JSPM's
Imperial College of Engineering and Research, Wagholi, Pune.
(Approved by AICTE, Delhi & Govt. of Maharashtra, affiliated to Savitribai Phule Pune University)
Gat.No.720,Pune-Nagar road,Wagholi,Pune,412207
Phone No. 020-67335102 **website**: www.icoer.in **Email**- principal.imperial2016@gmail.com
Department of Electronics and Telecommunication Engineering

Experiment No. 1

Title: To Write VHDL code, simulate with test Bench, Synthesis, for 4 Bit ALU for add, subtract, AND, NAND, OR, XOR & XNOR.

Date of Performance: **Roll No:**

Date of Submission: **University Seat No:**

Signature of Staff:

JSPM's
Imperial College of Engineering and Research, Wagholi, Pune.
(Approved by AICTE, Delhi & Govt. of Maharashtra, affiliated to Savitribai Phule Pune University)
Gat.No.720,Pune-Nagar road,Wagholi,Pune,412207
Phone No. 020-67335102 **website**: www.icoer.in **Email**- principal.imperial2016@gmail.com
Department of Electronics and Telecommunication Engineering

Experiment No. 1

4 Bit ALU for add, subtract, AND, NAND, OR, XOR & XNOR.

Title: To Write VHDL code, Simulate with test Bench, Synthesis, of 4 Bit ALU add, subtract, AND, NAND, OR, XOR &XNOR.

Aim:

Write a VHDL Code for 4 Bit ALU which Consist of Arithmetic and Logical Operations which includes add, subtract, AND, NAND, XOR, XNOR, OR, & ALU pass.

Equipments Required:

Software Required: Xilinx ISE
Hardware Required: PC, Spartan-3 Protoboard.

Theory:

Introduction:-

The design and implementation of FPGA based Arithmetic Logic Unit is of core significance in digital technologies as it is an integral part of central processing unit. ALU is capable of calculating the results of a wide variety of basic arithmetical and logical computations. The ALU takes, as input, the data to be operated on (called operands) and a code, from the control unit, indicating which operation to perform. The output is the result of the computation.

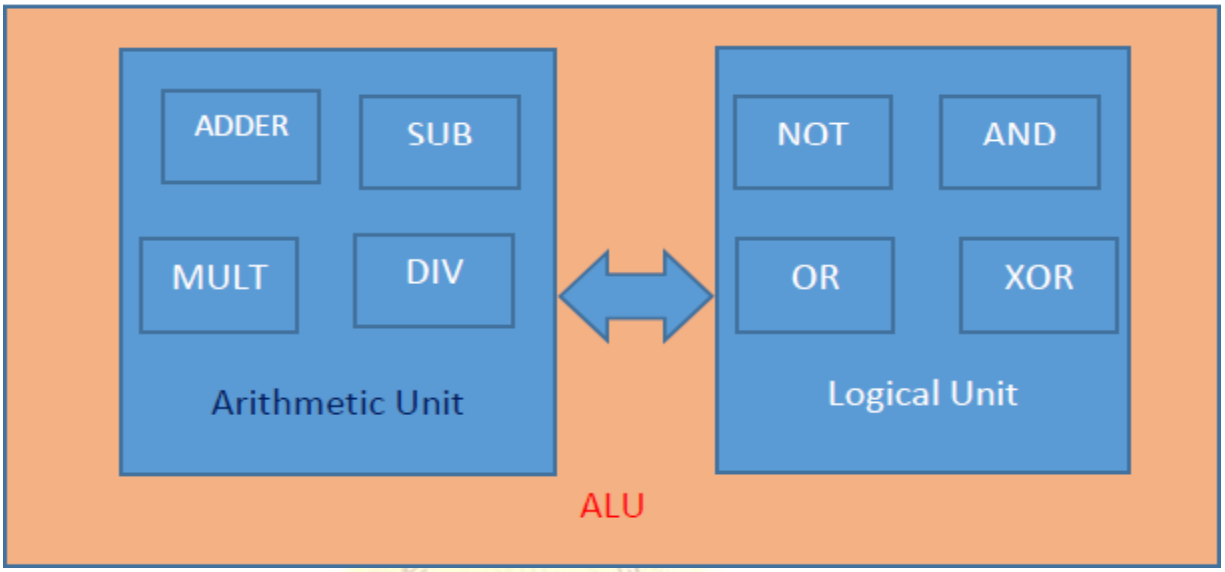
Designed ALU will perform the following operations:

- Arithmetic operations
- Bitwise logic operations

Operation of ALU:

There are two kinds of operation which an ALU can perform first part deals with arithmetic computations and is referred to as Arithmetic Unit. It is capable of addition, subtraction, multiplication, division, increment and decrement. The second part deals with the Gated results in the shape of AND, OR, XOR, inverter, rotate, left shift and right shift,

which is referred to as Logic Unit. The functions are controlled and executed by selecting operation or control bits.



Design Description:-

Truth Table of ALU:

OPCODE	OPERATION	SPECIFICATIONS
0000	A	Y is assigned the value of A (input)
0001	NOT A	Y is assigned the value of NOT A
0010	B	Y is assigned the value of B (input)
0011	NOT B	Y is assigned the value of NOT B
0100	A AND B	Y is assigned the value of A AND B
0101	A OR B	Y is assigned the value of A OR B
0110	A NAND B	Y is assigned the

		value of A NAND B
0111	A NOR B	Y is assigned the value of A NOR B
1000	A XOR B	Y is assigned the value of A XOR B
1001	A+1	Y is assigned the value of increment A
1010	B+1	Y is assigned the value of increment B
1011	A + B	Y is assigned the value of A + B
1100	A - 1	Y is assigned the value of A – 1
1101	B - 1	Y is assigned the value of B-1
1110	A - B	Y is assigned the value of A – B
1111	A XNOR B	Y is assigned the value of A XNOR B

Experimental Procedure:

1. Start the Xilinx Project Navigator by using the desktop shortcut or by using the Start a Program a Xilinx ISE a Project Navigator
2. In the Project Navigator window go to FILE → New project → Give name → Next → Project settings → Next → Spartan 3 → Next → Finish.
3. Click on the symbol of FPGA device and then right Click on new source → Select VHDL module → Give the name ALU → click on Next.
4. Generate the Behavioral VHDL Code for ALU
5. Check syntax, and remove errors if present.
6. Synthesize the design using XST.

   Highlight ALU file in the Sources in Project window. To run synthesis, right click on Synthesize, and choose the Run option, or double-click on Synthesize in the Processes for Current Source window. Synthesis will run, and a green check will

appear next to Synthesize when it is successfully completed. A yellow exclamation mark indicates that a warning was generated, and a red cross indicates an error was generated. Warnings are OK. If there are any errors, you can view the error through the console window otherwise continue on to the next step.

7. For creating Test Bench → Right click on Device → Select New Source→ Test Bench Waveform → Enter file name → Next →Finish.
8. Apply input as per requirement and save.
9. Double click on Behavioral Check Syntax. If process is complete successfully → Double click on simulate Behavioral model.
10. For Implementation Constraint file (UCF). Right click on Device→ Select New Source → Give File Name → Next → Source type Window appears.
11. After double click on I/O pin planning (Plan Ahead) Pre- Synthesis. → Click on Yes.
12. Assign the package pins using reference manual → save, the UCF file will be added in your project.
13. Click on Save and Exit the plan Ahead.
14. After successful implementation → Double click on 'Generating programming file'
15. For downloading .bit file working with USB JTAG cable.
16. Go to Flash Programmer Folder then double click on CDMV2.12.00 WHQL Certified.
17. Double Click on 'CDMV2.12.00 WHQL Certified' then FTDI CDM Drivers.
18. Click on Extract→ Next → accept agreement → Next → Finish.
19. Click on browse button and select respective .bit file of program and click on program button.
20. After downloading program successfully 'Green LED (LD1) will turned ON' on Board.
21. To download the new program Press 'RESET (RST)' button on Board.

Conclusion:

JSPM's
Imperial College of Engineering and Research, Wagholi, Pune.
(Approved by AICTE, Delhi & Govt. of Maharashtra, affiliated to Savitribai Phule Pune University)
Gat.No.720,Pune-Nagar road,Wagholi,Pune,412207
Phone No. 020-67335102 **website**: www.icoer.in **Email**- principal.imperial2016@gmail.com
Department of Electronics and Telecommunication Engineering

Experiment No. 2

Title: To write VHDL code, simulate with test bench, synthesis for MOD-N Counter.

Date of Performance: **Roll No:**

Date of Submission: **University Seat No:**

Signature of Staff:

Experiment No. 2

To write VHDL code, simulate with test bench, synthesis for MOD-N Counter.

Title: To write VHDL code, simulate with test bench, synthesis for MOD-N Counter.

Aim:

Write VHDL code, simulate with test bench, synthesis, for MOD-8 up Counter.

