204224 ปฏิบัติการวงจรตรรกะ

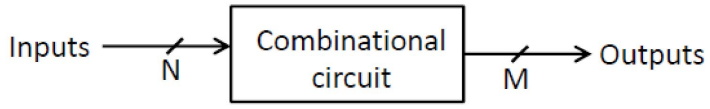
ปฏิบัติการที่ 5 Sequential Logic Design

ทฤษฎีที่ต้องเข้าใจ

1) Sequential logic Circuits

จากที่ผ่านมาในวงแบบ **Combination logic** การเปลี่ยนของ output แปรเปลี่ยนไปตาม input โดยที่ระบบวงจรไม่มีสถานะอะไร การเปลี่ยน output ไม่ได้ขึ้นกับสิ่งเกิดขึ้นในอดีต

- Output is computed when inputs are present
- System has no internal state or no memory
- Nothing computed in the present can depend on what happened in the past!

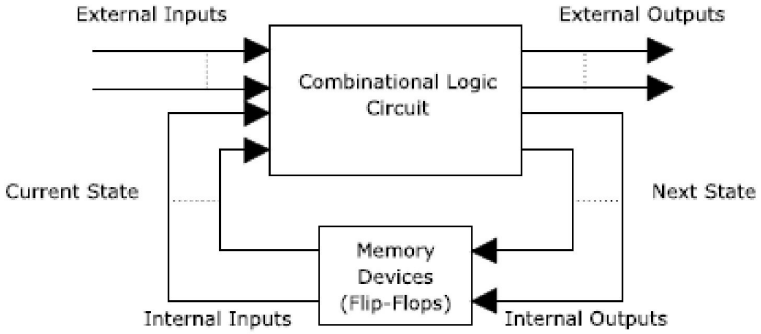


รูปที่ 1 การทำงานของวงแบบ Combination logic ซึ่งการเปลี่ยนของ output แปรเปลี่ยนไปตาม input เท่านั้น

อย่างไงก็ตามระบบดิจิทัลหลายระบบที่ใช้งานรอบๆตัวเราคือวงจรแบบ **Sequential logic circuit** นั้นมี memory เพื่อ

- หน่วยความจำพวกนั้นใช้จำค่าของ input ก่อนหน้า (past input)
- Output ของวงจรไม่ได้ขึ้นกับค่า input เท่านั้นขึ้นอยู่ State ของค่าในหน่วยความจำด้วย

1.1 Sequential logic Circuits Model



รูปที่ 2 Sequential logic Circuits Model

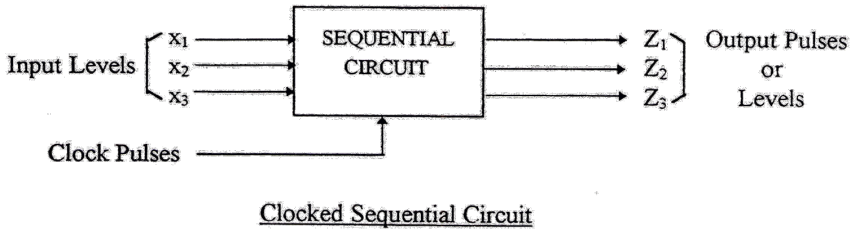
บางครั้งก็เรียกว่า Finite State Machine ซึ่งประกอบไปด้วย

- inputs จากภายนอกระบบ
- แล้วสร้าง outputs ออกไปนอกระบบ
- มีหน่วยความจำ เพื่อเก็บ State เอาไว้
- มี combinational logic ที่สร้าง output จาก input และค่าข้อมูลที่จำไว้เดิม (State)
- มีเปลี่ยนแปลงข้อมูลในหน่วยความจำในขณะผลิตค่า output

1.2 ชนิดของ Sequential logic Circuits

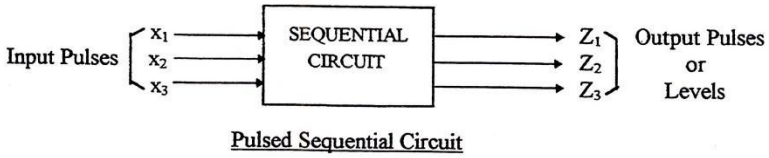
Sequential logic Circuits มี 2 ชนิดคือ synchronous และ asynchronous ทั้งสองมีความแตกต่างกันดังนี้

1) ชนิด Synchronous จะใช้ระดับการเปลี่ยนแปลงของ input และสัญญาณนาฬิกา (Clock) กระตุ้นให้นำค่า state ปัจจุบัน(ก่อนการเปลี่ยนแปลง) เข้าไปเก็บในหน่วยความจำเช่นพวก Flip-Flops



รูปที่ 3 Synchronous Sequential logic

2) ชนิด Asynchronous ไม่ใช้สัญญาณนาฬิกา แต่การทำงานจะทำทันทีที่มีระดับการเปลี่ยนแปลงของ input



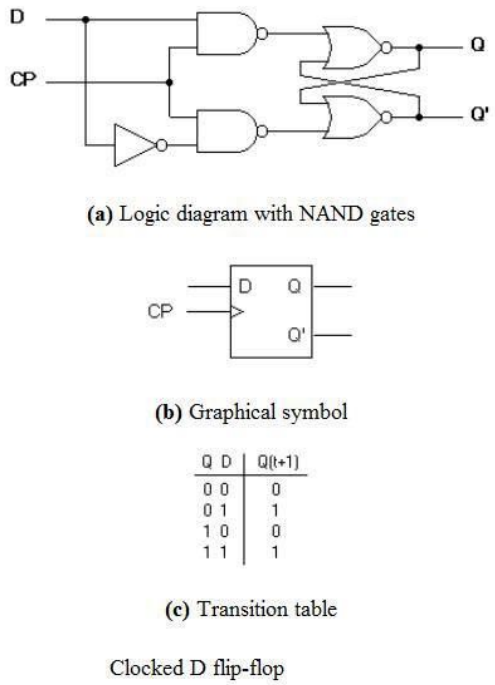
รูปที่ 4 Asynchronous Sequential logic

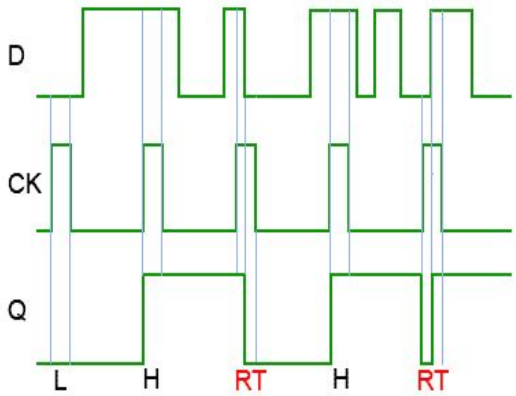
ในการทดลองทั้งหมดนี้เราจะใช้แบบ synchronous เท่านั้น

