FIFO Implementation Using FPGA

Submitted in partial fulfillment of the requirements

of the degree of

Bachelor of Technology

by

Mahesh Baraskar **(Reg. No. 2016BEC151)**

Mukul Lokhande **(Reg. No. 2016BEC077)**

Pushpak Ghatode **(Reg. No. 2016BEC153)**

Tejaswini Khairnar **(Reg. No. 2017BEC513)**

Ratnamala Patil **(Reg. No. 2016BEC041)**

Supervisor (s):

Dr. Y. V. Joshi

Dr. S. S. Gajre

Mr. Vaibbhav Taraate

Department of Electronics and Telecommunication Engineering,

Shri Guru Gobind Singhji Institute of Engineering & Technology,

Vishnupuri, Nanded, Maharashtra, India, 431606.

2019-20

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Mahesh Baraskar (Reg. No. 2016BEC151)
Mukul Lokhande (Reg. No. 2016BEC077)
Pushpak Ghatode (Reg. No. 2016BEC153)
Tejaswini Khairnar (Reg. No. 2017BEC513)
Ratnamala Patil (Reg. No. 2016BEC041)

Date: 2nd Dec 2019.

CERTIFICATE

This is to certify that the report entitled **"FIFO Implementation Using FPGA"** *being submitted by* Mahesh Baraskar, Mukul Lokhande, Pushpak Ghatode, Tejaswini Khairnar & Ratnamala Patil *to* **Shri Guru Gobind Singhji Institute of Engineering and Technology, Vishnupuri, Nanded (M.S.), India,** *as partial fulfillment for the award of the degree of* **Bachelor of Technology** *in* **Electronics and Telecommunication Engineering***, is a record of bonafide work carried out by them under our supervision and guidance. The matter contained in this report has not been submitted to any other university for the award of any degree or diploma.*

Dr. Y. V. Joshi	**Dr. M. B. Kokare**
Dr. S. S. Gajre	**H.O.D.**
(Elect. & Telecom. Engg. Dept.)	**(Elect. & Telecom. Engg. Dept.)**

Mr. Vaibbhav Taraate
Industrial guide
(One Rupee Semi. Tech.)

APPROVAL SHEET

This report entitled "**FIFO Implementation Using FPGA**" *by* Mahesh Baraskar, Mukul Lokhande, Pushpak Ghatode, Tejaswini Khairnar & Ratnamala Patil is approved for the degree of Bachelor of Technology.

Examiner (s)

____

____

____

Supervisor (s)

____

____

____

Date : _____________

Place : Nanded

ACKNOWLEDGEMENT

The Project report of **“FIFO Implementation Using FPGA”** is an outcome of guidance, moral support and devotion bestow on us throughout our work.

For this we acknowledge and express our sincere gratitude and thanks to everybody who have been source of inspiration during the project prepartion.

First and foremost we offer out sincere phrases of thanks to **Dr. Y. V. Joshi, Dr. S. S. Gajre, Mr. Vaibbhav Taraate** for their guidance, valuable suggestion and providing help whenever necessary, during the project preparation.

Finally we would like to express thanks from bottom our heart to everyone who directly or indirectly help us during this project.

We are going to use property of impedance which changes accodingly material deformation or combination for different application.

Mahesh Baraskar (Reg. No. 2016BEC151)

Mukul Lokhande (Reg. No. 2016BEC077)

Pushpak Ghatode (Reg. No. 2016BEC153)

Tejaswini Khairnar (Reg. No. 2017BEC513)

Ratnamala Patil (Reg. No. 2016BEC041)

Abstract

First-In-First-Out (FIFO) memory buffers are widely used to link to different clock domain systems. The design of the FIFO buffer architecture and demonstrating it on the Spartan 3E FPGA Board using Xilinx ISE 14.7 is done here. The type of FIFO designed is the circular type FIFO. So it has its own advantages over other architectures. It also uses level synchronizers and other important logic.

Keywords:

FIFO implementation, FPGA implementation, Multi Clock Domain Design, Clock Domain Crossing, Write Control, Read Control, Synchronizer, SRAM, Flag Logic.

CONTENTS

List of Figures i

Sr. No.	Title	Page No.
1.	**Introduction**	1
2.	**Literature Survey**	**2**
	2.1 Implementation of FIFO buffer	2
	2.2 Types of FIFOs	2
	2.3 Types of FIFO	3
	2.4 How a FIFO performs read and write?	3
3.	**Specifications & Top-Level Block Diagram**	**5**
	3.1 Specifications	5
	3.2 Top Level Diagram	6
4.	**Prerequisites**	**7**
	4.1 FPGA (Field programmable gate arrays)	**7**
	4.2 Xilinx ISE Design Suite - Web PACK Edition	**7**
	4.3 SPARTAN 3E FPGA Board	**7**
5.	**Write Control Block**	**8**
	5.1 Pin Functionality	8
	5.2 RTL – Write Control	9
	5.3 Test cases for read control block	10
	5.4 Simulation waveforms	10
6.	**Read Control Block**	**11**
	6.1 Pin Functionality	11
	6.2 RTL - Read Control	12
	6.3 Test cases for read control block	13
	6.4 Simulation waveforms	13
	6.5 Binary UP Counter	14

7.	**Memory**	**15**
	7.1 Dual-Port SRAM:	15
	7.2 Benefits of the DP SRAM	15
	7.3 Pin Functionality	16
	7.4 Testcases	18
	7.5 Simulation Results	20
8.	**Flag logic in FIF0 memory**	**21**
	8.1 Top Level Flag Logic	21
	8.2 Pin description	22
	8.3 RTL schematic of flag logic	23
	8.4 Address pointer gap generation	23
	8.5 Full and Empty Flag Generation logic	23
	8.6 Test bench for flag logic	24
	8.7 Simulation of flag logic	24
9.	**Synchronizer**	**25**
	9.1 Top Level RTL _ Synchronizer Block	25
	9.2 RTL _Synchronizer Block	26
	9.3 Synchronization of write pointer	26
	9.4 Synchronization of read pointer	26
	9.5 Top Level RTL _ Binary to Gray Converter	27
	9.6 Top Level RTL _Gray to Binary Converter	27
	9.7 RTL _ Binary to Gray Converter	28
	9.8 RTL _Gray to Binary Converter	28
	9.9 Simulation waveforms	29
10.	**Synthesis Report**	**30**
	10.1 HDL synthesis report	30
	10.2 Final report	31
	10.3 Device utilization summary	32
	10.4 Partition resource summary	32

11.	**Conclusion** 11.1 Advantages 11.2 Disadvantages 11.3 General Applications 11.4 Conclusion	**33** 33 33 34
12.	**Contribution and Future Scope** 12.1 Contribution 12.2 Future Scope	**35** 35 35

References **36**

List of Figures

Sr. No.	Title of figure	Page No.
3.1	Top Level Diagram	6
5.1	Write Control Block	8
5.2	RTL - Write Control	9
5.3 , 5.4	Simulation waveforms of Write Control	10
6.1	Read Control Block	11
6.2	RTL - Read Control Block	12
6.3 , 6.4	Simulation waveforms of Read Control	13
6.5	Binary UP Counter	14
7.1	DP_SRAM Top level diagram	16
7.2	The RTL of DP_SRAM block	18
7.3	Simulation Results of the DP_SRAM	20
8.1	Top Level Flag Logic	21
8.2	RTL Schematic of flag logic	23
8.3	Simulation of flag logic	24
9.1	Top Level RTL of Synchronizer Block	25
9.2	RTL of Synchronizer Block	26
9.3	Top Level RTL of Binary to Gray Converter	27
9.4	RTL of Binary to Gray Converter	27

9.5	Top Level RTL of Gray to Binary Converter	28
9.6	RTL of Gray to Binary Converter	28
9.7	Simulation of Binary to Gray Converter	29
9.8	Simulation of Gray to Binary Converter	29

CHAPTER 1

Introduction

Exchange of data between different PCBs. It requires the need of intermediate buffering elements as the data on receiving PCB arrive at higher speed than the ability of PCB to process it. That is different Input and Output clocks in simple words. Then such intermediates are called FIFO memories.

Two electronic systems are invariably connected to the input and output of a FIFO - one that writes and one that reads. The first to be implemented is an Exclusive FIFOs as it is easy to do so. It means only reading or writing is possible at a time. But now-a-days, Concurrent FIFOs are popular and mostly used because so numerous applications need synchronous read / write versions and it is easy to modify it to use as Exclusive FIFO.

More examples of such kind of system are -

1. Customer queue at the shop working on the basis of First Come First Serve.

2. Electronics system with different components of different propagation delays.

3. A Compact Disk (CD) player compensating the data rate of rotation of the disk and analog to digital converter by using Buffering Element.

CHAPTER 2

Literature Survey

2.1 Implementation of FIFO buffer:

FIFOs can be implemented using software as well as hardware. The choice between a software and a hardware solution depends on the application and the features desired. The only advantage software has is that when requirements change, a software FIFO easily can be adapted to new requirements by modifying its program, while a hardware FIFO may demand a new circuit board. Software is more flexible than hardware. The advantage of the hardware FIFOs can be seen in their operating speed and other performance parameter.

2.2 Types of FIFOs:

There are 3 different types of the FIFOs. They are as follows

2.2.1 Shift Register -

It has invariable or fixed number of stored data word. So that they are more robust to the changes. They are not used usually, because they are based on the Fall Back architecture and speed is very low as compared to new types.

2.2.2 Exclusively Read/Write FIFO -

It has variable number of stored data word. There are various synchronous Timing restrictions. The writing of data is dependent of how the data are read. To use it in between two circuits with different frequencies there is need of a synchronization circuit, but it usually results in decrease in data rate.

2.2.3 Concurrent Read/Write FIFO -

It has variable number of stored data word. but it has no timing restriction. There is no interdependence between the writing and reading of data. Simultaneous Reading and Writing is possible in overlapping or in successive fashion. Two circuits operating with different frequencies can be connected together without a synchronizing circuit. So that the synchronization task will be done in FIFO itself.