Equipment's Required:

Software Required: Xilinx ISE

Theory:

Introduction:-

The job of a counter is to count by advancing the contents of the counter by one count with each clock pulse. Counters which advance their sequence of numbers or states when activated by a clock input are said to operate in a "count-up" mode. Likewise, counters which decrease their sequence of numbers or states when activated by a clock input are said to operate in a "countdown" mode. Counters that operate in both the UP and DOWN modes are called bidirectional counters. Counters are sequential logic devices that are activated or triggered by an external timing pulse or clock signal. A counter can be constructed to operate as a synchronous circuit or as an asynchronous circuit. With synchronous counters, all the data bits change synchronously with the application of a clock signal. Whereas an asynchronous counter circuit is independent of the input clock so the data bits change state at different times one after the other. Then counters are sequential logic devices that follow a predetermined sequence of counting states which are triggered by an external clock (CLK) signal. The number of states or counting sequences through which a particular counter advances before returning once again back to its original first state is called the modulus (MOD). In other words, the modulus (or just modulo) is the number of states the counter counts and is the dividing number of the counter. Modulus Counters, or simply MOD

counters, are defined based on the number of states that the counter will sequence through before returning back to its original value. For example, a 2-bit counter that counts from 002 to 112 in binary, that is 0 to 3 in decimal, has a modulus value of 4 (00 → 1 → 10 → 11, and return back to 00) so would therefore be called a modulo-4, or mod-4, counter. Note also that it has taken four clock pulses to get from 00 to 11. Therefore, a "Mod-N" counter will require "N" number of flip-flops connected together to count a single data bit while providing 2n different output states, (n is the number of bits). Note that N is always a whole integer value.

Experimental Procedure:

1. Start the Xilinx Project Navigator by using the desktop shortcut or by using the Start a Program a Xilinx ISE a Project Navigator
2. In the Project Navigator window go to FILE → New project → Give name → Next → Project settings → Next → Spartan 3 → Next → Finish.
3. Click on the symbol of FPGA device and then right Click on new source → Select VHDL module → Give the name mod8 → click on Next.
4. Generate the Behavioral VHDL Code for mod8
5. Check syntax, and remove errors if present.
6. Synthesize the design using XST.
   Highlight mod8 file in the Sources in Project window. To run synthesis, right click on Synthesize, and choose the Run option, or double-click on Synthesize in the Processes for Current Source window. Synthesis will run, and a green check will appear next to Synthesize when it is successfully completed. A yellow exclamation mark indicates that a warning was generated, and a red cross indicates an error was generated. Warnings are OK. If there are any errors, you can view the error through the console window otherwise continue on to the next step.
7. For creating Test Bench → Right click on Device → Select New Source→ Test Bench Waveform → Enter file name → Next →Finish.
8. Apply input as per requirement and save.
9. Double click on Behavioral Check Syntax. If process is complete successfully → Double click on simulate Behavioral model.
10. For Implementation Constraint file (UCF). Right click on Device→ Select New Source → Give File Name → Next → Source type Window appears.
11. After double click on I/O pin planning (Plan Ahead) Pre- Synthesis. → Click on Yes.
12. Assign the package pins using reference manual → save, the UCF file will be added in your project.

13. Click on Save and Exit the plan Ahead.
14. After successful implementation → Double click on 'Generating programming file'
15. For downloading .bit file working with USB JTAG cable.
16. Go to Flash Programmer Folder then double click on CDMV2.12.00 WHQL Certified.
17. Double Click on 'CDMV2.12.00 WHQL Certified' then FTDI CDM Drivers.
18. Click on Extract→ Next → accept agreement → Next → Finish.
19. Click on browse button and select respective .bit file of program and click on program button.
20. After downloading program successfully 'Green LED (LD1) will turned ON' on Board.
21. To download the new program Press 'RESET (RST)' button on Board.

Conclusion:

JSPM's
Imperial College of Engineering and Research, Wagholi, Pune.
(Approved by AICTE, Delhi & Govt. of Maharashtra, affiliated to Savitribai Phule Pune University)
Gat.No.720,Pune-Nagar road,Wagholi,Pune,412207
Phone No. 020-67335102 **website**: www.icoer.in **Email**- principal.imperial2016@gmail.com
Department of Electronics and Telecommunication Engineering

Experiment No. 3

Title: To Write VHDL code of Universal Shift register mode selection input for SISO, SIPO, PISO, & PIPO modes.

Date of Performance: **Roll No:**

Date of Submission: **University Seat No:**

Signature of Staff:

Experiment No. 3

UNIVERSAL SHIFT REGISTER

Title: To Write VHDL code of Universal Shift register mode selection input for SISO, SIPO, PISO, & PIPO modes.

Aim:

1. To study Shift register
2. Universal Shift register mode selection input for SISO, SIPO, PISO, & PIPO modes.

Equipments Required:

Software Required: Xilinx ISE
Hardware Required: PC, Spartan-3 Protoboard.

Theory:

Shift registers are a type of sequential logic circuits, mainly for storage of digital data. They are group of FF's connected in chain so that o/p from one FF becomes i/p to next FF. All FF's are driven by a common clock and all are set or reset simultaneously. Data can be entered in serial or parallel form or can be retrieved in serial or parallel form. Shift registers are used for data storage, data transfer and arithmetic and logic operations.

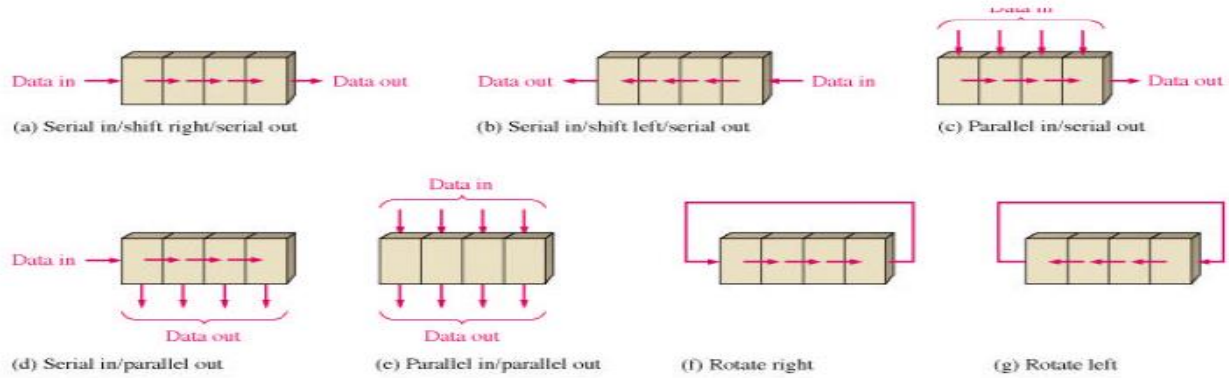


Fig.2.1 Basic data movement in shift registers

Design Description:-

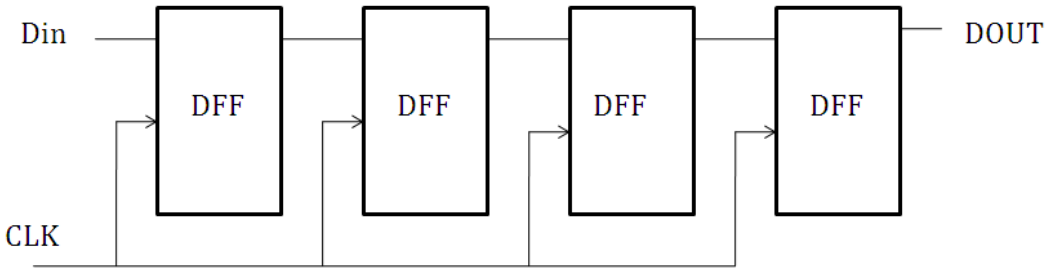


Fig. 2.2: Block diagram of 4 bit shift register

Truth Table fort Flip Flop:

CLR	CLK	Din	S3	S2	S1	S0
1	1	1	0	0	0	0
0	2	0	1	0	0	0
0	3	0	0	1	0	0
0	4	0	0	0	1	0
0	5	0	0	0	0	1

Serial-In – Serial-Out Shift Registers

The serial in/serial out shift register accepts data serially – that is, one bit at a time on a single line. It produces the stored information on its output also in serial form.

A basic four-bit shift register can be constructed using four D flip-flops, as shown in Figure 2.1.

The operation of the circuit is as follows.

- The register is first cleared, forcing all four outputs to zero.
- The input data is then applied sequentially to the D input of thefirst flip-flop on the left (FF0).
- During each clock pulse, one bit is transmitted from left to right.
- Assume a data word to be 1001.

- The least significant bit of the data has to be shifted through the register from FF0 to FF3.

In order to get the data out of the register, they must be shifted out serially. This can be done destructively or non-destructively. For destructive readout, the original data is lost and at the end of the read cycle, all flip-flops are reset to zero.

FF0	FF1	FF2	FF3	WRITE
0	0	0	0	1001

The data is loaded to the register when the control line is HIGH (i.e. WRITE). The data can be shifted out of the register when the control line is LOW (i.e. READ). Clear

FF0	FF1	FF2	FF3	READ
1	0	0	1	0000

WRITE:

FF0	FF1	FF2	FF3	
1	0	0	1	0000

READ:

FF0	FF1	FF2	FF3	
1	0	0	1	1001

Serial In - Parallel Out Shift Registers

For this kind of register, data bits are entered serially in the same manner as discussed in the last section. The difference is the way in which the data bits are taken out of the register. Once the data are stored, each bit appears on its respective output line, and all bits are available simultaneously. A construction of a four-bit serial in - parallel out register is shown below.

