S. B. JAIN INSTITUTE OF TECHNOLOGY,MANAGEMENT & RESEARCH,NAGPUR.

Practical No. 04

Aim: Write behavioural VHDL program for 4-bit Shift register.

Name of Student	**: Aadesh R. Motghare**
Roll No.	**: 41(ET20065)**
Semester/Year	**: 6th Sem/3rd Year**
Academic Session	**: 2022-23**
Date of Performance	**:**
Date of Submission	**:**

AIM: Write behavioural VHDL program for 4-bit Shift register.

OBJECTIVE:

- To verify the functionality of shift register.
- To design a digital system using shift register.

SOFTWARE: - Xilinx ISE14.7.

THEORY:-

SHIFT REGISTER

A flip flop is capable of storing one bit of information. A register is a group of flip flops. So an n-bit register consists of a group of n flip flops. This n bit register can store n-bit information. The information in these registers can be transferred by connecting the flip flops. These connected flip flops are known as shift registers and the bits stored in such registers can be made to move within the registers and in/out of the registers by applying clock pulses.

Shift registers are basically of 4 types. These are:

1. Serial In Serial Out shift register
2. Serial In parallel Out shift register
3. Parallel In Serial Out shift register
4. Parallel In parallel Out shift register

Serial In Serial Out shift register (SISO)

The shift register, which allows serial input and produces a serial output is known as Serial-In Serial-Out shift register.
So SISO shift register has one data input line and one data output line.
The logic circuit below shows a 4 bit SISO shift register. It consists of 4 D FFs which are serially connected and driven by a common clock(clk) signal. The reset (rst) signal is connected in addition to the clock signal to all the 4 flip flops in order to RESET them.Serial Input is given through the din and serial output taken through dout.

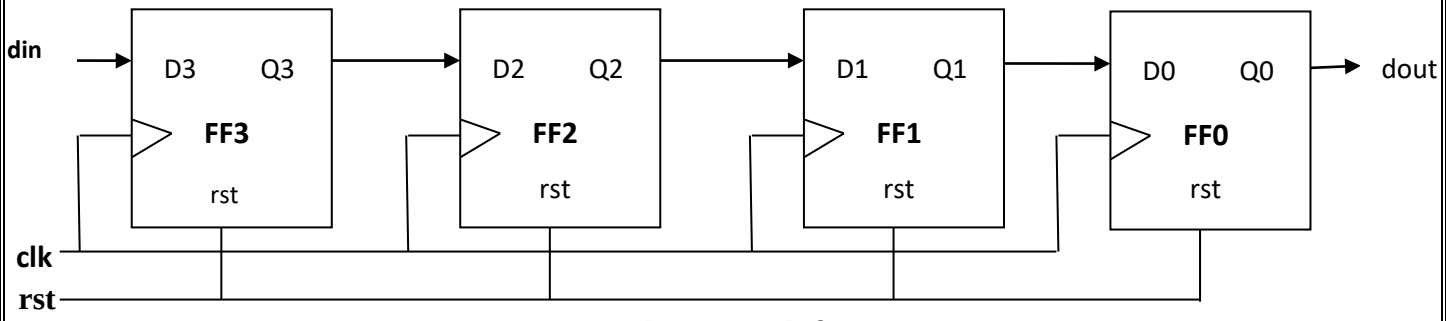


Fig 1. 4 bit SISO shift register

Serial-In Parallel-Out Shift Register (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 below shows a 4 bit SIPO shift register. It consists of 4 DFFs connected as shown in fig2. ,all flip flops are driven by same clock (clk) signal. The reset (rst) signal is connected in addition to the clock signal to all the 4 flip flops in order to RESET them.

Serial input is given through din and Q3,Q2, Q1,Q0 are parallel data output lines.

So SIPO shift register has one data input line and 4 (n)data output lines.

