Consensus Theorem:

$F = (A+B)(A'+C)(B+C)$

$\Rightarrow F = (A+B)(A'+C)$

$F = AB + BC' + AC$

$\Rightarrow F = BC' + AC$

Boolean Algebra:

$B = \{0, 1\}$ $\sim, +, \cdot$ { simplify logical expression

Postulates

Theorems

e.g. $f = \bar{a}\bar{b} + a\bar{b} + bd + cd$

cost: $4 \times 6 + 10 + 4 = 38$ switches

$f = (\bar{a}+a)\bar{b} + bd + cd$

$= \bar{b} + bd + cd$ $= \bar{b} + d + cd = \bar{b} + d$ → adsorption theorem

$= (c+\bar{c})\bar{b} + bd + cd$

$= \bar{b}c + bd + cd + \bar{b}\bar{c}$] consensus theorem

$= \bar{b}c + bd + \bar{b}\bar{c}$

$= \bar{b}(c+\bar{c}) + bd$

$= \bar{b} + bd$] Adsorption theorem

$= \bar{b} + d$

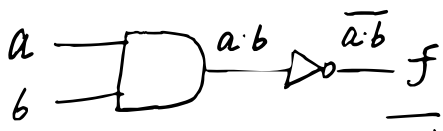
cost: $6 + 2 = 8$ switches

DeMorgan's Theorem:

$\overline{a \cdot b} \equiv \bar{a} + \bar{b}$

$\overline{a+b} \equiv \bar{a} \cdot \bar{b}$ } Only two operators necessary $[\sim, \cdot]$ or $[\sim, +]$

$[\sim, \cdot, +]$ → basic operators



$\Rightarrow$ NAND $f = \overline{a \cdot b}$

$\Rightarrow$ Conjecture:
All operations can be expressed using NAND operator because it has the two necessary operators.

Proof:

NOT: $f = \overline{a \cdot b}$

$b = a \Rightarrow f = \overline{a \cdot a} = \bar{a}$

AND: $a \cdot b = \overline{\overline{a \cdot b}} = \bar{f} = \overline{f \cdot f} = \overline{(\overline{a \cdot b}) \cdot (\overline{a \cdot b})}$

3 NAND operators

OR: $\overline{a+b} = \bar{a} \cdot \bar{b}$

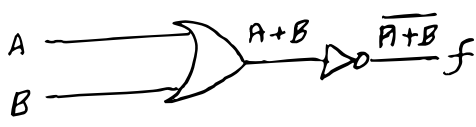
$a + b = \overline{\bar{a} \cdot \bar{b}} = \overline{(\overline{a \cdot a}) \cdot (\overline{b \cdot b})}$

3 NAND operators

$\therefore$ All operations can be expressed.

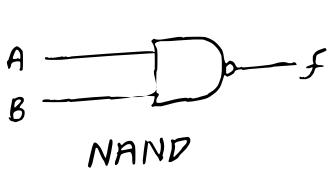
NAND → Universal Operator

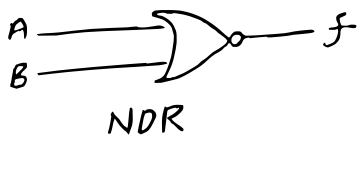
Similarly,



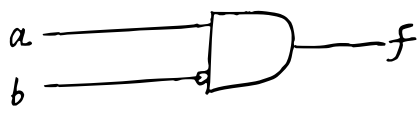
$\Rightarrow$ NOR $f = \overline{A+B}$

NOR → Universal Operator.





BUT NOT:



$f = a \cdot \bar{b}$

$a \cdot \bar{a} = 0 \rightarrow a \cdot \bar{0} = a \cdot 1 = a$

NOT: set $a = 1 \Rightarrow f = a \cdot \bar{b} = 1 \cdot \bar{b} = \bar{b}$

AND: $a \cdot b = a \cdot \bar{\bar{b}} = a \cdot (\bar{b}) = a \cdot (1 \cdot \bar{b})$

OR: $a + b = \overline{\bar{a} \cdot \bar{b}} = 1 \cdot (\overline{\bar{a} \cdot \bar{\bar{b}}}) = 1 \cdot (\overline{\bar{a} \cdot \bar{\bar{b}}})$

$= 1 \cdot (\overline{(1 \cdot \bar{a}) \cdot \bar{b}})$

BUT NOT → Universal Operator

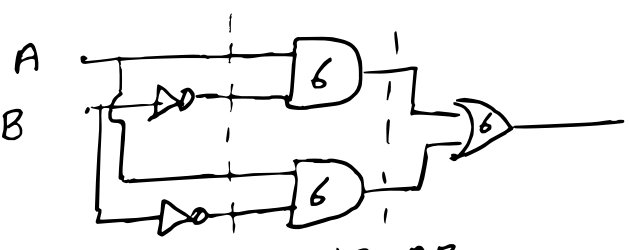
Universality

Truth table

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

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

SOP → Sum of Products

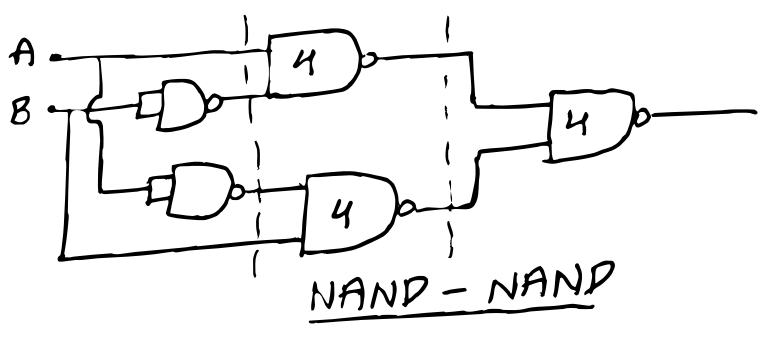


AND-OR

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

$C = \bar{\bar{C}} = \overline{\overline{A\cdot\bar{B} + \bar{A}\cdot B}}$

$= \overline{(\overline{A\cdot\bar{B}})\cdot(\overline{\bar{A}\cdot B})}$



NAND - NAND

TRUTH-TABLE → SOP → Optimize (Boolean Algebra) → SOP (AND-OR) → SOP (NAND-NAND)

	A	B	Z
m_0	0	0	0
m_1	0	1	1
m_2	1	0	1
m_3	1	1	0

2^n bits

$Z = A\cdot\bar{B} + \bar{A}\cdot B$ (SOP)

$Z = m_1 + m_2 = \Sigma(1,2)$

→ ON SET (min terms)

→ OFF SET (max terms)

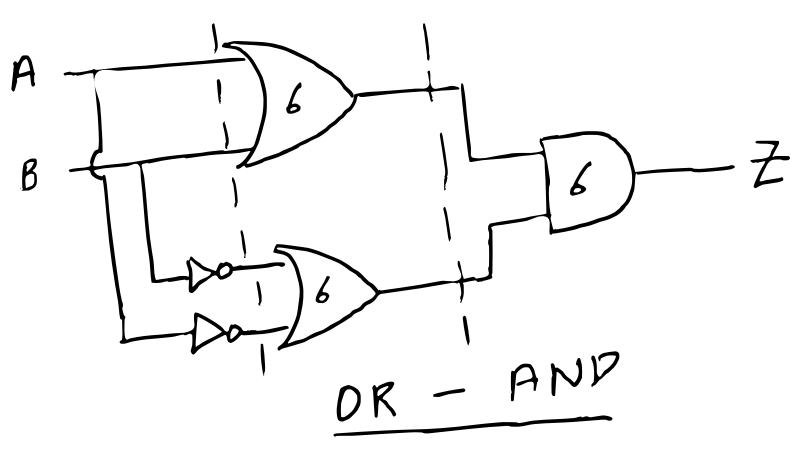
$\bar{Z} = \bar{A}\cdot\bar{B} + AB$

$Z = \overline{\bar{A}\cdot\bar{B} + AB} = (\overline{\bar{A}\cdot\bar{B}})\cdot(\overline{AB})$

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

$\therefore Z = (A+B)\cdot(\bar{A}+\bar{B})$

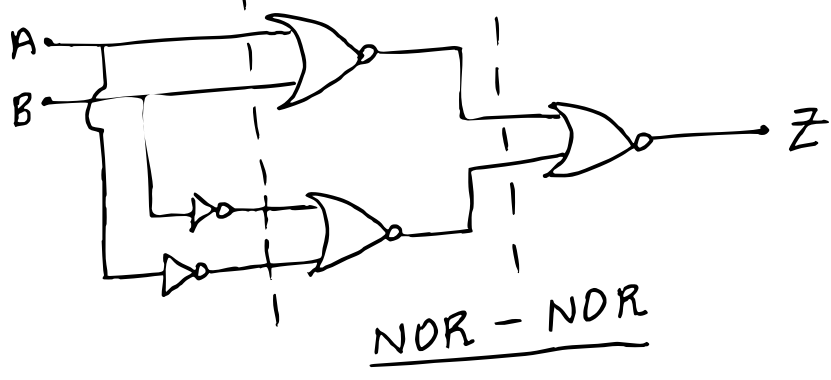
↳ Product of Sum (POS)



OR - AND

$Z = (A+B)\cdot(\bar{A}+\bar{B})$

$\bar{\bar{Z}} = \overline{(A+B)\cdot(\bar{A}+\bar{B})} = \overline{\overline{(A+B)} + \overline{(\bar{A}+\bar{B})}}$



NOR - NOR

	A	B	Z
$\bar{A}+\bar{B} = M_0$	0	0	0
$\bar{A}+B = M_1$	0	1	1
$A+\bar{B} = M_2$	1	0	1
$A+B = M_3$	1	1	0

$Z = (A+B)\cdot(\bar{A}+\bar{B})$

$= M_0\cdot M_3$

$= \Pi(0,3)$

Representations:

- Truth table
- SOm (Sum of minterms) } canonical (unique)
- POM (Product of maxterms)
- SOP (Sum of Product) } Non-canonical
- POS (Product of Sum)

minterms → product of all variables (Intersection → min.)

maxterms → sum of all variables (Union → max.)

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

$\bar{A}\cdot B \rightarrow Z$ ⇓ Implication

Implicant

A	B	C	Z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

$Z = \bar{A}\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C} + ABC$

↳ SOm & SOP

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

↳ SOP but not SOm

n inputs

ABC → 0-implicant ⇒ n-0 literals

$\bar{B}C$ → 1-implicant ⇒ n-1 literals

A → 2-implicant ⇒ n-2 literals

Literals → variables in either normal form or complement form

($A, \bar{A}$ → different literals)

Representations:

TRUTH TABLE
SOm
POM
} Canonical

SOP
POS
} Non-Canonical

Cost (min # switches):

$Z = AB + CD$

$= \overline{\overline{AB} \cdot \overline{CD}}$

AND-OR

No. of literals + No. of product terms

4 + 2

$(4+2) \times 2 = 12$

CMOS

Optimization:

KARNAUGH (BELL LAB)

Graphical Method

A	Z
0	1
1	1

$Z = 1$

A: 0 1

1 1 → 0-implicant

$Z = A + \bar{A}$

$= 1$ // → 1-implicant

A	B	Z
0	0	1
0	1	1
1	0	0
1	1	0

B \ A	0	1
0	1	0
1	1	0

$Z = \bar{A}\bar{B} + \bar{A}B$

$Z = \bar{A}(\bar{B}+B) = \bar{A} \cdot 1$

$Z = \bar{A}$

product of literals: term

sum of literals: clause

AB	00	01	11	10
	1	1	0	0

$Z = \bar{A}$

1-implicant

AB	00	01	11	10
	1	0	0	1

Pair

1-implicant

$Z = \bar{B}$

AB	00	01	11	10
	1	1	1	1

2-implicant Quad

$Z = 1$

$Z = A + \bar{A} = 1$

B \ A	0	1
0	1	1
1	1	1

2-implicant Quad

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

3-implicant octet

n-variable → m-implicant → n-m literals

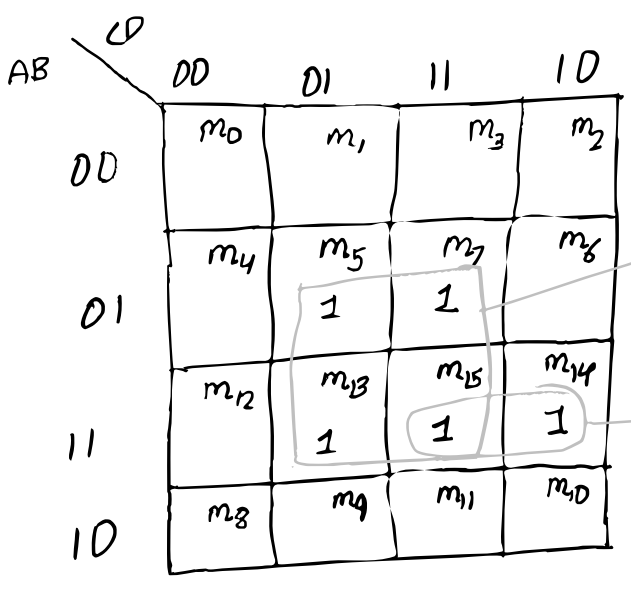
Prime-Implicant:

When implicants cannot grow further, they are called prime-Implicant.

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

K-Map :

$f(A, B, C, D)$



(BD) (PI1)

Prime-Implicant

↑

2-implicant (quad)

(ABC) (PI2)

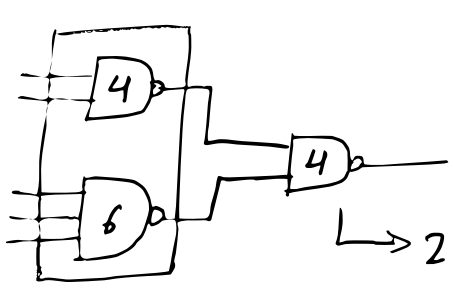
1-implicant (pair)

↓

Prime Implicant

minterm ⟶ 0-implicant

$\therefore f(A, B, C, D) = BD + ABC$



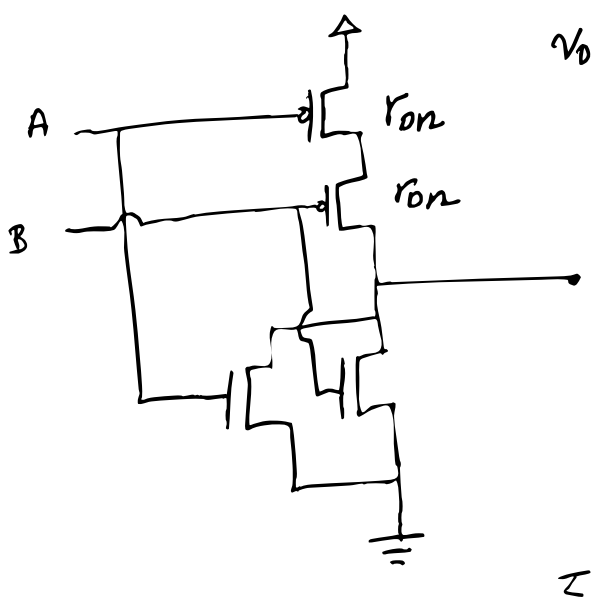
⟶ 2 × (no. of terms)

1D

= 2 × (5 literals)

cost = 2 (No. of literals + no. of terms)

Performance Delay ⟶ Time taken between input & output.



Voltage change

↳ $e^{-t/\tau}$ ($\tau = RC$)

$\tau \propto R$

↓

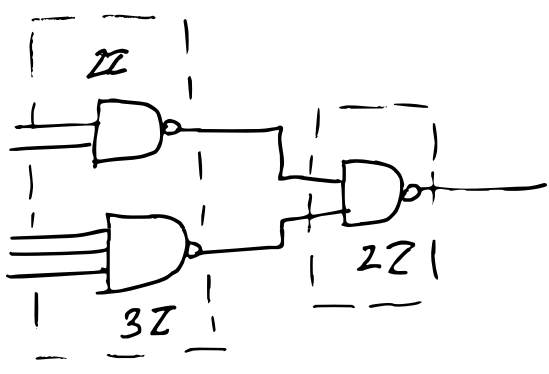
$\therefore \tau \propto$ no. of inputs

↓

$\tau \propto$ delay

$\therefore$ **delay $\propto$ no. of inputs**

↳ (worst-case)



Delay = 5τ

$\therefore$ Delay = # terms + max{# literals in a term}.

$\therefore$ min. delay = min{ max{# literals in a term} + # terms }.

min. cost = min { # literals + # terms }

 # terms need to be minimized in both.

PRIME-IMPLICANT

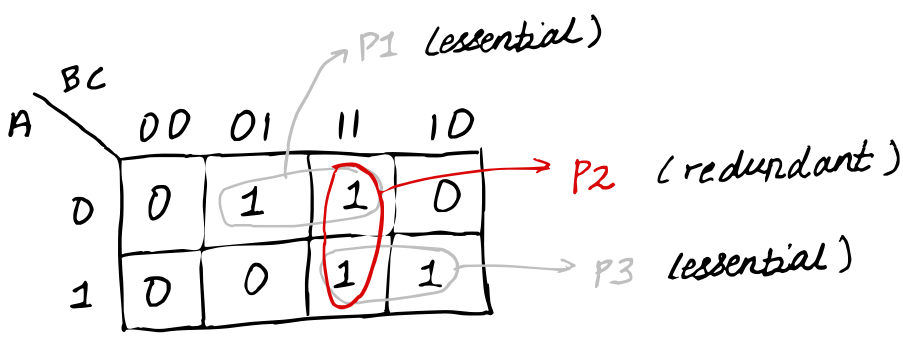
↳ min. no. of literals in any term.

Prime-Implicant :

⟶ Essential P.I. (EPI)

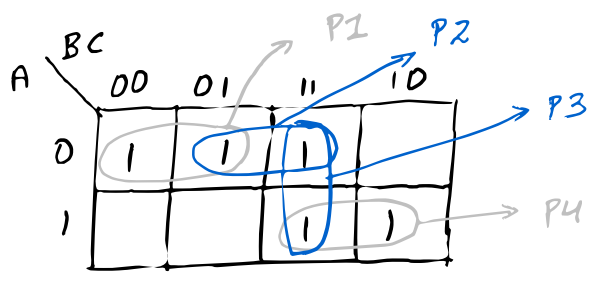
⟶ Redundant P.I. (RPI)

⟶ Selective P.I. (SPI)



$f = P1 + P2 + P3 = P1 + P3$

↳ redundant



$f = P1 + (P2 + P3) + P4$

EPI EPI

Selective Prime Implicants (SPI)

How to select SPIs ?

$SPI = \{P_2, P_3\}$

PATRICK'S METHOD:

$(P_2 + P_3) = 1$

Hypothetically, say

m_3, m_7, m_{12} ⟶ minterms not covered by EPIs

$(P_1 + P_2)(P_2 + P_3)(P_2 + P_4) = 1$

POS ↓ SOP

$= P_1P_2 + P_2 + P_1P_3P_2 + P_2P_3 + P_1P_2P_4 + P_2P_4 + P_2P_3P_4 + P_1P_3P_4 = 1$

↓

anyone has to be 1.

Now, **minimize # literals.**

↓

1

Representation:

1. Truth Table
2. SOm
3. POM
4. SOP
5. POS

} Canonical

} Non-canonical

K-Map

minimize COST or DELAY or POWER (∝ area, ∝ # of switches)

COST → # of switches → # of literals + # of terms

DELAY → max # of literals in a term + # of terms

EPI

RPI X

SPI ← optimization ← Patrick's Method.

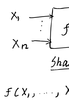
Pause discussion on optimization.

Back to Representation.

