Lect 09

QM Methods and Combinational Block design

CS221: Digital Design

Dr. A. Sahu

Dept of Comp. Sc. & Engg.

Indian Institute of Technology Guwahati

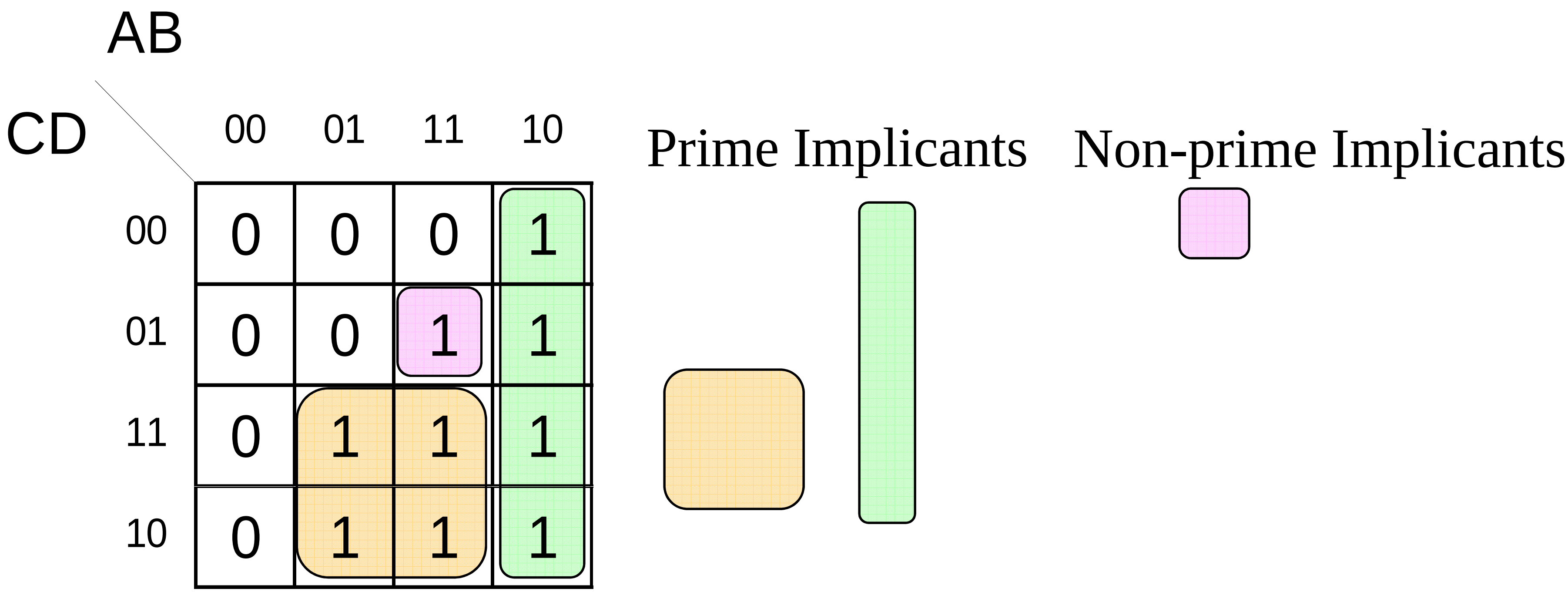
Outline

- Summery and Rest from K-MAP
- QM Methods : Tabular
- Combinational Block
  - Adder, Substractor, Multiplier, BCD Adder
- Mux and Demux
- Other Encoders

Prime Implicants

- A group of one or more 1's which are adjacent and can be combined on a Karnaugh Map is called an implicant.
- The *biggest* group of 1's which can be circled to cover a given 1 is called a prime implicant.
  - They are the only implicants we care about.

Prime Implicants

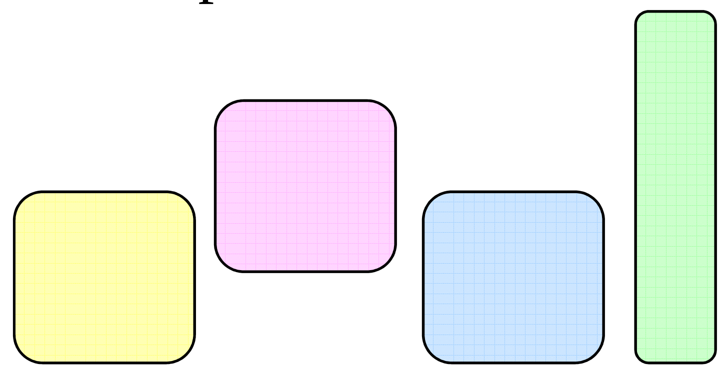


Are there any additional prime implicants in the map that are not shown above?