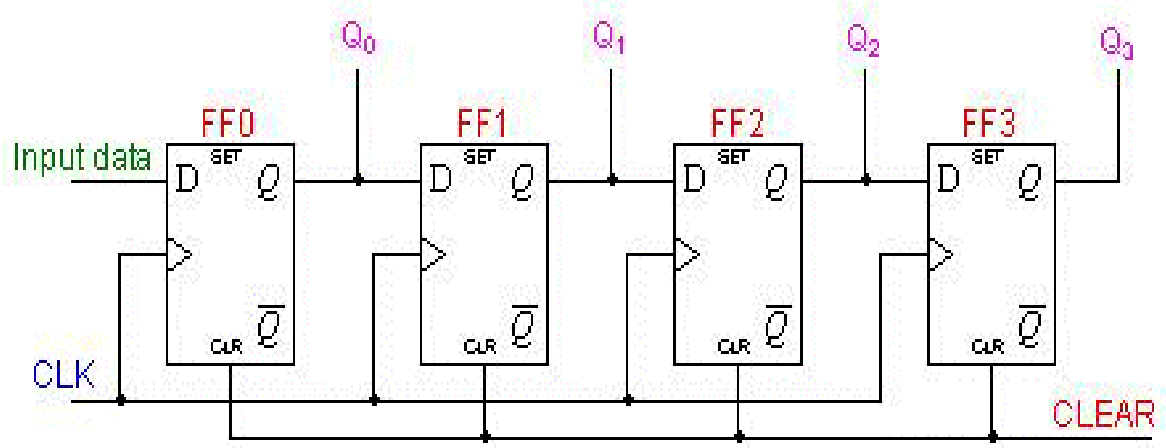


Fig. 2.3 Serial In - Parallel Out Shift Registers

In the table below, we can see how the four-bit binary number 1001 is shifted to the Q outputs of the register.

CLEAR	FF0	FF1	FF2	FF3
1001	0	0	0	0
	1	0	0	0
	0	1	0	0
	0	0	1	0
	1	0	0	1

Parallel In – Serial-Out Shift Registers

A four-bit parallel in - serial out shift register is shown below. The Circuit uses D flip-flops and NAND gates for entering data (i.e. writing) to the register.

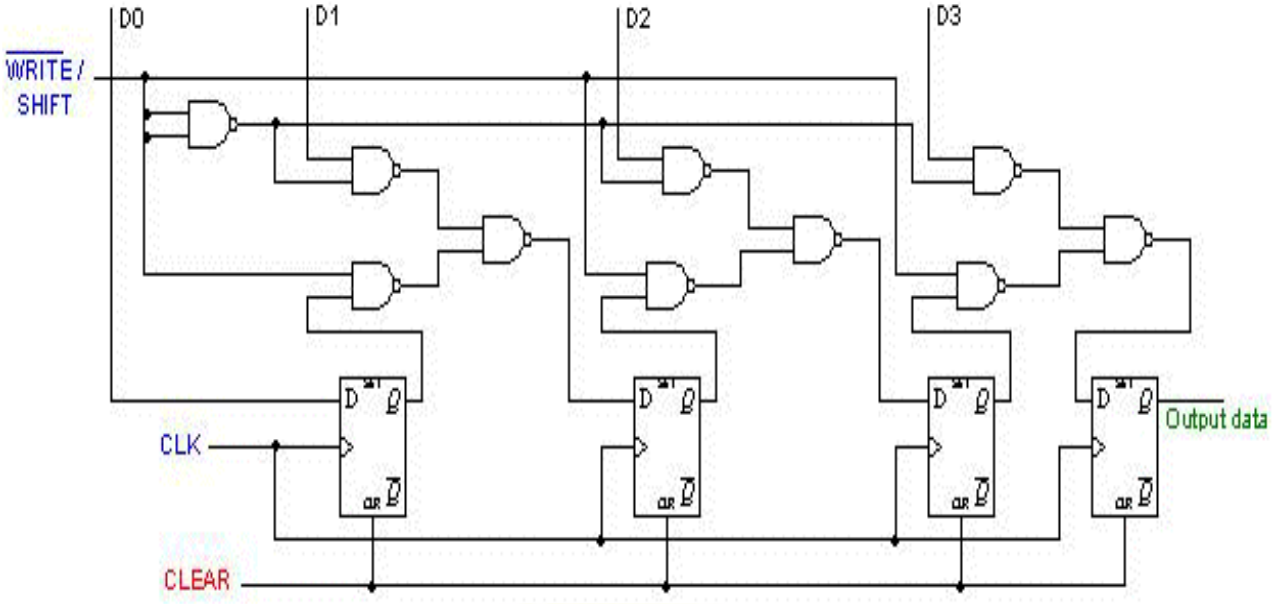


Fig.2.4 Parallel In - Serial Out Shift Registers

D0, D1, D2 and D3 are the parallel inputs, where D0 is the most significant bit and D3 is the least significant bit. To write data in, the mode control line is taken to LOW and the data is clocked in. The data can be shifted when the mode control line is HIGH as SHIFT is active high. The register performs right shift operation on the application of a clock pulse, as shown in the table below.

	Q0	Q1	Q2	Q3	
CLEAR	0	0	0	0	
WRITE	1	0	0	1	
SHIFT	1	0	0	1	
	1	1	0	0	1
	1	1	1	0	01
	1	1	1	1	001
	1	1	1	1	1001

Parallel In - Parallel Out Shift Registers

For parallel in - parallel out shift registers, all data bits appear on the parallel outputs immediately following the simultaneous entry of the data bits. The following circuit is a four-bit parallel in - parallel out shift register constructed by D flip-flops.

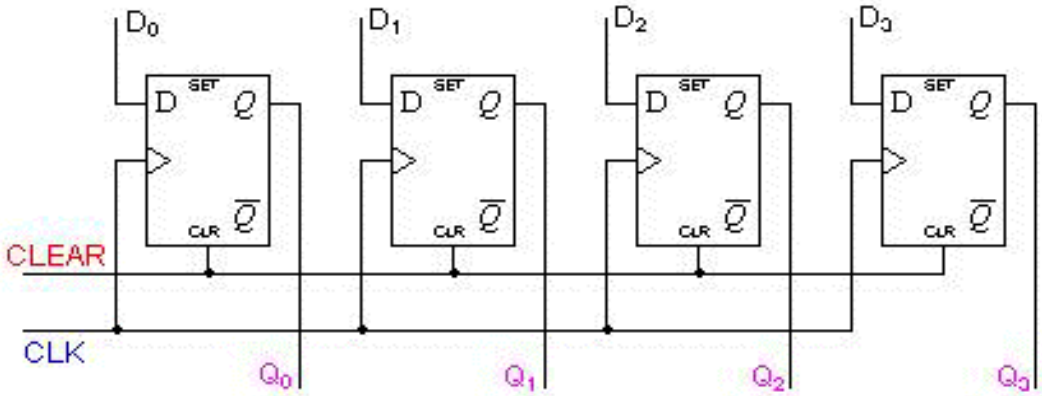


Figure 2.5

The D's are the parallel inputs and the Q's are the parallel outputs. Once the register is clocked, all the data at the D inputs appear at the corresponding Q outputs simultaneously.

Experimental Setup:

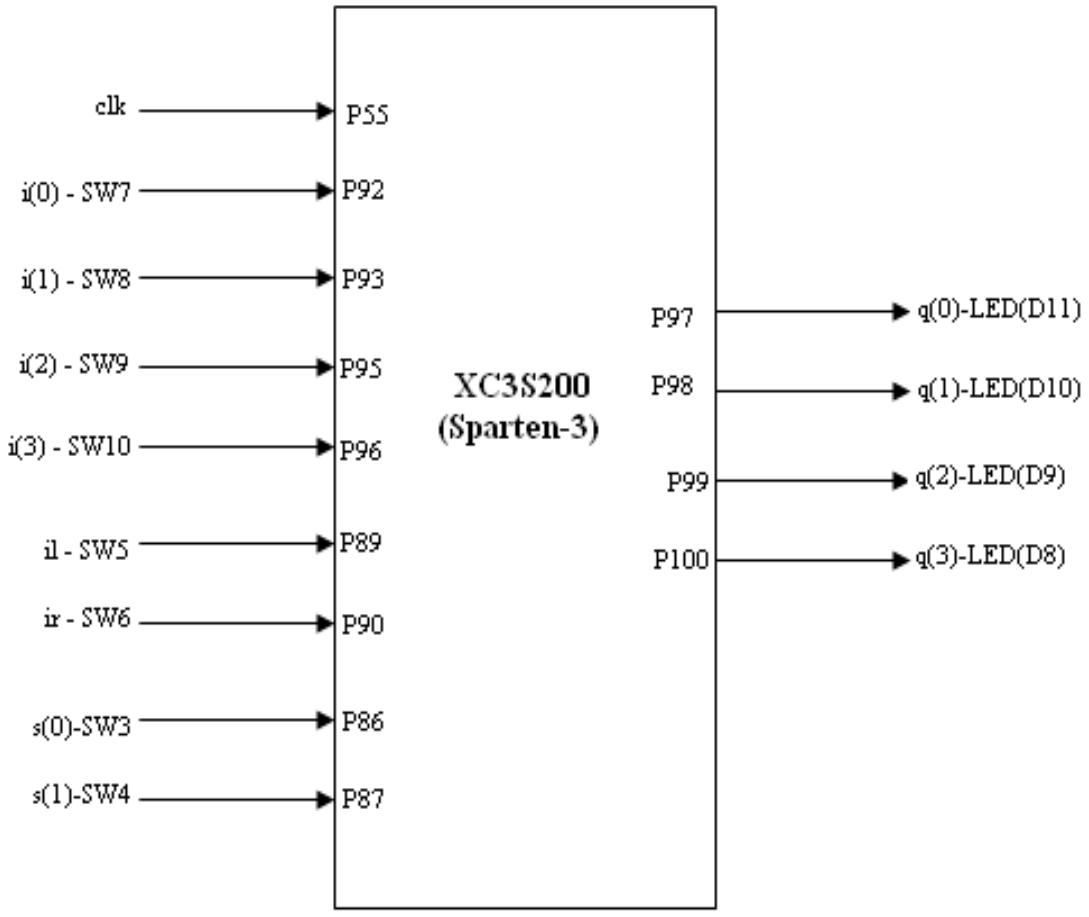


Fig. 2.1 Universal Shift Register

Experimental Procedure:

1. Start the Xilinx Project Navigator by using the desktop shortcut or by using the
2. Start a Program a Xilinx ISE a Project Navigator
3. In the Project Navigator window go to FILE → New project → Give name → Next → Project settings → Next → Spartan 3 → Next → Finish.
4. Click on the symbol of FPGA device and then right Click on new source →
5. Select VHDL module → Give the name Shift_regd. → click on Next.
6. Generate the Behavioral VHDL Code for Shift_regd.
7. Check syntax, and remove errors if present.

8. Synthesize the design using XST.

9. Highlight Shift_regd.vhd file in the Sources in Project window. To run synthesis, right-click on Synthesize, and choose the Run option, or double-click on Synthesize in the Processes for Current Source window. Synthesis will run, and a green check will appear next to Synthesize when it is successfully completed. A yellow exclamation mark indicates that a warning was generated, and a red cross indicates an error was generated. Warnings are OK. If there are any errors, you can view the error through the console window otherwise continue on to the next step.

10. For creating Test Bench → Right click on Device → Select New Source→ Test Bench Waveform → Enter file name → Next →Finish.

11. Apply input as per requirement and save.

12. Double click on Behavioral Check Syntax. If process is complete successfully → Double click on simulate Behavioral model.

13. For Implementation Constraint file (UCF). Right click on Device→ Select New Source → Give File Name → Next → Source type Window appears.

14. After double click on I/O pin planning (Plan Ahead) Pre- Synthesis. → Click on Yes.

15. Assign the package pins using reference manual → save, the UCF file will be added in your project.

16. Click on Save and Exit the plan Ahead.

17. After successful implementation → Double click on 'Generating programming file'

18. For downloading .bit file working with USB JTAG cable.

19. Go to Flash Programmer Folder then double click on CDMV2.12.00 WHQL Certified.

20. Double Click on 'CDMV2.12.00 WHQL Certified' then FTDI CDM Drivers.

21. Click on Extract→ Next → accept agreement → Next → Finish.

22. Click on browse button and select respective .bit file of program and click on program button.

23. After downloading program successfully 'Green LED (LD1) will turned ON' on Board.

24. To download the new program Press 'RESET (RST)' button on Board.

Apply input through DIP Switches and observe the output displayed on LEDs.

Conclusion:

JSPM's
Imperial College of Engineering and Research, Wagholi, Pune.
(Approved by AICTE, Delhi & Govt. of Maharashtra, affiliated to Savitribai Phule Pune University)
Gat.No.720,Pune-Nagar road,Wagholi,Pune,412207
Phone No. 020-67335102 **website**: www.icoer.in **Email**- principal.imperial2016@gmail.com
Department of Electronics and Telecommunication Engineering

Experiment No. 4

Title: To study VHDL code for LCD interface.

Date of Performance: **Roll No:**

Date of Submission: **University Seat No:**

Signature of Staff:

JSPM's
Imperial College of Engineering and Research, Wagholi, Pune.
(Approved by AICTE, Delhi & Govt. of Maharashtra, affiliated to Savitribai Phule Pune University)
Gat.No.720,Pune-Nagar road,Wagholi,Pune,412207
Phone No. 020-67335102 **website**: www.icoer.in **Email**- principal.imperial2016@gmail.com
Department of Electronics and Telecommunication Engineering

Experiment No. 4

LCD INTERFACE

Title: To study VHDL code for LCD interface.

Aim:

1. To Study the VHDL code for LCD.
2. To study LCD interface for CPLD/FPGA.

Equipments Required:

Software Required: Xilinx ISE
Hardware Required: PC, Spartan-3 Protoboard.

Theory:

Introduction:-

This LCD controller is a VHDL component for use in CPLDs and FPGAs. VHDL component allows simple LCD integration into practically any programmable logic application. Figure 1 depicts the controller implemented to interface between an LCD module and a user's custom logic.

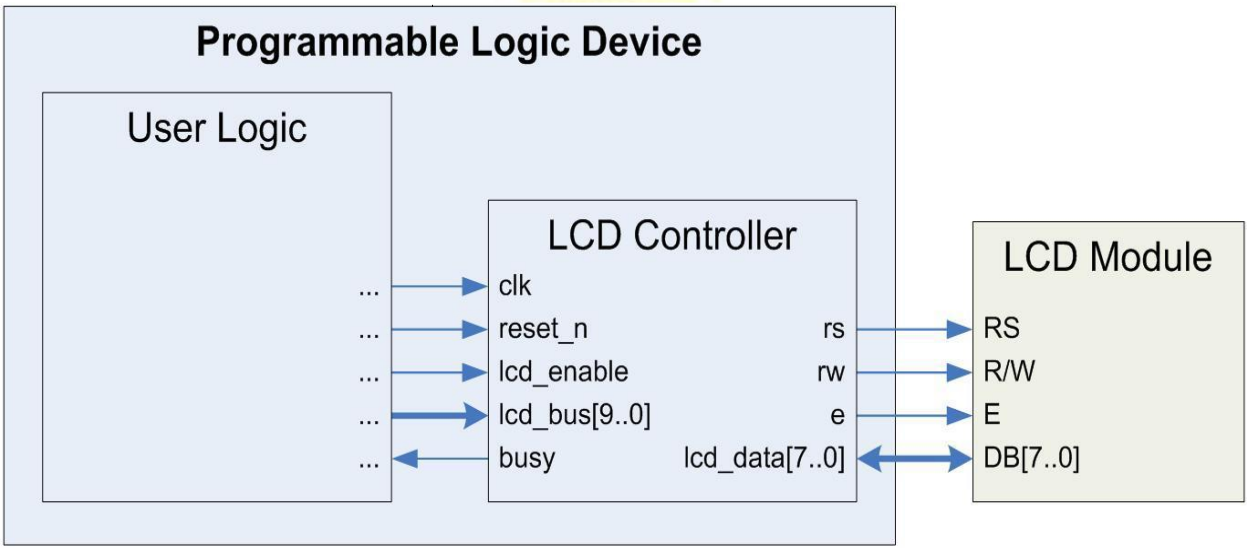
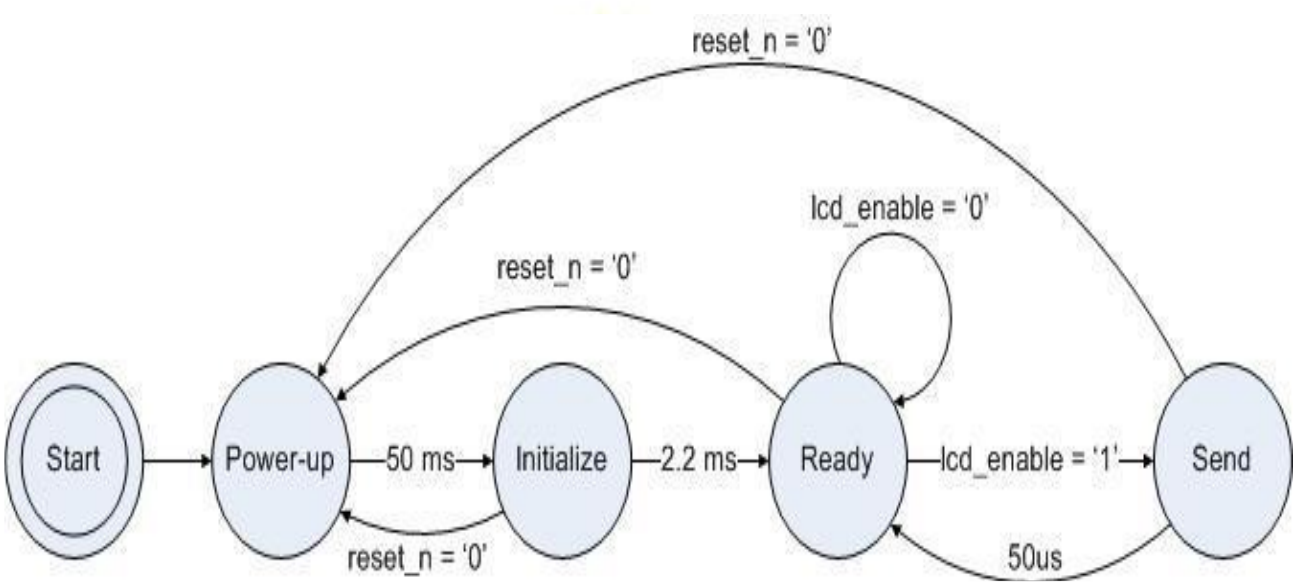


Figure 3.1. LCD Controller Implementation in PLD

State Machine:

The LCD controller state machine consists of five states. Upon start up, it immediately enters the Power-up state, where it waits 50ms to ensure the supply voltage has stabilized.It then proceeds to an Initialize state. The controller cycles the LCD through its initialization sequence, setting the LCD's parameters to default values defined in the hardware. This process completes in approximately 2.2ms, and the controller subsequently assumes a Ready state. It waits in this state until the lcd_enable input is asserted, then advances to the Send state. Here, it communicates the appropriate information to the LCD, as defined by the lcd_bus input. After 50us, it returns to the Ready state until further notice. If a low logic level is applied to the reset_n input at any time for a minimum of one clock cycle, the controller resets to the Power-up state and re-initializes. Figure 2 illustrates the LCD controller state machine.



