



ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

ภาคการศึกษาที่.....ปีการศึกษา.....

รหัสวิชา 010113026

ชื่อวิชา Digital Laboratory

ตอนเรียน หมายเลขโต๊ะ.....

รหัสนักศึกษา..... ชื่อ-นามสกุล.....

อาจารย์ผู้สอน..... เวลาที่ทำการทดลอง วันที่.....

การทดลองที่ 3

Ripple Adder and BCD Adder

วัตถุประสงค์

1. เพื่อให้สามารถใช้โปรแกรมคอมพิวเตอร์จำลองการทำงานของวงจรลอจิกเกตได้
2. เพื่อให้เข้าใจคุณลักษณะและข้อจำกัดด้านเวลาหน่วงของวงจรบวกเลขแบบ Ripple Adder
3. เพื่อให้เข้าใจคุณลักษณะของวงจร BCD adder
4. เพื่อให้สามารถสร้างวงจร Ripple adder และวงจร BCD adder โดยใช้ภาษา VHDL ได้

อุปกรณ์

1. ระบบคอมพิวเตอร์ 1 เครื่อง พร้อมติดตั้งโปรแกรม Quartus II เวอร์ชัน 8.0 (Student Edition) ขึ้นไป

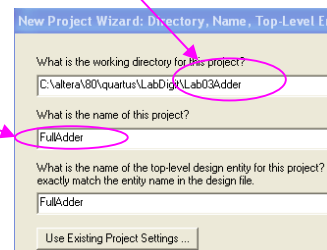
การทดลองตอนที่ 1

สร้าง 4-Bit Ripple Adder ด้วย Symbolic Tools (ขั้นตอนที่ 1-4)

คำสั่งการทดลอง

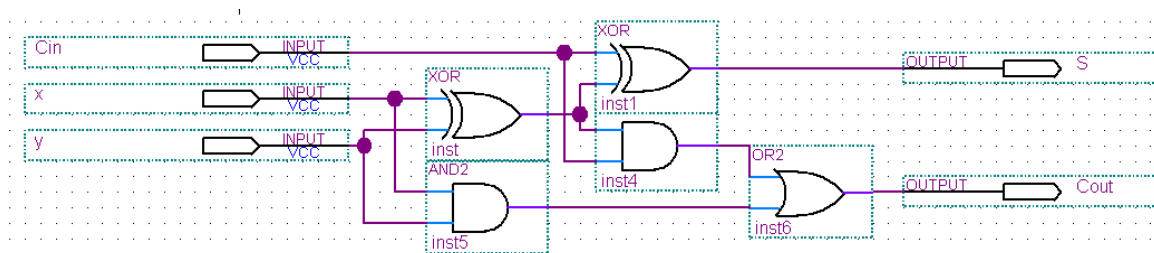
1. ให้ น.ศ. สร้างไฟล์เตอร์สำหรับเก็บงานโปรเจกต์ไฟล์ ขึ้นใหม่เพื่อเก็บงานที่จะทดลองในการทดลองนี้ จากนั้น ให้สร้างโปรเจกต์ชื่อ “FullAdder” ขึ้นมา ดังรูปที่ 1 และให้ใช้ชิพ EP3C10E144C8

ไฟล์เตอร์ที่ต้องการใช้เก็บงานการทดลอง



รูปที่ 1

2. เขียนวงจร Full Adder ในรูปที่ 2 ด้วย Graphic Editor Tool และทำการคอมไพล์ให้เรียบร้อย



รูปที่ 2



3. ทำการสร้าง Symbol ของวงจร Full Adder โดยไปที่เมนู

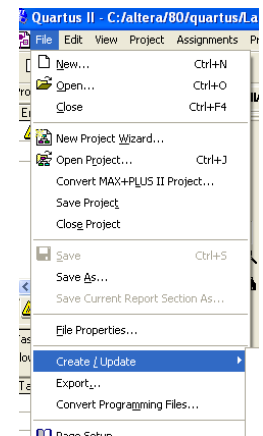
File >> Create/Update

ดังรูปที่ 3

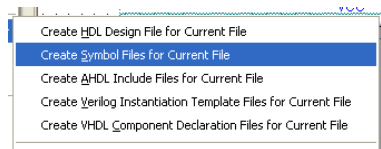
จากนั้นเลือกแถบเมนู

Create Symbol for Current file

ดังรูปที่ 4

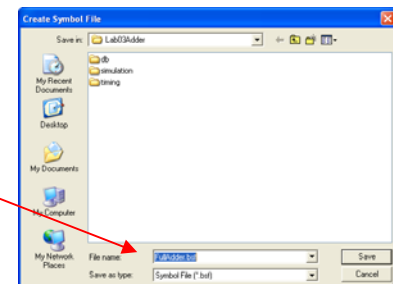


รูปที่ 3



รูปที่ 4

โปรแกรมจะทำการสร้าง symbol file ขึ้นมาให้โดยจะมีชื่อเดียวกันกับชื่อของโปรเจกต์ไฟล์
ทำการ save ไฟล์ให้เรียบร้อย



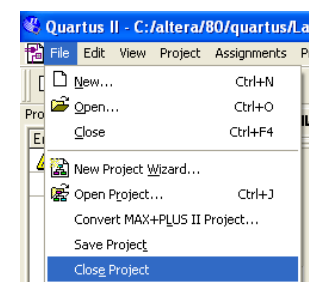
รูปที่ 5

คำแนะนำ : การสร้าง **Symbol File** นั้นหมายความว่า ชิ้นงานที่เราสร้างขึ้นและคอมไพล์เสร็จแล้วจะถูกนำไปใส่หีบห่อ เรียกว่าโมดูล symbol ทำให้เรามองเห็นเป็นอุปกรณ์ชนิดใหม่ (ที่เป็นของเราเองไม่ได้อยู่ในไลบรารีของ Quartus II มาแต่แรก) สามารถนำมาต่อใช้งานในวงจรลอจิกเช่นเดียวกับอุปกรณ์อื่นๆ ที่มีในไลบรารี การทำเช่นนี้จะช่วยให้เราสร้างสรรค์ชิ้นงานที่มีความซับซ้อน ใหญ่มากขึ้นได้เรื่อยๆ แทบไม่มีขีดจำกัด จะมีขีดจำกัดอย่างเดียวของเราคือ ขีดจำกัดด้านความกระตือรือร้นในการแสวงหาความรู้ของเราเองเท่านั้น

4. ให้ทำการปิดโปรเจกต์ที่สร้างมาในขั้นตอนที่ 1 – 3 ก่อนที่จะทำการทดลองต่อไป โดยเข้าไปที่เมนู

File >> Close Project

ดังรูปที่ 6



รูปที่ 6

คำเตือน : น.ศ. จะต้องทำตามขั้นตอน 1-4 อย่าง
เคร่งครัด ถ้าไม่เช่นนั้นจะประสบปัญหาในการคอมไพล์
และส่งผลให้ทำการทดลองมีการผิดพลาดขึ้น



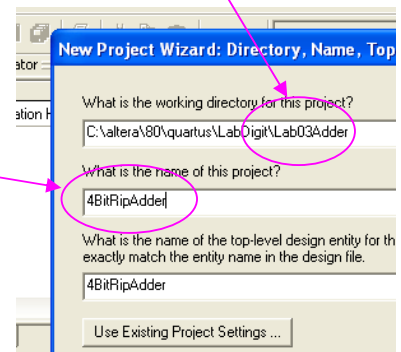
5. ให้ น.ศ. สร้างโปรเจกต์ชื่อ “4BitRipAdd” ขึ้นมาใหม่ และให้เก็บไว้ในโฟลเดอร์เดียวกันกับงานโปรเจกต์ที่สร้างในข้อที่ 1-3 โดยให้ใช้ชื่อ EP3C10E144C8 เช่นเดิม ดังรูปที่ 7

คำว่า “4BitRipAdder” เกิดจากการผสมคำมาจาก 4bit + Ripple + Adder น.ศ. จะพบกับการผสมคำในทำนองนี้ตลอดไปทุกการทดลองของวิชานี้

โปรเจกต์ “4BitRipAdd” ที่กำลังจะสร้างขึ้น

รูปที่ 7

เก็บไว้ในโฟลเดอร์เดียวกันกับการทดลองข้อ 1-3

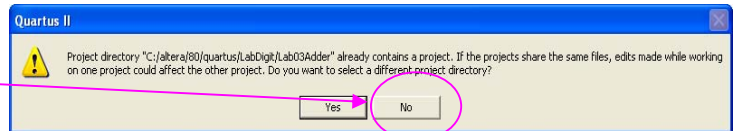


เนื่องจากมีงานโปรเจกต์ที่เราทำไปในข้อ 1 - 3 ค้างอยู่แล้วโปรแกรมจึงสอบถามว่า เราต้องการจะเปลี่ยนเป็นโฟลเดอร์อื่นหรือไม่ ?

ให้เลือก NO

ดังรูปที่ 8

รูปที่ 8



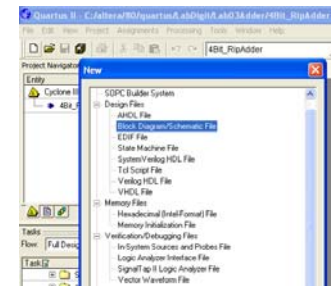
จากนั้นก็ดำเนินการต่อไปจนเสร็จขั้นตอนของการสร้างโปรเจกต์ตามปกติ

6. ทำการเปิดไฟล์ขึ้นมาใหม่เพื่อสร้างวงจร 4bit Ripple Adder โดยไปที่เมนู

File >> New >> Block Diagram/Schematic file

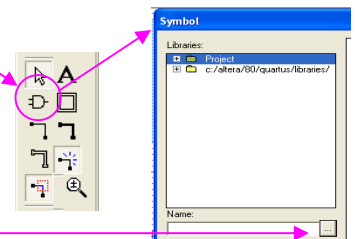
ดังรูปที่ 9

รูปที่ 9



7. ทำการเลือกอุปกรณ์ลอจิกจากไอคอนเครื่องมือ Symbol Tool โดยเปิดหน้าต่าง Symbol ขึ้นมา ดังรูปที่ 10

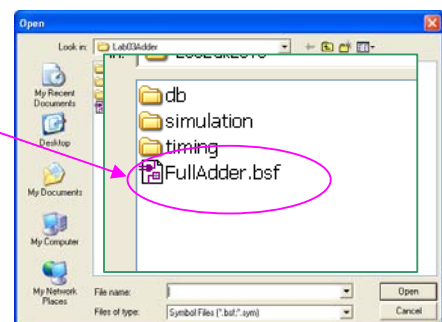
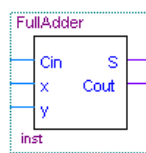
รูปที่ 10



กดปุ่ม work space list ที่อยู่ในหน้าต่างย่อยชื่อ Name จะปรากฏหน้าต่าง แสดง symbol files ขึ้นมาดังรูปที่ 11 ให้เลือกไฟล์ FullAdder.bsf ซึ่งเป็นงานจากการสร้างไว้เสร็จแล้วในขั้นตอนที่ 1-3

เมื่อกดปุ่ม Open จะปรากฏรูปอุปกรณ์ FullAdder ขึ้นมาดังรูปที่ 12

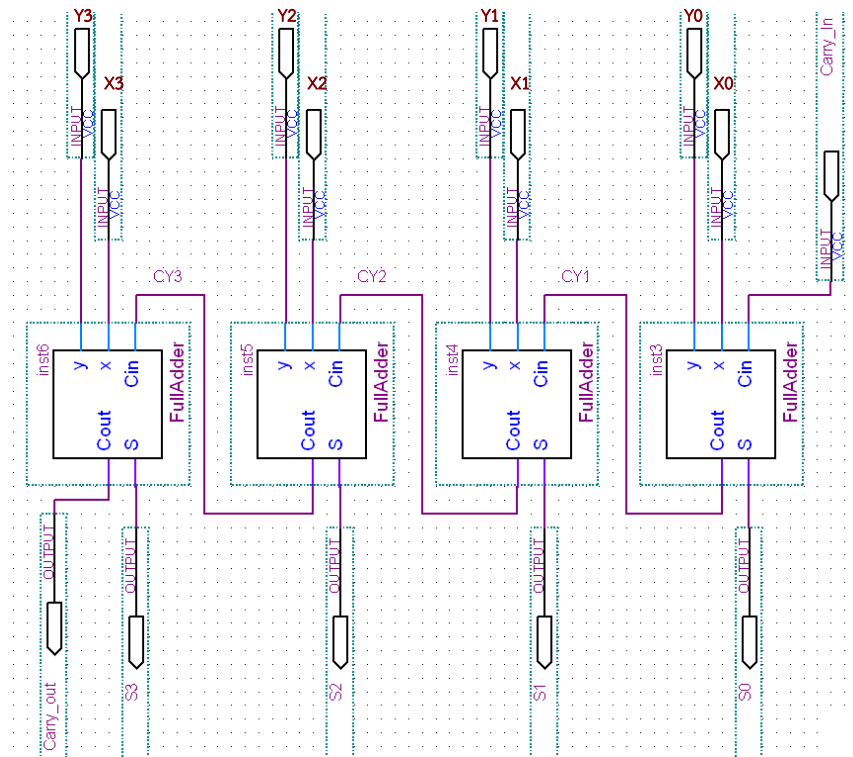
รูปที่ 12



รูปที่ 11



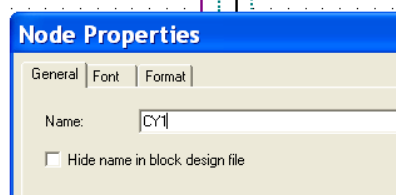
- ให้สร้างวงจรของ “4BitRipAdder” ขึ้นมาดังรูปที่ 13 และตั้งชื่อสายสัญญาณ (ดูวิธีการทำได้ในกล่องข้อความด้านล่าง) ที่ขา Cout เป็น CY3 ,CY2 , CY1 แล้วทำการคอมไพล์
- ทำการสร้าง symbol ของ 4BitRipAdder (ให้ทำเช่นเดียวกับขั้นตอนที่ 3) เพื่อไว้ในตอนท้ายการทดลอง



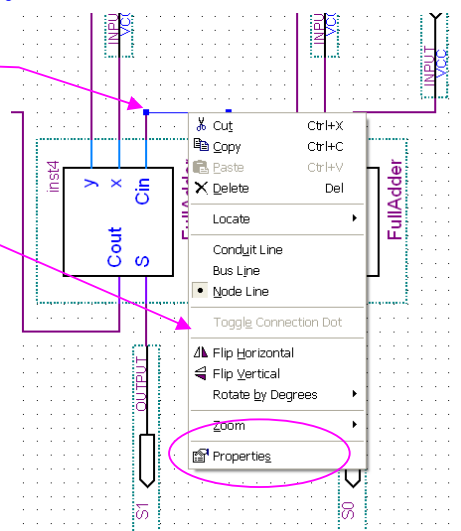
รูปที่ 13

วิธีการตั้งชื่อให้กับสายสัญญาณ (Optional : ไม่ตั้งชื่อสายสัญญาณก็ได้)

- ใช้เมาส์คลิกที่สายสัญญาณที่ต้องการ จนสีเปลี่ยนจากสีน้ำตาลเป็นสีน้ำเงิน
- ยังให้เมาส์ชี้ที่สายเช่นเดิมแต่คลิกปุ่มขวา จะปรากฏเมนูขึ้นมาให้เลือกดังรูปที่ 14
- เลือก Property และตั้งชื่อสายตัวนำดังในรูปที่ 15



รูปที่ 15



รูปที่ 14



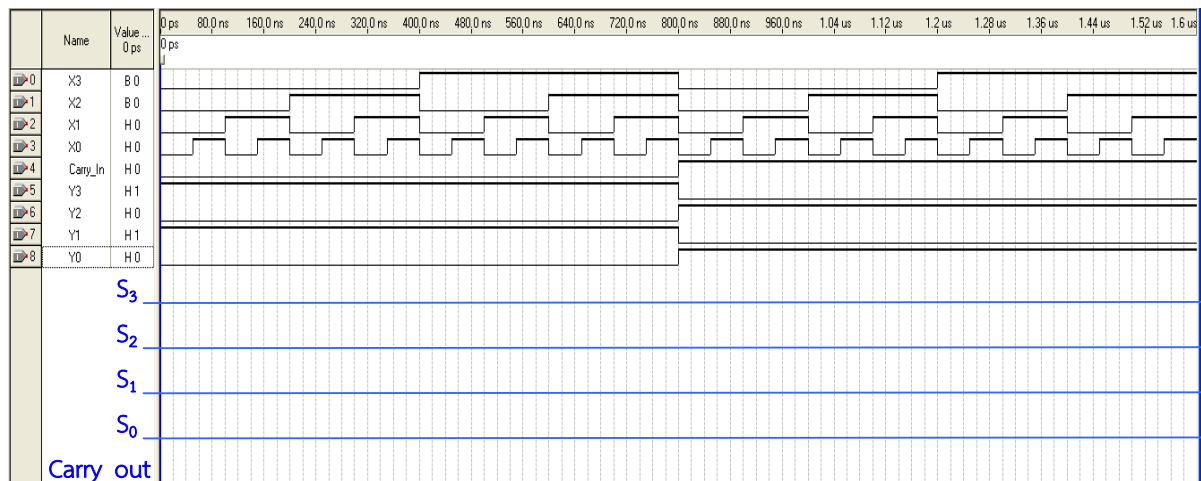
10. กำหนดแผนภาพทางเวลาดังรูปที่ 16

- กราฟของ $X_3X_2X_1X_0$ ค่าเริ่มจาก 0000 ไปจน ถึง 1111 ส่องรอบ (X_0 มี period =100ns)
- ค่า $Y_3Y_2Y_1Y_0 = 1010$ ในรอบแรก เป็น 0101 ในรอบที่สอง
- ค่าของ Carry_in = '0' ในรอบแรก และเป็น '1' ในรอบที่ 2

11. จำลองการทำงานในโหมด Functional mode และบันทึกผลลัพธ์ที่ได้ลงในกราฟรูปที่ 16

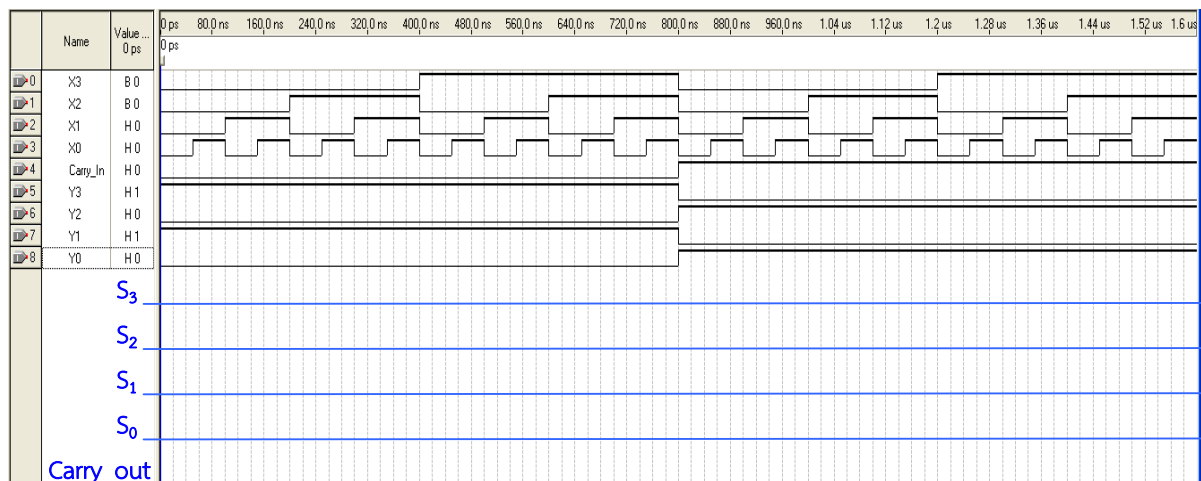
เปลี่ยนจำลองการทำงานเป็นโหมด Timing mode และบันทึกผลลัพธ์ที่ได้ลงในกราฟรูปที่ 17

ผลจำลองการทำงานในโหมด Functional mode



รูปที่ 16

ผลจำลองการทำงานในโหมด Timing mode



รูปที่ 17

จากรูปที่ 17 ผลลัพธ์ของการบวกที่ถูกต้องเกิดขึ้นช้ากว่าการเปลี่ยนแปลงค่าของอินพุต (ให้ช้ชื่อย่อกราฟอย่างละเอียด)

ก) ได้ผลบวกที่ถูกต้องโดยใช้เวลาน้อยที่สุดเมื่อ

$$X_3X_2X_1X_0 = \dots\dots\dots Y_3Y_2Y_1Y_0 = \dots\dots\dots \text{Carry_in} = \dots\dots\dots \text{ด้วยเวลา} \dots\dots\dots \text{ ns}$$

ข) ได้ผลบวกที่ถูกต้องโดยใช้เวลามากที่สุดเมื่อ

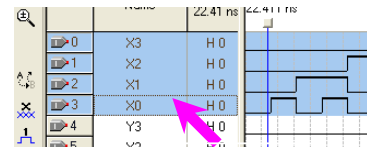
$$X_3X_2X_1X_0 = \dots\dots\dots Y_3Y_2Y_1Y_0 = \dots\dots\dots \text{Carry_in} = \dots\dots\dots \text{ด้วยเวลา} \dots\dots\dots \text{ ns}$$



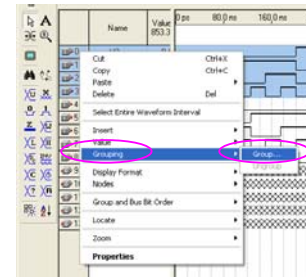
คำแนะนำ

กราฟของ $X_3X_2X_1X_0$, $Y_3Y_2Y_1Y_0$, $S_3S_2S_1S_0$ ในรูปที่ 16 และ 17 ในตอนนี้จะแสดงเป็นกราฟแบบดิจิทัล (สัญญาณละ 1 เส้นกราฟ) ของแต่ละบิตอยู่ในอีกทางหนึ่งเราสามารถมองเป็นเลขขนาด 4 บิตได้ (เช่นเป็น X และมีการเรียงบิตกันตามลำดับของค่านัยสำคัญอยู่แล้ว) ดังนั้นเราจะลองทำการปรับเปลี่ยนรูปแบบของรูปกราฟที่เดิมแสดงค่าแบบ Binary เพียงเส้นละ 1 บิตให้เป็นรูปแบบเลขฐาน 16 ในกราฟเดียว หรือเป็นกราฟแบบเลขฐาน 16 โดยทำได้ดังนี้

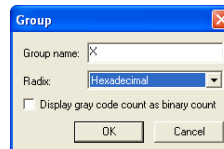
- 1 ทำการรวมกลุ่มของกราฟแบบไบนารีของ $X_3X_2X_1X_0$ ด้วยการกดปุ่ม shift บนแป้นพิมพ์แล้วใช้เมาส์กดเลือกที่ X_3, X_2, X_1 และ X_0 ตามลำดับจะปรากฏแถบสีน้ำเงินขึ้นบนสัญญาณที่ถูกเลือกดังรูปที่ 18
- 2 ใช้เมาส์ชี้ไว้บนพื้นที่แถบสีน้ำเงิน และคลิกเมาส์ปุ่มขวาจะปรากฏหน้าต่างขึ้นมาดังรูปที่ 19
- 3 ตั้งชื่อกลุ่มกราฟของสัญญาณชุดนี้ให้ชื่อ X และเลือกแสดงค่าระบบเลขเป็นแบบเลขฐาน Radix: Hexadecimal ดังรูปที่ 20
- 4 สัญญาณอื่นๆ ที่เหลือก็ทำเช่นเดียวกัน
- 5 ผลที่ได้คือกราฟที่แสดงจะเปลี่ยนค่าไปจากเดิมโดยแสดงค่าเป็นตัวเลขฐาน 16 ทำให้สามารถอ่านค่าได้ง่ายขึ้นดังรูปที่ 21



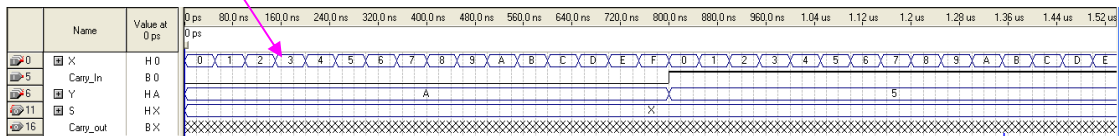
รูปที่ 18



รูปที่ 19



รูปที่ 20



รูปที่ 21

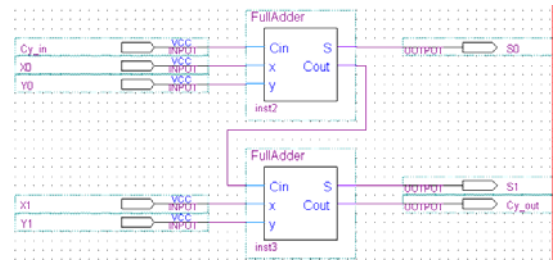
การทดลองตอนที่ 2

วงจรบวกเลขฐานสิบแบบ One-digit BCD Adder ขนาด 1 หลัก

12. ให้ทำการ**ปิดโปรเจกต์เดิมก่อน** (เริ่มต้นเหมือนสร้างโปรเจกต์ใหม่) จะทำการทดลองต่อไปโดยเข้าไปที่เมนู

File >> Close Project

13. ทำการสร้างวงจรบวกเลข 2-bit ในรูปที่ 22 โดยใช้ชื่อโปรเจกต์เป็น 2bitAdder และ**เก็บงานโปรเจกต์ไว้ในโฟลเดอร์เดียวกันกับ 4BitRipAdder** (จากการทดลองตอนที่ 1) ส่วนวิธีการสร้างก็ทำขั้นตอนแบบเดียวกันกับในข้อที่ 5-9



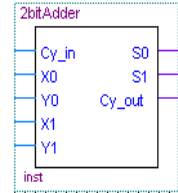
รูปที่ 22



และให้สร้าง symbol file ของวงจรด้วย (ทำขั้นตอนตามข้อ 3) เพื่อจะนำไปใช้ในข้อที่ 14 ต่อไป
เมื่อดำเนินการเสร็จแล้วให้ทำการปิดโปรเจกต์ก่อนที่จะทำการทดลองต่อไป

คำแนะนำ ในขั้นนี้จะ Quartus II จะสร้างอุปกรณ์สำหรับบวกเลขไบนารีขนาด 2 บิตซึ่งมี symbol ดังรูปที่ 23 เก็บไว้ในโฟลเดอร์ที่เราทำงานมาแต่แรกแล้วหากแต่ น.ศ. จะยังไม่เห็นจนกว่าจะดำเนินการทดลองในขั้นตอนถัดจากนี้ไปซึ่งจำเป็นต้องเปิดโปรเจกต์ขึ้นใหม่ก่อน

รูปที่ 23



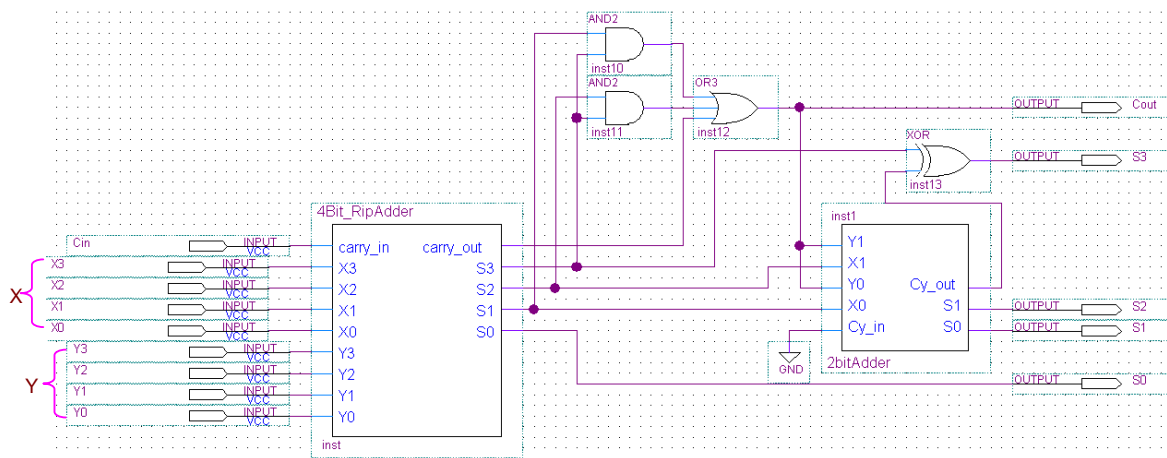
14. ให้ทำการสร้างวงจรบวกเลขแบบ One-digit BCD Adder ที่เขียนไว้ในรูปที่ 24 โดยมีลำดับดังนี้

a) ให้ทำการปิดโปรเจกต์เดิมก่อนที่จะทำการทดลองต่อไป โดยเข้าไปที่เมนู

File >> Close Project

b) จากนั้นให้สร้างโปรเจกต์ขึ้นใหม่ ชื่อ **BCD_Adder** โดยยังคงเก็บงานโปรเจกต์นี้ไว้ในโฟลเดอร์เดียวกันกับ 4BitRipAdd และ 2bitAdder (งานของข้อ 1 -13)

c) สร้างไฟล์สำหรับเขียนวงจรขึ้นใหม่ชื่อ **BCD_Adder** เช่นเดียวกับชื่อของโปรเจกต์ นำอุปกรณ์ **4BitRipAdd** และ **2bitAdder** ที่ได้สร้างไว้แต่แรกมาใช้ในวงจร



Input X , Y : เป็นเลขฐานสิบ (BCD) แต่เขียนแทนด้วยระบบไบนารีขนาด 4 บิต (0000,0001, ... , 1000, 1001)

Cin : เป็นเลขตัวทด ทางด้านอินพุต (มีค่า '0' หรือ '1')

Output S (S₃S₂S₁S₀) : เป็นเลขฐานสิบ (BCD) แต่เขียนแทนด้วยระบบไบนารีขนาด 4 บิต (0000,0001, ... , 1000, 1001)

Cout : เป็นเลขตัวทด ทางด้านเอาท์พุต (มีค่า '0' หรือ '1')

รูปที่ 24

15. คอมพิวเตอร์และจำลองการทำงาน โดยจัดให้แผนภาพทางเวลา (Timing diagram) เป็นดังนี้

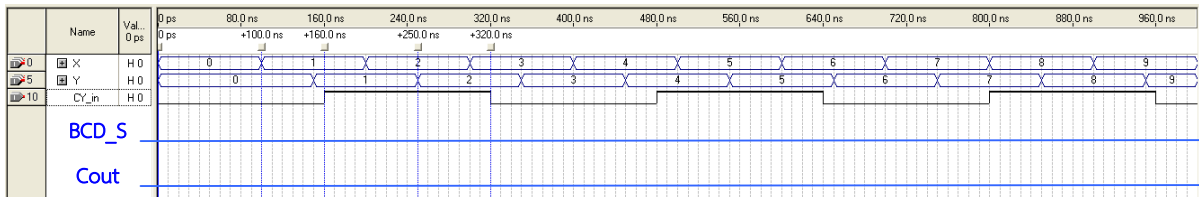
- จัดกลุ่มของสัญญาณให้เป็น **X, Y** และ **BCD_S** เพื่อให้อ่านค่าแบบเลขฐานสิบได้ดังรูปที่ 25
- สัญญาณ Cin ให้เป็นแบบสัญญาณ clock (มีค่า '1' และ '0' สลับกัน) มี Period 320 ns
- สัญญาณ X ให้เป็นจำนวนนับ (Count) เพิ่มครั้งละ 1 ทุกๆ 100 ns เริ่มต้นนับที่เวลา t = 0 ns
- สัญญาณ Y ให้เป็นจำนวนนับ (Count) เพิ่มครั้งละ 1 ทุกๆ 100 ns เริ่มต้นนับที่เวลา t = 50 ns



- จำลองการทำงานในโหมด Functional mode และบันทึกผลจำลองการทำงานลงในกราฟรูปที่ 25
- จำลองการทำงานในโหมด Timing mode และบันทึกผลจำลองการทำงานลงในกราฟรูปที่ 26

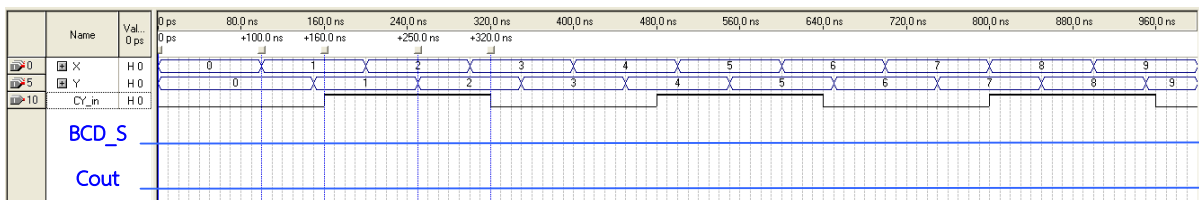
บันทึกผลการทดลอง

ผลจำลองการทำงานในโหมด Functional mode



รูปที่ 25

ผลจำลองการทำงานในโหมด Timing mode



รูปที่ 26

จากรูปที่ 26 ผลลัพธ์ของการบวกที่ถูกต้องเกิดขึ้นช้ากว่าการเปลี่ยนแปลงค่าของอินพุต (ให้ชুমขยายดูกราฟอย่างละเอียด)

ก) เกิดขึ้นเร็วที่สุดเมื่อ X = Y = CY_in = ด้วยเวลา ns

ข) เกิดขึ้นช้าที่สุดเมื่อ X = Y = CY_in = ด้วยเวลา ns

ค) ให้ชুমขยายดูรายละเอียดของกราฟกรณีที่ช้าที่สุดตาม ข) และเขียนรูปกราฟอย่างละเอียด

X _____
Y _____
CY_in _____
BCD_S _____
Cout _____

ง) จากรูปกราฟที่ น.ศ. วาดถ้าเริ่มนับจากจุดที่สัญญาณอินพุตมีการเปลี่ยนแปลงไปสิ้นสุดจุดที่ได้ค่าผลบวกที่ถูกต้อง มีการเปลี่ยนแปลงของค่าผลลัพธ์ BCD_S ทั้งหมด ครั้ง มีค่าเท่าใดบ้าง

(น.ศ.สามารถใช้เคอร์เซอร์เลื่อนไปวางไว้ตรงจุดที่ต้องการ ค่าของ BCD_S จะแสดงขึ้นที่ขอบกราฟด้านซ้าย)

.....
.....
.....
.....

ลายเซ็นอาจารย์ผู้ควบคุม..... /...../.....



การทดลองตอนที่ 3 สร้างวงจรบวกเลขด้วยภาษา VHDL :

ตอนที่ 3-1 วงจรบวกเลข 4-bit Adder

16. ให้เขียนภาษา VHDL เพื่อสร้างวงจรบวกเลข Full Adder ขึ้นมาก่อน ดังแสดงในรูปที่ 27 โดยดำเนินการดังนี้

- ให้สร้างโปรเจกต์ขึ้นใหม่ชื่อ **fulladd**
- ใช้ Text editor สร้างไฟล์ชื่อ **fulladd.vhd** และเก็บงานนี้ไว้ในโฟลเดอร์เดียวกันกับการทดลองก่อนหน้านี้ จากนั้นทำการคอมไพล์
- ให้ทำการสร้าง **symbol file**
- ปิดโปรเจกต์ไฟล์ โดยเข้าไปที่เมนู
File >> Close Project
ก่อนที่จะทำการทดลองขั้นตอนต่อไป

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY fulladd IS
    PORT ( Cin, x, y      : IN  STD_LOGIC ;
          s, Cout         : OUT STD_LOGIC ) ;
END fulladd ;

ARCHITECTURE LogicFunc OF fulladd IS
BEGIN
    s <= x XOR y XOR Cin ;
    Cout <= (x AND y) OR (Cin AND x) OR (Cin AND y) ;
END LogicFunc ;
```

รูปที่ 27

17. ให้เขียนภาษา VHDL เพื่อสร้างวงจรบวกเลข 4 บิตดังแสดงในรูปที่ 28 โดยดำเนินการดังนี้

- สร้างโปรเจกต์ขึ้นใหม่ชื่อ **adder4**
- ใช้ Text editor สร้างไฟล์ชื่อ **add4.vhd** และให้เก็บงานนี้ไว้ในโฟลเดอร์เดียวกันกับการทดลองในข้อ 16 หากไม่เช่นนั้นจะคอมไพล์คำสั่ง COMPONENT ไม่ได้
- จากนั้นให้ทำการสร้าง **symbol file** ขึ้นมาเพื่อเตรียมไว้ใช้ในการทดลองต่อไป

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY adder4 IS
    PORT ( CY_in      : IN  STD_LOGIC ;
          x3, x2, x1, x0 : IN  STD_LOGIC ;
          y3, y2, y1, y0 : IN  STD_LOGIC ;
          s3, s2, s1, s0 : OUT STD_LOGIC ;
          Cout         : OUT STD_LOGIC ) ;
END adder4 ;

ARCHITECTURE Structure OF adder4 IS
    SIGNAL c1, c2, c3 : STD_LOGIC ;
    COMPONENT fulladd
        PORT ( Cin, x, y      : IN  STD_LOGIC ;
              s, Cout         : OUT STD_LOGIC ) ;
    END COMPONENT ;
BEGIN
    stage0: fulladd PORT MAP ( CY_in, x0, y0, s0, c1 ) ;
    stage1: fulladd PORT MAP ( c1, x1, y1, s1, c2 ) ;
    stage2: fulladd PORT MAP ( c2, x2, y2, s2, c3 ) ;
    stage3: fulladd PORT MAP (
        Cin => c3, Cout => Cout, x => x3, y => y3, s => s3 ) ;
END Structure ;
```

รูปที่ 28

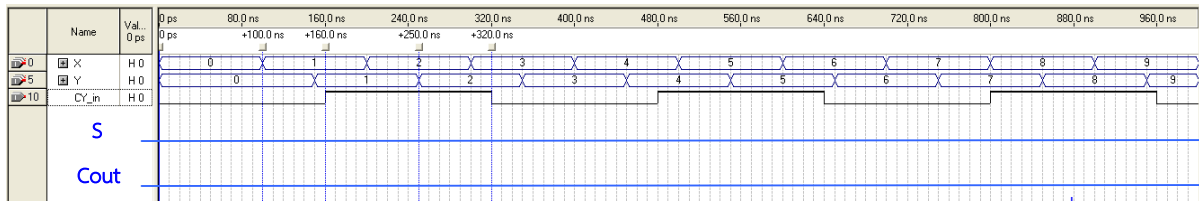
18. สร้างไฟล์แผนภาพทางเวลา (Timing diagram) ทำการจัดกลุ่มสัญญาณให้เป็นกลุ่มเลขไบนารี 4 บิต (nibble) **X, Y** และ **S** สำหรับอ่านค่าแบบเลขฐานสิบหก ดังรูปที่ 29

- 1) จำลองการทำงานในโหมด Functional mode บันทึกผลลงในกราฟรูปที่ 29
- 2) จำลองการทำงานในโหมด Timing mode บันทึกผลลงในกราฟรูปที่ 30



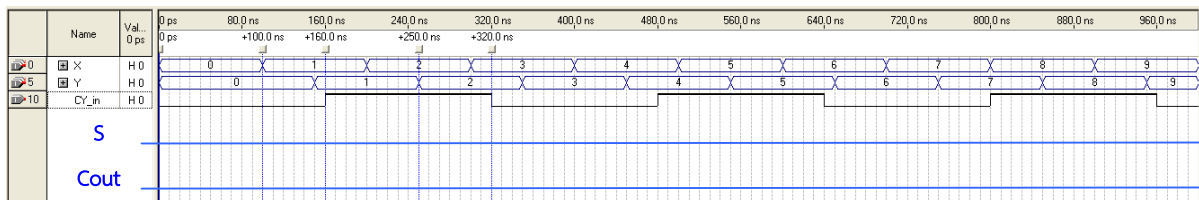
บันทึกผลการทดลอง

ผลจำลองการทำงานในโหมด Functional mode



รูปที่ 29

ผลจำลองการทำงานในโหมด Timing mode



รูปที่ 30

จากรูปที่ 30 ผลลัพธ์ของการบวกที่ถูกต้องเกิดขึ้นช้ากว่าการเปลี่ยนแปลงค่าของอินพุต (ให้ชুমขยายดูกราฟอย่างละเอียด) ก) ได้ผลบวกที่ถูกต้องโดยใช้เวลาน้อยที่สุดเมื่อ

$$X_3X_2X_1X_0 = \dots\dots\dots Y_3Y_2Y_1Y_0 = \dots\dots\dots Cin = \dots\dots\dots \text{ด้วยเวลา} \dots\dots\dots \text{ ns}$$

ข) ได้ผลบวกที่ถูกต้องโดยใช้เวลามากที่สุดเมื่อ

$$X_3X_2X_1X_0 = \dots\dots\dots Y_3Y_2Y_1Y_0 = \dots\dots\dots Cin = \dots\dots\dots \text{ด้วยเวลา} \dots\dots\dots \text{ ns}$$

ค) เปรียบเทียบค่าเวลาหน่วง (delay time) จากข้อ ก) ข) ของรูปที่ 17 และรูปที่ 30 มีค่าต่างกันคิดเป็นร้อยละ

ง) ให้เขียนวงจรบวกที่ได้จากการคอมไพล์โปรแกรม (ดูในเมนู Tool >> Netlist viewer >> RTL Viewer)

ลายเซ็นอาจารย์ผู้ควบคุม..... /...../.....



ตอนที่ 3-2 วงจรบวกเลขแบบ BCD Adder ขนาด 1 หลัก

การทำงานของวงจรบวกเลขแบบ BCD ในรูปที่ 31 เขียนอธิบายพฤติกรรมของโครงสร้างนี้ด้วยภาษา VHDL จะได้เป็นโปรแกรมดังในรูปที่ 32

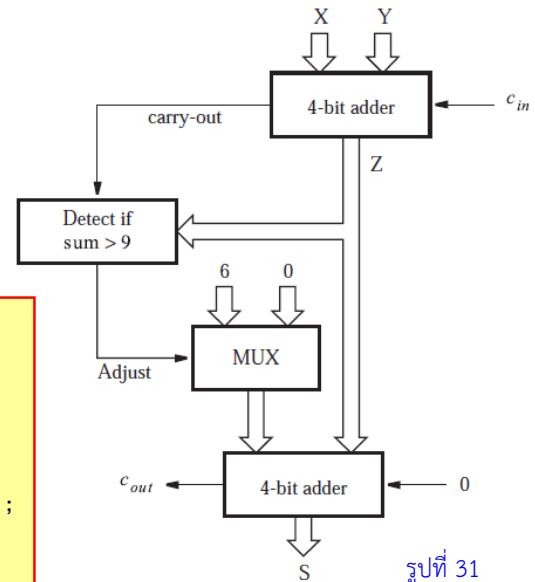
19. ให้ น.ศ. ดำเนินการทดลองดังต่อไปนี้

- ปิดโปรเจกต์เดิม (งานของข้อ 16-18)
- ให้สร้างโปรเจกต์ขึ้นใหม่ชื่อ bcd
- ใช้ Text editor สร้างไฟล์ชื่อ bcd.vhd และเก็บงานนี้ไว้ในโฟลเดอร์เดิม จากนั้นก็ทำการคอมไพล์

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_unsigned.all ;

ENTITY BCD IS
PORT ( X, Y : IN STD_LOGIC_VECTOR(3 DOWNTO 0) ;
      CY_in : IN STD_LOGIC;
      C_out : OUT STD_LOGIC;
      S : OUT STD_LOGIC_VECTOR(4 DOWNTO 0) );
END BCD ;

ARCHITECTURE Behavior OF BCD IS
SIGNAL Z : STD_LOGIC_VECTOR(4 DOWNTO 0) ;
SIGNAL Adjust : STD_LOGIC ;
BEGIN
Z <= ('0' & X) + Y + CY_in;
Adjust <= '1' WHEN Z > 9 ELSE '0' ;
S <= Z WHEN (Adjust = '0') ELSE Z + 6 ;
C_out <= Adjust;
END Behavior ;
```



รูปที่ 31

รูปที่ 32

20. สร้างไฟล์แสดงแผนภาพทางเวลา (Timing diagram) โดยกำหนดดังนี้

X และ Y เป็นสัญญาณแบบ Count value (ปุ่มตัว C ที่แถบเครื่องมือด้านซ้ายของหน้าต่าง)

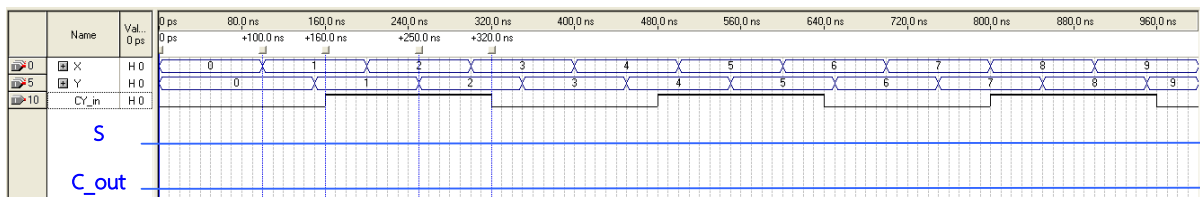
X เริ่มนับจาก 0 ที่เวลา t = 0 เพิ่มครั้งละ 1 ทุกๆ 100 ns

Y เริ่มนับจาก 0 ที่เวลา t = 50 ns เพิ่มครั้งละ 1 ทุกๆ 100 ns

จำลองการทำงานในโหมด Functional บันทึกผลลงในกราฟรูปที่ 33 และจำลองการทำงานในโหมด Timing บันทึกผลลงในกราฟรูปที่ 34

บันทึกผลการทดลอง

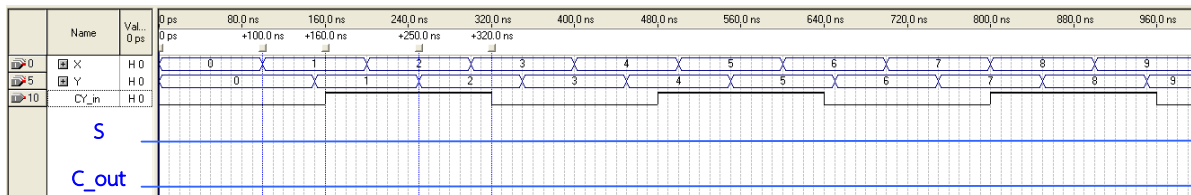
ผลจำลองการทำงานในโหมด Functional mode



รูปที่ 33



ผลจำลองการทำงานในโหมด Timing mode



รูปที่ 34

จากรูปที่ 34 ผลลัพธ์ของการบวกที่ถูกต้องเกิดขึ้นช้ากว่าการเปลี่ยนแปลงค่าของอินพุต (ให้ชুমขยายดูกราฟอย่างละเอียด)

ก) ได้ผลบวกที่ถูกต้องโดยใช้เวลาน้อยที่สุดเมื่อ

$$X_3X_2X_1X_0 = \dots\dots\dots Y_3Y_2Y_1Y_0 = \dots\dots\dots CY_in = \dots\dots\dots \text{ด้วยเวลา} \dots\dots\dots \text{ ns}$$

ข) ได้ผลบวกที่ถูกต้องโดยใช้เวลามากที่สุดเมื่อ

$$X_3X_2X_1X_0 = \dots\dots\dots Y_3Y_2Y_1Y_0 = \dots\dots\dots CY_in = \dots\dots\dots \text{ด้วยเวลา} \dots\dots\dots \text{ ns}$$

ค) เปรียบเทียบค่าเวลาหน่วง (delay time) จากข้อ ก) ข) ของรูปที่ 17 และรูปที่ 34 มีค่าต่างกันคิดเป็นร้อยละ

ง) ให้เขียนวงจรบวกที่ได้จากการคอมไพล์โปรแกรม (ดูในเมนู Tool >> Netlist viewer >> RTL Viewer)

ลายเซ็นอาจารย์ผู้ควบคุม..... /...../.....

งานมอบหมายท้ายการทดลอง

(ให้เขียนลงบนกระดาษ A4 ที่มีเส้นบรรทัดและรวมใส่ท้ายเอกสารการทดลอง ส่งอาจารย์ผู้สอนในคราวถัดไป)

1. ให้สรุปความรู้ที่ได้จากการทดลองนี้
2. ให้อธิบายสรุปเรื่องปัญหาด้านเวลาหน่วง (Time delay) ที่เกิดในวงจรบวกแบบ Ripple Adder