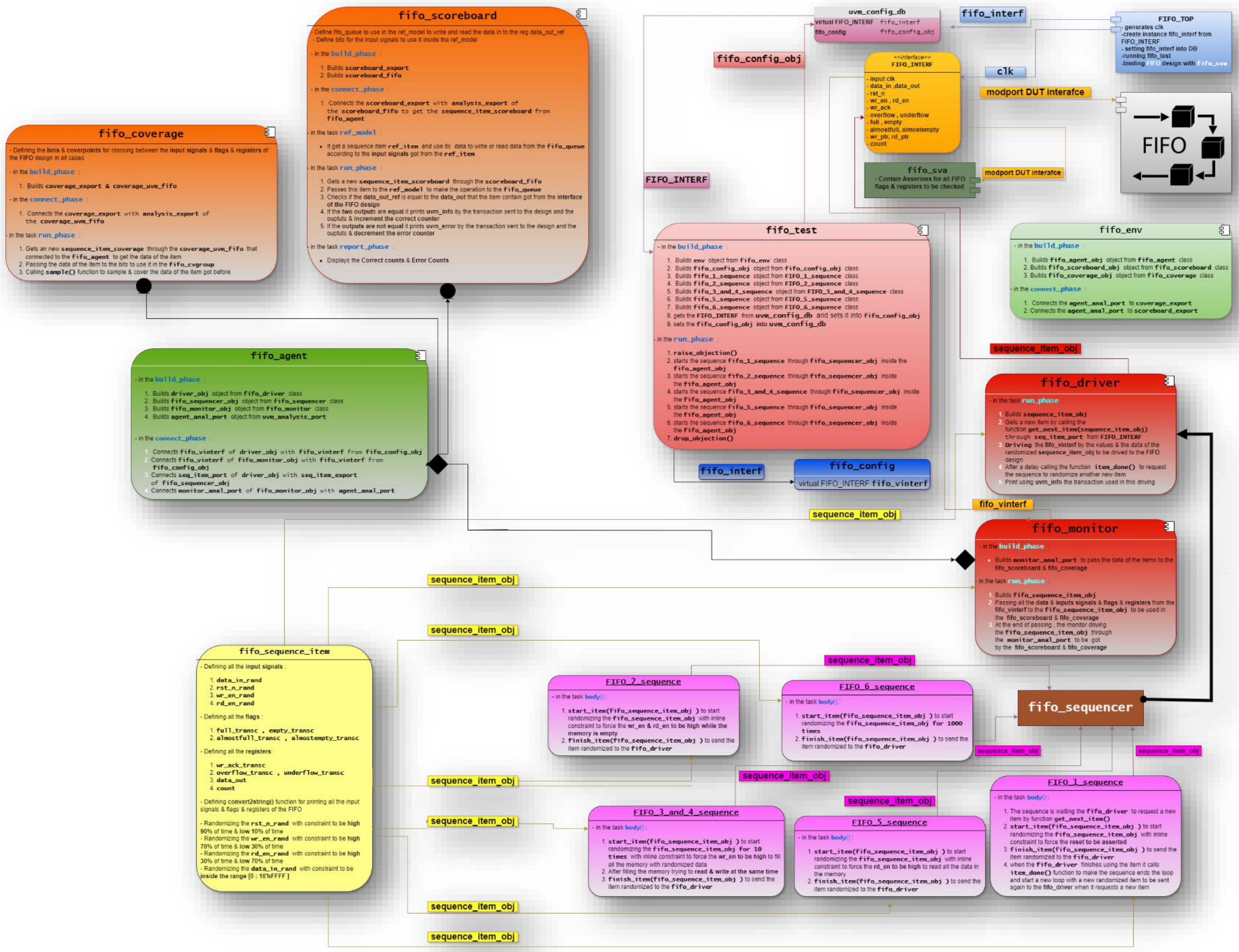
UVM Synch. FIFO

Shehab ElDin Tarek Foaad Mohammed
DIGITAL VERIFICATION DIPLOMA | ENG/KAREEM WASEEM

➔UVM Testbench Flow with UVM Structure:



fifo_scoreboard

- Define fifo_queue to use in the ref_model to write and read the data in to the reg data_out_ref
- Define bits for the input signals to use it inside the ref_model

- in the build_phase :

1. Builds **scoreboard_export**
2. Builds **scoreboard_fifo**

- in the connect_phase :

1. Connects the **scoreboard_export** with **analysis_export** of the **scoreboard_fifo** to get the **sequence_item_scoreboard** from **fifo_agent**

- in the task ref_model :

- It get a sequence item **ref_item** and use its data to write or read data from the **fifo_queue** according to the **input signals** got from the **ref_item**

- in the task run_phase :

1. Gets a new **sequence_item_scoreboard** through the **scoreboard_fifo**
2. Passes this item to the **ref_model** to make the operation to the **fifo_queue**
3. Checks if the **data_out_ref** is equal to the **data_out** that the item contain got from the **interface of the FIFO design**
4. If the **two outputs are equal** it prints **uvm_info** by the transaction sent to the design and the ouptuts & increment the correct counter
5. If the **outputs are not equal** it prints **uvm_error** by the transaction sent to the design and the ouptuts & decrement the error counter

- in the task report_phase :

- Displays the Correct counts & Error Counts

fifo_coverage

- Defining the **bins** & **coverpoints** for crossing between the **input signals** & **flags** & **registers** of the FIFO design in all cases

- in the build_phase :

1. Builds **coverage_export & coverage_uvm_fifo**

- in the connect_phase :

1. Connects the **coverage_export** with **analysis_export** of the **coverage_uvm_fifo**

- in the task run_phase :

1. Gets an new **sequence_item_coverage** through the **coverage_uvm_fifo** that connected to the **fifo_agent** to get the data of the item
2. Passing the data of the item to the bits to use it in the **fifo_cvgroup**
3. Calling **sample()** function to sample & cover the data of the item got before

<<interface>> FIFO_INTERF

- input clk
- data_in ,data_out
- rst_n
- wr_en , rd_en
- wr_ack
- overflow , underflow
- full , empty
- almostfull, almostempty
- wr_ptr, rd_ptr
- count

FIFO_TOP

- generates clk
- create instance fifo_interf from FIFO_INTERF
- setting fifo_interf into DB
- running fifo_test
- binding FIFO design with fifo_sva

fifo_sva

- Contain Asserions for all FIFO flags & registers to be checked

fifo_test

- in the build_phase :

1. Builds **env** object from **fifo_env** class
2. Builds **fifo_config_obj** object from **fifo_config_obj** class
3. Builds **fifo_1_sequence** object from **FIFO_1_sequence** class
4. Builds **fifo_2_sequence** object from **FIFO_2_sequence** class
5. Builds **fifo_3_and_4_sequence** object from **FIFO_3_and_4_sequence** class
6. Builds **fifo_5_sequence** object from **FIFO_5_sequence** class
7. Builds **fifo_6_sequence** object from **FIFO_6_sequence** class
8. gets the **FIFO_INTERF** from **uvm_config_db** and sets it into **fifo_config_obj**
9. sets the **fifo_config_obj** into **uvm_config_db**

- in the run_phase :

1. **raise_objection()**
2. starts the sequence **fifo_1_sequence** through **fifo_sequencer_obj** inside the **fifo_agent_obj**
3. starts the sequence **fifo_2_sequence** through **fifo_sequencer_obj** inside the **fifo_agent_obj**
4. starts the sequence **fifo_3_and_4_sequence** through **fifo_sequencer_obj** inside the **fifo_agent_obj**
5. starts the sequence **fifo_5_sequence** through **fifo_sequencer_obj** inside the **fifo_agent_obj**
6. starts the sequence **fifo_6_sequence** through **fifo_sequencer_obj** inside the **fifo_agent_obj**
7. **drop_objection()**

fifo_env

- in the build_phase :

1. Builds **fifo_agent_obj** object from **fifo_agent** class
2. Builds **fifo_scoreboard_obj** object from **fifo_scoreboard** class
3. Builds **fifo_coverage_obj** object from **fifo_coverage** class

- in the connect_phase :

1. Connects the **agent_anal_port** to **coverage_export**
2. Connects the **agent_anal_port** to **scoreboard_export**

fifo_agent

- in the build_phase :

1. Builds **driver_obj** object from **fifo_driver** class
2. Builds **fifo_sequencer_obj** object from **fifo_sequencer** class
3. Builds **fifo_monitor_obj** object from **fifo_monitor** class
4. Builds **agent_anal_port** object from **uvm_analysis_port**

- in the connect_phase :

1. Connects **fifo_vinterf** of **driver_obj** with **fifo_vinterf** from **fifo_config_obj**
2. Connects **fifo_vinterf** of **fifo_monitor_obj** with **fifo_vinterf** from **fifo_config_obj**
3. Connects **seq_item_port** of **driver_obj** with **seq_item_export** of **fifo_sequencer_obj**
4. Connects **monitor_anal_port** of **fifo_monitor_obj** with **agent_anal_port**

fifo_driver

- in the task run_phase :

1. Builds **sequence_item_obj**
2. Gets a new item by calling the function **get_next_item(sequence_item_obj)** through **seq_item_port** from **FIFO_INTERF**
3. **Driving** the **fifo_vinterf** by the values & the data of the randomized **sequence_item_obj** to be drived to the FIFO design
4. After a delay calling the function **item_done()** to request the sequence to randomize another new item
5. Print using **uvm_info** the transaction used in this driving

fifo_monitor

- in the build_phase :

- Builds **monitor_anal_port** to pass the data of the items to the **fifo_scoreboard** & **fifo_coverage**

- in the task run_phase :

1. Builds **fifo_sequence_item_obj**
2. Passing all the data & **inputs signals** & **flags** & **registers** from the fifo_vinterf to the **fifo_sequence_item_obj** to be used in the **fifo_scoreboard** & **fifo_coverage**
3. At the end of passing , the monitor driving the **fifo_sequence_item_obj** through the **monitor_anal_port** to be got by the **fifo_scoreboard** & **fifo_coverage**

fifo_sequence_item

- Defining all the **input signals** :

1. **data_in_rand**
2. **rst_n_rand**
3. **wr_en_rand**
4. **rd_en_rand**

- Defining all the **flags** :

1. **full_transc , empty_transc**
2. **almostfull_transc , almostempty_transc**

- Defining all the **registers**:

1. **wr_ack_transc**
2. **overflow_transc , underflow_transc**
3. **data_out**
4. **count**

- Defining **convert2string()** function for printing all the input signals & flags & registers of the FIFO

- Randomizing the **rst_n_rand** with constraint to be **high** 90% of time & low 10% of time
- Randomizing the **wr_en_rand** with constraint to be **high** 70% of time & low 30% of time
- Randomizing the **rd_en_rand** with constraint to be **high** 30% of time & low 70% of time
- Randomizing the **data_in_rand** with constraint to be inside the range [0 : 16'hFFFF]

FIFO_2_sequence

- in the task body() :

1. **start_item(fifo_sequence_item_obj)** to start randomizing the **fifo_sequence_item_obj** with inline constraint to force the wr_en & rd_en to be high while the memory is empty
2. **finish_item(fifo_sequence_item_obj)** to send the item randomized to the **fifo_driver**

FIFO_6_sequence

- in the task body() :

1. **start_item(fifo_sequence_item_obj)** to start randomizing the **fifo_sequence_item_obj for 1000 times**
2. **finish_item(fifo_sequence_item_obj)** to send the item randomized to the **fifo_driver**

FIFO_3_and_4_sequence

- in the task body() :

1. **start_item(fifo_sequence_item_obj)** to start randomizing the **fifo_sequence_item_obj for 10 times** with inline constraint to force the wr_en to be high to fill all the memory with randomized data
2. After filling the memory trying to **read & write at the same time**
3. **finish_item(fifo_sequence_item_obj)** to send the item randomized to the **fifo_driver**

FIFO_5_sequence

- in the task body() :

1. **start_item(fifo_sequence_item_obj)** to start randomizing the **fifo_sequence_item_obj** with inline constraint to force the rd_en to be high to read all the data in the memory
2. **finish_item(fifo_sequence_item_obj)** to send the item randomized to the **fifo_driver**

FIFO_1_sequence

- in the task body() :

1. The sequence is waiting the **fifo_driver** to request a new item by function **get_next_item()**
2. **start_item(fifo_sequence_item_obj)** to start randomizing the **fifo_sequence_item_obj** with inline constraint to force the reset to be asserted
3. **finish_item(fifo_sequence_item_obj)** to send the item randomized to the **fifo_driver**
4. when the **fifo_driver** finishes using the item it calls **item_done()** function to make the sequence ends the loop and start a new loop with a new randomized item to be sent again to the fifo_driver when it requests a new item

UVM Testbench Flow with UVM Structure Figure

➔Bug Report :

The Bugs noticed in the Design RTL:

1. Missing if condition for the NOTE when **read** and **write** are **high** and the memory is **full**
2. The **underflow** should be register as it is **sequential**
3. Missing assigning the **underflow** register
4. Including the condition when **read** and **write** are **high** and the memory is **empty**
5. The **almostfull** should be assigned when the **count** is 7 not 6

➔Design Code:

```
import fifo_pack::*;
module FIFO(FIFO_INTERF.FIFO_DUT_MODE fifo_interf);
logic [(FIFO_WIDTH-1):0]data_in;
logic wr_ack,overflow,underflow;
logic [(FIFO_WIDTH-1):0]data_out;
bit clk,rst_n,wr_en,rd_en,full,empty,almostfull,almostempty;

 assign data_in=fifo_interf.data_in;
 assign clk=fifo_interf.clk;
 assign rst_n=fifo_interf.rst_n;
 assign wr_en=fifo_interf.wr_en;
 assign rd_en=fifo_interf.rd_en;

 assign fifo_interf.wr_ack=wr_ack;
 assign fifo_interf.overflow=overflow;
 assign fifo_interf.full=full;
 assign fifo_interf.empty=empty;
 assign fifo_interf.almostfull=almostfull;
 assign fifo_interf.almostempty=almostempty;
 assign fifo_interf.underflow=underflow;
 assign fifo_interf.data_out=data_out;


localparam max_fifo_addr = $clog2(FIFO_DEPTH);
logic [max_fifo_addr-1:0] wr_ptr, rd_ptr;
logic [max_fifo_addr:0] count;
```

```
assign fifo_interf.wr_ptr=wr_ptr;
assign fifo_interf.rd_ptr=rd_ptr;
assign fifo_interf.count=count;

logic [FIFO_WIDTH-1:0] mem [FIFO_DEPTH-1:0];
///////////////////////////////////////////////////////
always @(posedge clk or negedge rst_n) begin//always block of the writing
	if (!rst_n) begin    ///// reseting all the registers /////
		wr_ptr <= 0;
		rd_ptr <= 0;
		count<=0;
		underflow<=0;
		overflow<=0;
		wr_ack<=0;
	end
	else if ((wr_en && (count < FIFO_DEPTH)) || (rd_en && (count == 0) &&
wr_en)) begin
		mem[wr_ptr] <= data_in;
		wr_ack <= 1;
		wr_ptr <= wr_ptr + 1;
	end
	else begin
		wr_ack <= 0;
		if (full & wr_en)begin
			overflow <= 1;
		end
		else begin
			overflow <= 0;

		end

	end
end

///////////////////////////////////////////////////////

always @(posedge clk or negedge rst_n) begin//always block of the reading
	if (!rst_n) begin   ///// reseting all the registers /////
		wr_ptr <= 0;
		rd_ptr <= 0;
		count<=0;
		underflow<=0;
		overflow<=0;
		wr_ack<=0;
	end
	else if ((rd_en && (count > 0) &&(!wr_en))  || (rd_en && (count ==
FIFO_DEPTH) &&wr_en)) begin
		data_out <= mem[rd_ptr];
```

```
            rd_ptr <= rd_ptr + 1;
      end
      else begin
            if (empty & rd_en)begin
                  underflow <= 1;
            end

            else begin
                  underflow <= 0;
            end

      end
end

///////////////////////////////////////////////////////

always @(posedge clk or negedge rst_n) begin//always block of the counting
      if (!rst_n) begin    ///// reseting all the registers /////
            wr_ptr <= 0;
            rd_ptr <= 0;
            count<=0;
            underflow<=0;
            overflow<=0;
            wr_ack<=0;
      end
      else begin
            if    ((wr_en && (count < FIFO_DEPTH)) || (rd_en && (count == 0)
&& wr_en))//including the condition when read and write are high and the
memory is empty
                  count <= count + 1;
            else if ((rd_en && (count > 0) &&(!wr_en))|| (rd_en && (count
==FIFO_DEPTH)&&wr_en))//including the condition when read and write are high
and the memory is full
                  count <= count - 1;
      end
end

///////////////////////////////////////////////////////


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

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

///////////////////////////////////////////////////////

endmodule
```

