

ลอจิกเกต (Logic Gate) เป็นตัวดำเนินการทางตรรกศาสตร์ ซึ่งจะรับข้อมูลเข้าอย่างน้อยหนึ่งตัวมา ประมวลผลทางตรรกะ แล้วส่งข้อมูลออกหนึ่งตัว โดยจะใช้เป็นวงจรคิจิตอลซึ่งจะมีสภาพการทำงาน 2 สภาวะ คือ สูง (High) และ ต่ำ (Low) ในทางไฟฟ้าสามารถแทนการทำงานด้วยระดับแรงดันไฟฟ้า คือ ที่เป็นลอจิก 1 แทนด้วยระดับแรงดันไฟสูง และที่เป็นลอจิก 0 แทนด้วยระดับแรงดันไฟต่ำ เช่นถ้าวงจรที่ใช้ไฟฟ้าปกติเป็นไฟ 5 โวลต์ ถ้าล็อกิจ 1 จะดีอ้วว่าเท่ากับ 5 โวลต์ และล็อกิจ 0 จะเป็น 0 โวลต์ ในระบบสมัยใหม่มักจะพยายามลดระดับแรงดันไฟฟ้าที่ใช้ให้น้อยลงเพื่อลดการใช้พลังงานให้น้อยลง

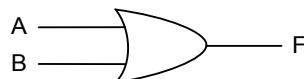
ส่วนประกอบพื้นฐานในวงจรคิจิตอลจะประกอบด้วย ลอจิกเกต (Logic Gate) ซึ่งจะทำงานด้วยระบบเลขฐานสอง (Binary) เป็นวงจรอิเล็กทรอนิกส์ที่ใช้ระดับไฟเป็นตัวแปรทางลอจิก ในการทำงานของวงจรที่สัญญาณเข้าและสัญญาณออก โดยจะแสดงเป็นพีชคณิตบูลีน (Boolean algebra) และตารางความจริง (Truth Table) ซึ่งมีเพียง 2 ค่า คือจริง (True) และ เท็จ (False) และในทางวงจรล็อกิกจะมีค่าเป็น Logic 1 และ Logic 0 เมื่อกำหนดให้ตัวแปรสัญญาณทางด้านเข้า (Input) เป็น A, B และให้ผลลัพธ์ของสัญญาณทางด้านขาออก (Output) เป็น F เกตที่ใช้มีล็อกิกพื้นฐาน 3 แบบดังนี้

AND Gate เป็นเกตที่มีอินพุตตั้งแต่สองเส้นขึ้นไป มีความหมายเดียวกับตรรกะ "และ" โดยวงจรจะได้สภาวะล็อกิกทางด้านเอาท์พุตเป็น 1 ก็ต่อเมื่ออินพุตที่เข้ามาทั้งหมดเป็น 1 เรียกว่า การคูณทางล็อกิก มีสัญลักษณ์เป็นเครื่องหมายคูณ (\cdot) เทียบการทำงานของเกตอยู่ในรูปพีชคณิตบูลีน ได้สมการล็อกิกทางด้านเอาท์พุตคือ $F = A \cdot B$ ซึ่งสามารถเขียนรูปสัญลักษณ์และมีตารางค่าความจริง (Truth Table) ได้ดังนี้



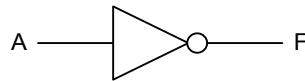
Input		Output
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate เป็นเกตที่มีอินพุตตั้งแต่สองเส้นขึ้นไป มีความหมายเดียวกับตรรกะ "หรือ" โดยวงจรจะได้สภาวะล็อกิกทางด้านเอาท์พุตเป็น 1 ก็ต่อเมื่ออินพุตที่เข้ามาตัวใดตัวหนึ่งเป็น 1 เรียกว่า การบวกทางล็อกิก มีสัญลักษณ์เป็นเครื่องหมายบวก ($+$) เทียบเป็นสมการล็อกิกทางด้านเอาท์พุตได้คือ $F = A + B$ ซึ่งสามารถเขียนรูปสัญลักษณ์และมีตารางค่าความจริง (Truth Table) ได้ดังนี้



Input		Output
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

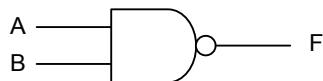
NOT Gate หรือ Inverter เป็นเกตที่มีอินพุตเพียงสีนเดียว ทางด้านเอาท์พุตจะได้สภาวะลอจิกตรงข้ามกับอินพุตที่เข้ามา เรียกว่า การคอมพลีเม้นต์ (Complement) ทางลอจิก โดยมีสัญลักษณ์เป็นเครื่องหมายชี้ดับ (bar) บนตัวแปร (\neg) เนียนเป็นสมการลอจิกทางด้านเอาท์พุตได้คือ $F = \overline{A}$ ซึ่งสามารถเขียนรูปสัญลักษณ์และมีตารางค่าความจริง (Truth Table) ได้ดังนี้



Input	Output
A	F
0	1
1	0

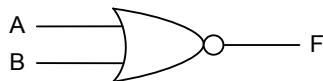
จากเกตพื้นฐานที่ใช้งานทั้ง 3 แบบ สามารถนำมาผสานรวมกันได้เกตเพิ่มเติมที่ใช้โดยทั่วไปดังนี้

NAND Gate เป็นเกตที่มีอินพุตตั้งแต่สองสีนขึ้นไป จะได้สภาวะลอจิกทางด้านเอาท์พุตเป็น 0 ก็ต่อเมื่ออินพุตที่เข้ามาทั้งหมดเป็น 1 เนียนเป็นสมการลอจิกทางด้านเอาท์พุตได้คือ $F = \overline{A \cdot B}$ ซึ่งสามารถเขียนรูปสัญลักษณ์และมีตารางค่าความจริง (Truth Table) ได้ดังนี้



Input		Output
A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

NOR Gate เป็นเกตที่มีอินพุตตั้งแต่สองสีนขึ้นไป จะได้สภาวะลอจิกทางด้านเอาท์พุตเป็น 0 ก็ต่อเมื่ออินพุตที่เข้ามาตัวใดตัวหนึ่งเป็น 1 เนียนเป็นสมการลอจิกทางด้านเอาท์พุตได้คือ $F = \overline{A + B}$ ซึ่งสามารถเขียนรูปสัญลักษณ์และมีตารางค่าความจริง (Truth Table) ได้ดังนี้

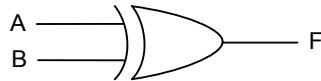


Input		Output
A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

XOR Gate หรือ Exclusive OR Gate จะได้สภาวะลอจิกทางด้านเอาท์พุทเป็น 1 ก็ต่อเมื่ออินพุทที่เข้ามาทั้งสองตัวมีสภาวะต่างกัน เนียนเป็นสมการลอจิกทางด้านเอาท์พุทได้คือ

$$F = A \cdot \bar{B} + \bar{A} \cdot B = A \oplus B$$

ซึ่งสามารถเขียนรูปสัญลักษณ์ และมีตารางค่าความจริง (Truth Table) ได้ดังนี้



Input		Output
A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

XNOR Gate หรือ Exclusive NOR Gate จะได้สภาวะลอจิกทางด้านเอาท์พุทเป็น 1 ก็ต่อเมื่ออินพุทที่เข้าทั้งสองตัวมีสภาวะเหมือนกัน เนียนเป็นสมการลอจิกทางด้านเอาท์พุทได้คือ

$$F = \overline{A \cdot \bar{B}} + \overline{\bar{A} \cdot B}$$

โดยใช้กฎเดอมอร์แกน (De Morgan's laws) จะได้

$$F = \overline{(A \cdot \bar{B})} \cdot \overline{(\bar{A} \cdot B)}$$

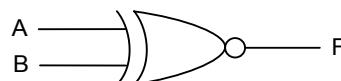
$$F = (\bar{A} + B) \cdot (A + \bar{B})$$

$$F = \bar{A} \cdot A + A \cdot B + \bar{A} \cdot \bar{B} + B \cdot \bar{B}$$

$$F = A \cdot B + \bar{A} \cdot \bar{B}$$

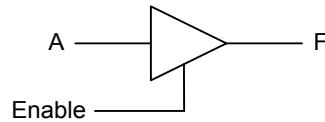
$$F = \overline{A \oplus B}$$

ซึ่งสามารถเขียนรูปสัญลักษณ์ และมีตารางความจริง (Truth Table) ได้ดังนี้



Input		Output
A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

Tri-State Buffer จะได้สภาวะล็อกิจิกจากอินพุทออกไปทางด้านเอาท์พุท ก็ต่อเมื่อขา Enable มีค่าเป็น 1 และในกรณีที่ขา Enable มีค่าเป็น 0 จะได้สภาวะล็อกิจิกทางด้านเอาท์พุทเป็น Z เรียกว่า Hi Impedance คือไม่มีข้อมูลออก เปรียบเสมือนไม่ได้ต่อสายสัญญาณหรือสายสัญญาณของวงจรถูกตัดขาด ดังนั้นสภาวะล็อกิกทางด้านเอาท์พุทสามารถเป็นได้สามสถานะคือ HIGH, LOW, และ Hi Impedance ซึ่งสามารถเขียนรูปสัญลักษณ์และมีตารางค่าความจริง (Truth Table) ได้ดังนี้



Input		Output
A	Enable	F
0	1	0
1	1	1
0	0	Z
1	0	Z

การทดลองเรื่อง วงจรลอจิกเกต

จุดประสงค์

1. เพื่อศึกษาการทำงานของล็อกจิกเกต
2. เพื่อให้สามารถวิเคราะห์การทำงานของวงจรดิจิตอล
3. เพื่อให้สามารถออกแบบวงจรลอจิกได้

เครื่องมือ

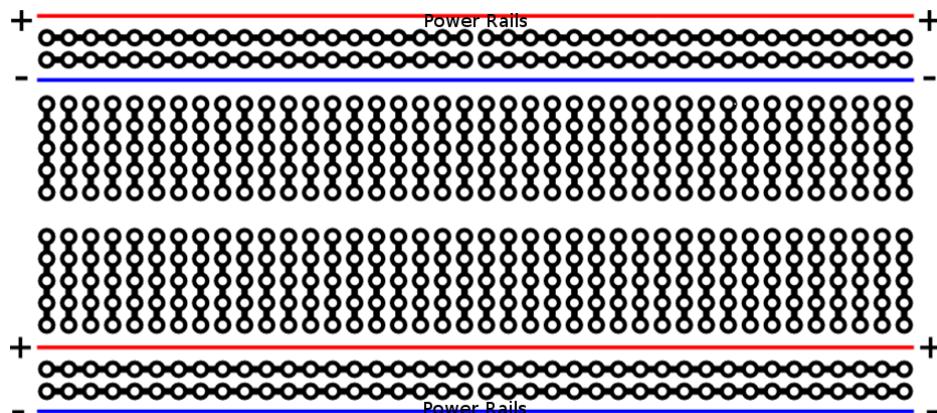
- | | |
|----------------------------|-----------|
| 1. มิเตอร์ | 1 เครื่อง |
| 2. แหล่งจ่ายไฟกระแสตรง 5 V | 1 เครื่อง |

อุปกรณ์

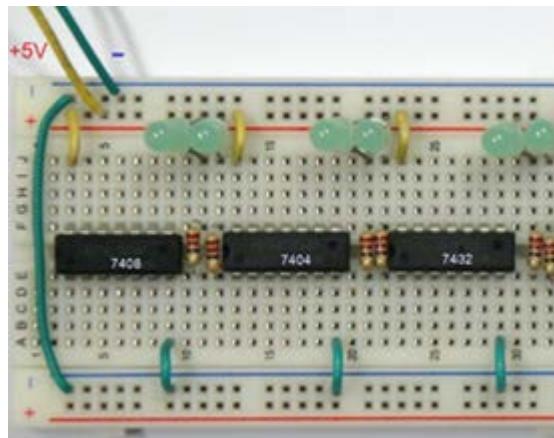
- | | |
|------------------------------------|--------|
| 1. ไอซี 7404 , 7408 , 7432 | 1 ตัว |
| 2. LED | 10 ดวง |
| 3. ตัวต้านทาน 220 โอห์ม 0.25 วัตต์ | 10 ตัว |

การทดลอง

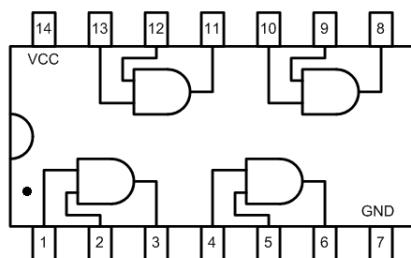
การต่อวงจรที่จะใช้ในการทดลองจะใช้ Protoboard (Protoboard) หรือ Breadboard ลักษณะเป็นแผ่นพลาสติกหนาด้านบนของแผ่นจะมีรูเรียงกันภายในรูมีตัวนำไฟฟ้าที่เชื่อมต่อกันในรูปแบบที่มีกำหนดไว้ดังภาพ การทดลองจะต้องเสียบขาของอุปกรณ์อิเล็กทรอนิกส์ลงไปให้ด้านนำภายในช่องต่อวงจร และใช้สายไฟเสียบลงรูเพื่อเชื่อมต่อวงจรไฟฟ้าให้ถึงกัน



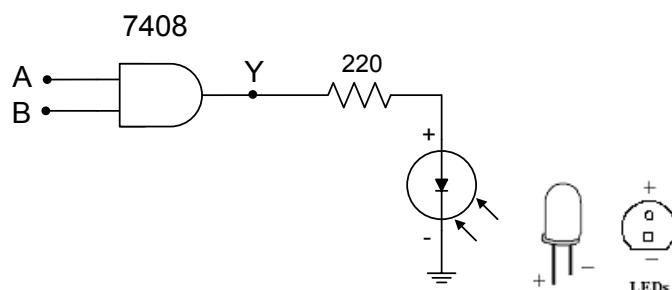
1. ประกอบวงจรลงบน Protoboard ดังแสดงในรูป ให้ต่อแหล่งจ่ายไฟเดี่ยงของวงจรเป็นแรงดันไฟขนาด 5 โวลต์ จาก Power Supply ไปยัง ไอซีแต่ละตัว โดยให้ขา Vcc ของไอซีต่อ กับไฟ +5V และให้ขา Ground ต่อ กับไฟลบของแหล่งจ่ายไฟ IC 7408 , 7404 และ 7432 จะมีขาที่ 14 เป็นแรงไฟ +5V และขา 7 จะเป็นกราว์ด การนับตำแหน่งขาไอซี ขาที่ 1 จะเริ่ม ตรงจุดเครื่องหมายแล้วนับวนเข้มนาฬิกาไปจนครบ



การทดลองการทำงานของ AND Gate จะใช้ IC 7408 โดยภายในมี AND Gate ที่มีอินพุท 2 เส้นจำนวน 4 ตัว และมีตัวแทนง่ายดังรูป



2. ประกอบวงจรตามรูป โดยให้เอาท์พุทของ AND Gate ต่อผ่านตัวความต้านทาน $220\ \Omega$ และ LED ลงกราวด์ เพื่อขับ LED ใหสว่าง การทำงานของ LED จะสว่างเมื่อมีสัญญาณไฟสูง (High) ลงมาจากวงจรเกต เรียกว่าการทำงานแบบ Active High



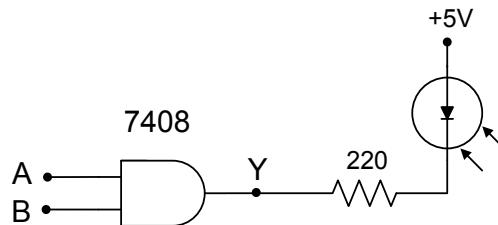
3. ให้ทดลองป้อนค่าแรงดันไฟเข้าที่อินพุท และบันทึกผลการทดลองที่ได้จากเอาท์พุท คือ แรงดันไฟฟ้า , ค่าลอจิก และ LED ติดสว่างหรือไม่ ลงในตาราง

Input		Y		
A	B	Volt	Logic	LED
0	0	0	0	灭
0	1	0	0	灭
1	0	0	0	灭
1	1	5	1	亮

4. ให้คำนวณหาค่ากระแสที่ไหลผ่าน LED โดยใช้กฏของโอลิม์ ว่ามีค่าเท่าไร

$$V=IR \quad I=\frac{V}{R} = \frac{5}{220} = 0.0227 A$$

5. ประกอบวงจรตามรูป โดยให้เอาท์พุตของ AND Gate ต่อผ่านตัวความต้านทาน $220\ \Omega$ และ LED เข้ากับไฟ $+5V$. เพื่อขับ LED ให้สว่าง การทำงานของ LED ในลักษณะนี้เปรียบเสมือนทำงานแบบ Active Low คือเมื่อมีสัญญาณไฟต่ำ (Low) ส่อง光มาจากการจราเกตแล้วหลอด LED จึงจะสว่าง



6. ให้ทดลองป้อนแรงดันไฟเข้าคินพุทและบันทึกผลการทำงานที่ได้จากเอาท์พุท คือ แรงดันไฟฟ้า , ค่าลดจิก และ LED ติดสว่างหรือไม่ ลงในตาราง

Input		Y		
A	B	Volt	Logic	LED
0	0	0	0	สว่าง
0	1	0	0	สว่าง
1	0	0	0	ไม่ติด
1	1	5	1	ติด

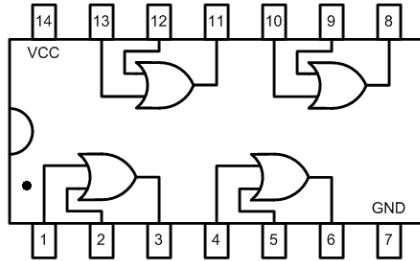
7. เอาท์พุตของ IC ที่ต่อเข้า LED ถ้ากำหนดให้ LED ทำงานคือสว่างเป็น Logic 1 แล้ว การทำงานของวงจรในตารางที่ได้ต้องกับเกตชนิดใด .. **NAND**

8. ให้คำนวณหาค่ากระแสที่ไหลผ่าน LED มีค่าเท่ากับ

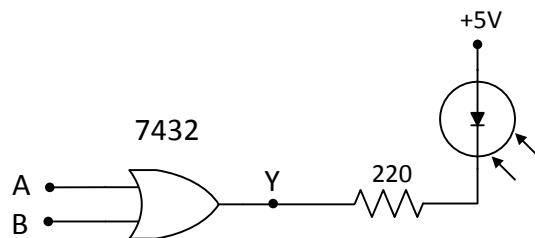
$$V=IR \quad I=\frac{V}{R} = \frac{5}{220} = 0.0227 A$$

9. ระดับแรงดันไฟที่ต้องได้และความสว่างของ LED ในการทดลองข้อ 6 มีความแตกต่างกับการทดลองข้อ 3 อย่างไร **แรงดันไฟต่ำกว่า 5V ทำให้ LED ไม่ติด** **logic ต้องต่ำกว่า Active Low** **จะต้องต่อ LED 逆相才能 Active High**

การทดลองการทำงานของ OR Gate จะใช้ IC 7432 โดยภายในมี OR Gate ที่มีอินพุต 2 เส้น จำนวน 4 ตัว และมีตำแหน่งขาดังรูป



10. ประกอบวงจรตามรูป โดยให้เอาท์พุตของ OR Gate ต่อผ่านตัวความต้านทาน $220\ \Omega$ และ LED เข้ากับไฟ $+5\text{V}$. เพื่อขับ LED ให้สว่าง

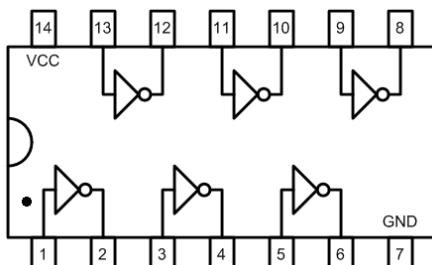


11. ให้ทดลองป้อนแรงดันไฟเข้าที่อินพุต และบันทึกผลการทดลองที่ได้จากเอาท์พุต คือ แรงดันไฟฟ้า , ค่าลอกจิก และ LED ติดสว่างหรือไม่ ลงในตาราง

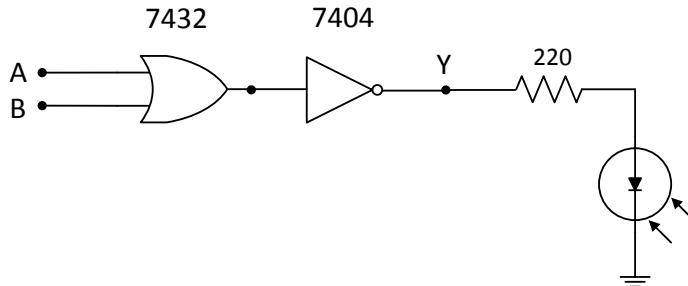
Input		Y		
A	B	Volt	Logic	LED
0	0	0	0	ส่อง
0	1	5	1	สบ
1	0	5	1	สบ
1	1	5	1	สบ

12. เอาท์พุตของ IC ที่ต่อเข้า LED ให้กำหนดให้ LED ทำงานคือสว่างเป็น Logic 1 และ การทำงานของวงจรในตารางที่ได้ตั้งกับเกตชนิดใด **OR Gate**.....

การทดลองการทำงานของ NOT Gate จะใช้ IC 7404 โดยภายในมี NOT Gate ที่มีอินพุต 1 เส้น และเอาท์พุต 1 เส้น จำนวน 6 ตัว และมีตำแหน่งขาดังรูป



13. ประกอบวงจรตามรูป โดยให้เอาท์พุตของ OR Gate ต่อ กับ NOT Gate ผ่านอุปกรณ์ความต้านทาน $220\ \Omega$ และ LED ลงกราวด์ เพื่อขับ LED ให้สว่าง



14. ให้ทดลองป้อนแรงดันไฟเข้าอินพุต และบันทึกผลการทดลองที่ได้จากເອົາທີ່ພູກ ຄືອ แรงดันไฟພໍາໄວ, ອ່າລອອິຈິກ ແລະ LED ຕິດສ່ວ່າງຫຸ້ນໄໝ ລັງໃນຕາຮາງ

Input		Y		
A	B	Volt	Logic	LED
0	0	5	1	ສະບັບ
0	1	0	0	ດັບ
1	0	0	0	ດັບ
1	1	0	0	ດັບ

15. ເອົາທີ່ພູກຂອງໄອົມື່ທີ່ຕ່ອເຂົ້າ LED ດ້ວຍກຳທັນໄທ້ LED ທຳມະນຸດໃຫ້ LED ທຳມະນຸດ ຂຶ້ນສ່ວ່າງເປັນ Logic 1 ແລ້ວ ກາຣ ທຳມະນຸດຂອງຈະຈຸນາຕາຮາງທີ່ເຕີດຕະກັບເກຕະນິດໄດ້ **NOR Gate**.....
16. ให้ทดลองเปลี่ยนຕົວ LED ເປັນສີຕ່າງໆ ອຳຍ່ານ້ອຍ 5 ສີ ແລ້ວວັດແຮງດັນໄຟພໍາທີ່ຕົກຄ່ອມ LED ໃນຂະແໜນທຳມະນຸດໄວ້ກ່ຽວຂ້ອງ ແລະອີນບາຍຄວາມແຕກຕ່າງຂອງຄ່າທີ່ໄດ້ **LED ສີເງິນຂົງຕົ້ນໄປປາກຮ່ວມເກົ່າກົນ 2.87V, LED ສີເໜັກ = 2.03V, LED ສີເໜັບ = 2.07, LED ສີເກົກ = 1.98V, LED ສີນິງ = 2.83V**
- ເພິ່ນ: ພາກເກົ່າກົນຂົງຕົ້ນໄປປາກຮ່ວມກ່າວກ່າວໄຫວ່າມີກົນຕົ້ນໄປປາກຮ່ວມກ່າວກ່າວ
-

17. ให้เพิ่ມຄ່າຕົວຄວາມຕ້ານທານເປັນສອງເທົ່າໄດຍຕ່ອອນຸກວມກັນ 2 ຕັ້ງເປັນ 440 Ω ແລະ ทดลองປ້ອນແຮງດັນໄຟເຂົ້າອິນພູກແລະບັນທຶກຜົດກາຣທົດລອງທີ່ໄດ້ຈາກເອົາທີ່ພູກ ຄືອ ແຮງດັນໄຟພໍາໄວ, ອ່າລອອິຈິກ ແລະ LED ຕິດສ່ວ່າງຫຸ້ນໄໝ

Input		Y		
A	B	Volt	Logic	LED
0	0	4.70	1	ດັບ
0	1	0.05	0	ໂມຕົດ
1	0	0.03	0	ໂມຕົດ
1	1	0.03	0	ໂມຕົດ

18. ให້ອອີນບາຍເບີ່ຍບຜົດກາຣທົດລອງທີ່ໃຊ້ຄ່າຕົວຄວາມຕ້ານທານ 220 Ω ກັບຄ່າຕົວຄວາມຕ້ານທານ 440 Ω ວ່າມີຄວາມແຕກຕ່າງກັນຍ່າງໃຈ **ເນື້ອຕົກມາກົດໄຟປາກຮ່ວມເກົ່າກົນ**
-

ทฤษฎีพิชคณิตบูลีน (Boolean Algebra Theorem) ใช้ในการลดรูปสมการลอจิกให้เหลือน้อยลง เพื่อให้การสร้างวงจรโลจิกใช้อุปกรณ์น้อยลง เป็นการลดการหน่วงเวลาของเกตๆ ในวงจร
ทฤษฎีพิชคณิตบูลีนพื้นฐานหนึ่งตัวแปร

$$A + 0 = A$$

$$A + 1 = 1$$

$$A \cdot 1 = A$$

$$A \cdot 0 = 0$$

$$A + \bar{A} = 1$$

$$A \cdot \bar{A} = 0$$

กฎของเอกลักษณ์ (Identity Law)

$$A + A = A$$

$$A \cdot A = A$$

กฎการนิเสธ (Negation Law)

$$(\bar{A}) = \bar{A}$$

$$\bar{\bar{A}} = A$$

กฎการสลับที่ (Commutative Law)

$$A + B = B + A$$

$$A \cdot B = B \cdot A$$

กฎการจัดขั้นกลุ่ม (Associative Law)

$$A + (B + C) = (A + B) + C$$

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

กฎการกระจาย (Distributive Law)

$$A \cdot (B + C) = A \cdot B + A \cdot C$$

$$A + B \cdot C = (A + B) \cdot (A + C)$$

กฎการลดทอน (Redundance Law)

$$A + A \cdot B = A$$

$$A \cdot (A + B) = A$$

$$A + \bar{A} \cdot B = A + B$$

$$A \cdot (\bar{A} + B) = A \cdot B$$

กฎเดอมอร์กัน (De Morgan's laws)

$$\overline{A + B} = \bar{A} \cdot \bar{B}$$

$$\overline{A \cdot B} = \bar{A} + \bar{B}$$

การเขียน Switching Function หรือ Boolean Equation จากตารางค่าความจริง (Truth Table) สามารถเลือกเขียนเฉพาะเอาท์พุทที่เป็น 1 หรือ 0 ก็ได้ โดยใช้หลักการของ Minterm หรือ Maxterm ตัวอย่าง Truth Table ที่มี 3 ตัวแปร คือ A, B และ C ดังนี้

Input			Output F
A	B	C	
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

สมการเอาท์พุทของวงจรดิจิตอล สามารถเขียนได้ 2 แบบ คือ

ผลบวกของผลคูณ (Sum of Product - SOP) โดยการนำเอาสัญญาณอินพุทที่มีตัวแพรอยู่ตีมจำนวน ซึ่งอาจอยู่ในรูปปกติและรูปคณิตศาสตร์ มากระทำการ AND กัน เราเรียกเทอมที่ AND กันนี้ว่า เทอมผลคูณ (Minterm) แล้วจึงนำมินเทอมแต่ละเทอมรวมกันโดยการ OR กันอีกที ตัวอย่างจาก Truth Table ด้านบนสามารถเขียนเป็นฟังก์ชัน SOP ได้ดังนี้

$$f(A, B, C) = \bar{A} \cdot B \cdot \bar{C} + \bar{A} \cdot B \cdot C + A \cdot \bar{B} \cdot \bar{C} + A \cdot B \cdot \bar{C}$$

นอกจากนี้ อาจเขียนให้อยู่ในรูปของ Minterm (m) ได้ดังนี้

$$f(A, B, C) = m_2 + m_3 + m_4 + m_6$$

เพื่อความสะดวกจะใช้เครื่องหมาย \sum (ซึ่กม่า) แทนคำว่าผลบวก เขียนได้ดังนี้

$$f(A, B, C) = \sum m (2, 3, 4, 6)$$

ผลคูณของผลบวก (Product of Sum - POS) โดยการนำเอาสัญญาณอินพุท ที่อยู่ในรูปปกติ และรูปคณิตศาสตร์ มา OR กัน ซึ่งเราเรียกเทอมที่ OR กันนี้ว่า เทอมผลบวก (Maxterm) จากนั้นจึงนำแมกเทอม แต่ละเทอมมา AND กัน ตัวอย่างจาก Truth Table ด้านบนสามารถเขียนเป็นฟังก์ชัน POS ได้ดังนี้

$$f(A, B, C) = (A + B + C) \cdot (A + B + \bar{C}) \cdot (\bar{A} + B + \bar{C}) \cdot (\bar{A} + \bar{B} + \bar{C})$$

เขียนให้อยู่ในรูปของ Maxterm (M) ได้ดังนี้

$$f(A, B, C) = M_0 \cdot M_1 \cdot M_5 \cdot M_7$$

เพื่อความสะดวกจะใช้เครื่องหมาย \prod (พาย) แทนคำว่าผลคูณ เขียนได้ดังนี้

$$f(A, B, C) = \prod M (0, 1, 5, 7)$$

ผังการ์นอร์ (Karnaugh map) หรือเรียกว่า K-map เป็นวิธีหนึ่งในการลดรูปใช้ในการออกแบบวงจร Combinational Logic สามารถใช้กับสมการที่มีตัวแปรขนาด 2, 3, 4 หรือ 5 ตัวแปร โดยช่องที่อยู่ติดกันในตาราง K-map จะมีค่าต่างกัน 1 บิต เป็นการเรียงค่าในลักษณะของ Gray Code ขนาดของตารางที่ใช้จะขึ้นอยู่กับจำนวนของตัวแปรในวงจรดังนี้

K-map ชนิด 2 ตัวแปร จะใช้ตารางขนาด 2×2 จะมีทั้งหมด $2^2 = 4$ ช่อง

	B	0	1
	0	$\bar{A}\bar{B}$	$\bar{A}B$
	1	$A\bar{B}$	AB

K-map ชนิด 3 ตัวแปร จะใช้ตารางขนาด 2×4 จะมีทั้งหมด $2^3 = 8$ ช่อง

	BC	00	01	11	10
	0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$	$\bar{A}BC$	$\bar{A}B\bar{C}$
	1	$A\bar{B}\bar{C}$	$A\bar{B}C$	ABC	$A\bar{B}\bar{C}$

K-map ชนิด 4 ตัวแปร จะใช้ตารางขนาด 4×4 จะมีทั้งหมด $2^4 = 16$ ช่อง

	CD	00	01	11	10
	00	$\bar{A}\bar{B}\bar{C}\bar{D}$	$\bar{A}\bar{B}\bar{C}D$	$\bar{A}\bar{B}CD$	$\bar{A}B\bar{C}\bar{D}$
	01	$\bar{A}B\bar{C}\bar{D}$	$\bar{A}B\bar{C}D$	$\bar{A}BCD$	$\bar{A}BC\bar{D}$
	11	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$ABC\bar{D}$	$ABC\bar{D}$
	10	$A\bar{B}\bar{C}\bar{D}$	$A\bar{B}\bar{C}D$	$A\bar{B}CD$	$A\bar{B}C\bar{D}$

การลดรูป Boolean expression หรือ Switching function โดยใช้ Karnaugh map มีวิธีการดังนี้

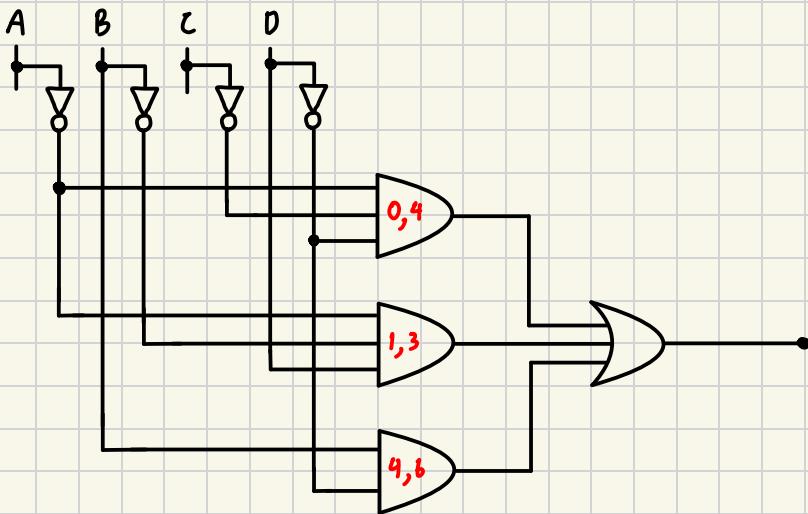
1. ค่าที่ใช้ส่องใน Karnaugh map ถ้าเป็น Minterm มี Logic เป็น 1 และ Maxterm มี Logic เป็น 0 ตามช่องค่าของมัน
2. การจัดกลุ่มให้คุ้งทั้งแนวตั้งและแนวนอน จับตัวที่อยู่ติดกันขนาดของกลุ่มต้องเป็นสี่เหลี่ยม (Rectangles) โดยมีหลักเกณฑ์ที่ว่าจับคู่ได้ที่มีค่าขักกล้ำ 2 เท่านั้น เช่น 1, 2, 4, 8, 16, ...
3. การจับคู่จะต้องจับคู่ที่มีขนาดใหญ่ก่อน เช่น ถ้าจับคู่ได้ 8 ตัว ก็อย่าไปจับคู่แบบ 4 ตัว 2 ครั้ง เพราะจะทำให้ผลลัพธ์ที่ได้ลดจำนวนตัวแปรได้มากกว่า
4. ตัวที่ถูกจับคู่ไปแล้วสามารถนำไปจับคู่กับตัวอื่นได้อีกถ้าจำเป็น
5. เมื่อจับคู่ได้แล้วก็ดำเนินการหาผลลัพธ์ วิธีการหาผลลัพธ์ทำได้โดย นำตัวที่ถูกจับคุ้นนั้นมองดูค่าทางด้านบนและด้านข้าง ค่าที่ซ้ำกันคือผลลัพธ์ที่ต้องการ

AB	CD	00	01	11	10
00	0000	0001	0011	0010	01
01	0100	0101	0111	0110	10
11	1100	1101	1111	1110	11
10	1000	1001	1011	1010	10

$$\bar{A}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{D} + B\bar{D}$$

66011314

66011314



ตัวอย่างจาก Truth Table ด้านบน สามารถใส่ลงใน K-map ได้ดังนี้

		B	C		
		00	01	11	10
A	0	0 ⁰	0 ¹	1 ³	1 ²
	1	1 ⁴	0 ⁵	0 ⁷	1 ⁶

ในการนี้ของ Sum of Product จะเลือกเฉพาะกลุ่มที่เป็นโลจิก 1 ให้เป็น $F = \bar{A} \cdot B + A \cdot \bar{C}$

ถ้ามีอินพุตบางค่าที่ไม่เกิดขึ้นในวงจร จะกำหนดให้เป็น Don't care (d) ซึ่งการเลือกใน K-map จะถูกกำหนดให้เป็น 0 หรือ 1 ก็ได้ตามความเหมาะสมเพื่อให้ได้วงจรน้อยที่สุด

ตัวอย่าง การออกแบบวงจร Combination logic ที่ใช้สำหรับเปลี่ยนรหัสจาก Excess-3 code เป็นรหัส BCD(8421) code โดยที่รหัสทั้งหมดมีจำนวนเท่ากับเลขฐานสิบ คือ 10 รหัส เมื่อเปรียบเทียบกับ Karnaugh map ชนิด 4 ด้านไปริมีอยู่ 16 ช่อง จะเห็นว่า มีจำนวนช่องมากกว่าจำนวนรหัสอยู่ 6 ช่อง ดังนั้นจำนวน 6 ช่องที่มากกว่าอยู่นี้ จึงแทนด้วย Don't care term ซึ่งประโยชน์ในการใส่ Don't care term ลงใน K-map เพื่อให้การออกแบบวงจร Logic จาก Function ของ Output มีจำนวน Logic Gate น้อยที่สุด

Dec	Excess – 3 code				BCD code			
	A	B	C	D	W	X	Y	Z
0	0	0	1	1	0	0	0	0
1	0	1	0	0	0	0	0	1
2	0	1	0	1	0	0	1	0
3	0	1	1	0	0	0	1	1
4	0	1	1	1	0	1	0	0
5	1	0	0	0	0	1	0	1
6	1	0	0	1	0	1	1	0
7	1	0	1	0	0	1	1	1
8	1	0	1	1	1	0	0	0
9	1	1	0	0	1	0	0	1

CD \ AB	00	01	11	10
00	d		1	
01	d		d	
11			d	1
10	d		d	

$$W = AB + ACD$$

CD \ AB	00	01	11	10
00	d			1
01	d		d	1
11		1	d	
10	d		d	1

$$X = \overline{BC} + \overline{BD} + BCD$$

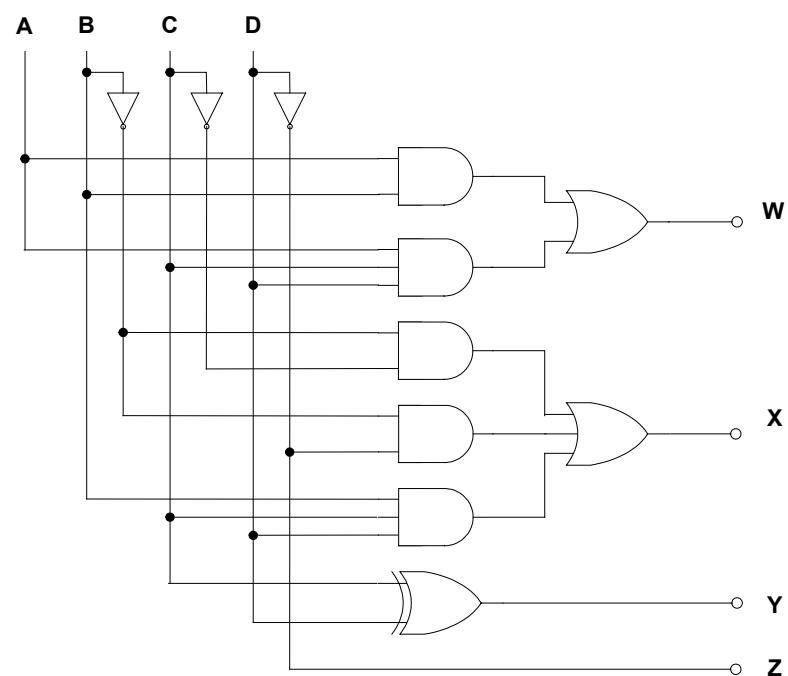
CD \ AB	00	01	11	10
00	d			
01	d	1	d	1
11			d	
10	d	1	d	1

$$Y = \overline{CD} + CD$$

$$= C \oplus D$$

CD \ AB	00	01	11	10
00	d	1	1	1
01	d		d	
11			d	
10	d	1	d	1

$$Z = \overline{D}$$



ระบบเลขฐาน หลักการของระบบตัวเลขจะถูกนำมาใช้ในการทำงานของไมโครโปรเซสเซอร์และวงจรดิจิตอล ซึ่งโดยทั่วไปจะมีสภาวะการทำงานได้ 2 สภาวะ คือ สภาวะเปิดและสภาวะปิด สามารถแทนสภาวะด้วยค่าคงที่เป็นเลขฐานสองได้คือ 1 และ 0 ระบบตัวเลขที่ไมโครโปรเซสเซอร์รับรู้และเข้าใจคือ เลขฐานสอง (Binary Number) ในไมโครโปรเซสเซอร์จะมีสายตัวนำสำหรับสัญญาณที่เรียกว่าสายข้อมูล กลุ่มของสายข้อมูลเรียกว่า ค่าตัวบัส (Data Bus) แต่ละสายข้อมูล 1 เส้น จะประกอบจากหน่วยข้อมูลที่เรียกว่า บิต (Bit)

มนูญของเราใช้เลขฐานสิบอาจจะเป็นเพียงเรมี 10 นิ้ว แต่ในไมโครโปรเซสเซอร์ข้อมูลต่างๆ จะถูกแสดงเป็นชุดของตัวเลขในระบบเลขฐานสองเรียกว่าไบนารี (Binary) เพราะใน 1 หลักจะมีเพียงสองตัวคือ เลขคูนย์และหนึ่ง

กลุ่มของตัวเลข 8 บิตจะเรียกว่าไบต์ (Byte) ใน 1 ไบต์จะมีจำนวนข้อมูลทั้งหมดเท่ากับ $2^8 = 256$ ค่าโดยไบต์ที่มีค่าเป็น 0 จะแสดงเป็น 00000000 และไบต์ที่ไม่ใช่คูนย์จะเป็นการผสมกันของเลข 1 และ 0 เช่น 01001011 บิตที่อยู่ทางซ้ายสุดของชุดข้อมูลไบนารีเรียกว่า บิตที่มีนัยสำคัญมากที่สุด (Most Significant Bit) เรียกย่อว่า MSB และบิตขวาสุดจะเรียกว่า บิตที่มีนัยสำคัญน้อยที่สุด (Least Significant Bit) ย่อว่า LSB

