Synchronous FIFO

1. Verification Plan:

- Create interface with 3 port modes (TB , DUT , MONITOR).
- Adjust the design to take an interface and change the file extension to sv and add the assertions.
- Create package named shared_pkg .
- Create 3 classes (FIFO_transaction, FIFO_scoreboard, FIFO_coverage) with their internal requirements every file have only one class.

2. verification requirement document

Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
FIFO_1	assert the reset	N/A	Automatic genration of a bits	Data Output Checked against refrence model made in Scoreboard Package - Others Signals are checked with assertioms inside the dut module
FIFO_2	Randomizations with Constriants	1. Assert reset less often 2. Constraint the write enable to be high with distribution of the value 70 and to be low with 30 3.Constraint the read enable to be high with distribution of the value 30 and to be low with 70	Sampling the cover points.	Data Output Checked against refrence model made in Scoreboard Package - Others Signals are checked with assertioms inside the dut module

Design Code:

```
1  ////////////////////////////////////////////////////////////////////////
2  // Author: Kareem Waseem
3  // Course: Digital Verification using SV & UVM
4  //
5  // Description: FIFO Design
6  //
7  ////////////////////////////////////////////////////////////////////////
8  module FIFO(FIFO_intf.dut intf);
9  localparam  FIFO_WIDTH = intf.FIFO_WIDTH;
10 localparam FIFO_DEPTH = intf.FIFO_DEPTH;
11
12 //! I comment all the inputs and ouputs and definded them to logic and assign them to input and output come from interface
13 // input [FIFO_WIDTH-1:0] data_in;
14 // input clk, rst_n, wr_en, rd_en;
15 // output reg [FIFO_WIDTH-1:0] data_out;
16 // output reg wr_ack, overflow;
17 // output full, empty, almostfull, almostempty, underflow;
18
19 //input
20 logic [FIFO_WIDTH-1:0] data_in;
21 logic clk, rst_n, wr_en, rd_en;
22 logic [FIFO_WIDTH-1:0] data_out;
23 logic wr_ack, overflow;
24 logic full, empty, almostfull, almostempty, underflow;
25
26   //!assign inputs to interface!
27   assign data_in=intf.data_in;
28   assign clk=intf.clk;
29   assign rst_n=intf.rst_n;
30   assign wr_en=intf.wr_en;
31   assign rd_en=intf.rd_en;
32
33   //!assign outputs to interface!
34   assign intf.data_out=data_out;
35   assign intf.wr_ack=wr_ack;
36   assign intf.overflow=overflow;
37   assign intf.full=full;
38   assign intf.empty=empty;
39   assign intf.almostfull=almostfull;
40   assign intf.almostempty=almostempty;
41   assign intf.underflow=underflow;
42
43   localparam max_fifo_addr = $clog2(FIFO_DEPTH);
44
45   reg [FIFO_WIDTH-1:0] mem [FIFO_DEPTH-1:0];
46
47   reg [max_fifo_addr-1:0] wr_ptr, rd_ptr;
48   reg [max_fifo_addr:0] count;
```

There is two bugs here (I comment them)!

```
always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        wr_ptr <= 0;
    end
    else if (wr_en && count < FIFO_DEPTH) begin
        mem[wr_ptr] <= data_in;
        //wr_ack <= 1; //? error here
        wr_ptr <= wr_ptr + 1;
    end
    //? error here
    /*else begin
        wr_ack <= 0;
        if (full & wr_en)
            overflow <= 1;
        else
            overflow <= 0;
    end
    */
end
```

```
70  always @(posedge clk or negedge rst_n) begin
71      if (!rst_n) begin
72          rd_ptr <= 0;
73      end
74      else if (rd_en && count != 0) begin
75          data_out <= mem[rd_ptr];
76          rd_ptr <= rd_ptr + 1;
77      end
78  end
79
80  always @(posedge clk or negedge rst_n) begin
81      if (!rst_n) begin
82          count <= 0;
83      end
84      else begin
85          if  ( ({wr_en, rd_en} == 2'b10) && !full)
86              count <= count + 1;
87          else if ( ({wr_en, rd_en} == 2'b01) && !empty)
88              count <= count - 1;
89      end
90  end
```

One bug here and I correct all the bugs!

```
assign full = (count == FIFO_DEPTH)? 1 : 0;
assign empty = (count == 0)? 1 : 0;
assign underflow = (empty && rd_en)? 1 : 0;
//assign almostfull = (count == FIFO_DEPTH-2)? 1 : 0; //? error here must be FIFO_DEPTH-1 not -2.
assign almostempty = (count == 1)? 1 : 0;

//! I correct them  to work
assign wr_ack=((count!=FIFO_DEPTH) && wr_en)?1:0;
assign overflow=(full && wr_en)?1:0;
assign almostfull = (count == FIFO_DEPTH-1)? 1 : 0;
```

