Logic Synthesis

Page No.
Date

* RTL to gate level netlist ?

How to represent ?

(i) Truth table — Not practical ← f_{ON} / f_{OFF}

Reduce to min terms (f^{ON}), max terms (f^{OFF})

This is called textual way of repr

(ii) Graphical

BDD – Binary Decision ~~Diagram~~ Diagram

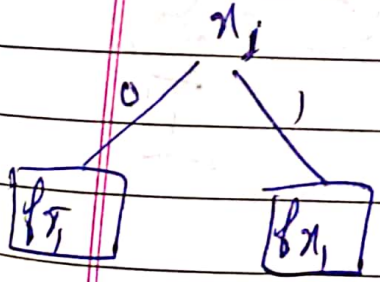
⇒ Sharon's expression

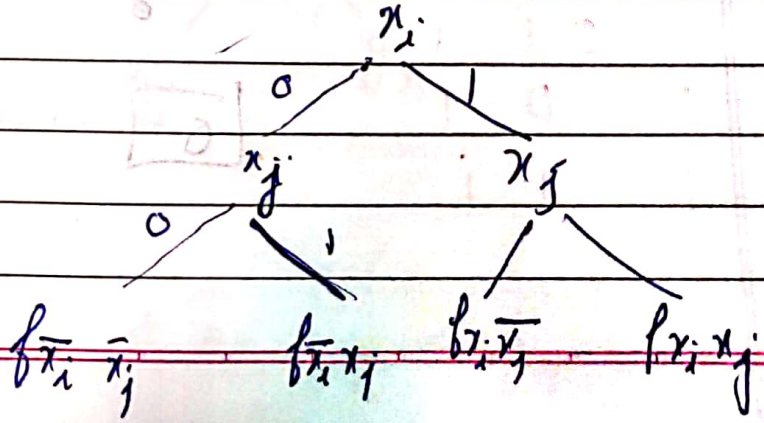
$$f(x_1, \ldots x_n) = x_i \cdot f(x_1, \ldots x_i = 1, \ldots x_n) + \overline{x_i} \cdot f(x_1, \ldots x_i = 0, \ldots x_n)$$

$= x_i$ cofactor + $\overline{x_i}$ co-kernel

f_{x_i} $f_{\overline{x_i}}$

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

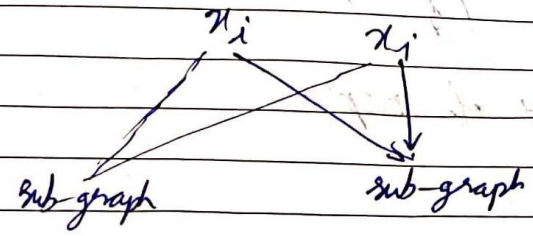




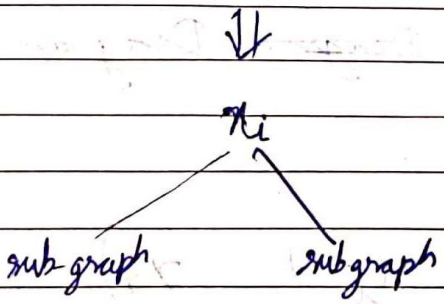
No of elements in the tree = $1+2+\cdots 2^n$
$= 2^{n+1} - 1$

This requires a lot of space for storage but lot of redundancies are present.

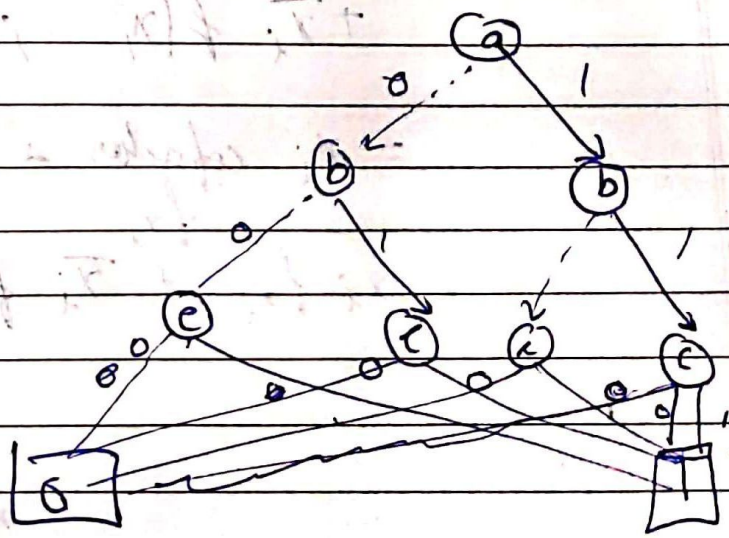
One way is to store only 1's and 0's in last layer and these are only elements which func evaluates to

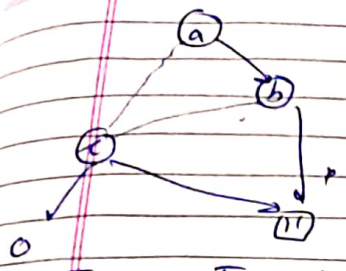


x_i, x_j are isomorphic nodes

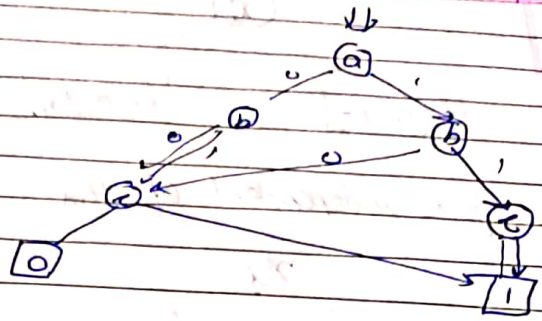


a	b	c	z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1





compact



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

$= ab + c$

Reduced order BDD — Unique form (for a given order of variables)

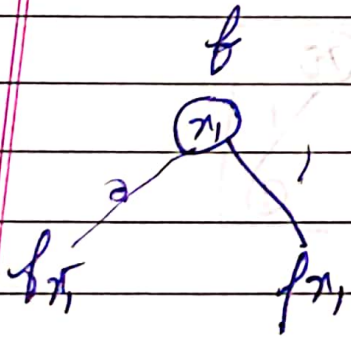
Order BDD is not unique

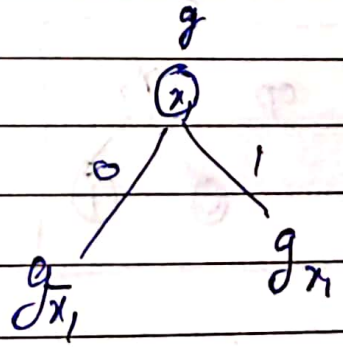
ROBDD — depend on orders of variables

Finding best order is NP-hard.

⟶ Practically one has to start with BDD - $O(2^{n+1})$ space. So effectively, this is useless

If we could find a way to outgrow a graph into ROBDD, it can be helpful.

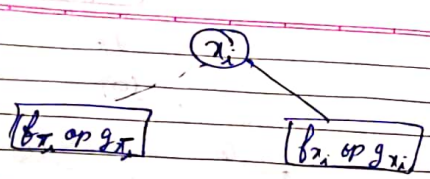




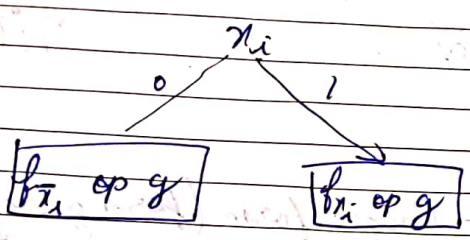
$h = f \text{ op } g$

$= \bar{x}_i \left(f_{\bar{x}_i} \text{ op } g_{\bar{x}_i} \right) + x_i \left(f_{x_i} \text{ op } g_{x_i} \right)$

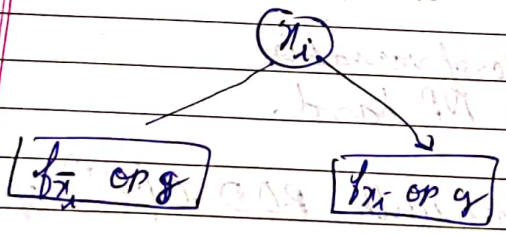
$(x_i \bar{f}_{\bar{x}_i} + \bar{x}_i f_{\bar{x}_i})$ op $(\bar{x}_i g_{x_i})$



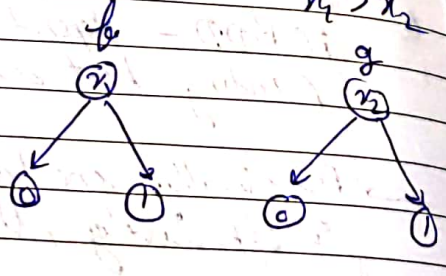
If g is independent of x_i

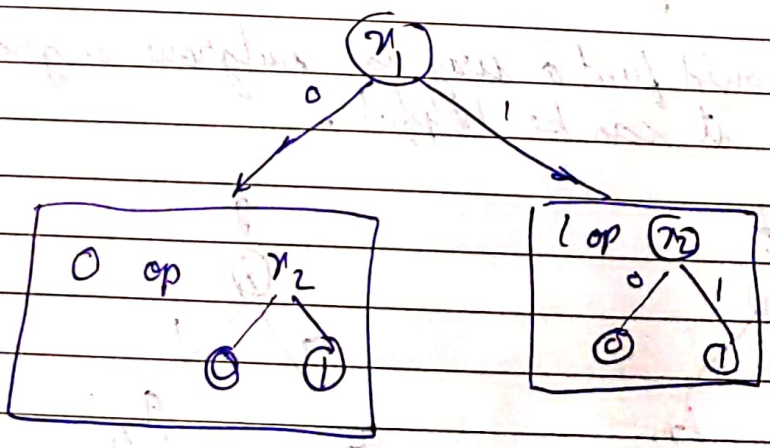


If g is ind^n of x_i



Take priority order $x_1 > x_2$

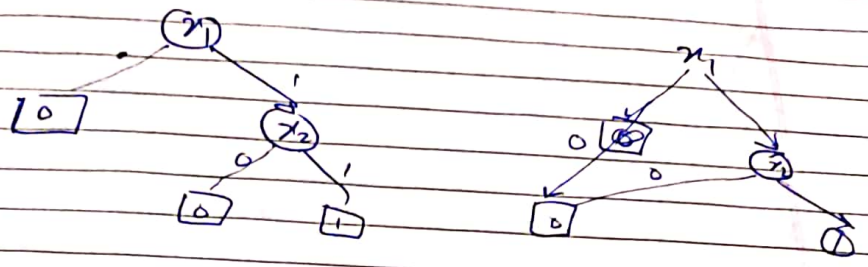




eg

eg

eg – $x_1 . x_2$



eg – $\overline{x_1 x_2}$ ⇒ interchange 1's and 0's

– $\bar{x}_1 \bar{x}_2$ ⇒ interchange solid and dash lines

$x_1 \oplus x_2$

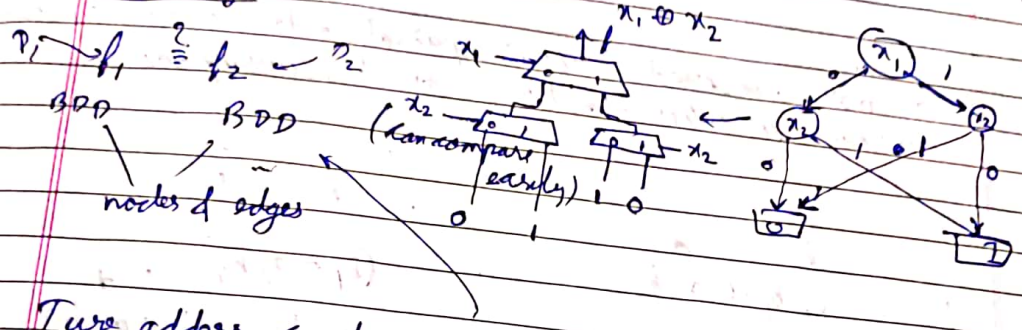
$x_1 \odot x_2 = x_1 x_2 + \bar{x}_1 \bar{x}_2$

Start with one node, then every add will add two nodes.