Graphical Representation:

Binary Decision Diagram

$f(X_1, \cdots, X_n)$

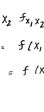


Shannons Decomposition

$f(X_1, \cdots, X_n) = X_1 \underbrace{f(X_1=1, \cdots, X_n)}_{\text{cofactor } (f_{X_1})} + \overline{X_1} \underbrace{f(X_1=0, \cdots, X_n)}_{\text{cofactor } (f_{\overline{X_1}})}$

$\therefore f(X_1, X_2, \cdots, X_n) = X_1 f_{X_1} + \overline{X_1} f_{\overline{X_1}}$

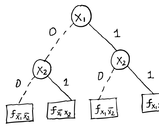
$\equiv X_1 f_{X_1} \oplus \overline{X_1} f_{\overline{X_1}}$ (A, B are orthogonal, if $A \cdot B = 0$, $A + B \equiv A \oplus B$)



$f_{X_1} = X_2 f_{X_1 X_2} + \overline{X_2} f_{X_1 \overline{X_2}}$

$f_{X_1 X_2} = f(X_1=1, X_2=1, \cdots, X_n)$

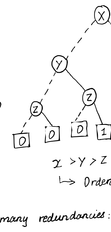
$f_{X_1 \overline{X_2}} = f(X_1=1, X_2=0, \cdots, X_n)$



and so on...

example:

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

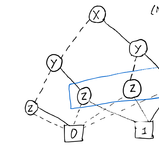


X > Y > Z

↳ Ordered Binary Decision Tree

But this has many redundancies:

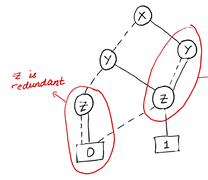
→ Only 2 bits required for 0,1.



(No. of nodes = $2^n - 1 + 2 = 2^n + 1$)

same nodes → isomorphic (each have same mapping 0 → 0, 1 → 1)

→ Isomorphic nodes can be merged together.



Z is redundant

Y is redundant

→ Redundant nodes can be eliminated.

Reduced Ordered Binary Decision Diagram (BDD)

(X > Y > Z)

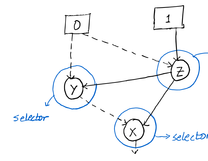
(Not a tree anymore)

(ROBDD)

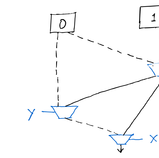
ROBDD is canonical.

→ Can be used to verify equivalence of two functions.

→ Can be directly used for synthesis of circuit.



selector → can be replaced by 2:1 MUX

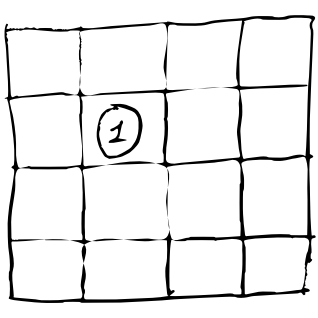


∴ Cost = # NODES × 5 (↳ no. of switches in 1 MUX.)

∴ Delay = Delay of 1 MUX × length of longest path

Length of longest path ≤ no. of inputs

K-Map:



PI → Least number of literals

EPI, RPI, SPI → Patrick's Method
✓ ×

RDBDD
ordered
① Binary Decision Tree → ($2^{N+1}-1$ nodes)

② 3 Reduction Rules → Directed Acyclic Graph
⇓
RDBDD (Canonical)

$f_1 = f_2$

(Adder → linear growth)
(Multiplier → Exponential growth)

Uses:
① Verification
$f_1 = f_2$

② Synthesis

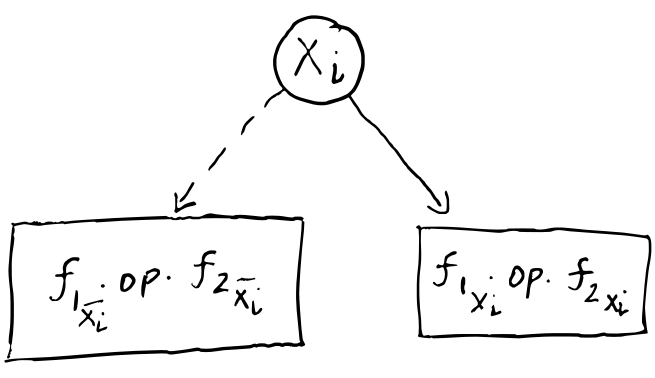
$$f_1(x_1, x_2, \cdots, x_i, \cdots, x_n) = x_i f_{1x_i} + \overline{x_i} f_{1\overline{x_i}}$$

$$f_2(x_1, x_2, \cdots, x_i, \cdots, x_n) = x_i f_{2x_i} + \overline{x_i} f_{2\overline{x_i}}$$

$$f_1 + f_2 = x_i (f_{1x_i} + f_{2x_i}) + \overline{x_i} (f_{1\overline{x_i}} + f_{2\overline{x_i}})$$

$$f_1 f_2 = x_i (f_{1x_i} f_{2x_i}) + \overline{x_i} (f_{1\overline{x_i}} f_{2\overline{x_i}})$$

$$\therefore f_1 \cdot op \cdot f_2 = x_i (f_{1x_i} \cdot op \cdot f_{2x_i}) + \overline{x_i} (f_{1\overline{x_i}} \cdot op \cdot f_{2\overline{x_i}})$$

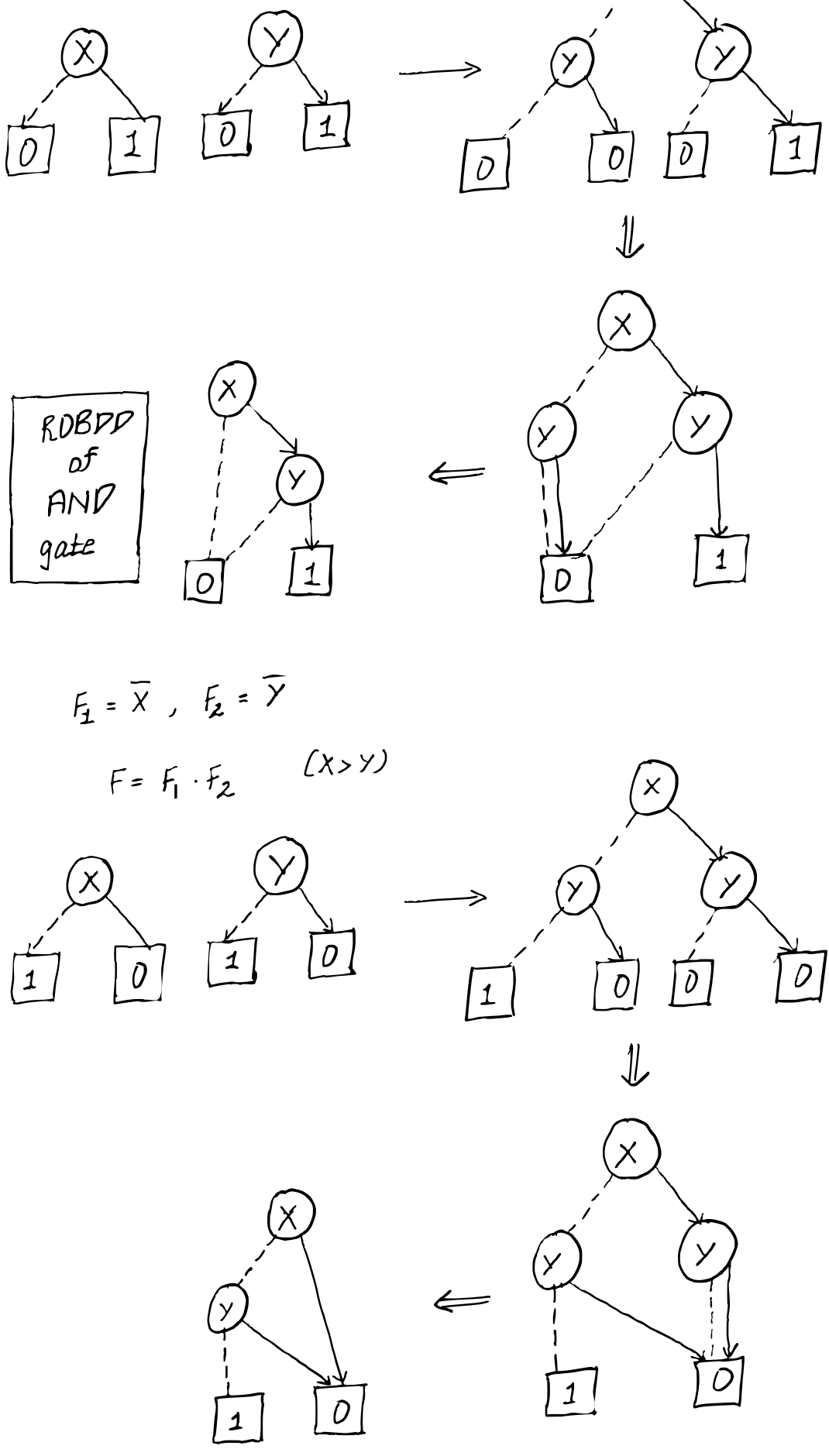


What if f_2 does not depend on x_i?

$$f_2 = x_i f_2 + \overline{x_i} f_2 \quad (= (x_i + \overline{x_i}) f_2 = f_2)$$

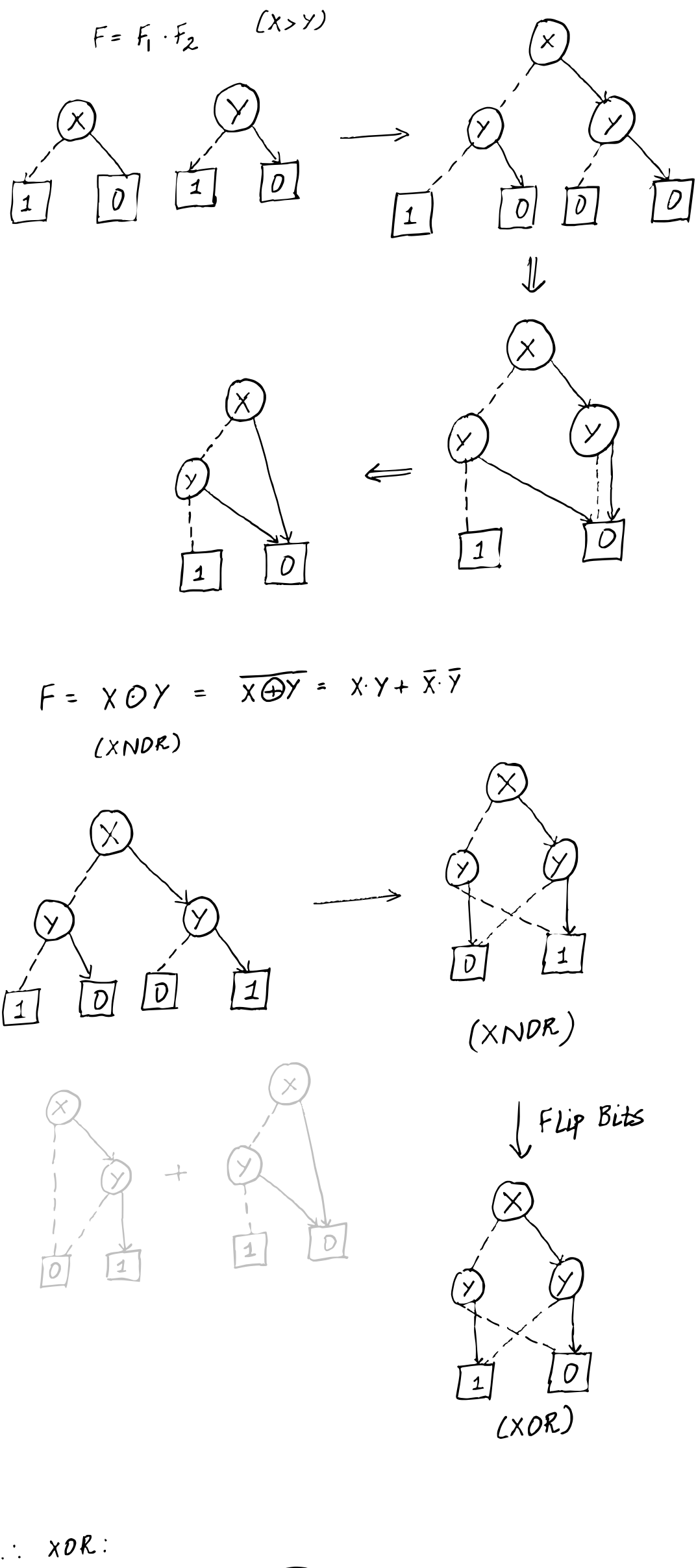
Example:
$F_1 = X$, $F_2 = Y$

$F = F_1 \cdot F_2$ $(X > Y)$



$F_1 = \overline{X}$, $F_2 = \overline{Y}$

$F = F_1 \cdot F_2$ $(X > Y)$



$F = X \odot Y = \overline{X \oplus Y} = X \cdot Y + \overline{X} \cdot \overline{Y}$

(XNOR)

∴ XOR:

equivalent to
(same RDBDD)

Representation:

TRUTH TABLE
SOm
POM
ROBDD
} CANONICAL

SOP
POS
} NON-CANONICAL

RM ← Reed Muller Decomposition

Shannon's Decomposition:

$f(x_1, x_2, \ldots, x_n) = x_i f_{x_i} + \overline{x_i} f_{\overline{x_i}}$

$= x_i f_{x_i} \oplus \overline{x_i} f_{\overline{x_i}}$

Positive Davio Decomposition:

$f(x_1, x_2, \ldots, x_n) = f_{\overline{x_i}} \oplus x_i (f_{\overline{x_i}} \oplus f_{x_i})$

↓ All literals un-complemented. (pure form)

Negative Davio Decomposition:

$f(x_1, \ldots x_n) = f_{x_i} \oplus \overline{x_i} (f_{\overline{x_i}} \oplus f_{x_i})$

↓ All literals complemented.

$f = x_i f_{x_i} \oplus \overline{x_i} f_{\overline{x_i}}$] $\overline{x_i} = 1 \oplus x_i$

$= x_i f_{x_i} \oplus (1 \oplus x_i) f_{\overline{x_i}}$

$= x_i f_{x_i} \oplus f_{\overline{x_i}} \oplus x_i f_{\overline{x_i}}$

$= f_{\overline{x_i}} \oplus x_i (f_{x_i} \oplus f_{\overline{x_i}})$

$f_{x_i} = f_{x_i \overline{x_j}} \oplus x_j (f_{x_i x_j} \oplus f_{x_i \overline{x_j}})$

$f_{\overline{x_i}} = f_{\overline{x_i}\,\overline{x_j}} \oplus x_j (f_{\overline{x_i} x_j} \oplus f_{\overline{x_i}\,\overline{x_j}})$

$f(x_1, x_2, \ldots, x_n) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_3 x_3 \ldots$

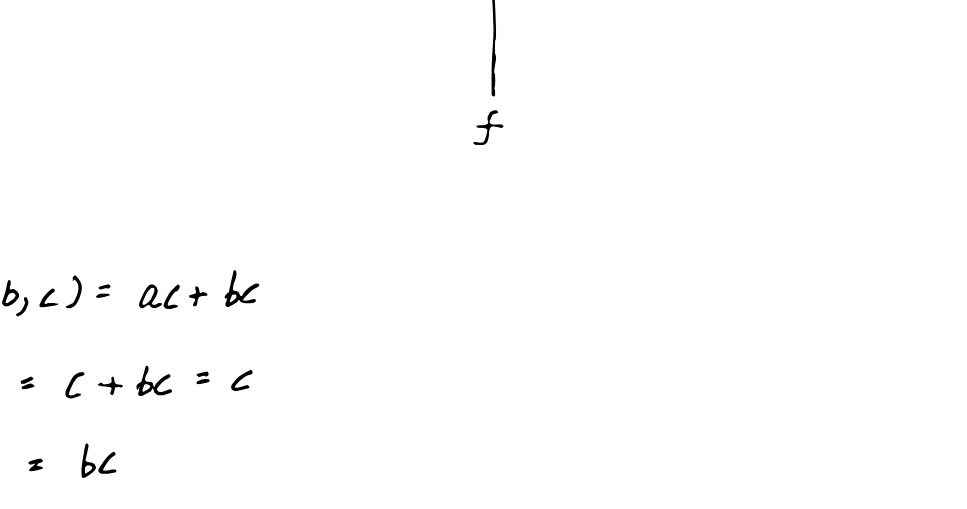
REED MULLER REPRESENTATION → $\oplus\, a_{n+1} x_1 x_2 \oplus a_{n+2} x_1 x_3 \ldots$

$\oplus\, a_r x_1 x_2 x_3 \oplus a_{r+1} x_1 x_2 x_4$

$\oplus \ldots \oplus a_m x_1 x_2 \cdots x_n$

where $a_i \in \{0, 1\}$.

∴ Only AND, XOR Gates needed



a_i → switch

If 1 → on
0 → off

GENERIC

$f(a,b,c) = ac + bc$

$f_a = c + bc = c$

$f_{\bar{a}} = bc$

$\therefore f = f_{\bar{a}} \oplus a(f_{\bar{a}} \oplus f_a)$

$= bc \oplus a(bc \oplus c)$

$= bc \oplus ac \oplus abc$

$f(a,b) = ab + \bar{a}\bar{b}$

$f_{\bar{a}} = \bar{b}$

$f_a = b$

$\therefore f = f_{\bar{a}} \oplus a(f_{\bar{a}} \oplus f_a)$

$= \bar{b} \oplus a(\bar{b} \oplus b)$

$= \bar{b} \oplus a$

$= 1 \oplus b \oplus a$

REED MULLER REPRESENTATION CANNOT HAVE COMPLEMENTED LITERALS.

$f_{\bar{b}} = \bar{a} = 1 \oplus a$

$f_b = a$

$\therefore f = f_{\bar{b}} \oplus b(f_{\bar{b}} \oplus f_b)$

$= 1 \oplus a \oplus b(1 \oplus \underbrace{a \oplus a}_{0})$

$= 1 \oplus a \oplus b$

Intuition — Visualization

$a + b \equiv a \oplus b$ if a, b are orthogonal (disjoint).

a \ bc	00	01	11	10
0	0	0	1	0
1	0	1	1	0

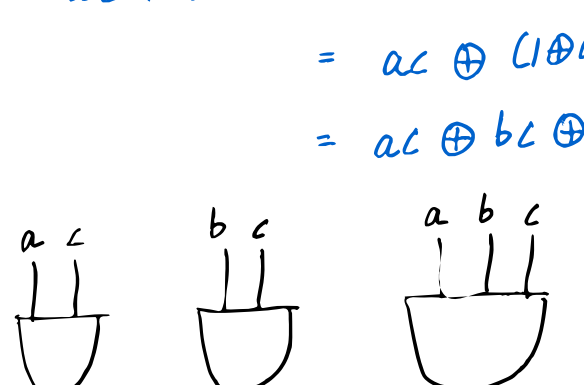
?→ X (we need disjoint groupings)

$\therefore f = ac + bc$ → SOP

$f = ac + \bar{a}bc = ac \oplus \bar{a}bc$

$= ac \oplus (1 \oplus a)bc$

$= ac \oplus bc \oplus abc$ → RM



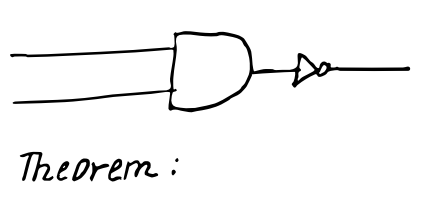
(Reed Muller)
RM forms
Better Testable Circuits
(Used for Quantum Circuits)

Representation:

TRUTH TABLE
SOm
POM
ROBDD
RM
} CANONICAL

SOP
POS
} NON-CANONICAL

AND - INVERT GRAPH: (AIG)

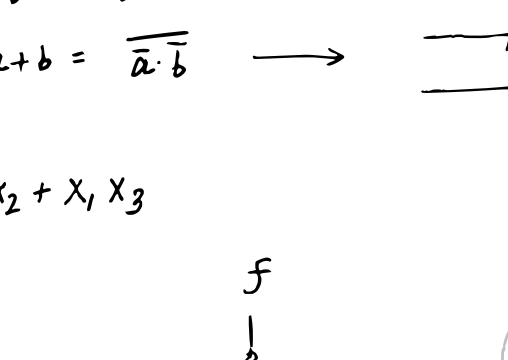


DeMorgan's Theorem:

$\overline{a+b} = \bar{a} \cdot \bar{b}$

$\therefore a + b = \overline{\bar{a} \cdot \bar{b}}$ →

$f = x_1 x_2 + x_1 x_3$

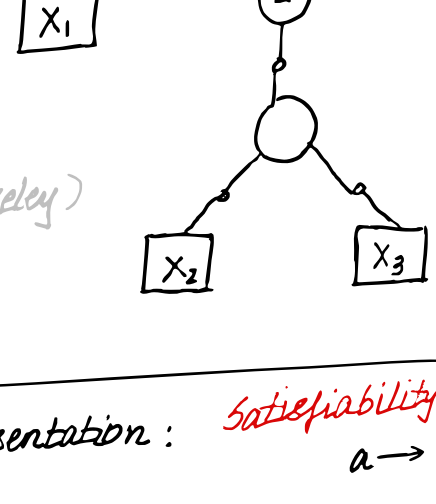


O → AND
o → NOT (Invert)

AIG is not canonical.

