Abdelrahman Khaled Fouad -----> Project 1 (FIFO project)

1) Verification plan:

Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
FIFO_1	Testing the Active low Async reset, when it's active outputs must be low while constraining reset signal to be inactive most of the time	Directed testing at the start of the simulation		Assertion in the design to check that the outputs are low
FIFO_2	Testing that Overflow flag is high if write enable is high and FIFO is full	Randomized testing throughout the simulation	bins for cross coverage between wr_en and overflow	Assertion in the design to check that Overflow is high
FIFO_3	Testing that Underflow flag is high if read enable is high and FIFO is empty	Randomized testing throughout the simulation	bins for cross coverage between rd_en and underflow	Assertion in the design to check that Underflow is high
FIFO_4	Testing that write acknowledge is high if write operation is done (Write_enable is high and FIFO is not full)	Randomized testing throughout the simulation	bins for cross coverage between wr_en and wr_ack	Assertion in the design to check that Wr_ack is high
FIFO_5	Testing that full flag is high when count equal to FIFO_DEPTH	Randomized testing throughout the simulation	bins for cross coverage between wr_en, rd_en and Full	Assertion in the design to check that full is high
FIFO_6	Testing that Empty flag is high when count equal to zero	Randomized testing throughout the simulation	bins for cross coverage between wr_en, rd_en and Empty	Assertion in the design to check that empty is high
FIFO_7	Testing that almostfull flag is high when count equal to FIFO_DEPTH-1	Randomized testing throughout the simulation	bins for cross coverage between wr_en, rd_en & almostfull	Assertion in the design to check that almostfull is high
FIFO_8	Testing that almostempty flag is high when count equal to one	Randomized testing throughout the simulation	bins for cross coverage between wr_en, rd_en & almostempty	Assertion in the design to check that almostempty is high

2) Do file:

Excluding invalid cases to happen from the coverage as wr_en=1, rd_en=0, empty=1

run.do | FIFO_top.sv | FIFO_tb.sv | FIFO_monitor.sv | FIFO_if.sv | FIFO.sv | FIFO_shared_pkg.sv | FIFO_transaction_pkg.sv | FIFO_coverage_pkg.sv | FIFO_s

D: > new hard > Sessions Digital verification > Sync FIFO project > run.do

```
vlib work
vlog FIFO_if.sv FIFO.sv FIFO_shared_pkg.sv FIFO_transaction_pkg.sv FIFO_coverage_pkg.sv  FIFO_scoreboard_pkg.sv FIFO_tb.sv FIFO_monitor.sv FIFO_top.sv  +cover -covercells
vsim -voptargs=+acc work.FIFO_top -cover
add wave -position insertpoint sim:/FIFO_top/FIFOif/*
add wave -position insertpoint  \
sim:/FIFO_top/FIFOdut/mem \
sim:/FIFO_top/FIFOdut/wr_ptr \
sim:/FIFO_top/FIFOdut/rd_ptr \
sim:/FIFO_top/FIFOdut/count \
sim:/FIFO_shared_pkg::Error_count \
sim:/FIFO_shared_pkg::Correct_count \
sim:/FIFO_shared_pkg::test_finished
run -all
add wave -position insertpoint  \
sim:/FIFO_top/FIFOmonitor/FIFO_tr_mon \
sim:/FIFO_top/FIFOmonitor/FIFO_sb_mon
restart
run -all
coverage save FIFO_top.ucdb -onexit
coverage exclude -cvgpath {/FIFO_coverage_pkg/FIFO_coverage/FIFO_coverage__1/FIFO_coverage/wr_rd_wr_ack/<auto[0],auto[1],auto[1]>}
{/FIFO_coverage_pkg/FIFO_coverage/FIFO_coverage__1/FIFO_coverage/wr_rd_wr_ack/<auto[0],auto[0],auto[1]>}
{/FIFO_coverage_pkg/FIFO_coverage/FIFO_coverage__1/FIFO_coverage/wr_rd_overflow/<auto[0],auto[1],auto[1]>}
{/FIFO_coverage_pkg/FIFO_coverage/FIFO_coverage__1/FIFO_coverage/wr_rd_overflow/<auto[0],auto[0],auto[1]>}
{/FIFO_coverage_pkg/FIFO_coverage/FIFO_coverage__1/FIFO_coverage/wr_rd_underflow/<auto[1],auto[0],auto[1]>}
{/FIFO_coverage_pkg/FIFO_coverage/FIFO_coverage__1/FIFO_coverage/wr_rd_underflow/<auto[0],auto[0],auto[1]>}
{/FIFO_coverage_pkg/FIFO_coverage/FIFO_coverage__1/FIFO_coverage/wr_rd_full/<auto[1],auto[1],auto[1]>}
{/FIFO_coverage_pkg/FIFO_coverage/FIFO_coverage__1/FIFO_coverage/wr_rd_full/<auto[0],auto[1],auto[1]>}
```