2.3 Types of FIFO

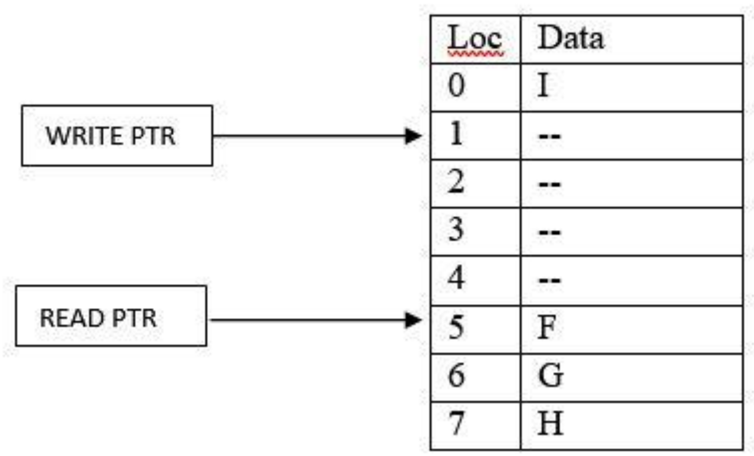
Depending on the Control signal there are two types of the Concurrent FIFOs. They are as follows -

1. Synchronous FIFOs
2. Asynchronous FIFOs

In Concurrent FIFOs, the synchronization is achieved by using the internal synchronization circuits. A simple synchronization circuit can be made by using a series of FF or a single FF. The clock used in these networks is mainly the local clock, so that the input data is synchronized with that.

2.4 How a FIFO performs read and write?

A FIFO is first in first out memory. Data being entered at one end and removed at the other, with the amount of data in the FIFO stored up to certain maximum limit. But shifting data either way in memory is costly in hardware. Hence, It is convenient to use a memory normally and function like a circular buffer by pointing to the next address to write to or to read from. These addresses are stored in different registers named the read or write pointer. The simple FIFO using an 8 word memory is shown below:



The input word is written at the empty address 1 (present write pointer), and the write pointer increased by 1. The next output read will fetch the word writtem at address 5 (present read pointer), and the read pointer is increased by 1. Here, the addresses are circular for 3 bit wide pointers. Adding 1 to 7 gives 0 in 3 bit unsigned addition. Whole FIFO systems requires to identify the full and empty conditions. There are different methods to do so. Another register can be used to count words in the FIFO. Another way of firmware implementation is to compare the read pointer and the write pointer. If both are same, then FIFO is empty and write pointer one place behind read pointer (following) is full for FIFO.

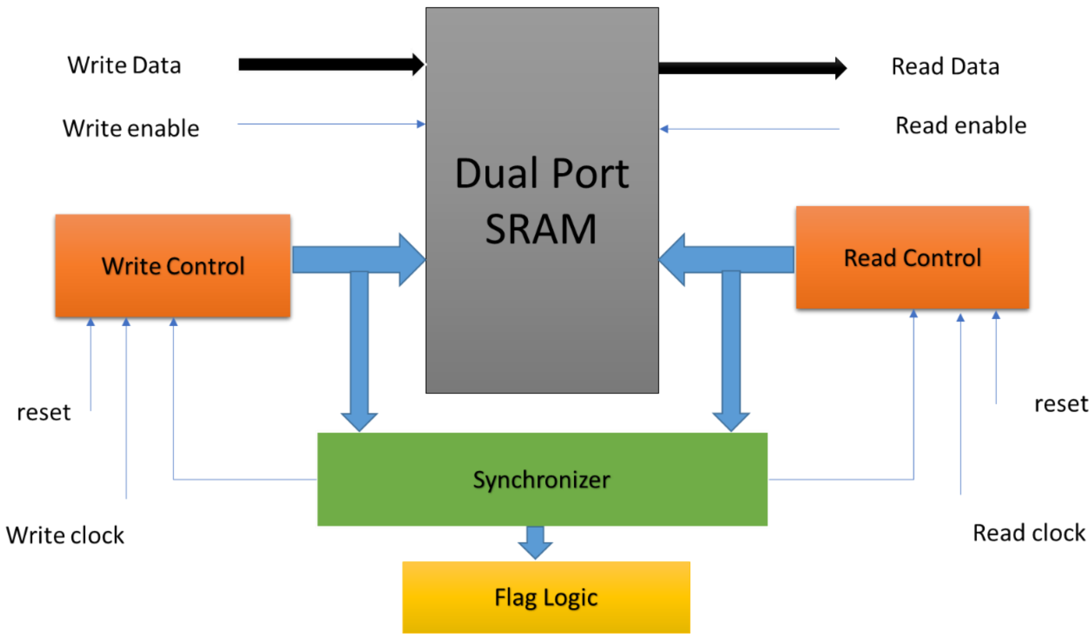
CHAPTER 3

Specifications & Top-Level Block Diagram

3.1 Specifications:

- BURST Size: 1.5K bytes
- FIFO Size: 1 K bytes
- Input Data Pins: 8
- Input Pins: 2 (R/W enable)
- Output Data Pins: 8
- Write Pointer Pins: 10
- Read Pointer Pins: 10
- Flag Pins: 5 (EMPTY, FULL, HALF FULL / EMPTY, ALMOST, FULL, ALMOST EMPTY)
- Reset: 1
- Write Clock: 100 MHz
- Read Clock: 33 MHz

3.2 Top Level Diagram:



Top Level Diagram fig.3.1

CHAPTER 4

Prerequisites

4.1 Field Programmable Gate Array:

FPGAs are semiconductor devices consisting of a matrix with Configurable Logic Blocks connected with programmable interconnects. FPGAs are reprogrammed for desired functionality based on requirements after manufacturing. It distinguishes FPGAs from ASICs which is customized for special tasks. Now multiple advanced FPGAs are available with a greater number of functionalities. These chips have on board DSP processor, memory (Block RAM), Digital Clock Manager (DCM), Input Output Block (IOBs), Configurable Logic Blocks (CLBs). Fig1.3 shows functional blocks of FPGA.

4.2 Xilinx ISE Design Suite - WebPACK Edition:

Xilinx ISE Design Suite is a software designed by Xilinx Inc. for synthesis and analysis of hardware designed using HDLs. It enables users to synthesize designs, perform timing analysis, view RTL diagrams, and configure target device.

ISE WebPACK provides design flow providing access to the ISE functionality free of cost.

4.3 SPARTAN 3E FPGA Board:

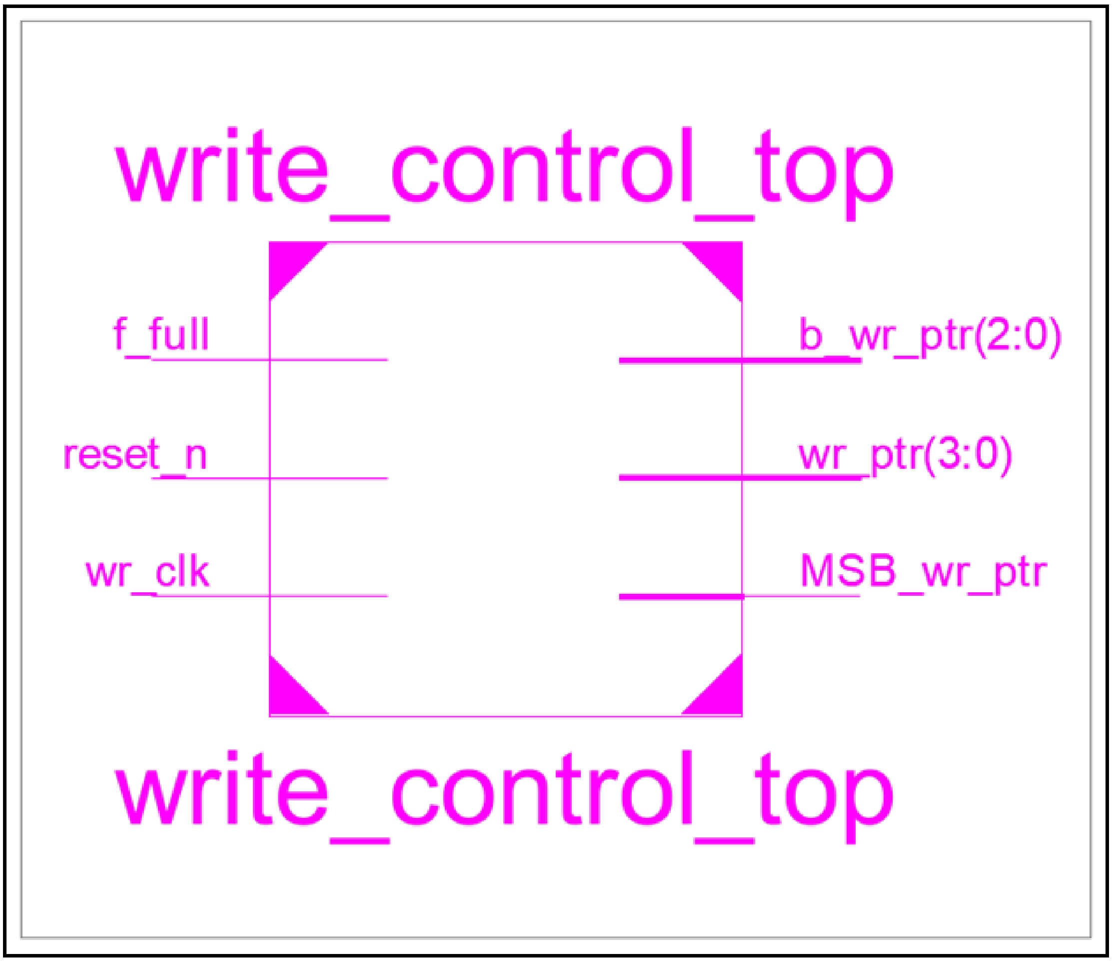
The board provides a platform for design to implemented on Xilinx Spartan 3E FPGA. This board features 500,000 gates for a complex and high-volume designs and 64 MB DDR RAM. It supports USB and JTAG parallel programing interface. It has LCD and LED interfaces for testing purpose.

CHAPTER 5

Write Control Block

Write control block controls the data writing operation to FIFO memory. It contains the signals to enable or disable the write operation, generates address to where data is to be stored in FIFO memory. Write pointer always points to the next memory address where data is to be written.

