บทที่ ๒

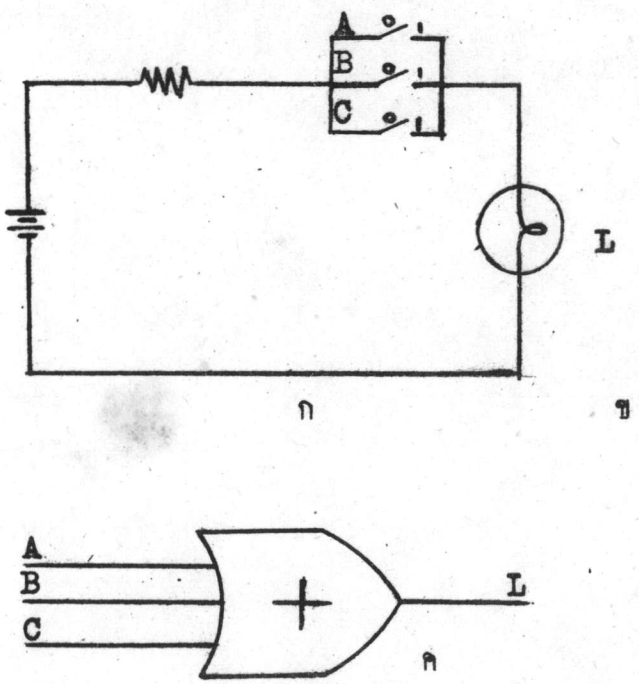
ลอจิกเกทต่าง ๆ

บทนำ

ลอจิกเกท (**Logic gate**) เป็นวงจรพื้นฐานที่สำคัญในเครื่องคอมพิวเตอร์ที่มีความเร็วในการทำงานสูง ลอจิกเกทที่สำคัญ ๆ มีอยู่จำนวนไม่มากนัก เมื่อทำความเข้าใจดีแล้ว ก็ย่อมสามารถจะนำไปใช้ออกแบบสร้างวงจรดิจิตอลต่าง ๆ ได้

OR gate

วงจร **OR gate** นี้มีอินพุท (**input**) ตั้งแต่ ๒ อินพุทขึ้นไปและมีหนึ่งเอาท์พุท (**output**) เอาท์พุทของ **OR gate** จะเป็นสัญญาณไฟสูง (**logic 1**) หรือสัญญาณไฟต่ำ (**logic 0**) เท่านั้น เราสามารถอธิบายการทำงานของ **OR gate** ได้ด้วยวงจรไฟฟ้า และ **Truth table** ได้ดังรูปที่ ๑



ข

A	B	C	L
0	0	0	0
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	1

รูปที่ ๑ วงจร OR gate (ก) วงจรใช้อธิบาย (ข) **Truth Table**
(ค) สัญญลักษณ์

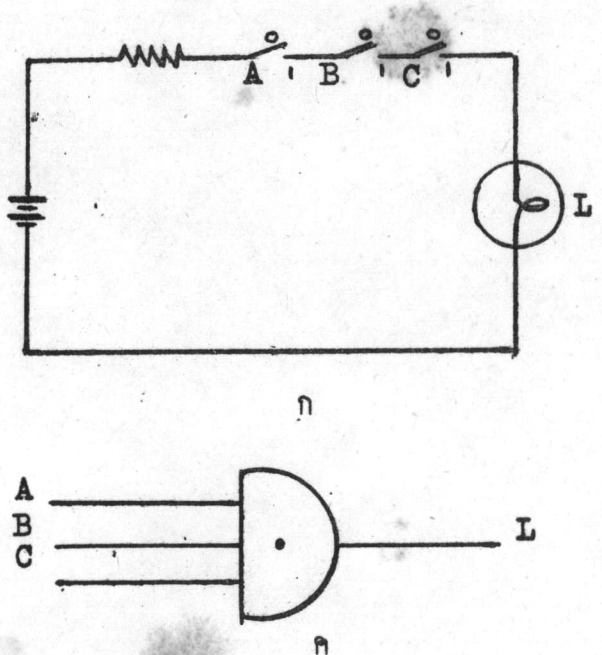
ก. ถ้าอินพุทอันใดอันหนึ่งมีค่าเป็น ๑ แล้ว เอาท์พุทจะเป็น ๑ ด้วย

