

Digital System

デジタルシステム
Digital System

-1-

SCCS 211 Introduction to Digital System
(Logic & Computer Design)

สรุปโดย: ต้า

[Ta]
sec.1 5188018
CT#6

Aj. Chaivat & Chomtip

↳ 086-9721-750: chaivatptc@hotmail.com

Evaluation

Homework + Quiz	20%
Midterm	30%
Final	50%

* สรุปเนื้อเรื่องทั่วไป lecture ของ sec. 2 (Aj. Chaivat) ใช้เป็นราก!

!!!!!! สรุปคร่าวๆ เนื้อหาหลักของ midterm
ไฟฟ้าพลังงาน: 0 ถึง 1 ที่มีอยู่กัน
midterm จะมีต่อไปจะ: เรียนดูๆ ๆ ๆ
(concept หลัก ก็จะแล้วก็จะกัน)

ก้ามเขียนเชิง Programming
ทั้งในแล้ว software

... Digital System เป็นเครื่องที่ต้องการทำงานของ computer ซึ่งจะมีโครงสร้าง Hardware!

Computer

Add - 1

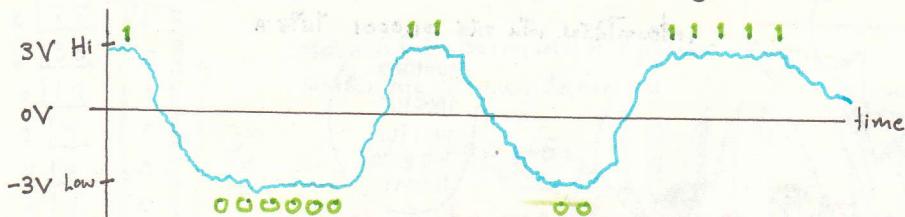
Sub - 0

Multi - 01

การสั่งงาน Hardware

จากโปรแกรม สั่ง เมื่อลง 0 ออก 1

computer รู้ว่า 0 กับ 1 ซึ่งหมายความว่า บวกกันได้ 0 Voltage Hi ไม่ Low



สัญญาณทางไฟฟ้าจะเท่ากับ 3V คือ Hi แทนด้วย 1 และ Low -3V คือ Low แทนด้วย 0

clock

Intel 2.0 GHz : 1 Hz = 1 cycle/sec

$$1 \text{ kHz} = 10^3$$

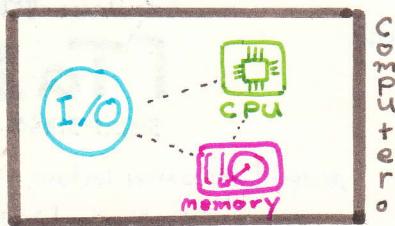
$$1 \text{ MHz} = 10^6$$

$$2.0 \text{ GHz} = 2 \times 10^9$$

→ แปลง

1 sec ผ่านต่อไปเรื่อยๆ 1 ครั้ง

ส่วนประกอบของ Computer



I/O to input, output ห้องรับข้อมูล ॥ จะ ॥ ส่งผล

- | | | | |
|----------------|---|-----------------|---|
| <u>input</u> - | Key Board
- mouse
- Joystick
- Camara/Scanner
- microphon
- Touch Screen | <u>output</u> - | Monitor
- printer
- Speaker
- Touch Screen |
|----------------|---|-----------------|---|

CPU ชื่อจาก Central Processing Unit

เขียนชื่อนี้มีความหมายว่า ใจกลางด้วย

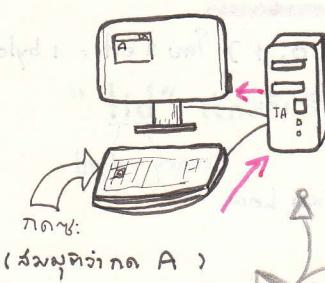
- CU (control unit)
- ALU (arithmetic logical unit)
- Register

Memory

ก็ต่อไปใช้เก็บข้อมูลนั้นแน่ๆ

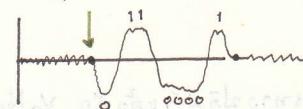
- | | |
|---------------|--------------|
| - Hard Disk | - RAM |
| - Flash Drive | - Drum |
| - Type | - Microfiche |

เวลา input จะเกิดอะไรขึ้นใน computer ?



สมมุติว่า เรา กด 'A' บน keyboard จะทำให้เกิดการ แกรบข้อมูล Voltage แรงบันดาลเพิ่ง: 7 จังหวะที่ 7 จังหวะนี้ CPU ต้องประมวลผล ต่อไป

กด A ตามนี้แน่น:



เครื่องจะได้รับ เมื่อรหัส 0110001 ไฟ A

1001000
1000101
1001100
1001100
1001111

หายใจอย่างน่ารัก
มุกหัวใจ

comment: เชิญโน๊ตบุ๊ค

ฟื้นชีพ บาร์บี... เป็นคนหาดูนูน...

รู้สึกว่า 55+
(ไม่ต้องเรียกต่างหาก
บุตร เด็กๆ ก็ได้)

Digital-TAN~

หัวติดผ้า
ไฟ NOT GATE

ลายชุดนอน
สีน้ำเงินดิจิตอล

หัวเชื่อม USB Port

(ต้องเชื่อมต่อแบบไหนเจล้อ ต่อสายยาน้ำทึบหรือเปล่าเนี่ย?)

Ans: ไม่แน่.. ไม่ต้องเชื่อมต่อถ้าใช้ USB 55+

Illustration: ~Mudmeen~

Base Number

基数

↳ 9 นับถือ computer รู้จักแค่ เลข [0] แล้ว [1], 2, 3, 4 พอกันชั้นไปรู้จักเลข... รู้จักกี่ 9 ใน computer นั่นคงต้อง จังหวะ เช่นฐาน 2 ก็ได้ก่อนเรา นับเลขไป

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

* เลขฐาน 2 ก็เหมือน เลขฐาน 10 นั่นแหละ: ถ้าก่อนนี้ก็เช่น บวกตัวยกหัวดังนี้ $[1] + [1] = [1]$ แล้ว $[1] + [1] + [1] = [1]$ เมื่อ หลักสัมม \max แล้ว ข้อกติดไปต่อไปเรื่อยๆ แต่ $[1] + [1] + [1] + [1] = [1]$ แล้วหลักสัมม \max set ก้อนไปเป็น $[0]$ นั่นเอง

Convert Number

① Decimal \rightarrow Binary

จะแปลงฐาน 10 เป็นฐาน 2 (หรือก็คือ n) ก็เอา n หาร เลขฐาน 10 ต้องหารไปเรื่อยๆ จนกว่า

$$\text{find } 77_{10} = ?_2$$

$$\begin{array}{r} 1 \\ 2 \longdiv{77} \\ 38 \\ \hline 19 \\ 2 \longdiv{19} \\ 9 \\ 2 \longdiv{9} \\ 4 \\ 2 \\ 1 \\ 0 \end{array}$$

หลังจากหารแล้ว 10 ได้ 2^0 ที่ได้หักนั้นดูมาเรียงกัน นั่นแหละ: ดังต่อไปนี้

$$\therefore 77_{10} = 1001101_2$$

หากฐานที่ต้อง convert 7_2

* ดำเนินการเปลี่ยนเป็นฐานอื่น ก็ต้อง ที่ฐานนั้น

② Binary \rightarrow Decimal

จะแปลงฐานอื่นๆ ดำเนินการ เปลี่ยนเป็นฐาน 10 ก็เอาแต่ละหลักไป ฐานนั้น ด้วยค่าประจำหลัก เช่น

$$\text{find } 111000111_2 = ?_{10}$$

$$\begin{aligned} &= (1 \times 2^8) + (1 \times 2^7) + (1 \times 2^6) + (0 \times 2^5) + (0 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + 1 \\ &= 256 + 128 + 64 + 4 + 2 + 1 \quad (\text{ต้องหา } n \text{ ให้ } 2^n \geq \text{หลัก }) \\ &= 455_{10} \end{aligned}$$

Note

▶ เวลาจานที่ต้องคำนวณ

ใช้ Decimal ฐาน 10

Binary ฐาน 2

Hexadecimal ฐาน 16

6 10

decimal point

▷ กี่ต่อหน่วย เราแบ่งเป็น เลขฐาน 8 ในส่วนของเร็ว int ॥ตัวต่อ 10 แบ่งเป็น จุดก่อสร้างชี้ทำไงนะ?

Ex. find $224.78125_{10} = ?_8$

เขียนตามแรกนี้ แยกส่วน "หน้าจุด" กับ "หลังจุด" 00 กามาก่อน

- ส่วนแรก 224 9 นิยมใช้ติดเนื่องเดียว กันที่ด้านขวา (10 8 หาร 10 แต่ไม่ได้เรื่องๆ) จะได้ 340_8 00 กามา
9 นิยมเอารีบไว้ก่อน แล้วไปติดส่วนหลังต่อ

ติดตามสูตรนี้

$$\begin{aligned} 0.x_1y_1z_1 \times n &= a.x_2y_2z_2 \\ 0.x_2y_2z_2 \times n &= b.x_3y_3z_3 \\ \vdots & \\ 0.x_7y_7z_7 \times n &= \end{aligned}$$

ans

เน้นให้เด็ดๆ แล้วจะช่วยลดเวลา: หาดูหัวอย่างกัน

- ติดส่วนหลังจุด ด้วย 0.78125

$$\begin{array}{r} 0.78125 \times 8 = 6.25 \\ 0.25 \times 8 = 2.0 \end{array}$$

$$\therefore = 0.62_8$$

ติดตามอยู่นี่

แทนดูกันแล้ว $10 \times 0.62_8$

(ลองยกตัวเป็น int ด้วย)

สรุป $224.78125_{10} = 340.62_8$

ผู้สอนรายฯ : เราต้องรีบติดแบบ int กับแบบ int ไปติด ส่วนหลังก็ต้องก่อมา ดูกันด้วยฐาน 8 ของมัน ใจ เวลา 0 กามา เก็บไว้ในร' ส่วน int ดี๊ด๊า ก็จะ \Rightarrow ดูกันด้วยฐาน 8 ของฐาน 8 แล้วมันจะเป็น 0 จึงก่อ หลังจุด จะเป็น 0

? กี่นี่มันจะมีมูลฐานมี 0 ดูกันเก็บไว้ ใจตัวนั้นซึ่งจุด มันก็ไม่เป็น 0 ซึ่งกี่แบบนั้น กันตันนั้น หารฐาน 10 แล้วมันไปลงตัวซักกี่ต่อ:

เช่น จะแปลง $0.3012_{10} \rightarrow ?_4$

$$\begin{aligned} 0.3012 \times 4 &= 1.2048 \\ 0.2048 \times 4 &= 0.8192 \\ 0.8192 \times 4 &= 3.2768 \\ 0.2768 \times 4 &= 1.1072 \\ 0.1072 \times 4 &= 0.4288 \\ 0.4288 \times 4 &= 1.7152 \\ \vdots & \end{aligned}$$

$$\therefore = 0.103101\dots$$

อ: ...งานเข้า แล้วเว้ย จะได้ 10 ว่า ควรบุคก์ ก็ต้องไปหาติดโดยดูที่ฐาน 10 หลังจุดมัน ณ 4 หลัก ก่อ 10 ฐานที่ 4 แปลงยกกลับ 4⁴ = 256 10 ที่นี่มันเป็น กว่า 256 ก็ต้องแล้ว 1031 มากกว่า 256 → บุคก์ก็ได้แล้ว!

$$\therefore 0.3012_{10} = 0.1031_4 \text{ ดี๊ด๊า ก่อแล้ว!}$$

▶ ส่วนจะแปลงฐาน 8 ที่ต้องการเป็นฐาน 10 ก็ต้องเนื้อเดิมอีก: และต่อไปนี้นักวิชาติดลบเพิ่มขึ้นเรื่อยๆ อ:

+ - × ÷

↪ ฐาน 10 บวก $+, -, \times, \div$ แล้ว ฐาน สอง ก็ทำได้ ... แต่ใน digital เราจะ เก็บเป็น binary (ฐาน 2)

plus

1011011
+ 1100011

1011100

॥புந்துமட 0+1 = 1 பு 1

→ $1+1=2$ នៃទីនាម 2 \rightarrow ព័ត៌មាន: 0 កែនវិគី 1 កែនបុគ្គលិកដែរ

→ 1+0 ເນັ້ນອີງຕົວອາໄຫດ ພັນຍິນ ສົ່ງກາຍເວັນ 1+1 ແມ່ນອັນກອບຕົວ

→ 1+1 ॥ กنمตัวหลัก เมื่อ 1+1+1 = 3 = 111 ก็ตัด 1 เก็บไว้, 1 เติมหน้า

សំណើជាប្រើប្រាស់ការ (នៅលម្អិតទីផ្សារ) និងសំណើជាប្រើប្រាស់ការ (នៅលម្អិតទីផ្សារ)

Subtraction (नियन्त्रण)

॥ମୁଦ୍ରଣକାରୀ ।-୦୭୫ ।

๗ ๑๐๑ ๐-๑ សិន្លឹកពេង តែងចាប់ផ្តើម ១ គោលការណ៍ (នីមួយការណ៍ផ្តើម ១) គោលការណ៍ ទៅការបាន (នីមួយការណ៍ ០) ផ្តើមនា ១ (ខែងខែងការងារថ្មីរបស់ ៧: វគ្គំ ២ ២-១ = ១.

๕. ก. ข้อมูลนี้เป็นข้อมูลที่ได้รับมาโดยทางการและถูกต้องตามกฎหมาย

* ก រ ុ ម ស ា ॥ ស ែ វ េ ង ॥ ក ព ត ន ុ ល ុ ប ់ ៗ ក ៩ ១០ ទ ិ រ ស ា ន ី រ ោ ដ ោ ៗ ១០
(ច ំ ស ា ជ ំ ស ា ទ ោ ក ក ប ្រ ុ ត ប ្រ ុ ត ប ្រ ុ ត !)

$$\text{เนื่อง } 10_2 - 1_2 \Rightarrow 2-1 = 1$$

multiply

$$\begin{array}{r}
 \textcircled{A} \rightarrow 11001 \\
 \textcircled{B} \rightarrow \underline{10011} \\
 & 11001 \\
 & 11001 \\
 & \underline{00000} \\
 & 00000 \\
 & 1001 \\
 \hline
 & 11011011
 \end{array}$$

107 Ⓐ เป็นตัวตั้ง ดูนะ ก้ม Ⓛ กี ลະ หົວ (เน้นอุนจุน 10)

↪ ③ ເນັ້ນ, 0:75 ອານຸ, ໄກສະຫວັນໄດ້

↳ ③ ແມ່ນວ່າ ອັນດີ ທັງໝົດ ນັ້ນແລ້ວ!

卷之三

15112100
15112100

સાધુ

ຄອນຫົວໜ້າ

نیازمند



divide

$$\begin{array}{r}
 100101 \\
 1001) 10101000 \\
 \underline{1001} \quad \downarrow \quad \downarrow \text{ซัก漉วะ}
 \end{array}$$

1100
 1001
 1100
 1001
 11

Ans 100101 10101000

หารก็หารแบบธรรมชาติ แต่ฐาน 2 มากที่สุดด้วย 1

(ไม่ต้องห่วงถูกต้อง: บันทึก 1 ให้ครบตัวเดียว = ต้องบันทึก)

ใช้ตัวลบๆ ตัด เส้นนหลักไว้เรื่อยๆ จนกว่าตัวหารจะเหลือ 0 บันทึกช่องตัวเลขเดียว 1 นำบันทึก 9 ต่อไป 9 ต่อไป

ASCII & BCD

ASCII code เป็นรหัสของตัวอักษร ตัวเลข และสัญลักษณ์ รวมไปจนถึงปุ่มทุกปุ่มบน keyboard เช่น Enter, Esc, Shift, Ctrl ฯลฯ ปุ่มพากน์ (รวมถึงสัญลักษณ์ที่ไม่สามารถเขียนบน keyboard เช่น '!', '♥') ทุกตัวจะมีค่า ASCII เป็นสองตัวเลข และไม่มีซ้ำกัน

'A' 01000001 = $64 + 1 = 65$

'a' 01000001 = 97

'B' 01000010 = $64 + 2 = 66$

'b' 01000010 = 98

⋮

'Z' 01011010 = $64 + 16 + 8 + 2 = 90$

'z' 01011010 = 122

'0' 00110000 = 48

'1' 00110001 = 49

'9' 00111001 = 57

ตัวอักษรตัวหนึ่ง กับ เล็ก ต่าง กัน กันตั้นหลักนี่

Note

ส่วนนี้ของ main frame
เราเรียกว่า EBCDIC

EBCDIC ไม่เป็นซักๆ



I want only BCD!!!



คิดเห็นอย่างไร? แล้วนะ?

อย่างเดียว... ใจร้ายมากจริงๆ ไม่ใช่หรือ

Binary Coded Decimal

นร. ที่เรียกว่า BCD code เป็น code รักประเจก แนวที่ 01 ไว้เก็บตัวเลขฐาน 10 โดยเฉพาะ ဂบ 4 บิต ต่อ เลขฐาน 10 1 หลัก

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

0-9 อยู่ในช่วงนี้กันแน่
กับ BCD ชุดเดียว

พื้นที่ต่อ 10 ขึ้นไปจนถึง 99
ต้องมี BCD 2 ชุดจะเก็บได้

นับ 10 นับหน่วย

!ระวัง BCD เก็บฐาน 10 ซึ่งมีตัว max ที่ 9 ไม่รวม 10: นั่นคือ BCD ก็ สูงสุดได้แค่ 1001 และต่อกัน ... ต่อจาก 1010, 1111 ล้วนผิดแล้ว

การบวก ลบ BCD

- ขั้นตอน basic ธรรมชาติ เช่น

$$\begin{array}{r}
 25 \\
 + 12 \\
 \hline
 37
 \end{array}
 \quad
 \begin{array}{r}
 \overset{2}{\textcolor{blue}{0010}} \quad \overset{5}{\textcolor{blue}{0101}} \\
 \textcolor{red}{1} \textcolor{red}{0001} \quad \textcolor{red}{2} \textcolor{red}{0010} \\
 \hline
 \textcolor{brown}{0011} \quad \textcolor{brown}{0111}
 \end{array}$$

3 7

$$\begin{array}{r}
 25 \\
 - 12 \\
 \hline
 13
 \end{array}
 \quad
 \begin{array}{r}
 \overset{2}{\textcolor{blue}{0010}} \quad \overset{5}{\textcolor{blue}{0101}} \\
 \textcolor{red}{1} \textcolor{red}{0001} \quad \textcolor{red}{2} \textcolor{red}{0010} \\
 \hline
 \textcolor{brown}{0001} \quad \textcolor{brown}{0011}
 \end{array}$$

1 3

กรณีบวก แล้ว บวกกันเลขโดยแยกตัวที่ละหลัก (ในกรณีที่ไม่มีตัวหลักสองชั้น)

Basic Identities สมบัติเบสิก: ของ Boolean จำเป็นมาก!

$$\begin{array}{lll} 1. X + 0 = X & 5. X + X = X & 9. \bar{\bar{X}} = X \\ 2. X + 1 = 1 & 6. X \cdot X = X & \\ 3. X \cdot 0 = 0 & 7. X + \bar{X} = 1 & \\ 4. X \cdot 1 = X & 8. X \cdot \bar{X} = 0 & \end{array}$$

commutative การสลับที่

$$10. X + Y = Y + X$$

$$11. X \cdot Y = Y \cdot X$$

Associative การgroup กลุ่ม

$$12. X + (Y + Z) = (X + Y) + Z$$

$$13. X(YZ) = (XY)Z$$

Distribute การกระจาย

$$14. X(Y + Z) = XY + XZ$$

$$15. X + YZ = (X + Y)(X + Z)$$

De Morgan กฎเดอมองร์แกน

$$16. \overline{X+Y} = \overline{X} \cdot \overline{Y}$$

$$17. \overline{XY} = \overline{X} + \overline{Y}$$

|| ลักษณะ

$$X \oplus Y = X\bar{Y} + \bar{X}Y$$

$$\overline{X \oplus Y} = \bar{X}\bar{Y} + XY$$

! ดูต่อไป $\overline{XY} \neq \bar{X}\bar{Y}$

Literals นามบัญญัติ ลักษณะที่ไม่ต้องคำนึงถึง เช่น $X, Y, A, B, \bar{X}, \bar{O}, 1, 0$ ไม่มี \times

Product term นามบัญญัติ ก่อนตัวแปรที่ AND กัน เช่น $XY, AB, A\bar{X}, A\bar{B}\bar{C}$

Minterm คือ product ที่ใช้พื้นแปรครบทุกตัว (product term with all input variable)

Ex. input $x, y, z \rightarrow XYZ, X\bar{Y}Z, X\bar{Y}\bar{Z}, \dots$ ~~XX~~, ~~XY~~ $\bar{Y}Z$ ไม่ OR ไม่ AND

Sum term นามบัญญัติ ก่อนตัวแปรที่ OR กัน เช่น $X+Y, A+B, A+\bar{X}, A+\bar{B}+C$

Maxterm คือ sum ที่ใช้พื้นแปรครบทุกตัว (sum term with all input variable)

Ex. input $x, y, z \rightarrow X+Y+Z, \bar{X}+\bar{Y}+Z, \dots$ ~~X+Y~~, ~~X+YZ~~ $\bar{X}Z$ ไม่ AND

number of maxterm, minterm $\rightarrow 2^n$

no.

all of minterm

all of Maxterm

$$0 \ 0 \ 0$$

$$m_0 : \bar{X}\bar{Y}\bar{Z}$$

$$M_0 : \bar{X} + \bar{Y} + \bar{Z}$$

$$0 \ 0 \ 1$$

$$m_1 : \bar{X}\bar{Y}Z$$

$$M_1 : \bar{X} + \bar{Y} + Z$$

$$0 \ 1 \ 0$$

$$m_2 : \bar{X}YZ$$

$$M_2 : \bar{X} + Y + \bar{Z}$$

$$0 \ 1 \ 1$$

$$m_3 : \bar{X}YZ$$

$$M_3 : \bar{X} + Y + Z$$

$$1 \ 0 \ 0$$

$$m_4 : X\bar{Y}\bar{Z}$$

$$M_4 : X + \bar{Y} + \bar{Z}$$

$$1 \ 0 \ 1$$

$$m_5 : X\bar{Y}Z$$

$$M_5 : X + \bar{Y} + Z$$

$$1 \ 1 \ 0$$

$$m_6 : XY\bar{Z}$$

$$M_6 : X + Y + \bar{Z}$$

$$1 \ 1 \ 1$$

$$m_7 : XYZ$$

$$M_7 : X + Y + Z$$



variable 3 ตัว จะมี minterm, Maxterm อย่างละ $2^3 = 8$ ตัว!

No Battery!!!