➔Top Module Code:

```
module FIFO_TOP;
import uvm_pkg::*;
`include "uvm_macros.svh"
import fifo_test_pack::*;
bit clk;
initial begin
    clk = 0;
    forever begin
      #1 clk =~clk;
    end
  end

FIFO_INTERF fifo_interf(clk);

FIFO fifo_dut(fifo_interf);

initial begin
  uvm_config_db#(virtual
FIFO_INTERF)::set(null,"uvm_test_top","FIFO_INTERF",fifo_interf);
  run_test("fifo_test");
end

bind FIFO fifo_sva fifo_sva_inst(fifo_interf);
endmodule : FIFO_TOP
```

➔Test Module Code:

```
package fifo_test_pack;
import uvm_pkg::*;
`include "uvm_macros.svh"

import fifo_env_pack::*;
import fifo_config_pack::*;
import fifo_sequencer_pack::*;
import fifo_1_sequence_pack::*;
import fifo_2_sequence_pack::*;
import fifo_3_and_4_sequence_pack::*;
import fifo_5_sequence_pack::*;
import fifo_6_sequence_pack::*;

import fifo_agent_pack::*;

class fifo_test extends uvm_test;
`uvm_component_utils(fifo_test)

fifo_env env;
fifo_config fifo_config_obj;

FIFO_1_sequence fifo_1_sequence;
FIFO_2_sequence fifo_2_sequence;
FIFO_3_and_4_sequence fifo_3_and_4_sequence;
FIFO_5_sequence fifo_5_sequence;
FIFO_6_sequence fifo_6_sequence;



function new (string name="fifo_test",uvm_component parent = null);
super.new(name,parent);
endfunction


function void build_phase (uvm_phase phase);
super.build_phase(phase);
env=fifo_env::type_id::create("env",this);
fifo_config_obj=fifo_config::type_id::create("fifo_config_obj");
fifo_1_sequence=FIFO_1_sequence::type_id::create("fifo_1_sequence",this);
fifo_2_sequence=FIFO_2_sequence::type_id::create("fifo_2_sequence",this);
fifo_3_and_4_sequence=FIFO_3_and_4_sequence::type_id::create("fifo_3_and_4_se
quence",this);
fifo_5_sequence=FIFO_5_sequence::type_id::create("fifo_5_sequence",this);
fifo_6_sequence=FIFO_6_sequence::type_id::create("fifo_6_sequence",this);
```

```
if (!(uvm_config_db#(virtual
FIFO_INTERF)::get(this,"","FIFO_INTERF",fifo_config_obj.fifo_vinterf)))
      `uvm_fatal("build_phase","Unable to get the FIFO_INTERF");


      uvm_config_db#(fifo_config)::set(this,"*","FIFO_CONFIG_OBJ",fifo_config
_obj);
endfunction



task run_phase (uvm_phase phase);
super.run_phase(phase);
phase.raise_objection(this);
`uvm_info("run_phase","Reset asserted",UVM_LOW)
fifo_1_sequence.start(env.fifo_agent_obj.fifo_sequencer_obj);
`uvm_info("run_phase","Reset deasserted",UVM_LOW)

`uvm_info("run_phase","FIFO_2 Transaction generation started",UVM_LOW)
fifo_2_sequence.start(env.fifo_agent_obj.fifo_sequencer_obj);
`uvm_info("run_phase","FIFO_2 Transaction generation ended",UVM_LOW)



`uvm_info("run_phase","FIFO_3 & FIFO_4 Transaction generation
started",UVM_LOW)
fifo_3_and_4_sequence.start(env.fifo_agent_obj.fifo_sequencer_obj);
`uvm_info("run_phase","FIFO_3 & FIFO_4 Transaction generation ended",UVM_LOW)

`uvm_info("run_phase","FIFO_5 Transaction generation started",UVM_LOW)
fifo_5_sequence.start(env.fifo_agent_obj.fifo_sequencer_obj);
`uvm_info("run_phase","FIFO_5 Transaction generation ended",UVM_LOW)




`uvm_info("run_phase","FIFO_6 Transaction generation started",UVM_LOW)
fifo_6_sequence.start(env.fifo_agent_obj.fifo_sequencer_obj);
`uvm_info("run_phase","FIFO_6 Transaction generation ended",UVM_LOW)

phase.drop_objection(this);
endtask

endclass
endpackage
```

➔FIFO Package:

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

static bit test_finished;
static int error_count=0;
static int correct_count=0;

endpackage
```

➔FIFO SVA:

```
module fifo_sva (FIFO_INTERF.FIFO_DUT_MODE fifo_interface);
import fifo_pack::*;
bit clk;

logic [(FIFO_WIDTH-1):0] data_in;
bit rst_n, wr_en, rd_en;
logic  [(FIFO_WIDTH-1):0] data_out;
logic wr_ack, overflow;
bit full, empty, almostfull, almostempty;
logic underflow;
localparam max_fifo_addr = $clog2(FIFO_DEPTH);
logic [max_fifo_addr-1:0] wr_ptr, rd_ptr;
logic [max_fifo_addr:0] count;

assign clk=fifo_interface.clk;
assign rst_n=fifo_interface.rst_n;
assign data_in=fifo_interface.data_in;
assign data_out=fifo_interface.data_out;
assign wr_en=fifo_interface.wr_en;
assign wr_ack=fifo_interface.wr_ack;
assign rd_en=fifo_interface.rd_en;
assign overflow=fifo_interface.overflow;
assign underflow=fifo_interface.underflow;
assign full=fifo_interface.full;
assign empty=fifo_interface.empty;
assign almostfull=fifo_interface.almostfull;
assign almostempty=fifo_interface.almostempty;
assign count=fifo_interface.count;
assign wr_ptr=fifo_interface.wr_ptr;
assign rd_ptr=fifo_interface.rd_ptr;



 property reset_flags_property;
  @(posedge clk) disable iff(rst_n) (rst_n == 0) |=> (full == 0
&& empty== 1 && almostfull== 0 && almostempty== 0 );
endproperty



 property reset_regs_property;
  @(posedge clk) disable iff(rst_n) (rst_n == 0) |=> (overflow
== 0 && underflow== 0 && wr_ack== 0 && wr_ptr== 0 && rd_ptr== 0
&& count== 0);
endproperty


 property full_property;
  @(posedge clk) disable iff(!rst_n) (count == FIFO_DEPTH) |->
(full == 1);
endproperty


 property empty_property;
  @(posedge clk) (count== 0) |-> (empty== 1);
endproperty


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

```
 property almostempty_property;
  @(posedge clk) disable iff(!rst_n)(count == 1) |->
(almostempty== 1);
endproperty


 property ov_property;
  @(posedge clk) disable iff(!rst_n)(count == FIFO_DEPTH
&&(wr_en == 1)) |=> (overflow== 1);
endproperty


 property un_property;
  @(posedge clk) disable iff(!rst_n)(count== 0 &&(rd_en == 1))
|=> (underflow== 1);
endproperty


 property wr_ack_property;
  @(posedge clk) disable iff(!rst_n) ((wr_en && (count <
FIFO_DEPTH)) || (rd_en && (count == 0) && wr_en)) |=> (wr_ack==
1);
endproperty
assert property(reset_regs_property); cover
property(reset_regs_property);
assert property(reset_flags_property); cover
property(reset_flags_property);
assert property(full_property); cover property(full_property);
assert property(empty_property); cover property(empty_property);
assert property(almostfull_property); cover
property(almostfull_property);
assert property(almostempty_property); cover
property(almostempty_property);
assert property(ov_property); cover property(ov_property);
assert property(un_property); cover property(un_property);
assert property(wr_ack_property); cover
property(wr_ack_property);
endmodule : fifo_sva
```

➔FIFO interface:

```
interface FIFO_INTERF(clk);
import fifo_pack::*;
input  clk;

logic [(FIFO_WIDTH-1):0] data_in;
bit rst_n, wr_en, rd_en;
logic  [(FIFO_WIDTH-1):0] data_out;
logic wr_ack, overflow;
bit full, empty, almostfull, almostempty;
logic underflow;


localparam max_fifo_addr = $clog2(FIFO_DEPTH);
logic [max_fifo_addr-1:0] wr_ptr, rd_ptr;
logic [max_fifo_addr:0] count;


modport FIFO_DUT_MODE (input clk,data_in, wr_en, rd_en,
rst_n,output full, empty, almostfull, almostempty, wr_ack,
overflow, underflow, data_out,count,wr_ptr,rd_ptr);
endinterface
```

➔FIFO Configuration Object:

```
package fifo_config_pack;
import uvm_pkg::*;
`include "uvm_macros.svh"

class fifo_config extends uvm_object;
`uvm_object_utils(fifo_config)
virtual FIFO_INTERF fifo_vinterf;

function new(string name = "fifo_config");
super.new(name);
endfunction
endclass
endpackage
```

➔Environment Class Code:

```
package fifo_env_pack;
      import uvm_pkg::*;
`include "uvm_macros.svh"
import fifo_agent_pack::*;
import fifo_scoreboard_pack::*;
import fifo_coverage_pack::*;

class fifo_env extends uvm_env;
`uvm_component_utils(fifo_env)

function new (string name="fifo_env",uvm_component parent = null);
super.new(name,parent);
endfunction
fifo_agent fifo_agent_obj;
fifo_scoreboard fifo_scoreboard_obj;
fifo_coverage fifo_coverage_obj;

function void build_phase(uvm_phase phase);
super.build_phase(phase);
fifo_agent_obj=fifo_agent::type_id::create("fifo_agent_obj",this);
fifo_scoreboard_obj=fifo_scoreboard::type_id::create("fifo_scoreboard_obj",th
is);
fifo_coverage_obj=fifo_coverage::type_id::create("fifo_coverage_obj",this);
endfunction


function void connect_phase(uvm_phase phase);
      super.connect_phase(phase);

      fifo_agent_obj.agent_anal_port.connect(fifo_coverage_obj.coverage_expor
t);
      fifo_agent_obj.agent_anal_port.connect(fifo_scoreboard_obj.scoreboard_e
xport);
endfunction
endclass
endpackage : fifo_env_pack
```

➔Sequence Item Code:

```
package fifo_sequence_item_pack;
import uvm_pkg::*;
`include "uvm_macros.svh"
import fifo_pack::*;
class fifo_sequence_item extends  uvm_sequence_item;
     `uvm_object_utils(fifo_sequence_item)


          int RD_EN_ON_DIST;
          int WR_EN_ON_DIST;

function new (int RD=30,int WR=70,string name ="fifo_sequence_item");
     super.new(name);
     this.RD_EN_ON_DIST=RD;
     this.WR_EN_ON_DIST=WR;
endfunction

function string convert2string();
return $sformatf("%s
rst_n=%b,wr_en=%b,rd_en=%b,data_in=%0h,data_out=%0h,count=%0d,wr_ack=%
b,full=%b,empty=%b,overflow=%b,underflow=%b,almostfull=%b,almostempty=
%b",
                    super.convert2string(),
                    rst_n_rand,wr_en_rand,rd_en_rand,
                    data_in_rand,data_out,count,
                    wr_ack_transc,full_transc,empty_transc,
                    overflow_transc,underflow_transc,
                    almostfull_transc,almostempty_transc);
endfunction

function string convert2string_stim();
return
$sformatf("rst_n=%b,wr_en=%b,rd_en=%b,data_in=%0h,data_out=%0h,count=%
0d,wr_ack=%b,full=%b,empty=%b,overflow=%b,underflow=%b,almostfull=%b,a
lmostempty=%b",
                    rst_n_rand,wr_en_rand,rd_en_rand,
                    data_in_rand,data_out,count,
                    wr_ack_transc,full_transc,empty_transc,
                    overflow_transc,underflow_transc,
                    almostfull_transc,almostempty_transc);
endfunction
```

```
bitfull_transc,empty_transc,almostfull_transc,almostempty_transc,wr_ac
k_transc, overflow_transc,underflow_transc;

rand logic [FIFO_WIDTH-1:0]data_in_rand;
logic [FIFO_WIDTH-1:0] data_out;
rand bit rst_n_rand,wr_en_rand,rd_en_rand;
bit clock_transc;
logic [3:0] count;

constraint rst_constraint {rst_n_rand dist{1'b0:=10,1'b1:=90};}
constraint write_constraint {wr_en_rand dist{1'b0:=(100-
WR_EN_ON_DIST),1'b1:=WR_EN_ON_DIST};}
constraint read_constraint {rd_en_rand dist{1'b0:=(100-
RD_EN_ON_DIST),1'b1:=RD_EN_ON_DIST};}
constraint data_in_constraint {data_in_rand inside {[0:16'hFFFF]};}


endclass : fifo_sequence_item

endpackage : fifo_sequence_item_pack
```

➔ **Sequencer Code:**

```
package fifo_sequencer_pack;
     import uvm_pkg::*;
`include "uvm_macros.svh"
import fifo_sequence_item_pack::*;


     class fifo_sequencer extends  uvm_sequencer
#(fifo_sequence_item);
`uvm_component_utils(fifo_sequencer)

function new(string name ="fifo_sequencer",uvm_component parent
=null);
     super.new(name,parent);
endfunction
     endclass : fifo_sequencer

endpackage : fifo_sequencer_pack
```

➔FIFO_1 Sequence Code:

```
package fifo_1_sequence_pack;

      import fifo_sequence_item_pack::*;

import uvm_pkg::*;
`include "uvm_macros.svh"

class FIFO_1_sequence extends  uvm_sequence #(fifo_sequence_item);
      `uvm_object_utils(FIFO_1_sequence)
fifo_sequence_item fifo_sequence_item_obj;



      function new(string name = "FIFO_1_sequence");
            super.new(name);
      endfunction


task body();
      fifo_sequence_item_obj=fifo_sequence_item::type_id::create("fifo_sequen
ce_item_obj");
start_item(fifo_sequence_item_obj);

            assert(fifo_sequence_item_obj.randomize() with
{data_in_rand==16'h0BCC;wr_en_rand==0;

      rd_en_rand==0;rst_n_rand==0;});



finish_item(fifo_sequence_item_obj);
endtask : body


endclass
endpackage
```

➔FIFO_2 Sequence Code:

```
package fifo_2_sequence_pack;

      import fifo_sequence_item_pack::*;

import uvm_pkg::*;
`include "uvm_macros.svh"

class FIFO_2_sequence extends  uvm_sequence #(fifo_sequence_item);
      `uvm_object_utils(FIFO_2_sequence)
fifo_sequence_item fifo_sequence_item_obj;

      function new(string name = "FIFO_2_sequence");
            super.new(name);
      endfunction


task body();


      fifo_sequence_item_obj=fifo_sequence_item::type_id::create("fifo_sequen
ce_item_obj");
start_item(fifo_sequence_item_obj);

            assert(fifo_sequence_item_obj.randomize() with {wr_en_rand==1;

      rd_en_rand==1;rst_n_rand==1;});





finish_item(fifo_sequence_item_obj);

endtask : body

endclass
endpackage
```

➔FIFO_3_and_4 Sequence Code:

```
package fifo_3_and_4_sequence_pack;

      import fifo_sequence_item_pack::*;

import uvm_pkg::*;
`include "uvm_macros.svh"

class FIFO_3_and_4_sequence extends  uvm_sequence #(fifo_sequence_item);
      `uvm_object_utils(FIFO_3_and_4_sequence)
fifo_sequence_item fifo_sequence_item_obj;

      function new(string name = "FIFO_3_and_4_sequence");
            super.new(name);
      endfunction

task body();

        for (int i = 0; i < 10; i++) begin

fifo_sequence_item_obj=fifo_sequence_item::type_id::create("fifo_sequence_ite
m_obj");
        start_item(fifo_sequence_item_obj);
            assert(fifo_sequence_item_obj.randomize() with {wr_en_rand==1;

      rd_en_rand==0;rst_n_rand==1;});

            if (i == 8) begin
                  assert(fifo_sequence_item_obj.randomize() with
{wr_en_rand==1;

      rd_en_rand==1;rst_n_rand==1;});
            end
            finish_item(fifo_sequence_item_obj);
      end
endtask : body
endclass
endpackage
```