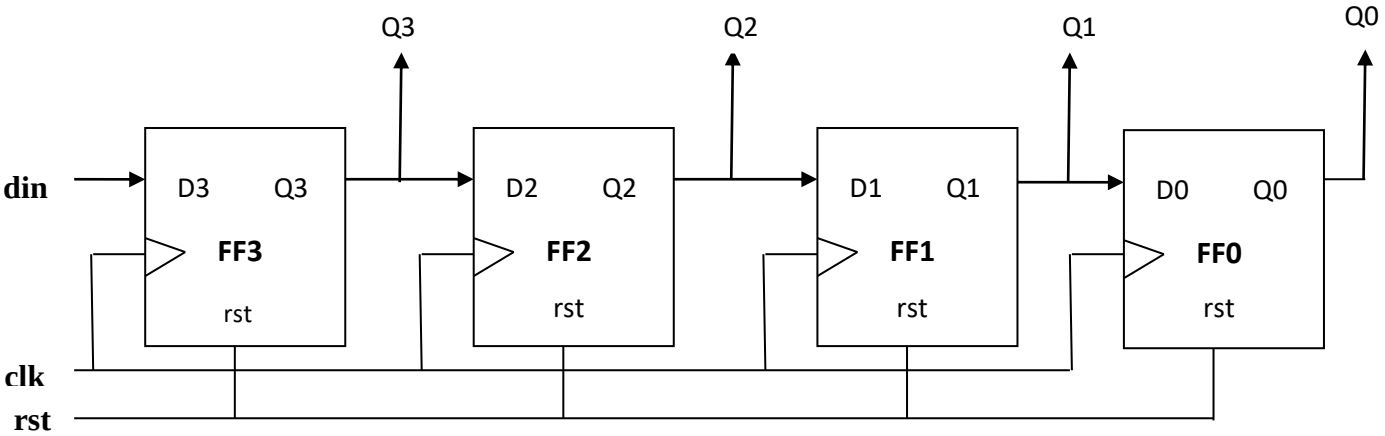


Fig 2: 4 bit SIPO shift register

Parallel In Serial Out

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 shown in fig.3 shows a 4 bit PISO register. It consists of 4 DFFs connected as shown in the fig,all flip flops driven by same clock signal clk. It also has rst input to reset the output. The parallel inputs are given through 2x1 MUX's to each of the flip flops individually. The other input of MUX is connected to output of the previous flip flop as can be seen the fig.2. The output of the MUX is connected to the input of next flip flop.
D3,D2,D1,D0 are the parallel inputs and SO is the serial output.
LD/$\overline{\text{SH}}$ signal loads the data into flip flops when 1 and shifts the data when 0.