Port Descriptions:

Table 1 describes the LCD controller's interface.

I/O Name	Width	Mode	Description	Interface
clk	1	input	Clock for LCD controller. Default set for 50MHz. If a different frequency is desired, change the constant freq in the architecture declarations to reflect the new frequency in MHz.	system clock
reset_n	1	input	Active low synchronous reset pin. This pin must be set high to	user logic

			implement the LCD controller. Setting the pin low for one or more clock cycles restarts the LCD controller state machine.	
lcd_enable	1	input	Data latch for LCD controller. H: initiates a transaction using the data currently on the lcd_bus, L: no transaction is initiated and any data on lcd_bus is ignored	user logic
lcd_bus	10	input	Data/instructions to be sent to the LCD module. The MSB is the rs signal, followed by the rw signal. The other 8 bits are the data bits. The LSB on the bus corresponds to the least significant data bit.	user logic
busy	1	output	Feedback on the state of the LCD controller. H: the controller is busy initializing or conducting a transaction with the LCD module, any instructions/data sent will be ignored, L: the controller is idle and ready to accept commands for a transaction	user logic
rs	1	output	LCD module Register Select Signal; H: sending data, L: sending instructions	LCD pin 4
rw	1	output	LCD module Read/Write Select Signal; H: Read, L: Write	LCD pin 5
e	1	output	LCD module enable signal	LCD pin 6
lcd_data	8	bidir	Data bus to the LCD module / busy signal from the LCD	LCD pins 7-14

Initialization:

The LCD controller executes an initialization sequence each time it is powered-up or the reset_n pin is deasserted for a minimum of one clock cycle. The controller asserts the busy pin during initialization. Once initialization completes, the busy pin desserts, and the LCD controller waits in the Ready state for input from the user logic.

The initialization sequence specifies several LCD parameters: function, display control, display clear, and entry mode. The LCD controller instantiates the following default set of these options.

1 Function Set: 2-line mode, display on

2 Display Control: display on, cursor off, blink off

3 Entry Mode: increment mode, entire shift off

The user can send commands to the LCD to change any parameters after initialization. Alternatively, the user can edit the VHDL to change the default parameters. This simply requires commenting out the current VHDL line and uncommenting the line with the desired parameter setting. Table 2 lists the options available in the code.

Experimental Procedure:

1. Start the Xilinx Project Navigator by using the desktop shortcut or by using the
2. Start a Program a Xilinx ISE a Project Navigator
3. In the Project Navigator window go to FILE → New project → Give name → Next → Project settings → Next → Spartan 3 → Next → Finish.
4. Click on the symbol of FPGA device and then right Click on new source →
5. Select VHDL module → Give the name Shift_regd. → click on Next.
6. Generate the Behavioral VHDL Code for Shift_regd.
7. Check syntax, and remove errors if present.
8. Synthesize the design using XST.
9. Highlight Shift_regd.vhd file in the Sources in Project window. To run synthesis, right-click on Synthesize, and choose the Run option, or double-click on Synthesize in the Processes for Current Source window. Synthesis will run, and a green check will appear next to Synthesize when it is successfully completed. A yellow exclamation mark indicates that a warning was generated, and a red cross indicates an error was generated. Warnings are OK. If there are any errors, you can view the error through the console window otherwise continue on to the next step.

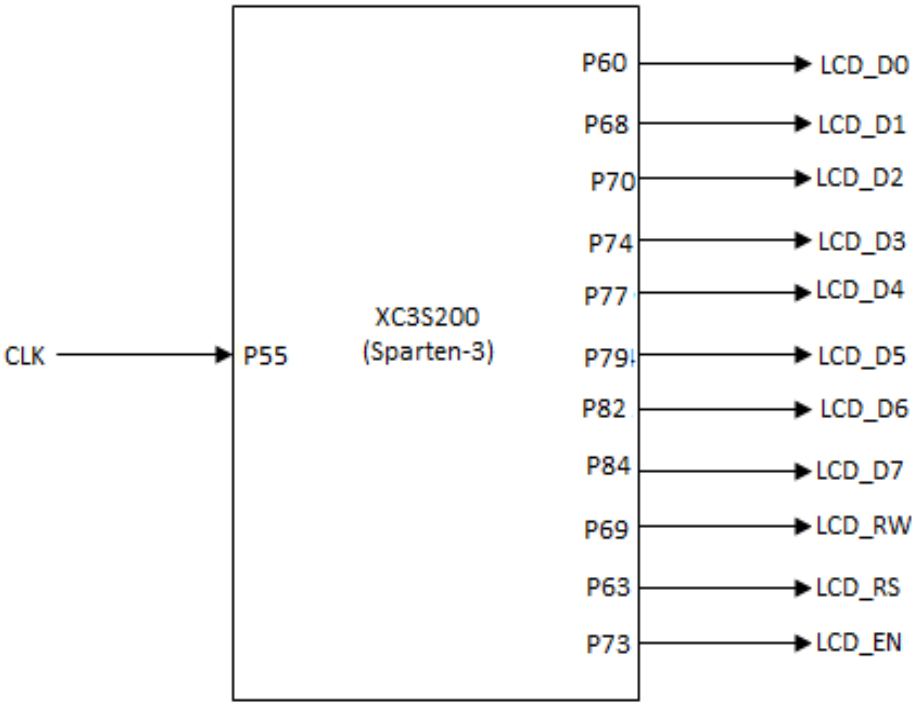
10. For creating Test Bench → Right click on Device → Select New Source→ Test Bench Waveform → Enter file name → Next →Finish.
11. Apply input as per requirement and save.
12. Double click on Behavioral Check Syntax. If process is complete successfully → Double click on simulate Behavioral model.
13. For Implementation Constraint file (UCF). Right click on Device→ Select New Source → Give File Name → Next → Source type Window appears.
14. After double click on I/O pin planning (Plan Ahead) Pre- Synthesis. → Click on Yes.
15. Assign the package pins using reference manual → save, the UCF file will be added in your project.
16. Click on Save and Exit the plan Ahead.
17. After successful implementation → Double click on 'Generating programming file'
18. For downloading .bit file working with USB JTAG cable.
19. Go to Flash Programmer Folder then double click on CDMV2.12.00 WHQL Certified.
20. Double Click on 'CDMV2.12.00 WHQL Certified' then FTDI CDM Drivers.
21. Click on Extract→ Next → accept agreement → Next → Finish.
22. Click on browse button and select respective .bit file of program and click on program button.
23. After downloading program successfully 'Green LED (LD1) will turned ON' on Board.
24. To download the new program Press 'RESET (RST)' button on Board.
25. Apply input through DIP Switches and observe the output displayed on LEDs.

Table 2. Initialization Options in the VHDL

Options	Choices	VHDL Line	Code
Function Set	2-line mode, display on*	93	lcd_data <= "00111100";
	1-line mode, display on	94	lcd_data <= "00110100";
	1-line mode, display off	95	lcd_data <= "00110000";

	2-line mode, display off	96	lcd_data <= "00111000";
Display ON/OFF	display on, cursor off, blink off*	104	lcd_data <= "00001100";
	display on, cursor off, blink on	105	lcd_data <= "00001101";
	display on, cursor on, blink off	106	lcd_data <= "00001110";
	display on, cursor on, blink on	107	lcd_data <= "00001111";
	display off, cursor off, blink off	108	lcd_data <= "00001000";
	display off, cursor off, blink on	109	lcd_data <= "00001001";
	display off, cursor on, blink off	110	lcd_data <= "00001010";
	display off, cursor on, blink on	111	lcd_data <= "00001011";
Entry Mode Set	increment mode, entire shift off*	127	lcd_data <= "00000110";
Entry Mode Set	increment mode, entire shift off*	127	lcd_data <= "00000110";
	increment mode, entire shift on	128	lcd_data <= "00000111";
	decrement mode, entire shift off	129	lcd_data <= "00000100";
	decrement mode, entire shift on	130	lcd_data <= "00000101";

Experimental set up:



Conclusion:

JSPM's
Imperial College of Engineering and Research, Wagholi, Pune.
(Approved by AICTE, Delhi & Govt. of Maharashtra, affiliated to Savitribai Phule Pune University)
Gat.No.720,Pune-Nagar road,Wagholi,Pune,412207
Phone No. 020-67335102 **website**: www.icoer.in **Email**- principal.imperial2016@gmail.com
Department of Electronics and Telecommunication Engineering

Experiment No. 5

Title: To study VHDL Code for Keyboard Interface for CPLD/FPGA Architecture.

