## Amrutvahini College of Engineering, Sangamner Department of Electronics and Telecommunication Engineering

**Experiment No.: 03**

## Date of Performance:

**AIM: -** To write VHDL code for Universal Shift Register. **OBJECTIVE: -** To study PIPO, PISO, SISO and SIPO. **THEORY: -**

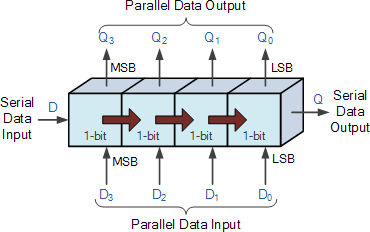
# Universal Shift Register:

The **Shift Register** is another type of sequential logic circuit that can be used for the storage or the transfer of data in the form of binary numbers. This sequential device loads the data present on its inputs and then moves or “shifts” it to its output once every clock cycle, hence the name “shift register”. A shift register basically consists of several single bit “D-Type Data Latches”, one for each data bit, either a logic “0” or a “1”, connected together in a serial type daisy-chain arrangement so that the output from one data latch becomes the input of the next latch and so on. Data bits may be fed in or out of a shift register serially, that is one after the other from either the left or the right direction, or all together at the same time in a parallel configuration. The number of individual data latches required to make up a single **Shift Register** device is usually determined by the number of bits to be stored with the most common being 8-bits (one byte) wide constructed from eight individual data latches.

*Shift Registers* are used for data storage or for the movement of data and are therefore commonly used inside calculators or computers to store data such as two binary numbers before they are added together, or to convert the data from either a serial to parallel or parallel to serial format. The individual data latches that make up a single shift register are all driven by a common clock (Clk ) signal making them synchronous devices. Shift register IC’s are generally provided with a *clear* or *reset* connection so that they can be “SET” or “RESET” as required. Generally, shift registers operate in one of four different modes with the basic movement of data through a shift register being:

* Serial-in to Parallel-out (SIPO) - the register is loaded with serial data, one bit at a time, with the stored data being available at the output in parallel form.
* Serial-in to Serial-out (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.
* Parallel-in to Serial-out (PISO) - the parallel data is loaded into the register simultaneously and is shifted out of the register serially one bit at a time under clock control.
* Parallel-in to parallel-out (PIPO) - the parallel data is loaded simultaneously into the register, and transferred together to their respective outputs by the same clock pulse.

The effect of data movement from left to right through a shift register can be presented graphically as:



Also, the directional movement of the data through a shift register can be either to the left, (left shifting) to the right, (right shifting) left-in but right-out, (rotation) or both left and right shifting within the same register thereby making it *bidirectional*. In this tutorial it is assumed that all the data shifts to the right, (right shifting).

m (1:0)

Universal

Shift Register

d (3:0) q (3:0)

clk

load

clear

|  |  |
| --- | --- |
| **m** | **Operation** |
| **00** | **PIPO** |
| **01** | **PISO** |
| **10** | **SISO** |
| **11** | **SIPO** |

**Conclusion:**

# Questions:

Q:1. Write a short note on a) Design validation b) Clock skew c) Clock Jitter

# Reference Book:

Douglas L Perry, “Data Types” in *VHDL Programming by Example*, 4th Edition, McGraw-Hill, New Delhi, India, 2002, ch.4, pp-73-105

# VHDL Program:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_arith.ALL;

use IEEE.STD\_LOGIC\_unsigned.ALL;

use IEEE.numeric\_std.ALL;

entity UNI\_SHIFT\_3 is

Port ( din : in STD\_LOGIC\_VECTOR (3 downto 0);

mod1 : in STD\_LOGIC\_VECTOR (1 downto 0);

clk : in STD\_LOGIC;

rst : in STD\_LOGIC;

dout : inout STD\_LOGIC\_VECTOR (3 downto 0));

end UNI\_SHIFT\_3;

architecture Behavioral of UNI\_SHIFT\_3 is

signal MSBIN, LSBIN : std\_logic; --used to perform LF & Rt SISO

signal temp : std\_logic\_vector(3 downto 0); --use for SIPO

begin

MSBIN<= din(3);

LSBIN<= din(0);

PROCESS(CLK, RST)

begin

if(rst='1') then

dout <= "0000";

temp <= "0000";

elsif (clk'event and clk ='1') then

case mod1 is

when "00" =>

dout <= MSBIN & dout(3 downto 1); --- SISO right shift

when "01" =>

dout <= din; ----PIPO

when "10" =>

temp <= MSBIN & temp (3 downto 1);

dout<= temp; -----SIPO

when "11" =>

dout <= dout (2 downto 0) & LSBIN ; ---- SISO left shift

when others =>

dout <= "0000";

end case;

end if ;

end process;

end Behavioral ;

# Testbench Program:

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.std\_logic\_unsigned.all; USE ieee.numeric\_std.ALL;

ENTITY test\_vhd IS END test\_vhd;

ARCHITECTURE behavior OF test\_vhd IS COMPONENT main

PORT(

din: IN std\_logic\_vector(3 downto 0);

clk : IN std\_logic;

rst: IN std\_logic;

mod1: IN std\_logic\_vector(1 downto 0);

dout: OUT std\_logic\_vector(3 downto 0)

);

END COMPONENT;

SIGNAL clk : std\_logic := '0'; SIGNAL rst : std\_logic := '0';

SIGNAL din : std\_logic\_vector(3 downto 0) := (others=>'0'); SIGNAL mod1 : std\_logic\_vector(1 downto 0) := (others=>'0');

--Outputs

SIGNAL dout : std\_logic\_vector(3 downto 0);

BEGIN

-- Instantiate the Unit Under Test (UUT) uut: main PORT MAP(

din => din,

clk => clk, rst => rst,

mod1 => mod1,

dout => dout

);

PROCESS BEGIN

wait for 10 ns ; clk<= not clk;

END PROCESS;

PROCESS

BEGIN m<="11";

d<="1001";

load<='1';

wait for 60 ns ; load<='0';

wait for 300 ns; load<='1'; m<="01";

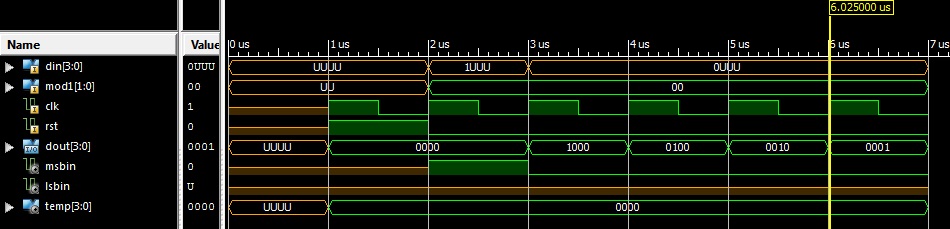
wait for 50 ns; load<='0'; wait;

END PROCESS;

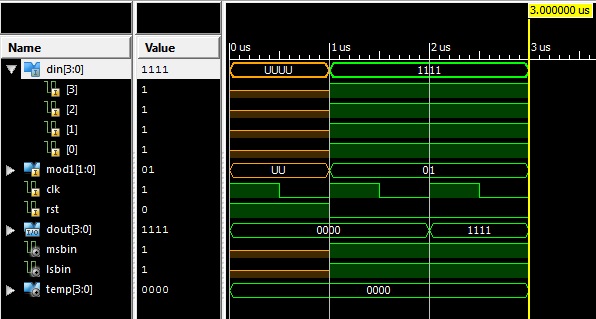
END;

# Testbench Waveform Generated:

# Output of SISO right shift

****

# Output of PIPO operation

****