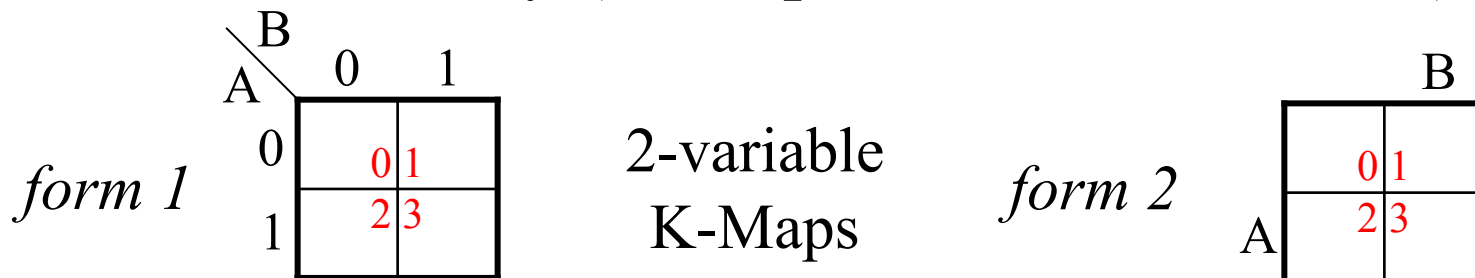


# Karnaugh Maps (K-map)

- Alternate representation of a truth table
  - Red decimal = minterm value
    - Note that A is the MSB for this minterm numbering
  - Adjacent squares have distance = 1
- Valuable tool for logic minimization
  - Applies most Boolean theorems & postulates automatically (when procedure is followed)



# Karnaugh Maps (K-map)

- Alternate forms of 3-variable K-maps

- Note end-around adjacency

- Distance = 1
    - Note: A is MSB, C is LSB for minterm numbering

*form 1*

		BC			
		00	01	11	10
A	0	0 1		3 2	
	1	4 5		7 6	

*form 2*

		B			
		0 1		3 2	
A	0	0 1		3 2	
	1	4 5		7 6	
		C			

		C	
		0	1
AB	00	0 1	
	01	2 3	
	11		6 7
	10	4 5	

		C		B
		0 1		
A	0	0 1		
	1	4 5		
	0		6 7	
	1		2 3	

# K-mapping & Minimization Steps

Step 1: generate K-map

- Put a 1 in all specified minterms
- Put a 0 in all other boxes (optional)

Step 2: group all adjacent 1s without including any 0s

- All groups (aka *prime implicants*) must be rectangular and contain a “power-of-2” number of 1s
  - 1, 2, 4, 8, 16, 32, ...
- An essential group (aka *essential prime implicant*) contains at least 1 minterm not included in any other groups
  - A given minterm may be included in multiple groups

Step 3: define product terms using variables common to all minterms in group

Step 4: sum all essential groups plus a minimal set of remaining groups to obtain a minimum SOP

# K-map Minimization Example

- $Z = \sum_{A,B,C}(1,3,6,7)$

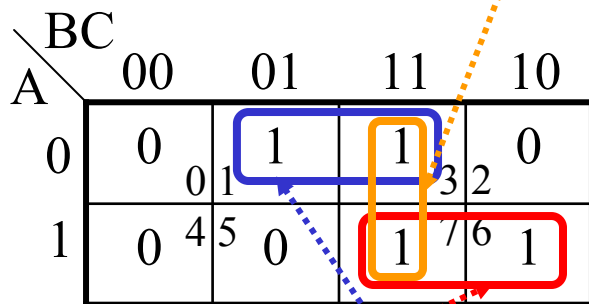
- Recall SOP minterm implementation

- 8 gates
- 27 gate I/O

- K-map results

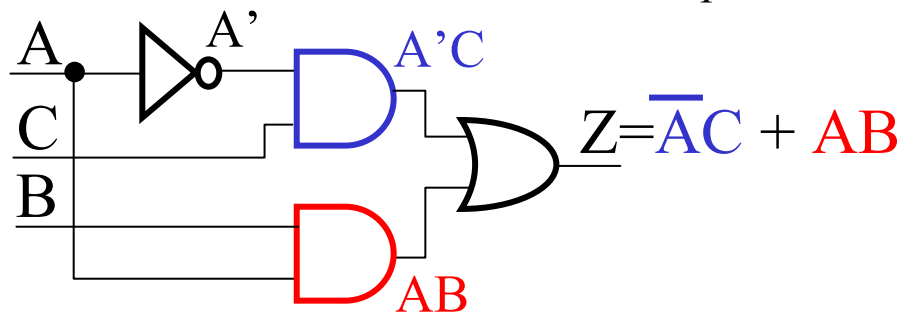
- 4 gates
- 11 gate I/O

*Note: this group not needed since 1s are already covered*



*essential prime implicants*

A	B	C	Z	Row value
0	0	0	0	0
0	0	1	1	1
0	1	0	0	2
0	1	1	1	3
1	0	0	0	4
1	0	1	0	5
1	1	0	1	6
1	1	1	1	7