1.3 หน่วยความจำ Flip-Flops

ในวงจร synchronous Sequential logic เราจำเป็นต้องจำสถานะ (State) ของระบบไว้เพื่อใช้ในรอบต่อไป หน่วยความที่เรามักนิยมใช้ D Flip-Flop หรือ T Flip-Flop

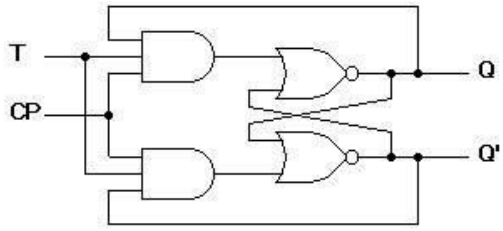
D Flip-Flop – หลักการทำงานคือค่า input D จะถูกนำไปเก็บในหน่วยความจำและนำออกที่ Output Q (ส่วน Q' คือค่าตรงข้าม) ในภาพนั้น Q คือค่าเดิม Q(t+1) ค่าใหม่ที่จะเกิดขึ้น การทำงานจะขึ้นอยู่สัญญาณนาฬิกาขอบขาขึ้น



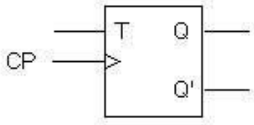


รูปที่ 5 การทำงานของ D Flip-Flop

T Flip-Flop – หลักการทำงานตคือค่า input ของ T ที่เป็นหนึ่งจะสลับค่า Output เสมอตามจังหวะสัญญาณนาฬิกาให้สังเกตุค่า Q และ Q(t=1) จะสลับกันไปมาเมื่อค่า T เป็น 1



(a) Logic diagram



(b) Graphical symbol

Q T	Q(t+1)
0 0	0
0 1	1
1 0	1
1 1	0

(c) Transition table

Clocked T flip-flop

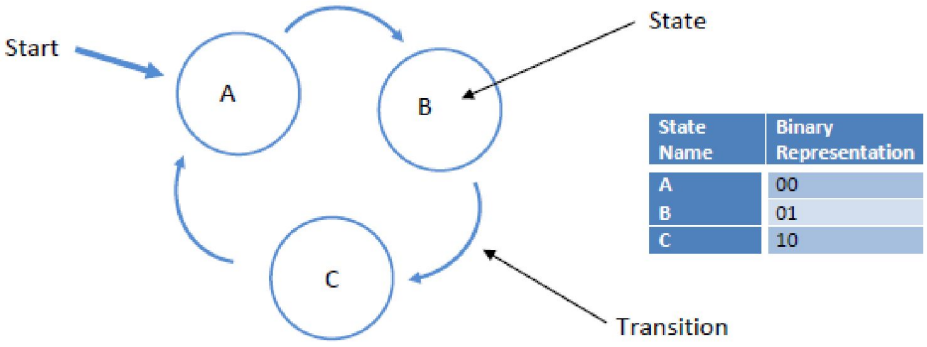
รูปที่ 6 การทำงานของ T Flip-Flop

1.4 State Machine

หากนิสิตสังเกตุอุปกรณ์ดิจิทัลที่ใช้งานมักจะเป็นอุปกรณ์ Sequential logic เกือบทั้งหมด ออกที่นิสิตจะออกแบบวงลอจิกเหล่านี้ได้ต้องมีความเข้าในเรื่อง State machine ก่อน (ใน Lab จะสอนให้เพียงเข้าในหลักการณ์ส่วนรายละเอียดให้นิสิตคอยตามในชั่วโมงบรรยาย)

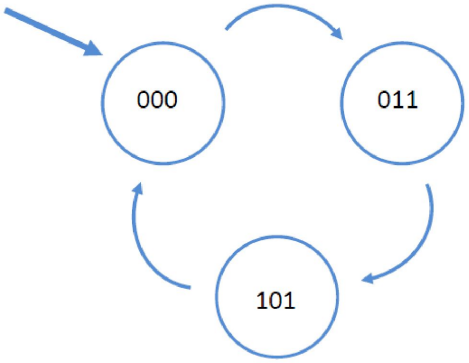
State machine คือไดอะแกรมแสดงถึงการเปลี่ยนสถานะ (State) input อะไรทำให้เปลี่ยนและจากสถานะไหนไปสถานะอื่น

ตัวอย่างที่ 1 State machine แบบง่ายๆ คือการนับอักษร A B และ C แล้ววนกลับ ซึ่งมีจำนวนสถานะจำกัด(Finite State Machine) เพียงแค่ 3 สถานะคือ A B และ C หรือแทนด้วยรหัสไบนารี่จาก 00 01 และ10 ซึ่งการเปลี่ยนสถานะจะเปลี่ยนแบบอัตโนมัติตามจังหวะสัญญาณนาฬิกา (ไม่มี input จากภายนอกมาทำให้เปลี่ยนสถานะ) ชื่อสถานะอาจจะเป็นชื่อเรียกตามภาษาพูด/เขียน แต่เราจะต้องแทนมันด้วยรหัสฐานสองดังรูป



