Name: Yousef Wael Mahmoud Alkattan

Question1)

1. Fixed Design & Assertions

```
import shared_pkg::*;

module FIFO(FIFO_interface.dut intf);


localparam max_fifo_addr = $clog2(FIFO_DEPTH);

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

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

always @(posedge intf.clk or negedge intf.rst_n) begin
        if (!intf.rst_n) begin
                wr_ptr <= 0;
                intf.wr_ack <= 0; //fixed
                intf.overflow <= 0; //fixed
        end
        else if (intf.wr_en && count < FIFO_DEPTH) begin
                mem[wr_ptr] <= intf.data_in;
                intf.wr_ack <= 1;
                wr_ptr <= wr_ptr + 1;
        end
        else begin
                intf.wr_ack <= 0;
                if (intf.full && intf.wr_en) //fixed was & only
                        intf.overflow <= 1;
                else
                        intf.overflow <= 0;
        end
end

always @(posedge intf.clk or negedge intf.rst_n) begin
        if (!intf.rst_n) begin
                rd_ptr <= 0;
                intf.underflow <= 0; //fixed
                intf.data_out <= 0;  //fixed
        end
        else if (intf.rd_en && count != 0) begin
                intf.data_out <= mem[rd_ptr];
                rd_ptr <= rd_ptr + 1;
        end
        else begin  //fixed underflow is sequential logic
                if (intf.empty && intf.rd_en)
                        intf.underflow <= 1;
                else
                        intf.underflow <= 0;
        end
end
```

```
always @(posedge intf.clk or negedge intf.rst_n) begin
        if (!intf.rst_n) begin
                count <= 0;
        end
        else begin
                if (({intf.wr_en, intf.rd_en} == 2'b11) && intf.full) //fixed fine if both on but full not handled
                        count <= count - 1;
                else if (({intf.wr_en, intf.rd_en} == 2'b11) && intf.empty) //fixed fine if both on but empty not handled
                        count <= count + 1;
                else if ( ({intf.wr_en, intf.rd_en} == 2'b10) && !intf.full)
                        count <= count + 1;
                else if ( ({intf.wr_en, intf.rd_en} == 2'b01) && !intf.empty)
                        count <= count - 1;
        end
end

assign intf.full = (count == FIFO_DEPTH)? 1 : 0;
assign intf.empty = (count == 0 && intf.rst_n)? 1 : 0; //fixed

assign intf.almostfull = (count == FIFO_DEPTH-1)? 1 : 0; //fixed -1 not -2
assign intf.almostempty = (count == 1)? 1 : 0;


//assertions

`ifdef SIM

// 1. Reset Behavior
always_comb begin : rst_n_assert
    if (!intf.rst_n) begin
        assert final (count == 0);
        assert final (wr_ptr == 0);
        assert final (rd_ptr == 0);

        assert final (intf.wr_ack == 0);
        assert final (intf.overflow == 0);
        assert final (intf.underflow == 0);
        assert final (intf.data_out == 0);

        assert final (intf.full == 0);
        assert final (intf.empty == 0);
        assert final (intf.almostfull == 0);
        assert final (intf.almostempty == 0);
    end
end
```

```
// 2. Write Acknowledge
ack: assert property (@(posedge intf.clk) disable iff(!intf.rst_n)
    (intf.wr_en && !intf.full) |=> intf.wr_ack);

// 3. Overflow Detection
overflow: assert property (@(posedge intf.clk) disable iff(!intf.rst_n)
    (intf.full && intf.wr_en) |=> intf.overflow);

// 4. Underflow Detection
underflow: assert property (@(posedge intf.clk) disable iff(!intf.rst_n)
    (intf.empty && intf.rd_en) |=> intf.underflow);

// 5. Empty Flag Assertion
empty: assert property (@(posedge intf.clk) disable iff(!intf.rst_n)
    (count == 0) |-> intf.empty);

// 6. Full Flag Assertion
full: assert property (@(posedge intf.clk) disable iff(!intf.rst_n)
    (count == FIFO_DEPTH) |-> intf.full);

// 7. Almost Full Condition
almostfull: assert property (@(posedge intf.clk) disable iff(!intf.rst_n)
    (count == FIFO_DEPTH-1) |-> intf.almostfull);

// 8. Almost Empty Condition
almostempty: assert property (@(posedge intf.clk) disable iff(!intf.rst_n)
    (count == 1) |-> intf.almostempty);

// 9. Pointer Wraparound
wr_ptr_assert: assert property (@(posedge intf.clk) disable iff(!intf.rst_n)
    (intf.wr_en && count < FIFO_DEPTH) |=>
    (wr_ptr == 0 ? $past(wr_ptr) == FIFO_DEPTH-1 : wr_ptr == $past(wr_ptr) + 1));

rd_ptr_assert: assert property (@(posedge intf.clk) disable iff(!intf.rst_n)
    (intf.rd_en && count != 0) |=>
    (rd_ptr == 0 ? $past(rd_ptr) == FIFO_DEPTH-1 : rd_ptr == $past(rd_ptr) + 1));

// 10. Pointer Threshold
count_range: assert property (@(posedge intf.clk) disable iff(!intf.rst_n)
    (count <= FIFO_DEPTH));

wr_ptr_range: assert property (@(posedge intf.clk) disable iff(!intf.rst_n)
    (wr_ptr < FIFO_DEPTH));

rd_ptr_range: assert property (@(posedge intf.clk) disable iff(!intf.rst_n)
    (rd_ptr < FIFO_DEPTH));

`endif


endmodule
```

