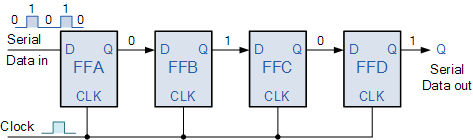
**LAB EXPERIMENT 9**

**Aim:** To design all the four types of shift registers in Xilinx software using Verilog Language of Programming and viewing our output in the simulator.

**Theory:** There are four types of shift registers:

1. SISO: The data is shifted serially “IN” and “OUT” of the register, one bit at a time in either a left or right direction under clock control. The SISO shift register is one of the simplest of the four configurations as it has only three connections, the serial input (SI) which determines what enters the left hand flip-flop, the serial output (SO) which is taken from the output of the right hand flip-flop and the sequencing clock signal (Clk).

Circuit Diagram:



1. SIPO: The shift register, which allows serial input (one bit after the other through a single data line) and produces a parallel output is known as Serial-In Parallel-Out shift register.The logic circuit given below shows a serial-in-parallel-out shift register. The circuit consists of four D flip-flops which are connected. The clear (CLR) signal is connected in addition to the clock signal to all the 4 flip flops in order to RESET them. The output of the first flip flop is connected to the input of the next flip flop and so on. All these flip-flops are synchronous with each other since the same clock signal is applied to each flip flop.

Circuit Diagram:

Diagram

Description automatically generated

1. PISO: The shift register, which allows parallel input (data is given separately to each flip flop and in a simultaneous manner) and produces a serial output is known as Parallel-In Serial-Out shift register.The logic circuit given below shows a parallel-in-serial-out shift register. The circuit consists of four D flip-flops which are connected. The clock input is directly connected to all the flip flops but the input data is connected individually to each flip flop through a multiplexer at the input of every flip flop. The output of the previous flip flop and parallel data input are connected to the input of the MUX and the output of MUX is connected to the next flip flop. All these flip-flops are synchronous with each other since the same clock signal is applied to each flip flop.

Circuit Diagram:

A picture containing text, whiteboard

Description automatically generated

1. PIPO: The shift register, which allows parallel input (data is given separately to each flip flop and in a simultaneous manner) and also produces a parallel output is known as Parallel-In parallel-Out shift register. The logic circuit given below shows a parallel-in-parallel-out shift register. The circuit consists of four D flip-flops which are connected. The clear (CLR) signal and clock signals are connected to all the 4 flip flops. In this type of register, there are no interconnections between the individual flip-flops since no serial shifting of the data is required. Data is given as input separately for each flip flop and in the same way, output also collected individually from each flip flop.

Circuit Diagram:

A picture containing diagram

Description automatically generated

**Verilog Code of the Program and Outputs:**

1. **SISO**
2. **Verilog Code of the Program:**

module SISO\_062(clk,reset,si,so);

input clk,si,reset;

output so;

reg so;

reg [3:0] tmp;

always @(posedge clk )

begin

if (reset)

tmp <= 4'b0000;

else

tmp <= tmp << 1;

tmp[0] <= si;

so = tmp[3];

end

endmodule

1. **Screenshots of the Program and Outputs:**

Graphical user interface, text, application

Description automatically generated

A picture containing text, screenshot, indoor

Description automatically generated

1. **RTL Schematics:**

A picture containing text, indoor, screenshot, electronics

Description automatically generated

A picture containing text, indoor, screenshot, electronics

Description automatically generated

1. **SIPO:**
2. **Verilog Code of the Program:**

module SIPO\_062(clk, reset, si, po);

input clk,si,reset;

output [3:0] po;

reg [3:0] tmp;

reg [3:0] po;

always @(posedge clk)

begin

if (reset)

tmp <= 4'b0000;

else

tmp <= tmp << 1;

tmp[0] <= si;

po = tmp;

end

endmodule

1. **Screenshots of the Program and Outputs:**

Graphical user interface, text, application

Description automatically generated

A picture containing text, screenshot, indoor

Description automatically generated

1. **RTL Schematics:**

A picture containing text, indoor, screenshot, display

Description automatically generated

A picture containing text, indoor, electronics, display

Description automatically generated

1. **PISO:**
2. **Verilog Code of the Program:**

module PISO(Clk,Parallel\_In,rst,Serial\_Out);

input Clk,rst;

input[3:0]Parallel\_In;

output reg Serial\_Out;

reg[3:0]tmp;

always@(posedge Clk,posedge rst)

begin

if(rst==1'b1)

begin

Serial\_Out<=1'b0;

tmp<=Parallel\_In;