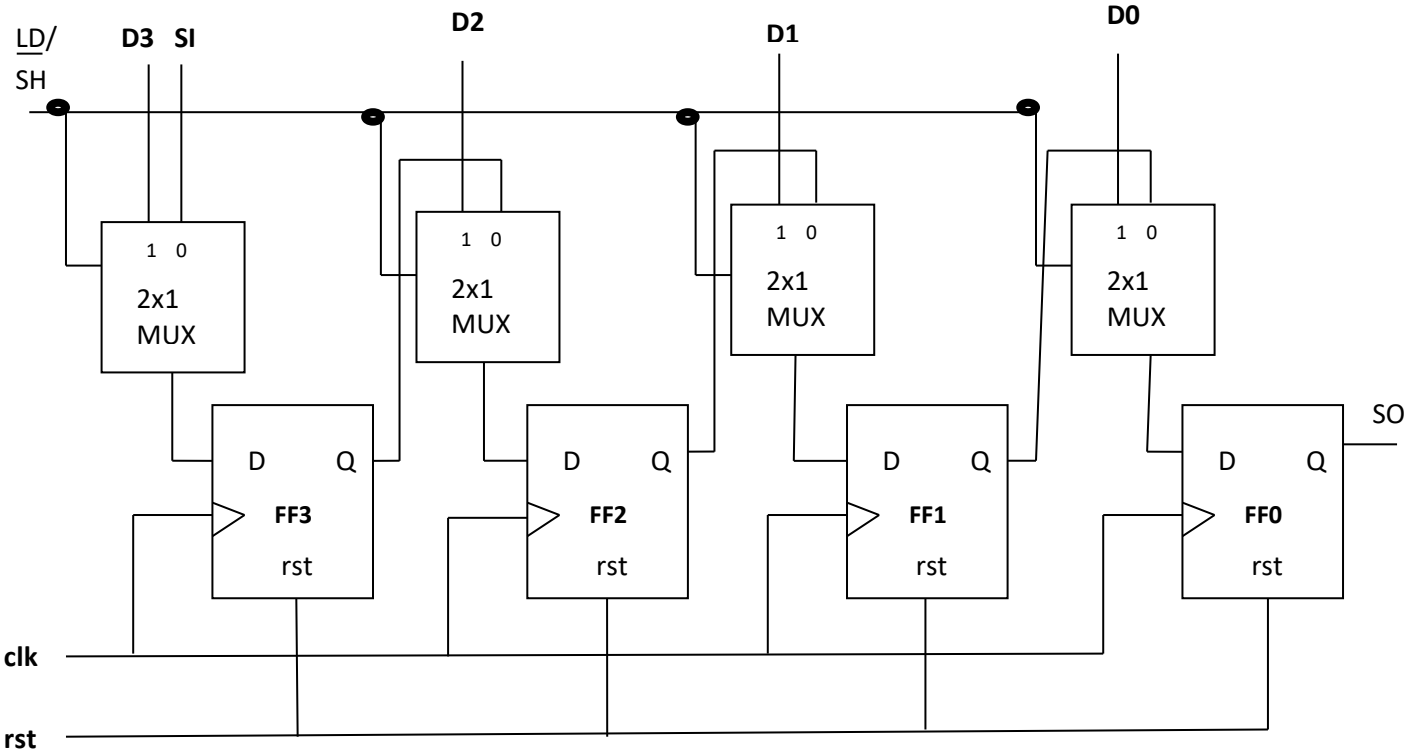


Fig 3. 4 bit PISO shift register

Parallel-In Parallel-Out Shift Register (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 in fig.4 shows a parallel-in-parallel-out shift register. The circuit consists of four D flip-flops which are connected. The reset (rst) signal and clock(clk) 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.

D3,D2,D1,D0 are parallel inputs.

Q3,Q2,Q1,Q0 are parallel output.

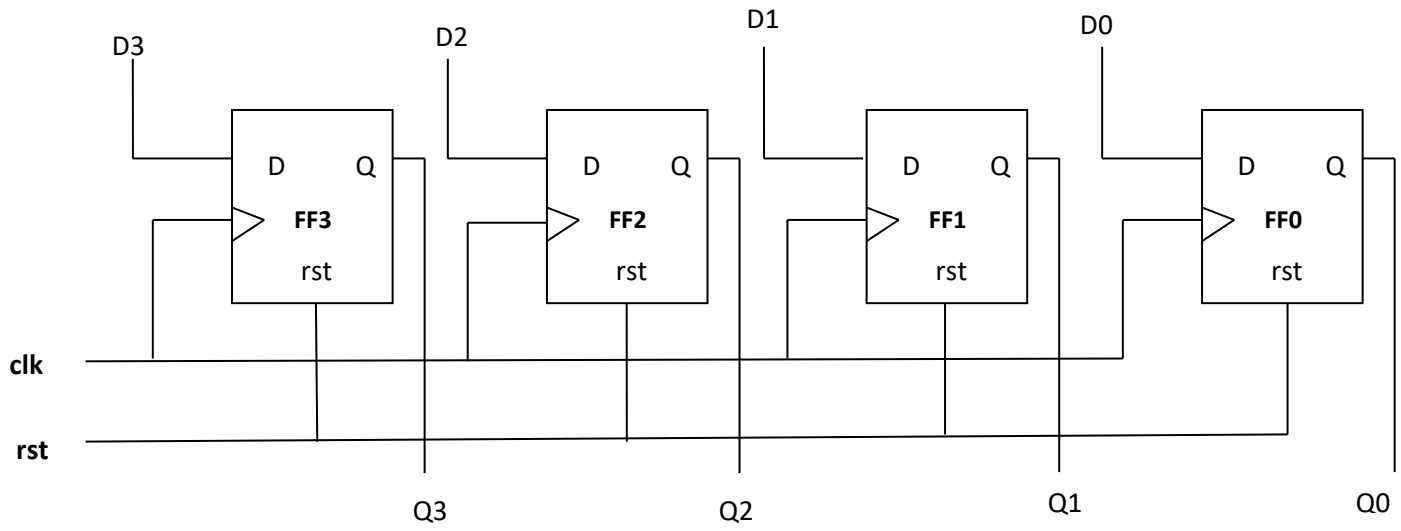


Fig 4. 4 bit PIPO shift register

VHDL CODE:-

<u>STEPS FOR PROGRAM:-</u>

Step 1.Library /Package Declaration : Involves declaration of all libraries and respective packages used in the design.

```
LIBRARY library_name;
USE library_name.package_name.all;
```

Step 2.Entity:

```
ENTITY entity_name is
        PORT(signal_name(s): mode signal_type;
                signal_name(s): mode signal_type;
                …);
end ENTITY entity_name;
```

Signals of the same mode and signal_type can be grouped on 1 line.

MODE describes the direction data is transferred through port

- in – data flows into the port
- out – data flows out of port *only*
- buffer – data flows out of port *as well as read* internally.
- inout – bi-directional data flow into and out of port

SIGNAL_TYPE defines the data type for the signal(s)

- bit – single signals that can have logic values 0 and 1.
- bit_vector – bus signals(vector form of bit) that can have logic values 0 and 1.
- std_logic – part of std_logic_1164 package of IEEE library. Used to represent 2 value logical values i.e 0 and 1 as well as other values such as high impedance ,don't care and others as described below.
- std_logic_vector – bus signals (vector form of std_logic) but IEEE standard for simulation and synthesis note that all vectors must have a range specified example for a 4 bit bus: bit_vector (3 downto 0) or std_logic_vector (3 downto 0).

