AXI4-Compliant Memory-Mapped Slave

Rana ayman hamdy Mahmoud

Contents

Introduction

This project implements a simplified AXI4-based system that enables communication between an AXI4 master

and an internal memory module. The goal is to model, simulate, and verify the behavior of AXI4 transactions

such as burst reads and writes, address decoding, and proper data handshaking.

Test Memory in AXI Slave with small testbench

Bugs found

1)

```
Transcript
VSIM 11> run -all
# ERROR  mem[   0] = 00000000 while intended wr_data = 12153524
# ERROR  mem[   1] = 00000000 while intended wr_data = c0895e81
# ERROR  mem[   2] = 00000000 while intended wr_data = 8484d609
# ERROR  mem[   3] = 00000000 while intended wr_data = b1f05663
# ERROR  mem[   4] = 00000000 while intended wr_data = 06b97b0d
# ERROR  mem[   5] = 00000000 while intended wr_data = 46df998d
# ERROR  mem[   6] = 00000000 while intended wr_data = b2c28465
# ERROR  mem[   7] = 00000000 while intended wr_data = 89375212
# ERROR  mem[   8] = 00000000 while intended wr_data = 00f3e301
# ERROR  mem[   9] = 00000000 while intended wr_data = 06d7cd0d
# ERROR  mem[  10] = 00000000 while intended wr_data = 3b23f176
# ERROR  mem[  11] = 00000000 while intended wr_data = 1e8dcd3d
# ERROR  mem[  12] = 00000000 while intended wr_data = 76d457ed
# ERROR  mem[  13] = 00000000 while intended wr_data = 462df78c
# ERROR  mem[  14] = 00000000 while intended wr_data = 7cfde9f9
# ERROR  mem[  15] = 00000000 while intended wr_data = e33724c6
# ERROR  mem[  16] = 00000000 while intended wr_data = e2f784c5
# ERROR  mem[  17] = 00000000 while intended wr_data = d513d2aa
# ERROR  mem[  18] = 00000000 while intended wr_data = 72aff7e5
# ERROR  mem[  19] = 00000000 while intended wr_data = bbd27277
# ERROR  mem[  20] = 00000000 while intended wr_data = 8932d612
# ERROR  mem[  21] = 00000000 while intended wr_data = 47ecdb8f
# ERROR  mem[  22] = 00000000 while intended wr_data = 793069f2
# ERROR  mem[  23] = 00000000 while intended wr_data = e77696ce
# ERROR  mem[  24] = 00000000 while intended wr_data = f4007ae8
# ERROR  mem[  25] = 00000000 while intended wr_data = e2ca4ec5
# ERROR  mem[  26] = 00000000 while intended wr_data = 2e58495c
# ERROR  mem[  27] = 00000000 while intended wr_data = de8e28bd
# ERROR  mem[  28] = 00000000 while intended wr_data = 96ab582d
# ERROR  mem[  29] = 00000000 while intended wr_data = b2a72665
# ERROR  mem[  30] = 00000000 while intended wr_data = b1ef6263
# ERROR  mem[  31] = 00000000 while intended wr_data = 0573870a
# ERROR  mem[  32] = 00000000 while intended wr_data = c03b2280
# ERROR  mem[  33] = 00000000 while intended wr_data = 10642120
# ERROR  mem[  34] = 00000000 while intended wr_data = 557845aa
# ERROR  mem[  35] = 00000000 while intended wr_data = cecccc9d
# ERROR  mem[  36] = 00000000 while intended wr_data = cb203e96
Ln:  78  Col:  0   Project : mem   Now: 20,490 ns  Delta: 1
```

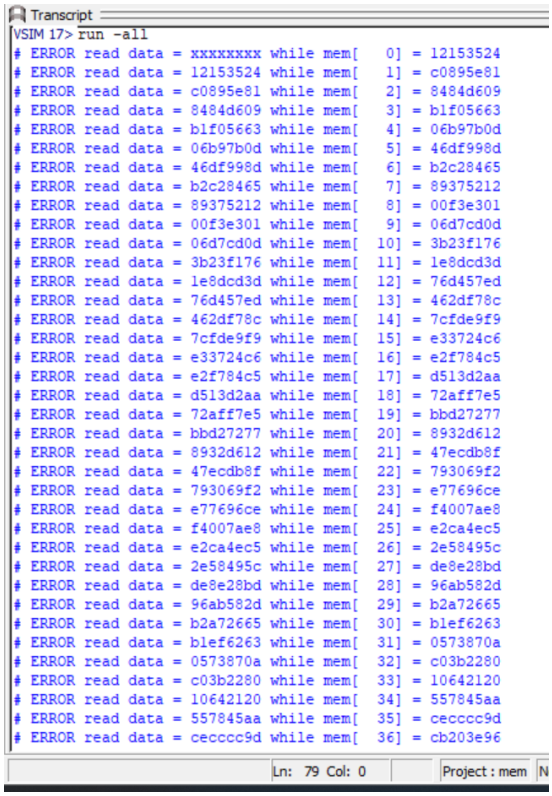
This is because that reset in design active high while in spec active low

The bug and its correction

```
// Memory write
always @(posedge clk) begin
    if (rst_n)
        mem_rdata <= 0;
    else if (mem_en) begin
        if (mem_we)
            memory[mem_addr] <= mem_wdata;
        else
            mem_rdata <= memory[mem_addr-1];
    end
end
```

```
// Memory write
always @(posedge clk) begin
    if (!rst_n)
        mem_rdata <= 0;
    else if (mem_en) begin
        if (mem_we)
            memory[mem_addr] <= mem_wdata;
        else
            mem_rdata <= memory[mem_addr-1];
    end
end
```

2)



This because design read from memory in location address -1

Bug and its correction

```
    else if (mem_en) begin
        if (mem_we)
            memory[mem_addr] <= mem_wdata;
        else
          mem_rdata <= memory[mem_addr-1];
    end
```

```
    else if (mem_en) begin
        if (mem_we)
            memory[mem_addr] <= mem_wdata;
        else
          mem_rdata <= memory[mem_addr];
    end
```