Sum of Product กี่บ่อกัน Product ห้าม OR กัน เช่น $XY + Y\bar{Z}$, $XY + Z$, $\frac{X}{x+0}$

Sum of minterm (A, B, C) กี่บ่อกัน minterm ห้าม OR กัน เช่น ABC , $A\bar{B}\bar{C} + A\bar{B}C$, $A\bar{B}C + A + BC$

Minimum Sum of Product sum of product กี่ถูกดัง จนเริ่กซี่สุดแล้ว เช่น XZ , $X + YZ$, $X + \cancel{XY}$ $\cancel{\text{บ่อกัน!}}$

$$= X + \bar{X}Y, \\ = (\bar{X} + X)(X + Y)$$

$$= X + Y$$

Product of Sum กี่บ่อกัน sum ห้าม AND กัน เช่น XY , $X(Y+Z)$, $(X+Y)(Y+Z)$

Product of Maxterm (x, y, z) กี่บ่อกัน Maxterm ห้าม AND กัน เช่น $X + Y + Z$, $(X+Y+Z)(X+\bar{Y}+\bar{Z})$

Maximum Product of Sum product of sum ห้าม max แล้ว เช่น $(X+Y)$, $(X+Y)(Y+Z)$

ตัวอย่างโจทย์ Boolean Algebra

● Prove this Boolean algebra \downarrow นิสัยจังหวะเมืองการพูดคุย "true"

$$1. \bar{X}\bar{Y} + \bar{X}Y + XY = \bar{X} + Y$$

$$\begin{aligned} \bar{X}(\bar{Y} + Y) + XY &= \bar{X} + Y \\ \bar{X} + XY &= \bar{X} + Y \\ (\bar{X} + X)(\bar{X} + Y) &= \bar{X} + Y \\ \bar{X} + Y &= \bar{X} + Y \end{aligned}$$

$$2. \bar{A}\bar{B} + \bar{B}\bar{C} + AB + \bar{B}C = 1$$

$$\begin{aligned} \bar{A}\bar{B} + AB + \bar{B}\bar{C} + \bar{B}C &= 1 \\ (\bar{A} + A)B + (\bar{C} + C)\bar{B} &= 1 \\ B + \bar{B} &= 1 \end{aligned}$$

$$3. Y + \bar{X}Z + X\bar{Y} = X + Y + Z$$

$$\begin{aligned} \text{add } \bar{X} \text{ ให้เข้าไป} \\ Y(X + \bar{X}) + \bar{X}Z + X\bar{Y} &= X + Y + Z \\ YX + Y\bar{X} + \bar{X}Z + X\bar{Y} &= X + Y + Z \\ \bar{X}Y + \bar{X}\bar{Y} + \bar{X}Z + \bar{X}Z &= X + Y + Z \quad \therefore X + (Y + Z) = X + Y + Z \\ X(Y + \bar{Y}) + \bar{X}(Y + Z) &= X + Y + Z \\ X + \bar{X}(Y + Z) &= X + Y + Z \\ (X + \bar{X})(X + (Y + Z)) &= X + Y + Z \end{aligned}$$

* กรณี: ใช้เข้า

$$X + \bar{X}(Y + Z)$$

ไม่ใช่

$$X + YZ = (X + Y)(X + Z)$$

▶ สรุป ถ้า $(Y+Z)$ นั้นแล้ว:

Given that $A \cdot B = 0$, $A + B = 1$ (for Problem 4)

$$4. (A+C)(\bar{A}+B)(B+C) = B \cdot C$$

$$\begin{aligned} &= ((A+C)\bar{A} + (A+C)B)(B+C) \\ &= (A\bar{A} + \bar{A}C + AB + CB)(B+C) \\ &= \bar{A}CB + ABB + CB\bar{B} + \bar{A}C\bar{B} + ABC + CCB \\ &= \bar{A}CB + AB + CB + \bar{A}C + ABC + CB \quad \text{ซึ่งกัน เซ็งนั้นต้อง OR (+)} \\ &= \bar{A}CB + CB + \bar{A}C + ABC \quad \text{หักกันได้ไม่เหลือตัวหนึ่ง} \\ &= CB(\bar{A} + 1) + \bar{A}C + ABC \\ &= CB + ABC + \bar{A}C \quad \text{สับซ้อนร่วมกัน} \\ &= BC(A + \bar{1}) + \bar{A}C \end{aligned}$$

$$\begin{aligned} &= \bar{A}C + BC \\ &= \bar{A}C(A + B) + BC \quad \text{add } \bar{A} \text{ ให้เข้าไปแล้ว} \\ &= \bar{A}CA + \bar{A}BC + BC \quad \text{กรณี: ใช้เข้า} \\ &= BC(\bar{A} + 1) \quad \text{ซึ่งกัน} \\ &= BC \end{aligned}$$

$$A + B \text{ เป็น } 1$$

กรณี: ใช้เข้า

ซึ่งกัน

หักกันได้ไม่เหลือตัวหนึ่ง

จะ!

$$\therefore BC = BC$$

1D consensus Theorem

$$\underline{\underline{XY}} + \underline{\underline{\bar{X}Z}} + \underline{\underline{YZ}} = XY + \bar{X}Z \quad (\text{term } YZ \text{ is terminated})$$

proof: $XY + \bar{X}Z + YZ = XY + \bar{X}Z + YZ (X + \bar{X})$ add \rightarrow เพิ่ม $X + \bar{X}$ ทำให้ มีค่า = 1
 $= XY + \bar{X}Z + XYZ + \bar{X}YZ$
 $= XY(1+Z) + \bar{X}Z(1+Y)$
 $= XY + \bar{X}Z$

1D complementing function

func. ที่ไม่ complement (not) \rightarrow use "De Morgan's"

Ex. $F = X(\bar{Y}\bar{Z} + YZ)$

$$\begin{aligned} \bar{F} &= \overline{X(\bar{Y}\bar{Z} + YZ)} \\ &= \bar{X} + \overline{(\bar{Y}\bar{Z} + YZ)} \\ &= \bar{X} + (\bar{Y}\bar{Z})(\bar{Y}Z) \\ &= \bar{X} + (Y + Z)(\bar{Y} + \bar{Z}) \end{aligned}$$

สำคัญ

1. ใช้ not ให้ถูกต้อง

2. วงเล็บ อยู่ที่เดิม (ตัวที่อยู่ในวงเล็บก็ต้องใช้วงเล็บ)

3. เปลี่ยน OR เป็น AND
AND เป็น OR

จบ.

- Reduce this algebra to the indicated number of literals:

ลดจำนวนตัวต่อไปให้เหลือ 3 literals (กี่ตัวก็ได้)

1. $X + Y(Z + \bar{X}Z)$ to 2 literals

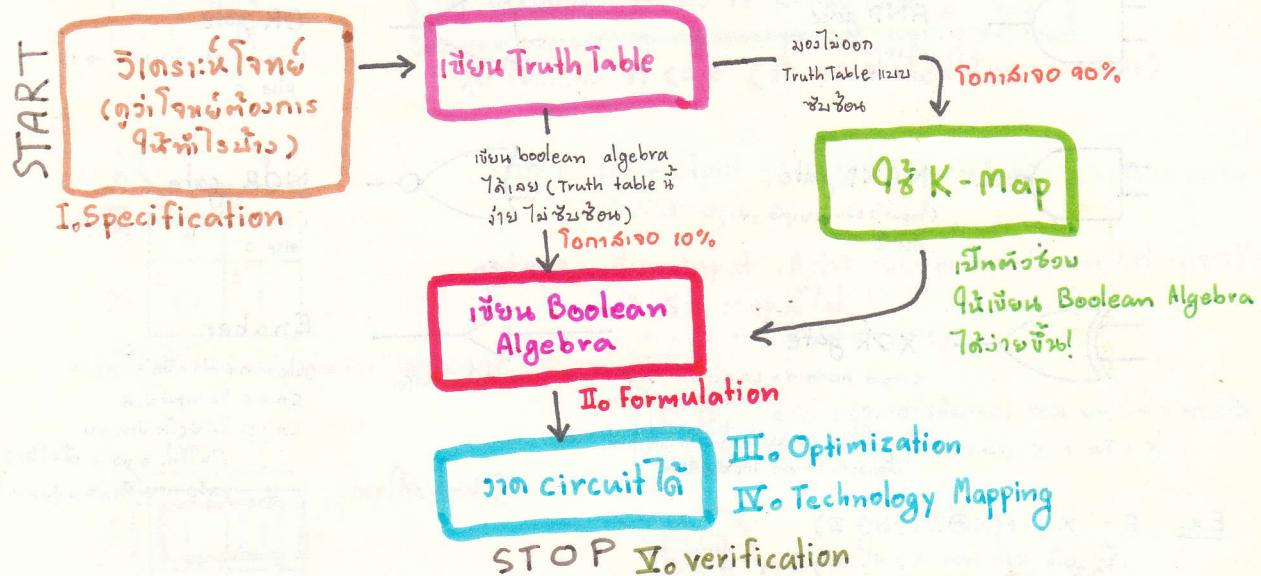
$$\begin{aligned} &= X + Y(Z + \bar{X}Z) \quad \leftarrow \text{De Morgan} \\ &= X + Y(Z + \bar{X}) \quad \leftarrow \text{กรณี 1 แบบแล้วตัด } Z + \bar{Z} \text{ ออก} \\ &= X + YZ + \bar{X}Y \quad \text{add} \\ &= X(\bar{Y} + Y) + YZ + \bar{X}Y \\ &= XY + X\bar{Y} + YZ + \bar{X}Y \\ &= Y + X\bar{Y} + YZ \\ &= Y + \bar{X}Y \\ &= (X + Y)(Y + \bar{Y}) \\ &= X + Y \end{aligned}$$

2. $\bar{W}X(\bar{Z} + \bar{Y}Z) + X(W + \bar{W}YZ)$ to 1 literal

$$\begin{aligned} &= \bar{W}X\bar{Z} + \bar{W}XYZ + XW + \bar{W}XYZ \\ &= \bar{W}X(\bar{Z} + \bar{Y}Z + YZ) + XW \quad \leftarrow \text{สูตร DOR ใช้: ตัด } \bar{W}XYZ \text{ ออก} \\ &= \bar{W}X(\bar{Z} + Z) + XW \\ &= \bar{W}X + XW \\ &= X(W + \bar{W}) \\ &= X \end{aligned}$$

Truth Table K-Map Circuit

↳ Circuit ສົ່ງ ວັດໄຟນີ້ ພຶບໆ ຖະແຈກຮັບອະນຸມັດ ຈາກ computer ໂດຍຈະຕັ້ງ circuit ໄສ ຕົວ...



Ex. ພັນຍຸ 2 ພັນ x ກມ ຍ ຕ້າ ກດ ພຣອມກັນທີ 2 ພັນ ນລວກໄປ ຈະຕິດ ນອກນັ້ນນລວກໄປດັ່ງ

ກົດມອບໃຈນີ້ ທັງສອງ ຖຣຸතາບ ..

		F	กด F แทน output
X	Y	F	
0	0	0	→ ไม่กดทั้ง X และ Y output เป็น 0
0	1	0	→ กด Y ตัวเดียว output ก็จะเป็น 0
1	0	0	→ กด X ตัวเดียว ก็ไม่เกิดการซ่อนกัน
1	1	1	→ กด X กับ Y 一起 output เป็น 1 (ตามโจทย์ส่วน 1)

กรดทั้งหมด

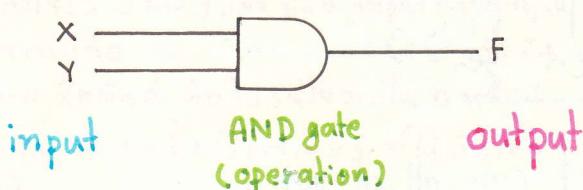
(กຳລົງ variable ນັ້ນ
ຈະແມ່ 2ⁿ ກວດຫຼື)

▶ ก็นີ້ ລົດວ່າ Truth Table ຫັນນີ້ ສັນດວຍເງິນ Truth Table ວິວ AND (ນຳ ၁) ເຮັດວຽກ

$$F = X \cdot Y$$

សិរីបាននៅក្បែង !
តាមតំណែន K-map

Step 10 ไฟก็ต้องสามารถที่จะไปต่อเป็น circuit



The diagram shows the standard logic symbol for an AND gate, which consists of a rectangle with a diagonal line from the top-left corner to the bottom-right corner, representing the logical operation where both inputs must be high for the output to be high.

ເນື້ອ gate ກໍາມັດຖະວົນຕີ ນອງ AND
ດັ່ງຕີໄປນີ້: ເກມາເນື້ອ | ກາເສັ້ນ
output ດັ່ງຈະເນື້ອ |

K-Map

ผังมาชาก "Karnaugh map"

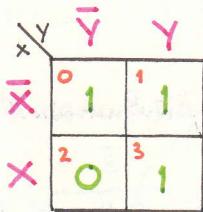
→ เมื่อถูกตั้งค่า ตารางที่ได้ไว้ แบ่ง成 Truth Table เช่น สมการ 9 บิตนี้ที่ บ่งจาก Truth Table 7 บิตนั้น ใช่...

X	Y	F
0	0	1
1	0	1
2	1	0
3	1	1

แบบนี้ดี: ก็ต้องบอกว่า สมการส่วนใด ก็ใช้ K-map ช่วย

มี variable n ตัว ใช้ K-map iff 2^n ช่อง

* กรณี $n \leq 4$ (ถ้า $n > 4$ ต้องใช้แบบ 3 มิติ)

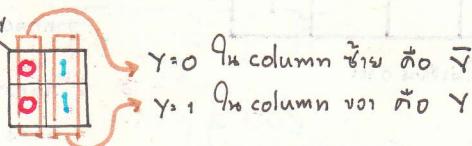
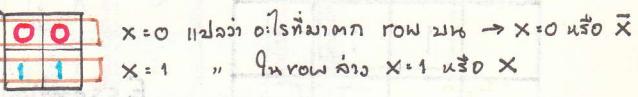
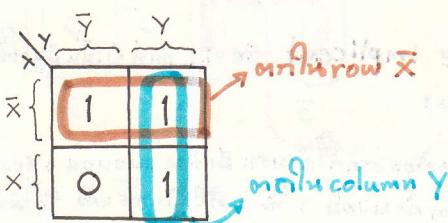


กล่องๆ นึงใน Kmap เรียกว่า "implicant"

step1 เอก output (F) จาก TruthTable มากี่ตัวตาม
ลำดับ $0, 1, 2, \dots, 2^n - 1$

step2 เก็บ input ที่เก็บ row และ column (ถ้าเก็บช่อง
แล้ว ต้อง step นี้ๆ)

หมายเหตุ: x, y จาก Truth Table มากี่ตัว



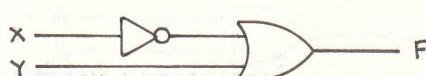
step3 วัดกี่ตัว ช่องของ 1 ให้ครบ
ทุกตัว (0 ไม่ต้องสนใจ)

โดยการวัดกี่ตัว ช่องของ 1 ต้องเขียนไปตาม
กฎ selection rule!

ให้ต้องอย่าง ว่าได้ 2 กล่องไว้ได้มาตั้ง \bar{x} กับ y
ให้เข้มทุกตัวที่ได้มาต้อง OR

$$\therefore F = \bar{x} + y \quad (\text{หมายความเรียงต่อ} \\ \text{ที่สุด not})$$

ถ้าเดาไปใช้บน circuit คงได้...



- !* ของจาก 0 กับ 1 แล้ว ซึ่งมี input สองตัว: output
- อีกแบบดีอี X (อ่านว่า Don't Care) ซึ่งมัน
ก็สามารถบูล์ฟันต์: ต้องสนใจว่า เป็น 0 หรือ 1
ก็ได้ ซึ่งบูล์ฟันต์แล้ว แต่ เคบว่า จะให้บูล์ฟัน
อะไร

หมายเหตุ: กรณีที่มี

Selection Rule

1. Select the 2^n box.

เลือก瓜ดล่อง 1 โอดเป็นกล่อง
มีร่องร่วม แนว 2^n (รวมในนี้ได้)
(ให้ในสูตรที่ได้ ที่ต้องมาก่อน)

2. Minimize the overlap
among prime implicant
as much as possible.

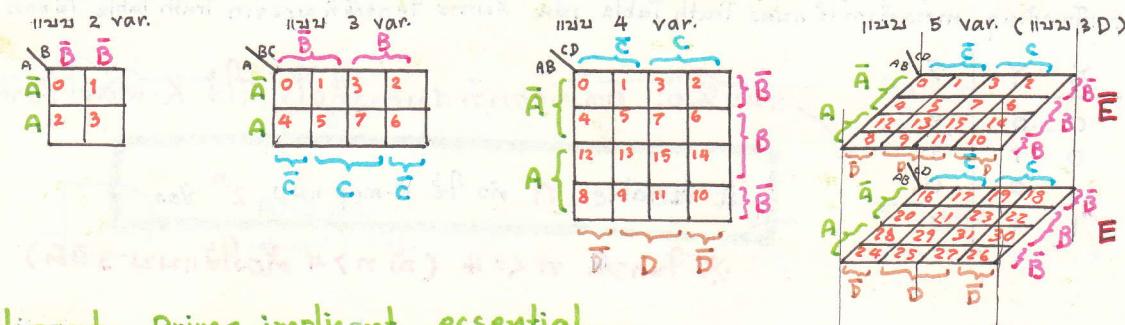
หลักการเลือก การหักกันของกล่อง
ที่ให้มา กันทุกตัว (หากที่จำเป็น)

3. In particular, in the final
solution make sure that
each prime implicant selected
includes at least one minterm,
not included in any other
prime implicant selection.

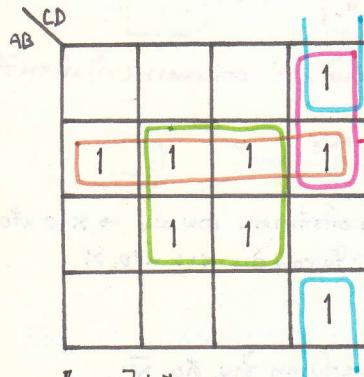
อย่างนี้แล้ว ก็ต้องที่ ถูก ก่อตั้ง หัน
แนวตั้ง ... อย่างต่อๆ ต้องเปลี่ยน 1 ช่องที่
ไม่ใช่ 0 ให้ หันกลับ



K-map သူရုပ်ပောင်နဲ့ကု တစ်ခုတွင် input variable



implicant, Prime implicant, essential.



! ဆုံးဂျာများ အောင် 0 မှာ:

K-Map ဆုံးဂျာများ အောင်

ပိုက်လော့ ပို မှာ ↔ လော့
အဲ: အောင် ↔ ဟာ မေးမြန်ကော်
မှာ ခံပါ။

Implicant ကိုကလ်ဘဲလော့လော့ ပိုက် 1 ကဲ့တော့ ဖော်စိန်သွေးပြား 1 ဝယ်
ပိုမ်း 8 ထော်ချုပ် 8 Implicant

Prime Implicant ကိုကလ် 2ⁿ ဘဲလော့လော့ (ဝေါ်ပိုက် ပိုစွဲတော့ ဘာဘာဘာဘာ)
မှုတော်ဝယ် 4 ကဲ့တော့ ပိုမ်း 4 PI.

Essential Prime Implicant ထဲ 0 PI ဘေးရဲ့ ရဲ့ မော်မှုများ အောင်
ဘာကလ် ဆိုမြတ် 1

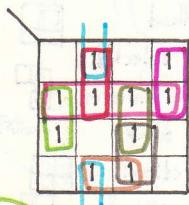
→ ကဲ့လ် နဲ့ မျိုးစား အောင် ဘာဂ်ဒါ ဖော်: မှာကလ် မော်မှု 4 ခံ ကဲ့
ကဲ့လ် တော်များ လို့မြတ် 1 ပို့ 2 တော် ဟော လော့ ပိုမ်း အောင်
ကဲ့လ် နဲ့ မျိုးစား အောင် \therefore မျိုးစား ပိုမ်း 3 ပေါ်

Note! မြတ်မှု ဒါ တော် အောင် Eng.

- **Implicant** one box in the K-map
- **Prime Implicant** the 2^n box that biggest and not included by other box
- **Essential PI.** Prime Implicant that are necessary to cover all 1 in K-map

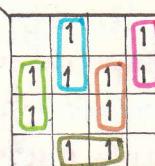
သေခာတော်များ အောင် (ဝေါ်ပိုက် ဆိုမြတ်) သေခာတော်များ အောင် Essential မှုတော်များ

Ex.



ကိုရေးမယ် PI စဲ မျိုးစား မာကမယ်
အောင် မျိုးစား

$$F = \bar{A}B + B\bar{C}\bar{D} + \bar{B}C\bar{D} + \bar{A}\bar{C}D + BCD + \bar{A}C\bar{D} + A\bar{B}D + ACD$$



သေခာတော်များ အောင် Essential မှုတော်များ
အောင် မျိုးစား

$$F = B\bar{C}\bar{D} + \bar{A}\bar{C}D + BCD + \bar{A}C\bar{D} + A\bar{B}D + ACD$$

[💡] သေခာတော်များ အောင် ကဲ့လ် ပိုက် ပိုစွဲ အောင် အောင် Essential ဘဲလော့

Ex. วาต circuit จาก Truth Table นี้

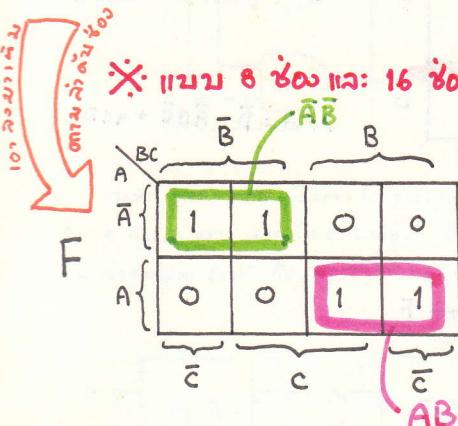
A	B	C	F	G
0	0	0	1	1
1	0	0	1	0
2	0	1	0	1
3	0	1	0	1
4	1	0	0	1
5	1	0	0	1
6	1	1	1	1
7	1	1	1	1

มี output 2 ตัวก็ไม่ใช่ 0: ใจนรก ก็แยกกันกิดดู

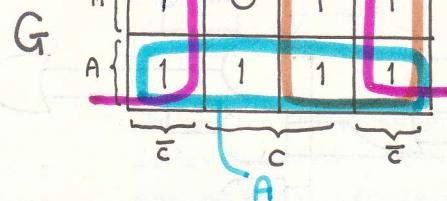
	BC	00	01	11	10
A	0	0	1	3	2
O	0	1	5	7	6
1	4	5	7	6	

หาก 01 ตามด้วย 11
แล้วต้องเป็น 10 เป็น gray code (เปลี่ยนทีละ bit ไป 01 หรือ 10 มันเปลี่ยนทีละ 2 bit อย่างที่ทำ) ทำงานช้าๆ วะ

* บน 8 ชุด และ 16 ชุด เวลาเรียงเลขซึ่ง 2 ชุดสุดท้ายจะสลับที่กัน!