Abdelrahman Khaled Fouad -----> Project 1 (FIFO project)

3) Codes:

Design:

FIFO_if.sv | FIFO.sv | FIFO_shared_pkg.sv | FIFO_transaction_pkg.sv | FIFO_coverage_pkg.sv | FIFO_scoreboard_pkg.sv

D: > new hard > Sessions Digital verification > Sync FIFO project > FIFO.sv

```
////////////////////////////////////////////////////////////////////////////////
// Author: Kareem Waseem
// Course: Digital Verification using SV & UVM
//
// Description: FIFO Design
//
////////////////////////////////////////////////////////////////////////////////
module FIFO(FIFO_if.DUT FIFOif);
parameter FIFO_WIDTH = 16;
parameter FIFO_DEPTH = 8;

logic [FIFO_WIDTH-1:0] data_in,data_out;
logic clk, rst_n, wr_en, rd_en,wr_ack, overflow,full, empty, almostfull, almostempty, underflow;

assign clk = FIFOif.clk;
assign rst_n = FIFOif.rst_n;
assign data_in = FIFOif.data_in;
assign wr_en = FIFOif.wr_en;
assign rd_en = FIFOif.rd_en;
assign FIFOif.data_out=data_out;
assign FIFOif.wr_ack=wr_ack;
assign FIFOif.overflow=overflow;
assign FIFOif.full=full;
assign FIFOif.empty=empty;
assign FIFOif.almostfull=almostfull;
assign FIFOif.almostempty=almostempty;
assign FIFOif.underflow=underflow;

localparam max_fifo_addr = $clog2(FIFO_DEPTH);

reg [FIFO_WIDTH-1:0] mem [FIFO_DEPTH-1:0];

initial $readmemb("FIFO_init.dat", mem);                                //Initializing the FIFO to be empty

reg [max_fifo_addr-1:0] wr_ptr, rd_ptr;
reg [max_fifo_addr:0] count;

always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        wr_ptr <= 0;
        underflow <= 0;                                                 //Reseting the Underflow
        overflow <= 0;                                                  //Reseting the Overflow
        wr_ack <=0;
    end
    else if (wr_en && count < FIFO_DEPTH) begin
        mem[wr_ptr] <= data_in;
        wr_ack <= 1;
        wr_ptr <= wr_ptr + 1;
        overflow <= 0;                                                  //Adding condition that if writing then overflow is zero
    end
    else begin
        wr_ack <= 0;
        if (full & wr_en)
            overflow <= 1;
        else
            overflow <= 0;
    end
end

```

Abdelrahman Khaled Fouad -----> Project 1 (FIFO project)

```
always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        rd_ptr <= 0;
    end
    else if (rd_en && count != 0) begin
        data_out <= mem[rd_ptr];
        rd_ptr <= rd_ptr + 1;
        underflow <= 0;                                    //Adding condition that if reading then Underflow is zero
    end
    else begin                                             //Adding the underflow statements to be sequential and not combinational
        if (empty & rd_en)
            underflow <= 1;
        else
            underflow <= 0;
    end
end

always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        count <= 0;
    end
    else begin
        if  ( ({wr_en, rd_en} == 2'b10) && !full)
            count <= count + 1;
        else if ( ({wr_en, rd_en} == 2'b01) && !empty)
            count <= count - 1;
        else if (({{wr_en, rd_en} == 2'b11}) && full)      //Added conition when FIFO is full and wr and rd enables are high system reads only
            count <= count - 1;
        else if (({{wr_en, rd_en} == 2'b11}) && empty)     //Added conition when FIFO is empty and wr and rd enables are high system writes only
            count <= count + 1;
    end
end

assign full = (count == FIFO_DEPTH)? 1 : 0;
assign empty = (count == 0)? 1 : 0;
assign almostfull = (count == FIFO_DEPTH-1)? 1 : 0;        //AlmostFull is when count equal to FIFO_DEPTH-1 Not FIFO_DEPTH-2 as count is incremented on ptr by 1
assign almostempty = (count == 1)? 1 : 0;

////////////////////////////////Assertions//////////////////////////////
property reset_p;
    @(posedge clk) !rst_n |-> (count==0 && !overflow && !underflow && !full && empty && !almostfull && !almostempty && !wr_ack);
endproperty

property overflow_p;
    @(posedge clk) disable iff (!rst_n) (wr_en && count==8) |=> (overflow && !wr_ack);
endproperty

property underflow_p;
    @(posedge clk) disable iff (!rst_n) (rd_en && count==0) |=> (underflow);
endproperty

property wr_ack_p;
    @(posedge clk) disable iff (!rst_n) (count!=8 && wr_en) |=> wr_ack;
endproperty

property full_p;
    @(posedge clk) disable iff (!rst_n) (count==8) |-> full;
endproperty

property empty_p;
    @(posedge clk) disable iff (!rst_n) (count==0) |-> empty;
endproperty

property almostfull_p;
    @(posedge clk) disable iff (!rst_n) (count==7) |-> almostfull;
endproperty

property almostempty_p;
    @(posedge clk) disable iff (!rst_n) (count==1) |-> almostempty;
endproperty

reset_assertion: assert property(reset_p);
overflow_assertion: assert property(overflow_p);
underflow_assertion: assert property(underflow_p);
wr_ack_assertion: assert property(wr_ack_p);
full_assertion: assert property(full_p);
empty_assertion: assert property(empty_p);
almostfull_assertion: assert property(almostfull_p);
almostempty_assertion: assert property(almostempty_p);

reset_coverage: cover property(reset_p);
overflow_coverage: cover property(overflow_p);
underflow_coverage: cover property(underflow_p);
wr_ack_coverage: cover property(wr_ack_p);
full_coverage: cover property(full_p);
empty_coverage: cover property(empty_p);
almostfull_coverage: cover property(almostfull_p);
almostempty_coverage: cover property(almostempty_p);


endmodule
```