AXI4 Wrapper Verification

Test Plan

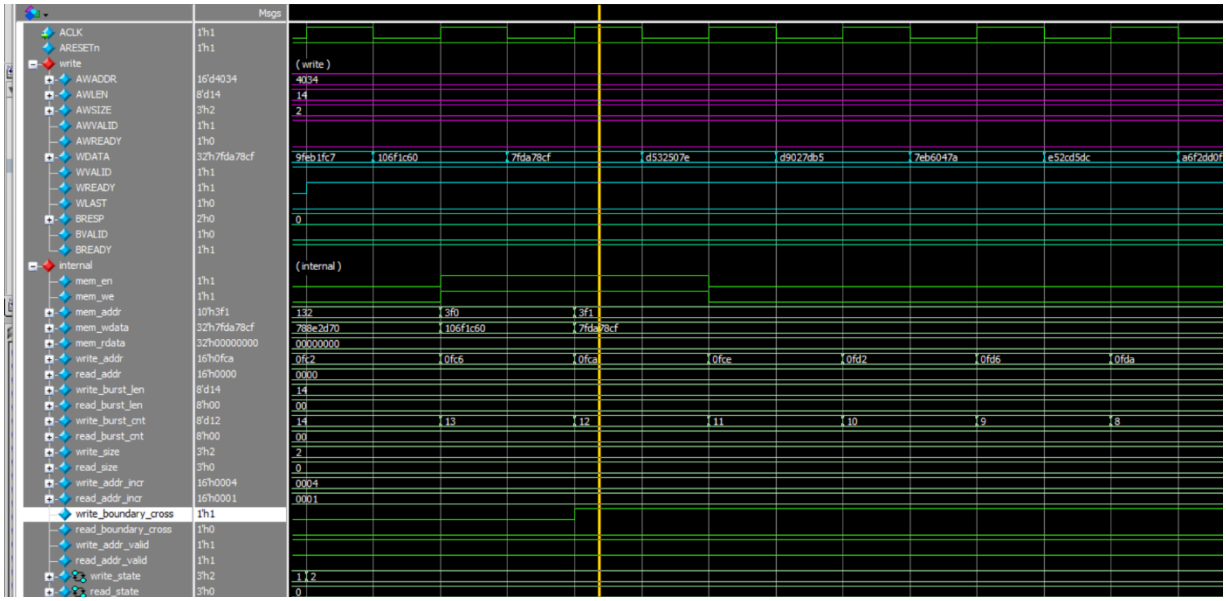
Feature	Check List	Stimulus
Clk checker	- check clk toggle with the wanted period by making clk_temp to save past value through half period	
Rst checker	- At -ve edge ARESETn and when ARESETn = 0 >> AWREADY, WREADY ,BRESP ,BVALID ,ARREADY , RDATA ,RRESP	- ARESETn = 0 - ARESETn = 1

	,RVALID and RLAST become zero - At ARESETn = 1 system function as expected	
Write address to slave	- Valid asserted until ready come from slave to accept address (AWREADY = 1)	- AWVALID = 1
slave accept write data	- Valid asserted until ready come from slave to accept data (WREADY = 1)	- WVALID = 1
Response check	- If WLAST asserted then !WREADY ,BVALID and BRESP = OKAY or SLVERR	- WLAST = 1
Boundary check	- AxADDR >> 2 + AxLEN + 1 < MEMORY_DEPTH - Negative check by violate boundary with address > MEMORY_DEPTH	- AxADDR + AxLEN + 1 < MEMORY_DEPTH * 4 - AxADDR + AxLEN + 1 > MEMORY_DEPTH * 4 - AxSIZE = 2
Read address from slave	- Valid asserted until ready come from slave to accept address (ARREADY = 1)	- ARVALID = 1
Read data from slave (internal memory)	- When master is ready slave assert RVALID and put data on RDATA also send RRESP and on reading last beat assert RLAST	- RREADY = 1

Bugs found

1)

```
# ================================================================
# burst write of length 15 from byte address fc2 (word address 3f0)
# Write response success
#  check that if data written successfully in memory
# time =            143480000 Write success: Expected 106f1c60, Got 106f1c60 addr 3f0
# time =            143480000 Write success: Expected 7fda78cf, Got 7fda78cf addr 3f1
#  time =            143480000 Write error: Expected d532507e, Got d68ab4c8 addr 3f2
#  time =            143480000 Write error: Expected d9027db5, Got c61f434d addr 3f3
#  time =            143480000 Write error: Expected 7eb6047a, Got fa2fce77 addr 3f4
#  time =            143480000 Write error: Expected e52cd5dc, Got 5097119d addr 3f5
#  time =            143480000 Write error: Expected a6f2dd0f, Got 67d4b9dc addr 3f6
#  time =            143480000 Write error: Expected 2a041810, Got 0 addr 3f7
#  time =            143480000 Write error: Expected 9a980b12, Got 0 addr 3f8
#  time =            143480000 Write error: Expected a69efa4b, Got 0 addr 3f9
#  time =            143480000 Write error: Expected 49d343bb, Got c27a2295 addr 3fa
#  time =            143480000 Write error: Expected afc46bd7, Got f9f1e32c addr 3fb
#  time =            143480000 Write error: Expected 977785e9, Got e6d5d362 addr 3fc
#  time =            143480000 Write error: Expected ffe1f68, Got 0 addr 3fd
#  time =            143480000 Write error: Expected c3230b66, Got 0 addr 3fe
# ================================================================
```



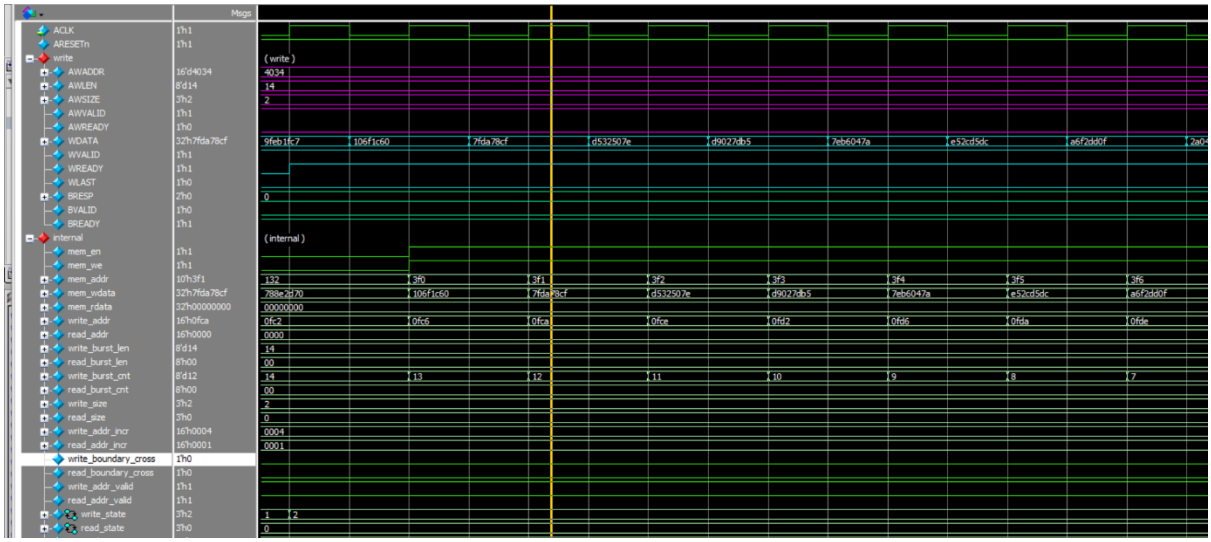

```
    // Address boundary check (4KB boundary = 12 bits)
    assign write_boundary_cross = ((write_addr & 12'hFFF) + (write_burst_len << write_size)) > 12'hFFF;
```

The design asserts write_boundary_cross and prevents the write operation to memory, even though no actual boundary crossing occurs.

Based on the calculation (4034 × 4 × 15 = 4094), the address range remains within the allowed boundary, so the write_boundary_cross signal should not be asserted in this scenario.

