ECE 1733: Switching Theory

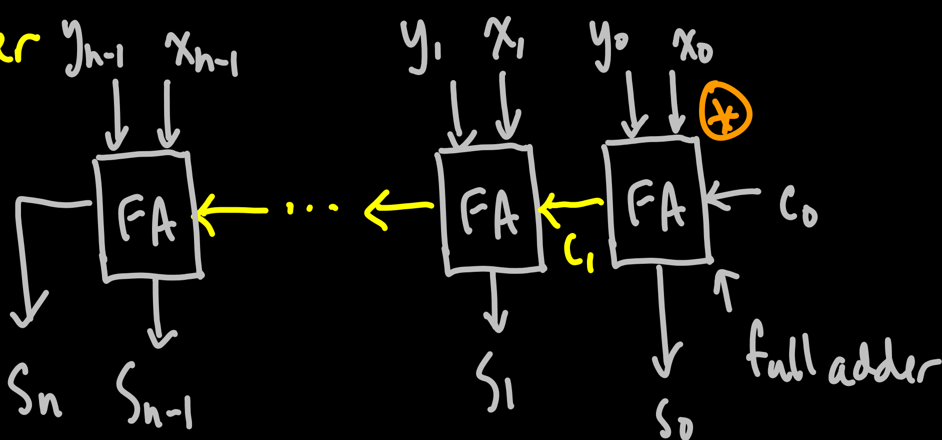
Steve Brown

- Review of combinational circuits ✓
- " of sequential circuits
- Optimization of logic functions using algorithmic methods
    - Assign #1: implement a program that performs 2-level optimization
- Functional decomposition
- Binary Decision Diagrams (BDDs)
    - Assign #2: implement a "BDD Package".
- Boolean Satisfiability (SAT)
- Assign #3: paper summary presentation

Representation of Logic Functions

Truth tables, sum-of-minterms (sum-of-products), Boolean expressions, Karnaugh maps, cubical notation, binary decision diagrams (BDDs), and-inverter graph (AIG)

Example: Adder



FA: Truth table

C_0	y_0	x_0	C_1	S_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

minterms

m_0 $\bar{C}_0\bar{y}_0\bar{x}_0$

m_1 $\bar{C}_0\bar{y}_0 x_0$

⋮

m_7 $C_0 y_0 x_0$

↑

AND

OR

↓

Sum-of-minterms

$S_0 = \Sigma m(1,2,4,7)$

$C_1 = \Sigma m(3,5,6,7)$

Boolean Expressions

Canonical S-of-P

$$S_0 = \bar{C}_0\bar{y}_0x_0 + \bar{C}_0y_0\bar{x}_0 + C_0\bar{y}_0\bar{x}_0 + C_0y_0x_0$$

$$C_1 = \bar{C}_0y_0x_0 + C_0\bar{y}_0x_0 + C_0y_0\bar{x}_0 + C_0y_0x_0$$

Karnaugh map

C_1

x_0 \ C_0y_0	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$C_1 = y_0x_0 + C_0x_0 + C_0y_0$ (majority function)

C_0	y_0	x_0	C_1	S_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

S_0

x_0 \ C_0y_0	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$S_0 = \bar{C}_0\bar{y}_0x_0 + \bar{C}_0y_0\bar{x}_0 + C_0\bar{y}_0\bar{x}_0 + C_0y_0x_0$$

$$S_0 = C_0 \oplus y_0 \oplus x_0 \quad \text{(odd function)}$$

Boolean Algebra

Axioms:

$0 \cdot 0 = 0$ ($\cdot$ means AND)

$1 + 1 = 1$ (+ means OR)

$\vdots$

Rules:

$x \cdot 0 = 0$

$x + 1 = 1$

$\vdots$

if $x = 0$, $\bar{x} = 1$ (" $-$ " means NOT, or

$x = 1$, $\bar{x} = 0$ complement)

Identities:

$$\left.\begin{aligned} x \cdot y &= y \cdot x \\ x + y &= y + x \end{aligned}\right\} \text{commutative}$$

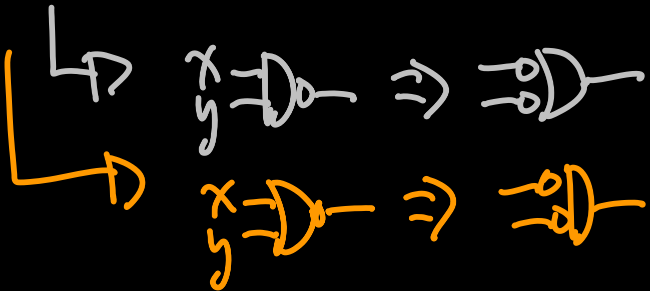
$$\left.\begin{aligned} x \cdot (y + z) &= xy + xz \\ x + (y \cdot z) &= (x + y) \cdot (x + z) \end{aligned}\right\} \text{distributive}$$

$$\left.\begin{aligned} (x + y) + z &= x + (y + z) \\ (x \cdot y) \cdot z &= x \cdot (y \cdot z) \end{aligned}\right\} \text{associative}$$

$$\left.\begin{aligned} &\rightarrow xy + x\bar{y} = x \\ &\rightarrow (x + y) \cdot (x + \bar{y}) = x \end{aligned}\right\} \text{combining}$$

$$\rightarrow xx + xy + yx + y\bar{y} = x + xy = x$$

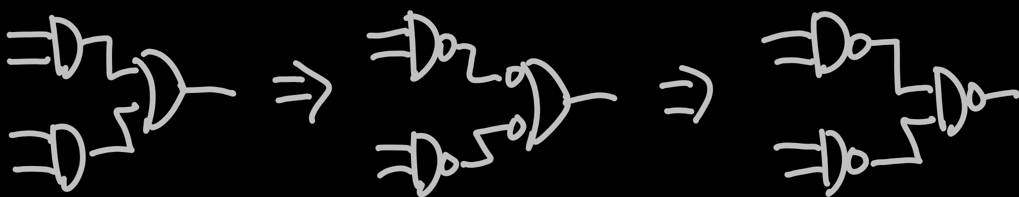
$$\left.\begin{aligned} \overline{xy} &= \bar{x} + \bar{y} \\ \overline{x + y} &= \bar{x}\bar{y} \end{aligned}\right\} \text{Demorgan's theorem}$$



- Using NAND Gates

4 trans. 6 trans.

SOP



- Product-of-Sums (POS)

c_0	y_0	x_0	c_1	s_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Maxterm

M_0 $\quad c_0 + y_0 + x_0$

M_1 $\quad c_0 + y_0 + \bar{x}_0$

$\vdots$

M_7 $\quad \bar{c}_0 + \bar{y}_0 + \bar{x}_0$

POS: $s_0 = \prod M(0,3,5,6)$

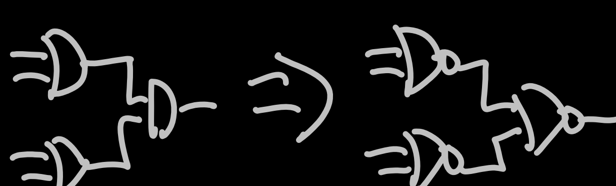
$C_1 = \prod M(0,1,2,4)$

C_1

x_0 \ $c_0 y_0$	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$C_1 = (c_0 + x_0)\cdot(c_0 + y_0)\cdot(y_0 + x_0)$

- Using DeMorgan's



Using Boolean Algebra

$$c_1 = \bar{c}_0 y_0 x_0 + c_0 \bar{y}_0 x_0 + c_0 y_0 \bar{x}_0 + c_0 y_0 x_0$$

$$\bar{c}_0 y_0 x_0 + c_0 y_0 x_0 = y_0 x_0$$

$$c_0 \bar{y}_0 x_0 + c_0 y_0 x_0 = c_0 x_0$$

3 times.