1/10/19

BDD

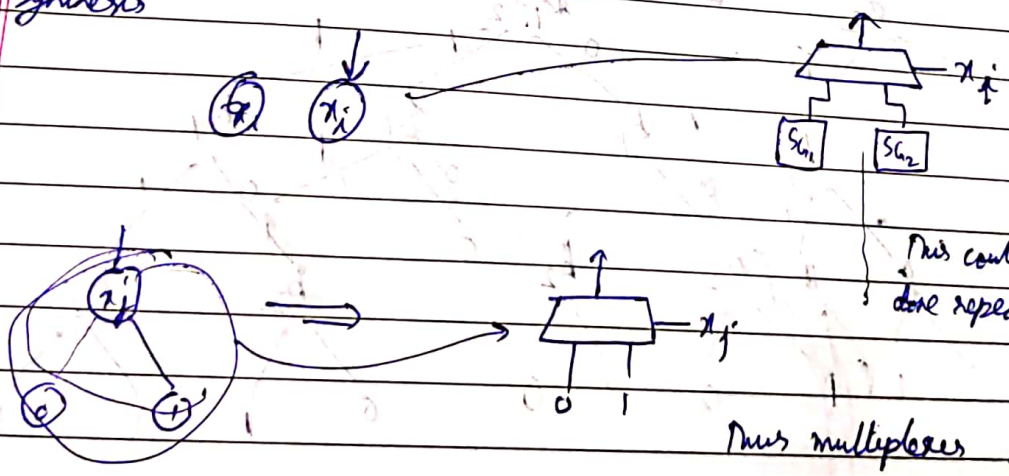
Compact Representation
Grows linearly
Complexity



Two adders can be compared ther in this way (same ripple carry adder with other complicated adders)

Most representations done using BDD's.

Synthesis



Thus multiplexer is universal gate.

Size: No of nodes in BDD. ∈ order of variable ROBDD

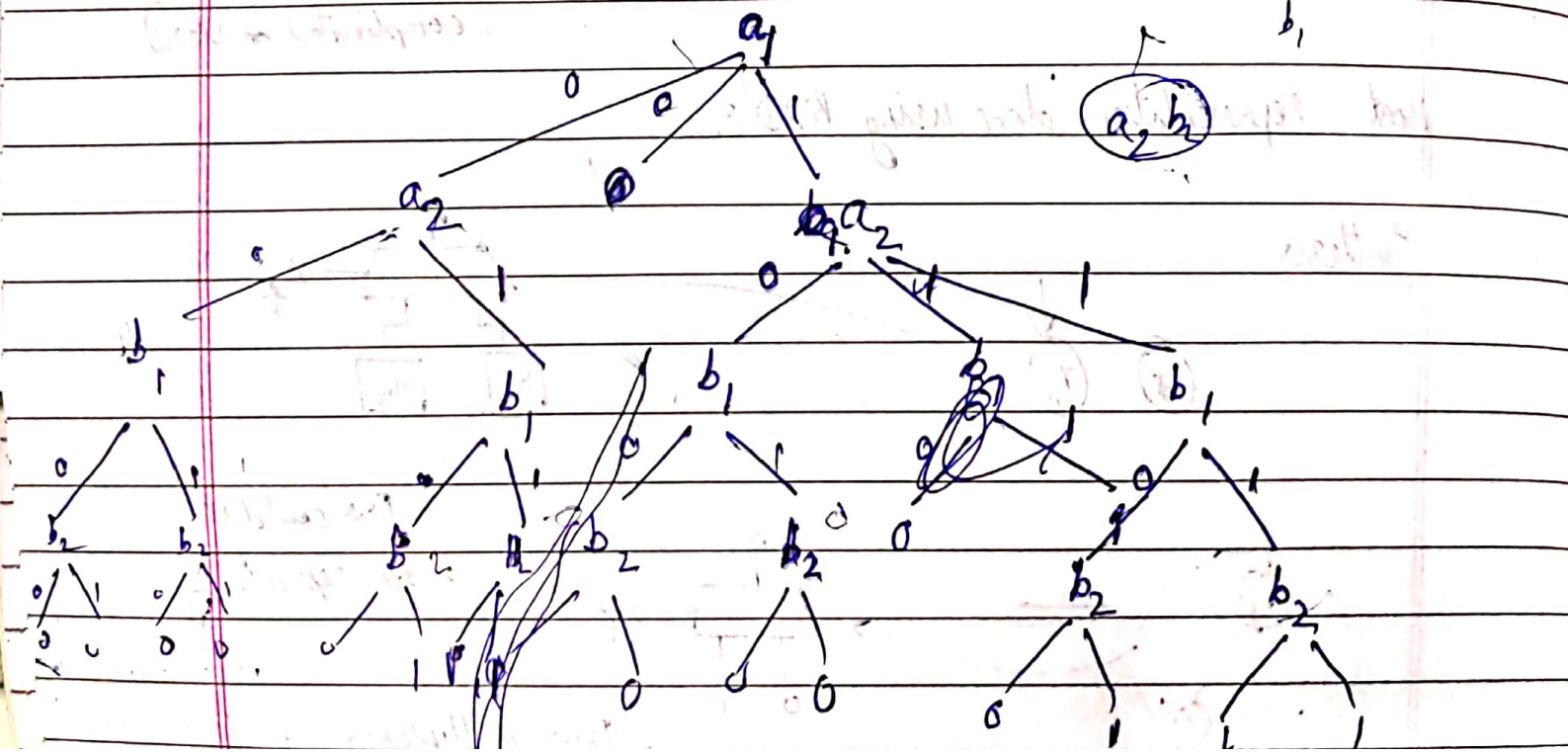
Performance: $\#x_m \times$ #Variables → height of tree

Optimal order of variables ~~which can~~ / Heuristic solution
↓
Greedy kind of algo

We are more concerned about optimal order during synthesis rather than correctness checking. due to recurring cost

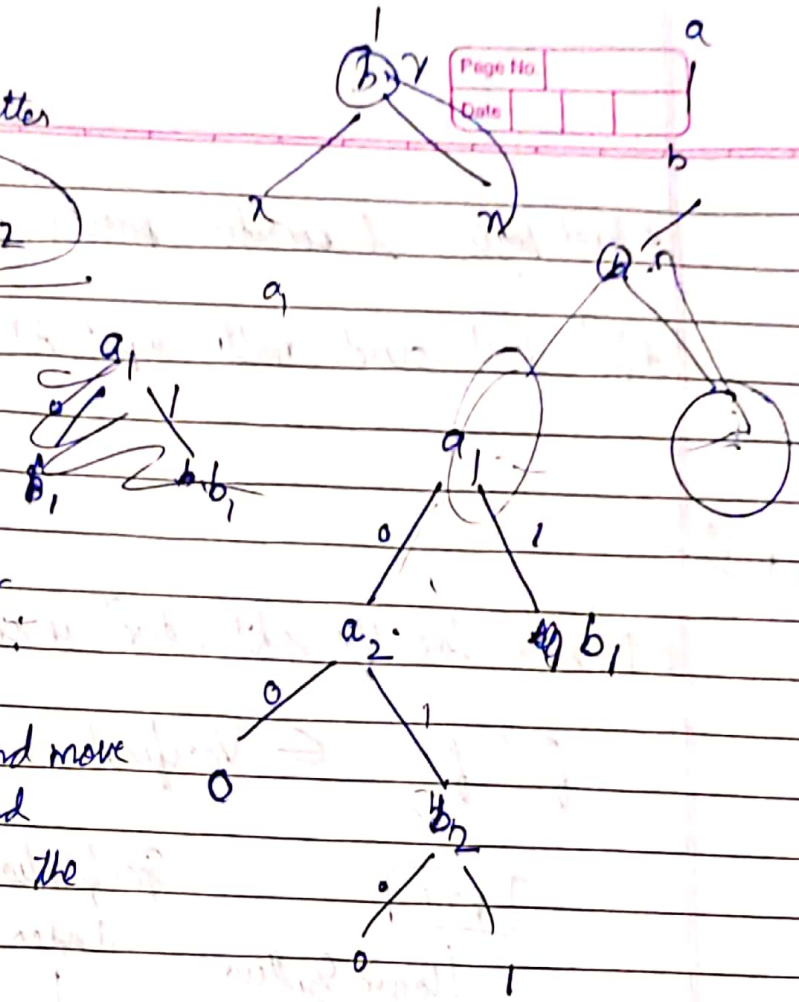
$f = \bar{a}_1 b_1 + a_2 b_2$

$a_1 > a_2 > b_1 > b_2$



This graph is much better

$a_1 > b_1 > a_2 > b_2$



Heuristic (greedy) alg^{n}:

Shifting:
Take one variable and move it to lower level and check if it reduces the order of graph.

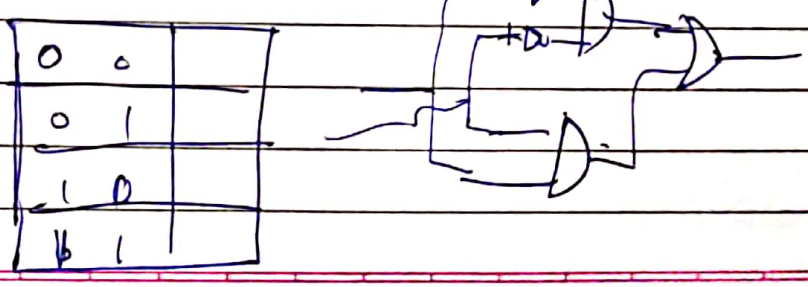
If it reduces, then accept the change and try for other variables greedy alg^{n} as a shifting done once is ~~never~~ never changed.

Swapping: (check)
Here we interchange only among adjacent nodes

~~For ste~~

The BDD described above uses Multi-level-logic.

To:
2 level-logic



3 level logic if consider positive and negative literals at diff^r level and multi-input AND gates available

3 Oct 19

BDD's — store the ckt. desc^n in compact format

$f_1 \overset{?}{=} f_2$ ⇐ Verfical

RTL → Logic Synthesis → Gate level netlist → Transistor level netlist

Specification → order → Implementation

logical exp^n → ROBDD → Replace nodes by mux → Gate level netlist

Truth table / set of min terms → KMap → ckt

7 Oct-19

SAT

$a + b + c$

$a = 0$

$(\quad)(\quad)(\quad) \stackrel{?}{=} 1$

clause — C1 C2 C3 — X — Satisfiable if ✓

$C1 = 1$

$C2 =$

DPLL alg:

assign: a value to a variable in some order (alphabetic order)

fails - back track

Pure literal rule:

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

Clauses which ~~do~~ have just one literal present ~~[illegible]~~ can be directly assigned a value.

eg – d after assigning $a = 0$, we could put $c = 1$

Concensus theorem :

~~$(a+b)(\bar{a}+b)$~~

$(a+b)(\bar{a}+c) \equiv (a+b)(\bar{a}+c)(b+c)$ → These clauses have to be satisfied if LHS is satisfied

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

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

keep on doing it (applying concensus) – similar to that in CS228(as

~~Sto~~

Staldmeric method :

$(\bar{a}+b)$ $(\bar{a}+c)$ $(\bar{b}+d)$ $(\bar{c}+d)$ $= 1$

C1 C2 C3 C4

literal based alg

$a=1 \rightarrow c=1 \rightarrow d=1$ (unit clause rule)

$a=0 \rightarrow b=1 \rightarrow d=1$ (unit clause)

Essential assignment ↓ $d=1$

We should try to keep essential variables at root of tree

Recursive learning (RL) – Does it clause by clause first clause satisfied by which literals.