2. Interface

File Edit Format View Help

```
interface FIFO_interface (clk);
import shared_pkg::*;
    input bit clk;

    reg [FIFO_WIDTH-1:0] data_in;
    reg rst_n, wr_en, rd_en;

    reg [FIFO_WIDTH-1:0] data_out;
    reg wr_ack, overflow;
    reg full, empty, almostfull, almostempty, underflow;

    modport dut (
        input clk,
        input data_in, rst_n, wr_en, rd_en,

        output data_out,
        output wr_ack, overflow,
        output full, empty, almostfull, almostempty, underflow
    );


    modport tb (
        input clk,
        input data_out,
        input wr_ack, overflow,
        input full, empty, almostfull, almostempty, underflow,

        output data_in, rst_n, wr_en, rd_en
    );

    modport moniter (
        input clk,
        input data_in, rst_n, wr_en, rd_en,

        input data_out,
        input wr_ack, overflow,
        input full, empty, almostfull, almostempty, underflow

    );
endinterface
```

3. Shared Pkg

```
File  Edit  Format  View  Help
package shared_pkg;
        parameter FIFO_WIDTH = 16;
        parameter FIFO_DEPTH = 8;

        bit test_finished;
        int error_counter = 0;
        int correct_counter = 0;

endpackage
```

4. Transaction

```
package transaction_pkg;

import shared_pkg::*;

class FIFO_transaction;

    // FIFO inputs and outputs
    rand bit [FIFO_WIDTH-1:0] data_in;
    rand bit rst_n, wr_en, rd_en;
    bit [FIFO_WIDTH-1:0] data_out;
    bit wr_ack, overflow;
    bit full, empty, almostfull, almostempty, underflow;

    // Distribution control variables
    rand int RD_EN_ON_DIST, WR_EN_ON_DIST;

    // Constructor with default values
    function new(int rd_dist = 30, int wr_dist = 70);
        this.RD_EN_ON_DIST = rd_dist;
        this.WR_EN_ON_DIST = wr_dist;
    endfunction

    // Constraint 1
    constraint reset_less_often {
        rst_n dist {1 := 99, 0 := 1};
    }

    // Constraint 2:
    constraint wr_en_distribution {
        wr_en dist {1 := WR_EN_ON_DIST, 0 := 100 - WR_EN_ON_DIST};
    }

    // Constraint 3:
    constraint rd_en_distribution {
        rd_en dist {1 := RD_EN_ON_DIST, 0 := 100 - RD_EN_ON_DIST};
    }

endclass

endpackage
```