In order to use std_logic and std_logic_vector we must include the library and package usage declarations in the VHDL model before the entity statement as follows:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
```

Values for std-logic:

U un-initialized (undefined logic value)
X forced unknown logic value
0 Logic low
1 Logic High
Z high impedance (tri-state)
W weak unknown
L weak 0
H weak 1
- don't care value (for synthesis minimization)

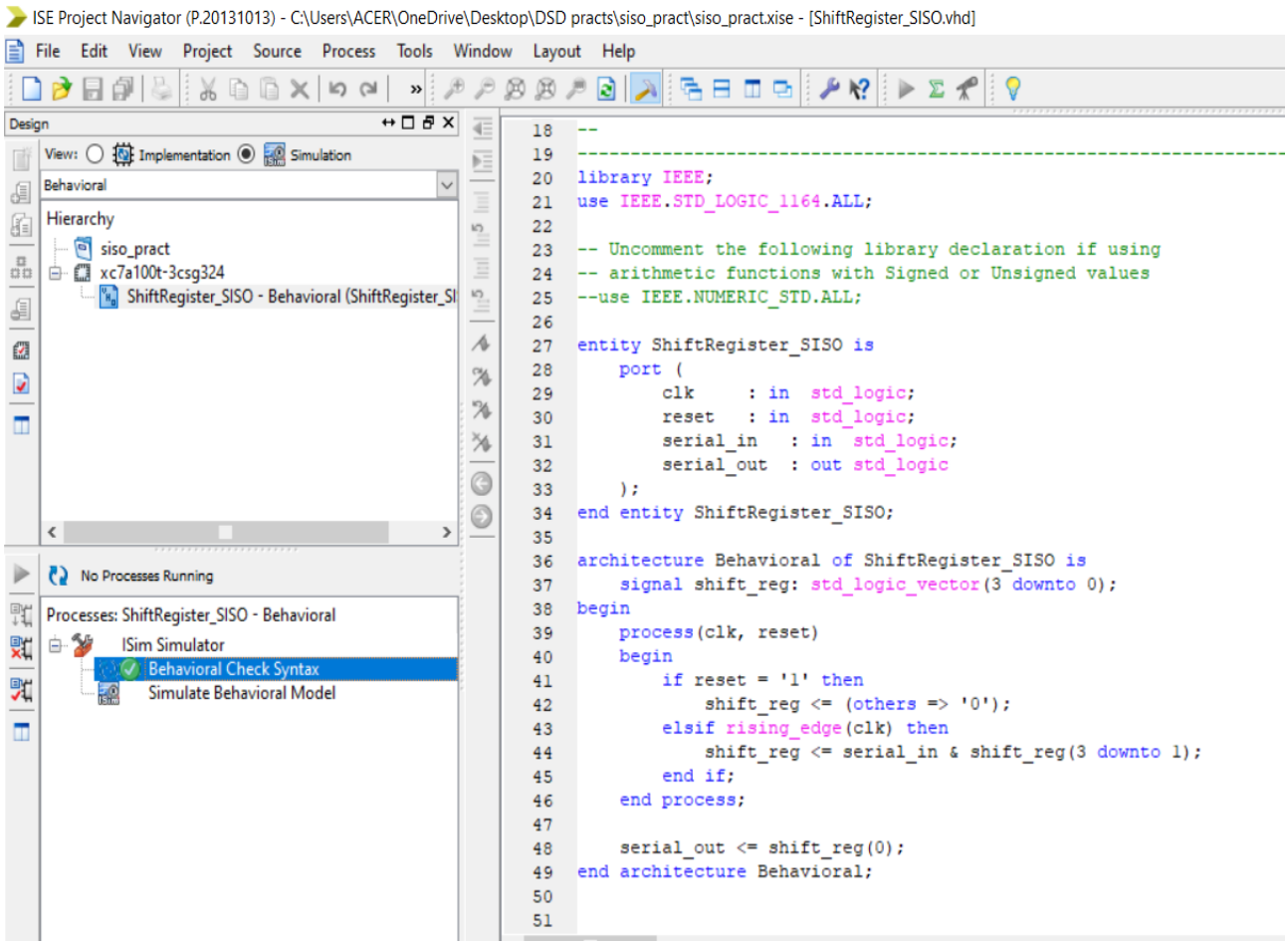
Step 3: Architecture Declaration

