SYNCHRONOUS FIFO
UVM PROJECT

Project Report

Submitted by

Ahmed Marzouk

Submitted to

Eng. Kareem Wassem

CONTENTS

1. Overview of the project 3

The primary functionality of the FIFO includes 3

FIFO Signals 4

2. Verification plan 5

3. UVM Structure 6

4- How it works? 8

5- Questasim Snippets waveform 9

Reset sequence 9

Write only sequence 9

Read sequence only 10

Write Read sequence 11

Cover groups 12

Cover directives 12

Assertions 13

Transcript 14

6- Code and functional coverage. 16

7- Bug report 17

8- Assertion table 18

9- Do file. 19

Src.txt 20

10- Project codes 21

Fifo_top 21

fifo_test 22

fifo_config_obj 23

Reset sequence 23

Main sequence 24

fifo_env 26

fifo_interface 27

fifo_scoreboard 27

fifo_coverage_collector 29

fifo_agent 31

fifo_sequencer 32

fifo_driver 32

fifo_monitor 34

fifo_seq_item 35

FIFO design 36

fifo_sva 38

1. Overview of the project

A synchronous FIFO (First-In, First-Out) is a type of data buffer used to transfer data between two systems or processes that operate on the same clock domain but may have different processing speeds. It operates based on a single clock signal for both the write and read operations, ensuring synchronous timing.

The primary functionality of the FIFO includes:

1- Clock Domain: Operates on a single clock signal, eliminating complexities like clock domain crossing.

2- Memory Structure: Uses an internal memory array to store data temporarily.

3- Pointers:

- **Write Pointer (wr_ptr):** Indicates the next memory location to write data.
- **Read Pointer (rd_ptr):** Indicates the next memory location to read data.

Key Features:

- **FIFO_WIDTH**: Defines the width of data input and output buses, as well as the memory word width (default: 16 bits).
- **FIFO_DEPTH**: Determines the depth of the FIFO, representing the number of data entries that can be stored (default: 8 entries).

FIFO Signals:

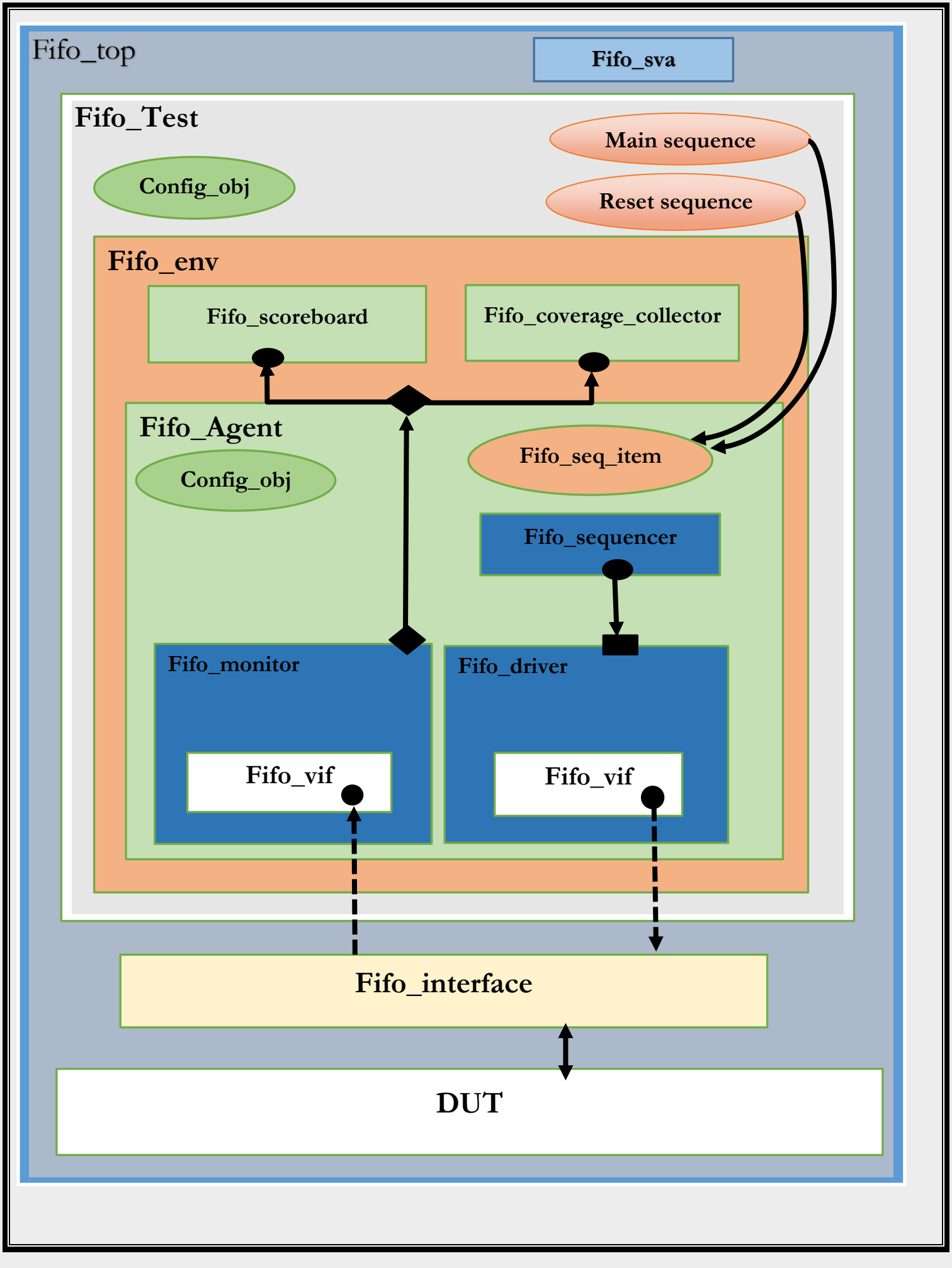
Signal	Direction	Description
data_in	Input	Write Data: The input data bus used when writing to the FIFO.
wr_en	Input	Write Enable: Enables data writing when the FIFO is not full.
rd_en	Input	Read Enable: Enables data reading when the FIFO is not empty.
clk	Input	Clock: The clock signal for synchronizing operations.
rst_n	Input	Reset: Active-low asynchronous reset signal.
data_out	Output	Read Data: The data output bus when reading from the FIFO.
full	Output	Full Flag: Indicates the FIFO is full and cannot accept more data.
almostfull	Output	Almost Full: Indicates one more write can be performed before full.
empty	Output	Empty Flag: Indicates the FIFO is empty.
almostempty	Output	Almost Empty: Indicates one more read can be performed before empty.
overflow	Output	Overflow: Indicates a rejected write operation due to full FIFO.
underflow	Output	Underflow: Indicates a rejected read operation due to empty FIFO.
wr_ack	Output	Write Acknowledge: Indicates a successful write operation.

2. Verification plan

	A	B	C	D	E
1	Label	Description	Stimulus Generation	Functional Coverage (Later)	Functionality Check
2	Reset sequence	Initial Reset for the fifo	Directed at the start of the simulation		A checker in the testbench to make sure the output is correct when reset is asserted
3	Main Sequence	Randomizing the stimulus inputs of fifo divided into 3 sections (3 repeats) 1000 iterations for write only sequence,1000 for read only sequence and 1000 for write read iterations	Directed during the simulation		A checker in the testbench to make sure the output is correct when the transaction is sent
4	almostfull_p	when the reset deasserted and count is equal FIFO_DEPTH - 1,the almostfull flag is asserted	concurrent assert		concurrent assertion to check for the almostfull flag functionality
5	property full_p	when the reset deasserted and write enable is high and count is lower than FIFO_DEPTH,the full flag is asserted	concurrent assert		concurrent assertion to check for the full flag functionality
6	empty_p	when the reset deasserted and read enable is high and count is equal to 0 the empty flag is asserted	concurrent assert		concurrent assertion to check for the empty flag functionality
7	almostempty_p	when the reset deasserted and count is equal 1,the almostempty flag is asserted	concurrent assert		concurrent assertion to check for the almostempty flag functionality
8	overflow_p	when the reset deasserted and write enable is high and the full flag is asserted then the overflag flag is asserted	concurrent assert		concurrent assertion to check for the overflag flag functionality
9	underflow_p	when the reset deasserted and read enable is high and the empty flag is asserted then the underflag flag is asserted	concurrent assert		concurrent assertion to check for the underflow flag functionality
10	wr_ack_p	when the reset deasserted and write enable is high and the full flag is not asserted then the wr_ack flag is asserted	concurrent assert		concurrent assertion to check for the wr_ack flag functionality

