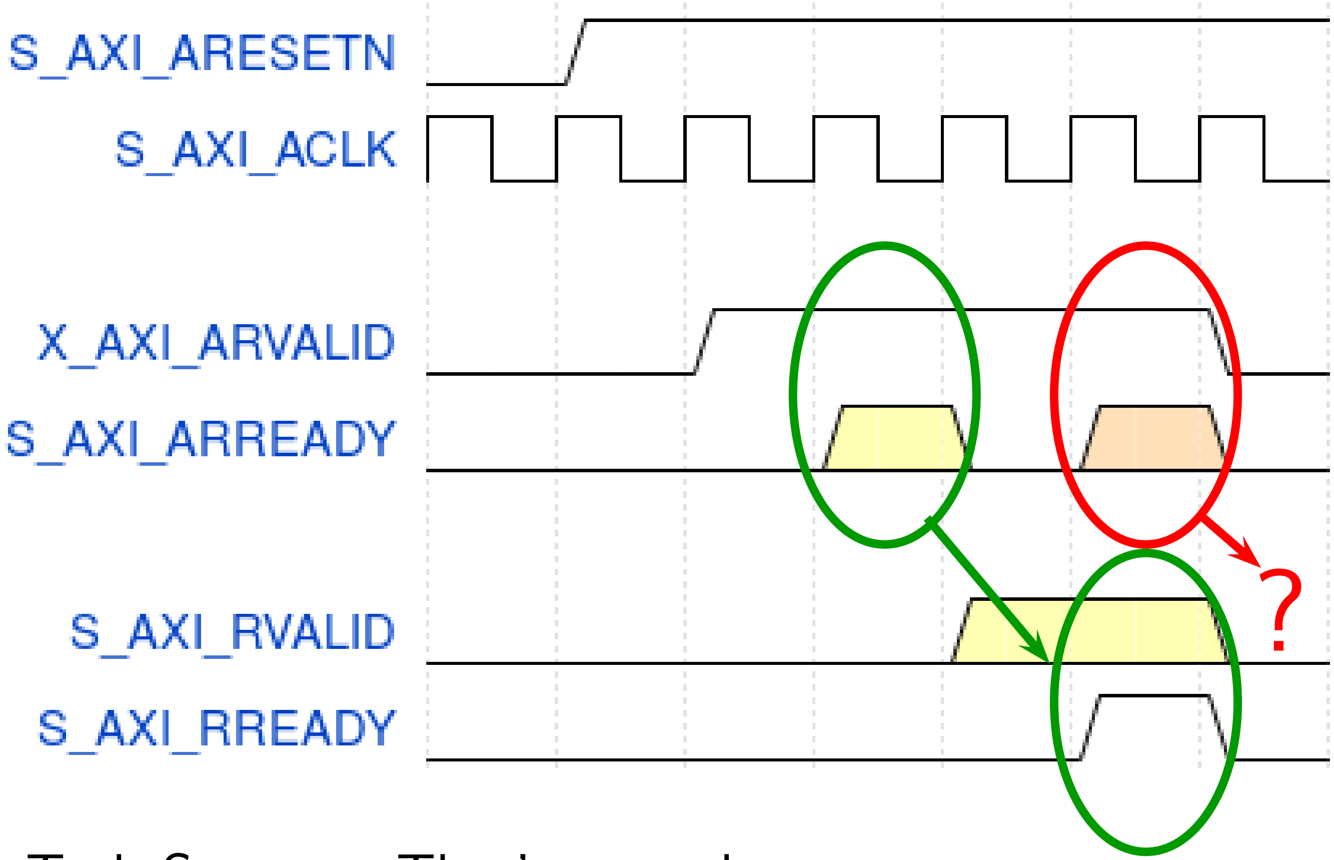
Vivado 2016.1, AXI-lite READ



Two requests, one response: design will hang

Bugs Found

Vivado 2016.1, AXI-lite READ

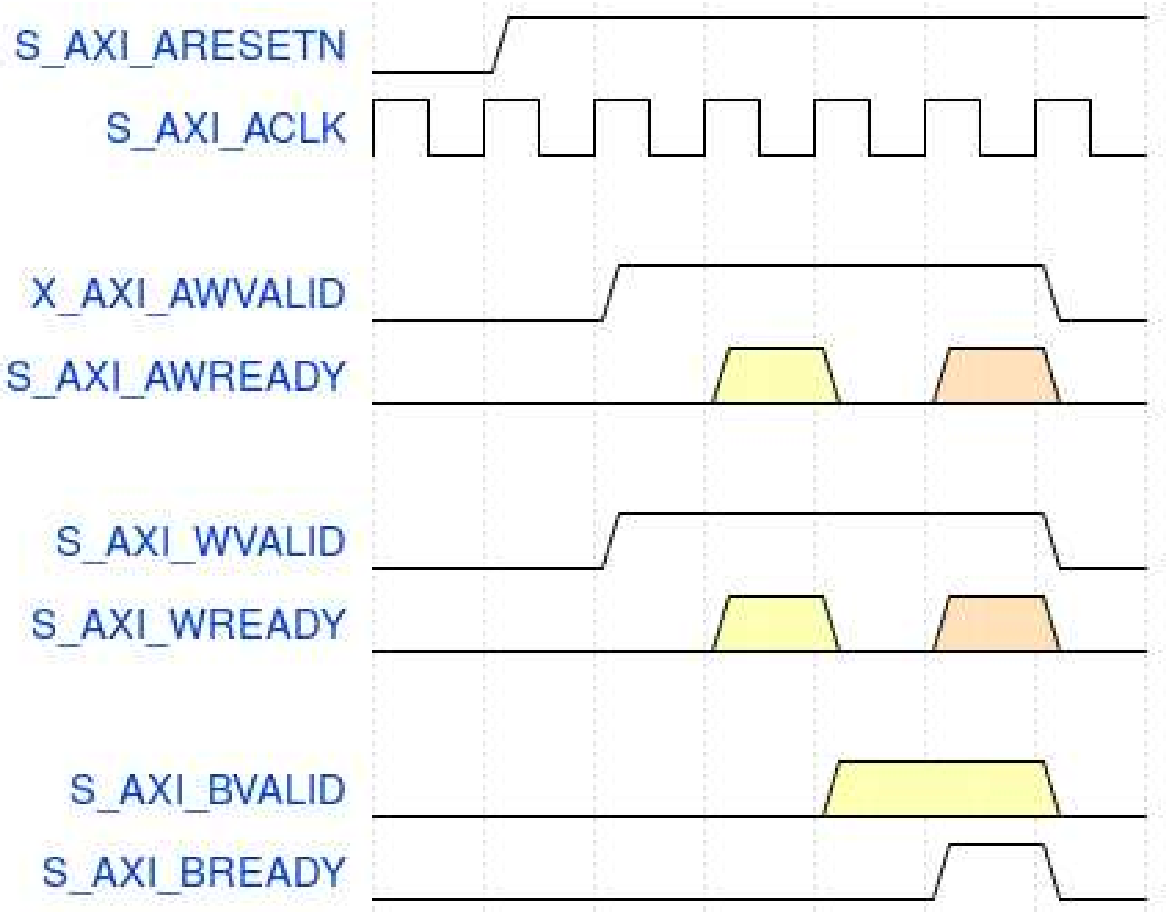


Xilinx Tech Support: That's not a bug

Bugs Found

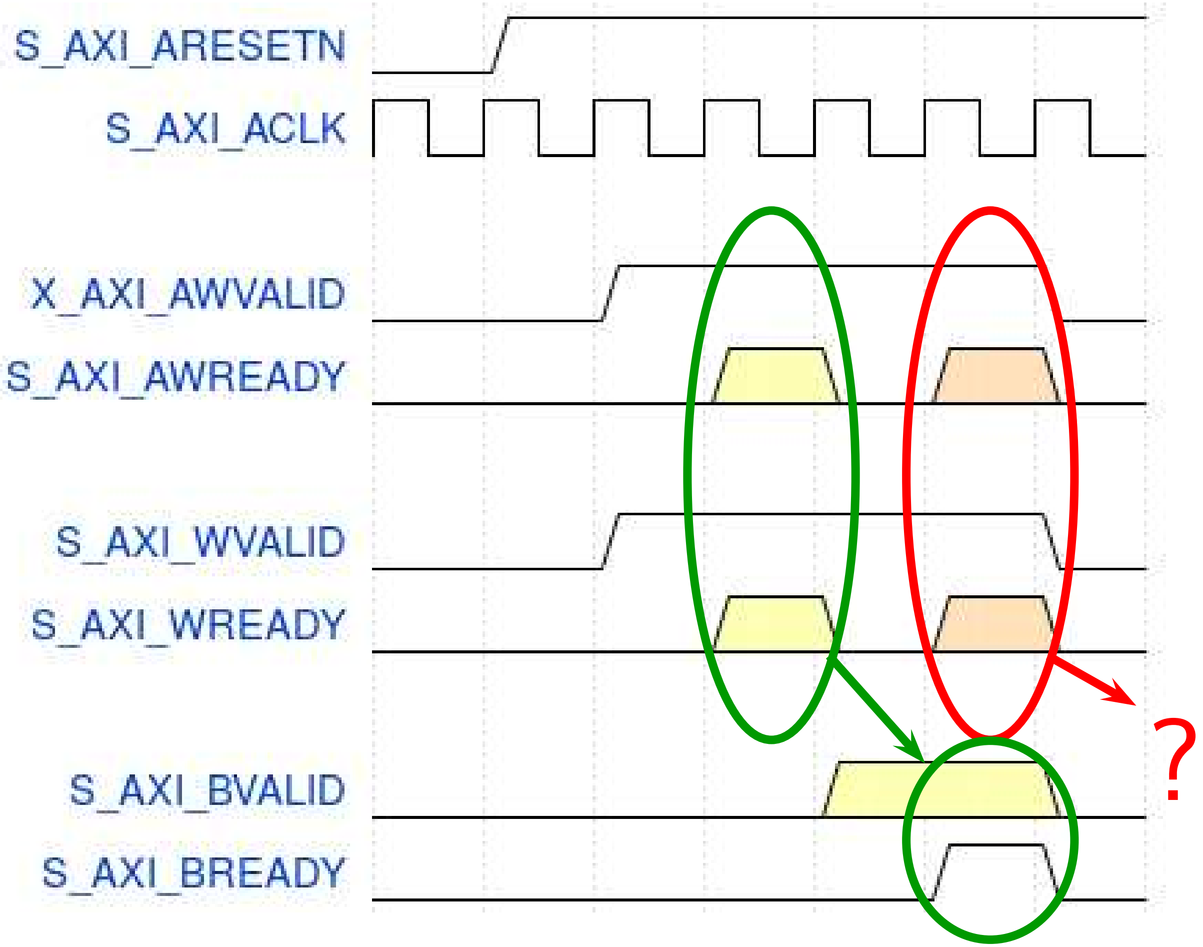
Everywhere
Challenge
Formal
Overview
Properties
Test Set
▻ Bugs Found
Xilinx's AXI-lite demo
Personal Examples
AXI-lite is powerful
The Problem
Conclusions
Backup Slides
How Long?
Initial
Full AXI Properties
Intel Bugs
Xilinx Bugs
Xilinx's AXI demo
Better AXI demo
Stories
Bus Fault Isolator

Vivado 2016.1, AXI-lite WRITE



Bugs Found

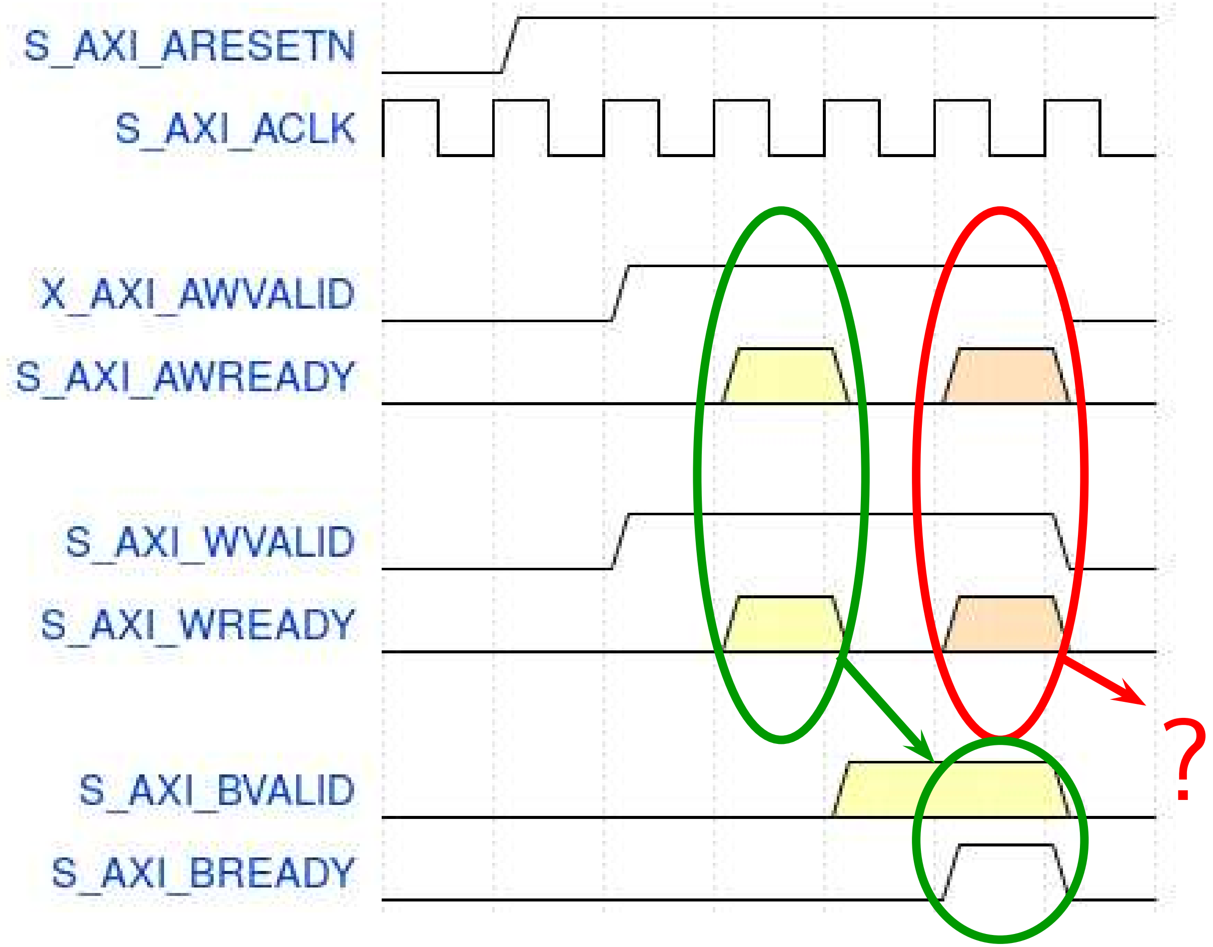
Vivado 2016.1, AXI-lite WRITE



Two write requests, one response: design will hang

Bugs Found

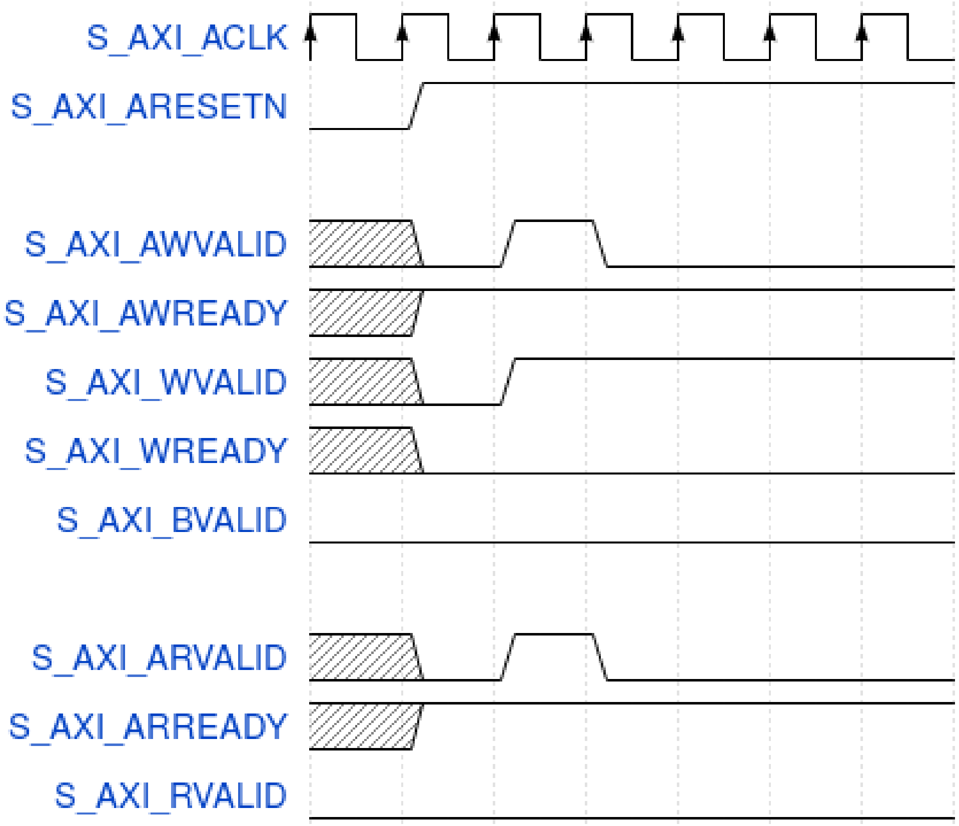
Vivado 2016.1, AXI-lite WRITE



As of 2019.1, only one of these bugs was fixed

Bugs Found

From the TinyTPU:

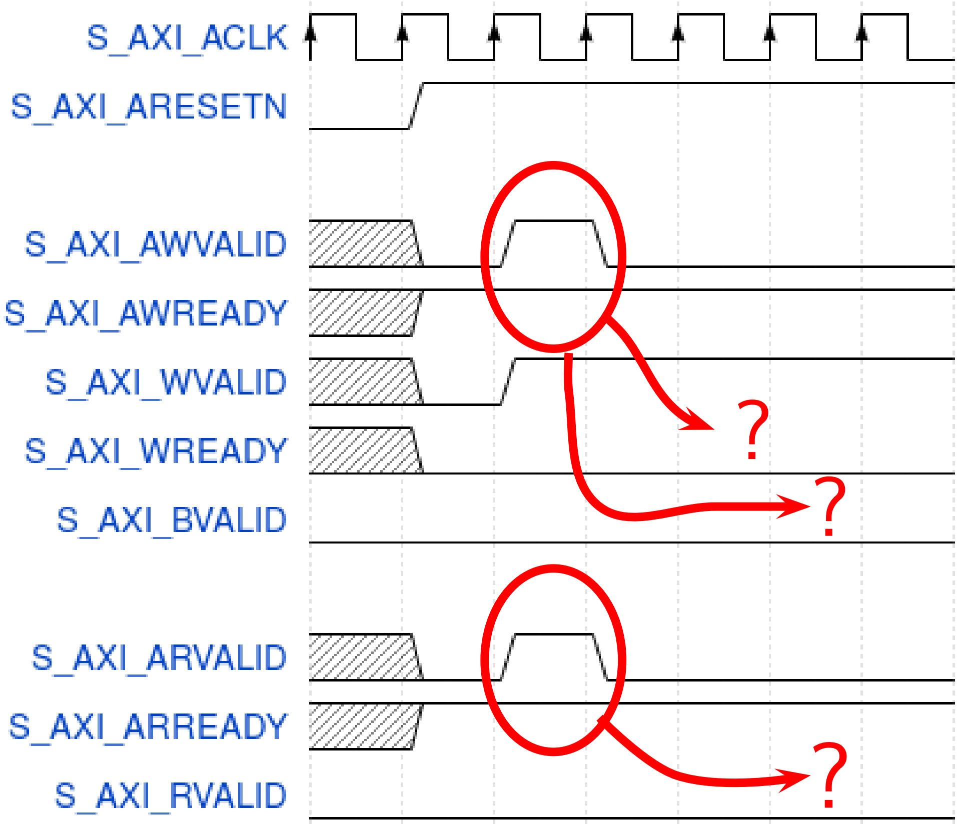


If ever `AWVALID` and `ARVALID` are both true at once . . .

- The state machine remains in idle

Bugs Found

From the TinyTPU:

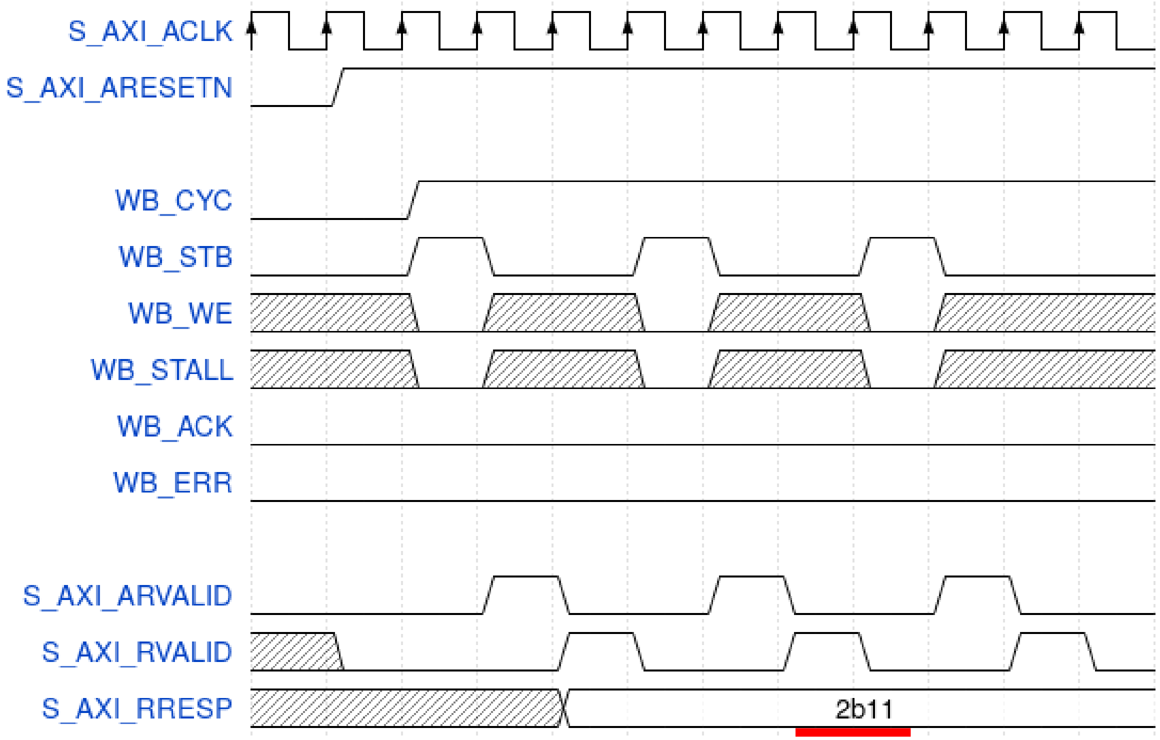


If ever `AWVALID` and `ARVALID` are both true at once . . .

- The state machine remains in idle

Bugs Found

From a WishboneAXI bridge project

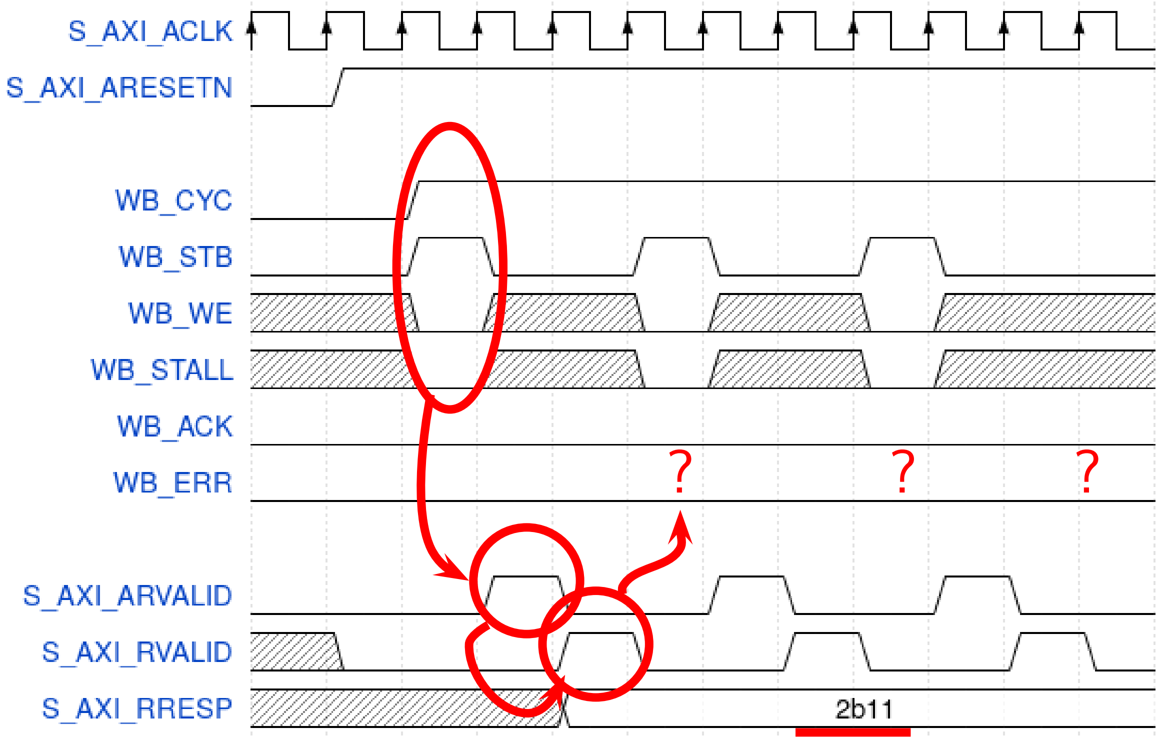


□ 2'b01: EXOKAY (Illegal in AXI-lite)

□ 2'b11: DECERR (*Return was dropped*)

Bugs Found

From a WishboneAXI bridge project – since fixed



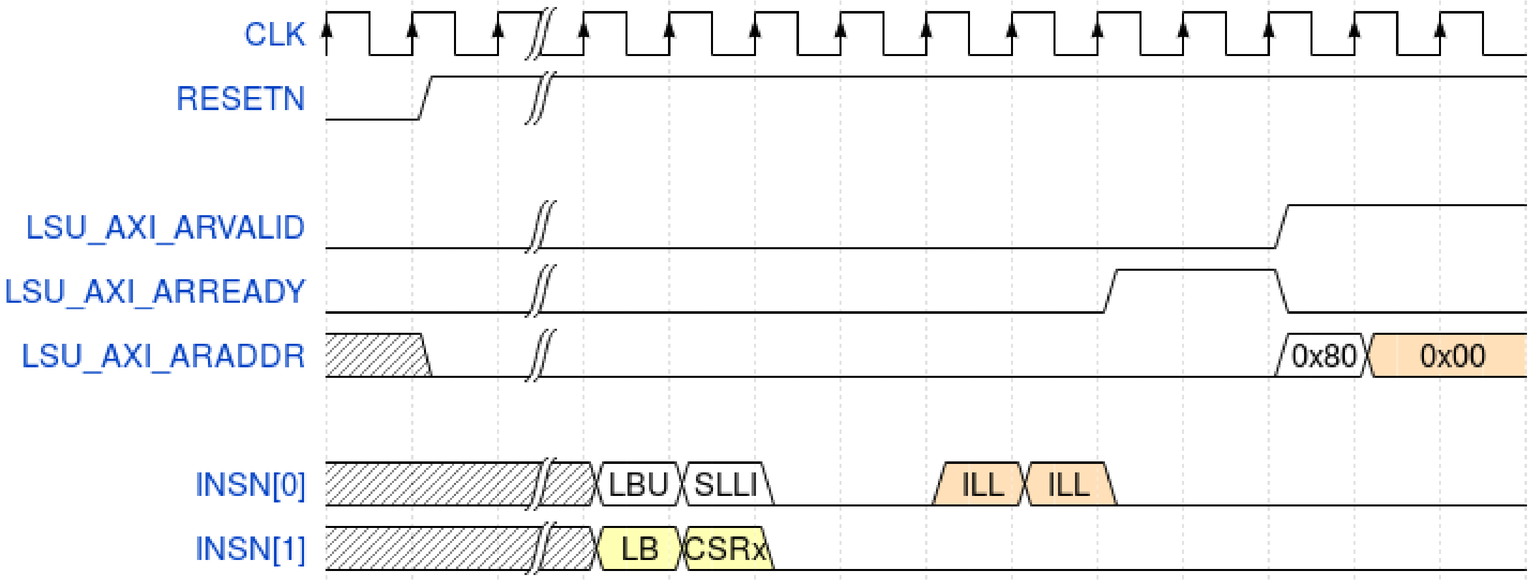
Unchecked return codes

- 2'b01: EXOKAY (Illegal in AXI-lite)
- 2'b11: DECERR (*Return was dropped*)

Bugs Found

Everywhere
Challenge
Formal
Overview
Properties
Test Set
▷ Bugs Found
Xilinx's AXI-lite demo
Personal Examples
AXI-lite is powerful
The Problem
Conclusions
Backup Slides
How Long?
Initial
Full AXI Properties
Intel Bugs
Xilinx Bugs
Xilinx's AXI demo
Better AXI demo
Stories
Bus Fault Isolator

Github: (Major vendor) didn't believe formal added any value



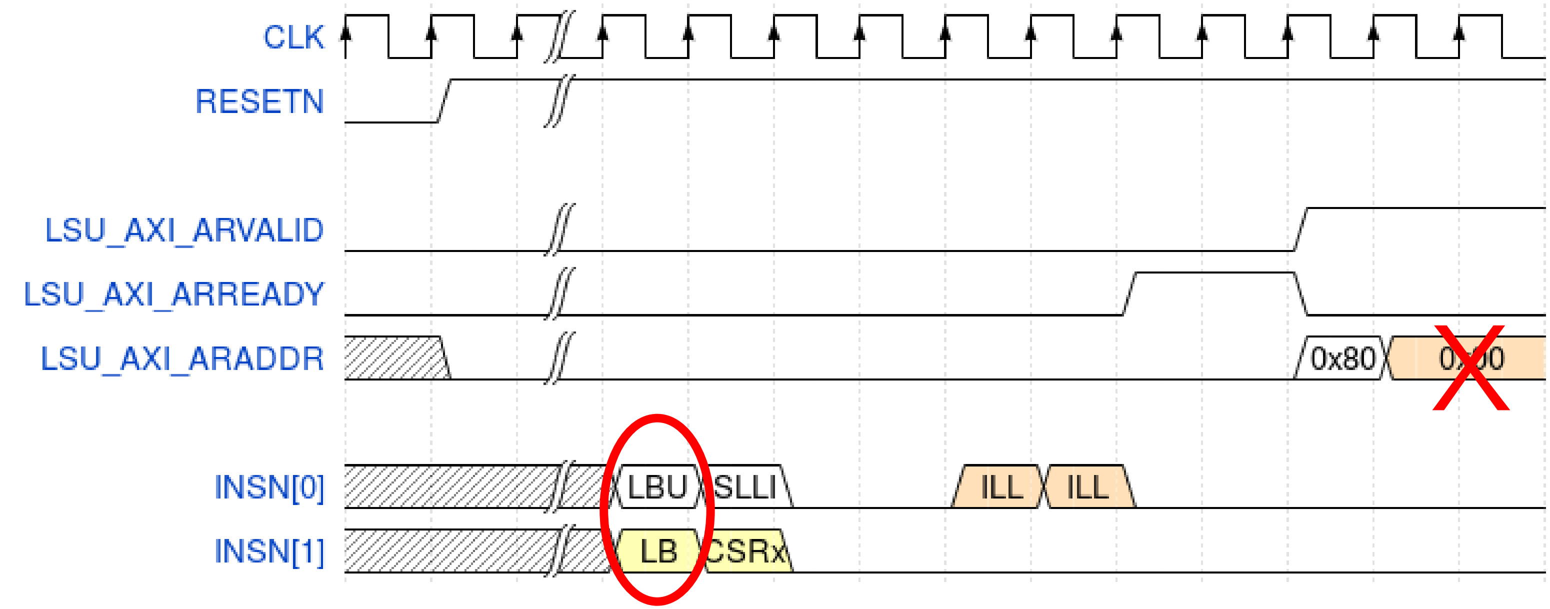
A dual-instruction-issue CPU

- □
- □
- □

Bugs Found

Everywhere
Challenge
Formal
Overview
Properties
Test Set
▷ Bugs Found
Xilinx's AXI-lite demo
Personal Examples
AXI-lite is powerful
The Problem
Conclusions
Backup Slides
How Long?
Initial
Full AXI Properties
Intel Bugs
Xilinx Bugs
Xilinx's AXI demo
Better AXI demo
Stories
Bus Fault Isolator

