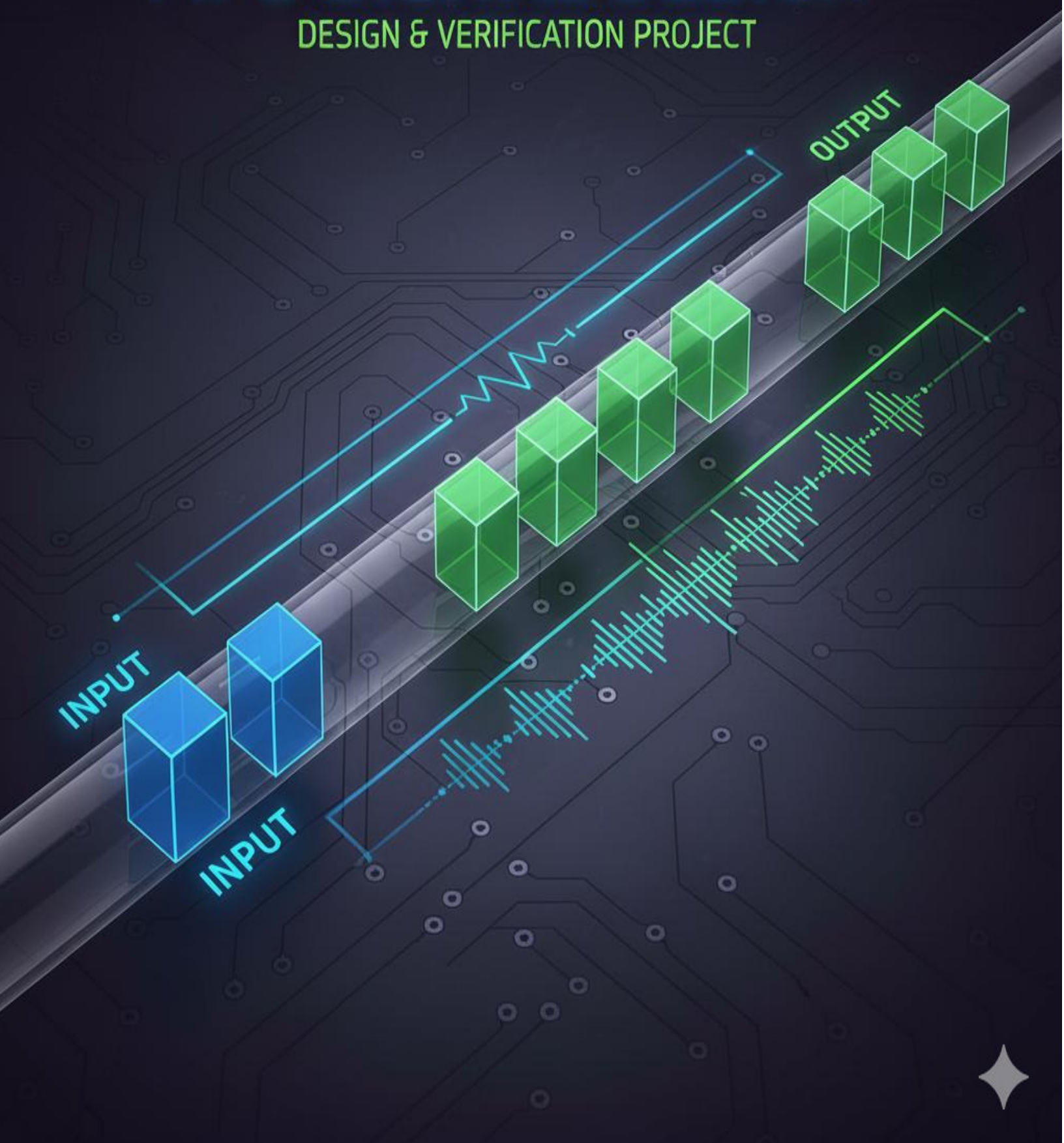
FIFO DIGITAL DESIGN

DESIGN & VERIFICATION PROJECT



Contents

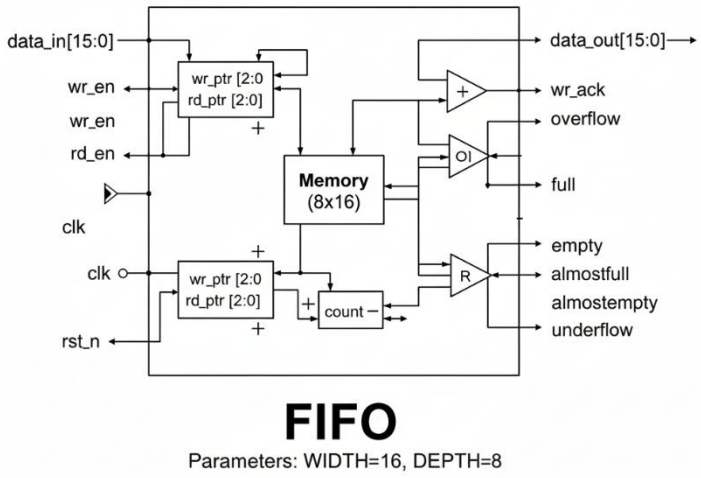
Introduction

In modern digital systems, efficient data transfer between subsystems operating at the same clock frequency is essential for maintaining throughput and reliability. A First-In First-Out (FIFO) buffer serves as a fundamental building block for this purpose, enabling sequential data storage and retrieval while preventing data loss during burst transfers or mismatched processing rates. This project focuses on the design and verification of a **Synchronous FIFO**, which operates under a single clock domain and ensures predictable, cycle-accurate data handling.

The FIFO module is parameterized by **FIFO_WIDTH** (default: 16) and **FIFO_DEPTH** (default: 8), offering flexibility in adapting to different system requirements. Its architecture incorporates critical control signals and status flags, including *full, almost full, empty, almost empty, overflow, underflow,* and *write acknowledgment,* which collectively guarantee robust data integrity under varying operational conditions.

To validate the design, a structured **verification environment** is implemented using SystemVerilog. The testbench employs randomized stimulus generation, functional coverage collection, scoreboarding, and assertion-based verification to ensure compliance with the specified requirements. Key verification objectives include validating reset behavior, pointer wraparound, status flag correctness, and robust handling of boundary conditions such as simultaneous read and write operations. The process also emphasizes achieving **100% code coverage, functional coverage, and assertion coverage** to guarantee design correctness and completeness.

Through this project, both the design and verification perspectives of a synchronous FIFO are explored, highlighting practical techniques for building reusable verification components and ensuring reliable RTL implementation. The outcome not only demonstrates the correct functionality of the FIFO but also reinforces best practices in modern hardware verification methodologies.



FIFO

Parameters: WIDTH=16, DEPTH=8

Design Specifications

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.

Bugs are Fixed

```
always @(posedge if_c.clk or negedge if_c.rst_n) begin
    if (!if_c.rst_n) begin
        wr_ptr <= 0;
        if_c.overflow <= 0; //fix
        if_c.wr_ack <= 0; //fix
    end
```

Overflow flag an wr_ack flag should be zero when the reset is asserted

```
else if(if_c.empty && if_c.rd_en) begin //fix
            if_c.underflow <= 1;
        end
        else begin
            if_c.underflow <= 0;
        end
```

Underflow flag should be non blocking assign in the always block

```
else if( {if_c.wr_en, if_c.rd_en} == 2'b11) begin //fix
                if(if_c.full)
                    count <= count - 1;
                else if(if_c.empty)
                    count <= count + 1;
                else
                    count <= count;
            end
```

Handiling the case of wr_en and rd_en are ones

```
assign if_c.almostfull = (count == if_c.FIFO_DEPTH-1)? 1 : 0; // almost full when one space is left not two
```

