RTL Design Document for Module 2

Table of Contents

Revision History

Version	Date (dd/mm/yy)	Description	Author (s)	Reviewer (s)
1	22/02/2024		Sampath	

Objective

This document consists of the design approach of all the assignments in module 2.

Assignment -1 : 8 bit Register with AXI Stream interface

1. Introduction

This axi_reg module is used to minimize Propagation delay caused by Combinational Path. Thus, input is registered and then given as output which has 1 clock cycle delay. We can also use registers to get more than 1 cycle delay.

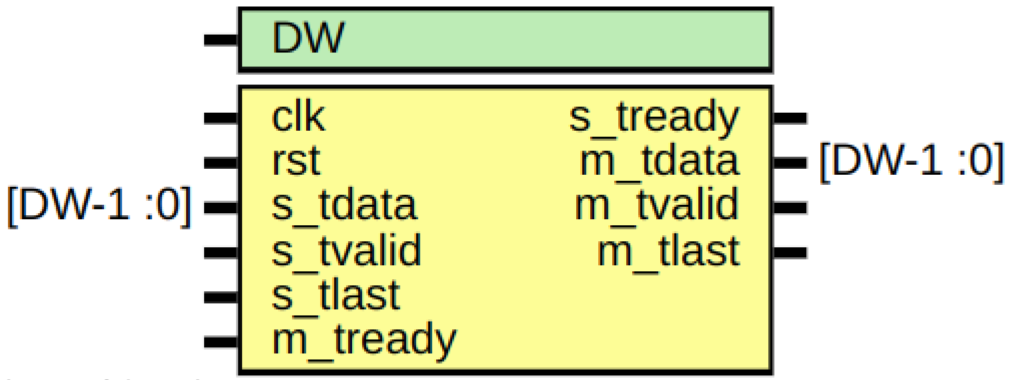
2. Functional Description

The axi_reg module has a Parameter (DW) which tells input and output data widths and uses AXI Streaming Interface.

The AXI Streaming ports used in this module are **Data, Valid, Ready and Last.**

Connect input ready to output ready by registering because whenever output is ready to accept data input will also process some data and output will be seen in the next clock cycle because it has been registered. Therefore, the module has 1 cycle latency.

3. Module Description



3.1 Port Description

Port name	Direction	Type	Description
clk	input	wire	Clock Signal.
rst	input	wire	active high reset.
s_tdata	input	[DW-1:0] wire	Input data bus carrying 8-bit wide data.
s_tvalid	input	wire	Input valid signal which tells the availability of valid input data.
s_tlast	input	wire	Input last signal which tells the end of an input data packet.
s_tready	output	wire	Output ready signal which tells that module is ready to accept incoming data.

m_tdata	output	[DW-1:0] wire	Output data bus carrying 8-bit wide data.
m_tvalid	output	wire	Output valid signal indicating the availability of valid output data.
m_tlast	output	wire	Output last signal which tells the end of an input data packet.
m_tready	input	wire	Input ready signal indicates that another module is ready to accept data.

3.2 Module Code

https://github.com/ksgovardhan/training/blob/2430e05b9d58088db2e2126ae803abd476e68493/Module_2/A1/axi_reg.v

3.3 Implementation Details

- ➢ Whenever m_tready is high module will start processing data
- ➢ Whenever s_valid and s_ready is high data is being stored into a register m_tdata_i.
- ➢ In the next cycle stored value is given out to m_tdata.

4. Testing Procedure

4.1 Testbench code

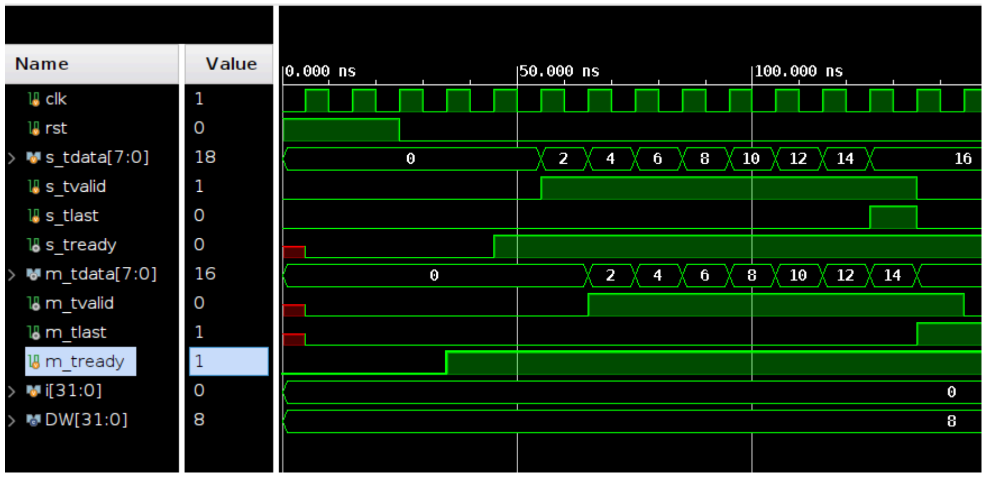
https://github.com/ksgovardhan/training/blob/2430e05b9d58088db2e2126ae803abd476e68493/Module_2/A1/tb_axi_reg.v

4.2 Test Cases

4.2.1 Test 1:

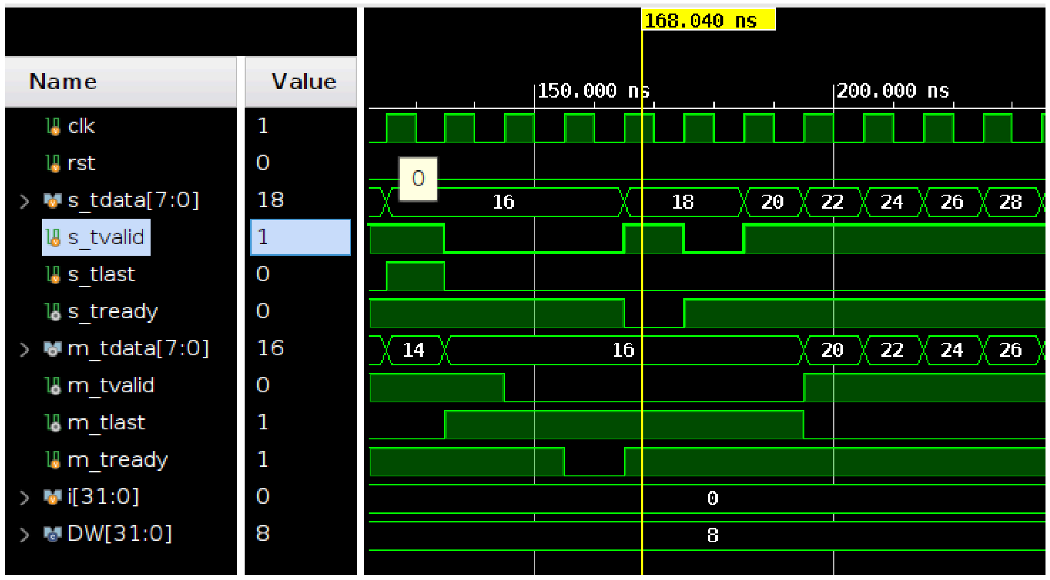
When reset is asserted no input and no output transfer takes place.

Provide valid input data (input_tdata) with appropriate valid signal and last signal.