$a=1 \rightarrow c=1 \rightarrow d=1$

$b=1 \rightarrow d=1 \rightarrow$

$\Rightarrow d=1$ is essential

Random assignment:

$a=1 \quad b=1 \quad c=1 \quad d=0$

$(a+b)\ (\bar{a}+c)\ \underbrace{(\bar{b}+d)}_{\text{UNSAT}}\ \underbrace{(\bar{c}+d)}_{\text{UNSAT}}$

Put one by one of flip which appear in UNSAT clauses

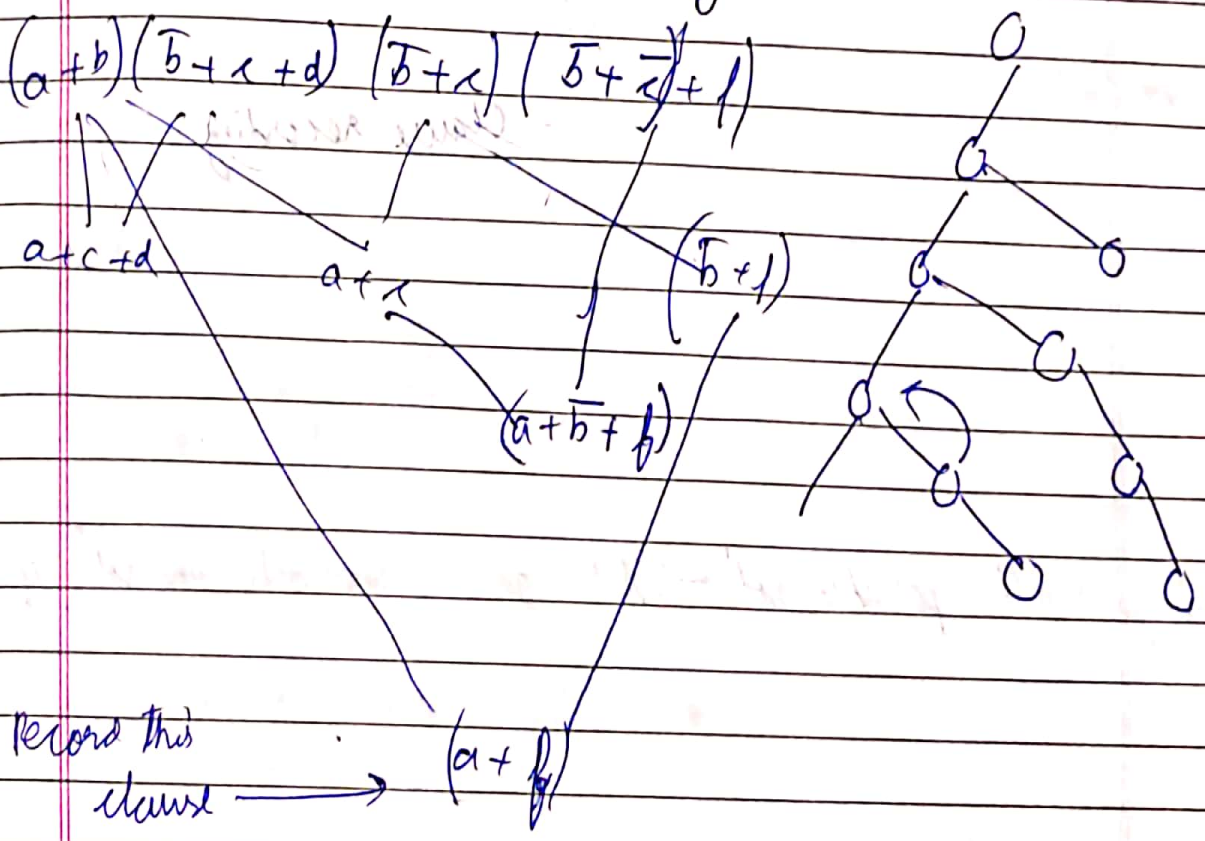
We could flip d which appears in more UNSAT variables

Local Search methods

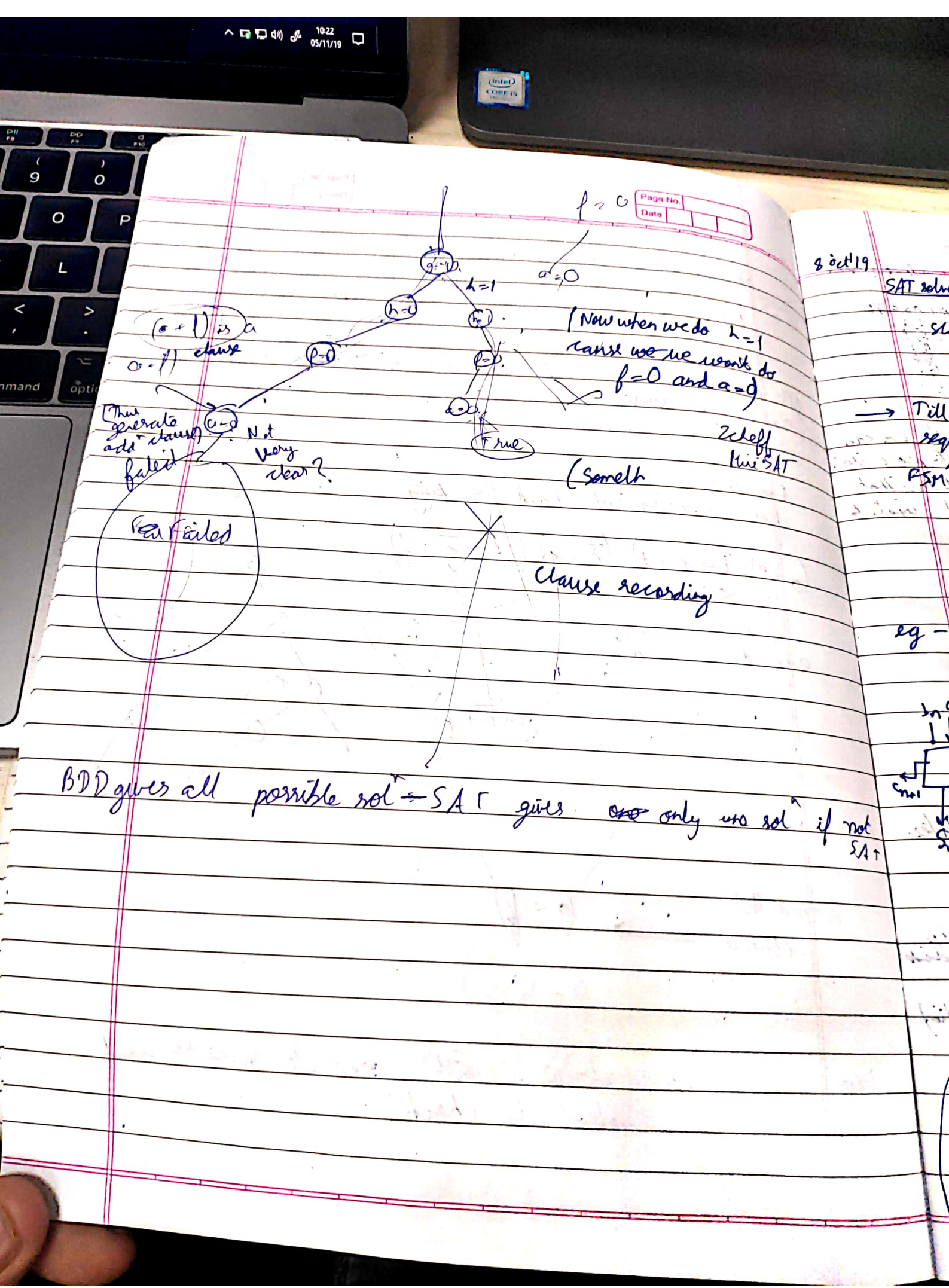
Other heuristics could be to do order variables as per their occurence in clauses.

Figure

Non-chronological back-tracking



This record clause will generate more unit clauses to check.



8 Oct '19

SAT solvers $f_1 = f_2$ Q M $\begin{matrix}0000\\1010\end{matrix}$ → 10_0

Static

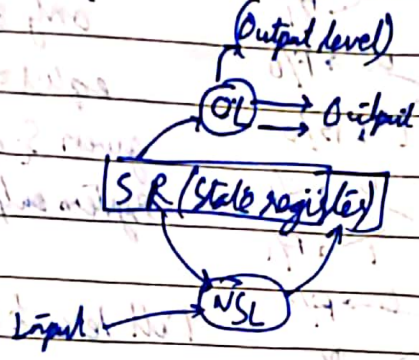
[Exact / Heuristic] ← 2 level ckt

[Set covering Problem] → EPI (essential prime implicant)

SPI : prime implicant

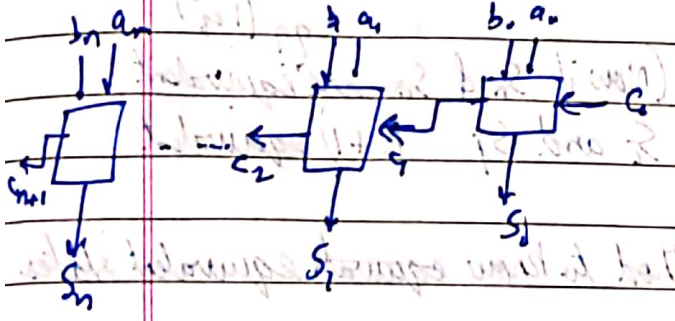
→ Till now we have static equations but now we will see sequential circuit (thus non-static equation)

FSM

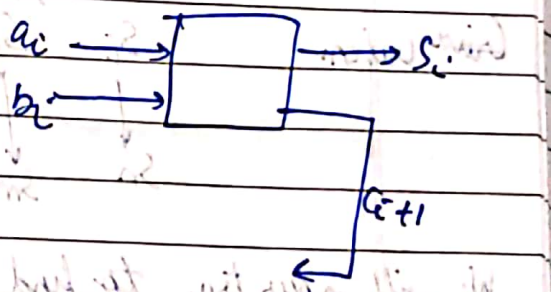


eg - Adder:

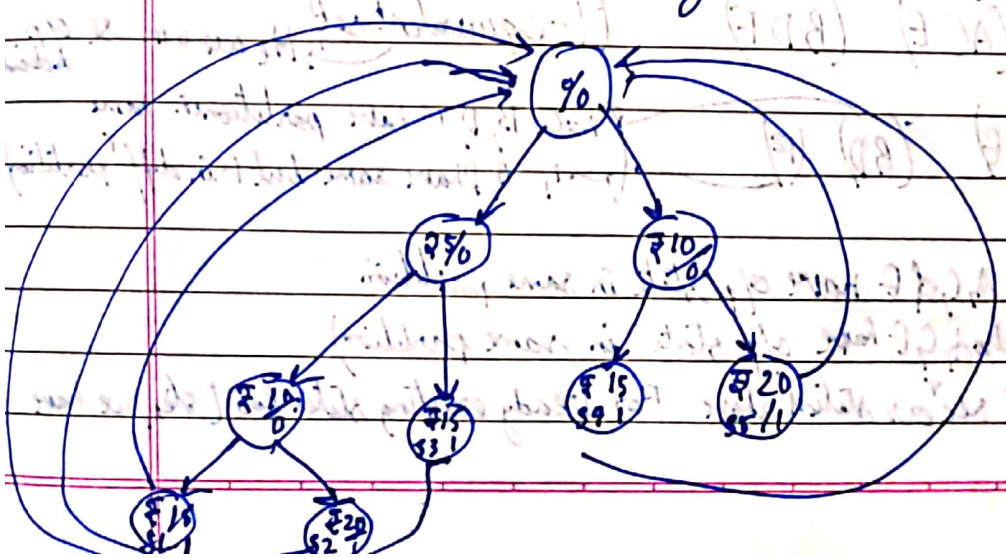
Without FSM:



With FSM:



State transition diagram



All S_1, S_2, S_3, S_4 S_5 are doing same job we could minimise the state.