Abdelrahman Khaled Fouad -----> Project 1 (FIFO project)

Testbench:

FIFO_tb.sv | FIFO_monitor.sv | FIFO_if.sv | FIFO.sv | FIFO_shared_pkg.sv | FIFO_transaction_pkg.sv | FIFO_coverage_p

D: > new hard > Sessions Digital verification > Sync FIFO project > FIFO_tb.sv

```
import FIFO_transaction_pkg::*;
import FIFO_scoreboard_pkg::*;
import FIFO_shared_pkg::*;

module FIFO_tb(FIFO_if.tb FIFOif);

FIFO_transaction #(16,8) FIFO_tr =new();

parameter FIFO_WIDTH = 16;
parameter FIFO_DEPTH = 8;

logic [FIFO_WIDTH-1:0] data_in,data_out;
logic clk, rst_n, wr_en, rd_en,wr_ack, overflow,full, empty, almostfull, almostempty, underflow;

assign clk = FIFOif.clk;
assign data_out = FIFOif.data_out;
assign wr_ack = FIFOif.wr_ack;
assign overflow = FIFOif.overflow;
assign underflow = FIFOif.underflow;
assign full=FIFOif.full;
assign empty = FIFOif.empty;
assign almostfull = FIFOif.almostfull;
assign almostempty = FIFOif.almostempty;
assign FIFOif.rst_n = rst_n;
assign FIFOif.data_in = data_in;
assign FIFOif.wr_en = wr_en;
assign FIFOif.rd_en = rd_en;

initial begin

    rst_n=0; data_in=0; wr_en=0; rd_en=0;
    @(negedge clk);

    repeat(1000) begin
        assert(FIFO_tr.randomize());
        rst_n = FIFO_tr.rst_n;
        data_in = FIFO_tr.data_in;
        wr_en = FIFO_tr.wr_en;
        rd_en = FIFO_tr.rd_en;
        @(negedge clk);
    end
    test_finished=1;
end

endmodule
```

Interface:

FIFO_if.sv | FIFO_tb.sv | FIFO.sv | FIFO_shared_pkg.sv | FIFO_transaction_pkg.sv | FIFO_coverage_pkg.sv | FIFO_scoreboa

D: > new hard > Sessions Digital verification > Sync FIFO project > FIFO_if.sv

```
interface FIFO_if(clk);

input bit clk;

parameter FIFO_WIDTH = 16;
parameter FIFO_DEPTH = 8;

logic [FIFO_WIDTH-1:0] data_in;
logic rst_n, wr_en, rd_en;
logic [FIFO_WIDTH-1:0] data_out;
logic wr_ack, overflow;
logic full, empty, almostfull, almostempty, underflow;

modport DUT (
    input data_in,clk, rst_n, wr_en, rd_en,
    output data_out,wr_ack, overflow,full, empty, almostfull, almostempty, underflow
);

modport tb (
    input clk,data_out,wr_ack, overflow,full, empty, almostfull, almostempty, underflow,
    output data_in, rst_n, wr_en, rd_en
);

modport MONITOR(
    input clk,data_in, rst_n, wr_en, rd_en,data_out,wr_ack, overflow,full, empty, almostfull, almostempty, underflow
);

endinterface
```

Abdelrahman Khaled Fouad -----> Project 1 (FIFO project)

Top module:

FIFO_top.sv | FIFO_tb.sv | FIFO_monitor.sv | FIFO_if.sv | FIFO.sv

D: > new hard > Sessions Digital verification > Sync FIFO project > FIFO_top.sv

```
module FIFO_top();
bit clk;

initial begin
    clk=0;
    forever begin
        #1 clk=~clk;
    end
end

FIFO_if FIFOif(clk);
FIFO FIFOdut(FIFOif);
FIFO_tb FIFOtest(FIFOif);
FIFO_monitor FIFOmonitor(FIFOif);

endmodule
```

Monitor:

FIFO_monitor.sv | FIFO_if.sv | FIFO.sv | FIFO_shared_pkg.sv | FIFO_transaction_pkg.sv | FIFO_coverage_pkg.sv

D: > new hard > Sessions Digital verification > Sync FIFO project > FIFO_monitor.sv

```
import FIFO_transaction_pkg::*;
import FIFO_coverage_pkg::*;
import FIFO_scoreboard_pkg::*;
import FIFO_shared_pkg::*;

module FIFO_monitor(FIFO_if.MONITOR FIFOif);

FIFO_transaction FIFO_tr_mon = new();
FIFO_coverage FIFO_cv_mon = new();
FIFO_scoreboard FIFO_sb_mon = new();

initial begin

    forever begin

        @(negedge FIFOif.clk);

        FIFO_tr_mon.clk = FIFOif.clk;
        FIFO_tr_mon.rst_n = FIFOif.rst_n;
        FIFO_tr_mon.data_in = FIFOif.data_in;
        FIFO_tr_mon.wr_en = FIFOif.wr_en;
        FIFO_tr_mon.rd_en = FIFOif.rd_en;
        FIFO_tr_mon.data_out = FIFOif.data_out;
        FIFO_tr_mon.wr_ack = FIFOif.wr_ack;
        FIFO_tr_mon.overflow = FIFOif.overflow;
        FIFO_tr_mon.underflow = FIFOif.underflow;
        FIFO_tr_mon.full = FIFOif.full;
        FIFO_tr_mon.empty = FIFOif.empty;
        FIFO_tr_mon.almostfull = FIFOif.almostfull;
        FIFO_tr_mon.almostempty = FIFOif.almostempty;

    fork
        begin
            FIFO_cv_mon.sample_data(FIFO_tr_mon);
        end
        begin
            FIFO_sb_mon.check_data(FIFO_tr_mon);
        end
    join

    if(test_finished==1) begin
        $display("Error_count=%0d,Correct_count=%0d",Error_count,Correct_count);
        $stop;
    end
    end
end
endmodule
```

Abdelrahman Khaled Fouad -----> Project 1 (FIFO project)

Shared package:

FIFO_if.sv | FIFO_tb.sv | FIFO.sv | FIFO_shared_pkg.sv

D: > new hard > Sessions Digital verification > Sync FIFO project > FIFO_shared_pkg.sv

```
package FIFO_shared_pkg;

int Error_count=0,Correct_count=0;
bit test_finished=0;

endpackage
```

Transaction package:

FIFO_if.sv | FIFO.sv | FIFO_shared_pkg.sv | FIFO_transaction_pkg.sv | FIFO_coverage_pkg.s

D: > new hard > Sessions Digital verification > Sync FIFO project > FIFO_transaction_pkg.sv

```
package FIFO_transaction_pkg;

import FIFO_shared_pkg::*;

class FIFO_transaction #(int FIFO_WIDTH = 16 , int FIFO_DEPTH = 8);

logic [FIFO_WIDTH-1:0] data_out;
logic clk,wr_ack, overflow,full, empty, almostfull, almostempty, underflow;
rand logic rst_n, wr_en, rd_en;
rand logic [FIFO_WIDTH-1:0] data_in;
int RD_EN_ON_DIST,WR_EN_ON_DIST;

function new(int rd_percent = 30,int wr_percent = 70);
    RD_EN_ON_DIST=rd_percent;
    WR_EN_ON_DIST=wr_percent;
endfunction

constraint reset{rst_n dist {0:=5,1:=95};}
constraint write_enable{wr_en dist{0:=(100-WR_EN_ON_DIST),1:=WR_EN_ON_DIST};}
constraint read_enable{rd_en dist{0:=(100-RD_EN_ON_DIST),1:=RD_EN_ON_DIST};}


endclass

endpackage
```

Coverage package:

FIFO_if.sv | FIFO.sv | FIFO_shared_pkg.sv | FIFO_transaction_pkg.sv | FIFO_coverage_pkg.sv