$f = x_1 x_2 + x_1 x_3$

$= x_1 (x_2 + x_3)$



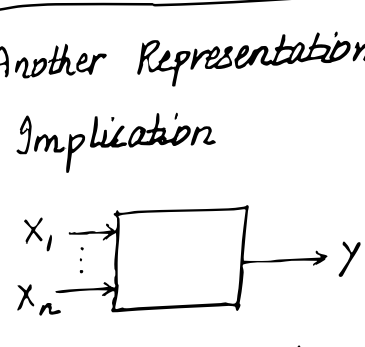
COMPACT
(ABC → UC Berkeley)

No. of AND operations
↑
COST → # NODES
(Ignore NOT)
↓ why?
Because can be converted to NAND-NAND

DELAY
↓
max. path length

Another Representation: Satisfiability Formulation

Implication



$f(x_1, \ldots, x_n, Y)$

$a \to b$

a	b	a → b
0	0	1
0	1	1
1	0	0
1	1	1

NOT:

X	Y
0	1
1	0

$\bar{X} \to Y$

$(\bar{\bar{X}} + Y) = (X + Y) = 1$ ← clause

$X \to \bar{Y}$

$(\bar{X} + \bar{Y}) = (\bar{X} + \bar{Y}) = 1$ ← clause

$\therefore f = (X+Y) \cdot (\bar{X} + \bar{Y}) = 1$ → NOT CANONICAL

DO SAME FOR AND, OR, XOR, XNOR.

AND:

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$AB \to Y$

$(\overline{AB} + Y) = 1$

$\therefore (\bar{A} + \bar{B} + Y) = 1$ → (i)

$\bar{A} \to \bar{Y}$

$(\bar{\bar{A}} + \bar{Y}) = 1$

$\therefore (A + \bar{Y}) = 1$ → (ii)

$\bar{B} \to \bar{Y}$

$(\bar{\bar{B}} + \bar{Y}) = 1$

$\therefore (B + \bar{Y}) = 1$ → (iii)

(i), (ii) & (iii) must be satisfied simultaneously.

↓

$(\bar{A} + \bar{B} + Y)(A + \bar{Y})(B + \bar{Y}) = 1$ → NOT CANONICAL

OR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

$\bar{A}\bar{B} \to \bar{Y}$

$(\overline{\bar{A}\bar{B}} + \bar{Y}) = 1$

$\therefore (A + B + \bar{Y}) = 1$

$A \to Y$ $B \to Y$

$\bar{A} + Y = 1$ $\bar{B} + Y = 1$

$\therefore (A + B + \bar{Y})(\bar{A} + Y)(\bar{B} + Y) = 1$ → NOT CANONICAL

XOR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

$\bar{A}\bar{B} \to \bar{Y}$

$(\overline{\bar{A}\bar{B}} + \bar{Y}) = 1 \to (A + B + \bar{Y}) = 1$

$A\bar{B} \to Y$

$(\overline{A\bar{B}} + Y) = 1 \to (\bar{A} + B + Y) = 1$

$\bar{A}B \to Y$

$(\overline{\bar{A}B} + Y) = 1 \to (A + \bar{B} + Y) = 1$

$AB \to \bar{Y}$

$(\overline{AB} + \bar{Y}) = 1 \to (\bar{A} + \bar{B} + \bar{Y}) = 1$

$\therefore (A + B + \bar{Y})(\bar{A} + B + Y)(A + \bar{B} + Y)(\bar{A} + \bar{B} + \bar{Y}) = 1$

XNOR:

A	B	Y
0	0	1
1	0	0
0	1	0
1	1	1

$\bar{A}\bar{B} \to Y$

$(\overline{\bar{A}\bar{B}} + Y) = 1 \to (A + B + Y) = 1$

$A\bar{B} \to \bar{Y}$

$(\overline{A\bar{B}} + \bar{Y}) = 1 \to (\bar{A} + B + \bar{Y}) = 1$

$\bar{A}B \to \bar{Y}$

$(\overline{\bar{A}B} + \bar{Y}) = 1 \to (A + \bar{B} + \bar{Y}) = 1$

∴ NOT → 2 clauses

AND, OR → 3 clauses ← NAND, NOR

XOR, XNOR → 4 clauses

NAND

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

$AB \to \bar{Y}$

$(\overline{AB} + \bar{Y}) \to (\bar{A} + \bar{B} + \bar{Y}) = 1$

$\bar{A} \to Y$

$(\bar{\bar{A}} + Y) \to (A + Y) = 1$

$\bar{B} \to Y$

$(\bar{\bar{B}} + Y) \to (B + Y) = 1$

$\therefore \underbrace{(\bar{A} + \bar{B} + \bar{Y})(B + Y)(A + Y)}_{\text{3 clauses}} = 1$

NOR

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

$\bar{A}\bar{B} \to Y$

$(\overline{\bar{A}\bar{B}} + Y) \to (A + B + Y) = 1$

$A \to \bar{Y}$

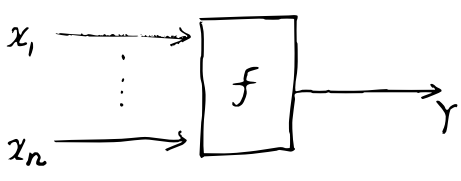
$(\bar{A} + \bar{Y}) \to (\bar{A} + \bar{Y}) = 1$

$B \to \bar{Y}$

$(\bar{B} + \bar{Y}) \to (\bar{B} + \bar{Y}) = 1$

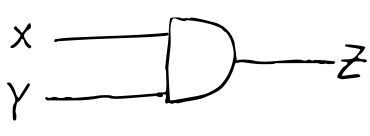
$\therefore \underbrace{(A + B + Y)(\bar{B} + \bar{Y})(\bar{A} + \bar{Y})}_{\text{3 clauses}} = 1$

Satisfiability Formulation:



$g(x_1, \ldots, x_n, Y)$

e.g. x —▷o— Y $\Rightarrow (X+Y)(\bar{X}+\bar{Y}) = 1$



$\Rightarrow (\bar{X}+\bar{Y}+Z)(X+\bar{Z})(Y+\bar{Z}) = 1$

clause

POS

Conjunctive Normal Form (CNF)

conjunction of disjunctions

SOP → Disjunctive Normal form (DNF)

Satisfiability (SAT) Problem:

For what values of variables is an expression satisfied?

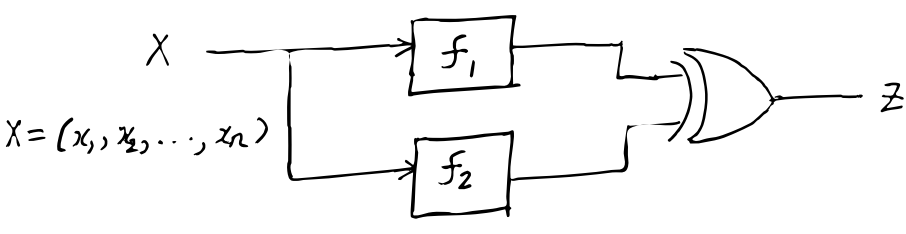
i.e. say $(X+\bar{Z})(Y+\bar{Z})(\bar{X}+\bar{Y}+Z) = 1$

FOR WHAT X, Y, Z?

(1, 1, 1 works here)

Equivalence Checking:

To check if two functions are equivalent or not, we can use XOR.

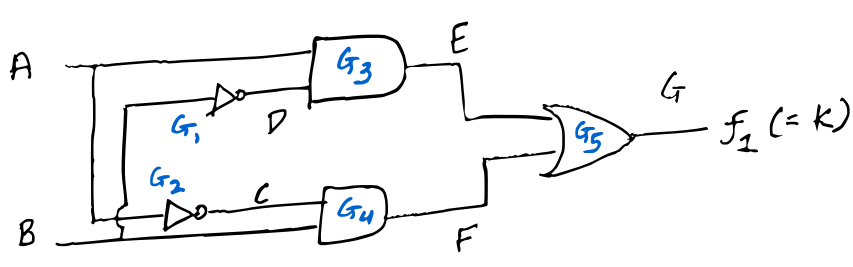


If $f_1 \equiv f_2$, $f_1 \oplus f_2 = 0 \quad \forall X$

Thus, if we find even 1 X for which $f_1 \oplus f_2 = 1$, we can say that the two functions are not equivalent.

i.e. If we find a solution to the SAT problem for $f_1 \oplus f_2$, then f_1, f_2 are not equivalent.

e.g. Let $f_1 = \bar{A}B + A\bar{B} = K$



SAT problem for f_1:

$G_1 : (B+D)(\bar{B}+\bar{D}) = 1$ — 2

$G_2 : (A+C)(\bar{A}+\bar{C}) = 1$ — + 2

$G_3 : (\bar{A}+\bar{D}+E)(A+\bar{E})(D+\bar{E}) = 1$ — + 3

$G_4 : (\bar{B}+\bar{C}+F)(B+\bar{F})(C+\bar{F}) = 1$ — + 3

$G_5 : (E+F+\bar{G})(\bar{E}+G)(\bar{F}+G) = 1$ — + 3

13 clauses

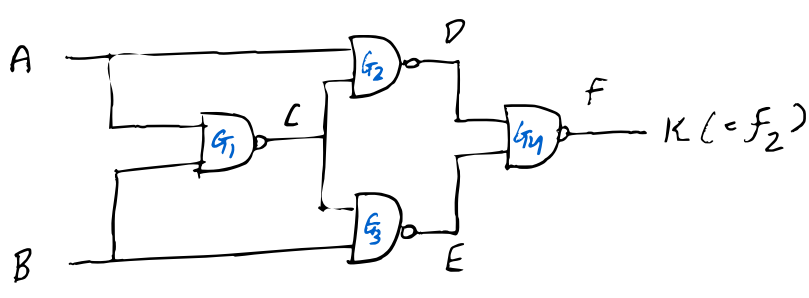
$G_1 \cdot G_2 \cdot G_3 \cdot G_4 \cdot G_5 = 1$

Π (13 clauses) $= 1$

Through satisfiability and equivalence, we can debug a circuit.

???

Let f_2 be



SAT problem:

$G_1 \cdot G_2 \cdot G_3 \cdot G_4 = 1$

each has 3 clauses → $4 \times 3 = 12$ clauses

Continuity ???

Solving SAT problem: (DPLL Method:

Davis-Putnam ← DP → Introduced

Logemann-Loveland ← LL → Refined ???)

based on backtracking.

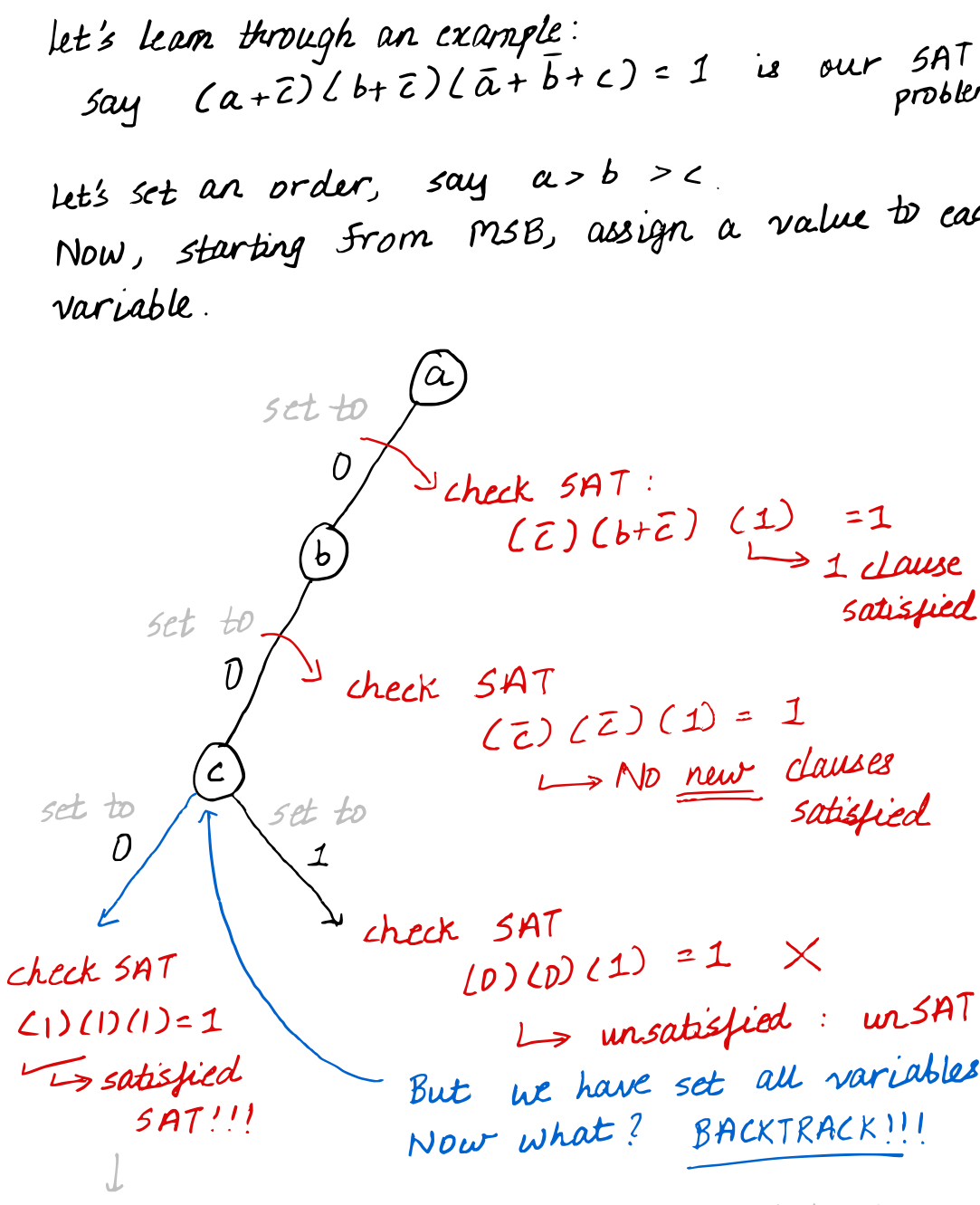
what? Let's see.

let's learn through an example:

say $(a+\bar{c})(b+\bar{c})(\bar{a}+\bar{b}+c) = 1$ is our SAT problem.

Let's set an order, say $a > b > c$.

Now, starting from MSB, assign a value to each variable.



If it would have been unSAT, we would have backtracked to b, then to a until we would have exhausted all possibilites.

↳ i.e. Worst Case Backtracking : 2^n for n variables.

Can we improve this?

WILL CONTINUE TOMORROW

Q.] Is usage of SAT, unSAT being done in correct context?

Representation:

- Truth Table
- SOm
- POM
- ROBDD
- RM

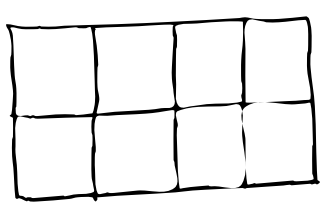
} Canonical

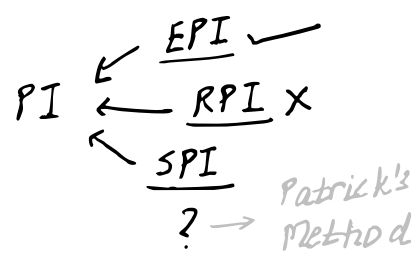
- SOP
- POS
- AIG

} Non-canonical

Optimization:

K-Map





Metric → Cost : Min{ # Literals + # terms }

Delay : min{ max{ # literals in a term } + # terms }

Power : similar to cost

Can be visualized till 6 variables (3D), beyond that, it is difficult. ← Limitation of K-Map.

Different Approach?

Quine, McCluskey → Tabular Method (fairly scalable)

e.g. $f(a,b,c) = \Sigma(0,1,3,5,6,7)$

Place numbers with same number of 1's in the same group.

G0	0	000	(0, 1)	00 –	(1,3,5,7) – – 1
G1	1	001	(1, 3)	0 – 1	(1,5,3,7) – – 1
G2	3	011	(1, 5)	– 0 1	
	5	101	(3, 7)	– 1 1	
	6	110	(5, 7)	1 – 1	
G3	7	111	(6, 7)	1 1 –	

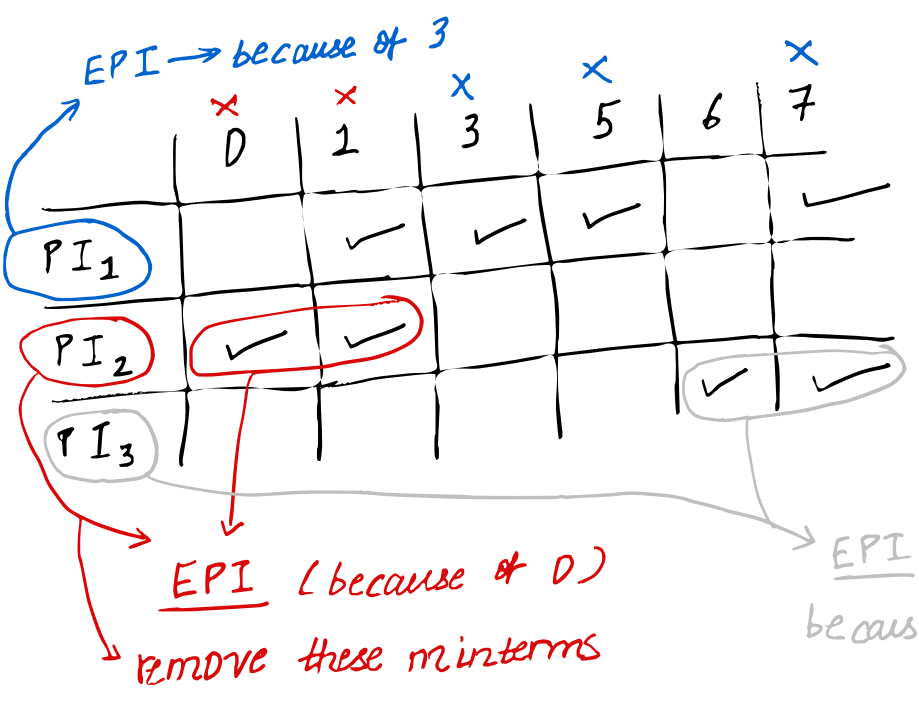
PI_2 = 00 –; PI_1 = – – 1; PI_3 = 1 1 –

Pairing only possible between adjacent groups i.e. (G_0, G_1), (G_1, G_2) & (G_2, G_3)

ALL unpaired groupings → PI_s

$\therefore f(a,b,c) = PI_1 + PI_2 + PI_3$

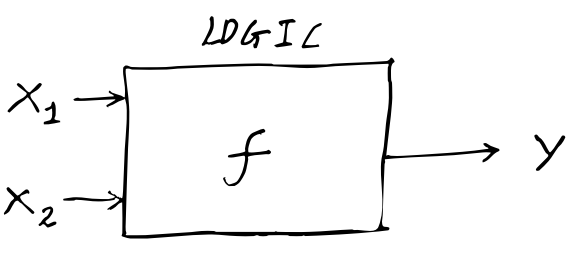
$= c + \bar{a}\bar{b} + ab$



If you get SPI : Patrick's Method

or

Integer Linear Programming (ILP)



What we have done?

