บทที่ 4

การหาค่าตรรกะของเกตในโปรแกรมวิเคราะห์การทำงานของวงจรตรรกะ

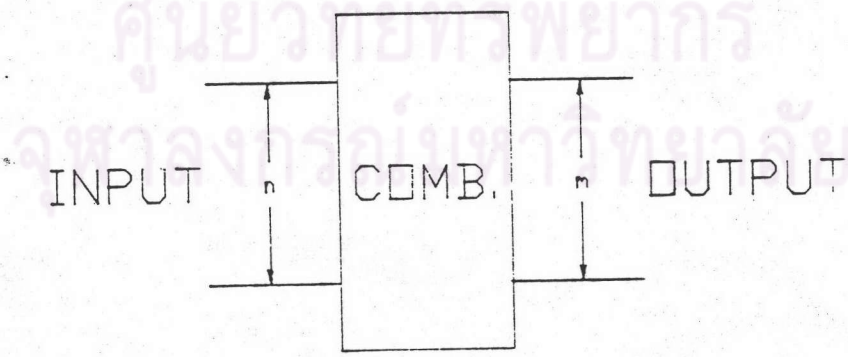
4.1 บทนำ

ในบทที่ 3 เราได้กล่าวถึงเทคนิคการวิเคราะห์วงจรตรรกะซึ่งอาศัยวิธี event driven ทำให้สามารถวิเคราะห์วงจรตรรกะอันประกอบด้วยค่าเวลาประวิงต่างกันได้ แต่การดำเนินกรรมวิธีตามหลักการข้างต้นจะต้องมีการหาค่าตรรกะออกของเกตเมื่อมีสัญญาณเข้ามาทางด้านเข้าของเกตที่เวลานั้นเสมอ บทนี้กล่าวถึงวิธีต่างๆที่จะเก็บและหาค่าตรรกะของเกต รวมถึงวิธีที่ใช้ในโปรแกรมวิเคราะห์การทำงานของวงจรตรรกะด้วย

4.2 การจำแนกชนิดของเกต

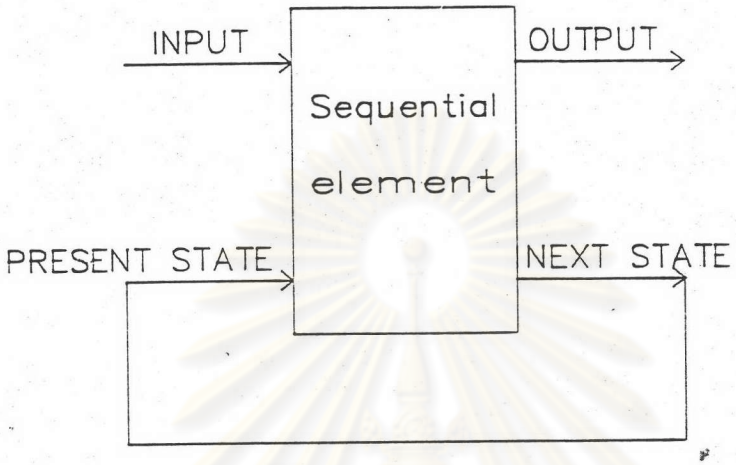
เกตโดยทั่วไปจะแบ่งออกเป็น 2 ชนิด คือ

1. เกตแบบจัดหมู่ (combinational) ซึ่งค่าตรรกะออกที่ขณะใดๆ ก็ตามขึ้นกับค่าตรรกะเข้าของเกตนั้นเท่านั้น



รูปที่ 4.1 เกตแบบจัดหมู่

2. <u>เกตแบบลำดับ</u> (sequential) ซึ่งค่าตรรกะออกของเกตนอกจากจะขึ้นกับค่าตรรกะเข้าแล้ว ยังขึ้นกับสถานะ (state) ของเกตในขณะนั้นด้วย



รูปที่ 4.2 เกตแบบลำดับ

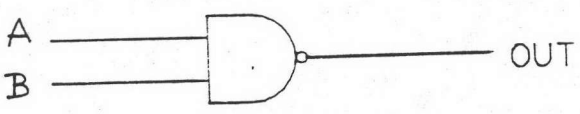
ตัวอย่างเกตจัดหมู่ ได้แก่ เกตชนิดต่างๆเช่น AND, OR, NAND และ NOR ส่วนเกตแบบลำดับได้แก่ ฟลิปฟลอป ประเภทต่างๆ เช่น D-FF, T-FF, JK-FF เป็นต้น

4.3 <u>การหาค่าตรรกะของเกตแบบจัดหมู่</u>

ในการหาค่าตรรกะของเกตแบบจัดหมู่นั้นมีวิธีการต่างๆกัน ซึ่งจะนำมากล่าวถึงพร้อมทั้งพิจารณาข้อดีข้อเสียของแต่ละวิธีดังต่อไปนี้