# K-map Minimization Goals

- Larger groups:
  - Smaller product terms
    - Fewer variables in common
  - Smaller AND gates
    - In terms of number of inputs
- Fewer groups:
  - Fewer product terms
    - Fewer AND gates
    - Smaller OR gate
      - ✓ In terms of number of inputs
- Alternate method:
  - Group 0s
    - Could produce fewer and/or smaller product terms
  - Invert output
    - Use NOR instead of OR gate



# 5-variable K-map

- Note adjacency between maps when overlaid
  - distance=1
- Variable order for this minterm numbering:
  - A,B,C,D,E (A is MSB, E is LSB)

BC \ DE	00	01	11	10
00	0 1		3 2	
01	4 5		7 6	
11				
10	12 13		15 14	
	8 9		11 10	

A=0

BC \ DE	00	01	11	10
00	16 17		19 18	
01	20 21		23 22	
11				
10	28 29		31 30	
	24 25		27 26	

A=1

# 5-variable K-map

- Changing the variable used to separate maps changes minterm numbering
- Same variable order for this minterm numbering:
  - A,B,C,D,E (A is MSB, E is LSB)

AB \ CD	00	01	11	10
00	0 2		6 4	
01	8 10		14 12	
11				
10	24 26		30 28	
10	16 18		22 20	

E=0

AB \ CD	00	01	11	10
00	1 3		7 5	
01	9 11		15 13	
11				
10	25 27		31 29	
10	17 19		23 21	

E=1

# 6-variable K-map

- Variable order for minterm numbers: ABCDEF

A=0

CD \ EF	00	01	11	10
00	01		32	
01	45		76	
11	1213		1514	
10	89		1110	

CD \ EF	00	01	11	10
00	1617		1918	
01	2021		2322	
11	2829		3130	
10	2425		2726	

A=1

CD \ EF	00	01	11	10
00	3233		3534	
01	3637		3938	
11	4445		4746	
10	4041		4342	

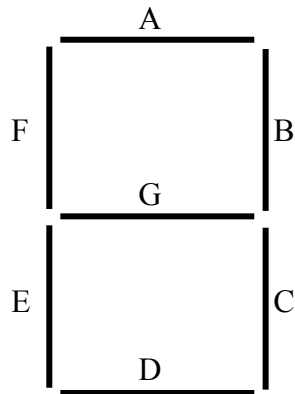
CD \ EF	00	01	11	10
00	4849		5150	
01	5253		5554	
11	6061		6362	
10	5657		5958	

B=0

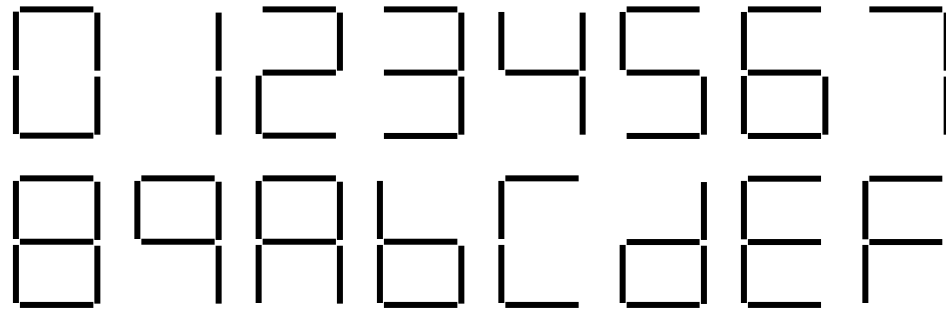
B=1

# Design Example

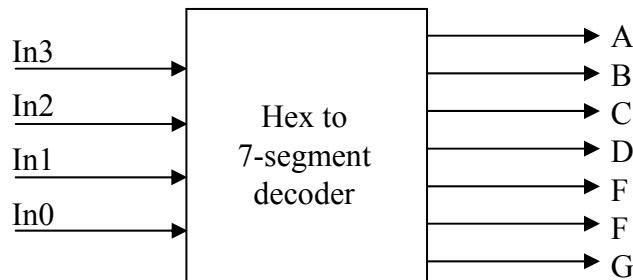
- Hexadecimal to 7-segment display decoder
  - A common circuit in calculators
  - 7-segments (A-G) to represent digits (0-9 & A-F)
    - A logic 1 turns on given segment