Almostfull flag will be high when *one space is left not two*

Verification Plan

Lablel	Design Requirement Description	Stimulus Generation	Functional Covarage	Functionality Check
FIFO_1	when rst_n is asserted the data_out signal will be zero and wr_ack flag will be low, overflowflag will be low, full flag will be low, empty flag will be high, almostfull flag will be low, almostempty flag will be low, underflow flag will be low	Directed at the start of simulation	$wr_en_cp that cover the value of wr_en in 2 bins 1- zero : that cover the zero value of wr_en 2- one : that cover the one value of wr_en $rd_en_cp that cover the value of rd_en in 2 bins 1- zero : that cover the zero value of rd_en 2- one : that cover the one value of rd_en $underflow_cp that cover the value of underflow in 2 bins 1- zero : that cover the zero value of underflow 2- one : that cover the one value of underflow $wr_ack_cp that cover the value of wr_ack in 2 bins 1- zero : that cover the zero value of wr_ack 2- one : that cover the one value of wr_ack $overflow_cp that cover the value of overflow in 2 bins 1- zero : that cover the zero value of overflow 2- one : that cover the one value of overflow $full_cp that cover the value of full in 2 bins 1- zero : that cover the zero value of full 2- one : that cover the one value of full $empty_cp that cover the value of empty in 2 bins 1- zero : that cover the zero value of empty 2- one : that cover the one value of empty $almostempty_cp that cover the value of almostempty in 2 bins 1- zero : that cover the zero value of almostempty 2- one : that cover the one value of almostempty $cross_wr_ack that cover the all crossing between wr_en_cp & rd_en_cp & wr_ack_cp ignoring the crossing of {wr_en_cp.zero} and { wr_ack_cp.one} $cross_overflow that cover the all crossing between wr_en_cp & rd_en_cp & overflow_cp ignoring the crossing of {wr_en_cp.zero} and { overflow_cp.one} $cross_underflow that cover the all crossing between wr_en_cp & rd_en_cp & underflow_cp ignoring the crossing of {rd_en_cp.zero} and {underflow_cp.one} $cross_full that cover the all crossing between wr_en_cp & rd_en_cp & full_cp ignoring the crossing of {rd_en_cp.one} and {full_cp.one} $cross_empty that cover the all crossing between wr_en_cp & rd_en_cp & empty_cp $cross_almostempty that cover the all crossing between wr_en_cp & rd_en_cp & almostempty_cp $cross_almostfull that cover the all crossing between wr_en_cp & rd_en_cp & almostfull_cp	A checker in the scoreboard class and assertion in the design file to make sure the output is correct
FIFO_2	verify when wr_en is high and rd_en is low - the data_in will stored in the FIFO untill it will be fill -underflow flag still zero - verify when the FIFO Is empty the empty flag will be high - verify whan only one element is stored the almostempty flag will be high - verify when it remain only one element to be stored the almostfull flag will be high - verify when fifo is filled the full flag will be high - verify when the FIFO is full and wr_en is high the overflow flag will be high	Randomized using FIFO_Transaction Class		A checker in the scoreboard class and assertion in the design file to make sure the output is correct
FIFO_3	verify when rd_en is high and wr_en is low - the data_out will get the elemnt from FIFO memory -overflow flag still zero - verify when it remain only one element to be stored the almostfull flag will be high - verify when the FIFO Is empty the empty flag will be high - verify whan only one element is stored the almostempty flag will be high - verify when FIFO does not have any element the empty flag will be high - verify when the FIFO is empty and rd_en is high the underflow flag will be high	Randomized using FIFO_Transaction Class		A checker in the scoreboard class and assertion in the design file to make sure the output is correct
FIFO_4	verify the read and write operation and check all the flag - verify when it remain only one element to be stored the almostfull flag will be high - verify when the FIFO Is empty the empty flag will be high - verify whan only one element is stored the almostempty flag will be high - verify when FIFO does not have any element the empty flag will be high - verify when the FIFO is empty and rd_en is high the underflow flag will be high - verify when the FIFO is full and wr_en is high the overflow flag will be high	Randomized using FIFO_Transaction Class		A checker in the scoreboard class and assertion in the design file to make sure the output is correct

Design Code & Assertion

```
////////////////////////////////////////////////////////////////////////////////
// Author: Kareem Waseem
// Course: Digital Verification using SV & UVM
// Description: FIFO Design
////////////////////////////////////////////////////////////////////////////////
module FIFO(FIFO_interface.dut if_c);

    logic [if_c.max_fifo_addr-1:0] wr_ptr, rd_ptr;
    logic [if_c.max_fifo_addr:0] count;
    logic [if_c.FIFO_WIDTH-1:0] mem [if_c.FIFO_DEPTH-1:0];

    reset_assert: assert property (@(if_c.rst_n)
    (!if_c.rst_n |->
    (wr_ptr == 0 && rd_ptr == 0 && count == 0 && if_c.overflow == 0 && if_c.wr_ack == 0 && if_c.data_out == 0)));
    reset_cover: cover property (@(if_c.rst_n)
    (!if_c.rst_n |->
    (wr_ptr == 0 && rd_ptr == 0 && count == 0 && if_c.overflow == 0 && if_c.wr_ack == 0 && if_c.data_out == 0)));

    always @(posedge if_c.clk or negedge if_c.rst_n) begin
        if (!if_c.rst_n) begin
            wr_ptr <= 0;
            if_c.overflow <= 0; //fix
            if_c.wr_ack <= 0; //fix
        end
        else if (if_c.wr_en && count < if_c.FIFO_DEPTH) begin
            mem[wr_ptr] <= if_c.data_in;
            if_c.wr_ack <= 1;
            wr_ptr <= (wr_ptr + 1)%8;
            if_c.overflow <= 0;
        end
        else begin
            if_c.wr_ack <= 0;
            if (if_c.full && if_c.wr_en)
                if_c.overflow <= 1;
            else
                if_c.overflow <= 0;
        end
    end

    always @(posedge if_c.clk or negedge if_c.rst_n) begin
        if (!if_c.rst_n) begin
            rd_ptr <= 0;
            if_c.data_out <= 0; //fix
            if_c.underflow <= 0;
        end
        else if (if_c.rd_en && count != 0) begin
            if_c.data_out <= mem[rd_ptr];
            rd_ptr <= (rd_ptr + 1)%8;
            if_c.underflow <= 0;
        end
        else if(if_c.empty && if_c.rd_en) begin //fix
            //if_c.data_out <= 0;
            if_c.underflow <= 1;
        end
        else begin
            if_c.underflow <= 0;
        end
    end

    always @(posedge if_c.clk or negedge if_c.rst_n) begin
        if (!if_c.rst_n) begin
            count <= 0;
        end
        else begin
            if  ( ({if_c.wr_en, if_c.rd_en} == 2'b10) && !if_c.full)
                count <= count + 1;
            else if ( ({if_c.wr_en, if_c.rd_en} == 2'b01) && !if_c.empty)
                count <= count - 1;
            else if( {if_c.wr_en, if_c.rd_en} == 2'b11) begin //fix
                if(if_c.full)
                    count <= count - 1;
                else if(if_c.empty)
                    count <= count + 1;
                else
                    count <= count;
            end
        end
    end

    assign if_c.full = (count == if_c.FIFO_DEPTH)? 1 : 0;
    assign if_c.empty = (count == 0)? 1 : 0;
    assign if_c.almostfull = (count == if_c.FIFO_DEPTH-1)? 1 : 0; // almost full when one space is left not two
    assign if_c.almostempty = (count == 1)? 1 : 0;