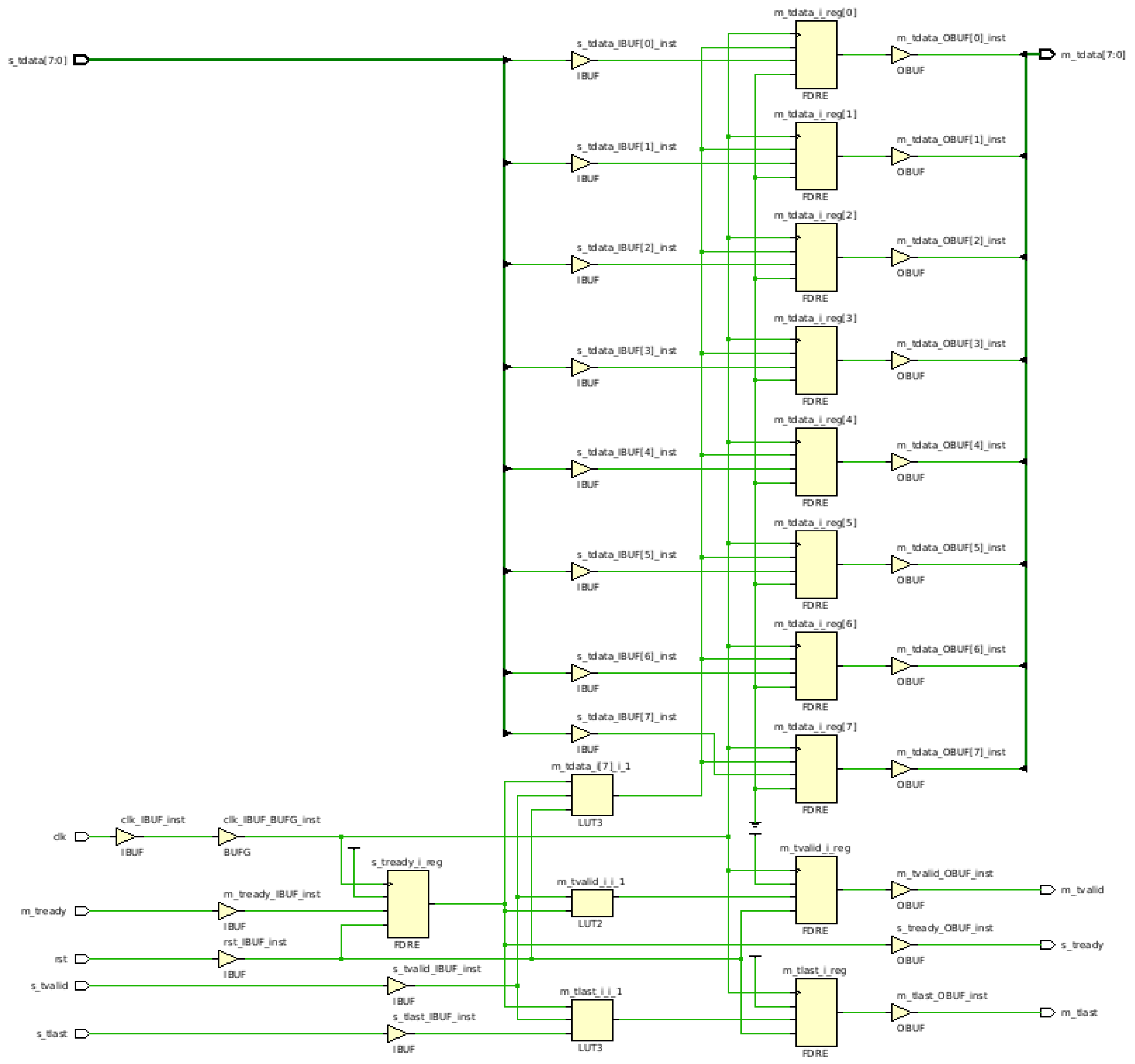


4.2.2 Test 2:

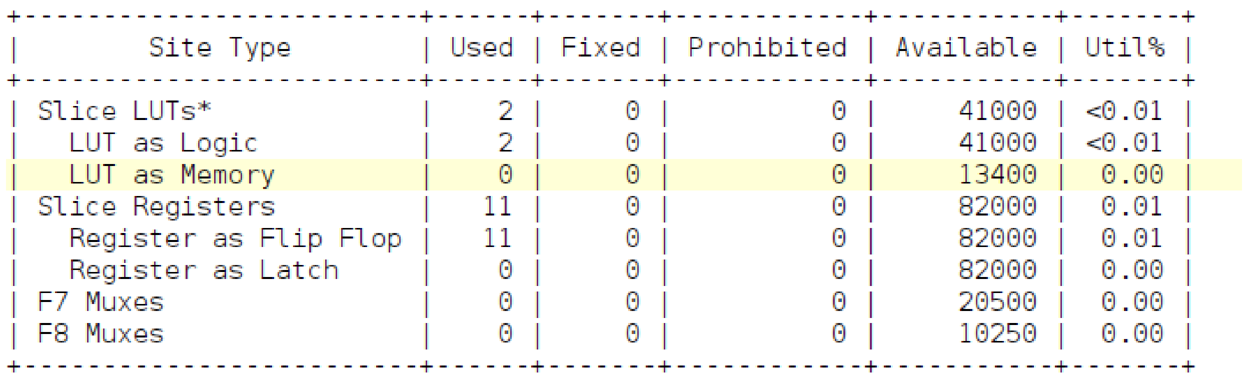
What will be happening when we give valid data when input is not ready.



5. Schematic diagram



6. Resource Utilization



```
+-------------------------+------+-------+------------+-----------+-------+
|        Site Type        | Used | Fixed | Prohibited | Available | Util% |
+-------------------------+------+-------+------------+-----------+-------+
| Slice LUTs*             |    2 |     0 |          0 |     41000 | <0.01 |
|   LUT as Logic          |    2 |     0 |          0 |     41000 | <0.01 |
|   LUT as Memory         |    0 |     0 |          0 |     13400 |  0.00 |
| Slice Registers         |   11 |     0 |          0 |     82000 |  0.01 |
|   Register as Flip Flop |   11 |     0 |          0 |     82000 |  0.01 |
|   Register as Latch     |    0 |     0 |          0 |     82000 |  0.00 |
| F7 Muxes                |    0 |     0 |          0 |     20500 |  0.00 |
| F8 Muxes                |    0 |     0 |          0 |     10250 |  0.00 |
+-------------------------+------+-------+------------+-----------+-------+
```

Assignment - 2: 2x1 Mux with AXI stream

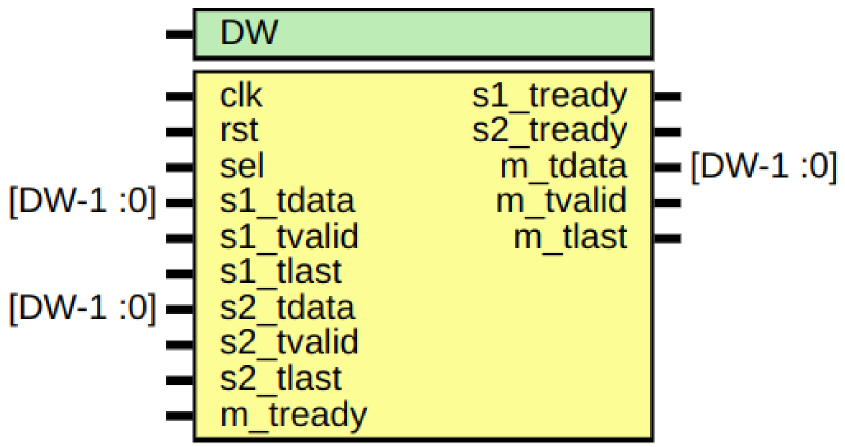
1. Introduction

This mux_2_1 module is designed to function as a 2x1 multiplexer (MUX) with AXI Stream interface. It selects between two input data streams based on a select signal (sel) and sends corresponding stream data to the output port.

2. Functional Description

- ➢ The mux_2_1 module has a Parameter (DW) which tells input and output data widths and uses AXI Streaming Interface.
- ➢ The AXI Streaming ports used in this module are **Data, Valid, Ready and Last.**
- ➢ I registered input data and sent it to output based on sel signal (*sel?s1:s2*).
- ➢ Connect m_tready to s1_tready and s2_tready by registering because whenever output is ready to accept data input will also process some data based on sel signal.

3. Module Description



WiSig Networks

3.1 Port Description

Port name	Direction	Bit Width	Description
clk	input		Clock signal input
rst	input		Active high reset signal
sel	input		Control signal to select between input data streams
s1_tdata	input	[DW-1:0]	Input s1 data bus carrying 8-bit wide data.
s1_tvalid	input		Input s1 valid signal which tells the availability of valid input data.
s1_tlast	input		Input s1 last signal which tells the end of an input data packet.
s1_tready	output		Output s1 ready signal which tells that module is ready to accept incoming data.

s2_tdata	input	[DW-1:0]	Input s2 data bus carrying 8-bit wide data.
s2_tvalid	input		Input s2 valid signal which tells the availability of valid input data.
s2_tlast	input		Input s2 last signal which tells the end of an input data packet.
s2_tready	output		Output s2 ready signal which tells that module is ready to accept incoming data.
m_tdata	output	[DW-1:0]	Output data bus carrying 8-bit wide data.
m_tvalid	output		Output valid signal indicating the availability of valid output data.
m_tlast	output		Output last signal which tells the end of an input data packet.

m_tready	input		Input ready signal indicates that another module is ready to accept data.

3.2 Module Code

https://github.com/ksgovardhan/training/blob/b295792add3d3c4a66d49ceaabd176c036adafbf/Module_2/A2/mux_2_1.sv

3.3 Implementation Details

- Based on m_tready signal s1_tready or s2_tready are selected using sel signal.
- Code is written in a way that If logic is not true output data holds on to its previous state and if reset it is zero.