$x = x + x + x$

$$c_0 y_0 \bar{x}_0 + c_0 y_0 x_0 = c_0 y_0$$

Ex.

$$f = (\bar{x}_1 \bar{x}_2)\cdot(x_3 x_4) + (x_1 + x_2)\cdot(x_3 + x_4)$$

- let $k = (x_1 + x_2)$

$x = x + xy$

then, $f = \bar{k}\cdot(x_3 x_4) + k\cdot(x_3 + x_4)$

$$= \bar{k} x_3 x_4 + k x_3 + k x_4$$

$$= \bar{k} x_3 x_4 + k x_3 x_4 + k x_3 + k x_4$$

$$= x_3 x_4 + k(x_3 + x_4)$$

$$= x_3 x_4 + (x_1 + x_2)(x_3 + x_4)$$

4-Variable K-map

$f(x_1, x_2, x_3, x_4)$

x_3x_4 \ x_1x_2	00	01	11	10
00	m_0	4	12	8
01	m_1	5	13	9
11	3	7	15	11
10	2	6	14	10

x_1

$\bar{x}_2 \bar{x}_4$

minterms

x_1	x_2	x_3	x_4	
0	0	0	0	m_0
0	0	0	1	m_1
⋮				

$m_1 + m_3 = \bar{x}_1\bar{x}_2\bar{x}_3x_4 + \bar{x}_1\bar{x}_2x_3x_4 = \bar{x}_1\bar{x}_2x_4$

$m_5 + m_7 + m_{13} + m_{15} = \bar{x}_1x_2\bar{x}_3x_4 + \bar{x}_1x_2x_3x_4 + x_1x_2\bar{x}_3x_4 + x_1x_2x_3x_4$

$= x_2x_4\left(\underbrace{\bar{x}_1\bar{x}_3 + \bar{x}_1x_3 + x_1\bar{x}_3 + x_1x_3}_{1}\right)$

$= x_2x_4$

Example

x_3x_4 \ x_1x_2	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	1	1	1	0
10	1	1	0	1

SoP: $\bar{x}_1x_3 + \bar{x}_2\bar{x}_4 + x_2x_3x_4$

PoS: $(x_3+\bar{x}_4)\cdot(\bar{x}_2+x_3)\cdot(\bar{x}_1+x_2+\bar{x}_4)\cdot(\bar{x}_1+\bar{x}_2+x_4)$

Terminology

Literal: a variable, or it's complement

e.g. $x_1\bar{x}_2$ has two literal

Implicant: for a given function, the implicants are the product terms that are covered by the function. E.g. minterms, groups of two 1's in K-map, groups of 4, ...

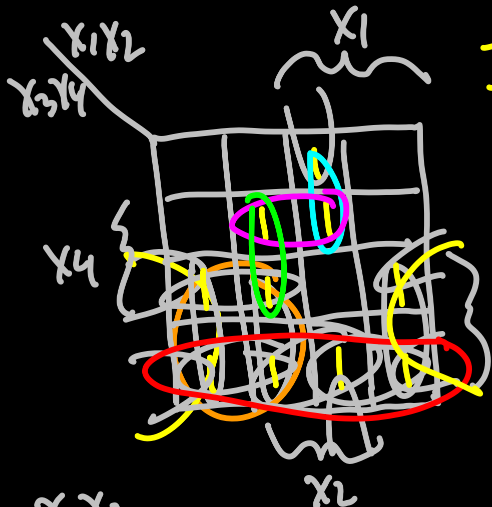
Prime Implicant: Informally: the biggest groups of 1's in a functions K-map.

Formally: any implicant for which it is not possible to remove any literal and still have a valid implicant

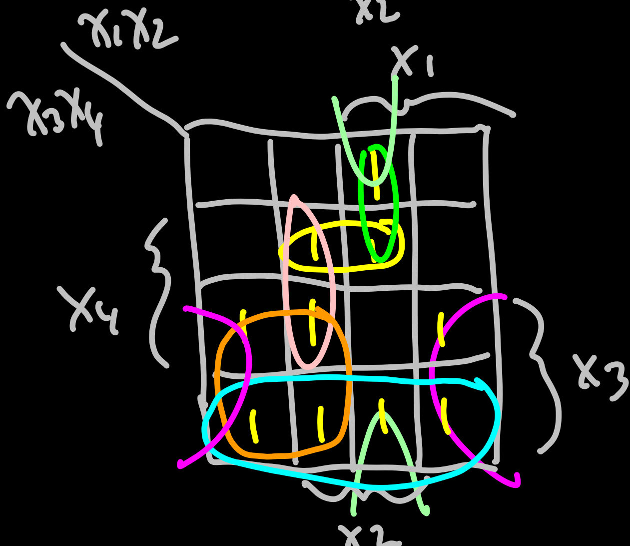
Essential PI : a PI that covers at least one minterm which is not covered by any other PI.

Cover : any sum of implicants that covers all of the minterms of a function.

Example



Implicants : {10 minterms}, groups of 2 1's {$x_1x_2\bar{x}_3$, $x_2\bar{x}_3x_4$, $\bar{x}_1x_2x_4$, plus 11 others}, $\bar{x}_1x_3$, $x_3\bar{x}_4$, $\bar{x}_2x_3$



Prime Implicants : $\bar{x}_1x_3$, $\bar{x}_2x_3$, $x_3\bar{x}_4$, $x_2\bar{x}_3x_4$, $x_1x_2\bar{x}_3$, $\bar{x}_1x_2x_4$, $x_1x_2\bar{x}_4$

Ess. PIs : $\bar{x}_2x_3$

Cover : $\bar{x}_2x_3$, $x_3\bar{x}_4$, $x_1x_2\bar{x}_3$, $\bar{x}_1x_2x_4$

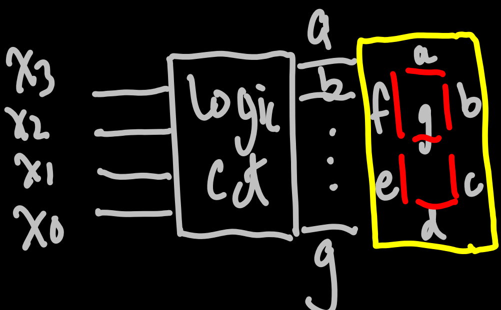
In general, we have to consider many (or even all) of PI choice-combinations to find the minimal cover(s).

Incompletely specified logic functions

Often, we know that some combinations of a function's inputs won't occur in practise, or if they do occur, we don't care about the function's output in that case. These cases are Don't Care inputs.