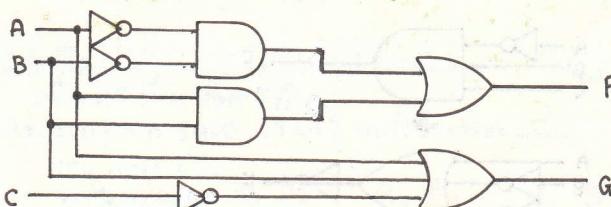
$$\therefore F = \bar{A}\bar{B} + AB$$



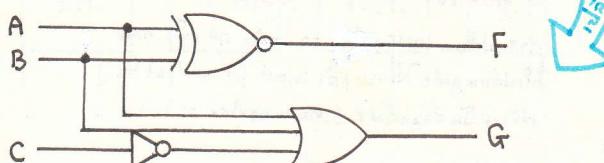
$$\therefore G = A + B + \bar{C}$$



สังเกตว่า F ไม่มี C เกี่ยว เช่น หมายความว่า ไม่ว่าจะ กดปุ่ม C หรือไม่ ก็จะสั่นตัว กด F เคย สั่นขึ้น กับ A และ B ด้วยเดียวกัน ▷ ผู้ใดก็ตามที่ติดภัยแก้ไขในสูตร ก็เลิกสนใจนี่



แต่เพื่อปรับง่าย gate $\bar{A}\bar{B} + AB = A \oplus B$



Ex. จง circuit ของ output F, G

$$\bar{F}(A, B, C) = \sum m(1, 3, 6) \rightarrow F(A, B, C) = \sum m(0, 2, 4, 5, 7)$$

$$\bar{G}(A, B, C, D) = \prod M(0, 2, 4, 5, 8, 10, 11, 15) \rightarrow G(A, B, C, D) = \sum m(0, 2, 4, 5, 8, 10, 11, 15)$$

	BC	\bar{B}	B	$\bar{A}C$
A	1	0	3	0
A	4	1	1	1
		\bar{C}	C	\bar{C}
		$\bar{A}\bar{B}$	AC	$\bar{A}C$

$$\therefore F = A\bar{B} + AC + \bar{A}\bar{C}$$

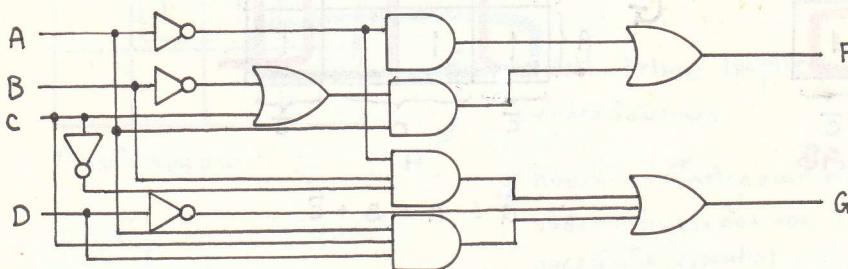
$$= A(\bar{B} + C) + \bar{A}\bar{C}$$

	CD	\bar{C}	C	\bar{B}
AB	0	1	0	2
AB	4	1	1	0
	12	0	0	6
	13	0	0	7
	15	1	1	0
	14	1	1	1
	9	0	0	1
	1	0	1	1
	10	0	1	0
	11	1	1	0
	12	1	1	1
	13	0	0	0
	14	0	0	1
	15	1	1	0
	16	1	1	1
	17	0	0	0
	18	0	0	1
	19	1	1	1
	20	1	1	0
	21	0	0	0
	22	0	0	1
	23	1	1	1
	24	1	1	0
	25	0	0	1
	26	0	0	0
	27	1	1	0
	28	1	1	1
	29	0	0	0
	30	0	0	1
	31	1	1	1

Σm = sum of minterm

$\prod M$ = product of maxterm

$$\therefore G = \bar{D} + \bar{A}B\bar{C} + ACD$$



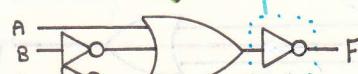
การเปลี่ยน gate

► inverter



ก้อนก็ได้แล้ว invertor สามารถ
เปลี่ยนไปใช้ NAND หรือ NOR
แทนได้ (input ต้องมี 2 เส้น)

► AND - OR



ถ้ามีหัว OR น้ำ OR หรือ
AND ควบรวมกัน
เป็น NOR หรือ
NAND ก็จะ

การเปลี่ยน AND เป็น OR หรือ OR เป็น AND
ก็จะเปลี่ยน gate ไปเลย แล้ว input และ output ก็จะ
เปลี่ยนไปตามส่วนกลับกัน (ตัว invertor 除外)

ADDER

ສືບຕົກນອກອີ່ມ ແລ້ວວ່າ add ກີເນີນ ວຈຣາຕ້ອໄຮ ນິກ ເພນີ້

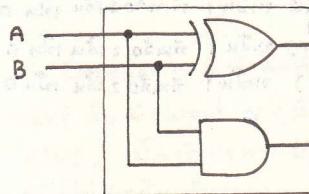
*x. ជំងឺ ADDER នៃ

↳ มี 2 ปุ่ม: เก็บตัว - half adder 10 บิต บวก 1 บิต (นับก้าวเดียว)

- half adder 107/25 บวกเลข 1 bit (นับก้าเดียว)

- full adder ນັກໃສ່ນລາຍ bit ດົງໝາຍ ຫຼື half adder ນາງໝາຍ ຕີ່

Half Adder

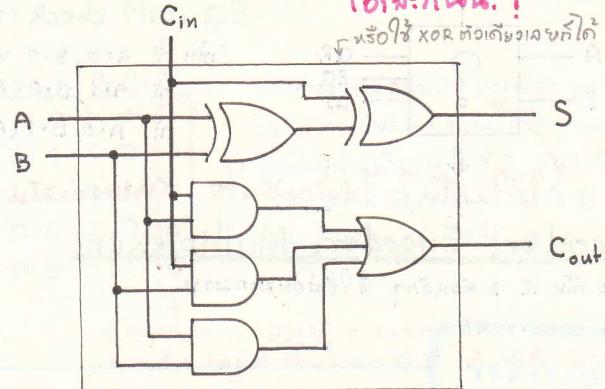


การทำงานของ half adder ดี๐ รับ input 2 ตัว

ສົ່ງ ຊັກນິຍາ ນາ ດາວກກັນ ສັງ output ເນື້ນ (sum)

๑๙: សំណើ output តាកច្ចាស់ carry (ឡាតក)

Full Adder



* ឧបករណ៍ circuit នេះ
គឺខាងក្រោមនេះ Truth Table
1010: តួនាទី: |

ເຫັນໃຈ້ xOB ທົວດີຍ

100

PHOTOGRAPH BY

S

```

graph LR
    HA[H.A.] -- A --> HA
    HA -- B --> HA
    HA -- S --> FA
    HA -- C --> FA
    FA[F.A.] -- S --> Cout
    FA -- Cout --> Cout
    subgraph Stage1 [ ]
        HA
        FA
    end
    subgraph Stage2 [ ]
        Cout
    end

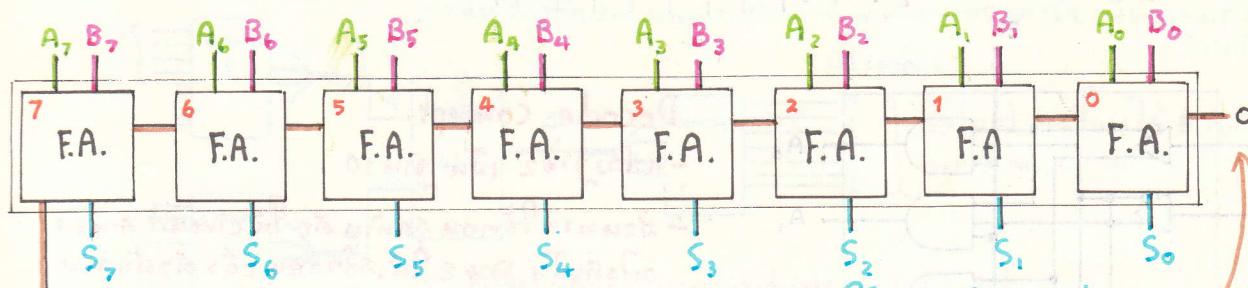
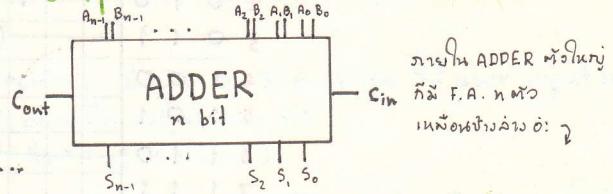
```

ใน circuit ก็ต้องการมากน้อยแค่ไหน เมื่อช่องว่าง Full ก็แปลงค่า
มันต้อง บอกเล่นคนควบคุม bit (นัก) ให้ จังหวัดลงมือการ
ติดตัวหัวใจที่มาจากการนักก่อโน่นหน้านั้นเมื่อหัวใจ (เริงกกว่า

เวลาเตาไฟใช้จริง ก็ H.A. แล้ว F.A. ว่า เป็นกลบงได้เบบ (ก็โจกบลไม่ห้าม) เนื่องจาก
มันเป็นดีซีบุ๊ก วงจรมาตราฐานที่ใช้ช่องช่อง แต่เขียนบนกระดาษกันว่า
กลบงนี้ถูกต้อง: สำหรับ input ใน output 9 ตัวอย่างมันดูบุ๊บ เนื่องจาก เดิมๆ 90
กลบงพากันนี้ถูกเบบ:

107 F.A. ມາດຕະກຳ ສຽງ ADDER

ສມານຕົວ: ນັກເລີນ A ແກ້ B (8 bit) ເຮັດວຽກ



* การบวกต้องทำที่ละหลัก มีเลข 8 bit ก็ต้องใช้ F.A. ถึง 8 รอบ

Cin Moush

三

ପ୍ରମାଣ ପରିଚୟ

⚠ Cout หัวสุดท้ายเป็น 0 ให้ check ว่าเกิด overflow ($Cout = 1$)
ดังเลขที่บวกกันได้ เกินกว่าที่ memory 8 bit จัดเก็บพอยต์

circuit ស្ថិកអេឡិចត្រូនុយ៉ា!?

Integration Circuit

インテグレーション

サーキット

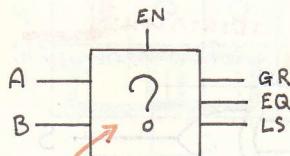
เป็นวงจรสำหรับประมวลผล 9 ช่องงาน ภาคเรียน
ก่อสร้าง ได้โดย เรียงบ่งๆ ว่า วงจร IC

7.17
MaiCh. ๑๙

→ เอกซ์เพรสส์ Integration อย่างผู้เชี่ยวชาญ calculus เฟรด: Integration คือหัวใจที่ทำให้เครื่องของเรามีความสามารถในการคำนวณ (นิดหนึ่ง!)

IC เป็น circuit ที่เก็บอยู่ เวลาเราใช้ไว้จะแลกเปลี่ยนไปต่อวงจรต่อไป

เรียกว่า EQ (Equality Checker)



EQ 用来 check ว่า A กับ B เท่ากัน (equal) หรือไม่

ถ้า $A=0, B=0$ หรือ $A=1, B=1$ output EQ จะเป็น 1 ทันที 2 แล้ว เป็น 0

ถ้า $A=1, B=0 (A > B)$ output GR (greater) จะเป็น 1 ทันที 2 แล้ว เป็น 0

ถ้า $A=0, B=1 (A < B)$ output LS (least) จะเป็น 1 ทันที 2 แล้ว เป็น 0

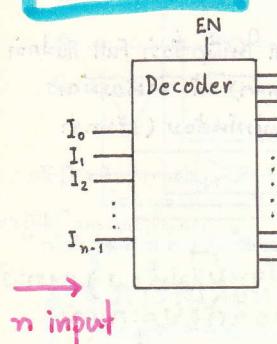
จึงใช้เป็น 7 ไบต์ของระบบ แทนแต่ละชุดที่ต้องการ

Decoder, Encoder, Multiplexer

↪ เป็น IC 3 หัวนักกษา ที่ใช้บ่อยมากวิธี

* EN ต้อง Enable เว้นแต่บุญ สวิง
ถ้า EN=0 output ทุกตัวเป็น 0!

Decoder

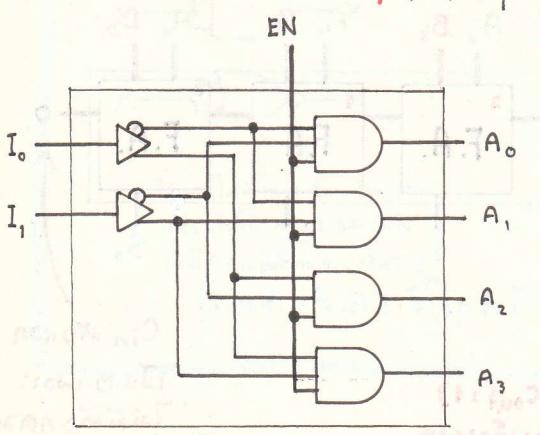


2ⁿ output

Decoder (หัวตอนตระหง่าน) ผันน้ำที่ แบ่งวงล้อฐาน 2 ให้เป็น ฐาน 10
ฐาน 10 ให้ตัวที่ 2 ไม่ได้หมายความว่า ตัวที่ 2 ของกัน ตามสัญญาณฐาน 10
(ข้อศึกษา 0 กับ 1 อยู่ด้วยกัน) แต่ต้อง กบ. ของเส้นที่ 0 กับ
จะเป็นฐาน 10 ... งั้น ? วัดดูตัวอย่าง

I ₂	I ₁	I ₀	A ₀	A ₁	A ₂	A ₃	A ₄	A ₅	A ₆	A ₇
0	0	0	0	1						
1	0	0	1							
2	0	1	0		1					
3	0	1	1			1				
4	1	0	0				1			
5	1	0	1					1		
6	1	1	0						1	
7	1	1	1							1

Input จะเป็น เงินฐาน 2 (I_0 แทนหลัก
หน่วย, I_1 แทนหลักที่ 2, I_2 แทนหลักที่ 3)
สมมุติว่า input จะเป็น 011 ซึ่งสอด
เข้ากับ 3 output A₁ เส้นที่ 3 (สี A₃)
ดังนั้น 1 กับเส้นที่ 0 ก็เป็น 0
ถ้า input 110 ($110_2 = 6_{10}$) A₆ ก็จะ
เป็น 1 หากเส้น 0 ก็เป็น 0 ก็เป็น 1



circuit ภายในDecoder 2-to-4

Decode Concept

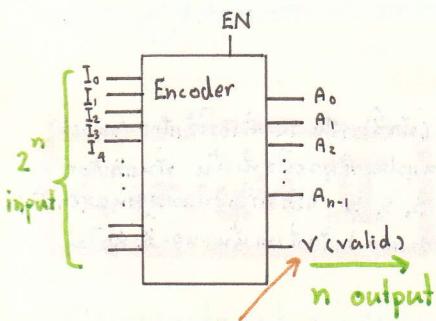
- แบ่งฐาน 2 เป็น ฐาน 10

- ส่วนมากใช้ก้อนลดที่อยู่ต่อ 9 หัว circuit ที่หนึ่ง
0: ให้เป็น ฐาน 2 ให้เป็น ฐาน 10 แล้ว ตัวอันดับ
ก็จะได้ 0 1 2 3 4 5 6 7 8 9

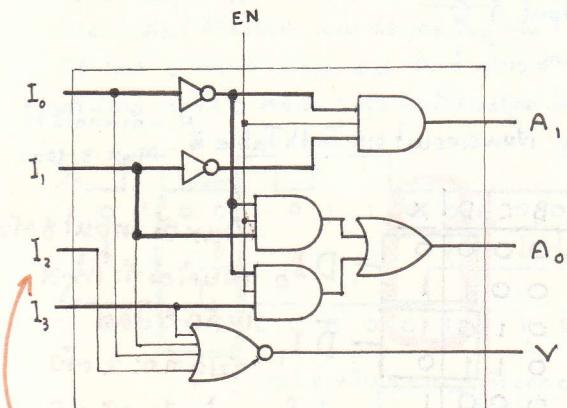
- output ก็เป็น 1 หรือ 0 เท่านั้น เมื่อ 0

- ให้ตัวอย่าง 0 ก็จะเป็น 0 ก็จะเป็น 0 ก็จะเป็น 0

Encoder



- Encoder ა: ეს თავისუბანი ტარგათ
Decoder და ოსტრეტ ვ (მას შემდეგ და
ეს ცარება) მძიმე დანართის მართვის ტრენი
სა ნარჩენის (დასაყვანი ნართის user იკვე
ნავიზე)



I₂, I₃ ໄສໄດ້ຮັບເລືຍ (ສົດວ່າການທີ່ I₀, I₁ ແລ້ວ

▶ Encoder (ตัวเข้ารหัส) หน้าที่ กดสัมภาระ Decoder สี 0 ไปครุยงาน 10 ให้เป็น
งาน 2 ด้วยๆ เมื่อต้องรับข้อมูล เช่น ปุ่ม เบอร์ 0-3 ให้ user กด ถ้ากด
0 - output = 00 , 1 - output = 01 , 2 - output = 10 , 3 - output = 11
แปลงมาเป็นฐาน 2 เป็น 0101 นำไปให้ circuit อ่านฯ สามารถดูนิพจน์ผลผูกพัน
เพื่อ circuit ออกແນบນมาสั่นรับ เลขงาน 2 (ແນบ adder 7 位 ไว้บวก
ลงราก 2)

I_3	I_2	I_1	I_o	A_1	A_0	V
0	0	0	0	X	X	1
X	X	X	1	0	0	0
X	X	1	0	0	1	0
X	1	0	0	1	0	0
1	0	0	0	1	1	0

หลักการทำงานกลับกัน Decoder ทุกอย่าง
ยกเว้น ก้า input มาก 2 เส้น (ไม่ว่า จะ เต็ม
1 หน่วยไปแล้ว เมื่อ 2 เต็ม) ยังจะได้อว่า
เต็ม 2 เต็ม ก้า I_0 ก่อน เช่น กด I_1 กัน I_3
output จะได้ 01 ๗๘๙๘ ๑ !

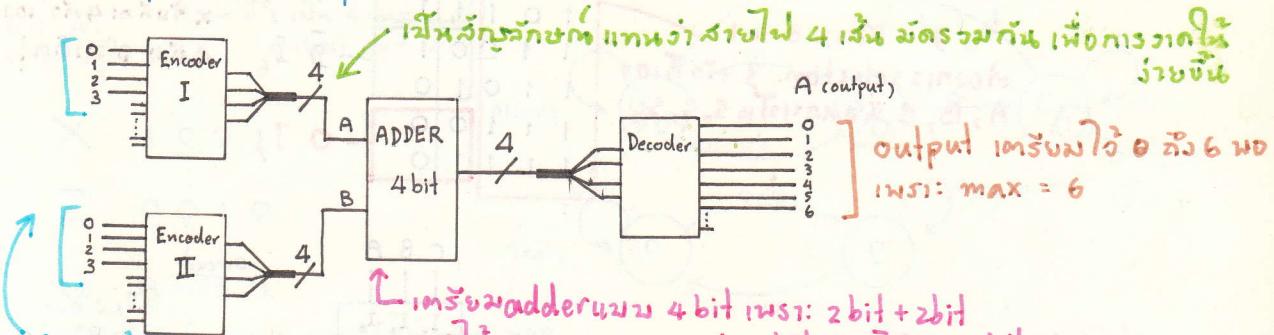
Encoder ของแบบนี้ ต้องดูว่าเป็น 1 ทิศทางเท่านั้น
หากเว้น input ทิศทางใด ก็จะได้ ค่าสัญญาณ

Encoder Concept

- ແປ່ນຂົງຈຸນ 10 ເພີ້ນຈຸນ 2
 - ສ່ອນມາກເຕັມໄວ້ຮັບດໍາລັກ user ແລ້ວແປ່ນ
ເພີ້ນຈຸນ 2 ສ່ວນໄດ້ດໍາລັກ
 - ຖັນກຮັບສິນ input ລາຍເນັດ ມະນີຕ່າງໆທີ່ຖືກລົງໄວ້
 - ມີ V ເພີ້ນຕົວນັກງ່າວສິນ input ຂອງຂ່າຍ
 - ເຫັນມາກ 100 ພົມ

Ex. ពីរបៀវការនេះ Decoder ក្នុង Encoder 9-to circuit, ដូចខាងក្រោមនេះ និង user input នៃលេខទូរសព្ទ 101 និង input នៅ ទូរសព្ទ 2 គួរមានចងការណ៍ ដែលបាន output តាមលេខ

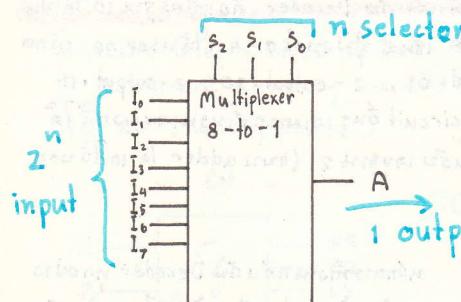
ເບື້ນ ຂອງ 1 ນາ ② ຂອງ 2 ນາ ③, output $A_5 = 1$ (ນອກນິນດອ)



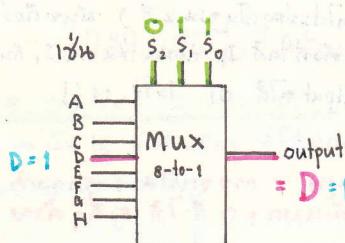
ก้าวเดิน
ที่ ๐ ถึง ๓ เดินน้ำ
ก้าวเดินชั่งมัน

* ໄກນທີ່ຈະຕືອນມາຈາດ circuit ໃນປົງໂຄ (ແກ່ input ກີ່ 8 ນັ້ນແລ້ວດີ່
K-map 3D ມີ ຕາງພົດຕະ) ອີ່ຈະກໍລົງມາຊ່ວຍ ກ່າຍຫຸ້ນເບຍໂທ + ເສັ້ນການດັວຍໄ

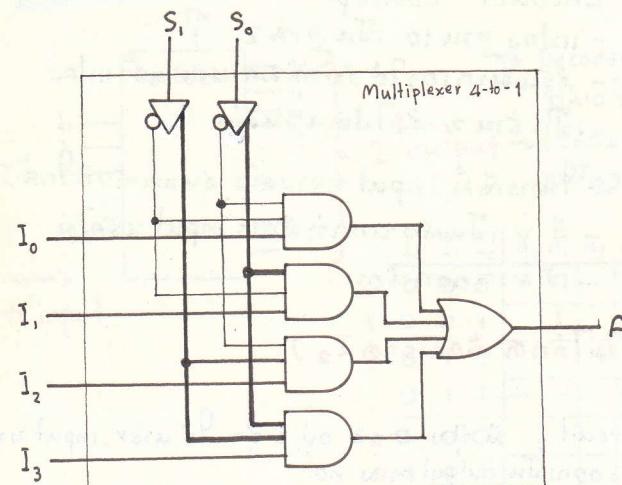
Multiplexer



► Multiplexer หรือ ชุดๆ ว่า Mux (มัคซ์) เป็น IC ที่เอาไว้เลือก (select) ข้อมูลแล้วส่งผ่านไป ใจมีการเปลี่ยนแปลงที่อยู่อาศัยในตัวเลือก อย่างเดียว... แล้วเลือกชิ้นไหน ก็ต้อง S_2, S_1, S_0 จะต้องให้ในรูปแบบนั้น 2 (สองจ้า) และ S จะเป็นเมื่อรูปแบบ 10 ได้เท่านั้น คือ 1 input เดียวที่ทำได้ 8 แบบ: สิ่งต่อไปนี้



สมมุติให้ selector = 011
 $011 = 3$ ให้ส่งเงิน I_3 (สี D) มาเป็น output
 ก็ D มาเป็น 0 output ก็ 0
 1 " 1

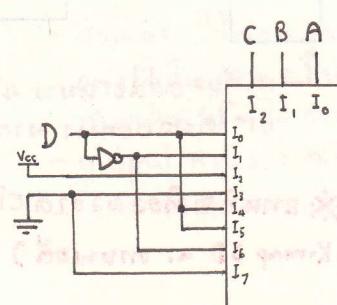


► Mux 8-to-1
 ต้องการ selector 3 ตัว ก็ต้อง A, B, C มีแบบเป็น S_2, S_1, S_0

Ex. รับ circuit ตาม Truth Table นี้ - กำหนดให้ 9 ตัว

A	B	C	D	X
0	0	0	0	$D I_0$
0	0	0	1	$\bar{D} I_1$
0	0	1	0	$1 I_2$
0	0	1	1	$0 I_3$
0	1	0	0	$D I_4$
0	1	0	1	$\bar{D} I_5$
0	1	1	0	$0 I_6$
0	1	1	1	$\bar{D} I_7$
1	0	0	0	$D I_0$
1	0	0	1	$\bar{D} I_1$
1	0	1	0	$1 I_2$
1	0	1	1	$0 I_3$
1	1	0	0	$D I_4$
1	1	0	1	$\bar{D} I_5$
1	1	1	0	$0 I_6$
1	1	1	1	$\bar{D} I_7$

Mux ต้อง input 8 ตัว
 $D I_0$ เลยต้องทำให้ x
 เปลี่ยน 8 รอบ
 จำนวนที่ก็: 2 รอบ
 แล้วเทียบกับ D
 (input หัวสุดท้าย
 ก็เปลี่ยนด้วย)
 ※ ถ้าเป็น Mux 4-to-1
 ก็ต้อง A, B เป็น Selector
 x จำนวน: 4 ตัว 10
 คัน D จะได้!



Circuit

- สำหรับ Mux
- ต้องใช้ K-map
- แล้วเขียนสมการมา
- ก่อนต้องวาด circuit

Midterm
 / fino

Digital After Midterm

ກວດບ່ອງກໍາດ້ວຍ!
ສັເລະສ! ກວດຕົວໃນວິນເລື່ອວ່າດໍາເນີນຮູ້ຮັບ!!
ນີ້ລະ: ຂອງຈົວ... ອັດຕະໂຫຍກໄດ້ ທີ່ຈົວງາງ

part II (for Final)

ສຽງໂຄຍ ຕີ້
TTad #
sec.1 5188018 ICT

Circuit II

Circuit ຂອງ 2 ຊົດ

combinatorial

(ໄຊວະກິດ Midterm ສັນນົດ)

Sequential

(ໄຊວະຈຳກັນ memory ... output ກະນຸມໃຊ້ຈຳກັນ input ສັກຮັງ!!)

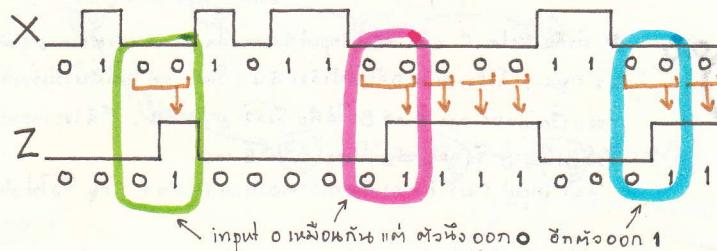
Combinatorial circuit : output depends on input ONLY!
composed of gate only.

state ໄກສດວຍຫັງ Q

Sequential circuit : output depends on input and STATE
composed of flip-flops and some gate.

ສຽງຈຳກັນ Sequential ຫຼື circuit ທີ່: memory Data in
ເຕັມໃຊ້ສັບຍໍາໄປດີກ່ຽວກັບ input ທີ່ສັງເກີນໄກນ໌ ເຊັ່ນ
ຕັດກົດວ່າລັດເລີ້ມ ມັນຕີ້ວິ Latch ກໍ່ຕົກ clock
ເພີ່ມເຫັນໄປ (ງວດ: ດີ? ດ້ວຍຕົກລົງ).

