บทที่ 8
กรณีศึกษา

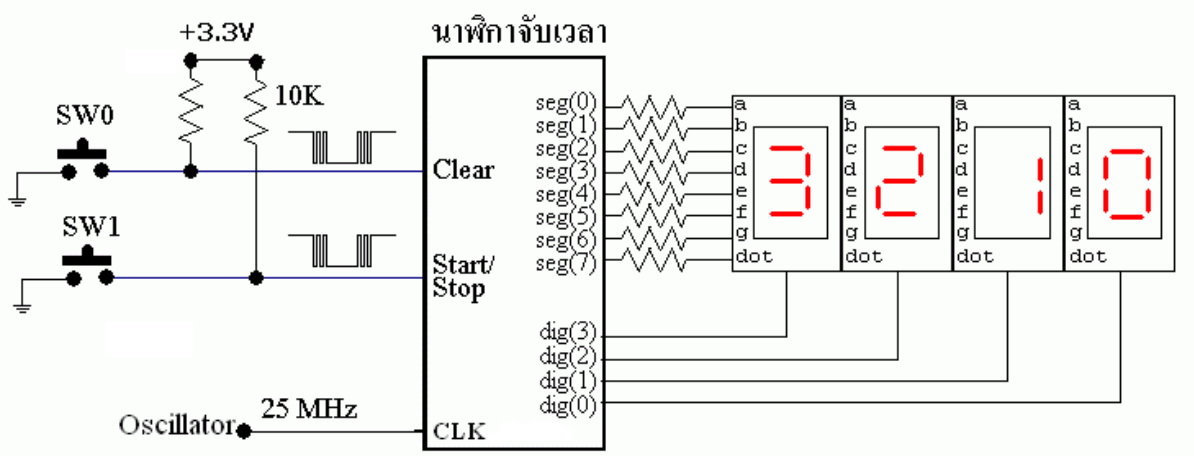
8.1 บทนำ

ในบทที่ผ่านมาเราได้ศึกษาการออกแบบวงจรแบบต่างๆ เช่นวงจรถอดรหัส วงจรถอดรหัสบีซีดีเป็นรหัสสำหรับแอลอีดี 7 ส่วน วงจรเข้ารหัส วงจรมัลติเพลกซ์ วงจรดีมัลติเพลก์ วงจรรีจิสเตอร์ และวงจรนับ แต่ทั้งหมดก็เป็นการศึกษาแบบแยกส่วนแยกเป็นวงจรย่อยๆ ยังไม่ได้นำมาประกอบร่วมกัน วัตถุประสงค์ของบทนี้เพื่อแสดงให้เห็นถึงการนำวงจรย่อยๆแต่ละส่วน มาประกอบเข้าด้วยกันให้เป็นเครื่องมือหรือวงจรขนาดใหญ่ที่มีความซับซ้อนมากขึ้น ทำให้เกิดความเข้าใจในภาพรวมของการออกแบบเป็นระบบ

ตัวอย่างที่ยกมาเป็นกรณีศึกษาเป็น นาฬิกาจับเวลา ซึ่งสร้างบน FPGA เบอร์ XC3S200-5TQ144 โดยใช้ ซอฟท์แวร์ทูล Xilinx – ISE 9.2i แต่ละตัวอย่างมีรายละเอียดต่อไปนี้

