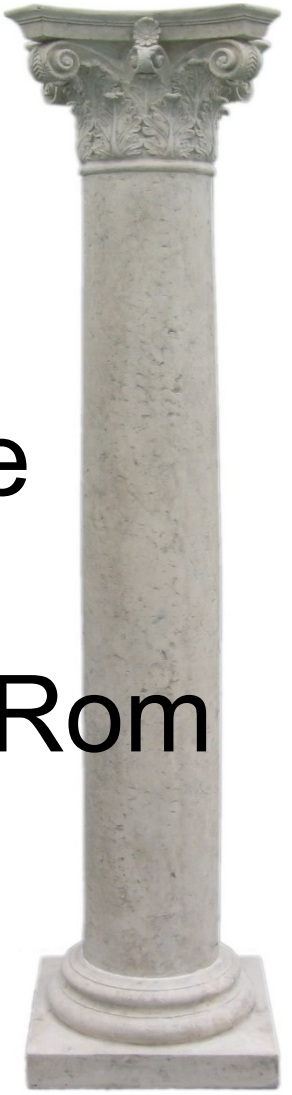
CS121 Digital Logic and Smart Device

Chapter 5

Sequential Circuit

By Dr. Paween Khoenkaw
Computer Science MJU

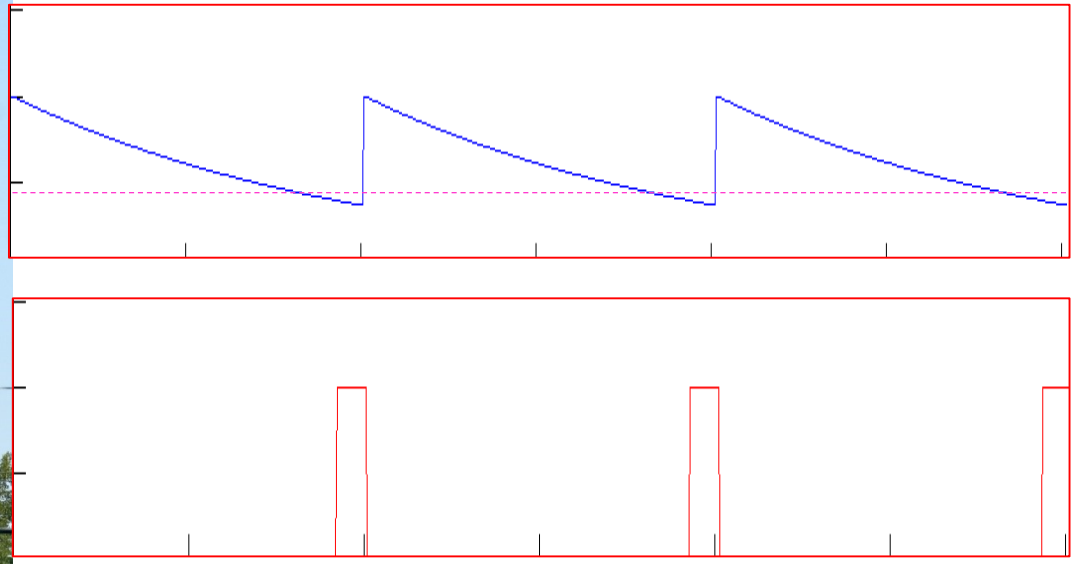
- สถานะ (State)
- สัญญาณนาฬิกา (Clock)
- ฟลิปฟล็อป (Flip Flop)
- เครื่องจักรสถานะจำกัด (Finite State Machine)
- การออกแบบวงจรด้วยรอม และ แรม (Rom Ram Circuit Implementation)



วงจรควบคุมปั๊มน้ำอัตโนมัติ

เซ็นเซอร์วัดระดับน้ำ a (0=ไม่จุ่มน้ำ, 1=จุ่มน้ำ)

$$y = \bar{a}$$

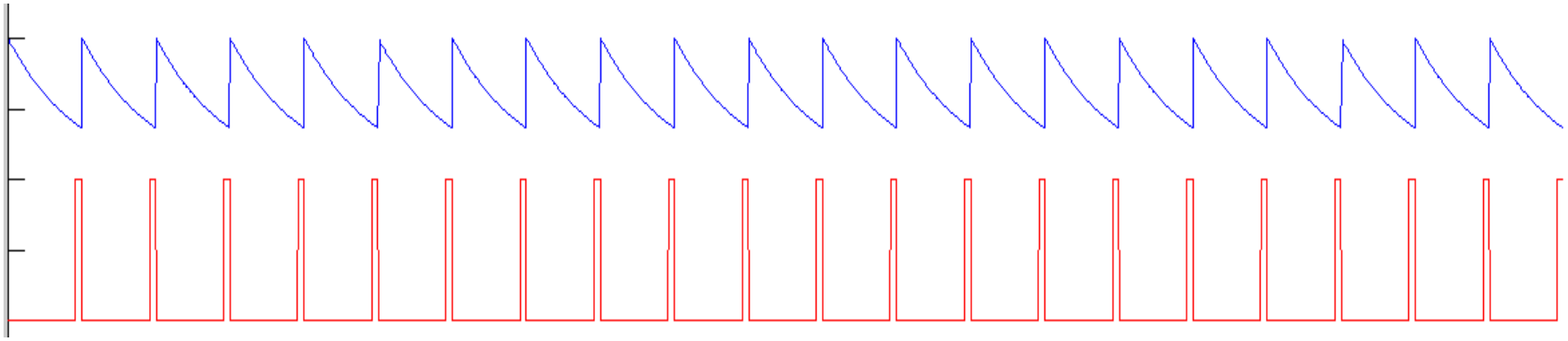


ระดับน้ำ

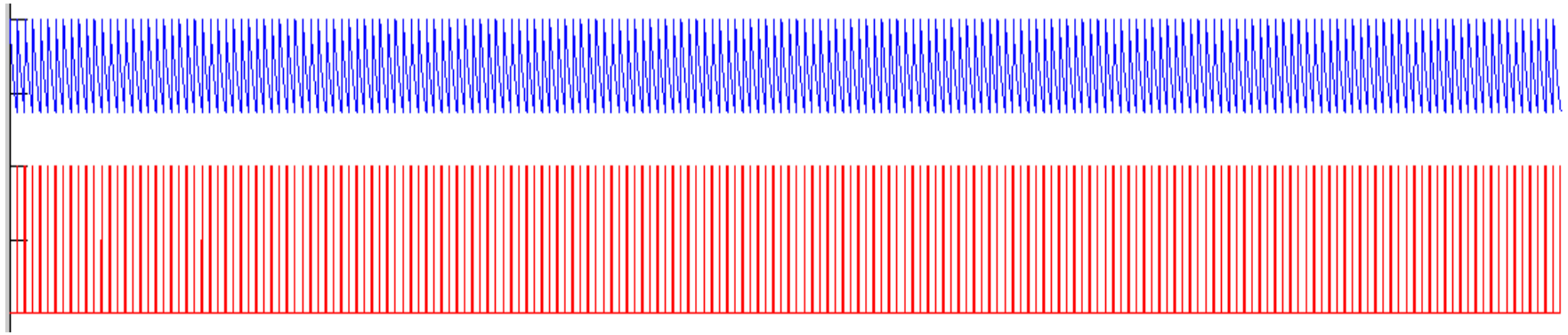
ปั๊ม

ปั๊มน้ำ y (0 =ปิด, 1=เปิด)

สถานะของวงจร



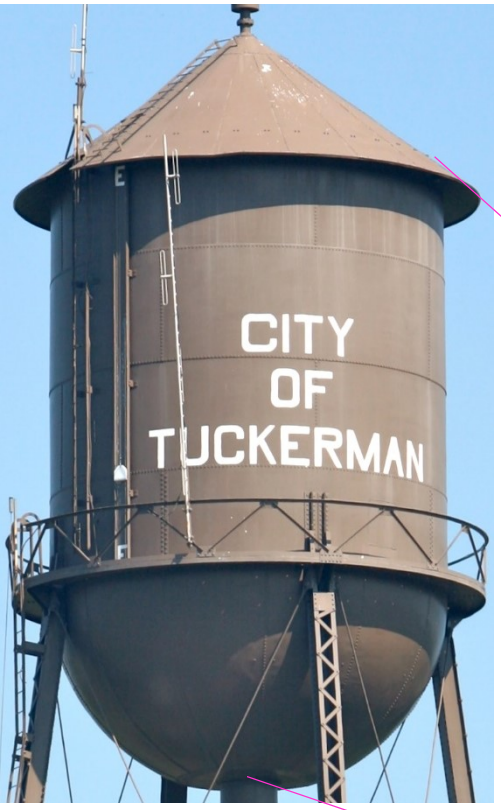
1 นาที

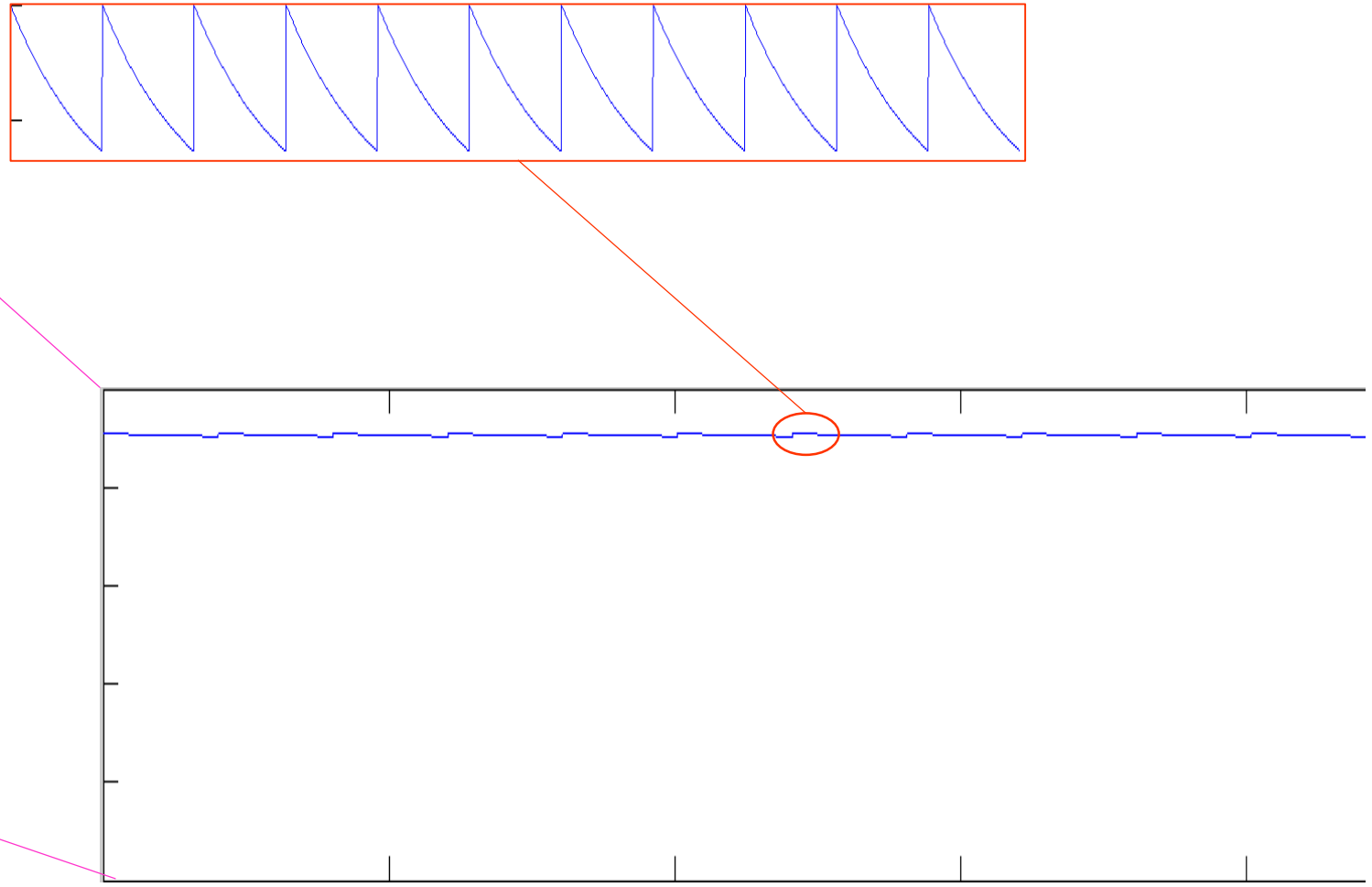


1 ชั่วโมง

1 วัน

วงจรควบคุมปั๊มน้ำอัตโนมัติ





ปัญหานี้เรียกว่า การกระดอน (bouncing)

หากเกิดขึ้น อุปกรณ์อิเล็กทรอนิกส์อาจเสียหายได้

สถานะของวงจร

เซ็นเซอร์วัดระดับน้ำ a (0=ไม่จุ่มน้ำ, 1=จุ่มน้ำ)

เซ็นเซอร์วัดระดับน้ำ b (0=ไม่จุ่มน้ำ, 1=จุ่มน้ำ)

b เป็น 0 ให้เปิดปั๊ม (y=1)

a เป็น 1 ให้ปิดปั๊ม (y=0)

a	b		y
0	0	ไม่มีน้ำ	1
0	1	น้ำไม่เต็มถัง	1
1	0	เป็นไปไม่ได้	0
1	1	น้ำเต็ม	0

ปั๊มน้ำ y (0 =ปิด, 1=เปิด)

$$y = \bar{a}$$

เกิดการกระดอนอยู่ดี

สถานะของวงจร

เซ็นเซอร์วัดระดับน้ำ a (0=ไม่จุ่มน้ำ, 1=จุ่มน้ำ)

เซ็นเซอร์วัดระดับน้ำ b (0=ไม่จุ่มน้ำ, 1=จุ่มน้ำ)

b เป็น 0 ให้เปิดปั๊ม (y=1)

a เป็น 1 ให้ปิดปั๊ม (y=0)

a	b		y
0	0	ไม่มีน้ำ	1
0	1	น้ำไม่เต็มถัง	0
1	0	เป็นไปไม่ได้	0
1	1	น้ำเต็ม	0

ปั๊มน้ำ y (0 =ปิด, 1=เปิด)

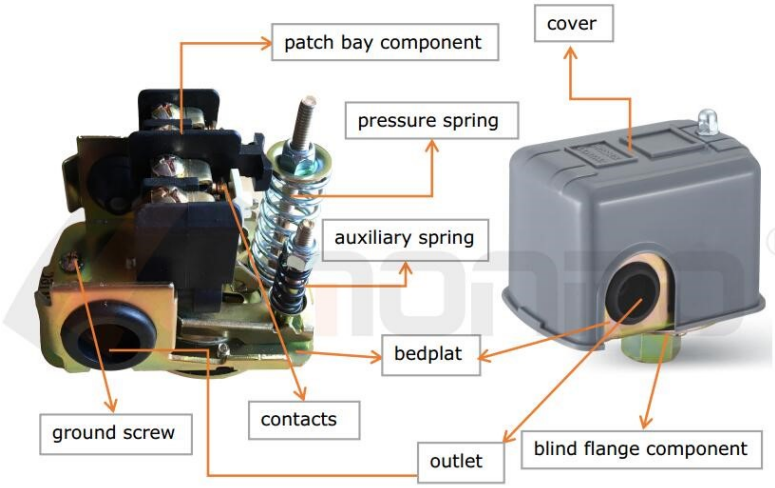
$$y = \overline{ab}$$

เกิดการกระดอนด้านล่างอยู่ดี และน้ำจะไม่มีวันเต็มถัง

สถานะของวงจร

ปัญหานี้แก้ไม่ได้ด้วยวงจรแบบจัดกลุ่ม (combination circuit)

วงจร Analog ก็แก้ไม่ได้ ปัญหานี้ ต้องใช้วิธีการเชิงกล



สวิตซ์ของปั๊มน้ำจะมีสปริง 2 ตัว

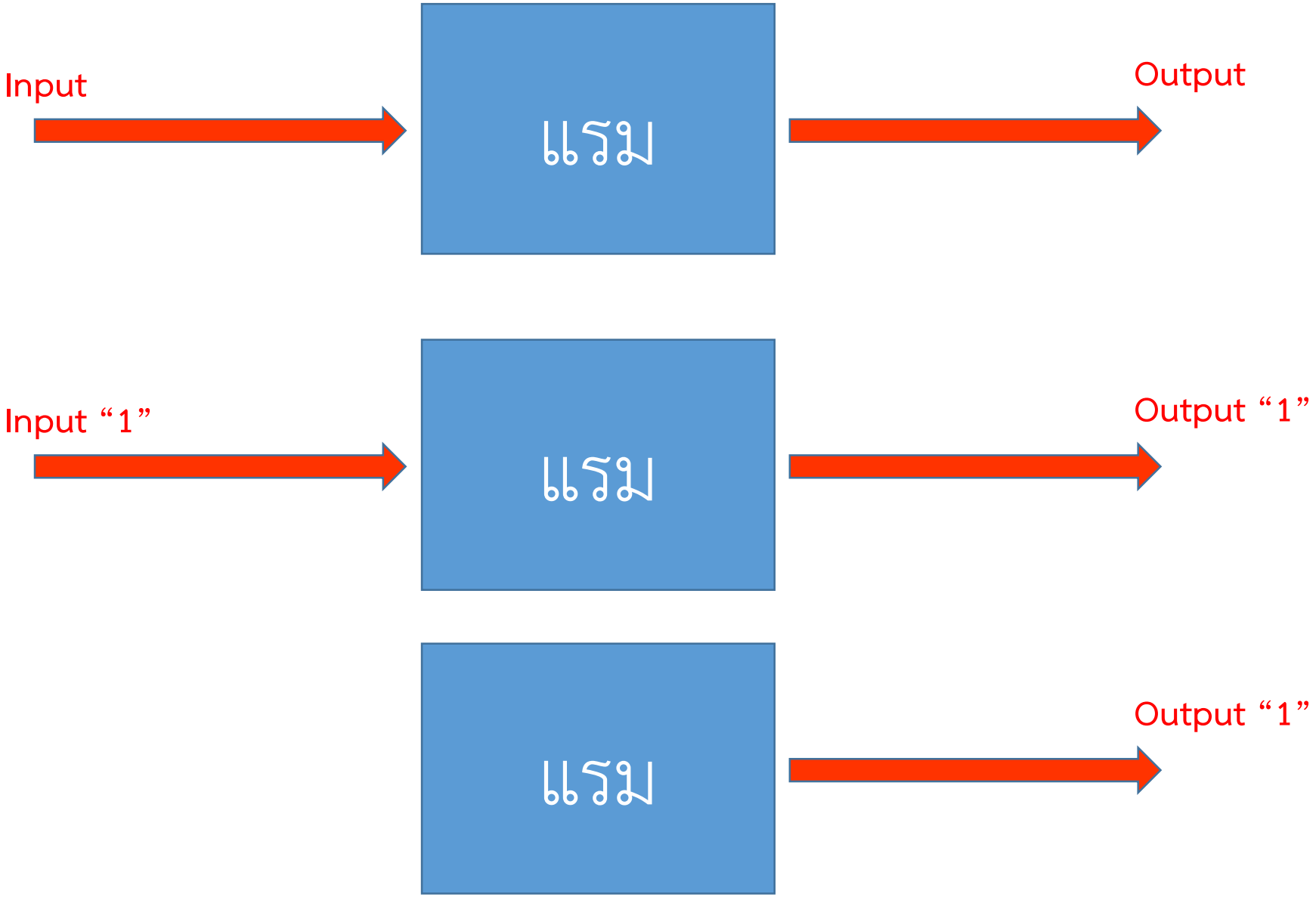
ขณะที่ความดันค่อยๆสูงขึ้น(มีการเติมน้ำ) สปริงทั้ง2 ตัวจะค่อยๆแน่นขึ้น จนถึงจุดหนึ่ง มันจะดันสวิตซ์ให้ตัดวงจรและสปริงตัวแรกจะถูกล๊อคไว้

เมื่อความดันลดลอง(ปล่อยน้ำ) สปริงตัวที่สองจะค่อยๆคลายลง เมื่อคลายถึงจัดที่ตั้งไว้ มันจะปลดล๊อคสปริงตัวแรก ทำให้สวิตซ์กลับมาเชื่อมวงจรเหมือนเดิม

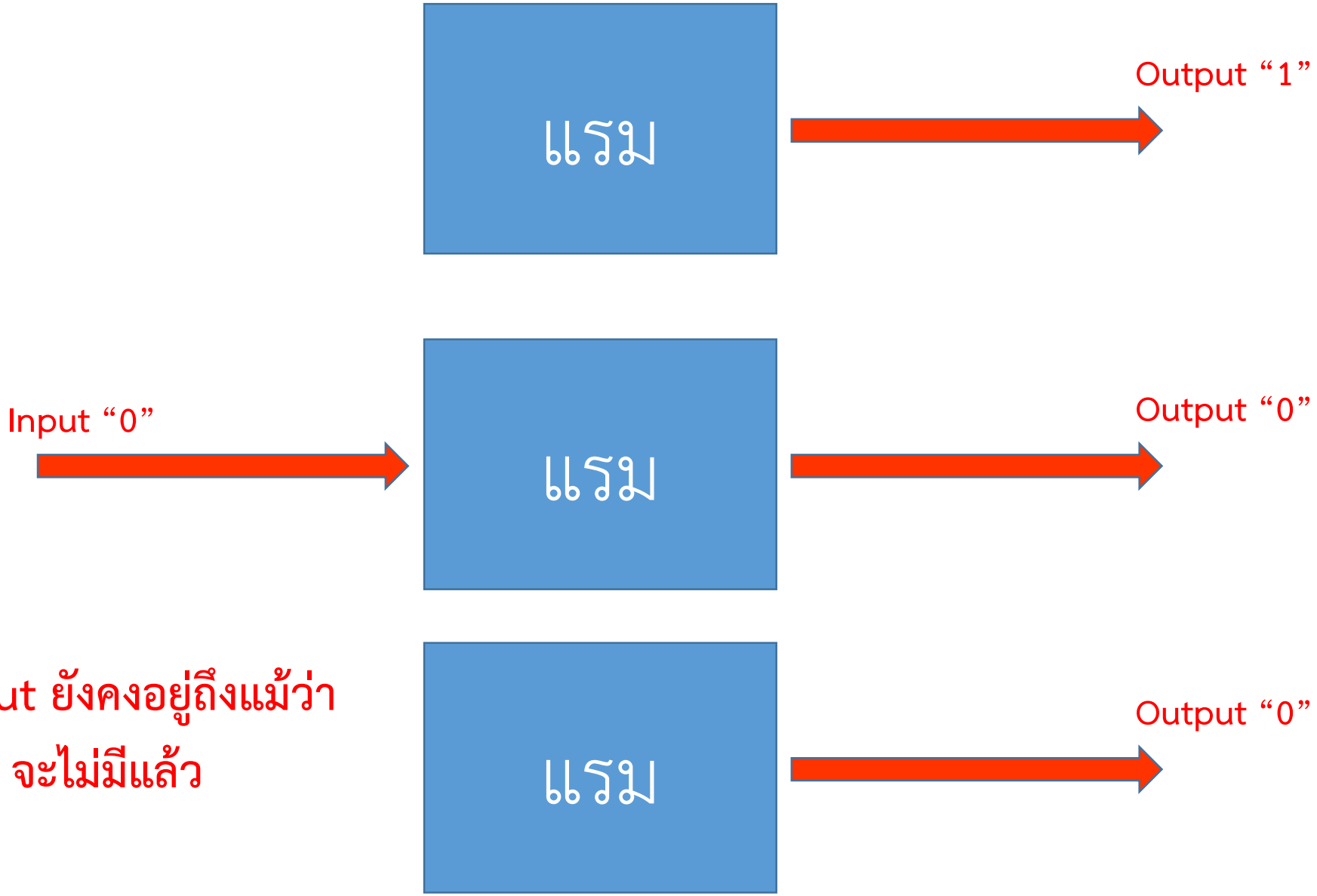
มีการจดจำสถานะ ว่ากำลังเติมน้ำ หรือกำลังปล่อยน้ำ โดยใช้ สปริงในการจดจำสถานะ

วงจรดิจิทัล จะทำแบบเดียวกันได้ไหม . .? ใช้แรม แทนสปริง ?

แล้วแรมทำงานอย่างไร ?



แล้วแรมทำงานอย่างไร ?



Output ยังคงอยู่ถึงแม้ว่า Input จะไม่มีแล้ว

สถานะของวงจร

Output ยังคงอยู่ถึงแม้ว่า Input จะไม่มีแล้ว

การคงค่า Input นั้นเรียกว่าการ “ Latch”

หรือเรียกว่าการคงสถานะ (state) ของวงจร

การทำให้สถานะของวงจรเป็น “logic 1” คงอยู่ เรียกว่าการ set

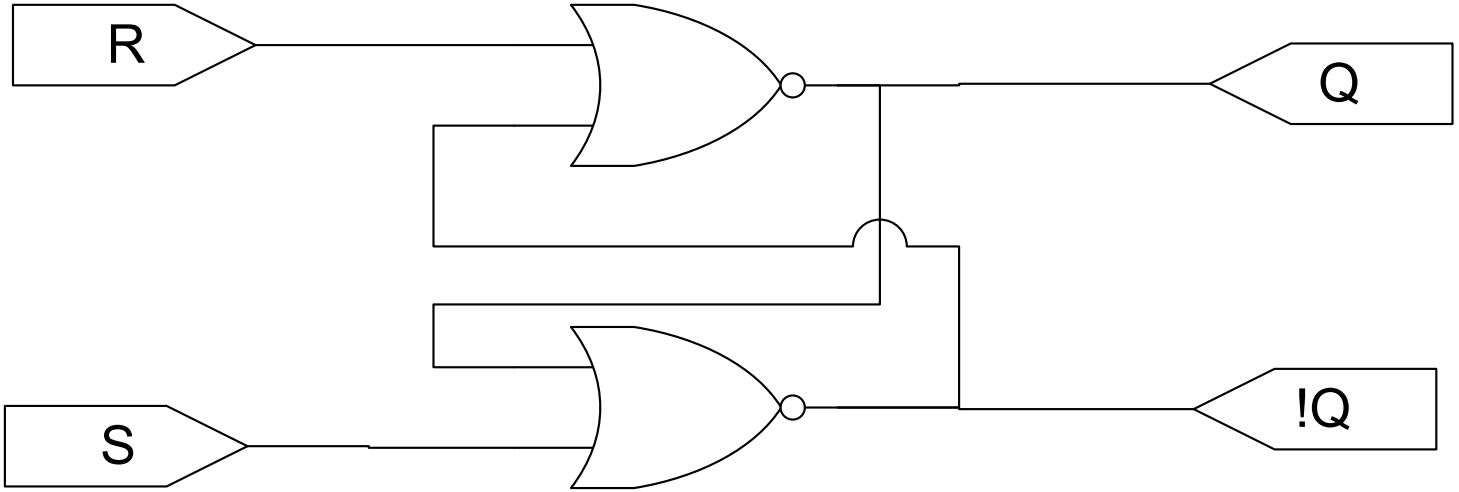
การทำให้สถานะของวงจรเป็น “logic 0” คงอยู่ เรียกว่าการ reset

วงจรที่ใช้ในการ Latch สถานะลอง logic เรียกว่าวงจร ฟลิปฟล็อป (Flip Flop)

ฟลิปฟล็อป

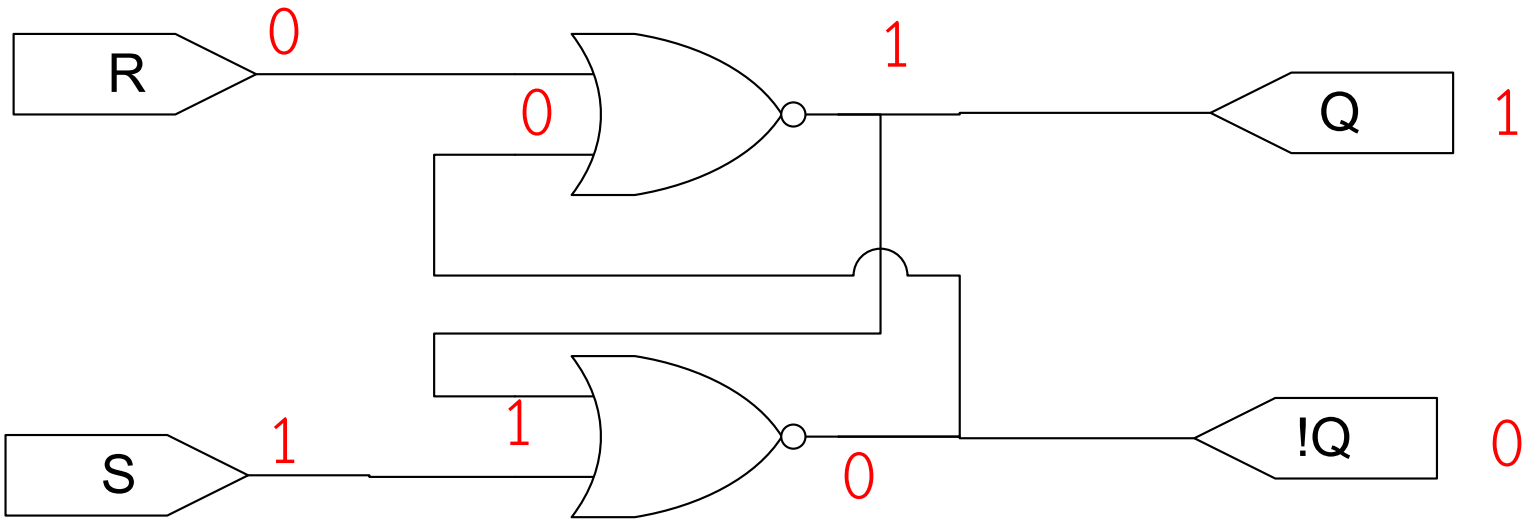
ฟลิปฟล็อป (Flip Flop) คือวงจรที่สามารถคงสถานะของตัวเองได้ โดยมีอยู่ สองสถานะ โดยมีหลักการทำงานเป็นแบบ ป้อนกลับ (feed back) ซึ่งสถานะของวงจรสามารถเปลี่ยนแปลงได้ด้วยการป้อนสัญญาณทางไฟฟ้า

วงจร Set-Reset Flip flop (SR FF) แบบง่าย โดยใช้ NOR gate



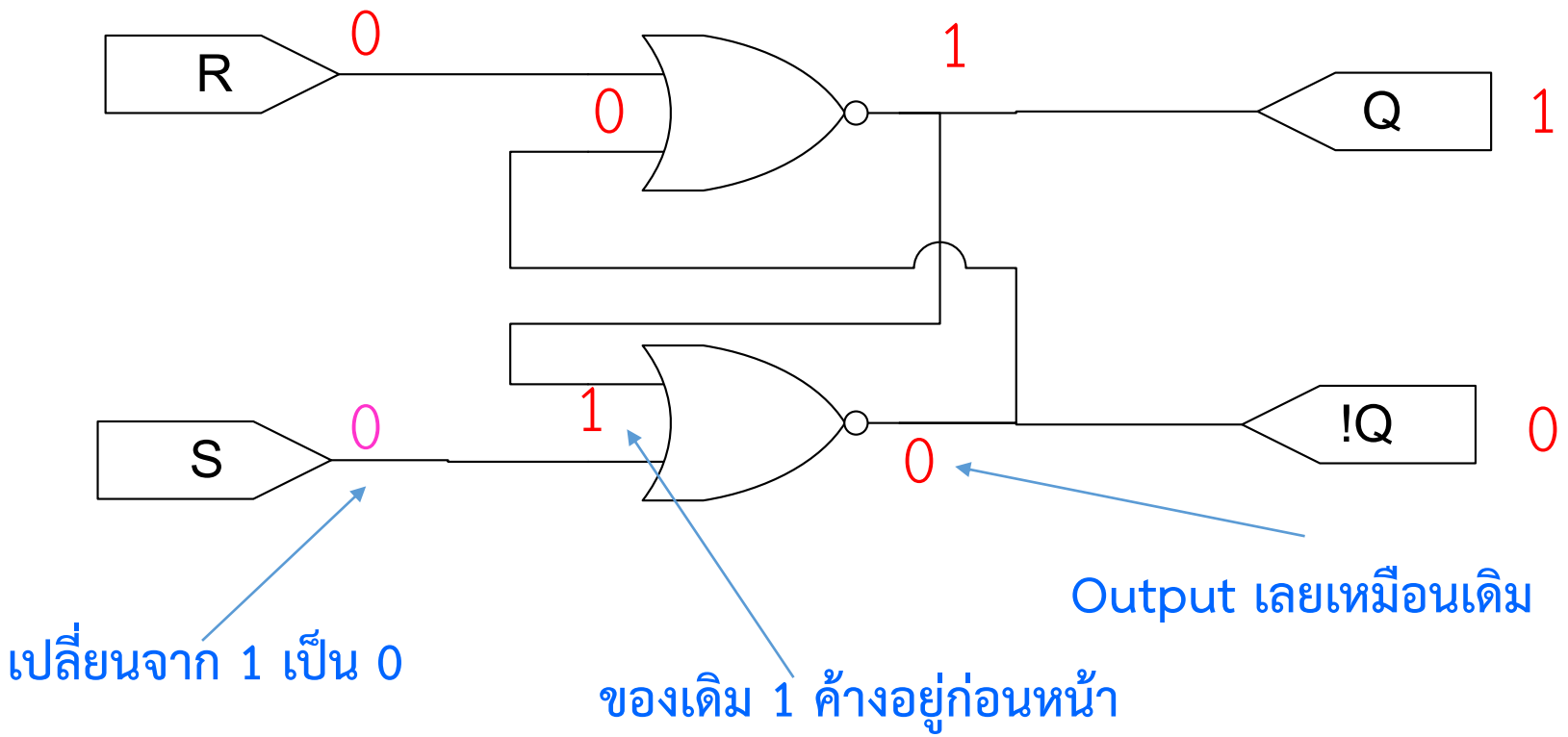
วงจร Set-Reset Flip flop (SR FF) แบบง่าย โดยใช้ NOR gate

การทำให้วงจรมีสถานะ “1” เรียกว่าการ set ทำได้โดยกำหนดให้ S=1 และ R =0



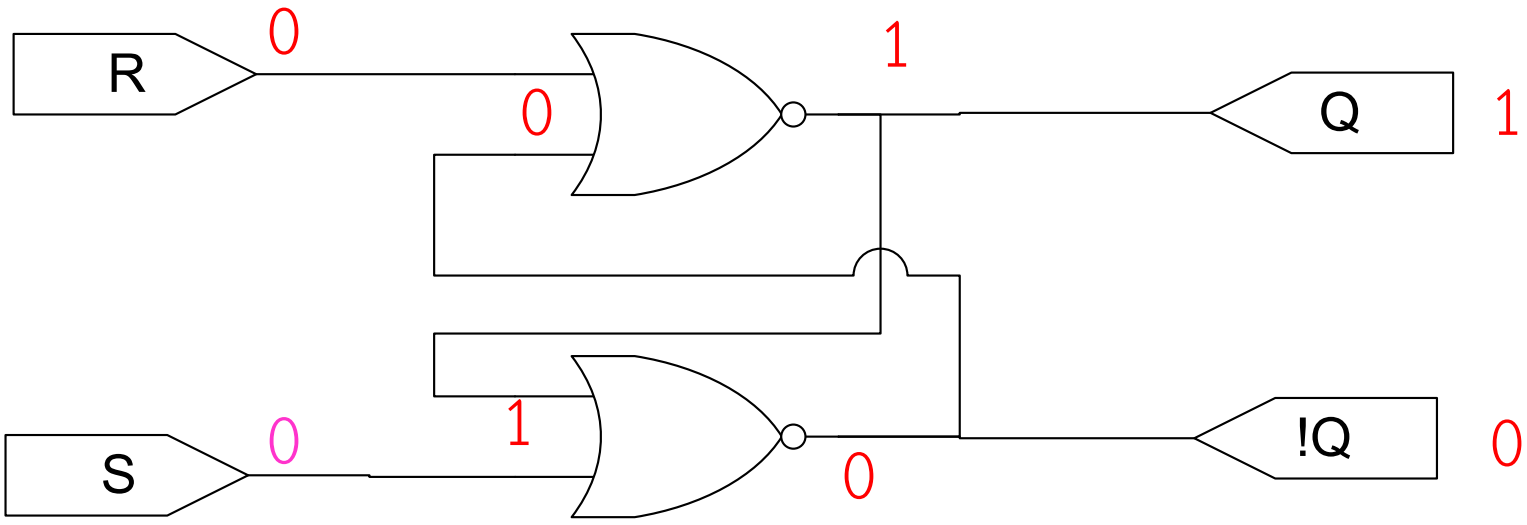
วงจร Set-Reset Flip flop (SR FF) แบบง่าย โดยใช้ NOR gate

เมื่อเปลี่ยน Input เป็น S=0, R=0 วงจรจะคงสถานะเดิมไว้ สถานะนี้เรียกว่า Q



วงจร Set-Reset Flip flop (SR FF) แบบง่าย โดยใช้ NOR gate

เมื่อเปลี่ยน Input เป็น S=0, R=0 วงจรจะคงสถานะเดิมไว้ สถานะนี้เรียกว่า Q

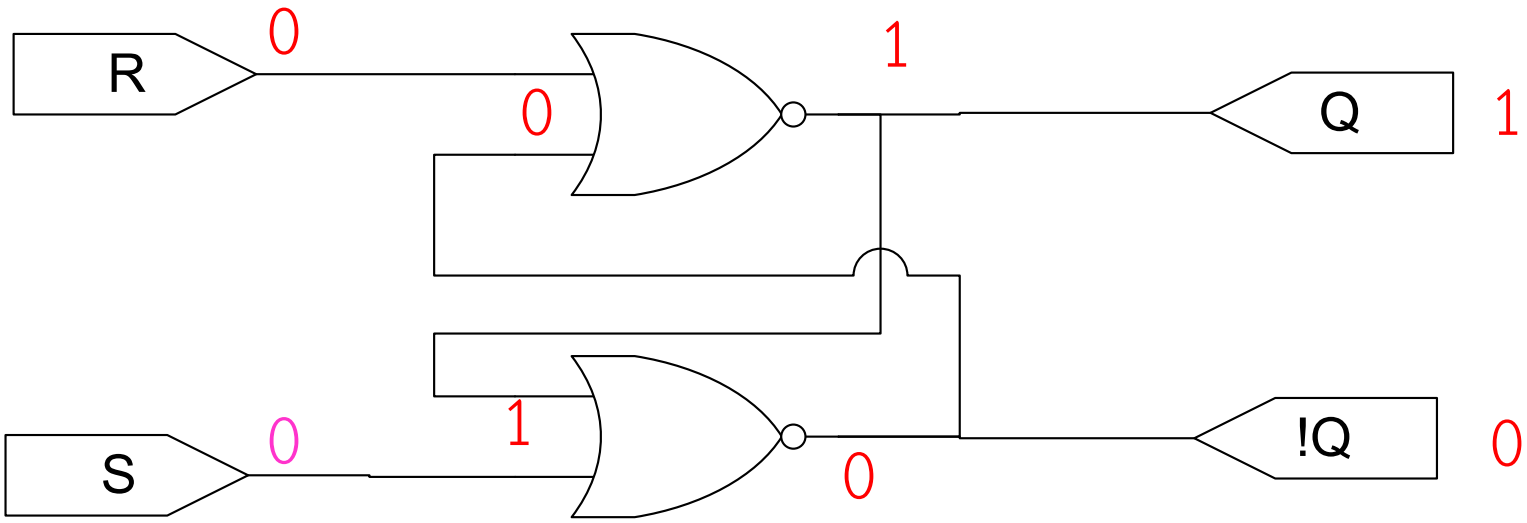


logic ที่เดินทางไปถึง Q จะเรียกว่า Q_{next} หรือ Q ที่จะเกิดขึ้นถัดไป เขียนได้อีกอย่างว่า Q_{n+1}

วงจร Set-Reset Flip flop (SR FF) แบบง่าย โดยใช้ NOR gate

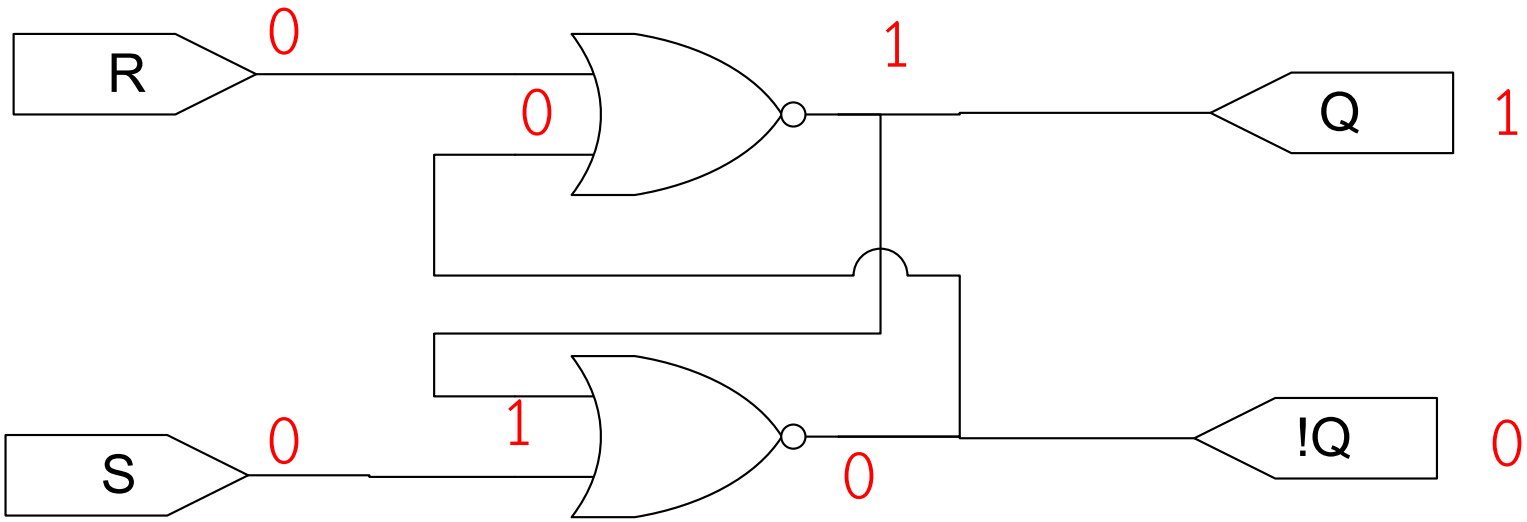
Q ตัวปัจจุบัน มักนิยมเรียกว่า Q_n

ดังนั้นวงจรนี้เมื่อป้อน S=0,R=0 จะได้ว่า $Q_{n+1} = Q_n$



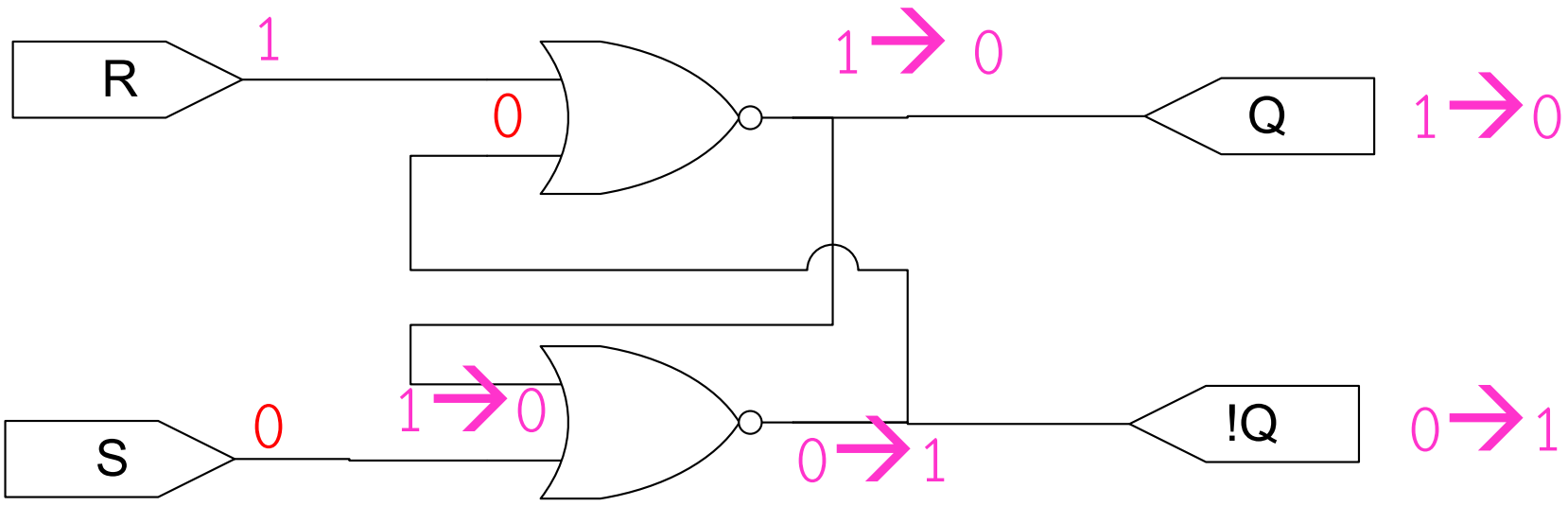
วงจร Set-Reset Flip flop (SR FF) แบบง่าย โดยใช้ NOR gate

การทำให้วงจรมีสถานะเป็น “0” เรียกว่า reset ทำได้โดยกำหนดให้ S=0,R=1



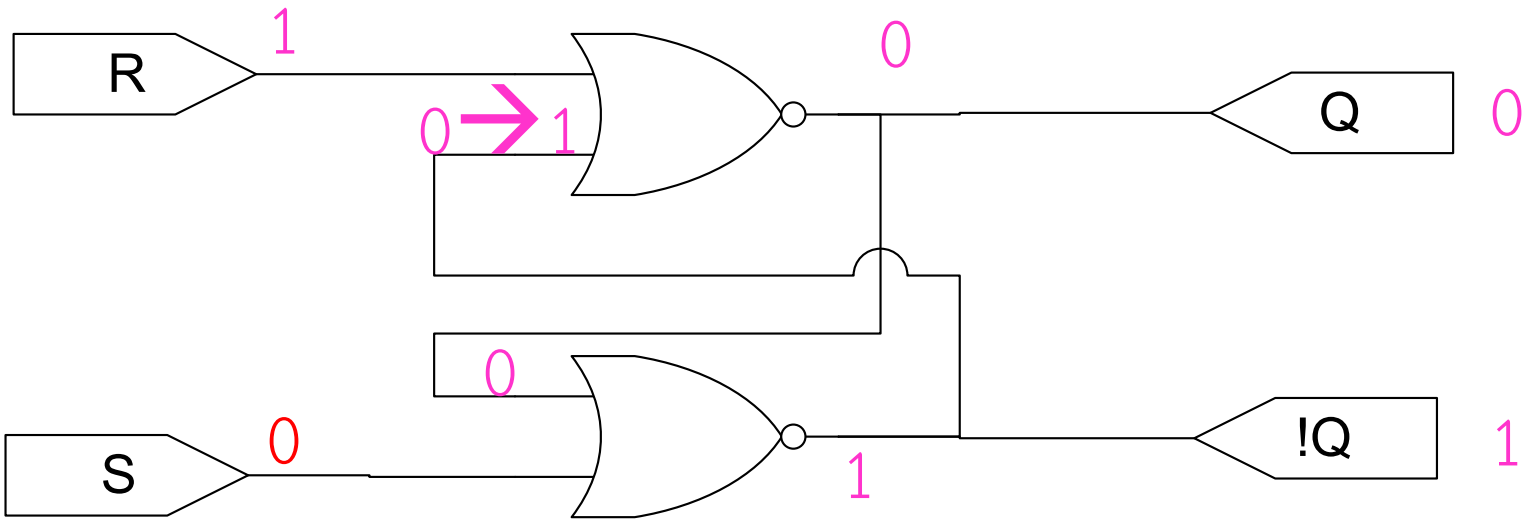
วงจร Set-Reset Flip flop (SR FF) แบบง่าย โดยใช้ NOR gate

การทำให้วงจรมีสถานะเป็น “0” เรียกว่า reset ทำได้โดยกำหนดให้ S=0,R=1



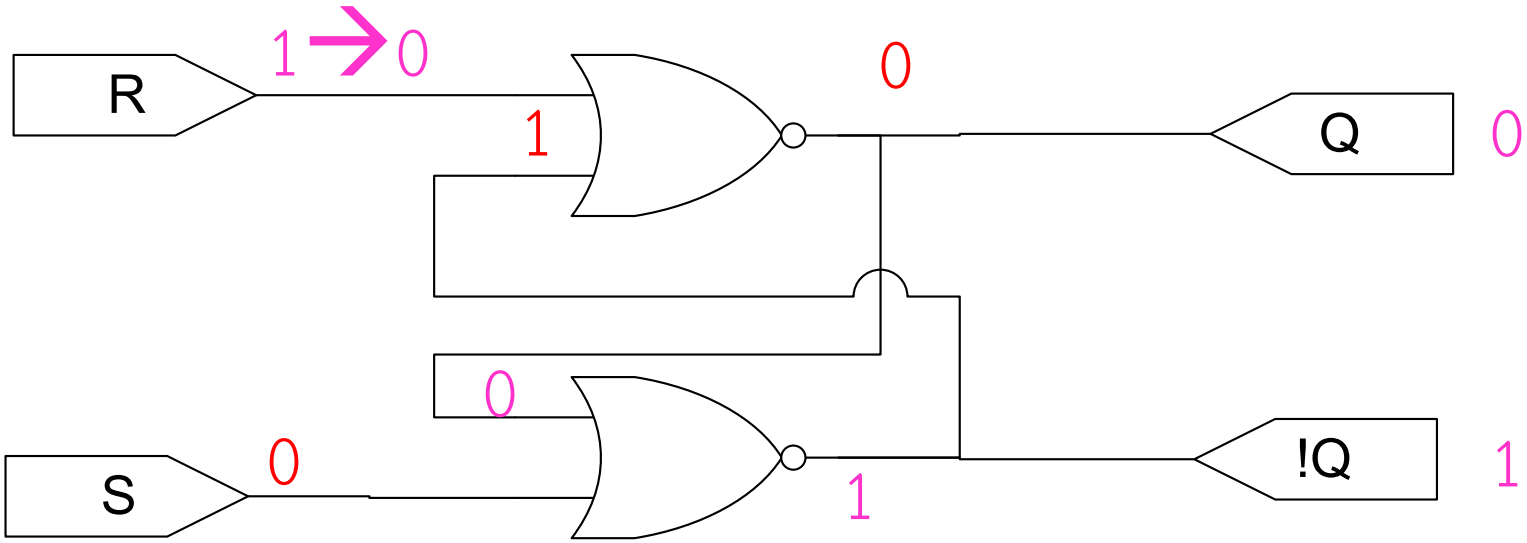
วงจร Set-Reset Flip flop (SR FF) แบบง่าย โดยใช้ NOR gate

การทำให้วงจรมีสถานะเป็น “0” เรียกว่า reset ทำได้โดยกำหนดให้ S=0,R=1



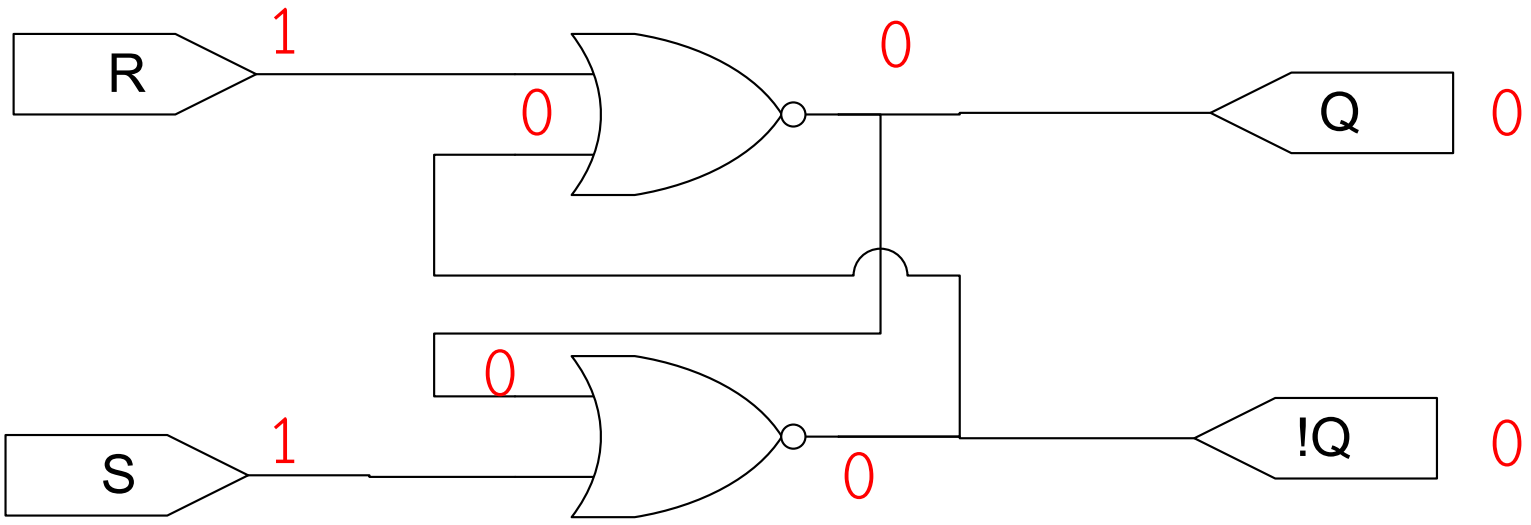
วงจร Set-Reset Flip flop (SR FF) แบบง่าย โดยใช้ NOR gate

เมื่อเปลี่ยน input เป็น S=0,R=0 วงจรจะคงสถานะ



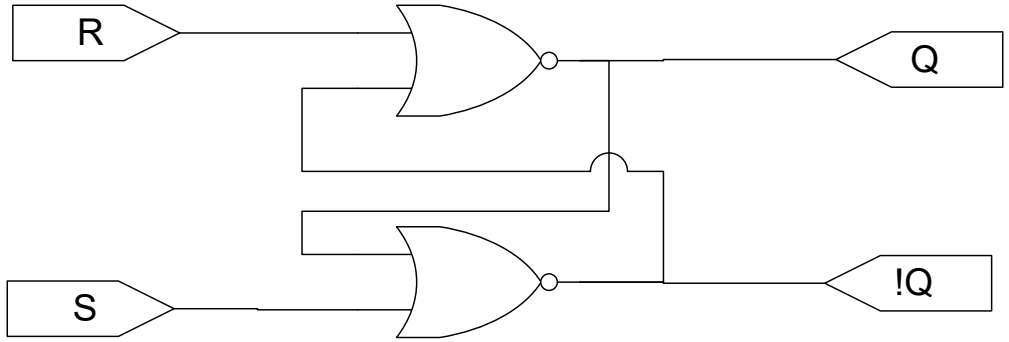
วงจร Set-Reset Flip flop (SR FF) แบบง่าย โดยใช้ NOR gate

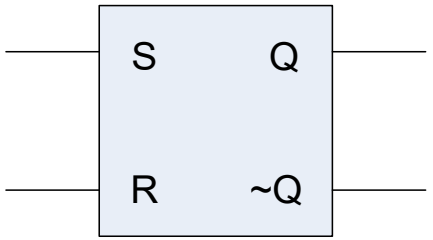
S=1,R=1 วงจรจะเป็นอย่างไร ?



จะเกิด Q = !Q ซึ่งเป็นไปไม่ได้

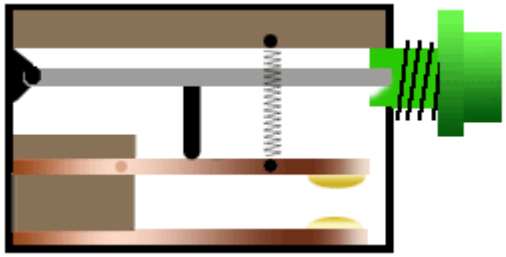
วงจร Set-Reset Flip flop (SR FF) แบบง่าย โดยใช้ NOR gate

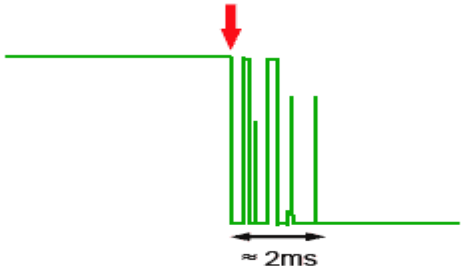




Input		Output	ความหมาย
S	R	Q_{n+1}	
0	0	Q_n	ไม่เปลี่ยนแปลงสถานะ
0	1	0	เปลี่ยนเป็น 0
1	0	1	เปลี่ยนเป็น 1
1	1	ไม่นิยาม	ห้ามป้อน

วงจรควบคุมมอเตอร์



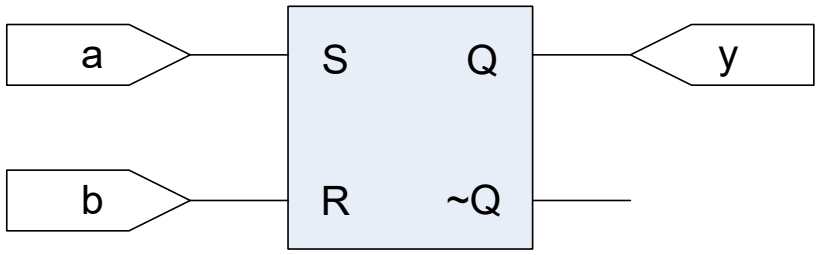


สวิตซ์แบบกดปุ่มโดยทั่วไป ไม่สามารถนำมาใช้เปิด และ ปิดมอเตอร์ไฟฟ้าได้ เนื่องจากอาจเกิดการกระดอน (Switch Bounce) ซึ่งอาจทำให้มอเตอร์เกิดความเสียหาย หรือสวิตซ์อาจเกิดการอาร์คหรือลุกไหม้ได้

วงจรควบคุมมอเตอร์

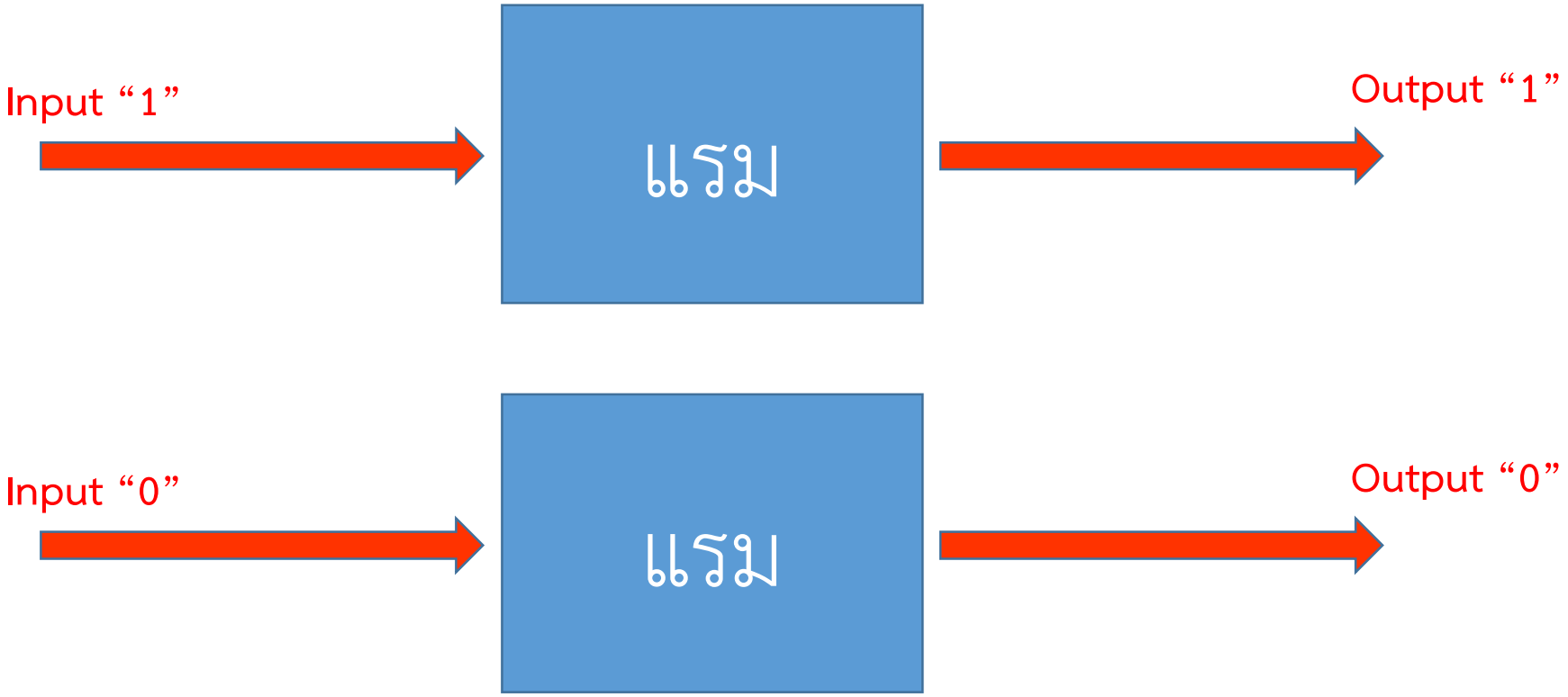
แผงควบคุมมอเตอร์ ประกอบด้วยสวิตซ์แบบ กดติด ปล่อยดับ จำนวน 2 ปุ่ม โดยกำหมดให้ มอเตอร์ทำงานเมื่อกดปุ่มสีเขียว เพียงหนึ่งครั้ง และหยุดทำงาน เมื่อกดปุ่มสำเขียวแดง 1 ครั้ง

Input		Output
ปุ่มสีเขียว(a)	ปุ่มสีแดง(b)	มอเตอร์(y)
0	0	ไม่เปลี่ยนแปลง
0	1	0
1	0	1
1	1	ไม่นิยาม



RS Flip flop สามารถใช้เป็นหน่วยความจำได้หรือไม่ ?