Github: (Major vendor) didn't believe formal added any value



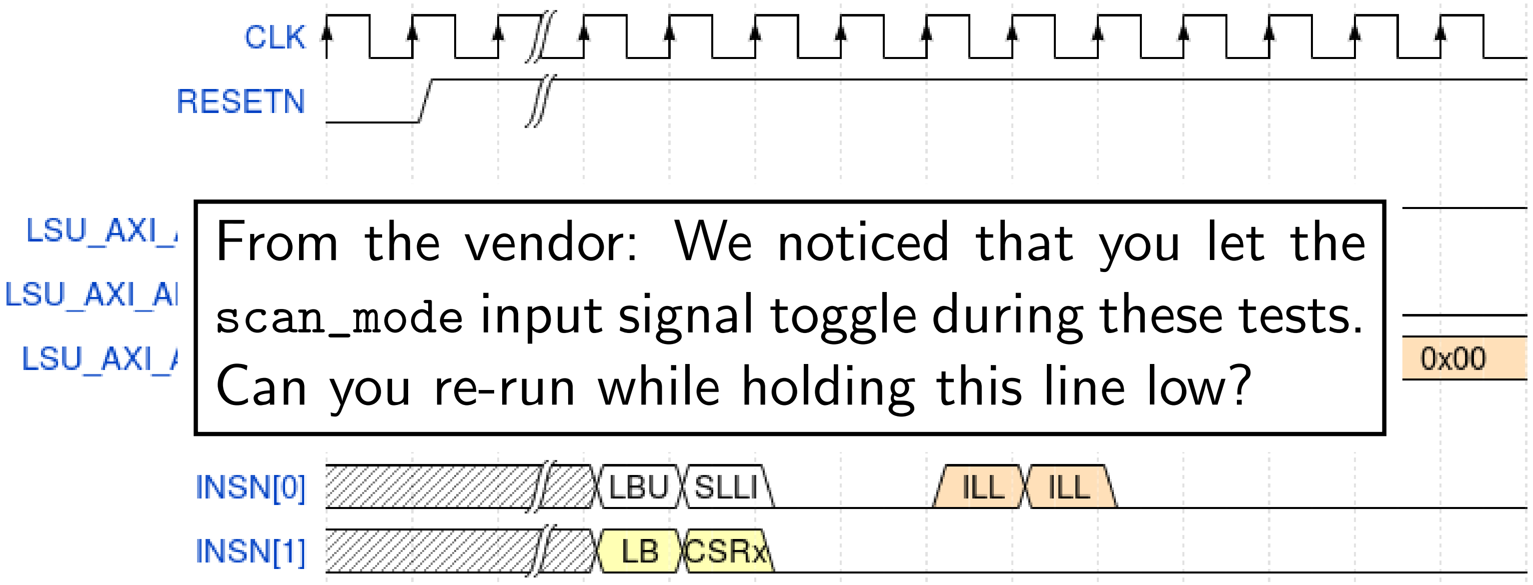
A dual-instruction-issue CPU

- Found over six protocol violations
- Most revolved around issuing adjacent load/store insns
- Their own assertions: passed simulation, failed formal

Bugs Found

Everywhere
Challenge
Formal
Overview
Properties
Test Set
▻ Bugs Found
Xilinx's AXI-lite demo
Personal Examples
AXI-lite is powerful
The Problem
Conclusions
Backup Slides
How Long?
Initial
Full AXI Properties
Intel Bugs
Xilinx Bugs
Xilinx's AXI demo
Better AXI demo
Stories
Bus Fault Isolator

Github: (Major vendor) didn't believe formal added any value



From the vendor: We noticed that you let the `scan_mode` input signal toggle during these tests. Can you re-run while holding this line low?

A dual-instruction-issue CPU

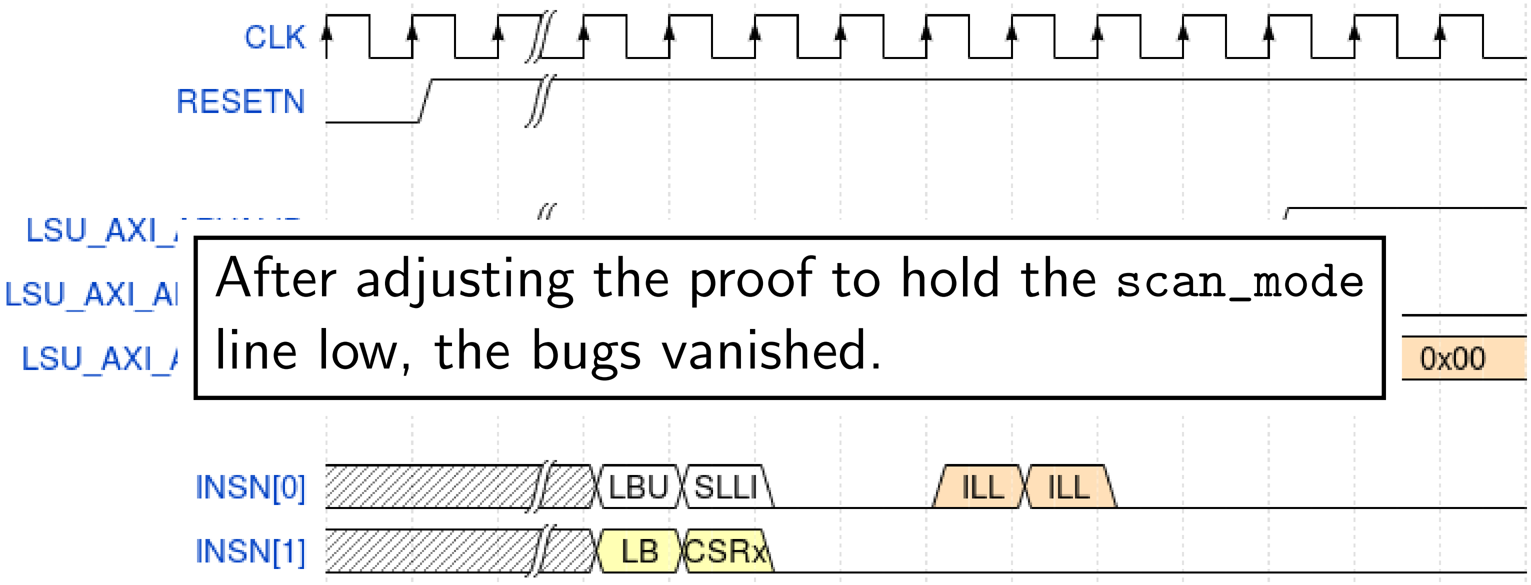
□ `scan_mode`?

□ Oops!

□

Bugs Found

Github: (Major vendor) didn't believe formal added any value



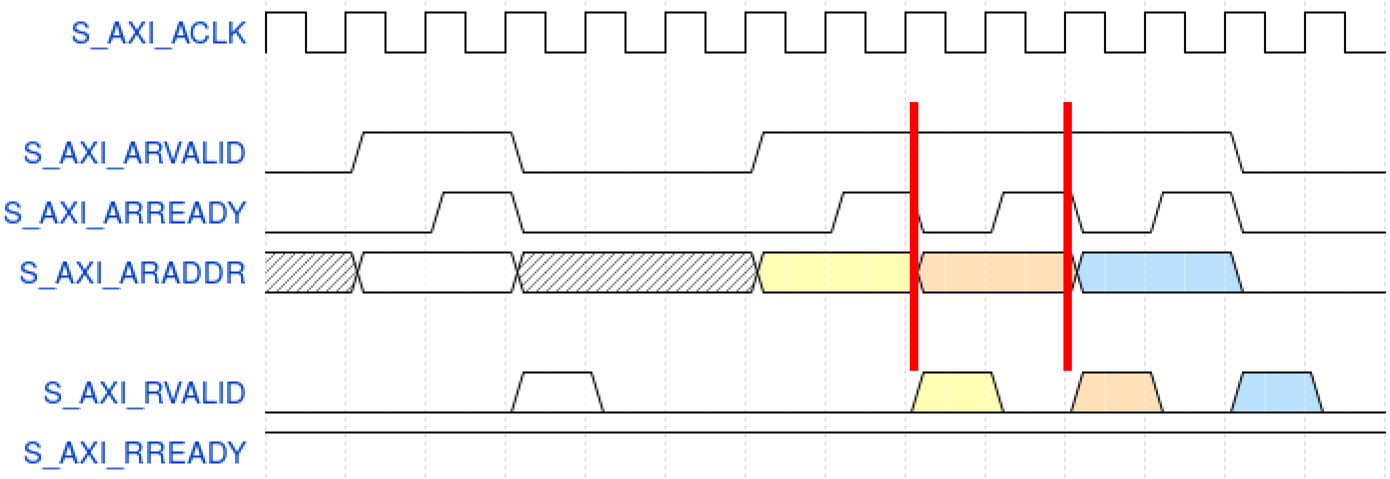
After adjusting the proof to hold the `scan_mode` line low, the bugs vanished.

A dual-instruction-issue CPU

- Violations vanished – Doh!
- (There's still an internal error that hasn't been resolved)

Xilinx's AXI-lite demo

ARM recommends leaving `xREADY` high when possible



Xilinx's demo doesn't follow this recommendation

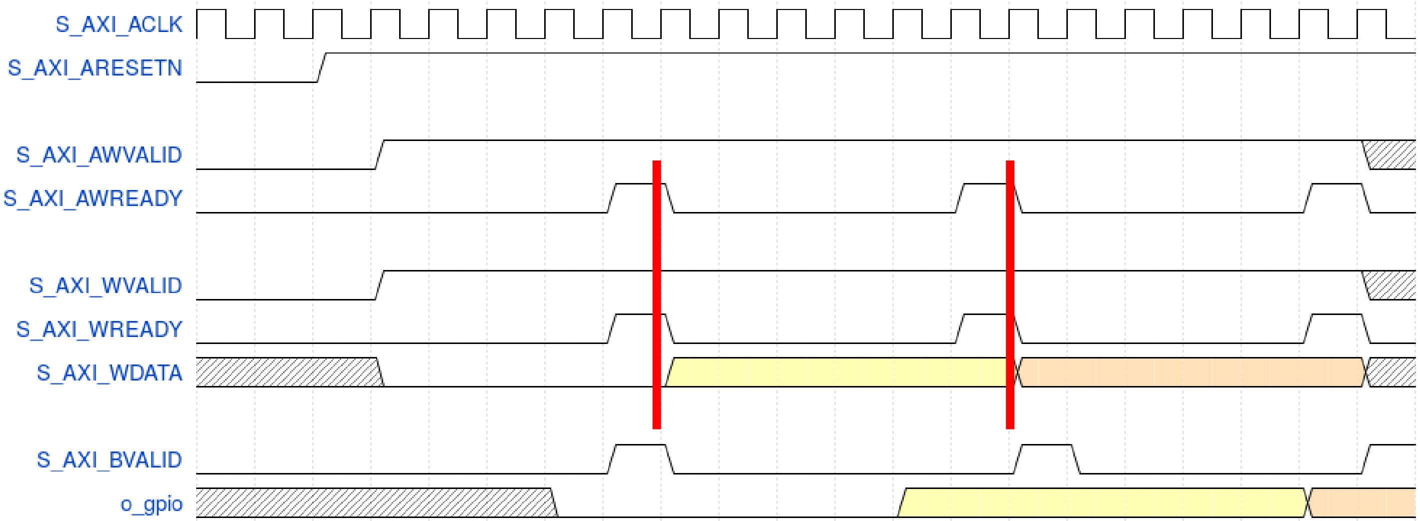
- Their demonstration design gets at best 50% throughput

Link: Xilinx's 2016.1 AXI-lite demonstration design

Xilinx's AXI GPIO

cover() can be used to determine a core's performance

- cover(third_write_return);



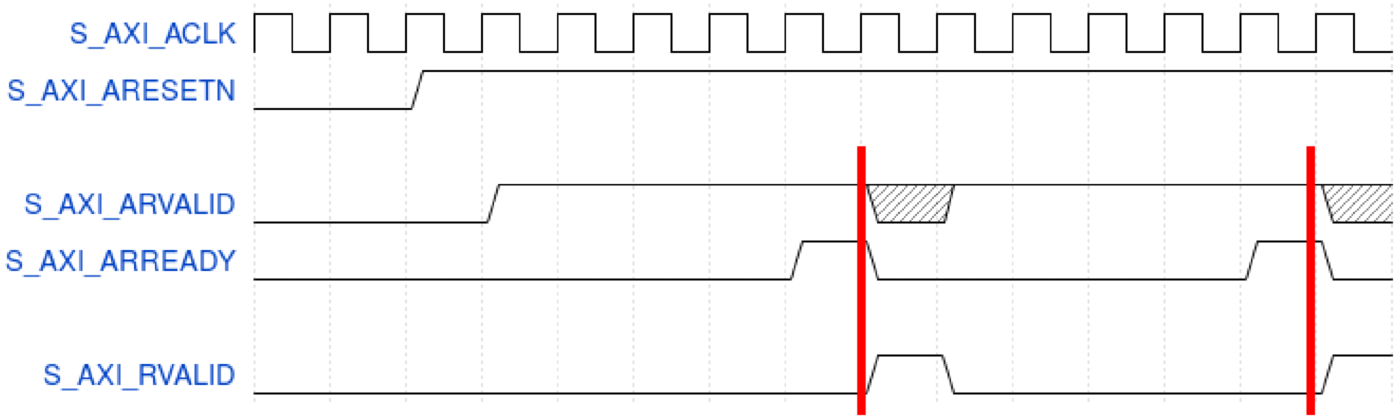
Max write throughput: One transaction every six clocks

- This would contribute to why so many folk complain of blinky being slow when using either ARM or MicroBlaze

Xilinx's AXI GPIO

cover() can be used to determine a core's performance

- □ **cover**(`second_read_return`);



Max read throughput: One transaction every six clocks

Personal Examples

My own designs may be found on Github

- AXI-lite demonstration design
- (Full) AXI demonstration design
- AXI-lite to AXI bridge
  - Unlike Xilinx's design, costs no logic, and no clocks
- AXI to AXI-lite bridge
  - This is really a high performance, high throughput design
  - Keeps up with the pace of AXI, no lost beats
- Crossbars: AXI, AXI-lite (Wishbone pipeline)
- AXI Stream to Memory Mapped (S2MM)
- AXI Memory Mapped to Stream (MM2S)

Yes, these are all Open Source

Personal Examples

My own designs may be found on Github

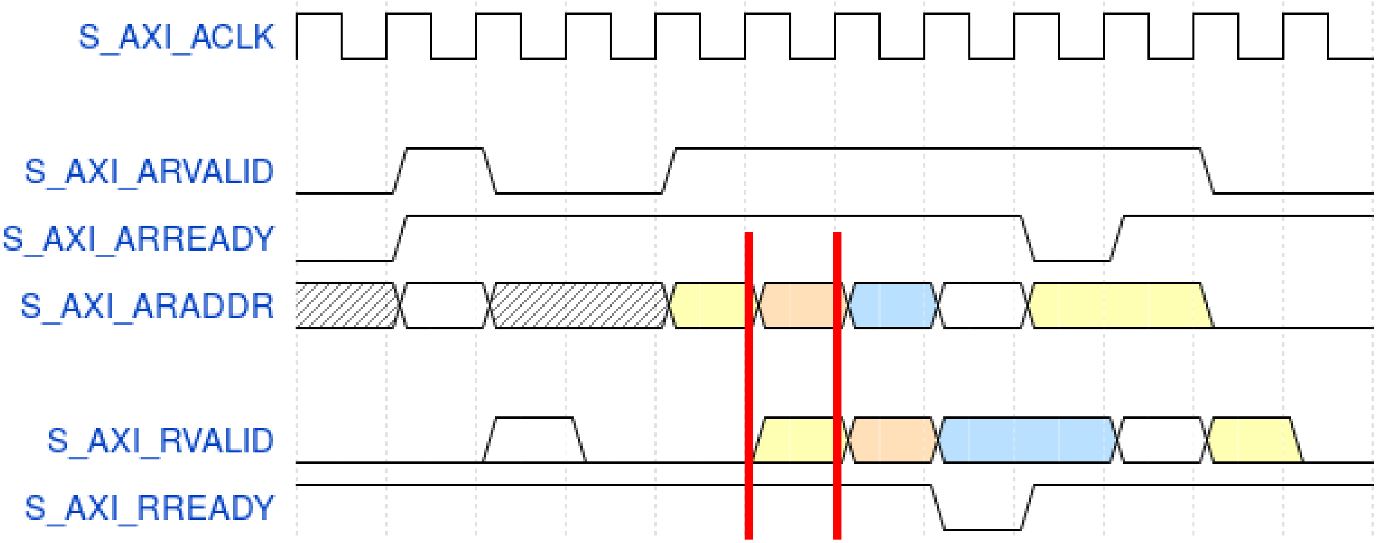
- □ AXI-lite demonstration design
- □ (Full) AXI demonstration design
- □ AXI-lite to AXI bridge
  - – Unlike Xilinx's design, costs no logic, and no clocks
- □ AXI to AXI-lite bridge
  - – This is really a high performance, high throughput design
  - – Keeps up with the pace of AXI, no lost beats
- □ Crossbars: AXI, AXI-lite (Wishbone pipeline)
- □ AXI Stream to Memory Mapped (S2MM)
- □ AXI Memory Mapped to Stream (MM2S)