4. Testing Procedure

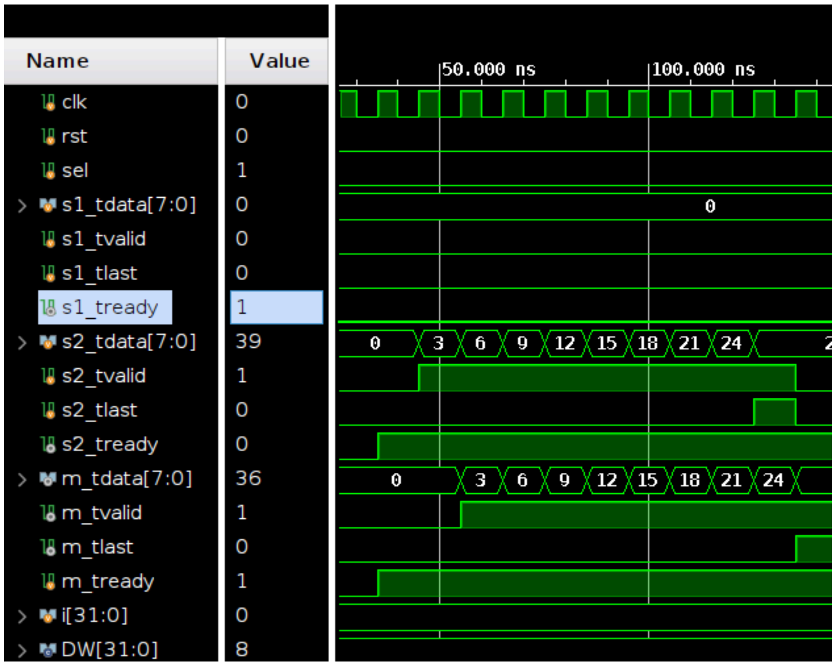
4.1 Testbench code

https://github.com/ksgovardhan/training/blob/b295792add3d3c4a66d49ceaabd176c036adafbf/Module_2/A2/tb_mux_2_1.sv

4.2 Test Cases

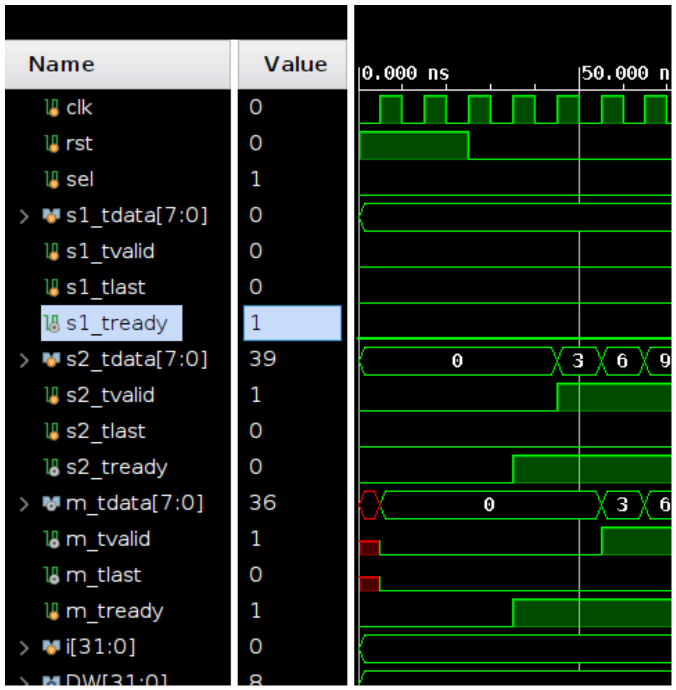
4.2.1 Test 1:

Provide valid input data (input_tdata) with appropriate valid signal and last signal with sel signal.



4.2.2 Test 2:

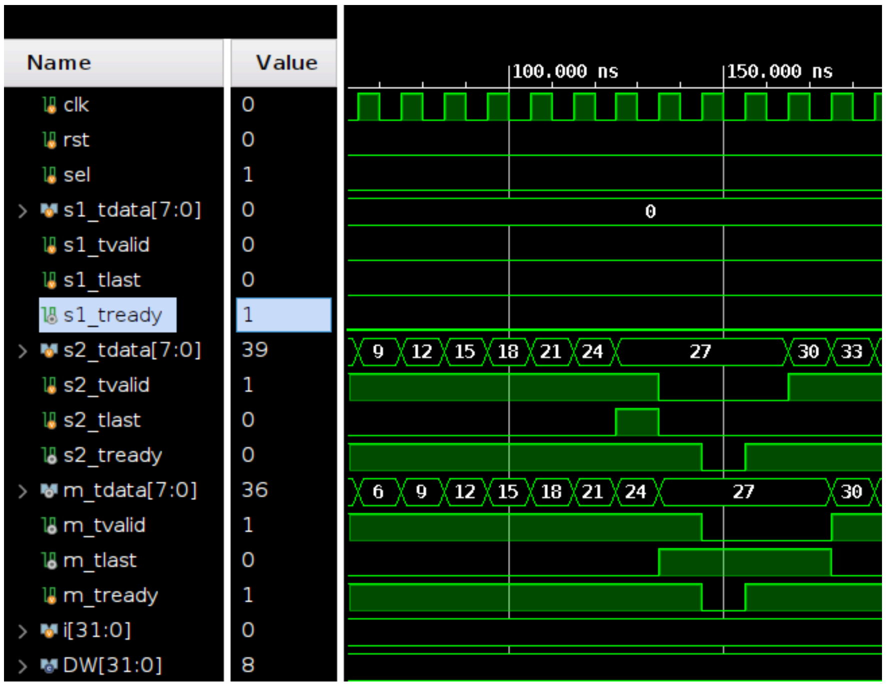
When reset is asserted no input and no output transfer takes place?



Name	Value
clk	0
rst	0
sel	1
s1_tdata[7:0]	0
s1_tvalid	0
s1_tlast	0
s1_tready	1
s2_tdata[7:0]	39
s2_tvalid	1
s2_tlast	0
s2_tready	0
m_tdata[7:0]	36
m_tvalid	1
m_tlast	0
m_tready	1
i[31:0]	0
DW[31:0]	8

4.2.3 Test 3:

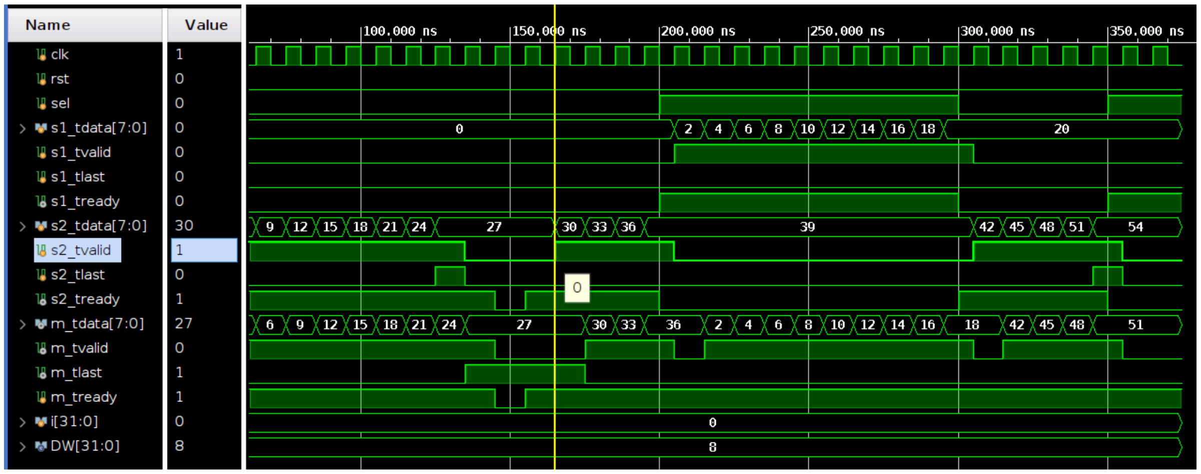
What will happen if the m_tready is low?