State Name	Binary Representation
A	00
B	01
C	10

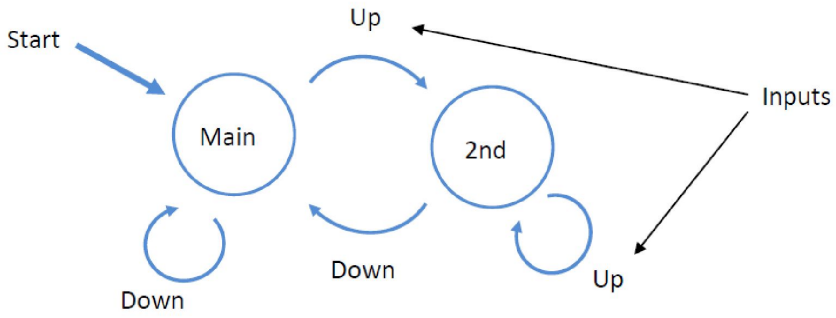
รูปที่ 7 ตัวอย่าง Finite State Machine อย่างง่าย



รูปที่ 8 Finite State Diagram หลังจากเปลี่ยนจากชื่อสถาะเป็นรหัสเลขฐาน 2 ในรูปเราใช้ 3 แทนสถานะ

ในกรณีปกติมี 3 สถานะเราสามารถใช้แค่ 2 บิตก็ได้ แต่ในรูป เราใช้รหัส 3 บิต นิสิตจะทราบเองตอนออกแบบวงจรว่าจะรหัสบิตของสถานะยิ่งมาก ยิ่งต้องใช้ หน่วยความจำ (Flip-Flop) เพิ่มมากขึ้นไปด้วยในการจำสถานะ เช่นสถานะ 2 บิตใช้ Flip-Flop 2 ตัว และ สถานะ 3 บิต ต้องใช้ Flip-Flop ทิ้ง 3 ตัว

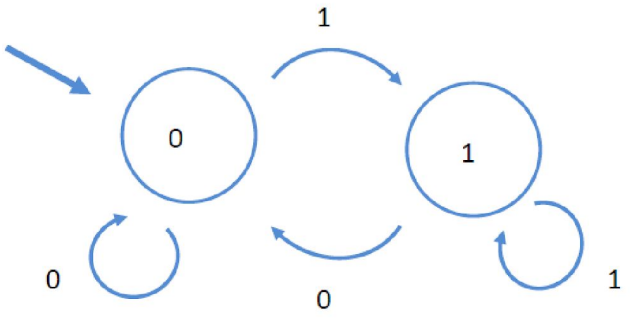
ตัวอย่างที่ 2 State machine ที่การเปลี่ยนสถานะจะขึ้นกับค่า input ด้วย ตัวอย่างนี้จะสมมุติว่า ลิฟต์ที่มีเพียง 2 ชั้นคือชั้นล่าง (Main) กับ ชั้นที่สอง (2nd) โดยปุ่มกดใน ลิฟต์คือ ปุ่มขึ้น (Up) และปุ่มลง (Down) ซึ่งเราจะเขียนไดอะแกรมได้ดังนี้



รูปที่ 9 Finite State Diagram ของการทำงานระบบลิฟต์แค่ 2 ชั้น

จากไดอะแกรมถ้านิสิตอยู่ที่ชั้น main แล้วกดปุ่ม Up ลิฟต์จะขึ้นไปชั้น 2 แล้วถ้ากดปุ่ม Up ลิฟต์จะอยู่ที่เดิม แต่ถ้าปุ่ม Down ลิฟต์ก็จะเลื่อนลงชั้น main

นิสิตจะเห็นว่าเราสามารถเปลี่ยนคำพูดต่างเป็นรหัสไบนารี่ได้ดังรูปถัดไป



State Name	Binary Representation
Main	0
2nd	1

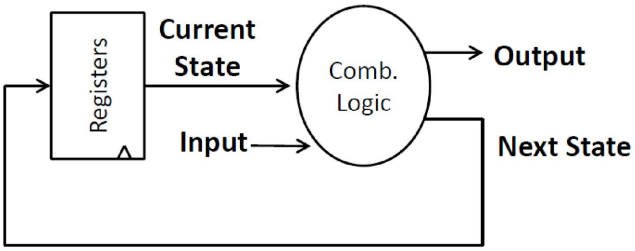
Button/ Input	Binary Representation
Down	0
Up	1

รูปที่ 10 Finite State Diagram ของระบบลิฟต์เมื่อให้รหัสไบนารี่แล้ว

1.5Mealy and Moore Machines

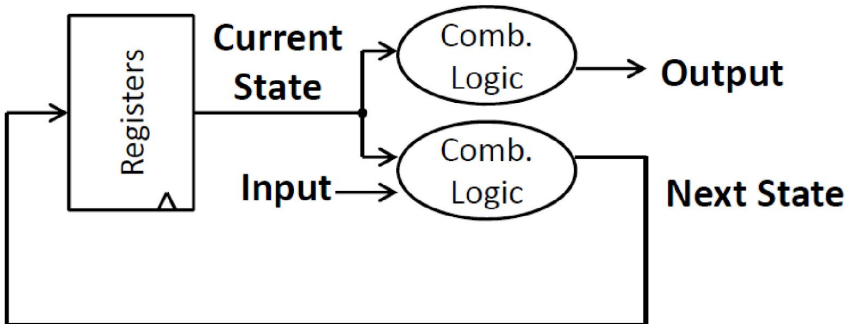
Finite State Machine ยังสามารถพิจารณาการทำงานจากความสัมพันธ์ของ Output กับ สถานะปัจจุบัน (Current State) กับ input อีกด้วย

1) Mealy Machine การผลิตค่า Output จะขึ้นกับสถานะปัจจุบัน (Current State) กับ input



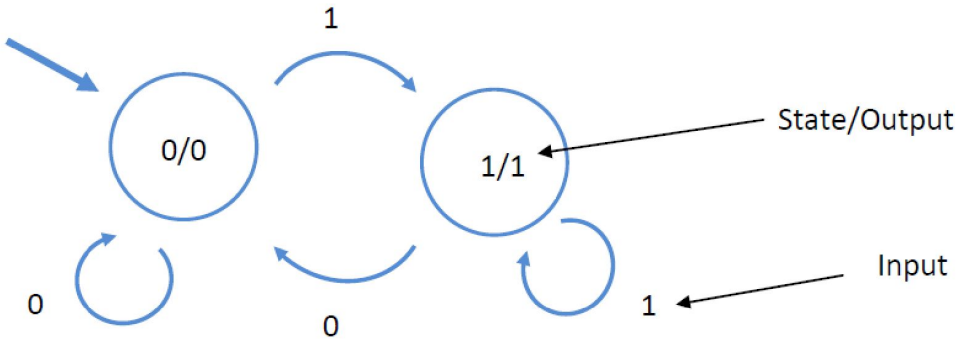
รูปที่ 11 Mealy Machines

2) Moore Machine การเปลี่ยนการผลิตค่า Output จะขึ้นกับสถานะปัจจุบัน (Current State) อย่างเดียวเท่านั้น



