ECE 1733: Switching Theory

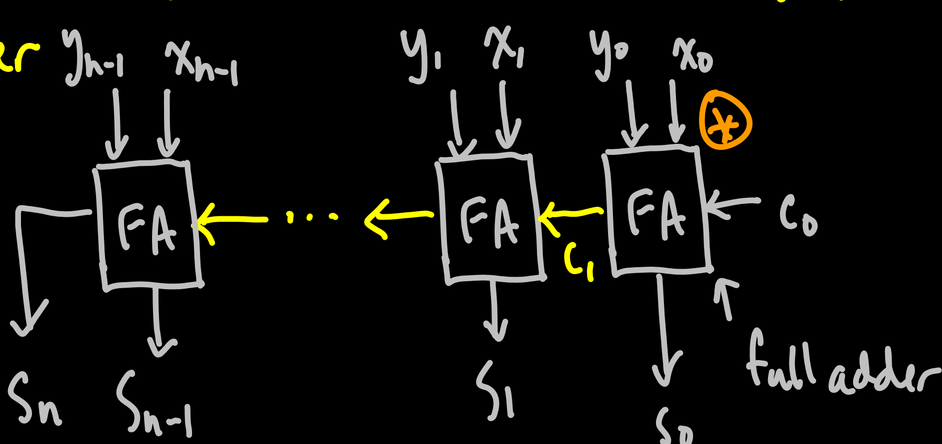
Steve Brown

- 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 diagram (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 $C_0y_0x_0$

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

$S_0 = \bar{C_0}\bar{y_0}x_0 + \bar{C_0}y_0\bar{x_0} + C_0\bar{y_0}\bar{x_0} + C_0y_0x_0$

$C_1 = \bar{C_0}y_0x_0 + C_0\bar{y_0}x_0 + C_0y_0\bar{x_0} + C_0y_0x_0$

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

$S_0 = \bar{C_0}\bar{y_0}x_0 + \bar{C_0}y_0\bar{x_0} + C_0\bar{y_0}\bar{x_0} + C_0y_0x_0$

$S_0 = C_0 \oplus y_0 \oplus x_0$ (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:

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

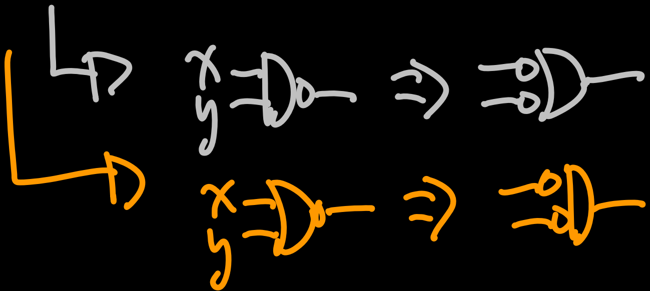
$$\left.\begin{aligned} x \cdot (y + z) &= xy + xz \\ x + (y \cdot z) &= (x + y) \cdot (x + z) \end{aligned}\right\} \text{distributive}$$

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

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

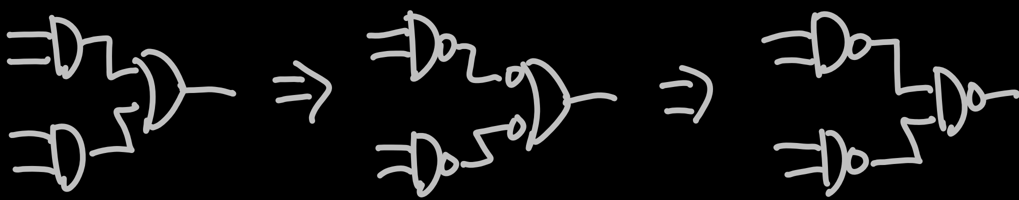
$\rightarrow xx + x\bar{y} + yx + y\bar{y} = x + xy = x$

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



- Using NAND Gates

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

⋮

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

POS: $S_0 = \prod M(0,3,5,6)$

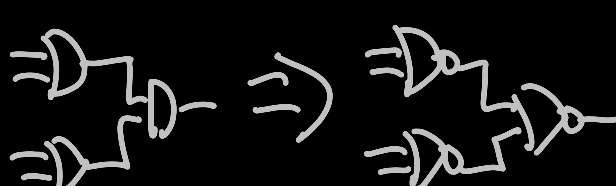
$C_1 = \prod M(0,1,2,4)$

C_1

x_0 \ c_0y_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
⋮				

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

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

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

$$= x_2x_4$$

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 its 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, ...