D: > new hard > Sessions Digital verification > Sync FIFO project > FIFO_coverage_pkg.sv

```
package FIFO_coverage_pkg;

import FIFO_transaction_pkg::*;
import FIFO_shared_pkg::*;


class FIFO_coverage #(int FIFO_WIDTH = 16 , int FIFO_DEPTH = 8);
FIFO_transaction #(16,8) F_cvg_txn = new();
covergroup FIFO_coverage ;

    data_out_cp: coverpoint F_cvg_txn.data_out {
        bins data_out_bin = {[0:$]};
        option.weight=0;
        }

    wr_rd_data_out: cross F_cvg_txn.wr_en,F_cvg_txn.rd_en,data_out_cp;
    wr_rd_wr_ack: cross F_cvg_txn.wr_en,F_cvg_txn.rd_en,F_cvg_txn.wr_ack;
    wr_rd_overflow: cross F_cvg_txn.wr_en,F_cvg_txn.rd_en,F_cvg_txn.overflow;
    wr_rd_underflow: cross F_cvg_txn.wr_en,F_cvg_txn.rd_en,F_cvg_txn.underflow;
    wr_rd_full: cross F_cvg_txn.wr_en,F_cvg_txn.rd_en,F_cvg_txn.full;
    wr_rd_empty: cross F_cvg_txn.wr_en,F_cvg_txn.rd_en,F_cvg_txn.empty;
    wr_rd_almostfull: cross F_cvg_txn.wr_en,F_cvg_txn.rd_en,F_cvg_txn.almostfull;
    wr_rd_almostempty: cross F_cvg_txn.wr_en,F_cvg_txn.rd_en,F_cvg_txn.almostempty;
    //option.auto_bin_max=0;

endgroup

function new();
    FIFO_coverage = new();
endfunction

function sample_data(FIFO_transaction #(16,8) F_txn);
    F_cvg_txn = F_txn;
    FIFO_coverage.sample();
endfunction
endclass
endpackage
```

Abdelrahman Khaled Fouad -----> Project 1 (FIFO project)

Scoreboard Package:

FIFO_if.sv | FIFO.sv | FIFO_shared_pkg.sv | FIFO_transaction_pkg.sv | FIFO_coverage_pkg.sv | FIFO_scoreboard_pkg.sv | FIFO_monitor.sv | FIFO_tb.sv | FIFO_

D: > new hard > Sessions Digital verification > Sync FIFO project > FIFO_scoreboard_pkg.sv

```
package FIFO_scoreboard_pkg;

import FIFO_transaction_pkg::*;
import FIFO_shared_pkg::*;

class FIFO_scoreboard #(int FIFO_WIDTH = 16 , int FIFO_DEPTH = 8);

//FIFO queue
logic [FIFO_WIDTH-1:0] FIFO_queue [$];

int count=0;
logic [FIFO_WIDTH-1:0] data_out_ref;
logic wr_ack_ref, overflow_ref,underflow_ref,full_ref, empty_ref, almostfull_ref, almostempty_ref;

function void check_data(FIFO_transaction #(16,8) FIFO_check);
    reference_model(FIFO_check);

    if((FIFO_check.data_out!=data_out_ref) || (FIFO_check.wr_ack!=wr_ack_ref) || (FIFO_check.overflow!=overflow_ref) || (FIFO_check.underflow!=underflow_ref) ||
    (FIFO_check.full!=full_ref) || (FIFO_check.empty!=empty_ref) || (FIFO_check.almostfull!=almostfull_ref) || (FIFO_check.almostempty!= almostempty_ref)) begin
        Error_count++;
        $display("Error in Checking values");
    end
    else
        Correct_count++;
endfunction

function void reference_model(FIFO_transaction #(16,8) FIFO_ref);

    ////////////////////////////////Reset Signal//////////////////////////////
    if(!FIFO_ref.rst_n) begin
        FIFO_queue.delete();
        count=0; wr_ack_ref=0; overflow_ref=0; underflow_ref=0; full_ref=0; empty_ref=1; almostfull_ref=0; almostempty_ref=0;
    end
    else begin

    ////////////////////////////////Reading and Writing//////////////////////////////
        //write only
        if(FIFO_ref.rd_en && FIFO_ref.wr_en && count==0) begin
            underflow_ref=1;
            FIFO_queue.push_front(FIFO_ref.data_in);
            wr_ack_ref=1;
            overflow_ref=0;
            almostempty_ref=1;
            almostfull_ref=0;
            empty_ref=0;
            full_ref=0;

        count++;
        end
        //Read only
        else if(FIFO_ref.rd_en && FIFO_ref.wr_en && count==8) begin
            overflow_ref=1;
            data_out_ref=FIFO_queue.pop_back();
            wr_ack_ref=0;
            underflow_ref=0;
            almostempty_ref=0;
            almostfull_ref=1;
            empty_ref=0;
            full_ref=0;

        count--;
        end
        //Write and Read Together
        else begin
            if(FIFO_ref.wr_en && count < FIFO_DEPTH) begin
                FIFO_queue.push_front(FIFO_ref.data_in);
                count++;
            end
            if(FIFO_ref.rd_en && (count != 0)) begin
                data_out_ref=FIFO_queue.pop_back();
                count--;
            end

    //All Control signals
    ////////////////////////////////Over and Under flow Signals//////////////////////////////
        //Overflow signal
        if(FIFO_ref.wr_en && full_ref)  overflow_ref=1;
        else    overflow_ref=0;
        //Underflow signal
        if(FIFO_ref.rd_en && empty_ref)  underflow_ref=1;
        else    underflow_ref=0;
    ////////////////////////////////Write acknowledge Signal//////////////////////////////
        if((FIFO_ref.rd_en && FIFO_ref.wr_en && empty_ref)||(FIFO_ref.wr_en && count <= FIFO_DEPTH && overflow_ref==0)) wr_ack_ref=1;
        else wr_ack_ref=0;
    ////////////////////////////////full and empty  Signals//////////////////////////////
        //Full signal
        if(count == FIFO_DEPTH)  full_ref=1;
        else    full_ref =0;
        //Empty signal
        if(count == 0)  empty_ref=1;
        else    empty_ref=0;
```

