

A REPORT
ON
UNIVERSAL ASYNCHRONOUS RECEIVER AND TRANSMITTER,
DUAL PORT RANDOM ACCESS MEMORY,
SYNCHRONOUS FIFO

By

Names of the students

Enrolment No.

HARSHAVARDHAN SAI MACHINENI
SREENIVASULU TANGUTURI
HARSHITH NIMMAGADDA

1700315C204
1700359C204
1700316C204

Prepared in the partial fulfillment of the
Practice School II Course

AT

SION SEMICONDUCTORS PVT LTD, BANGALORE

A Practice School II Station of



BML MUNJAL UNIVERSITY

(May-July 2019)

TABLE OF CONTENTS

PROJECT 1: UNIVERSAL ASYNCHRONOUS RECEIVER AND TRANSMITTER

1. INTRODUCTION.....	5
1.1. History.....	5
1.2. Problem statement and objective.....	5
1.3. Motivation.....	6
1.4. Method of tackling problem.....	6
1.5. Requirements.....	7
1.6. Assumptions.....	7
1.7. Related work (Literature review)	8
2. COMPANY PROFILE.....	9
2.1. company.....	9
2.2. Technology.....	9
2.3. Products.....	10
2.4. Training programs.....	10
2.5. Company Affiliations.....	11
3. PROJECT METHODOLOGY.....	12
3.1. Introduction.....	12
3.2. Overview on UART.....	13
3.3. Black box view of Transmitter.....	14
3.4. Working of Transmitter.....	15
3.5. Black box view of Receiver.....	16
3.6. Working of Receiver.....	17
4. RESULT AND DISCUSSION.....	18
4.1. Simulations of Transmitter and Receiver.....	18
4.2. RTL schematics.....	25
4.2.1. UART Top Module.....	25
4.2.2. UART Transmitter.....	25
4.2.3. UART Receiver.....	26
4.3. Synthesis Report.....	27
4.4. Post place and Route simulation.....	37
CONCLUSION AND RECOMMENDATIONS.....	38
APPENDIX 1: LIST OF FIGURES AND TABLES.....	39
APPENDIX 2: REFERENCES.....	40

PROJECT 2: DUAL PORT RANDOM ACCESS MEMORY

1. INTRODUCTION.....	44
1.1. History.....	44
1.2. Problem statement and objective.....	44
1.3. Motivation.....	44
1.4. Method of tackling problem.....	45
1.5. Requirements.....	46
1.6. Assumptions.....	46
2. PROJECT METHODOLOGY.....	46
2.1. Introduction.....	46
2.2. Write and read operations triggering.....	47
2.3. Port width configurations.....	49
2.4. Mixed-width port configuration.....	49
2.5. Maximum Block Depth Configuration.....	49
2.6. Clocking modes.....	50
2.7. Error code correction and freeze logic.....	50
2.8. DPRAM black box view.....	51
3. RESULT AND DISCUSSION.....	52
3.1. Simulations.....	52
3.2. Verilog code for DPRAM design.....	54
3.3. RTL schematics.....	55
3.4. Synthesis report.....	55
3.5. Power report.....	61
CONCLUSION AND RECOMMENDATIONS.....	62
..	
APPENDIX 1: LIST OF FIGURES AND TABLES.....	64
APPENDIX 2: REFERENCES.....	64

PROJECT 3: SYNCHRONOUS FIFO

1. INTRODUCTION.....	65
1.1. Problem statement and objective.....	65
1.2. Motivation.....	65
1.3. Method of tackling problem.....	65
1.4. Requirements.....	66
1.5. Assumptions.....	66
1.6. Related work (Literature review)	66
2. PROJECT METHODOLOGY.....	67
2.1. BLACK BOX VIEW OF FIFO.....	67
2.2. WORKING OF FIFO.....	68
3. RESULT AND DISCUSSION.....	69
3.1. Simulation.....	69
3.2. Verilog code.....	73
3.3. RTL schematics.....	75
3.4. Utilization report.....	76
3.5. Synthesis report.....	86
4. CONCLUSION AND RECOMMENDATIONS.....	86
5. APPENDIX 2: REFERNCES.....	87

Certificate of authenticity

CERTIFICATE

This is to certify that Practice School Project of HARSHAVARDHAN SAI MACHINENI titled UNIVERSAL ASYNCHRONOUS RECEIVER AND TRANSMITTER, DUAL PORT RANDOM ACCESS MEMORY AND SYNCHRONOUS FIFO is an original work and that this work has not been submitted anywhere in any form. Indebtedness to other works/publications has been duly acknowledged at relevant places. The project work was carried during 22-05-2019 to 12-07-2019 in SION SEMICONDUCTORS PVT LTD, BANGALORE.

Signature of PS-II faculty	Signature of industry mentor/Supervisor
Name: Dr. M.B. SRINIVAS	Name: MR.C.S. BASHA
Designation: DEAN SOET	Designation: SENIOR DV ENGINEER
<i>(Seal of the organization with Date)</i>	<i>(Seal of the organization with Date)</i>

Certificate of authenticity

CERTIFICATE

This is to certify that Practice School Project of SREENIVASULU TANGUTURI titled UNIVERSAL ASYNCHRONOUS RECEIVER AND TRANSMITTER, DUAL PORT RANDOM ACCESS MEMORY AND SYNCHRONOUS FIFO is an original work and that this work has not been submitted anywhere in any form. Indebtedness to other works/publications has been duly acknowledged at relevant places. The project work was carried during 22-05-2019 to 12-07-2019 in SION SEMICONDUCTORS PVT LTD, BANGALORE.

Signature of PS-II faculty	Signature of industry mentor/Supervisor
Name: Dr. M.B. SRINIVAS	Name: MR.C.S. BASHA
Designation: DEAN SOET	Designation: SENIOR DV ENGINEER
<i>(Seal of the organization with Date)</i>	<i>(Seal of the organization with Date)</i>

Certificate of authenticity

CERTIFICATE

This is to certify that Practice School Project of HARSHITH NIMMAGADDA titled UNIVERSAL ASYNCHRONOUS RECEIVER AND TRANSMITTER, DUAL PORT RANDOM ACCESS MEMORY AND SYNCHRONOUS FIFO is an original work and that this work has not been submitted anywhere in any form. Indebtedness to other works/publications has been duly acknowledged at relevant places. The project work was carried during 22-05-2019 to 12-07-2019 in SION SEMICONDUCTORS PVT LTD, BANGALORE.

Signature of PS-II faculty	Signature of industry mentor/Supervisor
Name: Dr. M.B. SRINIVAS	Name: MR.C.S. BASHA
Designation: DEAN SOET	Designation: SENIOR DV ENGINEER
<i>(Seal of the organization with Date)</i>	<i>(Seal of the organization with Date)</i>

Acknowledgment

It gives me immense pleasure to present the report of the Practice School-2 undertaken during B.Tech. Second Year stationed at SION SEMICONDUCTORS PVT.LTD, Bangalore. I thank **Dr MB Srinivas**, Dean SOET and **Prof. Manoj K. Arora**, Vice Chancellor BMU to present me with an industrial hands-on learning opportunity through PS-2 and I owe special debt of gratitude for also being my faculty mentor. I also thank **Dr Ashok Suhag** Head of ECE department. BML Munjal University and my industrial mentor **Mr. Shaik Chand Basha**, sr. DV engineer at SION semiconductors for their constant support and guidance throughout the course of my work. Their sincerity, thoroughness and perseverance have been a constant source of inspiration for me. It is only their cognizant efforts that my endeavors have seen light of the day.

I also take the opportunity to acknowledge the contribution of **Dr Tabish Rasheed** and **Dr Maheshwar Dwivedi** for heading and organizing the PS-2 and helping me with my queries. Last but not the least; I acknowledge my co-trainees for their contribution in the completion of the project.

I perceive this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, to attain desired career objectives.



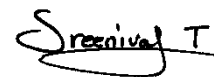
HARSHA VARDHAN SAI MACHINENI

Acknowledgment

It gives me immense pleasure to present the report of the Practice School-2 undertaken during B.Tech. Second Year stationed at SION SEMICONDUCTORS PVT.LTD, Bangalore. I thank **Dr MB Srinivas**, Dean SOET and **Prof. Manoj K. Arora**, Vice Chancellor BMU to present me with an industrial hands-on learning opportunity through PS-2 and I owe special debt of gratitude for also being my faculty mentor. I also thank **Dr Ashok Suhag** Head of ECE department. BML Munjal University and my industrial mentor **Mr. Shaik Chand Basha**, sr. DV engineer at SION semiconductors for their constant support and guidance throughout the course of my work. Their sincerity, thoroughness and perseverance have been a constant source of inspiration for me. It is only their cognizant efforts that my endeavors have seen light of the day.

I also take the opportunity to acknowledge the contribution of **Dr Tabish Rasheed** and **Dr Maheshwar Dwivedi** for heading and organizing the PS-2 and helping me with my queries. Last but not the least; I acknowledge my co-trainees for their contribution in the completion of the project.

I perceive this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, to attain desired career objectives.



SREENIVASULU TANGUTURI

Acknowledgment

It gives me immense pleasure to present the report of the Practice School-2 undertaken during B.Tech. Second Year stationed at SION SEMICONDUCTORS PVT.LTD, Bangalore. I thank **Dr MB Srinivas**, Dean SOET and **Prof. Manoj K. Arora**, Vice Chancellor BMU to present me with an industrial hands-on learning opportunity through PS-2 and I owe special debt of gratitude for also being my faculty mentor. I also thank **Dr Ashok Suhag** Head of ECE department. BML Munjal University and my industrial mentor **Mr. Shaik Chand Basha**, sr. DV engineer at SION semiconductors for their constant support and guidance throughout the course of my work. Their sincerity, thoroughness and perseverance have been a constant source of inspiration for me. It is only their cognizant efforts that my endeavors have seen light of the day.

I also take the opportunity to acknowledge the contribution of **Dr Tabish Rasheed** and **Dr Maheshwar Dwivedi** for heading and organizing the PS-2 and helping me with my queries. Last but not the least; I acknowledge my co-trainees for their contribution in the completion of the project.

I perceive this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, to attain desired career objectives.



HARSHITH NIMMAGADDA

PROJECT 1: UNIVERSAL ASYNCHRONOUS RECEIVER AND TRANSMITTER

1.INTRODUCTION

1.1. HISTORY:

In early days at the time of Technology revolution(1950's-1960's) the whole world is craving for better Serial communication protocols. This type of communication had begun when the Chinese threw smoke into the air as a signal and later these protocols changed their shape into Current loops, Morse Code, Teletype, Baudot Code, ASCII code. Teletype held as excellent I/O device for basic computers. Later, in 1960s C. Gordon Bell is an American Electrical engineer, worked as an employee and manager for DEC who made several designs for their PDP's. Bell designed the first UART for their PDP's the main objective for this UART its feasibility of sampling and convert to the digital domain, UART is configurable anyway according to their requirements. Later many MNC's started designing their own communication protocols for their PC's such as, Western Digital developed largely available UART the WD1402A in 1971, they started growing precipitously. SCN2651 from Signetic2650 family, CIA from MOS Technology INC, SCI from Motorola, SCC from Zilog, then IBM replaced RS-232 COM ports by USB. Early times these RS-232 COM ports are made standard ports for serial communication still many processors and MCU's use UART's in their chips to give the ability for hardware engineers to interface with other chips which uses RS-232 as their default ports.

1.2 PROBLEM STATEMENT AND OBJECTIVE:

Since, nowadays we can see the use of serial communication everywhere, in telecommunication and several types of data transmission, for example, to project a picture through one's PC they require a Port which works on serial data transmission mechanism. Thus, here lies our objective i.e. to Develop a basic UART (Universal Asynchronous Receiver/Transmitter) protocol using Verilog and demonstrate its working through simulations. Which can be further modified and can be dumped into FPGA's.

1.3 Motivation:

The motivation behind this project was, as a Hardware programmer it's the basic requirement to know the flow of communication between two systems. As, I observed in our college that whenever the VGA cable connected to a laptop for projection, I was longing to know how data is transferred through that cable. I have taken this question to my Industry mentor and he asked me to do the project on UART protocol so that I can find my answer. This project will be the basis of one of the communication protocols and It helps many fellow students for a better way of learning on communication through Verilog Hardware Description Language.

1.4 Method of tackling a problem:

We confirm the given problem statement and hypothesis by designing the UART receiver and transmitter individually. Due to the less complexity of Transmitter than Receiver, we will implement it first at certain Baud Rates. Baud rates are the rate which measures the speed of communication between two systems at bits per second(bps). This needs to be same for both receiver and transmitter technically the most common baud rates for simple stuff, where speed isn't important is 9600bps and there are many baud rates but there are some limits also how fast the data can be transferred/received.

The steps we are following for this project are:

1. Firstly, we will approach the FSM (Finite state machine) technique for both Receiver and Transmitter. FSM these are the computational methods used to solve sequential logic, these are mainly used in mathematical computation, artificial intelligence, logic, etc.,
2. After writing the FSM algorithms we will write the Verilog code for FSM, and simulate it Xilinx, ModelSIM, etc., these are the software tools which are used for synthesis and analysis of HDL designs.
3. After implementing the code in these simulators, we will get the schematics and simulations of the Receiver and Transmitter, by seeing we can analyze the design.

1.5 Requirements:

Tools Required for this project are:

- Xilinx Vivado
- ModelSIM

Xilinx Vivado is used for synthesis and Implementation and for power report, Errors, Routing. ModelSIM is used only for simulation purpose to get the output waveform.

1.6 Assumptions:

- The assumption we have taken in this project is that the baud rate of the receiver and transmitter can be any one of these 1200 bps, 2400bps, 4800bps, 9600bps, 19200bps, 38400bps, 57600bps, 115200bps.
- The receiver is sampling 8 times the data transmitted.

1.7 Related work (Literature review):

Mitu raj in Indian Journal of Science and Technology, Vol 10(25), DOI:10.17485/ijst/2017/v10i25/115278, July 2017 describes “UART Receiver Synchronization: Investigating the Maximum Tolerable Clock Frequency Deviation”. In this paper, Mitu raj Aim was to analyze the maximum tolerable clock frequency Deviation in the UART receiver for accurate data reception and shown mathematically that Maximum tolerable clock frequency deviation increases with increasing oversampling and it was independent of Baud rate of operation.

Khan Zarrarahmed et al in International Journal of Scientific & Engineering Research, Volume 5, Issue 5, May-2014 945 ISSN 2229-5518, describes "Design and Simulation of UART for Communication between FPGA and TDC using VHDL". In this paper, Khan Zarrarahmed et al objective was to design a UART for the communication between FPGA and TDC (Treatment delivery controller). FPGA has many advantages over controlling. In medical LINAC for the cancer treatment delivery controller has many modules like Pulse modulator cabinet, Drive stand, Gantry, Mirror assembly, Collimator, etc and to control each module the FPGA is used. Generally, the FPGA's interfaces with the hardware to have a communication with the TDC.

Amrutha Gorabal and Nayana D K in International Journal of Engineering & Technology, 7 (3.12) (2018) 23-27, describes “FPGA Implementation of UART with Single Error Correction and Double Error Detection (UART-SEC-DED)” the objectives of this paper were to design a UART with single error correction and Double error correction and this introduced methodology uses Hamming encoders and decoders for error correction.



Sion semiconductors profile

Company:

Sion semiconductor is a world class company that offers several technological services and products in semiconductors and embedded fields. Their products have strong fully verified and powerful FPGA IPS and they also produce low-cost rapid prototyping FPGA development boards. They are well known for their Quality services and their vision & mission.

Technology:



Fig:1 Technology services of SION SEMICONDUCTORS.

Sion Semiconductors own team of highly experienced designers possess immense SoC / ASIC / FPGA design experience ranges from architecture to netlist on high frequency chips used in several industries - automotive, mobile, networking, transmission and processor industries. In addition to these they have hands-on experience on various EDA tool chains from Cadence, Mentor Graphics and Synopsys, and on several latest FPGA devices from Xilinx and Altera.

Products:

Unlike other companies, SION semiconductors mainly aimed to produce high quality FPGA IPS not only they verify but also, they develop the protocols used in video/Image processing communication. On the other hand, they also develop memory models like Hybrid memory cube (HBC), NAND flash, NOR flash etc.,

Training Programs:

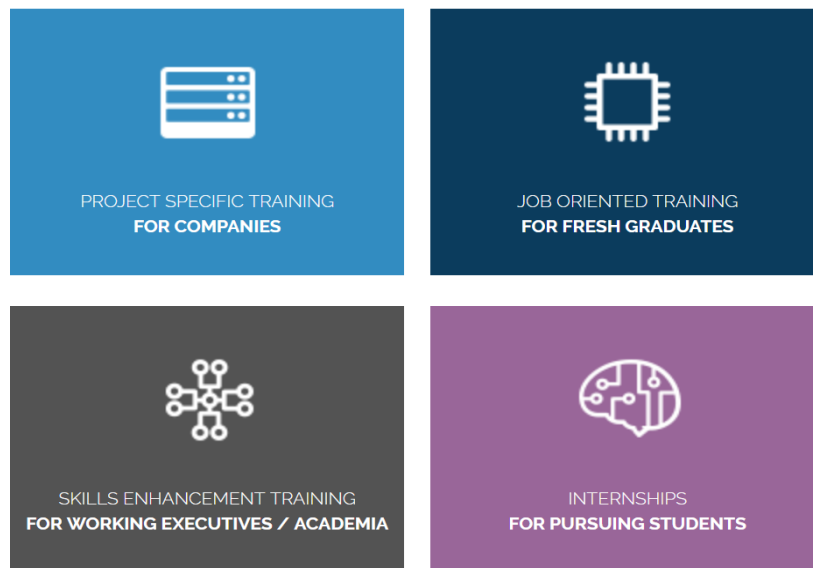


Fig:2 Training Programs offered by SION.

Truly, this company doesn't crave for experienced people they want only passionate people, for them they offer many training programs for beginners they encourage having 2-months period Internship and 6- month period Internship. They also offer corporate training based on the company requirements, they also have skill- enhancement training, Job orienting training programs.

Their career training programs are:

VLSI Technology Training Offerings:

- [Advanced VLSI Design & Verification \(Front-end\)](#)
- [Advanced Verification using System Verilog/UVM](#)
- [Advanced FPGA Design](#)

Embedded Systems Training Offerings:

- [Advanced Embedded Systems Design](#)
- [Automotive Embedded Systems Design](#)

Company affiliations:

Sion Semiconductors not only famous for their products but also, they have very good and standard collaboration across the nation. The list of their clients/ EDA vendors/ sales partners:

1. Infosys
2. BHEL (Bharath electronics Limited)
3. ARM
4. FPGA Bench
5. EACT Technologies
6. IESA (India Electronics and semiconductors association)
7. Xilinx
8. Altera

3.PROJECT METHODOLOGY

3.1 INTRODUCTION:

UART (Universal Asynchronous Receiver and Transmitter) is an individual IC that is placed in every MCU and used for serial communications through serial ports like (RS-232, RS-422, RS-423, etc..). UART operates in different modes i.e.

- Simplex
- Half-Duplex
- Full-Duplex

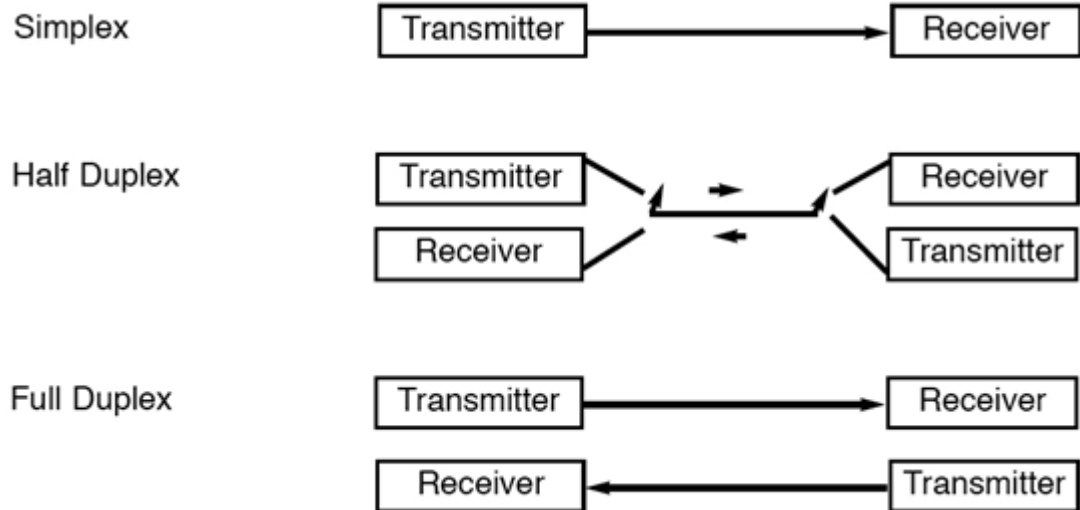


Fig:3 Modes of serial communication.