Yes, these are all Open Source

- □ *The full AXI DMA remains dreamware*

OpenSource AXI-lite demo

ARM recommends leaving `xREADY` high when possible



- The trace above was generated from a **cover**() statement
- This core gets twice the throughput–*without the bugs*

Link: AXI-lite demonstration design

AXI-lite is powerful

If you want speed, AXI-lite makes throughput easier than AXI

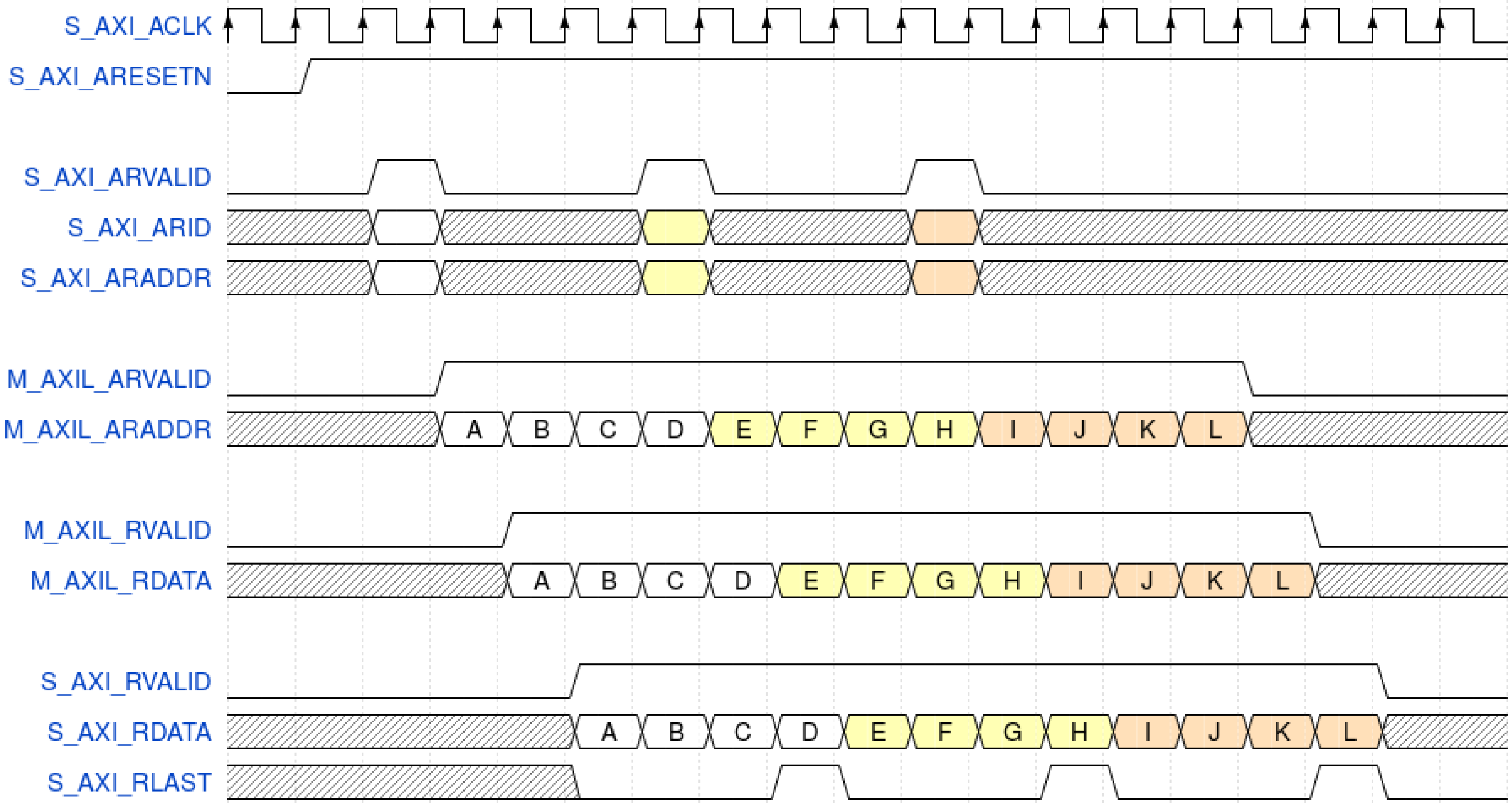
- Costs nothing to bridge from AXI-lite to AXI

Xilinx's examples tend to artificially limit AXI-lite

- Their AXI to AXI-lite bridge only allows one burst in flight
  - Artificially restricts that burst to either read or write
- Demo AXI-lite core drops throughput 50%
- GPIO core drops throughput 83%

(Their example AXI(full) core similarly cripples AXI as well.)

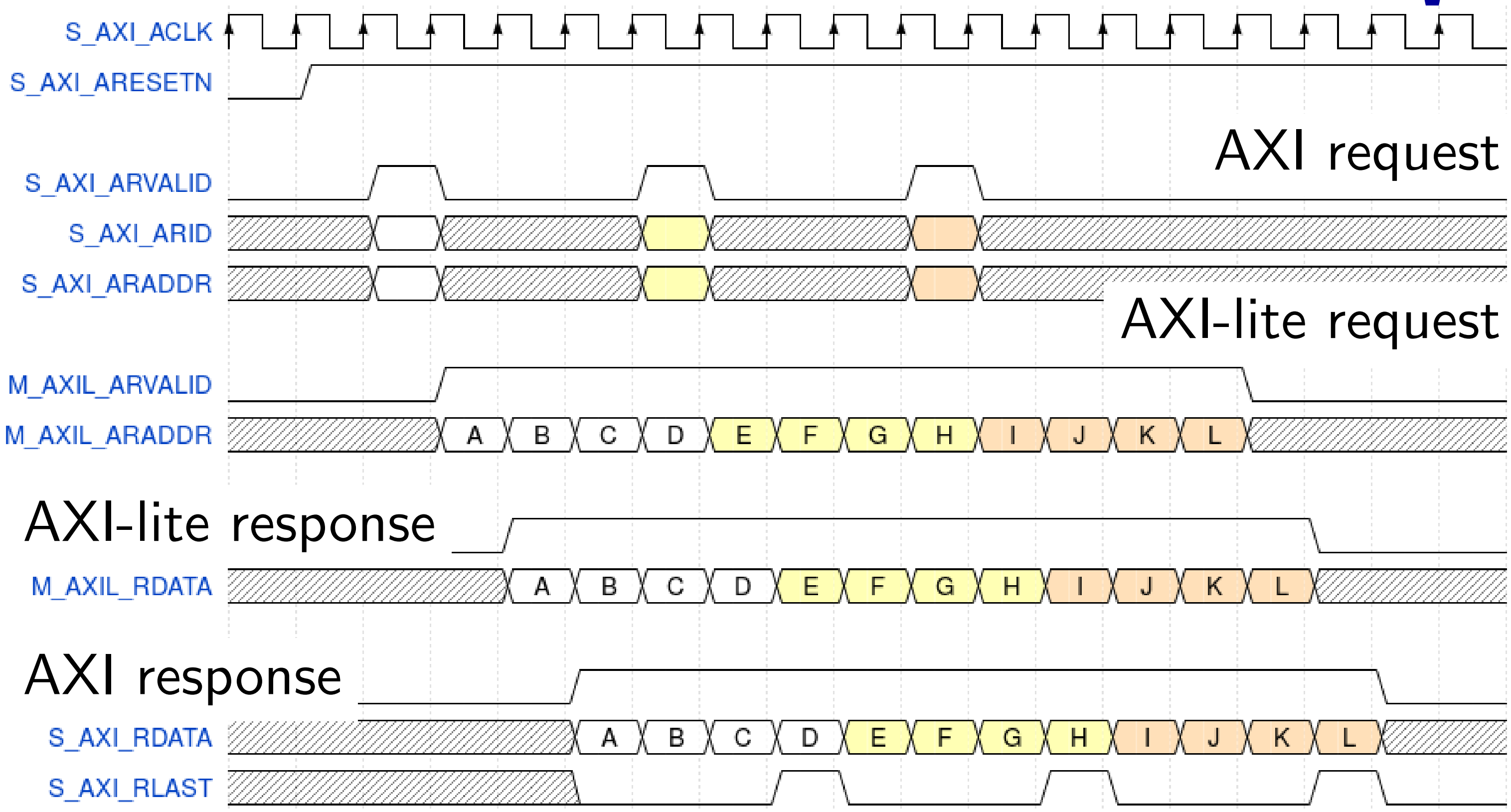
AXI-lite is powerful



That's how an AXI to AXI-lite bridge should work

- AXI protocol forces a minimum of one clock lost per bridge

AXI-lite is powerful



That's how an AXI to AXI-lite bridge should work

- No wasted clock cycles

The Problem

Why were so many cores broken?

- Most cores are tested via simulation and verification IP
- Simulation is dependent upon the creativity of the tester
  - Not checking what happens when xREADY low
  - Not checking what happens with back-to-back transactions
  - Not anticipating AWVALID & ARVALID at the same time
  - Only ever checking for one bus error type, never both
- The vendor verification materials are incomplete
- The vendor training examples have bugs!

Formal methods will check for bugs you aren't expecting

Conclusions

Everywhere
Challenge
Formal
Overview
Properties
Test Set
Bugs Found
Xilinx's AXI-lite demo
Personal Examples
AXI-lite is powerful
The Problem
▷ Conclusions
Backup Slides
How Long?
Initial
Full AXI Properties
Intel Bugs
Xilinx Bugs
Xilinx's AXI demo
Better AXI demo
Stories
Bus Fault Isolator

JE (Junior Engr): My AXI-lite IP isn't working. Could Xilinx's code be broken?
Me: Have you formally verified it?
JE: (Ignores me, switches to Xilinx'sforums) My IP isn't working. Your interconnect must be broken
Me: Have you formally verified your design?
JE: Here's the trace proving the problem
Me: In your trace, one bus request is producing two responses.

Formal Verification would've caught this

Backup Slides

How Long?

Everywhere
Challenge
Formal
Overview
Properties
Test Set
Bugs Found
Xilinx's AXI-lite demo
Personal Examples
AXI-lite is powerful
The Problem
Conclusions
Backup Slides
▷ How Long?
Initial
Full AXI Properties
Intel Bugs
Xilinx Bugs
Xilinx's AXI demo
Better AXI demo
Stories
Bus Fault Isolator

How long does it take to verify a design?

- □ Most of the work is really in the setup (1hr +)
- □ Here are some tool times

AXI-lite		
Wishbone to AXI-lite bridge	2	sec
AXI-lite to Wishbone bridge	13	sec
Basic AXI-lite slave	40	sec
Basic AXI-lite crossbar	2.1	mins
AXI		
Basic AXI (full) slave	10	sec
Wishbone to AXI bridge	40	sec
Bus Fault Isolator	4	min
Basic 4x8 AXI crossbar	25	min

- □ Finding a bug is faster than proving there are none

Initial Values

Everywhere
Challenge
Formal
Overview
Properties
Test Set
Bugs Found
Xilinx's AXI-lite demo
Personal Examples
AXI-lite is powerful
The Problem
Conclusions
Backup Slides
How Long?
▷ Initial
Full AXI Properties
Intel Bugs
Xilinx Bugs
Xilinx's AXI demo
Better AXI demo
Stories
Bus Fault Isolator

Initial value criteria is the least followed

- Some cores take many clocks to reset
  Xilinx's AXI implementations require 16 clocks of reset
- Few AXI/AXI-lite cores use initial values

```
always @(*)
if (!f_past_valid) // i.e. if first clock tick
begin
        assume(!S_AXI_ARVALID);
        // ...
        assert(!S_AXI_RVALID);
```

where

```
initial f_past_valid = 0;
always @(posedge S_AXI_ACLK)
        f_past_valid <= 1;
```

Full AXI Properties

68 full AXI slave properties, 51 master properties

- Mostly a superset of the AXI-lite properties
  - Counters for both bursts and beats
- `S_AXI_AxBURST` can only be `FIXED`, `INCR`, or `WRAP`
- `WRAP` bursts can only have `S_AXI_AxLEN` of 2, 4, 8, or 16
- `S_AXI_WSTRB` must be consistent with both `S_AXI_AWSIZE` and `S_AXI_AWADDR`
- `S_AXI_AxSIZE`: Must not be greater than the bus width
- Transactions may not cross 4kB boundaries
- A number of exclusive access checks (not yet tested)

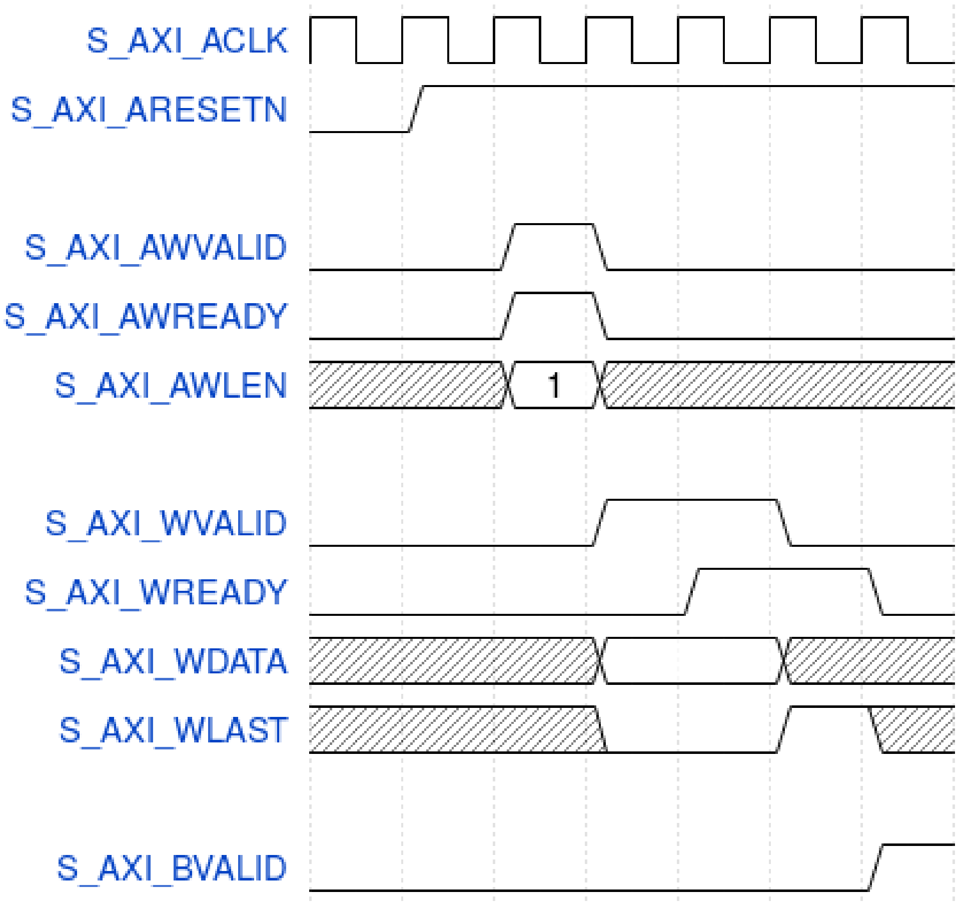
The following properties are really difficult (but checked anyway)

- No response without a prior request of the same ID
- xLAST must indicate the end of a read/write burst

