

## FIFO\_design

The image shows two side-by-side windows of a code editor, likely Vivado, displaying System Verilog code for a FIFO module. Both windows have identical titles and file paths: "File Edit Selection View Go Run Terminal Help" and "D:\digital design > session 1 > FIFO.sv". The code is as follows:

```
7 //////////////////////////////////////////////////////////////////
8 module FIFO(FIFO_if.DUT fif);
9 parameter FIFO_WIDTH = 16;
10 parameter FIFO_DEPTH = 8;
11
12 localparam max_fifo_addr = $clog2(FIFO_DEPTH);
13
14 logic [FIFO_WIDTH-1:0] mem [FIFO_DEPTH-1:0];
15
16 logic [max_fifo_addr-1:0] wr_ptr, rd_ptr;
17 logic [max_fifo_addr:0] count;
18
19 always @(posedge fif.clk or negedge fif.rst_n) begin
20     if (!fifo.rst_n) begin
21         wr_ptr <= 0;
22         fif.wr_ack <= 0;
23         fif.overflow<=0;
24     end
25     else if (fifo.wr_en && count < FIFO_DEPTH) begin
26         mem[wr_ptr] <- fifo.data_in;
27         fifo.ack <= 1;
28         wr_ptr <= wr_ptr + 1;
29     end
30     else if(fifo.wr_en && fifo.full) begin
31         fifo.overflow<=1;
32     end
33     else begin
34         fifo.overflow<=0;
35         fifo.wr_ack<=0;
36     end
37 end
38
39 always @(posedge fif.clk or negedge fif.rst_n) begin
40     if (!fifo.rst_n) begin
41         rd_ptr <= 0;
42         fifo.underflow<=0;
43     end
44
45     else if (fifo.rd_en && count != 0) begin
46         fifo.data_out <= mem[rd_ptr];
47         rd_ptr <= rd_ptr + 1;
48     end
49     else if(fifo.rd_en && fifo.empty)begin
50         fifo.underflow<=1;
51     end
52     else begin
53         fifo.underflow<=0;
54     end
55 end
56
57 always @(posedge fif.clk or negedge fif.rst_n) begin
58     if (!fifo.rst_n) begin
59         count <= 0;
60     end
61     else begin
62         if ((fifo.wr_en, fifo.rd_en) == 2'b10) && !fifo.full)
63             count <= count + 1;
64         else if ( (fifo.wr_en, fifo.rd_en) == 2'b01) && !fifo.empty)
65             count <= count - 1;
66     end
67 end
68
69 assign fifo.full = (count == FIFO_DEPTH)? 1 : 0;
70 assign fifo.empty = (count == 0)? 1 : 0;
71 assign fifo.almostfull = (count == FIFO_DEPTH-1)? 1 : 0; //FIFO_DEPTH - 2 is wrong
72 assign fifo.almostempty = (count == 1)? 1 : 0;
73
74
```

The status bar at the bottom of each window indicates "Ln 53, Col 27 Tab Size: 4 UTF-8 CRLF System Verilog". The taskbar at the bottom of the screen shows various icons for system applications.

FIFO.sv

```

D: > digital design > session 1 > FIFO.sv
57  always @(posedge fif.clk or negedge fif.rst_n) begin
61    else begin
64      else if ( (fif.wr_en, fif.rd_en) == 2'b01) && !fif.empty
65        count <= count - 1;
66      end
67    end
68
69    assign fif.full = (count == FIFO_DEPTH)? 1 : 0;
70    assign fif.empty = (count == 0)? 1 : 0;
71    assign fif.almostfull = (count == FIFO_DEPTH-1)? 1 : 0; //FIFO_DEPTH -2 is wrong
72    assign fif.almostempty = (count == 1)? 1 : 0;
73
74
75    assert property(@(posedge fif.clk)(fif.wr_en && fif.full |> fif.overflow));
76
77    assert property(@(posedge fif.clk)(fif.rd_en && fif.empty |> fif.underflow));
78
79    assert property (@(posedge fif.clk) disable iff(!fif.rst_n) (fif.wr_en && fif.almostfull && !fif.rd_en) |> fif.full);
80    assert property (@(posedge fif.clk) disable iff(!fif.rst_n) (fif.almostempty && fif.rd_en && !fif.wr_en) |> fif.empty);
81    assert property(@(posedge fif.clk)((fif.wr_en && !fif.full) |> fif.wr_ack);
82
83    assert property(@(posedge fif.clk)count==FIFO_DEPTH |> fif.full);
84
85    assert property(@(posedge fif.clk)(count==FIFO_DEPTH-1) |> fif.almostfull);
86
87    assert property(@(posedge fif.clk)(count==0) |> fif.empty);
88
89    assert property(@(posedge fif.clk)(count==1) |> fif.almostempty);
90
91  endmodule

```

FIFO\_configuration.sv

```

D: > digital design > session 1 > FIFO_configuration.sv
1 package FIFO_config_package;
2 import uvm_pkg::*;
3 `include "uvm_macros.svh"
4
5 class FIFO_config extends uvm_object;
6   `uvm_object_utils(FIFO_config)
7   virtual FIFO_if FIFO_vif;
8   function new(string name= "FIFO_config");
9     super.new(name);
10    endfunction //new()
11  endclass //FIFO_config extends superClass
12
13 endpackage