Simplex: In Simplex form, the data can be only transmitted it's a one-way process.

Half-Duplex: In this form, the data can be either transmitted or received at a time.

Full-Duplex: In this form, the data can be transmitted or received from both the directions at a time it's a two-way process most of the UART's are full-duplex.

There is also USART (Universal Synchronous/Asynchronous Receiver and Transmitter) it is faster than UART and slower than USB. Here, the data transmission and reception is synchronized with the clock. UART communication is Asynchronous in which there is no relationship between receiver and transmitter they require two separate clocks and they both need to agree to the same baud rates.

Before designing the UART we need to define our data frame (which format our data needs to be sent or received. For this project our data frame is:

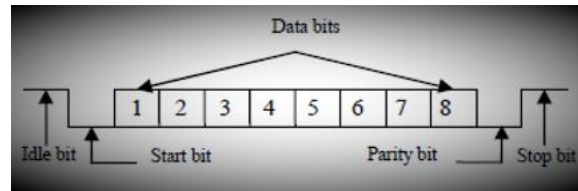


Fig:4 Data Frame for UART.

We can see our data frame consists of

- Start bit-1
- Data bits- {0:7} 8 bits data
- Parity bit-1(even or odd)
- Stop bit-1

3.2 OVERVIEW ON UART:

The Basic concept of UART is the transmitter in UART fetches data parallelly and transmits serially bit by bit. On the other hand, the receiver fetches data serially and transmits data parallelly. Since UART is Asynchronous the receiver doesn't know when the data is coming so the receiver generates a local clock to synchronize with the transmitter. Before transmission or reception, both agree to each other timing parameters and special bits are added. Whenever the data is sent to a receiver, commonly the bus will be in IDLE state (i.e. Logic high '1') whenever it notices the fall of triggering (Logic low '0') it starts receiving the data and it continues to receive the data one stop bit must need to be provided. The serial port used for UART is RS-232.

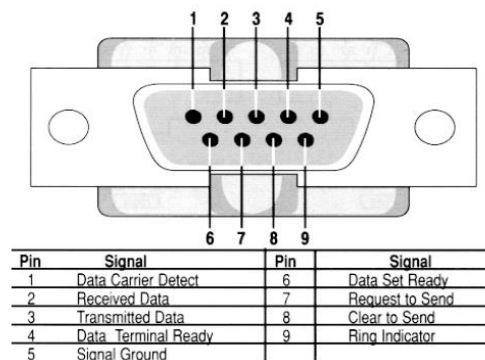


Fig:5 Pin diagram of RS-232.

3.3 BLACK BOX VIEW OF TRANSMITTER:

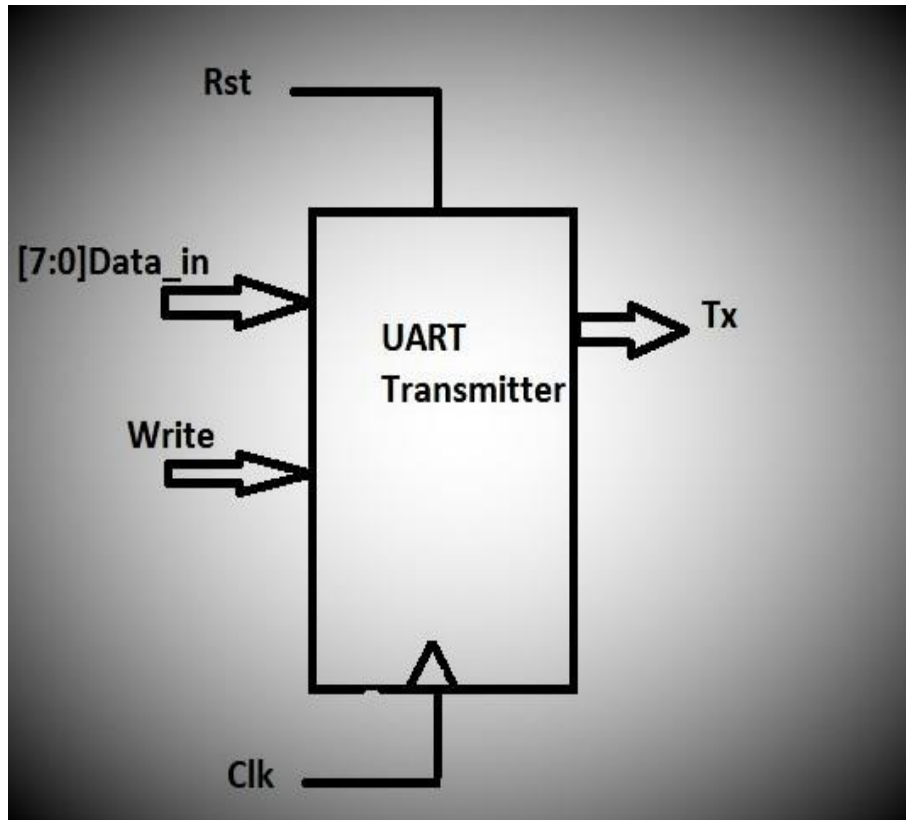


Fig:6 Black Box view of Transmitter.

UART transmitter the data is written parallelly at a time and transmits the loaded data bit by bit at the certain clock frequency. We need 4 input ports and 1 output ports and some temporary registers.

Where

Clk: System Clk.

Write Indicates transmitter to write the data.

Data_in: 8-bit data to be transmitted.

Rst: Resets the transmitter.

Tx: Output data that need to be transmitted.

Tx_start: It indicates that the data is ready to transmit.

3.4 Working of Transmitter:

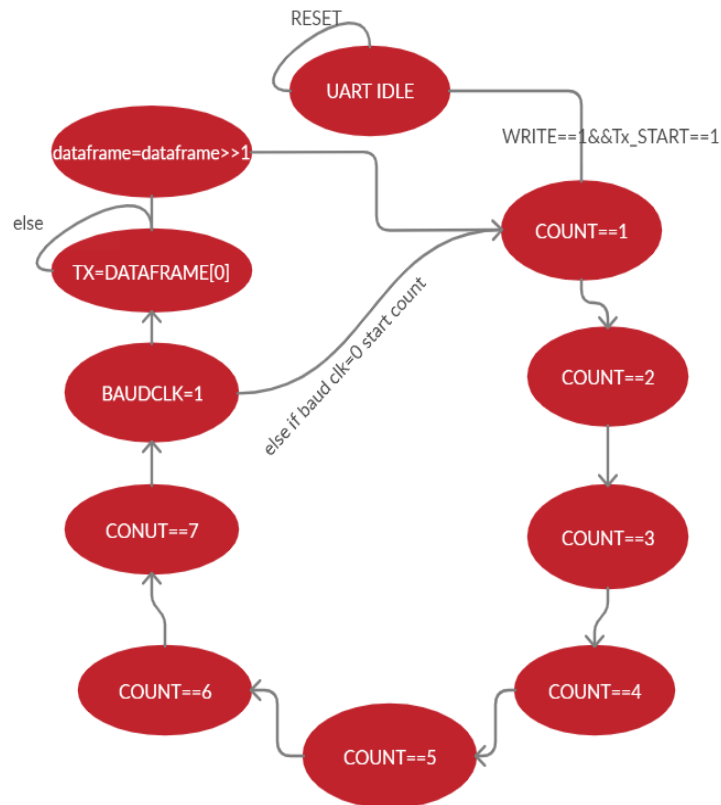


Fig:7 FSM of Transmitter.

- Whenever the reset comes the UART transmitter stays in IDLE state.
- When the write and Tx_start are enabled counter begins counting until 7.
- After 7 the baud clk turns to 1 at the rising edge of the clock.
- At next rising edge of the clock, the Tx loads the first bit of the data frame.
- At the rising edge of the clock, the data frame starts shifting the data towards the right.
- If baud clock isn't '1' the count begins from the current state.
- After shifting the data again count begins.

3.5 Black Box view of Receiver:

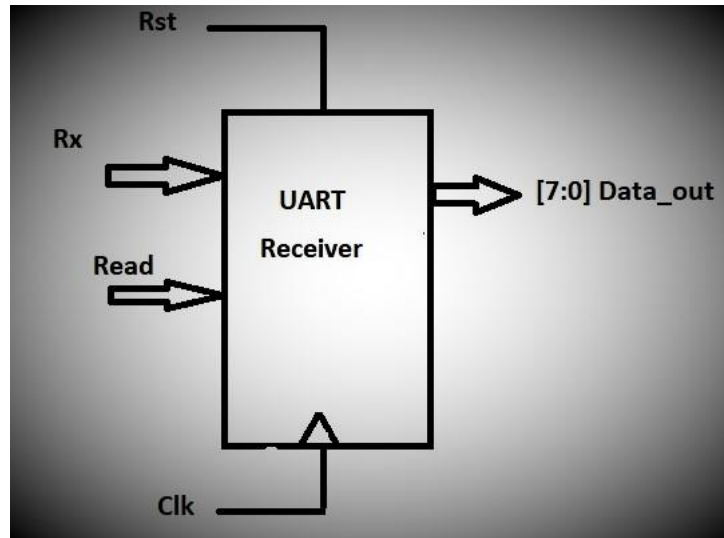


Fig:8 Black boxes of view of Receiver.

UART receiver which receives data serially and transmits parallelly. It requires 4 input ports and 1 output port and some temporary registers.

Where

Clk: system clk.

Rst: Resets the system.

Rx: Receives data bit by bit.

Read: Indicates that the receiver to receive data.

Data_out: Data that needs to be transmitted

Rx_start: Indicates that the receiver is ready to receive the data.

3.6 Working of Receiver:

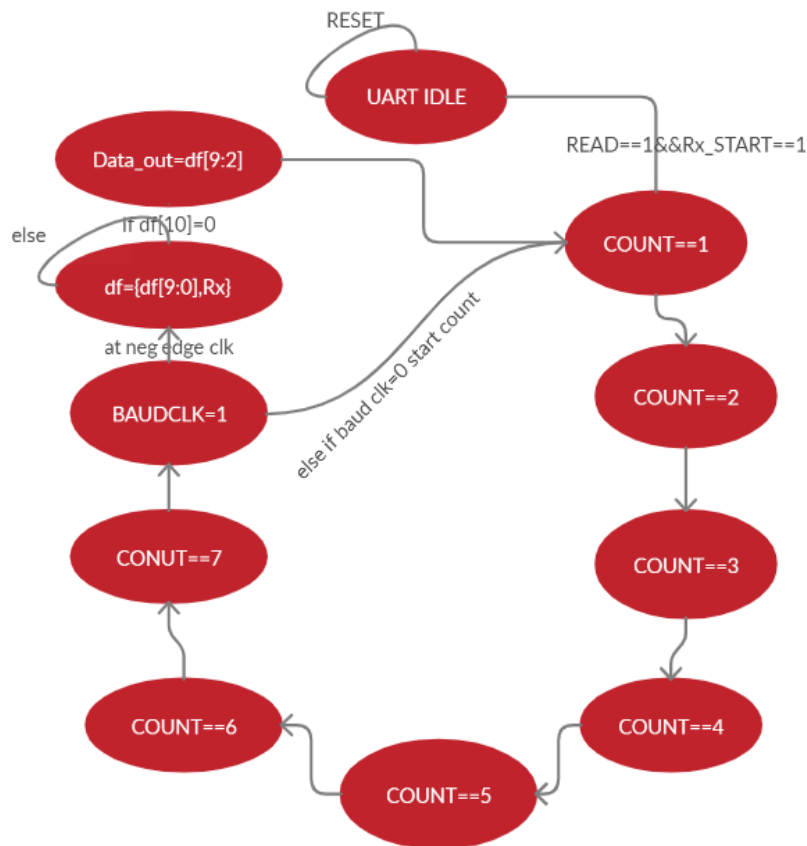


Fig:9 FSM of Receiver.

- Whenever the reset comes the UART transmitter stays in IDLE state.
- When the Read and Rx_start are enabled counter begins counting until 7.
- After 7 the baud clk turns to 1 at the rising edge of the clock.
- At next rising edge of the clock, the Data frame loads the data in a {Data frame [9:0], Rx} format.
- At the falling edge of the clock, the Data_out will be data frame [9:2].
- If baud clock isn't '1' the count begins from the current state.
- In the top module, the Tx will be given as input to Rx and each bit transmitted through Tx will be received by Rx after the stop bit recognized the data will be out from the receiver.

4.RESULTS AND DISCUSSION

4.1 SIMULATIONS:

- TRANSMITTER

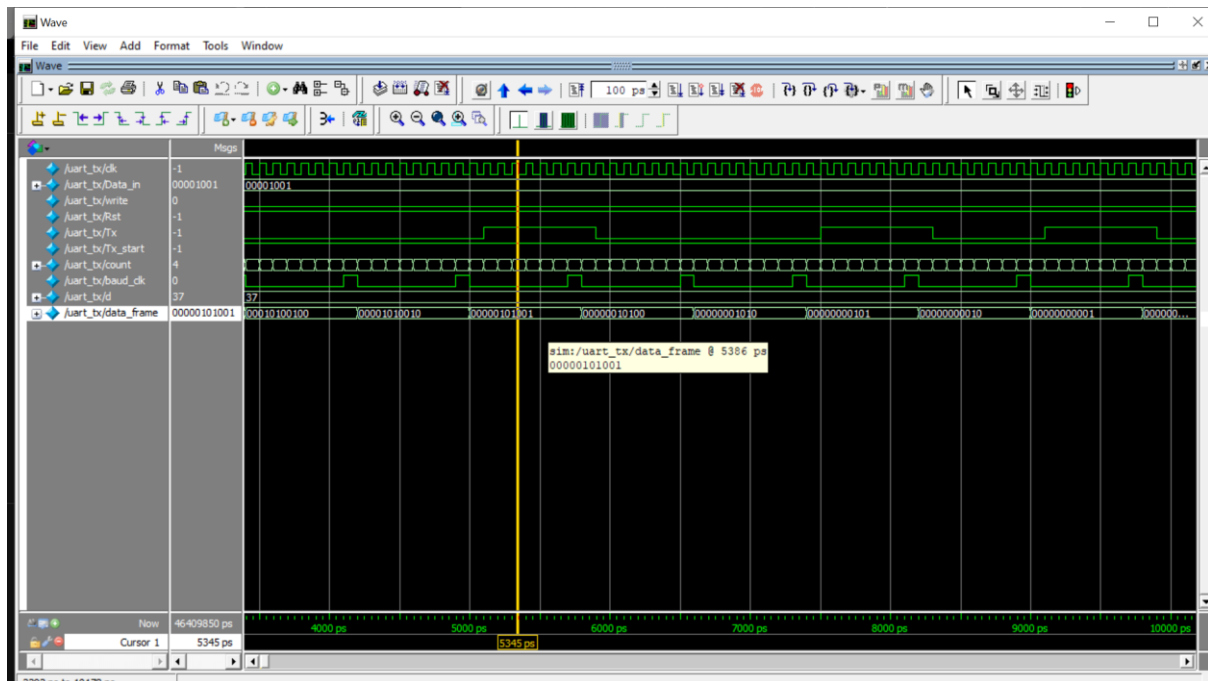


Fig:10 Waveform of the transmitter module.

The above graph shows that
 When a clock is set to high,
 When Reset is high and write is low,
 The counter starts and at the count==7 the baud clock is set to high for one clock period and
 Tx starts transmitting bits serially at the output.

Verilog HDL for Transmitter module:

```
module uart_tx(input clk,input[7:0]Data_in,input write,Rst,output
reg Tx);
reg Tx_start;
reg [3:0]count;
reg baud_clk;
reg [10:0] d,data_frame;

always@(posedge clk or negedge Rst)
```

```
begin
if(Rst==0)
begin
Tx_start<=0;
count<=0;
baud_clk<=0;
end
else
begin
if(write==0)
Tx_start<=1;
else
begin
Tx_start<=0;
count<=0;
baud_clk<=0;
end
end
end
always@(posedge clk)
begin
if(Tx_start==1)
begin
if(count<3'b111)
begin
count<=count+3'b001;
baud_clk<=1'b0;
end
else
begin
count<=3'b000;
baud_clk<=1'b1;
end
end
end
initial
begin
d={1'b0,Data_in,^Data_in,1'b1};
data_frame={d[0],d[1],d[2],d[3],d[4],d[5],d[6],d[7],d[8],d[9],d[10];
$display("d=%b",d);
end
always@(posedge clk)
begin
if(baud_clk==1'b1)
begin
```



```
Tx = data_frame[0];
end
else

Tx=data_frame[0];
end
always@( posedge clk )
begin
if(baud_clk==1'b1)
data_frame= data_frame>>1;
else
data_frame<=data_frame;

end
end module
```

- **RECEIVER:**

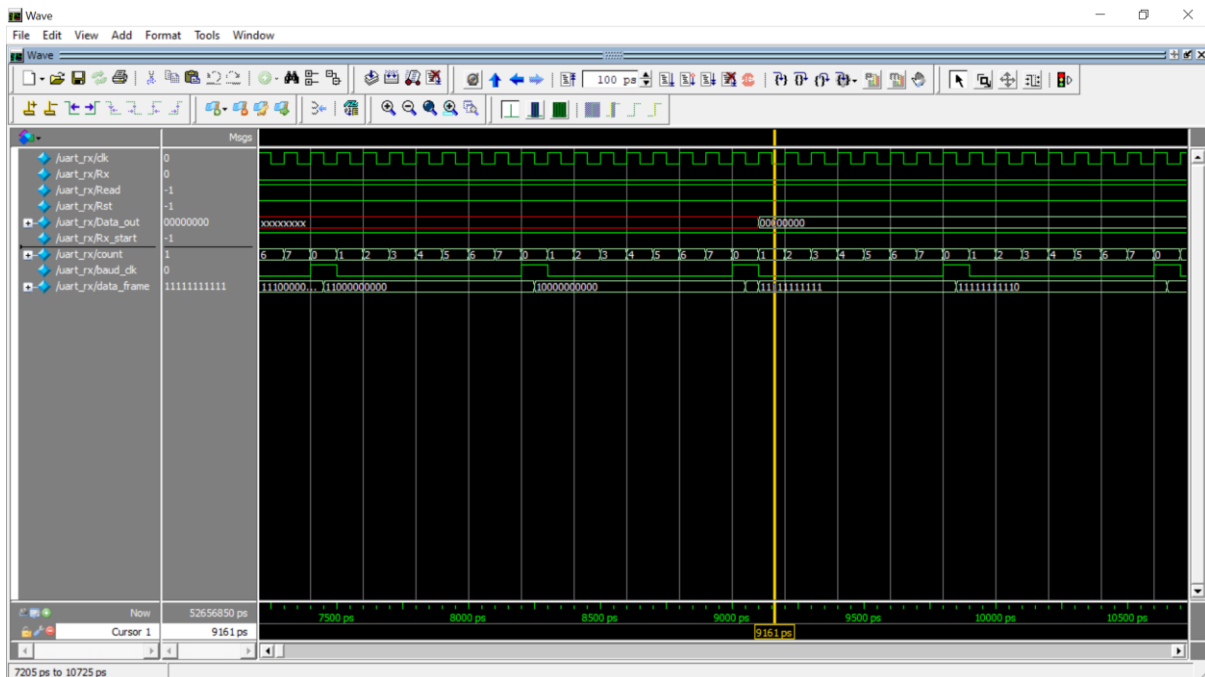


Fig:11 Waveform of the receiver module.

The above graph tells that
 When a clock is set to high,
 When reset is set to high and read to high,
 The receiver starts receiving each bit and after 10th bit becomes zero it becomes idle and ready to transmit bit parallelly.