➔FIFO_5 Sequence Code:

```
package fifo_5_sequence_pack;

      import fifo_sequence_item_pack::*;

import uvm_pkg::*;
`include "uvm_macros.svh"

class FIFO_5_sequence extends  uvm_sequence #(fifo_sequence_item);
      `uvm_object_utils(FIFO_5_sequence)
fifo_sequence_item fifo_sequence_item_obj;

      function new(string name = "FIFO_5_sequence");
            super.new(name);
      endfunction


task body();

      repeat (10) begin

      fifo_sequence_item_obj=fifo_sequence_item::type_id::create("fifo_sequen
ce_item_obj");
            start_item(fifo_sequence_item_obj);
            assert(fifo_sequence_item_obj.randomize() with {wr_en_rand==0;

      rd_en_rand==1;rst_n_rand==1;});

            finish_item(fifo_sequence_item_obj);
      end


endtask : body





endclass
endpackage
```

➔FIFO_6 Sequence Code:

```
package fifo_6_sequence_pack;

      import fifo_sequence_item_pack::*;

import uvm_pkg::*;
`include "uvm_macros.svh"

class FIFO_6_sequence extends  uvm_sequence #(fifo_sequence_item);
      `uvm_object_utils(FIFO_6_sequence)
fifo_sequence_item fifo_sequence_item_obj;

      function new(string name = "FIFO_6_sequence");
            super.new(name);
      endfunction


task body();
      repeat (1000) begin
      fifo_sequence_item_obj=fifo_sequence_item::type_id::create("fifo_sequen
ce_item_obj");
            start_item(fifo_sequence_item_obj);
            assert(fifo_sequence_item_obj.randomize());
            finish_item(fifo_sequence_item_obj);
      end
endtask : body




endclass
endpackage
```

➔FIFO Agent Code:

```
package fifo_agent_pack;
import uvm_pkg::*;
`include "uvm_macros.svh"
import fifo_sequence_item_pack::*;
import fifo_sequencer_pack::*;
import fifo_driver_pack::*;
import fifo_monitor_pack::*;
import fifo_config_pack::*;
class fifo_agent extends  uvm_agent;
`uvm_component_utils(fifo_agent)

fifo_sequencer fifo_sequencer_obj;
fifo_driver driver_obj;
fifo_monitor fifo_monitor_obj;

fifo_config fifo_config_obj;
uvm_analysis_port #(fifo_sequence_item)  agent_anal_port;


function new(string name ="fifo_agent",uvm_component parent =null);
      super.new(name,parent);
endfunction

function void build_phase(uvm_phase phase);
super.build_phase(phase);
if
(!(uvm_config_db#(fifo_config)::get(this,"","FIFO_CONFIG_OBJ",fifo_config_obj
)))
      `uvm_fatal("build_phase","Unable to get the FIFO_CONFIG_OBJ");

driver_obj=fifo_driver::type_id::create("driver_obj",this);
fifo_sequencer_obj
=fifo_sequencer::type_id::create("fifo_sequencer_obj",this);
fifo_monitor_obj=fifo_monitor::type_id::create("fifo_monitor_obj",this);
agent_anal_port=new("agent_anal_port",this);
endfunction

function void connect_phase(uvm_phase phase);
      super.connect_phase(phase);
      driver_obj.fifo_vinterf=fifo_config_obj.fifo_vinterf;
      fifo_monitor_obj.fifo_vinterf=fifo_config_obj.fifo_vinterf;
      driver_obj.seq_item_port.connect(fifo_sequencer_obj.seq_item_export);
      fifo_monitor_obj.monitor_anal_port.connect(agent_anal_port);
endfunction
endclass

endpackage : fifo_agent_pack
```

➔FIFO Driver Code:

```
package fifo_driver_pack;
import fifo_sequence_item_pack::*;

import fifo_config_pack::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

class fifo_driver extends uvm_driver#(fifo_sequence_item);
      `uvm_component_utils(fifo_driver);

fifo_sequence_item sequence_item_obj;

virtual FIFO_INTERF fifo_vinterf;
function new (string name="fifo_driver",uvm_component parent = null);
super.new(name,parent);
endfunction

task run_phase(uvm_phase phase);
    super.run_phase(phase);
    forever begin
        sequence_item_obj =
fifo_sequence_item::type_id::create("sequence_item_obj");


        seq_item_port.get_next_item(sequence_item_obj);
        fifo_vinterf.rst_n= sequence_item_obj.rst_n_rand;
        fifo_vinterf.wr_en= sequence_item_obj.wr_en_rand;
        fifo_vinterf.rd_en= sequence_item_obj.rd_en_rand;
        fifo_vinterf.data_in= sequence_item_obj.data_in_rand;

        @(negedge fifo_vinterf.clk);
        seq_item_port.item_done();

`uvm_info("run_phase",sequence_item_obj.convert2string_stim(),UVM_HIGH)
    end

endtask

endclass


endpackage : fifo_driver_pack
```

➔FIFO Monitor Code:

```
package fifo_monitor_pack;
      import uvm_pkg::*;
`include "uvm_macros.svh"
import fifo_sequence_item_pack::*;


class fifo_monitor extends  uvm_monitor;
      `uvm_component_utils(fifo_monitor);
virtual FIFO_INTERF fifo_vinterf;
fifo_sequence_item fifo_sequence_item_obj;
uvm_analysis_port#(fifo_sequence_item) monitor_anal_port;
function new (string name="fifo_monitor",uvm_component parent = null);
super.new(name,parent);
endfunction

function void build_phase (uvm_phase phase);
super.build_phase(phase);
monitor_anal_port=new("monitor_anal_port",this);
endfunction
task run_phase(uvm_phase phase);
      super.run_phase(phase);
forever begin
      fifo_sequence_item_obj=fifo_sequence_item::type_id::create("fifo_sequen
ce_item_obj");
@(negedge fifo_vinterf.clk);
fifo_sequence_item_obj.rst_n_rand = fifo_vinterf.rst_n;
fifo_sequence_item_obj.wr_en_rand = fifo_vinterf.wr_en;
fifo_sequence_item_obj.wr_ack_transc = fifo_vinterf.wr_ack;
fifo_sequence_item_obj.rd_en_rand = fifo_vinterf.rd_en;
fifo_sequence_item_obj.full_transc = fifo_vinterf.full;
fifo_sequence_item_obj.empty_transc = fifo_vinterf.empty;
fifo_sequence_item_obj.almostfull_transc = fifo_vinterf.almostfull;
fifo_sequence_item_obj.almostempty_transc = fifo_vinterf.almostempty;
fifo_sequence_item_obj.overflow_transc = fifo_vinterf.overflow;
fifo_sequence_item_obj.underflow_transc = fifo_vinterf.underflow;
fifo_sequence_item_obj.data_in_rand = fifo_vinterf.data_in;
fifo_sequence_item_obj.data_out = fifo_vinterf.data_out;
fifo_sequence_item_obj.count=fifo_vinterf.count;
monitor_anal_port.write(fifo_sequence_item_obj);
`uvm_info("run_phase",fifo_sequence_item_obj.convert2string(),UVM_HIGH);
end
endtask : run_phase
endclass : fifo_monitor

endpackage : fifo_monitor_pack
```

➔FIFO Scoreboard Code:

```
package fifo_scoreboard_pack;
import uvm_pkg::*;
`include "uvm_macros.svh"
import fifo_sequence_item_pack::*;
import fifo_pack::*;


class fifo_scoreboard extends  uvm_scoreboard;
`uvm_component_utils(fifo_scoreboard)

fifo_sequence_item sequence_item_scoreboard;
uvm_analysis_export #(fifo_sequence_item) scoreboard_export;
uvm_tlm_analysis_fifo #(fifo_sequence_item) scoreboard_fifo;

logic [FIFO_WIDTH-1:0] fifo_queue[$];
logic [FIFO_WIDTH-1:0] data_out_ref,data_in_ref;
bit wr_en_ref, rd_en_ref,rst_n_ref,full_ref,empty_ref;
static int count_ref=0;

int error_count=0;
int correct_count=0;

function new(string name ="fifo_scoreboard",uvm_component parent =null);
      super.new(name,parent);
endfunction

function void build_phase(uvm_phase phase);
super.build_phase(phase);
scoreboard_export=new("scoreboard_export",this);
scoreboard_fifo=new("scoreboard_fifo",this);
endfunction

function void connect_phase(uvm_phase phase);
super.connect_phase(phase);

      scoreboard_export.connect(scoreboard_fifo.analysis_export);
endfunction
```

```
task run_phase(uvm_phase phase);
      super.run_phase(phase);

      forever begin
            scoreboard_fifo.get(sequence_item_scoreboard);
            ref_model(sequence_item_scoreboard);

             if (sequence_item_scoreboard.data_out !== data_out_ref) begin
                  $display("------------------------");
      `uvm_error("run_phase",$sformatf("Comparison Failed !!, Transac. sent
to the dut is:%s",
                        sequence_item_scoreboard.convert2string()));
      $display("data_out_ref=%0h",data_out_ref);
      $display("DUT data_out=%0h",sequence_item_scoreboard.data_out);
      $display("------------------------");
                  error_count++;
            end
            else begin
      $display("------------------------");
                  `uvm_info("run_phase",$sformatf("Correct , Transac. sent to
the dut is:%s",
                        sequence_item_scoreboard.convert2string()),UVM_LOW);
                  $display("data_out_ref=%0h",data_out_ref);
      $display("DUT data_out=%0h",sequence_item_scoreboard.data_out);
      $display("------------------------");
                  correct_count++;
            end



            end

endtask : run_phase
```

```
task ref_model(fifo_sequence_item ref_item);

data_in_ref=ref_item.data_in_rand;
       wr_en_ref=ref_item.wr_en_rand;
       rd_en_ref=ref_item.rd_en_rand;
       rst_n_ref=ref_item.rst_n_rand;


if (rst_n_ref == 0) begin
      wr_en_ref=0;
      rd_en_ref=0;
      count_ref=0;
      fifo_queue.delete();
      data_out_ref=ref_item.data_out;

end
else if ((wr_en_ref && (count_ref < FIFO_DEPTH)) || (rd_en_ref && (count_ref
== 0) && wr_en_ref) ) begin
     fifo_queue.push_back(data_in_ref);
     data_out_ref=ref_item.data_out;
       count_ref++;
end
else if ((rd_en_ref && (count_ref > 0) &&(!wr_en_ref))|| (rd_en_ref &&
(count_ref ==FIFO_DEPTH)&&wr_en_ref)) begin
       data_out_ref=fifo_queue.pop_front();
     count_ref--;
end
else data_out_ref=ref_item.data_out;

if (count_ref == FIFO_DEPTH) ref_item.full_transc=1;
if (count_ref == 0) ref_item.empty_transc=1;
if (count_ref == 1) ref_item.almostempty_transc=1;
if (count_ref == 7) ref_item.almostfull_transc=1;

endtask : ref_model


function void report_phase(uvm_phase phase);
      super.report_phase(phase);
      `uvm_info("report_phase",$sformatf("Correct_count:%0d",correct_count),U
VM_MEDIUM);
      `uvm_info("report_phase",$sformatf("Error_count:%0d",error_count),UVM_M
EDIUM);
endfunction
endclass : fifo_scoreboard
endpackage : fifo_scoreboard_pack
```

➔FIFO Coverage Collector Code:

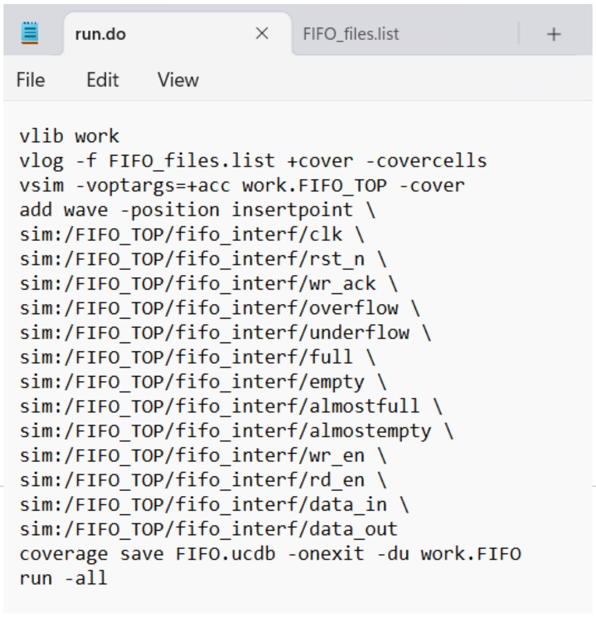
```
package fifo_coverage_pack;
import uvm_pkg::*;
`include "uvm_macros.svh"
import fifo_sequence_item_pack::*;
import fifo_pack::*;

      class fifo_coverage extends  uvm_component;
            `uvm_component_utils(fifo_coverage)

fifo_sequence_item sequence_item_coverage;
uvm_analysis_export #(fifo_sequence_item) coverage_export;
uvm_tlm_analysis_fifo #(fifo_sequence_item) coverage_uvm_fifo;

            bit cv_clk;
        bit cv_wr;
        bit cv_rd;
        bit cv_full;
        bit cv_empty;
        bit cv_almostfull;
        bit cv_almostempty;
        bit cv_ov;
        bit cv_un;
        bit cv_wr_ack;


covergroup fifo_cvgroup;


cv_wr_f_e:cross cv_wr,cv_full,cv_empty
{
bins wr_full_high= binsof(cv_wr) intersect {1} && binsof(cv_full) intersect
{1};
bins wr_empty_high= binsof(cv_wr) intersect {1} && binsof(cv_empty) intersect
{1};
bins wr_full_low= binsof(cv_wr) intersect {1} && binsof(cv_full) intersect
{0};
bins wr_empty_low= binsof(cv_wr) intersect {1} && binsof(cv_empty) intersect
{0};
option.cross_auto_bin_max=0;
}
```

```
cv_rd_f_e:cross cv_rd,cv_full,cv_empty
{

bins rd_full_high= binsof(cv_rd) intersect {1} && binsof(cv_full) intersect
{1};
bins rd_empty_high= binsof(cv_rd) intersect {1} && binsof(cv_empty) intersect
{1};

bins rd_full_low= binsof(cv_rd) intersect {1} && binsof(cv_full) intersect
{0};
bins rd_empty_low= binsof(cv_rd) intersect {1} && binsof(cv_empty) intersect
{0};
option.cross_auto_bin_max=0;
}


cv_wr_rd_almostf_almoste:cross cv_wr,cv_rd,cv_almostfull,cv_almostempty
{
bins wr_almostfull_high= binsof(cv_wr) intersect {1} && binsof(cv_almostfull)
intersect {1};
bins wr_almostempty_high= binsof(cv_wr) intersect {1} &&
binsof(cv_almostempty) intersect {1};

bins rd_almostfull_high= binsof(cv_rd) intersect {1} && binsof(cv_almostfull)
intersect {1};
bins rd_almostempty_high= binsof(cv_rd) intersect {1} &&
binsof(cv_almostempty) intersect {1};

bins wr_almostfull_low= binsof(cv_wr) intersect {1} && binsof(cv_almostfull)
intersect {0};
bins wr_almostempty_low= binsof(cv_wr) intersect {1} &&
binsof(cv_almostempty) intersect {0};

bins rd_almostfull_low= binsof(cv_rd) intersect {1} && binsof(cv_almostfull)
intersect {0};
bins rd_almostempty_low= binsof(cv_rd) intersect {1} &&
binsof(cv_almostempty) intersect {0};

      option.cross_auto_bin_max=0;
}
```

```
cv_wr_rd_ov_un:cross cv_wr,cv_rd,cv_ov,cv_un
{
bins wr_ovf_high= binsof(cv_wr) intersect {1} && binsof(cv_ov) intersect {1};
bins wr_unf_high= binsof(cv_wr) intersect {1} && binsof(cv_un) intersect {1};

bins rd_ovf_high= binsof(cv_rd) intersect {1} && binsof(cv_ov) intersect {1};
bins rd_unf_high= binsof(cv_rd) intersect {1} && binsof(cv_un) intersect {1};

bins wr_ovf_low= binsof(cv_wr) intersect {1} && binsof(cv_ov) intersect {0};
bins wr_unf_low= binsof(cv_wr) intersect {1} && binsof(cv_un) intersect {0};

bins rd_ovf_low= binsof(cv_rd) intersect {1} && binsof(cv_ov) intersect {0};
bins rd_unf_low= binsof(cv_rd) intersect {1} && binsof(cv_un) intersect {0};
      option.cross_auto_bin_max=0;
}


cv_wr_rd_wrack:cross cv_wr,cv_rd,cv_wr_ack
{
      bins wr_ack_high= binsof(cv_wr) intersect {1} && binsof(cv_wr_ack)
intersect {1};
      bins wr_ack_low= binsof(cv_wr) intersect {1} && binsof(cv_wr_ack)
intersect {0};

      bins rd_ack_high= binsof(cv_rd) intersect {1} && binsof(cv_wr_ack)
intersect {1};
      bins rd_ack_low= binsof(cv_rd) intersect {1} && binsof(cv_wr_ack)
intersect {0};
      option.cross_auto_bin_max=0;
}

endgroup



function new (string name="fifo_coverage",uvm_component parent = null);
super.new(name,parent);
fifo_cvgroup=new();
endfunction

function void build_phase(uvm_phase phase);
super.build_phase(phase);
coverage_export=new("coverage_export",this);
coverage_uvm_fifo=new("coverage_uvm_fifo",this);
endfunction
```