The fig. shows top level of write control which includes following inputs/ outputs:



Write Control Block fig.5.1

5.1 Pin Functionality:

f_full: When data word is written to the FIFO, it is necessary to check whether there is space available in the FIFO memory. It is done by inquiring the f_full. When FIFO memory is full, this signal activates and stop the writing operation. f_full act as disable signal to write operation.

reset_n: Active low reset is provided to FIFO. On reset write pointer points to the zeroth memory location.

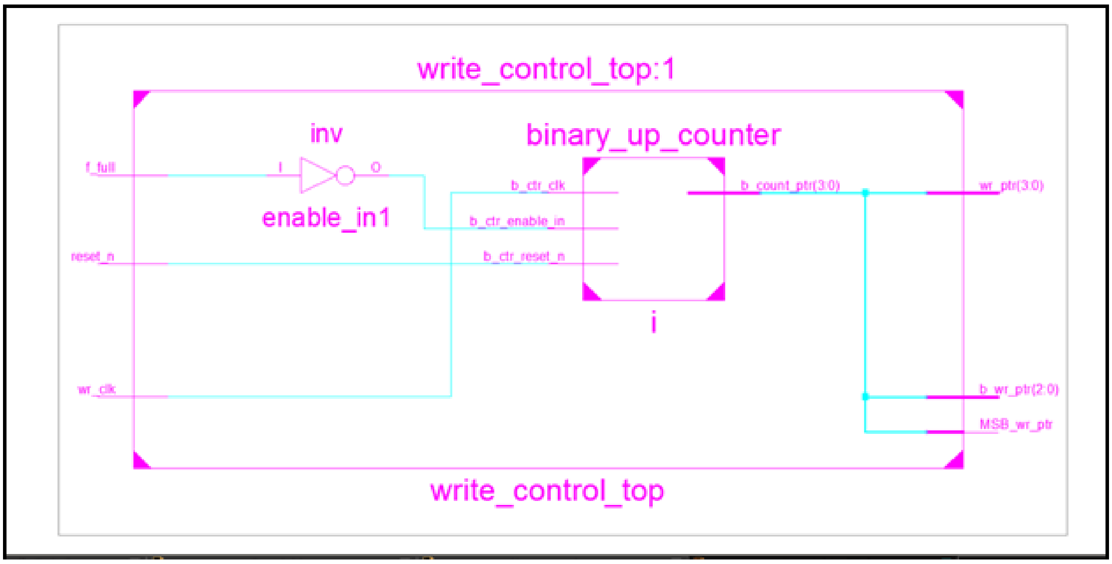
wr_clk: Every write operation is completed on the positive edge of the write clock. Write clock is dependent on the clock frequency of data writing system to FIFO.

b_wr_ptr: It is the (n-1) bits of write pointer which points location address of FIFO memory where data has to be stored. Write pointer increments on a positive edge of the write clock.

MSB_wr_ptr: Its n^{th} bit of write pointer used to generate Full or Empty status of FIFO.

5.2 RTL – Write Control

Following fig. shows the RTL of write control block



RTL- Write Control fig.5.2

Here, inverted input of f_full is given to enable pin of binary counter. When f_full is High, it's inverted output is Low which disables the counter and counter stop counting. Clock to the counter is given from writing system. Active low reset is given to reset the counter. The counter used here is n-bit counter out of which (n-1) bits (b_wr_ptr) are used as pointer to point the memory locations and nth bit (MSB_wr_ptr) is used to generate flag logic.

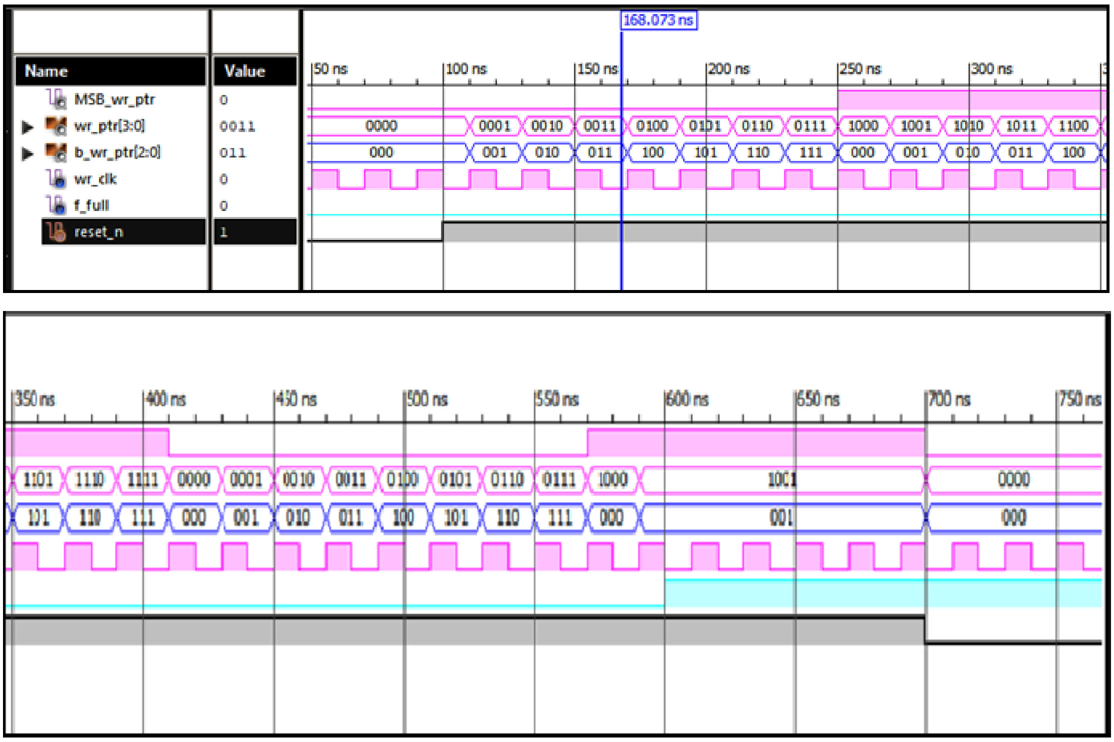
5.3 Test cases for read control block:

1. reset_n = 0, f_full = x, wr_clk = x
   output: b_wr_ptr = 000

2. reset_n = 1, f_full = 1, wr_clk = x
   output: b_wr_ptr = hold previous value

3. reset_n = 1, f_empty = 0, rd_clk = positive edge
   output: b_wr_ptr = increment by 1

5.4 Simulation waveforms:



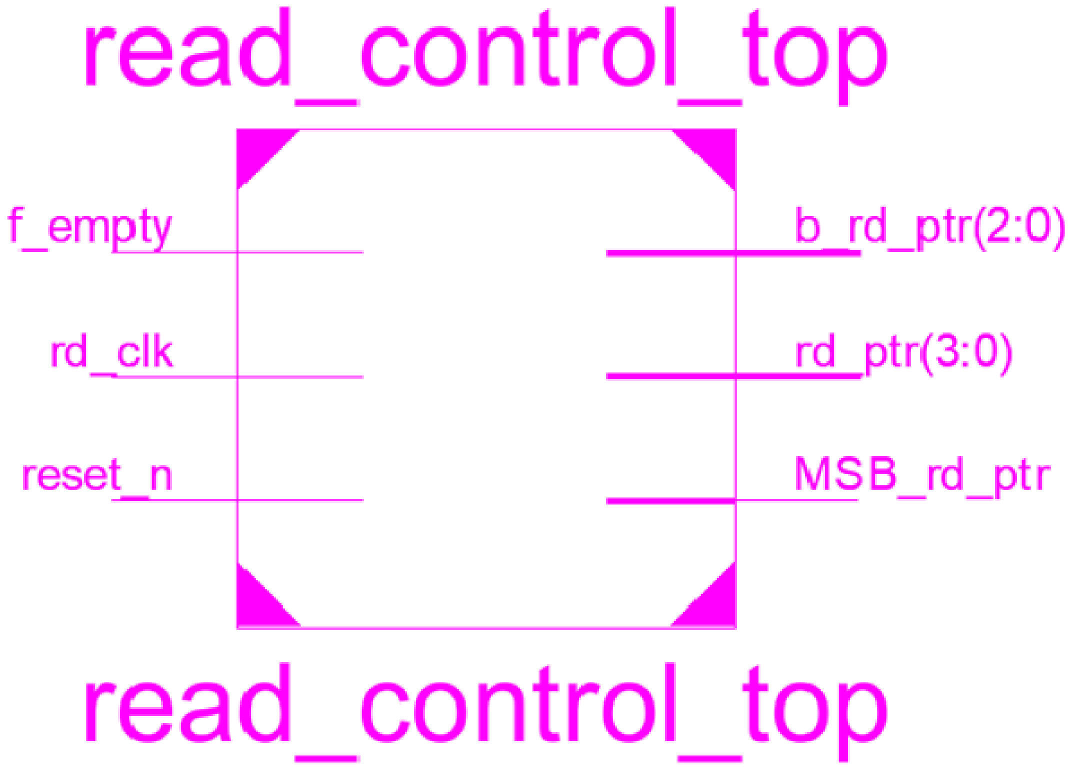
Simulation waveforms of Write Control fig 5.3,5.4

CHAPTER 6

Read Control Block

Read control block controls the data reading operation from FIFO memory. It contains the signals to enable or disable the read operation, generates address from where data is being read. Read pointer always points to the current memory addresss from where data is to be read.

The fig. shows top level of read control which includes following inputs/ outputs:



Read Control Block fig.6.1

6.1 Pin Functionality:

f_empty: When data word is read from the FIFO, it is necessary to check if data is available in the FIFO memory. It is done by inquiring the f_empty. When FIFO memory is empty, this signal is activated and stop the reading operation. f_empty act as disable signal to read operation.

rd_clk: Every read operation is performed on the positive edge of the read clock. Read clock is dependent on the clock frequency of data reading system from FIFO.