10		wr_ack flag is asserted			
11	inc_count_p	when the reset deasserted and write enable is high and the full flag is not asserted then the wr_ack flag is asserted	concurrent assert		concurrent assertion to check for the count varaible functionality
12	dec_count_p	when the reset deasserted and write enable is high and the full flag is not asserted then the wr_ack flag is asserted	concurrent assert		concurrent assertion to check for the count varaible functionality
13	always_comb	cuz the rst is asynch. Signal so we must use always comb to chech the variables count and wr_ptr and rd_ptr	immediate assert		immediate assertion to check for the functionality of the var count and the pointers using assert final when the reset signal is asserted
14	Data_out_ref	to calculate the expected data_out	Directed		using the golden model (ref model) caculated in the scoreboard package
15	rst_n_cons	Constraint for reset to be deasserted most of the time	Constraint random		A checker in the testbench to make sure that all values are covered.
16	write_en_cons	Constraint for wr_en to be asserted with WR_EN_ON_DIST which is predetermined	Constraint random		A checker in the testbench to make sure that all values are covered.
17	read_en_cons	Constraint for rd_en to be asserted with RD_EN_ON_DIST which is predetermined	Constraint random		A checker in the testbench to make sure that all values are covered.

3. UVM Structure

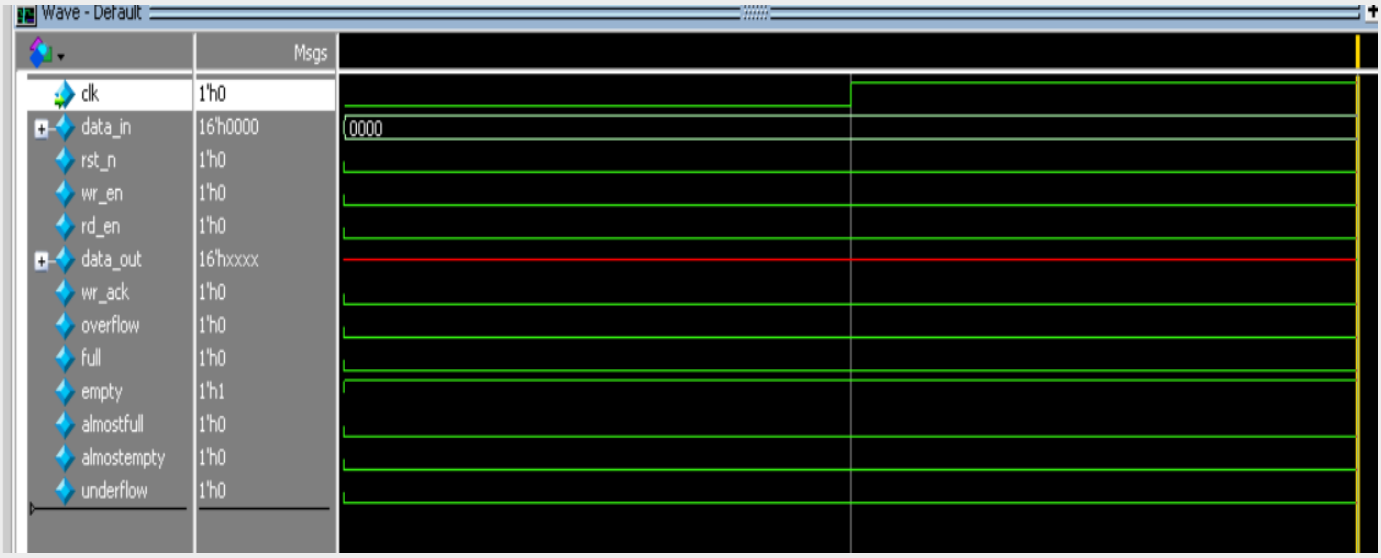

Fifo_top
Fifo_sva
Fifo_Test
Main sequence
Config_obj
Reset sequence
Fifo_env
Fifo_scoreboard
Fifo_coverage_collector
Fifo_Agent
Fifo_seq_item
Config_obj
Fifo_sequencer
Fifo_monitor
Fifo_driver
Fifo_vif
Fifo_vif
Fifo_interface
DUT

4- How it works?

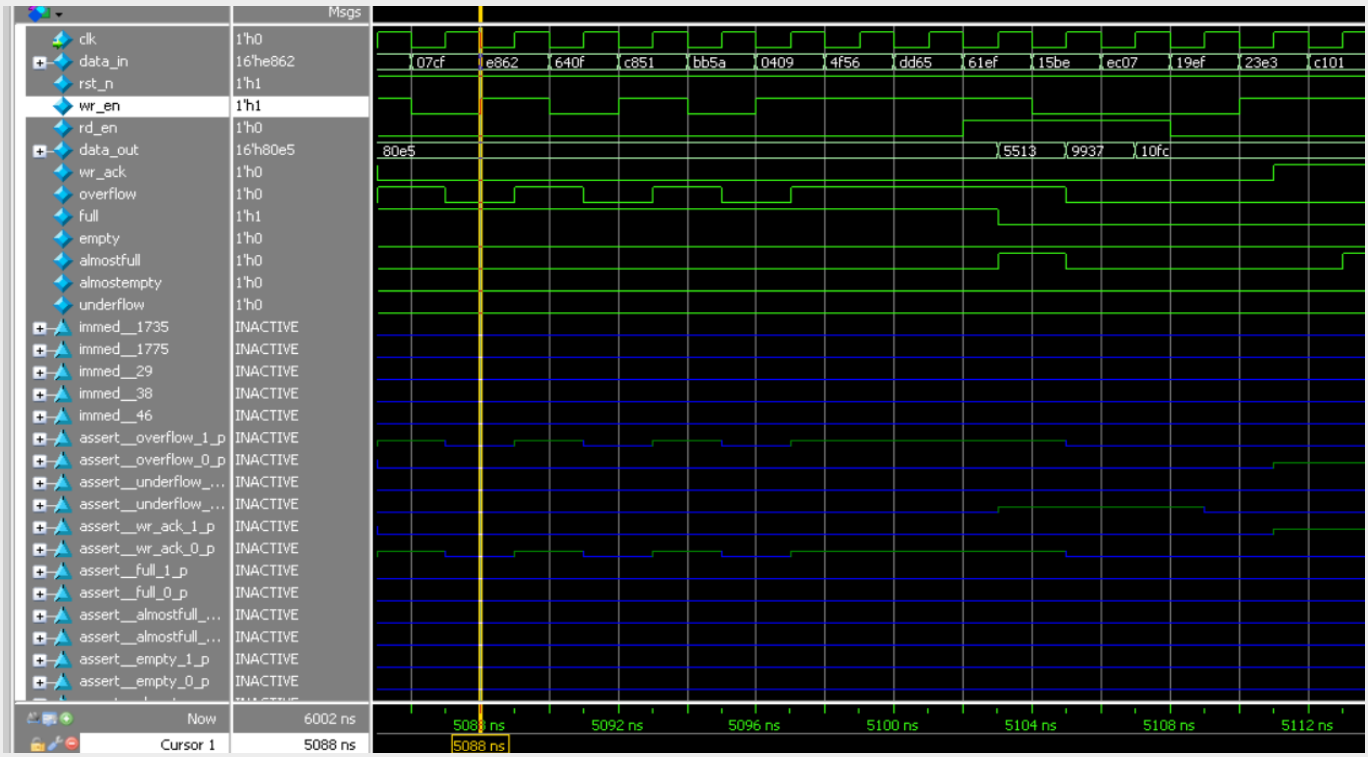
- The actual DUT which is the fifo only interacts back and forth with the interface (fifo_interface). The interface communicates receives stimulus from the driver (fifo_driver) using a virtual interface (fifo_vif) and sends to the DUT, while the interface sends what the DUT outputs to the monitor (fifo_monitor) using a virtual interface. This virtual interface is accessed through the configuration database (fifo_config_obj) which holds its pointer.
- The driver receives sequences from the sequencer (fifo_sequencer), the sequencer is responsible for regulating and organizing the sequences (reset_sequence & main_sequence) which are sequence items (fifo_seq_item).
- The agent (fifo_agent) encapsulates the monitor, driver, and the sequencer, it communicates using an analysis export with the scoreboard (fifo_scoreboard) and the coverage collector (fifo_coverage_collector).
- The scoreboard is responsible for checking the DUT's output against the reference model, and keeping count for the correct and error counts.
- The coverage collector is responsible for checking coverage of the testbench, using covergroups, coverpoints, and sampling.
- The environment (fifo_env) encapsulates the scoreboard, coverage collector, and the agent. The test encapsulates the environment and the sequences that will be passed to the sequencer.
- The top module encapsulates the whole environment and the assertions file (fifo_sva) and in it is instantiated the interface, DUT, and binded to the SVA file, and also where the clock is generated and passed to the whole testbench.