```

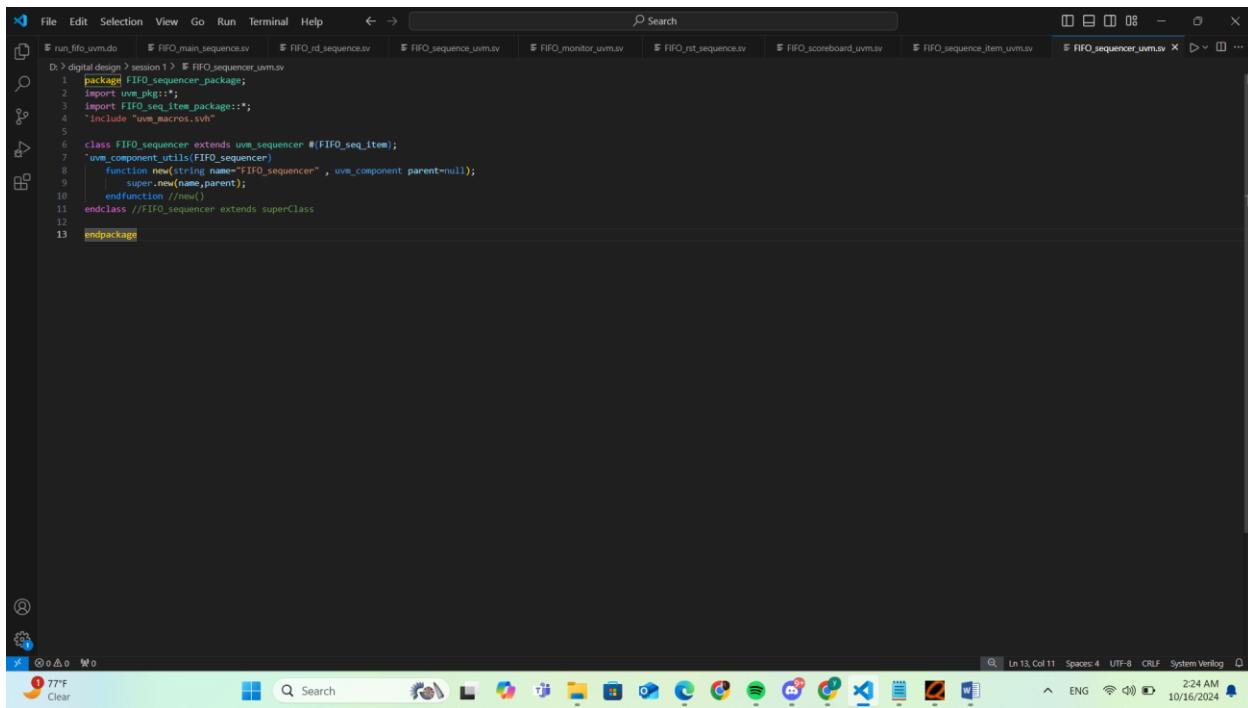
## FIFO\_configuration

## FIFO\_driver

## FIFO\_sequence\_item

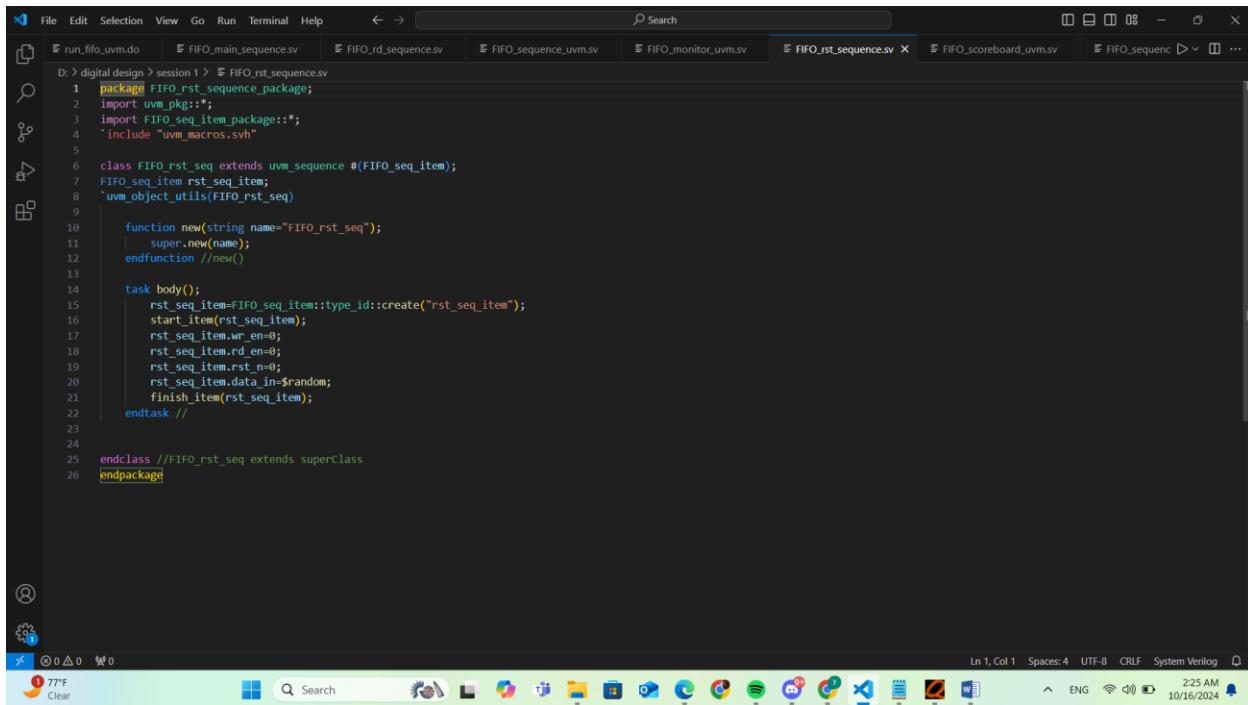
```
D:\> digital design > session1 > FIFO_sequence_item_uvm.sv
1 package FIFO_seq_item_package;
2 import uvm_pkg::*;
3 `include "uvm_macros.svh"
4 class FIFO_seq_item extends uvm_sequence_item;
5   `uvm_object_utils(FIFO_seq_item)
6   parameter FIFO_WIDTH = 8;
7   parameter FIFO_DEPTH = 8;
8   rand logic [FIFO_WIDTH-1:0] data_in;
9   rand logic rst_n, wr_en, rd_en;
10  logic [FIFO_WIDTH-1:0] data_out;
11  logic wr_ack, overflow;
12  logic full, empty, almostfull, almostempty, underflow;
13  integer RD_EN_ON_DIST,WR_EN_ON_DIST;
14
15  function new(string name="FIFO_seq_item");
16    super.new(name);
17  endfunction //new()
18
19  function string convertToString();
20    return $sformatf("rst_n=0x%08hb , wr_en=0x%08hb , rd_en=0x%08hb , data_in=0x%08hb ,
21    wr_ack=0x%08hb , overflow=0x%08hb , full=0x%08hb , empty=0x%08hb , almostfull=0x%08hb , almostempty=0x%08hb , underflow=0x%08hb" ,super.convertToString(), rst_n , wr_en , rd_en , data_in ,
22    wr_ack , overflow , full , empty , almostfull , almostempty , underflow);
23
24  endfunction
25
26  constraint cl{rst_n dist{1:-90, 0:-10}};
27  constraint c2{wr_en dist{1:-WR_EN_ON_DIST , 0:-(100-WR_EN_ON_DIST)};};
28  constraint c3{rd_en dist{1:-RD_EN_ON_DIST , 0:-(100-RD_EN_ON_DIST)};};
29 endclass //classname extends superClass
30
31 endpackage
```

## FIFO\_sequencer



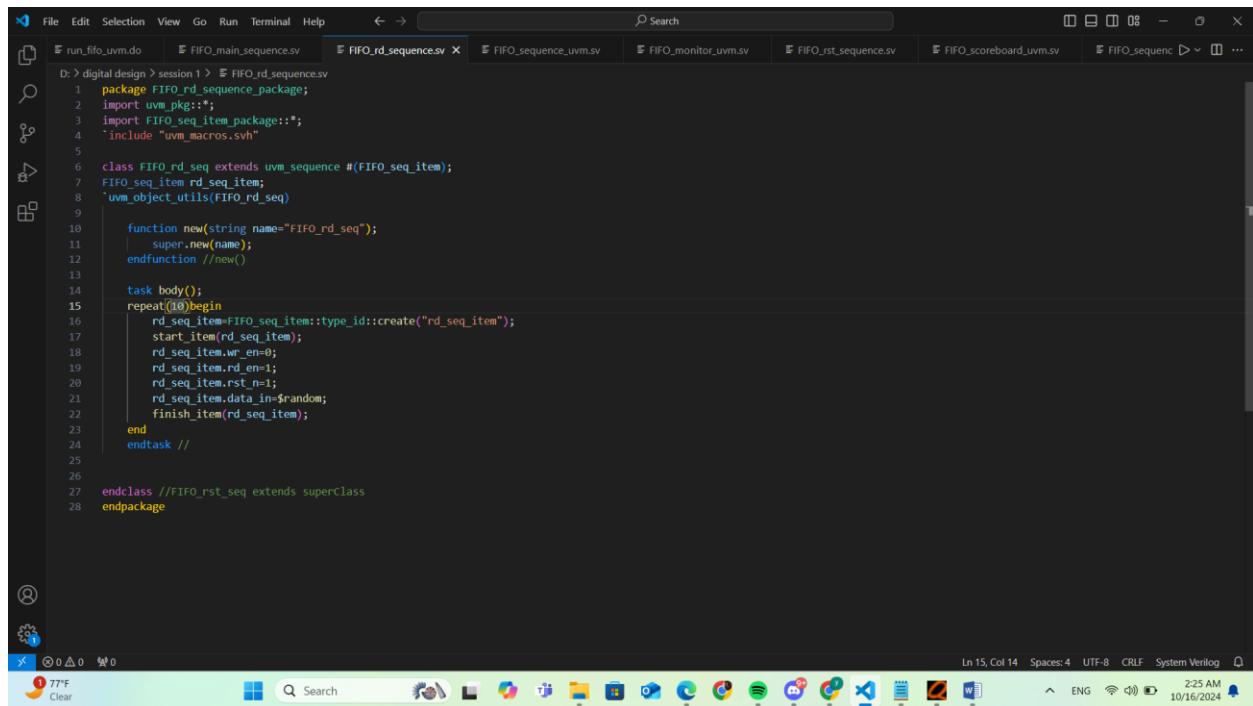
```
File Edit Selection View Go Run Terminal Help Search
D:\digital design > session 1 > FIFO_sequencer_uvm.sv FIFO_main_sequence.sv FIFO_rd_sequence.sv FIFO_sequence_uvm.sv FIFO_monitor_uvm.sv FIFO_RST_sequence.sv FIFO_scoreboard_uvm.sv FIFO_sequence_item_uvm.sv FIFO_sequencer_uvm.sv
1 package FIFO_sequencer_package;
2 import uvm_pkg::*;
3 import FIFO_seq_item_package::*;
4 `include "uvm_macros.svh"
5
6 class FIFO_sequencer extends uvm_sequencer #(FIFO_seq_item);
7 `uvm_component_utils(FIFO_sequencer)
8   function new(string name="FIFO_sequencer", uvm_component parent=null);
9     super.new(name,parent);
10    endfunction //new()
11  endclass //FIFO_sequencer extends superClass
12
13 endpackage
```

