Logic Synthesis: 2 level Logic - Basics

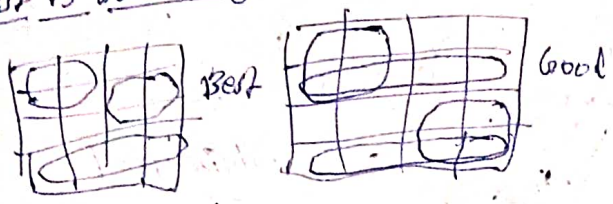
Fewest AND gates & fewest I/p wires
Boolean Algebra // hard
Kmaps // hard
Tabular Solution: Quine McCluskey // exponential complexity

Better Strategy

#Idea 1: Don't try for best perfect answer but good answer

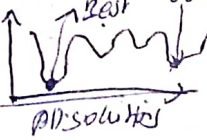
#Idea 2: Iterative improvement / retype solution

Best vs Good Enough



"Prime Implicant"

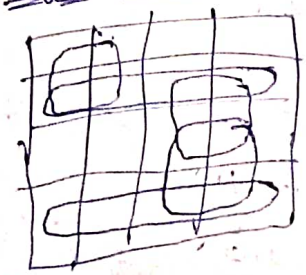
1950: Best solution has Prime Implicants
Both solutions are Irredundant
Need different Idea



Consider

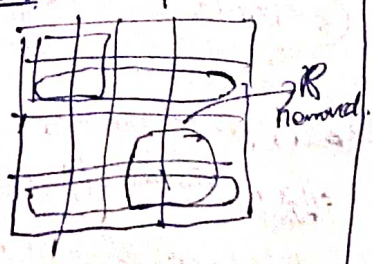
abcd	F	cube-label
0-0-	1	P
0-10	1	Q
1001	1	R
1101	1	S
1-11	1	T

"Expand" heuristic → make each cube "bigger as possible"
"Remove"



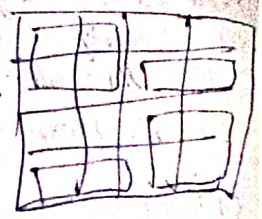
Remove redundant cubes

abcd	F
0-03	1
0-10	1
100-	1
1101	1
1-11	1



"Reduce"

Shrink it, don't uncover
None of them overlap



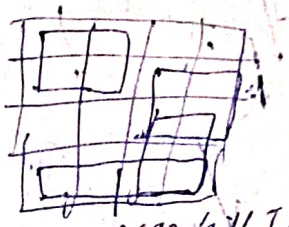
- It may not be prime cover

abcd	F
0-0-	1
0-10	1
1001	1
1101	1
1-11	1

- we can still get better

Expand again

abcd	F
0-0-	1
0-10	1
1001	1
1101	1
1-1-	1



we can kill T cube

Final; optimized, Prime and Irredundant cover
Good local solutions, Fast

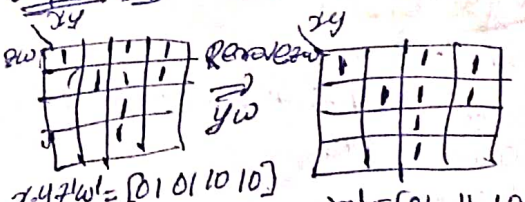
Reduce ⇒ Expand ⇒ Irredundant

Tool: ESPRESSO, 2 level minimization

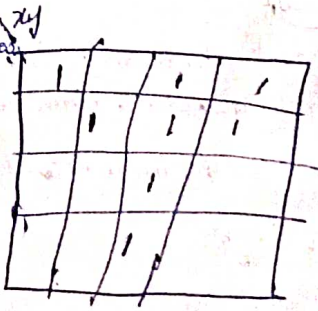
Details for One Step

PCN - positional cube notation
clever ordering of cubes, ORP

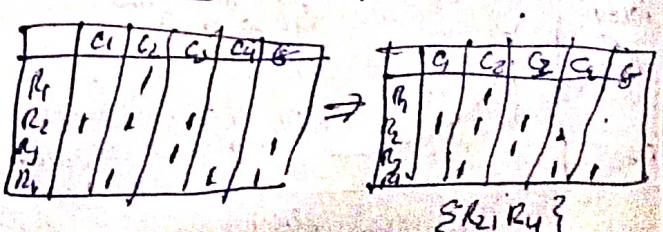
Expand step



Remove



Covering Problem (R x C)



