EE652 - IC DESIGN
UNIVERSAL VERIFICATION METHOD
REPORT

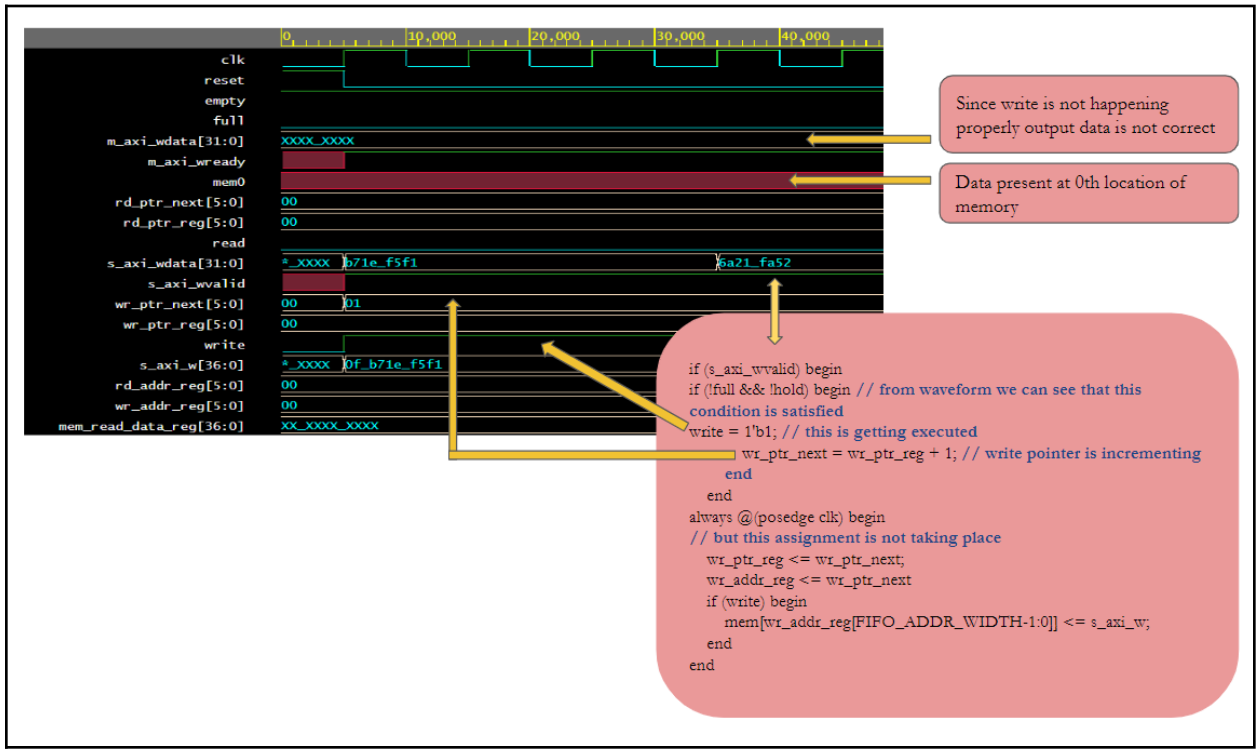
-Sukanya More (20110205)
-Daniel Giftson(20110051)

TEST MODULE 1:
Verification of AXI_FIFO:

AXI_FIFO has two memories, one is associated with the master and the other with the slave.