4.3.1 <u>การใช้ตารางค่าความจริงในการหาค่าตรรกะ</u>

เราสามารถเขียนตารางค่าความจริงของเกต ซึ่งคือความสัมพันธ์ระหว่างค่าตรรกะออกกับตรรกะเข้าทุกค่า ยกตัวอย่างเช่น เกต NAND จะมีตารางค่าความจริงดังรูปที่ 4.3



A	B	OUT
0	0	1
0	1	1
1	0	1
1	1	0

รูปที่ 4.3 ตารางค่าความจริงของเกต NAND 2 อินพุต

การหาค่าตรรกะในลักษณะนี้ จะทำโดยการเก็บตารางค่าความจริงไว้ในหน่วยความจำ และหาค่าตรรกะขาออกโดยใช้ค่าตรรกะด้านเข้าเป็นตัวชี้ไปยังตารางค่าความจริงของเกตนั้นๆ ซึ่งสามารถหาค่าตรรกะทางด้านออกได้ วิธีนี้มีข้อดีหลายประการคือ

1. มีความยืดหยุ่นมาก สามารถนำไปใช้หาค่าตรรกะของเกตจัดหมู่ชนิดใดก็ได้ซึ่งสามารถสร้างตารางค่าความจริงรวมไปถึงวงจรซึ่งมีความซับซ้อนเช่น วงจรถอดรหัส

2. การแก้ไขตารางค่าความจริงนั้นไม่ต้องมีการแก้ไขโปรแกรม ทำให้การเพิ่มเกตลงไปในโปรแกรมสามารถทำได้ง่าย

3. มีความเร็วในการทำงานสูง

ข้อเสียของวิธีดังกล่าว คือ สิ้นเปลืองหน่วยความจำมาก โดยเฉพาะกรณีซึ่งเกตมีสถานะทางตรรกะมากกว่า 2 ซึ่งจะต้องใช้เลขขนาด $\log_2(n)$ bit สำหรับแต่ละค่าอินพุต โดย n เป็นจำนวนสถานะ ที่ใช้งาน เช่นในโปรแกรมที่พัฒนาขึ้นนั้น ต้องใช้ข้อมูล 2 บิตต่อ 1 ค่าอินพุต ดังนั้นในการวิเคราะห์การทำงานของเกตซึ่งมีขั้วเข้าจำนวนมาก จะไม่สามารถทำได้ เช่น เกต AND 8 INPUT นั้น ต้องการตารางขนาด $(4^8) = 2^{16}$ หรือ 64 กิโลไบท์สำหรับเกตเพียงตัวเดียว

4.3.2 <u>การหาค่าตรรกะโดยใช้วิธีสร้างโปรแกรมย่อย</u>

วิธีนี้อาศัยการเขียนโปรแกรมย่อยขึ้นมาเพื่อจำลองแบบการทำงานของเกต แต่ละชนิด เช่น ในการวิเคราะห์การทำงานของเกต AND ก็จะมีโปรแกรมซึ่งทำการคำนวนหาค่าเอาพุตท์ของเกตนั้น ซึ่งในที่นี้ก็คือค่าตรรกะออกของเกต AND เมื่อมีการวิเคราะห์การทำงานโปรแกรมจะทำการตรวจสอบว่าเกตที่จะทำการหาค่าตรรกะนั้นเป็นเกตชนิดใด จากนั้นจะเรียกโปรแกรมย่อยที่ถูกต้องให้ทำการหาค่าตรรกะของเกตนั้น วิธีการดังที่กล่าวมานี้มีข้อดีหลายประการ คือ

1. กินเนื้อที่ในหน่วยความจำน้อยกว่าวิธีแรกในกรณีที่เกตมีขั้วอินพุตจำนวนมาก

2. สามารถเพิ่มเติมเทคนิคพิเศษบางประการ เพื่อลดเวลาการทำงานและลดเนื้อที่ในหน่วยความจำ โดยอาศัยคุณสมบัติเฉพาะของเกตนั้นๆ กรณีของเกต AND สามารถใช้วิธีนับการเกิดอินพุตที่เป็น 0 ได้ ค่าตรรกะทางเอาพุตท์จะเป็น 0 ทันทีที่ตรรกะเข้าที่ขั้วใดขั้วหนึ่งเป็น 0 ทำให้สามารถวิเคราะห์การทำงานของเกต AND มีขั้วเข้าจำนวนมากได้

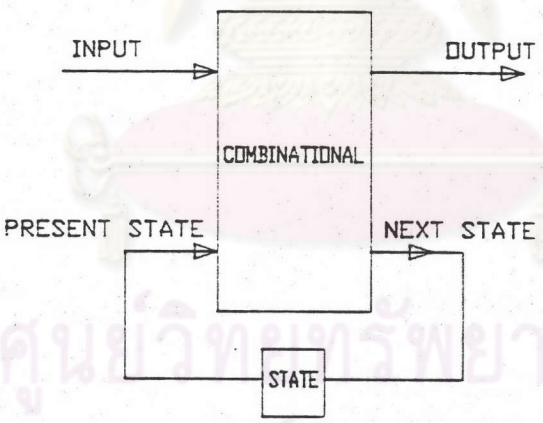
ซึ่งถ้าใช้การเก็บแบบตารางค่าความจริง จะไม่สามารถทำได้เนื่องจากต้องใช้หน่วยความจำมากจนเกินไป

ข้อเสียของวิธีนี้ก็คือ การแก้ไขเพิ่มเติมเกตใหม่ลงไปในโปรแกรมจะต้องมีการแก้ไขโปรแกรมเสมอทำให้เกิดความยุ่งยาก นอกจากนั้นเกตที่ซับซ้อนยังยากแก่การเขียนโปรแกรมให้ทำงานครอบคลุมถึงตารางค่าความจริงทุกค่าโดยสมบูรณ์อีกด้วย

เนื่องจากโปรแกรมต้นแบบที่พัฒนาขึ้นได้มุ่งเน้นในการวิเคราะห์การทำงานโดยวิธี Event Driven ดังนั้น การพัฒนาส่วนจำลองแบบการทำงานของเกตจึงได้ใช้วิธีทั้งสองแบบร่วมกัน เพื่อให้ได้โปรแกรมที่สั้นและมีประสิทธิภาพในการวิเคราะห์การทำงาน ซึ่งเทคนิคที่ใช้สำหรับเกตแต่ละชนิดจะได้กล่าวถึงในหัวข้อต่อไป

4.4 การหาค่าตรรกะของเกตแบบลำดับ

การหาค่าตรรกะของเกตแบบลำดับ นั้นจะอาศัยหลักการดังนี้



รูปที่ 4.4 แบบจำลองของเกตแบบลำดับ

เนื่องจากเกตแบบลำดับเช่น ฟลิปฟลอป นั้นค่าตรรกะออกจะขึ้นกับค่าตรรกะเข้าและสถานะปัจจุบัน ซึ่งแสดงความสัมพันธ์ไว้ดังรูปที่ 4.4 ดังนั้น ในการหาค่าตรรกะออกนั้นจะใช้เทคนิคเดียวกับที่ใช้ในเกตจัดหมู่ แต่ดัดแปลงวิธีการโดยการเพิ่มค่าตัวแปรสถานะ (state) สำหรับเกตแบบลำดับ โดยการหาค่าตรรกะออกจะนำตัวแปรสถานะนี้มาเป็นตัวแปรขาเข้าหนึ่งของเกต ผลที่ได้คือ ค่าตรรกะออกและค่าสถานะถัดไปของเกตนั้นซึ่งจะถูกนำไปเก็บให้เป็นค่าตัวแปรสถานะในเวลาต่อมา

วิธีการนี้นอกจากจะสามารถทำการวิเคราะห์การทำงานของเกต เช่น ฟลิปฟลอบ ได้แล้ว ยังใช้กับเกตที่มีความซับซ้อนขึ้นเช่น วงจรนับ วงจรเลื่อนทะเบียน ได้เป็นอย่างดี ในหัวข้อต่อไป เราจะกล่าวถึงการนำเอาหลักการต่างๆไปประยุกต์ใช้ในการพัฒนาโปรแกรมต้นแบบสำหรับใช้วิเคราะห์การทำงานของวงจรตรรกะ

4.5 การหาค่าตรรกะของเกตในโปรแกรมต้นแบบสำหรับวิเคราะห์วงจรตรรกะ

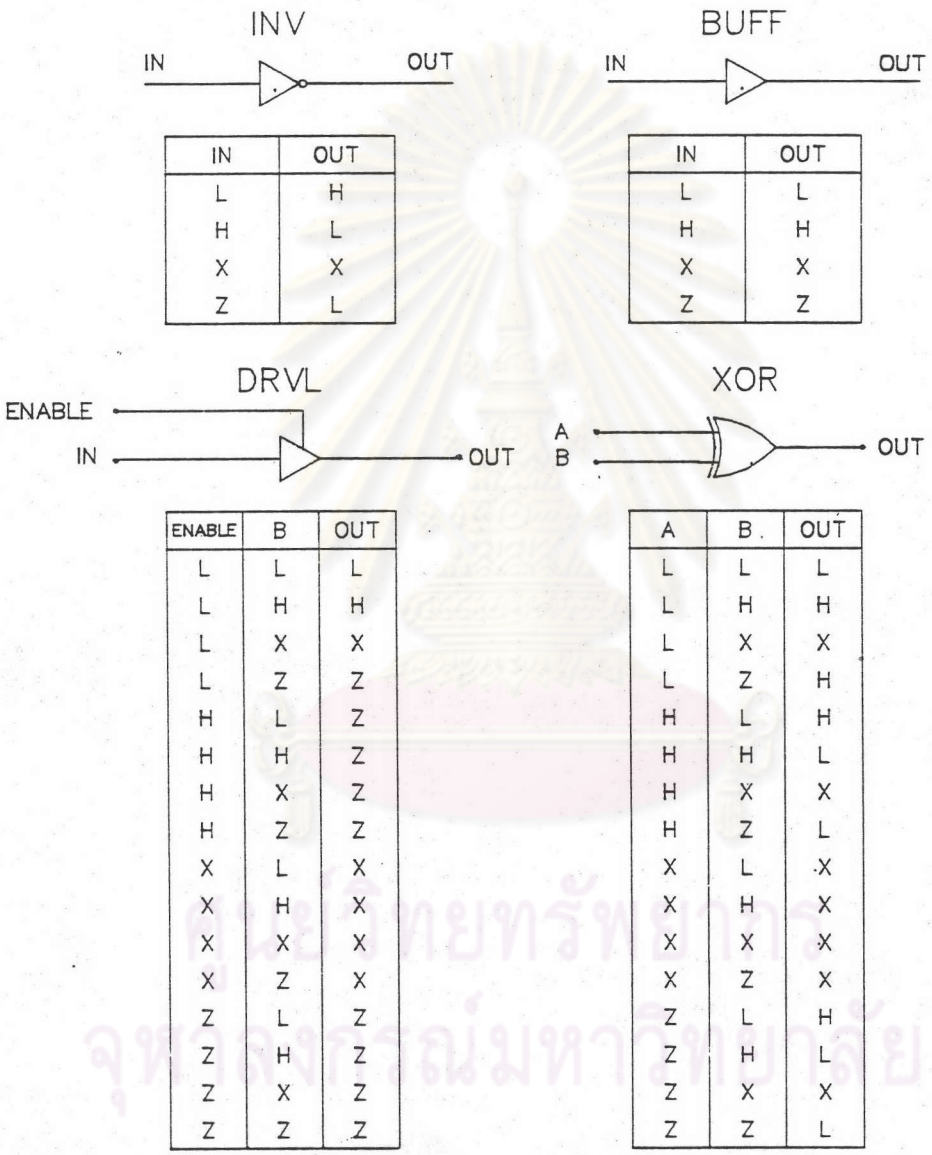
หัวข้อนี้จะได้กล่าวถึงการหาค่าตรรกะของเกตต่างๆ ที่มีในโปรแกรมต้นแบบสำหรับวิเคราะห์การทำงานของวงจรตรรกะซึ่งพัฒนาขึ้น โดยแบ่งออกได้เป็น 4 ประเภทด้วยกันคือ

1. เกตแบบจัดหมู่ทั่วไป
2. เกตแบบลำดับทั่วไป
3. เกตแบบจัดหมู่เฉพาะ
4. เกตแบบลำดับเฉพาะ

4.5.1 เกตแบบจัดหมู่ทั่วไป

การหาค่าตรรกะของเกตในประเภทนี้ จะอาศัยการเปิดตารางค่าความจริงในหน่วยความจำเป็นหลัก เนื่องจากโปรแกรมทำงานโดยมีสถานะทางตรรกะ 4 สถานะด้วยกัน คือ HIGH, LOW, UNKNOWN และ HIGH IMPEDANCE ดังนั้นเราต้องใช้จำนวนข้อมูล 2 บิทในการแทนค่าตรรกะ ซึ่งกำหนดดังนี้ คือ L(LOW)=00 H(HIGH)=01 X(UNKNOWN)=10 และ Z(HIGH IMPEDANCE)=11 (จำนวนบิท n = $\log_2$ (จำนวนสถานะ))

เกตซึ่งจัดอยู่ในประเภทนี้ได้แก่ เกต XOR, เกต DRVL(Driver with active low enable) เกต INV(Inverter) และ เกต BUFF(Buffer) ซึ่งมีตารางค่าความจริงดังรูปที่ 4.5