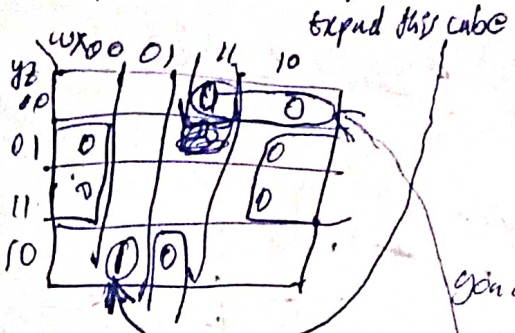
Blocking matrix

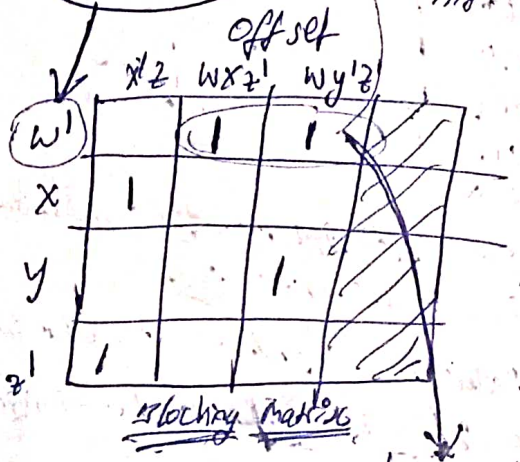
$F = w'y'z' + xz + z'yz' + w'xyz'$

↓ URP complement

$F' = z'z + wxz' + wy'z'$

Off set





Espresso

Reduce
Expand
Irredundant

- clever URP + covering problem

Other methods:

- minimize several functions at the same time. (shared and gate)
- don't cares

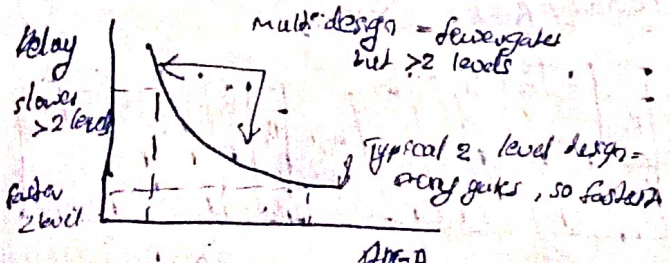
Very Fast/Robust

ESPRESSO: Complement, Expand, Irredundant, Essentials, Reduce, Various Optimizations

→ Reduce - expand - irredundant loop

Multi Level Logic

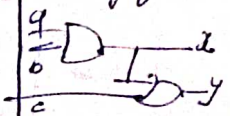
2-level - Restrictive



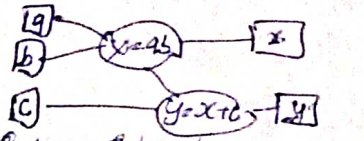
Have to use 2-level for big things

Boolean Logic Network

Ordinary logic

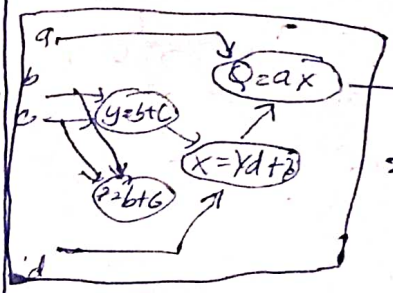


Boolean logic nw



What it will optimize?

- Total literal count



Data Structure: operators

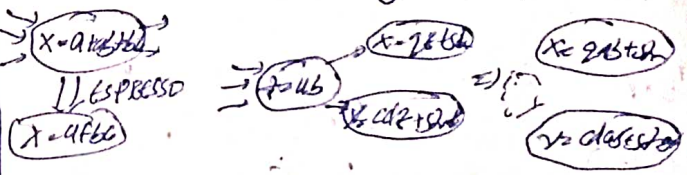
Simplify one node: ESPRESSO

Remove one node: Collapse too small nodes. substitute fanouts, simple SOP algebra

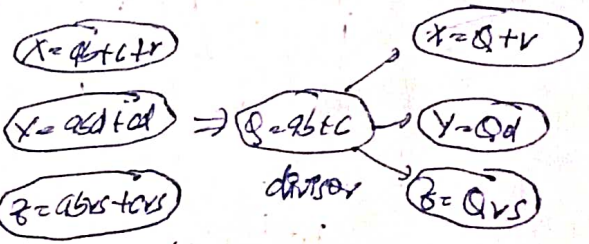
Add new one node: factoring, split into smaller nodes

Simplify

Removing a node



Adding: Factoring.