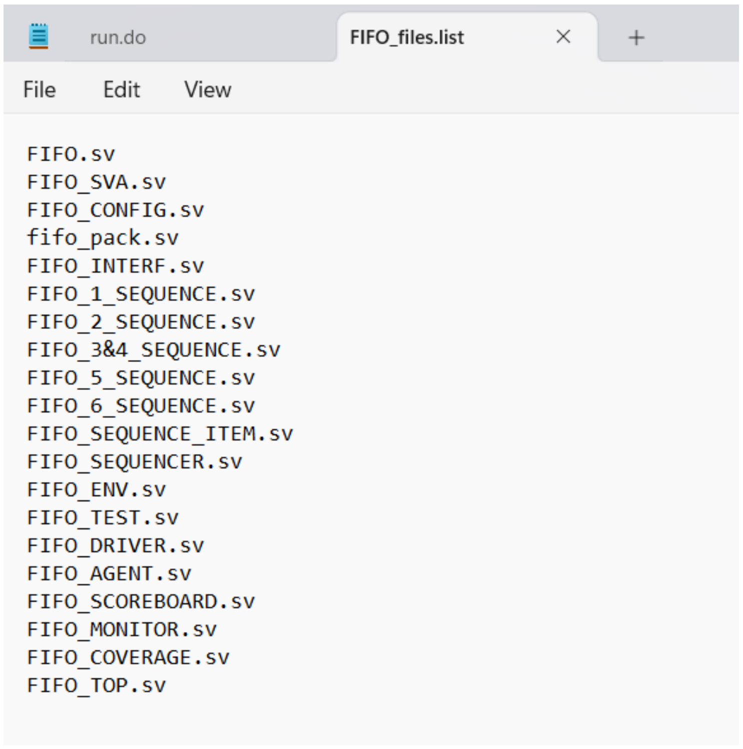
```
function void connect_phase(uvm_phase phase);
      super.connect_phase(phase);
coverage_export.connect(coverage_uvm_fifo.analysis_export);
endfunction


task run_phase(uvm_phase phase);
      super.run_phase(phase);

      forever begin
            coverage_uvm_fifo.get(sequence_item_coverage);

            cv_wr = sequence_item_coverage.wr_en_rand;
            cv_rd = sequence_item_coverage.rd_en_rand;
            cv_full = sequence_item_coverage.full_transc;
            cv_empty = sequence_item_coverage.empty_transc;
            cv_almostfull = sequence_item_coverage.almostfull_transc;
            cv_almostempty = sequence_item_coverage.almostempty_transc;
            cv_ov = sequence_item_coverage.overflow_transc;
            cv_un = sequence_item_coverage.underflow_transc;
            cv_wr_ack = sequence_item_coverage.wr_ack_transc;


            fifo_cvgroup.sample();
                  end

endtask : run_phase

      endclass : fifo_coverage

endpackage
```

➔ Do File:



```
vlib work
vlog -f FIFO_files.list +cover -covercells
vsim -voptargs=+acc work.FIFO_TOP -cover
add wave -position insertpoint \
sim:/FIFO_TOP/fifo_interf/clk \
sim:/FIFO_TOP/fifo_interf/rst_n \
sim:/FIFO_TOP/fifo_interf/wr_ack \
sim:/FIFO_TOP/fifo_interf/overflow \
sim:/FIFO_TOP/fifo_interf/underflow \
sim:/FIFO_TOP/fifo_interf/full \
sim:/FIFO_TOP/fifo_interf/empty \
sim:/FIFO_TOP/fifo_interf/almostfull \
sim:/FIFO_TOP/fifo_interf/almostempty \
sim:/FIFO_TOP/fifo_interf/wr_en \
sim:/FIFO_TOP/fifo_interf/rd_en \
sim:/FIFO_TOP/fifo_interf/data_in \
sim:/FIFO_TOP/fifo_interf/data_out
coverage save FIFO.ucdb -onexit -du work.FIFO
run -all
```

➔FIFO files list file:



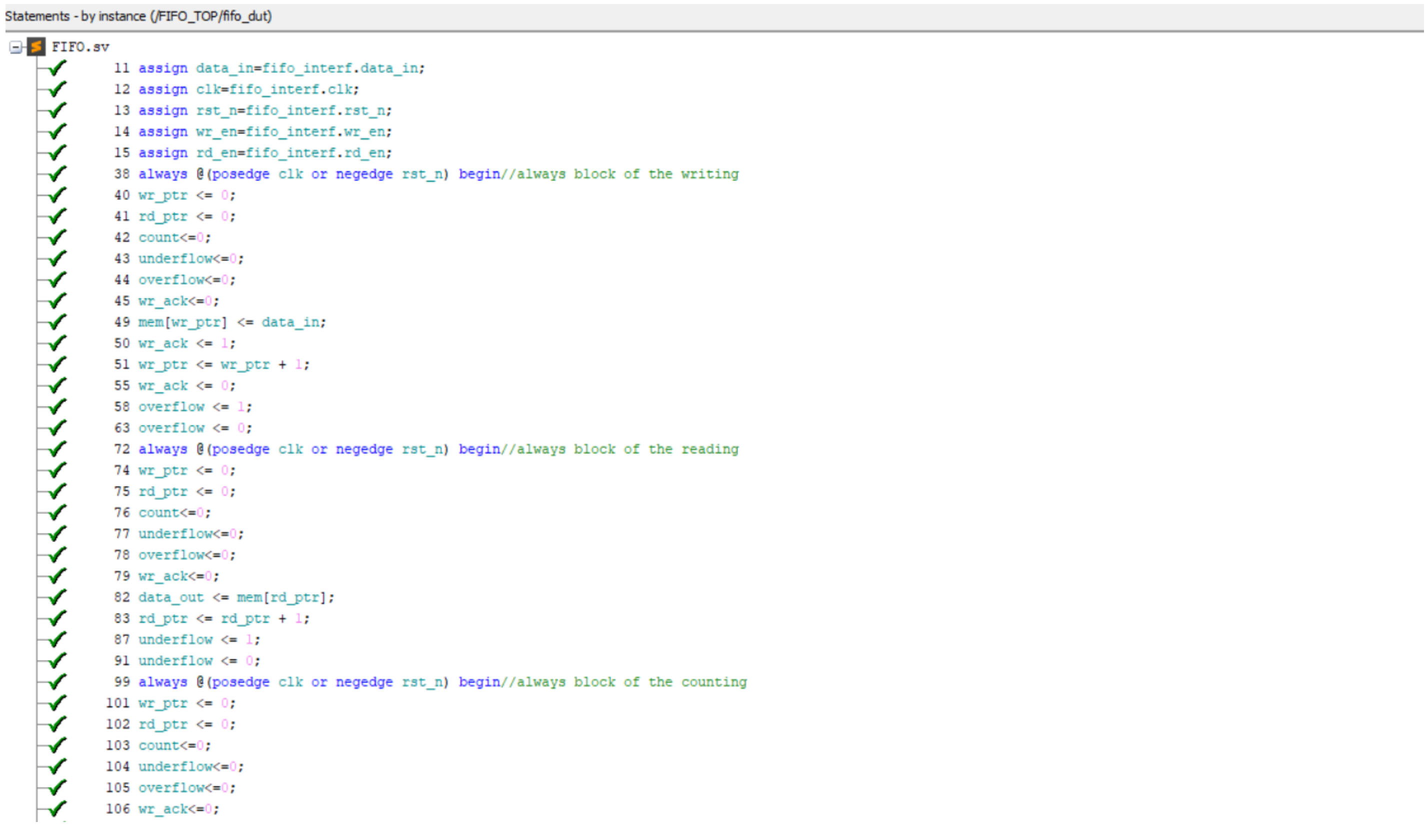
```
FIFO.sv
FIFO_SVA.sv
FIFO_CONFIG.sv
fifo_pack.sv
FIFO_INTERF.sv
FIFO_1_SEQUENCE.sv
FIFO_2_SEQUENCE.sv
FIFO_3&4_SEQUENCE.sv
FIFO_5_SEQUENCE.sv
FIFO_6_SEQUENCE.sv
FIFO_SEQUENCE_ITEM.sv
FIFO_SEQUENCER.sv
FIFO_ENV.sv
FIFO_TEST.sv
FIFO_DRIVER.sv
FIFO_AGENT.sv
FIFO_SCOREBOARD.sv
FIFO_MONITOR.sv
FIFO_COVERAGE.sv
FIFO_TOP.sv
```

➔ FIFO Verification Plan:

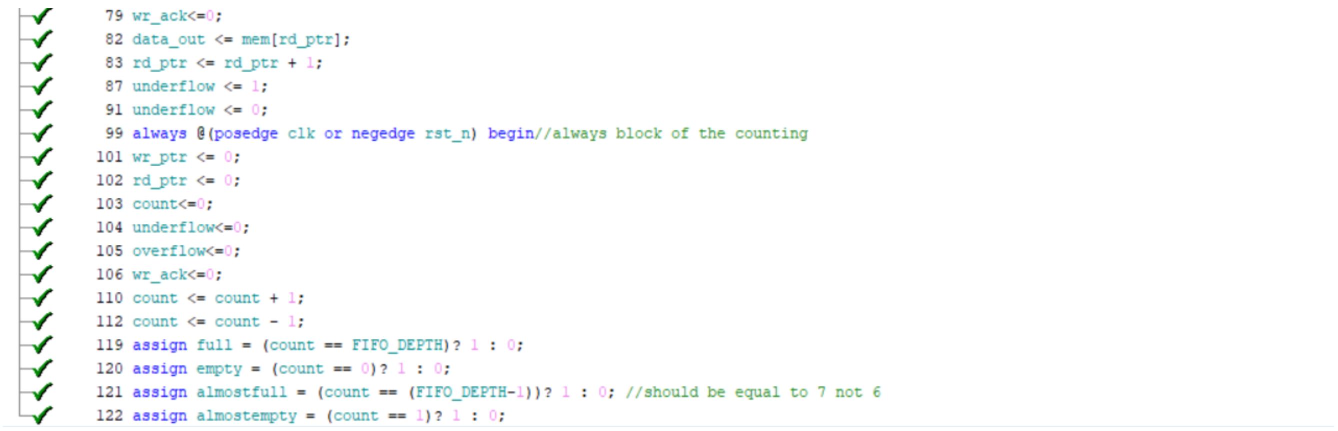
Label	Description	Stimulus Generation	Functional Coverage (Later)	Functionality Check
FIFO_1	When the reset is asserted,all the internal signal registers should be low	Directed at the start of the simulation,randomized with constrain that the rst_n is low	-	Concurrent assertion to check the reseting of all the internal registers and internal signals
FIFO_2	When the wr_en & rd_en are asserted immediately after the reset, the writing should be done as the meory is empty	Randomized with constrain that the wr_en & rd_en are high & the data_in in range [0:FFFFh]	Cover the wr_en & rd_en with the empty flag and cover the rd_en with the underflow register	Concurrent assertion to check the empty flag & the underflow register & the almostempty flag
FIFO_3	When the wr_en is asserted 10 times , the writing should be done till filling the memory	Randomized with constrain that the wr_en is high & rd_en are low & the data_in in range [0:FFFFh]	Cover the wr_en with the full flag & the overflow register &the wr_ack register	Concurrent assertion to check the full flag & the overflow register & the almostfull flag & the wr_ack register
FIFO_4	After the wr_en is asserted 8 times , the rd_en & wr_en are asserted while the memory is full,the reading should be done	Randomized with constrain that the wr_en & rd_en are high & the data_in in range [0:FFFFh]	Cover the rd_en with the full flag & the overflow register	Concurrent assertion to check the full flag & the overflow register & the wr_ack register
FIFO_5	When the rd_en is asserted 10 times , the reading should be done till the memory is empty	Randomized with constrain that the wr_en is low & rd_en is high	Cover the rd_en with the full flag & the overflow register & the empty flag & the underflow register	Concurrent assertion to check the empty flag & the underflow register & the almostempty flag
FIFO_6	Randomizatoin of the internal signals to cover all cases of writing & reading	Randomized with constrain that the wr_en to be low 30% & high 70% & rd_en to be low 70% & high 30% & the data_in in range [0:FFFFh]	Cover the rd_en & the wr_en with all the cases of the internal signals	Concurrent assertion to check all the flags & all the internal registers

➔ Code Coverage :

1- Statements Coverage :

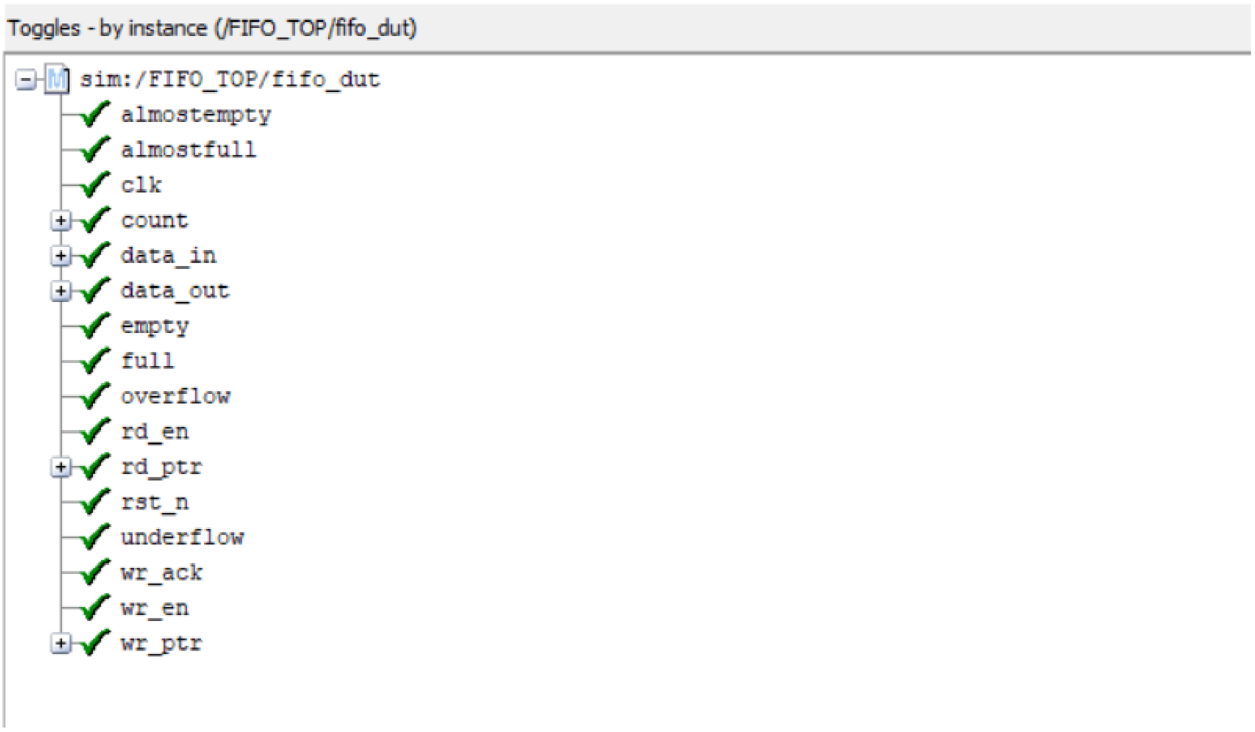


```
Statements - by instance (/FIFO_TOP/fifo_dut)
FIFO.sv
✓  11 assign data_in=fifo_interf.data_in;
✓  12 assign clk=fifo_interf.clk;
✓  13 assign rst_n=fifo_interf.rst_n;
✓  14 assign wr_en=fifo_interf.wr_en;
✓  15 assign rd_en=fifo_interf.rd_en;
✓  38 always @(posedge clk or negedge rst_n) begin//always block of the writing
✓  40 wr_ptr <= 0;
✓  41 rd_ptr <= 0;
✓  42 count<=0;
✓  43 underflow<=0;
✓  44 overflow<=0;
✓  45 wr_ack<=0;
✓  49 mem[wr_ptr] <= data_in;
✓  50 wr_ack <= 1;
✓  51 wr_ptr <= wr_ptr + 1;
✓  55 wr_ack <= 0;
✓  58 overflow <= 1;
✓  63 overflow <= 0;
✓  72 always @(posedge clk or negedge rst_n) begin//always block of the reading
✓  74 wr_ptr <= 0;
✓  75 rd_ptr <= 0;
✓  76 count<=0;
✓  77 underflow<=0;
✓  78 overflow<=0;
✓  79 wr_ack<=0;
✓  82 data_out <= mem[rd_ptr];
✓  83 rd_ptr <= rd_ptr + 1;
✓  87 underflow <= 1;
✓  91 underflow <= 0;
✓  99 always @(posedge clk or negedge rst_n) begin//always block of the counting
✓ 101 wr_ptr <= 0;
✓ 102 rd_ptr <= 0;
✓ 103 count<=0;
✓ 104 underflow<=0;
✓ 105 overflow<=0;
✓ 106 wr_ack<=0;
```