5- Questasim Snippets waveform

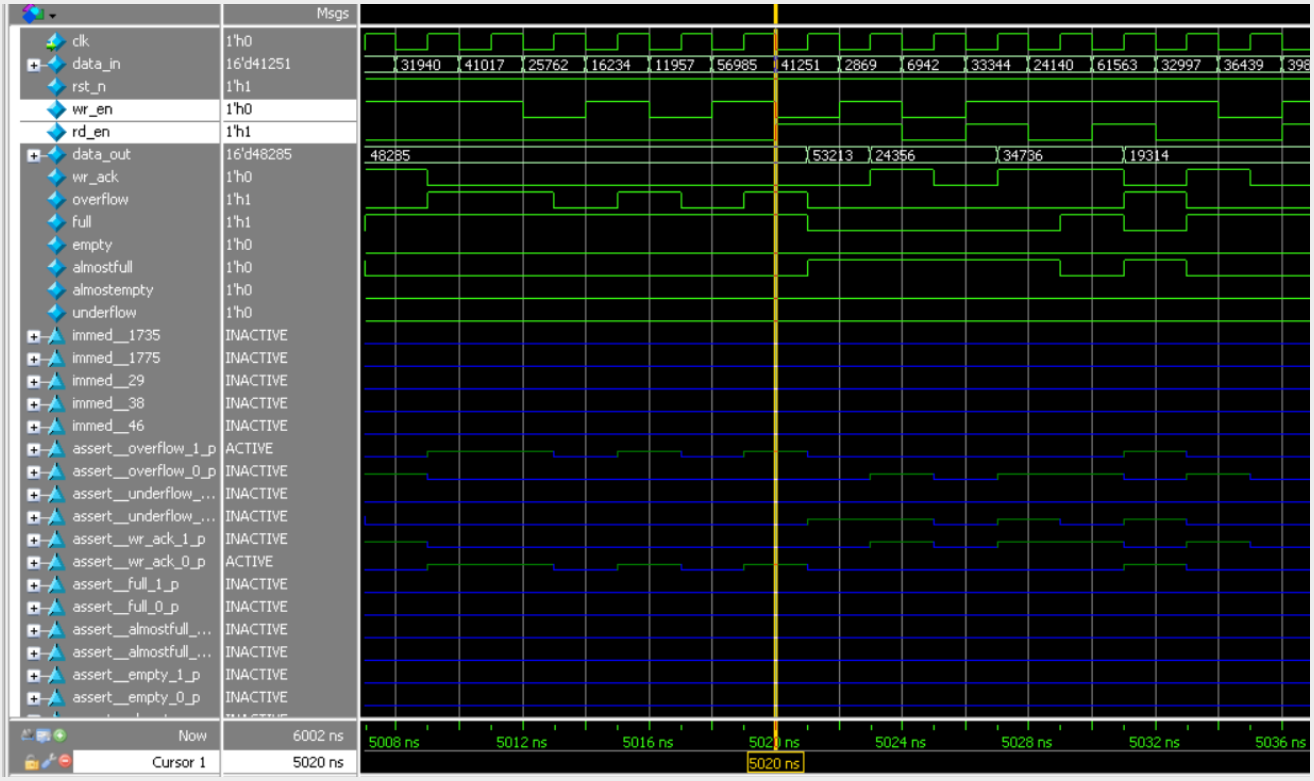
Reset sequence



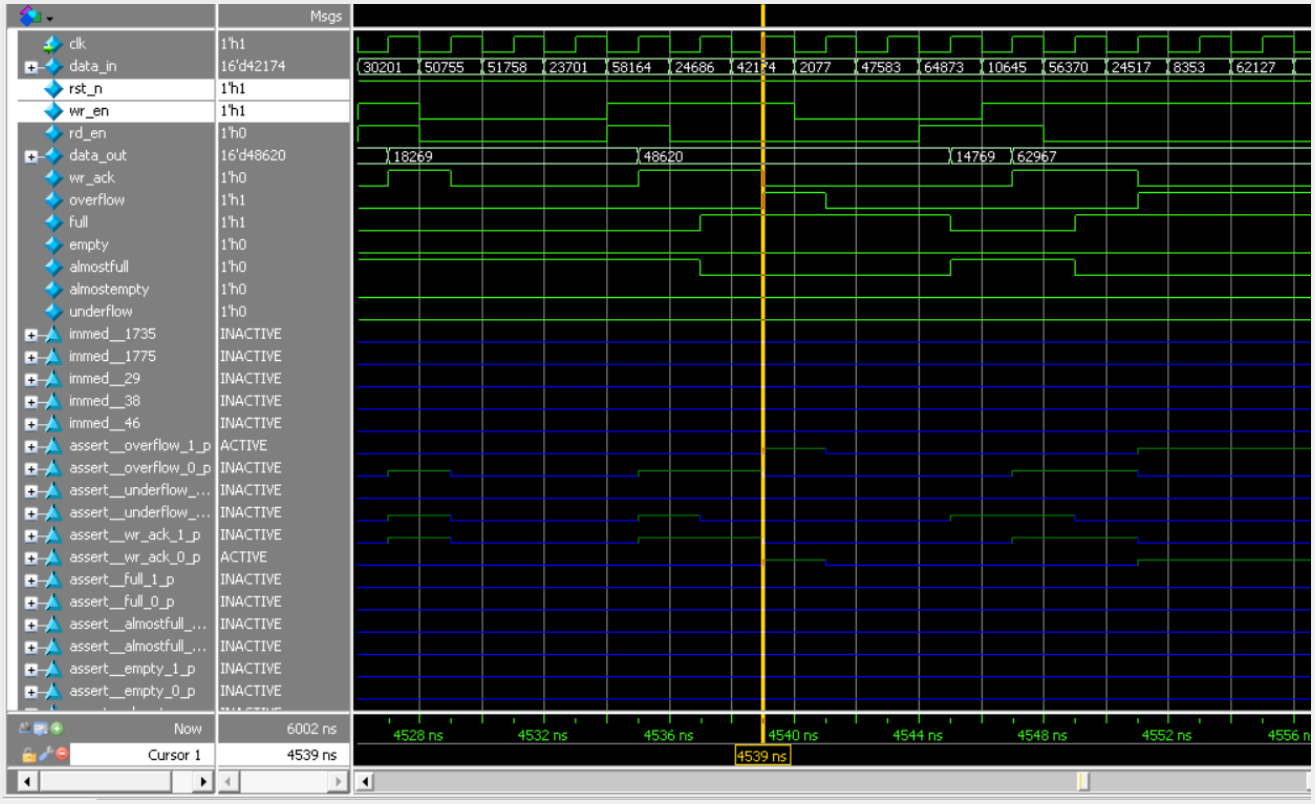
Write only sequence



Read sequence only



Write Read sequence



Cover groups

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage
/fifo_coverage_coll...		93.75%						
TYPE cov		93.75%	100	93.75%		✓	auto(1)	