Pb! If last 2 inputs were 00 then output is 1 else 0.
ດ້ວຍກັບ input 0 ຕ້ອງກັບ 2 ຜັກ ດ້ວຍໃຫ້ output ເປັນ 1 ກຽດຕັກເປັນ 0. [ດູຫວາຍ Timing diagram ທີ່ນຳກັນ]



ດ້ວຍເວົ້ວ ວານເຫັນ! ... ລວງດີດຽວ
ສ່ານແບນນີ້ເປັນ Truth Table
ກັນ K-map 7 ຊົດ 5 ອຸນເສື້ອ input
ແນວດັນກັນແລ້ວ output ຕ່າງກັນ
ແບນນີ້ຕົວຢູ່ State ພາຊົມ!

X	Z
1	0
0	?

Truth Table ສອງນີ້
ໄດ້ຮູ້ຈະຕົນໄວ?

State Diagram

ステート ດයັກ-ເກົລ

Moore
Mealy

ສິ້ງ:
output ອູ້ໃນ state
output ອູ້ທີ່ເຊັ່ນ

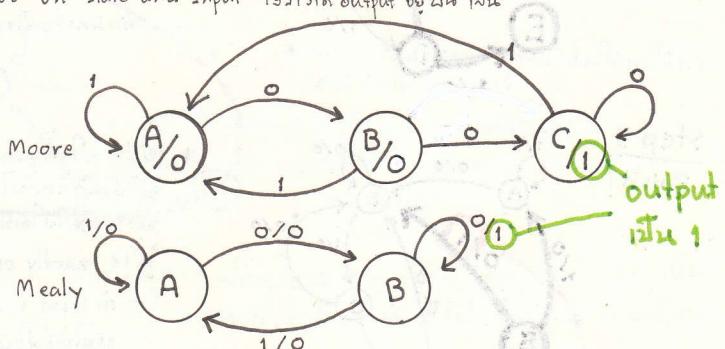
X 1 0 0 1
Z 0 0 1 0
Q A A → B A

ໃນ state A ດີ່ວ່າລາຍ: ຕອນຍົມໄຈ 0 0 ນຸ້ຮັບ
ເປັນ 1 0 0 ຕ້ອງກັບ

State B ດີ່ວ່າ 1 0 0 ຕ້ອງ 2 ເກຣ: ດັບເປັນເພື່ອ

ດ້ວຍໃຫ້ state B output ທະອອກເມົາ!

State C (ສິ້ງຮັບ Moore) ດີ່ວ່າ ມົນທີ output
ກຳລັງອອກເມົາ 1 0 0



state ດີ່ວ່າປະດຳໄດ້ຕາມ input ໄປສ່ວນ output
ຕ້ອນເປັນ Moore ອູ້ໃນ state (ກວມຈຳ)
Mealy ອູ້ນັ້ນ / (ບໍ່ນັ້ນ ຕ້ອນຈຳ)

ແນວ State Diagram ວາດໄວ ມັນລັກກາຣົດຕົ້ນ...?

→ ॥៩៩៩៩៩ ១០១ ១១១ ៧៩៩១០១ ១១១១ នេះ០ ១១១១...

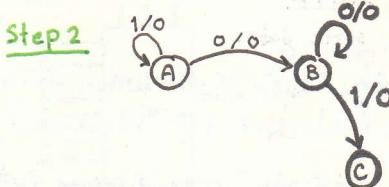
▶ การฯ: เป็น 1 ลักษณะงอกซึ่งกัน ที่ ก็คือต้องผ่านการดอง 0 แม้จะถูกดูแลดี

សម្រេច នឹង ១០០ [01110] នៅលើ output នេះ ឱ្យ
input → output

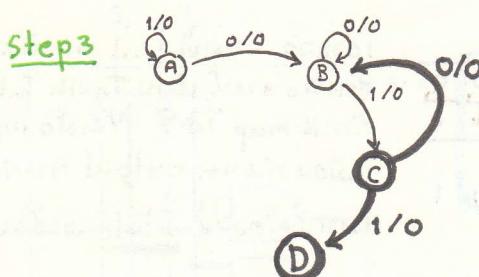


▶ เริ่มต้นยังการสร้าง State A มา ก่อน กันต่อไปต่อการหนึ่งก่อนเพื่อ make sure ว่า input ที่ต้องมา (ของเก่า ก่อนหน้าอีกด้วย Data ต่างๆ เป็น 1 มาก่อนแล้ว)

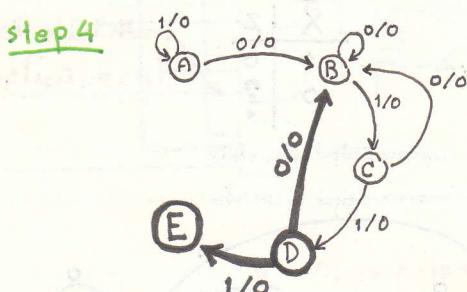
- ถ้า input 1 ก็บุํมัคทองนั้นที่ state เดิม ผ่านไป 1 หน่วย: 1:101 101 0:
 - ถ้า input 0 101 101 ตัวแรกแล้ว ธรรมดาวนั้นก็ต้องการ ก้าวเดินใน state ต่อไป แต่ปัจจุบันไม่ได้ก่อน 101 101 111 0 ตรงทางก้าว แล้วต้องผิดไปทาง 1



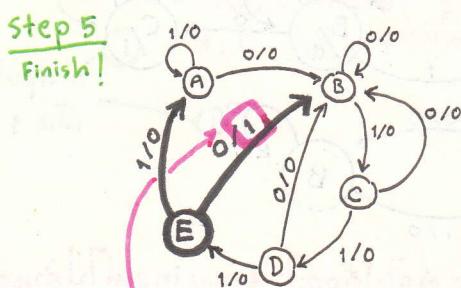
▷ 例 State B ที่รับ input 0 ก็ให้ผลตอบแทนเท่ากัน (0, 0) แต่หาก
ก่อ input มาเป็น 1 (ได้ 1 ห้องนอนแล้วว้า!) ก็จะ state ต่อไปได้



▶ ມາສົ່ງ State C ມີແນວດີ input ດັ່ງນີ້ເປົ້າ 0 (ເຊື້ອທິດການ 1 ຕົກ
ຕົກ 2 ຍຸ່ນໆ) ກີ່ຕົກ ຍຸ່ນກັນໄປເຮັດ ນັ້ນ 1 ຍຸ່ນໆ ແຕ່ໄປກັນໄປເຮັດກໍ່ A
ເພື່ອ: ເວັບໄລຂອງການ $A \rightarrow B$ ໄດ້ສົ່ງ ສ້າງວ່າ input = 0, ກີ່ໃຫ້ input 0 ນາວ:
ສ້າງວ່າເປົ້າ 0 ຈຶ່ງກົດລົບໄປ ຮັດ 1 ແລ້ວ B ພວຍ
- ດີ input 1 ພາ ກົດຮຽນ ກັນທີເຮົາຕົດການການ 1 ຢັ້ງສັກ 2 ຍຸ່ນໆ ຈຶ່ງໄປ state ຕົກປົງ



▷ ចំណាំ... កើតម្រោះវគ្គសាខាត្រឹមលើ input នៅលាស 0 ដោលកែល្អ បញ្ជីនៅក្នុង 1 មាត្រាបច្ចុប្បន្ន ក្នុងការបង្កើតឡើង ឱ្យបាន និងវាមិន Step ការបង្កើតឡើងទៅ B
 - ជាបន្ទាល់ input នៅ ក្នុង state នៅវាទេ ឈាននៃវាត្រូវតើ 111 ក្នុង 3 មាត្រាបច្ចុប្បន្ន
 ॥១. output នៅខាងក្រោម ឱ្យបានរចនាបានដោយ 1 step ធោរ: ពីចុងចាយទីនៅ
 វាត្រូវបានបង្កើតឡើងដើម្បីទី 1 ធោរ: ឧបាទីនៅក្នុងរឿង 1111 4 ពេលវេលាការណ៍
 និងធានាដើម្បី!

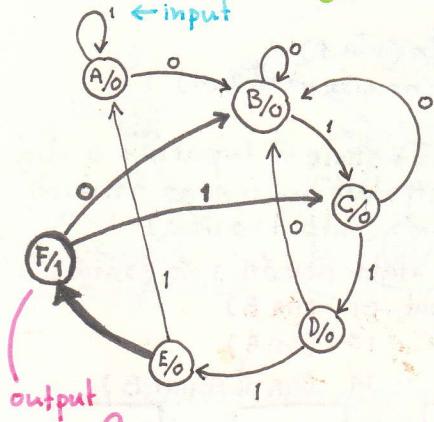


ສິນກາແກນຕາງເພື່ອ output
ທີ່ໃນທີ່ສຸດກໍຈະຍັງມີມີ, ຈົດ
ຫັວໜ້າກວດ: !!

- ถ้า input 1 มำส์...!? งานเข้าหนักมากที่สุด! ！ เกิน 3 ตัวแล้วหัวงู
กลับไปเริ่มใหม่ นะ ใจป้ำๆ เริ่มที่ B (หัวชี้ลงทางไว้เริ่ล 8 กันนนนด้วย!
input = 0) แต่ที่ input=1 หัวลงกลับไป หัว Step 1 ไปนี่ ก็ต้องกลับ
ไว้ต่อ รู้ว่า ตัวแรก ก็ A อะ !!

ຈົບແລ້ວ! (ໄມ່) ຈະຍືນສົມ...?
ຖຸເຊັນບໍ່ໄວຕ່ອງ...

▶ หัวข้อ State Diagram เมื่อกำหนดเป็นนิรនทร์แบบ Moore System.



► In Moore System output: ពីរឹងចុះ នៅ State F ចុះទៅក្នុងការសម្រេចដែលត្រូវការសម្រេចដែលត្រូវការសម្រេច

- 例 7: กำหนด \overline{B} ให้เป็น Mealy state ที่มีต่อไปนี้ ให้ \overline{B} เป็น output ของ:

output
નિયાયિક

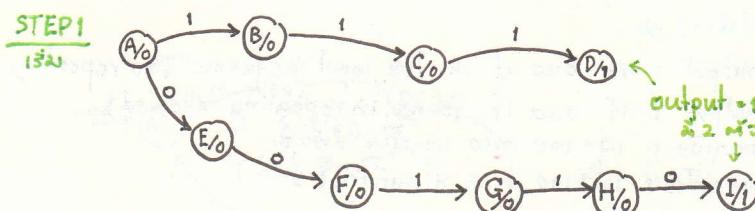
State

Pb.3. output is 1 if either or
in Moore System.

[input has 3 consecutive 1s (111)

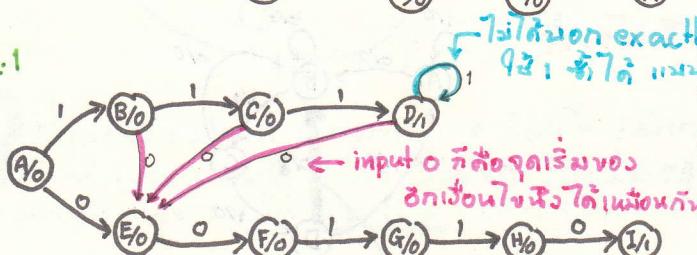
{ ๑๗๐ । ๓๕๙/๑๕๖/๒๖

input has 2 0s followed by exactly 2 1s (00110)
ນີ້ແມ່ນ 0 2 ໂດຍມີນຳນັ້ນ 11 (1 2 ໂດຍມີນຳນັ້ນ)



▶ เริ่มต้นการ เบิกบัญชี 2 เดือนแรก ก่อน (แยกกันก็ได้) :

Step 2.1

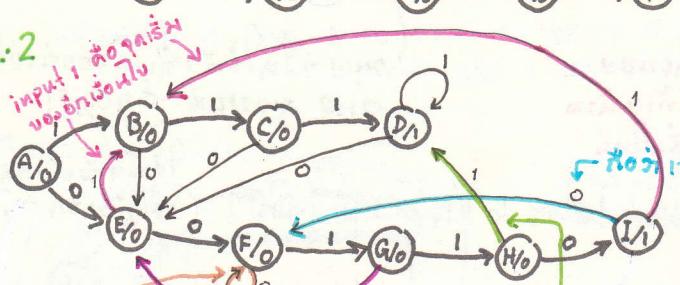


► ຕົກສິນອອກຈາກ State 9² ດວບ

► ពេលវេលាលើករារក្នុងរដ្ឋបាល State និងរដ្ឋបាល

* เส้นทาง State ที่ 1 ไป State ที่ 2
(input: input บ 0 กม 1) ผลลัพธ์
จะเป็นไปได้ก็ได้

Step 2. 2



▶ หาดผิดก็เข้าใจห้องน้ำ State ไม่ลงในมันก็ได้ ก็
จะหนีเข้ากับ Logic นี้แล้ว:

ในตัวที่ 1 ของ เรื่องนี้ 2
(0.93-0.77 คือ 甲状 ของ
exactly 11)

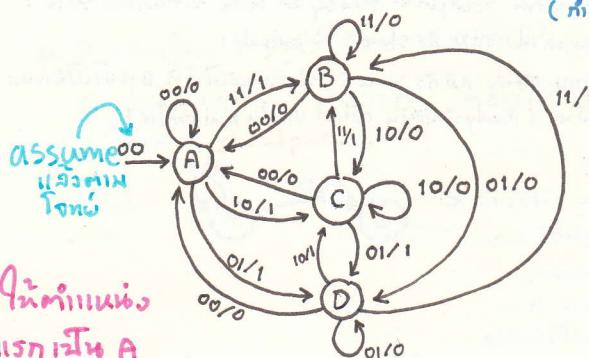
ກຳລັງທຽບເນື້ອໄປ 2 ອົບໆ ແລ້ວ ດັນໄດ້ 111 ວັດເຮັນກັນ
ແກນ ກີເຄຍໂດດ ຂໍາມໄປເສັ້ນກາງ ບະແກນ

๑๗๐ ○ มองหา exactly 11
งานให้ก่อนไปเริ่มนากันมีด้วย

ຈະໄລ້ (ຮອນ 27) ນັກຄວນບັນຫຼືບັງ?
ເຫັນຢ່າງຕໍ່ໄປຈະໄດ້ຮັບນິຍາມເນີນ Step ເລື່ອນ!

Pb.4 output is 1 if inputs A and B either show a rise in value
(assume all A, B has zero value at start)

output 001 เมื่อ input A หรือ B ตัวใดก็บันทึก เมื่อซึ่งหนึ่ง (12pm 1)
(ก็คือ A หรือ B ก่อนจะเริ่มจาก 0)



ก็คือเมื่อเริ่มต้น

แรกเป็น A

แล้วเป็น B

ผลลัพธ์ output

* หมายความว่าต้องมีการเปลี่ยนแปลง
ดังต่อไปนี้
(นั่นคือต้องมีการเปลี่ยนแปลงที่ไม่ซ้ำกัน:
ไม่ว่าจะเป็น A หรือ B ต้องมีการเปลี่ยนแปลงที่ไม่ซ้ำกัน)

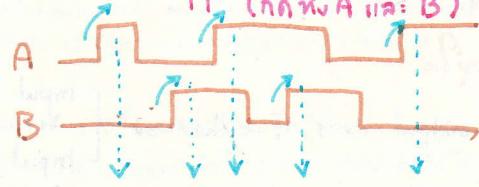
ถ้า A เป็น state ที่ input ไหน 0 ก็ได้
ไม่ต้อง state ไหน 1 ก็ได้ แต่ต้อง A ทุกตัว (output = 0 ถ้า 0)

▶ เมื่อ State เปลี่ยนราก 3 ตัว (3 ครั้ง)

input 01 (ไม่ B)

10 (ไม่ A)

11 (ทั้ง A และ B)

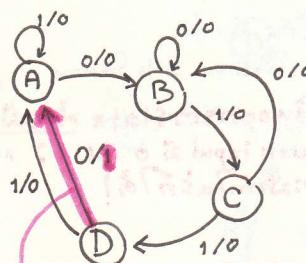


Pb.5 5.1 draw Mealy System 5.1 output 1 if 0110 is in the input sequence (no repeating)
or 5.2 output 1 if 0110 is found. (repeating allowed)

5.1 0001 เป็น 1 เมื่อ 110 0110 ไม่มีอะไรซ้ำซ้อน

5.2 0001 เป็น 1 เมื่อ 110 0110 ที่เกินก็ได้ (ซ้ำได้)

5.1

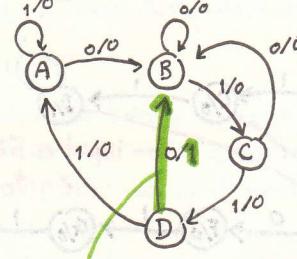


ต้องการแบบนี้ ไม่ซ้ำกันแบบ
0110: 0110; ดังนั้น 0 ห้ามติดกัน
ดังนี้!
ก็ต้องกดปุ่มนี้เรื่อยๆ

ก็ต้อง



5.2



อนุญาตให้มีซ้ำซ้อนได้ เช่น
0110 จะสามารถ [0][1][1][0]

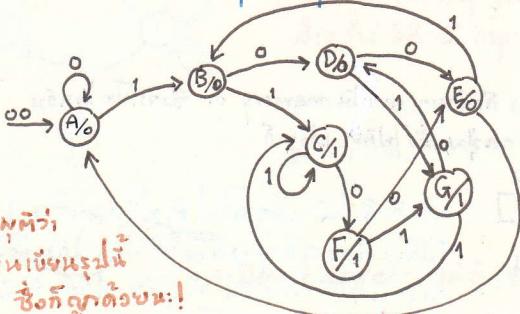
ได้เมื่อผู้ตัวเลือกต้อง^{กด}
แค่ก้อนแรกจะพอ

Pb.6 output is 1 if there are more 1s than 0s in last 3 input

— 27 —

(assume: 0s were in the last input at start)

ออกเมื่อ 1 เวลา 3 input ล่าสุด มี 1 มากกว่า 0, กำหนดให้ input ก่อนหน้า มาเป็น 000...

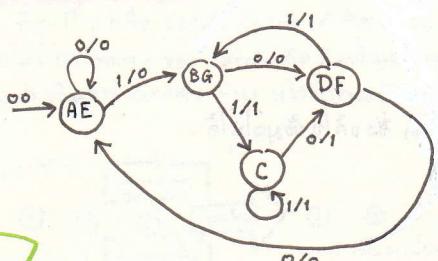


สมมุติว่า
ทั้งเท่านี้เป็นไปได้
มากกว่า ซึ่งก็ถูกต้องบน!

	I_0	I_1	I_2	Z	Next State
A	0	0	0	0	A
B	0	0	1	0	D
C	0	1	1	1	C
D	0	1	0	0	E
E	1	0	0	0	A
F	1	1	0	1	E
G	1	0	1	1	D

- กำหนดนี้ก็ถือว่า เสร็จเรียบร้อยแล้ว: แต่ ใช่กัน! ดู State มันเบองไว้: แบบ...
- ▶ B กับ G เว็บ nextState เหมือนกันเดียว ▶ A กับ E นี่ยังเหมือนกัน next State
 - ▶ D กับ F นี่ก็รักกันเดียวที่ next State เหมือนกันเวย ก: output เอง

∴ ก็แปลกว่า B กับ G, D กับ F สอง State กันได้ (แต่เท่ากันทาง output (Z) ไม่เหมือนกัน ก็ต้องรู้ว่า สองตัวเขียนแบบ Mealy เท่า output จะได้เป็นตัวเดียวกัน State ที่ไม่เหมือนแบบ Moore ขั้นบน!) ... ส่วน A กับ E เนี่ย แม้แต่ output ยังไม่เหมือนกันก็ยังรวมกันได้ แต่ C ใจกว้างรวมกันได้โดยเด็ดขาด เพราะเป็น State ที่เต็มไปด้วย 1 อย่างเดียว (ถ้าบุ๊ฟแล้ว 1 มากกว่า 0 แน่นอน)



◀ เป็นมั้ย!? ดูว่าเขียนง่าย:
※ ยังไงเมื่อ มันໄ่บี๊ก ก็แนบมาไว้กัน!

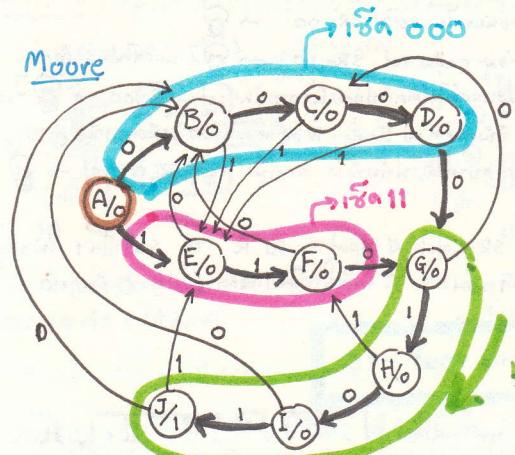
การรวม State จากมาก → น้อยโดยสังเกต next state ของมันว่า มันซึ่งกันและร่วมมือได้ เรียกว่า **State Reduction**. ภาระ: ประโยชน์ลดลง很多
ใน circuit เพราะจะห้าม flip-flop ซ้ำๆ กัน (บ้านต้องห้าม...เดียว ก็พอแล้ว!)

หลัง ใจ path
ภาคผนวกเล่ม:
หน้า 60.

107 ใจว่าง
เพื่อไม่บี๊กเมื่อ

Pb.7 output=0 if has 3 0s (3 zero) or 2 1s (2 one) followed by 0101

ก้าว 0 สามตัว หรือ 1 สองตัว ที่ตามกันด้วย 0101



▶ จะเห็นว่า ดังนี้: แยกวิธี กันตัดในตอนแรก
ประกอบหน้า 0101 ก็ใช้ชุด กันได้

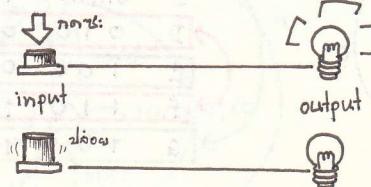
รีด 0101

Latch & flip-flop

ลัช แอนด์ ฟลิป-ฟล็อป

► Latch ๙๙: flip-flop เมื่อหัวเก็บว่างอนนี้อยู่ใน State ไหน ก็จะอยู่ ก: เป็น memory ๔: ซึ่งจำเป็นสำหรับ Sequential circuit มาก่อน ... สมมุติว่าถ้าเรามี circuit กดปุ่มแล้วไฟติด แล้วก็ ก

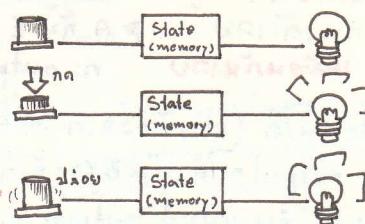
combinatorial
(แบบธรรมชาต)



กราฟ: ให้ใน หลอดไฟ ติดใน circuit บนนี้ ทำเมื่อต้องกดปุ่มตัวไว้ กี่วินาทีไฟก็ติด กันบังกอกได้ circuit ที่ แบบ: ปล่อยก็ตัวไปติดอยู่?

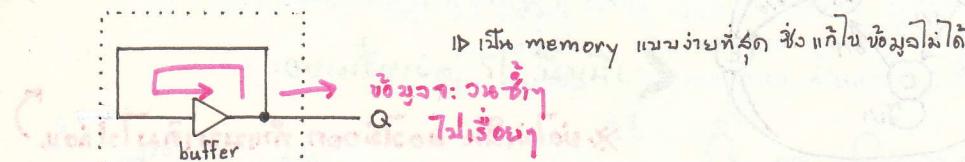
Sequential
(มี memory)

ปล่อยเบนเมื่อันกัน
แล้วเมมเบรนเปิด
หลอดไฟยังคงติดอยู่



sequential ๙: ผังวงจร flip-flop มีใช้ mem. ก: output กดปุ่ม: เป็น: ๔: ซึ่งเขียนกับ State + input ໄว้เขียน input อย่างเดียวเหมือน combinatorial

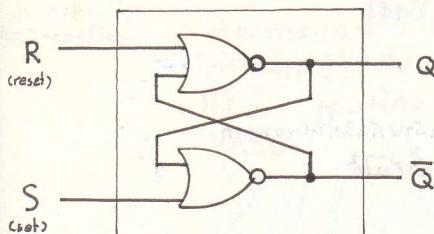
ROM (read only memory)



Latch

เป็นร้อง memory ที่ไม่ซึ้งกับเวลา (clock) บันทึก flip-flop ที่ไม่ต้อง clock ทุกแบบ:
Latch มี 4 ประเภท SR, D, T, JK

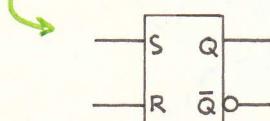
SR-Latch (set-reset)



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

undefined

!
น้ำหนัก SR
พร้อมกัน
ไม่ซ้อนกัน!