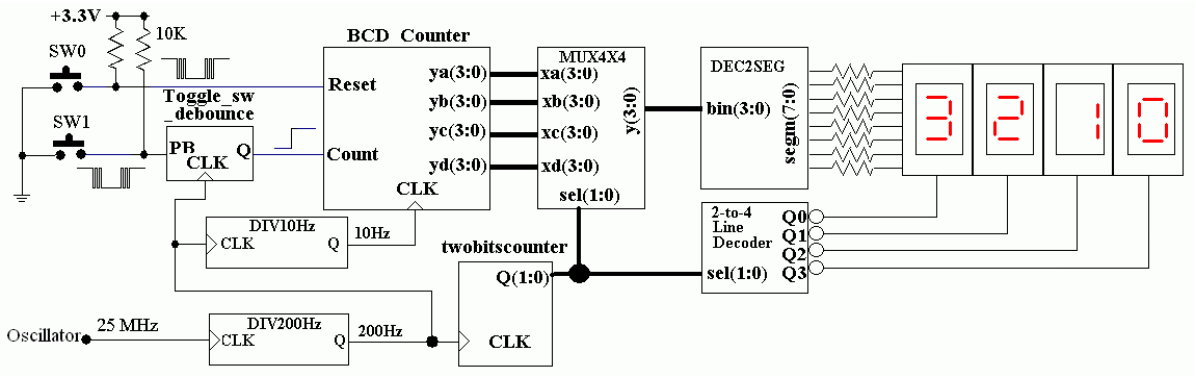
8.2 นาฬิกาจับเวลา

โครงสร้างของนาฬิกาจับเวลามีลักษณะตามรูปที่ 8.1 ประกอบด้วยแอลอีดี 7 ส่วนสำหรับแสดงค่าเวลามีหน่วยเป็น วินาที แสดงค่าได้ตั้งต่ 000.0 วินาที จนถึง 999.9 วินาที ในตัวอย่างนี้ยังไม่ได้ปิดแอลอีดีหลักสูงในกรณีที่เป็นเลข 0 ยังคงให้แสดงเป็นเลข 0 อยู่ และยังไม่ได้แสดงจุดทศนิยม เพราะว่าต้องการนำวงจรที่ออกแบบไว้มาใช้ประกอบ สวิตช์ SW0 เป็นสวิทช์รีเซต ให้เวลาเป็น 0000 ส่วนสวิตช์ SW1 เป็นสวิตช์สั่งงานให้เริ่มและหยุดการจับเวลา สัญญาณนาฬิกาของระบบได้จากออสซิลเลเตอร์ความถี่ 25 MHz



รูปที่ 8-1 ไดอะแกรมของนาฬิกาจับเวลา

ภายในระบบประกอบด้วย วงจรหารความถี่ DIV200Hz และ DIV10Hz วงจรนับเลขฐานสิบ BCD Counter วงจนับเลขฐานสองขนาด 2 บิต Twobitscounter วงจรมัลติเพลกซ์ MUX4X4 วงจรแปลงรหัสไบนารีเป็นแอลอีดี 7 ส่วน DEC2SEG และวงจรถอดรหัส 2-to-4 Line Decoder แต่ละวงจรการทำงานและรายละเอียดของ VHDL ดังนี้



รูปที่ 8-2 บล็อกไดอะแกรมของนาฬิกาจับเวลา

วงจรหารความถี่

ประกอบด้วยวงจร DIV200Hz และ DIV10Hz สำหรับ DIV200Hz ทำหน้าที่หารความถี่ของระบบ 25MHz ให้เป็นความถี่ 200 Hz เพื่อใช้ควบคุมการแสดงผลของแอลอีดี 7 ส่วน สำหรับ DIV10Hz ทำหน้าที่หารความถี่ 200Hz ให้เหลือความถี่ 10 Hz เพื่อใช้เป็นฐานเวลาของการนับ ดังนั้นถ้าต้องการเปลี่ยนหน่วยเวลาให้เปลี่ยนค่าการหารของวงจรนี้ การออกแบบด้วย VHDL ของทั้งสองวงจรเขียนเหมือนกันเพียงแต่กำหนดค่าความถี่ขาเข้ากับค่าความถี่ขาออกในส่วน generic (fin: integer := ....; fout: integer := .....); เท่านั้นก็จะได้วงจรตามต้องการ

