Digital Verification Using SystemVerilog and UVM

Project 1

Synchronous FIFO SV Verification

Name	Sohaila Ahmed Mohamed
Project 1	Synchronous FIFO

Project

Synchronous FIFO

Read the important notes and the requirements carefully at the end of the document.

Parameters

- FIFO_WIDTH: DATA in/out and memory word width (default: 16)
- FIFO_DEPTH: Memory depth (default: 8)

Ports

Port	Direction	Function
data_in	Input	Write Data: The input data bus used when writing the FIFO.
wr_en		Write Enable: If the FIFO is not full, asserting this signal causes data (on data_in) to be written into the FIFO
rd_en		Read Enable: If the FIFO is not empty, asserting this signal causes data (on data_out) to be read from the FIFO
clk		Clock signal
rst_n		Active low asynchronous reset
data_out	Output	Read Data: The sequential output data bus used when reading from the FIFO.
full		Full Flag: When asserted, this combinational output signal indicates that the FIFO is full. Write requests are ignored when the FIFO is full, initiating a write when the FIFO is full is not destructive to the contents of the FIFO.
almostfull		Almost Full: When asserted, this combinational output signal indicates that only one more write can be performed before the FIFO is full.
empty		Empty Flag: When asserted, this combinational output signal indicates that the FIFO is empty. Read requests are ignored when the FIFO is empty, initiating a read while empty is not destructive to the FIFO.
almostempty		Almost Empty: When asserted, this output combinational signal indicates that only one more read can be performed before the FIFO goes to empty.
overflow		Overflow: This sequential output signal indicates that a write request (wr_en) was rejected because the FIFO is full. Overflowing the FIFO is not destructive to the contents of the FIFO.
underflow		Underflow: This sequential output signal Indicates that the read request (rd_en) was rejected because the FIFO is empty. Under flowing the FIFO is not destructive to the FIFO.
wr_ack		Write Acknowledge: This sequential output signal indicates that a write request (wr_en) has succeeded.

A) Bugs Detection

1. The design file before

```
////////////////////////////////////////////////////////////////////////////////
// Author: Kareem Waseem
// Course: Digital Verification using SV & UVM
//
// Description: FIFO Design
//
////////////////////////////////////////////////////////////////////////////////
module FIFO(data_in, wr_en, rd_en, clk, rst_n, full, empty, almostfull, almostempty, wr_ack, overflow, underflow, data_out);
parameter FIFO_WIDTH = 16;
parameter FIFO_DEPTH = 8;
input [FIFO_WIDTH-1:0] data_in;
input clk, rst_n, wr_en, rd_en;
output reg [FIFO_WIDTH-1:0] data_out;
output reg wr_ack, overflow;
output full, empty, almostfull, almostempty, underflow;

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 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;
        wr_ptr <= wr_ptr + 1;
    end
    else begin
        wr_ack <= 0;
        if (full & wr_en)
            overflow <= 1;
        else
            overflow <= 0;
    end
end

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;
    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;
    end
end

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;
assign almostempty = (count == 1)? 1 : 0;

endmodule
```