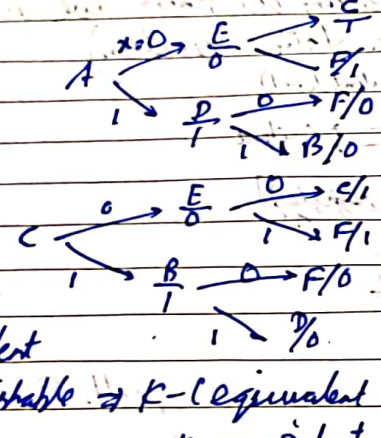
So you can minimise the states if the states have same output & same next state on same input

eg:

	NS & O/P (x=0)	NS & O/P (x=1)
A	E, 0	D, 1
B	B, 0	D, 0
C	E, 0	B, 1
D	E, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0

A, B: If we apply x=0 we see o/p=0 in both
But if we apply x=1, we see o/p=0 in B & o/p=1
So 1 distinguishable

→ A, C:



Till here it is 2 equivalent

Note: Physically we can only see output so to prove state equivalence, we must have that given S_i & S_j any no of inputs to System output is same.

Till here it is 2 equivalent
& distinguishable ⇒ k-1 equivalent
k equivalent

Given: S_i S_j

S_n S_m

(Now if S_n & S_m are equivalent (1 eq^n)
S_i and S_j (k+1) equivalent

will now try to find a method to know equivalent states

$P_0 = (A\,B\,C\,D\,E\,F)$ (0-equivalent)

$P_1 = (A\,C\,E)\,(B\,D\,F)$ (1-equivalent set, Here we use o/p values)

$P_2 = (A\,C\,E)\,(B\,D)\,(F)$ → (x=0 B, D, F have partitions same
x=1, B, D have same but F in diff^r partition)

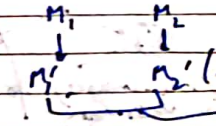
(for x=0, A, C, & E have o/p state in same partition
for x=1 also A, C, E have o/p state in same partition)

we use idea of rec^n as stated above For already existing states for (k+1), we have equivalence)

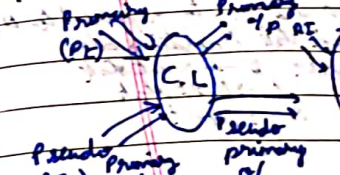
$P_3 = (AE)(C)(BD)(F)$
$P_4 = (AE)(C)(BD)(F)$ same

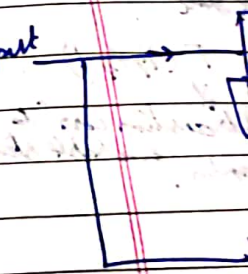
Now given 2 random state
→ Now possible way

Another way



⇒ We can use com





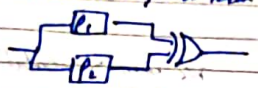
Total no of

Output (

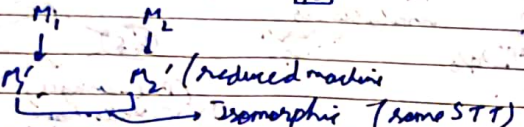
$P_3 = (A)(E)(B)(F)$
$P_4 = (A)(E)(B)(F)$ } same - Time step

Max. no of steps will be n (no of states)

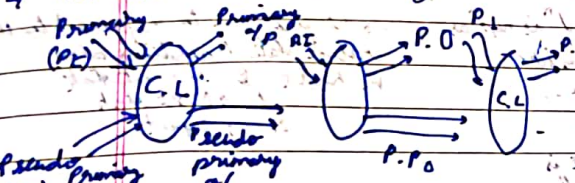
Now given 2 random state machines, how can we check equivalence.
→ Now possible way



Another way

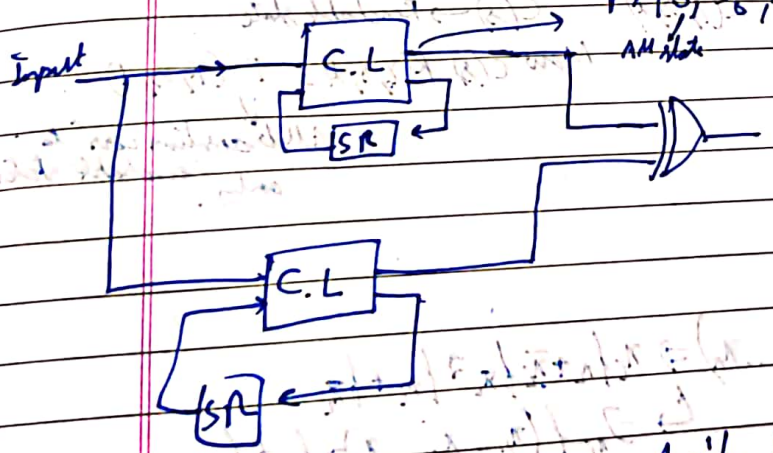


⇒ We can use comb^n logic also



This should be given to Mtlbs, after unrolling both machines with XOR of all o/p & AND of all. MST & R can give non-equivalence fast but not equivalence.

- Another way we should start for well defined initial state need not be same for both



$M(S, S_0, X, O, S, \lambda)$ → output definition
All state, state transition f^n

$M_1 = (S', S_0', X, O, S', \lambda')$

$M_2 = (S^2, S_0^2, X, O, S^2, \lambda^2)$

Now $M_1 \stackrel{?}{=} M_2$

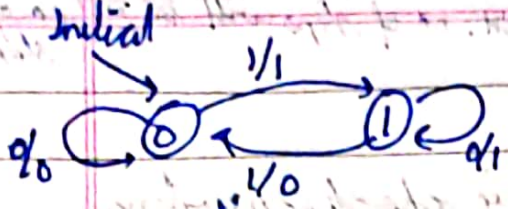
Total no of state this machine can have is $|S'| \times |S^2|$ [Product machine, as total no of state is product of both]

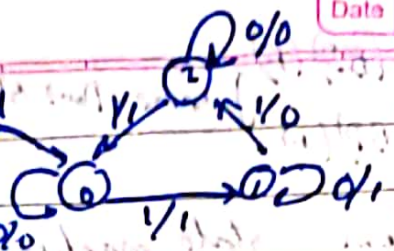
Output (MITER) = $\begin{cases} 0 & \text{when } q_1 \equiv q_2 \\ 1 & \text{otherwise} \end{cases}$

We can care about reachability states here. We do reachability Analysis

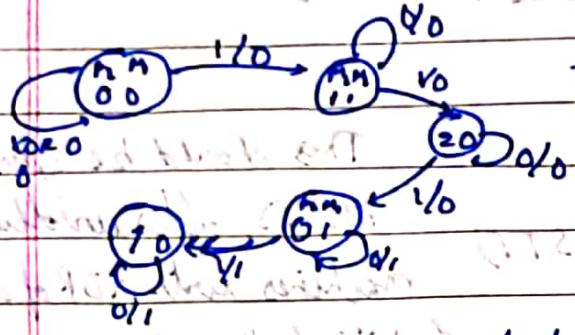
Page No.
Date

eg: Initial

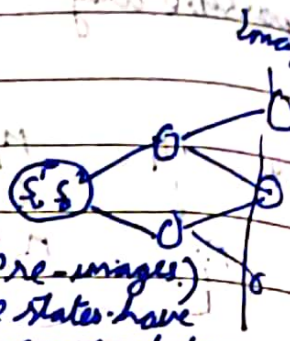




Product machine



We stop reachability analysis as we reach 0/0



These states have images as reachables

We now have to represent states in product machine - Binary rep^n / boolean f^n

eg- Boolean f^n:

	x_2	x_1	x_0	
→	0	0	0	1
	0	0	1	0
	0	1	0	0
→	0	1	1	1
	1	0	0	0
	1	0	1	0
→	1	1	0	1
→	1	1	1	1

→ This f^n is rep^n as a function like $\bar{x}_2 \bar{x}_1 \bar{x}_0 + \ldots$

ROBDD

Now if new state is added we basically add to function.

→ We also have to represent valid state transitions also we have s → Current state, t → Next state, x → Input.

Again, we represent as boolean f^n:

x	s_0	s_1	t_0	t_1	
0	0	0	0	0	1
0	0	0	0	1	0

So, transition relation $R(x, s, t)$

$C(s)$ → Reachable state

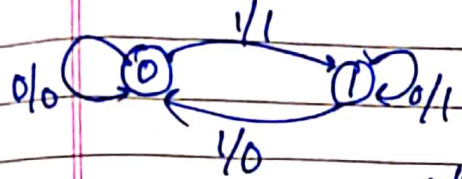
Now $C(s) \cdot R(x, s, t) = T(x, s, t)$

10 Oct 19:

2 state machines

$m_1 \equiv m_2$ $\quad M_1(S_1', S_0', X, O, \delta', \lambda') = M_2(S^2, S_0^2, X, O, \delta^2, \lambda^2)$

Reachability Analysis for $m = m_1 \times m_2 = (S', S_0', X, O, \delta'', \lambda'')$



$O = \begin{cases} 0 & (O_1 \equiv O_2) \\ 1 & \text{otherwise} \end{cases}$

2 states ⇒ 1 Boolean val for state

$\lambda(X, S)$

	0	1
0	0	1
1	1	0

$\lambda(X, S) = X \oplus S_0$

$\delta(X, S)$:

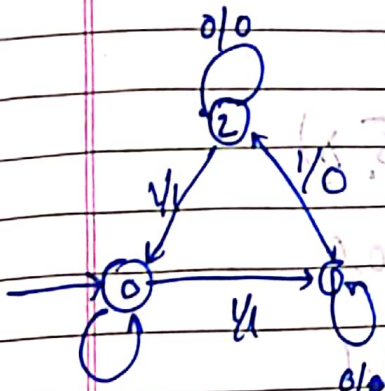
X \ S	0	1
0	0	1
1	1	0

$\lambda(X, S) = X \oplus S_0$

$S_0' = S_0$

$X = \{0, 1\}$

$S = \{0, 1\}$



$S_0^2 = \bar{S}_2 \bar{S}_1$

(first state check)

$\delta_1^2 = \bar{S}_2 \bar{S}_1 X + S_1 \bar{X}$

$\delta_2^2 = S_1 X + S_2 \bar{X}$

$\lambda^2 = S_1 \bar{X} + \bar{S}_1 X$

S_2S_1 \ X	00	01	11	10
0	0	0	X	1
1	0	1	X	0

X \ S_2S_1	00	01	11	10
0	0	1	X	0
1	1	0	X	1

X \ S_2S_1	00	01	11	10
0	0	1	X	0
1	1	0	X	1

$R(X, S, t) = (t_1 \equiv \delta_1').(t_2 \equiv \delta_2').(t_3 \equiv \delta_1')$

$= \Pi_i (t_i \equiv \delta_i')$

$= (t_0 \equiv S_0 \bar{X} + \bar{S}_0 X) \,/\, t_1 \equiv \bar{S}_2 \bar{S}_1 X + S_1 \bar{X}).(t_2 \equiv S_1 X + \bar{S}_2$

$S \longrightarrow s_2 s_1 s_0$

$t \longrightarrow t_2 t_1 t_0$

$t_0 \equiv \delta_1'$

$t_1 \equiv \delta_1^2$

$t_2 \equiv \delta_2^2$

$(X, S_2, S_1, S_0, t_2, t_1, t_0)$

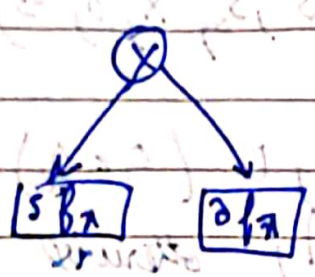
$(\bar{t}_2 \bar{t}_1 \bar{t}_0)(t_1 \equiv S_0 \bar{X} + \bar{S}_0 X)(t_2 \equiv \bar{S}_2 \bar{S}_1 X + S_1 \bar{X})$

$(t_1 \equiv S_1 X + \bar{S}_2 \bar{X})$

$\exists X (\bar{S}_2 S_1 \bar{S}_0)($ $)$

$C(s) = \bar{s}_2 \bar{s}_1 \bar{s}_0$

$= C(s)\, R(x, s, t)$ → First state time reachable → $C(s)$



$\delta_{fx} + \delta_{f\bar{x}} = \delta$

$T(t) = s_2 s_1 s_0 \,|\, t_2 t_1 t_0$

$C(s) + T(t)$ → obtain all reachable states from current state.