## FIFO\_reset\_sequence



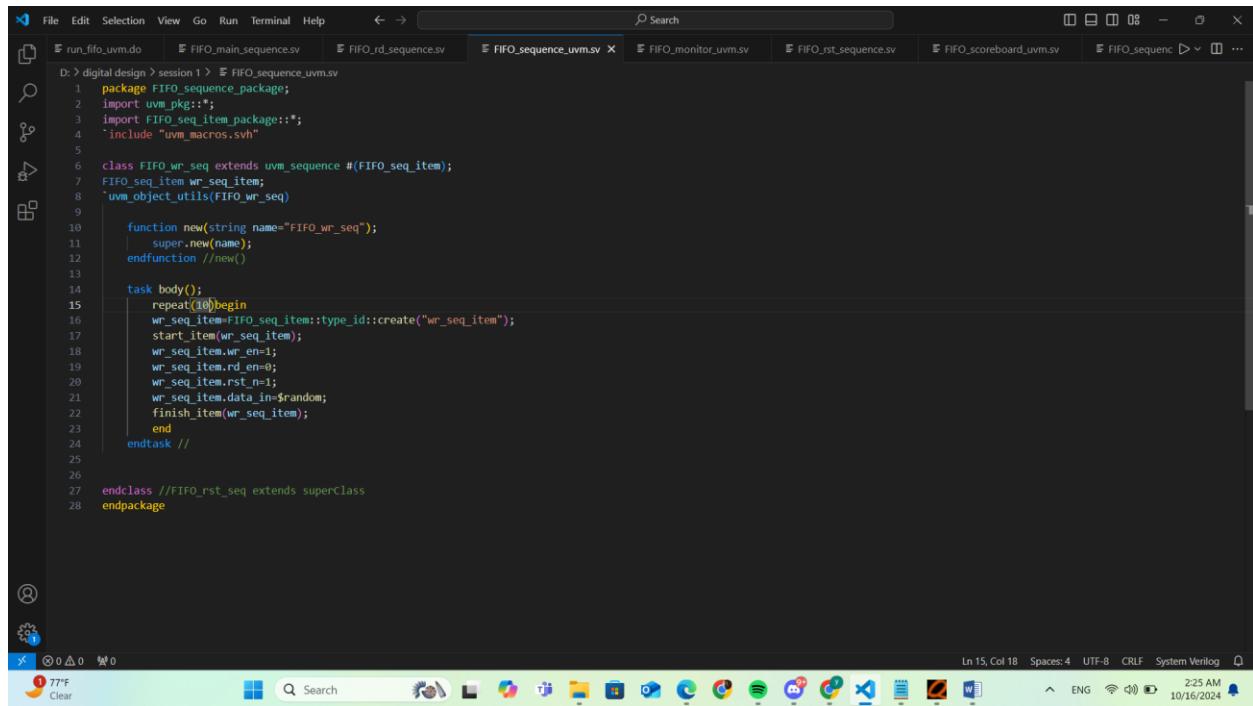
```
File Edit Selection View Go Run Terminal Help Search
D:\digital design > session 1 > FIFO_RST_sequence.sv FIFO_main_sequence.sv FIFO_rd_sequence.sv FIFO_sequence_uvm.sv FIFO_monitor_uvm.sv FIFO_RST_sequence.sv FIFO_scoreboard_uvm.sv FIFO_sequence_item_uvm.sv FIFO_sequencer_uvm.sv
1 package FIFO_RST_sequence_package;
2 import uvm_pkg::*;
3 import FIFO_seq_item_package::*;
4 `include "uvm_macros.svh"
5
6 class FIFO_RST_seq extends uvm_sequence #(FIFO_seq_item);
7 `uvm_object_utils(FIFO_RST_seq)
8
9   function new(string name="FIFO_RST_seq");
10    super.new(name);
11   endfunction //new()
12
13   task body();
14     rst_seq_item=FIFO_seq_item::type_id::create("rst_seq_item");
15     start_item(rst_seq_item);
16     rst_seq_item.wr_en=0;
17     rst_seq_item.rd_en=0;
18     rst_seq_item.rst_n=0;
19     rst_seq_item.data_in=$random;
20     finish_item(rst_seq_item);
21   endtask //
22
23
24
25 endclass //FIFO_RST_seq extends superClass
26 endpackage
```

## FIFO\_read\_sequence



```
D:\> digital design > session 1 > FIFO_rd_sequence.sv
1 package FIFO_rd_sequence_package;
2 import uvm_pkg::*;
3 import FIFO_seq_item_package::*;
4 `include "uvm_macros.svh"
5
6 class FIFO_rd_seq extends uvm_sequence #(FIFO_seq_item);
7 FIFO_seq_item rd_seq_item;
8 `uvm_object_utils(FIFO_rd_seq)
9
10 function new(string name="FIFO_rd_seq");
11     super.new(name);
12 endfunction //new()
13
14 task body();
15     repeat(10)begin
16         rd_seq_item= FIFO_seq_item::type_id::create("rd_seq_item");
17         start_item(rd_seq_item);
18         rd_seq_item.wr_en=0;
19         rd_seq_item.rd_en=1;
20         rd_seq_item.rst=1;
21         rd_seq_item.data_in=$random;
22         finish_item(rd_seq_item);
23     end
24     endtask //
25
26
27 endclass //FIFO_rd_seq extends superclass
28 endpackage
```

## FIFO\_write\_sequence



```
D:\> digital design > session 1 > FIFO_sequence_uvm.sv
1 package FIFO_sequence_package;
2 import uvm_pkg::*;
3 import FIFO_seq_item_package::*;
4 `include "uvm_macros.svh"
5
6 class FIFO_wr_seq extends uvm_sequence #(FIFO_seq_item);
7 FIFO_seq_item wr_seq_item;
8 `uvm_object_utils(FIFO_wr_seq)
9
10 function new(string name="FIFO_wr_seq");
11     super.new(name);
12 endfunction //new()
13
14 task body();
15     repeat(10)begin
16         wr_seq_item= FIFO_seq_item::type_id::create("wr_seq_item");
17         start_item(wr_seq_item);
18         wr_seq_item.wr_en=1;
19         wr_seq_item.rd_en=0;
20         wr_seq_item.rst=1;
21         wr_seq_item.data_in=$random;
22         finish_item(wr_seq_item);
23     end
24     endtask //
25
26
27 endclass //FIFO_wr_seq extends superclass
28 endpackage
```

## FIFO sequence with both read and write

The screenshot shows a Verilog code editor with the following code:

```
D:\> digital design > session 1 > FIFO_main_sequence.sv
1 package FIFO_main_sequence_package;
2 import uvm_pkg::*;
3 import FIFO_seq_item_package::*;
4 `include "uvm_macros.svh"
5
6 class FIFO_main_seq extends uvm_sequence #(FIFO_seq_item);
7 FIFO_seq_item main_seq_item;
8 `uvm_object_utils(FIFO_main_seq)
9
10 function new(string name="FIFO_main_seq");
11     super.new(name);
12 endfunction //new()
13
14 task body();
15 repeat(10)begin
16     main_seq_item=FIFO_seq_item::type_id::create("main_seq_item");
17     start_item(main_seq_item);
18     main_seq_item.wr_en=1;
19     main_seq_item.rd_en=1;
20     main_seq_item.rst_n=1;
21     main_seq_item.data_in=Random;
22     finish_item(main_seq_item);
23 end
24 endtask //
25
26
27 endclass //FIFO_rst_seq extends superClass
28 endpackage
```

The code defines a class `FIFO_main_seq` that extends `uvm_sequence`. It includes a sequence item `main_seq_item` and a task `body()` that repeats 10 times, creating a sequence item, starting it, setting `wr_en` to 1, setting `rd_en` to 1, setting `rst_n` to 1, setting `data_in` to a random value, and then finishing the item.