Verilog HDL for Receiver module:

```
module uart_rx(input clk,input Rx,Read,Rst,output reg [7:0] Data_out);
reg Rx_start;
```

```

reg [3:0]count;
reg baud_clk;
reg [10:0]data_frame;
always@(posedge clk or negedge Rst)
begin
if(Rst==0)
begin
Rx_start<=0;
count<=0;
baud_clk<=0;
data_frame<=11'b1111111111;
end
else
begin
if(Read==1)
Rx_start<=1;
else
begin
Rx_start<=0;
count<=0;
baud_clk<=0;
data_frame<=11'b1111111111;
end
end
end
always@(posedge clk)
begin
if(Rx_start==1)
begin
if(count<3'b111)
begin
count<=(count+3'b001);
baud_clk<=0;
end
else
begin
count<=3'b000;
baud_clk<=1;

end
end
end

```

```

always@(negedge clk)
begin
if(baud_clk==1)
data_frame ={data_frame[9:0],Rx};
end

```

```

always@(posedge clk )
begin
if(data_frame[10]==0)
Data_out=data_frame[9:2];
end

initial
begin
data_frame<=11'b1111111111;
end
always@(posedge clk)
begin
if(data_frame==0)
begin
data_frame<=11'b1111111111;
end
end
endmodule

```

- **TOP MODULE:**

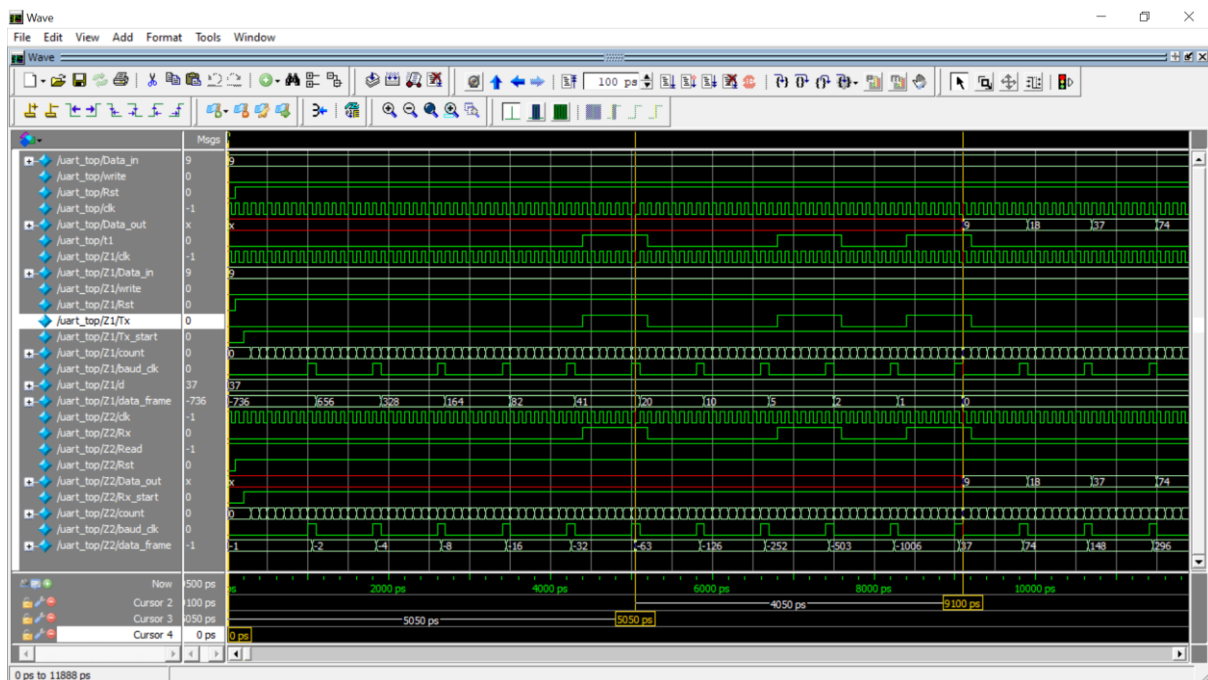


Fig: 12 Waveform of UART Top module.

The above top module graph tells that
When a clock is set high, write is low

Then transmitter module starts loading the data into it and then at the baud clock it starts transmitting each bit serially to receiver and receiver samples the data at the falling edge of the clock it acknowledges the data.

Verilog HDL for Top module:

```
module uart_top(input [7:0] Data_in,input write,Rst,clk,output [7:0]
Data_out);
wire t1;
uart_tx Z1(.Data_in(Data_in),.write(write),.Rst(Rst),.clk(clk),.Tx(t1));
uart_rx Z2(.Rx(t1),.Read(~write),.Rst(Rst),.clk(clk),.Data_out(Data_out));
endmodule
```

4.2. RTL SCHEMATICS:

4.2.1 UART Top Module

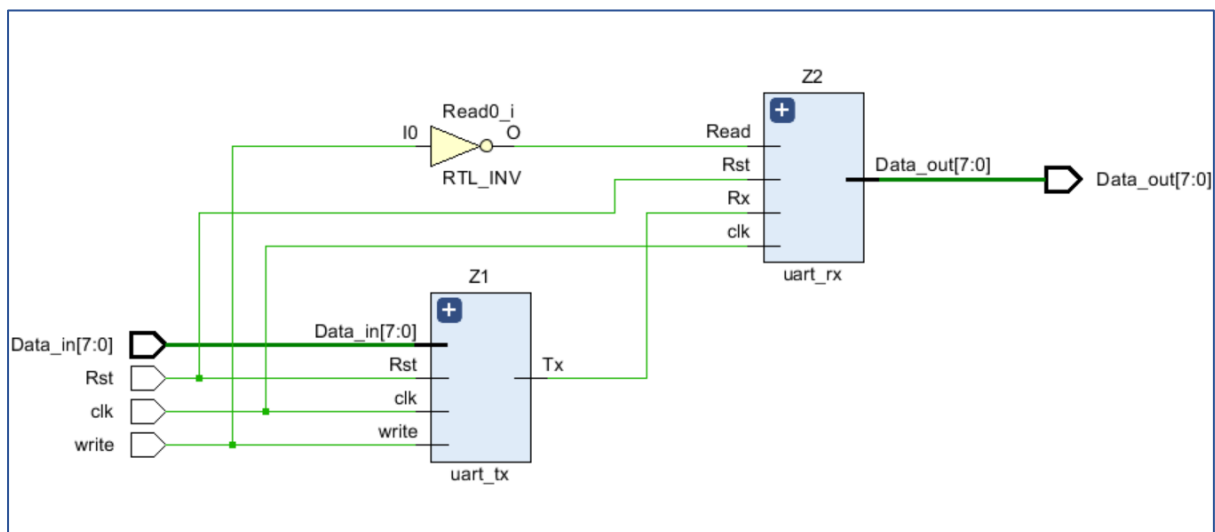


Fig:13 RTL schematic of the UART top module.

4.2.2 Transmitter

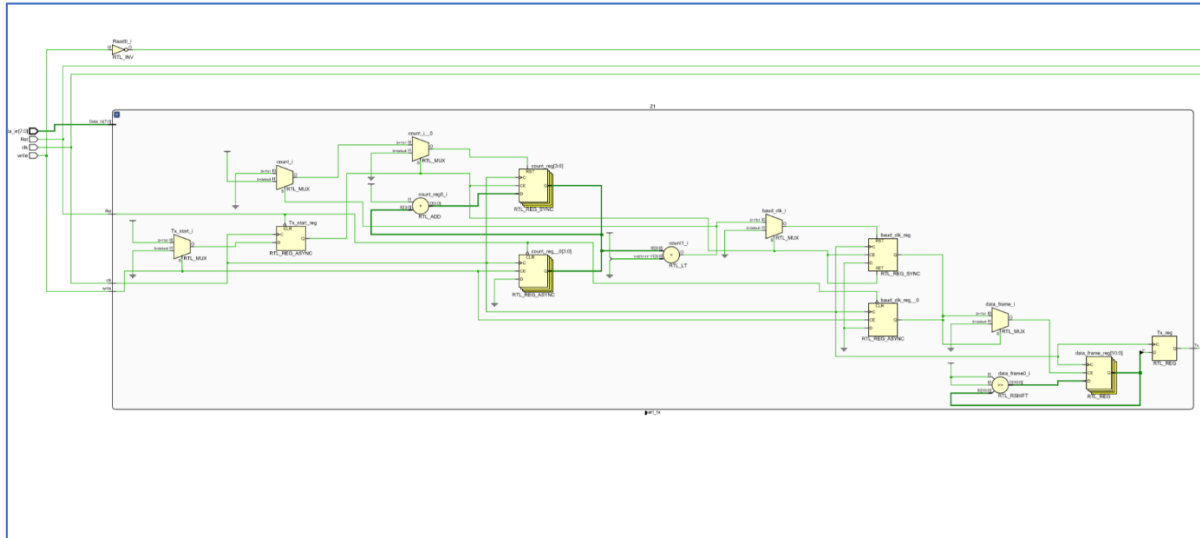


Fig:14 RTL schematic of the UART Transmitter module.

4.2.3 Receiver

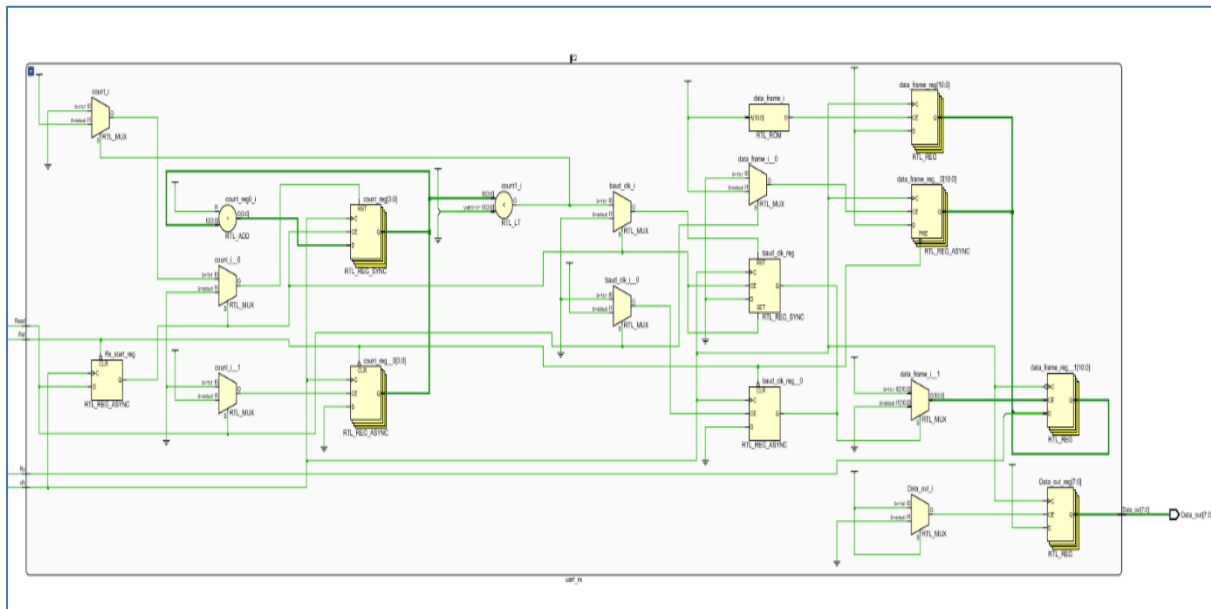


Fig:15 RTL schematic of the UART receiver module.

4.3 SYNTHESIS REPORT:

Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

```
-----
| Tool Version : Vivado v.2018.2 (win64) Build 2258646 Thu Jun 14
20:03:12 MDT 2018
| Date: Tue Jul 9 14:48:32 2019
| Host: HK007 running 64-bit major release (build 9200)
| Command      : report_utilization -file
uart_top_utilization_synth.rpt -PB uart_top_utilization_synth.pb
| Design       : uart_top
| Device: 7k70tfbv676-1
| Design State: Synthesized
-----
```

Utilization Design Information

Table of Contents

- ```

1.Summary of Registers by Type
2. Memory
3. DSP
4. IO and GT Specific
5. Clocking
6. Specific Feature
7. Primitives
8. Black Boxes
9. Instantiated Netlists
```

##### 1. Summary of Registers by Type

| Total | Clock Enable | Synchronous | Asynchronous |
|-------|--------------|-------------|--------------|
| 0     | -            | -           | -            |
| 0     | -            | -           | Set          |
| 0     | -            | -           | Reset        |
| 0     | -            | Set         | -            |
| 0     | -            | Reset       | -            |

## 2. Memory

-----

| Site Type      | Used | Fixed | Available | Util% |
|----------------|------|-------|-----------|-------|
| Block RAM Tile | 0    | 0     | 135       | 0.00  |
| RAMB36/FIFO*   | 0    | 0     | 135       | 0.00  |
| RAMB18         | 0    | 0     | 270       | 0.00  |

\* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Block RAM Tile, that tile can still accommodate a RAMB18E1

## 3. DSP

-----

| Site Type | Used | Fixed | Available | Util% |
|-----------|------|-------|-----------|-------|
| DSPs      | 0    | 0     | 240       | 0.00  |

## 4. IO and GT Specific

-----

| Site Type                 | Used | Fixed | Available | Util% |
|---------------------------|------|-------|-----------|-------|
| Bonded IOB                | 8    | 0     | 300       | 2.67  |
| Bonded IPADs              | 0    | 0     | 26        | 0.00  |
| Bonded OPADs              | 0    | 0     | 16        | 0.00  |
| PHY_CONTROL               | 0    | 0     | 6         | 0.00  |
| PHASER_REF                | 0    | 0     | 6         | 0.00  |
| OUT_FIFO                  | 0    | 0     | 24        | 0.00  |
| IN_FIFO                   | 0    | 0     | 24        | 0.00  |
| IDELAYCTRL                | 0    | 0     | 6         | 0.00  |
| IBUFDS                    | 0    | 0     | 288       | 0.00  |
| GTXE2_COMMON              | 0    | 0     | 2         | 0.00  |
| GTXE2_CHANNEL             | 0    | 0     | 8         | 0.00  |
| PHASER_OUT/PHASER_OUT_PHY | 0    | 0     | 24        | 0.00  |

|                             |   |   |     |      |
|-----------------------------|---|---|-----|------|
| PHASER_IN/PHASER_IN_PHY     | 0 | 0 | 24  | 0.00 |
| IDELAYE2/IDELAYE2_FINEDELAY | 0 | 0 | 300 | 0.00 |
| ODELAYE2/ODELAYE2_FINEDELAY | 0 | 0 | 100 | 0.00 |
| IBUFDS_GTE2                 | 0 | 0 | 4   | 0.00 |
| ILOGIC                      | 0 | 0 | 300 | 0.00 |
| OLOGIC                      | 0 | 0 | 300 | 0.00 |

## 5. Clocking

| Site Type  | Used | Fixed | Available | Util% |
|------------|------|-------|-----------|-------|
| BUFGCTRL   | 0    | 0     | 32        | 0.00  |
| BUFIO      | 0    | 0     | 24        | 0.00  |
| MMCME2_ADV | 0    | 0     | 6         | 0.00  |
| PLLE2_ADV  | 0    | 0     | 6         | 0.00  |
| BUFMRCE    | 0    | 0     | 12        | 0.00  |
| BUFHCE     | 0    | 0     | 96        | 0.00  |
| BUFR       | 0    | 0     | 24        | 0.00  |

## 6. Specific Feature

| Site Type   | Used | Fixed | Available | Util% |
|-------------|------|-------|-----------|-------|
| BSCANE2     | 0    | 0     | 4         | 0.00  |
| CAPTUREE2   | 0    | 0     | 1         | 0.00  |
| DNA_PORT    | 0    | 0     | 1         | 0.00  |
| EFUSE_USR   | 0    | 0     | 1         | 0.00  |
| FRAME_ECCE2 | 0    | 0     | 1         | 0.00  |
| ICAPE2      | 0    | 0     | 2         | 0.00  |
| PCIE_2_1    | 0    | 0     | 1         | 0.00  |
| STARTUPE2   | 0    | 0     | 1         | 0.00  |
| XADC        | 0    | 0     | 1         | 0.00  |



## 7. Primitives

-----

| Ref Name | Used | Functional Category |
|----------|------|---------------------|
| OBUF     | 8    | IO                  |

## 8. Black Boxes

-----

| Ref Name | Used |
|----------|------|
|----------|------|

## 9. Instantiated Netlists

-----

| Ref Name | Used |
|----------|------|
|----------|------|

## Synthesis Design information

### 1.Start RTL Component Statistics

-----

-----

Detailed RTL Component Info :

+---Adders :

2 Input 4 Bit Adders := 2

+---Registers :

11 Bit Registers := 1

8 Bit Registers := 1

1 Bit Registers := 3

+---Muxes :

2 Input 1 Bit Muxes := 1

-----

-----

## Finished RTL Component Statistics

## 2.Start RTL Hierarchical Component Statistics

### Hierarchical RTL Component report

#### Module uart\_tx

##### Detailed RTL Component Info :

##### +---Adders :

2 Input 4 Bit Adders := 1

##### +---Registers :

11 Bit Registers := 1

1 Bit Registers := 2

#### Module uart\_rx

##### Detailed RTL Component Info :

##### +---Adders :

2 Input 4 Bit Adders := 1

##### +---Registers :

8 Bit Registers := 1

1 Bit Registers := 1

##### +---Muxes :

2 Input 1 Bit Muxes := 1

## Finished RTL Hierarchical Component Statistics

## 3.Report BlackBoxes:

| BlackBox name | Instances |
|---------------|-----------|
|               |           |

## 4.Report Cell Usage:

| Cell | Count |
|------|-------|
| 1    | 8     |

## 5.Report Instance Areas:

|   | Instance | Module | Cells |
|---|----------|--------|-------|
| 1 | top      |        | 8     |

## Power Report of UART:

Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

```

| Tool Version : Vivado v.2018.2 (win64) Build 2258646 Thu Jun
14 20:03:12 MDT 2018
| Date : Tue Jul 9 14:50:10 2019
| Host : HK007 running 64-bit major release (build
9200)
| Command : report_power -file uart_top_power_routed.rpt -
pb uart_top_power_summary_routed.pb -rpx uart_top_power_routed.rpx
| Design : uart_top
| Device : xc7k70tfbv676-1
| Design State : routed
| Grade : commercial
| Process : maximum
Characterization : Production


```

## Power Report

### Table of Contents

- ```
-----
1. Summary
1.1 On-Chip Components
1.2 Power Supply Summary
1.3 Confidence Level
2. Settings
2.1 Environment
2.2 Clock Constraints
3. Detailed Reports
3.1 By Hierarchy
```

1. Summary

Total On-Chip Power (W)	0.145
Design Power Budget (W)	Unspecified*
Power Budget Margin (W)	NA
Dynamic (W)	0.000
Device Static (W)	0.145
Effective TJA (C/W)	1.9
Max Ambient (C)	84.7
Junction Temperature (C)	25.3
Confidence Level	High
Setting File	---
Simulation Activity File	---
Design Nets Matched	NA

* Specify Design Power Budget using, set_operating_conditions - design_power_budget <value in Watts>

1.1 On-Chip Components

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	4	---	---
Slice Logic	0.000	1	---	---
Others	0.000	1	---	---
I/O	0.000	8	300	2.67
Static Power	0.145			
Total	0.145			

1.2 Power Supply Summary

Source	Voltage (V)	Total (A)	Dynamic (A)	Static (A)
Vccint	1.000	0.518*	0.000	0.518
Vccaux	1.800	0.057*	0.000	0.057
Vcco33	3.300	0.000	0.000	0.000
Vcco25	2.500	0.000	0.000	0.000
Vcco18	1.800	0.000	0.000	0.000
Vcco15	1.500	0.000	0.000	0.000
Vcco135	1.350	0.000	0.000	0.000
Vcco12	1.200	0.000	0.000	0.000
Vccaux_io	1.800	0.000	0.000	0.000
Vccbram	1.000	0.040*	0.000	0.040
MGTAVcc	1.000	0.000	0.000	0.000
MGTAVtt	1.200	0.000	0.000	0.000
MGTVccaux	1.800	0.000	0.000	0.000
Vccadc	1.800	0.030	0.000	0.030