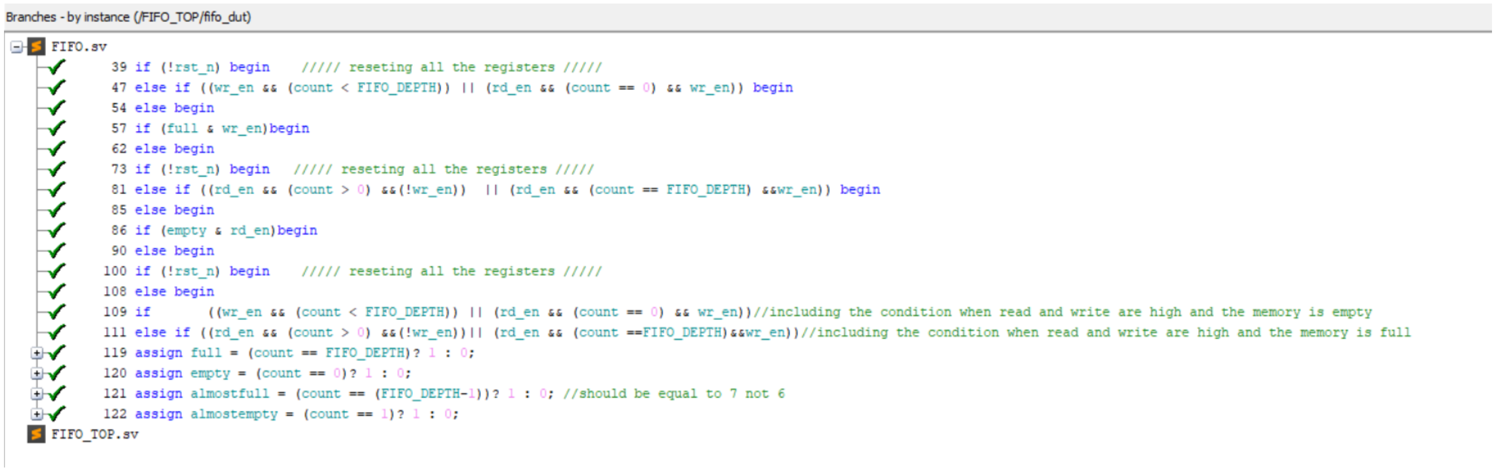
```
✓  79 wr_ack<=0;
✓  82 data_out <= mem[rd_ptr];
✓  83 rd_ptr <= rd_ptr + 1;
✓  87 underflow <= 1;
✓  91 underflow <= 0;
✓  99 always @(posedge clk or negedge rst_n) begin//always block of the counting
✓ 101 wr_ptr <= 0;
✓ 102 rd_ptr <= 0;
✓ 103 count<=0;
✓ 104 underflow<=0;
✓ 105 overflow<=0;
✓ 106 wr_ack<=0;
✓ 110 count <= count + 1;
✓ 112 count <= count - 1;
✓ 119 assign full = (count == FIFO_DEPTH)? 1 : 0;
✓ 120 assign empty = (count == 0)? 1 : 0;
✓ 121 assign almostfull = (count == (FIFO_DEPTH-1))? 1 : 0; //should be equal to 7 not 6
✓ 122 assign almostempty = (count == 1)? 1 : 0;
```

2- Toggles Coverage :



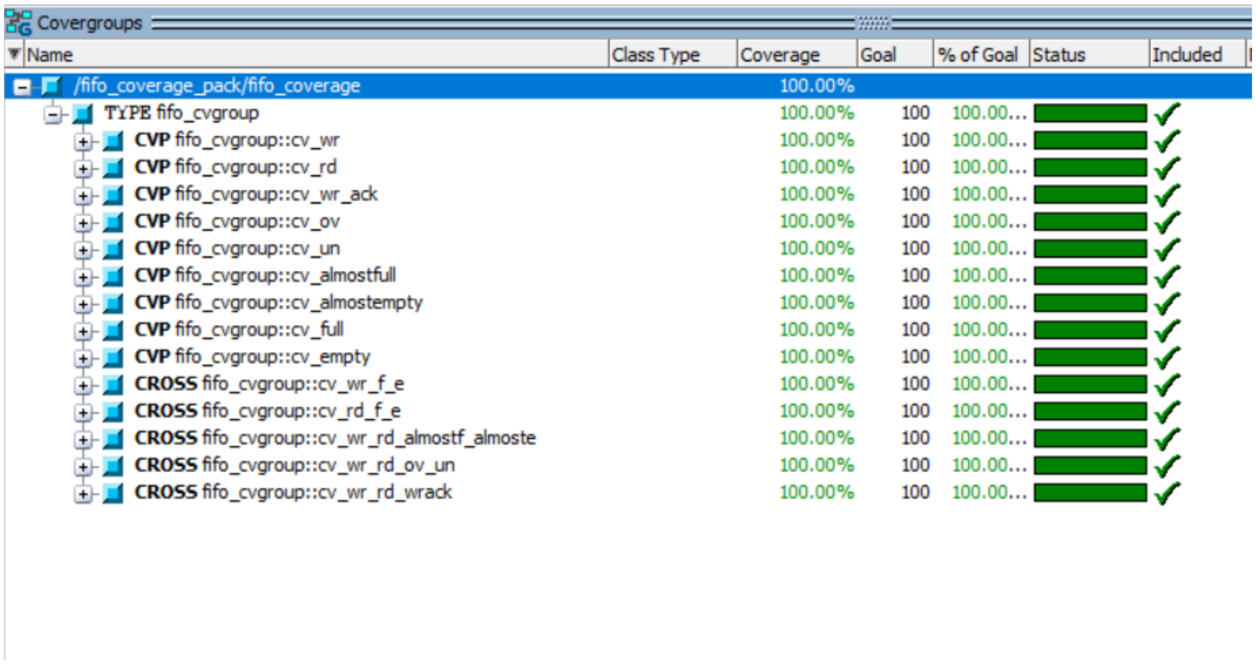
```
Toggles - by instance (/FIFO_TOP/fifo_dut)

sim:/FIFO_TOP/fifo_dut
  ✓ almostempty
  ✓ almostfull
  ✓ clk
  ✓ count
  ✓ data_in
  ✓ data_out
  ✓ empty
  ✓ full
  ✓ overflow
  ✓ rd_en
  ✓ rd_ptr
  ✓ rst_n
  ✓ underflow
  ✓ wr_ack
  ✓ wr_en
  ✓ wr_ptr
```

3- Branches Coverage:



```
Branches - by instance (/FIFO_TOP/fifo_dut)

FIFO.sv
  ✓  39 if (!rst_n) begin    ///// reseting all the registers /////
  ✓  47 else if ((wr_en && (count < FIFO_DEPTH)) || (rd_en && (count == 0) && wr_en)) begin
  ✓  54 else begin
  ✓  57 if (full & wr_en)begin
  ✓  62 else begin
  ✓  73 if (!rst_n) begin   ///// reseting all the registers /////
  ✓  81 else if ((rd_en && (count > 0) &&(!wr_en))  || (rd_en && (count == FIFO_DEPTH) &&wr_en)) begin
  ✓  85 else begin
  ✓  86 if (empty & rd_en)begin
  ✓  90 else begin
  ✓ 100 if (!rst_n) begin    ///// reseting all the registers /////
  ✓ 108 else begin
  ✓ 109 if        ((wr_en && (count < FIFO_DEPTH)) || (rd_en && (count == 0) && wr_en))//including the condition when read and write are high and the memory is empty
  ✓ 111 else if ((rd_en && (count > 0) &&(!wr_en))|| (rd_en && (count ==FIFO_DEPTH)&&wr_en))//including the condition when read and write are high and the memory is full
  ✓ 119 assign full = (count == FIFO_DEPTH)? 1 : 0;
  ✓ 120 assign empty = (count == 0)? 1 : 0;
  ✓ 121 assign almostfull = (count == (FIFO_DEPTH-1))? 1 : 0; //should be equal to 7 not 6
  ✓ 122 assign almostempty = (count == 1)? 1 : 0;
FIFO_TOP.sv
```

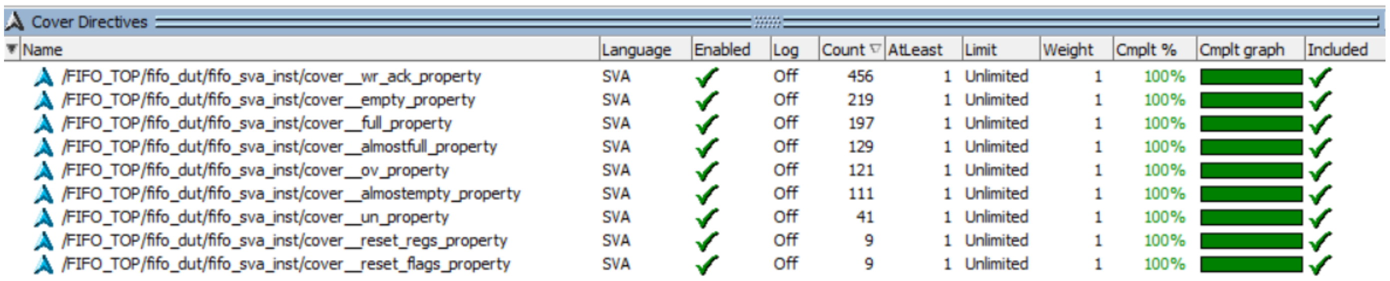
➔ Functional Coverage :



Covergroups

Name	Class Type	Coverage	Goal	% of Goal	Status	Included
/fifo_coverage_pack/fifo_coverage		100.00%				
TYPE fifo_cvgroup		100.00%	100	100.00...	🟩	✓
CVP fifo_cvgroup::cv_wr		100.00%	100	100.00...	🟩	✓
CVP fifo_cvgroup::cv_rd		100.00%	100	100.00...	🟩	✓
CVP fifo_cvgroup::cv_wr_ack		100.00%	100	100.00...	🟩	✓
CVP fifo_cvgroup::cv_ov		100.00%	100	100.00...	🟩	✓
CVP fifo_cvgroup::cv_un		100.00%	100	100.00...	🟩	✓
CVP fifo_cvgroup::cv_almostfull		100.00%	100	100.00...	🟩	✓
CVP fifo_cvgroup::cv_almostempty		100.00%	100	100.00...	🟩	✓
CVP fifo_cvgroup::cv_full		100.00%	100	100.00...	🟩	✓
CVP fifo_cvgroup::cv_empty		100.00%	100	100.00...	🟩	✓
CROSS fifo_cvgroup::cv_wr_f_e		100.00%	100	100.00...	🟩	✓
CROSS fifo_cvgroup::cv_rd_f_e		100.00%	100	100.00...	🟩	✓
CROSS fifo_cvgroup::cv_wr_rd_almostf_almoste		100.00%	100	100.00...	🟩	✓
CROSS fifo_cvgroup::cv_wr_rd_ov_un		100.00%	100	100.00...	🟩	✓
CROSS fifo_cvgroup::cv_wr_rd_wrack		100.00%	100	100.00...	🟩	✓

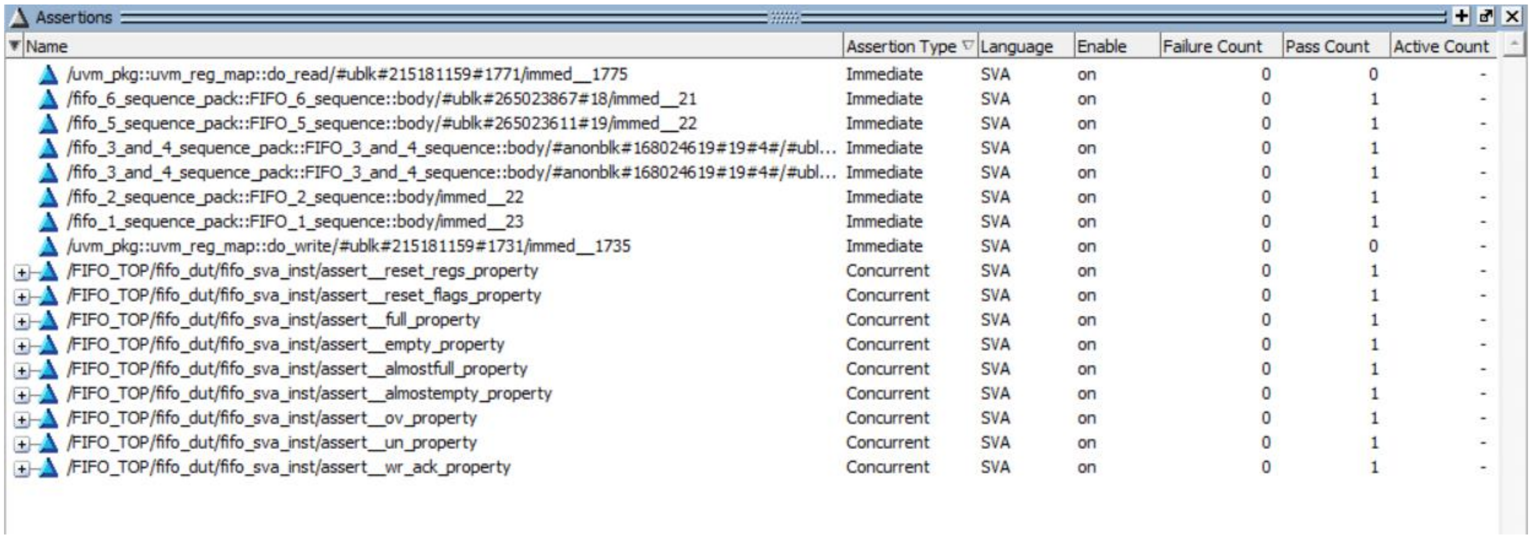
➔ Assertions Coverage :



Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included
/FIFO_TOP/fifo_dut/fifo_sva_inst/cover__wr_ack_property	SVA	✓	Off	456	1	Unlimited	1	100%	🟩	✓
/FIFO_TOP/fifo_dut/fifo_sva_inst/cover__empty_property	SVA	✓	Off	219	1	Unlimited	1	100%	🟩	✓
/FIFO_TOP/fifo_dut/fifo_sva_inst/cover__full_property	SVA	✓	Off	197	1	Unlimited	1	100%	🟩	✓
/FIFO_TOP/fifo_dut/fifo_sva_inst/cover__almostfull_property	SVA	✓	Off	129	1	Unlimited	1	100%	🟩	✓
/FIFO_TOP/fifo_dut/fifo_sva_inst/cover__ov_property	SVA	✓	Off	121	1	Unlimited	1	100%	🟩	✓
/FIFO_TOP/fifo_dut/fifo_sva_inst/cover__almostempty_property	SVA	✓	Off	111	1	Unlimited	1	100%	🟩	✓
/FIFO_TOP/fifo_dut/fifo_sva_inst/cover__un_property	SVA	✓	Off	41	1	Unlimited	1	100%	🟩	✓
/FIFO_TOP/fifo_dut/fifo_sva_inst/cover__reset_regs_property	SVA	✓	Off	9	1	Unlimited	1	100%	🟩	✓
/FIFO_TOP/fifo_dut/fifo_sva_inst/cover__reset_flags_property	SVA	✓	Off	9	1	Unlimited	1	100%	🟩	✓

➔Assertions:



Assertions

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count
/uvm_pkg::uvm_reg_map::do_read/#ublk#215181159#1771/immed__1775	Immediate	SVA	on	0	0	-
/fifo_6_sequence_pack::FIFO_6_sequence::body/#ublk#265023867#18/immed__21	Immediate	SVA	on	0	1	-
/fifo_5_sequence_pack::FIFO_5_sequence::body/#ublk#265023611#19/immed__22	Immediate	SVA	on	0	1	-
/fifo_3_and_4_sequence_pack::FIFO_3_and_4_sequence::body/#anonblk#168024619#19#4#/#ubl...	Immediate	SVA	on	0	1	-
/fifo_3_and_4_sequence_pack::FIFO_3_and_4_sequence::body/#anonblk#168024619#19#4#/#ubl...	Immediate	SVA	on	0	1	-
/fifo_2_sequence_pack::FIFO_2_sequence::body/immed__22	Immediate	SVA	on	0	1	-
/fifo_1_sequence_pack::FIFO_1_sequence::body/immed__23	Immediate	SVA	on	0	1	-
/uvm_pkg::uvm_reg_map::do_write/#ublk#215181159#1731/immed__1735	Immediate	SVA	on	0	0	-
/FIFO_TOP/fifo_dut/fifo_sva_inst/assert__reset_regs_property	Concurrent	SVA	on	0	1	-
/FIFO_TOP/fifo_dut/fifo_sva_inst/assert__reset_flags_property	Concurrent	SVA	on	0	1	-
/FIFO_TOP/fifo_dut/fifo_sva_inst/assert__full_property	Concurrent	SVA	on	0	1	-
/FIFO_TOP/fifo_dut/fifo_sva_inst/assert__empty_property	Concurrent	SVA	on	0	1	-
/FIFO_TOP/fifo_dut/fifo_sva_inst/assert__almostfull_property	Concurrent	SVA	on	0	1	-
/FIFO_TOP/fifo_dut/fifo_sva_inst/assert__almostempty_property	Concurrent	SVA	on	0	1	-
/FIFO_TOP/fifo_dut/fifo_sva_inst/assert__ov_property	Concurrent	SVA	on	0	1	-
/FIFO_TOP/fifo_dut/fifo_sva_inst/assert__un_property	Concurrent	SVA	on	0	1	-
/FIFO_TOP/fifo_dut/fifo_sva_inst/assert__wr_ack_property	Concurrent	SVA	on	0	1	-

➔Assertions Table :

Feature	Assertions
Whenever the rst_n is asserted,The internal flags (full,empty,almostfull,almostempty) always be 0	@(posedge clk) disable iff(rst_n) (rst_n == 0) |=> (full == 0 && empty== 1 && almostfull== 0 && almostempty== 0);
Whenever the rst_n is asserted,The internal registers(overflow,underflow,wr_ack,wr_ptr,rd_ptr,count) always be 0	@(posedge clk) disable iff(rst_n) (rst_n == 0) |=> (overflow == 0 && underflow== 0 && wr_ack== 0 && wr_ptr== 0 && rd_ptr== 0 && count== 0);
Whenever the count equals the FIFO_DEPTH , The full flag should be 1	@(posedge clk) disable iff(!rst_n) (count == FIFO_DEPTH) |-> (full == 1);
Whenever the count equals 0 , The empty flag should be 1	@(posedge clk) (count== 0) |-> (empty== 1);
Whenever the count equals (FIFO_DEPTH-1) , The almostfull flag should be 1	@(posedge clk) disable iff(!rst_n)(count == FIFO_DEPTH-1) |-> (almostfull== 1);
Whenever the count equals 1 , The almostempty flag should be 1	@(posedge clk) disable iff(!rst_n)(count == 1) |-> (almostempty== 1);
Whenever the count equals FIFO_DEPTH && the wr_en equals 1, The overflow flag should be 1	@(posedge clk) disable iff(!rst_n)(count == FIFO_DEPTH &&(wr_en == 1)) |=> (overflow== 1);
Whenever the count equals 0 && the rd_en equals 1, The underflow flag should be 1	@(posedge clk) disable iff(!rst_n)(count== 0 &&(rd_en == 1)) |=> (underflow== 1);
Whenever the wr_en equals 1 & (count < FIFO_DEPTH) Or wr_en & rd_en are 1 & the memory is empty	@(posedge clk) disable iff(!rst_n) ((wr_en && (count < FIFO_DEPTH)) || (rd_en && (count == 0) && wr_en)) |=> (wr_ack== 1);

➔ Coverage Report:

```
Coverage Report by instance with details

=====================================================================================
=== Instance: /\FIFO_TOP#fifo_dut /fifo_sva_inst
=== Design Unit: work.fifo_sva
=====================================================================================

Assertion Coverage:
  Assertions                9       9       0   100.00%
--------------------------------------------------------------------
Name             File(Line)               Failure     Pass
                              Count      Count
--------------------------------------------------------------------
/\FIFO_TOP#fifo_dut /fifo_sva_inst/assert__wr_ack_property
             FIFO_SVA.sv(102)              0        1
/\FIFO_TOP#fifo_dut /fifo_sva_inst/assert__un_property
             FIFO_SVA.sv(100)              0        1
/\FIFO_TOP#fifo_dut /fifo_sva_inst/assert__ov_property
             FIFO_SVA.sv(99)               0        1
/\FIFO_TOP#fifo_dut /fifo_sva_inst/assert__almostempty_property
             FIFO_SVA.sv(97)               0        1
/\FIFO_TOP#fifo_dut /fifo_sva_inst/assert__almostfull_property
             FIFO_SVA.sv(96)               0        1
/\FIFO_TOP#fifo_dut /fifo_sva_inst/assert__empty_property
             FIFO_SVA.sv(94)               0        1
/\FIFO_TOP#fifo_dut /fifo_sva_inst/assert__full_property
             FIFO_SVA.sv(93)               0        1
/\FIFO_TOP#fifo_dut /fifo_sva_inst/assert__reset_flags_property
             FIFO_SVA.sv(91)               0        1
/\FIFO_TOP#fifo_dut /fifo_sva_inst/assert__reset_regs_property
             FIFO_SVA.sv(90)               0        1
```

```
Directive Coverage:
   Directives                 9       9       0   100.00%

DIRECTIVE COVERAGE:
-----------------------------------------------------------------------------------------
Name                              Design Design   Lang File(Line)      Hits Status
                             Unit   UnitType
-----------------------------------------------------------------------------------------
/\FIFO_TOP#fifo_dut /fifo_sva_inst/cover__wr_ack_property
                             fifo_sva Verilog  SVA  FIFO_SVA.sv(102) 456 Covered
/\FIFO_TOP#fifo_dut /fifo_sva_inst/cover__un_property
                             fifo_sva Verilog  SVA  FIFO_SVA.sv(100)  41 Covered
/\FIFO_TOP#fifo_dut /fifo_sva_inst/cover__ov_property
                             fifo_sva Verilog  SVA  FIFO_SVA.sv(99)  121 Covered
/\FIFO_TOP#fifo_dut /fifo_sva_inst/cover__almostempty_property
                             fifo_sva Verilog  SVA  FIFO_SVA.sv(97)  111 Covered
/\FIFO_TOP#fifo_dut /fifo_sva_inst/cover__almostfull_property
                             fifo_sva Verilog  SVA  FIFO_SVA.sv(96)  129 Covered
/\FIFO_TOP#fifo_dut /fifo_sva_inst/cover__empty_property
                             fifo_sva Verilog  SVA  FIFO_SVA.sv(94)  219 Covered
/\FIFO_TOP#fifo_dut /fifo_sva_inst/cover__full_property
                             fifo_sva Verilog  SVA  FIFO_SVA.sv(93)  197 Covered
/\FIFO_TOP#fifo_dut /fifo_sva_inst/cover__reset_flags_property
                             fifo_sva Verilog  SVA  FIFO_SVA.sv(91)    9 Covered
/\FIFO_TOP#fifo_dut /fifo_sva_inst/cover__reset_regs_property
                             fifo_sva Verilog  SVA  FIFO_SVA.sv(90)    9 Covered
```

```
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------            ----      ----    ------  --------
    Statements                    16        16       0   100.00%

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

Statement Coverage for instance /\FIFO_TOP#fifo_dut /fifo_sva_inst --

    Line         Item                      Count     Source
    ----         ----                     -----     ------
  File FIFO_SVA.sv
    3                                  module fifo_sva (FIFO_INTERF.FIFO_DUT_MODE fifo_interface);
    4                                  import fifo_pack::*;
    5                                  bit clk;
    6
    7                                  logic [(FIFO_WIDTH-1):0] data_in;
    8                                  bit rst_n, wr_en, rd_en;
    9                                  logic  [(FIFO_WIDTH-1):0] data_out;
    10                                  logic wr_ack, overflow;
    11                                  bit full, empty, almostfull, almostempty;
    12                                  logic underflow;
    13
    14
    15                                  localparam max_fifo_addr = $clog2(FIFO_DEPTH);
    16                                  logic [max_fifo_addr-1:0] wr_ptr, rd_ptr;
    17                                  logic [max_fifo_addr:0] count;
    18
    19           1                   2046     assign clk=fifo_interface.clk;
    20           1                    168     assign rst_n=fifo_interface.rst_n;
    21
    22           1                   1023     assign data_in=fifo_interface.data_in;
    23           1                    132     assign data_out=fifo_interface.data_out;
    24
    25           1                    444     assign wr_en=fifo_interface.wr_en;
    26           1                    479     assign wr_ack=fifo_interface.wr_ack;
    27           1                    432     assign rd_en=fifo_interface.rd_en;
    28
    29           1                    122     assign overflow=fifo_interface.overflow;
    30           1                     82     assign underflow=fifo_interface.underflow;
    31
    32           1                    167     assign full=fifo_interface.full;
    33           1                    185     assign empty=fifo_interface.empty;
    34
    35           1                    213     assign almostfull=fifo_interface.almostfull;
    36           1                    205     assign almostempty=fifo_interface.almostempty;
    37
    38          1                712    assign count=fifo_interface.count;
    39
    40          1                576    assign wr_ptr=fifo_interface.wr_ptr;
    41          1                181    assign rd_ptr=fifo_interface.rd_ptr;

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

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

Toggle Coverage for instance /\FIFO_TOP#fifo_dut /fifo_sva_inst --

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

Total Node Count     =         53
Toggled Node Count   =         53
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (106 of 106 bins)
```

```
=====================================================================================
=== Instance: /\FIFO_TOP#fifo_dut
=== Design Unit: work.FIFO
=====================================================================================
Branch Coverage:
  Enabled Coverage          Bins     Hits    Misses  Coverage
  ----------------          ----     ----    ------  --------
  Branches                    23       23         0   100.00%

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

Branch Coverage for instance /\FIFO_TOP#fifo_dut

  Line       Item                   Count    Source
  ----       ----                   -----    ------
 File FIFO.sv
------------------------------------IF Branch------------------------------------
  39                                 1105    Count coming in to IF
  39           1                      176      if (!rst_n) begin   ///// reseting all the registers /////
  47           1                      502      else if ((wr_en && (count < FIFO_DEPTH)) || (rd_en && (count == 0) && wr_en)) begin
  54           1                      427      else begin
Branch totals: 3 hits of 3 branches = 100.00%

------------------------------------IF Branch------------------------------------
  57                                  427    Count coming in to IF
  57           1                      130                  if (full & wr_en)begin
  62           1                      297                  else begin
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
  73                                 1081    Count coming in to IF
  73           1                      173      if (!rst_n) begin   ///// reseting all the registers /////
  81           1                      131      else if ((rd_en && (count > 0) &&(!wr_en))  || (rd_en && (count == FIFO_DEPTH) &&wr_en))
begin
  85           1                      777      else begin
Branch totals: 3 hits of 3 branches = 100.00%

------------------------------------IF Branch------------------------------------
  86                                  777    Count coming in to IF
  86           1                       44                  if (empty & rd_en)begin
  90           1                      733                  else begin
Branch totals: 2 hits of 2 branches = 100.00%
```

```
------------------------------------IF Branch------------------------------------
   100                          1081    Count coming in to IF
   100          1                173    if (!rst_n) begin    ///// reseting all the registers /////
   108          1                908    else begin
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
   109                           908    Count coming in to IF
   109          1                502                if          ((wr_en && (count < FIFO_DEPTH)) || (rd_en && (count == 0) &&
wr_en))//including the condition when read and write are high and the memory is empty
   111          1                131                else if ((rd_en && (count > 0) &&(!wr_en))|| (rd_en && (count
==FIFO_DEPTH)&&wr_en))//including the condition when read and write are high and the memory is full
                               275    All False Count
Branch totals: 3 hits of 3 branches = 100.00%

------------------------------------IF Branch------------------------------------
   119                           711    Count coming in to IF
   119          1                 83    assign full = (count == FIFO_DEPTH)? 1 : 0;
   119          2                628     assign full = (count == FIFO_DEPTH)? 1 : 0;
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
   120                           711    Count coming in to IF
   120          1                 92    assign empty = (count == 0)? 1 : 0;
   120          2                619     assign empty = (count == 0)? 1 : 0;
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
   121                           711    Count coming in to IF
   121          1                106     assign almostfull = (count == (FIFO_DEPTH-1))? 1 : 0; //should be equal to 7 not 6
   121          2                605     assign almostfull = (count == (FIFO_DEPTH-1))? 1 : 0; //should be equal to 7 not 6
Branch totals: 2 hits of 2 branches = 100.00%

------------------------------------IF Branch------------------------------------
   122                           711    Count coming in to IF
   122          1                102     assign almostempty = (count == 1)? 1 : 0;
   122          2                609     assign almostempty = (count == 1)? 1 : 0;
Branch totals: 2 hits of 2 branches = 100.00%
```

```
Condition Coverage:
   Enabled Coverage          Bins   Covered   Misses  Coverage
   ----------------          ----     ----    ------  --------
   Conditions                  10       10       0   100.00%

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

Condition Coverage for instance /\FIFO_TOP#fifo_dut  --

 File FIFO.sv
-----------Focused Condition View (Bimodal)--------------
Line      81 Item    1  (((rd_en && (count > 0)) && ~wr_en) || ((rd_en && (count == 8)) && wr_en))
Condition totals: 4 of 4 input terms covered = 100.00%

     Input Term   Covered  Reason for no coverage              Hint
    -----------  --------  ---------------------------------  --------------
        rd_en       Y
    (count > 0)       Y
        wr_en       Y
   (count == 8)       Y

   Rows:   Hits(->0)   Hits(->1)  FEC Target          Non-masking condition(s)

---------  ----------  ----------  -------------------  -------------------------
 Row   1:         1         0  rd_en_0            -
 Row   2:         0         1  rd_en_1            (~wr_en && (count > 0)), (wr_en && (count == 8))
 Row   3:         1         0  (count > 0)_0       (~((rd_en && (count == 8)) && wr_en) && rd_en)
 Row   4:         0         1  (count > 0)_1       (~wr_en && rd_en)
 Row   5:         0         1  wr_en_0            (rd_en && (count > 0)), (~((rd_en && (count > 0)) && ~wr_en) && (rd_en && (count =
8)))
 Row   6:         1         1  wr_en_1            (~((rd_en && (count == 8)) && wr_en) && (rd_en && (count > 0))), (rd_en && (count
8))
 Row   7:         1         0  (count == 8)_0       (~((rd_en && (count > 0)) && ~wr_en) && rd_en)
 Row   8:         0         1  (count == 8)_1       (wr_en && rd_en)

----------------Focused Condition View-------------------
Line      86 Item    1  (empty & rd_en)
Condition totals: 2 of 2 input terms covered = 100.00%

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

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

----------------Focused Condition View-------------------
Line      119 Item    1  (count == 8)
Condition totals: 1 of 1 input term covered = 100.00%

   Input Term   Covered  Reason for no coverage   Hint
  -----------  --------  ----------------------  --------------
 (count == 8)        Y

   Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
 Row   1:        1  (count == 8)_0       -
 Row   2:        1  (count == 8)_1       -

----------------Focused Condition View-------------------
Line      120 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      121 Item    1  (count == (8 - 1))
Condition totals: 1 of 1 input term covered = 100.00%

       Input Term   Covered  Reason for no coverage   Hint
      -----------  --------  ----------------------  --------------
 (count == (8 - 1))        Y

   Rows:       Hits  FEC Target            Non-masking condition(s)
 ---------  ---------  --------------------  -------------------------
 Row   1:        1  (count == (8 - 1))_0  -
 Row   2:        1  (count == (8 - 1))_1  -

----------------Focused Condition View-------------------
Line      122 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 Coverage:
    Enabled Coverage            Bins      Hits    Misses  Coverage
    ----------------            ----      ----    ------  --------
    Statements                    42        42         0   100.00%

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

Statement Coverage for instance /\FIFO_TOP#fifo_dut  --

    Line         Item                 Count     Source
    ----         ----                 -----     ------
  File FIFO.sv
    4                                           module FIFO(FIFO_INTERF.FIFO_DUT_MODE fifo_interf);
    5
    6                                           logic [(FIFO_WIDTH-1):0]data_in;
    7                                           logic wr_ack,overflow,underflow;
    8                                           logic [(FIFO_WIDTH-1):0]data_out;
    9                                           bit clk,rst_n,wr_en,rd_en,full,empty,almostfull,almostempty;
    10
    11              1                  1023     assign data_in=fifo_interf.data_in;
    12              1                  2046     assign clk=fifo_interf.clk;
    13              1                   168     assign rst_n=fifo_interf.rst_n;
    14              1                   444     assign wr_en=fifo_interf.wr_en;
    15              1                   432     assign rd_en=fifo_interf.rd_en;
    16
    17                                          assign fifo_interf.wr_ack=wr_ack;
    18                                          assign fifo_interf.overflow=overflow;
    19                                          assign fifo_interf.full=full;
    20                                          assign fifo_interf.empty=empty;
    21                                          assign fifo_interf.almostfull=almostfull;
    22                                          assign fifo_interf.almostempty=almostempty;
```

```
23                               assign fifo_interf.underflow=underflow;
24                               assign fifo_interf.data_out=data_out;
25
26
27                               localparam max_fifo_addr = $clog2(FIFO_DEPTH);
28                               logic [max_fifo_addr-1:0] wr_ptr, rd_ptr;
29                               logic [max_fifo_addr:0] count;
30
31                               assign fifo_interf.wr_ptr=wr_ptr;
32                               assign fifo_interf.rd_ptr=rd_ptr;
33                               assign fifo_interf.count=count;
34
35                               logic [FIFO_WIDTH-1:0] mem [FIFO_DEPTH-1:0];
36                               ////////////////////////////////////////////////////////
37
38       1          1105            always @(posedge clk or negedge rst_n) begin//always block of the writing
39                                  if (!rst_n) begin    ///// reseting all the registers /////
40       1           176                    wr_ptr <= 0;
41       1           176                    rd_ptr <= 0;
42       1           176                    count<=0;
43       1           176                    underflow<=0;
44       1           176                    overflow<=0;
45       1           176                    wr_ack<=0;
46                                  end
47                                  else if ((wr_en && (count < FIFO_DEPTH)) || (rd_en && (count == 0) && wr_en)) begin
48
49       1           502                    mem[wr_ptr] <= data_in;
50       1           502                    wr_ack <= 1;
51       1           502                    wr_ptr <= wr_ptr + 1;
52
53                                  end
54                                  else begin
55       1           427                    wr_ack <= 0;
56
57                                          if (full & wr_en)begin
58       1           130                            overflow <= 1;
59
60                                          end
61
62                                          else begin
63       1           297                            overflow <= 0;
64
65                                          end
66
67                                  end
68                               end
69
70                               ////////////////////////////////////////////////////////
```

```
  71
  72     1        1081   always @(posedge clk or negedge rst_n) begin//always block of the reading
  73                     if (!rst_n) begin   ///// reseting all the registers /////
  74     1         173           wr_ptr <= 0;
  75     1         173           rd_ptr <= 0;
  76     1         173           count<=0;
  77     1         173           underflow<=0;
  78     1         173           overflow<=0;
  79     1         173           wr_ack<=0;
  80                     end
  81                     else if ((rd_en && (count > 0) &&(!wr_en))  || (rd_en && (count == FIFO_DEPTH) &&wr_en))
begin
  82     1         131           data_out <= mem[rd_ptr];
  83     1         131           rd_ptr <= rd_ptr + 1;
  84                     end
  85                     else begin
  86                             if (empty & rd_en)begin
  87     1          44                   underflow <= 1;
  88                             end
  89
  90                             else begin
  91     1         733                   underflow <= 0;
  92                             end
  93
  94                     end
  95                 end
  96
  97                 ////////////////////////////////////////////////////////
  98
  99     1        1081   always @(posedge clk or negedge rst_n) begin//always block of the counting
  100                    if (!rst_n) begin   ///// reseting all the registers /////
  101     1        173           wr_ptr <= 0;
  102     1        173           rd_ptr <= 0;
  103     1        173           count<=0;
  104     1        173           underflow<=0;
  105     1        173           overflow<=0;
  106     1        173           wr_ack<=0;
  107                    end
  108                    else begin
  109                            if       ((wr_en && (count < FIFO_DEPTH)) || (rd_en && (count == 0) &&
wr_en))//including the condition when read and write are high and the memory is empty
  110     1        502                   count <= count + 1;
  111                            else if ((rd_en && (count > 0) &&(!wr_en))|| (rd_en && (count
==FIFO_DEPTH)&&wr_en))//including the condition when read and write are high and the memory is full
  112     1        131                   count <= count - 1;
  113                    end
  114                 end
  115
  116                             /////////////////////////////////////////////////////
  117
  118
  119        1            712    assign full = (count == FIFO_DEPTH)? 1 : 0;
  120        1            712    assign empty = (count == 0)? 1 : 0;
  121        1            712    assign almostfull = (count == (FIFO_DEPTH-1))? 1 : 0; //should be equal to 7 not 6
  122        1            712    assign almostempty = (count == 1)? 1 : 0;
```