Date of Performance: **Roll No:**

Date of Submission: **University Seat No:**

Signature of Staff:

JSPM's
Imperial College of Engineering and Research, Wagholi, Pune.
(Approved by AICTE, Delhi & Govt. of Maharashtra, affiliated to Savitribai Phule Pune University)
Gat.No.720,Pune-Nagar road,Wagholi,Pune,412207
Phone No. 020-67335102 **website**: www.icoer.in **Email**- principal.imperial2016@gmail.com
Department of Electronics and Telecommunication Engineering

Experiment No. 5

KEYBOARD INTERFACE

Title: To study VHDL Code for Keyboard Interface for CPLD/FPGA Architecture.

Aim:

To study VHDL Code for Keyboard Interface for CPLD/FPGA Architecture.

Equipments Required:

Software Required: Xilinx ISE
Hardware Required: PC, Spartan-3 Protoboard.

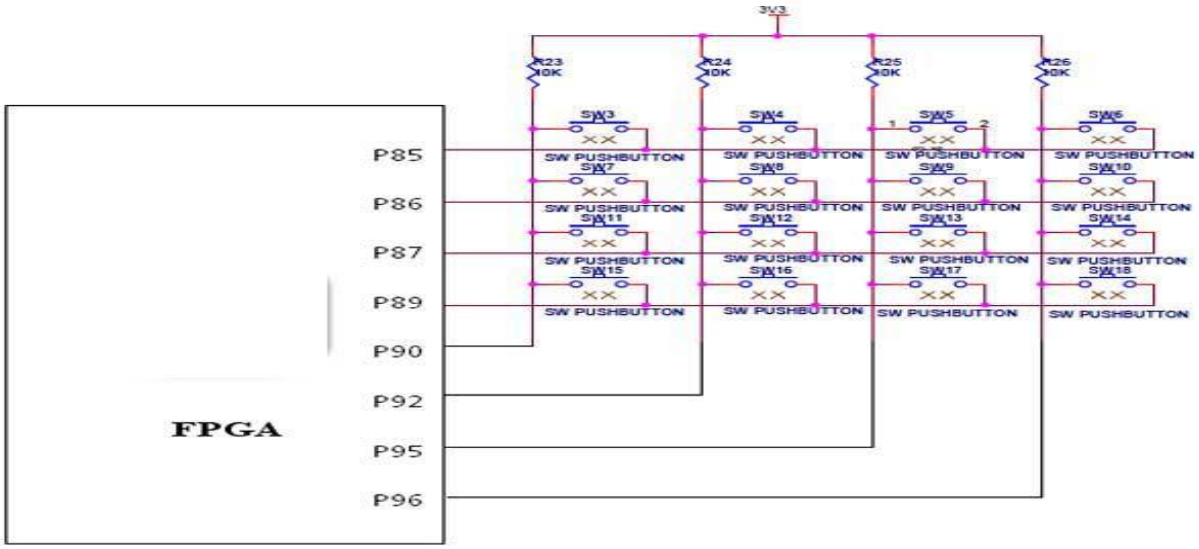
Theory:

Matrix keypad consists of a set of buttons similar to an alphanumeric keyboard provided with keys usually marked with letters or numbers and various extra keys. Embedded systems which require user interaction must be interfaced with devices that accept user input such as a keypad.

Interfacing Keypad with Spartan-3 Primer FPGA

The Spartan-3 Primer board has 4x4 Matrix Keypad, indicated as in Figure 1. Keypads arranged by matrix format, each row and column section pulled by high, all row lines and column lines connected directly by the I/O pins.

Circuit Diagram to Interface PB with Spartan-3 Primer FPGA



Experimental Procedure:

1. Start the Xilinx Project Navigator by using the desktop shortcut or by using the Start Programs a Xilinx ISE a Project Navigator
2. In the Project Navigator window go to FILE a New project a Select Device. Click on the symbol of FPGA device and then right Click on new source Select VHDL module and give the name LCD and click on Next.
3. Generate the Behavioral VHDL Code for LCD
4. Check syntax, and remove errors if present.
5. Synthesize the design using XST.
6. Highlight KEYPAD file in the Sources in Project window. To run synthesis, right-click on Synthesize, and choose the Run option, or double-click on Synthesize in the Processes for Current Source window. Synthesis will run, and a green check will appear next to Synthesize when it is successfully completed. A yellow exclamation mark indicates that a warning was generated, and a red cross indicates an error was generated. Warnings are OK. If there are any errors, you can view the error through the console window otherwise continue on to the next step.
7. Simulate the design using Model-sim. Highlight KEYPAD file in the Sources in Project window. To run the Functional Simulation, Click on the symbol of FPGA device and then right Click on new source. Click on test bench waveform Give file name Select entity Finish Give inputs Click on simulate behavioral model see the output.

8. Write User Constraint file wherein the FPGA pins are locked as per the Spartan-3 hardware.

9. Run the Xilinx Implementation Tools.

10. Once synthesis is complete, you can place and route your design to fit into a Xilinx device (Spartan-3 400k), and you can also get some post place-and-route timing information about the design. This procedure runs you through the basic flow for implementation. Right-click on Implement Design, and choose the Run option, or double left-click on Implement Design.

11. Right-click on Generate Programming File, and choose the Run option, Or double left-click on Generate Programming File. This will generate the Bitstream.

12. Double click on Configure Device to download the bit stream.

13. Apply input through DIP Switches output is displayed on OUTPUT LEDs.

Conclusion:

JSPM's
Imperial College of Engineering and Research, Wagholi, Pune.
(Approved by AICTE, Delhi & Govt. of Maharashtra, affiliated to Savitribai Phule Pune University)
Gat.No.720,Pune-Nagar road,Wagholi,Pune,412207
Phone No. 020-67335102 **website**: www.icoer.in **Email**- principal.imperial2016@gmail.com
Department of Electronics and Telecommunication Engineering

Experiment No. 6

Title: To prepare CMOS layout in selected technology, simulate with and without capacitive load, comment on rise, and fall times. Inverter, NAND and NOR gates

Date of Performance: **Roll No:**

Date of Submission: **University Seat No:**

Signature of Staff:

JSPM's
Imperial College of Engineering and Research, Wagholi, Pune.
(Approved by AICTE, Delhi & Govt. of Maharashtra, affiliated to Savitribai Phule Pune University)
Gat.No.720,Pune-Nagar road,Wagholi,Pune,412207
Phone No. 020-67335102 **website**: www.icoer.in **Email**- principal.imperial2016@gmail.com
Department of Electronics and Telecommunication Engineering

Experiment No. 6

Inverter, NAND and NOR gates

Title: To prepare CMOS layout in selected technology, simulate with and without capacitive load, comment on rise, and fall times. Inverter, NAND,and NOR gates.

Aim:

1. To Study CMOS layout for Inverter.
2. To Study CMOS layout for NAND.
3. To Study CMOS layout for NOR.

Software Required: Microwind and DSCH.

Theory:

Microwind

Microwind is a computer-aided design program for layout diting,circuit extraction and circuit simulation of micro electronic circuits on a physical layout level.

The implemented version of Microwind (2.6k) program runs on Windows XP (Note:This is not the most recent release of Microwind, but has worked we ll for our education).

The development of Microwind was initiated by professor Etienne Sicard in France at l'Institut National des Sciences Appliquées de Toulouse (INSA)

The program has been used to a large extent worldwide for introductory courses in CMOS design both for engineers in industry as well as university students in many countries.

The layout editor contains a large number of commands for integrated circuit layout editing, for example copy, cut, past, duplicate, stretch, and move etc. commands which will be used in later laborations and for your project work.

The IC layout can be shown in different so called views.

You can for example plot the transfer characteristic of MOS transistors and view 2-dimensional cuts through the geometry of semiconductor devices you made in a layout.

The layout program contains a built-in circuit simulator similar to the well-known Spice circuit simulator. Therefore, to make a circuit simulation in Micro-Wind, you need no "external" simulator. With the built-in circuit simulator you will be able to simulate a circuit

model of the designed layout, after you have done a circuit extraction from your layout that will generate a PSpice netlist from the physical layout.

The built-in simulator contains a model library for circuit simulation with a number of different geometrical and physical model aspects (classified by a LEVEL number) for the descriptions for MOS transistors.

The circuit simulator calculates voltages and currents as a function of time using "real" circuit delays calculated based on layout geometry and component data extracted from the geometric description and the model for the physical layout. The power consumption for a given circuit with given input signal stimuli can be calculated. The circuit simulator can manage a wide spectrum of technologies from 1.2 sμm down to 70 nm

Introduction:-

CMOS configuration is called complementary MOS. The circuit topology is complementary push-pull in the sense that for high input, the NMOS transistor drives (pull -down), the output node, while the PMOS transistor act as the load, and for the low input the PMOS transistor drives (pull -up), output node, while NMOS acts as load. Consequently both devices contribute equally to the circuit operation characteristics.