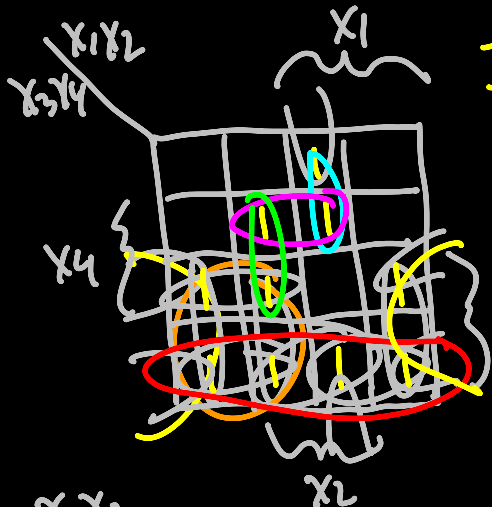
Prime Implicant: Informally: the biggest groups of 1's in a functions K-map.

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

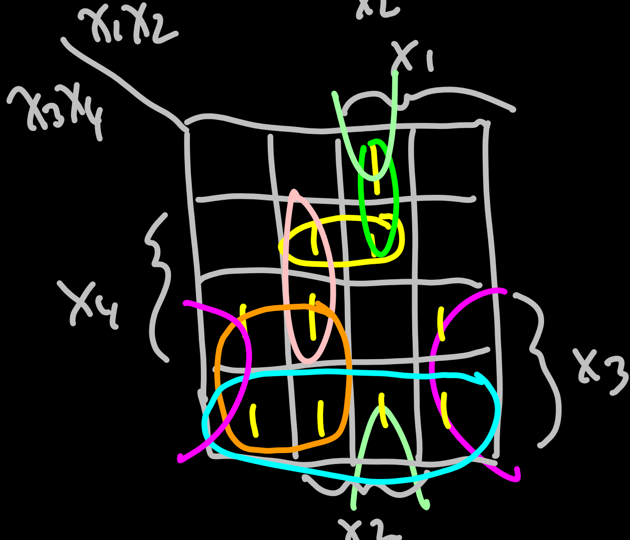
Essential PI : a PI that covers at least one minterm which is not covered by any other PI.

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 : $\bar{x}_2x_3$

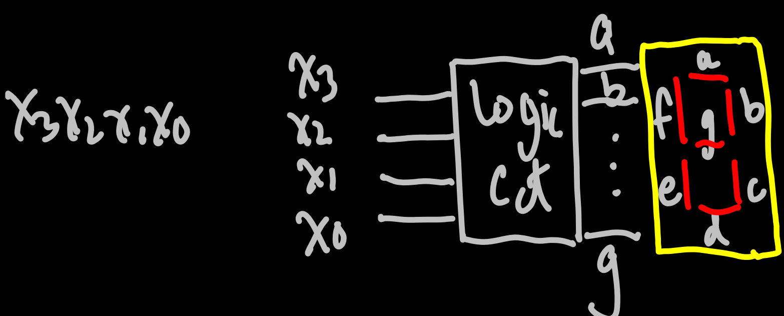
Cover: $\bar{x}_2x_3$, $x_3\bar{x}_4$, $x_1x_2\bar{x}_3$, $\bar{x}_1x_2x_4$

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)



	$x_3x_2x_1x_0$	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

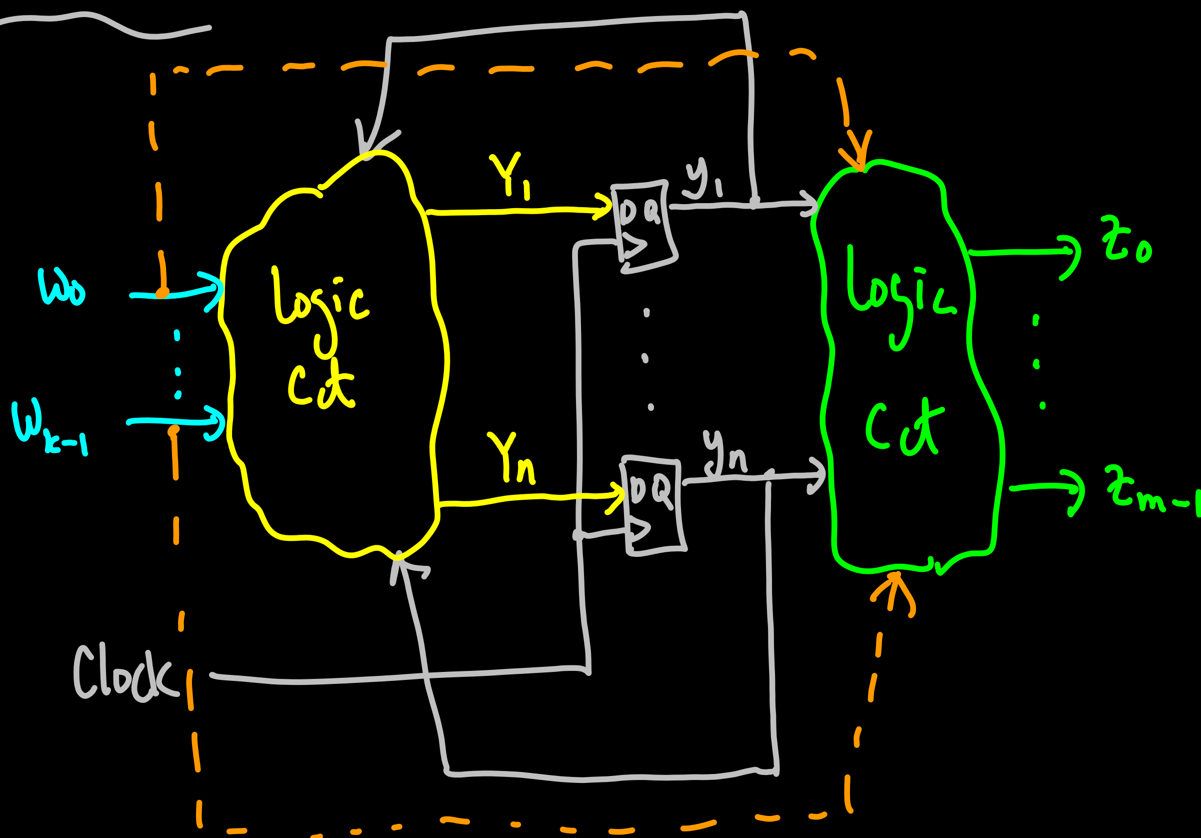
$\bar{x}_2\bar{x}_0$

$x_1\bar{x}_0$

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. - -)

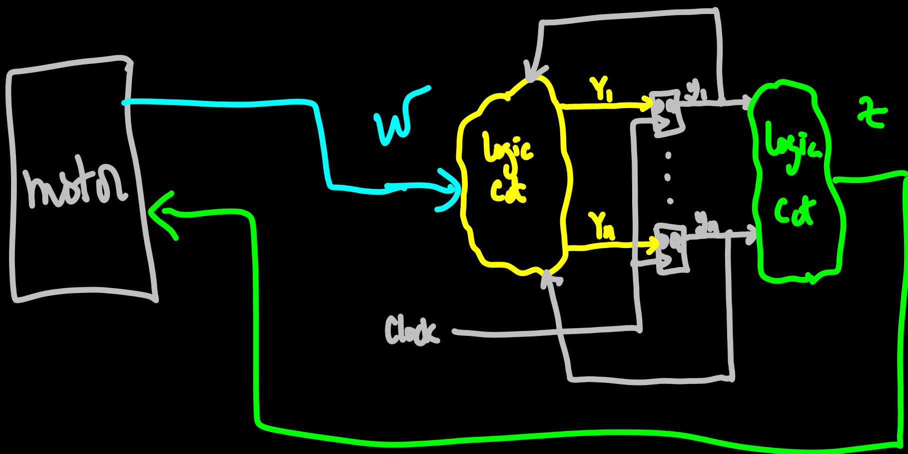
Mealy-type FSM (with - -)

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 $z=0$.