* ก้าวเดียวหากลับแล้วให้ใช้ S
อยู่ชั่วขณะ R ก็ได้!

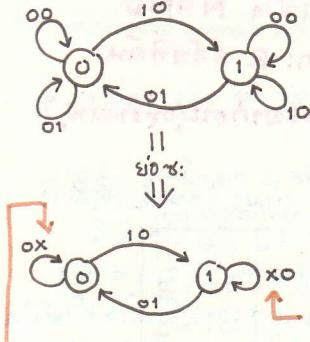
► ต้องบ่งช่อง ๒ ช่องให้ตัด ณ output Q สมมุติให้มีเงื่อนไข: กรณีแรกให้ SR=00 \rightarrow หลอดไฟติดแล้ว กดปุ่ม set SR=10 \rightarrow หลอดไฟติดแล้ว กดปุ่ม reset SR=00 \rightarrow หลอดไฟติดแล้ว กดปุ่มต่อไป ! ที่สุด กด input เป็น 00 เหลือบการกดแล้ว แจ้งขากไฟไฟลับก็ต้องตัด กรณี reset คือ SR=01 \rightarrow หลอดไฟติดแล้ว

ที่ SR-Latch มี input S กับ R ให้ S ปั๊ม ให้ output Q=1 จนกว่า R จะเข้ามานี่ (reset) Q ก็จะ = 0

กด set \rightarrow 1 กด reset \rightarrow 0

สมมุติว่า Q=1 อยู่แล้วแล้วกด set ต่อ so Q ก็จะ变成 1 อยู่ต่อไปแล้ว กด reset ลาก ก็จะ变成 0

State Diagram von SR



Latch (หรือ flip-flop) มี 2 State คือ 0 หรือ 1
สั่งรับ SR 作为 input 2 ตัว (S ก่อนแล้วถึง R)

- ▶ กดเข้ากันตรง S (10) จะได้ state ① output เสียง 1
- ▶ กด กด R (01) ก็ได้ กดซ้ำไป state ② output คือ 0

Don't Care : กรณี S จะเป็น 0 ใจ กด ไลน์ reset ก็จะ 0 เว้นๆ

Don't Care : กดไลน์ set จะ กด output ก็จะ 0 ต่อไป

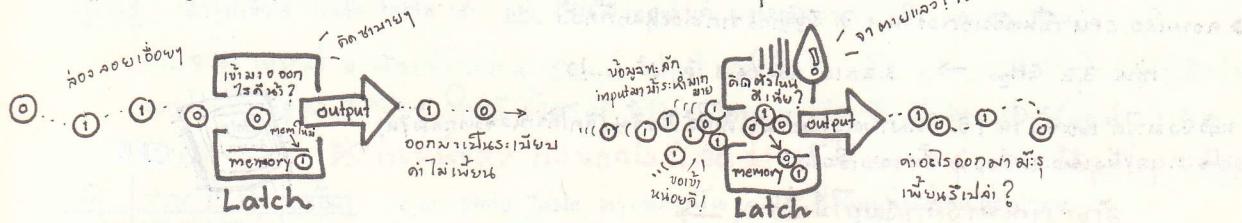
Note

Latch, flip-flop 1 ชุด เก็บข้อมูล (memory) ได้ 1 bit = 2 state
ก็มี n ตัว ก็เก็บได้ 2^n state

เช่น ก็มี state ทั้งหมด 5 state ก็ต้องใช้ Latch (หรือ flip-flop) 3 ตัวเก็บ (หรือ 2 ตัวได้ 4 state 3 ตัวได้ 8 state 10 ตัวมากกว่า นี้เรื่องที่กูไม่เข้าใจ)

SR-flip flop

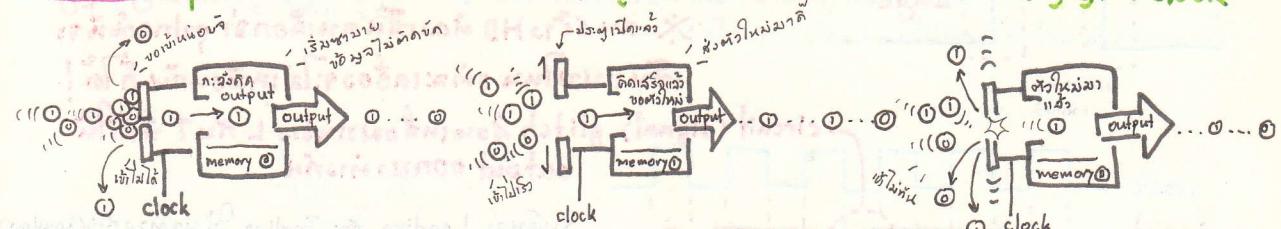
↳ flip-flop ก็ต้อง Latch ร่วมกัน กับติด clock (สัญญาณเวลา จาก CPU) และ... แล้วติด clock เห้ๆไปสู่ประบบที่จะรุปแบบการทำงานของ Latch ดังรับข้อมูล (input) บาง 101 นิ, ดังๆ กับข้อมูลเดิมที่อยู่ใน (memory) และต่อ output...
แล้ว กด ข้อมูลใหม่ (input) ดังนั้นการวนว่างาน อีก 1 รอบ ก็ต้องมี output ใหม่ๆ



สถานการณ์ที่ Latch กำลังหิงหองอยู่
input ต้องๆ กด แล้วค่อยๆ ติด output

Latch ที่ กด ลงมา! input มากเลย ติดไม่ทันแล้ว
แต่ ยังต้อง รับ output (ลงมา 1 00 กว่า...) ดังนั้นว่า กด แล้ว
ข้อมูลตอนนี้อยู่ input ยังคง แล้วอีก !!

flip-flop จึงเกิดขึ้นมา ตั้งงบเหตุผลนี้ โดย flip-flop นั้นจะ ทำงานเหมือน Latch ทุกอย่าง
ยกเว้น input กับเข้ามา (อย่างนี้ด้วย) จะ ถูกจัดการให้เข้ามา ใช้วิธีนี้ สัญญาณ clock



flip-flop นี้ clock ช่วงๆ เลี้ยง input ให้:
เพื่อให้ตัว 0 ได้สักกัน

เมื่อสิ่ง ร่องรอย: แจ้ง clock ก็จะ:
ขอนี้ใน ข้อมูลผ่านมาเข้าไป

แต่ input ผ่านเข้าไปได้แล้ว
แม้เดียว เห็นนี้ เพื่อไปใน flip flop ก็ต้องทัน!

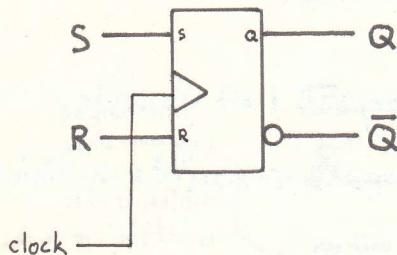
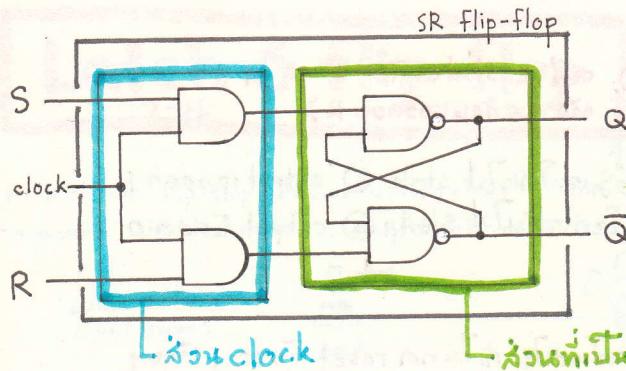
ต้องแบบนี้วิธีการนั้นๆ...



↳ input ที่มองไม่เห็น: เป็นยัง Latch อย่างก็ต้องทัน



ซึ่ง input ก็ ควร จ้วนๆ: clock เท่านั้น



ต้องป้องกันเมื่อกลับค่าให้
แบบนี้เมื่อ clock หันมา
จากไหนก็ได้แต่เขียนงบ
ด้วยว่า เมื่อ "clock"
หันเข้า flip-flop จะรักษาอยู่

*: เปลี่ยน NOR เป็น NAND
เพื่อนิ้ว input S กับ R ต้องหันกัน
(มันชื่อ SR ต้องรวมมาก่อน(อยู่ข้างบน))

ID ส่วนที่เป็น input เพื่อมาเก็บหันให้วัน
Latch ก็ต้องไป AND กับ clock ก็
แบบง่ายๆ ถ้า สัญญาณ 1 จาก clock
เข้าไปเท่านั้น input 1: มาปะปันไว้ก็ได้แล้ว
แต่ถ้าสัญญาณจาก clock หายแล้วต้องหัน
input หันไปเท่า ส่วน Latch

แล้วสัญญาณ clock มาจากหน้า?

↳ มาจาก CPU ... เดียวกันซึ่งก็ หมายความว่า CPU เริ่มทำให้ในรูปแบบไฟ 2.0 GHz ช้าไป 3.2 GHz ช้ากว่า รู้สึกนั้นหนักต้อง clock!

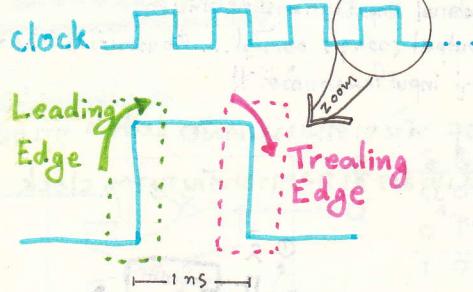
▶ ตอนเริ่ม CPU เป็นตัวบังคับๆ กัน 1 วิ. สัญญาณจาก clock หายก็ครับ

$$\text{ เช่น } 3.2 \text{ GHz} \rightarrow 3.2 \times 10^9 \text{ ครั้ง/วินาที } 3. \text{ (วินาที...)!}$$

แล้วซึ่งมาได้ เช่น 1. เวลา: ชั้น 1 วิ. กันนยาดความต่างบันไดต่อเรื่องซึ่ง (ติดต่อเรื่องความมากน้อยซึ่ง)

↳ ∴ เลขเชิงเสียง: เครื่องทุกตัวจะเลขเริ่มต้น

สัญญาณมาทั้งหมดไปเรื่องๆ →



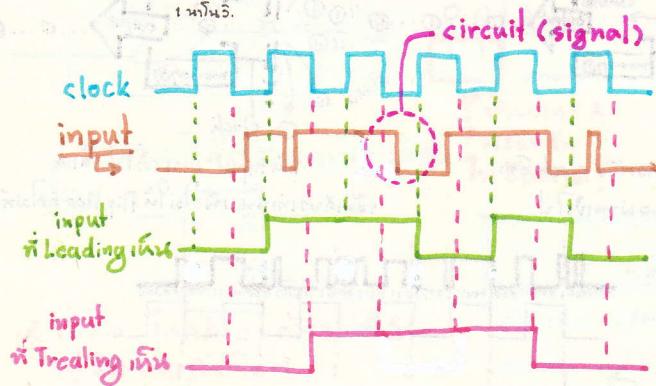
▶ แต่สิ่งที่สี clock ก็ยังมีสัญญาณบันทุณี่เรื่องนี้ด้วย ส่วนที่เป็น 1 ของ clock หันตัวเองเข้ามายาว (ส่วนรับไฟฟ้า ก็ต่อ 1 วิ หมุนเวียนกัน 1 วิ)
มอง เตรียม Hardware จังหวะ เล็กๆ ๆ ติดจุดนี้ เท่านั้นรับรอง

- มองที่ต้องเปลี่ยนจาก 0 เป็น 1 เรียกว่า **Leading Edge**
- มองที่ต้องเปลี่ยนกลับจาก 1 เป็น 0 เรียกว่า **Trailing Edge**

*: ตอนนี้ HD ต้องเขียนคนเดียวกันจึง อุปกรณ์นี้จะ:

เขียนแบบไหน ก็ตาม เครื่องทุกตัวจะเขียนแบบนั้นกันได้!
circut (signal) หันเปลี่ยนรูปแบบ L กับ T ก็ทำให้ output 0000 มาต่างกัน

จะเห็นว่า Leading กับ Trailing ที่มีผลต่อหัวเข้า(ผูกหัวเข้า)
แต่เราต้องการหันเปลี่ยนหัวเข้าไปได้ ต้องเลือกใช้ชุดต่อ:



} input ไม่เหลือหัวเข้า แต่เปลี่ยนรูปแบบ
ด้วยการเปลี่ยนหัวเข้า

How to... state Diagram \rightarrow circuit

Ex. วาต circuit จาก State Diagram นี้ ให้เป็น SR flip-flop

Step 1 มีพื้นที่ 3 State ต้องใช้ flip-flop พื้นที่ 2 อัน (1 อันเก็บได้ 2 state 7 บิต)

โดยเราราชกันเลย (100) ให้

A = 00
B = 01
C = 10

ให้มี 11 เรื่อง: ผู้แล่ 3 state

<ตอนนี้มีรู้จัก A, B, C เป็นแทนตัวบอร์ดซึ่งเป็น 2 bit \rightarrow 9 ตัว flip flop ก็จะ

Step 2 เขียน State table ต่อการลงที่น่องก่อ จากร A 7 ไปต่อที่ 7 บิต ... ฯ

Q	Q'		Z	
	x=0	x=1	x=0	x=1
A	A	B	0	0
B	C	B	1	0
C	A	B	0	0

เช่น Q = A กับ input 0 นั้น

(a) Diagram) ก็จะเข้าห้อง A

output บน 0 ก็จะเป็น 0 ในรูปนี้:

Q คือ Present State
(state ที่อยู่ตอนนี้)

Q' คือ Next State
(state ที่จะมาถึง)

Z คือ output
(output ที่ตอนนี้)

State Table จะก่อผลลัพธ์ 8 อย่าง

Present State \rightarrow Next State \rightarrow output

เมื่อกำหนดก่อ จากร A กับ input 0 วา รูปนี้ state ไหนต่อ output 000 วา ปริมาณนี้

(ดูจาก State Diagram ဝอมาเขียน)

Step 3 107 ไปเขียน Truth table ว่า: ให้ 7 บิต input แต่ x ตัวเดียว: ถ้า Q₁, ก: Q₂ ต้อง!

ก็จะมี input x ตัวเดียว แต่ Sequential circuit หรือ circuit ที่ต้องมี Data ในการผลิต output ด้วย... ดูที่นี่เราใช้ flip-flop 2 ตัว ซึ่งมี input (สัญญาณ) เหมือน

อีก 2 ตัว !! ※ ใจน้ำ 27 ก็จะบังกล่าวว่า วา state 7 ตัว: 7 บิต (เรา): flip flop ที่น้องๆ

<101 State Table ของไปรษณีย์ code ที่กำเนิดโดยต้อง แรกก่อน

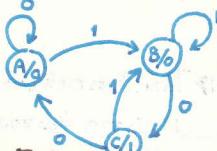
ให้ input คือ: output 1 ตัว (พูดง่ายๆ ก็คือไฟเส้นเดียว) เก็บ แสง: สูง ผ่าน Data 7 ตัว ต้อง 0 ก: 1 ให้แน่น แต่ Q กับ Q' จะต้อง 2 bit เลยต้องแยก 00 ก 01

Q₁, Q₂ \rightarrow Q'₁, Q'₂

Q	Q'		Z	
	x=0	x=1	x=0	x=1
00	00	00	0	0
01	10	01	1	0
10	00	01	0	0

► แล้วก็ 10 ตารางนี้ แปลงเป็น Truth Table !

* ก็ไม่พอ... ต้องหนีต่อไปอีกอย่าง หนึ่งอีกซัก ๑๙



Step 3 (ต่อ)

X	Q ₁ Q ₂	Q' ₁ Q' ₂	Z
0	0 0	0 0	0
0	0 1	1 0	1
0	1 0	0 0	0
0	1 1	X X	X
1	0 0	0 1	0
1	0 1	0 1	0
1	1 0	0 1	0
1	1 1	X X	X

► ได้มา 8 บรรทัดแล้ว แล้ว Truth Table ที่ซึ่งไม่เกิด
เรื่องของ SR flip-flop 7 จุด

- แม้ SR จะดีมากก็ตาม

S₁, R₁, S₂, R₂

Q₁ ได้จาก State Table
State Table ใหม่ที่เก็บ
Q₂ ก็ตาม
State Table ใหม่ที่เก็บ
Q₁ ก็ตาม State Table
มี 2 ชุด (x=0, x=1)

X	Q ₁ Q ₂	Q' ₁ Q' ₂	Z	S ₁ , R ₁ S ₂ , R ₂
0	0 0	0 0	0	0 X 0 X
0	0 1	1 0	1	1 0 0 1
0	1 0	0 0	0	0 1 0 X
0	1 1	X X	X	X X X X
1	0 0	0 1	0	0 X 1 0
1	0 1	0 1	0	0 X X 0
1	1 0	0 1	1	0 1 1 0
1	1 1	X X	X	X X X X

แล้ว 101
ได้ 5 หัวน้ำ
7 ปุ่มเขียง
สมการ
(ใช้ K-map)

step 4 เช่น K-map + สมการ

$$\begin{array}{c} x \otimes Q_2 \\ \text{---} \\ \boxed{0 \ 1 \ X \ 0} \\ \text{---} \\ 0 \ 0 \ X \ 0 \end{array}$$

$$\begin{array}{c} x \otimes Q_2 \\ \text{---} \\ \boxed{0 \ 1 \ X \ 0} \\ \text{---} \\ 0 \ 0 \ X \ 0 \end{array}$$

$$\begin{array}{c} x \otimes Q_2 \\ \text{---} \\ \boxed{X \ 0 \ X \ 1} \\ \text{---} \\ X \ X \ X \ 1 \end{array}$$

$$\begin{array}{c} x \otimes Q_2 \\ \text{---} \\ \boxed{0 \ 0 \ X \ 0} \\ \text{---} \\ 1 \ X \ X \ 1 \end{array}$$

$$\begin{array}{c} x \otimes Q_2 \\ \text{---} \\ \boxed{X \ 1 \ X \ X} \\ \text{---} \\ 0 \ 0 \ X \ 0 \end{array}$$

$$Z = \bar{X} Q_2$$

$$S_1 = \bar{X} Q_2$$

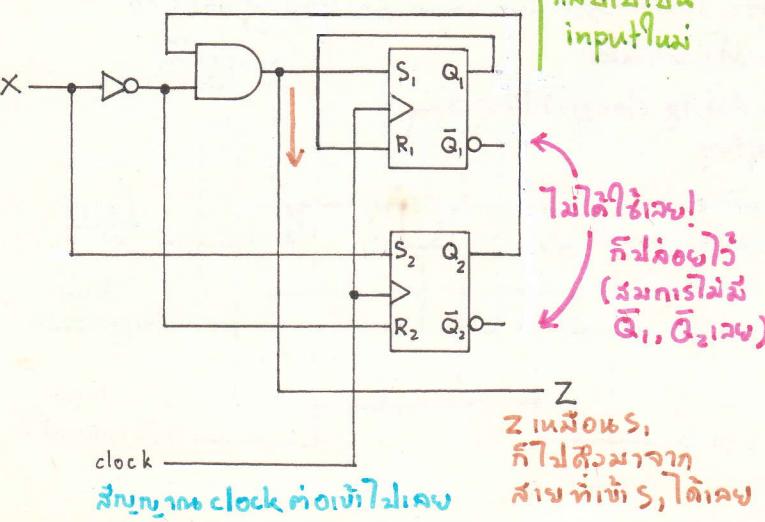
$$R_1 = Q_1$$

$$S_2 = X$$

$$R_2 = \bar{X}$$

step 5 ลัพธ์สมการแล้ว ให้เขียน circuit ไป

กับปุ่มไปปุ่ม
input ใหม่

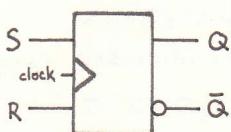


ไม่ได้ใช้!
ก็ยังคงไว้
(สมการไม่ใช่
 Q_1, Q_2 จริง)

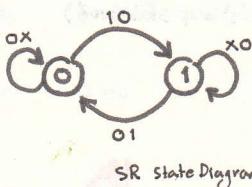
Z เนื่องจาก S₁
ก็ไปสู่มาจาก
ส่วนที่เข้า S₁ ได้เลย

4 type of flip-flop

I. SR flip-flop (set-reset)



S	R	C	Q
X	X	0	no change
0	0	1	no change
0	1	1	0
1	0	1	1
1	1	1	undefined



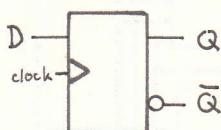
Set Reset

เมื่อ 2 input ดังนี้ set กับ reset

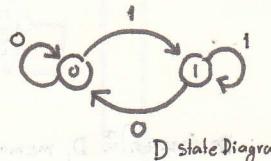
set = 1 output จะออก 1 ไม่ต้อง管 งาน กว่า
ที่管 reset = 1 output ต้อง管 ชั้นในเป็น 0
* ต้อง管 ชั้นใน管 ต้องต่อ ชั้นใน!



II. D flip-flop (Delay ของ Data)



D	C	Q
X	0	no change
0	1	0
1	1	1



Delay (Data)

D-flipflop ถ้า input มี值เดียว

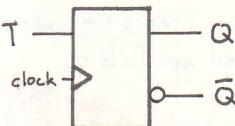
ถ้า input 1 Q จะออก 1

ถ้า input 0 Q จะออก 0

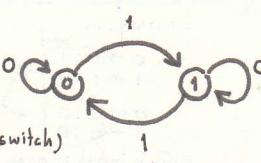
โง่ output ห้องจะ delay ไป 1 ชั้น clock

หมายความว่า เก็บ data ไว้แล้ว นานเข้า 1 รอบ
จะๆ Delay flip flop ลักษณะ คือ กดแล้วปล่อย
ก็ยังคงอยู่ กับ output ถ้าปล่อยเมื่อไหร่ output
คือ!

III. T flip-flop (Toggle)



T	C	Q
X	0	no change
0	1	no change
1	1	\bar{Q} (change/switch)

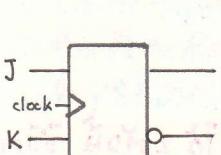


Toggle

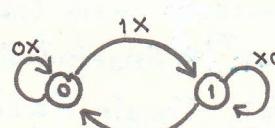
Toggle เมื่อ ตัวตั้งร่วม ถ้าเกิด toggle ขึ้น แล้ว
พอกลับกัน ให้เปลี่ยน ดังนั้น เมื่อ input = 1
จะเปลี่ยนไป output จากเดิม ที่เปลี่ยน ซึ่ง
ก็เดิมเป็น 0 จะเปลี่ยนเป็น 1, ก็เดิมเป็น 1
จะเปลี่ยนเป็น 0 จึงเปลี่ยนเป็น 0

แต่ถ้า input ทางเป็น 0 จะไม่เกิดการเปลี่ยน
อะไรเลย คงจะได้อยู่ ก็คงต้องตั้งต่อไป

IV. JK flip-flop



J	K	C	Q
X	X	0	no change
0	0	1	no change
0	1	1	0
1	0	1	1
1	1	1	\bar{Q} (change/switch)



JK

เมื่อ flip flop แห่งนี้ ที่เกิดจากการ input
SR กับ T งานผสานกัน ช่วงผลลัพธ์ได้ดี...

Flip flop SR ต้องมารัก input ให้ได้

โดยการ input 11 ให้จะได้ผลลัพธ์

Toggle input 1 ดังนั้น กับ Q จาก 0 → 1

แปลว่า $1 \rightarrow 0$ และ $0 \rightarrow 1$ ในส่วนที่ "JK" อยู่

J แทน Set, K แทน Reset

(กด JK พร้อมกัน = Toggle)

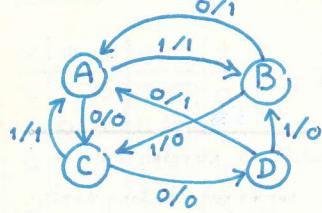
* ที่ State Diagram ของ 4 ตัวนี้
ทราบว่าใช้เวลา 2 ชั้นเวลา คือ 2 ตัวนั้นนัดเดียว!

Note

ในการใช้งาน D กับ JK ต้องหาที่สุด โดยเฉพาะ: JK (Don't care หรือ... สมการเล็ก)

ส่วน SR กับ T หลังจาก JK ดังนี้ ก็ต้องตามด้วย ก็ไม่ต้องมีคนนี้ อีกจะเป็น:
JK ตัวเดียวแล้ว ต้องสูงชั้นตัวเอง 2 ตัวนั้นนัดเดียว!