All The Prime Implicants

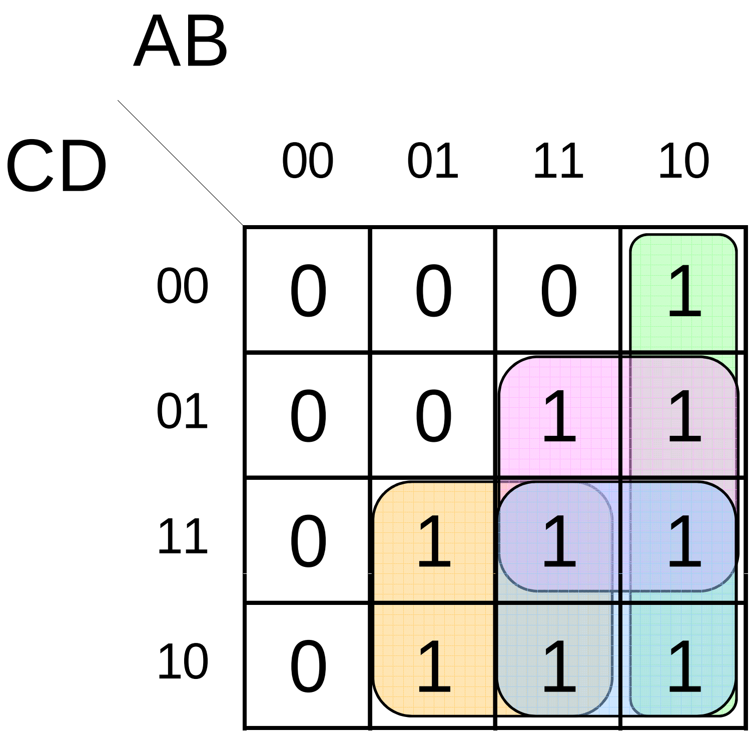
CD \ AB	00	01	11	10
00	0	0	0	1
01	0	0	1	1
11	0	1	1	1
10	0	1	1	1

Prime Implicants



When looking for a minimal solution –
only circle prime implicants…
A minimal solution will ***never*** contain
non-prime implicants

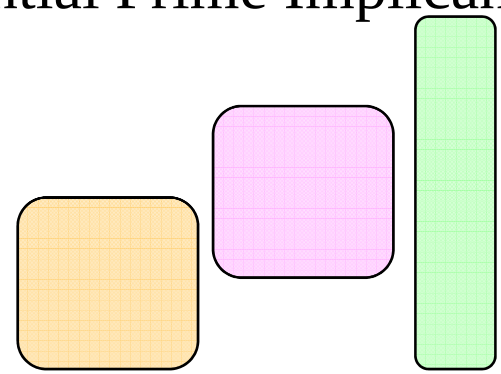
Essential Prime Implicants



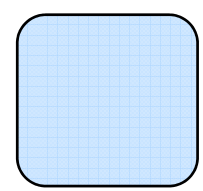
Not all prime implicants are required…

A prime implicant which is the only cover of some 1 is ***essential*** – a minimal solution requires it.

