EE403

(1) 06101980

T		S	K
7.55	7.35	7.30	8.10
9	8.30	8.30	9
10.15	9.45	9.30	10
11.15	10.45	10.30	12
11.45	12	12.30	13.45
12.45	12.45	14.15	15.45
14.15	13.45	16.15	16.45
15.15	14.45	17.30	19
16.15	15.45	19.30	20
17.15	16.45	20.30	22
18.15	14.45	22.30	23.15
20	19.30	23.45	
21	22		

must be known :

- numbers and number systems
- Arithmetics
- Coding BCD, XS-3, gray
- (+) and (-) numbers and their representations in any base
- Laws of Boolean algebra
- Simplification of switching functions using algebraic manipulations.
- Basic idea about the design procedure of combinational cct's.

Summary of the course :

- Combinational cct's.
- Hazards in comb. cct's.

MT1
- Analysis and synthesis of synchronous switching cct's.

MT2
- Analysis and synthesis of asynchronous swt. cct's.

References

- Logical design of digital systems. Donald Dietmage.
- Introduction to switching theory and logical design. Hill and Peterson
- Switching and finite automata. Z. Kohavi
- Mano

07101980

Example : given $B = \{1, 3\}$ together with the operations $\forall a, b$ in B.

$a + b \equiv$ The least common multiple of a and b. (max a, b)

$a \cdot b \equiv$ The greatest common divisor (Min a, b)

$a' = \frac{3}{a}$

Check whether satisfies the laws of Boolean algebra :

+	1	3
1	1	3
3	3	3

·	1	3
1	1	1
3	1	3

a	a'
1	3
3	1

A1 – Commutativity holds. (Can be seen through the tables)

A2 –

$1 + 1 = 1$ $\quad 3 + 1 = 3$

$1 \cdot 3 = 1$ $\quad 3 \cdot 1 = 1$ $\quad \rightarrow \quad 1_+ \equiv 1$

$1 \cdot 1 = 1$ $\quad 3 + 3 = 3$ $\quad\quad 1_\cdot \equiv 3$

$1 + 3 = 3$ $\quad 3 \cdot 3 = 3$

$1_\cdot \cdot a = a$

$1_+ + a = a$

$1_\cdot + a = 1_\cdot$

$1_+ \cdot a = 1_+$

A3 – $a + (b \cdot c) = (a+b) \cdot (a+c)$

Let $a = 1$

$1 + (b \cdot c) = (1+b) \cdot (1+c)$

$b \cdot c$; b (look at the table) ; c

$= b \cdot c$ ✓

Let $a = 3$

$3 + (b \cdot c) = (3+b) \cdot (3+c)$

$3 = 3 \cdot 3$

$3 = 3 \Rightarrow$ A3 holds. ✓

Check: $a \cdot (b+c) = a \cdot b + a \cdot c$

A4 –

$a + a' = i_.$ $\quad \{ i_. = 3 \}$

$a \cdot a' = i_+$ $\quad \{ i_+ = 1 \}$

Let $a = 1$: $1+3 = 3\ (= i_.)$; $1 \cdot 3 = 1\ (i_+)$

Let $a = 3$: $3+1 = 3\ (i_.)$; $3 \cdot 1 = 1\ (i_+)$ ✓

Exercise $B = \{1, 2, 3, 6\}$

$a + b \equiv$ given previous ex.

$a \cdot b \equiv$

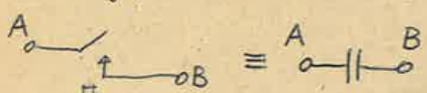
$a' = \frac{6}{a}$

Switching algebra :

if there is a transmission between points A and B $= 1$ |

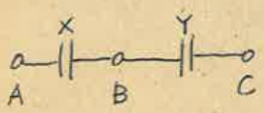
" " " no " " " " " $= 0$ |

1. Normally open contact :



2. Normally closed contact :

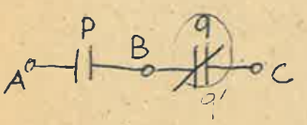
If two contacts are connected in series ;



→ AND

1 means energizing

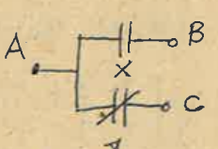
0 means de-energizing



P	q	T_{AC}
0	0	0
0	1	0
1	0	1
1	1	0

$T_{AC} = P \cdot q'$

Transfer contact :



Same relay coil controls both of the contacts.

when $x = 0$

$T_{AB} = 0$

$T_{AC} = 1$

$x = 1$

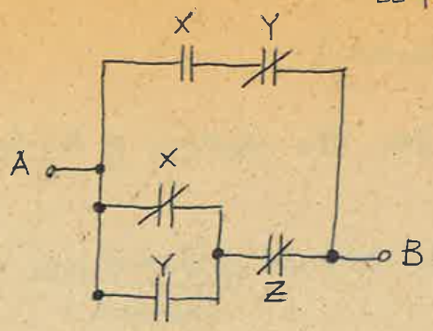
$T_{AB} = 1$

$T_{AC} = 0$

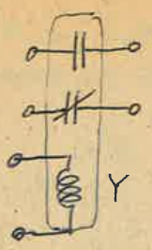
$T_{AB} = x$

$T_{AC} = x'$.

Example :



$T_{AB} = ?$

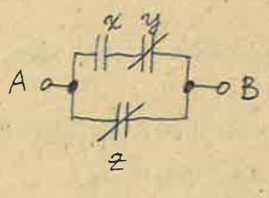


$$T_{AB} = x \cdot y' + (x' + y) \cdot z'$$

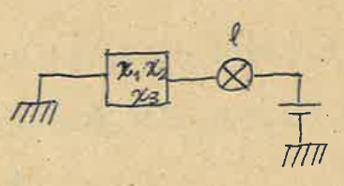
$$= xy' + (xy')' \cdot z'$$

$a + a'b$

$$= xy' + z'$$



Example :



$l = 0$ OFF
$l = 1$ ON

$l = 0$ when $x_1, x_2, x_3 = 0$
$l = 0$ when any two of switches are 1
$l = 1$ when only one of the switches is 1
$l = 1$ when all the switches are 1

x_1	x_2	x_3	l	l'
0	0	0	0	1
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

Canonical sum-of-products (CSP) expression

$$l = x_1' x_2' x_3 + x_1' x_2 x_3' + x_1 x_2' x_3' + x_1 x_2 x_3$$

Canonical product-of sums (CPS) expression.

$$l' = x_1' x_2' x_3' + x_1' x_2 x_3 + x_1 x_2' x_3 + x_1 x_2 x_3'$$

$$(l')' = l = (x_1 + x_2 + x_3)(x_1 + x_2' + x_3')(x_1' + x_2 + x_3')(x_1' + x_2' + x$$

Definition : With an 'n' variable switching function, the form

$m_k = x_1 x_2 x_3 \ldots x_n$ is called a minterm.

Definition : An expression containing only minterms is called canonical sum-of-products (CSP) expression.

→ if the # of and order of the variables are known m_k has a unique meaning.

$n = 3$ abc order : cab

$m_6 = cab' = ab'c$

13101980

Canonical sum of product = canonical minterm expression :

$$f(x_1, x_2 \ldots x_n) = \sum_{i=0}^{2^n - 1} \alpha_i m_i$$

n : # of variables
m : ith minterm
α_i : either 0 or 1

example : $n = 3$

$$f(x_1, x_2, x_3) = \sum_{i=0}^{7} \alpha_i m_i = \alpha_0 m_0 + \cdots + \alpha_7 m$$

where $m_0 = x_1' x_2' x_3'$

$m_1 = x_1' x_2' x_3$

Definition : An expression of the form (for n variable function)
$M_k = (x_1 + x_2 + \cdots + x_n)$ is called a maxterm.
x_i is the ith variable in primed or unprimed form. The expression of the function f in the form :

$$f(x_1, x_2, \cdots, x_n) = \prod_{i=0}^{2^n-1} (\alpha_i + M_i)$$ is called canonical product of sum expression.
"maxterm"

n = # of variables
M_i = ith maxterm
α_i = coefficient of maxterm (either 0 or 1)

example

x_1	x_2	x_3	f	f'	
0	0	0	0	1	←
0	0	1	1	0	
0	1	0	1	0	
0	1	1	0	1	←
1	0	0	0	1	←
1	0	1	0	1	←
1	1	0	1	0	
1	1	1	0	1	←

maxterm expression :
$\alpha_6, \alpha_5, \alpha_1$ are 1 all the others are zero

$$f(x_1, x_2, x_3) = (\alpha_0 + M_0)(\alpha_1 + M_1)(\alpha_2 + M_2)(\alpha_3 + M_3)(\alpha_4 + M_4)(\alpha_5 + M_5)(\alpha_6 + M_6)(\alpha_7 + M_7)$$

$$f' = x_1'x_2'x_3' + x_1'x_2x_3 + x_1x_2'x_3' + x_1x_2'x_3 + x_1x_2x_3$$

$$(f)' = (x_1 + x_2 + x_3)(x_1 + x_2' + x_3')(M_3)(M_2)(M_0)(\alpha_6 + M_6)(\alpha_5 + M_5)(\alpha_1 + M_1)$$
(M_7, M_4)

$1 + m = 1$ drop these terms.

Definition : The maxterm M_k has a unique meaning if a) # of variables
b) order of variables are given
$2^n - 1 \geq k \geq 0$

example : variables order : A B r q
$M_9 = (A + B' + r' + q)$
$9 = 1\ 0\ 0\ 1$

theorem : for an n variable function CSP expression is unique.

Shannon's expansion :
any function with n variables can be written as,

1) $f(x_1, x_2, \cdots, x_n) = x_i f(x_1, \cdots, x_{i-1}, 1, x_{i+1}, \cdots x_n) + x_i' \cdot f(x_1, \cdots, x_{i-1}, 0, x_{i+1}, \cdots, x_n)$

2) $f(x_1, x_2, \cdots, x_n) = [x_i + f(x_1, x_2, x_3, \cdots, x_{i-1}, 0, x_{i+1}, \cdots, x_n)][x_i' + f(x_1, x_2, \cdots, x_{i-1}, 1, x_{i+1}, \cdots, x_n)]$

Successive application of Shannon's expansion (1) and (2) to a Boolean function gives CSP or CPS expression of f respectively.

17-10-1980

Example

$f(A,B,C) = AB + C$
apply Shannon's expansion :

$$f(A,B,C) = A[1 \cdot B + C] + A'[0 \cdot B + C]$$
$$= A\{B[1 \cdot 1 + C] + B'[1 \cdot 0 + C]\} + A'\{B'[0 \cdot 1 + C] + B'[0 \cdot 0 + C]\}$$
$$= AB[1 + C] + AB'[C] + A'B[C] + A'B'[C]$$
$$= AB\{C[1 + 1] + C'[1 + 0]\} + AB'\{C[1] + C'[0]\} + A'B\{C[1] + C'[0]\}$$
$$+ A'B'\{C[1] + C'[0]\}$$
$$= ABC + AB\cancel{C} + A'BC + A'B'C + AB'C$$
(M_7)

Conversion between CSP ↔ CPS

$$f(x_1, x_2, \dots, x_n) = \sum_{i=0}^{2^n-1} \alpha_i M_i \rightarrow \alpha_1, \alpha_3, \alpha_5, \alpha_6, \alpha_7 \rightarrow 1 \text{ for above example}$$

$$f'(x_1, x_2, \dots, x_n) = \sum_{i=0}^{2^n-1} \alpha_i' M_i \quad \text{(take the other minterms)}$$

Generalized De Morgan's law:

$$f(x_1, x_2 \dots x_n) = \left(\sum_{i=0}^{2^n-1} \alpha_i' M_i \right)' = \prod_{i=0}^{2^n-1} (\alpha_i + M_i') = \prod_{i=0}^{2^n-1} (\alpha_i + M_{2^n-1-i})$$

↓ Maxterms

CSP ➡ CPS :

1- Take the other terms which are missing in the CSP expression of f.
2- Subtract the index of missing terms from 2^n-1 to find M_i's

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

CPS ➡ CSP

1- Take the missing ~~terms~~ M_i's in CPS exp.
2- Subtract the index i from 2^n-1 to find ~~m_i~~ in CSP expression.