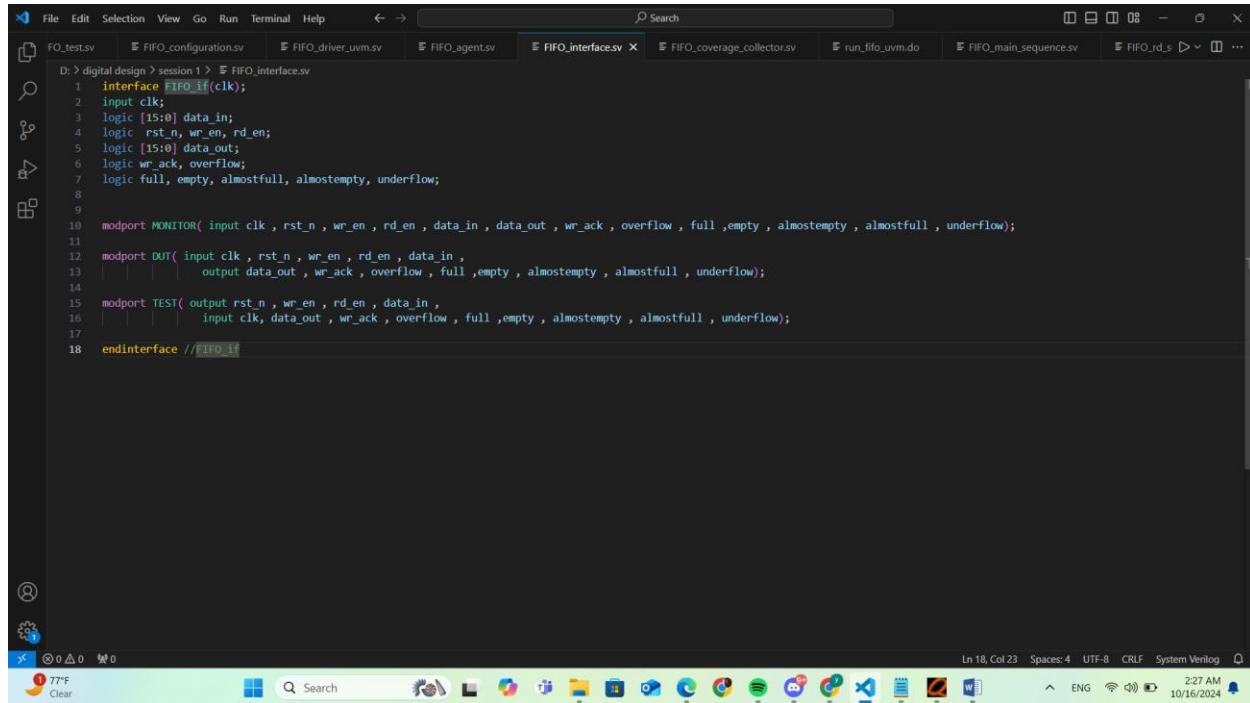
## FIFO\_monitor

The screenshot shows a Verilog code editor with the following code:

```
D:\> digital design > session 1 > FIFO_monitor_uvm.sv
1 package FIFO_monitor_package;
2 import uvm_pkg::*;
3 import FIFO_seq_item_package::*;
4 `include "uvm_macros.svh"
5
6 class FIFO_monitor extends monitor;
7     `uvm_component_utils(FIFO_monitor)
8     virtual FIFO_if FIFO_vif;
9     FIFO_seq_item seq_item;
10    FIFO_analysis_port #(FIFO_seq_item) mon_ap;
11
12    function new(string name="FIFO_monitor" , uvm_component parent=null);
13        super.new(name,parent);
14    endfunction //new()
15
16    function void build_phase(uvm_phase phase);
17        super.build_phase(phase);
18        mon_ap=new("mon_ap",this);
19    endfunction
20
21    task run_phase(uvm_phase phase);
22        super.run_phase(phase);
23        forever begin
24            seq_item=FIFO_seq_item::type_id::create("seq_item");
25            @(nedge FIFO_vif.clk);
26            seq_item.n_pFIFO_vif_rd_en=0;
27            seq_item.n_pFIFO_vif_wr_en=0;
28            seq_item.rd_en=FIFO_vif.rd_en;
29            seq_item.data_in=FIFO_vif.data_in;
30            seq_item.wr_ack=FIFO_vif.wr_ack;
31            seq_item.almostempty=FIFO_vif.almostempty;
32            seq_item.empty=FIFO_vif.empty;
33            seq_item.almostfull=FIFO_vif.almostfull;
34            seq_item.full=FIFO_vif.full;
35            seq_item.overflow=FIFO_vif.overflow;
36            seq_item.underflow=FIFO_vif.underflow;
37            mon_ap.write(seq_item);
38            `uvm_info("run_phase" , seq_item.convert2String(), UVM_HIGH)
39        end
40    endtask //
41 endclass //FIFO_monitor extends superClass
42
43 endpackage
```

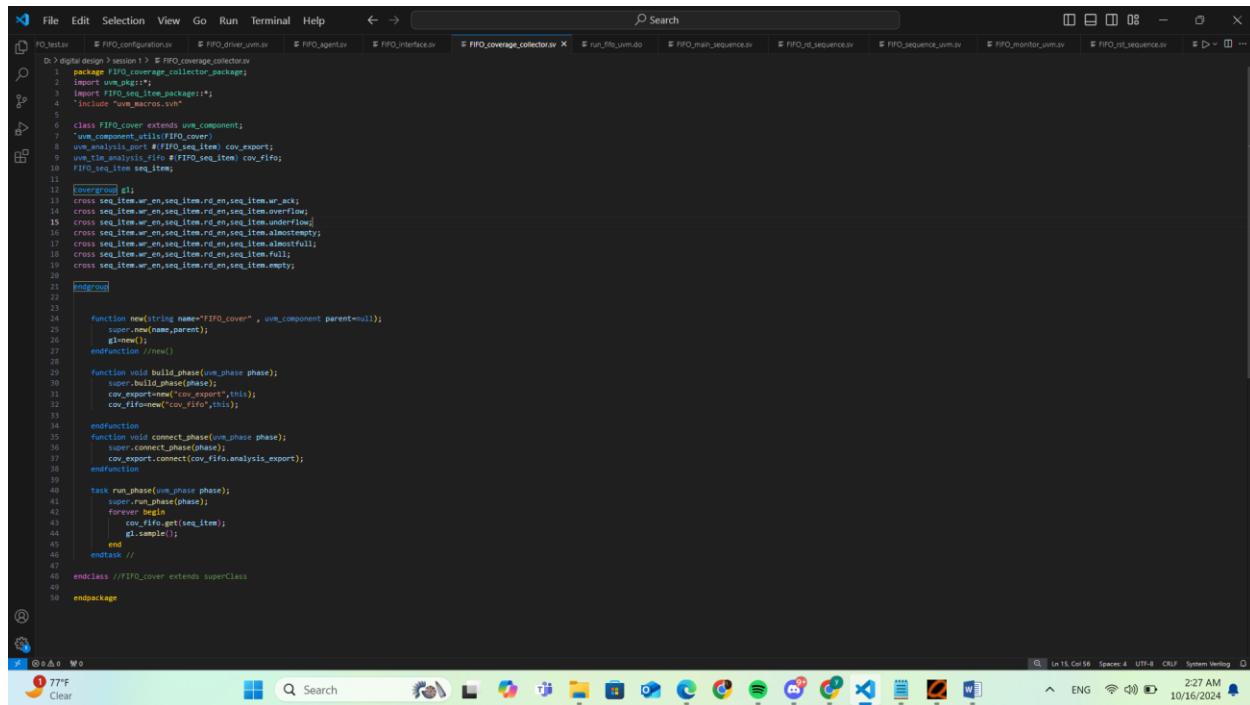
The code defines a class `FIFO_monitor` that extends `monitor`. It includes a virtual interface `FIFO_if`, a sequence item `seq_item`, and an analysis port `mon_ap`. The `build_phase` method creates the analysis port. The `run_phase` method repeatedly creates a sequence item, waits for a rising edge on the `FIFO_vif.clk`, sets `n_pFIFO_vif_rd_en` to 0, sets `n_pFIFO_vif_wr_en` to 0, sets `rd_en` to the value of `FIFO_vif.rd_en`, sets `data_in` to the value of `FIFO_vif.data_in`, sets `wr_ack` to the value of `FIFO_vif.wr_ack`, checks if `almostempty` is true, checks if `empty` is true, checks if `almostfull` is true, checks if `full` is true, checks if `overflow` is true, checks if `underflow` is true, writes the sequence item to the analysis port, and prints an informational message using `uvm_info`.

## FIFO\_interface



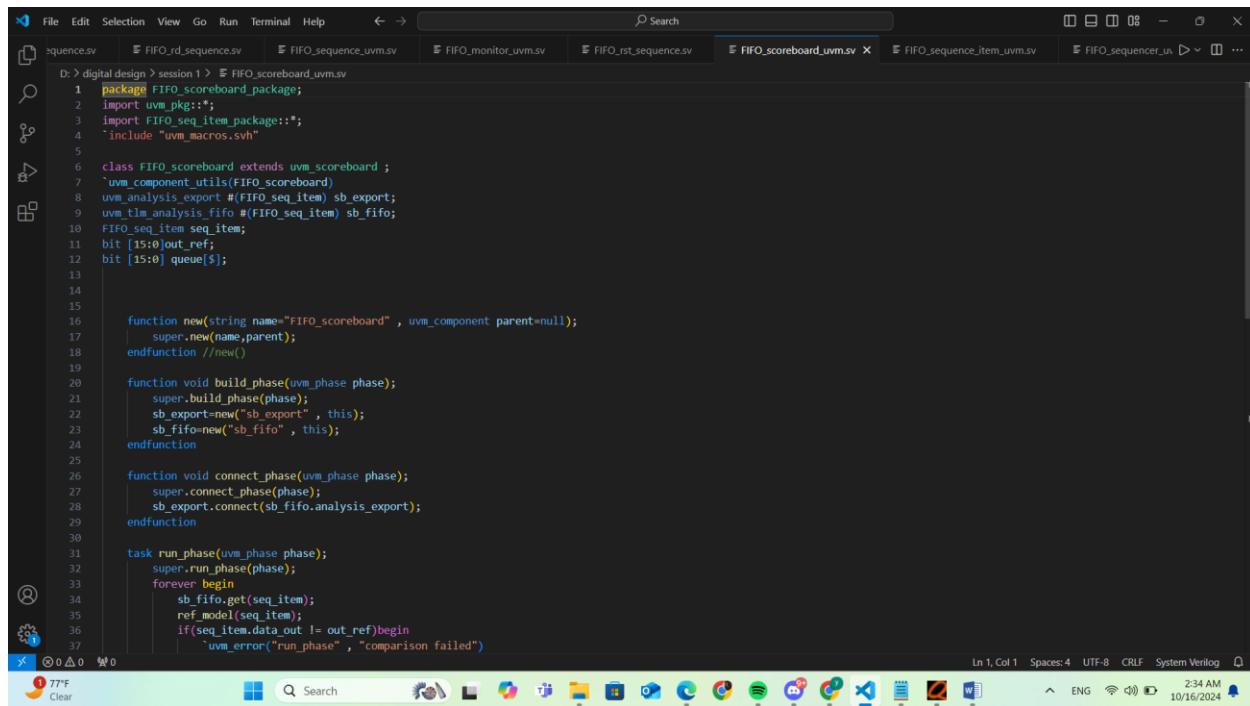
```
D:\> digital design > session 1 > FIFO_interface.sv
File Edit Selection View Go Run Terminal Help ⏪ ⏩ Search
FO_test.sv FIFO_configuration.sv FIFO_driver_uvm.sv FIFO_agent.sv FIFO_interface.sv FIFO_coverage_collector.sv run_fifo_uvm.do FIFO_main_sequence.sv FIFO_rd.s ...
1 interface FIFO_if(clk);
2   input clk;
3   logic [15:0] data_in;
4   logic rst_n, wr_en, rd_en;
5   logic [15:0] data_out;
6   logic wr_ack, overflow;
7   logic full, empty, almostfull, almostempty, underflow;
8
9   modport MONITOR( input clk, rst_n, wr_en, rd_en, data_in, data_out , wr_ack , overflow , full ,empty , almostempty , almostfull , underflow);
10
11  modport OUT( input clk, rst_n, wr_en, rd_en, data_in ,
12 | | | | | output data_out, wr_ack , overflow , full ,empty , almostempty , almostfull , underflow);
13
14  modport TEST( output rst_n , wr_en , rd_en , data_in ,
15 | | | | | input clk, data_out , wr_ack , overflow , full ,empty , almostempty , almostfull , underflow);
16
17
18 endinterface //FIFO_if
```