```

```
    `ifdef SIM
        property wr_ptr_prop;
            @(posedge if_c.clk) disable iff(!if_c.rst_n) (if_c.wr_en && count < if_c.FIFO_DEPTH |=> wr_ptr == ($past(wr_ptr) + 1)%if_c.FIFO_DEPTH)
        endproperty
        wr_ptr_assert       : assert property (wr_ptr_prop);
        wr_ptr_cover        : cover property (wr_ptr_prop);

        property rd_ptr_prop;
            @(posedge if_c.clk) disable iff(!if_c.rst_n) (if_c.rd_en && count != 0 |=> rd_ptr == ($past(rd_ptr) + 1)%if_c.FIFO_DEPTH)
        endproperty
        rd_ptr_assert       : assert property (rd_ptr_prop);
        rd_ptr_cover        : cover property (rd_ptr_prop);

        property wr_ptr_wrap_prop;
            @(posedge if_c.clk) (wr_ptr == if_c.FIFO_DEPTH - 1 && if_c.wr_en && count < if_c.FIFO_DEPTH |=> wr_ptr == 0)
        endproperty
        wr_ptr_wrap: assert property (wr_ptr_wrap_prop);
        wr_ptr_wrap_cover: cover property (wr_ptr_wrap_prop);

        property rd_ptr_wrap_prop;
            @(posedge if_c.clk) disable iff(!if_c.rst_n) (rd_ptr == if_c.FIFO_DEPTH - 1 && if_c.rd_en && count != 0 |=> rd_ptr == 0)
        endproperty
        rd_ptr_wrap: assert property (rd_ptr_wrap_prop);
        rd_ptr_wrap_cover: cover property (rd_ptr_wrap_prop);

        property counter1_prop;
            @(posedge if_c.clk)  disable iff(!if_c.rst_n) (({if_c.wr_en, if_c.rd_en} == 2'b10) && !if_c.full |=> count == $past(count) + 1)
        endproperty
        counter_assert1     : assert property (counter1_prop);
        counter_cover1      : cover property (counter1_prop);

        property counter2_prop;
            @(posedge if_c.clk) disable iff(!if_c.rst_n) (({if_c.wr_en, if_c.rd_en} == 2'b01) && !if_c.empty |=> count == $past(count) - 1)
        endproperty
        counter_assert2     : assert property (counter2_prop);
        counter_cover2      : cover property (counter2_prop);

        property counter3_prop;
            @(posedge if_c.clk) disable iff(!if_c.rst_n) (({if_c.wr_en, if_c.rd_en} == 2'b11) && if_c.full |=> count == $past(count) - 1)
        endproperty
        counter_assert3     : assert property (counter3_prop);
        counter_cover3      : cover property (counter3_prop);

        property counter4_prop;
            @(posedge if_c.clk) disable iff(!if_c.rst_n) (({if_c.wr_en, if_c.rd_en} == 2'b11) && if_c.empty |=> count == $past(count) + 1)
        endproperty
        counter_assert4     : assert property (counter4_prop);
        counter_cover4      : cover property (counter4_prop);

        property wr_ack1_prop;
            @(posedge if_c.clk) disable iff(!if_c.rst_n) (if_c.wr_en && count < if_c.FIFO_DEPTH |=> if_c.wr_ack == 1)
        endproperty
        wr_ack_assert1      : assert property (wr_ack1_prop);
        wr_ack_cover1       : cover property (wr_ack1_prop);

        property wr_ack2_prop;
            @(posedge if_c.clk) disable iff(!if_c.rst_n) (!if_c.wr_en || !(count < if_c.FIFO_DEPTH) |=> if_c.wr_ack == 0)
        endproperty
        wr_ack_assert2      : assert property (wr_ack2_prop);
        wr_ack_cover2       : cover property (wr_ack2_prop);

        property overflow_prop;
            @(posedge if_c.clk) disable iff(!if_c.rst_n) (if_c.wr_en && count == if_c.FIFO_DEPTH |=> if_c.overflow == 1)
        endproperty
        overflow_assert     : assert property (overflow_prop);
        overflow_cover      : cover property (overflow_prop);

        property underflow_prop;
            @(posedge if_c.clk) disable iff(!if_c.rst_n) (if_c.rd_en && count == 0 |=> if_c.underflow == 1)
        endproperty
        underflow_assert    : assert property (underflow_prop);
        underflow_cover     : cover property (underflow_prop);
    `endif
endmodule
```

Interface

```
interface FIFO_interface (input clk);
    parameter FIFO_WIDTH = 16;
    parameter FIFO_DEPTH = 8;
    localparam max_fifo_addr = $clog2(FIFO_DEPTH);

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

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

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

FIFO Top

```
module FIFO_top();
    bit clk;

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

    FIFO_interface if_c(clk);
    FIFO dut(if_c);
    FIFO_monitor monitor(if_c);
    FIFO_tb tb(if_c);

    `ifdef SIM
        always_comb begin
            if(!if_c.rst_n) begin
                wr_ptr_reset: assert final(dut.wr_ptr == 0);
                rd_ptr_reset: assert final(dut.rd_ptr == 0);
                count_reset: assert final(dut.count == 0);
                full_reset: assert final(if_c.full == 0);
                empty_reset: assert final(if_c.empty == 1);
                almostfull_reset: assert final(if_c.almostfull == 0);
                almostempty_reset: assert final(if_c.almostempty == 0);
            end
        end
    `endif
endmodule
```

FIFO Testbench

```
import FIFO_transaction_pkg::*;
import shared_pkg::*;
module FIFO_tb(FIFO_interface.tb if_c);

    FIFO_transaction FIFO_c;

    initial begin
        test_finished = 0;
        FIFO_c = new();
        // FIFO_1
        assert_reset();
        // FIFO_2
        if_c.wr_en = 1;
        if_c.rd_en = 0;
        for(int i = 0; i < 10; i++) begin
            assert(FIFO_c.randomize());
            if_c.data_in = FIFO_c.data_in;
            @(negedge if_c.clk);
            -> driver_finished;
        end
        // FIFO_3
        if_c.wr_en = 0;
        if_c.rd_en = 1;
        for(int i = 0; i < 10; i++) begin
            assert(FIFO_c.randomize());
            if_c.data_in = FIFO_c.data_in;
            @(negedge if_c.clk);
            -> driver_finished;
        end
        // FIFO_4
        for(int i = 0; i < 980; i++) begin
            assert(FIFO_c.randomize());
            if_c.rst_n = FIFO_c.rst_n;
            if_c.data_in = FIFO_c.data_in;
            if_c.wr_en = FIFO_c.wr_en;
            if_c.rd_en = FIFO_c.rd_en;
            @(negedge if_c.clk);
            -> driver_finished;
        end

        test_finished = 1;
        $display("Test finished with
 errors and error_count, correct_count);
 correct."
        $stop;
    end

    task assert_reset;
        if_c.rst_n = 0;
        @(posedge if_c.clk);
        if_c.rst_n = 1;
    endtask
endmodule
```

FIFO Transaction

```
package FIFO_transaction_pkg;

    parameter FIFO_WIDTH = 16;
    parameter FIFO_DEPTH = 8;
    parameter max_fifo_addr = $clog2(FIFO_DEPTH);

    class FIFO_transaction;
        rand logic [FIFO_WIDTH-1:0] data_in;
        rand logic clk, 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;
        int WR_EN_ON_DIST;

        constraint reset_c {
            rst_n dist {0:/5 , 1:/95};
        }
        constraint wr_en_c {
            wr_en dist {0:/(100-WR_EN_ON_DIST) , 1:/WR_EN_ON_DIST};
        }
        constraint rd_en_c {
            rd_en dist {0:/(100-RD_EN_ON_DIST) , 1:/RD_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;
        endfunction

    endclass
endpackage
```

FIFO Shared Package

```
package shared_pkg;

    logic test_finished = 0;
    int correct_count = 0, error_count = 0;
    event driver_finished;

endpackage
```

FIFO Coverage

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

        FIFO_transaction F_cvg_txn;

        covergroup fifo_cg;
            wr_en_cp: coverpoint F_cvg_txn.wr_en{
                bins zero = {0};
                bins one = {1};
            }
            rd_en_cp: coverpoint F_cvg_txn.rd_en{
                bins zero = {0};
                bins one = {1};
            }
            underflow_cp: coverpoint F_cvg_txn.underflow{
                bins zero = {0};
                bins one = {1};
            }
            wr_ack_cp: coverpoint F_cvg_txn.wr_ack{
                bins zero = {0};
                bins one = {1};
            }
            overflow_cp: coverpoint F_cvg_txn.overflow{
                bins zero = {0};
                bins one = {1};
            }
            full_cp: coverpoint F_cvg_txn.full{
                bins zero = {0};
                bins one = {1};
            }
            empty_cp: coverpoint F_cvg_txn.empty{
                bins zero = {0};
                bins one = {1};
            }
            almostempty_cp: coverpoint F_cvg_txn.almostempty{
                bins zero = {0};
                bins one = {1};
            }
            cross_wr_ack: cross wr_en_cp, rd_en_cp, wr_ack_cp{
                ignore_bins wr_en0_ack1 = binsof(wr_en_cp.zero) && binsof(wr_ack_cp.one);
            }
            cross_overflow: cross wr_en_cp, rd_en_cp, overflow_cp{
                ignore_bins wr_en0_overflow1 = binsof(wr_en_cp.zero) && binsof(overflow_cp.one);
            }
            cross_underflow: cross wr_en_cp, rd_en_cp, underflow_cp{
                ignore_bins rd_en0 = binsof(rd_en_cp.zero)&& binsof(underflow_cp.one);
            }
            cross_full: cross wr_en_cp, rd_en_cp, full_cp{
                ignore_bins rd_full = binsof(rd_en_cp.one) && binsof(full_cp.one);
            }
            cross_empty: cross wr_en_cp, rd_en_cp, empty_cp;
            cross_almostfull: cross F_cvg_txn.wr_en, F_cvg_txn.rd_en, F_cvg_txn.almostfull;
            cross_almostempty: cross wr_en_cp, rd_en_cp, almostempty_cp;
        endgroup

        function new();
            fifo_cg = new();
        endfunction

        function void sample_data(FIFO_transaction F_txn);
            F_cvg_txn = F_txn;
            fifo_cg.sample();
        endfunction
    endclass
endpackage
```

