FIFO design and verification

Youssef mohamed Abdellal

Eng. Kareem Wassem

Table of Contents

1. Specifications of design

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.

2. Verification plan

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
RESET_1	When reset is asserted (rst_n = 0), all internal registers (wr_ptr, rd_ptr, count) should be cleared to 0	Apply rst_n=0 at start of simulation, then randomize with 3% probability during test using constraint reset	Cover reset assertion with all possible FIFO states (empty, full, partial)	Assert that wr_ptr = 0, rd_ptr = 0, count = 0, overflow = 0 , underflow = 0 immediately after reset
WRITE_1	When wr_en = 1 and FIFO is not full, data should be written to memory at wr_ptr location & count should incremented	Randomize data_in with wr_en=1 using 70% distribution, ensure FIFO not full	Cover write operations at different fill levels: empty, partial, almost full	Assert that wr_ack arrised on next cycle and data is stored correctly & Verify count increments correctly
WRITE_2	When wr_en = 1 and FIFO is not full, wr_ptr should increment (with wrap-around at FIFO_DEPTH)	Generate consecutive writes until FIFO is full	Cover wr_ptr increment from 0 to FIFO_DEPTH-1, cover wrap-around case	Assert property wrap1: wr_ptr wraps to 0 after reaching FIFO_DEPTH-1
WRITE_3	When wr_en = 1 and FIFO is not full, count should increment by 1 & wr_ack arrised after storing data	Generate single writes with no reads	Cover count transitions from 0 to FIFO_DEPTH	Verify count increments correctly
WRITE_4	When wr_en=1 and FIFO is full, overflow flag should be set and write rejected	Generate write attempts when count=FIFO_DEPTH	Cover overflow condition: cross cp_wr_en, cp_full, cp_overflow	Assert overflow=1 when wr_en=1 and full=1
READ_1	When rd_en=1 and FIFO is not empty, data should be read from memory at rd_ptr location & count should be decremented	Randomize rd_en=1 using 30% distribution, ensure FIFO not empty	Cover read operations at different fill levels: full, partial, almost empty	Check that data_out matches expected value from reference queue & verify that count decrement correctly
READ_2	When rd_en=1 and FIFO is not empty, rd_ptr should increment (with wrap-around at FIFO_DEPTH)	Generate consecutive reads until FIFO is empty	Cover rd_ptr increment from 0 to FIFO_DEPTH-1, cover wrap-around case	Assert property rd_ptr wraps to 0 after reaching FIFO_DEPTH-1
READ_3	When rd_en=1 and FIFO is empty, underflow flag should be set and read rejected	Generate read attempts when count=0	Cover underflow condition: cross cp_rd_en, cp_empty, cp_underflow	Assert property underflow=1 when rd_en=1 and empty=1
SIMUL_WR_RD_1	When wr_en=1 and rd_en=1 simultaneously and FIFO is neither empty nor full, count should increment	Generate simultaneous wr_en=1 and rd_en=1 when 0<count<FIFO_DEPTH	Cross coverage: cp_wr_en=1, cp_rd_en=1 with various count values	Verify count remains stable and both operations succeed
SIMUL_WR_RD_2	When wr_en=1 and rd_en=1 simultaneously and FIFO is empty, only write should occur and count should increment	Generate simultaneous wr_en=1 and rd_en=1 when count=0	Cross coverage: cp_wr_en=1, cp_rd_en=1, cp_empty=1	Verify count increments by 1, underflow=0, write succeeds
SIMUL_WR_RD_3	When wr_en=1 and rd_en=1 simultaneously and FIFO is full, only read should occur and count should decrement	Generate simultaneous wr_en=1 and rd_en=1 when count=FIFO_DEPTH	Cross coverage: cp_wr_en=1, cp_rd_en=1, cp_full=1	Verify count decrements by 1, overflow=0, read succeeds
STATUS_FULL	Full flag should be asserted when count equals FIFO_DEPTH	Fill FIFO completely by writing FIFO_DEPTH times without reads	Cover full flag assertion: cp_full=1	Assert property ful: full=1 when count=FIFO_DEPTH
STATUS_EMPTY	Empty flag should be asserted when count equals 0	Empty FIFO completely by reading all entries	Cover empty flag assertion: cp_empty=1	Assert property emp: empty=1 when count=0
STATUS_AFULL	Almost full flag should be asserted when count equals FIFO_DEPTH-1	Write FIFO_DEPTH-1 entries	Cover almost full: cross_wre_rd_afull with count=FIFO_DEPTH-1	Assert property Aful: almostfull=1 when count=FIFO_DEPTH-1
STATUS_AEMPTY	Almost empty flag should be asserted when count equals 1	Read until only 1 entry remains	Cover almost empty: cross_wre_rd_aempty with count=1	Assert property Aemp: almostempty=1 when count=1
WR_ACK	Write acknowledge should be asserted only when wr_en=1 and FIFO is not full	Generate various wr_en patterns with different FIFO states	Cover wr_ack with illegal bins: wr_ack=1 when wr_en=0	Assert property ack: wr_ack=1 follows successful write
DATA_INTEGRITY	Data written to FIFO should match data read out in FIFO order	Write sequence of unique random data, then read back	Cover full range of data_in values (16-bit)	Scoreboard compares data_out with expected values from reference queue
PTR_BOUNDS	Write and read pointers should always remain within valid range [0, FIFO_DEPTH-1]	Generate extensive random sequences over 10000 cycles	Cover boundary cases: wr_ptr and rd_ptr at min/max values	Assert property FIFO_internal_bounds: wr_ptr<FIFO_DEPTH and rd_ptr<FIFO_DEPTH
COUNT_BOUNDS	Count should always remain within valid range [0, FIFO_DEPTH]	Generate random write/read patterns	Cover all count values from 0 to FIFO_DEPTH	Assert property wrap3: 0<=count<=FIFO_DEPTH at all times
STRESS_TEST	FIFO should handle continuous random operations without errors	Run 10000 random cycles with 70% write / 30% read distribution	Achieve 100% coverage on all coverpoints and crosses	Monitor reports correct_count vs error_count, expect 0 errors

3. BUGS found in design

3.1 BUGS in write case :

```
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
```

BUGS fixing:

```
always @(posedge FIFO_DUT.clk or negedge FIFO_DUT.rst_n) begin
    if (!FIFO_DUT.rst_n) begin
        wr_ptr <= 0;
        FIFO_DUT.overflow <= 0;
    end
    else if ((({FIFO_DUT.wr_en , FIFO_DUT.rd_en} == 2'b10) && (count < FIFO_DEPTH))
                || (({FIFO_DUT.wr_en , FIFO_DUT.rd_en} == 2'b11) && (count < FIFO_DEPTH))) begin
        mem[wr_ptr] <= FIFO_DUT.data_in;
        FIFO_DUT.wr_ack <= 1;
        wr_ptr <= wr_ptr + 1;
    end
    else begin
        FIFO_DUT.wr_ack <= 0;
        if (FIFO_DUT.full && FIFO_DUT.wr_en)
            FIFO_DUT.overflow <= 1;
        else
            FIFO_DUT.overflow <= 0;
    end
end
```

- **Simultaneous Read/Write Handling:**
  When both *write enable* (wr_en) and *read enable* (rd_en) are asserted in the same clock cycle, the design gives **priority to the write operation** if the FIFO is **empty**. This ensures that valid data is written into the FIFO before any read operation occurs.
- **Overflow Flag Reset Behavior:**
  The **overflow flag** is **asynchronously cleared** when the **reset signal** is asserted, ensuring that no residual overflow condition persists after system reset.