2. The design file after

```
8   module FIFO(fifo_if.DUT fif);
9   localparam max_fifo_addr = $clog2(fif.FIFO_DEPTH);
10
11  reg [fif.FIFO_WIDTH-1:0] mem [fif.FIFO_DEPTH-1:0];
12
13  reg [max_fifo_addr-1:0] wr_ptr, rd_ptr;
14  reg [max_fifo_addr:0] count;
15
16  always @(posedge fif.clk or negedge fif.rst_n) begin
17      if (!fif.rst_n) begin
18          wr_ptr <= 0;
19          fif.overflow <= 0; //update_2
20          fif.wr_ack <= 0; //update_2
21      end
22      else if (fif.wr_en && count < fif.FIFO_DEPTH) begin
23          mem[wr_ptr] <= fif.data_in;
24          fif.wr_ack <= 1;
25          wr_ptr <= wr_ptr + 1;
26      end
27      else begin
28          fif.wr_ack <= 0;
29          if (fif.full & fif.wr_en)
30              fif.overflow <= 1;
31          else
32              fif.overflow <= 0;
33      end
34  end
35
36  always @(posedge fif.clk or negedge fif.rst_n) begin
37      if (!fif.rst_n) begin
38          rd_ptr <= 0;
39          //fif.data_out <= 0; //update_3
40          fif.underflow <= 0; //update_3
41      end
42      else if (fif.rd_en && count != 0) begin
43          fif.data_out <= mem[rd_ptr];
44          rd_ptr <= rd_ptr + 1;
45      end
46      else begin //update_4
47          if (fif.empty & fif.rd_en)
48              fif.underflow <= 1;
49          else
50              fif.underflow <= 0;
51      end
52  end
53
54  always @(posedge fif.clk or negedge fif.rst_n) begin
55      if (!fif.rst_n) begin
56          count <= 0;
57      end
58      else begin
59          if  ( ({fif.wr_en, fif.rd_en} == 2'b10) && !fif.full)
60              count <= count + 1;
61          else if ( ({fif.wr_en, fif.rd_en} == 2'b01) && !fif.empty)
62              count <= count - 1;
63          else if ( ({fif.wr_en, fif.rd_en} == 2'b11) ) begin //update_5
64              if (fif.full)
65                  count <= count - 1;
66              else if (fif.empty)
67                  count <= count + 1;
68          end
69      end
70  end
71
72
73
74  assign fif.full = (count == fif.FIFO_DEPTH)? 1 : 0;
75  assign fif.empty = (count == 0)? 1 : 0;
76  //assign fif.underflow = (fif.empty && fif.rd_en)? 1 : 0; //incorrect implementation
77  assign fif.almostfull = (count == fif.FIFO_DEPTH-1)? 1 : 0; //update_1
78  assign fif.almostempty = (count == 1)? 1 : 0;
79
```

```
`ifdef SIM
    property reset_behavior;
        @(posedge fif.clk) (!fif.rst_n) |=> (!wr_ptr && !rd_ptr && !count);
    endproperty

    property write_acknowledge;
        @(posedge fif.clk) disable iff(!fif.rst_n) (fif.wr_en && !fif.full) |=> (fif.wr_ack);
    endproperty

    property  overflow_detection;
        @(posedge fif.clk) disable iff(!fif.rst_n) (fif.wr_en && fif.full) |=> (fif.overflow);
    endproperty

    property  underflow_detection;
        @(posedge fif.clk) disable iff(!fif.rst_n) (fif.rd_en && fif.empty) |=> (fif.underflow);
    endproperty

    property  empty_flag_assrt;
        @(posedge fif.clk) disable iff(!fif.rst_n) (!count) |-> (fif.empty);
    endproperty

    property  full_flag_assrt;
        @(posedge fif.clk) disable iff(!fif.rst_n) (count == fif.FIFO_DEPTH) |-> (fif.full);
    endproperty

    property  almost_full_cond;
        @(posedge fif.clk) disable iff(!fif.rst_n) (count == fif.FIFO_DEPTH-1) |-> (fif.almostfull);
    endproperty

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

    property  wr_ptr_wrap;
        @(posedge fif.clk) disable iff(!fif.rst_n) (fif.wr_en && (wr_ptr == fif.FIFO_DEPTH-1) && !fif.full) |=> (!wr_ptr);
    endproperty

    property  rd_ptr_wrap;
        @(posedge fif.clk) disable iff(!fif.rst_n) (fif.rd_en && (rd_ptr == fif.FIFO_DEPTH-1)) |=> (!rd_ptr);
    endproperty

    property  wr_ptr_threshold;
        @(posedge fif.clk) disable iff(!fif.rst_n) wr_ptr < fif.FIFO_DEPTH;
    endproperty

    property  rd_ptr_threshold;
        @(posedge fif.clk) disable iff(!fif.rst_n) rd_ptr < fif.FIFO_DEPTH;
    endproperty

    assert property (reset_behavior);
    assert property (write_acknowledge);
    assert property (overflow_detection);
    assert property (underflow_detection);
    assert property (empty_flag_assrt);
    assert property (full_flag_assrt);
    assert property (almost_full_cond);
    assert property (almost_empty_cond);
    assert property (wr_ptr_wrap);
    assert property (rd_ptr_wrap);
    assert property (wr_ptr_threshold);
    assert property (rd_ptr_threshold);

    cover property (reset_behavior);
    cover property (write_acknowledge);
    cover property (overflow_detection);
    cover property (underflow_detection);
    cover property (empty_flag_assrt);
    cover property (full_flag_assrt);
    cover property (almost_full_cond);
    cover property (almost_empty_cond);
    cover property (wr_ptr_wrap);
    cover property (rd_ptr_wrap);
    cover property (wr_ptr_threshold);
    cover property (rd_ptr_threshold);
`endif

endmodule
```

3. Bugs Report

a) Outputs such as overflow, underflow, and wr_ack were not reset with the asynchronous reset.

b) There was a typo in the almostfull threshold (was supposed to be -1, but instead it was -2).

c) The underflow implementation was not correct (it should be the same as overflow, but with the rd_en and empty).

d) In the count always block, the handle of the simultaneous read and write was missing

B) SV Test Files

1. Scoreboard

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

    class FIFO_scoreboard;
        parameter FIFO_WIDTH = 16;
        parameter FIFO_DEPTH = 8;

        logic [FIFO_WIDTH-1:0] data_out_ref;

        //we will use a queue for testing
        logic [FIFO_WIDTH-1:0] test_queue[$];

        function void check_data(FIFO_transaction tr);
            reference_model(tr);

            if (data_out_ref !== tr.data_out) begin
                error_count = error_count + 1;
                $display("Error at time %0t",$time);
            end
            else begin
                correct_count = correct_count + 1;
                $display("Correct implenmentation");
            end
        endfunction : check_data

        function void reference_model(FIFO_transaction tr);
            if(!tr.rst_n) begin
                test_queue.delete();
            end
            else begin
                if (tr.wr_en && tr.rd_en) begin
                    if(test_queue.size() == 0)
                        test_queue.push_back(tr.data_in);
                    else if (test_queue.size() == FIFO_DEPTH)
                        data_out_ref = test_queue.pop_front();
                    else begin
                        data_out_ref = test_queue.pop_front();
                        test_queue.push_back(tr.data_in);
                    end
                end

                else if (tr.wr_en && !tr.rd_en) begin
                    if (test_queue.size() != FIFO_DEPTH)
                        test_queue.push_back(tr.data_in);
                end

                else if (!tr.wr_en && tr.rd_en) begin
                        if (test_queue.size() != 0)
                            data_out_ref = test_queue.pop_front();
                end

            end

        endfunction : reference_model
    endclass : FIFO_scoreboard


endpackage : FIFO_scoreboard_pkg
```

2. Transaction

```
package FIFO_transaction_pkg;
    parameter FIFO_WIDTH = 16;
    parameter FIFO_DEPTH = 8;