Essential Prime Implicants



Non-essential Prime Implicants



A Minimal Solution Example

CD \ AB	00	01	11	10
00	0	0	0	1
01	0	0	1	1
11	0	1	1	1
10	0	1	1	1

$F = \boxed{AB'} + \boxed{BC} + \boxed{AD}$

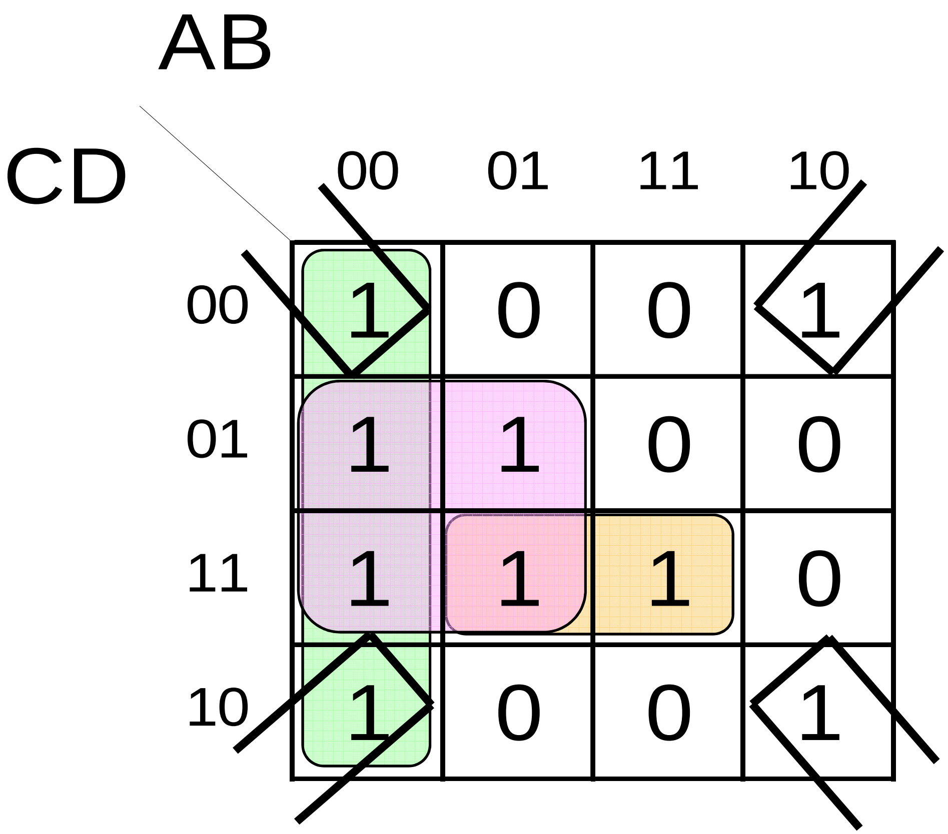
Minimum

Not required…

Another Example

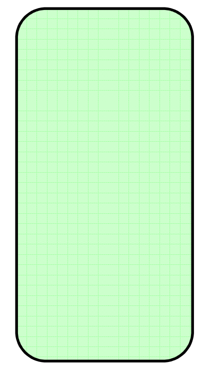
CD \ AB	00	01	11	10
00	1	0	0	1
01	1	1	0	0
11	1	1	1	0
10	1	0	0	1

Another Example



F = A'D + BCD + B'D'

Minimum



A'B' is not required…

Every one one of its locations is covered by multiple implicants

After choosing essentials, everything is covered…

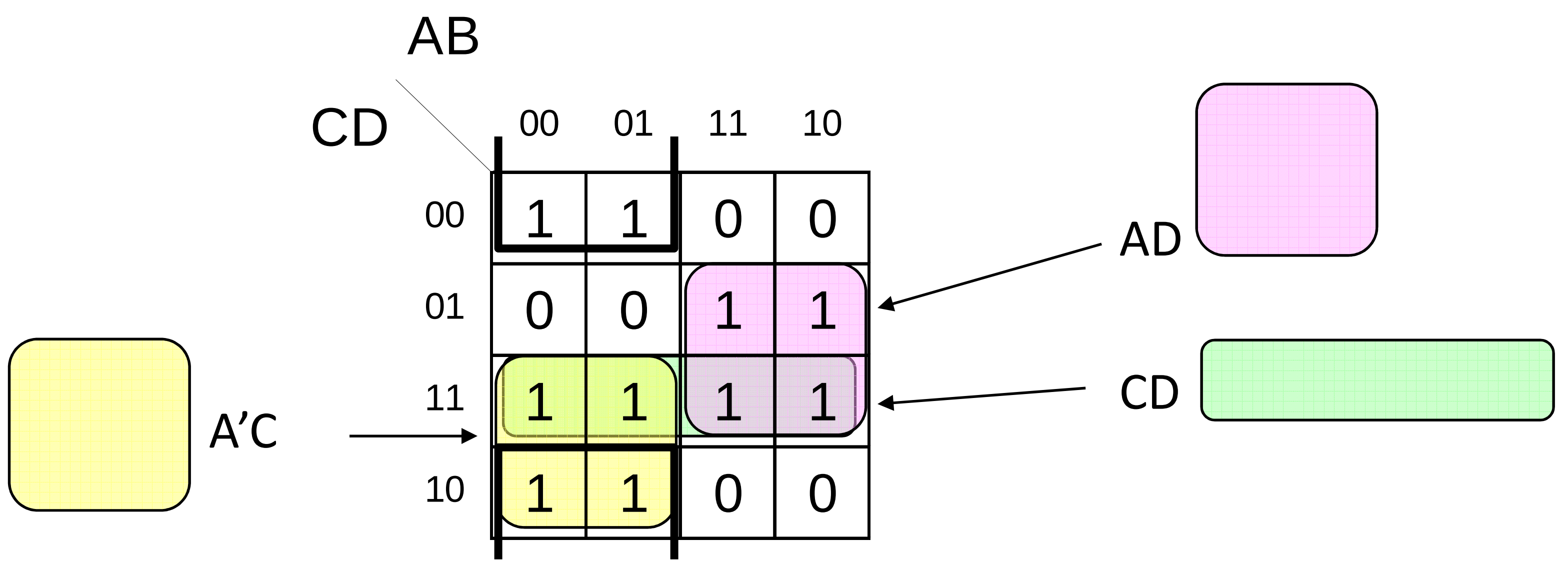
Finding the Minimum Sum of Products

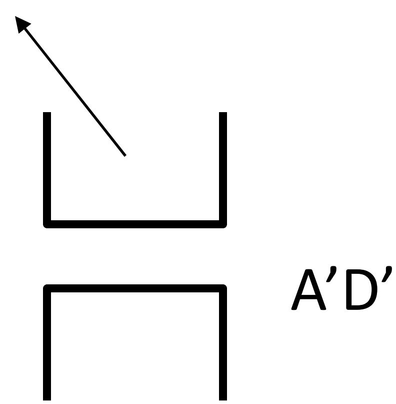
1. Find each essential prime implicant and include it in the solution.
2. Determine if any minterms are not yet covered.
3. Find the minimal # of remaining prime implicants which finish the cover.

Yet Another Example
(Use of non-essential primes)

CD \ AB	00	01	11	10
00	1	1	0	0
01	0	0	1	1
11	1	1	1	1
10	1	1	0	0

Yet Another Example (Use of non-essential primes)





Essentials: A'D' and AD

Non-essentials: A'C and CD

Solution: A'D' + AD + A'C

or

A'D' + AD + CD

KMap Summary

- A Kmap is simply a folded truth table
  - where physical adjacency implies logical adjacency
- KMaps are most commonly used hand method for logic minimization
- KMaps have other uses for visualizing Boolean equations
  - you may see some later.

Quine-McCluskey (QM) Method
for
Logic Minimization

Quine-McCluskey Method for Minimization

- KMAP methods was practical for at most 6 variable functions
- Larger number of variables: need method that can be applied to computer based minimization
- **Quine-McCluskey** method

- For example:

$$\sum m(0,1,2,3,5,7,13,15)$$

QM Method

- Phase I : finding Pis
  - Tabular methods: Grouping and combining
- Phase II: Covers minimal PIs

<u>QM Method</u>

- Minterms that differ in one variable’s value can be combined.
- Thus we list our minterms so that they are in groups with each group having the same number of 1s.
- So the first step is ordering the minterms according to their number of 1s (0-cube list)
- only minterms residing in adjacent groups have the chance to be combined.):

QM Method

$$\sum m(0,1,2,3,5,7,13,15)$$

0_Cube	
0	0000
1	0001
	0010
2	0011
	0101
3	0111
	1101
4	1111

QM Method: Combining Adjacent

- Compare minterms of a group with those of an adjacent one to form 1-cube list.
- When doing the combining, we put checkmark alongside the minterms in the 0-cube list that have been combined.

QM Method

0_Cube	
0	0000 √
1	0001 √
	0010 √
2	0011 √
	0101 √
3	0111 √
	1101 √
4	1111 √

1_Cube		
0	000X	0,1
	00X0	
1	00X1	1,2
	0X01	
	001X	
2	0X11	2,3
	01X1	
	X101	
3	X111	3,4
	11X1	

QM Method: Combining Adjacent

- Do same combination of comparing adjacent group minterms
  - To form 2-cubes, 3-cubes and so on.

- Only minterms of adjacent groups have the chance of being combined
  - **Which have an X in the same position.**

QM Method

1_Cube	
0	000X √
	00X0 √
1	00X1 √
	0X01 √
	001X √
2	0X11 √
	01X1 √
	X101 √
3	X111 √
	11X1 √

2_Cube	
0	00XX *
1	0XX1 *
2	X1X1 *

Q-M Method: Cover PIs

- PIs : terms left without checkmarks.
- After identifying our PIs, we list them against the minterms needed to be covered

$$\sum m(0,1,2,3,5,7,13,15)$$

	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 1	0 1 1 1	1 1 0 1	1 1 1 1
0 0 x x	√	√	√	√				
0 x x 1		√		√	√	√		
x 1 x 1					√	√	√	√
Func	√	√	√	√	√	√	√	√

QM Method : Covers

- To find a minimal cover, we first need to find essential Pis
- To do this we need to find columns that only have one checkmark in them, the according row will thus show the essential PI.
- After identifying essential PIs, that are necessarily part of the cover, we cover any remaining minterms using a minimal set of PIs.

QM Method : Covers

	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 1	0 1 1 1	1 1 0 1	1 1 1 1
0 0 x x	√	√	√	√				
0 x x 1		√		√	√	√		
x 1 x 1					√	√	√	√
Func	√	√	√	√	√	√	√	√

Essential I

Essential II

Redundant

In this example:

$$F = A'B' + BD$$

Quine-McCluskey (QM) Method
-Example II

QM Method: Another Example

Tabular method to systematically find all prime implicants

$f(A,B,C,D) = \sum m(4,5,6,8,9,10,13) + \sum d(0,7,15)$

Stage 1: Find all prime implicants

Step 1: Fill Column 1 with ON-set and DC-set minterm indices. Group by number of 1's.

Cube0		
0000 √		
0100 √		
1000 √		
0101 √		
0110 √		
1001 √		
1010 √		
0111 √		
1101 √		
1111 √		

Quine-McCluskey Method

Step 2: Apply Uniting Theorem:
Compare elements of group w/ N 1's against those with N+1 1's.
Differ by one bit implies adjacent.

Eliminate variable and place in next column.
E.g., 0000 vs. 0100 yields 0-00
0000 vs. 1000 yields -000

When used in a combination, mark with a check. If cannot be combined, mark with a star. These are the prime implicants.

Repeat until no further combinations can be made.

Cube0	Cube 1	Cube2
0000 √	0x00 *	
0100 √	x000 *	
1000 √	010x √	
0101 √	01x0 √	
0110 √	100x *	
1001 √	10x0 *	
1010 √	01x1 √	
0111 √	x101 √	
1101 √	011x √	
1111 √	1x01 *	
	x111 √	
	11x1 √	

Quine Mcluskey Method

Step 2: Apply Uniting Theorem:
Compare elements of group w/ N 1's against those with N+1 1's.
Differ by one bit implies adjacent.

Eliminate variable and place in next column.
E.g., 0000 vs. 0100 yields 0-00
0000 vs. 1000 yields -000

When used in a combination, mark with a check. If cannot be combined, mark with a star. These are the prime implicants.

Repeat until no further combinations can be made.

Cube0	Cube 1	Cube2
0000 √	0x00 *	01xx *
0100 √	x000 *	x1x1 *
1000 √	010x √	
0101 √	01x0 √	
0110 √	100x *	
1001 √	10x0 *	
1010 √	01x1 √	
0111 √	x101 √	
1101 √	011x √	
1111 √	1x01 *	
	x111 √	
	11x1 √	

Finding the Minimum Cover

- We have so far found all the prime implicants
- 2nd step of the Q-M procedure is to find the smallest set of prime implicants to cover the complete on-set of the function

Finding the Minimum Cover

- This is accomplished through the prime implicant chart
  - Columns are labeled with the minterm indices of the onset
  - Rows are labeled with the minterms covered by a given prime implicant
  - Example a prime implicant (-1-1) becomes minterms 0101, 0111, 1101, 1111, which are indices of minterms m5, m7, m13, m15

Coverage Table/ Chart

	4	5	6	8	9	10	13
0,4(0-00)	X						
0,8(-000)				X			
8,9(100-)				X	X		
8,10(10-0)				X		X	
9,13(1-01)					X		X
4,5,6,7(01--)	X	X	X				
5,7,13,15(-1-1)		X					X

Note: Don't include DCs in coverage table; they don't have covered by the final logic expression!

rows = prime implicants
columns = ON-set elements
place an "X" if ON-set element is
covered by the prime implicant

Coverage Table/ Chart

	4	5	6	8	9	10	13
0,4(0-00)	X						
0,8(-000)				X			
8,9(100-)				X	X		
8,10(10-0)				X		X	
9,13(1-01)					X		X
4,5,6,7(01--)	X	X	X				
5,7,13,15(-1-1)		X					X

rows = prime implicants
columns = ON-set elements
place an "X" if ON-set element is
covered by the prime implicant

If column has a single X, than the implicant associated with the row is essential. It must appear in minimum cover

Coverage Table/ Chart: Eliminate

	4	5	6	8	9	10	13
0,4(0-00)	X						
0,8(-000)				X			
8,9(100-)				X	X		
8,10(10-0)				X		X	
9,13(1-01)					X		X
4,5,6,7(01--)	X	X	X				
5,7,13,15(-1-1)		X					X

rows = prime implicants
columns = ON-set elements
place an "X" if ON-set element is
covered by the prime implicant

If column has a single X, than the implicant associated with the row is essential. It must appear in minimum cover

Eliminate all columns covered by essential primes

Coverage Table/ Chart: Eliminate

	4	5	6	8	9	10	13
0,4(0-00)	X						
0,8(-000)				X			
8,9(100-)				X	X		
8,10(10-0)				X		X	
9,13(1-01)					X		X
4,5,6,7(01--)	X	X	X				
5,7,13,15(-1-1)		X					X

rows = prime implicants
columns = ON-set elements
place an "X" if ON-set element is
covered by the prime implicant

Find minimum set of rows that cover the remaining columns

Eliminate all columns covered by essential primes

Coverage Table/ Chart: Eliminate

	4	5	6	8	9	10	13
0,4(0-00)	X						
0,8(-000)				X			
8,9(100-)				X	X		
8,10(10-0)				X		X	
9,13(1-01)					X		X
4,5,6,7(01--)	X	X	X				
5,7,13,15(-1-1)		X					X

If all are covered:
Write the Implicants

F= AB'D' + AC'D + A'B

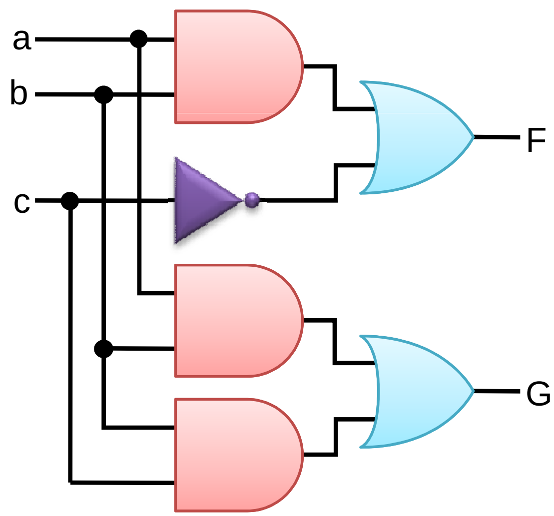
Design
of
Combinational Circuit Block

Study of Components

- Decoder, Encoder
- Multiplexor
- Logic Implementation Using MUX & Decoder
- Mux: 7 Segment Display
- 4 Bit Adder
- N- Bit Adder

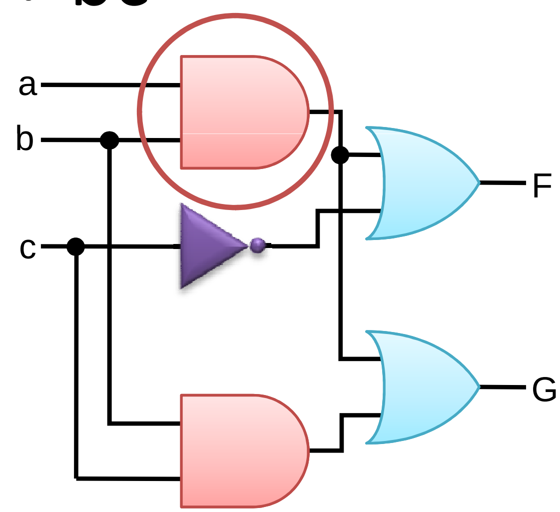
Multiple-Output Circuits

- Many circuits have more than one output
- Can give each a separate circuit, or can share gates
- Ex: F = ab + c’, G = ab + bc



(a)

Option 1: Separate circuits



(b)

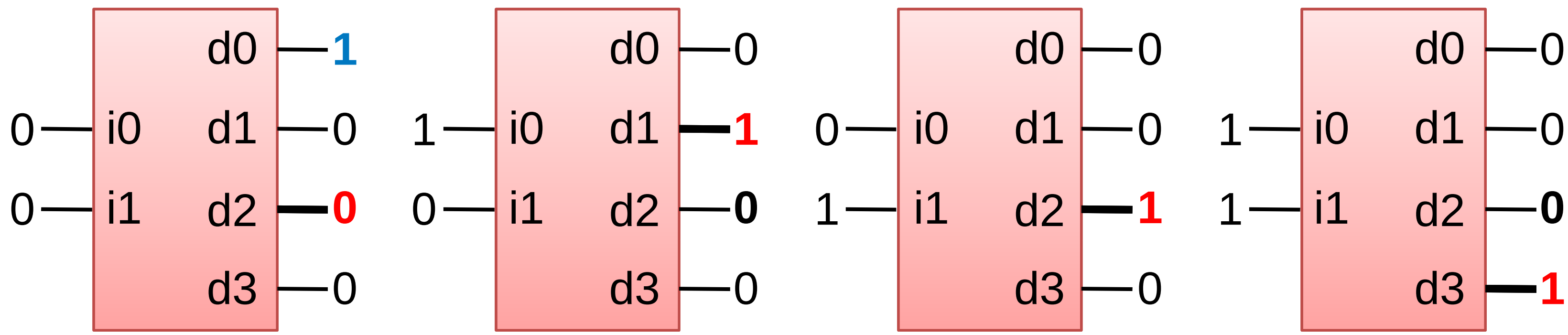
Option 2: Shared gates

Decoder

- Reception counter : When you reach a Academic Institute
  - Receptionist Ask: Which Dept to Go ?
  - Customer : CSE
  - Receptionist Redirect you to some building according to your Answer. == > Go to Core II
- Decoder : knows what to do with this: Decode
- Digital Case: == > N input: 2^N output
- Memory Addressing
  - Address to a particular location

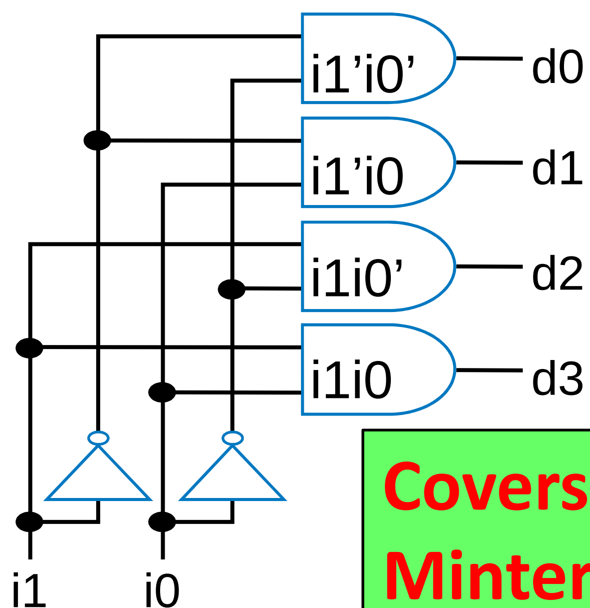
Decoders

- **Decoder**: Popular combinational logic building block, in addition to logic gates
  - Converts input binary number to one high output
- 2-input decoder: four possible input binary numbers
  - So has four outputs, one for each possible input binary number

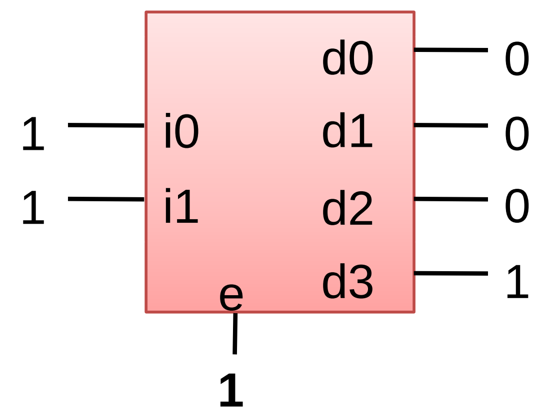


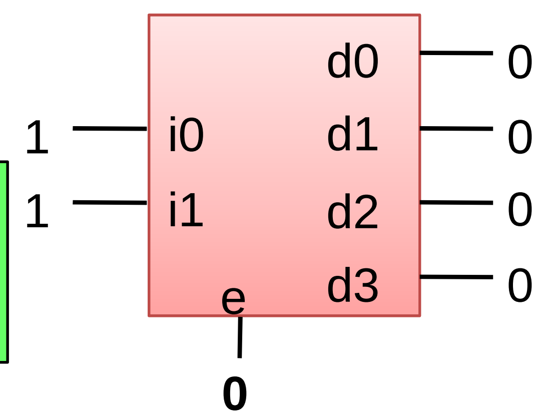
Decoders and Muxes

- Internal design
  - AND gate for each output to detect input combination
- Decoder with enable e
  - Outputs all 0 if e=0, Regular behavior if e=1
- n-input decoder: 2^n outputs