7 segments



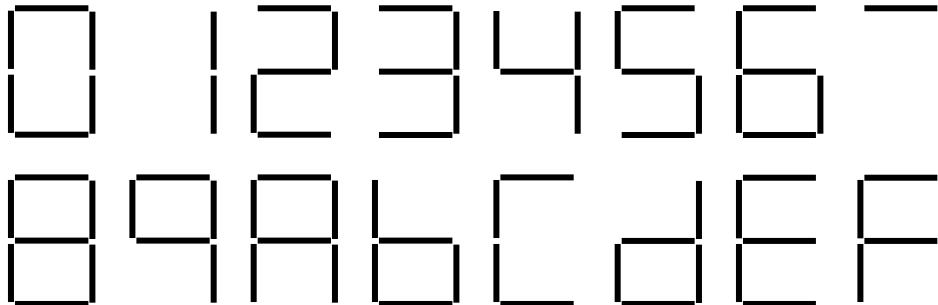
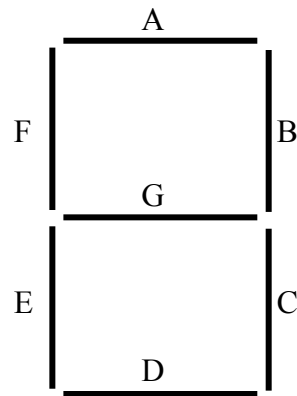
active (on) segments  
for a given HEX value



Circuit block diagram

# HEX to 7-seg Design Example

- Create truth table from specification

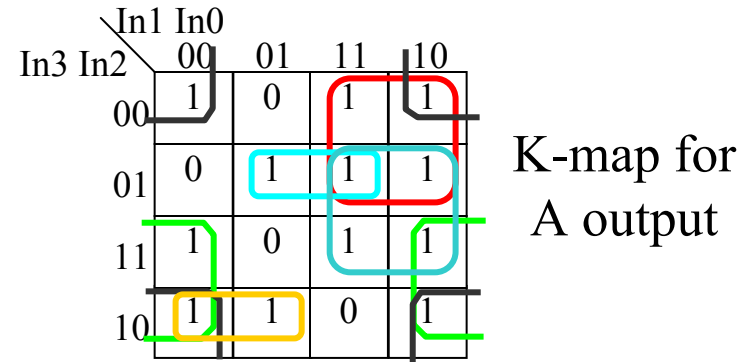


In3	In2	In1	In0	A	B	C	D	E	F	G
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1

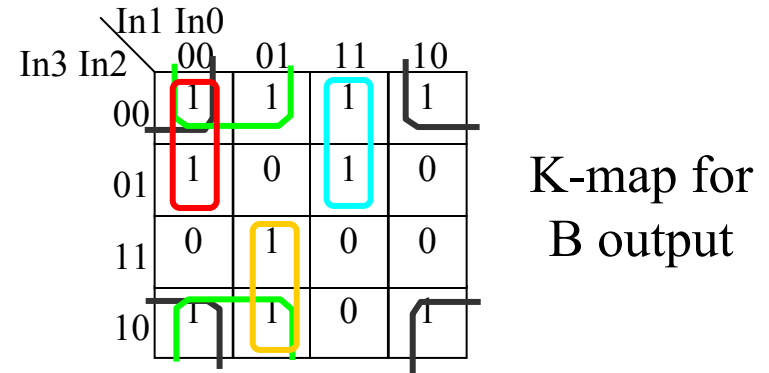
# HEX to 7-seg Design Example

- Generate K-maps & obtain logic equations

In3	In2	In1	In0	A	B	C	D	E	F	G
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1