ตอบ: ไม่เหมาะ เนื่องจากไม่สามารถป้อนข้อมูลเป็น 1 หรือ 0 ให้มันจำได้

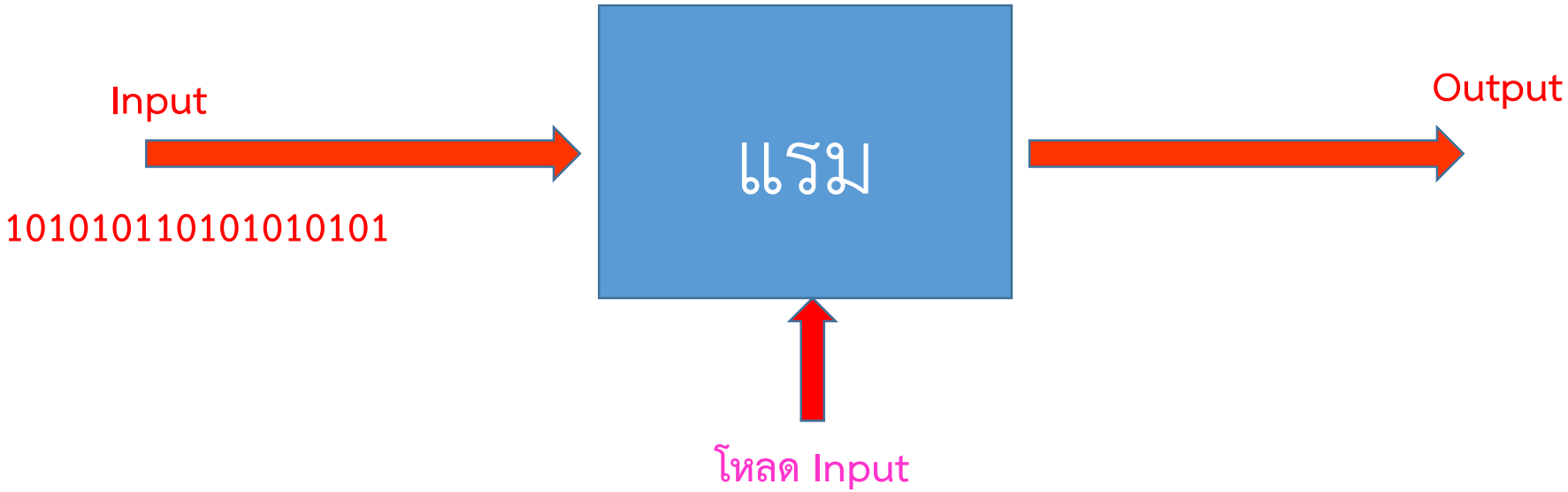


RS Flip flop ทำแบบนี้ไม่ได้

ฟลิปฟล็อป

แรมจะต้องสามารถโหลดข้อมูลเข้า และคงข้อมูลที่โหลดได้

ต้องระบุจังหวะที่ต้องการโหลดได้ เพื่อป้องกันไม่ให้ข้อมูลในแรมเปลี่ยนไป หาก input นั้นเปลี่ยนเป็นอย่างอื่นไปแล้ว

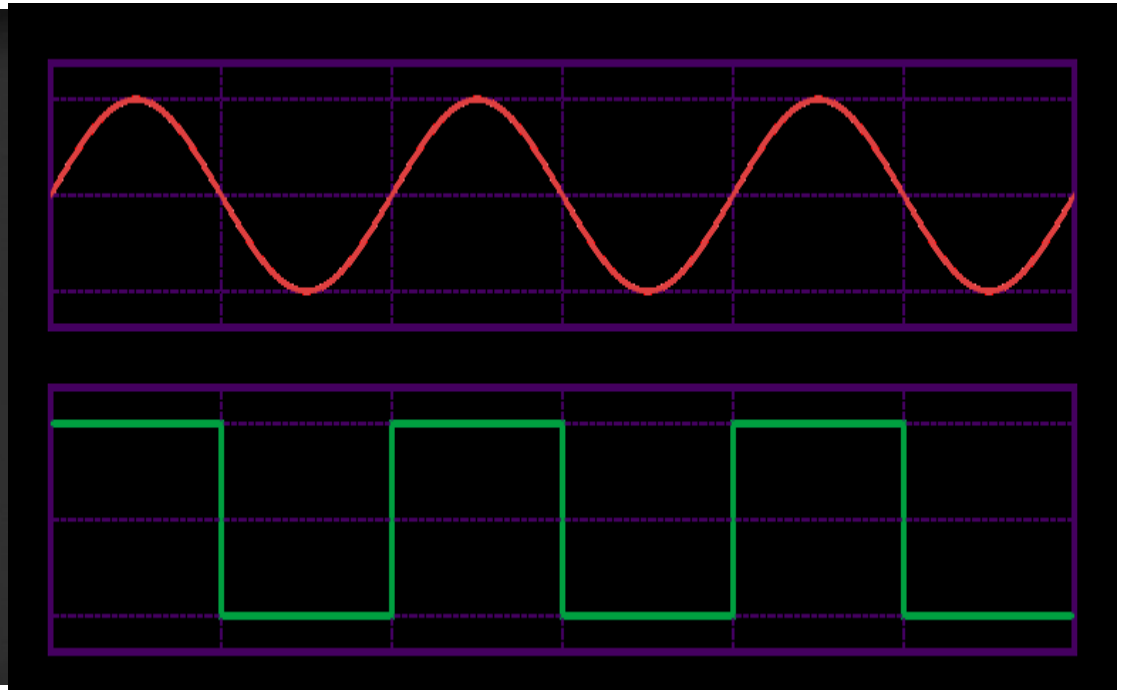


จังหวะของการโหลดนั้น จะต้องเกิดขึ้นและจบลงอย่างรวดเร็วในช่วงเวลาจำกัด เวลาหนึ่ง

สัญญาณที่ใช้กำหนดจังหวะคือสัญญาณนาฬิกา (Clock)

สัญญาณนาฬิกา

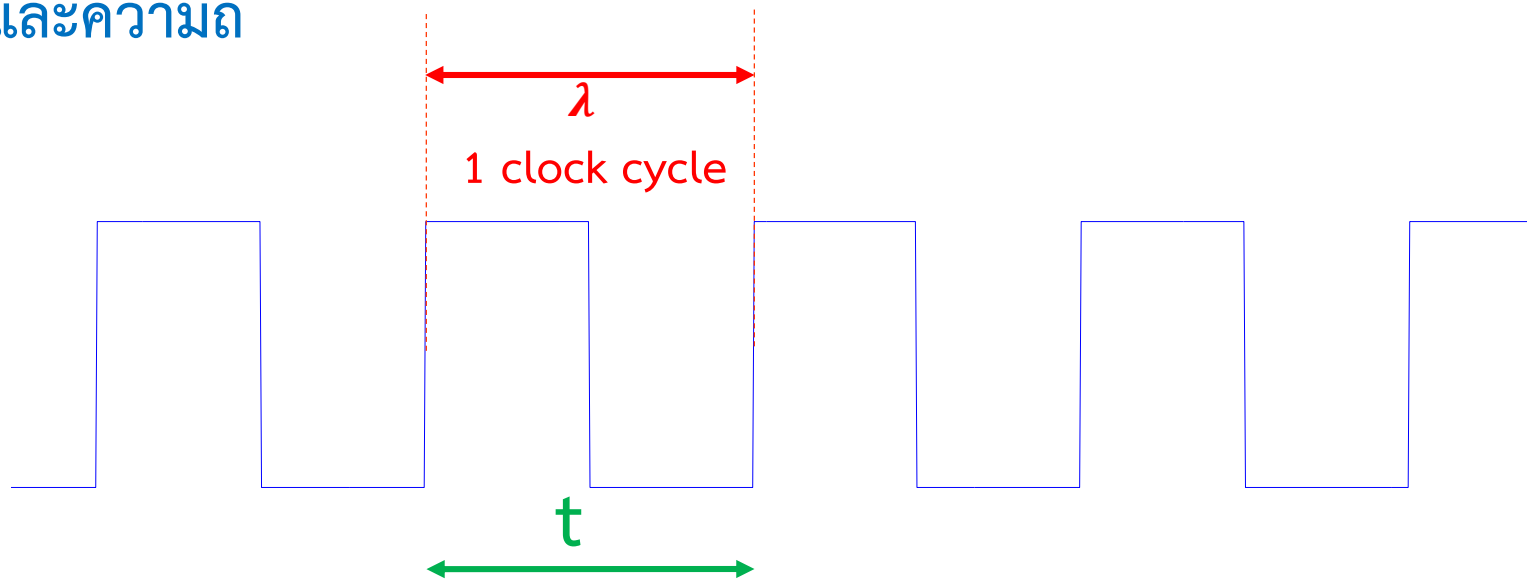
สัญญาณนาฬิกา (Clock signal) คือสัญญาณที่มีการเกิดซ้ำๆระหว่างค่าสูงและต่ำ ที่ถูกใช้ในการกำหนดจังหวะการทำงานของวงจรดิจิทัล โดยทั่วไปนิยมใช้สัญญาณ คลื่นจัตุรัส (square wave) ที่มีรอบการทำงาน (duty cycle) 50%



λ

1 clock cycle

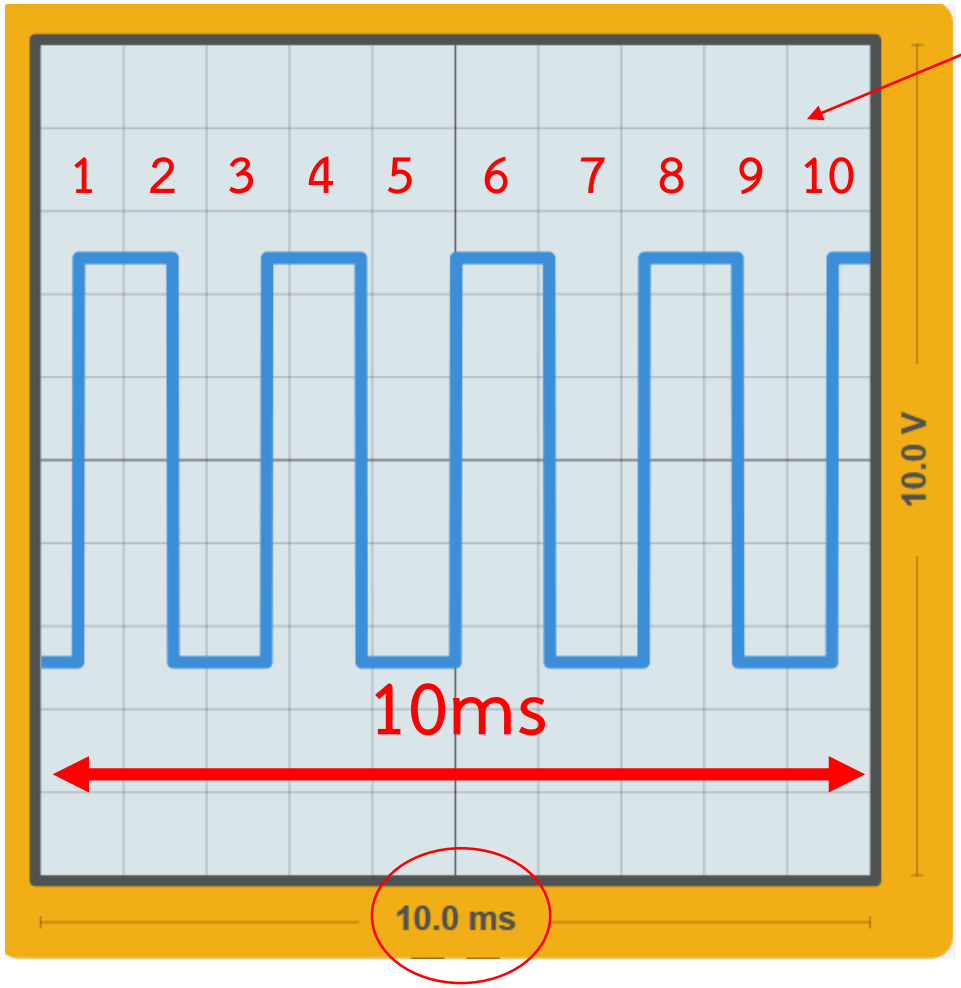
คาบเวลาและความถี่



สัญญาณนาฬิกา จะต้องเป็นสัญญาณที่เกิดซ้ำๆกัน ส่วนเล็กที่สุดที่ซ้ำกัน เรียกว่า “cycle”

ความกว้างของ cycle วัดในหน่วยเวลา เรียกว่า “คาบเวลา” หรือ t

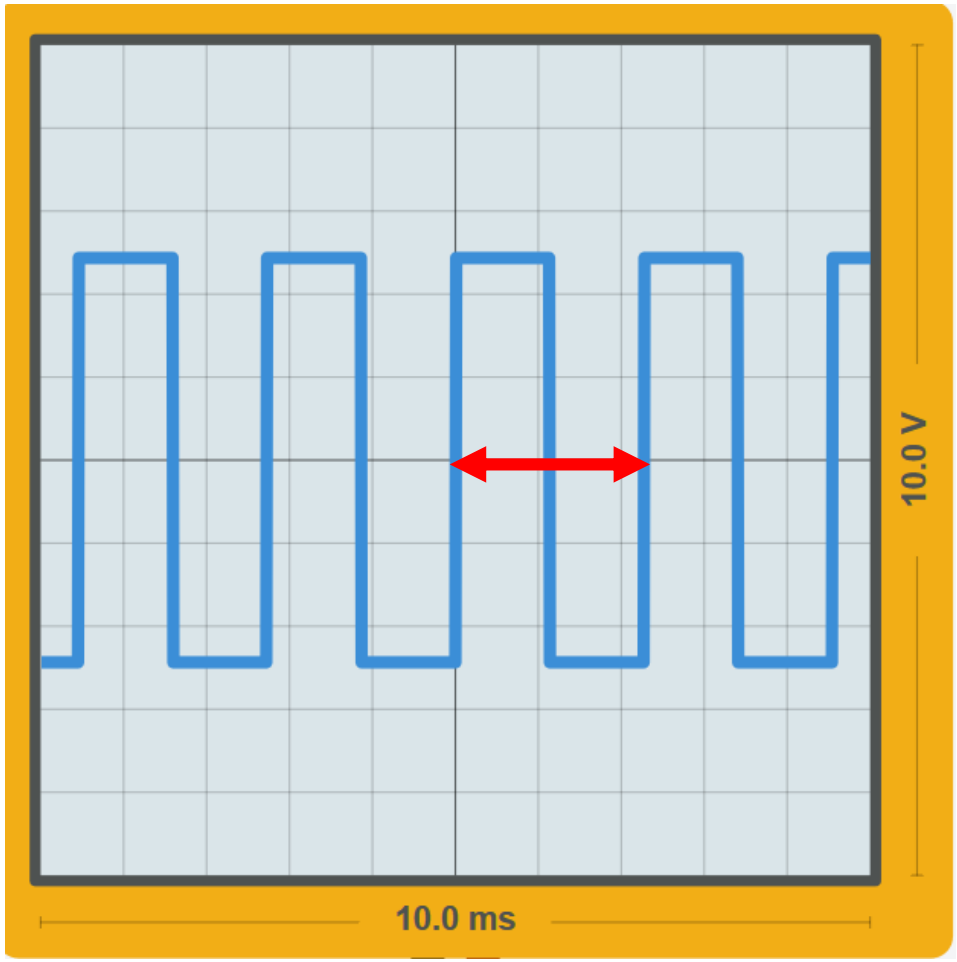
สัญญาณนี้มีค่าคาบเวลาเท่าใด ?



1 หาค่าคาบเวลาของ 1 ช่อง

ดังนั้น 1 ช่องจะกว้าง $\frac{10m}{10} = \frac{0.01}{10} = .001 = 1m$

สัญญาณนี้มีค่าคาบเวลาเท่าใด ?

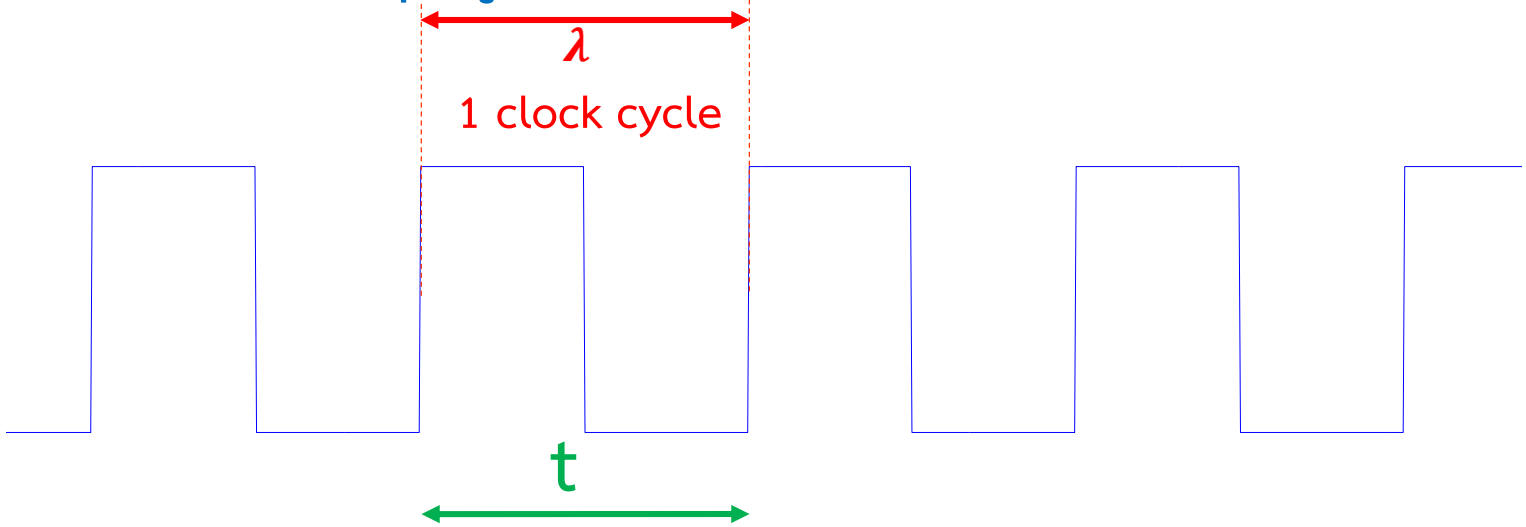


ประมาณ 2.27 ช่อง

2 หาค่าคาบเวลาของสัญญาณ

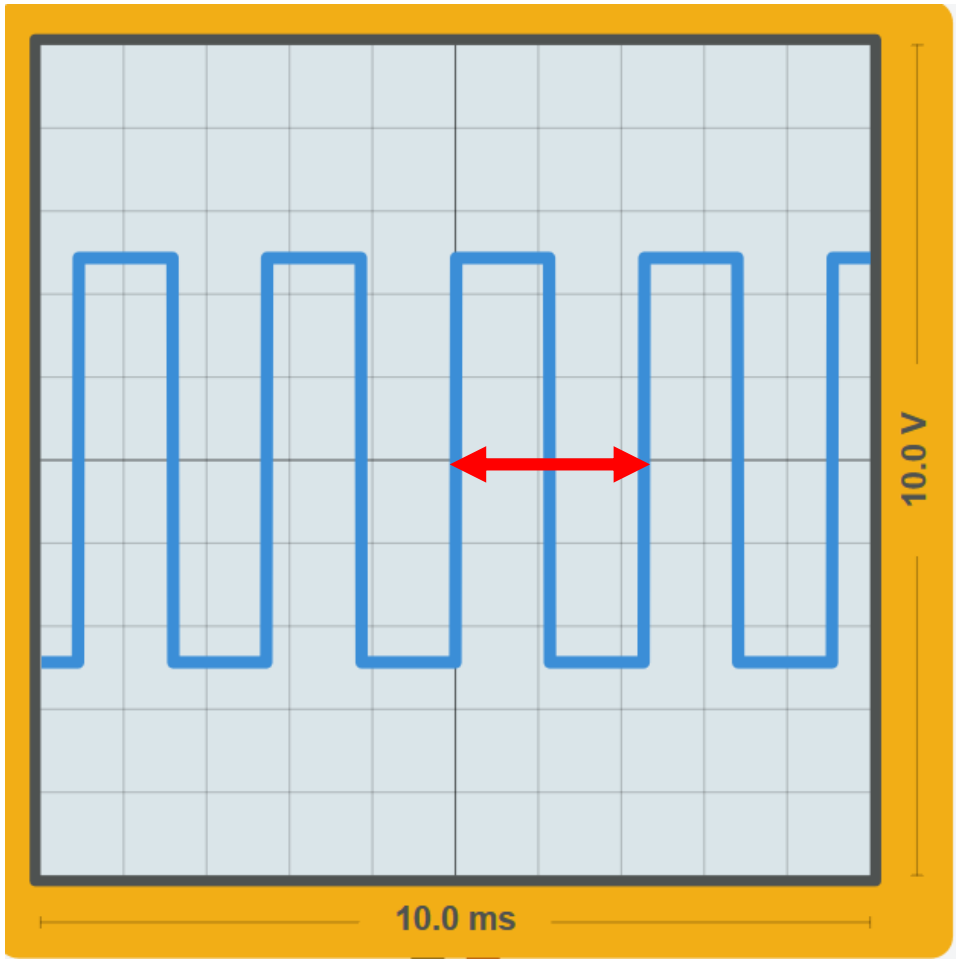
ดังนั้นคาบเวลาของสัญญาณนี้คือ $2.25 \times 1m = 2.27m = 0.00227$ วินาที

สัญญาณนาฬิกามักนิยมระบุในรูปแบบของความถี่



ความถี่ (f) คำนวณได้จาก $$f = \frac{1}{t}$$

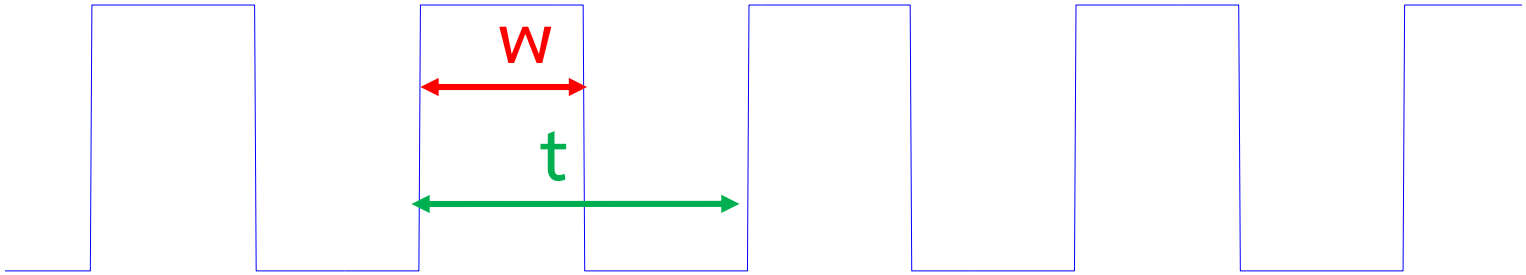
สัญญาณนี้มีความถี่เท่าใด ?



2.27ms

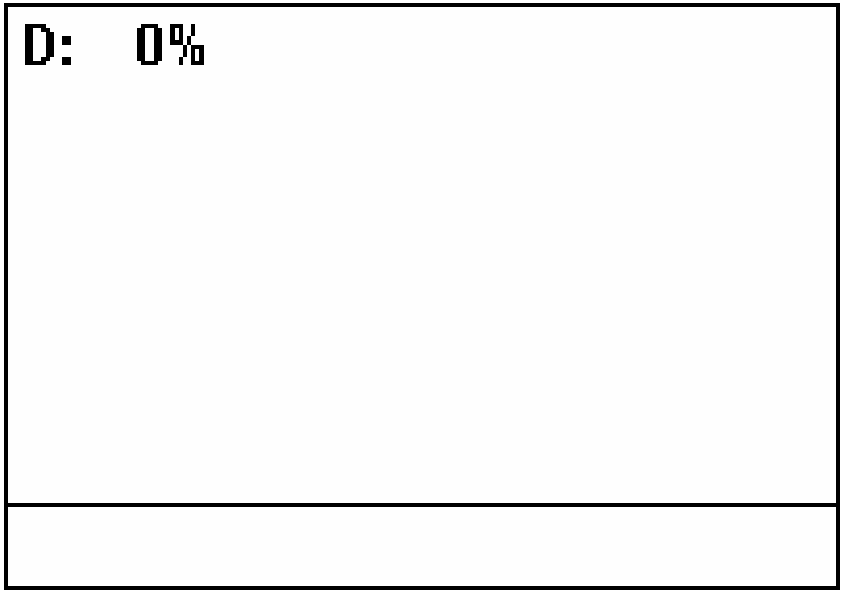
$$f = \frac{1}{t} = \frac{1}{0.00227} = 440.5\ Hz$$

อัตราส่วนระหว่างส่วนของความกว้างของ pulse ต่อคาบเวลาเรียกว่าค่า duty cycle

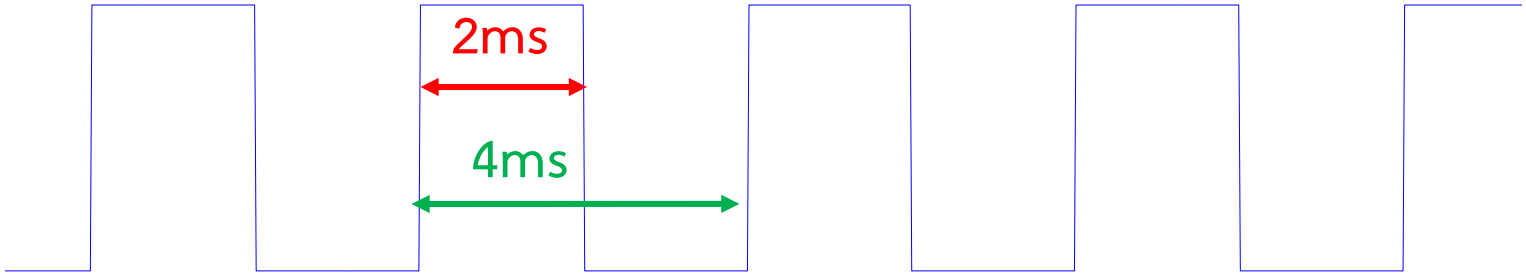


duty cycle (d) สามารถคำนวณได้จาก

$$d = \frac{w}{t} \times 100$$



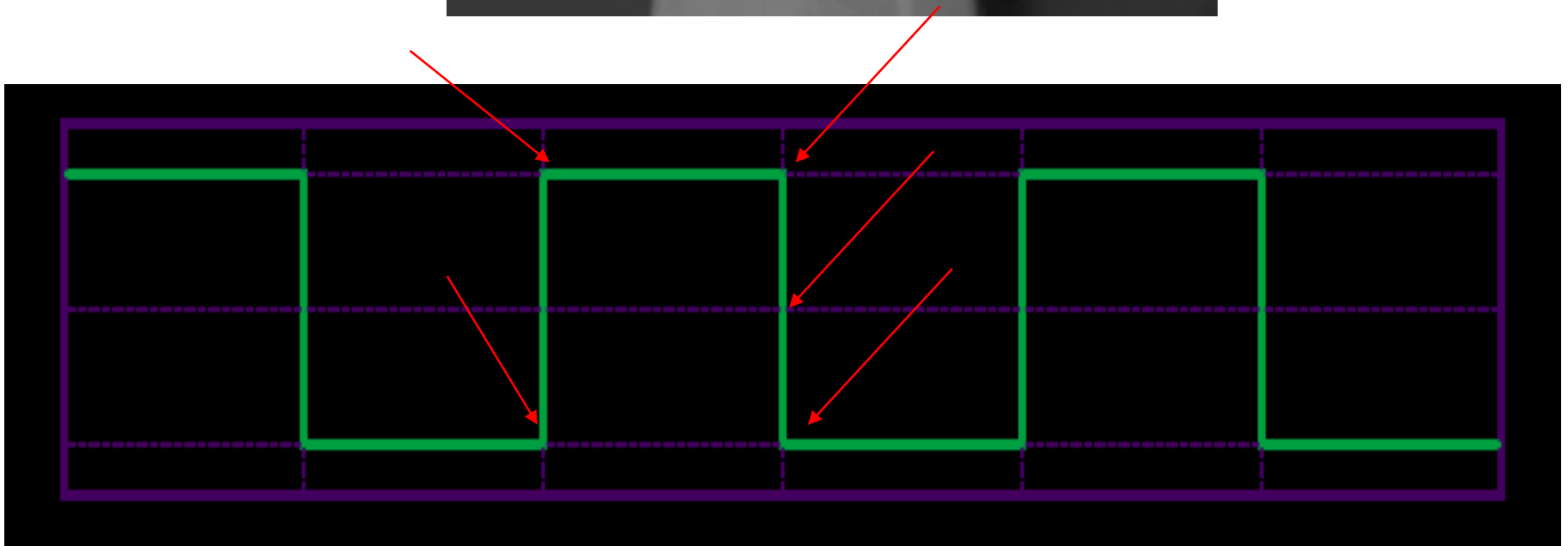
สัญญาณนี้มีค่า duty cycle เท่าใด ?



$$d = \frac{w}{t} \times 100 = \frac{2}{4} \times 100 = 50$$

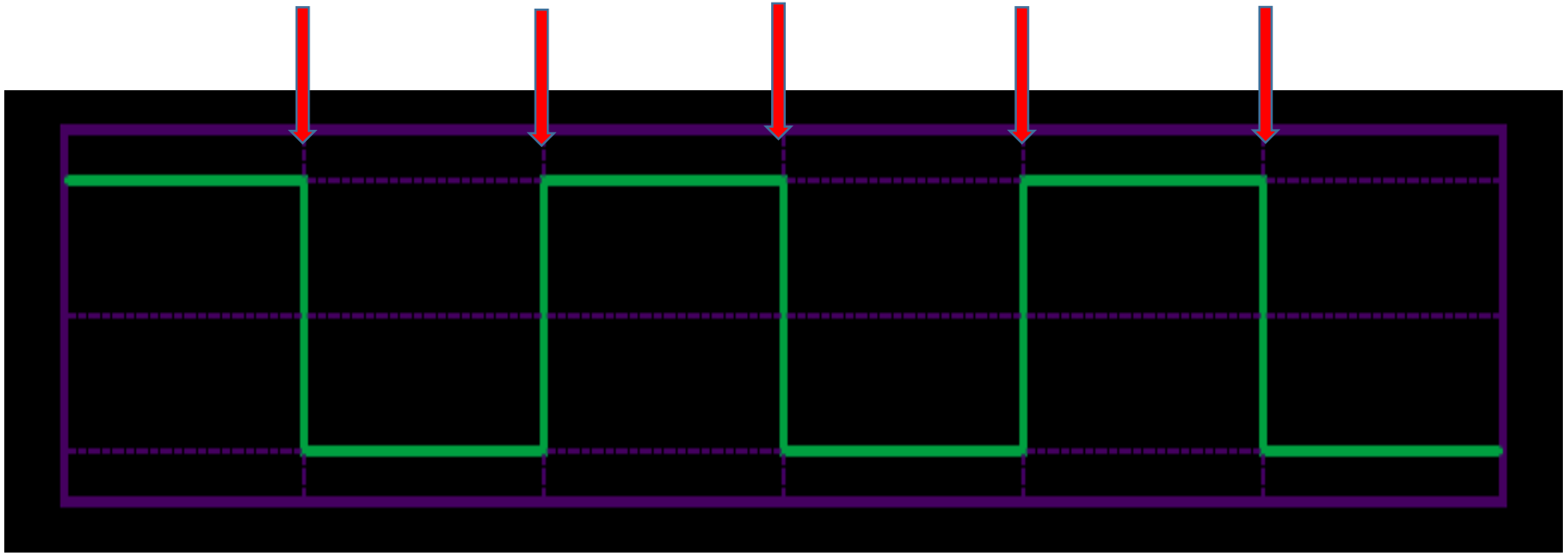
ตอบ 50%

วงจรจะใช้สัญญาณส่วนไหนในการเข้าจังหวะ ?



โดยทั่วไปแล้ว เรานิยมใช้ขอบ(edge)ของสัญญาณในการกระตุ้น(trig)การทำงานของวงจร

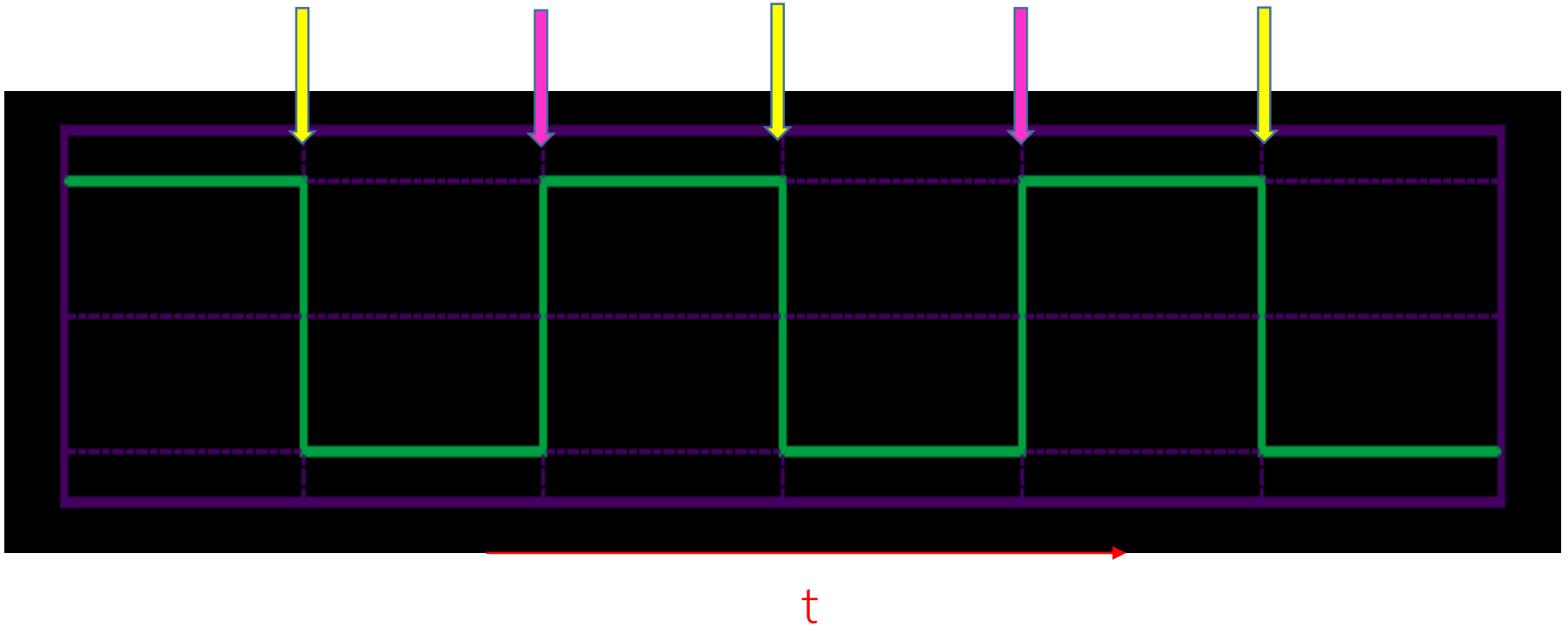
ขอบ (edge) คือตำแหน่งที่สัญญาณมีการเปลี่ยนแปลงระดับ เช่น 0 เปลี่ยนเป็น 1 หรือ 1 เปลี่ยนเป็น 0



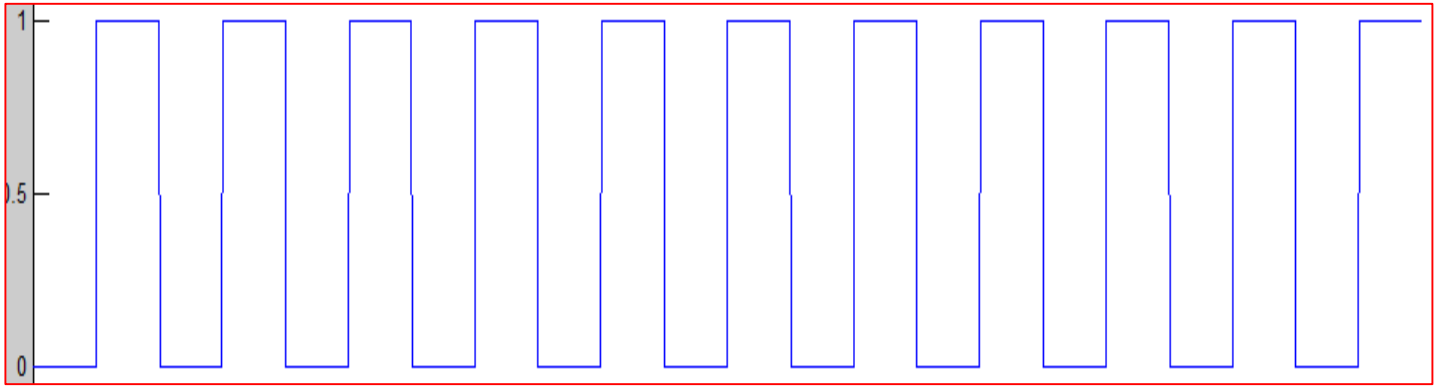
ขอบ (edge)

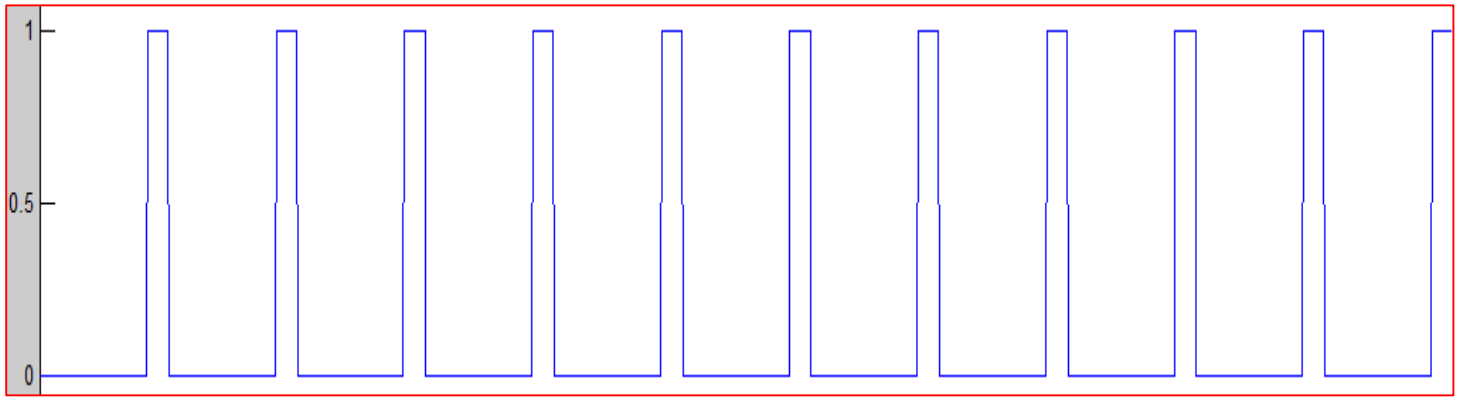
ขอบที่เกิดจากการเปลี่ยนระดับจาก 0 เป็น 1 เรียกว่า Positive edge

1 เป็น 0 เรียกว่า Negative edge



Positive pulse and Negative pulse

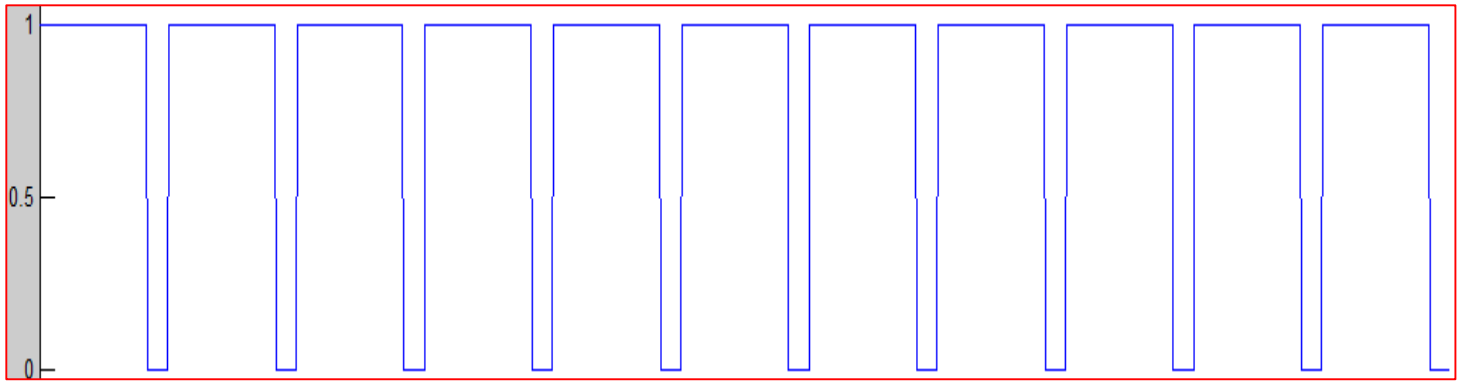




Positive pulse

แรงดันต่ำ=0

แรงดันสูง = 1

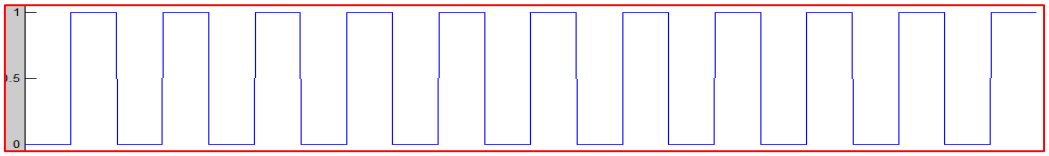


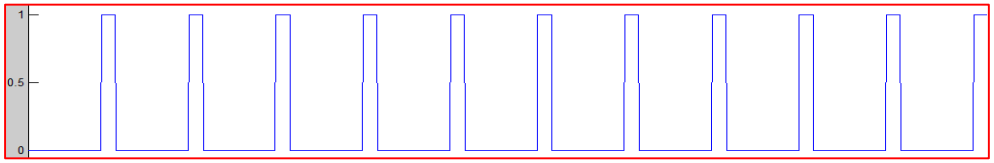
Negative pulse

แรงดันสูง=0

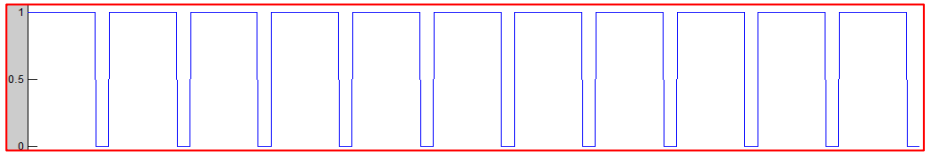
แรงดันต่ำ = 1

ในชีวิตจริงเราจะใช้ Pulse ทั้งสองแบบปนๆกัน

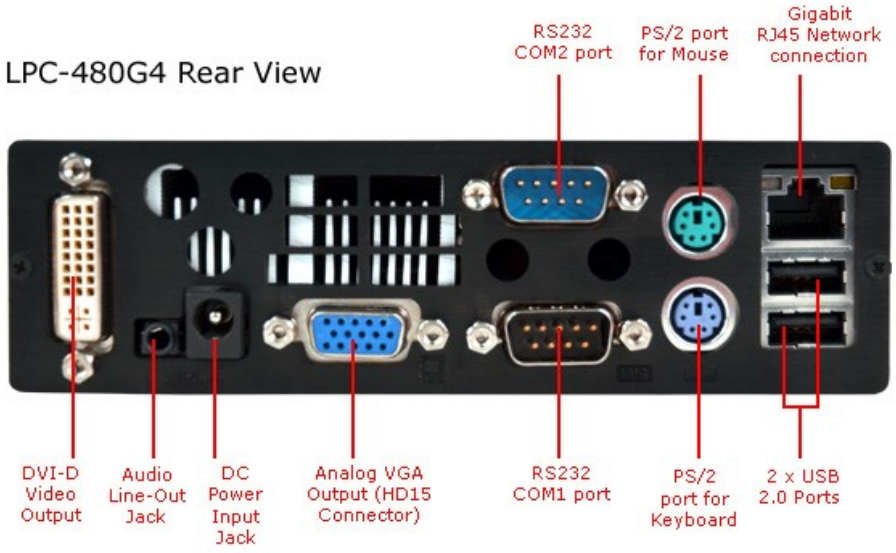




Positive pulse



Negative pulse



Positive edge และ Negative edge ของสัญญาณทั้ง 2 แบบจะตรงข้ามกัน

ถ้าใช้นิยามนี้ในการออกแบบวงจรที่มีสัญญาณ pulse ทั้ง 2 แบบปนๆกัน คงมึนน่าดู

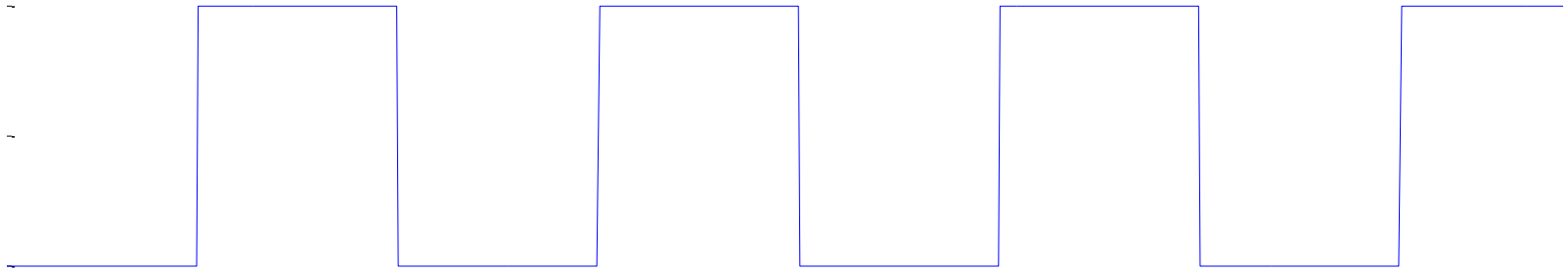
อีกวิธีในการนิยามตำแหน่งของ pulse คือ leading edge และ trailing edge

Trailing edge คือการเปลี่ยนระดับสัญญาณครั้งแรก

Leading edge คือการเปลี่ยนระดับสัญญญาณครั้งที่ 2

โดยจะเปลี่ยนจาก 0 ไป 1 หรือ 1 ไป 0 ก็ได้

การเปลี่ยนแปลงนั้นนับจากจุดเริ่มต้น t0



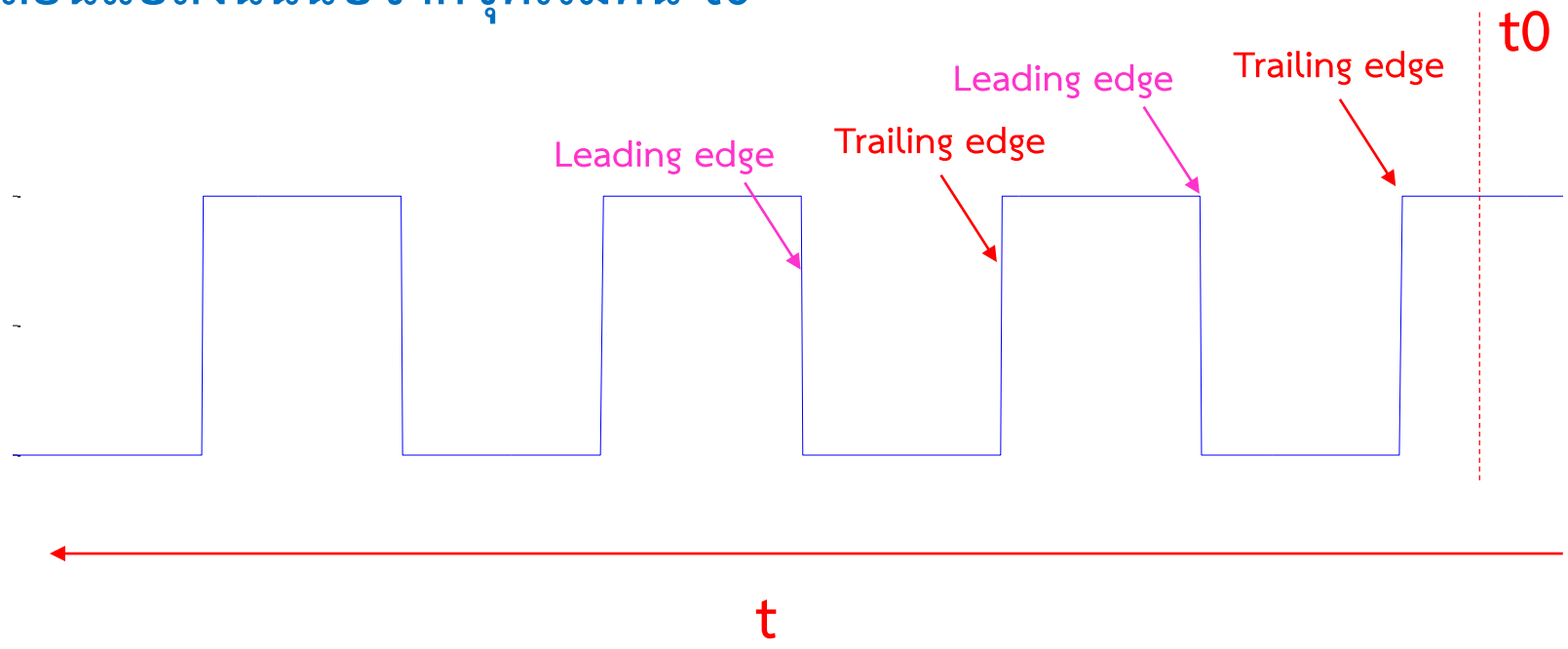
อีกวิธีในการนิยามตำแหน่งของ pulse คือ leading edge และ trailing edge

Trailing edge คือการเปลี่ยนระดับสัญญาณครั้งแรก

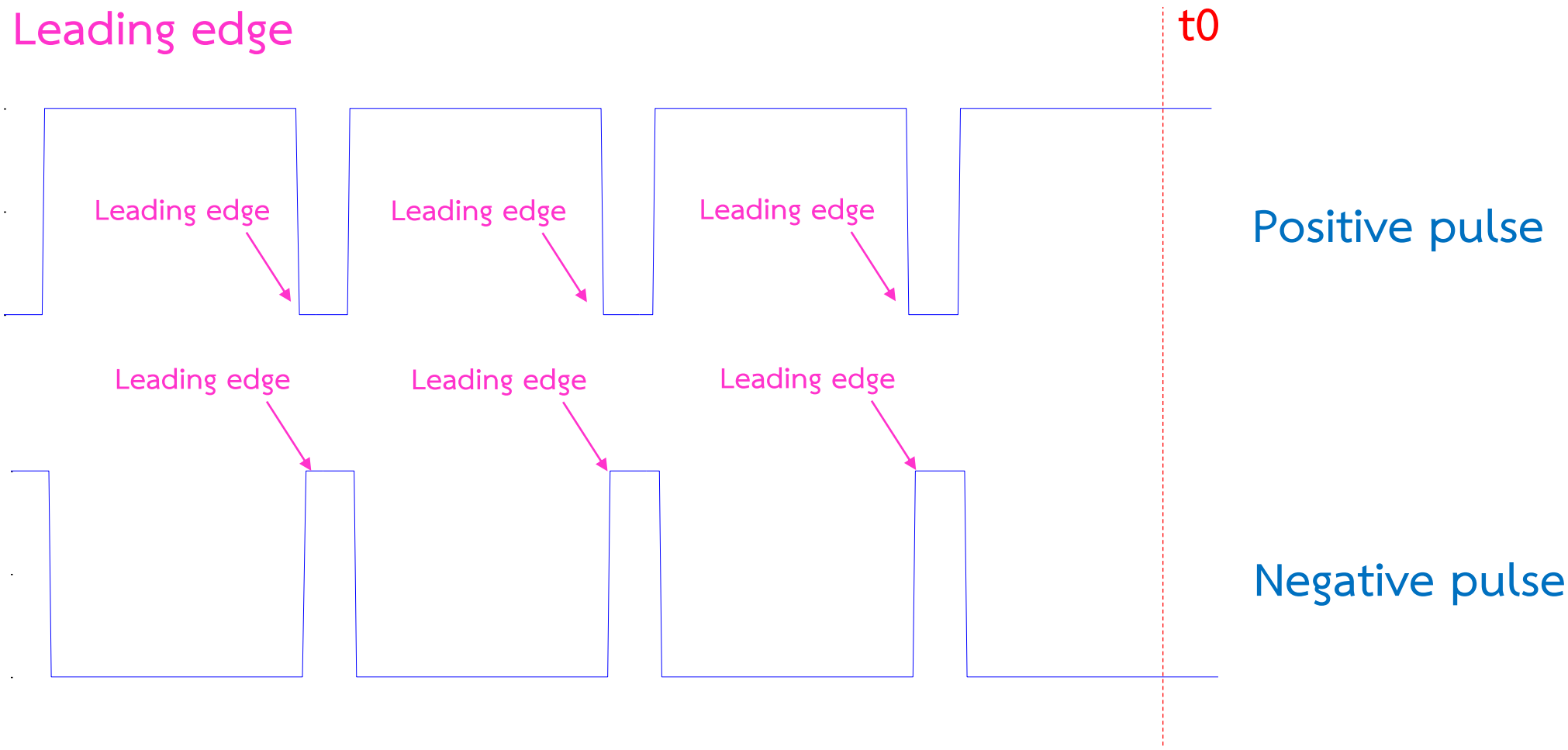
Leading edge คือการเปลี่ยนระดับสัญญาณครั้งที่ 2

โดยจะเปลี่ยนจาก 0 ไป 1 หรือ 1 ไป 0 ก็ได้

การเปลี่ยนแปลงนั้นนับจากจุดเริ่มต้น t0

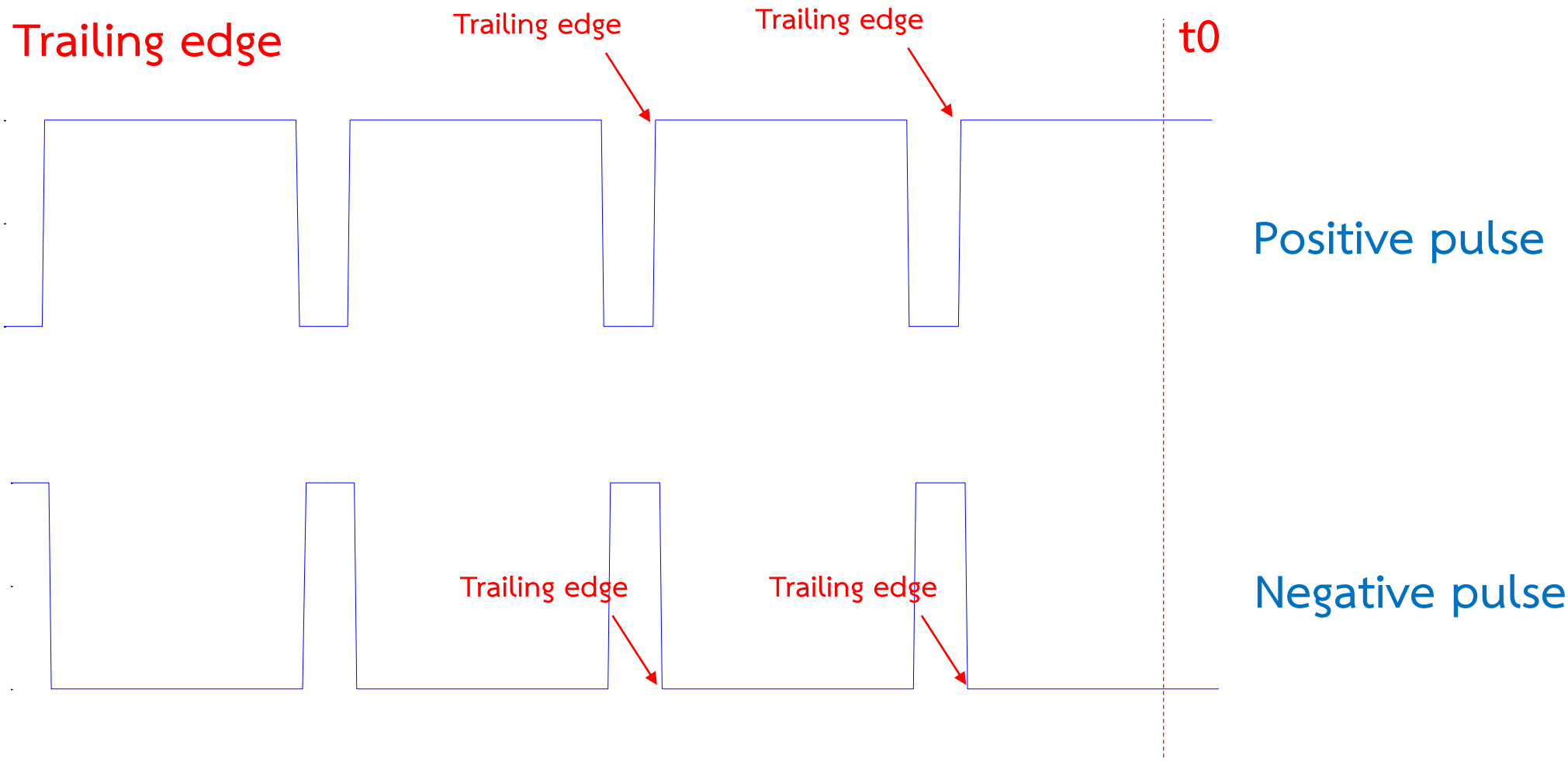


สัญญาณ Positive pulse และ Negative pulse



Leading edge อยู่ตำแหน่งเดียวกัน

สัญญาณ Positive pulse และ Negative pulse

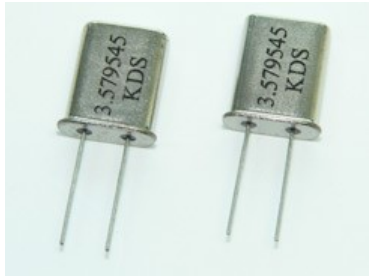


Trailing edge อยู่ตำแหน่งเดียวกัน

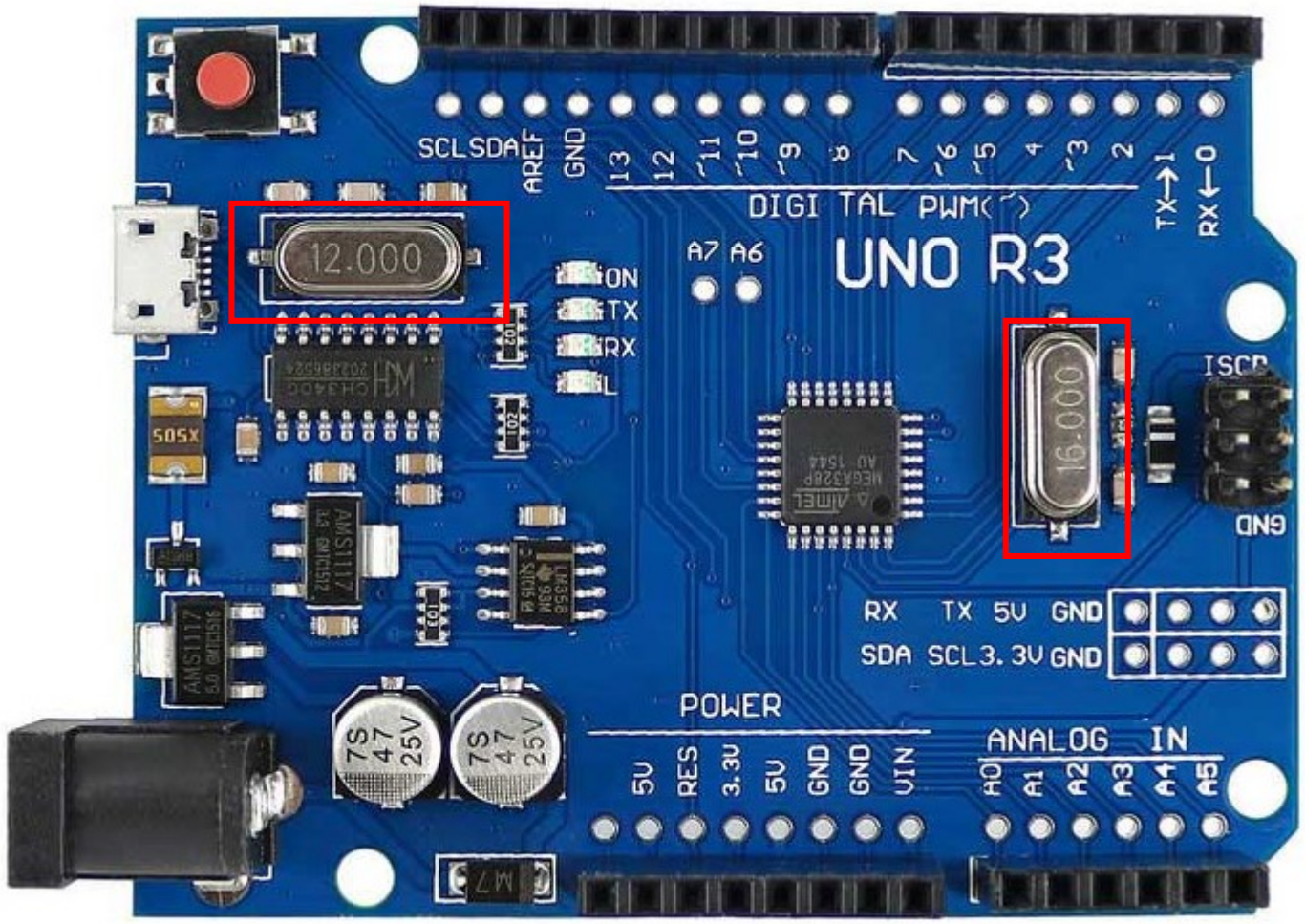
วิธีการสร้างสัญญาณนาฬิกา

1) ใช้สวิตซ์ (switch)

2) ใช้วงจรเอนกระรัว (Multi-vibrator circuit)

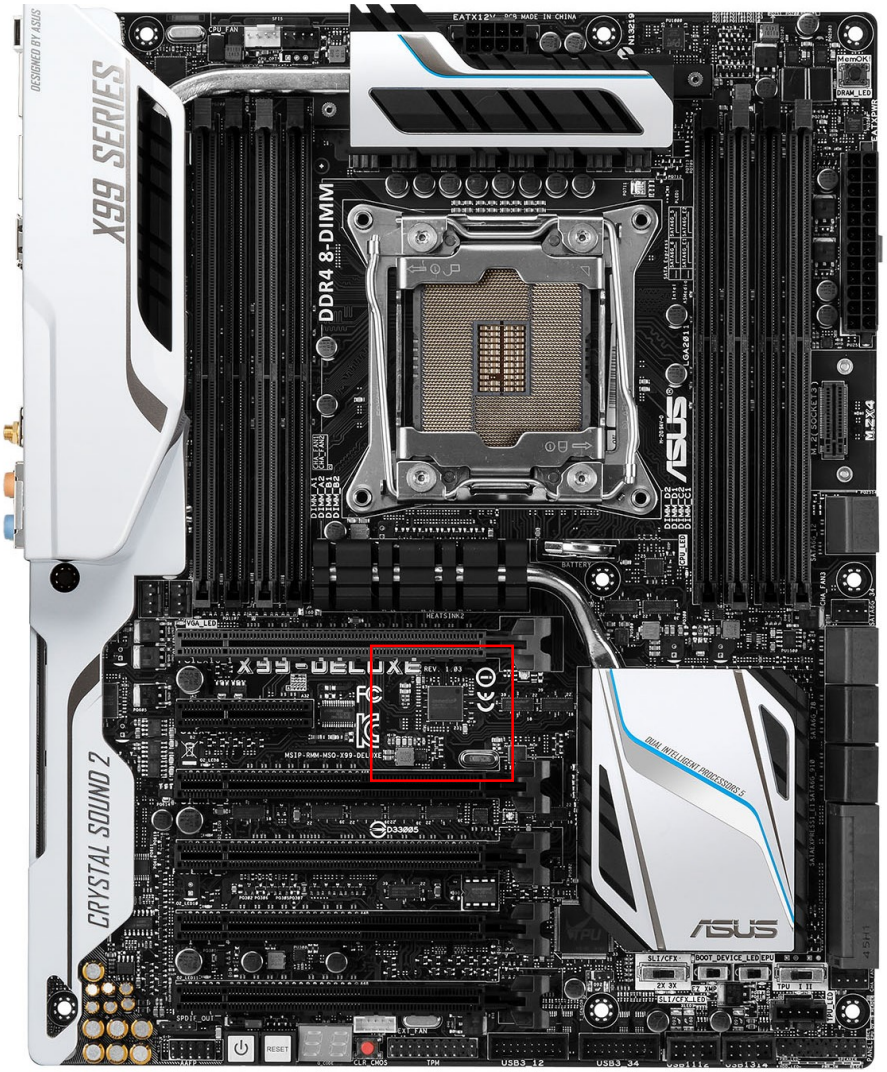


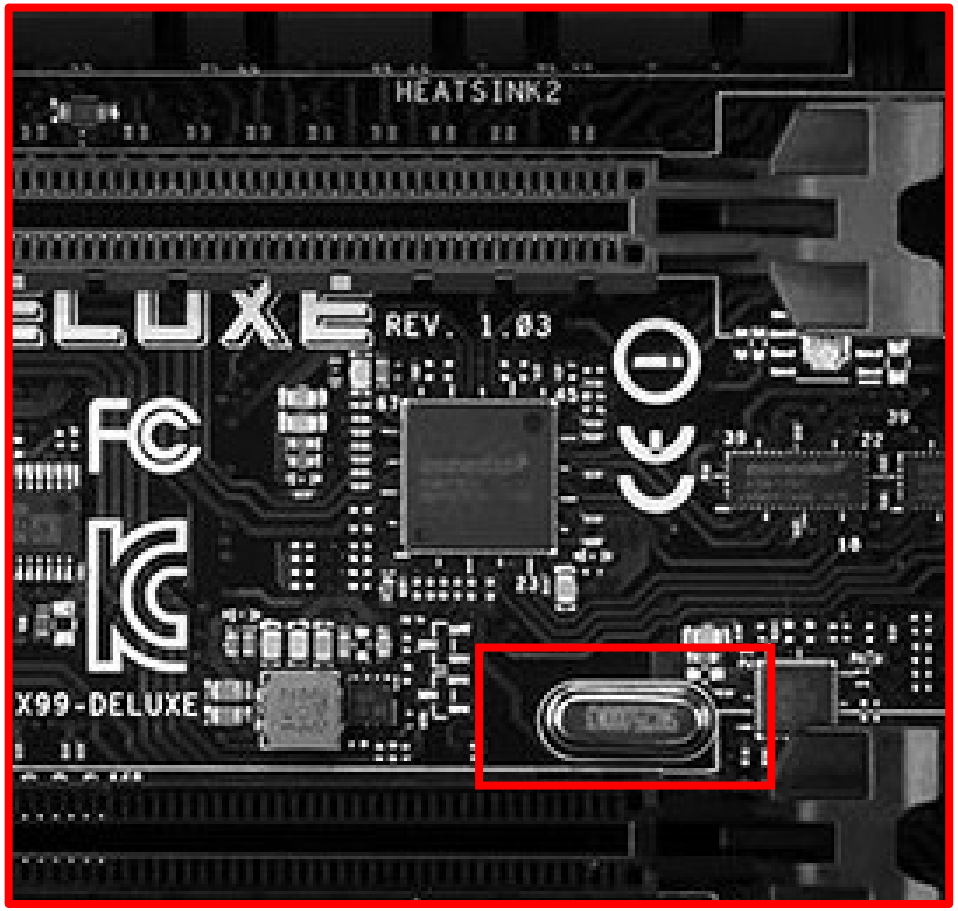
ใช้วงจรเอนกระรัว (Multi-vibrator circuit) มีการทำงานโดยอาศัยการป้อนกลับของสัญญาณ เพื่อให้เกิดการ oscillation ขึ้นโดยใช้ตัวกรองความถี่เข้ามาช่วยทำให้เกิดการ oscillation ในความถี่ที่ต้องการ



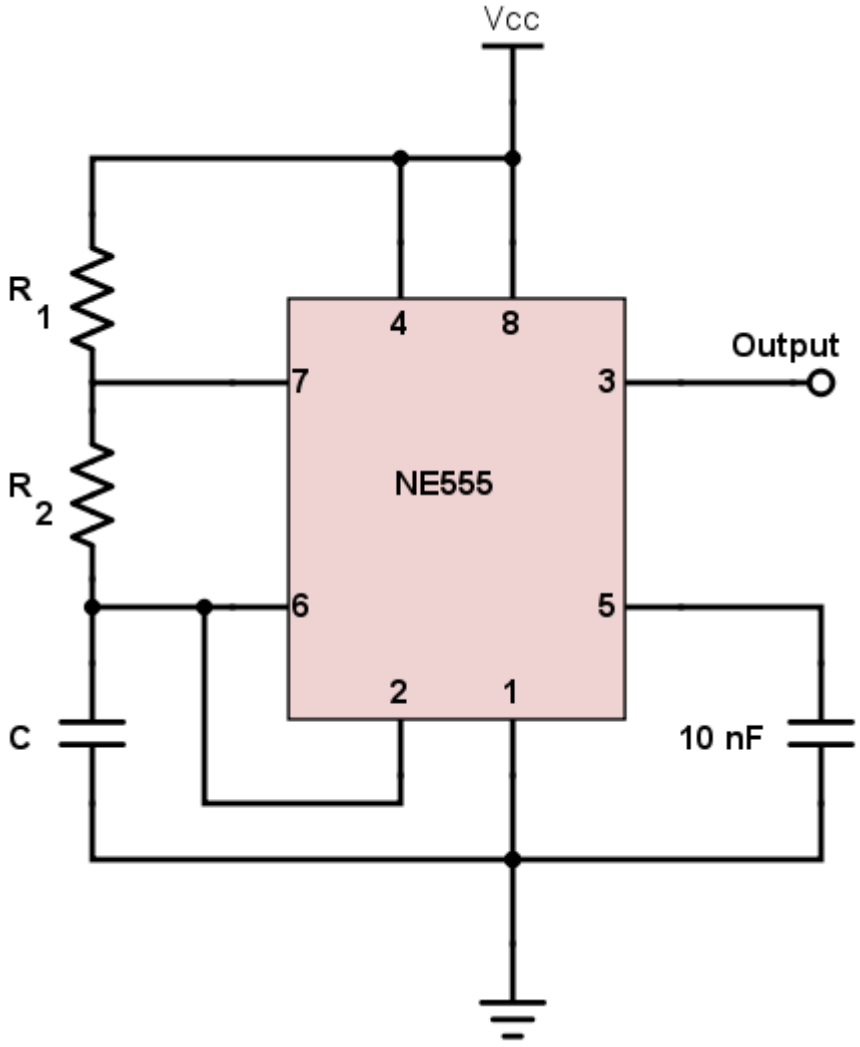
ตัวอย่างการกรองความถี่โดยใช้แร่

ตัวอย่างวงจรสร้างสัญญาณนาฬิกาของ PC

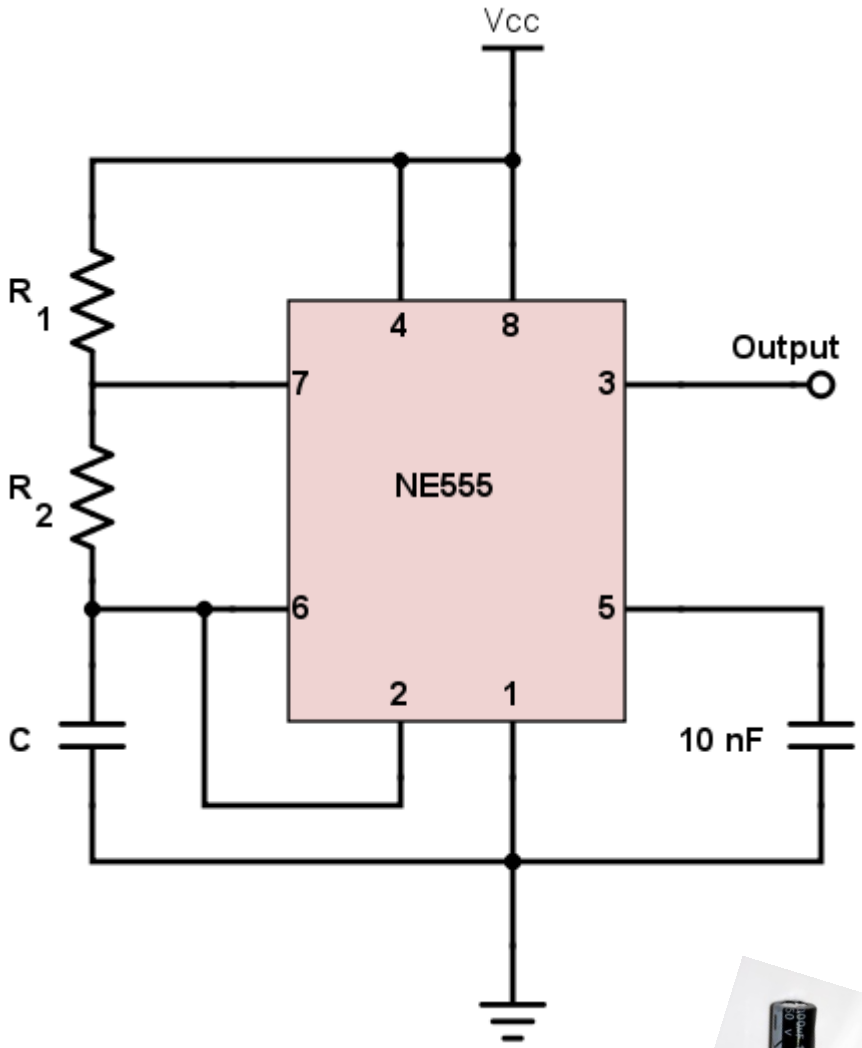


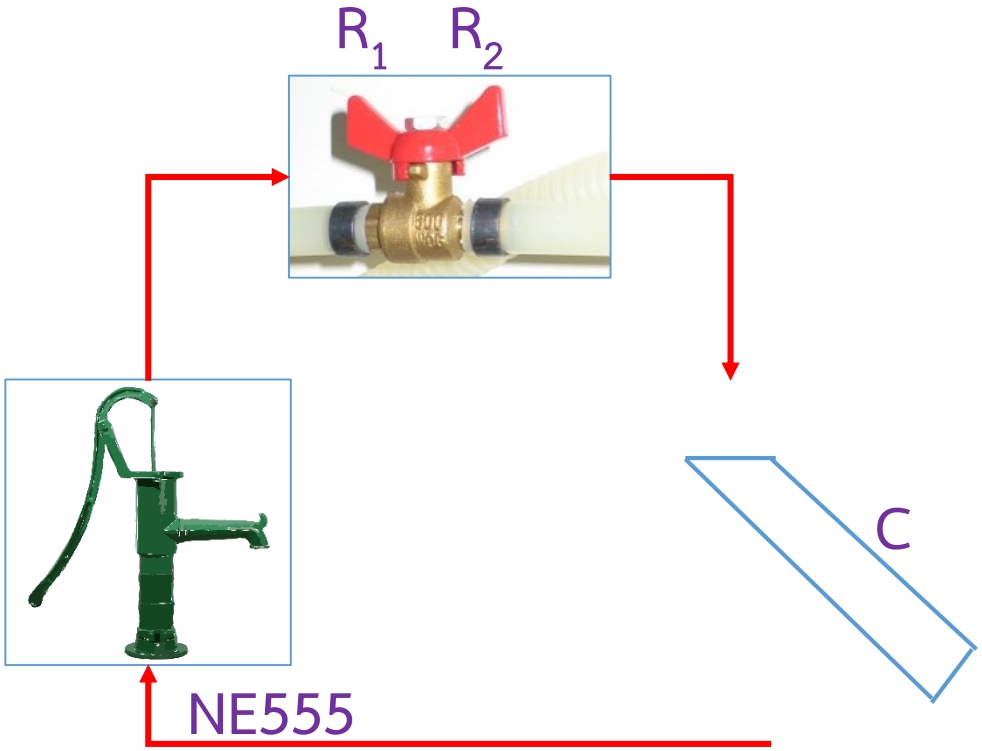


การสร้างสัญญาณนาฬิกาอย่างง่ายด้วย IC เบอร์ NE555



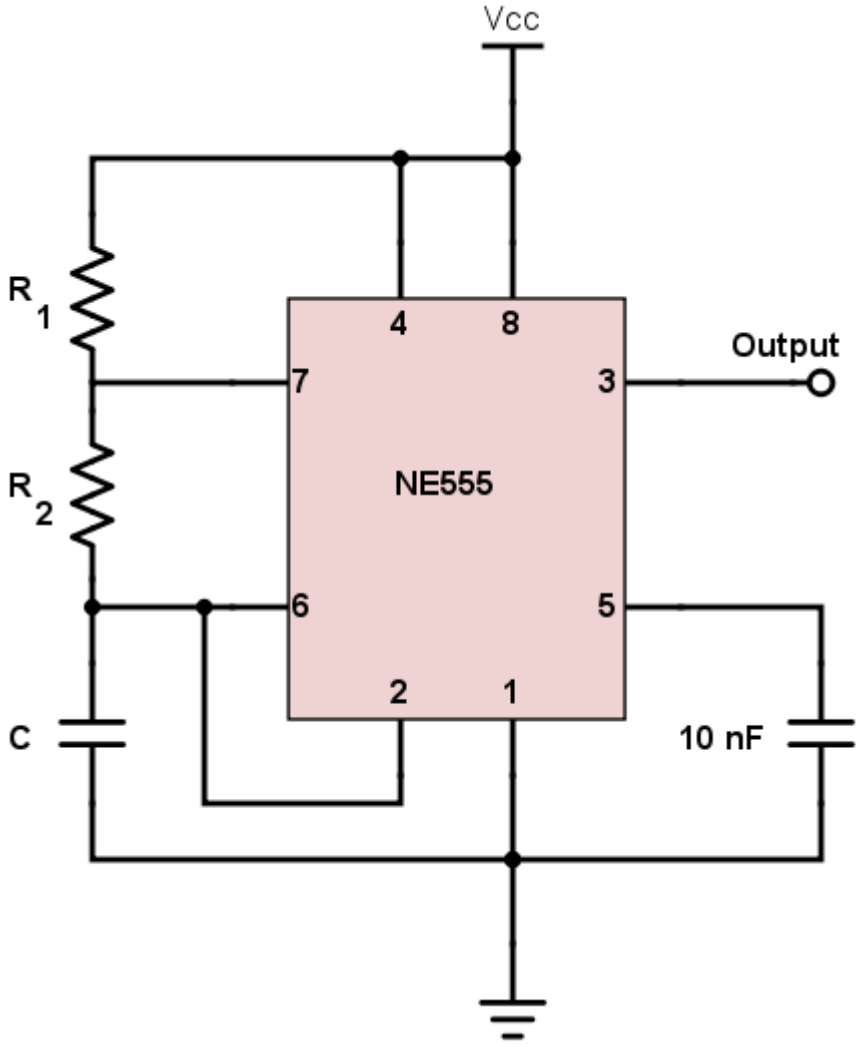
การสร้างสัญญาณนาฬิกาอย่างง่ายด้วย IC เบอร์ NE555