## FIFO\_coverage\_collector

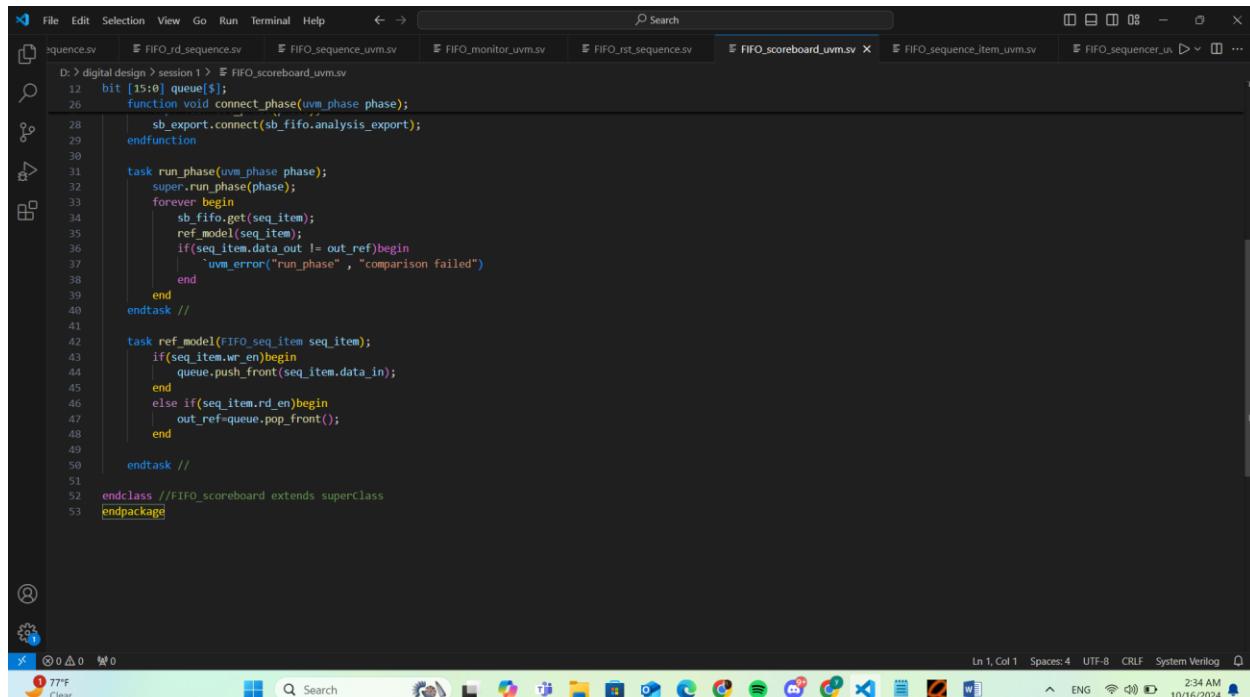


```
D:\> digital design > session 1 > FIFO_coverage_collector.sv
File Edit Selection View Go Run Terminal Help ⏪ ⏩ Search
FO_test.sv FIFO_configuration.sv FIFO_driver_uvm.sv FIFO_agent.sv FIFO_interface.sv FIFO_coverage_collector.sv run_fifo_uvm.do FIFO_main_sequence.sv FIFO_rd_sequence.sv FIFO_sequence_uvm.sv FIFO_monitor_uvm.sv FIFO_rd_sequence.sv ...
1 package FIFO_coverage_collector_package;
2 import uvm_pkg::*;
3 import FIFO_seq_item_package::*;
4 `include "uvm_macros.svh"
5
6 class FIFO_cover extends uvm_component;
7   `uvm_component_utils(FIFO_cover)
8   uvm_sequencer #(FIFO_seq_item) cov_export;
9   `uvm_tlm_analysis_fifo #(FIFO_seq_item, cov_fifo) cov_fifo;
10  FIFO_seq_item seq_item;
11
12  `uvm_group();
13  cross seq_item.wr_en,seq_item.rid,en,seq_item.wr_ack;
14  cross seq_item.wr_en,seq_item.rd_en,seq_item.overflow;
15  cross seq_item.wr_en,seq_item.rd_en,seq_item.underflow;
16  cross seq_item.wr_en,seq_item.en,seq_item.almostempty;
17  cross seq_item.wr_en,seq_item.en,seq_item.almostfull;
18  cross seq_item.wr_en,seq_item.rd_en,seq_item.full;
19  cross seq_item.wr_en,seq_item.rd_en,seq_item.empty;
20
21  `uvm_group();
22
23
24  function new(string name="FIFO_cover", uvm_component parent=null);
25    super.new(name,parent);
26    gnew();
27  endfunction //new()
28
29  function void add_phase(uvm_phase phase);
30    phase.bind(this);
31    cov_export=new("cov_export",this);
32    cov_fifo=new("cov_fifo",this);
33  endfunction
34
35  function void connect_phase(uvm_phase phase);
36    super.connect_phase(phase);
37    cov_export.connect(cov_fifo.analysis_export);
38  endfunction
39
40  task run_phase(uvm_phase phase);
41    super.run_phase(phase);
42    forever begin
43      cov_fifo.get(seq_item);
44      gl.sample();
45    end
46  endtask //
47
48 endclass //FIFO_cover extends superClass
49
50 endpackage
```

## FIFO\_scoreboard

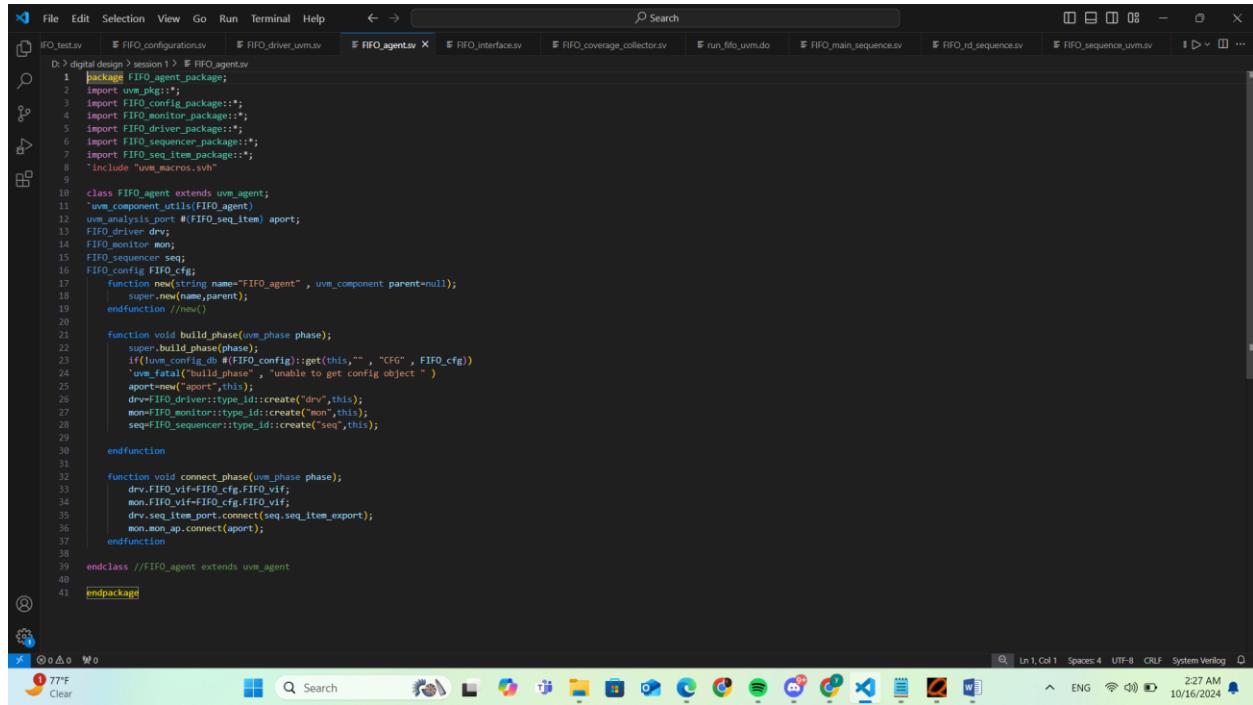


```
D:\> digital design > session 1 > FIFO_scoreboard.uvm.sv
sequence.sv  FIFO_rd_sequence.sv  FIFO_sequence_uvm.sv  FIFO_monitor_uvm.sv  FIFO_rst_sequences.sv  FIFO_scoreboard_uvm.sv  FIFO_sequence_item_uvm.sv  FIFO_sequencer_uvm.sv
1 package FIFO_scoreboard_package;
2 import uvm_pkg::*;
3 import FIFO_seq_item_package::*;
4 `include "uvm_macros.svh"
5
6 class FIFO_scoreboard extends uvm_scoreboard ;
7 `uvm_component_utils(FIFO_scoreboard)
8 uvm_analysis_export #(FIFO_seq_item) sb_export;
9 uvm_tlm_analysis_fifo #(FIFO_seq_item) sb_fifo;
10 FIFO_seq_item seq_item;
11 bit [15:0]out_ref;
12 bit [15:0] queue[$];
13
14
15
16 function new(string name="FIFO_scoreboard", uvm_component parent=null);
17 super.new(name,parent);
18 endfunction //new()
19
20 function void build_phase(uvm_phase phase);
21 super.build_phase(phase);
22 sb_export=new("sb_export", this);
23 sb_fifo=new("sb_fifo", this);
24 endfunction
25
26 function void connect_phase(uvm_phase phase);
27 super.connect_phase(phase);
28 sb_export.connect(sb_fifo.analysis_export);
29 endfunction
30
31 task run_phase(uvm_phase phase);
32 super.run_phase(phase);
33 forever begin
34     sb_fifo.get(seq_item);
35     ref_model(seq_item);
36     if(seq_item.data_out != out_ref)begin
37         `uvm_error("run_phase", "comparison failed")
38     end
39 end
40 endtask //
41
42 task ref_model(FIFO_seq_item seq_item);
43     if(seq_item.wr_en)begin
44         queue.push_front(seq_item.data_in);
45     end
46     else if(seq_item.rd_en)begin
47         out_ref=queue.pop_front();
48     end
49 endtask //
50
51 endclass //FIFO_scoreboard extends superClass
52 endpackage
```



```
D:\> digital design > session 1 > FIFO_scoreboard.uvm.sv
sequence.sv  FIFO_rd_sequence.sv  FIFO_sequence_uvm.sv  FIFO_monitor_uvm.sv  FIFO_rst_sequences.sv  FIFO_scoreboard_uvm.sv  FIFO_sequence_item_uvm.sv  FIFO_sequencer_uvm.sv
12 bit [15:0] queue[$];
26 function void connect_phase(uvm_phase phase);
27     sb_export.connect(sb_fifo.analysis_export);
28 endfunction
29
30
31 task run_phase(uvm_phase phase);
32     super.run_phase(phase);
33     forever begin
34         sb_fifo.get(seq_item);
35         ref_model(seq_item);
36         if(seq_item.data_out != out_ref)begin
37             `uvm_error("run_phase", "comparison failed")
38         end
39     end
40 endtask //
41
42 task ref_model(FIFO_seq_item seq_item);
43     if(seq_item.wr_en)begin
44         queue.push_front(seq_item.data_in);
45     end
46     else if(seq_item.rd_en)begin
47         out_ref=queue.pop_front();
48     end
49 endtask //
50
51 endclass //FIFO_scoreboard extends superClass
52 endpackage
```