Available as part of the Symbiotic EDA Suite

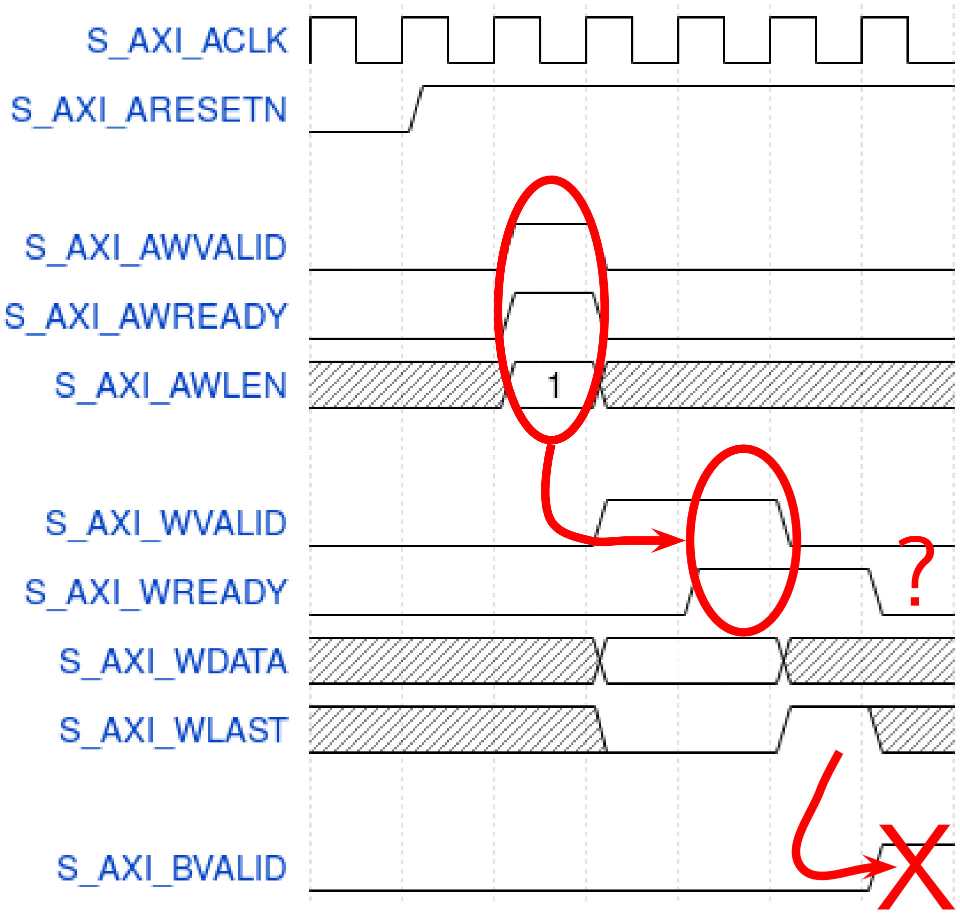
Intel Bugs

Intel's AXI3 demo



Intel Bugs

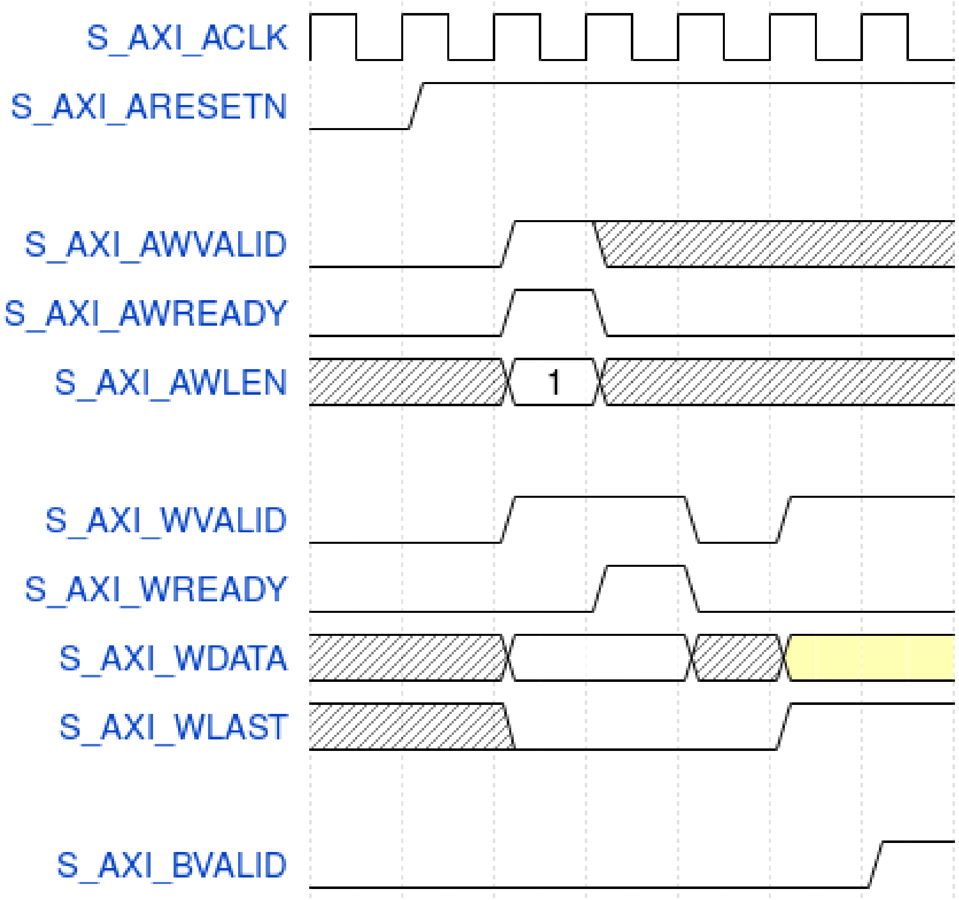
Intel's AXI3 demo



`WLAST` isn't ignored when `!WVALID` like it should be

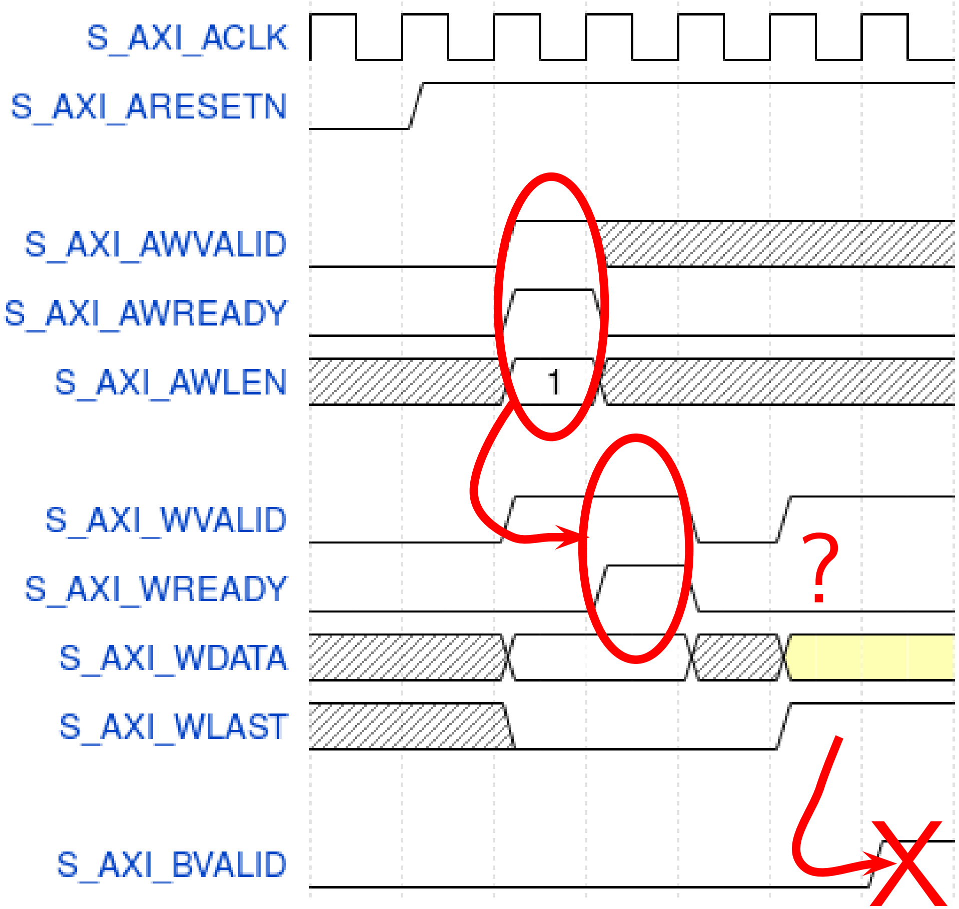
Intel Bugs

Intel's AXI3 demo



Intel Bugs

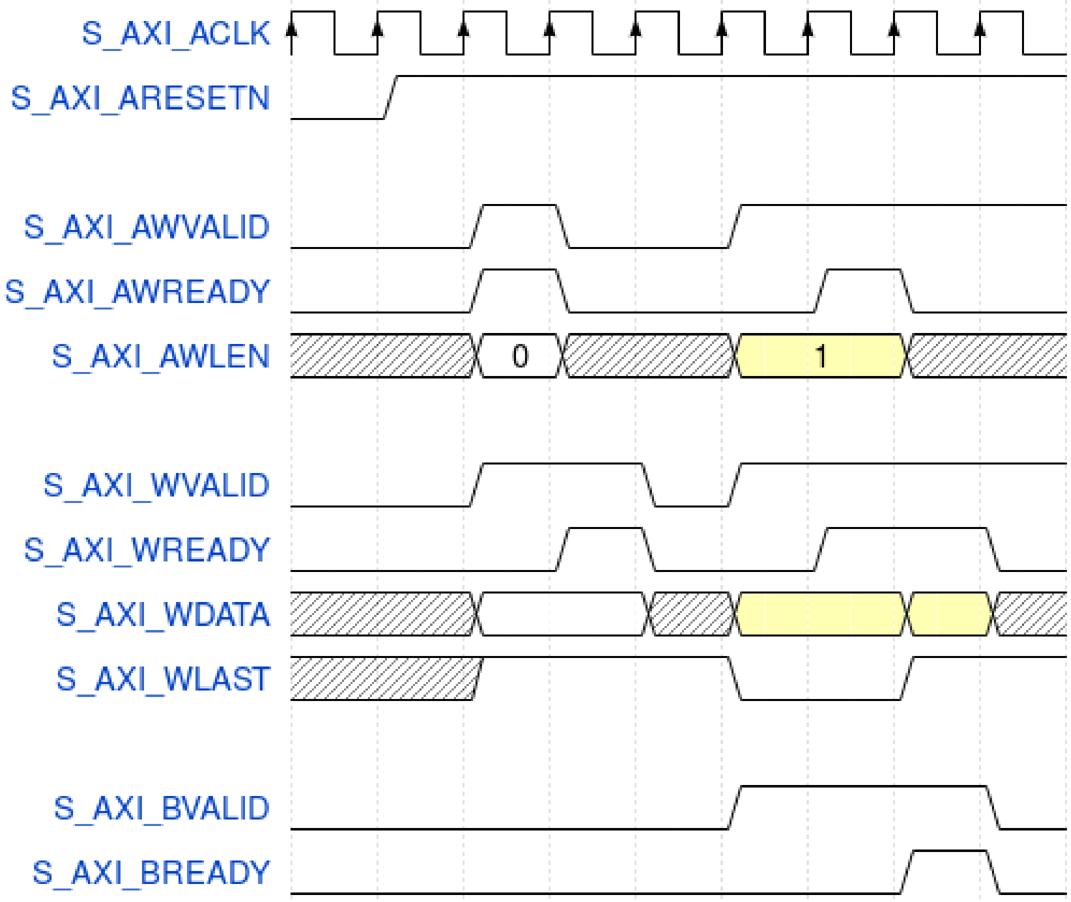
Intel's AXI3 demo



`WLAST` isn't ignored when `!WREADY`

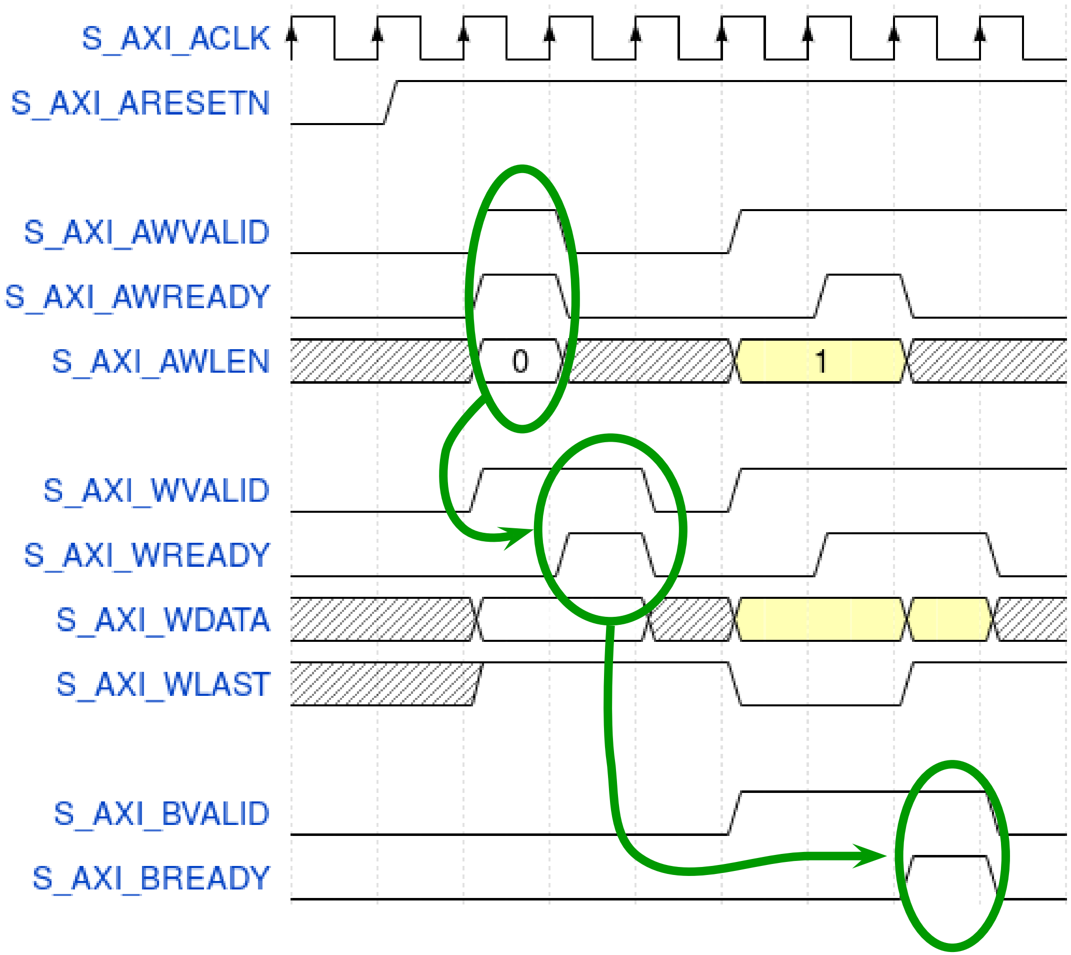
Bugs Found

Intel's AXI3 demo



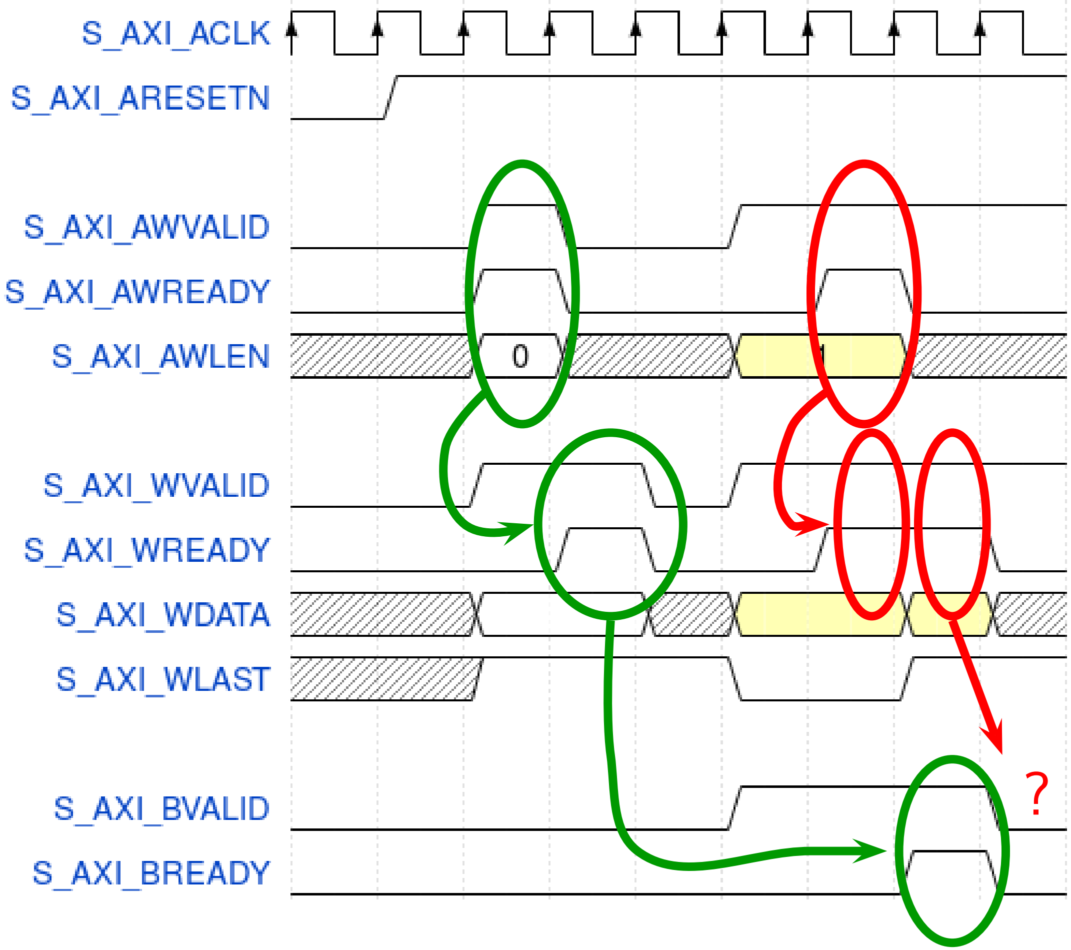
Bugs Found

Intel's AXI3 demo