3.2 BUGS in read case :

```
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
```

```
assign underflow = (empty && rd_en)? 1 : 0;
```

BUGS fixing:

```
always @(posedge FIFO_DUT.clk or negedge FIFO_DUT.rst_n) begin
    if (!FIFO_DUT.rst_n) begin
        rd_ptr <= 0;
        FIFO_DUT.underflow <= 0 ;
    end
    else if ((({FIFO_DUT.wr_en , FIFO_DUT.rd_en} == 2'b01) && count != 0)
                || ( ({FIFO_DUT.wr_en , FIFO_DUT.rd_en} == 2'b11) && count == FIFO_DEPTH) ) begin
        FIFO_DUT.data_out <= mem[rd_ptr];
        rd_ptr <= rd_ptr + 1;
    end
    else begin
        if (FIFO_DUT.empty && FIFO_DUT.rd_en)
            FIFO_DUT.underflow <= 1;
        else
            FIFO_DUT.underflow <= 0;
    end
end
```

- **Simultaneous Read/Write When FIFO Is Full:**
  In the case where both *write enable* (wr_en) and *read enable* (rd_en) are asserted while the FIFO is **full**, the design gives **priority to the read operation**. The read is performed normally, and the **write request is ignored** in that cycle. This prevents data corruption and ensures correct FIFO behavior under concurrent access conditions.
- **Sequential Overflow Flag Implementation:**
  The **overflow flag** has been modified to operate as a **sequential signal**, in accordance with the design specifications.

3.3 Updating counter BUGS :

```
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
```

BUGS fixing:

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

- **Updating the FIFO Internal Count:**
  The FIFO's internal counter is updated to accurately reflect the current number of stored elements while covering **all possible combinations** of *write enable* (wr_en) and *read enable* (rd_en). Priority cases are carefully handled to ensure consistency:

- When both (wr_en) and (rd_en) are asserted simultaneously, **priority rules** determine whether the count is incremented, decremented, or remains unchanged based on FIFO status (full or empty).
- The counter increments on valid write operations, decrements on valid read operations, and remains stable during invalid or simultaneous conflicting operations. This mechanism ensures **precise tracking of FIFO occupancy** under all operational conditions and maintains **synchronization between control flags and stored data**.

3.4 Almostfull flag bug:

```
1  assign almostfull = (count == FIFO_DEPTH-2)? 1 : 0;
```

BUGS fixing:

```
1  assign FIFO_DUT.almostfull = (count == FIFO_DEPTH-1)? 1 : 0;
```

- **Almost-Full Flag Update:**
  The **almost-full flag** is asserted when the **FIFO count reaches one location less than the maximum FIFO depth**, as specified in the design requirements. This ensures that the flag accurately indicates the FIFO is nearing full capacity, allowing preventive control actions before an overflow condition occurs.

4. Fixed DUT & assertions

```
import SHARED_pkg::*;
module FIFO(FIFO_if.DUT FIFO_DUT);
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 FIFO_DUT.clk or negedge FIFO_DUT.rst_n) begin
    if (!FIFO_DUT.rst_n) begin
        wr_ptr <= 0;
        FIFO_DUT.overflow <= 0;
    end
    else if ((({FIFO_DUT.wr_en , FIFO_DUT.rd_en} == 2'b10) && (count < FIFO_DEPTH))
                || (({FIFO_DUT.wr_en , FIFO_DUT.rd_en} == 2'b11) && (count < FIFO_DEPTH))) begin
        mem[wr_ptr] <= FIFO_DUT.data_in;
        FIFO_DUT.wr_ack <= 1;
        wr_ptr <= wr_ptr + 1;
    end
    else begin
        FIFO_DUT.wr_ack <= 0;
        if (FIFO_DUT.full && FIFO_DUT.wr_en)
            FIFO_DUT.overflow <= 1;
        else
            FIFO_DUT.overflow <= 0;
    end
end

always @(posedge FIFO_DUT.clk or negedge FIFO_DUT.rst_n) begin
    if (!FIFO_DUT.rst_n) begin
        rd_ptr <= 0;
        FIFO_DUT.underflow <= 0 ;
    end
    else if ((({FIFO_DUT.wr_en , FIFO_DUT.rd_en} == 2'b01) && count != 0)
                || ( ({FIFO_DUT.wr_en , FIFO_DUT.rd_en} == 2'b11) && count == FIFO_DEPTH) ) begin
        FIFO_DUT.data_out <= mem[rd_ptr];
        rd_ptr <= rd_ptr + 1;
    end
    else begin
        if (FIFO_DUT.empty && FIFO_DUT.rd_en)
            FIFO_DUT.underflow <= 1;
        else
            FIFO_DUT.underflow <= 0;
    end
end
```

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

assign FIFO_DUT.full = (count == FIFO_DEPTH)? 1 : 0;
assign FIFO_DUT.empty = (count == 0)? 1 : 0;
assign FIFO_DUT.almostfull = (count == FIFO_DEPTH-1)? 1 : 0;
assign FIFO_DUT.almostempty = (count == 1)? 1 : 0;


property reset_behave;
    @(posedge FIFO_DUT.clk) (!FIFO_DUT.rst_n) |-> (wr_ptr == 0 && rd_ptr == 0 && count == 0) ;
endproperty

property write_ACK;
    @(posedge FIFO_DUT.clk) (FIFO_DUT.rst_n && FIFO_DUT.wr_en && !FIFO_DUT.full) |=> (FIFO_DUT.wr_ack) ;
endproperty

property overflow_detection;
    @(posedge FIFO_DUT.clk) (FIFO_DUT.rst_n && FIFO_DUT.wr_en && FIFO_DUT.full) |=> (FIFO_DUT.overflow) ;
endproperty

property underflow_detection;
    @(posedge FIFO_DUT.clk) (FIFO_DUT.rst_n && FIFO_DUT.rd_en && FIFO_DUT.empty) |=> (FIFO_DUT.underflow) ;
endproperty

property emptyflag_detection;
    @(posedge FIFO_DUT.clk) (FIFO_DUT.rst_n && count == 0 ) |-> (FIFO_DUT.empty ) ;
endproperty

property fullflag_detection;
    @(posedge FIFO_DUT.clk) (FIFO_DUT.rst_n && count == FIFO_DEPTH ) |-> (FIFO_DUT.full) ;
endproperty

property almostfull_detection;
    @(posedge FIFO_DUT.clk) (FIFO_DUT.rst_n && count == FIFO_DEPTH - 1  ) |-> (FIFO_DUT.almostfull) ;
endproperty

property almostempty_detection;
    @(posedge FIFO_DUT.clk) (FIFO_DUT.rst_n && count == 1  ) |-> (FIFO_DUT.almostempty) ;
endproperty

property wrpointer_wrapground_detection;
    @(posedge FIFO_DUT.clk) (FIFO_DUT.rst_n && wr_ptr == FIFO_DEPTH-1 && FIFO_DUT.wr_en && !FIFO_DUT.full ) |=> (wr_ptr == 0) ;
endproperty

property rdpointer_wrapground_detection;
    @(posedge FIFO_DUT.clk) (FIFO_DUT.rst_n && rd_ptr == FIFO_DEPTH-1 && FIFO_DUT.rd_en && !FIFO_DUT.empty && FIFO_DUT.full ) |=> (rd_ptr == 0) ;
endproperty

property counter_wrapground_detection;
    @(posedge FIFO_DUT.clk) (!FIFO_DUT.rst_n && count == FIFO_DEPTH ) |=> (count == 0) ;
endproperty

property wrpointer_threshold;
  @(posedge FIFO_DUT.clk)
    FIFO_DUT.rst_n |-> (wr_ptr < FIFO_DEPTH);
endproperty

property rdpointer_threshold;
  @(posedge FIFO_DUT.clk)
    FIFO_DUT.rst_n |-> (rd_ptr < FIFO_DEPTH);
```