* Power-up current

1.3 Confidence Level

User Input Data	Confidence	Details
Action		
Design implementation state	High	Design is routed
Clock nodes activity	High	User specified more
than 95% of clocks		
I/O nodes activity	High	User specified more
than 95% of inputs		
Internal nodes activity	High	User specified more
than 25% of internal nodes		
Device models	High	Device models are
Production		

Overall	confidence level	High	
+-----+-----+			
-----+-----+			

2. Settings

2.1 Environment

+-----+-----+	
Ambient Temp (C)	25.0
ThetaJA (C/W)	1.9
Airflow (LFM)	250
Heat Sink	medium (Medium Profile)
ThetaSA (C/W)	3.4
Board Selection	medium (10"x10")
# of Board Layers	12to15 (12 to 15 Layers)
Board Temperature (C)	25.0
+-----+-----+	

2.2 Clock Constraints

+-----+-----+		
Clock	Domain	Constraint (ns)
+-----+-----+		

3. Detailed Reports

3.1 By Hierarchy

+-----+-----+	
Name	Power (W)
+-----+-----+	
uart_top	0.000
+-----+-----+	

Noise Analysis:

SSN Report Summary

Settings	State
Created on: Tue Jul 9 15:42:31 2019 Results Name: ssn_4 Project Name: Final Project Family: Kintex-7 Project Part: xc7k70tfbv676 Temperature Grade: commercial SSN Data Version: Production Package Version: FINAL 2012-06-26 Package Pin Delay Version: VERS. 2.0 2012-06-26	Status: Full Analysis; Passed Messages: <ul style="list-style-type: none"> • 0 critical warning(s) • 0 warning(s) • 1 info(s)
Possible SSN Ports: 8 port(s) Analyzed Ports: 8/8 port(s) (100.0%) Ports within SNN Margin: 8/8 port(s) (100.0%) Ports Exceeding SNN Margin: 0/8 port(s) (0.0%) Unplaced Ports: 0 port(s)	

IO Bank	VCC O	Signal Name	Pin Number	IO Standard	Drive (mA)	Slew Rate	OFFCHIP_TERM	Remaining Margin (%)	Result
13	1.8	Data_out[0]	U16	LVC MOS18	12	SLOW	FP_VTT_50	90.7	PASS
13	1.8	Data_out[1]	P18	LVC MOS18	12	SLOW	FP_VTT_50	85.2	PASS
13	1.8	Data_out[2]	R18	LVC MOS18	12	SLOW	FP_VTT_50	85.2	PASS
13	1.8	Data_out[3]	T17	LVC MOS18	12	SLOW	FP_VTT_50	85.2	PASS
13	1.8	Data_out[4]	U17	LVC MOS18	12	SLOW	FP_VTT_50	85.2	PASS
13	1.8	Data_out[5]	M19	LVC MOS18	12	SLOW	FP_VTT_50	86.4	PASS
13	1.8	Data_out[6]	N18	LVC MOS18	12	SLOW	FP_VTT_50	87.5	PASS
13	1.8	Data_out[7]	R17	LVC MOS18	12	SLOW	FP_VTT_50	85.6	PASS

Table 1: Report of SSN Analysis:

4.4 POST PLACE AND ROUTE SIMULATION:

Design Route Status

	# nets:
-----	-----
# of logical nets.....:	9:
# of nets not needing routing.....:	8:
# of internally routed nets.....:	8:
# of routable nets.....:	1:
# of fully routed nets.....:	1:
# of nets with routing errors.....:	0:
-----	-----

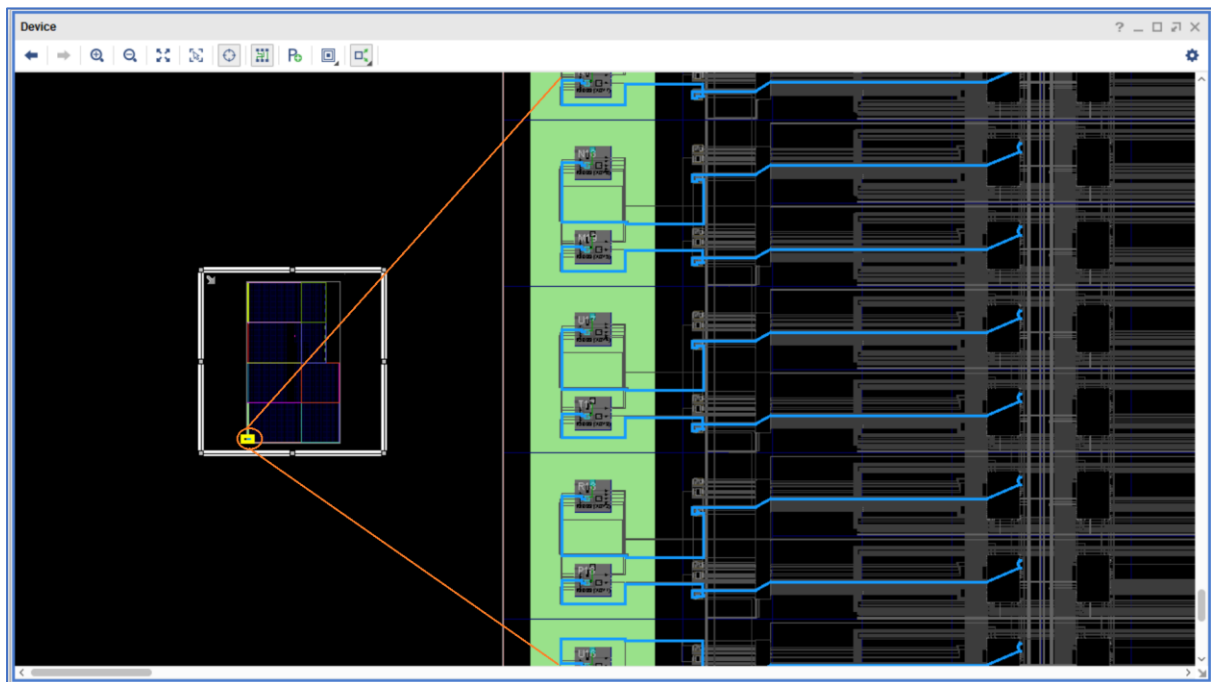


Fig:16 Floor plan of UART in VIVADO device.

5.CONCLUSIONS AND RECOMMENDATIONS

- ❖ The objective of this project is accomplished i.e. the design of UART studied in a detailed manner and written Verilog hardware description language for UART and simulated in ModelSim-Altera 6.6d and verified it's a simulation by transmitting and receiving data and the obtained results are well appreciable.
- ❖ For synthesizing and implementation of UART we have used Xilinx Vivado tool and the Floorplan report and power, and RTL schematics are done with this tool.
- ❖ We can see the power report the confidence level is high and the power dissipation is very minute. We have achieved a very less powered UART protocol.
- ❖ As a recommendation, further work can be added by making the UART as flexible for any baud rates and 9-bit data transfer and mostly we can dump this UART by making further modifications in FPGA.
- ❖ We can also make USART (Universal Synchronous /Asynchronous receiver and transmitter) which is faster than UART.

APPENDIX 1: LIST OF TABLES AND FIGURES

Fig:1 Technology services of SION SEMICONDUCTORS.

Fig:2 Training Programs offered by SION.

Fig:3 Modes of serial communication.

Fig:4 Data Frame for UART.

Fig:5 Pin diagram of RS-232.

Fig:6 Black Box view of Transmitter.

Fig:7 FSM of Transmitter.

Fig:8 Black boxes of view of Receiver.

Fig:9 FSM of Receiver.

Fig:10 Waveform of a transmitter module

Fig:11 Waveform of the receiver module.

Fig: 12 Waveform of UART Top module.

Fig:13 RTL schematic of the UART top module.

Fig:14 RTL schematic of the UART Transmitter module.

Fig:15 RTL schematic of the UART receiver module.

Fig:16 Floor plan of UART in VIVADO device.

Table 1: Report on SSN Analysis.

APPENDIX 2: REFERENCES

- [1] UART Receiver Synchronization: Investigating the Maximum Tolerable Clock Frequency Deviation by Mitu Raj Department of Electronics and Communication, Electronics Research and Development Center of India - Institute of Technology, Center for Development of Advanced Computing, Vellayambalam, Trivandrum – 695010.
- [2] Analysis of Universal Asynchronous Receiver and Transmitter for Reliable Data Transmission by Yogeesh.K., Venkatesh Kumar, Rohith. (ECE (PG student), Nagarjuna college of Eng... & Technology/VTU, Bengaluru India).
- [3] Universal Asynchronous Receiver and Transmitter (UART) by Umakanta Nanda, Sushant Kumar Patnaik Department of Electronics and Communication Engineering Silicon Institute of Technology Bhubaneswar, India.
- [4] Design Implementation of UART and SPI in single FPGA
M.Poorani ECE Department, Sri Manakula Vinayagar Engineering College, Puducherry, India.
- [5] Xilinx Vivado Video tutorials.
- [6] Reference Material is given by SION SEMICONDUCTORS.

2. DUAL PORT RANDOM ACCESS MEMORY

1. INTRODUCTION:

1.1. HISTORY:

The first form of RAM came about in 1947 with the use of the Williams tube. It contains a cathode ray tube; the data was stored on the face as electrically charged spots. The second widely used form of RAM was magnetic-core memory, invented by Frederick Viehe in 1947. Magnetic-core memory works using tiny metal rings and wires connecting to each ring. One bit of data could be stored per ring and accessed at any time. However, RAM as we know it today, as solid-state memory, was first invented in 1968 by Robert Dennard. Known specifically as dynamic random access memory, or DRAM, transistors were used to store bits of data. Intel's 1103 was the world's first available DRAM chip.

1.2. PROBLEM STATEMENT AND OBJECTIVE:

Storing the data in a systematic format and accessing the data is a huge problem faced by the world before 1950s. After the invention of RAM this problem had its end. Random-access memory (RAM) is a form of computer data storage that stores data and machine code currently being used. A random-access memory device allows data to be read or written in almost the same amount of time irrespective of the physical location of data inside the memory. So, it is more important to know how we can read the data from RAM and Write the data on to a RAM. Thus, our problem statement is to learn how a Dual port RAM (RAM containing two i/o ports, where we can read and write the data most simultaneously from both ports) works, Designing a 16X8 bit DPRAM and implementation on FPGA (Spartan 7).

1.3 MOTIVATION:

The motivation behind this project was, as a Hardware programmer it's the basic requirement to know how a DPRAM read/write the data. Now a days as the technology increases the size of the RAM and features are increasing. This project helps me and my fellow students to learn about designing RAM using Hardware descriptive language and FPGA implementation of the design.

1.4 METHOD OF TACKLING A PROBLEM:

The working principles and circuits for RAM have been learned in the University. Dual-ported RAM (DPRAM) is a type of random-access memory that allows multiple reads or writes to occur at the same time, whereas single-ported RAM which allows only one access at a time. Video RAM is a common form of dual-ported dynamic RAM mostly used for video memory, allowing the CPU to draw the image at the same time the video hardware is reading it out to the screen. Apart from VRAM, most other types of dual-ported RAM are based on static RAM technology. So here we are developing a dual port static RAM.

The steps we are following for this project are:

1. Firstly, we developed a schematic of an DPRAM. For designing using Verilog code we approach if else logics.
2. Developing the Verilog code for the read and write operation for two ports using ModelSIM Altera and Xilinx VIVADO. These are the software tools which are used for simulation, synthesis and analysis of HDL designs.
3. After compilation of the Verilog code we simulate and synthesize design schematics.
4. Development of bit stream and FPGA implementation process.

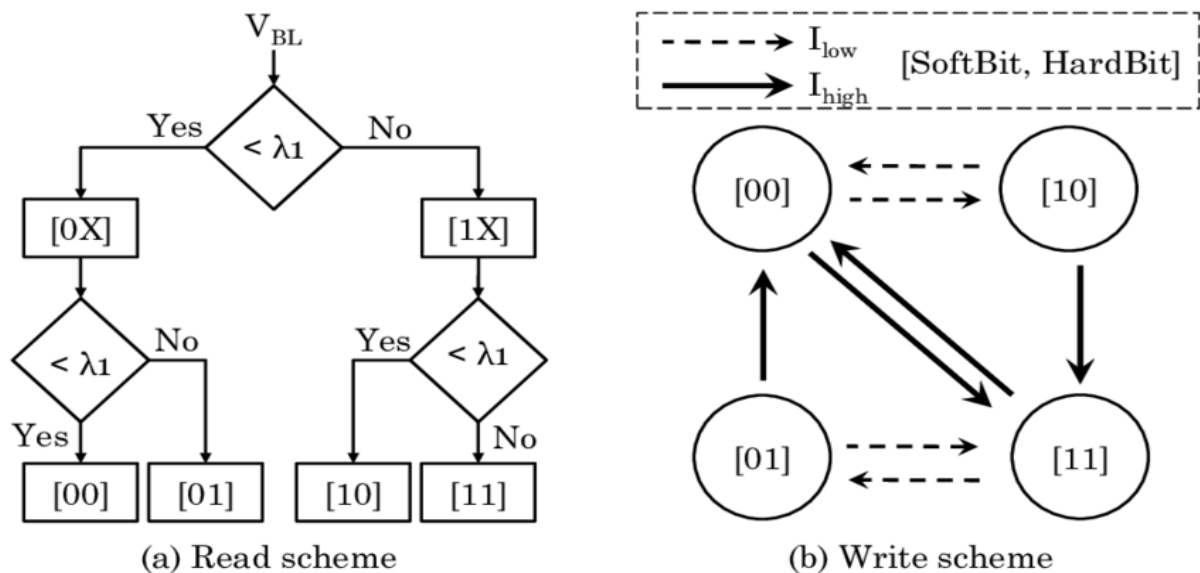


Fig:1 READ AND WRITE SCHEMATICS

1.5 REQUIREMENTS:

Tools Required for this project are:

- Xilinx Vivado
- ModelSIM

Xilinx Vivado is used for synthesis and Implementation (spartan 7) and for power report, Errors, Routing.

ModelSIM is used only for simulation purpose to get the output waveform.

1.6 ASSUMPTIONS:

Memory size 64X8 bits.

2.PROJECT METHODOLOGY

2.1 INTRODUCTION:

In today's technology, Random-access memory takes the form of integrated circuits (ICs). RAM operates in different modes i.e.

- Single port RAM
- Simple DPRAM
- True DPRAM

Memory IP	Supported Memory Mode	Features
RAM: 1-PORT	Single-port RAM	<ul style="list-style-type: none"> • Non-simultaneous read and write operations from a single address. • Read enable port to specify the behavior of the RAM output ports during a write operation, to overwrite or retain existing value. • Supports freeze logic feature.
RAM: 2-PORT	Simple dual-port RAM	<ul style="list-style-type: none"> • Simultaneous one read and one write operations to different locations. • Supports error correction code (ECC). • Supports freeze logic feature.
	True dual-port RAM	<ul style="list-style-type: none"> • Simultaneous two reads. • Simultaneous two writes. • Simultaneous one read and one write at two different clock frequencies. • Supports freeze logic feature.

Table :1 Different Modes of RAM

2.2 WRITE AND READ OPERATIONS TRIGGERING:

The embedded memory blocks like RAM vary slightly in its supported features and behaviors. One important variation is the difference in the write and read operations triggering.

Valid Write Operation that Triggers at Rising Clock Edges: This figure assumes that t_{wc} is the maximum write cycle time interval. Write operation of data 03 through port B does not meet the criteria and causes write contention with the write operation at port A, which result in unknown data at address 01. The write operation at the next rising edge is valid because it meets the criteria and data 04 replaces the unknown data.

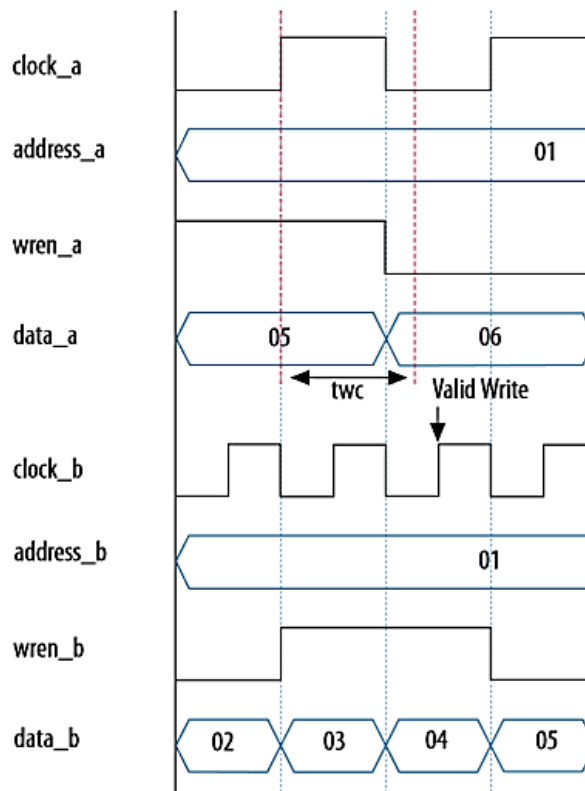


Fig:2 Write operation triggering at posedge of clk

Valid Write Operation that Triggers at Falling Clock Edges: This figure assumes that t_{wc} is the maximum write cycle time interval. Write operation of data 04 through port B does not meet the criteria and therefore causes write contention with the write operation at port A that result in unknown data at address 01. The next data (05) is latched at the next rising clock edge that meets the criteria and is written into the memory block at the falling clock edge.

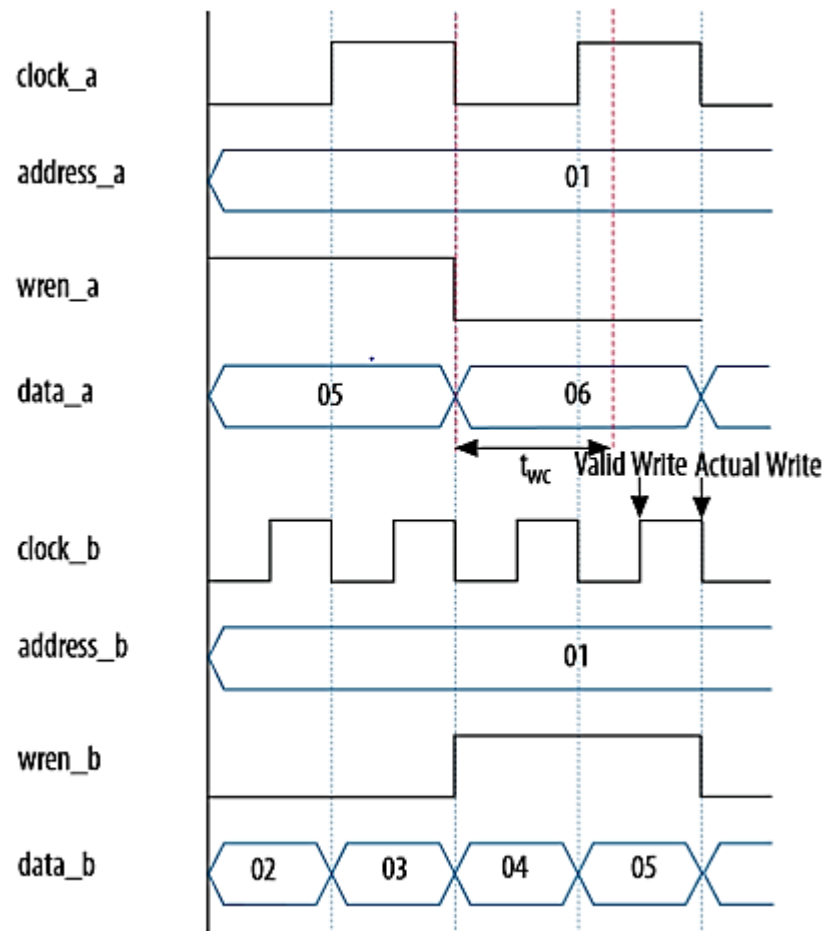


Fig:3 Write operation triggering at negedge of clk

2.3. PORT WIDTH CONFIGURATIONS:

The following equation defines the port width configuration: Memory depth (number of words) \times Width of the data input bus.