16 literals

- 5 GFPs of basic operator

Algebraic Model: Factoring, Tradeoff

Polynomials of real numbers,

Algebraic division: weak division.

axioms

Real numbers	Boolean algebra
$a \cdot b = b \cdot a$	$a \cdot b = b \cdot a$
$a+b = b+a$	$a+b = b+a$
$a \times (b \cdot c) = (a \cdot b) \cdot c$	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$
$a+(b+c) = (a+b)+c$	$a+(b+c) = (a+b)+c$
$a \cdot (b+c) = a \cdot b + a \cdot c$	$a \cdot (b+c) = a \cdot b + a \cdot c$
$a \cdot 1 = a$ $a \cdot 0 = 0$	$a \cdot 1 = a$ $a \cdot 0 = 0$
$a+0=a$	$a+0=a$

Not allowed: $a+a'=1$ $a \cdot a' = 0$

$a \cdot a = a$ $a+a=a$ → idempotent

$a+1=1$

$a+(b \cdot c) = (a+b) \cdot (a+c)$ distributive

$F = ab + a'x + b'y$ let $R = a'$, $S = b'$

⇓

$ab + Rx + Sy$

Algebraic SOP polynomials

set of variables eg a, b, R, y

list of these products (cubes)

eg $ab + Rx = ab, Rx$

Algebraic Division

$F = D \cdot Q + R$

divisor, Quotient, Reminder

Strategy: Cubewise walk through cubes.

```
Algebraic division (F, D) {
    for (each cube d in divisor D) {
        let C = {Cubes in F that has product term d};
        if (C is empty) {
            return (quotient = 0, reminder = F);
        }
        let C = cross out literals of d in each Cj
        if (d is the first cube we have look at D)
            let Q = Cj
        else Q = Q ∩ Cj
    }
    R = F - (Q · D);
    return (quotient = Q, reminder = R);
}
```

F cube	D cube: ax C = ...	D cube: b C = ...
axc	axc → c	—
axd	axd → d	—
axe	axe → e	—
bc	—	bc → c
bd	—	bd → d
de	—	—
	$Q = c+d+e$	$Q = c+d$

⇒ $c+d$ → quotient

$R = (axc + axd + axe + bc + bd + de) - (c+d) \cdot (ax+b)$

$= axe + de$ = reminder

Redundant cubes

R - must have no redundant cube

Multilevel synthesis models

F/D, $F = Q \cdot D + R$

algebraic model, set of good common divisors

$F_1 = ab + C + x$ — factor → $Q = ab + C$ → $F_1 = Q + x$

$F_2 = abx + Cx + q$ → $F_2 = Qx + q$

$F_3 = ab + q$ → $F_3 = ab + q$

---

where to look for divisors of F → in Kernels of F

$K(F)$ → SOP,

$K \in K(F)$? Algebraically divide F by one of its co-kernels

Kernel

Cube free quotient K obtained by dividing F by single cube C. C → co kernel

divisor D / expression F	co kernel C / expression F
quo Q, rem R	kernel K of cube free, rem R
$F = D \cdot Q + R$	$F = C \cdot K + R$

cube free → no one cube product

Expression F	F = d·Q + R	cube free
a	a(1) + 0	No
a+b	–	Yes
abc+ac	(b+1)(a) + c	No
abc+abd	ab(c+d)	No
ab+acd+bd	–	Yes

Kernal K(F), F = abc + abd + cd

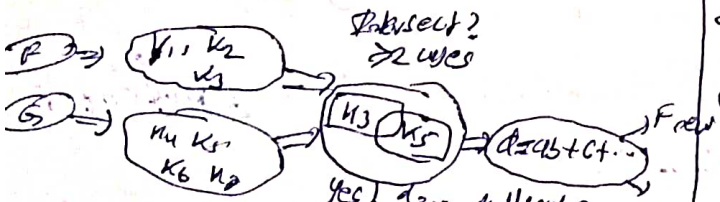
Divisor used	F = dQ + R	Is it kernal of F
1	(1)(abc+abd+bd) + 0	No
a	a(bc+bd) + bd	No
b	(b)(ac+ad+cd) + 0	Yes, kernal, co-kernal = (b)
ab	(ab)(c+d) + bd	Yes, kernal, co-kernal = (ab)

#Result Brayton & McMullen Theorem.

Expression F, G have a common multiple cube divisor d if and only if there are kernals $k_1 \in K(F)$, $k_2 \in K(G)$ such that $d_1 = k_1 \cap k_2$ (sop of common cubes) & d is an expression with atleast 2 cubes in it

~~#~~

multiple cube divisors → Extensive of kernals



F = cube1 · kernal2 + remainder1 ⇒

G = cube2 · kernal2 + remainder2 ⇒

F = cube1 · [X+Y] + [cube1 stuff + rem2]

G = cube2 · [X+Y] + [cube2 stuff + rem2]

d = X+Y → F = d(cube1 stuff ...)