FIFO Scoreboard

```
package FIFO_scoreboard_pkg;
    import FIFO_transaction_pkg::*;
    import shared_pkg::*;
    class FIFO_scoreboard;
        logic [FIFO_WIDTH-1:0] data_out_ref = 0;
        logic full_ref = 0, empty_ref = 1, almostfull_ref = 0, almostempty_ref = 0, underflow_ref = 0, overflow_ref = 0, wr_ack_ref = 0;

        logic [3:0] count = 0;

        logic done = 0;

        logic [FIFO_WIDTH-1:0] mem [$];

        function void check_data(FIFO_transaction F_txn);
            reference_model(F_txn);
            if (F_txn.overflow != overflow_ref) begin
                $display("ERROR: Overflow mismatch %b != %b", F_txn.overflow, overflow_ref);
                error_count++;
            end
            else if (F_txn.underflow != underflow_ref) begin
                $display("ERROR: Underflow mismatch %b != %b", F_txn.underflow, underflow_ref);
                error_count++;
            end
            else if (F_txn.wr_ack != wr_ack_ref) begin
                $display("ERROR: Wr_ack mismatch %b != %b", F_txn.wr_ack, wr_ack_ref);
                error_count++;
            end
            else if (F_txn.full != full_ref) begin
                $display("ERROR: Full mismatch %b != %b", F_txn.full, full_ref);
                error_count++;
            end
            else if (F_txn.empty != empty_ref) begin
                $display("ERROR: Empty mismatch %b != %b", F_txn.empty, empty_ref);
                error_count++;
            end
            else if (F_txn.almostfull != almostfull_ref) begin
                $display("ERROR: Almostfull mismatch %b != %b", F_txn.almostfull, almostfull_ref);
                error_count++;
            end
            else if (F_txn.almostempty != almostempty_ref) begin
                $display("ERROR: Almostempty mismatch %b != %b", F_txn.almostempty, almostempty_ref);
                error_count++;
            end
            else if (F_txn.data_out != data_out_ref) begin
                $display("ERROR: Data mismatch %h != %h", F_txn.data_out, data_out_ref);
                error_count++;
            end
            else begin
                correct_count++;
            end
        endfunction

        function void reference_model(FIFO_transaction F_txn);
            done = 0;
            if (!F_txn.rst_n) begin
                wr_ack_ref     = 0;
                overflow_ref   = 0;
                data_out_ref   = 0;
                underflow_ref  = 0;
                mem.delete();
            end
            else begin
                if(F_txn.wr_en && F_txn.rd_en) begin
                    if(full_ref) begin
                        data_out_ref = mem.pop_front();
                        underflow_ref = 0;
                        overflow_ref = 1;
                        wr_ack_ref = 0;
                        done = 1;
                    end
                    else if(empty_ref) begin
                        mem.push_back(F_txn.data_in);
                        wr_ack_ref = 1;
                        overflow_ref = 0;
                        underflow_ref = 1;
                        done = 1;
                    end
                end

                if(!done) begin
                    if (F_txn.wr_en && mem.size() < FIFO_DEPTH) begin
                        mem.push_back(F_txn.data_in);
                        wr_ack_ref = 1;
                        overflow_ref = 0;
                    end
                    else begin
                        wr_ack_ref = 0;
                        if (full_ref && F_txn.wr_en)
                            overflow_ref = 1;
                        else
                            overflow_ref = 0;
                    end

                    if (F_txn.rd_en && mem.size() != 0) begin
                        data_out_ref = mem.pop_front();
                        underflow_ref = 0;
                    end
                    else if(F_txn.rd_en && mem.size() == 0) begin
                        underflow_ref = 1;
                    end
                    else begin
                        underflow_ref = 0;
                    end
                end
            end

            full_ref      = (mem.size() == FIFO_DEPTH) ? 1 : 0;
            empty_ref     = (mem.size() == 0) ? 1 : 0;
            almostfull_ref  = (mem.size() == FIFO_DEPTH - 1) ? 1 : 0;
            almostempty_ref = (mem.size() == 1) ? 1 : 0;
        endfunction

    endclass
endpackage
```

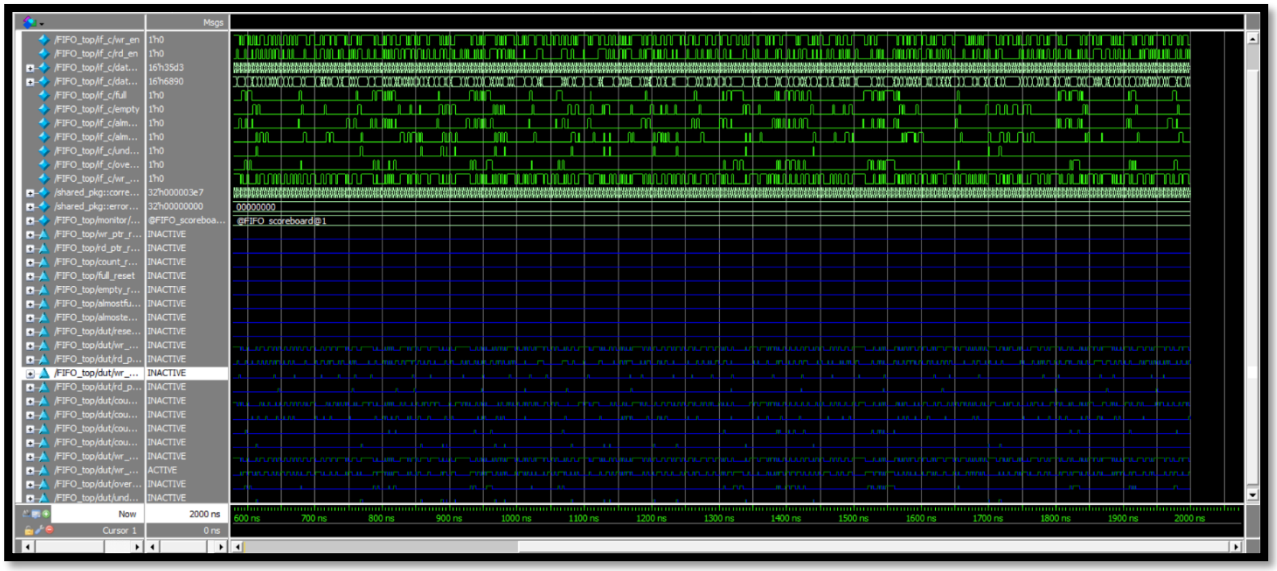
FIFO Monitor

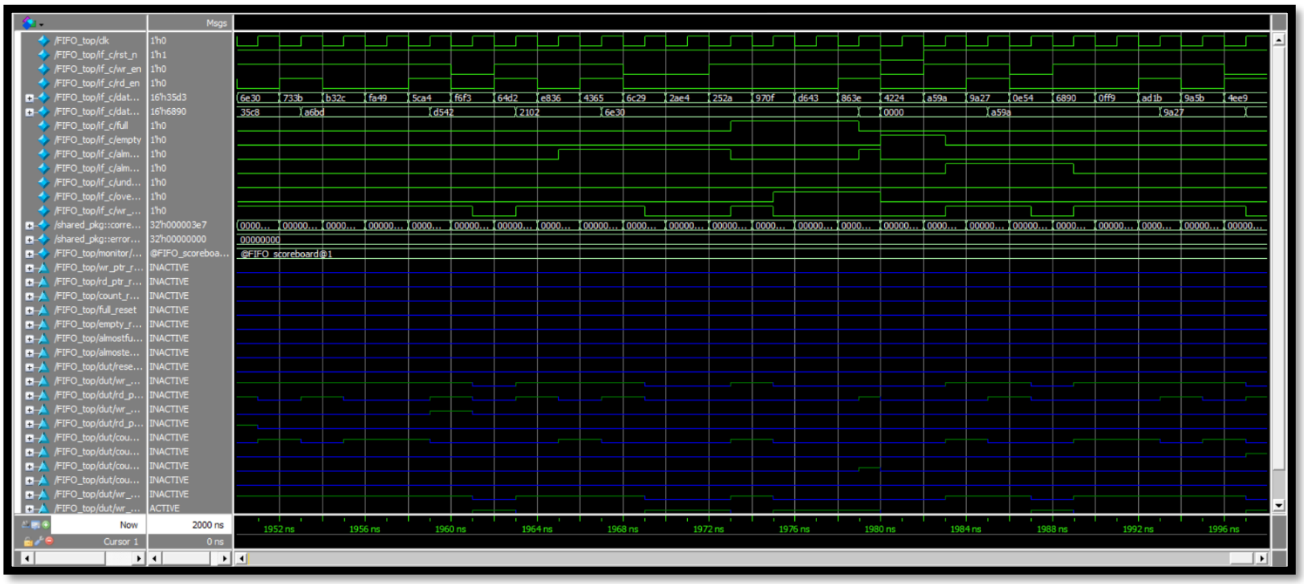
```
import FIFO_scoreboard_pkg::*;
import FIFO_transaction_pkg::*;
import FIFO_coverage_pkg::*;
import shared_pkg::*;
module FIFO_monitor(FIFO_interface.monitor if_c);
    FIFO_transaction F_txn = new();
    FIFO_scoreboard F_scoreboard = new();
    FIFO_coverage F_coverage = new();

    initial begin
        forever begin
            wait(driver_finished.triggered);
            @(negedge if_c.clk);
            F_txn.rst_n = if_c.rst_n;
            F_txn.data_in = if_c.data_in;
            F_txn.clk = if_c.clk;
            F_txn.wr_en = if_c.wr_en;
            F_txn.rd_en = if_c.rd_en;
            F_txn.data_out = if_c.data_out;
            F_txn.wr_ack = if_c.wr_ack;
            F_txn.overflow = if_c.overflow;
            F_txn.full = if_c.full;
            F_txn.empty = if_c.empty;
            F_txn.almostfull = if_c.almostfull;
            F_txn.almostempty = if_c.almostempty;
            F_txn.underflow = if_c.underflow;

            fork
                begin
                    F_coverage.sample_data(F_txn);
                end
                begin
                    F_scoreboard.check_data(F_txn);
                end
            join
            if(test_finished == 1) begin
                break;
            end
        end
    end
endmodule
```