Its correction

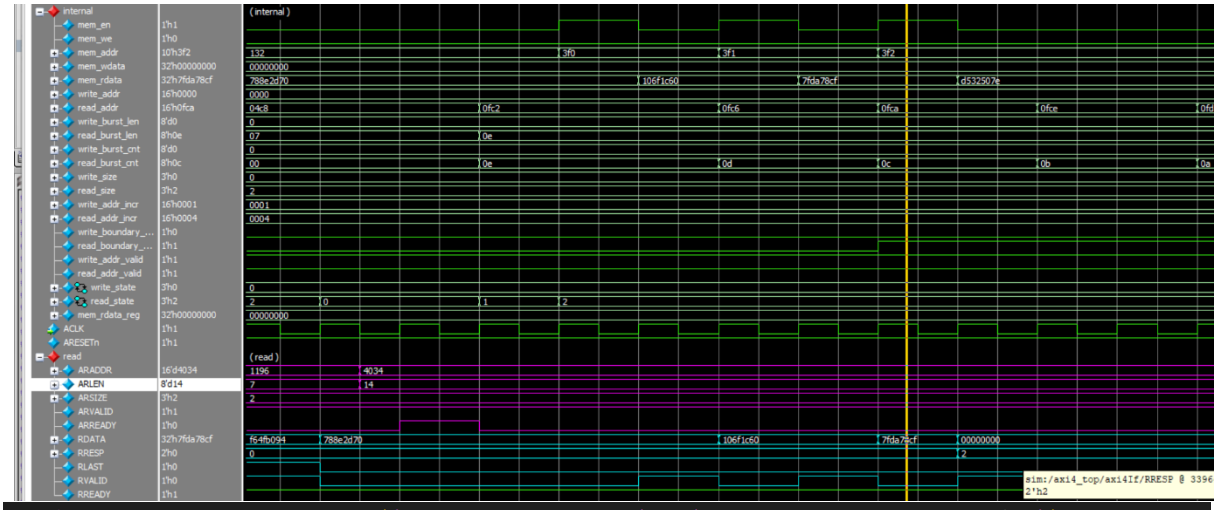
```
// Address boundary check (4KB boundary = 12 bits)
//assign write_boundary_cross = ((write_addr & 12'hFFF) + (write_burst_len << write_size)) > 12'hFFF;
assign write_boundary_cross = ((write_addr & 12'hFFF) + ((write_burst_cnt + 1) << write_size)) > 12'hFFF;
```




```
# ================================================================================
# burst write of length 15 from byte address fc2 (word address 3f0)
# Write response success
#  check that if data written successfully in memory
# time =            143480000 Write success: Expected 106f1c60, Got 106f1c60 addr 3f0
# time =            143480000 Write success: Expected 7fda78cf, Got 7fda78cf addr 3f1
# time =            143480000 Write success: Expected d532507e, Got d532507e addr 3f2
# time =            143480000 Write success: Expected d9027db5, Got d9027db5 addr 3f3
# time =            143480000 Write success: Expected 7eb6047a, Got 7eb6047a addr 3f4
# time =            143480000 Write success: Expected e52cd5dc, Got e52cd5dc addr 3f5
# time =            143480000 Write success: Expected a6f2dd0f, Got a6f2dd0f addr 3f6
# time =            143480000 Write success: Expected 2a041810, Got 2a041810 addr 3f7
# time =            143480000 Write success: Expected 9a980b12, Got 9a980b12 addr 3f8
# time =            143480000 Write success: Expected a69efa4b, Got a69efa4b addr 3f9
# time =            143480000 Write success: Expected 49d343bb, Got 49d343bb addr 3fa
# time =            143480000 Write success: Expected afc46bd7, Got afc46bd7 addr 3fb
# time =            143480000 Write success: Expected 977785e9, Got 977785e9 addr 3fc
# time =            143480000 Write success: Expected ffe1f68, Got ffe1f68 addr 3fd
# time =            143480000 Write success: Expected c3230b66, Got c3230b66 addr 3fe
# ================================================================================
```

2)

```
# ================================================================================
# burst read of rlength 15 from byte raddress fc2 (word raddress 3f0)
#  time =            339606000 Read success: Expected 106f1c60, Got 106f1c60 raddr 3f0
#  time =            339626000 Read success: Expected 7fda78cf, Got 7fda78cf raddr 3f1
#  time =            339646000 Read error: Expected d532507e, Got 0 raddr 3f2
#  time =            339666000 Read error: Expected d9027db5, Got 0 raddr 3f3
#  time =            339686000 Read error: Expected 7eb6047a, Got 0 raddr 3f4
#  time =            339706000 Read error: Expected e52cd5dc, Got 0 raddr 3f5
#  time =            339726000 Read error: Expected a6f2dd0f, Got 0 raddr 3f6
#  time =            339746000 Read error: Expected 2a041810, Got 0 raddr 3f7
#  time =            339766000 Read error: Expected 9a980b12, Got 0 raddr 3f8
#  time =            339786000 Read error: Expected a69efa4b, Got 0 raddr 3f9
#  time =            339806000 Read error: Expected 49d343bb, Got 0 raddr 3fa
#  time =            339826000 Read error: Expected afc46bd7, Got 0 raddr 3fb
#  time =            339846000 Read error: Expected 977785e9, Got 0 raddr 3fc
#  time =            339866000 Read error: Expected ffe1f68, Got 0 raddr 3fd
#  time =            339886000 Read error: Expected c3230b66, Got 0 raddr 3fe
# ================================================================================
```



```
assign read_boundary_cross = ((read_addr & 12'hFFF) + (read_burst_len << read_size)) > 12'hFFF;
```

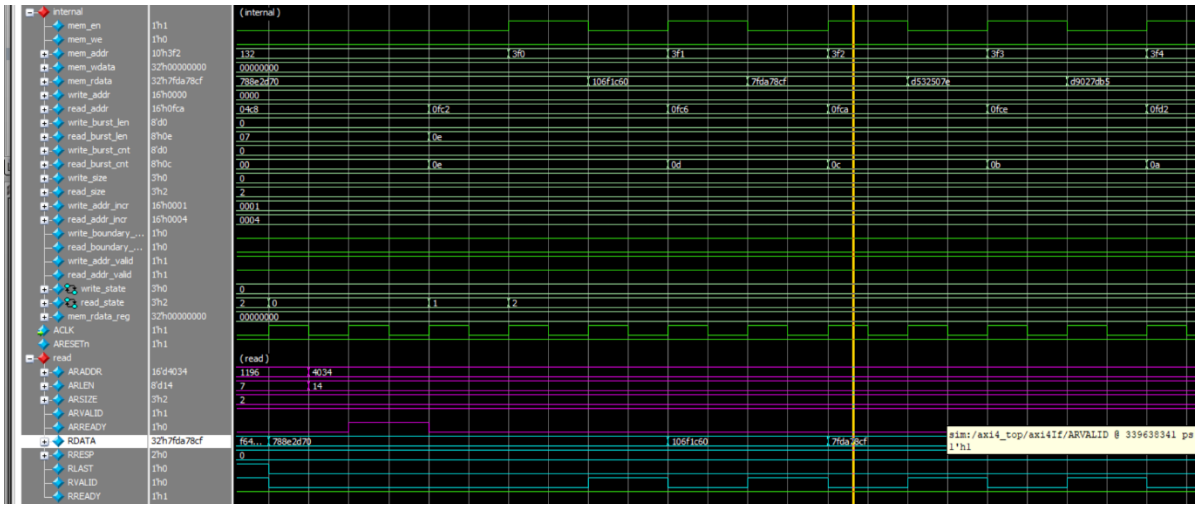
The design asserts read_boundary_cross and prevents the memory read operation, even though no boundary crossing actually occurs.