Example :

$$f(A,B,C,D) = \prod (15, 14, 13, 10, 9, 7, 6, 5, 2, 1)$$

missing terms (maxterms) : 0, 3, 4, 8, 11, 12

corresponding minterms : m_{15} m_{12} m_{11} m_7 m_4 m_3

$$f(A,B,C,D) = \sum (3, 4, 7, 11, 12, 15)$$

Example - Binary full adder -

	A B C_i	S	C_o
0	0 0 0	0	0
1	0 0 1	1	0
2	0 1 0	1	0
3	0 1 1	0	1
4	1 0 0	1	0
5	1 0 1	0	1
6	1 1 0	0	1
7	1 1 1	1	1

$$S = \sum 1, 2, 4, 7$$

$$C_o = \sum 3, 5, 6, 7$$

CSP expressions

$m_6 = ABC_i'$

~~$M_4 = (A + B' + C_i')$~~

$$S = \prod (7, 4, 2, 1)$$

$$C_o \prod (7, 6, 5, 3)$$

CPS expressions

Minimization of Switching functions :

Definition : (Minimization) given $f(x_1, \dots, x_n)$ find another expression $g(x_1, \dots, x_n)$ which is equivalent to $f(x_1 \dots x_n)$ and minimizes some cost criteria.

- Minimal # (# of gates) of terms with minimal # literals in every term in a SP expression

literal : variables in primed or unprimed form.

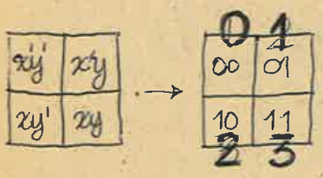
example :

$f = x'y'z' + x'y'z + xy'z + xyz + xyz' + xyz' = \sum(0,1,2,5,6,7)$

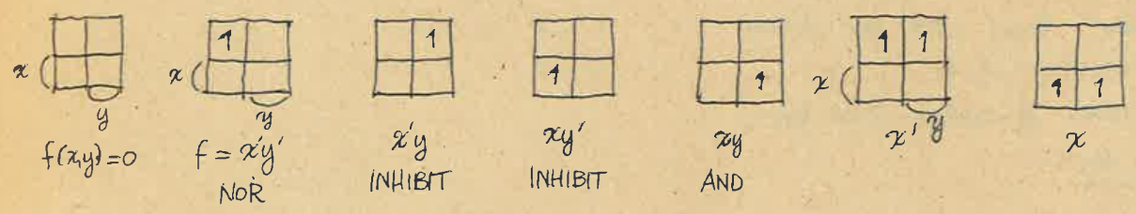
$= x'y'(z+z') + xz(y+y') + yz' \rightarrow SP$

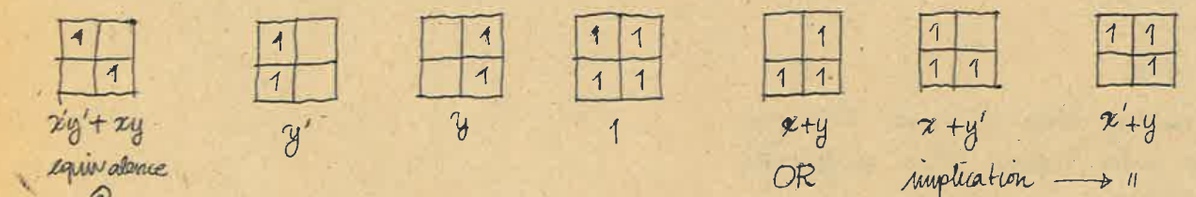
$= x'z' + y'z + yx \rightarrow SP$
(0,2) (1,5) (7,6)

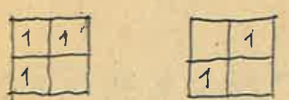
Map method for minimization (Karnaugh map)



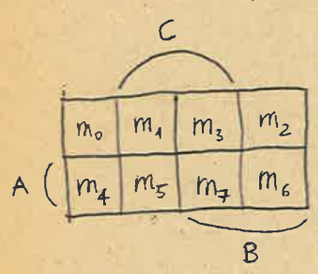
2^{2^n}





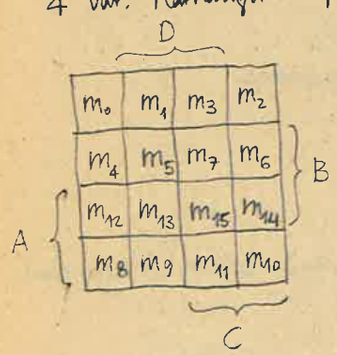


27/01/1980

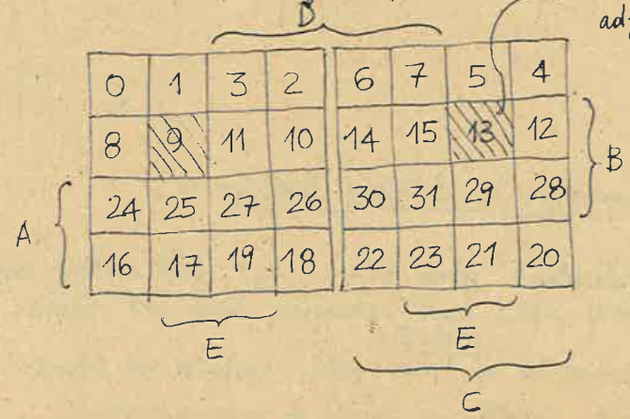


3 var. Karnaugh Map. n variables → 2^n cells.

4 var. Karnaugh Map :



5 variable Karnaugh Map :



Adjacent cells : Two cells which differ only one variable value (or the Hamming distance between them is 1) are said to be adjacent cells.
Each cell in an n variable map has n adjacent cells.

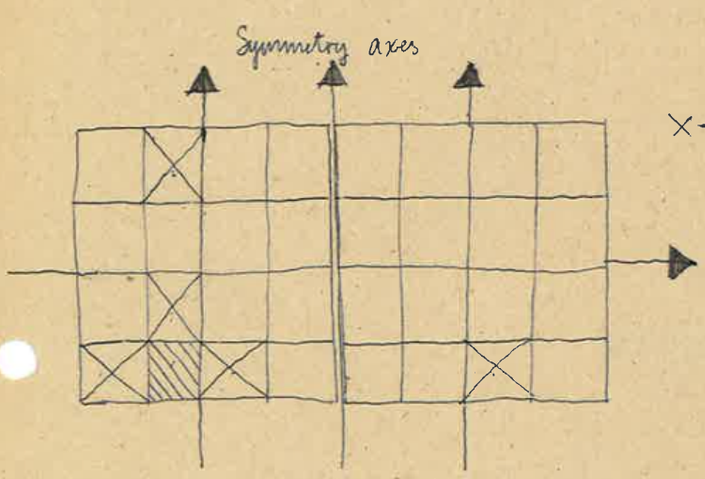
Hamming distance : $A = a_0\ a_1 \dots a_n$
$B = b_0\ b_1 \dots b_n$

$$H = \sum_{i=0}^{n} (a_i \oplus b_i)$$

(Hamming distance)

$A = 011001$
$B = 111000$ → A and B are not adjacent.

$H = 1+0+0+0+1+1 = 3$ → $H>1$

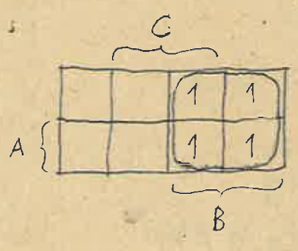


X → adjacent cells of the given cell.

17 → 16, 25, 19, 1, 21

Definition : A collection of 2^m (m integer) cells each adjacent to m cells of the collection is called a SUBCUBE
Each subcube can be expressed by a product term containing $n-m$ literals
n : # of variables
m : order of the subcube
A subcube is said to cover the cells belonging it.

Example :



$2^m = 4$ order $m = 2$ $\quad n = 3$ $\quad m = 2$ $\quad n-m = 1$

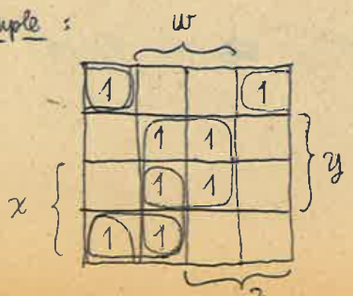
18
40% { MT1 : 25 Kasım
MT2 230 Aralık
15% 6 Ocak Lab final

28101980

To minimize an expression one must cover all the 1-cells (= 1 points) of the expression with the smallest # of subcubes such that each subcube as large as possible

An SP expression from which no term or literal can be deleted without changing its logical value is called an IRREDUNDANT or IRREDUCABLE form which is not unique in general.

Example :

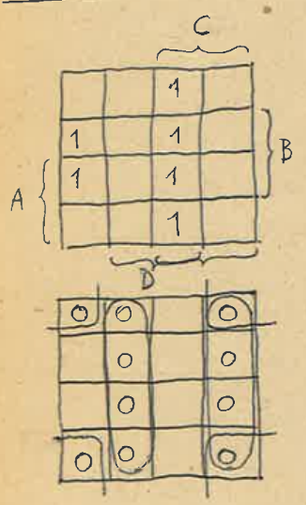


$$f(x,y,z,w) = \sum (0,2,5,7,8,9,13,15)$$

Minimal PS expressions:

- form subcubes from the 0-points of the function
- Select largest and min. # of subcubes to cover the 0-points of f.
- Write corresponding min. SP expression which gives f'.
- complement f' to find min. PS expr.

Example :



$F = CD + BC'D'$

minimal SP expression

$F'(A,B,C,D) = B'D' + C'D + CD'$

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

minimal PS expression

Don't cares

ORIGINS:
- Some input combinations may not appear at the input.
- For some input combinations, related output may not be necessary.

If there are k don't cares, 2^k different switching functions can be generated.

"don't cares can be assumed as 1 or 0"

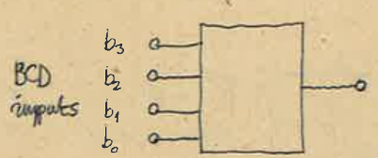
Example :

"BCD prime detector"

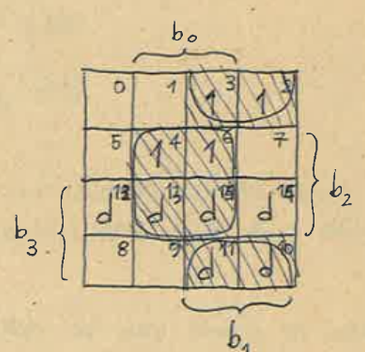
0 : not prime
1 : not prime
then k is prime.
if k = k-1

	b_3	b_2	b_1	b_0	f
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
	1	0	1	0	d
	1	0	1	1	d
	1	1	0	0	d
	1	1	0	1	d
	1	1	1	0	d
	1	1	1	1	d

don't cares

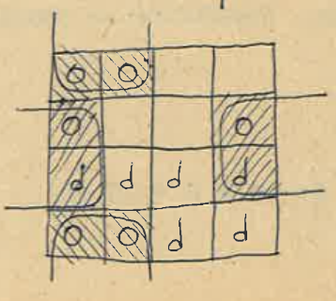


f = 1 if
$b_0 b_1 b_2 b_3$ is prime



$F(b_3, b_2, b_1, b_0) = b_2 b_0 + b_2' b_1$

min. SP expression



$F' = b_2' b_1' + b_0 b_2$

$F = (b_2 + b_1)(b_0 + b_2')$ ➡ min PS exp.