→ G = cube2 · d + ...

kernal1 ∩ kernal2 = [X+Y]

⇒ a multicube divisor of F&G

Consider F, G

F = ae + be + cde + ab G = ad + ae + bd + be + bc

K(F) kernal	Co-kernal
a+b+cd	e
b+e	a
a+e	b
ae+be+cde+ab	1

K(G) kernal	Co-kernal
a+b	de
d+e	a or b
d+e+c	b
ad+ae+bd+be+bc	1

2 kernals (a+b+cd) ∩ (a+b) ⇒ a+b

multicube divisor

Kernals (F and a kernal K_1 of K(F))

→ F = cube1 · K_1 + remainder1

K_2 is inside K_1.

→ K_1 = cube2 + K_2 + remainder2

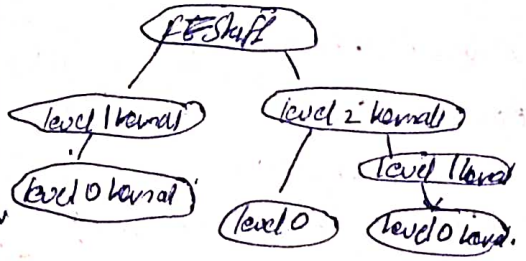
→ F = cube1 · [cube2 · K_2 + remainder2] + remainder1

= [cube1 cube2] [K_2] + [cube1 · remainder2 + remainder1]

= (cube)(cube free Q) + [other stuff]

⇒ K_2 is also kernal of F with ~~co-kernal~~ (cube1 · cube2)

$K \in K(F)$.



Kernal Hierarchy

Brayton et al

co-kernals of sop correspond to intersection of 2 or more cubes.

Ex: - ace + bce + de

ace ∩ bce = ce → co-kernal

ace ∩ bce · de = e → e → co-kernal.

Recursively, findkernal(G)

--- co-kernals at intersection of cubes

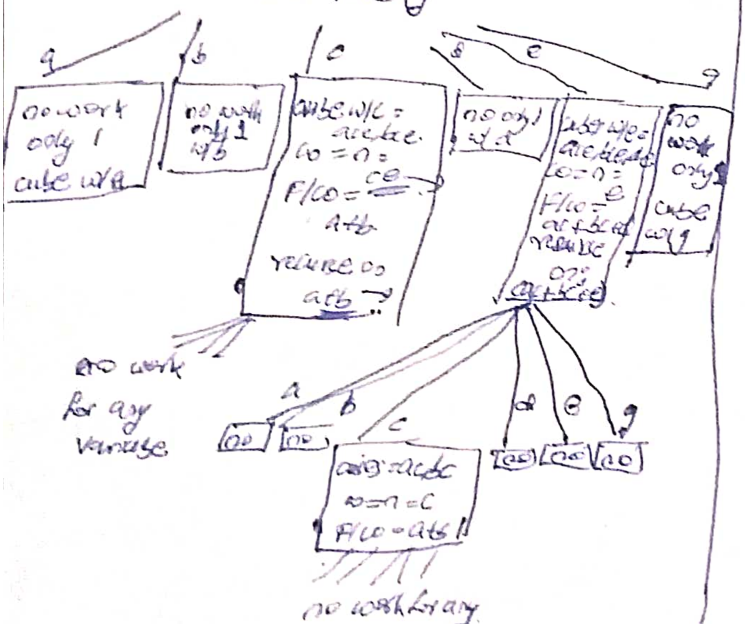
--- atleast 2 cubes, intersection of cubes for co-kernals

→ Need to start with cube free function F

Algorithm : Kernal & co-kernals

```
FindKernals (cube free SOP expression F){
    K = empty;
    for (each variable x in F){
        if (there are atleast 2 cubes in F that
            have variable x){
            let S = {cubes in F that have x in them}
            let co = cubes that intersects of cubes in S.
                product of each root & each cubes;
            K = K U FindKernals(F/co);   <- F/co -> kernal
        }
    }
    K = K U F;
    return(K)    (cube free F co-kernals)
}
```

F = ace + bce + de + g

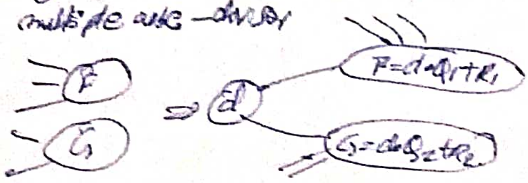


revisit same kernal multiple times

Using Kernals & co-kernals

Single cube - divisor

multiple cube - divisor



Single cube divisor : co-kernals

multiple cube divisor : Kernals