Do File

```
vlib work
vlog -f src_files.list +define+SIM -cover bcesf +cover +acc -covercells
vsim -coverage -voptargs=+acc FIFO_top -cover
add wave *
add wave -position insertpoint  \
sim:/FIFO_top/if_c/rst_n \
sim:/FIFO_top/if_c/wr_en \
sim:/FIFO_top/if_c/rd_en \
sim:/FIFO_top/if_c/data_in \
sim:/FIFO_top/if_c/data_out \
sim:/FIFO_top/if_c/full \
sim:/FIFO_top/if_c/empty \
sim:/FIFO_top/if_c/almostfull \
sim:/FIFO_top/if_c/almostempty \
sim:/FIFO_top/if_c/underflow \
sim:/FIFO_top/if_c/overflow \
sim:/FIFO_top/if_c/wr_ack

add wave -position insertpoint  \
sim:/shared_pkg::correct_count \
sim:/shared_pkg::error_count

add wave -position insertpoint  \
sim:/FIFO_top/monitor/F_scoreboard


coverage save FIFO_top.ucdb -onexit
run 0
run -all
```

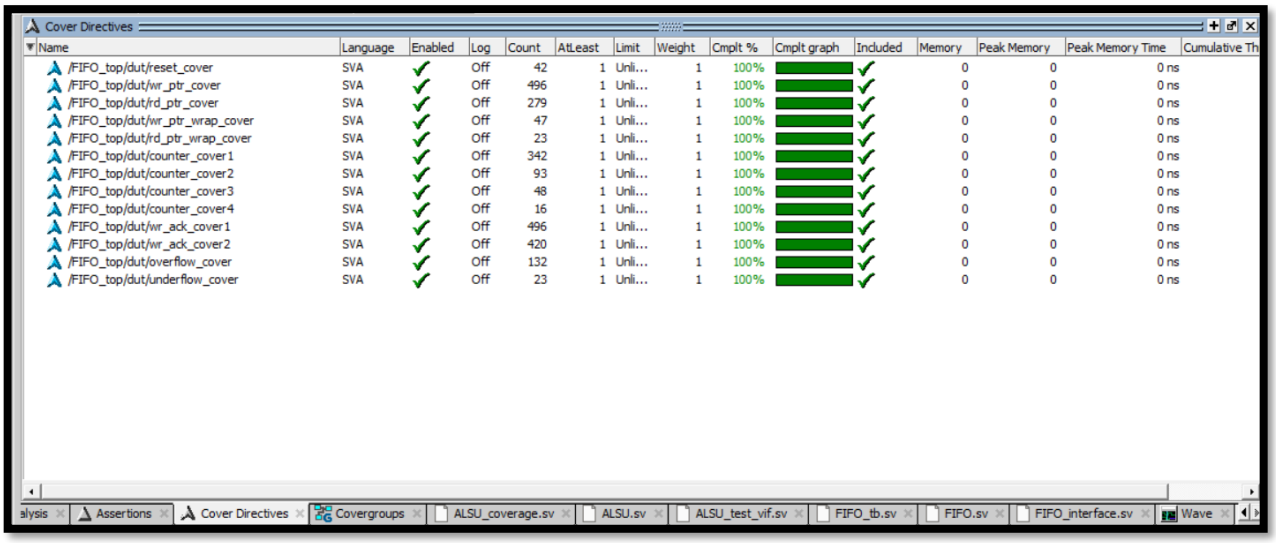
Questa Sim Snippets

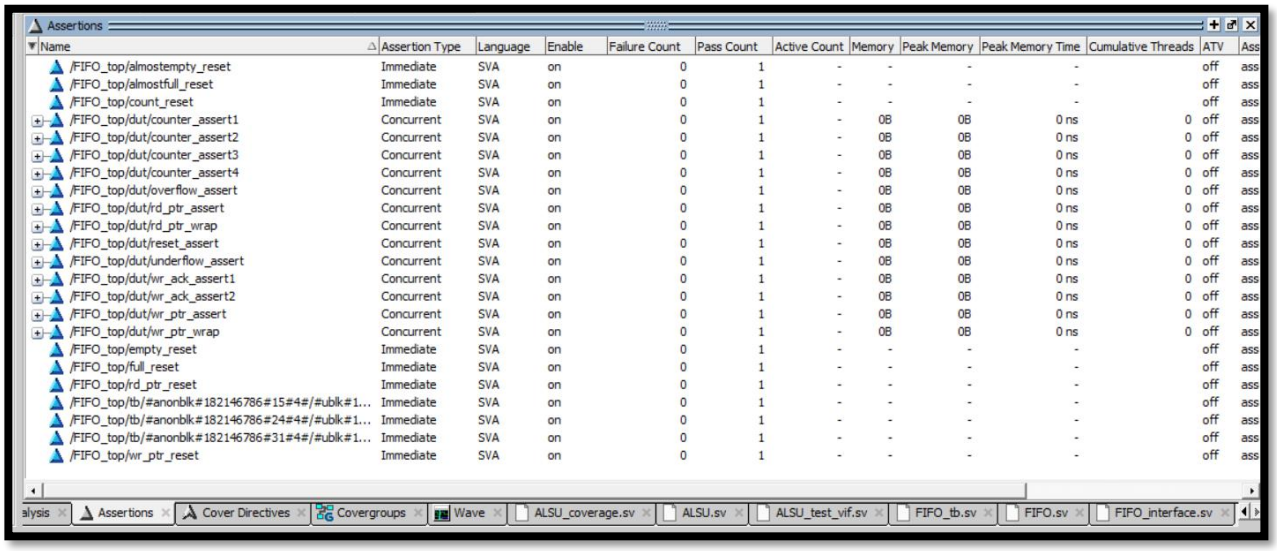




```
# Test finished with 0 errors and 999 correct.
# ** Note: $stop    : FIFO_tb.sv(45)
#    Time: 2 us  Iteration: 1  Instance: /FIFO_top/tb
# Break in Module FIFO_tb at FIFO_tb.sv line 45

VSIM 46>
```

