**Lab Experiment-8**

**Title of the Experiment:** To design 4-bit Shift Registers (SISO, SIPO, PISO, PIPO,USR) using Verilog code and compare with their respective truth tables.

**Objective/Motivation:** In this lab, 4-bit Shift Registers (SISO, SIPO, PISO, PIPO,USR) are designed. The objective will be to test these designs on Xilinx simulation tool. The tests will be performed for all the possible combinations of inputs to verify their functionality. Moreover, the knowledge gained will be used to design much larger and complex logic designs.

**Equipment required:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S No** | **Name of The Components/Tool** | **Version** | **Quantity** |
| 1. | Xilinx Vivado Design Suite **/** EDA play Ground software | V23.1 | 1 |
| 2. | Zybo board | Zynq XC7Z010 | 1 |
| 3. | Personal Computer | - | 1 |

**Logic diagram(s) and Truth tables:**

|  |  |
| --- | --- |
| **Logic CKT name** | **Logic Diagram** |
| **SISO** | Shift RegisterShift RegisterShift RegisterShift Register |
| **SIPO** | Shift Register |
| **PIPO** | Shift Register |

|  |  |
| --- | --- |
| **Logic CKT name** | **Logic Diagram** |
| **PISO** | Shift Register |
| Universal Shift Register | Shift Register |

**Operation Procedure:**

* A group of flip flops which is used to store multiple bits of data and the data is moved from one flip flop to another is known as **Shift Register**.
* A **Shift Register** can shift the bits either to the left or to the right. A **Shift Register**, which shifts the bit to the left, is known as **"Shift left register"**, and it shifts the bit to the right, known as **"Right left register"**.
* The shift register is classified into the following types:
* Serial In Serial Out
* Serial In Parallel Out
* Parallel In Serial Out
* Parallel In Parallel Out
* Universal Shift Register

## Serial IN Serial OUT

* In "Serial Input Serial Output", the data is shifted "IN" or "OUT" serially. In SISO, a single bit is shifted at a time in either right or left direction under clock control.
* initially, all the flip-flops are set in "reset" condition i.e. Y3 = Y2 = Y1 = Y0 = 0. If we pass the binary number 1111, the LSB bit of the number is applied first to the Din bit. The D3 input of the third flip flop, i.e., FF-3, is directly connected to the serial data input D3. The output Y3 is passed to the data input d2 of the next flip flop. This process remains the same for the remaining flip flops. The block diagram of the **"Serial IN Serial OUT"** is given below.

## Serial IN Parallel OUT: In the ****"Serial IN Parallel OUT"**** shift register, the data is passed serially to the flip flop, and outputs are fetched in a parallel way. The data is passed bit by bit in the register, and the output remains disabled until the data is not passed to the data input. When the data is passed to the register, the outputs are enabled, and the flip flops contain their return value

## Parallel IN Serial OUT: In the ****"Parallel IN Serial OUT"**** register, the data is entered in a parallel way, and the outcome comes serially. A four-bit ****"Parallel IN Serial OUT"**** register is designed below. The input of the flip flop is the output of the previous Flip Flop. The input and outputs are connected through the combinational circuit. Through this combinational circuit, the binary input B0, B1, B2, B3 are passed. The ****shift mode**** and the ****load mode**** are the two modes in which the ****"PISO"**** circuit works.

## Parallel IN Parallel OUT: In ****"Parallel IN Parallel OUT"****, the inputs and the outputs come in a parallel way in the register. The inputs A0, A1, A2, and A3, are directly passed to the data inputs D0, D1, D2, and D3 of the respective flip flop. The bits of the binary input is loaded to the flip flops when the negative clock edge is applied. The clock pulse is required for loading all the bits. At the output side, the loaded bits appear.

## Universal Shift Register

* A register where the data is shifted in one direction is known as the **"uni-directional"** shift register. A register in which the data is shifted in both the direction is known as **"bi-directional"** shift register. A **"Universal"** shift register is a special type of register that can load the data in a parallel way and shift that data in both directions, i.e., right and left.
* The input M, i.e., the mode control input, is set to 1 to perform the parallel loading operation. If this input set to 0, then the serial shifting operation is performed. If we connect the mode control input with the ground, then the circuit will work as a **"bi-directional"** register. The diagram of the universal shift register is given below. When the input is passed to the **serial input**, the register performs the "serial left" operation. When the input is passed to the input **D**, the register performs the serial right operation.