Exercise : BCD → excess 3 translator
EX-3

separately!

Definition: $f(x_1, x_2, \cdots, x_n)$ is said to cover $g(x_1, x_2, \cdots, x_n)$ $\{f \geq g\}$

if $g=1 \Longrightarrow f=1$ or $g \Rightarrow f$

g implies f

g is an implicant of f

example:

	① $A'B'C$	③ $A'BC$	⑥ ABC'	⑦ ABC	$A'C$	AB	BC	$B'C$	f
ABC									
000	1	0	0	0	0	0	0	0	0
001	1	0	0	0	1	0	0	1	1
010	0	0	0	0	0	0	0	0	0
011	0	1	0	0	1	0	1	0	1
100	0	0	0	0	0	0	0	0	0
101	0	0	0	0	0	0	0	1	0
110	0	0	0	0	0	1	0	0	1
111	0	0	0	1	0	1	1	0	1

①, ③, ⑥, ⑦, A'C, AB, BC are implicants of f(A,B,C)
but B'C is not an implicant of f.

Definition: Prime implicant

A prime implicant P of function f is a product term which is covered by f, such that the deletion of any literal from P results in a new product term which isn't covered by f (or is not an implicant of f)

Example: f: above.

Take ① $A'B'C \longrightarrow B'C$

$A'B'C \longrightarrow A'C$

$A'B'C$ is not an prime implicant. (p.i)

take ⑦ $ABC \longrightarrow AB$ is implicant

so ⑦ is not aprime implicant of f.

take $A'C \rightarrow A'$ since A' and C are not implicants of f, therefore $A'C$ is a p.i of f.
$\searrow C$

Similarly AB and BC are p.i of f.

⇨ * Minimal SP expression is obtained by properly selecting p.i. of f.

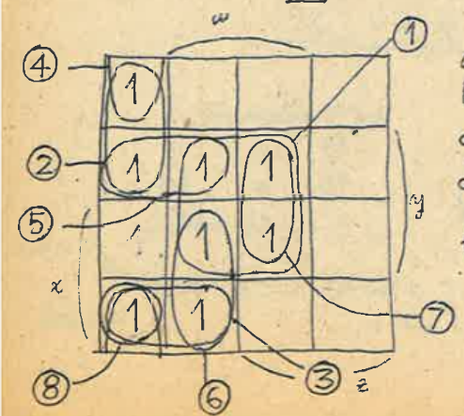
Definition: Essential P.I.

A p.i. P of f is essential p.i if it covers at least one 1-point of f which is not covered by any other p.i

Example

03111980

$f(x,y,z,w) = \sum (0,4,5,7,8,9,13,15)$



a) ① → ⑧ are implicants of f.
b) ⑦ → ⑧ are not prime implicants.
c) ① → ⑥ are prime implicants.
d) ① is essential prime implicant.

$f(x,y,z,w) = ① + ② + ③ = yw + x'w'z' + xy'z'$ unique min. SP expression.

Procedure for minimal SP expression:

④ Find all the prime implicants of the function determining the essential ones.
② Include all the essential prime implicants in the sum.
③ Remove the 1 points of f which are covered by essential prime implicants.
④ For the remaining 1 points select additional prime implicants so that the 1 points of f are covered completely

Quine Mc Cluskey tabulation method

① Find the canonical SP expression of f.
② Arrange all minterms in groups such that all terms in the same group have the same # of 1's in their binary representation or the same # of prime variable.
- Start with the least # of 1's continue with groups of increasing # of 1's. (# of 1's in a minterm called index)

③ Compare another terms of the smallest index group with each term in the successive group whenever possible combine the two by means of the following equality in Boolean algebra:

$$\phi x + \phi' x = x$$

Repeat this by comparing each term in a group of index i with every term in the group of index $i+1$ until the table is finished. The combined term consists of original fixed representation with the different digit is replaced by a (–) → 'dash'

④ The new terms generated will be again compared between them. A new term is generated by combining two terms which
a – Differ by only a single digit and
b – Whose (–)'s are in the same position

The process continues until no further combinations are possible, which gives the set of all prime implicants

Example:

$$f(a,b,c,d) = \sum (0,1,2,3,11,13,15)$$

a'b'c'd' – 0000
a'b'c'd – 0001
a'b'cd' – 0010
a'b'cd – 0011
ab'c'd' – 1011
ab'cd – 1101
abcd – 1111

0	0000 ✓	000– (0,1) ✓	00–– (0,1,2,3) ✓ Same
1	0001 ✓	00–0 (0,2) ✓	00–– (0,2,1,3) ✓
2	0010 ✓	00–0 (1,3) ✓	
3	0011 ✓	001– (2,3) ✓	
11	1011 ✓	–011 (3,11)	b'cd, acd, abd, a'b'
13	1101 ✓	1–11 (11,15)	
15	1111 ✓	11–1 (13,15)	

	0	1	2	3	11	13	15
(3,11) b'cd				X	X		
(11,15) acd					X		X
(13,15) abd						X	X
(0,1,2,3) a'b'	X	X	X	X			

minimal expression:
$f(a,b,c,d) = a'b' + abd \;(+b'cd \text{ or } +acd)$

Example (Don't cares)

$$f(a,b,c,d) = \sum (1,5,10,13,14,15)$$
$$f(a,b,c,d) = \sum_{d} (0,2,7,9)$$

	abcd	(A)			
d(0)	0000	000–	(0,1)		–––01 (1,5,9,13)
1	0001	00–0	(0,2)	d	––01 (1,9,5,13) (E)
d(2)	0010	0–01	(1,5)	✓	–1–1 (5,7,13,15) (F)
5	0101	–001	(1,9)	✓	–1–1 (5,13,7,15)
(d)9	1001	–010 (B)	(2,10)		
10	1010	01–1	(5,7)	✓	
(d)7	0111	–101	(5,13)	✓	
13	1101	1–10	(10,14)	(C)	
14	1110	1–01	(9,13)	✓	
15	1111	–111	(7,15)		
		11–1	(13,15)	✓	
		111–	(14,15)	(D)	

f	1	5	10	13	14	15
A	X					
B			X			
C			X		X	
D					X	X
E	X	X		X		
F		X		X		X

no essential p.i
E dominates A
C dominates B
∴ A and B removed

Col 5 dominates 1
13 " 1
14 " 10
∴ 5, 13, 14 deleted

$F(a,b,c,d) = E \cdot C \cdot (D+F) = c'd + acd' \;(+abc \text{ or } +bd)$
and and or

Procedure of finding min. SP. expression:

① If in a column there is only a single X then the corresponding p.i. is essential include this in the min. SP expr. and remove all these columns and rows from the chart.

② Def: Row K is said to dominate row M if K covers every column covered by M

③ then if row K dominates row M, delete row M from the chart.

Def: Column i is said to dominate column j if i has a (X) in every row in which j has a (X).
The dominating columns can be deleted from the chart.

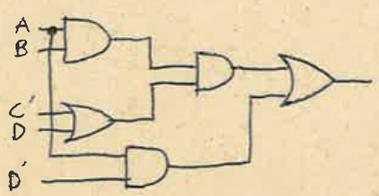
11111980

Exercise $f(A,B,C,D) = \sum(0,1,5,7,8,10,14,15)$ Tabulation method minimization

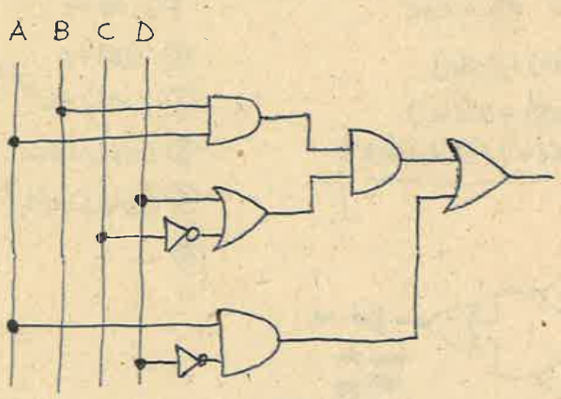
Synthesis of single output combinational cct.

$F = AB(C'+D) + AD'$

i - If complemented input variables are available at the inputs:



ii - if complemented (= primed) variables are not available.



① The functions of the following form

$F_1 = AB$ $F_2 = A+B$

are called first-order expressions Because they can be implemented by a single gate cct. The obtained gating cct. is called Single level (= one level) cct. (if complemented input variables are available)

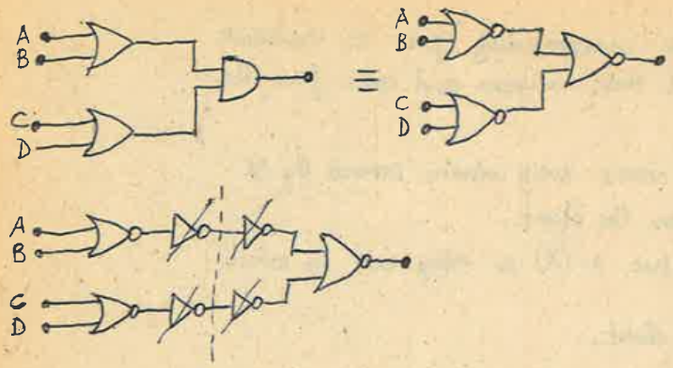
② The functions: $F_3 = AB+C+D$ or $F_4 = (A+B)CD$ are called second order functions. The gating cct. obtained from these functions are two-level cct's. (If primed variables are available)

③ The functions $F_5 = (AB+C+D)E$, $F_6 = (A+B)CD+E$ are called third-order functions. Obtained cct's are three level circuits.

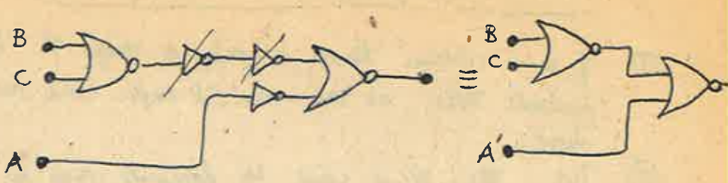
➡ NAND/NOR
NOR/NAND realizations are not possible for any switching function.

NAND or NOR realizations of functions :

Ex 1



Ex 2 $F = A(B+C)$



Rule for PS → NOR realization
SP → NAND

① Bracket each factor.
② Complement any literal which appears by itself in PS SP
③ If there is only one term such as

$(a+b+\ldots+n)$
$(a.b.\ldots.n)$ complement this term.

④ Replace (+) and (·) operations by NOR operation (↓)
NAND (/)

↑ means NOR

Ex: $F = (A+B)(C+D)$

① $(A+B)(C+D)$
② —
③ —
④ $(A\downarrow B)\downarrow(A\downarrow B)$

Ex: $F = A+B+C+D$

① $(A+B+C+D)$
② —
③ $(A+B+C+D)'$
④ $(A\downarrow B\downarrow C\downarrow D)'$

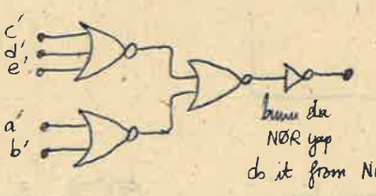
Rules for SP → NOR

① Bracket each term
② Complement each literals of each product unless the product consisted of only one literal.
③ Replace +, · by ↓ (NOR)
④ Complement entire function
⑤ If SP form has only one term don't complement it.

(NOTE: Does not give a two-level ckt in general)

$F = ab + cde$

① $(ab)+(cde)$
② $(a'b')+(c'd'e')$
③ $(a'\downarrow b')\downarrow(c'\downarrow d'\downarrow e')$
④ [ ]'



bcuz the NOR gate do it from NOR.