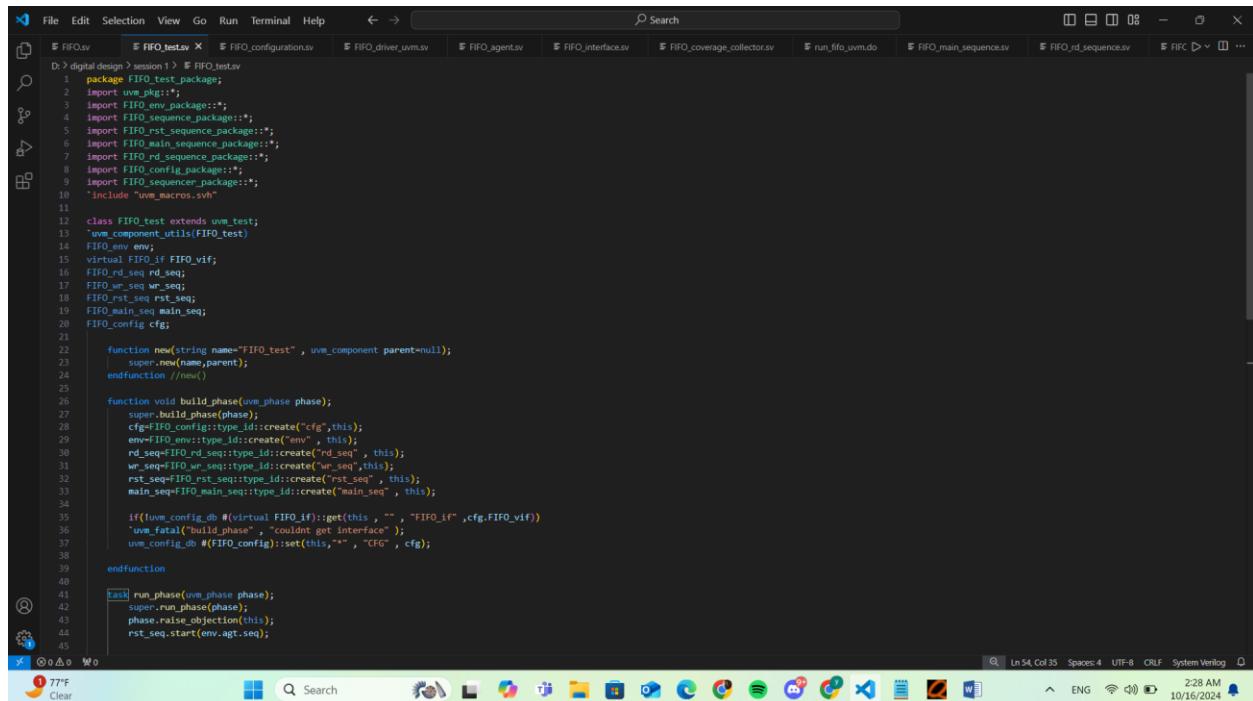
## FIFO\_agent



The screenshot shows a Verilog IDE interface with multiple tabs open. The current tab is 'FIFO\_agent.sv'. The code implements a FIFO agent with a build\_phase and connect\_phase. It uses FIFO config, driver, monitor, sequencer, and seq\_item packages. It also includes a uvm\_macros.svh include statement.

```
D:\> digital design > session 1 > FIFO_agent.sv
1 package FIFO_agent_package;
2 import uvm_pkg::*;
3 import FIFO_config_package::*;
4 import FIFO_monitor_package::*;
5 import FIFO_driver_package::*;
6 import FIFO_sequencer_package::*;
7 import FIFO_seq_item_package::*;
8 `include "uvm_macros.svh"
9
10 class FIFO_agent extends uvm_agent;
11   `uvm_component_utils(FIFO_agent)
12   uvm_analysis_port #(FIFO_seq_item) aport;
13   FIFO_driver driver;
14   FIFO_monitor mon;
15   FIFO_sequencer seq;
16   FIFO_config FIFO_cfg;
17   function new(string name="FIFO_agent", uvm_component parent=null);
18     super.new(name,parent);
19   endfunction //new()
20
21   function void build_phase(uvm_phase phase);
22     super.build_phase(phase);
23     if(!uvm_config_db #(FIFO_config)::get(this,"", "CFG", FIFO_cfg))
24       `uvm_fatal("FIFO_agent","unable to get config object")
25     aport=aport.type_id::create("aport",this);
26     driver=driver.type_id::create("drv",this);
27     mon=mon.type_id::create("mon",this);
28     seq=seq.type_id::create("seq",this);
29
30   endfunction
31
32   function void connect_phase(uvm_phase phase);
33     driver.vif=FIFO_cfg.FIFO_vif;
34     mon.FIFO_vif=FIFO_cfg.FIFO_vif;
35     driver.seq_item_port.connect(seq.seq_item_export);
36     mon.mon_ap.connect(aport);
37   endfunction
38
39 endclass //FIFO_agent extends uvm_agent
40
41 endpackage
```

