- Above is the formal procedure for combining 2 graphs (compared to the graph combining we did by hand calculation)

⇒ Reachable states till now =

(00) → (11)

i.e. $\bar{s}_2 \bar{s}_1 \bar{s}_0 + \bar{s}_2 s_1 s_0$ = new state

$\bar{s}_2 \bar{s}_1 \bar{s}_0$: generates the same states

$\bar{s}_2 s_1 s_0$: generates new states

$$T = (\bar{s}_2 \bar{s}_1 \bar{s}_0 + \bar{s}_2 s_1 s_0)\, R(x, s, t)$$

$$= (t_1 \equiv \bar{s}_2 s_1 s_0 \bar{x} + \ldots)(t_2 \equiv \bar{s}_2 s_1 s_0 \bar{x} + \ldots)$$

$$(t_3 \equiv \bar{s}_2 s_1 s_0 x)$$

at $x = 0$: $T = (t_1 = 1)(t_2 = 1)(t_3 = 1)$ ⇒ 11

$x = 1$: $T = (t_1 = 0)(t_2 = 0)(t_3 = 1)$ ⇒ 02

14/10

LOGIC SYNTHESIS

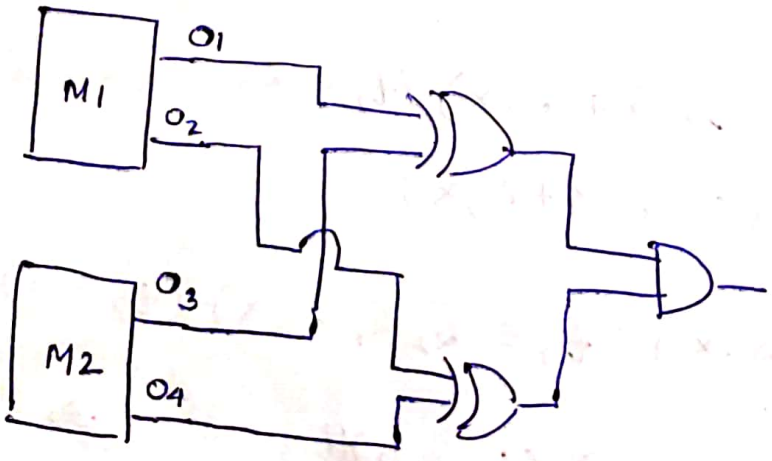
Representation

i) Truth table

ii) Minterms } Cube pdt

iii) Maxterms }

iv) ROBDD



- Need to optimize for
  - area
  - performance
  - power
  - testability

  → using ROBDD → "min" no. of nodes req. to implement logic ⇒ less MUX's ⇒ less area
  (Need to find correct order of variables?)
  use → Sifting (while doing swapping, intermediately the BDD might grow but finally become compact. However, we need enough memory (~500GB) to handle that intermediate increase)
- Verify using
  - SAT
  - BDD
- Test (how to do that?)

- Sifting:
  requires multilevel circuit
  ↓
  maynot be performance optimal

- K-Map: good for human visualisation
  however inefficient for computers
  ↳ textual way better

a	b	c	z
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	
1	0	1	
1	1	0	
1	1	1	

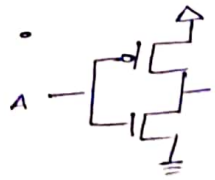
(doesn't affect the output (automatically comes adjacent in K-Map and hence combined) (similar thing happens in BDD where 2 edges going to same output are combined)

→ cube/implicants

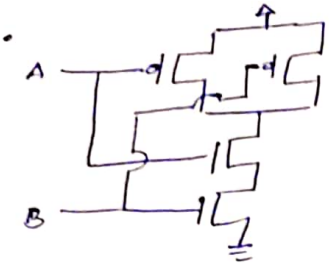
→ variables = literal (uncomplement or uncomplement form)

Ex: $\bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}b\bar{c} + \bar{a}bc = F$