```
`ifdef SIM
// Reset behavior
assert_reset_behave: assert property(reset_behave)
  else $error("Reset did not clear wr_ptr, rd_ptr, or count!");
cover_reset_behave: cover property(reset_behave);

// Write acknowledge
assert_write_ACK: assert property(write_ACK)
  else $error("Write acknowledge failed when wr_en && !full");
cover_write_ACK: cover property(write_ACK);

// Overflow detection
assert_overflow_detection: assert property(overflow_detection)
  else $error("Overflow not detected when write attempted on full FIFO!");
cover_overflow_detection: cover property(overflow_detection);

// Underflow detection
assert_underflow_detection: assert property(underflow_detection)
  else $error("Underflow not detected when read attempted on empty FIFO!");
cover_underflow_detection: cover property(underflow_detection);

// Empty flag
assert_emptyflag_detection: assert property(emptyflag_detection)
  else $error("Empty flag not asserted when count == 0!");
cover_emptyflag_detection: cover property(emptyflag_detection);

// Full flag
assert_fullflag_detection: assert property(fullflag_detection)
  else $error("Full flag not asserted when count == FIFO_DEPTH!");
cover_fullflag_detection: cover property(fullflag_detection);

// Almost full
assert_almostfull_detection: assert property(almostfull_detection)
  else $error("Almost full flag not asserted when count == FIFO_DEPTH-1!");
cover_almostfull_detection: cover property(almostfull_detection);

// Almost empty
assert_almostempty_detection: assert property(almostempty_detection)
  else $error("Almost empty flag not asserted when count == 1!");
cover_almostempty_detection: cover property(almostempty_detection);

// Write pointer wraparound
assert_wrpointer_wraparound: assert property(wrpointer_wrapground_detection)
  else $error("Write pointer did not wrap around to 0!");
cover_wrpointer_wraparound: cover property(wrpointer_wrapground_detection);

// Read pointer wraparound
assert_rdpointer_wraparound: assert property(rdpointer_wrapground_detection)
  else $error("Read pointer did not wrap around to 0!");
cover_rdpointer_wraparound: cover property(rdpointer_wrapground_detection);

assert_counter_reset: assert property (counter_wrapground_detection)
  else $error("Counter did not reset to 0!");
cover_counter_reset: cover property (counter_wrapground_detection);

assert_wrpointer_threshold: assert property (wrpointer_threshold)
  else $error("Write pointer exceeded FIFO_DEPTH-1!");
cover_wrpointer_threshold: cover property (wrpointer_threshold);

assert_rdpointer_threshold: assert property (rdpointer_threshold)
  else $error("Read pointer exceeded FIFO_DEPTH-1!");
cover_rdpointer_threshold: cover property (rdpointer_threshold);

assert_counter_threshold: assert property (counter_threshold)
  else $error("Counter exceeded FIFO_DEPTH!");
cover_counter_threshold: cover property (counter_threshold);

`endif

endmodule
```

5. Testbench Components

5.1. Interface file

```
interface FIFO_if (clk);
    import SHARED_pkg::*;
    input clk ;
    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 (
        output data_in, rst_n, wr_en, rd_en,
        input  clk, data_out, wr_ack, overflow,
            full, empty, almostfull, almostempty, underflow
    );

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

5.2 Shared package

```
package SHARED_pkg;
    parameter FIFO_WIDTH = 16;
    parameter FIFO_DEPTH = 8 ;
    integer error_count   = 0 ;
    integer correct_count = 0 ;
    bit test_finished ;
endpackage
```

5.3 TOP module

```
import FIFO_transaction_pkg::*;
import FIFO_coverage_pkg::*;
module FIFO_top();
    bit clk ;
    initial begin
        clk = 0;
        forever begin
            #2 clk = ~clk;
        end
    end
    FIFO_if      inst_if (clk);
    FIFO_tb      inst_test (inst_if);
    FIFO         inst_DUT (inst_if);
    FIFO_monitor inst_monitor (inst_if);
endmodule
```

5.4 TEST module

```
import FIFO_transaction_pkg::*;
import SHARED_pkg::*;
module FIFO_tb(FIFO_if.TEST FIFO_tb);
    initial begin
        FIFO_transaction obj = new ;
        FIFO_tb.rst_n = 0 ;
        @(negedge FIFO_tb.clk) ;
        @(negedge FIFO_tb.clk) ;
        FIFO_tb.rst_n = 1 ;
        repeat (9000) begin
            assert (obj.randomize());
            FIFO_tb.data_in = obj.data_in;
            FIFO_tb.wr_en   = obj.wr_en;
            FIFO_tb.rd_en   = obj.rd_en;
            @(negedge FIFO_tb.clk) ;
        end
        test_finished = 1 ;
    end
endmodule
```

5.5 FIFO transaction

```
import SHARED_pkg::*;
package FIFO_transaction_pkg;
    class FIFO_transaction ;
        logic clk ;
        rand logic [SHARED_pkg::FIFO_WIDTH-1:0] data_in;
        rand logic rst_n, wr_en, rd_en;
        logic [SHARED_pkg::FIFO_WIDTH-1:0] data_out;
        logic wr_ack, overflow;
        logic full, empty, almostfull, almostempty, underflow;
        integer RD_EN_ON_DIST , WR_EN_ON_DIST ;

        function new (int a = 40 , int b = 60);
            RD_EN_ON_DIST = a ;
            WR_EN_ON_DIST = b ;
        endfunction

        constraint reset {
            rst_n dist {1:/97 , 0:/3};
        }

        constraint write_enable {
            wr_en dist {1:/WR_EN_ON_DIST , 0:/100-WR_EN_ON_DIST};
        }

        constraint read_enable {
            rd_en dist {1:/RD_EN_ON_DIST , 0:/100-RD_EN_ON_DIST};
        }

    endclass
endpackage
```

5.6 FIFO coverage

```
package FIFO_coverage_pkg;
import FIFO_transaction_pkg::*;
    class FIFO_coverage;
        FIFO_transaction F_cvg_txn = new() ;
        covergroup FIFO_cg;
            write_enable: coverpoint F_cvg_txn.wr_en ;
            read_enable : coverpoint F_cvg_txn.rd_en ;
            write_ack   : coverpoint F_cvg_txn.wr_ack ;
            overflow    : coverpoint F_cvg_txn.overflow ;
            full        : coverpoint F_cvg_txn.full ;
            empty       : coverpoint F_cvg_txn.empty;
            almostfull  : coverpoint F_cvg_txn.almostfull;
            almostempty : coverpoint F_cvg_txn.almostempty;
            underflow   : coverpoint F_cvg_txn.underflow;

            cross write_enable, read_enable, write_ack {
                illegal_bins wr_ack = binsof (write_enable) intersect{0} && binsof (write_ack) intersect{1} ;
            }
            cross write_enable, read_enable, overflow{
                illegal_bins overflow = binsof (write_enable) intersect{0} && binsof (overflow) intersect{1} ;
            }
            cross write_enable, read_enable, full{
                illegal_bins overflow = binsof (write_enable) intersect{0} && binsof (full) intersect{1}
                                        && binsof (read_enable) intersect{1} ;
            }
            cross write_enable, read_enable, empty{
                illegal_bins overflow = binsof (write_enable) intersect{1} && binsof (empty) intersect{1};
            }
            cross write_enable, read_enable, almostfull;
            cross write_enable, read_enable, almostempty;
            cross write_enable, read_enable, underflow{
                illegal_bins underflow = binsof (read_enable) intersect{0} && binsof (underflow) intersect{1} ;
            }

        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
```