end

else

begin

Serial\_Out<=tmp[0];

tmp<=tmp>>1'b1;

end

end

endmodule

1. **Screenshots of the Program and Outputs:**

Graphical user interface, text, application

Description automatically generated

1. **RTL Schematics:**

A picture containing text, indoor, monitor, screenshot

Description automatically generated

A picture containing text, indoor

Description automatically generated

1. **PIPO:**
2. **Verilog Code of the Program:**

module PIPO\_062(clk,rst,Parallel\_In,Parallel\_Out);

input clk,rst;

input[3:0]Parallel\_In;

output[3:0]Parallel\_Out;

reg[3:0]Parallel\_Out;

always@(posedge clk,posedge rst)

begin

if (rst==1'b1)

Parallel\_Out<=4'b0000;

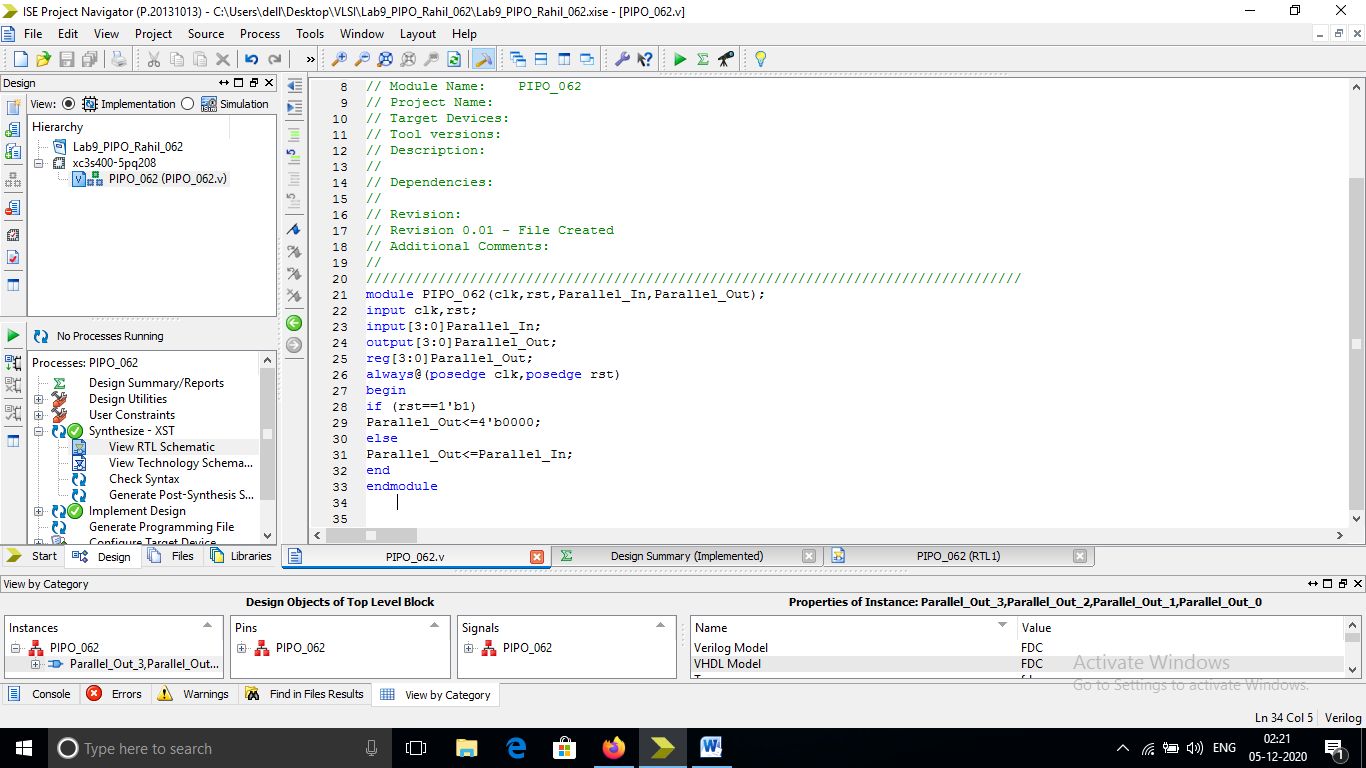
else

Parallel\_Out<=Parallel\_In;

end

endmodule

1. **Screenshots of the Program and Outputs:**



1. **RTL Schematics:**

A picture containing text, indoor, screenshot

Description automatically generated

Graphical user interface

Description automatically generated

**Conclusion: From this experiment we learnt how to design the shift registers in Xilinx Software.**