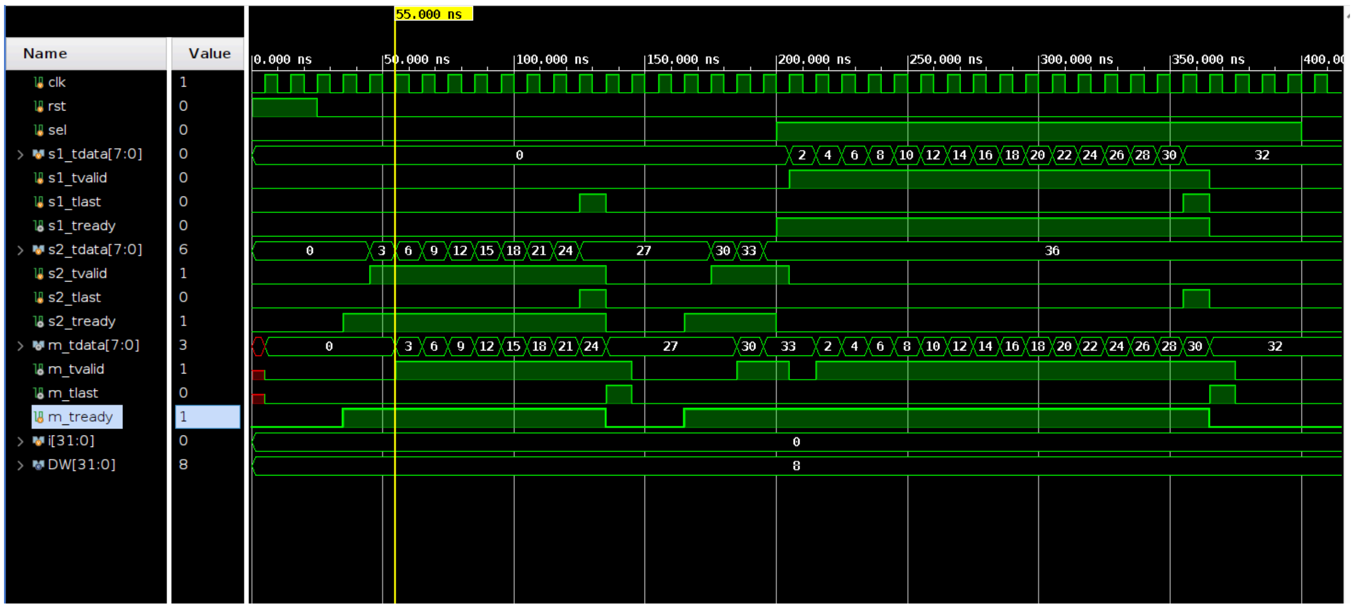
Name	Value
clk	0
rst	0
sel	1
s1_tdata[7:0]	0
s1_tvalid	0
s1_tlast	0
s1_tready	1
s2_tdata[7:0]	39
s2_tvalid	1
s2_tlast	0
s2_tready	0
m_tdata[7:0]	36
m_tvalid	1
m_tlast	0
m_tready	1
i[31:0]	0
DW[31:0]	8

4.2.4 Test 4:

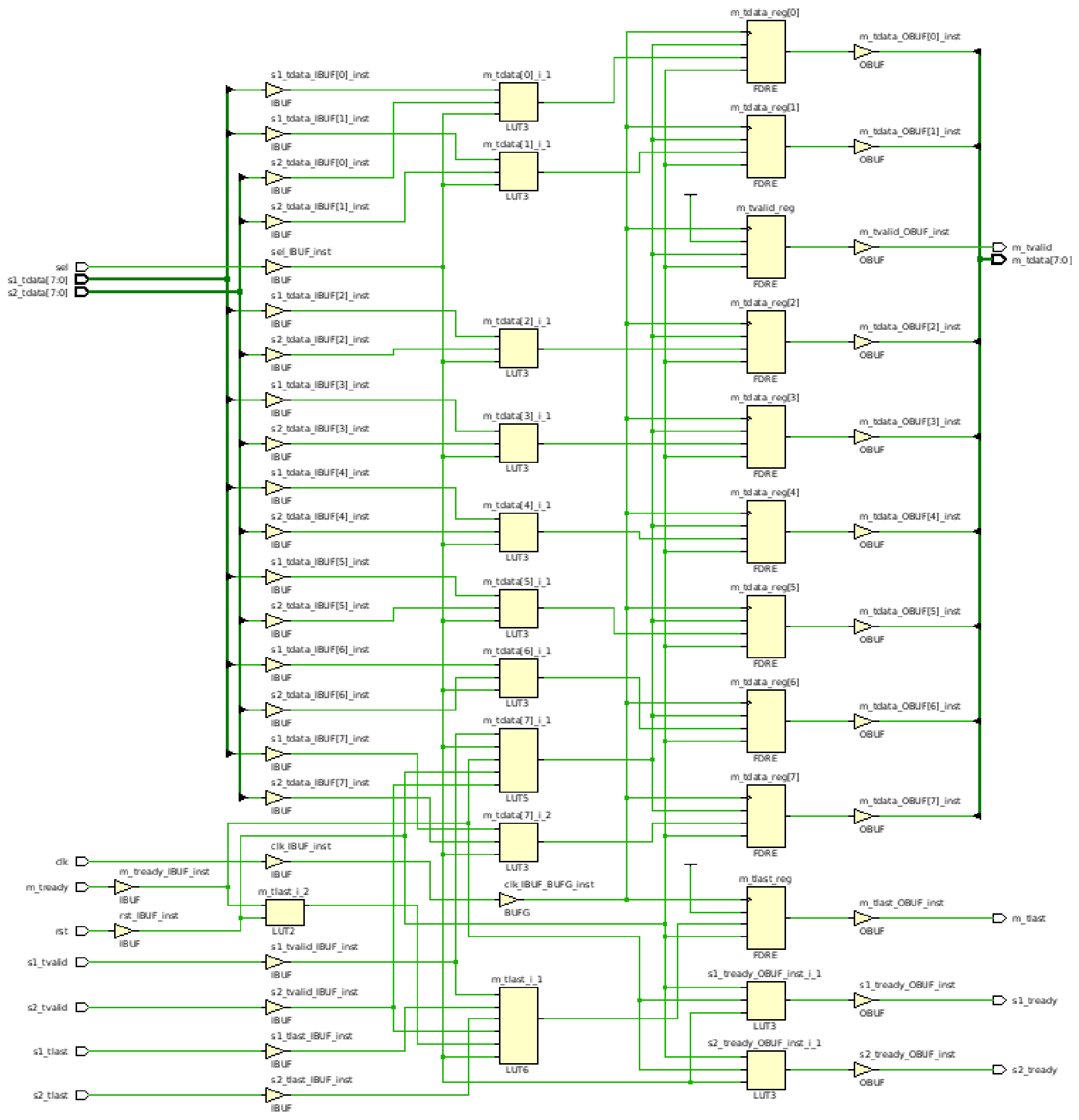
Does the module read correct inputs based on sel signal?



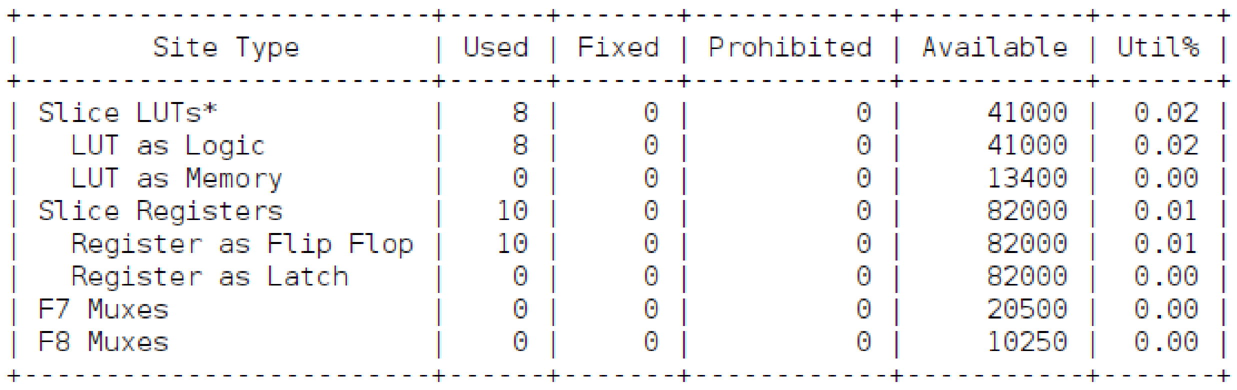
Name	Value
clk	1
rst	0
sel	0
s1_tdata[7:0]	0
s1_tvalid	0
s1_tlast	0
s1_tready	0
s2_tdata[7:0]	30
s2_tvalid	1
s2_tlast	0
s2_tready	1
m_tdata[7:0]	27
m_tvalid	0
m_tlast	1
m_tready	1
i[31:0]	0
DW[31:0]	8


55.000 ns
0.000 ns
50.000 ns
100.000 ns
150.000 ns
200.000 ns
250.000 ns
300.000 ns
350.000 ns
400.00
Name
Value
clk 1
rst 0
sel 0
s1_tdata[7:0] 0
s1_tvalid 0
s1_tlast 0
s1_tready 0
s2_tdata[7:0] 6
s2_tvalid 1
s2_tlast 0
s2_tready 1
m_tdata[7:0] 3
m_tvalid 1
m_tlast 0
m_tready 1
i[31:0] 0
DW[31:0] 8