ข. เมื่ออินพุททุกอันเป็น ๐ แล้ว เอาท์พุทจะเป็น ๐ ด้วย

จากรูปที่ ๑ เมื่อ A หรือ B หรือ C อันใดอันหนึ่งมีค่า ๑ หลอดไฟ L ก็จะติด

AND gate

AND gate นี้มีอินพุทตั้งแต่ ๒ อินพุทขึ้นไป และมีหนึ่งเอาท์พุท เอาท์พุทของ AND gate จะเป็นสัญญาณไฟสูงหรือสัญญาณไฟต่ำเท่านั้น เราสามารถใช้อธิบายการทำงานของ AND gate ได้ด้วยวงจรไฟฟ้า และ Truth Table ดังรูปที่ ๒



ก

ค

A	B	C	L
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

ข

รูปที่ ๒ วงจร AND gate

ก. วงจรใช้อธิบาย

ข. Truth Table

ค. สัญญลักษณ์

ก. ทุกอินพุทเข้ามาต้องมีค่าเป็น ๑ หมด เอาท์พุทจึงจะเป็น ๑

ข. ถ้าอินพุทตัวใดตัวหนึ่งเข้ามามีค่าเป็น ๐ เอาท์พุทจะเป็น ๐

จากรูปที่ ๒ เมื่อ A และB และ C มีค่าเป็น ๑ หลอดไฟ L จะดับ

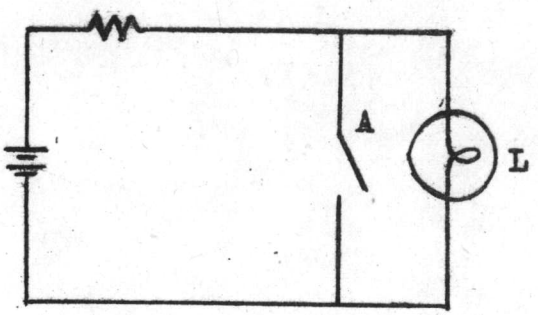
Not gate

Not gate นี้มีหนึ่งอินพุท และหนึ่งเอาท์พุท

ก. ถ้าอินพุทมีค่าเป็น ๑ เอาท์พุทจะมีค่าเป็น ๐

ข. ถ้าอินพุทมีค่าเป็น ๐ เอาท์พุทจะมีค่าเป็น ๑

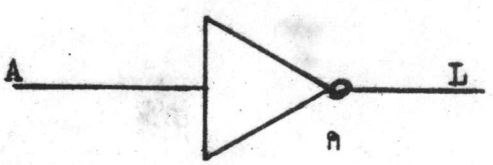
เขียนเป็นวงจรไฟฟ้า และ Truth Table ได้ดังรูปที่ ๓



ก

A	L
0	1
1	0

ข



ค

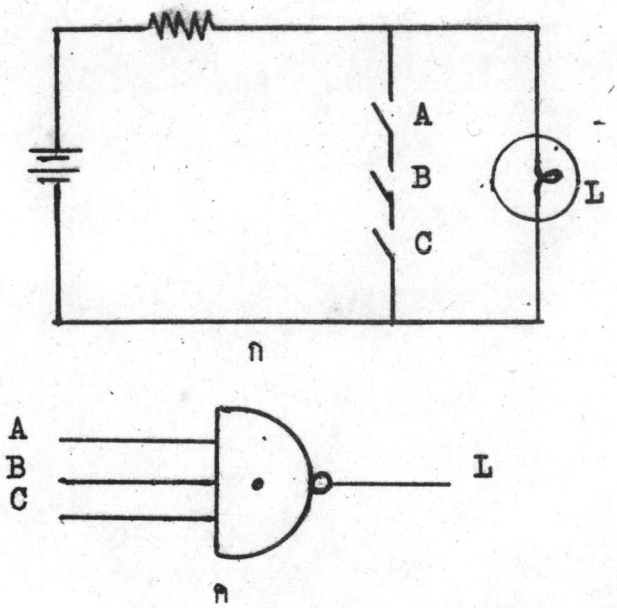
รูปที่ ๓ วงจร Not gate ก. วงจรใช้อธิบาย ข. Truth Table