$F = ab + c$

① $(ab)+c$ ← don't complement.
② $(a'b')+c$
③ $(a'\downarrow b')\downarrow c$
④ $[(a'\downarrow b')\downarrow c]'$
⑤ —

Rules for PS → NAND

① Bracket each term.
② Complement each literals of each factor unless the factor consisted of only one literal
③ Replace +, · by / (NAND)
④ Complement entire function
⑤ If PS form has only one factor don't complement it.

(NOTE: Does not give a two level ckt in general)

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

① $(A+B)(C+D)$
② $(A'+B')(C'+D')$
③ $(A'/B')/(C'/D')$
④ $[(A'/B')/(C'/D')]'$
⑤ —

$F = A(B+C)$

① $(A)(B+C)$
② $(A)(B'+C')$
③ $(A)/(B'/C')$
④ $[(A)/(B'/C')]'$
⑤ —

Multiple Output cct's and Minimization :

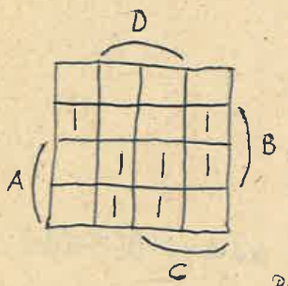
Example :

$y_1(A,B,C,D) = \sum (4,6,13,14,15,11,9)$

$y_2(A,B,C,D) = \sum (1,3,5,7,9,13,11,10,15)$

$y_3(A,B,C,D) = \sum (1,3,4,5,6,7,10,11,14,15)$

y_1 :



y_2 :

y_3 :

$y_1 = AD + A'BD' + \begin{cases} BCD' \\ ABC \end{cases}$

$y_2 = D + AB'C$

$y_3 = A'D + AC + A'B$

Complemented Variables are available, using AND/OR gating cct. :

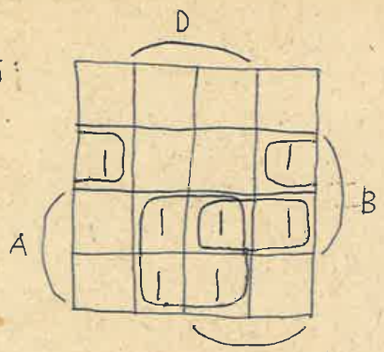
y_1 = 4 gates + 11 gate inputs.

y_2 = 2 gates + 5 gate inputs

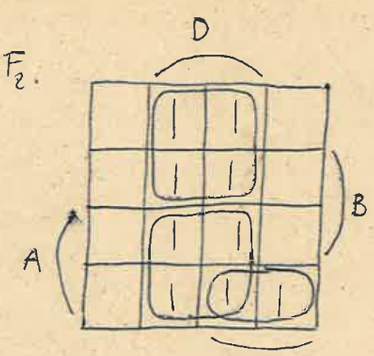
y_3 = 4 gates + 9 gate inputs

for $y_1\ y_2\ y_3$: 10 gates + 25 gate inputs.

F_1 :

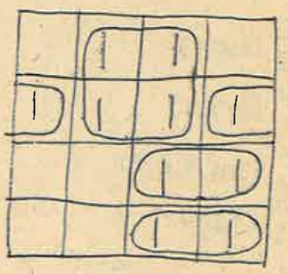


F_2 :



17111980

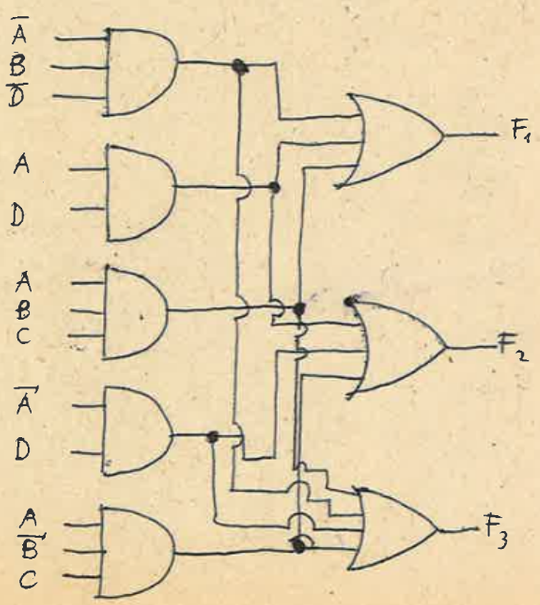
F_3



$F_1 = AD + \bar{A}B\bar{D} + ABC$

$F_2 = AD + \bar{A}D + A\bar{B}C$

$F_3 = \bar{A}D + \bar{A}B\bar{D} + ABC + A\bar{B}C$



8 GATES + 23 INPUTS

Tabulation Method for Multiple output cct's

$F_1(a,b,c) = \sum(0,4,7)$

$F_2(a,b,c) = \sum(0,3,4)$

$F_3(a,b,c) = \sum(3,4,5,6,7)$

TAG

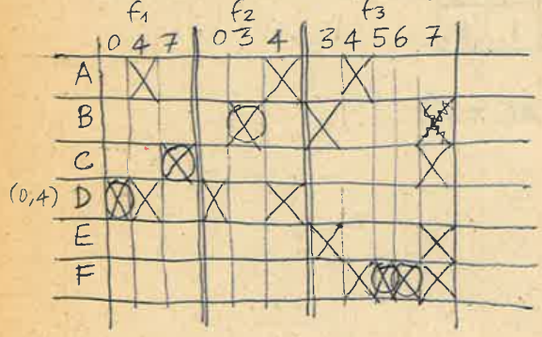
	a b c	f_1	f_2	f_3
0	000 ✓	1	1	0
(A) * 4	100	1	1	1
(B) * 3	011	0	1	1
5	101 ✓	0	0	1
6	110 ✓	0	0	1
(C) * 7	111	1	0	1

performing an AND operation bit by bit between the TAG's of 0,4

If TAG of new term is equal to previous ones put a '✓' near old terms. (delete them)

(0,4)	−00	110 *(D)
(4,5)	10− ✓	001
(4,6)	1−0 ✓	001
(4,5,6,7)	1−−	001 *(F)
(4,6,5,7)	1−−	001
(3,7)	−11	001 *(E)
(5,7)	1−1 ✓	001
(6,7)	11− ✓	001

unchecked terms are multiple output prime implicants (mopi's).



To cover f_1 you have to take D then A or D and C D(A+D)C

f_2: DB(A+D)

f_3: (B+E)(A+F)F(C+E+F)

to cover f_1, f_2, f_3 : D(A+D)C · DB(A+D)(B+E)(A+F)F(C+E+F)

DCBF ← RESULT

F_1: DC = b'c' + abc

F_2: DB = b'c' + a'bc

F_3: FB = a + a'bc

Example:

F_1: $\sum(0,4,7)$

F_2: $\sum(0,3,4)$

F_3: $\sum(3,4,5,6,7)$

F_4: $\sum(1,2)$

MT I

Map Method to minimize multiple output ckt.

Procedure

i- Find all p.i's of functions.
ii- Find all possible products of functions, find their p.i's.
iii- Label all p.i which are distinct in the table. (Starting from bottom to top)
iv- Form p.i table.

Exercise: Don't cares → 1 points

$F_1(A,B,C) = \sum(0,4,7)$

$F_2(A,B,C) = \sum(0,3,4)$

$F_3(A,B,C) = \sum(3,4,5,6,7)$

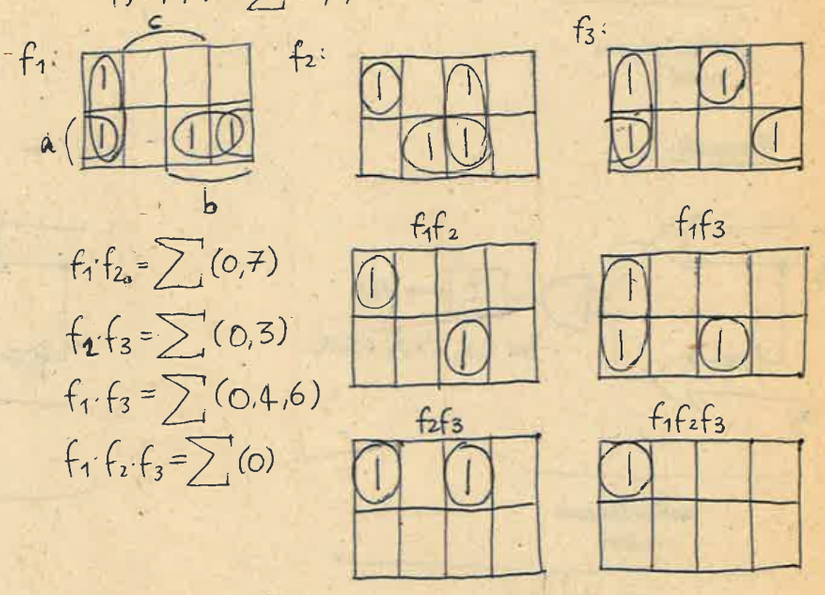
$d = \sum(1,2)$

answer: 7 gates 19 gates.

$f_1(a,b,c) = \sum(0,4,6,7)$

$f_2(a,b,c) = \sum(0,3,5,7)$

$f_3(a,b,c) = \sum(0,3,4,6)$



$f_1 \cdot f_2 = \sum(0,7)$

$f_1 \cdot f_3 = \sum(0,3)$

$f_1 \cdot f_3 = \sum(0,4,6)$

$f_1 \cdot f_2 \cdot f_3 = \sum(0)$

f_1: $b\bar{c}$, $a\bar{c}$, (6,7) ab (H)

f_2: $\bar{a}b\bar{c}$, bc (F), ac (G)

f_3: $\bar{a}bc$, $b\bar{c}$, $a\bar{c}$

f_1f_2: $\bar{a}b\bar{c}$, abc (E)

f_1f_3: $b\bar{c}$ (C), $a\bar{c}$ (D)

f_2f_3: $\bar{a}b\bar{c}$, $\bar{a}bc$ (B)

$f_1f_2f_3$: $\bar{a}b\bar{c}$ (A)

	f_1: 0 4 6 7	f_2: 0 3 5 7	f_3: 0 3 4 6
f_1: H (6,7)	X X		
f_2: F (3,7)		X X	
G (5,7)		⊗ X	
f_1f_2 E (7)	X	X	
f_1f_3 C (0,4)	X X		X X X (X)
D (4,6)	X X		
f_2f_3 B (3)		X	(X)
$f_1f_2f_3$ (0) A	X	(X)	X

f_1: ~~(A+C)(C+D)(H+D)~~(H+E)

f_2: ~~A(F+B)G(E+G+E)~~

f_3: ~~(C+A)~~B~~(C+D)~~D

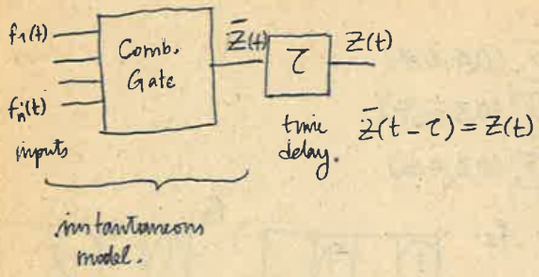
AGBD(H+E) preferring H

f_1: ADH f_3: BDA

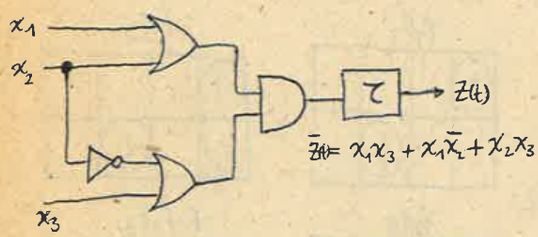
f_2: ABG

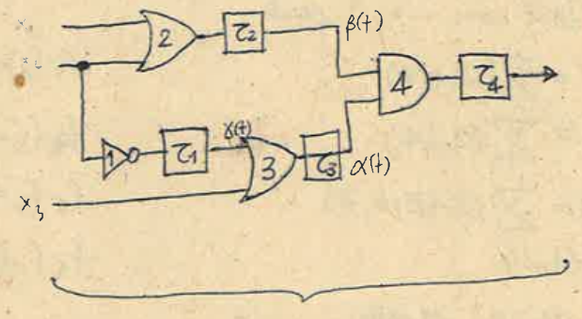
25/11/1980

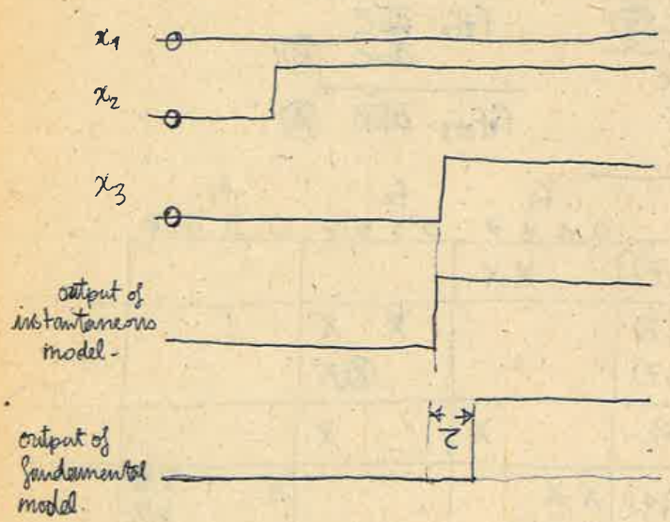
Combinational Hazards :