Example: using binary-coded decimal (BCD)

$x_3x_2x_1x_0$



	$x_3x_2x_1x_0$	a	…	e	f	g
0	0000	1		1		
1	0001	0		0		
2	0010	1		1		
3	0011	1		0		
4	0100	0	…	0	…	
5	0101	1		0		
6	0110	1		1		
7	0111	1		0		
8	1000	1		1		
9	1001	1		0		

K-map a

$x_1x_0 \backslash x_3x_2$	00	01	11	10
00	1	0	d	1
01	0	1	d	1
11	1	1	d	d
10	1	1	d	d

x_3

x_1

$\bar{x}_2\bar{x}_0$

x_2x_0

$x_1x_0 \backslash x_3x_2$	00	01	11	10
00	1	0	d	1
01	0	0	d	0
11	0	0	d	d
10	1	1	d	d

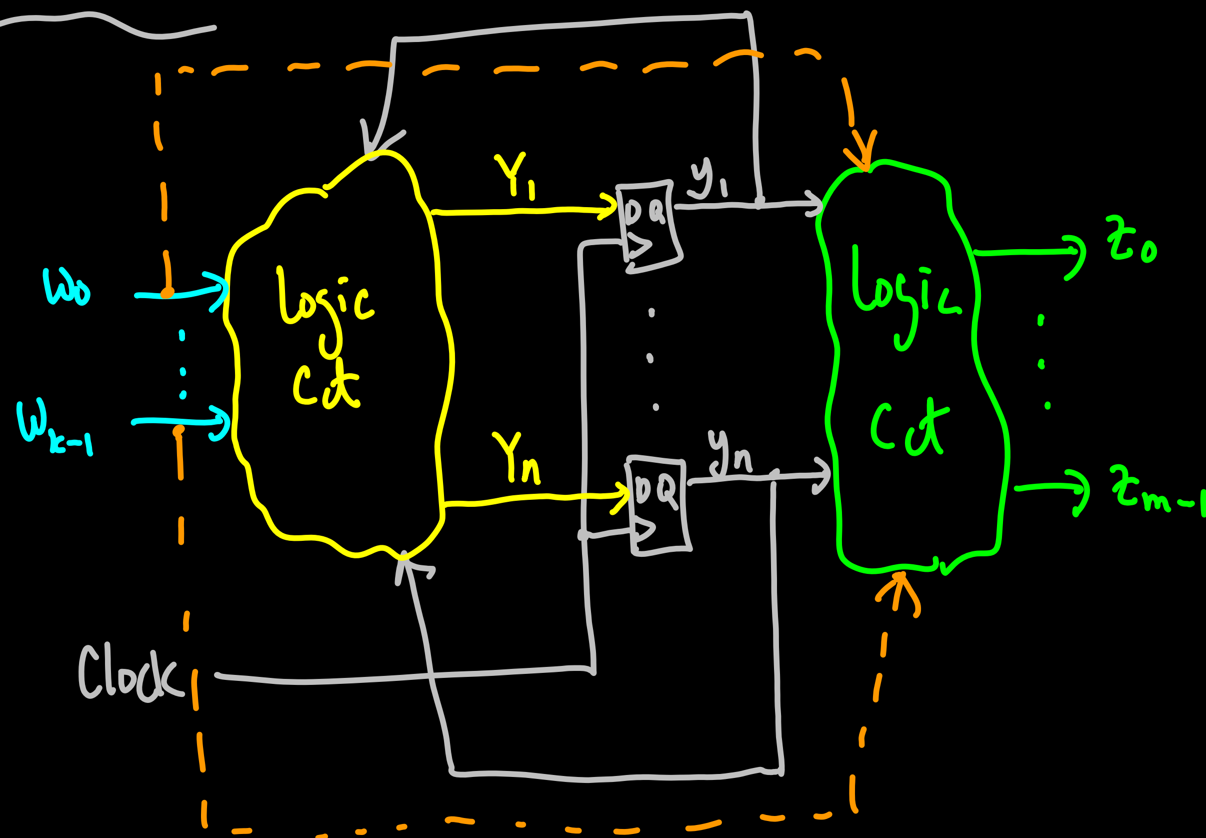
$\bar{x}_2\bar{x}_0$

$x_1\bar{x}_0$

Sequential Circuits (review)

Def'n: a seq. ckt is one in which outputs depend on both the current and previous inputs. Hence, the ckt includes stored state information.

General model



Seq. ckts. are also called finite state machines (FSMs).

Moore-type FSM (w.o. - -)

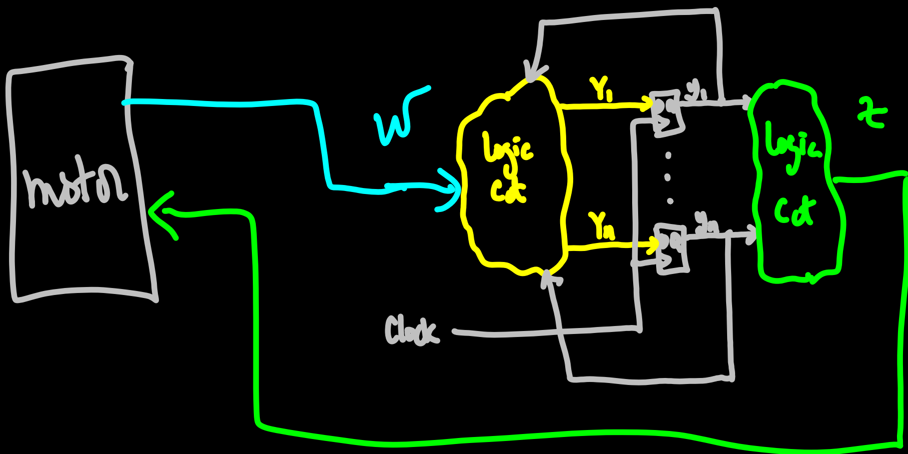
Mealy-type FSM (with - -)

FSM Design Steps (review)

Problem spec: design a ckt to control a motor. An input w is monitored by our FSM. If $w = 1, 0, 1$ over three clock cycles, then the motor has malfunctioned. Our FSM has to set an output $z = 1$ in the next clock cycle to reset the motor. Otherwise $z = 0$.

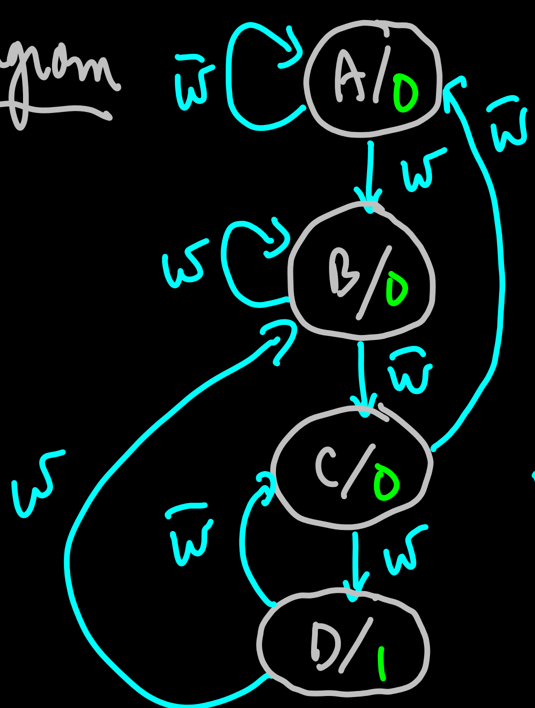
w 0 1 1 0 0 0 1 0 1 0 1 . . .

z 0 0 0 0 0 0 0 0 0 1 0 1 . . .



101

1. State Diagram



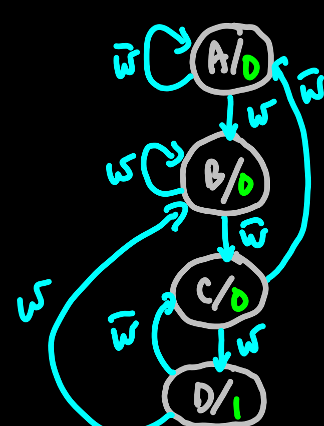
11...1

...10

101

State Table

Present state	Next state $w=0$	Next state $w=1$	z
A	A	B	0
B	C	B	0
C	A	D	0
D	C	B	1



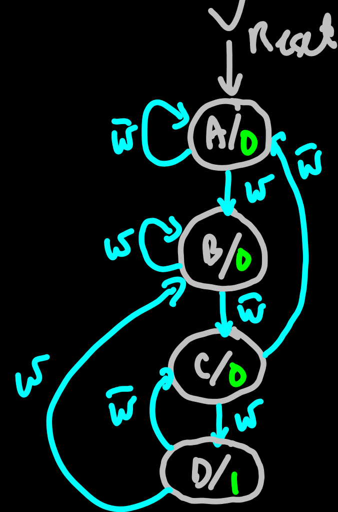
Choose the # of FFs.