5. Coverage

```
package coverage_pkg;
import transaction_pkg::*;
import shared_pkg::*;

class FIFO_coverage;
  FIFO_transaction F_cvg_txn = new();

  covergroup CVG;

    // Coverpoints
    wr_en_cp:           coverpoint F_cvg_txn.wr_en { bins active = {1}; bins inactive = {
    rd_en_cp:           coverpoint F_cvg_txn.rd_en { bins active = {1}; bins inactive = {

    wr_ack_cp:          coverpoint F_cvg_txn.wr_ack { bins active = {1}; bins inactive =
    full_cp:            coverpoint F_cvg_txn.full { bins active = {1}; bins inactive = {0
    empty_cp:           coverpoint F_cvg_txn.empty { bins active = {1}; bins inactive = {
    almostfull_cp:      coverpoint F_cvg_txn.almostfull { bins active = {1}; bins inactiv
    almostempty_cp:     coverpoint F_cvg_txn.almostempty { bins active = {1}; bins inacti
    underflow_cp:       coverpoint F_cvg_txn.underflow {bins active = {1};bins inactive =
    overflow_cp:        coverpoint F_cvg_txn.overflow {bins active = {1};bins inactive =

    // 7 required cross coverages
    cross_wr_rd_wrack:        cross wr_en_cp, rd_en_cp, wr_ack_cp;
    cross_wr_rd_full:         cross wr_en_cp, rd_en_cp, full_cp;
    cross_wr_rd_empty:        cross wr_en_cp, rd_en_cp, empty_cp;
    cross_wr_rd_almostfull:   cross wr_en_cp, rd_en_cp, almostfull_cp;
    cross_wr_rd_almostempty:  cross wr_en_cp, rd_en_cp, almostempty_cp;
    cross_wr_rd_underflow:   cross wr_en_cp, rd_en_cp, underflow_cp;
    cross_wr_rd_overflow:  cross wr_en_cp, rd_en_cp, overflow_cp;

  endgroup

  function new();
    CVG = new();
  endfunction

  function void sample_data(FIFO_transaction F_txn);
    F_cvg_txn = F_txn;
    CVG.sample();
  endfunction

endclass
endpackage
```

6. Scoreboard

```
package scoreboard_pkg;
  import transaction_pkg::*;
  import shared_pkg::*;

  class FIFO_scoreboard;

    // Output reference variables
    bit [FIFO_WIDTH-1:0] data_out_ref;
    bit wr_ack_ref, overflow_ref;
    bit full_ref, empty_ref, almostfull_ref;
    bit almostempty_ref, underflow_ref;

    // Reference model
    function void reference_model(
      input bit [FIFO_WIDTH-1:0] data_out1,
      input bit wr_ack1, overflow1, full1, empty1,
      input bit almostfull1, almostempty1, underflow1
    );
      data_out_ref     = data_out1;
      wr_ack_ref       = wr_ack1;
      overflow_ref     = overflow1;
      full_ref         = full1;
      empty_ref        = empty1;
      almostfull_ref   = almostfull1;
      almostempty_ref  = almostempty1;
      underflow_ref    = underflow1;
    endfunction

    function void check_data(input FIFO_transaction F_txn);

      reference_model(F_txn.data_out, F_txn.wr_ack, F_txn.overflow,
                      F_txn.full, F_txn.empty, F_txn.almostfull,
                      F_txn.almostempty, F_txn.underflow);

      // Compare actual vs reference
      if (data_out_ref !== F_txn.data_out) begin
        $error("%t: Mismatch in data_out: Expected = %0d, Actual = %0d",
               $time, data_out_ref, F_txn.data_out);
        error_counter++;
      end
      else begin
        correct_counter++;
      end

    endfunction


    function new();
    endfunction

  endclass
endpackage
```

7. Monitor

```
import shared_pkg::*;
import scoreboard_pkg::*;
import transaction_pkg::*;
import coverage_pkg::*;

module FIFO_monitor (FIFO_interface.moniter intf);

    FIFO_transaction tr;
    FIFO_scoreboard sb;
    FIFO_coverage cov;

    initial begin
        tr = new();
        sb = new();
        cov = new();

        forever begin
            @(negedge intf.clk);

            // Sample interface into transaction
            tr.data_in          = intf.data_in;
            tr.rst_n            = intf.rst_n;
            tr.wr_en            = intf.wr_en;
            tr.rd_en            = intf.rd_en;
            tr.data_out         = intf.data_out;
            tr.wr_ack           = intf.wr_ack;
            tr.overflow         = intf.overflow;
            tr.full             = intf.full;
            tr.empty            = intf.empty;
            tr.almostfull       = intf.almostfull;
            tr.almostempty      = intf.almostempty;
            tr.underflow        = intf.underflow;

            fork
                begin
                cov.sample_data(tr);
                end
                begin
                sb.check_data(tr);
                end
            join

            if (test_finished) begin
                $display("************ Simulation Summary ************");
                $display("Errors: %0d, Correct: %0d", error_counter, correct_counter);
                $stop;
            end
        end
    end

endmodule
```