**Verilog Source Code for All gates in various Models or various Level of Abstrction:**

|  |  |  |
| --- | --- | --- |
| **Gate name** | **Source Code** | **Test Bench** |
| **SISO** | // SISO Shift Register-Source Code  module siso\_d(clk,rst,d,v);  input clk,rst,d;  output v;  reg [3:0]q;  always @(posedge clk)  begin  if (rst==1)  q<=4'b0000;  else  q<=q>>1;  q[3]<=d;  end  assign v=q[0];  endmodule | // // SISO Shift Register-Test bench Code  module siso\_tb();  reg clk,rst,d;  wire v;  siso\_d dut(.clk(clk),.rst(rst),.d(d),.v(v));  initial begin  clk=0;  rst=1;  d=0;  #10;  rst=0;  end  always #1 clk=~clk;  always #2 d=~d;  initial $monitor("clk=%b,d=%b,v=%b",clk,d,v);  initial #100 $stop();  endmodule |

|  |  |  |
| --- | --- | --- |
| **Gate name** | **Source Code** | **Test Bench** |
| **SIPO** | // SIPO Shift Register-Source Code  module sipo(input rst,clk,d,output reg[3:0]q);  reg [3:0]temp;  always @(posedge clk)  if (rst==1)  q=4'b0000;  else  begin  temp=q>>1;  q={d,temp[2:0]};  end  endmodule | // SIPO Shift Register-Test bench Code  module sipo\_tb();  reg rst,clk,d;  wire [3:0]q;  sipo dut(.rst(rst),.clk(clk),.d(d),.q(q));  initial begin  rst=1;  clk=0;  d=0;  #10  rst=1'b0;  end  always #1 clk=~clk;  always #2 d=~d;  initial $monitor("d=%b,q=%b",d,q);  initial #50 $stop();  endmodule |
| **PISO** | // PISO Shift Register-Source Code  module piso(v,clk,sel,rst,d);  input clk,sel,rst;  input [3:0]d;  output reg v;  reg [3:0]q;  always @(posedge clk)  if (rst==1)  q=4'b0000;  else if (sel==0)  q<=d;  else  begin  v<=q[0];  q<=q>>1;  end  endmodule | // PISO Shift Register-Test bench Code  module piso\_tb();  reg clk,sel,rst;  reg [3:0]d;  wire v;  piso dut(.clk(clk),.sel(sel),.rst(rst),.d(d),.v(v));  initial begin  clk=0;  rst=1;  #2 rst=0;  sel=0;  d=4'b0100;  #2 sel=1;  #10 sel=0;  d=4'b1110;  #2 sel=1;  end  always #1 clk=~clk;  initial $monitor("clk=%b,d=%b,v=%b",clk,d,v);  initial #25 $stop();  endmodule |
| **PIPO** | // PIPO Shift Register-Source Code  module pipo(q,clk,rst,d);  input clk,rst;  input [3:0]d;  output reg [3:0]q;  always @(posedge clk)  if (rst==1)  q<=4'b0000;  else  q=d;  endmodule | // PIPO Shift Register-Test bench Code  module pipo\_tb();  reg clk,rst;  reg [3:0]d;  wire [3:0]q;  pipo dut(.clk(clk),.rst(rst),.d(d),.q(q));  initial begin  clk=0;  rst=1;  d=4'b1010;  #5 rst=0;  #2 d=4'b0000;  #2 d=4'b0011;  #2 d=4'b1011;  end  always #1 clk=~clk;  initial $monitor("clk=%b,d=%b,q=%b",clk,d,q);  initial #14 $stop();  endmodule |

|  |  |  |
| --- | --- | --- |
| **Gate name** | **Source Code** | **Test Bench** |
| **USR** | // Universal Shift Register-Source Code  module usr(sl,sr,din,clk,rst,q);  input sl, sr, din, clk,rst;  output [7:0]q;  reg [7:0]q;  always @ (posedge clk)  begin  if (~rst)  begin  if (sl)  begin  q<=#2{q[6:0],din};  end  else if (sr)  begin  q<=#2{din,q[7:1]};  end  end  end  always @ (posedge rst)  begin  q<=8'b00000000;  end  endmodule | // // Universal Shift Register-Test bench Code  odule usr\_tb();  reg clk,rst,din,sl,sr;  wire [7:0]q;  usr uut(sl,sr,din,clk,rst,q);  initial begin  forever begin  clk <= 0;  #5  clk <= 1;  #5  clk <= 0;  end  end  initial begin  rst = 1;  #12  rst = 0;  #90  rst=1;  #12  rst=0;  end  initial begin  sl = 1;  sr = 0;  #50  sl = 0;  #12  sr = 1;  end  initial begin  forever begin  din = 0;  #7  din = 1;  #8  din = 0;  end  end  endmodule |

**Result:** The simulation waveforms are obtained with various shift process and verified.