→ How to Represent

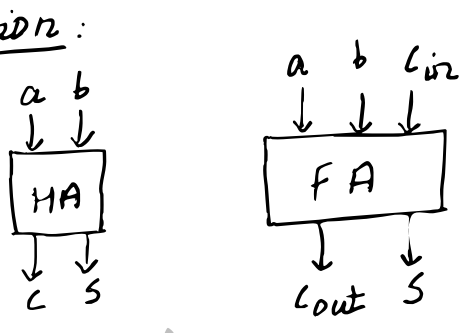
→ How to Minimize

→ How to check equivalence

Arithmetic functions :

- ADDITION
- SUBTRACTION
- MULTIPLICATION
- DIVISION

Addition :



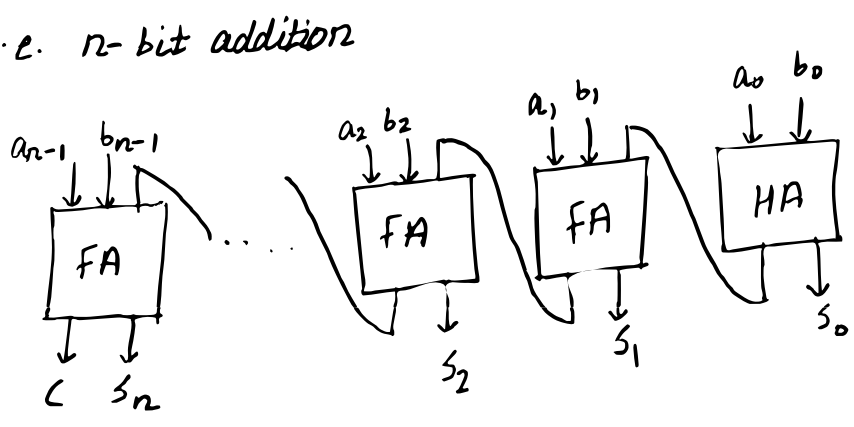
(1-bit adder)

Now, what if we want to add

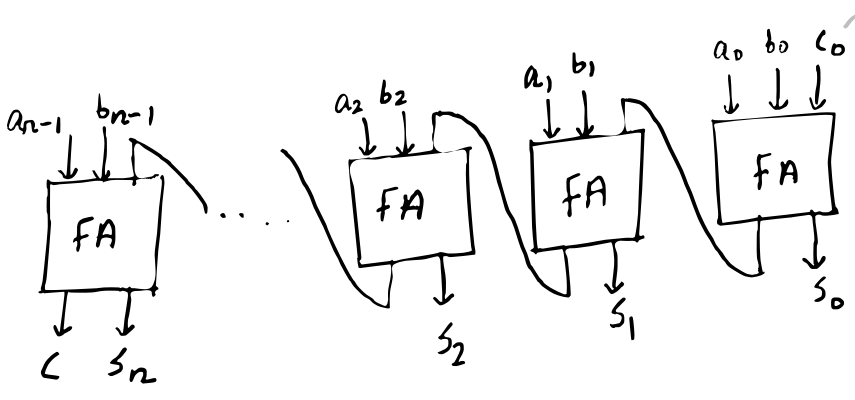
A a_{n-1} a_{n-2} ..... a_1 a_0

B b_{n-1} b_{n-2} ..... b_1 b_0

i.e. n-bit addition



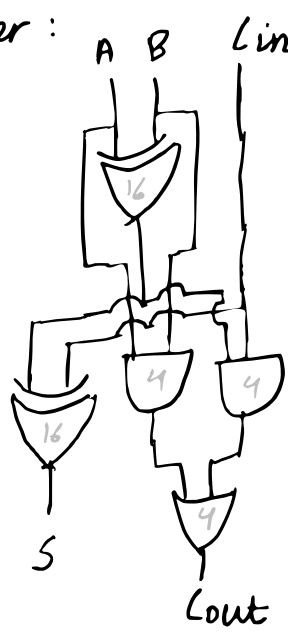
If we want it to do both addition and subtraction,



If cost of 1 FA = δ

Total cost = $n\delta$

Full Adder :



cost :

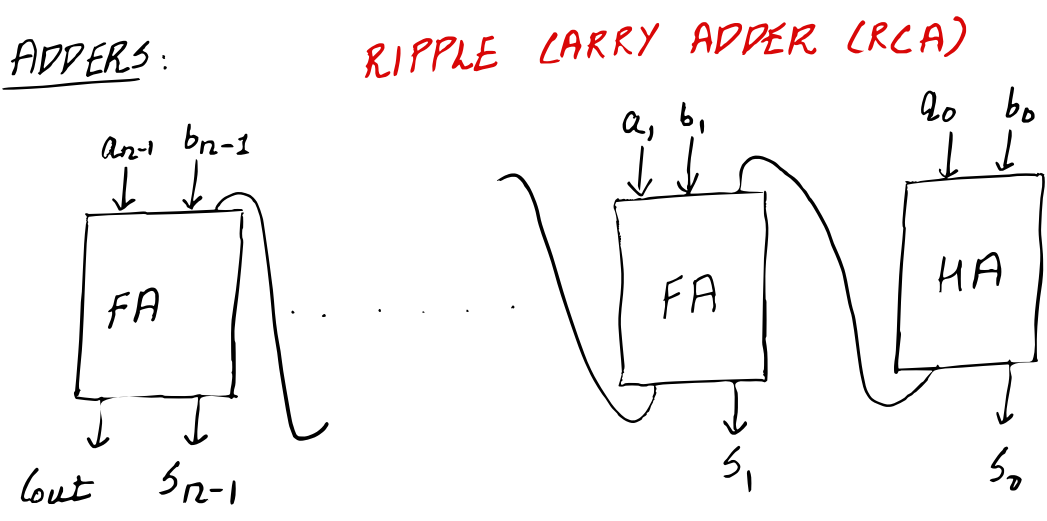
44 switches for 1 Full Adder

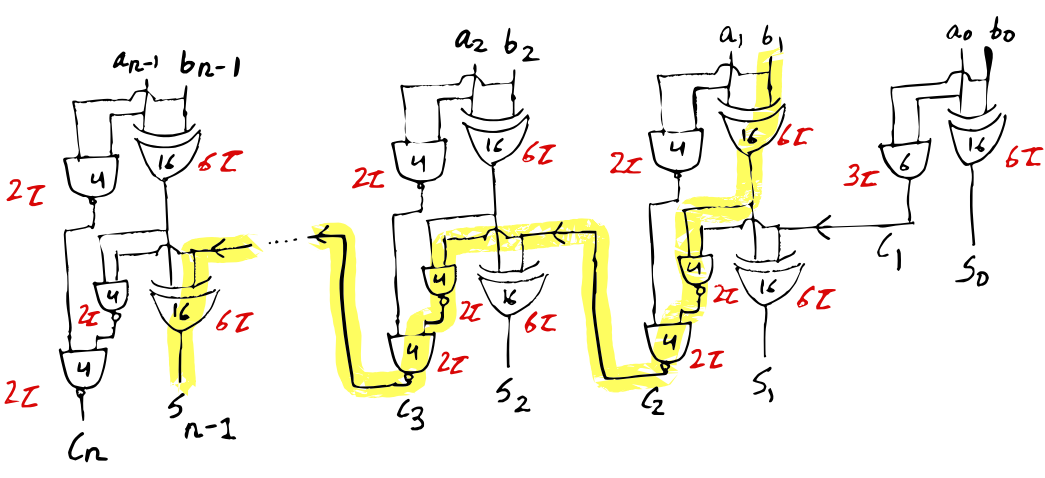
∴ total cost :

$44n$

ADDERS:

RIPPLE CARRY ADDER (RCA)





total cost = $22 + 44(n-1)$

Delay = max. path length

$= XOR_1 + 2 \times NAND_{(1 \to n-2)} + XOR_{n-1}$

$= 6I + 2 \times 2I(n-2) + 6I$

$= 12I + 4nI - 8I$

$= \underline{4(n+1)I}$

All adders start simultaneously.
So, delay comes due to sequential inputs. (carry)

Why 0^{th}-bit adder not included in adder?

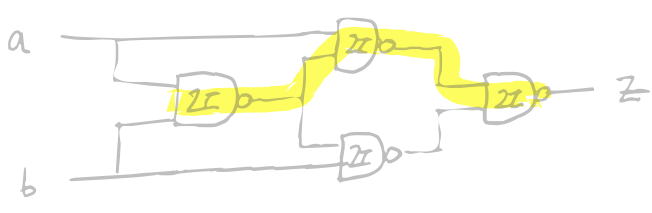
↳ Delay from 0-bit : 3I

Delay from 1^{st}-bit XOR : 6I

We take max.

XOR:

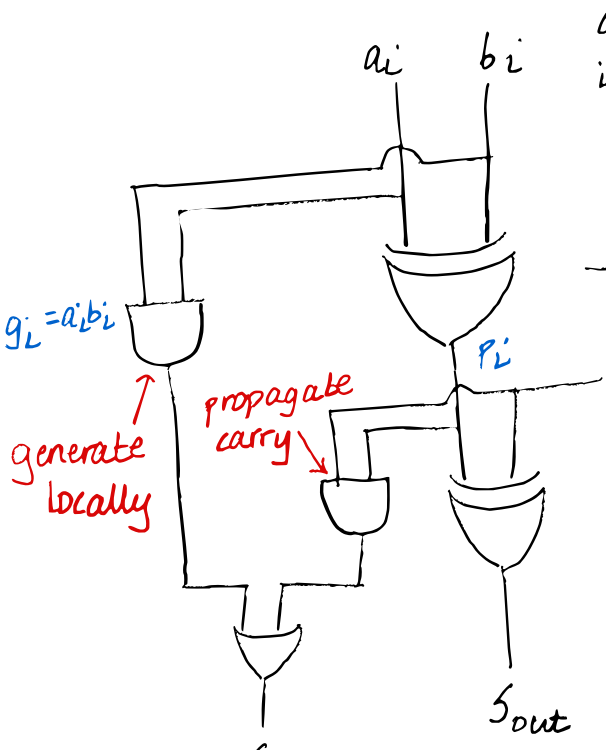
delay : 6I



Can we reduce delay?

Yes, why wait for C_{in} to come when we can calculate it ourselves?

C_{in}	0	0	0	0	1	1	1	1
a_i	0	0	1	1	0	0	1	1
b_i	0	1	0	1	0	1	0	1
C_0	0	0	0	1	0	1	1	1



C_{i+1}

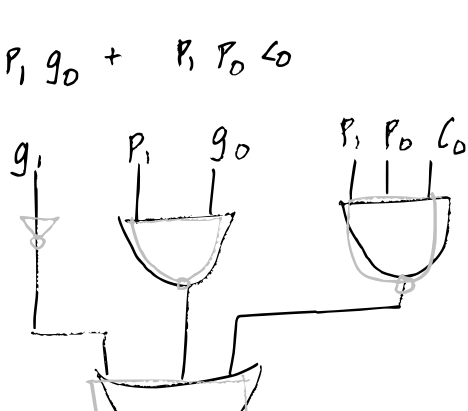
$\therefore C_{i+1} = g_i + C_i P_i$

$\therefore C_1 = g_0 + C_0 P_0 \longrightarrow$ (In case of adder $\to C_0 = 0$)

$C_2 = g_1 + C_1 P_1$

$= g_1 + (g_0 + C_0 P_0) P_1$

$= g_1 + P_1 g_0 + P_1 P_0 C_0$

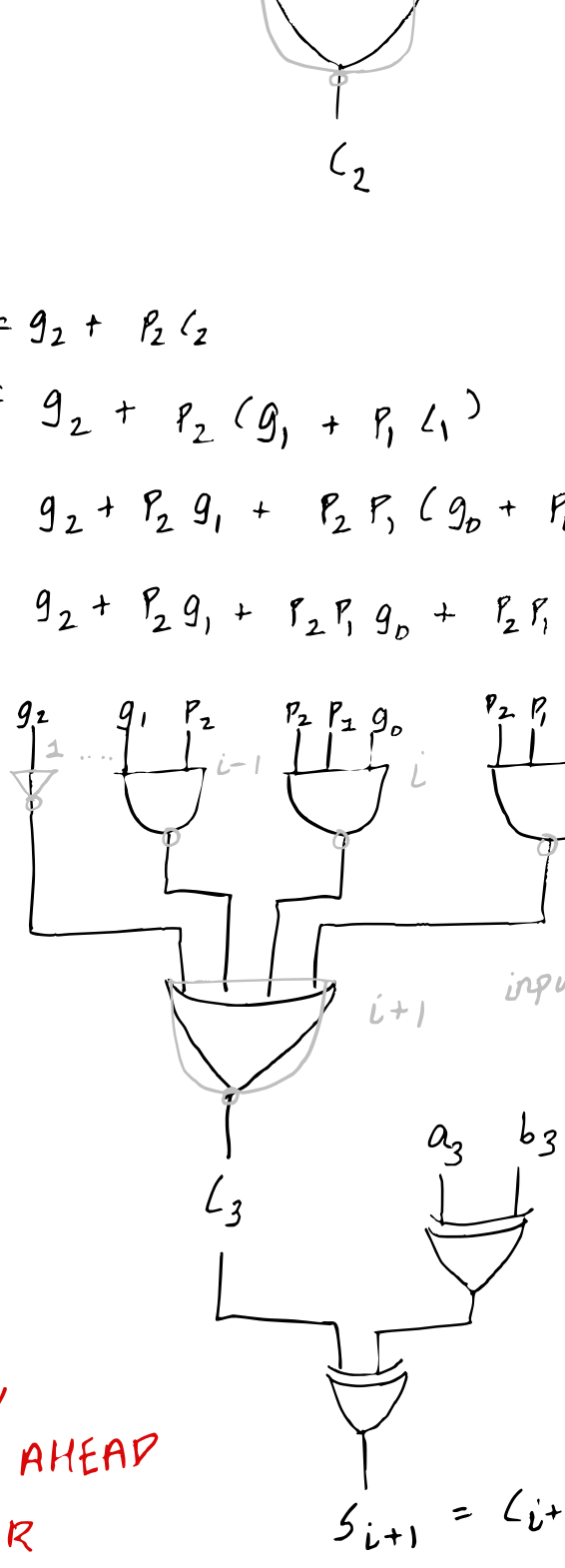


$C_3 = g_2 + P_2 C_2$

$= g_2 + P_2 (g_1 + P_1 C_1)$

$= g_2 + P_2 g_1 + P_2 P_1 (g_0 + P_0 C_0)$

$= g_2 + P_2 g_1 + P_2 P_1 g_0 + P_2 P_1 P_0 C_0$



CARRY LOOK AHEAD ADDER

$S_{i+1} = C_{i+1} \oplus a_{i+1} \oplus b_{i+1}$

Total cost = $\sum_{K=1}^{i+1} 2(K) + 2(i+1)$

↓ for C_{i+1}

$= (i+1)(i+2) + 2(i+1)$

$= (i+1)(i+4)$

$\therefore$ Total cost for i^{th} cell = 2 XOR + C_{i+1}

$= 2 \times 16 + (i+1)(i+4)$

$= 32 + (i+1)(i+4)$

$g_i = a_i b_i \longrightarrow$ cost : 6

$P_i = a_i + b_i \longrightarrow$ cost : 6

n g_i's and n P_i's $\longrightarrow$ $6n + 6n = \underline{12n}$

$\therefore$ Delay = 3I + (i+1)I + (i+1)I + 6I

3I ↓ from P_i ; (i+1)I AND ; (i+1)I OR ; 6I XOR

For i^{th} cell.

But we only take max. → for $(n-1)^{th}$ cell

$\therefore$ Delay = 3I + nI + nI + 6I

$= (2n + 9)I$

Delay reduced from $\alpha\, 4n$ to $\alpha\, 2n$.

Earlier, TTL was used, in TTL, delay does not depend on no. of inputs. Thus, delay was fixed to 3T for above circuit → considered very fast. But now, with CMOS, we lost that advantage.

ADDER

- RCA ⟶ Slowest $\propto 4n$
- CLA ⟶ $\propto 2n$ (CMOS)

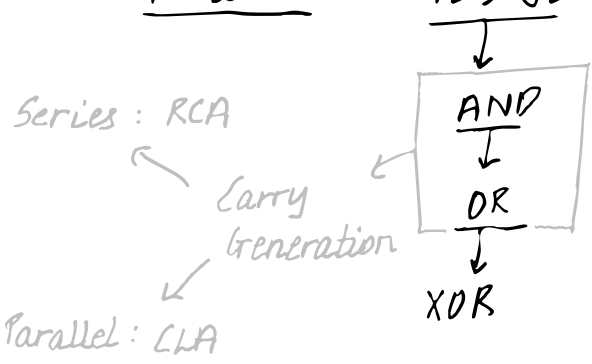
↑ Fast for TTL/BJT

↓

Delay $\propto$ # of Levels

4 - Levels

P_i, g_i (AND/OR)



$g_i = a_i \cdot b_i$

$P_i = a_i + b_i$

$= a_i \oplus b_i$

Fast Large Adder

PREFIX - ADDER

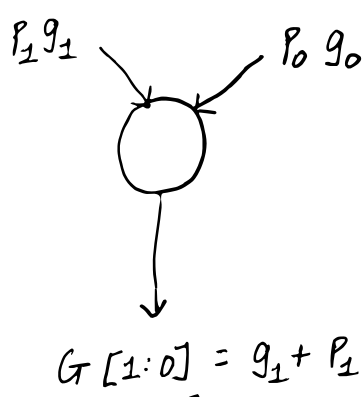
Group :

a_1 a_0

b_1 b_0

$g_1 = a_1 b_1$ $\quad$ $g_0 = a_0 b_0$

$P_1 = a_1 \oplus b_1$ $\quad$ $P_0 = a_0 \oplus b_0$

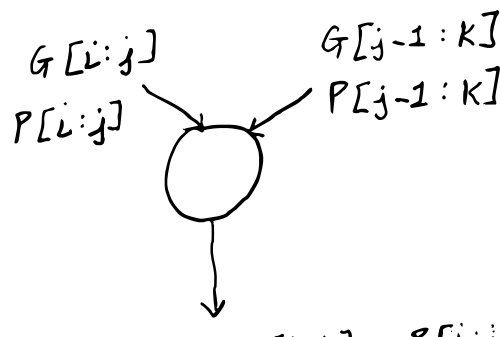


$G[1:0] = g_1 + P_1 g_0$

$P[1:0] = P_1 \cdot P_0$

Instead of making a group of all like in CLA, make smaller groups (say, of two).