```
architecture architecture name of entity_name
  architecture_declarative_part;
begin
Statements;
 end architecture_name;
```

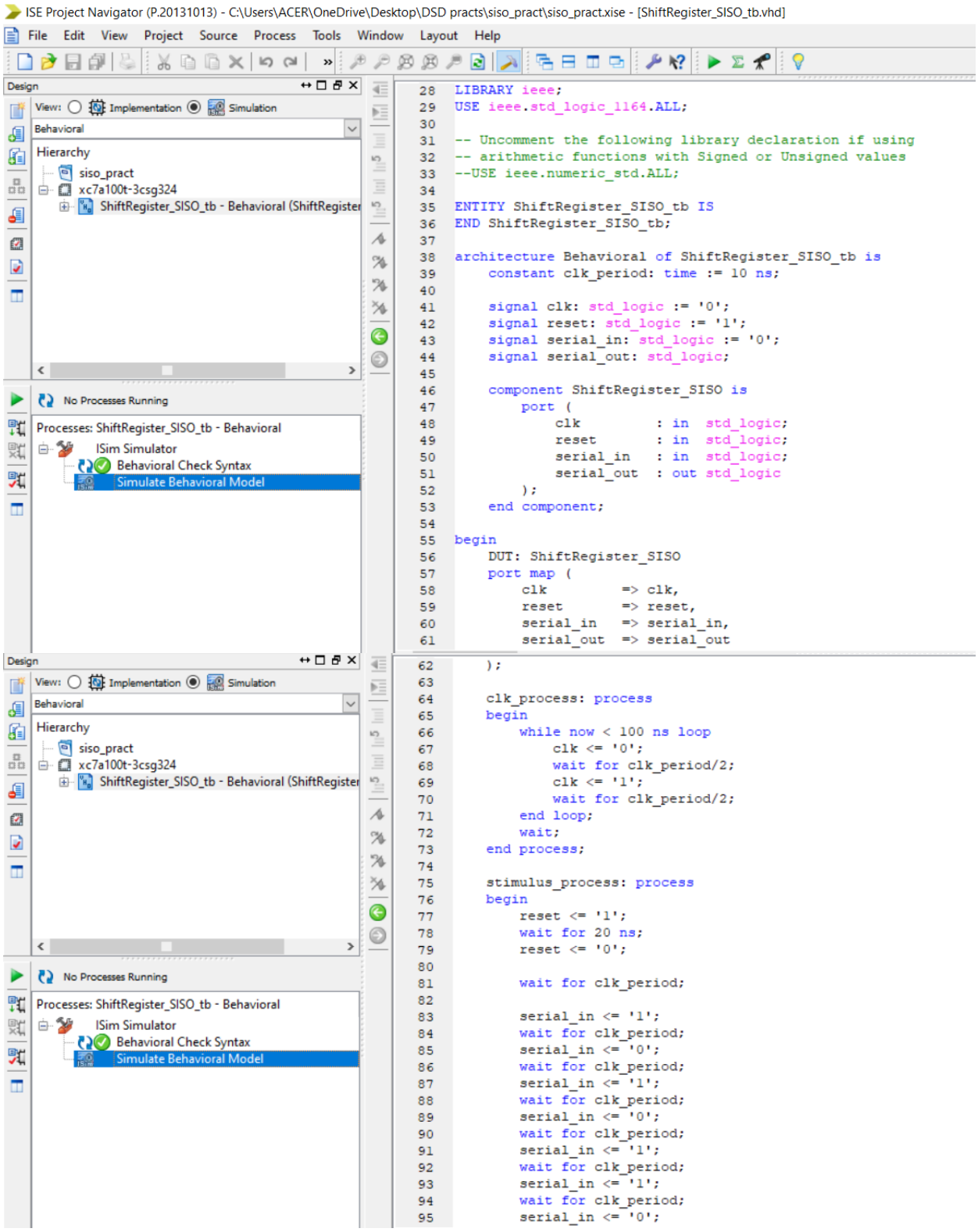
Here we should specify the entity name for which we are writing the architecture body. The architecture statements should be inside the begin and end keyword. Architecture declarative part may contain variables, constants, or component declaration.

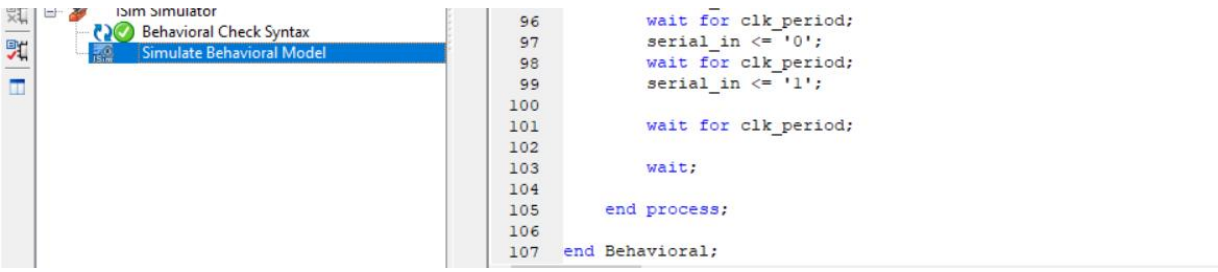
Step 4: Simulate the VHDL code and remove the syntax errors if any.

Step 5: Write the testbench and verify the design .
CODE: SISO Shift Register

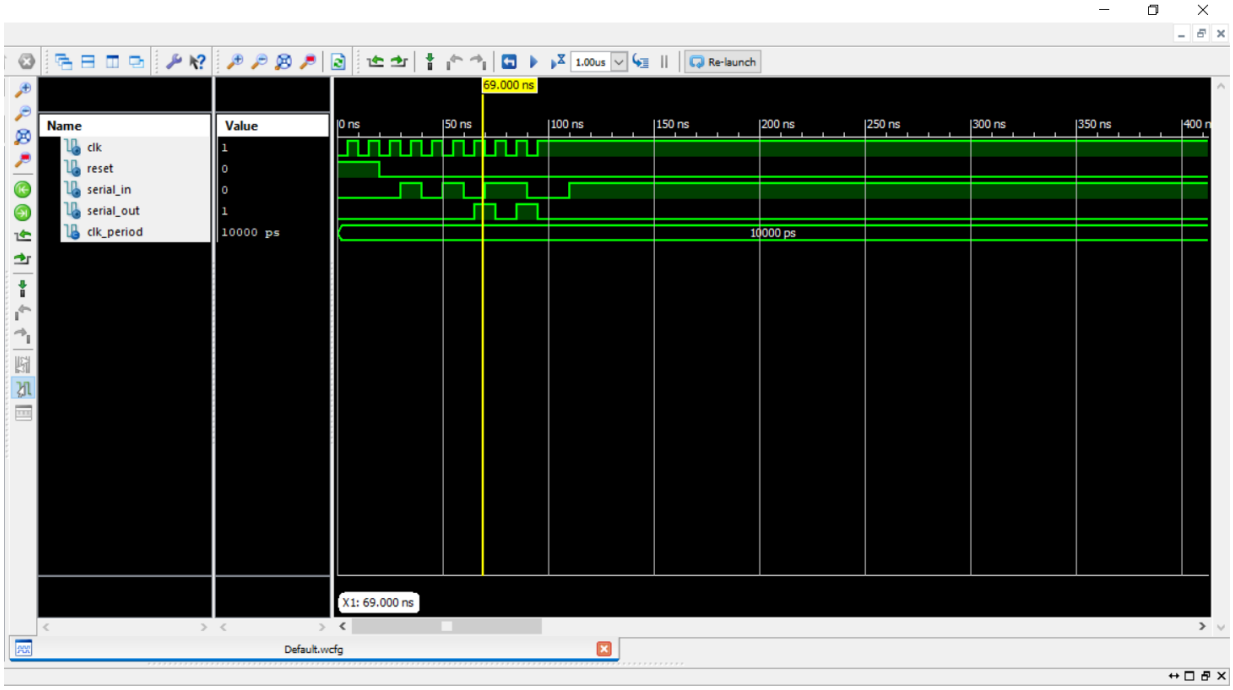


TESTBENCH:

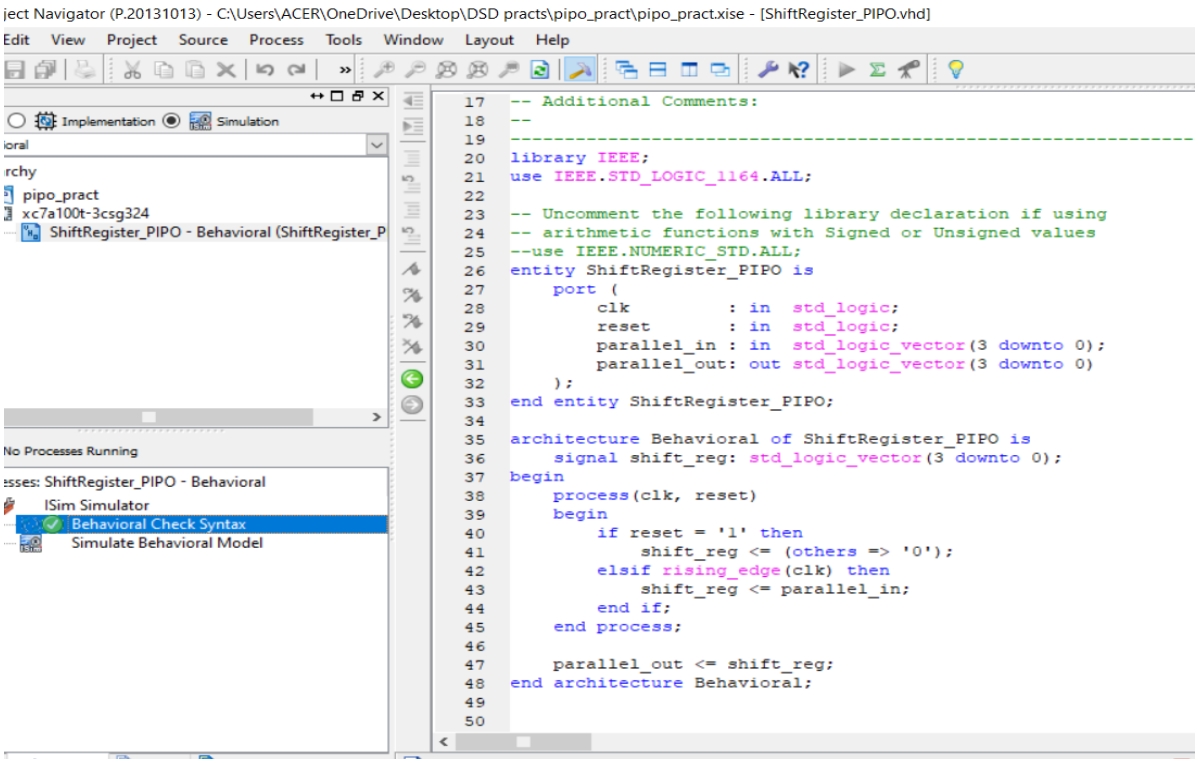

ISE Project Navigator (P.20131013) - C:\Users\ACER\OneDrive\Desktop\DSD practs\siso_pract\siso_pract.xise - [ShiftRegister_SISO_tb.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
siso_pract
xc7a100t-3csg324
ShiftRegister_SISO_tb - Behavioral (ShiftRegister
No Processes Running
Processes: ShiftRegister_SISO_tb - Behavioral
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;
ENTITY ShiftRegister_SISO_tb IS
END ShiftRegister_SISO_tb;
architecture Behavioral of ShiftRegister_SISO_tb is
constant clk_period: time := 10 ns;
signal clk: std_logic := '0';
signal reset: std_logic := '1';