5. Schematic diagram



6. Resource Utilization



```
+-------------------------+------+-------+------------+-----------+-------+
|        Site Type        | Used | Fixed | Prohibited | Available | Util% |
+-------------------------+------+-------+------------+-----------+-------+
| Slice LUTs*             |    8 |     0 |          0 |     41000 |  0.02 |
|   LUT as Logic          |    8 |     0 |          0 |     41000 |  0.02 |
|   LUT as Memory         |    0 |     0 |          0 |     13400 |  0.00 |
| Slice Registers         |   10 |     0 |          0 |     82000 |  0.01 |
|   Register as Flip Flop |   10 |     0 |          0 |     82000 |  0.01 |
|   Register as Latch     |    0 |     0 |          0 |     82000 |  0.00 |
| F7 Muxes                |    0 |     0 |          0 |     20500 |  0.00 |
| F8 Muxes                |    0 |     0 |          0 |     10250 |  0.00 |
+-------------------------+------+-------+------------+-----------+-------+
```

Assignment - 3: 4096 FIFO with 2 2048 FIFOs using AXI interface

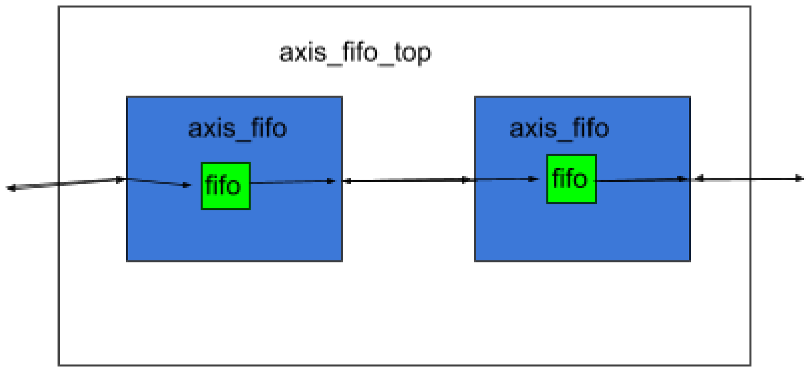
1. Introduction

This module is designed to store and retrieve data in fifo of depth 4096 using 2 2048 fifo modules which are cascaded. It provides functionality for both writing data into the FIFO and reading data from fifo simultaneously.

2. Functional Description

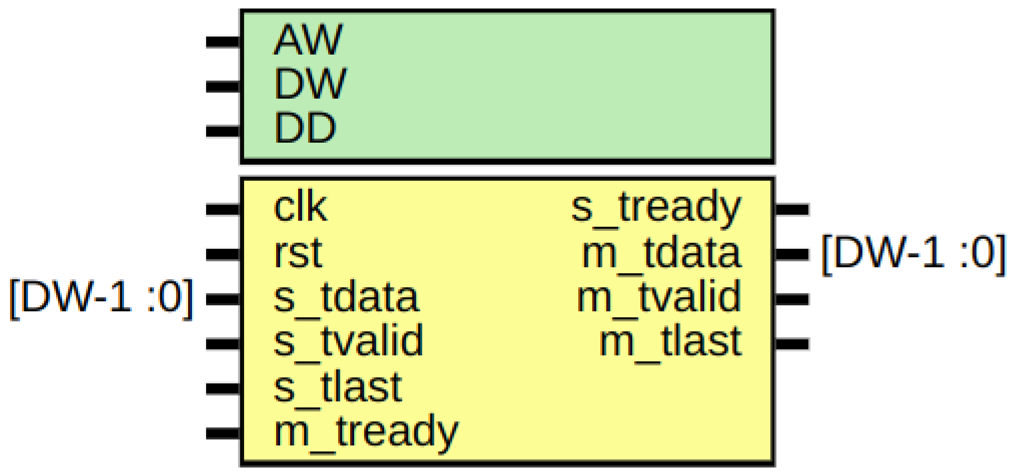
- Both axis_fifo modules work on streaming data and fifo1 and fifo2 modules do not support streaming.
- Input and output data is 16 bits and is represented by a parameter DW.
- Depth is represented by a parameter DD which can support different depths.
- Both fifo1 and fifo2 are 2048 in depth and data width is 17 and I/O are given from the axis_fifo module.
- AW is the address width of fifo which is parameterized but can be derived from DD.
- **Note:** The full and empty signals used in this design are having 1 clock cycle delay of actual full and empty signals. Actual signals are found in the fifo module (full is fill and empty is flag).

3. Block Diagram



4. Module Description

4.1 axis_fifo_top



4.1.1 Port Description

Port name	Direction	Bit Width	Description
clk	input		Clock Signal.
rst	input		active high reset.

s_tdata	input	[DW-1:0]	Input data bus carrying 8-bit wide data.
s_tvalid	input		Input valid signal which tells the availability of valid input data.
s_tlast	input		Input last signal which tells the end of an input data packet.
s_tready	output		Output ready signal which tells that module is ready to accept incoming data.
m_tdata	output	[DW-1:0]	Output data bus carrying 8-bit wide data.
m_tvalid	output		Output valid signal indicating the availability of valid output data.
m_tlast	output		Output last signal which tells the end of an input data packet.

m_tready	input		Input ready signal indicates that another module is ready to accept data.

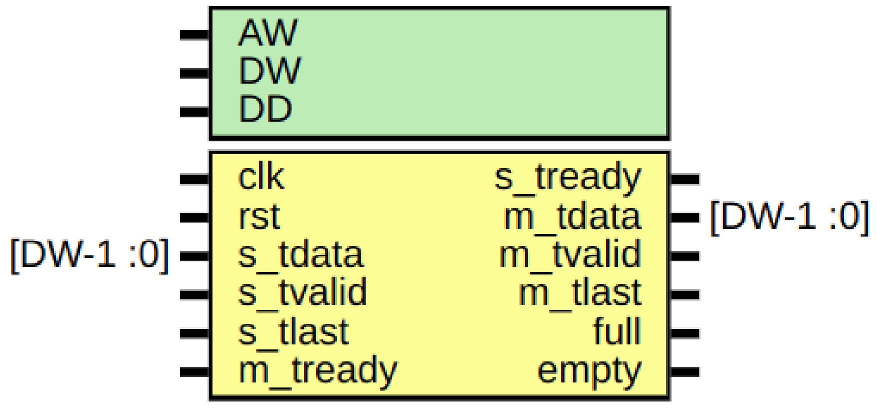
4.1.2 Module Code

https://github.com/ksgovardhan/training/blob/b295792add3d3c4a66d49ceaabd176c036adafbf/Module_2/A3/axis_fifo_top.sv

4.1.3 Implementation Details

➢ axis_fifo_top is the top module (4096 depth fifo) which instantiates two axi_fifo (2048 depth fifo) which are then cascaded.
➢ All the slave ports are connected to axis_fifo1 and Master ports are connected to axis_fifo2.
➢ Based on Input Stream data, data is stored into fifo1 based on input ready signal.
➢ Output ready signal is given to fifo2 so that it will send the values stored in fifo2 first and then fetch from fifo1, else if it is empty it will fetch the values from fifo1 accordingly.

4.2 axis_fifo



4.2.1 Port Description

Port name	Direction	Type	Description
clk	input		Clock Signal.
rst	input		active high reset.
s_tdata	input	[DW-1:0]	Input data bus carrying 8-bit wide data.

s_tvalid	input		Input valid signal which tells the availability of valid input data.
s_tlast	input		Input last signal which tells the end of an input data packet.
s_tready	output		Output ready signal which tells that module is ready to accept incoming data.
m_tdata	output	[DW-1:0]	Output data bus carrying 8-bit wide data.
m_tvalid	output		Output valid signal indicating the availability of valid output data.
m_tlast	output		Output last signal which tells the end of an input data packet.
m_tready	input		Input ready signal indicates that another module is ready to accept data.

full	output		It is a reference signal which has 1 cycle delay of full signal in fifo and is used for streaming logic reference
empty	output		It is a reference signal which has 1 cycle delay of full signal in fifo and is used for streaming logic reference

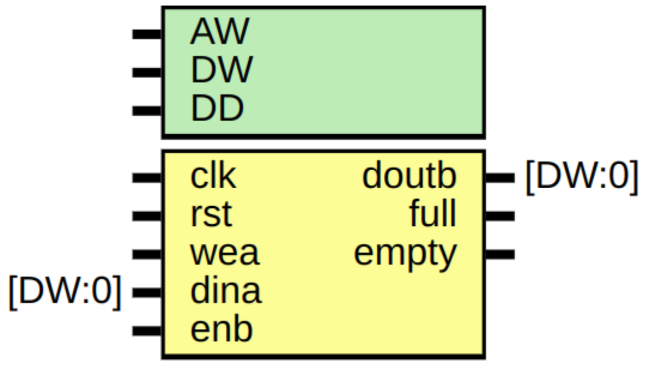
4.2.2 Module Code

https://github.com/ksgovardhan/training/blob/b295792add3d3c4a66d49ceaabd176c036adafbf/Module_2/A3/axis_fifo.sv

4.2.3 Implementation Details

- axis_fifo module handles the streaming data and generates control signals to read/write data from/to fifo. And then these control signals are passed to the fifo module.
- **Input data and last signals are packed inside the axis_fifo and stored in fifo.** So that we can know that data is the last value in a packet when I am reading from fifo.
- Input ready signal is high when fifo is not full and input data is written to fifo until fifo is not full.
- Similarly, data is read from fifo until fifo is empty based on output ready signal.