First state $= \bar{s}_2 \bar{s}_1 \bar{s}_0$

$= (t_1 = \bar{s}_2 \bar{s}_1 \bar{s}_0 x) \cdot (t_2 = \bar{s}_2 \bar{s}_1 \bar{s}_0 x)$

$x = 0 \Rightarrow t_1 = 0,\ t_2 = 0,\ t_3 = 0$ → 0,0

$x = 1 \Rightarrow t_1 = 1,\ t_2 = 1,\ t_3 = 0$ → 1,1

$t_1 = 1$; $t_2 t_2$: 0 1

$\Rightarrow 0 \to 1$ in m_1, $00 \to 01$ in m_2

$= \bar{s}_2 \bar{s}_1 \bar{s}_0 + \bar{s}_2 s_1 s_0$

19 Oct - 1

Page No.
Date

Logic Synthesis

Representation

1. Truth table
2. Min Terms → Cube Product
3. Max Terms
4. ROBDD

ROBDD → optimise for area / perf / power / testability

SAT
Reach

Verify ✓ SAT/BDD
Test ✓

Perf optimal

Important to find out good order of variables → Shifting → Multi level circuit

Important when you want to use for synthesis

Synopsis

Need to go through all n-mult ÷

K Map : → May be good for human beings (not for machines)

Not making any diff

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

Area
literal
Performance (level of imp ↑)

cube/implicants
cube/implicants

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

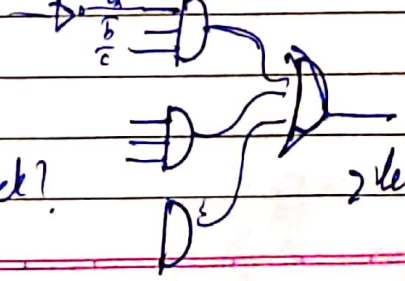
Roughly – area is prop to no of inputs. (app)

In TTL tech delay did not depend on no of inputs.

Buffers in CMOS delay depends on no of inputs. → Draw the CMOS circuits of gates and check?

→ delay & no of inputs

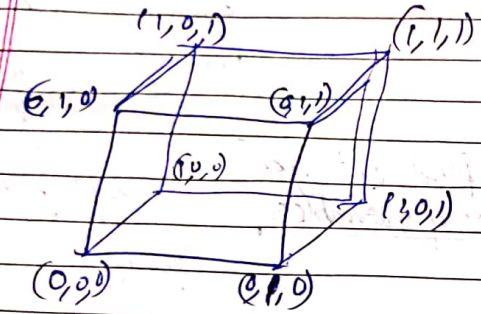
Restrict to 2/3 levels



2 level / 3 level

Page No.
Date

Delay of ckt $\approx 8 + 3\delta + 4\delta$
$\approx 8\delta$

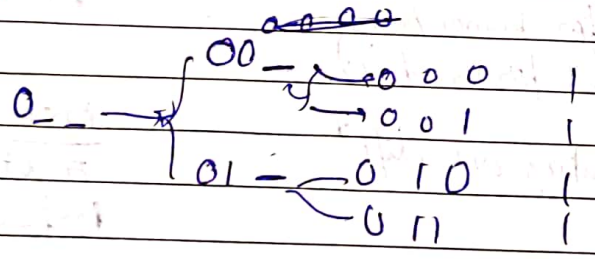


Quine — Logic optimization

Sharon — Also figured out minimizing digital logic

Quine - McCluskey method — scalable ✓

Tabular method



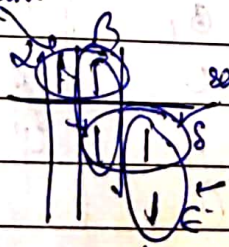
Pure implicant ← prime implicant

We have can use this

We could all prime implicants to implement a ckt - But may not be optimal

We could have essential prime implicants which have to be present. (Some terms are only covered by these)

Essential



Essential

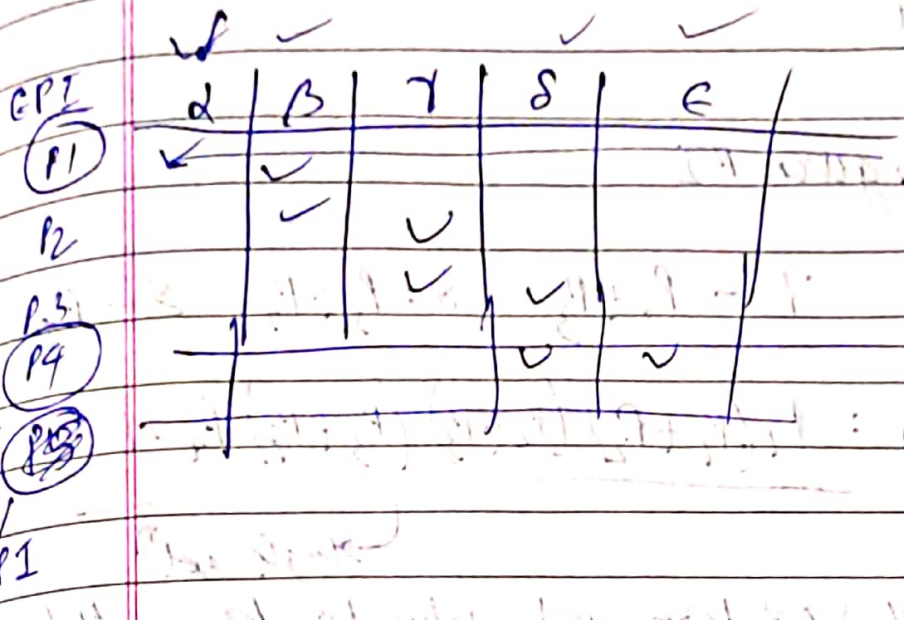
A, B are prime implicants but there are no redundancies.

EPI
P1
P2
P3
P4
EPI
EPI

Only thi
could be d

Only this which of selective PI's should be chosen?

could be done in 2 ways — set covering problem



For 1 we can select P2 or P3.

Patrick method

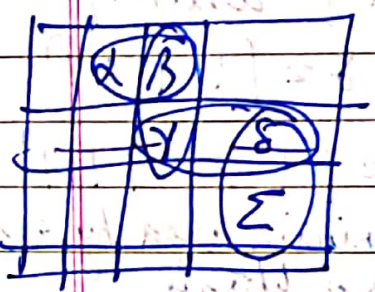
$\alpha \longrightarrow P_1$

$\beta \longrightarrow P_1 + P_2$

$\gamma \longrightarrow P_2 + P_3$

$\delta \longrightarrow P_3 + P_4$

$\epsilon \longrightarrow P_4$



Where we use logic exp?

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

should be satisfied

logic sat

SOP

choose the one with least no of literals

SAT

BDD implies OBDD

Logic optimisation

Exact optimisation

Patride's method

list all covers for given PI

$\alpha = P_1 \quad \beta = P_1 + P_2 \quad \gamma = P_2 + P_3 \quad \delta = P_3 + P_4 \quad \varepsilon = P_4$

For covering, satisfy : $P_1(P_1+P_2)(P_2+P_3)(P_3+P_4)P_4 = 1$ → logic sol^n

POS form → Convert to SOP form and solve for term with least no of literals.

15/Oct/19

Convert to matrix problem $Ax = b$ $(\alpha, \beta, \gamma, \delta, \varepsilon)$ = minterms P_1, P_2, P_3, P_4

$$\begin{array}{c} \\ \alpha \\ \beta \\ \gamma \\ \delta \\ \varepsilon \end{array} \begin{array}{c} \begin{matrix} P_1 & P_2 & P_3 & P_4 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

Let's assume, $\delta = [1\ 1\ 0\ 1]$ Not clear

$b = [1\ 2\ 1\ 1\ 1]$ (!)

→ minimise area by reducing no of literals

Find $Ax \geq 1$ Find x (min no of 1's)

→ Performance = literals + product terms

ROBDD ⟶ n level logic

2 level ckt ⇒ To synthesise ckt construct

Shannon's expression

$f(x_1, \ldots x_n) = x_i f_{x_i} + \bar{x}_i f_{\bar{x}_i} = x_i f_{x_i} \oplus \bar{x}_i f_{\bar{x}_i}$

~~boleanBoolean~~ difference

$\frac{\partial f}{\partial x_i} = f_{x_i} \oplus f_{\bar{x}_i} \Rightarrow \frac{\partial f}{\partial x_i} = 1 \Rightarrow f^n$ is dependent on x_i

$\Rightarrow f^n$ is independent of x_i

Properties

① $\frac{\partial f}{\partial x_i} = \frac{\partial f}{\partial \bar{x}_i}$ ② $\frac{\partial f}{\partial x_i} = \frac{\partial \bar{f}}{\partial x_i}$ ③ $\frac{\partial}{\partial x_i}(f \oplus g) = \frac{\partial f}{\partial x_i} \oplus \frac{\partial g}{\partial x_i}$

④ $\frac{\partial}{\partial x_i}(f+g) = f\frac{\partial g_i}{\partial x_i} + \bar{g}\frac{\partial f}{\partial x_i} + \frac{\partial f}{\partial x_i}\cdot\frac{\partial g}{\partial x_i}$

⑤ $\frac{\partial}{\partial x_i}(fg) = f\cdot\frac{\partial g_i}{\partial x_i} + g\frac{\partial f_i}{\partial x_i} + \frac{\partial f}{\partial x_i}\cdot\frac{\partial g}{\partial x_i}$

consensus $= f_{x_i} \cdot f_{\bar{x}_i}$ Smoothing $= f_{x_i} + f_{\bar{x}_i}$ ← Independence

$f = \sum_{i=1}^{n} \phi_i f_{\phi_i}$ $\phi_i \cdot \phi_j = 0 \Rightarrow \phi_i, \phi_j$ are orthogonal

$f = \phi_i f_{\phi_i} + \phi_j f_{\phi_j} \Rightarrow$ eg: $f = ab + \bar{b}c + ac$