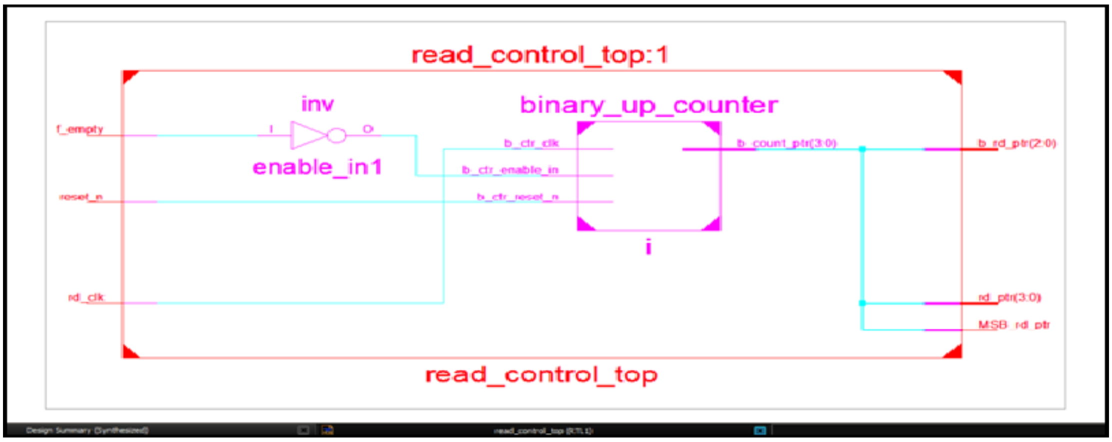
reset_n: Active low reset is provided to FIFO. On reset read pointer points to the zero memory location.

b_rd_ptr: It is the (n-1) bits of read pointer which points the current memory location address of FIFO memory from where data has to be read. Read pointer increments on a positive edge of the read clock.

MSB_rd_ptr: Its n^{th} bit of read pointer used to generate Full or Empty status of FIFO.

6.2 RTL - Read Control

Following fig. shows the RTL of read control block



RTL-Read Control Block fig.6.2

Here, inverted input of f_empty is given to **enable** pin of binary counter. When f_empty is High, it's inverted output is Low which disables the counter and counter stop counting. Clock to the counter is given from reading system. Active low reset is given to reset the counter. The counter used here is n-bit counter out of which (n-1) bits (b_rd_ptr) are used as pointer to point the memory locations and n^{th} bit (MSB_rd_ptr) is used to generate flag logic.

6.3 Test cases for read control block:

1. reset_n = 0, f_empty = x, rd_clk = x

   output: b_rd_ptr = 000

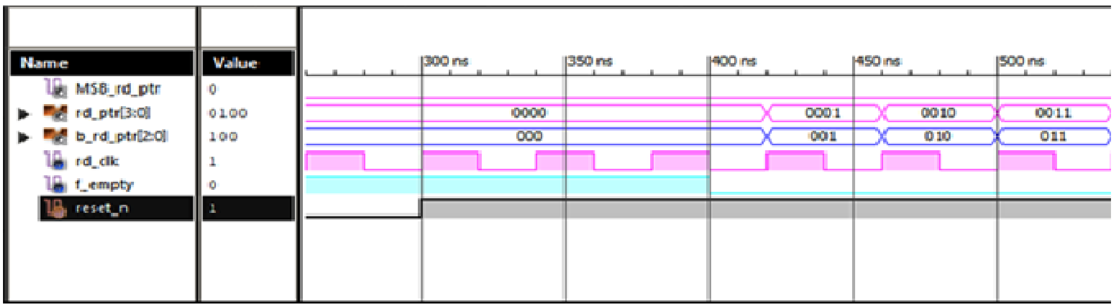
2. reset_n = 1, f_empty = 1, rd_clk = x

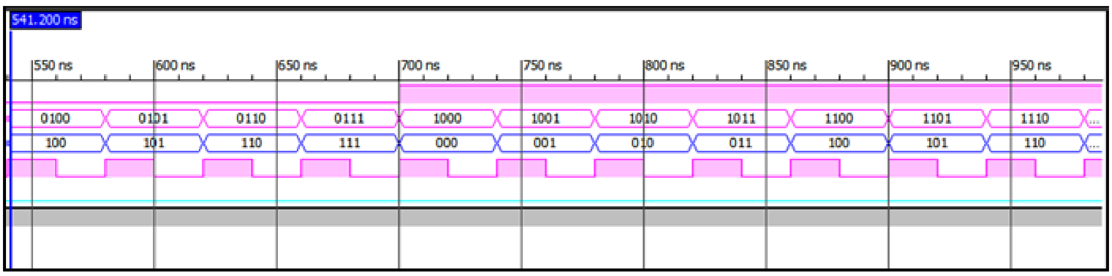
   output: b_rd_ptr = hold previous value

3. reset_n = 1, f_empty = 0, rd_clk = positive edge

   output: b_rd_ptr = increment by 1

6.4 Simulation waveforms:

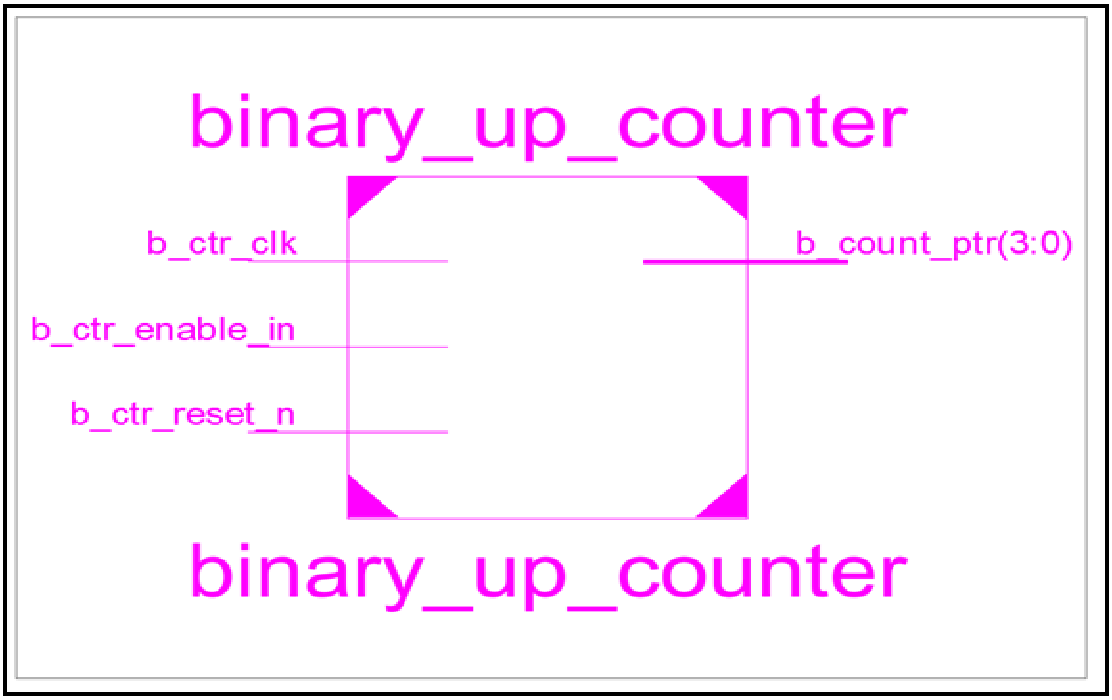




Simulation waveforms of Read Control fig.6.3,6.4

6.5 Binary UP Counter:

Read control block and Write control block contains an n bit binary counter which operates on read clock and write clock respectively. On every positive edge of clock, counter is incremented by one. N-bit up counter is designed as a separate module and is instantiated in FIFO module for read binary counter and write binary counter.



Binary UP Counter fig 6.5

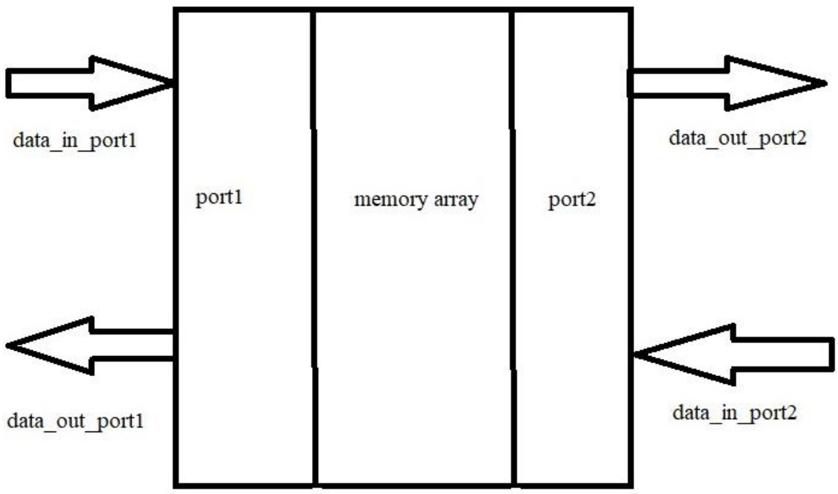
CHAPTER 7

Memory

The memory is the central part of the FIFO buffer. Memories are available in various types such as SRAM, DRAM, ROM, etc. But for the concept of the circular type of FIFO we need to use a memory which is having two different ports for the exchange of data with the external world. Such type of memory is used here known as Dual-Port SRAM.

7.1 Dual-Port SRAM:

The concept of Dual-Port SRAM can be well understood by looking at the following diagram.



This RAM allows multiple read or write operation at the same time, unlike Single Port RAM allowing only one operation at a time. Even though we need the DP SRAM for employing a circular memory there are other benefits of the DP SRAM.

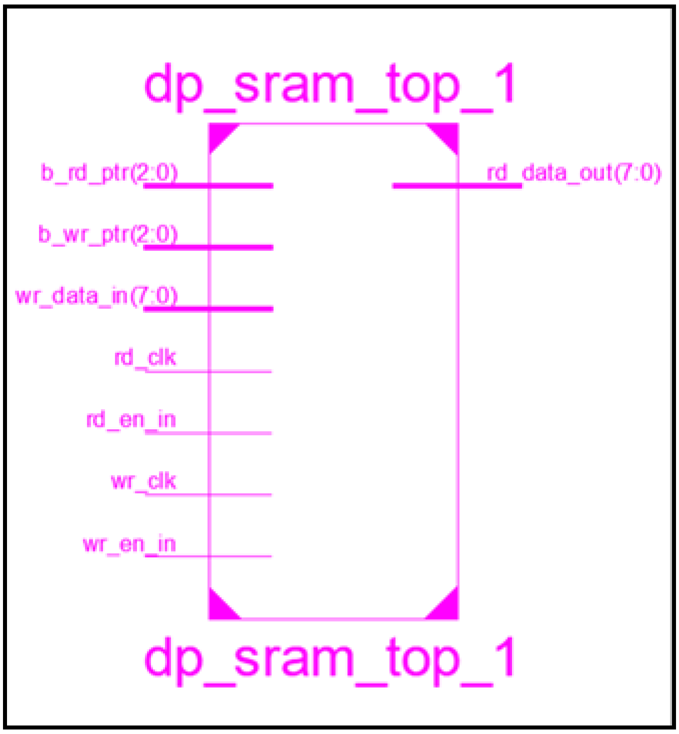
7.2 Benefits of the DP SRAM:

1. Both ports have an addressable separate memory and provide random access to data irrespective of other.
2. Both ports provides buffering and high bandwidth.
3. There is no interface mismatch.
4. Both ports provide support and flexible interfaces for common standards.