4.3 fifo



4.3.1 Port Description

Port name	Direction	Type	Description
clk	input		Clock signal.
rst	input		active high reset.
wea	input		When asserted, data is written into the FIFO. It comes from axis_fifo.

enb	input		When asserted, data is read from the FIFO. It also comes from axis_fifo
dina	input	[DW-1:0]	Input data to be written into the FIFO.
doutb	output	[DW-1:0]	Output data read from the FIFO.
full	output		tells the status of fifo with 1 clock cycle delay.
empty	output		tells the status of fifo with 1 clock cycle delay.

4.3.2 Module Code

https://github.com/ksgovardhan/training/blob/b295792add3d3c4a66d49ceaabd176c036adafbf/Module_2/A3/fifo.v

4.3.3 Implementation Details

- ➢ This fifo module does not support streaming data; it takes only control signals from the axis_fifo module and stores/reads data accordingly by generating some address pointers internally.
- ➢ This is the BRAM component which is visible after Synthesis in netlist.

5. Testing Procedure

- ➢ Testbench is a Self Checking Testbench which takes input data along with a last signal from this file and writes the self Checking Output to this file by comparing output with this file .

5.1 Testbench Code

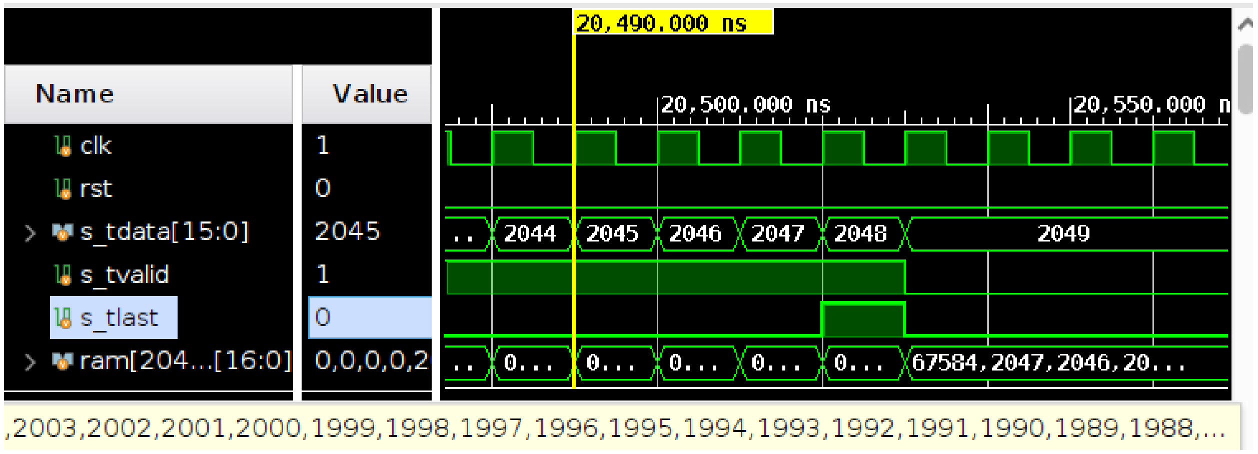
tb_axis_fifo_top:
https://github.com/ksgovardhan/training/blob/b295792add3d3c4a66d49ceaabd176c036adafbf/Module_2/A3/tb_axis_fifo_top.sv

tb_axis_fifo :
https://github.com/ksgovardhan/training/blob/b295792add3d3c4a66d49ceaabd176c036adafbf/Module_2/A3/tb_axis_fifo.sv

5.2.1 Test Cases for axi_fifo module

5.2.1.1 Test 1

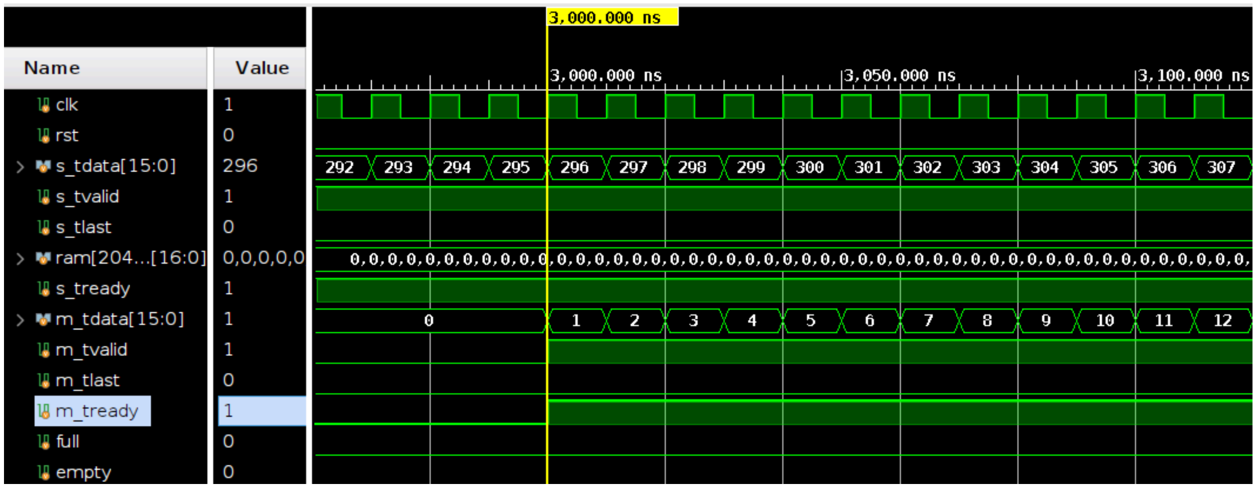
Provide valid Input and Check whether it is stored in the fifo.



,2003,2002,2001,2000,1999,1998,1997,1996,1995,1994,1993,1992,1991,1990,1989,1988,...

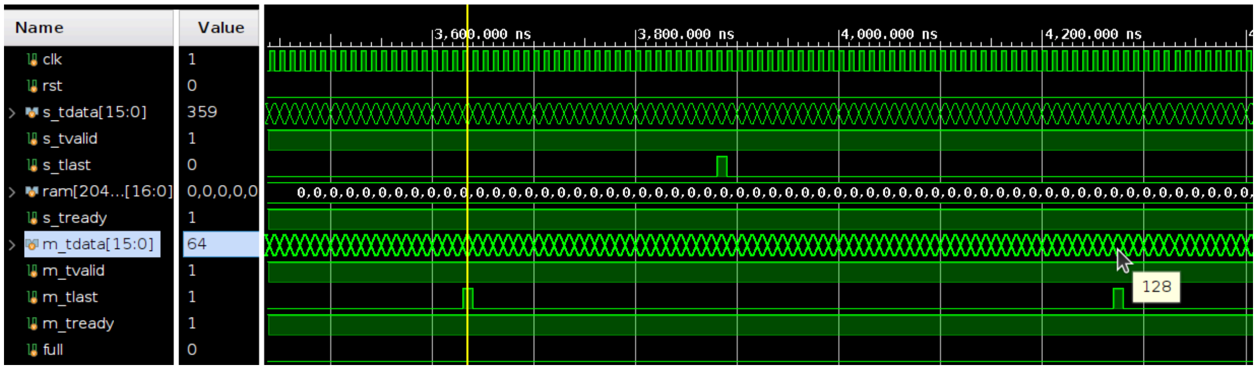
5.2.1.2 Test 2

Does data capture correctly when we are reading from fifo?