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

        int RD_EN_ON_DIST,WR_EN_ON_DIST;

        function new(int RD_EN_ON_DIST = 30,int WR_EN_ON_DIST = 70);
            this.RD_EN_ON_DIST = RD_EN_ON_DIST;
            this.WR_EN_ON_DIST = WR_EN_ON_DIST;
            wr_en = 0;
            rd_en = 0;
            rst_n = 1;
            data_in = 0;
        endfunction

        constraint rst_c {
            rst_n dist {1:=97, 0:=3};
        }

        constraint wr_c {
            wr_en dist {1:=WR_EN_ON_DIST, 0:=(100-WR_EN_ON_DIST)};
        }

        constraint rd_c {
            rd_en dist {1:=RD_EN_ON_DIST, 0:=(100-RD_EN_ON_DIST)};
        }
    endclass : FIFO_transaction

endpackage : FIFO_transaction_pkg
```

3. Coverage

```
package FIFO_coverage_pkg;
    import FIFO_transaction_pkg::*;

    class FIFO_coverage;
        FIFO_transaction F_cvg_txn;

        covergroup covgrp;
            wr_cp: coverpoint F_cvg_txn.wr_en;
            rd_cp: coverpoint F_cvg_txn.rd_en;
            ack_cp: coverpoint F_cvg_txn.wr_ack;
            over_cp: coverpoint F_cvg_txn.overflow;
            full_cp: coverpoint F_cvg_txn.full;
            empty_cp: coverpoint F_cvg_txn.empty;
            almostfull_cp: coverpoint F_cvg_txn.almostfull;
            almostempty_cp: coverpoint F_cvg_txn.almostempty;
            under_cp: coverpoint F_cvg_txn.underflow;

            crs_1: cross wr_cp,rd_cp,ack_cp{
                ignore_bins ack_zero_wr = binsof(wr_cp) intersect {0} && binsof(ack_cp) intersect {1};
            }
            crs_2: cross wr_cp,rd_cp,over_cp{
                ignore_bins ack_zero_wr = binsof(wr_cp) intersect {0} && binsof(over_cp) intersect {1};
            }
            crs_3: cross wr_cp,rd_cp,full_cp{
                ignore_bins ack_zero_wr = binsof(rd_cp) intersect {1} && binsof(full_cp) intersect {1};
            }
            crs_4: cross wr_cp,rd_cp,empty_cp;
            crs_5: cross wr_cp,rd_cp,almostfull_cp;
            crs_6: cross wr_cp,rd_cp,almostempty_cp;
            crs_7: cross wr_cp,rd_cp,under_cp{
                ignore_bins ack_zero_wr = binsof(rd_cp) intersect {0} && binsof(under_cp) intersect {1};
            }
        endgroup : covgrp

        function new();
            covgrp = new;
            F_cvg_txn = new;
        endfunction

        function void sample_data(FIFO_transaction F_txn);
            F_cvg_txn = F_txn;
            covgrp.sample();
        endfunction
    endclass : FIFO_coverage

endpackage : FIFO_coverage_pkg
```

- **Comment (Justification for ignored bins)**
  1) wr_ack cross: wr_ack can't be high unless there is a write operation.
  2) overflow cross: overflow can't be high unless there is a write operation taking place when the fifo is full.
  3) full cross: if the fifo is full and there is a read operation the fifo will immediately turn to be almostfull so we can't capture this case.
  4) underflow cross: underflow can't be high unless there is a read operation taking place when the fifo is empty.

4. monitor

```
import cnt_pkg::*;

import FIFO_transaction_pkg::*;
import FIFO_coverage_pkg::*;
import FIFO_scoreboard_pkg::*;
module monitor (fifo_if.MONITOR fif);
    FIFO_transaction tr;
    FIFO_scoreboard sb;
    FIFO_coverage cv;


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

        forever begin
            wait(etrigger.triggered);
            @(negedge fif.clk);
            tr.data_in = fif.data_in;
            tr.wr_en = fif.wr_en;
            tr.rd_en = fif.rd_en;
            tr.rst_n = fif.rst_n;
            tr.full = fif.full;
            tr.empty = fif.empty;
            tr.almostfull = fif.almostfull;
            tr.almostempty = fif.almostempty;
            tr.wr_ack = fif.wr_ack;
            tr.overflow = fif.overflow;
            tr.underflow = fif.underflow;
            tr.data_out = fif.data_out;

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

            if (test_finished) begin
                $display("Simulation Stopped");
                $display("error_count = %d, correct_count = %d",error_count, correct_count);
                $stop;
            end
        end
    end
endmodule : monitor

```

5. Interface

```
interface fifo_if (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 clk, data_in, rst_n, wr_en, rd_en, output data_out,wr_ack,overflow,
        full, empty, almostfull, almostempty, underflow);

    modport TEST (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_out,wr_ack,overflow, full, empty, almostfull, almostempty
        , underflow, data_in, rst_n, wr_en, rd_en);

endinterface : fifo_if

```

6. Top

```
module top ();
    bit clk;