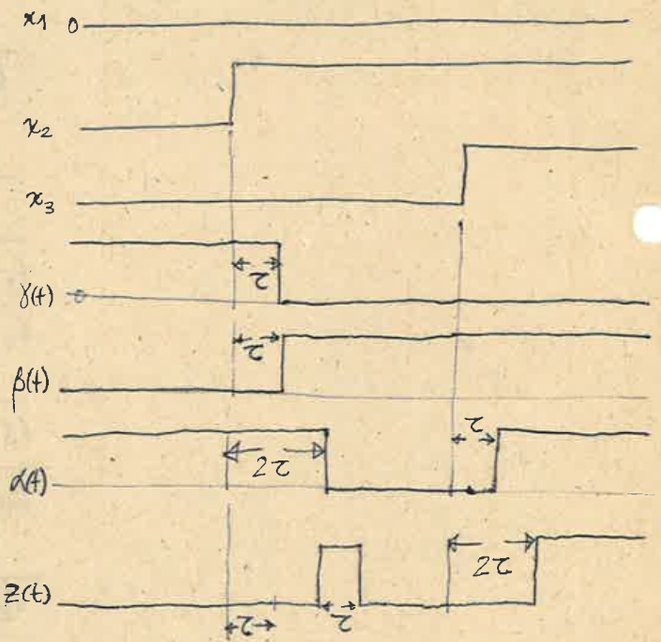


Example :









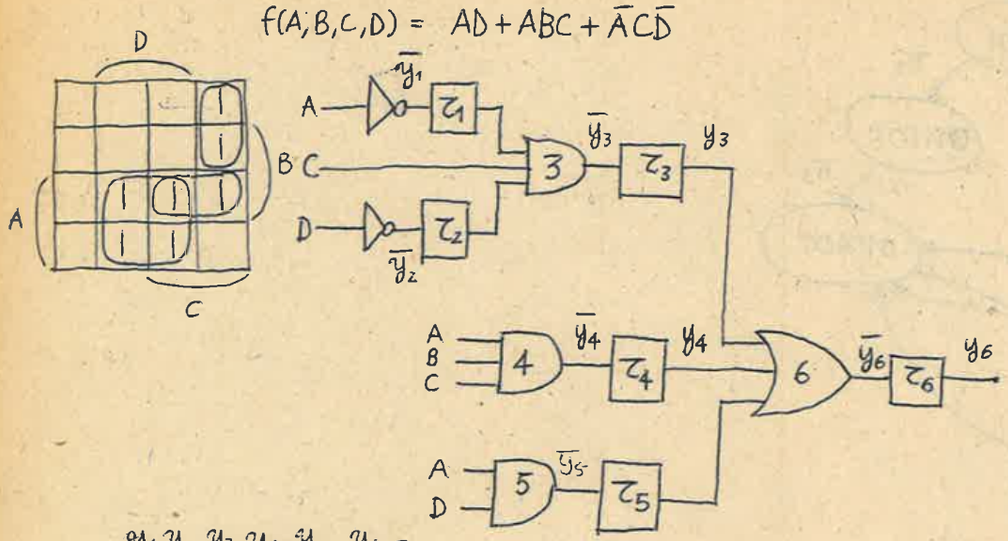
Combinational cct. hazards

- steady state hazards (comp. errors)
- transitory hazards
  - static hazards
  - dynamic hazards.

Static hazards (Due to single input changes only)

Example : $f(A,B,C,D) = \sum (2,6,9,11,13,14,15)$

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



$y_1\ y_2\ y_3\ y_4\ y_5\ y_6$ → State of the network

$\bar{y}_1\bar{y}_2\ \bar{y}_3\ \bar{y}_4\ \bar{y}_5\ \bar{y}_6$ → Next state of network

next state equations :

$\bar{y}_1 = A'$ $\quad \bar{y}_4 = ABC$

$\bar{y}_2 = D'$ $\quad \bar{y}_5 = AD$

$\bar{y}_3 = C y_1 y_2$ $\quad \bar{y}_6 = y_3 + y_4 + y_5$

ABCD : 0110

State of ckt :

$y_1 = 1$ $\quad y_3 = 1$ $\quad y_5 = 0$ } Present state

$y_2 = 1$ $\quad y_4 = 0$ $\quad y_6 = 1$ } 111001

ABCD → 1110

next states :

$\bar{y}_1 = 0$ $\quad \bar{y}_4 = 1$

$\bar{y}_2 = 1$ $\quad \bar{y}_5 = 0$

$\bar{y}_3 = 1$ $\quad \bar{y}_6 = 1$

present state : 111001

next state : 011101

this is also 0 → 1

this variable is trying to change from 1 to 0

$\Rightarrow$ if $\bar{y}_i \neq y_i$ then y_i is trying to change its state

ABCD = 0110 111001

ABCD = 1110 111001

011101

if $z_1 < z_4$ → 011001 PS

010101 NS

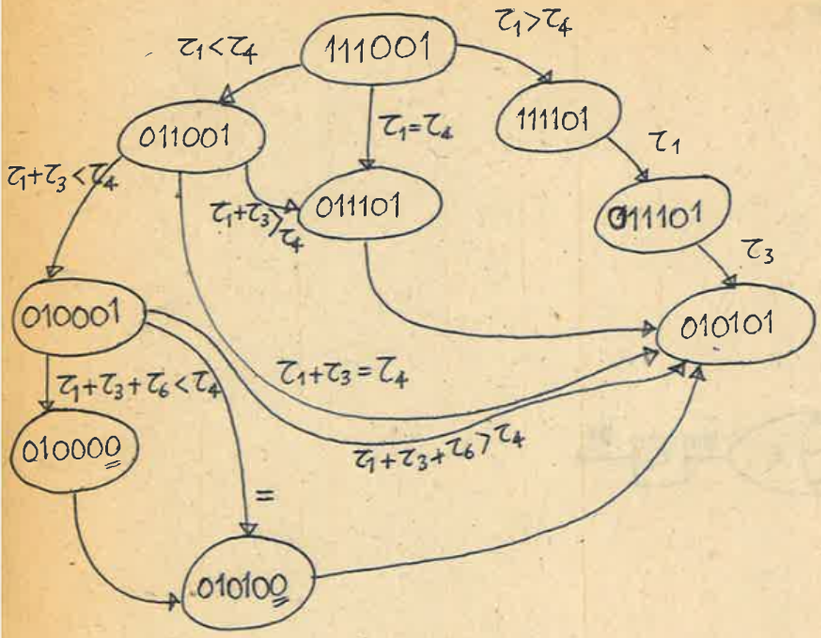
if $z_1 = z_4$ → 011101

010101

if $z_4 < z_1$ → 111101

010101

ABCD : 0110
ABCD : 1110



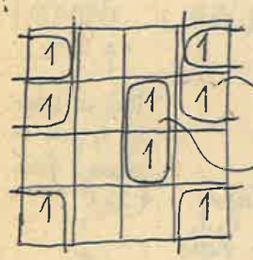
Definition: If x^i and x^j are adjacent input symbols and a switching function has the same value for both input symbols;

$$f(x^i) = f(x^j) = 1 \text{ (or } 0)$$

then a network which realizes that function f contains a STATIC hazard if a "0" (or "1") can appear temporarily on its output line when the input symbol is changed from x^i to x^j, or vice versa.

To eliminate static hazards due to single input changes: every pair of adjacent minterms must be covered by at least one selected prime implicant.

example:



$$F_1 = \bar{A}\bar{D} + \bar{B}\bar{D} + BCD + \bar{A}BC$$

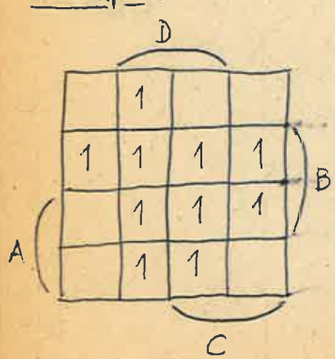
m_6 : 0110
m_7 : 0111

these two adjacent terms are not included in a prime implicant. Because of this a change of $m_6 \leftrightarrow m_7$ may give a static hazard. We must include them with addition of a redundant p.i

If multiple input changes are allowed:

i. Generalization of static hazard.
ii. Function hazard.

Example:



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

No static hazard due to single input changes.

But if there will be a change between $m_{15} \leftrightarrow m_5$ (two inputs are changed) There may be hazard.

We must include another redundant gate to eliminate this hazard: BD

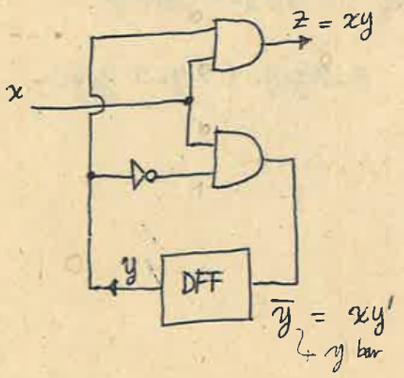
Definition : If x^j and x^i are different input symbols a static hazard exists if $f(x^i) = f(x^j) = 1$ (or 0) and $f = (0)$ or (1) can appear temporarily when input symbol is changed x^i to x^j. This hazard may be eliminated by including in the SP expression of function f a prime implicant that covers both x^i and x^j, if such p.i exists.