IN	OUT
L	H
H	L
X	X
Z	L

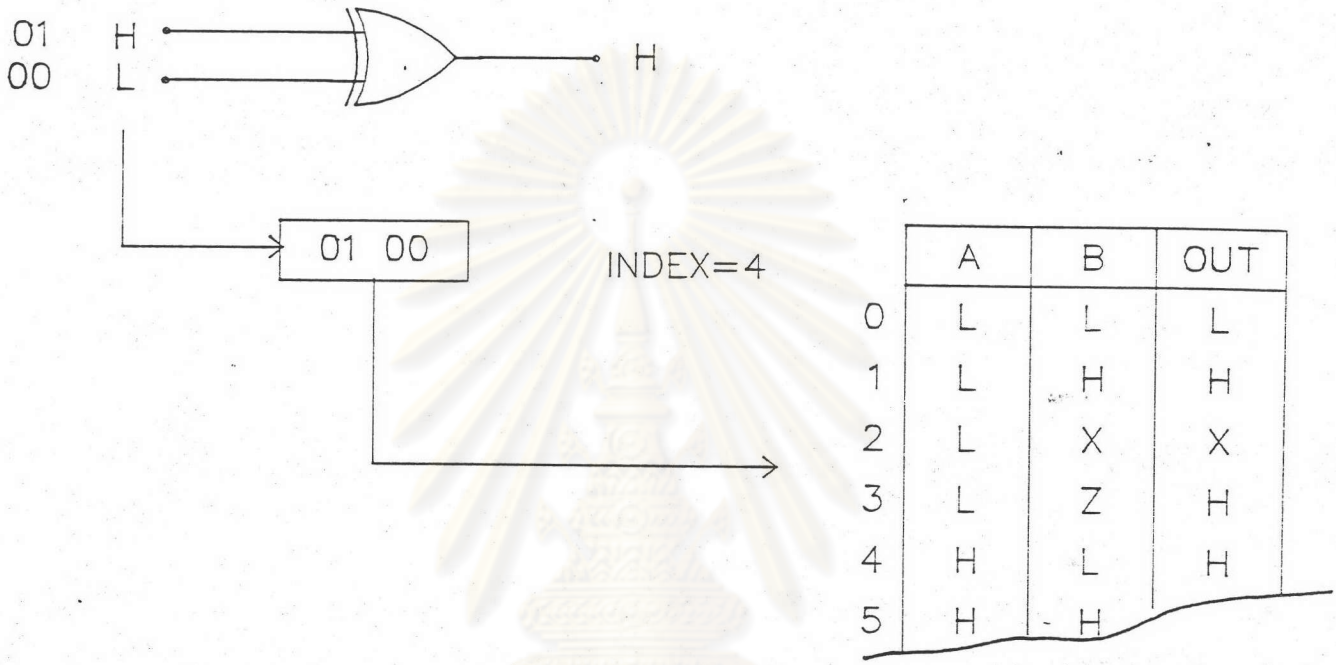
IN	OUT
L	L
H	H
X	X
Z	Z

ENABLE	B	OUT
L	L	L
L	H	H
L	X	X
L	Z	Z
H	L	Z
H	H	Z
H	X	Z
H	Z	Z
X	L	X
X	H	X
X	X	X
X	Z	X
Z	L	Z
Z	H	Z
Z	X	Z
Z	Z	Z

A	B	OUT
L	L	L
L	H	H
L	X	X
L	Z	H
H	L	H
H	H	L
H	X	X
H	Z	L
X	L	X
X	H	X
X	X	X
X	Z	X
Z	L	H
Z	H	L
Z	X	X
Z	Z	L

รูปที่ 4.5 แสดงตารางค่าความจริงของเกตจัดหมู่ INV, BUFF, DRVL, XOR

เกตจัดหมู่แบบทั่วไปนี้จะมีตารางค่าความจริงเก็บอยู่ภายในหน่วยความจำ การหาค่าตรรกะออกของเกตเริ่มโดยการหาตำแหน่งของตารางได้จากโมเดลของเกต จากนั้นหาค่าตัวชี้ไปยังตำแหน่งในตารางซึ่งเก็บค่าตรรกะด้านออกไว้ ตัวชี้นี้สามารถคำนวนได้จากตรรกะเข้า ยกตัวอย่างเช่น ในรูป 4.6 ซึ่งแสดงการหาค่าตรรกะออกของเกต XOR



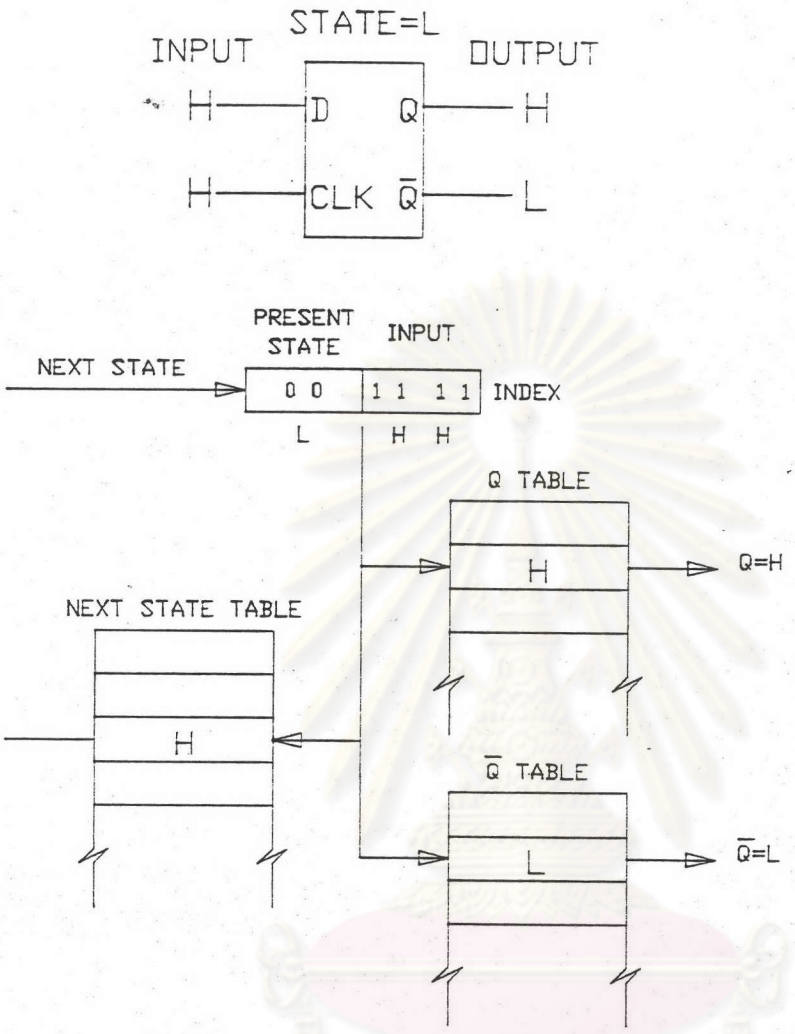
	A	B	OUT
0	L	L	L
1	L	H	H
2	L	X	X
3	L	Z	H
4	H	L	H
5	H	H	

รูป 4.6 การหาค่าตรรกะของเกต XOR

การคำนวนหาค่าของตัวชี้ทำโดยการเรียงค่าตรรกะเข้าทั้งหมดเข้าด้วยกันซึ่งในการใช้เทคนิคแบบนี้สามารถขยายให้ใช้ได้กับเกตจำนวนกี่อินพุตก็ได้

4.5.2 <u>เกตแบบลำดับทั่วไป</u>

หลักการในการหาค่าตรรกะออกของเกตแบบลำดับทั่วไปซึ่งได้กล่าวถึงในหัวข้อ 4.4 นั้น ไม่ได้แตกต่างไปมากนักจากการหาค่าตรรกะของเกตจัดหมู่แบบทั่วไปมากนัก เพียงแต่ในการคำนวนหาตัวชี้นั้นจะต้องเอาค่าของตัวแปรสถานะซึ่งเก็บอยู่ในเกตนั้น มาเป็นตรรกะด้านเข้าหนึ่งของเกตด้วย ผลจากการหาค่าตรรกะจะได้ค่าตรรกะออกและค่าสถานะถัดไปซึ่งจะนำไปเก็บไว้ในตัวแปรสถานะของเกตอีกทีหนึ่ง การทำงานจะเป็นไปดังที่แสดงในรูป 4.7



รูป 4.7 แสดงการหาค่าตรรกะของเกตแบบลำดับทั่วไป

ในโครงสร้างข้อมูลของเกตแต่ละประเภทจะมีการเก็บค่าตำแหน่งเริ่มต้นของตารางค่าความจริงของแต่ละขั้วออกไว้ รวมทั้งตำแหน่งเริ่มต้นของตารางบอกค่าของสถานะถัดไปซึ่งขึ้นกับสถานะปัจจุบันและตรรกะเข้า รายละเอียดโครงสร้างข้อมูลดังกล่าวจะกล่าวถึงในบทที่ 5 เกตประเภทนี้ได้แก่ T-FF, D-FF และ JK-FF

4.5.3 <u>เกตจัดหมู่เฉพาะ</u>

การวิเคราะห์การทำงานของเกตที่มีขั้วเข้าจำนวนมากโดยใช้วิธีแบบทั่วไปนั้น จะเกิดปัญหาในเรื่องของหน่วยความจำซึ่งกล่าวถึงไว้ในหัวข้อ 4.3.1 การพัฒนาโปรแกรมต้นแบบสำหรับวิเคราะห์การทำงานของวงจรตรรก จึงได้มีการประยุกต์ใช้เทคนิคในหัวข้อ 4.3.2

คือ สร้างโปรแกรมย่อยสำหรับเกตบางประเภทที่มีจำนวนขั้วเข้าจำนวนมาก เช่น เกต AND, OR, NAND, NOR และ BUS ผลที่ได้ทำให้ สามารถวิเคราะห์การทำงานของเกตนั้นได้ ซึ่งจะไม่สามารถทำได้โดยใช้การเปิดตารางค่าความจริง เช่นเดียวกับเกตจัดหมู่แบบทั่วไป ในการหาค่าตรรกะของเกตดังกล่าวทำได้โดยเทคนิคดังนี้

1. <u>เกต AND</u> ทำโดยการตวจสอบที่ขั้วเข้า ถ้ามีสัญญาณที่ขั้วเข้าใดก็ตามเป็น LOW จะได้ตรรกะด้านออกเป็น LOW ถ้าเป็น UNKNOWN เพียงขั้วเดียวค่าตรรกะออกจะเป็น UNKNOWN ด้วย กรณีอื่นจะได้ตรรกะด้านออกเป็น HIGH ซึ่งก็คือ กรณีที่ขั้วเข้าทุกขั้วเป็นตรรกะ HIGH หมด ขั้นตอนแสดงในรูป 4.8

```
output = HIGH;
for ( all input pins )
begin
  if ( input is LOW or UNKNOWN) then
    begin
      output = input;
      break;
    end;
end;
```