5.7 FIFO monitor

```
import FIFO_coverage_pkg::*;
import FIFO_scoreboard_pkg::*;
import FIFO_transaction_pkg::*;
import SHARED_pkg::*;
module FIFO_monitor(FIFO_if.MONITOR FIFO_mon);
    FIFO_transaction F_txn = new();
    FIFO_scoreboard  F_scb = new();
    FIFO_coverage    F_cvg = new();

    initial begin
        forever begin
            @(posedge FIFO_mon.clk);
            #1;
            F_txn.rst_n      = FIFO_mon.rst_n;
            F_txn.wr_en      = FIFO_mon.wr_en;
            F_txn.rd_en      = FIFO_mon.rd_en;
            F_txn.data_in    = FIFO_mon.data_in;
            F_txn.data_out   = FIFO_mon.data_out;
            F_txn.full       = FIFO_mon.full;
            F_txn.empty      = FIFO_mon.empty;
            F_txn.almostfull = FIFO_mon.almostfull;
            F_txn.almostempty= FIFO_mon.almostempty;
            F_txn.overflow   = FIFO_mon.overflow;
            F_txn.underflow  = FIFO_mon.underflow;
            F_txn.wr_ack     = FIFO_mon.wr_ack;

            fork
                F_cvg.sample_data (F_txn) ;
                F_scb.check_data  (F_txn) ;
            join

            if (test_finished) begin
                $display("========================================");
                $display("          The test summary report       ");
                $display("========================================");
                $display("the corect count is %d",correct_count);
                $display("the error count is %d",error_count);
                $display("========================================");
                $stop;
            end
        end
    end
endmodule
```

5.8 FIFO scoreboard

- Task check_data

```
package FIFO_scoreboard_pkg;
import FIFO_transaction_pkg::*;
import SHARED_pkg::*;
    class FIFO_scoreboard;
        logic [15:0] data_out_ref;
        logic wr_ack_ref, overflow_ref;
        logic full_ref, empty_ref, almostfull_ref, almostempty_ref, underflow_ref;

        logic [2:0] wr_ptr_test , rd_ptr_test  ;
        logic [FIFO_WIDTH-1:0] test_mem[$];
        logic [3:0] count_test ;

        task check_data (input FIFO_transaction txn);
            refrence_model(txn) ;
            if (txn.data_out   !== data_out_ref   ||
                txn.full       !== full_ref       ||
                txn.empty      !== empty_ref      ||
                txn.almostfull !== almostfull_ref ||
                txn.almostempty!== almostempty_ref||
                txn.overflow   !== overflow_ref   ||
                txn.underflow  !== underflow_ref  ||
                txn.wr_ack     !== wr_ack_ref) begin
                    error_count = error_count + 1 ;
                $display("test fail check the problem");
                end
            else begin
                $display("test pass");
                correct_count = correct_count + 1 ;
            end
        endtask
```

- **Reference model**

```
        task refrence_model (input FIFO_transaction txn);
            if (!txn.rst_n) begin
                wr_ptr_test  = 0 ;
                rd_ptr_test  = 0 ;
                count_test   = 0 ;
                test_mem.delete();
                overflow_ref = 0 ;
                underflow_ref = 0 ;
            end
            else begin
                if (txn.wr_en && txn.rd_en && !full_ref) begin
                    test_mem [wr_ptr_test] = txn.data_in ;
                    wr_ptr_test = wr_ptr_test + 1 ;
                    count_test  = count_test + 1 ;
                    wr_ack_ref  = 1 ;
                end
                else if (txn.wr_en && txn.rd_en && full_ref) begin
                    data_out_ref = test_mem[rd_ptr_test];
                    count_test   = count_test  - 1 ;
                    rd_ptr_test  = rd_ptr_test + 1 ;
                    wr_ack_ref   = 0;
                    if (full_ref && txn.wr_en ) overflow_ref = 1 ;
                    else overflow_ref = 0  ;
                end
                else begin
                    if (txn.wr_en && !txn.rd_en && !full_ref) begin
                        test_mem [wr_ptr_test] = txn.data_in ;
                        wr_ptr_test = wr_ptr_test + 1 ;
                        count_test  = count_test + 1 ;
                        wr_ack_ref  = 1 ;
                    end
                    else begin
                        wr_ack_ref = 0 ;
                        if (full_ref && txn.wr_en ) overflow_ref = 1 ;
                        else overflow_ref = 0  ;
                    end
                    if (!txn.wr_en && txn.rd_en && !empty_ref) begin
                        data_out_ref = test_mem[rd_ptr_test];
                        count_test   = count_test - 1 ;
                        rd_ptr_test  = rd_ptr_test + 1 ;
                    end
                end
                if (empty_ref && txn.rd_en ) underflow_ref = 1 ;
                else underflow_ref = 0  ;
            end

            if (count_test == FIFO_DEPTH) full_ref = 1 ;
            else full_ref = 0 ;

            if (count_test == 0) empty_ref = 1 ;
            else empty_ref = 0 ;

            if (count_test == FIFO_DEPTH - 1 ) almostfull_ref = 1 ;
            else almostfull_ref = 0 ;

            if (count_test == 1) almostempty_ref = 1 ;
            else almostempty_ref = 0 ;
         endtask
     endclass
endpackage
```

5.9 DO file

```
vlib work
vlog +define+SIM +cover *.sv
vsim -voptargs=+acc FIFO_top -cover
add wave *
coverage save FIFO.ucdb -onexit
run -all
```