สัญญาณ CLK เป็นสัญญาณนาฬิกาอินพุท
Q เป็นสัญญาณนาฬิกาเอาท์พุท

โมเดล VHDL ของ DIVIDER200Hz

```
library IEEE;
use IEEE.std_logic_1164.all;
entity DIVIDER200Hz is
  generic (fin: integer :=25000000;      --Input frequency
        fout: integer :=200);            --Output frequency
  port (CLK: in std_logic;
      Q : out std_logic );
end DIVIDER200Hz;

architecture RTL of DIVIDERHz is

  signal COUNT : integer range 0 to (fin/(2*fout)) ;
  signal qs : std_logic := '0';
  begin
  process (CLK)

    begin
      if CLK'event and CLK = '1' then
        if (COUNT >= (fin/(2*fout)-1)) then
          COUNT <= 0;
          qs <= not(qs);
        else
          COUNT <= COUNT +1;
        end if;
      end if;
  end process;
  Q <= qs ;
end RTL;
```

โมเดล VHDL ของ DIVIDER10Hz

```
library IEEE;
use IEEE.std_logic_1164.all;
entity DIVIDER10Hz is
  generic (fin: integer := 200;              --Input frequency
        fout: integer := 10);   --Output frequency
  port (CLK: in std_logic;
      Q : out std_logic );
end DIVIDER10Hz;

architecture RTL of DIVIDER10Hz is

  signal COUNT : integer range 0 to (fin/(2*fout)) ;
  signal qs : std_logic := '0';
  begin
  process (CLK)

    begin
      if CLK'event and CLK = '1' then
        if (COUNT >= (fin/(2*fout)-1)) then
          COUNT <= 0;
          qs <= not(qs);
        else
          COUNT <= COUNT +1;
        end if;
      end if;
  end process;
  Q <= qs ;
end RTL;
```

วงจรนับเลขฐานสองขนาด 2 บิต Twobitscounter

ทำหน้าที่สร้างสัญญาณควบคุมการแสดงผล โดยรับสัญญาณนาฬิกาความถี่ 200 Hz มาจาก DIV200Hz นำมาสร้างสัญญาณไบนารีขนาด 2 บิต ส่งไปให้วงจรถอดรหัสและวงจรมัลติเพลกซ์

สัญญาณ CLK เป็นสัญญาณนาฬิกาอินพุท
Q(1:0) เป็นสัญญาณไบนารีเอาท์พุท

โมเดล VHDL ของ twobitscounter

```
library IEEE;
use IEEE.std_logic_1164.all;
entity twobitscounter is
  port (CLK: in std_logic;
      Q : out integer range 0 to 3);
end twobitscounter;

architecture RTL of twobitscounter is
  signal COUNT : integer range 0 to 3;
  begin
  process (CLK)
    begin
      if CLK'event and CLK = '1' then
        if (COUNT >= 3) then
          COUNT <= 0;
        else
```

```
            COUNT <= COUNT +1;
          end if;
        end if;
    end process;
    Q <= count ;
end RTL;
```

วงจรแก้เบาสน์และสร้างสัญญาณทอกเกิล Toggle_sw_debounce

ทำหน้าที่สร้างสัญญาณควบคุมการนับและหยุดนับของวงจรนับเลขฐานสิบ แต่เนื่องจากอุปกรณ์ที่ใช้ป้อนสัญญาณเป็นสวิตช์แบบกดติดปล่อยดับ เมื่อกดสวิทช์ 1 ครั้งจะเกิดสัญญาณเป็นพลัส 0 ออกมาหลายพลัส อันเป็นสาเหตุมาจากการสัมผัสกันของหน้าสัมผัสที่เป็นกลไกภายในสวิตช์ อาก ารนี้เรียกว่าเบาสน์ (Bounce) ดังนั้นวงจรต้องมีการแก้ไขบัญหานี้เพื่อให้การกดสวิตช์ 1 ครั้งได้พลัสเพียง 1 ลูก