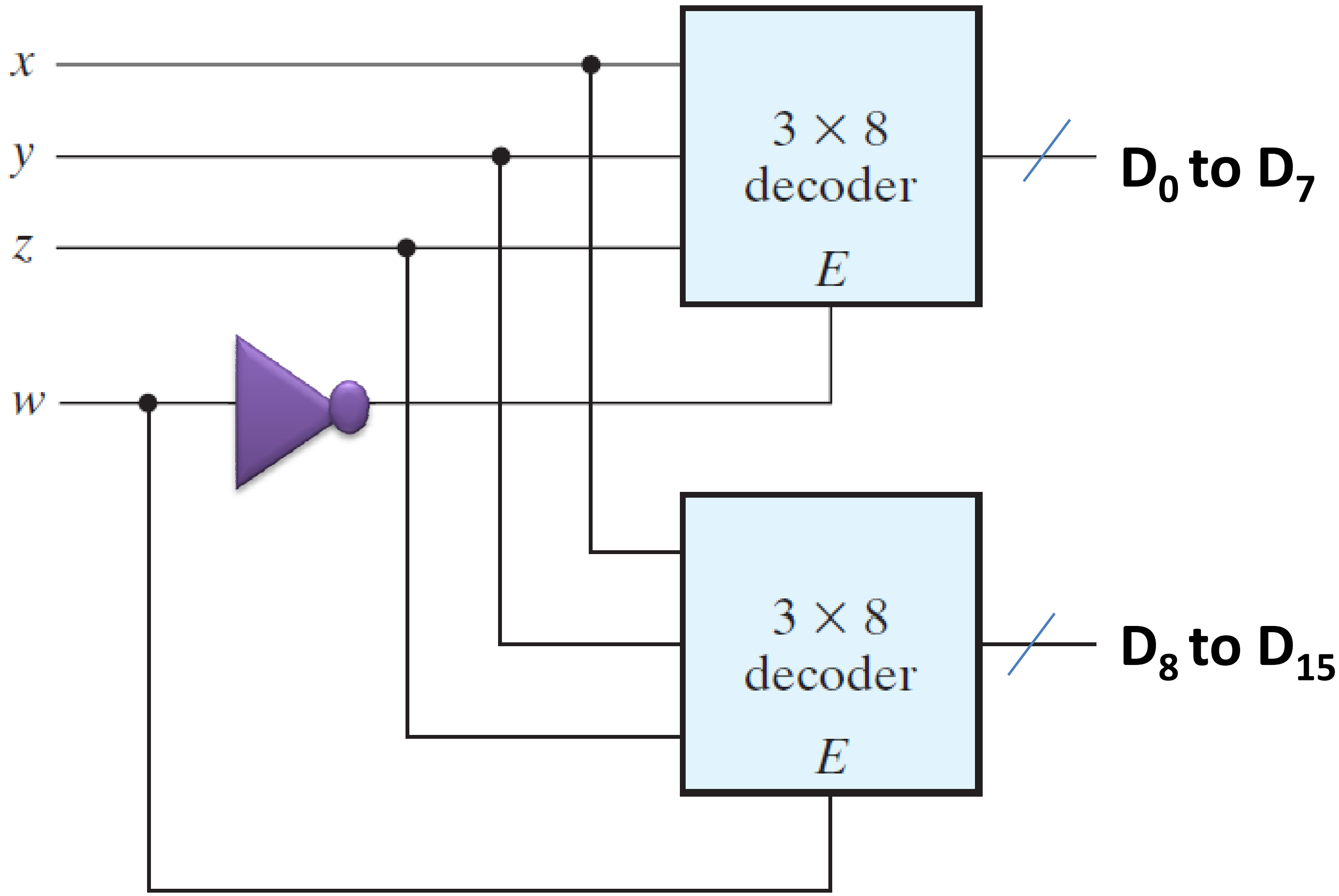


Covers All Minterms





4-to-16 Decoder using two 3-to-8 Decoders



Boolean Function Implementation using Decoders

- **As Decoder covers all the Minterms**
- Using a n-to-2n decoder and OR gates any functions of n variables can be implemented.
- Example: Full Adder

  $S(x,y,z)= \Sigma(1,2,4,7)$, $C(x,y,z)=\Sigma(3,5,6,7)$
- Functions S and C can be implemented using a 3-to-8 decoder and two 4-input OR gates

Decoder: Covers All Minterms

Implementation of S and C