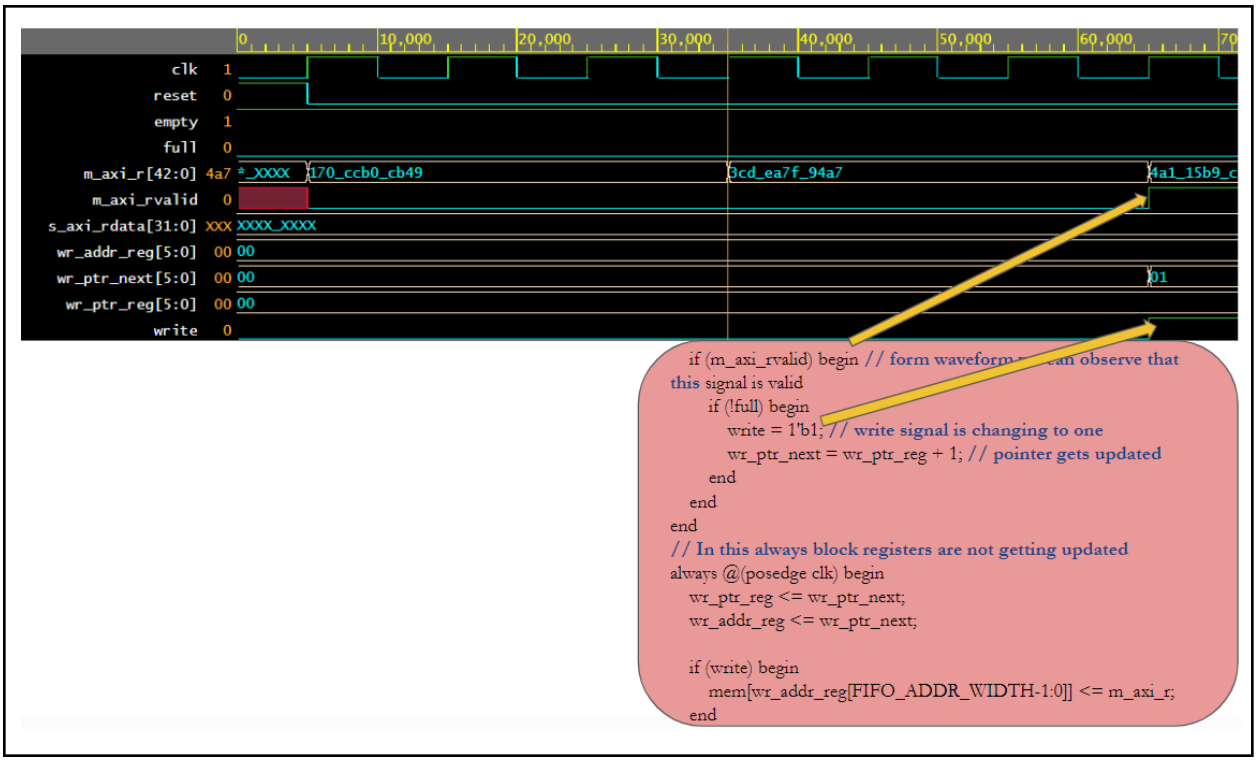
Slave memory status:

All the input signals are reaching correctly, but the always block data is not getting updated, and that's why we are not getting output data.

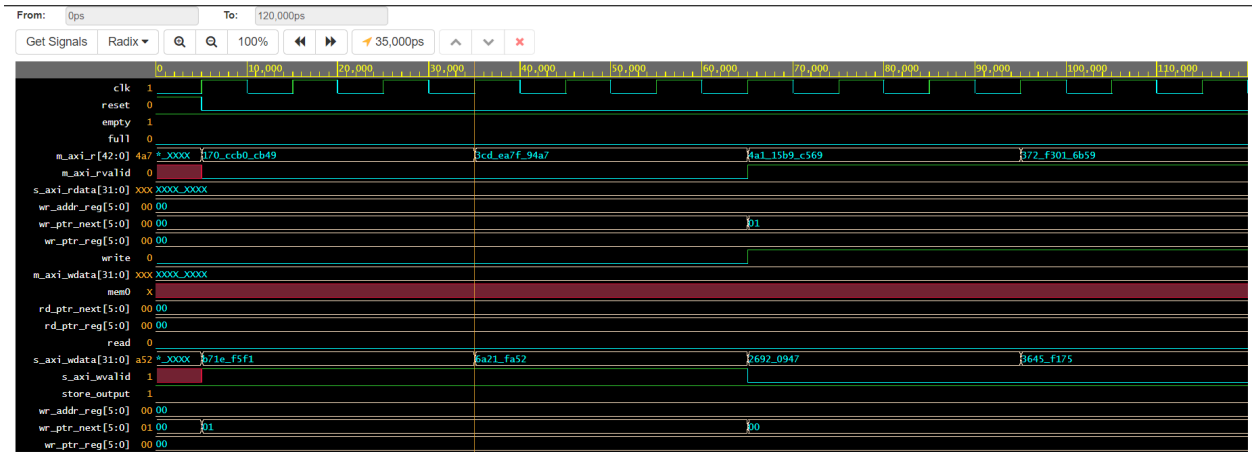


Master Memory Status:

All the input signals are reaching correctly, but in always block data is not getting updated, and that's why we are not getting output data.



Final Result:



Since we were not getting the correct output, we checked another module to understand the UVM testbench.

We checked FIFO that we designed in our assignment.