Abdelrahman Khaled Fouad -----> Project 1 (FIFO project)

```
92      ////////////////////////////Almost empty and full Signals////////////////////////////
93          //Almost empty signal
94          if(count == 1)    almostempty_ref=1;
95          else    almostempty_ref=0;
96          //Almost full signal
97          if(count==(FIFO_DEPTH-1)) almostfull_ref=1;
98          else    almostfull_ref=0;
99
100         end
101     end
102 endfunction
103 endclass
104 endpackage
```

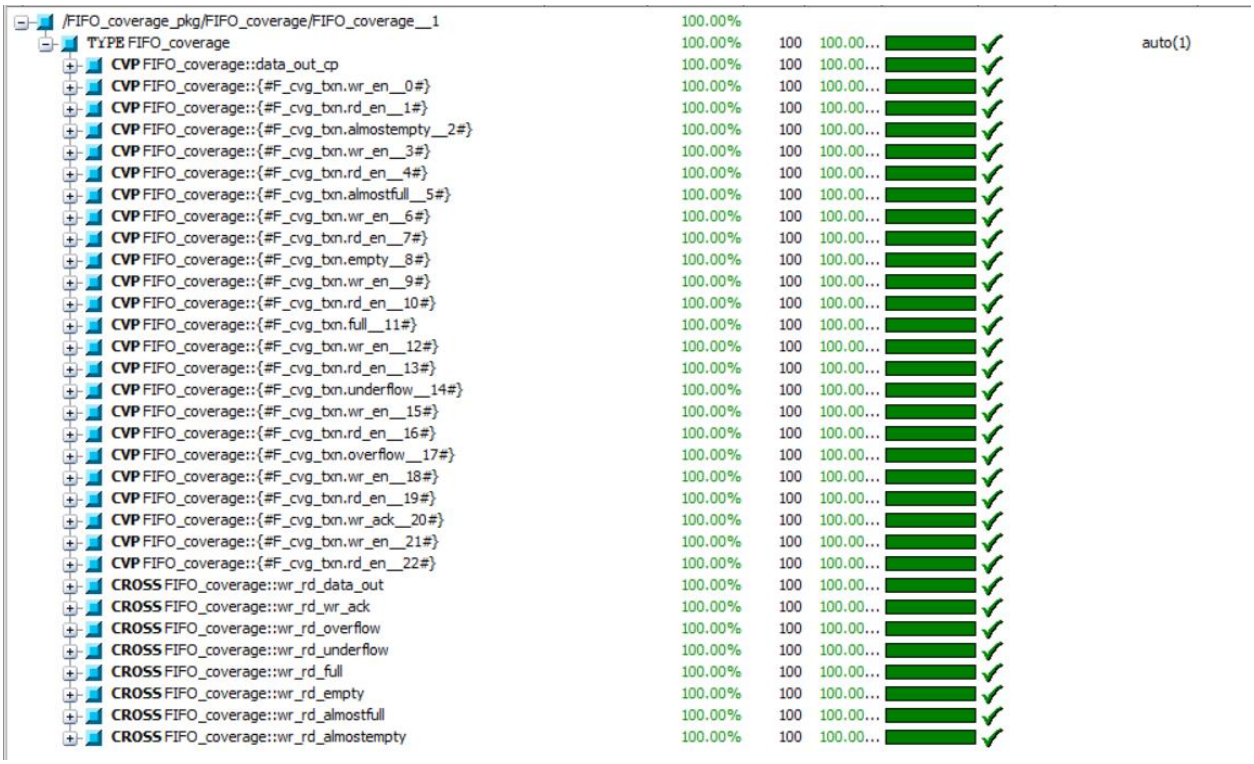
4) Questa snippets:

Bugs in the design are explained in the comments section and updated in the design.