- If your port width configuration (either the depth or the width) is more than the amount an internal memory block can support, additional memory blocks (of the same type) are used. For example, if you configure your M9K as 512×36 , which exceeds the supported port width of 512×18 , two M9Ks are used to implement your RAM.
- In addition to the supported configuration provided, you can set the memory depth to a non-power of two, but the actual memory depth allocated can vary. The variation depends on the type of resource implemented.
- If the memory is implemented in dedicated memory blocks, setting a non-power of two for the memory depth reflects the actual memory depth.
- When you implement your memory using dedicated memory blocks, refer to the Fitter report to check the actual memory depth.

2.4. MIXED-WIDTH PORT CONFIGURATION:

DPRAM and DPROM support mixed-width port configuration for all memory block types except when they are implemented with logic elements LEs. The support for mixed-width port depends on the width ratio between port A and port B. In addition, the supporting ratio varies for various memory modes, memory blocks, and target devices.

Example: Memory depth of 1 word is not supported in simple dual-port and true dual-port RAMs with mixed-width port. The parameter editor prompts an error message when the memory depth is less than 2 words. For example, if the width for port A is 4 bits and the width for port B is 8 bits, the smallest depth supported by the RAM is 4 words. This configuration results in memory size of 16 bits (4×4) and can be represented by memory depth of 2 words for port B. If you set the memory depth to 2 words that results in memory size of 8 bits (2×4), it can only be represented by memory depth of 1 word for port B, and therefore the width of the port is not supported.

2.5. MAXIMUM BLOCK DEPTH CONFIGURATION:

We can limit the maximum block depth of the dedicated memory block you use. The memory block can be sliced to your desired maximum block depth.

EXAMPLE: The capacity of an M9K block is 9,216 bits, and the default memory depth is 8K, in which each address can store 1 bit ($8K \times 1$). If you set the maximum block depth to 512, the M9K block is sliced to a depth of 512 and each address can store up to 18 bits (512×18).

We can use this option to save power usage in our devices. However, this parameter might increase the number of LEs and affects the design performance. When the RAM is sliced shallower, the dynamic power usage decreases. However, for a RAM block with a depth of 256, the power used by the extra LEs starts to outweigh the power gain achieved by shallower slices. We can also use this option to reduce the total number of memory blocks used (but at the expense of LEs). The $8K \times 36$ RAM uses 36 M9K RAM blocks with a default slicing of $8K \times 1$. By setting the maximum block depth to 1K, the $8K \times 36$ RAM can fit into 32 M9K blocks. The maximum block depth must be in a power of two, and the valid values vary among different dedicated memory blocks.

2.6. CLOCKING MODES:

Clocks can be in different modes like,

- Single clock for all ports
- Read/Write clocks
- Input/output clocks
- Clock independent

2.7. ERROR CODE CORRECTION AND FREEZE LOGIC:

Error Correction Code The error correction code (ECC) feature detects and corrects output data errors. You have the option to use pipeline registers to improve performance. The ECC feature is supported only in the following conditions:

- Memory blocks and not MLABs or logic cells
- Simple dual-port mode
- Same-width ports
- Byte-enable feature is disabled

When the ECC feature is enabled, the result of a RDW in a mixed-port configuration is always Don't care.

Freeze Logic: The freeze logic feature specifies whether to implement clock-enable circuitry for use in a partial reconfiguration region.

2.8. DPRAM BLACK BOX VIEW:

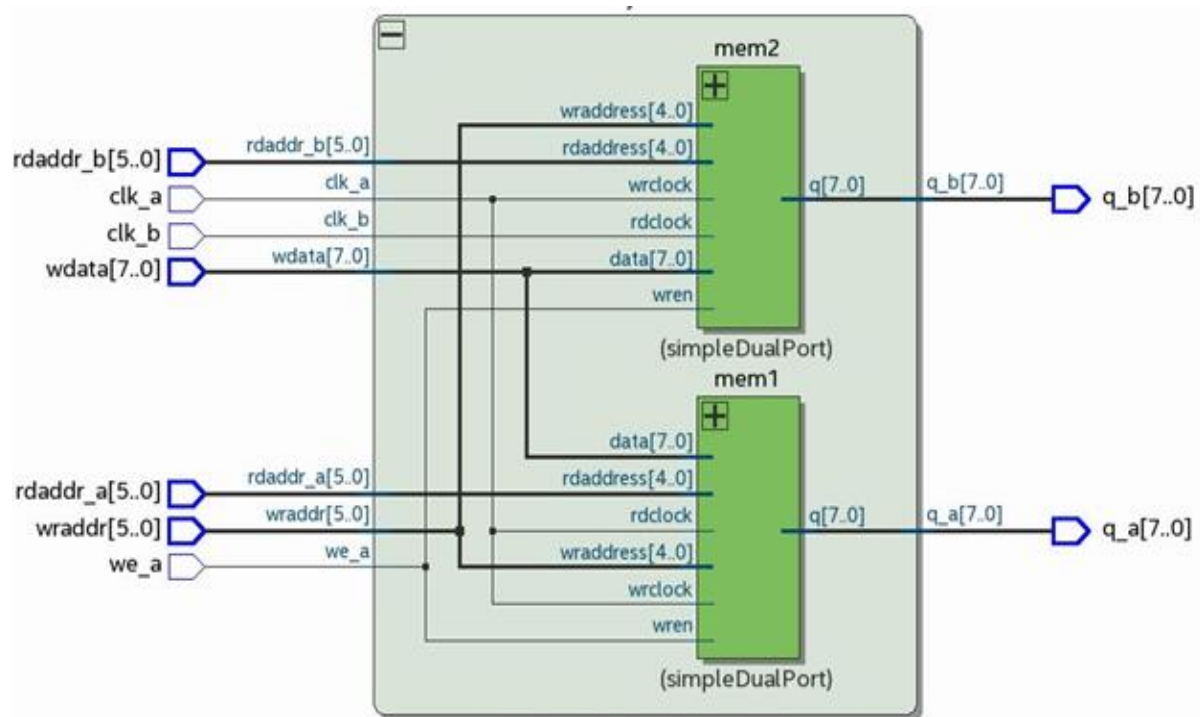


Fig:4 Black Box view of Dual port RAM

3.RESULTS AND DISCUSSION

3.1. SIMULATIONS:

READ & WRITE OPERATIONS:

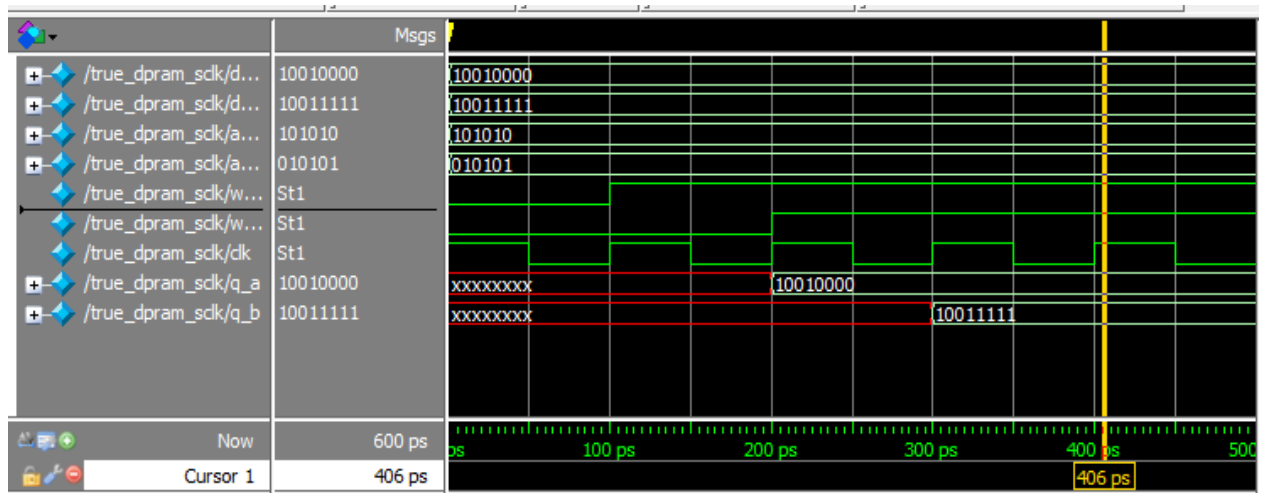


Fig:5 Simulation of DPRAM

The above simulation graph of “true_dpram_sclk”
at

100 ps	we_a “0”, we_b “0”, clk “high”,	simultaneous read operation.
200 ps	we_a “1”, we_b “0”, clk “high”,	simultaneous read & write operation.
300 ps	we_a “0”, we_b “0”, clk “high”,	simultaneous write operation.

READ WHILE WRITE OR WRITE WHILE READING:

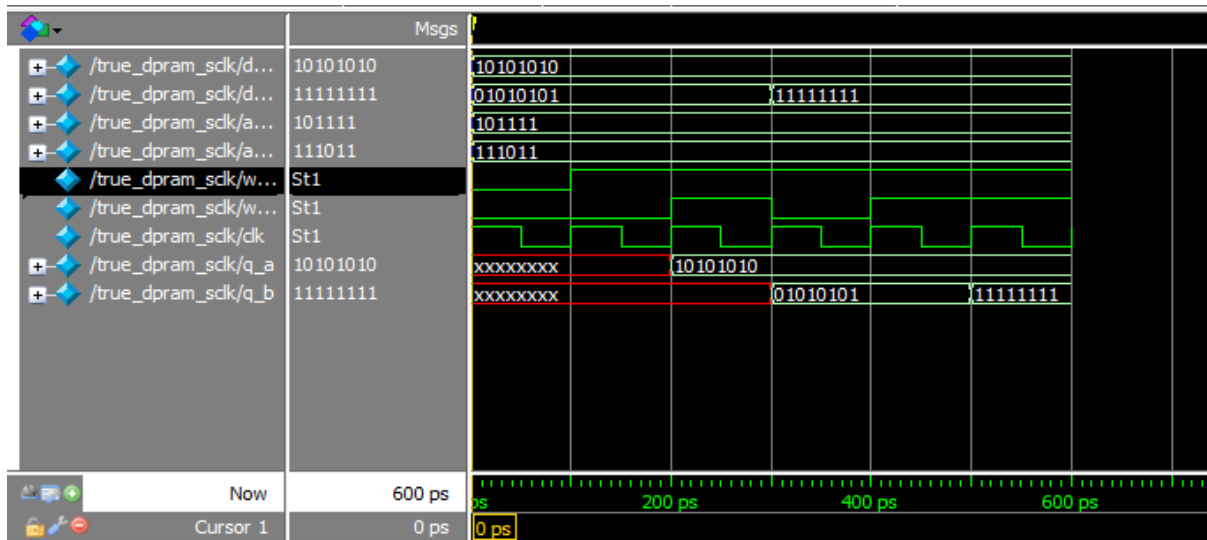


Fig:5 Simulation of DPRAM

The above simulation graph of “true_dpram_sclk”

at

100 ps	we_a “0”, we_b “0”, clk “high”,	simultaneous read operations.
200 ps	we_a “1”, we_b “0”, clk “high”,	simultaneous write while read operation.
300 ps	we_a “0”, we_b “0”, clk “high”,	simultaneous write operation.
400 ps	we_a “1”, we_b “0”, clk “high”,	simultaneous write while read operation(port b).

Here at 400 ps port B write the previous data “01010101” and read the new data “11111111”.

3.2. VERILOG CODE FOR DPRAM DESIGN:

```
module true_dpram_sclk
(
input [7:0] data_a, data_b,
input [5:0] addr_a, addr_b,
input we_a, we_b, clk,
output reg [7:0] q_a, q_b
);
// Declaration of the RAM variable
reg [7:0] ram[63:0];

// Port A
always @ (posedge clk)
begin
if (we_a)
begin
ram[addr_a] <= data_a;
q_a <= data_a;
end
else
begin
q_a <= ram[addr_a];
end
end

// Port B
always @ (posedge clk)
begin
if (we_b)
begin
ram[addr_b] <= data_b;
q_b <= data_b;
end
else
begin
q_b <= ram[addr_b];
end
end

endmodule
```

3.3. RTL SCHEMATICS:

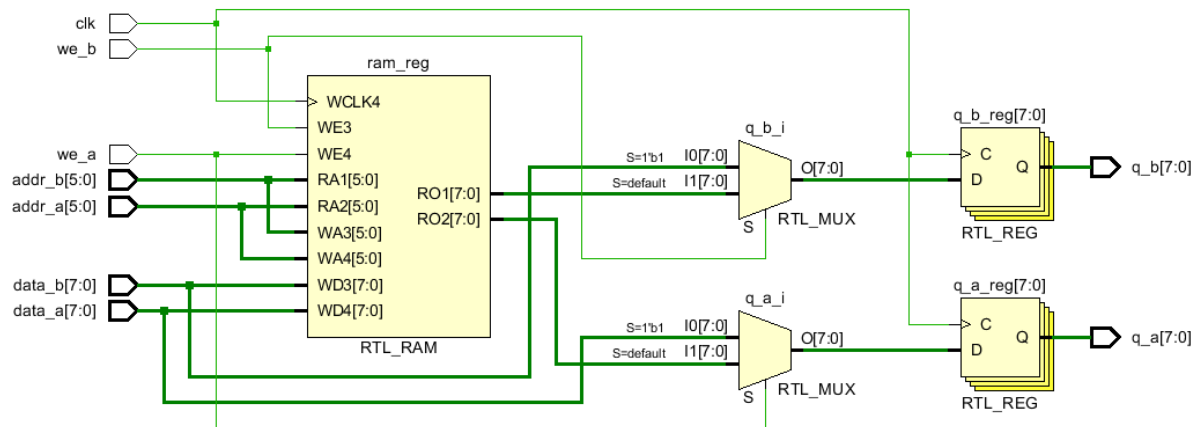


Fig7: RTL schematic of the top dpram.

3.4. SYNTHESIS REPORT:

Copyright 1986-2019 Xilinx, Inc. All Rights Reserved.

| Tool Version : Vivado v.2019.1 (win64) Build 2552052 Fri May 24 14:49:42 MDT 2019
 | Date : Thu Jul 11 16:36:06 2019
 | Host : SREENIVAS running 64-bit major release (build 9200)
 | Command : report_utilization -file true_dpram_sclk_utilization_synth.rpt -pb
 true_dpram_sclk_utilization_synth.pb
 | Design : true_dpram_sclk
 | Device : 7s6ftgb196-1
 | Design State : Synthesized

Utilization Design Information

Table of Contents

-
- 1. Slice Logic
 - 1.1 Summary of Registers by Type
 - 2. Memory
 - 3. DSP
 - 4. IO and GT Specific
 - 5. Clocking

6. Specific Feature

7. Primitives

8. Black Boxes

9. Instantiated Netlists

1. Slice Logic

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	0	0	3750	0.00
LUT as Logic	0	0	3750	0.00
LUT as Memory	0	0	2400	0.00
Slice Registers	0	0	7500	0.00
Register as Flip Flop	0	0	7500	0.00
Register as Latch	0	0	7500	0.00
F7 Muxes	0	0	4000	0.00
F8 Muxes	0	0	2000	0.00

* Warning! The Final LUT count, after physical optimizations and full implementation, is typically lower. Run opt_design after synthesis, if not already completed, for a more realistic count.

1.1 Summary of Registers by Type

Total	Clock Enable	Synchronous	Asynchronous
0	-	-	-
0	-	-	Set
0	-	-	Reset
0	-	Set	-
0	-	Reset	-
0	Yes	-	-
0	Yes	-	Set
0	Yes	-	Reset
0	Yes	Set	-
0	Yes	Reset	-

2. Memory

Site Type	Used	Fixed	Available	Util%
Block RAM Tile	0.5	0	5	10.00
RAMB36/FIFO*	0	0	5	0.00
RAMB18	1	0	10	10.00
RAMB18E1 only	1			

* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Block RAM Tile, that tile can still accommodate a RAMB18E1

3. DSP

Site Type	Used	Fixed	Available	Util%
DSPs	0	0	10	0.00

4. IO and GT Specific

Site Type	Used	Fixed	Available	Util%
Bonded IOB	47	0	100	47.00
PHY_CONTROL	0	0	2	0.00
PHASER_REF	0	0	2	0.00
OUT_FIFO	0	0	8	0.00
IN_FIFO	0	0	8	0.00
IDELAYCTRL	0	0	2	0.00
IBUFDS	0	0	96	0.00
PHASER_OUT/PHASER_OUT_PHY	0	0	8	0.00
PHASER_IN/PHASER_IN_PHY	0	0	8	0.00
IDELAYE2/IDELAYE2_FINEDELAY	0	0	100	0.00
ILOGIC	0	0	100	0.00
OLOGIC	0	0	100	0.00

5. Clocking

Site Type	Used	Fixed	Available	Util%
BUFGCTRL	1	0	16	6.25
BUFIO	0	0	8	0.00
MMCME2_ADV	0	0	2	0.00
PLLE2_ADV	0	0	2	0.00
BUFMRCE	0	0	4	0.00
BUFHCE	0	0	24	0.00
BUFR	0	0	8	0.00

6. Specific Feature

Site Type	Used	Fixed	Available	Util%
BSCANE2	0	0	4	0.00
CAPTUREE2	0	0	1	0.00
DNA_PORT	0	0	1	0.00
EFUSE_USR	0	0	1	0.00
FRAME_ECCE2	0	0	1	0.00
ICAPE2	0	0	2	0.00
STARTUPE2	0	0	1	0.00

7. Primitives

Ref Name	Used	Functional Category
IBUF	31	IO
OBUF	16	IO
RAMB18E1	1	Block Memory
BUFG	1	Clock

8. Black Boxes

```
+-----+-----+
| Ref Name | Used |
+-----+-----+
```

9. Instantiated Netlists

```
+-----+-----+
| Ref Name | Used |
+-----+-----+
```

Start RTL Component Statistics

Detailed RTL Component Info :

+---Registers :

8 Bit Registers := 2

+---RAMs :

512 Bit RAMs := 1

Finished RTL Component Statistics

Start RTL Hierarchical Component Statistics

Hierarchical RTL Component report

Module true_dpram_sclk

Detailed RTL Component Info :

+---Registers :

8 Bit Registers := 2

+---RAMs :

512 Bit RAMs := 1

Finished RTL Hierarchical Component Statistics

Start Part Resource Summary

Part Resources:

DSPs: 10 (col length:20)

BRAMs: 10 (col length: RAMB18 20 RAMB36 10)

Finished Part Resource Summary

Start Cross Boundary and Area Optimization

Warning: Parallel synthesis criteria is not met

INFO: [Synth 8-3971] The signal "true_dpram_sclk/ram_reg" was recognized as a true dual port RAM template.

Finished Cross Boundary and Area Optimization : Time (s): cpu = 00:00:22 ; elapsed = 00:00:34 . Memory (MB): peak = 756.609 ; gain = 292.199

Start ROM, RAM, DSP and Shift Register Reporting

Block RAM: Preliminary Mapping Report (see note below)

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
+-----+-----+									
Module Name		RTL Object		PORT A (Depth x Width)		W R		PORT B (Depth x Width)	
W R		Ports driving FF		RAMB18		RAMB36			
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
+-----+-----+									
true_dpram_sclk		ram_reg		64 x 8(WRITE_FIRST)		W R		64 x 8(WRITE_FIRST)	
R		Port A and B		1 0					
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
+-----+-----+									

Note: The table above is a preliminary report that shows the Block RAMs at the current stage of the synthesis flow. Some Block RAMs may be reimplemented as non Block RAM primitives later in the synthesis flow. Multiple instantiated Block RAMs are reported only once.