Cover directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cum
/fifo_top/DUT/sva_inst/cover__rd_...	SVA	✓	Off	288	1	Unli...	1	100%		✓	0	0	0 ns	
/fifo_top/DUT/sva_inst/cover__wr_...	SVA	✓	Off	517	1	Unli...	1	100%		✓	0	0	0 ns	
/fifo_top/DUT/sva_inst/cover__sam...	SVA	✓	Off	111	1	Unli...	1	100%		✓	0	0	0 ns	
/fifo_top/DUT/sva_inst/cover__dec...	SVA	✓	Off	88	1	Unli...	1	100%		✓	0	0	0 ns	
/fifo_top/DUT/sva_inst/cover__inc_...	SVA	✓	Off	399	1	Unli...	1	100%		✓	0	0	0 ns	
/fifo_top/DUT/sva_inst/cover__alm...	SVA	✓	Off	2893	1	Unli...	1	100%		✓	0	0	0 ns	
/fifo_top/DUT/sva_inst/cover__alm...	SVA	✓	Off	56	1	Unli...	1	100%		✓	0	0	0 ns	
/fifo_top/DUT/sva_inst/cover__em...	SVA	✓	Off	1922	1	Unli...	1	100%		✓	0	0	0 ns	
/fifo_top/DUT/sva_inst/cover__em...	SVA	✓	Off	1079	1	Unli...	1	100%		✓	0	0	0 ns	
/fifo_top/DUT/sva_inst/cover__alm...	SVA	✓	Off	2702	1	Unli...	1	100%		✓	0	0	0 ns	
/fifo_top/DUT/sva_inst/cover__alm...	SVA	✓	Off	299	1	Unli...	1	100%		✓	0	0	0 ns	
/fifo_top/DUT/sva_inst/cover__full...	SVA	✓	Off	554	1	Unli...	1	100%		✓	0	0	0 ns	
/fifo_top/DUT/sva_inst/cover__full...	SVA	✓	Off	1154	1	Unli...	1	100%		✓	0	0	0 ns	
/fifo_top/DUT/sva_inst/cover__wr_...	SVA	✓	Off	1138	1	Unli...	1	100%		✓	0	0	0 ns	
/fifo_top/DUT/sva_inst/cover__wr_...	SVA	✓	Off	517	1	Unli...	1	100%		✓	0	0	0 ns	
/fifo_top/DUT/sva_inst/cover__und...	SVA	✓	Off	288	1	Unli...	1	100%		✓	0	0	0 ns	
/fifo_top/DUT/sva_inst/cover__und...	SVA	✓	Off	975	1	Unli...	1	100%		✓	0	0	0 ns	
/fifo_top/DUT/sva_inst/cover__ove...	SVA	✓	Off	517	1	Unli...	1	100%		✓	0	0	0 ns	
/fifo_top/DUT/sva_inst/cover__ove...	SVA	✓	Off	1138	1	Unli...	1	100%		✓	0	0	0 ns	

Assertions

Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression	Include
Immediate	SVA	on	0	0	-	-	-	-		off	assert ($cast(seq,o))	✗
Immediate	SVA	on	0	0	-	-	-	-		off	assert ($cast(seq,o))	✗
Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
Immediate	SVA	on	0	1	-	-	-	-		off	assert (randomize(...))	✓
Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) ((fifo...	✓
Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) ((fifo...	✓
Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) (DUT...	✓
Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) (DUT...	✓
Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) (DUT...	✓
Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) (DUT...	✓
Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	assert(@(posedge fifo_if.clk) disa...	✓
Immediate	SVA	on	0	1	-	-	-	-		off	assert (DUT.count==0)	✓
Immediate	SVA	on	0	1	-	-	-	-		off	assert (DUT.wr_ptr==0)	✓
Immediate	SVA	on	0	1	-	-	-	-		off	assert (DUT.rd_ptr==0)	✓

Transcript

```
# UVM_INFO fifo_scoreboard.sv(49) @ 1968: uvm_test_top.env.sb [Run_phase] Correct,Data_out = x while data_out_ref = x
# UVM_INFO fifo_scoreboard.sv(49) @ 1970: uvm_test_top.env.sb [Run_phase] Correct,Data_out = x while data_out_ref = x
# UVM_INFO fifo_scoreboard.sv(49) @ 1972: uvm_test_top.env.sb [Run_phase] Correct,Data_out = x while data_out_ref = x
# UVM_INFO fifo_scoreboard.sv(49) @ 1974: uvm_test_top.env.sb [Run_phase] Correct,Data_out = x while data_out_ref = x
# UVM_INFO fifo_scoreboard.sv(49) @ 1976: uvm_test_top.env.sb [Run_phase] Correct,Data_out = x while data_out_ref = x
# UVM_INFO fifo_scoreboard.sv(49) @ 1978: uvm_test_top.env.sb [Run_phase] Correct,Data_out = x while data_out_ref = x
# UVM_INFO fifo_scoreboard.sv(49) @ 1980: uvm_test_top.env.sb [Run_phase] Correct,Data_out = x while data_out_ref = x
# UVM_INFO fifo_scoreboard.sv(49) @ 1982: uvm_test_top.env.sb [Run_phase] Correct,Data_out = x while data_out_ref = x
# UVM_INFO fifo_scoreboard.sv(49) @ 1984: uvm_test_top.env.sb [Run_phase] Correct,Data_out = x while data_out_ref = x
# UVM_INFO fifo_scoreboard.sv(49) @ 1986: uvm_test_top.env.sb [Run_phase] Correct,Data_out = x while data_out_ref = x
# UVM_INFO fifo_scoreboard.sv(49) @ 1988: uvm_test_top.env.sb [Run_phase] Correct,Data_out = x while data_out_ref = x
# UVM_INFO fifo_scoreboard.sv(49) @ 1990: uvm_test_top.env.sb [Run_phase] Correct,Data_out = x while data_out_ref = x
# UVM_INFO fifo_scoreboard.sv(49) @ 1992: uvm_test_top.env.sb [Run_phase] Correct,Data_out = x while data_out_ref = x
# UVM_INFO fifo_scoreboard.sv(49) @ 1994: uvm_test_top.env.sb [Run_phase] Correct,Data_out = x while data_out_ref = x
# UVM_INFO fifo_scoreboard.sv(49) @ 1996: uvm_test_top.env.sb [Run_phase] Correct,Data_out = x while data_out_ref = x
# UVM_INFO fifo_scoreboard.sv(49) @ 1998: uvm_test_top.env.sb [Run_phase] Correct,Data_out = x while data_out_ref = x
# UVM_INFO fifo_scoreboard.sv(49) @ 2000: uvm_test_top.env.sb [Run_phase] Correct,Data_out = x while data_out_ref = x
# UVM_INFO fifo_scoreboard.sv(49) @ 2002: uvm_test_top.env.sb [Run_phase] Correct,Data_out = x while data_out_ref = x
# UVM_INFO fifo_scoreboard.sv(49) @ 2004: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 7141 while data_out_ref = 7141
# UVM_INFO fifo_scoreboard.sv(49) @ 2006: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 17318 while data_out_ref = 17318
# UVM_INFO fifo_scoreboard.sv(49) @ 2008: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2010: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2012: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2014: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2016: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2018: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2020: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2022: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2024: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2026: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2028: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2030: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2032: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2034: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2036: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2038: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2040: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2042: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2044: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2046: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2048: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2050: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
# UVM_INFO fifo_scoreboard.sv(49) @ 2052: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42849 while data_out_ref = 42849
```

```
# UVM_INFO fifo_scoreboard.sv(49) @ 4330: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 7830 while data_out_ref = 7830
# UVM_INFO fifo_scoreboard.sv(49) @ 4332: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 63492 while data_out_ref = 63492
# UVM_INFO fifo_scoreboard.sv(49) @ 4334: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 63492 while data_out_ref = 63492
# UVM_INFO fifo_scoreboard.sv(49) @ 4336: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 63492 while data_out_ref = 63492
# UVM_INFO fifo_scoreboard.sv(49) @ 4338: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 63492 while data_out_ref = 63492
# UVM_INFO fifo_scoreboard.sv(49) @ 4340: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 63455 while data_out_ref = 63455
# UVM_INFO fifo_scoreboard.sv(49) @ 4342: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 63455 while data_out_ref = 63455
# UVM_INFO fifo_scoreboard.sv(49) @ 4344: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 34475 while data_out_ref = 34475
# UVM_INFO fifo_scoreboard.sv(49) @ 4346: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 34475 while data_out_ref = 34475
# UVM_INFO fifo_scoreboard.sv(49) @ 4348: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 34475 while data_out_ref = 34475
# UVM_INFO fifo_scoreboard.sv(49) @ 4350: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 60631 while data_out_ref = 60631
# UVM_INFO fifo_scoreboard.sv(49) @ 4352: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 60631 while data_out_ref = 60631
# UVM_INFO fifo_scoreboard.sv(49) @ 4354: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 60631 while data_out_ref = 60631
# UVM_INFO fifo_scoreboard.sv(49) @ 4356: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 8626 while data_out_ref = 8626
# UVM_INFO fifo_scoreboard.sv(49) @ 4358: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 8626 while data_out_ref = 8626
# UVM_INFO fifo_scoreboard.sv(49) @ 4360: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 8626 while data_out_ref = 8626
# UVM_INFO fifo_scoreboard.sv(49) @ 4362: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 8626 while data_out_ref = 8626
# UVM_INFO fifo_scoreboard.sv(49) @ 4364: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 8626 while data_out_ref = 8626
# UVM_INFO fifo_scoreboard.sv(49) @ 4366: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 8626 while data_out_ref = 8626
# UVM_INFO fifo_scoreboard.sv(49) @ 4368: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 8626 while data_out_ref = 8626
# UVM_INFO fifo_scoreboard.sv(49) @ 4370: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 8626 while data_out_ref = 8626
# UVM_INFO fifo_scoreboard.sv(49) @ 4372: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 8626 while data_out_ref = 8626
# UVM_INFO fifo_scoreboard.sv(49) @ 4374: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 8626 while data_out_ref = 8626
# UVM_INFO fifo_scoreboard.sv(49) @ 4376: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 8626 while data_out_ref = 8626
# UVM_INFO fifo_scoreboard.sv(49) @ 4378: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42274 while data_out_ref = 42274
# UVM_INFO fifo_scoreboard.sv(49) @ 4380: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42274 while data_out_ref = 42274
# UVM_INFO fifo_scoreboard.sv(49) @ 4382: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42274 while data_out_ref = 42274
# UVM_INFO fifo_scoreboard.sv(49) @ 4384: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42274 while data_out_ref = 42274
# UVM_INFO fifo_scoreboard.sv(49) @ 4386: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42274 while data_out_ref = 42274
# UVM_INFO fifo_scoreboard.sv(49) @ 4388: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 7538 while data_out_ref = 7538
# UVM_INFO fifo_scoreboard.sv(49) @ 4390: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42145 while data_out_ref = 42145
# UVM_INFO fifo_scoreboard.sv(49) @ 4392: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 42145 while data_out_ref = 42145
# UVM_INFO fifo_scoreboard.sv(49) @ 4394: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 6527 while data_out_ref = 6527
# UVM_INFO fifo_scoreboard.sv(49) @ 4396: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 21962 while data_out_ref = 21962
# UVM_INFO fifo_scoreboard.sv(49) @ 4398: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 21962 while data_out_ref = 21962
# UVM_INFO fifo_scoreboard.sv(49) @ 4400: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 21962 while data_out_ref = 21962
# UVM_INFO fifo_scoreboard.sv(49) @ 4402: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 62676 while data_out_ref = 62676
# UVM_INFO fifo_scoreboard.sv(49) @ 4404: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 62676 while data_out_ref = 62676
# UVM_INFO fifo_scoreboard.sv(49) @ 4406: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 45993 while data_out_ref = 45993
# UVM_INFO fifo_scoreboard.sv(49) @ 4408: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 37858 while data_out_ref = 37858
```

```
# UVM_INFO fifo_scoreboard.sv(49) @ 5978: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 18704 while data_out_ref = 18704
# UVM_INFO fifo_scoreboard.sv(49) @ 5980: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 34267 while data_out_ref = 34267
# UVM_INFO fifo_scoreboard.sv(49) @ 5982: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 34267 while data_out_ref = 34267
# UVM_INFO fifo_scoreboard.sv(49) @ 5984: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 34267 while data_out_ref = 34267
# UVM_INFO fifo_scoreboard.sv(49) @ 5986: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 34267 while data_out_ref = 34267
# UVM_INFO fifo_scoreboard.sv(49) @ 5988: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 34267 while data_out_ref = 34267
# UVM_INFO fifo_scoreboard.sv(49) @ 5990: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 34267 while data_out_ref = 34267
# UVM_INFO fifo_scoreboard.sv(49) @ 5992: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 24766 while data_out_ref = 24766
# UVM_INFO fifo_scoreboard.sv(49) @ 5994: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 11790 while data_out_ref = 11790
# UVM_INFO fifo_scoreboard.sv(49) @ 5996: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 11790 while data_out_ref = 11790
# UVM_INFO fifo_scoreboard.sv(49) @ 5998: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 11790 while data_out_ref = 11790
# UVM_INFO fifo_scoreboard.sv(49) @ 6000: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 11790 while data_out_ref = 11790
# UVM_INFO fifo_scoreboard.sv(49) @ 6002: uvm_test_top.env.sb [Run_phase] Correct,Data_out = 63625 while data_out_ref = 63625
# UVM_INFO fifo_test.sv(48) @ 6002: uvm_test_top [run_phase] Stimulus Generation Ended
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 6002: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO fifo_scoreboard.sv(82) @ 6002: uvm_test_top.env.sb [Report_phase] Succseeful checks : 3001
# UVM_INFO fifo_scoreboard.sv(83) @ 6002: uvm_test_top.env.sb [Report_phase] Unsuccseeful checks : 0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO : 3011
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [Report_phase]     2
# [Run_phase]  3001
# [TEST_DONE]     1
# [run_phase]     4
# ** Note: $finish    : C:/questasim64_2021.1/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 6002 ns  Iteration: 61  Instance: /fifo_top
# 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
```