Present state	Next state $w=0$	Next state $w=1$	z
A	A	B	0
B	C	B	0
C	A	D	0
D	C	B	1

First choice: one FF for each state, with state codes

	y_1	y_2	y_3	y_4
A	1	0	0	0
B	0	1	0	0
C	0	0	1	0
D	0	0	0	1

} one-hot encoding



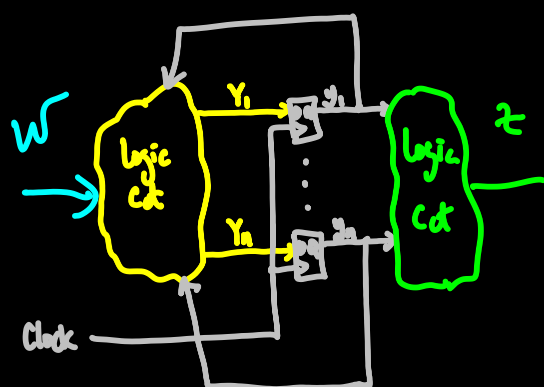
$$Y_1 = \bar{w}y_1 + \bar{w}y_3 = \bar{w}(y_1+y_3)$$

$$Y_2 = w(y_1+y_2+y_4)$$

$$Y_3 = \bar{w}(y_2+y_4)$$

$$Y_4 = wy_3$$

$$z = y_4$$



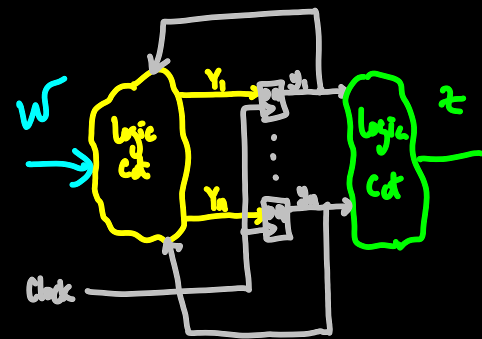
Second choice: minimum # of FF (2), with codes

	y_1	y_2
A	0	0
B	0	1
C	1	0
D	1	1

State-assigned Table

	PS	NS		z
		$w=0$	$w=1$	
	y_1y_2	Y_1Y_2	Y_1Y_2	
A	00	00	01	0
B	01	10	01	0
C	10	00	11	0
D	11	10	01	1

Present state	Next state		z
	$w=0$	$w=1$	
A	A	B	0
B	C	B	0
C	A	D	0
D	C	B	1



Y_1:

y_1y_2 \ w	0	1
00	0	0
01	1	0
11	1	0
10	0	1

$\bar{w}y_2$

$wy_1\bar{y}_2$

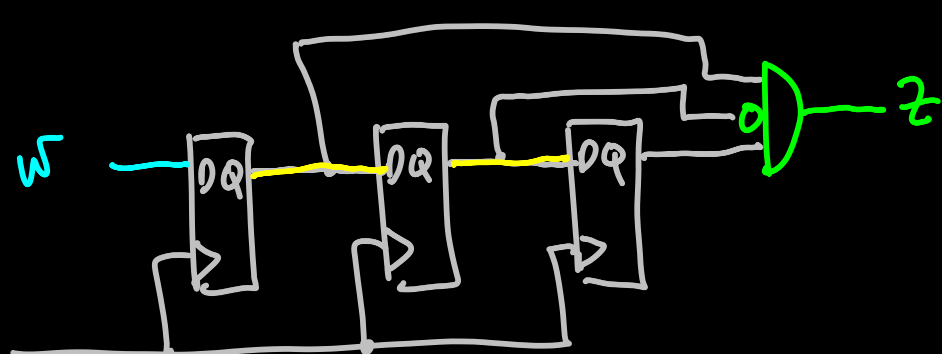
$\therefore \quad Y_1 = \bar{w}y_2 + wy_1\bar{y}_2$

$Y_2 = w$

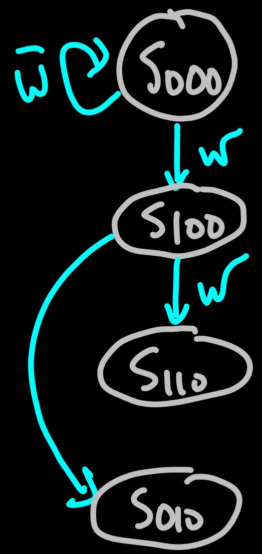
$z = y_1y_2$

Third choice: use 3 FFs, with codes that correspond to the last three values of w.

Result (from intuition) will be:



State Diagram



⋮ etc!

State Table

PS	NS $w=0$	NS $w=1$	z
S_{000}	S_{000}	S_{100}	0
S_{001}	S_{000}	S_{100}	0
S_{010}	S_{001}	S_{101}	0
S_{011}	S_{001}	S_{101}	0
S_{100}	S_{010}	S_{110}	0
S_{101}	S_{001}	S_{110}	1
S_{110}	S_{011}	S_{111}	0
S_{111}	S_{011}	S_{111}	0

- eventually we will get the shift register result.

State Minimization

- Given a set of states for an FSM, we can determine which states may be equivalent and are therefore not needed.

Def'n: for two states S_i, S_j, the states are equivalent if the FSM will produce an identical sequence of outputs independent of starting at S_i or S_j.

Example: Given the set of states $\{S_{000}, S_{001}, \dots, S_{111}\}$ from our previous example, find the minimum # of states

Step 1: partition states by output values

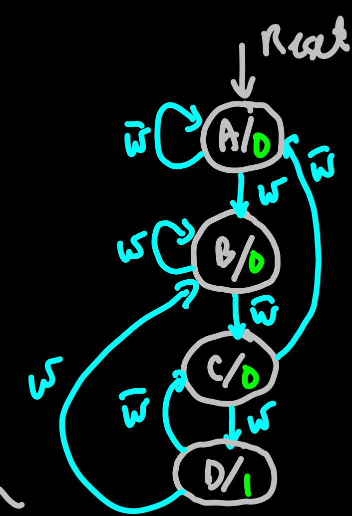
$\{S_{000}, S_{001}, S_{010}, S_{011}, S_{100}, S_{110}, S_{111}\} \{S_{101}\}$

Step 2: consider the 0-successors and 1-successors within each set.

S_{000}	S_{000}	S_{100}
S_{001}	S_{000}	S_{100}
S_{010}	S_{001}	S_{101}
S_{011}	S_{001}	S_{101}
S_{100}	S_{010}	S_{110}
S_{101}	S_{010}	S_{110}
S_{110}	S_{011}	S_{111}
S_{111}	S_{011}	S_{111}

$\{S_{000}, S_{001}, S_{100}, S_{110}, S_{111}\}\{S_{101}\}\{S_{010}, S_{011}\}$

$\underbrace{\{S_{000}, S_{001}\}}_{A} \underbrace{\{S_{100}, S_{110}, S_{111}\}}_{B} \underbrace{\{S_{101}\}}_{C} \underbrace{\{S_{010}, S_{011}\}}_{D}$



Algorithmic Methods for Minimization

Definition:

Cube is an implicant (product term). A function can be represented as a set of cubes

Example cubes: a cube is an n-tuple, where each digit can have three values: 0, 1, x. If a variable is uncomplemented in the cube, then it is a 1; if complemented it is a ϕ; if that variable is not present, then it is x

abcd	Cube
$\bar{a}\bar{b}\bar{c}\bar{d}$	0000
$abcd$	1111
$a\bar{b}c$	101x
cd	xx11

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

$$= \{x1x1, 0x10, 1001, 0xx1\}$$

Quine-McClusky

Step 1: arrange the minterms of the function by the cardinality of 1's

$$f = \Sigma m(1,3,5,6,7,8,14)$$

Def'n: **Size** of a cube is defined as the number of digits equal to x. 1010 (0-cube), x111 (1-cube), ...

	a b c d				
1.	0001 ✓	(1,3)	00X1 ✓	(1,3,5,7)	$\boxed{0XX1}$
8.	$\boxed{1000}$	(1,5)	0X01 ✓	(1,5,3,7)	0XX1
3.	0011 ✓	(3,7)	0X11 ✓		
5.	0101 ✓	(5,7)	01X1 ✓		
6.	0110 ✓	(6,7)	$\boxed{011X}$		
7.	0111 ✓	(6,14)	$\boxed{X110}$		
14	1110 ✓				

Step 2 : within each neighbourly group, try to combine all pairs of cubes (i.e., look for cubes that differ in one digit only)

- The result of this iterative procedure is the prime implicants of the function.