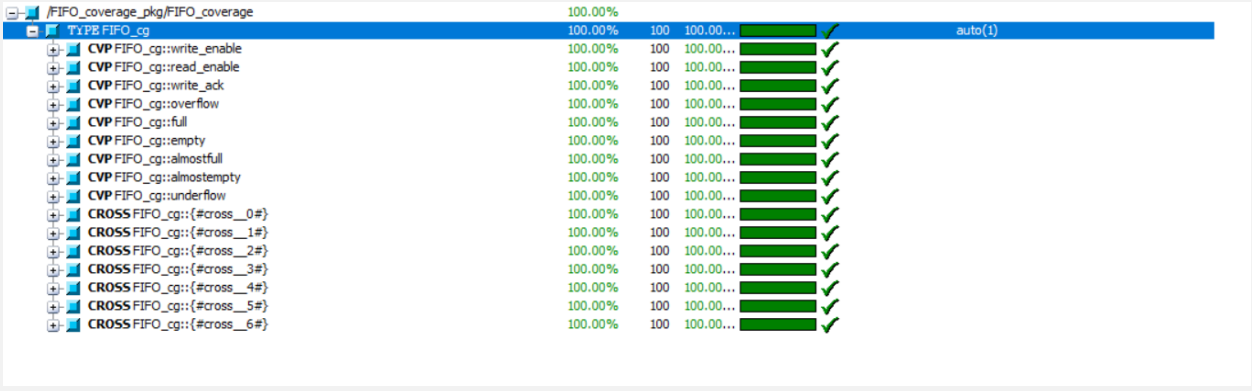
6. Simulation result

6.1 Assertion coverage

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threads
/FIFO_top/inst_DUT/cover_reset_behave	SVA	✓	Off	1	1	Uni...	1	100%		✓	0	0	0 ns	0
/FIFO_top/inst_DUT/cover_write_ACK	SVA	✓	Off	25872	1	Uni...	1	100%		✓	0	0	0 ns	0
/FIFO_top/inst_DUT/cover_overflow_detection	SVA	✓	Off	28046	1	Uni...	1	100%		✓	0	0	0 ns	0
/FIFO_top/inst_DUT/cover_underflow_detection	SVA	✓	Off	1	1	Uni...	1	100%		✓	0	0	0 ns	0
/FIFO_top/inst_DUT/cover_emptyflag_detection	SVA	✓	Off	3	1	Uni...	1	100%		✓	0	0	0 ns	0
/FIFO_top/inst_DUT/cover_fullflag_detection	SVA	✓	Off	46585	1	Uni...	1	100%		✓	0	0	0 ns	0
/FIFO_top/inst_DUT/cover_almostfull_detection	SVA	✓	Off	31660	1	Uni...	1	100%		✓	0	0	0 ns	0
/FIFO_top/inst_DUT/cover_almostempty_detection	SVA	✓	Off	13	1	Uni...	1	100%		✓	0	0	0 ns	0
/FIFO_top/inst_DUT/cover_wrpointer_wraparound	SVA	✓	Off	3234	1	Uni...	1	100%		✓	0	0	0 ns	0
/FIFO_top/inst_DUT/cover_rdpointer_wraparound	SVA	✓	Off	2339	1	Uni...	1	100%		✓	0	0	0 ns	0
/FIFO_top/inst_DUT/cover_wrpointer_threshold	SVA	✓	Off	90001	1	Uni...	1	100%		✓	0	0	0 ns	0
/FIFO_top/inst_DUT/cover_rdpointer_threshold	SVA	✓	Off	90001	1	Uni...	1	100%		✓	0	0	0 ns	0
/FIFO_top/inst_DUT/cover_counter_threshold	SVA	✓	Off	90001	1	Uni...	1	100%		✓	0	0	0 ns	0

```
# TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 13
#
# ASSERTION RESULTS:
# ---------------------------------------------------------------------
# Name                File(Line)                    Failure      Pass
#                                                   Count        Count
# ---------------------------------------------------------------------
# /FIFO_top/inst_test/#ublk#182146786#4/#ublk#182146786#10/immed__11
#                     FIFO_tb.sv(11)                  0            1
# /FIFO_top/inst_DUT/assert_reset_behave
#                     FIFO.sv(127)                    0            1
# /FIFO_top/inst_DUT/assert_write_ACK
#                     FIFO.sv(132)                    0            1
# /FIFO_top/inst_DUT/assert_overflow_detection
#                     FIFO.sv(137)                    0            1
# /FIFO_top/inst_DUT/assert_underflow_detection
#                     FIFO.sv(142)                    0            1
# /FIFO_top/inst_DUT/assert_emptyflag_detection
#                     FIFO.sv(147)                    0            1
# /FIFO_top/inst_DUT/assert_fullflag_detection
#                     FIFO.sv(152)                    0            1
# /FIFO_top/inst_DUT/assert_almostfull_detection
#                     FIFO.sv(157)                    0            1
# /FIFO_top/inst_DUT/assert_almostempty_detection
#                     FIFO.sv(162)                    0            1
# /FIFO_top/inst_DUT/assert_wrpointer_wraparound
#                     FIFO.sv(167)                    0            1
# /FIFO_top/inst_DUT/assert_rdpointer_wraparound
#                     FIFO.sv(172)                    0            1
# /FIFO_top/inst_DUT/assert_wrpointer_threshold
#                     FIFO.sv(176)                    0            1
# /FIFO_top/inst_DUT/assert_rdpointer_threshold
#                     FIFO.sv(180)                    0            1
# /FIFO_top/inst_DUT/assert_counter_threshold
#                     FIFO.sv(184)                    0            1
```