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

    fifo_if fif(clk);
    FIFO DUT(fif);
    fifo_tb TEST(fif);
    monitor MONITOR(fif);

    always_comb begin
        if(!fif.rst_n)
            a_reset_asrt: assert final ((fif.wr_ack == 1'h0) && (fif.empty == 1'h1)
                && (fif.full == 1'h0) && (fif.overflow == 1'h0) && (fif.underflow == 1'h0) && (fif.almostfull == 1'h0)
                && (fif.almostempty == 1'h0));
            a_reset_cov: cover final ((fif.wr_ack == 1'h0) && (fif.empty == 1'h1)
                && (fif.full == 1'h0) && (fif.overflow == 1'h0) && (fif.underflow == 1'h0) && (fif.almostfull == 1'h0)
                && (fif.almostempty == 1'h0));
    end

endmodule : top
```

7. Test bench

```
import FIFO_transaction_pkg::*;
import cnt_pkg::*;
module fifo_tb (fifo_if.TEST fif);
    FIFO_transaction txn;

    initial begin
        fif.data_in = 0;
        fif.wr_en = 0;
        fif.rd_en = 0;
        txn = new();

        //reset test
        fif.rst_n = 0;
        @(negedge fif.clk);
        -> etrigger;

        fif.rst_n = 1;

        //random test
        repeat(1000) begin
            assert(txn.randomize());
            fif.rst_n = txn.rst_n;
            fif.data_in = txn.data_in;
            fif.wr_en = txn.wr_en;
            fif.rd_en = txn.rd_en;
            @(negedge fif.clk);
            -> etrigger;
        end

        //end the test
        test_finished = 1;
    end
endmodule : fifo_tb
```

8. Global counters and flags package

```
package cnt_pkg;
    int test_finished = 0;
    int error_count = 0;
    int correct_count = 0;
    event etrigger;
endpackage : cnt_pkg
```

C) Verification plan

Label	Description	Stimulus Generation	Functional Coverage	Functionality Check
FIFO_1	when reset asserted, outputs, internal counters, and pointers value should be reseted	Directed at the start of the simulation then randomized with constraint to be off 97% of the time	cover all the reset values	immediate and concurrent assertions
FIFO_2	when wr_en is high and the fifo is not full, data_in should get in fifo and wr_ack should be high	randomized with constraint on wr_en to be on 70% of the time	cover all the wr_en and wr_ack values	A checker in the testbench to make sure the output is correct (data_out)+ concurrent assertion
FIFO_3	when rd_en is high and the fifo is not empty data_in should get out of the fifo and assigned to data_out	randomized with constraint on rd_en to be on 30% of the time	cover all the rd_en and data_in values	A checker in the testbench to make sure the output is correct (data_out)+ concurrent assertion
FIFO_4	when wr_en is high and the fifo is full, overflow should be high and no write operations will happen	randomized with constraint on wr_en to be on 70% of the time	cover all the overflow values values	A checker in the testbench to make sure the output is correct (data_out)+ concurrent assertion
FIFO_5	when rd_en is high and the fifo is empty underflow should be high and no write operations will happen	randomized with constraint on rd_en to be on 30% of the time	cover all the underflow values values	A checker in the testbench to make sure the output is correct (data_out)+ concurrent assertion
FIFO_6	when the fifo has only one location to be written, almostfull should be high	randomized	cover all the almostfull values values	A checker in the testbench to make sure the output is correct (data_out)+ concurrent assertion
FIFO_7	when the fifo has only one element, almostempty should be high	randomized	cover all the almostempty values values	A checker in the testbench to make sure the output is correct (data_out)+ concurrent assertion

D) Do file and src list

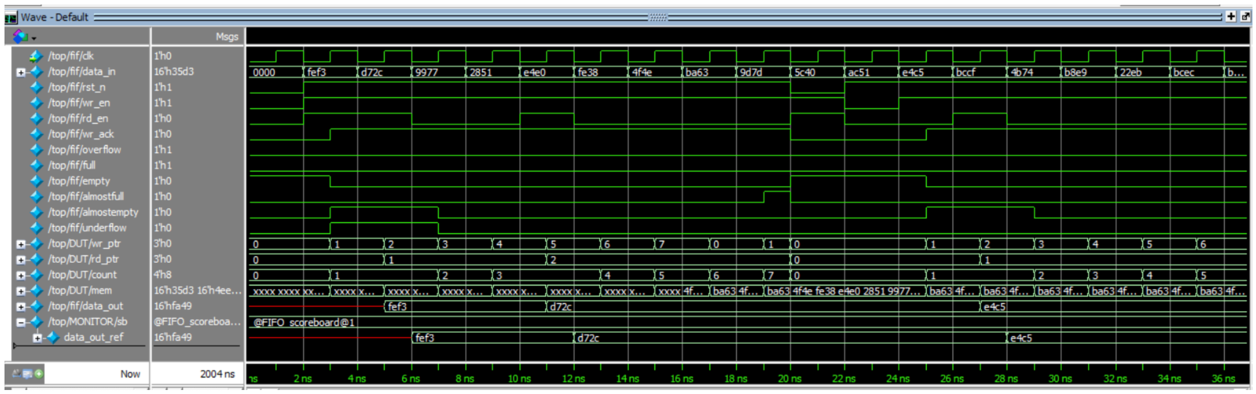
```
vlib work
vlog -f src_files.list +define+SIM +cover -covercells
vsim -voptargs=+acc top -cover
run 0
add wave -position insertpoint sim:/top/fif/*
add wave -position insertpoint sim:/top/DUT/*
add wave -position insertpoint sim:/top/DUT/mem
add wave -position insertpoint sim:/top/MONITOR/sb/test_queue
coverage save top.ucdb -onexit
run -all
```

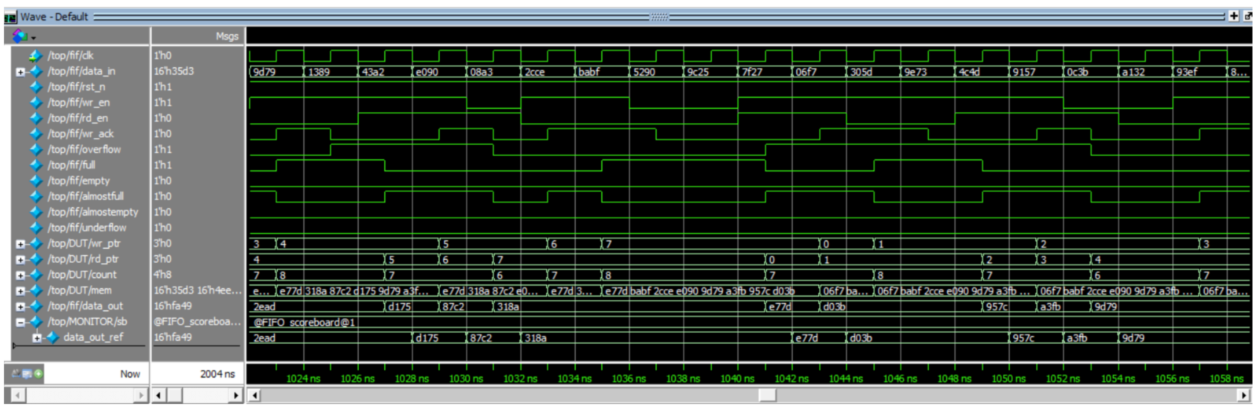
src_files.list - Notepad

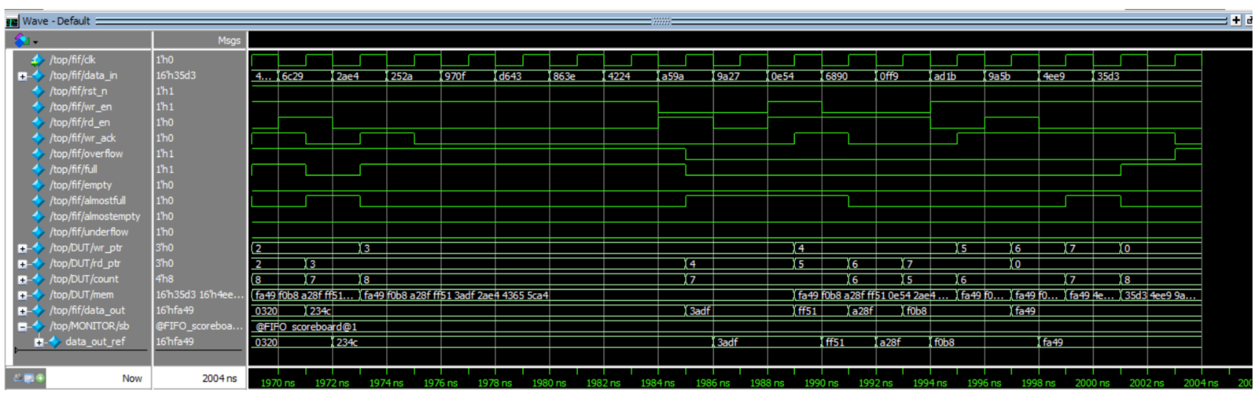
File Edit Format View Help