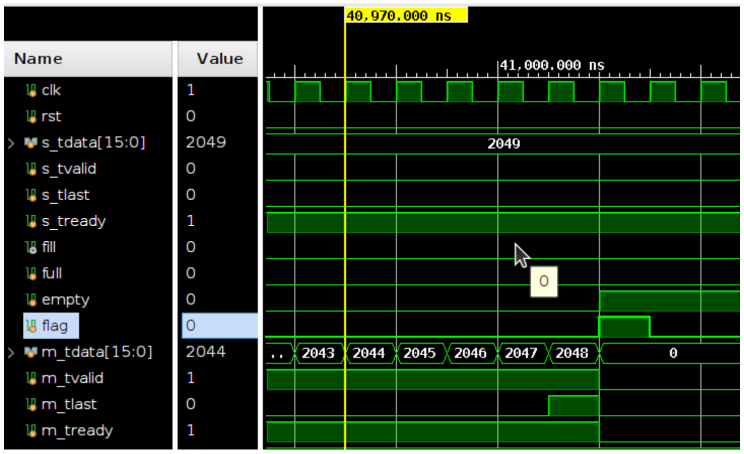
5.2.1.3 Test 3

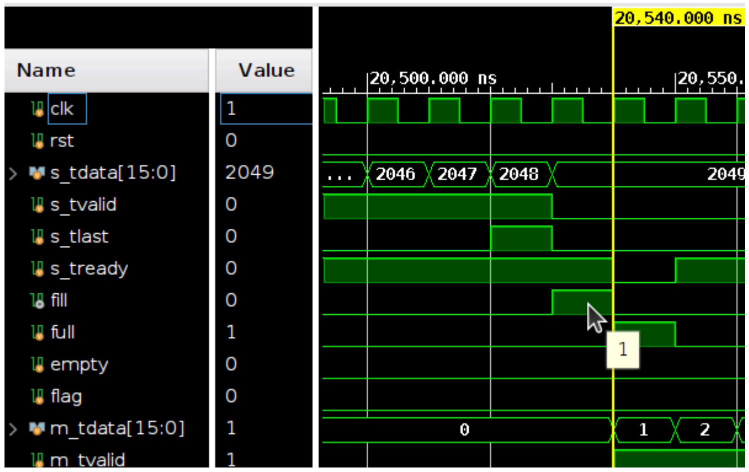
Are we getting vaid last signals when we are reading data from fifo?



5.2.1.4 Test 4

Are we getting valid empty and full signals?

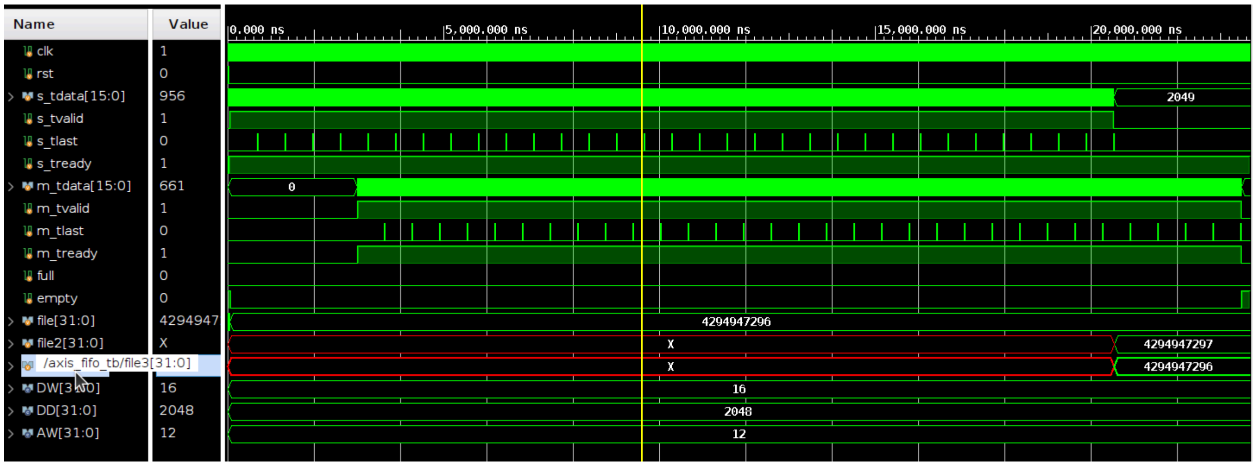




5.2.1.5 Test 5

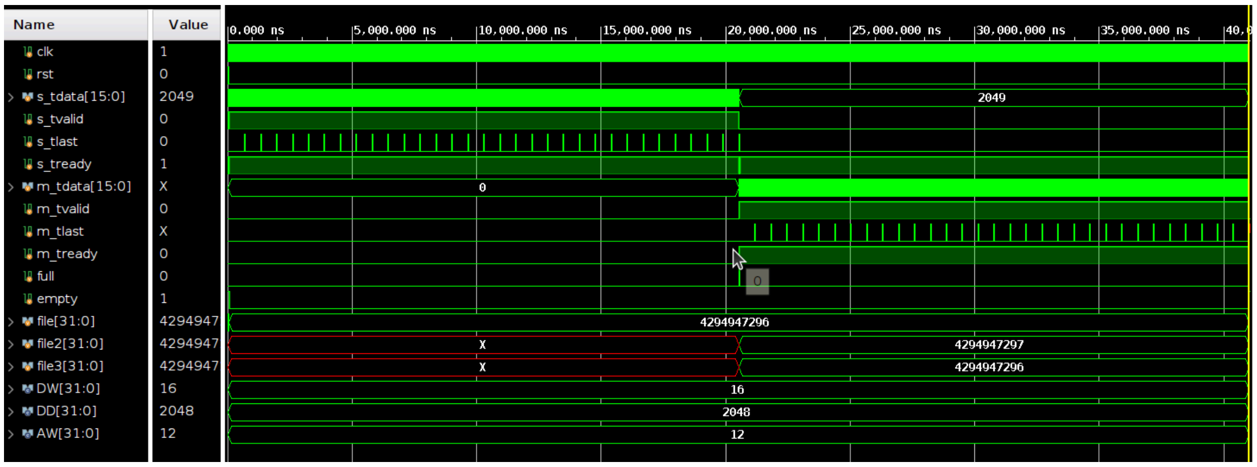
Top level view:

1. Making output ready high when data is been written into fifo



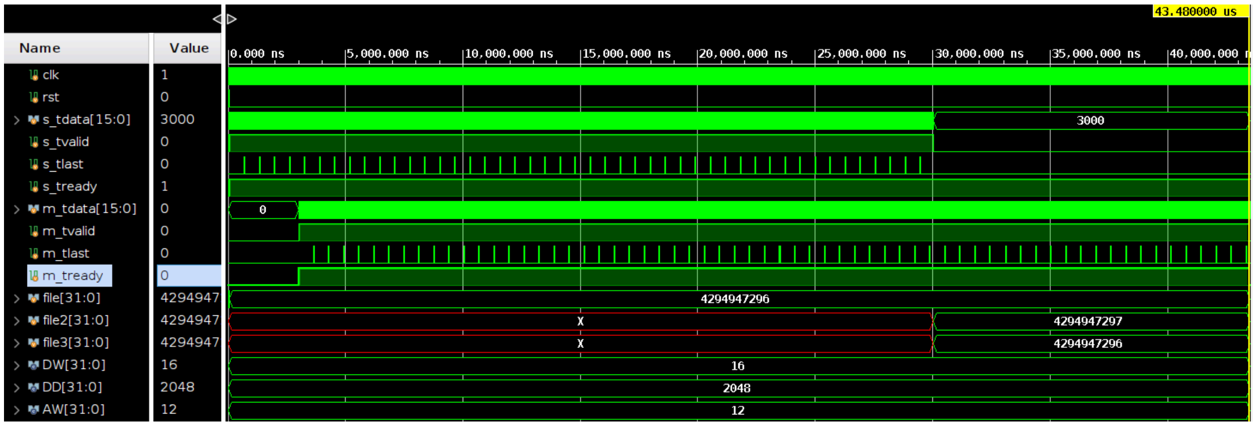
Name	Value
clk	1
rst	0
s_tdata[15:0]	956
s_tvalid	1
s_tlast	0
s_tready	1
m_tdata[15:0]	661
m_tvalid	1
m_tlast	0
m_tready	1
full	0
empty	0
file[31:0]	4294947
file2[31:0]	X
/axis_fifo_tb/file3[31:0]	
DW[31:0]	16
DD[31:0]	2048
AW[31:0]	12

2. Making Output ready high when fifo is full (this case is used to see the status of full signal)



Name	Value
clk	1
rst	0
s_tdata[15:0]	2049
s_tvalid	0
s_tlast	0
s_tready	1
m_tdata[15:0]	X
m_tvalid	0
m_tlast	X
m_tready	0
full	0
empty	1
file[31:0]	4294947
file2[31:0]	4294947
file3[31:0]	4294947
DW[31:0]	16
DD[31:0]	2048
AW[31:0]	12