6.2 Covergroup coverage:



/FIFO_coverage_pkg/FIFO_coverage	100.00%			
TYPE FIFO_cg	100.00%	100	100.00...	auto(1)
CVP FIFO_cg::write_enable	100.00%	100	100.00...	
CVP FIFO_cg::read_enable	100.00%	100	100.00...	
CVP FIFO_cg::write_ack	100.00%	100	100.00...	
CVP FIFO_cg::overflow	100.00%	100	100.00...	
CVP FIFO_cg::full	100.00%	100	100.00...	
CVP FIFO_cg::empty	100.00%	100	100.00...	
CVP FIFO_cg::almostfull	100.00%	100	100.00...	
CVP FIFO_cg::almostempty	100.00%	100	100.00...	
CVP FIFO_cg::underflow	100.00%	100	100.00...	
CROSS FIFO_cg::{#cross__0#}	100.00%	100	100.00...	
CROSS FIFO_cg::{#cross__1#}	100.00%	100	100.00...	
CROSS FIFO_cg::{#cross__2#}	100.00%	100	100.00...	
CROSS FIFO_cg::{#cross__3#}	100.00%	100	100.00...	
CROSS FIFO_cg::{#cross__4#}	100.00%	100	100.00...	
CROSS FIFO_cg::{#cross__5#}	100.00%	100	100.00...	
CROSS FIFO_cg::{#cross__6#}	100.00%	100	100.00...	

```
# COVERGROUP COVERAGE:
# ----------------------------------------------------------------------------------------------------------------
# Covergroup                                          Metric       Goal       Bins    Status
#
# ----------------------------------------------------------------------------------------------------------------
#  TYPE /FIFO_coverage_pkg/FIFO_coverage/FIFO_cg      100.00%        100          -    Covered
#     covered/total bins:                                 65         65          -
#     missing/total bins:                                  0         65          -
#     % Hit:                                         100.00%        100          -
#     Coverpoint write_enable                        100.00%        100          -    Covered
#         covered/total bins:                              2          2          -
#         missing/total bins:                              0          2          -
#         % Hit:                                     100.00%        100          -
#         bin auto[0]                                  35860          1          -    Covered
#         bin auto[1]                                  54141          1          -    Covered
#     Coverpoint read_enable                         100.00%        100          -    Covered
#         covered/total bins:                              2          2          -
#         missing/total bins:                              0          2          -
#         % Hit:                                     100.00%        100          -
#         bin auto[0]                                  54007          1          -    Covered
#         bin auto[1]                                  35994          1          -    Covered
#     Coverpoint write_ack                           100.00%        100          -    Covered
#         covered/total bins:                              2          2          -
#         missing/total bins:                              0          2          -
#         % Hit:                                     100.00%        100          -
#         bin auto[0]                                  64312          1          -    Covered
#         bin auto[1]                                  25689          1          -    Covered
#     Coverpoint overflow                            100.00%        100          -    Covered
#         covered/total bins:                              2          2          -
#         missing/total bins:                              0          2          -
#         % Hit:                                     100.00%        100          -
#         bin auto[0]                                  54798          1          -    Covered
#         bin auto[1]                                  35204          1          -    Covered
#     Coverpoint full                                100.00%        100          -    Covered
#         covered/total bins:                              2          2          -
#         missing/total bins:                              0          2          -
#         % Hit:                                     100.00%        100          -
#         bin auto[0]                                  42839          1          -    Covered
#         bin auto[1]                                  47163          1          -    Covered
#     Coverpoint empty                               100.00%        100          -    Covered
#         covered/total bins:                              2          2          -
#         missing/total bins:                              0          2          -
#         % Hit:                                     100.00%        100          -
#         bin auto[0]                                  89980          1          -    Covered
#         bin auto[1]                                     22          1          -    Covered
```

```
#     Coverpoint almostempty                          100.00%        100          -    Covered
#         covered/total bins:                               2          2          -
#         missing/total bins:                               0          2          -
#         % Hit:                                      100.00%        100          -
#         bin auto[0]                                   89966          1          -    Covered
#         bin auto[1]                                      36          1          -    Covered
#     Coverpoint underflow                            100.00%        100          -    Covered
#         covered/total bins:                               2          2          -
#         missing/total bins:                               0          2          -
#         % Hit:                                      100.00%        100          -
#         bin auto[0]                                   89987          1          -    Covered
#         bin auto[1]                                      15          1          -    Covered
#     Cross #cross__0#                                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]>             10305          1          -    Covered
#             bin <auto[1],auto[0],auto[1]>             15384          1          -    Covered
#             bin <auto[1],auto[1],auto[0]>             11233          1          -    Covered
#             bin <auto[0],auto[1],auto[0]>             14456          1          -    Covered
#             bin <auto[1],auto[0],auto[0]>             17219          1          -    Covered
#             bin <auto[0],auto[0],auto[0]>             21404          1          -    Covered
#         Illegal and Ignore Bins:
#             illegal_bin wr_ack                            0                     -    ZERO
#     Cross #cross__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]>             13965          1          -    Covered
#             bin <auto[1],auto[0],auto[1]>             21239          1          -    Covered
#             bin <auto[1],auto[1],auto[0]>              7573          1          -    Covered
#             bin <auto[0],auto[1],auto[0]>             14456          1          -    Covered
#             bin <auto[1],auto[0],auto[0]>             11364          1          -    Covered
#             bin <auto[0],auto[0],auto[0]>             21404          1          -    Covered
#         Illegal and Ignore Bins:
#             illegal_bin overflow                          0                     -    ZERO
```

```
#     Cross #cross__2#                                100.00%        100          -    Covered
#         covered/total bins:                               7          7          -
#         missing/total bins:                               0          7          -
#         % Hit:                                      100.00%        100          -
#         Auto, Default and User Defined Bins:
#             bin <auto[1],auto[1],auto[1]>              7550          1          -    Covered
#             bin <auto[1],auto[1],auto[0]>             13988          1          -    Covered
#             bin <auto[0],auto[1],auto[0]>             14456          1          -    Covered
#             bin <auto[1],auto[0],auto[1]>             28536          1          -    Covered
#             bin <auto[1],auto[0],auto[0]>              4067          1          -    Covered
#             bin <auto[0],auto[0],auto[1]>             11077          1          -    Covered
#             bin <auto[0],auto[0],auto[0]>             10327          1          -    Covered
#         Illegal and Ignore Bins:
#             illegal_bin overflow                          0                     -    ZERO
#     Cross #cross__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[0],auto[1],auto[1]>                17          1          -    Covered
#             bin <auto[0],auto[0],auto[1]>                 4          1          -    Covered
#             bin <auto[1],auto[1],auto[0]>             21538          1          -    Covered
#             bin <auto[0],auto[1],auto[0]>             14439          1          -    Covered
#             bin <auto[1],auto[0],auto[0]>             32603          1          -    Covered
#             bin <auto[0],auto[0],auto[0]>             21400          1          -    Covered
#         Illegal and Ignore Bins:
#             illegal_bin overflow                          0                     -    ZERO
#     Cross #cross__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]>             13291          1          -    Covered
#             bin <auto[0],auto[1],auto[1]>              7633          1          -    Covered
#             bin <auto[1],auto[0],auto[1]>              2952          1          -    Covered
#             bin <auto[0],auto[0],auto[1]>              7601          1          -    Covered
#             bin <auto[1],auto[1],auto[0]>              8247          1          -    Covered
#             bin <auto[0],auto[1],auto[0]>              6823          1          -    Covered
#             bin <auto[1],auto[0],auto[0]>             29651          1          -    Covered
#             bin <auto[0],auto[0],auto[0]>             13803          1          -    Covered
```

```
#     Cross #cross__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]>                   6          1          -    Covered
#             bin <auto[0],auto[1],auto[1]>                  16          1          -    Covered
#             bin <auto[1],auto[0],auto[1]>                   4          1          -    Covered
#             bin <auto[0],auto[0],auto[1]>                  10          1          -    Covered
#             bin <auto[1],auto[1],auto[0]>               21532          1          -    Covered
#             bin <auto[0],auto[1],auto[0]>               14440          1          -    Covered
#             bin <auto[1],auto[0],auto[0]>               32599          1          -    Covered
#             bin <auto[0],auto[0],auto[0]>               21394          1          -    Covered
#     Cross #cross__6#                                  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]>                   6          1          -    Covered
#             bin <auto[1],auto[1],auto[0]>               21532          1          -    Covered
#             bin <auto[0],auto[1],auto[1]>                   9          1          -    Covered
#             bin <auto[0],auto[1],auto[0]>               14447          1          -    Covered
#             bin <auto[1],auto[0],auto[0]>               32603          1          -    Covered
#             bin <auto[0],auto[0],auto[0]>               21404          1          -    Covered
#         Illegal and Ignore Bins:
#             illegal_bin underflow                           0                     -    ZERO
#
# TOTAL COVERGROUP COVERAGE: 100.00%  COVERGROUP TYPES: 1
```

6.3 code coverage

```
FIFO.sv
        11 if (!FIFO_DUT.rst_n) begin
        15 else if ((({FIFO_DUT.wr_en , FIFO_DUT.rd_en} == 2'b10) && (count < FIFO_DEPTH))
        21 else begin
        23 if (FIFO_DUT.full && FIFO_DUT.wr_en)
        25 else
        31 if (!FIFO_DUT.rst_n) begin
        35 else if ((({FIFO_DUT.wr_en , FIFO_DUT.rd_en} == 2'b01) && count != 0)
        40 else begin
        41 if (FIFO_DUT.empty && FIFO_DUT.rd_en)
        43 else
        49 if (!FIFO_DUT.rst_n) begin
        52 else begin
        53 if       ( ({FIFO_DUT.wr_en , FIFO_DUT.rd_en} == 2'b10) && !FIFO_DUT.full)
        55 else if ( ({FIFO_DUT.wr_en , FIFO_DUT.rd_en} == 2'b01) && !FIFO_DUT.empty)
        57 else if ( ({FIFO_DUT.wr_en , FIFO_DUT.rd_en} == 2'b11) && !FIFO_DUT.full)
        59 else if ( ({FIFO_DUT.wr_en , FIFO_DUT.rd_en} == 2'b11) && FIFO_DUT.full)
        64 assign FIFO_DUT.full = (count == FIFO_DEPTH)? 1 : 0;
        65 assign FIFO_DUT.empty = (count == 0)? 1 : 0;
        66 assign FIFO_DUT.almostfull = (count == FIFO_DEPTH-1)? 1 : 0;
        67 assign FIFO_DUT.almostempty = (count == 1)? 1 : 0;
```

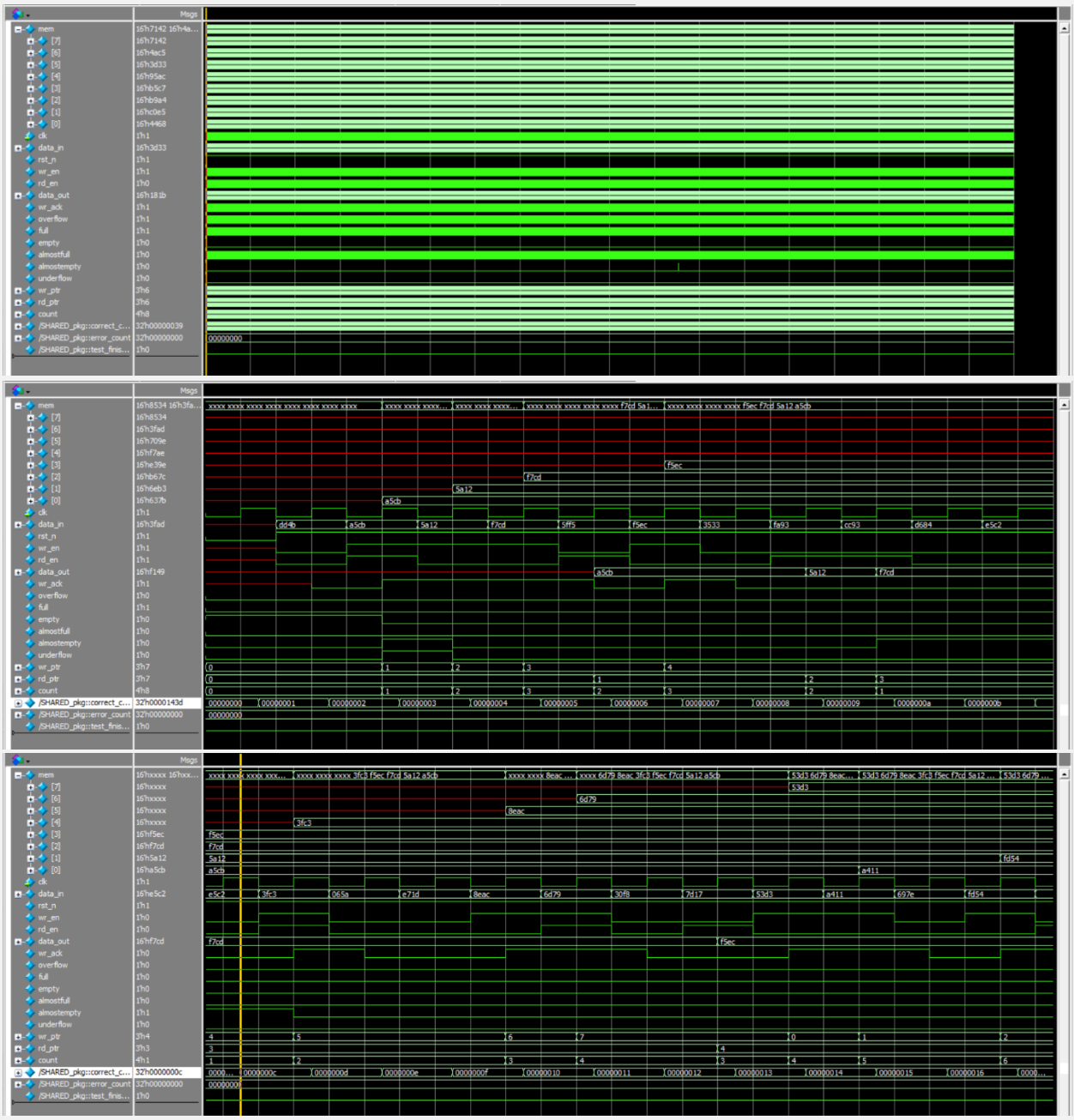
```
# Branch Coverage:
#     Enabled Coverage              Bins      Hits    Misses  Coverage
#     ----------------              ----      ----    ------  --------
#     Branches                        25        25         0   100.00%
#
# ================================Branch Details================================
#
# Branch Coverage for instance /FIFO_top/inst_DUT
#
#     Line         Item                      Count     Source
#     ----         ----                      -----     ------
#   File FIFO.sv
# ------------------------------------IF Branch------------------------------------
#     11                                    90003     Count coming in to IF
#     11             1                          2      if (!FIFO_DUT.rst_n) begin
#
#     15             1                      25689      else if ((({FIFO_DUT.wr_en , FIFO_DUT.rd_en} == 2'b10) && (count < FIFO_DEPTH))
#
#     21             1                      64312      else begin
#
# Branch totals: 3 hits of 3 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     23                                    64312     Count coming in to IF
#     23             1                      28452       if (FIFO_DUT.full && FIFO_DUT.wr_en)
#
#     25             1                      35860       else
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     31                                    90003     Count coming in to IF
#     31             1                          2      if (!FIFO_DUT.rst_n) begin
#
#     35             1                      25681      else if ((({FIFO_DUT.wr_en , FIFO_DUT.rd_en} == 2'b01) && count != 0)
#
#     40             1                      64320      else begin
#
# Branch totals: 3 hits of 3 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     41                                    64320     Count coming in to IF
#     41             1                         14       if (FIFO_DUT.empty && FIFO_DUT.rd_en)
#
#     43             1                      64306       else
#
# Branch totals: 2 hits of 2 branches = 100.00%
```