Based on the calculation (4034 × 4 × 15 = 4094), the address range does not exceed the boundary limit, so the read_boundary_cross signal should not be asserted in this case.

Its correction

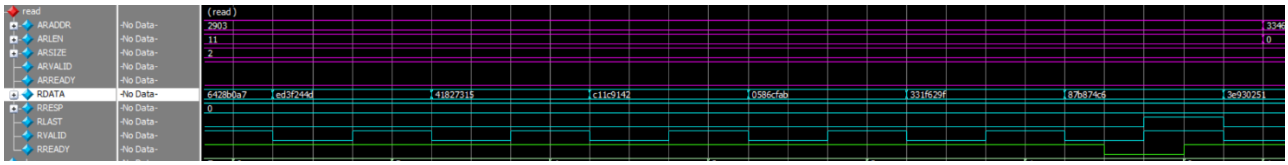
```
//assign read_boundary_cross = ((read_addr & 12'hFFF) + (read_burst_len << read_size)) > 12'hFFF;
assign read_boundary_cross = ((read_addr & 12'hFFF) + ((read_burst_cnt + 1) << read_size)) > 12'hFFF;
```

```
# ================================================================
# burst read of rlength 15 from byte raddress fc2 (word raddress 3f0)
#  time =              339606000 Read success: Expected 106f1c60, Got 106f1c60 raddr 3f0
#  time =              339626000 Read success: Expected 7fda78cf, Got 7fda78cf raddr 3f1
#  time =              339646000 Read success: Expected d532507e, Got d532507e raddr 3f2
#  time =              339666000 Read success: Expected d9027db5, Got d9027db5 raddr 3f3
#  time =              339686000 Read success: Expected 7eb6047a, Got 7eb6047a raddr 3f4
#  time =              339706000 Read success: Expected e52cd5dc, Got e52cd5dc raddr 3f5
#  time =              339726000 Read success: Expected a6f2dd0f, Got a6f2dd0f raddr 3f6
#  time =              339746000 Read success: Expected 2a041810, Got 2a041810 raddr 3f7
#  time =              339766000 Read success: Expected 9a980b12, Got 9a980b12 raddr 3f8
#  time =              339786000 Read success: Expected a69efa4b, Got a69efa4b raddr 3f9
#  time =              339806000 Read success: Expected 49d343bb, Got 49d343bb raddr 3fa
#  time =              339826000 Read success: Expected afc46bd7, Got afc46bd7 raddr 3fb
#  time =              339846000 Read success: Expected 977785e9, Got 977785e9 raddr 3fc
#  time =              339866000 Read success: Expected ffe1f68, Got ffe1f68 raddr 3fd
#  time =              339886000 Read success: Expected c3230b66, Got c3230b66 raddr 3fe
```



3)

```
# ================================================================
# burst read of rlength 12 from byte raddress b57 (word raddress 2d5)
#   time =            171545000 Read error: Expected f406ab10, Got 829e5889 raddr 2d5
#   time =            171565000 Read error: Expected 342dcba, Got f406ab10 raddr 2d6
#   time =            171585000 Read error: Expected 72aee25a, Got 342dcba raddr 2d7
#   time =            171605000 Read error: Expected 17b5f753, Got 72aee25a raddr 2d8
#   time =            171625000 Read error: Expected 6428b0a7, Got 17b5f753 raddr 2d9
#   time =            171645000 Read error: Expected ed3f244d, Got 6428b0a7 raddr 2da
#   time =            171665000 Read error: Expected 41827315, Got ed3f244d raddr 2db
#   time =            171685000 Read error: Expected c11c9142, Got 41827315 raddr 2dc
#   time =            171705000 Read error: Expected 586cfab, Got c11c9142 raddr 2dd
#   time =            171725000 Read error: Expected 331f629f, Got 586cfab raddr 2de
#   time =            171745000 Read error: Expected 87b874c6, Got 331f629f raddr 2df
#   time =            171765000 Read error: Expected 3e930251, Got 87b874c6 raddr 2e0
# ----------------------------------------------------------------
```



According to the AXI specification, read data must be transferred in the same cycle when RVALID is asserted and RREADY is high. Additionally, RLAST must be asserted during the cycle in which the final data beat of the burst is transferred.

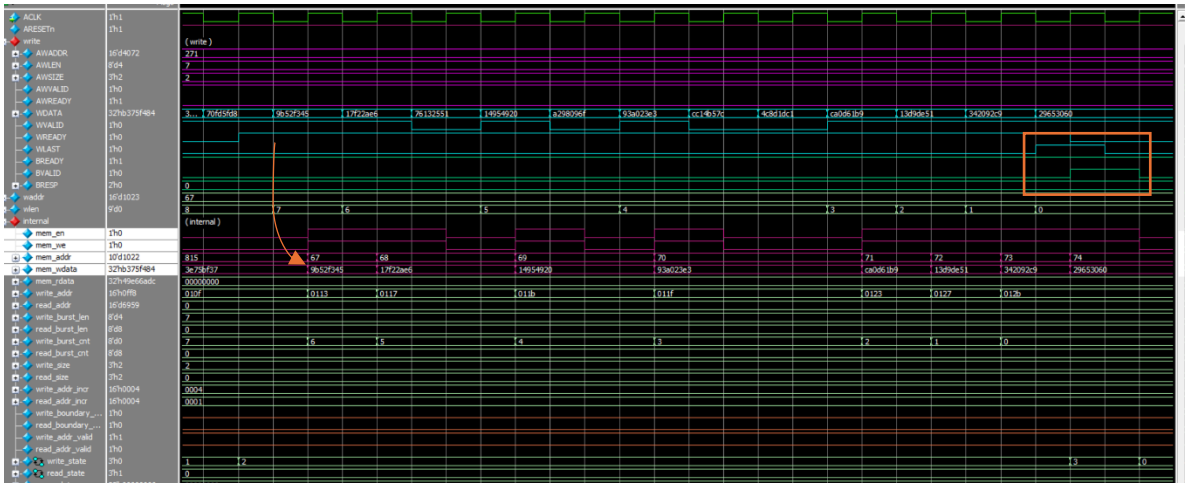
However, in the current implementation, the design outputs the read data one clock cycle later than expected after RVALID is asserted.

Please update the checker temporarily so that it tolerates this one-cycle latency and does not report a failure until the design team corrects the behavior.