Assertions:

```
`ifdef SIM

    property read_count;
        @(posedge clk) disable iff(!rst_n) (rd_en&&!wr_en&&count!=0) |=>($past(count)-1'b1==count);
    endproperty


    property write_count;
        @(posedge clk) disable iff(!rst_n) (!rd_en&&wr_en&&count!=FIFO_DEPTH) |=>($past(count)+1'b1==count)
    endproperty

    property read_write_count;
        @(posedge clk) disable iff(!rst_n) (rd_en&&wr_en&&count!=0&&count!=FIFO_DEPTH) |=>($past(count)==count);
    endproperty

    property read_ptr;
        @(posedge clk) disable iff(!rst_n) (rd_en&&count!=0) |=>($past(rd_ptr)+1'b1==rd_ptr);
    endproperty

    property write_ptr;
        @(posedge clk) disable iff(!rst_n) (wr_en&&count!=FIFO_DEPTH) |=>($past(wr_ptr)+1'b1==wr_ptr);
    endproperty
```

```
    //assertions
    rd_count_assert:assert property (read_count);
    wr_count_assert:assert property (write_count);
    rd_wr_count_assert:assert property (read_write_count);
    rd_ptr_assert:assert property (read_ptr);
    wr_ptr_assert:assert property (write_ptr);

    //cover assertions
    rd_count_cover:cover property (read_count);
    wr_count_cover:cover property (write_count);
    rd_wr_count_cover:cover property (read_write_count);
    rd_ptr_cover:cover property (read_ptr);
    wr_ptr_cover:cover property (write_ptr);
```

```
    always_comb begin
        if(!rst_n)begin
        rst_count_assert:assert final (count==0);
        rst_rd_ptr_assert:assert final (rd_ptr==0);
        rst_wr_ptr_assert:assert final (wr_ptr==0);
        end

        if(count==0)begin
            empty_assert:assert final(empty==1);
        end

        if(count==0 && rd_en)begin
            underflow_assert:assert final(underflow==1);
        end

        if (count==1) begin
            almostempty_assert:assert final(almostempty==1);
        end

        if (count==FIFO_DEPTH-1) begin
            almostfull_assert:assert final(almostfull==1);
        end

        if(count==FIFO_DEPTH)begin
            full_assert:assert final(full==1);
        end

        if(count==FIFO_DEPTH && wr_en)begin
            overflow_assert:assert final(overflow==1);
        end

        if(count!=FIFO_DEPTH && wr_en)begin
            wr_ack_assert:assert final(wr_ack==1);
        end
    end
`endif

endmodule
```

TOP Code:

```
module top();

  bit clk ;

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

  FIFO_intf intf (clk) ;
  FIFO  dut (intf);
  monitor mon (intf) ;
  test tb (intf);

endmodule
```

Testbench Code:

```
1   import shared_pkg ::*;
2   import FIFO_transaction_pkg::*;
3
4   module test (FIFO_intf.tb intf) ;
5      FIFO_transaction trans ;
6
7      initial begin
8         trans = new() ;
9
10        assert_reset;
11
12        repeat(10000) begin
13           assert(trans.randomize());
14           intf.data_in=trans.data_in;
15           intf.wr_en=trans.wr_en;
16           intf.rd_en=trans.rd_en;
17           intf.rst_n=trans.rst_n;
18           @(negedge intf.clk);
19        end
20        test_finished =  1;
21     end
22
23     task assert_reset ;
24        intf.rst_n = 0 ;
25        @(negedge intf.clk) ;
26        intf.rst_n= 1 ;
27     endtask
28
29  endmodule
```