ค. สัญญลักษณ์

จากรูปที่ ๓ ถ้า A เป็นวงจรปิด หลอดไฟ L จะดับ เพราะการระแสผ่าน A ไปหมด

NAND gate

NAND gate มีลักษณะการทำงานเหมือนกับการเอา Not gate มาต่อ อนุกรมกับเอาท์พุทของ AND gate เราสามารถอธิบายการทำงานของ NAND gate ได้ด้วยวงจรและ Truth Table ดังรูปที่ ๔



A	B	C	L
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	0

ข

รูปที่ ๔ วงจร NAND gate ก. วงจรใช้อธิบาย ข. Truth Table
ค. สัญญลักษณ์

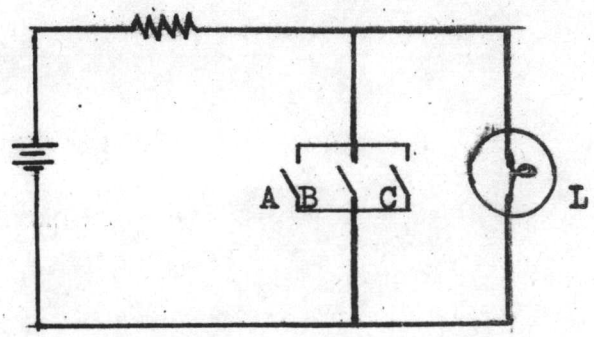
ก. ถ้าทุกอินพุทมีค่าเป็น ๑ เอาท์พุทก็มีค่าเป็น ๐

ข. ถ้าทุกอินพุท หรืออินพุทอันใดอันหนึ่งมีค่าเป็น ๐ เอาท์พุทจะมีค่าเป็น ๑

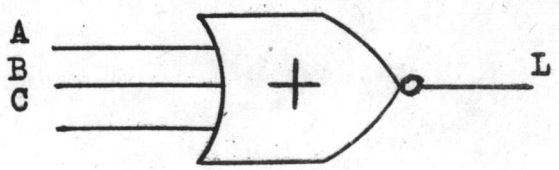
จากรูปที่ ๔ เมื่อ A และB และ C มีค่าเป็น ๑ หมด หลอดไฟ L ก็จะดับ

NOR gate

NOR gate มีลักษณะการทำงานเหมือนกับการเอา NOT gate มาต่ออนุกรมกับเอาท์พุทของ OR gate เราสามารถอธิบายการทำงานของ NOR gate ได้ด้วยวงจรและ Truth Table ได้ดังรูปที่ ๕



ก



ค

A	B	C	L
0	0	0	1
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	0

ข

รูปที่ ๕ วงจร NOR gate ก. วงจรใช้อธิบาย ข. Truth Table
ค. สัญญลักษณ์

ก. ถ้าอินพุทอันใดอันหนึ่งมีค่าเป็น ๑ เอาท์พุทก็มีค่า ๐

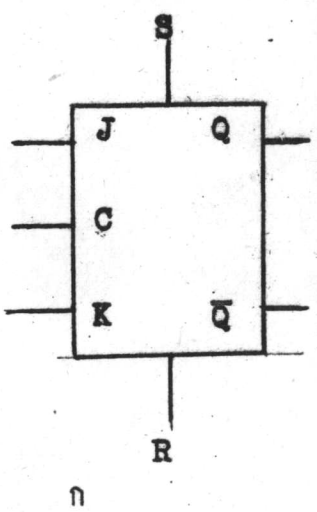
ข. ถ้าทุกอินพุทมีค่าเป็น ๐ เอาท์พุทจะมีค่าเป็น ๑

จากรูปที่ ๕ เมื่อ A และ B และ C มีค่าเป็น ๐ หลอดไฟ L จะติด

J-K Flip-Flop

อุปกรณ์ลอจิกเกทต่าง ๆ ที่ได้กล่าวมาแล้วข้างต้นล้วนแต่ต้องใช้ระดับอินพุทที่มีระดับคงที่สำหรับป้อนเข้ามาทางอินพุท เมื่อระดับอินพุทเปลี่ยนไป บรรดาเกทต่าง ๆ ก็ไม่สามารถที่จะจำอินพุทอันเก่าที่เคยใช้อยู่ได้ ทั้งนี้เพราะเกทต่าง ๆ ที่ได้กล่าวมานั้นไม่มีความจำของตัวเองโดยเฉพาะ เช่น AND gate เมื่อป้อนสัญญาณไฟสูงให้อินพุททุกขาจะได้เอาท์พุทออกมา แต่ถ้าเปลี่ยนสัญญาณไฟต่ำเข้าแทนเอาท์พุทจะเปลี่ยนทันที ดังนั้นระดับอินพุทไม่คงที่จะทำให้เอาท์พุทเกิดการเปลี่ยนแปลงได้ แต่มีอุปกรณ์ทางวงจรไฟฟ้าที่สามารถคงสถาพเดิมของตัวมันเองได้ ถึงแม้ว่าทางอินพุทจะหายไปหรือเปลี่ยนไป อุปกรณ์ชนิดนี้ที่สามารถเก็บความจำได้ คือ ฟลิพฟลอพ (flip-flop) ซึ่งสามารถอธิบายให้เข้าใจได้ง่าย ๆ โดยการเปรียบเทียบกับการปิดเปิดไฟฟ้า เมื่อฟลิพฟลอพได้รับคำสั่งให้อยู่ในภาวะ ๑ นั้น ก็จะอยู่ในภาวะนั้น จนได้รับคำสั่งให้เปลี่ยนจึงจะเปลี่ยนได้

ฟลิพฟลอพมีอยู่หลายชนิด ฟลิพฟลอพที่นำมาใช้งานมากคือ J-K Flip-Flop มีสัญญลักษณ์และ Truth Table ดังรูปที่ ๖



ก

J	K	After Pulse C
0	0	คงเดิม
0	1	0
1	0	1
1	1	เปลี่ยนตรงกันข้าม

ข

R	S	Q	Q̄
0	1	1	0
1	0	0	1

ค

รูปที่ ๖ J-K Flipflop ก. สัญญลักษณ์ ข. Truth Table J-K F.F.
ค. Truth Table R-S F.F.

กล่าวคือ มีอินพุทอยู่จำนวน ๕ อันได้แก่ S,R,C,J และ K และมีเอาท์พุทอยู่ ๒ อัน ได้แก่ Q กับ $\bar{Q}$

เมื่อมีสัญญาณไฟสูงมาป้อนให้กับอินพุท S (Set) สัญญาณไฟอยู่ในลักษณะ S= 1, R = 0 แล้ว เอาท์พุท Q = 1, $\bar{Q}$ = 0

เมื่อมีสัญญาณไฟสูงมาป้อนให้กับอินพุท R (Reset) สัญญาณไฟอยู่ในลักษณะ S = 0 , R = 1 แล้ว เอาท์พุท Q = 0 , $\bar{Q}$ = 1 ดังจะเห็นได้จากรูปที่ ๖ ค สำหรับอินพุท C ใช้สำหรับป้อนสัญญาณ clock เข้า ซึ่งจะเปลี่ยนสภาพของเอาท์พุท Q ตามสัญญาณไฟเข้าที่ J และ K ในขณะนั้น ดังในรูปที่ ๖ ข มีดังนี้คือ

๑. เมื่อ J= 1 และ K= 1 clock pulseจะทำให้เอาท์พุท Q เปลี่ยนเป็นตรงกันข้าม คือ จากสัญญาณไฟสูงเปลี่ยนเป็นสัญญาณไฟต่ำ หรือจากสัญญาณไฟต่ำเปลี่ยนเป็นสัญญาณไฟสูง

๒. เมื่อ J=1 และ K=0 clock pulseจะทำให้เอาท์พุท Q เป็นสัญญาณไฟสูงเสมอ

๓. เมื่อ J=0 และ K=1 clock pulse จะทำให้เอาท์พุท Q เป็นสัญญาณไฟต่ำเสมอ

๔. เมื่อ J = 0 และ K=0 clock pulse จะไม่ทำให้เอาท์พุท Q แปลี่ยนจากเดิมไปอยู่ในสภาพอื่น สัญญาณไฟสูงก็คงเดิม หรือสัญญาณไฟต่ำก็คงตามเดิม