Assertion Coverage QuestaSim



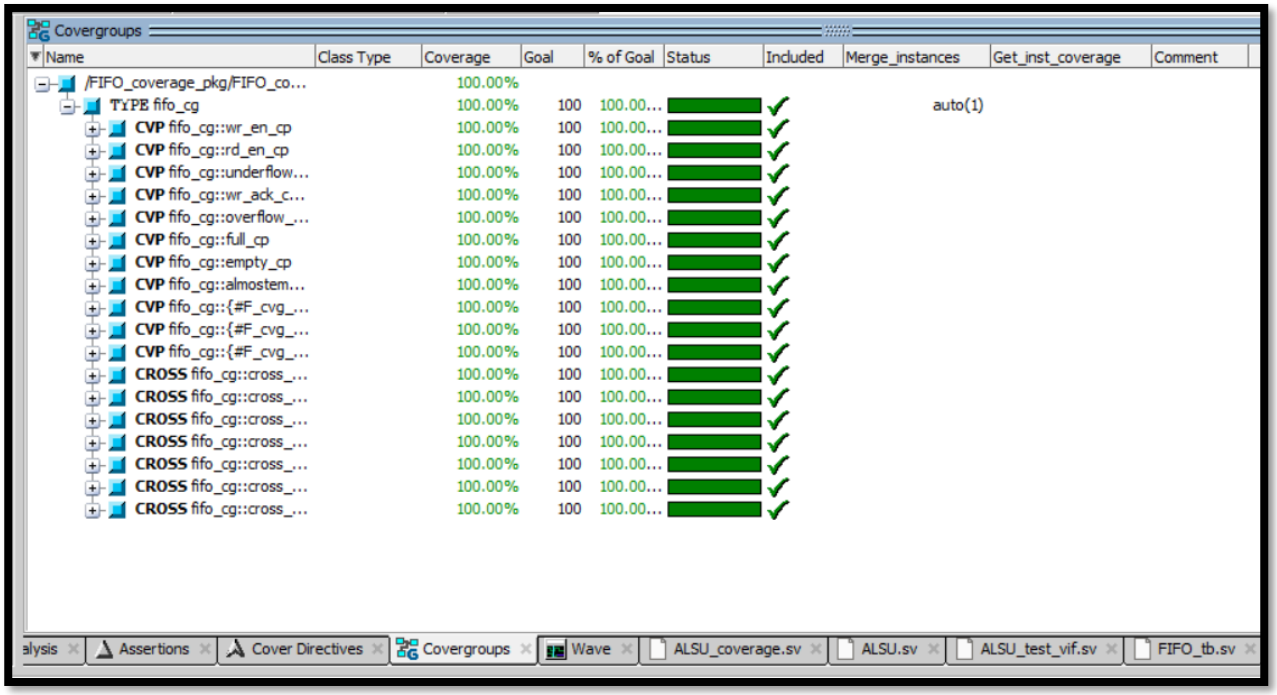


Assertion Coverage Report

```
Assertion Coverage:
    Assertions                      13        13         0   100.00%
--------------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
--------------------------------------------------------------------
/FIFO_top/dut/reset_assert
                     FIFO.sv(14)                        0          1
/FIFO_top/dut/wr_ptr_assert
                     FIFO.sv(89)                        0          1
/FIFO_top/dut/rd_ptr_assert
                     FIFO.sv(95)                        0          1
/FIFO_top/dut/wr_ptr_wrap
                     FIFO.sv(101)                       0          1
/FIFO_top/dut/rd_ptr_wrap
                     FIFO.sv(107)                       0          1
/FIFO_top/dut/counter_assert1
                     FIFO.sv(113)                       0          1
/FIFO_top/dut/counter_assert2
                     FIFO.sv(119)                       0          1
/FIFO_top/dut/counter_assert3
                     FIFO.sv(125)                       0          1
/FIFO_top/dut/counter_assert4
                     FIFO.sv(131)                       0          1
/FIFO_top/dut/wr_ack_assert1
                     FIFO.sv(137)                       0          1
/FIFO_top/dut/wr_ack_assert2
                     FIFO.sv(143)                       0          1
/FIFO_top/dut/overflow_assert
                     FIFO.sv(149)                       0          1
/FIFO_top/dut/underflow_assert
                     FIFO.sv(155)                       0          1
```

```
DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------------
Name                                     Design Design   Lang File(Line)      Hits Status
                                         Unit   UnitType
--------------------------------------------------------------------------------------------------
/FIFO_top/dut/reset_cover                FIFO   Verilog  SVA  FIFO.sv(17)      42 Covered
/FIFO_top/dut/wr_ptr_cover               FIFO   Verilog  SVA  FIFO.sv(90)     496 Covered
/FIFO_top/dut/rd_ptr_cover               FIFO   Verilog  SVA  FIFO.sv(96)     279 Covered
/FIFO_top/dut/wr_ptr_wrap_cover          FIFO   Verilog  SVA  FIFO.sv(102)     47 Covered
/FIFO_top/dut/rd_ptr_wrap_cover          FIFO   Verilog  SVA  FIFO.sv(108)     23 Covered
/FIFO_top/dut/counter_cover1             FIFO   Verilog  SVA  FIFO.sv(114)    342 Covered
/FIFO_top/dut/counter_cover2             FIFO   Verilog  SVA  FIFO.sv(120)     93 Covered
/FIFO_top/dut/counter_cover3             FIFO   Verilog  SVA  FIFO.sv(126)     48 Covered
/FIFO_top/dut/counter_cover4             FIFO   Verilog  SVA  FIFO.sv(132)     16 Covered
/FIFO_top/dut/wr_ack_cover1              FIFO   Verilog  SVA  FIFO.sv(138)    496 Covered
/FIFO_top/dut/wr_ack_cover2              FIFO   Verilog  SVA  FIFO.sv(144)    420 Covered
/FIFO_top/dut/overflow_cover             FIFO   Verilog  SVA  FIFO.sv(150)    132 Covered
/FIFO_top/dut/underflow_cover            FIFO   Verilog  SVA  FIFO.sv(156)     23 Covered

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 13
```

Functional Coverage QuestaSim



Functional Coverage Report

```
================================================================================
=== Instance: /FIFO_coverage_pkg
=== Design Unit: work.FIFO_coverage_pkg
================================================================================

Covergroup Coverage:
    Covergroups                     1        na        na   100.00%
        Coverpoints/Crosses        18        na        na        na
            Covergroup Bins        70        70         0   100.00%
--------------------------------------------------------------------------------------------------------------
Covergroup                                          Metric       Goal       Bins    Status

--------------------------------------------------------------------------------------------------------------
 TYPE /FIFO_coverage_pkg/FIFO_coverage/fifo_cg      100.00%        100          -    Covered
    covered/total bins:                                 70         70          -
    missing/total bins:                                  0         70          -
    % Hit:                                         100.00%        100          -
    Coverpoint wr_en_cp                            100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
        bin zero                                       317          1          -    Covered
        bin one                                        682          1          -    Covered
    Coverpoint rd_en_cp                            100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
        bin zero                                       667          1          -    Covered
        bin one                                        332          1          -    Covered
    Coverpoint underflow_cp                        100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
        bin zero                                       973          1          -    Covered
        bin one                                         26          1          -    Covered
    Coverpoint wr_ack_cp                           100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
        bin zero                                       481          1          -    Covered
        bin one                                        518          1          -    Covered
    Coverpoint overflow_cp                         100.00%        100          -    Covered
        covered/total bins:                              2          2          -
        missing/total bins:                              0          2          -
        % Hit:                                     100.00%        100          -
        bin zero                                       861          1          -    Covered
        bin one                                        138          1          -    Covered
```

Coverpoint full_cp	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin zero	775	1	-	Covered
bin one	224	1	-	Covered
Coverpoint empty_cp	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin zero	920	1	-	Covered
bin one	79	1	-	Covered
Coverpoint almostempty_cp	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin zero	882	1	-	Covered
bin one	117	1	-	Covered
Coverpoint #F_cvg_txn.wr_en__0#	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	317	1	-	Covered
bin auto[1]	682	1	-	Covered
Coverpoint #F_cvg_txn.rd_en__1#	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	667	1	-	Covered
bin auto[1]	332	1	-	Covered
Coverpoint #F_cvg_txn.almostfull__2#	100.00%	100	-	Covered
covered/total bins:	2	2	-	
missing/total bins:	0	2	-	
% Hit:	100.00%	100	-	
bin auto[0]	846	1	-	Covered
bin auto[1]	153	1	-	Covered
Cross cross_wr_ack	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 <one,one,one>	164	1	-	Covered
bin <one,zero,one>	354	1	-	Covered
bin <one,one,zero>	57	1	-	Covered
bin <zero,one,zero>	111	1	-	Covered
bin <one,zero,zero>	107	1	-	Covered
bin <zero,zero,zero>	206	1	-	Covered
Illegal and Ignore Bins:				
ignore_bin wr_en0_ack1	0		-	ZERO
[illegible]				
Cross cross_overflow	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 <one,one,one>	49	1	-	Covered
bin <one,zero,one>	89	1	-	Covered
bin <one,one,zero>	172	1	-	Covered
bin <zero,one,zero>	111	1	-	Covered
bin <one,zero,zero>	372	1	-	Covered
bin <zero,zero,zero>	206	1	-	Covered
Illegal and Ignore Bins:				
ignore_bin wr_en0_overflow1	0		-	ZERO
Cross cross_underflow	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 <one,one,one>	19	1	-	Covered
bin <one,one,zero>	202	1	-	Covered
bin <zero,one,one>	7	1	-	Covered
bin <zero,one,zero>	104	1	-	Covered
bin <one,zero,zero>	461	1	-	Covered
bin <zero,zero,zero>	206	1	-	Covered
Illegal and Ignore Bins:				
ignore_bin rd_en0	0		-	ZERO
Cross cross_full	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 <one,one,zero>	221	1	-	Covered
bin <zero,one,zero>	111	1	-	Covered
bin <one,zero,one>	173	1	-	Covered
bin <one,zero,zero>	288	1	-	Covered
bin <zero,zero,one>	51	1	-	Covered
bin <zero,zero,zero>	155	1	-	Covered
Illegal and Ignore Bins:				
ignore_bin rd_full	0		-	ZERO