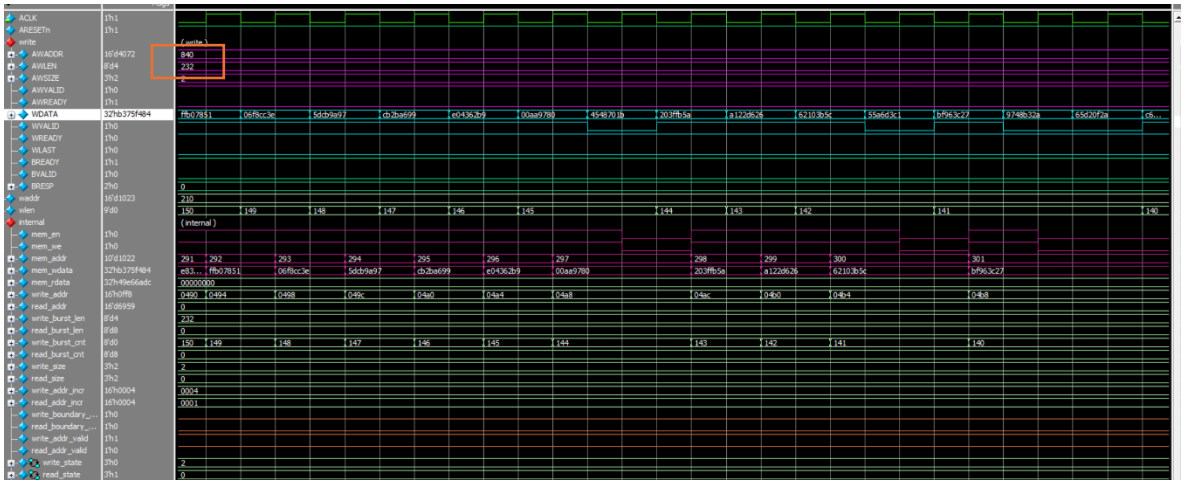
```
# ================================================================
# burst read of rlength 12 from byte raddress b57 (word raddress 2d5)
#   time =            171405000 Read success: Expected f406ab10, Got f406ab10 raddr 2d5
#   time =            171425000 Read success: Expected 342dcba, Got 342dcba raddr 2d6
#   time =            171445000 Read success: Expected 72aee25a, Got 72aee25a raddr 2d7
#   time =            171465000 Read success: Expected 17b5f753, Got 17b5f753 raddr 2d8
#   time =            171485000 Read success: Expected 6428b0a7, Got 6428b0a7 raddr 2d9
#   time =            171505000 Read success: Expected ed3f244d, Got ed3f244d raddr 2da
#   time =            171525000 Read success: Expected 41827315, Got 41827315 raddr 2db
#   time =            171545000 Read success: Expected c11c9142, Got c11c9142 raddr 2dc
#   time =            171565000 Read success: Expected 586cfab, Got 586cfab raddr 2dd
#   time =            171585000 Read success: Expected 331f629f, Got 331f629f raddr 2de
#   time =            171605000 Read success: Expected 87b874c6, Got 87b874c6 raddr 2df
#   time =            171625000 Read success: Expected 3e930251, Got 3e930251 raddr 2e0
# ----------------------------------------------------------------
```

Waveform

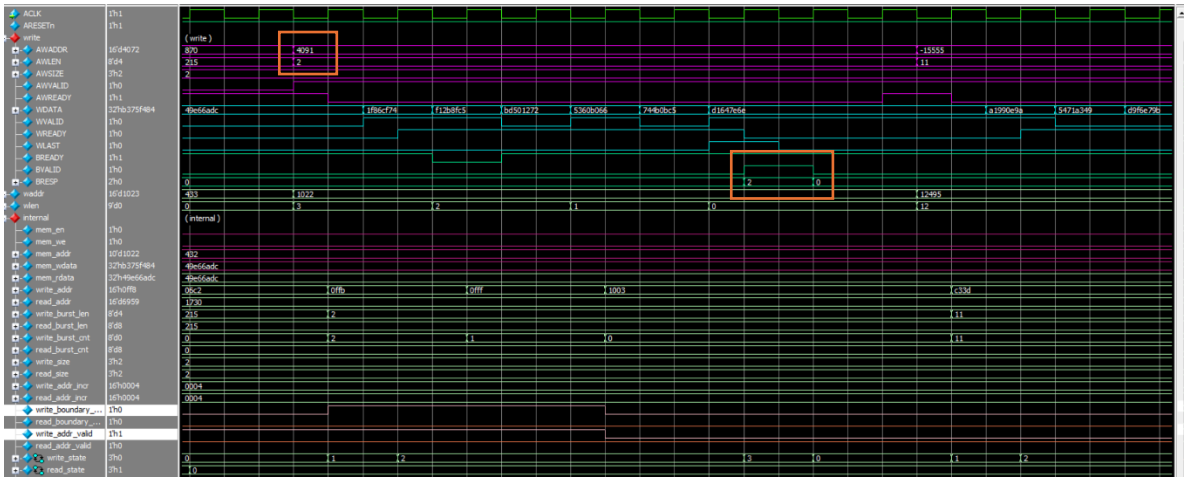
Write



As WVALID & WREADY is high data written in memory and at the end WLAST asserted and response OKAY (0)