Monitor Code:

```
import FIFO_transaction_pkg::*;
import FIFO_scoreboard_pkg::*;
import cover_pkg::*;
import shared_pkg::*;
module monitor(FIFO_intf.monitor dut_if);

    FIFO_transaction f_txn;
    FIFO_scoreboard f_scoreboard = new();
    FIFO_coverage f_coverage = new();

    initial begin

        f_txn = new();

        forever begin
            f_txn.data_in = dut_if.data_in;
            f_txn.wr_en = dut_if.wr_en;
            f_txn.rd_en = dut_if.rd_en;
            f_txn.rst_n = dut_if.rst_n;
            @(negedge dut_if.clk);
            f_txn.data_out = dut_if.data_out;
            f_txn.full = dut_if.full;
            f_txn.almostfull = dut_if.almostfull;
            f_txn.empty = dut_if.empty;
            f_txn.almostempty = dut_if.almostempty;
            f_txn.overflow = dut_if.overflow;
            f_txn.underflow = dut_if.underflow;
            f_txn.wr_ack = dut_if.wr_ack;
            @(posedge dut_if.clk);
```

```
        fork
            begin
                f_coverage.sample_data(f_txn);
            end

            begin

                f_scoreboard.check_data(f_txn);
            end
        join

        if (shared_pkg::test_finished) begin
            $display("Test finished. Correct: %0d, Errors: %0d", correct_count, error_count);
            $stop;
        end

    end
  end
ndmodule
```

FIFO interface Code:

```
1   interface FIFO_intf(input bit clk);
2
3   parameter FIFO_WIDTH = 16;
4   parameter FIFO_DEPTH = 8;
5
6   logic [FIFO_WIDTH-1:0] data_in;
7   logic  rst_n, wr_en, rd_en;
8   logic [FIFO_WIDTH-1:0] data_out;
9   logic wr_ack, overflow, full, empty, almostfull, almostempty, underflow;
10
11  // DUT Modport
12  modport dut (
13      input data_in,clk, rst_n, wr_en, rd_en,
14      output data_out,wr_ack, overflow,full, empty, almostfull, almostempty, underflow
15  );
16
17  //Testbench Modport
18  modport tb (
19      input data_out,wr_ack, overflow,full, empty, almostfull, almostempty, underflow,clk,
20      output data_in,rst_n, wr_en, rd_en
21  );
22
23  //Monitor Modport
24  modport monitor (
25      input data_in,clk, rst_n, wr_en, rd_en,
26      data_out,wr_ack, overflow,full, empty, almostfull, almostempty, underflow
27  );
28
29  endinterface
```

Coverage Package:

```
package cover_pkg;
    import FIFO_transaction_pkg::*;

    class FIFO_coverage;

        FIFO_transaction F_cvg_txn;

        covergroup cg;
            wr_en_point:coverpoint F_cvg_txn.wr_en;
            rd_en_point:coverpoint F_cvg_txn.rd_en;
            wr_ack_point:coverpoint F_cvg_txn.wr_ack;
            overflow_point:coverpoint F_cvg_txn.overflow;
            full_point:coverpoint F_cvg_txn.full;
            empty_point:coverpoint F_cvg_txn.empty;
            almostfull_point:coverpoint F_cvg_txn.almostfull;
            almostempty_point:coverpoint F_cvg_txn.almostempty;
            underflow_point:coverpoint F_cvg_txn.underflow;

            wr_ack_point_cross:cross wr_en_point,rd_en_point,wr_ack_point;
            overflow_point_cross:cross wr_en_point,rd_en_point,overflow_point;
            full_point_cross:cross wr_en_point,rd_en_point,full_point;
            empty_point_cross:cross wr_en_point,rd_en_point,empty_point;
            almostfull_point_cross:cross wr_en_point,rd_en_point,almostfull_point;
            almostempty_point_cross:cross wr_en_point,rd_en_point,almostempty_point;
            underflow_point_cross:cross wr_en_point,rd_en_point,underflow_point;
        endgroup

        function new;
            cg=new();
            cg.start();//? TODO
        endfunction

        function void sample_data(FIFO_transaction F_txn);
            F_cvg_txn=F_txn;
            cg.sample();//? TODO
        endfunction

    endclass
endpackage
```

Scoreboard Package:

```
package FIFO_scoreboard_pkg;
    import FIFO_transaction_pkg::*;
    import shared_pkg::*;


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

        localparam max_fifo_addr=$clog2(FIFO_DEPTH);
        bit [FIFO_WIDTH-1:0] data_out_ref;
        bit full_ref, empty_ref, almost_full_ref, almost_empty_ref, overflow_ref, underflow_ref;

        bit [FIFO_WIDTH]fifo_mem[FIFO_DEPTH];
        bit [max_fifo_addr]read_ptr = 0;
        bit [max_fifo_addr] write_ptr = 0;
        bit [max_fifo_addr+1] count;

        function void reference_model(FIFO_transaction trans);

        if (!trans.rst_n) begin
            read_ptr = 0;
            write_ptr = 0;
            count = 0;
        end
        else begin
```

```
            if (trans.rd_en && count!=0) begin
                data_out_ref = fifo_mem[read_ptr];
                read_ptr++;
            end
            if (trans.wr_en && count != FIFO_DEPTH) begin
                fifo_mem[write_ptr] = trans.data_in;
                write_ptr++;
            end
            if(trans.wr_en && !trans.rd_en && count != FIFO_DEPTH)begin
                count++;
            end
            if(!trans.wr_en && trans.rd_en && count != 0) begin
                count--;
            end
        end
        endfunction

        function void check_data(FIFO_transaction trans);
            reference_model(trans);
            if (trans.data_out != data_out_ref) begin
                $display("Error");
                shared_pkg::error_count++;
            end else begin
                $display("Correct data.");
                shared_pkg::correct_count++;
            end
        endfunction
    endclass
endpackage
```