Bugs Found

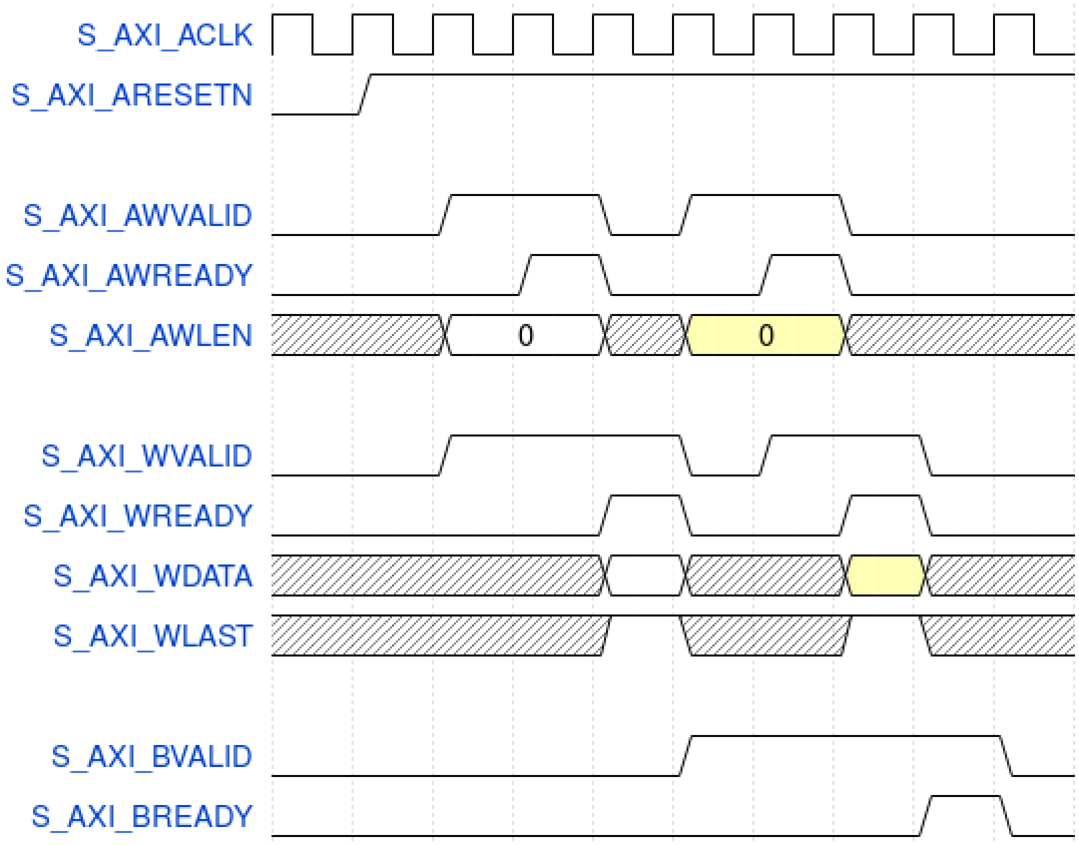
Intel's AXI3 demo



Any backpressure, and write returns gets dropped

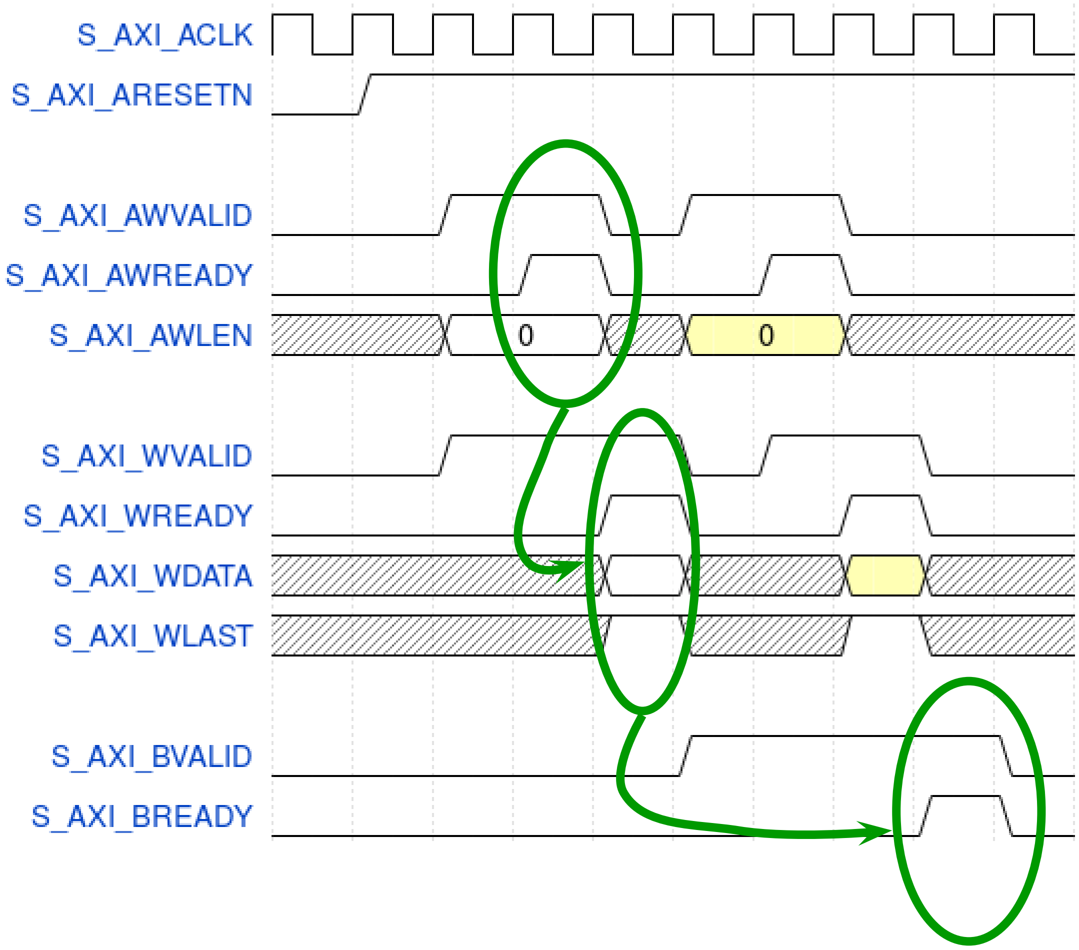
Xilinx Bugs

Vivado's 2019.1 AXI (full) demo



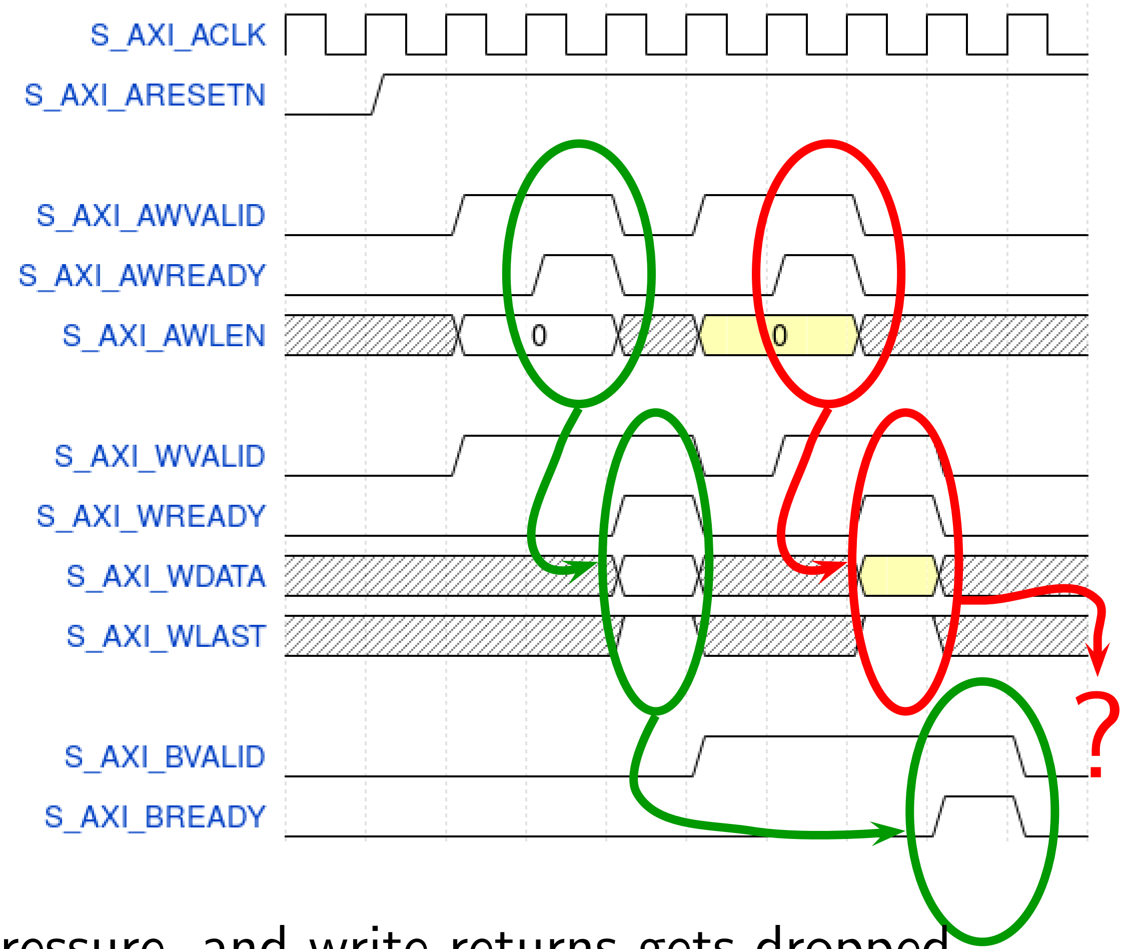
Xilinx Bugs

Vivado's 2019.1 AXI (full) demo



Xilinx Bugs

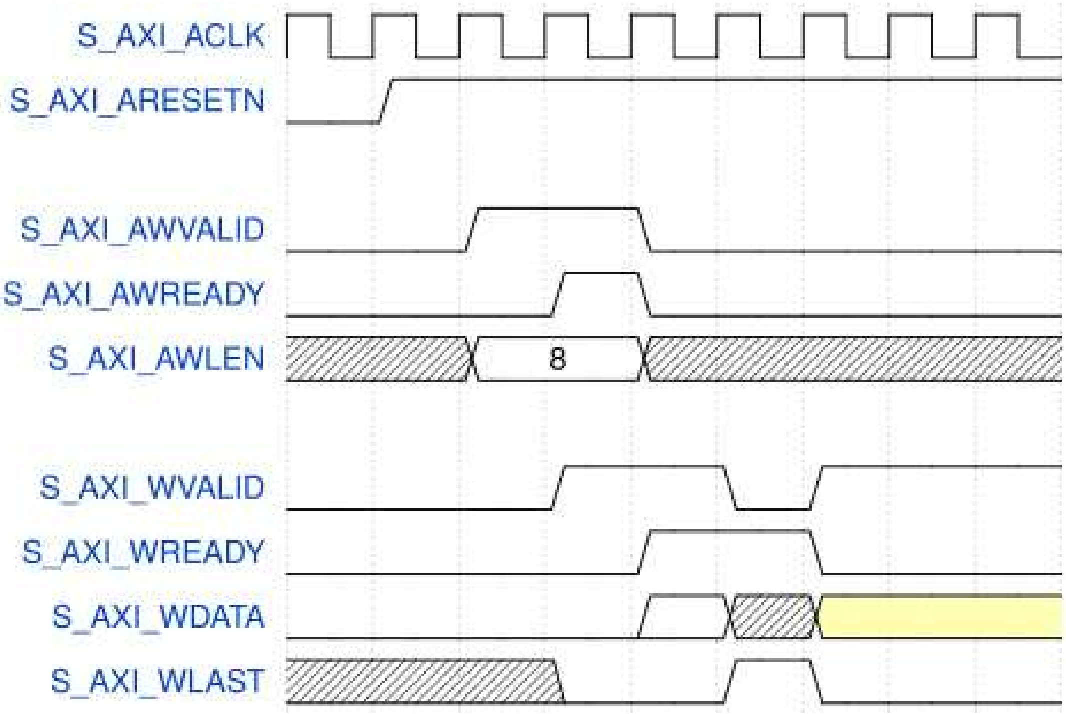
Vivado's 2019.1 AXI (full) demo



Any backpressure, and write returns gets dropped

Bugs Found

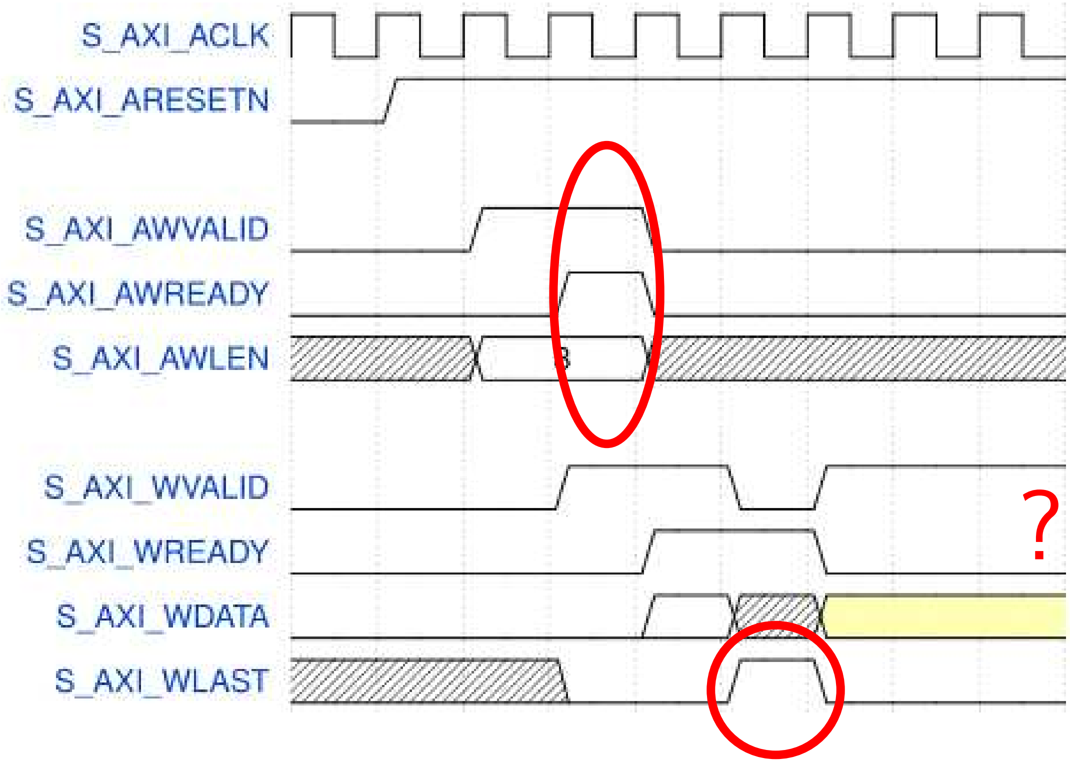
Vivado's 2019.1 AXI (full) demo



WLAST should be a don't care if WVALID is low.