signal serial_in: std_logic := '0';
signal serial_out: std_logic;
component ShiftRegister_SISO is
port (
clk : in std_logic;
reset : in std_logic;
serial_in : in std_logic;
serial_out : out std_logic
);
end component;
begin
DUT: ShiftRegister_SISO
port map (
clk => clk,
reset => reset,
serial_in => serial_in,
serial_out => serial_out
);
clk_process: process
begin
while now < 100 ns loop
clk <= '0';
wait for clk_period/2;
clk <= '1';
wait for clk_period/2;
end loop;
wait;
end process;
stimulus_process: process
begin
reset <= '1';
wait for 20 ns;
reset <= '0';
wait for clk_period;
serial_in <= '1';
wait for clk_period;
serial_in <= '0';
wait for clk_period;
serial_in <= '1';
wait for clk_period;
serial_in <= '0';
wait for clk_period;
serial_in <= '1';
wait for clk_period;
serial_in <= '1';
wait for clk_period;
serial_in <= '0';



WAVEFORM:



CODE: PIPO Shift Register



TESTBENCH:

ISE Project Navigator (P.20131013) - C:\Users\ACER\OneDrive\Desktop\DSD practs\pipo_pract\pipo_pract.xise - [ShiftRegister_PIPO_tb.vhd]

File Edit View Project Source Process Tools Window Layout Help

Design

View: Implementation Simulation

Behavioral

Hierarchy

- pipo_pract
- xc7a100t-3csg324
  - ShiftRegister_PIPO_tb - Behavioral (ShiftRegiste

No Processes Running