ส่วนเอาท์พุท Q กับ $\bar{Q}$ นั้น จะมีค่าเป็นตรงกันข้ามกันเสมอ ถ้า Q เป็นสัญญาณไฟสูง $\bar{Q}$ จะเป็นสัญญาณไฟต่ำ หรือ Q เป็นสัญญาณไฟต่ำ $\bar{Q}$ จะเป็นสัญญาณไฟสูง เราสามารถสร้าง J-K Flipflop ขึ้นได้จากลอจิกเกทที่กล่าวมาข้างต้น

หมายเหตุ J-K Flipflop ที่จะใช้ในการวิจัยครั้งนี้นั้นเปลี่ยนแปลงลอจิกของเอาท์พุท Q โดยใช้ clock pulse จะเกิดขึ้นเฉพาะเวลาที่ลอจิกของ clock pulse เปลี่ยนจาก ๑ เป็น ๐ เท่านั้น

Counter

การนับเลขในเครื่องคอมพิวเตอร์มีระบบการนับเลขเป็นระบบไบนารี่ ซึ่งจะใช้เลขระบบ ๒ ตัว คือ ๐ และ ๑ ทั้งนี้เพื่อสะดวกในการแสดงสภาวะสองประการ คือ สวิทช์เปิดหรือปิด ซึ่งทรานซิสเตอร์จะอยู่ในภาวะนำไฟฟ้าหรือไม่นำ วงจรที่ใช้ในการนับคือ Counter ซึ่งโดยมากจะใช้นับเลขระบบไบนารี่ การนับเลขต่าง ๆ เช่น ตัวอย่างที่แสดงไว้ในตารางที่ ๑

เลขฐานสิบ	เลขระบบไบนารี่			
	2^3	2^2	2^1	2^0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	1
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

ตารางที่ ๑ เลขฐานสิบ เปรียบเทียบกับเลขระบบไบนารี่

โดยปรกติเลขฐานสิบที่มาใช้กันอยู่จะแสดงออกเป็นหลักหน่วย หลักสิบ หลักร้อย หลักพัน เลขในระบบไบนารี่นั้นในหลักแรก $= 2^0 = 1$