$[i:K]$ ⟶



$G[i:K] = G[i:j] + P[i:j]\, G[j-1:K]$

$P[i:K] = P[i:j] \cdot P[j-1:K]$

$\therefore \quad S_i = C_i \oplus a_i \oplus b_i$

$C_{i+1} = G[i:K] + P[i:K] \cdot C_K$

We have C_0,

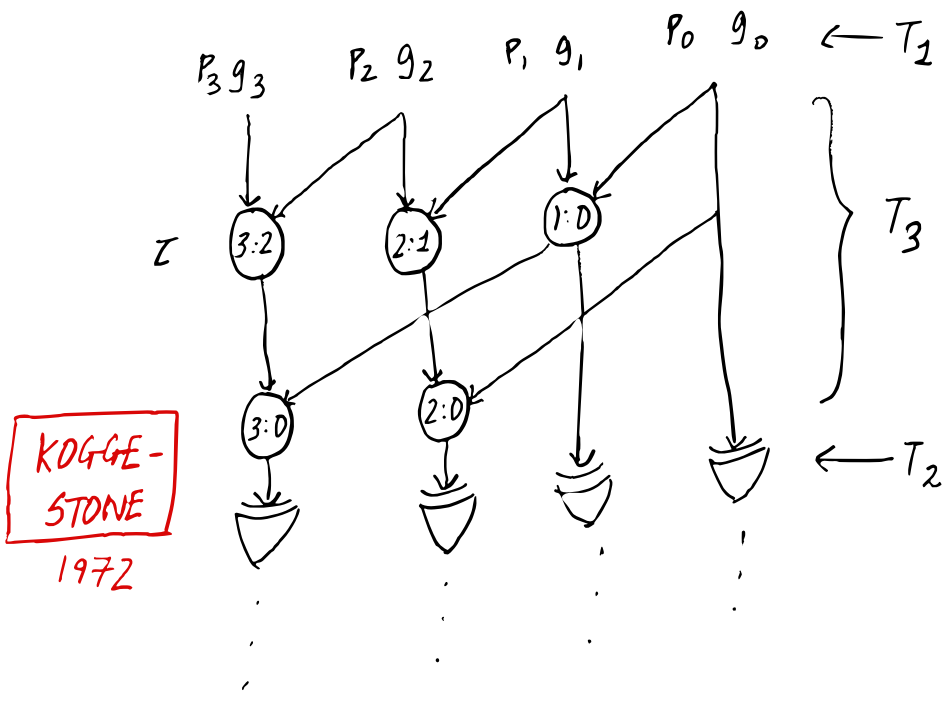
Thus, $C_{i+1} = G[i:0] + P[i:0]\, C_0$

example:

Consider 4-bit

$A = a_3 \; a_2 \; a_1 \; a_0$

$B = b_3 \; b_2 \; b_1 \; b_0$



$\therefore$ Delay $= T_1 + T_2 + T_3$

T_1, T_2 → constant

$T_3 = I \times \log n$ → no. of levels (max.)

$\therefore$ **Delay $\propto \log n$**

Sometimes referred to as logarithmic adder.

Other architectures:

⟶ BRENT-KUNG ⟶ $2 \log n$ levels (Depth)

1960

Midsem Practice:

Distributivity:

$a(b+c) = ab + ac$

★ $a + bc = (a+b)\cdot(a+c)$

Thm:

adsorption

$a + \bar{a}b = (a+\bar{a})(a+b) = \underline{a+b}$

$a\cdot(\bar{a}+b) = a\cdot\bar{a} + a\cdot b = \underline{ab}$

consensus

$ab + \bar{a}c + bc$

$= (a+\bar{a}c+bc)(b+\bar{a}c+bc)$

$= (a+c+bc)(\bar{a}c+b)$

$= (a+c)(\bar{a}c+b)$

$= (a+c)(b+c)(\bar{a}+b)$

$=$

$(ab+\bar{a}c+b)(ab+\bar{a}c+c)$

$(b+\bar{a}c)(ab+c)$

$(b+\bar{a}c)(ab+c) = ab + bc + \bar{a}c$

$ab + a'c + bc = ab + a'c + abc + a'bc$ ($bc(a+a')$)
$= ab + a'c$

$AB + BC' + AC = \underline{BC' + AC}$

consensus:

$ab + a'c + bc = ab + a'c$

$(a+b)(a'+c)(b+c) = (a+b)(a'+c)$

cost: AND – OR

$2\times[\text{no. of literals} + \text{no. of product terms}]$

$AB + CD$

$[4+2]\times 2 = 6\times 2 = 12$

AND OR

$AB + AC$

$[3+2]\times 2 = 5\times 2 = 10$?

AND – OR

$A\cdot(B+C)$

$= (2+3)\times 2 = 10$

$AB + AC + BC$?

$(3+3)\times 2 = 12$?

NOT 18?

EE 224 Midsem 1

Q1] $(AB+C+D)(C+\bar{D}) = \overline{\bar{C}(\bar{A}+\bar{B}+D)}$

$= ABC + AB\bar{D} + C + C\bar{D} + CD$

$= ABC + AB\bar{D} + C(1+D+\bar{D})$ — absorption ($1+D+\bar{D} = 1$)

$= ABC + AB\bar{D} + C$

$= AB\bar{D} + (AB+1)C$ — absorption

$= AB\bar{D} + C$

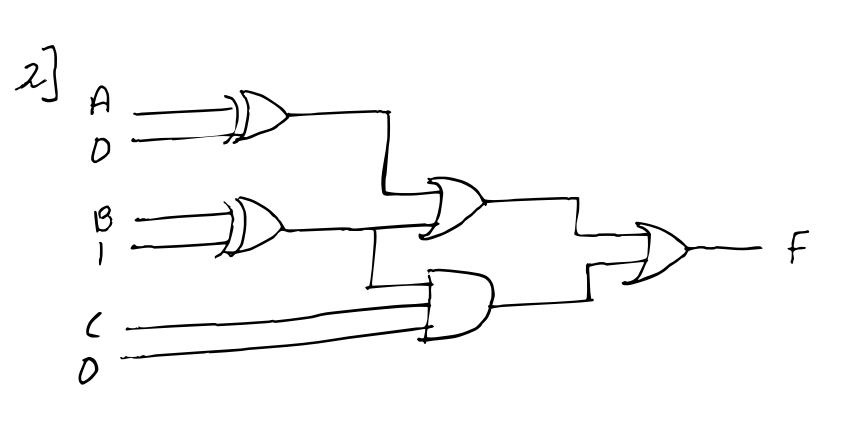
$= \overline{\overline{AB\bar{D}+C}} = \overline{\bar{C}\cdot(\overline{AB\bar{D}})}$ — De Morgan's

involution $= \overline{\bar{C}\cdot(\bar{A}+\bar{B}+D)}$ — De Morgan's

$=$ RHS.

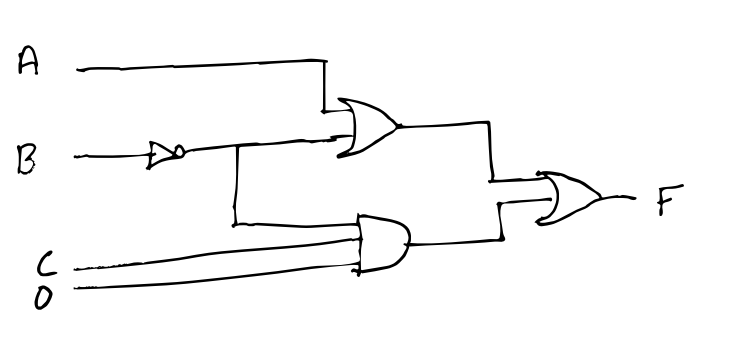
Hence Proved!!!

2]



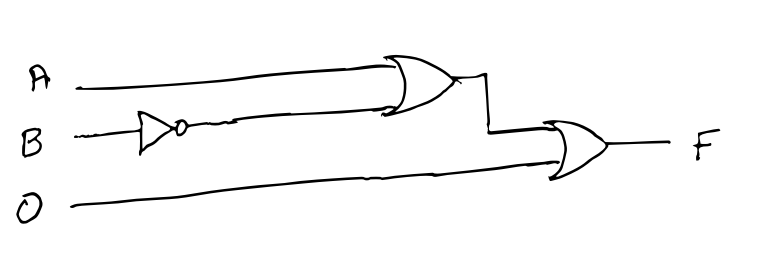
$A\oplus 0 = A \qquad B\oplus 1 = \bar{B}$

∴



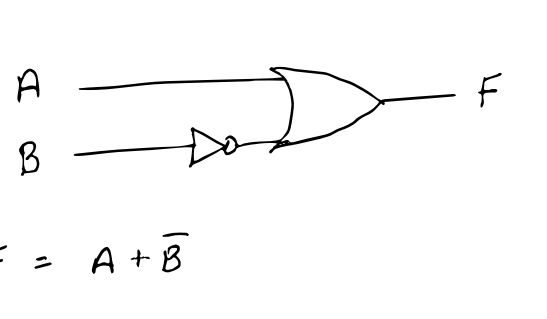
$0\cdot C\cdot\bar{B} = 0$

∴



$A + \bar{B} + 0 = A + \bar{B}$

∴



∴ $f = A + \bar{B}$

Q.3] r, s unknown positive integers

$w, x, y, z \geq 0$

$w, x \leq r \qquad y, z \leq s$

$(w1.x)_r = (y2.z)_s$

∴ $w\cdot r^1 + 1\cdot r^0 + x\cdot r^{-1} = y\cdot s^1 + 2\cdot s^0 + z\cdot s^{-1}$ ($\in \mathbb{Z}$, $\in \mathbb{Z}$)

∴ $wr + 1 = ys + 2$

$\frac{x}{r} = \frac{z}{s} \Rightarrow xs = rz$

Q.4] $f(a,b,c,d) = (\bar{b}+c+\bar{d})(\bar{a}+\bar{b}+\bar{d})(a+\bar{c}+d)$

Involution:

$f = \bar{\bar{f}} = \overline{\overline{(\bar{b}+c+\bar{d})(\bar{a}+\bar{b}+\bar{d})(a+\bar{c}+d)}}$

$= \overline{(\overline{\bar{b}+c+\bar{d}}) + (\overline{\bar{a}+\bar{b}+\bar{d}}) + (\overline{a+\bar{c}+d})}$

$= \overline{(b\bar{c}d + abd + \bar{a}c\bar{d})}$

$= \overline{((a+\bar{a})(b\bar{c}d) + ab(c+\bar{c})d + \bar{a}(b+\bar{b})c\bar{d})}$

$= \overline{(ab\bar{c}d + \bar{a}b\bar{c}d + abcd + ab\bar{c}d + \bar{a}bc\bar{d} + \bar{a}\bar{b}c\bar{d})}$

$= \overline{(abcd + ab\bar{c}d + \bar{a}b\bar{c}d + \bar{a}bc\bar{d} + \bar{a}\bar{b}c\bar{d})}$

$= (\overline{abcd})(\overline{ab\bar{c}d})(\overline{\bar{a}b\bar{c}d})(\overline{\bar{a}bc\bar{d}})(\overline{\bar{a}\bar{b}c\bar{d}})$

$f(a,b,c,d) = (\bar{a}+\bar{b}+\bar{c}+\bar{d})(\bar{a}+\bar{b}+c+\bar{d})(a+\bar{b}+c+\bar{d})(a+\bar{b}+\bar{c}+d)(a+b+\bar{c}+d)$

Product of Maxterm form

Q.5] $f(A,B,C) = (A+\bar{B})C + \bar{A}BC$

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

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

$= ABC + A\bar{B}C + A\bar{B}C + \bar{A}\bar{B}C + \bar{A}BC$

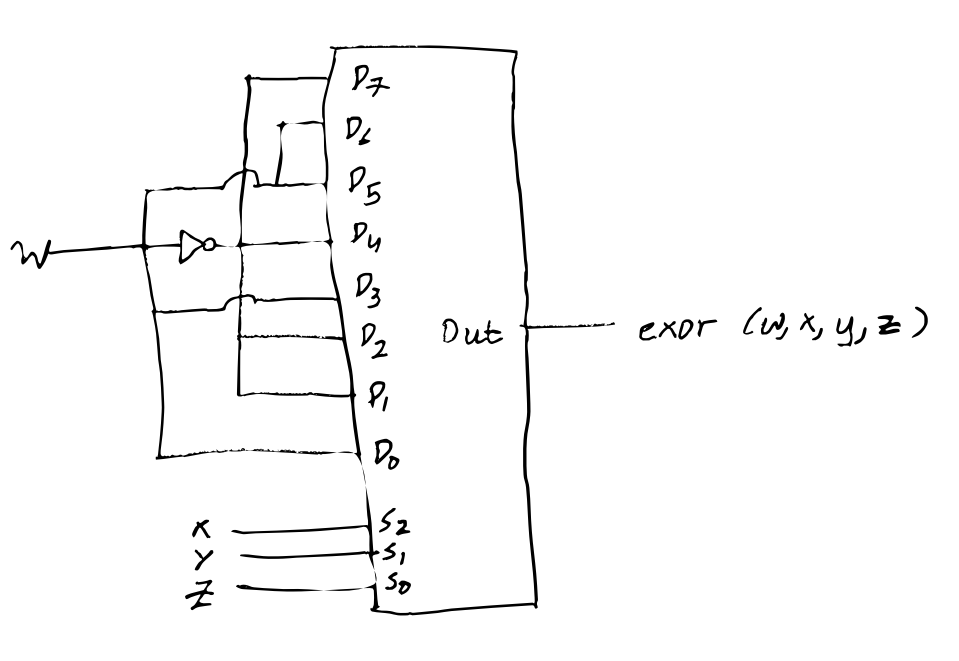
$f(A,B,C) = ABC + A\bar{B}C + \bar{A}\bar{B}C + \bar{A}BC$

Sum of min-term form

Q.6] $exor(w,x,y,z) = w\oplus x\oplus y\oplus z$

Truth Table:

	w	x	y	z	exor(w,x,y,z)	
D_0	0	0	0	0	0	$= w$
	1	0	0	0	1	
D_1	0	0	0	1	1	$= \bar{w}$
	1	0	0	1	0	
D_2	0	0	1	0	1	$= \bar{w}$
	1	0	1	0	0	
D_3	0	0	1	1	0	$= w$
	1	0	1	1	1	
D_4	0	1	0	0	1	$= \bar{w}$
	1	1	0	0	0	
D_5	0	1	0	1	0	$= w$
	1	1	0	1	1	
D_6	0	1	1	0	0	$= w$
	1	1	1	0	1	
D_7	0	1	1	1	1	$= \bar{w}$
	1	1	1	1	0	



EE 224 Midsem 2

1] $f(a,b,c,d) = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{c}d + bcd + ac$

$f_a = c + bcd = c(1+bd) = c$ //

$f_{\bar{a}} = \bar{b}\bar{c} + \bar{c}d + bcd$

$= \bar{b}\bar{c}(d+\bar{d}) + (b+\bar{b})\bar{c}d + bcd$

$= \bar{b}\bar{c}d + \bar{b}\bar{c}\bar{d} + b\bar{c}d + \bar{b}\bar{c}d + bcd$

$= \bar{b}\bar{c}d + \bar{b}\bar{c}\bar{d} + b\bar{c}d + bcd$

$= \bar{b}\bar{c}(d+\bar{d}) + bd(c+\bar{c})$

$= \bar{b}\bar{c} + bd$ //

$f_{\bar{a}b} = d$ //

$f_{\bar{a}\bar{b}} = \bar{c} = 1\oplus c$ //

∴ $f = f_{\bar{a}} \oplus a(f_{\bar{a}}\oplus f_a)$

$= f_{\bar{a}\bar{b}} \oplus b(f_{\bar{a}\bar{b}}\oplus f_{\bar{a}b}) \oplus a(f_{\bar{a}\bar{b}} \oplus b(f_{\bar{a}\bar{b}}\oplus f_{\bar{a}b}) \oplus f_a)$

∴ $f = 1\oplus c \oplus b(1\oplus c\oplus d) \oplus a(1\oplus c\oplus b(1\oplus c\oplus d) \oplus c)$

$= 1\oplus c\oplus b\oplus bc\oplus bd \oplus a(1\oplus c\oplus b\oplus bc\oplus bd\oplus c)$

$\oplus a(1\oplus c\oplus c \oplus b\oplus bc\oplus bd)$ ($c\oplus c = 0$)

$\oplus a(1\oplus 0\oplus b\oplus bc\oplus bd)$ ($1\oplus 0 = 1$)

$= 1\oplus c\oplus b\oplus bc\oplus bd \oplus a(1\oplus b\oplus bc\oplus bd)$

$= 1\oplus a\oplus b\oplus c\oplus ab\oplus bc\oplus bd\oplus abc\oplus abd$

Q.2] $\overline{f(a,b,c)} = \bar{a}\bar{c} + \bar{c}b + b\cdot a + a\cdot c$

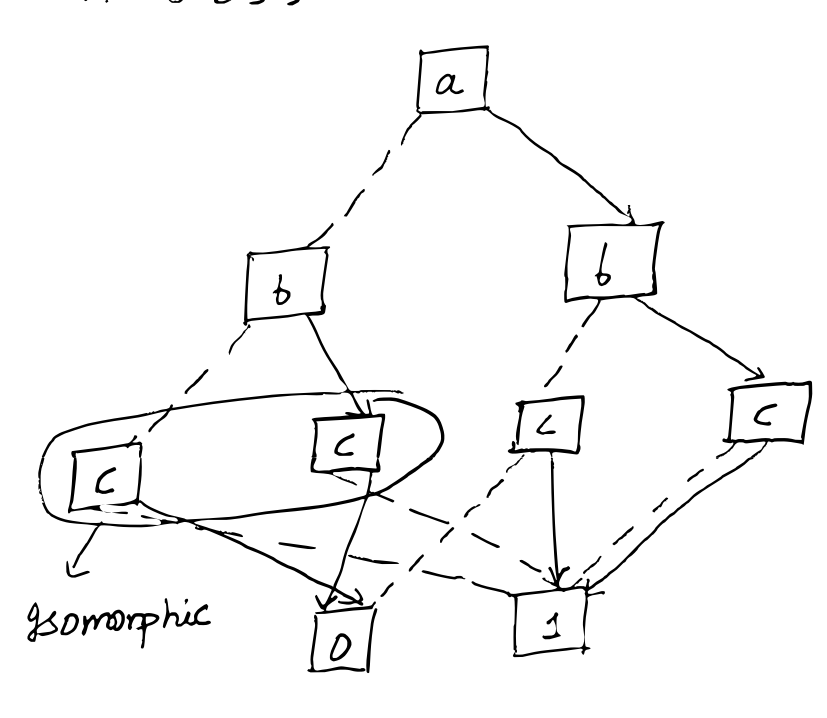
$= \bar{a}(b+\bar{b})\bar{c} + (a+\bar{a})b\bar{c} + ab(c+\bar{c}) + a(b+\bar{b})c$

$= \bar{a}b\bar{c} + \bar{a}\bar{b}\bar{c} + ab\bar{c} + \bar{a}b\bar{c} + abc + ab\bar{c} + abc + a\bar{b}c$

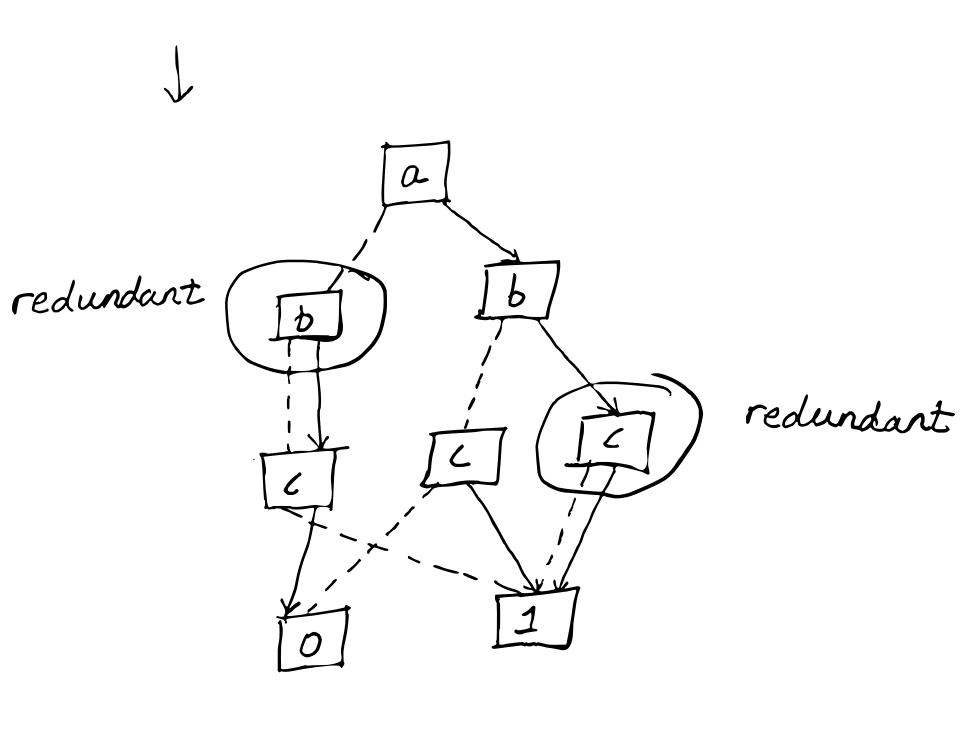
$= \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + a\bar{b}c + ab\bar{c} + abc$

assume $a > b > c \rightarrow$ order

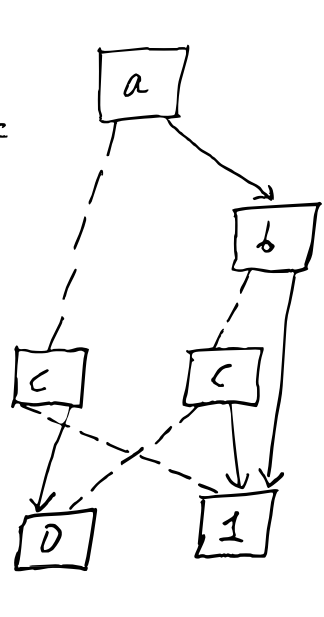
∴ $\overline{f(a,b,c)}$:



↓

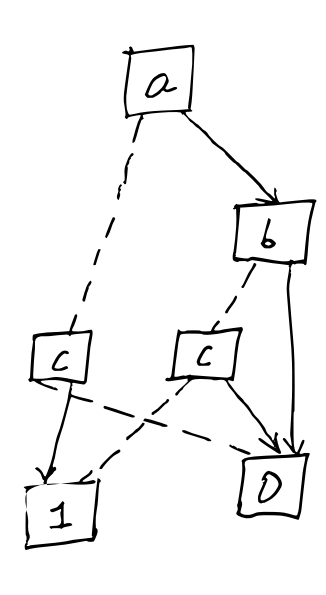


$\overline{f(a,b,c)} =$

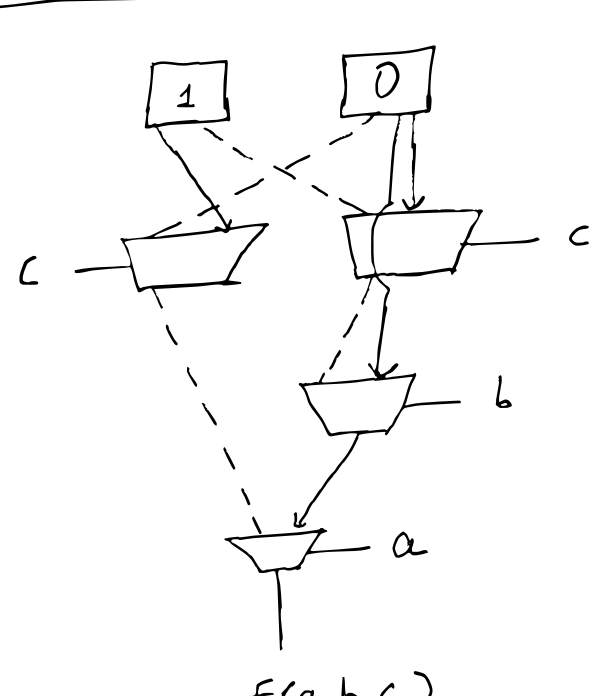


RODBDD

∴ $f(a,b,c) =$



Circuit Synthesis:



Q.3] $f(a,b,c,d) = a\oplus b\oplus c\oplus d$

$= (a\bar{b}+\bar{a}b)\oplus(c\bar{d}+\bar{c}d)$

$= (\bar{a}\bar{b}+ab)(c\bar{d}+\bar{c}d) + (a\bar{b}+\bar{a}b)(\bar{c}\bar{d}+cd)$

$= \bar{a}\bar{b}\bar{c}d + \bar{a}\bar{b}c\bar{d} + a\bar{b}\bar{c}\bar{d} + a\bar{b}cd + ab\bar{c}d + abc\bar{d} + \bar{a}b\bar{c}\bar{d} + \bar{a}bcd$

$= f(1, 3, 5, 7, 9, 11, 13, 15)$

$\overline{f(a,b,c,d)} = f(2, 4, 6, 8, 10, 12, 14, 0)$

$= \bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}c\bar{d} + \bar{a}b\bar{c}\bar{d} + a\bar{b}\bar{c}\bar{d} + \bar{a}bc\bar{d} + a\bar{b}c\bar{d} + ab\bar{c}\bar{d} + abc\bar{d}$

∴ $f(a,b,c,d) = \overline{(\bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}c\bar{d} + \bar{a}b\bar{c}\bar{d} + a\bar{b}\bar{c}\bar{d} + \bar{a}bc\bar{d} + a\bar{b}c\bar{d} + ab\bar{c}\bar{d} + abc\bar{d})}$

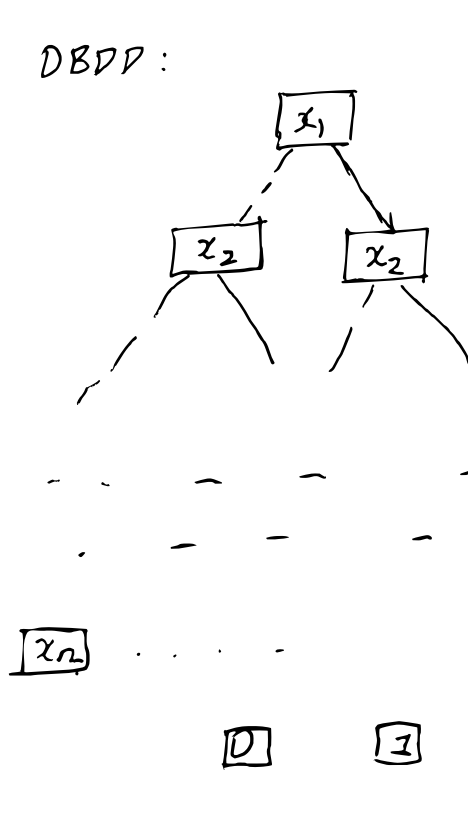
$= (\overline{\bar{a}\bar{b}\bar{c}\bar{d}})(\overline{\bar{a}\bar{b}c\bar{d}})(\overline{\bar{a}b\bar{c}\bar{d}})(\overline{a\bar{b}\bar{c}\bar{d}})(\overline{\bar{a}bc\bar{d}})(\overline{a\bar{b}c\bar{d}})(\overline{ab\bar{c}\bar{d}})(\overline{abc\bar{d}})$

$= (a+b+c+d)(a+b+\bar{c}+d)(a+\bar{b}+c+d)(\bar{a}+b+c+d)(a+\bar{b}+\bar{c}+d)(\bar{a}+b+\bar{c}+d)(\bar{a}+\bar{b}+c+d)(\bar{a}+\bar{b}+\bar{c}+d)$

∴ f is self-dual //

Q.4] for order $x_1 > x_2 > \ldots > x_n$

DBDD:



$x_1 \rightarrow 1$ node

$x_2 \rightarrow 2^{2-1} = 2$ nodes

$x_j \rightarrow 2^{j-1}$ nodes

Q.5] $\frac{\partial}{\partial x}(f\oplus g) = \frac{\partial f}{\partial x} \oplus \frac{\partial g}{\partial x}$

define $\frac{\partial f}{\partial x} = f_{\bar{x}} \oplus f_x$

∴ $(f\oplus g)_{\bar{x}} \oplus (f\oplus g)_x$

$= f_{\bar{x}}\oplus g_{\bar{x}} \oplus f_x \oplus g_x$

$= (f_{\bar{x}}\oplus f_x)\oplus(g_{\bar{x}}\oplus g_x)$

$= \frac{\partial f}{\partial x} \oplus \frac{\partial g}{\partial x} =$ RHS

Hence Proved!!!

EE224 Midterm

2]

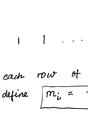
$f = \bar{A}\bar{C}\bar{D} + \bar{A}B\bar{C} + \bar{A}\bar{B}C + \bar{A}C\bar{D} + AB\bar{D} + A\bar{B}C + A\bar{B}\bar{D}$

3] $f'(A,B,C) = A'B' + A'C + BC + AB$

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

or

$A'B' + CB + A'C + AB$



AB', AB

SOP I

4] n variables

a_1	a_2	a_3	...	a_n
0	0	0	...	0
0	0	0	...	1
.				.
.				.
.				.
1	1	1	...	1

For each row of the table,

define $m_i = \prod_{j} \hat{a}_j$

where $\hat{a}_j$ can be a_j or $\overline{a_j}$ depending on the value 1 or 0 respectively.

define $M_i = \sum_{j} \hat{a}_j$

In SOM form,

$f = \sum m_k$

$\overline{m_i}$ = 1's complement of $\hat{a}_j$

$\therefore \overline{m_i} = M_{2^n-1-i}$

If $f = \sum_{k \in S} m_k$

$\bar{f} = \left(\sum_{k \in S} m_k\right)$

$\therefore \bar{f} = \prod_{k \in S} \overline{m_k}$

$= \prod_{k \in S} M_{2^n-1-i}$

$Y = \bar{B}C + BD$

$F = \bar{Y}A$

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

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

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

$= AB\bar{D} + A\bar{B}\bar{C} + A\bar{C}D$

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

By consensus theorem,

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

NOT → $A = z$, $B = D$

Input $C \rightarrow F = \bar{z}$

AND → $ABCD$

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

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

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

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

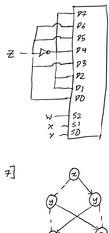
$A + \bar{A}B = AB$

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

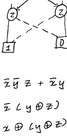
$A(\bar{A} + B) = AB$

XOR $(w, x, y, z) = w \oplus x \oplus y \oplus z$

w	x	y	z	XOR
0	0	0	0	0
0	0	0	1	$1 = z$
0	0	1	0	1
0	0	1	1	$0 = \bar{z}$
0	1	0	0	1
0	1	0	1	$0 = \bar{z}$
0	1	1	0	0
0	1	1	1	$1 = z$
1	0	0	0	1
1	0	0	1	$0 = \bar{z}$
1	0	1	0	0
1	0	1	1	$1 = z$
1	1	0	0	0
1	1	0	1	$1 = z$
1	1	1	0	1
1	1	1	1	$0 = \bar{z}$



Q 7]



$F = \bar{x}\bar{y}z + \bar{x}y\bar{z} + xy\bar{z} + x\bar{y}\bar{z}$

$= \bar{x}(y \oplus z) + x(\bar{y} \oplus z)$

$= x \oplus (y \oplus z)$

Q 1] $5x^2 - 50x + 125 = 0$

$x^2 - 10x + 25 = 0$

$(x - 5)(x - 5) = 0$

$x^2 - (5+5)x + (5 \times 5) = 0$

Let's say they had r fingers

$\therefore \quad 125 = r^2 + 2r + 5$

$50 = 5r$

$\therefore \quad x^2 - rx + \left(\frac{r^2 + 2r + 5}{5}\right) = 0$

$\frac{r^2 + 2r + 5}{5} = 5 \times 8$

$-r = -(5 + 8)$

$r = 13$

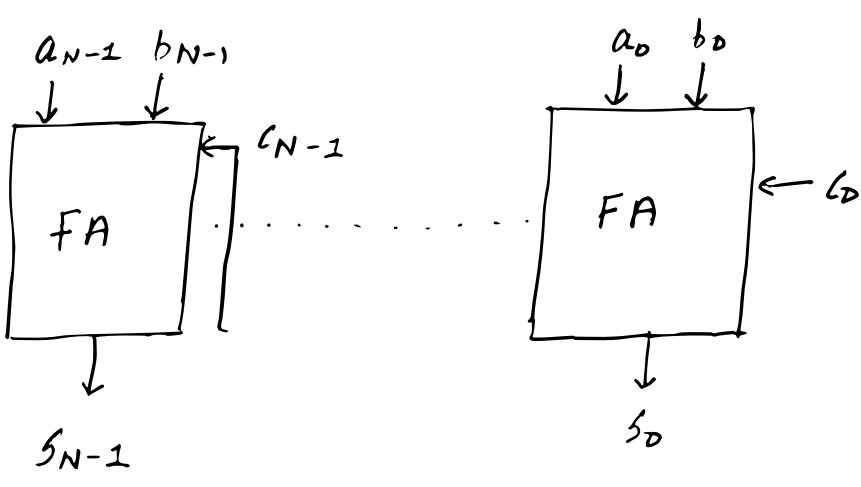
$13^2 \quad 169 + 26 + 5$

$174 + 26 = 200$

ADDERS:

1. RCA $\propto$ 4N
2. CLA $\propto$ 2N
3. LOGARITHM $\propto \log_2 N$ / $\propto \log N$
   - $\log_2 N$ → CMOS
   - $\log N$ → TTL

RCA:



'N' Full Adders needed

If cost of 1 FA = X

Cost of RCA = N · X

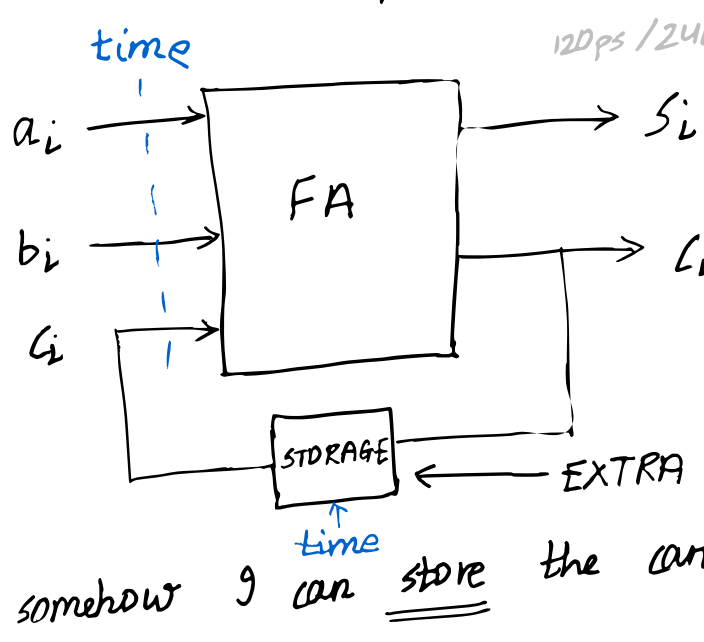
$A = a_{N-1}\ a_{N-2} \ldots a_1\ a_0$

$B = b_{N-1}\ b_{N-2} \ldots b_1\ b_0$

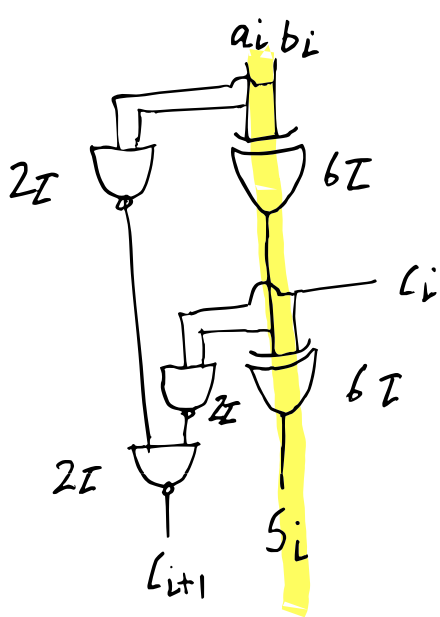
S_0

. . . .

But, can I use the same FA and just change the input sequentially?



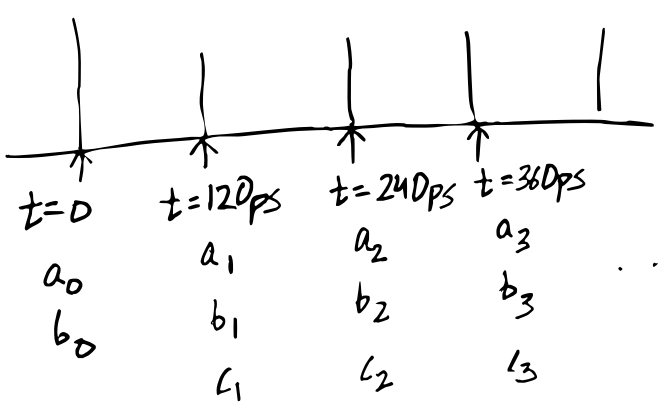
If somehow I can store the carry generated, I can implement it.



Delay = 12I

If I = 10 ps,

Delay = 120 ps



∴ For 'N' bit, delay = 120 × N ps

∴ **delay $\propto$ N**

↳ But hardware substantially decreased.

Also suitable for variable input bits as hardware is common.

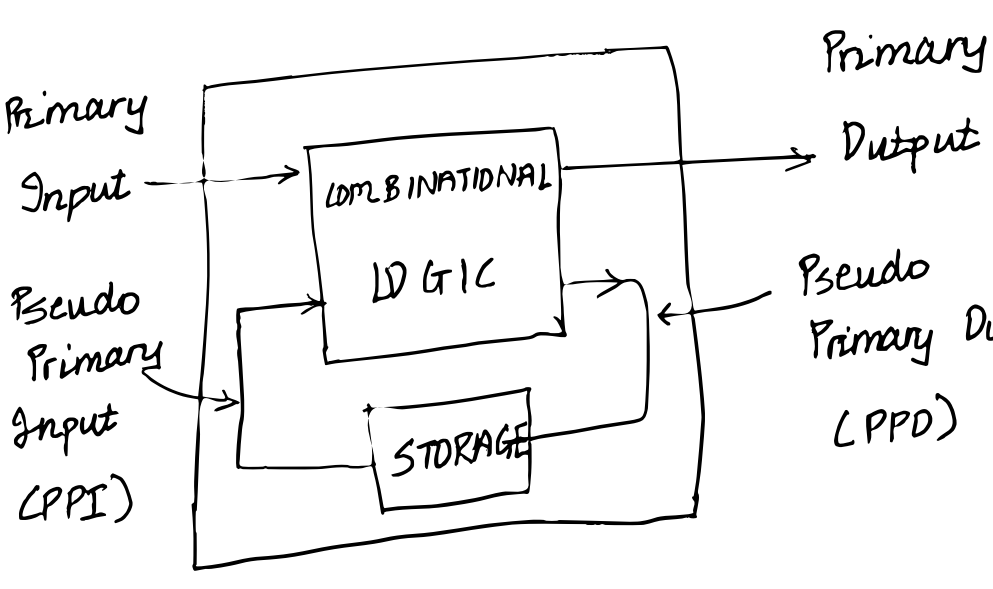
called SERIAL ADDER.

But how will we know when to send next input?

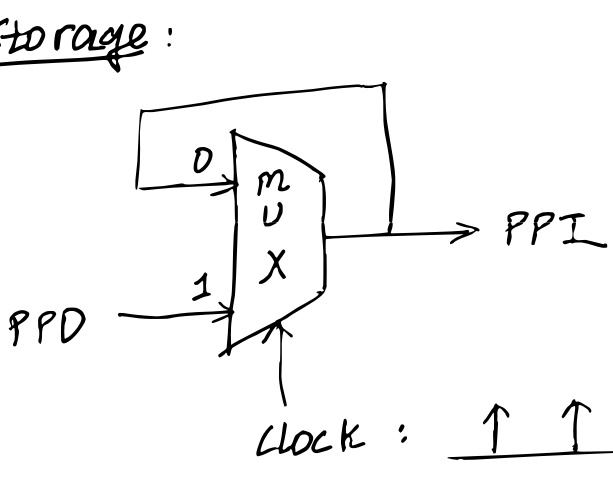
↳ Extra time marker needed.

- ↳ One for sending a_i, b_i, C_i
- ↳ One for storing C_{i+1} in storage.

We usually call this time marker 'clock'.