รูปที่ 12 Moore Machines

จากกรณีตัวอย่างของลิฟต์ 2 ชั้นเป็น moore machine ถ้าเราสมมุติว่า แต่ละชั้น (สถานะ) ผลิต output เป็นค่าลอจิกตามชั้นของลิฟต์ที่อยู่ หรือพูดง่ายคือ แต่ละสถานะจะมีค่าประจำตัวอยู่แล้วซึ่งในที่นี้เราสมมุติให้ค่าของสถานะเป็นค่า Output ด้วยเลย ดังภาพ

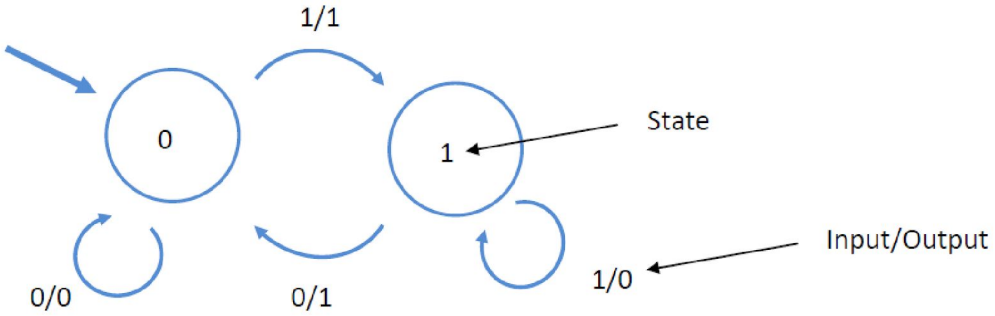


รูปที่ 13 ลิฟต์ 2 ชั้นแบบ moore machine

แต่ถ้าเราเปลี่ยนความต้องการใหม่ เช่น

- เมื่อกดปุ่ม 1 (ปุ่ม Up ตามรหัสที่เราตั้ง) ถ้าเราต้องการให้ลิฟต์ขึ้น และกดปุ่ม 0 (ปุ่ม Down) เมื่อเราต้องการให้ลิฟต์ลง
- ให้ระบบผลิตสัญญาฯ Output เป็น 1 เมื่อต้อเปลี่ยนชั้น ถ้าไม่เปลี่ยนชั้นให้ Output เป็น 0

การเปลี่ยนแปลงนี้ เราจะได้ State Machine เป็น mealy machine ดังรูป เพราะ Output จะขึ้นอยู่กับทั้ง สถานะและ input ด้วย (แต่จริงๆ นิสิตจะเห็นว่า เราสามารถเปลี่ยน mealy machine ไปเป็น moore machine ได้ด้วยการเพิ่มสถานะ (State) แล้วกำหนดค่า Output ให้เป็นไปตามค่าประจำของสถานะ (รายละเอียดจะได้ศึกษาในชั่วโมงบรรยายอีกครั้ง)



รูปที่ 14 ลิฟต์ 2 ชั้นแบบ mealy machine

และในการทดลองนี้เราจะพิจารณาเพียงกรณี Moore Machine ก่อน

1.6การใช้งาน Flip-flops เพื่อจำสถานะ

เพื่อเก็บสถานะ (State) เอาใช้ในการพิจารณาการเปลี่ยนสถานะใหม่เราจำเป็นต้องมีการเก็บค่าสถานะไว้ ในทางปฏิบัติมีหลายวิธี เช่น RS D T JK Filp-Flop แต่ในการทดลองจะใช้ D Flip-Flop เป็นหลัก รวมทั้งในการจำลองด้วย

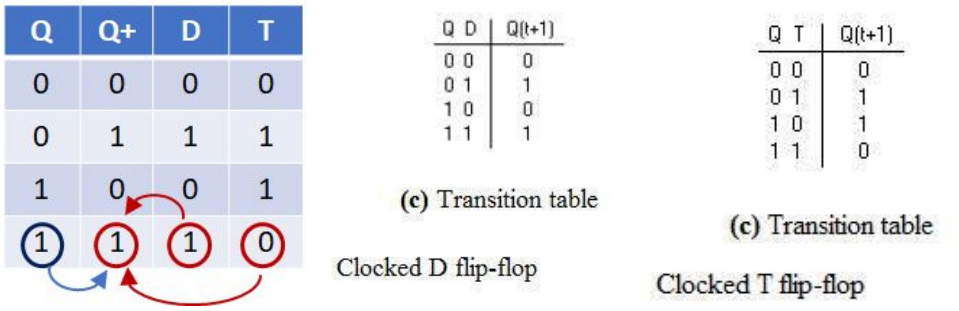
- *D Flip-Flop* ใช้เพื่อเป็นหน่วยความจำในการเก็บได้โดยตรง (นิสิตย้อนกลับดูการทำงาน D Flip-Flop) เพราะมันจะบันทึกค่าสถานะใหม่เข้าไปเก็บตัว
- *T Flip-Flop* ใช้เพื่อเป็นค่านับ (Counter) เพราะค่าที่ถูกเก็บจะสลับไปสลับมาเสมอ ถ้าค่า input เป็น 1

รูปที่ 15 D และ T Flip-Flop

เราสามารถสร้าง D/T Flip-Flop ได้จาก JK Flip-Flop

รูปที่ 16 D/T Flip-Flop สร้างจาก JK Flip-Flop

เพื่อให้เกิดความเข้าใจมากขึ้นจะแสดงการเปลี่ยนแปลงสถานะ (Next State Table) และค่าของ D/T Flip-Flop ให้เห็นพร้อมกัน



Q	Q+	D	T
0	0	0	0
0	1	1	1
1	0	0	1
1	1	1	0

Q D	Q(t+1)
0 0	0
0 1	1
1 0	0
1 1	1

(c) Transition table

Clocked D flip-flop

Q T	Q(t+1)
0 0	0
0 1	1
1 0	1
1 1	0

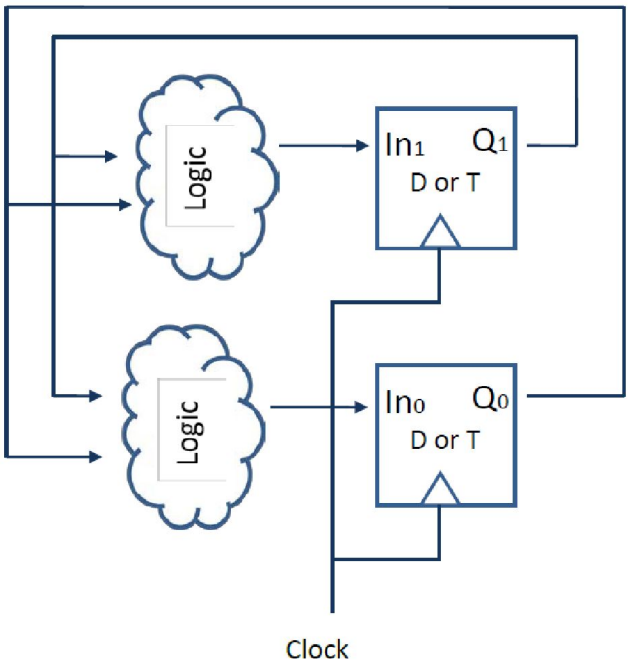
(c) Transition table

Clocked T flip-flop

รูปที่ 17 ตารางการเปลี่ยนแปลงสถานะ (Next State Table) และค่าของ D/T Flip-Flop ที่ใช้กำหนดเพื่อใช้เก็บค่าสถานะ

ถ้า Q คือสถานะปัจจุบัน ส่วน Q+ สถานถัดไป ดังในรูปเมื่อ Q เป็น 1 ซึ่งถัดไปยังคงเป็น 1 คือ Q+ = 1 เมื่อสัญญาณนาฬิกามากระทบ D Flip-Flop จะเก็บค่า 1 ไว้ให้ (หรือในกรณี Q=1 แล้วถัดไปเปลี่ยนเป็น 1 Q+ = 0 D Flip-Flop ก็จะเก็บค่า 0 ให้) ส่วนของ T Flip-Flop ก็จะตรงการตารางการทำงานทุกประการ

จากตารางขางต้นรูปที่ 17 สิ่งที่เราต้องการคือ ค่า D (หรือ T ถ้าใช้ T Flip-Flop) ดังนั้นเราเพียงแค่หาความสัมพันธ์ระหว่าง D และ Q ทำนองเดี๋ยวกับ T ก็ต้องหาความสัมพันธ์ในรูปสมการลอจิก (Combination Logic) วงจร Sequential logic ที่เราจะได้จะเป็นดังในรูปโดยคราวๆ



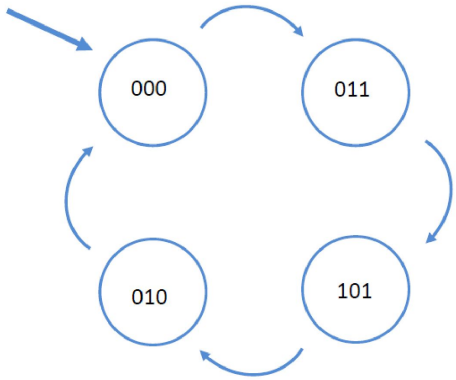
รูปที่ 18 วงจร Sequential logic ที่หาได้จาก Combination Logic D และ Q0 และ Q1

การทดลองที่ 5.1 สร้างและจำลอง Sequential logic ของ การนับ A B C และ D

อุปกรณ์ที่ต้องใช้ โปรแกรม logicism ไม่ต้องบอร์ดทดลองจริง

การทดลอง

การนับ A B C และ D ซึ่งสามารถ เขียน State Diagramได้ดังนี้



1) กำหนดใช้ 3 บิตแทน State ที่ เรียกว่า Q0 (000) Q1 (011) Q2 (101) และ Q3 (010)โดย Next State คือ Q0+ Q1+ Q2+ และ Q3+สร้างตารางแสดง Current State และ Next State (State machine นี้ไม่ทั้ง output และ input)

2) กำหนดให้ใช้ D Flip-Flop เมื่อเรากำหนดรหัส 3 บิตกับ State ที่มี ดังนั้นเราต้องใช้ ใช้ D Flip-Flop ถึง 3 ตัว ให้สร้างตารางการเปลี่ยนสถานะ และกำหนดค่า D0 D1 และ D2 ให้ตรงกันที่ให้ข้างบนข้อ 1.6 เพื่อจำสถานะใหม่

Q	Q+	D
0	0	0
0	1	1
1	0	0
1	1	1

Current and Next State Table

Q2	Q1	Q0	Q2+	Q1+	Q0+
0	0	0	0	1	1
0	1	1	1	0	1
1	0	1	0	1	0
0	1	0	0	0	0

D Values Required

D2	D1	D0
0	1	1
1	0	1
0	1	0
0	0	0

นิสิตจะได้

D2 Generation Table (from Q2/Q1/Q0 values)

	Q2/Q1			
Q0	00	01	11	10
0	0	0	x	x
1	x	1	x	0

D2 = Q1 & Q0

D1 Generation Table (from Q2/Q1/Q0 values)

	Q2/Q1			
Q0	00	01	11	10
0	1	0	x	x
1	x	0	x	1

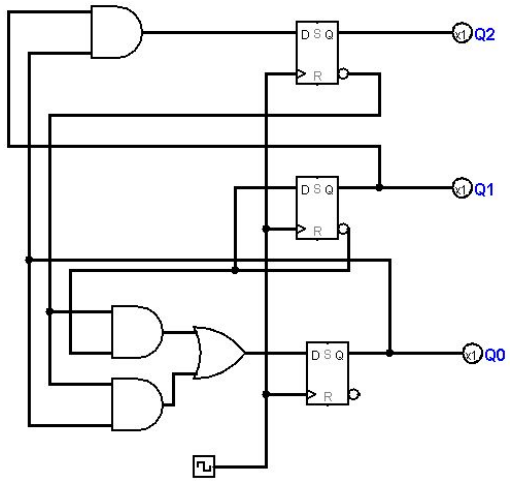
D1 = 'Q1

D0 Generation Table (from Q2/Q1/Q0 values)

	Q2/Q1			
Q0	00	01	11	10
0	1	0	x	x
1	x	1	x	0

D0 = 'Q2 & 'Q1 | 'Q2 & Q0

3) จาก K-map นิสิตจะได้วงจรลอจิกดังนี้ ให้นิสิตต่อวงจรใน logisim พร้อมจำลองการทำงานว่าตรงไหม

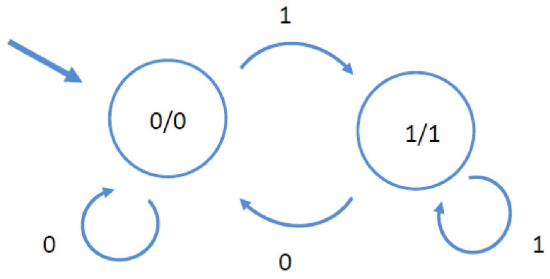


การทดลองที่ 5.2 สร้างและจำลอง Sequential logic ของ ลิฟท์
อุปกรณ์ที่ต้องใช้ โปรแกรม logicism และบอร์ดทดลองจริง

No.	Description of Item	Quantity
1	NX-4i Board	1
2	74ls74	1
3		
4		
5		
6		

การทดลอง

1) เพื่อให้นิสิตเห็นภาพของ Sequential logic ที่มี input ทำให้เกิดการเปลี่ยนแปลงสถานะ เราจะใช้ตัวอย่างง่ายที่สุดของระบบ ลิฟต์ 2 ชั้นที่อธิบายไปแล้วข้างต้น ซึ่งเป็นวงจรแบบ moore machine เขียน State Diagramได้ดังนี้



2) การวิเคราะห์เรากำหนดรหัสให้สถานะเพียง บิตเดียว ดังนั้นเราใช้ D Flip-Flop แค่ตัวเดียว นิสิตเขียนสร้างตารางการเปลี่ยนสถานะ และกำหนดค่า D ให้ตรงกันที่ให้ข้างบนข้อ 1.6 เพื่อจำสถานะใหม่

Q0	Input	Q0+	Output
0	0	0	0
0	1	1	1
1	0	0	0
1	1	1	1

⇨

D0
0
1
0
1

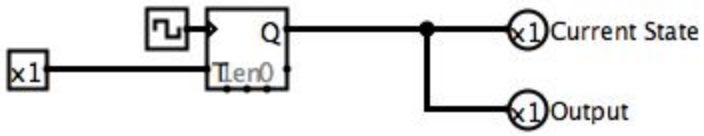
Q	Q+	D
0	0	0
0	1	1
1	0	0
1	1	1

3) จากนั้นให้นิสิตใช้ K-map หาความสัมพันธ์ของ D กับ Q0 และ input เช่นเดียวกัน หาความสัมพันธ์ของ Output กับ Q0 และ input

	Q0	
Input	0	1
0	0	0
1	1	1
D= Input		

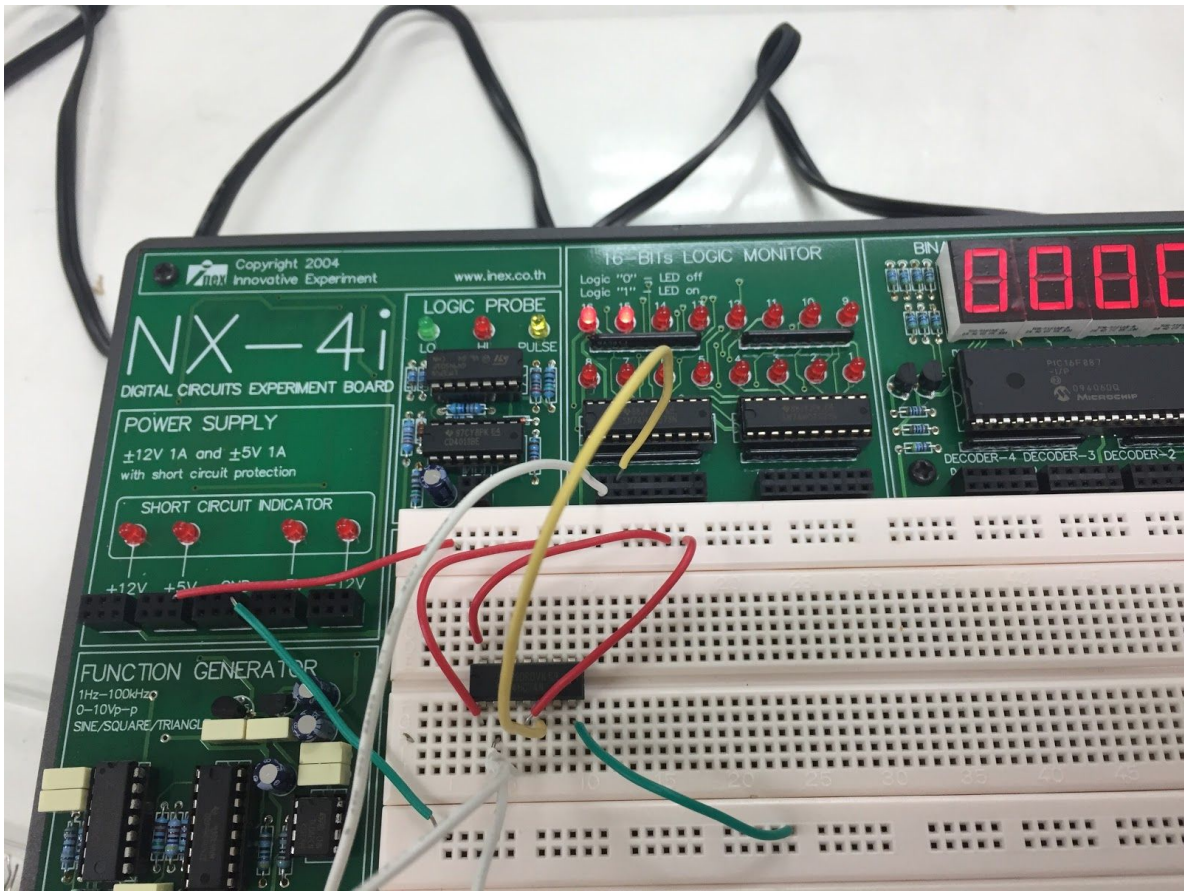
Q0		
Input	0	1
0	0	0
1	1	1
Output= Input		

4) วาดวงจรใน Logisim พร้อมจำลองการทำงาน



ภาพไดอะแกรมวงจร

5) ต่อวงจรใน NX-4i Board ทดสอบการทำงาน



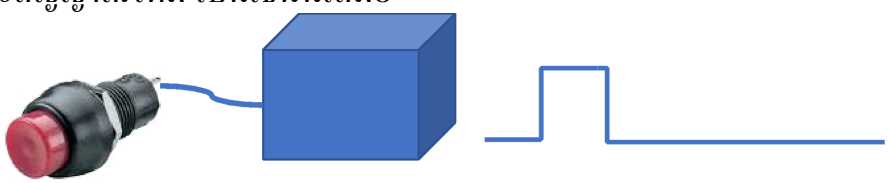
ภาพถ่ายวงจรบนบอร์ดทดลอง NX-4i

การทดลองที่ 5.3 สร้างและจำลอง Sequential logic ของสร้างสัญญาณออกเป็น pulse ระดับสูงเพียงแค่ลูกคลื่นเดียว

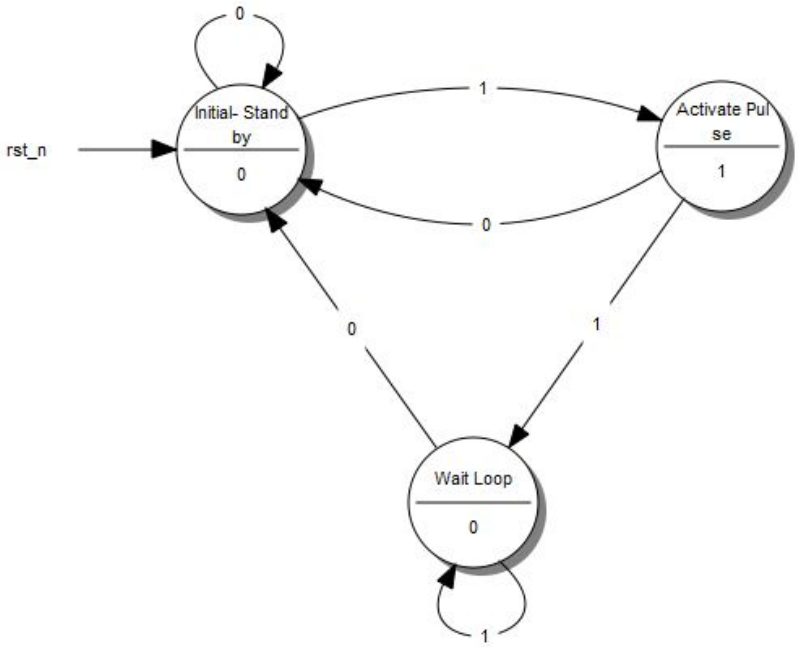
อุปกรณ์ที่ต้องใช้ โปรแกรม logicism ไม่ต้องบอร์ดทดลองจริง

การทดลอง

ให้นิสิตสร้างวงจร Sequential logic ที่ทำงานเมื่อปุ่มถูกกดจะวงจรจะสร้างสัญญาณ pulse ออกมาเพียงหนึ่งลูกคลื่น และจะไม่สร้างอะไรออกมาอีก ถ้าไม่ปล่อยปุ่มที่กด (หมายถึงถ้ากดปุ่มค้างไว้) แต่เมื่อปล่อยปุ่มแล้วกดใหม่ก็จะสัญญาณใหม่ เป็นเช่นนี้เสมอ



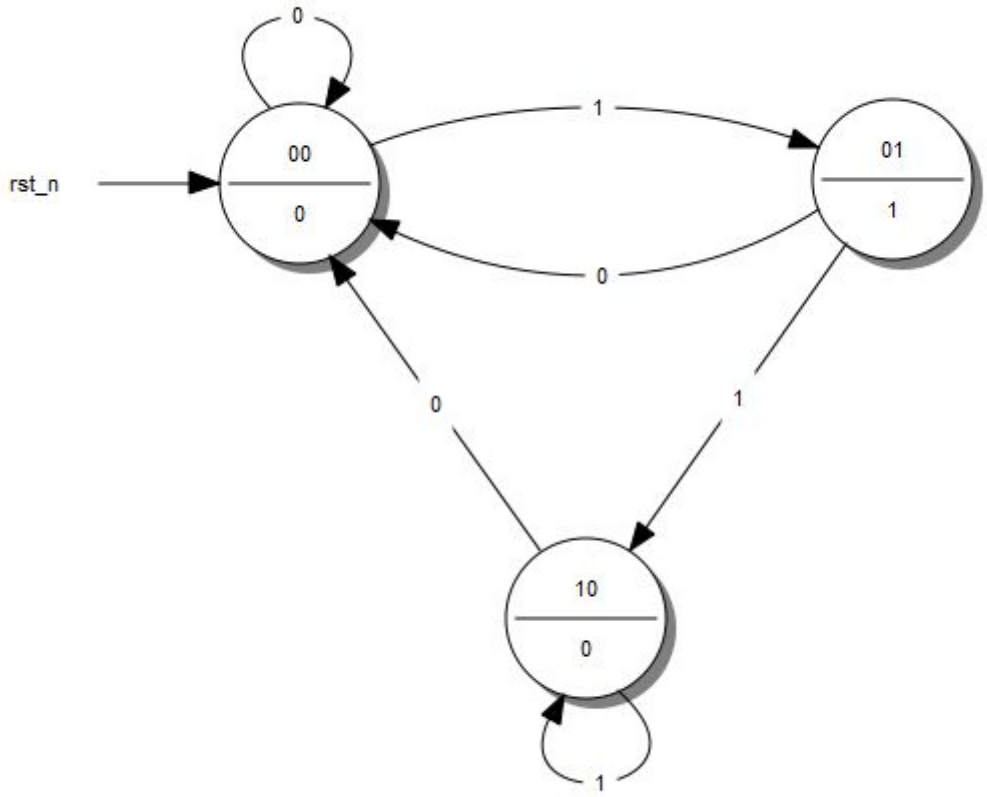
1) จากนิยามการทำงานของวงจรสามารถเขียน State diagram ได้ดังนี้