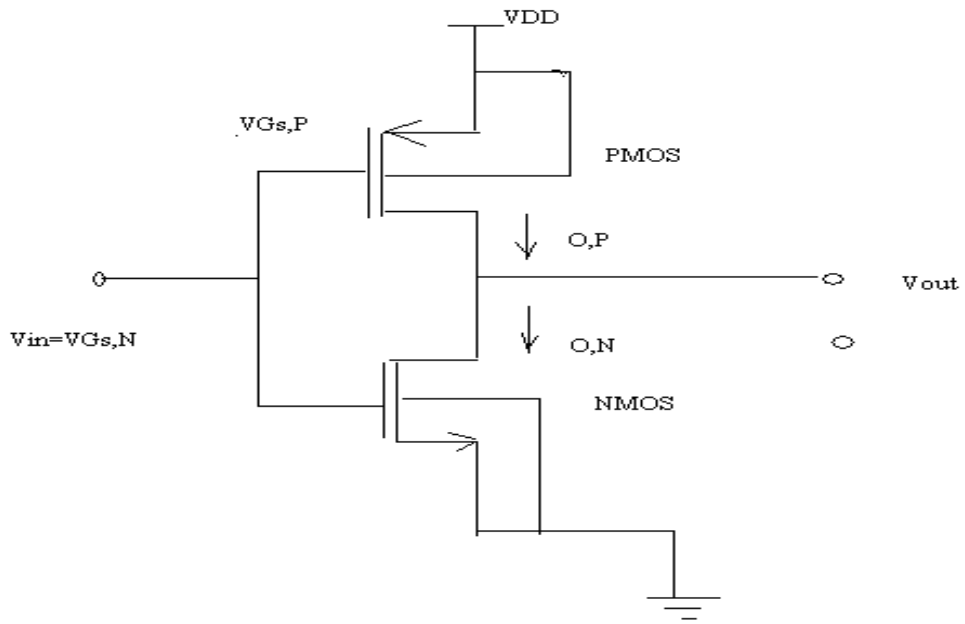


FIG. 1. CMOS INVERTER CIRCUIT

The CMOS inverter has 2 input advantages over the other inverter configuration. The trend of increasing sub threshold leakage current in deep sub-micron technologies causes great design challenges. in all other inverter structures examined so far ,a non-zero steady state current is drown from the power source when the driver transistor is turned-on , which result in a significant dc power consumption . The other advantages of CMOS configuration are that the voltage swing between 0v and VDD and that the Vth is usually very sharp

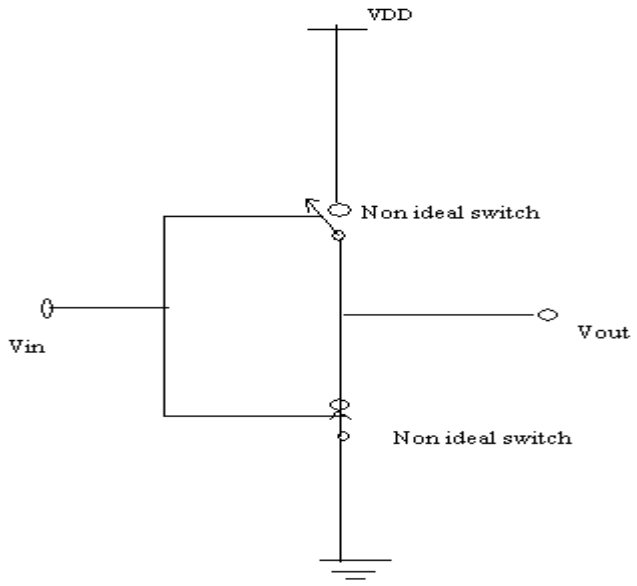


FIG 2.SIMPLIFIED VIEW OF CMOS

Table no 1: Operating regions of CMOS Inverter

REGION	Vin	Vout	NMOS	PMOS
A	< VTO,n	VoH	Cut-off	linear
B	ViL	High~VoH	saturation	linear
C	VtH	VtH	saturation	saturation
D	ViH	Low~VoL	linear	saturation
E	>(VDD+VTo,p)	VoL	linear	Cut-off

NAND and NOR Theory: A two-input NAND gate (F = A·B). The PDN network consists of two NMOS devices in series that conduct when both A and B are high. The PUN is the dual network, and consists of two parallel PMOS transistors. This means that F is 1 if A = 0 or B = 0, which is equivalent to F = A·B. The truth table for the simple two input NAND gate is given

in Table 1. It can be verified that the output F is always connected to either VDD or GND, but never to both at the same time.

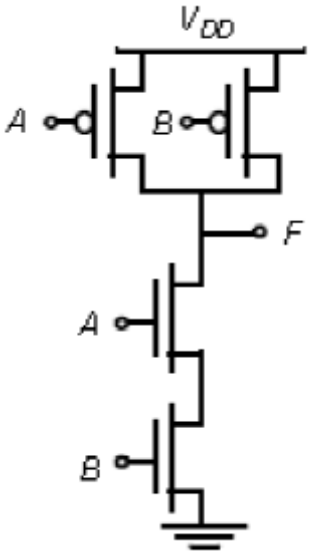


Fig 1: Two-input NAND gate in complementary static CMOS style.

A	B	OUTPUT
0	0	1
0	1	1
1	0	1
1	1	0

Table 1: Truth Table for 2 input NAND

The first step in the synthesis of the logic gate is to derive the pull-down network. The fact is that NMOS devices in series implements the AND function and parallel device implements the OR function. The next step is to use duality to derive the PUN in a hierarchical fashion. The PDN network is broken into smaller networks (i.e., subset of the PDN) called sub-nets that simplify the derivation of the PUN.

Design Consideration: The important point to take away from is that the noise margins are input pattern dependent. A smaller input glitch will cause a transition at the output if only one of the inputs makes a transition. Therefore, this condition has a lower low noise margin. A common practice when characterizing gates such as NAND and NOR is to connect all the

inputs together. This unfortunately does not represent the worst-case static behavior. The data dependencies should be carefully modeled.

A	B	OUTPUT
0	0	1
0	1	0
1	0	0
1	1	0

Table 2: Truth Table for 2 input NOR

The output of this NOR network is high, if and only if both inputs A and B are low. The worst-case pull-down transition happens when only one of the NMOS devices turns on (i.e., if either A or B is high). Since the pull-down path in the worst case is a single device, the NMOS devices (M1 and M2) can have the same device widths as the NMOS device in the inverter. For the output to be pulled high, both devices must be turned on. Since the resistances add, the devices must be made two times larger compared to the PMOS in the inverter. Since PMOS devices have a lower mobility relative to NMOS devices, stacking devices in series must be avoided as much as possible. A NAND implementation is clearly preferred over a NOR implementation for implementing generic logic

Procedure:

1. Select Foundry
2. Calculate W/L ratio for CMOS Inverter,NAND,NOR gates, Half adder.
3. Select NMOS from palette with calculated width and length.
4. Select NMOS from palette with calculated width and length.
5. Join the gates of PMOS & NMOS, label the inputs.
6. Connect sources of PMOS to VDD rail and source of NMOS to GND rail with metal.
7. Make the connections as per the layout with metal.
8. Run DRC & connect the errors if any.

9. Run Simulation and verify the output waveforms.

Conclusion:

JSPM's
Imperial College of Engineering and Research, Wagholi, Pune.
(Approved by AICTE, Delhi & Govt. of Maharashtra, affiliated to Savitribai Phule Pune University)
Gat.No.720,Pune-Nagar road,Wagholi,Pune,412207
Phone No. 020-67335102 **website**: www.icoer.in **Email**- principal.imperial2016@gmail.com
Department of Electronics and Telecommunication Engineering

Experiment No. 7

Title: To prepare CMOS layout in selected technology, simulate with and without capacitive load, comment on rise, and fall times. 2:1 multiplexer using logic gates and transmission gates.

Date of Performance: **Roll No:**

Date of Submission: **University Seat No:**

Signature of Staff:

JSPM's
Imperial College of Engineering and Research, Wagholi, Pune.
(Approved by AICTE, Delhi & Govt. of Maharashtra, affiliated to Savitribai Phule Pune University)
Gat.No.720,Pune-Nagar road,Wagholi,Pune,412207
Phone No. 020-67335102 **website**: www.icoer.in **Email**- principal.imperial2016@gmail.com
Department of Electronics and Telecommunication Engineering

Experiment No. 7

2:1 MULTIPLEXER

Title: To prepare CMOS layout in selected technology, simulate with and without capacitive load, comment on rise, and fall times. 2:1-multiplexer using logic gates and transmission gates.

Aim:

To prepare CMOS layout for 2:1 MUX using logic gate and transmission gate and compare the performance in terms of speed, space and power

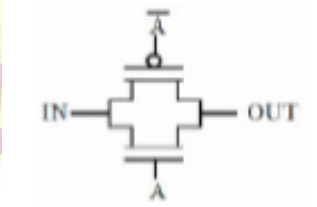
Software Required:

Microwind and DSCH.