6- Code and functional coverage.

```
project > = FIFO_coverage.txt
1   Coverage Report by instance with details
2
3   ================================================================================
4   === Instance: /fifo_top/DUT/sva_inst
5   === Design Unit: work.fifo_sva
6   ================================================================================
7
8   Assertion Coverage:
9       Assertions                      22        22         0   100.00%
10  ----------------------------------------------------------------
11  Name                 File(Line)                   Failure      Pass
12                                                    Count        Count
13  ----------------------------------------------------------------
14  /fifo_top/DUT/sva_inst/assert__rd_ptr_p
15                       fifo_sva.sv(119)                   0          1
16  /fifo_top/DUT/sva_inst/assert__wr_ptr_p
17                       fifo_sva.sv(118)                   0          1
18  /fifo_top/DUT/sva_inst/assert__same_count_p
19                       fifo_sva.sv(104)                   0          1
20  /fifo_top/DUT/sva_inst/assert__dec_count_p
21                       fifo_sva.sv(103)                   0          1
```

```
================================================================================

Assertion Coverage:
    Assertions                      3         3         0   100.00%
----------------------------------------------------------------
Name                 File(Line)                   Failure      Pass
                                                  Count        Count
----------------------------------------------------------------
/fifo_main_sequence_pkg/fifo_main_sequence/body/#ublk#124017991#18/immed__29
                     fifo_main_sequence.sv(29)          0          1
/fifo_main_sequence_pkg/fifo_main_sequence/body/#ublk#124017991#33/immed__38
                     fifo_main_sequence.sv(38)          0          1
/fifo_main_sequence_pkg/fifo_main_sequence/body/#ublk#124017991#42/immed__46
                     fifo_main_sequence.sv(46)          0          1

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

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

Covergroup Coverage:
    Covergroups                     1        na        na    93.75%
        Coverpoints/Crosses        18        na        na        na
            Covergroup Bins       140       131         9    93.57%
--------------------------------------------------------------------------------------------------------------
Covergroup                                              Metric       Goal       Bins    Status

--------------------------------------------------------------------------------------------------------------
 TYPE /fifo_coverage_collector_pkg/fifo_coverage_collector/cov
                                                        93.75%        100          -    Uncovered
    covered/total bins:                                    131        140          -
    missing/total bins:                                      9        140          -
    % Hit:                                              93.57%        100          -
    Coverpoint data_in_cvp                             100.00%        100          -    Covered
        covered/total bins:                                 64         64          -
        missing/total bins:                                  0         64          -
        % Hit:                                         100.00%        100          -
```

7- Bug report

- **Almost full flag must be (count == FIFO_DEPTH -1) instead of (count ==FIFO_DEPTH -2)**
- **Underflow signal is a sequential output can’t be used in assign statement, must be handle it in always block.**

8- Assertion table

Feature	Assertion
When reset is asserted, wr_ack is low	@(posedge fifo_if.clk) !rst_n |=> !wr_ack
When writing the last count, full flag will be high	@(posedge fifo_if.clk) disable iff(!rst_n) (wr_en && count == FIFO_DEPTH) |-> full
When writing the next to last count, almostfull flag will be high	@(posedge fifo_if.clk) disable iff(!rst_n) (wr_en && count == FIFO_DEPTH-1) |-> almostfull
When reading the last count, empty flag will be high	@(posedge fifo_if.clk) disable iff(!rst_n) (rd_en && count == 0) |-> empty
When reading the next to last count, almostempty flag will be high	@(posedge fifo_if.clk) disable iff(!rst_n) (rd_en && count == 1) |-> almostempty
When writing and counter is full, overflow flag will be high	@(posedge fifo_if.clk) disable iff(!rst_n) (full && wr_en) |=> overflow;
When reading and counter is empty, underflow flag will be high	@(posedge fifo_if.clk) disable iff(!rst_n) (empty && rd_en) |=> underflow;
When writing and counter is not full, wr_ack will be high	@(posedge fifo_if.clk) disable iff(!rst_n) (wr_en && (!full)) |=> wr_ack
When reset is asserted, count will be low	@(posedge fifo_if.clk) ! rst_n |=> ! count

When reset is asserted, wr_ack is low	@(posedge fifo_if.clk) !rst_n |=> !wr_ack
When writing the last count, full flag will be high	@(posedge fifo_if.clk) disable iff(!rst_n) (wr_en && count == FIFO_DEPTH) |-> full
When writing the next to last count, almostfull flag will be high	@(posedge fifo_if.clk) disable iff(!rst_n) (wr_en && count == FIFO_DEPTH-1) |-> almostfull
When reading the last count, empty flag will be high	@(posedge fifo_if.clk) disable iff(!rst_n) (rd_en && count == 0) |-> empty

9- Do file.