$\phi_i = ab \quad \phi_j = \bar{a} + \bar{b}$ $\}\ \phi_i \cdot \phi_j = 0$

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

f_{ab} → cofactor terms of $ab = 1$

$\bar{a}+\bar{b} = 1$ iff $a/b = 1$

$f_{ab} = 1 \Rightarrow f_{(\bar{a}+\bar{b})} = bc + ac$

Positional vectors :

2 bit vectors

00 → ϕ

01 → 1

10 → 0

11 → X

$ab + bc + ac \rightarrow abc + abc$

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

ab ⇒

	a	b	c
	01	01	11

$bc \rightarrow$

	a	b	c
	11	01	01

$ac \Rightarrow$

	a	b	c
	01	11	01

$\bar{c}$	10	11	11

12 Oct 19

Logic optimisation → Enumerate
Exact optimisation

① Enumerate PI — QM
② Solve a set covering problem to find out the minimum PI

P_1, P_2 → containment

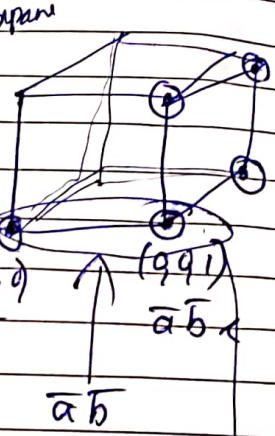
ESPRESSO
ESPRESSO - EXACT

ESPRESSO - Heuristic

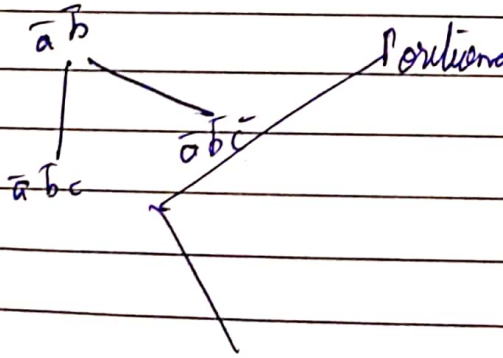
① Expansion
② Reduction
③ Reshape
④ Redundancy check

No further expansion

redundant



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



Positional representation

No expansion possible
- now reduce to cover the adjacent side

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

Positional rep:

a b
2 bits

00 → ϕ
01 → 1
10 → 0
11 → X don't care

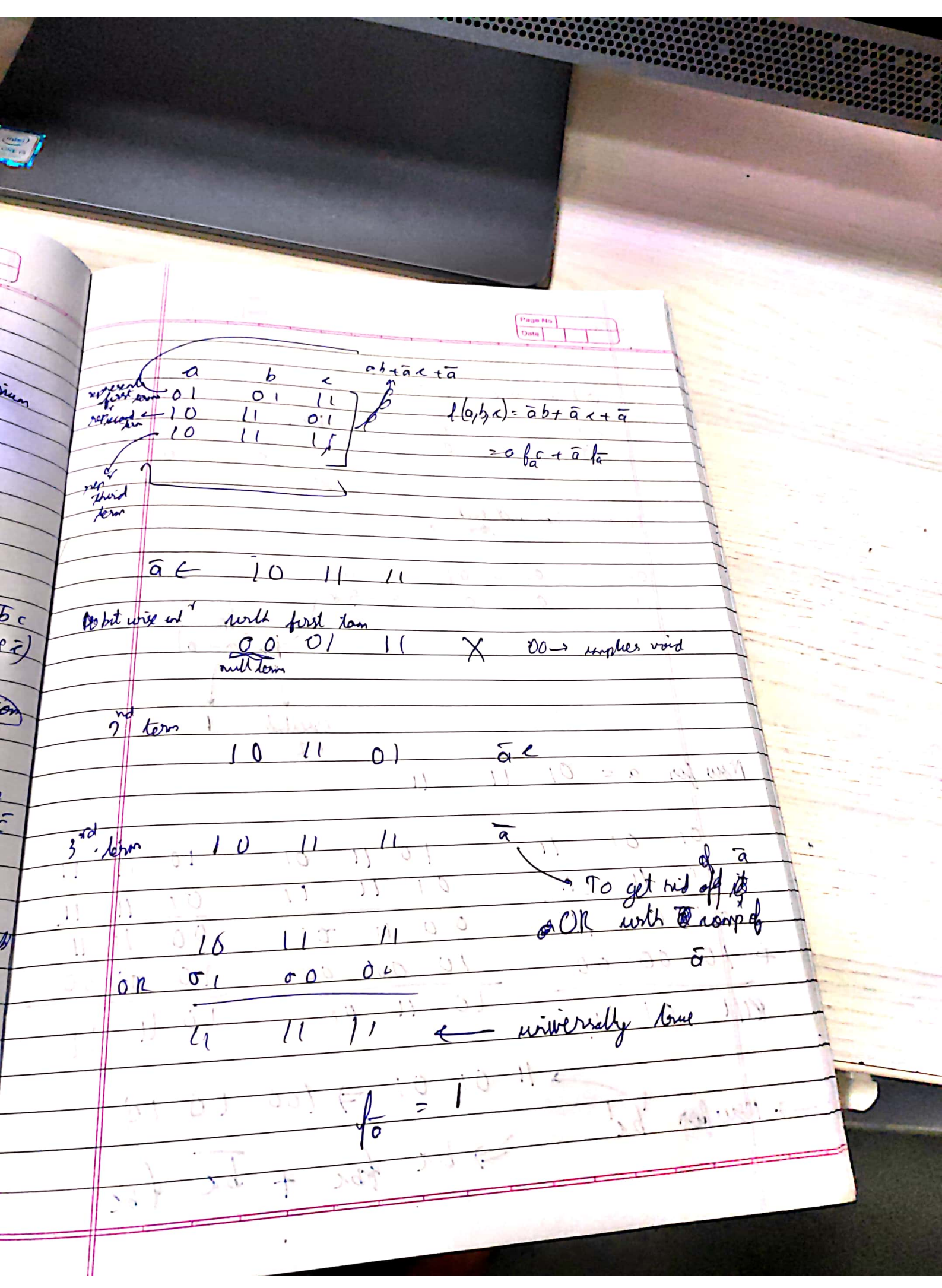
a b
01 01
10 11
10 11

represent first term
represent second term
rep third term

$\bar{a}$ ←
Do bit wise or

2nd term

3rd term



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

01 01 11 bc.
10 11 01
10 11 11 11 01 01

Do intersection with each term

01 01 01 ← ab
10 01 01 ← $\bar{a}c$

f_{bc}
$bc(a+\bar{a})$
essential P.I 1

Now for a = 01 11 11

01 01 11 10 11 01 10 11 11
01 11 11 01 11 11 01 11 11
01 01 11 00 11 01 00 11 11
10 00 00 10 00 00 10 00 00
11 01 11 10 11 01 10 11 11

→ Now for 'bc' → 11 01 01 ⇒ 00 10 10

⇒ $bc\, f_{bc} + \bar{b}c\, f_{bc}$

For each term

01	01	11
11	01	01
01	01	01
00	10	10
001	11	11

10	11	01
11	01	01
10	01	01
+ 00	10	10
10	11	11

$\bar{a}$

10	11	11
11	01	01
10	01	01
+ 00	10	10
10	11	11

$\bar{a}$

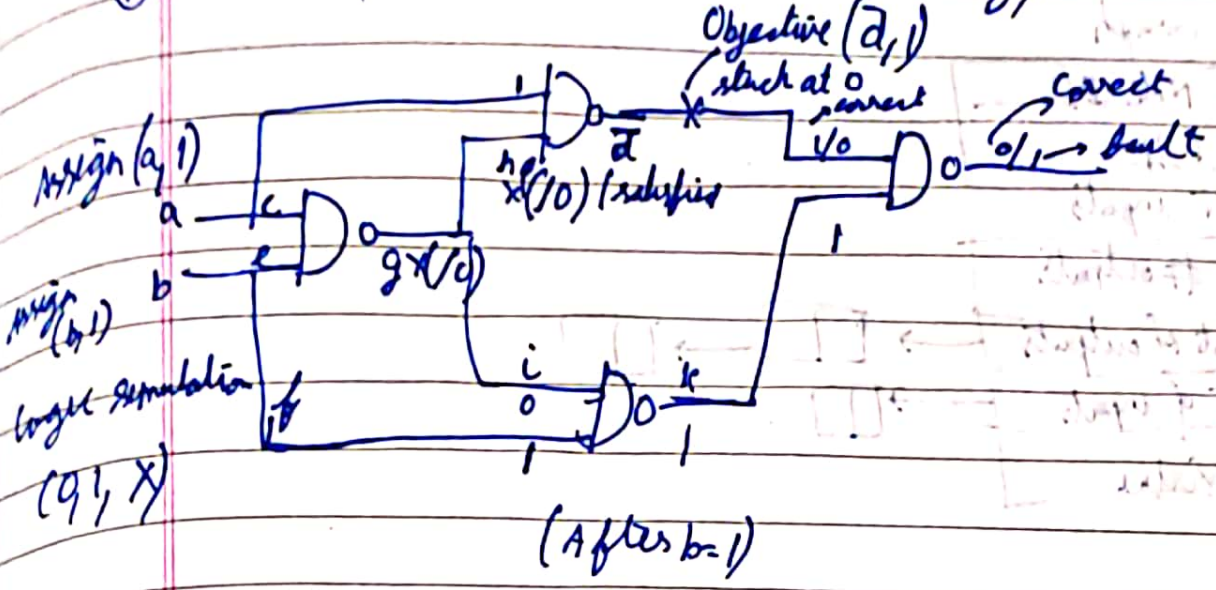
$a + \bar{a} = 1 \Rightarrow f_{bc} = 1 \Rightarrow$ Tautology

22 Oct 19
Test generation

Page No.
Date

① PODEM (Path Oriented Decision Making)

SAT for test generation

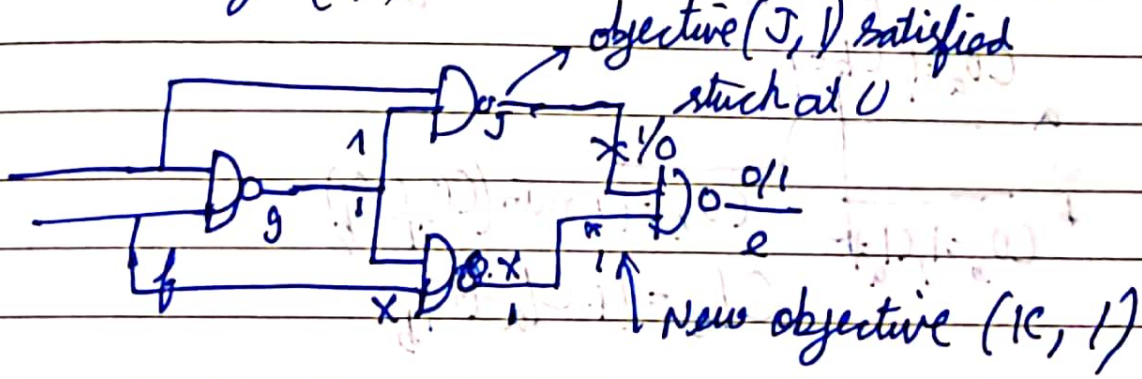


We satisfy objective

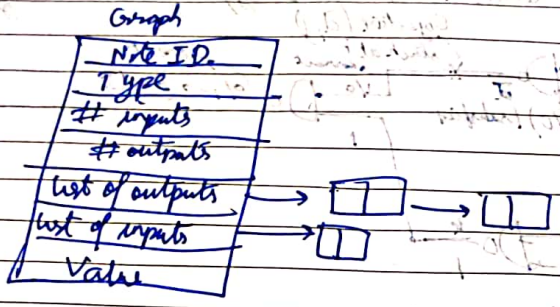
→ Not satisfied: still X → If not satisfied - re-traverse the path.