8. Testbench

```
File  Edit  Format  View  Help
import shared_pkg::*;
import transaction_pkg::*;

module FIFO_testbench (FIFO_interface.tb intf);
    FIFO_transaction tr = new();

    initial begin

        reset;

        repeat(1000) begin
                assert(tr.randomize());
                intf.data_in = tr.data_in;
                intf.rst_n   = tr.rst_n;
                intf.wr_en   = tr.wr_en;
                intf.rd_en   = tr.rd_en;
                @(negedge intf.clk);
        end

        reset;

        // Finalize test
        test_finished = 1;
        @(negedge intf.clk);
    end

    // Reset task
    task reset;
        intf.rst_n = 0;
        tr.rst_n = intf.rst_n;
        repeat(5) @(negedge intf.clk);
        intf.rst_n = 1;
        tr.rst_n = intf.rst_n;
    endtask
endmodule
```

9. Top Module

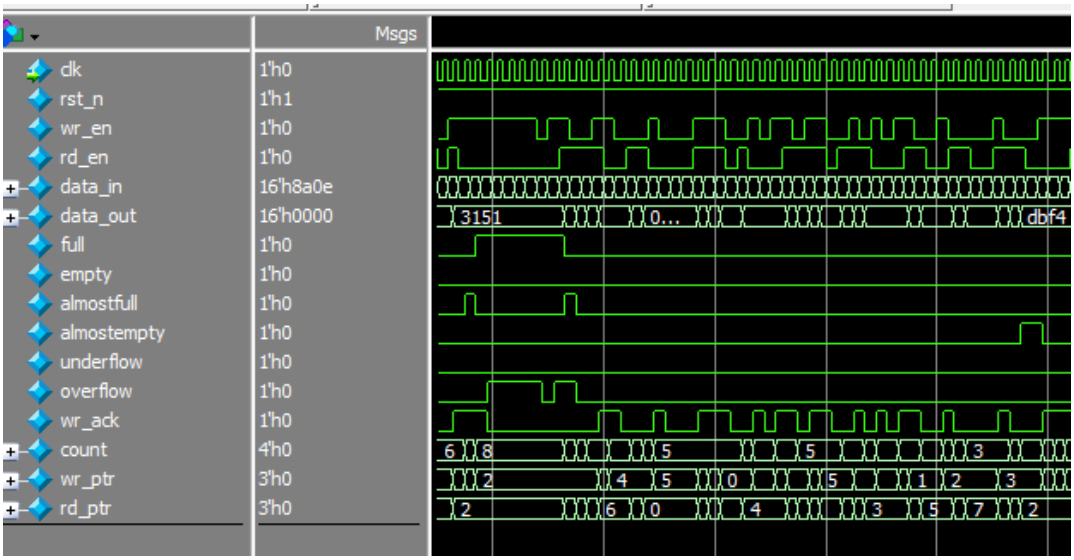
```
module  FIFO_top ();
    bit clk;
    initial begin
        forever #1 clk = ~clk;
    end

    FIFO_interface intf(clk);

    FIFO dut(intf);
    FIFO_testbench tb(intf);
    FIFO_monitor mon(intf);
endmodule
```