Storage:

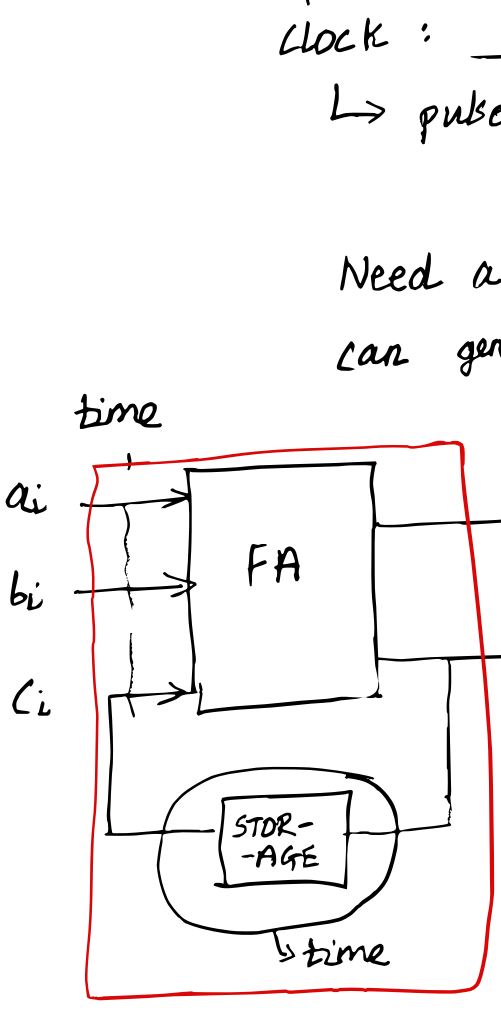


↳ pulse train / impulse train

↑

Need a time marker that can generate such an output.

time



C_i depends on previous inputs

∴ Output depends on current as well as past inputs.

↳ i.e. the circuit exhibits temporal behaviour.

Such circuits are called SEQUENTIAL CIRCUITS.

∴ SEQUENTIAL CIRCUITS:

f (current input, previous inputs)

↓

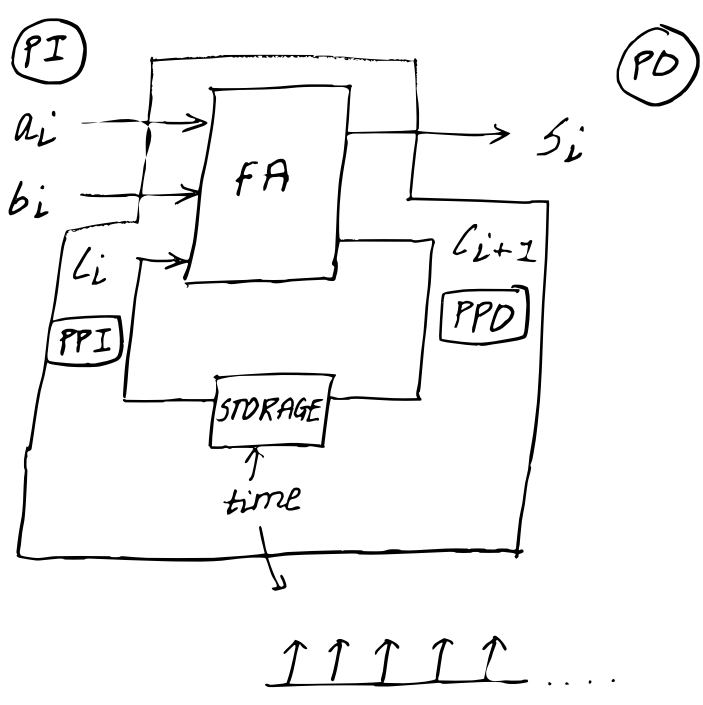
Need combinational logic, storage, time marker (clock)

STATE:

possible values of storage at any time?

Will discuss in next class.

SERIAL ADDER

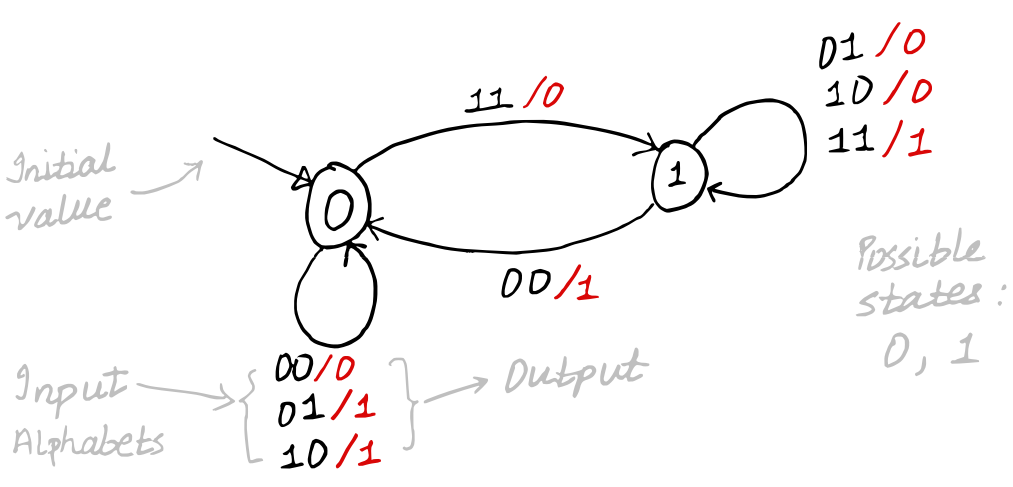


Storage ⟶ store state of the system

combinational logic ⟶ Produce PO & PPO

Clock (periodic pulses)

STATE TRANSITION SYSTEM:



STATE TRANSITION DIAGRAM / GRAPH (STG)

PPD

PS \ NS	00	01	10	11
0	0	0	0	1
1	0	1	1	1

PO — Output (sum)

PS	00	01	10	11
0	0	1	1	0
1	1	0	0	1

∴ State Machine:

$M = (S, S_0, I, O, \delta, \lambda)$

- O — output alphabets
- λ — output function
- S — set of possible states $\{0,1\}$ in above case
- S_0 — initial state
- I — state transition function
- Σ — input alphabets that machine recognizes. $\Sigma = \{00, 10, 01, 11\}$ in above case

$\delta : S \times I \longrightarrow S$ (S: state, I: input)

$\lambda : S \times I \longrightarrow O$ (MEALY machine)

$S \longrightarrow O$ (MOORE machine) ↓ output does not depend on input.

∴ δ :

S \ ab	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$\delta = sb + ab + as = ab + s(a+b)$

↳ carry

S → state not sum

∴ λ :

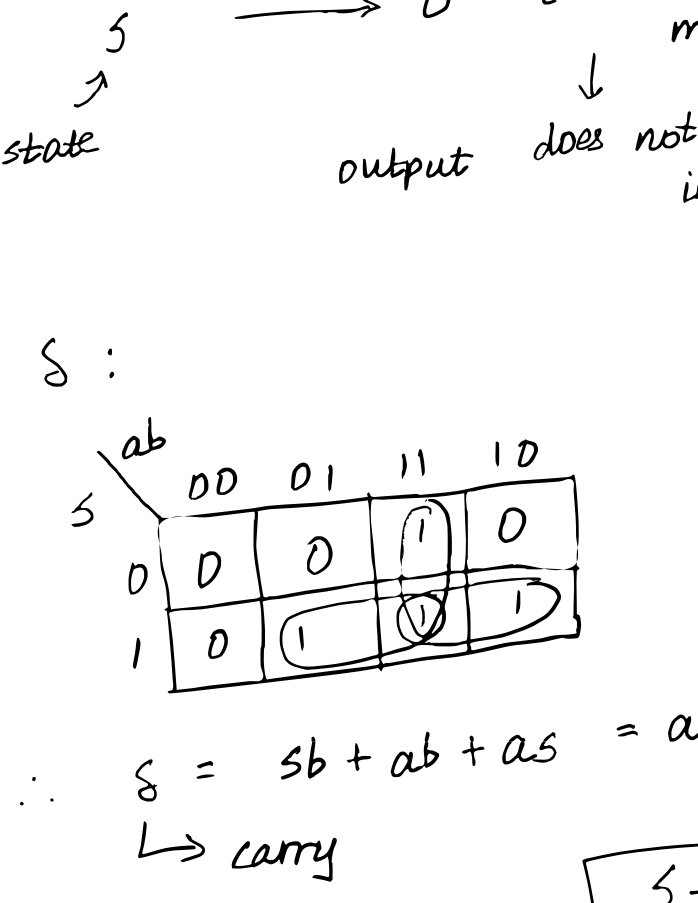
S \ ab	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$\lambda = \bar{s}\bar{a}b + \bar{s}a\bar{b} + s\bar{a}\bar{b} + sab$

$\lambda = a \oplus b \oplus s$

↳ sum

MEALY MACHINE: (Serial adder)



VHDL process:

```
if (clk = 1) then
    q <= d
```

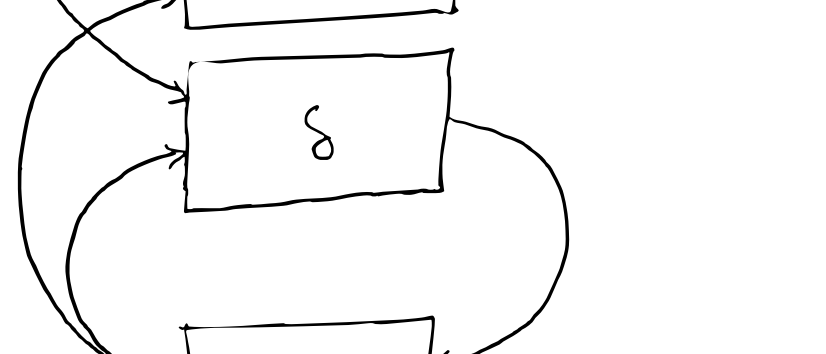
} partially specified behaviour

Here's where VHDL, Verilog or any other HDLs get ambiguous.

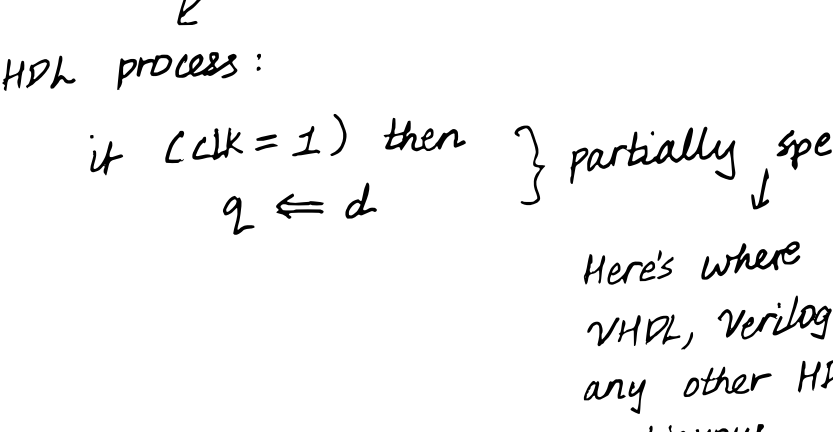
In this case, synthesizer will synthesize to default value; in most cases, default value retains previous value.

combinational logic behaviour needs to be fully specified.

MOORE MACHINE: (Pulse Generator)



Counter: (mod 4 counter)



$S = \{S_0, S_1, S_2, S_3\}$

$M = (S, s_0, I, O, \delta, \lambda)$

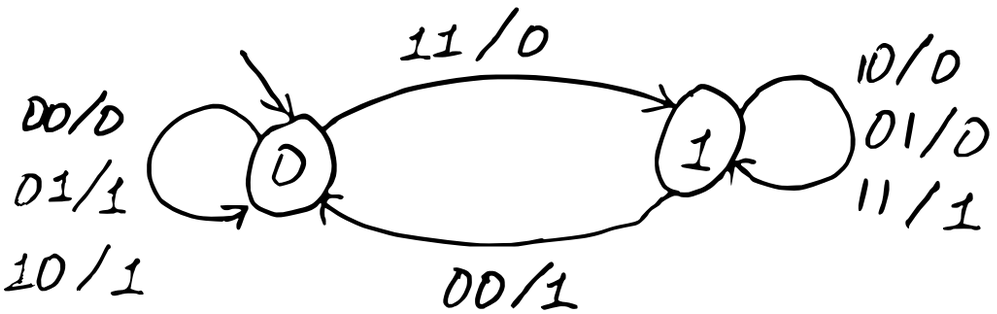
$\delta : S \times I \longrightarrow S$

STATE TRANSITION FUNCTION

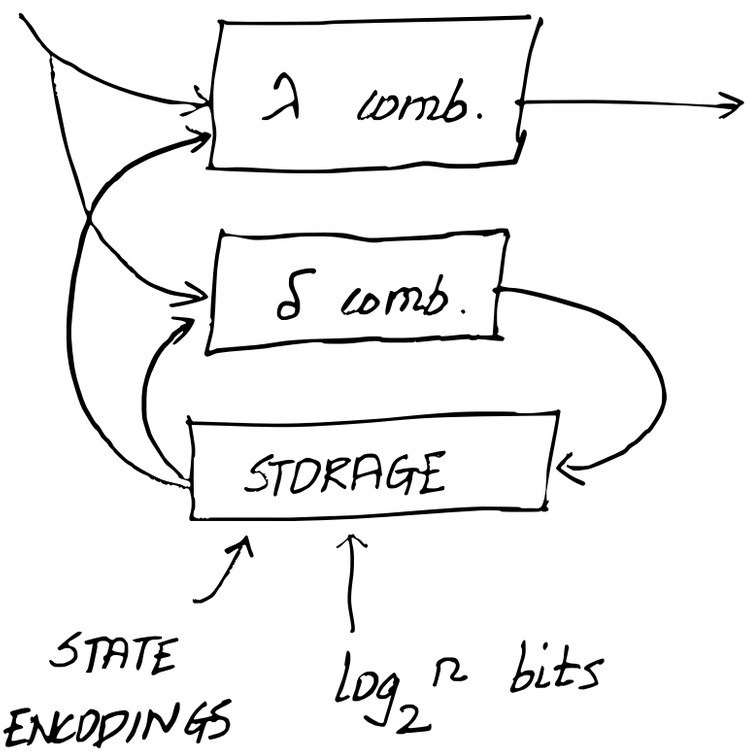
$\lambda : S \times I \longrightarrow O$ (MEALY)

$S \longrightarrow O$ (MOORE)

STATE TRANSITION GRAPH



(Full Adder)



e.g. one-hot encoding

↳ 'n' states → n bits

e.g. S_0 0001

S_1 0010

S_2 0100

S_3 1000

e.g. vending machine → 'greedy'

↳ does not have change

cost → ₹15

$I = \{₹0, ₹5, ₹10\}$

$O = \{0, 1\}$

First, (Intuitional)

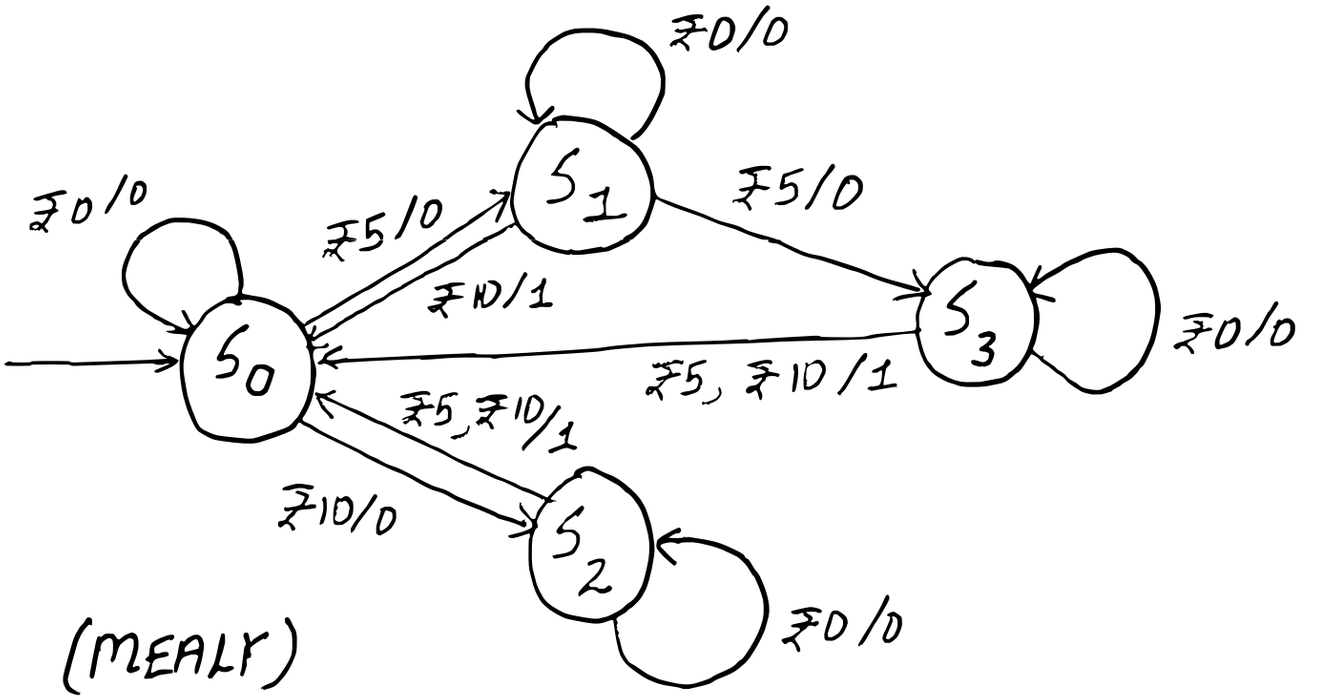
S_0 = ₹0

S_1 = 1 ₹5 coin

S_2 = 1 ₹10 coin

S_3 = 2 ₹5 coins

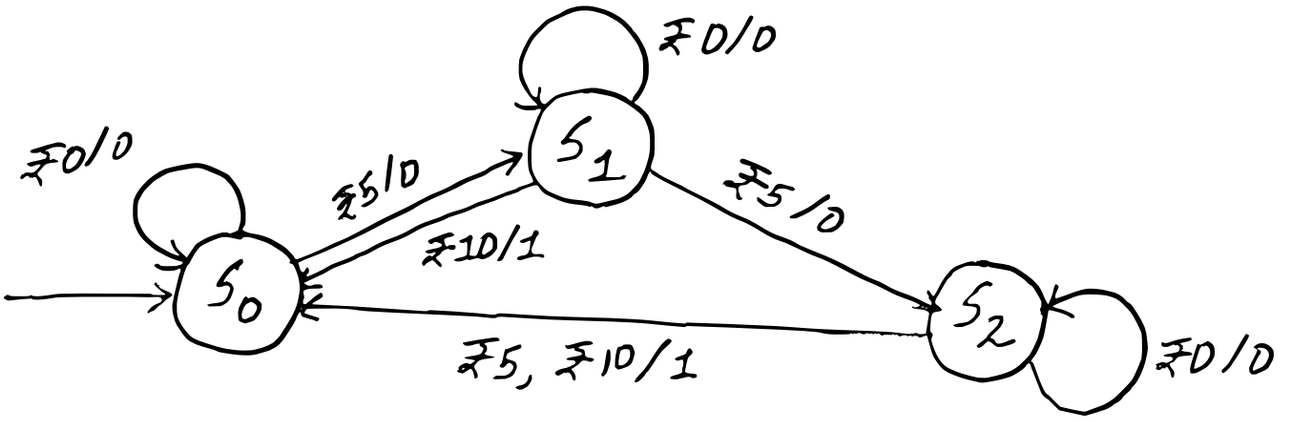
Same from functionality point of view



(MEALY)

Second,
↓
(Reduced)

S_0 = ₹0
S_1 = ₹5
S_2 = ₹10



(MEALY)

MOORE MACHINE:

Say:

S_0 : ₹0 coin

S_1 : 1 ₹5 coin

S_2 : 1 ₹10 coin

S_3 : 2 ₹5 coin

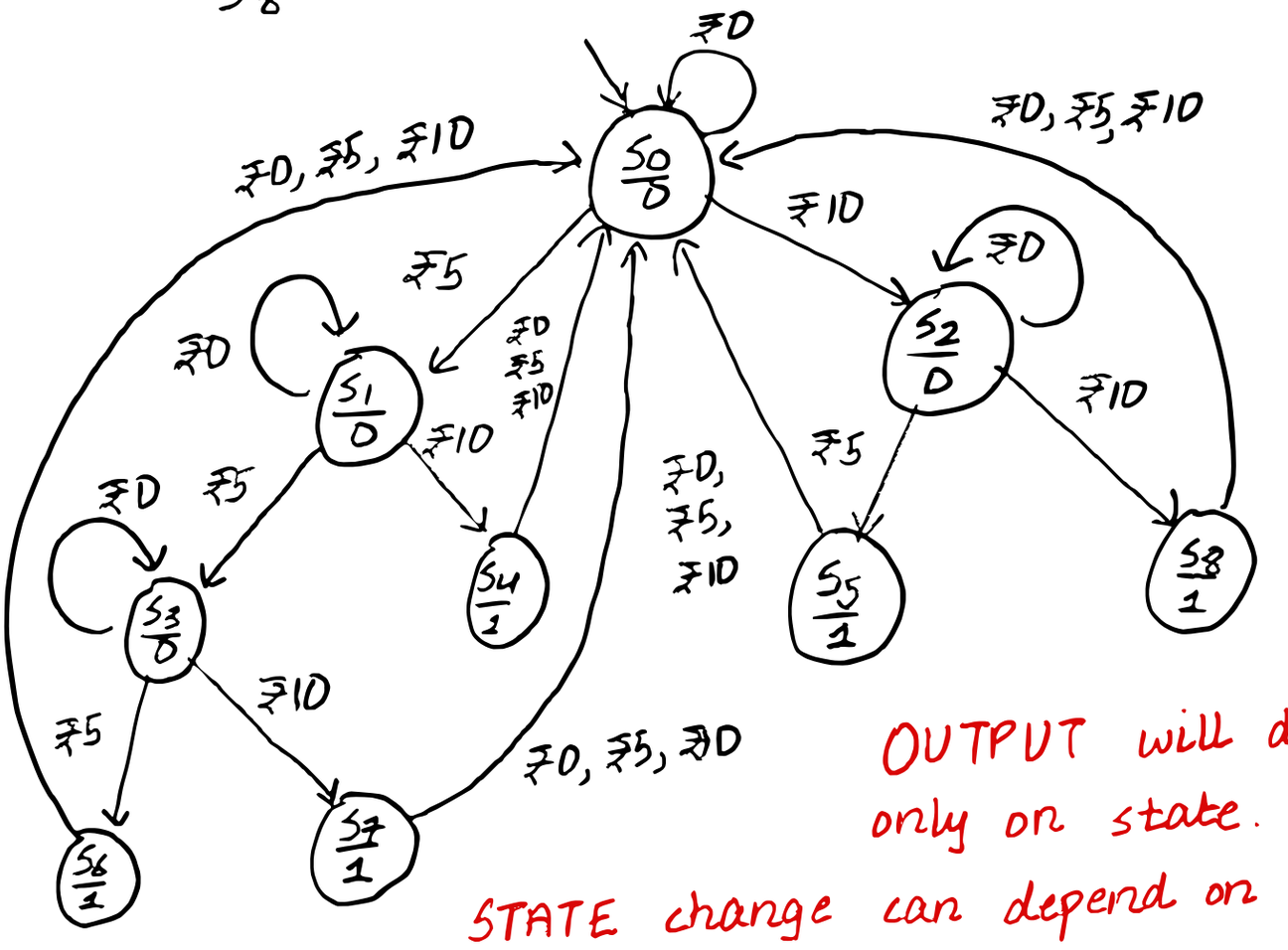
S_4 : 1 ₹5, 1 ₹10 coin

S_5 : 2 ₹10 coin

S_6 : 3 ₹5, coins

S_7 : 2 ₹5, 1 ₹10 coins

S_8 : 1 ₹10, 1 ₹5 coins

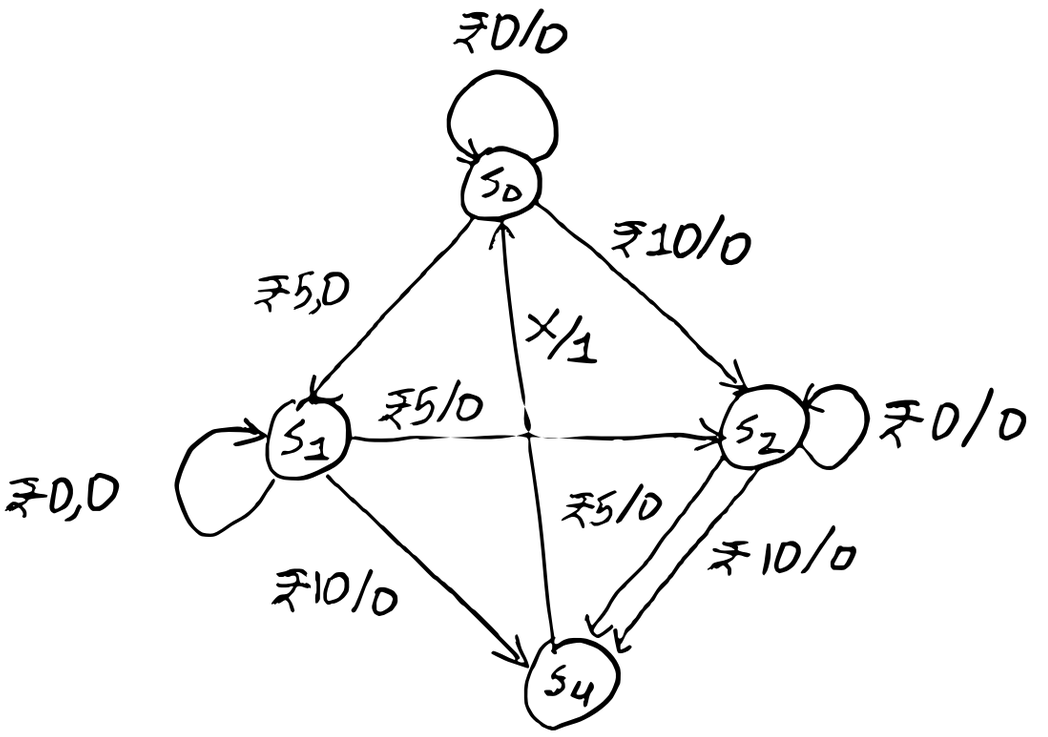


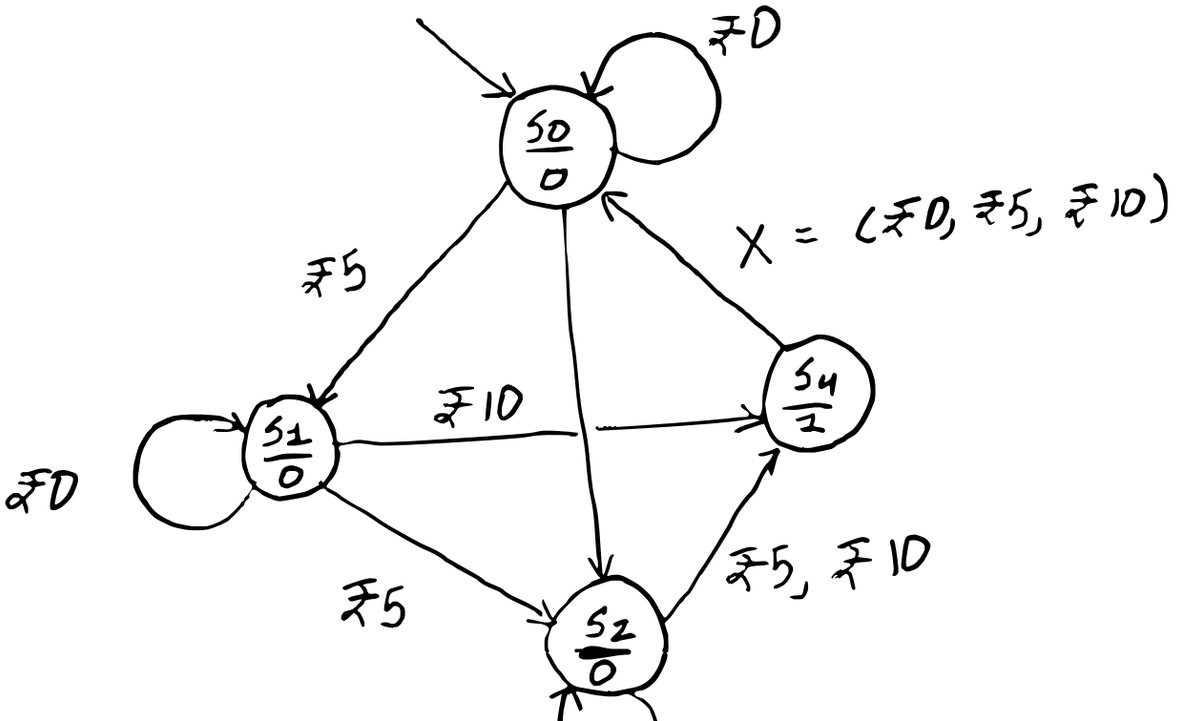
OUTPUT will depend only on state.

STATE change can depend on input.

But for functionality,

$$S_2 = S_3 \quad \& \quad \underbrace{S_4 = S_5 = S_6 = S_7 = S_8}_{\text{isomorphic}}$$



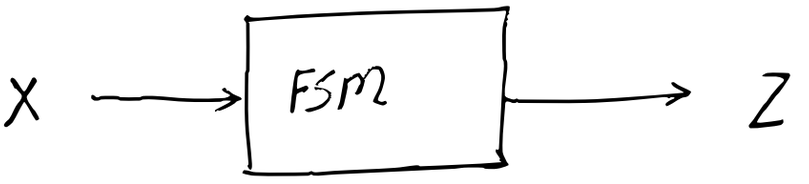


How to minimize the number of states?

<u>Equivalent states</u>: We cannot see states, only OUTPUT

✗ If for same input, map to same state.

✓ If for an infinite long sequence of inputs, you cannot distinguish the output from two state S_i, S_j, then $S_i \equiv S_j$

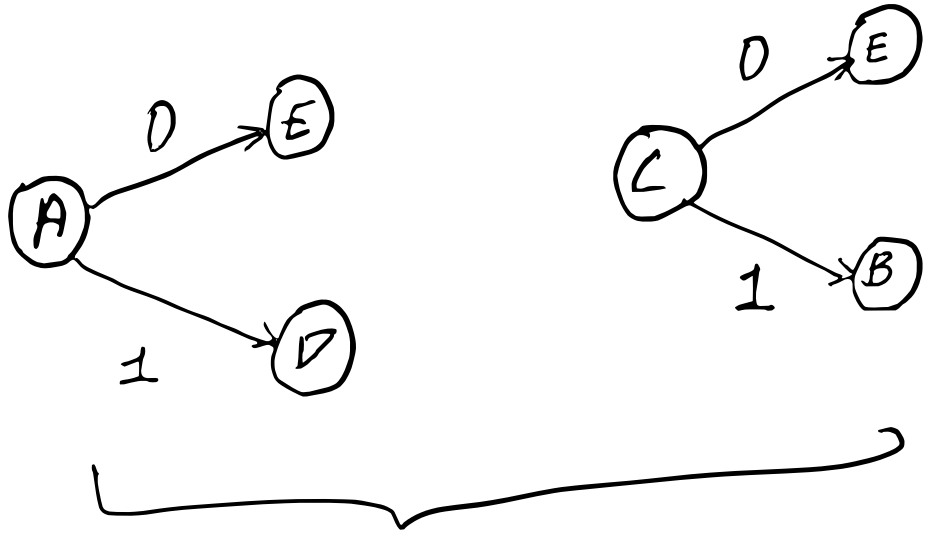


States	NS, Z X = 0	X = 1
A	E, 0	D, 1
B	F, 0	D, 0
C	E, 0	B, 1
D	F, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0

We can only see output Z, <u>not NS</u>

$S_i = A$

$S_j = C$

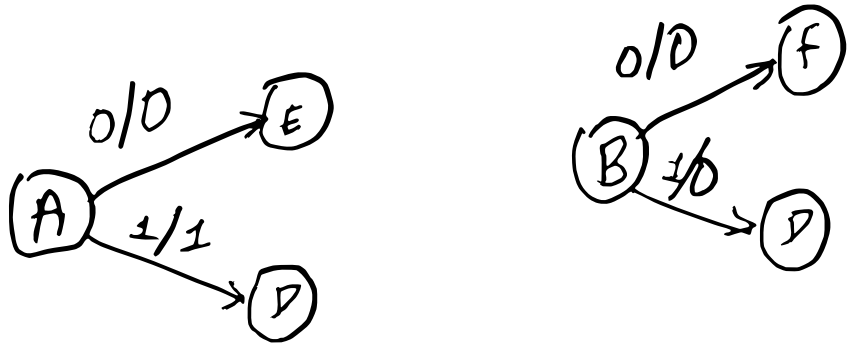


1 - input IN distinguishable states

1 - equivalent states

$S_i = A$

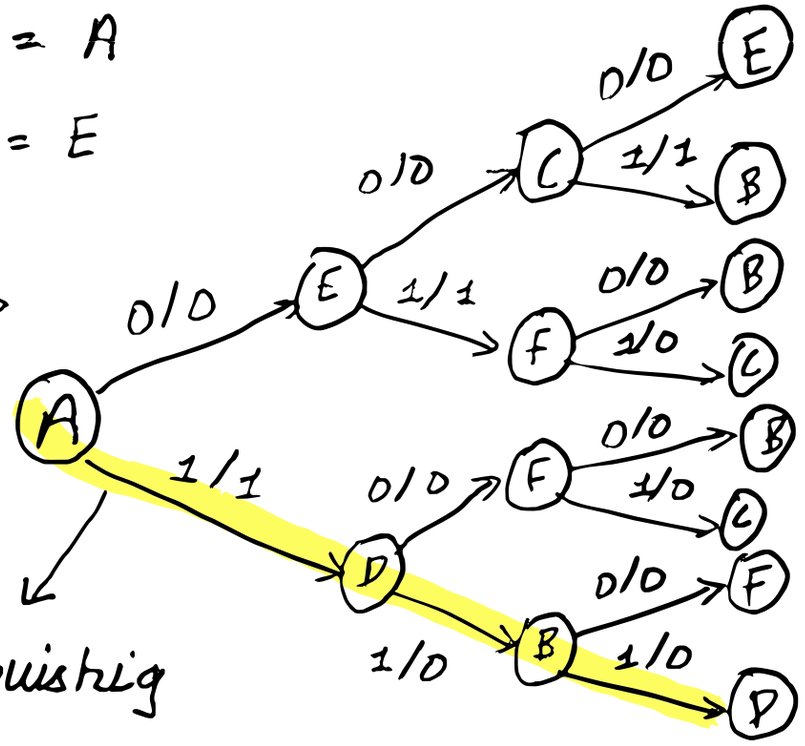
$S_j = B$



1 - input distinguishable states.

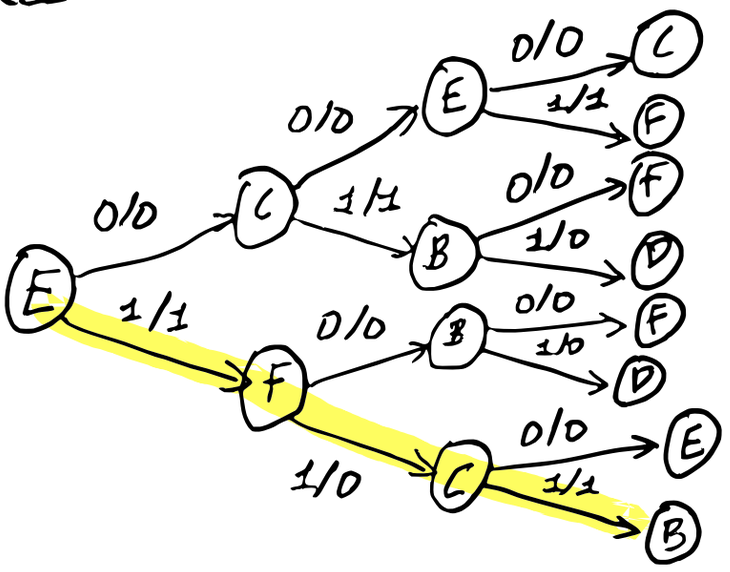
$S_i = A$

$S_j = E$



Distinguishing sequence

000 → 000
001 → 001
010 → 010
011 → 010
100 → 100
101 → 100
110 → 100
$\boxed{111 \rightarrow 100}$



000 → 000
001 → 001
010 → 010
011 → 010
100 → 100
101 → 100
110 → 100
$\boxed{111 \rightarrow 101}$

Distinguishing sequence

1- equivalent

2 - equivalent.

But how do I check for ∞ ?

Theorem: If a set of states are K- equivalent and then when we check for (K+1) equivalence and they map to sets same as K-equivalent sets, then they are ∞- equivalent.

$P_i \longrightarrow$ after 'i' inputs :

$P_0 = (ABCDEF)$

$P_1 = (ACE)(BDF)$

$P_2 = (ACE)(BD)(F)$

$$\boxed{\begin{aligned} P_3 &= (AC)(E)(BD)(F) \\ P_4 &= (AC)(E)(BD)(F) \end{aligned}}$$

$\alpha \quad \beta \quad \gamma \quad \delta$

$\longrightarrow$ Repeating $\downarrow$ ∞-equivalent states found

Hence for 'n' states,
max. n inputs required.

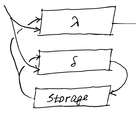
$M = (S, s_0, I, O, \delta, \lambda)$

$\delta : S \times I \longrightarrow S$

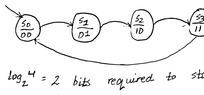
$\lambda : S \times I \longrightarrow O$ (Mealy)

$S \longrightarrow O$ (Moore)

Always, no. of states : Mealy < Moore



Counter :



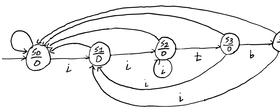
$\log_2 4 = 2$ bits required to store the state

String detector :

Lit b

I = a-z, _, 0-9

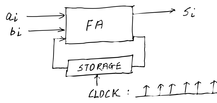
whitespace



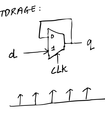
String matching → compilers

using FSM or finite automata

ceil ($\log_2 5$) = 3 bits to store state (3 storage elements)



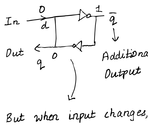
STORAGE :



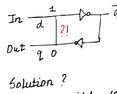
But this is too expensive, needs AND, OR, NOT gates.

Cheaper ?

ONE OF THE CHEAPEST STORAGE ELEMENTS

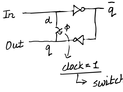


But when input changes, it becomes unstable :



Solution ?

Add a switch controlled by clock.



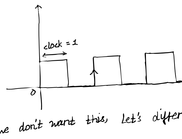
D-type latch

clock = 1 → switch opens

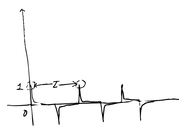
clock = 0 → switch closes

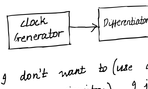
How to generate clock pulses ?

Using multivibrator, we get



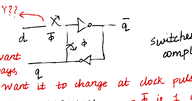
But we don't want this, let's differentiate





But I don't want to (use a differentiator and add more circuitry). I just want to use the generator signal. What do I do ?

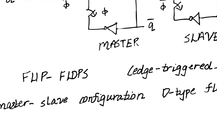
WHY?? We don't want D=Q always



Switches are complements

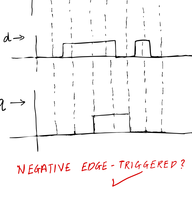
We only want it to change at clock pulse, so USE TWO LATCHES!

Switch is closed ← $\bar{S}$ is 1 only when clock is high.



FLIP FLOPS (edge-triggered)

master-slave configuration D-type flip flop



NEGATIVE EDGE-TRIGGERED?