3 i/p AND



1 input = 2 gate
= 2× area



2 input = 4 gates
= 2× area

- 111 cy 3 input NOR requires 6× area

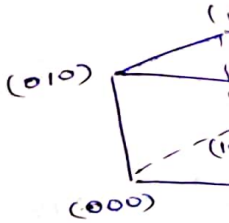
⇒ Area is proportional to number of literal
⇒ to minimize area need to minimize literals

- Performance ⇒ minimise delay of path
⇒ no. of gates in critical path determine delay of ckt
→ delay of gate depends on no. of inputs

- If S is delay of 1 inverter then delay of F(3 stages) = S + 3S + 4S
↳ scaled up by the amount of fanin of gate having max^m literal, in that stage

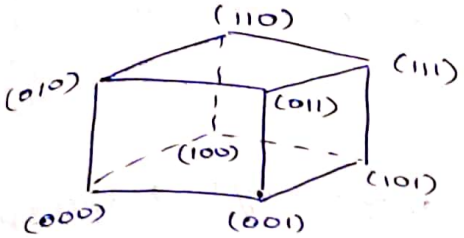
→ Thus, to improve performance and reduce area we need to reduce the number of literals

- Bigger gates



- K-Map
- Quinn-

- Bigger gates dissipate more power



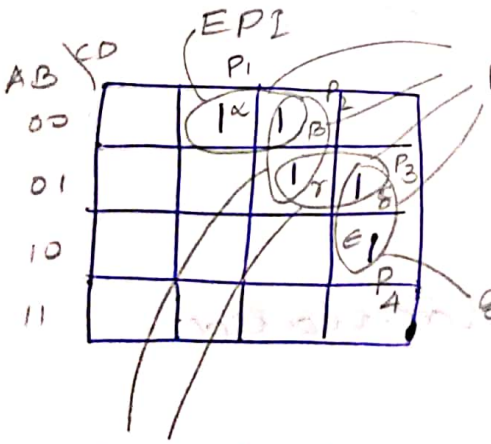
- K-Map is not scalable
- Quinn-McCluskey method is scalable

a	b	c		
0	0	0	00_	0--
0	0	1		
0	1	0	01_	
0	1	1		
1	0	0	10_	1--
1	0	1		
1	1	0	11_	
1	1	1		

→ Tabular Method

The largest size implicant we can get by merging implicants = prime implicant

↳ helps to minimize ckt



Prime Implicants (PI)
- however there are redudant PI's

Essential Prime Implicant (EPI)
(Since it is the only PI that covers $A\bar{B}CD$)

Selective Prime Implicant (SPI)

- From selective prime implicants we need to choose one
  - do so by forming a set covering problem
  - or use logical solution

	P_1	P_2	P_3	P_4	
α	✓				→ covered by 1 ⇒ P_1 is $\in P_1$
β	✓	✓			
γ		✓	✓		
δ			✓	✓	
ϵ				✓	→ covered by 1 ⇒ P_4 is $\in P_1$

- **Patrick Method** (Logical Method)

$\alpha = P_1$ ⇒ each minterm should be covered

$\beta = P_1 + P_2$

$\gamma = P_2 + P_3$

$\delta = P_3 + P_4$

$\epsilon = P_4$

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

→ logical solution that gives PI

→ however one solution of this is $P_1 = P_2 = P_3 = P_4 = 1$

hence convert it into SOP and choose the term which has least number of literals

15/10

- This problem can be translated in matrix problem

$$Ax = b$$

Minterms: $\alpha, \beta, \gamma, \delta, \epsilon$

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array}\begin{bmatrix} P_1 & P_2 & P_3 & P_4 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

PI

A

$b_i \geq 1$

if suppose one solution is [1101]

then $b = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 1 \\ 1 \end{bmatrix}$ → covered twice

Need to find x such that we get minimum no. of ones.

- If we minimize for

  Area = no. of literals

  Perf = no. of literals + product terms

  Use ROBDD to find x

  ↳ gives n-level ckt

  → to get a two level ckt

  (path1) + (path2) + ...