SEQUENTIAL CIRCUITS

Flip-flops:

- RS flip flop. (RS-FF)
- Triggered flip-flop (T-FF)
- JK flip flop
- Delay flip flop (D-FF)
- Gated latch flip flop (GLFF)

09121980



$\bar{y}' = \psi(x^j, y^k)$

$\psi \Rightarrow \bar{y} = x \cdot y'$

$\psi(x^0, y^0) = y^0$
$\psi(x^0, y^1) = y^0$
$\psi(x^1, y^0) = y^1$
$\psi(x^1, y^1) = y^0$

$\xi \Rightarrow z = x \cdot y$

$\xi(x^0, y^0) = z^0$
$\xi(x^0, y^1) = z^0$
$\xi(x^1, y^0) = z^0$
$\xi(x^1, y^1) = z^1$

ψ:

x	y	$\bar{y}$
0	0	0
0	1	0
1	0	1
1	1	0

state transition table

ξ

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

output table

x	y	$\bar{y}$	z
0	0	0	0
0	1	0	0
1	0	1	0
1	1	1	1

State table

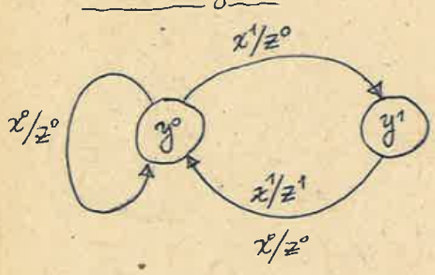
$\equiv$

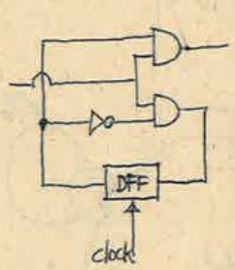
y	x=0	x=1
0	0/0	1/0
1	0/0	0/1

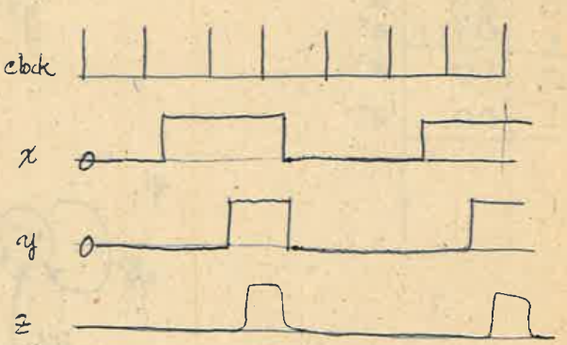
$\equiv$

y	x^0	x^1
y_0	y^0/z^0	y^1/z^0
y^1	y^0/z^0	y^0/z^1

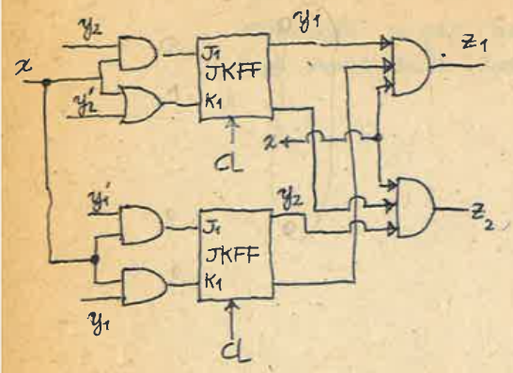
State diagram:







Example : MoD 3 counter



given $X = \{x^0, x^1\}$ input alphabet

$Y = \{y^0, y^1, y^2, y^3\}$ state alphabet

$Z = \{z_0, z^1, z^2, z^3\}$

Ψ : $J_1 = y_2 x$ | $J_2 = y_1' x$

$K_1 = y_2' + x$ | $K_2 = y_1 x$

General next state equation of JK FF $\bar{y} = Jy' + K'y$

next state equations

$$\bar{y}_1 = J_1 y_1' + K_1' y_1 = (x y_2) y_1' + (x + y_2')' y_1$$

$$= x y_1' y_2 + x' y_1 y_2$$

$$\bar{y}_2 = (x y_1') y_2' + (\cancel{x y_1'})' y_2$$

$$= x y_1' y_2' + x' y_2 + y_1' y_2$$

Ψ : state transition table

x	y_1	y_2	$\bar{y}_1$	$\bar{y}_2$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	1	1
1	1	0	0	0
1	1	1	0	0

$\Rightarrow$ Ψ

x	y	$\bar{y}$
x^0	y^0	y^0
x^0	y^1	y^1
x^0	y^2	y^0
x^0	y^3	y^3
x^1	y^0	y^1
x^1	y^1	y^3
x^1	y^2	y^0
x^1	y^3	y^0

$Y = \{y^0, y^1, y^3\}$

ζ : $z_1 = x y_1 y_2'$

$z_2 = x y_1' y_2$

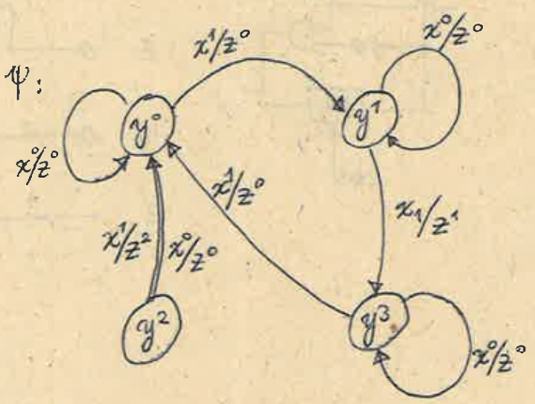
$Z = \{z_0, z_1\}$

x	y_1	y_2	z_1	z_2	
0	0	0	0	0	z^0
0	0	1	0	0	z^0
0	1	0	0	0	z^0
0	1	1	0	0	z^0
1	0	0	0	0	z^0
1	0	1	0	1	z^1
1	1	0	1	0	z^2
1	1	1	0	0	z^0

PS	NS x^0	NS x^1	Out x^0	Out x^1
y^0	y^0	y^1	z^0	z^0
y^1	y^1	y^3	z^0	z^1
y^2	y^0	y^0	z^0	z^2
y^3	y^3	y^0	z^0	z^0

State table

Ψ :



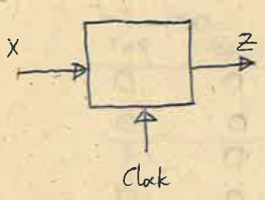
Design of Synchronous Sequential Circuits

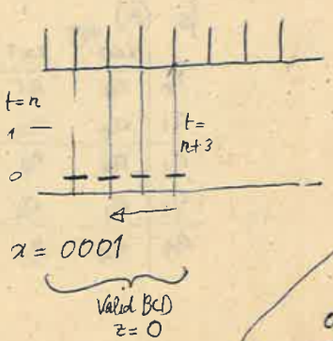
Design Procedure:

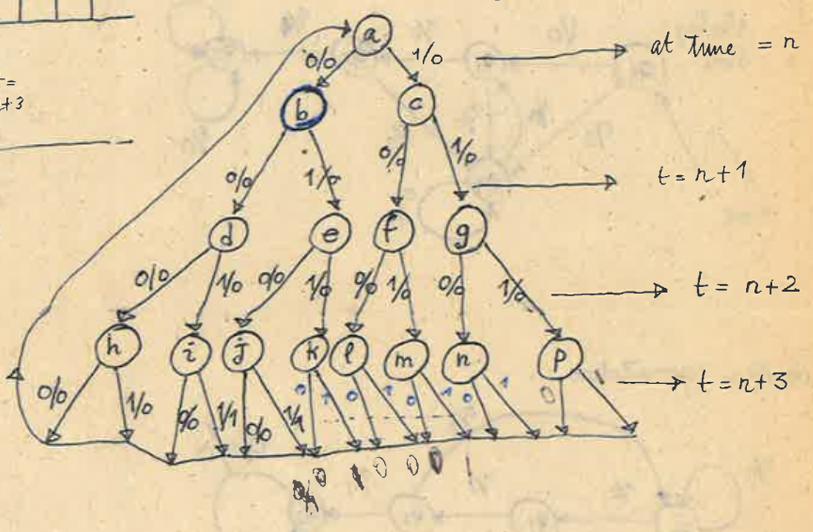
i. Form a model from the word description of the problem. (Find state diagram)(Stable)
ii. Simplify the state table
iii. Make a state assignment
iv. Decide the type of FF's and find input equations.
v. Design the remaining necessary combinational cct.'s

Example: Single input – single output (x/z)

The circuit determines whether the input bits at four successive times constitute a valid BCD digit representation, when $z=0$ at the fourth clock time indicates that previous three inputs and present bit constitute a valid BCD representation. When $z=1$ it indicates an invalid BCD representation. z is to be zero at all other times. Least significant bit of the code words appears first in time.

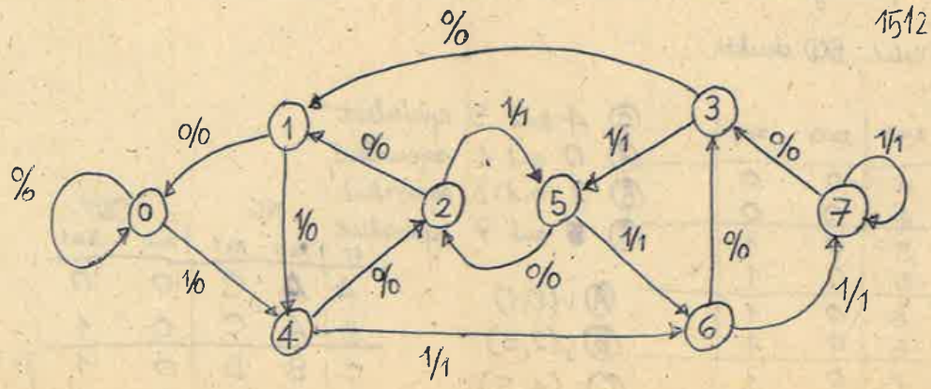






NS/output

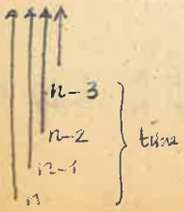
PS	x = 0	x = 1
a	b,0	c,0
b	d,0	e,0
c	f,0	g,0
d	h,0	i,0
e	j,0	k,0
f	l,0	m,0
g	n,0	p,0
h	a,0	a,0
i	a,0	a,1
j	a,0	a,1
⋮	⋮	⋮
p	a,0	a,1



input
15121980

n	n-1	n-2	n-3
0	0	0	0
1	0	0	0
0	0	0	1

d000 → 0
d001 → 1
⋮
d111 → 7

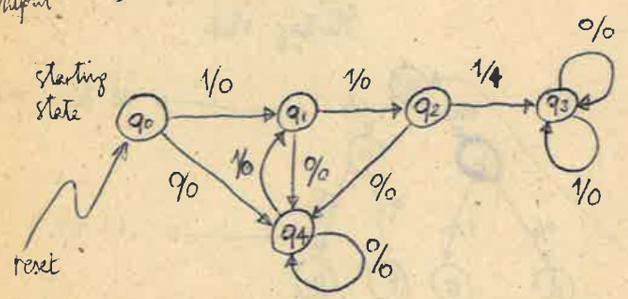


PS	I/Out NS x=0	x=1
0	0,0	4,0
1	0,0	4,0.
2	1,0	5,1
3	1,0	5,1
4	2,0	6,1
5	2,0	6,1
6	3,0	7,1
7	3,0	7,1
~~8~~		
~~9~~		

X Example : A clocked sequential ckt. will be designed which will have an output of 1 only at the clock time coinciding with the third of a sequence of three 1's. The ckt. will be provided a separate reset mechanism to place it in a starting state say q_1.

Input : 00110101101111010 11...
Output 00000000000100...

(A)

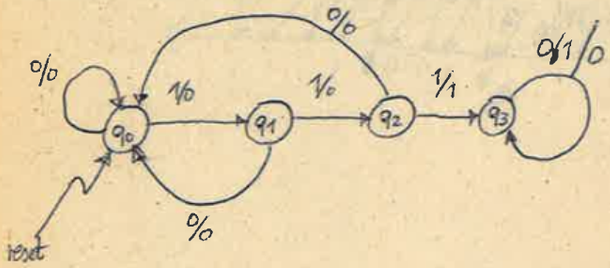


for (A)

	NS x=0	x=1	OUT x=0	x=1
q_0	q_4	q_1	0	0
q_1	q_4	q_1	0	0
q_2	q_4	q_3	0	1
q_3	q_3	q_3	0	0
q_4	q_4	q_1	0	0

Another representation :

(B)



STEP 2 State minimization

Definition : Equivalent states : Two states p and q of a machine are said to be equivalent if for every input symbols signals applied to machine starting from p and from q generates the same output symbols and results in equivalent next states, state

Example : Look at Valid BCD checker.