UVM project > do_file.do

```
1   vlib work
2   vlog -f src.txt
3   vsim -voptargs=+acc work.fifo_top -classdebug -uvmcontrol=all
4   add wave /fifo_top/fifo_if/*
5   coverage save fifo_top.ucdb -onexit
6   run -all
```

Src.txt

UVM project > src.txt

```
1   FIFO.sv
2   fifo_interface.sv
3   fifo_seq_item.sv
4   fifo_monitor.sv
5   fifo_config_obj.sv
6   fifo_driver.sv
7   fifo_sequencer.sv
8   fifo_agent.sv
9   fifo_scoreboard.sv
10  fifo_coverage_collector.sv
11  fifo_env.sv
12  fifo_reset_sequence.sv
13  fifo_main_sequence.sv
14  fifo_test.sv
15  fifo_sva.sv
16  fifo_top.sv
```

10- Project codes

Fifo_top

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

import fifo_test_pkg::*;

module fifo_top();
    bit clk;

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

    fifo_interface fifo_if(clk);
    FIFO DUT(fifo_if);

    //Bind Assertion(correlate the assertion with the design)
    bind FIFO fifo_sva sva_inst(fifo_if);

    initial begin
        uvm_config_db #(virtual
fifo_interface)::set(null,"uvm_test_top","FIFO_IF",fifo_if);
        run_test("fifo_test");
    end
endmodule
```

fifo_test

```
package fifo_test_pkg;
    import uvm_pkg::*;
    `include"uvm_macros.svh"
    import fifo_env_pkg::*;
    import fifo_config_obj_pkg::*;
    import fifo_main_sequence_pkg::*;
    import fifo_reset_sequence_pkg::*;

    class fifo_test extends uvm_test;
        `uvm_component_utils(fifo_test);
        fifo_env env;
        fifo_config_obj fifo_CFG;
        virtual fifo_interface fifo_vif;
        fifo_main_sequence main_seq;
        fifo_reset_sequence rst_seq;

        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_CFG = fifo_config_obj::type_id::create("fifo_CFG",this);
            main_seq = fifo_main_sequence::type_id::create("main_seq",this);
            rst_seq = fifo_reset_sequence::type_id::create("rst_seq",this);

            if(!uvm_config_db #(virtual
fifo_interface)::get(this,"","FIFO_IF",fifo_CFG.fifo_vif))begin
                `uvm_fatal("bulid phase","Unable to get the virtual interface from
the configuration database from the top");
            end

            uvm_config_db #(fifo_config_obj)::set(this,"*","CFG",fifo_CFG);
        endfunction

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

            //Reset sequence
            `uvm_info("run_phase","Reset Asserted",UVM_LOW)
            rst_seq.start(env.agt.sqr);
```

```
            `uvm_info("run_phase","Reset Deasserted",UVM_LOW)
            //Main sequence
            `uvm_info("run_phase","Stimulus Generation Started",UVM_LOW)
            main_seq.start(env.agt.sqr);
            `uvm_info("run_phase","Stimulus Generation Ended",UVM_LOW)

            phase.drop_objection(this);
        endtask

    endclass

endpackage
```

fifo_config_obj

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

    class fifo_config_obj extends uvm_object;
        `uvm_object_utils(fifo_config_obj);
        virtual fifo_interface fifo_vif;

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

Reset sequence

```
package fifo_reset_sequence_pkg;

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

    class fifo_reset_sequence extends uvm_sequence #(fifo_seq_item);
        `uvm_object_utils(fifo_reset_sequence);
        fifo_seq_item seq_item;

        function new(string name = "fifo_reset_sequence");
```

```
            super.new(name);
        endfunction

        task body;
            seq_item = fifo_seq_item::type_id::create("seq_item");
            start_item(seq_item);
            seq_item.rst_n = 0;
            seq_item.data_in = 0;
            seq_item.wr_en = 0;
            seq_item.rd_en = 0;
            finish_item(seq_item);
        endtask
    endclass

endpackage
```

Main sequence

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

    class fifo_main_sequence extends uvm_sequence #(fifo_seq_item);
        `uvm_object_utils(fifo_main_sequence);
        fifo_seq_item seq_item;

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

        task body;
            //write_only_sequence
            repeat(1000)begin
                seq_item = fifo_seq_item::type_id::create("seq_item");
                start_item(seq_item);

                //disable the all constarints
                seq_item.constraint_mode(0);

                /*here only enable the reset constraint and keep the others
disabled,
```

```
                cus here i want to write only and no need to the distribution of the
write constraint..
                i want the write operation 100 % */
                seq_item.rst_n_cons.constraint_mode(1);
                assert(seq_item.randomize() with{wr_en == 1;rd_en == 0;});  //soft
constraint
                finish_item(seq_item);
            end
            //read_only_sequence
            repeat(1000)begin
                seq_item = fifo_seq_item::type_id::create("seq_item");
                start_item(seq_item);
                seq_item.constraint_mode(0);
                seq_item.rst_n_cons.constraint_mode(1);
                assert(seq_item.randomize() with{wr_en == 0;rd_en == 1;});
                finish_item(seq_item);
            end
            // //write_read_sequence
            repeat(1000)begin
                seq_item = fifo_seq_item::type_id::create("seq_item");
                start_item(seq_item);
                //here write & read together so i need to the constraints of wr and
rd.
                assert(seq_item.randomize());
                finish_item(seq_item);
            end
        endtask
    endclass

endpackage
```

fifo_env

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

    import fifo_scoreboard_pkg::*;
    import fifo_coverage_collector_pkg::*;
    import fifo_agent_pkg::*;

    class fifo_env extends uvm_env;
        `uvm_component_utils(fifo_env);
        fifo_scoreboard sb;
        fifo_coverage_collector cov;
        fifo_agent agt;

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

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
            sb = fifo_scoreboard::type_id::create("sb",this);
            cov = fifo_coverage_collector::type_id::create("cov",this);
            agt = fifo_agent::type_id::create("agt",this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            agt.agt_ap.connect(sb.sb_export);
            agt.agt_ap.connect(cov.cov_export);
        endfunction
    endclass

endpackage
```

fifo_interface

```
    interface fifo_interface(clk);
        /*******Parameters******/
        parameter FIFO_WIDTH = 16;
        parameter FIFO_DEPTH = 8;
        /*****Signals ports*****/
        input bit clk;
        logic [FIFO_WIDTH-1:0] data_in;
        logic rst_n, wr_en, rd_en;
        logic [FIFO_WIDTH-1:0] data_out;
        logic wr_ack, overflow,full, empty, almostfull, almostempty, underflow;