Finished ROM, RAM, DSP and Shift Register Reporting

3.5. POWER REPORT

Table of Contents

- 1. Summary
 - 1.1 On-Chip Components
 - 1.2 Power Supply Summary
 - 1.3 Confidence Level
- 2. Settings
 - 2.1 Environment
 - 2.2 Clock Constraints
- 3. Detailed Reports
 - 3.1 By Hierarchy

1. Summary

+-----+-----+			
Total On-Chip Power (W)	0.095		
Design Power Budget (W)	Unspecified*		
Power Budget Margin (W)	NA		
Dynamic (W)	0.014		
Device Static (W)	0.081		
Effective TJA (C/W)	1.9		
Max Ambient (C)	84.8		
Junction Temperature (C)	25.2		
Confidence Level	Low		
Setting File	---		
Simulation Activity File	---		
Design Nets Matched	NA		
+-----+-----+			

* Specify Design Power Budget using, set_operating_conditions -design_power_budget <value in Watts>

1.1 On-Chip Components

+-----+-----+-----+-----+-----+					
On-Chip	Power (W)	Used	Available	Utilization (%)	
+-----+-----+-----+-----+-----+					
Clocks	<0.001	4	---	---	
Slice Logic	0.000	2	---	---	
Others	0.000	2	---	---	

Signals		<0.001		46		---		---	
Block RAM		0.002		0.5		135		0.37	
I/O		0.010		47		300		15.67	
Static Power		0.081							
Total		0.095							
+-----+-----+-----+-----+-----+									

1.2 Power Supply Summary

+-----+-----+-----+-----+-----+									
Source		Voltage (V)		Total (A)		Dynamic (A)		Static (A)	
+-----+-----+-----+-----+-----+									
Vccint		1.000		0.025		0.003		0.022	
Vccaux		1.800		0.013		0.001		0.012	
Vcco33		3.300		0.000		0.000		0.000	
Vcco25		2.500		0.000		0.000		0.000	
Vcco18		1.800		0.006		0.005		0.001	
Vcco15		1.500		0.000		0.000		0.000	
Vcco135		1.350		0.000		0.000		0.000	
Vcco12		1.200		0.000		0.000		0.000	
Vccaux_io		1.800		0.000		0.000		0.000	
Vccbram		1.000		0.001		0.000		0.000	
MGTAVcc		1.000		0.000		0.000		0.000	
MGTAVtt		1.200		0.000		0.000		0.000	
MGTVccaux		1.800		0.000		0.000		0.000	
Vccadc		1.800		0.020		0.000		0.020	
+-----+-----+-----+-----+-----+									

2. Settings

2.1 Environment

+-----+-----+-----+-----+-----+									
Ambient Temp (C)		25.0							
ThetaJA (C/W)		1.9							
Airflow (LFM)		250							
Heat Sink		medium (Medium Profile)							
ThetaSA (C/W)		3.4							
Board Selection		medium (10"x10")							
# of Board Layers		12to15 (12 to 15 Layers)							
Board Temperature (C)		25.0							

+-----+-----+

2.2 Clock Constraints

+-----+-----+

Clock	Domain	Constraint (ns)
-------	--------	-----------------

+-----+-----+

UART	clk	10.0
------	-----	------

+-----+-----+

3. Detailed Reports

3.1 By Hierarchy

+-----+-----+

Name	Power (W)
------	-----------

+-----+-----+

DPRAM	0.014
-------	-------

+-----+-----+

CONCLUSION AND RECOMMENDATIONS:

- The objective of this project is accomplished i.e. the design of DPRAM is studied in a detailed manner and written Verilog hardware description language and simulated in ModelSim-Altera 6.6d and verified it's a simulated and the obtained results are well appreciable.
- This project implementation is done for Spartan 7 FPGA implementation since Vivado tool is capable of implement bit stream on this board. We can use other tools also like Xilinx ISE tool for other board implementations.

APPENDIX 1: LIST OF TABLES AND FIGURES

Fig:1 Read and write schematics.

Fig:2 Write operation triggering at posedge of clk

Fig:3 Write operation triggering at negedge of clk

Fig:4 Black Box view of Dual port RAM

Fig:5 Simulation of DPRAM

Fig:6 Simulation of DPRAM

Fig7: RTL schematic of the top dpram.

Table :1 Different Modes of RAM

APPENDIX 2: REFERENCES

- [1]<http://pgandhi189.blogspot.com/2014/11/dual-port-ram-implementation-in-verilog.html>
- [2]<https://anysilicon.com/dual-port-ram-design-bit-design-goals/>
- [3]<https://www.edaboard.com/showthread.php?142409-read-write-from-dual-port-ram>
- [4]https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_ram_rom.pdf
- [5] Xilinx Vivado Video tutorials.
- [6] Reference Material is given by SION SEMICONDUCTORS.

PROJECT 3: SYNCHRONOUS FIFO

1. INTRODUCTION

1.1. PROBLEM STATEMENT AND OBJECTIVE:

We have mainly four type of replacement algorithms are LRU, FIFO, LFU, Random Algorithm. In these LRU has the highest priority. We are taking FIFO as our project. This project is focusing on whether the data getting to the destination is the same which is given first at source. We need to check whether the operation is correct depending on enabling of control signals. First Input First Output project is used for developing the access time of data blocks. Control logic manages read and write operations. FIFO is mostly used in control the flow of data. We have synchronous and asynchronous FIFO's depending on whether same or different clock signal to control the read and write operations. In this project the aim is to design and verify the synchronous FIFO using write and read pointers (binary). We use 'ModelSIM Altera Starter' to design and verify. We use Verilog VHDL as RTL description.

1.2. MOTIVATION:

VLSI has more scope in future as we can reduce the chip size and increases the speed of circuitry. We designed the digital logics circuits in 3rd semester using Verilog in 'XILINX'. We studied basics of FIFO and LRU's in 4th semester, so I got the interest to design a FIFO using Verilog. FIFO can be used to match the throughputs of the Source and the Requestor.

1.3. METHOD:

We will mainly divide FIFO into 3 blocks: memory block, read logic signal and write logic signal. Memory block is an array of flipflops. Number of data words that the memory array can store is often named as data depth of the FIFO. Length of the data word is named as data width of the FIFO. If write enable signal is high data present on write data is written into the row pointed by write pointer on the next rising edge of the clock signal clk. Similarly read logic. We have syntax for defining memory as `reg [wordsize:0] array name [0: array size]`. We also have status registers FULL and EMPTY.

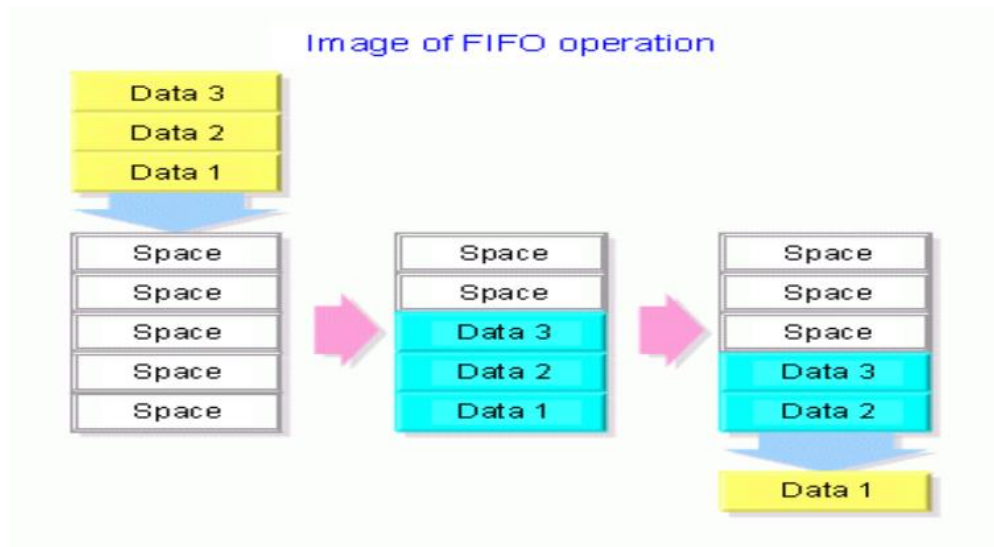


Fig.1: Basic FIFO Operation

1.4. REQUIREMENTS:

We use modelsim ALTERA STARTER to write rtl code and we compile and simulate the Verilog code in that. For implementation we use vivado Xilinx.

1.5. ASSUMPTIONS:

The only restriction we kept is that the simultaneous read and write logic shouldn't given to a particular memory location. For write enable high and read enable low it works as $\text{count} = \text{count} + 1$, and for write enable low and read enable high $\text{count} = \text{count} - 1$.

1.6. LITERATURE RESEARCH:

Harish sharma and Charu rana in International Journal of Computer Applications (0975 – 8887) Volume 63– No.16, February 2013.describes "Designing of 8-bit Synchronous FIFO Memory using Register File".

In this report their aim was to design the fifo using register file instead of Ram. They built code with basic architecture of register file.

2. PROJECT METHODOLOGY

A synchronous FIFO refers to a FIFO design where data values are written sequentially into a memory array using a clock signal, and the data values are sequentially read out from the memory array using the same clock signal.

2.1. BLACK BOX VIEW OF FIFO:

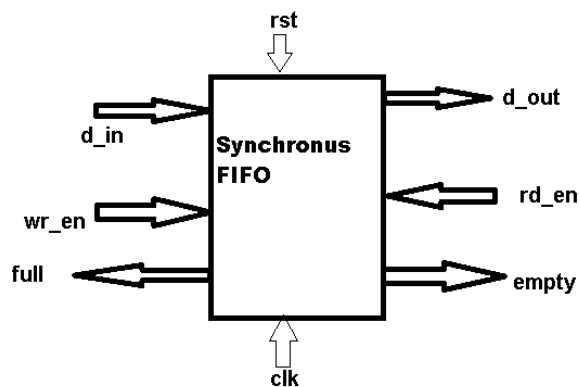


Fig2: Black box of FIFO

PORT LIST:

NAME	I/O	DESCRIPTION
clk	Input	Clk input to the fifo
rst	Input	Active high input to fifo
wr_en	Input	Write select signal for the fifo
rd_en	Input	Input signal to read the data from fifo
d_in	Input	Data input port(source)
d_out	Output	Destination port
Full	Output	It provides the information regarding status of fifo
Empty	Output	It provides the information regarding status of fifo

Table 1: Port list

2.2. WORKING OF FIFO:

The Synchronous FIFO has a single port of clk for both read and write operation. When `wr_en` signal is high the data present in the `d_in` port will be written in the next empty location of memory on next rising edge (posedge). When the full output port is high it means there is no empty memory locations. So we can't write the data until the data is read. When the Empty status port has high value we can't read the data from the memory until it changes to low (writing the data). We will declare the internal variables such as read and write pointers. For a reset high value the pointers go to zero. The operations we discussed above are going to happen only if the value of reset is low. We can increase and decrease the depth and width of the data easily by parameterization. We use depth count as a variable to know the status of the memory. If depth count is equal to depth of the data then full status register shows high and for depth count is equal to zero empty status register shows high value. The value of depth count varies depending upon read and write input signals. The Verilog code is attached at the last in appendix. We can easily understand by the following fig

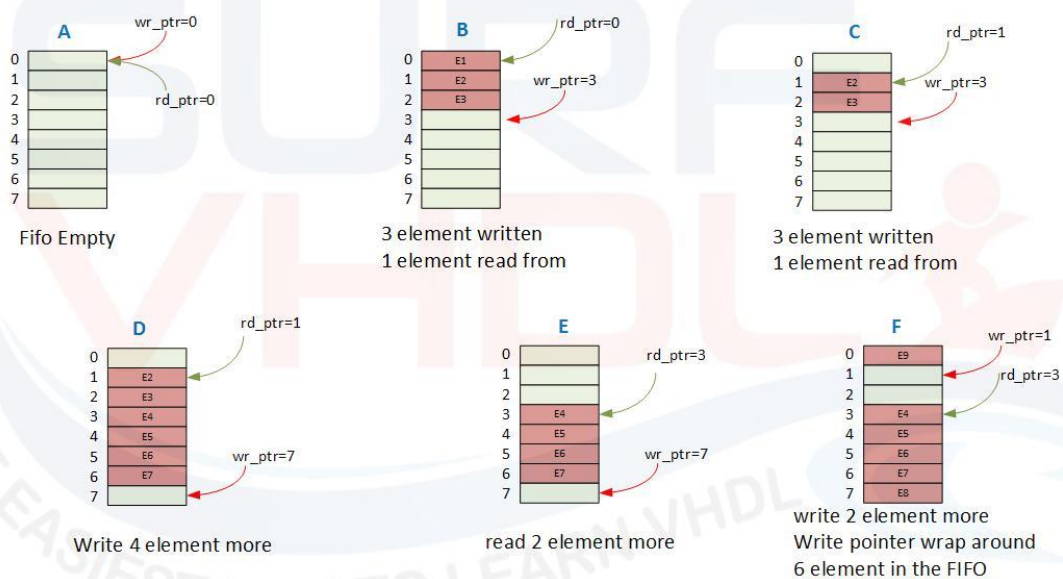


Fig3 : Working of Pointers

We use different always blocks to write a code so that the production of extra latches will decrease. This is the typical industrial method. It is also easy to understand the code.

3. RESULTS AND DISCUSSION

3.1.SIMULATIONS:

For reset high the values of read and write pointers and depth count goes to zero irrespective of read and write enable signals.

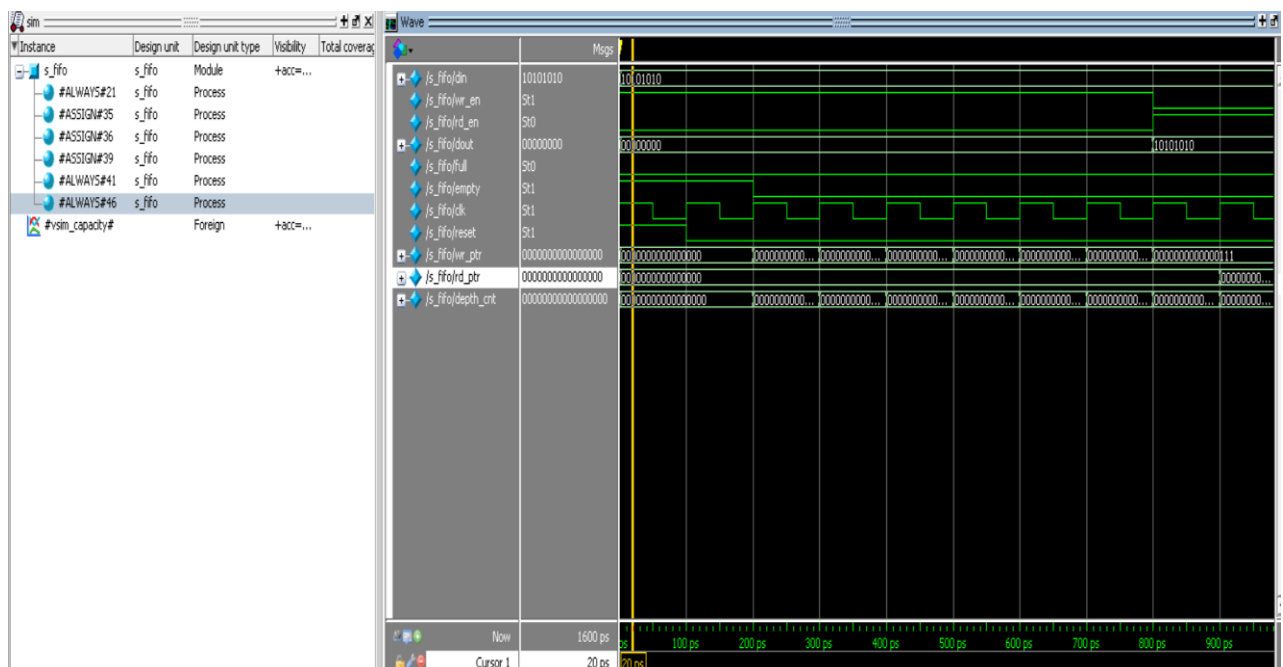


Fig4: Simulation waveform when reset goes to high

For reset value low and write enable high both depth count and write pointers varies(increases)

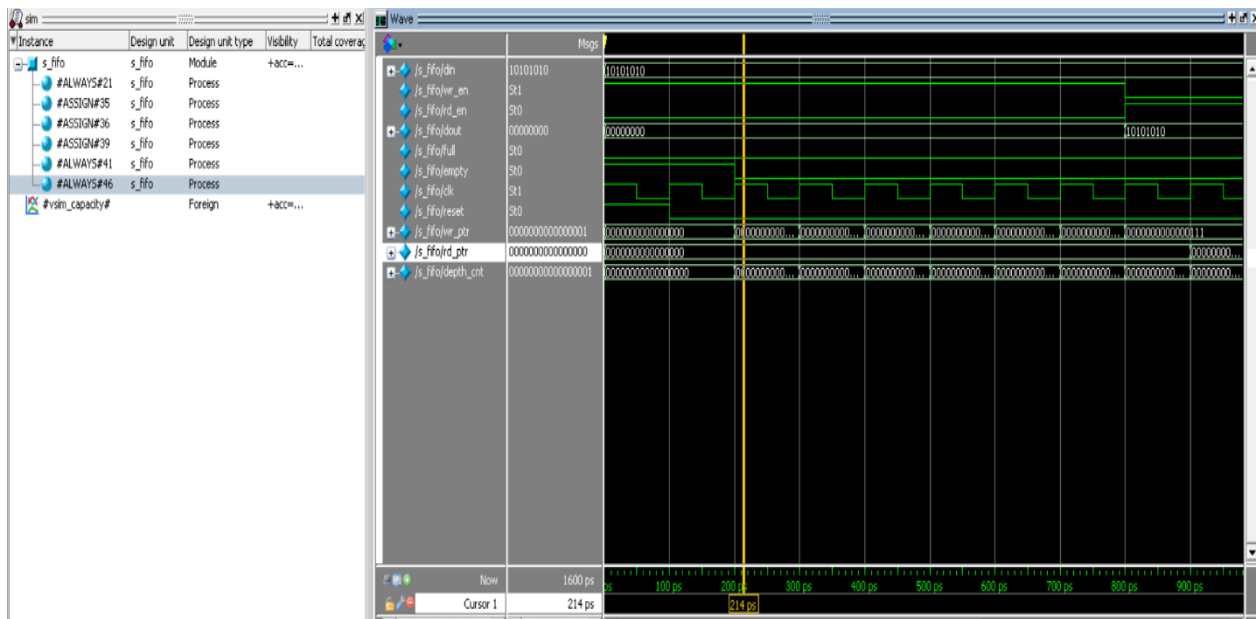


Fig 5: Waveform when reset triggered to zero and write enable to high

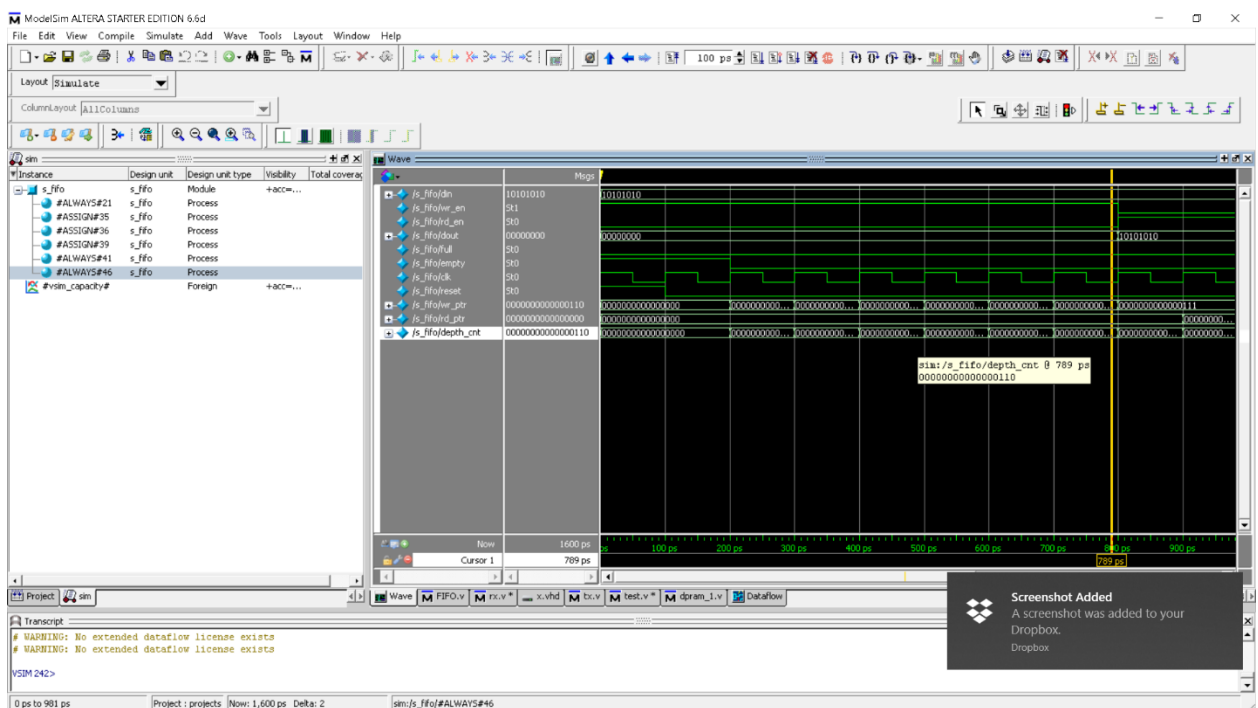


Fig 6

To read the data from memory write enable should turn to low value and read enable should be high. As we discussed in methodology the depth count will start decrease until the status empty shows high and the data can be read from the d_out port