Step 3 : draw a minterm cover table for selecting prime implicants. Identify the Essential PIs

PI	m_1	m_3	m_5	m_6	m_7	m_8	m_{14}
1000						✓	
011X				✓	✓		
X110				✓			✓
0XX1	✓	✓	✓		✓		

Here, the Ess. PIs are 1000, X110, 0XX1. These cover all minterms of f.

$$\therefore \quad f = \{1000, X110, 0XX1\}$$

$$f = \sum m(0,2,3,4,6,7,9,12,13,15,16,23,24,25,29,31)$$

00000 ✓	
00010 ✓	
00100 ✓	
10000 ✓	
00011 ✓	
00110	
01001	
01100	
11000	
00111	
01101	
11001	
01111	
10111	
11101	
11111	

(0,2)	000X0
(0,4)	00X00
(0,16)	X0000
(2,3)	0001X
⋮	

⇒ Prime Implicants

Step 2	0	2	3	4	6	7	9	12	13	15	16	23	24	25	29	31
00XX0	✓	✓		✓	✓											
X0000	✓										✓					
00X1X		✓	✓		✓	✓										
0X100				✓				✓								
1X000											✓		✓			
X1X01							✓		✓					✓	✓	
0110X								✓	✓							
1100X													✓	✓		
XX111						✓				✓		✓				✓
X11X1									✓	✓					✓	✓

Iteration: create a simplified table, removing the selected PIs and the covered minterms

	0	4	12	16	24	
00xx0	✓	✓				
x0000	✓			✓		
0x100		✓	✓			
1x000				✓	✓	
~~0110x~~			✓			dominated
~~1100x~~					✓	dominated
~~x11x1~~						useless

Row dominance: if one row dominates another, then delete the dominated row.

Redraw the simplified table:

	0	4	12	16	24
00xx0	✓	✓		✓	
x0000	✓				
0x100		✓	✓		
1x000				✓	✓

Now, we select "Essential PIs" 0x100, 1x000. Then, to cover m0 select 00xx0 (lower cost than x0000).

Note: for some cover tables, column dominance occurs:

	m_0	m_4	m_{12}	m_{16}	m_{24}
00XX0	✓	✓			
X0000	✓			✓	
0X100		✓	✓		
1X000				✓	✓

dominates (*) dominates

(*) The m_{12} column tells us that we need 0X100. Since this PI also covers m_4, we don't need the m_4 column.

Summary

1. Generate all PIs from minterms through iterative combing
2. Create a cover table, and identify Ess. PIs.
3. if there are uncovered minterms, make a reduced cover table.
4. Use row/column dominance to reduce the cover table.

5. Identify new "Ess. PIs".

6. Iterate from step 3.

Note: it may be necessary to make "arbitrary" choices at the end. Example:

	m_i	m_j	m_k
PI_A	✓	✓	
PI_B		✓	✓
PI_C	✓		✓

- look at all combinations (branch & bound)

Using *-operation and #-operation for minimization

*-operation

- consider two 0-cubes (minterms) $A = 010$, $B = 110$

$A = 010$ A_1 A_2 A_3

$B = 110$ B_1 B_2 B_3

- two step *-operation process:

① Use Table

A_i \ B_i	0	1	X
0	0	$\emptyset$	0
1	$\emptyset$	1	1
X	0	1	X

intuition: the *-op table returns what A_i, B_i have "in common"

For our example

$A = 010$ $\qquad A_1 * B_1 = 0*1 = \phi$

$B = 110$ $\qquad A_2 * B_2 = 1*1 = 1$

$\qquad\qquad\quad A_3 \times B_3 = 0*0 = 0$

Step 2: form the overall result $C = A*B$

Two cases: 1. $C = \phi$ if $A_i * B_i = \phi$ for more than one i.

2. Otherwise

$C_i = A_i * B_i$ when $\neq \phi$

$C_i = x$ when $= \phi$

Example:

$A = 1x0$, $B = 1x1$

$A_1 * B_1 = 1$

$A_2 * B_2 = x$

$A_3 * B_3 = \phi$

Case 2: $C = 1xx$

Algebraicly: $A = x_1 \bar{x}_3$

$B = x_1 x_3$

$A + B = x_1$

$x_3 \backslash x_1x_2$	00	01	11	10
0	1	1		
1			1	1

Example:

$A = 0x0$

$B = 1x1$

$C = A*B = \phi$

Example: $A = 0X0$
$B = X11$

$A_1 * B_1 = 0$

$A_2 * B_2 = 1$

$A_3 * B_3 = \phi$

$\therefore C = A * B$
$= 01X$

$x_3 \backslash x_1x_2$	00	01	11	10
0	1	1		
1		1	1	

- The *-operation can be used to generate all implicants, and therefore all prime implicants.

$f(x_1, \ldots x_4) = \{ XX00, 110X, 1X11, 10X0 \}$

$A_i \backslash B_i$	0	1	X
0	0	ϕ	0
1	ϕ	1	1
X	0	1	X

$x_3x_4 \backslash x_1x_2$	00	01	11	10
00	1	1	1	1
01			1	
11			1	1
10				1

$110X * 1X11 = 11\phi1$

$11X1$

$1X11 * 10X0 = 101\phi$

$101X$

Procedure for Using *-op to Find All PIs

- let C^k be a cover (set of cubes) for a function
- generate a new set of cubes G^{k+1} by finding *-op for all pairs of cubes c^i, c^j.

$G^{k+1} = c^i * c^j \ \forall \ c^i, c^j \quad i \neq j$

- then $C^{k+1} = C^k \cup G^{k+1} -$ (redundant cubes)
- repeat until $C^{k+1} = C^k$

Example $f = m(1,3,5,6,7)$

$C^0 = \{001, 011, 101, 110, 111\}$

0x1 x01 ∅ ∅

∅ ∅ x11

∅ 1x1

11x

$\therefore G^1 = \{0x1, x01, x11, 1x1, 11x\}$

$C^1 = C^0 \cup G -$ (redundant)

$C^1 = G^1$

-Repeat: $C^1 = \{0x1, x01, x11, 1x1, 11x\}$

001 011 xx1 x11

xx1 101 1x1

111 111

111

$G^2 = \{001, 011, xx1, x11, 101, 1x1, 111\}$

$C^2 = C^1 \cup G^2 - (\text{redundant})$

$C^2 = \{\underline{xx1}, \underline{11x}\}$

~repeat $G^3 = 111$

$C^3 = C^2 \cup G^3 - (\text{redundant}) = C^2$ Done!

check: $f = m(1,3,5,6,7)$

$x_3 \backslash x_1x_2$	00	01	11	10
0			1	
1	1	1	1	1

xx1

11x

Problem: design a cct for which $f=1$ iff

- $(x_1 = x_2)$ and $(x_3 = x_4)$, but $x_1 \neq x_3$ or
- $(x_1 = x_3)$ and $(x_2 = x_4)$, but $x_1 \neq x_2$ or
- $x_1 = x_4 = 1$
- $x_1 = 1$, $x_2 = x_3 = x_4 = 0$

$C^0 = \{0011, 1100, 0101, 1010, 1xx1, 1000\}$

$$C^0 = \begin{pmatrix} 0011 \\ 1100 \\ 0101 \\ 1010 \\ 1XX1 \\ 1000 \end{pmatrix} \qquad G^1 = \begin{pmatrix} X011 \\ 110X \\ 1X00 \\ X101 \\ 101X \\ 10X0 \\ 100X \end{pmatrix}$$

$$C^1 = \begin{pmatrix} 1XX1 \\ X011 \\ 110X \\ 1X00 \\ X101 \\ 101X \\ 10X0 \\ 100X \end{pmatrix}$$

(skip redundant)

$$G^2 = \begin{pmatrix} 1X0X \\ 10XX \end{pmatrix}$$

$$C^2 = \begin{pmatrix} 1XX1 \\ 1X0X \\ 10XX \\ X011 \\ X101 \end{pmatrix}$$

PIs

Then, we would find that

$C^3 = C^2$. Done

$x_3x_4 \backslash x_1x_2$	00	01	11	10
00			1	1
01		1	1	1
11	1		1	1
10				1

10XX

1X0X

1XX1

X011

X101

$C^0 = \{0011, 1100, 0101, 1010, 1XX1, 1000\}$

- In this process we need a way of identifying redundant cubes

#-Operation

- While $*$-op identifies the common parts of cubes, the #-op identifies how cubes "differ".

#-op $A \# B$ yields the part of A that is not covered by B.

Example: $A = 11X$, $B = XX0$ $\qquad 11X \# XX0 = 111$

$XX0 \# 11X = 0X0, X00$

$x_3 \backslash x_1x_2$	00	01	11	10
0	1	1	1	1
1			1	

Step 1: Find coordinate #-op $C_i = A_i \# B_i$ using table:

$A_i \backslash B_i$	0	1	X
0	ε	ϕ	ε
1	ϕ	ε	ε
X	1	0	ε

Case 1: A is not covered at all by B

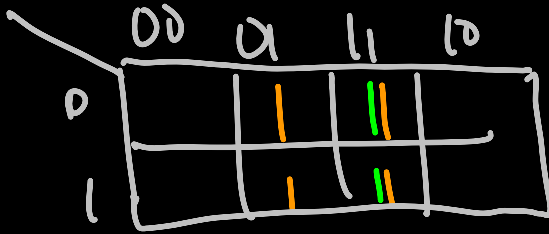
	00	01	11	10
0	1	1		
1			1	1

$A = 0X0 \qquad A \# B = \phi \varepsilon \phi$

$B = 1X1 \qquad B \# A = \phi \varepsilon \phi$

Rule 1: if $A_i \# B_i = \phi$ for any i, then $C = A \# B = A$

Case 2: A is fully covered by B



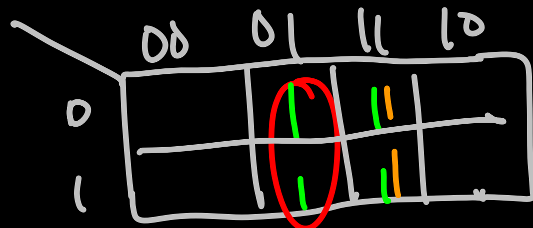
$A = 11X$

$B = X1X$

$A \# B = \varepsilon\varepsilon\varepsilon$

Rule 2: when $A_i \# B_i = \varepsilon\ \forall i$, then $C = A \# B = \phi$

Case 3: part of A is covered by B



$A = X\boxed{1X}$

$B = \boxed{1}1X$

$\boxed{0}\boxed{1X}$

- from the table, $A \# B = 0\varepsilon\varepsilon$

Example (case 3): $A = \boxed{11}X$, $B = XX0$ Table: $\varepsilon\varepsilon 1$

$C = 111$

$A = XX0$, $B = 11X$

000, 010 → 0X0 ✓

100 → X00 ✓

~~110~~

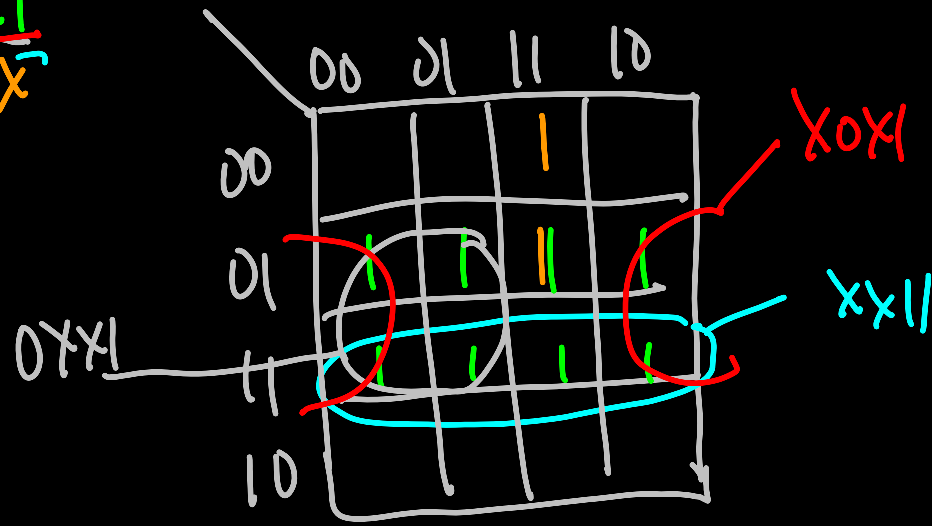
Rule 3: When calculating $C = A \# B$, for each i in which $A_i = X$ but $B_i \neq X$, there will be a cube produced. In this cube $C_i = \overline{B_i}$ and all other digits from A.

Example: $A = XXX1$

$B = 110X$

$C = A \# B$

$= \{0XX1, X0X1, XX11\}$



Using #-op to Identify Redundant Cubes

- given $C^k = \{0X0, 00X, 100, 0XX\}$
- to check if a cube p^i is redundant

$$p^i \# p^j = \emptyset \text{ , for any } i \neq j$$

- in this example:

$p^i = 0X0$

$0X0 \# 00X = \varepsilon 1 \varepsilon \neq \emptyset$

$0X0 \# 100 = \emptyset \quad \neq \emptyset$

$0X0 \# 0XX = \varepsilon\varepsilon\varepsilon = \emptyset$ Red!

$A_i \backslash B_i$	0	1	X
0	ε	$\emptyset$	ε
1	$\emptyset$	ε	ε
X	1	0	ε

$p^i = 00X$

$00X \# 0X0 = \varepsilon\varepsilon 1 \neq \emptyset$

$00X \# 100 = \emptyset \quad \neq \emptyset$

$00X \# 0XX = \varepsilon\varepsilon\varepsilon = \emptyset$ Red!

etc.! (100, 0XX are Not Red.)

Complete Procedure for Find a minimal Cover

1. Use $*$-op to find all implicants, and remove redundant cubes using #-op $\Rightarrow$ PIs.
2. Using {PIs}, identify which ones are **Essential**.

Recall: an Ess. PI covers at least one minterm not covered by any other PI.

