Dig. logic.

⊕ Literals : จำนวน Gate inputs

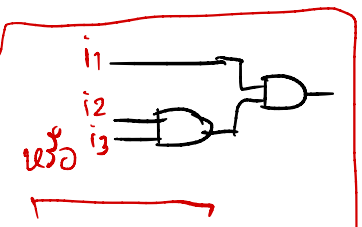
Symbols of Logic Gates

⊕ Gates take time! (Delay)

bubble (1→0, 0→1)

1. Source (var) , Output
2. NOT (') ↔ X'
3. AND (•) ↔ $X \cdot Y$
4. OR (+) ↔ $X+Y$

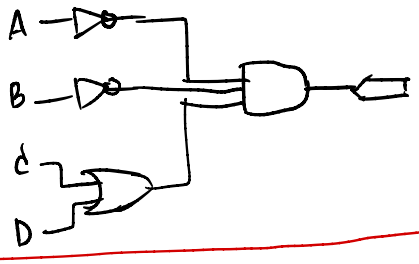
Buffer



e.g. $Z = A' \cdot B' \cdot (C+D)$

T_2

T_1



5. Not And (NAND) ↔ $(X \cdot Y)'$, $\overline{X \cdot Y}$
6. Not Or (NOR) ↔ $(X+Y)'$, $\overline{X+Y}$
7. Exclusive OR (XOR) ↔ , $(x \cdot y)' \cdot (x+y)$ $\equiv (x'+y') \cdot (x+y)$

Black box

└ $x \oplus y$

8. Exclusive NOR (XNOR) ↔ $(x \oplus y)'$, $\overline{x \oplus y}$

Goals :

- Reduce – literals
  - Gates
  - Lvl. of Gates (จำนวนจาก ▷ ไป ▷)
  - Fan-in (จำนวน inp. ที่ gate รับได้ เช่น =D = 2, =D = 3)

} Optimization

Yields

Faster, Min. Delay, Costs, Heat, Pow. Consumption, Size

จาก Transistor Wait และ Cable Length

Variables : A, B, C

Literals : นับทั้งหมด $(A+AB)(AA')$, # = 5

Canonical Form (รูปมาตรฐาน : ไม่ใช่รูปที่ลดทอนได้)
(SOP)

- Sum of Products

eg.

A	B	C	→	F	F'	
0	0	0		0	1	$A'B'C'$ } ไม่เอา (discard)
0	0	1		0	1	$A'B'C$
0	1	0		0	1	$A'BC'$
0	1	1		1	0	→ $A'BC$ } อย่างใดอย่างหนึ่ง (OR, +)
1	0	0		1	0	$AB'C'$
1	0	1		1	0	$AB'C$
1	1	0		1	0	ABC'
1	1	1		1	0	ABC

desired output.

$\therefore F = A'BC + ABC' + AB'C + ABC' + ABC$ ⇝ ใช้ De Morgan's

$F' = A'B'C' + A'B'C + A'BC'$

$F = (F')' = (A+B+C)(A+B+C')(A+B'+C)$ → "Product of Sums"

หรือคิดจาก Minterms (จำนวนของพจน์ที่ทำให้ F=1 , m) ในกรณีนี้ คือ เทอมเดียว (products)

→ $F(A,B,C) = m_3 + m_4 + m_5 + m_6 + m_7$

- Product of Sums.

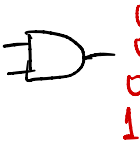
↳ คิดจาก Max terms (# ของพจน์ที่ทำให้ F=0, M) → sums

⇝ ใช้ Boolean Operations ช่วยในการ Minimization (การลดรูป)

ข้อเสีย "Timing glitches" ที่เกิดจากการเปลี่ยนแปลง input : ⇝ steady state ในขณะที่ยังไม่ถูกต้อง ↳ "Transient error"

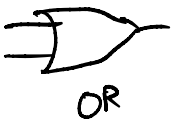
Alternative Gate (duality)

L	L	→ L
L	H	→ L
H	L	→ L
H	H	→ H

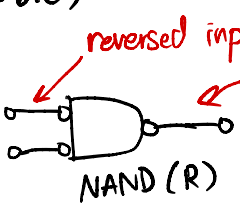
≡ 

(negative) ≡ 

Negative Logic (bubble)



OR



NAND (R)

เรียกว่า "active low"

$A+B \equiv \overline{(A+B)} \equiv \overline{A}\cdot\overline{B}$

Incompletly-specified func.

eg. BCD (Binary Coded Decimal)

$2^4 = 16$

	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	1	0
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	1	0	0
4	0	1	0	0	0	1	0	1
5	0	1	0	1	0	1	1	0
6	0	1	1	0	0	1	1	1
7	0	1	1	1	1	0	0	0
8	1	0	0	0	1	0	0	1
9	1	0	0	1	1	0	1	0
	1010 – 1111				b (Don't care / DC / ~~Disconnected~~)			

offset of W (ตัวที่ทำให้ W=0)

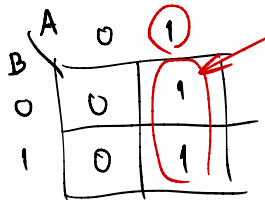
onset of W (ตัวที่ทำให้ W=1)

(For positive Op. : OR with DC.
For negative Op. : AND with DC.

Karnaugh Map (K-Map)

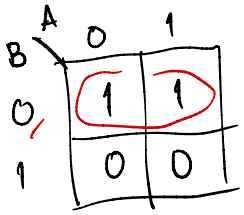
* เลือกกลุ่ม, ช่องข้างกัน alternate แค่ 1, เทียบเท่า boolean cube ในรูป adjacency
** นำไปทำ Sum of Product (minterms) / $\sum^{N} m(0,1,\ldots,N)$
*** ดูตัวที่คงที่ = นำมาใช้, ตัวไม่คงที่สลับทั้งได้ไม่มีผล (invariant)
* ยิ่งกรอบใหญ่ ยิ่งดี

eg.



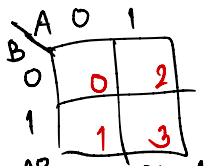
ต้องเรียก "Prime Implicant"

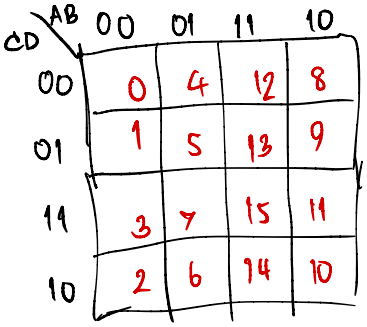
F = A

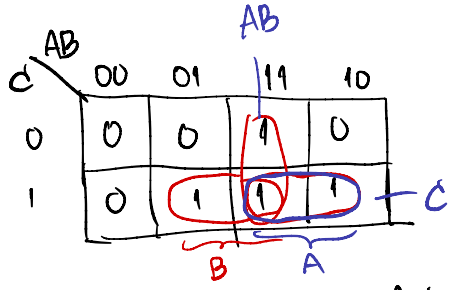


G = $\bar{B}$

ผัง TT:



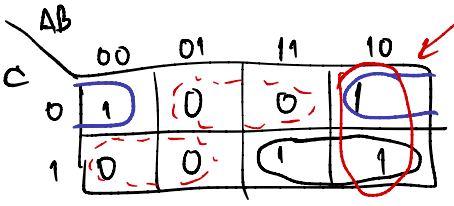




F = AB + BC + AC

(A ไม่เปลี่ยน) (B ไม่เปลี่ยน)
∴ เปลี่ยนแต่เพลิดพรีไม่เปลี่ยน

overlap ตรงไม่จำเป็น



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

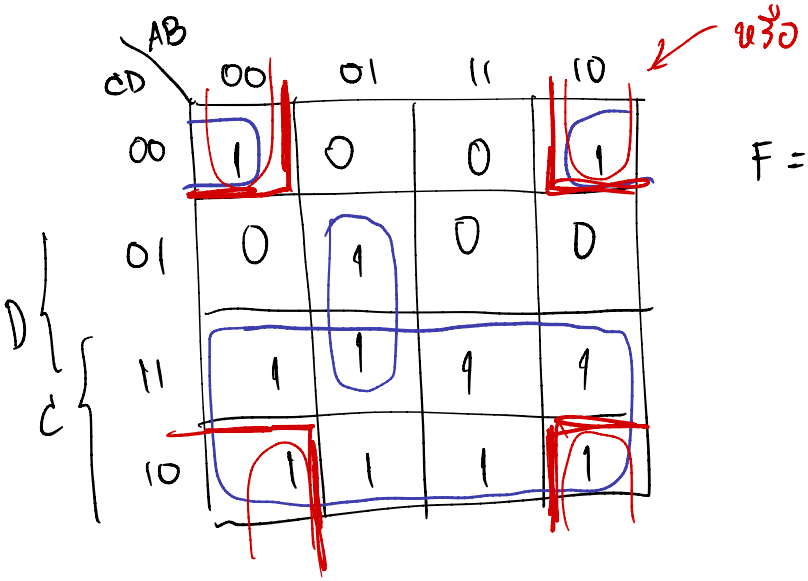
F = $AC + \bar{B}\bar{C}$

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

เกิดจาก swap (0,1) ของ K-O ได้ (off-set) → คือ 0 ถ้า on-set คือ 1

ขยับ F = $(\bar{B}+C)(A+\bar{C})$

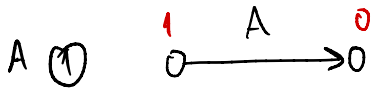
สลับ 0 เป็น 1
1 เป็น 0

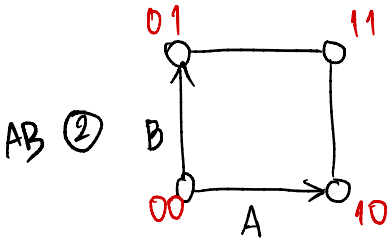


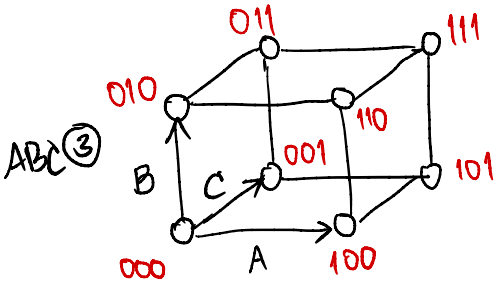
$F = C + \bar{A}BD + \bar{B}\bar{C}\bar{D}$ ← worse

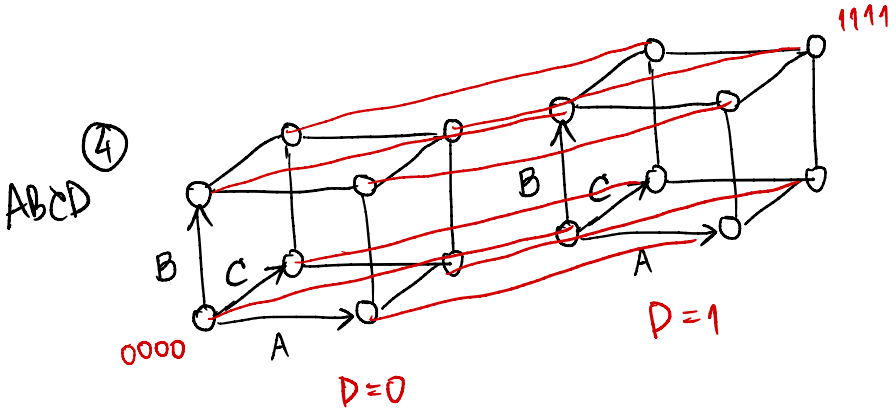
$+ \bar{B}\bar{D}$ ← better

Boolean Cube









Quine McClusky Method

eg.) Minimize F to SoP : $F(A,B,C,D) = \sum m(2,4,5,7,11) + \sum d(1,6)$

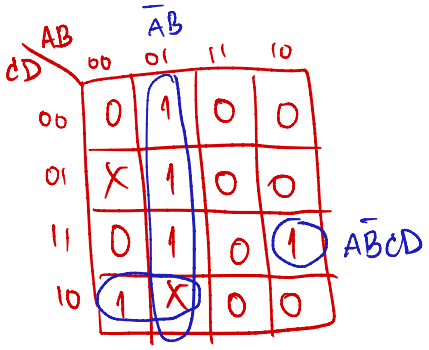
(m_2) 0010	1: (m_1) 0001 ✓	1-5 : 0-01 *	4-6-5-7 : 01-- } *
(m_4) 0100	m_2 0010 ✓	2-6 : 0-10 *	4-5-6-7 : 01--
(m_5) 0101	m_4 0100 ✓	4-5 : 010- ✓	
(m_7) 0111	2: m_5 0101 ✓	4-6 : 01-0 ✓	
(m_{11}) 1011	(m_6) 0110 ✓	5-7 : 01-1 ✓	
(m_1) 0001	3: m_7 0111 ✓	6-7 : 011- ✓	
(m_6) 0110	m_{11} 1011 *		

(onset: $m_2, m_4, m_5, m_7, m_{11}$ → 1: (m_1), m_2, m_4; 2: m_5, (m_6); 3: m_7, m_{11} — Tabulation; DC: (m_1), (m_6))

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

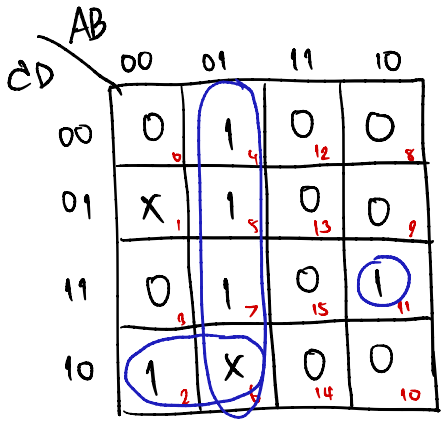
	2	4	5	7	11
m(11)					X
m(1,5)			X		
m(2,6)	X				
m(4,5,6,7)		X	X	X	

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



Don't Care

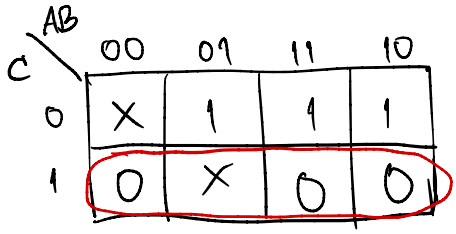
eg.) Minimize F to SoP : $F(A,B,C,D) = \sum m(2,4,5,7,11) + \sum d(1,6)$



$F = A'B + A'CD' + AB'CD$

01-- 0-10 1011

eg.) Minimize F to PoS $F(A,B,C) = \prod M(1,5,7) \prod D(0,3)$



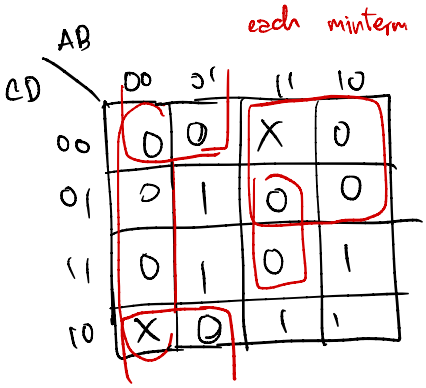
-1 : $F = C'$

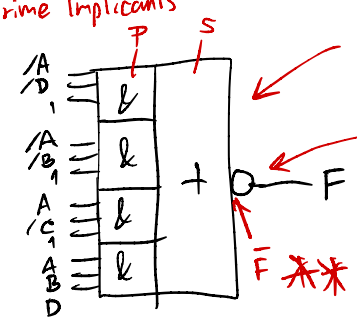
eg. $F(A,B,C,D) = \Sigma m(5,7,10,11,14) + \Sigma d(2,12)$

ออกแบบวงจร F โดยใช้ 3 input - 4 stack AOI 1 ตัว

maximum no. of each minterm = 3

4 Prime Implicants





SoP

เลือก 0 แต่ทำ SoP

ขณะเดียวกัน 0 ถูก 1 ผิด

1 → 0

แต่

01 = 01

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

Multi-level Logic

Conversion → ใช้รวม Gate ที่ทำงานเหมือนกัน → เปลี่ยนกัน

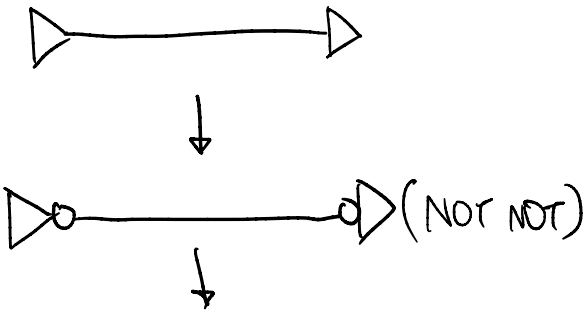
OR: $A+B \sim \overline{\bar{A}\cdot\bar{B}}$

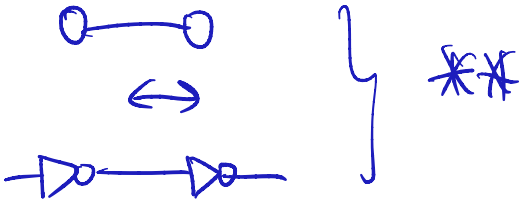
NAND: $\bar{A}+\bar{B} \sim \overline{A\cdot B}$

AND: $A\cdot B \sim \overline{\bar{A}+\bar{B}}$

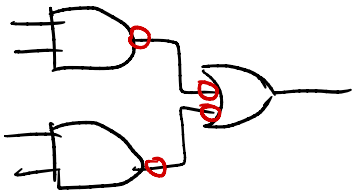
NOR: $\bar{A}\cdot\bar{B} \sim \overline{A+B}$

Wire Conversion :