Ex. \Rightarrow circuit \Rightarrow State Diagram \Rightarrow D-flip flop , IIA: T-flip flop



Step 1 กົນຄອດໃຫຍວ່າ flip-flop ນີ້: code ລະບົບ State

1D 9² flip flop 2 ตัว
 $(2 \text{ ตัว})^2 = 4$ State 万多ตัว

1) A = 00
B = 01
C = 10
D = 11

step2 狹化 State Table

<u>Q</u>	<u>Q'</u>		<u>Z</u>	
	<u>x=0</u>	<u>x=1</u>	<u>x=0</u>	<u>x=1</u>
A	C	B	0	1
B	A	C	1	0
C	D	A	0	1
D	A	B	1	0

ໄປລືບນໍາເນີນ code ທີ່ນີ້ໄວ້

Q'	Q	Z	
	$x = 0$	$x = 1$	
0 0	1 0	0 1	0 1
0 1	0 0	1 0	1 0
1 0	1 1	0 0	0 1
1 1	0 0	0 1	1 0

step 3

X	Q ₁	Q ₂	Q' ₁	Q' ₂	Z	D ₁	T ₂
0	0	0	1	0	0	1	0
0	0	1	0	0	1	0	1
0	1	0	1	1	0	1	1
0	1	1	0	0	1	0	1
1	0	0	0	1	1	0	1
1	0	1	1	0	0	1	1
1	1	0	0	0	1	0	0
1	1	1	0	1	0	0	0

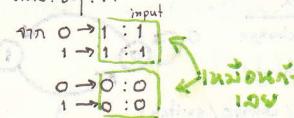
$$Q_1 \xrightarrow{D_1} Q'_1$$

▷ ពីរនេះនឹង D_1 memory digit ឱ្យស្នើសុំ (Q_1) T_2 memory digit ឱ្យស្នើសុំ (Q_2)



Delay (Data)

๑๖๙



ຈະເປັນວ່າ input ມາຕາລ
State ນີ້ແກ່ $\text{P}(r_i) = \frac{1}{n}$ ເພື່ອ r_i ດີເລີມ
ດີເລີມ copy Q ກຳ $\| \text{ກບມາຈຸນ } Q$

Toggle

ເມື່ອກົດວິທີ state ຈະປະສົບນີ້ເນື້ອ input ລວມ
ເຮົາກີ່ຈຳລັງ... ກໍາລັກງານ \rightarrow 0 ນັ້ນ 0 \rightarrow
ສູງສູງ ແລ້ວ ນອກນິ້ນິ້ສູງ 0

Tip จะเห็นว่าตอนนี้เขียน Truth Table อย่างไร... Delay n: Toggle ทำให้เวลาขยับ
มากมาย เพราะ: ให้ร่องตัวนี้เป็น input แล้วต้องเดินทาง ไกล Delay ก็จะ
แล้ว Q' ตัวหนัง (ถ้าไม่ต้องดูเอง), Toggle ก็ต้องแล้วว่า ก่อ Q กะ Q' ต่างกันกว่า 1 ก้าวเมื่อตอน
กันกว่า 0 100 เพราะ: แบบนี้ เวลาทำ ก่อ โจทย์ไม่กำหนด แนวโน้ม Q จะไป 2 หัวนี้ ซึ่งคง:
ว่ายังไง: ๘๘

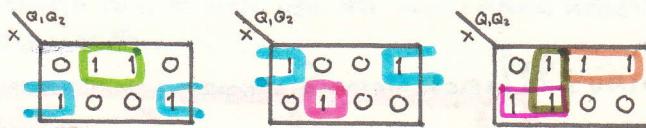


ส่วน JK จะเป็นส่วนผนวกของ SR และ T แต่ JK จะตีกับ SR มากกว่า

ໃນຕີ: don't care ເປດ: ສາກສາຍ! (ກັບໂປ່ງ State Diagram ວອງ JK ຈົດ)

▶ សំគាល់តាមវិធីរាល់ State Table ដែលបង្ហាញថា ការសម្រេចនៃស្ថាបន្ទាន់ ត្រូវបានរៀបចំជាព័ត៌មាន Truth Table ទៅក្នុងលក្ខណៈ (ព័ត៌មាននេះ ដោយស្នើសុំនៅលើ input ទាំងអស់ x, q_1, q_2) ទៅនឹង x នូវ ទំនួរ នៅក្នុង K-map ដែលត្រូវបានរៀបចំឡើង។

step 4 K-map และ สมการ



$$Z = \bar{X}Q_2 + X\bar{Q}_2$$

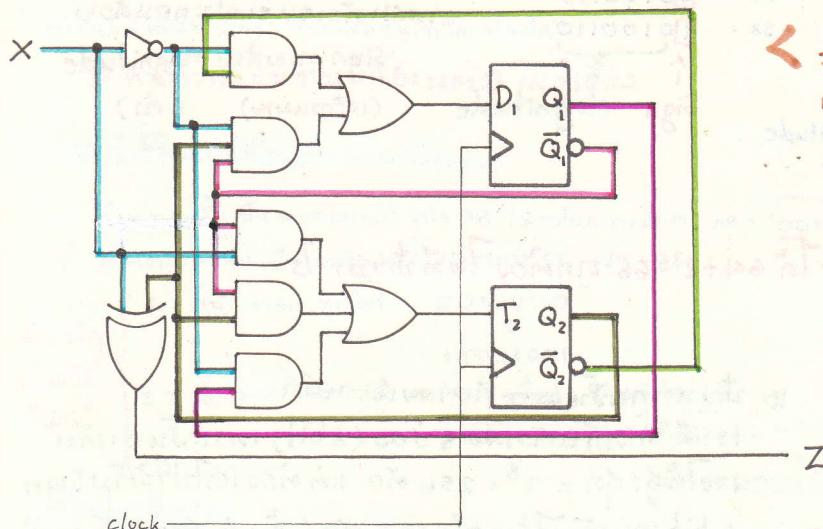
$$D_1 = \bar{X}\bar{Q}_2 + \bar{X}\bar{Q}_1 Q_2$$

$$T_2 = X\bar{Q}_1 + \bar{Q}_1 Q_2 + \bar{X}Q_1$$

step 4.1 กลับสมการให้เป็นรูปดังนี้

$$Z = X \oplus Q_2 \leftarrow \text{เปลี่ยนเป็น XOR ลง gate ที่ 3 ต่อ}$$

Step 5



วง circuit ที่ 4 อยู่ในลักษณะ
แบบขั้นบันได เช่น ชากะบก

\bar{Q} อย่างไร Q จะมีสี invertor (not) ไม่

กรณี \bar{Q} มีสีของวงจร flip flop อยู่เบื้องหลัง!
อย่าไปลืม!



从 state Diagram → circuit

1. กำหนด State สอง: ห้ามอน กะปะ: เก็บ flip-flop
2. เขียน State Table (จาก State Diagram)
3. แปลง State Table เป็น Truth Table
→ สอง flip-flop อยู่ทาง Q ที่ \bar{Q}
4. ทำ K-map ที่: ขั้นบันได
5. วาด circuit ... จบ



SR Flip-Flops → รูปที่ 10: ^

comment: ไม่ต้อง ห้องเดียว แต่... เก็บรัก !?

reply: ก็ต้อง 0. ห้องเดียว ก็จะหัวเราะ กัน



แก้ไขที่

การทำเรื่องนี้ให้เร็ว ๆ ดังนี้
รูป State Diagram ของ flip-flop
ที่ 4 ดังนี้คือ

(ต้องเข้าใจ properties (คุณสมบัติ)
ของ flip-flop ที่ 4 ถึงได้ ก็ไม่ต้อง
ดู... รูป State Diagram ของ
ความเห็นใจได้เลย)

note: ใช้เวลาหนึ่งวัน ในการ check
pattern ระหว่างการเขียน State Table ไป
"shift Register" มาก่อนแล้ว
ซึ่งทำให้ ต้อง circuit ร่วมกันสอง

(ค่า 51)

Sign-Magnitude & 2's Complement

ร: บันเลข ก 939 นกอน พิองเตอร์ 2 แบบ

- สอนคุณครัว เราจะบันกอก computer ว่า เลขตอ 38 มันก็จะเท่ากับ 100110_2
แต่ถ้าเราจะบันกอก -38 ว่า ? ... -100110_2 สมมติไม่ใช้ไบต์默 รูปซักคมนา: 0 ก 1 (เกร็งหมาย ไว้สีเขียว)
เราซึ่งมี 7 bit แรกใน การบันกอก เลขตัวนี้ ติดลบไว้เป็นล ดูต่อไปนี้

Sign

0 แทน Int⁺ (บวกบวก)
1 แทน Int⁻ (บวกตกลง)

- สำนัก 38 ก 9: 7/2

$$+38 = \boxed{0}0100110$$

$$-38 = \boxed{1}0100110$$

↑
Sign Magnitude

ร: บันนี่เรียกว่า SM (sign-Magnitude)

เพรา: หัวลง 9: ปะ: กอนด้วง

Sign ตามด้วย Magnitude
(เกร็งหมาย) (ต่อ)

Ex. บันกอก 59 กับ 7 แบบ sign-Magnitude

$$\begin{array}{r} 00111011 \\ + 00000111 \\ \hline 01000010 \end{array}$$

$$\leftarrow 7 \times 64 + 2 = 66 \text{ ถูกต้อง } 7 \text{ บวก } 2 \text{ บวก } 7$$

$$\begin{array}{r} 01100001 \\ + 00100011 \\ \hline 10000100 \end{array}$$

↓
sign(+) 1 90 (+) ดันบันกอกได้ (-)

* สรุนการบวก กับการลบได้ดังนี้เลย:
กติกาบันกอกบวก (mem ให้บวก กดบันกอก ก็ได้บวก)
ก็ได้บวก (บวก) ซึ่งทำให้บวกได้บวกบวก
แต่ไม่สามารถ overflow หรือเสียบก่อ...

Underflow

แต่ยังไงก็ตาม... บวกห้ามบวกลบ (0 แต่ 2 ตัวก่อน) เวลา หงวนะบ่น R.M. ก็ยังมีบวกบานห่วงตัวที่
ช่วยมากน้ำ ทราบ: บีบ ดู: บันกอก || บันกอกนี้ไม่มีบวกบานห่วง แต่สำหรับตอน มันมีบวกบาน... ดู -
เฉพาะ ตอน "ลบ" เคย

Ex. $42 + (-28)$ → ดูตัวบวกบานห่วง
บวกบานห่วง

$$\begin{array}{r} 00101010 \\ + 10011100 \\ \hline 00001110 \end{array}$$

↓
Sign ต้องรู้เองว่า
เป็น + เพรา: $42 > 28$

► บวกบานห่วงนี้ ให้ดัง เลขบวกบานห่วง
เราบวก ซึ่งบันกอกบวก 4 บิต (4bit) เพรา: วันเราเก็บ
เลขได้สูงสุด $= 2^8 = 256$ ตัว แต่ต้อง เบี้ยไป 9 ตัวเก็บ Sign
1 bit เลขเก็บได้ $= 2^1 = 128$ ตัว (หัวแต่ -127 \rightarrow 127)

แต่ $95 + 35 = 130$ มันเกิน 127 แล้ว... เกิดเหตุ
แบบนี้เราเรียกว่า **Overflow**

วิธีแก้ bug ของ overflow ผังเดินทางด้านซ้ายที่นี่
เก็บบวกบานห่วง!
9 ตัว float เก็บบวกบานห่วง double แทนอ:

► บวกบานห่วง ... บวกบานห่วงติด ตอนบันกอกบานห่วง อ: บวกบานห่วง เจ็บ วันก่อน 0
ร: บันกอกนี้ต้อง circuit ต: บวกแต่ๆ แทน sign ก็ไม่แน่ใจอักว่า 0:0000:0000

→ ตอน ADDER อาจใช้มีบวกบานห่วง แต่ตอน SUBTRACTOR
พั่งนันให้แน่นอน (แล้ว ยังมีกรอก 1 ตัว 0 ตัว เคย int⁺ ไปบวก
กับ int⁻ ห้ามกากว่าหัวบันกอก และ กรอก int⁻ บวก int⁺ และ
... ฯลฯ บวกบานห่วง circuit ต้องมันลวย แบบน้ำ
ซึ่งไม่ใช่บวกบานห่วงแล้ว!)

ເນັ້ນຈາກວ່າ: ບະນຸຍາກວ່າ ສາມາດກຳລັງ ຕົວມາຈົບວິທີ ດີກວະບະນຸຍາກວ່າ ນັ້ນໄວ້ໃນລົ້ນ ທີ່ຈຶ່ງຫຼືດໍາ
ຮຽນມັນຕື່ອງໄວ້ວ່າ ຈະ ບວກ ນັ້ນໂລກ ລວມ ໄດ້ຕົ້ນຕົກລວມ ກີ່ຄົວດີໂລກທາງ ອື່ສີແລ້ວ circuit ADDER ລວມຕົກລວມໄດ້
ກີ່ເພື່ອຍຸ່ນພວມແລ້ວ...

2's Complement

↳ ឧំនុយពេលវេលា ឬ 2's Complement ត្រួតតើខាងក្រោម ថា 2C (ក្បច្ច) មែនគឺជានុយតិចបែន្នឹងសំណង់របៀប ការអនុញ្ញាតការងារទាំងនេះ...

- ▶ จัดการกับ sign +,- ได้ง่าย , ต่อ และ จัดการ circuit ง่าย
 - ▶ แต่ยังคงมีวิธีเชื่อม (เล็กๆ) ซึ่งอาจไม่สนใจครั้งหนึ่งนักกัน
 - ▶ ตอนต่อไป เรียนรู้แบบนี้ (บวกตัว) ไม่ใช่เรื่อง

How to... change to 2c

1. ก้าวที่ 1 เลขห้ามเป็น บวกบัญชีแล้ว เช่น 38 ให้แปลงแบบบวกก็ 38 = 00100110 นั่นก็คือบวก
 2. ก้าวที่ 2 เลขห้ามเป็นบวกแล้วต้องลบให้ได้ เช่น 2C ใช้ -38

มี 2 step { 2.1 ให้ invert ทุก bit 00100110
 ↓
 11011001
เริ่บก็ต้อง "2C" { 2.2 101 เก็บที่ invert และบวก increment 1 (บวกเพิ่ม 1)
 = 11011010 ← 1 ตัวสุดท้าย "-38" ให้บวก 2C, อ่านไม่รู้เรื่อง
 มันเก็บ 0 ไว้ ก: 38

Note

NOTE: ถ้าแม้ว่าในระบบ 2C จะไม่สามารถตัดสินใจได้ว่า เลขผู้นับนั้น เป็นบวกหรือ (-) ก็ตาม bit แรกจะ (0 คือบวก, 1 คือลบ) เช่นเดียวกับระบบ

Sign-Magnitude 124... ដែលត្រូវការពារទៅ 20

សំគិតនាយករដ្ឋមន្ត្រីនរោងចន្ទ 2C ដែលបានការពារឡើង

ก้าวที่ 2C ถูกหักน้ำหนัก 11011010 (-38) $-2C \rightarrow 00100110$ (38) แปลงว่า บันลือค่า = -38

$1XXXXXXX - 2C \rightarrow 0XXXXXXX$

↑ 8-bit ↑ 8-bit

ຫົວໜ້າມດໍາເກີນຫົວດ້ານຂວາ
ທີ່ມີຄົດຄົດລວມ



รับ 2'C แก้ปัญหาทุกอย่าง S.M. ได้แก้ไข
ยกเว้นเรื่อง Overflow ซึ่งเป็นเรื่องที่ก้ามเมื่อขนาด
ของ mem. ที่ใช้เกินไป แก้ได้โดยใช้ตัวเดียวๆ กัน

-Tip 2'c สั้นๆ

ກារແປສ້ນຫະ 2 C ໂພນເຮືອວ່າ ອິນທາ "1" ຕັກ
ດູບໜາງສູດ ເຊັ່ນ

00100110
M96

เจอกันนั่นถือแล้วชุดทางเข้าของ "๑" ตัวนั้น
101|| เกต์นั้น|| แล้ว... invert มันจะ: จะได้
→ ห้องที่ไม่มีหลอดบุ้ง

110110110
^
1 អេឡាច្រក
ឯកសារនៃគណន៍

00100110
2C7n²
11011010

ចំណាំ
ជីវិត
នៃបាន

การบวก และ ลบ เศษที่ 2C นั้น จะทำผ่านการ ADD อย่างเดียว

กรอบ 1 A + B (โดยที่ที่ A และ B เป็นเลขบวกบวกหรือลบบวก)

Ex. $35 + 27$

$$\begin{array}{r} 00100011 \\ + 00011011 \\ \hline 00111110 \end{array}$$

$\hookrightarrow = 62$ ถูกต้อง

Ex. $35 + (-27)$

step1 นำตัว -27 มาบวก

$$\begin{array}{r} 00100011 \\ + 11100101 \\ \hline 000001000 \end{array}$$

$27 = 00011011$

$2C \rightarrow 11100101$

\times 1 ตัว 8

overflow
เกิดกรณีที่ต้องบวกกันไปแล้ว

$$35 + (-27) = 8$$
 ถูกต้อง

Ex. $(-35) + 27$

step1 นำตัว -35 มาบวก

$$\begin{array}{r} 35 = 00100011 \\ 2C \rightarrow 11011101 \\ + 000011011 \\ \hline 111111000 \end{array}$$

ต้องยกเว้นตัว 0 ไว้ไม่รีทิร์น

ตัดเศษ (เพรา: bit แรกเป็น 1)
อย่างรุ่งๆ ตัดเศษเท่าไหร่ ก็ 2C ได้

$$111111000 \text{ เท่าไหร่ } 2C \text{ ได้ } 00001000$$

$$\downarrow \text{มีตัว } = 8 \therefore \text{ผลลัพธ์ } = -8$$

Ex. $(-35) + (-27)$

$$\begin{array}{r} 11011101 \\ + 11100101 \\ \hline \text{X} @ 1000010 \end{array}$$

overflow
เกิดกรณีที่ต้องบวกกันไปแล้ว

ต้องยกเว้นเศษเท่าไหร่ ก็ 2C ได้ ... ดูตัว กดลงมา 2C ตัว

$$11000010 \text{ เท่าไหร่ } 2C \text{ ได้ } 00111110$$

$$\downarrow \text{มีตัว } = 62$$

$\therefore \text{ผลลัพธ์ } = -62$

กรอบ 2 A - B (โดยที่ที่ A และ B เป็นตัวบวกบวกและตัดเศษหนอนต้ม)

→ การคำนวณ A - B ให้ A กับ B ที่ $2C$ (ซึ่ง $A \geq B$) แล้ว เอา ไม้เข้ารูปแบบ $A + (-B)$ แทน
ขั้นตอนนี้: โดย... → ถ้า B เป็น (+) อยู่ ก็ หัก $A + (-B)$
→ ถ้า B เป็น (-) อยู่ ก็ หัก $A + B$

Ex. $35 - 27$

$$= 35 + (-27)$$

$$\begin{array}{r} 00100011 \\ + 11100101 \\ \hline \text{X} @ 00001000 \end{array}$$

\downarrow ตัดเศษ!

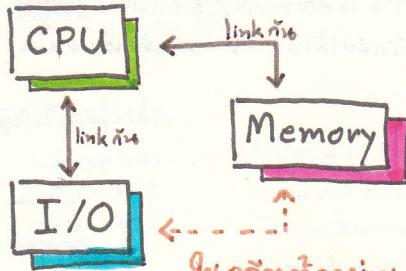
7 ตัว 8



สรุปวิธีทำ $2C$

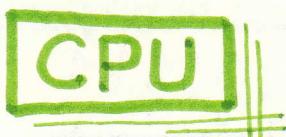
1. A กับ B ไม่ว่า + หรือ - ให้เต็ม "บวก" กันเสริม
2. นำตัว $2C$ ซึ่ง A กับ B ให้ได้ตามรูปแบบ
3. Overflow สามารถตัดทิ้งได้ ถ้า เอา C_n กับ C_{n-1} มา $C_n \oplus C_{n-1}$ ให้ = 0 (ถ้า 0 ออกเข็น ต้องมี overflow)

Computer Architecture



Communication

สำหรับ กองเมือง ลี CPU เป็นแหล่งรวม ทำท่อไป จัดตั้ง ก็ต้องใช้คนพาก CPU



Management Organization Decision making เป็นส่วนที่ทำให้เกิดกระบวนการ ตัวอย่าง เป็นส่วนสังกัด

Analysis (calculation, computation) เป็นส่วนที่เกี่ยวกับการคำนวณ Computation (logical decision) กับค่า Logic

▷ Sequential circuit
- clock
- memory

▷ combinational circuit
- ADDER
- SUBTRACTOR
- พอกวนของนักลงทุน

เป็นส่วนรับข้อมูลมา เช่น 0:101 A + B ต้องมาเข้ากับ Register ตัวหน้า B ตัวหน้า แล้วค่อยส่งขึ้นไปให้ ADDER (ตัวอย่าง parameter เอามาไว้รับต่อ:)

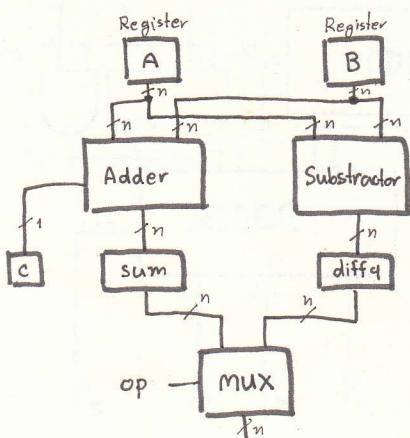
△ ส่วนประมวลผล
ของ CPU

ALU

เป็น circuit ที่เกี่ยวกับการคิดเลข เช่น บวกลบ ADDER, Substracter, Multiplier, Divider

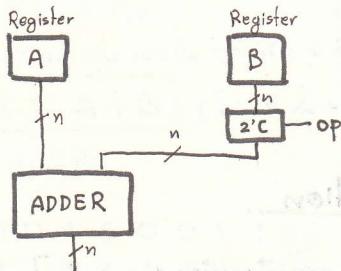
Adder & Substractor

แบบที่ 2 คือ circuit แบบ Sign-Magnitude และ 2's Complement



Sign-Magnitude

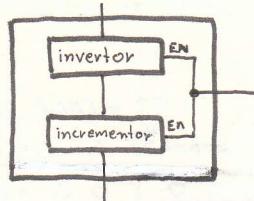
▷ circuit นี้ ก. ลบ ก. เมื่อต้องหัน A,B ให้เป็น Adder ต้องหัน A,B
▷ ถ้า Register "Sum" รอดิจิตน้อย, ซึ่ง A,B โน้นให้ Substractor ต้องหันไปด้านหลัง เก็บใน Register "diff"
▷ หักห้ามได้รับการหักห้าม บวก 1 รหัส ให้เป็น operation (op) กับ 0 หักห้าม (-1)
▷ ซึ่งบันทึกกับหัวต่อไปแต่เงื่อนไขให้ + เม้น 0 - เม้น 1 แล้วใช้ MUX
▷ เม้นต้องเลือกหัวต่อไป sum หรือ diff หักห้าม



2's complement

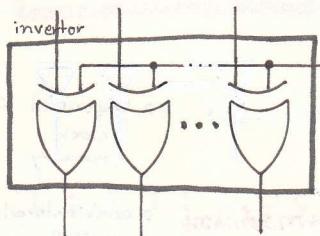
► เนื่องจาก รูปแบบ ADDER (จริงๆ มีก็ circuit subtractor ให้ลองกันดู) ก่อนอื่นไปที่ส่วนตัว A โอนเข้าไปห้องเลข ส่วน B ให้ไปอยู่ OP ว่า เท่ากันมากหรือ ถ้าลง ตัวบวก ($OP = 0$) 2's C ไม่ต้องห้ามอะไร ส่วน B ผ่านไปเข้า ADDER เลย แต่ถ้า ลง OP:1 (เท่ากันต่ำ) ห้ามห้ามใช้ -B ห้อง แล้วต้องใช้借位

แบบกล่อง 2's complement



หลักการ 2C คือ invert ตามด้วย add 1

อย่างแรกเราต้อง ตัวบวก invert แล้ว ตามด้วยการ + เพิ่ม 1



► รูปแบบของ inverter ไม่ได้มีดังนี้ \square แต่ใช้ XOR หาก เพราะต้องการ ให้ได้ค่า 0 ถ้า EN=0 และ 1 ถ้า EN=1

* สำหรับ XOR... ถ้ารับ enable เท่ากับ 0 output ก็จะตามตามเดิม แต่ถ้า enable เป็น 1 output ก็จะ เป็น invert ของตัวเดิม

สำหรับ incrementor ก็สามารถใช้ circuit ADDER มาใช้ได้ (ให้เลื่อนหัวเป็น A ส่วน B เป็น 00000001 (ก็จะ加เพิ่ม 1 ต่อ))

► หัวตอนหัวจริงๆ แล้ว แต่บวกเพิ่ม 1 ตัว ต้องเอา circuit ADDER หัวหัวมานำไปบวกกับเปลี่ยนอยู่ เขาจะบด circuit ใหม่ซึ่งมี...