เพื่อให้เห็นค่าของข้อมูลแต่ละบิตในรูปเลขฐานสิบ จะเขียนให้อ่ายในรูปของเลขชี้กำลัง (Exponent) โดยให้ตำแหน่งบิตเป็นค่าของตัวเลขที่แสดงค่ายกกำลัง และมีฐานเป็นเลขสอง เริ่มจากบิตที่มีค่าน้อยที่สุดทางขวา (LSB) ฐานที่มีค่าเป็น 2 ยกกำลัง 0 จะมีค่าผลลัพธ์เท่ากับ 1 บิตต่อมาเรียงตามลำดับตำแหน่งบิตถัดไปทางด้านซ้ายที่คละ 1 บิต เลขชี้กำลังที่ได้แต่ละค่าจะเพิ่มขึ้นที่ละ 1 ดังนั้นหลักต่อมาจะได้ฐาน 2 ยกกำลัง 1 มีค่าเท่ากับ 2 เมื่อข้อมูลไบนารีมีขนาดเท่ากับ 1 ไบต์ บิตที่มีนัยสำคัญมากที่สุด (MSB) จะมีค่าเลขชี้กำลังเป็น 7

ตำแหน่งบิตของข้อมูลมักจะถูกเขียนขึ้นเป็นตัวอักษร D โดยเริ่มจาก D0, D1, D2, D3, ฯลฯ ดังนั้นผลลัพธ์ของข้อมูลในแต่ละบิตในรูปเลขฐานสิบ (Decimal) จะแสดงได้ดังนี้

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Base exponent	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal	128	64	32	16	8	4	2	1

ตัวอย่าง ไบนารีที่มีค่า 10100011 ถ้าต้องการทำให้อ่ายในรูปเลขฐานสิบให้ทำการแยกตำแหน่งค่าไบนารี 10100011 ในแต่ละบิตได้ดังนี้

1 0 1 0 0 0 1 1

$$(1 * 128) + (0 * 64) + (1 * 32) + (0 * 16) + (0 * 8) + (0 * 4) + (1 * 2) + (1 * 1)$$

แล้วนำเอาผลลัพธ์ของบิตที่มีค่าเป็น 1 ให้อบรวมกับเข้าด้วยกัน ผลลัพธ์รวมจะได้

$$128 + 32 + 2 + 1 = 163$$

การแสดงตัวเลขไบนารีเมื่อมีค่าของข้อมูลจำนวนมาก จะทำให้มีจำนวนหลักของข้อมูลที่มาก ซึ่งยุ่งยากต่อการเขียน เพื่อให้ง่ายเข้าโดยทั่วไปในการแสดงผลของข้อมูลจึงนิยมเขียนให้อ่ายในรูปของเลขฐานหก (Hexadecimal) หรือย่อว่า HEX โดยจะใช้เลขฐานสิบหกเบ่ง 1 ไบต์ ออกเป็นสองกลุ่ม กลุ่มละ 4 บิต ถ้าทั้ง 4 บิตมีค่า 0 จะได้ 0000 มีค่าเท่ากับ 0 แต่ถ้าทุกบิตของมันมีค่าเป็น 1 หมวด จะได้ 1111 มีค่าเท่ากับ $8 + 4 + 2 + 1 = 15$

ดังนั้นเราจึงได้จำนวนข้อมูลทั้งหมดเท่ากับ 16 ค่า คือ จาก 0 ถึง 15 การใช้เลขฐานสิบหกสามารถทำให้เขียนได้ง่ายขึ้น โดยแทน 0 ถึง 9 เมื่อเป็นเลขฐานสิบ และเฉพาะค่าเลข 10 ถึง 15 ซึ่งเป็นเลข 2 หลัก จะเปลี่ยนให้เป็น 1 หลัก โดยแทนด้วยตัวอักษร A ถึง F ตารางต่อไปนี้แสดงค่าของเลขฐานสิบหก (Hexadecimal) เทียบกับเลขฐานสอง (Binary) และเลขฐานสิบ (Decimal)

Binary	Decimal	Hexadecimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

การเขียนค่าเลขฐานสิบหกในภาษา Assembly ให้ต่อท้ายด้วย h และถ้าเป็นภาษา C ให้ใส่นำหน้าด้วย 0x ค่าข้อมูลในแต่ละบิตแสดงเป็นเลขฐานสิบหกจะได้ดังนี้

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Base exponent	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal	128	64	32	16	8	4	2	1
Hexadecimal	80	40	20	10	08	04	02	01

การกำหนดค่าตัวเลขจำนวนเต็ม (Integer) ในภาษาชั้นสูง ซึ่งส่วนมากมักจะใช้ย่อว่า Int สามารถกำหนดค่าเพื่อให้แสดงผลของตัวเลขที่มีค่าเป็นเลขบวกเท่านั้นเรียกว่า Unsigned หรืออาจกำหนดให้เป็นตัวเลขที่มีค่าเป็นลบด้วยและต้องเนื่องไปเป็นค่าบวกเรียกว่า Signed

เมื่อกำหนดตัวเลขที่เก็บเป็น Integer ขนาด 16 บิตหรือ 2 ไบต์ จะได้จำนวนข้อมูลทั้งหมดเท่ากับ $2^{16} = 65,535$ ค่า ข้อมูลแบบ Unsigned จะมีค่าไฝ์ตั้งแต่ 0 ถึง 65,535 และเมื่อกำหนดให้ข้อมูลเป็นแบบ Signed จะใช้บิต MSB ในที่นี้คือบิตที่ 15 เป็นบิตเครื่องหมาย (Sign Bit) ถ้าบิตนี้เป็น 0 ข้อมูลที่ได้จะเป็นค่าวา ก (Positive) อยู่ในช่วงตั้งแต่ 0 ถึง 32,767 แต่ถ้าบิตนี้มีค่าเป็น 1 ข้อมูลที่ได้จะเป็นค่าลบ (Negative) ตั้งแต่ -1 ถึง -32,768

เมื่อมีข้อมูลเป็นค่าลบแบบ Signed ถ้าต้องการทราบว่าจะเท่ากับค่าเท่าไรในแบบ Unsigned จะหาความสัมพันธ์ได้ดังนี้

$$\text{Unsigned} = \text{Signed} + 65536$$

เช่น ตัวเลขแบบ Signed ที่มีค่าเท่ากับ -1 ถ้าต้องการเปลี่ยนให้เป็นแบบ Unsigned จะได้ $-1 + 65536 = 65535$ เท่ากับ 65535 ตัวอย่าง ตัวเลขฐานสิบหก (Hex) ขนาด 16-bit ที่แสดงเป็นค่าเป็นแบบ Signed และ Unsigned ได้ดังนี้

HEX	0000	0001	7FFE	7FFF	8000	8001	FFFE	FFFF
Signed	0	1	32766	32767	-32768	-32767	-2	-1
Unsigned	0	1	32766	32767	32768	32769	65534	65535

ถ้าตัวเลขจำนวนเต็มที่จะจัดเก็บมีค่ามากกว่าหลักหมื่นก็ต้องเพิ่มขนาด Integer ให้ใหญ่ขึ้นเป็น 32 บิต หรือ 4 ไบต์ จะได้จำนวนข้อมูลทั้งหมดเท่ากับ $2^{32} = 4,294,967,296$ ค่า ข้อมูลแบบ Unsigned จะอยู่ในช่วงระหว่าง 0 ถึง 4,294,967,295 (FFFFFFF) และได้ข้อมูลเป็นแบบ Signed จะอยู่ในช่วงระหว่าง -2,147,483,648 ถึง +2,147,483,647 แต่ถ้าข้อมูลตัวเลขจำนวนเต็มที่จะจัดเก็บมีค่ามากกว่าหลักพันล้านก็จะต้องเพิ่มขนาดของ Integer ให้ใหญ่ขึ้นไปอีก โดยมากมักจะเพิ่มขนาดที่ละหนึ่งเท่าตัวเป็น 64 บิต และต่อไปเป็น 128 บิต และ 256 บิตตามลำดับ

หน่วยที่ใช้วัดจำนวนข้อมูลของคอมพิวเตอร์ นิยมวัดเป็น กิโลไบต์ (Kilobyte) , เมกะไบต์ (Megabyte) , กิกะไบต์ (Gigabyte) และเทระไบต์ (Terabyte) ซึ่งในแต่ละหน่วยจะมีค่าตัวคูณต่างกันเท่ากับ $2^{10} = 1,024$ และมนุษย์เรา ซึ่งใช้เลขฐานสิบจะประมาณค่าตัวคูณไว้ที่ 1,000 หน่วยเพื่อความสะดวกในการคำนวณ โดยสามารถสรุปเป็นจำนวนค่าข้อมูลที่เก็บได้ดังนี้

$$1 \text{ กิโลไบต์ (KB)} = 2^{10} = 1,024 \text{ ไบต์}$$

$$1 \text{ เมกะไบต์ (MB)} = 2^{20} = 1,048,576 \text{ ไบต์ หรือ } 1,024 \text{ กิโลไบต์}$$

$$1 \text{ กิกะไบต์ (GB)} = 2^{30} = 1,073,741,824 \text{ ไบต์ หรือ } 1,024 \text{ เมกะไบต์}$$

$$1 \text{ เทระไบต์ (TB)} = 2^{40} = 1,099,511,627,776 \text{ ไบต์ หรือ } 1,024 \text{ กิกะไบต์}$$

วงจรบวกเลขไบนาเรี่ย (Binary Adder) หลักการบวกเลขไบนาเรี่ยเหมือนกับการบวกเลขฐานสิบสามารถทำได้โดยนำเลขฐานสอง 2 จำนวนนับมาเข้าด้วยกันโดยตรงที่ละ 1 บิต แบ่งได้เป็น 2 แบบ ดังนี้

การบวกเลขแบบไม่คิดตัวทด (Half Adder) ในการบวกเลขฐานสองที่ละบิต สามารถทำได้โดยใช้กฎการบวกดังนี้

$$1. \ 0 + 0 = 0$$

$$2. \ 0 + 1 = 1$$

$$3. \ 1 + 0 = 1$$

$$4. \ 1 + 1 = 10$$

จากข้อ 1 ถึง 3 ผลบวกที่ได้จะไม่มีตัวทด ส่วนข้อ 4 ผลบวกจะเท่ากับ 0 และมีตัวทด 1 เพื่อใช้ในการบวกของหลักต่อไป วงจร Half Adder สามารถสร้างได้โดยมีอินพุตเป็นเลขไบนาเรี่ย 2 ค่าที่จะบวกคือ A เป็นตัวตั้ง และ B เป็นตัวบวก ผลลัพธ์ที่ได้ออกที่เอ้าท์พุท 2 ค่าคือ S เป็นผลบวก (Sum) และ C เป็นตัวทด (Carry) สามารถเขียน Block Diagram และตารางความจริง (Truth Table) ของการบวกเลขขนาด 1 บิตได้ดังนี้

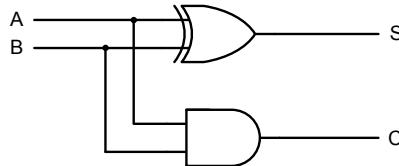


Input		Output	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

จากตารางความจริงสามารถนำมาเขียนเป็นสมการลอจิกและวงจร Half Adder ได้ดังนี้

$$S = A \cdot \bar{B} + \bar{A} \cdot B = A \oplus B$$

$$C = A \cdot B$$



การบวกเลขแบบคิดตัวทด (Full Adder) จะใช้ในกรณีที่มีการบวกเลขตั้งแต่บิตที่สองขึ้นไปเนื่องจากการบวกเลขแบบไม่คิดตัวทด (Half Adder) จะใช้สำหรับบิตที่มีนัยสำคัญต่ำสุด (LSB) ซึ่งในบิตที่สูงขึ้นไปหรือบิตถัดไปจะต้องใช้การบวกที่รวมตัวทดของบิตที่ทำการบวกก่อนหน้านี้ไว้ด้วย วงจร Full Adder จึงต้องมีอินพุต 3 ค่า คือ A เป็นตัวตั้ง, B เป็นตัวบวก และ Cin เป็นตัวทดเข้า โดยมีเอาท์พุท 2 ค่าคือ S เป็นผลบวก และ Cout เป็นตัวทดออก สามารถเขียน Block Diagram และตารางความจริง (Truth Table) ได้ดังนี้

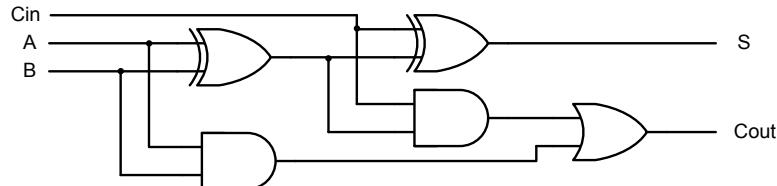


Input			Output	
Cin	A	B	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

จากตารางความจริงสามารถนำมาเขียนเป็นสมการลอจิกและวงจร Full Adder ได้ดังนี้

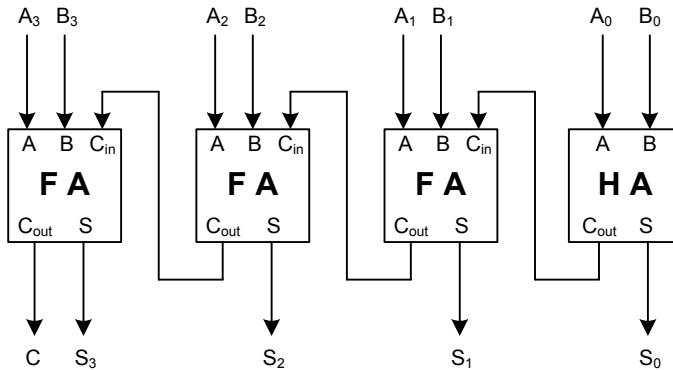
$$S = \overline{Cin} \cdot \bar{A} \cdot B + \overline{Cin} \cdot A \cdot \bar{B} + Cin \cdot \bar{A} \cdot \bar{B} + Cin \cdot A \cdot B = Cin \oplus (A \oplus B)$$

$$Cout = \overline{Cin} \cdot A \cdot B + Cin \cdot \bar{A} \cdot B + Cin \cdot A \cdot \bar{B} + Cin \cdot A \cdot B = A \cdot B + Cin \cdot (A \oplus B)$$



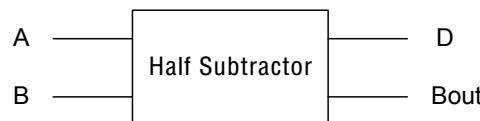
การบวกเลขแบบขนาน (Parallel Adder) การบวกแบบนี้ทุกๆ บิตของข้อมูลจะถูกป้อนเข้าสู่อินพุตพร้อมกัน เอ้าท์พุทที่ได้จะเป็นผลลัพธ์ของการบวก การบวกเลขฐานสองแบบหลายบิตนี้สามารถนำเอาระบบเลข Half Adder หนึ่งตัวและ Full Adder หลายตัวมาใช้ร่วมกัน โดยในบิตแรก (LSB) จะใช้วงจรแบบ Half Adder และในบิตต่อๆไปจะใช้วงจรแบบ Full Adder โดยการบวกตั้งแต่บิตที่สองเป็นต้นไปจะต้องนำตัวทดในหลักที่ต่ำกว่าที่อยู่ก่อนหน้านี้ มาเป็นอินพุต Cin เพื่อนำมาบวกรวมกับบิต A และ B การบวกเลขในลักษณะนี้ จนถึงบิตสูงสุด (MSB) ค่าผลลัพธ์ทางเอ้าท์พุทของการบวกเลขในรูปแบบ n บิตอาจมีค่าเกินจำนวนบิตทั้งหมดอยู่หนึ่งไถ่ คือ ไถ่จำนวนเป็น $n + 1$ บิต ซึ่งเอ้าท์พุทของบิตที่เกินมาจะกำหนดให้เป็นบิตตัวทด (Carry Bit) ตัวอย่างการบวกเลขขนาด 4 บิตและวงจรจะได้ดังนี้

$$\begin{array}{r}
 & A_3 & A_2 & A_1 & A_0 \\
 & B_3 & B_2 & B_1 & B_0 \\
 \hline
 C & S_3 & S_2 & S_1 & S_0
 \end{array} +$$



วงจรลบเลขเลขไบนารี่ (Binary Subtractor) สามารถทำได้โดยใช้การลบกันโดยตรง หรืออาจจะทำโดยการ 2's complement ตัวลบแล้ววกเข้ากับตัวตั้งก็ได้ การลบเลขไบนารีนี้สำหรับที่เป็นตัวตั้งมีค่าน้อยกว่าตัวลบจะต้องมีการยื้อจากบิตที่สูงกว่า ทำให้บิตที่ถูกยื้อมีค่าน้อยลงหนึ่ง และจะใช้เป็นตัวตั้งของบิตถัดไป วงจรลบเลขไบนารีทำได้ 2 แบบ เมื่อมองการบวกดังนี้

การลบแบบไม่คิดตัวยื้อ (Half Subtractor) เป็นวงจรที่ใช้ในการลบเลขไบนารีขนาด 2 บิตเข้าด้วยกัน สามารถสร้างได้โดยมีอินพุตเป็นเลขไบนารี 2 ค่า ที่จะนำมารบกันคือ A เป็นตัวตั้ง และ B เป็นตัวลบ ผลลัพธ์ที่ได้ออกที่เอาท์พุต 2 ค่าคือ D เป็นผลต่าง (Difference) และ Bout เป็นตัวยื้อ (Borrow out) โดยการลบแบบ Half Subtractor นี้จะใช้สำหรับบิตที่มีนัยสำคัญต่ำสุด (LSB) สามารถเขียน Block Diagram และตารางความจริง (Truth Table) ของการลบเลขขนาด 1 บิตได้ดังนี้

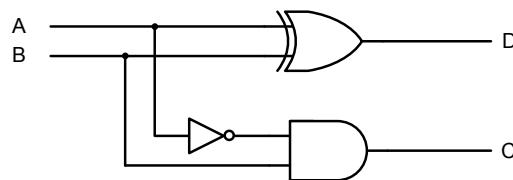


Input		Output	
A	B	D	Bout
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

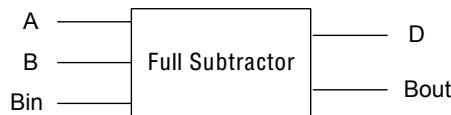
จากตารางความจริงสามารถนำมาเขียนเป็นสมการลอจิกและวงจร Half Subtractor ได้ดังนี้

$$D = A \cdot \bar{B} + \bar{A} \cdot B = A \oplus B$$

$$\text{Bout} = \bar{A} \cdot B$$



การลบเลขแบบบิดตัวบีม (Full Subtractor) เป็นวงจรที่ใช้ในการลบเลขไบนารี 2 บิตและตัวบีมอีก 1 บิต ใช้ในกรณีที่เป็นการลบของบิตที่ถัดจากบิตแรก ตั้งแต่บิตที่สองขึ้นไป ซึ่งจะต้องใช้การลบที่รวมตัวบีมของบิตที่ทำการลบก่อนหน้านี้ วงจร Full Subtractor จะมีอินพุท 3 ค่า คือ A เป็นตัวตั้ง, B เป็นตัวลบ และ Bin เป็นตัวบีมที่ลูกหลักต่ำกว่าของบีมไป โดยมีเอาท์พุท 2 ค่าคือ D เป็นผลต่าง และ Bout เป็นตัวบีมที่จะขอรับในหลักที่สูงกว่ามา สามารถเขียน Block Diagram และตารางความจริง(Truth Table) ของวงจร Full Subtractor ได้ดังนี้

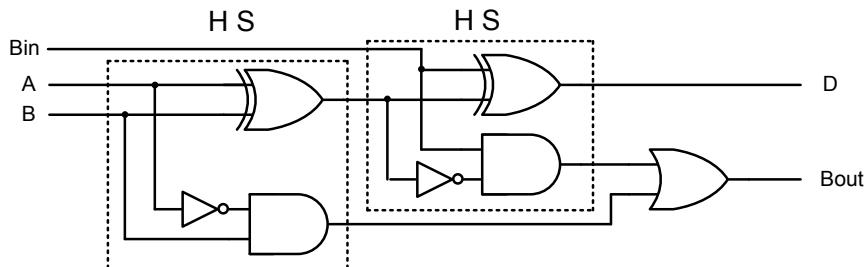


Bin	Input		Output	
	A	B	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

จากตารางความจริงสามารถนำมาเขียนเป็นสมการลอจิก และวงจร Full Subtractor ได้ดังนี้

$$S = \overline{Bin} \cdot \bar{A} \cdot B + \overline{Bin} \cdot A \cdot \bar{B} + Bin \cdot \bar{A} \cdot \bar{B} + Bin \cdot A \cdot B = Bin \oplus (A \oplus B)$$

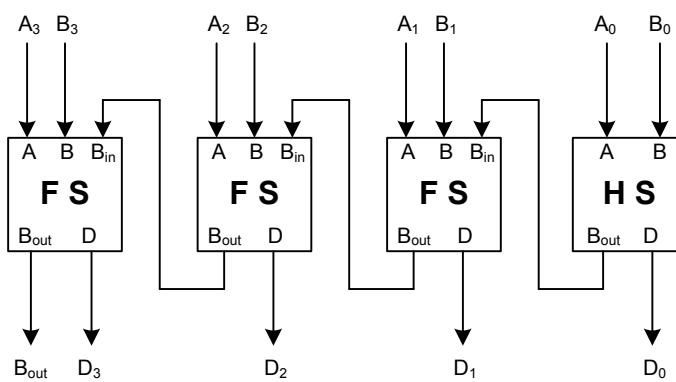
$$Bout = \overline{Bin} \cdot \bar{A} \cdot B + Bin \cdot \bar{A} \cdot \bar{B} + Bin \cdot \bar{A} \cdot B + Bin \cdot A \cdot B = \bar{A} \cdot B + Bin \cdot \overline{(A \oplus B)}$$



การลบเลขแบบบานาน (Parallel Subtractor) การลบแบบนี้ทุกๆ บิตของข้อมูลจะถูกป้อนเข้าสู่อินพุทของวงจรในเวลาเดียวกัน คล้ายกับการต่อวงจรแบบบานาน การลบเลขฐานสองแบบหลายบิตนี้จะใช้วงจรคบเลข Half Subtractor หนึ่งตัวและ Full Subtractor หลายตัวมาประกอบรวมกัน โดยในบิตแรก (LSB) จะใช้วงจรแบบ Half Subtractor และในบิตต่อๆ ไปจะใช้วงจรแบบ Full Subtractor

โดยการลบตั้งแต่บิตที่สองเป็นต้นไปจะต้องนำเอาท์พุตตัวบีม Bout จากการลบในหลักที่ต่ำกว่าที่อยู่ก่อนหน้านี้ มาเป็นอินพุท Bin เพื่อใช้ในการลบของหลักนั้น และได้ Bout ค่าใหม่ซึ่งจะถูกนำไปคบกับอินพุตตัวบีมของบิตถัดไป ค่าผลลัพธ์ทางเอาท์พุตของการลบเลขในเรื่องนาด n บิตอาจมีค่าเท่ากับจำนวนบิตทั้งหมดคือ $n+1$ บิต ซึ่งเอาท์พุตของบิตที่เกินมาจะกำหนดให้เป็นบิตตัวบีม Bout ตัวอย่างการลบเลขนาด 4 บิต และวงจรที่ได้เป็นดังนี้

$$\begin{array}{r} A_3 \ A_2 \ A_1 \ A_0 \\ B_3 \ B_2 \ B_1 \ B_0 \\ \hline \text{Bout} \quad S_3 \ S_2 \ S_1 \ S_0 \end{array} -$$



ตัวอย่าง การบวกเลขขนาด 1 Byte **add a,b**

$$\begin{array}{ccccccccccccc}
 & A_7 & A_6 & A_5 & A_4 & A_3 & A_2 & A_1 & A_0 \\
 & B_7 & B_6 & B_5 & B_4 & B_3 & B_2 & B_1 & B_0 \\
 [C] & A_7 & A_6 & A_5 & A_4 & A_3 & A_2 & A_1 & A_0 \\
 & & & & & & & & + \\
 \end{array}$$

ผลลัพธ์ค่าของ Flag มีดังนี้ Carry , Sign, Overflow , Zero

Dec	Hex	Bin	
42	2A	0 0 1 0 1 0 1 0	
<u>69</u>	<u>45</u>	<u>0 1 0 0 0 1 0 1</u>	+
111	6F	0 0 1 1 0 1 1 1 1	

C=0 , S=0 , O=0 , Z=0
(+)

Dec	Hex	Bin	
93	5D	0 1 0 1 1 1 0 1	
<u>81</u>	<u>51</u>	<u>0 1 0 1 0 0 0 1</u>	+
174	AE	0 1 0 1 0 1 1 1 0	
(-82)			

C=0 , S=1 , O=1 , Z=0
(-)

Dec	Hex	Bin	
-106	96	1 0 0 1 0 1 1 0	
<u>-45</u>	<u>D3</u>	<u>1 1 0 1 0 0 1 1</u>	+
-151	69	1 0 1 1 0 1 0 0 1	
105			

C=1 , S=0 , O=1 , Z=0
(+)

Dec	Hex	Bin	
61	3D	0 0 1 1 1 1 0 1	
-123	85	<u>1 0 0 0 0 1 0 1</u>	+
-62	C2	0 1 1 0 0 0 0 1 0	

C=0 , S=1 , O=0 , Z=0
(-)

Dec	Hex	Bin	
-39	D9	1 1 0 1 1 0 0 1	
<u>83</u>	<u>53</u>	<u>0 1 0 1 0 0 1 1</u>	+
44	2C	1 0 0 1 0 1 1 0 0	

C=1 , S=0 , O=0 , Z=0
(+)

รหัส (Code) ระบบการรับส่งข้อมูลจะมีการเข้ารหัสด้วยจุดประสงค์ที่แตกต่างกัน ภาษาเขียนที่ใช้กันก็เป็นรหัสที่สร้างขึ้นมาเพื่อใช้ในการสื่อสาร โดยเป็นรหัสที่คนส่วนใหญ่เข้าใจตรงกัน จึงเป็นรหัสที่เป็นสากล แต่รหัสที่ใช้ในการสื่อสารบางอย่างอาจต้องการให้ใช้เฉพาะผู้ที่เกี่ยวข้องเท่านั้น และไม่ต้องการให้ผู้อื่นรู้ จึงต้องเข้ารหัสเพื่อให้เป็นความลับ ในส่วนของ IBM โคโรโพรেชันที่มีการทำงานในลักษณะสากลชั้นนำ โดยเป็นรูปแบบของเลขฐานสอง รหัสจึงเป็นการเปลี่ยนค่าของตัวเลข, ตัวอักษร หรือรหัสควบคุณ ให้เป็นกลุ่มของเลขฐาน ซึ่งมีการกำหนดความหมายในรหัสที่นำไปใช้งานที่แตกต่างกัน เราจึงต้องศึกษารหัสต่าง ๆ เพื่อให้เข้าใจและสามารถนำไปใช้

Binary Coded Decimal (BCD) เป็นรหัสที่ใช้แทนเลขฐานสิบ (Decimal) จำนวน 1 หลัก ด้วยเลขฐานสอง (Binary) จำนวน 4 บิต รหัส BCD เป็นรหัสที่มีน้ำหนัก (Weighted Code) ในการแทนค่า มีมากน้อยหลายชนิด แต่ที่นิยมใช้จะเป็น BCD-8421 Code โดยเป็นรหัสเลขฐานสองที่มีจำนวนเท่ากับจำนวนของเลขฐานสิบ คือ 10 รหัส ซึ่งในแต่ละบิตของรหัส BCD-8421 จะเป็นน้ำหนักประจำหลักเกิดจากค่า 1 หรือ 0 โดยกำหนดมิตางbam ว่ามีอสูร์น้ำหนักเป็น 1 ถัดมาเป็น 2, 4 และซ้ายมีอสูร์เป็น 8 ตามลำดับ นอกจากนี้ยังมีรหัส BCD แบบอื่นๆ อีกหลายแบบดังนี้

ตาราง แสดงรหัส BCD และ Excess-3

Decimal	BCD-8421	BCD-7421	BCD-5421	BCD-2421	BCD-4221	Excess-3
0	0000	0000	0000	0000	0000	0011
1	0001	0001	0001	0001	0001	0100
2	0010	0010	0010	0010	0010	0101
3	0011	0011	0011	0011	0011	0110
4	0100	0100	0100	0100	1000	0111
5	0101	0101	1000	1011	0111	1000
6	0110	0110	1001	1100	1100	1001
7	0111	1000	1010	1101	1101	1010
8	1000	1001	1011	1110	1110	1011
9	1001	1010	1100	1111	1111	1100

รหัสเกิน 3 (Excess-3 code) ตัวแปลงมาจาก BCD-8421 code รหัส Excess-3 จะมีค่าเท่ากับ BCD-8421 บวกเพิ่มอีก 3 รหัสชนิดนี้จะใช้แทนเลขฐานสิบเช่นกัน รหัส Excess-3 เป็นรหัสที่ไม่มีน้ำหนัก (Nonweighted Code) เพราะค่าน้ำหนักของแต่ละบิตจะไม่เท่ากับค่าจำนวนที่ใช้แทนของเลขฐานสอง รหัส Excess-3 จะถูกนำไปใช้ประโยชน์ในการบวกเลข การเปลี่ยนรหัส BCD-8421 Code มาเป็น Excess-3 Code ที่เพื่อผลทางคณิตศาสตร์ โดยผลบวกของ Excess-3 Code จะให้ผลลัพธ์เป็น BCD-8421 Code เช่น

$$\begin{array}{r}
 5 \\
 + \quad 1000 \\
 \hline
 8 \quad \underline{1011} \\
 \hline
 13 \quad 1 \quad 0011
 \end{array}
 \begin{array}{l}
 \text{Excess-3 Code} \\
 \text{Excess-3 Code} \\
 \text{BCD-8421 Code}
 \end{array}$$

Gray Code เป็นรหัสที่ใช้ในระบบที่เป็นอุปกรณ์อินพุตและเอาท์พุต เช่น ระบบการตรวจจับสัญญาณด้วยแสง ระบบทำ Code ด้วยเกณฑ์หมุนทางกล (Mechanics) เพื่อบอกตำแหน่งของเพลาหมุน รหัสแบบนี้เป็นแบบไม่มีน้ำหนักเช่นกัน ซึ่งในระหว่างกลุ่มรหัส (Code Group) ที่เรียงลำดับกันไป รหัสที่อยู่ในตำแหน่งที่

ติดกันจะมีการเปลี่ยนแปลงของรหัสครั้งละ 1 บิตเท่านั้น ทำให้โอกาสที่จะเกิดความผิดพลาดในการรับรหัส เป็นไปได้น้อย

ตาราง แสดงรหัส Gray code กับ Binary

Decimal	Binary				Gray Code			
	B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

วิธีการแปลงเลข Binary ไปเป็น Gray Code จะเริ่มจากนำบิตแรกที่มีนัยสำคัญมากที่สุด (MSB) ซึ่งอยู่ทางซ้ายของเลข Binary มาเป็นบิตแรกของ Gray code และนำ ผลบวกของเลข Binary ในบิตแรกกับบิตที่ 2 มาเป็นบิตที่ 2 ของ Gray code แล้วบิตต่อๆไปได้จากการนำบิตที่ 2 บวกกับบิตที่ 3 และบิตที่ 3 บวกกับบิตที่ 4 ตามลำดับกันไปของเลข Binary เรื่อยๆ ผลบวกที่ได้ในแต่ละครั้งจะเป็น Gray Code โดยการบวกใช้หลักเกณฑ์ว่า $0+0=0$, $0+1=1$, $1+0=1$ และ $1+1=0$ (ตัวทศ 1 ให้ตัดทิ้งไป) หรืออาจใช้อีกคลูชีฟอร์ (Exclusive OR) แทนได้ดังนี้

$$\begin{aligned} G_3 &= B_3 \\ G_2 &= B_3 \oplus B_2 \\ G_1 &= B_2 \oplus B_1 \\ G_0 &= B_1 \oplus B_0 \end{aligned}$$

การแปลง Gray Code ให้เป็น Binary จะเริ่มจากนำบิตแรกที่เป็น MSB ของเลข Gray Code มาเป็นบิตแรกของ Binary และให้นำผลบวกในบิตที่ 2 ของ Gray Code กับบิตแรกของเลข Binary มาเป็นบิตที่ 2 ของเลข Binary แล้วบิตต่อๆไปได้จากการนำบิตที่ 3 ของ Gray Code บวกกับบิตที่ 2 ของเลข Binary และบิตที่ 4 ของ Gray Code บวกกับบิตที่ 3 ของเลข Binary ผลบวกที่ได้ในแต่ละครั้งจะเป็นเลข Binary เรียงตามลำดับกันไปเรื่อยๆ

$$\begin{aligned} B_3 &= G_3 \\ B_2 &= B_3 \oplus G_2 \\ B_1 &= B_2 \oplus G_1 \\ B_0 &= B_1 \oplus G_0 \end{aligned}$$

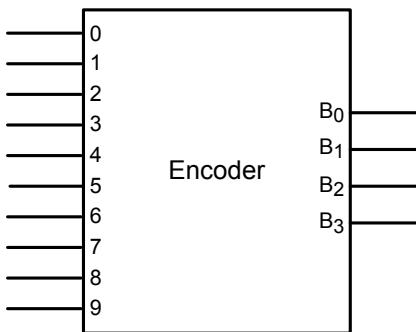
รหัสแอลกิ (ASCII) ย่อมาจาก รหัสมำตรฐานของสหรัฐอเมริกาเพื่อการแลกเปลี่ยนสารสนเทศ (American Standard Code for Information Interchange) เป็นรหัสที่ใช้แทนตัวอักษร (Character) รวมทั้งรหัสควบคุมต่าง ๆ ด้วย ใช้เป็นรหัสมำตรฐานในคอมพิวเตอร์ และเครื่องมือสื่อสารแบบดิจิตอล โดยมีขนาด 7 บิต เก็บได้ทั้งหมด 128 ตัว ภายหลังเพิ่มขนาดขึ้นอีก 1 บิตทำให้มีขนาดเป็น 256 ตัว ใช้เพิ่มอักษรในภาษาของแต่ละท้องถิ่น

ตาราง แสดงรหัส ASC II

วงจรเข้ารหัส (Encoder) หมายถึง วงจรที่ทำหน้าที่แปลงข้อมูลอินพุทที่ป้อนเข้ามา ซึ่งอาจจะเป็นตัวเลข ตัวอักษร หรือรหัสอื่นๆ ที่อยู่ในรูปของระดับแรงดันหรือสัญญาณalogic ให้ออกที่เอาท์พุทเป็นสัญญาณข้อมูลเลขฐานสองตามรหัสที่ต้องการ ตัวอย่างของวงจรเข้ารหัส ได้แก่

การกดสวิตช์เลขหมาย หรือกด Keyboard จะได้ออกท์พุทของวงจรอ ก็เป็นสัญญาณalogic ของรหัส เลขฐานสอง ซึ่งข้อมูลเอาท์พุทที่ออกจะเป็นรหัสอะไรก็ขึ้นอยู่กับการกำหนดค่าตามความต้องการของผู้ใช้

การเข้ารหัสจากเลขฐานสิบให้เป็นรหัส BCD 8421 (Decimal to BCD Encoder) โดยที่สัญญาณเข้ามีจำนวนอินพุทเท่ากับ 2^n เส้น และเมื่อเข้ารหัสแล้วจะได้สัญญาณเอาท์พุทจำนวน n เส้น เช่น สัญญาณเข้า 10 เส้น สามารถเข้ารหัสได้สัญญาณด้านเอาท์พุทออก 4 เส้น โดยมีลักษณะ ดังรูป



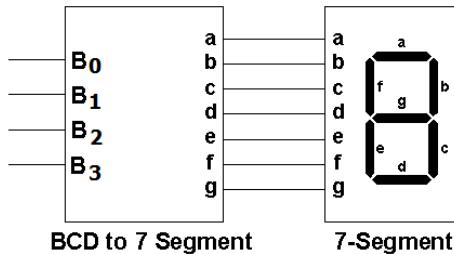
จะได้ Truth Table ของวงจรการเข้ารหัสจากเลขฐานสิบให้เป็นรหัส BCD 8421 ดังนี้

ตาราง Truth Table

Input										Output			
0	1	2	3	4	5	6	7	8	9	B_3	B_2	B_1	B_0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	0	1	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

วงจรอคกรหัส (Decoder) หมายถึง วงจรที่ทำหน้าที่เปลี่ยนรหัสใดรหัสหนึ่ง ซึ่งอยู่ในรูปเลขฐานสองที่ถูกป้อนเข้ามาทางอินพุท ให้กลายเป็นรหัสอื่นตามต้องการ โดยที่แต่ละสายสัญญาณเอาท์พุทอาจจะถูกจัดลำดับตามความสำคัญหรือ ได้รับการจัดหมวดหมู่ตามความเหมาะสม วงจรเข้ารหัสและอคกรหัสนี้ สามารถสร้างขึ้นมาจากการใช้ Diode ที่ต่อ กันในลักษณะ Matrix หรืออาจสร้างจากเกตที่ต่อเป็นวงจร Combination Logic ก็ได้ ตัวอย่างของวงจรอคกรหัส ได้แก่

วงจรที่รับสัญญาณalogic ในรูปของรหัส BCD แล้วแปลงให้ออกมาเป็นเลขฐานสิบ โดยการอคกรหัส สัญญาณที่เป็นเลขฐานสองจำนวน n เส้น จะให้สัญญาณออกสูงสุดจำนวน 2^n เส้น เช่น การอคกรหัส BCD to Decimal Decoder มีรหัสเลขฐานสองเป็นอินพุทจำนวน 4 เส้น และให้ออกท์พุทสัญญาณออก 10 เส้น



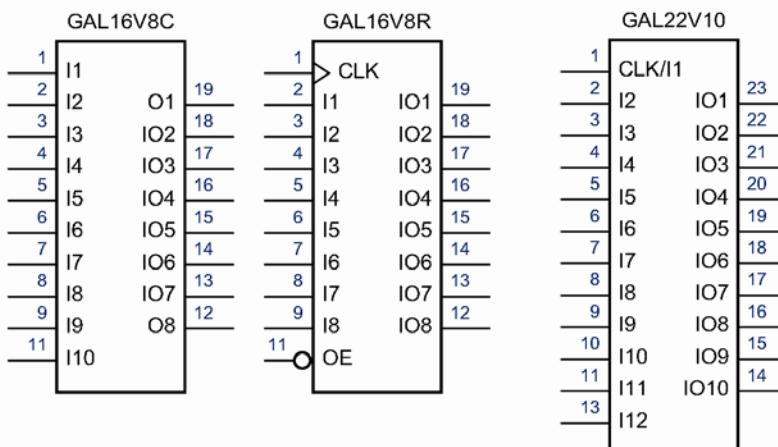
รูปแสดง Block Diagram การแปลง Binary เป็น 7 Segment

จากรูปเป็น Block Diagram ของวงจร BCD to 7 Segment Decoder ที่ทำหน้าที่เปลี่ยนรหัส BCD ให้ไปแสดงผลบน LED ในลักษณะ Seven Segment ซึ่งการถอดรหัสแบบนี้จะได้สัญญาณออกพร้อมมา กันหลายสัญญาณที่แสดงสภาวะต่างๆ กัน โดยการประยุกต์ใช้งานของวงจรถอดรหัสที่มีอินพุตเป็น BCD และมีเอาท์พุตเป็น LED ต่อกันเป็น 7 Segment เพื่อแสดงผลให้ออกมาเป็นเลขฐานสิบ และมี Truth Table ของวงจรดังตาราง

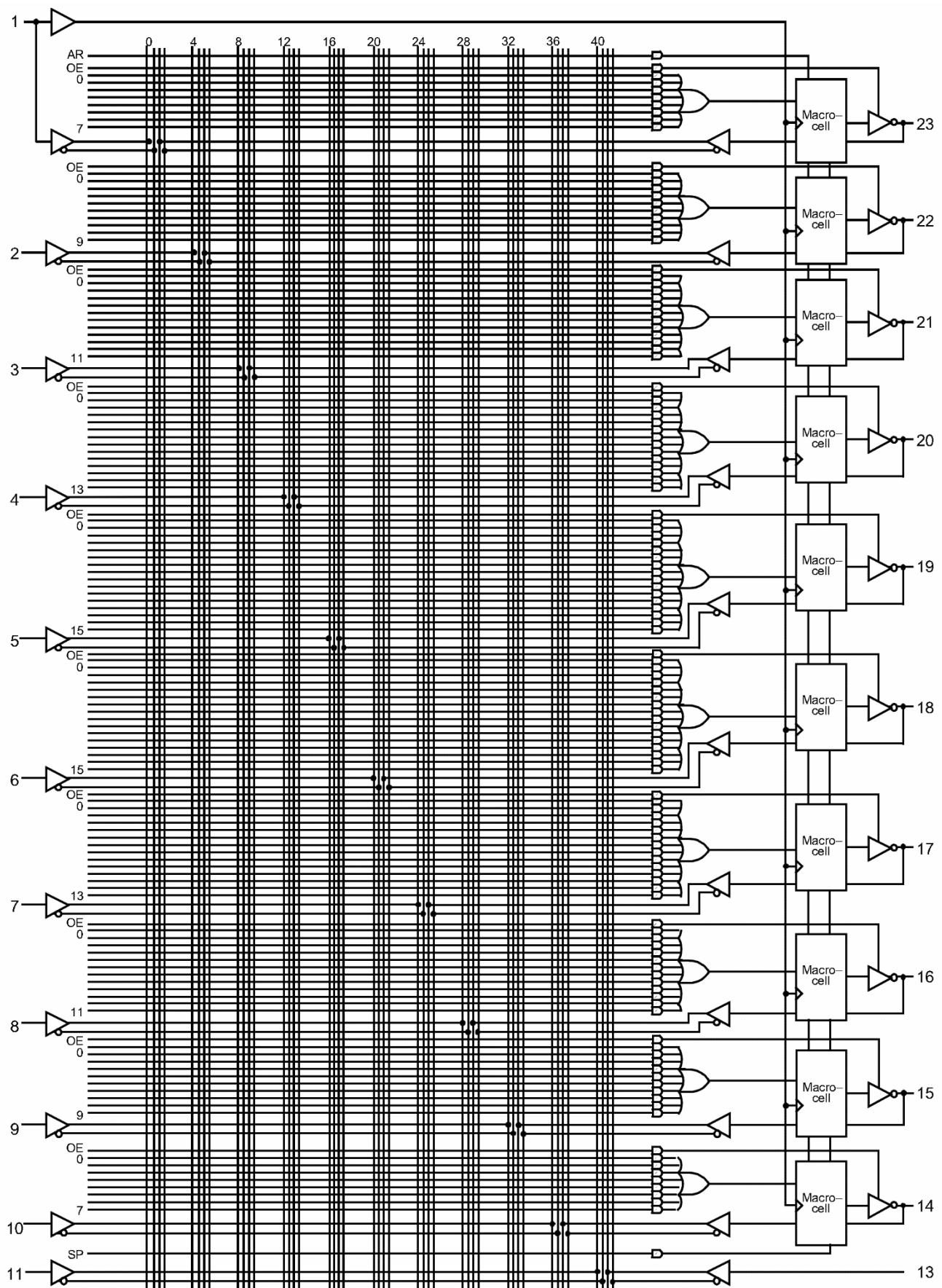
ตาราง Truth Table

Input				Output							Digit
B ₃	B ₂	B ₁	B ₀	A	b	C	d	e	f	g	
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	1	0	1	1	9