Transcript:

```
# Error_count=0,Correct_count=1002
# ** Note: $stop    : FIFO_monitor.sv(45)
#    Time: 2004 ns  Iteration: 1  Instance: /FIFO_top/FIFOmonitor
# Break in Module FIFO_monitor at FIFO_monitor.sv line 45
```

Cross Coverage:



Abdelrahman Khaled Fouad -----> Project 1 (FIFO project)

Assertion coverage and Directive coverage:

Assertions

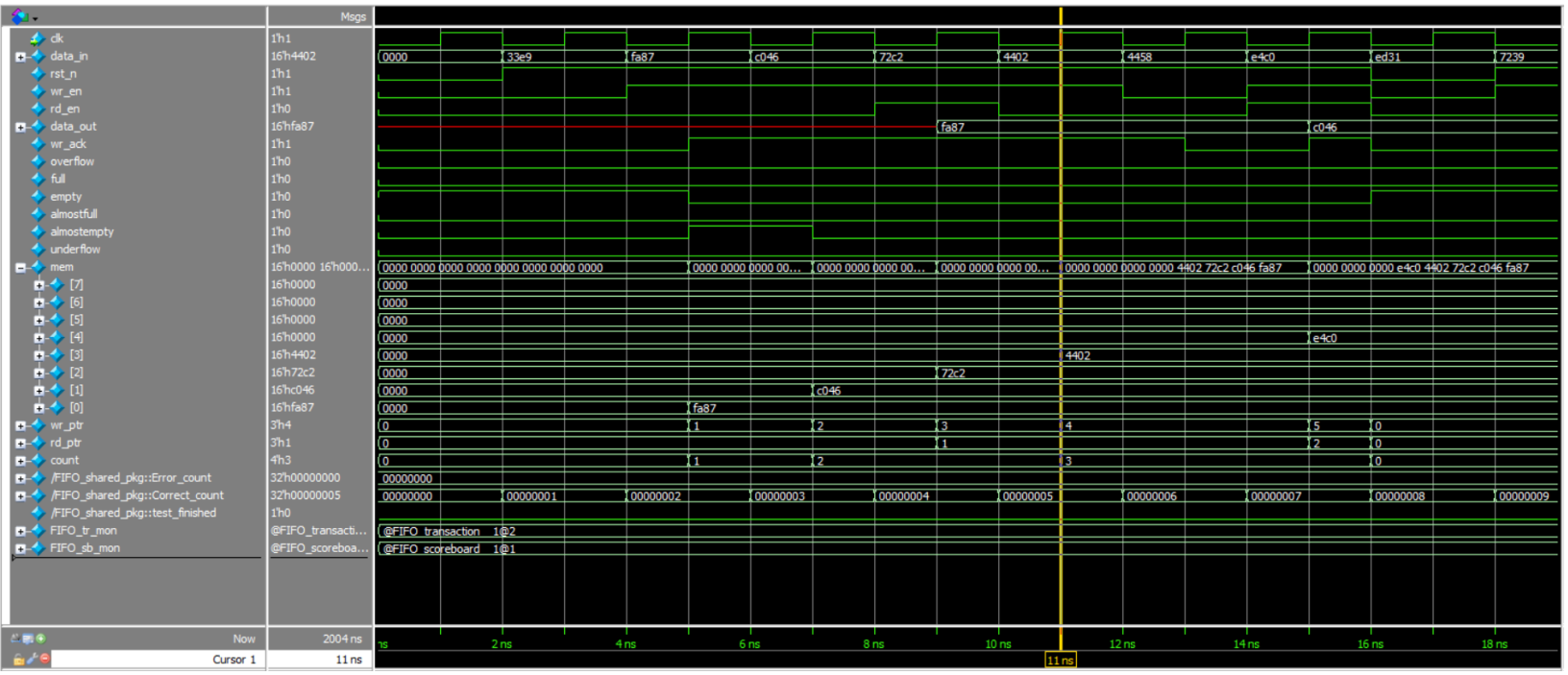
Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Pe
/FIFO_top/FIFOdut/reset_assertion	Concurrent	SVA	on	0	1	-	0B	
/FIFO_top/FIFOdut/overflow_assertion	Concurrent	SVA	on	0	1	-	0B	
/FIFO_top/FIFOdut/underflow_assertion	Concurrent	SVA	on	0	1	-	0B	
/FIFO_top/FIFOdut/wr_ack_assertion	Concurrent	SVA	on	0	1	-	0B	
/FIFO_top/FIFOdut/full_assertion	Concurrent	SVA	on	0	1	-	0B	
/FIFO_top/FIFOdut/empty_assertion	Concurrent	SVA	on	0	1	-	0B	
/FIFO_top/FIFOdut/almostfull_assertion	Concurrent	SVA	on	0	1	-	0B	
/FIFO_top/FIFOdut/almostempty_assertion	Concurrent	SVA	on	0	1	-	0B	
/FIFO_top/FIFOtest/#ublk#182146786#35/immed__36	Immediate	SVA	on	0	1	-	-	

Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak
/FIFO_top/FIFOdut/reset_coverage	SVA	✓	Off	56	1	Unli...	1	100%		✓	0	0	
/FIFO_top/FIFOdut/overflow_coverage	SVA	✓	Off	128	1	Unli...	1	100%		✓	0	0	
/FIFO_top/FIFOdut/underflow_coverage	SVA	✓	Off	22	1	Unli...	1	100%		✓	0	0	
/FIFO_top/FIFOdut/wr_ack_coverage	SVA	✓	Off	482	1	Unli...	1	100%		✓	0	0	
/FIFO_top/FIFOdut/full_coverage	SVA	✓	Off	204	1	Unli...	1	100%		✓	0	0	
/FIFO_top/FIFOdut/empty_coverage	SVA	✓	Off	88	1	Unli...	1	100%		✓	0	0	
/FIFO_top/FIFOdut/almostfull_coverage	SVA	✓	Off	136	1	Unli...	1	100%		✓	0	0	
/FIFO_top/FIFOdut/almostempty_coverage	SVA	✓	Off	102	1	Unli...	1	100%		✓	0	0	

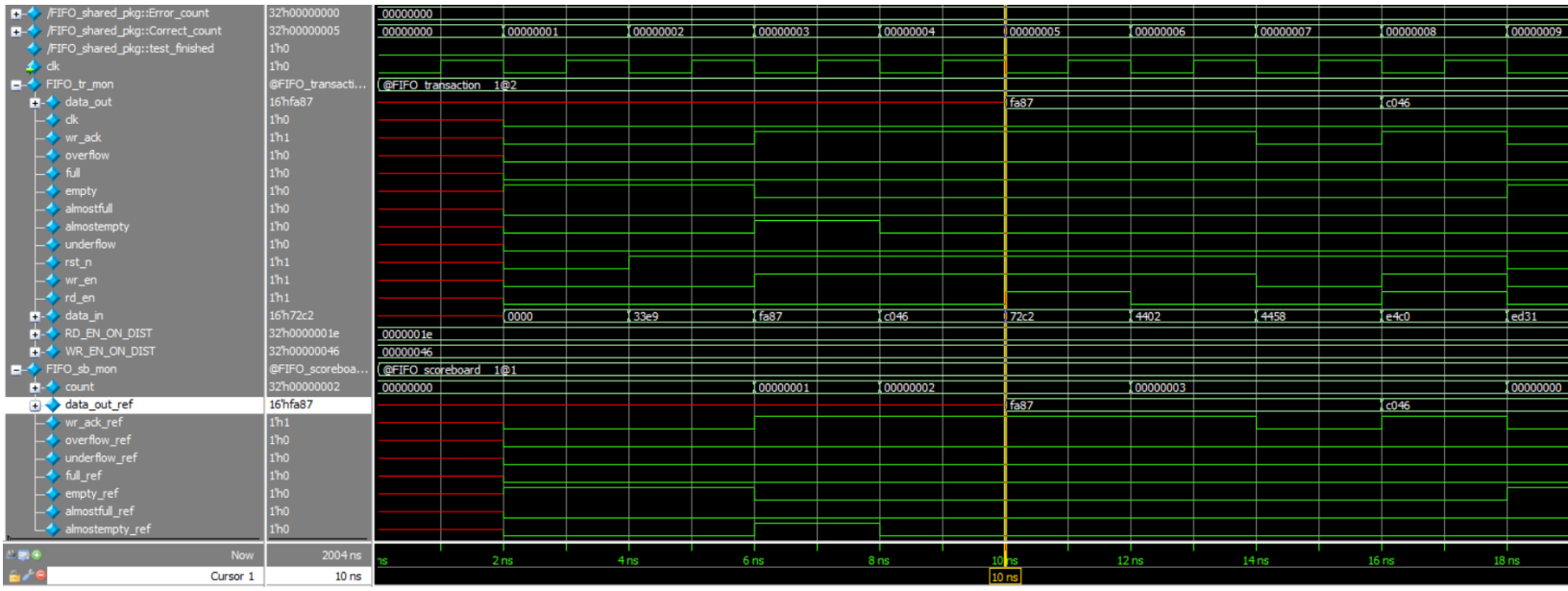
Test cases:

Initializing the FIFO with zeros then activating the reset then random write and read operations.



Abdelrahman Khaled Fouad -----> Project 1 (FIFO project)

Comparing Reference model with Transaction model.



Overall view of the simulation.