  ↳ a path that evaluates to 1

- Shannon's Theorem

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

(co-factor)

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

Boolean Difference

$$\frac{\partial f}{\partial x} = f_{x_i} \oplus f_{\overline{x_i}}$$

- if they are different this evaluates to 1
  ↳ ⇒ that f is depend on x_i
- if they come out to be same then f is independent of x_i

i) $\frac{\partial f}{\partial x_i} \equiv \frac{\partial f}{\partial \overline{x_i}}$

ii) $\frac{\partial f}{\partial x_i} \equiv \frac{\partial \overline{f}}{\partial x_i}$

iii) $\frac{\partial}{\partial x_i}(f \oplus g) \equiv \frac{\partial f}{\partial x_i} \oplus \frac{\partial g}{\partial x_i}$

iv) $\frac{\partial}{\partial x_i}(f+g) \equiv \overline{f}\frac{\partial g}{\partial x_i} + \overline{g}\frac{\partial f}{\partial x_i} + \frac{\partial f}{\partial x_i}\frac{\partial g}{\partial x_i}$

v) $\frac{\partial}{\partial x_i}(f \cdot g) \equiv f \cdot \frac{\partial g}{\partial x_i} + g\frac{\partial f}{\partial x_i} + \frac{\partial f}{\partial x_i}\frac{\partial g}{\partial x_i}$

- Consensus

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

Smoothing

$= f_{x_i} + f_{\overline{x_i}}$ → minterms independent of x_i

- we can decompose a function w.r.t. orthogonal functions rather than the literals (literals are always orthogonal)

For Φ_i & Φ_j, $\Phi_i \cdot \Phi_j = 0$

$\Rightarrow f = \Phi_i f_{\Phi_i} + \Phi_j f_{\Phi_j}$

∴ for $f = ab + bc + ac$

$\Phi_1 = ab$

$\Phi_2 = \bar{a} + \bar{b}$ } $\Phi_1 \cdot \Phi_2 = 0$

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

$f_{ab} = 1$ — need to make this possible for computer to compute

$f_{(\bar{a}+\bar{b})} = bc + ac + 0$ (when $a=0$, when $b=a$, when both $=0$)

- Positional vector: any function can be expressed using these vectors
- Any variable can be represented in 2 bits:

00 → ϕ
01 → 1
10 → 0
11 → x (don't care)

	a	b	c	
①	01	01	11	→ for ab
②	11	01	01	→ for bc
③	01	11	01	→ for ca
④	10	11	11	→ $\bar{a}$
	00			

bit-wise operation

17/10

LOGIC OPTIMISATION

Exact optimization

i) Enumerate PI → use QM

ii) Solve a set covering problem to find the minimum PI

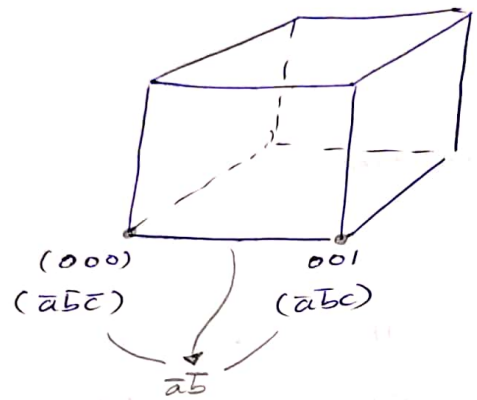
Containment: P_1 shouldn't be covered in P_2

Heuristic way to optimize

→ dont explicitly enumerate

Espresso, Espresso - exact, Espresso - heuristic

i) Expansion
ii) Reduction
iii) Reshape
iv) Redundancy check



$\bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c = \bar{a}\bar{b}(\bar{c}+c) = \bar{a}\bar{b}$: Expansion (= reduction of literals)

expansion of area covered

Expansion: reduce implicants to prime implicants

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

Reduction: size of cube reduces (no. of literals ~~reduce~~ increase)

→ reduction is important if we want to explore other possibility of expansion

↳ 'Reshape'

o - heuristic

while doing reshaping recursively, need to check for redundancy.