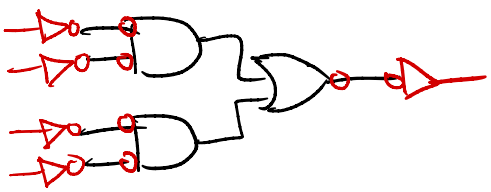




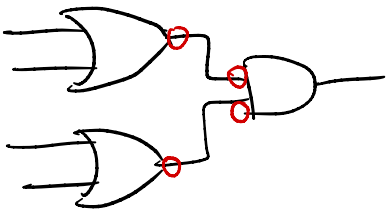
eg. AND/OR → NAND/NAND



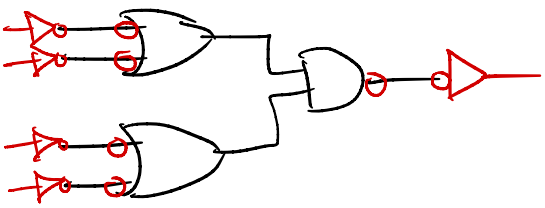
AND/OR → NOR/NOR



eg. OR/AND → NOR/NOR



OR/AND → NAND/NAND

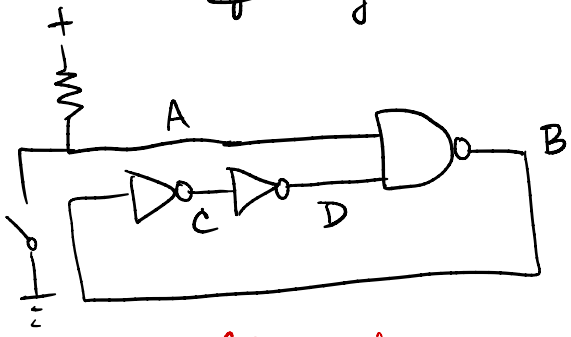


Time Response (Gate Delay)

→ Assume that every gate has 1 unit of delay.

Pulse shaping circuits

eg. Ring Circuit (Intentional delay)



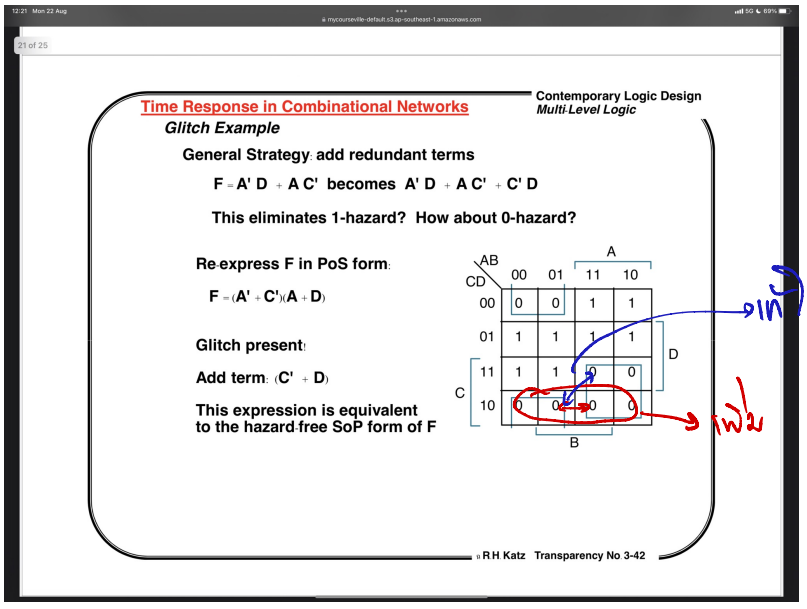
Initial, everything is zero (0).

(Basic clock)

* Hazards/Glitches (Unintentional delay; Unwanted switching)

① Static 1-hz.

Fix it with redundant terms in K-map.



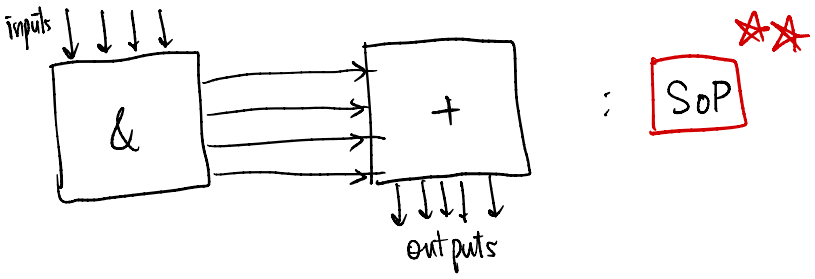
เส้นทแยง (diag. cross)

เพิ่ม redundant เพื่อกัน adjacent cross

Chapt #4. Programmable & Steering Logic

- PAL, PLA (Programmable Array Logic, Programmable Logic Array)
- Switch, MUX-selector, Decoder
- Combinational.

• PAL & PLA



PAL → Constrained OR, generalized AND arrays

PLA → Generalized OR & AND arrays (ข้อสอบ : ข้อใน LAB)

2-bit Gray code

00
01
11
10

mirror

3-bit Gray code

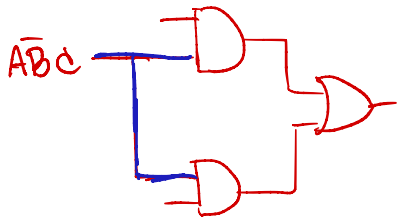
000
001
011
010
110
111
101
100

4-bit

0000
0001
0011
0010
0110
0111
0101
0100
1100
1101
1111
1110
1010
1011
1001
1000

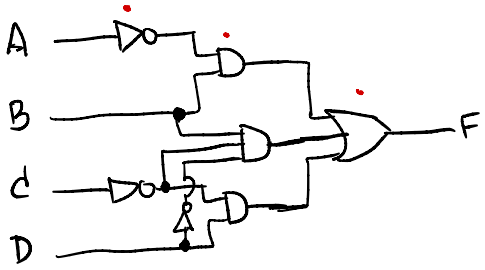
ใน Personality matrix (คือ Input ของ gate)

เช่น



eg. $F(A,B,C,D) = \bar{A}B + \bar{C}D + B\bar{C}\bar{D}$ มี Hazard หรือไม่

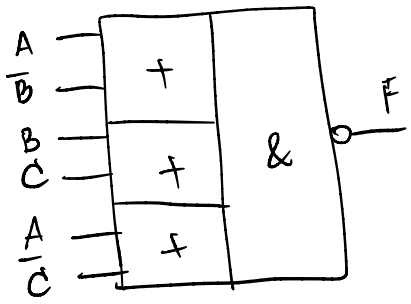
วงจรเขียนวงจรใหม่ ที่ไม่เกิด Hazard และเขียน Timing Wave Form ประเภทที่เกิด Hazard ได้.

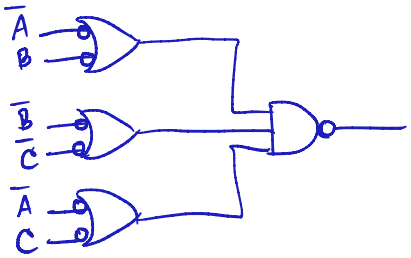


CD \ AB	00	01	11	10
00	0	1	1	0
01	1	1	1	1
11	0	1	0	0
10	0	1	0	0

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

eg. NAND/NAND , NOR/NOR หรือ OAI :

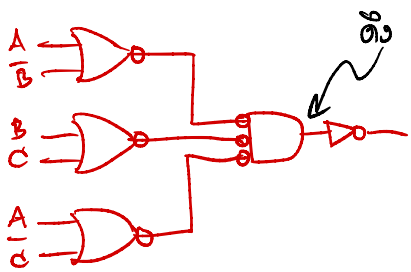




NAND/NAND

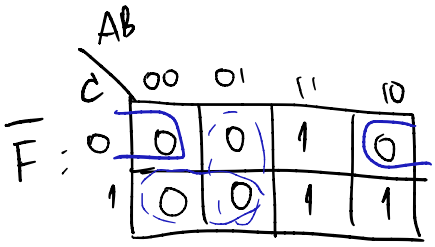
$(A+\bar{B})(A+\bar{C})$

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



ต้อง bubble มาข้างนอก

NOR/NOR

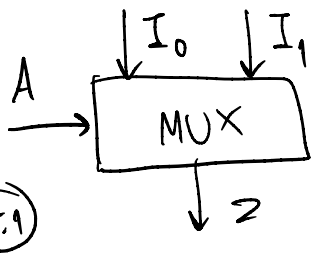


$\bar{F}$:

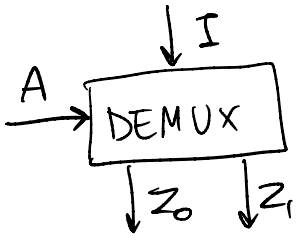
$=A$

Multiplexer / Demultiplexer (MUX / DEMUX)

↳ "Selector" ↳ "Decoder"