## FIFO\_Test



The screenshot shows a Verilog IDE interface with multiple tabs open. The current tab is 'FIFO.sv'. The code implements a FIFO test with a build\_phase and run\_phase. It uses FIFO config, env, driver, monitor, sequencer, and seq\_item packages. It also includes a uvm\_macros.svh include statement.

```
D:\> digital design > session 1 > FIFO.sv
1 package FIFO_test_package;
2 import uvm_pkg::*;
3 import FIFO_env_package::*;
4 import FIFO_driver_package::*;
5 import FIFO_monitor_package::*;
6 import FIFO_main_sequence_package::*;
7 import FIFO_rd_sequence_package::*;
8 import FIFO_config_package::*;
9 import FIFO_sequencer_package::*;
10 `include "uvm_macros.svh"
11
12 class FIFO_test extends uvm_test;
13   `uvm_component_utils(FIFO_test)
14   FIFO_env env;
15   FIFO_if FIFO_vif;
16   FIFO_rd_seq rd_seq;
17   FIFO_wr_seq wr_seq;
18   FIFO_rst_seq rst_seq;
19   FIFO_main_seq main_seq;
20   FIFO_config cfg;
21
22   function new(string name="FIFO_test", uvm_component parent=null);
23     super.new(name,parent);
24   endfunction //new()
25
26   function void build_phase(uvm_phase phase);
27     super.build_phase(phase);
28     cfg=FIFO_config.type_id::create("cfg",this);
29     env=FIFO_env.type_id::create("env",this);
30     rd_seq=FIFO_rd_Sequence.type_id::create("rd_seq",this);
31     wr_seq=FIFO_wr_Sequence.type_id::create("wr_seq",this);
32     rst_seq=FIFO_rst_Sequence.type_id::create("rst_seq",this);
33     main_seq=FIFO_main_Sequence.type_id::create("main_seq",this);
34
35     if(!uvm_config_db #(virtual FIFO_if)::get(this, "", "FIFO_if",cfg.FIFO_vif))
36       `uvm_fatal("FIFO_if","couldn't get interface");
37     uvm_config_db #(FIFO_config)::set(this,"", "CFG", cfg);
38
39   endfunction
40
41   task run_phase(uvm_phase phase);
42     super.run_phase(phase);
43     phase.raise_objection(this);
44     rst_seq.start(env.agt.seq);
45   endtask
46
47 endclass
```

```

File Edit Selection View Go Run Terminal Help ← → Search
D:> digital design > session 1 > FIFO_testav
FIFO.sv FIFO_testav FIFO_configuration.sv FIFO_driver_uvm.sv FIFO_agent.sv FIFO_interface.sv run_fifo_uvm.dv FIFO_main_sequence.sv FIFO_rd_sequence.sv FIFO.v
28   FIFO_config cfg;
29
30   function void build_phase(uvm_phase phase);
31     super.build_phase(phase);
32     cfg=FIFO_configuration::id::create("cfg",this);
33     env=FIFO_env::type_id::create("env",this);
34     rd_seq=FIFO_rd_seq::type_id::create("rd_seq",this);
35     wr_seq=FIFO_wr_seq::type_id::create("wr_seq",this);
36     rst_seq=FIFO_rst_seq::type_id::create("rst_seq",this);
37     main_seq=FIFO_main_seq::type_id::create("main_seq",this);
38
39     if(!uvm_config_db #(virtual FIFO_if)::get(this , "" , "FIFO_if" ,cfg.FIFO_vif))
40       `uvm_fatal("build_phase" , "couldn't get interface");
41     uvm_config_db #(FIFO_config)::set(this,"" , "CFG" , cfg);
42
43   endfunction
44
45   task run_phase(uvm_phase phase);
46     super.run_phase(phase);
47     phase.raise_objection(this);
48     rst_seq.start(env.agt.seq);
49
50     wr_seq.start(env.agt.seq);
51
52     rd_seq.start(env.agt.seq);
53
54     main_seq.start(env.agt.seq);
55
56     rd_seq.start(env.agt.seq);
57
58     phase.drop_objection(this);
59
60   endtask
61 endclass //FIFO_test extends uvm_test
62
endpackage

```

Ln 54, Col 35 Spaces: 4 UTF-8 CRLF System Verilog 2:28 AM 10/16/2024

## FIFO\_top

```

File Edit Selection View Go Run ... ← → Search
D:> digital design > session 1 > FIFO_top_uvm.sv
FIFO.sv FIFO_test.sv FIFO_top_uvm.sv FIFO_configuration.sv FIFO_driver_uvm.sv FIFO_agent.sv FIFO_interface.sv FIFO_co
1   import uvm_pkg::*;
2   import FIFO_config_package::*;
3   import FIFO_test_package::*;
4   module FIFO_top_uvm (
5   );
6     bit clk;
7
8   initial begin
9     clk=0;
10    forever begin
11      #1 clk = ~clk;
12    end
13  end
14
15  FIFO_if FIFO_if(clk);
16
17  FIFO f1(FIFO_if);
18
19  initial begin
20    uvm_config_db #(virtual FIFO_if)::set(null , "uvm_test_top" , "FIFO_if", FIFO_if );
21    run_test("FIFO_test");
22  end
23
24 endmodule

```

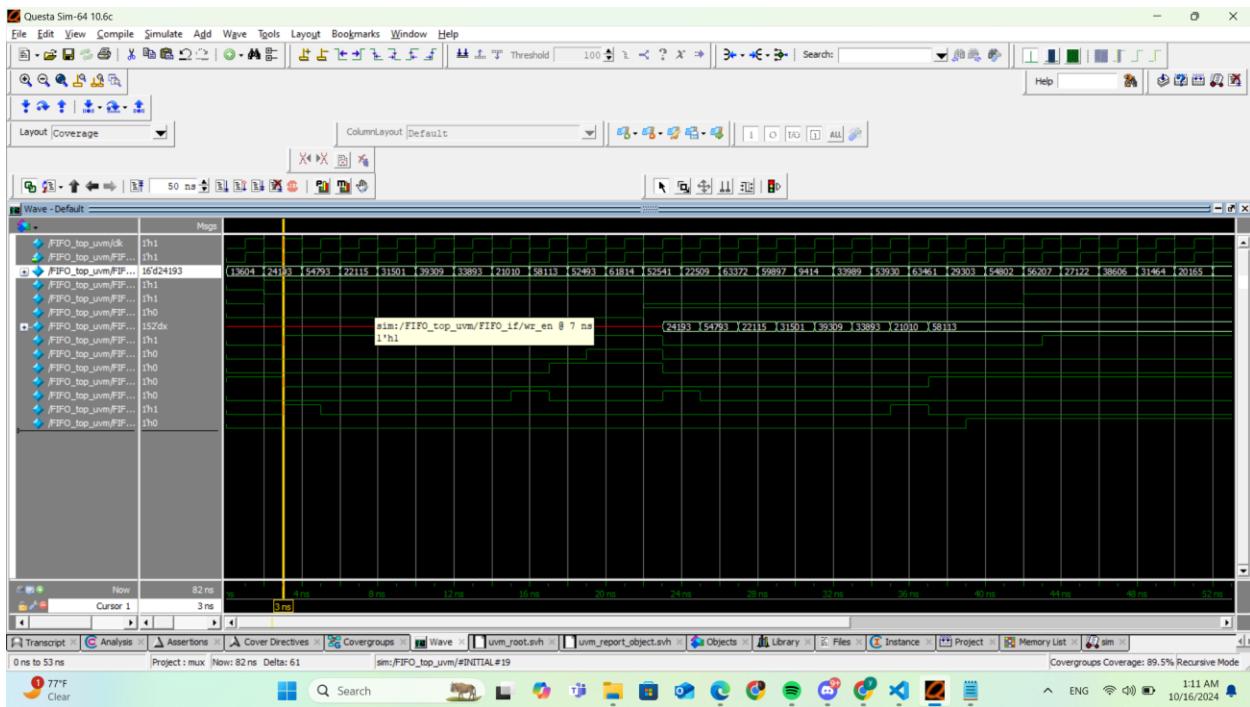
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF System Verilog 2:28 AM 10/16/2024

## Do\_file

```
D:\> digital design > session 1 > run_fifo_uvm.do
1 vlib work
2 vlog -f src_files.list +cover -covercells
3 vsim -voptargs+=acc work.FIFO_top_uvm -cover
4 add wave /FIFO_top_uvm/clk
5 add wave /FIFO_top_uvm/FIFO_if/*
6 coverage save -du FIFO FIFO_top_uvm.ucdb -onexit
7 run -all
```

The screenshot shows a terminal window with a dark background. It displays a command-line script named 'run\_fifo\_uvm.do'. The script contains commands to set up a UVM environment, log files, simulate the FIFO\_top\_uvm module, and save coverage data. The terminal window has a standard Windows-style title bar and status bar at the bottom.

## Simulation



## Assertions

Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATP	Assertion Expression	Included
Avm_pkg::uvm_r...	Immediate	SVA	on	0	0	-	-	-	-	off	assert( \$cast(tec_o))		
Avm_pkg::uvm_r...	Immediate	SVA	on	0	0	-	-	-	-	off	assert( \$cast(tec_o))		
FIFO_top_uvm/f...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert( @posedge ff.clk) (@ff.vr_...	X	
FIFO_top_uvm/f...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert( @posedge ff.clk) (@ff.r_d_...	✓	
FIFO_top_uvm/f...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert( @posedge ff.clk) disable(...)	✓	
FIFO_top_uvm/f...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert( @posedge ff.clk) disable(...)	✓	
FIFO_top_uvm/f...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert( @posedge ff.clk) disable(...)	✓	
FIFO_top_uvm/f...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert( @posedge ff.clk) count=...	✓	
FIFO_top_uvm/f...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert( @posedge ff.clk) count=...	✓	
FIFO_top_uvm/f...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert( @posedge ff.clk) count=...	✓	
FIFO_top_uvm/f...	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0 off	assert( @posedge ff.clk) count=...	✓	

Transcript | Analysis | Assertions | Cover Directives | Covergroups | Wave | uvm\_root.svh | uvm\_report\_object.svh | Objects | Library | Files | Instance | Project | Memory List | sim | Project : mux | Now: 82 ns Delta: 61 | sim:FIFO\_top\_uvm/#INITIAL#19 | Assertions Filter: NoFilter | Covergroups Coverage: 89.5% Recursive Mode | 77°F Clear | Search | 11:11 AM 10/16/2024