Transactions Package:

```
package FIFO_transaction_pkg ;
    parameter FIFO_WIDTH = 16;
    parameter FIFO_DEPTH = 8;
    int WR_EN_ON_DIST=70 ;
    int RD_EN_ON_DIST=30 ;

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

            constraint rst_con {
            rst_n dist {0:=2 , 1:=98 } ;
            }

            constraint wr_en_con {
            wr_en dist {1:=WR_EN_ON_DIST , 0:=100-WR_EN_ON_DIST } ;
            }

            constraint rd_en_con {
            rd_en dist {1:=RD_EN_ON_DIST , 0:=100-RD_EN_ON_DIST } ;
            }

    endclass
endpackage
```

Shared Package:

```
package shared_pkg;
    int error_count = 0;
    int correct_count = 0;
    bit test_finished;
endpackage

```

Do File:

```
vlog -f src_files.list -mfcu +define+SIM +cover
vsim -voptargs=+acc work.top -cover
add wave *
coverage save fifocoveragereport.ucdb -onexit -du work.top
run -all
coverage report -detail -cvg -directive -comments -output fcover_report.txt /cover_pkg/FIFO_coverage/cg
quit -sim
vcover report fifocoveragereport.ucdb -details -annotate -all -output fifocoveragereport.txt

```

```
packages/shared_pkg.sv
packages/FIFO_transaction.sv
packages/cover_pkg.sv
packages/FIFO_scoreboard_pkg.sv
modules/*
```

QuestaSim Snippets:

```
# Test finished. Correct: 10001, Errors: 0
# Correct data.
# Correct data.
# Correct data.
# Correct data.
# Correct data.
# Correct data.
# Correct data.
# Correct data.
# Correct data.
# Correct data.
# Correct data.
# Correct data.
```

Cover Report:

```
================================================================================
=== Instance: /\top /dut
=== Design Unit: work.FIFO
================================================================================

Assertion Coverage:
    Assertions                      15        15         0   100.00%
--------------------------------------------------------------------

Name                 File(Line)                    Failure      Pass
                                                   Count        Count
```

```
Directive Coverage:
    Directives                       5         5         0   100.00%

DIRECTIVE COVERAGE:
-------------------------------------------------------------------------------------------
Name                                 Design Design   Lang File(Line)      Hits Status
                                     Unit   UnitType
-------------------------------------------------------------------------------------------
/\top /dut/rd_count_cover            FIFO   Verilog  SVA  modules/FIFO.sv(135)
                                                                  812 Covered
/\top /dut/wr_count_cover            FIFO   Verilog  SVA  modules/FIFO.sv(136)
                                                                 2009 Covered
/\top /dut/rd_wr_count_cover         FIFO   Verilog  SVA  modules/FIFO.sv(137)
                                                                  802 Covered
/\top /dut/rd_ptr_cover              FIFO   Verilog  SVA  modules/FIFO.sv(138)
                                                                 2790 Covered
/\top /dut/wr_ptr_cover              FIFO   Verilog  SVA  modules/FIFO.sv(139)
                                                                 2901 Covered
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      23        23         0   100.00%

================================Statement Details================================
```

```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                        106       106         0   100.00%

--------------------------------Toggle Details--------------------------------
```

Cover Group: it’s not 100 because we can’t write when full is high and we can’t read when empty is high and scenarios like these.

```
Coverage Report by instance with details

================================================================================
=== Instance: /cover_pkg
=== Design Unit: work.cover_pkg
================================================================================

Covergroup Coverage:
    Covergroups                     1        na        na    94.53%
        Coverpoints/Crosses        16        na        na       na
            Covergroup Bins        74        67         7    90.54%
-------------------------------------------------------------------------------------------
Covergroup                                          Metric       Goal       Bins    Status

-------------------------------------------------------------------------------------------
```