Cross cross_empty	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 <one,one,one>	8	1	-	Covered
bin <zero,one,one>	26	1	-	Covered
bin <one,zero,one>	18	1	-	Covered
bin <zero,zero,one>	27	1	-	Covered
bin <one,one,zero>	213	1	-	Covered
bin <zero,one,zero>	85	1	-	Covered
bin <one,zero,zero>	443	1	-	Covered
bin <zero,zero,zero>	179	1	-	Covered
Cross cross_almostfull	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]>	74	1	-	Covered
bin <auto[0],auto[1],auto[1]>	22	1	-	Covered
bin <auto[1],auto[0],auto[1]>	31	1	-	Covered
bin <auto[0],auto[0],auto[1]>	26	1	-	Covered
bin <auto[1],auto[1],auto[0]>	147	1	-	Covered
bin <auto[0],auto[1],auto[0]>	89	1	-	Covered
bin <auto[1],auto[0],auto[0]>	430	1	-	Covered
bin <auto[0],auto[0],auto[0]>	180	1	-	Covered
Cross cross_almostempty	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 <one,one,one>	47	1	-	Covered
bin <zero,one,one>	13	1	-	Covered
bin <one,zero,one>	39	1	-	Covered
bin <zero,zero,one>	18	1	-	Covered
bin <one,one,zero>	174	1	-	Covered
bin <zero,one,zero>	98	1	-	Covered
bin <one,zero,zero>	422	1	-	Covered
bin <zero,zero,zero>	188	1	-	Covered

Code Coverage

Interface Toggle

```
================================================================================
=== Instance: /FIFO_top/if_c
=== Design Unit: work.FIFO_interface
================================================================================
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         86        86         0   100.00%

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

Toggle Coverage for instance /FIFO_top/if_c --

                                           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 Cover

```
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                        26        26         0   100.00%

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

Branch Coverage for instance /FIFO_top/dut

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File FIFO.sv
------------------------------------IF Branch------------------------------------
    20                                      1042     Count coming in to IF
    20             1                         84                if (!if_c.rst_n) begin
    25             1                        518                else if (if_c.wr_en && count < if_c.FIFO_DEPTH) begin
    31             1                        440                else begin
Branch totals: 3 hits of 3 branches = 100.00%

------------------------------------IF Branch------------------------------------
    33                                      440     Count coming in to IF
    33             1                        138                      if (if_c.full && if_c.wr_en)
    35             1                        302                      else
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    41                                      1042     Count coming in to IF
    41             1                         84                if (!if_c.rst_n) begin
    46             1                        295                else if (if_c.rd_en && count != 0) begin
    51             1                         26                else if(if_c.empty && if_c.rd_en) begin //fix
    55             1                        637                else begin
Branch totals: 4 hits of 4 branches = 100.00%

------------------------------------IF Branch------------------------------------
    61                                      936     Count coming in to IF
    61             1                         84                if (!if_c.rst_n) begin
    64             1                        852                else begin
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    65                                      852     Count coming in to IF
    65             1                        354                      if      ( ({if_c.wr_en, if_c.rd_en} == 2'b10) && !if_c.full)
    67             1                        101                      else if ( ({if_c.wr_en, if_c.rd_en} == 2'b01) && !if_c.empty)
    69             1                        189                      else if( {if_c.wr_en, if_c.rd_en} == 2'b11) begin //fix
                                            208     All False Count
Branch totals: 4 hits of 4 branches = 100.00%
```

```
------------------------------------IF Branch------------------------------------
    70                                      189     Count coming in to IF
    70             1                         49                       if(if_c.full)
    72             1                         19                       else if(if_c.empty)
    74             1                        121                       else
Branch totals: 3 hits of 3 branches = 100.00%

------------------------------------IF Branch------------------------------------
    80                                      565     Count coming in to IF
    80             1                         84        assign if_c.full = (count == if_c.FIFO_DEPTH)? 1 : 0;
    80             2                        481        assign if_c.full = (count == if_c.FIFO_DEPTH)? 1 : 0;
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    81                                      565     Count coming in to IF
    81             1                         58        assign if_c.empty = (count == 0)? 1 : 0;
    81             2                        507        assign if_c.empty = (count == 0)? 1 : 0;
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    82                                      565     Count coming in to IF
    82             1                        102        assign if_c.almostfull = (count == if_c.FIFO_DEPTH-1)? 1 : 0; // almost full when one space is left not two
    82             2                        463        assign if_c.almostfull = (count == if_c.FIFO_DEPTH-1)? 1 : 0; // almost full when one space is left not two
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
    83                                      565     Count coming in to IF
    83             1                         71        assign if_c.almostempty = (count == 1)? 1 : 0;
    83             2                        494        assign if_c.almostempty = (count == 1)? 1 : 0;
Branch totals: 2 hits of 2 branches = 100.00%
```

Condition Cover

```
Condition Coverage:
    Enabled Coverage              Bins   Covered    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Conditions                      20        20         0   100.00%

================================Condition Details================================

Condition Coverage for instance /FIFO_top/dut --

  File FIFO.sv
----------------Focused Condition View-------------------
Line       25 Item    1  (if_c.wr_en && (count < if_c.FIFO_DEPTH))
Condition totals: 2 of 2 input terms covered = 100.00%

                Input Term   Covered  Reason for no coverage   Hint
               -----------  --------  ----------------------  --------------
               if_c.wr_en         Y
  (count < if_c.FIFO_DEPTH)       Y

     Rows:       Hits  FEC Target                  Non-masking condition(s)
 ---------  ---------  --------------------        -------------------------
  Row   1:          1  if_c.wr_en_0                -
  Row   2:          1  if_c.wr_en_1                (count < if_c.FIFO_DEPTH)
  Row   3:          1  (count < if_c.FIFO_DEPTH)_0  if_c.wr_en
  Row   4:          1  (count < if_c.FIFO_DEPTH)_1  if_c.wr_en

----------------Focused Condition View-------------------
Line       33 Item    1  (if_c.full && if_c.wr_en)
Condition totals: 2 of 2 input terms covered = 100.00%

  Input Term   Covered  Reason for no coverage   Hint
 -----------  --------  ----------------------  --------------
   if_c.full         Y
  if_c.wr_en         Y

     Rows:       Hits  FEC Target           Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  if_c.full_0          -
  Row   2:          1  if_c.full_1          if_c.wr_en
  Row   3:          1  if_c.wr_en_0         if_c.full
  Row   4:          1  if_c.wr_en_1         if_c.full

----------------Focused Condition View-------------------
Line       46 Item    1  (if_c.rd_en && (count != 0))
Condition totals: 2 of 2 input terms covered = 100.00%

   Input Term   Covered  Reason for no coverage   Hint
  -----------  --------  ----------------------  --------------
   if_c.rd_en         Y
 (count != 0)         Y

    Rows:       Hits  FEC Target           Non-masking condition(s)
---------  ---------  --------------------  -------------------------
 Row   1:          1  if_c.rd_en_0          -
 Row   2:          1  if_c.rd_en_1          (count != 0)
 Row   3:          1  (count != 0)_0        if_c.rd_en
 Row   4:          1  (count != 0)_1        if_c.rd_en

----------------Focused Condition View-------------------
Line       51 Item    1  (if_c.empty && if_c.rd_en)
Condition totals: 2 of 2 input terms covered = 100.00%

  Input Term   Covered  Reason for no coverage   Hint
 -----------  --------  ----------------------  --------------
  if_c.empty         Y
  if_c.rd_en         Y

    Rows:       Hits  FEC Target           Non-masking condition(s)
---------  ---------  --------------------  -------------------------
 Row   1:          1  if_c.empty_0          -
 Row   2:          1  if_c.empty_1          if_c.rd_en
 Row   3:          1  if_c.rd_en_0          if_c.empty
 Row   4:          1  if_c.rd_en_1          if_c.empty

----------------Focused Condition View-------------------
Line       65 Item    1  ((~if_c.rd_en && if_c.wr_en) && ~if_c.full)
Condition totals: 3 of 3 input terms covered = 100.00%
```

```
  Input Term   Covered  Reason for no coverage   Hint
 -----------  --------  -----------------------  --------------
  if_c.rd_en         Y
  if_c.wr_en         Y
   if_c.full         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  if_c.rd_en_0          (~if_c.full && if_c.wr_en)
  Row   2:          1  if_c.rd_en_1          -
  Row   3:          1  if_c.wr_en_0          ~if_c.rd_en
  Row   4:          1  if_c.wr_en_1          (~if_c.full && ~if_c.rd_en)
  Row   5:          1  if_c.full_0           (~if_c.rd_en && if_c.wr_en)
  Row   6:          1  if_c.full_1           (~if_c.rd_en && if_c.wr_en)