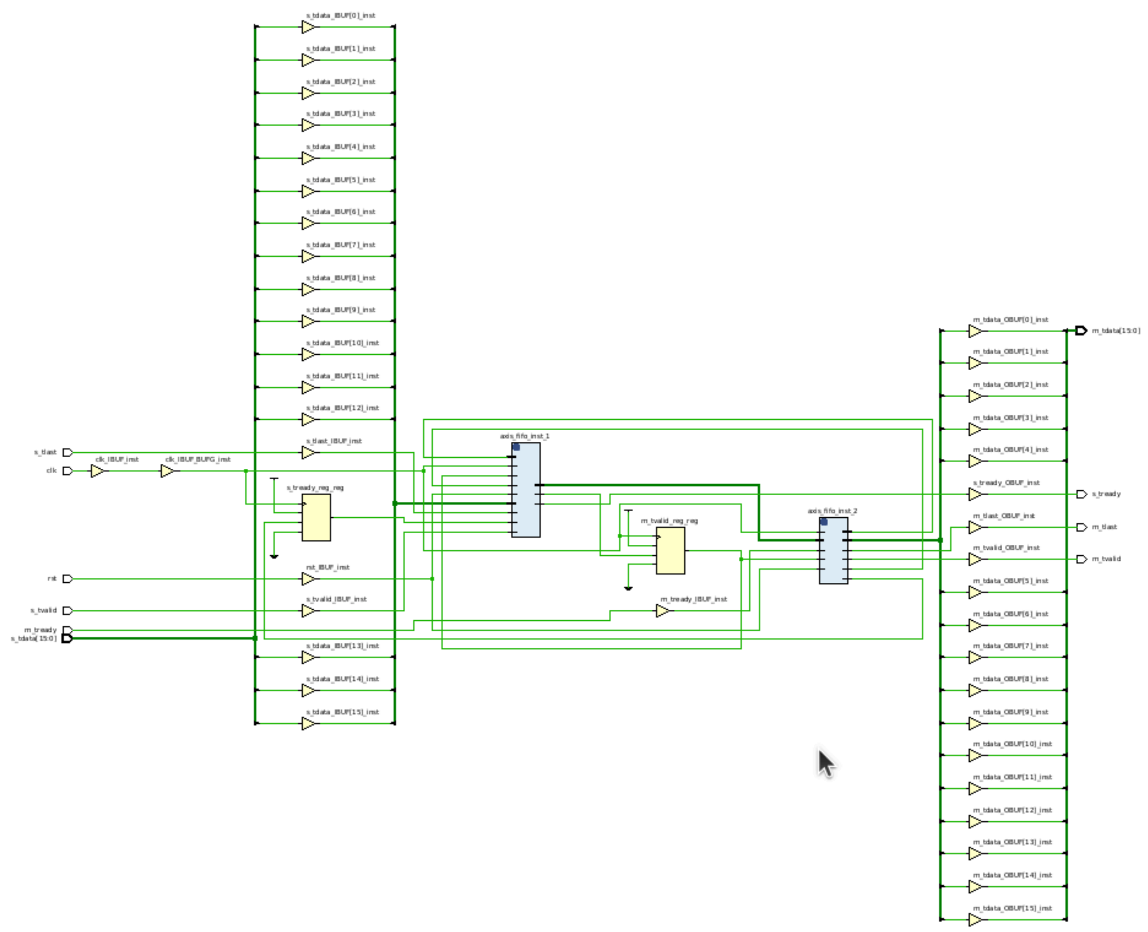
5.2.2 Test Cases for top module axi_fifo_top



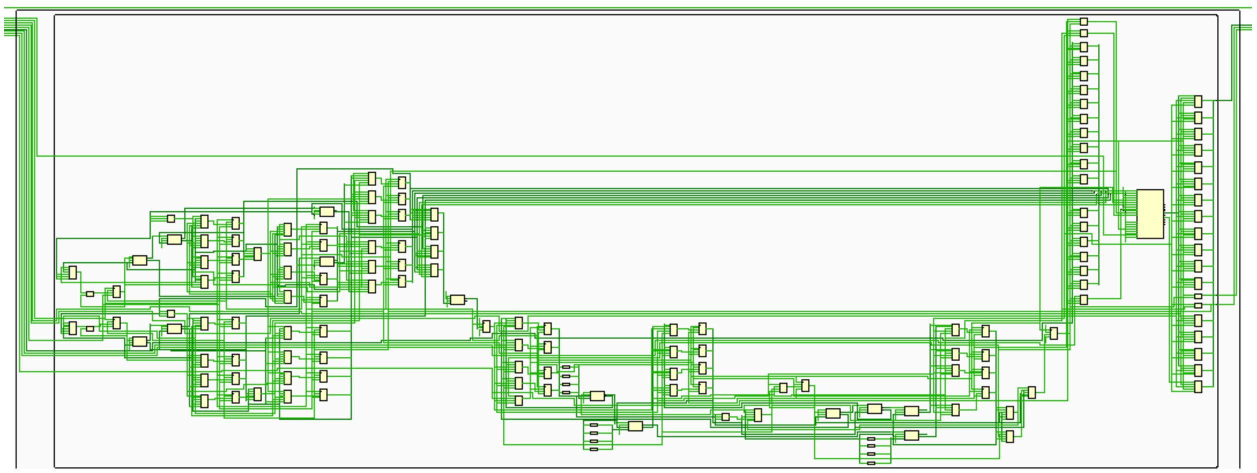
Name	Value
clk	1
rst	0
s_tdata[15:0]	3000
s_tvalid	0
s_tlast	0
s_tready	1
m_tdata[15:0]	0
m_tvalid	0
m_tlast	0
m_tready	0
file[31:0]	4294947
file2[31:0]	4294947
file3[31:0]	4294947
DW[31:0]	16
DD[31:0]	2048
AW[31:0]	12

6. Schematic diagram

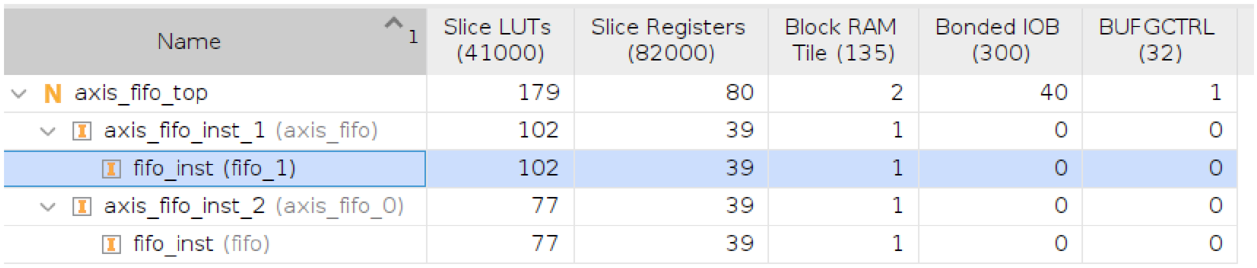
6.1 axis_fifo_top



6.2 axis_fifo and fifo (fifo is instantiated as BRAM Block)

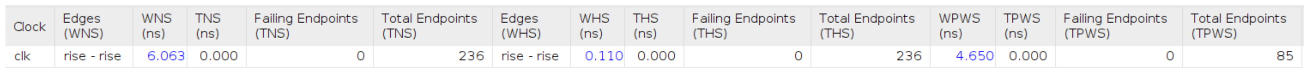


7. Resource Utilization



Name	Slice LUTs (41000)	Slice Registers (82000)	Block RAM Tile (135)	Bonded IOB (300)	BUFGCTRL (32)
axis_fifo_top	179	80	2	40	1
axis_fifo_inst_1 (axis_fifo)	102	39	1	0	0
fifo_inst (fifo_1)	102	39	1	0	0
axis_fifo_inst_2 (axis_fifo_0)	77	39	1	0	0
fifo_inst (fifo)	77	39	1	0	0

8. Timing and Fmax



Clock	Edges (WNS)	WNS (ns)	TNS (ns)	Failing Endpoints (TNS)	Total Endpoints (TNS)	Edges (WHS)	WHS (ns)	THS (ns)	Failing Endpoints (THS)	Total Endpoints (THS)	WPWS (ns)	TPWS (ns)	Failing Endpoints (TPWS)	Total Endpoints (TPWS)
clk	rise - rise	6.063	0.000	0	236	rise - rise	0.110	0.000	0	236	4.650	0.000	0	85

Fmax= 1/(10-6.063) = 254 MHz