5. It reduces the design complexity.

7.3 Pin Functionality:

Now, let us examine the basic structure of the Dual Port SRAM. Every basic element has some input and output pins / ports, DP SRAM have other pins / ports other than output and input data ports as shown in fig. For a DP_SRAM to be used in the FIFO buffer the two ports are assigned to two different particular functions, one is reading port and other one is writing port. The pin description of DP SRAM can be shown by using top level diagram as:



DP_SRAM Top level diagram fig.7.1

b_rd_ptr: It is the address/location, where the current read operation will be performed. It is of [n-1:0] width (where 2^n = number of words can be stored in the memory).

b_wr_ptr: It is the address/location, where the next write operation will be performed. It is of [n-1:0] width (where 2^n = number of words can be stored in the memory).

wr_data_in: It is a port of input data. The data present on this port is sampled on the positive edge of wr_clk and stored in the memory at the location specified by the b_wr_ptr. It is of [d_length-1:0] width (where d_length = the length of the single word).

rd_clk: It is the input to supply the rd_clk to the block. On the positive edge of this signal the data from the location specified by b_rd_ptr in the memory will be given to the output port (rd_data_out).

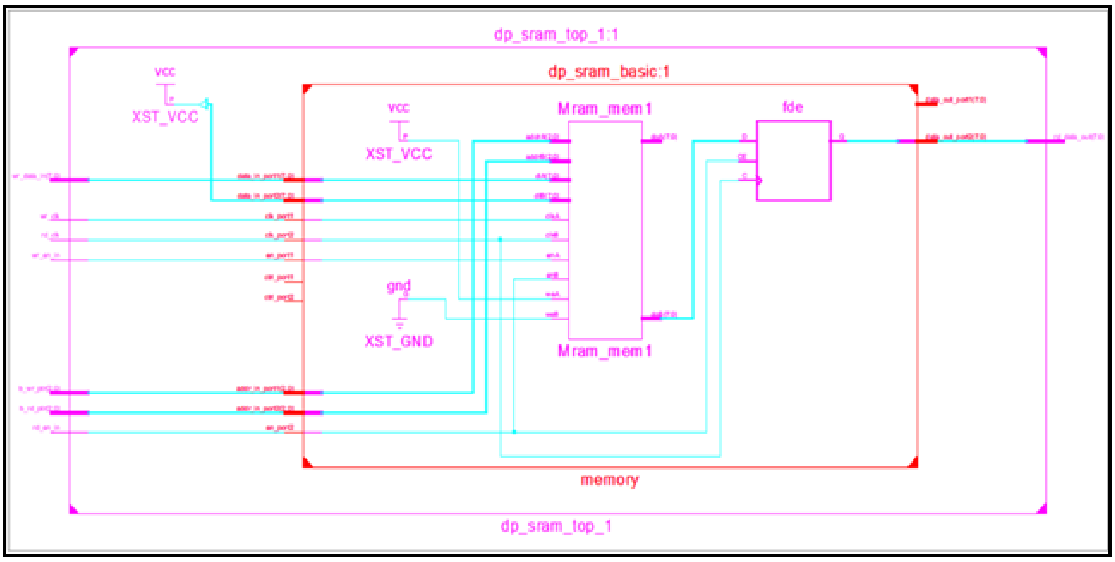
wr_clk: It is the input to supply the wr_clk to the block. On the positive edge of this signal the data present on the input port(wr_data_in) will be sampled and stored in the memory at the location specified by the b_wr_ptr.

rd_en_in: It is an enable pin for the read port. If this pin is 1 or HIGH then only the read operation will be performed on the positive edge of rd_clk signal, if not the data will not be read from the memory even if positive edge of rd_clk signal.

wr_en_in: It is an enable pin for the write port. If this pin is 1 or HIGH then only the write operation will be performed on the positive edge of wr_clk signal, if 0 or LOW the data will not be written to the memory even if positive edge of wr_clk signal.

rd_data_out: It is a port of output data. The data from memory present at the location specified by the b_rd_ptr port will be read out on this port on the positive edge of rd_clk signal. It is of [d_length-1:0] width (where d_length = the length of the single word).

Here we will discuss the internal structure of the block. The fig.1.3 shows the RTL for the DP_SRAM block. It consists of internal blocks such as output register, main memory and internal connections. Here we can see that the port 1 is used for the only writing operation and port 2 is used only for reading operation. The data input port of port 2 is connected to the high impedance (logic state: z) and output data port of port 1 is left unconnected.



The RTL of DP_SRAM block fig.7.2

This RTL is verified by using the following test-cases with the help of the testbench and simulation features of Xilinx ISE 14.7.

7.4 Testcases:

Keeping clock signal as, the time period of wr_clk signal is 10 nsec and that of the rd_clk signal is 20 nsec.

(Start condition initial values of clock signals and other ports.)

1) For 9 nsec: wr_data_in =8’h0
   b_wr_ptr = 3'b001
   wr_clk = 0
   wr_en_in = 0
   b_rd_ptr = 3'b000
   rd_clk = 0
   rd_en_in = 0

(Writing to memory at some locations.)

For all above testcases the output on rd_data_out will be don’t care only.

rd_data_out = 8’hxx

2) For 11 nsec: wr_data_in = 8’h84
   b_wr_ptr = 3'b101
   wr_en_in = 1

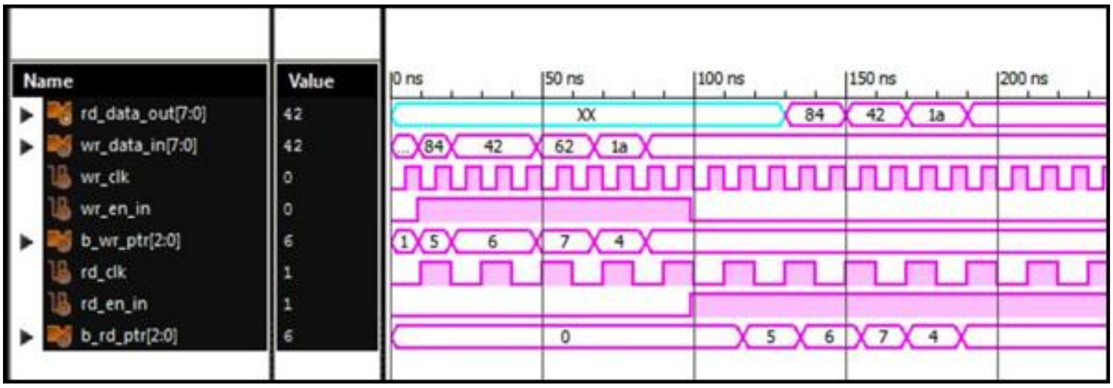
3) For 28 nsec: wr_data_in = 8'h42
b_wr_ptr = 3'b110
4) For 19 nsec: wr_data_in = 8'h62
b_wr_ptr = 3'b111
5) For 17 nsec: wr_data_in = 8'h1a
b_wr_ptr = 3'b100
6) For 15 nsec: wr_data_in = 8'h42
b_wr_ptr = 3'b110

(Reading the data that is stored on locations initially.)

7) For 17 nsec: rd_en_in = 1
wr_en_in = 0
b_rd_ptr = 3'b000
output: rd_data_out = 8'hxx
8) For 19 nsec: b_rd_ptr = 3'b101
output: rd_data_out = 8'h84
9) For 20 nsec: b_rd_ptr = 3'b110
output: rd_data_out = 8'h42
10) For 20 nsec: b_rd_ptr = 3'b111
output: rd_data_out = 8'h62
11) For 18 nsec: b_rd_ptr = 3'b100
output: rd_data_out = 8'h1a
12) For 18 nsec: b_rd_ptr = 3'b110
output: rd_data_out = 8'h84

7.5 Simulation Results:

The Simulation results that came out for the given RTL and the given Testcases is as shown in fig.6.3



Simulation Results of the DP_SRAM fig.7.3

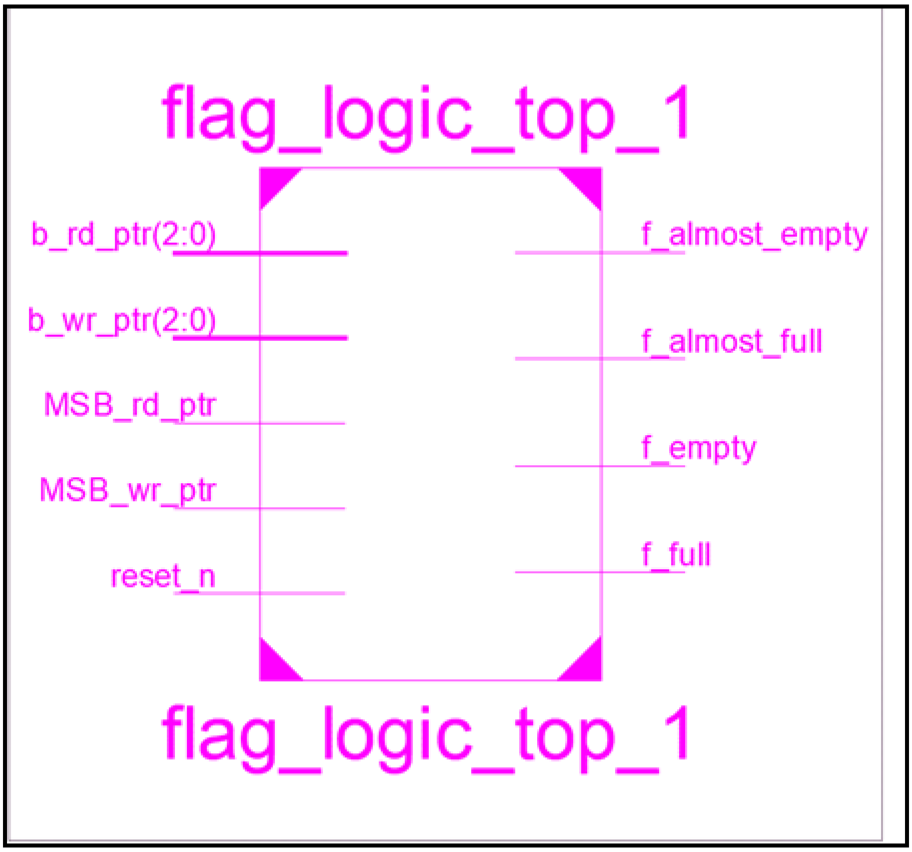
CHAPTER 8

Flag logic in FIFO memory

The empty flag, almost empty flag, full flags are used to determine the FIFO status. These different flags are generated by comparing the read pointer and the write pointer. The flag logic also stops read operation from an empty FIFO and write operation to a full FIFO.