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

    endinterface
```

fifo_scoreboard

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

    class fifo_scoreboard extends uvm_scoreboard;
        `uvm_component_utils(fifo_scoreboard);
        uvm_analysis_export #(fifo_seq_item)sb_export;
        uvm_tlm_analysis_fifo#(fifo_seq_item)sb_fifo;
        fifo_seq_item sb_seq_item;

        /*Related to ref model*/
        parameter FIFO_WIDTH = 16;
        parameter FIFO_DEPTH = 8;
        logic [FIFO_WIDTH-1:0]fifo_queue[$];
        logic [FIFO_WIDTH-1:0]data_out_ref;

        integer correct_count = 0;
        integer error_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);
            sb_export = new("cov_export",this);
            sb_fifo = new("sb_fifo",this);
        endfunction

         function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            //connect the analysis export of the coverage with the analysis export
of the fifo(internally the coverage)
            sb_export.connect(sb_fifo.analysis_export);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                sb_fifo.get(sb_seq_item);
                ref_model(sb_seq_item);
                if(sb_seq_item.data_out != data_out_ref)begin
                    `uvm_error("Run_phase",$sformatf("Failed,Data_out = %0d while
data_out_ref = %0d",sb_seq_item.data_out,data_out_ref));
                    error_count++;
                end
                else if (sb_seq_item.data_out === data_out_ref)begin
                    `uvm_info("Run_phase",$sformatf("Correct,Data_out = %0d while
data_out_ref = %0d",sb_seq_item.data_out,data_out_ref),UVM_LOW);
                    correct_count++;
                end
            end
        endtask

        task ref_model(fifo_seq_item chk_seq_item);
            if(!chk_seq_item.rst_n)begin
                fifo_queue.delete();
            end
            else begin
                if(chk_seq_item.wr_en && chk_seq_item.rd_en && fifo_queue.size() !=
0 && fifo_queue.size() != FIFO_DEPTH)begin
                   data_out_ref = fifo_queue.pop_back();
                   fifo_queue.push_front(chk_seq_item.data_in);
                end
                else if(chk_seq_item.wr_en && chk_seq_item.rd_en
&&  fifo_queue.size() == FIFO_DEPTH)begin
                    data_out_ref = fifo_queue.pop_back();
```

```
                end
                else if(chk_seq_item.wr_en && chk_seq_item.rd_en
&&  fifo_queue.size() == 0)begin
                    fifo_queue.push_front(chk_seq_item.data_in);
                end
                else if(chk_seq_item.wr_en && fifo_queue.size() < FIFO_DEPTH)begin
                    fifo_queue.push_front(chk_seq_item.data_in);
                end
                else if(chk_seq_item.rd_en &&  fifo_queue.size() != 0)begin
                    data_out_ref = fifo_queue.pop_back();
                end
            end
        endtask


        function void report_phase(uvm_phase phase);
            super.report_phase(phase);
            `uvm_info("Report_phase",$sformatf("Succseeful checks :
%0d",correct_count),UVM_MEDIUM);
            `uvm_info("Report_phase",$sformatf("Unsuccseeful checks :
%0d",error_count),UVM_MEDIUM);
        endfunction

    endclass


endpackage
```

fifo_coverage_collector

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

    import fifo_seq_item_pkg::*;

    class fifo_coverage_collector extends uvm_component;
        `uvm_component_utils(fifo_coverage_collector);
        uvm_analysis_export #(fifo_seq_item)cov_export;
        uvm_tlm_analysis_fifo#(fifo_seq_item)cov_fifo;
        fifo_seq_item cov_seq_item;

        //Autogenerate for all Signals except the output data_out
        covergroup cov;
```

```
            data_in_cvp: coverpoint cov_seq_item.data_in;
            rst_n_cvp: coverpoint cov_seq_item.rst_n;
            wr_en_cvp: coverpoint cov_seq_item.wr_en;
            rd_en_cvp: coverpoint cov_seq_item.rd_en;
            wr_ack_cvp: coverpoint cov_seq_item.wr_ack;
            overflow_cvp: coverpoint cov_seq_item.overflow;
            full_cvp: coverpoint cov_seq_item.full;
            empty_cvp: coverpoint cov_seq_item.empty;
            almostfull_cvp: coverpoint cov_seq_item.almostfull;
            almostempty_cvp: coverpoint cov_seq_item.almostempty;
            underflow_cvp: coverpoint cov_seq_item.underflow;

            CR_full:cross wr_en_cvp,rd_en_cvp,full_cvp;
            CR_almostfull:cross wr_en_cvp,rd_en_cvp,almostfull_cvp;
            CR_almostempty:cross wr_en_cvp,rd_en_cvp,almostempty_cvp;
            CR_empty:cross wr_en_cvp,rd_en_cvp,empty_cvp;
            CR_overflow:cross wr_en_cvp,rd_en_cvp,overflow_cvp;
            CR_underflow:cross wr_en_cvp,rd_en_cvp,underflow_cvp;
            CR_wr_ack:cross wr_en_cvp,rd_en_cvp,wr_ack_cvp;
        endgroup

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

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

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            //connect the analysis export of the coverage with the analysis export
of the fifo(internally the coverage)
            cov_export.connect(cov_fifo.analysis_export);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                //get the next item from the fifo
                cov_fifo.get(cov_seq_item);
```

```
                cov.sample();
            end
        endtask
    endclass

endpackage
```

fifo_agent

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

    import fifo_sequencer_pkg::*;
    import fifo_driver_pkg::*;
    import fifo_monitor_pkg::*;
    import fifo_config_obj_pkg::*;
    import fifo_seq_item_pkg::*;

    class fifo_agent extends uvm_agent;
        `uvm_component_utils(fifo_agent);

        fifo_sequencer sqr;
        fifo_driver drv;
        fifo_monitor mon;
        fifo_config_obj fifo_CFG;
        uvm_analysis_port #(fifo_seq_item) agt_ap;

        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_obj)::get(this,"","CFG",fifo_CFG))begin
                `uvm_fatal("build_phase","Agent unable to get the configuration
object")
            end

            sqr = fifo_sequencer::type_id::create("sqr",this);
            drv = fifo_driver::type_id::create("drv",this);
            mon = fifo_monitor::type_id::create("mon",this);
```

```
            agt_ap = new("agt_ap",this);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            drv.fifo_vif = fifo_CFG.fifo_vif;
            mon.fifo_vif = fifo_CFG.fifo_vif;
            drv.seq_item_port.connect(sqr.seq_item_export);
            mon.mon_ap.connect(agt_ap);//connect the analysis port in the monitor
with the analysis port in the agent.
        endfunction
    endclass

endpackage
```

fifo_sequencer

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

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

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

fifo_driver

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

    import fifo_config_obj_pkg::*;
    import fifo_seq_item_pkg::*;

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

```
        virtual fifo_interface fifo_vif;
        // fifo_config_obj fifo_CFG;
        fifo_seq_item stim_seq_item;

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

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
        endfunction

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

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                stim_seq_item = fifo_seq_item::type_id::create("stim_seq_item");
                seq_item_port.get_next_item(stim_seq_item); //pull the seq item from
the sqr

                fifo_vif.rst_n = stim_seq_item.rst_n;
                fifo_vif.data_in = stim_seq_item.data_in;
                fifo_vif.wr_en = stim_seq_item.wr_en;
                fifo_vif.rd_en = stim_seq_item.rd_en;

                @(negedge fifo_vif.clk);
                seq_item_port.item_done();
                `uvm_info("Run_phase",stim_seq_item.convert2string_stimulus(),UVM_HI
GH);
            end
        endtask
    endclass


endpackage
```

fifo_monitor

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

    class fifo_monitor extends uvm_monitor;
        `uvm_component_utils(fifo_monitor);
        virtual fifo_interface fifo_vif;
        fifo_seq_item mon_seq_item;
        uvm_analysis_port #(fifo_seq_item)mon_ap;

        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);
            mon_ap = new("mon_ap",this);
        endfunction
        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            forever begin
                mon_seq_item = fifo_seq_item::type_id::create("mon_seq_item");
                @(negedge fifo_vif.clk);
                mon_seq_item.rst_n = fifo_vif.rst_n;
                mon_seq_item.data_in = fifo_vif.data_in;
                mon_seq_item.wr_en = fifo_vif.wr_en;
                mon_seq_item.rd_en = fifo_vif.rd_en;
                //outputs
                mon_seq_item.wr_ack = fifo_vif.wr_ack;
                mon_seq_item.overflow = fifo_vif.overflow;
                mon_seq_item.full = fifo_vif.full;
                mon_seq_item.empty = fifo_vif.empty;
                mon_seq_item.almostfull = fifo_vif.almostfull;
                mon_seq_item.almostempty = fifo_vif.almostempty;
                mon_seq_item.underflow = fifo_vif.underflow;
                mon_seq_item.data_out = fifo_vif.data_out;
                mon_ap.write(mon_seq_item);
                `uvm_info("Run_phase",mon_seq_item.convert2string(),UVM_HIGH);
            end
        endtask
    endclass
endpackage
```

fifo_seq_item

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

    class fifo_seq_item extends uvm_sequence_item;
        `uvm_object_utils(fifo_seq_item);
        parameter FIFO_WIDTH = 16;
        parameter FIFO_DEPTH = 8;
        rand logic [FIFO_WIDTH-1:0] data_in;
        rand logic rst_n, wr_en, rd_en;
        logic [FIFO_WIDTH-1:0]data_out;
        logic wr_ack, overflow,full, empty, almostfull, almostempty, underflow;

        integer RD_EN_ON_DIST = 30;
        integer WR_EN_ON_DIST = 70;


        //Constraints
        constraint rst_n_cons
        {
            rst_n dist{0:=2,1:=98}; //active low
        }
        constraint write_en_cons
        {
            wr_en dist{1:=WR_EN_ON_DIST,0:=(100-WR_EN_ON_DIST)};
        }
        constraint read_en_cons
        {
            rd_en dist{1:=RD_EN_ON_DIST,0:=(100-RD_EN_ON_DIST)};
        }

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

        function string convert2string();
            return $sformatf("%s reset = %0d,datain = %0d,wr_en = %0d,rd_en =
%0d,data_out = %0d,wr_ack = %0d,overflow = %0d,full = %0d,empty = %0d,almostfull =
%0d,almostempty = %0d,underflow = %0d",super.convert2string(),
            rst_n,data_in,wr_en,rd_en,data_out,wr_ack,overflow,full,empty,almostfull
,almostempty,underflow);
        endfunction
```

```
        function string convert2string_stimulus();
            return $sformatf("reset = %0d,datain = %0d,wr_en = %0d,rd_en =
%0d",rst_n,data_in,wr_en,rd_en);
        endfunction
    endclass


endpackage
```