Conflict: Not satisfied confirm → Go back and change path assⁿ.

Other case assign (g, 0)



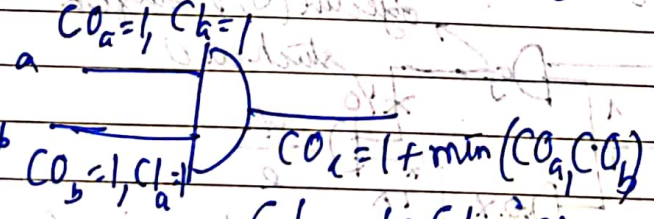
data structure to store this is a graph like it



We can have multiple paths like for we could go either go to a or b.

Ideally we want shortest path as it will be more controllable or observable.

So we know quantify observability and controllability



We can analyse this way on whole graph and set the controllability Now we use this metric to set path

eg - observability

Here we start for

O_A — O_C
O_B C_{I_b}

$O_A = 1 +$

$O_B = 1 +$

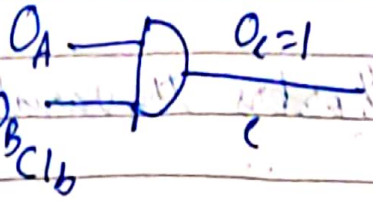
We flow fro observabl

→ While se simula

Secon once

B

Here we start for primary o/p & do reverse propagation.



Observability is same for 1 & 0 so only 1 value.

$O_A = 1 + O_c + C_{1_B}$ → (We need to assign B = 1 only then we can see O_A at output).

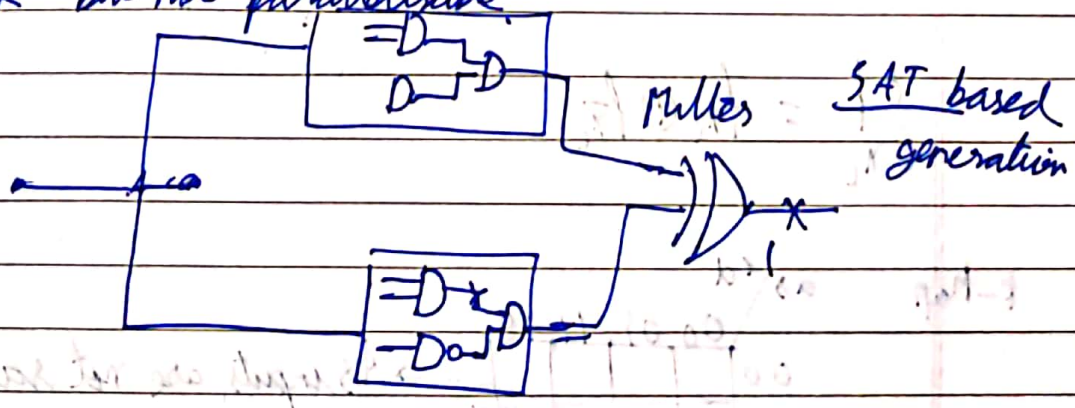
$\sim O_B = 1 + O_c + C_{1_A}$

We flow from primary output, to primary input & find observability.

→ While simulating, we must first decide the sequence of simulating the gates.

Second way → is event driven method ie. we simulate gate once we see a change in input.

Every change leads to 10-15% events, so we reduce work but not parallelisable.



Side note: ~~Given~~ Given a circuit path from primary output to input which is controllable and observable.

We can decide according to path orders the variable order better for BDD formation.

→ Synthesis (We will focus on ways to do/ease synthesis for better testability)

$$f(x_1, \ldots x_n) = x_i f_{x_i} \oplus \bar{x}_i f_{\bar{x}_i}$$

(Read Muller Decomposition)

$$f_{x_i} \oplus \bar{x}_i \frac{\partial f}{\partial x_i} \quad \text{OR} \quad f_{\bar{x}_i} \oplus x_i \frac{\partial f}{\partial x_i}$$

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

$$= f_{\bar{x}_i} \oplus x_i (f_{x_i} \oplus f_{\bar{x}_i})$$

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

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

K-Map as ↓d

	00	01	11	10
00				
01		1	1	
11			1	
10				

→ As inputs are not same, thus OR & XOR are same. So XOR

$\bar{a}bd \oplus abcd$ → Verify this is

2/10/19

Page No.
Date

Testability :

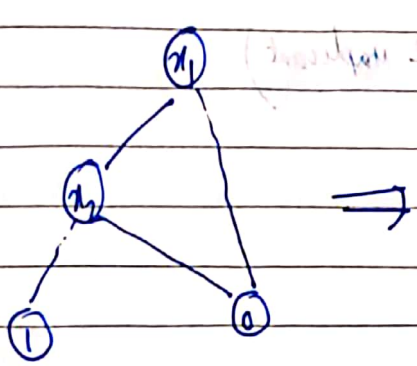
↓

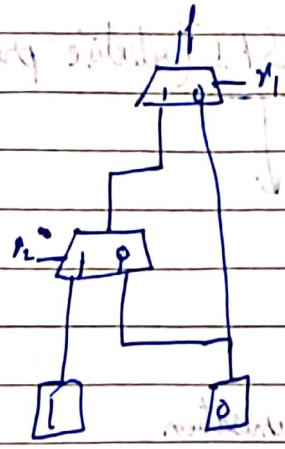
Read-Muller Expansion

XOR & AND | 100% testable | (n+4) test vectors (↗ universal vectors) | ROBDD

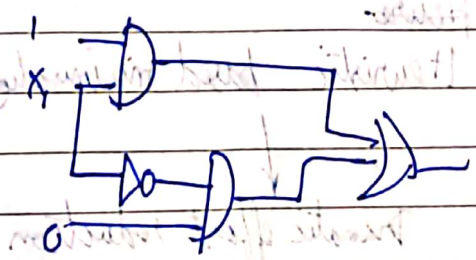
Faults may develop over time.

Often at power on, we check whether chip behaves correctly.

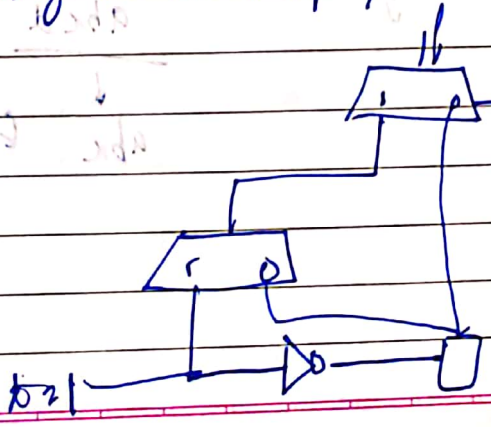




If we make MUX as



By having one extra input, we can make it 100% ~~testable~~ testable.



How to minimise a circuit using Exact method

Tabular method (1) QM → minimisation → Expresso / Presto

set of PI

set of essential PI &

Set of SPI (selective prime implicant)

(1) Cover
(2) CLP
(3) Matrix
(4) Boolean minimisation

~~Heuro~~

Heuristic based minimisation → No guarantees are ~~given~~ given

Dramatic effort reduction

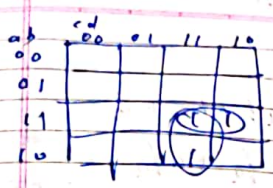
Optimal practically

(1) Expansion
(2) Reduction
(3) Reshape
(4) Irredundancy

abcd

↓

abc Expansion



Minterms

abcd

abc acd

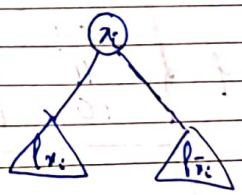
ON set
OFF set
DC set

Cost
(a) Terms containment
(b) Checkwhether conflict with OFF set

ON set defines function f (ignoring DC)

Some way of easy computation (computation of complement)

$$f(x_1, \ldots x_n) = x_i f_{x_i} + \bar{x}_i f_{\bar{x}_i}$$



$$\bar{f} = \overline{\left(x_i f_{x_i} + x_i f_{\bar{x}_i}\right)}$$

$$= \left(\bar{x}_i + \bar{f}_{x_i}\right) \cdot \left(x_i + \bar{f}_{\bar{x}_i}\right)$$

$$\boxed{\bar{f} = x_i \bar{f}_{x_i} + \bar{x}_i \bar{f}_{\bar{x}_i}}$$ ← Inversion of co-factor

- Unate : We say a function is unate w.r.t x_i if x_i is positive always uncomplemented

$\bar{a}$.

$\bar{a}bc + a$

$= \bar{a}bc + a(1+bc)$

$= bc$

Positional Representation of variable

$00 \longrightarrow \phi$

$01 \longrightarrow 1$

$10 \longrightarrow 0$

$11 \longrightarrow X$

$f\bar{a} = 1$ (tautology)

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

$f_a = \bar{a} \cdot f_{\bar{a}} + a \cdot f_a$

$f_{\bar{a}} \;\&\; f_a$

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

$f_a = bc + b\bar{c} + \bar{b}$

$= 1$

Using positional representation

$f = a\bar{b} + bc + \bar{a}$

f	01	01	11
f_a	11	01	01
$f_{\bar{a}}$	10	11	11

$a.b +$

co-factor w.r.t $\bar{a}$

$= \bar{a}$

	00	01	00
	01	01	11
	11	01	
	10		

Bit wise AND with

$\bar{a} \rightarrow$ 10 11 11

X	00	01	01
$\bar{a}bc$	10	01	01
$\bar{a}$	10	11	11

	10	01	01
OR	01	00	00
	11	01	01

	10	11	11
OR	01	00	00
	11	11	11

$\leftarrow$ TAUT

when we do

However to with input of $f_{\bar{a}}$

If any t ... is created.

let's compute fo

Terms repres

To get ... OR with ... representation

When we do first AND we get $\bar{a}$ in each term.

However to remove $\bar{a}$, we OR each term obtained with ~~invert of~~ bit wise invert of positional representation of $f_{a\bar{a}}$ (Check Why??) Not clear.

If any term gives all ~~one~~ 11's, we get tautology

OFF-set's is created from all ~~all~~ sets which give reverse tautology like $f_{a\bar{b}} = 0$

Let's compute for f_a $\quad a \rightarrow$ 01 11 11

Terms representation →

01 01 01
01 01 01
00 11 11 X repeted

To get rid of a's.

OR with compl of representation of a.

01 01 01
10 00 00
11 01 01 ← $b\bar{x}$ (ab)

01 01 11
10 00 00
11 01 11 ← $\bar{a}$ (b)
01

So now we do with $b = 11\ 01\ 11$

We will find fab & fa$\bar{b}$

$fab := 11\ 01\ 11$

$\quad\quad 00\ 10\ 00$

2nd term from prev

11 01 11

11 01 00

11 01 11

+ 00 10 00

11 11 11

Keep on doing

Tautology

fab

11 01 01

11 01 11 ← f_b

11 01 01

+ 00 10 00 ← $f_{\bar{b}}$

11 11 00

If a term f_{ab} evaluates to tautology, it (ab) is contained with function → it gives guarantee of containment

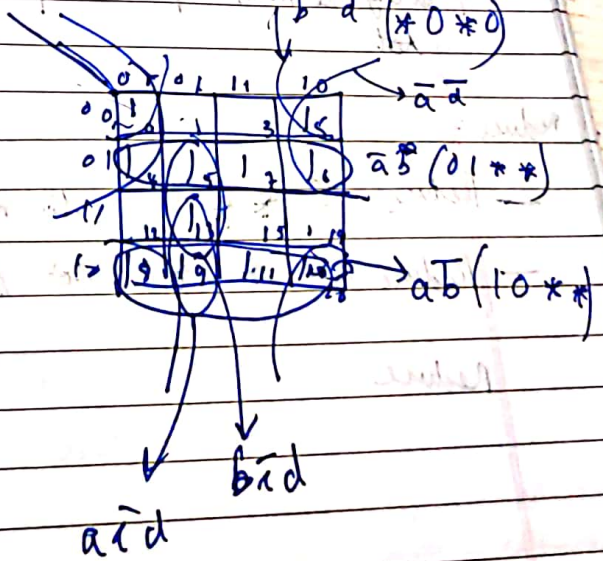
$f_{bc} = 1$ ← Tautology

(a-b)

Irredundant : There should not exist a term which on removal should not change function.

Example:

- Initial cover



Expansion: This is done for all terms present.

Creates prime implicants

- Expand 0000 to α = 0**0
  - Drops 0100, 0010, 0110 from covers
- Expand 1000 to β = *0*0
  - Drop 1010 from cover
- Expand 0101 to γ = 0*01 **
  - Drop 1011 from cover
- Expand 1101 to

If every term is part of multiple covers, reduction will not

Example of reshape.

- Reshape $\{\beta, \delta\}$ to $\{\beta, \delta'\}$

  $\{\beta, \gamma, \delta', \epsilon'\}$ — like changing a line to square and so on replacing square to a line.

- Second ~~expand~~ expansion

  $\delta' = 10*1$ to $\delta = 10**$ } → after
  $\epsilon' = 1101$ to $\epsilon = (*01)$ }

- Expansion — cover $\{\alpha, \beta, \gamma, \delta, \epsilon\}$
  — prime, redundant, minimal w.r.t. s.c.c

Steps taken by MINT

- Expansion
- Reduce
- Reshape
- Second expansion

→ This may redundant terms.

Finally after second expansion claim is that they are irredundant

Examples: Steps taken by EXPRESSO

- Expansion

  Cover $\{\alpha, \beta, \gamma, \delta, \epsilon\}$

  Prime, redundant, minimal w.r.t. s.c.c

  → None of the terms is contained with others.

- Irredundant

  Covers $\{\beta, \gamma, \delta, \epsilon\}$

  Prime, irredundant

Ordering heuristics

- Expand cubes that are unlikely to be covered by other cubes.

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

DC set $= ab\bar{c}$

Ordering

- Vectors $[3\ 1 \quad 3\ 1 \quad 3\ 1]^T$ → sum of

10	10	10
01	10	10
10	01	10
10	10	01

Thus choose terms in ~~order~~ ascending order.

→ And keep expanding so that it does not lead to ~~error~~ encroachment.

This is created recursively while expansion like $f_{x_0}=0$ earlier

OFF set

Expand - 01 10 10

01 11 01

11 01 01

11 10 10 valid

10 11 10 valid

11 11 11 invalid ← intersection with off set gives non-empty set.

~Expand 10 10 01

11 10 01 invalid } Expand

10 11 01 invalid

10 10 11 valid

Expand cover 10 10 11

31 Oct '19

3 level implementation :: Performance

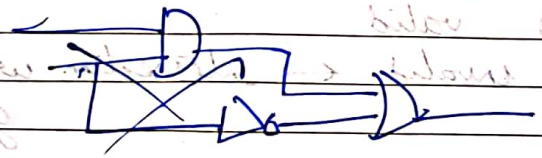
INV ADD OR

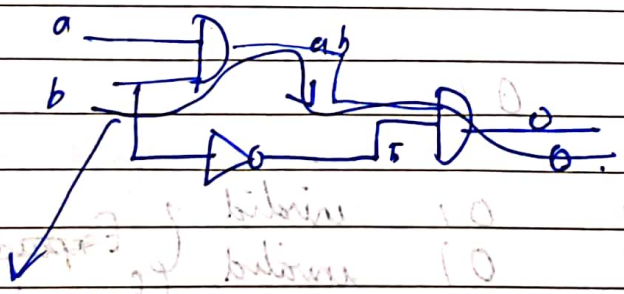
Area → # gates & # literals

~~multi~~ multi-level

Logic sharing = $a.b + a.c = a.(b+c)$

Delays are characterised by fan-outs.



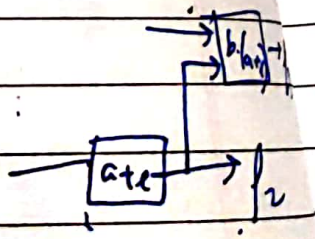


(non-sensitizable path)

Wherever we use longest path for det^r clock cycle, we should be using sensitizable path.

Say : $f_1 = ab + bc$
$f_2 = a + c$

} By sharing, we can do better



Transformations :

Eliminate one

Example.

Decompo

$Y = a$

Extracti

Opti

Transformations :

Eliminate one func" from network

Example. $s = r + b'$ $r = p + a'$

$s = p + a' + b'$

Decomposition : Introduce new variables /blocks

$v = a'd +$

Extraction : Common sub expressions

Simplify local expression

Optimization approaches

Boolean alg and algebraic methods

Use don't care op"

Support set of quotient and remainder could ort & in boolean algebra.

4 Nov '19

Libraries and Mapping

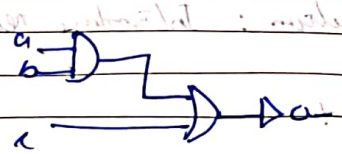
LS → Logic gate level netlist

3 level
n-level

Either input ... cells (2 ...

OR have multiple ...

AOI gate

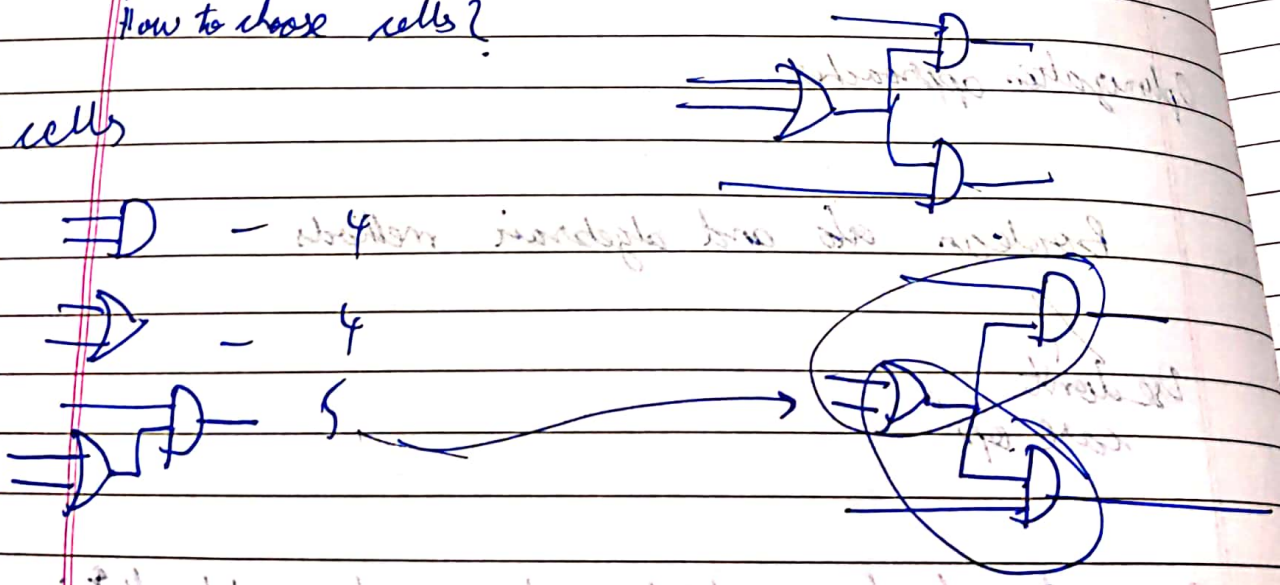


Newer technology gates interconnect as well.

Multiple library cells: Interconnect instances of library cells

~~We should have cost~~
How to choose cells?

3 cells



Vertex ~~cover~~ covering → Ve

m_1 : v_1, OR2
m_2 : v_2, AND2
m_3 : v_3, AND2
m_4 : v_1, v_2, OA21
m_5 : v_1, v_3, OA21

v_1 : {m_1 + m_4 + m_5}
v_2 : {m_2 + m_4}
v_3 : {m_3 + m_5}

Input compatability

Set problem

We can a

Convert ... cost of

Partitioning

Input compatability :

Match m_2 requires m_1 $(m_1 \rightarrow m_2)$

Match m_3 requires m_3 $(m_3 \rightarrow m_3)$

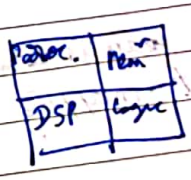
$m_3' + m_3$

Set problem :

We can a terrible solution if $m_1 = m_2 = m_3 = \cdots = 1$

Convert in SOP and evaluate cost of all and now evaluate cost of all.

Partitioning and Floor Mapping

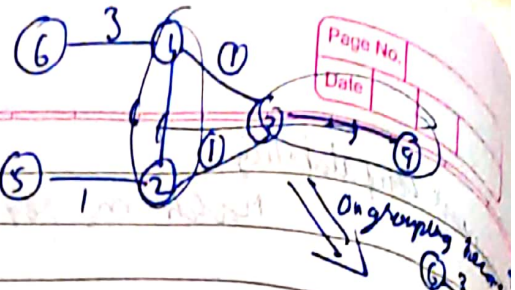


(Planning)

5 Nov '19
41

Here we first group those vertices with heaviest weight bet^n them.

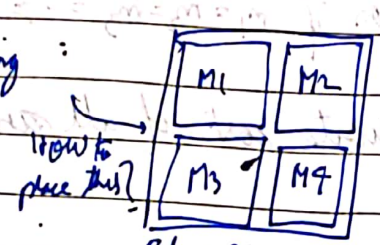
K – L algorithm



Collapse vertices into single groups and ~~equivalent~~ also combine required edges. $O(n^3)$

Fiduccia – Mattheyses (FM) alg^n – $O(n^2)$

Partitioning :



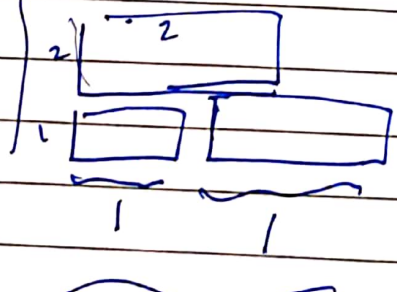
Floor Plan / placement

Pin Input / Output pins in each module – exact location.

Constraint on shape of block

way is to map

Easiest mapping can be done to bin packing problem



Min (W x L)

Only horizontal and vertical placements are done

Find out co-ordinates (of each right-hand top corner)

of each rectangle placed such that min (W, L) gives no. rectangles (modules) do not overlap.

(for

constraints

Standard cell - to Derivations of each cell is design

module [Sub-module]

Could also use FPGA's where all gates are standard cells

Wiring length estimated as $\sum_{i,j} (c_{ij} * d_{ij})$ — c_{ij} - Connectivity bet^n i & j; d_{ij} - Manhattan dist^n bet^n centres of i & j

Manhattan distance bet^n (i_1, j_1) & $(i_2, j_2) = |i_2 - i_1| + |j_2 - j_1|$

(i_1, j_1) Euclidean (i_2, j_2)

We would minimise $\alpha A + (1-\alpha) L$

Slicing structure - Place a tree (minimise empty space)

Routing wires: How to connect wires bet^n modules (which channels to use

Routing

Global (How you want to route? Through which channels?)

Detailed exact routing (How exactly connect)

If routing does not happen on same layer, we may need a new layer (costs a lot)