While reading empty FIFO, the output will show the last valid data read or write operation to full FIFO are proceeded to respective next clock edge. The empty flag is a synchronized with the read clock. The full flag is synchronized with write clock. Synchronization of the flag with the respective clock eliminates the use of external synchronization continuously. The write logic for a FIFO must check if the FIFO is not full before write operation. Similarly, the read logic examines for empty flag before read operation.

8.1 Top Level Flag Logic:



Top Level Flag Logic fig.8.1

8.2 Pin description:

There are status flags which shows the flag logic are as follows:

f_full: The status signal to notify full status of the memory. Before giving the wr_enable signal, it is checked on write request signal.

f_empty: The status signal to notify the empty status of the memory. The rd_enable signal is given once read request signal is checked.

reset_n: On the reset both pointer (read and write) are set to zero.

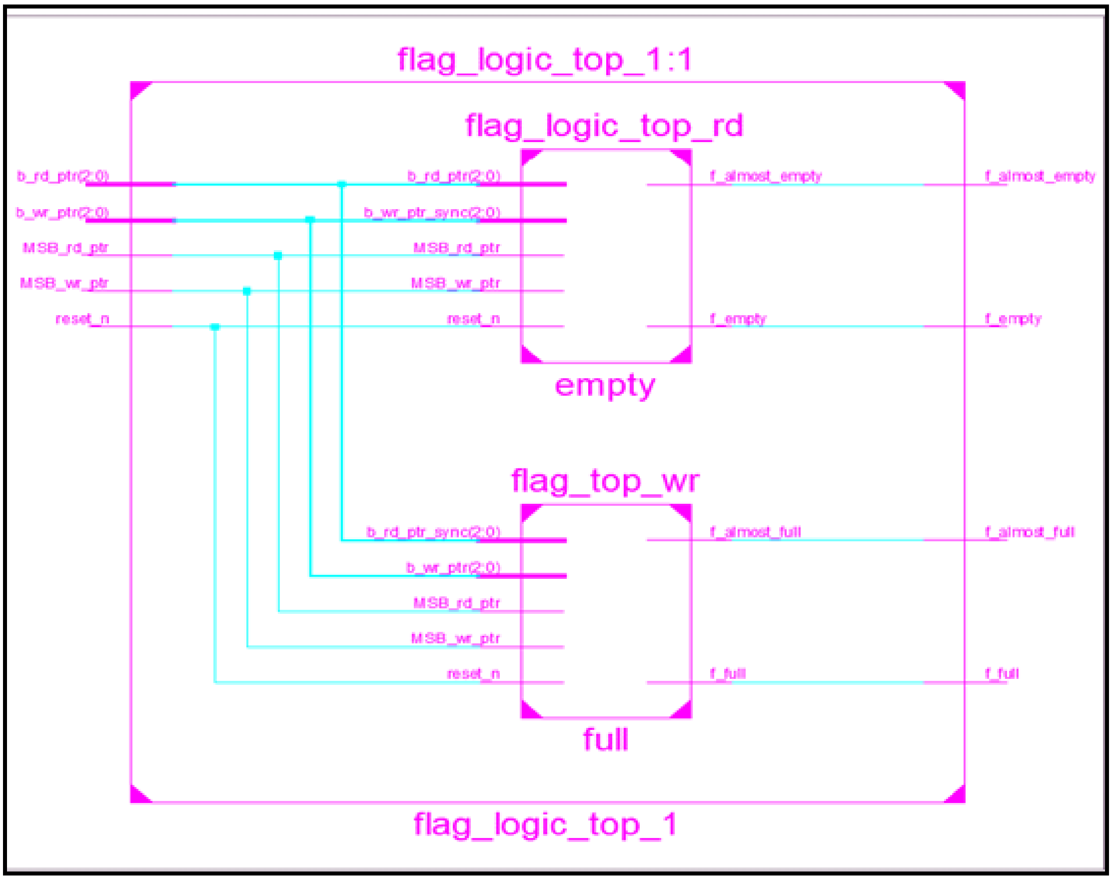
f_almost_full: This pin shows the status of the FIFO that it is almost full as the pointer difference between the write pointer and read pointer is reach to the set value of almost full flag get.

f_almost_empty: This pin shows the status of the FIFO that it is almost empty as the pointer difference between the write pointer and read pointer is reach to the set value of almost empty.

b_rd_ptr and b_wr_ptr: The output from the read and write control block are given as the input to flag logic.

MSB_wr_ptr and MSB_rd_ptr: Two input from the read and write control block are given to flag logic as input and according to value of MSB_wr_ptr and MSB_rd_ptr the FULL and EMPTY flags are set.

8.3 RTL Schematic of flag logic:



RTL Schematic of flag logic fig 8.2

8.4 Address pointer gap generation:

RTL schematic show the block comprises of different comparators and adder, subtractor. On change in rd_ptr and wr_ptr, these modules calculate the difference between rd_ptr and wr_ptr. Thus, the status of the FIFO is dynamically reflected by ptr_diff . This adaptivity of the design permits to use different read and write clock frequencies. It takes read and write addresses as input and displays the difference of two address pointers. If wr_ptr is greater than rd_ptr, it finds memory size using relation: ptr_diff = w_ptr - r_ptr.

8.5 Full and Empty Flag Generation logic:

For the required condition, the logic takes ptr_diff as input . For ptr_diff = 0 and same MSB of read pointer and write pointer, empty condition is generated. For ptr_diff = 0 and different

MSBs of read pointer and write pointer, full condition is generated.
In similar manner, almost full and almost empty condition are generated at predecided values. Flag logic generation is immediate without clock delay, as difference between pointers is calculated for both read clock and write clock.

8.6 Test bench for flag logic:

MSB_wr_ptr = 0; MSB_wr_ptr = 0; b_wr_ptr = 3’b000; b_rd_ptr = 3’b000; reset_n = 0;
Output = No operation performs

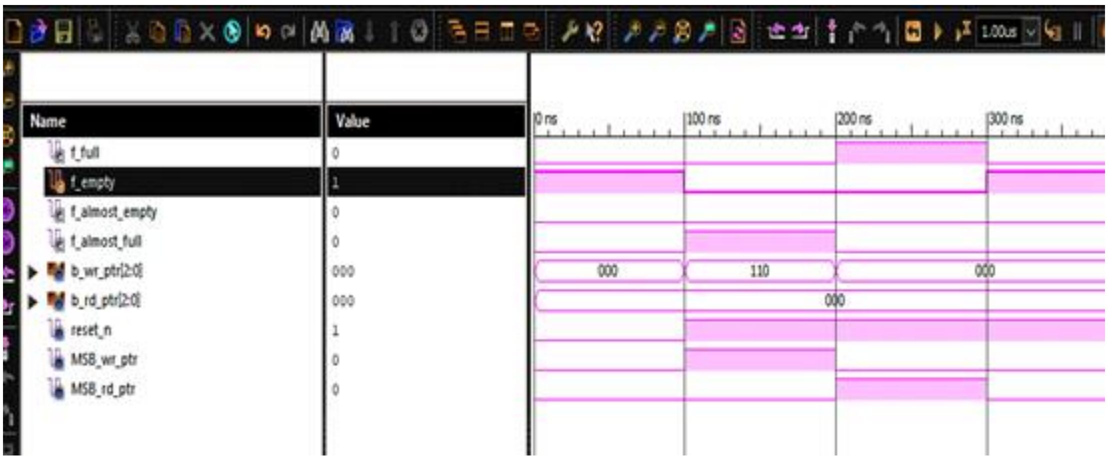
MSB_wr_ptr = 0; MSB_rd_ptr = 0; b_wr_ptr = 3’b000; b_rd_ptr = 3’b000; reset_n = 1;
Output = FIFO is Empty

MSB_wr_ptr = 0; MSB_rd_ptr = 0; b_wr_ptr = 3’b011; b_rd_ptr = 3’b001; reset_n = 1;
Output = FIFO is Almost Empty

MSB_wr_ptr = 1; MSB_rd_ptr = 0; b_wr_ptr = 3’b010; b_rd_ptr = 3’b010; reset_n = 1;
Output = FIFO is Full

MSB_wr_ptr = 1; MSB_rd_ptr = 0; b_wr_ptr = 3’b110; b_rd_ptr = 3’b000; reset_n = 1;
Output = FIFO is almost Full