## Transcript

```

# (C) 2006-2013 Synopsys, Inc.
# (C) 2011-2013 Cypress Semiconductor Corp.

***** IMPORTANT RELEASE NOTES *****

You are using a version of the UVM library that has been compiled
with `UVM_NO_DEPRECATED` undefined.
See http://www.eda.org/svdb/view.php?id=3313 for more details.

You are using a version of the UVM library that has been compiled
with `UVM_OBJECT_HOST_HAVE_CONSTRUCTOR` undefined.
See http://www.eda.org/svdb/view.php?id=3770 for more details.

(Specify +UVM_NO_RELNOTES to turn off this notice)

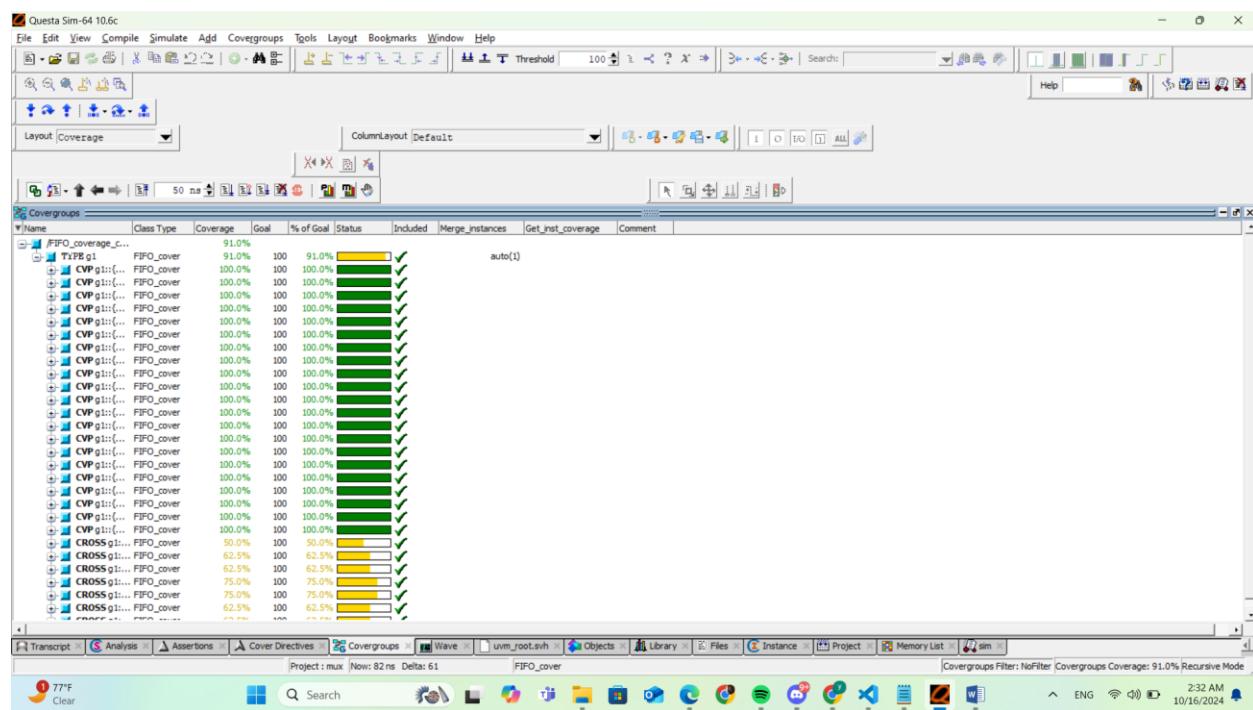
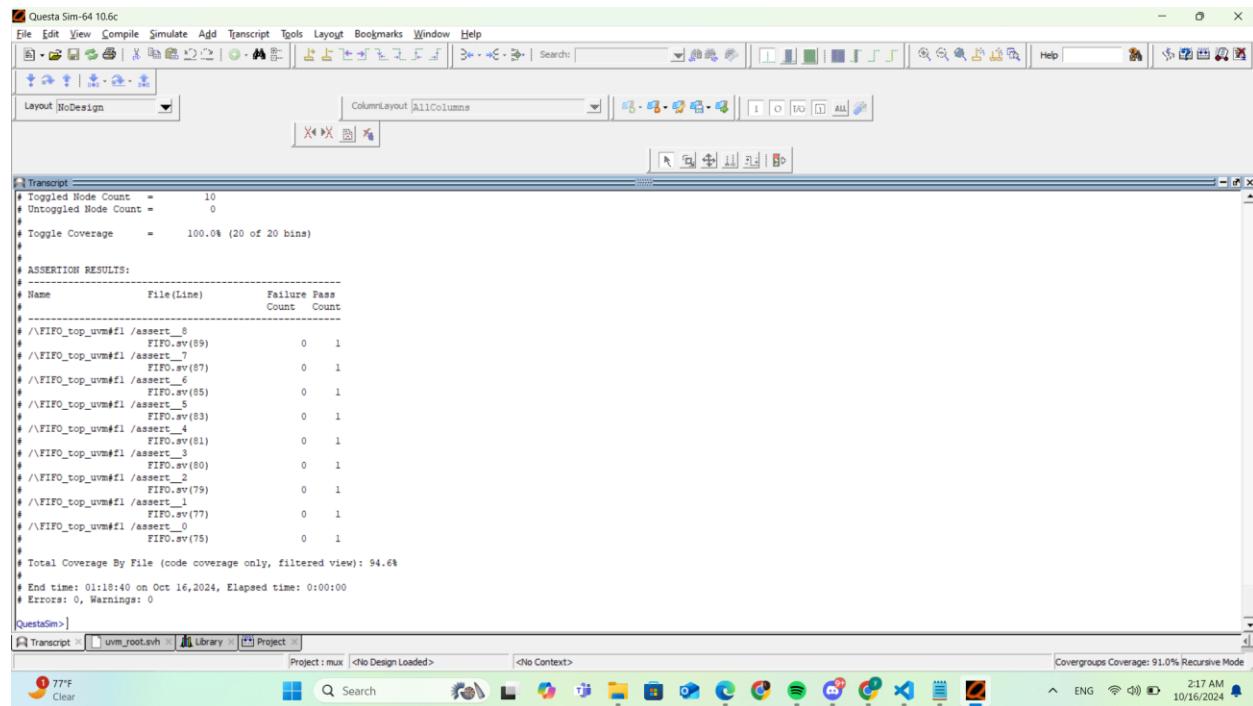
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(215) # 0: reporter [Questa UVM] QUESTA_UVM-1.2.3
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(217) # 0: reporter [Questa UVM]  questauvm::init(+struct)
# UVM_INFO # 0: reporter [RNTST] Running test FIFO_test...
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_object.svh(126) # 82: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase

--- UVM Report Summary ---
** Report counts by severity
# UVM_INFO : 4
# UVM_WARNING : 0
# UVM_ERROR : 0
# UVM_FATAL : 0
** Report counts by id
# [Questa UVM] 2
# [RNTST] 1
# [TEST_DONE] 1
** Note: $finish : C:/questasim64_10.6c/win64/..verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
# Time: 82 ns Iteration: 61 Instance: /FIFO_top_uvm

```

Transcript | Analysis | Assertions | Cover Directives | Covergroups | Wave | uvm\_root.svh | uvm\_report\_object.svh | Objects | Library | Files | Instance | Project | Memory List | sim | Project : mux | Now: 82 ns Delta: 61 | sim:FIFO\_top\_uvm/#INITIAL#19 | Covergroups Coverage: 89.5% Recursive Mode | 77°F Clear | Search | 11:12 AM 10/16/2024

## coverage



## Verification Plan

Feature	assertion
Whenever the FIFO is full and there is a write operation the overflow flag becomes high	<pre>assert property(@(posedge fif.clk)(fif.wr_en &amp;&amp; fif.full  =&gt; fif.overflow));</pre>
Whenever the FIFO is empty and there is read operation the underflow flag becomes high	<pre>assert property(@(posedge fif.clk)(fif.rd_en &amp;&amp; fif.empty  =&gt;fif.underflow));</pre>
Whenever the FIFO almostfull flag is high and there is write operation the full flag becomes high	<pre>assert property (@(posedge fif.clk)  disable iff(!fif.rst_n) (fif.wr_en &amp;&amp; fif.almostfull &amp;&amp; !fif.rd_en)  =&gt; fif.full);</pre>
Whenever the almost empty flag is high and there is a read operation the empty flag becomes high	<pre>assert property (@(posedge fif.clk)  disable iff(!fif.rst_n) (fif.almostempty &amp;&amp; fif.rd_en &amp;&amp; !fif.wr_en)  =&gt; fif.empty);</pre>
Whenever there is a write operation and the FIFO is not full the wr_ack becomes high	<pre>assert property(@(posedge fif.clk)((fif.wr_en &amp;&amp; !fif.full)  =&gt; fif.wr_ack));</pre>
Whenever the count is maximum the FIFO is full	<pre>assert property(@(posedge fif.clk)(count==FIFO_DEPTH)  -&gt; fif.full);</pre>
Whenever the count is 0 the FIFO is empty	<pre>assert property(@(posedge fif.clk)(count==0)  -&gt; fif.empty);</pre>
Whenever the count is 1 the FIFO is almostempty	<pre>assert property(@(posedge fif.clk)(count==1)  -&gt; fif.almostempty);</pre>
Whenever the count is before the maximum the almostfull flag becomes high	<pre>assert property(@(posedge fif.clk)(count==FIFO_DEPTH-1)  -&gt; fif.almostfull);</pre>