```
cnt_pkg.sv
fifo_if.sv
FIFO.sv
FIFO_transaction_pkg.sv
FIFO_coverage_pkg.sv
FIFO_scoreboard_pkg.sv
fifo_tb.sv
top.sv
monitor.sv
```

E) QuestaSim snippets







Covergroups

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/FIFO_coverage_pkg/FIFO_coverage		100.00%							
TYPE covgrp		100.00%	100	100.00...				auto(1)	
CVP covgrp::wr_cp		100.00%	100	100.00...					
CVP covgrp::rd_cp		100.00%	100	100.00...					
CVP covgrp::ack_cp		100.00%	100	100.00...					
CVP covgrp::over_cp		100.00%	100	100.00...					
CVP covgrp::full_cp		100.00%	100	100.00...					
CVP covgrp::empty_cp		100.00%	100	100.00...					
CVP covgrp::almostfull_cp		100.00%	100	100.00...					
CVP covgrp::almostempty_cp		100.00%	100	100.00...					
CVP covgrp::under_cp		100.00%	100	100.00...					
CROSS covgrp::crs_1		100.00%	100	100.00...					
CROSS covgrp::crs_2		100.00%	100	100.00...					
CROSS covgrp::crs_3		100.00%	100	100.00...					
CROSS covgrp::crs_4		100.00%	100	100.00...					
CROSS covgrp::crs_5		100.00%	100	100.00...					
CROSS covgrp::crs_6		100.00%	100	100.00...					
CROSS covgrp::crs_7		100.00%	100	100.00...					

Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/top/a_reset_cov	SVA	✓	Off	71	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/cover__rd_ptr_...	SVA	✓	Off	969	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/cover__wr_ptr...	SVA	✓	Off	969	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/cover__rd_ptr_...	SVA	✓	Off	22	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/cover__wr_ptr...	SVA	✓	Off	45	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/cover__almost_...	SVA	✓	Off	75	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/cover__almost_...	SVA	✓	Off	206	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/cover__full_fla...	SVA	✓	Off	303	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/cover__empty_...	SVA	✓	Off	48	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/cover__underfl...	SVA	✓	Off	15	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/cover__overflo...	SVA	✓	Off	199	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/cover__write_a...	SVA	✓	Off	465	1	Unli...	1	100%		✓	0	0	0 ns	0
/top/DUT/cover__reset_b...	SVA	✓	Off	33	1	Unli...	1	100%		✓	0	0	0 ns	0

Assertions

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression	Include
/top/a_reset_asrt	Immediate	SVA	on	0	1	-	-	-	-		off	assert (~fif.wr_ack&fif.empty...	✓
/top/DUT/assert__reset_behavior	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) (~fif...	✓
/top/DUT/assert__write_acknowledge	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disa...	✓
/top/DUT/assert__overflow_detection	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disa...	✓
/top/DUT/assert__underflow_detection	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disa...	✓
/top/DUT/assert__empty_flag_assrt	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disa...	✓
/top/DUT/assert__full_flag_assrt	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disa...	✓
/top/DUT/assert__almost_full_cond	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disa...	✓
/top/DUT/assert__almost_empty_cond	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disa...	✓
/top/DUT/assert__wr_ptr_wrap	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disa...	✓
/top/DUT/assert__rd_ptr_wrap	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disa...	✓
/top/DUT/assert__wr_ptr_threshold	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disa...	✓
/top/DUT/assert__rd_ptr_threshold	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fif.clk) disa...	✓
/top/TEST/#ublk#217929410#22/immed__23	Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓

Transcript