TEST MODULE 2:
FIFO CODE:

```
module top_fifo(valid, w_en,r_en,din,rst,clk,full,empty,half_full,dout,counter,write_pointer, read_pointer  );
input bit w_en,r_en,clk,rst;
parameter N=16;
parameter M=32;
input  bit [N-1:0] din;
output bit [N-1:0] dout;
output bit full,empty,half_full;
bit [N-1:0]mem[M-1:0]; // declaring an unpacked 2D array
output bit [4:0] write_pointer;
output bit [4:0] read_pointer;
output bit [5:0] counter;
input bit valid;

assign full=(counter==6'b100000);
assign empty=(counter==6'b000000);
assign half_full=(counter==6'b010000);
```

Variable declaration and full, empty, Half_full conditions

```
always_ff @(posedge clk)
begin
  if (valid)
    begin

      priority if(rst)  //active high reset
          begin
          counter<=6'b0;
          write_pointer<=5'b0;
          read_pointer<=5'b0;
          end

          else if ((!r_en)&(!w_en))
          begin
          counter<=counter; //when both read and write enable signals are inactive, counter remains the same
          end

          else if(6'b000000<counter && counter<6'b100000)
          begin

              if(r_en)
              begin
              counter=counter-6'b000001; //decrementing counter
              dout<=mem[read_pointer]; //reading from memory
              read_pointer<=read_pointer+5'b00001; //incrementing read pointer
              end

              if(w_en)
              begin
              counter=counter+6'b000001; //incrementing counter
              mem[write_pointer]<=din; //writing to memory
              write_pointer<=write_pointer+5'b00001; //incrementing write pointer
              end
          end
```

Except counter= 0 & counter= 32 read & write conditions

```
else if(counter==6'b100000)
begin
  assert(!w_en)
      begin
     if(r_en)begin
     $display("Only read can be performed");
     end
 end
  else $error("Can't write, memory is full");
    if (!w_en) //assert statement if write enable is active and counter is at maximum
    begin
    if(r_en) begin
        counter<=counter-6'b00001;
        dout<=mem[read_pointer];
        read_pointer<=read_pointer+5'b00001;
        end
    end
end
else if(counter==6'b0)
begin
  assert(!r_en)begin
    if(w_en)begin
      $display("Only write can be performed");
    end
  end
  else $error("Can't read, memory is empty");
    if (!r_en)  begin
    if(w_en) begin
        counter<=counter+6'b000001;
        mem[write_pointer]<=din;
        write_pointer<=write_pointer+5'b00001;
        end
    end
end
```

Two extreme conditions: when counter =0 & counter =32

OUTPUT:

```
----------DRIVER-------------
- w_en = 1, r_en = 0,din=47755
- full = 0, empty =0, half_full=0, dout=61636, counter= 2
------------------------
----------MONITOR-------------
- w_en = 1, r_en = 0,din=47755
- full = 0, empty =0, half_full=0, dout=61636, counter= 2
------------------------
written data:47755
write is done
write_pointer=2
counter after write operation =3
------------------------
----------DRIVER-------------
- w_en = 1, r_en = 1,din=  147
- full = 0, empty =0, half_full=0, dout=15159, counter= 2
------------------------
----------MONITOR-------------
- w_en = 1, r_en = 1,din=  147
- full = 0, empty =0, half_full=0, dout=15159, counter= 2
------------------------
written data:  147
write is done
write_pointer=3
counter after write operation =4
read_pointer=0
counter after read operation =4
Expected data:61636
Actual data:61636
read is done
read_pointer=0
counter after read operation =3
------------------------
----------DRIVER-------------
- w_en = 1, r_en = 1,din=46252
- full = 0, empty =0, half_full=0, dout=47755, counter= 2
------------------------
----------MONITOR-------------
- w_en = 1, r_en = 1,din=46252
- full = 0, empty =0, half_full=0, dout=47755, counter= 2
------------------------
written data:46252
write is done
write_pointer=0
counter after write operation =4
read_pointer=1
counter after read operation =4
Expected data:15159
Actual data:15159
read is done
```

Write enable = 1 din = 47755

Here dout is the previous data that is written in memory

Read enable = 1 and Dout = 47755

Scoreboard output

```
written data:57668
write is done
write_pointer=1
counter after write operation =4
read_pointer=2
counter after read operation =4
Expected data:47755
Actual data:47755
read is done
read_pointer=2
counter after read operation =3
```

TEST MODULE 3:

AXI_DUAL_PORT_RAM:

We implemented UVM testbench for dual port ram module but not getting the correct output.