เนื่องจากการควบคุมการนับของวงจรนับเลขฐานสิบต้องการเป็นโลจิก คือถ้าเป็นโลจิก 1 ให้นับต่อ แต่ถ้าเป็นโลจิก 0 ให้หยุดนับ ดังนั้นเมื่อมีสัญญาณพลัสจากการกดสวิตช์แต่ละครั้ง ต้องทำการเปลี่ยนโลจิกของสัญญาณเอาท์พุทให้สลับกันไปทุกครั้ง เพื่อเป็นการสั่งให้วงจรนับและหยุดนับ

สัญญาณ CLK เป็นสัญญาณนาฬิกาอินพุทใช้ความถี่ 200 Hz
PR เป็นสัญญาณนาฬิกาอินพุท ใช้ต่อกับสวิตช์
Q เป็นสัญญาณเอาท์พุท

โมเดล VHDL ของ Toggle_sw_debounce

```
LIBRARY IEEE;
USE  IEEE.STD_LOGIC_1164.all;
USE  IEEE.STD_LOGIC_ARITH.all;
USE  IEEE.STD_LOGIC_UNSIGNED.all;

ENTITY Toggle_sw_debounce IS
        PORT(pb, clock_200Hz : IN   STD_LOGIC;
                Q                   : OUT     STD_LOGIC);
END Toggle_sw_debounce;

ARCHITECTURE Behavioral OF Toggle_sw_debounce IS
        SIGNAL SHIFT_PB         : STD_LOGIC_VECTOR(3 DOWNTO 0);
        signal state            : STD_LOGIC := '0';
        signal qs               : STD_LOGIC := '0';
BEGIN

        -- Debounce clock should be approximately 10ms or 100Hz
        PROCESS
        BEGIN
                WAIT UNTIL (clock_200Hz'EVENT) AND (clock_200Hz = '1');
                -- Use a shift register to filter switch contact bounce
                SHIFT_PB(2 DOWNTO 0) <= SHIFT_PB(3 DOWNTO 1);
                SHIFT_PB(3) <= PB;
                case shift_pb is
                        when "0000" =>  state <= '0';
                        when "1111" =>  state <= '1';
                        when others =>   state <= state;
                end case;
        END PROCESS;

        PROCESS(state)
        BEGIN
```

```
                if state'event and state = '1' then
                        qs <= not(qs);
                else
                        qs <= qs;
                end if;
                Q <= qs;
        END PROCESS;

END Behavioral;
```

วงจรมัลติเพลกซ์ MUX4X4

ทำหน้าที่เลือกสัญญาณบีซีดีแต่ละหลักของวงจรนับเลขฐานสิบส่งออกแสดงผลที่แอลอีดี 7 ส่วนโดยใช้สัญญาณจาก **Twobitscounter เป็นสัญญาณเลือก**

สัญญาณ xa xb xc และ xd เป็นสัญญาณบีซีดีอินพุท
sel เป็นสัญญาณควบคุมการเลือก
y เป็นสัญญาณบีซีดีเอาท์พุท

โมเดล VHDL ของ MUX4X4

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity mux4x4 is
   Port (xa, xb,xc ,xd : in integer range 0 to 15;
        sel : in integer range 0 to 3;
        y : out integer range 0 to 15);
end mux4x4;

architecture Behavioral of mux4x4 is
begin
with sel select
  y <= xa when 0,
        xb when 1,
        xc when 2,
        xd when others;
end Behavioral;
```

วงจรแปลงรหัสไบนารีเป็นแอลอีดี 7 ส่วน DEC2SEG

ทำหน้าที่แปลงรหัสบีซีดีเป็นรหัสสำหรับแอลอีดี 7 ส่วน

สัญญาณ bin เป็นสัญญาณบีซีดีอินพุท
segm เป็นสัญญาณเอาท์พุท

โมเดล VHDL ของ DEC2SEG

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity dec2seg is
  port (bin : in integer range 0 to 15 ;
        segm : out std_logic_vector(7 downto 0));
end dec2seg;
```

```
architecture Behavioral of dec2seg is
begin
        with bin select
                segm <= "11111100" when 0,
                        "01100000" when 1,
                        "11011010" when 2,
                        "11110010" when 3,
                        "01100110" when 4,
                        "10110110" when 5,
                        "10111110" when 6,
                        "11100000" when 7,
                        "11111110" when 8,
                        "11110110" when 9,
                        "00000000" when others;   --Blank
end Behavioral;
```