PS	NS x=0	x=1	OUT x=0	x=1
0	0	4	0	0
1	0	4	0	0
2	1	5	0	1
3	1	5	0	1
4	2	6	0	1
5	2	6	0	1
6	3	7	0	1
7	3	7	0	1

(C) 4 and 5 equivalent
(A) 0 and 1 equivalent
(B) 2 and 3 equivalent
(D) 6 and 7 equivalent

(A);(0,1)
(B);(2,3)
(C);(4,5)
(D),(6,7)

PS	NS x=0	x=1	OUT x=0	x=1	
A	A	C	0	0	
B	A	C	0	1	
C	B	D	0	1	States C,D are equivalent too. (C,D) : a
D	B	D	0	1	

16121980

Partitioning algorithm: (only for completely specified m/c)

i. Partition the set of all states into sets such that all members of a set have identical output rows in the state table.

$S1 = \{a, b, c, d, e, f, g, h, l\}$

$S2 = \{i, j, k, m, n, p\}$

ii. Under each state for each input symbol record the number of the set of which the next state is a member.

$S1 = \{a, b, c, d, e, f, g, h, l\}$
1,1 1,1 1,1 1,2 2,2 1,2 2,2 1,1 1,1

$S2 = \{i, j, k, m, n, p\}$
1,1 1,1 1,1 1,1 1,1 1,1

iii. Divide existing state sets so that all members of a new state set possess the same subscripts. When no new sets are formed the algorithm terminates, if new sets are formed repeat step ii.

$S1 = \{a, b, c, h, l\}$
1,1 2,3 2,3 1,1 1,1

$S2 = \{d, f\}$
1,4 1,4

$S3 = \{e, g\}$
4,4 4,4

$S4 = \{i, j, k, m, n, p\}$
1,1 1,1 1,1 1,1 1,1 1,1

Now repeat step ii; step iii:

$S1 = \{a, h, l\}$
2,2 1,1 1,1

$S2 = \{b, c\}$
3,4 3,4

$S3 = \{d, f\}$

$S4 = \{e, g\}$

$S5 = \{i, j, k, m, n, p\}$

Again repeat:

A → {a}
B → {h, l}
C → {b, c}
D → {d, f}
E → {e, g}
F → {i, j, k, m, n, p}

we have started with a m/c which has 15 states and arrived a m/c which has 6 states. (Performing the same work)

To the top of the page.

	NS x=0	NS x=1	OUT x=0	OUT x=1
A	C	C	0	0
B	A	A	0	0
C	D	E	0	0
D	B	F	0	0
E	F	F	0	0
F	A	A	0	1

minimal state equivalent of valid BCD checker.

* There is no minimal state for incompletely specified m/c.

STEP 3: STATE ASSIGNMENT

if m = # of states, to realize this m/c, let p = min # of required FF's.

$$p = \lceil \log_2 m \rceil \qquad \lceil a \rceil = \text{smallest integer equal to or greater than } a.$$

If $2^{p-1} \leq m \leq 2^p$ satisfied

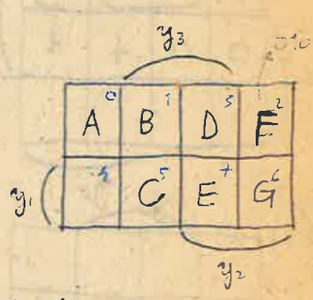
then $\dfrac{2^p!}{(2^p - m)!}$ different assignments can be given to the states

State Assignment Rules

Rule I ⓐ Check for rows of the state table which have identical next state entries in every column. Such rows should be given adjacent assignments. If possible next state entries in these rows should be given adjacent assignments according to rule II

ⓑ Check for the rows of the ST which have the same next state entries but in different column order such rows should be given adjacent assignments if the next state entries can be given adjacent assignments.

	x=0	x=1	x=0	x=1
A	B	C	0	0
B	D	E	0	0
C	E	D	0	0
D	F	G	0	0
E	G	F	0	0
F	A	A	1	0
G	A	A	0	1



→ Rule Ia: assign adjacent codes for F and G.

→ Rule Ib: DE must be adjacent if FG adjacent.

→ Rule Ib: BC must be adjacent if if DE adjacent.

DONALD L. DIETMEYER / LOGIC DESIGN OF DIGITAL SYSTEMS

22/2/1980

Rule © : Rows with identical next state entries in some, but not at all, columns should be given adjacent assignments with rows having more identical columns having the higher priority.

	×	×	×	×
A	A	B	M	D
D	A	B	M	M

Rule II : Next state entries for a given row should be given adjacent assignments.

Rule III : Assignments should be done so as to simplify the output expressions.

STEP 4 : FF INPUT EQUATIONS

Without considering the state assignment rules;

let ;

A = 000
B = 100
C = 010
D = 101
E = 011
F = 001

PS	NS x=0	NS x=1	OUT x=0	OUT x=1
A	C	C	0	0
B	A	A	0	0
C	D	E	0	0
D	B	F	0	0
E	F	F	0	0
F	A	A	0	1

PS $y_3y_2y_1$	NS $Y_3Y_2Y_1$ x=0	NS $Y_3Y_2Y_1$ x=1	OUT x=0	OUT x=1
000	010	010	0	0
100	000	000	0	0
010	101	011	0	0
101	100	001	0	0
011	001	001	0	0
001	000	000	0	1
110	NOT USED			
111				

encoded state table

$d = \sum(6, 7, 14, 15)$

$\bar{y}_1 = x'y_1y_3 + x'y_2y_3'$

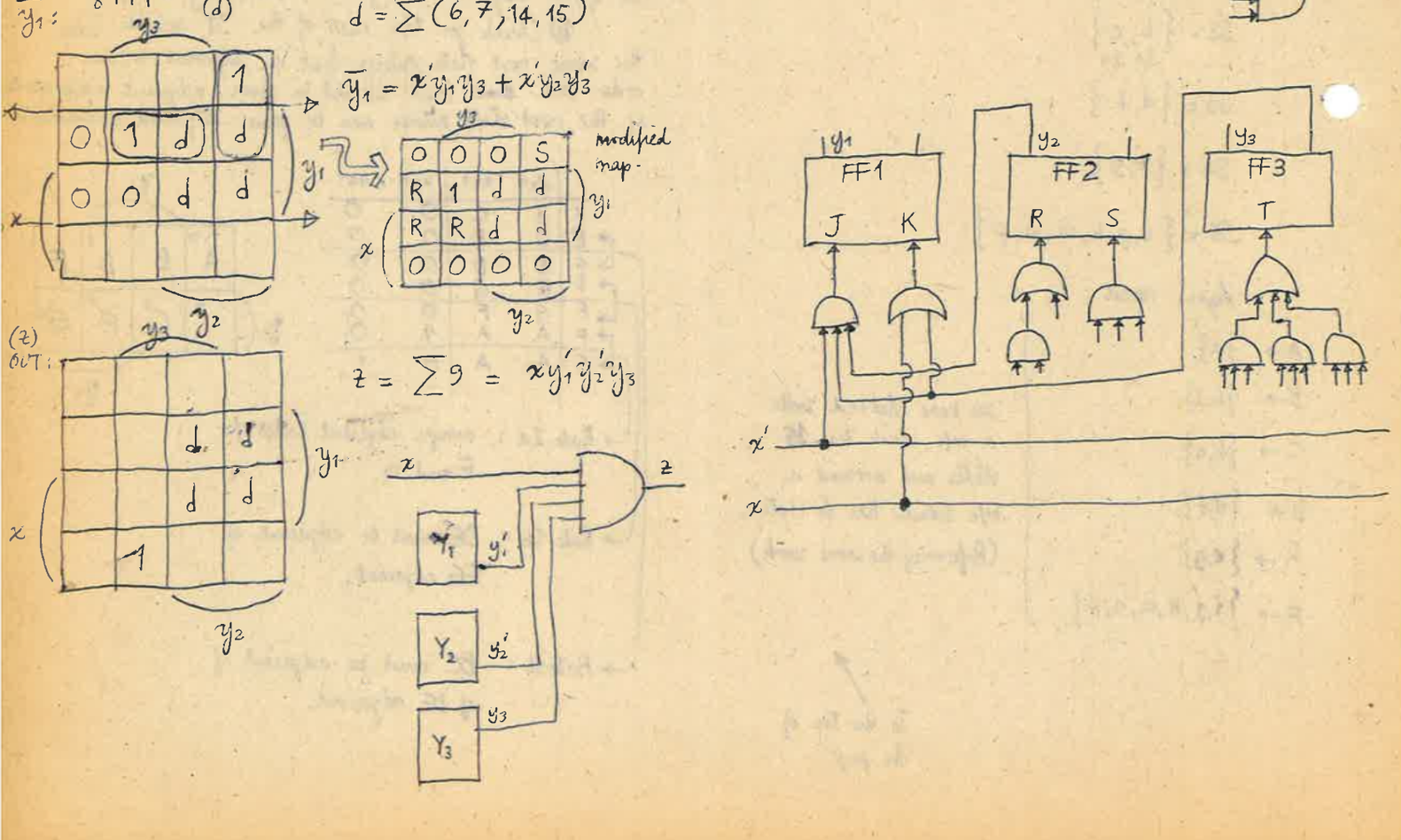
$z = \sum 9 = xy_1'y_2'y_3$

3 FF:

y_1 : JK FF
y_2 : SR FF
y_3 : T FF

$y \to \bar{y}$	J	K	S	R	T	G	L
0→0	0	d	0	d	0	0	d
0→1	1	d	1	0	1	1	1
1→0	d	1	0	1	1	1	0
1→1	d	0	d	0	0	0	d

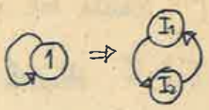
$J_1 = x'y_2y_3'$
$K_1 = x + y_3'$
$S_2 = y_1'y_2'y_3'$
$R_2 = y_3 + x'y_2$
$T_3 = y_2y_3' + xy_2'y_3 + y_1'y_2'y_3$
$z = xy_1'y_2'y_3$



STATE REDUCTION OF INCOMPLETELY SPECIFIED SEQUENTIAL SYSTEMS : 29121980

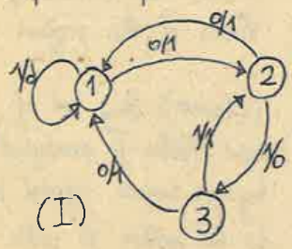
Definition : A system where not all the output sequences and/or next states are specified for given input sequences.

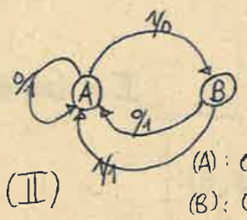
	NS x=0	NS x=1	OUT x=0	OUT x=1
I_1	2	I_2	1	0
I_2	2	I_1	1	1
2	I_1	3	1	0
3	I_1	2	1	1



Apply partitioning method $(I_1,2)$, $(I_2,3)$
(A) (B)

PS	NS x=0	NS x=1	OUT x=0	OUT x=1
A	A	B	1	0
B	A	A	1	1





applicable
(A) : 00100
(B) : 01010
not applicable

Starting state in m/c I say is 1
Starting state in m/c II is (either I_1 or I_2) say I_1

apply X : 00110 to both of m/c and check results.

X = 001100
I = Z: 11dd11
II = Z: 1101 11

Definition : Applicable input sequence I_S.
i) An applicable input sequence to a system S with S's starting state.
If and only if I_S leads to a specified next state after each input of the sequence except possibly the last.

Definition : State t of a system T is said to cover state s of a system S if and only if for all I_S applied to the system S starting in state s and to system T starting in state t, the outputs of T are the same as the outputs of S whenever the latter is specified,

I_S : 010110
Output T : 111110 — starting state t
" S : 11d1d0 — starting state s.
then t covers s " t>s "

Definition : System T is said to cover system S iff for every state s in S there exist a state t in T such that t>s

State reduction : given S find system T having smaller # of states than S and satisfy T>S

30121980

S → T
T > S

1. Computation of maximal compatible sets.
2. Finding the reduced state m/c using the compatible sets.

Definition : Compatible output sequences : Two output sequences are said to be compatible if the corresponding outputs of two sequences are equal whenever they are both specified.

