VENDING MACHINE

010123107 Logic Design of Digital Systems 1/2559

Mini project

นายสาริกข์ คำปาน 5801012610164

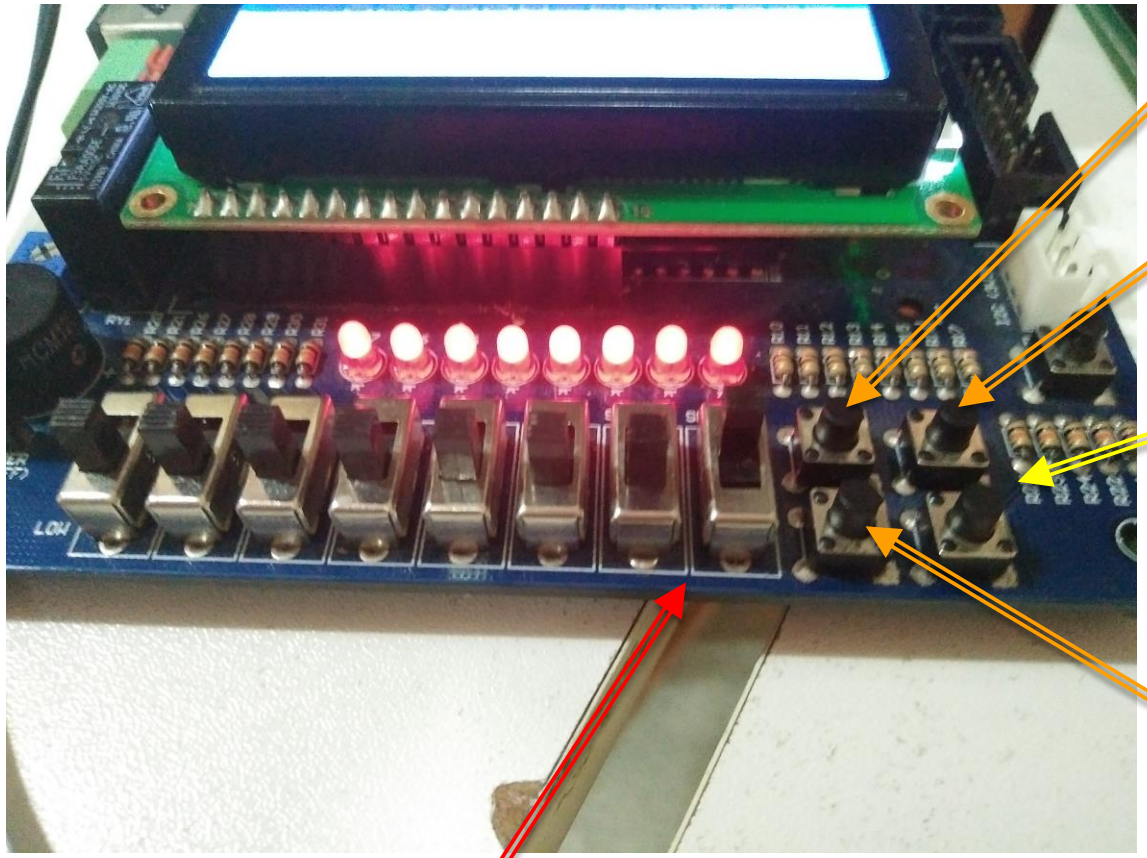
นายชัยสิทธิ์ แซ่ตัน 5801012630050

ความสามารถของโค้ด VHDL ที่ออกแบบมา

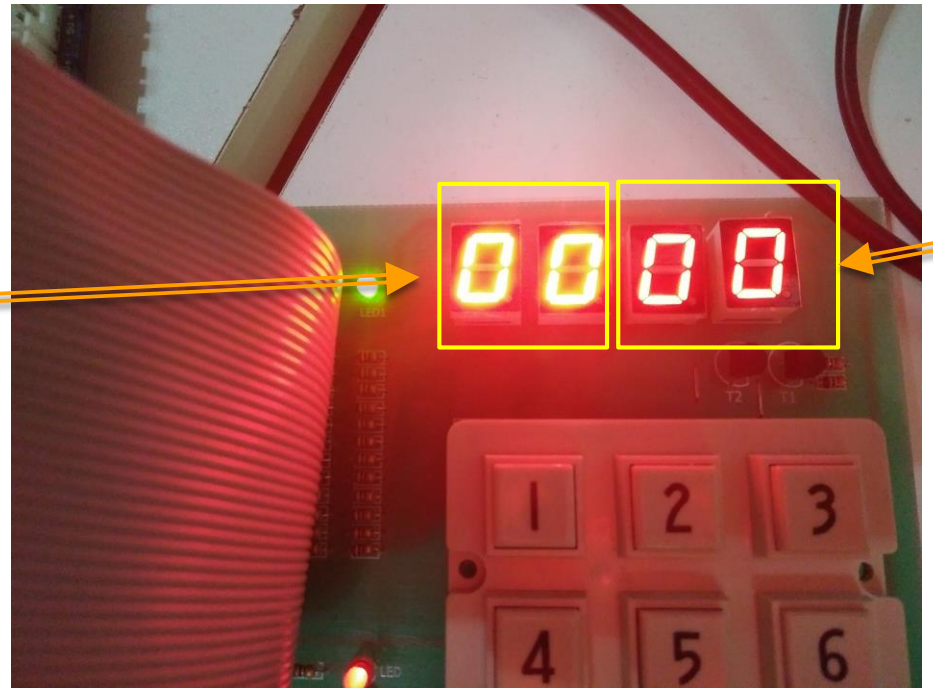
- สถานะเริ่มต้นการทำงาน เงินที่ผู้ซื้อต้องจ่ายและเงินทอน เป็น 0
- จากสถานะเริ่มต้น ไม่สามารถกดปุ่ม จ่ายเงินใดๆได้
- จากสถานะเริ่มต้น หากกดปุ่ม buy 1 ครั้ง หลังปล่อย จะมีการแสดงราคาสินค้า ตาม slide switch ที่ผู้ซื้อเลือกอยู่ ณ ขณะที่กดปุ่ม และเข้าสู่ สภาวะเลือกสินค้าอยู่
- หาก อยู่ในสภาวะ เลือกสินค้าอยู่แล้ว กดปุ่ม buy อีก ครั้ง จะทำให้ มีเสียง buzzer ดังยาวต่อเนื่องประมาณ 1 วินาที และ ถ้าหากมีการหยอดเหรียญไปก่อนหน้านี้ ระบบจะแสดงเงินทอนให้ จากนั้น ระบบ reset จะกลับเข้าสู่สภาวะเริ่มต้น
- หากอยู่ในสภาวะเลือกสินค้าแล้ว สามารถหยอดเหรียญ ได้ตามปกติ ซึ่งหากชำระเงินครบ หรือ เกินแล้ว จะมีเสียง buzzer ดังเป็นจังหวะประมาณ 1 วินาที และหากชำระเงินเกิน จะมีการแสดง เงินทอนด้วย หลังจากนั้นระบบจะเข้าสู่ สภาวะเริ่มต้น
- การเลือกสินค้า ใช้ slide switch 2 ตัว ซึ่งทำให้สามารถ เลือกสินค้าได้ 4 ชนิด