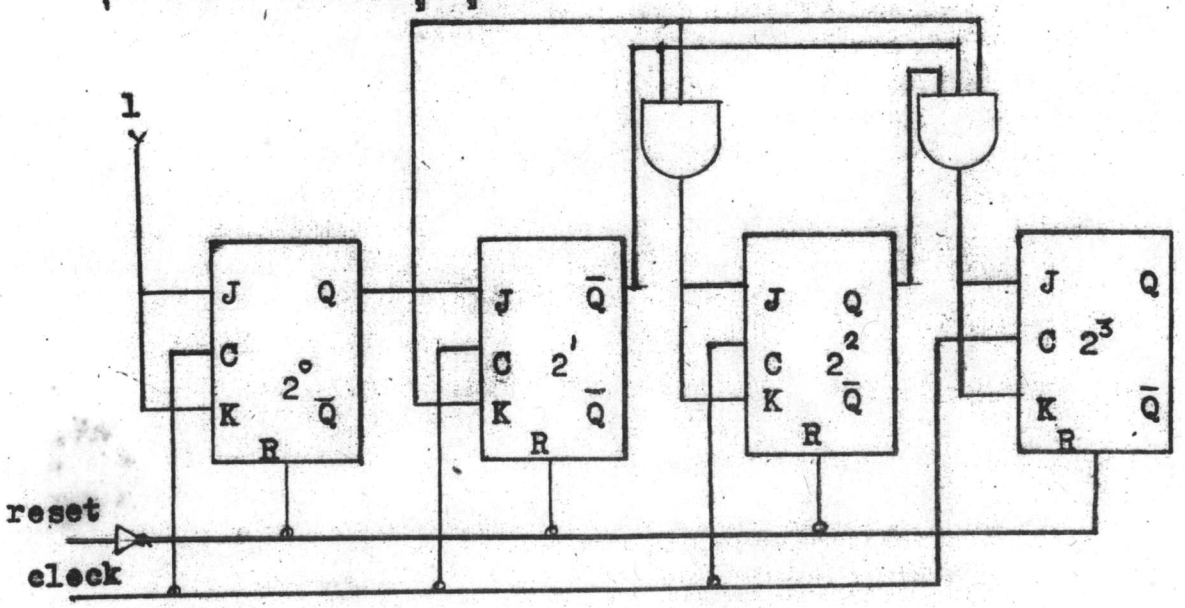
แล้วหลักถัดไป $= 2^1 = 2$

แล้วหลักถัดขึ้นมา $= 2^2 = 4$

เลขหลักถัดขึ้นมา $= 2^3 = 8$ เป็นต้น

ดังนั้นถ้าเลขหลักใดแสดงด้วย ๑ ก็นำเลขหลักนั้นมาบวกกันเข้าจะเท่ากับเลขฐานสิบ เลขหลักใดแสดงด้วย ๐ แสดงว่าเลขหลักนั้นเป็นศูนย์ เช่น จากตารางที่ ๑ เลขฐานสิบเลข ๕ แสดงเป็นเลขไบนารี่ได้ 0101 = ๐+๔+๐+๑ = ๕ การแสดงออกของเลขตัวอื่น ๆ ก็เช่นเดียวกัน

เราสามารถสร้าง Counter ได้จาก J-K Flipflop และลอจิกเกทต่าง ๆ ที่ได้กล่าวมาแล้ว J-K Flipflop สามารถแทนตัวเลขของแต่ละหลักของเลขระบบไบนารี่ได้ เช่น ฟลิพฟลอพอยู่ในภาวะ ๑ เราก็หมายความว่าเลขหลักนั้นเป็น ๑ ถ้าฟลิพฟลอพอยู่ในภาวะ ๐ เราหมายความว่าเลขหลักนั้นเป็น ๐ ดังตัวอย่าง Synchronous up Counter ซึ่งเป็น Counter ชนิดที่ฟลิพฟลอพทุกตัวจะทำงานพร้อมกันอยู่ในรูปที่ ๙



รูปที่ ๙ Synchronous up counter

จากรูปที่ ๙ เอาท์พุทของ

ฟลิพฟลอพ A ใช้แทน 2^0 (ถ้า เอาท์พุทเป็น ๑ เลขหลักนั้นนับเป็น ๑)

ฟลิพฟลอพ B ใช้แทน 2^1

ฟลิพฟลอพ C ใช้แทน 2^2

ฟลิพฟลอพ D ใช้แทน 2^3

การนับของวงจรจะนับได้ดังตารางที่ ๒

clock pulse	Q_D	Q_C	Q_B	Q_A
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
16	0	0	0	0

ตารางที่ ๒ การนับเลขของวงจร **Synchronous up counter**

เราอาจสรุปได้ว่า โดยทั่ว ๆ ไปแล้วในการนับแบบ **Synchronous** นี้ เลขหลักใด ๆ จะเปลี่ยนเป็นค่า **Complement** ของมันก็ต่อเมื่อตัวเลขนั้นที่น้อยกว่าเลขหลักนั้นทุก ๆ ตัวมีสถานะของลอจิกเป็น ๑ หมด ทั้งนี้เพราะใช้ **Clock** อันเดียวกันหมดสำหรับฟลิพฟลอพทุก ๆ ตัว

Module N

วงจรในรูปที่ ๗ จะเรียกไปก็เป็น **Module 16 Counter** เพราะการนับเลขจะเริ่มตั้งแต่ ๐-๑๕ จากนั้นก็เริ่มนับ ๐ ใหม่จนไปถึง ๑๕ (๐-๑๕ , ๐-๑๕) ดังนั้นการนับเลขในวงจรนับแบบ **recycle** ซึ่งเรียกว่า **Module** วงจรสำคัญที่ใช้วิจัยครั้งนี้ ได้แก่

Module 6 นับ 0-5

Module 10 นับ 0-9

การนับของวงจร **Module 6 Counter** แสดงเป็น **Truth Table** ดังตารางที่ ๓

clock	Q_C	Q_B	Q_A
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	0	0	0

ตารางที่ ๓ การนับเลขของ **Module 6**

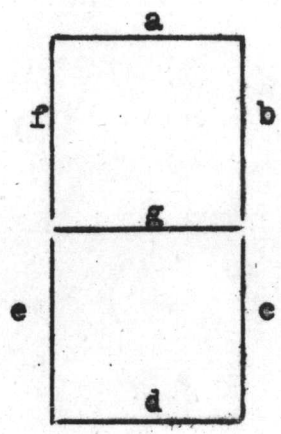
การนับเลขของวงจร Module 10 แสดงเป็น Truth Table ได้ดังตารางที่ ๔

clock	Q_D	Q_C	Q_B	Q_A
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	0	0	0	0

ตารางที่ ๔ การนับเลขของ Module 10

7-Segment display

อุปกรณ์ชิ้นหนึ่งที่จะแสดงข้อมูลของเลข B C D (binary Coded decinal) (ซึ่งหมายถึงเลขในระบบฐานสิบใช้รหัสฐานสอง) ให้อยู่ในตัวเลขอ่านได้ทันทีโดยใช้ 7-Segment display สัญญลักษณ์ของ 7-Segment display แสดงในรูปที่ ๔



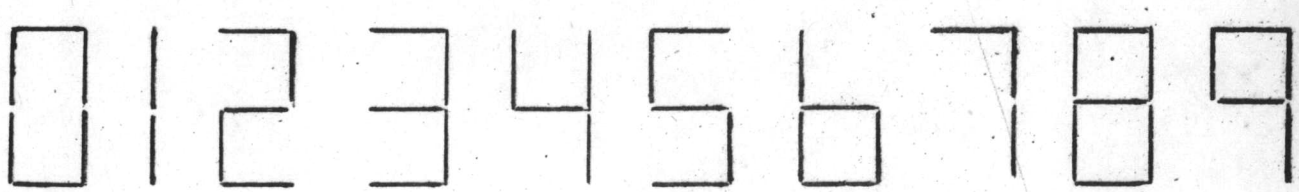
	a	b	c	d	e	f	g
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	0	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	0	0	1	1

1 = Segment on

0 = Segment off

รูปที่ 8 7-Segment display and Truth table

กล่าวคือถ้าให้สัญญาณไฟสูงให้แก่ segment ใด segment นั้นจะติดขึ้นมา ดังนั้นถ้าป้อนสัญญาณไฟสูงให้ segment ตามที่ต้องการให้ติดเป็นเลขตัวใดย่อมที่จะทำได้ เช่น เมื่อต้องการเลข ๐ จะต้องป้อนสัญญาณไฟสูงให้กับ Segment ต่าง ๆ ยกเว้น segment g ดังนั้น ตัวเลขที่ปรากฏออกมาเป็นเลข ๐ เราสามารถแสดงเลขต่าง ๆ ได้ จาก 7-segment display ได้ดังนี้



B C D to 7-Segment Decoder

Decoder เป็นส่วนที่มีความจำเป็นมากในการแปลงข้อมูลให้คอมพิวเตอร์รับเข้าไว้ได้ การแปลงรหัสหนึ่งให้เป็นอีกรหัสหนึ่งจึงเป็นส่วนจำเป็นในคอมพิวเตอร์ การรับรหัสตัวเลขจะรับในระบบไบนารี่ การที่จะแสดงเลขไบนารี่ออกมาให้เป็นตัวเลขนั้น เราสามารถแสดงได้ด้วย 7-Segment display วงจร B C D to 7-Segment Decoder คือ วงจรที่เปลี่ยน B C D ให้มีสัญญาณไฟสูงป้อนให้กับ 7-Segment Decoder แล้วเกิดตัวเลขตามที่ต้องการแสดงตาม Truth Table ในตารางที่ ๕

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

ตารางที่ ๕ Truth Table of B C D to 7-Segment Decoder

จากตารางที่ ๕ เราเขียนแต่ละ Segment อยู่ในรูปของ Boolean Algebra ฟังชั่นได้ดังนี้