- Given a cover consisting of PIs

$C^k = \{PIs\}$, and p^i is a cube in that set, to check if p^i is essential

$$p^i \# (C^k - p^i) \neq \phi$$

e.g. $C^k = \{p^1, p^2, p^3, p^4\}$

To check if p^2 is essential:

$$((p^2 \# p^1) \# p^3) \# p^4 \neq \phi$$

Example: $C^K = \{\underline{0X0}, 01X, X11, 1X1\}$

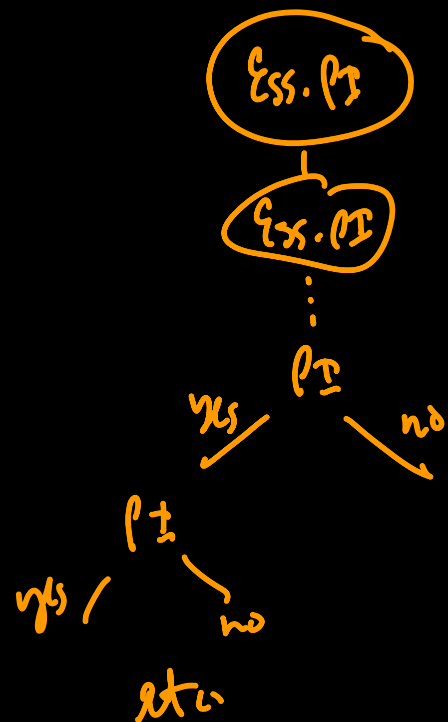
1. $0X0 \# 01X = 000 \# X11 = 000 \# 1X1 = 000$ $\therefore$ essential!
2. $01X \# 0X0 = 011 \# X11 = \phi$ $\therefore$ not essential.

... etc.

3. Include the Ess. PIs in the cover. If all minterms of the function are covered, then <u>done</u>.

4. Choose additional PIs to cover the minterms.

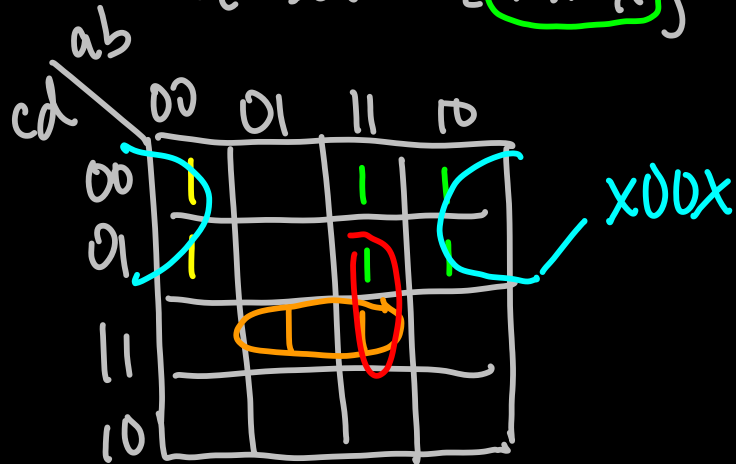
Branch & Bound



Incorporating Don't Cares

<u>Example</u>: f has ON-set = {000x, x111}

DC-set = {1x0x}



Step 1: Use $C^0 = \{\text{ON-set}\} \cup \{\text{DC-set}\}$ to find PIs, using $*$-op (and #-op for removing redundant cubes).

NOTE: resulting PIs should be checked (with #-op) to make sure that they cover at least one ON-set minterm, and not just DC-set minterms.

Here, PIs = {X00X, X111, 11X1, ~~1X0X~~}

DC!

Step 2: find Ess. PIs.

$$\left(\left(\left(P^2 \# P^1\right) \# P^3\right) \# P^4\right) \# \{\text{DC-set}\}$$

Example: 11X1 # X00X = 11X1 # X111 = 1101 # 1X0X = ϕ

$\therefore$ not Ess.

Marking of Assign #1:

Input format is flexible? 0-1
Intermediate results displayed? 0-1
Output format is readable? 0-2?
All minimal sol'ns found? 0-2
Thoroughness of testing: 0-2
Code structure: 0-2

Code uses sensible variable names & comments : 0-2

Run time: 0-1

Multilevel Representation of Logic Functions

Example: Implement a function f that is 1 when (at least one of inputs x_1, x_2 is equal to 1) and (both x_3, x_4 are 1). Also, f is 1 if ($x_1 = x_2 = 0$) and either (x_3, x_4 is 1).

$$f = (x_1 + x_1)(x_3 x_4) + \bar{x}_1 \bar{x}_2 (x_3 + x_4)$$

Note: $\underline{\bar{a}\boxed{bc}} + a(b+c)$

$= bc + a(b+c)$.

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

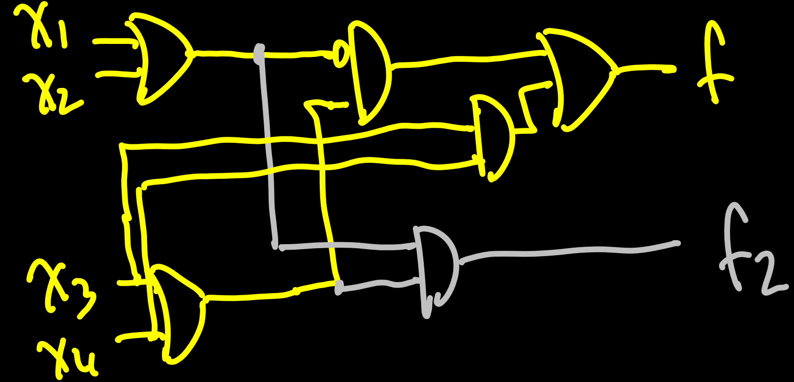
$$f = (x_3 x_4) + \bar{x}_1 \bar{x}_2 (x_3 + x_4)$$

- Assume we have a second function f_2 which is equal to 1 in all cases except when (both x_1, x_2 are 0) or (both x_3, x_4 are 0)

$$\bar{f}_2 = \bar{x}_1 \bar{x}_2 + \bar{x}_3 \bar{x}_4$$

$$\therefore f_2 = (x_1 + x_2)(x_3 + x_4)$$

$$f = (x_3 x_4) + \bar{x}_1 \bar{x}_2 (x_3 + x_4)$$
$$= (x_3 x_4) + \overline{(x_1 + x_2)}(x_3 + x_4)$$



Example: $f = \Sigma m(4, 7, 9, 10, 12, 13, 14, 15)$

$x_3x_4 \backslash x_1x_2$	00	01	11	10
00		1	1	
01			1	1
11		1	1	
10			1	1

$f = x_2 \bar{x}_3 \bar{x}_4 + x_1 \bar{x}_3 x_4 +$
$x_2 x_3 x_4 + x_1 x_3 \bar{x}_4$

Cost = # gates + # gate-inputs
= 5 + 16 = 21

Functional Decomposition

- Find subfunctions of some, but not all, of the variables that we can use in a multilevel expression.
- Example: find subfunction of x_3, x_4

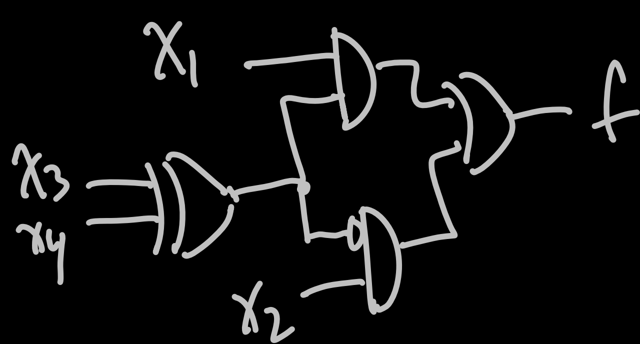
$$f = x_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_3x_4 + x_2x_3x_4 + x_1x_3\bar{x}_4$$

$$= x_1(\bar{x}_3x_4 + x_3\bar{x}_4) + x_2(\bar{x}_3\bar{x}_4 + x_3x_4)$$

Now, subfunction $g = \bar{x}_3x_4 + x_3\bar{x}_4 = x_3 \oplus x_4$

then $\bar{g} = \bar{x}_3\bar{x}_4 + x_3x_4 = x_3 \odot x_4$

$$\Rightarrow f = x_1 g + x_2\bar{g}$$



$$\boxed{g = x_2\bar{g} + x_1 g}$$

same

Cost : 5 gates + 9 = 14 < 21

Deriving the multilevel Expression from K-map

g

$x_3x_4 \backslash x_1x_2$	00	01	11	10
00		1	1	
01			1	1
11		1	1	
10			1	1

x_3x_4	g
00	0
01	1
10	1
11	0

$g = x_3 \oplus x_4$. The function f is defined by g when $x_1x_2 = 10$. So, this is $g \cdot x_1\bar{x}_2$. The second column is $\bar{g}$, and represents this part of f : $\bar{g} \cdot \bar{x}_1 x_2$. The column 11 is