2:1



Truth Table

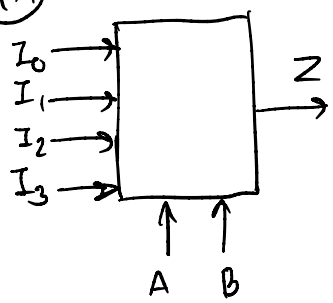
Logical Form

I_1	I_0	A	Z
⋮			⋮

Functional Form

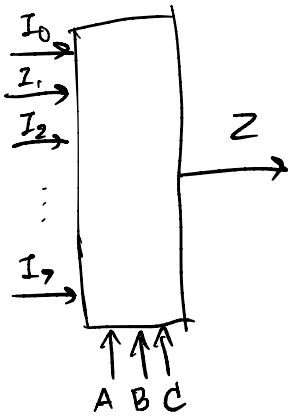
A	Z
0	I_0
1	I_1

4:1



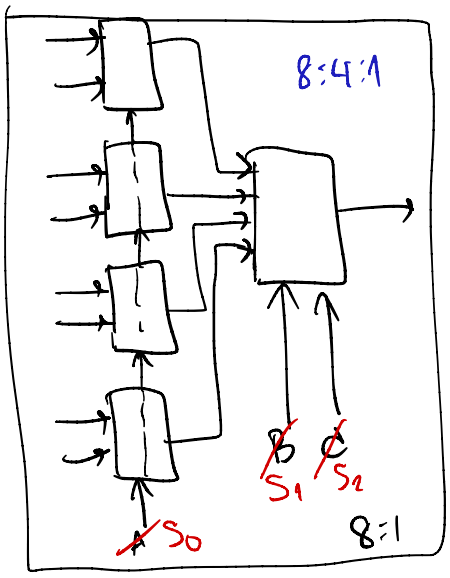
A	B	Z
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

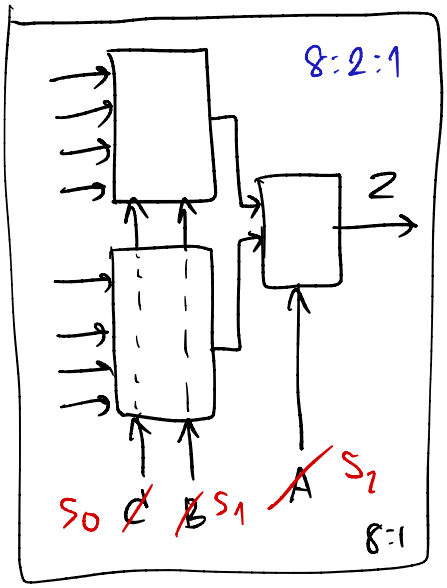
8:1



ขยาย Cascade ด้วย 2:1, 4:1

General: $Z = \sum_{k=0}^{2^n-1} m(k) I_k$



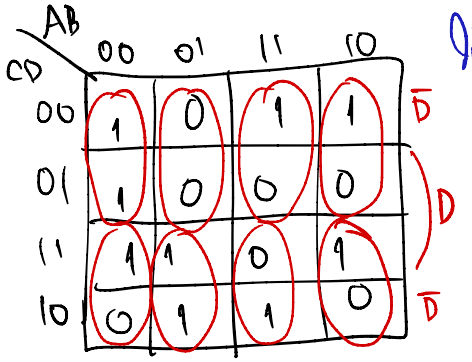


Functions from MUX

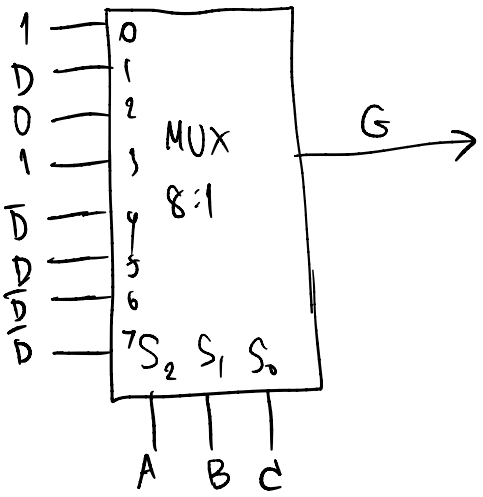
eg. $F(A,B,C) = \Sigma m(0,2,6,7)$

$F(..)$ → ใช้ MUX $2^{n-1}:1$ ได้เสมอ
└ n ตัวแปร

Generalization : K-map : $G(A,B,C,D)$



ใช้ A,B,C เป็น Control, D เป็น Input



→ เหมาะกับการทำ TTL Package (แต่ไม่ Gate efficient)

ในวงจรแบบ Analog Circuit : input สามารถเป็น output ได้ [Analog MUX = Analog DEMUX]

DEMUX (Decoder)

1:2, 2:4, 3:8, 4:16, ..., $n : 2^n$

Default เป็น Active Low Enable ตรงข้ามกับ Active High Enable

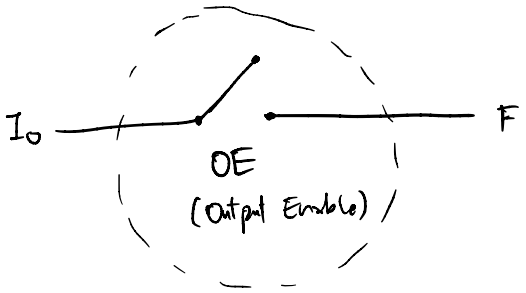
(PIN E, Enb, G)

Same things

The Third State (Tri-state, Open collector, Open drain, ...)

- Logic gates : 0, 1
- Don't Care : X
- 3rd State : Z (High impedance, $R \to \infty$ (D/C))

Disconnected.

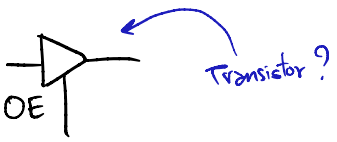


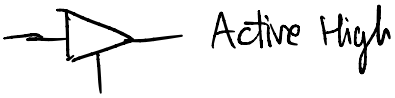
OE	I_0	F
0	0	Z
0	1	Z
1	0	0
1	1	1

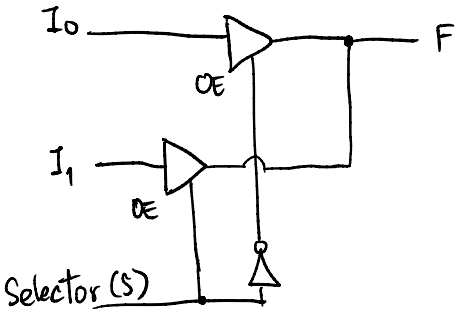
ตัดทิ้ง

Active High

Symbol :

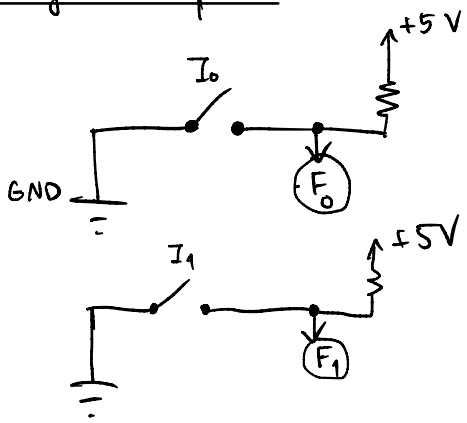




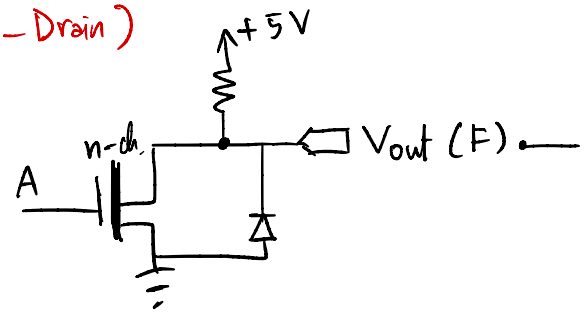


S	F
0	I_0
1	I_1

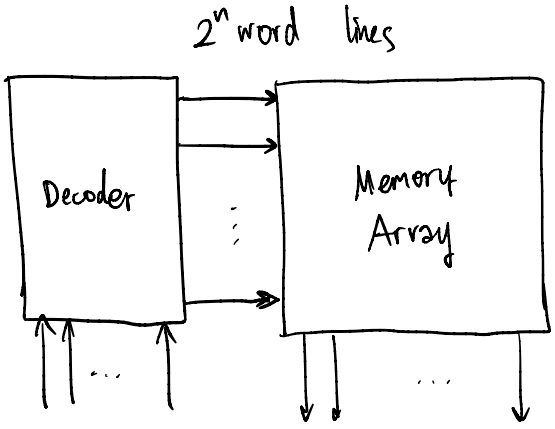
Using pull-up R :



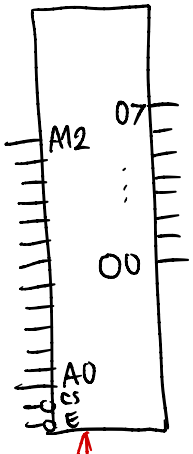
(Open – Drain)