$$\bar{a} = A.\bar{B}.\bar{C}.\bar{D} + \bar{A}.C.$$

$$\bar{b} = A.\bar{B}.C. + \bar{A}.B.C.$$

$$\bar{c} = \bar{A}.B.\bar{C}$$

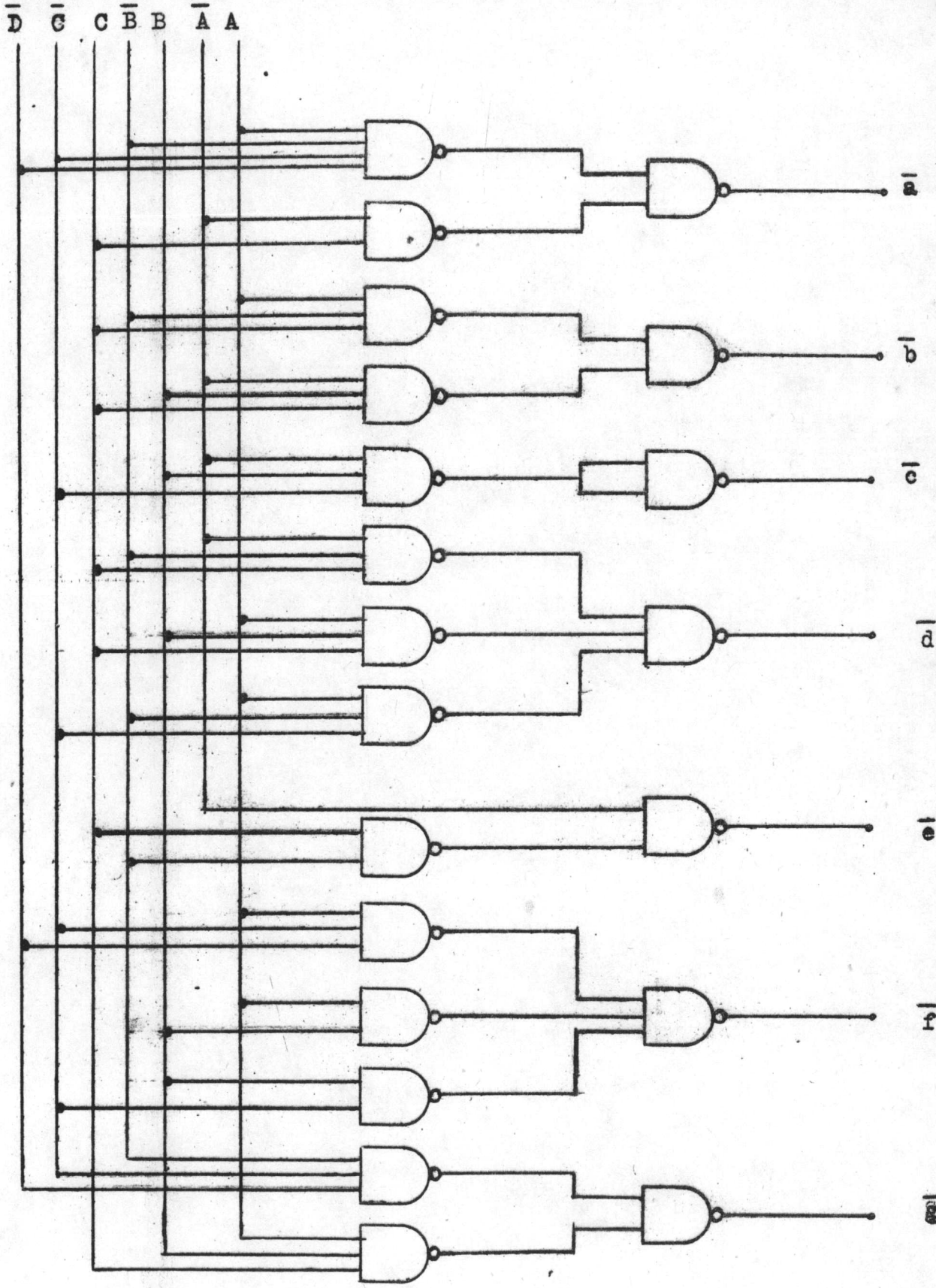
$$\bar{d} = \bar{A}.\bar{B}.C + A.B.C + A.\bar{B}.\bar{C}.$$

$$\bar{e} = \bar{A}.\bar{B}.C$$

$$\bar{f} = A.\bar{D}.\bar{C} + B.A. + B.\bar{C}$$

$$\bar{g} = \bar{B}.\bar{D}.\bar{C} + A.B.C$$

เราสามารถใช้ฟังชั่นเหล่านี้สร้างวงจรขึ้นมาได้ดังแสดงอยู่ในรูปที่ ๙



รูปที่ ๔ วงจร B C D to 7-Segment Decoder