Let Z_S : output sequence of length r, obtained by an applicable input sequence I_S of length r, starting state s.

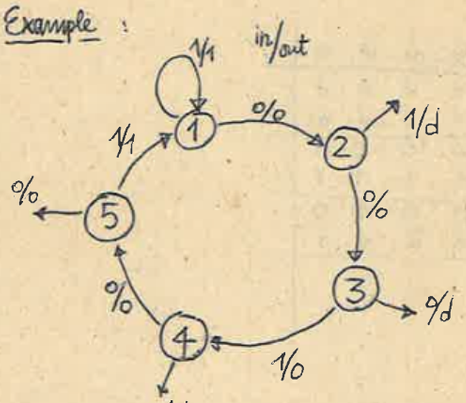
Definition : Compatible states (i c j) two states i and j of system S are said to be compatible iff the output sequences Z_i^r and Z_j^r are compatible for all input sequences I_i^r and I_j^r.

Definition : n mutually compatible states : A set of n states of system S is said to be mutually compatible iff every pair of the states of the set is compatible.

{i, j, k}

Definition : Maximal compatible states : a set of n mutually compatible states is said to be a set of maximal compatible states iff it does not form a proper subset of another set of states that is mutually compatible.

Example :



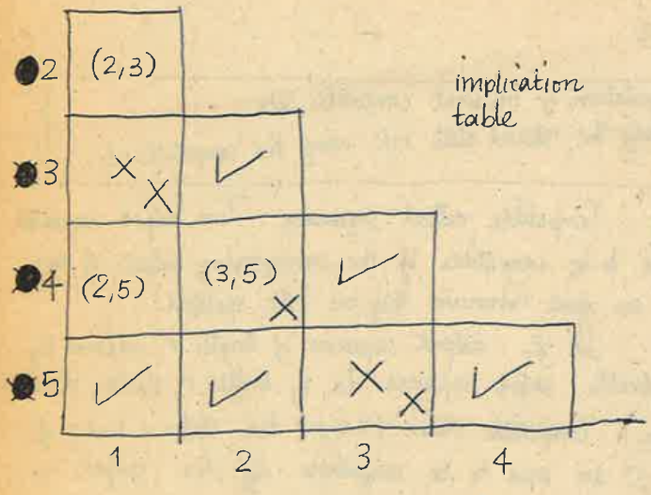
PS	NS x=0	NS x=1	OUT x=0	OUT x=1
1	2	1	0	1
2	3	d	0	d
3	d	4	d	0
4	5	2	0	d
5	d	1	0	1

Find compatible state pairs ;

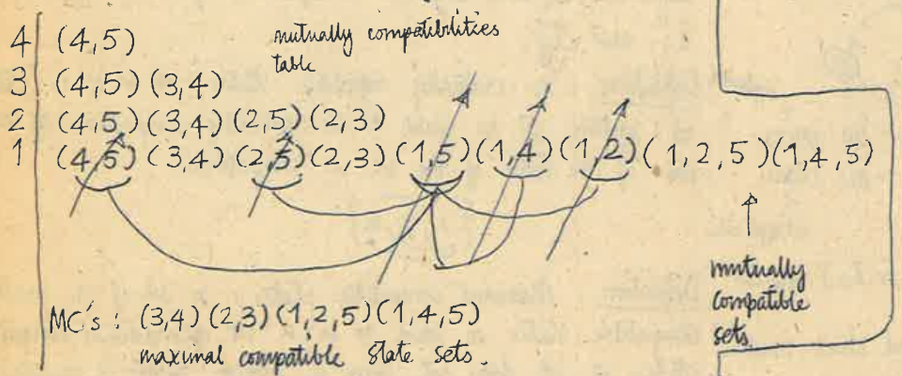
(1,5) are compatible
(1,3) not compatible
(1,2) are compatible if 2 and 3 are compatible
(1,2) → (2,3)
↑ implied state pair.

Paull Unger Tabulation method to find maximal compatible states sets:

i. implication table
ii. mutually compatibilities table. → MC's



implication table



mutually compatibilities table

mutually compatible sets

MC's: (3,4)(2,3)(1,2,5)(1,4,5)
maximal compatible state sets.

S: original m/c $S_1, S_2 \dots S_j$

Δ: m/c obtained by taking all maximal compatibilities as its states.
states are: $\delta_1 \delta_2 \dots \delta_n$

T: reduced m/c of states $P_1, P_2 \dots P_k$ $k \le n$

T > S obtained by taking some MC's sets of S.

H = upper bound of k $\min\{j, n\}$
ex: upperbound of $\min\{6,5\} = 5$

$k \le H$

The selected set of MC's to form the m/c T of states $P_1, P_2 \dots P_k$ must satisfy the followings:

I: Completeness: The union of states in all P sets must contain all the states of the original system.

II: Consistency: (Closure) Any set of next states P produced by an input symbol applied to the system S with any of the states of a P set as a starting state must be either a p set or a subset of of a P set of T.

	P_1	P_2	P_3
	(2,5)	(1,4)	(0,3)
00	(2,5)		
01	(3,4) x		
10	(5,2)		
11	(4,1)		

III. Minimality: The number of states of T (=k)

$k \le H$

lower bound $U = \max\{l_1, l_2, \dots l_m\}$ where l_j is a ~~set of~~ maximal incompatible set.

4 (4,5) not compatible
3 (4,5)(3,4)
2 (4,5)(3,4)(2,3)
1 (4,5)(3,4)(2,3)(1,2)
0 (4,5) (3,4) (2,3) (1,2)

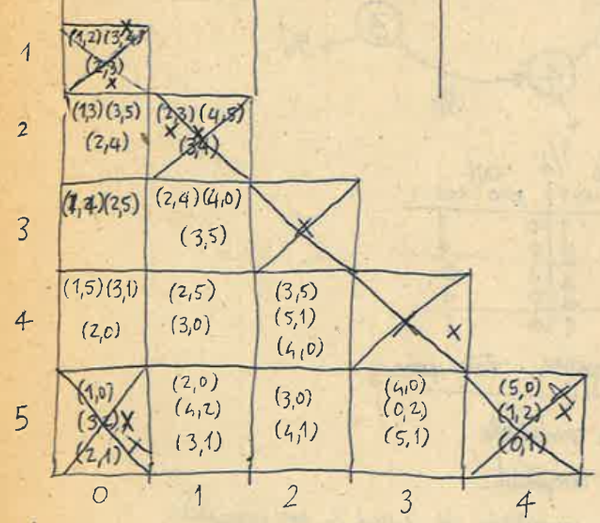
maximal incompatible sets of m/c S

$U = \max\{2, 2, 2, 3\}$

$2 \le k \le 5$

Example:

	00	01	10	11	00	01	10	11
0	0	1	3	2	d	d	d	d
1	1	2	4	3	d	d	d	d
2	2	3	5	4	0	0	0	0
3	3	4	0	5	1	1	1	1
4	4	5	1	0	0	0	0	0
5	5	0	2	1	d	d	d	d



4 x
3 (3,5)
2 (3,5)(2,5)(2,4)
1 (3,5)(2,5)(2,4)(1,5)(1,4)(1,3)(1,3,5)
0 (3,5)(2,5)(2,4)(1,5)(1,4)(1,3)(1,3,5)(0,4)(0,3)(0,2)(0,2,4)

MC's: (2,5)(1,4)(1,3,5)(0,2,4)(0,3)

Example :

PS	NS x=0	NS x=1	OUT x=0	OUT x=1
1	2	6	0	0
2	3	1	1	1
3	d	4	d	0
4	d	5	d	0
5	3	d	1	d
6	7	d	1	d
7	d	8	d	0
8	d	d	d	1

Answer: 3 state m/c

Example :

	NS, OUT x^1	x^2	x^3	x^4
1	2,d	3,0	d,d	4,d
2	3,0	5,0	d,d	d,d
3	4,d	6,d	3,d	d,d
4	5,d	3,0	d,d	1,d
5	d,d	6,0	d,d	d,d
6	d,d	d,1	4,d	2,d

Answer: 4 state m/c

1. MC's (0,2,4)(1,3,5)(2,5)(1,4)(0,3)
2. MC's (4,5)(3,4)(2,3)(1,2)(0,5)(0,1)

$H = 5$

$L = \max(2,2,2,2,2,2) = 2$

$5 \leq k \leq 2$

1. Completeness
2. Consistency
3. Minimality

(0,2,4)(1,3,5)(1,4)(2,5)(0,3)

more stable → (0,2,4)(1,3,5)(2,5)

Take:

1. (0,2,4)(1,3,5)

(0,2,4)(1,3,5) = {0,1,2,3,4,5}

Table of the next states :

INPUTS	PS (0,2,4)	PS (1,3,5)
0,0	(0,2,4)	(1,3,5)
0,1	(1,3,5)	(2,4,0)
1,0	(3,5,1)	(4,0,2)
1,1	(2,4,0)	(3,5,1)

NS (obtained from original table)

(0,2,4) = A
(1,3,5) = B

Consistency requirement is obtained (since # of NS are equal to # of PS in the table)

PS	NS 00	01	10	11	OUT 00	01	10	11
A	A	B	B	A	0	0	0	0
B	B	A	A	B	1	1	1	1

Example :

PS	NS 00	01	10	11	OUT 00	01	10	11
1	3	4	2	5	d	d	d	d
2	4	d	d	d	d	d	d	d
3	6	7	6	8	d	d	d	d
4	7	d	d	8	d	d	d	d
5	8	7	d	d	d	d	d	d
6	1	1	1	1	00	01	00	10
7	1	1	1	1	01	01	01	10
8	1	1	1	1	10	01	10	10

Find Maximal Compatible's (MC's):

$(1, \overset{C_1}{2}, 3, 6)(1, \overset{C_2}{2}, 3, 7)$

$C_3(1,2,3,8)(1,4,6)\,C_4$ ← No of Comp. state.

$C_5(1,4,7)(1,4,8)\,C_6$

$C_7(1,5,6)(1,5,7)\,C_8$

$C_9(1,5,8)$

Maximal Incompatible's (MIC's):

(6,7,8)(3,4,5)(2,4,5)

$H = \min(8,9) = 8$ → interesting thing is that the # of MC's are greater than the # of states of m/c

$L = \max(3,3) = 3$

$8 \geq k \geq 3$

$C_1C_5C_9 = C_1 \cup C_5C_9 = \{1,2 \cdots 8\}$

$C_1 \cup C_6 \cup C_8 = \{1,2 \cdots 8\}$

$C_2 \cup C_4 \cup C_9 = \{1,2 \cdots 8\}$

$C_2 \cup C_6 \cup C_7 = \{\cdots\}$

$C_3 \cup C_7 \cup C_5 = \{\cdots\}$

$C_5C_4C_8 = \{\cdots\}$

	(1,2,3,6)	(1,4,7)	(1,5,8)	(3,7,1)	(1,3,8)
00	(3,6,1)	(3,7,1)*	(1,3,8)*	(3,6,1)	(3,6,1)
01	(1,4,7)	(1,4)	(1,4,7)	(4,7,1)	(4,7,1)
10	(1,2,6)	(1,2)	(1,2)	(2,6,1)	(2,6,1)
11	(1,5,8)	(1,5,8)	(1,5)	(5,8,1)	(5,8,1)

Since (3,7,1) is neither equal nor a subset of PS, consistency requirement is not obtained [(1,3,8) is also]

For all other combinations satisfying completeness, not possible to satisfy consistency !

If we take $C_1C_5C_9$ we must take

$C_1C_5C_9 + (3,7,1) + (1,3,8)$ So we have to add

C_2 C_3

to present states C_2 and C_3 and also find the NS's of C_2, C_3. Thus our m/c becomes a 5 state m/c and satisfies