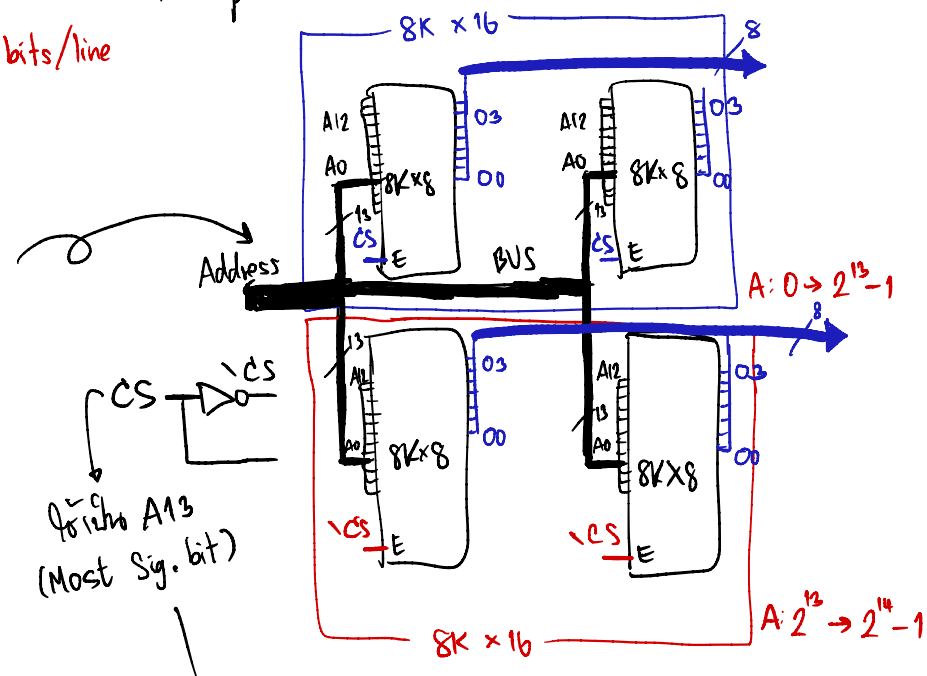
ROM – PLA



(address)
8192 lines × 8 bits/line
8K×8 EPROM



Chip Select (เลือก Chip)
* Active Low (ค่าตรรกะ default)



ถ้าต้องการใช้งานมากกว่า ใช้ A13, A14, A15, ...
โดยดูจาก output ของ decoder

⊕ CS Pin เพิ่มด้วยของ RAM 2x..., 1x... : Single/Dual/Quad Rank

RAM

2x... → มี CS ข้อดีกว่า, เปลี่ยนการเพิ่ม Bandwidth (การทำสิ่งต่างๆ พร้อม, burst)

1x... → ไม่มี CS จะช้า เพราะต้องรอ clock cycle.

เสริม "Interleaving", CPU Bandwidth รับได้แค่ 64-bit แต่มี Quad จะได้ 256 bit (CS0, CS1) เพื่อสร้าง queue ที่พร้อมเอาข้อมูลออกมาส่งที่ ระหว่างรอ cycle.

- Combinational Logic Word Problems

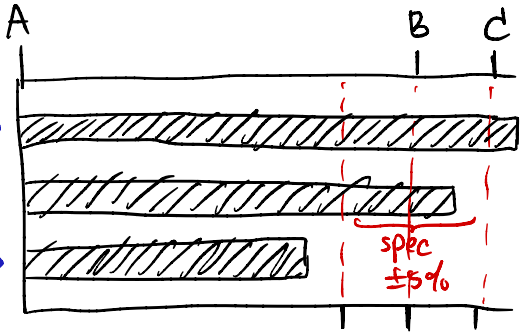
1. Understanding problem : - circuit ควรจะทำอะไร
   - Inputs (control, Data) → Outputs
   - Block Diagram
2. Formulate eg. truth table, ...
3. Implementation eg. DMUX, MUX, PLA, PAL, ...
4. Procedure eg. K-Map, espresso, ...

e.g. Process Line Control

Rod >10% → ถังสีแดง 111

(spec) Rod ±5-10% → ถังสีเขียว 110

Rod <10% → ย้อนกลับ Belt 100



Sensors: 3x light barrier (LED + Photocell) 0 1

ที่เหลือคือ Light barrier ถูก trip เมื่อมีวัตถุมาขั้น (1)

100 → too short

110 → in spec (O1 = 1)

111 → too long (O2 = 1)

else → X (Don't Cares)

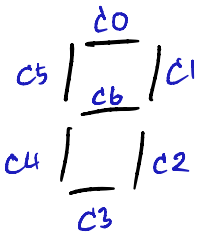
A	B	C	F	O1	O2
1	0	0	Sh	0	0
1	1	0	Sp	1	0
1	1	1	Ln	0	1
else			X	X	X

(เลือกแค่ case)

เป็น 0,0 อีกอัน

$\therefore O1 = AB\bar{C}, O2 = ABC$

e.g. BCD → 7-Segment



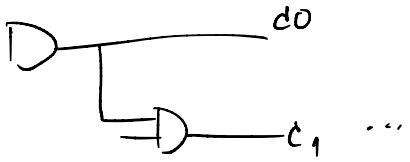
Sol 1. combi

อาจใช้ PAL ได้ ← strict OR

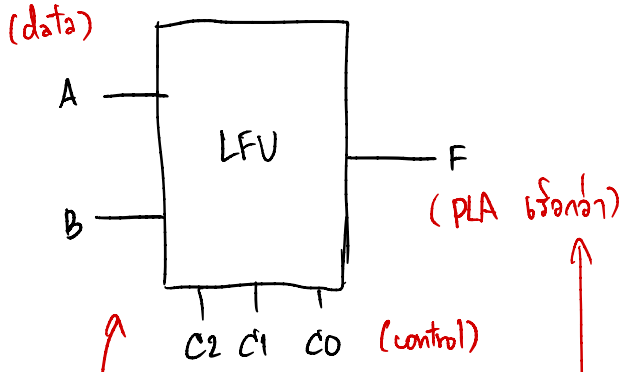
เช่น มี 14 unique product terms (จาก K-Map)
แต่ถ้าใช้ espresso อาจได้ 9 unique prod. terms

Sol 2. Multi Level

ต้องเทอมออกมาใช้ซ้ำ



eg. Logical Function Unit (Not ALU) ที่ใช้ร่วมกันใน ALU

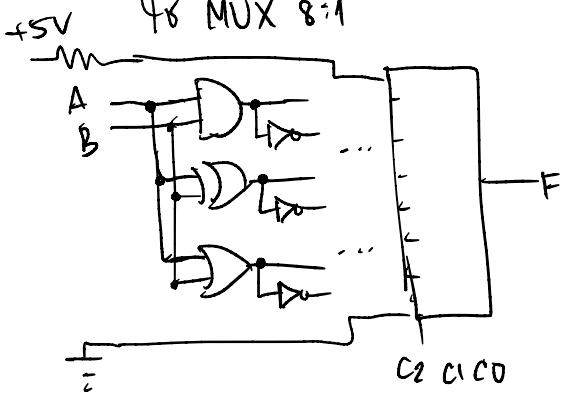


(PLA หรือก่อน)

eg. LFU

C2	C1	C0	F	
0	0	0	1	0
0	0	1	A+B	1
			$\overline{A-B}$	2
			$A \oplus B$	3
			$\overline{A \oplus B}$	4
			A-B	5
			$\overline{A+B}$	6
1	1	1	0	7

ใช้ MUX 8:1



(Better Implementation)

ROM Access , EPROM

Chapt.#5 : Arithmetic Circuits

- Arithmetic Logic Unit (ALU)
- Space – Time Tradeoffs

- Binary addition , Full adder n-bits

- Binary representation → จำนวนบวก, จำนวนลบ → สำหรับคิดเลขกับ Arithmetic Ops.

MAJOR มี 3 แบบ

– Sign & Magnitude

Sign | Mag.

$S_N S_{N-1} \dots S_2 S_1 S_0$

ปัญหา 1. มี 0 ที่ 0000, 1000 [-7, 7]

2. บวกจำนวนบวกกับลบไม่ได้! (ผลลัพธ์ออกมาประหลาด)

3. บวกจำนวนบวกได้

[ไม่สน sign มาเทียบ magnitude แล้วค่อยใส่ sign]

adding

$0 \to 2^{n-1}-1$

$0 \to -2^{n-1}+1$

– One's complement $x \to$ Flip bits

ปัญหา 1. มี 0 ที่ 0000, 1111 [-7, 7]

2. บวกจำนวนบวกกับลบได้, Inverse ได้ แต่ไม่ Natural

[end around carry]

$0 \to 2^{n-1}-1$

$-2^{n-1}+1 \to 0$

↳ $M - N = M + \bar{N}$

Modern CP ทุกคนใช้ตอนนี้

– Two's complement $x \to$ Flip bits $\to +1$

1. แก้ 0 ซ้ำได้ → ใช้ธงเพิ่ม [-8, 7]
2. ทำกลับไปกลับมาได้เสมอ
   - 2cmp (2cmp (X)) = X "
3. Full adding operation!

[carry out] Cin, Cout

$0 \to 2^{n-1}-1$

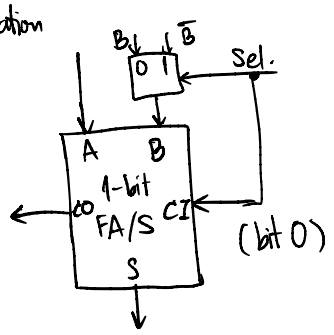
$-2^{n-1} \to 0$

Adder Circuits

- Half Adder : A, B → S, CO
- Full Adder : A, B, CI → S, CO

- Adder/Subtractor

① Implementation ด้วย



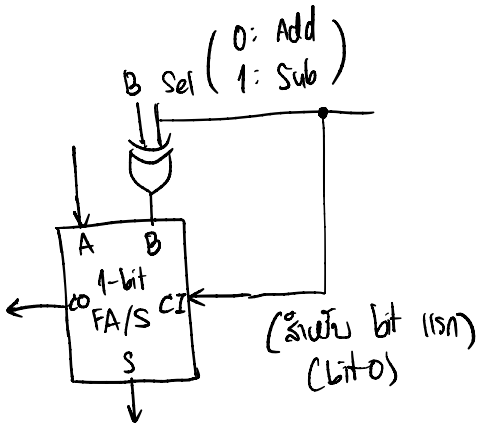
Sel	Bout
0	B
1	B̄

Using XOR

Sel	B	Bout = B⊕Sel.
0	0	0
0	1	1
1	0	1
1	1	0

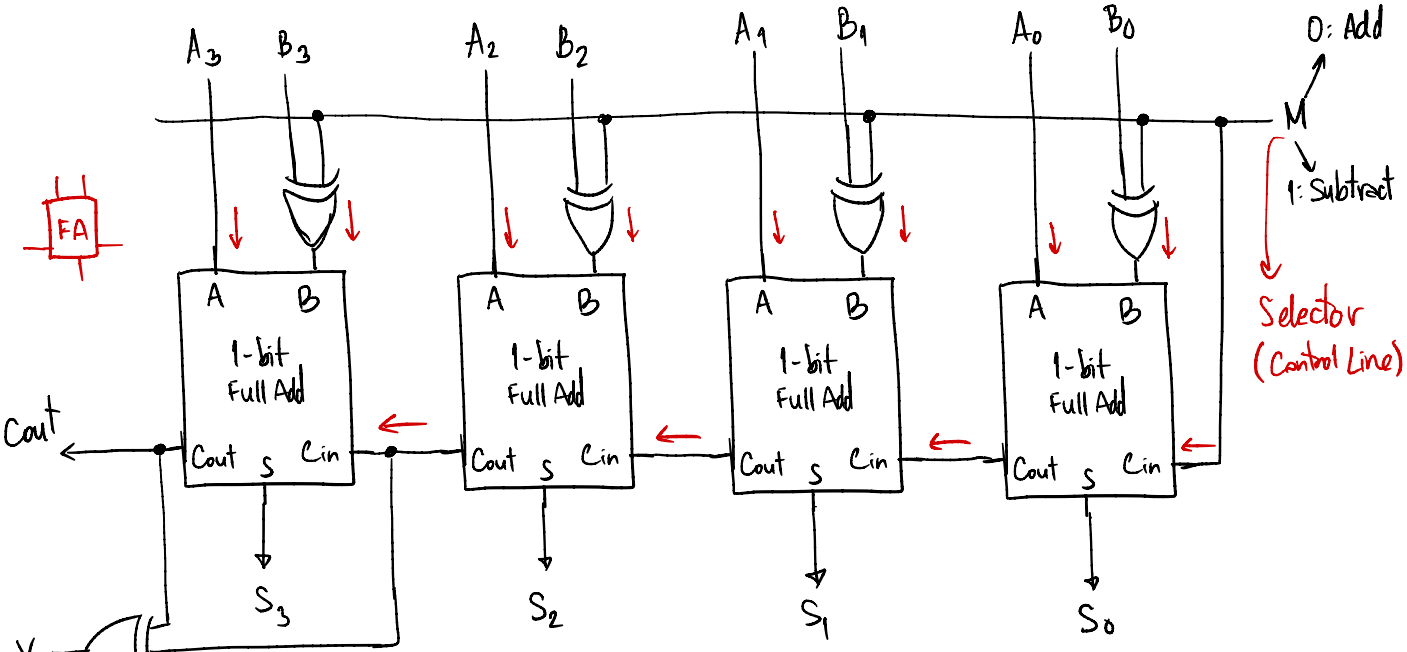
} เหมือน Flip bit

② XOR (ดีกว่า)



Ripple Carry Adder

↳ 4-bit Full Adder / Subtractor for 2's complement (signed)



Overflow Indicator: [Last 2 carries]

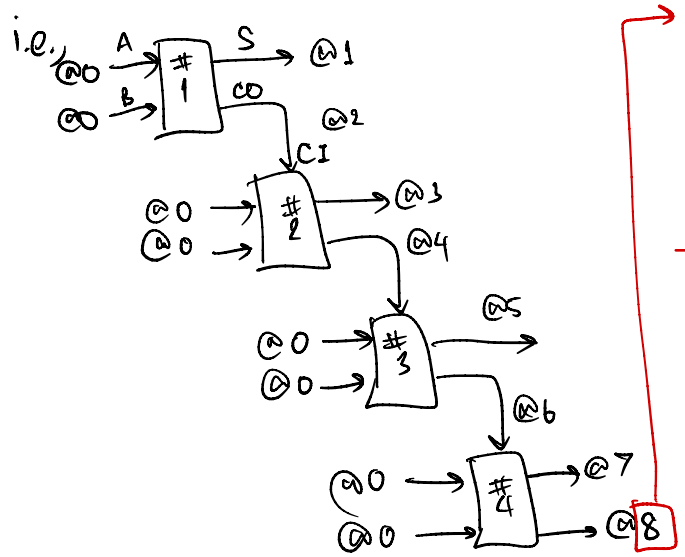
0 = Valid (OK)

1 = Invalid (Overflow)

(CLA)

– Carry Lookahead ⟶ เพื่อ Optimize การบวกเลขให้เร็วขึ้น

CI รับค่าจาก CO ของ CO มี Gate Delay จาก A_i, B_i, CI = @N+2 (2 Gate Delays)



→ 4 Bit → 8 Gate Delay

32 Bit → 64 Gate Delay

! BAD (Critical Delay)

A	B	Cin	Cout
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

P, P

$(A \oplus B)C_{in} = C_i P_i$

$= (A+B)_{Cin}$ *

G

$AB = G_i$

G

Improve

↳ Carry Generate (G_i) & Propagate (P_i)

"บิตผลลัพธ์ที่ i+1"

"บิตผลลัพธ์ที่ 1"

$\therefore C_{i+1} = G_i + C_i P_i$

$G_i = A_i B_i$

$P_i = A_i \oplus B_i$

CI = CO here.

Generate Carry (C_i) when A = B = 1

$S_i = A_i \oplus B_i \oplus C_i = P_i \oplus C_i$

$C_{i+1} = A_i B_i + A_i C_i + B_i C_i$

$= A_i B_i + C_i (A_i + B_i)$ ⟵ special case

$= A_i B_i + C_i (A_i \oplus B_i)$

$$C_{i+1} = G_i + C_i P_i \quad \text{(Recurrence)}$$

ดังนั้น $C_1 = G_0 + P_0 C_0$

$C_2 = G_1 + P_1(G_0 + P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0$

⋮

C_n ⟶ expands into pseudo 2-level logic (faster)! [GOOD]

i.e., C_4 เทียบกับ P_3 ใช้ 3 Gate Delays แต่ ออกแบบยากมาก! [BAD]