, positional representation

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

a	b	c
01	01	11
10	11	01
10	11	11

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

$= ab + \bar{a}$

$f_a = b, \quad f_{\bar{a}} = 1$

- to find $f_{\bar{a}}$ we intersect $\bar{a}$(10 11 11) with each term of f. If it comes out to be ϕ, then that term not present in $f_{\bar{a}}$

⇒ with ab : (00) 01 11

ϕ ⇒ ab not present in $f_{\bar{a}}$

$\bar{a}c$: 10 11 01

⇒ $\bar{a}c$ is a valid term, to get the term to be added we OR the term with $\bar{a}$'s complement

⇒

10	11	01
01	00	00
11	11	01

⇒ c

$\bar{a}$: 10 11 11

⇒

10	11	11
01	00	00
11	11	11

⇒ 1

Thus, $f_{\bar{a}} = c + 1 = 1$ ⟶ Tautology

(f^n evaluates to 1)

f_a: a = 01 11 11

⇒ ab = 01 01 11

→

01	01	11
10	00	00
11	01	11

⇒ b

$\bar{a}c$ = (00) 11 01

ϕ

$\bar{a}$ = (00) 11 11

ϕ

⇒ $f_a = b$

→ Finding partial function w.r.t. a term

bc = 11 01 01

ab = 01 01 01

01	01	01
00	10	10
01	11	11

⇒ a

$\bar{a}c$ = 10 01 01

10	01	01
00	10	10
10	11	11

⇒ $\bar{a}$

$\bar{a}$ = 10 01 01

ology

alculus to 1

10	01	01	
00	10	01	
10	10	10	
	11	11	$\Rightarrow \bar{a}$

$\Rightarrow f_{bc} = 1$ $\Rightarrow$ bc is contained in f.
$\Rightarrow$ containment

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

(due to consensus theorem)

$\Rightarrow$ Redundancy check

Tautology check tells us if there are redundancies present
$\Rightarrow$ Redundancy check

pg 21/10

Heuristic Based Minimization
i) expand
ii) reduce
iii) reshape
iv) irredundancy check

Testability

i) Single-stuck-at fault model
→ only fault occurs at a time
→ linear $\Rightarrow$ no. of faults $\propto$ # gates # nets

Test generation:
find a vector which can give distinguishable faulty or fault-free behaviour

- $f(x_1, x_2, \ldots, x_n)$

TV for a fault s.a.0. at x_i

i) algebraic method

ii) algorithmic

$f_\alpha(x_1, x_2, \ldots, x_n)$: f having fault

$\Rightarrow$ f & f_α should be distinguishable

i.e. $f \oplus f_\alpha = 1$

if α = s.a.0

$\Rightarrow f_\alpha = f_{\bar{x_i}}$

Now we want

$$f \oplus f_{\bar{x_i}} = 1$$

$$x_i f_{x_i} \oplus \bar{x_i} f_{\bar{x_i}} \oplus f_{\bar{x_i}} = 1$$

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

$$x_i f_{x_i} \oplus f_{\bar{x_i}} x_i = 1$$

$$x_i [f_{x_i} \oplus f_{\bar{x_i}}] = 1$$

$\Rightarrow$ $x_i = 1$ $\longrightarrow$ fault sensitizat^n / fault excitation

(have opposite value as the fault value)

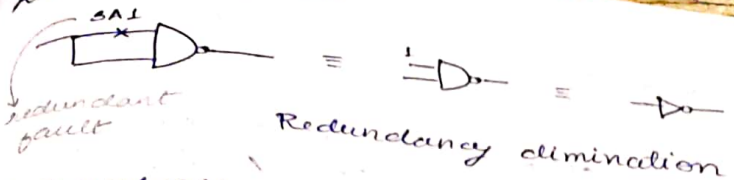
$$f_{x_i} \oplus f_{\bar{x_i}} = 1$$

$$\frac{\partial f}{\partial x_i} = 1$$

$\longrightarrow$ fault Propagation

(whether the given input progates to the P.O. and helps us know if fault indeed occured)