(1) $\cdot\ x_1x_2 \cdot \quad \therefore f = g x_1 \bar{x}_2 + \bar{g}\bar{x}_1 x_2 + x_1x_2$

- We can think of this as a 2-level expression of a 3-variable function $h[x_1, x_2, g]$

$\therefore f(x_1,x_3,x_3,x_4) = h[x_1, x_2, g(x_3, x_4)]$

- We can optimize h

$g \backslash x_1x_2$	00	01	11	10
0		1	1	
1			1	1

$\therefore g = x_2\bar{g} + x_1 g$

Example

$x_5 = 0$

$x_3x_4 \backslash x_1x_2$	00	01	11	10
00	1			
01		1	1	1
11	1			
10		1	1	1

$x_5 = 1$

$x_3x_4 \backslash x_1x_2$	00	01	11	10
00				
01	1	1	1	1
11				
10	1	1	1	1

$\bar{g}$

$g = x_1 + x_2 + x_5$

Example: explore subfunctions of x_1, x_2, x_5

- The part of f represented by g is

$$g\cdot(\bar{x}_3x_4 + x_3\bar{x}_4) = g\cdot(x_3 \oplus x_4)$$

- The part of f represented by $\bar{g}$ is

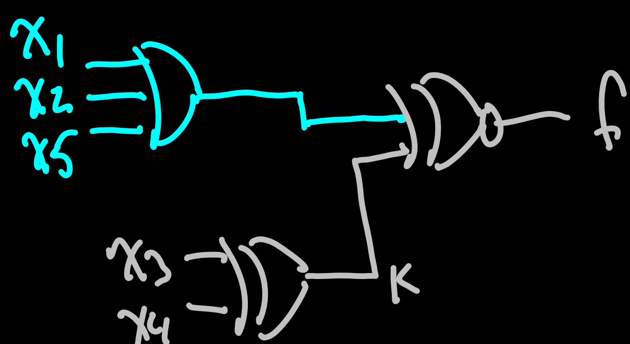
$$\bar{g}\cdot(x_3 \odot x_4)$$

$$\therefore \quad f = g(x_3 \oplus x_4) + \bar{g}\,\overline{(x_3 \oplus x_4)}$$

- let $k = x_3 \oplus x_4$

Then, $f = gk + \bar{g}\bar{k}$

$$= \overline{(g \oplus k)} = g \odot k$$



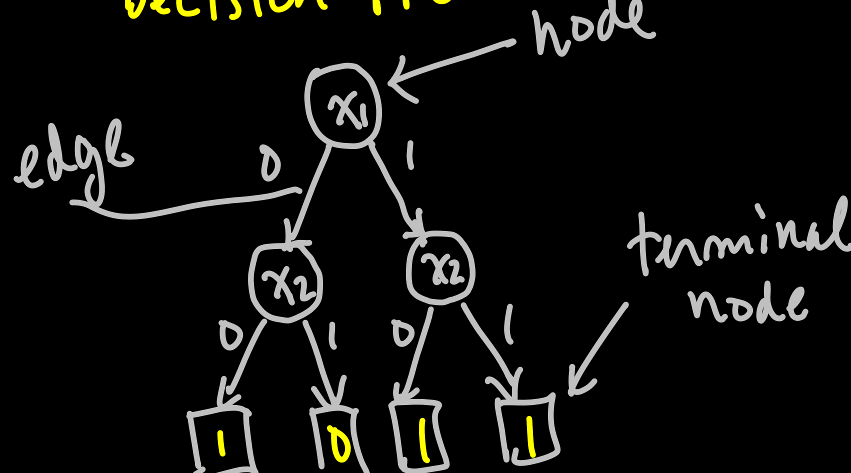
Binary Decision Diagrams (BDDs)

Def'n: a BDD is a graph that represents a logic function. For a specific order of variables, a BDD is a canonical representation.

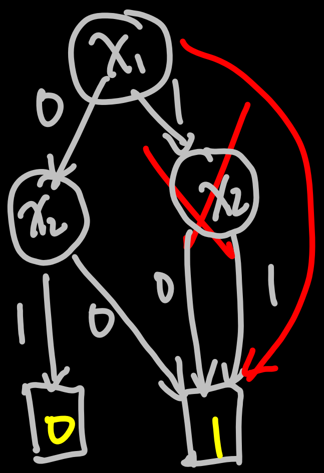
Example:

$x_1 x_2$	f
00	1
01	0
10	1
11	1

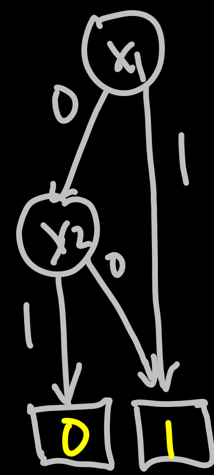
Decision Tree



① Merge identical nodes, starting from the bottom



② . remove redundant nodes

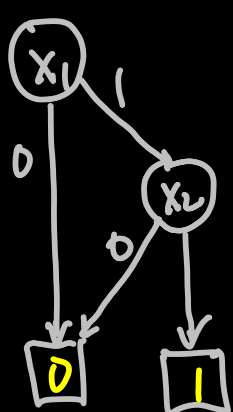


BDD (*)

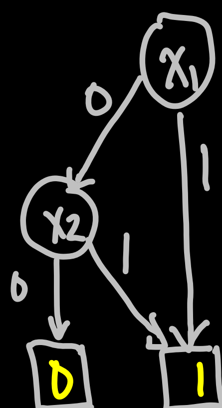
(*) BDD, a.k.a. Reduced Ordered BDD

AND, OR, XOR BDDs

$f = x_1 x_2$

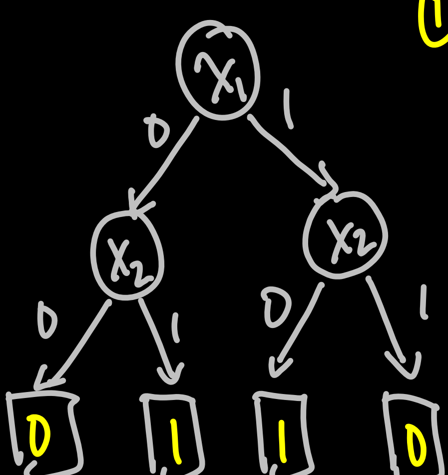


$f = x_1 + x_2$

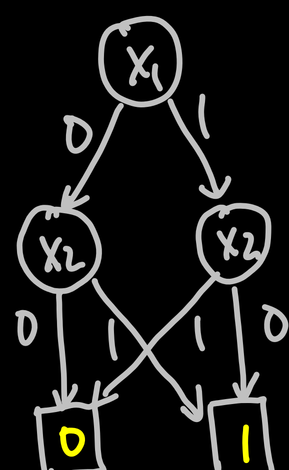


$f = x_1 \oplus x_2$

① merge



$\Rightarrow$



$f = x_1 \oplus x_2 \oplus x_3$