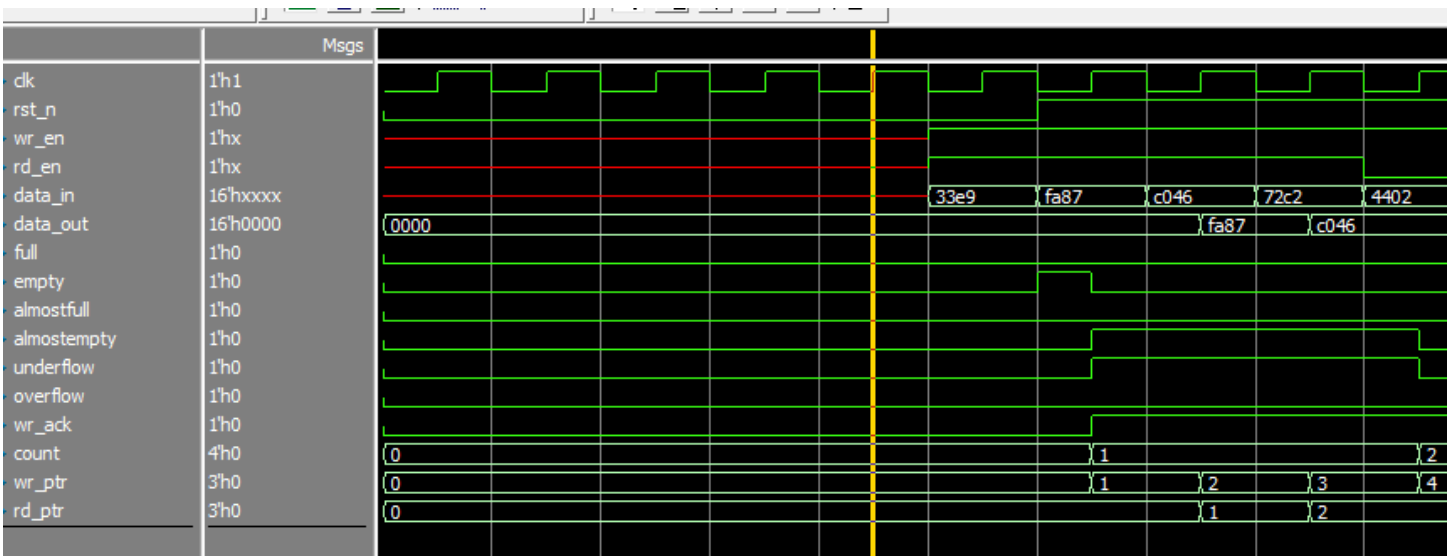
10. FIFO_files and Do file

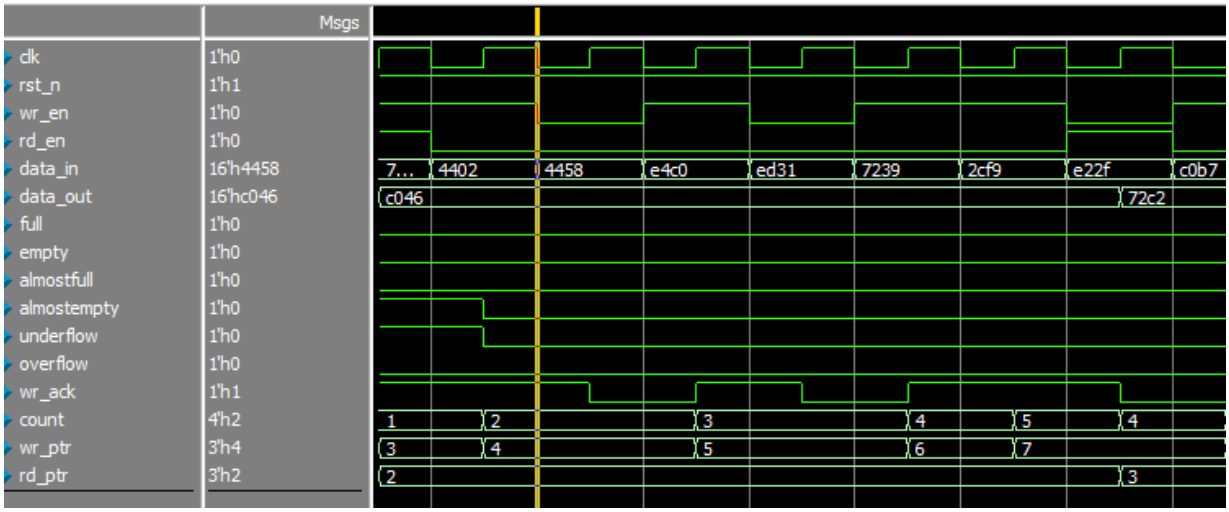
```
shared_pkg.sv
FIFO_interface.sv
FIFO.sv
FIFO_transaction.sv
FIFO_coverage.sv
FIFO_scoreboard.sv
FIFO_moniter.sv
FIFO_testbench.sv
FIFO_top.sv
```