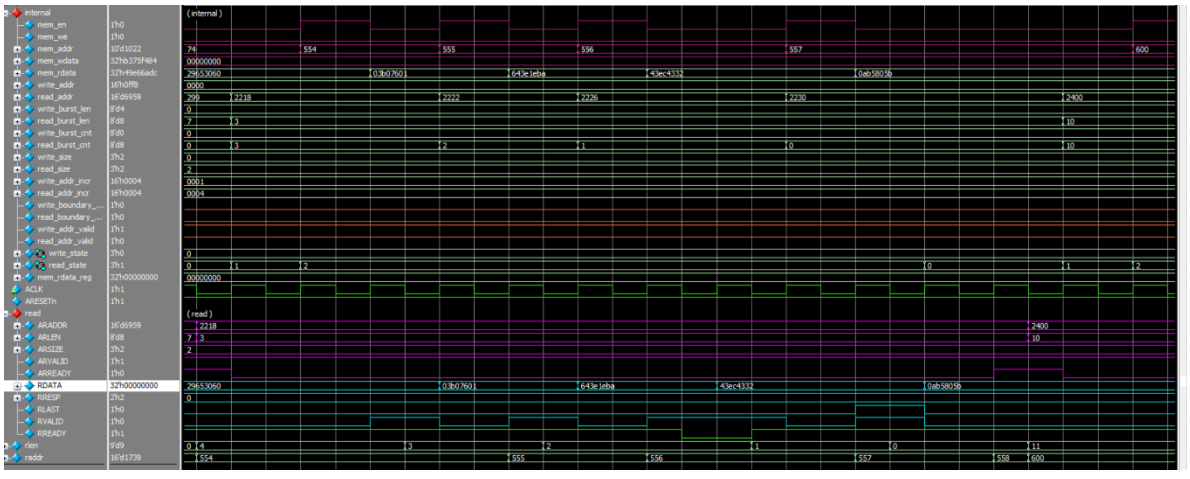


Long burst

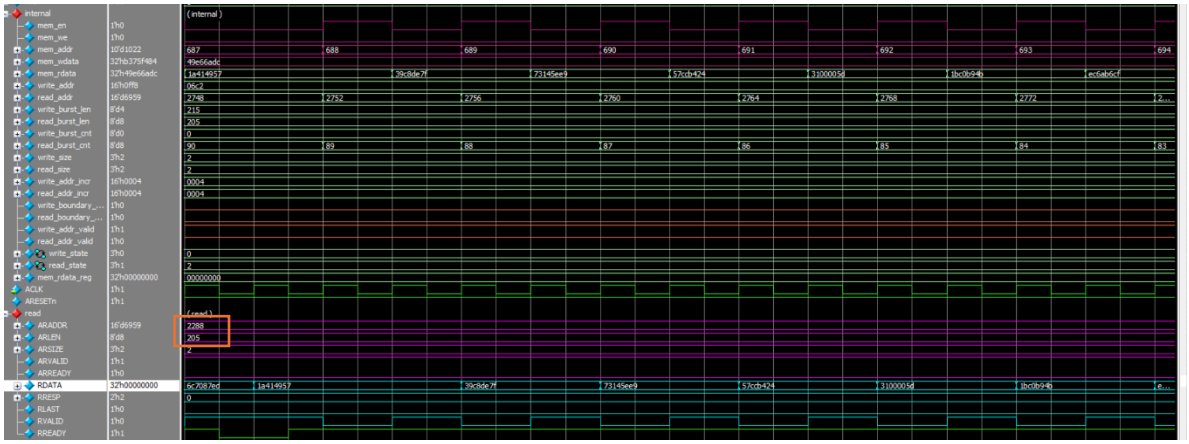


Write burst out of bound so didn’t written in memory and response = 2 (SLVERR)

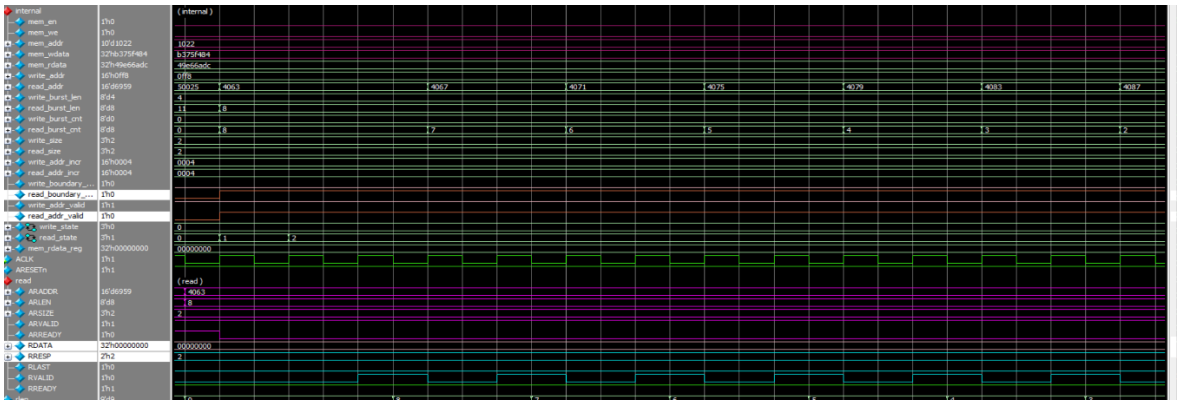
Read



When RVALID & RREADY are high we read from memory and at the end of burst RLAST is high



Long burst



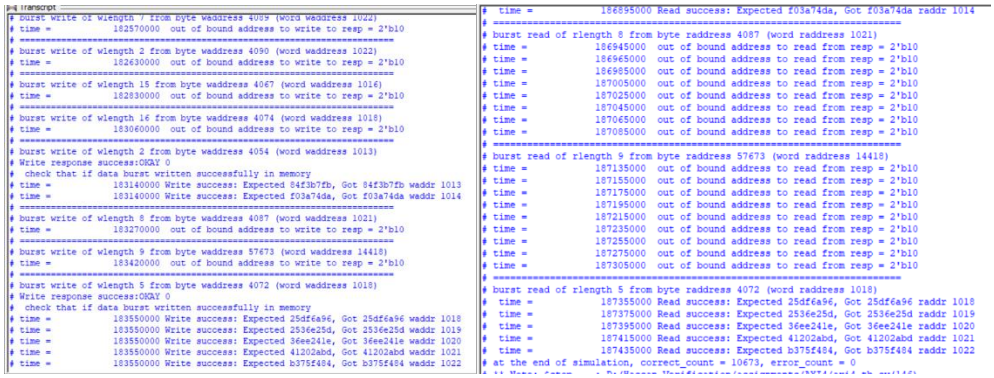
Read burst out of bound so RDATA = 0 and response = 2 (SLVERR)