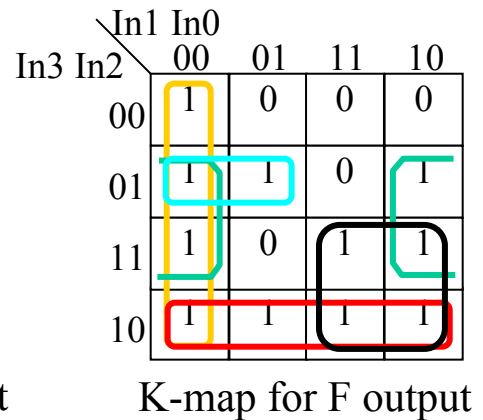
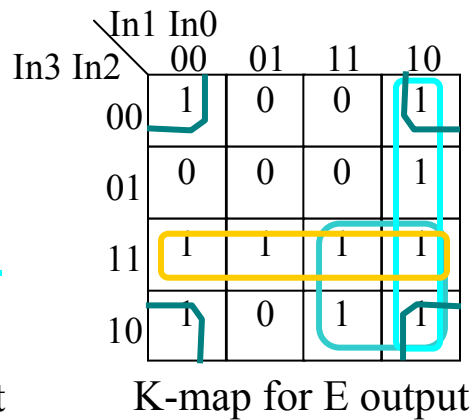
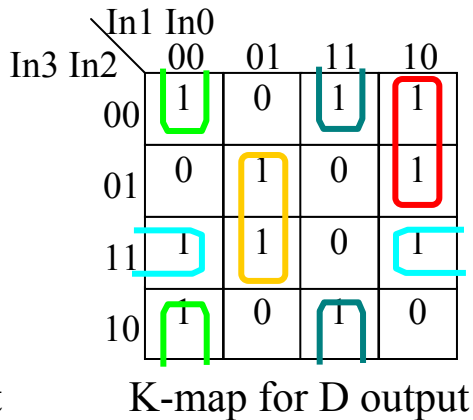
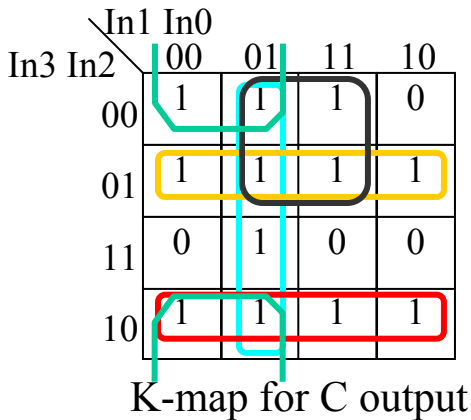
$$A = \text{In2}'\text{In0}' + \text{In3}'\text{In1} + \text{In2 In1} + \text{In3 In0}' + \text{In3}'\text{In2 In0} + \text{In3 In2}'\text{In1}'$$



$$B = \text{In2}'\text{In0}' + \text{In2}'\text{In1}' + \text{In3}'\text{In1}'\text{In0}' + \text{In3 In1}'\text{In0} + \text{In3}'\text{In1 In0}$$

# HEX to 7-seg Design Example

- K-maps & logic equations for outputs C-G



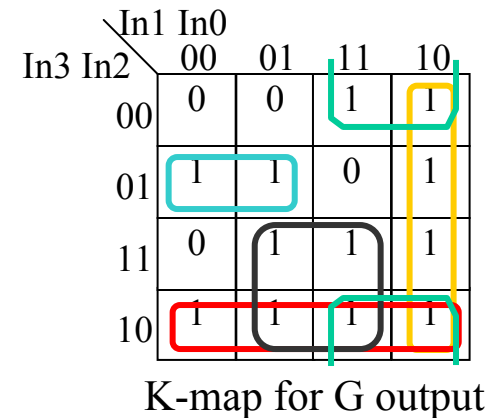
$$C = \text{In3 In2}' + \text{In1}'\text{In0} + \text{In2}'\text{In1}' + \text{In3}'\text{In0} + \text{In3}'\text{In2}$$

$$D = \text{In3 In2 In0}' + \text{In2}'\text{In1}'\text{In0}' + \text{In2 In1}'\text{In0} + \text{In2}'\text{In1 In0} + \text{In3}'\text{In1 In0}'$$

$$E = \text{In2}'\text{In0}' + \text{In3 In2} + \text{In1 In0}' + \text{In3 In1}$$

$$F = \text{In1}'\text{In0}' + \text{In3 In2}' + \text{In2 In0}' + \text{In3 In1} + \text{In3}'\text{In2 In1}'$$