→ Max = 4 bit ที่ make sense

ปัญหา
* Improve ยังไง
* ใช้ระบบ Cascaded Carry Lookahead

Note

$C_1 = G_0 + C_0 P_0$

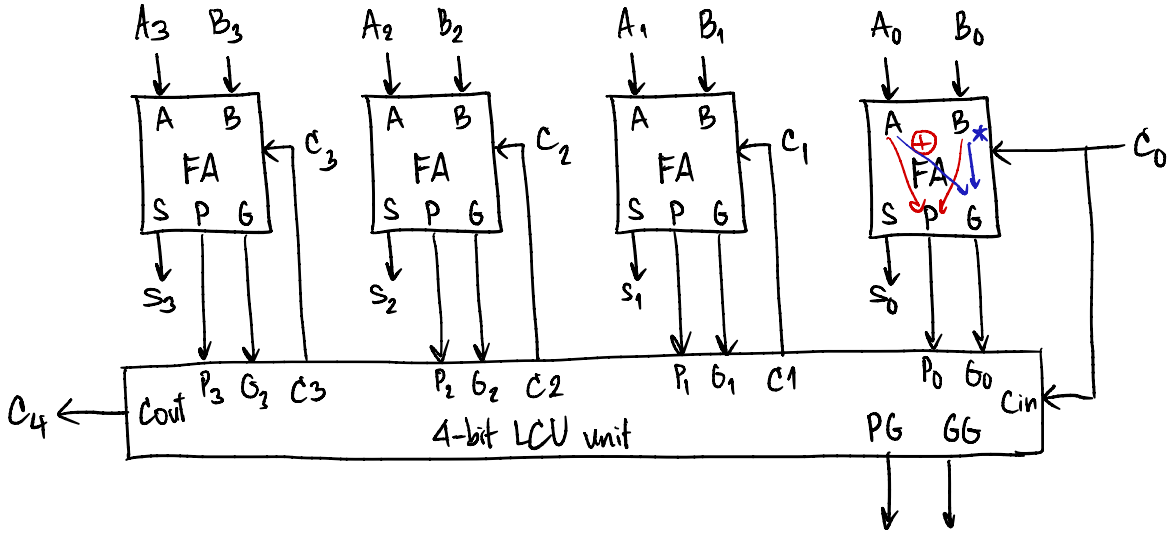
$C_2 = G_1 + G_0 P_1 + C_0 P_0 P_1$

$C_3 = G_2 + G_1 P_2 + G_0 P_1 P_2 + C_0 P_0 P_1 P_2$

$C_4 = G_3 + G_2 P_3 + G_1 P_2 P_3 + G_0 P_1 P_2 P_3 + C_0 P_0 P_1 P_2 P_3$

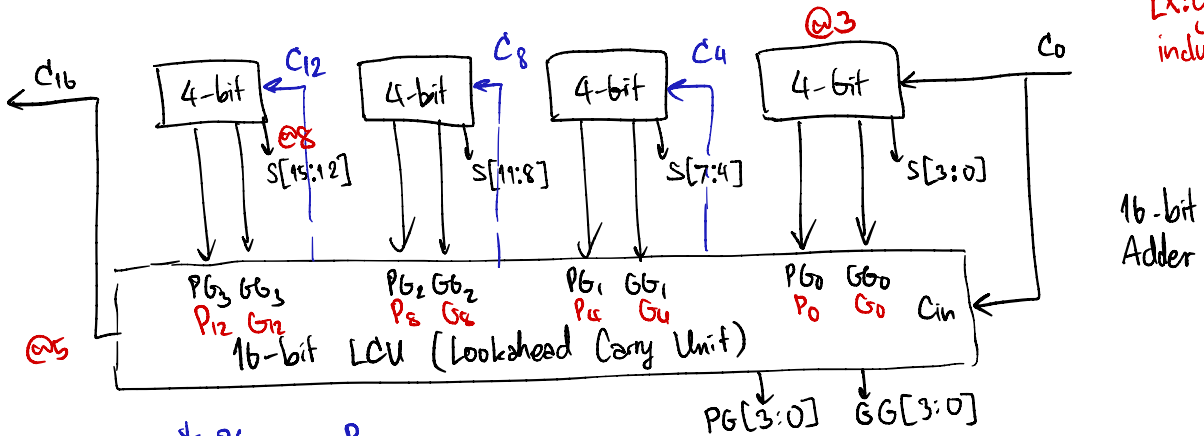
4-bit CLA

จะได้ $CG = GG + C_{in} \cdot PG$ → เอาไปทำ 16-bit Adder



16-bit CLA using Group Propagate, Group Generate (PG/GG) : LCU

Note
[x:y] inclusion.



* $PG_n \rightarrow P_{LCU}$
$GG_n \rightarrow G_{LCU}$

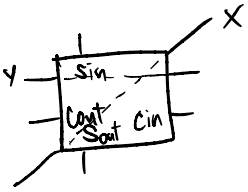
Alternative : "Carry Select Adder" เพิ่ม hardware ที่ redundant

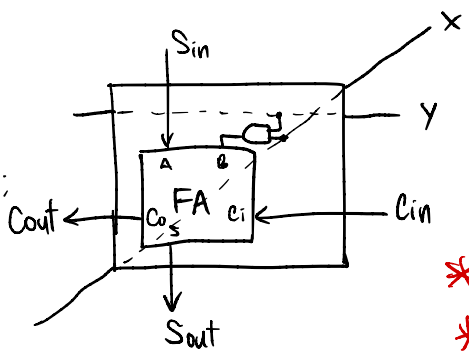
↳ คิดทั้ง case carry = 0 และ 1 แล้วใช้ MUX เลือกภายหลัง

(ไม่ค่อยมีคนใช้)

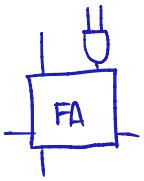
Multipliers : Array of Adder & Carrying "Array Multiplier" (Pipelined)