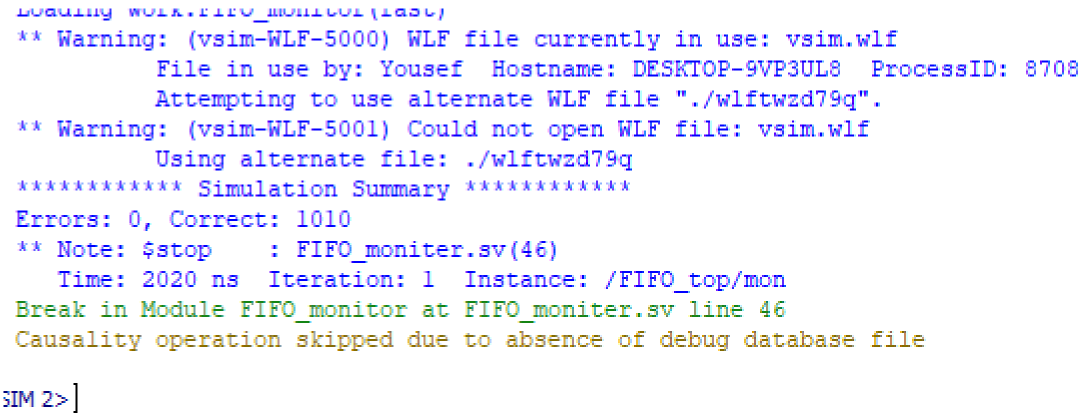
```
vlib work
vlog -f FIFO_files.list
vsim FIFO_top

add wave /FIFO_top/intf/clk
add wave /FIFO_top/intf/rst_n
add wave /FIFO_top/intf/wr_en
add wave /FIFO_top/intf/rd_en
add wave /FIFO_top/intf/data_in
add wave /FIFO_top/intf/data_out
add wave /FIFO_top/intf/full
add wave /FIFO_top/intf/empty
add wave /FIFO_top/intf/almostfull
add wave /FIFO_top/intf/almostempty
add wave /FIFO_top/intf/underflow
add wave /FIFO_top/intf/overflow
add wave /FIFO_top/intf/wr_ack
add wave /FIFO_top/dut/counter
add wave /FIFO_top/dut/wr_ptr
add wave /FIFO_top/dut/rd_ptr
run -all
```

11. QuestaSim snippets



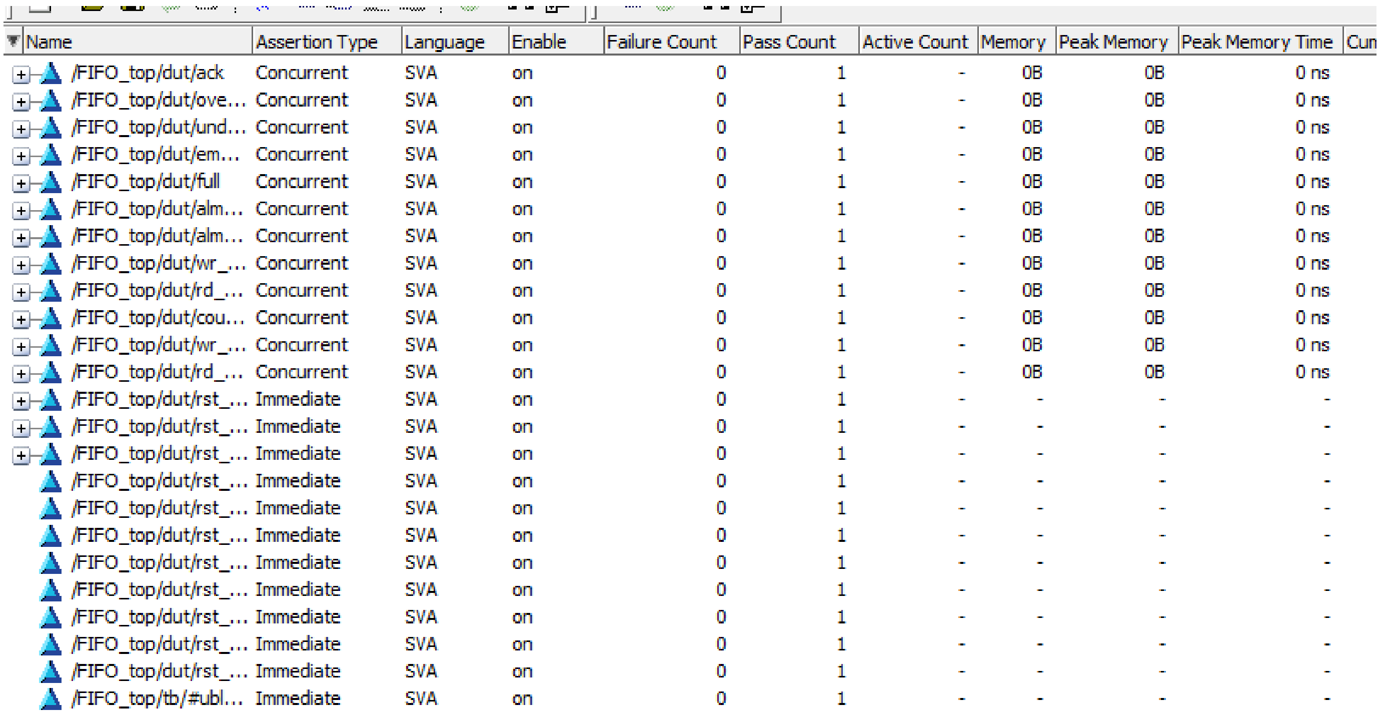






```
Loading work.FIFO_monitor(fast)
** Warning: (vsim-WLF-5000) WLF file currently in use: vsim.wlf
          File in use by: Yousef  Hostname: DESKTOP-9VP3UL8  ProcessID: 8708
          Attempting to use alternate WLF file "./wlftwzd79q".
** Warning: (vsim-WLF-5001) Could not open WLF file: vsim.wlf
          Using alternate file: ./wlftwzd79q
************ Simulation Summary ************
Errors: 0, Correct: 1010
** Note: $stop    : FIFO_moniter.sv(46)
   Time: 2020 ns  Iteration: 1  Instance: /FIFO_top/mon
Break in Module FIFO_monitor at FIFO_moniter.sv line 46
Causality operation skipped due to absence of debug database file

SIM 2>
```