Log

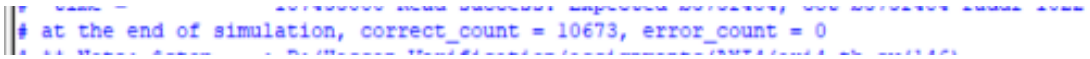
```
# burst write of wlength 216 from byte waddress 870 (word waddress 217)
# Write response success:OKAY 0
#  check that if data burst written successfully in memory
# time =              91780000 Write success: Expected 7a6dcd6, Got 7a6dcd6 waddr 217
# time =              91780000 Write success: Expected 42af01da, Got 42af01da waddr 218
# time =              91780000 Write success: Expected dcecace, Got dcecace waddr 219
# time =              91780000 Write success: Expected b367737e, Got b367737e waddr 220
# time =              91780000 Write success: Expected b7f8c479, Got b7f8c479 waddr 221
# time =              91780000 Write success: Expected 370cc8db, Got 370cc8db waddr 222
# time =              91780000 Write success: Expected fb3d71d6, Got fb3d71d6 waddr 223
# time =              91780000 Write success: Expected aa81d54b, Got aa81d54b waddr 224
# time =              91780000 Write success: Expected 5c05bd61, Got 5c05bd61 waddr 225
# time =              91780000 Write success: Expected 3dd90748, Got 3dd90748 waddr 226
# time =              91780000 Write success: Expected b40b8d13, Got b40b8d13 waddr 227
# time =              91780000 Write success: Expected 120e4b16, Got 120e4b16 waddr 228
# time =              91780000 Write success: Expected b796249f, Got b796249f waddr 229
# time =              91780000 Write success: Expected f3f98c5d, Got f3f98c5d waddr 230
# time =              91780000 Write success: Expected 3ec95156, Got 3ec95156 waddr 231
# time =              91780000 Write success: Expected 69777ad3, Got 69777ad3 waddr 232
# time =              91780000 Write success: Expected 33fb9650, Got 33fb9650 waddr 233
# time =              91780000 Write success: Expected 8b991bf1, Got 8b991bf1 waddr 234
# time =              91780000 Write success: Expected 1117b1d2, Got 1117b1d2 waddr 235
# time =              91780000 Write success: Expected 9cf66107, Got 9cf66107 waddr 236
# time =              91780000 Write success: Expected 8ccfb1e1, Got 8ccfb1e1 waddr 237
# time =              91780000 Write success: Expected 3d928b16, Got 3d928b16 waddr 238
# time =              91780000 Write success: Expected 541b1c68, Got 541b1c68 waddr 239
# time =              91780000 Write success: Expected af373bb0, Got af373bb0 waddr 240
# time =              91780000 Write success: Expected 142bc51e, Got 142bc51e waddr 241
# time =              91780000 Write success: Expected 20491a41, Got 20491a41 waddr 242
# time =              91780000 Write success: Expected 25718d9f, Got 25718d9f waddr 243
# time =              91780000 Write success: Expected 52849b06, Got 52849b06 waddr 244
# time =              91780000 Write success: Expected 5ace2a90, Got 5ace2a90 waddr 245
```

```
# ================================================================
# burst read of rlength 216 from byte raddress 870 (word raddress 217)
# time =             176355000 Read success: Expected 7a6dcd6, Got 7a6dcd6 raddr 217
# time =             176375000 Read success: Expected 42af01da, Got 42af01da raddr 218
# time =             176395000 Read success: Expected dcecace, Got dcecace raddr 219
# time =             176415000 Read success: Expected b367737e, Got b367737e raddr 220
# time =             176435000 Read success: Expected b7f8c479, Got b7f8c479 raddr 221
# time =             176455000 Read success: Expected 370cc8db, Got 370cc8db raddr 222
# time =             176475000 Read success: Expected fb3d71d6, Got fb3d71d6 raddr 223
# time =             176495000 Read success: Expected aa81d54b, Got aa81d54b raddr 224
# time =             176515000 Read success: Expected 5c05bd61, Got 5c05bd61 raddr 225
# time =             176535000 Read success: Expected 3dd90748, Got 3dd90748 raddr 226
# time =             176565000 Read success: Expected b40b8d13, Got b40b8d13 raddr 227
# time =             176585000 Read success: Expected 120e4b16, Got 120e4b16 raddr 228
# time =             176615000 Read success: Expected b796249f, Got b796249f raddr 229
# time =             176635000 Read success: Expected f3f98c5d, Got f3f98c5d raddr 230
# time =             176655000 Read success: Expected 3ec95156, Got 3ec95156 raddr 231
# time =             176675000 Read success: Expected 69777ad3, Got 69777ad3 raddr 232
# time =             176695000 Read success: Expected 33fb9650, Got 33fb9650 raddr 233
# time =             176715000 Read success: Expected 8b991bf1, Got 8b991bf1 raddr 234
# time =             176735000 Read success: Expected 1117b1d2, Got 1117b1d2 raddr 235
# time =             176755000 Read success: Expected 9cf66107, Got 9cf66107 raddr 236
# time =             176775000 Read success: Expected 8ccfb1e1, Got 8ccfb1e1 raddr 237
# time =             176795000 Read success: Expected 3d928b16, Got 3d928b16 raddr 238
# time =             176815000 Read success: Expected 541b1c68, Got 541b1c68 raddr 239
# time =             176835000 Read success: Expected af373bb0, Got af373bb0 raddr 240
# time =             176855000 Read success: Expected 142bc51e, Got 142bc51e raddr 241
# time =             176875000 Read success: Expected 20491a41, Got 20491a41 raddr 242
# time =             176895000 Read success: Expected 25718d9f, Got 25718d9f raddr 243
# time =             176915000 Read success: Expected 52849b06, Got 52849b06 raddr 244
# time =             176935000 Read success: Expected 5ace2a90, Got 5ace2a90 raddr 245
# time =             176955000 Read success: Expected 40c5e1f5, Got 40c5e1f5 raddr 246
```

Long burst

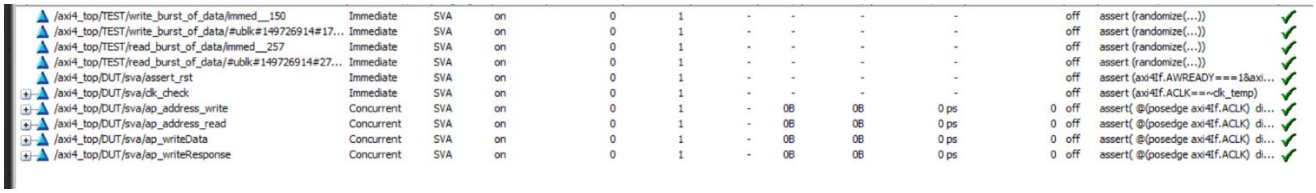


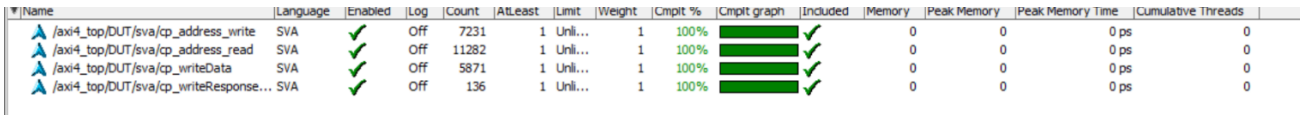
Read and write operation also out of bound operation



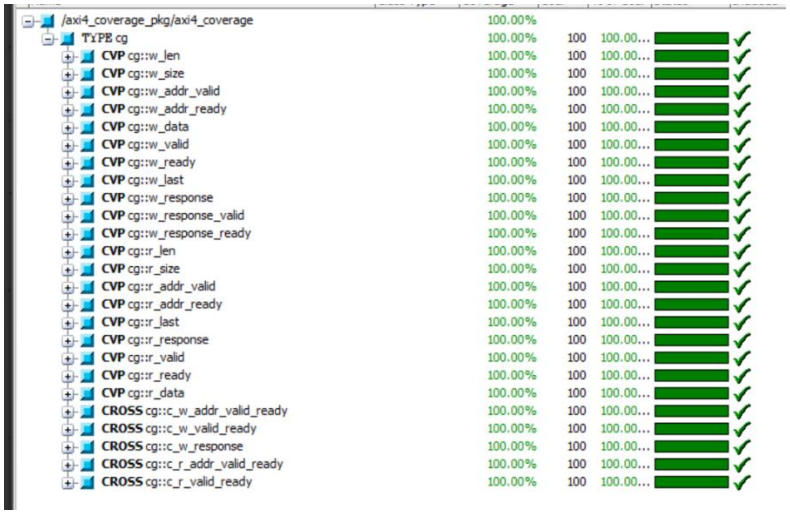
Coverage

Assertion Coverage





Functional Coverage



Code Coverage

Memory Code Coverage

```
Expression Coverage:
    Enabled Coverage              Bins    Covered     Misses  Coverage
    ----------------              ----    -------     ------  --------
    Expressions                      5          5          0   100.00%
```

```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                       7         7         0   100.00%
```

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         5         5         0   100.00%
```

```
==============================Toggle Details================================

Toggle Coverage for instance /\axi4_top#DUT /mem_inst --

                                    Node      1H->0L      0L->1H  "Coverage"
                                    ----------------------------------------
                                     clk           1           1      100.00
                                  j[31-0]          0           0        0.00
                               mem_addr[0-9]       1           1      100.00
                                  mem_en           1           1      100.00
                              mem_rdata[31-0]      1           1      100.00
                              mem_wdata[0-31]      1           1      100.00
                                  mem_we           1           1      100.00
                                   rst_n           1           1      100.00

Total Node Count     =        110
Toggled Node Count   =         78
Untoggled Node Count =         32