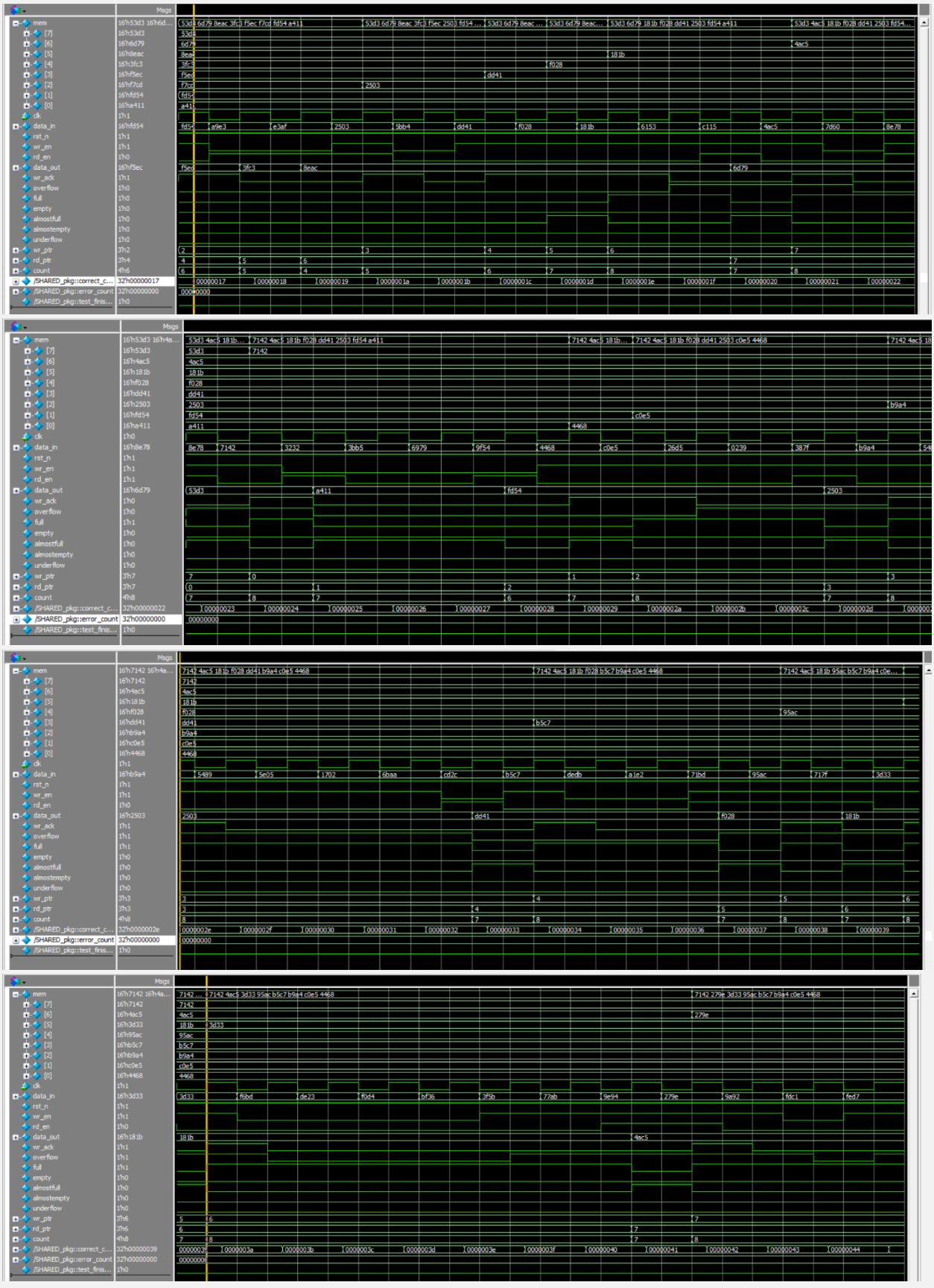
```
# ------------------------------------IF Branch------------------------------------
#     49                                    78676     Count coming in to IF
#     49             1                          2      if (!FIFO_DUT.rst_n) begin
#
#     52             1                      78674      else begin
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     53                                    78674     Count coming in to IF
#     53             1                      15384       if ( ({FIFO_DUT.wr_en , FIFO_DUT.rd_en} == 2'b10) && !FIFO_DUT.full)
#
#     55             1                      14448       else if ( ({FIFO_DUT.wr_en , FIFO_DUT.rd_en} == 2'b01) && !FIFO_DUT.empty)
#
#     57             1                      10305       else if ( ({FIFO_DUT.wr_en , FIFO_DUT.rd_en} == 2'b11) && !FIFO_DUT.full)
#
#     59             1                      11233       else if ( ({FIFO_DUT.wr_en , FIFO_DUT.rd_en} == 2'b11) && FIFO_DUT.full)
#
#                                           27304     All False Count
# Branch totals: 5 hits of 5 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     64                                    51371     Count coming in to IF
#     64             1                      18867     assign FIFO_DUT.full = (count == FIFO_DEPTH)? 1 : 0;
#
#     64             2                      32504     assign FIFO_DUT.full = (count == FIFO_DEPTH)? 1 : 0;
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     65                                    51371     Count coming in to IF
#     65             1                         10     assign FIFO_DUT.empty = (count == 0)? 1 : 0;
#
#     65             2                      51361     assign FIFO_DUT.empty = (count == 0)? 1 : 0;
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
# ------------------------------------IF Branch------------------------------------
#     66                                    51371     Count coming in to IF
#     66             1                      23876     assign FIFO_DUT.almostfull = (count == FIFO_DEPTH-1)? 1 : 0;
#
#     66             2                      27495     assign FIFO_DUT.almostfull = (count == FIFO_DEPTH-1)? 1 : 0;
#
# Branch totals: 2 hits of 2 branches = 100.00%
```

```
#
# ------------------------------------IF Branch------------------------------------
#     67                                    51371     Count coming in to IF
#     67             1                         26     assign FIFO_DUT.almostempty = (count == 1)? 1 : 0;
#
#     67             2                      51345     assign FIFO_DUT.almostempty = (count == 1)? 1 : 0;
#
# Branch totals: 2 hits of 2 branches = 100.00%
#
```

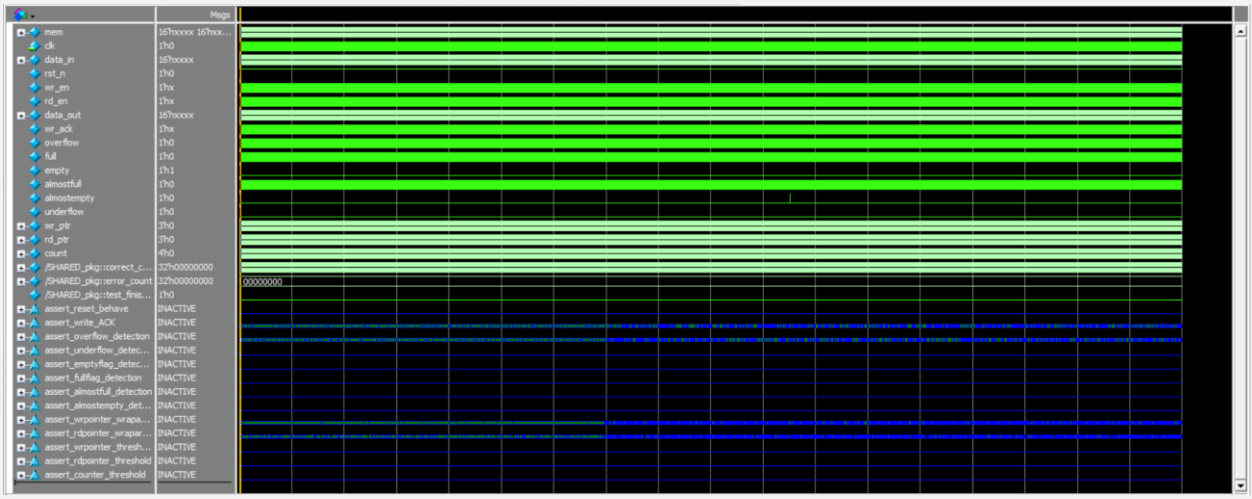
6.4 Questasim waveform

- Snippets show the data flow in side the FIFO




Msgs
mem
clk
data_in
rst_n
wr_en
rd_en
data_out
wr_ack
overflow
full
empty
almostfull
almostempty
underflow
wr_ptr
rd_ptr
count
/SHARED_pkg::correct_c...
/SHARED_pkg::error_count
/SHARED_pkg::test_finis...

- **Snippet showing no assertion fails**



6.5 Transcript summary message

```
# ============================================
#            The test summary report
# ============================================
# the corect count is       90002
# the error count is            0
# ============================================
# ** Note: $stop    : FIFO_monitor.sv(39)
```