Bugs Found

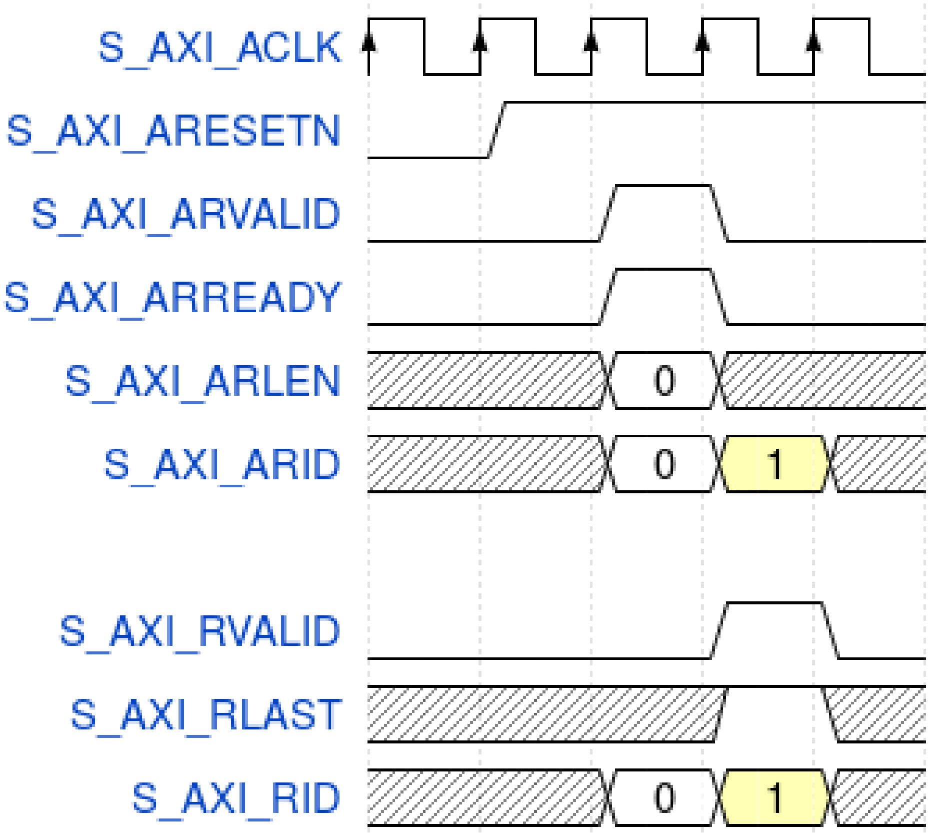
Vivado's 2019.1 AXI (full) demo



Xilinx's core stops accepting data if this ever happens

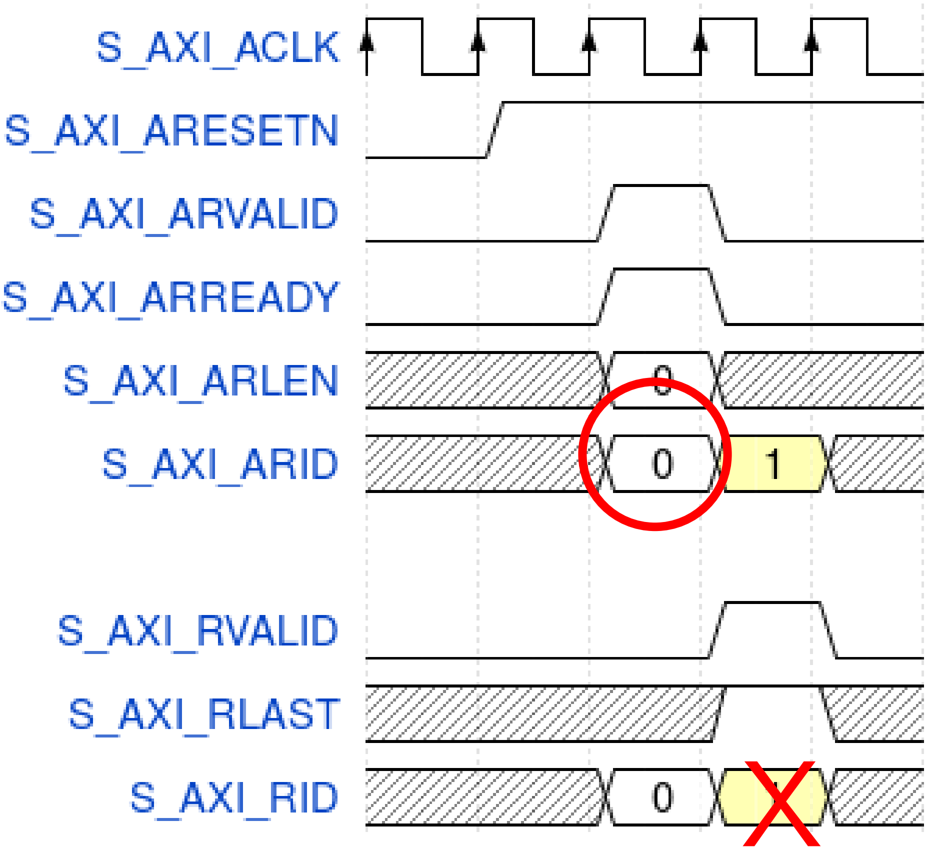
Bugs Found

Vivado's 2019.1 AXI (full) demo



Bugs Found

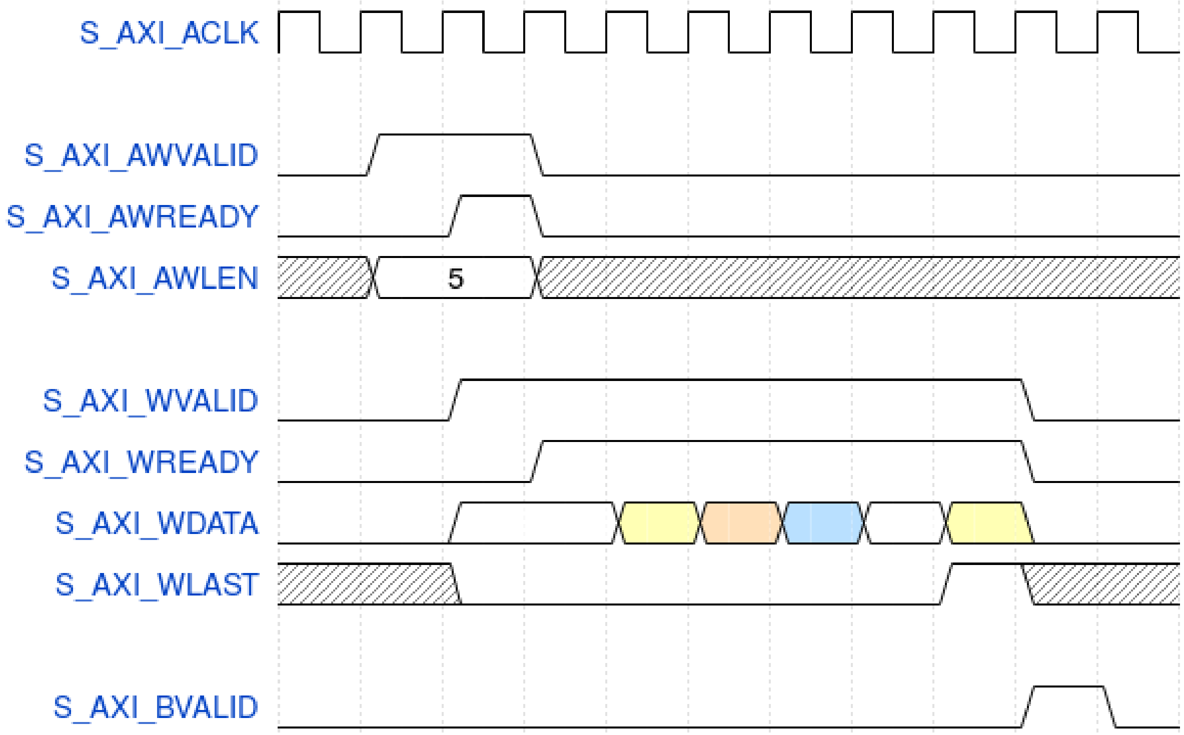
Vivado's 2019.1 AXI (full) demo



Return IDs (read or write) don't necessarily match request ID's

Xilinx's AXI demo

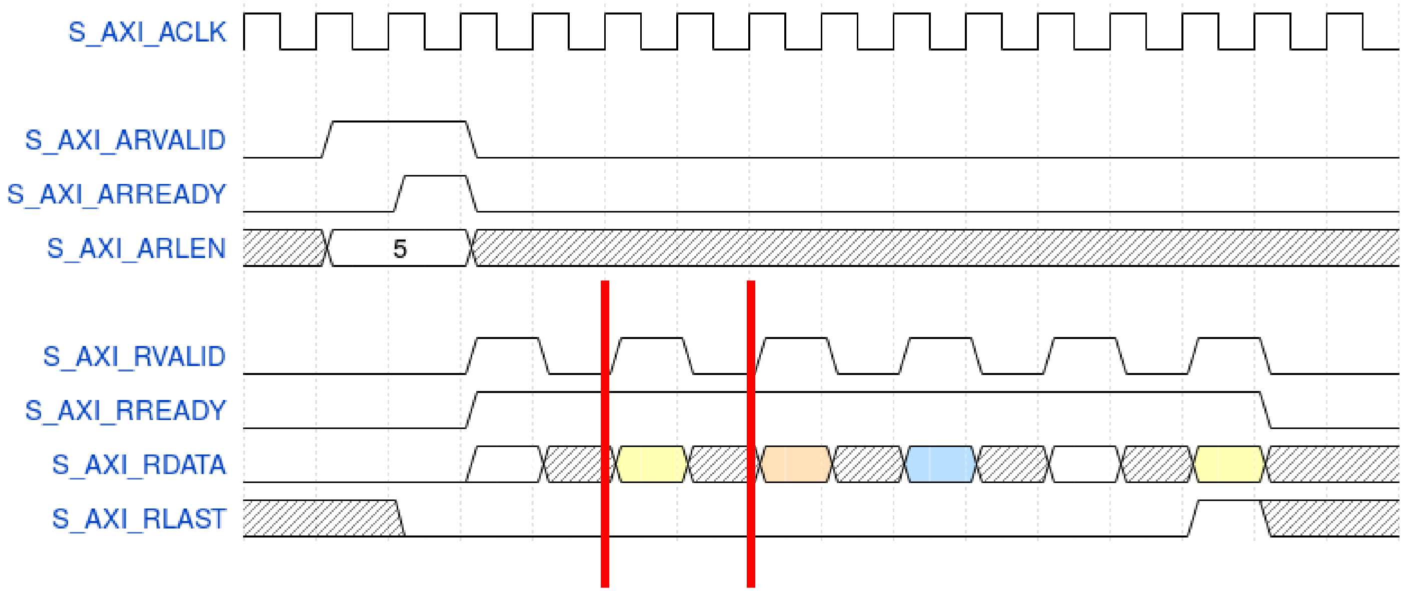
Vivado's 2019.1 AXI (full) demo burst write performance



Not bad, could drop latency by one

Xilinx's AXI demo

Vivado's 2019.1 AXI (full) demo burst read performance

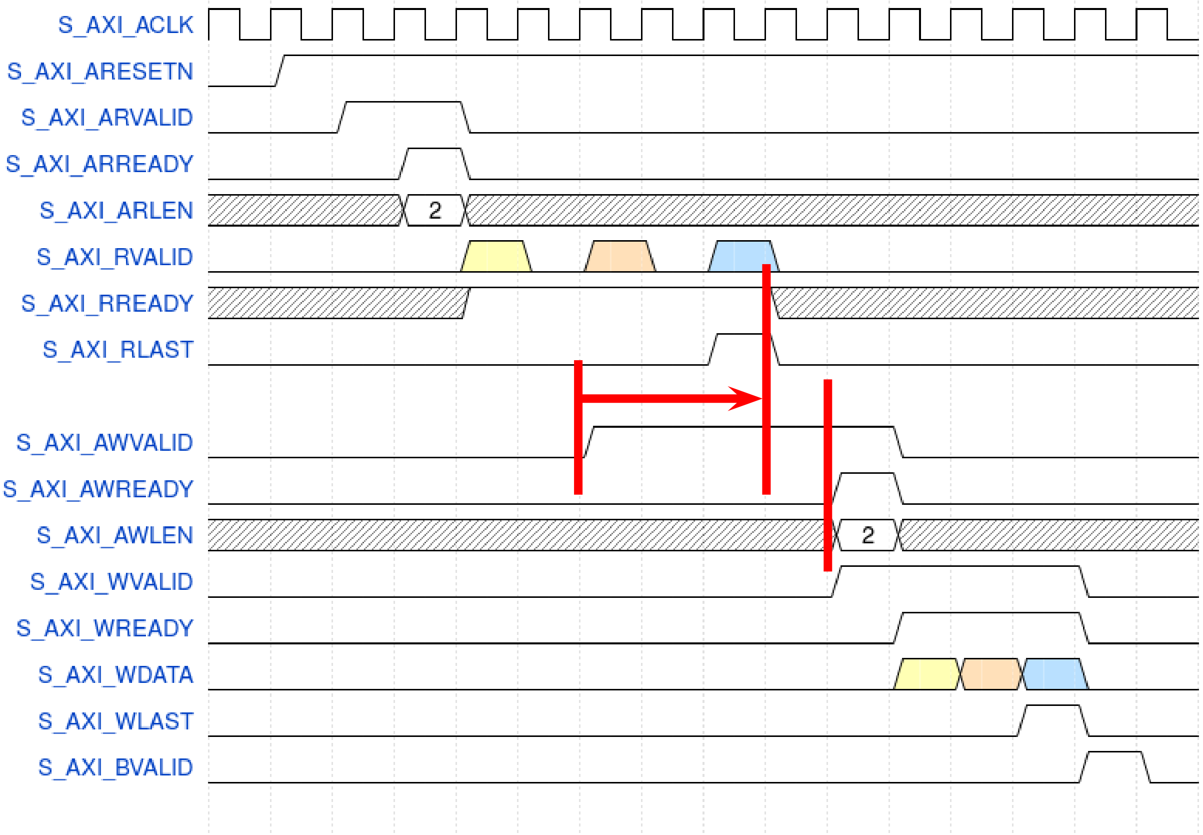


50% throughput??