Building Block





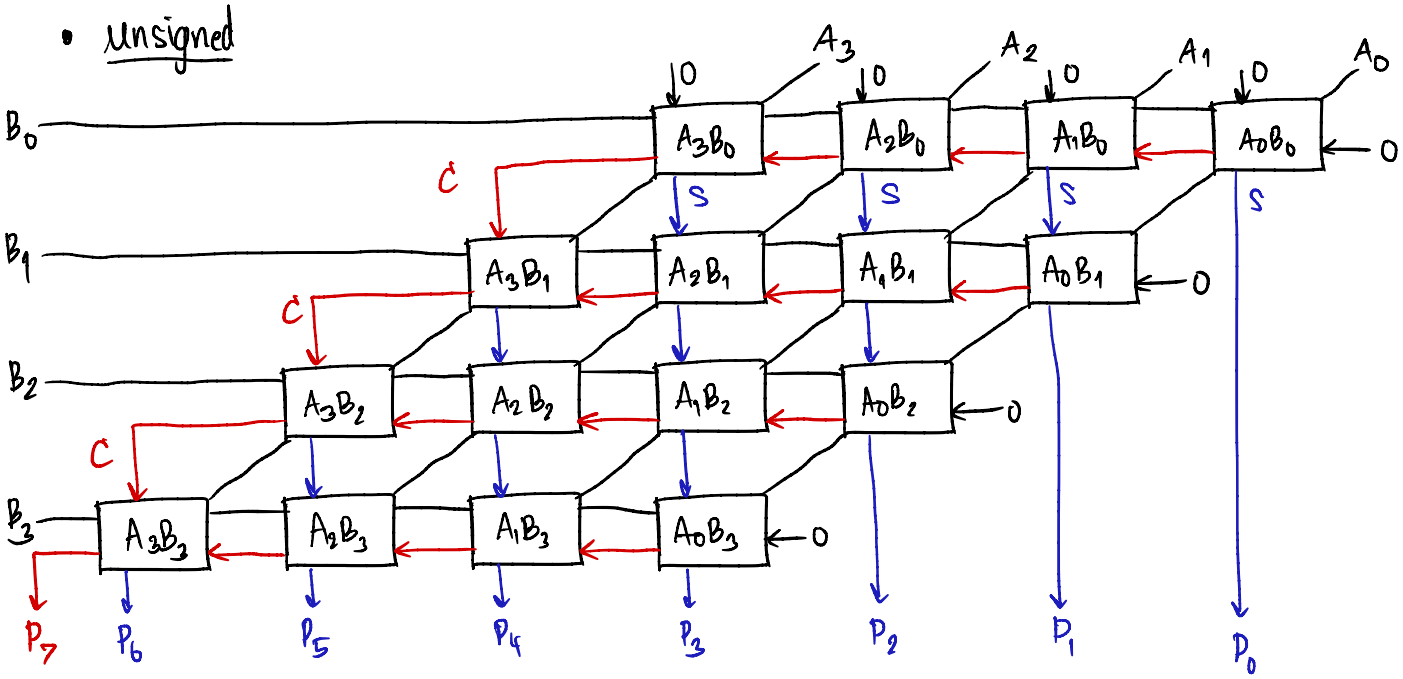
Alternatively,



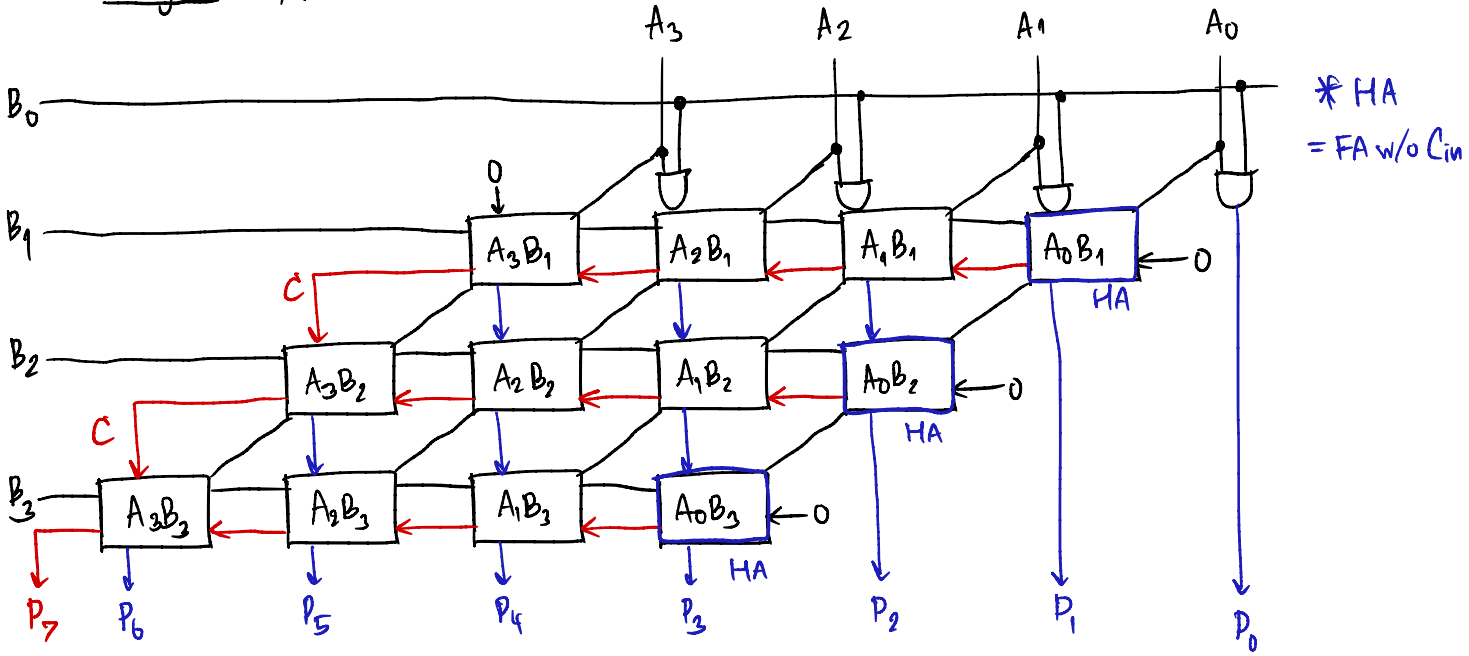
* Each block is partial product (&)
* ในแต่ละ block A_iB_0, A_0B_i ใช้ HA แทน

4×4 combinational multiplier ($A_3A_2A_1A_0 \times B_3B_2B_1B_0 = P_7P_6P_5P_4\ P_3P_2P_1P_0$)

- unsigned

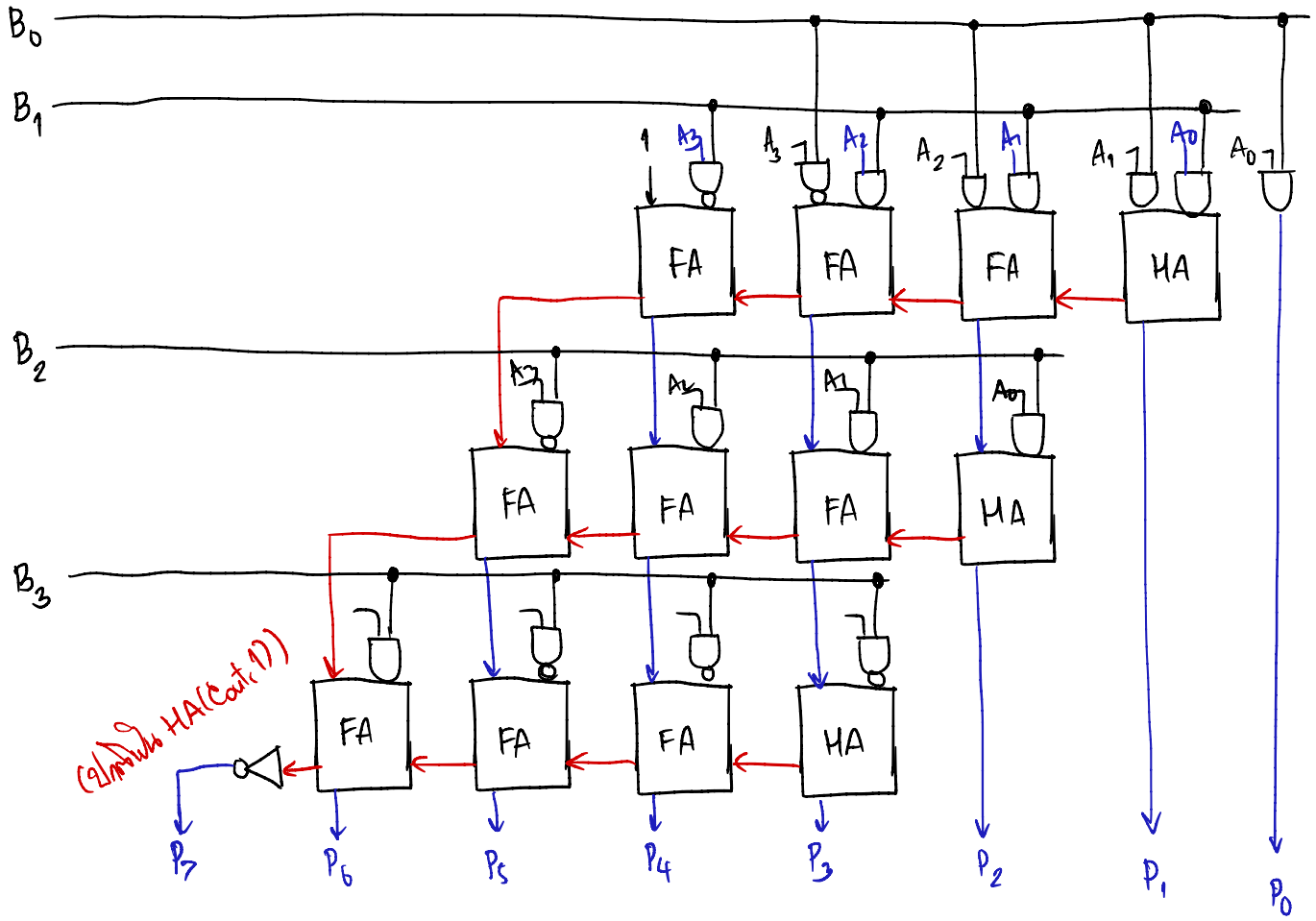


- unsigned (bypassed)



* HA = FA w/o Cin

• signed (2's complement)



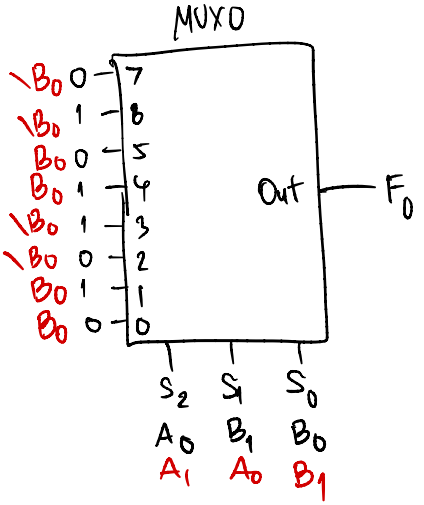
Tutoring

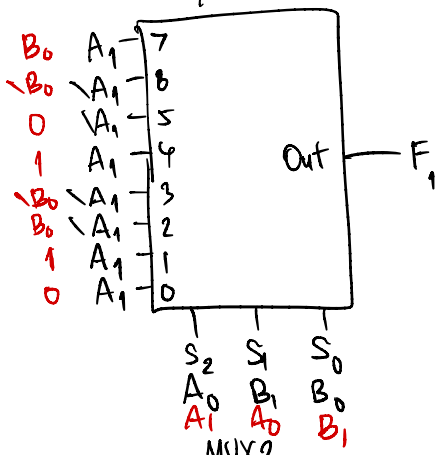
2 Bit Adder using 8:1 MUX

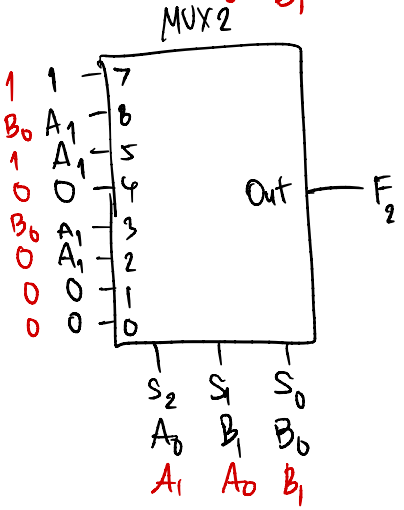
A_1	A_0	B_1	B_0	F_2	F_1	F_0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

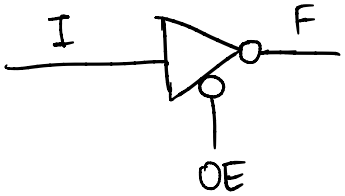
MUX2 MUX1 MUX0

↳ Alternative (Easier)









OE	I	F
0	0	1
0	1	0
1	0	Z
1	1	Z