----------------Focused Condition View-------------------
Line       67 Item    1  ((if_c.rd_en && ~if_c.wr_en) && ~if_c.empty)
Condition totals: 3 of 3 input terms covered = 100.00%

  Input Term   Covered  Reason for no coverage   Hint
 -----------  --------  -----------------------  --------------
  if_c.rd_en         Y
  if_c.wr_en         Y
  if_c.empty         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  if_c.rd_en_0          -
  Row   2:          1  if_c.rd_en_1          (~if_c.empty && ~if_c.wr_en)
  Row   3:          1  if_c.wr_en_0          (~if_c.empty && if_c.rd_en)
  Row   4:          1  if_c.wr_en_1          if_c.rd_en
  Row   5:          1  if_c.empty_0          (if_c.rd_en && ~if_c.wr_en)
  Row   6:          1  if_c.empty_1          (if_c.rd_en && ~if_c.wr_en)

----------------Focused Condition View-------------------
Line       69 Item    1  (if_c.rd_en & if_c.wr_en)
Condition totals: 2 of 2 input terms covered = 100.00%

  Input Term   Covered  Reason for no coverage   Hint
 -----------  --------  -----------------------  --------------
  if_c.rd_en         Y
  if_c.wr_en         Y
```

```
     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  if_c.rd_en_0          if_c.wr_en
  Row   2:          1  if_c.rd_en_1          if_c.wr_en
  Row   3:          1  if_c.wr_en_0          if_c.rd_en
  Row   4:          1  if_c.wr_en_1          if_c.rd_en

----------------Focused Condition View-------------------
Line       80 Item    1  (count == if_c.FIFO_DEPTH)
Condition totals: 1 of 1 input term covered = 100.00%

                 Input Term   Covered  Reason for no coverage   Hint
                -----------  --------  ----------------------  --------------
  (count == if_c.FIFO_DEPTH)         Y

     Rows:       Hits  FEC Target                    Non-masking condition(s)
 ---------  ---------  --------------------          -------------------------
  Row   1:          1  (count == if_c.FIFO_DEPTH)_0  -
  Row   2:          1  (count == if_c.FIFO_DEPTH)_1  -

----------------Focused Condition View-------------------
Line       81 Item    1  (count == 0)
Condition totals: 1 of 1 input term covered = 100.00%

    Input Term   Covered  Reason for no coverage   Hint
   -----------  --------  ----------------------  --------------
  (count == 0)         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  (count == 0)_0        -
  Row   2:          1  (count == 0)_1        -

----------------Focused Condition View-------------------
Line       82 Item    1  (count == (if_c.FIFO_DEPTH - 1))
Condition totals: 1 of 1 input term covered = 100.00%

                      Input Term   Covered  Reason for no coverage   Hint
                     -----------  --------  ----------------------  --------------
  (count == (if_c.FIFO_DEPTH - 1))         Y

     Rows:       Hits  FEC Target                          Non-masking condition(s)
 ---------  ---------  --------------------                -------------------------
  Row   1:          1  (count == (if_c.FIFO_DEPTH - 1))_0  -
  Row   2:          1  (count == (if_c.FIFO_DEPTH - 1))_1  -

----------------Focused Condition View-------------------
Line       83 Item    1  (count == 1)
Condition totals: 1 of 1 input term covered = 100.00%

    Input Term   Covered  Reason for no coverage   Hint
   -----------  --------  ----------------------  --------------
  (count == 1)         Y

     Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
  Row   1:          1  (count == 1)_0        -
  Row   2:          1  (count == 1)_1        -
```

Statement Cover

```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      31        31         0   100.00%

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

Statement Coverage for instance /FIFO_top/dut --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File FIFO.sv
    6                                              module FIFO(FIFO_interface.dut if_c);
    7
    8                                                 logic [if_c.max_fifo_addr-1:0] wr_ptr, rd_ptr;
    9                                                  logic [if_c.max_fifo_addr:0] count;
    10                                                logic [if_c.FIFO_WIDTH-1:0] mem [if_c.FIFO_DEPTH-1:0];
    11
    12                                                reset_assert: assert property (@(if_c.rst_n)
    13                                                (!if_c.rst_n |->
    14                                                (wr_ptr == 0 && rd_ptr == 0 && count == 0 && if_c.overflow == 0 && if_c.wr_ack == 0 && if_c.data_out == 0)));
    15                                                reset_cover: cover property (@(if_c.rst_n)
    16                                                (!if_c.rst_n |->
    17                                                (wr_ptr == 0 && rd_ptr == 0 && count == 0 && if_c.overflow == 0 && if_c.wr_ack == 0 && if_c.data_out == 0)));
    18
    19              1                       1042      always @(posedge if_c.clk or negedge if_c.rst_n) begin
    20                                                        if (!if_c.rst_n) begin
    21              1                         84                      wr_ptr <= 0;
    22              1                         84                      if_c.overflow <= 0; //fix
    23              1                         84                      if_c.wr_ack <= 0; //fix
    24                                                        end
    25                                                        else if (if_c.wr_en && count < if_c.FIFO_DEPTH) begin
    26              1                        518                      mem[wr_ptr] <= if_c.data_in;
    27              1                        518                      if_c.wr_ack <= 1;
    28              1                        518                      wr_ptr <= (wr_ptr + 1)%8;
    29              1                        518                      if_c.overflow <= 0;
    30                                                        end
    31                                                        else begin
    32              1                        440                      if_c.wr_ack <= 0;
    33                                                                if (if_c.full && if_c.wr_en)
    34              1                        138                              if_c.overflow <= 1;
    35                                                                else
    36              1                        302                              if_c.overflow <= 0;
    37                                                        end
    38                                                end
    39
    40              1                       1042      always @(posedge if_c.clk or negedge if_c.rst_n) begin
    41                                                        if (!if_c.rst_n) begin
    42              1                         84                      rd_ptr <= 0;
    43              1                         84                      if_c.data_out <= 0; //fix
    44              1                         84                      if_c.underflow <= 0;
    45                                                        end
    46                                                        else if (if_c.rd_en && count != 0) begin
    47              1                        295                      if_c.data_out <= mem[rd_ptr];
    48              1                        295                      rd_ptr <= (rd_ptr + 1)%8;
    49              1                        295                      if_c.underflow <= 0;
    50                                                        end
    51                                                        else if(if_c.empty && if_c.rd_en) begin //fix
    52                                                                //if_c.data_out <= 0;
    53              1                         26                      if_c.underflow <= 1;
    54                                                        end
    55                                                        else begin
    56              1                        637                      if_c.underflow <= 0;
    57                                                        end
    58                                                end
    59
    60              1                        936      always @(posedge if_c.clk or negedge if_c.rst_n) begin
    61                                                        if (!if_c.rst_n) begin
    62              1                         84                      count <= 0;
    63                                                        end
    64                                                        else begin
    65                                                                if      ( ({if_c.wr_en, if_c.rd_en} == 2'b10) && !if_c.full)
    66              1                        354                              count <= count + 1;
    67                                                                else if ( ({if_c.wr_en, if_c.rd_en} == 2'b01) && !if_c.empty)
    68              1                        101                              count <= count - 1;
    69                                                                else if( {if_c.wr_en, if_c.rd_en} == 2'b11) begin //fix
    70                                                                        if(if_c.full)
    71              1                         49                                    count <= count - 1;
    72                                                                        else if(if_c.empty)
    73              1                         19                                    count <= count + 1;
    74                                                                        else
    75              1                        121                                    count <= count;
    76                                                                end
    77                                                        end
    78                                                end
    79
    80              1                        566      assign if_c.full = (count == if_c.FIFO_DEPTH)? 1 : 0;
    81              1                        566      assign if_c.empty = (count == 0)? 1 : 0;
    82              1                        566      assign if_c.almostfull = (count == if_c.FIFO_DEPTH-1)? 1 : 0; // almost full when one space is left not two
    83              1                        566      assign if_c.almostempty = (count == 1)? 1 : 0;
```

Toggle Cover

```
================================Toggle Details================================

Toggle Coverage for instance /FIFO_top/dut --

                                          Node      1H->0L      0L->1H  "Coverage"
                                          ----------------------------------------
                                    count[3-0]           1           1      100.00
                                   rd_ptr[2-0]           1           1      100.00
                                   wr_ptr[2-0]           1           1      100.00

Total Node Count     =         10
Toggled Node Count   =         10
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (20 of 20 bins)
```