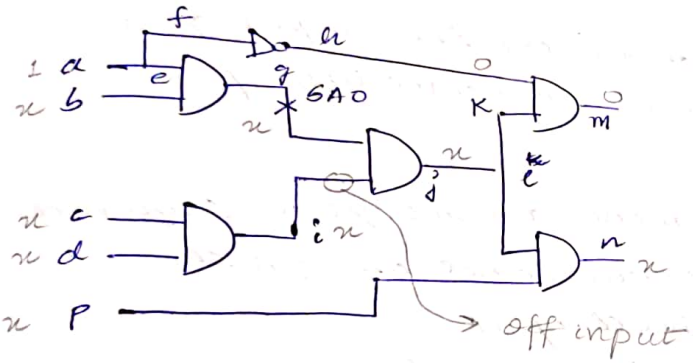
Redundant fault



i) Regularity

ex: for a RCA : need 1 HA & 3 FA
but we use 4 FA for regularity

ii) Timing optimization



i) fault excitation [objective : $g = 1$]

ii) fault propagation

→ to excite the fault, we need to ~~make~~ give input at P.I.

⇒ traverse a path from fault to PI

→ assign a value to PI ;

(Objective value) ⊕ (inversion parity of path)

→ Next do the forward simulation

→ check if objective satisfied

→ If not, go back to the PI's ; take another path

[make b=1 ⇒ objective satisfied]

→ check if value propagated to primary output
- if yes, terminate
- else, choose one of the gates from the fanout and propagate (⇒ objective now changes to the other input of the gate)
  ⇒ OFF INPUTS
    ↳ give non controlling value
  → then repeat the above steps

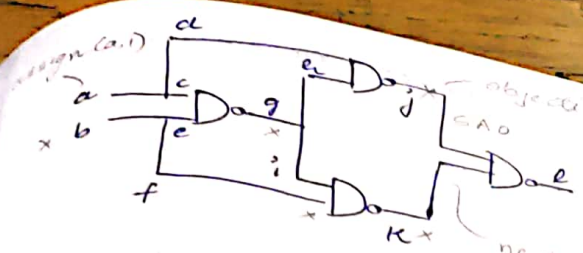
→ finally, choose the the output gate which as a non-controlling input

- If assignment doesn't give required result, go back and assign the complemented input
  so maintain a stack

0	e=1	e=0	e=x
0	d=1		
0	c=1	c=0	
0	b=1		
0	a=1		

24/10 **Test Generation**

i) PODEM (Path oriented decision making)



Logic Simulation

i/p: (0, 1, x)

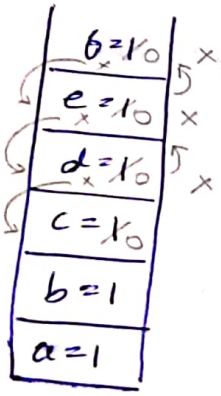
→ satisfies (propagate to PO)

→ doesn't satisfy (go back and assign another i/p)

→ conflict (assignment gives wrong objective & value, hence go back and flip value)

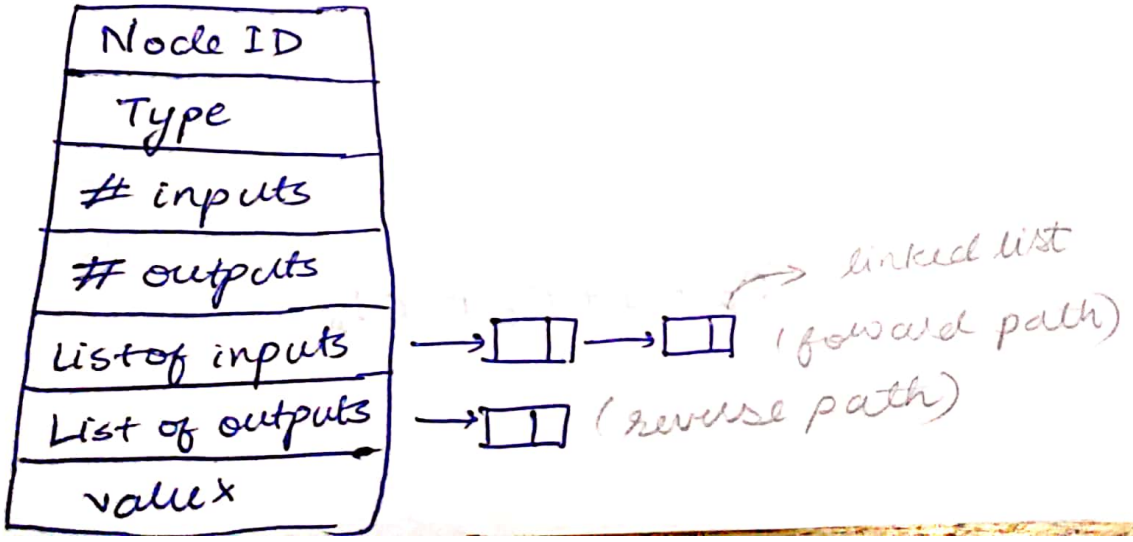
→ if objective not satisfied then go back and make it x and then flip previous assignment)