รูป 4.8 การหาค่าตรรกะของเกต AND

2. <u>เกต NAND</u> ใช้เทคนิคเช่นเดียวกับเกต AND หากแต่ค่าตรรกะออกจะกลับกับเกต AND

3. <u>เกต OR</u> ทำโดยการตรวจสอบที่ขั้วเข้า ถ้ามีสัญญาณที่ขั้วเข้าใดก็ตามเป็น HIGH จะได้ตรรกะด้านออกเป็น HIGH ถ้าเป็น UNKNOWN เพียงขั้วเดียวค่าตรรกะออกจะเป็น UNKNOWN ด้วย กรณีอื่นจะได้ตรรกะด้านออกเป็น LOW ขั้นตอนดังกล่าวแสดงไว้ในรูปที่ 4.9

```
output = LOW;
for (all input pins)
begin
        if (input is HIGH or UNKNOWN)
         begin
             output=input;
             break;
         end;
end;
```

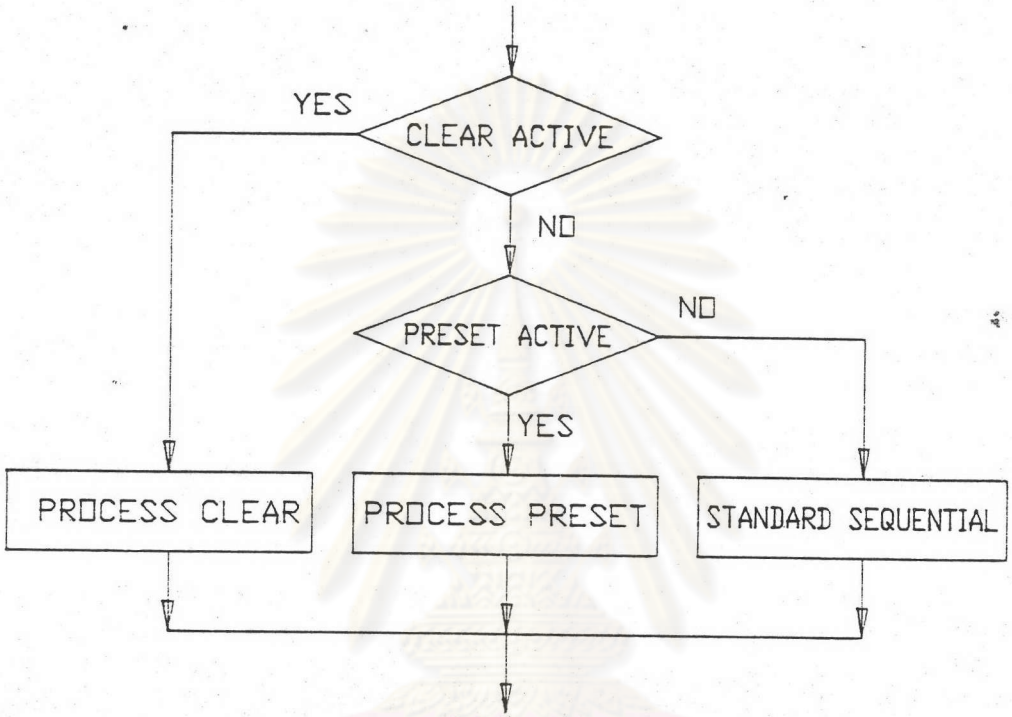
รูปที่ 4.9 การหาค่าตรรกะของเกต OR

4. <u>เกต NOR</u> ขั้นตอนเช่นเดียวกับเกต OR เพียงแต่ค่าตรรกะทางออกที่ได้จะกลับกับ เกต OR

5. <u>เกต BUS</u> เป็นเกตพิเศษที่ใช้แทนการต่อขั้วออกมารวมกันในลักษณะของบัส ซึ่งจะแยกกล่าวในรายละเอียดภายหลังในเรื่องเกี่ยวกับการตรวจสอบการเกิด bus contention

4.5.4 <u>เกตลำดับแบบเฉพาะ</u>

เกตแบบลำดับซึ่งได้บรรจุไว้ในโปรแกรมอันได้แก่ T-FF, D-FF และ JK-FF นั้นไม่มีขั้ว preset และ clear เพราะในการเพิ่มขั้ว preset และ clear นั้น จะทำให้ตารางค่าความจริงมีขนาดใหญ่ขึ้นถึง $(4)^2$ หรือ 16 เท่าสำหรับเกตนั้น โปรแกรมจึงได้แยกประเภทเกตนี้ออกมาและทำการตรวจสอบค่าสัญญาณทั้งสองขั้วนั้นก่อน หากมีการ preset หรือ clear เกิดขึ้น ก็จะได้ค่าตรรกะออกและสถานะที่ตั้งไว้ แต่ในกรณีที่ไม่มีการ preset หรือ clear ก็จะทำการหาค่าตรรกะทางด้านออกด้วยวิธีเดียวกันกับเกตแบบลำดับทั่วไป เกตประเภทนี้ในโปรแกรมได้แก่ ฟลิปฟลอบชนิด JK, T และ D ซึ่งมีขั้ว preset และ clear ขั้นตอนของการตรวจสอบแสดงไว้ในรูปที่ 4.10