No Assertion errors)



Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cun
/FIFO_top/dut/ack	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/ove…	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/und…	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/em…	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/full	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/alm…	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/alm…	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/wr_…	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/rd_…	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/cou…	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/wr_…	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/rd_…	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	
/FIFO_top/dut/rst_…	Immediate	SVA	on	0	1	-	-	-	-	
/FIFO_top/dut/rst_…	Immediate	SVA	on	0	1	-	-	-	-	
/FIFO_top/dut/rst_…	Immediate	SVA	on	0	1	-	-	-	-	
/FIFO_top/dut/rst_…	Immediate	SVA	on	0	1	-	-	-	-	
/FIFO_top/dut/rst_…	Immediate	SVA	on	0	1	-	-	-	-	
/FIFO_top/dut/rst_…	Immediate	SVA	on	0	1	-	-	-	-	
/FIFO_top/dut/rst_…	Immediate	SVA	on	0	1	-	-	-	-	
/FIFO_top/dut/rst_…	Immediate	SVA	on	0	1	-	-	-	-	
/FIFO_top/dut/rst_…	Immediate	SVA	on	0	1	-	-	-	-	
/FIFO_top/dut/rst_…	Immediate	SVA	on	0	1	-	-	-	-	
/FIFO_top/dut/rst_…	Immediate	SVA	on	0	1	-	-	-	-	
/FIFO_top/tb/#ubl…	Immediate	SVA	on	0	1	-	-	-	-	

Name	Class Type	Coverage	Goal	% of Goal	Status	Included
/coverage_pkg/FIF...		100.00%				
TYPE CVG		100.00%	100	100.00...		✓
CVP CVG::...		100.00%	100	100.00...		✓
CVP CVG::r...		100.00%	100	100.00...		✓
CVP CVG::...		100.00%	100	100.00...		✓
CVP CVG::f...		100.00%	100	100.00...		✓
CVP CVG::...		100.00%	100	100.00...		✓
CVP CVG::...		100.00%	100	100.00...		✓
CVP CVG::...		100.00%	100	100.00...		✓
CVP CVG::...		100.00%	100	100.00...		✓
CVP CVG::...		100.00%	100	100.00...		✓
CROSS CV...		100.00%	100	100.00...		✓
CROSS CV...		100.00%	100	100.00...		✓
CROSS CV...		100.00%	100	100.00...		✓
CROSS CV...		100.00%	100	100.00...		✓
CROSS CV...		100.00%	100	100.00...		✓
CROSS CV...		100.00%	100	100.00...		✓
CROSS CV...		100.00%	100	100.00...		✓