วงจรนับเลขฐานสิบ BCD Counter และวงจรถอดรหัส 2-to-4 Line Decoder

วงจรนับเลขฐานสิบ ใช้นับฐานเวลาความถี่ 10 Hz เพื่อแสดงค่าเวลาที่นับได้ ส่วนวงจรถอดรหัส 2-to-4 Line Decoder ทำหน้าที่เลือกหลักของแอลอีดี 7 ส่วนที่ต้องการให้แสดง โดยใช้สัญญาณจาก Twobitscounter เป็นสัญญาณเลือกเหมือนกับวงจรมัลติเพลกซ์ ดังนั้น นตัวเลขที่นำออกแสดงจะสอดคล้องกับหลักของแอลอีดีที่ติด วงจรทั้งสองนี้เขียนอยู่ภายในโมดูลหลักของระบบ

โมเดล VHDL ของวงจรจับเวลา

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity timer is
    Port (clk, reset, count : in std_logic;
            seg : out std_logic_vector(7 downto 0);
            dig : out std_logic_vector(3 downto 0));
end timer;

architecture Behavioral of timer is

component dec2seg is
  port (bin : in integer range 0 to 15 ;
       segm : out std_logic_vector(7 downto 0));
end component;

component twobitscounter is
   port (CLK: in std_logic;
         Q : out integer range 0 to 3);
end component;

component mux4x4 is
    Port (xa, xb,xc ,xd : in integer range 0 to 15;
            sel : in integer range 0 to 3;
            y : out integer range 0 to 15);
end component;
```

```
component DIVIDER10Hz is
  port (CLK: in std_logic;
        Q : out std_logic );
end component;

component DIVIDER200Hz is
  port (CLK: in std_logic;
        Q : out std_logic );
end component;

component Toggle_sw_debounce IS
  PORT(pb, clock_200Hz : IN     STD_LOGIC;
       Q               : OUT STD_LOGIC);
end component;

signal ya, yb,yc, yd, bus4 : INTEGER range 0 to 15;
signal ssel : integer range 0 to 3;
signal clk1, clk2, count_s : std_logic;

begin

process (CLK1, reset, count_s)                    -- BCD Counter
begin
  if Reset='0' then
    ya <= 0;
    yb <= 0;
    yc <= 0;
    yd <= 0;
  else
      if count_s = '1' then
          if CLK1='1' and CLK1'event then
              if ya >= 9 then
                  ya <= 0;
                  if yb >= 9 then
                      yb <= 0;
                      if yc >= 9 then
                          yc <= 0;
                          if yd >= 9 then
                              yd <= 0;
                          else
                              yd <= yd + 1;
                          end if;
                      else
                          yc <= yc + 1;
                      end if;
                  else
                      yb <= yb + 1;
                  end if;
              else
                  ya <= ya + 1;
              end if;
          else
              ya <= ya;
              yb <= yb;
              yc <= yc;
              yd <= yd;
          end if;
      end if;
  end if;
```

```
end process;

with ssel select                                        --2 to 4 line decoder
  dig <= "1110" when 0,
         "1101" when 1,

         "1011" when 2,
         "0111" when others;

   c1: mux4x4 port map(ya, yb, yc, yd, ssel, bus4);
   c2: dec2seg port map(bus4, seg);
   c3: twobitscounter port map(clk2, ssel);
   c4: DIVIDER200Hz port map(clk,clk2);
   c5: DIVIDER10Hz port map(clk2, clk1);
   c6: Toggle_sw_debounce port map(count, clk2,count_s);
end Behavioral;
```