8.7 Simulation of flag logic:



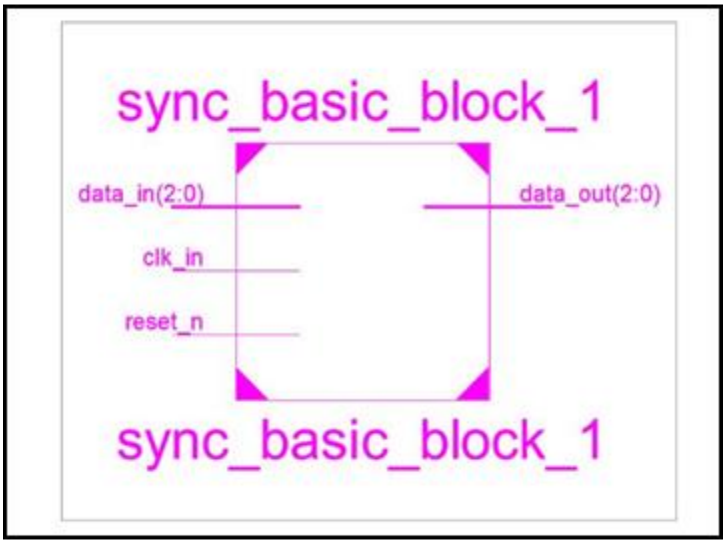
Simulation of flag logic fig 8.3

CHAPTER 9

Synchronizer

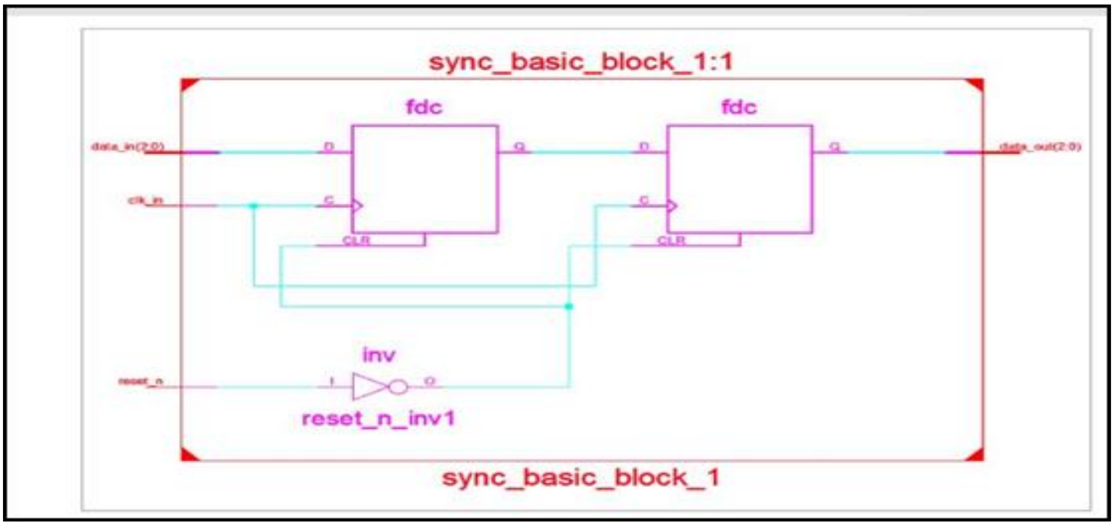
FIFO is operated on two different clock domains different for read and write. Hence, there is a requirement of synchronization between the write pointer and read pointer, for the generation of empty and full logic which can be sequentially used for addressing the FIFO.
Here, the write pointer is passed to D flipflop operating on read clock and the read pointer is given to a D flipflop operating on write clock. Output read pointer which is operated on read clock and synchronized write pointer which is also operated on read clock. Similarly, write pointer and synchronized read pointer operates on read clock.

9.1 Top Level RTL Synchronizer Block:



Top Level RTL of Synchronizer Block fig 9.1

9.2 RTL _Synchronizer Block:



RTL of Synchronizer Block fig 9.2

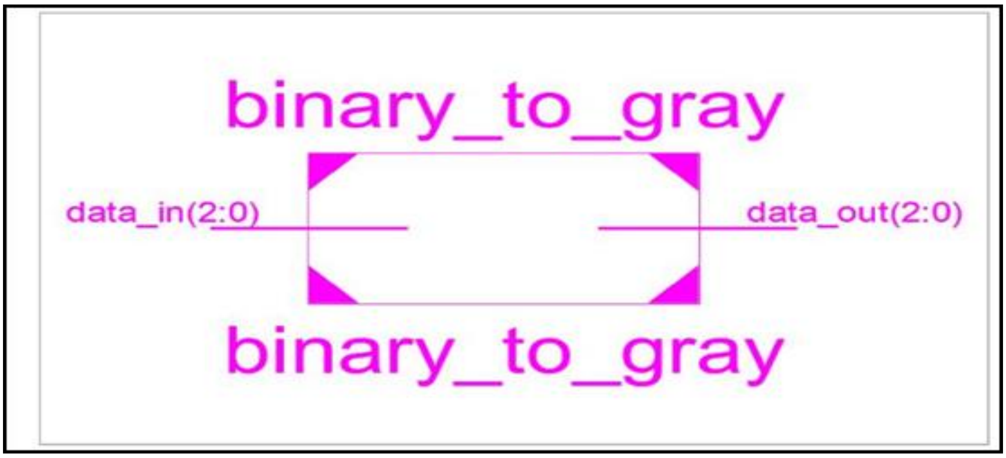
9.3 Synchronization of write pointer:

The Grey code is used to prevent metastability problem between the different clock domains inn synchronizer. Hence, binary code is converted to gray code. The write pointer is converted to Grey code using binary to grey code converter and passed within synchronizer. Later, at output read side, it is encoded back to binary. For generation of empty signal, read pointer is compared with synchronized write pointer.

9.4 Synchronization of read pointer:

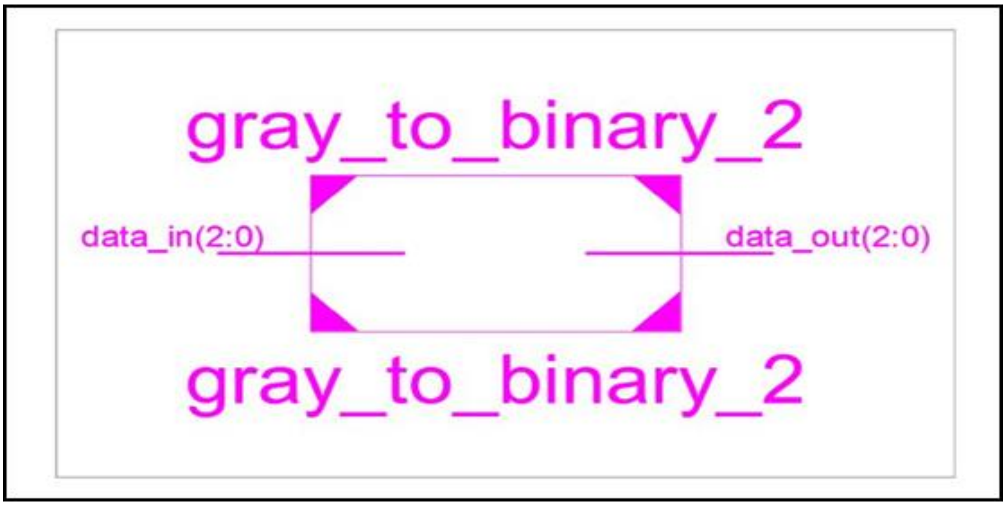
To overcome the different uncertainties during crossing between clock domains, the Grey code is used in synchronization of read pointer. In write clock domain, it prevents malfunctioned read pointer. The read pointer is converted to Grey code using binary to grey code converter and passed within synchronizer. Later, at output write side, it is encoded back to binary. For generation of full signal, write pointer is compared with synchronized read pointer.

9.5 Top Level RTL _ Binary to Gray Converter:



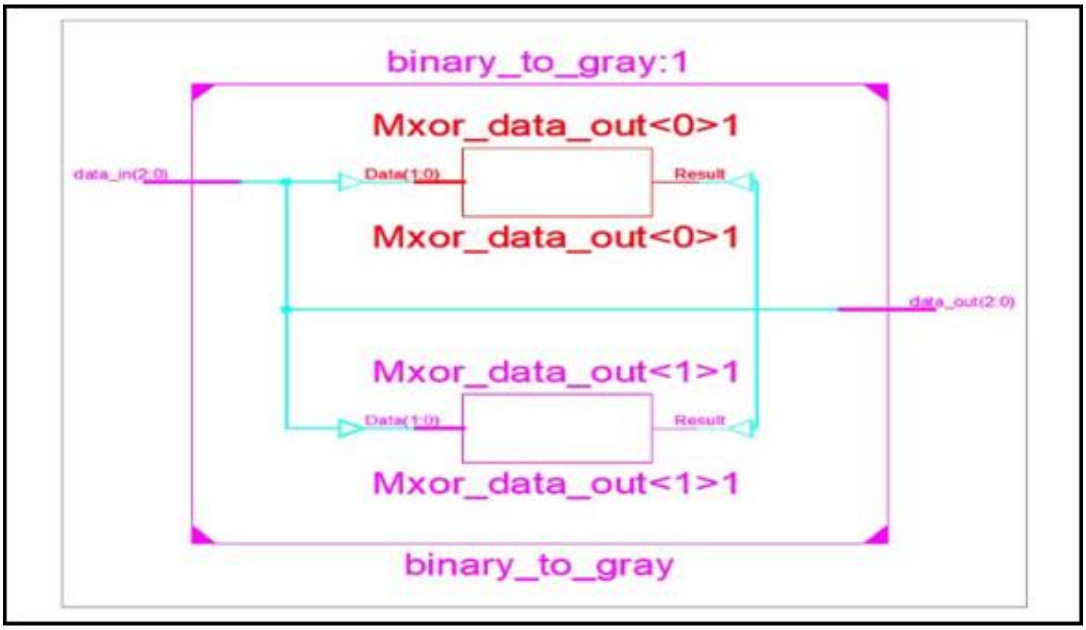
Top Level RTL of Binary to Gray Converter fig 9.3