Input จาก ผู้ ซื้อ

Slide Switch	ราคา
00	45
01	38
10	18
11	9

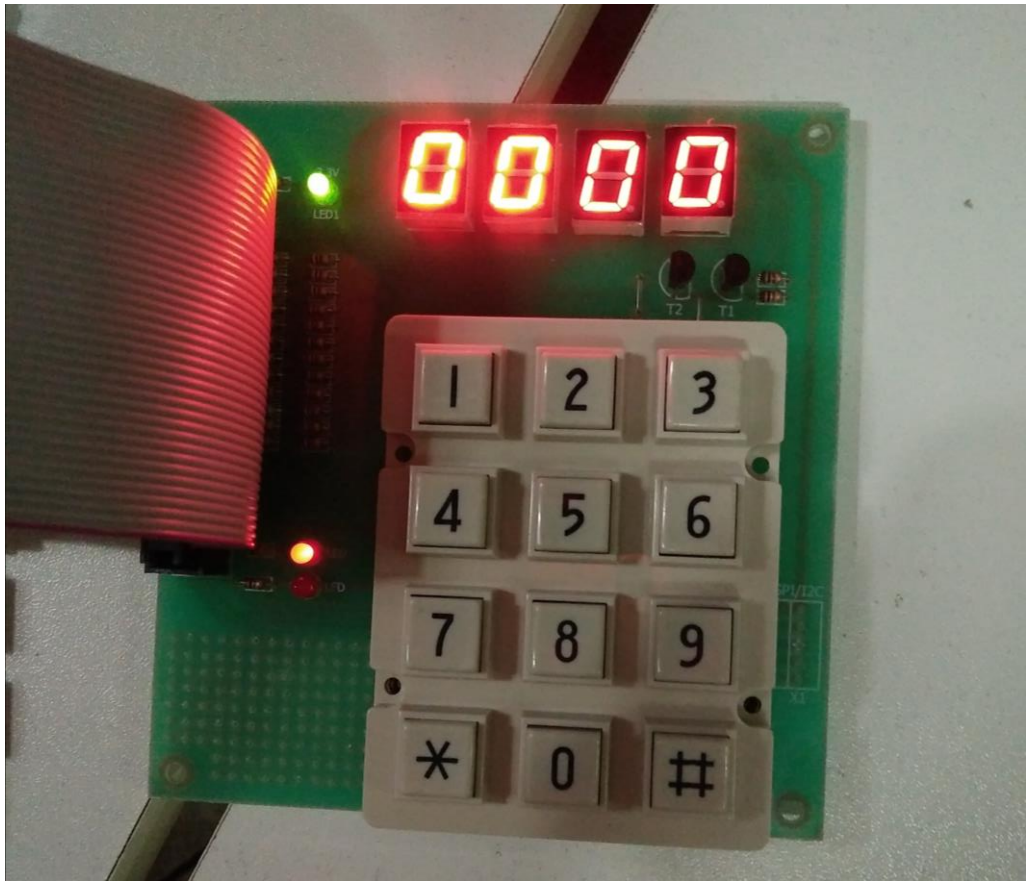


Output – 7 segment



สภาวะเริ่มต้น – อัพโหลด โค้ดลงบอร์ดครั้งแรก, กดreset, จ่ายเงินครบ

เป็น สภาวะ ที่ เกิดในขณะที่ ได้ทำการอัพโหลดโค้ดลงไปครั้งแรก หรือ หลังจากการ ยกเลิกการซื้อสินค้า หรือ เกิดหลังจากการจ่ายเงินครบแล้ว ในสภาวะนี้ จะไม่สามารถ หยอดเหรียญ ใดๆ ได้ ทำได้เพียง ปรับราคาสินค้า ซึ่งการปรับราคาสินค้า จะใช้ slide switch ในการปรับ จากนั้น กดปุ่ม buy เพื่อ เลือกสินค้านั้นๆ



สภาวะ เลือกสินค้าอยู่ – กด buy 1 ครั้งจาก สภาวะเริ่มต้น

หลังจาก กด ปุ่ม buy 1 ครั้งจาก สภาวะเริ่มต้น ระบบจะแสดงราคาสินค้าที่ ผู้ซื้อเลือกอยู่ใน ขณะที่กด ปุ่ม buy แสดงขึ้นมา ทาง 7 segment ด้านขวามือ ในรูปด้านขมือนี้ คือ ราคา 38 บาท ในสภาวะนี้สามารถกด จ่ายเงินได้ตามปกติ หรือจะกด buy อีก ครั้งเพื่อยกเลิกการเลือกซื้อสินค้าได้

การกดยกเลิก – กรณี หยอดเงินไปแล้ว 2 บาท

หลังจากเลือกสินค้าแล้ว ผู้ซื้อ หยอดเหรียญ ลงไปแล้ว 2 บาท จากนั้น ผู้ซื้อ ได้ยกเลิกการซื้อสินค้า ระบบ จะ แสดง เงินทอน 2 บาท ตามที่หยอดไป และ มีเสียง buzzer ดังต่อเนื่องประมาณ 1 วินาที