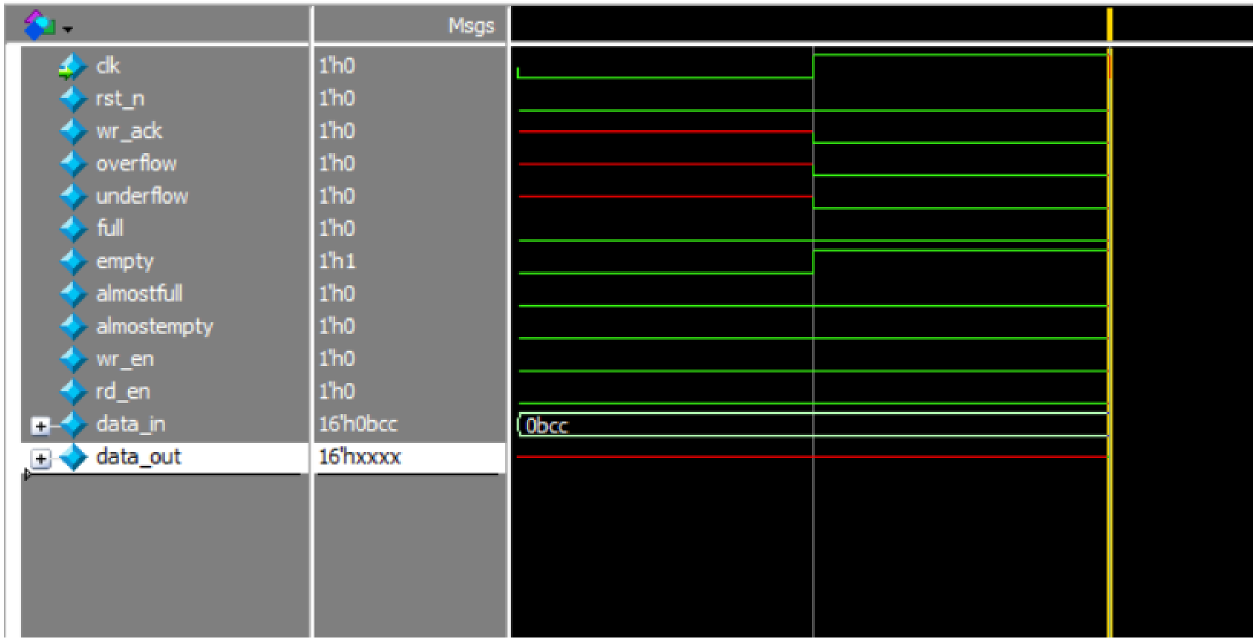
```
Toggle Coverage:
  Enabled Coverage          Bins     Hits   Misses  Coverage
  ----------------          ----     ----   ------  --------
  Toggles                    106      106        0  100.00%

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

Toggle Coverage for instance /\FIFO_TOP#fifo_dut --

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

Total Node Count     =        53
Toggled Node Count   =        53
Untoggled Node Count =         0

Toggle Coverage      =    100.00% (106 of 106 bins)
```

```
DIRECTIVE COVERAGE:
--------------------------------------------------------------------------------------------
Name                          Design Design   Lang File(Line)      Hits Status
                              Unit   UnitType
--------------------------------------------------------------------------------------------
/\FIFO_TOP#fifo_dut /fifo_sva_inst/cover__wr_ack_property
                              fifo_sva Verilog  SVA  FIFO_SVA.sv(102) 456 Covered
/\FIFO_TOP#fifo_dut /fifo_sva_inst/cover__un_property
                              fifo_sva Verilog  SVA  FIFO_SVA.sv(100)  41 Covered
/\FIFO_TOP#fifo_dut /fifo_sva_inst/cover__ov_property
                              fifo_sva Verilog  SVA  FIFO_SVA.sv(99)  121 Covered
/\FIFO_TOP#fifo_dut /fifo_sva_inst/cover__almostempty_property
                              fifo_sva Verilog  SVA  FIFO_SVA.sv(97)  111 Covered
/\FIFO_TOP#fifo_dut /fifo_sva_inst/cover__almostfull_property
                              fifo_sva Verilog  SVA  FIFO_SVA.sv(96)  129 Covered
/\FIFO_TOP#fifo_dut /fifo_sva_inst/cover__empty_property
                              fifo_sva Verilog  SVA  FIFO_SVA.sv(94)  219 Covered
/\FIFO_TOP#fifo_dut /fifo_sva_inst/cover__full_property
                              fifo_sva Verilog  SVA  FIFO_SVA.sv(93)  197 Covered
/\FIFO_TOP#fifo_dut /fifo_sva_inst/cover__reset_flags_property
                              fifo_sva Verilog  SVA  FIFO_SVA.sv(91)    9 Covered
/\FIFO_TOP#fifo_dut /fifo_sva_inst/cover__reset_regs_property
                              fifo_sva Verilog  SVA  FIFO_SVA.sv(90)    9 Covered

TOTAL DIRECTIVE COVERAGE: 100.00%  COVERS: 9
```

```
ASSERTION RESULTS:
-------------------------------------------------------------------
Name             File(Line)              Failure    Pass
                                  Count      Count
-------------------------------------------------------------------
/\FIFO_TOP#fifo_dut /fifo_sva_inst/assert__wr_ack_property
              FIFO_SVA.sv(102)              0       1
/\FIFO_TOP#fifo_dut /fifo_sva_inst/assert__un_property
              FIFO_SVA.sv(100)              0       1
/\FIFO_TOP#fifo_dut /fifo_sva_inst/assert__ov_property
              FIFO_SVA.sv(99)              0       1
/\FIFO_TOP#fifo_dut /fifo_sva_inst/assert__almostempty_property
              FIFO_SVA.sv(97)              0       1
/\FIFO_TOP#fifo_dut /fifo_sva_inst/assert__almostfull_property
              FIFO_SVA.sv(96)              0       1
/\FIFO_TOP#fifo_dut /fifo_sva_inst/assert__empty_property
              FIFO_SVA.sv(94)              0       1
/\FIFO_TOP#fifo_dut /fifo_sva_inst/assert__full_property
              FIFO_SVA.sv(93)              0       1
/\FIFO_TOP#fifo_dut /fifo_sva_inst/assert__reset_flags_property
              FIFO_SVA.sv(91)              0       1
/\FIFO_TOP#fifo_dut /fifo_sva_inst/assert__reset_regs_property
              FIFO_SVA.sv(90)              0       1

Total Coverage By Instance (filtered view): 100.00%
```

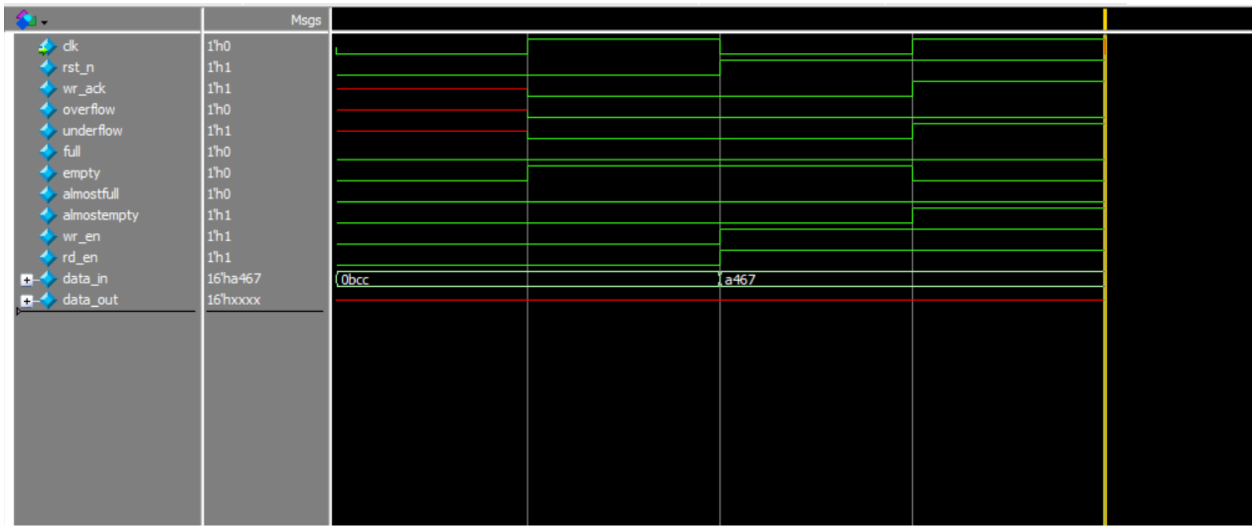
➔ Questasim Waveform Snippets:

1- FIFO_1 Sequence Waveform:



FIFO_1 Sequence Waveform

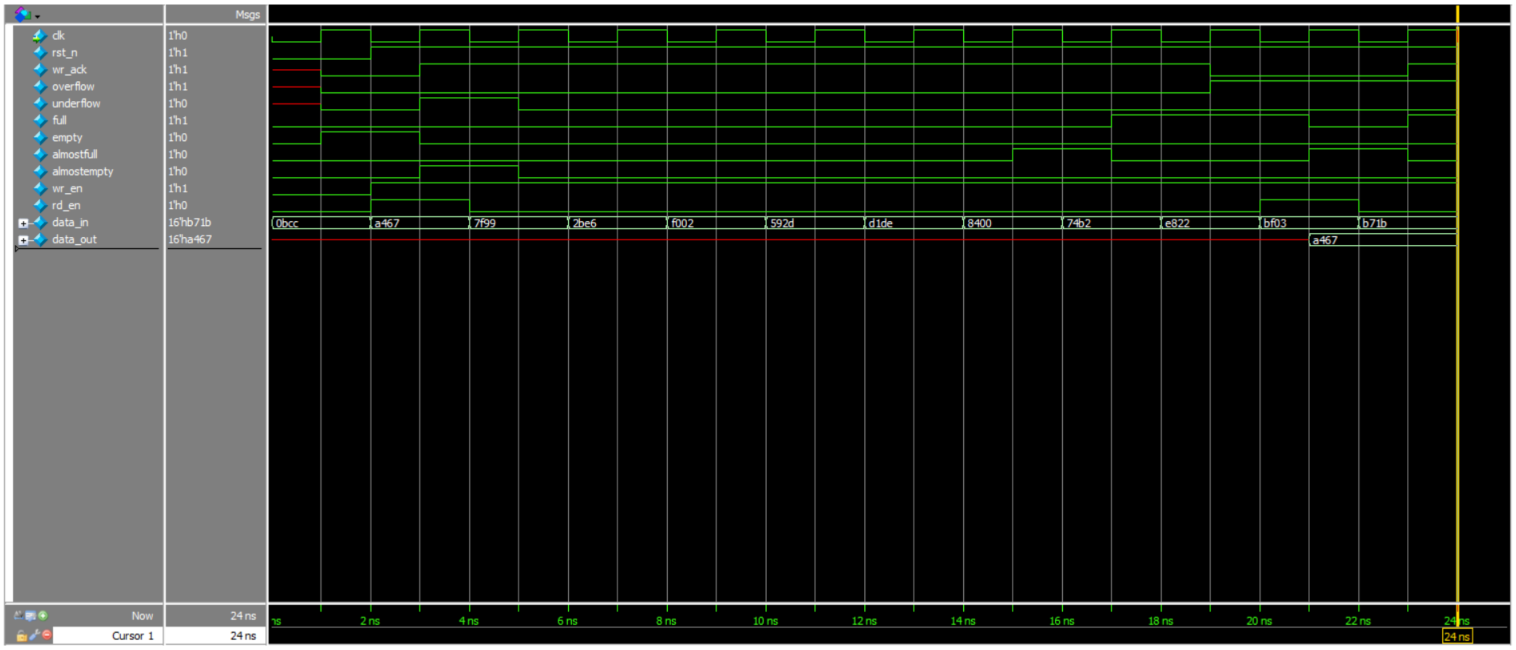
2- FIFO_1 & FIFO_2 Sequence Waveform:
-One Read operation when the memory is empty



FIFO_1 & FIFO_2 Sequence Waveform

3- **FIFO_1 & FIFO_2 & FIFO_3_and_4 Sequence Waveform:**

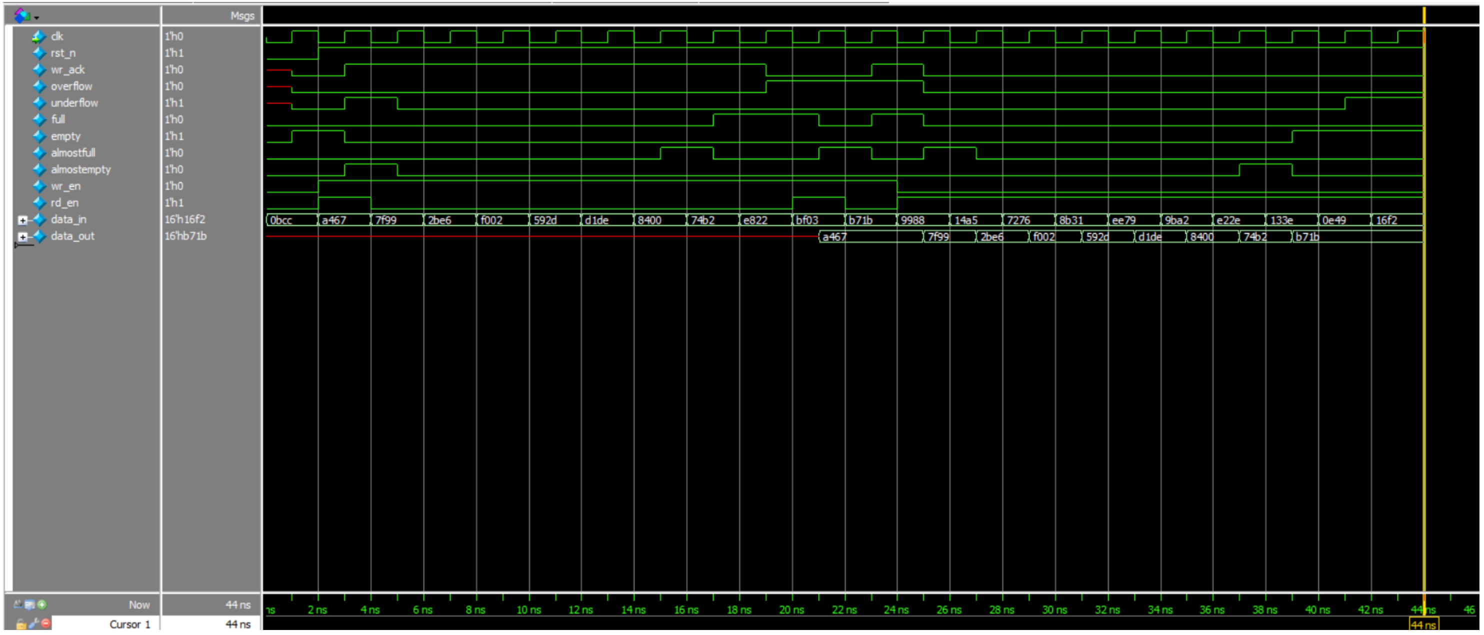
   - **Write only sequence with one Read operation when the memory is full**



FIFO_1 & FIFO_2 & FIFO_3_and_4 Sequence Waveform

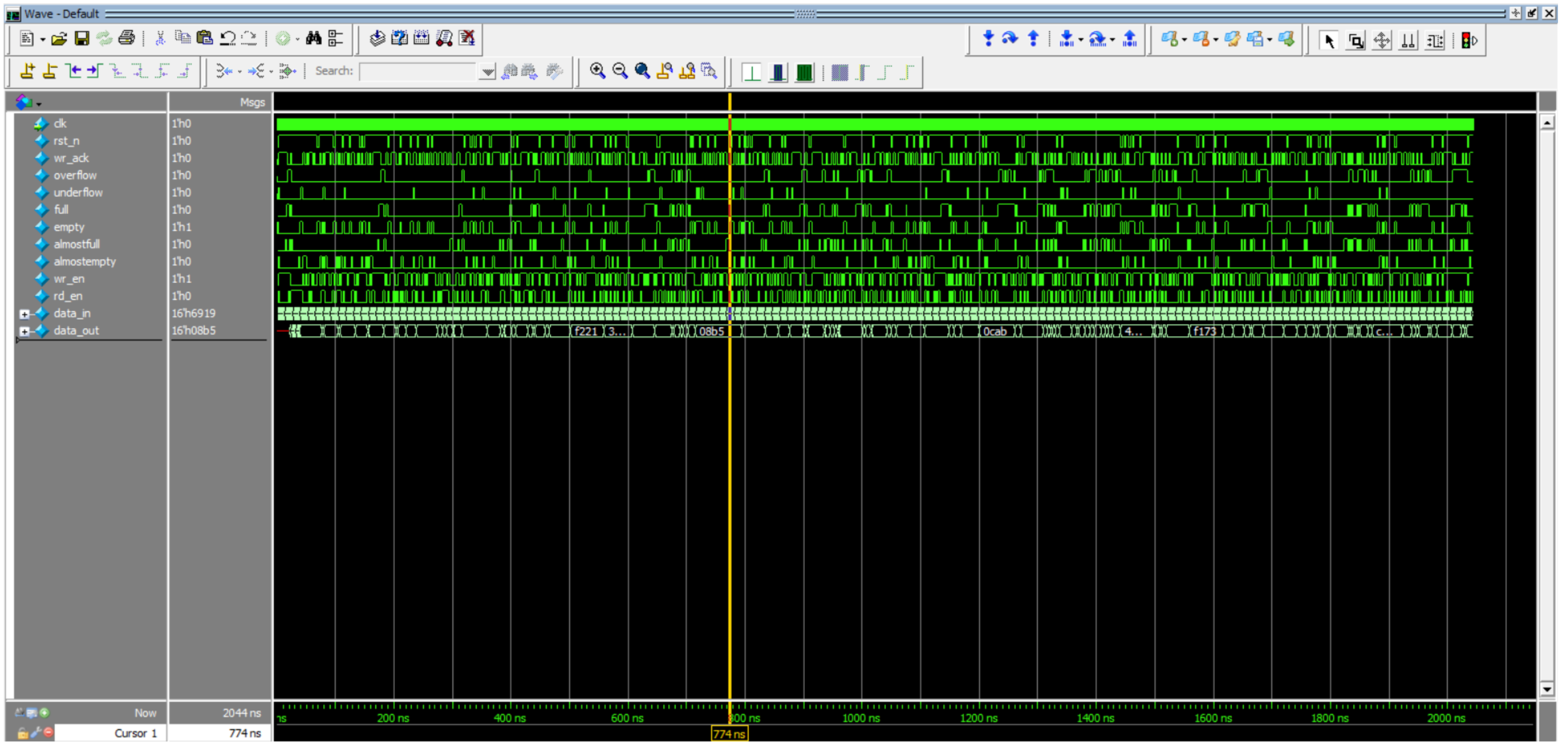
4- **FIFO_1 & FIFO_2 & FIFO_3_and_4 & FIFO_5 Sequence Waveform:**

   - **Read only sequence till the memory become empty**



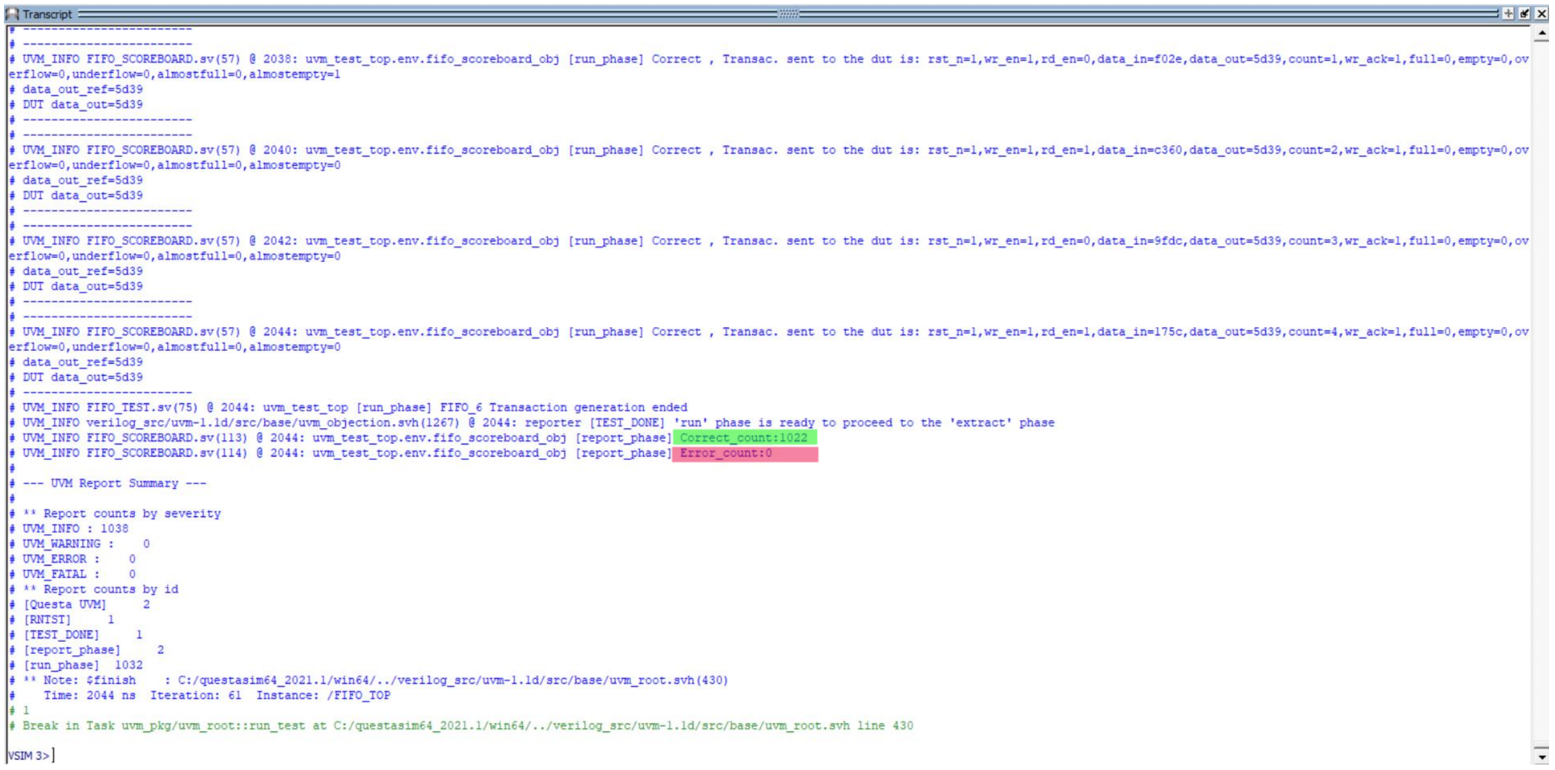
FIFO_1 & FIFO_2 & FIFO_3_and_4 & FIFO_5 Sequence Waveform

5- All the Sequences Waveform:



All Sequences Waveform

➔Transcript:



```
Transcript
# ------------------------
# ------------------------
# UVM_INFO FIFO_SCOREBOARD.sv(57) @ 2038: uvm_test_top.env.fifo_scoreboard_obj [run_phase] Correct , Transac. sent to the dut is: rst_n=1,wr_en=1,rd_en=0,data_in=f02e,data_out=5d39,count=1,wr_ack=1,full=0,empty=0,ov
erflow=0,underflow=0,almostfull=0,almostempty=1
# data_out_ref=5d39
# DUT data_out=5d39
# ------------------------
# ------------------------
# UVM_INFO FIFO_SCOREBOARD.sv(57) @ 2040: uvm_test_top.env.fifo_scoreboard_obj [run_phase] Correct , Transac. sent to the dut is: rst_n=1,wr_en=1,rd_en=1,data_in=c360,data_out=5d39,count=2,wr_ack=1,full=0,empty=0,ov
erflow=0,underflow=0,almostfull=0,almostempty=0
# data_out_ref=5d39
# DUT data_out=5d39
# ------------------------
# ------------------------
# UVM_INFO FIFO_SCOREBOARD.sv(57) @ 2042: uvm_test_top.env.fifo_scoreboard_obj [run_phase] Correct , Transac. sent to the dut is: rst_n=1,wr_en=1,rd_en=0,data_in=9fdc,data_out=5d39,count=3,wr_ack=1,full=0,empty=0,ov
erflow=0,underflow=0,almostfull=0,almostempty=0
# data_out_ref=5d39
# DUT data_out=5d39
# ------------------------
# ------------------------
# UVM_INFO FIFO_SCOREBOARD.sv(57) @ 2044: uvm_test_top.env.fifo_scoreboard_obj [run_phase] Correct , Transac. sent to the dut is: rst_n=1,wr_en=1,rd_en=1,data_in=175c,data_out=5d39,count=4,wr_ack=1,full=0,empty=0,ov
erflow=0,underflow=0,almostfull=0,almostempty=0
# data_out_ref=5d39
# DUT data_out=5d39
# ------------------------
# UVM_INFO FIFO_TEST.sv(75) @ 2044: uvm_test_top [run_phase] FIFO_6 Transaction generation ended
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 2044: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO FIFO_SCOREBOARD.sv(113) @ 2044: uvm_test_top.env.fifo_scoreboard_obj [report_phase] Correct_count:1022
# UVM_INFO FIFO_SCOREBOARD.sv(114) @ 2044: uvm_test_top.env.fifo_scoreboard_obj [report_phase] Error_count:0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO : 1038
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [report_phase]     2
# [run_phase]  1032
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 2044 ns  Iteration: 61  Instance: /FIFO_TOP
# 1
# Break in Task uvm_pkg/uvm_root::run_test at C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430

VSIM 3>
```

Transcript

- **The Correct_Count = 1022 ✓ ✓**
- **The Error_Count = 0 ✓ ✓**