จากเดิม $A+B$ โดยหัวทศ = 0 และ B มีตัว 1 เราต้องเปลี่ยนเป็น $A+B$ โดยหัวทศ = 0 แต่หัวทศ (C_{in}) = 1 หากซึ่งก็ต้องหัวบวกเพิ่ม 1 ให้ต่อ... แต่ แบบนี้จะต้องหัวบวกเพิ่ม 1 ต่อ

original ADDER (one cell)

$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = AB + AC_{in} + BC_{in}$$

► เผื่องหัวก้านหัวแล้วว่า B ก็ต้องมีเป็น 0

เราหัวทศ 0 แทนที่ B ทุกหัวในส่วนการ

หัวที่ต้องหัวสมการให้มีสีเขียว

incrementer โดยเฉพาะ:

<จะเห็นว่า circuit หลักจะยัง:

incrementer (middle cell)

$$S = A \oplus C_{in}$$

$$C_{out} = AC_{in}$$

rightmost cell

$$S = A$$

$$C_{out} = A$$

แบบ สำหรับ cell increment

หัวแรก(หัวก้านหัว) เราต้อง

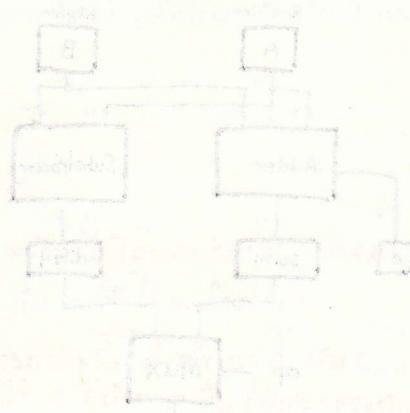
ยังคงต้องหัวบวก เพราะ: เวลาหัวทศ C_{in}

ของหัวนี้ต้อง 1 แทน หัวต้องการ

หาก $C_{in}=1$ จะไปหัวสมการหัว

หัวเปลี่ยน A หัวเพิ่ม

(เช่นกันหัวทศ = 0 หัวบวกหัวทศ = 1)



Multiplication

$$\begin{array}{r} A \quad (\text{m bit}) \\ \times B \quad (\text{n bit}) \\ \hline (A \times B) \quad (\text{m+n bit}) \end{array}$$

การคูณเลข m bit กับ n bit ต้องเตรียมช่องเก็บข้อมูล (Register) ก่อนหน้า m+n bit ให้ไว้เก็บ

Sign-Magnitude

$$\begin{array}{r} A \quad (\text{m bit}) \\ \times B \quad (\text{n bit}) \\ \hline (\text{m+n-1 bit}) \end{array}$$

▶ สำหรับรูปแบบ S.M. ที่นี่
ต้องบวกกับ m+n-1 สอง
เพื่อ: bit ด้านขวาของหัวข้อ
ที่ 2 หัวข้อเป็น sign ไม่ต้อง
ต้องมาติด



การคูณ (multiplication)
ต้องเอา magnitude มาติดหัวข้อนี้
Sign ไว้ต้องเอาไว้... ถ้าเป็นรูปแบบ 2's
Complement แปลงกลับเร็ว (+) ให้หมดกัน และ
ต้องลบตามเดิม

รูปวงจร circuit Multiplication กัน (วงจรตามภาราย ข้อมูล)

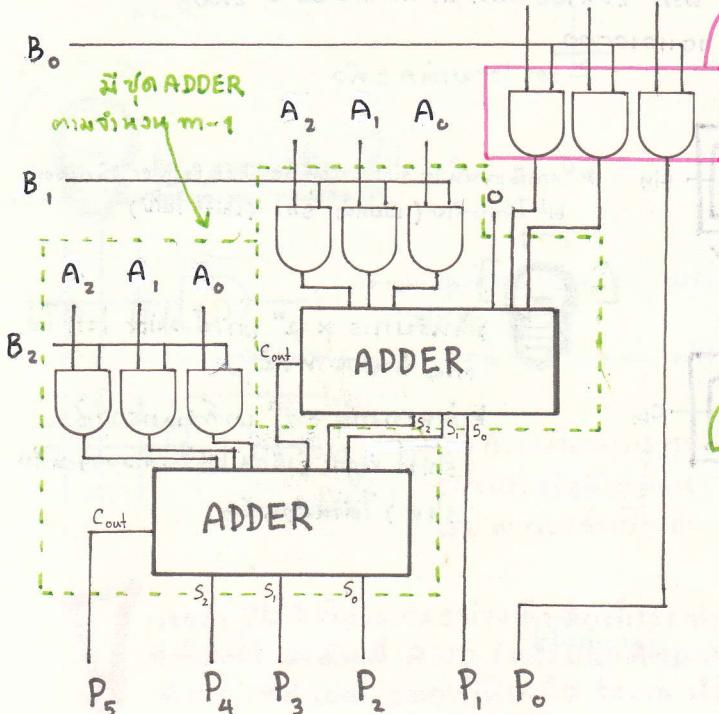
$$\begin{array}{r} A_4 \quad A_3 \quad A_2 \quad A_1 \quad A_0 \\ 1 \quad 0 \quad 1 \quad 1 \quad 0 \times \\ \times \quad B_2 \quad B_1 \quad B_0 \\ \hline 0 \quad 0 \quad 0 \quad 0 \quad 0 \\ , \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \\ 1 \quad 0 \quad 1 \quad 1 \quad 0 \\ \hline 1 \quad 0 \quad 1 \quad 1 \quad 0 \\ P_7 \quad P_6 \quad P_5 \quad P_4 \quad P_3 \quad P_2 \quad P_1 \quad P_0 \end{array}$$

- ▶ เวลา ดู ดูแล เดชะ: ถูกใจให้เสร็จก่อนแล้วค่อย เดา ผลหักบวกมา บวกกัน เป็นเดียวจบใช่ไหม?
แต่หันกลับมีนี่ชักบุญเสียนั้น ดูดู. ห้ามเก็บพอย: บวกเลขน้อยๆ ตัวใดก็ได้ก็ได้ (ก็ Full Adder
ของเรามาทำได้แล้ว A+B 102 ถือ 10 A+B+C+D... ก็ซึ่งนี่เนี่ยต้องกัน
▶ เพื่อ: บนนั้นนี่เวลาต่อ circuit เราจะ: เดา A มาคูณกับ B หัก: หลัก บวกกันให้เสร็จ แล้วกัน
ดูกันหลักด้วย และ บวกเพิ่มหัวของเดา บนนั้นนี่ไปเรื่อยๆ

↑ สืบต่อ: 5 bit x 3 bit

ได้ 8 bit ผล

ผู้ดูแล circuit ของ Multiplier 3x3 bit



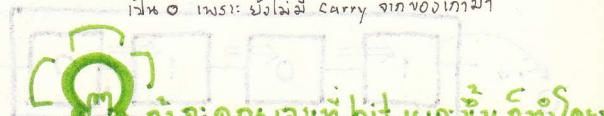
Array of AND ตามจำนวน bit ของ A
(ก หัว)

▶ การคูณกันต้องเม้น 1 หัวคู่ ผล จึง 100 กว่า 1 (ใช้ 1 กับ 0 แทน AND)

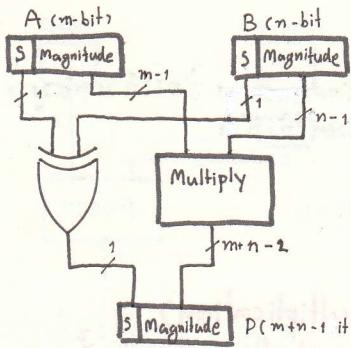
- bit ลูกทับหัวไป เช่น กิตติบุนย์ได้เรียบ เพราะ ไม่ต้องบวกกัน
จะได้แล้ว

- กับแล้วเดา 2 ตัว ส่วนไป บวก กับ A×B หัวต่อไป
แล้วส่วน ดูไป หุ่งห่ำ ขอ กัน ไม่ ต้อง หัวต่อ หัวต่อ หัวต่อ

* ในการบวกบัฟเฟอร์ เรา กำหนดให้ digit ที่ 3, 1 เม้น 0. เพื่อ: บังไงล่ะ carry จาก หัว เก่า มา



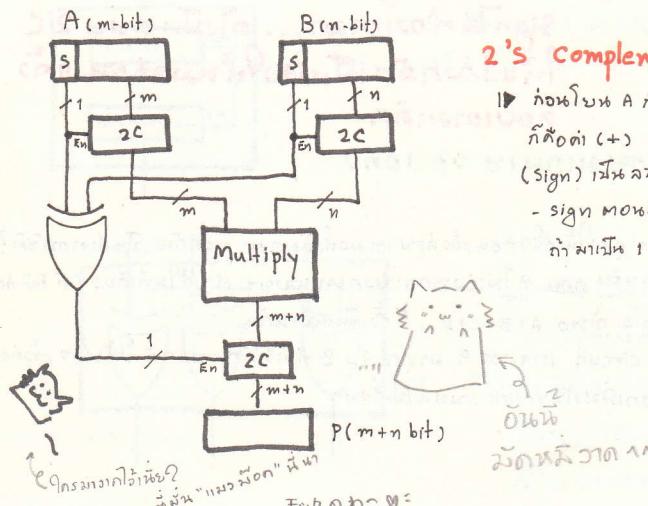
- ก้าว: ดูแล เดา หัวที่ bit เดอะ นี่ ก็ ทำ ใจ
1. เพิ่ม AND gate ตามจำนวน bit
(แล้วเพิ่มขนาด ADDER 2)
2. เพิ่มชุดของ ADDER ที่มีขนาดหัวที่



Sign-Magnitude

- ▶ การคูณ สองตัว magnitude ผลิต ก็จะขึ้นตัวที่ $n-1$ ตัว ให้ยกเว้น Multiply ส่วน bit แรก ที่เป็น Sign ก็ต้องดูด้วย
 - ถ้า Sign เหล่านั้นเป็น (+,+,-,-) sign ตัวที่ห้ามคูณ + (0)
 - ถ้า Sign ต่างกัน (+,-,-) sign ตัวที่ห้ามคูณ - (1)

ตัวห่างกัน 00101 เลข 101 XOR มาได้



2's complement

- ▶ ก้อนบิท A กับ B เข้า Multiplier เราต้องทำให้มีบิทใน magnitude ก่อนซึ่ง ก็ต้อง (+) เลย 1 ตัว circuit 2C หวานไว้ แล้วต้อง Enabler ว่า: ทำงานตอนที่ bit แรก (Sign) เป็น 0 ลง (1)
 - sign ทางด้านหัวหอย ก็ต้องเหมือน S.M. ถ้า ทางเป็น 0 (บวก) ก็ไม่ต้อง 2C ถ้า เป็น 1 (ลบ) ก็ 2C กลับนี้ magnitude ก็ได้ساภากลายเป็นตัวลบ

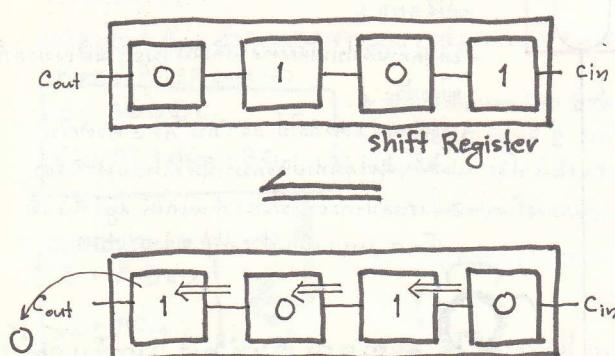


Multiplier สำหรับ $A \times B$ เมื่อ B เป็น 2^n (อย่างเช่น พลก 10, 1000 พลกที่มีบิท "1" หนึ่งตัว และทางด้าน "0" ก็เปลี่ยนไปหัวก็ได้)

▷ เมื่อเขียนกันเลขฐาน 10 พลก 25×100 ก็ 10×25 ห้อ 00 = 2500

ก้าวเดียว พลก 1011010×1000 ก็จะได้ = 1011010000
"0" ตัวเดียว

↑ ตัว "0" มากัน 3 ตัว



▶ ในการนี้ เราใช้ circuit ชื่อ Shift Register เพื่อเลื่อน bit ไปทางซ้าย (เรียกว่า Shl (shift left))



พื้นฐานการ $\times 2^n$ ใช้ shift left 7 ตัว ต้องหัวหน้า หกตัว

▶ สำหรับการ $\div 2^n$ เรา ก็จะมากรดีด shift right (เลื่อน bit ไปทางขวา นึง shr) 7 ตัวเหมือนกัน

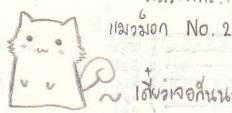
* สำหรับ Divider ห้ามหาร 0: ! ... ช่วงกันไป

ROM

Hardware Programming

เนื้อ Hardware ที่เราสามารถโปรแกรม circuit ได้ มี 3 ชั้นๆ...

- Rom (Read only Memory)
- PLA (Programmable Logic Array)
- PAL (Programmable Array Logic)



ตัวเรียนภาษา..
กิตติภูมิ No. 2

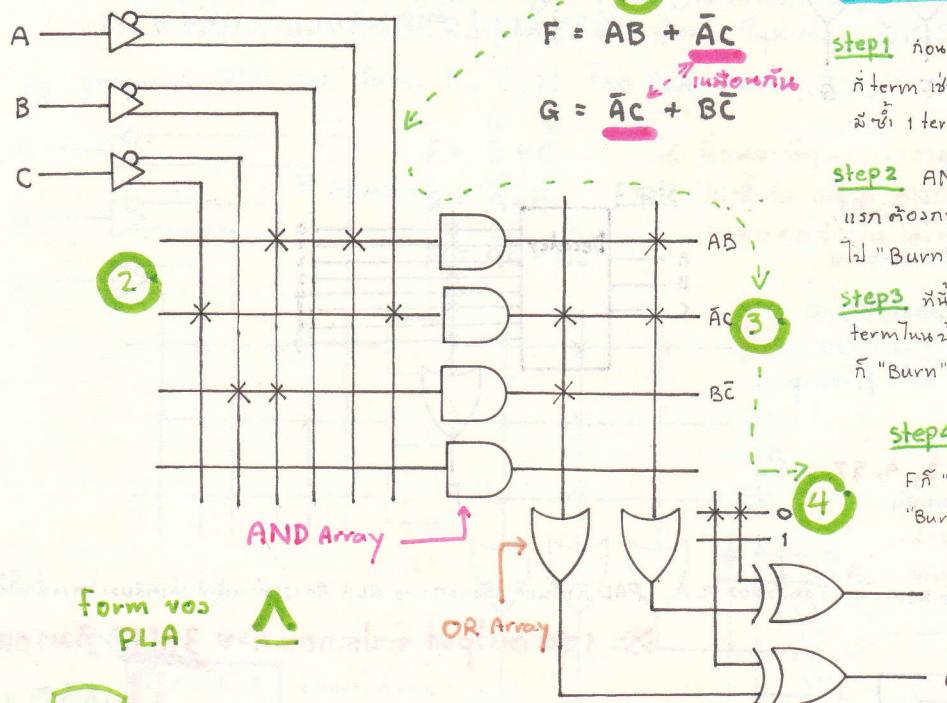
ใช้เวลาอ่านหนังสือมาก...

จัดการตัวเอง!



PLA

ใช้สุด... เพราะ เราสามารถโปรแกรมได้ทุกที่เลย



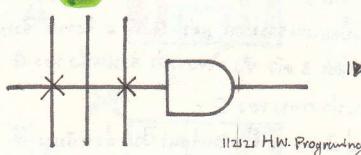
หลักการ

step1 ก่อนเข้ามาดู... ถูว่า สมการเรา มีกี่บันทัด
กี่ term แล้วให้บันทัดที่มี กี่บันทัด 4 term แต่
บันทัด 1 term จะไม่เหลือ 3 term.

step2 AND 1 หัว 1 หก 1 term, 1 หก 1 term-
แรกต้องการ A กับ B และ AND กับ เวลา ก'
ไป "Burn" (กากบาท X) ที่เว้น A กับ B ๆ:

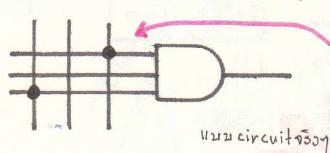
step3 หันก็จะดูว่า func. ในห้องกิจ
term ไหนมีไว้ เช่น F ต้องการ AB กะ $\bar{A}C$
ก็ "Burn" ตรงๆ AB กับ $\bar{A}C$ ๆ:

step4 เลือกว่า: 101 F หรือ \bar{F} , ก็ 101
F ก็ "Burn" 0 ก็ 101 invert ก'
"Burn" 1 เป็นอันจบ!



▶ในทางชีววิศว์ที่เรียก "บันทัด" ... รูป form ของ H.W. Programming จะเขียนแบบย่อ (เพื่อให้เข้า
ส่ายไฟ ดูง่าย ง่าย)

- การ Burn หมายความว่า ให้ตั้งส่ายไฟ เช่น ก็ Burn ไปเบี้ย gate



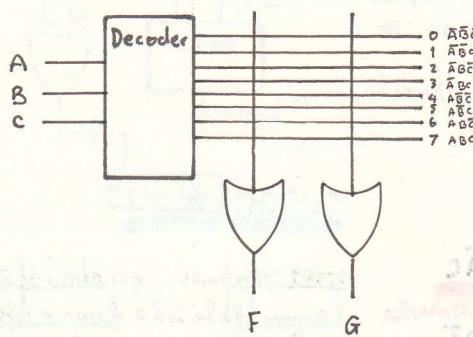
ดำเนินการต่อ
circuit จริงกันเมื่อ
เรา • ทราบไปฟรี:



การทำ PLA ต้อง; วันอย่างเช่นต้องถ้า ก็ Step1 แล้วพบว่า มี กี่บันทัด n term
ที่ต้องใช้ แต่ต้นมี AND (ก็ 1 หก AND Array ต่อ) นั่นบวกว่า ก ห้า... ใจกว้าง
ก็การ reduce สมการให้เหลือ term น้อยลง ทำให้เก็บตัว AND

ROM

→ เองาทำ ROM ให้คิดง่ายๆ ว่า เราทำ PLA ในส่วนที่ ③ อยู่ ดังนั้นการแทนทุก term ที่มีในแล้ว



▷ เราจึงรู้ได้ว่า สาย 0-7 แทน term 0-7 ตามที่เราลงไว้ใน K-map หรือ Truth table ด้านล่างนี้เลย

◀ การทำ ROM ต้องใส่ sum of product
 $F = \bar{A} + \bar{B}\bar{C}$ ที่อยู่ในรูป minterm แบบนี้!

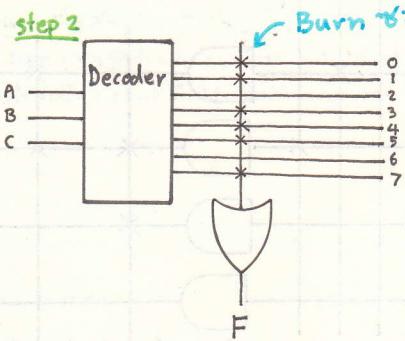
$$\bar{A} + \bar{B}\bar{C} = 2$$

$$\text{Ex. } F = C + A\bar{B}\bar{C} + AC$$

Step 1 จัดให้มี minterm โอล กซ์
boolean Algebra นี้ก็ใช้ K-map ช่วย

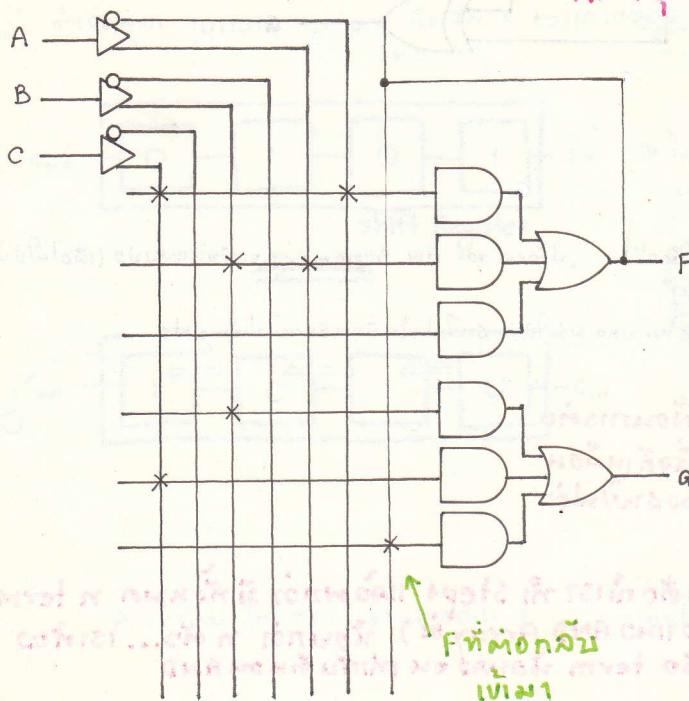
A\BC	0	1	2	3	4	5	6	7
0	1	1	1	0	1	0	1	0
1	1	0	1	1	0	1	0	1
2	1	1	0	1	0	1	0	1
3	0	1	1	1	0	1	0	1
4	1	0	1	0	1	0	1	0
5	0	1	0	1	1	0	1	0
6	1	0	0	1	0	1	0	1
7	0	1	0	0	1	0	1	0

→ ผู้ที่บันทึก 0, 1, 3, 4, 5, 7 ลงใน ROM
Burn ช่องที่ ROM ตามนั้น

PAL

→ ภายนอก ROM เป็นตรรกะของ PLA, PAL ก็เป็นแต่ครึ่ง แห่ง PLA ดัง เราทำหน้าที่แต่เรียบ term ที่ใช้พอ

* 1 ชุด output จะประกอบด้วย 3 AND กับ 1 OR กัน



Ex.
 $F = \bar{A}C + AB$
 $G = B + C + \bar{A}C + AB$

▷ F ก็ต้องบันทึกลง ROM แต่ G มี 4 term ซึ่งเรา
จะ AND แล้ว 3 ตัว ซึ่งต้องระบุว่า ส่วนไหนของ G
เมื่อส่วนไหนจะเป็น output ของ F
- ซึ่งไปสัง F ที่ เป็น output ไปแล้วก็จะไม่ใช่
ใน ังกอร์

Note.