การกดยกเลิก – กรณีไม่ได้หยอดเงิน

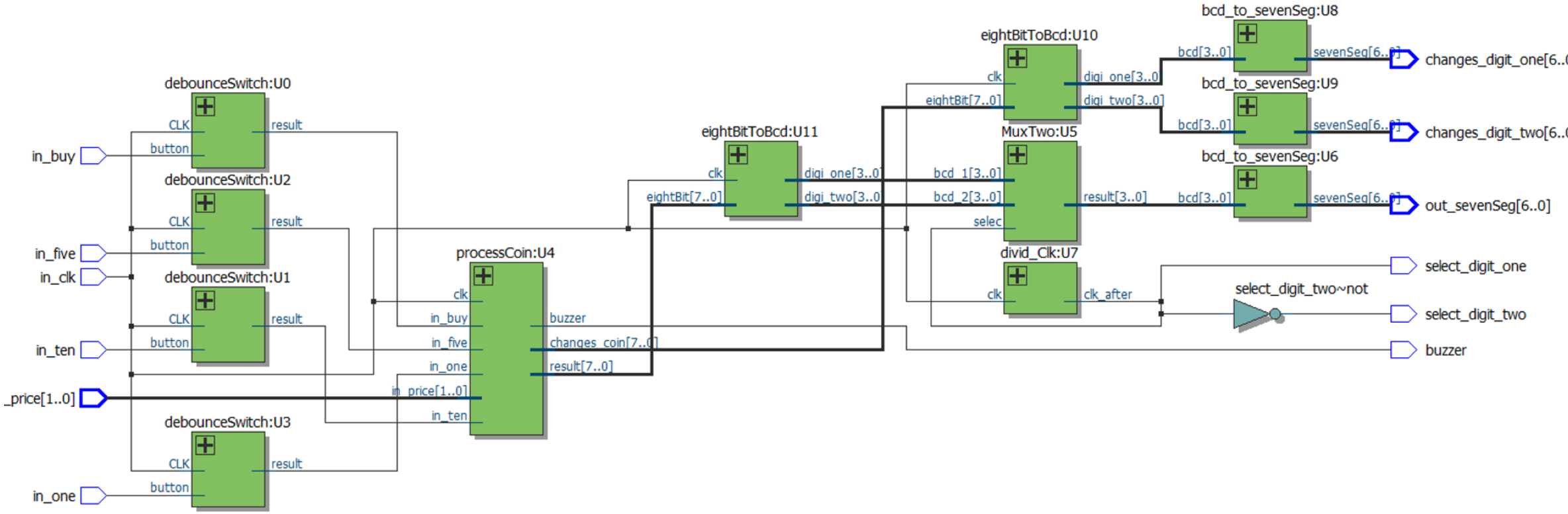
หลังจากเลือกสินค้า ผู้ซื้อ ได้ยกเลิกการซื้อสินค้า ระบบ จะ มีเสียง buzzer ดังต่อเนื่องประมาณ 1 วินาที และ ไม่มีการทอนเงินใดๆ เนื่องจาก ผู้ซื้อ ยังไม่ได้ หยอดเหรียญ ใดๆลงไป

การจ่าย เงินครบ หรือ จ่าย เงิน เกิน ราคาสินค้า

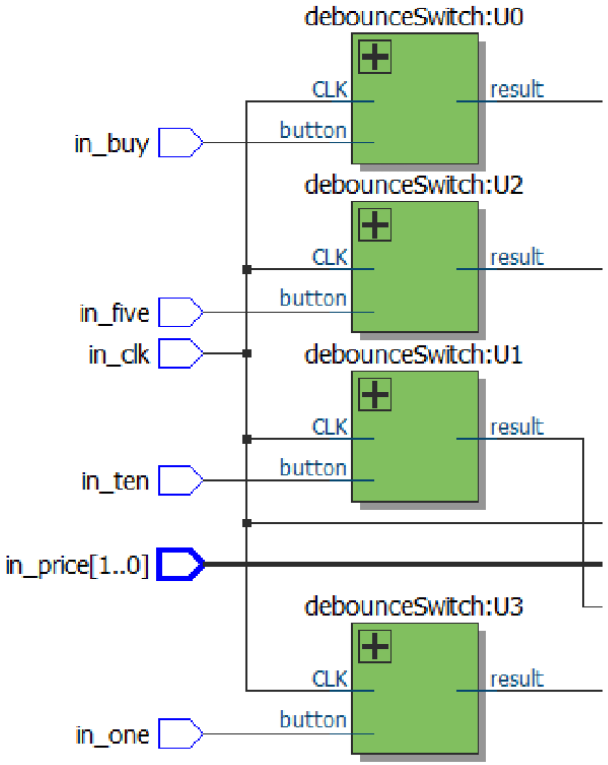
เมื่อผู้ซื้อเลือกสินค้าและได้ชำระเงิน ครบตามราคาสินค้าแล้ว ระบบจะมีเสียง buzzer ดังเป็น จังหวะ ประมาณ 1 วินาที และ แสดง เงินทอน ตามที่จ่ายเกินไป หรือไม่แสดง ถ้าหากว่าจ่ายครบพอดี ในที่นี้ คือ จ่ายเงินเกินไป 5 บาท จึง แสดง เลข 5 ออกมาทาง 7 segment ทาง ซ้ายมือ พอเสียง buzzer ดับ ก็จะเข้าสู่ สภาวะเริ่มต้น

โค้ด VHDL

ภาพ การประกอบ component



มี Debounce switch เพื่อตัดสัญญาณ รบกวนที่เกิดจากการสั่นของหน้าสัมผัส ของ switch

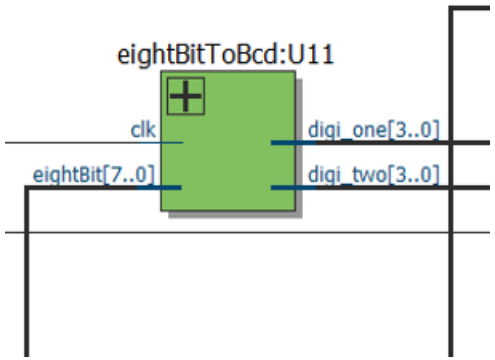


หลักการการทำงานคร่าวๆ คือ เมื่อ input เปลี่ยนแปลง จากค่าเดิม แล้วนับ ตามจำนวน clk ได้ตามที่ กำหนดไว้ ก็ อัพเดท ค่านั้น ให้ output แต่ถ้า ค่าที่เปลี่ยนแปลงนั้น ไม่คงที่มีการเปลี่ยนไปเปลี่ยนมา ก็ จะให้ reset การนับ ไปเรื่อยๆ

* ใช้ not keepResult เพราะต้องการ ให้ จากที่กดปุ่มแล้วได้ Low ได้ เป็น hight แทน เพื่อ สดวกในการเขียนโค้ดส่วนอื่น

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity debounceSwitch is
6    port(CLK: in std_logic;
7         button: in std_logic;
8         result: out std_logic);
9  end debounceSwitch;
10
11 architecture behave of debounceSwitch is
12   signal inff : std_logic_vector(1 downto 0);
13   constant cnt_max : integer:=30000;
14   signal count: integer range 0 to cnt_max:=0;
15   signal keepResult: std_logic :='1';
16
17 begin
18   result <= not keepResult;
19   process(CLK)
20   begin
21     if(rising_edge(CLK)) then
22       inff <= inff(0) & button;
23         if(inff(0)/=inff(1))then
24           count <= 0;
25         elsif(count<cnt_max) then
26           count <= count +1;
27         else
28           keepResult <= inff(1);
29         end if;
30     end if;
31   end process;
32 end behave;
```

ส่วน ที่ใช้แปลง เลข ให้อยู่ใน รูป หลักหน่วยและหลักสิบ



การทำงานคือ รับ input 8 bit มา จากนั้นแปลง เป็น integer แล้วนำ ค่านั้น ไป หาร 10 และ มอด 10 เพื่อ หา หลักหน่วยและหลัก สิบ ซึ่งการหาร 10 จะได้หลักสิบ การ มอด 10 จะได้หลักหน่วย แล้วค่อยแปลง ค่านั้นกลับเป็น 4 bit

```
5  entity eightBitToBcd is
6      port(     clk:in std_logic;
7           eightBit:in std_logic_vector(7 downto 0);
8           digi_one:out std_logic_vector(3 downto 0);
9           digi_two:out std_logic_vector(3 downto 0));
10 end eightBitToBcd;
11
12 architecture Behavioral of eightBitToBcd is
13     signal value_input: integer :=to_integer(signed(eightBit));
14     signal value_digit_one: integer :=value_input/10;
15     signal value_digit_two: integer :=value_input mod 10;
16 begin
17     process(clk)
18     begin
19         if(rising_edge(clk))then
20             digi_one <= std_logic_vector(to_signed(value_digit_one, 4));
21             digi_two <= std_logic_vector(to_signed(value_digit_two, 4));
22         end if;
23    end process;
24 end Behavioral;
```

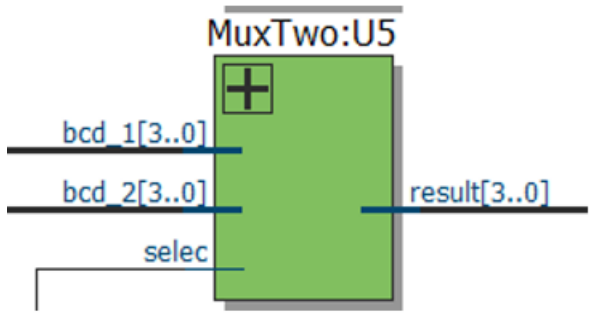
ส่วน แปลง BCD ให้ แสดงผลตัวเลข บท 7 segment