```
# Correct implenmentation
# Correct implenmentation
# Correct implenmentation
# Correct implenmentation
# Correct implenmentation
# Correct implenmentation
# Correct implenmentation
# Correct implenmentation
# Correct implenmentation
# Simulation Stopped
# error_count =           0, correct_count =        1001
# ** Note: $stop    : monitor.sv(45)
#    Time: 2004 ns  Iteration: 1  Instance: /top/MONITOR
# Break in Module monitor at monitor.sv line 45

VSIM(paused)>
```

F) Coverage

1. Toggle, Branch and Statement

```
================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         86        86         0   100.00%

================================Toggle Details================================

Toggle Coverage for instance /top/fif --

                                          Node      1H->0L      0L->1H  "Coverage"
                                          ----------------------------------------
                                   almostempty           1           1      100.00
                                    almostfull           1           1      100.00
                                           clk           1           1      100.00
                                  data_in[15-0]          1           1      100.00
                                 data_out[15-0]          1           1      100.00
                                         empty           1           1      100.00
                                          full           1           1      100.00
                                      overflow           1           1      100.00
                                         rd_en           1           1      100.00
                                         rst_n           1           1      100.00
                                     underflow           1           1      100.00
                                        wr_ack           1           1      100.00
                                         wr_en           1           1      100.00

Total Node Count     =         43
Toggled Node Count   =         43
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (86 of 86 bins)
```

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        27        27         0   100.00%

================================Branch Details================================

Branch Coverage for instance /top/DUT

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File FIFO.sv
------------------------------------IF Branch------------------------------------
    17                                      1035     Count coming in to IF
    17              1                         66        if (!fif.rst_n) begin

    22              1                        475        else if (fif.wr_en && count < fif.FIFO_DEPTH) begin

    27              1                        494        else begin

Branch totals: 3 hits of 3 branches = 100.00%

------------------------------------IF Branch------------------------------------
    29                                       494     Count coming in to IF
    29              1                        211                if (fif.full & fif.wr_en)

    31              1                        283                else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    37                                      1035     Count coming in to IF
    37              1                         66        if (!fif.rst_n) begin

    42              1                        272        else if (fif.rd_en && count != 0) begin

    46              1                        697        else begin //update_4

Branch totals: 3 hits of 3 branches = 100.00%

------------------------------------IF Branch------------------------------------
    47                                       697     Count coming in to IF
```

```
------------------------------------IF Branch------------------------------------
    47                                       697     Count coming in to IF
    47              1                         15                if (fif.empty & fif.rd_en)

    49              1                        682                else

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    55                                       899     Count coming in to IF
    55              1                         66        if (!fif.rst_n) begin

    58              1                        833        else begin

Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    59                                       833     Count coming in to IF
    59              1                        343                if      ( ({fif.wr_en, fif.rd_en} == 2'b10) && !fif.full)

    61              1                         85                else if ( ({fif.wr_en, fif.rd_en} == 2'b01) && !fif.empty)

    63              1                        177                else if ( ({fif.wr_en, fif.rd_en} == 2'b11) ) begin //update_5

                                             228     All False Count
Branch totals: 4 hits of 4 branches = 100.00%

------------------------------------IF Branch------------------------------------
    64                                       177     Count coming in to IF
    64              1                         65                      if (fif.full)

    66              1                         10                      else if (fif.empty)

                                             102     All False Count
Branch totals: 3 hits of 3 branches = 100.00%

------------------------------------IF Branch------------------------------------
    74                                       536     Count coming in to IF
    74              1                        113     assign fif.full = (count == fif.FIFO_DEPTH)? 1 : 0;

    74              2                        423     assign fif.full = (count == fif.FIFO_DEPTH)? 1 : 0;
```

```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      27        27         0   100.00%

================================Statement Details================================

Statement Coverage for instance /top/DUT --

    Line         Item                     Count     Source
    ----         ----                     -----     ------
  File FIFO.sv
    8                                               module FIFO(fifo_if.DUT fif);

    9                                               localparam max_fifo_addr = $clog2(fif.FIFO_DEPTH);

    10

    11                                              reg [fif.FIFO_WIDTH-1:0] mem [fif.FIFO_DEPTH-1:0];

    12

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

    14                                              reg [max_fifo_addr:0] count;

    15

    16              1                        1035     always @(posedge fif.clk or negedge fif.rst_n) begin

    17                                                  if (!fif.rst_n) begin

    18              1                          66               wr_ptr <= 0;

    19              1                          66               fif.overflow <= 0; //update_2

    20              1                          66               fif.wr_ack <= 0; //update_2

    21                                                  end
```

