



ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

ภาคการศึกษาที่ 1 ปีการศึกษา 2564

รหัสวิชา 010113026 ชื่อวิชา Digital Laboratory

ตอนเรียน 4 หมายเลขโต๊ะ..... -

รหัสนักศึกษา..... 6201011621168

ชื่อ-นามสกุล..... พงษ์โสภณ สุขสมบูรณ์

อาจารย์ผู้สอน..... CSP

เวลาที่ทำการทดลอง 13.00-16.00 วันที่ 23/09/64

การทดลองที่ 8 4-Digits Counters

วัตถุประสงค์

1. เพื่อให้สามารถใช้โปรแกรมคอมพิวเตอร์จำลองการทำงานของวงจรลอจิกเกตได้
2. เพื่อให้สามารถประยุกต์ใช้วงจรและอุปกรณ์ดิจิทัลเพื่อออกแบบระบบงานที่ซับซ้อนได้
3. เพื่อประยุกต์ใช้วงจรนับแบบ BCD Counter ร่วมกับวงจร 7-Segment ได้

อุปกรณ์

1. ระบบคอมพิวเตอร์ 1 เครื่อง พร้อมติดตั้งโปรแกรม Quartus II เวอร์ชัน 8.0 (Student Edition) ขึ้นไป
2. บอร์ดทดลอง Cyclone3-Lab01 1 บอร์ด
3. สาย J-TAG 1 เส้น ใช้รุ่น USB-Blaster (สำหรับเครื่อง Notebook) หรือรุ่น Byte-Blaster (สำหรับเครื่อง PC)
4. บอร์ดแสดงผล 7-segment (รุ่นแป้นพิมพ์ Keypad สีขาว)

การทดลอง การสร้างวงจรรับเลขฐานสิบ ขนาด 4 หลัก

ขั้นที่ 0 เตรียมการ ด้วยการนำเอาอุปกรณ์ที่เคยสร้างไว้ใน การทดลองที่ 5, 7 มาใช้ (ไม่ต้องเขียนขึ้นใหม่)

คำสั่งการทดลอง

1. ให้ น.ศ. สร้างโฟลเดอร์สำหรับเก็บงานขึ้นใหม่เพื่อเก็บงานในการทดลองนี้ชื่อ “Lab08_4DigitCounter”
2. นำวงจรที่เคยสร้างไว้ใน การทดลองที่ 5, 7 (ทำการ copy ไฟล์ดังรายชื่อด้านล่าง) มาไว้ในโฟลเดอร์นี้
 - จากการทดลองที่ 7 ไฟล์ mux2to1.qpf และ mux2to1.bdf
 - ไฟล์ AsynchronousCounter.qpf และ AsynchronousCounter.bdf
 - ไฟล์ ClockDivider.qpf และ ClockDivider.bdf
 - จากการทดลองที่ 5 ไฟล์ VHD_7SEGM.qpf และ VHD_7SEGM.vhd
3. ให้ทำการเปิดโปรเจกต์ทั้ง 4 โปรเจกต์ตามในข้อ 2 มาทำการคอมไพล์ใหม่และสร้าง symbol ทั้งหมด โดยดำเนินการตามขั้นตอนดังนี้
 - 3.1 ไปที่เมนู File >> Open Project... >> เลือกชื่อโปรเจกต์ที่ต้องการจะเปิด
 - 3.2 เปิดไฟล์วงจรขึ้นมา (Logic Diagram หรือ Schematic : *.gdf หรือ ไฟล์ *.vhd)
 - 3.3 ตั้งค่าอุปกรณ์โดยไปที่เมนู Assignments >> Device >> เลือกใช้ชิพเบอร์ EP3C10E144C8
 - 3.4 ทำการคอมไพล์ และสร้าง Symbol file ของอุปกรณ์
 - 3.5 ปิดโปรเจกต์ File >> Close Project

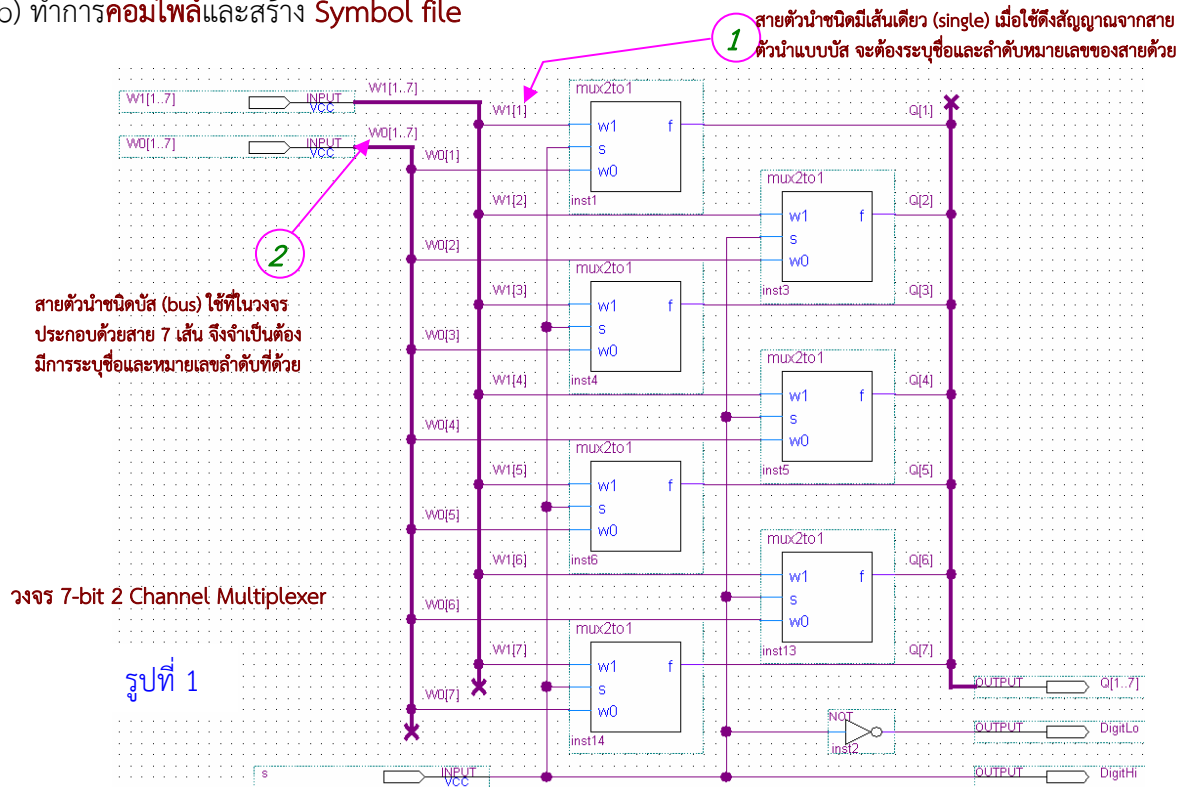
น.ศ. จะต้องทำตามขั้นตอน 3.1 – 3.5 ให้ครบทั้ง 4 อุปกรณ์ก่อนจึงทำข้อต่อไป



ขั้นที่ 1 สร้างตัวมัลติเพล็กซ์เซอร์ขนาด 7 บิต (7-bit 2 Channel Multiplexer)

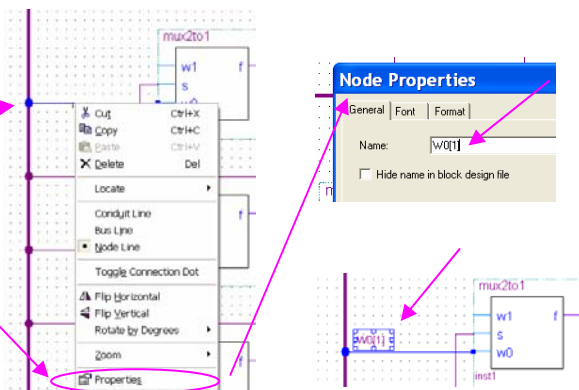
4. ให้**ปิดโปรเจกต์**ที่สร้างมาในขั้นตอนที่ 2-3 ก่อนที่จะดำเนินการต่อไป

- ให้สร้างโปรเจกต์ชื่อ 7CH_2to1MUX และเปิดไฟล์ขึ้นมาเพื่อเขียนวงจรในรูปที่ 1 ซึ่งเป็นมัลติเพล็กซ์เซอร์ขนาด 7 บิต (ภายในประกอบไปด้วย mux2to1 จำนวน 7 ตัว)
- ทำการ**คอมไพล์**และสร้าง **Symbol file**



หมายเหตุ 1 วิธีการตั้งชื่อให้กับสายสัญญาณ

- ใช้เมาส์คลิกที่สายสัญญาณที่ต้องการจนสีเปลี่ยนจากสีน้ำตาลเป็นสีน้ำเงิน
- ยังให้เมาส์ชี้ที่สายเช่นเดิมแต่คลิกปุ่มขวา จะปรากฏเมนูขึ้นมาให้เลือก Property
- ที่หน้าต่าง Property ให้ตั้งชื่อสายตัวนำส่วนเครื่องหมาย [...] นั้นหมายถึงลำดับที่ของสายบัสที่ต่ออยู่กับปลายสายเส้นนี้



ขั้นที่ 2 สร้างตัวหารความถี่ (Clock divider) จาก 20 MHz ให้เหลือ 25Hz, 50Hz

5. ให้**ปิดโปรเจกต์**ที่สร้างมาในขั้นตอนที่ 4 ก่อนที่จะทำการทดลองต่อไป

- ให้สร้างโปรเจกต์ชื่อ **Clock_Divider25xHz** เพื่อเป็นวงจรหารความถี่นาฬิกา (Clock divider) จาก 20 MHz ลงเหลือประมาณ 25 , 50 Hz ตามวงจรในรูปที่ 2 เมื่อสร้างเสร็จให้ทำการ**คอมไพล์**และ**สร้าง Symbol file**



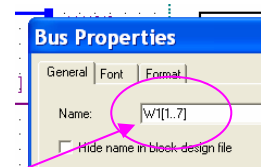
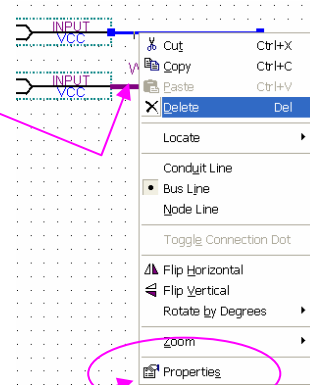
หมายเหตุ ② การกำหนดขนาดความกว้างของบัสข้อมูล
เส้นหนาที่บัสมีวงเชื่อมในวงจรมีชื่อเรียกว่า Orthogonal bus
ซึ่งจะต้องระบุ

1. ชื่อของสาย และจำนวนเส้นของสายตัวนำ
2. การเชื่อมต่อปลายทั้งด้านต้นทาง/ปลายทาง

สามารถดำเนินการได้ดังนี้

- ก) ใช้เมาส์คลิกที่เส้นบัส (Orthogonal bus) ที่ต้องการ
จะปรากฏสีของบัสเปลี่ยนเป็นสีน้ำเงินจากนั้นใช้ปลาย
เมาส์ชี้ที่เส้นบัสแล้วคลิกขวา
เลือกเมนู

>> Property

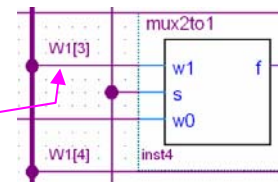


- ข) จะปรากฏหน้าต่างให้ตั้งชื่อบัส ตั้งชื่อเป็น W1[1..7]

W1 = ชื่อของบัส

[1..7] = จำนวนเส้นของบัสมีทั้งหมด 7 เส้น มีหมายเลขกำกับแต่ละเส้นคือ [1], ..., [7]

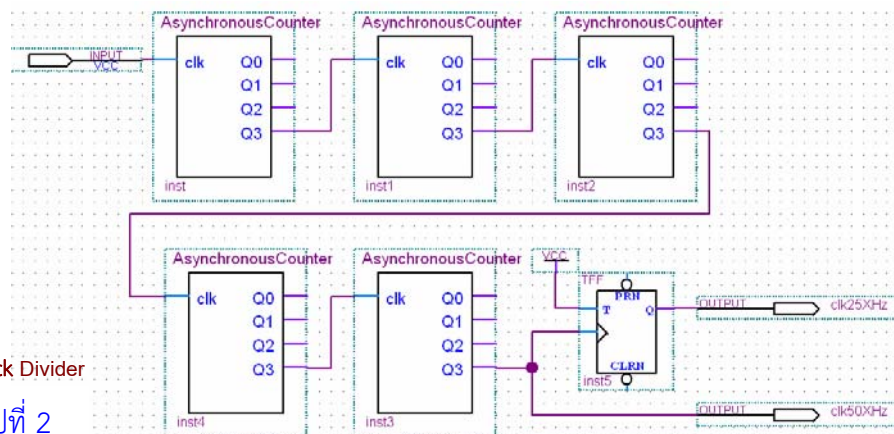
- ค) การเลือกเชื่อมต่อเฉพาะแต่ละปลายของสายจะใช้วิธีเขียนชื่อ
ให้ตรงกัน ระหว่างสายเส้นเล็กสีม่วง (Orthogonal node)
กับสายที่บัสมีวงเชื่อม (Orthogonal bus)



ดังตัวอย่าง ปลายสายเส้นที่ 3 หรือ W1[3] จากบัส W1[1..7] ถูกเชื่อม
ต่อไปที่ขา w1 ของอุปกรณ์ชื่อ inst4 ซึ่งเป็นอุปกรณ์ประเภท mux2to1

วงจร Clock Divider

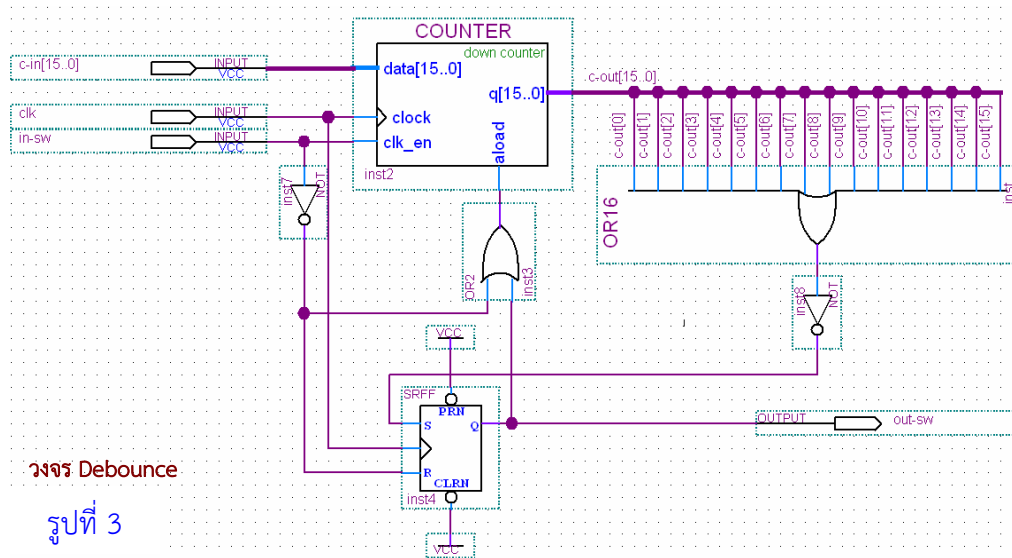
รูปที่ 2





ขั้นที่ 3 สร้างตัวแก้สัญญาณบ๊อช (การรบกวนแบบบอชที่หน้าสัมผัส) ที่สวิตช์ (Debounce)

7. ให้ปิดโปรเจกต์ที่สร้างมาในขั้นตอนที่ 6 ก่อนที่จะดำเนินการต่อไป
8. สร้างอุปกรณ์ Debounce ซึ่งเป็นวงจรสำหรับแก้ปัญหาเรื่อง bounce (bounce : การสวิตช์ระหว่างค่า '1' และ '0' ถัดๆ ในขณะที่กดสวิตช์) โดยดำเนินการตามขั้นตอนดังนี้
 - 8.1 สร้างโปรเจกต์ชื่อ "Debounce" ให้เก็บไว้ในโฟลเดอร์เดียวกันกับโปรเจกต์ที่สร้างตอนก่อนหน้า จากนั้นสร้างวงจรในรูปที่ 3 ใช้ชิพ EP3C10E144C8



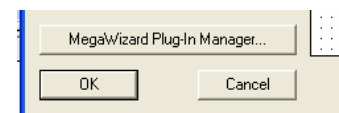
- 8.2 จากรูปที่ 3 จะพบว่ามีอุปกรณ์อยู่ 2 ตัวที่ไม่ใช่อุปกรณ์ทั่วไป และเราไม่รู้จักคือ COUNTER และ OR16

วิธีการสร้างอุปกรณ์ชื่อ COUNTER ให้ดำเนินการโดยไปที่เมนู symbol Tools

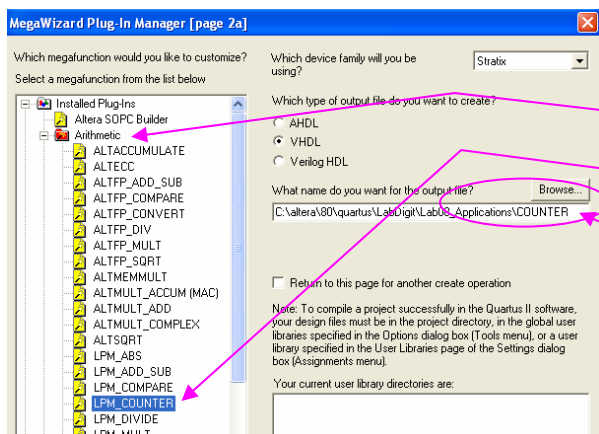
จากนั้นกดปุ่ม button ชื่อ MegaWizard Plug-In Manager
ดังรูปที่ 4 แล้วเลือก

>> Create a new custom megafunction variation

จะปรากฏหน้าต่างสำหรับให้เลือกอุปกรณ์ขึ้นมา



รูปที่ 4



- 8.3 ที่หน้าต่าง MegaWizard... ให้เลือกสร้างอุปกรณ์ดังนี้

ประเภท Arithmetic

ตัวอุปกรณ์เป็น LPM_COUNTER

ตั้งชื่อเป็น COUNTER

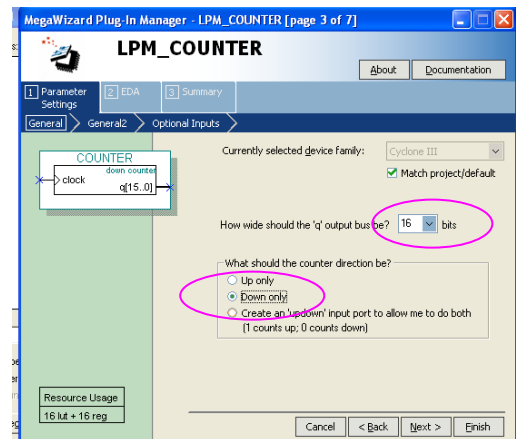
และกดปุ่ม Next

รูปที่ 5



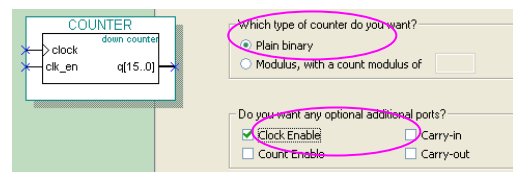
8.4 เลือกจำนวนบิตเป็น 16 บิต และเป็นวงจรมานับแบบนับลง ดังรูปที่ 6

รูปที่ 6

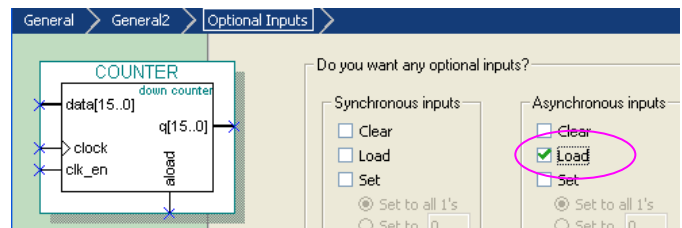


8.5 เลือกวิธีการนับเป็นแบบปกติ

- Plain binary
- ให้มีขาควบคุม Enable แบบ synchronous ดังรูปที่ 7
- เลือกความสามารถสำหรับการกำหนดค่าตั้งต้นนับได้ ดังรูปที่ 8



รูปที่ 7



รูปที่ 8

เมื่อทำไปจนจบขั้นตอนก็จะได้วงจรนับเลขขนาด 16 บิตสามารถนำไปใช้งานได้ทันที

ค่าตัวเลขนับสูงสุดจะถูกกำหนดไว้จากที่อื่นและส่งมาให้ขา data[15..0]

ค่าตัวเลขนี้จะเป็นตัวกำหนดให้วงจร debounce ถ่วงเวลาไว้ก่อนเมื่อสวิตช์ถูกกด โดยการรอนี้จะนานเท่ากับ จำนวนเลขคูณกับค่าคาบสัญญาณนาฬิกา ก่อนที่จะปล่อยเอาท์พุทที่ถูกต้องออกมาให้เรา

8.6 วิธีการสร้างอุปกรณ์ OR16 (OR gate ขนาด 16 อินพุต โดยปกติจะไม่มีให้ใช้งานทั่วไป จึงต้องสร้างขึ้นมาใช้เอง)

สามารถสร้างได้จากหน้าต่าง symbol

โดยเปิดที่หน้าต่าง MegaWizard...

>> MegaWizard Plug-In Manager

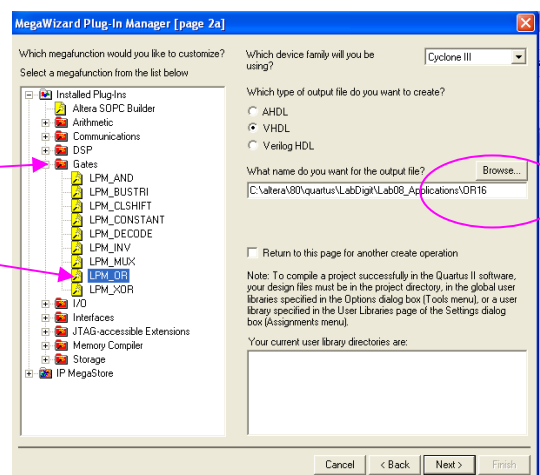
เลือกสร้างอุปกรณ์ประเภท Gates

และสร้างอุปกรณ์เป็น LPM_OR

จากนั้นให้ตั้งชื่อเป็น OR16

ดังรูปที่ 9

รูปที่ 9





9. เมื่อได้อุปกรณ์ COUNTER, OR16 แล้วก็เขียนวงจรในรูปที่ 3 ต่อและทำการคอมไพล์ พร้อมสร้าง symbol file ของวงจรขึ้นมาเตรียมไว้ใช้ในขั้นถัดไป

ขั้นที่ 4 สร้างวงจรนับเลขฐานสิบ ขนาด 4 หลัก (นับจาก 0000 ถึง 9999)

10. ให้ปิดโปรเจกต์ที่สร้างมาในขั้นตอนที่ 8-9 ก่อนที่จะดำเนินการต่อไป

11. สร้างโปรเจกต์ชื่อ “Last4DigitStudentID_VHDL” ขึ้นมาและให้เก็บไว้ในโฟลเดอร์เดิม

a) สร้างวงจรนับเลขในรูปที่ 10 ใช้ภาษา vhdl

b) ใช้ชิพเบอร์ EP3C10E144C8 ให้ทำการคอมไพล์ และสร้าง symbol file ไว้ใช้งานขั้นต่อไป

```

1  LIBRARY ieee ;
2  USE ieee.std_logic_1164.all ;
3  USE ieee.std_logic_unsigned.all ;
4  USE ieee.std_logic_arith.all ;
5
6  ENTITY Last4DigitStudentID_VHDL IS
7  PORT (Clock      : IN    STD_LOGIC ;
8        Clear, En   : IN    STD_LOGIC ;
9        BCD3, BCD2  : OUT   STD_LOGIC_VECTOR(3 DOWNTO 0);
10       BCD1, BCD0  : OUT   STD_LOGIC_VECTOR(3 DOWNTO 0) );
11  END Last4DigitStudentID_VHDL ;
12
13  ARCHITECTURE Behavior OF Last4DigitStudentID_VHDL IS
14  SIGNAL E , EQUAL : STD_LOGIC;
15  SIGNAL PBCD1, PBCD0 : STD_LOGIC_VECTOR(3 DOWNTO 0);
16  SIGNAL PBCD3, PBCD2 : STD_LOGIC_VECTOR(3 DOWNTO 0);
17  CONSTANT DIGIT0 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0100" ; -- Digit0 = 4
18  CONSTANT DIGIT1 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "1000" ; -- Digit1 = 8
19  CONSTANT DIGIT2 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0111" ; -- Digit2 = 7
20  CONSTANT DIGIT3 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0110" ; -- Digit3 = 6
21  BEGIN
22  PROCESS ( Clock )
23  BEGIN
24  IF Clock'EVENT AND Clock = '1' THEN
25  IF Clear = '1' THEN
26  PBCD1 <= "0000" ; PBCD0 <= "0000" ;
27  PBCD3 <= "0000" ; PBCD2 <= "0000" ;
28  ELSEIF E = '1' THEN
29  IF PBCD0 = "1001" THEN
30  PBCD0 <= "0000" ;
31  IF PBCD1 = "1001" THEN
32  PBCD1 <= "0000" ;
33  IF PBCD2 = "1001" THEN
34  PBCD2 <= "0000" ;
35  IF PBCD3 = "1001" THEN
36  PBCD3 <= "0000" ;
37  ELSE
38  PBCD3 <= PBCD3 + '1' ;
39  END IF ;
40  ELSE
41  PBCD2 <= PBCD2 + '1' ;
42  END IF ;
43  ELSE
44  PBCD1 <= PBCD1 + '1' ;
45  END IF ;
46  ELSE
47  PBCD0 <= PBCD0 + '1' ;
48  END IF ;

```

น.ศ. สามารถคัดลอกโปรแกรมได้จาก
โค๊ดภาษา VHDL ที่ด้านท้ายเอกสารแนบ
(ไฟล์เอกสารอิเล็กทรอนิกส์ ***.pdf)

รูปที่ 10



```

49         END IF ;
50     END IF;
51 END PROCESS;
52 E <= En AND EQUAL;
53 BCD3 <= PBCD3;
54 BCD2 <= PBCD2;
55 BCD1 <= PBCD1;
56 BCD0 <= PBCD0;
57 --
58 PROCESS ( PBCD3,PBCD2,PBCD1,PBCD0 )
59 BEGIN
60     IF PBCD3 = DIGIT3 THEN
61         IF PBCD2 = DIGIT2 THEN
62             IF PBCD1 = DIGIT1 THEN
63                 IF PBCD0 = DIGIT0 THEN
64                     EQUAL <= '0' ;
65                 ELSE
66                     EQUAL <= '1' ;
67                 END IF;
68             END IF;
69         ELSE
70             EQUAL <= '1' ;
71         END IF;
72     END PROCESS;
73 END Behavior ;
74

```

รูปที่ 10 (ต่อ)

ขั้นที่ 5 ปรับแต่ง (ดัดแปลง) ให้ชุดวงจร 7-segment ให้แสดงผลได้อย่างเหมาะสม

เนื่องจากบอร์ดทดลองสำหรับแสดงผลเป็นตัวเลข 7-Segment มีการทำงานที่เรียกว่า active low (ต้องป้อนสัญญาณลอจิก '0' ให้ตัวหลอด LED; ไดโอดส่องแสง) วงจร LED จะส่องแสงสีแดงขึ้นมาได้ก็ต่อเมื่อเราต้องจ่ายแรงดันไฟสถานะลอจิก '0' ให้มันเท่านั้น ซึ่งถ้าหากเราใช้ชุดวงจร VHD_7SEGM.vhd จากการทดลองที่ 5 ก็จะแสดงผลตัวเลขได้ไม่เหมาะสม (LED ดวงที่ควรจะติดก็จะดับ ส่วนดวงที่ควรจะดับกลับกลายเป็นติดแทน) จึงต้องมีการปรับเปลี่ยนโปรแกรมบ้างเล็กน้อยด้วยการเพิ่ม NOT gate และสายไฟตัวนำชื่อ temp ใส่เข้าไป ดังรูปที่ 11

```

1  LIBRARY ieee ;
2  USE ieee.std_logic_1164.all ;
3
4  ENTITY VHD_7SEGM IS
5      PORT ( bcd      : IN      STD_LOGIC_VECTOR(3 DOWNTO 0) ;
6            leds      : OUT     STD_LOGIC_VECTOR(1 TO 7) ) ;
7  END VHD_7SEGM ;
8
9  ARCHITECTURE Behavior OF VHD_7SEGM IS
10
11      signal    temp :    STD_LOGIC_VECTOR(1 TO 7);
12
13  BEGIN
14      PROCESS ( bcd )
15      BEGIN
16          CASE bcd IS
17              WHEN "0000" => temp <= "1111110" ;
18              WHEN "0001" => temp <= "0110000" ;
19              WHEN "0010" => temp <= "1101101" ;
20              WHEN "0011" => temp <= "1111001" ;
21              WHEN "0100" => temp <= "0110011" ;
22              WHEN "0101" => temp <= "1011011" ;
23              WHEN "0110" => temp <= "1011111" ;
24              WHEN "0111" => temp <= "1110000" ;
25              WHEN "1000" => temp <= "1111111" ;
26              WHEN "1001" => temp <= "1110011" ;
27              WHEN OTHERS => temp <= "0000000" ;
28          END CASE ;
29      END PROCESS ;
30
31      leds <= NOT temp;
32
33  END Behavior ;

```

ดัดแปลงแก้ไขด้วยการใส่เพิ่ม "NOT" เกท เข้าไปในวงจรก่อนที่จะส่งสัญญาณออกที่พอร์ทชื่อ leds
หมายเหตุ: น.ศ.จะต้องเปิดโปรเจกขึ้นมาทำเช่นเดียวกันกับที่เคยทำในขั้นตอนที่ 2-3 เพียงแต่ในขั้นนี้ให้น.ศ. จะต้องมีการปรับแก้ไขโปรแกรมด้วย

รูปที่ 11

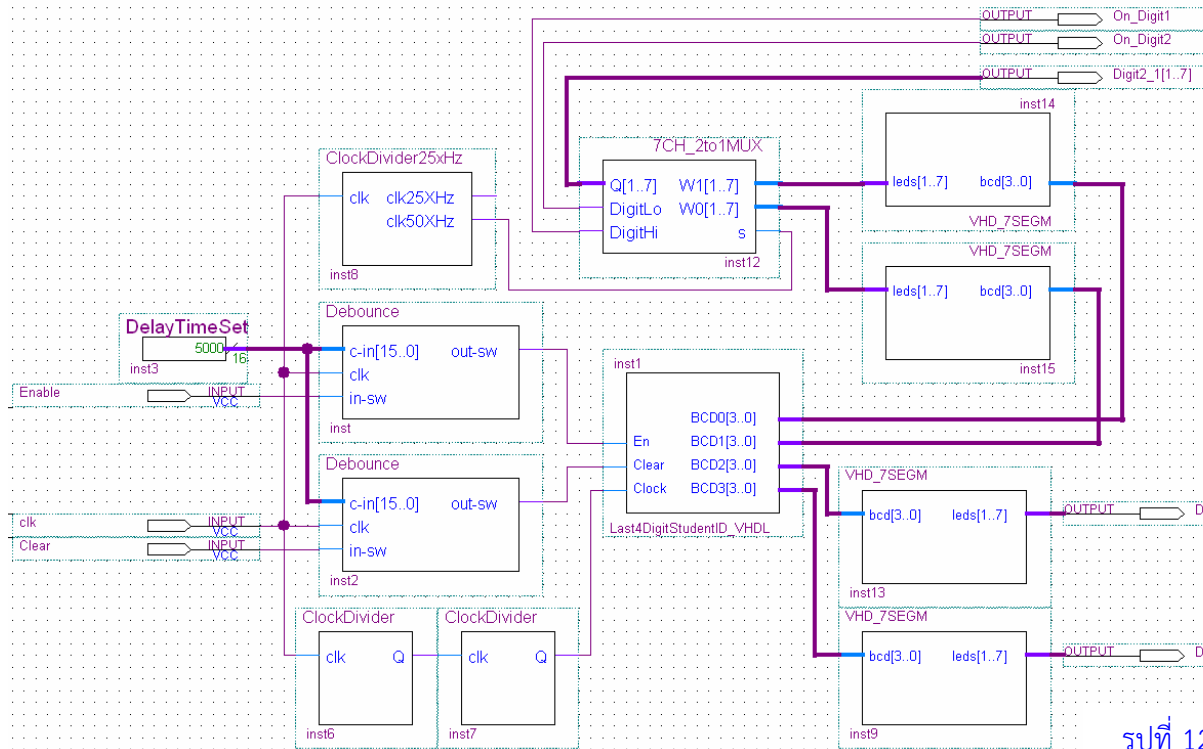


ขั้นที่ 6 ทำการประกอบวงจรที่สร้างมาทั้งหมดมารวมเข้าด้วยกันเป็นวงจรที่สมบูรณ์

12. ให้ปิดโปรเจกต์ที่สร้างมาในขั้นตอนที่ 11 ก่อนที่จะดำเนินการต่อไป

13. สร้างโปรเจกต์ชื่อ “CounterWithDebounce” ขึ้นมาและให้เก็บไว้ในโฟลเดอร์เดิม

a) เปิดไฟล์ขึ้นมาสำหรับเขียนวงจรในดังรูปที่ 12 ให้ใช้ชิพเบอร์ EP3C10E144C8 และคอมไพเลอร์วงจร



รูปที่ 12

หมายเหตุ : การสร้างอุปกรณ์ชื่อ DelayTimeSet

อุปกรณ์ DelayTimeSet ค่าคงที่ (constant) มีไว้สำหรับเป็นค่าการหน่วงเวลาให้กับวงจรสวิตช์ (debounce) สร้างได้โดยใช้เครื่องมือ MegaWizard...

>> MegaWizard Plug-In Manager

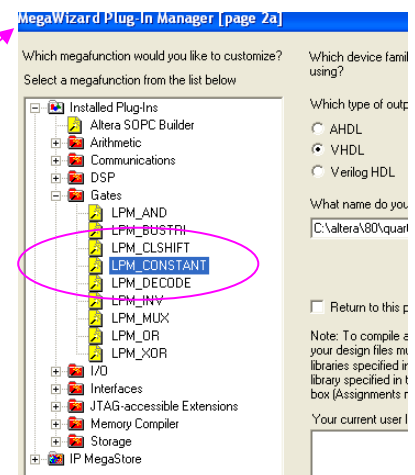
เลือกอุปกรณ์ที่จะสร้างเป็นประเภท

>> gates >> LPM_CONSTANT

ตั้งชื่อเป็น DelayTimeSet

ให้ขนาดความกว้างบัสข้อมูล = 16 บิต

และแสดงข้อมูลเป็นเลขฐานสิบ



ขั้นที่ 7 นำวงจรที่สมบูรณ์ไปทำการโปรแกรม (Configuration) ลงบอร์ดทดลอง

14. ทำการคอมไพล์วงจรและนำชิ้นงานต้นแบบที่ได้จากขั้นตอนที่ 13 ลงในบอร์ดทดลอง Cyclone3-Lab01



โดยกำหนดขาต่าง ๆ ตามรูปที่ 13

	Node Name	Direction	Location	I/O Bank	Vref Group	I/O Standard
1	Clear	Input	PIN_33	2	B2_NO	3.3-V LVTTL (default)
2	clk	Input	PIN_22	1	B1_NO	3.3-V LVTTL (default)
3	Digit2_1[1]	Output	PIN_110	7	B7_NO	3.3-V LVTTL (default)
4	Digit2_1[2]	Output	PIN_112	7	B7_NO	3.3-V LVTTL (default)
5	Digit2_1[3]	Output	PIN_111	7	B7_NO	3.3-V LVTTL (default)
6	Digit2_1[4]	Output	PIN_114	7	B7_NO	3.3-V LVTTL (default)
7	Digit2_1[5]	Output	PIN_113	7	B7_NO	3.3-V LVTTL (default)
8	Digit2_1[6]	Output	PIN_119	7	B7_NO	3.3-V LVTTL (default)
9	Digit2_1[7]	Output	PIN_115	7	B7_NO	3.3-V LVTTL (default)
10	Digit3[1]	Output	PIN_121	7	B7_NO	3.3-V LVTTL (default)
11	Digit3[2]	Output	PIN_125	7	B7_NO	3.3-V LVTTL (default)
12	Digit3[3]	Output	PIN_124	7	B7_NO	3.3-V LVTTL (default)
13	Digit3[4]	Output	PIN_127	7	B7_NO	3.3-V LVTTL (default)
14	Digit3[5]	Output	PIN_126	7	B7_NO	3.3-V LVTTL (default)
15	Digit3[6]	Output	PIN_129	8	B8_NO	3.3-V LVTTL (default)
16	Digit3[7]	Output	PIN_128	8	B8_NO	3.3-V LVTTL (default)
17	Digit4[1]	Output	PIN_133	8	B8_NO	3.3-V LVTTL (default)
18	Digit4[2]	Output	PIN_136	8	B8_NO	3.3-V LVTTL (default)
19	Digit4[3]	Output	PIN_135	8	B8_NO	3.3-V LVTTL (default)
20	Digit4[4]	Output	PIN_138	8	B8_NO	3.3-V LVTTL (default)
21	Digit4[5]	Output	PIN_137	8	B8_NO	3.3-V LVTTL (default)
22	Digit4[6]	Output	PIN_142	8	B8_NO	3.3-V LVTTL (default)
23	Digit4[7]	Output	PIN_141	8	B8_NO	3.3-V LVTTL (default)
24	Enable	Input	PIN_32	2	B2_NO	3.3-V LVTTL (default)
25	On_Digit1	Output	PIN_144	8	B8_NO	3.3-V LVTTL (default)
26	On_Digit2	Output	PIN_143	8	B8_NO	3.3-V LVTTL (default)

รูปที่ 13

- เมื่อตรวจสอบความถูกต้องของขาอุปกรณ์เรียบร้อยแล้วให้ทำการคอมไพล์วงจรอีกครั้ง
- นำบอร์ดทดลองมาต่อแหล่งจ่ายไฟ 12 โวลต์ เปิดสวิตช์ จากนั้นต่อสายสำหรับดาวน์โหลดจากคอมพิวเตอร์ลงในบอร์ดทดลองโดยใช้สาย
 - Byte Blaster** สำหรับเครื่องคอมพิวเตอร์ PC (ติดตั้งไว้แล้วที่ PC ทุกเครื่องในห้องปฏิบัติการ)
 - USB Blaster** สำหรับเครื่องคอมพิวเตอร์โน้ตบุค (น.ศ. ต้องติดตั้งไดรฟ์เวอร์ก่อน)
- ต่อ**บอร์ดแสดงผล 7-Segment** ของ Smart Prototype เข้ากับช่องต่อคอนเน็คเตอร์ Expansion Port B เปิดสวิตช์ป้อนไฟเลี้ยงให้กับวงจร
- ดาวน์โหลดโปรแกรมเพื่อทำ configuration ขึ้นงานต้นแบบลงบอร์ด

ผลการทดลอง

ก) จากคู่มือบอร์ดหน้า 19-20 (**4.WARRIOR CTCLONE3 Education Board User's Manual.PDF**)

- สวิตช์เลื่อน SW7 ถูกต่อเข้ากับพอร์ทชื่อ **PIN_33** ในวงจรรูปที่ 12 ทำหน้าที่เป็น **clear**
ถ้า ป้อนลอจิก '1' $\text{clear} = \text{high} \rightarrow$ ส่วนสิบ Output ถูกเคลียร์ และเริ่มนับใหม่
- สวิตช์เลื่อน SW6 ถูกต่อเข้ากับพอร์ทชื่อ **PIN_32** ในวงจรรูปที่ 12 ทำหน้าที่เป็น **Enable**
ถ้า ป้อนลอจิก '1' $\text{enable} = \text{high} \rightarrow$ วงจรเกิดทริกซ์เลข

ข) ปรับสวิตช์เลื่อน SW7 , SW6 บันทึกการทำงานของวงจร (สิ่งที่เห็นจาก 7-segment)

SW7= Low , SW6= Low การทำงานของวงจร **ไม่ทริกซ์เลข (Clear & En = 0)**

SW7= Low , SW6= High การทำงานของวงจร **ทริกซ์เลข (Clear=0 & En=1)**



SW7= High , SW6= Low การทำงานของวงจร ... ค่า Clear = 1 (SW=high) วงจรจะนับ
Output = 0000 ตลอดวงจรทำงาน

SW7= High , SW6= High การทำงานของวงจร ... แม้ว่า enable = 1 แต่ ค่า Clear = 1
ทำให้ Output ไม่แสดงค่าเลย เป็น 0000 (ถ้า Clear มีค่าเป็น 1 ตลอดเวลา)

ค) วงจรนี้ นับถึงเลขสุดท้ายและค้างค่าไว้ลำดับเลขที่ 6784

ถ้าต้องการจะเปลี่ยนเลขนี้ให้ตรงกับเลขท้าย 4 ตัวของรหัสนักศึกษาจะต้องทำอย่างไร

เปิดไฟล์ VHDL ชื่อ Last4DigitStudent-VHDL ไปที่คำสั่ง constant digit3-digit0
ให้เปลี่ยนจาก 0110 , 0111 , 1000 , 0100 เป็น 0001 , 0001 , 1000 และ 1000 ตามลำดับ

ง) ในรูปที่ 12 ที่ตัวอุปกรณ์ Clock_Divider25xHz ถ้าสัญญาณเข้าที่พุทถูกย้ายไปที่พอร์ท clk25XHz จะเกิด
อะไรขึ้น X เพราะเหตุใด

ไม่ได้ต่อบอร์ดทดลอง

จ) ถ้าต้องการให้การนับเลขมีความเร็วเพิ่มมากขึ้น หรือนับช้าลงจะต้องดัดแปลงวงจรอย่างไร

X

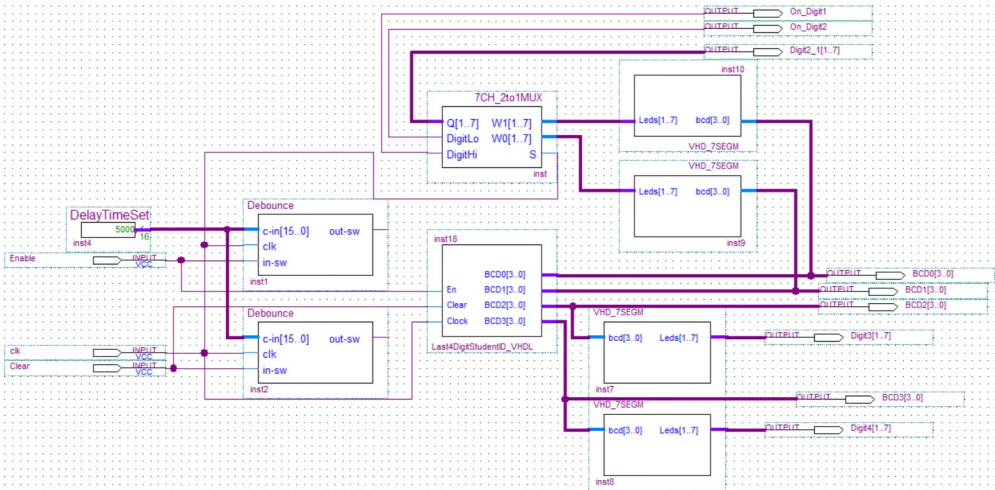
ไม่ได้ต่อบอร์ดทดลอง

งานมอบหมายท้ายการทดลอง

(ให้เขียนลงบนกระดาษ A4 ที่มีเส้นบรรทัดและรวมใส่ท้ายเอกสารการทดลอง ส่งอาจารย์ผู้สอนในคราวถัดไป)

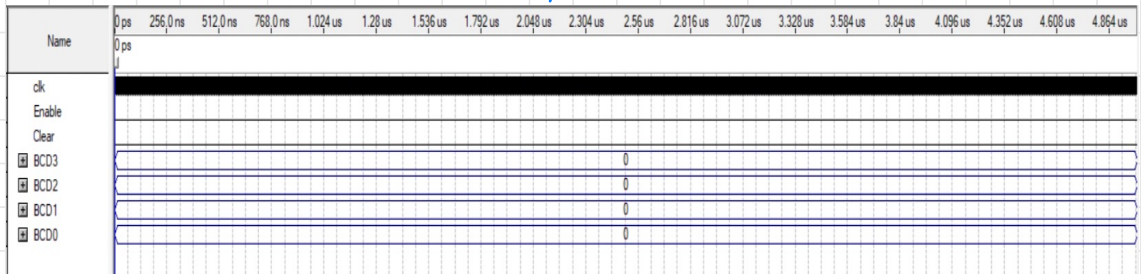
1. ให้อธิบายเรื่อง bounce และการแก้ปัญหาด้วยการทำ debounce
2. ให้ดัดแปลงวงจร โดยเริ่มนับจากรหัสนักศึกษา (4 ตัวท้าย) ลดค่าลงครั้งละ 1 และให้หยุดนับเมื่อลดลงถึง 0

Block Diagram (Lab 8)

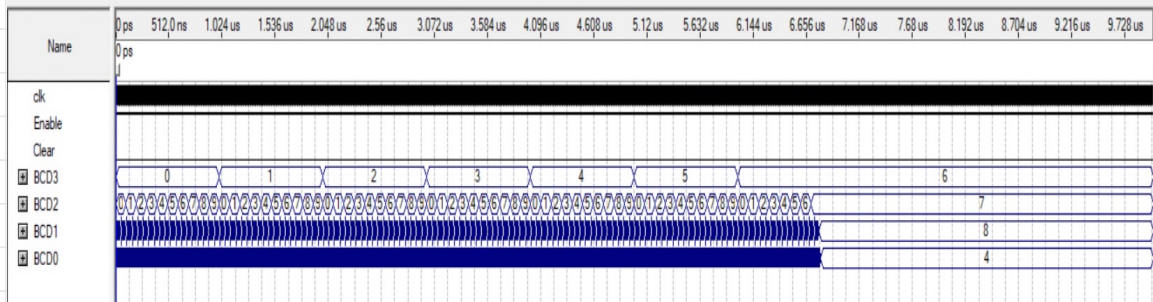


9.

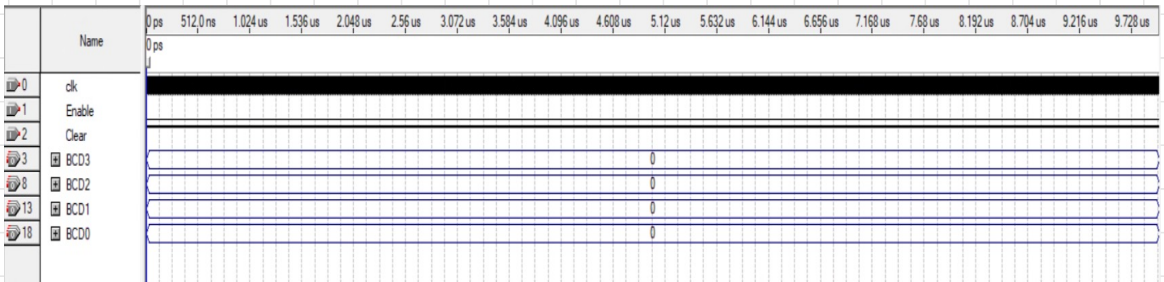
1) Simulation (sw7 low, sw6 low)



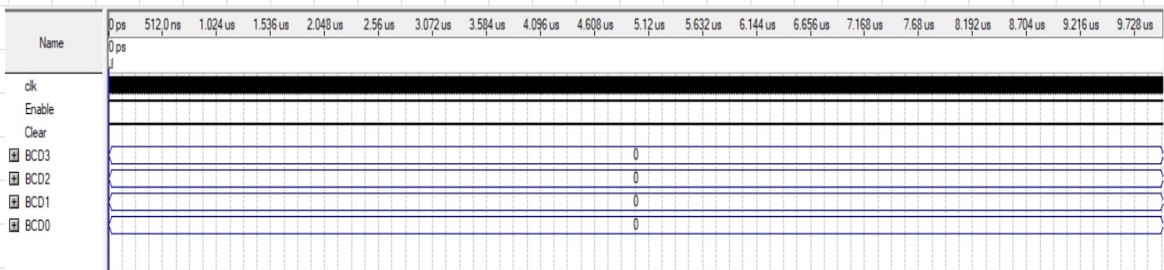
2) Simulation (sw7 low, sw6 high)



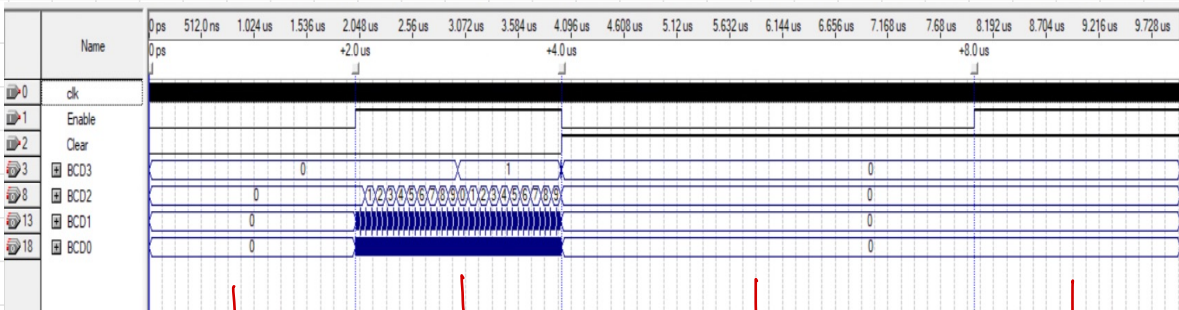
3.) Simulation (sw7 high, sw6 low)



4.) Simulation (sw7 high, sw6 high)



การที่เราจะกดปุ่ม Simulate ก็คือกด

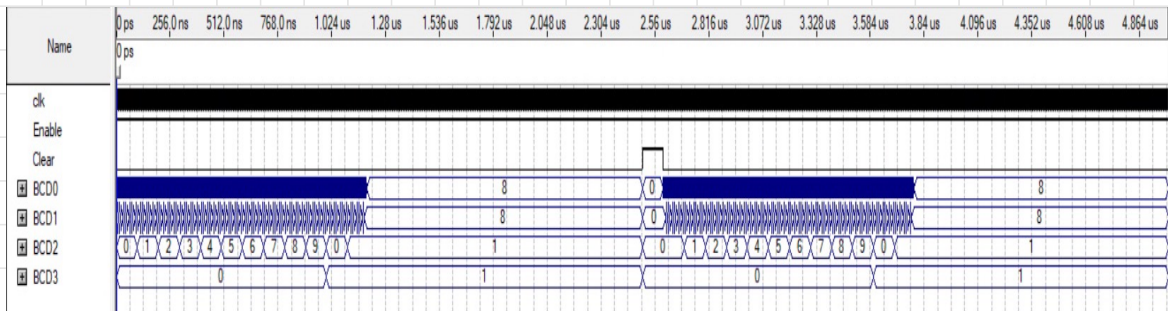


↓
sw7 = low
sw6 = low

↓
sw7 = low
sw6 = high

↓
sw7 = high
sw6 = low

↓
sw7 = high
sw6 = high



เปลี่ยนจาก ชุดที่ 6784 ไปชุดที่ 1188

งานมอบหมายท้ายการทดลอง

(ให้เขียนลงในกระดาษ A4 ที่มีเส้นบรรทัดและรวมใส่ท้ายเอกสารการทดลอง ส่งอาจารย์ผู้สอนในคราวถัดไป)

1. ให้อธิบายเรื่อง bounce และการแก้ปัญหาด้วยการทำ debounce

2. ให้ตัดแปลงวงจร โดยเริ่มนับจากการหัดนักศึกษา (4ตัวท้าย) ลดค่าลงครึ่งละ 1 และให้หยุดนับเมื่อลดลงถึง 0

① Bounce คือ ปัญหาที่เกิดจากการใช้สวิตช์แบบกด ขณะใช้ขาเปิดปิด ที่ทำให้เกิด สัญญาณ Low \rightarrow High ๑-เกิดทรานส์ ทำให้สัญญาณไม่นิ่ง สามารถแก้ไขโดยการทำ Debounce ✖

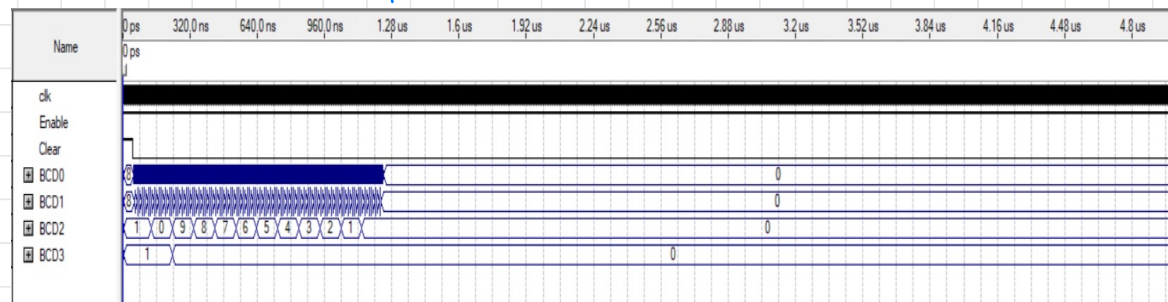
Debounce คือ การหับ Pulse ของสัญญาณไปจนกว่าสัญญาณนั้นจะเกิดทรานส์มากที่สุด เพื่อใช้ค่า มีควมถูกต้องมากที่สุด ✖

② คัดแปลงจากห้องเลขที่ 1188 → 0000

```
13 ARCHITECTURE Behavior OF Last4DigitStudentID_VHDL IS
14     SIGNAL E, EQUAL : STD_LOGIC;
15     SIGNAL PBCD1, PBCD0 : STD_LOGIC_VECTOR(3 DOWNTO 0);
16     SIGNAL PBCD3, PBCD2 : STD_LOGIC_VECTOR(3 DOWNTO 0);
17     CONSTANT DIGIT0 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0000" ; --DIGIT0 = 4
18     CONSTANT DIGIT1 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0000" ; --DIGIT1 = 8
19     CONSTANT DIGIT2 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0000" ; --DIGIT2 = 7
20     CONSTANT DIGIT3 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0000" ; --DIGIT3 = 6
21 BEGIN
22     PROCESS( Clock )
23     BEGIN
24         IF Clock'EVENT AND Clock = '1' THEN
25             IF Clear = '1' THEN
26                 PBCD1 <= "1000" ; PBCD0 <= "1000" ;
27                 PBCD3 <= "0001" ; PBCD2 <= "0001" ;
28             ELSIF E = '1' THEN
29                 IF PBCD0 = "0000" THEN
30                     PBCD0 <= "1001" ;
31                     IF PBCD1 = "0000" THEN
32                         PBCD1 <= "1001" ;
33                         IF PBCD2 = "0000" THEN
34                             PBCD2 <= "1001" ;
35                             IF PBCD3 = "0000" THEN
36                                 PBCD3 <= "1001" ;
37                             ELSE
38                                 PBCD3 <= PBCD3 - '1' ;
39                             END IF ;
40                         ELSE
41                             PBCD2 <= PBCD2 - '1' ;
42                             END IF ;
43                         ELSE
44                             PBCD1 <= PBCD1 - '1' ;
45                             END IF ;
46                         ELSE
47                             PBCD0 <= PBCD0 - '1' ;
48                             END IF ;
49                             END IF ;
```

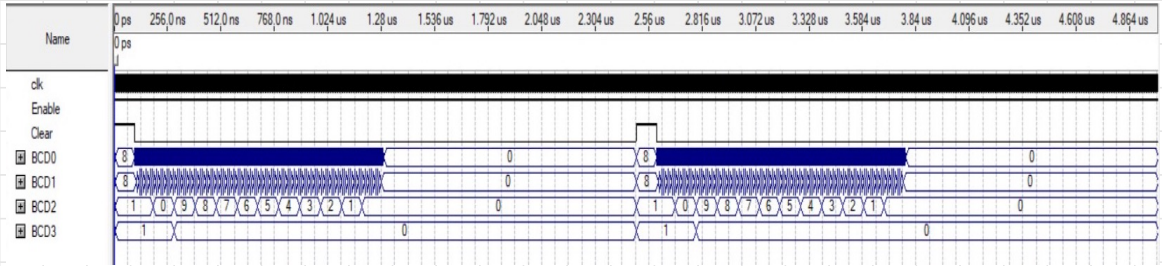
(Last4DigitStudent. VHDL)

“wams Simulation”



ห้องเลขที่ 1188 → 0000

กรณีเมื่อ Clear ให้อัตโนมัติ



(ถ้า Clear = '1' ให้อัตโนมัติ 2.5 - 2.6 us)