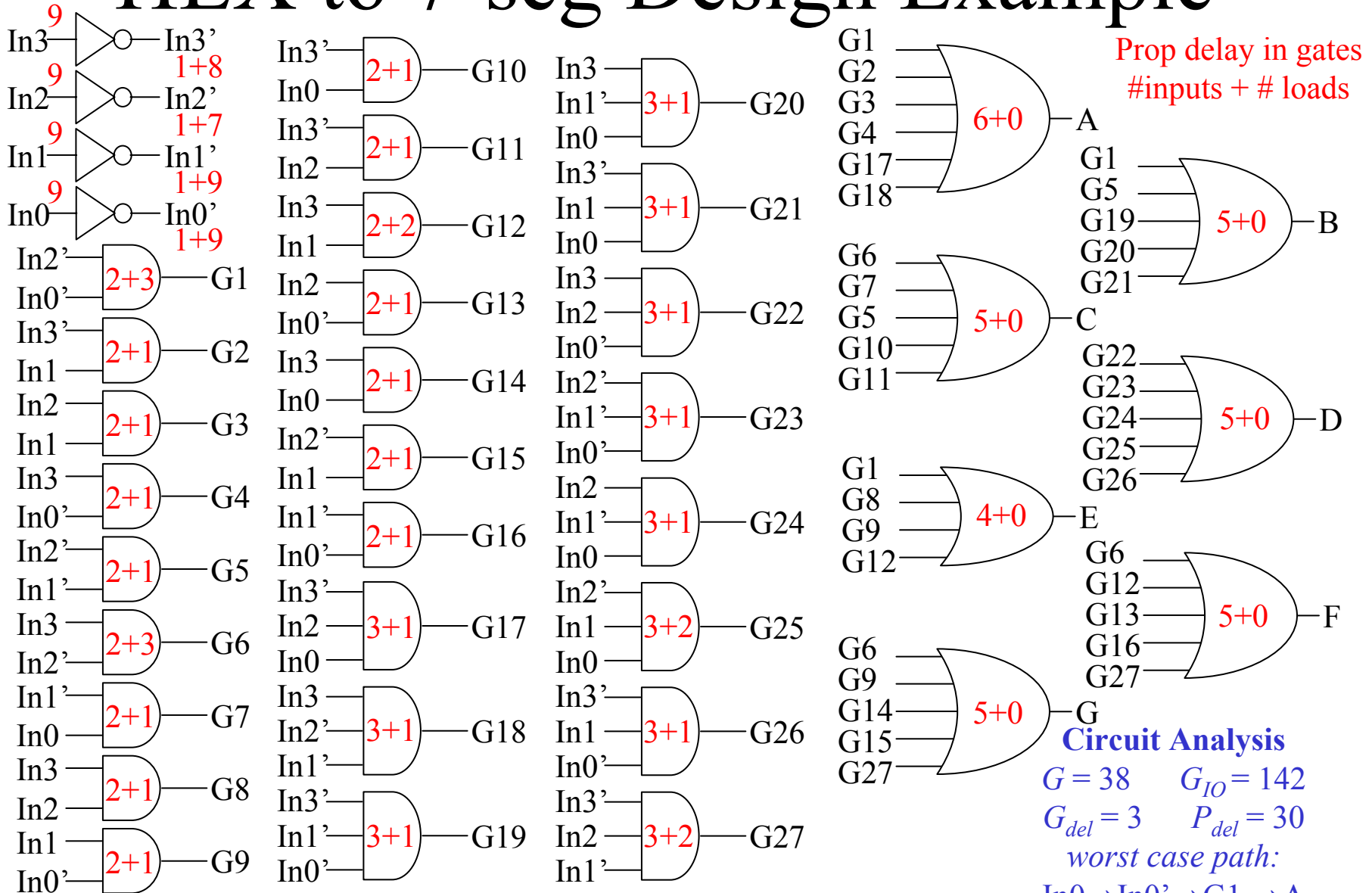
$$G = \text{In3 In2}' + \text{In1 In0}' + \text{In3 In0} + \text{In3}'\text{In2 In1}' + \text{In2}'\text{In1}$$



# HEX to 7-seg Design Example

- Remaining steps to complete design:
  - Draw logic diagram (sharing common gates)
    - Analyze for optimization metrics:  $G$ ,  $G_{IO}$ ,  $G_{del}$ ,  $P_{del}$ 
      - ✓ See next page for logic diagram & circuit analysis
  - Simulate circuit for design verification
    - Debug & fix problems when output is incorrect
      - ✓ Check truth table against K-map population
      - ✓ Check K-map groups against logic equation product terms
      - ✓ Check logic equations against schematic
  - Optimize circuit for area and/or performance
    - Use Boolean postulates & theorems
  - Re-simulate & verify optimized design

# HEX to 7-seg Design Example



C. E. