**LAB NOTE**

**Subject: Digital Design Principles**

**Topic: Counters**

**Student: Minh Quan Tran**

**July 12th, 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

### Ring Counter

A ring counter is a type of counter composed of a circular shift register, where the output of the last flip-flop is fed back to the input of the first flip-flop. In a ring counter, only one flip-flop is set to '1' (or 'high') at any time, and this '1' circulates around the ring. For an N-bit ring counter, the sequence repeats every N states. This type of counter is simple but has limited use due to the fixed number of states equal to the number of flip-flops.

### Johnson Counter

A Johnson counter, also known as a twisted ring counter or Möbius counter, is a modified version of the ring counter. In a Johnson counter, the complement of the output of the last flip-flop is fed back to the input of the first flip-flop. This configuration creates a sequence of states that is twice the number of flip-flops, effectively doubling the counting states compared to a standard ring counter. For an N-bit Johnson counter, the sequence repeats every 2N states, making it more efficient in terms of state utilization.

## Requirement

* Use VHDL and Verilog to implement a Counters and display to LED [0:7].
* 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
* Output will be LED[8:0].

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

A screenshot of a table

Description automatically generated

Figure 2‑1: Counters’s Truth Table

# VHDL and Verilog

## VHDL code

### Top entity

**library** ieee;

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

**entity** MTran\_Lab8\_VHDL\_Counters **is**

**port**

(

-- Inputs

BTN : **in** std\_logic;

Reset : **in** std\_logic;

Complement : **in** std\_logic;

-- Outputs

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

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

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

HEX4 : **out** std\_logic\_vector (6 **downto** 0)

);

**end** **entity**;

**architecture** Behavioral **of** MTran\_Lab8\_VHDL\_Counters **is**

--Signal Declaration

**signal** state: integer:= 0;

**signal** RingCounterState : integer := 0;

**signal** JohnsonCounterState: integer := 0;

-- Module Declaration

**component** MTran\_Lab8\_VHDL\_RingCounter **is**

**port**

(

-- Inputs

CLK : **in** std\_logic;

Reset : **in** std\_logic;

Complement : **in** std\_logic;

-- Outputs

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

stateNumber : **out** integer

);

**end** **component**;

**component** MTran\_Lab8\_VHDL\_JohnsonCounter **is**

**port**

(

-- Inputs

CLK : **in** std\_logic;

Reset : **in** std\_logic;

Complement : **in** std\_logic;

-- Outputs

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

stateNumber : **out** integer

);

**end** **component**;

-- 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**

-- Module Institiate

module\_RingCounter: MTran\_Lab8\_VHDL\_RingCounter

**port** **map**

(

-- Inputs

CLK => BTN ,

Reset => Reset,

Complement => Complement,

-- Output

Q => LED\_Ring,

stateNumber => RingCounterState

);

module\_JohnsonCounter: MTran\_Lab8\_VHDL\_JohnsonCounter

**port** **map**

(

-- Inputs

CLK => BTN ,

Reset => Reset,

Complement => Complement,

-- Output

Q => LED\_Johnson,

stateNumber => JohnsonCounterState

);

-- Process

**process**(RingCounterState, JohnsonCounterState) **is**

**begin**

HEX0 <= SevenSegmentDisplay(RingCounterState);

HEX4 <= SevenSegmentDisplay(JohnsonCounterState);

**end** **process**;

**end** Behavioral;

### Component Ring Counter

**library** ieee;

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

**entity** MTran\_Lab8\_VHDL\_RingCounter **is**

**port**

(

-- Inputs

CLK : **in** std\_logic;

Reset : **in** std\_logic;

Complement : **in** std\_logic;

-- Outputs

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

stateNumber : **out** integer

);

**end** **entity**;

**architecture** RTL **of** MTran\_Lab8\_VHDL\_RingCounter **is**

-- Signal Declaration

**signal** state : integer := 0;

**signal** complementMode : std\_logic := '0';

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

-- Procedure

**procedure** Increment (**signal** Counter: **inout** integer) **is**

**begin**

**if** Counter < 3 **then**

Counter <= Counter + 1;

**else**

Counter <= 0;

**end** **if**;

**end** **procedure**;

-- Main code

**begin**

-- Process

**process**(CLK,Reset)

**begin**

**if** Reset = '0' **then**

state <= 0;

**else**

**if** rising\_edge(CLK) **then**

Increment(state);

**end** **if**;

**end** **if**;

**end** **process**;

**process**(state, complementMode)

**begin**

**case**(state) **is**

**when** 0 => Output <= "1000";

**when** 1 => Output <= "0100";

**when** 2 => Output <= "0010";

**when** 3 => Output <= "0001";

**when** **others** => Output <= "0000";

**end** **case**;

stateNumber <= state;

**end** **process**;

-- Check BTN1 for complement mode

**process**(Complement)

**begin**

**if** rising\_edge(Complement) **then**

complementMode <= **not** complementMode;

**end** **if**;

**end** **process**;

-- Process complement mode

**process** (complementMode) **is**

**begin**

**if** complementMode = '1' **then**

Q <= **not** Output;

**else**

Q <= Output;

**end** **if**;

**end** **process**;

**end** RTL;

### Component Ring Counter

**library** ieee;

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

**entity** MTran\_Lab8\_VHDL\_JohnsonCounter **is**

**port**

(

-- Inputs

CLK : **in** std\_logic;

Reset : **in** std\_logic;

Complement : **in** std\_logic;

-- Outputs

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

stateNumber : **out** integer

);

**end** **entity**;

**architecture** RTL **of** MTran\_Lab8\_VHDL\_JohnsonCounter **is**

-- Signal Declaration

**signal** state : integer := 0;

**signal** complementMode : std\_logic := '0';

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

-- Procedure

**procedure** Increment (**signal** Counter: **inout** integer) **is**

**begin**

**if** Counter < 7 **then**

Counter <= Counter + 1;

**else**

Counter <= 0;

**end** **if**;

**end** **procedure**;

-- Main code

**begin**

-- Process

**process**(CLK,Reset)

**begin**

**if** Reset = '0' **then**

state <= 0;

**else**

**if** rising\_edge(CLK) **then**

Increment(State);

**end** **if**;

**end** **if**;

**end** **process**;

**process**(state, complementMode)

**begin**

**case**(state) **is**

**when** 0 => Output <= "0000";

**when** 1 => Output <= "1000";

**when** 2 => Output <= "1100";

**when** 3 => Output <= "1110";

**when** 4 => Output <= "1111";

**when** 5 => Output <= "0111";

**when** 6 => Output <= "0011";

**when** 7 => Output <= "0001";

**when** **others** => Output <= "0000";

**end** **case**;

stateNumber <= state;

**end** **process**;

-- Check BTN1 for complement mode

**process**(Complement)

**begin**

**if** rising\_edge(Complement) **then**

complementMode <= **not** complementMode;

**end** **if**;

**end** **process**;

-- Process complement mode

**process** (complementMode) **is**

**begin**

**if** complementMode = '1' **then**

Q <= **not** Output;

**else**

Q <= Output;

**end** **if**;

**end** **process**;

**end** RTL;

## Verilog code

### Top Module

**module MTran\_Lab7\_Verilog\_MagnitudeComparator**

(

**input** wire CLOCK\_50,

input wire [3:0] A,

input wire [3:0] B,

input mode,

input speed,

output reg [2:0] LED,

output reg [6:0] HEX0,

output reg [6:0] HEX1,

output reg [6:0] HEX2,

output reg [6:0] HEX3,

output reg [6:0] HEX4,

output reg [6:0] HEX5

)**;**

// Signals declaration

**reg** [31:0] intA**;**

**reg** [31:0] intB**;**

**reg** Enable**;**

**wire** [31:0] Seconds1**;**

**wire** [31:0] Minutes1**;**

**wire** [31:0] Hours1**;**

**wire** [31:0] Seconds2**;**

**wire** [31:0] Minutes2**;**

**wire** [31:0] Hours2**;**

// Module Instantiation

MTran\_Lab7\_Verilog\_Timer

#(

.MAXSECONDS(60),

.MAXMINUTES(60),

.MAXHOURS(24),

.CLKFREQUENCY(500000) // 500kHz => Clock 10 times faster

)

Tim1

(

.Clk(CLOCK\_50),

.Enable(Enable),

.Seconds(Seconds1),

.Minutes(Minutes1),

.Hours(Hours1)

);

MTran\_Lab7\_Verilog\_Timer

#(

.MAXSECONDS(60),

.MAXMINUTES(60),

.MAXHOURS(24),

.CLKFREQUENCY(50000000) // 50 MHz => Normal Clk

)

Tim2

(

.Clk(CLOCK\_50),

.Enable(Enable),

.Seconds(Seconds2),

.Minutes(Minutes2),

.Hours(Hours2)

);

// Function for Seven Segment Display

function [6:0] SevenSegmentDisplay;

**input** [3:0] Number**;**

**begin**

**case** **(**Number**)**

4'h0**:** SevenSegmentDisplay = 7'b1000000**;**

4'h1**:** SevenSegmentDisplay = 7'b1111001**;**

4'h2**:** SevenSegmentDisplay = 7'b0100100**;**

4'h3**:** SevenSegmentDisplay = 7'b0110000**;**

4'h4**:** SevenSegmentDisplay = 7'b0011001**;**

4'h5**:** SevenSegmentDisplay = 7'b0010010**;**

4'h6**:** SevenSegmentDisplay = 7'b0000010**;**

4'h7**:** SevenSegmentDisplay = 7'b1111000**;**

4'h8**:** SevenSegmentDisplay = 7'b0000000**;**

4'h9**:** SevenSegmentDisplay = 7'b0010000**;**

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

**endcase**

**end**

endfunction

**always** @(posedge CLOCK\_50)

**begin**

**if** (mode == 1'b0)

**begin**

// Disable Timer

Enable = 1'b0;

// Reset HEX2

HEX2 = 7'b1111111;

// Convert A and B from std\_logic to integer

intA = A;

intB = B;

// Display A and B in decimal

// A

HEX0 = SevenSegmentDisplay(intA % 10);

HEX1 = SevenSegmentDisplay(intA / 10);

// B

HEX4 = SevenSegmentDisplay(intB % 10);

HEX5 = SevenSegmentDisplay(intB / 10);

// Compare A and B

**if** (intA > intB)

**begin**

LED = 3'b100;

HEX3 = 7'b0100111;

**end**

**else** **if** (intA == intB)

**begin**

LED = 3'b010;

HEX3 = 7'b0110111;

**end**

**else**

**begin**

LED = 3'b001;

HEX3 = 7'b0110011;

**end**

**end**

**else**

**begin**

Enable = 1'b1;

LED = 3'b000;

**if** (speed == 1'b0) //normal mode

**begin**

// Display Second

HEX0 = SevenSegmentDisplay(Seconds2 % 10);

HEX1 = SevenSegmentDisplay(Seconds2 / 10);

// Display Minute

HEX2 = SevenSegmentDisplay(Minutes2 % 10);

HEX3 = SevenSegmentDisplay(Minutes2 / 10);

// Display Hour

HEX4 = SevenSegmentDisplay(Hours2 % 10);

HEX5 = SevenSegmentDisplay(Hours2 / 10);

**end**

**else** **if** (speed == 1'b1)

**begin**

// Display Second

HEX0 = SevenSegmentDisplay(Seconds1 % 10);

HEX1 = SevenSegmentDisplay(Seconds1 / 10);

// Display Minute

HEX2 = SevenSegmentDisplay(Minutes1 % 10);

HEX3 = SevenSegmentDisplay(Minutes1 / 10);

// Display Hour

HEX4 = SevenSegmentDisplay(Hours1 % 10);

HEX5 = SevenSegmentDisplay(Hours1 / 10);

**end**

**end**

**end**

endmodule

### Module Timer

**module MTran\_Lab7\_Verilog\_Timer**

#(

**parameter** MAXSECONDS = 60,

parameter MAXMINUTES = 60,

parameter MAXHOURS = 24,

parameter CLKFREQUENCY = 50000000

)

(

input wire Clk,

input wire Enable,

inout reg [31:0] Seconds,

inout reg [31:0] Minutes,

inout reg [31:0] Hours

)**;**

// Signal Declaration

**reg** [31:0] Ticks = 0**;**

// Main Logic

**always** @(posedge Clk)

**begin**

**if** (Enable)

**begin**

// Increment Ticks

Ticks = Ticks + 1;

**if** (Ticks == CLKFREQUENCY - 1)

**begin**

Ticks = 0;

Seconds = Seconds + 1;

**end**

**if** (Seconds == MAXSECONDS)

**begin**

Seconds = 0;

Minutes = Minutes + 1;

**end**

**if** (Minutes == MAXMINUTES)

**begin**

Minutes = 0;

Hours = Hours + 1;

**end**

**if** (Hours == MAXHOURS)

**begin**

Hours = 0;

**end**

**end**

**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

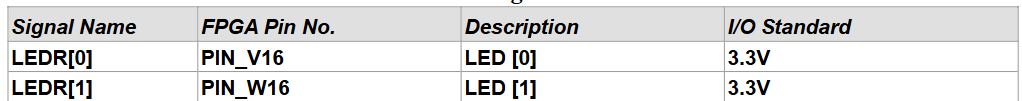


Figure 4‑3: LEDR0's Pin No

 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**