```
5  entity bcd_to_sevenSeg is
6    port(     bcd: in std_logic_vector (3 downto 0);
7         sevenSeg: out std_logic_vector (6 downto 0));
8  end bcd_to_sevenSeg;
9
10 architecture Structural of bcd_to_sevenSeg is
11 begin
12   process (bcd)
13     begin
14         case bcd is
15             when "0000" => sevenSeg <= "1000000";
16             when "0001" => sevenSeg <= "1111001";
17             when "0010" => sevenSeg <= "0100100";
18             when "0011" => sevenSeg <= "0110000";
19             when "0100" => sevenSeg <= "0011001";
20             when "0101" => sevenSeg <= "0010010";
21             when "0110" => sevenSeg <= "0000010";
22             when "0111" => sevenSeg <= "1111000";
23             when "1000" => sevenSeg <= "0000000";
24             when "1001" => sevenSeg <= "0011000";
25             when others => sevenSeg <= "0110000";
26          end case;
27     end process;
28 end Structural;
29
```

ใช้การเทียบ input ธรรมดา ซึ่ง output เรียงจาก G . . . A และ 0 หมายถึง ไฟติด

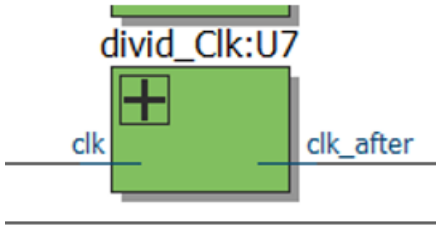
ส่วนเลือก การแสดงผล ให้กับ 7 segment ที่ใช้ input ร่วม

```
 5  entity MuxTwo is
 6    port(bcd_1, bcd_2: in std_logic_vector(3 downto 0);
 7         selec: in std_logic;
 8         result: out std_logic_vector(3 downto 0));
 9  end MuxTwo;
10
11  architecture behave of MuxTwo is
12  begin
13    result <= bcd_1 when selec = '0' else bcd_2;
14  end behave;
15
```



ใช้ สถานะ ของ selec เป็นตัวกำหนดว่าจะใช้ input ตัวไหนเป็น output

ส่วนหาร สัญญาณ clk

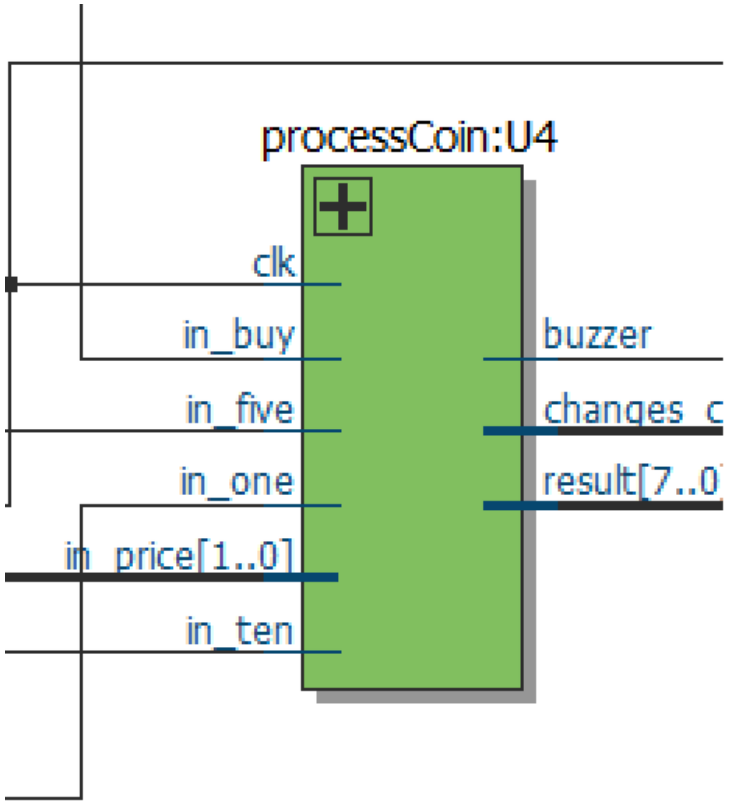


การทำงานคือ นับ clk ไปเรื่อยๆ จนถึงค่าที่กำหนด ไว้ ก็ จะสลับสถานะ ของ output จาก 1 เป็น0 จาก 0 เป็น1 ซึ่ง การที่จะได้ 1 คาบ นั้น จะต้องนับ ถึงค่าที่กำหนด 2 รอบ โดยเริ่มจาก 0 ซึ่ง จากโค้ด กำหนดไว้ 499999 ซึ่งต้องนับ 5 แสน ครั้ง 2 รอบ ก็ 1 ล้าน ซึ่ง การนับ 1 ล้าน clk จะได้ output 1 คาบ ก็คือการหารสัญญาณ จาก 50 Mhz ไปเป็น 50 hz ไปให้กับการเลือก output ของ mux กับ เปิดปิด digit 1 , 2 ของ ส่วนแสดง เงินที่ต้องจ่าย

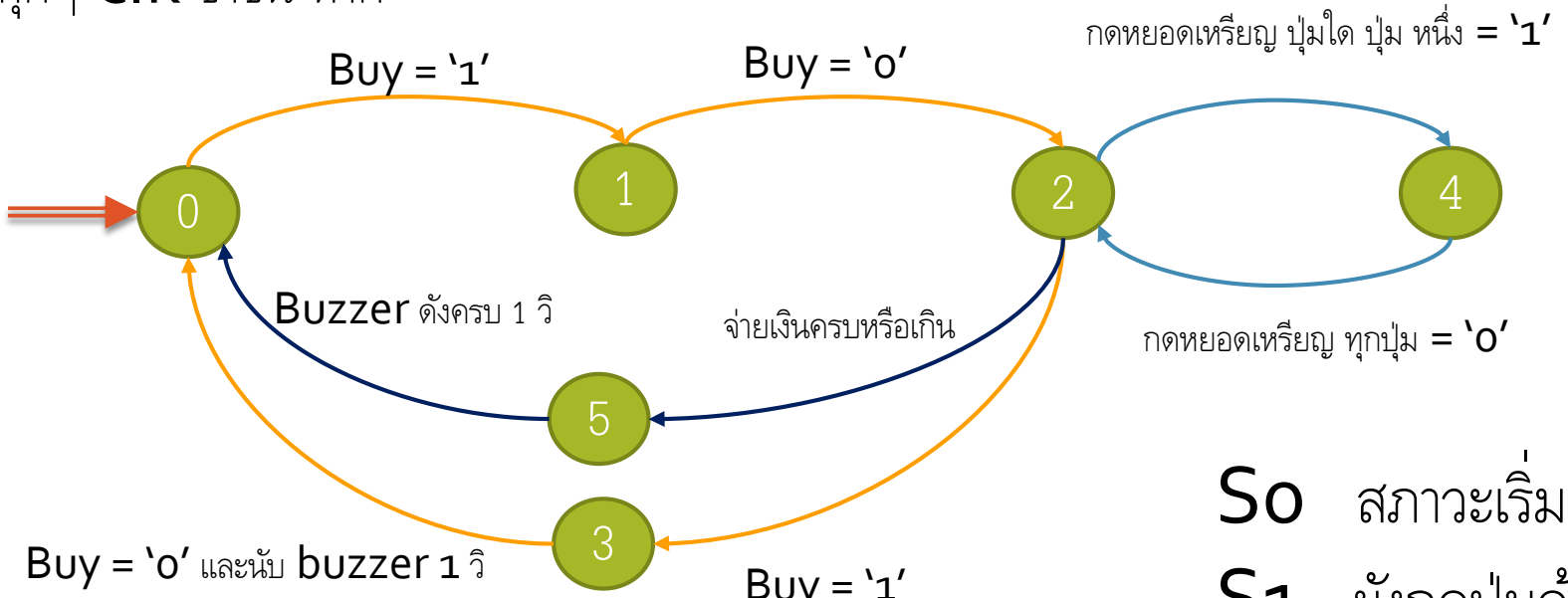
```
 5  entity divid_Clk is
 6    port(      clk: in std_logic;
 7         clk_after: out std_logic);
 8  end divid_Clk;
 9
10  architecture Structural of divid_Clk is
11      signal max_count: integer := 499999;
12      signal counter: integer range 0 to max_count := 0;
13      signal result: std_logic :=clk;
14  begin
15    clk_after <= result;
16    process (clk)
17      begin
18        if(rising_edge(clk))then
19            if(counter < max_count)then
20                counter <= counter + 1;
21            else
22                if(result = '0')then
23                    result <= '1';
24                else
25                    result <='0';
26                end if;
27                counter <= 0;
28            end if;
29        end if;
30      end process;
31  end Structural;
```

ส่วนคำนวน

มี 6 state



ทุกๆ clk ขาขึ้น หาก



s0

s2

S0 สภาวะเริ่มต้น

S1 ยังกดปุ่มค้างไว้อยู่

S2 สภาวะ รอชำระสินค้าหรือเลือกสินค้าอยู่

S3 กดยกเลิกสินค้า

S4 หยอดเหรียญ แล้วยังไม่ปล่อยปุ่ม

S5 จ่ายเงินครบหรือเกิน

S 0

```
when s0 =>
   buzzer <= '0';
   keep_re <= "00000000";
   changes <= "00000000";
   count_delay_changes <= 0;
   case in_price is
         when "00" => store_value <= 45;
         when "01" => store_value <= 38;
         when "10" => store_value <= 18;
         when "11" => store_value <= 9;
   end case;
   price_value <= store_value;
   if(in_buy = '1')then
      state <= s1;
   end if;
```

ตั้งค่าทุกอย่าง เป็นค่าเริ่มต้น และอ่านราคา ใหม่ ทุกๆ ขอบขาขึ้นของ clk บันทึกค่า ราคาที่แท้จริง กับราคาที่ผู้ซื้อต้องจ่าย เอาไว้ แต่ยังไม่อัพเดทให้แสดงออกมา หาก ปุ่ม buy มีสถานะเป็น 1 จะเข้า S 1

S1

```
when s1 =>
    if(in_buy = '0')then
        state <= s2;
    end if;
```

เข้า S 2 เมื่อ ปุ่ม buy ไม่ได้ถูกกดอยู่ เป็น state ที่ใช้กันการกดค้าง

S3

```
when s3 =>
    keep_re <= "00000000";
    if(count_delay_changes < max_delay)then
        buzzer <= '1';
        if(store_value < 0)then
            changes <= std_logic_vector(to_signed(store_value*(-1), 8));
        else
            count_delay_changes <= max_delay;
        end if;
        count_delay_changes <= count_delay_changes + 1;
    elsif(in_buy /= '0')then
        count_delay_changes <= 0;
    else
        buzzer <= '0';
        state <= s0;
    end if;
```

เป็น state ที่ใช้เปิดเสียง buzzer 1 วินาที โดยใช้หลักการนับ และ ถ้าหากปุ่มยังกดอยู่ ก็จะนับไม่เรื่อยๆ จนกว่าจะปล่อย และหากปล่อยปุ่มแล้ว รวมถึงนับ ถึง 1 วิ แล้วก็จะเข้าสู่ s0 , มีส่วนที่ใช้ในการแสดง เงินทอนด้วย

S2

เป็นส่วน สถานะเลือกสินค้า พร้อมชำระเงิน ซึ่ง ถ้าหากปุ่ม buy ถูกกดอยู่ จะคำนวณหาผลต่างว่าผู้ซื้อจ่ายเงินมาเท่าไหร จะให้กลายเป็นเงินทอน และ เข้า s3

กดปุ่มจ่ายเงินใดๆจะเข้า s 4 และเงินที่ต้องจ่ายจะลดลง

เมื่อจ่ายเงินครบ หรือเกิน จะเข้า s5

```
when s2 =>
   if(in_buy = '1')then
      if((price_value - store_value) > 0)then
         store_value <= store_value - price_value;
      end if;
      state <= s3;
   end if;
   if(in_ten = '1')then
      store_value <= store_value -10;
      state <= s4;
   elsif(in_five = '1')then
      store_value <= store_value -5;
      state <= s4;
   elsif(in_one ='1')then
      store_value <= store_value -1;
      state <= s4;
   end if;
   if(store_value < 1)then
      changes <= std_logic_vector(to_signed(store_value*(-1), 8));
      keep_re <= "00000000";
      state <= s5;
   else
      keep_re <= std_logic_vector(to_signed(store_value, 8));
      changes <= "00000000";
   end if;
```

S4

```
when s4 =>
   if(in_ten ='0' and in_five ='0' and in_one ='0')then
      state <= s2;
   end if;
```

ใช้ในการกันการกดหยอดเหรียญค้าง การปล่อยปุ่มหยอดเหรียญ ทั้งหมด จะเข้า สู่ S2

S5

```
when s5 =>
   if(count_delay_changes < max_delay)then
      buzzer <= '1';
      if(count_delay_buzzer > max_delay/10)then
         buzzer <= '0';
         count_delay_buzzer <= count_delay_buzzer +1;
         if(count_delay_buzzer > max_delay/5)then
            count_delay_buzzer <= 0;
         end if;
      else
         count_delay_buzzer <= count_delay_buzzer +1;
      end if;
      count_delay_changes <= count_delay_changes + 1;
    else
      count_delay_buzzer <= 0;
      state <= s0;
    end if;
end case;
```

จะมีการ นับ ให้ buzzer ทำงานเป็นจังหวะ ประมาณ 1 วิ ซึ่ง จังหวะ จะใช้ การหา อัตราส่วนของ max_delay ในการทำ หลังจาก ครบ 1 วิ จะเข้าสู่ S 0

END