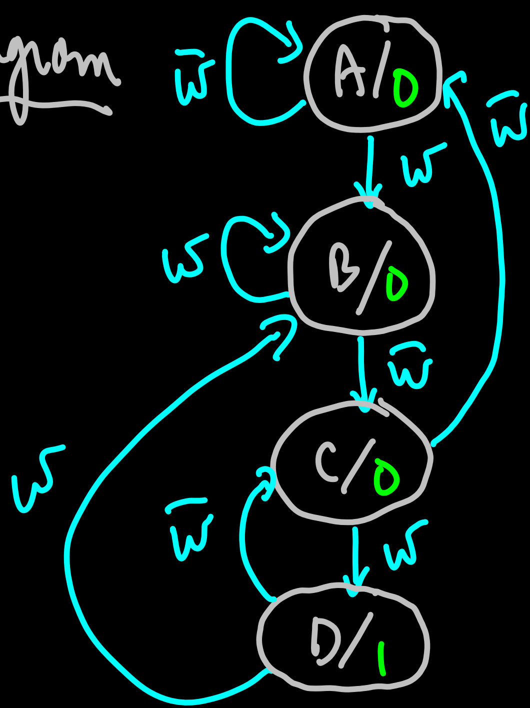
w 0 1 1 0 0 0 1 0 1 0 1 . . .

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



101

1. State Diagram



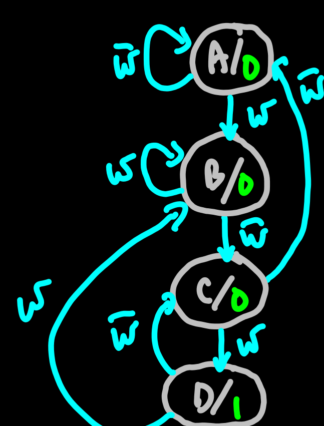
11...1

...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



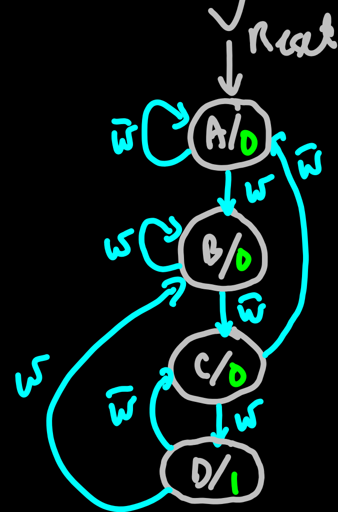
Choose the # of FFs.

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



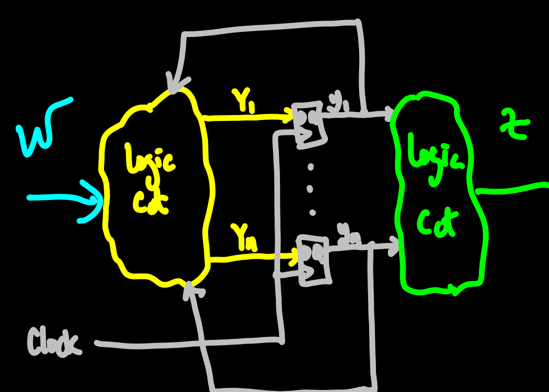
$$Y_1 = \bar{w} y_1 + \bar{w} y_3 = \bar{w}(y_1 + y_3)$$

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

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

$$Y_4 = w y_3$$

$$z = y_4$$

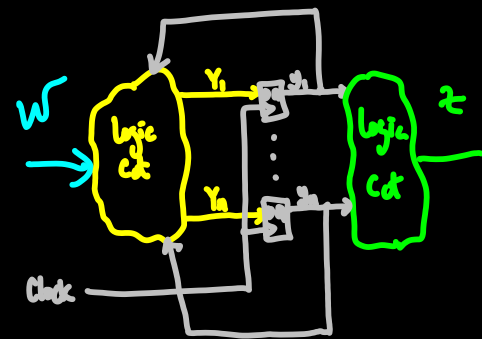


Second choice: minimum # of FF (2), with codes

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

State-assigned Table

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



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

Y_1:

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

$\bar{w}y_2$ $\quad$ $wy_1\bar{y}_2$

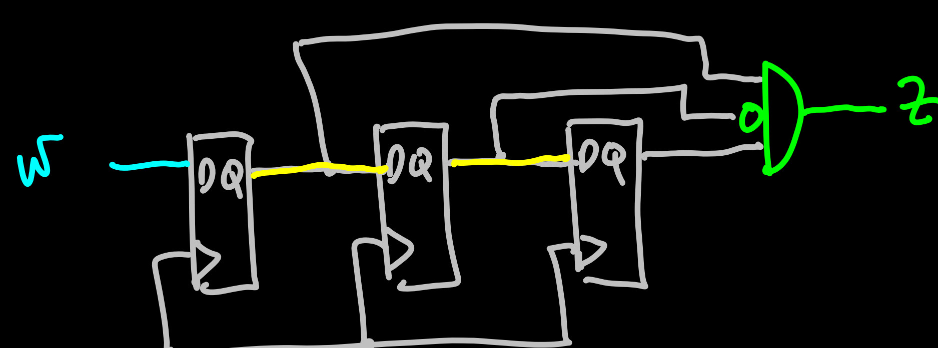
$\therefore\ Y_1 = \bar{w}y_2 + wy_1\bar{y}_2$

$Y_2 = w$

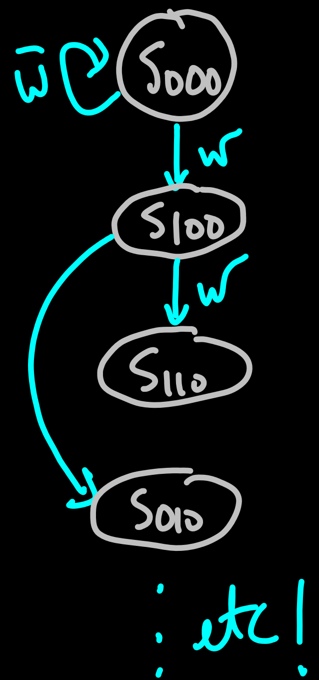
$z = y_1y_2$

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

Result (from intuition) will be:



State Diagram



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 $\{S_{000}, S_{001}, \ldots, S_{111}\}$ from our previous example, find the minimum # of states

Step 1: partition states by output values

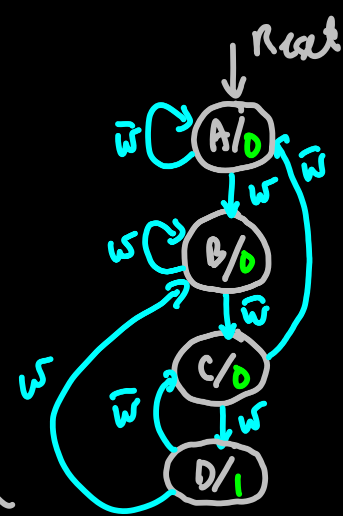
$\{S_{000}, S_{001}, S_{010}, S_{011}, S_{100}, S_{110}, S_{111}\}\ \{S_{101}\}$

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 ✓
8.	$\boxed{1000}$
3.	0011 ✓
5.	0101 ✓
6.	0110 ✓
7.	0111 ✓
14	1110 ✓

(1,3)	00X1 ✓
(1,5)	0X01 ✓
(3,7)	0X11 ✓
(5,7)	01X1 ✓
(6,7)	$\boxed{011X}$
(6,14)	$\boxed{X110}$

(1,3,5,7)	$\boxed{0XX1}$
(1,5,3,7)	0XX1

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

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

<u>Step 3</u>: 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 m_0 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

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

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

For our example

$A = 010$

$B = 110$

$A_1 * B_1 = 0 * 1 = \emptyset$

$A_2 * B_2 = 1 * 1 = 1$

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

<u>Step 2</u>: form the overall result $C = A * B$

Two cases:

1. $C = \emptyset$ if $A_i * B_i = \emptyset$ for more than one i.
2. Otherwise
   - $C_i = A_i * B_i$ when $\neq \emptyset$
   - $C_i = x$ when $= \emptyset$

<u>Example</u>:

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

$A_1 * B_1 = 1$

$A_2 * B_2 = x$

$A_3 * B_3 = \emptyset$

<u>Case 2</u>: $C = 1xx$

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

<u>Example</u>:

$A = 0x0$

$B = 1x1$

$C = A * B = \emptyset$

Example:

$A = 0X0$

$B = X11$

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

$A_1 * B_1 = 0$

$A_2 * B_2 = 1$

$A_3 * B_3 = \phi$

x_3 \ 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.