เมื่อเริ่มต้นทำงานวงจรจะไม่สร้างอะไรออกมาจนกว่าปุ่มจะถูกกด นิสิตอาจจะคิดว่ามีแค่ 2 State ก็พอ แต่เมื่อพิจารณาแล้วเราจำเป็นต้องมี State เพิ่ม เพราะในกรณีที่ไม่ได้ปล่อยปุ่มกดนั้นวงจรจะต้องไม่สร้างสัญญาณอะไรออกมาอีก ในรูปนั้น ในวงกลมคือชื่อ State ค่าใต้ State คือค่า output การกดปุ่มคือค่าลอจิก 1 และเมื่อปล่อยปุ่มคือค่าลอจิก 0

เมื่อปุ่มถูกกดระบบจะเปลี่ยน State จาก initial- Standby เป็น ที่ Active Pulse ซึ่งจะสร้างสัญญาณ Output ลอจิกสูง “1” ออกมา และหากปุ่มไม่ถูกปล่อยในระยะเวลาสัญญาณนาฬิกาลูกคลื่นต่อมานั้นระบบยังตรวจจับได้ว่าปุ่มยังไม่กด จังหวะนี้ระบบจเปลี่ยน State ไปเป็น Wait loop ซึ่งจะลดระดับลอจิก Output ลงเป็นต่ำ “0” นั้นคือที่มาของ State ที่เพิ่มขึ้น เมื่อปุ่มถูกปล่อย ระบบจะเปลี่ยน State กลับไป State เริ่มต้นอีก

2) ทำการให้รหัสกับชื่อ State เนื่องจากต้องใช้ในการสร้าง Combination logic ในระบบมี 3 State เราจึงต้องใช้ 2 บิต แทนที่ชื่อ State ซึ่งทำให้เราต้องใช้ D Flp-Flop 2 ตัวนั้นเองดังรูป



3) จากนั้นสร้างตารางการเปลี่ยนสถานะตามค่า input พร้อมกับค่า Output ประจำ State

Current State		Input	Next State		Outputs	Flip Flop Inputs	
A	B	I	A_{next}	B_{next}	Y	D_A	D_B
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	1
0	1	0	0	0	1	0	0
0	1	1	1	0	1	1	0
1	0	0	0	0	0	0	0
1	0	1	1	0	0	1	0
1	1	0	X	X	X	X	X
1	1	1	X	X	X	X	X

4) เลือกใช้ D Flip-Flop แล้วกำหนดค่า DA และ DB ให้ตรงกัน

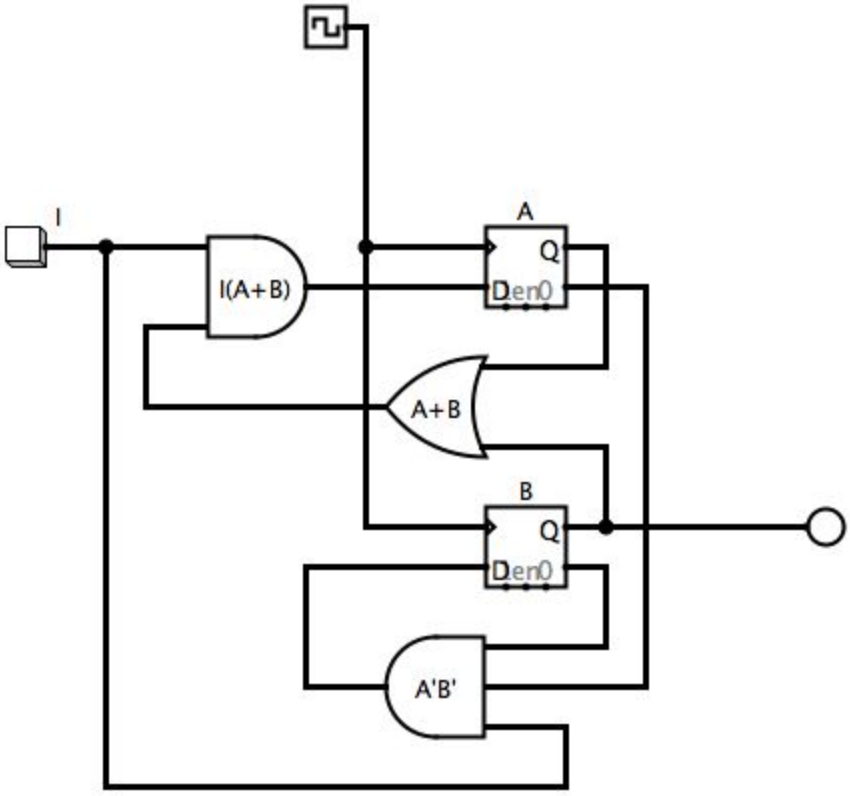
Current State		Input	Next State		Outputs	Flip Flop Inputs	
A	B	I	A_{next}	B_{next}	Y	D_A	D_B
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	1
0	1	0	0	0	1	0	0
0	1	1	1	0	1	1	0
1	0	0	0	0	0	0	0
1	0	1	1	0	0	1	0
1	1	0	X	X	X	X	X
1	1	1	X	X	X	X	X

5) ใช้ K-map หาความสัมพันธ์สมการลอจิก DA กับ State และ Input เช่นเดียวกัน หาความสัมพันธ์สมการลอจิก DB กับ State และ Input

	AB			
I	00	01	11	10
0	0	0	d	0
1	0	1	d	1
DA= BI+AI				

	AB			
I	00	01	11	10
0	0	0	d	0
1	1	0	d	0
DB= A’B’I				

6) วาดวงจรใน Logisim พร้อมจำลองการทำงาน



ภาพไดอะแกรมวงจร