การทดลอง จะออกแบบวงจรโดยใช้ไอซี 22V10 เพื่อโปรแกรมให้จำลองการทำงานของไอซี 7442 ที่เป็นวงจร Combination Logic ที่ประกอบด้วย AND Gate และ NOT Gate ทำหน้าที่แปลงรหัส BCD เป็นเลขฐานสิบ (BCD To Decimal Decoder) ซึ่งเอาท์พุตที่เวลาใดจะขึ้นอยู่กับอินพุตในเวลาหนึ่ง เท่านั้น ดังนั้นจะต้องมีสายสัญญาณอินพุตต่อไปตลอดเวลา ภายในไอซี 22V10 นี้จะสามารถโปรแกรมให้มีจำนวนเอาท์พุตได้มากกว่าไอซี 16V8 โดยสามารถกำหนดขาได้ดังรูป



วงศ์รายในแสดงได้ดังรูป



1. ให้ทคลองป้อนโปรแกรม GATE1.PLD ที่ใช้ออชี 16V8 เพื่อใช้ทำเป็น Gate ชนิดต่างๆ

```

Name      GATE1;
PartNo   IOT01;
Revision 01;
Date     30/7/22;
Designer Engineer;
Company  KMITL;
Assembly None;
Location None;
Device   g16v8;

/** Input Pins */
Pin 2 = a;
Pin 3 = b;
Pin 4 = c;
Pin 5 = d;

/** Output Pins */
Pin 12 = not;
Pin 13 = and;
Pin 14 = nand;
Pin 15 = or;
Pin 16 = nor;
Pin 17 = xor;
Pin 18 = xnor;
Pin 19 = F1;

/** Logic Equations */
Not    = !a;                      /* not gate */
And    = a & b;                   /* and gate */
Nand   = !(a & b);               /* nand gate */
Or     = a # b;                   /* or gate */
Nor    = !(a # b);                /* nor gate */
Xor    = a $ b;                   /* exclusive or gate */
Xnor   = !(a $ b);                /* exclusive nor gate */

!F1    = c & xnor;
F.loe = d;

```

การทดสอบการทำงานของ ออชี PLD ที่ทำขึ้นว่าสามารถทำงานได้ถูกต้องตามที่ออกแบบไว้หรือไม่ ทำได้โดยการสร้าง Simulation file (SI) ที่เขียนเป็น Timing diagram โดยที่จะกำหนดค่าเอาท์พุทไว้ก่อนหรือไม่ กำหนดค่าได้ ซึ่งโปรแกรมจะหาค่าเอาท์พุทให้จากการที่ทำขึ้น

2. ให้ทคลองป้อนโปรแกรม GATE1.SI ที่ได้เป็นดังนี้

```

Name      GATE1;
PartNo   IOT01;
Date     30/7/22;
Revision 01;
Designer Engineer;
Company  KMITL;
Assembly None;
Location None;
Device   g16v8;

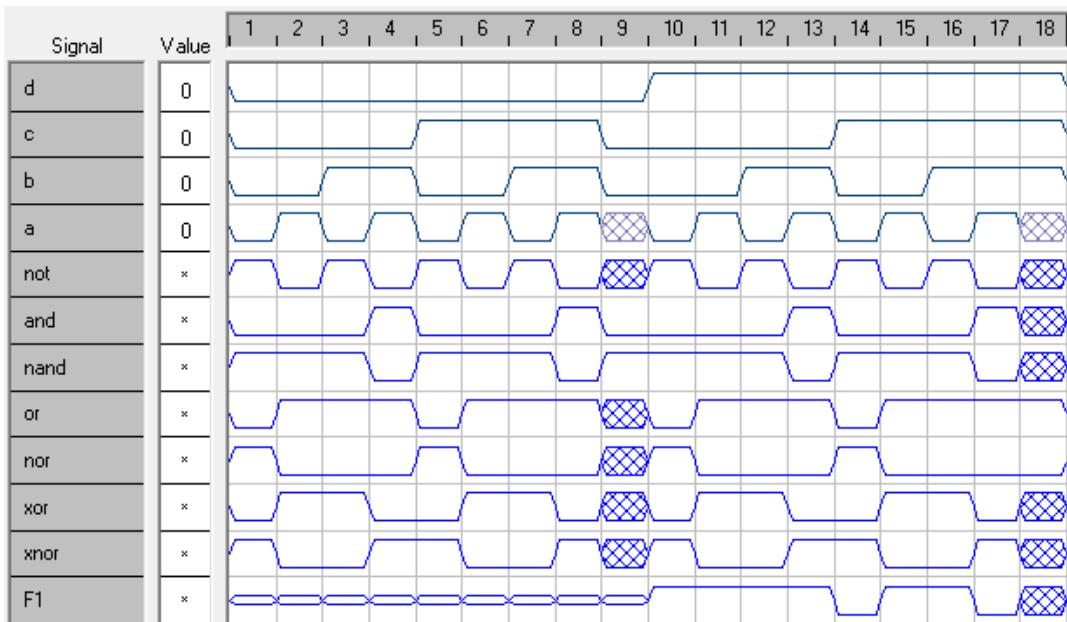
```

ORDER: d, c, b, a, not, and, nand, or, nor, xor, xnor, F1;

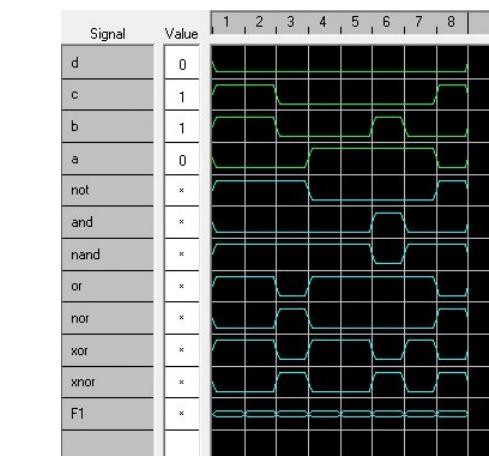
VECTORS:

- 0 0000*****
- 1 0001*****
- 2 0010*****
- 3 0011*****
- 4 0100*****
- 5 0101*****
- 6 0110*****
- 7 0111*****
- 000X*****
- 8 1000*****
- 9 1001*****
- 10 1010*****
- 11 1011*****
- 12 1100*****
- 13 1101*****
- 14 1110*****
- 15 1111*****
- 111X*****

3. ให้ทดสอบการทำงานของไอซี PLD ที่สร้างขึ้นโดยใช้ Simulation ให้เป็น Timing diagram เพื่อทดสอบการทำงานของวงจร Gate ต่างๆ ซึ่งจะต้องได้ดังรูป



4. ให้ทดสอบองค์กร์ในโปรแกรม GATE1.SI ที่ได้กำหนดให้อินพุทเป็นรหัสบัตเตอร์ทั้ง 8 หลัก โดยในแต่ละหลักให้แปลงเป็นเลขฐานสอง ป้อนเข้าที่อินพุทขา a,b,c,d แล้วให้ทดสอบการทำงานเป็น Timing diagram



5. ให้ทดสอบป้อนโปรแกรม GATE2.PLD ที่ใช้อิอี 22V10 เพื่อใช้ทำเป็น Gate ชนิดต่างๆ

```

Name      GATE2;
PartNo   IOT02;
Revision 01;
Date     30/7/22;
Designer Engineer;
Company  KMITL;
Assembly None;
Location None;
Device   g22V10;

/** Input Pins */
Pin 2 = a;
Pin 3 = b;
Pin 4 = c;
Pin 5 = d;

/** Output Pins */
Pin 15 = not;
Pin 16 = and;
Pin 17 = nand;
Pin 18 = or;
Pin 19 = nor;
Pin 20 = xor;
Pin 21 = xnor;
Pin 22 = F1;

/** Logic Equations */
not  = !a;                      /* not gate           */
and  = a & b;                  /* and gate           */
nand = !(a & b);              /* nand gate          */
or   = a # b;                  /* or gate            */
nor  = !(a # b);              /* nor gate           */
xor  = a $ b;                  /* exclusive or gate */
xnor = !(a $ b);              /* exclusive nor gate */

!F1   = c & xnor;
F1.oE = d;

```

6. ให้แก้ไขไฟล์โปรแกรม GATE1.SI ที่ได้เป็น GATE2.SI ดังนี้

Name จาก GATE1 ไปเป็น GATE2

Device จาก g16v8 ไปเป็น g22V10

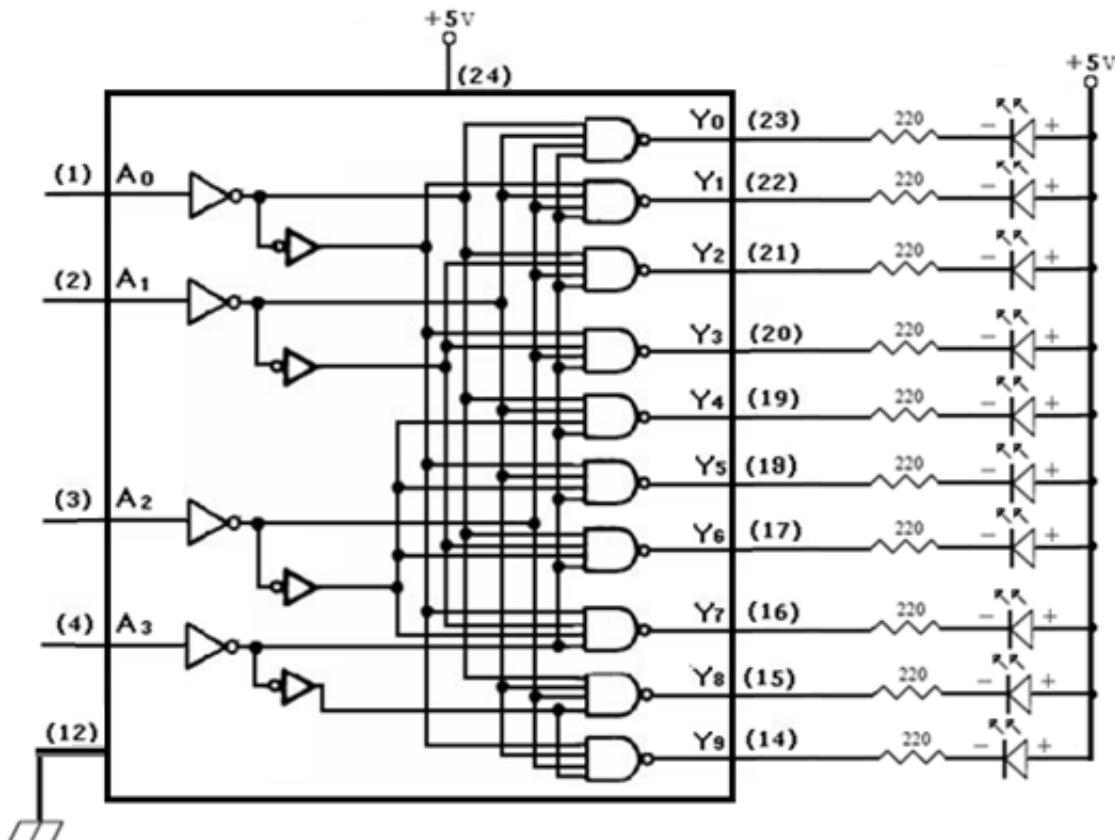
7. ทำการทดสอบการทำงานของไออี PLD ที่ทำขึ้นว่าสามารถทำงานได้ถูกต้องตามที่ออกแบบไว้หรือไม่ โดยการทดสอบ Simulation file (SI) ที่เขียนเป็น Timing diagram
8. ให้อธิบายผลการทดสอบที่ได้ และความแตกต่างของโปรแกรม GATE1 ที่ใช้อิอี 16V8 กับ GATE2 ที่ใช้อิอี 22V10
-
-
-
-
-

9. ให้เขียนโปรแกรมโดยใช้ไอซี 22V10 เพื่อจำลองการทำงานของไอซี 7442 ที่ทำหน้าที่เป็น 4-Line BCD To 10-Line Decimal Decoders โดยมีตัวอย่างของวงจรดังรูปด้านล่างและให้กำหนดตำแหน่งตามโปรแกรมดังนี้

```
/** Input Pins */
PIN [1..4] = [A0..3];      /* BCD 4-bit number */

/** Output Pins */
PIN [14..23] = [Y9..0];    /* Decimal BCD 10-bit number */
```

10. ประกอบวงจรที่จะใช้ในการทดลองเพื่อจำลองการทำงานของไอซี 7442 โดยใช้ไอซี 22V10 ดังรูป โดยเอาท์พุทจะต่อผ่านตัวความด้านทานค่า 220 โอห์ม และ LED เข้ากับไฟ +5 V. เพื่อขับ LED ให้สว่าง การทำงานของวงจรจะเป็นแบบ Active Low คือเมื่อได้ผลลัพธ์เป็นเลขฐานสิบตามที่ต้องการแล้ว วงจรจะให้อเอาท์พุทออกมา 1 เส้นมีสถานะเป็นลอจิก 0 (Low) ส่งผลให้ LED ที่ต่อ กับขาันนี้สว่างเพียงขาเดียว ส่วนเอาท์พุทขาที่เหลือจะมีสถานะเป็นลอจิก 1 ทั้งหมด



11. ให้ทดลองป้อนไฟเข้าที่อินพุตพร้อมกันจำนวน 4 เส้น และบันทึกค่าผลการทำงานที่ได้ลงในตาราง

Input				Output									เลขฐาน 10	
A3	A2	A1	A0	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9	
0	0	0	0											
0	0	0	1											
0	0	1	0											
0	0	1	1											
0	1	0	0											
0	1	0	1											
0	1	1	0											
0	1	1	1											
1	0	0	0											
1	0	0	1											
1	0	1	0											
1	0	1	1											
1	1	0	0											
1	1	0	1											
1	1	1	0											
1	1	1	1											

12. วงจรนี้ทำงานเป็นเลขฐานสิบ ได้ช่วงระหว่างเลขศูนย์ถึงเท่าไร
13. เมื่อป้อนอินพุทที่นอกเหนือจากเลขฐานสิบ เอาท์พุทที่ได้จะเป็นเช่นไร

.....

14. ถ้าอินพุของวงจรไม่ได้ถูกต่อไว้ จะได้อาท์พุเป็นอย่างไร

-
15. การทดลองให้นำแต่ละขาเอาท์พุ Q ของ D flipflop ที่สร้างเป็นวงจรนับ ที่ใช้ไอซี 16V8 ขาที่ 14 ถึง 17 ต่อเข้ามาเป็นอินพุ ขาที่ 1 ถึง 4 ของวงจร 4-Line BCD To 10-Line Decimal Decoders นี้ และ อธิบายวิธีการทำงาน และผลการทดลองที่ได้
-
-
-
-

16. ให้เขียนโปรแกรมโดยใช้ไอซี 22V10 เพื่อสร้างวงจรที่ทำงานเป็น BCD to 7 Segment Decoder ที่ทำหน้าที่เปลี่ยนรหัส BCD ให้ไปแสดงผลบน Seven Segment

```
/** Input Pins */
PIN [1..4] = [B0..3]; /* BCD 4-bit number */

/** Output Pins */
PIN 23 = a; /* 7-Segment Output a */
PIN 22 = b; /* 7-Segment Output b */
PIN 21 = c; /* 7-Segment Output c */
PIN 20 = d; /* 7-Segment Output d */
PIN 19 = e; /* 7-Segment Output e */
PIN 18 = f; /* 7-Segment Output f */
PIN 17 = g; /* 7-Segment Output g */
```

$b_0 b_1$	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	d	d	d	d
10	1	1	d	d

$$a = \bar{b}_1 \bar{b}_3 + b_0 \bar{b}_2 \\ + b_1 b_3 + b_2$$

$b_0 b_1$	00	01	11	10
00	1	0	1	1
01	1	0	1	0
11	d	d	d	d
10	1	1	d	d

$$b = \bar{b}_1 + \bar{b}_2 \bar{b}_3 + b_2 b_3$$

$b_0 b_1$	00	01	11	10
00	1	1	1	0
01	1	1	1	1
11	d	d	d	d
10	1	1	d	d

$$c = \bar{b}_2 + b_1 + \bar{b}_0 b_3$$

$b_0 b_1$	00	01	11	10
00	1	0	1	1
01	0	1	0	1
11	d	d	d	d
10	1	1	d	d

$$d = \bar{b}_1 \bar{b}_3 + b_0 + \bar{b}_1 b_2 \\ + b_1 \bar{b}_2 b_3 + b_1 b_2 \bar{b}_3$$

$b_0 b_1$	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	d	d	d	d
10	1	0	d	d

$$e = \bar{b}_1 \bar{b}_3 + b_2 \bar{b}_3$$

$b_0 b_1$	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11	d	d	d	d
10	1	1	d	d

$$f = b_0 + \bar{b}_2 \bar{b}_3 + b_1 \bar{b}_2 \\ + b_1 \bar{b}_3$$

$b_0 b_1$	00	01	11	10
00	0	0	1	1
01	1	1	0	1
11	d	d	d	d
10	1	1	d	d

$$g = b_0 + \bar{b}_1 b_2 + b_2 \bar{b}_3 + b_1 \bar{b}_2$$

261877072
6570211712071

66011314

```

Name      segment;
PartNo   ITE01;
Revision 01;
Date     15/1/24;
Designer Engineer;
Company  KMITL;
Assembly None;
Location None;
Device   gl6v8;

/** Input Pins */
Pin 2 = b0;
Pin 3 = b1;
Pin 4 = b2;
Pin 5 = b3;

/** Output Pins */
Pin 12 = a;
Pin 13 = b;
Pin 14 = c;
Pin 15 = d;
Pin 16 = e;
Pin 17 = f;
Pin 18 = g;

/** Logic Equations */
a = !b1 & !b3 # b0 & !b2 # b1 & b3 # b2;
b = !b1 # !b2 & !b3 # b2 & b3;
c = !b2 # b1 # !b0 & b3;
d = !b1 & !b3 # b0 # !b1 & b2 # b1 & !b2 & b3 # b1 & b2 & !b3;
e = !b1 & !b3 # b2 & !b3;
f = b0 # !b2 & !b3 # b1 & !b2 # b1 & !b3;
g = b0 # !b1 & b2 # b2 & !b3 # b1 & !b2;

```

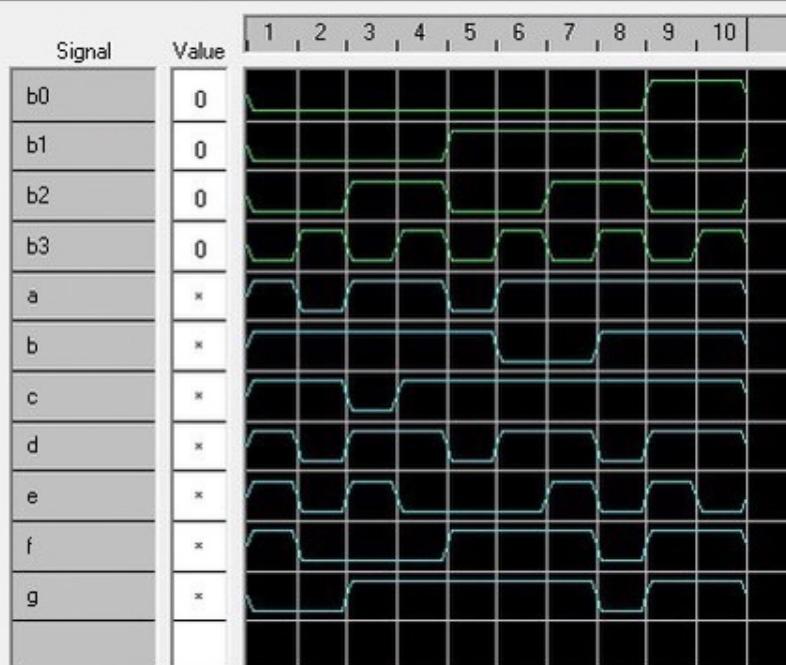
```

Name      segment;
PartNo   ITE01;
Date     15/1/24;
Revision 01;
Designer Engineer;
Company  KMITL;
Assembly None;
Location None;
Device   gl6v8;

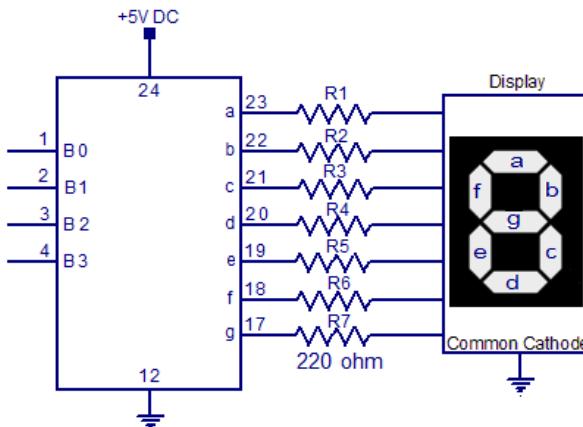
ORDER: b0, b1, b2, b3, a, b, c, d, e, f, g;

VECTORS:
0000*****
0001*****
0010*****
0011*****
0100*****
0101*****
0110*****
0111*****
1000*****
1001*****

```



17. ประกอบวงจรที่จะใช้ในการทดลองดังรูป ทดลองการทำงานของวงจร โดยให้อาทีพุทธของวงจร ต่อผ่านด้วยความต้านทานค่า 220 โอห์ม แล้วต่อเข้าตัว 7 segment Common cathode LED display และนำขา Common cathode ของ 7 segment ต่อลงกราวด์ ผลลัพธ์ที่ได้ส่งผลให้ LED ส่วน哪แสดง เป็นเลขฐานสิบตามค่าที่ป้อนเข้ามาที่อินพุททั้ง 4 เส้น ส่วนตำแหน่งขาของ 7-segment ให้ดูจาก Datasheet ของ 7-segment เบอร์นี้



18. ให้ทดลองป้อนไฟเข้าที่อินพุทร่วมกันจำนวน 4 เส้น และบันทึกค่าผลการทดลองที่ได้ลงในตาราง

Input				Output							เลขฐาน 10
B3	B2	B1	B0	a	b	c	D	e	f	g	
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	1	0	1	1	9

19. การทดลองให้นำแต่ละขาอาทีพุท Q ของ D flipflop ที่สร้างเป็นวงจรนับ ที่ใช้ออชี 16V8 ขาที่ 14 ถึง 17 ต่อเข้ามาเป็นอินพุท ขาที่ 1 ถึง 4 ของวงจร BCD to 7 Segment Decoder นี้ และอธิบายวิธีการทำ และผลการทดลองที่ได้เมื่อเปรียบเทียบกับวงจร 4 Line BCD To 10-Line Decimal Decoders
-
-
-
-
-

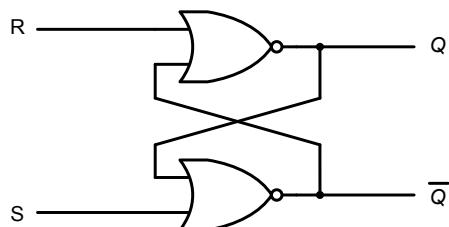
วงจร Logic แบ่งออกเป็น 2 ประเภท คือ

1. วงจร Combination เป็นวงจรที่สัญญาณเอาท์พุตจะขึ้นอยู่กับสัญญาณอินพุต ณ เวลาเดียวกัน

2. วงจร Sequential เป็นวงจรที่สัญญาณเอาท์พุตจะขึ้นอยู่กับสัญญาณอินพุต ณ เวลาต่อๆ กัน และ อินพุตที่เกิดขึ้นในอดีตด้วย ดังนั้นจึงต้องมีหน่วยความจำเพื่อกีบสถานะของวงจรในขณะนั้นหรือหรือ เก็บค่าสัญญาณอินพุตที่เกิดขึ้นก่อนหน้านั้น

Flip-Flop เป็นหน่วยความจำประเภทหนึ่งที่มีคุณสมบัติสามารถเก็บรักษาข้อมูลไว้ได้ มีสถานะ เสถียรอยู่ 2 สถานะ (Bistable) คือ ลอจิก 0 และ 1 โดย Flip-Flop หนึ่งตัวสามารถเก็บตัวเลขฐานสองได้ขนาด 1 บิต ประกอบด้วยสายเอาท์พุต 2 เส้น ซึ่งเอาท์พุตทั้งสองจะมีค่าตรงข้ามกันเสมอหรือเอาท์พุตจะ Complement ซึ่งกันและกันอยู่ ส่วนสายสัญญาณทางด้านอินพุตจะมีจำนวนเส้นต่างกันขึ้นอยู่กับว่าจะเป็น Flip-Flop ชนิดใด ค่าที่ป้อนทางอินพุตจะทำให้สภาวะของ Flip-Flop เปลี่ยนไป Flip-Flop ที่ใช้งานทั่วไปมีดังนี้

RS Flip-Flop จะมี 2 อินพุต คือ R (Reset) และ S (Set) และมี 2 เอาท์พุต คือ Q และ \bar{Q} โครงสร้าง ของ RS Flip-Flop แบบพื้นฐานที่สร้างจาก NOR Gate เป็นดังรูป



รูปแสดงวงจร RS Flip-Flop

การทำงานของ RS Flip-Flop มีดังนี้

- ค่าของอินพุต R เป็น 1 และ S เป็น 0 จะเป็นการ Reset ทำให้อเอาท์พุต Q มีค่าเป็น 0 และ \bar{Q} มีค่าเป็น 1
- ค่าของอินพุต R เป็น 0 และ S เป็น 1 จะเป็นการ Set ทำให้อเอาท์พุต Q มีค่าเป็น 1 และ \bar{Q} มีค่าเป็น 0
- ค่าของอินพุต R เป็น 0 และ S เป็น 0 เมื่อมองกัน จะทำให้อเอาท์พุต Q และ \bar{Q} มีค่าเหมือนเดิม เสมือนวงจร ไม่ได้ทำงาน
- ค่าของอินพุต R เป็น 1 และ S เป็น 1 เมื่อมองกัน จะทำให้อเอาท์พุต Q และ \bar{Q} มีค่าเหมือนกัน ซึ่งไม่ตรง ตามคุณสมบัติของ Flip-Flop ดังนี้จะต้องกำหนดค่าไม่ให้วางเรเกิดสถานะนี้ขึ้น

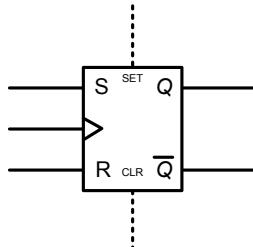
เราเขียนตารางความจริง (Truth Table) ของ RS Flip-Flop ได้ดังนี้

Truth Table ของ RS Flip Flop

R	S	Q	\bar{Q}
0	0	Q	\bar{Q}
0	1	1	0
1	0	0	1
1	1	-	-

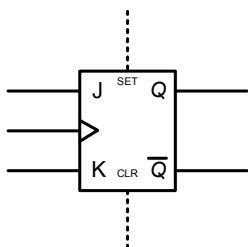
No Change
Set
Reset
Not Used

เราสามารถเพิ่มขาเข้าไปใน Flip-Flop โดยมีขา Clock เป็นสัญญาณนาฬิกาควบคุมการเปลี่ยนสภาวะ ซึ่งเอาท์พุตจะมีการเปลี่ยนสภาวะตามเงื่อนไข คือ เมื่อมีสัญญาณ Clock เป็นลอจิก 1 แต่ถ้าสัญญาณ Clock เป็นลอจิก 0 เอาท์พุตจะไม่เปลี่ยนสภาวะ และในบางกรณี เราอาจจำเป็นต้องทำให้อเอาท์พุต Q เป็นลอจิก 0 หรือลอจิก 1 โดยไม่ต้องมีสัญญาณ Clock มากратตุน ดังนั้นเราวิจัยจะเพิ่มขา Clear และ Preset เข้าไป เพื่อที่จะให้สามารถกำหนดค่าของเอาท์พุต Q ได้โดยตรง โดยจะได้สัญลักษณ์ของ RS Flip-Flop แสดงดังรูป



รูปแสดงสัญลักษณ์ RS Flip-Flop

JK Flip-Flop จะมี 2 อินพุต คือ J และ K และมี 2 เอาท์พุต คือ Q และ \bar{Q} การทำงานของ JK Flip-Flop มีลักษณะการทำงานเหมือนกับ RS Flip-Flop แต่มีคุณลักษณะเพิ่มเติมคือ สามารถกำหนดสภาวะทางลอจิกของอินพุตทั้งสองเส้นให้เป็นลอจิก 1 พร้อมกันได้ โดยมีสัญลักษณ์ดังนี้



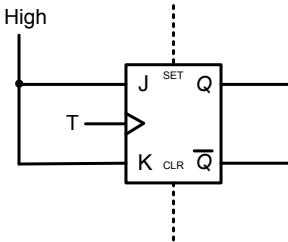
รูปแสดงสัญลักษณ์ JK Flip-Flop

การทำงานของ JK Flip-Flop ในสภาวะที่มีสัญญาณ Clock เข้ามามีดังนี้ เมื่ออินพุต J เป็น 0 และ K เป็น 0 เอาท์พุต จะไม่เปลี่ยนแปลง ถ้าอินพุต J เป็น 0 และ K เป็น 1 จะทำให้อเอาท์พุต Q เป็น 0 ถ้าอินพุต J เป็น 1 และ K เป็น 0 จะทำให้อเอาท์พุต Q เป็น 1 แต่ถ้าอินพุต J เป็น 1 และ K เป็น 1 จะทำให้อเอาท์พุตเปลี่ยนสภาวะเป็นสภาวะตรงกันข้ามกับสภาวะเดิม โดยมี Truth Table ของ JK Flip-Flop ดังนี้

Truth Table ของ JK Flip Flop

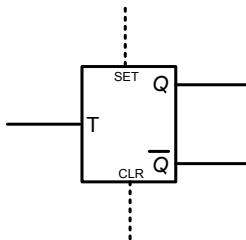
J	K	Q	\bar{Q}	
0	0	Q	\bar{Q}	No Change
0	1	0	1	Reset
1	0	1	0	Set
1	1	\bar{Q}	Q	Toggle

T Flip-Flop จะมีอินพุตเพียงเส้นเดียว คือ T และมี 2 เอาท์พุต คือ Q และ \bar{Q} เราสามารถนำเอา JK Flip-Flop มาทำเป็น T Flip-Flop ได้ โดยนำอินพุตทั้งสองเส้นคือ J และ K ไปต่อ กันไฟฟูงหรือลอจิก 1 แล้วนำขา Clock ของ JK Flip-Flop มาเป็นขา T ดังรูป



รูปแสดงวงจร T Flip-Flop

โดยมีสัญลักษณ์ของ T Flip-Flop ดังนี้



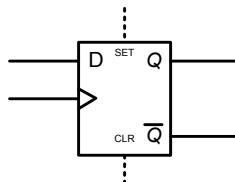
รูปแสดงสัญลักษณ์ T Flip-Flop

ลักษณะการทำงานของ T Flip-Flop จะเปลี่ยนสถานะทุกครั้งที่มีอินพุต T ป้อนเข้ามา ถ้าเอาท์พุตอยู่ในสถานะ โลจิก 0 เมื่อมีอินพุต T ป้อนเข้ามาจะเปลี่ยนสถานะเอาท์พุตเป็นโลจิก 1 และถ้าอินพุต T อันดับต่อมาป้อนเข้ามาอีกเอาท์พุตจะเปลี่ยนสถานะกลับเป็นโลจิก 0 หรืออาจกล่าวได้ว่าทุกครั้งที่มีอินพุต T ป้อนเข้ามาจะทำให้อเอาท์พุต Q เปลี่ยนสถานะเป็นสถานะตรงกันข้าม เปรียบเสมือนเป็นวงจรหารสอง โดยมี Truth Table ของ T Flip-Flop ดังนี้

Truth Table ของ T Flip Flop

T	Q	\bar{Q}	
0	Q	\bar{Q}	No Change
1	\bar{Q}	Q	Toggle

D Flip-Flop จะมีอินพุตที่เป็นข้อมูลคือ D และจะต้องมีอินพุตที่เป็นสัญญาณนาฬิกา (Clock) ด้วย โดยมี 2 เอาท์พุต คือ Q และ \bar{Q} โดยมีสัญลักษณ์ดังนี้



รูปแสดงสัญลักษณ์ D Flip-Flop

การทำงานของ D Flip-Flop ในสถานะที่สัญญาณ Clock เป็นโลจิก 0 ค่าของ D ไม่ว่าจะเป็นโลจิก 0 หรือ 1 จะไม่มีผลต่อเอาท์พุต คือเอาท์พุตจะไม่มีการเปลี่ยนสถานะ แต่ถ้าสัญญาณ Clock เป็นโลจิก 1 ค่า

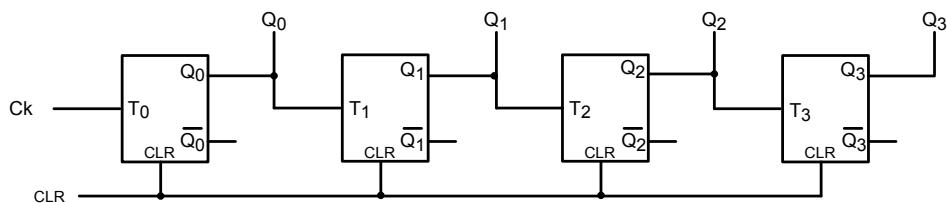
เอาท์พุตของ D Flip-Flop จะเปลี่ยนสถานะตามค่าอินพุต D ที่เข้ามา คือ ถ้า D เป็นโลจิก 0 จะได้อาท์พุต Q มีค่าเป็น 0 และถ้าอินพุต D เป็นโลจิก 1 จะได้อาท์พุต Q มีค่าเป็น 1 โดยมี Truth Table ของ D Flip-Flop ดังนี้

Truth Table ของ RS Flip Flop

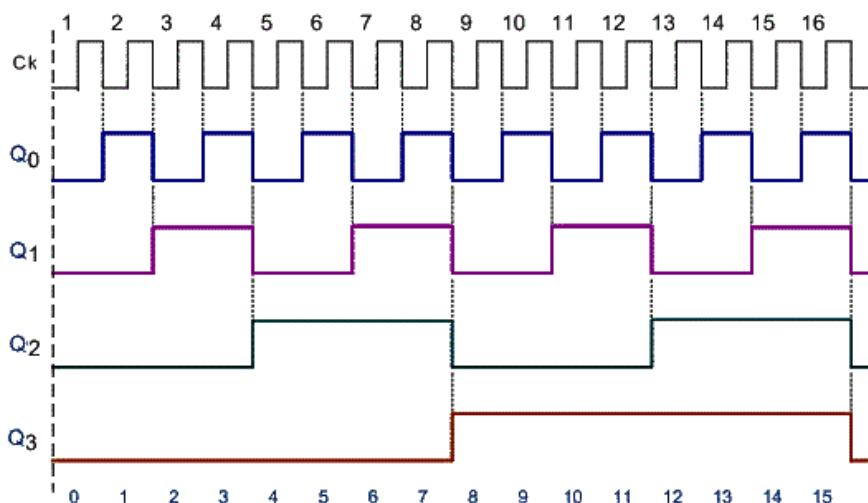
Clock	D	Q	\bar{Q}
1	0	0	1
1	1	1	0
0	x	Q	\bar{Q}

Reset
Set
No Change

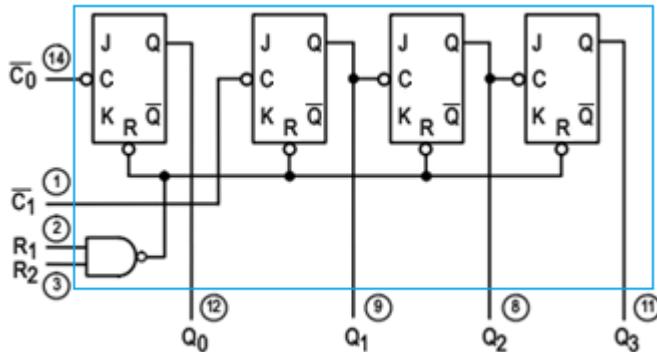
วงจรนับเลขฐาน 2 (Binary Ripple Counter) เป็นการประยุกต์ใช้งานของ T Flip-Flop โดยใช้หลักว่าที่เอาท์พุตของ T Flip-Flop จะถูกเปลี่ยนสถานะสลับกันไปมา (Toggle) ระหว่างโลจิก 0 และโลจิก 1 ทุกครั้งที่มีสัญญาณอินพุตป้อนเข้ามาทางขา T ดังนั้น Flip-Flop หนึ่งตัวสามารถนับได้ 2 ค่า คือ 0 และ 1 ถ้านำ T Flip-Flop มาต่อเข้าด้วยกันแบบอนุกรม 2 ตัวก็จะสามารถนับได้ 4 ค่า ถ้า 3 ตัวจะนับได้ 8 ค่า และถ้า 4 ตัวก็จะนับได้ ทั้งหมด 16 ค่า การต่อวงจรแสดงให้ดังรูป โดยถ้าต้องการให้เริ่มต้นนับใหม่ จะต้องกำหนดให้ขา CLR เป็น 1 ซึ่งจะทำให้อาท์พุตทุกเส้นกลับไปเป็น 0



ความถี่ของสัญญาณที่เอาท์พุตของ Flip-Flop แต่ละตัวจะเป็นวงจรหารความถี่เมื่อเทียบกับความถี่ของสัญญาณอินพุต โดยสัญญาณเอาท์พุตของ T Flip-Flop คือ Q_0 ที่ได้จากการด้วย 2 Q_1 จะถูกหารด้วย 4 Q_2 จะถูกหารด้วย 8 และ Q_3 จะถูกหารด้วย 16 เมื่อเทียบกับสัญญาณอินพุต โดยมี Timing Diagram แสดงดังรูป

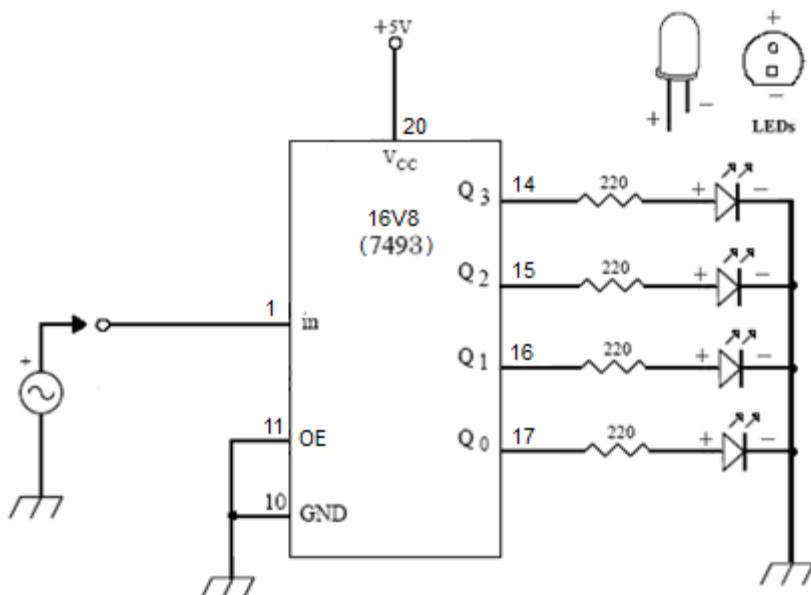


การทดลองการทำงานของไอซี 7493 เป็นวงจรนับเลขฐาน 2 (Binary Ripple Counter) ขนาด 4 bit ประกอบไปด้วย JK flip-flops จำนวน 4 ตัว โดยที่ขา J และ K ภายในตัวไอซีจะถูกต่ออยู่กับไฟบวก ดังนั้นโครงสร้างในลักษณะนี้จึงเรียกว่า T flip-flop ที่เอาท์พุต Q ของไอซีจะถูกเปลี่ยนสถานะ สลับกันไปมา (Toggle) ระหว่างเลขจิก 0 และ เลขจิก 1 ทุกครั้งที่มีสัญญาณป้อนเข้ามาทางขา 14 จำนวน 1 ลูก โดยภายในไอซีจะเป็น T flip-flop มาต่ออนุกรมกัน 4 ตัว ดังแสดงในรูปที่



ความถี่สัญญาณที่เอาท์พุตของ Flip-Flop แต่ละตัวจะเป็นวงจรอาร�ณ์ถี่เมื่อเทียบกับความถี่ของสัญญาณอินพุต โดยสัญญาณเอาท์พุต Q ที่ได้จะถูกหารด้วย 2 ที่ขา Q₀ , หารด้วย 4 ที่ขา Q₁ , หารด้วย 8 ที่ขา Q₂ และหารด้วย 16 ที่ขา Q₃ เอาท์พุตทั้ง 4 ของไอซี 7493 ยังสามารถแสดงการนับจำนวนของสัญญาณพัลส์ที่อินพุตได้อีกด้วย

1. ประกอบวงจรที่จะใช้ในการทดลองเพื่อแสดงถึงการทำงานของไอซี 7493 โดยใช้ไอซีที่ โปรแกรมได้เบอร์ 16V8 แทนคั่งรูป โดยมีตัว LED จะต่อ กับตัวด้านหนาค่า 220 โอห์ม ในลักษณะอนุกรมแต่ละขาเอาท์พุต Q ของไอซี ไปยังกราวด์ 7493 เพื่อแสดงผลของสัญญาณเอาท์พุตในแต่ละขาเมื่อมีเลขอじิกเป็น 1 จะทำให้ LED สว่าง



รูป แสดงการทำงานของวงจrnับเลขฐานสองโดยใช้ไอซี 16V8

----- ไฟล์ Counter7493.PLD ที่ใช้ในการสร้างวงจร

```

Name      Counter7493;
PartNo   IOT001;
Date     30/7/22;
Revision 01;
Designer Engineer;
Company  KMITL;
Assembly None;
Location None;
Device   g16v8;

/** Input Pins */
PIN 1      = CLK;           /* register clock */
PIN 11     = !OE;          /* register output enable */

/** Output Pins */
PIN [14..17] = [Q3..0];    /* register 4-bit */

/** Logic Equations */
field STATE = [Q3..0];    /* declare state bit field */
#define S0 'b'0000           /* define state */
#define S1 'b'0001
#define S2 'b'0010
#define S3 'b'0011
#define S4 'b'0100
#define S5 'b'0101
#define S6 'b'0110
#define S7 'b'0111
#define S8 'b'1000
#define S9 'b'1001
#define S10 'b'1010
#define S11 'b'1011
#define S12 'b'1100
#define S13 'b'1101
#define S14 'b'1110
#define S15 'b'1111

Sequenced STATE {
present S0      next S1;
present S1      next S2;
present S2      next S3;
present S3      next S4;
present S4      next S5;
present S5      next S6;
present S6      next S7;
present S7      next S8;
present S8      next S9;
present S9      next S10;
present S10     next S11;
present S11     next S12;
present S12     next S13;
present S13     next S14;
present S14     next S15;
present S15     next S0;
}

```

----- ไฟล์ Counter7493.SI ที่ใช้ในการทดสอบวงจร

```

Name      Counter7493;
PartNo   IOT001;
Date     30/7/22;
Revision 01;
Designer Engineer;
Company  KMITL;
Assembly None;
Location None;
Device   g16v8;

ORDER: CLK, !OE, Q0, Q1, Q2, Q3;

VECTORS:
C1**** /* test HI Impedance */
C0**** /* state 1 */
C0**** /* state 2 */
C0**** /* state 3 */
C0**** /* state 4 */
C0**** /* state 5 */
C0**** /* state 6 */
C0**** /* state 7 */
C0**** /* state 8 */
C0**** /* state 9 */
C0**** /* state 10 */
C0**** /* state 11 */
C0**** /* state 12 */
C0**** /* state 13 */
C0**** /* state 14 */
C0**** /* state 15 */
C0**** /* state 0 */
C0**** /* state 1 */

```

----- ไฟล์ COUNTER7493.sim เป็นสมการลอจิกที่โปรแกรมสร้างขึ้น -----

```
%SIGNAL
PIN 1 = CLK
PIN 11 = !OE
PIN 17 = Q0
PIN 16 = Q1
PIN 15 = Q2
PIN 14 = Q3
%END

%FIELD
FIELD STATE = Q3,Q2,Q1,Q0
%END

%EQUATION
Q0.d => !Q0

Q1.d => !Q0 & Q1
# Q0 & !Q1

Q2.d => !Q0 & Q1 & Q2
# Q0 & Q1 & !Q2
# !Q1 & Q2

Q3.d => !Q0 & Q1 & Q2 & Q3
# Q0 & Q1 & Q2 & !Q3
# !Q1 & Q2 & Q3
# !Q2 & Q
%END
```

----- ไฟล์ Counter7493.jed ใช้ในการโปรแกรมพิวส์ลงในตัวไอซ์ โดยใช้เครื่องโปรแกรม -----

```
CUPL(WM)      5.0a Serial# 60008009
Device        g16v8ms Library DLIB-h-40-11
Created       Thu Aug 15 00:43:10 2019
Name          Counter7493
Partno        IOT001
Revision      01
Date          30/7/22
Designer      Engineer
Company       KMITL
Assembly      None
Location      None
*QP20
*QF2194
*QV18
*G0
*F0
*L00512 11111111111011111111111111111111
*L00768 11111111111011011111111111111111
*L00800 11111111111011110111111111111111
*L01024 11111111111011011101111111111111
*L01056 11111111111011110111111111111111
*L01088 11111111111110110111111111111111
*L01280 11111111111011011101111111111111
*L01312 11111111111011101111011111111111
*L01344 11111111111111011011101111111111
*L01376 1111111111111101101111111111
*L02048 0011110001001001010100010000101
*L02080 00110000001100000011000100000000
*L02112 00000000110000111111111111111111
*L02144 11111111111111111111111111111111
*L02176 111111111111111101
*C2E83
*p 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
*v0001 CXXXXXXXN1XXZZZXXN
*v0002 CXXXXXXXN0XXLLLHXXN
*v0003 CXXXXXXXN0XXLLHLXXN
*v0004 CXXXXXXXN0XXLLHHXXN
*v0005 CXXXXXXXN0XXLHLLXXN
*v0006 CXXXXXXXN0XXLHLHXXN
*v0007 CXXXXXXXN0XXLHHHLXXN
*v0008 CXXXXXXXN0XXLHHHXXN
*v0009 CXXXXXXXN0XXHLLLXXN
*v0010 CXXXXXXXN0XXHLLHXXN
*v0011 CXXXXXXXN0XXHLHLXXN
*v0012 CXXXXXXXN0XXHLHHXXN
*v0013 CXXXXXXXN0XXHLLXXN
*v0014 CXXXXXXXN0XXHLHXXN
*v0015 CXXXXXXXN0XXHHHLXXN
*v0016 CXXXXXXXN0XXHHHHXXN
*v0017 CXXXXXXXN0XXLLLXXN
*v0018 CXXXXXXXN0XXLLLHXXN
*_5C50
```

ເນັດວຽກທະນີ ໄກສອງໂທນີ້ 66011314

2. ให้ทดลองป้อนค่าแรงดันไฟเข้าที่อินพุตเป็นລອຈິກ 0 และ 1 โดยใช้ความถี่ໜາກທານ 10K ຕ່ອອນຸกรມກັບ switch ແລ້ວບັນທຶກຄ່າຜູດກາຣໂທດົນທີ່ໄດ້ຈາກເວົາທີ່ພູກລອງໃນຕາງ

Input	Output				
A	Q3	Q2	Q1	Q0	ເລຂງຈານສີບ
1	0	0	0	0	0
1	1	1	0	1	13
1	1	1	1	0	14
1	1	1	1	1	15
1	0	1	1	1	7
1	1	0	0	1	9
1	1	0	1	0	10
1	1	1	0	0	12
1	1	1	0	1	13
1	1	0	0	1	9
1	1	0	0	0	8
1	0	0	1	1	3
1	0	1	0	1	5
1	0	1	1	1	7
1	1	0	1	1	11
1	1	1	0	0	12
1	1	1	1	0	14
1	0	0	1	0	2
1	0	1	0	0	4
1	0	1	0	1	5
1	1	0	0	1	9
1	1	0	1	1	11
1	1	1	0	0	12
1	1	1	1	0	14
1	1	1	1	1	15
1	0	0	1	0	2
1	0	1	1	0	6
1	0	0	0	1	1
1	1	0	0	1	9
1	0	1	0	1	5
1	1	1	0	1	13
1	1	1	1	0	14

2. หาอัตราส่วนเมื่อ LED ที่ขาเอาท์พุต Q₃ กระพริบ 1 ครั้ง จะได้ LED ที่เอาท์พุต Q₀ กระพริบไปกี่ครั้ง 4

3. การกระพริบของ LED ในแต่ละดวงจะมีลักษณะเป็นวงจรนับเลขฐาน 2 โดยนับได้ตั้งแต่ศูนย์จนถึงจำนวนเท่าไร 15

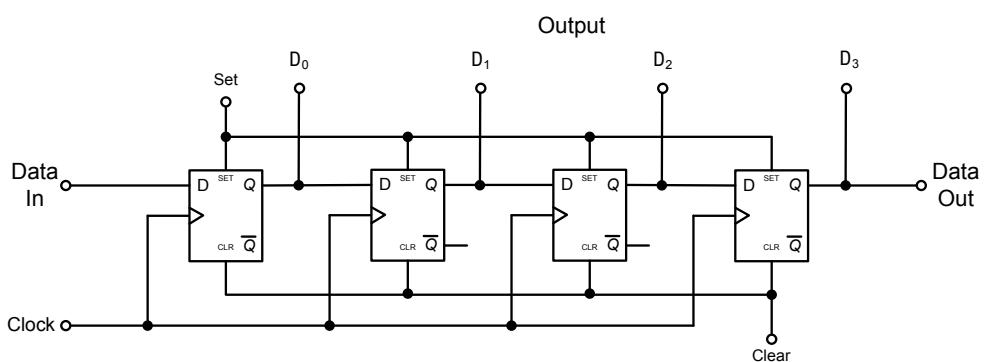
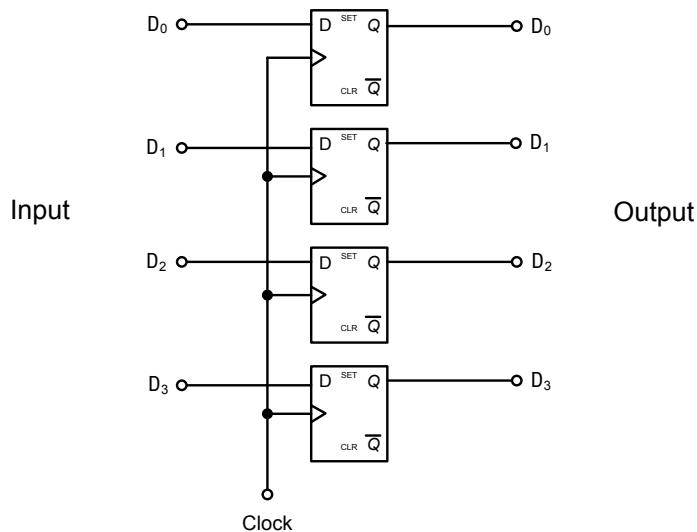
4. ให้นักศึกษาสังเกตว่าสัญญาณเอาท์พุตของไอซีมีการเปลี่ยนแปลงทุกครั้ง เมื่อเทียบกับสัญญาณอินพุต เลขที่ได้เรียงลำดับคิดต่อ กันหรือไม่ เพราะเหตุใด ให้ก็ต่อ ก็ต่อ เพราะ ปุ่มงานนั้นรักษา 0-1 ไม่สามารถหักห้าม ฝังตัว Bounce

รีจิสเตอร์ (Register) ถูกนำมาใช้เป็นหน่วยความจำชนิดหนึ่งที่อยู่ภายในครอปีซีชีพ ใช้สำหรับการเก็บข้อมูลชั่วคราวในการประมวลผลทางด้านการคำนวณหรือทางลอกิจในครอปีซีชีพสามารถติดต่อกับรีจิสเตอร์ได้โดยตรง ด้วยชุดคำสั่งที่ใช้เรียกชื่อของรีจิสเตอร์นั้น โดยไม่ต้องระบุตำแหน่งของหน่วยความจำ นอกจากนั้นแล้ว รีจิสเตอร์ยังสามารถใช้อ้างอิงในการระบุตำแหน่งของหน่วยความจำ หรือตำแหน่งของอุปกรณ์อินพุต/เอาท์พุต ใช้แสดงสถานะของข้อมูลต่างๆ

รีจิสเตอร์เลื่อนข้อมูล (Shift Register) เป็นอุปกรณ์ที่มีสำคัญมากในระบบดิจิตอลที่สร้างจากกลุ่มของ Flip-Flop มาต่อเรียงกัน จำนวนบิตของรีจิสเตอร์จะขึ้นอยู่กับจำนวนของ Flip-Flop ที่ถูกนำมาต่อกัน ใช้ในการเก็บข้อมูลหรือเป็นตัวกลางในการส่งผ่านข้อมูลและสามารถเลื่อนข้อมูลในรูปของเลขฐานสองได้ การนำข้อมูลเข้าหรือออกจากรีจิสเตอร์สามารถทำได้ทั้งแบบบานานหรือแบบอนุกรม โดยป้อนข้อมูลเข้าไปครั้งละหนึ่งบิต และเมื่อได้รับสัญญาณ Clock หรือเรียกว่า Shift pulse จะเกิดการเลื่อนข้อมูลที่ละบิต ตัวเลื่อนข้อมูลที่เป็นการนำ Flip-Flop ต่อกันใช้งาน จะมีลักษณะต่างกันดังนี้

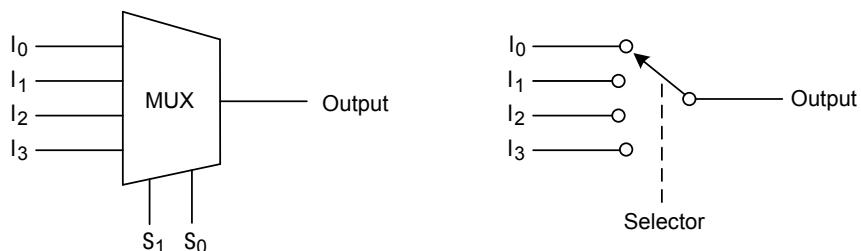
- Parallel in - Parallel out
- Parallel in - Serial out
- Serial in – Parallel out
- Serial in – Serial out

โครงสร้างภายในของรีจิสเตอร์ทำมาจาก Flip-Flop เช่น D Flip-Flop โดยที่ Flip-Flop 1 ตัวสามารถเก็บข้อมูลได้ 1 บิต ตัวอื่นๆ สามารถจราจรของรีจิสเตอร์ขึ้นมา 4 บิต แสดงดังรูป



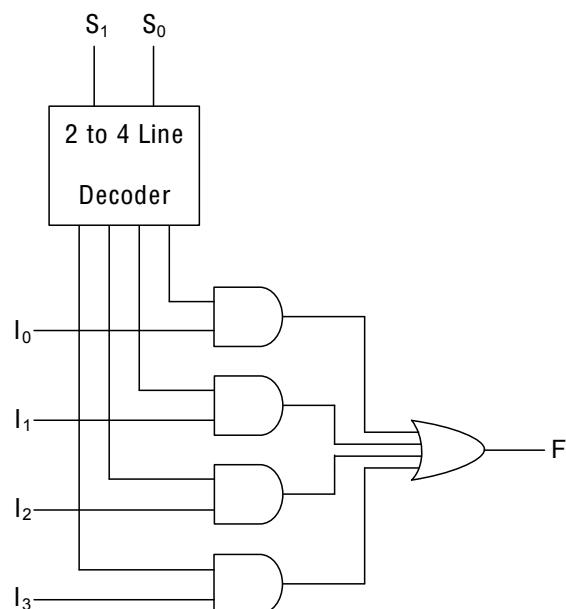
วงจรมาลติเพล็กเซอร์ (Multiplexer) เป็นวงจร Combinational Logic ที่ทำหน้าที่เลือกสัญญาณการเชื่อมต่อหรือส่งผ่านที่เข้ามาหลายอินพุตจากอุปกรณ์ต่างๆ ว่าจะให้สัญญาณอินพุตใดไปออกที่เอาท์พุต หรืออาจจะเรียกว่า Data Selector วงจร MUX ที่ทำหน้าที่เป็นสวิตช์นี้ จะต้องอาศัยแอดเดรส (Address) เป็นตัวกำกับใช้ในการสับสัญญาณสองสัญญาณหรือมากกว่านั้น แล้วส่งสัญญาณนั้นผ่านออกไปยังเอาท์พุตเพียงเส้นเดียว โดยที่จำนวนสัญญาณอินพุตทั้งหมดมีค่าเท่ากับ 2^n จะได้จำนวนแอดเดรสที่ใช้ในการควบคุมการเลือกเท่ากับ n เช่น ถ้ามีอินพุตที่จะเข้ามามากกว่า 8 เส้น จะต้องมีแอดเดรสที่ใช้ควบคุม (Control) การเลือกสัญญาณเท่ากับ 3 เส้น และให้มีสัญญาณที่ถูกเลือกออกไปที่เอาท์พุตจำนวน 1 เส้น

ตัวอย่าง วงจรมาลติเพล็กเซอร์ แบบ 4 อินพุต (4 Line to 1 Line Multiplexer) วงจร มีอินพุต 4 เส้น และมีเอาท์พุต 1 เส้น ดังนั้นจึงต้องมีขาควบคุมเท่ากับ 2 เส้น ($2^2 = 4$) เพื่อใช้เลือกข้อมูลใดข้อมูลหนึ่งออกไปทางเอาท์พุตจากอินพุต 4 เส้น โดยมีสัญลักษณ์ และ Truth Table ของวงจรดังรูป



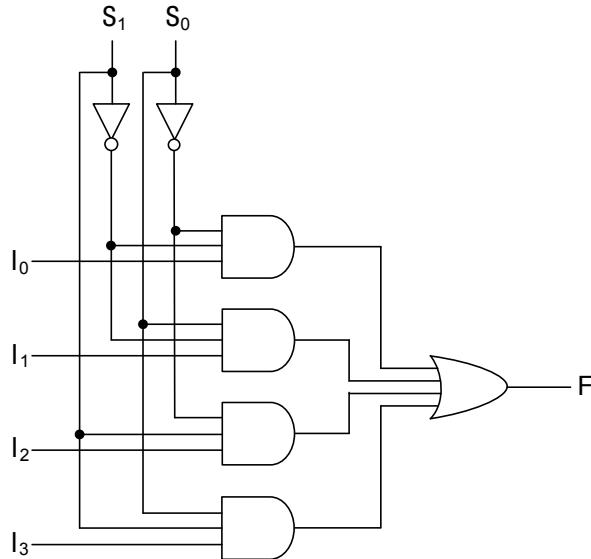
Selector		Input				Output
S ₁	S ₀	I ₃	I ₂	I ₁	I ₀	F
0	0	x	x	x	I ₀	I ₀
0	1	x	x	I ₁	x	I ₁
1	0	x	I ₂	x	x	I ₂
1	1	I ₃	x	x	x	I ₃

จากตารางความจริงสามารถเขียนเป็น Block Diagram ได้ดังรูป โดยที่ AND Gate ทั้ง 4 ตัวในวงจรอาจจะใช้เป็น Tri-State Buffer มาแทนและนำเอาเอาท์พุตทั้ง 4 เส้นเข้ามาร่วมกันโดยตรงได้



จาก Block Diagram ในส่วนของ Decoder ถ้านำมาเขียนรวมเป็นวงจรลอจิกเกตแล้ว จะได้สมการลําจิกและวงจรมาลติเพล็กซอร์แบบ 4 อินพุท ดังนี้

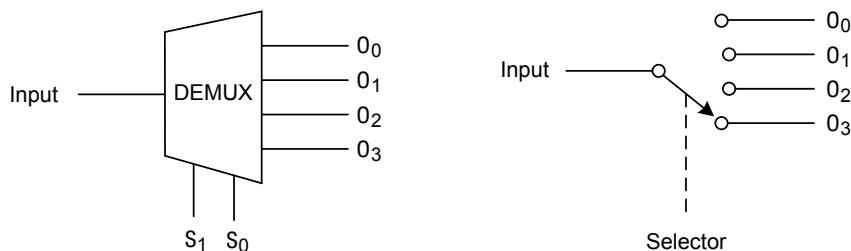
$$F = I_3 \cdot S_1 \cdot S_0 + I_2 \cdot S_1 \cdot \overline{S_0} + I_1 \cdot \overline{S_1} \cdot S_0 + I_0 \cdot \overline{S_1} \cdot \overline{S_0}$$



วงจรดีมัลติเพล็กซอร์ (Demultiplexer) เป็นวงจรที่ทำงานตรงกันข้ามกับ Multiplexer วงจรจะมีสัญญาณอินพุทที่เข้ามาเพียงเส้นเดียว และสามารถนำไปใช้งานเพื่อให้เลือกว่าจะให้สัญญาณอินพุทที่เข้ามานี้ออกไปที่เอาท์พุทเส้นใด หรืออาจจะเรียกว่า Data Distributor โดยมีตัวที่ทำหน้าที่เลือกสัญญาณออกเป็นค่า แอดเดรสหลายค่าตามจำนวนเอาท์พุท และมีจังหวะการส่งข้อมูลของเอาท์พุทแต่ละตัวแตกต่างกันตามเวลาที่กำหนด

วงจร Multiplexer และ Demultiplexer สามารถนำไปใช้ส่งข้อมูลหรือข่าวสารหลายๆอย่างพร้อมกัน โดยเลือกสัญญาณอินพุทจากหลายแหล่งของข้อมูล แล้วส่งออกไปสู่เอาท์พุทเพียงเส้นเดียว เพื่อที่จะส่งข้อมูลหรือข่าวสารนั้นไปยังที่ห่างไกลออกไป และเมื่อปลายทางได้รับสัญญาณนั้นแล้วจะต้องมีวงจร Demultiplexer เพื่อแยกสัญญาณข้อมูลที่มีอยู่หลายเอาท์พุท โดยจะเลือกให้ไปออกเอาท์พุทด้านค่าของช่องสัญญาณอินพุทที่ถูกต้อง ซึ่งโดยปกติแล้ววงจร Multiplexer จะต้อง Synchronous กับ Demultiplexer เสมอเพื่อให้ได้รหัสควบคุมที่เหมือนกัน และข้อมูลที่ออกมากจาก Demultiplexer จึงจะเป็นข้อมูลที่ถูกต้อง

ตัวอย่าง วงจรดีมัลติเพล็กซอร์ แบบ 4 เอาท์พุท (1 Line to 4 Line Demultiplexer) โดยวงจร มีอินพุต 1 เส้น และมีเอาท์พุท 4 เส้น ดังนั้นจึงต้องมีขาควบคุมท่ากับ 2 เส้น ($2^2 = 4$) เพื่อที่จะให้เลือกให้ข้อมูลออกตามเอาท์พุทที่กำหนด โดยมีสัญลักษณ์ และ Truth Table ของวงจรดังรูป



Selector		Input I	Output			
S_1	S_0		O_3	O_2	O_1	O_0
0	0	I	0	0	0	I
0	1	I	0	0	I	0
1	0	I	0	I	0	0
1	1	I	I	0	0	0

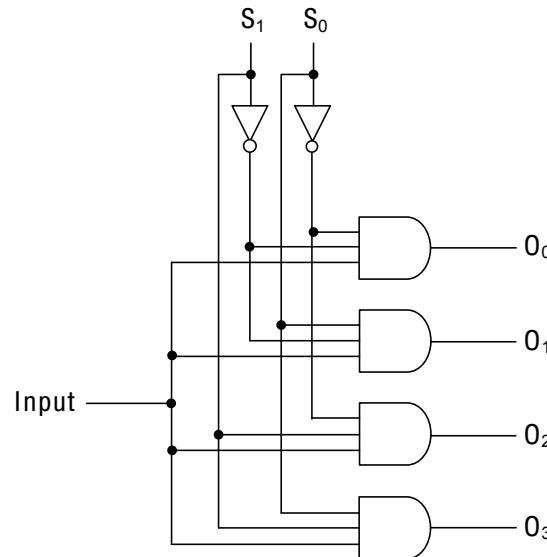
จากตารางความจริงสามารถเขียนเป็นสมการลอจิกและวงจรดิจิตอลก็จะมีรูปแบบ 4 เอกที่พุทธได้ดังนี้

$$O_0 = I \cdot \overline{S_1} \cdot \overline{S_0}$$

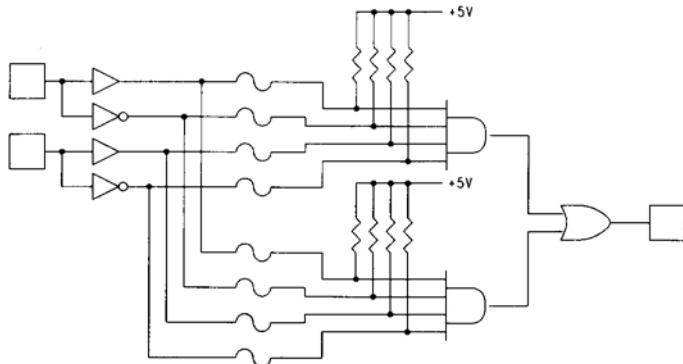
$$O_1 = I \cdot \overline{S_1} \cdot S_0$$

$$O_2 = I \cdot S_1 \cdot \overline{S_0}$$

$$O_3 = I \cdot S_1 \cdot S_0$$

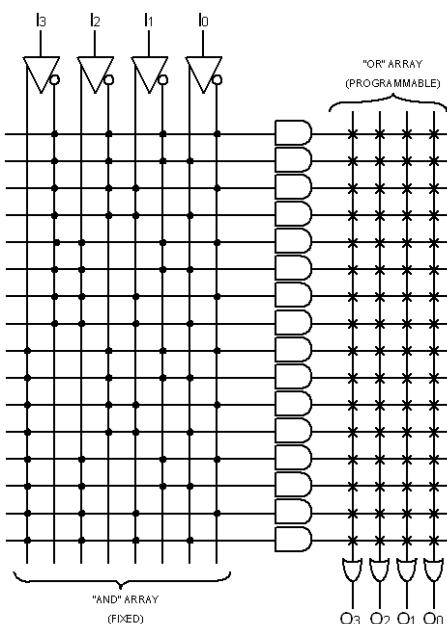


ไอซีประเภท Programmable Logic Devices (PLDs) มีโครงสร้างภายในเป็นวงจรลอจิกพื้นฐานต่อ กันอยู่เป็นกลุ่ม ซึ่งมีทั้งวงจรคอมบินेशัน(Combination) และซีเควนเชียล (Sequential) โดยวงจรภายในจะมีส่วนประกอบพื้นฐานอยู่ในรูปของ Sum of Product From เป็นวงจรคอมบินेशันที่เอา Thomoplus คูณที่เกิดจากการ AND ระหว่างตัวแปรแล้วนำมารวมกัน OR กัน สำหรับการโปรแกรมนั้นอินพุทของอุปกรณ์จะถูกต่อผ่านฟิวส์เข้ากับแหล่งสัญญาณดังรูป ซึ่งถ้าไม่ต้องการใช้สัญญาณใดก็จะโปรแกรมให้ตัดฟิวส์นั้นทิ้งไป จึงโปรแกรมได้เพียงครั้งเดียว และไอซี PLD บางชนิดจะใช้มอสทรานซิสเตอร์แทนฟิวส์ทำให้สามารถลบและโปรแกรมเข้าไปใหม่ได้หลายครั้ง โครงสร้างพื้นฐานของ PLD ในแต่ละแบบมีดังนี้



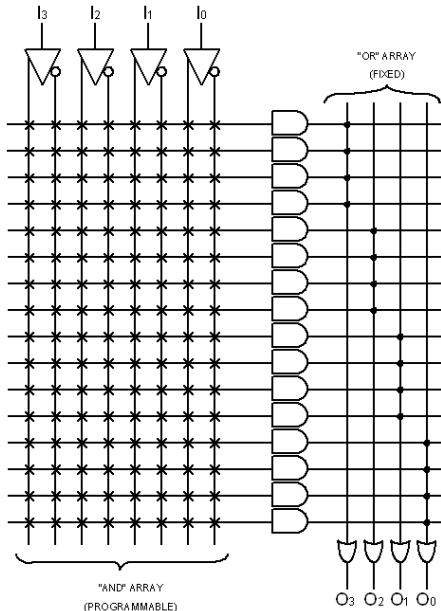
รูป แสดงตำแหน่งฟิวส์ของวงจรพื้นฐานใน PLD

PROM (Programmable Read Only Memory) โครงสร้างพื้นฐานทางลอจิกของ PROM ประกอบไปด้วย AND Array โดยเอาท์พุทจะถูกป้อนเข้าสู่ OR Array ที่สามารถทำการโปรแกรมได้ PROM จะมีราคาต่ำกว่า PLD และเหมาะสมสำหรับใช้เป็นโปรแกรมการจัดระบบ โดยล่วงมาคนนิยมใช้เรื่องโปรแกรมคอมพิวเตอร์ในการประยุกต์ใช้งานเพื่อเก็บข้อมูลระบบที่อินพุทเป็นตำแหน่งของหน่วยความจำ (Address) ในคอมพิวเตอร์ และเอาท์พุทคือค่าข้อมูล (Data) ที่บรรจุอยู่ในหน่วยความจำ ตามในรูปจะเป็น PROM ขนาด 16×4 บิต ที่มีอินพุตค่า Address ตั้งแต่ 0000 จนถึง 1111 โดยที่จำนวนอินพุท n เส้น จะมีค่า Address ที่ เป็นไปได้ทั้งหมดเท่ากับ 2^n และมีค่าจำนวนบิตที่เก็บในแต่ละ Address เท่ากับจำนวน OR Gate หรือ เอาท์พุทของวงจร



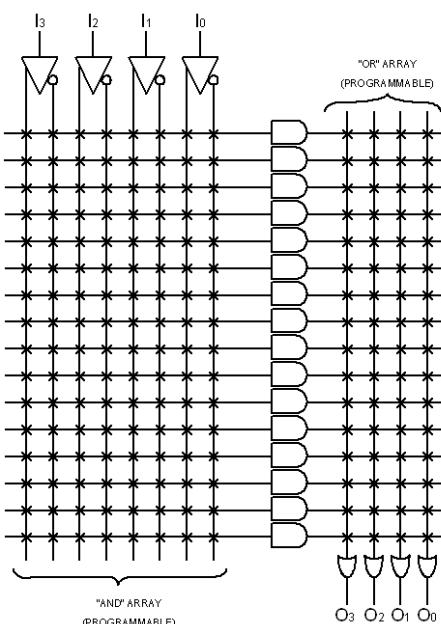
รูป แสดงโครงสร้างพื้นฐานทางลอจิกของ PROM

PAL (Programmable Array Logic) โครงสร้างพื้นฐานทางลогоจิกของ PAL ประกอบด้วย AND Array ที่สามารถทำการโปรแกรมได้ ซึ่งเอาท์พุทจะถูกป้อนเข้าสู่ OR Array ที่ต่อໄว้ต้ายตัว ดังรูป PAL เป็นอุปกรณ์ที่ถูกพัฒนามาเพื่อช่วยให้สามารถทำฟังก์ชันลogoจิกที่ยุ่งยากซับซ้อน ซึ่งประกอบด้วยหลายอินพุต และหลายเออท์พุตให้สามารถใช้ไอซีเพียงตัวเดียว การโปรแกรมจะจารอจิกลงในไอซี PAL จะต้องเลือกเบอร์ไอซีที่มีจำนวนอินพุตและเออท์พุตรวมทั้งขนาดและจำนวนเกตมากพอที่จะบรรจุวงจรลงได้



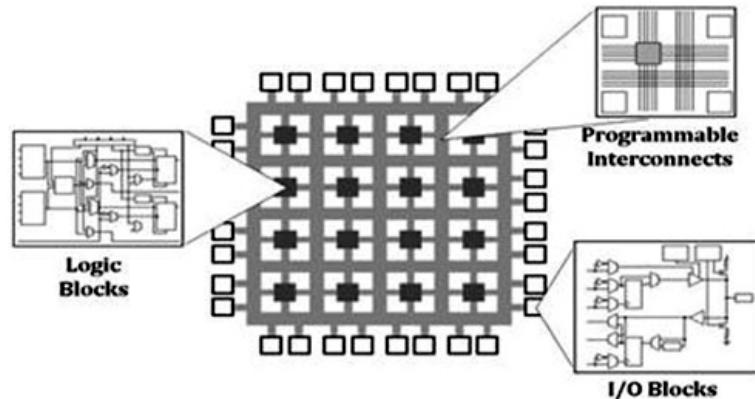
รูปแสดงโครงสร้างพื้นฐานทางลогоจิกของ PAL

FPLA (Field Programmable Logic Array) โครงสร้างพื้นฐานทางลогоจิกของ FPLA หรือ PLA จะประกอบด้วย AND Array ที่สามารถทำการโปรแกรมได้ โดยเอาท์พุทจะป้อนเข้าสู่ OR Array ที่สามารถทำการโปรแกรมได้ดังรูป ซึ่งผู้ออกแบบสามารถควบคุมทั้งอินพุตและเออท์พุตได้อย่างสมบูรณ์ PLA ถูกออกแบบให้ประยุกต์ใช้งานได้อย่างกว้างขวาง



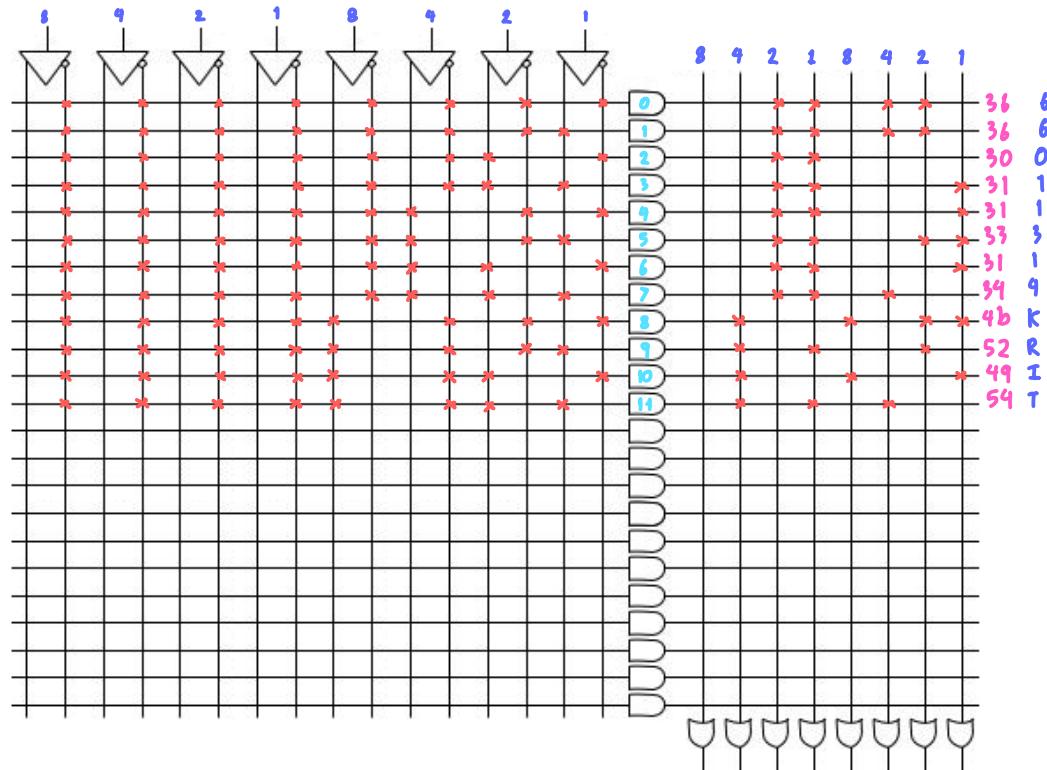
รูปแสดงโครงสร้างพื้นฐานทางลогоจิกของ PLA

FPGA (Field-Programmable Gate Array) เป็นอุปกรณ์ชนิดโปรแกรม ได้ที่มีความซับซ้อนมากไปอีก ระดับหนึ่ง ความจุเท่ากับในตัวชิพ FPGA ได้เพิ่มขึ้นจนถึงระดับล้านตัว โดยมีโครงสร้างการเชื่อมต่อภายในแบบ เมตริกซ์ดังรูป ประกอบด้วยตัวบล็อกของอุปกรณ์ต่างๆแบบโปรแกรมได้ และมีลักษณะของหน่วยความจำ หรือฟลิปฟล๊อป โครงสร้างภายในของ FPGA นั้นสามารถโปรแกรมให้มีหน้าที่การทำงานรวมกันหลายๆ ชนิด เช่น ทำงานจrocดอรหัส หรือพิ้งค์ชั้นทางคณิตศาสตร์ ซึ่งสามารถรองรับวงจรคิจิตอลที่มีความ слับซับซ้อนได้เป็นอย่างดี ในการออกแบบวงจรคิจิตอลที่มี FPGA อยู่บนแผงวงจรจะช่วยให้ผู้ออกแบบสามารถลดขนาดของ แผงวงจร รวมทั้งสามารถออกแบบพัฒนาและทดสอบได้ง่าย แล้วยังสามารถเข้าไปแก้ไขวงจรหลังจากใช้งานใน ภายหลังได้ ทำให้ในปัจจุบันเป็นที่นิยมและมีแนวโน้มที่จะนำมาใช้งานมากขึ้นเรื่อยๆ



รูปแสดงโครงสร้างพื้นฐานทางลогоจิกของ FPGA

1. จากรูปถ้ากำหนดเป็น PROM ให้ใส่รหัสประจำตัวและชื่อนักศึกษาลงไป โดยกำหนดให้ Address เริ่มต้นเป็น 0



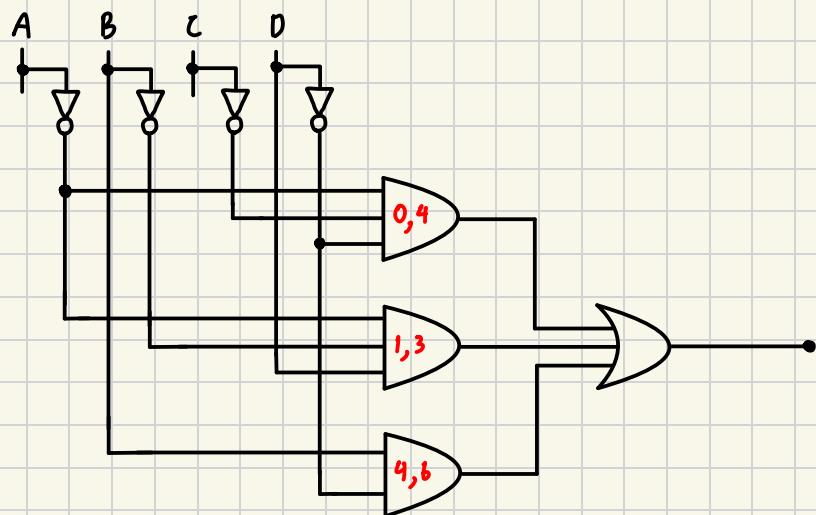
AB	CD	00	01	11	10
00	0000	0001	0011	0010	0110
01	0100	0101	0111	0110	0110
11	d	d	d	d	d
10	1000	1001	1011	1010	1010

$$\bar{A}\bar{C}\bar{D} + \bar{A}\bar{B}D + B\bar{D}$$

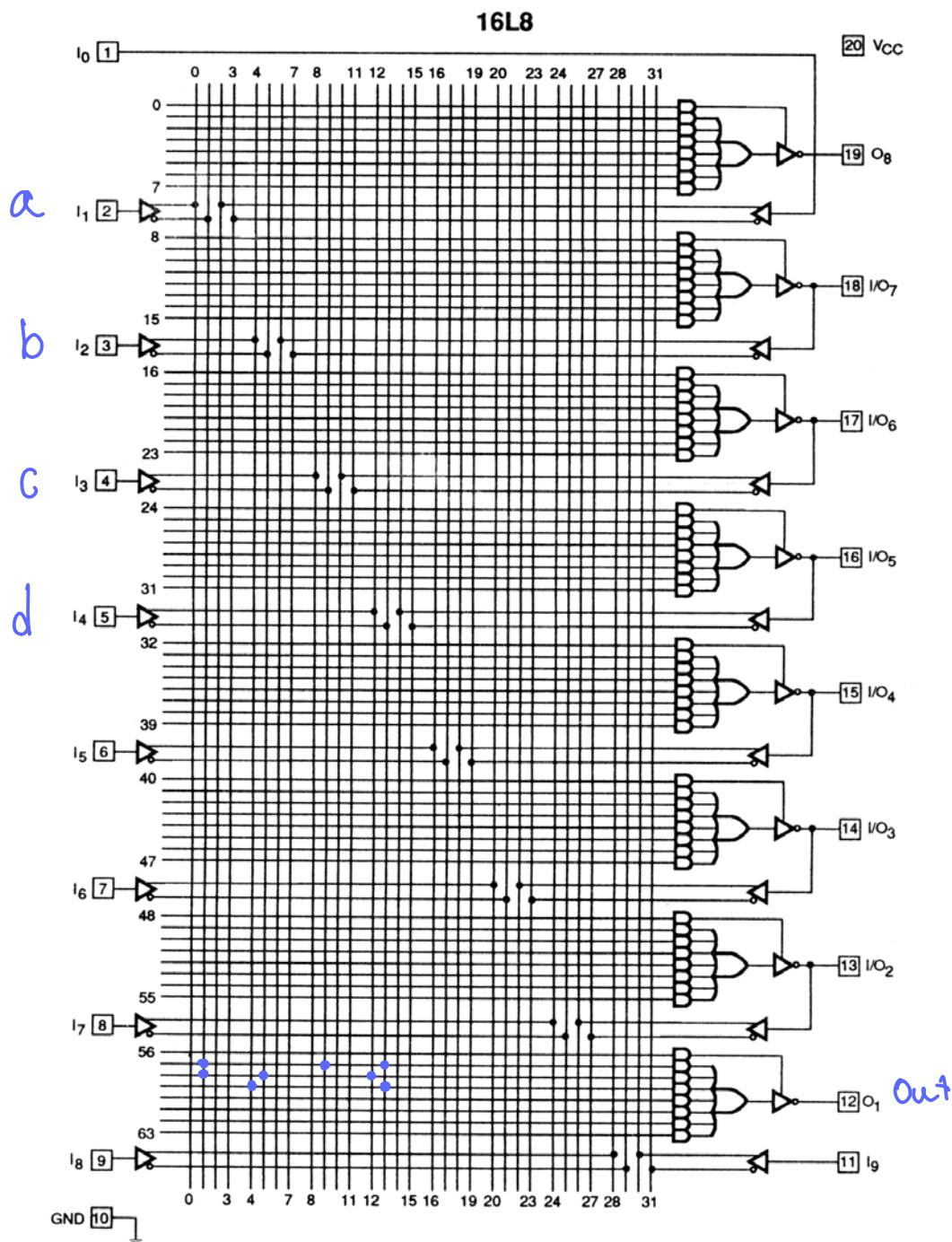
66011314

9678577876 657826776975

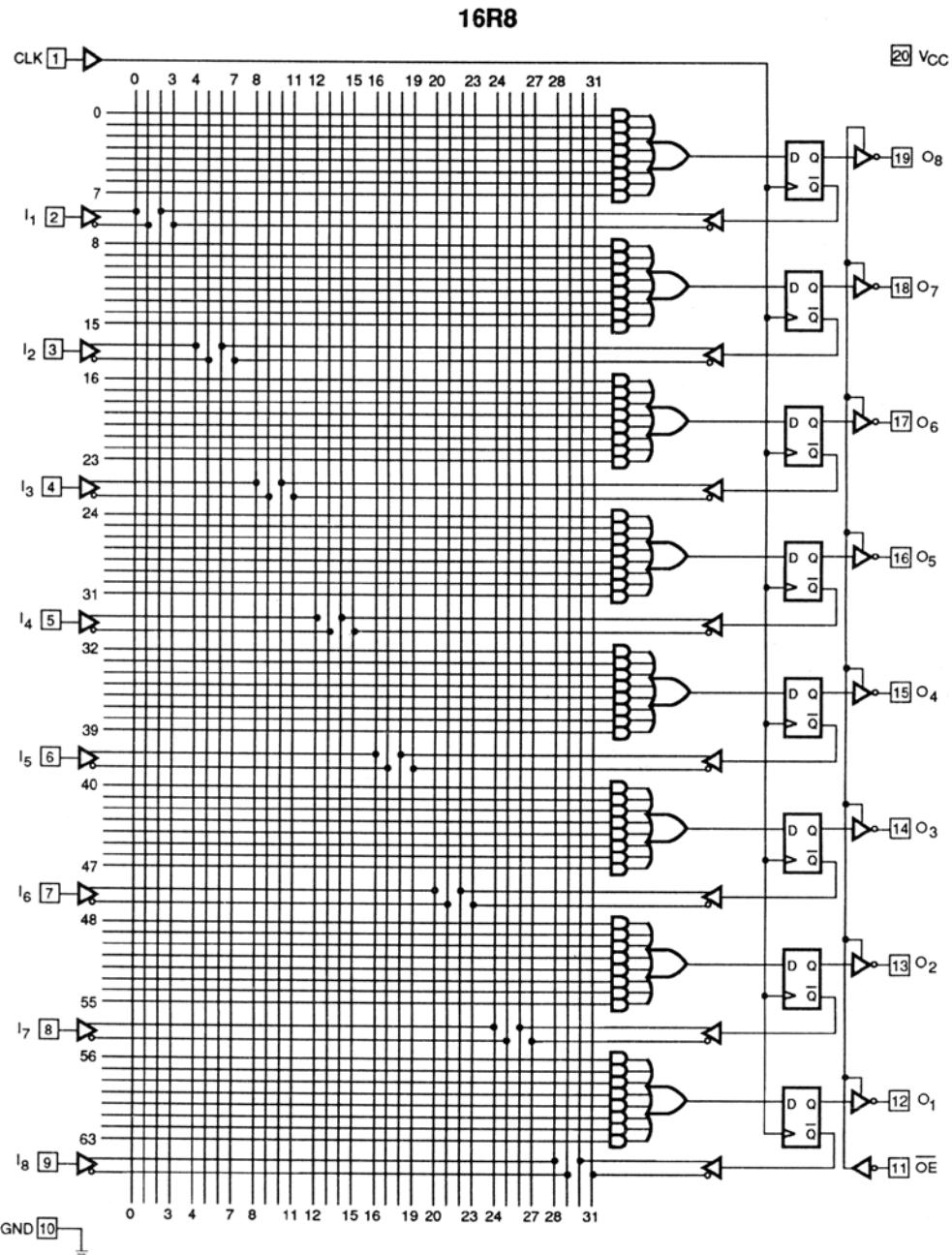
66011314



ตัวอย่างเช่น ไอซี PAL เบอร์ 16L8 ดังแสดงในรูป ชิ้นภายในจะประกอบไปด้วยอินพุททั้งหมดที่ต่อเข้ามา ไอซีจำนวน 10 เส้น และมีเอาท์พุททั้งหมด 8 เส้น จากเอาท์พุททั้ง 8 จะมีเอาท์พุทอีก 6 เส้นที่สามารถป้อนกลับมายังอินพุของวงจรได้อีกด้วย จึงสามารถโปรแกรมให้ขาทั้ง 6 เส้น อาจจะทำเป็นอินพุทอย่างเดียว หรือเอาท์พุทอย่างเดียว หรือเอาท์พุทที่มีการป้อนกลับไปยังอินพุของวงจร โดยสามารถเลือกกำหนดในขาใดก็ได้ที่ขึ้นอยู่กับสมการลอกิกที่ป้อนเพื่อไปโปรแกรมในตัวไฟว์ จากเบอร์ ไอซีด้านี้คือ 16L8 จะมีความหมายว่ามีจำนวนอินพุทสูงสุดของวงจร 16 เส้น เอาท์พุทของวงจรเป็น Active Low และมีจำนวนเอาท์พุทของวงจรสูงสุด 8 เส้น ที่เอาท์พุทของวงจรทั้ง 8 เส้นจะมี Inverter ที่เป็นแบบ 3 สถานะอยู่ ดังนั้นเอาท์พุทของวงจรจึงเป็นได้ 3 อย่างคือ High , Low และ High Impedance ในกรณีเป็น High Impedance ขา 13 ถึงขา 18 จะสามารถป้อนสัญญาณเข้าจากภายนอกได้ ไอซีเบอร์นี้สามารถกำหนดทำงานของลอกิกได้สูงสุด 8 สมการ ตามค่าเอาท์พุทที่มี 8 เส้น ในแต่ละสมการจะมีจำนวนเทอมผลลัพธ์สูงสุด 7 เทอม และในแต่ละเทอมจะมีตัวแปรภายในสูงสุด 16 ตัวแปร ตามอินพุทของ AND Gate ที่มีจำนวน 32 เส้น



ตัวอย่าง PAL เบอร์ 16R8 จะมี Register หรือ D Flip Flop อยู่ 8 ตัว ไว้เก็บค่าเอาท์พุททั้ง 8 เส้น ดังรูป



ไอซี PAL เบอร์ 16R8 จะใช้ทำงานจริงเมื่อตัว Memory ไว้เก็บสถานะของวงจร ได้ ซึ่งในไอซี PAL จะใช้เป็น D Flip Flop ภายในตัว ไอซีจะประกอบด้วยขาป้อนสัญญาณอินพุทจากภายนอกจำนวน 8 เส้น ขาป้อนสัญญาณ Clock 1 เส้น ขา Output Enable 1 เส้น และขาสัญญาณเอาท์พุททั้งหมด 8 เส้น โดยสัญญาณเอาท์พุททั้ง 8 เส้นจะสามารถป้อนกลับมาเป็นอินพุทธองวงจรได้อีก โดยผ่านทางขา \bar{Q} ของ Flip Flop ไอซีตัวนี้จะมี Register หรือ D Flip Flop อยู่ 8 ตัว ไว้เก็บค่าเอาท์พุททั้ง 8 เส้น โดยเมื่อเอาท์พุทธองวงจรเป็น Active Low ดังนั้น จึงมีสมการทั้งหมด 8 สมการ ในแต่ละสมการจะมีจำนวนเทอมผลคูณสูงสุด 8 เทอม ตามจำนวน AND Gate ที่มีอยู่ 8 ตัว และมีอินพุทของ AND Gate ทั้งหมดในแต่ละตัวจำนวน 32 เส้น จึงทำให้เทอมผลคูณหนึ่งเทอมจะมีตัวแปรได้สูงสุด 16 ตัวแปร

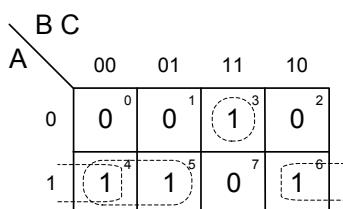
วงจร Sequential จะรับสัญญาณอินพุตจากภายนอกเข้ามา ค่าข้อมูลอินพุตที่เข้ามานี้จะถูกเก็บไว้ในหน่วยความจำที่เวลาใดๆ เรียกว่าสภาวะ (State) เราเรียกสภาวะของหน่วยความจำก่อนป้อนสัญญาณอินพุตนั้นว่าเป็นสภาวะปัจจุบัน (Present State) และเรียกสภาวะของหน่วยความจำหลังจากป้อนสัญญาณอินพุตเป็นสภาวะถัดไป (Next State) วงจร Sequential สามารถแบ่งได้เป็น 2 ประเภทใหญ่ๆ ตามลักษณะหน่วยความจำที่ใช้คือ Synchronous Sequential เป็นวงจรที่ใช้สัญญาณ Clock มาควบคุมการเปลี่ยนสภาวะ โดยหน่วยความจำต้องให้สัญญาณ Clock และอีกแบบคือ Asynchronous Sequential เป็นวงจรที่หน่วยความจำจะไม่ใช้สัญญาณ Clock

ตัวอย่าง วงจรนับ 8 ที่เป็นแบบ Synchronous Counter สามารถเขียนเป็น Truth Table ได้ดังตาราง

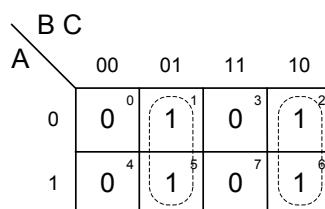
Truth Table ของวงจรนับ 8

n	Present State			Next State		
	B ₂	B ₁	B ₀	B ₂	B ₁	B ₀
0	0	0	0	0	0	1
1	0	0	1	0	1	0
2	0	1	0	0	1	1
3	0	1	1	1	0	0
4	1	0	0	1	0	1
5	1	0	1	1	1	0
6	1	1	0	1	1	1
7	1	1	1	0	0	0

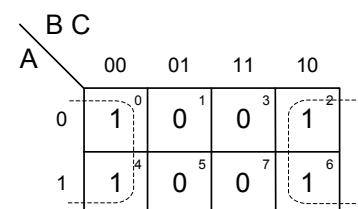
จากตาราง Truth Table ของวงจรนับ 8 จะต้องนำค่า Next State ไปหาผลจิกฟังก์ชัน โดยใช้ Karnaugh map จะได้ดังนี้



$$F1 = \bar{A} \cdot B \cdot C + A \cdot \bar{B} + A \cdot \bar{C}$$



$$F2 = \bar{B} \cdot C + B \cdot \bar{C}$$



$$F3 = \bar{C}$$

Truth Table ของ JK Flip Flop

Q_n	Q_{n+1}	J	K
0	0	0	d
0	1	1	d
1	0	d	1
1	1	d	0

จากตาราง Truth Table ของวงจรนับ 8 ที่จะใช้ JK Flip Flop เป็นหน่วยความจำ จะต้องนำค่า Next State ไปเทียบกับ Truth Table ของ JK Flip Flop จะได้เป็นตารางดังนี้

n	Output			Input		
	C	B	A	$J_c K_c$	$J_b K_b$	$J_a K_a$
0	0	0	0	0 d	0 d	1 d
1	0	0	1	0 d	1 d	d 1
2	0	1	0	0 d	d 0	1 d
3	0	1	1	1 d	d 1	d 1
4	1	0	0	d 0	0 d	1 d
5	1	0	1	d 0	1 d	d 1
6	1	1	0	d 0	d 0	1 d
7	1	1	1	d 1	d 1	d 1

จากตาราง เมื่อนำไปหาลอจิกฟังก์ชัน โดยใช้ Karnaugh map จะได้ดังนี้

C	BA			
	00	01	11	10
0	0	0	1	0
1	d	d	d	d

$$J_c = BA$$

C	BA			
	00	01	11	10
0	d	d	d	d
1	0	0	1	0

$$K_c = BA$$

C	BA			
	00	01	11	10
0	0	1	d	d
1	0	1	d	d

$$J_b = A$$

C	BA			
	00	01	11	10
0	d	d	1	0
1	d	d	1	0

$$K_b = A$$

C	BA			
	00	01	11	10
0	1	d	d	1
1	1	d	d	1

$$J_a = 1$$

C	BA			
	00	01	11	10
0	d	1	1	d
1	d	1	1	d

$$K_a = 1$$

จากการนับ 8 สามารถนำเอาค่า Next State ของตาราง Truth Table ไปหาผลจิกฟังก์ชันโดยใช้วิธี Quine-McClusky Method ที่ไม่ได้เป็น subset ของ PI กลุ่มอื่น เริ่มจากการเขียน minterm ให้อยู่ในรูปของ เลขฐานสอง เรียงลำดับเทอมที่อยู่ในรูปของเลขฐานสองโดยแบ่งเป็นกลุ่มตามจำนวนบิตของผลจิกหนึ่ง จากนั้นจะเป็นการจับคู่เบริญเพียบระหว่างเทอมที่มีจำนวนบิตต่างกัน 1 บิต ซึ่งจะเป็นการลดตัวแปรลงได้ครึ่งละ 1 บิต (แทนตัวแปรที่ลดไปด้วย -) และนำผลจากการลดตัวแปรแล้วมาจับคู่เบริญเพียบใหม่ จนไม่สามารถจับคู่ได้อีก จำนวนมากได้

ตัวอย่าง

$$\begin{aligned} F1(A, B, C) &= \Sigma m(3, 4, 5, 6) \\ F2(A, B, C) &= \Sigma m(1, 2, 5, 6) \\ F3(A, B, C) &= \Sigma m(0, 2, 4, 6) \end{aligned}$$

โปรแกรม Quine-McClusky ได้เป็นดังนี้

LIST 1			
/ [000]	F3	0	
/ [001]	F2	1	
/ [010]	F2 F3	2	
/ [100]	F1 F3	4	
P1 [011]	F1	3	
P2 [101]	F1 F2	5	
P3 [110]	F1 F2 F3	6	

LIST 2			
/ [-00]	F3	0, 4	
/ [0-0]	F3	0, 2	
P4 [10-]	F1	4, 5	
P5 [1-0]	F1 F3	4, 6	
P6 [-01]	F2	1, 5	
P7 [-10]	F2 F3	2, 6	

LIST 3			
P8 [--0]	F3	0, 2, 4, 6	

แล้วนำเทอมที่ไม่สามารถจับคู่ได้ ซึ่งที่คือการตัดเทอมที่ซ้ำกันออกไป จะได้เป็น PI ทั้งหมด ดังนี้

PI(1)	[011]	F1	3
PI(2)	[101]	F1 F2	5
PI(3)	[110]	F1 F2 F3	6
PI(4)	[10-]	F1	4, 5
PI(5)	[1-0]	F1 F3	4, 6
PI(6)	[-01]	F2	1, 5
PI(7)	[-10]	F2 F3	2, 6
PI(8)	[--0]	F3	0, 2, 4, 6

หลังจากนั้น จะเป็นการหา Essential Prime Implicants (EPI) คือ PI ที่มีสมาชิกอย่างน้อยหนึ่งตัวที่ไม่ได้อยู่ใน PI อื่นๆ โดยการพิจารณาว่ามี PI ที่มีเทอมที่ไม่ซ้ำกับเทอมใน PI อื่น ซึ่งจะเป็น PI หลักที่เราต้องเลือกก่อน ส่วนที่เหลือจะเป็น PI ที่มีเทอมที่ซ้ำกับเทอมใน PI อื่นๆ ซึ่งเราจะต้องพิจารณาเลือก PI ให้มีจำนวนน้อยที่สุด ที่สามารถครอบคลุมเทอมทั้งหมด ได้ดังนี้

[F1]

Minterm	3	4	5	6
* PI(1)	x			
* PI(2)			x	
* PI(3)				x
* PI(4)		x	x	
* PI(5)		x		x
* PI(6)				
* PI(7)				
* PI(8)				

[F2]

Minterm	1	2	5	6
* PI(1)				
* PI(2)			x	
* PI(3)				x
* PI(4)				
* PI(5)				
* PI(6)	x			x
* PI(7)		x		x
* PI(8)				

[F3]

Minterm	0	2	4	6
* PI(1)				
* PI(2)				
* PI(3)			x	
* PI(4)				
* PI(5)			x	x
* PI(6)				
* PI(7)		x		x
* PI(8)	x	x	x	x

จากค่า PI ที่เลือกทั้งหมด จะได้ผลลัพธ์เป็นสมการดังนี้

$$\begin{aligned} F1(A, B, C) &= \Sigma m(3, 4, 5, 6) \\ F2(A, B, C) &= \Sigma m(1, 2, 5, 6) \\ F3(A, B, C) &= \Sigma m(0, 2, 4, 6) \end{aligned}$$

$$\begin{aligned} F1 &= PI(1) + PI(4) + PI(5) \\ F2 &= PI(6) + PI(7) \\ F3 &= PI(8) \end{aligned}$$

$$\begin{aligned} F1 &= A'BC + AB' + AC' \\ F2 &= B'C + BC' \\ F3 &= C' \end{aligned}$$

$$\begin{aligned} F1 &= !A \& B \& C \# A \& !B \# A \& !C \\ F2 &= !B \& C \# B \& !C \\ F3 &= !C \end{aligned}$$

การออกแบบวงจรด้วย PLD

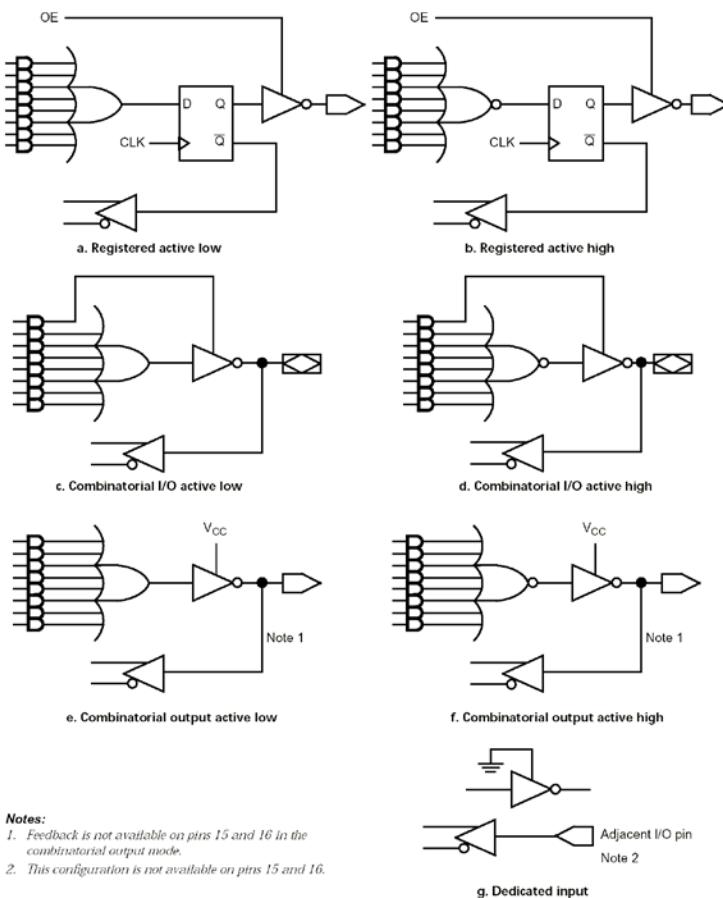
Designing with Programmable Logic Device

วัตถุประสงค์

1. ศึกษาวิธีการโปรแกรมวงจรลงในตัวไอซี PLD
2. เพื่อให้สามารถใช้ PLD แทนวงจรลอจิก
3. ศึกษาการออกแบบวงจร Sequential
4. เพื่อให้สามารถนำ PLD ไปประยุกต์ใช้ในการออกแบบวงจรต่างๆ ได้

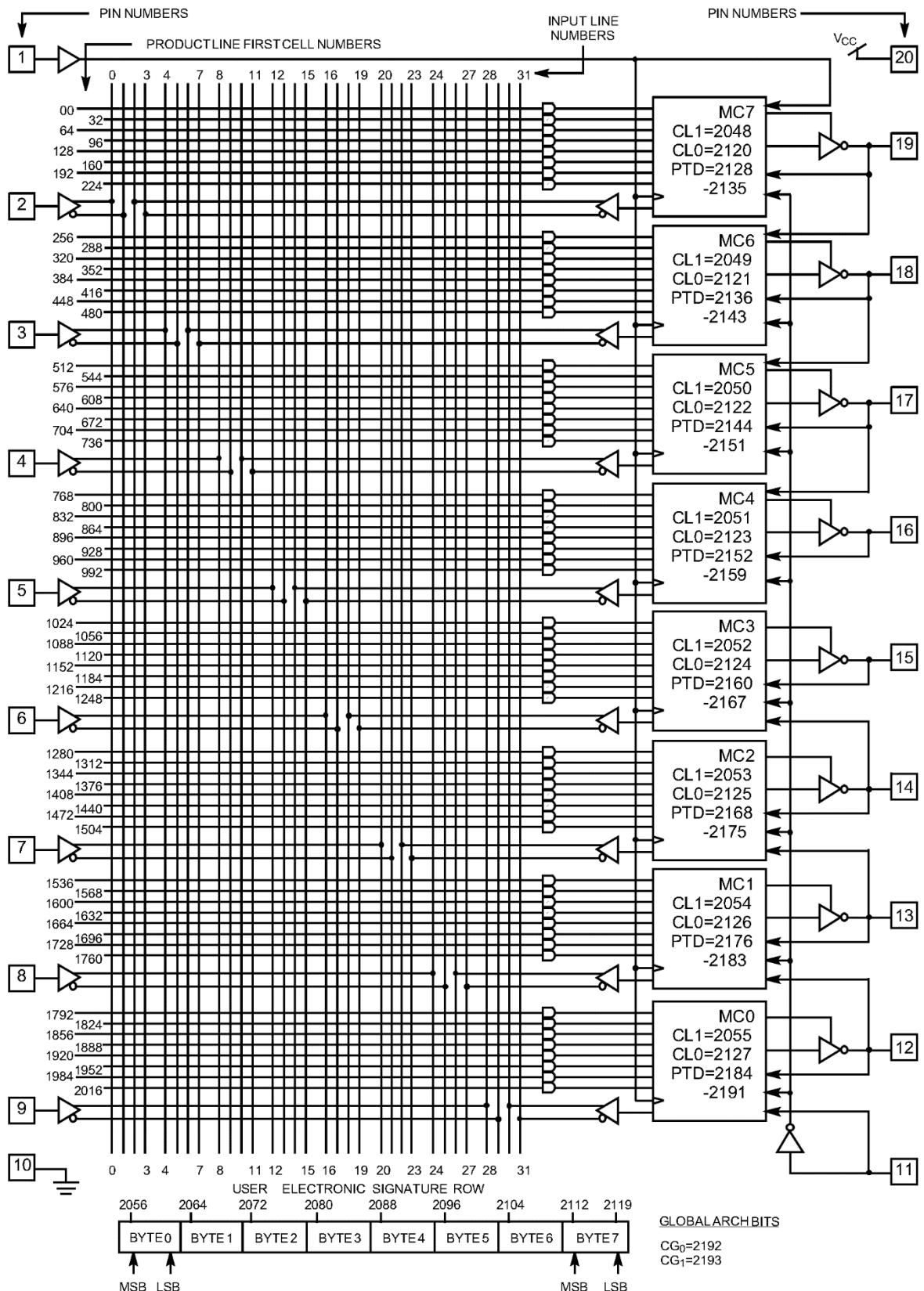
ทฤษฎี

การทดลอง จะใช้ไอซี GAL (Generic Array Logic) ที่ใช้เทคโนโลยี Electrically Erasable CMOS เป็นรุ่น 16V8 ภายในตัวไอซีจะประกอบด้วย Output Logic Macrocell ที่สามารถกำหนดค่าเอาท์พุททั้ง 8 เส้น คือขา 12 ถึงขา 19 ให้มีเอาท์พุทของวงจรเป็น Registered active low , Registered active high , Combinatorial output active low , Combinatorial output active high และ Combinatorial I/O active low , Combinatorial I/O active high ที่มี Inverter แบบ 3 สถานะอยู่ โดยสามารถป้อนสัญญาณจากภายนอกเข้าทางเอาท์พุทมาเป็นอินพุทของวงจรได้ นอกจากนี้ยังสามารถกำหนดให้เป็น Dedicated input ที่ป้อนสัญญาณเข้าจากภายนอก โดยที่เอาท์พุททั้งหมดที่กำหนดได้แสดงในรูป



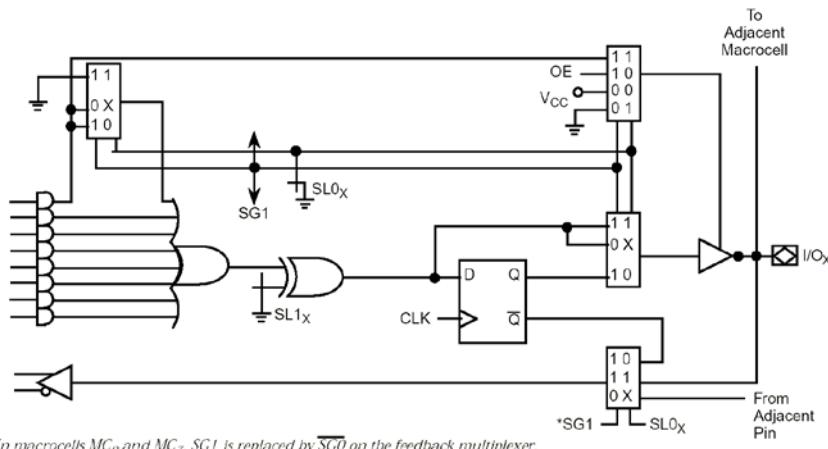
รูป แสดงแบบต่างๆ ที่ได้จาก Macrocell

วงจรทั้งหมดของไอซี 16V8 แสดงในรูป



รูป แสดงวงจร GAL 16V8

วงจรภายใน Macrocell ของ GAL 16V8 แสดงดังรูป



รูป แสดง GAL 16V8 Macrocell

การโปรแกรมวงจรลอดิจิตในไอซี จะต้องเลือกเบอร์ไอซีที่มีจำนวนอินพุตและเอาท์พุตรวมทั้งขนาด และจำนวนเกตมากพอที่จะบรรจุวงจรลงได้ หลังจากนั้นให้กำหนดค่าจะโปรแกรมให้เป็น active low หรือ active high มีการใช้หน่วยความจำที่เป็น Register หรือไม่ แล้วให้หาสมการลอดิจิกของวงจรที่จะนำไปใช้งาน และป้อนลงในโปรแกรม ซึ่งเรียกว่า Source Program และนำโปรแกรมที่ได้มามาทำการ Compile เพื่อให้ได้ JEDEC File ซึ่งจะนำไปใช้ในการโปรแกรมฟิวส์ลงในตัวไอซีโดยใช้เครื่องโปรแกรม

โดยมี Fuse ที่เริ่มต้นตั้งแต่ตำแหน่งแรกคือ L0000 ถึง L2047 จะเป็นเหมือนของไอซี PAL ถ้าตำแหน่งนี้มีค่าเป็น 0 หมายถึง Fuse ต้องชาร์จ และถ้าตำแหน่งนี้มีค่าเป็น 1 หมายถึง Fuse ละลายตัวเอง สรุปที่เหลือจะเป็นส่วนของ Output Logic Macrocell ที่มีขนาด 8 เอาท์พุต เป็นของไอซี GAL มีค่าตามบิทต่างๆ ดังนี้

L2048 ถึง L2055 เรียกว่า XOR หรือ SL1x มีความหมายคือถ้าเป็น 1 หมายถึง Output Active High และ 0 หมายถึง Output Active Low โดยเรียงตามตำแหน่งขา Output จากขา 12 ถึง 19

L2056 ถึง L2119 เป็น User's Electronic Signature กำหนดได้ทั้งหมด 64 ตัว

L2120 ถึง L2127 เรียกว่า AC1 หรือ SL0x เป็นตำแหน่งขา Output เรียงจากขา 12 ถึง 19 แสดงความหมายดังในตารางที่ 1

L2128 ถึง L2191 PRODUCT TERM disable fuses เรียงตาม AND Gate ทั้งหมดที่มีอยู่ 64 ตัว ถ้ากำหนดให้เป็น 1 คือ Enable

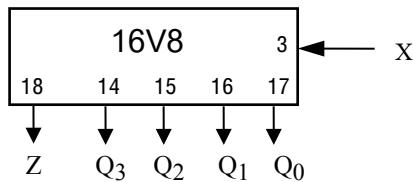
L2192 และ L2193 เรียกว่า SYN หรือ SG0 และ AC0 หรือ SG1 มีความหมายแสดงดังตารางที่ 1

ตารางที่ 1 Macrocell Configuration

SYN , SG0 L2192	AC0 , SG1 L2193	AC1 , SL0x L2120-2127	Cell Configuration
0	1	0 1	Registered Output Combinatorial I/O
1	0	0 1	Combinatorial Output Dedicated Input
1	1	1	Combinatorial I/O

การทดลองจะเป็นการออกแบบวงจร Sequence Detector ที่ทำหน้าที่เป็นวงจรนับ โดยจะให้ Output Sequence เป็น Logic 1 เมื่อมันตรวจจับได้ว่า Input Sequence ที่ป้อนเข้ามาในวงจรเป็น Logic 1 ที่มีจำนวนเท่ากับ 9 นอกจากข้อกำหนดดังกล่าวแล้วจะจะให้ Output เป็น Logic 0 ตลอด

การออกแบบวงจร Sequential โดยใช้ไอซี 16V8 จากข้อกำหนดของวงจร Synchronous Sequential ดังกล่าวที่จะต้องส่งข้อมูลเข้ามาทาง Input X เป็นแบบอนุกรม และจะต้องใช้หน่วยความจำไว้สำหรับเก็บข้อมูลทั้งหมดเท่ากับ 10 ค่า ดังนั้นจะต้องใช้หน่วยความจำทั้งหมดเท่ากับ 4 บิต หาได้จากจำนวนข้อมูลที่เก็บทั้งหมดเท่ากับ 2^4 เมื่อ k เป็นจำนวนตำแหน่งบิตของหน่วยความจำทั้งหมดที่ใช้ ตัวไอซีที่ได้จากการออกแบบแสดงดังรูป



รูป แสดงตำแหน่งขาไอซีที่จะออกแบบ

การออกแบบวงจรกำหนดให้ข้อมูลที่ป้อนเข้ามาทางอินพุตของไอซีเป็นขา X การป้อนข้อมูลต้องป้อนเข้ามาที่ละ 1 บิต และจะจะต้องจำสถานะก่อนหน้านี้ ดังนั้นจึงต้องมีส่วนของ Memory เพิ่มเข้ามา ในที่นี้จะใช้ D Flip-Flop ที่อยู่ในไอซี โดยที่หน่วยความจำที่เก็บ จะกำหนดให้เป็น State เป็นเลขตั้งแต่ 0 ถึง 9 ซึ่งมีทั้งหมด 4 เส้น กำหนดให้เป็น Q_3, Q_2, Q_1 และ Q_0 ขาเอาท์พุต มี 1 เส้น กำหนดให้เป็น Z เมื่อแทนค่าต่างๆ จากข้อกำหนดของวงจร Synchronous Sequential ดังกล่าวสามารถเปลี่ยนเป็น State Table ได้ดังตารางที่ 2

ตารางที่ 2 State Table

Present State	Next State		Output Z
	X = 0	X = 1	
S ₀	S ₀	S ₁	0
S ₁	S ₁	S ₂	0
S ₂	S ₂	S ₃	0
S ₃	S ₃	S ₄	0
S ₄	S ₄	S ₅	0
S ₅	S ₅	S ₆	0
S ₆	S ₆	S ₇	0
S ₇	S ₇	S ₈	0
S ₈	S ₈	S ₉	0
S ₉	S ₉	S ₀	1

- ให้นำการออกแบบวงจร Sequenced ตามค่าที่กำหนดใน State Table ไปเขียนเป็น PLD file ใส่ลงในโปรแกรมที่ชื่อ Counter.PLD ดังนี้

```

Name      Counter;
PartNo   IOT02;
Date     30/7/22;
Revision 01;
Designer Engineer;
Company  KMITL;
Assembly None;
Location None;
Device   g16v8;

```

```

/** Input Pins */
PIN 1 = CLK;           /* register clock */
PIN 2 = CLR;           /* clear input */
PIN 3 = X;             /* input X */
PIN 11 = !OE;          /* register output enable */

/** Output Pins */
PIN [14..17] = [Q3..0]; /* register 4-bit */
PIN 18 = Z;             /* output Z */

/** Declarations and Intermediate Variable Definitions */
field mode = [CLR,X];    /* declare mode control field */
LO      = mode:0;         /* define input low */
HI      = mode:1;         /* define input high */
CLEAR   = mode:[2..3];    /* define clear mode */

/** Logic Equations */
field STATE = [Q3..0];    /* declare state bit field */
#define S0 'b'0000
#define S1 'b'0001
#define S2 'b'0010
#define S3 'b'0011
#define S4 'b'0100
#define S5 'b'0101
#define S6 'b'0110
#define S7 'b'0111
#define S8 'b'1000
#define S9 'b'1001

Sequenced STATE {
present S0 if LO next S0;
            if HI next S1;
            if CLEAR next S0;

present S1 if LO next S1;
            if HI next S2;
            if CLEAR next S0;

present S2 if LO next S2;
            if HI next S3;
            if CLEAR next S0;

present S3 if LO next S3;
            if HI next S4;
            if CLEAR next S0;

present S4 if LO next S4;
            if HI next S5;
            if CLEAR next S0;

present S5 if LO next S5;
            if HI next S6;
            if CLEAR next S0;

present S6 if LO next S6;
            if HI next S7;
            if CLEAR next S0;

present S7 if LO next S7;
            if HI next S8;
            if CLEAR next S0;

present S8 if LO next S8;
            if HI next S9;
            if CLEAR next S0;

present S9 if LO next S9;
            if HI next S0;
            if CLEAR next S0;

out Z;           /* output Z */
}

```

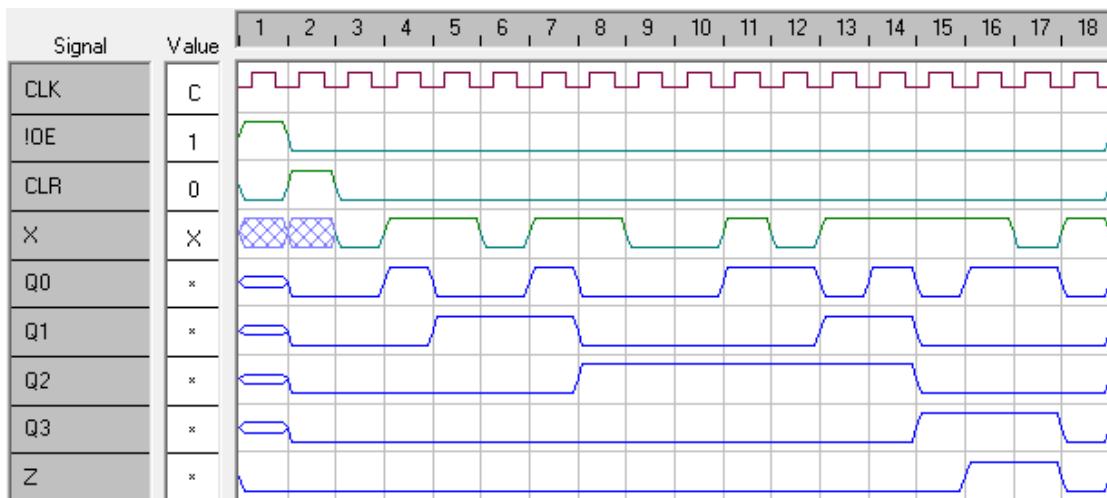
2. ทำการทดสอบการทำงานของไอซี PLD ที่ทำขึ้นว่าสามารถทำงานได้ถูกต้องตามที่ออกแบบไว้ หรือไม่ โดยการสร้าง Simulation file (SI) ที่เขียนขึ้นจาก Timing diagram โดยที่จะกำหนดค่าเอาท์พุทไว้ก่อน หรือไม่กำหนดค่าไว้ ซึ่งโปรแกรมจะหาค่าเอาท์พุทให้จากวงจรที่ทำขึ้น ดังนี้

```
Name      Counter;
PartNo   IOT02;
Date     30/7/22;
Revision 01;
Designer Engineer;
Company  KMITL;
Assembly None;
Location None;
Device   g16v8;

ORDER: CLK, !OE, CLR, X, Q0, Q1, Q2, Q3, Z;

VECTORS:
C10X***** /* test HI Impedance */
C01X***** /* clear to state 0 */
C000***** /* input LO ( 0 ) */
C001***** /* input HI ( 1 ) to state 1 */
C001***** /* input HI ( 1 ) to state 2 */
C000***** /* input LO ( 0 ) */
C001***** /* input HI ( 1 ) to state 3 */
C001***** /* input HI ( 1 ) to state 4 */
C000***** /* input LO ( 0 ) */
C000***** /* input LO ( 0 ) */
C001***** /* input HI ( 1 ) to state 5 */
C000***** /* input LO ( 0 ) */
C001***** /* input HI ( 1 ) to state 6 */
C001***** /* input HI ( 1 ) to state 7 */
C001***** /* input HI ( 1 ) to state 8 */
C001***** /* input HI ( 1 ) to state 9 */
C000***** /* input LO ( 0 ) */
C001***** /* input HI ( 1 ) to state 0 */
```

3. ทำการทดสอบ Simulation file โดยบรรทัดที่เขียนว่า ORDER เป็นชื่อขาไอซีที่จะใช้ทดสอบ สัญญาณ สำหรับ Vector 1 เป็นการป้อนสัญญาณ !OE = 1 ทำให้อาท์พุท Q ทั้ง 4 เส้นเป็น Hi Impedance และที่ Vector 2 ป้อน CLR = 1 จะทำให้เป็นสภาวะเริ่มต้นของวงจรได้อาท์พุทของหน่วยความจำทั้งหมดเป็น 0 หลังจากนั้นตั้งแต่ Vector ที่ 3 จึงเริ่มป้อนอินพุต X ค่า 0110110010111101 ได้อาท์พุทของหน่วยความจำ คือ Q0 ถึง Q3 และเอาท์พุท Z ตามลำดับค่าที่ป้อน โดยจะต้องได้เป็น Timing diagram ดังในรูป



รูป แสดง Timing diagram

การสร้าง Simulation file สามารถทำได้ตามขั้นตอนดังนี้

- เปิด Simulator program เลือก File > New คลิกที่ Design file และเปิด PLD file
- เลือก Signal > Add Signal ตามจำนวนเส้นของสัญญาณที่ต้องการจาก Timing diagram
- เลือก Signal > Add Vector ตามจำนวนครั้งของสัญญาณที่ต้องการจาก Timing diagram
- กดคลิกขวาที่ Timing diagram เลือกระดับสัญญาณต่างๆ ที่ต้องการทางอินพุต รวมทั้ง CLK และ CLEAR
- เลือก Simulator > Run Simulator จะได้ค่าเอาท์พุตของสัญญาณ และ Simulation file โดยค่าที่ได้จาก การทดสอบมีความหมายอยู่ในตารางที่ 3

ตารางที่ 3 แสดงความหมายของค่าใน Simulation file

Test Value	Used with	Description
0	Input	Drive low
1	Input	Drive high
C	Input	Drive (clock) low, high, low (in 1 cycle)
K	Input	Drive (clock) high, low, high (in 1 cycle)
L	Output	Test if low
H	Output	Test if high
Z	Output	Test for high impedance
X	Input or Output	High or Low (don't care)
N	Output	not tested
*	Output	simulator determines what the value is
..	Input	Encloses values to be expanded from a specified BASE
.. ..	Output	Encloses values to be expanded from a specified BASE