so we need to maintain a stack

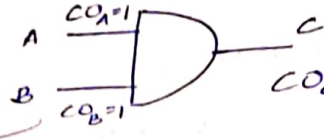


Assignment stack

- Graph



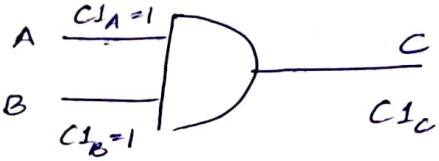
- Take the path which is easier to control or observe
- controlability of a node



$CO_C = 1 + \min(CO_A, CO_B)$

either of A or B being 0 helps

controlability of node B to assign it 0



$C1_C = 1 + C1_A + C1_B$

both path need to be traversed

We want to take the easier path.

- observability of a node
  - need to go backwards here
  - For the AND example

$$O_C = 0$$

$$O_A = 1 + O_C + C1_B$$

controllability of non-controlling value (I can see output when B = 1)

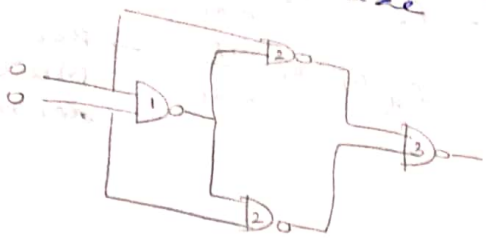
$$O_B = 1 + O_C + C1_A$$

level of a gate = 1 + max (level of inputs)

Evaluate all the gates at one level

→ compile-code simulation

- easy to parallelize

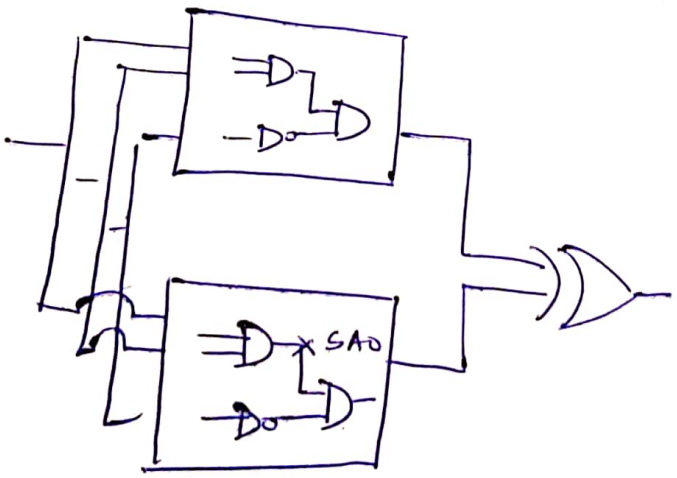


Event-driven simulation

- 10-15 % gates switch when we change input
- not great for parallelization

Eg SAT for test simulation

- fault shud be excited
- faulty & fault-free behaviour shud be distinguished



generate inputs such that output of XOR is 1

NOTE: when we traverse a ckt back and forth, we know which variable affects output more and hence helps to determine order of variable

- **Synthesis**

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

$$= f_{x_i} \oplus \overline{x_i} \frac{\partial f}{\partial x_i}$$

$$= f_{\overline{x_i}} \oplus x_i \frac{\partial f}{\partial x_i}$$

→ Read-Muller decomposition

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

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

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

$= f_{\overline{x_i}} \oplus x_i \frac{\partial f}{\partial x_i}$

- Irredundant terms = no overlap

→ when we have non-overlapping, OR & XOR are same