Processes: ShiftRegister_PIPO_tb - Behavioral

- ISim Simulator
  - Behavioral Check Syntax
  - Simulate Behavioral Model

View: Implementation Simulation

Behavioral

Hierarchy

- pipo_pract
- xc7a100t-3csg324
  - ShiftRegister_PIPO_tb - Behavioral (ShiftRegiste

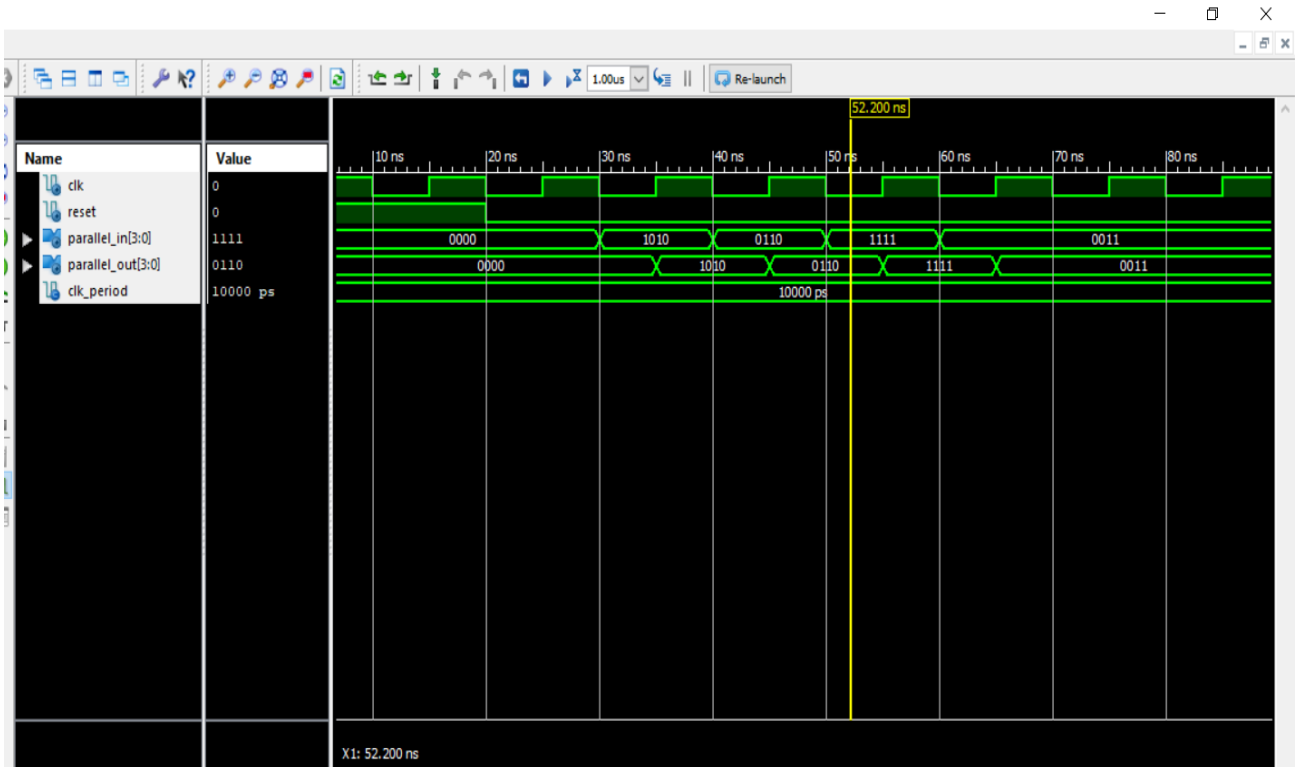
No Processes Running

Processes: ShiftRegister_PIPO_tb - Behavioral

- ISim Simulator
  - Behavioral Check Syntax
  - Simulate Behavioral Model

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY ShiftRegister_PIPO_tb IS
END ShiftRegister_PIPO_tb;


architecture Behavioral of ShiftRegister_PIPO_tb is
    constant clk_period: time := 10 ns;

    signal clk: std_logic := '0';
    signal reset: std_logic := '1';
    signal parallel_in: std_logic_vector(3 downto 0) := (others => '0
    signal parallel_out: std_logic_vector(3 downto 0);

    component ShiftRegister_PIPO is
        port (
            clk             : in  std_logic;
            reset           : in  std_logic;
            parallel_in     : in  std_logic_vector(3 downto 0);
            parallel_out    : out std_logic_vector(3 downto 0)
        );
    end component;

begin
    DUT: ShiftRegister_PIPO
    port map (
        clk             => clk,
        reset           => reset,
        parallel_in     => parallel_in,
        parallel_out    => parallel_out
    );

    clk_process: process
    begin
        while now < 100 ns loop
            clk <= '0';
            wait for clk_period/2;
            clk <= '1';
            wait for clk_period/2;
        end loop;
        wait;
    end process;

    stimulus_process: process
    begin
        reset <= '1';
        wait for 20 ns;
        reset <= '0';

        wait for clk_period;

        parallel_in <= "1010";
        wait for clk_period;
        parallel_in <= "0110";
        wait for clk_period;
        parallel_in <= "1111";
        wait for clk_period;
        parallel_in <= "0011";
        wait for clk_period;
        wait;
    end process;
end architecture Behavioral;
```

WAVEFORM:



RESULT:-

The VHDL code for shift register is executed and desired output is obtained.

CONCLUSION:

Here, I successfully design the serial in serial out shift register and parallel in parallel out shift register with their test benches to give stimulus.

DISCUSSION & VIVA VOCE

1) Design a universal shift register.
2) What are the applications of Shift registers.?
3) How to design serial adder using shift register?

REFERENCE:

- VHDL Primer–J Bhasker –Pearson Education
- NPTEL Video Lecture link- https://youtu.be/Iecj9xmIfXM?list=PL803563859BF7ED8C