4. จาก PLD file ให้ทำการ Compile โปรแกรม ซึ่งจะได้ไฟล์อิจิกฟังก์ชันเก็บอยู่ใน sim file โดยมี พล็อกพาร์เมตเตอร์ส่วนของ Logic Equation คือ สมการของ D flipflop ขา Q₀ ถึง Q₃ และ ขา Output Z ได้ สมการดังนี้

```

Q0.d => !CLR & Q0 & !Q1 & !Q2 & Q3 & !X
          # !CLR & !Q0 & !Q1 & !Q2 & Q3 & X
          # !CLR & Q0 & !Q3 & !X
          # !CLR & !Q0 & !Q3 & X

Q1.d => !CLR & Q0 & Q1 & !Q3 & !X
          # !CLR & Q0 & !Q1 & !Q3 & X
          # !CLR & !Q0 & Q1 & !Q3

Q2.d => !CLR & Q0 & Q1 & Q2 & !Q3 & !X
          # !CLR & Q0 & Q1 & !Q2 & !Q3 & X
          # !CLR & !Q0 & Q1 & Q2 & !Q3
          # !CLR & !Q1 & Q2 & !Q3

Q3.d => !CLR & Q0 & !Q1 & !Q2 & Q3 & !X
          # !CLR & Q0 & Q1 & Q2 & !Q3 & X
          # !CLR & !Q0 & !Q1 & !Q2 & Q3

Z => Q0 & !Q1 & !Q2 & Q3

```

โดยสัญลักษณ์ทาง Arithmetic และ Logic ที่ใช้ใน sim file มีความหมายดังนี้

+	Addition	!	NOT
-	Subtraction	&	AND
*	Multiplication	#	OR
/	Division	\$	XOR
%	Modulus		
**	Exponentiation		

5. ให้ทำการ Compile โปรแกรม PLD เพื่อให้ได้ JEDEC file (JED) ซึ่งจะนำไปใช้ในการโปรแกรม
วงจรล็อกิกลงในตัวไอซี โดยที่

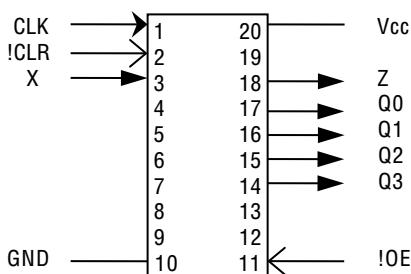
G0 หมายถึงไฟปีกตามหนังข้อมูลที่ Fuse หมายเลขอ

F0 หมายถึงทำการ Fill ค่า 0 ลงในตำแหน่งข้อมูลทั้งหมด คือให้ Fuse ทุกตัวในไอซีมีสถานะเป็นต่อ วงจรไว้ก่อน

C เป็นค่า Check sum ที่ใช้ตรวจสอบความผิดพลาดของข้อมูลในขั้นตอนของการโปรแกรมตัวไอซีโดยใช้เครื่องโปรแกรมไอซี ถ้าค่าของผลรวมบิตต่างๆ จากข้อมูลทั้งหมดในไอซีที่โปรแกรมเสร็จแล้วและใน JEDEC file ตรงกันแสดงว่าตัวไอซีที่โปรแกรมแล้วนี้ใส่ข้อมูลได้ถูกต้อง

L และหมายเลขอ้างฯ คือตำแหน่ง Fuse ในแต่ละบิทที่ได้จากการสร้างวงจรโลจิก เรียงจากหมายเลขอ้างต้นในแต่ละบรรทัดไปตามลำดับ บรรทัดละ 1 เกом ได้ผลลัพธ์เป็น datum สมการดังนี้

6. จาก JEDEC File ที่ได้ให้ทำการโปรแกรมพิวส์ลงในตัวไอซี 16V8 โดยใช้เครื่องโปรแกรม ซึ่งจะได้ไอซีที่ทำหน้าที่เป็นวงจรนับ โดยมีตำแหน่งขาไอซีทั้งหมดที่ใช้งาน รวมทั้งขาของไฟที่ใช้ป้อนให้วงจรทำงาน แสดงโครงสร้าง

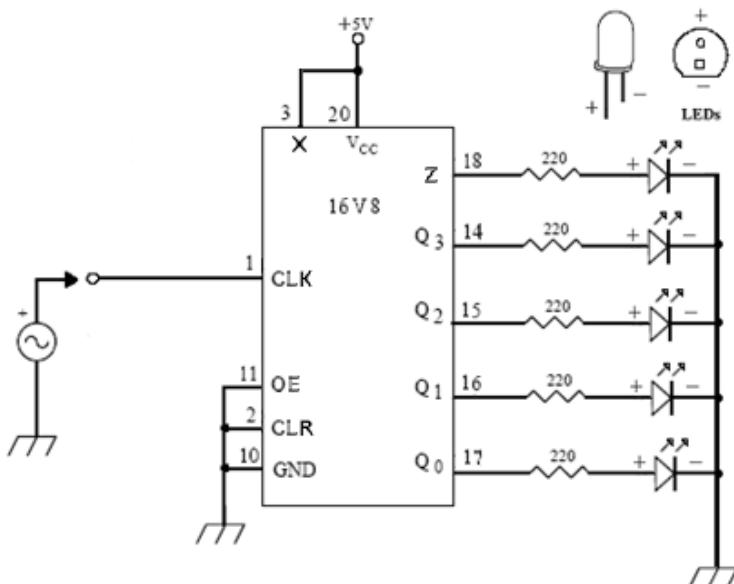


7. ให้ดาวน์โหลดทั้งหมดที่ใช้งานของไอซีที่โปรแกรมเสร็จแล้ว โดยคุณได้จาก sim file หรือ JEDEC file ที่ถูกสร้างขึ้นมาตอน Compile

8. ให้หัดลองแก้ไข Simulation file โดยให้ป้อนข้อมูลเข้าที่ไอซีตามค่าของรหัสประจำตัวนักศึกษา 4 หลัก สุดท้าย เช่น รหัส 2579 ค่าที่ป้อนเป็นเลขฐานสองคือ 0010010101111001 โดยป้อนค่าจากช้ายไปขวา และบันทึกผลแต่ละขาของ Simulation ที่ได้จาก Timing Diagram ลงในตารางที่ 4

ตารางที่ 4 ผลการทดสอบ

9. ประกอบวงจรที่จะใช้ในการทดลองไอซี 16V8 เพื่อจำลองการทำงานของไอซี 7493 ที่เป็น Binary Counter โดยให้ขาอินพุท X ต่อกับไฟบวก และให้ป้อนอินพุทเข้าทางขา Clock แทน โดยนำแต่ละขาเอาท์พุท Q ของ D flipflop ที่สร้างเป็นวงจรนับ ชาที่ 14 ถึง 17 ต่อเข้ากับตัว LED และตัวต้านทาน 220 โอห์มในลักษณะอนุกรม เพื่อแสดงผลของสัญญาณเอาท์พุทเมื่อมี脉冲จิกเป็น 1 จะทำให้ LED สว่าง



10. การกระพริบของ LED ในแต่ละดวงจะมีลักษณะเป็นวงจรนับเลขฐาน 2 โดยนับได้ตั้งแต่ สูนี จนถึงจำนวนเท่าไร
11. ให้นักศึกษาสังเกตว่าสัญญาณเอาท์พุตของ ไอซีมีการเปลี่ยนแปลงทุกครั้ง เมื่อเทียบกับสัญญาณ อินพุต ที่ขอบขาขึ้นหรือขอบขาลงของสัญญาณอินพุต
12. LED ที่เป็นสัญญาณเอาท์พุต Z จะติดสว่างเมื่อมีนับถึงเลขอะไร
13. ทดสอบออกแบบ PLD file โดยออกแบบวงจร Sequence Detector ที่จะให้ Output Sequence เป็น Logic 1 ทุกครั้งเมื่อมีนับตรวจจับได้ว่า Input Sequence ที่ป้อนเข้ามาในวงจรเป็น Logic 1 ที่มีจำนวน ห้าหมุดเท่ากับ 5 และจะเริ่มนับจำนวน Input ใหม่ทุกครั้งที่มี Output ออกไปแล้ว นอกจากข้อกำหนด ดังกล่าวแล้วจะจะให้ Output เป็น Logic 0 ตลอด โดยให้แก้ไขและเพิ่มในส่วน Sequenced STATE ของ PLD file ตามด้านล่างนี้

```

Sequenced STATE {
    Present S0      if LO      next _____
                    if HI      next _____
                    if CLEAR   next S0
    present S1      if LO      next _____
                    if HI      next _____
                    if CLEAR   next S0
    present S2      if LO      next _____
                    if HI      next _____
                    if CLEAR   next S0
    present S3      if LO      next _____
                    if HI      next _____
                    if CLEAR   next S0
    present S4      if LO      next _____
                    if HI      next _____
                    if CLEAR   next S0
    present S5      if LO      next _____
                    if HI      next _____
                    if CLEAR   next S0
                    out   Z;
}

```

14. ให้วาดรูป Timing diagram ที่ได้จากการทดสอบสัญญาณ Simulation file ว่าผลลัพธ์สามารถทำงานได้ ถูกต้องตามที่ออกแบบไว้ โดยสมมติอินพุตที่ป้อนเข้าที่ไอซีเป็นรหัสประจำตัวนักศึกษา 4 หลักสุดท้าย
15. ให้เขียน Logic Equation ที่ออกแบบไว้ คือ สมการของ D-flipflop ขา Q₀ ถึง Q₃ และ Output Z

Q0.d = _____

01236249 FUNDAMENTAL OF DIGITAL SYSTEM DESIGN

Q1.d = _____

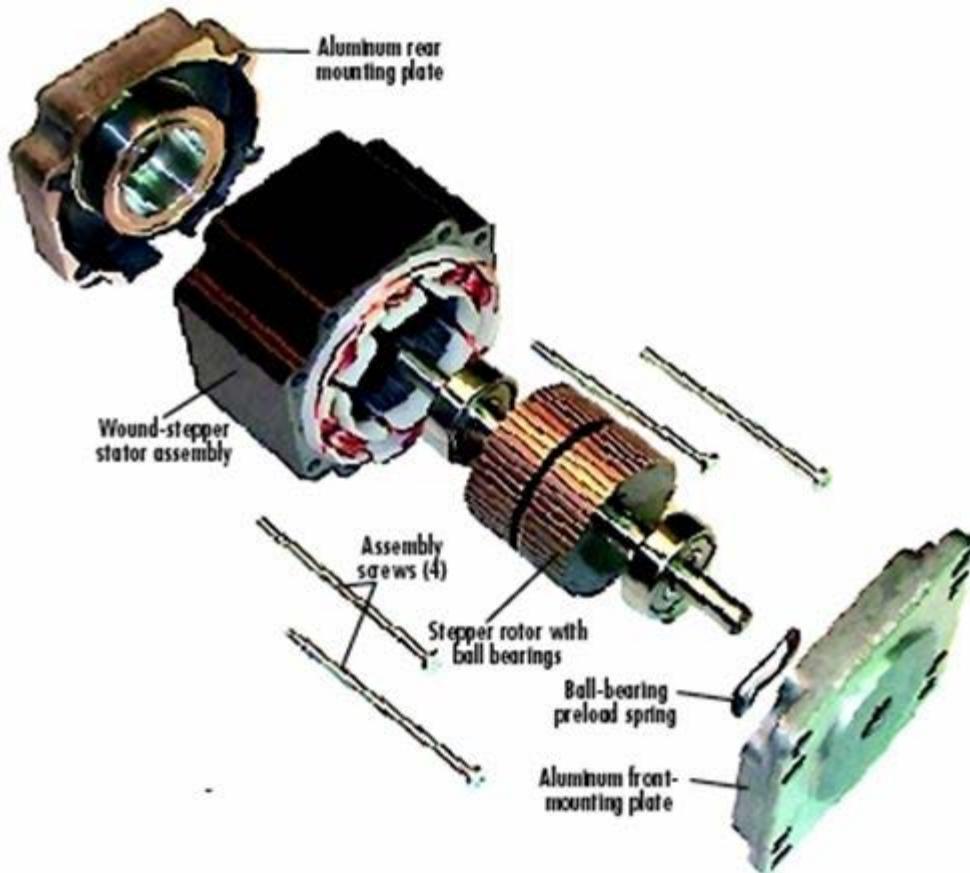
Q2.d = _____

Q3.d = _____

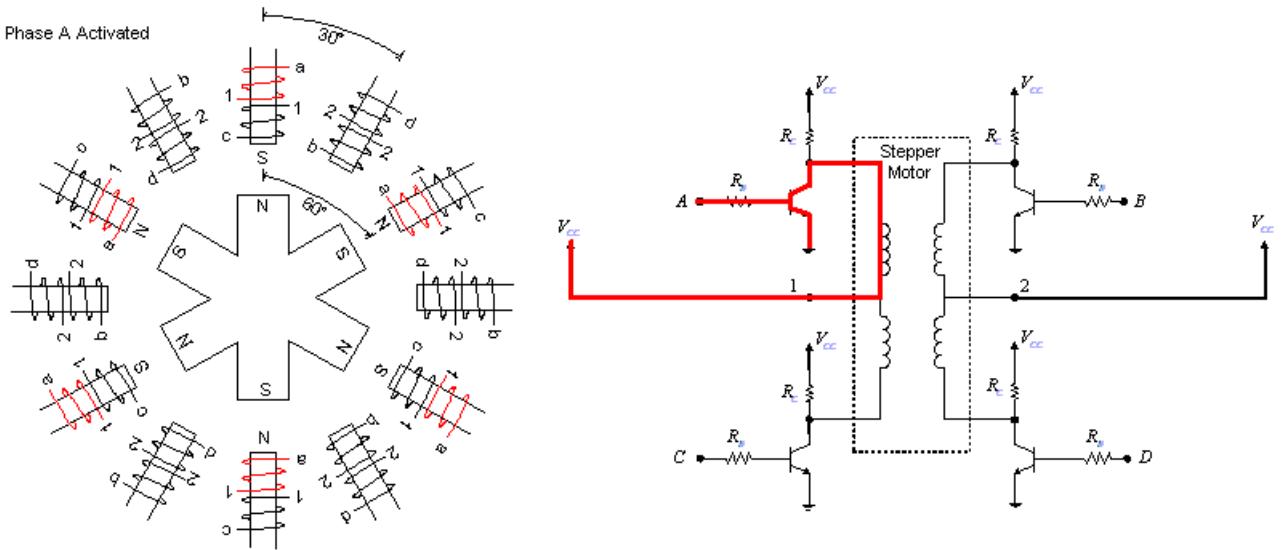
Z = _____

16. จากสมการที่ได้ D-flipflop Q_3 มีการใช้งานหรือไม่ เพราะอะไร

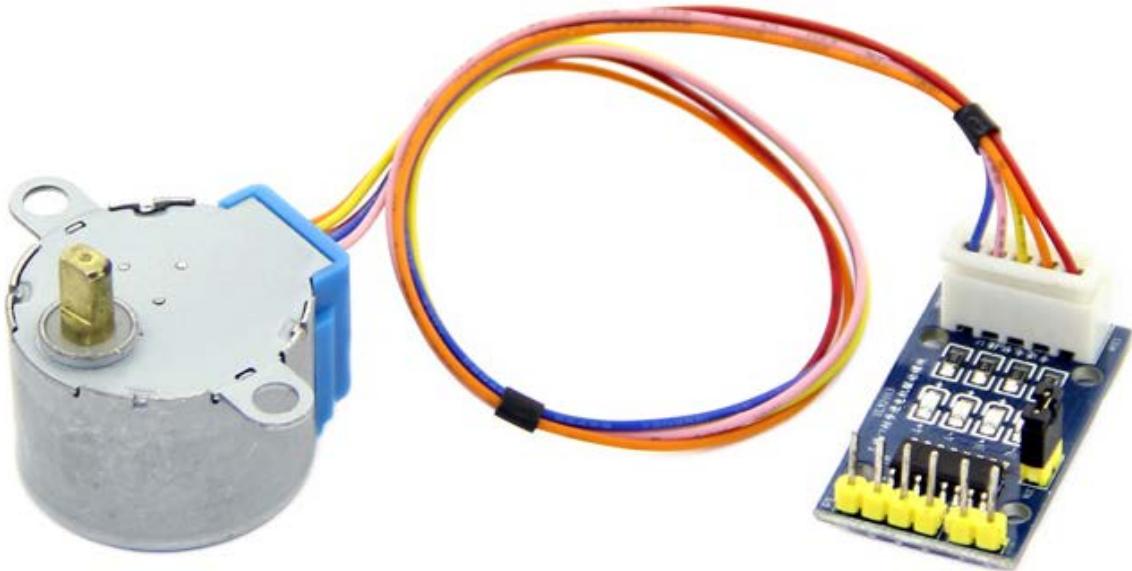
Stepper Motor หรือ **Stepping Motor** กือ มอเตอร์ที่มีการหมุนเป็นขั้นๆ (Step) เมื่อมีสัญญาณไฟฟ้าที่เป็นพัลส์มาป้อนเข้าที่สายตัวปั๊มอเตอร์จะทำให้เกนกลางของมอเตอร์หมุนเป็นมุมคงที่ คือมีค่าเป็นองศาต่อสเต็ป ทำให้สามารถควบคุมตำแหน่งของการหมุนได้อย่างแม่นยำ โดยไม่ต้องใช้การควบคุมแบบป้อนกลับ (Feedback Control) หรืออาศัยตัวตรวจสอบการหมุนมาควบคุมตำแหน่ง การควบคุมสเต็ปเปอร์มอเตอร์จะใช้ไมโครคอนโทรลเลอร์ส่งสัญญาณดิจิตอลมาควบคุมบังคับทิศทาง และความเร็วในการหมุนของแกนสเต็ปเปอร์มอเตอร์ได้โดยตรง โดยไม่ต้องอาศัยแปรรูปอินพุต จึงเป็นที่นิยมใช้ในอุปกรณ์ที่ต้องการควบคุมตำแหน่งและมุมได้อย่างแม่นยำ เช่น บริการ เตอร์ สแกนเนอร์ ชาร์ดดิสก์ กล้องวงจรปิด เครื่องปรับอากาศ และอุปกรณ์ของรถยนต์ เป็นต้น โดยที่ Stepping Motor จะต้องตัวจะมีความละเอียดของมุมหมุนที่แตกต่างกันขึ้นอยู่กับโครงสร้างการผลิต



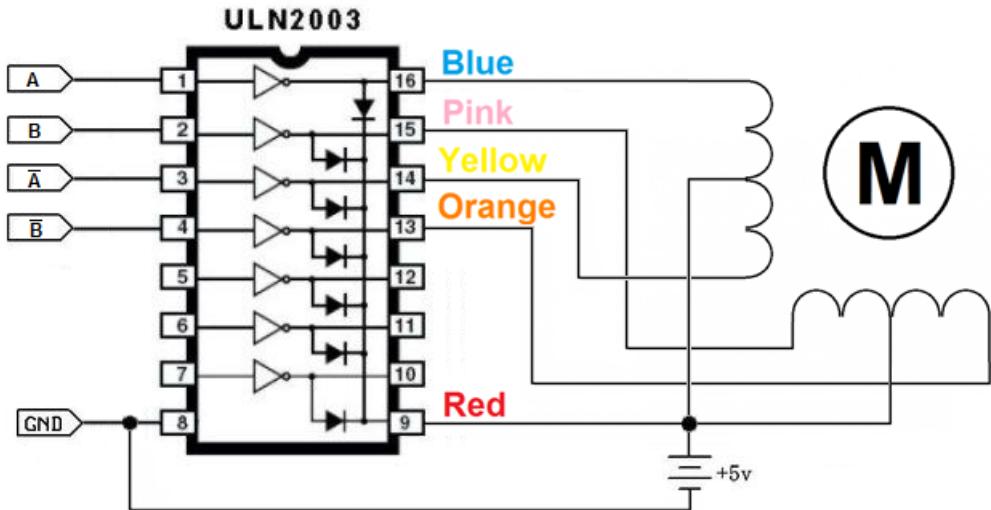
การทำงานของ Stepper Motor จะต้องป้อนแรงดันไฟฟ้าไปที่ขดลวดที่ติดตั้งบนสเตเตอร์ ให้ถูกต้องตามจังหวะเพื่อไปบังคับให้แม่เหล็กถาวร (Permanent Magnet) บนแกนโรเตอร์หมุนไปตามทิศทางที่กำหนด โดยทั่วไปสเต็ปเปอร์มอเตอร์แบ่งได้ 2 แบบ ตามลักษณะของโครงสร้างการต่อขดลวดภายในมอเตอร์ คือ Unipolar และ Bipolar ซึ่งหลักในการขับของ Stepper Motor ทั้งสองแบบทำงานจะทำงานคล้ายกัน คือการป้อนพัลส์เป็นช่วงๆเข้าไปยังขดลวดต่างๆ เพื่อให้ Stepper Motor หมุนไปตามองศาที่ต้องการ โดยปกติแบบ Bipolar จะมีสายไฟต่อเส้น 4 การต่อวงจรจะต้องใช้วงจร H-Bridge เข้ามาช่วยเพื่อกลับทิศทางของสนามแม่เหล็กภายในส่วนแบบ Unipolar จะมีสายไฟต่อเส้น 5 แต่การทำงานจะควบคุมจะทำได้ยากกว่า เนื่องจากไม่ต้องกลับทิศของกระแสไฟที่ป้อนเข้าไปที่ขดลวด



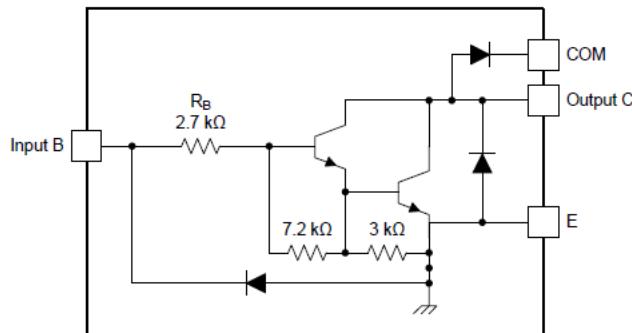
การทำงานของ Stepper Motor นั้นต้องใช้กระแสสูงเพื่อป้อนเข้าขดลวดให้ทำงาน ดังนั้นไอซี 16V8 จะไม่สามารถเชื่อมต่อโดยตรงได้ จึงต้องมีชุดขับกระแสใช้ไอซีเบอร์ ULN2003 การทดลองจะใช้วงจรควบคุมที่เป็น 4Phase Stepper Motor Driver ตามในรูป



การทำงานของวงจรจ่ายแรงดันไฟฟ้า 5V เข้าที่จุดต่อร่วม Common Anode ไปยัง Stepping Motor และป้อนสัญญาณจาก Microcontroller เป็นล็อกิกต่างๆจำนวน 4 บิต เข้าไปยัง IC 2003 ที่ทำหน้าที่เป็น Driver เพื่อขับเฟลต์ให้กับ Stepping Motor ทั้ง 4 เส้นเพื่อให้ลงกราดตามจังหวะของสัญญาณที่ป้อน ซึ่งสามารถดูการทำงานทั้ง เฟสได้จาก 4LED ที่ต่ออยู่บนบอร์ด โดยมีวงจรดังรูป



โดยที่วงจรภายในแต่ละขาจะใช้ทรานซิสเตอร์ต่อเป็นวงจรคาร์ลิงตัน (Darlington) เพื่อขับกระแสทำให้สามารถใช้กระแสได้ถึง 500 mA และมีชื้อทึกไคลโอด (Schottky Diode) ซึ่งเป็นไคลโอดที่มีค่าแรงดันต่ำคร่อมบนนำกระแสแล้วทำงานได้ที่ความถี่สูง มากหน้าที่ ป้องกันแรงดันไฟหักนกลับ (Negative Undershoot) ที่เกิดจากการทำงานของมอเตอร์ ซึ่งจะเป็นอันตรายทำให้วงจรควบคุมเสียหายได้



ULN2003A Block Diagram

- ให้เชื่อมต่อ Stepper Motor กับวงจร ULN2003A เข้ากับไอดี 16V8 ทางขา 16 ถึง 19 และต่อแหล่งจ่ายไฟฟ้า
- ป้อนโปรแกรม StepperMotor.PLD เพื่อกำหนดการทำงานของ Stepping Motor ดังนี้

```

StepperMotor;
PartNo 10T001;
Date 6/10/22;
Revision 01;
Designer Engineer;
Company KMITL;
Assembly None;
Location None;
Device g16v8;
```

```

/** Input Pins */
PIN 1      = CLK;      /* register clock */
PIN 11     = IOE;      /* register output enable */
```

```

/** Output Pins */
PIN [16..19] = [Q0..3]; /* register 4-bit */

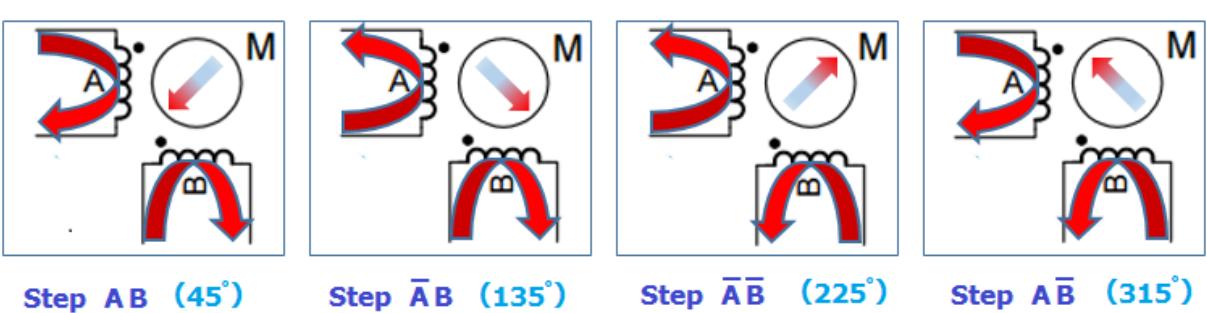
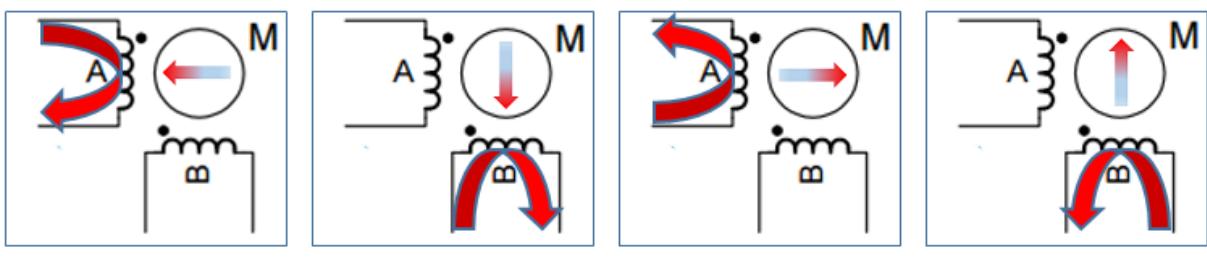
/** Logic Equations */

field STATE = [Q3..0]; /* declare state bit field */
$define S0 'b'0000
$define S1 'b'0001
$define S2 'b'0010
$define S3 'b'0011
$define S4 'b'0100
$define S5 'b'0101
$define S6 'b'0110
$define S7 'b'0111
$define S8 'b'1000
$define S9 'b'1001
$define S10 'b'1010
$define S11 'b'1011
$define S12 'b'1100
$define S13 'b'1101
$define S14 'b'1110
$define S15 'b'1111

Sequenced STATE {
    present S0      next S0;
}

```

การควบคุมการหมุนของสเตปเปอร์มอเตอร์ สามารถทำได้โดยการป้อนแรงดันไฟฟ้าคงที่เข้าไปที่ขั้วของขดลวดที่ควบคุมการหมุน เพื่อบังคับให้แม่เหล็กถาวรนําแกนโรเตอร์หมุนไปตามทิศการบังคับของขดลวดที่ติดตั้งบนสเตเตอร์ โดยจะต้องป้อนแรงดันให้ถูกต้องตามจังหวะเพื่อให้แกนโรเตอร์หมุนตั้งรูป



ตามตัวอย่างขดลวดแต่ละขดห่างกัน องศา การหมุน 90แบบง่ายที่สุดทำได้โดยการจ่ายกระแสไฟเข้าไปกระแสตู้นีละขดลวดในแต่ละเฟสตามลำดับ A B A-bar B-bar ถ้าหากต้องการให้กระแสไฟหลINLINEเฟสใดก็จะทำให้สถานะของเฟสนั้นเป็น High ซึ่งจะทำให้เกิดสนามแม่เหล็ก เพื่อไปคัดแม่เหล็กถาวรที่อยู่บนโรเตอร์ให้เคลื่อนที่ โดยมีทิศทางการหมุนตามลำดับการจ่ายกระแสไฟเข้าที่ขดลวดอยู่ รอบ 1 จังหวะต่อการหมุน 4

3. ทำการทดสอบการทำงานของไอซี PLD ที่ทำขึ้นว่าสามารถทำงานได้ถูกต้องตามที่ออกแบบไว้หรือไม่ โดยการสร้าง Simulation file (SI) ที่เขียนขึ้นจาก Timing diagram โดยที่จะกำหนดค่าเอาท์พุทไว้ก่อน หรือไม่กำหนดก็ได้ ซึ่งโปรแกรมจะหาค่าเอาท์พุทให้จากการที่ทำขึ้น ให้ป้อนโปรแกรม StepperMotor.SI เพื่อทดสอบการทำงานของ Stepping Motor ดังนี้

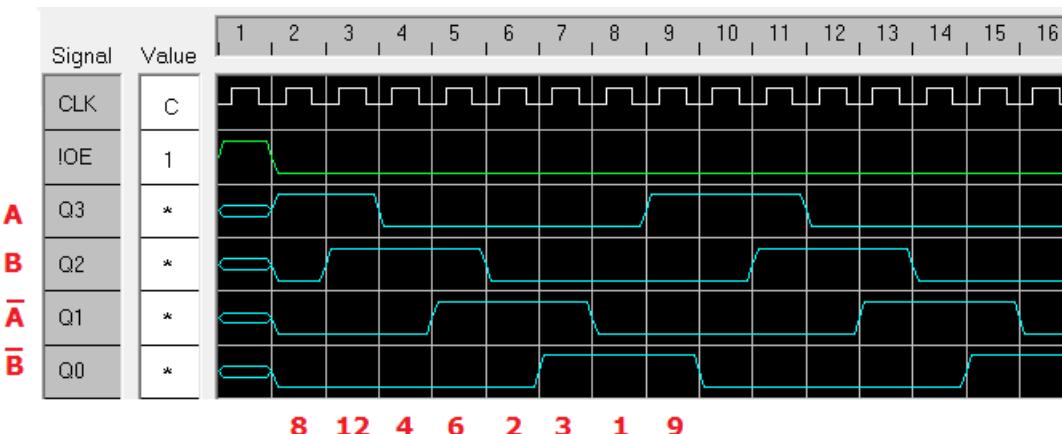
```
StepperMotor;
PartNo 10T001;
Date 6/10/22;
Revision 01;
Designer Engineer;
Company KMITL;
Assembly None;
Location None;
Device g16v8;
```

ORDER: CLK, !OE, Q3, Q2, Q1, Q0;

VECTORS:

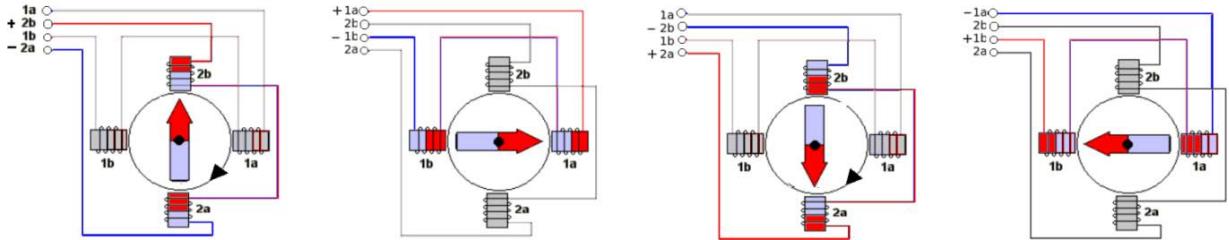
```
C1**** /* test HI Impedance */
C0**** /* state 1 */
C0**** /* state 2 */
C0**** /* state 3 */
C0**** /* state 4 */
C0**** /* state 5 */
C0**** /* state 6 */
C0**** /* state 7 */
C0**** /* state 8 */
C0**** /* state 9 */
C0**** /* state 10 */
C0**** /* state 11 */
C0**** /* state 12 */
C0**** /* state 13 */
C0**** /* state 14 */
C0**** /* state 15 */
```

4. ทำการทดสอบ Simulation file โดยบรรทัดที่เขียนว่า ORDER เป็นชื่อขาไอซีที่จะใช้ทดสอบสัญญาณ ส่วน Vector 1 เป็นการป้อนสัญญาณ !OE = 1 ทำให้อาท์พุท Q ทั้ง 4 เส้นเป็น Hi Impedance หลังจากนั้นตั้งแต่ Vector ที่ 2 จึงเริ่มการทดสอบการทำงานของโปรแกรม ได้อาท์พุทของ หน่วยความจำคือ Q0 ถึง Q3 โดยจะต้องได้เป็น Timing diagram ดังในรูป



5. จาก PLD file ที่ได้ จะต้องทำการสร้าง JEDEC file ซึ่งจะนำไปใช้ในการโปรแกรมวงจรโลジคัลในตัว ไอซี

การควบคุมการหมุนแบบ Wave Drive จะเป็นการป้อนกระแสไฟให้กับขดลวดของสเต็ปเพอร์มอเตอร์ที่จะขับ โดยจะป้อนกระแสเรียงตามลำดับกันไป ดังนั้นกระแสที่ไหลในขดลวดจะไหลในทิศทางเดียวกันทุกขด ควบคุมแบบนี้ทำได้ง่ายแต่แรงขับของสเต็ปปีซึ่งมอเตอร์ที่ไม่มีน้อย ความเร็วที่ได้จากการหมุนของสเต็ปเพอร์มอเตอร์จะขึ้นอยู่กับความถี่ของการป้อนกระแสไฟฟ้าให้กับขดลวดในแต่ละครั้งตามลำดับ ถ้าความถี่มีค่า น้อยลงมอเตอร์จะหมุนช้า และถ้าความถี่มีค่ามากก็อาจจะไม่เสถียรได้



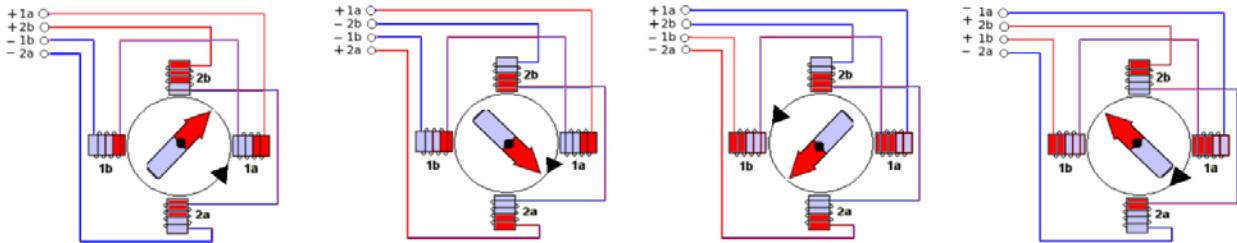
6. การทดลองการทำงาน จากโปรแกรมในข้อ 2 ให้เพิ่มโปรแกรมการส่งข้อมูลไปที่ Stepping Motor โดยใช้การควบคุมแบบ Wavedrive เพื่อจ่ายไฟให้ทำงานครั้งละ 1 ขด ซึ่งก็คือให้ทำงานครั้งละ 1 เฟส ทดลองการทำงานของโปรแกรมและให้บันทึกผลที่ได้

```
Sequenced STATE {
    present S0    next S8;
    present S8    next S4;
    present S4    next S2;
    present S2    next S1;
    present S1    next S8;
}
```

7. Stepping Motor ทำงานมีค่ามุมหมุนต่อ Step เท่ากับ 1.4 องศา
8. ถ้าต้องการให้หมุน 1 รอบจะต้องใช้ทั้งหมดเท่ากับ 256 Step
9. การหมุนของแต่ละ Step ในโปรแกรมเป็นการหมุนตามเข็มหรือทวนเข็มนาฬิกา 0111

10. ถ้าต้องการให้หมุนในทิศทางตรงกันข้ามกันต้องแก้ไขโปรแกรมในล่วงหน้า
 $S0 \rightarrow S1$
 $S1 \rightarrow S2$
 $S2 \rightarrow S3$
 $S3 \rightarrow S4$
 $S4 \rightarrow S5$

การควบคุมการหมุนแบบ Full Step จะเป็นการป้อนกระแสไฟให้กับขดลวดของสเต็ปเพอร์มอเตอร์แบบทีละ 2 เฟสพร้อมกัน โดยต้องป้อนกระแสเรียงตามลำดับกันไปครั้งละ 2 ขด ดังนั้นจึงมีกระแสไฟหลักในขดลวดของมอเตอร์มากขึ้นซึ่งทำให้มอเตอร์มีแรงบิดในการหมุนมากขึ้นตามไปด้วย



11. จากโปรแกรมในข้อ 2 ให้เพิ่มโปรแกรมการส่งข้อมูลไปที่ Stepping Motor โดยใช้การควบคุมแบบ Fullstep ให้ทดลองการทำงานของโปรแกรมและบันทึกผลที่ได้

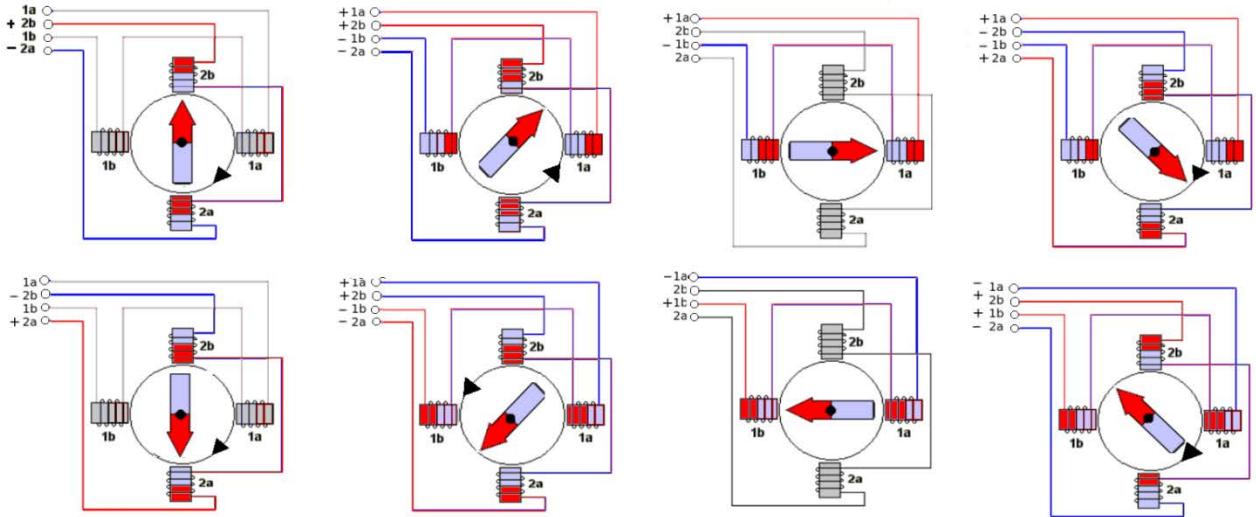
```
Sequenced STATE {
    present S0    next S12;
    present S12   next S6;
    present S6    next S3;
    present S3    next S9;
    present S9    next S12;
}
```

12. Stepping Motor ทำงานแบบ Full Step มีมุมหมุนต่อ Step เท่ากับแบบ Wavedrive หรือไม่

.....
เหตุ因

13. การทำงานในแต่ละ Step ของแบบ Full Step กับแบบ Wavedrive อยู่ในตำแหน่งองศาเดียวกันหรือไม่
.....
ไม่ใช่จะต้องเดินหลาย

การควบคุมการหมุนแบบ Half Step เป็นการป้อนกระแสไฟให้กับขดลวดของสเต็ปเพอร์มอเตอร์ครั้งละ 1 เฟส และ 2 เฟส สลับกันไป ทำให้สเต็ปเพอร์มอเตอร์มีความละเอียดของการหมุนเพิ่มขึ้น 2 เท่า ซึ่งจะหมุนได้ครึ่งคลื่นร่องสเต็ป โดยที่ไม่ต้องปรับเปลี่ยนชาร์ดแวร์เพียงแต่แก้ไขโปรแกรมวิธีการจ่ายกระแสไฟเข้าขดลวดให้เพิ่มมากขึ้น



14. จากโปรแกรมในข้อ 2 ให้เพิ่มโปรแกรมการส่งข้อมูลไปที่ Stepping Motor ด้วยวิธีการควบคุมแบบ Half Step โดยนำข้อมูลของทั้งสองแบบมารวมกัน ให้ทดลองการทำงานของโปรแกรมและบันทึกผลที่ได้

```
Sequenced STATE {
    present S0      next S8;
    present S8      next S12;
    present S12     next S4;
    present S4      next S6;
    present S6      next S2;
    present S2      next S3;
    present S3      next S1;
    present S1      next S9;
    present S9      next S8;
}
```

15. Stepping Motor ทำงานแบบ Half Step มีมุ่งหมายต่อ Step เท่ากับ 0.7 องศา

16. ถ้าต้องการให้แบบ Half Step หมุน 1 รอบจะต้องใช้ทั้งหมดเท่ากับ 512 Step

17. จากโปรแกรมในข้อ 14 ให้แก้ไขโปรแกรมเพิ่ม โดยกำหนดว่าเมื่อกดสวิตช์ให้ Stepping Motor หมุน และถ้าปล่อยสวิตช์ให้ Stepping Motor หยุดหมุน ตามตัวอย่างของโปรแกรมดังนี้

```
/** Input Pins */
PIN 1      = CLK;          /* register clock */
PIN 11     = !OE;          /* register output enable */
PIN 2      = CLR;          /* clear input */
PIN 3      = X;            /* input X */

/** Output Pins */
PIN [16..19] = [Q0..3];   /* register 4-bit */
PIN [12..15] = [Z3..0];   /* output 4-bit */

/** Declarations and Intermediate Variable Definitions */

field mode = [CLR,X];      /* declare mode control field */
LO      = mode:0;           /* define input low */
HI      = mode:1;           /* define input high */
CLEAR  = mode:[2..3];       /* define clear mode */

/** Logic Equations */
```

```

field STATE = [Q3..0]; /* declare state bit field */
$define S0 'b'0000 /* define state */
$define S1 'b'0001
$define S2 'b'0010
$define S3 'b'0011
$define S4 'b'0100
$define S5 'b'0101
$define S6 'b'0110
$define S7 'b'0111
$define S8 'b'1000
$define S9 'b'1001
$define S10 'b'1010
$define S11 'b'1011
$define S12 'b'1100
$define S13 'b'1101
$define S14 'b'1110
$define S15 'b'1111

Sequenced STATE {
    present S0    next S8;
    present S8    if LO      next S8;
                  if HI      next S12;
    present S12   if LO      next S12;
                  if HI      next S4;
    present S4    if LO      next S4;
                  if HI      next S6;
    present S6    if LO      next S6;
                  if HI      next S2;
    present S2    if LO      next S2;
                  if HI      next S3;
    present S3    if LO      next S3;
                  if HI      next S1;
    present S1    if LO      next S1;
                  if HI      next S9;
    present S9    if LO      next S9;
                  if HI      next S8;
}

```

18. จากโปรแกรมในข้อ 17 ให้เพิ่มโปรแกรม .SI เพื่อทดสอบการทำงานของ Stepping Motor ดังนี้

ORDER: CLK, !OE, CLR, X, Q3, Q2, Q1, Q0;

VECTORS:

```

C10X**** /* test HI Impedance */
C011**** /* clear to state 0 */
C001**** /* input HI (1) to state 1 */
C001**** /* input HI (1) to state 2 */
C001**** /* input HI (1) to state 3 */
C001**** /* input HI (1) to state 4 */
C001**** /* input HI (1) to state 5 */
C001**** /* input HI (1) to state 6 */
C001**** /* input HI (1) to state 7 */
C001**** /* input HI (1) to state 8 */
C001**** /* input HI (1) */
C001**** /* input HI (1) */
C000**** /* input LO (0) */

```

```

Sequenced STATE {
    present S0 next S8;
    present S8  if LO next S12;
                  if HI next S9;
    present S12 if LO next S4;
                  if HI next S8;
    present S4  if LO next S6;
                  if HI next S12;
    present S6  if LO next S2;
                  if HI next S4;
    present S2  if LO next S3;
                  if HI next S6;
    present S3  if LO next S1;
                  if HI next S2;
    present S1  if LO next S9;
                  if HI next S3;
    present S9  if LO next S8;
                  if HI next S1;
}

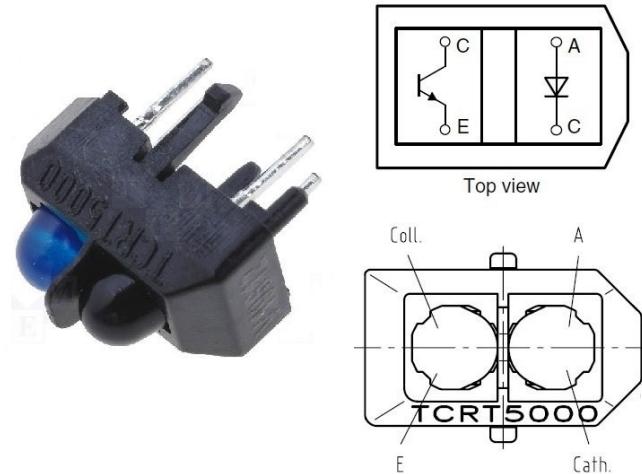
```



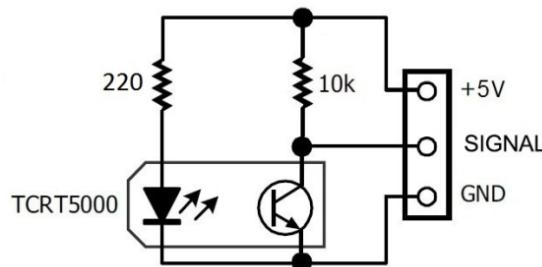
19. จากโปรแกรมในข้อ 18 ให้แก้ไขโปรแกรมเพิ่ม โดยกำหนดค่าเมื่อมีการกดสวิตช์ให้ Stepping Motor

หมุนตามเข็มนาฬิกา และถ้าปล่อยสวิตช์ให้ Stepping Motor หมุนทวนเข็มนาฬิกา

การทดลอง Sensor สำหรับตรวจจับแสง TCRT5000 โดยใช้ Infrared Emitting Diode ส่งแสงที่มีความยาวคลื่น 950 นาโนเมตรออกไป และเมื่อรับแสงใช้ Photo Transistor ทำหน้าที่รับแสงสะท้อนสามารถตรวจจับได้ที่ระยะห่างระหว่าง 0.2 ถึง 15 mm



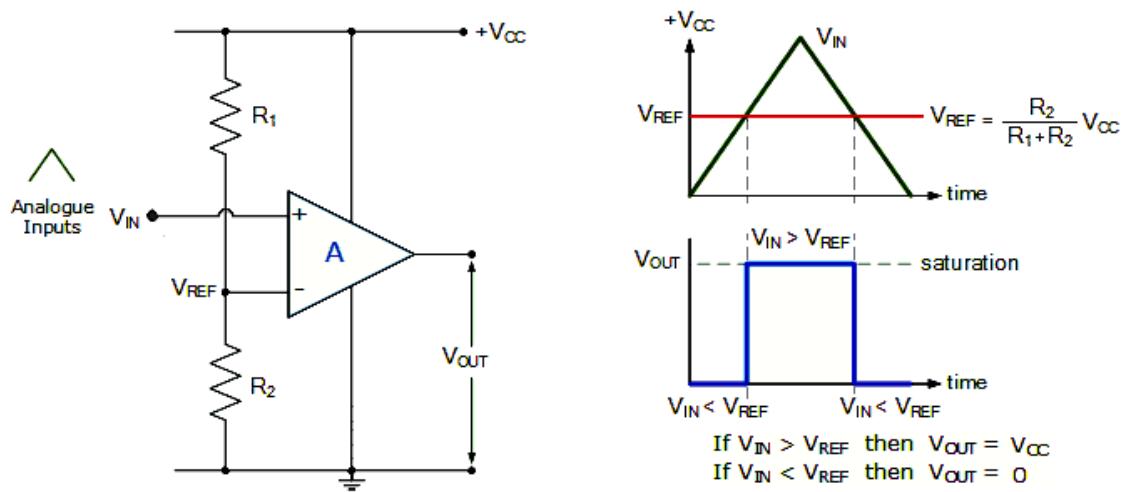
ใช้แรงดันไฟฟ้าในการทำงาน 5V ให้สัญญาณเอาท์พุตออกแบบมาแบบอนาล็อก



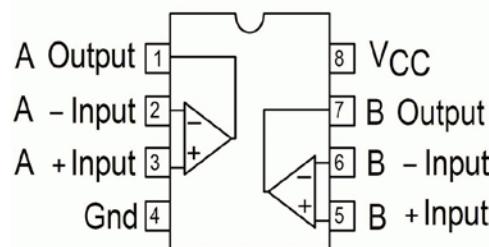
จากสัญญาณอนาล็อกที่ออกมานี้ออกมาที่เอาท์พุทจะต้องแปลงให้เป็นสัญญาณดิจิตอล โดยใช้ Operational Amplifier เบอร์ LM358 ทำเป็นวงจร comparator ให้ป้อนสัญญาณอนาล็อกเข้าที่อินพุตไปยังขั้วบวกของ OpAmp ส่วนขั้วลบจะถูกต่อ กับ ความต้านทานที่ทำหน้าที่เป็นวงจรแบ่งแรงดัน (Voltage Dividers) ที่ประกอบด้วยความต้านทาน R_1 และ R_2 ต่ออนุกรมคร่อมแหล่งจ่ายไฟ 5V แรงดันจากแหล่งจ่ายไฟจะถูกแบ่งระหว่างความต้านทานทั้งสอง ให้แรงดันเอาท์พุตเป็น V_{REF} ซึ่งก็คือแรงดันไฟที่คร่อมที่ R_2 ค่าที่ได้เป็น

$$V_{REF} = \frac{R_2}{R_1 + R_2} V_{cc}$$

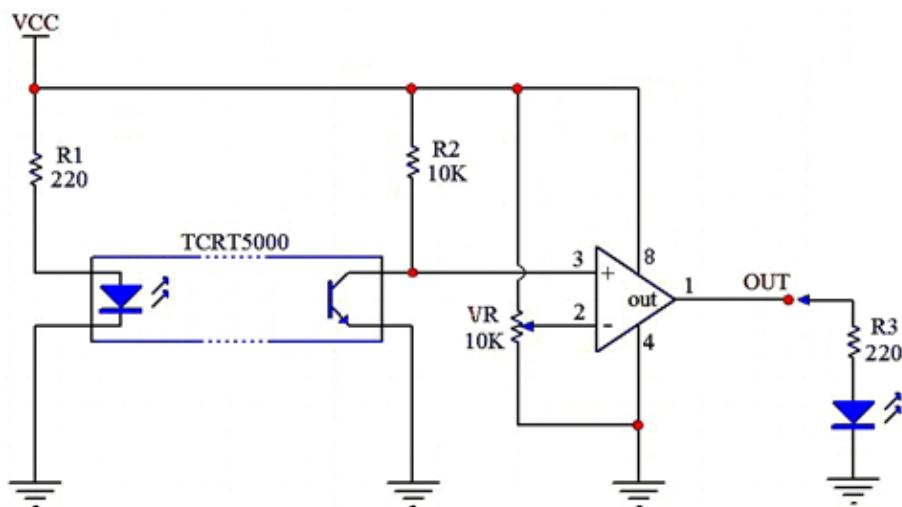
ถ้า V_{IN} มีค่ามากกว่า V_{REF} จะทำให้เอาท์พุตออกแบบเป็น โลจิก 1 และถ้า V_{IN} มีค่าน้อยกว่า V_{REF} จะให้เอาท์พุตออกแบบเป็น โลจิก 0 ดังแสดงในรูป



ตำแหน่งขาของไอซี LM358



5. ให้ค่าวงจรตามในรูป โดยใช้ข้อมูลของ OpAmp ต่อกับตัวความด้านท่านที่ปรับค่าได้ (VR)
ขนาด 10 K เพื่อทำหน้าที่เป็นวงจรแบ่งแรงดัน (Voltage Dividers) ให้เป็น V_{REF}



6. ให้ทดลองปรับค่าความด้านท่าน (VR) เพื่อให้ Sensor ตรวจจับ TCRT5000 ทำงานให้ได้
ระยะห่างที่มากที่สุดและบันทึกค่าที่ได้

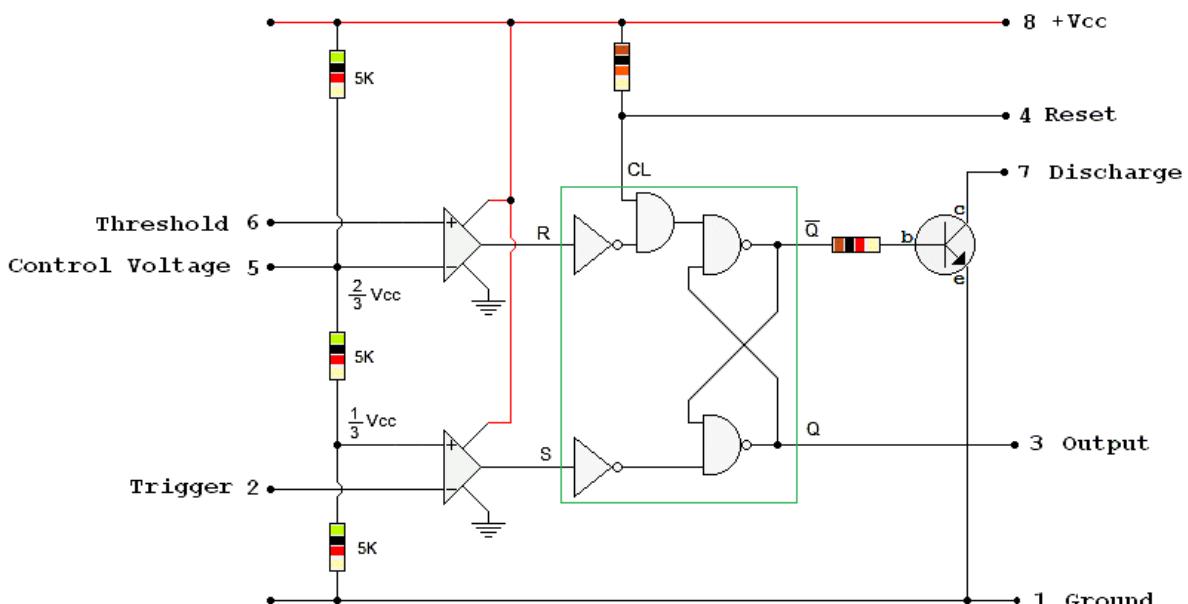
ระยะห่างที่ได้ 19 cm

ค่า R_1 ที่วัดได้ 98.1 Ω

ค่า R_2 ที่วัดได้ 9.61 Ω

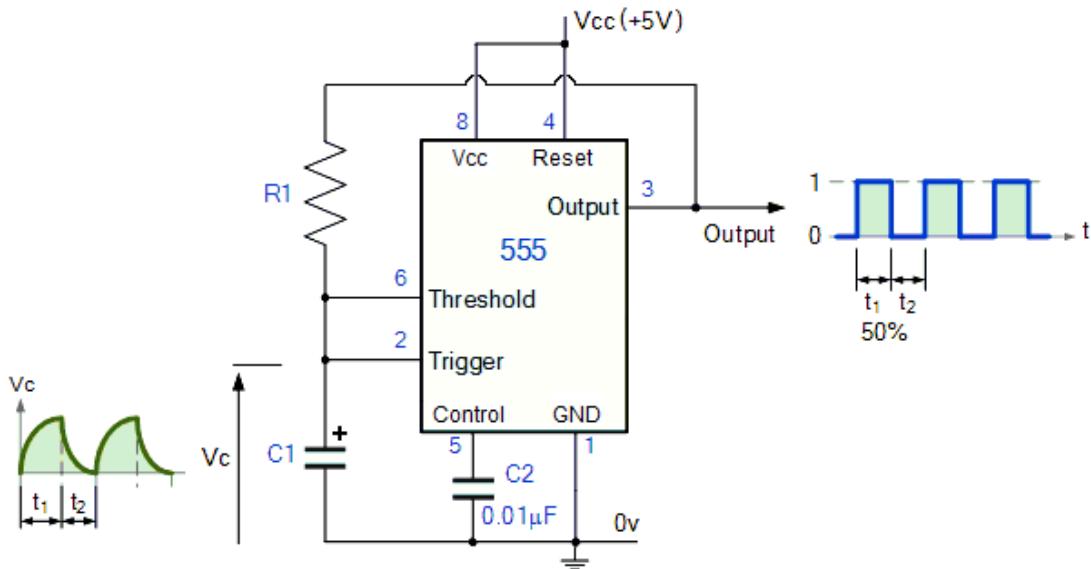
7. ให้นำสัญญาณที่เอาท์พุตขา 1 ของ LM358 ป้อนเข้าไปอินพุตของวงจรนับขาที่ 1 ของไอซี 16V8 แทนการใช้ความต้านทาน 10K ต่ออนุกรมกับ switch และอินบາຍเพลที่ได้ว่าแตกต่างกันอย่างไร。~~แต่ต้องรู้ว่า ก. ถ้าตั้งค่าต่อ 10K ก็จะต้องต่อ 10K กับ switch แต่ต้องไม่ต่อ 10K กับ Sensor แล้วต้องไปเป็น 10K แต่ถ้าตั้งค่าต่อ 5K ก็จะต้องต่อ 5K กับ Sensor แล้วต้องไม่ต่อ 5K กับ switch~~

การทดลองสร้างวงจรกำเนิดความถี่ (Oscillator) โดยใช้ไอซี NE555 เป็นวงจรรวม (Integrated circuit) ที่ใช้ควบคุมสัญญาณเวลาหรือใช้เป็นตัวจับเวลา (timer) สามารถใช้สร้างรูปสัญญาณต่างๆ เช่น สัญญาณรูปเหลี่ยม (square) สัญญาณรูปไซน์ (sine wave) และสัญญาณรูปสามเหลี่ยม (triangle wave) การออกแบบวงจรทำได้โดยการต่ออุปกรณ์ภายนอกร่วมกับไอซี 555 โดยจะใช้ค่าคงตัวเวลา (time constant) ในการอัดและคลายประจุของวงจรตัวต้านทาน-ตัวเก็บประจุ (RC circuit) เป็นตัวกำหนดค่าเวลาของสัญญาณ

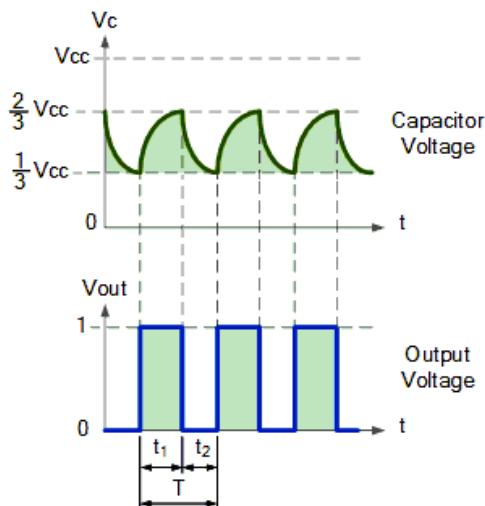


ภายในไอซี 555 นั้นจะประกอบด้วย ตัวต้านทานที่มีค่าเท่ากันจำนวน 3 ตัวที่ต่อเป็นวงจรแบ่งแรงดัน (voltage divider) เพื่อใช้ค่าระดับแรงดันให้กับตัวเปรียบเทียบแรงดัน (voltage comparator) เพิ่มกับแหล่งจ่ายไฟ Vcc โดยที่ตัวเปรียบเทียบแรงดันกับค่าเทรสโไฮล (Threshold) จะมีค่าเท่ากับ $2V_{cc}/3$ และที่ตัวเปรียบเทียบแรงดันกับขาทริกเกอร์ (Trigger) จะมีค่าเท่ากับ $V_{cc}/3$ ขนาดของตัวเปรียบเทียบแรงดันจะต่อเข้ากับขา S และขา R ของ RS flip-flop

การทดลองเป็นการอุดแบบให้ IC 555 เป็นวงจรอะสเตเบล (Astable) ซึ่งทำหน้าที่เป็นวงจรกำเนิดความถี่ที่มีค่า Duty Cycle เท่ากับ 50% โดยอาศัยการอัดและคลายประจุผ่านตัวต้านทานตัวเดียว เพื่อให้ความกว้างของสัญญาณเอาท์พุตมีค่าเท่ากัน โดยใช้วงจรที่ดังรูป



การทำงานของวงจรในสภาวะเริ่มต้นระดับแรงดันที่ C_1 จะมีค่าเป็น 0 โวลต์ และถูกส่งเข้า วงจร voltage comparator ทำให้ขา S = 1, R = 0 เป็นผลให้อาทพุทธของ RS flip-flop และอาทพุทธของ ไอซี 555 ได้เป็นล็อกิก 1 ช่วงนี้อาทพุทธของ ไอซี 555 จะทำการอัดประจุของตัวเก็บประจุ C_1 ผ่านทาง R₁ จนทำให้ขา 2 และ 6 มีแรงดันมากกว่า $V_{cc}/3$ จะทำให้ S = 0, R = 0 ซึ่งจะเป็นการคงสภาวะเดิม จน กะรังหี้ขา 2 และ 6 มีแรงดันมากกว่า $2V_{cc}/3$ จะทำให้ S = 0, R = 1 เป็นผลให้อาทพุทธของ RS flip-flop และอาทพุทธของ ไอซี 555 ได้เป็นล็อกิก 0 ในช่วงนี้จะเป็นการคายประจุของตัวเก็บประจุ C_1 ผ่าน R₁ จน ขา 2 และ 6 มีระดับแรงดันลดลงน้อยกว่า $2V_{cc}/3$ และ $V_{cc}/3$ ตามลำดับ ซึ่งจะทำให้อาทพุทธของ RS flip-flop และอาทพุทธของ ไอซี 555 กลับเป็นล็อกิก 1 อีกครั้ง ดังรูป



จากรูปจะเห็นได้ว่าความกว้างของสัญญาณฟลัสร์บนจะถูกกระตุ้น และขณะไม่ถูกกระตุ้นจะมีค่า เท่ากัน สามารถคำนวณเวลาและความถี่ของ สัญญาณเอาตพุตได้ดังนี้

$$T_1 = T_2 = \ln(2) R_1 C_1$$

$$f = \frac{1}{T_1 + T_2} = \frac{1}{2 \ln(2) R_1 C_1} = \frac{0.72}{R_1 C_1}$$

8. ถ้ากำหนดให้ตัวเก็บประจุ $C_1 = 47 \text{ nF}$ และตัวความต้านทาน $R_1 = 10\text{K}\Omega$ จะได้ความถี่เท่ากับ 1.5319 Hz

9. ให้ต่อวงจรที่ใช้อีซี 555 ตามในรูปด้านบน โดยใช้ตัวเก็บประจุ $C_1 = 47 \text{ nF}$ และตัวความต้านทาน $R_1 = 10\text{K}\Omega$ เพื่อสร้างวงจรไฟกระพริบ โดยใช้ตัวความต้านทาน 220Ω และหลอด LED ต่อเข้าที่เอาท์พุทของวงจร

10. จากการทดลองหลอด LED จะมีการกระพริบติด-ดับ มีช่วงเวลาติด-ดับเท่ากับ 0.326 วินาที

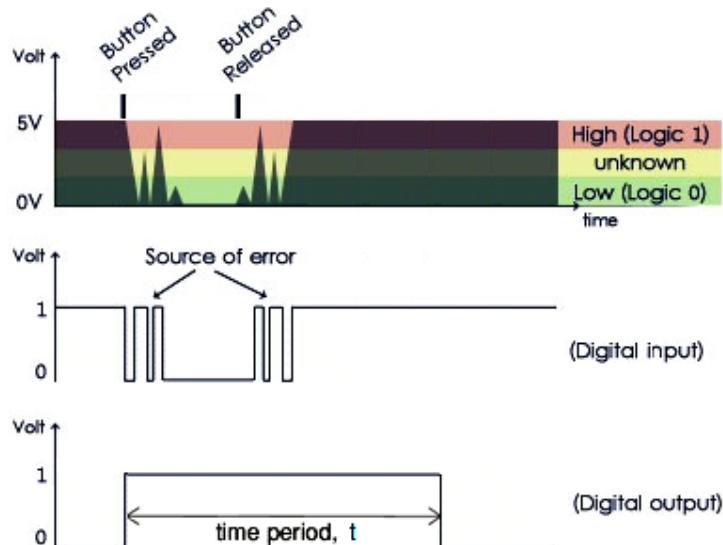
11. ให้ทดลองเปลี่ยนค่าความต้านทาน R_1 ให้น้อยลงและอธิบายผลที่ได้กับหลอด LED

11.0.๑ LED กระพริบเร็วๆ เมื่อกำหนดเวลา

.....

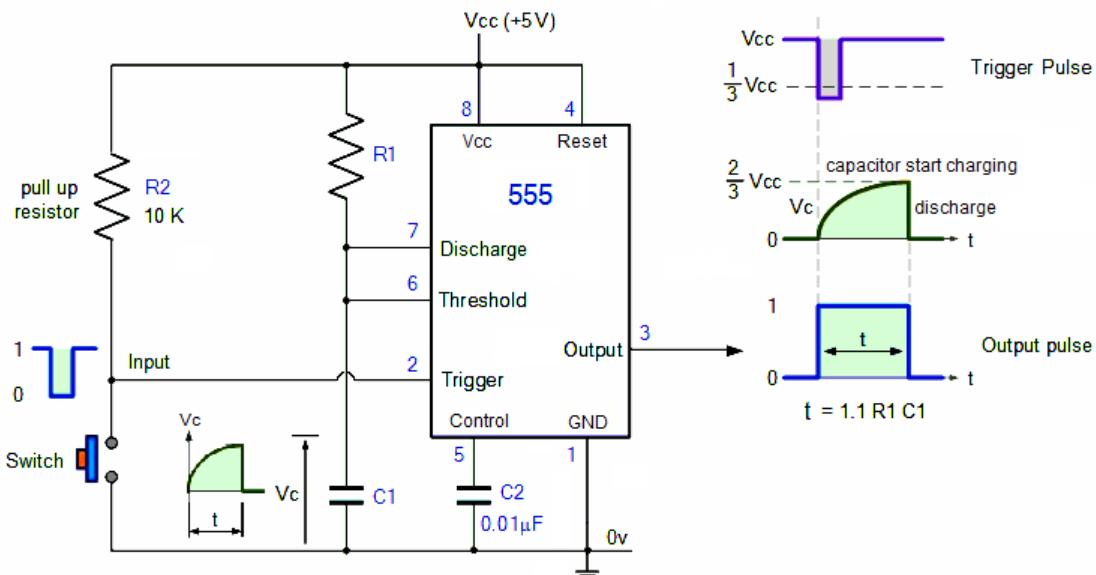
12. ให้ป้อนสัญญาณที่เอาท์พุทขา 3 ของ NE555 เข้าไปที่อินพุทของวงจรนับขา 14 ของไอซี 7493 และอธิบายผลที่ได้ 11.0.๒ LED กระพริบช้าๆ

การทดลองของจร Debounce เมื่อมีการกดปุ่มสวิตช์หน้าสัมผัส (Contact) ที่เป็นโลหะที่ทำการสัมผัสนั้น มักจะมีปัญหาการ Bounce ของสัญญาณที่เกิดจากการสั่นหรือยังไม่แนบสนิทดี ทำให้เกิดเป็นสัญญาณรบกวนที่สัญญาณแก่วงว่งขึ้นว่างลง ได้เป็นสัญญาณดิจิตอลที่มีการสลับไปมาระหว่าง High และ Low อย่างรวดเร็ว ก่อนที่จะเข้าสู่สภาวะเสถียร ซึ่งจะทำให้ค่าที่ส่งออกไปนั้นเกิดความคลาดเคลื่อน ซึ่งเป็นปัญหาที่ต้องแก้โดยใช้โปรแกรมหรืออุปกรณ์ภายนอกเข้ามาช่วยเพื่อป้องกันเหตุการณ์ที่จะเกิดขึ้นนี้ ถ้าหากเราทำการกดสวิตช์เพียง 1 ครั้ง จะทำให้วงจรได้รับสัญญาณที่แก่วงดังรูป



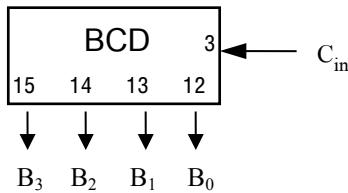
จากรูปด้านบนที่เป็นสัญญาณ Digital input สามารถแก้ปัญหาการ Bounce ได้เป็น Digital output ด้วยวิธีใช้การออกแบบวงจรโน�ต์เกล็อก (Monostable) โดยใช้ไอซี 555 ซึ่งเป็นวงจรที่มีการทำงานคือเมื่อถูก

กระตุ้นจากทางด้านอินพุทจะให้สัญญาณเอกสารที่พุทธเป็นพัลส์อคกามา 1 ถูก ที่มีความกว้างของสัญญาณ (t) ตามที่กำหนด โดยในสภาวะปกติขา 2 ที่เป็นอินพุทธของวงจรจะมีค่าเท่ากับ Vcc เนื่องจากมีความต้านทาน 10K ต่อแบบปูลอปไวร์ (pullup) ไว้ ทำให้ขา 7 จะเสมือนว่าต่อลบกราวด์มีค่า Vc เป็น 0 เมื่อมีการกดสวิตช์ จะมีสัญญาณเข้ามาทริกทำให้ขา 2 เป็น 0 ตัวคอมพารเตอร์จะส่งให้ขา S ของ RS flip-flop เป็น 1 เป็นผลให้อาพุทธเป็นโลจิก 1 และทรานซิสเตอร์เปิดวงจรขา 7 จะไม่ต่อลบกราวด์ ทำให้เกิดกระแสผ่าน R1 อัดประจุเข้าตัวเก็บประจุ C1 จนถึงช่วงเวลา $t = 1.1 R1 C1$ แล้ว จะทำให้ขา 6 มีค่า $V_c > 2V_{cc}/3$ ขา R ของ RS flip-flop จะเป็น 1 เป็นผลให้อาพุทธของวงจรกลับเป็นโลจิก 0 และทรานซิสเตอร์ขา 7 ทำงานจะถูกกลั่นวงจรลงกราวด์ ทำให้ C1 ถูกภายในประจุ (Discharge) ทันทีและขา 6 ได้เป็น 0 จะทำให้ตัวคอมพารเตอร์ส่งให้ขา R = 0 และ S = 0 เป็นผลให้ RS flip-flop คงสถานะเดิมและเอาท์พุทจะขังคงเป็นโลจิก 0 จนกว่าจะมีสัญญาณเข้ามาทริก



13. ถ้ากำหนดให้ ตัวเก็บประจุ $C_1 = 47 \mu\text{F}$ และตัวความต้านทาน $R_1 = 10\text{K}\Omega$ จะได้สัญญาณเอาท์พุทธเป็นพัลส์ที่มีช่วงเวลา t เท่ากับ 0.517
14. ให้ต่อวงจรที่ใช้ไอซี 555 ตามในรูปด้านบน โดยใช้ตัวเก็บประจุ $C_1 = 47 \mu\text{F}$ และตัวความต้านทาน $R_1 = 10\text{K}\Omega$ โดยใช้ตัวความต้านทาน 220Ω และ หลอด LED ต่อเข้าที่เอาท์พุทธของวงจร
15. ให้ทดลองเปลี่ยนค่าความต้านทาน R_1 และอัตราข่ายผลที่ได้กับหลอด LED
LED 闪光จะเร็วขึ้นเมื่อตัวเก็บประจุมากกว่า-
16. ให้ป้อนสัญญาณที่เอาท์พุทขา 3 ของ NE555 เข้าไปที่อินพุทธของวงจรนับขา 14 ของไอซี 7493 และอัตราข่ายผลที่ได้ LED 闪光เร็วขึ้น
-
17. ให้นักศึกษาสังเกตว่าสัญญาณเอาท์พุทของ ไอซีมีการเปลี่ยนแปลงทุกครั้ง เมื่อเทียบกับสัญญาณอินพุต ที่ขอบขาขึ้นหรือขอบขาลงของสัญญาโนินพุต (0 ไป 1)

การทดลอง การทำงานจะเปลี่ยนเลข Binary เป็น BCD จะสามารถทำได้โดยนำเลข Binary ในแต่ละบิต มาคูณด้วยเลข 2^n เมื่อ n เป็นค่าตามตำแหน่งบิต คือ 0, 1, 2, 3, ... (จากขวาไปซ้าย) และนำค่าที่ได้ทั้งหมดมา加กัน การทดลองจะสร้างวงจรที่แปลงเลข Binary เป็น BCD จำนวน 1 หลัก ซึ่งจะต้องใช้สายสัญญาณเท่ากับ 4 เส้น ค่านับฐาน 2 ที่ป้อนเข้ามายังเป็นแบบอนุกรม โดยจะต้องป้อนส่งค่าหลักที่มีนัยสำคัญสูงสุด (MSB) เข้ามา ก่อน ตัวไอซีที่ได้แสดงดังรูปที่ 9



รูป แสดงการแปลง Binary เป็น BCD

การออกแบบวงจรกำหนดให้ข้อมูล Binary ที่ป้อนเข้ามาทางอินพุทของไอซีเป็นขา C_{in} การป้อนข้อมูล Binary ต้องป้อนเข้าทีละ 1 บิต และวงจรจะต้องจำสถานะก่อนหน้านี้ ดังนั้นจึงต้องมีส่วนของ Memory เพิ่มเข้ามา ในที่นี้จะใช้ D Flip-Flop ที่อยู่ในไอซี จากหลักการแปลง Binary เป็น BCD สามารถเขียนเป็นสมการได้ดัง

$$BCD(\text{Next State}) = 2 * BCD(\text{Present State}) + C_{in}$$

โดยที่ BCD จะเป็นเลขตั้งแต่ 0 ถึง 9 ขาเอาท์พุทของ BCD ที่มี 4 เส้น กำหนดให้เป็น B_3, B_2, B_1 และ B_0 เมื่อแทนค่าต่างๆ ลงในสมการ สามารถเขียนเป็น State Table ได้ดังตารางที่ 2

ตารางที่ 2 State Table

N	Present State				Next State			
	B_3	B_2	B_1	B_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0	C_{in}
1	0	0	0	1	0	0	1	C_{in}
2	0	0	1	0	0	1	0	C_{in}
3	0	0	1	1	0	1	1	C_{in}
4	0	1	0	0	1	0	0	C_{in}
5	0	1	0	1	0	0	0	C_{in}
6	0	1	1	0	0	0	1	C_{in}
7	0	1	1	1	0	1	0	C_{in}
8	1	0	0	0	0	1	1	C_{in}
9	1	0	0	1	1	0	0	C_{in}

จากตารางที่ 2 เมื่อนำไปหาผลจิกฟังก์ชันจะได้ดังนี้

$B_3 B_2$	00	01	11	10
$B_1 B_0$	0	0	0	0
00	0	0	0	0
01	1	0	0	0
11	d	d	d	d
10	0	1	d	d

$B_3 B_2$	00	01	11	10
$B_1 B_0$	0	0	1	1
00	0	0	1	1
01	0	0	1	0
11	d	d	d	d
10	1	0	d	d

$$B_3 = B_0 \bar{B}_1 \bar{B}_2 B_3 + \bar{B}_0 \bar{B}_1 B_2 \bar{B}_3$$

$$B_3 = B_0 B_3 + \bar{B}_0 \bar{B}_1 B_2$$

$$B_2 = \bar{B}_0 \bar{B}_1 \bar{B}_2 B_3 + B_0 B_1 B_2 \bar{B}_3 + B_1 \bar{B}_2 \bar{B}_3$$

$$B_2 = \bar{B}_0 B_3 + B_0 B_1 + B_1 \bar{B}_2$$

$B_1 B_0$	00	01	11	10
$B_3 B_2$	00	0 1	1 1	0
	01	0 0	0 0	1
	11	d	d	d
	10	1	0	d

$B_1 B_0$	00	01	11	10
$B_3 B_2$	00	1 1	1 1	1
	01	1 1	1 1	1
	11	d	d	d
	10	1 1	d	d

$$B_1 = \bar{B}_0 \bar{B}_1 \bar{B}_2 B_3 + \bar{B}_0 B_1 B_2 \bar{B}_3 + B_0 \bar{B}_2 \bar{B}_3$$

$$B_0 = \bar{B}_1 B_2 \bar{B}_3 C_{in} + \bar{B}_1 \bar{B}_2 C_{in} + B_1 \bar{B}_3 C_{in}$$

$$B_1 = \bar{B}_0 B_3 + \bar{B}_0 B_1 B_2 + B_0 \bar{B}_2 \bar{B}_3$$

$$B_0 = C_{in}$$

1. จากสมการของวงจรลอจิกทั้งหมด ให้นำไปเขียนเป็น PLD file ที่ชื่อ BCD1.PLD ดังนี้

```

Name      BCD1;
PartNo   IOT03;
Date     30/7/22;
Revision 01;
Designer Engineer;
Company  KMITL;
Assembly None;
Location None;
Device   g16v8;

/** Input Pins */
PIN 1    = CLK;                      /* Register Clock */
PIN 2    = !CLR;                     /* Clear Input */
PIN 3    = CIN;                      /* Data Input */
PIN 11   = !OE;                      /* Register Output Enable */

/** Output Pins */
PIN [15..12] = [B3..0];             /* BCD 4-bit number */

/** Logic Equations */
B0.d =      CIN & !CLR ;           /* Convert B0 (LSB) */
B1.d =      !B0 & B3 & !CLR        /* Convert B1 */
# !B0 & B1 & B2 & !CLR
# B0 & !B2 & !B3 & !CLR ;
B2.d =      !B0 & B3 & !CLR        /* Convert B2 */
# B0 & B1 & !CLR
# B1 & !B2 & !CLR ;
B3.d =      B0 & B3 & !CLR        /* Convert B3 */
# !B0 & !B1 & B2 & !CLR ;

```

2. ในส่วนของ Logic Equation คือสมการ B_0 ถึง B_3 ถ้าไม่ต้องการหาเป็นสมการ สามารถทำได้อีกวิธี โดยนำการออกแบบที่เป็น Sequenced ตามค่าที่กำหนดใน State Table ใส่ลงในโปรแกรม ได้เป็น BCD2.PLD ดังนี้

01236249 FUNDAMENTAL OF DIGITAL SYSTEM DESIGN

```

Name          BCD2;
PartNo       IOT03;
Date         30/7/22;
Revision     02;
Designer     Engineer;
Company      KMITL;
Assembly    None;
Location     None;
Device       g16v8;

/** Input Pins */
PIN 1 = CLK;           /* Register Clock */
PIN 2 = !CLR;          /* Clear Input */
PIN 3 = CIN;           /* Data Input */
PIN 11 = !OE;          /* Register Output Enable */

/** Output Pins */
PIN [15..12] = [B3..0]; /* BCD 4-bit number */

/** Declarations and Intermediate Variable Definitions */
field BCD = [B3..0];           /* declare BCD bit field */
$define S0 'b'0000             /* define BCD states */
$define S1 'b'0001
$define S2 'b'0010
$define S3 'b'0011
$define S4 'b'0100
$define S5 'b'0101
$define S6 'b'0110
$define S7 'b'0111
$define S8 'b'1000
$define S9 'b'1001

field mode = [CLR,CIN];        /* declare mode control field */
CLEAR = mode:[2..3];           /* define clear mode */
CINO = mode:0;                 /* define input Lo */
CIN1 = mode:1;                 /* define input Hi */

/** Logic Equations */
Sequenced BCD {
present S0      if CINO      next S0;
                  if CIN1      next S1;
                  if CLEAR    next S0;

present S1      if CINO      next S2;
                  if CIN1      next S3;
                  if CLEAR    next S0;

present S2      if CINO      next S4;
                  if CIN1      next S5;
                  if CLEAR    next S0;

present S3      if CINO      next S6;
                  if CIN1      next S7;
                  if CLEAR    next S0;

present S4      if CINO      next S8;
                  if CIN1      next S9;
                  if CLEAR    next S0;

present S5      if CINO      next S0;
                  if CIN1      next S1;
                  if CLEAR    next S0;

present S6      if CINO      next S2;
                  if CIN1      next S3;
                  if CLEAR    next S0;

present S7      if CINO      next S4;
                  if CIN1      next S5;
                  if CLEAR    next S0;

present S8      if CINO      next S6;
                  if CIN1      next S7;
                  if CLEAR    next S0;

present S9      if CINO      next S8;
                  if CIN1      next S9;
                  if CLEAR    next S0;
}

```

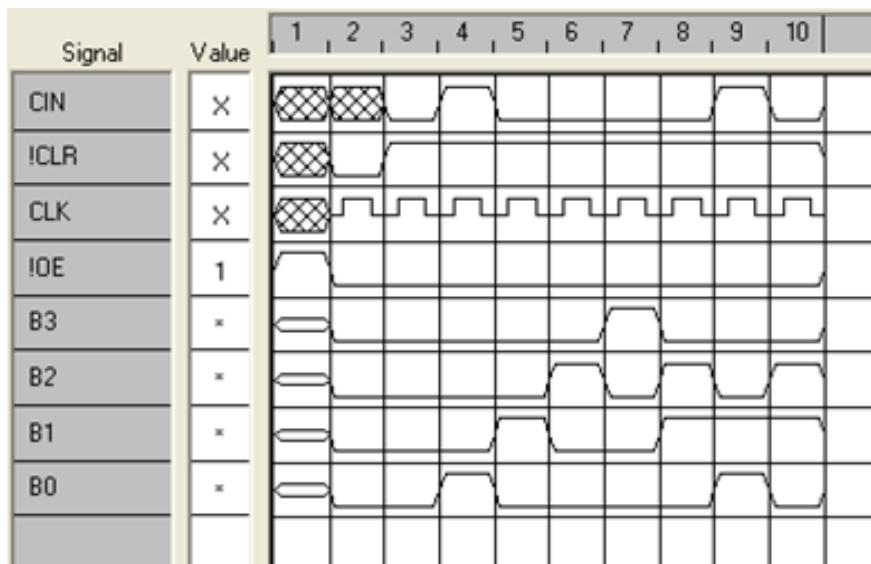
3. การทดสอบการทำงานของไอซี PLD ที่ทำขึ้นว่าสามารถทำงานได้ถูกต้องตามที่ออกแบบไว้หรือไม่ โดยการสร้างเป็น Simulation file (SI) ที่เขียนขึ้นจาก Timing diagram ที่ชื่อ BCD2.SI ดังนี้

```
Name      BCD2;
PartNo   IOT03;
Date     30/7/22;
Revision 02;
Designer Engineer;
Company  KMITL;
Assembly None;
Location None;
Device   g16v8;

ORDER: CIN, !CLR, CLK, !OE, B3, B2, B1, B0;

VECTORS:
XXX1**** /* TEST HI-Z */
X0C0**** /* CLEAR (0) */
01C0**** /* SERIAL INPUT L (0) */
11C0**** /* SERIAL INPUT H (1) */
01C0**** /* SERIAL INPUT L (2) */
01C0**** /* SERIAL INPUT L (4) */
01C0**** /* SERIAL INPUT L (8) */
01C0**** /* SERIAL INPUT L (16) */
11C0**** /* SERIAL INPUT H (33) */
01C0**** /* SERIAL INPUT L (66) */
```

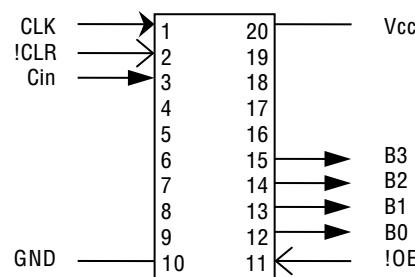
4. ให้ทำการทดสอบ Simulation file โดยที่บรรทัด ORDER เป็นชื่อขาไอซีที่จะใช้ทดสอบสัญญาณ ส่วน Vector 1 เป็นการป้อนสัญญาณ $\text{!OE} = 1$ ทำให้ออกพุท BCD ทั้ง 4 เส้นเป็น Hi Impedance และที่ Vector 2 ป้อน $\text{!CLR} = 0$ จะทำให้เป็นสภาวะเริ่มต้นของวงจร ได้ออกพุทเป็น 0 หลังจากนั้นตั้งแต่ Vector ที่ 3 จึงเริ่มป้อนอินพุต CIN ค่า 01000010B (66) ได้ออกพุท BCD คือ B_0 ถึง B_3 เป็นเลขฐานสิบฐาน 1 หลักที่เป็นหลักหน่วย ตามลำดับค่าที่ป้อน โดยแสดงเป็น Timing diagram ดังรูป



5. จาก PLD file ที่ได้ให้ทำการสร้าง JEDEC file ซึ่งจะนำไปใช้ในการโปรแกรมวงจรล็อกิกลงในตัวไอซี

```
*GO
*FO
*L01024 0111111111111111011110111011101
*L01056 01111111111111110110111101110
*L01280 011111111111111111111101111011101110
*L01312 0111111111111111111111110110111011101
*L01344 01111111111111111111111101111011011111
*L01536 01111111111111111111111101111011101110
*L01568 0111111111111111111111110110111011110
*L01600 01111111111111111111111101110111101110
*L01792 0110101111111111111111110110111101111
*L01824 01101011111111111111111111110111101111
*L01856 0110101111111111111111111111110111101111
*L02048 00001111001100000011000100000000
*L02112 000000000111100001111111111111111111111
*L02144 111111111111111111111111111111111111111111
*L02176 11111111111111111111101
*C2DE6
```

- #### 6. ให้วาดตัวแทนงำข้าໄວซີທີໄດ້



7. ประกอบวงจรที่จะใช้ในการทดลองไอซี 16V8 เพื่อทดสอบการทำงานของวงจรที่ทำหน้าที่เปลี่ยนเลข Binary ไปเป็น BCD โดยให้ป้อนอินพุตเข้าทางขา X และนำขาที่ 14 ถึง 17 ที่เป็นเอาท์พุตของวงจรไปต่อเข้ากับอินพุตขาที่ 1 ถึง 4 ของไอซี 22V10 ที่สร้างเป็นวงจร BCD to 7 Segment Decoder
 8. ทดลองป้อนเลข 13 และ เลขประจำตัวนักศึกษาสองหลักหลังที่แปลงเป็นเลขฐานสองเข้าที่ไอซี และ นับที่กบดุงต่ำลงในตารางที่ 4

ตารางที่ 4 ผลการทดสอบ

เลขฐาน 2	สัญญาณ CLK	ค่าที่ป้อนเข้า Cin	ค่า BCD ที่วัดได้				เลขฐาน 10 ที่คำนวณได้
			B3	B2	B1	B0	
0	C	0					0
01	C	1					1
011	C	1					3
0110	C	0					6
01101	C	1					13

- #### 9. แล้วว่าดูรูป Timing Diagram ที่ได้จากการทดลอง

10. ให้เขียนโปรแกรมทดสอบการทำงานของไอซี PLD เป็น Simulation file (SI) โดยใช้ค่าในตารางที่ 4

11. ให้ทำการ Simulation เพื่อให้ได้ Timing diagram

12. ให้ทำการอธิบายเบริญเพิบผลลัพธ์ที่ได้จากการ Simulation กับ ผลลัพธ์ที่ได้จากการทดลองต่อวงจร ว่าเป็นอย่างไร

.....

- ### 13. ให้วาดรูปวงรั้งทึ่งหมดที่ใช้ในการทดลอง

การทดลองออกแบบบางจาร Sequence Detector ที่ใช้เป็น Key Card โดยอาจจะทำเป็นบัตรชนิดบัตรเจ้ารู บัตรແຄນແມ່ເໜີກ หรือบัตรอื่นๆ ที่บัตรจะบันທຶກຮັສໄວ້ເຕັດຕ່າງກັນໃນແຕ່ລະຫົ່ອງ ເມື່ອນຳບັດສອດເຂົ້າໄປທີ່ຕ້ອນບັດຮອງປະຕູຫຼືອກາຍໃນຫ້ອງພັກ ຈະໄດ້ຄ່າ Input Sequence ປຶ້ອນເຂົ້າມາໃນງາງຈົກທີ່ຈະສ້າງຂຶ້ນນີ້ ຄໍາມີຄ່າ Logic ດຽວຕາມທີ່ກຳທັນດໄວ້ໃນແຕ່ລະຫົ່ອງ ວົງຈະໃຫ້ Output ເປັນ Logic 1 ເພື່ອໃຊ້ໃນເປົ້າໂປ່ດລື້ອກປະຕູຫຼືອເປີດໄຟຟ້າກາຍໃນຫ້ອງ



ຮູບທີ 12 ແສດງຕ້ອຍ່າງ Key Card ທີ່ເປັນບັດເຈົ້າ

ຕ້ອຍ່າງ ໂຮງແຮມແຫ່ງທີ່ເປັນດີກີ່ມີ 4 ອາຄາ ໃນແຕ່ລະອາຄາມີຈຳນວນຂັ້ນທັ້ງໝາດ 9 ຂັ້ນ ແລະ ໃນແຕ່ລະຂັ້ນມີອຸ່ນ 12 ຫ້ອງ ໂດຍການຈັດເຮືອງໝາຍເລີບບັດຈະໃໝ່ມາຕາຮູານທີ່ໄປເປັນຕ້າເລີບ 4 ລັກດັ່ງນີ້

- ໜັກທີ 1 ເປັນໝາຍເລີບອາຄາ
- ໜັກທີ 2 ເປັນໝາຍເລີບຂັ້ນ
- ໜັກທີ 3 ແລະ 4 ເປັນໝາຍເລີບຫ້ອງໃນຂັ້ນນັ້ນ

ສມຸດໜາຍເລີບບັດຄື່ອ 3712 ຈະເປັນຫ້ອງທີ່ອູ່ໃນ ອາຄາ 3 ຂັ້ນ 7 ຫ້ອງທີ່ 12

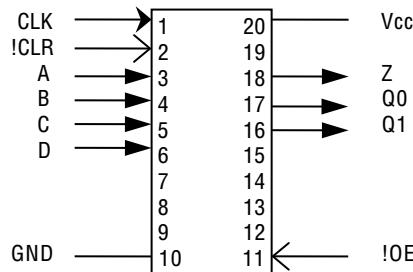
ຈະອຳນວຍແນບນັ້ນຂອງ Key Card ທີ່ໃຊ້ໃນແຕ່ລະຫົ່ອງ ແລະ ມີຮັສທີ່ສັນພັນທີ່ກັບໝາຍເລີບບັດ ໂດຍໃຫ້ງຈາກທີ່ອຳນວຍແນບນັ້ນມີ
ໝາດເລີກແລະໃໝ່ໜ່ວຍຄວາມຈຳນ້ອຍທີ່ສຸດ

ກາຮອກແນບຈະໃໝ່ໃອ໌ທີ່ເປັນ PLD ໂດຍໃຫ້ 1 ຕັວຕ່ອງ 1 ຫ້ອງ ຜົ່ງໃນແຕ່ລະຫົ່ອງຈະບັນທຶກຮັສໄໝ່
ເໝືອນກັນ ໂດຍຮັສທີ່ໄດ້ຈະກຸກສ່າງເຮືອງຕາມລຳດັບແບ່ງເປັນ ອາຄາ, ຂັ້ນ, ຫ້ອງ ອ່າງລະໝ່າງ ທີ່ສິ່ງຕາມຕ້ອຍ່າງຈະ
ເກີນໄດ້ວ່າໃນແຕ່ລະຂ່ອມູນທີ່ສ່າງເຂົ້າມາ ຄື່ອ ອາຄາ, ຂັ້ນ, ຫ້ອງ ຈະມີຄ່າໄດ້ສູງສຸດເຖິງກັບເລີບ 12 ດັ່ງນີ້ຈຶ່ງໃຫ້ງຈາກທີ່
ອຳນວຍແນບນັ້ນມີ Input ທັ້ງໝາດເຖິງກັບ 4 ເສັ້ນ (ຜົ່ງຈະໄດ້ຄ່າສູງສຸດເຖິງກັບ 15 ຢີ້ວີເຖິງກັບເລີບຮູານ 16 ພົ້ນໆໜັກ)

ສມຸດໜາຍເລີບບັດຄື່ອ 3712 ຂ່ອມູນທີ່ສ່າງເຂົ້າມາຄື່ອ 3 , 7 , 12 ເມື່ອເປັນລັບລູານຂ່ອມູນທີ່ໄດ້ຈາກບັດ
ຈະມີ Input 4 ເສັ້ນ ເຂົ້າມາຄື່ອ 0011 , 0111 , 1100 ດັ່ງນີ້ຄໍາເປັນຂ່ອມູນທີ່ເປັນບັດແຄນແມ່ເໜີກ ຈະບັນທຶກຂ່ອມູນເປັນ
03H , 07H , 0CH ແລະ ຄໍາເປັນບັດເຈົ້າ ຈະໄດ້ 3 ແລະ ຄື່ອ 1 ເປັນ 0011 , ແລະ ຄື່ອ 2 ເປັນ 0111 , ແລະ ຄື່ອ 3
ເປັນ 1100 (ໂດຍທີ່ 1 ຄື່ອເຈົ້າ , 0 ຄື່ອໄໝເຈົ້າ) ຂ່ອມູນໃນແຕ່ລະແດວຈະກຸກສ່າງເຂົ້າມາພຽມກັນທີ່ລະ 4 ເສັ້ນ
ຕ່ອນຫົ່ງແລວເຮືອງຕາມລຳດັບແລວທີ່ 1 ຄື່ອ 3 ແຕ່ອ່າງນ້ອຍໃນແຕ່ລະແດວຈະຕ້ອງມີ 1 ຮູ ໂດຍໃຫ້ແຕ່ລະແດວຈະມີຄ່າເລີບ
ຕັ້ງແຕ່ 1 ຄື່ອ 15 (1H ຄື່ອ FH) ແລະ ຈະຕ້ອງໄມ້ມີແລວທີ່ມີຄ່າເປັນຄ່າເລີບສູນຍີ ເພຣະຈະໄມ້ວິຮູອຜູ້ລົຍ

ດັ່ງນີ້ຕາມຕ້ອຍ່າງເມື່ອສອດບັດທີ່ປະຕູ ວົງຈະໃຫ້ອຳນວຍແນບຈະໃຫ້ Output ເປັນ Logic 1 ເພື່ອເປົ້າໂປ່ດລື້ອກປະຕູເມື່ອມັນທຽບຈັບໄດ້ວ່າ Input Sequence ທີ່ປຶ້ອນເຂົ້າມາໃນງາງຈົກເປັນ 3 , 7 , 12 ແລະ ຈະຄົງສກາງໃຫ້ Output
ເປັນ Logic 1 ຕລອດຈົນກວ່າຈະມີກາຮືບປະຕູ ຜົ່ງຈະໃຫ້ເກີດກາຮືບ Clear ໃຫ້ກັບໄປສູ່ສກາງເຮີມຕົ້ນໃຫ້ Output
ເປັນ Logic 0 ຄື່ອທຳກາຮືບປະຕູ

ตำแหน่งขาไอซีทั้งหมดที่ใช้งานในวงจร แสดงได้ดังรูป โดยมี Input ที่ได้จาก Key Card จำนวน 4 เส้น คือ ขา A, B, C, D



รูป แสดงตำแหน่งขาไอซีทั้งหมดที่ใช้งาน

ข้อมูลที่ป้อนเข้ามาทางอินพุตของไอซี มีทั้งหมด 4 เส้น แต่จะมีค่าข้อมูลเพียง 3 ค่า คือ อาคาร , ชั้น , ห้อง ดังนั้นวงจรต้องจำสถานะทั้ง 3 รวมทั้งสถานะเริ่มต้นทั้งหมดให้เป็น 4 สถานะ จึงใช้ Memory ทั้งหมด 2 ตัว ในที่นี้จะใช้ D Flip-Flop ที่อยู่ในไอซี โดยที่หน่วยความจำที่เก็บจะกำหนดให้เป็น State ตัวเดียว S0 ถึง S3 ซึ่งจะมีเส้นสัญญาณทั้งหมด 2 เส้น กำหนดให้เป็น Q_1 และ Q_0 ส่วนขาเอาท์พุต มี 1 เส้น กำหนดให้เป็น Z

เมื่อแทนค่าต่างๆ จากข้อกำหนดของวงจร Synchronous Sequential ดังกล่าว สามารถนำไปหาลอจิกฟังก์ชันได้ โดยกำหนดเงื่อนไขการเปลี่ยนสถานะตามค่า Input 3 ค่าที่กำหนดจากหมายเลขห้อง ตามตัวอย่างสมมุติ เป็นบัตร 3712 คือ อาคาร 3 ชั้น 7 ห้อง 12 นำไปเขียนเป็น PLD file ชื่อ KeyCard.PLD ได้ดังนี้

```

Name      KeyCard;
PartNo   02;
Date     1/6/22;
Revision 01;
Designer Engineer;
Company  KMITL;
Assembly None;
Location None;
Device   g16v8;

/** Input Pins */
PIN 1 = CLK;          /* register clock */
PIN 2 = CLR;          /* clear input */
PIN 3 = A;             /* input A */
PIN 4 = B;             /* input B */
PIN 5 = C;             /* input C */
PIN 6 = D;             /* input D */
PIN 11 = !OE;          /* register output enable */

/** Output Pins */
PIN [16..17] = [Q1..0]; /* register 2-bit */
PIN 18 = Z;            /* output Z */

/** Declarations and Intermediate Variable Definitions */
field KEY = [CLR,A,B,C,D]; /* declare KEY control field */
D0 = KEY:0;           /* define input 0H */
D1 = KEY:1;           /* define input 1H */
D2 = KEY:2;           /* define input 2H */
D3 = KEY:3;           /* define input 3H */
D4 = KEY:4;           /* define input 4H */
D5 = KEY:5;           /* define input 5H */
D6 = KEY:6;           /* define input 6H */
D7 = KEY:7;           /* define input 7H */
D8 = KEY:8;           /* define input 8H */
D9 = KEY:9;           /* define input 9H */
D10 = KEY:A;          /* define input AH */
D11 = KEY:B;          /* define input BH */
D12 = KEY:C;          /* define input CH */
D13 = KEY:D;          /* define input DH */
D14 = KEY:E;          /* define input EH */
D15 = KEY:F;          /* define input FH */
CLEAR = KEY:[10..1F]; /* define clear input */

```

```

/** Logic Equations **/

field STATE = [Q1..0];      /* declare state bit field */
#define S0 'b'00               /* define state */
#define S1 'b'01
#define S2 'b'11
#define S3 'b'10

Sequenced STATE {

present S0      if D3      next S1;      /* building */
                next S0;

present S1      if CLEAR  next S0;
if D7          next S2;      /* floor */
next S0;

present S2      if CLEAR  next S0;
if D12         next S3;      /* room */
next S0;

present S3      if CLEAR  next S0;
if !CLEAR     next S3;
out Z;          /* output Z */

}

```

ข้อมูลใน Key Card ของแต่ละห้องจะมีรหัสไม่เหมือนกัน ดังนั้นในห้องที่ต่างกันให้แก้ไขเฉพาะในส่วนของ Sequenced STATE เท่านั้น โดยการเปลี่ยนเงื่อนไขในส่วนของ Sequenced STATE ที่สร้างขึ้นนี้ อาจจะเปลี่ยนได้หลายแบบ ซึ่งยังคงได้ผลลัพธ์การทำงานของวงจรเหมือนเดิม เช่น

```

Sequenced STATE {

present S0      if D3      next S1;      /* building */
if !D3        next S0;

present S1      if D7      next S2;      /* floor */
if !D7        next S0;

present S2      if D12     next S3;      /* room */
if !D12       next S0;

present S3      if CLEAR   next S0;
if !CLEAR     next S3;
out Z;          /* output Z */

}

```

การทดสอบการทำงานของไอซี PLD ที่ออกแบบไว้ว่าสามารถทำงานได้ถูกต้องหรือไม่ ทำได้โดยใช้ Simulation file ที่เขียนขึ้นชื่อ KeyCard.SI กำหนดให้มี Input เริ่มที่ Vector 1 เป็นการป้อนสัญญาณ !OE = 1 ทำให้ออกที่พุทธ Q ทั้ง 2 เส้นเป็น Hi Impedance และที่ Vector 2 ป้อน CLR = 1 จะทำให้เป็นสถานะเริ่มต้นของวงจร ได้ออกที่พุทธของหน่วยความจำทั้งหมดเป็น 0 จากนั้นตั้งแต่ Vector ที่ 3 จึงเริ่มทดสอบป้อนสัญญาณอินพุทได้ดังนี้

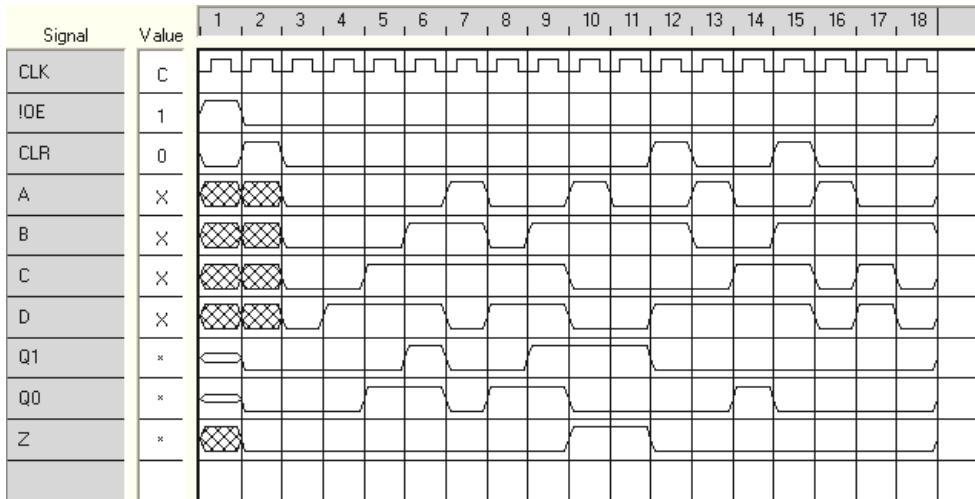
ORDER: CLK, !OE, CLR, A, B, C, D, Q1, Q0, Z;

```

VECTORS:
C10XXXX***      /* test HI Impedance      */
C01XXXX***      /* clear to state 0      */
C000000***      /* input 0                  */
C000001***      /* input 1                  */
C000011***      /* input 3 to state 1      */
C000111***      /* input 7 to state 2      */
C001110***      /* input 14                 */
C000011***      /* input 3 to state 1      */
C000111***      /* input 7 to state 2      */
C001100***      /* input 12 to state 3      */
C000100***      /* input 4                  */
C010101***      /* input 5, clear to state 0 */
C001001***      /* input 9                  */
C000011***      /* input 3 to state 1      */
C010111***      /* input 7, clear to state 0 */
C001100***      /* input 12                 */
C000111***      /* input 7                  */
C000100***      /* input 8                  */

```

การทดสอบสัญญาณจะได้ผลลัพธ์เป็นเอาท์พุตของหน่วยความจำ คือ Q_0 ถึง Q_1 และเอาท์พุตของวงจร Z ตามลำดับค่าอินพุตที่ป้อนเข้ามา แสดงเป็น Timing diagram ดังรูป



รูป แสดง Timing diagram

คำถาม

- 1 จากตัวอย่างของวงจรที่ออกแบบ ให้จากการทดลองที่ 2 ให้เขียน Logic Equation คือสมการของ D-flipflop ขา Q_0 ถึง Q_1 และ Output Z

$$Q0.d = \underline{\hspace{10cm}}$$

$$\underline{\hspace{10cm}}$$

$$\underline{\hspace{10cm}}$$

$$Q1.d = \underline{\hspace{10cm}}$$

$$\underline{\hspace{10cm}}$$

$$\underline{\hspace{10cm}}$$

$$Z = \underline{\hspace{10cm}}$$

- 2 ทดลองออกแบบ PLD file โดยออกแบบวงจร Sequence Detector ที่ใช้เป็น Key Card ชนิดบัตรเจาะรู ให้ใช้รหัสของบัตรเป็นเลขฐาน 10 จำนวน 3 หลักต่อนักศึกษาหนึ่งคน ซึ่งกำหนดจากรหัสประจำตัว 3 หลักหลังของนักศึกษา ถ้ามีหลักที่เป็นเลข 0 ($D0 = KEY:0$) ให้สมมติเป็น 10 ($D10 = KEY:A$)
- 3 ให้วาดรูป Timing diagram ที่ได้จากการทดสอบสัญญาณ Simulation file ว่าผลลัพธ์สามารถทำงานได้ถูกต้องตามที่ออกแบบไว้ โดยสมมติ Input มาอย่างน้อย 18 ค่า
- 4 ให้วาดรูปวงจรทั้งหมดที่ใช้งานของไอซีที่โปรแกรมเสร็จ

การทดลองของจรรับส่งและเข้ารหัสดิจิตอล

วัตถุประสงค์

1. เพื่อศึกษาวิธีการทำงานของวงจรเข้ารหัส และวงจรดอครหัส
2. เพื่อให้เข้าใจหลักการออกแบบและทดสอบการทำงานของวงจรดิจิตอลได้
3. เพื่อให้ตรวจสอบและแก้ไขความผิดพลาดที่เกิดขึ้นจากการสื่อสารข้อมูล
4. เพื่อให้สามารถตรวจสอบสัญญาณและวิเคราะห์ทางจรที่อยู่ภายในไอซีหรืออุปกรณ์ต่างๆ

อุปกรณ์ที่ใช้ในการทดลอง

1. ชุดทดลองของจรรบดิจิตอล
2. ไอซี ดิจิตอล
3. เครื่องคอมพิวเตอร์
4. โปรแกรมคอมไฟเลอร์
5. เครื่องโปรแกรมไอซี

การส่งสัญญาณในระบบดิจิตอลอาจจะเกิดข้อผิดพลาดขึ้นได้ ความผิดพลาดของข้อมูลต่างๆ อาจจะเกิดจากกระดับของสัญญาณที่ไม่ถูกต้อง สัญญาณอ่อน มีสัญญาณรบกวน (Noise) หรือความบกพร่องของสื่อที่ใช้ในการจัดเก็บข้อมูล เช่น งานแม่เหล็ก ชั้งร้อนบางชนิดสามารถตรวจสอบความผิดพลาดได้ การตรวจสอบความผิดพลาดแบบง่ายโดยการใส่บิทตรวจสอบ (Parity Bit) เพิ่มเข้าไปในการรับส่งสัญญาณดิจิตอล เพื่อเพิ่มความถูกต้องในการรับส่งข้อมูลให้ดีขึ้น พาริตี้เป็นบิทที่เพิ่มเติมลงไปในข้อมูล (Code Word) เพื่อทำให้ข้อมูลใหม่ที่ได้นั้นมีจำนวนของเลข 1 เป็นจำนวนคู่ (Even Parity) หรือ จำนวนคี่ (Odd Parity) ดังตารางที่ 1

ตารางที่ 1 แสดง Parity ที่ใช้ใน BCD-8421 code

Decimal	BCD-8421	Even Parity	Odd Parity
0	0000	0	1
1	0001	1	0
2	0010	1	0
3	0011	0	1
4	0100	1	0
5	0101	0	1
6	0110	0	1
7	0111	1	0
8	1000	1	0
9	1001	0	1

รหัสชนิดแก้ไขความผิดพลาดในตัวเอง (Error-Correcting Codes) รหัสแบบนี้เป็นรหัสที่มีความแม่นยำสูงมาก เพราะผู้รับสามารถตรวจหาความผิดพลาดของข้อมูล (Error Detection) และแก้ไขข้อมูลที่ผิดพลาดให้ถูกต้องค้ายังตัวเองได้ทันที รหัสแบบนี้จะนำไปใช้ในที่ต้องการความรวดเร็วและความถูกต้องของข้อมูลสูง รหัสกลุ่มนี้มีหลายแบบ แบบง่ายที่สุดคือ รหัสแฮมมิง (Hamming Code)

Hamming Code เป็นการเข้ารหัสที่สามารถตรวจจับและแก้ไขความผิดพลาดได้ โดยวิธีแก้ไขความผิดพลาดได้นี้จะทำได้เมื่อพิดไป 1 บิตต่อข้อมูล 1 Frame เท่านั้น การเข้ารหัสแบบนี้ต้องเพิ่มข้อมูลของ Parity Bit เข้าไป โดยจำนวนของ Parity Bit (P) ที่จะเพิ่มเข้าไปจะเป็นจำนวนเต็มบวกค่าดังนั้นแต่ 2 ขึ้นไป ถ้าให้ข้อมูลมีจำนวนบิตเท่ากับ D จะได้ความสัมพันธ์ของการเข้ารหัสดังนี้

$$D = 2^P - P - 1$$

สมมุติข้อมูล 1 Codeword มีขนาด 4 บิต จะมีบิตรตรวจสอบทั้งหมด 3 บิต ทำให้ Frame ข้อมูลที่ส่งขนาดจะรวมเป็น 7 บิต กำหนดให้บิตที่มีนัยสำคัญต่ำสุดอยู่ทางขวาสุดเป็นบิตที่ 1 และบิตที่มีนัยสำคัญสูงสุดทางซ้ายสุดเป็นบิตที่ 7 โดยตำแหน่งที่กำหนดให้ใช้เป็นบิตรับตรวจสอบ Parity ของข้อมูลที่ตำแหน่งต่างๆ จะเป็นบิทยอยตำแหน่งที่ 2^n คือ $2^0, 2^1, 2^2, \dots$ ซึ่งได้แก่บิตที่ 1, 2, 4, ... ส่วนตำแหน่งที่เหลือจะเป็นบิทข้อมูล หากกำหนดให้ D แทนบิทข้อมูล และ P แทนบิตรตรวจสอบ (Parity) รูปแบบโครงสร้างของ Frame ในการส่งรหัสแฮมมิงจะเป็นดังนี้

	D_7	D_6	D_5	P_4	D_3	P_2	P_1
--	-------	-------	-------	-------	-------	-------	-------

- โดยที่ D เป็นบิทข้อมูลขนาด 4 บิต
 P_1 เป็นบิทเพื่อให้เกิด Parity คู่ใน Frame อยู่ตำแหน่งบิตที่ 7, 5, 3, 1 (D_7, D_5, D_3, P_1)
 P_2 เป็นบิทเพื่อให้เกิด Parity คู่ใน Frame อยู่ตำแหน่งบิตที่ 7, 6, 3, 2 (D_7, D_6, D_3, P_2)
 P_4 เป็นบิทเพื่อให้เกิด Parity คู่ใน Frame อยู่ตำแหน่งบิตที่ 7, 6, 5, 4 (D_7, D_6, D_5, P_4)

ค่าตำแหน่งบิทของข้อมูลที่ใช้ในการตรวจสอบ Parity จะเลือกเฉพาะตำแหน่งของบิตที่มีค่าเป็น 1 เท่านั้น ดังแสดงในตารางที่ 2

ตารางที่ 2 แสดงตำแหน่งที่ใช้ในการตรวจสอบ Parity

ตำแหน่งบิทของ Frame	Binary			
	Decimal	P_4	P_2	P_1
1		0	0	1
2		0	1	0
3		0	1	1
4		1	0	0
5		1	0	1
6		1	1	0
7		1	1	1

ถ้าต้องการส่งข้อมูล BCD-8421 code ขนาด 4 บิต ต้องเริ่มจากการกระจายข้อมูล BCD-8421 เพื่อจัดเก็บใน Frame ข้อมูลขนาด 7 บิต แล้วพิจารณาเฉพาะตำแหน่งของบิตที่มีค่าเป็น 1 และทำการแปลงตำแหน่งนั้นเป็นเลขฐานสองขนาด 3 บิต เช่น เลข 7 จะได้ตำแหน่งที่ 6, 5, 3 เลข 9 จะได้ตำแหน่งที่ 7, 3 และทำการ Exclusive OR (\oplus) เข้าด้วยกัน จะได้บิต parity ดังตัวอย่าง

เลข 7		เลข 9	
ตำแหน่งบิท (ฐานสิบ)	ตำแหน่งบิท (ฐานสอง)	ตำแหน่งบิท (ฐานสิบ)	ตำแหน่งบิท (ฐานสอง)
6	=	1 1 0	
5	=	1 0 1	⊕
3	=	<u>0 1 1</u>	
บิทตรวจสอบ	=	0 0 0	
			บิทตรวจสอบ = <u>1 0 0</u>

นำบิท Parity ทั้งหมดบรรจุลงใน Frame ตามลำดับ จะได้ผลลัพธ์ของการเข้ารหัส Hamming สำหรับ BCD-8421 code ที่ใช้ในการส่ง ได้ดังแสดงในตารางที่ 3

ตารางที่ 3 แสดงการใช้รหัสแซมมิ่งสำหรับ BCD-8421 code

Decimal	BCD-8421	Hamming Code
	D ₇ D ₆ D ₅ P₄ D ₃ P₂ P ₁	D ₇ D ₆ D ₅ P₄ D ₃ P₂ P ₁
0	0 0 0 <u> </u> 0 <u> </u> <u> </u>	0 0 0 0 0 0 0
1	0 0 0 <u> </u> 1 <u> </u> <u> </u>	0 0 0 0 1 1 1
2	0 0 1 <u> </u> 0 <u> </u> <u> </u>	0 0 1 1 0 0 1
3	0 0 1 <u> </u> 1 <u> </u> <u> </u>	0 0 1 1 1 1 0
4	0 1 0 <u> </u> 0 <u> </u> <u> </u>	0 1 0 1 0 1 0
5	0 1 0 <u> </u> 1 <u> </u> <u> </u>	0 1 0 1 1 0 1
6	0 1 1 <u> </u> 0 <u> </u> <u> </u>	0 1 1 0 0 1 1
7	0 1 1 <u> </u> 1 <u> </u> <u> </u>	0 1 1 0 1 0 0
8	1 0 0 <u> </u> 0 <u> </u> <u> </u>	1 0 0 1 0 1 1
9	1 0 0 <u> </u> 1 <u> </u> <u> </u>	1 0 0 1 1 0 0

ตัวอย่างการรับส่งข้อมูล ถ้าต้องการส่ง BCD-8421 ที่มีค่าเท่ากับเลข 9 ด้วยรหัสແຍມມິງ 1 Frame ขนาด 7 ບົກ ດາວໂຫຼດສັບສົນໃຫຍ່ຈະເປັນ 1001100 ເນື້ອປລາຍທາງຮັບໄປແລະມີສົງຄູານຮັບກວນຈຸນເກີດຄວາມຜິດພາດຂອງ
ข้อมูล ສມມຸດຕິໄດ້ເປັນຄ່າ 1001000 ດ້ວຍເຫັນວ່າຈະເປັນ BCD ຈະໄດ້ຄ່າເປັນ 1000 ທີ່ເປັນເລີກ 8 ແຕ່ຖ້າ
ตรวจสอบບົກພາຣີຕີ່ດ້ວຍແລ້ວຈະເຫັນໄດ້ວ່າເກີດຄວາມຜິດພາດເປັນຮັບສິດທີ່ໄມ່ຄຸກຕ້ອງດັ່ງນີ້

บิท ข้อมูลที่ได้	D ₇	D ₆	D ₅	P ₄	D ₃	P ₂	P ₁	BCD เป็นเลข 8
P4 บิทที่ 7, 6, 5, 4 ได้	1	0	0	1	0	0	0	ค่า Parity ที่ได้ถูกต้อง
P2 บิทที่ 7, 6, 3, 2 ได้	1	0	0	1		0	0	ค่า Parity ที่ได้ผิด
P1 บิทที่ 7, 5, 3, 1 ได้	1		0		0		0	ค่า Parity ที่ได้ผิด
ข้อมูลเฉพาะที่ถูกต้องคือ	1	0	0					
ข้อมูลที่ผิดพลาดคือ						0		บิทที่ 3
ข้อมูลที่แก้ไขให้ถูกต้องแล้วคือ	1	0	0		1			BCD เป็นเลข 9

การตรวจสอบความถูกต้องของข้อมูลทำได้โดยการนำตำแหน่งของบิทที่มีค่าเป็น 1 ทั้งหมด (บิทข้อมูล และบิทพาริตี้) แปลงเป็นเลขฐานสองขนาด 3 บิท และทำการ Exclusive OR (\oplus) เข้าด้วยกัน หากได้ค่าของบิทตรวจสอบทั้งหมดมีค่าเป็น 0 แสดงว่าข้อมูลที่ได้รับถูกต้อง แต่ถ้าได้ค่าของบิทตรวจสอบทั้งหมดมีค่าไม่เป็น 0 แสดงว่าข้อมูลที่ได้รับไม่ถูกต้อง คือ มีบิทใดบิทหนึ่งหรือหลายบิทใน Frame ข้อมูลเกิดความผิดพลาด ดังเช่น

เลข 9

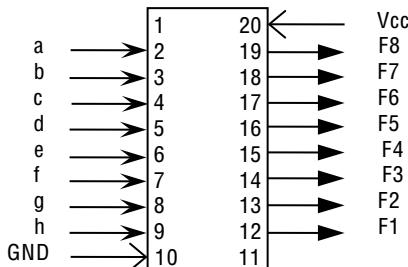
ตำแหน่งบิท (ฐานสิบ)	ตำแหน่งบิท (ฐานสอง)
7	= 1 1 1
4	= 1 0 0
3	= <u>0 1 1</u>
บิทตรวจสอบ	= 0 0 0 (ข้อมูลที่ได้รับถูกต้อง)

เลข 8

ตำแหน่งบิท (ฐานสิบ)	ตำแหน่งบิท (ฐานสอง)
7	= 1 1 1
4	= <u>1 0 0</u>
บิทตรวจสอบ	= 0 1 1 (ข้อมูลที่ได้รับไม่ถูกต้อง)

จากตัวอย่าง ข้อมูลมีความผิดพลาดจากเลข 9 เป็นเลข 8 จึงทำให้บิทตรวจสอบมีค่าไม่เป็น 0 ทั้งหมด เราสามารถแก้ไขความผิดพลาดของข้อมูลได้ โดยทำการแปลงบิทตรวจสอบจากเลขฐานสองคือ 011 เป็นเลขฐานสิบ ได้เท่ากับ 3 ทำให้รู้ว่าบิทที่เกิดความผิดพลาดคือตำแหน่งบิทที่ 3 แล้วให้ทำการเปลี่ยนค่าบิทที่ 3 จาก **0** เป็น **1** ทำให้ได้ข้อมูลที่ถูกต้องคือ 100**1** ซึ่งก็คือเลข 9

การทดลองจะใช้ไอซี 16V8 ให้นำไฟล์ JED ตามหมายเลขอุปกรณ์และทำการโปรแกรมข้อมูลเข้าที่ตัวไอซี โดยที่ตำแหน่งขาไอซีทั้งหมดที่ใช้ในการทดลอง รวมทั้งขาของไฟที่ใช้ป้อนให้วงจรทำงาน แสดงได้ดังรูปที่ 1 ไอซีที่โปรแกรมแล้วนี้จะมี Input จำนวน 8 เส้น คือ ขา a , b , c , d , e , f , g , h และมี Output ที่ได้จากไอซีจำนวน 8 เส้น คือ ขา F1 , F2 , F3 , F4 , F5 , F6 , F7 , F8



รูปที่ 1 แสดงตำแหน่งขาไอซีทั้งหมดที่ใช้งาน

1. การทดลองป้อนข้อมูลเข้าที่อินพุทของไอซีเติมค่าในตารางที่ 4 และบันทึกผลของสัญญาณเอาท์พุทที่วัดได้ในแต่ละขาลงในตาราง

ตารางที่ 4 ผลการทดสอบ

ค่าที่ป้อนเข้า			ค่าที่วัดได้	
a	b	c	F2	F3
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

2. ให้ใช้กฎของ Boolean ทำการหาผลลัพธ์ชั้น และเขียนผลลัพธ์ในแต่ละส่วนของ Logic Equation คือ สมการของเอาท์พุทขา F2 และ F3

F2 =

F3 =

3. ให้ว่าครูปวงจรล้อจิกที่ได้จากการทดสอบคือสมการ F2 และ F3

4. การทดลองป้อนข้อมูลเข้าที่ไอซีตามค่าในตารางที่ 5 และบันทึกผลลัพธ์โดยจิ๊กที่วัดได้ในแต่ละขา เอ้าที่พุทลงในตาราง

ตารางที่ 5 ผลการทดสอบ

ค่าที่ป้อนเข้า				ค่าที่รับได้			
a	b	c	d	F1	F2	F3	F4
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				

5. ให้ทำการหาอจิกฟังก์ชัน โดยใช้ K-Map และเบี่ยงฟังก์ชันการทำงานของจร

.....

.....

.....

.....

.....

.....

6. ให้อธิบายว่า วงศ์ไนซ์ทำหน้าที่แปลงรหัสแบบใด

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

7. วิธีที่ออกแบบไวน์สามารถป้อนอินพุตเพื่อให้วงจรทำงานเป็นเลข Binary จาก 0 ถึง 15 ได้หรือไม่ ถ้าไม่ได้ต้องแก้ไขวงจรอย่างไร และให้หาผลของผลลัพธ์ที่แก้ไขแล้ว

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

8. ให้วาดรูปวงจรทึ้งหมกที่ใช้งานได้หลังจากการแก้ไขแล้ว

9. การทดลองป้อนข้อมูลเข้าที่ไอซีตามค่าในตารางที่ 6 และบันทึกผลสัญญาณค์อิกที่วัดได้ในแต่ละขา เอาท์พุตลงในตาราง

ตารางที่ 6 ผลการทดลอง

ค่าที่ป้อนเข้า				ค่าที่วัดได้		
d	c	b	a	F5	F6	F7
0	0	0	0			
0	0	0	1			
0	0	1	0			
0	0	1	1			
0	1	0	0			
0	1	0	1			
0	1	1	0			
0	1	1	1			
1	0	0	0			
1	0	0	1			
1	0	1	0			
1	0	1	1			
1	1	0	0			
1	1	0	1			
1	1	1	0			
1	1	1	1			

10. จากค่าที่วัดได้ให้ใช้โปรแกรม Quine-McCluskey ป้อนอินพุตและเอาท์พุตตามที่กำหนด เพื่อหาค่า ลอจิกฟังก์ชัน และให้เปลี่ยนผลลัพธ์ที่ได้
-
-
-
-
-
-

11. จากผลลัพธ์ที่ได้จากการทดลองให้อธิบายว่าจะกรณีทำหน้าที่อะไร โดยอินพุตที่ป้อนเข้ามาจะต้องมีค่า อะไรบ้าง
-
-

12. ให้วาดรูปวงจรทึ้งหมวดที่จะนำไปใช้งานจริง โดยมีจำนวนสายที่ต่อไปใช้งาน ทางด้าน Input 4 เส้น และทางด้าน Output 7 เส้น

13. จากร่างในข้อ 12 ถ้าส่งข้อมูลออกไปและตรวจสอบทางด้านรับพบว่าเกิดความผิดพลาดของข้อมูล จงยกตัวอย่างว่าจะต้องตรวจสอบและแก้ไขความผิดพลาดที่เกิดขึ้นจากการสื่อสารข้อมูลอย่างไรเพื่อให้ได้ข้อมูลที่ถูกต้อง

14. การทดลอง กรณีว่างที่มีอินพุตหลายเส้น โดยที่เอาท์พุตจะเกิดสภาวะเป็นลอจิกหนึ่ง (Minterm) จำนวนมาก และมีสภาวะเป็น Hi Impedance ร่วมด้วย การทดลองนี้ เป็นการนำเอาวงจรภายในของ ไอซีตัวหนึ่งซึ่งใช้ในระบบสื่อสารทางด้านดิจิตอล โดยมีอินพุตเข้าที่ไอซีคือ a, b, c, d, e, f, g, h และมีเอาท์พุตออกมาที่ตำแหน่งงหา F8 ให้ทำการทดสอบสัญญาณอินพุตลอจิกต่างๆ ที่มีผล ต่อสัญญาณเอาท์พุต แล้วบันทึกค่าที่ได้

15. จากค่าที่วัดได้ให้วิเคราะห์ทางจรที่อยู่ภายในไอซ์

16. ให้นำผลลัพธ์ที่ได้จากข้อ 15 ไปเก็บเป็น PLD file โดยให้แก่ไฟล์ตัวอย่างชื่อ test.pld และทำ Simulation file ทดสอบการทำงานของไอซี PLD ที่ออกแบบไว้ว่าสามารถทำงานได้ถูกต้องเมื่อมีนั้นแบบหรือไม่ โดยให้ใช้ไฟล์ test.si ในการทดสอบ