**LAB NOTE**

**Subject: Digital Design Principles**

**Topic: Shift Register**

**Student: Minh Quan Tran**

**July 26th, 2024**

Table of Contents

[1. Objectives 4](#_Toc172325554)

[2. Theory and Design 6](#_Toc172325555)

[2.1 Theory 6](#_Toc172325556)

[2.1.1 Ring Counter 6](#_Toc172325557)

[2.1.2 Johnson Counter 6](#_Toc172325558)

[2.2 Requirement 6](#_Toc172325559)

[2.3 Solution 6](#_Toc172325560)

[3. VHDL and Verilog 6](#_Toc172325561)

[3.1 VHDL code 6](#_Toc172325562)

[3.1.1 Top entity 6](#_Toc172325563)

[3.1.2 Component Ring Counter 8](#_Toc172325564)

[3.1.3 Component Ring Counter 9](#_Toc172325565)

[3.2 Verilog code 10](#_Toc172325566)

[3.2.1 Top Module 10](#_Toc172325567)

[3.2.2 Module Timer 13](#_Toc172325568)

[4. Pin Planner 15](#_Toc172325569)

[4.1 Input and Output 15](#_Toc172325570)

[5. Result 17](#_Toc172325571)

**TABLE OF FIGURES**

[Figure 2‑1: Counters’s Truth Table 6](#_Toc172325572)

[Figure 4‑1: SCEMA5F31C6N board 15](#_Toc172325573)

[Figure 4‑2: SW0 and SW1 Pin No 16](#_Toc172325574)

[Figure 4‑3: LEDR0's Pin No 16](#_Toc172325575)

[Figure 4‑4: HEX's Pin No 16](#_Toc172325576)

[Figure 5‑1: When input = ‘000’. 17](#_Toc172325577)

[Figure 5‑2: When input = ‘111’ 17](#_Toc172325578)

[Figure 5‑3: When input ‘101’ 18](#_Toc172325579)

[Figure 5‑4: Input ‘101’ with Complement mode 18](#_Toc172325580)

[Figure 5‑5: Input ‘101’ with Multiline mode 19](#_Toc172325581)

[Figure 5‑6: Result for adding them up 19](#_Toc172325582)

# Objectives

* Design a Counters using Quartus software.
* Write a VHDL and Verilog code from the design.
* Push the code and run to the SCEMA5F31C6N board.

# Theory and Design

## Theory

A shift register is a digital circuit used for storing and transferring data, consisting of a series of flip-flops connected in a chain. Data is shifted through these flip-flops sequentially, either left or right, with each clock pulse. There are various types of shift registers, including Serial-In Serial-Out (SISO), Serial-In Parallel-Out (SIPO), Parallel-In Serial-Out (PISO), and Parallel-In Parallel-Out (PIPO), each serving different data input and output requirements. Shift registers are crucial in digital systems for tasks such as data storage, data transfer, and data conversion between serial and parallel formats.

## Requirement

* Use VHDL and Verilog to implement the shift and display to LED [1:4].
* Introduction to the VHDL Case statement.
* **Above and beyond:** Adding Complement mode

## Solution

To design this system, first need to identify input and output:

* Input will be from BTN0 to BTN3
  + BTN0 : CLK
  + BTN1 : Speed
  + BTN3 : Reset
* Output will be LED[4:1].

Then identify what the output will be, from the 2 inputs,

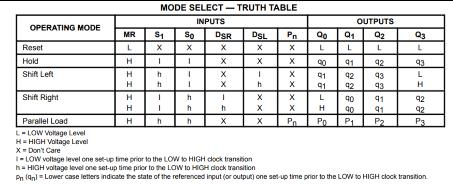


Figure 2‑1: Shift Register’s Truth Table

# VHDL and Verilog

## VHDL code

### Top entity

**library ieee;**

**use ieee.std\_logic\_1164.all;**

**entity MTran\_Lab9\_VHDL\_ShiftRegister is**

**port**

**(**

**-- Inputs**

**BTN : in std\_logic;**

**Speed : in std\_logic;**

**Reset : in std\_logic;**

**SW : in std\_logic\_vector (7 downto 0);**

**-- Outputs**

**LED : out std\_logic\_vector (3 downto 0);**

**HEX0 : out std\_logic\_vector (6 downto 0)**

**);**

**end entity;**

**architecture behavioral of MTran\_Lab9\_VHDL\_ShiftRegister is**

**-- Signal Declaration**

**signal Counter : integer := 1;**

**signal Output : std\_logic\_vector (3 downto 0) := "0000";**

**signal Sel : std\_logic\_Vector (1 downto 0);**

**-- Function declaration**

**function SevenSegmentDisplay (Number : integer) return std\_logic\_vector is**

**begin**

**case (Number) is**

**when 0 => return "1000000";**

**when 1 => return "1111001";**

**when 2 => return "0100100";**

**when 3 => return "0110000";**

**when 4 => return "0011001";**

**when 5 => return "0010010";**

**when 6 => return "0000010";**

**when 7 => return "1111000";**

**when 8 => return "0000000";**

**when 9 => return "0010000";**

**when others => return "1111111";**

**end case;**

**end SevenSegmentDisplay;**

**begin**

**-- Process**

**-- Check Sel**

**process(SW(6),SW(7)) is**

**begin**

**Sel <= SW(7) & SW(6);**

**end process;**

**-- Check Speed**

**process(Speed) is**

**begin**

**if Speed ='0' then**

**if Counter < 4 then**

**Counter <= Counter \* 2;**

**else**

**Counter <= 1;**

**end if;**

**end if;**

**end process;**

**-- Main code (Shift)**

**process(Sel, SW, BTN, Reset) is**

**begin**

**-- Reset BTN**

**if Reset = '0' then**

**Output <= "0000";**

**elsif rising\_edge(BTN) then**

**if Counter = 1 then**

**case (Sel) is**

**when "00" => NULL;**

**when "01" => Output <= Output(2 downto 0) & SW(0);**

**when "10" => Output <= SW(5) & Output(3 downto 1);**

**when "11" => Output <= SW(4 downto 1);**

**end case;**

**elsif Counter = 2 then**

**case (Sel) is**

**when "00" => NULL;**

**when "01" => Output <= Output(1 downto 0) & SW(0) & SW(0);**

**when "10" => Output <= SW(5) & SW(5) & Output(3 downto 2);**

**when "11" => Output <= SW(4 downto 1);**

**end case;**

**elsif Counter = 4 then**

**case (Sel) is**

**when "00" => NULL;**

**when "01" => Output <= SW(0) & SW(0) & SW(0) & SW(0);**

**when "10" => Output <= SW(5) & SW(5) & SW(5) & SW(5);**

**when "11" => Output <= SW(4 downto 1);**

**end case;**

**end if;**

**end if;**

**end process;**

**-- Output**

**process(Output, Counter)**

**begin**

**LED <= Output;**

**HEX0 <= SevenSegmentDisplay(Counter);**

**end process;**

**end Behavioral;**

## Verilog code

**module MTran\_Lab9\_Verilog\_ShiftRegister**

**(**

**input wire BTN,**

**input wire Speed,**

**input wire Reset,**

**input wire [7:0] SW,**

**output reg [3:0] LED,**

**output reg [6:0] HEX0**

**);**

**// Signal Declarations**

**reg [3:0] Output;**

**reg [1:0] Sel;**

**integer Counter = 1;**

**// Seven Segment Display function**

**function [6:0] SevenSegmentDisplay;**

**input integer Number;**

**begin**

**case (Number)**

**0: SevenSegmentDisplay = 7'b1000000;**

**1: SevenSegmentDisplay = 7'b1111001;**

**2: SevenSegmentDisplay = 7'b0100100;**

**3: SevenSegmentDisplay = 7'b0110000;**

**4: SevenSegmentDisplay = 7'b0011001;**

**5: SevenSegmentDisplay = 7'b0010010;**

**6: SevenSegmentDisplay = 7'b0000010;**

**7: SevenSegmentDisplay = 7'b1111000;**

**8: SevenSegmentDisplay = 7'b0000000;**

**9: SevenSegmentDisplay = 7'b0010000;**

**default: SevenSegmentDisplay = 7'b1111111;**

**endcase**

**end**

**endfunction**

**// Check Sel**

**always @(\*)**

**begin**

**Sel = {SW[7], SW[6]};**

**end**

**// Check Speed**

**always @(negedge Speed)**

**begin**

**if (Speed == 0)**

**begin**

**if (Counter < 4)**

**Counter = Counter \* 2;**

**else**

**Counter = 1;**

**end**

**end**

**// Main code (Shift)**

**always @(negedge BTN or negedge Reset)**

**begin**

**if (Reset == 0)**

**Output <= 4'b0000;**

**else if (BTN == 0)**

**begin**

**if (Counter == 1)**

**begin**

**case (Sel)**

**2'b00: Output = Output;**

**2'b01: Output = {Output[2:0], SW[0]};**

**2'b10: Output = {SW[5], Output[3:1]};**

**2'b11: Output = SW[4:1];**

**endcase**

**end**

**else if (Counter == 2)**

**begin**

**case (Sel)**

**2'b00: Output = Output;**

**2'b01: Output = {Output[1:0], SW[0], SW[0]};**

**2'b10: Output = {SW[5], SW[5], Output[3:2]};**

**2'b11: Output = SW[4:1];**

**endcase**

**end**

**else if (Counter == 4)**

**begin**

**case (Sel)**

**2'b00: Output = Output;**

**2'b01: Output = {SW[0], SW[0], SW[0], SW[0]};**

**2'b10: Output = {SW[5], SW[5], SW[5], SW[5]};**

**2'b11: Output = SW[4:1];**

**endcase**

**end**

**end**

**end**

**// Output**

**always @(\*)**

**begin**

**LED = Output;**

**HEX0 = SevenSegmentDisplay(Counter);**

**end**

**endmodule**

# Pin Planner

## Input and Output

A close-up of a circuit board

Description automatically generated

Figure 4‑1: SCEMA5F31C6N board

Assigning:

SW[0]: SW0

SW[1]: SW1

SW[2]: SW2

Mode[0]: SW7

Mode[1]: SW8

EN: SW9

LED[0]: LEDR0

LED[1]: LEDR1

LED[2]: LEDR2

LED[3]: LEDR3

LED[4]: LEDR4

LED[5]: LEDR5

LED[6]: LEDR6

LED[7]: LEDR7

HEX0[0]: HEX0[0]

HEX0[1]: HEX0[1]

HEX0[2]: HEX0[2]

HEX0[3]: HEX0[3]

HEX0[4]: HEX0[4]

HEX0[5]: HEX0[5]

HEX0[6]: HEX0[6]

From the DE1\_SoC\_User\_Manual,

A screenshot of a slide switch

Description automatically generated

Figure 4‑2: SW0 and SW1 Pin No

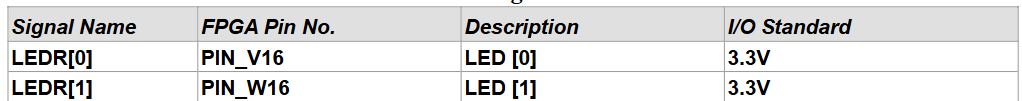


Figure 4‑3: LEDR0's Pin No

 A group of black text

Description automatically generated

Figure 4‑4: HEX's Pin No

# Result

A computer chip on a desk

Description automatically generated

Figure 5‑1: When input = ‘000’.

SW2 = 0, SW1 = 0, SW0 = 0 which should be 0, **LED0** asserted and the 7 segment LED displaying **0**.

A close up of a computer

Description automatically generated

Figure 5‑2: When input = ‘111’

SW2 = 1, SW1 = 1, SW0 = 1 which should be 7, **LED7** asserted and the 7 segment LED displaying **7**.

A close up of a computer chip

Description automatically generated

Figure 5‑3: When input ‘101’

SW2 = 1, SW1 = 0, SW0 = 1 which should be 5, **LED5** asserted and the 7 segment LED displaying **5**.

**Different mode:**

* **Complement mode:**

A close up of a computer

Description automatically generated

Figure 5‑4: Input ‘101’ with Complement mode

In this mode, rather than LED5 assert, all other LED except LED5 asserted. To activate this mode SW [8:7] = ‘01’.

* **Multiline Mode:**

A close up of a circuit board

Description automatically generated

Figure 5‑5: Input ‘101’ with Multiline mode

In this mode, when input is ‘101’ which is **5**, LED0 to LED5 lit up. To activate this mode SW [8:7] = ‘10’.

* **Complement Multiline Mode:**

A close up of a circuit board

Description automatically generated

Figure 5‑6: Result for adding them up

In this mode, it take the output of Multiline Mode and complemented it. To activate this mode SW [8:7] = ‘11’

**REFERENCES**