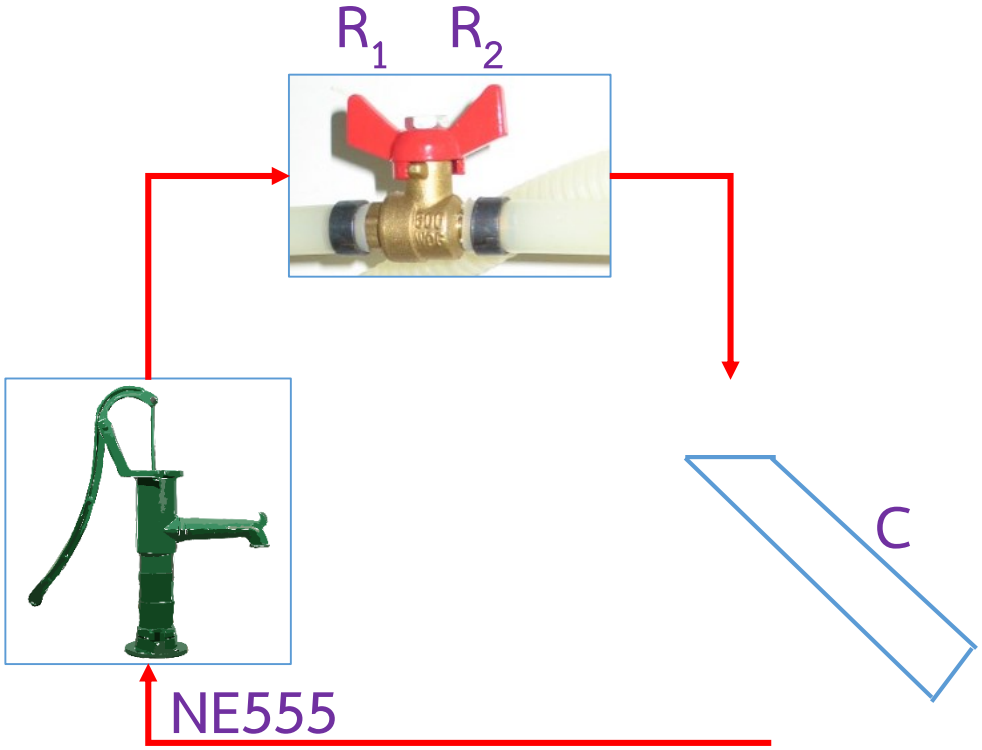




C คือตัวเก็บประจุ ทำงานคล้ายแบตเตอรี่แบบชาร์ทไฟได้ มีหน่วยเป็น F

การสร้างสัญญาณนาฬิกาอย่างง่ายด้วย IC เบอร์ NE555

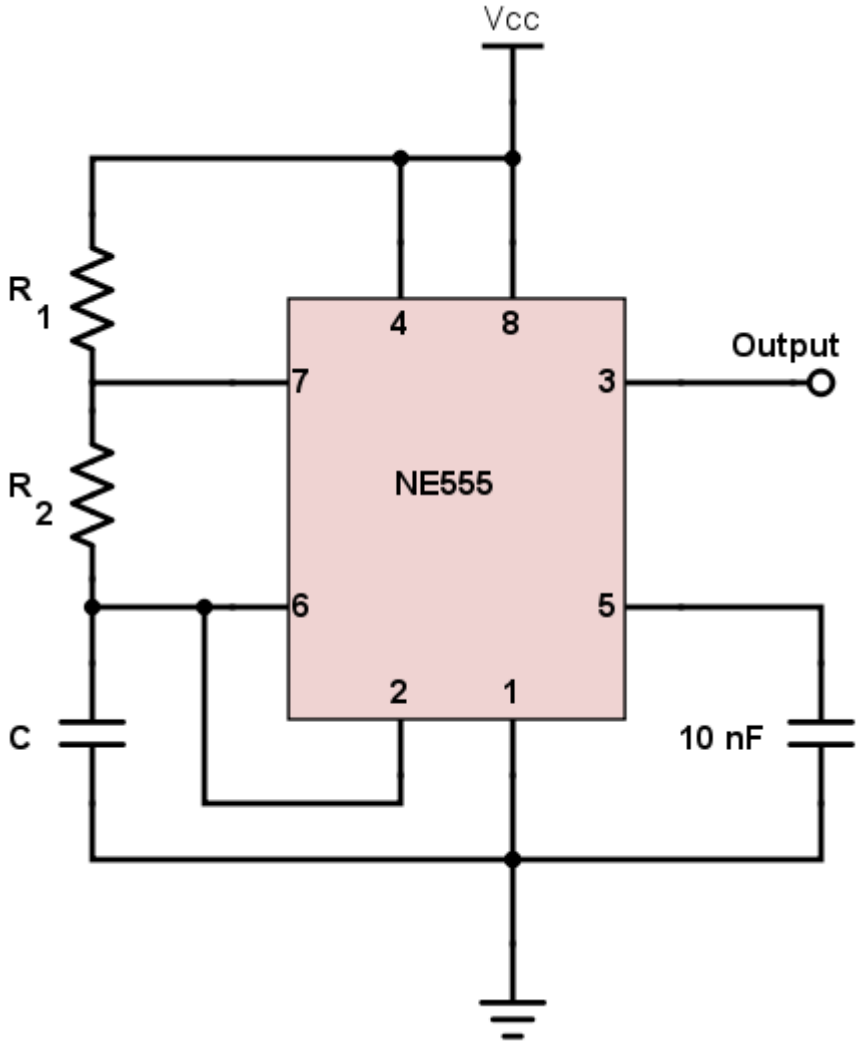




$$f = \frac{1.44}{(R_1 + 2R_2) \times C}$$

$$d = \frac{(R_1 + R_2)}{(R_1 + 2R_2)} \times 100$$

จงออกแบบวงจรสร้างสัญญาณนาฬิกาความถี่ 1Hz ด้วย NE555 กำหนดให้ใช้ C ขนาด 100μF



วิธีทำ กำหนด R1=1000

$$f = \frac{1.44}{(R_1 + 2R_2) \times C}$$

$$1 = \frac{1.44}{(1000 + 2R_2) \times 0.0001}$$

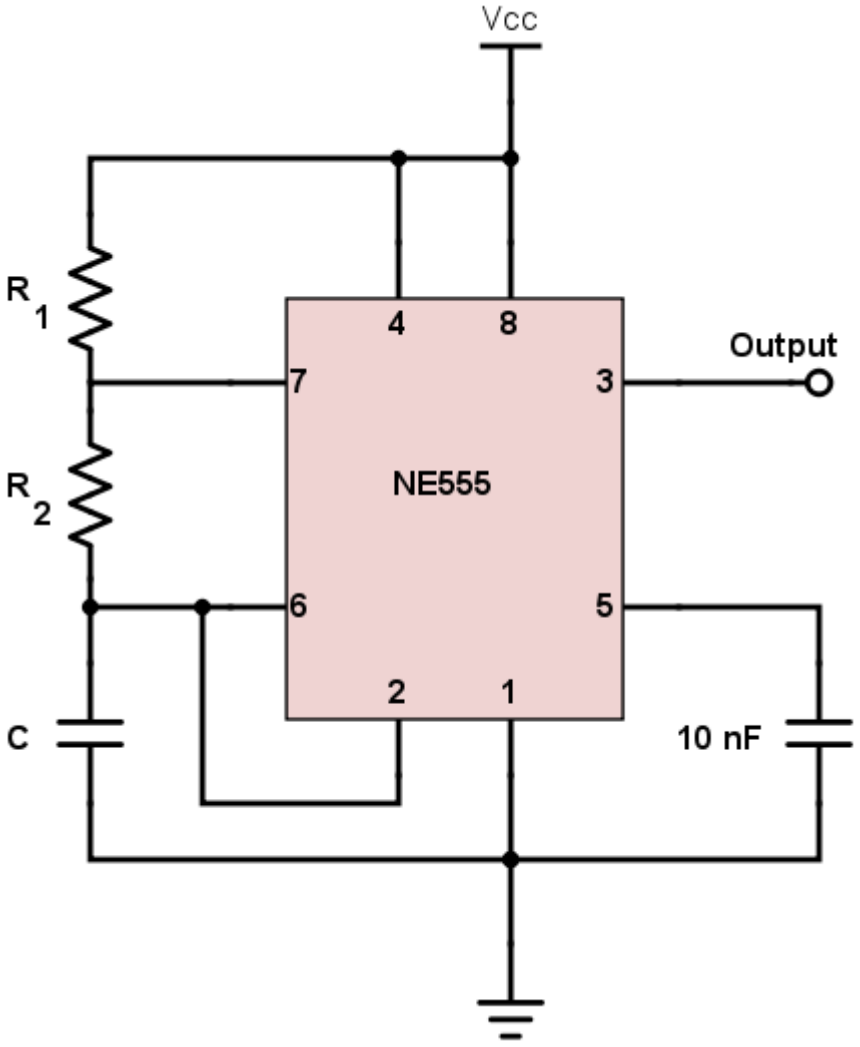
$$1 = \frac{1.44}{0.1 + 0.0002R_2}$$

$$R_2 = 6.7\text{k}$$

ตอบ R1=1kΩ, R2=6.7kΩ, C=100μF

แล้วสัญญาณที่สร้างจากวงจรนี้มี duty cycle เท่าใด ?

จงออกแบบวงจรสร้างสัญญาณนาฬิกาความถี่ 1Hz ด้วย NE555 กำหนดให้ใช้ C ขนาด 100μF



วิธีทำ

ตอบ R1=1kΩ, R2=6.7kΩ, C=100μF

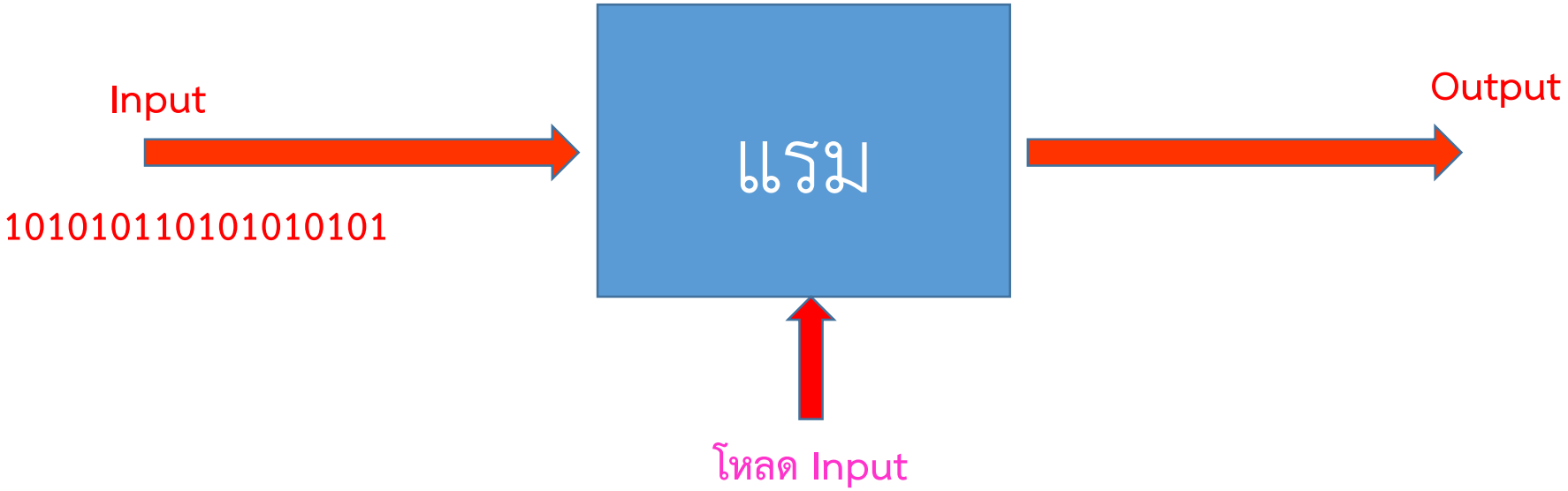
$$d = \frac{(R_1 + R_2)}{(R_1 + 2R_2)}$$

$$d = \frac{(1000 + 6700)}{(1000 + (2 \times 6700)}$$

$$d = \frac{7700}{14400} \times 100 = 53.4$$

ตอบ 53.4%

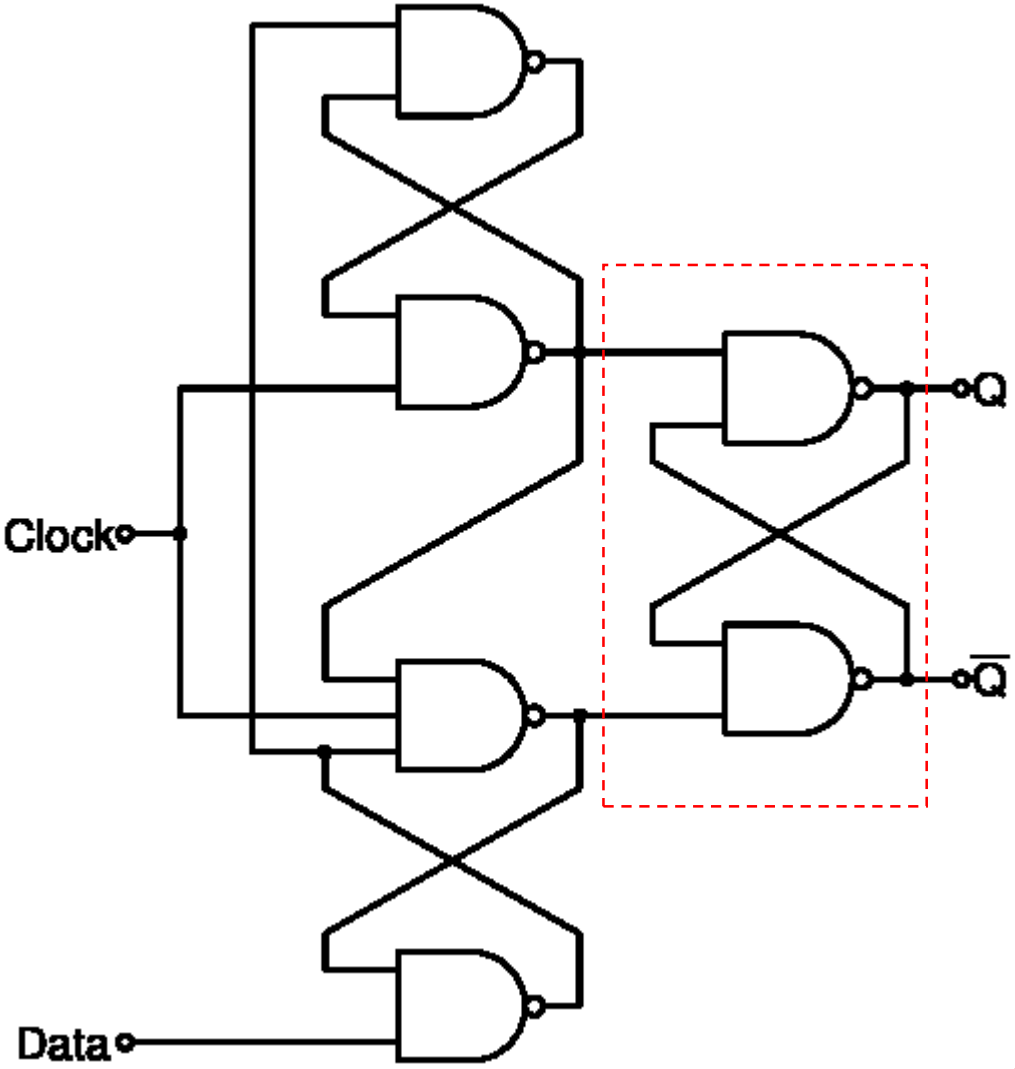
แรมจะต้องสามารถโหลดข้อมูลเข้า และคงข้อมูลที่โหลดได้



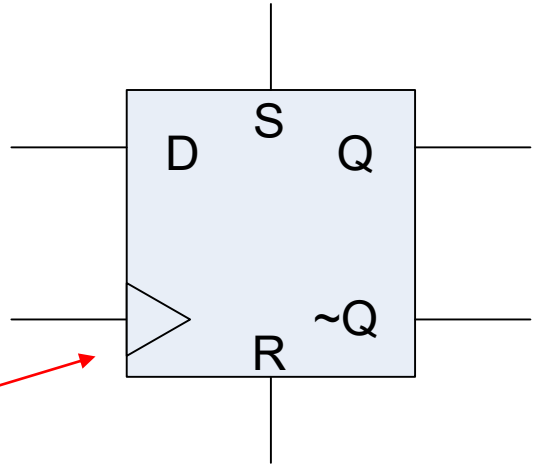
ควบคุมด้วยสัญญาณนาฬิกา

เราสามารถนำเอา SR Flip flop มาดัดแปลงให้ทำงานเป็นแรมขนาด 1 บิตได้ เรียกว่า Data Flip flop (D-FF)

D- Flip Flop

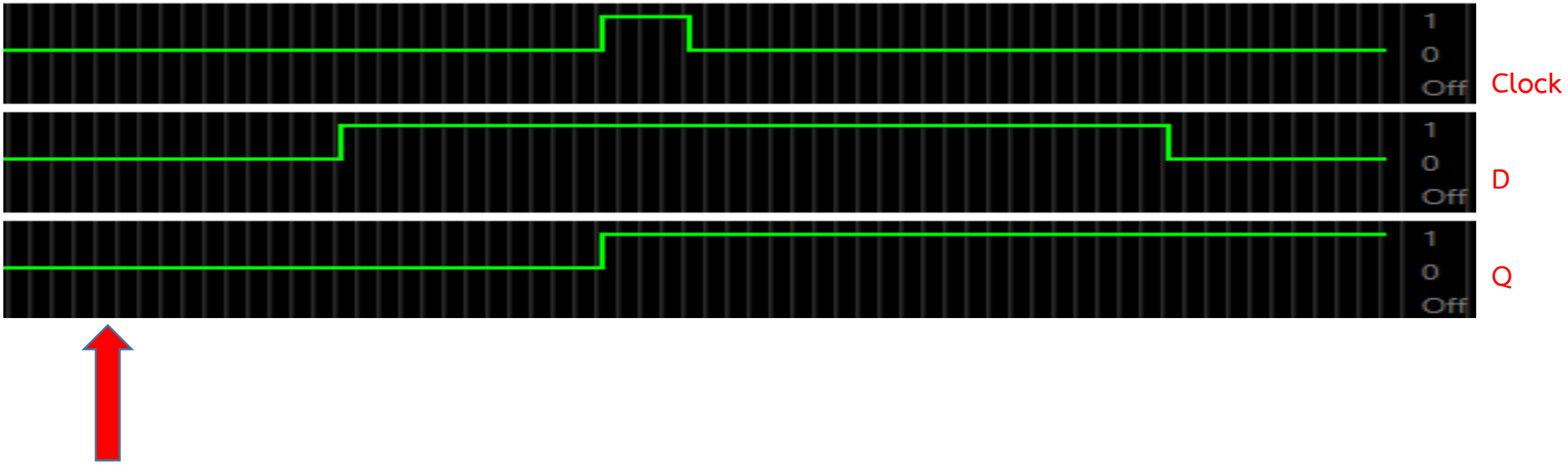


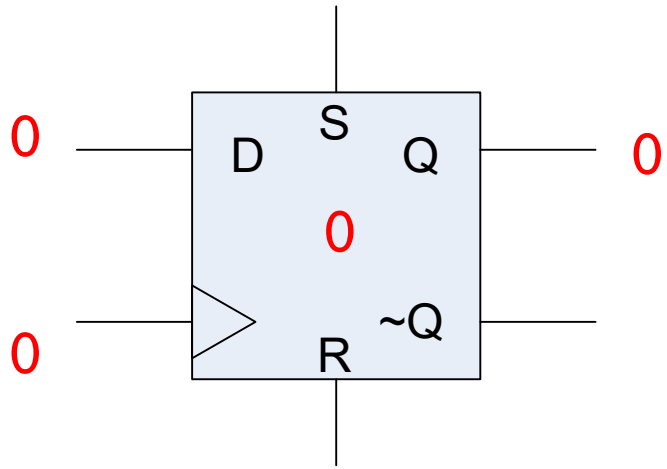
Input		Output	
D	Clock	Q_{n+1}	
0	Positive edge	0	โหลดข้อมูล
1	Positive edge	1	โหลดข้อมูล
X	Non	Q_n	อ่านข้อมูล



*Input ที่เป็นสัญญาณนาฬิกา นิยมเขียนแทนด้วยเครื่องหมาย 3 เหลี่ยม

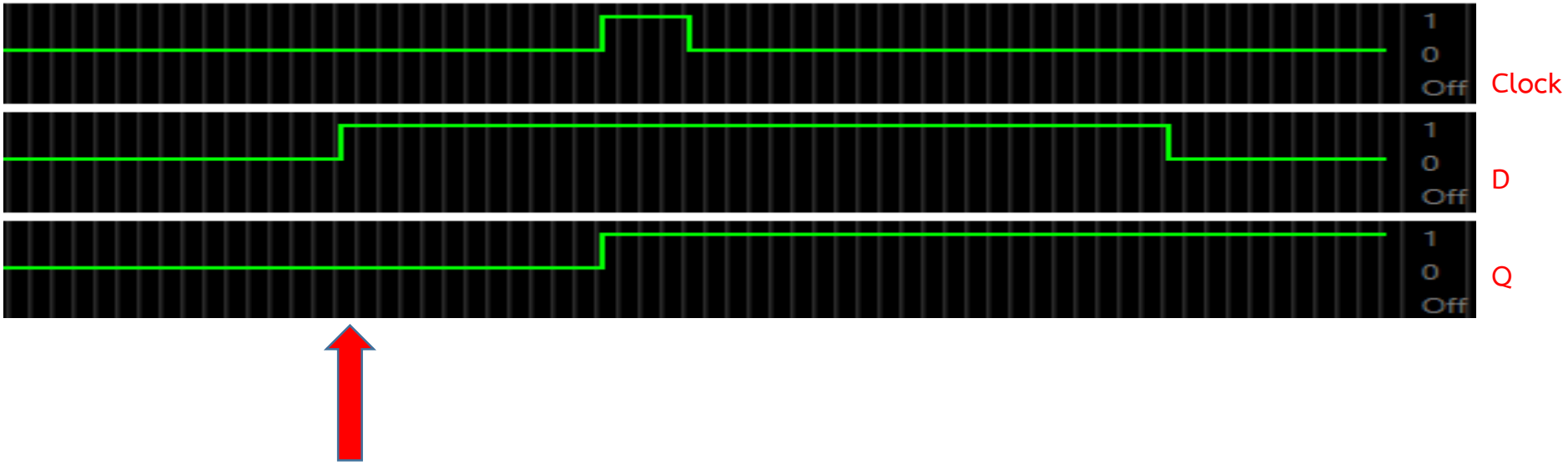
D- Flip Flop

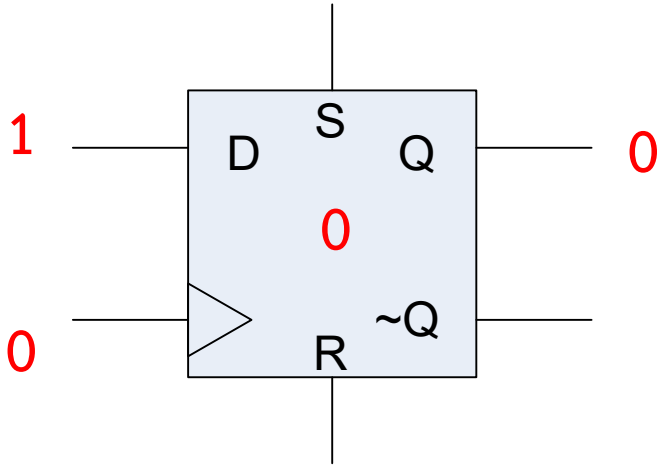




Input		Output	
D	Clock	Q_{n+1}	
0	Positive edge	0	โหลดข้อมูล
1	Positive edge	1	โหลดข้อมูล
X	Non	Q_n	อ่านข้อมูล

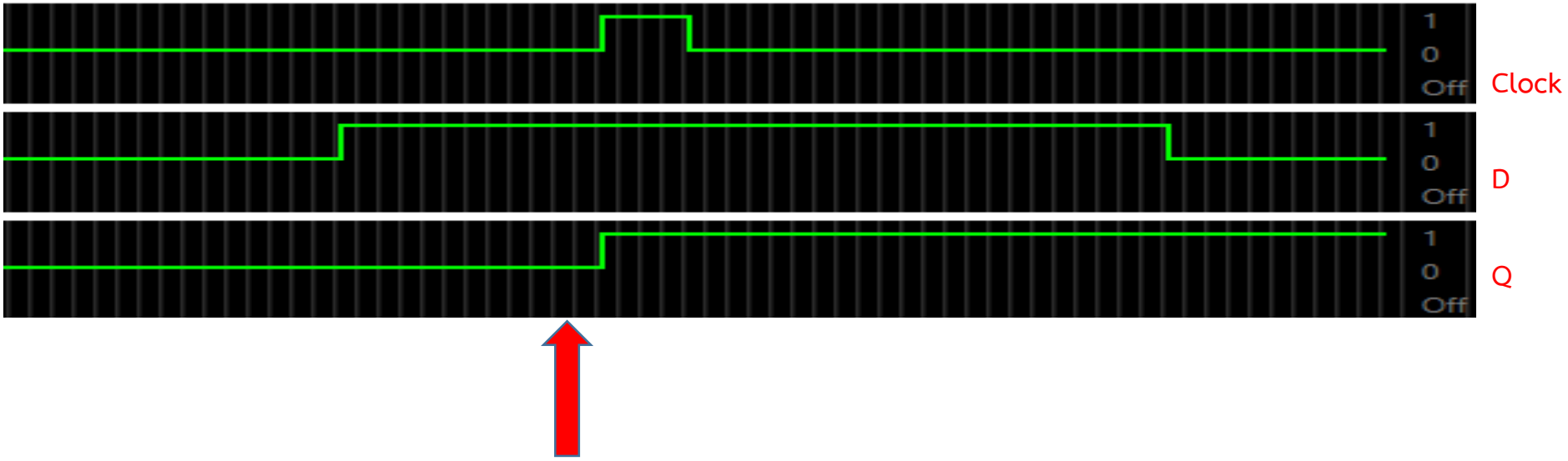
D- Flip Flop

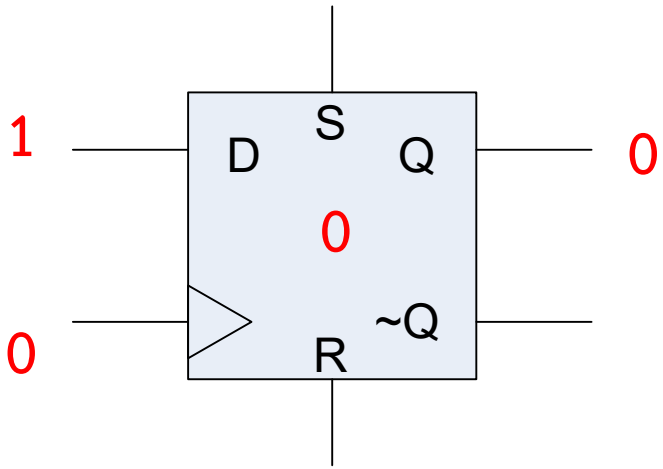




Input		Output	
D	Clock	Q_{n+1}	
0	Positive edge	0	โหลดข้อมูล
1	Positive edge	1	โหลดข้อมูล
X	Non	Q_n	อ่านข้อมูล

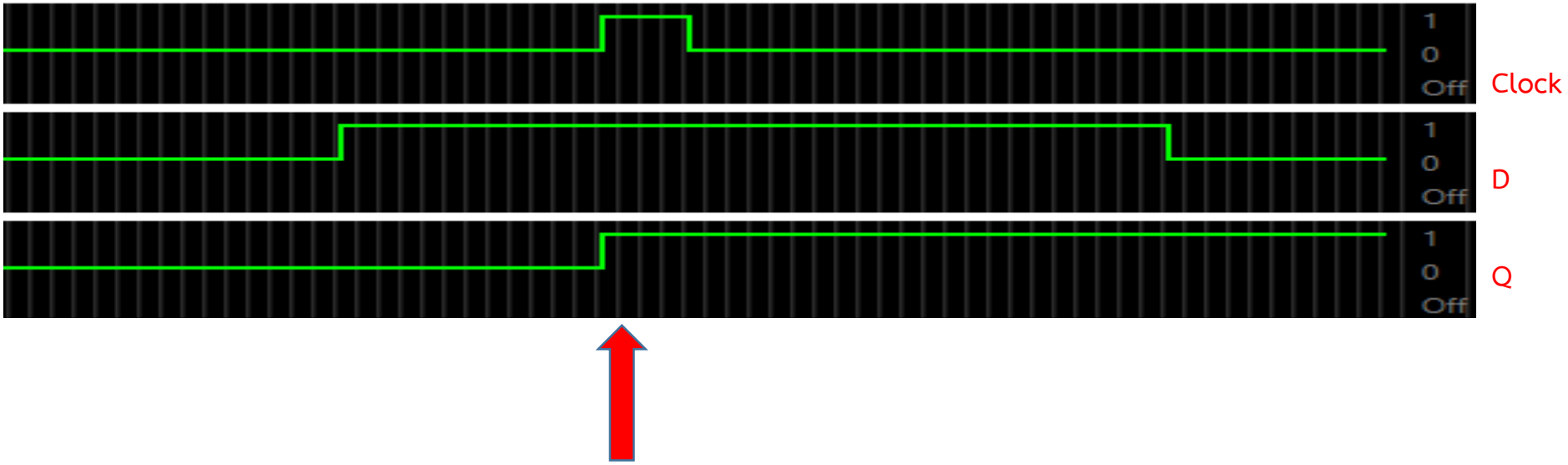
D- Flip Flop

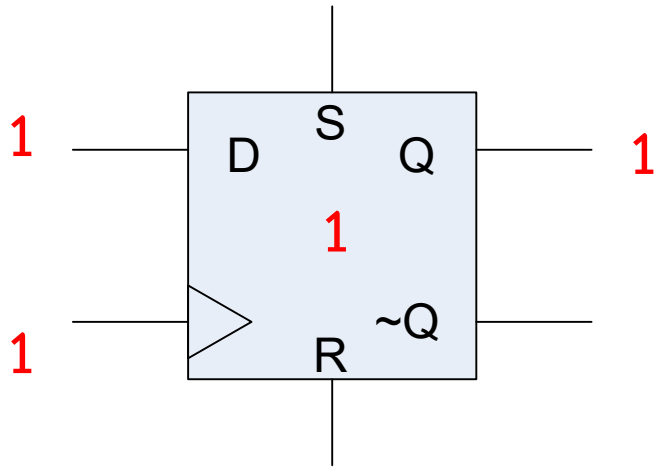




Input		Output	
D	Clock	Q_{n+1}	
0	Positive edge	0	โหลดข้อมูล
1	Positive edge	1	โหลดข้อมูล
X	Non	Q_n	อ่านข้อมูล

D- Flip Flop

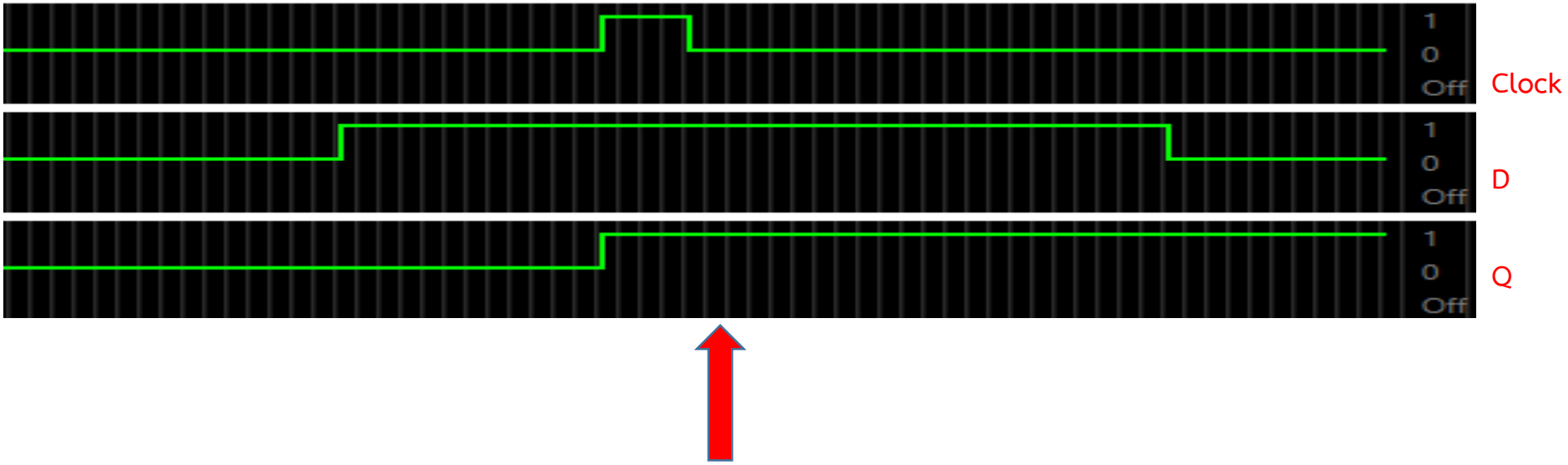


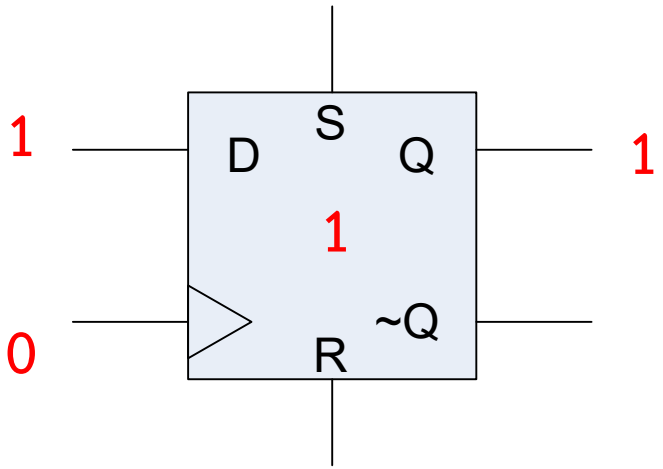


Input		Output	
D	Clock	Q_{n+1}	
0	Positive edge	0	โหลดข้อมูล
1	Positive edge	1	โหลดข้อมูล
X	Non	Q_n	อ่านข้อมูล

บันทึกข้อมูล

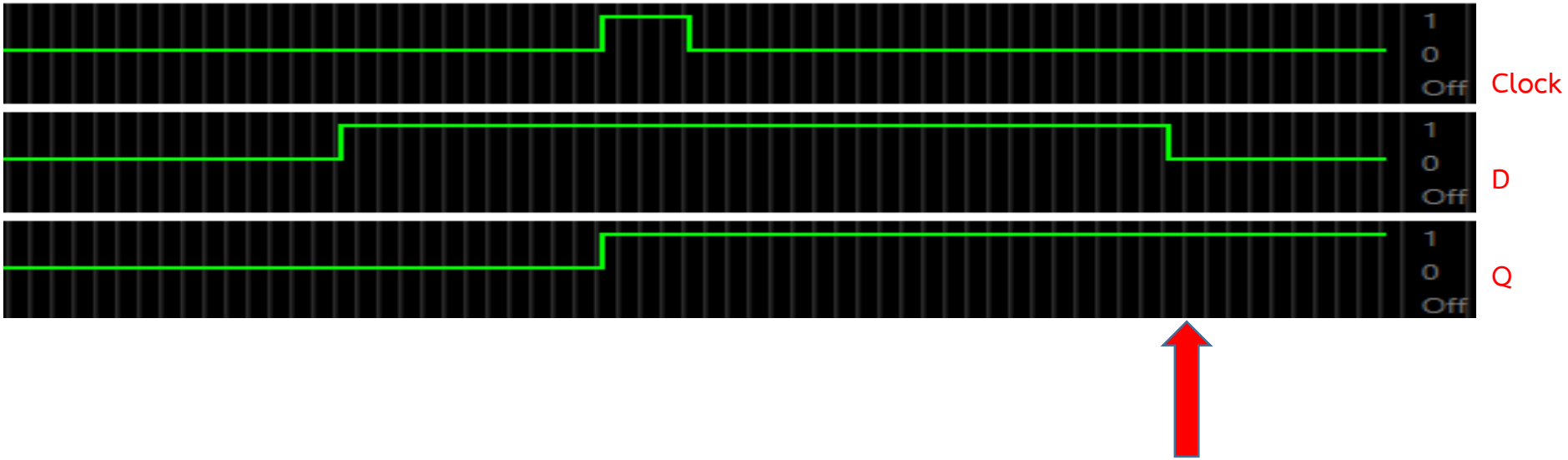
D- Flip Flop

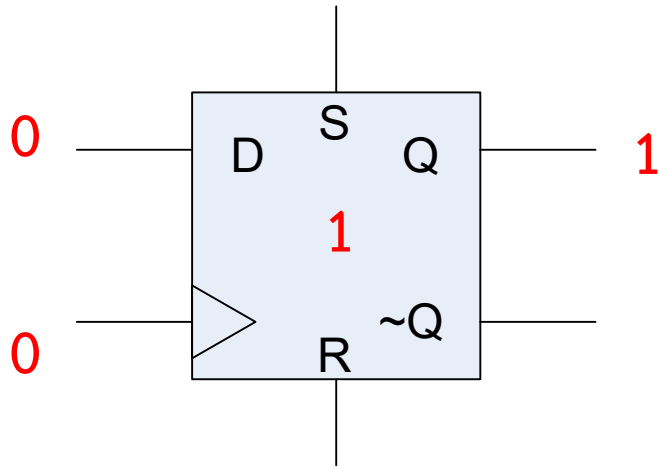




Input		Output	
D	Clock	Q_{n+1}	
0	Positive edge	0	โหลดข้อมูล
1	Positive edge	1	โหลดข้อมูล
X	Non	Q_n	อ่านข้อมูล

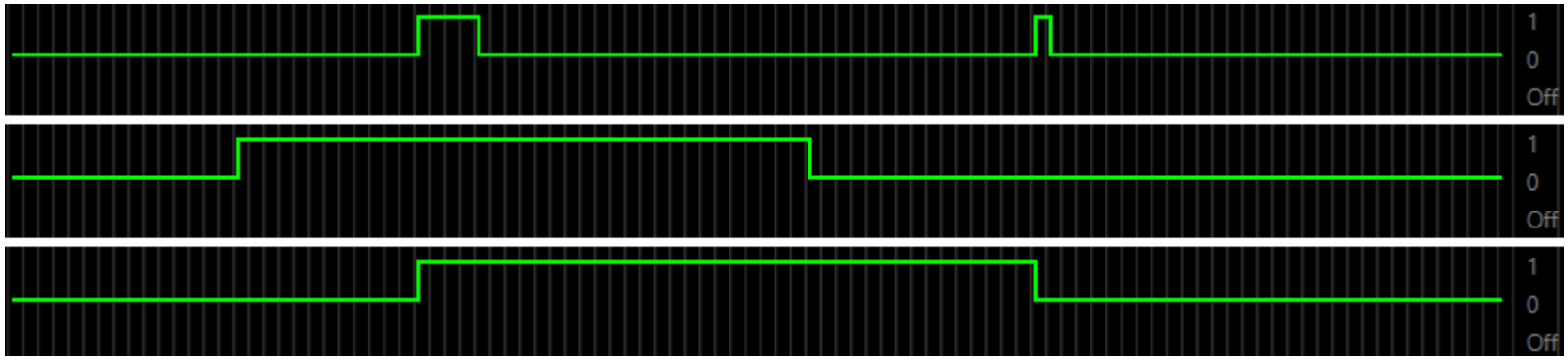
D- Flip Flop

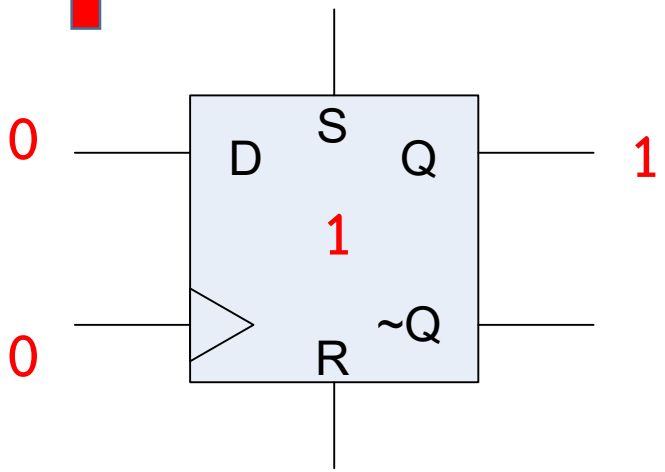




Input		Output	
D	Clock	Q_{n+1}	
0	Positive edge	0	โหลดข้อมูล
1	Positive edge	1	โหลดข้อมูล
X	Non	Q_n	อ่านข้อมูล

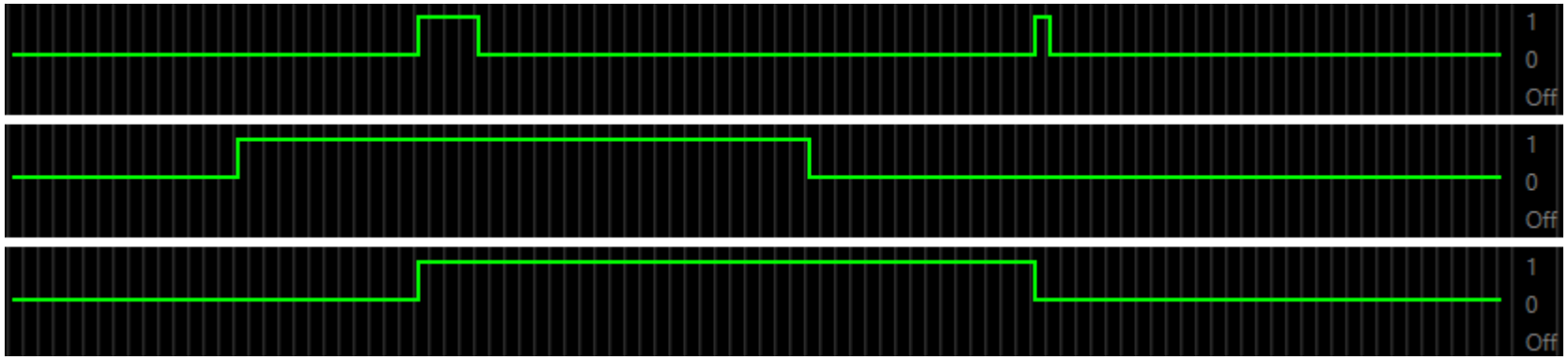
D- Flip Flop

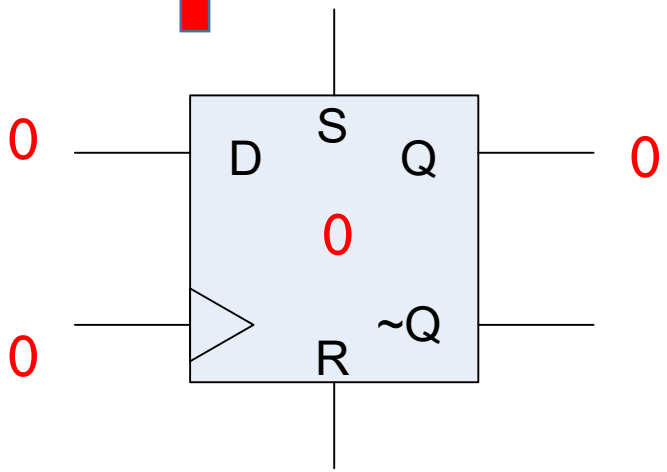




Input		Output	
D	Clock	Q_{n+1}	
0	Positive edge	0	โหลดข้อมูล
1	Positive edge	1	โหลดข้อมูล
X	Non	Q_n	อ่านข้อมูล

D- Flip Flop

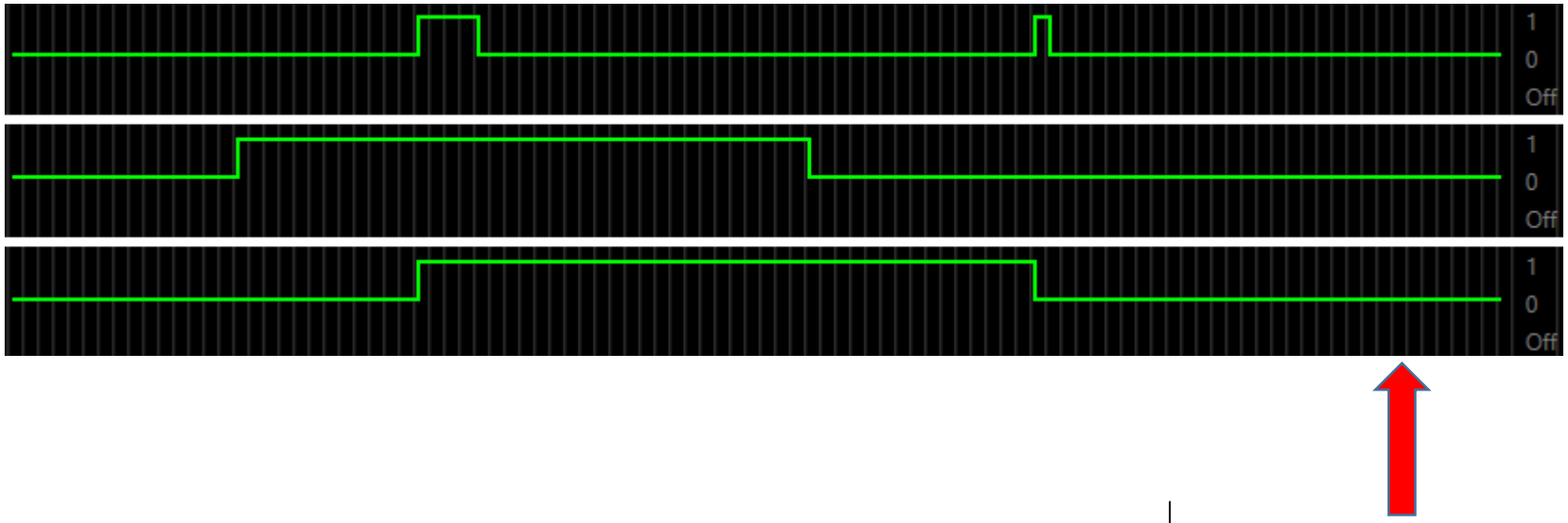


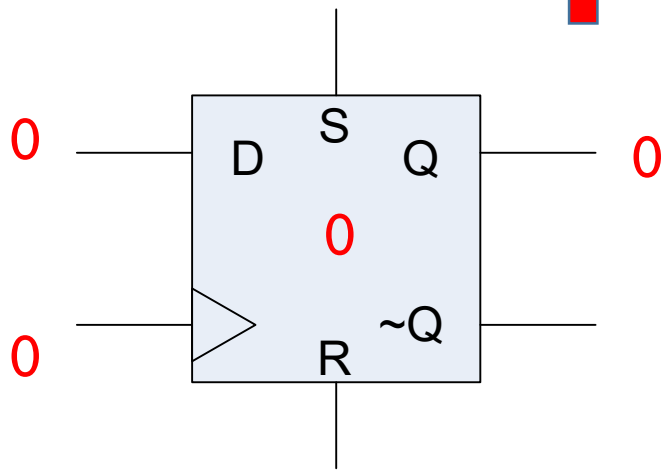


Input		Output	
D	Clock	Q_{n+1}	
0	Positive edge	0	โหลดข้อมูล
1	Positive edge	1	โหลดข้อมูล
X	Non	Q_n	อ่านข้อมูล

บันทึกข้อมูล

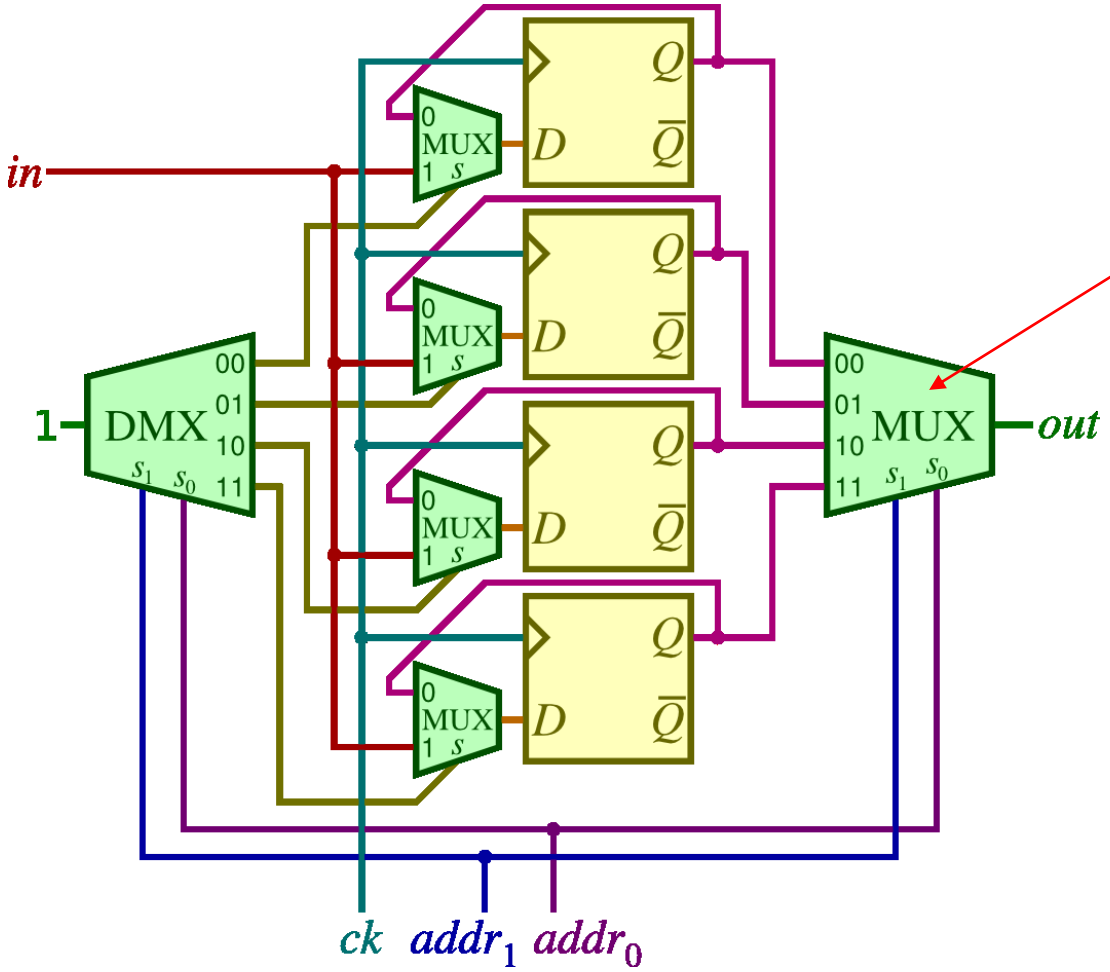
D- Flip Flop





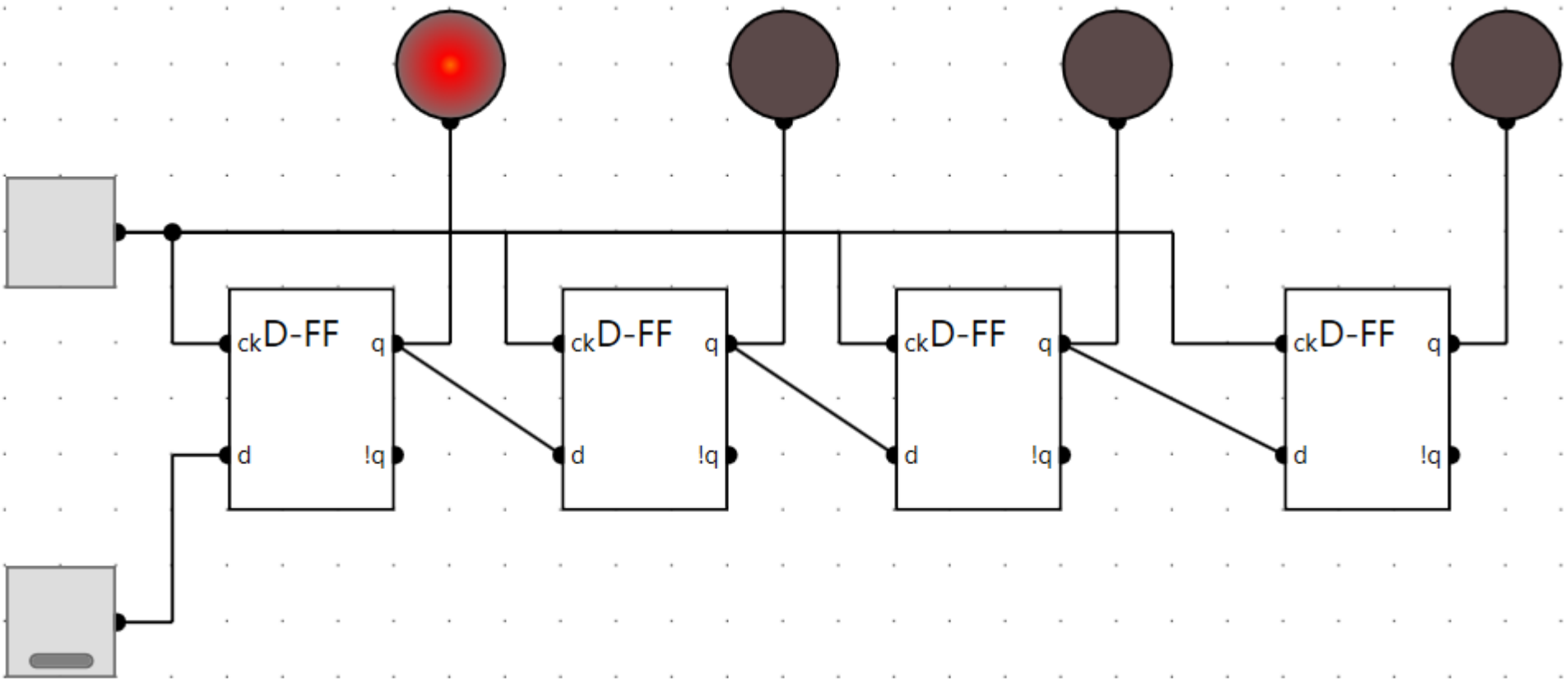
Input		Output	
D	Clock	Q_{n+1}	
0	Positive edge	0	โหลดข้อมูล
1	Positive edge	1	โหลดข้อมูล
X	Non	Q_n	อ่านข้อมูล

แรมเก็บข้อมูลมากกว่า 1 บิตได้ด้วยการใช้วงจร Multiplexer

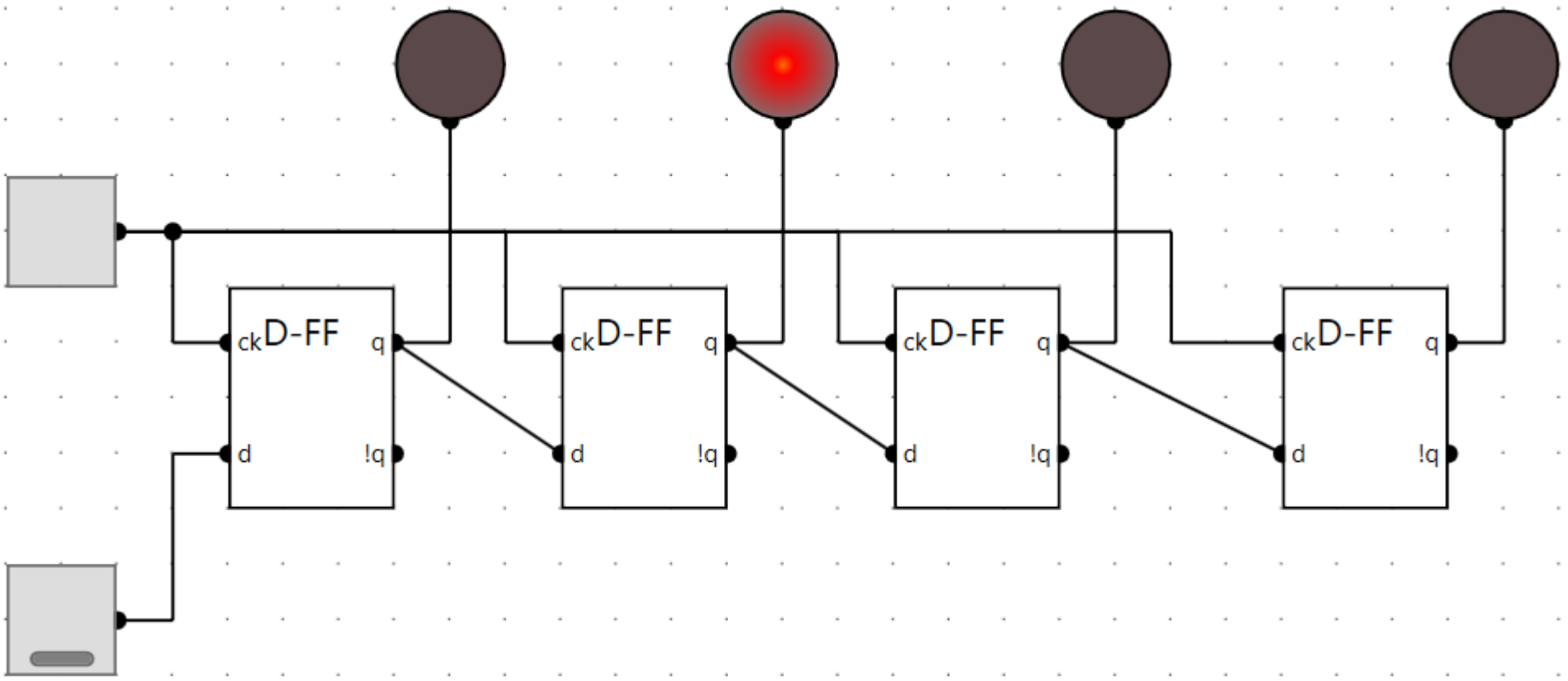


Input		Output			
S1	S0	Y3	Y2	Y1	Y0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

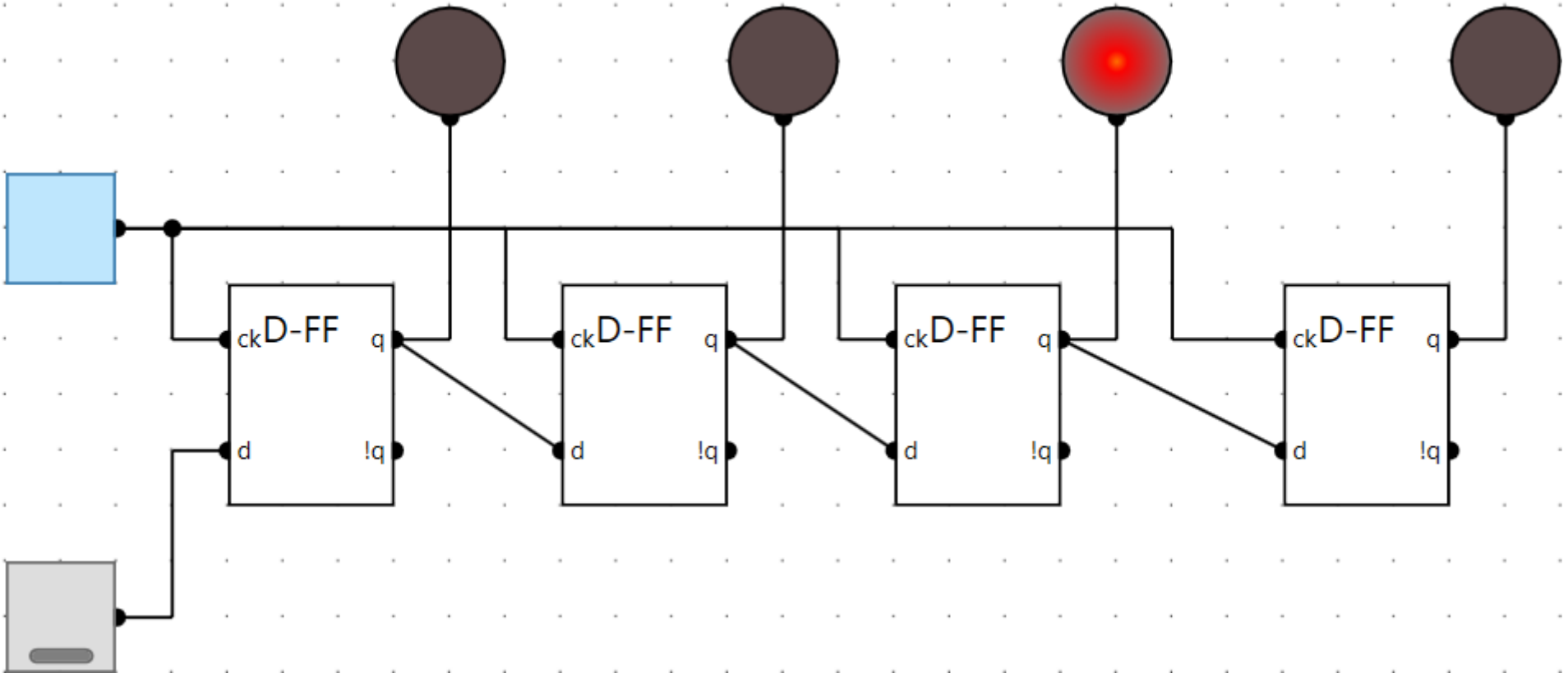
D- Flip flop สามารถพ่วงอนุกรมกันได้ เพื่อสร้างเป็นวงจร Shift register สำหรับเลื่อนข้อมูลได้



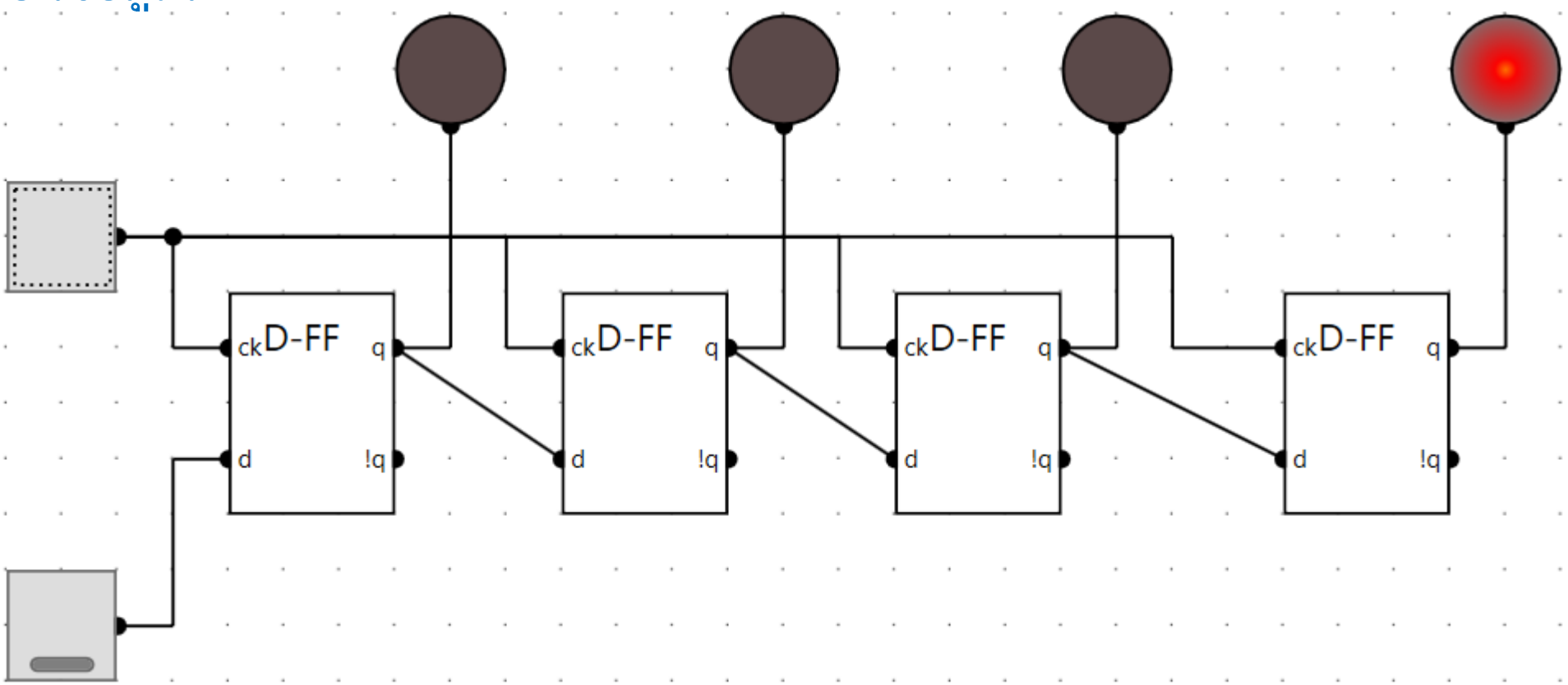
D- Flip flop สามารถพ่วงอนุกรมกันได้ เพื่อสร้างเป็นวงจร Shift register สำหรับเลื่อนข้อมูลได้



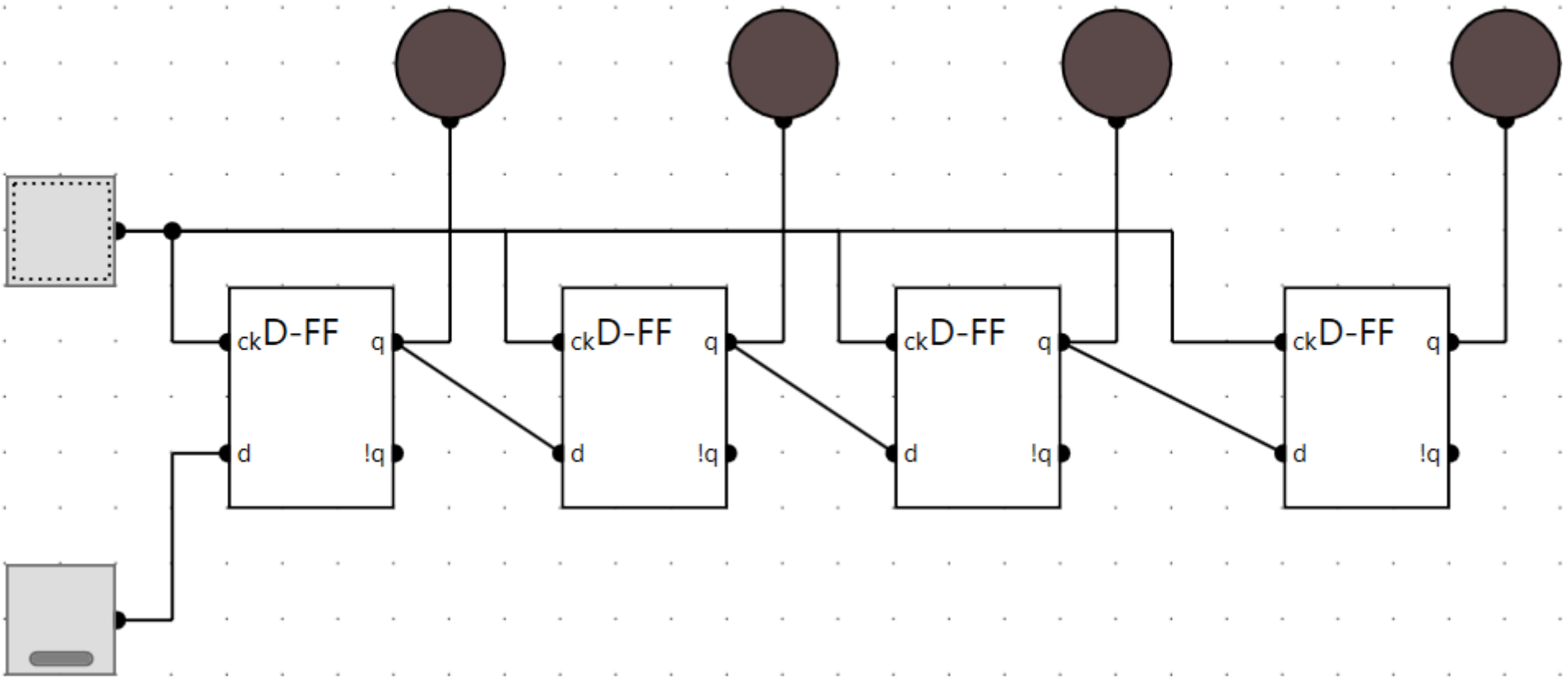
D- Flip flop สามารถพ่วงอนุกรมกันได้ เพื่อสร้างเป็นวงจร Shift register สำหรับเลื่อนข้อมูลได้



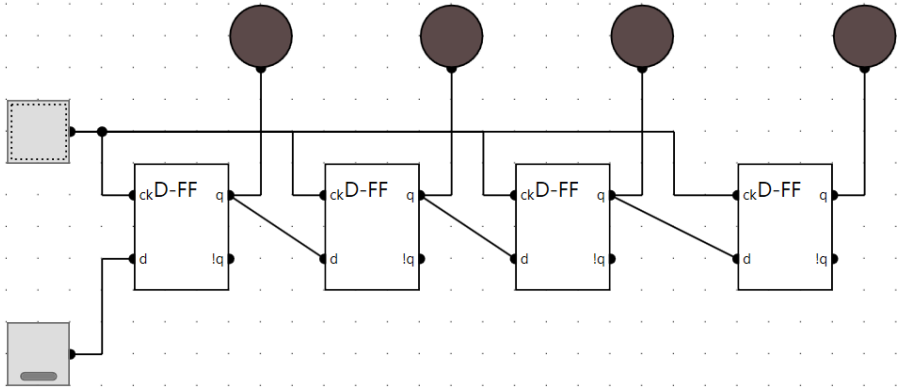
D- Flip flop สามารถพ่วงอนุกรมกันได้ เพื่อสร้างเป็นวงจร Shift register สำหรับเลื่อนข้อมูลได้



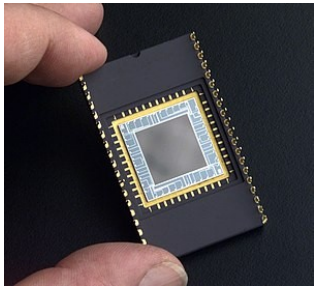
D- Flip flop สามารถพ่วงอนุกรมกันได้ เพื่อสร้างเป็นวงจร Shift register สำหรับเลื่อนข้อมูลได้



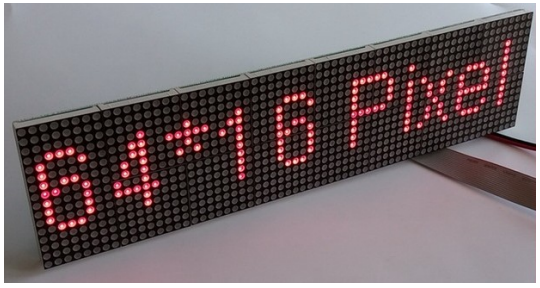
Shift register เป็นวงจรพื้นฐานในการสื่อสารข้อมูลดิจิทัล
และยังใช้ในวงจรคูณเลขฐานสองได้ด้วย



Sensor ของกล้องดิจิทัลก็ใช้วิธีนี้ในการดึงข้อมูลภาพออกมาทีละ pixel

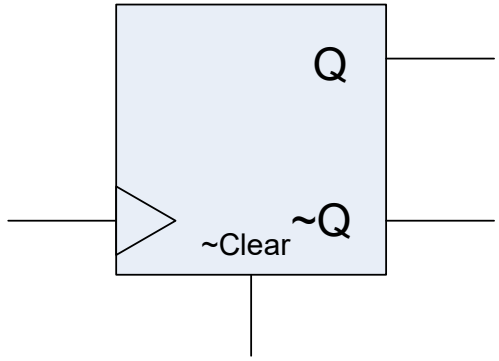


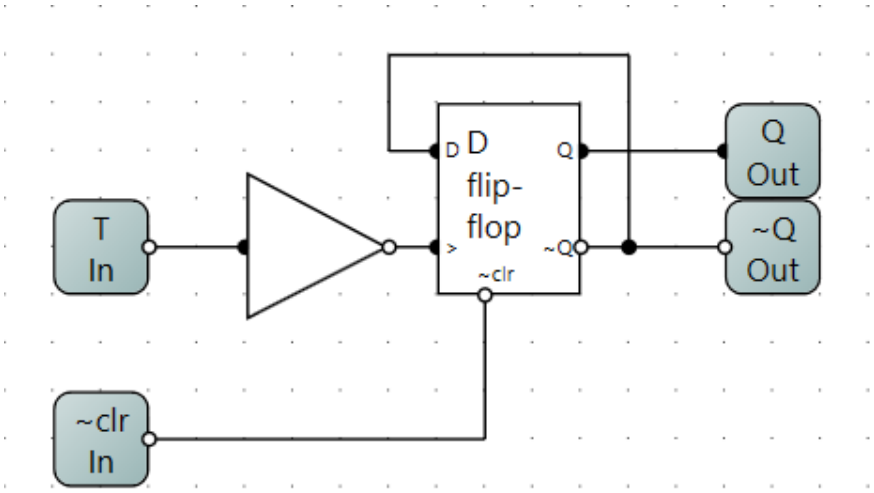
จอ LED แบบ dot matrix ก็ใช้วิธีนี้ในการป้อนภาพเข้าไปทีละจุด



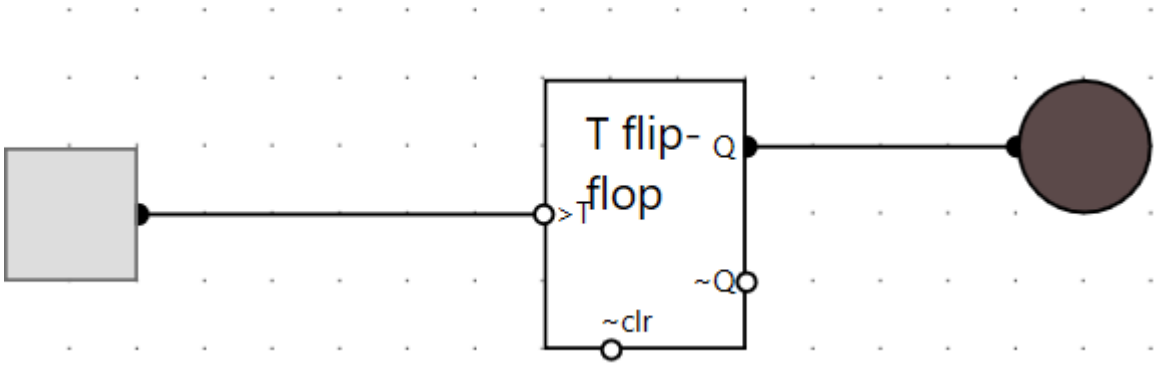
T- Flip Flop (Toggle Flip Flop) คือวงจรที่ดัดแปลงมาจาก D- flip flop ใช้สำหรับ Output ที่ตัวมันเองได้ Latch ไว้ให้เป็น 0 และ 1 เมื่อได้รับสัญญาณนาฬิกา

Input		Output		
Clock	~Clear	Q_{n+1}	$\sim Q_{n+1}$	
0	0	0	1	Reset
0	1	Q_n	$\sim Q_n$	ไม่เปลี่ยนแปลง
1	0	0	1	Reset
1	1	$\sim Q_n$	Q_n	เปลี่ยนเป็นตรงกันข้าม



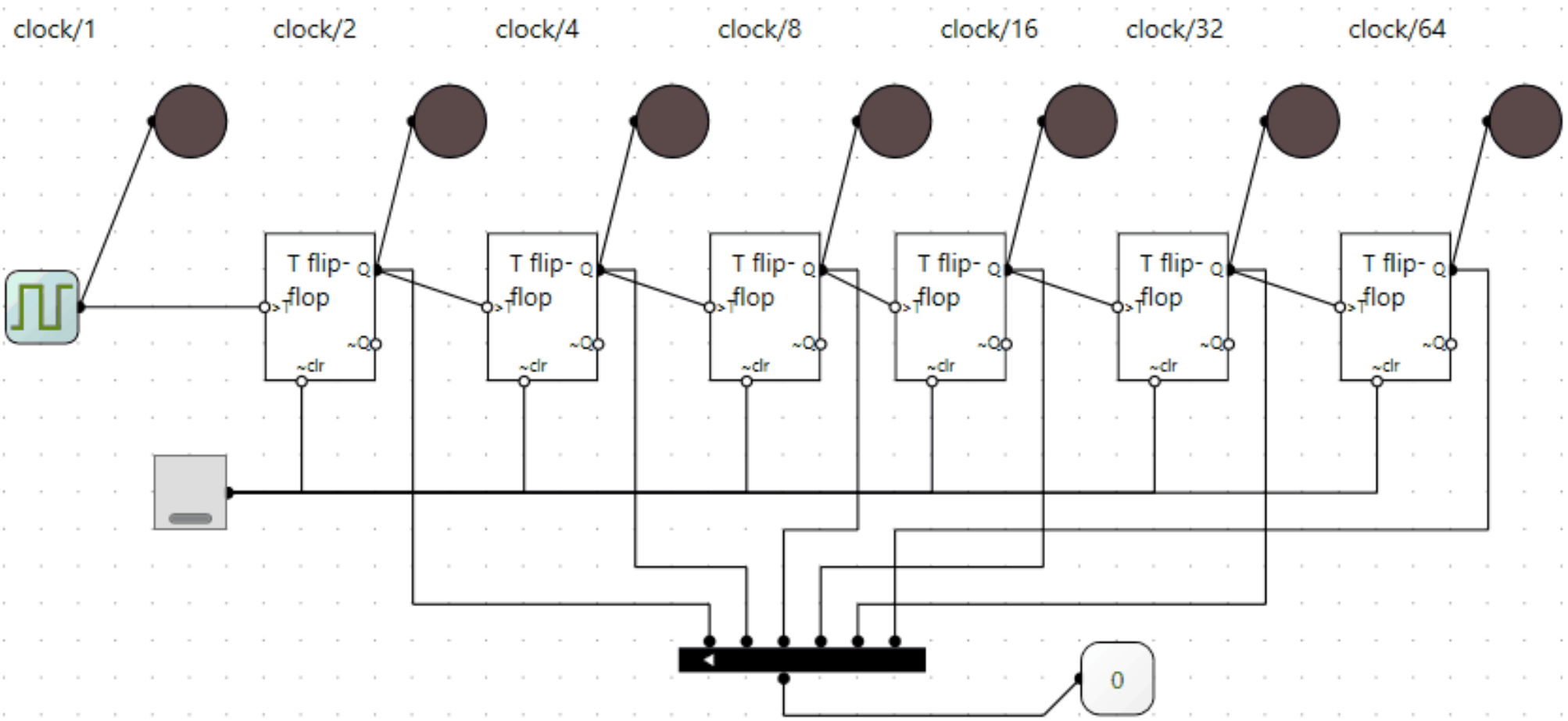


จงออกแบบวงจร Toggle Switch สำหรับเปิดและปิดไฟ โดยกำหนดให้ใช้ปุ่ม กดติดปล่อยดับ (push button) เดียวในการเปิดและปิด

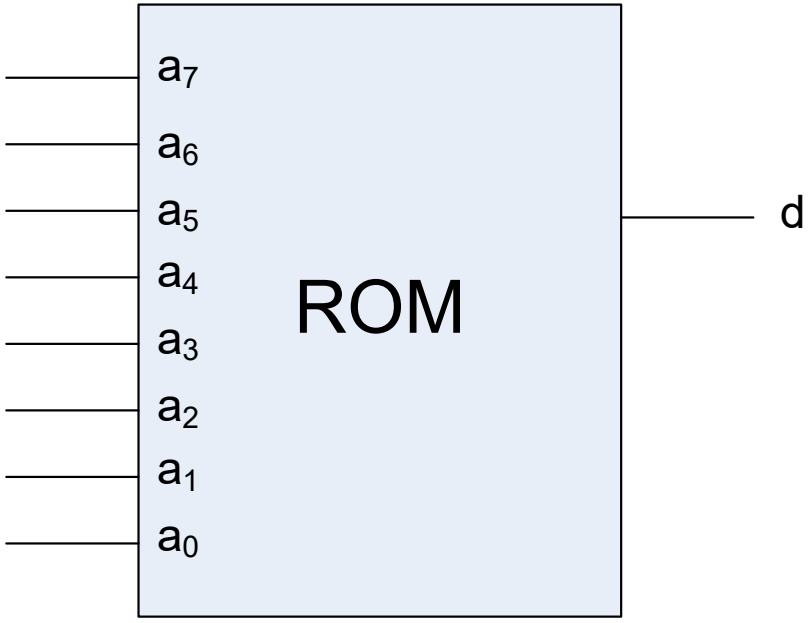


Input		Output		
Clock	~Clear	Q_{n+1}	$\sim Q_{n+1}$	
0	0	0	1	Reset
0	1	Q_n	$\sim Q_n$	ไม่เปลี่ยนแปลง
1	0	0	1	Reset
1	1	$\sim Q_n$	Q_n	เปลี่ยนเป็นตรงกันข้าม

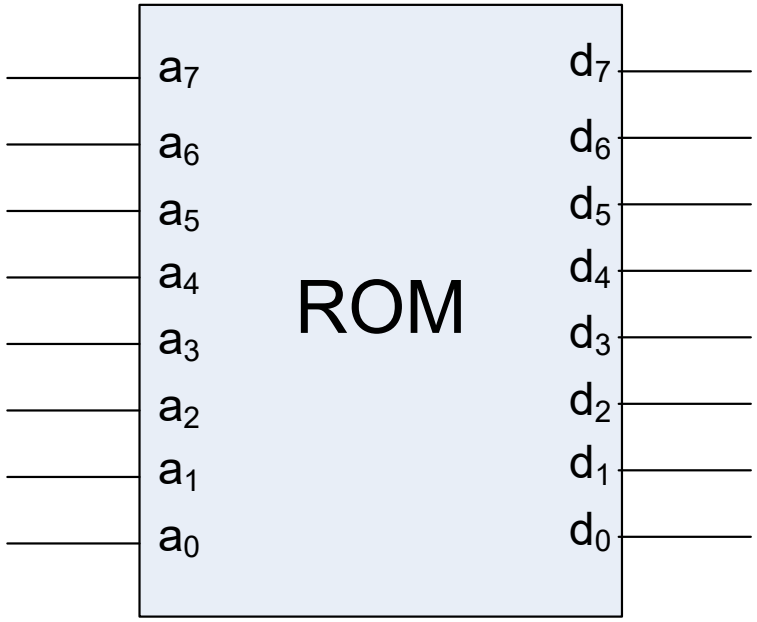
ตัวอย่างการใช้ T-Flip flop ในการสร้างวงจรหารความถี่ และวงจรนับ



แรมและรอม (Random Access Memory and Read Only Memory)
คือหน่วยความจำที่สร้างจากการรวม D Flip flop เข้าด้วยกันหลายตัว โดยสามารถอ่านข้อมูลจาก Flip flop ตัวใดก็ได้โดยทำการกำหนดตำแหน่ง (Address)
รอม จะอ่านข้อมูลได้อย่างเดียว
แรม จะบันทึกข้อมูลในตำแหน่งที่ต้องการได้ด้วย



รอมขนาด 256 bit



รอมขนาด 256 byte

แรมและรอม

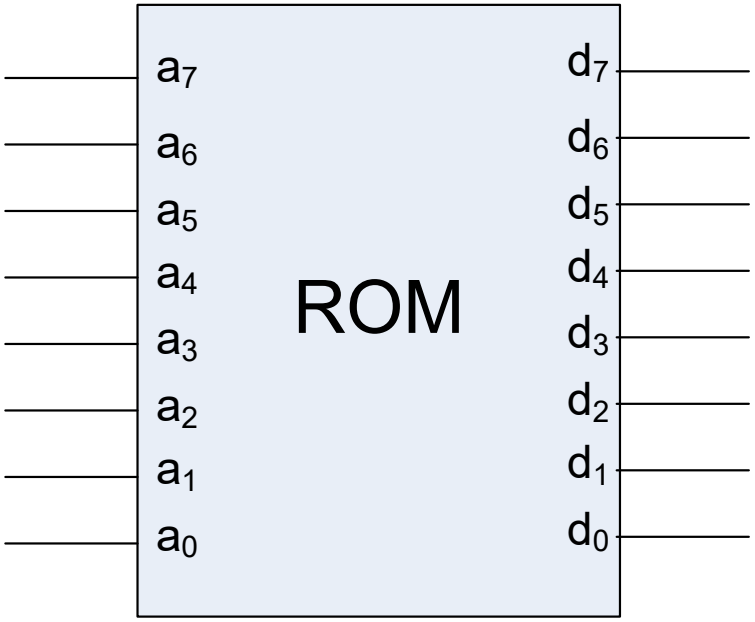
Address bit width: 8 Data bit width: 8

Data

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
2	00	00	00	01	0A	F0	00	00	00	00	00	00	00	00	00	00
3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

ตัวอย่างข้อมูลภายในรอมขนาด 256 bytes

การกำหนดตำแหน่งข้อมูลที่ต้องการอ่าน จะใช้เลขฐานสอง และข้อมูลที่อ่านได้ก็จะเป็นเลขฐานสองเช่นกัน



รอมขนาด 256 bytes

แรมและรอม

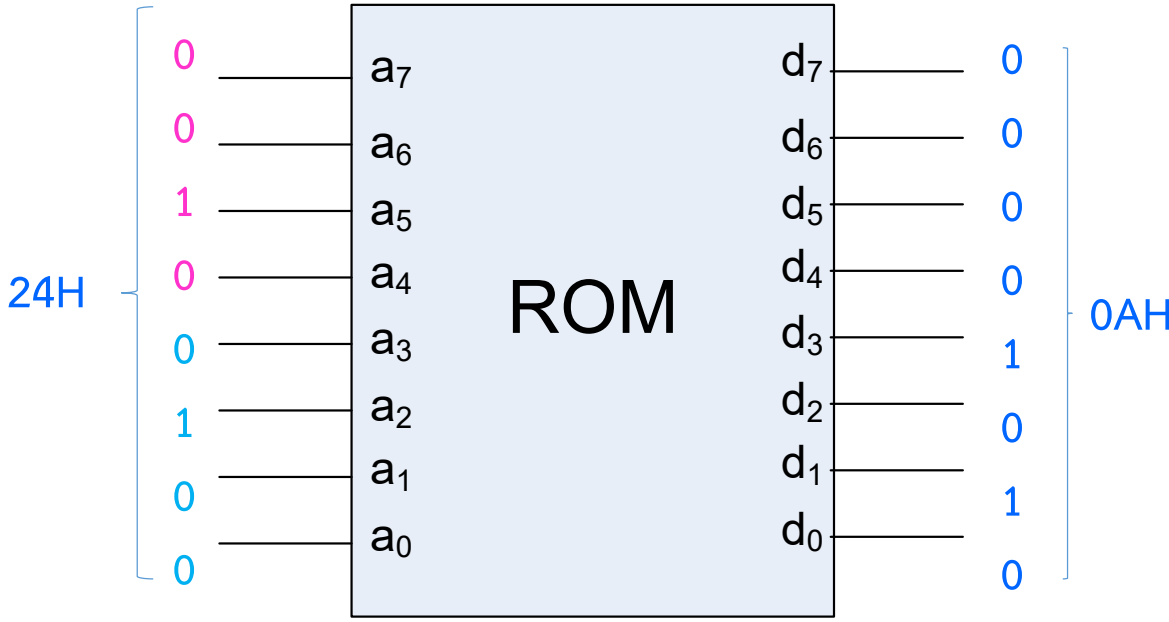
Address= 24H

Address bit width: 8 Data bit width: 8

Data

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
2	00	00	00	01	0A	F0	00	00	00	00	00	00	00	00	00	00
3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

0010 0100



ตัวอย่างข้อมูลภายในรอมขนาด 256 bytes

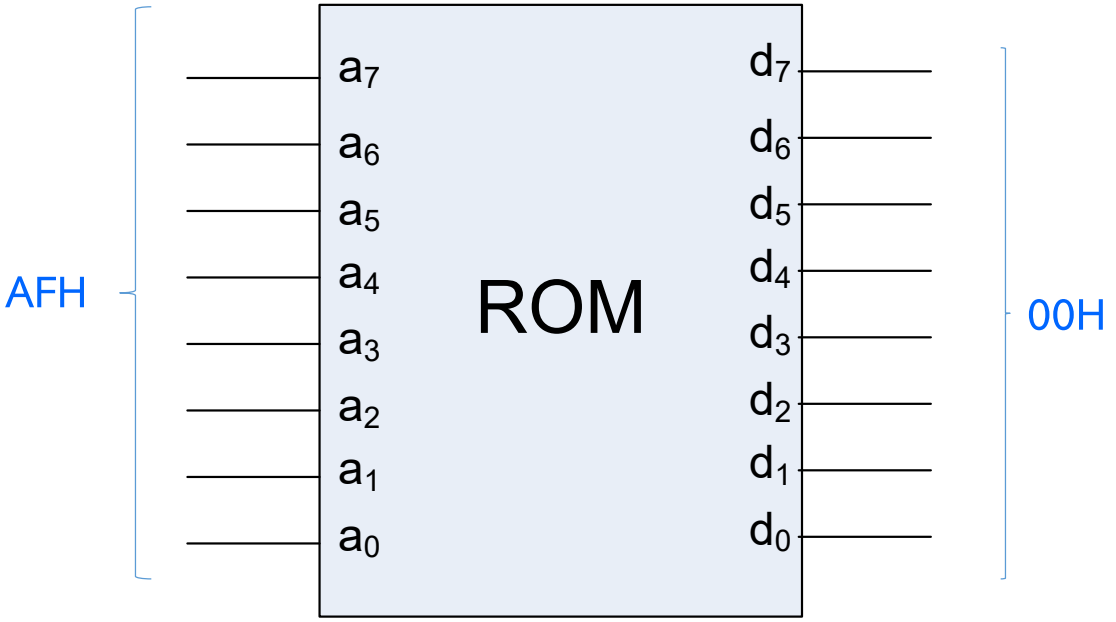
รอมขนาด 256 bytes

แรมและรอม

Address bit width: 8 Data bit width: 8

Data

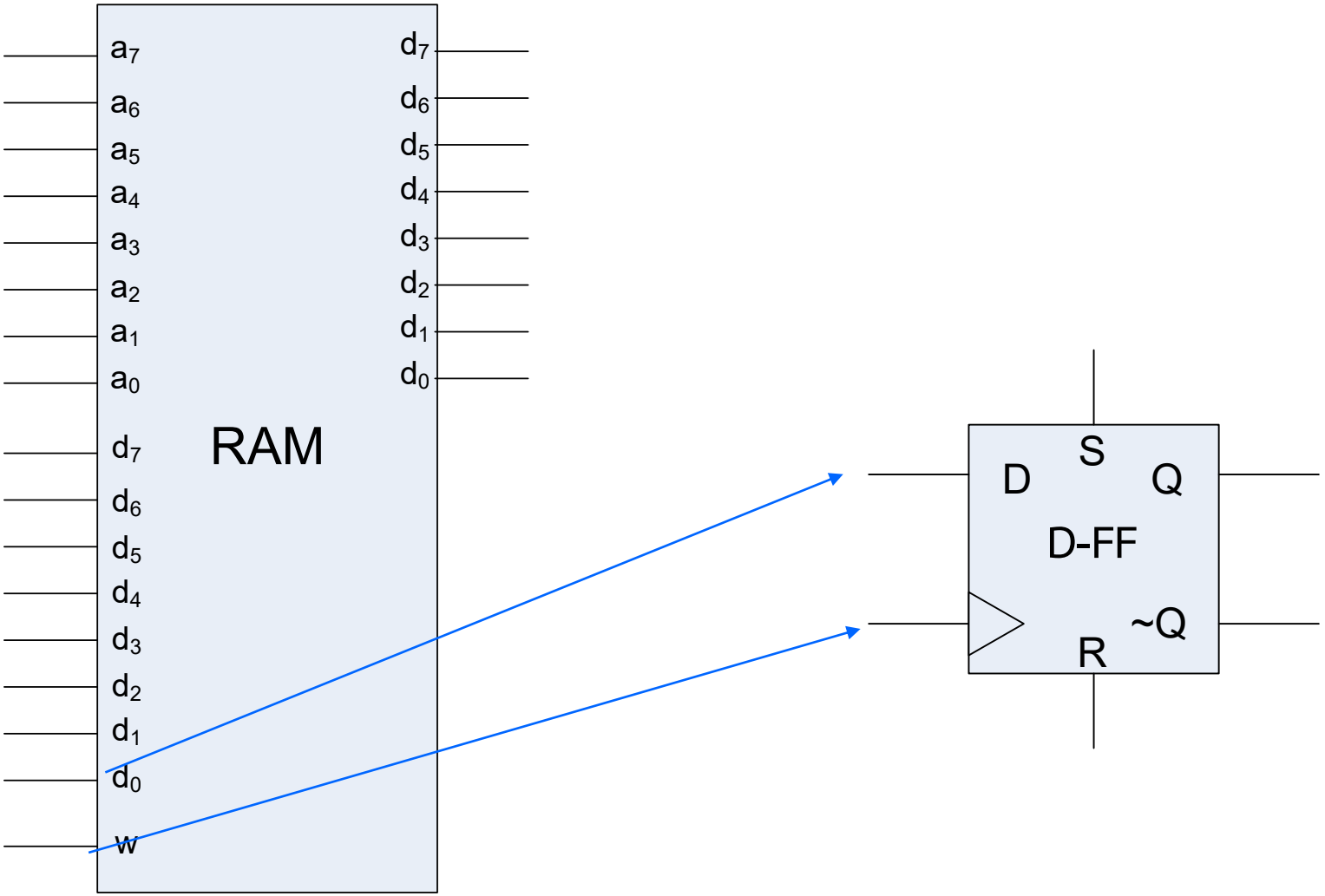
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
2	00	00	00	01	0A	F0	00	00	00	00	00	00	00	00	00	00
3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00



ตัวอย่างข้อมูลภายในรอมขนาด 256 bytes

รอมขนาด 256 bytes

แรม จะคล้ายกับรอม แต่จะมี input เพิ่มในส่วนของ data ที่ต้องการบันทึก และสัญญาณสั่งให้บันทึก



ตัวอย่างการบันทึกข้อมูลเข้าสู่แรม

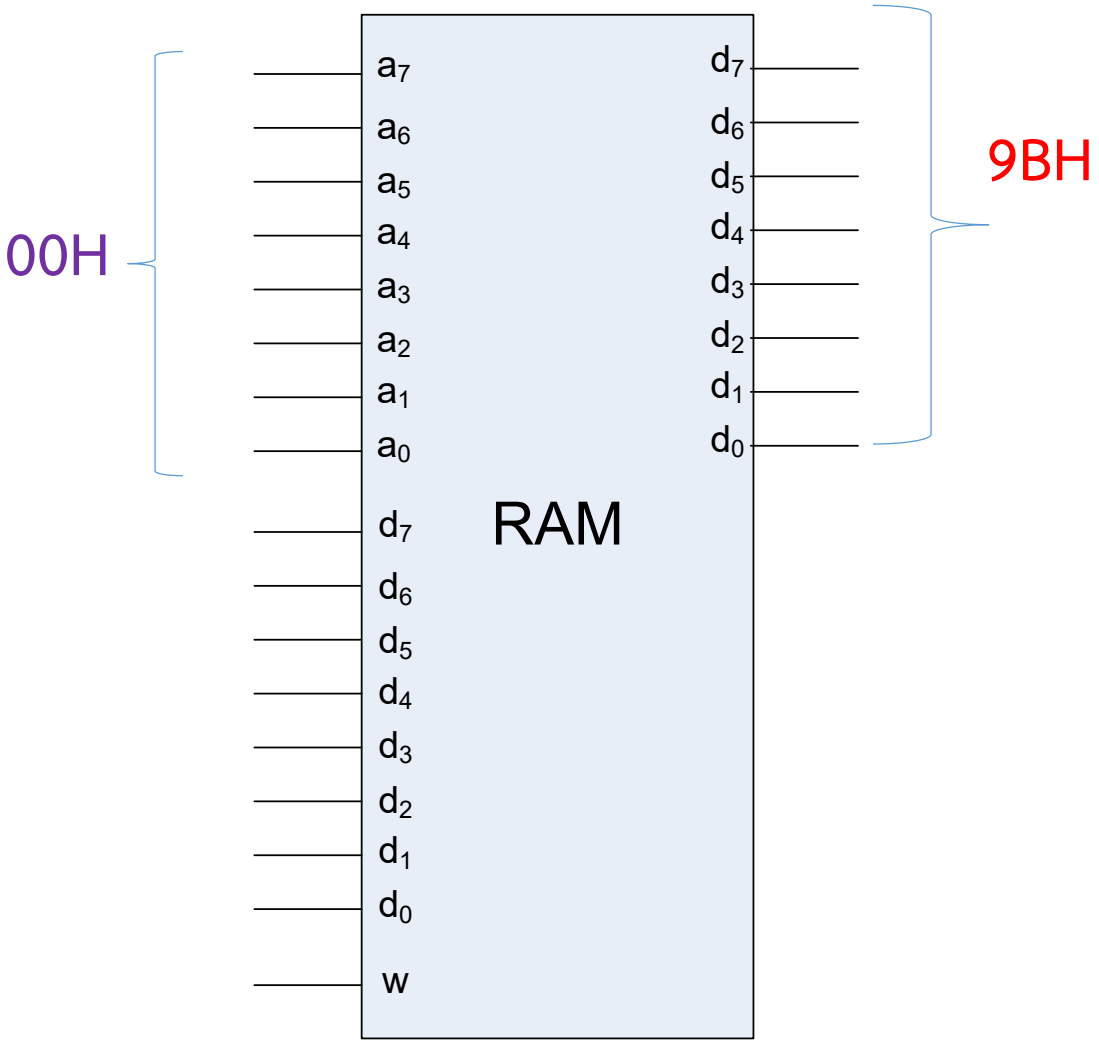
Address bit width: 8 Data bit width: 8

Data

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	9B	A1	21	EE	F0	3A	4B	81	FD	3C	7C	0D	22	55	91	A3
1	41	31	ED	4F	93	45	23	E3	2E	A8	A3	21	31	89	28	AB
2	34	11	18	98	82	65	C4	1D	BF	B9	82	5A	95	EA	68	6F
3	A1	15	8E	2E	94	3E	2C	55	7D	3D	C0	48	97	2A	50	74
4	14	D0	D8	10	3D	F9	21	DB	6C	CF	90	DA	C3	C9	4E	44
5	3F	34	7F	1B	FB	F9	17	F6	E5	C2	1A	44	A5	7C	86	94
6	69	0D	45	C4	11	57	32	A7	F4	B2	C2	C4	AE	52	91	B4
7	EF	7C	08	63	AA	34	78	1B	B9	E2	2B	7A	DE	DC	36	EF
8	49	FB	70	B5	83	89	32	E7	01	D8	27	49	37	53	47	93
9	31	65	54	29	73	23	EA	34	95	2A	0B	2F	2B	B7	C8	17
A	7B	D2	7A	B3	E1	DB	31	66	49	21	61	D2	0D	2F	E4	65
B	9A	98	49	78	87	0D	7C	26	10	3B	20	59	7D	02	BB	49
C	0F	10	CD	64	D8	94	B1	55	34	EE	DF	52	C1	18	27	65
D	45	A5	95	93	6E	50	8F	6D	56	54	A2	A5	BB	D8	E8	47
E	65	17	C3	FD	61	7F	8C	06	E6	F3	44	98	2E	29	64	F8
F	14	BA	13	D8	6D	28	DA	BF	80	5F	0E	84	F2	0C	9A	5C

ครั้งแรกที่จ่ายไฟเข้าสู่แรม ข้อมูลภายในจะเป็นเลขสุ่ม

การบันทึกข้อมูล 9A ลงในตำแหน่ง C4



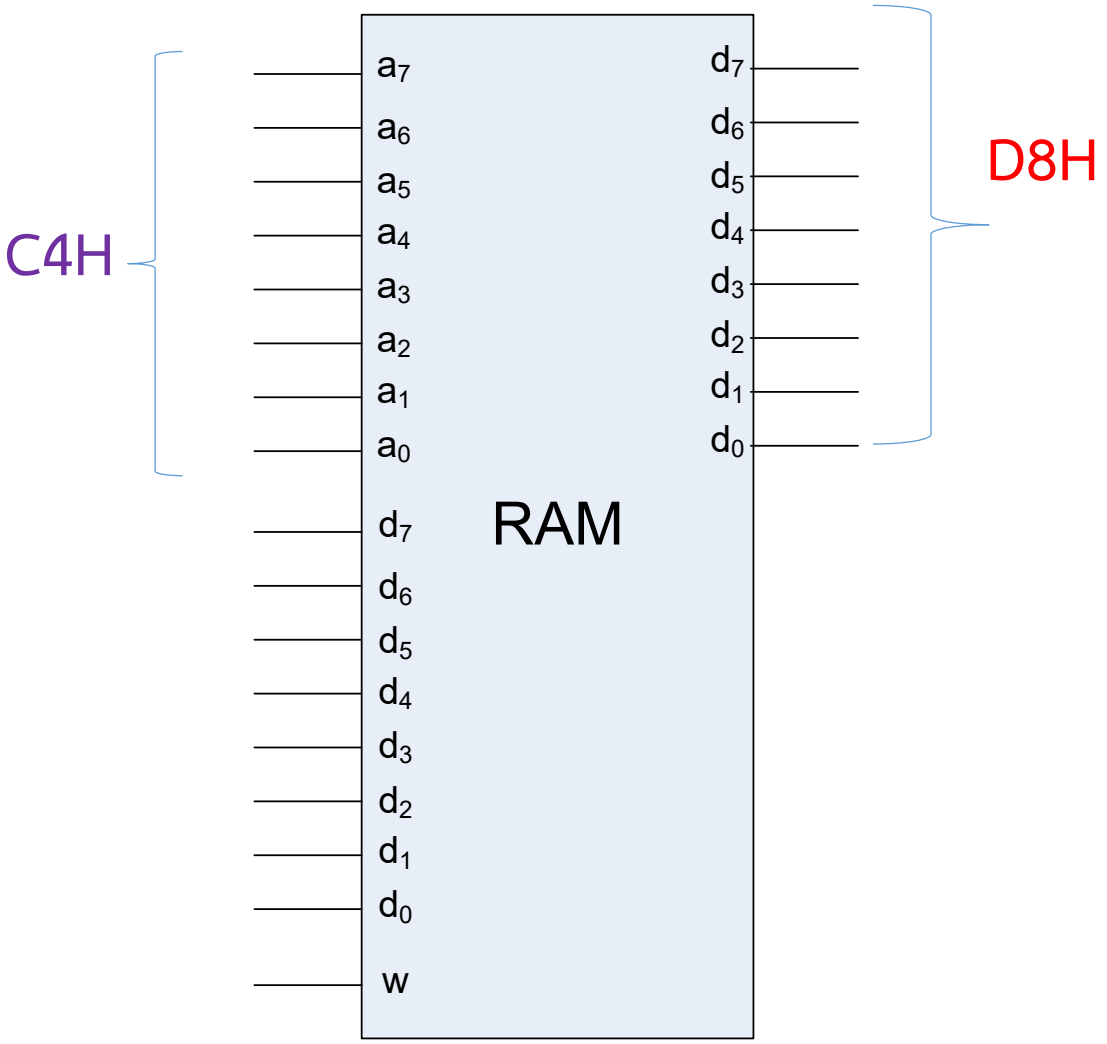
ตัวอย่างการบันทึกข้อมูลเข้าสู่แรม

Address bit width: 8 Data bit width: 8

Data

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	9B	A1	21	EE	F0	3A	4B	81	FD	3C	7C	0D	22	55	91	A3
1	41	31	ED	4F	93	45	23	E3	2E	A8	A3	21	31	89	28	AB
2	34	11	18	98	82	65	C4	1D	BF	B9	82	5A	95	EA	68	6F
3	A1	15	8E	2E	94	3E	2C	55	7D	3D	C0	48	97	2A	50	74
4	14	D0	D8	10	3D	F9	21	DB	6C	CF	90	DA	C3	C9	4E	44
5	3F	34	7F	1B	FB	F9	17	F6	E5	C2	1A	44	A5	7C	86	94
6	69	0D	45	C4	11	57	32	A7	F4	B2	C2	C4	AE	52	91	B4
7	EF	7C	08	63	AA	34	78	1B	B9	E2	2B	7A	DE	DC	36	EF
8	49	FB	70	B5	83	89	32	E7	01	D8	27	49	37	53	47	93
9	31	65	54	29	73	23	EA	34	95	2A	0B	2F	2B	B7	C8	17
A	7B	D2	7A	B3	E1	DB	31	66	49	21	61	D2	0D	2F	E4	65
B	9A	98	49	78	87	0D	7C	26	10	3B	20	59	7D	02	BB	49
C	0F	10	CD	64	D8	94	B1	55	34	EE	DF	52	C1	18	27	65
D	45	A5	95	93	6E	50	8F	6D	56	54	A2	A5	BB	D8	E8	47
E	65	17	C3	FD	61	7F	8C	06	E6	F3	44	98	2E	29	64	F8
F	14	BA	13	D8	6D	28	DA	BF	80	5F	0E	84	F2	0C	9A	5C

การบันทึกข้อมูล 9A ลงในตำแหน่ง C4



ขั้นตอนที่ 1 ตั้ง Address ไปยังตำแหน่ง 4CH

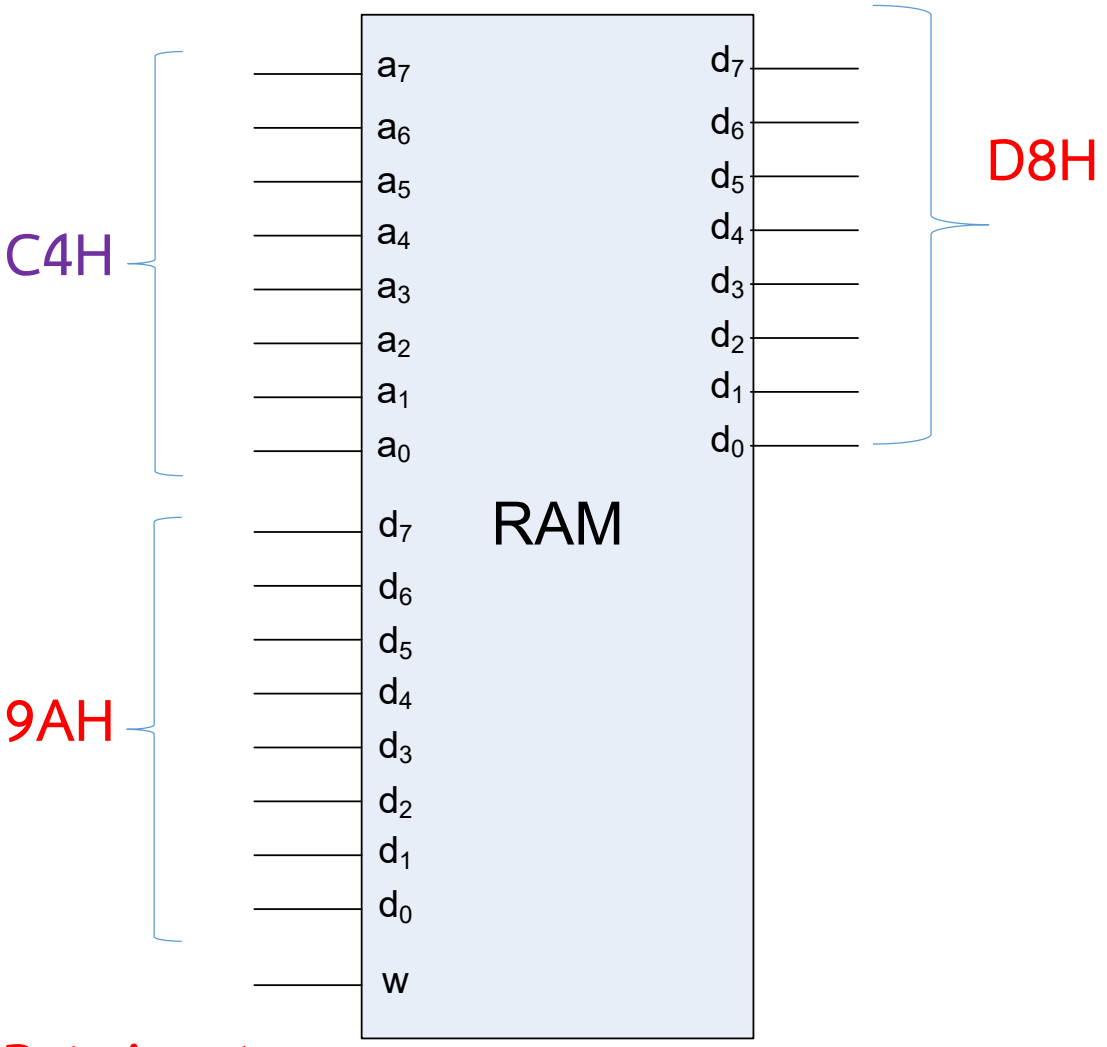
ตัวอย่างการบันทึกข้อมูลเข้าสู่แรม

การบันทึกข้อมูล 9A ลงในตำแหน่ง C4

Address bit width: 8 Data bit width: 8

Data

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	9B	A1	21	EE	F0	3A	4B	81	FD	3C	7C	0D	22	55	91	A3
1	41	31	ED	4F	93	45	23	E3	2E	A8	A3	21	31	89	28	AB
2	34	11	18	98	82	65	C4	1D	BF	B9	82	5A	95	EA	68	6F
3	A1	15	8E	2E	94	3E	2C	55	7D	3D	C0	48	97	2A	50	74
4	14	D0	D8	10	3D	F9	21	DB	6C	CF	90	DA	C3	C9	4E	44
5	3F	34	7F	1B	FB	F9	17	F6	E5	C2	1A	44	A5	7C	86	94
6	69	0D	45	C4	11	57	32	A7	F4	B2	C2	C4	AE	52	91	B4
7	EF	7C	08	63	AA	34	78	1B	B9	E2	2B	7A	DE	DC	36	EF
8	49	FB	70	B5	83	89	32	E7	01	D8	27	49	37	53	47	93
9	31	65	54	29	73	23	EA	34	95	2A	0B	2F	2B	B7	C8	17
A	7B	D2	7A	B3	E1	DB	31	66	49	21	61	D2	0D	2F	E4	65
B	9A	98	49	78	87	0D	7C	26	10	3B	20	59	7D	02	BB	49
C	0F	10	CD	64	D8	94	B1	55	34	EE	DF	52	C1	18	27	65
D	45	A5	95	93	6E	50	8F	6D	56	54	A2	A5	BB	D8	E8	47
E	65	17	C3	FD	61	7F	8C	06	E6	F3	44	98	2E	29	64	F8
F	14	BA	13	D8	6D	28	DA	BF	80	5F	0E	84	F2	0C	9A	5C



ขั้นตอนที่ 2 ตั้ง ใส่ข้อมูลที่ต้องการบันทึกลงใน Data input

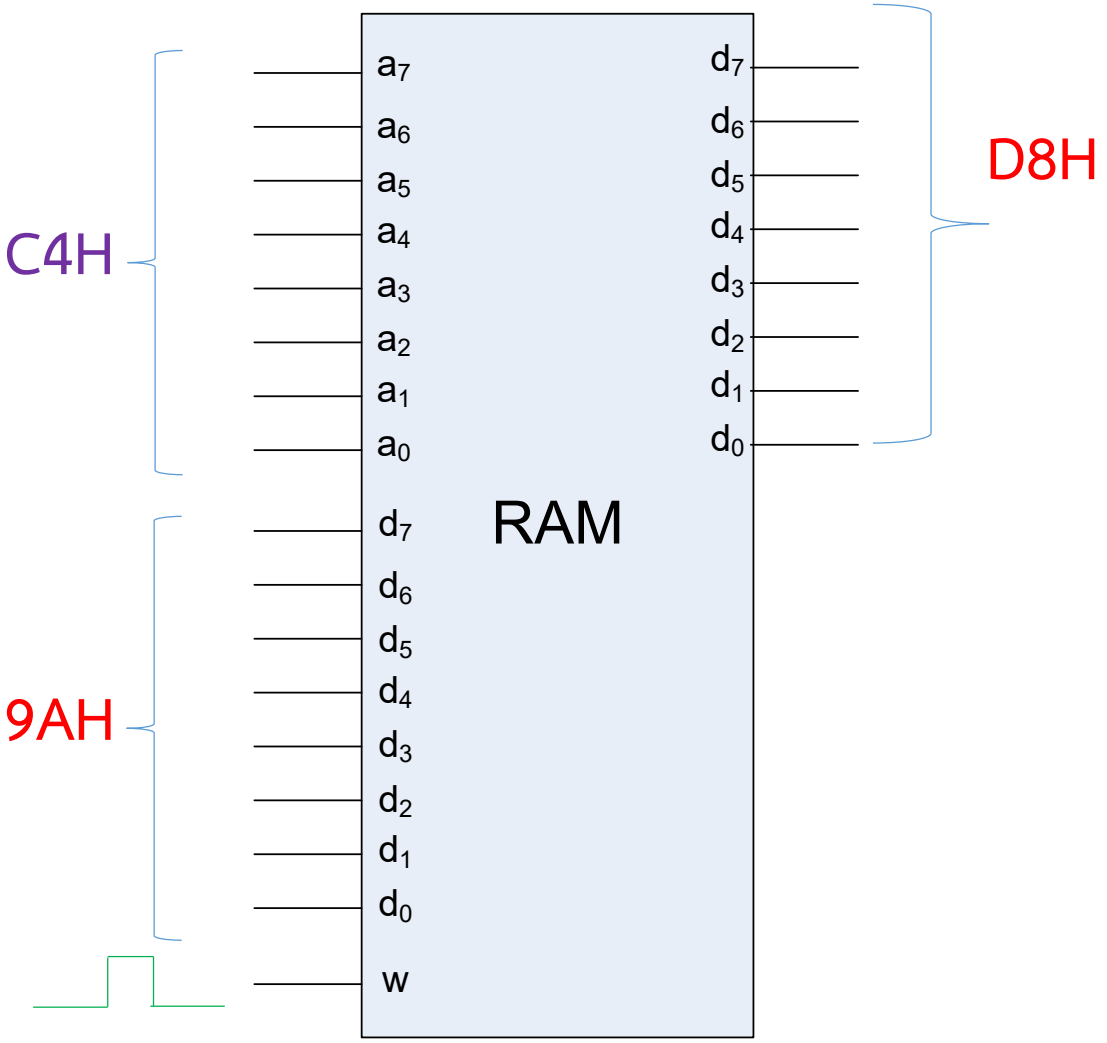
ตัวอย่างการบันทึกข้อมูลเข้าสู่แรม

Address bit width: 8 Data bit width: 8

Data

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	9B	A1	21	EE	F0	3A	4B	81	FD	3C	7C	0D	22	55	91	A3
1	41	31	ED	4F	93	45	23	E3	2E	A8	A3	21	31	89	28	AB
2	34	11	18	98	82	65	C4	1D	BF	B9	82	5A	95	EA	68	6F
3	A1	15	8E	2E	94	3E	2C	55	7D	3D	C0	48	97	2A	50	74
4	14	D0	D8	10	3D	F9	21	DB	6C	CF	90	DA	C3	C9	4E	44
5	3F	34	7F	1B	FB	F9	17	F6	E5	C2	1A	44	A5	7C	86	94
6	69	0D	45	C4	11	57	32	A7	F4	B2	C2	C4	AE	52	91	B4
7	EF	7C	08	63	AA	34	78	1B	B9	E2	2B	7A	DE	DC	36	EF
8	49	FB	70	B5	83	89	32	E7	01	D8	27	49	37	53	47	93
9	31	65	54	29	73	23	EA	34	95	2A	0B	2F	2B	B7	C8	17
A	7B	D2	7A	B3	E1	DB	31	66	49	21	61	D2	0D	2F	E4	65
B	9A	98	49	78	87	0D	7C	26	10	3B	20	59	7D	02	BB	49
C	0F	10	CD	64	D8	94	B1	55	34	EE	DF	52	C1	18	27	65
D	45	A5	95	93	6E	50	8F	6D	56	54	A2	A5	BB	D8	E8	47
E	65	17	C3	FD	61	7F	8C	06	E6	F3	44	98	2E	29	64	F8
F	14	BA	13	D8	6D	28	DA	BF	80	5F	0E	84	F2	0C	9A	5C

การบันทึกข้อมูล 9A ลงในตำแหน่ง C4



ขั้นตอนที่ 3 ป้อน clock เข้าไปยัง w

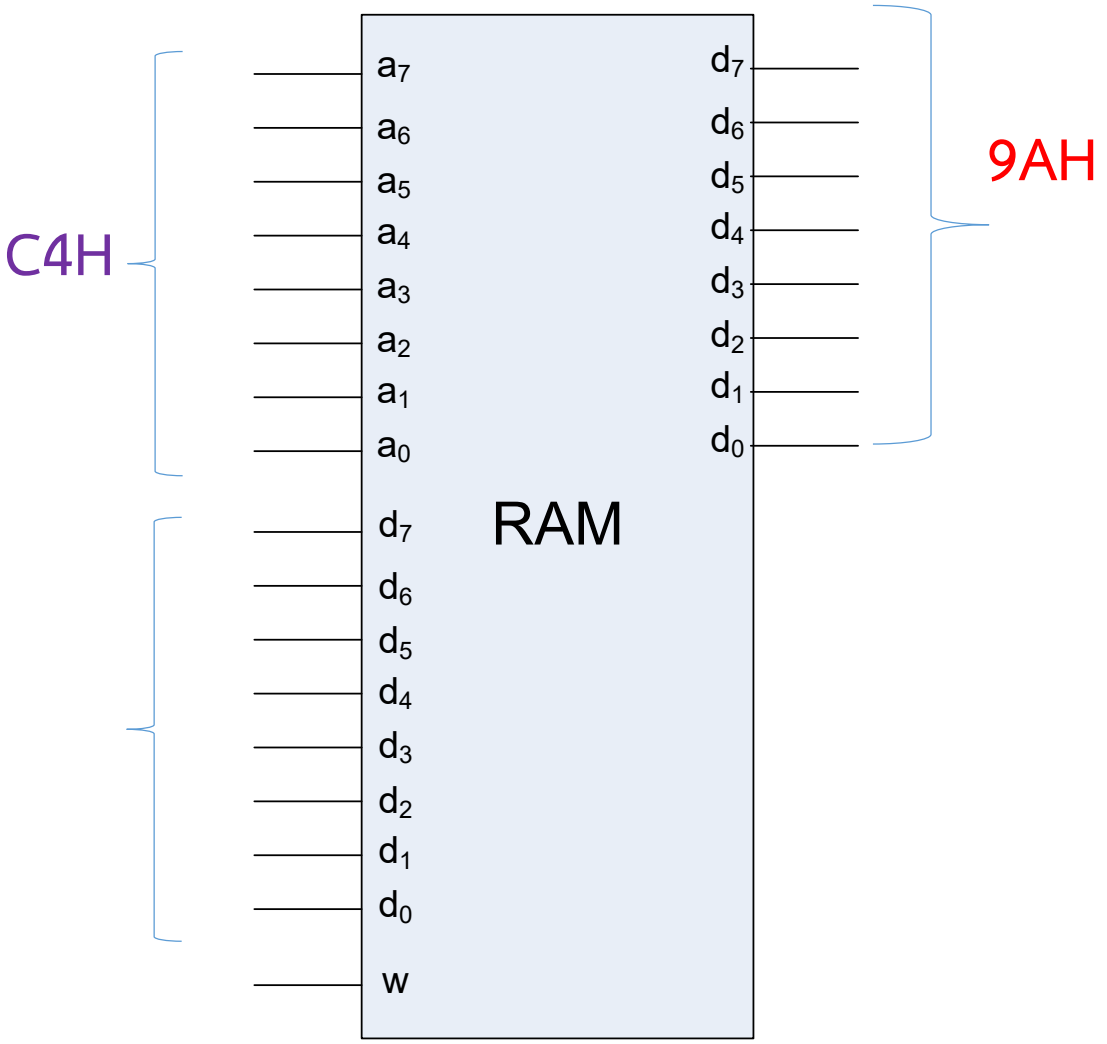
ตัวอย่างการบันทึกข้อมูลเข้าสู่แรม

Address bit width: 8 Data bit width: 8

Data

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	9B	A1	21	EE	F0	3A	4B	81	FD	3C	7C	0D	22	55	91	A3
1	41	31	ED	4F	93	45	23	E3	2E	A8	A3	21	31	89	28	AB
2	34	11	18	98	82	65	C4	1D	BF	B9	82	5A	95	EA	68	6F
3	A1	15	8E	2E	94	3E	2C	55	7D	3D	C0	48	97	2A	50	74
4	14	D0	D8	10	3D	F9	21	DB	6C	CF	90	DA	C3	C9	4E	44
5	3F	34	7F	1B	FB	F9	17	F6	E5	C2	1A	44	A5	7C	86	94
6	69	0D	45	C4	11	57	32	A7	F4	B2	C2	C4	AE	52	91	B4
7	EF	7C	08	63	AA	34	78	1B	B9	E2	2B	7A	DE	DC	36	EF
8	49	FB	70	B5	83	89	32	E7	01	D8	27	49	37	53	47	93
9	31	65	54	29	73	23	EA	34	95	2A	0B	2F	2B	B7	C8	17
A	7B	D2	7A	B3	E1	DB	31	66	49	21	61	D2	0D	2F	E4	65
B	9A	98	49	78	87	0D	7C	26	10	3B	20	59	7D	02	BB	49
C	0F	10	CD	64	9A	94	B1	55	34	EE	DF	52	C1	18	27	65
D	45	A5	95	93	6E	50	8F	6D	56	54	A2	A5	BB	D8	E8	47
E	65	17	C3	FD	61	7F	8C	06	E6	F3	44	98	2E	29	64	F8
F	14	BA	13	D8	6D	28	DA	BF	80	5F	0E	84	F2	0C	9A	5C

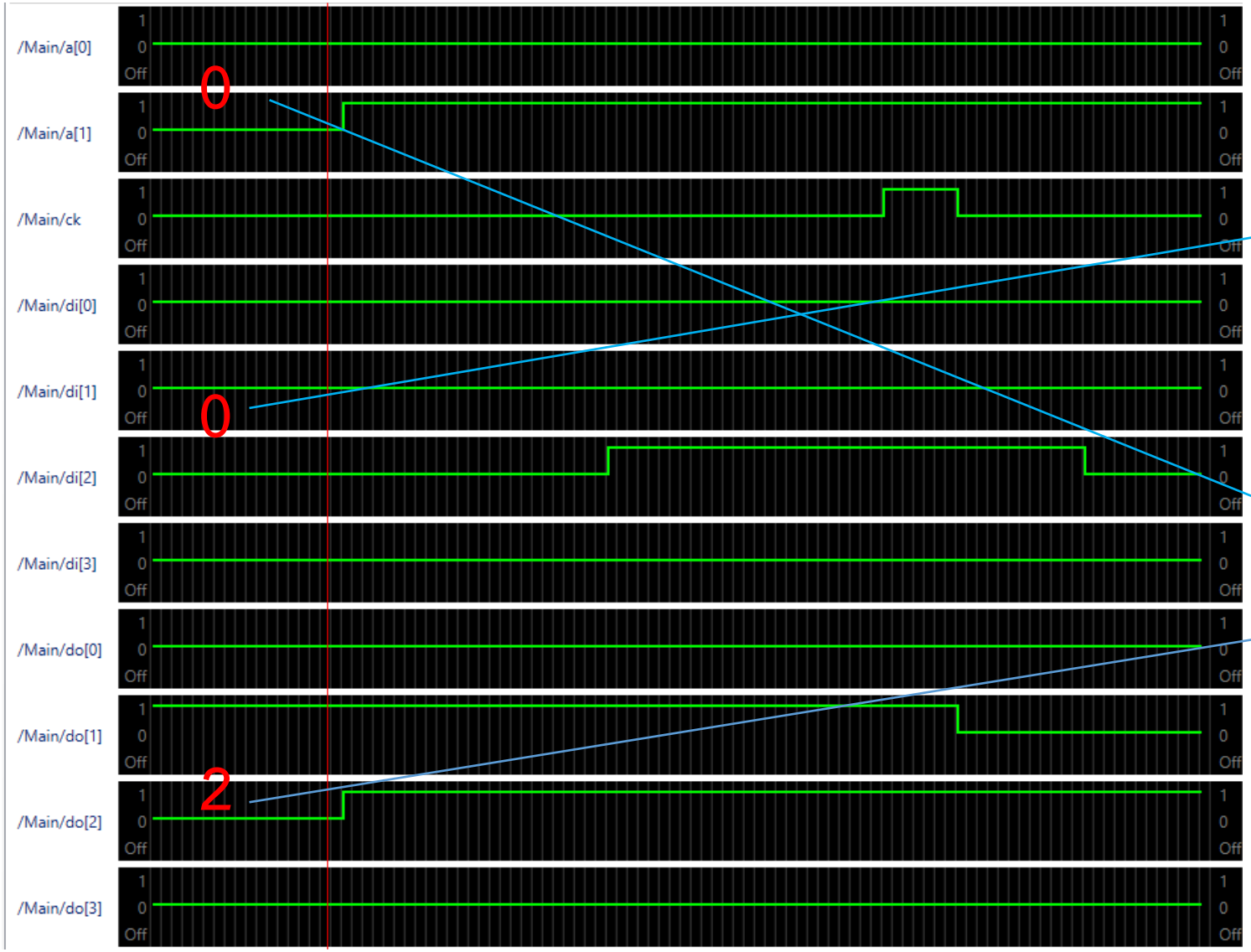
การบันทึกข้อมูล 9A ลงในตำแหน่ง C4

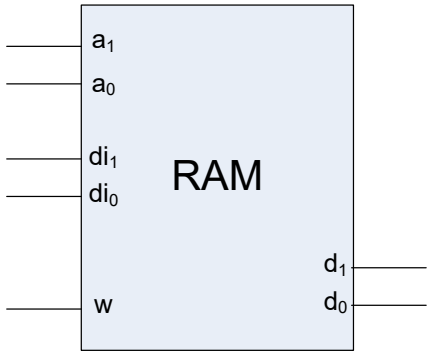


เสร็จสิ้นขั้นตอนการบันทึกข้อมูล

ตัวอย่างการบันทึกข้อมูลเข้าสู่แรม

การบันทึกข้อมูล 4 ลงในตำแหน่ง 2





Address bit width: 2 Data bit width: 4

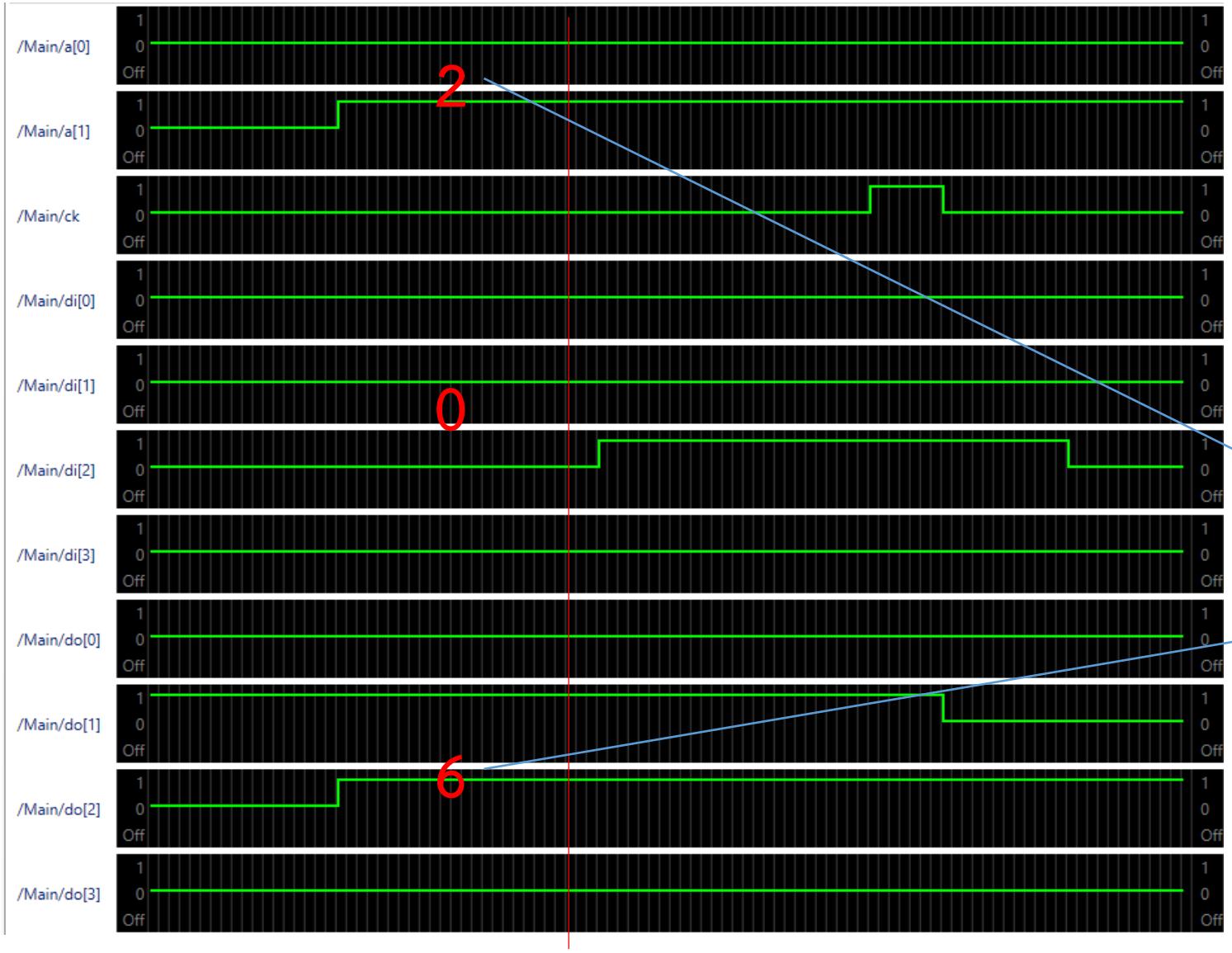
Data

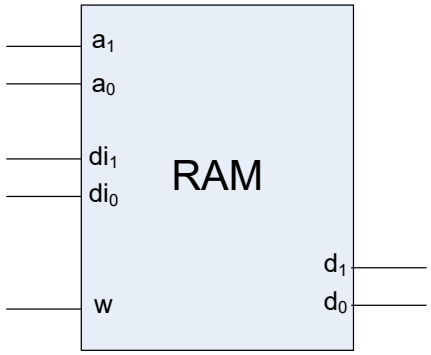
	0	1	2	3
0	2	A	6	9

จ่ายไฟเข้าแรม ยังไม่ได้ป้อนอะไร

ตัวอย่างการบันทึกข้อมูลเข้าสู่แรม

การบันทึกข้อมูล 4 ลงในตำแหน่ง 2





Address bit width: 2 Data bit width: 4

Data

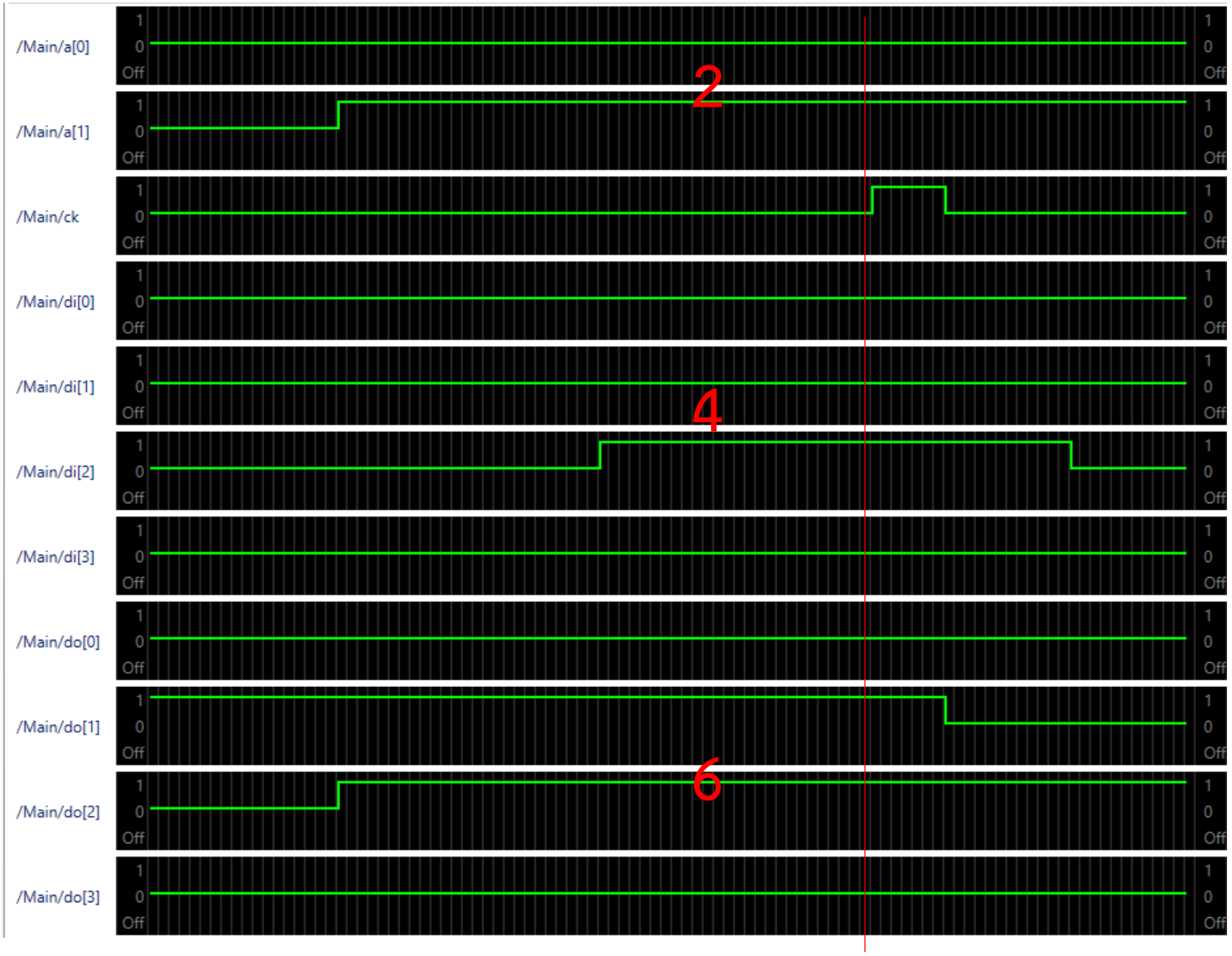
	0	1	2	3
0	2	A	6	9

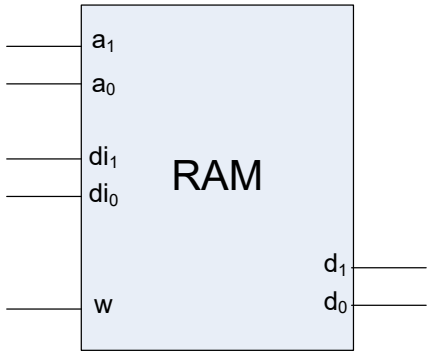
Input:ป้อน Address ที่จะเขียน

Output:ข้อมูลที่ค้างอยู่ในแรม

ตัวอย่างการบันทึกข้อมูลเข้าสู่แรม

การบันทึกข้อมูล 4 ลงในตำแหน่ง 2





Address bit width: 2 Data bit width: 4

Data

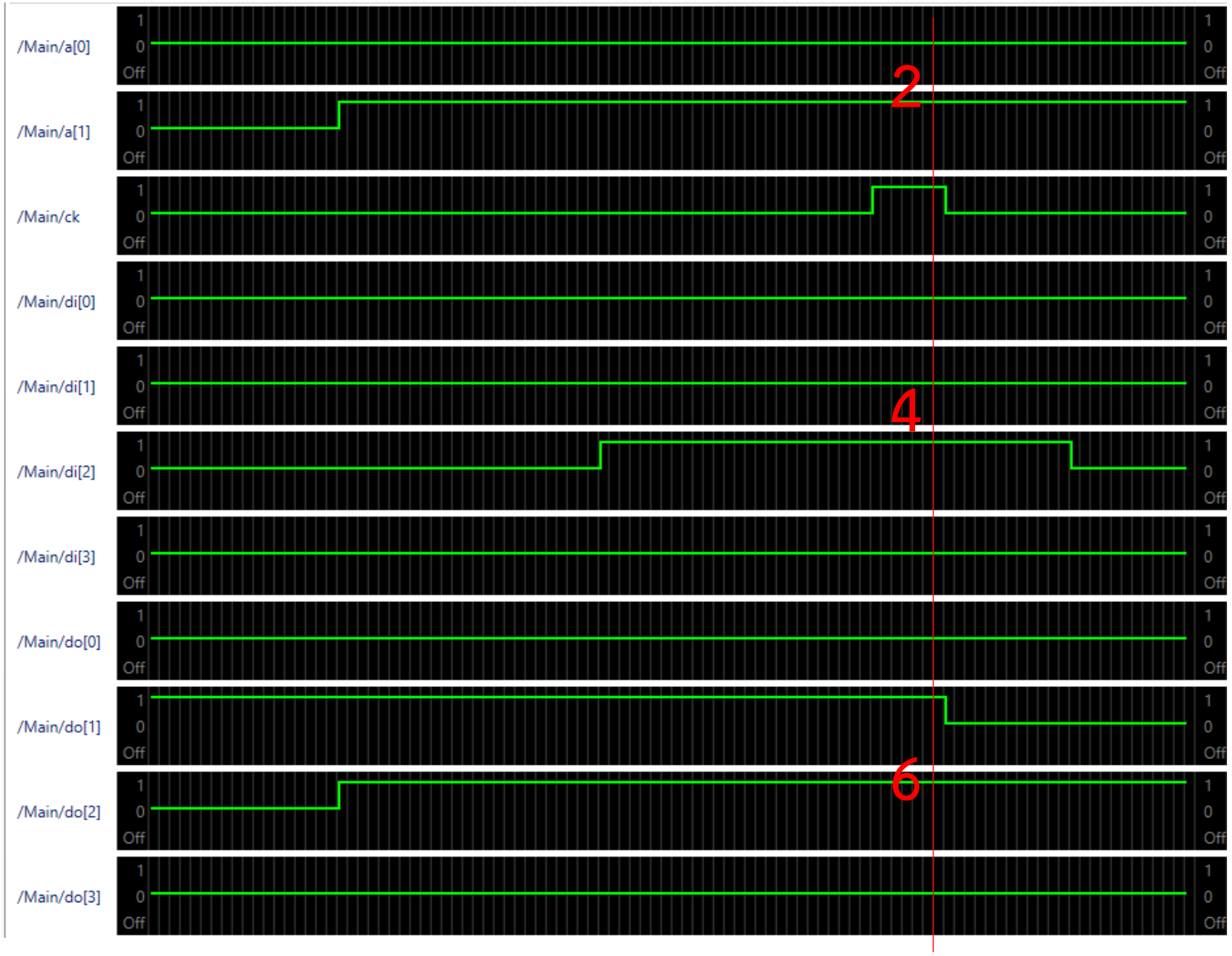
	0	1	2	3
0	2	A	6	9

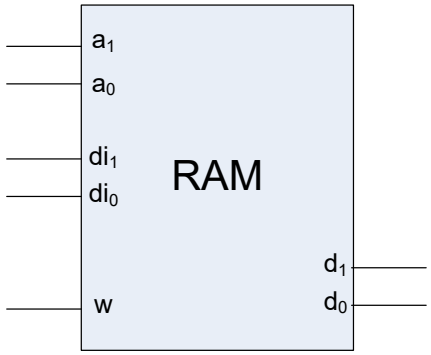
Input:ข้อมูลที่จะเขียน

Output:ข้อมูลที่ค้างอยู่ในแรม

ตัวอย่างการบันทึกข้อมูลเข้าสู่แรม

การบันทึกข้อมูล 4 ลงในตำแหน่ง 2





Address bit width: 2 Data bit width: 4

Data

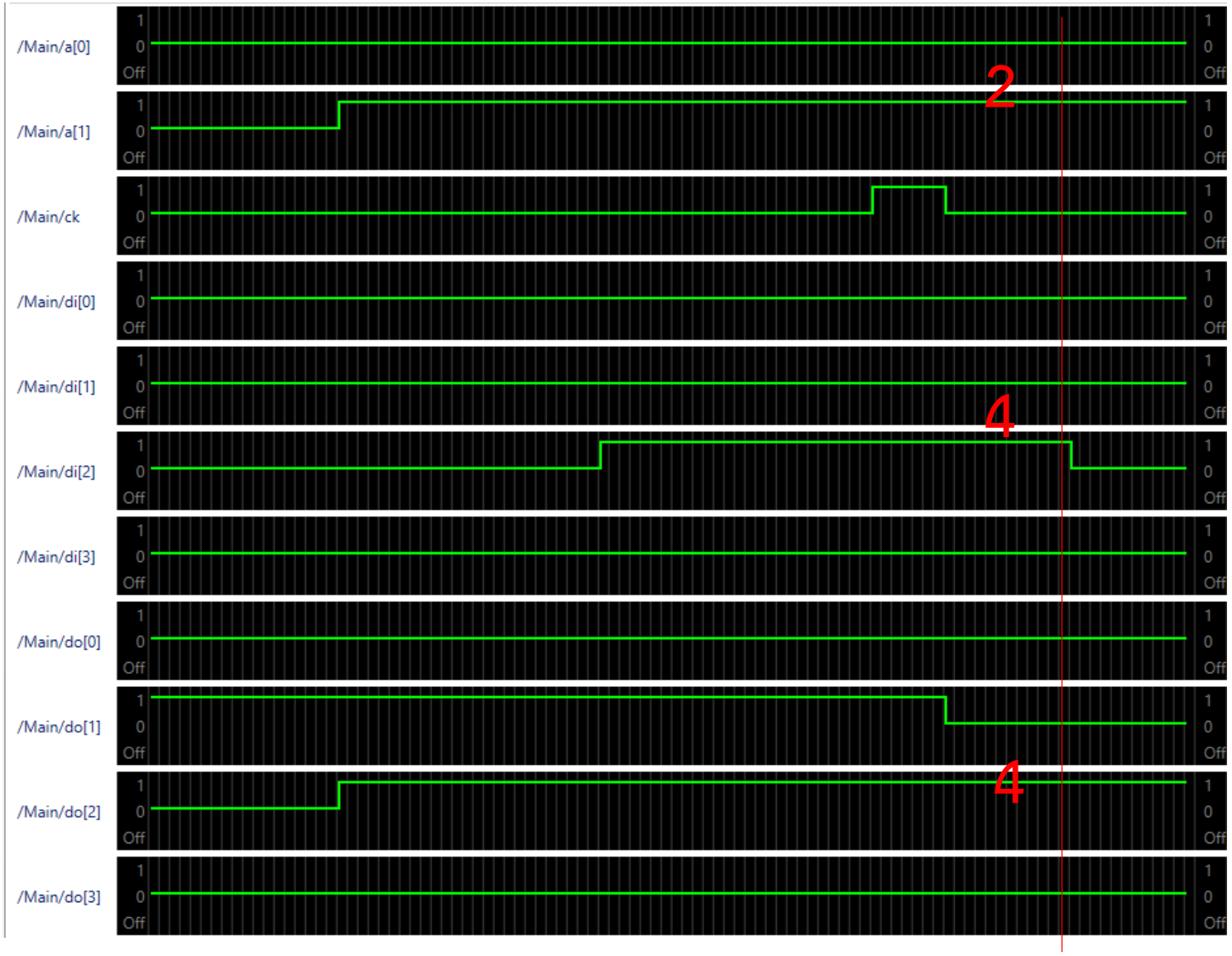
	0	1	2	3
0	2	A	6	9

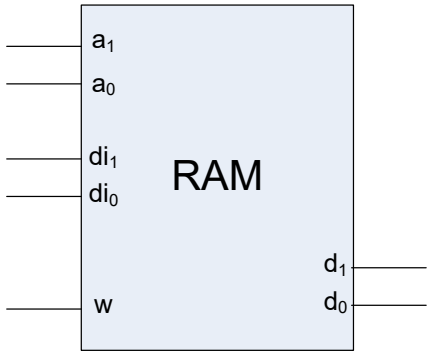
Input:นาฬิกา 0→1

Output:ข้อมูลที่ค้างอยู่ในแรม

ตัวอย่างการบันทึกข้อมูลเข้าสู่แรม

การบันทึกข้อมูล 4 ลงในตำแหน่ง 2





Address bit width: 2 Data bit width: 4

Data

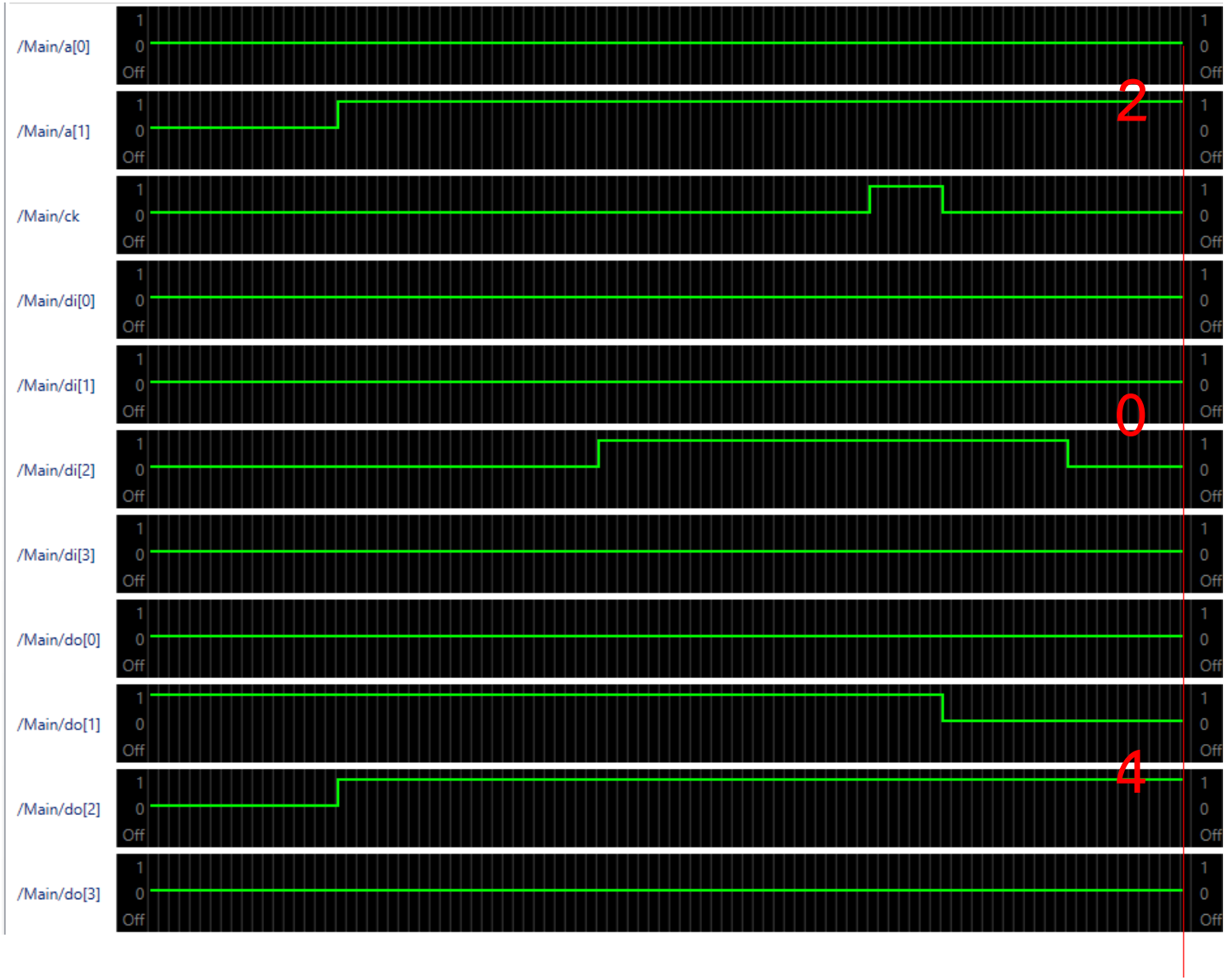
	0	1	2	3
0	2	A	4	9

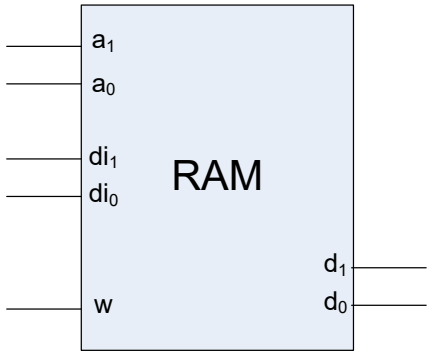
Input: นาฬิกา 1→0

Output: ข้อมูลถูกบันทึก

ตัวอย่างการบันทึกข้อมูลเข้าสู่แรม

การบันทึกข้อมูล 4 ลงในตำแหน่ง 2





Address bit width: 2 Data bit width: 4

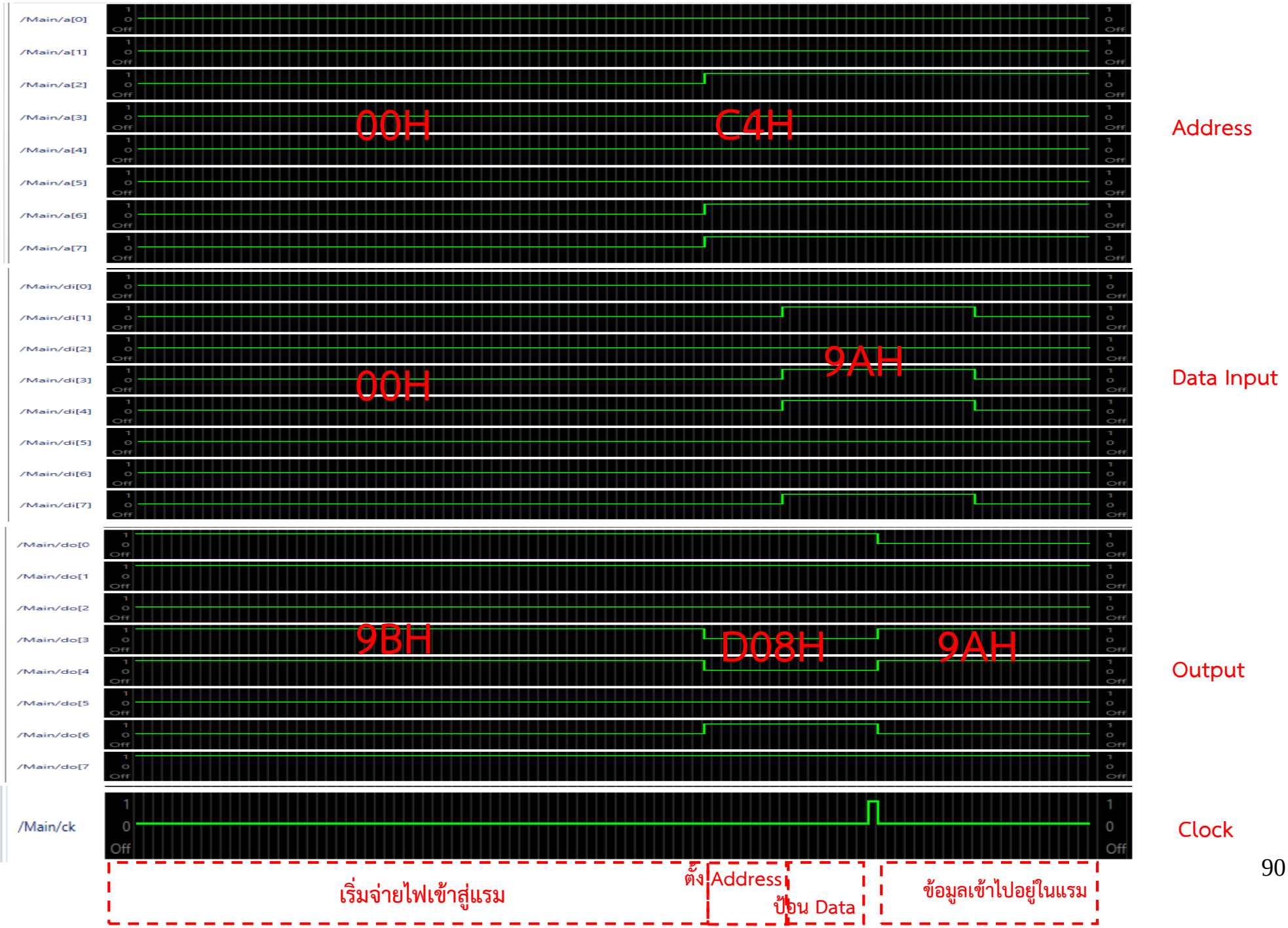
Data

	0	1	2	3
0	2	A	4	9

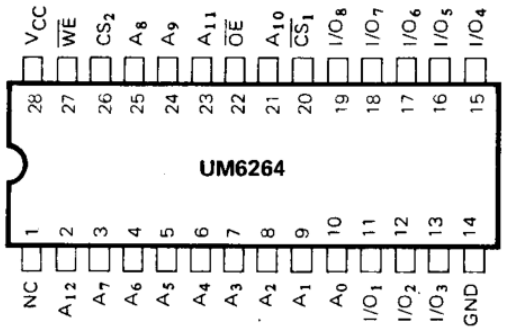
Input: ข้อมูลเปลี่ยนไปแล้ว

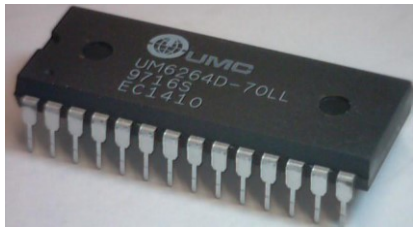
Output:ข้อมูลเดิมยังค้างอยู่

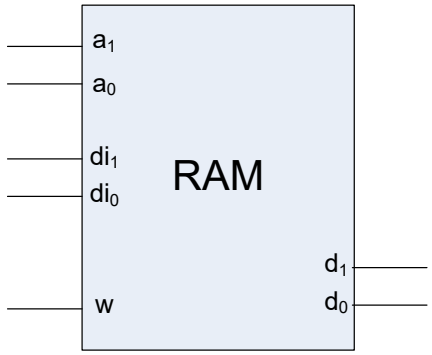
การบันทึกข้อมูล 9A ลงในตำแหน่ง C4



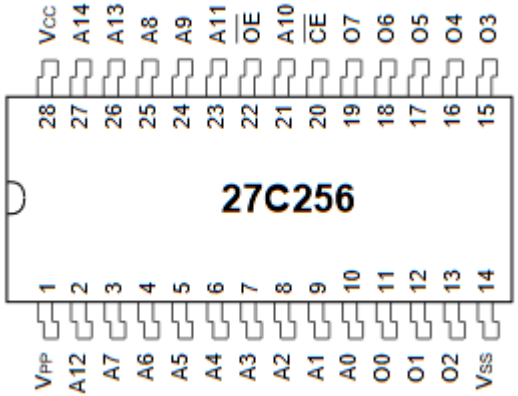
แรม : อ่านข้อมูล และบันทึกข้อมูลได้ ควบคุมด้วยสัญญาณ w

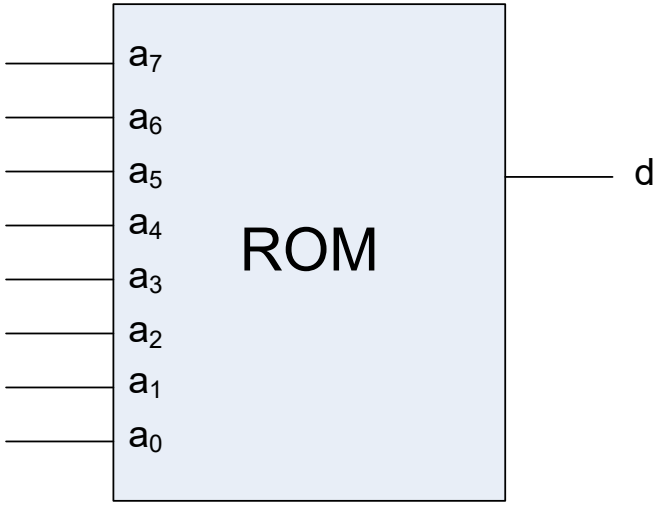






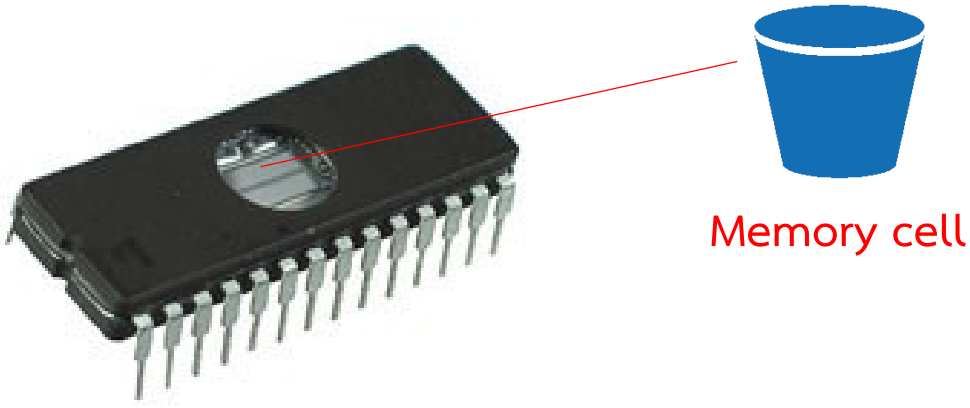
รอม : อ่านข้อมูลได้อย่างเดียว ไม่มีสัญญาณ w



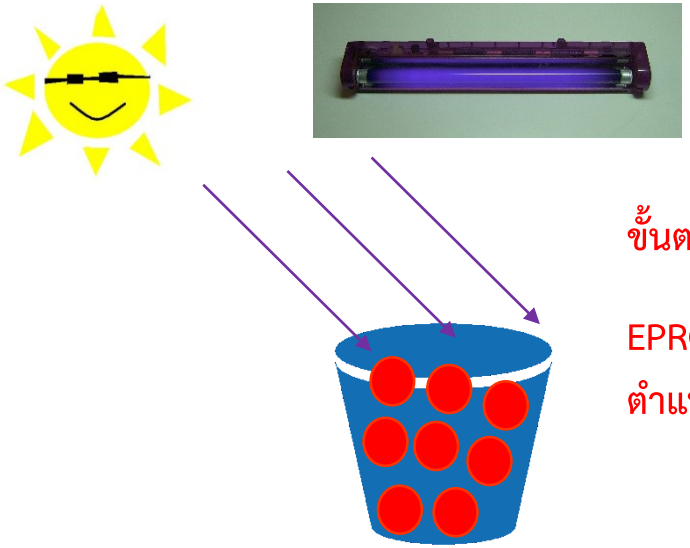


ถ้าเราบันทึกข้อมูลใส่รอมไม่ได้ แล้วข้อมูลในรอมมันเข้าไปอยู่ในนั้นได้อย่างไร ?

EPROM (Electrically Erasable Programmable Read-Only Memory)



ขั้นตอนที่ 1) เติม Electron ลงไปให้เต็ม Memory cell โดยการส่องด้วยแสง UV ขั้นตอนนี้เรียกว่าการ “ล้างข้อมูล”



ขั้นตอนนี้ จะทำให้ทุกบิตมีค่าเป็น “1” ทั้งหมด

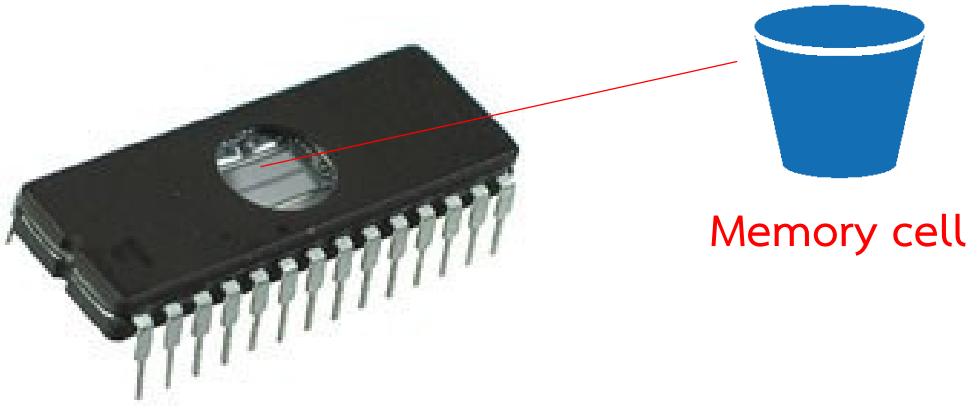
EPROM ที่ไม่มีข้อมูล จะอ่านค่าได้เป็น “FF” ในทุกตำแหน่ง

EPROM (Electrically Erasable Programmable Read-Only Memory)

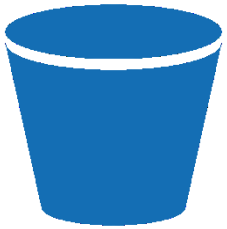
Memory cell

ขั้นตอนที่ 2) หากต้องการเปลี่ยนบิตไหนเป็น “0”ให้ทำการจ่าย Pulse ความถี่และแรงดันสูง ไปยังตำแหน่งนั้น เพื่อลบ Electron ออกไปจาก Memory cell นั้น

EPROM (Electrically Erasable Programmable Read-Only Memory)



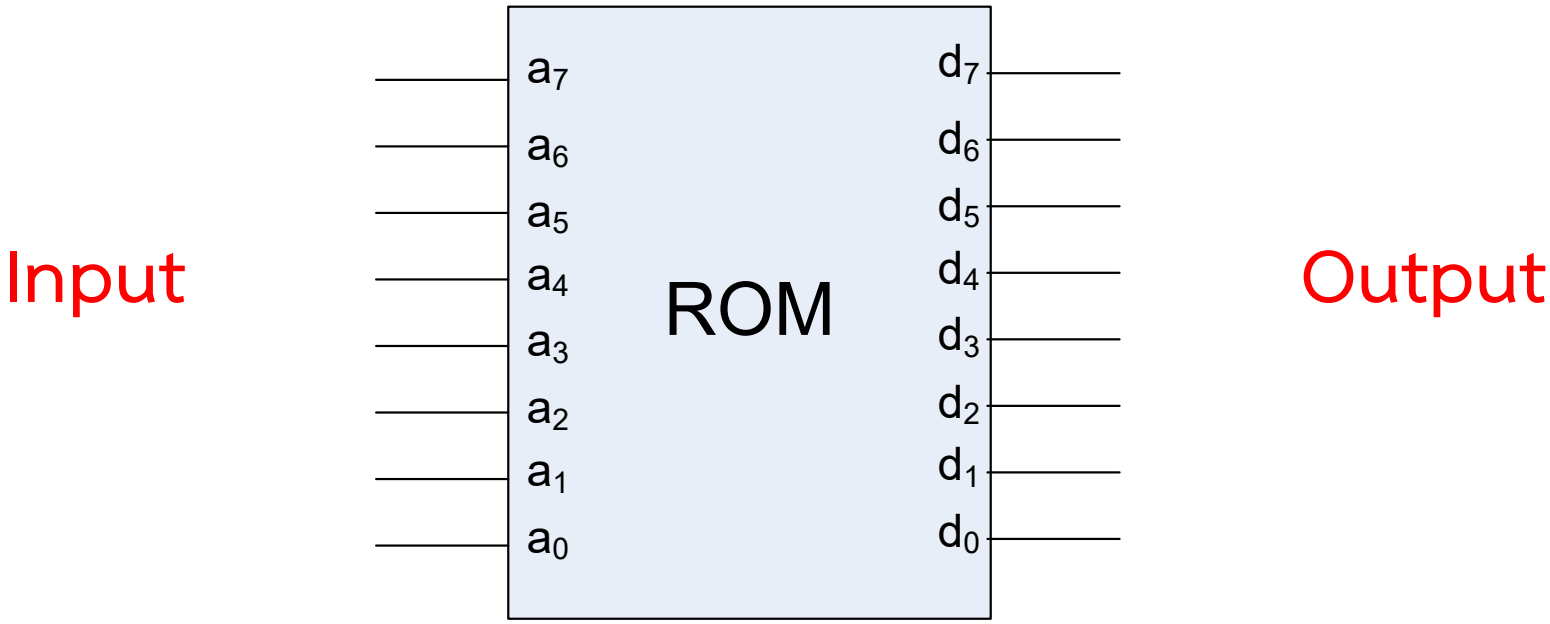
ขั้นตอนที่ 2) หากต้องการเปลี่ยนบิตไหนเป็น “0”ให้ทำการจ่าย Pulse ความถี่และแรงดันสูง ไปยังตำแหน่งนั้น เพื่อลบ Electron ออกไปจาก Memory cell นั้น



เมื่อไม่มี Electron บิตนั้นจะอ่านค่าได้เป็น “0”

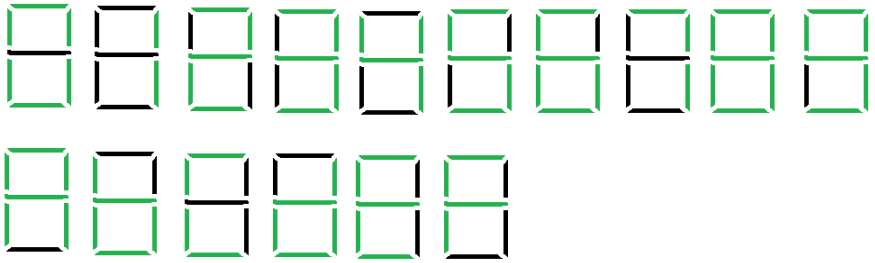
การออกแบบวงจรด้วยรอม

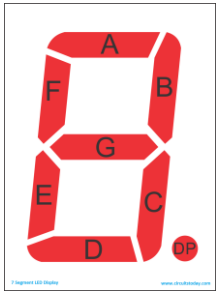
รอมสามารถนำไปประยุกต์ใช้ในการสร้างวงจร Combination อย่างง่ายได้ โดยการบันทึกตารางค่าความจริงลงไปในตัวรอม โดยใช้ Input เป็นตำแหน่งของหน่วยความจำ



การออกแบบวงจรด้วยรอม

ให้ทำการออกแบบวงจรแปลงเลขฐาน 2 ขนาด 4 บิต ให้เป็นเลขฐาน 16 ด้วยหลอด LED แบบ 7 ส่วนซึ่งมีรูปแบบการแสงผลดังนี้ (สีเขียวหมายถึงหลอดติด)





	Input				Output						
	d3	d2	d1	d0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0

	Input				Output						
	d3	d2	d1	d0	a	b	c	d	e	f	g
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
A	1	0	1	0	1	1	1	0	1	1	1
B	1	0	1	1	0	0	1	1	1	1	1
C	1	1	0	0	1	0	0	1	1	1	0
D	1	1	0	1	0	1	1	1	1	0	1
E	1	1	1	0	1	0	0	1	1	1	1
F	1	1	1	1	1	0	0	0	1	1	1

การออกแบบวงจรด้วยรอม

	Input				Output							Output
	d3	d2	d1	d0	a	b	c	d	e	f	g	HEX
0	0	0	0	0	1	1	1	1	1	1	0	7E
1	0	0	0	1	0	1	1	0	0	0	0	30
2	0	0	1	0	1	1	0	1	1	0	1	6D
3	0	0	1	1	1	1	1	1	0	0	1	79
4	0	1	0	0	0	1	1	0	0	1	1	33
5	0	1	0	1	1	0	1	1	0	1	1	5B
6	0	1	1	0	1	0	1	1	1	1	1	5F
7	0	1	1	1	1	1	1	0	0	0	0	70

	Input				Output							Output
	d3	d2	d1	d0	a	b	c	d	e	f	g	HEX
8	1	0	0	0	1	1	1	1	1	1	1	7F
9	1	0	0	1	1	1	1	1	0	1	1	7B
A	1	0	1	0	1	1	1	0	1	1	1	77
B	1	0	1	1	0	0	1	1	1	1	1	1F
C	1	1	0	0	1	0	0	1	1	1	0	4E
D	1	1	0	1	0	1	1	1	1	0	1	3D
E	1	1	1	0	1	0	0	1	1	1	1	4F
F	1	1	1	1	1	0	0	0	1	1	1	47

แปลง Input เป็นเลขฐาน 16

แปลง Output เป็นเลขฐาน 16

การออกแบบวงจรด้วยรอม

	Input				Output						
	d3	d2	d1	d0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0

	Input				Output						
	d3	d2	d1	d0	a	b	c	d	e	f	g
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
A	1	0	1	0	1	1	1	0	1	1	1
B	1	0	1	1	0	0	1	1	1	1	1
C	1	1	0	0	1	0	0	1	1	1	0
D	1	1	0	1	0	1	1	1	1	0	1
E	1	1	1	0	1	0	0	1	1	1	1
F	1	1	1	1	1	0	0	0	1	1	1

Output
HEX
7E
30
6D
79
33
5B
5F
70

Output
HEX
7F
7B
77
1F
4E
3D
4F
47

กำหนดให้ Input เป็น Address

Output เป็น Data

Address bit width: 8

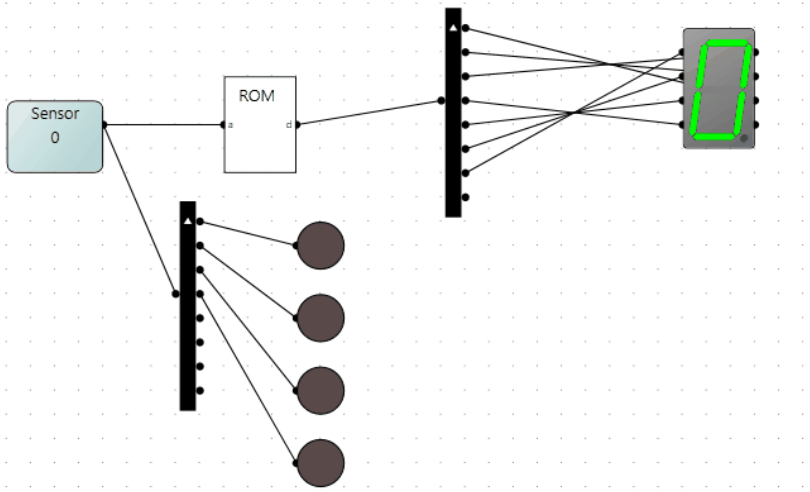
Data bit width: 8

Data

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	7E	30	6D	79	33	5B	5F	70	7F	7B	77	1F	4E	3D	4F	47
1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

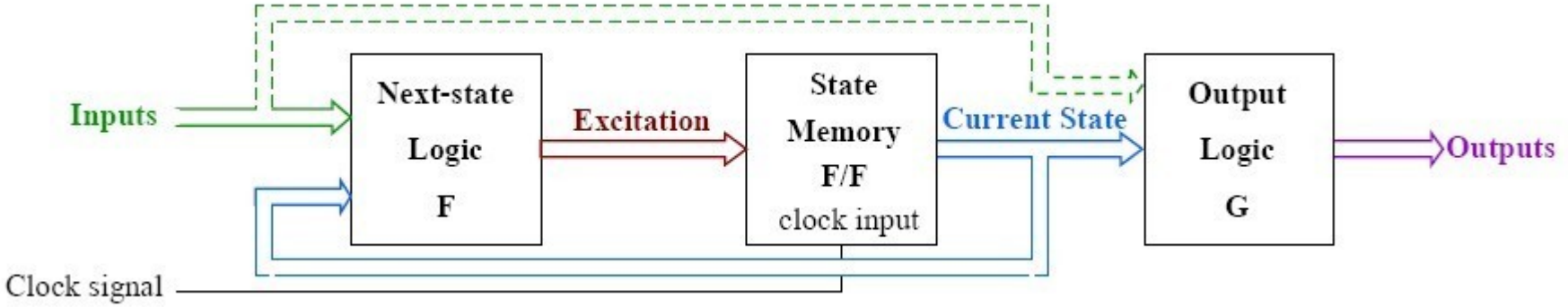
การออกแบบวงจรด้วยรอม

	Input				Output							Output
	d3	d2	d1	d0	a	b	c	d	e	f	g	HEX
0	0	0	0	0	1	1	1	1	1	1	0	7E
1	0	0	0	1	0	1	1	0	0	0	0	30
2	0	0	1	0	1	1	0	1	1	0	1	6D
3	0	0	1	1	1	1	1	1	0	0	1	79
4	0	1	0	0	0	1	1	0	0	1	1	33
5	0	1	0	1	1	0	1	1	0	1	1	5B
6	0	1	1	0	1	0	1	1	1	1	1	5F
7	0	1	1	1	1	1	1	0	0	0	0	70
	Input				Output							Output
	d3	d2	d1	d0	a	b	c	d	e	f	g	HEX
8	1	0	0	0	1	1	1	1	1	1	1	7F
9	1	0	0	1	1	1	1	1	0	1	1	7B
A	1	0	1	0	1	1	1	0	1	1	1	77
B	1	0	1	1	0	0	1	1	1	1	1	1F
C	1	1	0	0	1	0	0	1	1	1	0	4E
D	1	1	0	1	0	1	1	1	1	0	1	3D
E	1	1	1	0	1	0	0	1	1	1	1	4F
F	1	1	1	1	1	0	0	0	1	1	1	47



เครื่องจักรสถานะจำกัด

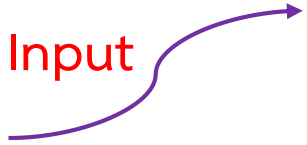
เครื่องจักรสถานะจำกัด หรือ ไฟไนต์สเตตแมชชีน (อังกฤษ: finite state machine) คือวงจรเชิงลำดับซึ่งออกแบบเป็นสถานะการทำงาน (state) ของวงจรออกเป็นหลายๆ สถานะ แต่ละสถานะมีลอจิกการทำงานที่ต่างกัน เพื่อกำเนิดค่าเอาต์พุตและค่าสถานะถัดไป มีสัญญาณสถานะที่กำหนดว่าสถานะปัจจุบันเป็นสถานะไหน สัญญาณของสถานะจะถูกเก็บไว้ในเรจิสเตอร์ ดังนั้นสถานะจะสามารถเปลี่ยนแปลงได้ที่ขอบขาของ clock เท่านั้น



การออกแบบเครื่องจักรสถานะจำกัด จะใช้แผนผังสถานะ (State diagram) ในการออกแบบ

สถานะ

(State)

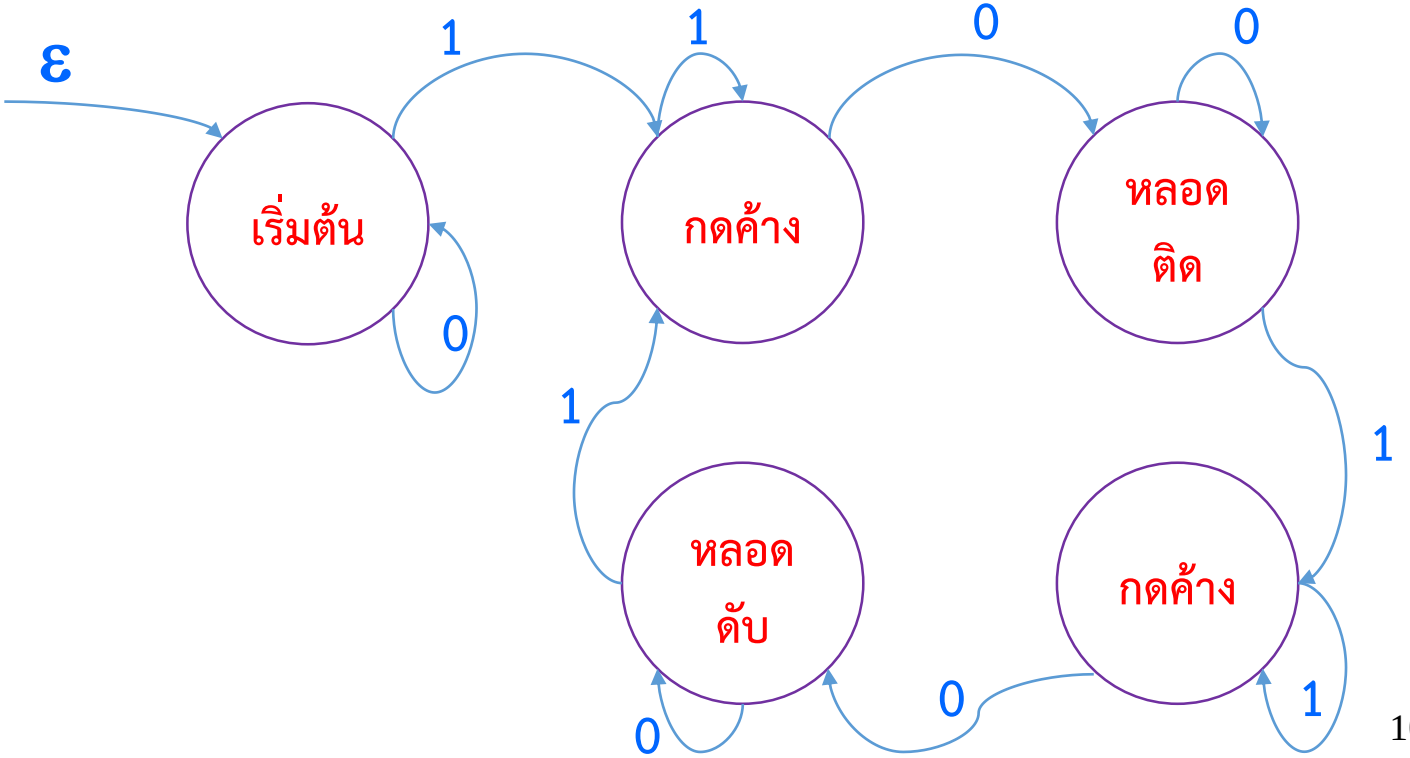


การเปลี่ยนแปลง

(Transition)

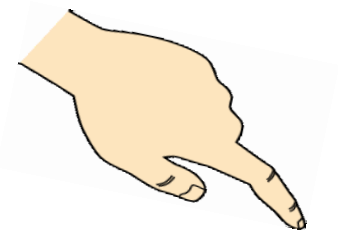
ตัวอย่างแผนผังสถานะ ของวงจรวงจรเปิด ปิด หลอดไฟ

I (ไม่กด=0,กด=1)



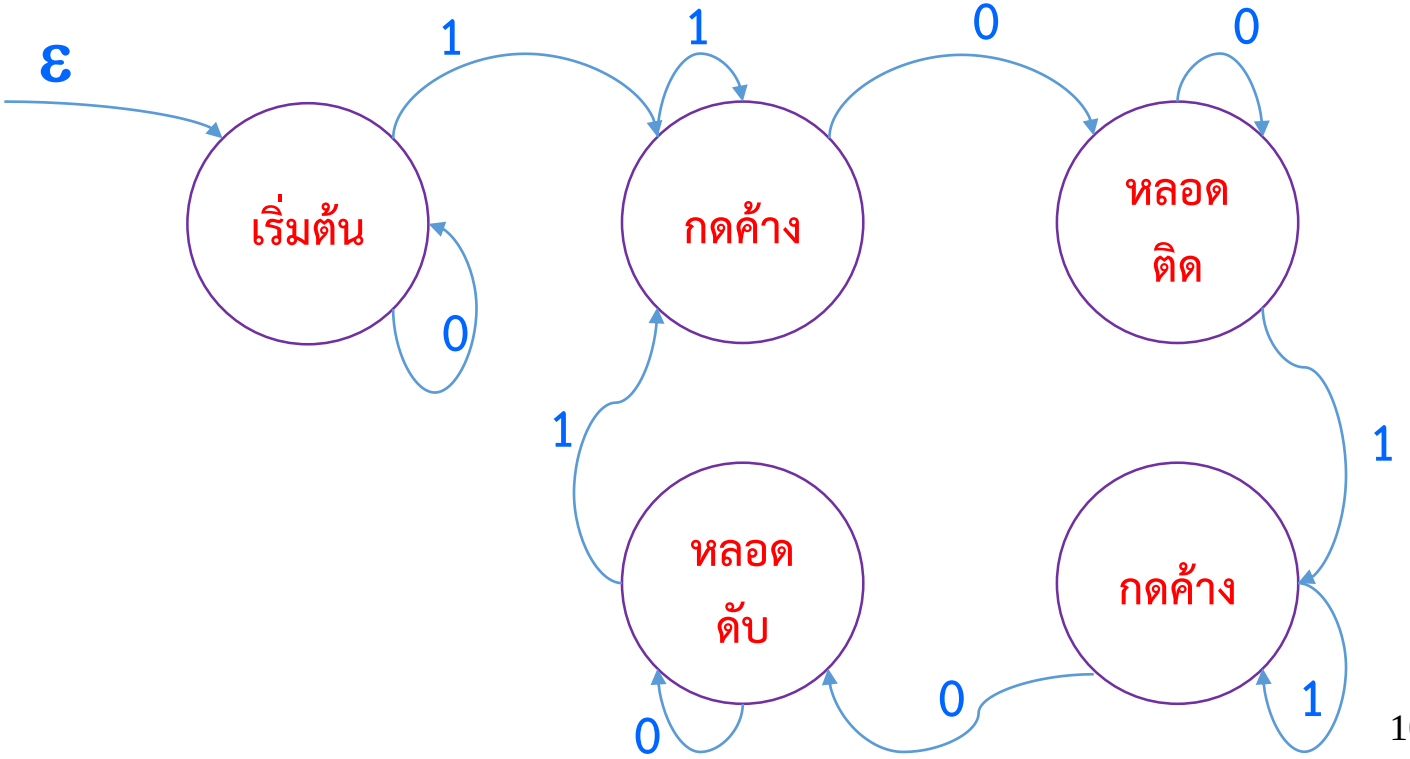
การออกแบบเครื่องจักรสถานะจำกัด จะใช้แผนผังสถานะ (State diagram) ในการออกแบบ

ตัวอย่างแผนผังสถานะ ของวงจรวงจรเปิด ปิด หลอดไฟ



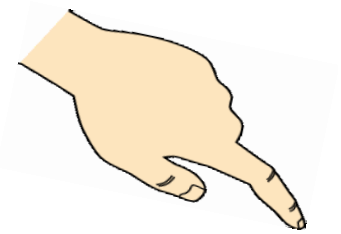
I (ไม่กด=0,กด=1)

I = 0



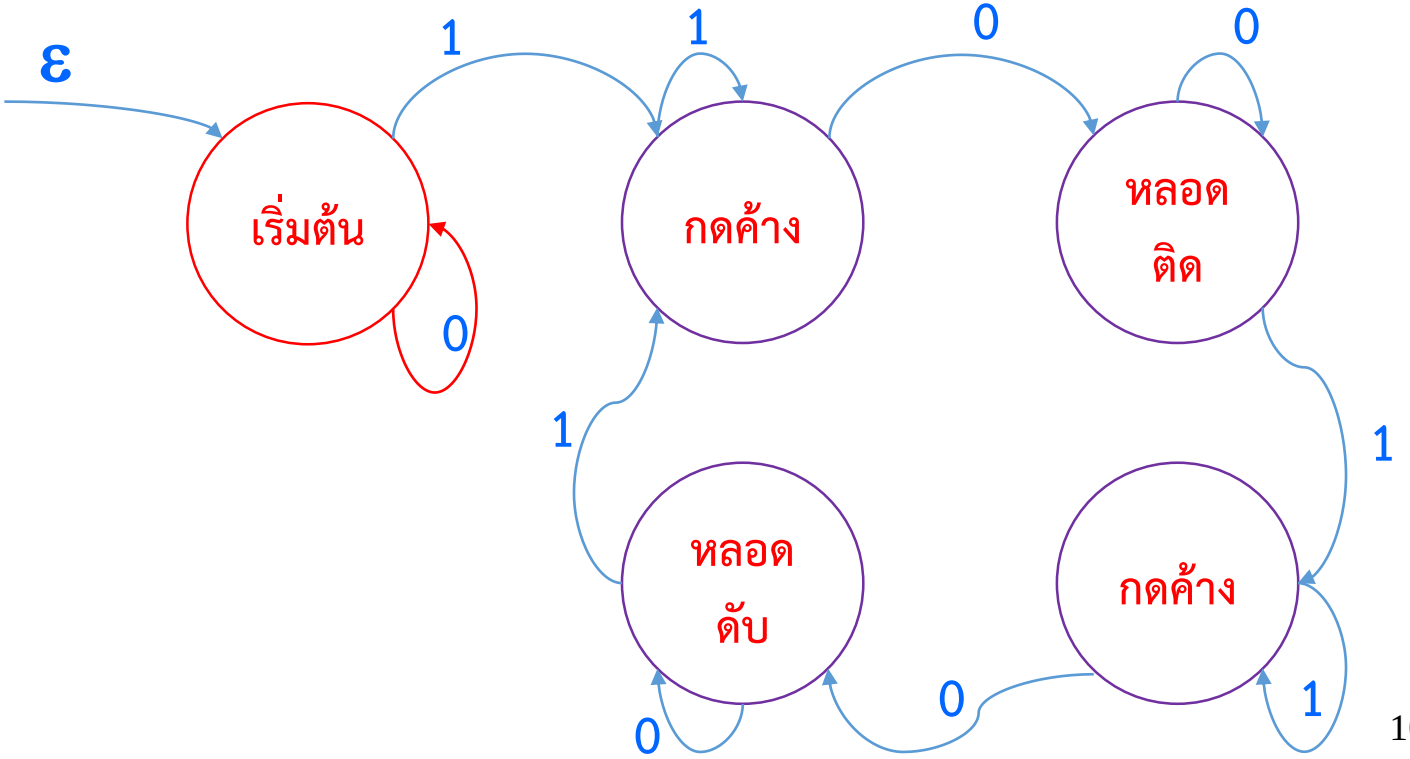
การออกแบบเครื่องจักรสถานะจำกัด จะใช้แผนผังสถานะ (State diagram) ในการออกแบบ

ตัวอย่างแผนผังสถานะ ของวงจรวงจรเปิด ปิด หลอดไฟ



I (ไม่กด=0,กด=1)

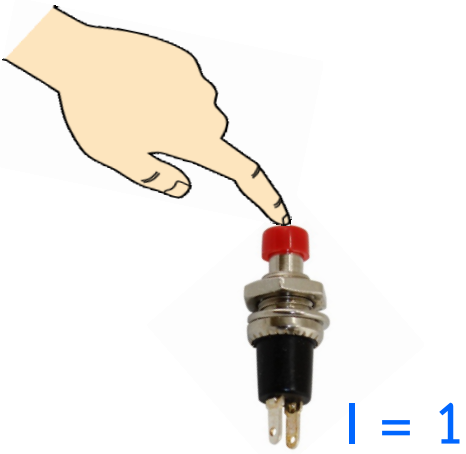
I = 0



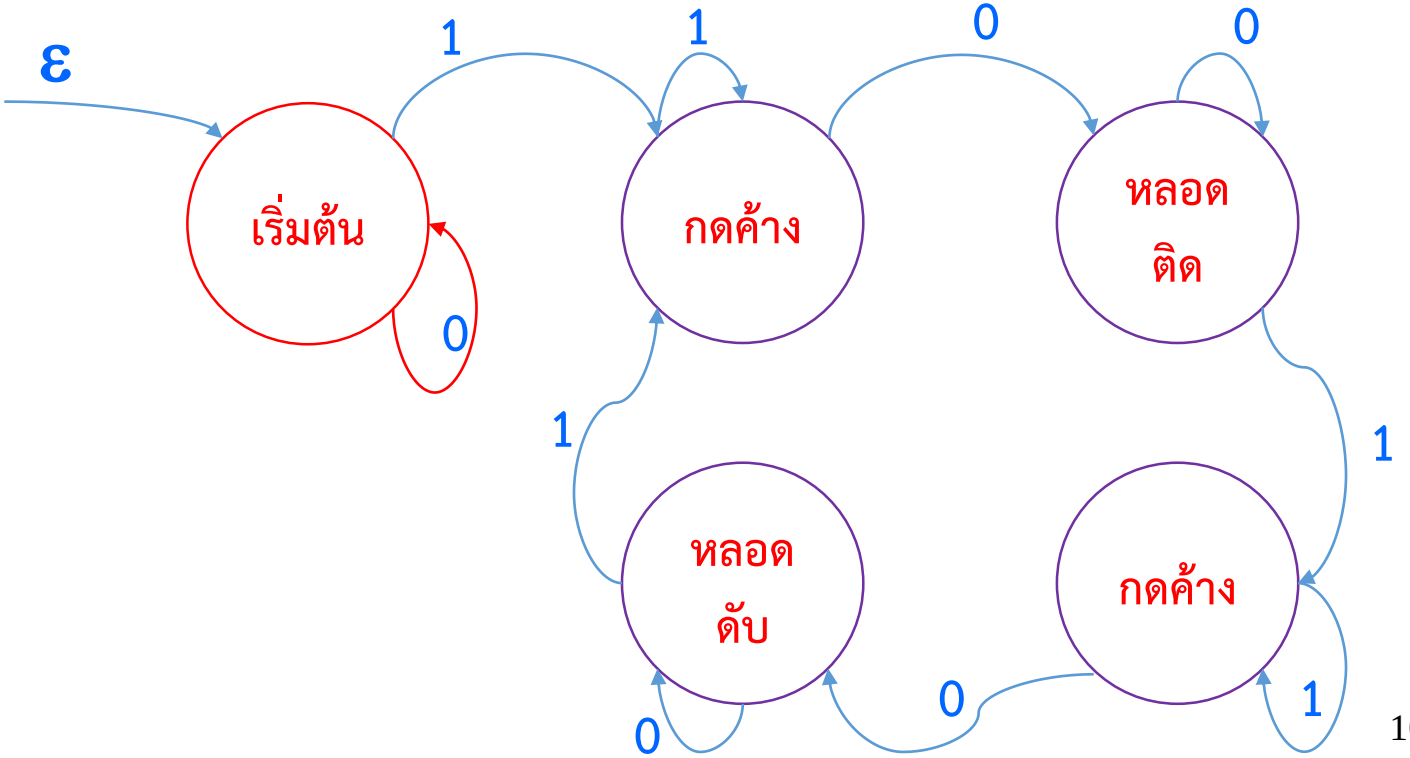
การออกแบบเครื่องจักรสถานะจำกัด จะใช้แผนผังสถานะ (State diagram) ในการออกแบบ

ตัวอย่างแผนผังสถานะ ของวงจรวงจรเปิด ปิด หลอดไฟ

I (ไม่กด=0,กด=1)



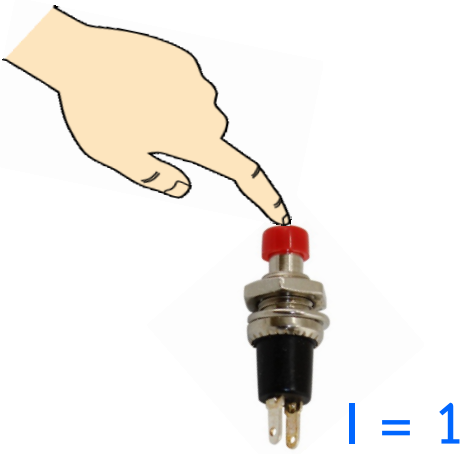
I = 1



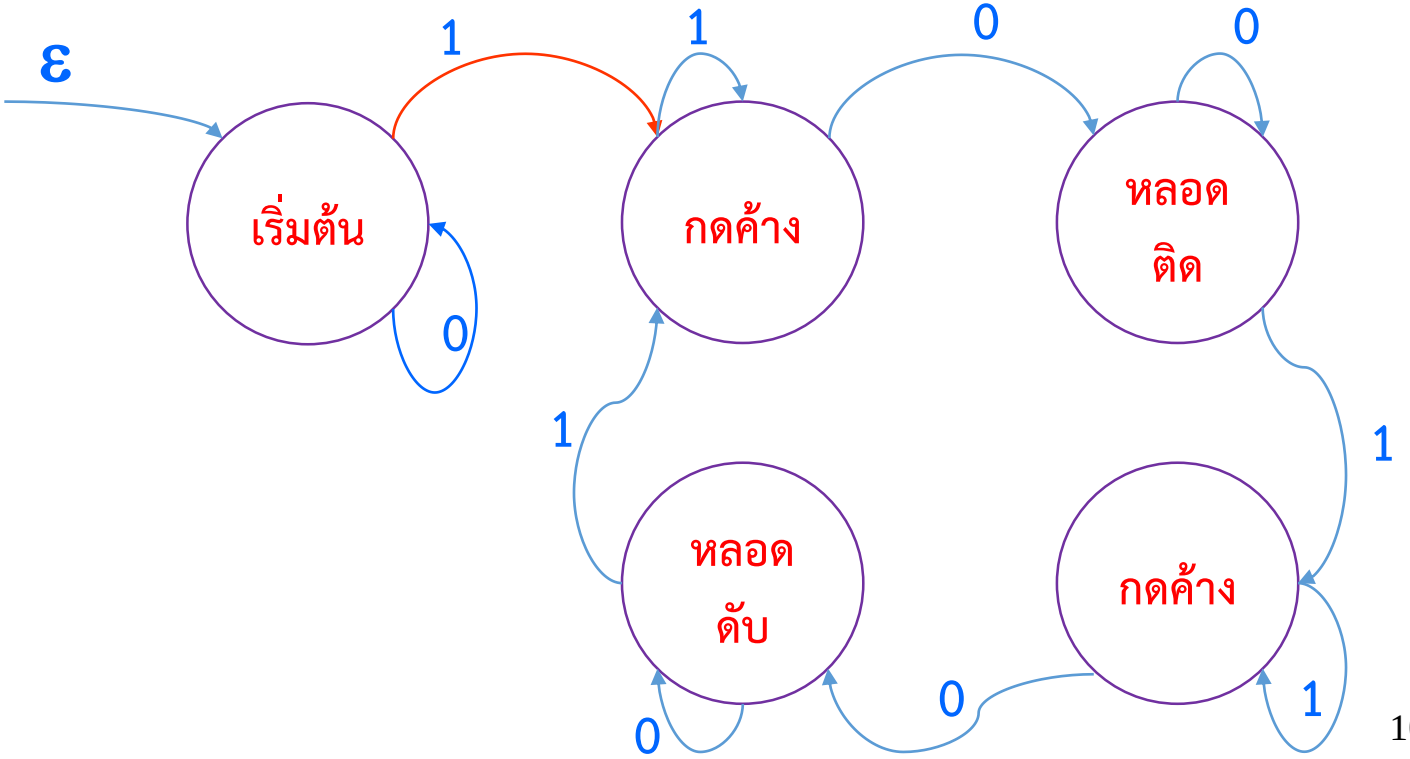
การออกแบบเครื่องจักรสถานะจำกัด จะใช้แผนผังสถานะ (State diagram) ในการออกแบบ

ตัวอย่างแผนผังสถานะ ของวงจรวงจรเปิด ปิด หลอดไฟ

I (ไม่กด=0,กด=1)



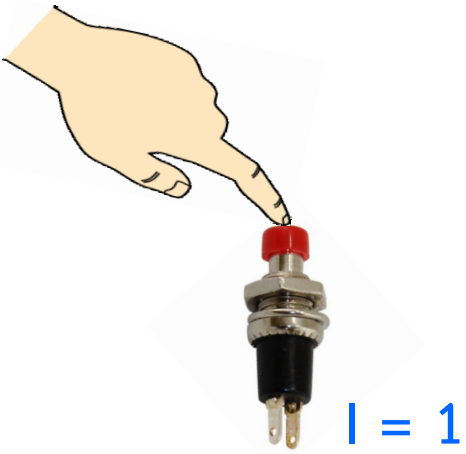
I = 1



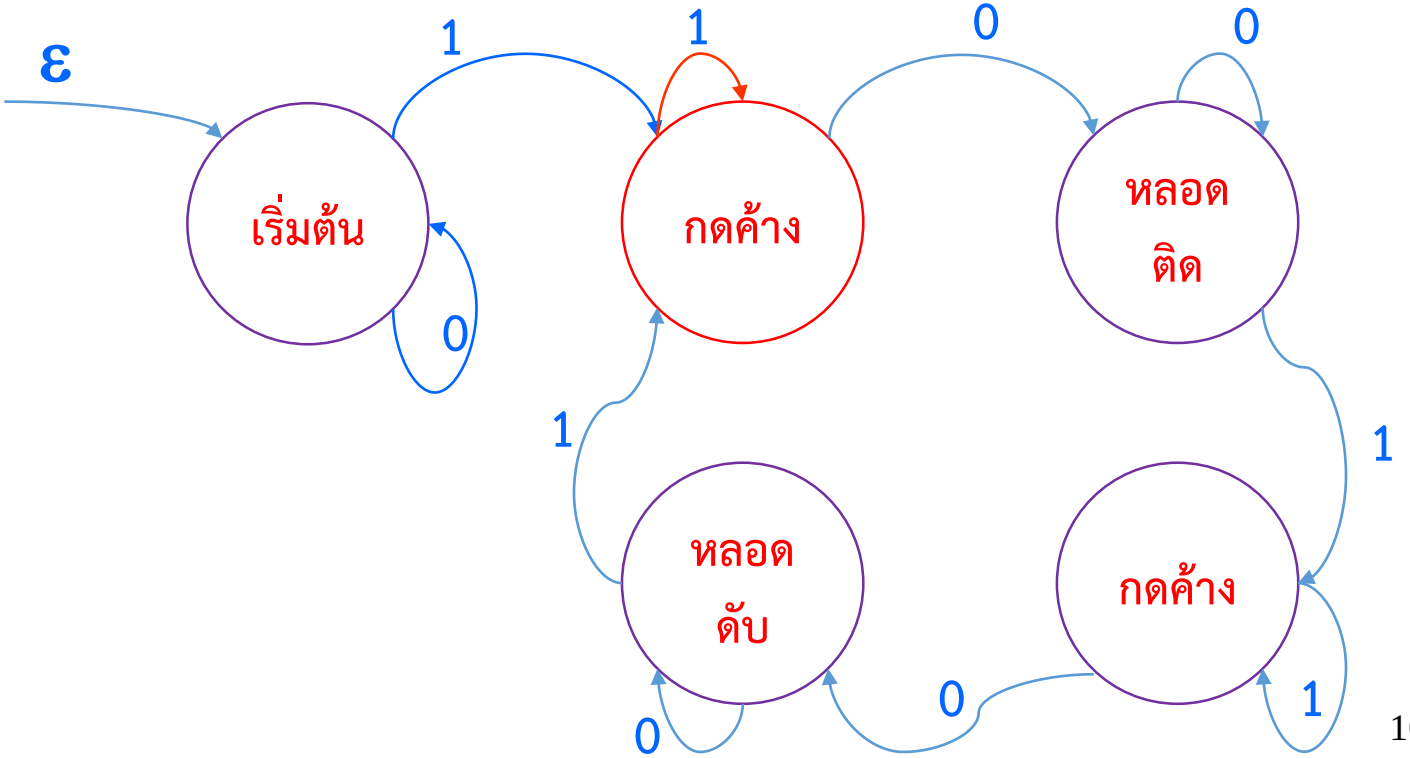
การออกแบบเครื่องจักรสถานะจำกัด จะใช้แผนผังสถานะ (State diagram) ในการออกแบบ

ตัวอย่างแผนผังสถานะ ของวงจรวงจรเปิด ปิด หลอดไฟ

I (ไม่กด=0,กด=1)



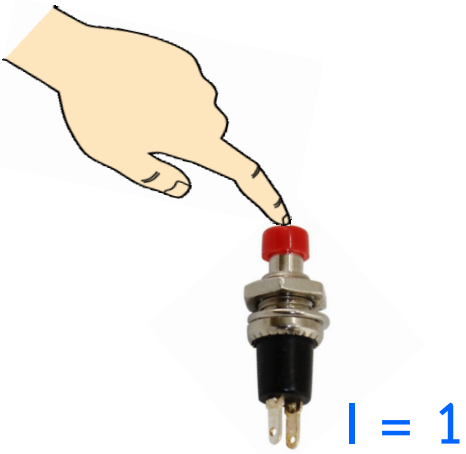
I = 1



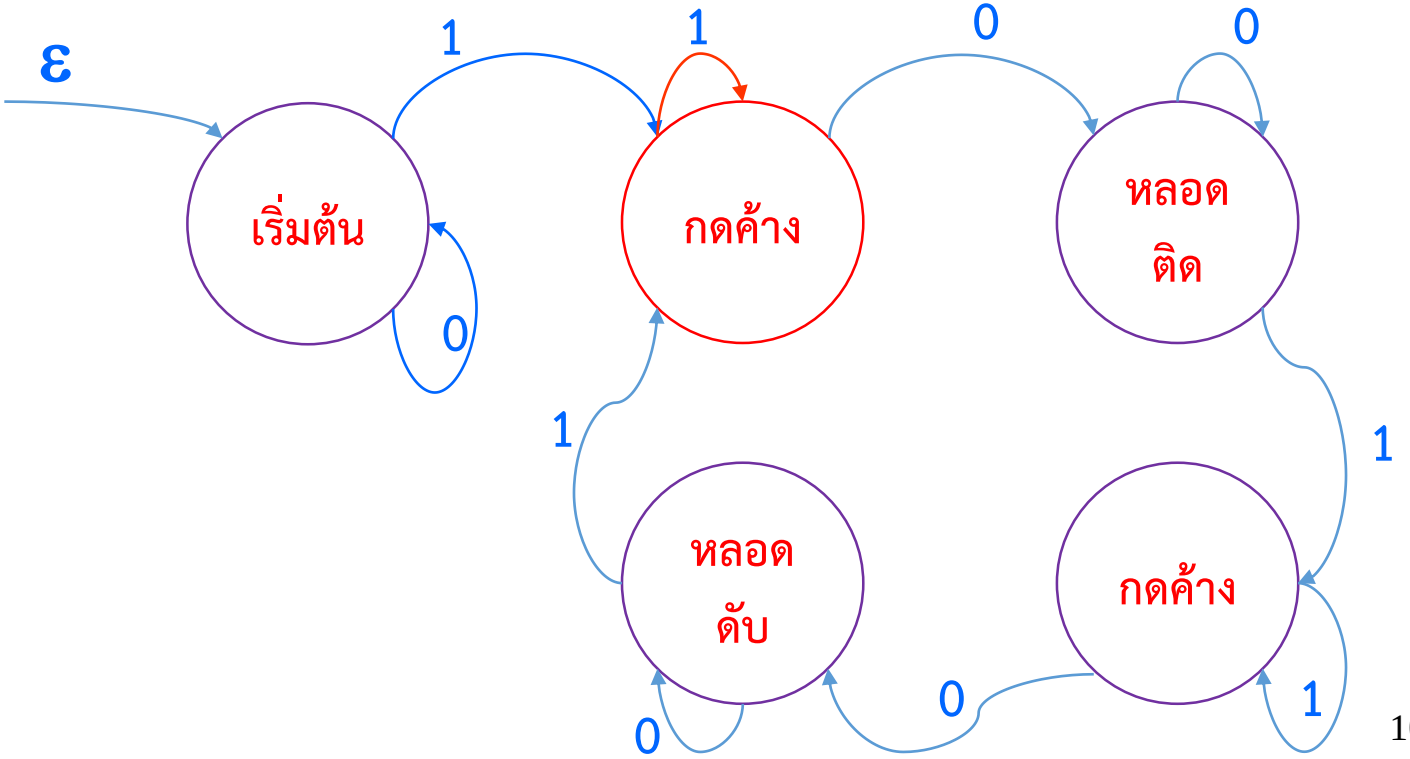
การออกแบบเครื่องจักรสถานะจำกัด จะใช้แผนผังสถานะ (State diagram) ในการออกแบบ

ตัวอย่างแผนผังสถานะ ของวงจรวงจรเปิด ปิด หลอดไฟ

I (ไม่กด=0,กด=1)

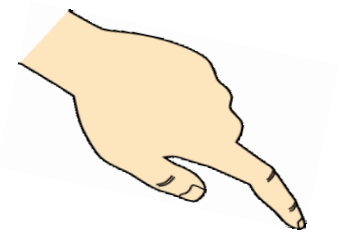


I = 1



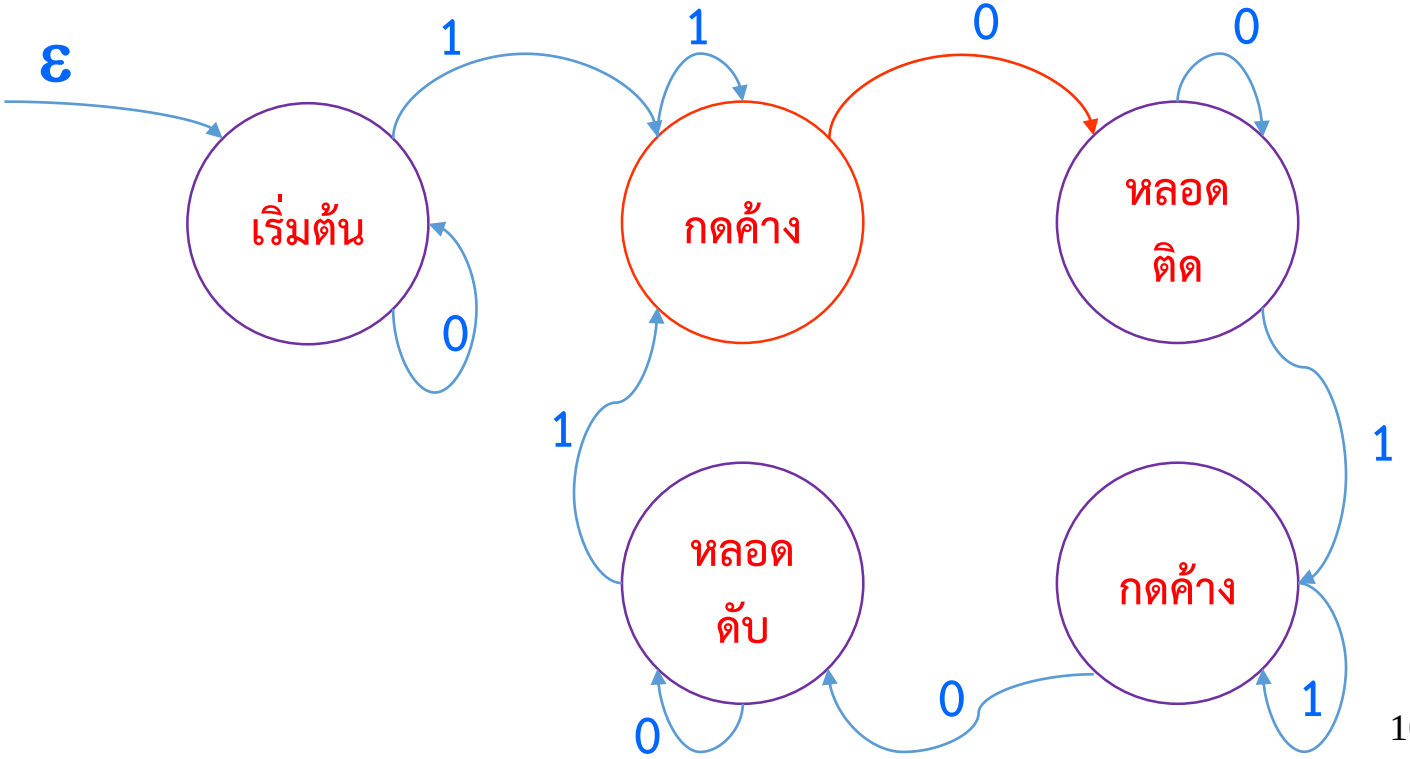
การออกแบบเครื่องจักรสถานะจำกัด จะใช้แผนผังสถานะ (State diagram) ในการออกแบบ

ตัวอย่างแผนผังสถานะ ของวงจรวงจรเปิด ปิด หลอดไฟ



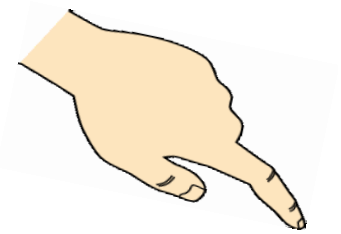
I (ไม่กด=0,กด=1)

I = 0



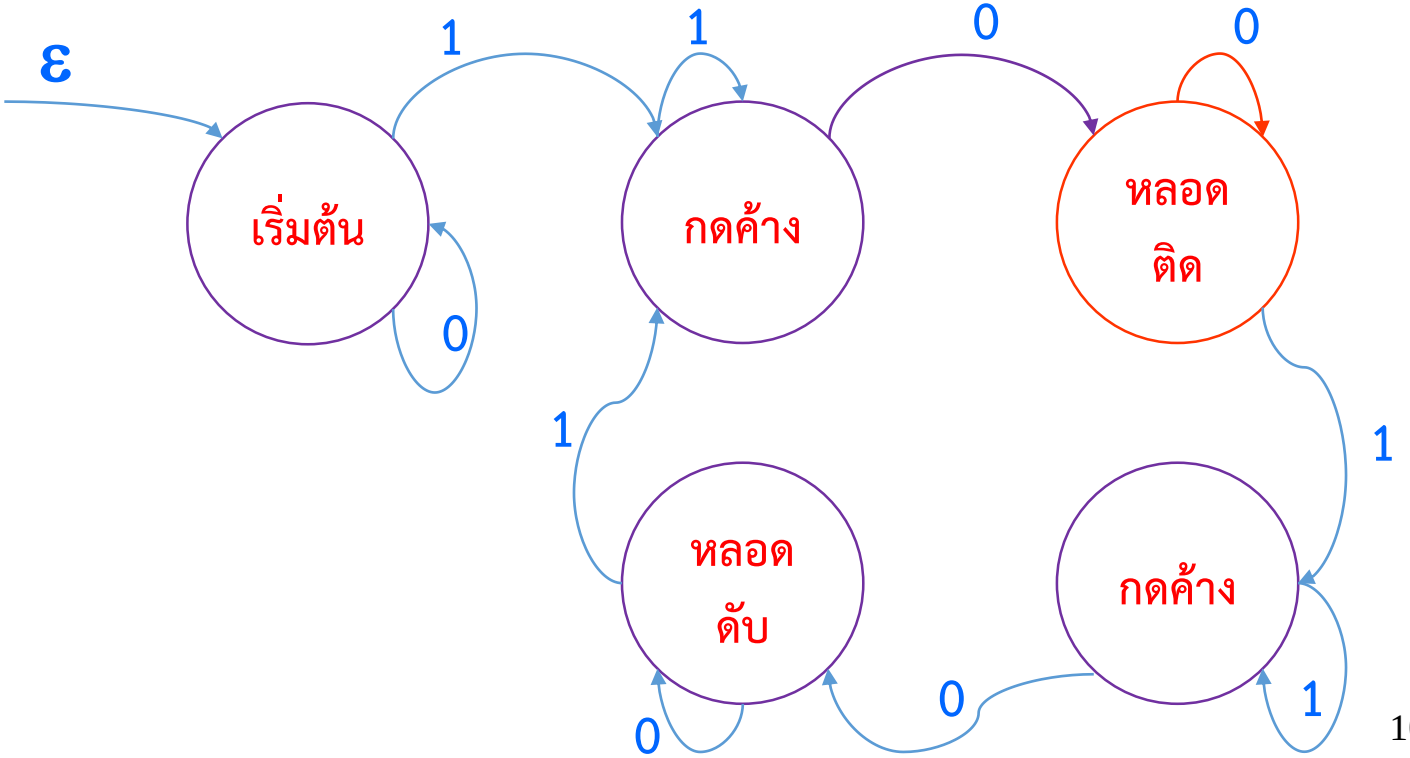
การออกแบบเครื่องจักรสถานะจำกัด จะใช้แผนผังสถานะ (State diagram) ในการออกแบบ

ตัวอย่างแผนผังสถานะ ของวงจรวงจรเปิด ปิด หลอดไฟ



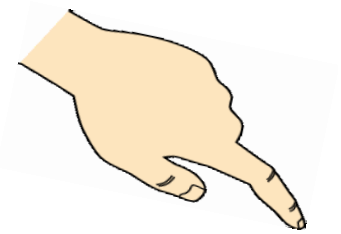
I (ไม่กด=0,กด=1)

I = 0



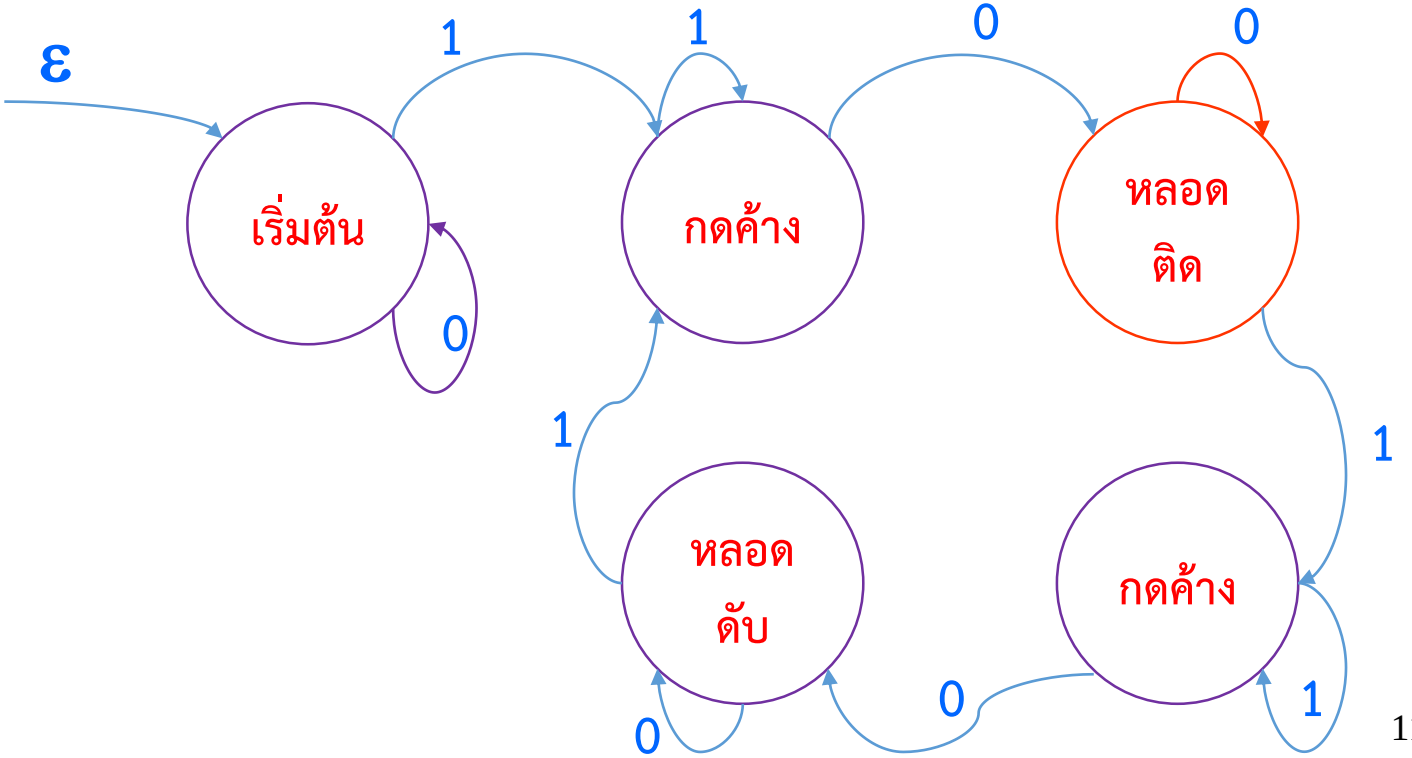
การออกแบบเครื่องจักรสถานะจำกัด จะใช้แผนผังสถานะ (State diagram) ในการออกแบบ

ตัวอย่างแผนผังสถานะ ของวงจรวงจรเปิด ปิด หลอดไฟ



I (ไม่กด=0,กด=1)

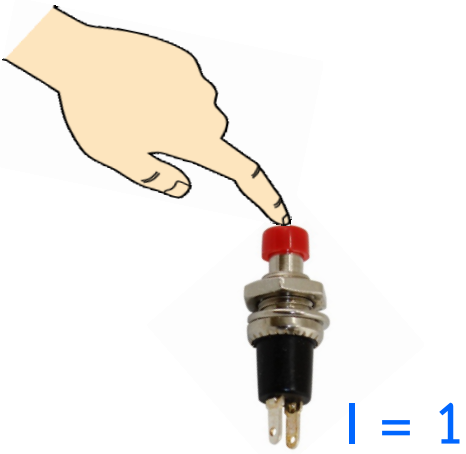
I = 0



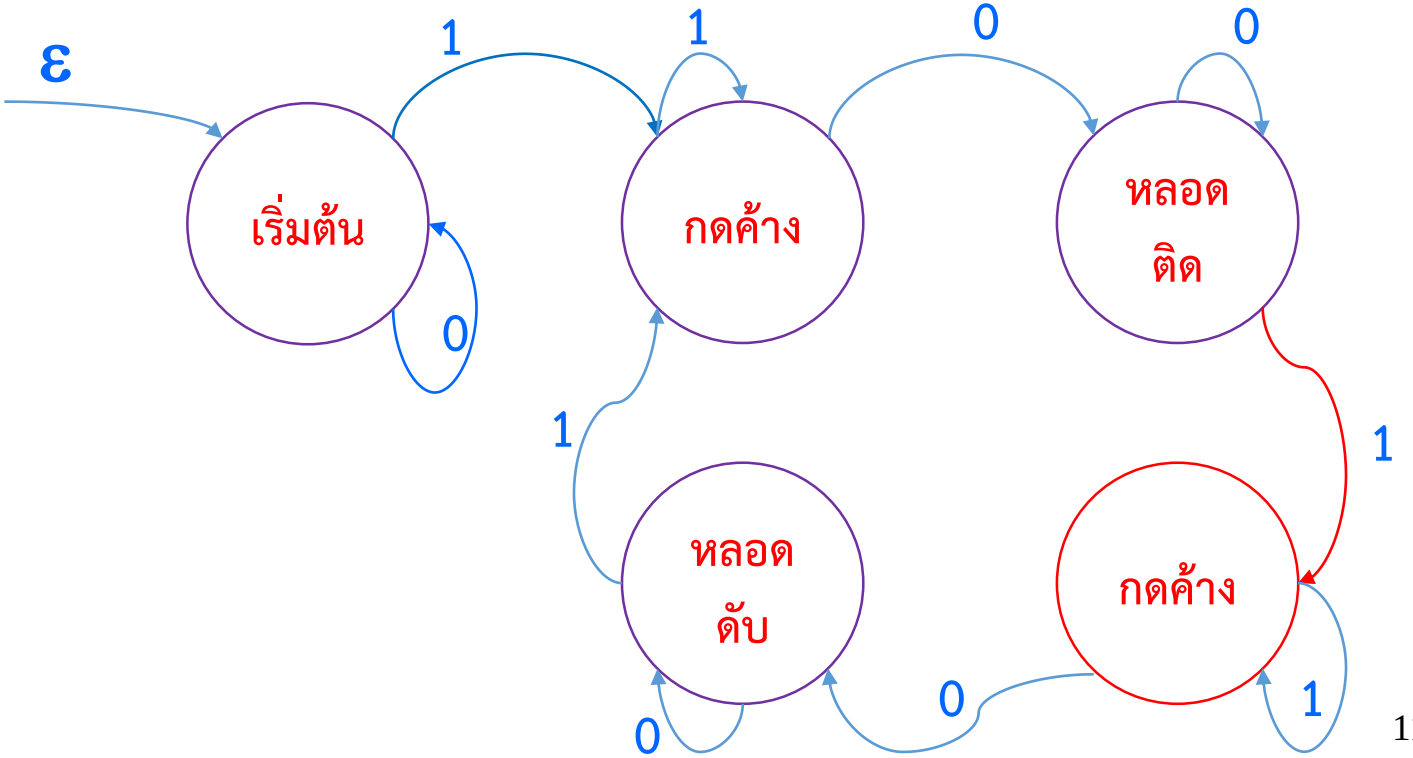
การออกแบบเครื่องจักรสถานะจำกัด จะใช้แผนผังสถานะ (State diagram) ในการออกแบบ

ตัวอย่างแผนผังสถานะ ของวงจรวงจรเปิด ปิด หลอดไฟ

I (ไม่กด=0,กด=1)



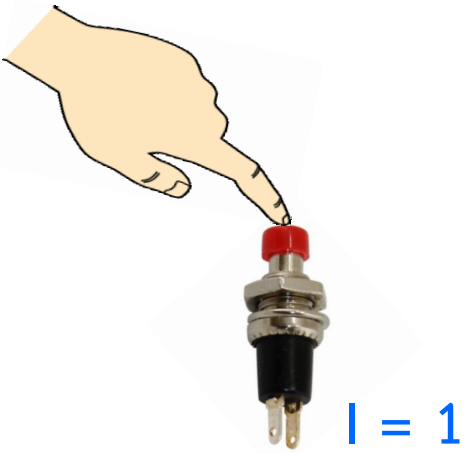
I = 1



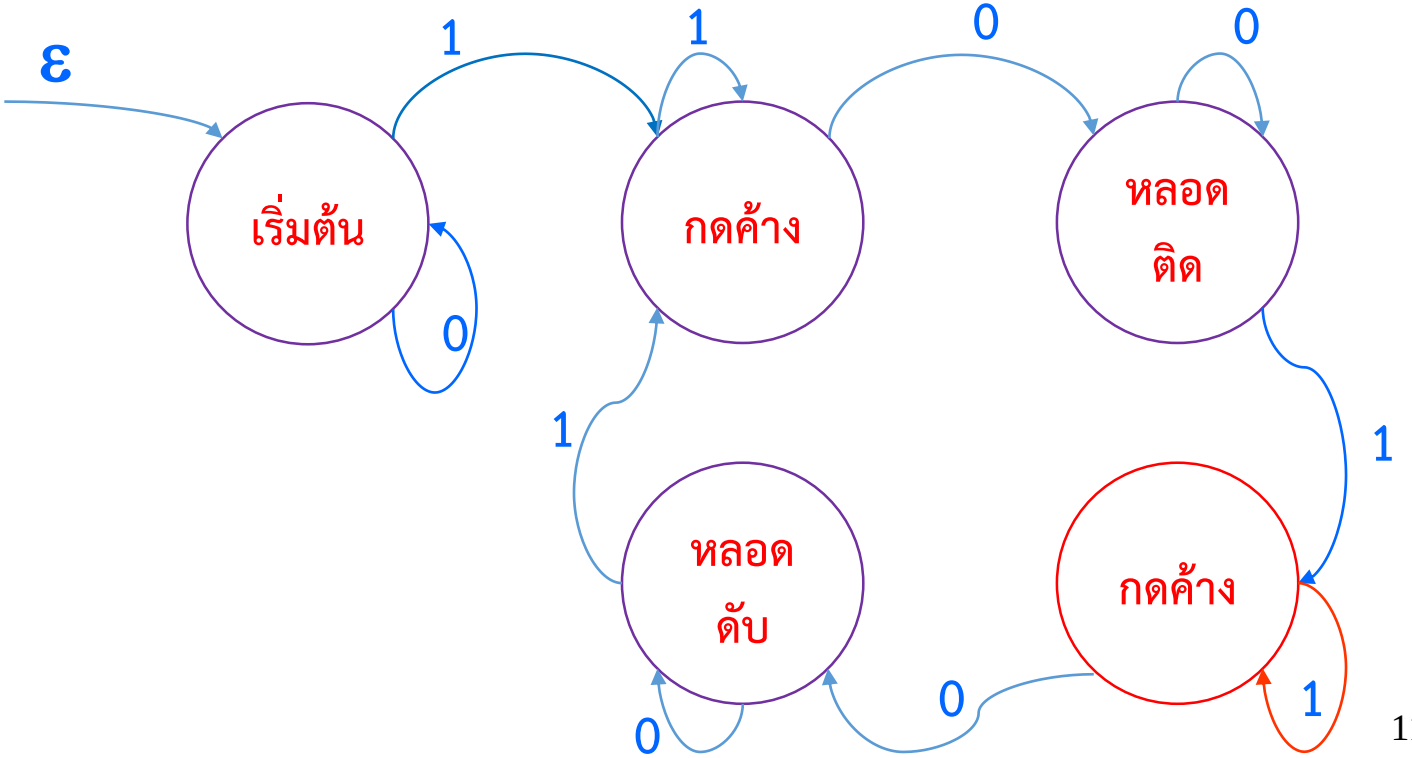
การออกแบบเครื่องจักรสถานะจำกัด จะใช้แผนผังสถานะ (State diagram) ในการออกแบบ

ตัวอย่างแผนผังสถานะ ของวงจรวงจรเปิด ปิด หลอดไฟ

I (ไม่กด=0,กด=1)



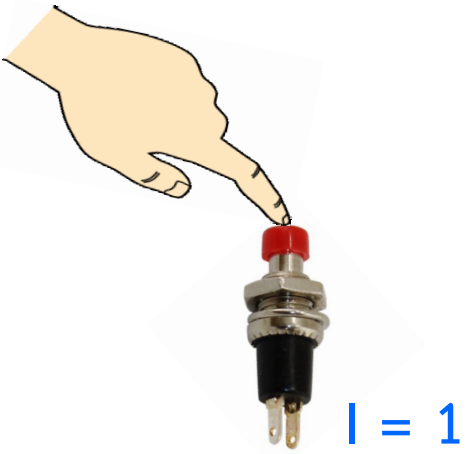
I = 1



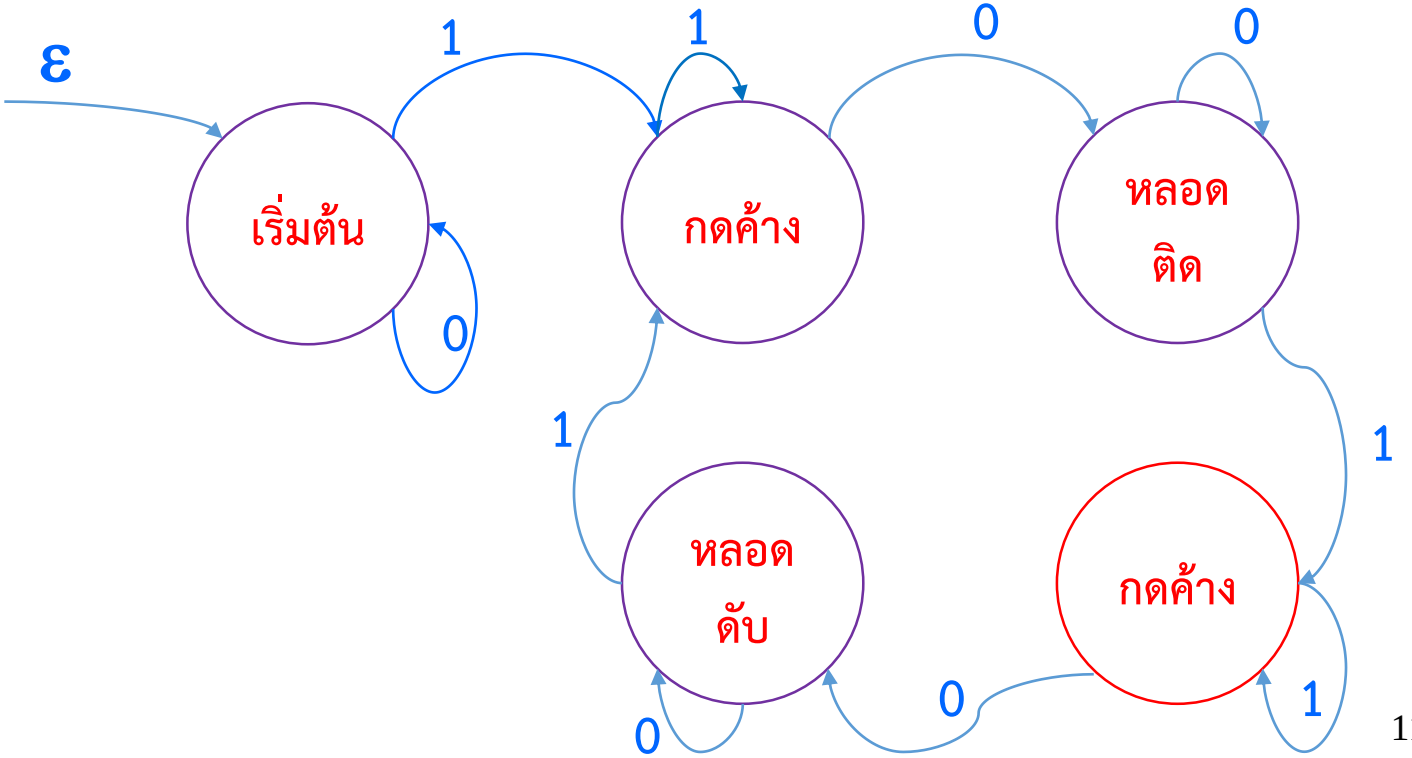
การออกแบบเครื่องจักรสถานะจำกัด จะใช้แผนผังสถานะ (State diagram) ในการออกแบบ

ตัวอย่างแผนผังสถานะ ของวงจรวงจรเปิด ปิด หลอดไฟ

I (ไม่กด=0,กด=1)

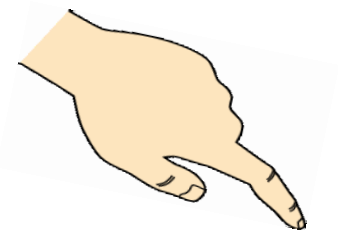


I = 1



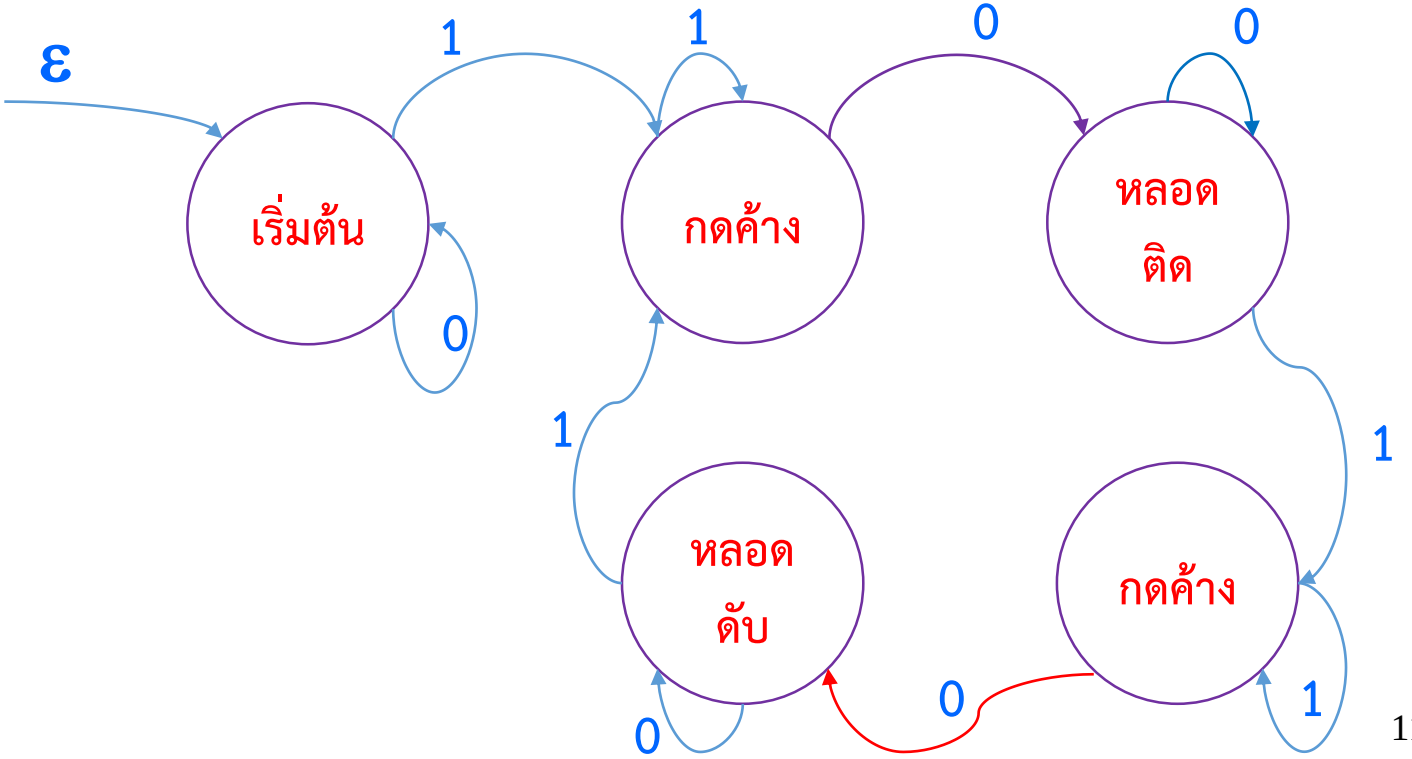
การออกแบบเครื่องจักรสถานะจำกัด จะใช้แผนผังสถานะ (State diagram) ในการออกแบบ

ตัวอย่างแผนผังสถานะ ของวงจรวงจรเปิด ปิด หลอดไฟ



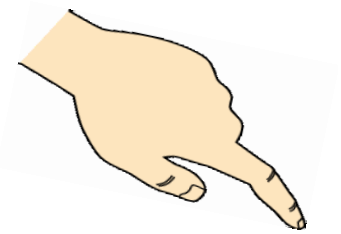
I (ไม่กด=0,กด=1)

I = 0



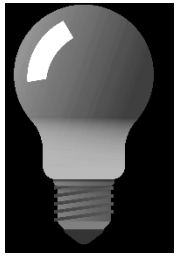
การออกแบบเครื่องจักรสถานะจำกัด จะใช้แผนผังสถานะ (State diagram) ในการออกแบบ

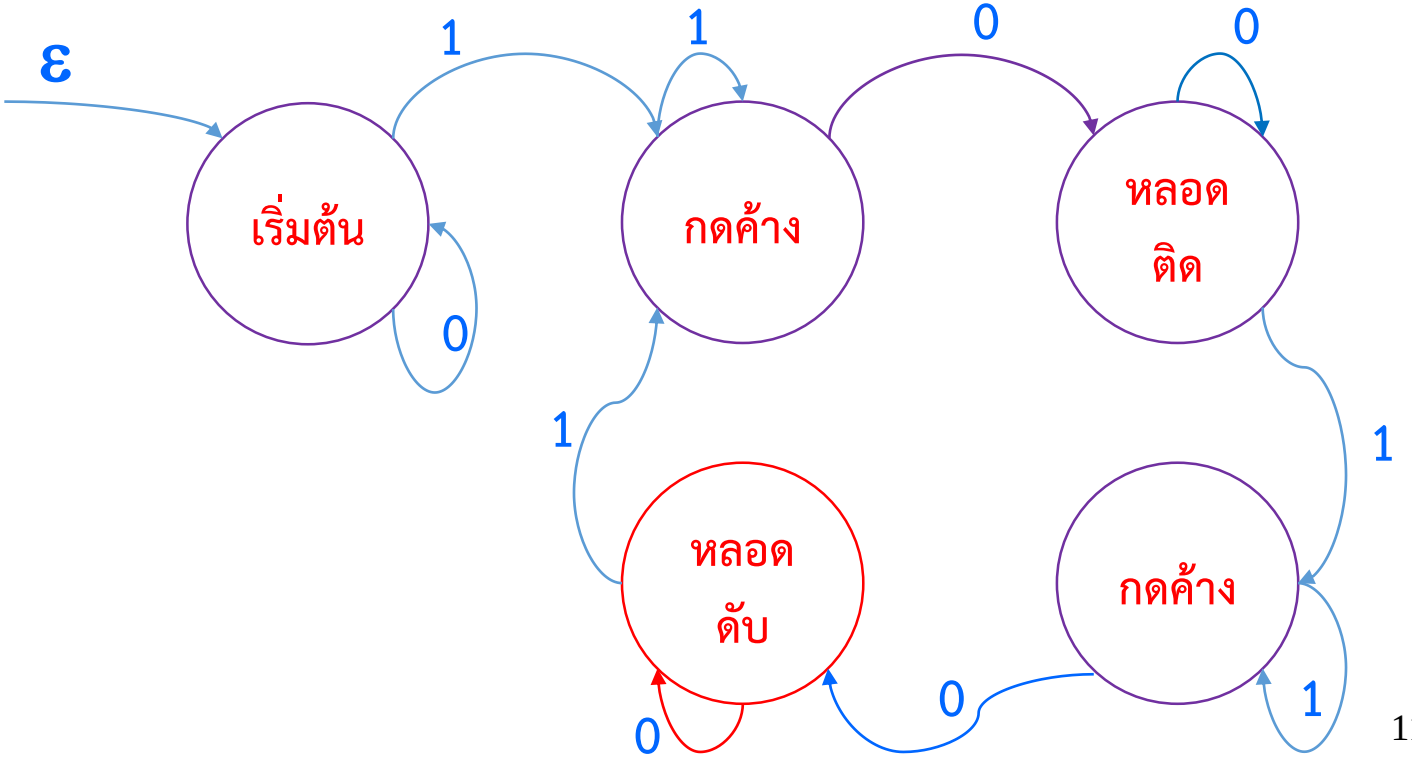
ตัวอย่างแผนผังสถานะ ของวงจรวงจรเปิด ปิด หลอดไฟ



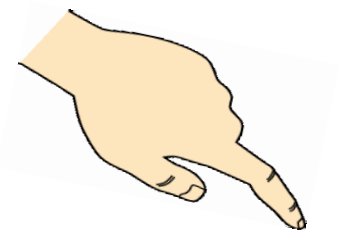
I (ไม่กด=0,กด=1)

I = 0





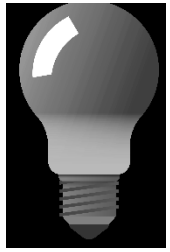
การออกแบบเครื่องจักรสถานะจำกัด จะใช้แผนผังสถานะ (State diagram) ในการออกแบบ

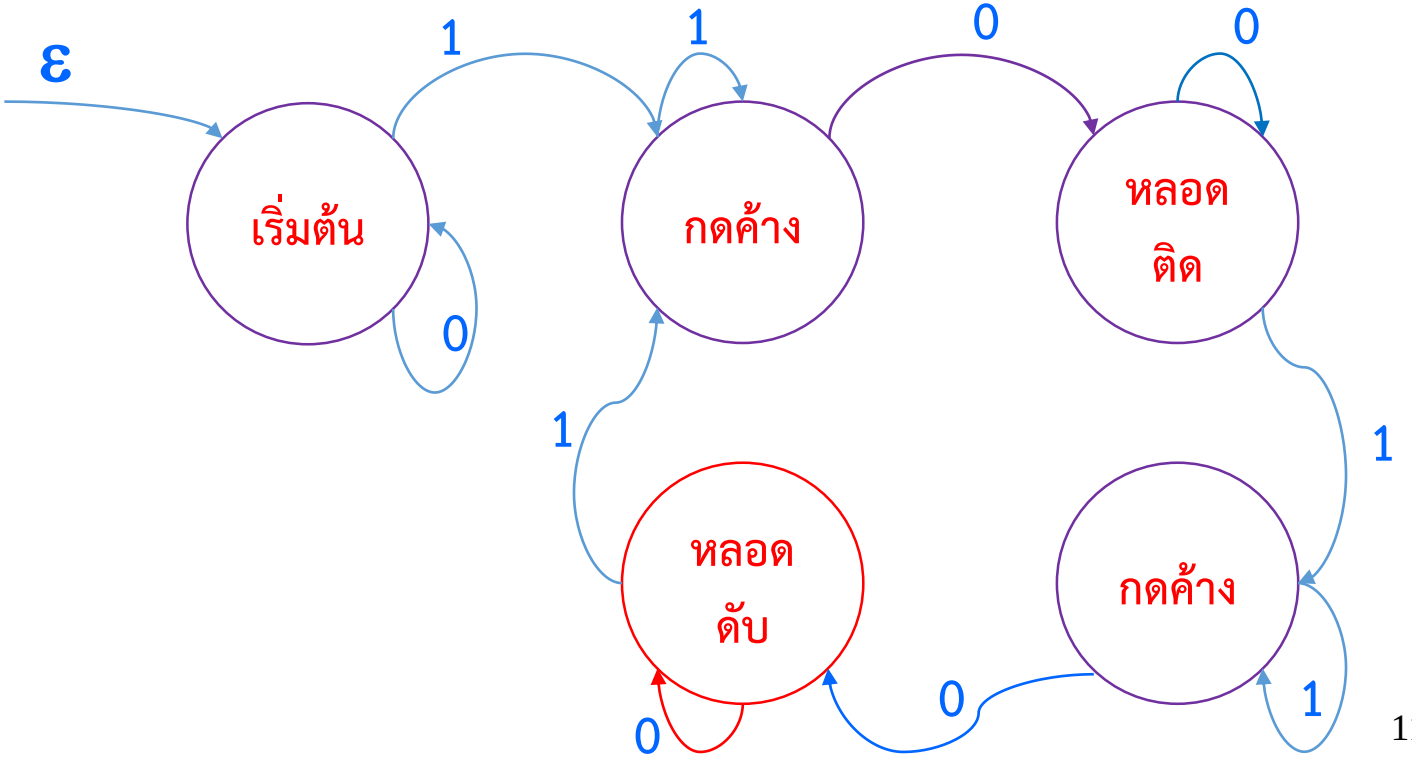


ตัวอย่างแผนผังสถานะ ของวงจรวงจรเปิด ปิด หลอดไฟ

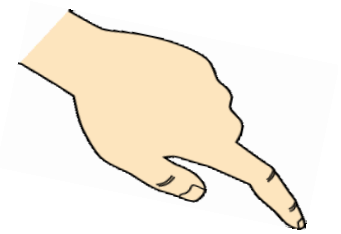
I (ไม่กด=0,กด=1)

I = 0





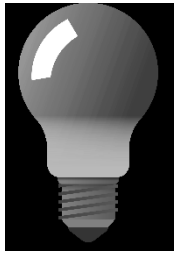
การออกแบบเครื่องจักรสถานะจำกัด จะใช้แผนผังสถานะ (State diagram) ในการออกแบบ

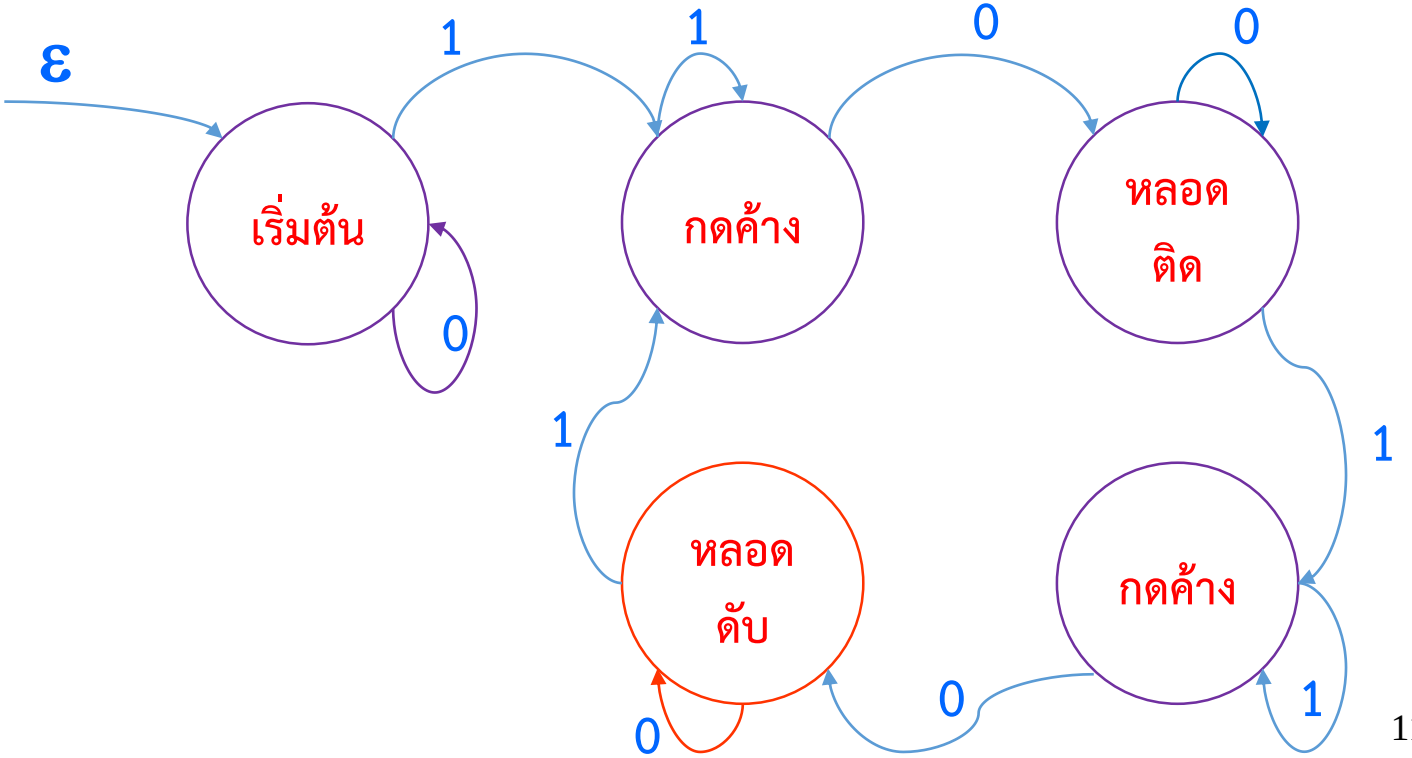


ตัวอย่างแผนผังสถานะ ของวงจรวงจรเปิด ปิด หลอดไฟ

I (ไม่กด=0,กด=1)

I = 0

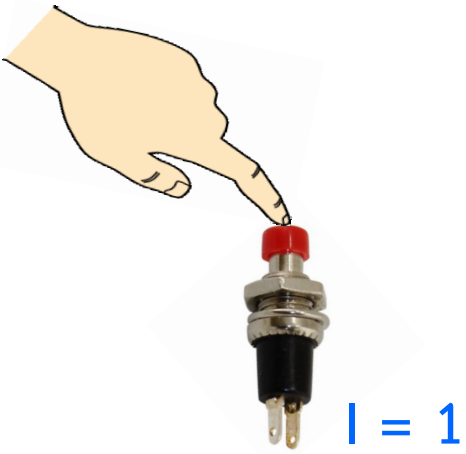




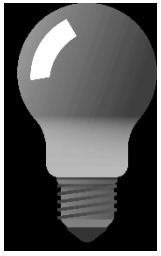
การออกแบบเครื่องจักรสถานะจำกัด จะใช้แผนผังสถานะ (State diagram) ในการออกแบบ

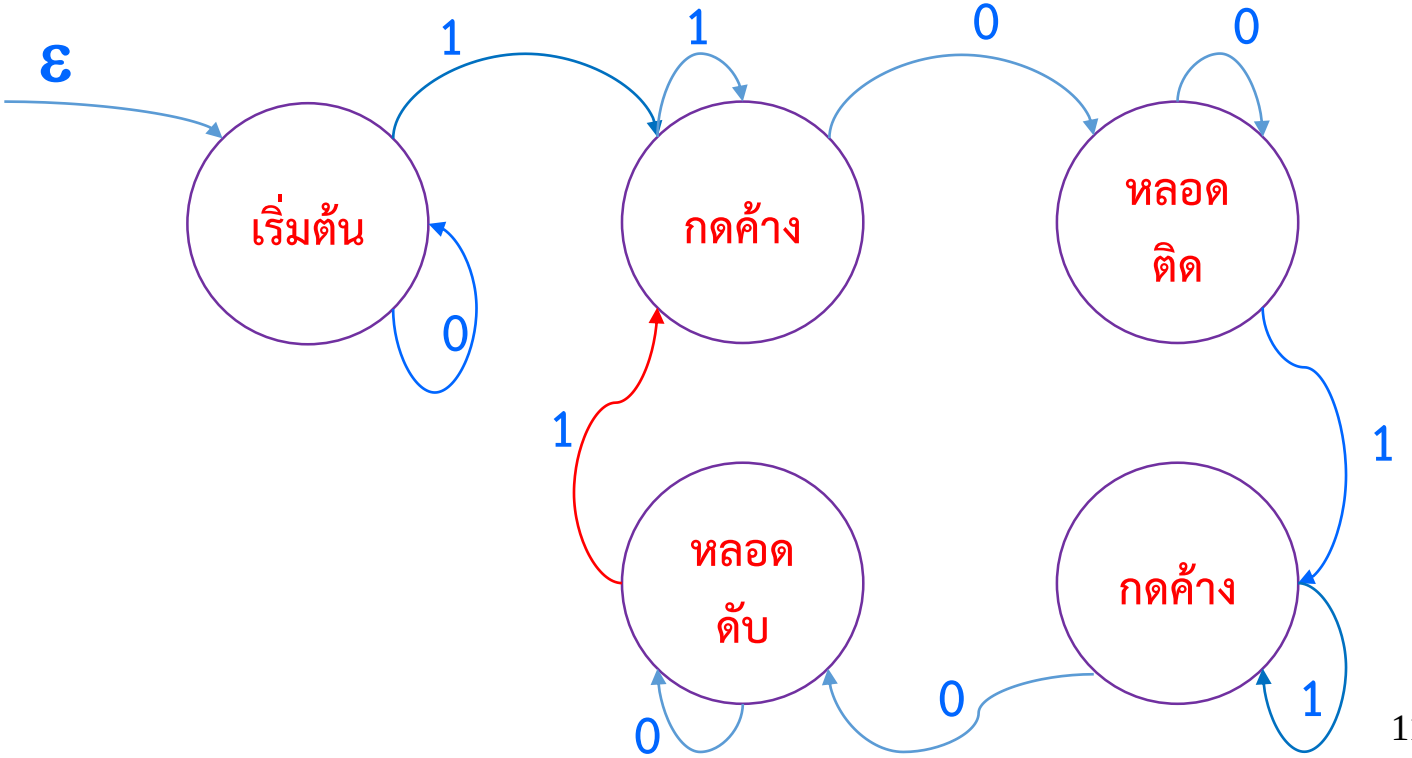
ตัวอย่างแผนผังสถานะ ของวงจรวงจรเปิด ปิด หลอดไฟ

I (ไม่กด=0,กด=1)



I = 1

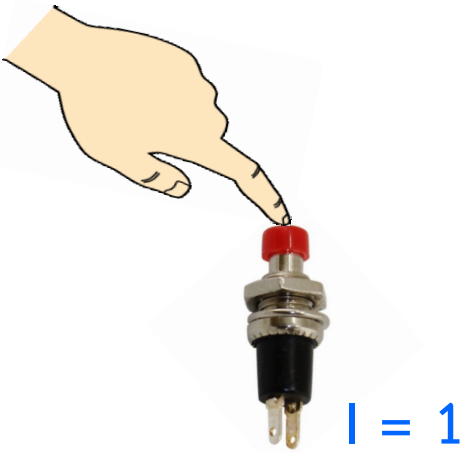




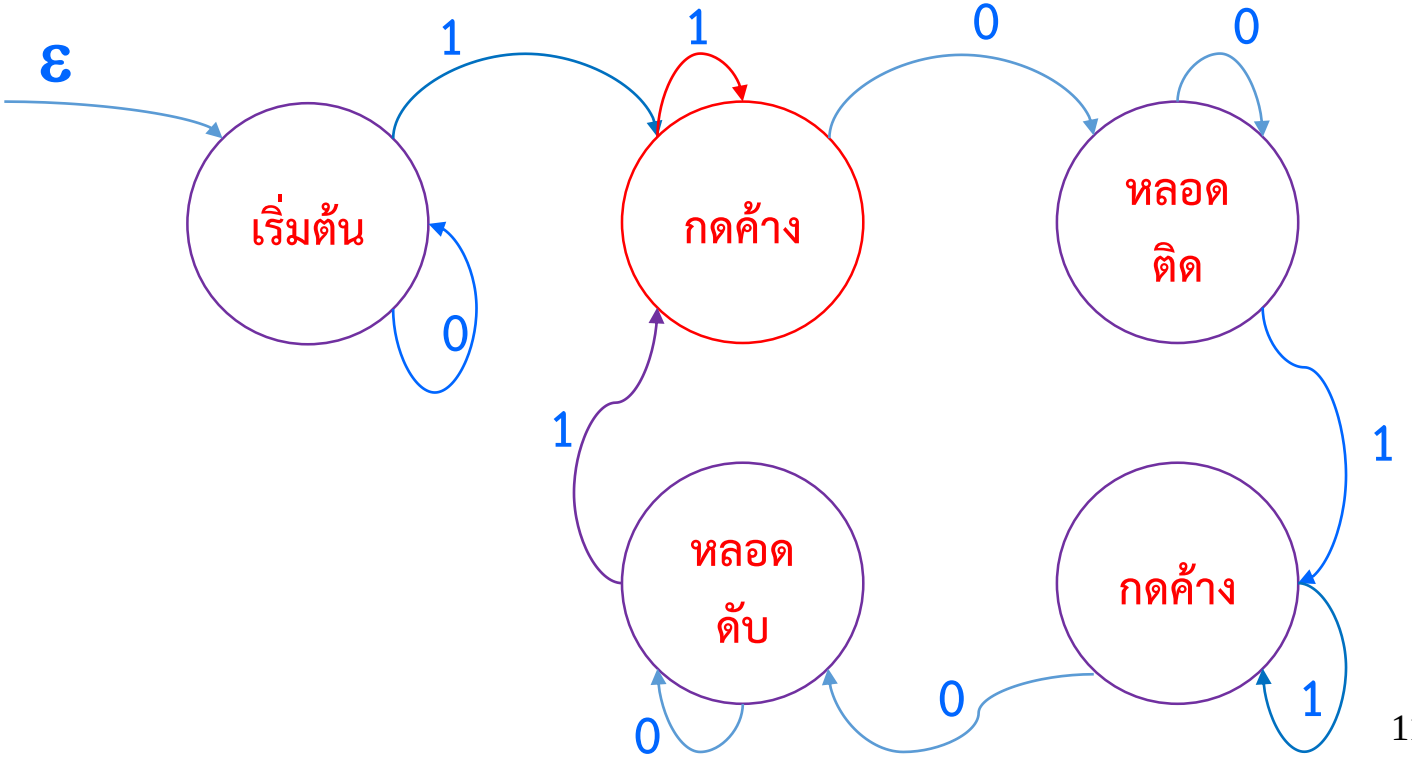
การออกแบบเครื่องจักรสถานะจำกัด จะใช้แผนผังสถานะ (State diagram) ในการออกแบบ

ตัวอย่างแผนผังสถานะ ของวงจรวงจรเปิด ปิด หลอดไฟ

I (ไม่กด=0,กด=1)



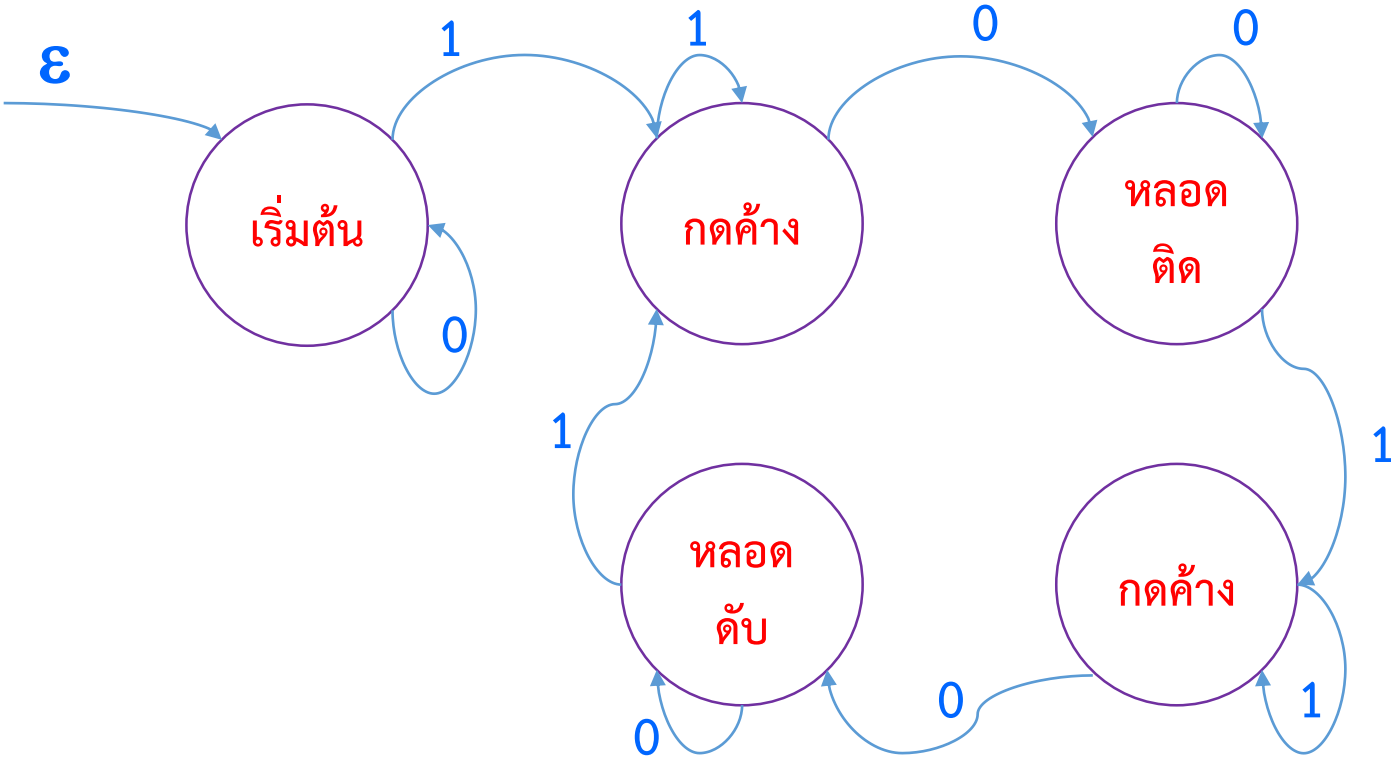
I = 1



การนำเอาแผนผังสถานะไปสร้างเป็นวงจร โดยใช้รอม และ แรม

ขั้นตอนที่ 1) กำหนดหมายเลขให้สถานะต่างๆ

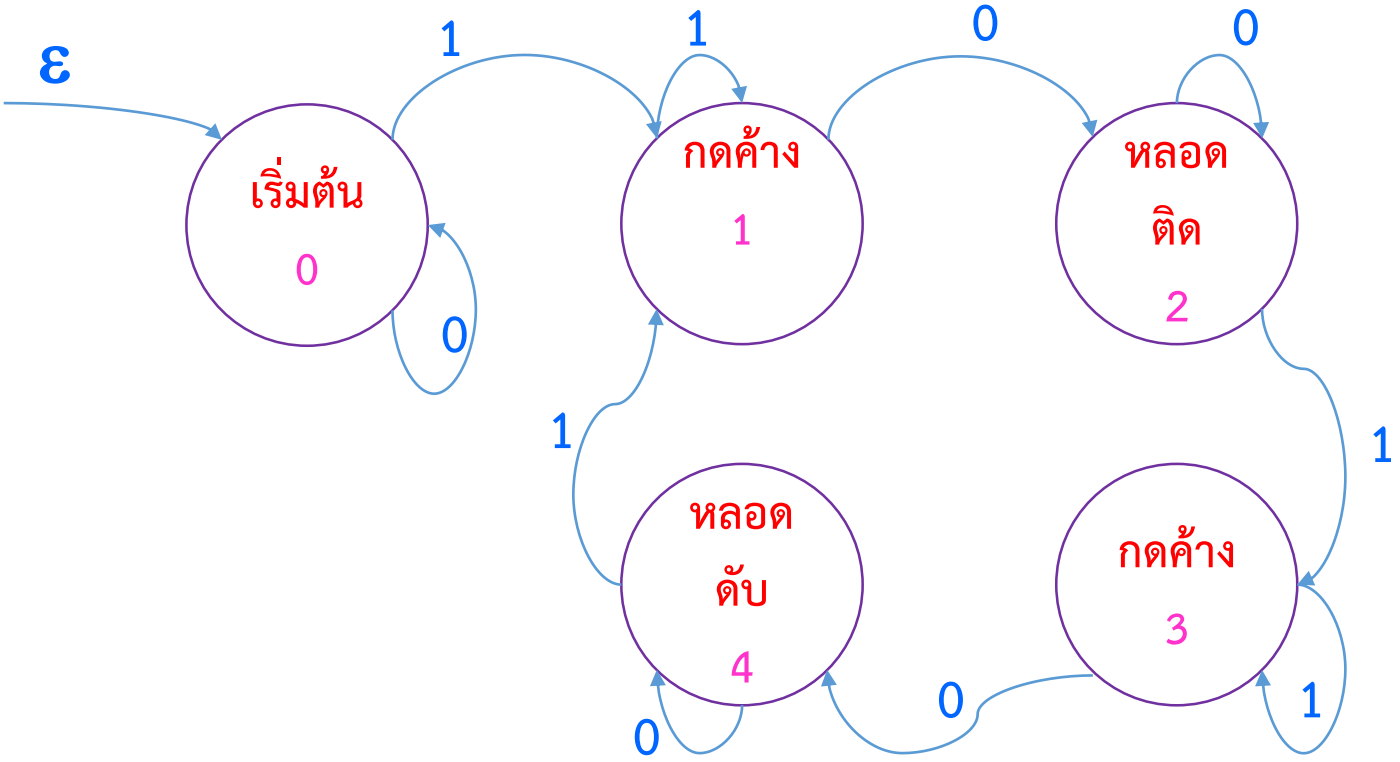
I (ไม่กด=0,กด=1)



การนำเอาแผนผังสถานะไปสร้างเป็นวงจร โดยใช้รอม และ แรม

ขั้นตอนที่ 1) กำหนดหมายเลขให้สถานะต่างๆ

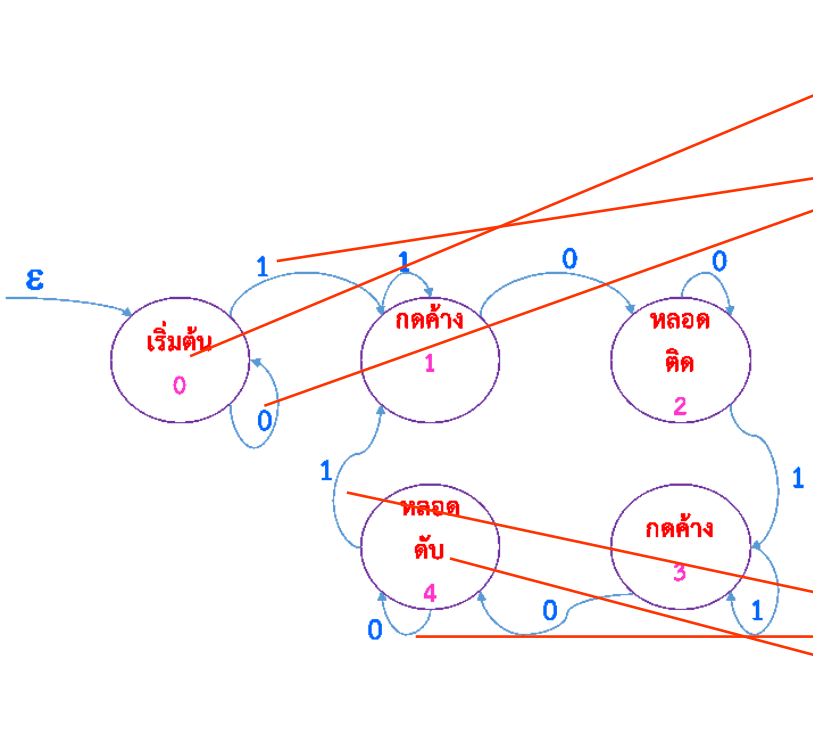
I (ไม่กด=0,กด=1)



การนำเอาแผนผังสถานะไปสร้างเป็นวงจร โดยใช้รอม และ แรม

ขั้นตอนที่ 1) กำหนดหมายเลขให้สถานะต่างๆ

ขั้นตอนที่ 2) สร้างตารางค่าความจริง

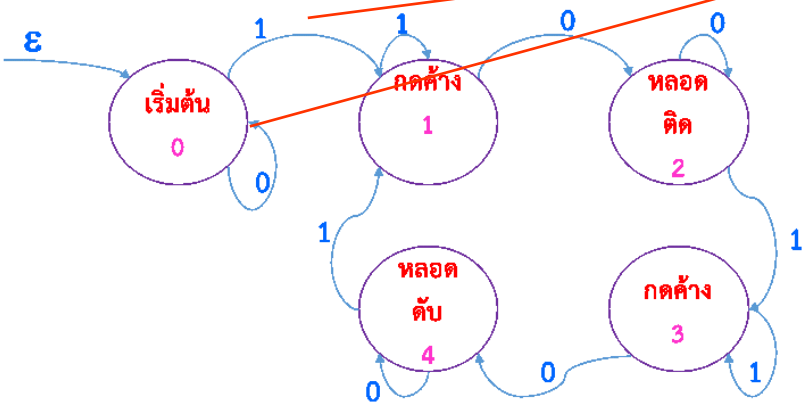


Input		Output	
สถานะปัจจุบัน	I	สถานะถัดไป	Y
0	0		
0	1		
1	0		
1	1		
2	0		
2	1		
3	0		
3	1		
4	0		
4	1		

การนำเอาแผนผังสถานะไปสร้างเป็นวงจร โดยใช้รอม และ แรม

ขั้นตอนที่ 1) กำหนดหมายเลขให้สถานะต่างๆ

ขั้นตอนที่ 2) สร้างตารางค่าความจริง



Input		Output	
สถานะปัจจุบัน	I	สถานะถัดไป	Y
0	0	0	
0	1	1	
1	0	2	
1	1	1	
2	0	2	
2	1	3	
3	0	4	
3	1	3	
4	0	4	
4	1	1	

การนำเอาแผนผังสถานะไปสร้างเป็นวงจร โดยใช้รอม และ แรม

ขั้นตอนที่ 1) กำหนดหมายเลขให้สถานะต่างๆ

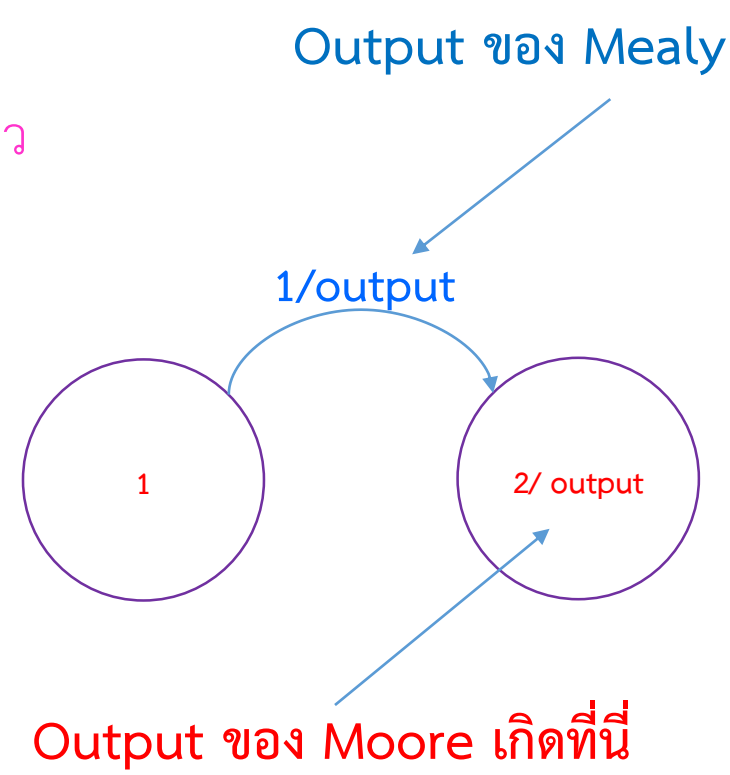
ขั้นตอนที่ 2) สร้างตารางค่าความจริง

ขั้นตอนที่ 3) กำหนด Output

การกำหนด Output เรียกอีกชื่อว่าการถอดรหัส (Decode output) สามารถทำได้ 2 วิธีคือ Mealy และ Moore จุดที่แตกต่างกันคือ

Mealy จะถอดรหัสจากสถานะปัจจุบันและ Input

แต่ Moore จะถอดรหัสจากสถานะปัจจุบันอย่างเดียว



การนำเอาแผนผังสถานะไปสร้างเป็นวงจร โดยใช้รอม และ แรม

การกำหนด Output เรียกอีกชื่อว่าการถอดรหัส (Decode output) สามารถทำได้ 2 วิธีคือ Mealy และ Moore จุดที่แตกต่างกันคือ

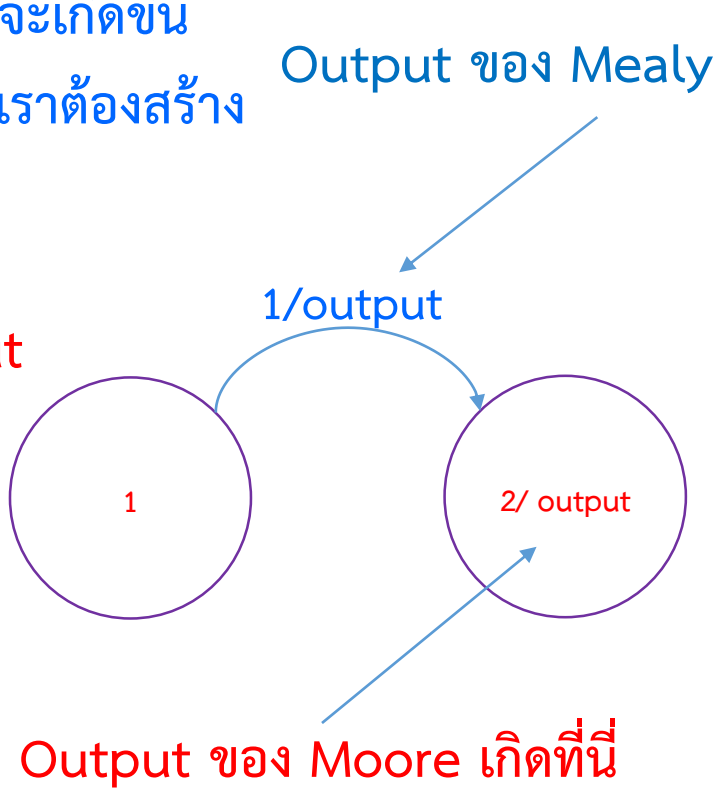
Mealy จะถอดรหัสจากสถานะปัจจุบันและ Input

แต่ Moore จะถอดรหัสจากสถานะปัจจุบันอย่างเดียว

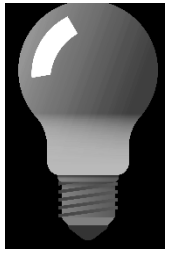
Mealy จะให้ output ก่อน Moore แต่ Output จะเกิดขึ้นในช่วงเวลาอันสั้น ถ้า Input ไม่มีวงจร Latch ไว้ เราต้องสร้างวงจรเพื่อ Latch ตัว Output ของวงจรเอง

Moore จะให้ output ช้ากว่า Moore แต่ Output จะค้างอยู่จนกว่าจะมีการเปลี่ยนแปลงสถานะ

วิชานี้เราจะใช้การถอดรหัสแบบ Moore !



การนำเอาแผนผังสถานะไปสร้างเป็นวงจร โดยใช้รอม และ แรม



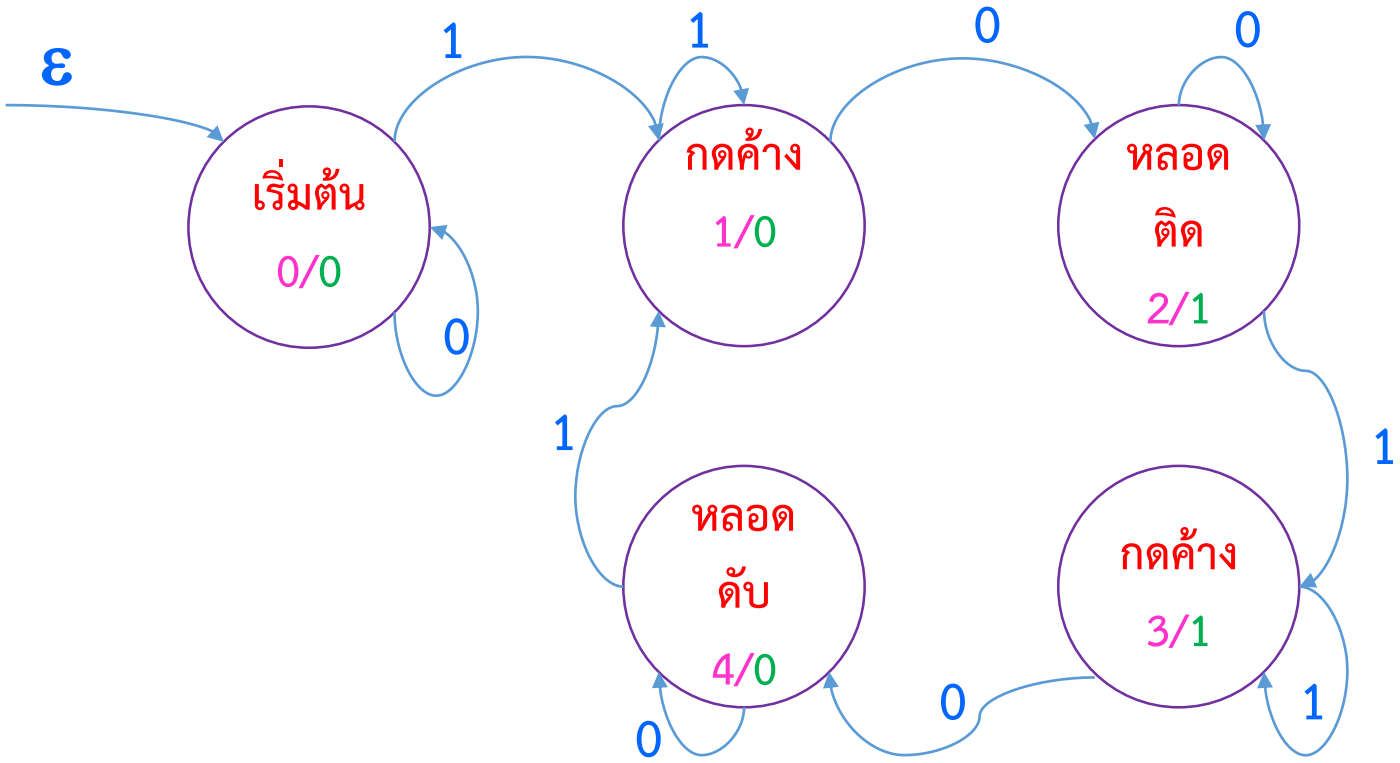
0 = ดับ

1 = ติด

ขั้นตอนที่ 1) กำหนดหมายเลขให้สถานะต่างๆ

ขั้นตอนที่ 2) สร้างตารางค่าความจริง

ขั้นตอนที่ 3) กำหนด Output (Moore)

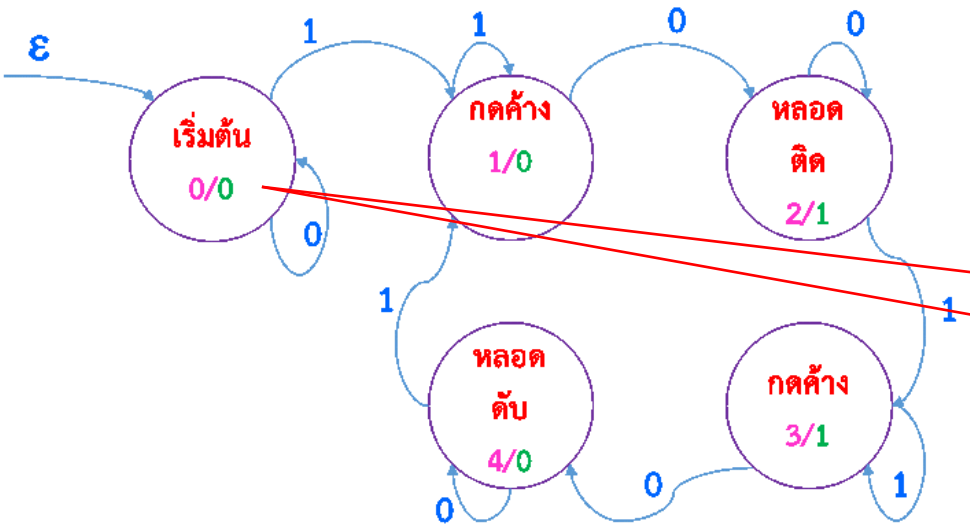


การนำเอาแผนผังสถานะไปสร้างเป็นวงจร โดยใช้รอม และ แรม

ขั้นตอนที่ 1) กำหนดหมายเลขให้สถานะต่างๆ

ขั้นตอนที่ 2) สร้างตารางค่าความจริง

ขั้นตอนที่ 3) กำหนด Output (Moore)



Input		Output	
สถานะปัจจุบัน	I	สถานะถัดไป	Y
0	0	0	0
0	1	1	0
1	0	2	1
1	1	1	0
2	0	2	1
2	1	3	1
3	0	4	0
3	1	3	1
4	0	4	0
4	1	1	0

Output ของสถานะเป็น 0 ก็เติม 0 ใน Y

ถ้าเป็น 1 ก็เติม 1

มันจะซ้ำไปซ้ำมา

การนำเอาแผนผังสถานะไปสร้างเป็นวงจร โดยใช้รอม และ แรม

ขั้นตอนที่ 1) กำหนดหมายเลขให้สถานะต่างๆ

ขั้นตอนที่ 2) สร้างตารางค่าความจริง

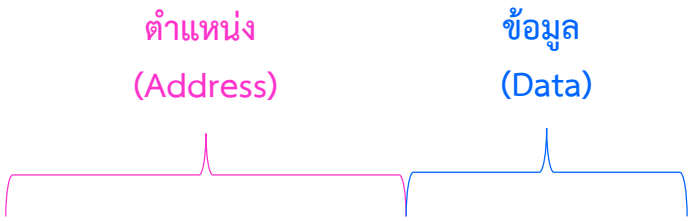
ขั้นตอนที่ 3) กำหนด Output (Moore)

ขั้นตอนที่ 4) เปลี่ยนสถานะเป็นเลขฐานสอง

Input		Output	
สถานะปัจจุบัน	I	สถานะถัดไป	Y
0	0	0	0
0	1	1	0
1	0	2	0
1	1	1	0
2	0	2	0
2	1	3	1
3	0	4	1
3	1	3	1
4	0	4	0
4	1	1	0

Input				Output	
a2	a1	a0	I	สถานะถัดไป	Y
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	2	1
0	0	1	1	1	0
0	1	0	0	2	1
0	1	0	1	3	1
0	1	1	0	4	0
0	1	1	1	3	1
1	0	0	0	4	0
1	0	0	1	1	0

การนำเอาแผนผังสถานะไปสร้างเป็นวงจร โดยใช้รอม และ แรม



Input				Output	
a2	a1	a0	I	สถานะถัดไป	Y
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	2	1
0	0	1	1	1	0
0	1	0	0	2	1
0	1	0	1	3	1
0	1	1	0	4	0
0	1	1	1	3	1
1	0	0	0	4	0
1	0	0	1	5	0

ขั้นตอนที่ 1) กำหนดหมายเลขให้สถานะต่างๆ

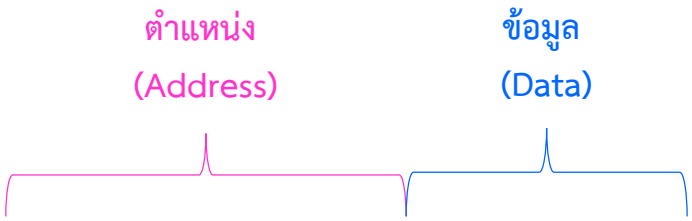
ขั้นตอนที่ 2) สร้างตารางค่าความจริง

ขั้นตอนที่ 3) กำหนด Output (Moore)

ขั้นตอนที่ 4) เปลี่ยนสถานะเป็นเลขฐานสอง

ขั้นตอนที่ 5) แปลงตารางเป็นตำแหน่งข้อมูลของรอม

การนำเอาแผนผังสถานะไปสร้างเป็นวงจร โดยใช้รอม และ แรม



Input				Output	
a2	a1	a0	I	สถานะถัดไป	Y
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	2	1
0	0	1	1	1	0
0	1	0	0	2	1
0	1	0	1	3	1
0	1	1	0	4	0
0	1	1	1	3	1
1	0	0	0	4	0
1	0	0	1	1	0

ขั้นตอนที่ 1) กำหนดหมายเลขให้สถานะต่างๆ

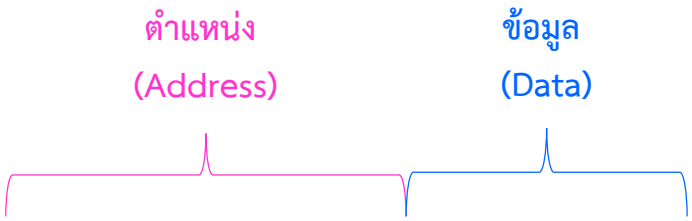
ขั้นตอนที่ 2) สร้างตารางค่าความจริง

ขั้นตอนที่ 3) กำหนด Output (Moore)

ขั้นตอนที่ 4) เปลี่ยนสถานะเป็นเลขฐานสอง

ขั้นตอนที่ 5) แปลงตารางเป็นตำแหน่งข้อมูลของรอม

การนำเอาแผนผังสถานะไปสร้างเป็นวงจร โดยใช้รอม และ แรม



Input				Output	
a2	a1	a0	I	สถานะถัดไป	Y
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	2	1
0	0	1	1	1	0
0	1	0	0	2	1
0	1	0	1	3	1
0	1	1	0	4	0
0	1	1	1	3	1
1	0	0	0	4	0
1	0	0	1	1	0

ขั้นตอนที่ 1) กำหนดหมายเลขให้สถานะต่างๆ

ขั้นตอนที่ 2) สร้างตารางค่าความจริง

ขั้นตอนที่ 3) กำหนด Output (Moore)

ขั้นตอนที่ 4) เปลี่ยนสถานะเป็นเลขฐานสอง

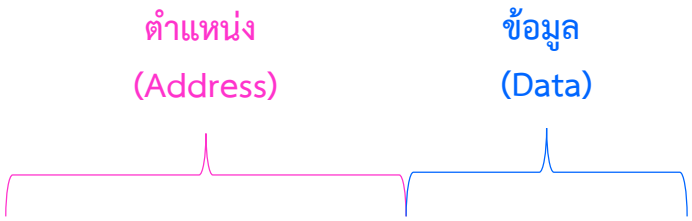
ขั้นตอนที่ 5) แปลงตารางเป็นตำแหน่งข้อมูลของรอม

ขั้นตอนที่ 6) บันทึกข้อมูลลงไปในรอม

เพื่อให้ง่ายในการป้อนข้อมูล เราจะแปลง Address ให้เป็นเลขฐาน 16

เครื่องจักรสถานะจำกัด

การนำเอาแผนผังสถานะไปสร้างเป็นวงจร โดยใช้รอม และ แรม



เพื่อให้ง่ายในการป้อนข้อมูล เราจะแปลง Address และข้อมูล ให้เป็นเลขฐาน 16

Input				Output	
a2	a1	a0	I	สถานะถัดไป	Y
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	2	1
0	0	1	1	1	0
0	1	0	0	2	1
0	1	0	1	3	1
0	1	1	0	4	0
0	1	1	1	3	1
1	0	0	0	4	0
1	0	0	1	5	0

	Output	
Address	สถานะถัดไป	Y
00	00	0
01	01	0
02	02	1
03	01	0
04	02	1
05	03	1
06	04	0
07	03	1
08	04	0
09	01	0

การนำเอาแผนผังสถานะไปสร้างเป็นวงจร โดยใช้รอม และ แรม

	Output	
Address	สถานะถัดไป	Y
00	00	0
01	01	0
02	02	1
03	01	0
04	02	1
05	03	1
06	04	0
07	03	1
08	04	0
09	01	0

Address bit width: 8

Data bit width: 8

Data

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	01	02	03	04	03	04	01	00	00	00	00	00	00
1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

การนำเอาแผนผังสถานะไปสร้างเป็นวงจร โดยใช้รอม และ แรม

	Output	
Address	สถานะถัดไป	Y
00	00	0
01	01	0
02	02	1
03	01	0
04	02	1
05	03	1
06	04	0
07	03	1
08	04	0
09	01	0

ขั้นตอนที่ 1) กำหนดหมายเลขให้สถานะต่างๆ

ขั้นตอนที่ 2) สร้างตารางค่าความจริง

ขั้นตอนที่ 3) กำหนด Output (Moore)

ขั้นตอนที่ 4) เปลี่ยนสถานะเป็นเลขฐานสอง

ขั้นตอนที่ 5) แปลงตารางเป็นตำแหน่งข้อมูลของรอม

ขั้นตอนที่ 6) บันทึกข้อมูลลงไปในรอม

ขั้นตอนที่ 7) ถอดรหัส Output จากสถานะถัดไป

การนำเอาแผนผังสถานะไปสร้างเป็นวงจร โดยใช้รอม และ แรม

	Output	
Address	สถานะถัดไป	Y
00	00	0
01	01	0
02	02	1
03	01	0
04	02	1
05	03	1
06	04	0
07	03	1
08	04	0
09	01	0

ขั้นตอนที่ 7) ถอดรหัส Output จากสถานะถัดไป

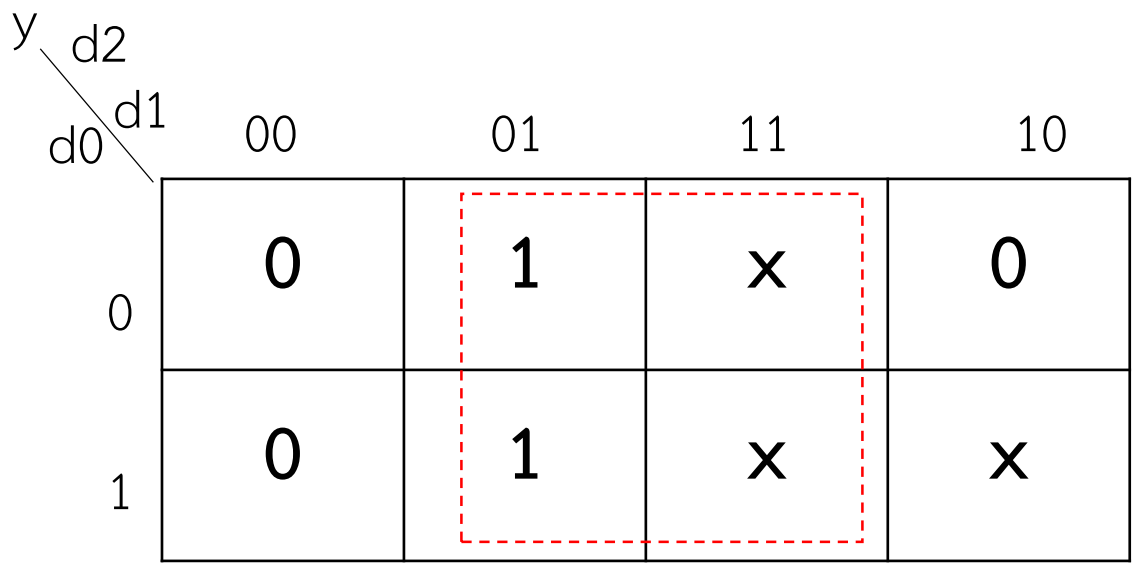
d2	d1	d0	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	X
1	0	0	X
1	0	1	X
1	1	0	X
1	1	1	X

ส่วนที่ไม่มีข้อมูล หรือไม่มีสถานะ ให้เติมเป็น don’t care

การนำเอาแผนผังสถานะไปสร้างเป็นวงจร โดยใช้รอม และ แรม

ขั้นตอนที่ 7) ถอดรหัส Output จากสถานะถัดไป

Input			Output
d2	d1	d0	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	X
1	0	0	X
1	0	1	X
1	1	0	X
1	1	1	X



y=d1

การนำเอาแผนผังสถานะไปสร้างเป็นวงจร โดยใช้รอม และ แรม

ขั้นตอนที่ 1) กำหนดหมายเลขให้สถานะต่างๆ

ขั้นตอนที่ 2) สร้างตารางค่าความจริง

ขั้นตอนที่ 3) กำหนด Output (Moore)

ขั้นตอนที่ 4) เปลี่ยนสถานะเป็นเลขฐานสอง

ขั้นตอนที่ 5) แปลงตารางเป็นตำแหน่งข้อมูลของรอม

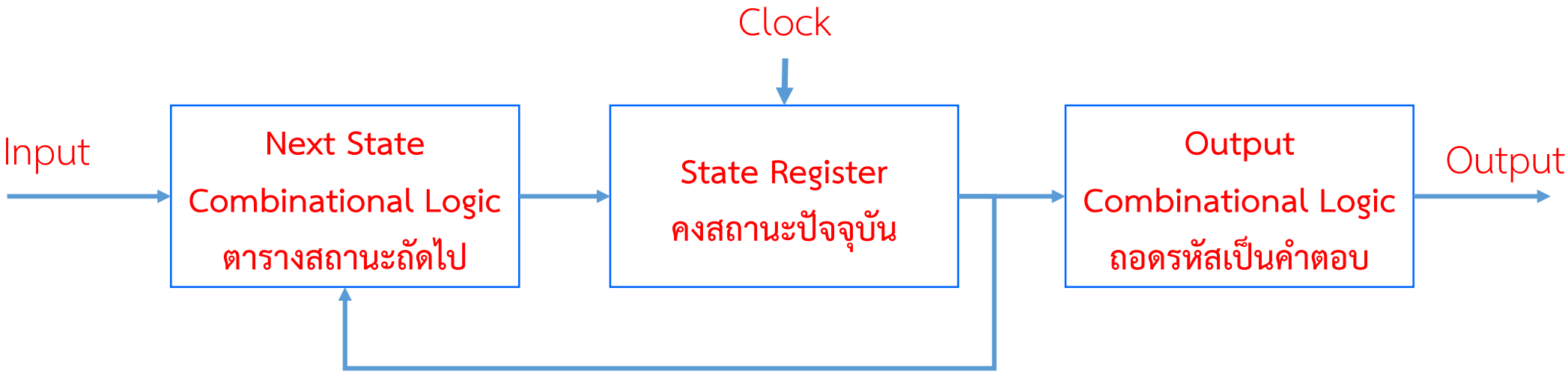
ขั้นตอนที่ 6) บันทึกข้อมูลลงไปในรอม

ขั้นตอนที่ 7) ถอดรหัส Output จากสถานะถัดไป

ขั้นตอนที่ 8) ต่อวงจร Moore Machine

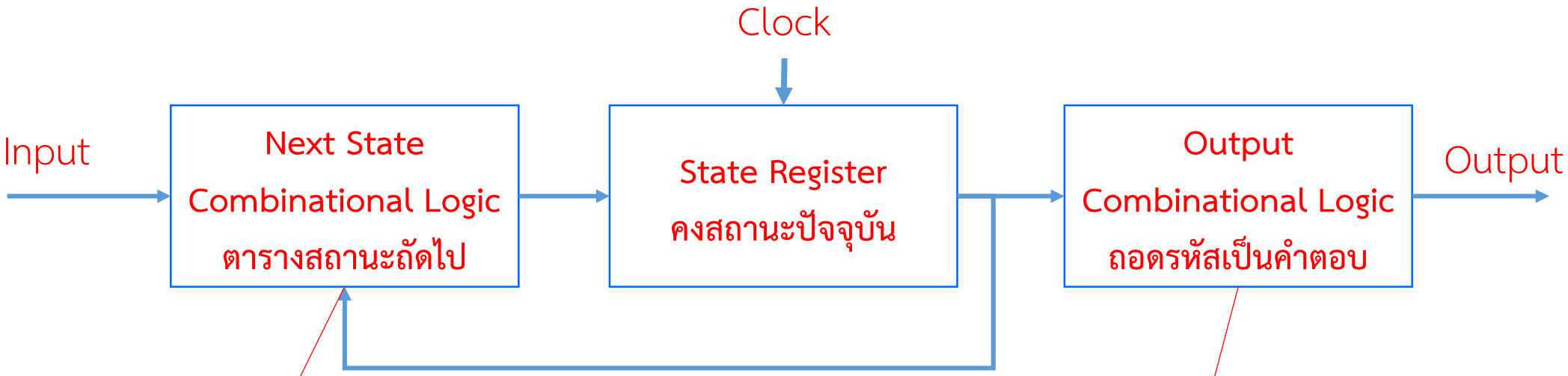
การนำเอาแผนผังสถานะไปสร้างเป็นวงจร โดยใช้รอม และ แรม

ขั้นตอนที่ 8) ต่อวงจร Moore Machine



ROM D-Flip Flop Combination Circuit

เครื่องจักรสถานะจำกัด



Address bit width: 8

Data bit width: 8

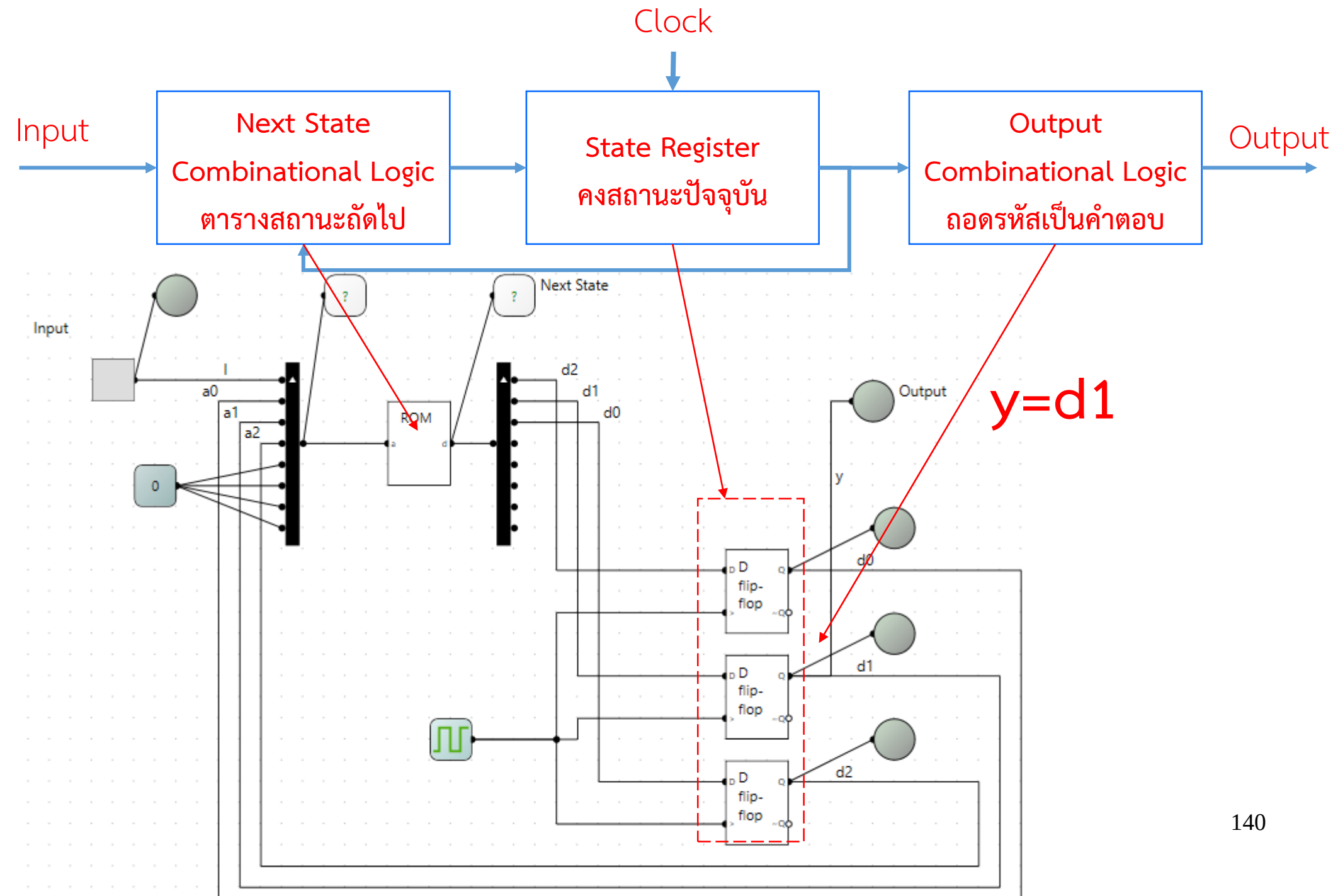
Data

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	01	02	03	04	03	04	01	00	00	00	00	00	00
1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

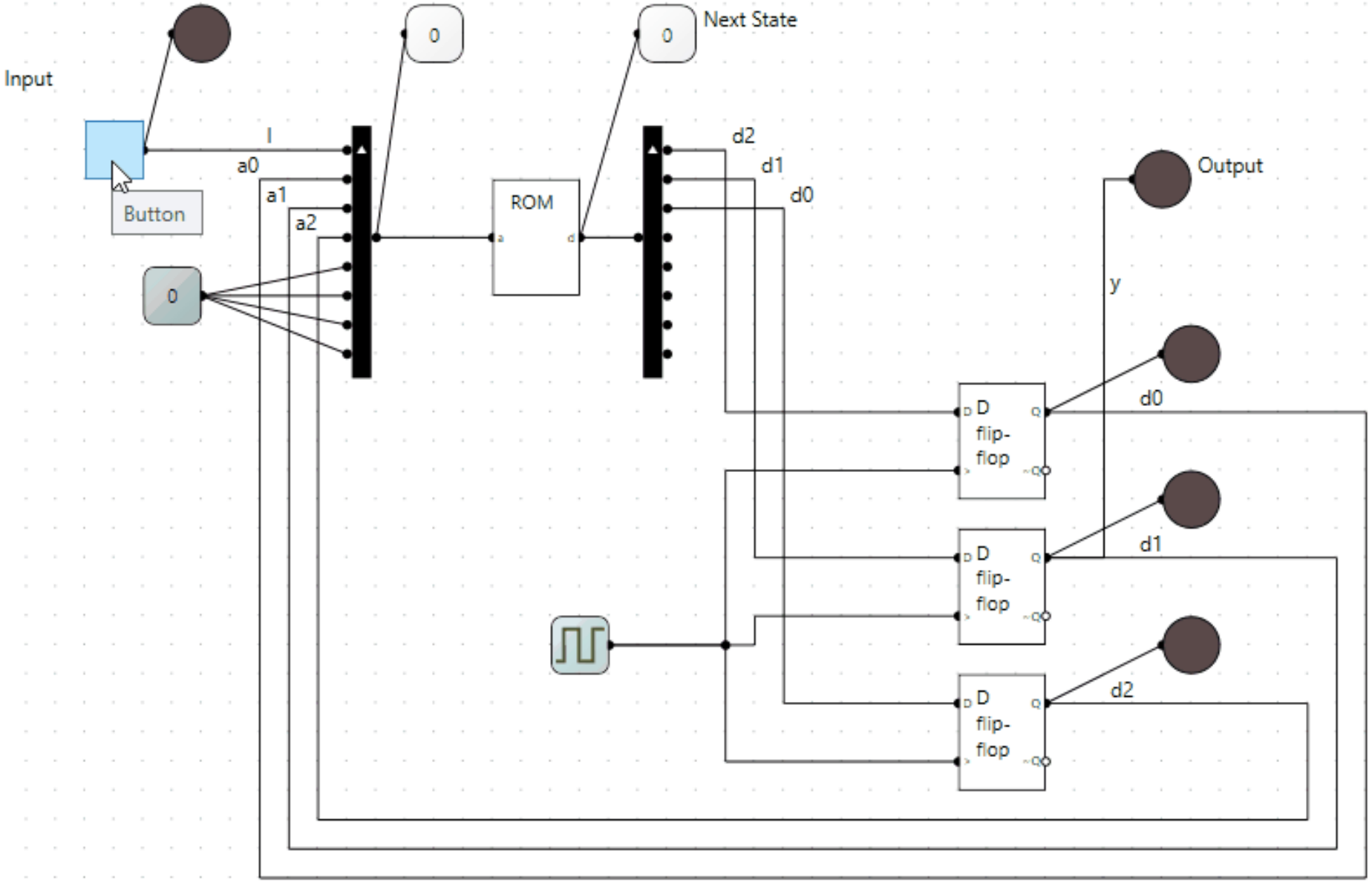
y (d2 d1 / d0)	00	01	11	10
0	0	1	x	0
1	0	1	x	x

y=d1

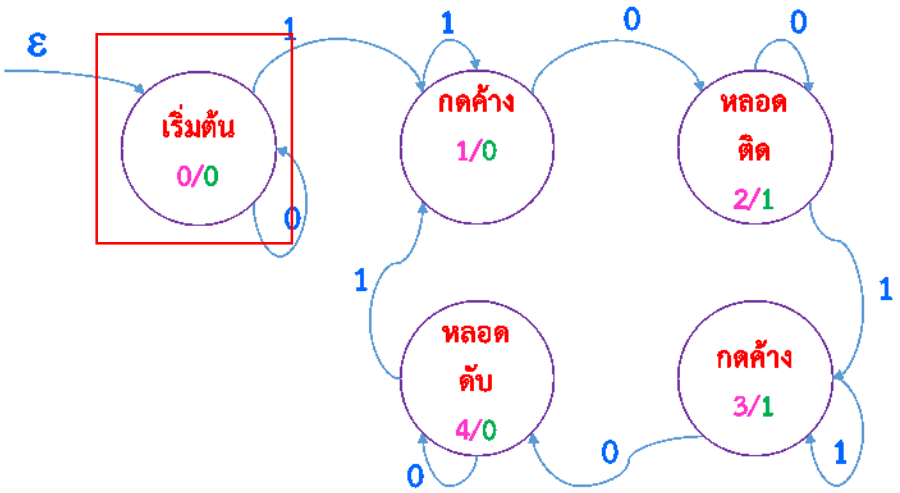
เครื่องจักรสถานะจำกัด

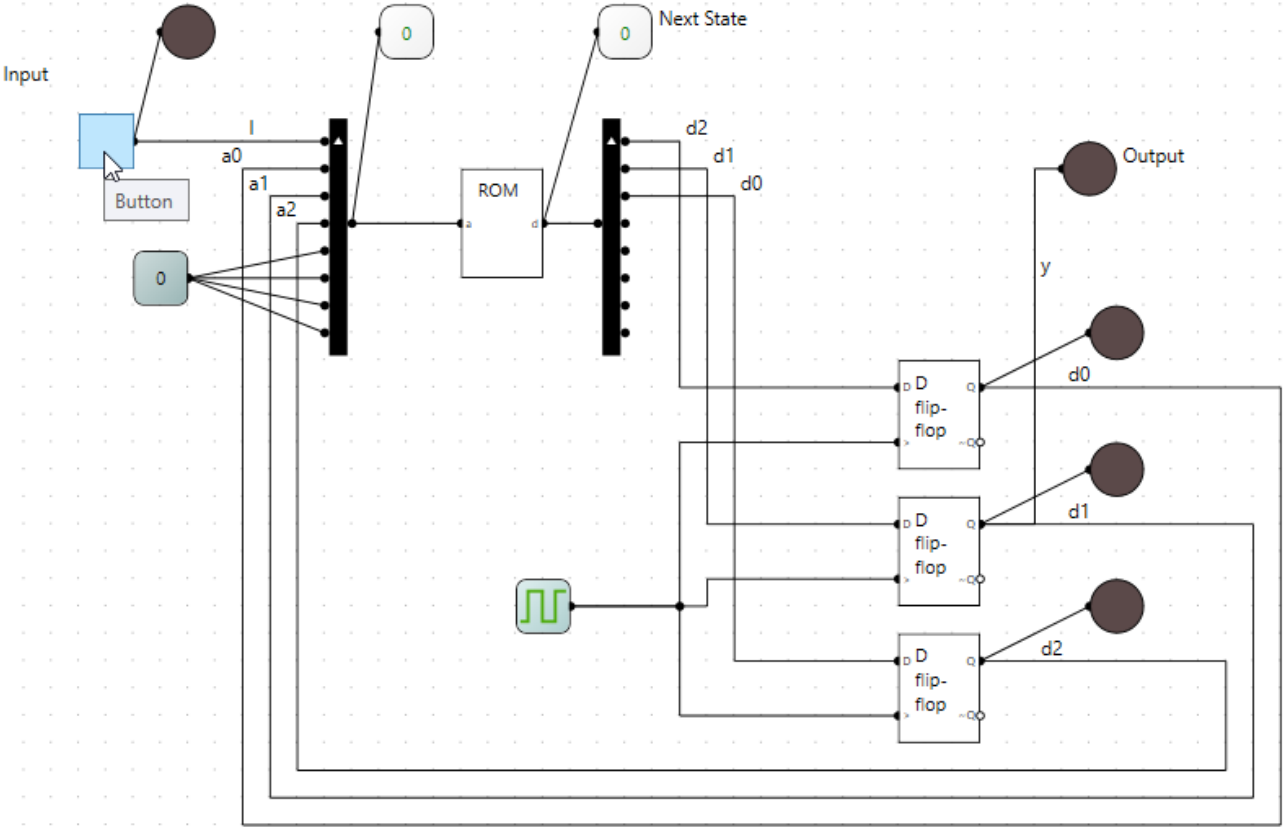


เครื่องจักรสถานะจำกัด



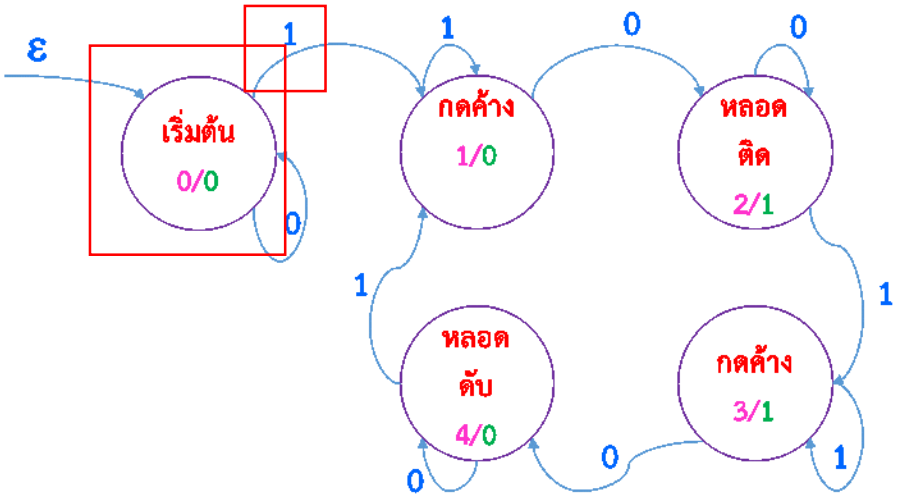
เครื่องจักรสถานะจำกัด



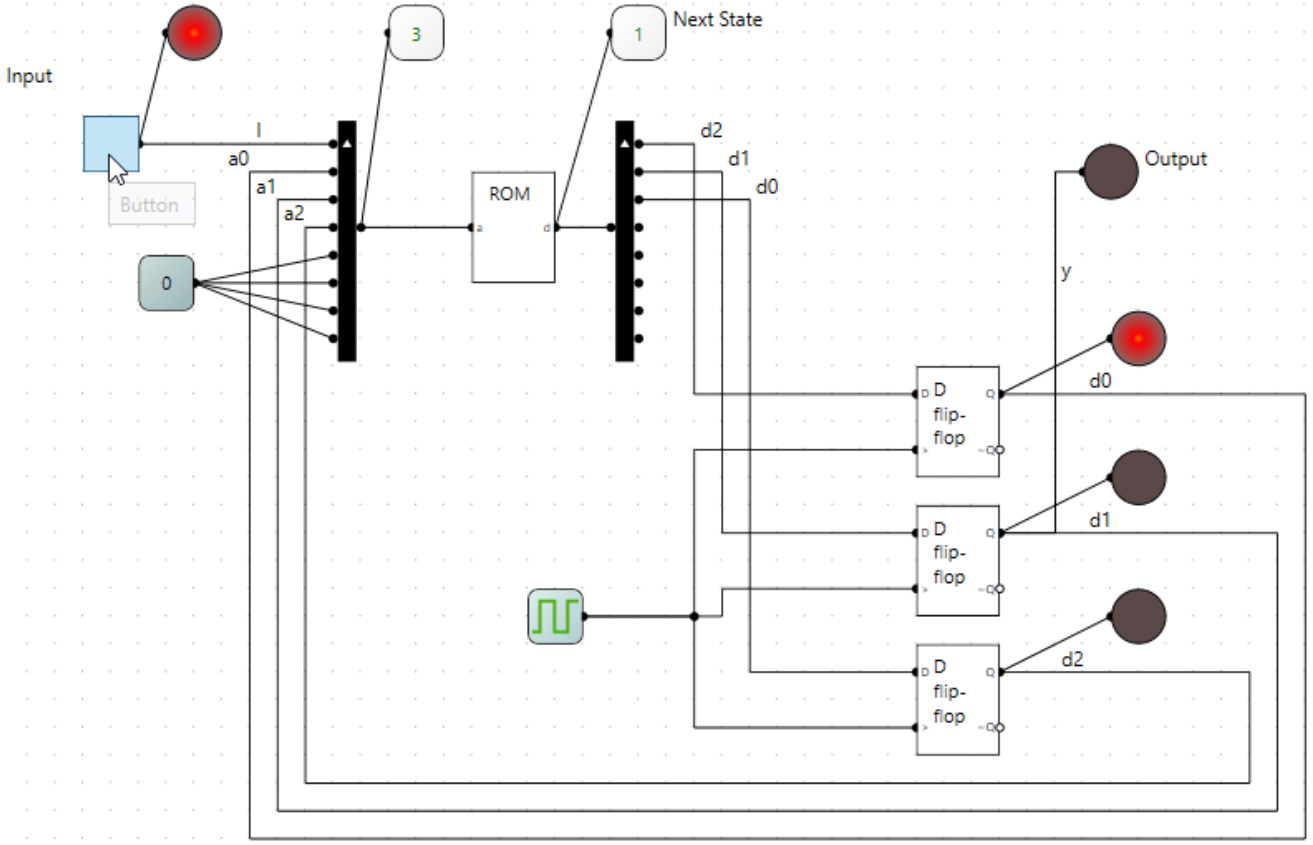


Input				Output	
a2	a1	a0	I	สถานะถัดไป	Y
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	2	1
0	0	1	1	1	0
0	1	0	0	2	1
0	1	0	1	3	1
0	1	1	0	4	0
0	1	1	1	3	1
1	0	0	0	4	0
1	0	0	1	5	0

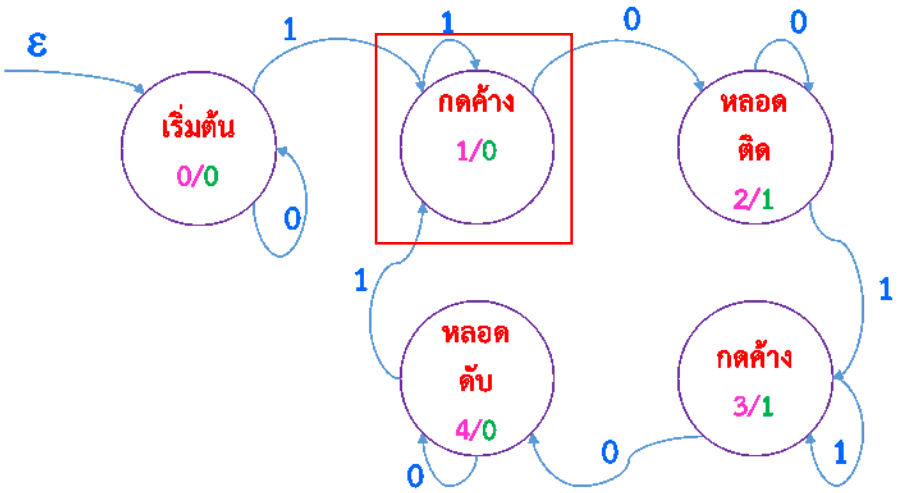
เครื่องจักรสถานะจำกัด



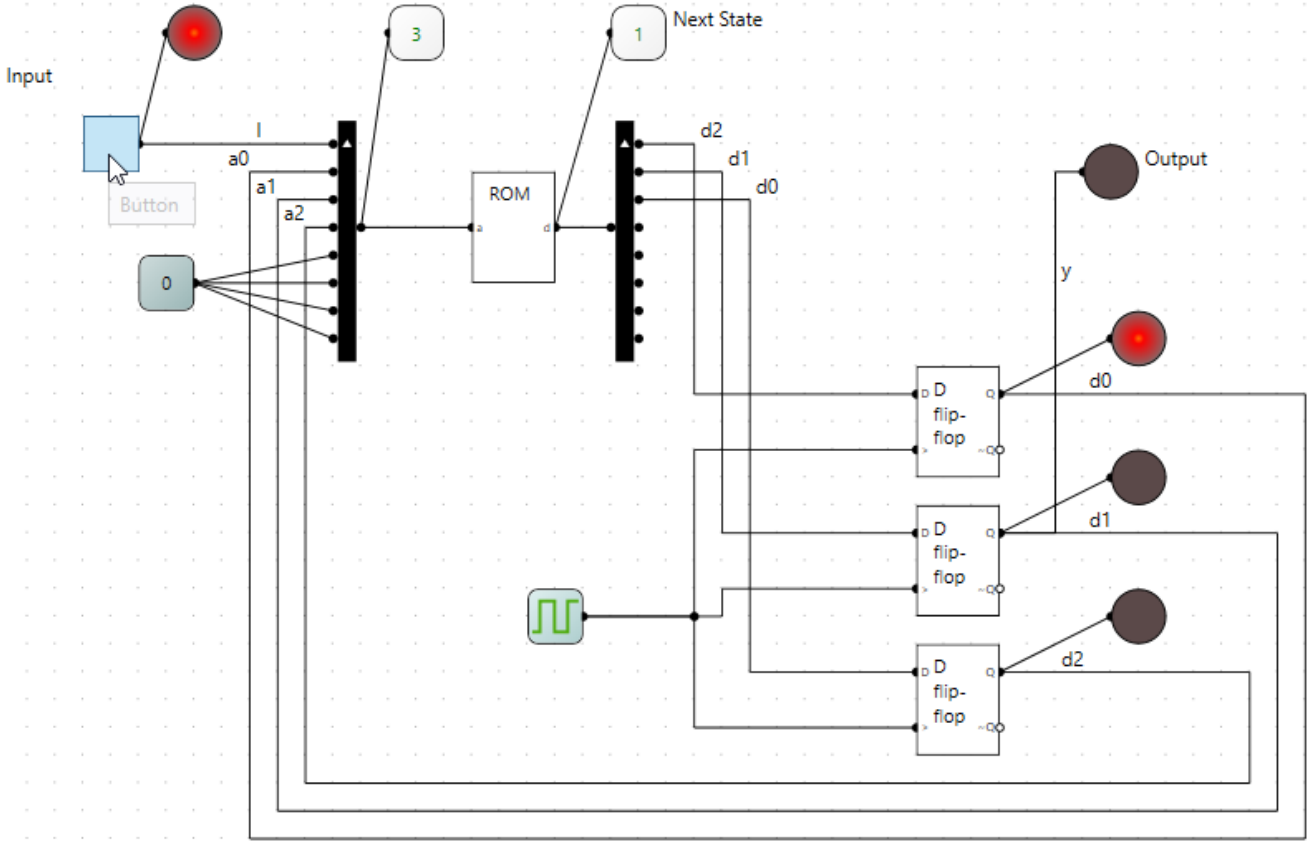
Input				Output	
a2	a1	a0	I	สถานะถัดไป	Y
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	2	1
0	0	1	1	1	0
0	1	0	0	2	1
0	1	0	1	3	1
0	1	1	0	4	0
0	1	1	1	3	1
1	0	0	0	4	0
1	0	0	1	5	0



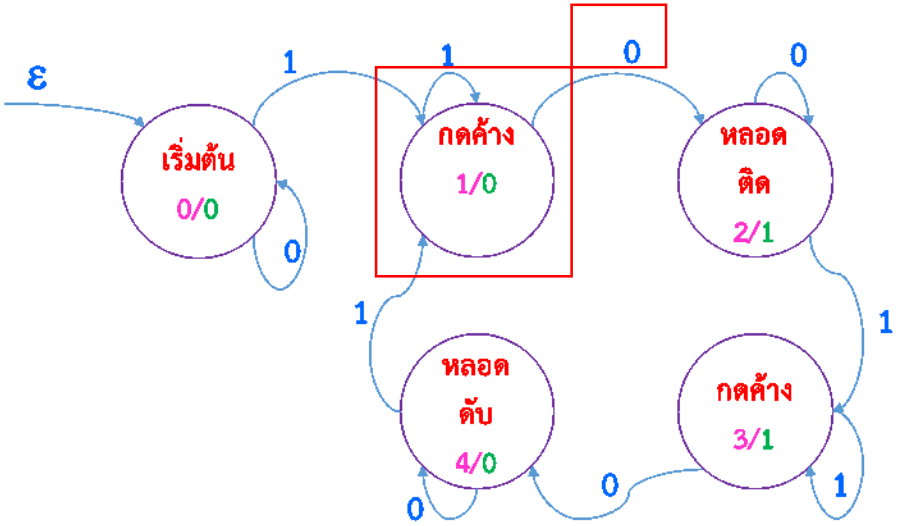
เครื่องจักรสถานะจำกัด



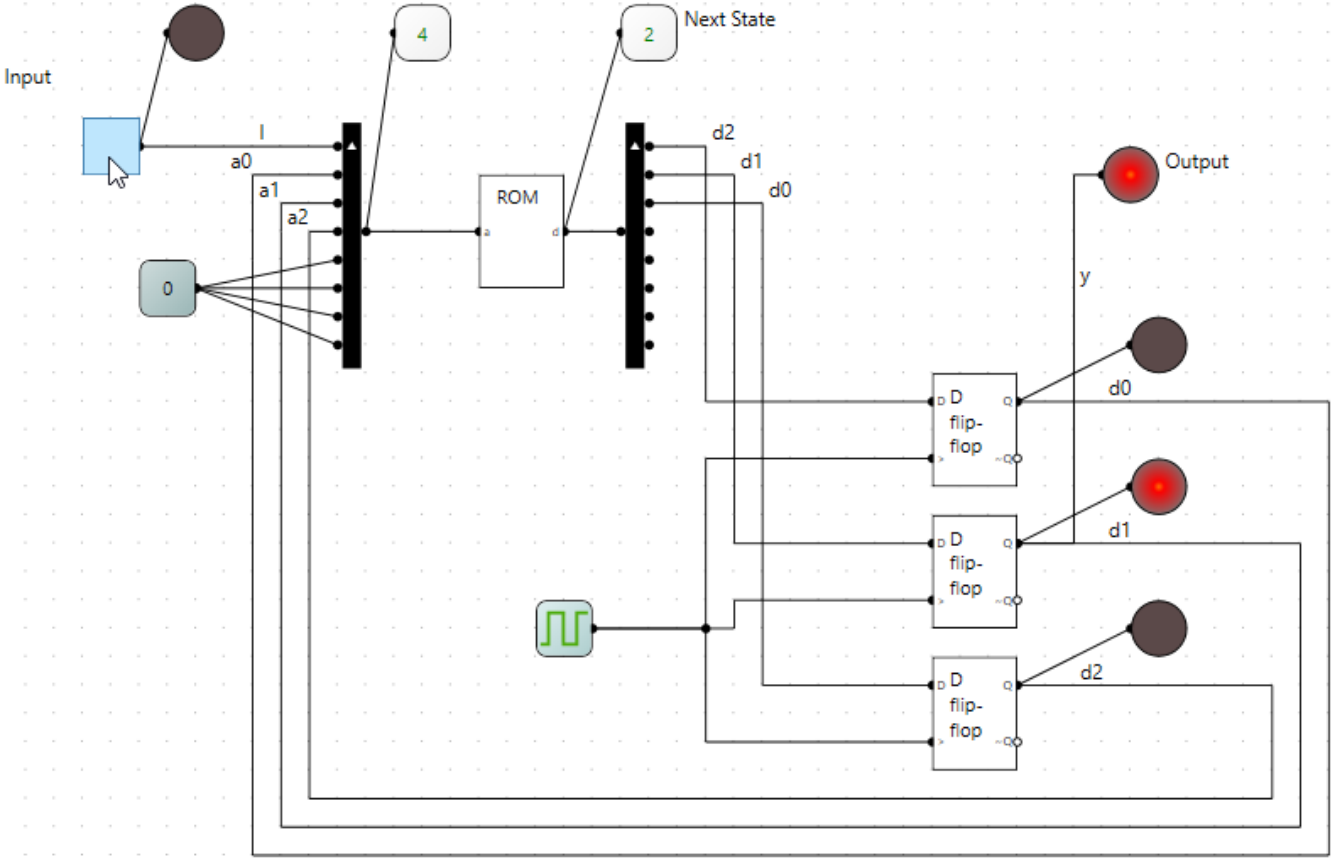
Input				Output	
a2	a1	a0	I	สถานะถัดไป	Y
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	2	1
0	0	1	1	1	0
0	1	0	0	2	1
0	1	0	1	3	1
0	1	1	0	4	0
0	1	1	1	3	1
1	0	0	0	4	0
1	0	0	1	5	0



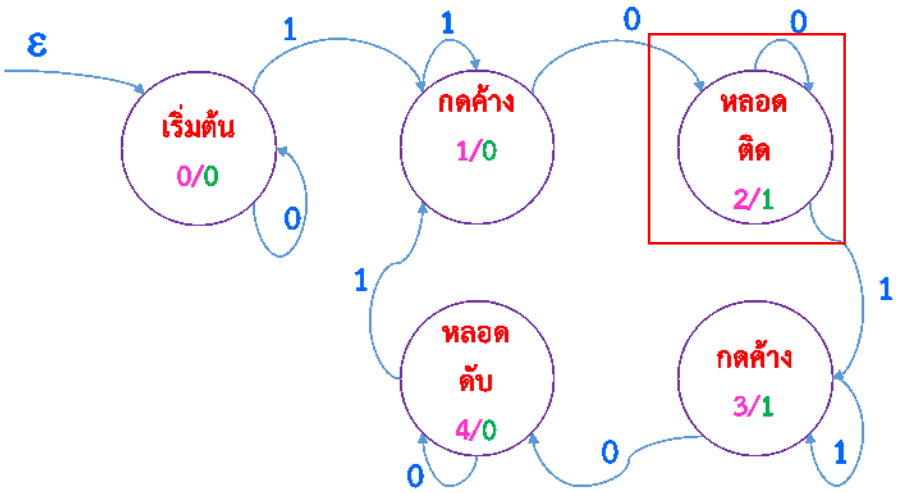
เครื่องจักรสถานะจำกัด

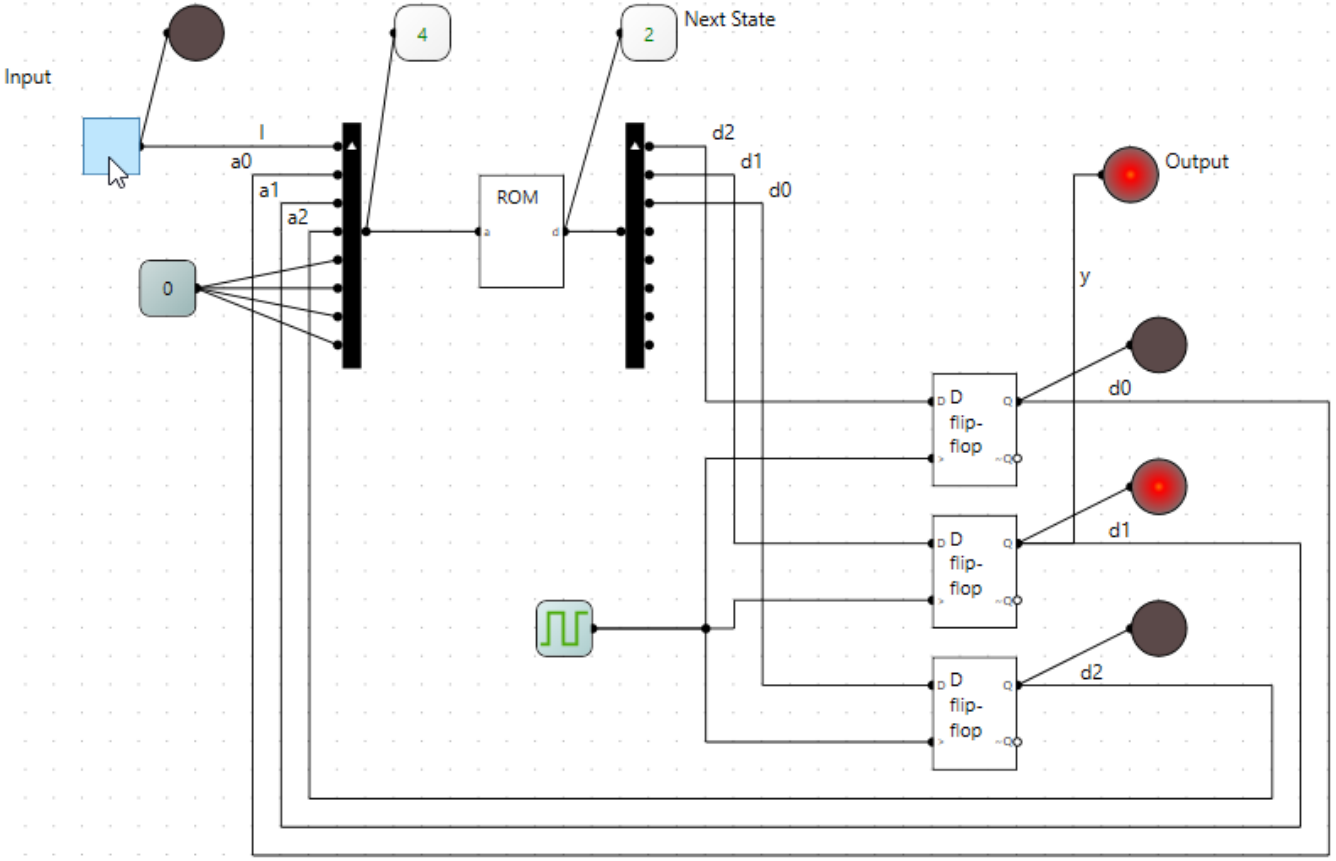


Input				Output	
a2	a1	a0	I	สถานะถัดไป	Y
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	2	1
0	0	1	1	1	0
0	1	0	0	2	1
0	1	0	1	3	1
0	1	1	0	4	0
0	1	1	1	3	1
1	0	0	0	4	0
1	0	0	1	5	0



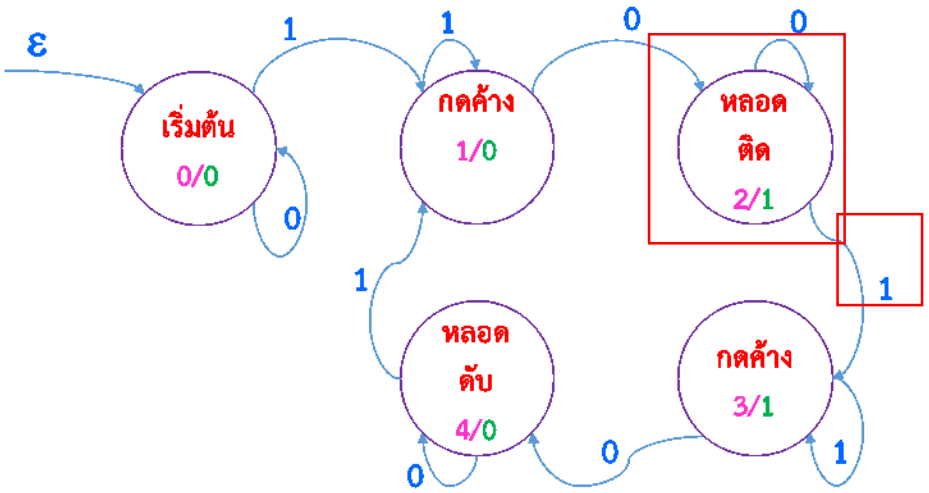
เครื่องจักรสถานะจำกัด

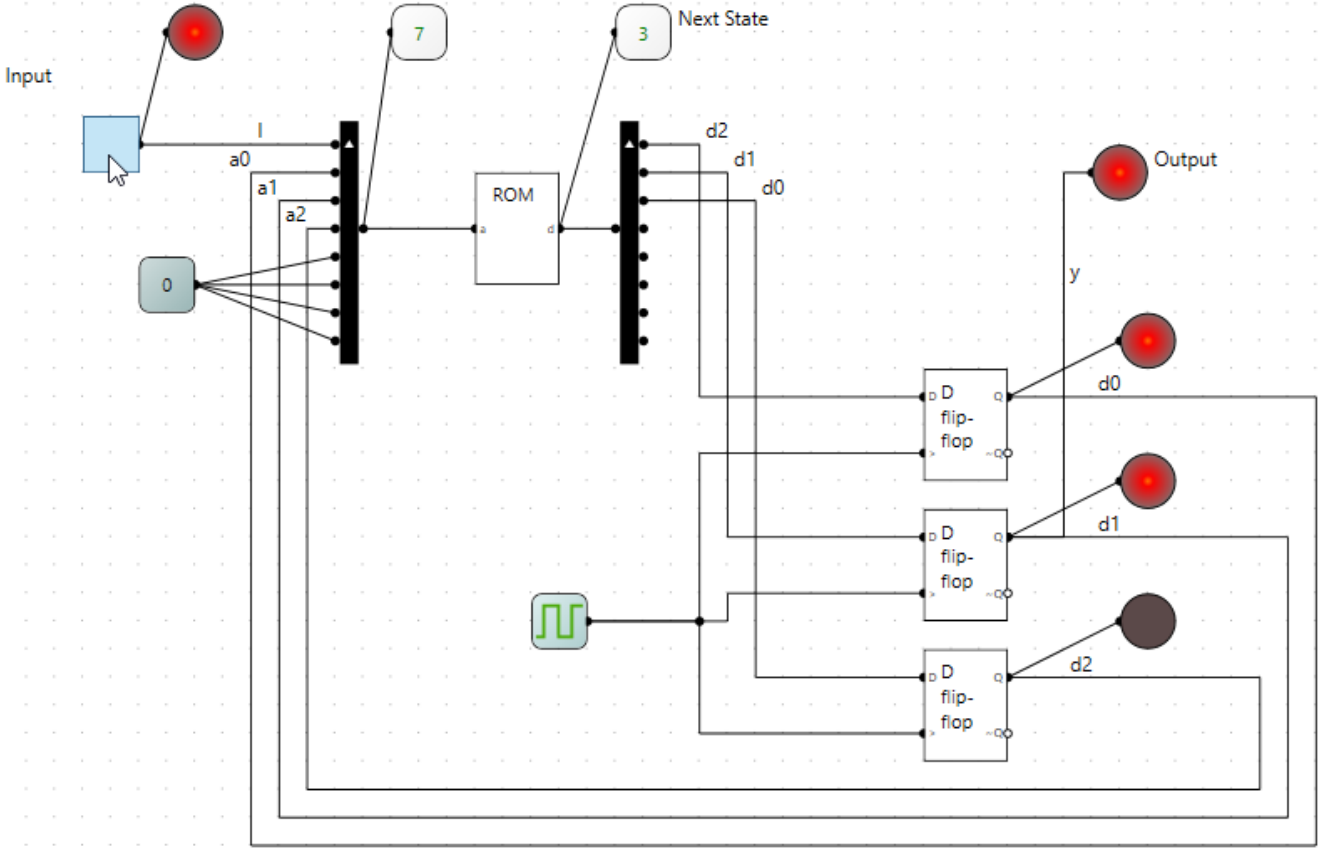




Input				Output	
a2	a1	a0	I	สถานะถัดไป	Y
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	2	1
0	0	1	1	1	0
0	1	0	0	2	1
0	1	0	1	3	1
0	1	1	0	4	0
0	1	1	1	3	1
1	0	0	0	4	0
1	0	0	1	5	0

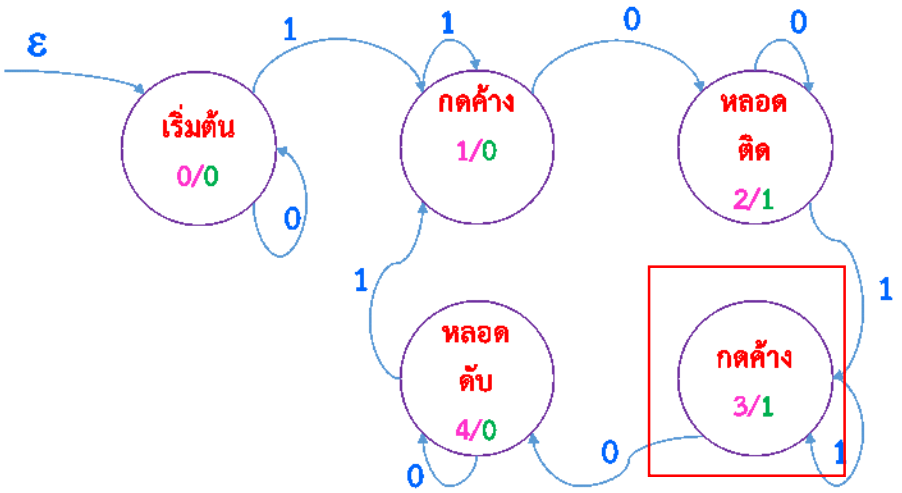
เครื่องจักรสถานะจำกัด

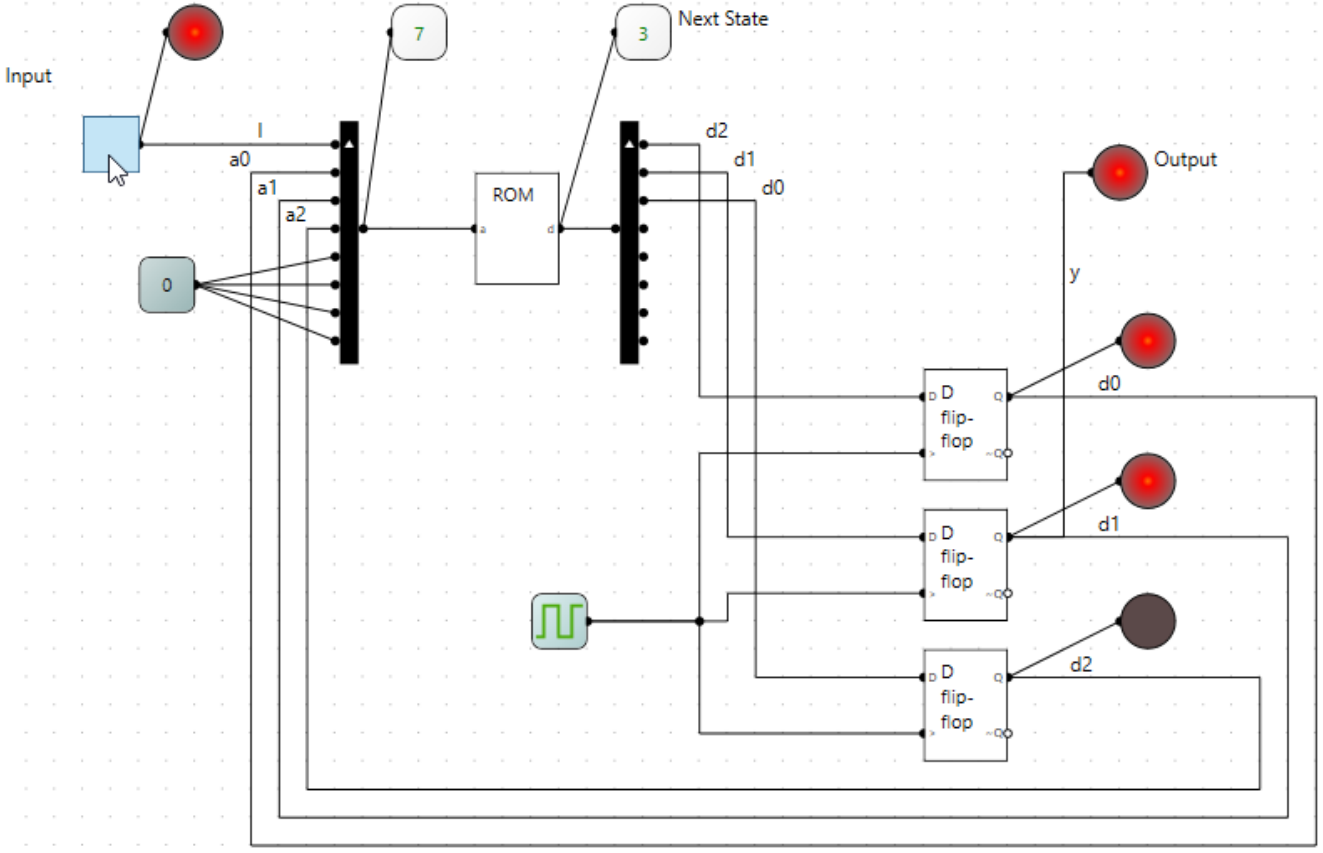




Input				Output	
a2	a1	a0	I	สถานะถัดไป	Y
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	2	1
0	0	1	1	1	0
0	1	0	0	2	1
0	1	0	1	3	1
0	1	1	0	4	0
0	1	1	1	3	1
1	0	0	0	4	0
1	0	0	1	5	0

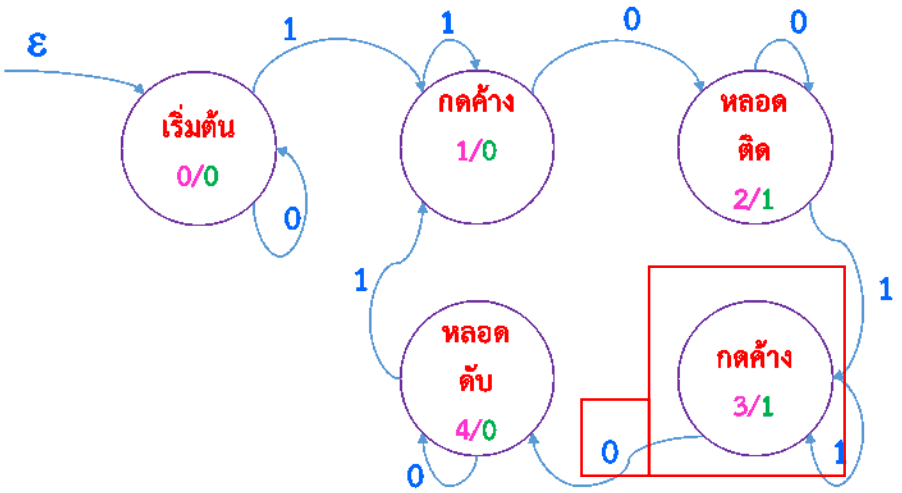
เครื่องจักรสถานะจำกัด



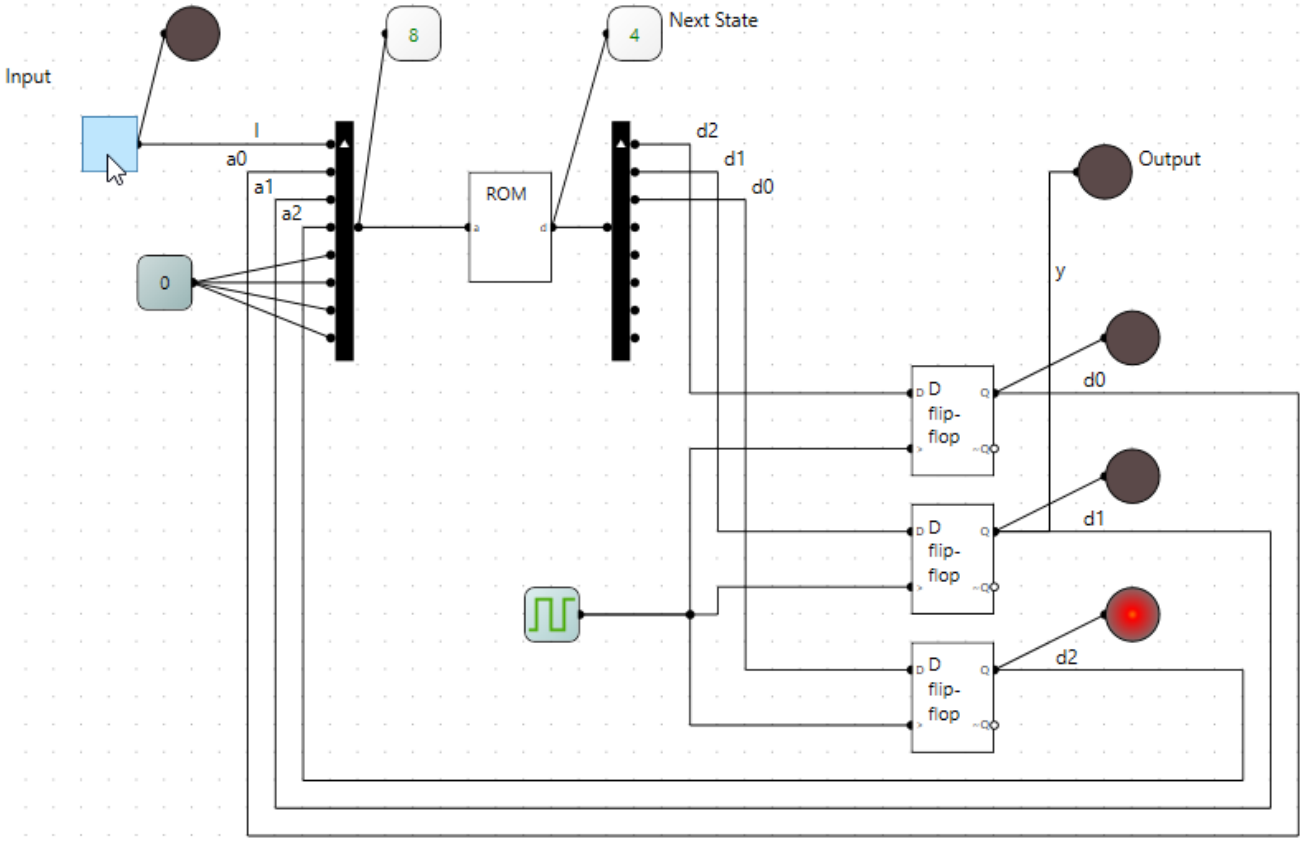


Input				Output	
a2	a1	a0	I	สถานะถัดไป	Y
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	2	1
0	0	1	1	1	0
0	1	0	0	2	1
0	1	0	1	3	1
0	1	1	0	4	0
0	1	1	1	3	1
1	0	0	0	4	0
1	0	0	1	5	0