Xilinx's AXI demo

Everywhere
Challenge
Formal
Overview
Properties
Test Set
Bugs Found
Xilinx's AXI-lite demo
Personal Examples
AXI-lite is powerful
The Problem
Conclusions
Backup Slides
How Long?
Initial
Full AXI Properties
Intel Bugs
Xilinx Bugs
▷ Xilinx's AXI demo
Better AXI demo
Stories
Bus Fault Isolator

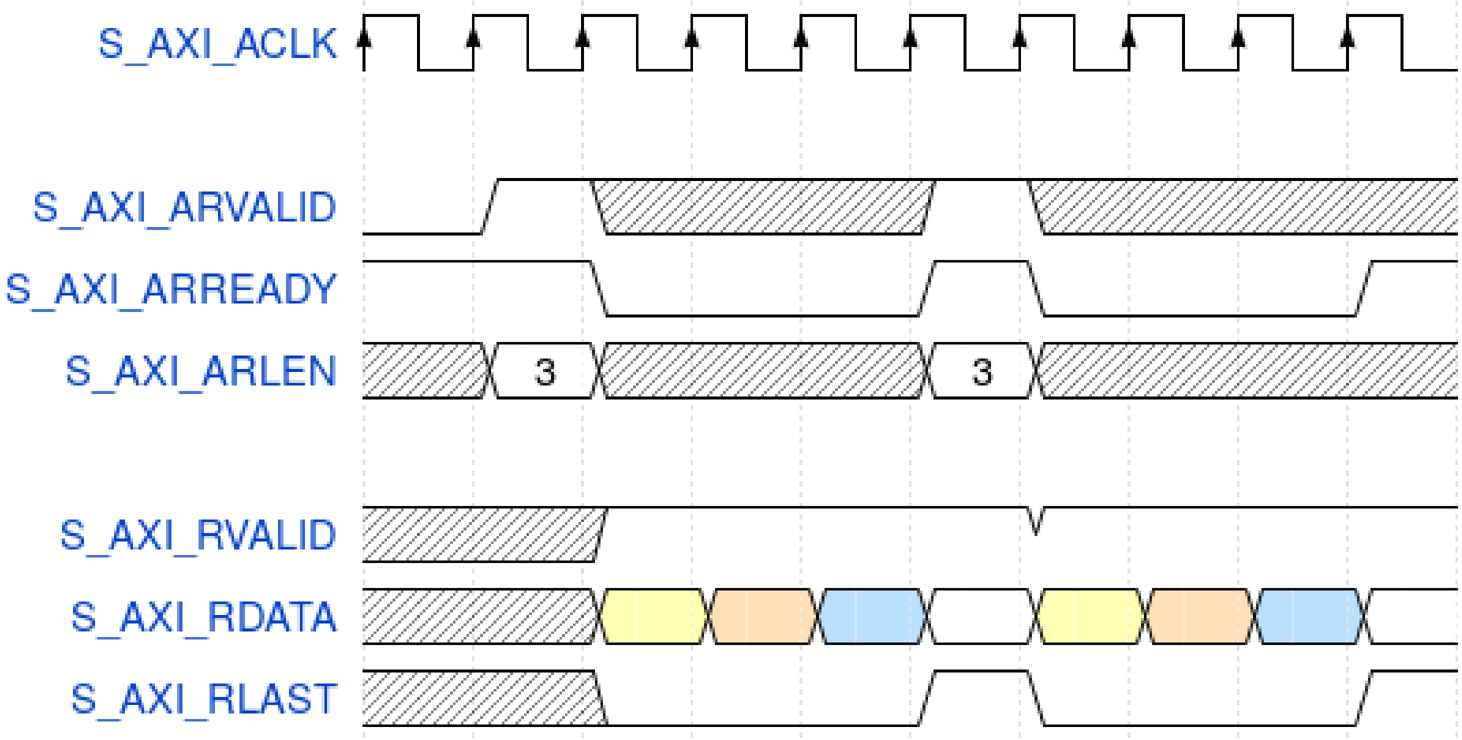
Vivado's 2019.1 AXI (full) demo burst write after read



Read transactions cause any pending write transactions to wait

Better AXI demo

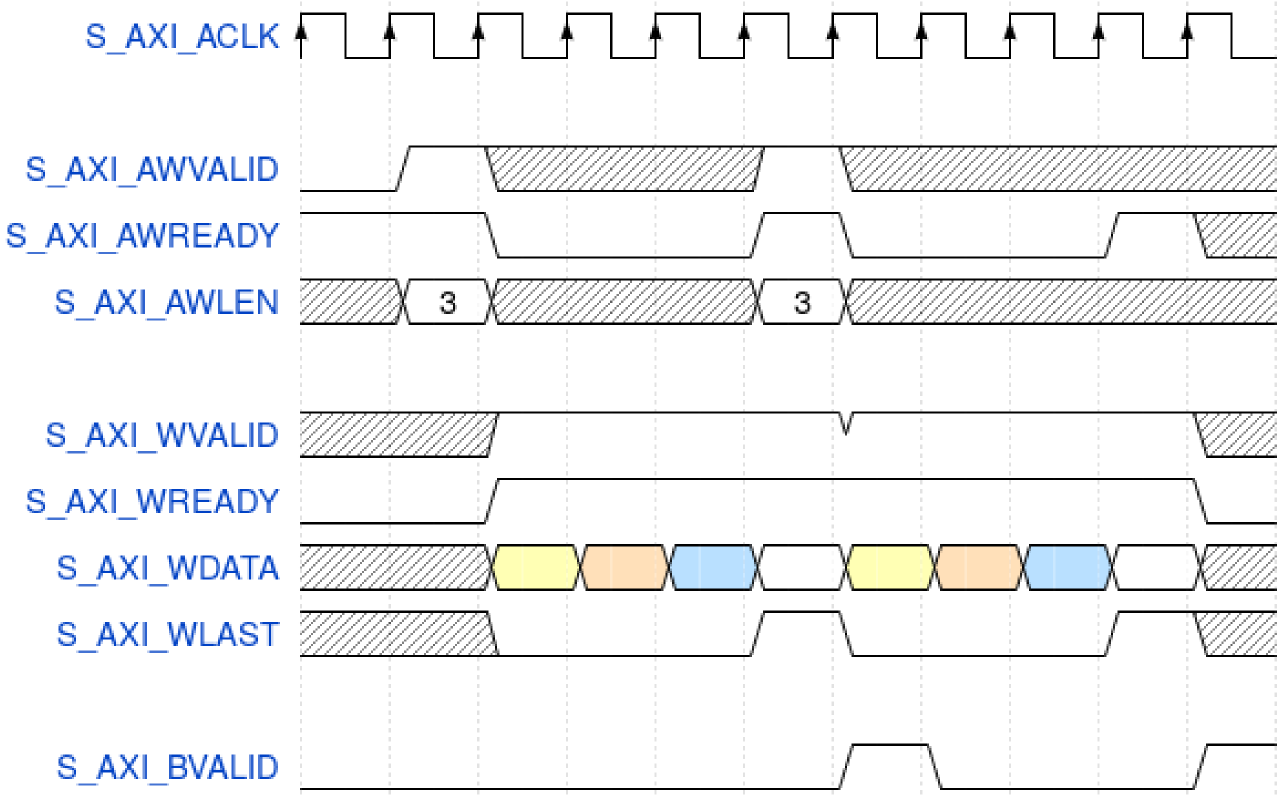
Better read performance



Link: Open Source AXI (full) demonstration design

Better AXI demo

Better write performance



Link: Open Source AXI (full) demonstration design

Easier to design

Spent over 2 months trying to build my first AXI slave

- Contract requirements made me move on several times
  - Failed to our promised performance in one case
- Once I had formal properties for AXI, design got easy
- I can now build a formally verified AXI core in a couple hours, start to finish
  - The DMA engines took a bit longer—about 20hrs
- These formally verified cores (typically) work the first time on real hardware

I can now design with confidence

MicroBlaze Story

FPGA Engineer left

- The design "worked" when he left

Software engineer then modified their MicroBlaze code

- Now issued two `strh` instructions in a row
- Design stopped working

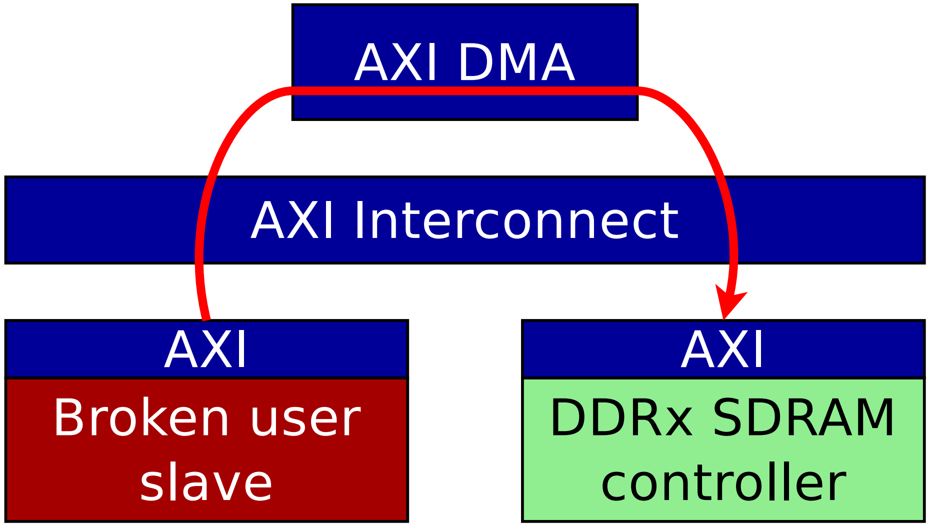
This is consistent with the bugs in Xilinx's demonstration designs

Link: (Until Xilinx deletes it)

AXI DMA

Customer wants to reset the DMA

- Copying data to/from his (AXI) core and SDRAM
- DMA freezes. How should it be reset?



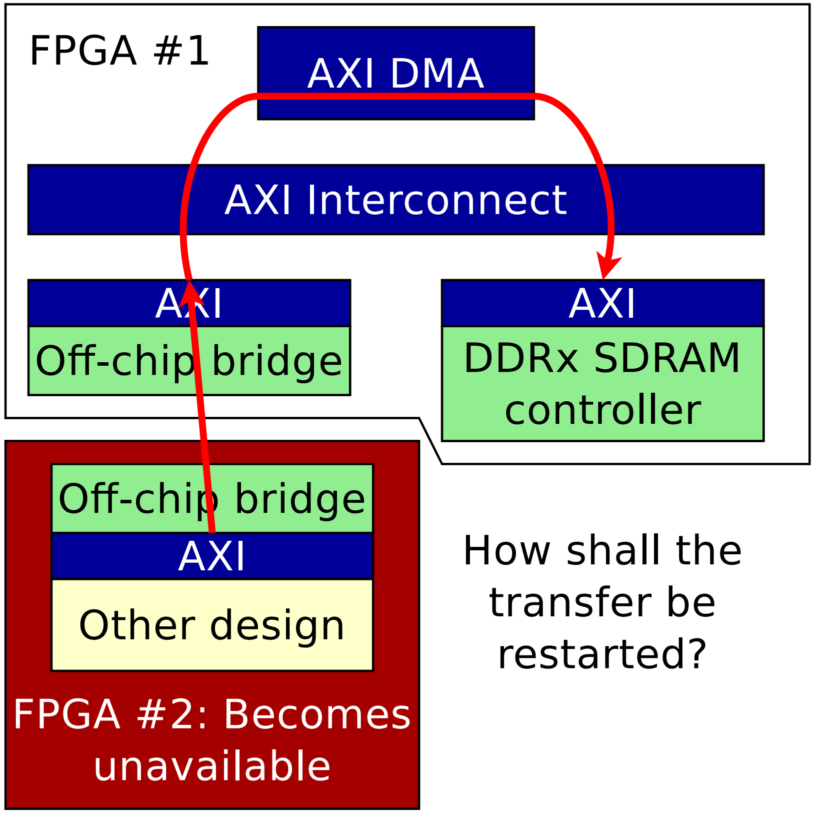
This is consistent with the bugs in Xilinx's designs

- AXI has no means of aborting a transaction
- If a transaction gets dropped, the design is dead until a reset

AXI DMA

Second customer wants to reset the DMA

- □ Copying data to/from an AXI-to-external bridge
- □ External device becomes unavailable, so the DMA freezes



How shall the transfer be restarted?

AXI DMA

Second customer wants to reset the DMA

- Copying data to/from an AXI-to-external bridge
- External device becomes unavailable, so the DMA freezes
- How should the transfer be restarted?

Xilinx's recommendation? Reset the DMA

- This would break any interconnect or other intermediate core that depends upon an outstanding transaction counter

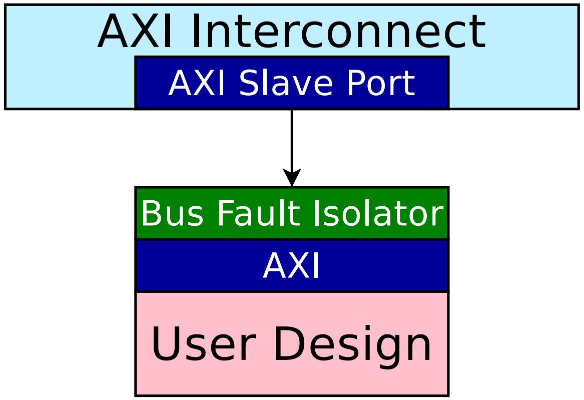
Bus Fault Isolator

Problem: One AXI mistake will hang a design

- AXI is complicated
- Neither Intel, nor Xilinx with their AXI VIP, could get it right

Solution: The bus fault isolator

- Sits between AXI infrastructure and slave design

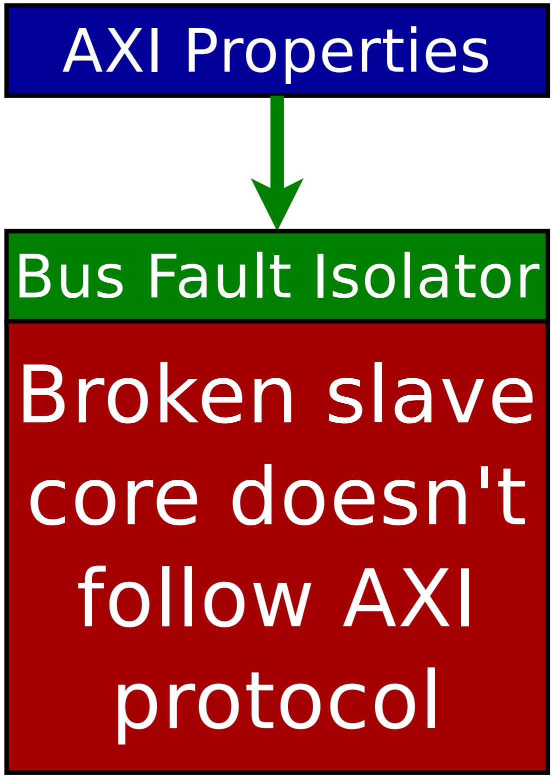


- Guarantees the slave design behaves
- Returns a valid AXI response (`SLVERR`) on any fault
- Sets flags to (potentially) trigger an internal logic analyzer
- Can be configured to reset the slave on error

Bus Fault Isolator

Two proofs:

1. Prove upstream will always be valid
   - Independent of whatever the slave might do

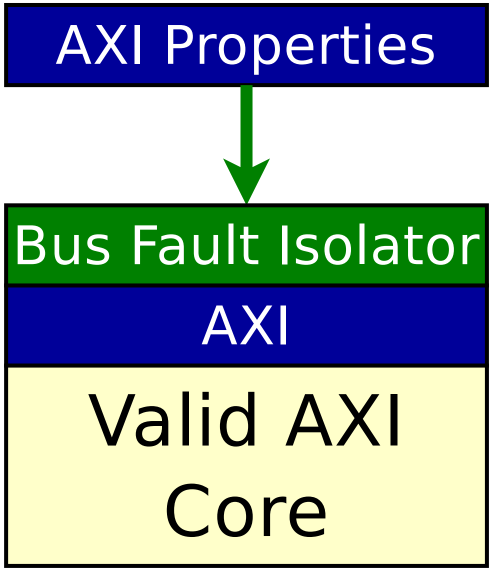


Prove that AXI properties are followed in spite of a (potentially) broken slave

Bus Fault Isolator

Two proofs:

1. Prove upstream will always be valid
   - □ Independent of whatever the slave might do
2. Assume downstream is valid
   - □ Prove no fault will be detected



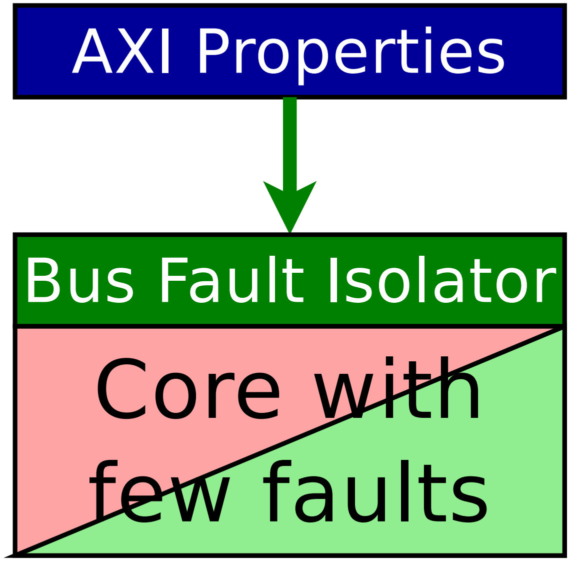
Prove that no faults are ever triggered as long as the slave core follows the AXI protocol

Bus Fault Isolator

Two proofs:

1. Prove upstream will always be valid
2. Prove valid AXI components never fault

Optional feature: Reset downstream core on any fault



Issue a reset upon a fault, repeat two proofs above

Link: axisafety.v

Unknown bug

Xilinx customer complains of a bug he cannot find

- □ Using their demo core
- □ Switches to one of my example cores–his bug vanishes

Open Source — it can work