Theory:

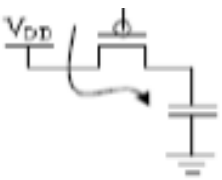
CMOS Transmission Gates:

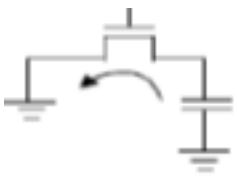
- A transmission gate is a complimentary CMOS switch.

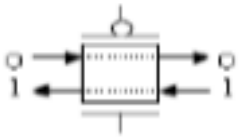


PMOS and NMOS are in parallel and controlled by a complimentary signal.

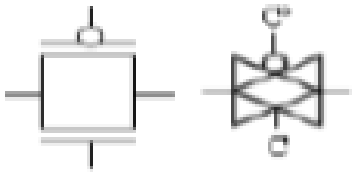
- Both transistors are ON or OFF simultaneously .
- The NMOS passes good '0' but poor '1'.
- The PMOS passes good '1' but poor '0'.
- Combining both we get a good '0' and good '1' in both directions.







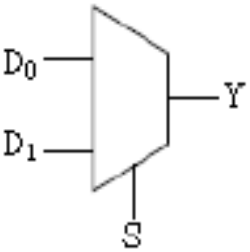
- Circuit symbols for TGs

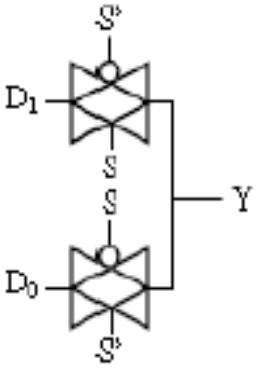


- **2-1 mux using TGs**

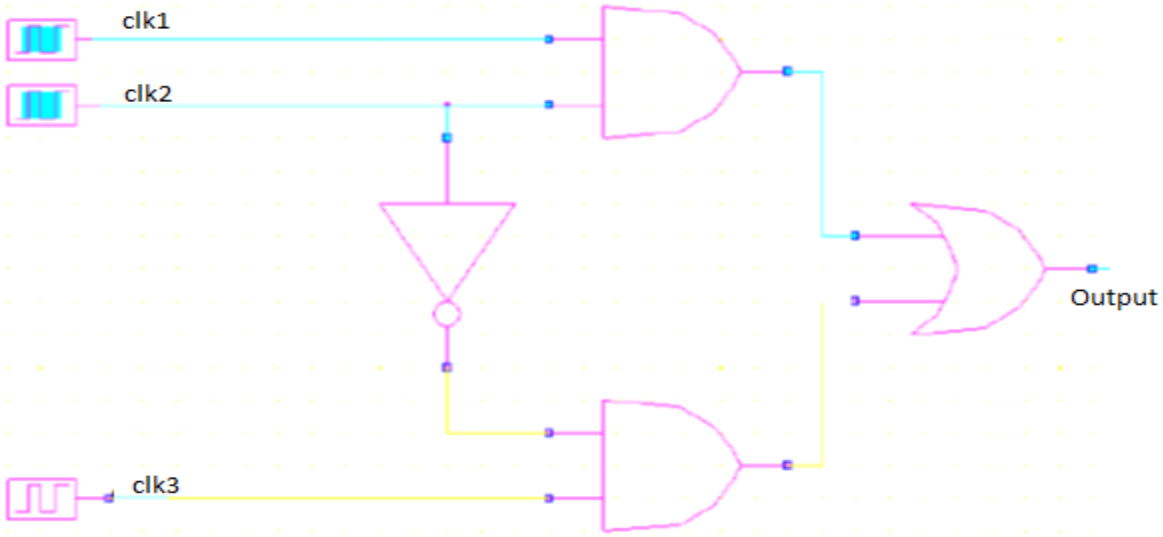
$F = S'.D_0 + S.D_1$

S	Y
0	D_0
1	D_1





2:1-MUX using conventional Logic Gates



Procedure:

1. Select Foundry
2. Calculate W/L ratio for CMOS 2:1 Mux.
3. Select NMOS from palette with calculated width and length.
4. Select NMOS from palette with calculated width and length.
5. Join the gates of PMOS & NMOS, label the inputs.
6. Connect sources of PMOS to VDD rail and source of NMOS to GND rail with metal.
7. Make the connections as per the layout with metal.
8. Run DRC & connect the errors if any.
9. Run Simulation and verify the output waveforms.

Conclusion:

JSPM's
Imperial College of Engineering and Research, Wagholi, Pune.
(Approved by AICTE, Delhi & Govt. of Maharashtra, affiliated to Savitribai Phule Pune University)
Gat.No.720,Pune-Nagar road,Wagholi,Pune,412207
Phone No. 020-67335102 **website**: www.icoer.in **Email**- principal.imperial2016@gmail.com
Department of Electronics and Telecommunication Engineering

Experiment No. 8

Title: To prepare CMOS layout in selected technology, simulate with and without capacitive load, comment on rise, and fall times. Single bit SRAM cell.

Date of Performance: **Roll No:**

Date of Submission: **University Seat No:**

Signature of Staff:

Experiment No. 8

SINGLE BIT SRAM CELL

Title: To prepare CMOS layout in selected technology, simulate with and without capacitive load, comment on rise, and fall times. Single bit SRAM cell.

Aim:

Design a layout for Single Bit SRAM Cell using CMOS Technology

Software Required:

Microwind and DSCH.

Theory:

Memory design

Memories are usually constructed as two dimensional arrays of bits. Thus a memory containing 2w words each of 2b bits will be configured as 2w rows by 2b columns. w address bits will be decoded to give the row and either the whole word will be output or multiplexor used to select asingle bit using a further b address bits.

Design a layout for Single Bit SRAM Cell using CMOS Technology

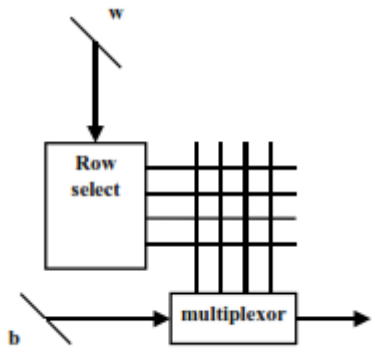


Fig 1:Basic memory storage operation

Read-only memory (ROM)

A read-only memory (ROM) is like a PLA with all the possible minterms being calculated. The individual memory cells can be very compact.

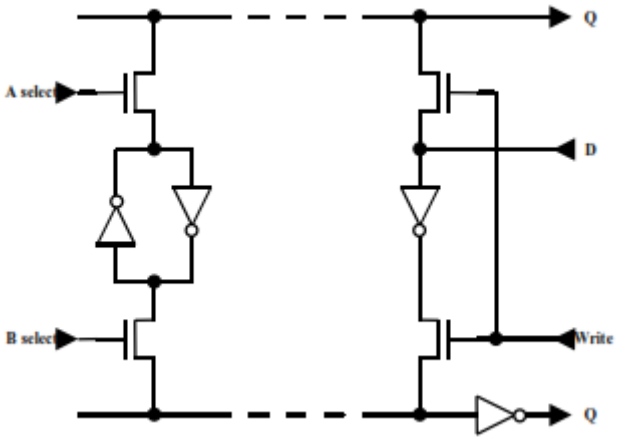
Programmable read-only memories (PROMs)

Programmable read-only memories (PROMs) allow the diffusion tabs to be switched in Electrically Erasable PROMs allow this switching to be reversed, either by exposure to ultraviolet light (EPROMs) or under digital control (electrically erasable PROMs or EEPROMs).

Static read/write memory

The simplest form of writeable memory (RAM) is static memory. A bit is stored in a pair of Cross-coupled invertors, with separate circuits to control the reading and writing of the data.

The memory has two independent ports for reading; both selection lines are opened for writing. Six transistors are required to store each bit, plus some overheard for the control circuitry.



Dynamic RAM

Fewer transistors are needed if the bit is stored as charge on the gate of a FET. The three-transistor memory cell operates as follows:

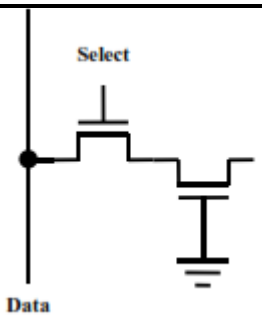
- Write by putting data on **Data** and strobing **Write**
- Read by pre-charging **Data** and strobing **Read;**

the value obtained has to be inverted.

Refresh by reading and re-writing at least every millisecond or so.

Less circuitry is required for each individual bit at the expense of more sophisticated control circuits.

This is taken to an extreme with a one-transistor memory cell:



The bit is stored as charge under the grounded gate of a second transistor. Again, refreshing is required and reading requires the use of subtle analogue sense amplifiers.

.

Procedure:

1. Select Foundry
2. Calculate W/L ratio for CMOS SRAM.
3. Select NMOS from palette with calculated width and length.
4. Select NMOS from palette with calculated width and length.
5. Join the gates of PMOS & NMOS, label the inputs.
6. Connect sources of PMOS to VDD rail and source of NMOS to GND rail with metal.
7. Make the connections as per the layout with metal.
8. Run DRC & connect the errors if any.
9. Run Simulation and verify the output waveforms.

Conclusion: