ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ ภาคการศึกษาที่ 1 ปีการศึกษา 2564

รหัสวิชา 010113026 ชื่อวิชา Digital Laboratory ตอนเรียน 7 หมายเลขโต๊ะ -

รหัสนักศึกษา 6201016311188 ชื่อ-นามสกุล พชรวิสรณ์ สุขสมบูรณ์

อาจารย์ผู้สอน CSP เวลาที่ทำการทดลอง 13.00-16.00 วันที่

การทดลองที่ 4

Combinational-Circuit Building Block

วัตถุประสงค์

1. เพื่อให้สามารถใช้โปรแกรมคอมพิวเตอร์จำลองการทำงานของวงจรลอจิกเกทได้
2. เพื่อให้เข้าใจวิธีการสร้างอุปกรณ์ที่ซับซ้อน โดยอาศัยการต่อวงจรด้วยเกทพื้นฐาน
3. เพื่อให้เข้าใจคุณลักษณะพื้นฐานของ มัลติเพล็กเซอร์ ดีมัลติเพล็กเซอร์
4. เพื่อให้เข้าใจคุณลักษณะพื้นฐานของ เอ็นโค้ดเดอร์ ดีโค้ดเดอร์
5. เพื่อให้เข้าใจหลักการทำงานของตัวเปรียบ 4-bit Comparator

เครื่องมือและอุปกรณ์

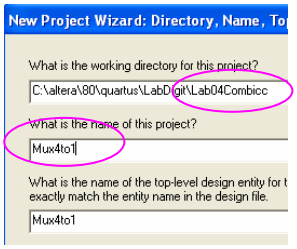
1. ระบบคอมพิวเตอร์ 1 เครื่อง พร้อมติดตั้งโปรแกรม Quartus II เวอร์ชั่น 8.0 (Student Edition) ขึ้นไป

การทดลองตอนที่ 1 หลักการทำงานของมัลติเพล็กเซอร์ 4-to-1 Multiplexer

คำแนะนำ: ในการทดลองนี้ จะเป็นมีรูปแบบ เป็นการสร้างโปรเจคที่ใหม่ขึ้นมาห่อหุ้มคลุมโปรเจคเก่า ไว้เป็นชั้นๆ เช่นในข้อ 1-3 เราจะสร้างอุปกรณ์ชื่อ MUX4to1 ขึ้นมา

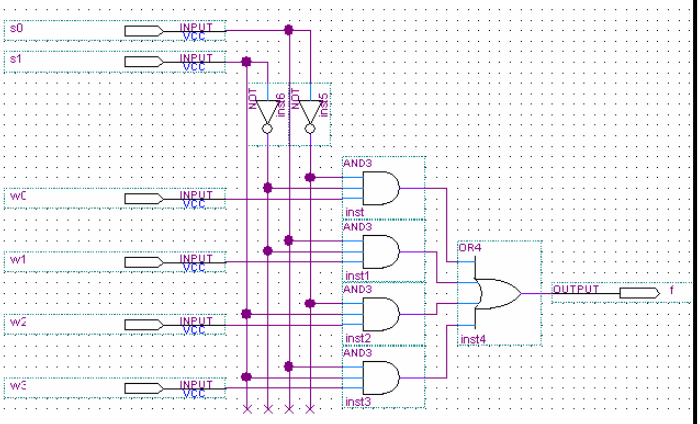
คำสั่งการทดลอง

1. ให้ น.ศ. สร้างโฟลเดอร์สำหรับเก็บงานขึ้นใหม่เพื่อเก็บงานที่จะทดลองในการทดลองนี้ชื่อ “Lab04Combicc”จากนั้นให้สร้างโปรเจคชื่อ “Mux4to1” ขึ้นมา ดังรูปที่ 1 ให้ใช้ชิพ EP3C10E144C8



รูปที่ 1

2. เขียนวงจรมัลติเพล็กเซอร์ขนาด 4-to-1 ดังรูปที่ 2 ด้วย Graphic Editor Tool
   - ทำการคอมไพล์วงจรให้เรียบร้อย
   - สร้าง symbol ของวงจร โดยไปที่เมนู
     File >> create/update
     >> Create symbol file for current file

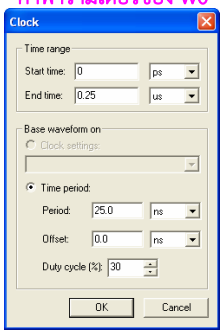


รูปที่ 2

3. สร้างไฟล์แสดงแผนภาพทางเวลา (Timing diagram)
   ตั้งค่า End Time = 1.0 us และ Grid Size = 1 ns (ดูในเมนู Edit >> End time)

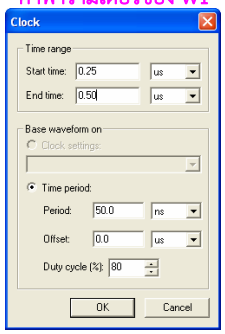
4. กำหนดสัญญาณอินพุท w0, w1, w2 และ w3 ให้มีรูปคลื่นเป็นแบบสัญญาณนาฬิกา โดยสัญญาณและเลือกที่แถบเครื่องมือสร้างรูปคลื่น ดังรูปที่ 3a จากนั้นกำหนดค่าต่างๆตามรูปที่ 3b, 3c, 3d, และ 3e ตามลำดับ

ค่าพารามิเตอร์ของ w0



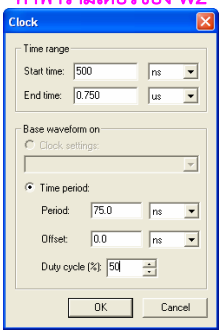
รูปที่ 3b

ค่าพารามิเตอร์ของ w1



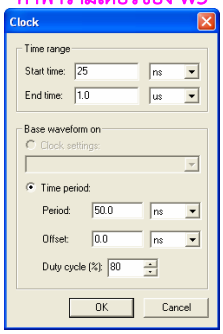
รูปที่ 3c

ค่าพารามิเตอร์ของ w2

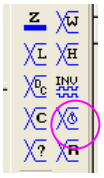


รูปที่ 3d

ค่าพารามิเตอร์ของ w3



รูปที่ 3e



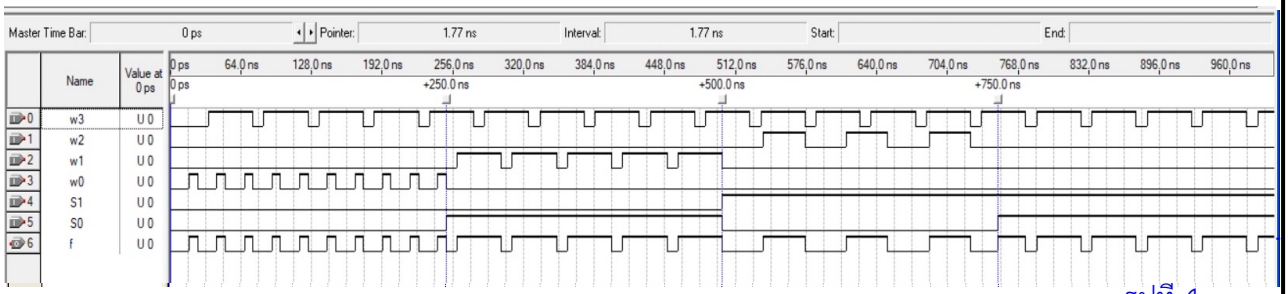
รูปที่ 3a

5. กำหนดของ s0 และ s1 ให้มีรูปคลื่นแบบนาฬิกาดังรูปที่ 4

s0 : Period = 500 ns s1 : Period 1000 ns

6. จำลองการทำงานในโหมด Functional บันทึกผลในกราฟรูปที่ 4

บันทึกผลการทดลอง



รูปที่ 4

เมื่อ s1= ‘0’ s0= ‘0’ สัญญาณที่ปรากฏที่เอ้าท์พุท f มีรูปร่างเหมือนสัญญาณ (อินพุท) ..... w0 .....

เมื่อ s1= ‘0’ s0= ‘1’ สัญญาณที่ปรากฏที่เอ้าท์พุท f มีรูปร่างเหมือนสัญญาณ (อินพุท) ..... w1 .....

เมื่อ s1= ‘1’ s0= ‘0’ สัญญาณที่ปรากฏที่เอ้าท์พุท f มีรูปร่างเหมือนสัญญาณ (อินพุท) ..... w2 .....

เมื่อ s1= ‘1’ s0= ‘1’ สัญญาณที่ปรากฏที่เอ้าท์พุท f มีรูปร่างเหมือนสัญญาณ (อินพุท) ..... w3 .....

สมการของ f = $\bar{S}_1\bar{S}_0W_0 + \bar{S}_1S_0W_1 + S_1\bar{S}_0W_2 + S_1S_0W_3$

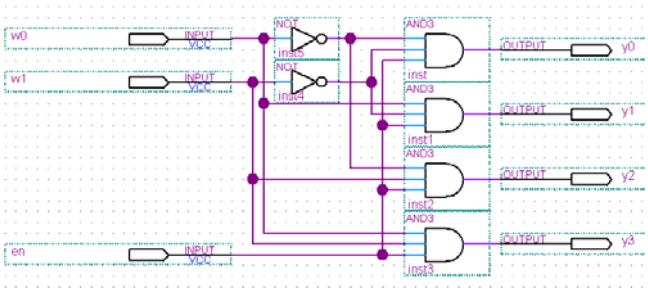
ขาสัญญาณ s1 s0 ของอุปกรณ์ที่ชื่อ Mux4to1 มีหน้าที่อะไร เป็น selector ทำหน้าที่เลือกสัญญาณ Input ให้แสดงผลที่ขา Output

ให้อธิบายหลักการทำงานของอุปกรณ์ที่ชื่อ Mux4to1 Input w[0...1] ถูกเชื่อมเข้าที่ขา logic gate แต่ละตัว และ selecter S_0, S_1 เป็นตัวกำหนดค่า Input ที่จะแสดงผลที่ขา Output

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

การทดลองตอนที่ 2 หลักการทำงานของ 2-to-4 Decoder

7. ให้ปิดโปรเจคเดิม
8. สร้างโปรเจคใหม่ชื่อ “2to4Decoder” โดย**เก็บไว้ในโฟลเดอร์เดิม**(ข้อที่ 1 – 6) ใช้ชิพ **EP3C10E144C8**
9. ใช้ Graphic Editor Tool เขียนวงจรดีโค๊ดเดอร์ขนาด 2-to-4 ดังรูปที่ 5 เก็บไว้ในไฟล์ชื่อเดียวกันกับโปรเจค แล้วทำการคอมไพล์ให้เรียบร้อย
10. สร้าง symbol file ของวงจร
    File >> create/update
    >> Create symbol file for current file

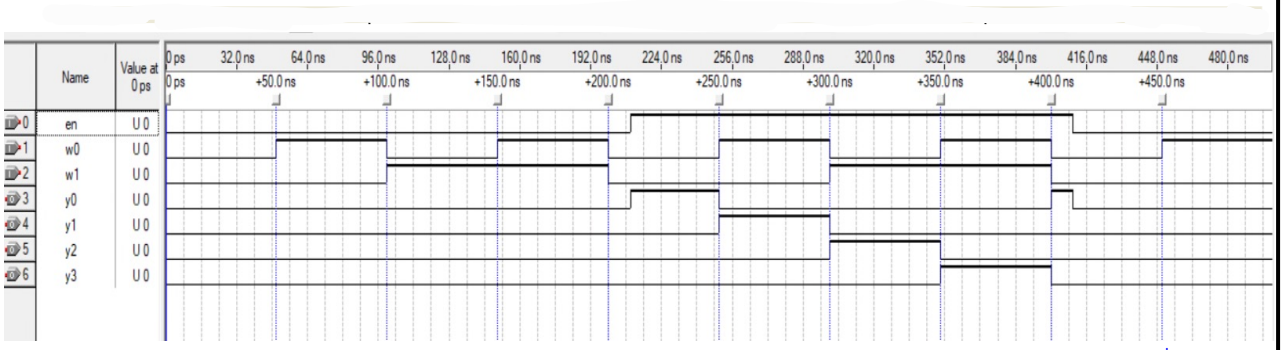


รูปที่ 5

11.สร้างไฟล์แสดงแผนภาพทางเวลา กำหนดค่าของพารามิเตอร์แสดงผลจำลองการทำงานโดยให้มีค่าดังนี้
- End Time = 500 ns Grid Size = 1 ns (ดูที่เมนู Edit >> Grid size)
- กำหนดสัญญาณอินพุท En , w0 และ w1 ให้มีรูปคลื่นดังในรูปที่ 6 โดย
  En กำหนดให้ Start time = 10 ns , Period = 400 ns Duty Cycle[%] = 50
  w1 กำหนดให้ Start time = 0 ps , Period = 200 ns Duty Cycle[%] = 50
  w0 กำหนดให้ Start time = 0 ps , Period = 100 ns Duty Cycle[%] = 50

12. ให้ทำการจำลองการทำงานโหมด Functional mode บันทึกผลลงในกราฟรูปที่ 6

บันทึกผลการทดลอง



รูปที่ 6

เมื่อ w1= ‘0’ w0= ‘0’ สัญญาณ y_0 เป็นเอ้าท์พุทที่ถูกเลือก (ค่าเป็น ‘1’ เพียงตัวเดียวจากทั้งหมด4ตัว)
เมื่อ w1= ‘0’ w0= ‘1’ สัญญาณ y_1 เป็นเอ้าท์พุทที่ถูกเลือก (ค่าเป็น ‘1’ เพียงตัวเดียวจากทั้งหมด4ตัว)
เมื่อ w1= ‘1’ w0= ‘0’ สัญญาณ y_2 เป็นเอ้าท์พุทที่ถูกเลือก (ค่าเป็น ‘1’ เพียงตัวเดียวจากทั้งหมด4ตัว)
เมื่อ w1= ‘1’ w0= ‘1’ สัญญาณ y_3 เป็นเอ้าท์พุทที่ถูกเลือก (ค่าเป็น ‘1’ เพียงตัวเดียวจากทั้งหมด4ตัว)

สมการของเอ้าท์พุท = $y_0 = \bar{w}_1\bar{w}_0\,en$ | $y_2 = w_1\bar{w}_0\,en$
$y_1 = \bar{w}_1 w_0\,en$ | $y_3 = w_1 w_0\,en$

ขาสัญญาณ w1 w0 ของอุปกรณ์ที่ชื่อ **2to4Decode** มีหน้าที่อะไร เป็นขาที่กำหนดค่า Output ทำให้แสดงผล ซึ่งจะมีค่า Output ได้เพียงค่าเดียว

ให้อธิบายหลักการทำงานของอุปกรณ์ที่ชื่อ **2to4Decode** ที่ Input w_1 w_0 เป็นตัวกำหนดค่า Output ที่ต้องให้แสดงผล และที่ En ย่อ Enable เป็นตัวกำหนดการทำงานของวงจรนี้ โดยที่ output จะแสดงผลได้เพียง 1 เดียว ไม่สามารถแสดงผลพร้อมกันได้

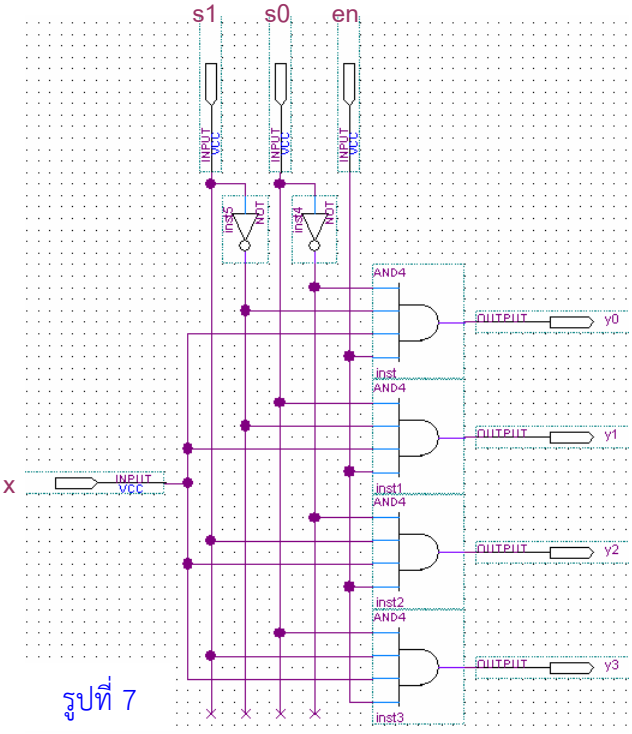
จากผลการทดลองในรูปที่ 6 ให้เขียนตารางความจริง

En	w1	w0	y0	y1	y2	y3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	x	x	0	0	0	0

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

การทดลองตอนที่ 3 หลักการทำงานของ 1-to-4 Demultiplexer

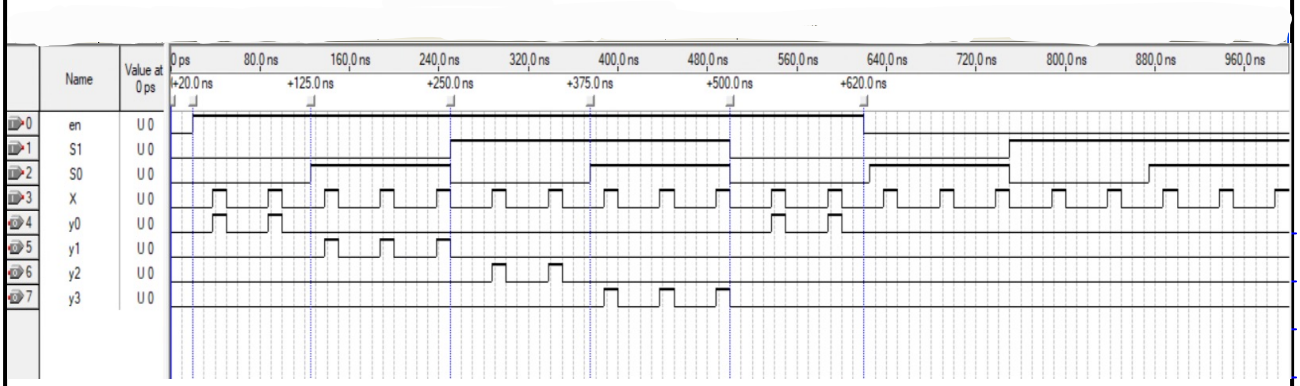
13. ให้ทำการ**ปิดโปรเจค**ที่สร้างมาในขั้นตอนที่ 7 – 12 ก่อนที่จะทำการทดลองต่อไป
14. สร้างโปรเจคชื่อ “1to4Demux” ขึ้นมาใหม่โดย**ให้เก็บไว้ในโฟลเดอร์เดิม**
    ให้ใช้ชิพ **EP3C10E144C8** จากนั้นเขียนวงจรดีมัลติเพล็กเซอร์ 1-to-4 ในรูปที่ 7 เก็บไว้ในไฟล์ชื่อเดียวกันกับชื่อโปรเจค



รูปที่ 7

15. ทำการคอมไพล์ พร้อมสร้าง symbol file
    File >> create/update
    >> Create symbol file for current file
16. สร้างไฟล์แสดงแผนภาพทางเวลาของของสัญญาณ โดยให้ s1 s0 และ en มีรูปคลื่นดังรูปที่ 8
    จำลองการทำงานในโหมด Functional mode บันทึกผลการทำงานลงในกราฟรูปที่ 8

บันทึกผลการทดลอง



หมายเหตุ : x เป็นสัญญาณแบบรายคาบ Period=50 ns, duty cycle=25%

รูปที่ 8

ถ้าสัญญาณขา en = '1' และ

s1= '0' s0= '0' สัญญาณจากอินพุทถูกเลือกให้ไปที่ขาเอ้าท์พุทชื่อ y_0 (มีเพียงขาเดียวจากที่มี4ขา)

s1= '0' s0= '1' สัญญาณจากอินพุทถูกเลือกให้ไปที่ขาเอ้าท์พุทชื่อ y_1 (มีเพียงขาเดียวจากที่มี4ขา)

s1= '1' s0= '0' สัญญาณจากอินพุทถูกเลือกให้ไปที่ขาเอ้าท์พุทชื่อ y_2 (มีเพียงขาเดียวจากที่มี4ขา)

s1= '1' s0= '1' สัญญาณจากอินพุทถูกเลือกให้ไปที่ขาเอ้าท์พุทชื่อ y_3 (มีเพียงขาเดียวจากที่มี4ขา)

ถ้าสัญญาณขา en = '0' มีสัญญาณอินพุทไปปรากฏที่เอ้าท์พุทหรือไม่ ไม่มี เพราะเหตุใด

เพราะ en หรือ enable เป็นขาที่กำหนดการทำงานของ วงจรนี้ หรือง่ายๆ คือ en เสมือน สวิตช์ไฟ หาก ปิด (en="0") วงจรก็จะไม่ทำงาน

ขาสัญญาณ s1 s0 อุปกรณ์ที่ชื่อ 1to4Demux มีหน้าที่อะไร

มีหน้าที่กำหนด Input ที่ต้องการให้แสดงผลที่ output

ขาสัญญาณ en มีหน้าที่อะไร เป็นสวิตช์กำหนดการทำงานของวงจร

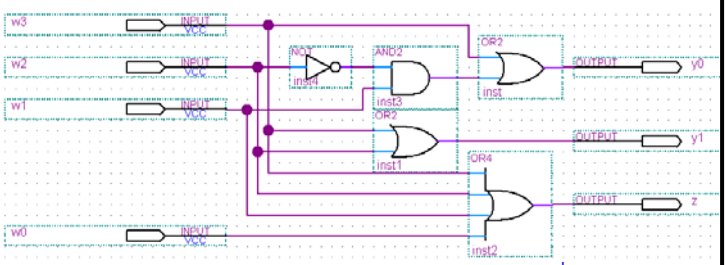
อุปกรณ์ที่ชื่อ 1to4Demux มีสมการเอ้าท์พุทคือ $y_0 = x\bar{s}_1\bar{s}_0 en$, $y_1 = x\bar{s}_1 s_0 en$

$y_2 = x s_1 \bar{s}_0 en$, $y_3 = x s_1 s_0 en$

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

การทดลองตอนที่ 4 หลักการทำงานของ 4-to-2 Priority Encoder

17. ให้ทำการ**ปิดโปรเจค**ที่สร้างมาในขั้นตอนที่ 13 – 16 ก่อนที่จะทำการทดลองต่อไป
18. สร้างโปรเจคชื่อ "4to2PrioEncode" ขึ้นมาใหม่โดย**ให้เก็บไว้ในโฟลเดอร์เดิม** เขียนวงจรในรูปที่ 9 บันทึกในไฟล์ชื่อเดียวกันกับโปรเจค ทำการคอมไพล์ให้เรียบร้อย



รูปที่ 9

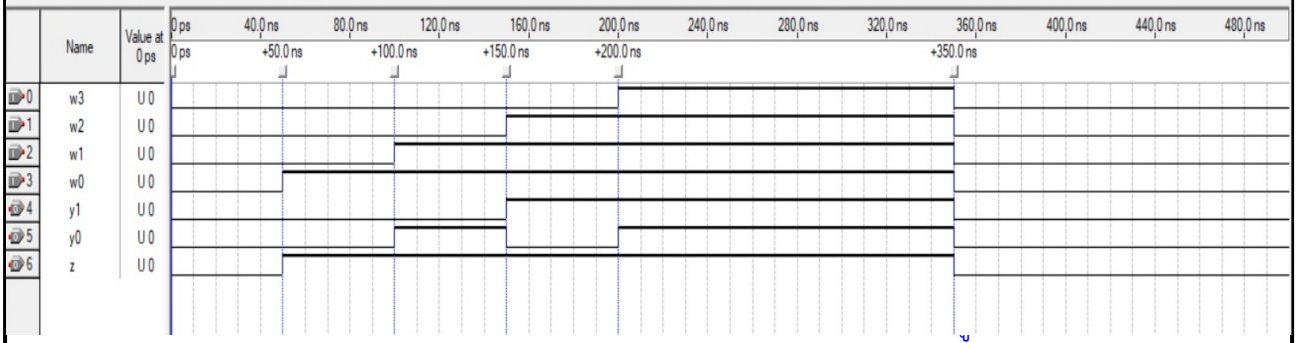
19. สร้าง symbol file เตรียมไว้ใช้งานขั้นต่อไป

File >> create/update >> Create symbol file for current file

20. สร้างไฟล์แสดงแผนภาพทางเวลาของของสัญญาณ โดยให้ w3 w2 w1 w0 มีรูปคลื่นดังรูปที่ 10 จำลองการทำงานในโหมด Functional mode บันทึกผลลงในกราฟรูปที่ 10

รูปที่ 10

บันทึกผลการทดลอง



เมื่อ w3='0',w2='0',w1='0',w0='0' ได้เอ้าท์พุท z = 0 y1= 0 y0=0 ค่าเลขฐานสิบของ y_1y_0= 0

เมื่อ w3='0',w2='0',w1='0',w0='1' ได้เอ้าท์พุท z = 1 y1= 0 y0=0 ค่าเลขฐานสิบของ y_1y_0= 0

เมื่อ w3='0',w2='0',w1='1',w0='1' ได้เอ้าท์พุท z = 1 y1= 0 y0=1 ค่าเลขฐานสิบของ y_1y_0= 1

เมื่อ w3='0',w2='1',w1='1',w0='1' ได้เอ้าท์พุท z = 1 y1= 1 y0=0 ค่าเลขฐานสิบของ y_1y_0= 2

เมื่อ w3='1',w2='1',w1='1',w0='1' ได้เอ้าท์พุท z = 1 y1= 1 y0=1 ค่าเลขฐานสิบของ y_1y_0= 3

ค่าของ z จะเป็น '1' เมื่อ มี Input ตัวใดตัวหนึ่ง หรือมากกว่า มีค่าลอจิกเป็น "1"

เป็น '0' เมื่อ ทุก Input มีค่าลอจิกเป็น "0"

ให้เปรียบเทียบทั้ง 4 กรณีข้างต้นที่มีค่าของ z = '1'

y_1y_0 (มีค่าเลขฐานสิบ) = 0 เมื่อมีอินพุทเข้าที่ขา W_0

y_1y_0 (มีค่าเลขฐานสิบ) = 1 เมื่อมีอินพุทเข้าที่ขา W_1

y_1y_0 (มีค่าเลขฐานสิบ) = 2 เมื่อมีอินพุทเข้าที่ขา W_2

y_1y_0 (มีค่าเลขฐานสิบ) = 3 เมื่อมีอินพุทเข้าที่ขา W_3

ระหว่างอินพุทขา w3, w2, w1, w0 ขาอินพุทใดที่มีความ**สำคัญมากที่สุด** W3

ขาอินพุทใดที่มีความ**สำคัญน้อยที่สุด** W0

ให้อธิบายหลักการทำงานของอุปกรณ์ Priority Encoder เป็นวงจรเข้ารหัส ที่จะจัดลำดับความสำคัญของ Input ที่เราป้อนค่า โดย Output จะแสดงผลตามลำดับความสำคัญของ Input เช่นเดียวกัน โดยที่ Input ที่มีความสำคัญมากที่สุดจะเป็นตัวกำหนดทิศทางของ Output

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

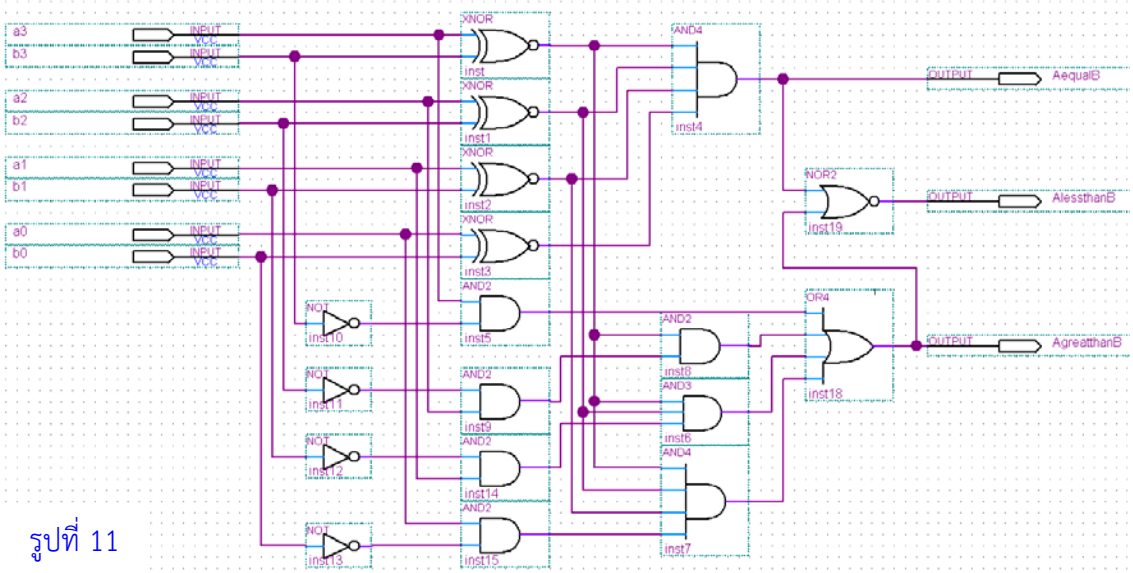
\

การทดลองตอนที่ 5 หลักการทำงานของ 4-bit Comparator Circuit

21. ให้ทำการ**ปิดโปรเจค**ที่สร้างมาในขั้นตอนที่ 17 – 20 ก่อนที่จะทำการทดลองต่อไป

22. สร้างโปรเจคชื่อ "4BitComparator" ขึ้นมาใหม่โดย**ให้เก็บไว้ในโฟลเดอร์เดิม**

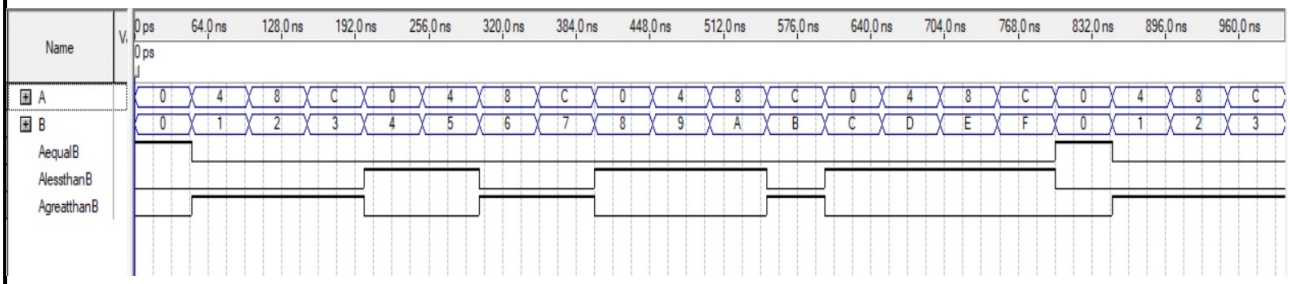
23. สร้างไฟล์ใหม่ขึ้นมาให้มีชื่อเดียวกันกับโปรเจคเพื่อเขียนวงจรในรูปที่ 11
24. คอมไพล์และสร้าง symbol file ของวงจรก่อนทำการทดลองขั้นต่อไป



รูปที่ 11

25. สร้างไฟล์แสดงแผนภาพทางเวลาดังรูปที่ 12 โดยให้ดำเนินการตามขั้นตอนต่อไปนี้
    a) กำหนดค่าของพารามิเตอร์สำหรับแสดงผลจำลองการทำงานโดยให้มีค่าดังนี้
        End Time = 1.0 us Grid Size = 1 ns
    b) จัดกลุ่มให้แสดงผลเป็นแบบ Hexadecimal โดย
        a3, a2, a1, a0 จัดเข้าเป็นกลุ่ม A ให้มีค่าเป็นแบบเลขนับ (Count value, ดูวิธีทำในหน้าถัดไป)
            เริ่มนับจาก 0 เพิ่มขึ้นครั้งละ 4 ทุกๆ 100 ns
        b3, b2, b1, b0 จัดเข้าเป็นกลุ่ม B ให้มีค่าเป็นแบบเลขนับ (Count value)
            เริ่มนับจาก 0 เพิ่มขึ้นครั้งละ 1 ทุกๆ 100 ns
26. จำลองการทำงานในโหมด Functional mode และบันทึกผลที่ได้ลงในรูปที่ 12

บันทึกผลการทดลอง



รูปที่ 12

AeqB จะมีสถานะลอจิกเป็น “1” เมื่อ A มีค่าเท่ากับ B
จะมีสถานะลอจิกเป็น “0” เมื่อ A มีค่าไม่เท่ากับ B
AltB จะมีสถานะลอจิกเป็น “1” เมื่อ A มีค่าน้อยกว่า B
จะมีสถานะลอจิกเป็น “0” เมื่อ A มีค่ามากกว่า B

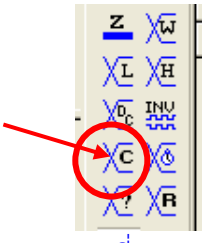
AgtB จะมีสถานะลอจิกเป็น “1” เมื่อ A มีค่ามากกว่า B

จะมีสถานะลอจิกเป็น “0” เมื่อ A มีค่าน้อยกว่า B

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

วิธีการสร้างสัญญาณแบบระบบเลขนับ (Count value) ตามการทดลองข้อ 25

1) ใช้เม้าส์เลือกสัญญาณที่ต้องการ เช่นสัญญาณ A จะปรากฏแถบสีน้ำเงินขึ้นที่รูปกราฟของสัญญาณนั้นๆ
2) จากนั้นใช้เม้าส์คลิ๊กที่แถบเครื่องมือแบบ Count ดังในรูปที่ 13
3) จะปรากฏเมนูให้ตั้งค่า สำหรับสัญญาณในรูปที่ 12 ให้กำหนดค่าต่างๆ ดังนี้



รูปที่ 13

สัญญาณอินพุท A

ที่แทป Counting ให้ Start value: 0 , Increment by: 4

ที่แทป Timing ให้ Period: 50 ns

สัญญาณอินพุท B

ที่แทป Counting ให้ Start value: 0 , Increment by: 1

ที่แทป Timing ให้ Period : 50 ns

การทดลองตอนที่ 6 ภาษา VHDL ของอุปกรณ์ที่มีฟังก์ชั่นการทำงานซับซ้อน

27. ให้เขียนภาษา VHDL ดังในรูปที่ 14 -18

a) ให้สร้างโปรเจคขึ้นใหม่ สำหรับเก็บงาน Designed ของแต่ละรูป และเก็บไฟล์ไว้ในโฟลเดอร์เดิม ไฟล์เก็บชิ้นงาน (Designed file) ให้ใช้ชื่อ *****.VHD (***** ดูคำอธิบายของในแต่ละรูป)

b) คอมไพล์และจำลองการทำงานในโหมด Functional เปรียบเทียบกับผลที่ได้จากการออกแบบโดยใช้เกทพื้นฐาน

```
1  --VHDL code for a 4-to-1 multiplexer.
2
3  LIBRARY ieee ;
4  USE ieee.std_logic_1164.all ;
5
6  ENTITY Multiplex4t1 IS
7      PORT ( w0, w1, w2, w3 : IN STD_LOGIC ;
8             s              : IN STD_LOGIC_VECTOR(1 DOWNTO 0) ;
9             f              : OUT STD_LOGIC ) ;
10 END Multiplex4t1 ;
11
12 ARCHITECTURE Behavior OF Multiplex4t1 IS
13     BEGIN
14         WITH s SELECT
15             f <=    w0  WHEN "00" ,
16                     w1  WHEN "01" ,
17                     w2  WHEN "10" ,
18                     w3  WHEN OTHERS ;
19 END Behavior ;
```

ชื่ออุปกรณ์ : 4-to-1 Multiplexer
ชื่อโปรเจค : Multiplex4t1
ชื่อไฟล์ : Multiplex4t1.vhd
ให้เปรียบเทียบผลการจำลองกับรูปที่ 4

รูปที่ 14

```
1  --VHDL code for a 2-to-4 binary decoder
2
3  LIBRARY ieee ;
4  USE ieee.std_logic_1164.all ;
5
6  ENTITY decoder2t4 IS
7      PORT ( w  : IN STD_LOGIC_VECTOR(1 DOWNTO 0) ;
8             En : IN STD_LOGIC ;
9             y  : OUT STD_LOGIC_VECTOR(0 TO 3) ) ;
10     END decoder2t4 ;
11
12 ARCHITECTURE Behavior OF decoder2t4 IS
13     SIGNAL Enw : STD_LOGIC_VECTOR(2 DOWNTO 0) ;
14     BEGIN
15         Enw <=   En & w ;
16         WITH Enw SELECT
17             y <=      "1000"  WHEN "100" ,
18                       "0100"  WHEN "101" ,
19                       "0010"  WHEN "110" ,
20                       "0001"  WHEN "111" ,
21                       "0000"  WHEN OTHERS ;
22     END Behavior ;
```

ชื่ออุปกรณ์ : 2-to-4 Binary Decoder
ชื่อโปรเจค : decoder2t4
ชื่อไฟล์ : decoder2t4.vhd
ให้เปรียบเทียบผลการจำลองกับรูปที่ 6

รูปที่ 15

```
1  --VHDL code for a Demultiplexer 1-to-4
2
3  LIBRARY ieee ;
4  USE ieee.std_logic_1164.all ;
5
6  ENTITY demultiplex1t4 IS
7      PORT ( x      :  IN   STD_LOGIC ;
8             s      :  IN   STD_LOGIC_VECTOR(1 DOWNTO 0) ;
9             y0, y1 : OUT   STD_LOGIC ;
10            y2, y3 : OUT   STD_LOGIC ) ;
11     END demultiplex1t4 ;
12
13 ARCHITECTURE Behavior OF demultiplex1t4 IS
14     BEGIN
15         y0 <=  x WHEN s="00" ELSE '0';
16         y1 <=  x WHEN s="01" ELSE '0';
17         y2 <=  x WHEN s="10" ELSE '0';
18         y3 <=  x WHEN s="11" ELSE '0';
19     END Behavior ;
```

ชื่ออุปกรณ์ : 1-to-4 Demultiplexer
ชื่อโปรเจค : demultiplex1t4
ชื่อไฟล์ : demultiplex1t4.vhd
ให้เปรียบเทียบผลการจำลองกับรูปที่ 8

รูปที่ 16

```
1  --VHDL code for a priority encoder
2
3  LIBRARY ieee ;
4  USE ieee.std_logic_1164.all ;
5
6  ENTITY priority IS
7      PORT ( w : IN  STD_LOGIC_VECTOR(3 DOWNTO 0) ;
8             y : OUT STD_LOGIC_VECTOR(1 DOWNTO 0) ;
9             z : OUT STD_LOGIC ) ;
10     END priority ;
11
12 ARCHITECTURE Behavior OF priority IS
13     BEGIN
14         y <=     "11"  WHEN w(3) = '1' ELSE
15                  "10"  WHEN w(2) = '1' ELSE
16                  "01"  WHEN w(1) = '1' ELSE
17                  "00" ;
18
19         z <= '0' WHEN w = "0000"  ELSE '1' ;
20     END Behavior ;
```

ชื่ออุปกรณ์ : 4-to-2 Priority Encoder
ชื่อโปรเจค : priority
ชื่อไฟล์ : priority.vhd
ให้เปรียบเทียบผลการจำลองกับรูปที่ 10

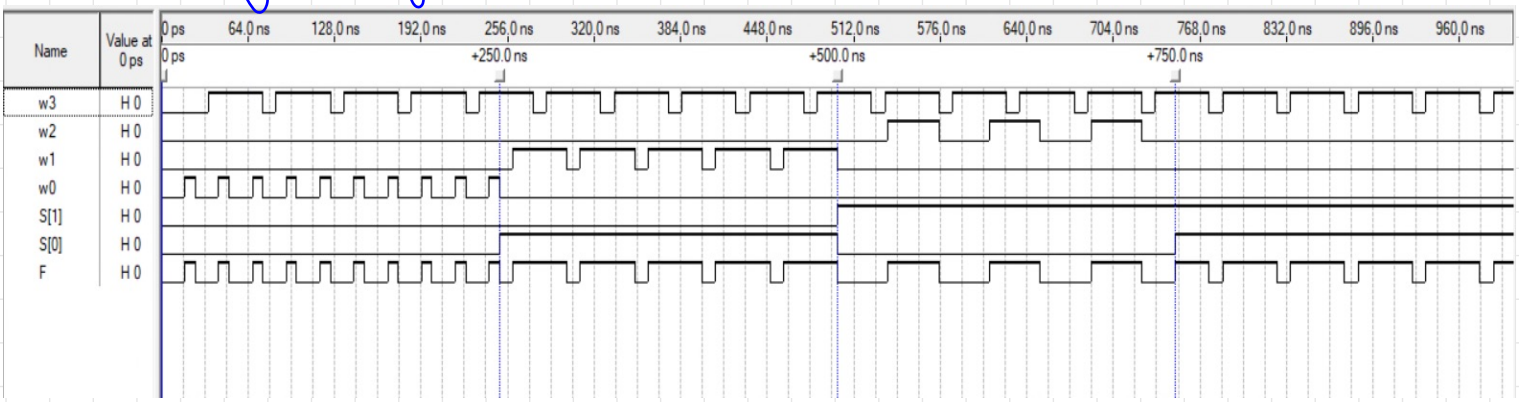
รูปที่ 17

การทดลองที่ 6 VHDL ของอุปกรณ์ที่มี function การทำงานซับซ้อน

6.1 4-to-1 Multiplexer

```
1   --VHDL code for a 4-to-1 multiplexer
2
3   LIBRARY ieee ;
4   USE ieee.std_logic_1164.all ;
5
6   ENTITY Multiplex4t1 IS
7   PORT ( w0, w1, w2, w3 : IN STD_LOGIC ;
8                         S : IN STD_LOGIC_VECTOR(1 DOWNTO 0) ;
9                         F : OUT STD_LOGIC ) ;
10  END Multiplex4t1 ;
11
12  ARCHITECTURE Behavoir OF Multiplex4t1 IS
13  BEGIN
14        WITH s SELECT
15            f <= w0 WHEN "00" ,
16                 w1 WHEN "01" ,
17                 w2 WHEN "10" ,
18                 w3 WHEN OTHERS ;
19  END Behavior ;
```

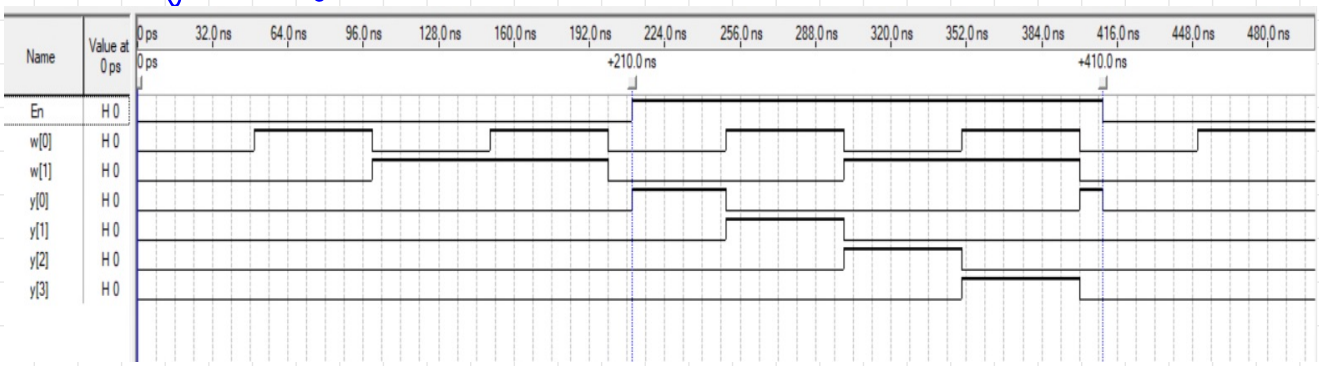
Timing Diagram



6.2 2-to-4 Binary Decoder

```
1   --VHDL code for a 2-to-4 binary decoder
2
3   LIBRARY ieee ;
4   USE ieee.std_logic_1164.all ;
5
6   ENTITY decoder2t4 IS
7       PORT (w : IN STD_LOGIC_VECTOR(1 DOWNTO 0) ;
8             En : IN STD_LOGIC ;
9             y : OUT STD_LOGIC_VECTOR(0 TO 3) ) ;
10  END decoder2t4 ;
11
12  ARCHITECTURE behavior OF decoder2t4 IS
13        SIGNAL Enw : STD_LOGIC_VECTOR(2 DOWNTO 0) ;
14        BEGIN
15          Enw <= En & w ;
16          WITH Enw SELECT
17             y <= "1000" WHEN "100" ,
18                  "0100" WHEN "101" ,
19                  "0010" WHEN "110" ,
20                  "0001" WHEN "111" ,
21                  "0000" WHEN OTHERS ;
22  END behavior ;
```

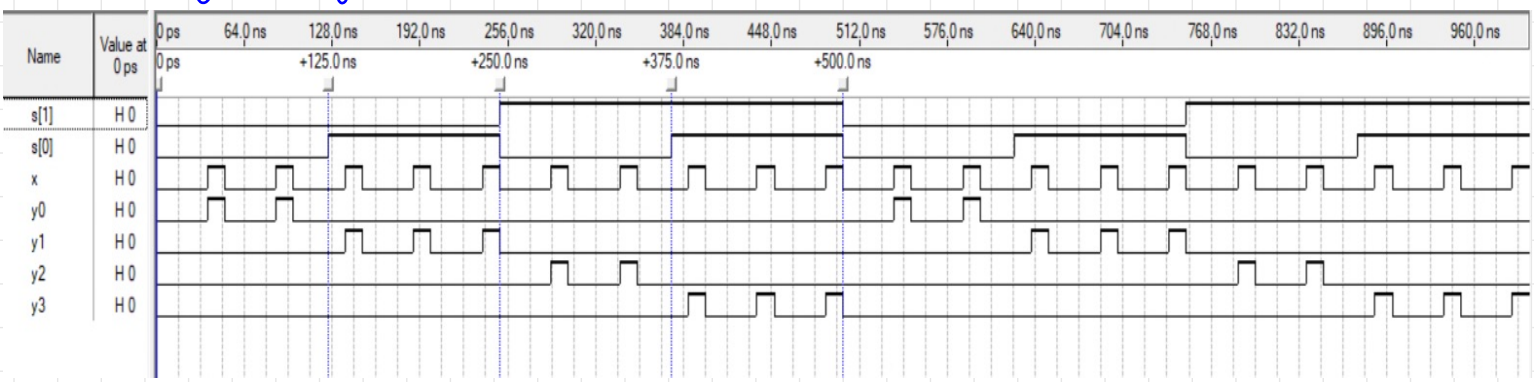
Timing diagram



6.3 1-to-4 Demultiplexer

```vhdl
-- VHDL code for a Demultiplexer 1-to-4

LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY demultiplex1t4 IS
   PORT (x     : IN STD_LOGIC ;
         s     : IN STD_LOGIC_VECTOR(1 DOWNTO 0) ;
         y0,y1 : OUT  STD_LOGIC ;
         y2,y3 : OUT STD_LOGIC ) ;

END demultiplex1t4 ;

ARCHITECTURE behavior OF demultiplex1t4 IS
    BEGIN
        y0 <= x WHEN s="00" ELSE '0' ;
        y1 <= x WHEN s="01" ELSE '0' ;
        y2 <= x WHEN s="10" ELSE '0' ;
        y3 <= x WHEN s="11" ELSE '0' ;
END behavior ;
```

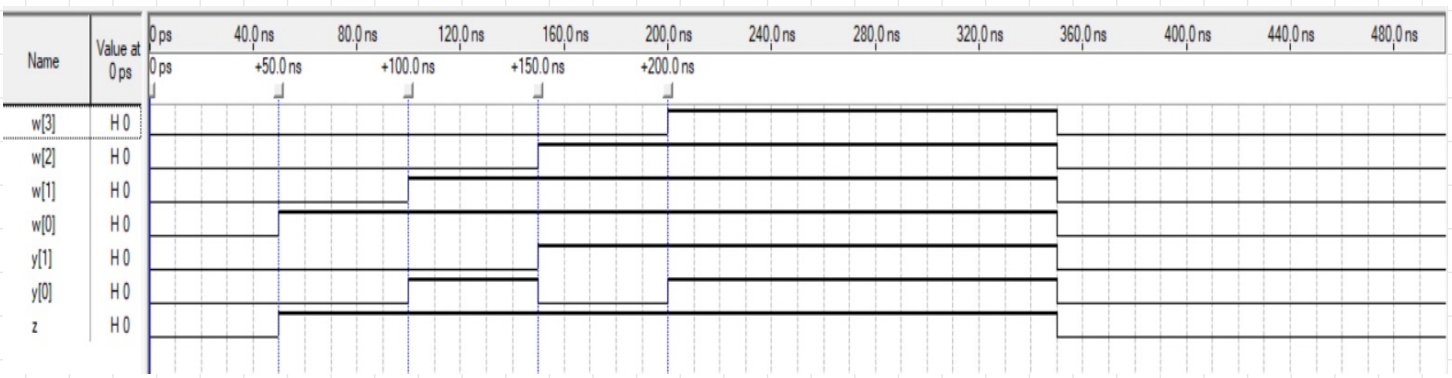
Timing Diagram



6.4 4-to-2 Priority Encoder

```vhdl
--VHDL code for a priority encoder

LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY priority IS
   PORT (w : IN STD_LOGIC_VECTOR(3 DOWNTO 0) ;
         y : OUT STD_LOGIC_VECTOR(1 DOWNTO 0) ;
         z : OUT STD_LOGIC ) ;
END priority ;

ARCHITECTURE behavior OF priority IS
   BEGIN
       y <= "11" WHEN w(3) = '1' ELSE
            "10" WHEN w(2) = '1' ELSE
            "01" WHEN w(1) = '1' ELSE
            "00" ;
       z <= '0' WHEN w="0000" ELSE '1' ;
END behavior ;
```

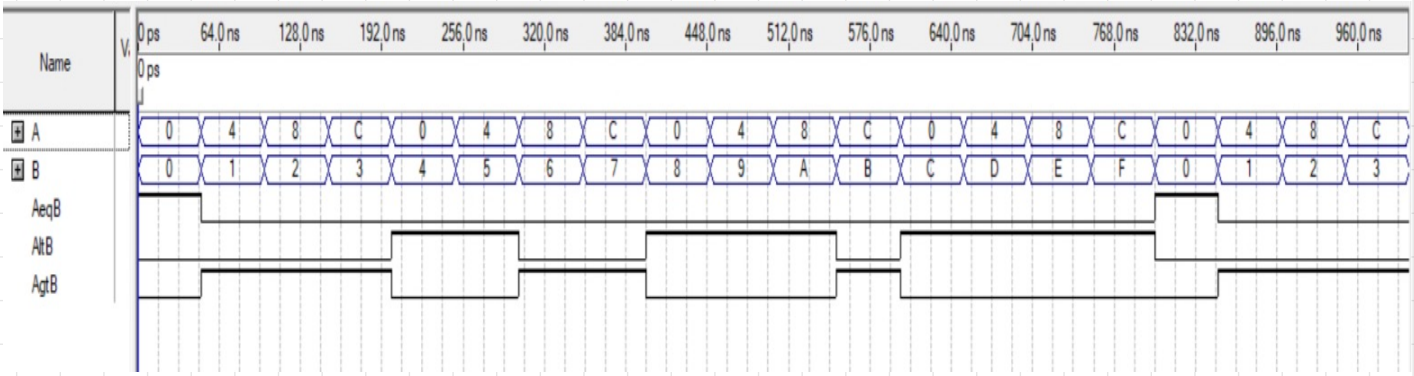
Timing Diagram



6.5 4-Bit Comparator

```
1   -- VHDL code for a four-bit comparator (unsigned)
2
3   LIBRARY ieee ;
4   USE ieee.std_logic_1164.all ;
5   USE ieee.std_logic_unsigned.all ;
6
7   ENTITY compare IS
8      PORT (A, B : IN STD_LOGIC_VECTOR(3 DOWNTO 0) ;
9            AeqB : OUT STD_LOGIC ;
10           AgtB : OUT STD_LOGIC ;
11           AltB : OUT STD_LOGIC ) ;
12  END compare ;
13
14  ARCHITECTURE Behavior OF compare IS
15      BEGIN
16           AeqB <= '1' WHEN A = B ELSE '0' ;
17           AgtB <= '1' WHEN A > B ELSE '0' ;
18           AltB <= '1' WHEN A < B ELSE '0' ;
19  END Behavior ;
20
```

Timing Diagram



```
1  -- VHDL code for a four-bit comparator (unsigned)
2
3  LIBRARY ieee ;
4  USE ieee.std_logic_1164.all ;
5  USE ieee.std_logic_unsigned.all ;
6
7  ENTITY compare IS
8      PORT ( A, B   : IN   STD_LOGIC_VECTOR(3 DOWNTO 0) ;
9             AeqB   : OUT  STD_LOGIC ;
10            AgtB   : OUT  STD_LOGIC ;
11            AltB   : OUT  STD_LOGIC ) ;
12     END compare ;
13
14 ARCHITECTURE Behavior OF compare IS
15     BEGIN
16     AeqB <=  '1' WHEN A = B ELSE '0' ;
17     AgtB <=  '1' WHEN A > B ELSE '0' ;
18     AltB <=  '1' WHEN A < B ELSE '0' ;
19 END Behavior ;
```

ชื่ออุปกรณ์ : 4-bit Comparator
ชื่อโปรเจค : compare
ชื่อไฟล์ : compare.vhd
ให้เปรียบเทียบผลการจำลองกับรูปที่ 12

รูปที่ 18

บันทึกของอาจารย์ผู้ควบคุม

โปรแกรม VHDL	ชื่ออุปกรณ์ที่ออกแบบ	ผลการทดลอง (ผ่าน/ไม่ผ่าน)	ลายเซ็นอาจารย์ผู้ควบคุม
รูปที่ 14 เทียบกับรูปที่ 4			
รูปที่ 15 เทียบกับรูปที่ 6			
รูปที่ 16 เทียบกับรูปที่ 8			
รูปที่ 17 เทียบกับรูปที่ 10			
รูปที่ 18 เทียบกับรูปที่ 12			

งานมอบหมายท้ายการทดลอง

(ให้เขียนลงบนกระดาษ A4 ที่มีเส้นบรรทัดแนบท้ายเอกสารการทดลองส่งคราวถัดไป)

1. ให้ทำรายงานสรุป เปรียบเทียบการออกแบบสร้างอุปกรณ์ดิจิทัลที่ใช้ในการทดลอง ระหว่างวิธีสร้างด้วยอุปกรณ์ลอจิกเกท และวิธีสร้างด้วยภาษา VHDL ดังนี้
   a) ชุดคำสั่ง/โครงสร้างภาษา VHDL ที่ทำพฤติกรรมเป็นอุปกรณ์ประเภท Multiplexer
   b) ชุดคำสั่ง/โครงสร้างภาษา VHDL ที่ทำพฤติกรรมเป็นอุปกรณ์ประเภท Decoder
   c) ชุดคำสั่ง/โครงสร้างภาษา VHDL ที่ทำพฤติกรรมเป็นอุปกรณ์ประเภท Demultiplexer
   d) ชุดคำสั่ง/โครงสร้างภาษา VHDL ที่ทำพฤติกรรมเป็นอุปกรณ์ประเภท Encoder
   e) ชุดคำสั่ง/โครงสร้างภาษา VHDL ที่ทำพฤติกรรมเป็นอุปกรณ์ประเภท Priority Encoder
   f) ชุดคำสั่ง/โครงสร้างภาษา VHDL ที่ทำพฤติกรรมเป็นอุปกรณ์ประเภท Comparator

พายโสภณ สุขสมบูรณ์
6201011631188 sec.7 (CSP)

งานมอบหมายท้ายการทดลอง

(ให้เขียนลงบนกระดาษ A4 ที่มีเส้นบรรทัดแนบท้ายเอกสารการทดลองส่งคราวถัดไป)

1. ให้ทำรายงานสรุป เปรียบเทียบการออกแบบสร้างอุปกรณ์ดิจิทัลที่ใช้ในการทดลอง ระหว่างวิธีสร้างด้วยอุปกรณ์ลอจิกเกท และวิธีสร้างด้วยภาษา VHDL ดังนี้
   a) ชุดคำสั่ง/โครงสร้างภาษา VHDL ที่ทำพฤติกรรมเป็นอุปกรณ์ประเภท Multiplexer
   b) ชุดคำสั่ง/โครงสร้างภาษา VHDL ที่ทำพฤติกรรมเป็นอุปกรณ์ประเภท Decoder
   c) ชุดคำสั่ง/โครงสร้างภาษา VHDL ที่ทำพฤติกรรมเป็นอุปกรณ์ประเภท Demultiplexer
   d) ชุดคำสั่ง/โครงสร้างภาษา VHDL ที่ทำพฤติกรรมเป็นอุปกรณ์ประเภท Encoder
   e) ชุดคำสั่ง/โครงสร้างภาษา VHDL ที่ทำพฤติกรรมเป็นอุปกรณ์ประเภท Priority Encoder
   f) ชุดคำสั่ง/โครงสร้างภาษา VHDL ที่ทำพฤติกรรมเป็นอุปกรณ์ประเภท Comparator

a) Multiplexer

จากการทดลองทั้งสองวิธี (ชุดคำสั่ง/VHDL) ไม่มีความแตกต่าง แต่กระบวนการได้มาซึ่ง output นั้นแตกต่างกัน VHDL เขียนง่าย เข้าใจง่าย สั้นๆได้ใจความ เช่น เราสามารถกำหนดเงื่อนไขการแสดงผลของ Output ให้เข้าใจง่าย เช่น f <= W_0 เมื่อ S_1S_0=00 แต่ถ้ามองจากชุดคำสั่งเราต้องไล่ค่าดูทีละรอบได้

b) Decoder

ชุดคำสั่ง/VHDL ไม่มีความแตกต่าง แต่การเขียนด้วย VHDL เข้าใจง่ายกว่า คำสั่งไม่ซ้ำซ้อน น่าสนใจ เมื่อ Input มีค่าหนึ่ง Output จะให้ค่าเป็น เช่น Input มีค่า 100 ค่า Output จะมีค่า 1000 แต่ถ้าเขียนด้วยชุดคำสั่งจะมีความซ้ำซ้อน เส้นโยงไปทั่ว

c) Demultiplexer

ชุดคำสั่ง/VHDL มีความแตกต่างกัน เนื่องจากการเขียนด้วย VHDL ไม่ได้กำหนดตัวแปร en ที่จะกำหนดการทำงานของวงจร ทำให้ VHDL มีการทำงานแบบอื่น

e) Priority Encoder

ชุดคำสั่ง/VHDL ไม่มีความแตกต่าง แต่การเขียนด้วยชุดคำสั่งเข้าใจยากกว่า VHDL ถ้าเราเข้าใจว่า Priority Encoder เป็นวงจรอะไร เราจะทราบการทำงานทันที เช่น

y <= "11" when w(3) = '1' else แปลว่า Output "y"
จะมีค่า 11 เมื่อ w(3)=1 โดยไม่สนใจ w(2), w(1), w(0)

e. Priority encoder

แต่ถ้าเราเขียนวงจรชุดคำสั่ง บางคนมองไม่ออก ดังนั้น VHDL มีความเข้าใจง่ายกว่า และ เขียนง่ายกว่า

F. Comparator

ชุดคำสั่ง / VHDL ให้ผลลัพธ์ไม่แตกต่างกัน แต่การเขียนด้วยชุดคำสั่ง มีความยากกว่ามาก และ ทำความเข้าใจค่อนข้างยาก เนื่องจากมี เส้นโยงกันหลายเส้น มีการเชื่อมลอจิกหลายชุด แรก ๆ จะมีความสับสน และ ผิดพลาดได้ แต่ VHDL เข้าใจง่าย เขียนง่าย ตัว Code มีความกระชับ และ เข้าใจง่ายกว่าชุดคำสั่ง