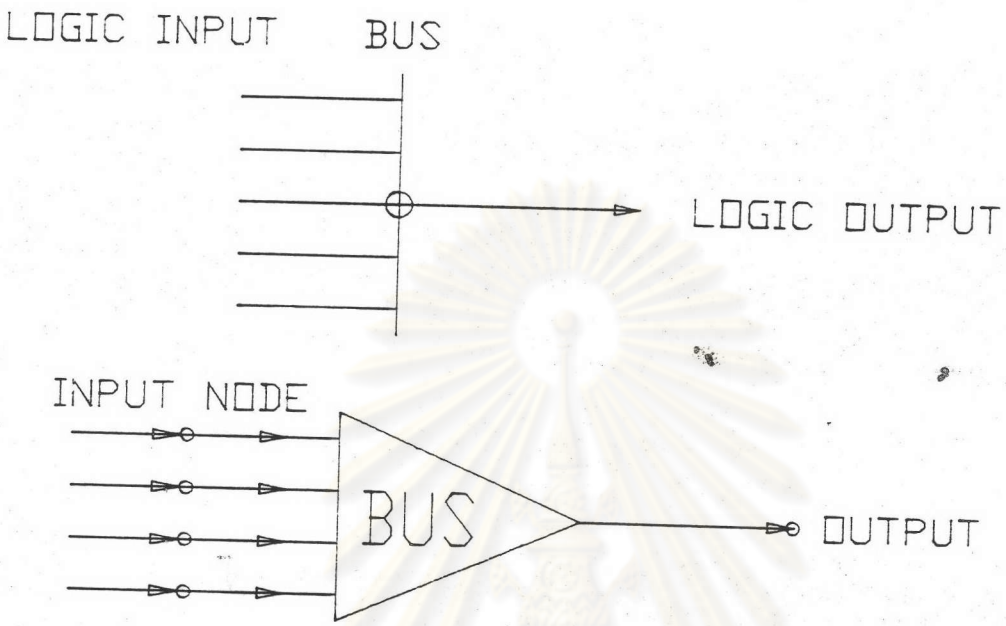


4.10 การตรวจสอบสัญญาณ preset และ clear

4.6 <u>การตรวจสอบเงื่อนไขการเกิด bus contention</u>

ในการวิเคราะห์การทำงานวงจรตรรกะซึ่งมีการต่อในลักษณะของบัส เช่นระบบไมโครโปรเซสเซอร์นั้น การที่จะทราบค่าตรรกะที่แน่นอนของบัสจะขึ้นกับเงื่อนไขที่ว่าต้องไม่มีอินพุตซึ่งขับบัสอยู่มีค่าตรรกะเป็น HIGH หรือ LOW พร้อมกันมากกว่า 1 อินพุต โดยปกติแล้วจะมีอินพุตที่ขับบัสอยู่เพียงสัญญาณเดียว ส่วนที่เหลือจะมีค่าตรรกะเป็น HIGH IMPEDANCE หมด

ในโปรแกรมวิเคราะห์การทำงานของวงจรตรรกะนี้ จะใช้วิธีกำหนดให้บัสเป็นเกตชนิดหนึ่งซึ่งทำให้สะดวกในการตรวจสอบเงื่อนไขดังที่ได้กล่าวมาข้างต้น



รูปที่ 4.11 เกตชนิดบัส

ในการตรวจสอบเงื่อนไขจะใช้เทคนิคโปรแกรมย่อย โดยเมื่อมีการวิเคราะห์การทำงานของบัส จะมีการเรียกใช้โปรแกรมย่อยมาตรวจสอบเงื่อนไขของบัสโดยการทดสอบสัญญาณเข้าทุกขั้วที่เข้ามายังบัส ถ้ามีสัญญาณทำงาน มากกว่า 1 สัญญาณขึ้นไป โปรแกรมย่อยก็จะรายงานการเกิด bus contention และให้ค่าตรรกะด้านออกเป็น UNKNOWN ขั้นตอนในการตรวจสอบจะเป็นไปดังรูป 4.12

```
BUS CONTENSION TEST:
        count = 0;
        for (all input to bus )
         begin
            if (input state is not HIGH IMPEDANCE) then
            begin
                    output = input;
                    increment count;
            end;
            if (count > 1)      /*  BUS CONTENSION detected */
            begin
                    output  = UNKNOWN;
                    status  = BUS CONTENSION;
            end
            else
                    status = NORMAL;
         end;
```

รูป 4.12 การประมวลผลเกตประเภทบัส