9.6 Top Level RTL _Gray To Binary Converter:



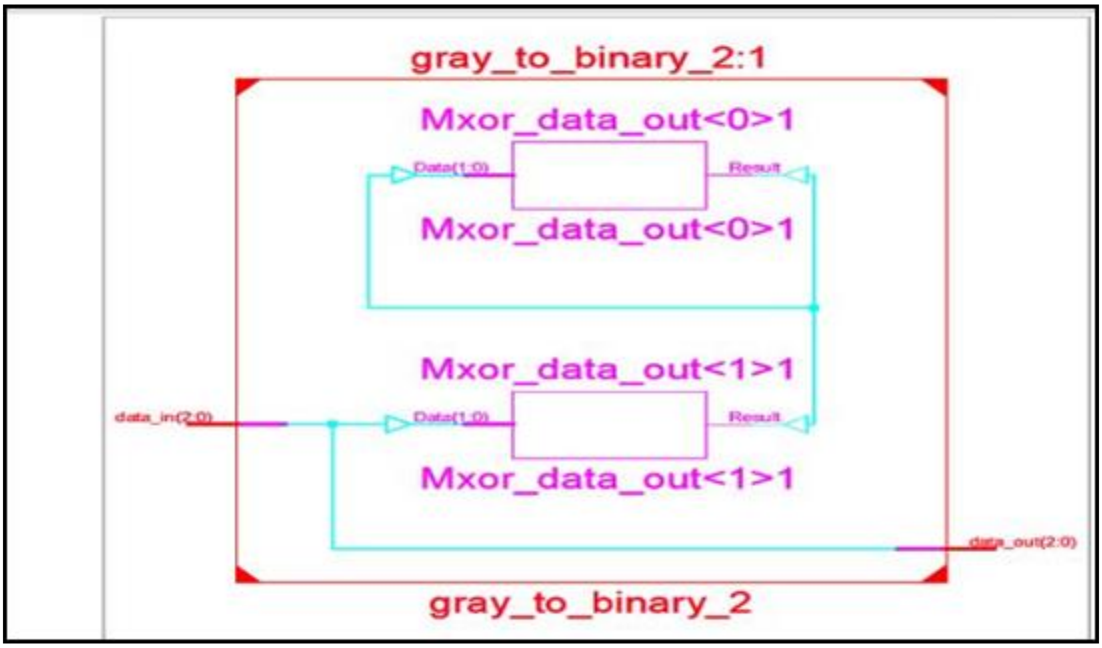
Top Level RTL of Gray to Binary Converter fig 9.4

9.7 RTL _ Binary To Gray Converter:



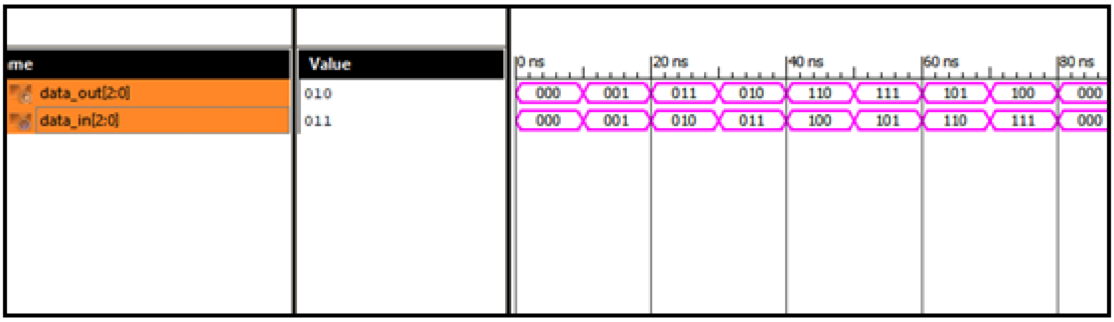
RTL of Binary to Gray Converter fig 9.5

9.8 RTL _Gray To Binary Converter:

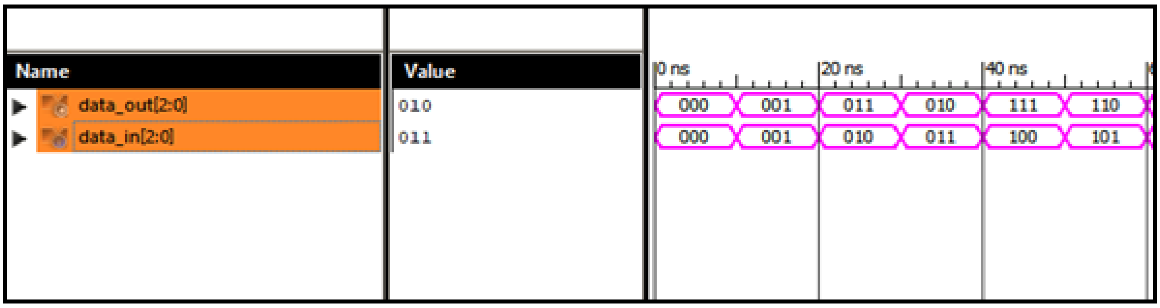


RTL of Gray to Binary Converter fig 9.6

9.9 Simulation waveforms:



Simulation of Binary to Gray Converter fig 9.7



Simulation of Gray to Binary Converter fig 9.8

CHAPTER 10

Synthesis Report

10.1 HDL Synthesis Report:

RAMs: 1

8x8 bit dual port RAM: 1

Adders/ Subtractors: 4

3 bit adder: 2

3 bit add-sub: 2

Counters: 2

4 bit up counter: 2

Registers: 6

3 bit register: 4

8 bit register: 2

Comparators: 2

3 bit comparator greater/equal: 2

Tristates: 8

1 bit tristate buffer: 8

Xors: 10

1 bit xor2ip : 10

Final Register Report:

Registers: 28

Flipflops: 28

10.2 Final Report:

Parameters used:

RTL Top Level Output File Name: fifo_top.ngr

Top Level Output File Name: fifo_top

Output Format: NGC

Optimization Goal: Normal

Keep Hierarchy : No

Design Statistics

1. IOs : 25
2. Cell Usage: Not mentioned
3. BELS: 54
4. GND: 1
5. INV: 3
6. LUT2: 7
7. LUT2_D: 1
8. LUT3: 8
9. LUT4: 25
10. LUT4_D: 3
11. LUT4_L: 2
12. MUXF5: 4
13. Flipflops / Latches: 28
14. FDC: 12
15. FDCE: 8
16. FDE: 8
17. RAMS: 8
18. RAM16X1D: 8
19. Clock Buffers: 2
20. BUFGP: 2
21. IO Buffers: 23
22. IBUF: 11
23. OBUF: 12

10.3 Device utilization summary:

- Selected Device: 3s500efg320-5
- Number of Slices: 35 out of 4656 [0%]
- Number of Slice Flip Flops: 28 out of 9312 [0%]
- Number of 4 input LUTs: 65 out of 9312 [0%]
- Number used as logic: 49
- Number used as RAMs: 16
- Number of IOs: 25
- Number of bonded IOBs: 25 out of 232 [10%]
- Number of GCLKs: 2 out of 24 [8%]

10.4 Partition Resource Summary:

No Partitions were found in this design.

CHAPTER 11

Conclusion

11.1 Advantages

- Data Rate Matching
- Buffering and bus matching parallel FIFO structure allow formulation of any word size while serial FIFO structure provide a rapid and simple structure.
- Inter-chip communication protocol problem is solved by FIFO.
- Only circuitry required is to create fast FIFO, control logic, flag logic.

11.2 Disadvantages

- Noise in signals is caused by switching operation and alteration of memory content.
- Synchronization circuit is required to synchronize WRITE CLOCK and READ CLOCK.

11.3 General Applications

For performing numerous system tasks, the FIFO is used preferably for sequential operations.

- **Frequency Coupling –** Digital system operates on different frequencies. When there is transfer from transmitter and receive with different frequency domains, FIFO can be used for coupling of frequency. It receives data at certain frequency and provides at another frequency. The different read clock signals and write clock signals are used for controlling the data transfer speed of various inputs and outputs.

- **Packet Buffering** – Until FIFO output is ready to communicate, It stores previously written data in System, which means buffering the data input from the source in FIFO, till the output port starts to accept the data. It can be used for routing arrangements or switching of network with multiple FIFOs with multiple input paths and a common output bus, where outputs are polled by the receiving network.

- **Bus Matching** – In different digital domains, working on different clock frequencies and bus width, while transferring data FIFO works as a commutator.

11.4 Conclusion

- The asynchronous FIFO is implemented using Xilinx ISE 14.7 on Verilog was used for synthesis. The data width in asynchronous FIFO is eight bits, while FIFO depth is one kilobyte.
- An asynchronous FIFO structure appropriate for RTL modeling implementation using the standard design tools and design flow is presented. An approach for construction of the asynchronous circuits using Verilog HDL is presented. This approach uses construction of RTL model using applicable basic blocks like flipflops and latches, which is represented with verilog code.
- The only drawback of the presented asynchronous FIFO design is inefficient as timing model constructed using circuit level. (Output is one cycle delayed in flag model).

CHAPTER 12

Contribution and Future Scope

12.1 Contribution:

The project was aimed at implementation of a FIFO (First In First Out) memory on the FGPA (Field Programmable Gate Array) Board. We have referred a variety of documents from various sources, including the research papers by IEEE (Institute of Electrical and Electronics Engineers) and prominent names from VLSI (Very Large Scale Integration) industry such as Clifford E. Cummings, also some books for basic understanding of the topic. The RTL diagram and various simulations are completed using Verilog HDL (Hardware Description Language). This whole work of the project provides a detailed understanding of the FIFO memories, their types and working. The results of the project also contribute to an understanding differences between using the asynchronous and synchronous memories. This design of the FIFO is a multiple clock level design including two different clock signals, so this project also contributes to provide a method to deal with such situation and problems related with it. The RTL design presented here provides a way to reduce the code length and aims to make the design more flexible, so called parameterized design.

12.2 Future Scope:

We have left all the options open so that if there is any other future requirement in the project by the user for the enhancement of the project then it is possible to implement them. As the design is made flexible and parameterized so that further modifications are possible. The design can be modified to give FIFO memory of any size and any new features can be added very easily. Even the terminal conditions can be changed just by altering the RTL design code. In the last we would like to express our humble gratitude to all the persons involved in the development of the project directly or indirectly. We are also thankful to the department for so much taken by them in helping to develop the project. We hope that the project will serve its purpose for which it is developed there by underlining success of process.

References

- Peter Forstner Mixed Signal Logic Products by Texas Instruments.
- Clifford E.cummings and Peter Alfke ,”Simulation and synthesis Techniques for Asynchronous FIFO Design with Asynchronous pointer comparisons, “SNUG 2002,SectionTB2,3rd paper.
- Clifford E.cummings, Synthesis and Scripting Techniques for Designing Multi-Asynchronous Clock Designs, SNUG 2001,Section MCI ,3rd paper.
- J.Ebergen “Squaring the FIFO in GasP,”in 7th International Conference on Asynchronous Circuit and System , March 2001.
- HDL Synthesis Report from Xillinx ISE 14.7 Project work.