Toggle Coverage      =      70.90% (156 of 220 bins)
```

J is just a loop iterator execute at zero time

Slave Code Coverage

Branch Coverage

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        35        33         2    94.28%
```

```
-----------------------------------CASE Branch-----------------------------------
   108                              18729     Count coming in to CASE
   109           1                  11708                   W_IDLE: begin

   126           1                    136                   W_ADDR: begin

   132           1                   6737                   W_DATA: begin

   163           1                    148                   W_RESP: begin

   171           1                  ***0***                 default: write_state <= W_IDLE;

Branch totals: 4 hits of 5 branches = 80.00%
```

```
-----------------------------------CASE Branch-----------------------------------
   178                              18729     Count coming in to CASE
   179           1                   7565                    R_IDLE: begin

   196           1                    135                    R_ADDR: begin

   206           1                  11029                    R_DATA: begin

   242           1                  ***0***                  default: read_state <= R_IDLE;

Branch totals: 3 hits of 4 branches = 75.00%
```

Normal as we expect not enter default during execution of design

Condition Coverage

```
Condition Coverage:
    Enabled Coverage              Bins   Covered    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Conditions                      23        20         3    86.95%
```

```
----------------Focused Condition View-------------------
Line       133 Item    1  (axi4If.WVALID && axi4If.WREADY)
Condition totals: 1 of 2 input terms covered = 50.00%

    Input Term   Covered  Reason for no coverage   Hint
    ----------  --------  ----------------------  --------------
  axi4If.WVALID         Y
  axi4If.WREADY         N  '_0' not hit            Hit '_0'

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  axi4If.WVALID_0       -
  Row   2:          1  axi4If.WVALID_1       axi4If.WREADY
  Row   3:    ***0***  axi4If.WREADY_0       axi4If.WVALID
  Row   4:          1  axi4If.WREADY_1       axi4If.WVALID
```

```
----------------Focused Condition View-------------------
Line       144 Item    1  (axi4If.WLAST || (write_burst_cnt == 0))
Condition totals: 1 of 2 input terms covered = 50.00%

            Input Term   Covered  Reason for no coverage   Hint
            ----------  --------  ----------------------  --------------
           axi4If.WLAST         Y
  (write_burst_cnt == 0)        N  '_1' not hit            Hit '_1'

     Rows:       Hits  FEC Target                Non-masking condition(s)
 ---------  ---------  --------------------      -------------------------
  Row   1:          1  axi4If.WLAST_0           ~(write_burst_cnt == 0)
  Row   2:          1  axi4If.WLAST_1           -
  Row   3:          1  (write_burst_cnt == 0)_0  ~axi4If.WLAST
  Row   4:    ***0***  (write_burst_cnt == 0)_1  ~axi4If.WLAST
```

```
----------------Focused Condition View-------------------
Line       164 Item    1  (axi4If.BREADY && axi4If.BVALID)
Condition totals: 1 of 2 input terms covered = 50.00%

    Input Term   Covered  Reason for no coverage   Hint
    ----------  --------  ----------------------  --------------
  axi4If.BREADY         Y
  axi4If.BVALID         N  '_0' not hit            Hit '_0'

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  axi4If.BREADY_0       -
  Row   2:          1  axi4If.BREADY_1       axi4If.BVALID
  Row   3:    ***0***  axi4If.BVALID_0       axi4If.BREADY
  Row   4:          1  axi4If.BVALID_1       axi4If.BREADY
```

FSM Coverage

```
    Uncovered Transitions :
    -----------------------
Line            Trans_ID         Transition
----            --------         ----------
  81                   2         W_ADDR -> W_IDLE
  81                   4         W_DATA -> W_IDLE


    Summary                      Bins      Hits    Misses  Coverage
    -------                      ----      ----    ------  --------
       FSM States                   4         4         0   100.00%
       FSM Transitions              6         4         2    66.66%
```

```
    Uncovered Transitions :
    -----------------------
Line            Trans_ID         Transition
----            --------         ----------
  82                   2         R_ADDR -> R_IDLE


    Summary                      Bins      Hits    Misses  Coverage
    -------                      ----      ----    ------  --------
       FSM States                   3         3         0   100.00%
       FSM Transitions              4         3         1    75.00%
```

This uncovered transition totally accepted as in our FSM its impossible to go from R_ADDR to R_IDLE or from W_ADDR ,W_DATA to W_IDLE

Statement Coverage

```
Statement Coverage:
    Enabled Coverage             Bins      Hits    Misses  Coverage
    ----------------             ----      ----    ------  --------
    Statements                     85        83         2    97.64%
```

```
  1               ***0***              default: read_state <= R_IDLE;
```

```
  1               ***0***              default: write_state <= W_IDLE;
```

Default never executed expected

Toggle Coverage

```
==============================Toggle Details================================

Toggle Coverage for instance /\axi4_top#DUT  --

                                    Node      1H->0L      0L->1H  "Coverage"
                                    ----------------------------------------
                               mem_addr[9-0]           1           1      100.00
                                      mem_en           1           1      100.00
                            mem_rdata[0-31]            1           1      100.00
                          mem_rdata_reg[31-0]          0           0        0.00
                            mem_wdata[31-0]            1           1      100.00
                                      mem_we           1           1      100.00
                           read_addr[15-0]             1           1      100.00
                         read_addr_incr[0]             1           1      100.00
                         read_addr_incr[1]             0           0        0.00
                         read_addr_incr[2]             1           1      100.00
                        read_addr_incr[3-15]           0           0        0.00
                            read_addr_valid            1           1      100.00
                        read_boundary_cross            1           1      100.00
                        read_burst_cnt[7-0]            1           1      100.00
                        read_burst_len[7-0]            1           1      100.00
                               read_size[2]            0           0        0.00
                               read_size[1]            1           1      100.00
                               read_size[0]            0           0        0.00
                              read_state[2]            0           0        0.00
                           read_state[1-0]             1           1      100.00
                          write_addr[15-0]             1           1      100.00
                         write_addr_incr[0]            1           1      100.00
                         write_addr_incr[1]            0           0        0.00
                         write_addr_incr[2]            1           1      100.00
                        write_addr_incr[3-15]          0           0        0.00
                            write_addr_valid           1           1      100.00
                        write_boundary_cross           1           1      100.00
                        write_burst_cnt[7-0]           1           1      100.00
                        write_burst_len[7-0]           1           1      100.00
                               write_size[2]           0           0        0.00
                               write_size[1]           1           1      100.00
                               write_size[0]           0           0        0.00
                              write_state[2]           0           0        0.00
                           write_state[1-0]            1           1      100.00

Total Node Count     =        220
Toggled Node Count   =        154
Untoggled Node Count =         66

Toggle Coverage      =     70.00% (308 of 440 bins)
```

Mem_rdata_reg is a register that never used in the design

read_size and write_size always take value 2

read_addr_incr and write_addr_incr always take value 4

read_state and write_state as I have specific state to go through them

Total Code Coverage

```
Total Coverage By Instance (filtered view): 91.16%
```

GitHub repository for code details

[RanaAyman54/AXI4-Compliant-Memory-Mapped-Slave]