PAL ห่วงมากใช้กับจอภาพ
เพื่อ: รหัสสีในดูดอง 3 ตัว คือ
Red Blue Green (RGB Mode)

PAL + ROM = PLA

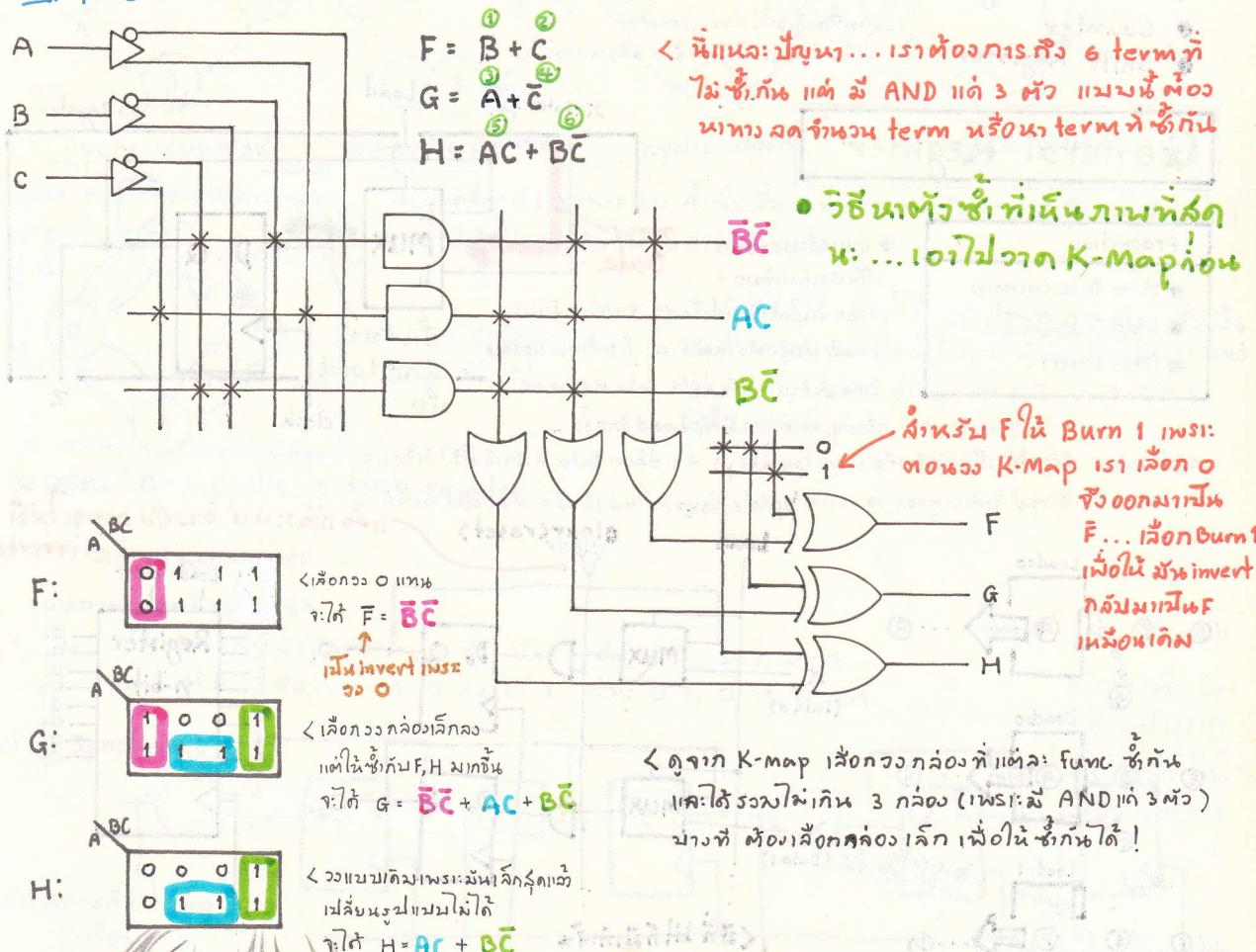
ข้อมูลให้ program
ที่ใช้จริงนั้น
(ก่อน AND Array)

ข้อมูลให้ program
ที่ใช้จริงนั้น
(ก่อนที่ OR)

โปรแกรมได้แก่ สองผู้เยี่ยม
(ฝรั่งเศสภาษาไทย)

- !... แต่ถ้า PLA จะโปรแกรมได้ทั้ง 2 แบบ แต่เวลาทำจริงๆ ROM จะง่ายสุด เพราะเราสามารถไปที่ K-map และ Burn ตามเงื่อนไขของ K-map ก็เสร็จแล้ว แต่ PLA (และ PAL) จะเสียบุคลาเรื่อง AND gate ไม่พอ
 - สำหรับ PLA ก็ต้องนา term ที่ซ้ำกันแล้วต่อ กันสักหน่อย (เนื่องจากต้อง 0 ใน K-map แล้วห้ามเป็น Maxterm)
 - ส่วน PLA ต้องนา term ที่ซ้ำกันให้มากที่สุด (อาจต้องเลือก 0 ใน K-map แล้วห้ามเป็น Maxterm)

Pb. program PLA จาก func. F, G, H โดย PLA เมื่อเท่านั้น 3×3 (AND 3 ตัว, OR 3 ตัว)



Light Shining!!

Register

リーチッタ

Def: smallest memory unit of computer
เป็นหน่วยความจำที่ถือว่าเล็กที่สุด (ตัว... และ flip-flop)
คือ หน่วย memory นี้, flip-flop มีหน่วยได้ 1 bit คือ:
ในเวลาการทำงาน ช่วง มีหน่วย 1 bit ที่ไว้เก็บงาน

- ▶ Register เป็นห้องเก็บข้อมูล ขนาดของ Register ส่วนมากจะมี 2^n (เช่น 8, 32, 64) ช่องเก็บหน่วย 2^3 (ปี 2009) Windows 7 เก็บได้ถึง 64 bit แล้ว ...! (โดย... หมายความ มากกว่า)
- ▶ Register จะประกอบด้วย flip-flop ที่อยู่ภายใน ... Register ที่เก็บได้ n -bit ก็หมายความว่า รวม flip-flop n ตัว (flip-flop 1 ตัว เก็บได้ 1 bit ไป)

3 type of Register

- General Register : basic คือ แบบธรรมดานะกันนะ
- Counter : มากซักอัน ก็ได้เรียกว่า count
- Shift Register : เอาไว้ลาก่อนหน้า ไม่ใช่ทางเดียว หรือทางขวา

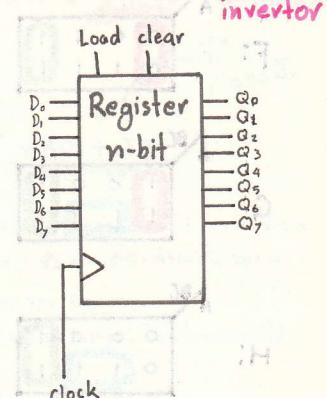
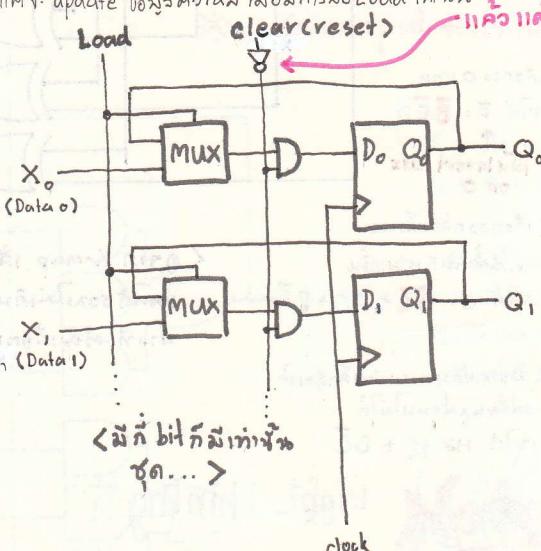
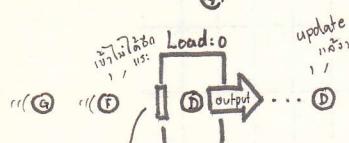
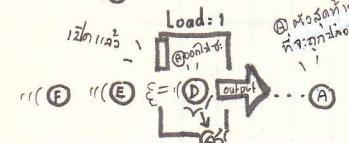
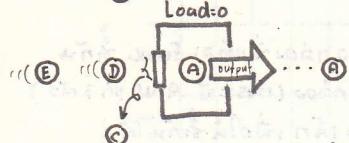
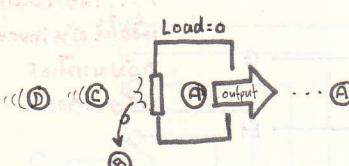
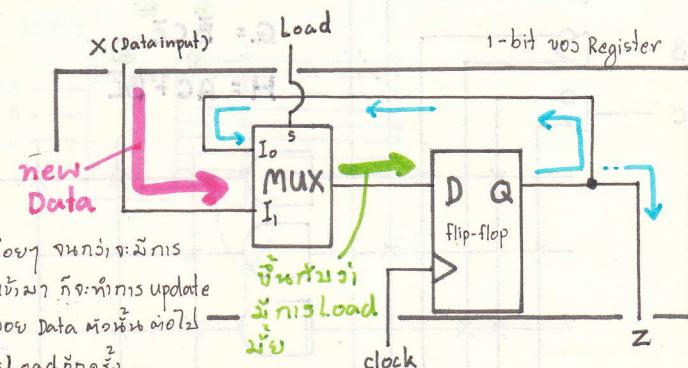
General Register

Properties
• Store Data (memory)
• Load Data
• Clear (reset)

▶ แบบธรรมดาก็คือ เป็นห้องเก็บข้อมูล + ปลดล็อกข้อมูลที่เก็บไว้ได้เรื่องๆ จนกว่าจะมีการ Load ข้อมูลตัวใหม่เข้ามา ก็จะทำการ update Data ที่เก็บไว้ ดังนั้น Data ห้องเดียวต้องรับ เรื่องๆ จนกว่าจะมีการ Load อีกครั้ง

สรุปง่ายๆ : Register เป็นห้องเดียวเก็บข้อมูล (memory) ก็คือ ปลดล็อก Data ที่มีหนึ่งตัวไว้เรื่องๆ

เมื่อ input รับเข้ามาตลอดเวลา ทำง่ายๆ update ข้อมูลตัวใหม่ เลื่อนลาการะบบ Load เอาหนึ่ง



※ กรณี Load เป็นการ update Data ใหม่ ก็ต้อง clear เป็นการรีเซ็ต Data ทุกตัวไว้เป็น 0 ๆ AND gate ลับกันเป็น flip-flop

↑ มี input + output ตลอดเวลา
ข้อดี : ไม่ต้องล็อกในรูป

Counter

Properties

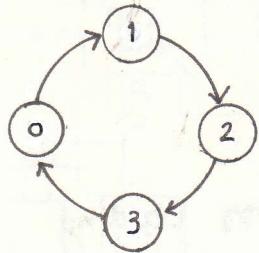
- Store Data (memory)
- Load Data (initialize)
- Clear (reset)
- clock (count)

► Register that goes a pre-defined sequence of state.

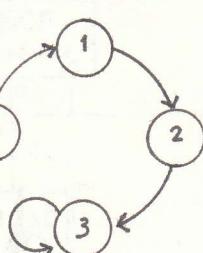
นี่คือ Register ที่เราสามารถตั้งชุดของห้องตัวต่อไปได้ (next State ไม่ต้องต่อ) ซึ่งหมายความว่า output ที่ไม่เกี่ยวกับตัวต่อ ก็ต้อง input มา...

เราใช้ counter เมื่อที่：“นับ” 0: ไปอย่างเดียว ไม่ว่าจะเมื่อไหร่ก็ตาม หรือ “ลดลง”

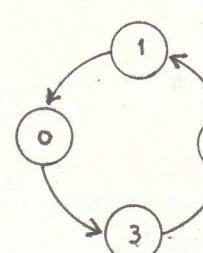
counter ก็มันตามสัญญาณ clock จะเรียกว่า “Synchronous”



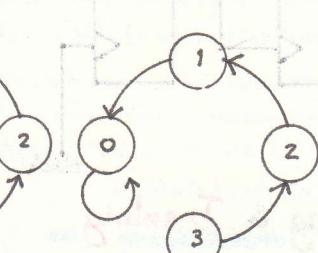
cyclic counter



non-cyclic counter

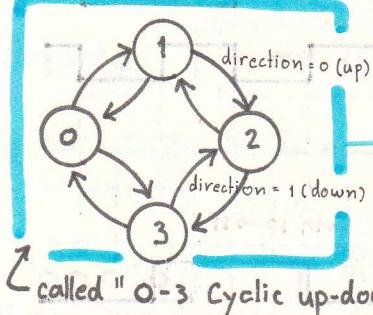


cyclic-down



non-cyclic-down

- counter ก็มีผลลัพธ์แบบ ก้าบหลังขึ้น หนบลง, หนบลง ก็จะมากกว่า สุด แล้วจะวนกลับ วนไปเรื่อยๆ จนกว่าจะถูกตัดขาด



ในการปฎิบัติจริงฯ
ใช้ห้องนี้

called "0-3 Cyclic up-down counter"

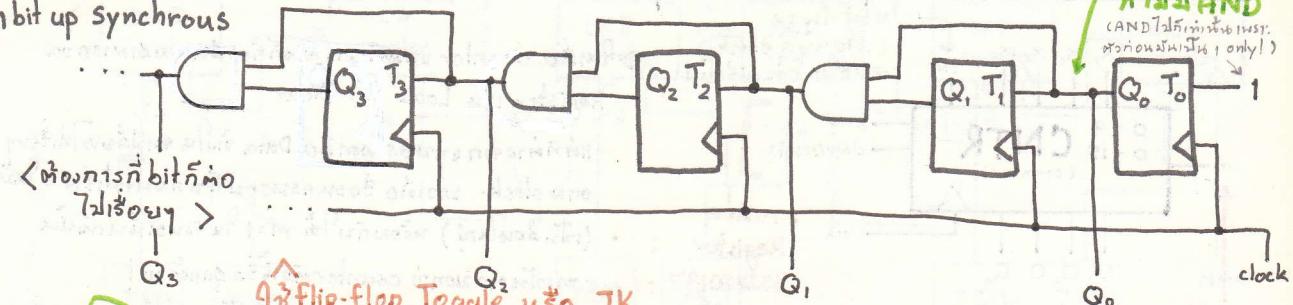
► เวลาที่ไปใช้ เราจะใช้ up-down counter ซึ่งเป็นห้องที่เราสามารถกำหนดได้ว่า จะให้หนบหรือลง หนบลง โดยเลือกผ่านตัวต่อ direction (เมื่อ input ตัวนี้)
ถ้าให้ direction = 0 มันจะขึ้น ถ้าให้ direction = 1 มันจะลง

ขนาดของ Counter

แบบมาตรฐานมีอยู่ 2 ชนิด

1. แบบ Binary ต้องบิตต์ตั้งแต่ 0 ถึง $2^n - 1$ เช่น 0-3, 0-7, 0-15...
2. แบบ BCD ต้องบิตต์ตั้งแต่ 0 ถึง $10^n - 1$ เช่น 0-9, 0-99, 0-999

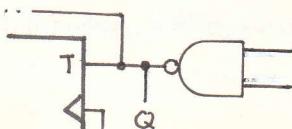
nbit up Synchronous



ใช้ flip-flop Toggle หรือ JK



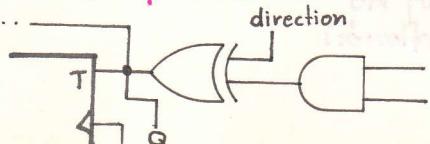
ถ้าอยากรู้ว่า down counter ทำได้ ไม่ใช่ AND gate → NAND gate
ทุกครั้งจะ (ในกรณี invert ไม่ต้อง)

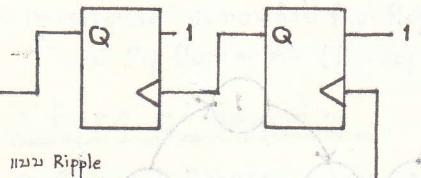
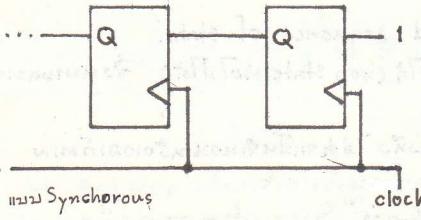


ตัวประกอบสำคัญ
ก้าวเดียว AND

(AND ไม่ต้องเพิ่ม NOT, แต่ต้องเพิ่ม NOT only!)

ก้าวเดียวการต่อ เส้นทาง up-down (direction)
ก็ใช้ XOR มาซึ่ง... ต่อไปนี้จะอธิบาย AND
ก่อนเข้า flip-flop ห้องต่อไป ก็ต้อง... ว่าจะต่อ





น่องใจการต่อแบบ Synchronous (flip-flop ทุกตัวใช้สัญญาณจาก clock ร่วมกัน) ยังสั้นกว่าต่อ circuit ต่อแบบเรียกว่าแบบ **Ripple** ซึ่งสัญญาณ clock นั้นจะมาจาก Q ของ flip-flop ตัวก่อนหน้ามัน!

▷ สังเกตดูว่า flip-flop ห้องใด ที่ทำงานเสร็จแล้ว Q จะเป็น **clock** เที่ยงนาฬิกา
จากแบบ Synchronous ที่ต้องว่า clock นั้นมาอยู่แล้ว ตอนแรก Q เรายังไม่รู้
ในแบบ Ripple ก็ต้องว่า Q มาอยู่แล้ว (always = 1) ที่เหลือก็รอแต่
สัญญาณ clock 罣罣...

Leading & Trealing

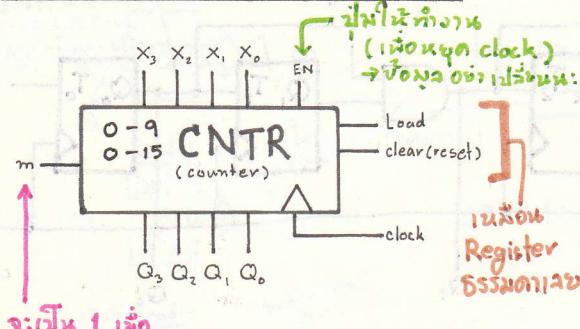
เรื่อง counter ที่มีปั๊มน้ำอย่างต่อเนื่องกับ clock แบบ Leading edge และ trealing edge ... มันจะทำให้ flip-flop เมล็ดข้าวไปพร้อมกัน ซึ่งหมายความว่า counter ห้องเดียว ก็จะไปใช้กับ Leading อย่างนั้น "down",
ก็ต้องไปใช้กับ trealing อย่างนั้นแบบ "up"

- สำหรับ Leading edge ผลของ clock 1 ก็จะต้องมีผลลัพธ์ไป
กับ flip-flop ห้องเดียวต่ออย่าง (ด้วย 1 ชิ้น invertor) ราก circuit
count up ก็จะลงกากบาทเมื่อ count down

- สำหรับ trealing edge นั่นไงได้รับผลลัพธ์ไป ก็ต้องมีผลลัพธ์ที่ต่อ
เนื่องกันไป

❖ ต้องยกให้ Leading edge นี้ count up
กิจกรรม invertor จะต้องรักษาอยู่ใน clock
ที่ กันแม่น!

หน้าตาของ Counter กันๆ กันๆ (แบบกล่องๆ ๆ)



จะเป็น 1 เพื่อ

บังคับตัวบ่ง

รวมของ counter

ที่นี่ 1 ก็จะ

กลับค่า

Overflow คือ

▷ ในแบบ counter แบบนี้ ... บังคับต้องมีความสามารถของ Register เช่น Load หรือ clear

- แต่ที่ต่างจากแบบธรรมชาติ Data ที่บังคับจะเปลี่ยนไปเรื่อยๆ
ตาม clock แล้วเมื่อวิ่งวนกรอบรอบ ก็คือ ก้อนไฟฟ้าเร็ว 0 ใหม่
(เรื่นร่องในกฎ) หลังกันไปที่ t=1 ในส่วนนี้นั่นเห็นด้วย

- ภาระตัวบ่งบอกว่า counter ต้องนั่งรอกลับค่า!

- ลักษณะ properties : Load กรณี: บังคับจะบังคับ Register ต่อตันนี้
แต่ตอนนี้จะต้องกิจกรรมต่อ!

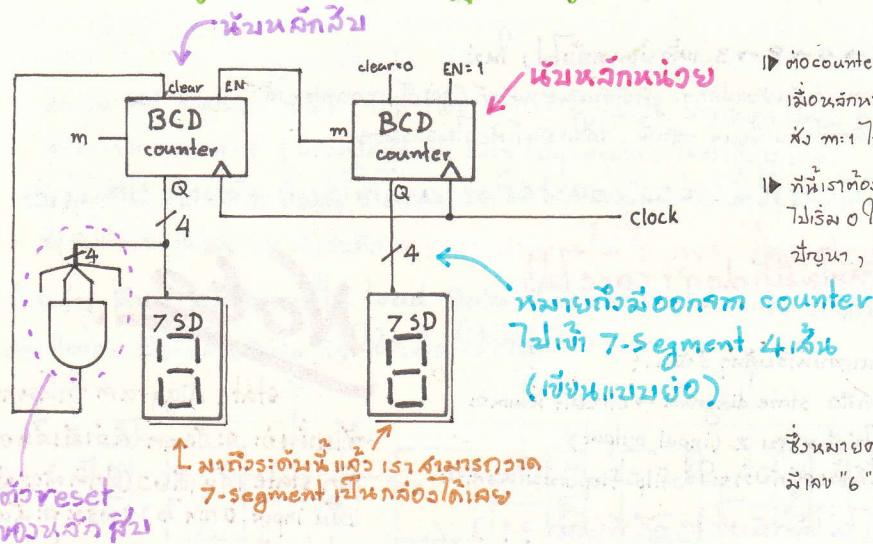
- นี่ Q 00 ก็เป็น เลข (ฐาน 2) ที่ counter นั้นอยู่ (เมล็ดข้าวเรื่อยๆ)



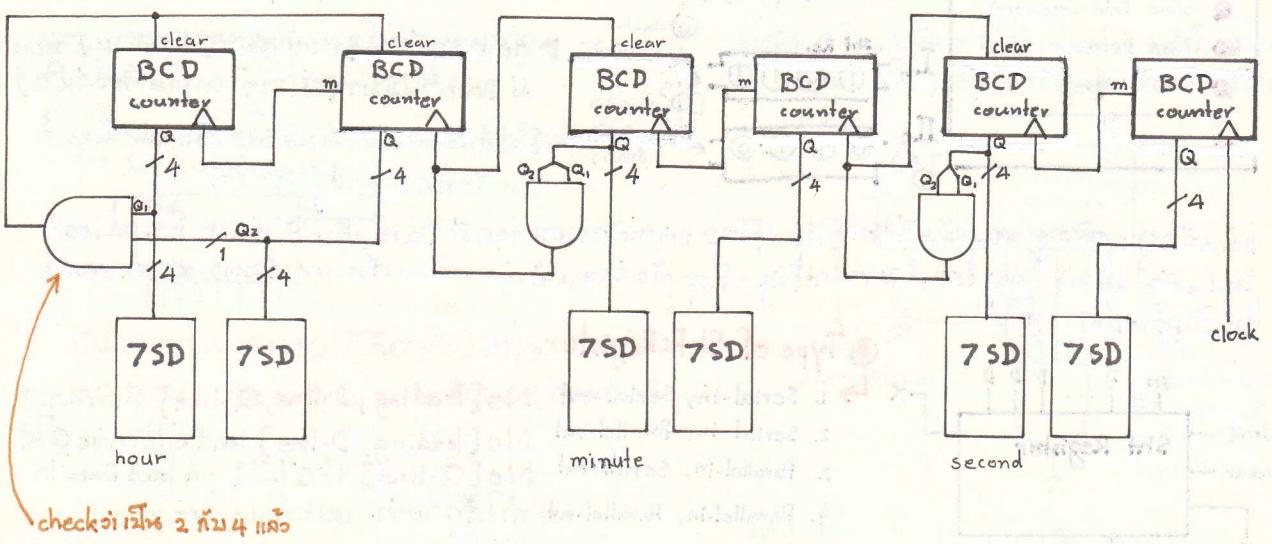
BCD counter... with digital

counter អត្ថរុកិចិត្តនឹងរួប "រូប" ថែរីករាយរវាង មនុសាតាំង ស្រីក និងភ្លើក ។

7 នៃ: នៅលើវត្ថុ BCD ទាំងនេះ Binary នេះ: អាជីវការត្រូវបានដោះស្រាយឡើងទៅ ដើម្បីបាន 10 បី (នៅចិត្តរបស់យើង អាជីវការត្រូវបាន 2 ?)



- เลข "6" เมื่อแปลงเป็น binary จะได้ [0110] ... กดต่อ เราจะ reset (clear) เมื่อ Q_1, Q_2 เป็น "1" ผู้ช่วยชี้มือ: เรา กด 101 AND gate มาตักเข้าไป 101 Q_1, Q_2 จะเท่ากับ 00 Q_0, Q_3 เว้นห้องไว้แล้ว กดต่อ เข้า ตัว clear ของ จัตุรัสบันก์ ปืนรันจะยัง
 - สี่นั้นรันตอนที่ติดอยู่ว่า "เราตักแต่ 2 ตัว แล้ว ก้ามันปืนเลิกดันบัน 7 (0111)" ร์ ปั้นก็ reset - เมื่อปั้นกันอีก 1 รอบ... ปั้นก็ใช้ แต่ จูเตอร์ นี่ หันปั้นบันย่าง เดียววน: มันหันไปอีก 1 รอบ กะ 6 ก็ reset แล้ว (กองใจ เลข 2 ตัวแต่ 0-5 ถูก , ไม่มีตัวไหน กับ Q_1, Q_2 เป็น 1 แล้ว)



check ว่า เป็น 2 กับ 4 แค่

reset (2 ສົ່ວ Q, ດັບງານເສັງ, 4 ສົ່ວ Q, ດັບງານເສັງ)