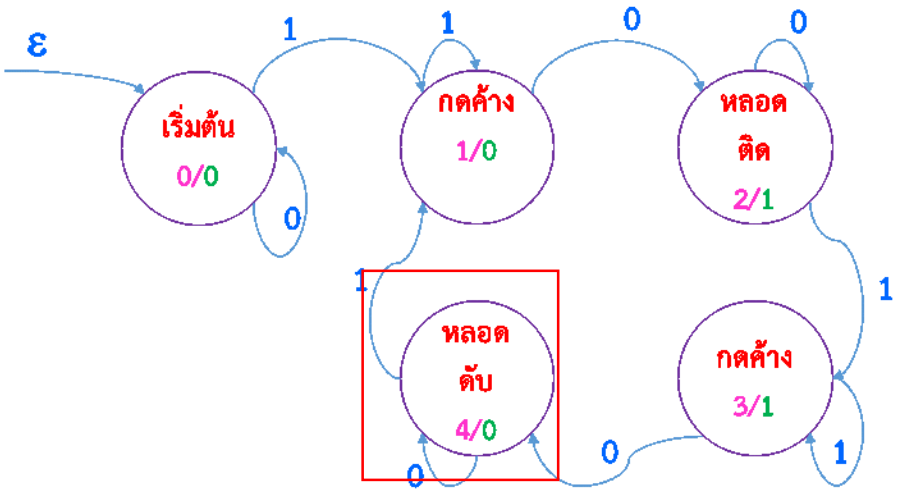
เครื่องจักรสถานะจำกัด



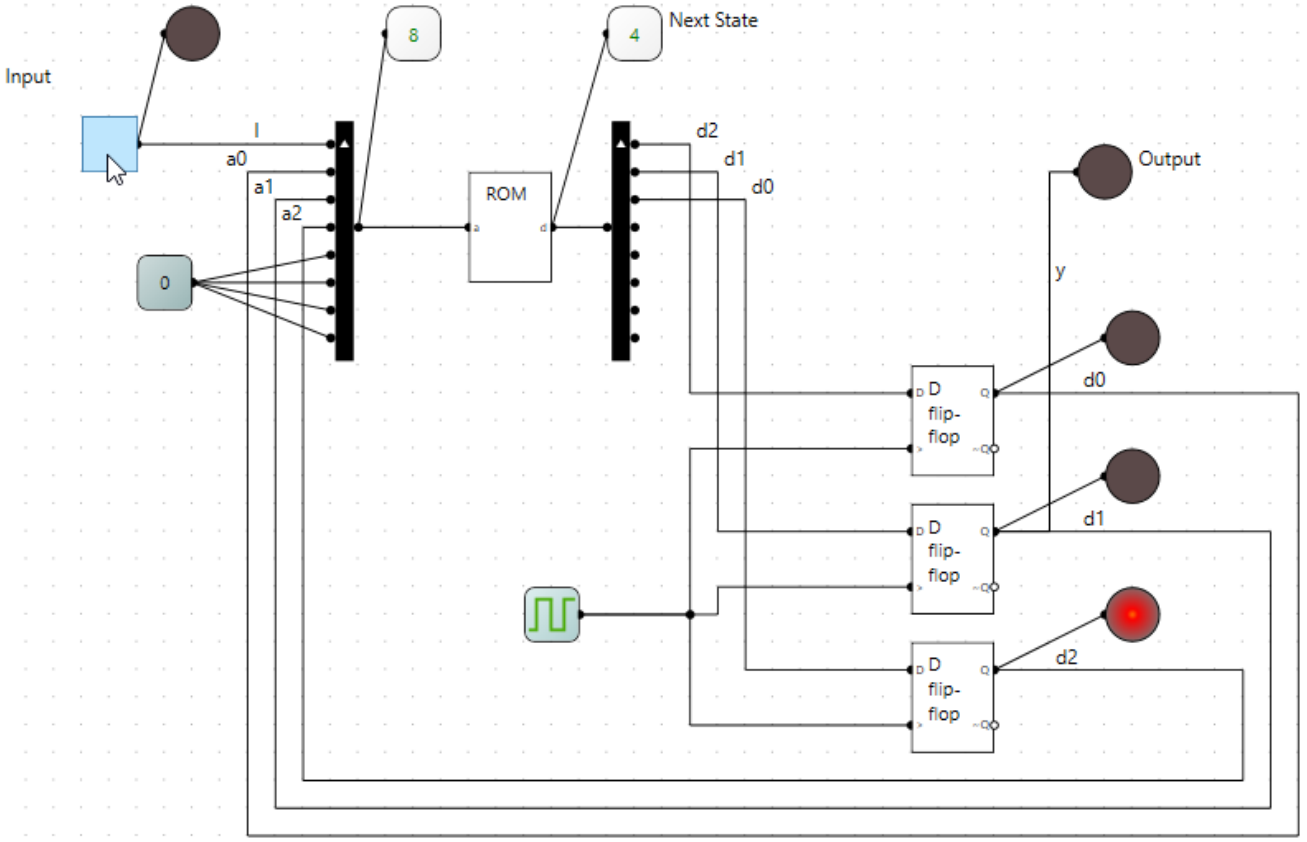
Input				Output	
a2	a1	a0	I	สถานะถัดไป	Y
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	2	1
0	0	1	1	1	0
0	1	0	0	2	1
0	1	0	1	3	1
0	1	1	0	4	0
0	1	1	1	3	1
1	0	0	0	4	0
1	0	0	1	5	0



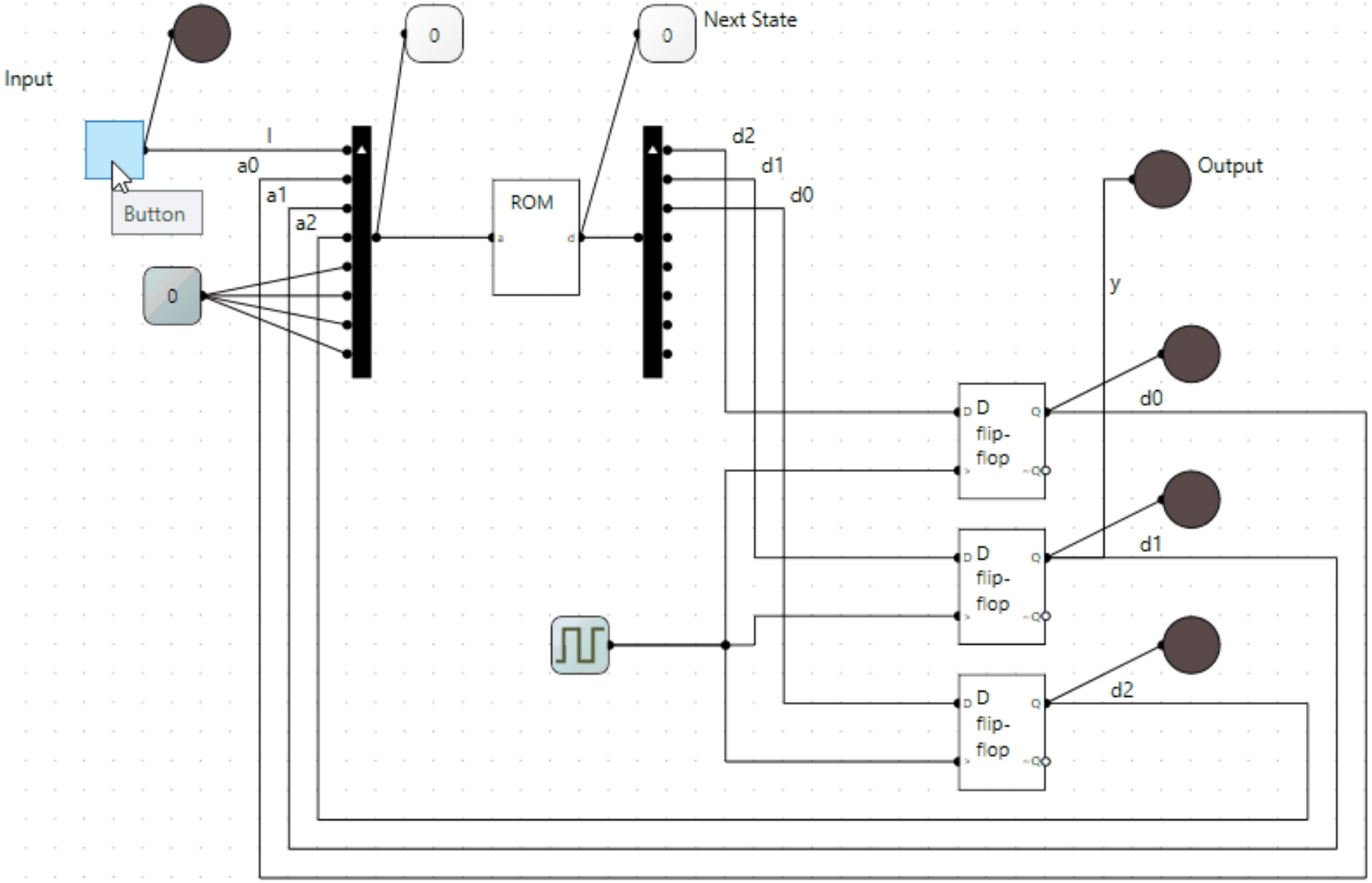
เครื่องจักรสถานะจำกัด



Input				Output	
a2	a1	a0	I	สถานะถัดไป	Y
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	2	1
0	0	1	1	1	0
0	1	0	0	2	1
0	1	0	1	3	1
0	1	1	0	4	0
0	1	1	1	3	1
1	0	0	0	4	0
1	0	0	1	5	0



เครื่องจักรสถานะจำกัด



สถานะของวงจร

เซ็นเซอร์วัดระดับน้ำ a (0=ไม่จุ่มน้ำ, 1=จุ่มน้ำ)

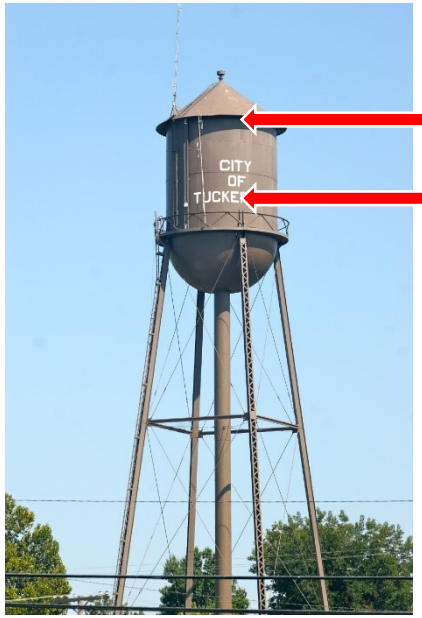
เซ็นเซอร์วัดระดับน้ำ b (0=ไม่จุ่มน้ำ, 1=จุ่มน้ำ)

b เป็น 0 ให้เปิดปั๊ม (y=1)

a เป็น 1 ให้ปิดปั๊ม (y=0)

ปั๊มน้ำ y (0 =ปิด, 1=เปิด)

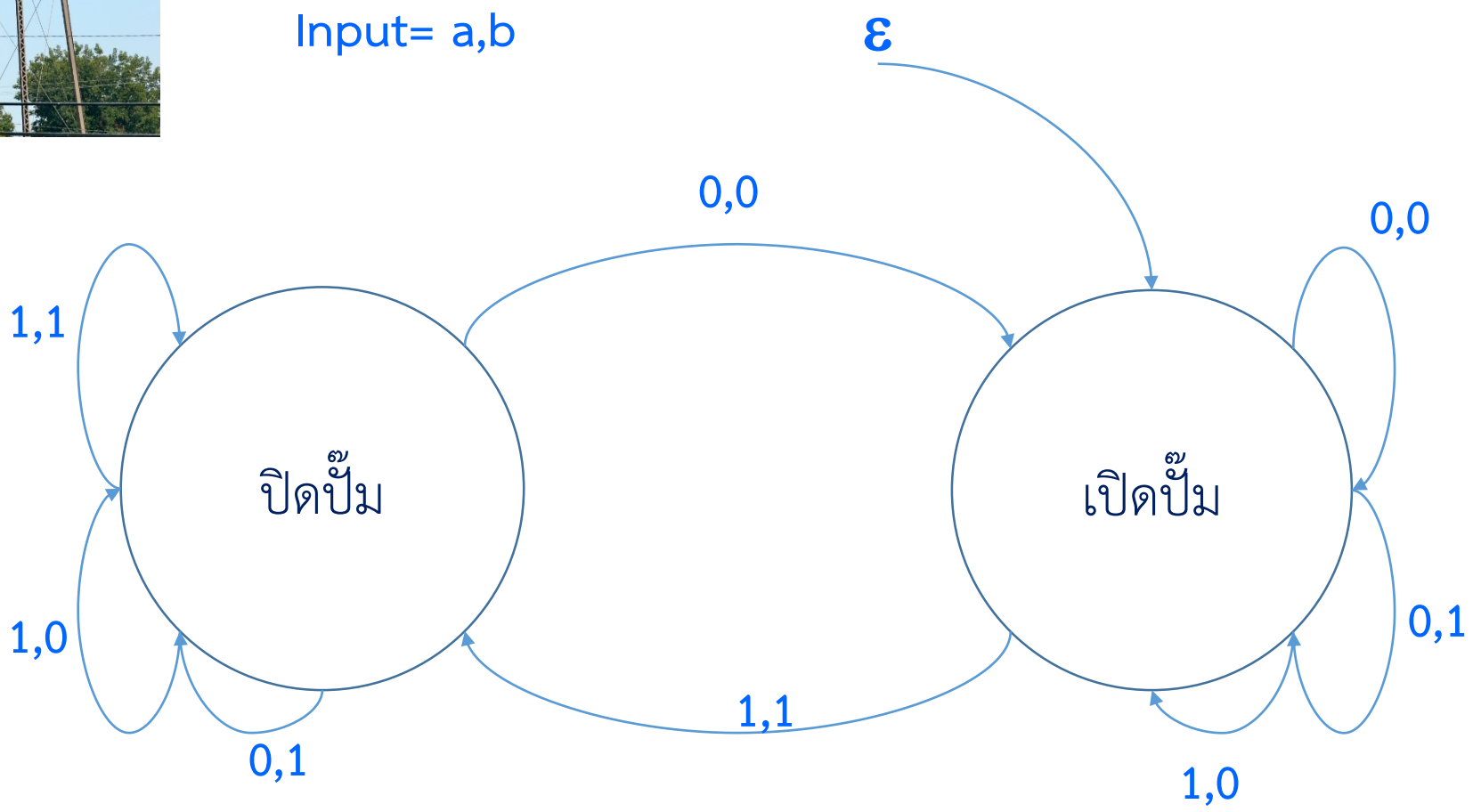
สถานะของวงจร



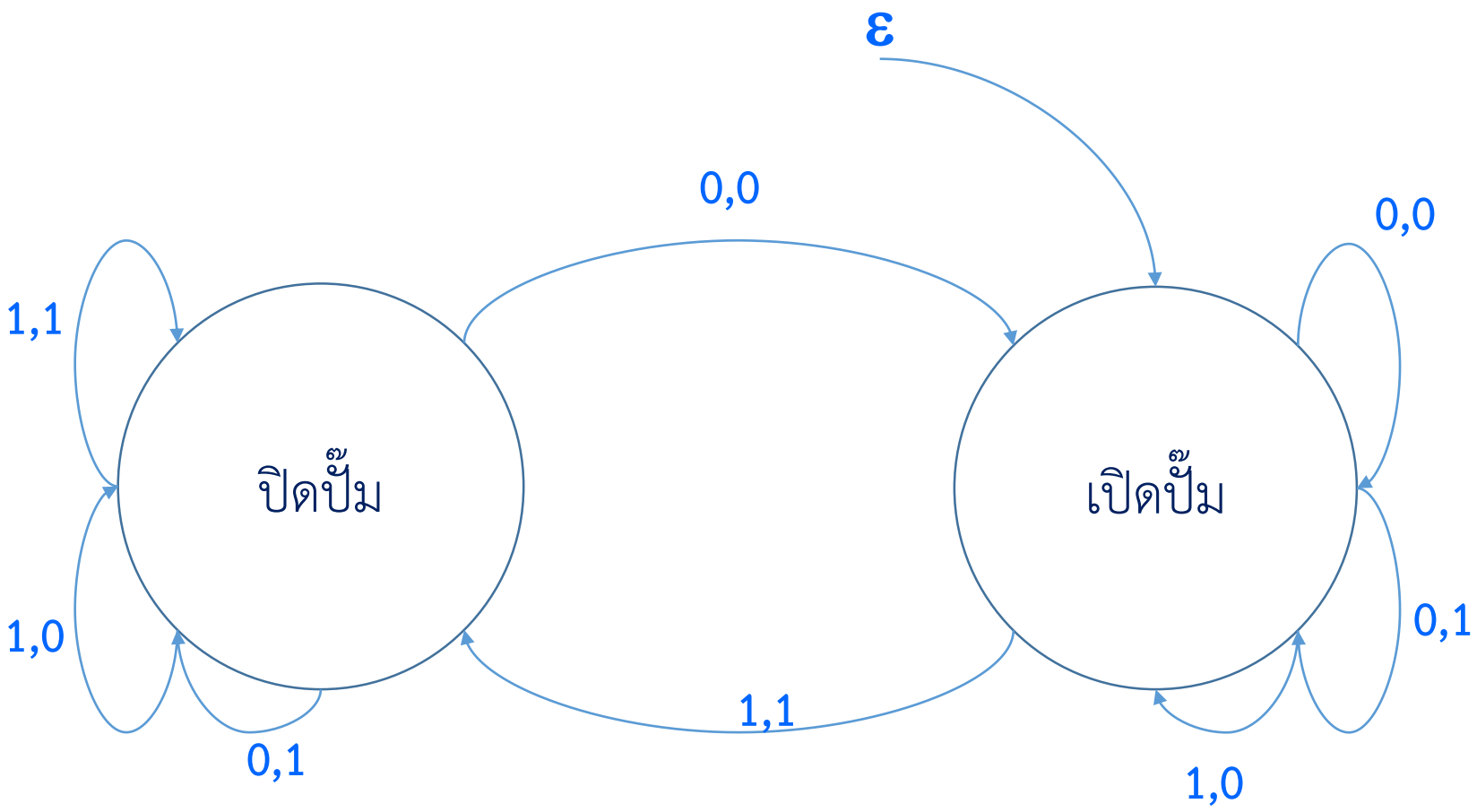
เซ็นเซอร์วัดระดับน้ำ a (0=ไม่จุ่มน้ำ, 1=จุ่มน้ำ)

เซ็นเซอร์วัดระดับน้ำ b (0=ไม่จุ่มน้ำ, 1=จุ่มน้ำ)

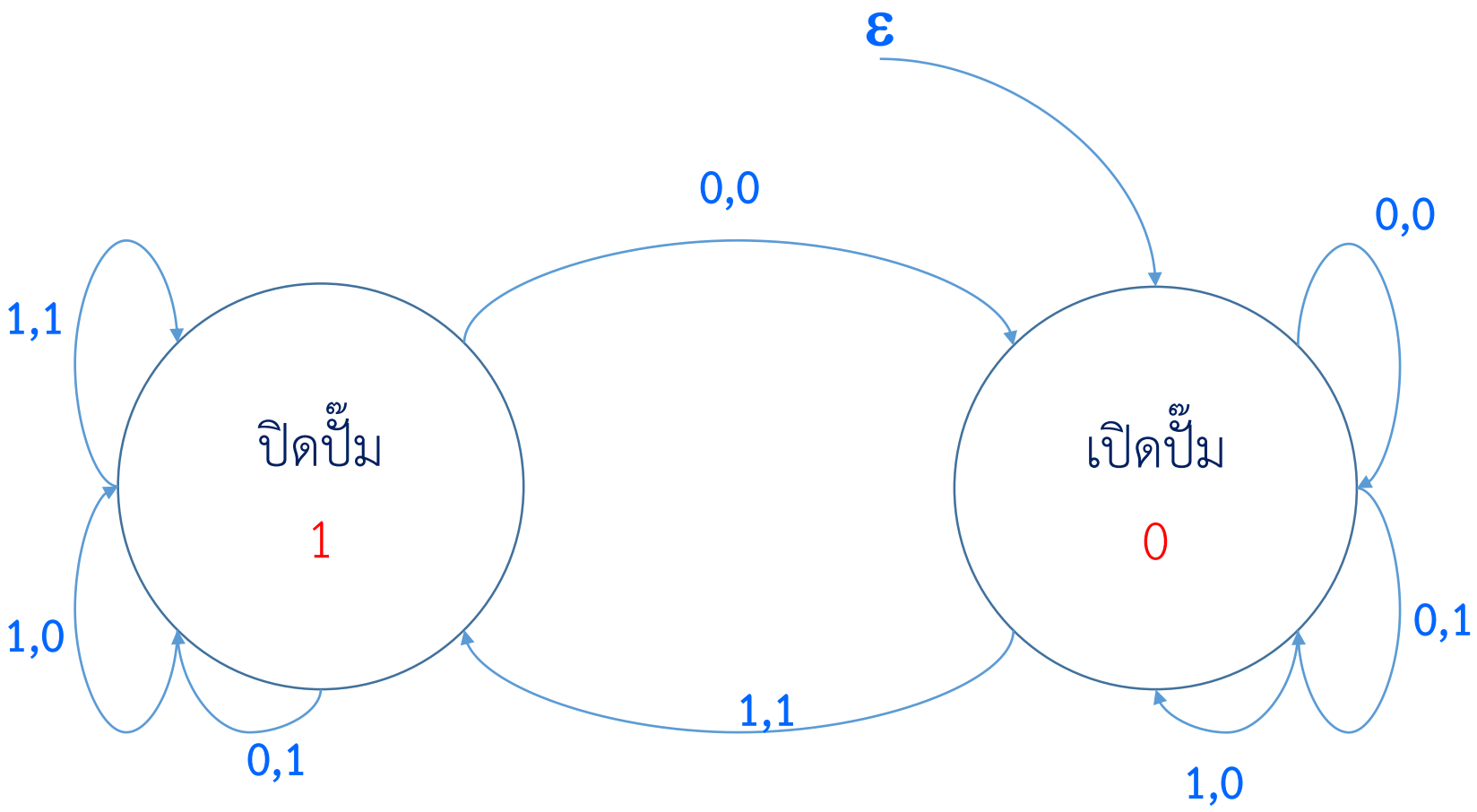
Input= a,b



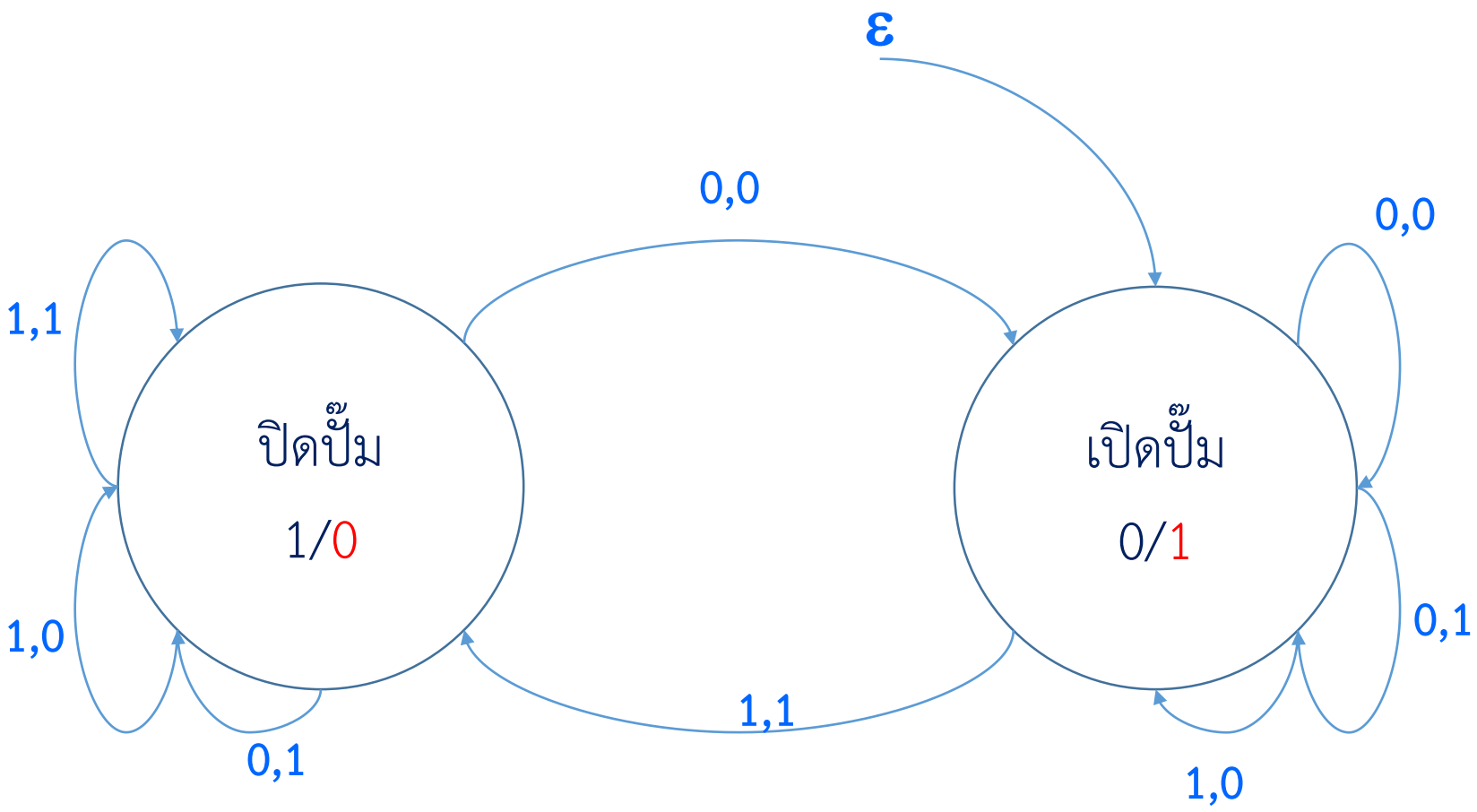
สร้างวงจรเชิงลำดับ



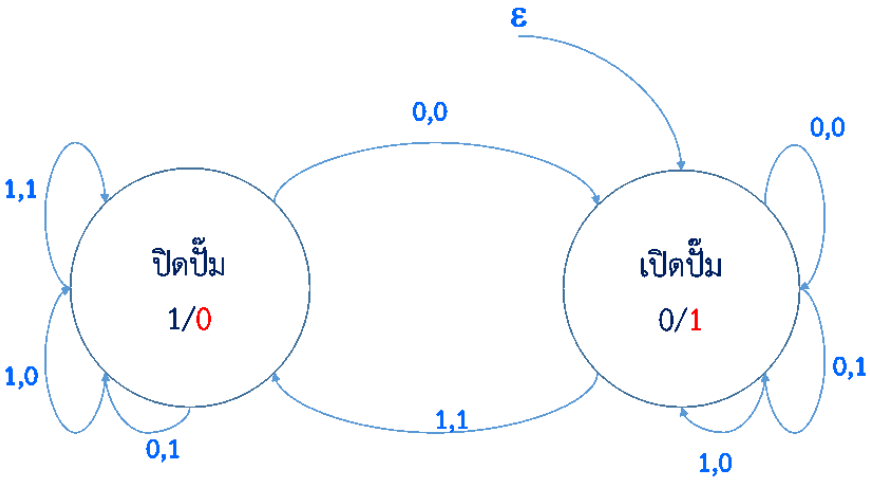
สร้างวงจรเชิงลำดับ :กำหนด State



สร้างวงจรเชิงลำดับ :กำหนด output แบบ Moore

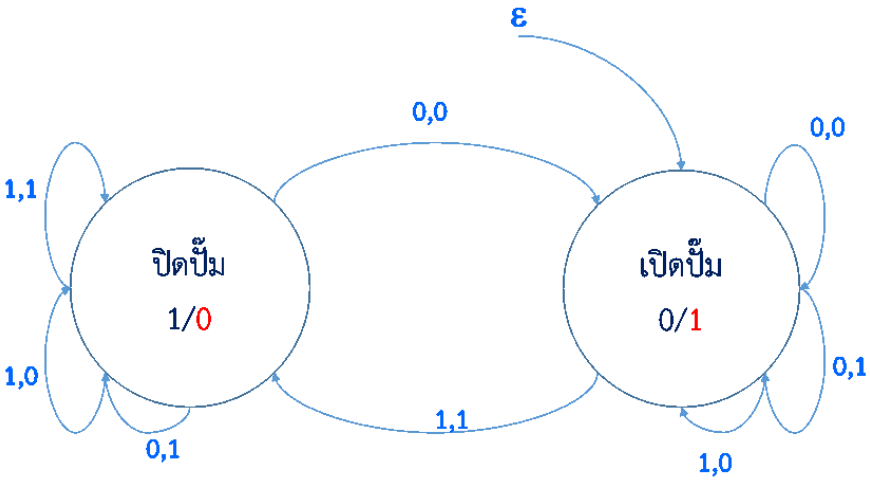


สร้างวงจรเชิงลำดับ :สร้างตารางสถานะ



Present State	Input a	Input b	Next State	Output
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

สร้างวงจรเชิงลำดับ :สร้างตารางสถานะ



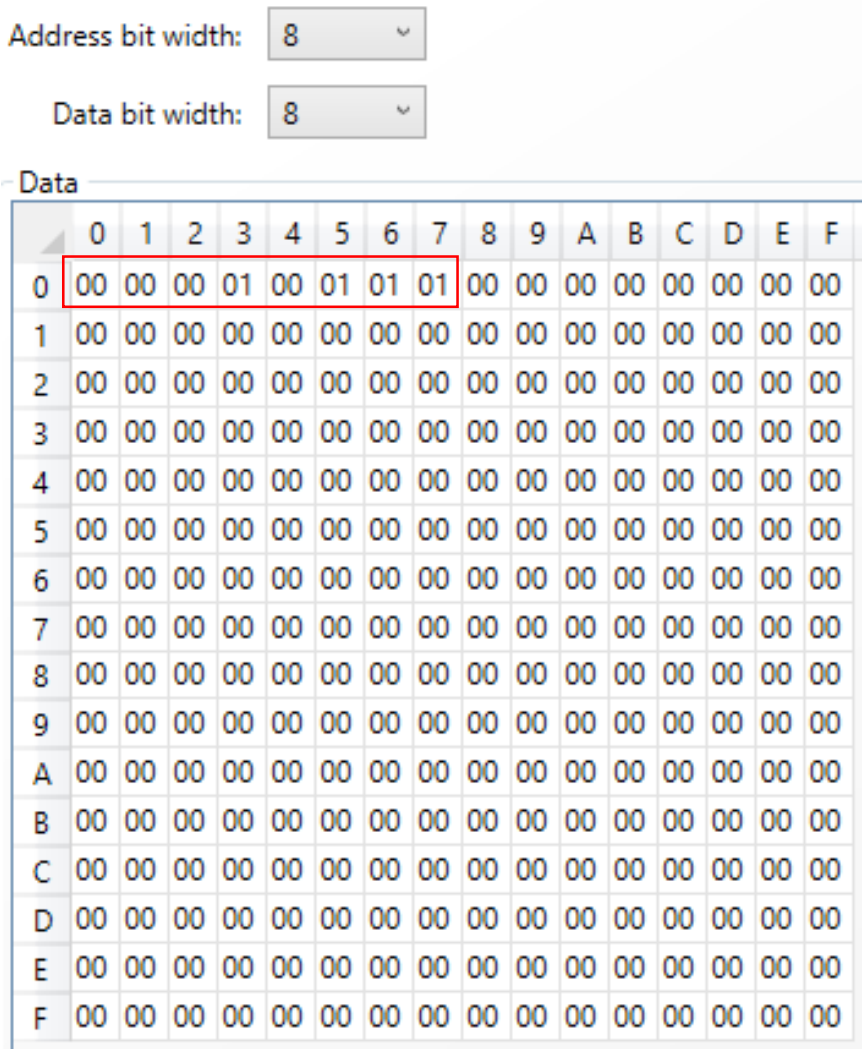
Present State	Input a	Input b	Next State	Output
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

วงจรนี้มีแค่ 2 สถานะ ตารางนี้จึงเป็นเลขฐาน 2 อยู่แล้ว

สร้างวงจรเชิงลำดับ :เปลี่ยนเป็นข้อมูลของรอม

Address Data

Present State	Input a	Input b	Next State	Output
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	00	00	01	00	01	01	01	00	00	00	00	00	00	00	00
1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

วงจรนี้มีแค่ 2 สถานะ ตารางนี้จึงเป็นเลขฐาน 2 อยู่แล้ว

สร้างวงจรเชิงลำดับ :ถอดรหัสคำตอบ

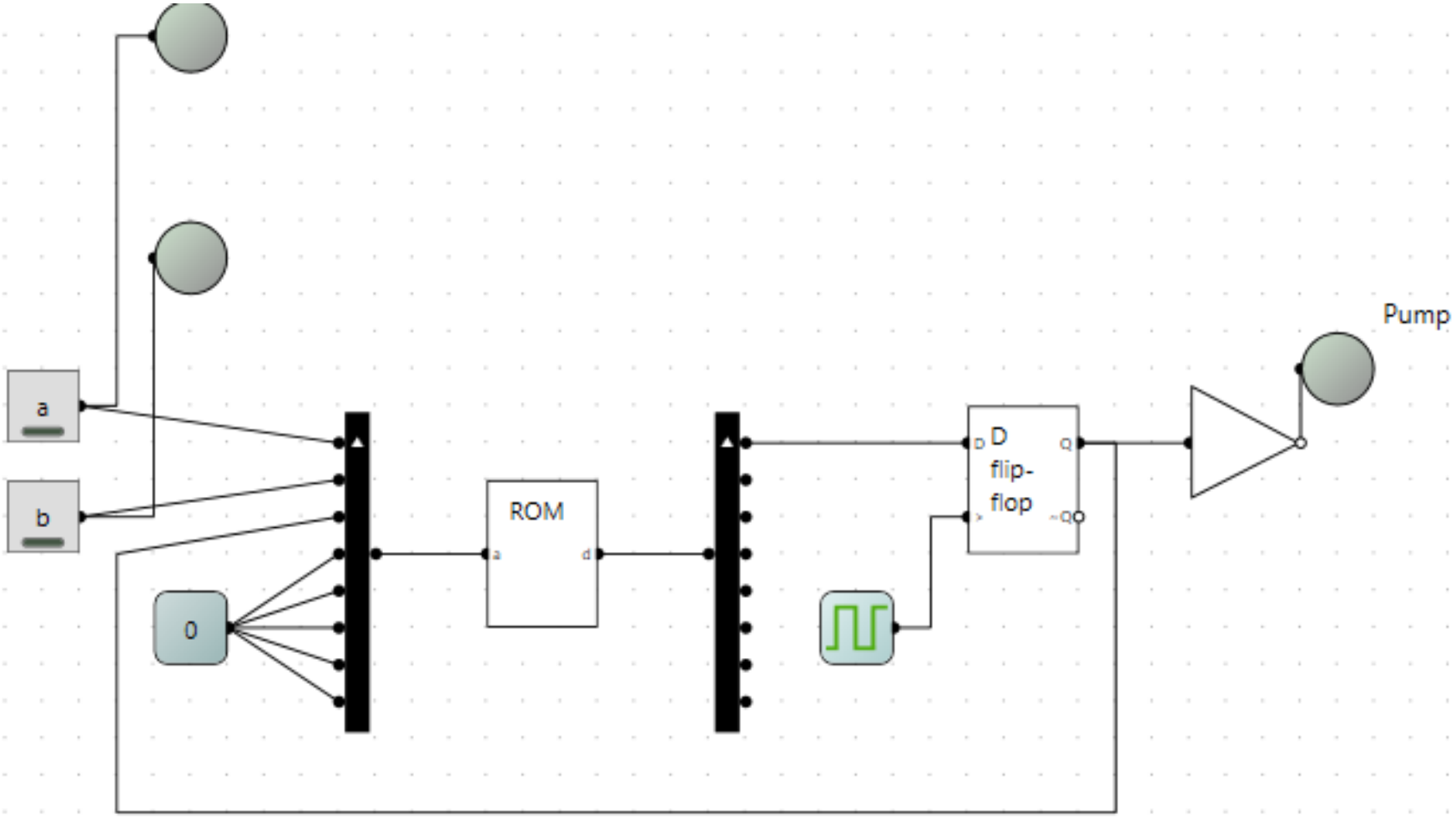
Address Data

Present State	Input a	Input b	Next State	Output
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

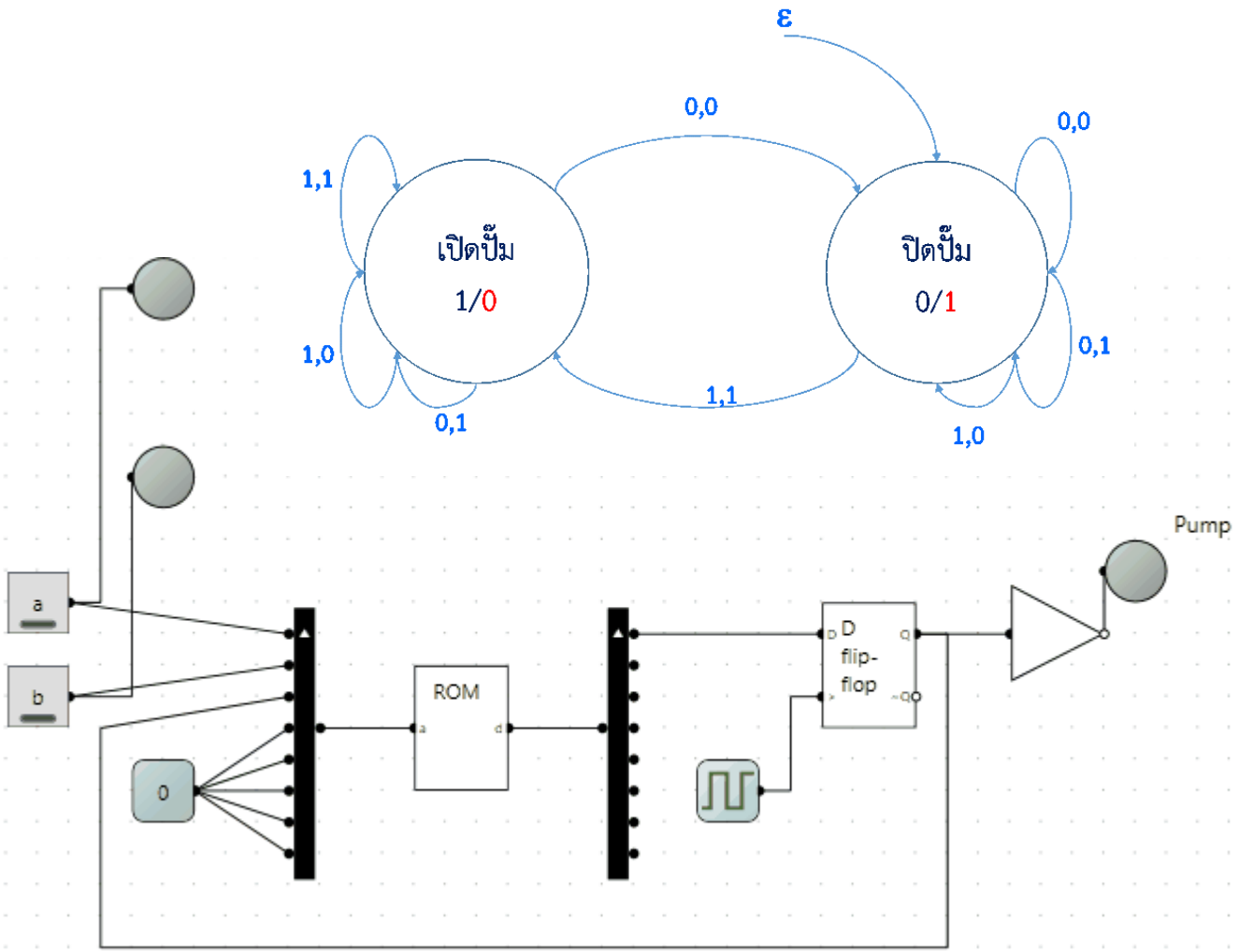
$$Output = \overline{Next\ State}$$

วงจรนี้มีแค่ 2 สถานะ ตารางนี้จึงเป็นเลขฐาน 2 อยู่แล้ว

สร้างวงจรเชิงลำดับ :สร้างวงจรแบบ Moore



สร้างวงจรเชิงลำดับ :สร้างวงจรแบบ Moore



สถานะของวงจร

ADD A,B

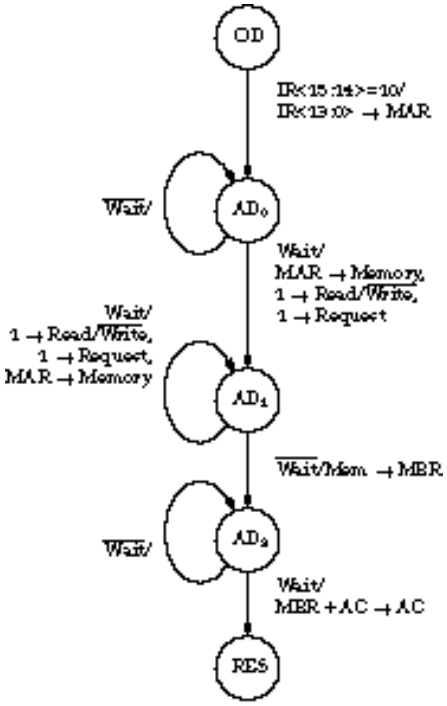


Figure 11.21 Add execution sequence.

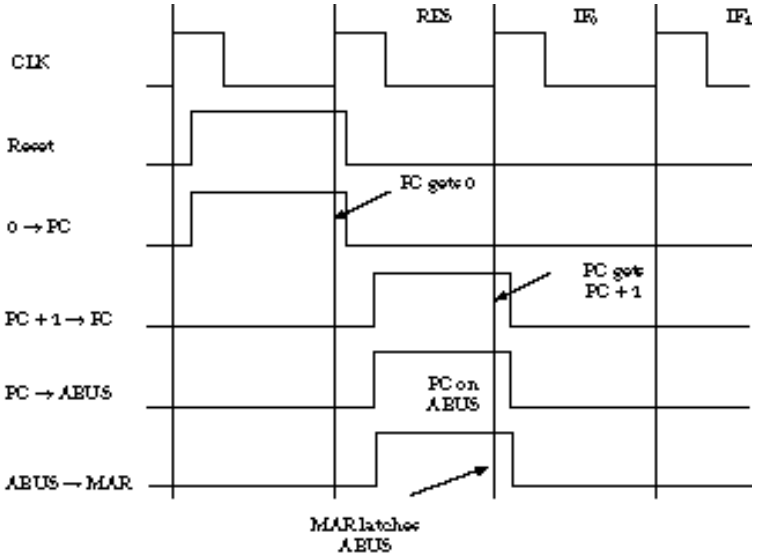


Figure 11.25 Timing of state changes and microoperations.

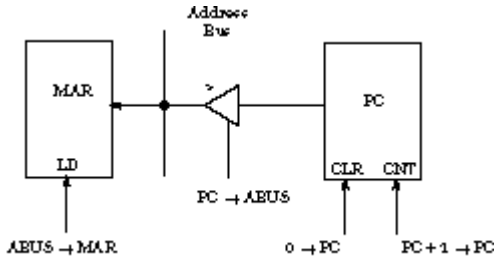


Figure 11.27 PC-to-MAR transfer with microoperations.

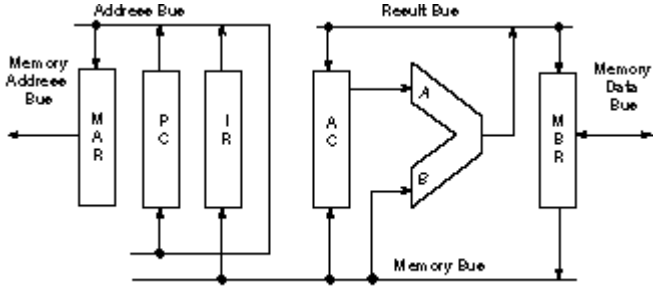


Figure 11.26 Three-bus processor datapath with dual-ported AC.

สรุป

เทอมนี้เรียนอะไรมาบ้างแล้ว

แอนะล็อก

- ความสัมพันธ์ของกระแส ความต้านทาน และแรงดัน
- กำลังงาน และพลังงาน
- การอ่านและเขียนแบบทางไฟฟ้า
- หลักเบื้องต้นและการใช้งานอุปกรณ์สารกึ่งตัวนำ

ดิจิทัล

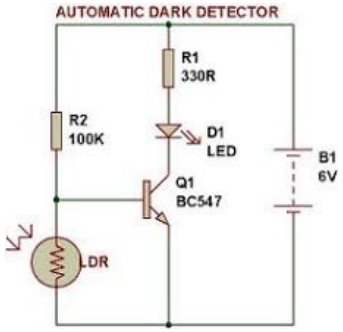
- ตัวกระทำทางตรรกะแบบต่างๆ
- การออกแบบวงจรเชิงจัดกลุ่ม
- การลดรูปวงจรดิจิทัล
- การออกแบบวงจรเชิงลำดับ

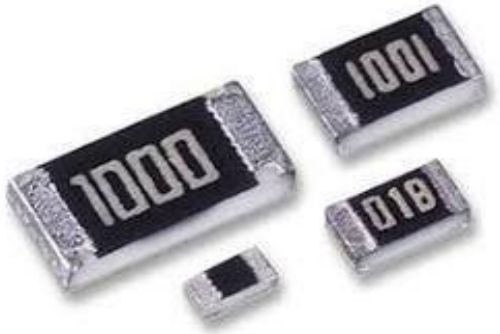
สรุป

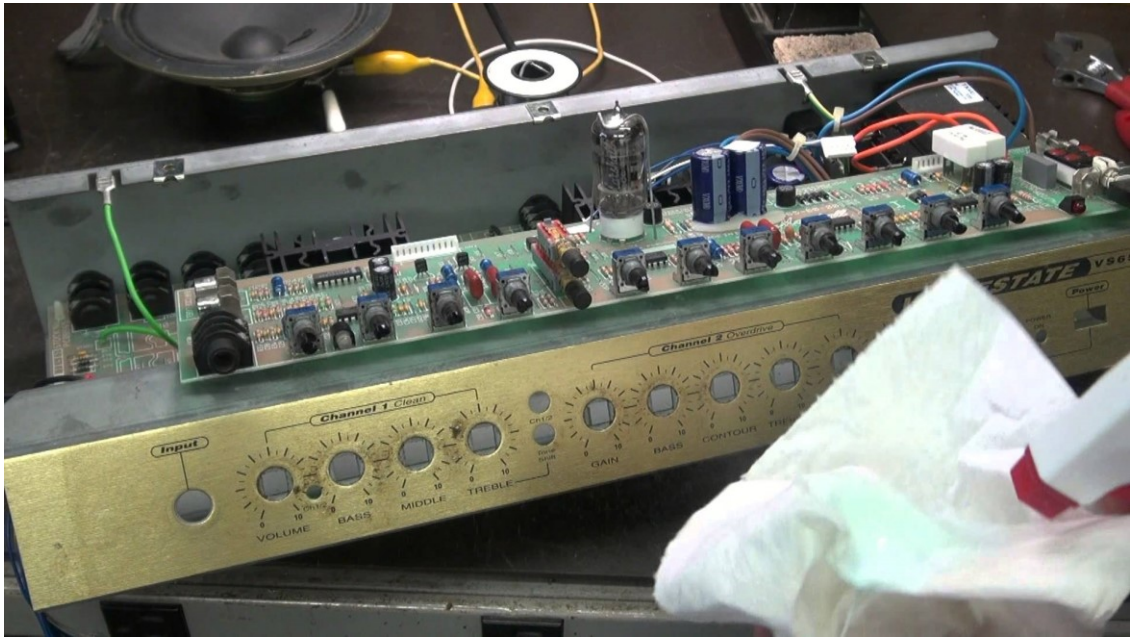
อดีตปัจจุบันและอนาคต

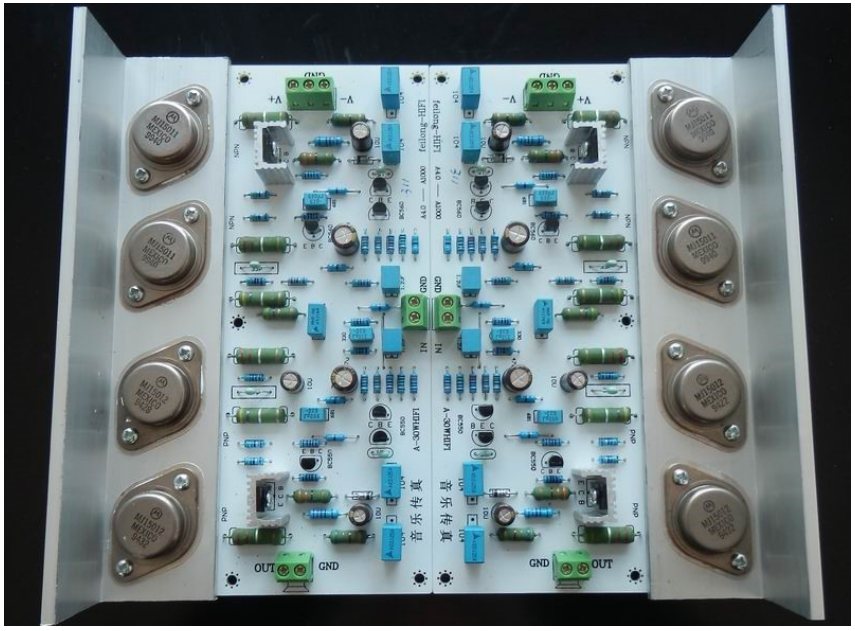
แอนะล็อก

อุปกรณ์มีขนาดเล็กลง ทำงานได้ดีขึ้น กินไฟน้อยลง แต่การใช้งานยังเหมือนเดิม







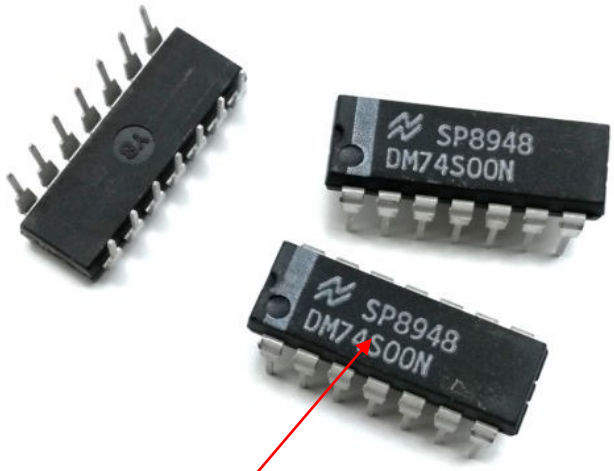


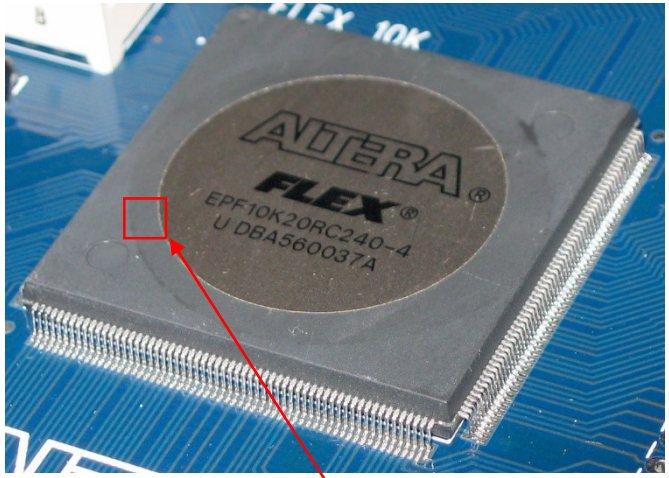
สรุป

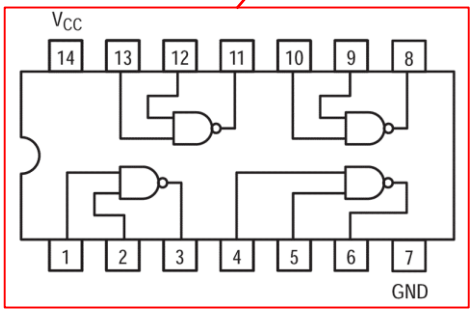
อดีตปัจจุบันและอนาคต

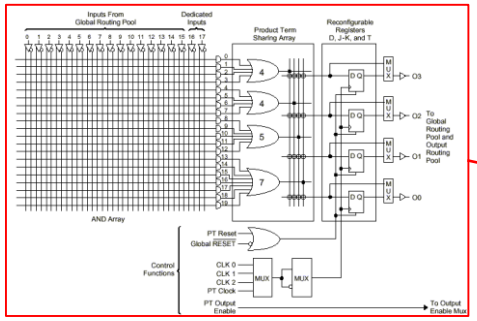
ดิจิทัล

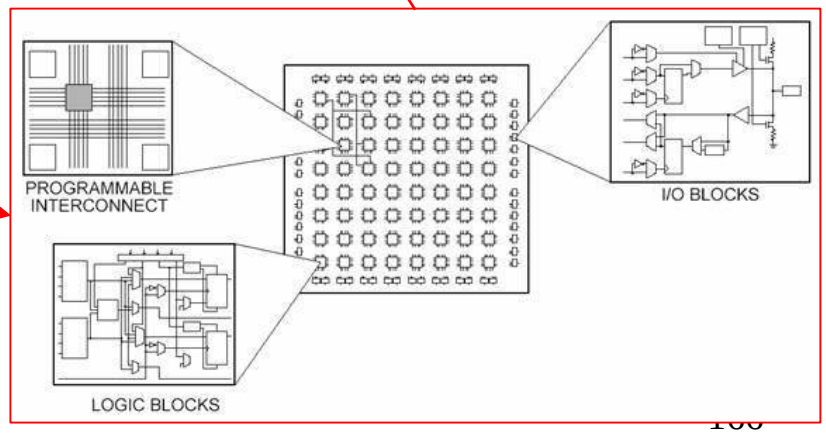
ปัจจุบันนี้เทคโนโลยีเปลี่ยนไปมาก IC 74xx ไม่มีขายแล้ว เพราะถูกแทนที่ด้วย FPGA









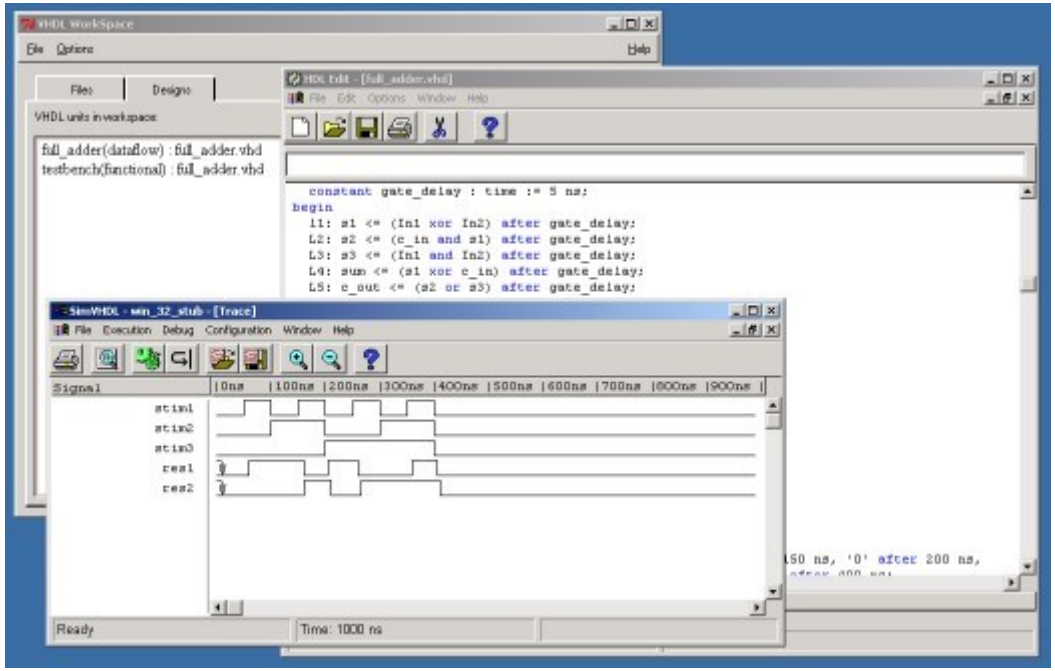


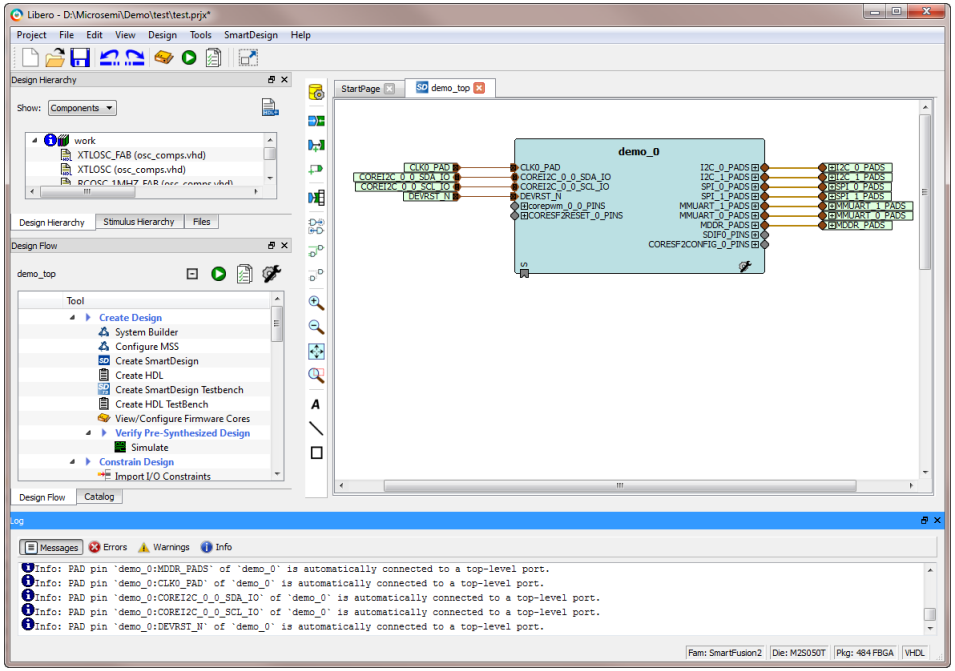
สรุป

อดีตปัจจุบันและอนาคต

ดิจิทัล

FPGA คือ ชิพสำเร็จรูปที่ภายในมี Logic gate บรรจุอยู่เป็นจำนวนมาก โดยเราสามารถจะโปรแกรมให้มันเชื่อมกันอย่างไรก็ได้





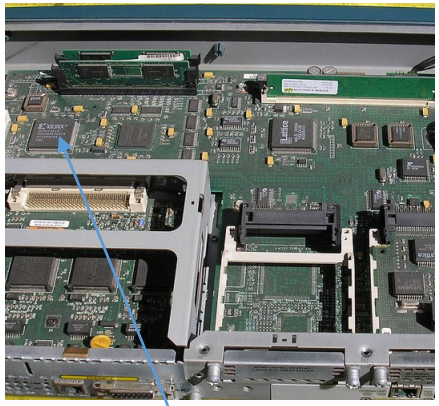
สรุป

ทำไมต้องเรียน วงจรดิจิทัล ไมโครคอนโทรเลอร์ก็ทำงานได้เหมือนกัน ?

งานที่ไม่ต้องการความแม่นยำ ความเร็ว ความเสถียร สามารถใช้ไมโครคอนโทรเลอร์สร้างได้

นอกจากนั้น ให้ใช้วงจรดิจิทัลสร้าง

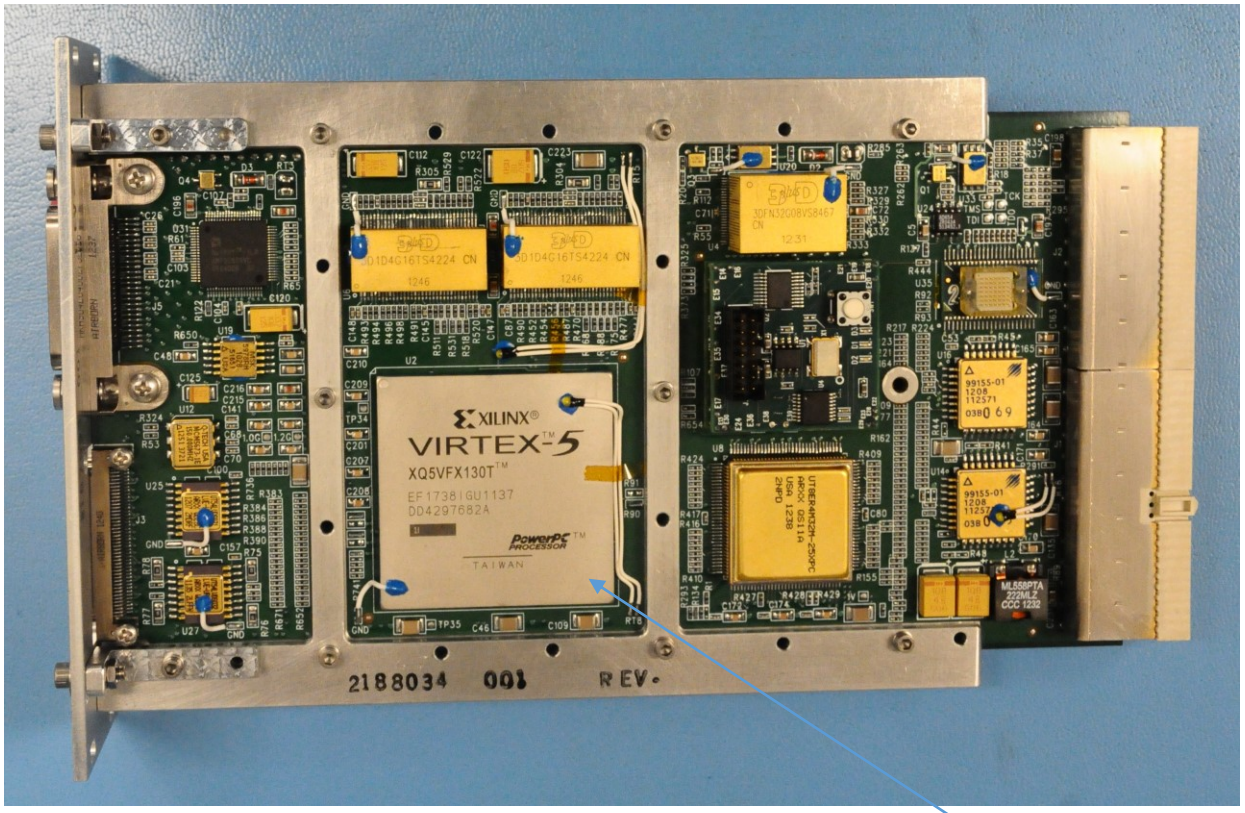
สรุป

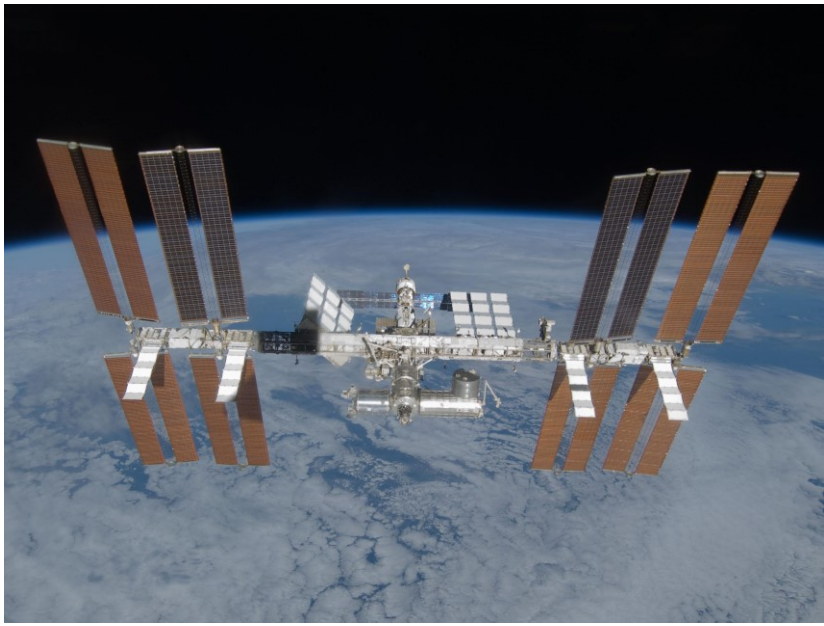


CPU

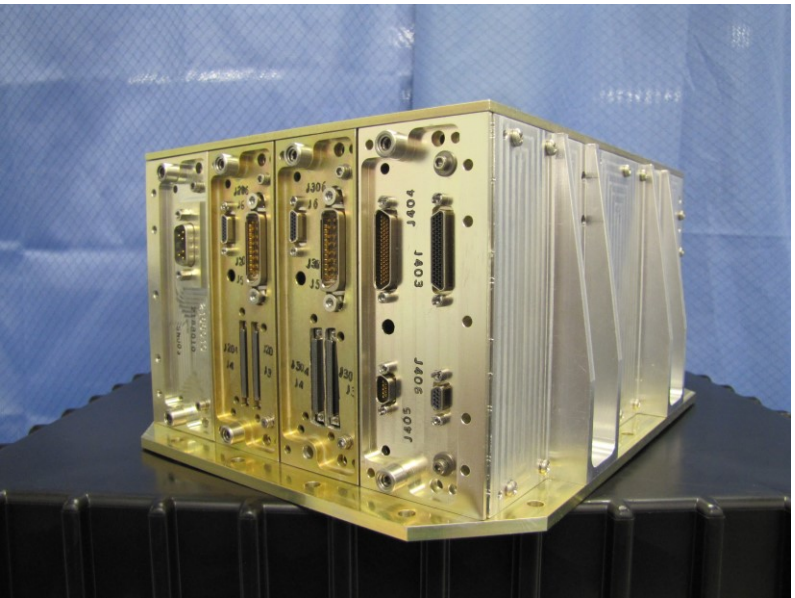
FPGA

สรุป



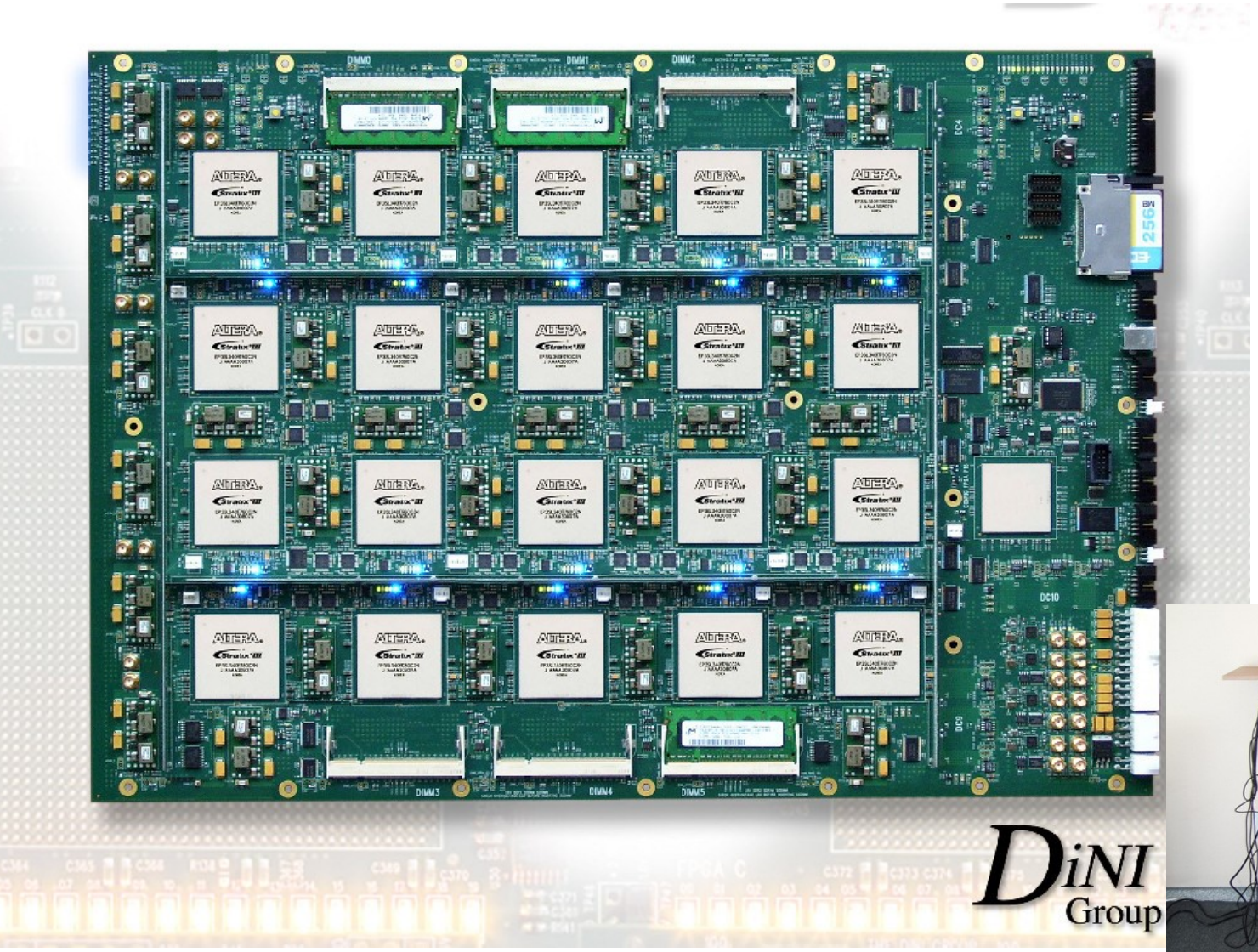


SpaceCube



FPGA

สรุป



สอบปลายภาค

วันที่ 27/2/2561

เวลา 12.00-15.00

นำเอกสารเข้าได้ 1 แผ่น A4

ให้นำเครื่องคิดเลขเข้าห้องสอบได้

ข้อสอบเก็บคะแนน 30%

มีทั้ง กา และ เติมคำ