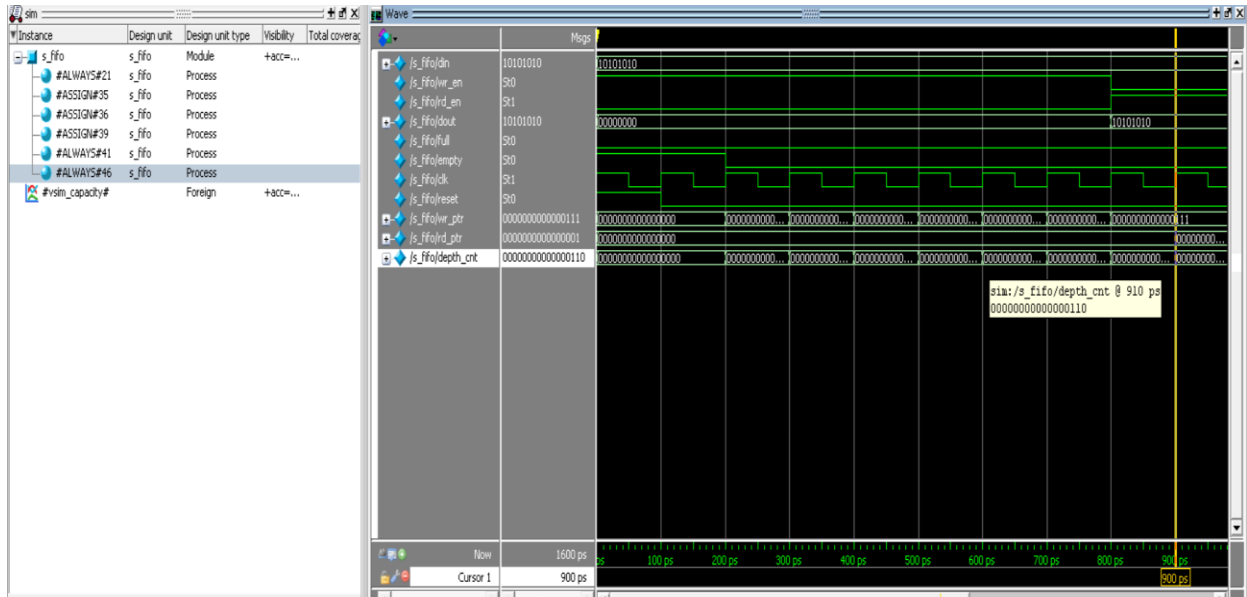
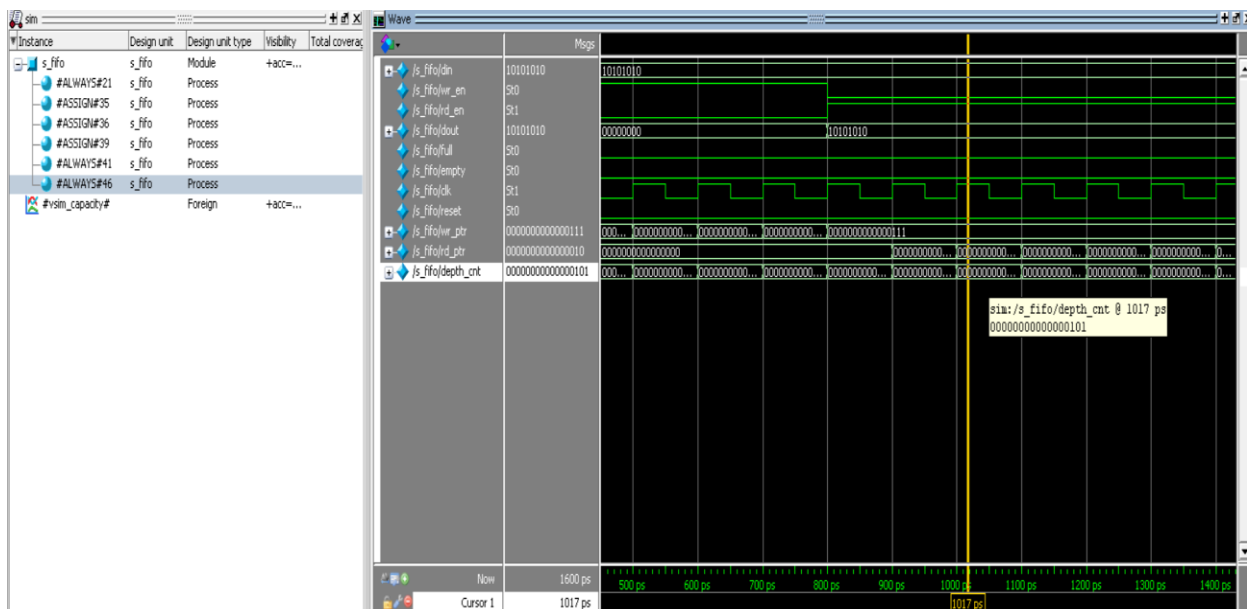


Fig 7: Reading the data



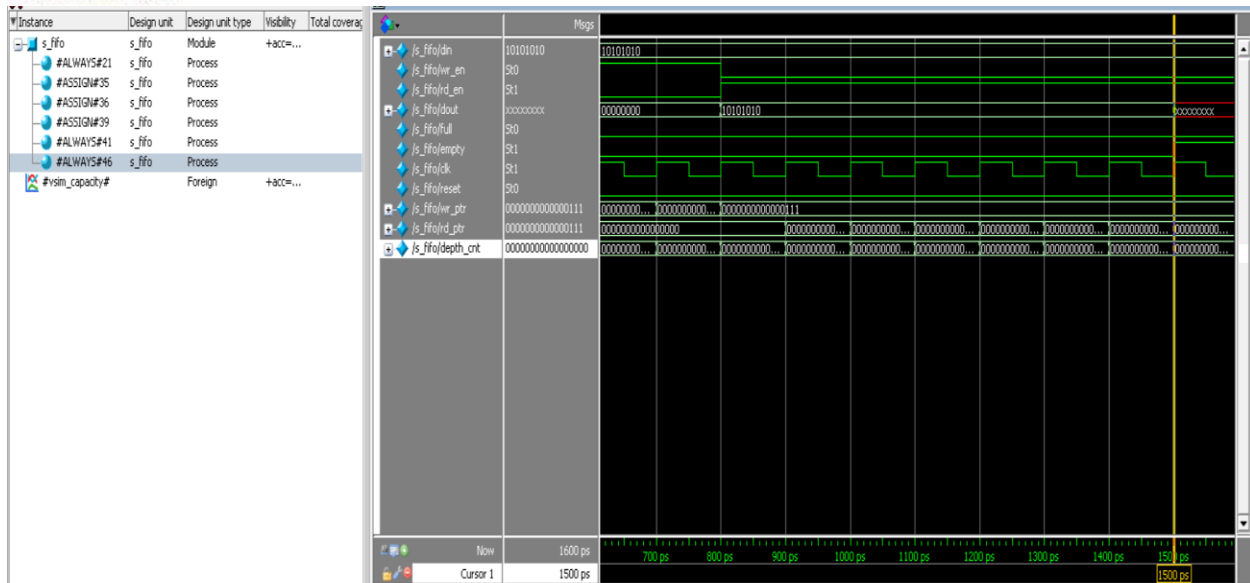


Fig 9: when we have no data in memory

If we read the data from memory after the empty status register having high value we will get don't cares's as an output.

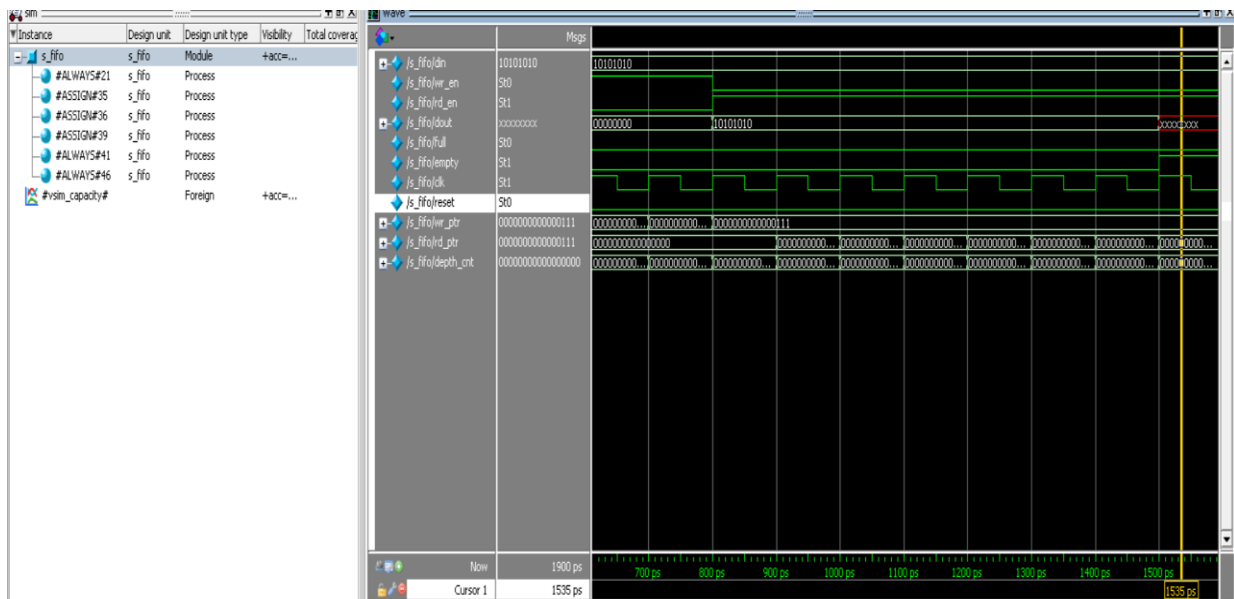


Fig 10: Getting output as don't care's

3.2. VERILOG CODE:

```
module s_fifo
(
    input [DATA_WIDTH-1:0] d_in,
    input wr_en,
    input rd_en,
    output [DATA_WIDTH-1:0] d_out,
    output full,
    output empty,
    input clk,
    input reset
);
(
    parameter DATA_WIDTH = 8,
    parameter DATA_DEPTH = 16
)
// Declaring internal variables
reg [DATA_DEPTH-1 : 0] rd_ptr;
reg [DATA_DEPTH-1 : 0] wr_ptr;
reg [DATA_WIDTH-1 : 0] mem[DATA_DEPTH-1 : 0]; // defining memory sram
reg [DATA_DEPTH : 0] depth_cnt;

always @(posedge clk) begin
    if(reset==1)
        begin
            wr_ptr <= 'h0;
            rd_ptr <= 'h0;
        end // end if
    else
        begin
            if(wr_en==1)
                begin
                    wr_ptr <= wr_ptr+1;
                end
            if(rd_en==1)
                rd_ptr <= rd_ptr+1;
        end //end else
    end//end always
```

```
assign empty= (depth_cnt=='h0); // status registers
```

```
assign full = (depth_cnt==DATA_DEPTH);
```

```
assign d_out = rd_en ? mem[rd_ptr]:'h0; // reading the data from memory and getting  
through d_out
```

```
always @(posedge clk) begin  
    if (wr_en==1)  
        mem[wr_ptr] <= d_in;  
end //end always
```

```
always @(posedge clk) begin  
    if (reset)  
        depth_cnt <= 'h0;  
    else begin  
        case({rd_en,wr_en})  
            2'b10 : depth_cnt <= depth_cnt-1;  
            2'b01 : depth_cnt <= depth_cnt+1;  
        endcase  
    end //end else  
end //end always
```

```
endmodule // end fifo
```

3.3. RTL SCHEMATICS:

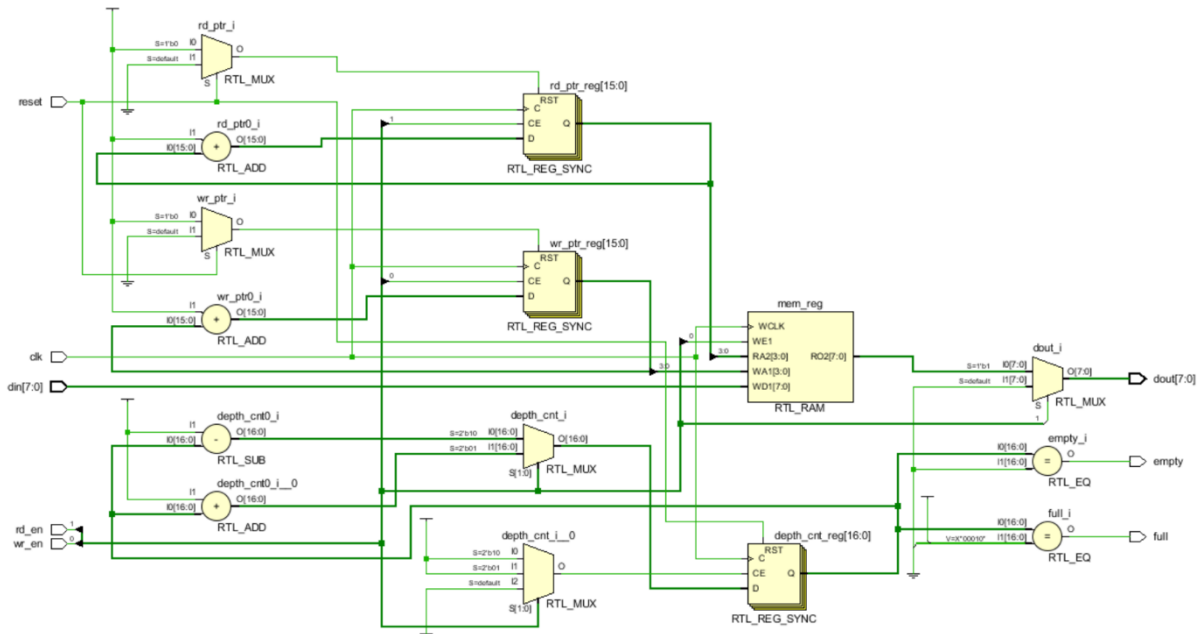


Fig11 : RTL schematics of FIFO

Post place and route simulation:

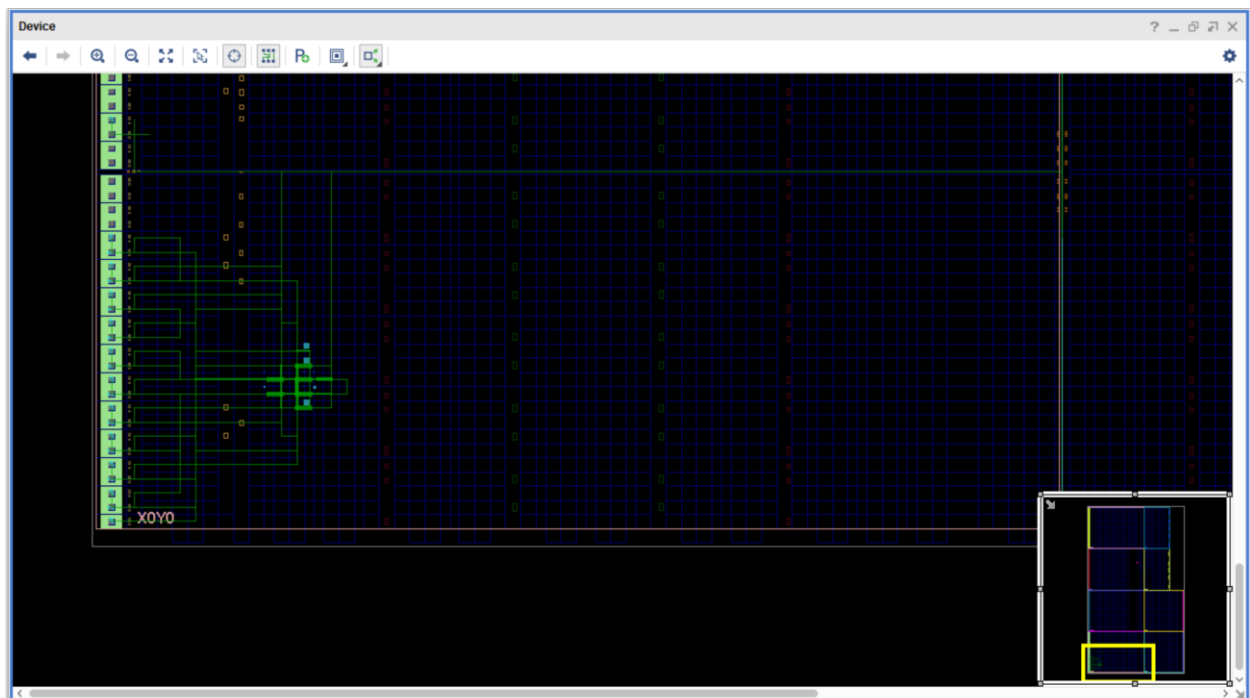


Fig 12: P&R in implementation

3.4. UTILIZATION REPORT:

Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

```
-----
| Tool Version : Vivado v.2018.2 (win64) Build 2258646 Thu Jun 14 20:03:12 MDT 2018
| Date       : Thu Jul 11 17:33:33 2019
| Host       : HK007 running 64-bit major release (build 9200)
| Command    : report_utilization -file sync_fifo_utilization_placed.rpt -pb
sync_fifo_utilization_placed.pb
| Design     : sync_fifo
| Device     : 7k70tfbv676-1
| Design State : Fully Placed
-----
```

Utilization Design Information

Table of Contents

- ```

1. Slice Logic
1.1 Summary of Registers by Type
2. Slice Logic Distribution
3. Memory
4. DSP
5. IO and GT Specific
6. Clocking
7. Specific Feature
8. Primitives
9. Black Boxes
10. Instantiated Netlists
```

##### 1. Slice Logic

```

```

| Site Type              | Used | Fixed | Available | Util% |
|------------------------|------|-------|-----------|-------|
| Slice LUTs             | 40   | 0     | 41000     | 0.10  |
| LUT as Logic           | 32   | 0     | 41000     | 0.08  |
| LUT as Memory          | 8    | 0     | 13400     | 0.06  |
| LUT as Distributed RAM | 8    | 0     |           |       |
| LUT as Shift Register  | 0    | 0     |           |       |
| Slice Registers        | 29   | 0     | 82000     | 0.04  |
| Register as Flip Flop  | 29   | 0     | 82000     | 0.04  |
| Register as Latch      | 0    | 0     | 82000     | 0.00  |
| F7 Muxes               | 0    | 0     | 20500     | 0.00  |

```

```

|          |   |   |       |      |
|----------|---|---|-------|------|
| F8 Muxes | 0 | 0 | 10250 | 0.00 |
| +-----+  |   |   |       |      |

## 1.1 Summary of Registers by Type

| +-----+ |              |             |              |  |
|---------|--------------|-------------|--------------|--|
| Total   | Clock Enable | Synchronous | Asynchronous |  |
| +-----+ |              |             |              |  |
| 0       | -            | -           | -            |  |
| 0       | -            | -           | Set          |  |
| 0       | -            | -           | Reset        |  |
| 0       | -            | Set         | -            |  |
| 0       | -            | Reset       | -            |  |
| 0       | Yes          | -           | -            |  |
| 0       | Yes          | -           | Set          |  |
| 0       | Yes          | -           | Reset        |  |
| 0       | Yes          | Set         | -            |  |
| 29      | Yes          | Reset       | -            |  |
| +-----+ |              |             |              |  |

## 2. Slice Logic Distribution

| +-----+                                 |      |       |           |       |
|-----------------------------------------|------|-------|-----------|-------|
| Site Type                               | Used | Fixed | Available | Util% |
| +-----+                                 |      |       |           |       |
| Slice                                   | 15   | 0     | 10250     | 0.15  |
| SLICEL                                  | 11   | 0     |           |       |
| SLICEM                                  | 4    | 0     |           |       |
| LUT as Logic                            | 32   | 0     | 41000     | 0.08  |
| using O5 output only                    | 0    |       |           |       |
| using O6 output only                    | 28   |       |           |       |
| using O5 and O6                         | 4    |       |           |       |
| LUT as Memory                           | 8    | 0     | 13400     | 0.06  |
| LUT as Distributed RAM                  | 8    | 0     |           |       |
| using O5 output only                    | 0    |       |           |       |
| using O6 output only                    | 0    |       |           |       |
| using O5 and O6                         | 8    |       |           |       |
| LUT as Shift Register                   | 0    | 0     |           |       |
| LUT Flip Flop Pairs                     | 20   | 0     | 41000     | 0.05  |
| fully used LUT-FF pairs                 | 0    |       |           |       |
| LUT-FF pairs with one unused LUT output | 20   |       |           |       |
| LUT-FF pairs with one unused Flip Flop  | 20   |       |           |       |

|                     |   |  |  |  |
|---------------------|---|--|--|--|
| Unique Control Sets | 4 |  |  |  |
|---------------------|---|--|--|--|

\* Note: Review the Control Sets Report for more information regarding control sets.

### 3. Memory

| Site Type      | Used | Fixed | Available | Util% |
|----------------|------|-------|-----------|-------|
| Block RAM Tile | 0    | 0     | 135       | 0.00  |
| RAMB36/FIFO*   | 0    | 0     | 135       | 0.00  |
| RAMB18         | 0    | 0     | 270       | 0.00  |

\* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Block RAM Tile, that tile can still accommodate a RAMB18E1

### 4. DSP

| Site Type | Used | Fixed | Available | Util% |
|-----------|------|-------|-----------|-------|
| DSPs      | 0    | 0     | 240       | 0.00  |

### 5. IO and GT Specific

| Site Type       | Used | Fixed | Available | Util% |
|-----------------|------|-------|-----------|-------|
| Bonded IOB      | 22   | 0     | 300       | 7.33  |
| IOB Master Pads | 11   |       |           |       |
| IOB Slave Pads  | 10   |       |           |       |
| Bonded IPADs    | 0    | 0     | 26        | 0.00  |
| Bonded OPADs    | 0    | 0     | 16        | 0.00  |
| PHY_CONTROL     | 0    | 0     | 6         | 0.00  |
| PHASER_REF      | 0    | 0     | 6         | 0.00  |
| OUT_FIFO        | 0    | 0     | 24        | 0.00  |
| IN_FIFO         | 0    | 0     | 24        | 0.00  |
| IDELAYCTRL      | 0    | 0     | 6         | 0.00  |

|                             |   |   |     |      |
|-----------------------------|---|---|-----|------|
| IBUFDS                      | 0 | 0 | 288 | 0.00 |
| GTXE2_COMMON                | 0 | 0 | 2   | 0.00 |
| GTXE2_CHANNEL               | 0 | 0 | 8   | 0.00 |
| PHASER_OUT/PHASER_OUT_PHY   | 0 | 0 | 24  | 0.00 |
| PHASER_IN/PHASER_IN_PHY     | 0 | 0 | 24  | 0.00 |
| IDELAYE2/IDELAYE2_FINEDELAY | 0 | 0 | 300 | 0.00 |
| ODELAYE2/ODELAYE2_FINEDELAY | 0 | 0 | 100 | 0.00 |
| IBUFDS_GTE2                 | 0 | 0 | 4   | 0.00 |
| ILOGIC                      | 0 | 0 | 300 | 0.00 |
| OLOGIC                      | 0 | 0 | 300 | 0.00 |
| +-----+-----+-----+-----+   |   |   |     |      |

## 6. Clocking

|                           |      |       |           |       |
|---------------------------|------|-------|-----------|-------|
| +-----+-----+-----+-----+ |      |       |           |       |
| Site Type                 | Used | Fixed | Available | Util% |
| +-----+-----+-----+-----+ |      |       |           |       |
| BUFGCTRL                  | 1    | 0     | 32        | 3.13  |
| BUFIO                     | 0    | 0     | 24        | 0.00  |
| MMCME2_ADV                | 0    | 0     | 6         | 0.00  |
| PLLE2_ADV                 | 0    | 0     | 6         | 0.00  |
| BUFMRCE                   | 0    | 0     | 12        | 0.00  |
| BUFHCE                    | 0    | 0     | 96        | 0.00  |
| BUFR                      | 0    | 0     | 24        | 0.00  |
| +-----+-----+-----+-----+ |      |       |           |       |

## 7. Specific Feature

|                           |      |       |           |       |
|---------------------------|------|-------|-----------|-------|
| +-----+-----+-----+-----+ |      |       |           |       |
| Site Type                 | Used | Fixed | Available | Util% |
| +-----+-----+-----+-----+ |      |       |           |       |
| BSCANE2                   | 0    | 0     | 4         | 0.00  |
| CAPTUREE2                 | 0    | 0     | 1         | 0.00  |
| DNA_PORT                  | 0    | 0     | 1         | 0.00  |
| EFUSE_USR                 | 0    | 0     | 1         | 0.00  |
| FRAME_ECCE2               | 0    | 0     | 1         | 0.00  |
| ICAPE2                    | 0    | 0     | 2         | 0.00  |
| PCIE_2_1                  | 0    | 0     | 1         | 0.00  |
| STARTUPE2                 | 0    | 0     | 1         | 0.00  |
| XADC                      | 0    | 0     | 1         | 0.00  |
| +-----+-----+-----+-----+ |      |       |           |       |

## 8. Primitives

-----

| Ref Name | Used | Functional Category |
|----------|------|---------------------|
| FDRE     | 29   | Flop & Latch        |
| LUT2     | 25   | LUT                 |
| RAMD32   | 12   | Distributed Memory  |
| IBUF     | 12   | IO                  |
| OBUF     | 10   | IO                  |
| CARRY4   | 8    | CarryLogic          |
| LUT1     | 5    | LUT                 |
| RAMS32   | 4    | Distributed Memory  |
| LUT6     | 4    | LUT                 |
| LUT4     | 2    | LUT                 |
| BUFG     | 1    | Clock               |

## 9. Black Boxes

-----

|   |          |   |       |   |
|---|----------|---|-------|---|
| + | -----    | + | ----- | + |
|   | Ref Name |   | Used  |   |
| + | -----    | + | ----- | + |

## 10. Instantiated Netlists

-----

|   |          |   |       |   |
|---|----------|---|-------|---|
| + | -----    | + | ----- | + |
|   | Ref Name |   | Used  |   |
| + | -----    | + | ----- | + |

[illegible]

Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.

```
| Tool Version : Vivado v.2018.2 (win64) Build 2258646 Thu Jun 14 20:03:12 MDT 2018
| Date : Thu Jul 11 17:32:39 2019
| Host : HK007 running 64-bit major release (build 9200)
| Command : report_utilization -file sync_fifo_utilization_synth.rpt -pb
sync_fifo_utilization_synth.pb
```



| Design : sync\_fifo  
| Device : 7k70tfbv676-1  
| Design State : Synthesized

---

## Utilization Design Information

### Table of Contents

---

1. Slice Logic
  - 1.1 Summary of Registers by Type
2. Memory
3. DSP
4. IO and GT Specific
5. Clocking
6. Specific Feature
7. Primitives
8. Black Boxes
9. Instantiated Netlists

### 1. Slice Logic

---

| Site Type              | Used | Fixed | Available | Util% |
|------------------------|------|-------|-----------|-------|
| Slice LUTs*            | 40   | 0     | 41000     | 0.10  |
| LUT as Logic           | 32   | 0     | 41000     | 0.08  |
| LUT as Memory          | 8    | 0     | 13400     | 0.06  |
| LUT as Distributed RAM | 8    | 0     |           |       |
| LUT as Shift Register  | 0    | 0     |           |       |
| Slice Registers        | 29   | 0     | 82000     | 0.04  |
| Register as Flip Flop  | 29   | 0     | 82000     | 0.04  |
| Register as Latch      | 0    | 0     | 82000     | 0.00  |
| F7 Muxes               | 0    | 0     | 20500     | 0.00  |
| F8 Muxes               | 0    | 0     | 10250     | 0.00  |

\* Warning! The Final LUT count, after physical optimizations and full implementation, is typically lower. Run opt\_design after synthesis, if not already completed, for a more realistic count.

### 1.1 Summary of Registers by Type

---



---

| Total | Clock Enable | Synchronous | Asynchronous |

| +-----+ |     |       |       |  |
|---------|-----|-------|-------|--|
| 0       | _   | -     | -     |  |
| 0       | _   | -     | Set   |  |
| 0       | _   | -     | Reset |  |
| 0       | _   | Set   | -     |  |
| 0       | _   | Reset | -     |  |
| 0       | Yes | -     | -     |  |
| 0       | Yes | -     | Set   |  |
| 0       | Yes | -     | Reset |  |
| 0       | Yes | Set   | -     |  |
| 29      | Yes | Reset | -     |  |
| +-----+ |     |       |       |  |

## 2. Memory

-----

| +-----+        |      |       |           |       |
|----------------|------|-------|-----------|-------|
| Site Type      | Used | Fixed | Available | Util% |
| +-----+        |      |       |           |       |
| Block RAM Tile | 0    | 0     | 135       | 0.00  |
| RAMB36/FIFO*   | 0    | 0     | 135       | 0.00  |
| RAMB18         | 0    | 0     | 270       | 0.00  |
| +-----+        |      |       |           |       |

\* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Block RAM Tile, that tile can still accommodate a RAMB18E1

## 3. DSP

-----

| +-----+   |      |       |           |       |
|-----------|------|-------|-----------|-------|
| Site Type | Used | Fixed | Available | Util% |
| +-----+   |      |       |           |       |
| DSPs      | 0    | 0     | 240       | 0.00  |
| +-----+   |      |       |           |       |

## 4. IO and GT Specific

-----

| +-----+   |      |       |           |       |
|-----------|------|-------|-----------|-------|
| Site Type | Used | Fixed | Available | Util% |
| +-----+   |      |       |           |       |

|                             |    |   |     |      |
|-----------------------------|----|---|-----|------|
| Bonded IOB                  | 22 | 0 | 300 | 7.33 |
| Bonded IPADs                | 0  | 0 | 26  | 0.00 |
| Bonded OPADs                | 0  | 0 | 16  | 0.00 |
| PHY_CONTROL                 | 0  | 0 | 6   | 0.00 |
| PHASER_REF                  | 0  | 0 | 6   | 0.00 |
| OUT_FIFO                    | 0  | 0 | 24  | 0.00 |
| IN_FIFO                     | 0  | 0 | 24  | 0.00 |
| IDELAYCTRL                  | 0  | 0 | 6   | 0.00 |
| IBUFDS                      | 0  | 0 | 288 | 0.00 |
| GTXE2_COMMON                | 0  | 0 | 2   | 0.00 |
| GTXE2_CHANNEL               | 0  | 0 | 8   | 0.00 |
| PHASER_OUT/PHASER_OUT_PHY   | 0  | 0 | 24  | 0.00 |
| PHASER_IN/PHASER_IN_PHY     | 0  | 0 | 24  | 0.00 |
| IDELAYE2/IDELAYE2_FINEDELAY | 0  | 0 | 300 | 0.00 |
| ODELAYE2/ODELAYE2_FINEDELAY | 0  | 0 | 100 | 0.00 |
| IBUFDS_GTE2                 | 0  | 0 | 4   | 0.00 |
| ILOGIC                      | 0  | 0 | 300 | 0.00 |
| OLOGIC                      | 0  | 0 | 300 | 0.00 |
| +-----+-----+-----+-----+   |    |   |     |      |

## 5. Clocking

-----

|                           |      |       |           |       |
|---------------------------|------|-------|-----------|-------|
| +-----+-----+-----+-----+ |      |       |           |       |
| Site Type                 | Used | Fixed | Available | Util% |
| +-----+-----+-----+-----+ |      |       |           |       |
| BUFGCTRL                  | 1    | 0     | 32        | 3.13  |
| BUFIO                     | 0    | 0     | 24        | 0.00  |
| MMCME2_ADV                | 0    | 0     | 6         | 0.00  |
| PLLE2_ADV                 | 0    | 0     | 6         | 0.00  |
| BUFMRCE                   | 0    | 0     | 12        | 0.00  |
| BUFHCE                    | 0    | 0     | 96        | 0.00  |
| BUFR                      | 0    | 0     | 24        | 0.00  |
| +-----+-----+-----+-----+ |      |       |           |       |

## 6. Specific Feature

-----

|                           |      |       |           |       |
|---------------------------|------|-------|-----------|-------|
| +-----+-----+-----+-----+ |      |       |           |       |
| Site Type                 | Used | Fixed | Available | Util% |
| +-----+-----+-----+-----+ |      |       |           |       |
| BSCANE2                   | 0    | 0     | 4         | 0.00  |
| CAPTUREE2                 | 0    | 0     | 1         | 0.00  |
| DNA_PORT                  | 0    | 0     | 1         | 0.00  |

|             |   |   |   |      |  |
|-------------|---|---|---|------|--|
| EFUSE_USR   | 0 | 0 | 1 | 0.00 |  |
| FRAME_ECCE2 | 0 | 0 | 1 | 0.00 |  |
| ICAPE2      | 0 | 0 | 2 | 0.00 |  |
| PCIE_2_1    | 0 | 0 | 1 | 0.00 |  |
| STARTUPE2   | 0 | 0 | 1 | 0.00 |  |
| XADC        | 0 | 0 | 1 | 0.00 |  |
| +-----+     |   |   |   |      |  |

## 7. Primitives

-----

| +-----+  |      |                     |
|----------|------|---------------------|
| Ref Name | Used | Functional Category |
| +-----+  |      |                     |
| FDRE     | 29   | Flop & Latch        |
| LUT2     | 25   | LUT                 |
| RAMD32   | 12   | Distributed Memory  |
| IBUF     | 12   | IO                  |
| OBUF     | 10   | IO                  |
| CARRY4   | 8    | CarryLogic          |
| LUT1     | 5    | LUT                 |
| RAMS32   | 4    | Distributed Memory  |
| LUT6     | 4    | LUT                 |
| LUT4     | 2    | LUT                 |
| BUFG     | 1    | Clock               |
| +-----+  |      |                     |

## 8. Black Boxes

-----

|          |      |
|----------|------|
| +-----+  |      |
| Ref Name | Used |
| +-----+  |      |

## 9. Instantiated Netlists

-----

|          |      |
|----------|------|
| +-----+  |      |
| Ref Name | Used |
| +-----+  |      |

| Utilization |             | Post-Synthesis | Post-Implementation |
|-------------|-------------|----------------|---------------------|
|             |             | Graph   Table  |                     |
| Resource    | Utilization | Available      | Utilization %       |
| LUT         | 40          | 41000          | 0.10                |
| LUTRAM      | 8           | 13400          | 0.06                |
| FF          | 29          | 82000          | 0.04                |
| IO          | 22          | 300            | 7.33                |
| BUFG        | 1           | 32             | 3.13                |

Fig 13: Utilization of FIFO in tabular form

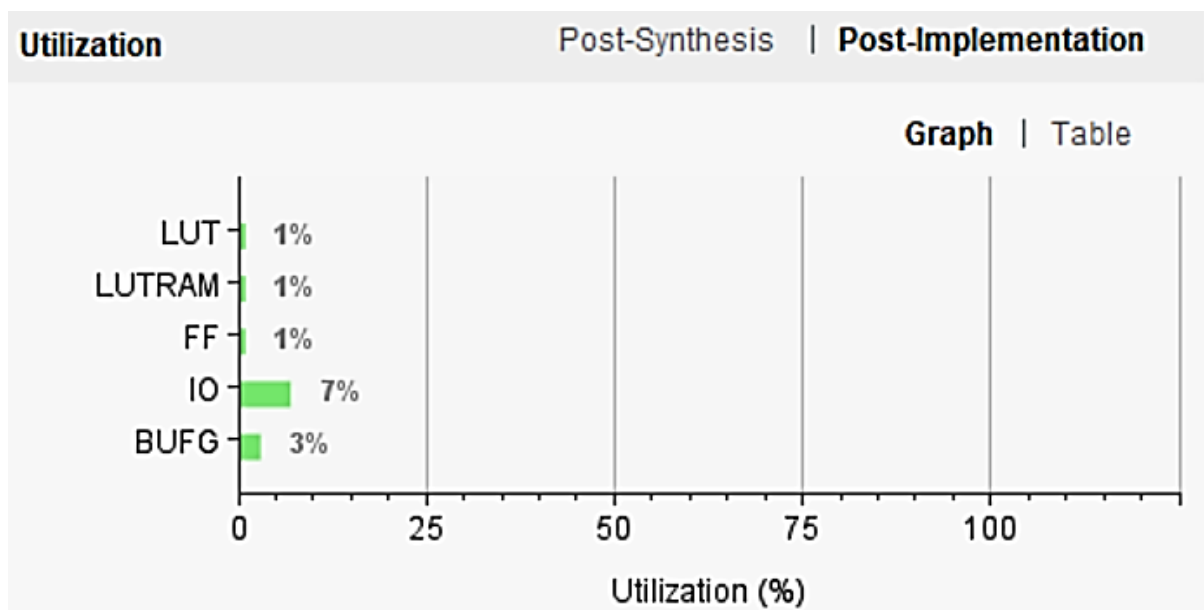


Fig 14: In graphical presentation

### 3.5. SYNTHESIS REPORT:

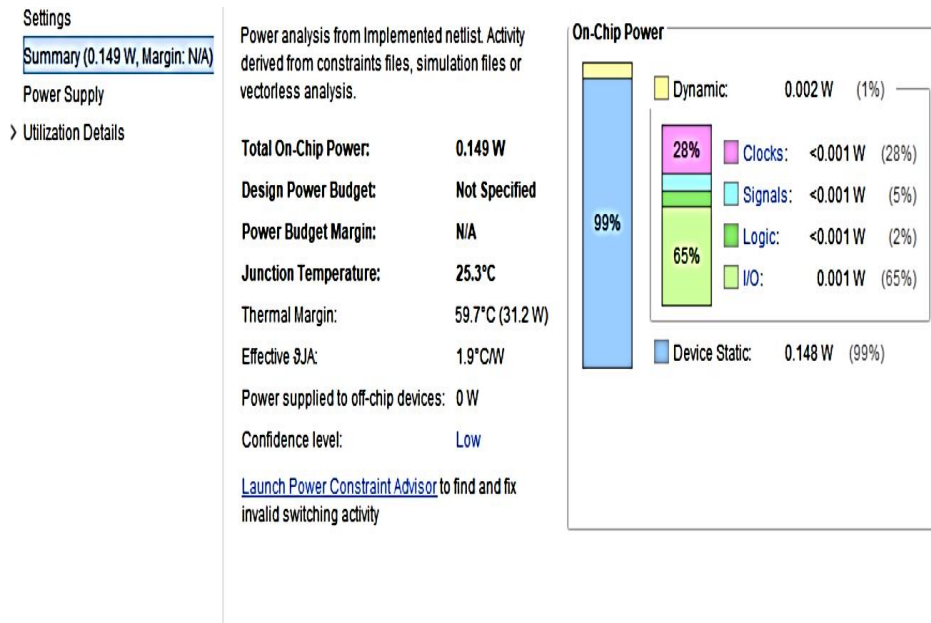


Fig15 : Summary report of the project

### CONCLUSION:

We have learnt the following accepts during the PS II duration:

- Recap of how to write RTL codes using Verilog HDL
- How to write behavioral codes using Verilog HDL to develop testbenches
- Design of basic logic elements like half adders, full adders, counters, Finite State Machines, sequence detectors etc.
- Design and application of Synchronous FIFO
- How to develop testbench and run simulations to verify the correctness of your design (using Modelsim)
- How to synthesize an RTL design to a gate-level design and power consumption report (using Vivado)

I believe, my set objectives are more than fulfilled.

## APPENDIX REFERENCES:

<https://scholarworks.rit.edu/cgi/viewcontent.cgi?article=11135&context=theses>  
<https://www.quora.com/What-is-the-Verilog-code-for-the-synchronous-FIFO-block-diagram>  
<https://www.fpga4student.com/2017/01/verilog-code-for-fifo-memory.html>  
<https://www.edaboard.com/showthread.php?233779-synchronous-FIFO-design>