2. Assertions Coverage

```
Assertion Coverage:
    Assertions                      12        12         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------
/top/DUT/assert__rd_ptr_threshold
                     FIFO.sv(140)                       0          1
/top/DUT/assert__wr_ptr_threshold
                     FIFO.sv(139)                       0          1
/top/DUT/assert__rd_ptr_wrap
                     FIFO.sv(138)                       0          1
/top/DUT/assert__wr_ptr_wrap
                     FIFO.sv(137)                       0          1
/top/DUT/assert__almost_empty_cond
                     FIFO.sv(136)                       0          1
/top/DUT/assert__almost_full_cond
                     FIFO.sv(135)                       0          1
/top/DUT/assert__full_flag_assrt
                     FIFO.sv(134)                       0          1
/top/DUT/assert__empty_flag_assrt
                     FIFO.sv(133)                       0          1
/top/DUT/assert__underflow_detection
                     FIFO.sv(132)                       0          1
/top/DUT/assert__overflow_detection
                     FIFO.sv(131)                       0          1
/top/DUT/assert__write_acknowledge
                     FIFO.sv(130)                       0          1
/top/DUT/assert__reset_behavior
                     FIFO.sv(129)                       0          1
Branch Coverage:
```

```
                     FIFO.sv(129)                       0          1
/top/TEST/#ublk#217929410#22/immed__23
                     fifo_tb.sv(23)                     0          1
```

```
Directive Coverage:
    Directives                      12        12         0   100.00%

DIRECTIVE COVERAGE:
-------------------------------------------------------------------------------------------
Name                                  Design Design   Lang File(Line)      Hits Status
                                      Unit   UnitType
-------------------------------------------------------------------------------------------
/top/DUT/cover__rd_ptr_threshold      FIFO   Verilog  SVA  FIFO.sv(153)     969 Covered
/top/DUT/cover__wr_ptr_threshold      FIFO   Verilog  SVA  FIFO.sv(152)     969 Covered
/top/DUT/cover__rd_ptr_wrap           FIFO   Verilog  SVA  FIFO.sv(151)      22 Covered
/top/DUT/cover__wr_ptr_wrap           FIFO   Verilog  SVA  FIFO.sv(150)      45 Covered
/top/DUT/cover__almost_empty_cond     FIFO   Verilog  SVA  FIFO.sv(149)      75 Covered
/top/DUT/cover__almost_full_cond      FIFO   Verilog  SVA  FIFO.sv(148)     206 Covered
/top/DUT/cover__full_flag_assrt       FIFO   Verilog  SVA  FIFO.sv(147)     303 Covered
/top/DUT/cover__empty_flag_assrt      FIFO   Verilog  SVA  FIFO.sv(146)      48 Covered
/top/DUT/cover__underflow_detection   FIFO   Verilog  SVA  FIFO.sv(145)      15 Covered
/top/DUT/cover__overflow_detection    FIFO   Verilog  SVA  FIFO.sv(144)     199 Covered
/top/DUT/cover__write_acknowledge     FIFO   Verilog  SVA  FIFO.sv(143)     465 Covered
/top/DUT/cover__reset_behavior        FIFO   Verilog  SVA  FIFO.sv(142)      33 Covered
Statement Coverage:
```

3. Group Coverage

COVERGROUP COVERAGE:

Covergroup	Metric	Goal	Bins	Status
TYPE /FIFO_coverage_pkg/FIFO_coverage/covgrp	100.00%	100	-	Covered
covered/total bins:	66	66	-	
missing/total bins:	0	66	-	
% Hit:	100.00%	100	-	
Coverpoint wr_cp	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	295	1	-	Covered
bin auto[1]	707	1	-	Covered
Coverpoint rd_cp	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	704	1	-	Covered
bin auto[1]	298	1	-	Covered
Coverpoint ack_cp	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	527	1	-	Covered
bin auto[1]	475	1	-	Covered
Coverpoint over_cp	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	732	1	-	Covered
bin auto[1]	270	1	-	Covered
Coverpoint full_cp	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	683	1	-	Covered
bin auto[1]	319	1	-	Covered
Coverpoint empty_cp	100.00%	100	-	Covered
bin auto[0]	683	1	-	Covered
bin auto[1]	319	1	-	Covered
Coverpoint empty_cp	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	954	1	-	Covered
bin auto[1]	48	1	-	Covered
Coverpoint almostfull_cp	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	791	1	-	Covered
bin auto[1]	211	1	-	Covered
Coverpoint almostempty_cp	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	924	1	-	Covered
bin auto[1]	78	1	-	Covered
Coverpoint under_cp	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	982	1	-	Covered
bin auto[1]	20	1	-	Covered
Cross crs_1	100.00%	100	-	Covered
covered/total bins:	6	6	-	
missing/total bins:	0	6	-	
% Hit:	100.00%	100	-	

Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	132	1	-	Covered
bin <auto[1],auto[0],auto[1]>	343	1	-	Covered
bin <auto[1],auto[1],auto[0]>	71	1	-	Covered
bin <auto[0],auto[1],auto[0]>	95	1	-	Covered
bin <auto[1],auto[0],auto[0]>	161	1	-	Covered
bin <auto[0],auto[0],auto[0]>	200	1	-	Covered
Illegal and Ignore Bins:				
ignore_bin ack_zero_wr	0		-	ZERO
Cross crs_2	100.00%	100	-	Covered
covered/total bins:	6	6	-	
missing/total bins:	0	6	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	79	1	-	Covered
bin <auto[1],auto[0],auto[1]>	191	1	-	Covered
bin <auto[1],auto[1],auto[0]>	124	1	-	Covered
bin <auto[0],auto[1],auto[0]>	95	1	-	Covered
bin <auto[1],auto[0],auto[0]>	313	1	-	Covered
bin <auto[0],auto[0],auto[0]>	200	1	-	Covered
Illegal and Ignore Bins:				
ignore_bin ack_zero_wr	0		-	ZERO
Cross crs_3	100.00%	100	-	Covered
covered/total bins:	6	6	-	
missing/total bins:	0	6	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[0]>	203	1	-	Covered
bin <auto[0],auto[1],auto[0]>	95	1	-	Covered
bin <auto[1],auto[0],auto[1]>	259	1	-	Covered
bin <auto[1],auto[0],auto[0]>	245	1	-	Covered
bin <auto[0],auto[0],auto[1]>	60	1	-	Covered
bin <auto[0],auto[0],auto[0]>	140	1	-	Covered
Illegal and Ignore Bins:				
ignore_bin ack_zero_wr	0		-	ZERO
Cross crs_4	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	6	1	-	Covered
bin <auto[0],auto[1],auto[1]>	15	1	-	Covered

Illegal and Ignore Bins:				
ignore_bin ack_zero_wr	0		-	ZERO
Cross crs_4	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	6	1	-	Covered
bin <auto[0],auto[1],auto[1]>	15	1	-	Covered
bin <auto[1],auto[0],auto[1]>	15	1	-	Covered
bin <auto[0],auto[0],auto[1]>	12	1	-	Covered
bin <auto[1],auto[1],auto[0]>	197	1	-	Covered
bin <auto[0],auto[1],auto[0]>	80	1	-	Covered
bin <auto[1],auto[0],auto[0]>	489	1	-	Covered
bin <auto[0],auto[0],auto[0]>	188	1	-	Covered
Cross crs_5	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	96	1	-	Covered
bin <auto[0],auto[1],auto[1]>	32	1	-	Covered
bin <auto[1],auto[0],auto[1]>	41	1	-	Covered
bin <auto[0],auto[0],auto[1]>	42	1	-	Covered
bin <auto[1],auto[1],auto[0]>	107	1	-	Covered
bin <auto[0],auto[1],auto[0]>	63	1	-	Covered
bin <auto[1],auto[0],auto[0]>	463	1	-	Covered
bin <auto[0],auto[0],auto[0]>	158	1	-	Covered
Cross crs_6	100.00%	100	-	Covered
covered/total bins:	8	8	-	
missing/total bins:	0	8	-	
% Hit:	100.00%	100	-	
Auto, Default and User Defined Bins:				
bin <auto[1],auto[1],auto[1]>	34	1	-	Covered

```
        Auto, Default and User Defined Bins:
            bin <auto[1],auto[1],auto[1]>                96          1          -    Covered
            bin <auto[0],auto[1],auto[1]>                32          1          -    Covered
            bin <auto[1],auto[0],auto[1]>                41          1          -    Covered
            bin <auto[0],auto[0],auto[1]>                42          1          -    Covered
            bin <auto[1],auto[1],auto[0]>               107          1          -    Covered
            bin <auto[0],auto[1],auto[0]>                63          1          -    Covered
            bin <auto[1],auto[0],auto[0]>               463          1          -    Covered
            bin <auto[0],auto[0],auto[0]>               158          1          -    Covered
    Cross crs_6                                      100.00%        100          -    Covered
        covered/total bins:                                8          8          -
        missing/total bins:                                0          8          -
        % Hit:                                       100.00%        100          -
        Auto, Default and User Defined Bins:
            bin <auto[1],auto[1],auto[1]>                34          1          -    Covered
            bin <auto[0],auto[1],auto[1]>                 2          1          -    Covered
            bin <auto[1],auto[0],auto[1]>                28          1          -    Covered
            bin <auto[0],auto[0],auto[1]>                14          1          -    Covered
            bin <auto[1],auto[1],auto[0]>               169          1          -    Covered
            bin <auto[0],auto[1],auto[0]>                93          1          -    Covered
            bin <auto[1],auto[0],auto[0]>               476          1          -    Covered
            bin <auto[0],auto[0],auto[0]>               186          1          -    Covered
    Cross crs_7                                      100.00%        100          -    Covered
        covered/total bins:                                6          6          -
        missing/total bins:                                0          6          -
        % Hit:                                       100.00%        100          -
        Auto, Default and User Defined Bins:
            bin <auto[1],auto[1],auto[1]>                14          1          -    Covered
            bin <auto[1],auto[1],auto[0]>               189          1          -    Covered
            bin <auto[0],auto[1],auto[1]>                 6          1          -    Covered
            bin <auto[0],auto[1],auto[0]>                89          1          -    Covered
            bin <auto[1],auto[0],auto[0]>               504          1          -    Covered
            bin <auto[0],auto[0],auto[0]>               200          1          -    Covered
        Illegal and Ignore Bins:
            ignore_bin ack_zero_wr                       0                     -    ZERO

TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1
```