FIFO design

```
module FIFO(fifo_interface.DUT fifo_if);

parameter FIFO_WIDTH = 16;
parameter FIFO_DEPTH = 8;

localparam max_fifo_addr = $clog2(FIFO_DEPTH);

reg [FIFO_WIDTH-1:0] mem [FIFO_DEPTH-1:0];

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

always @(posedge fifo_if.clk or negedge fifo_if.rst_n) begin
    if (!fifo_if.rst_n) begin
        wr_ptr <= 0;
        fifo_if.wr_ack <= 0;    //added
        fifo_if.overflow <= 0;  //added
    end
    else if (fifo_if.wr_en && count < FIFO_DEPTH) begin
        mem[wr_ptr] <= fifo_if.data_in;
        fifo_if.wr_ack <= 1;
        wr_ptr <= wr_ptr + 1;
        fifo_if.overflow <= 0;  //added
    end
    else if((fifo_if.wr_en && fifo_if.rd_en && fifo_if.empty))begin //added
        mem[wr_ptr] <= fifo_if.data_in;
        fifo_if.wr_ack <= 1;
        wr_ptr <= wr_ptr + 1;
        fifo_if.overflow <= 0;  //added
    end
    else begin
        fifo_if.wr_ack <= 0;
        if (fifo_if.full && fifo_if.wr_en)
```

```
            fifo_if.overflow <= 1;
        else begin
            fifo_if.overflow <= 0;
        end
    end
end

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

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

```
end

//Combinational logic
assign fifo_if.full = (count == FIFO_DEPTH)? 1 : 0;
assign fifo_if.empty = (count == 0)? 1 : 0;
assign fifo_if.almostfull = (count == FIFO_DEPTH-1)? 1 : 0;
assign fifo_if.almostempty = (count == 1)? 1 : 0;

endmodule
```

fifo_sva

```
module fifo_sva(fifo_interface.DUT fifo_if);
   parameter FIFO_WIDTH = 16;
   parameter FIFO_DEPTH = 8;

property overflow_1_p;
    @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) ((DUT.count == FIFO_DEPTH) &&
fifo_if.wr_en) |=> fifo_if.overflow;
endproperty
property overflow_0_p;
    @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) ((DUT.count != FIFO_DEPTH) &&
fifo_if.wr_en) |=> (fifo_if.overflow == 0);
endproperty
assert property (overflow_1_p);
assert property (overflow_0_p);
cover property (overflow_1_p);
cover property (overflow_0_p);

/**fifo_if.underflow**/
property underflow_1_p;
    @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) ((DUT.count == 0) &&
fifo_if.rd_en) |=> fifo_if.underflow;
endproperty
property underflow_0_p;
    @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) ((DUT.count != 0) &&
fifo_if.rd_en) |=> (fifo_if.underflow == 0);
endproperty
assert property (underflow_1_p);
assert property (underflow_0_p);
cover property (underflow_1_p);
cover property (underflow_0_p);
```

```
/**fifo_if.wr_ack**/
property wr_ack_1_p;
    @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (fifo_if.wr_en && (DUT.count
!= FIFO_DEPTH)) |=> fifo_if.wr_ack;
endproperty
property wr_ack_0_p;
    @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (fifo_if.wr_en && (DUT.count
== FIFO_DEPTH)) |=> (fifo_if.wr_ack == 0);
endproperty
assert property (wr_ack_1_p);
assert property (wr_ack_0_p);
cover property (wr_ack_1_p);
cover property (wr_ack_0_p);


/**********************Assertion on Combinational signals******************/

/**fifo_if.full**/
property full_1_p;
    @(posedge fifo_if.clk) (fifo_if.wr_en && DUT.count == FIFO_DEPTH) |->
fifo_if.full;
endproperty
property full_0_p;
    @(posedge fifo_if.clk) (fifo_if.wr_en && DUT.count != FIFO_DEPTH) |->
(fifo_if.full == 0);
endproperty
assert property(full_1_p);
assert property(full_0_p);
cover property(full_1_p);
cover property(full_0_p);

/**almostfifo_design.full**/
property almostfull_1_p;
    @(posedge fifo_if.clk) (DUT.count == (FIFO_DEPTH-1)) |-> fifo_if.almostfull;
endproperty
property almostfull_0_p;
    @(posedge fifo_if.clk) (DUT.count != (FIFO_DEPTH-1)) |-> (fifo_if.almostfull ==
0);
endproperty
assert property(almostfull_1_p);
assert property(almostfull_0_p);
cover property(almostfull_1_p);
cover property(almostfull_0_p);
```

```
/**fifo_if.empty**/
property empty_1_p;
    @(posedge fifo_if.clk) (DUT.count == 0) |-> fifo_if.empty;
endproperty
property empty_0_p;
    @(posedge fifo_if.clk) (DUT.count != 0) |-> (fifo_if.empty == 0);
endproperty
assert property (empty_1_p);
assert property (empty_0_p);
cover property (empty_1_p);
cover property (empty_0_p);

/**almostempty**/
property almostempty_1_p;
    @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (DUT.count == 1) |->
fifo_if.almostempty;
endproperty
property almostempty_0_p;
    @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (DUT.count != 1) |->
(fifo_if.almostempty == 0);
endproperty
assert property (almostempty_1_p);
assert property (almostempty_0_p);
cover property (almostempty_1_p);
cover property (almostempty_0_p);

/*******************Assertion on counters**********************/
property inc_count_p;
    @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (fifo_if.wr_en &&
!fifo_if.rd_en && (DUT.count != FIFO_DEPTH)) |=> (DUT.count == $past(DUT.count) +
1'b1);
endproperty
property dec_count_p;
    @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (!fifo_if.wr_en &&
fifo_if.rd_en && (DUT.count != 0)) |=> (DUT.count == $past(DUT.count) - 1'b1);
endproperty
property same_count_p;
    @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (fifo_if.wr_en &&
fifo_if.rd_en && (DUT.count != 0) && (DUT.count != FIFO_DEPTH)) |=> (DUT.count ==
$past(DUT.count));
endproperty
assert property(inc_count_p);
assert property(dec_count_p);
assert property(same_count_p);
cover property(inc_count_p);
```

```
cover property(dec_count_p);
cover property(same_count_p);


/******************Assertion on pointers*************************/
property wr_ptr_p;
    @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (fifo_if.wr_en && (DUT.count
!= FIFO_DEPTH)) |=> DUT.wr_ptr == $past(DUT.wr_ptr) + 1'b1;
endproperty
property rd_ptr_p;
    @(posedge fifo_if.clk) disable iff(!fifo_if.rst_n) (fifo_if.rd_en && (DUT.count
!= 0)) |=> DUT.rd_ptr == $past(DUT.rd_ptr) + 1'b1;
endproperty

assert property(wr_ptr_p);
assert property(rd_ptr_p);
cover property(wr_ptr_p);
cover property(rd_ptr_p);

always_comb begin
    if(!fifo_if.rst_n)begin
        rst_co_assert:assert final (DUT.count == 0);
        rst_wr_assert:assert final (DUT.wr_ptr == 0);
        rst_rd_assert:assert final (DUT.rd_ptr == 0);
        // rst_full_assert        :assert final (fifo_if.full        == 0);
        // rst_empty_assert       :assert final (fifo_if.empty       == 1);
        // rst_almostfull_assert  :assert final (fifo_if.almostfull  == 0);
        // rst_almostempty_assert :assert final (fifo_if.almostempty == 0);
        // rst_overflow_assert    :assert final (fifo_if.overflow    == 0);
        // rst_underflow_assert   :assert final (fifo_if.underflow   == 0);
        // rst_wr_ack_assert      :assert final (fifo_if.wr_ack      == 0);
    end
end


endmodule
```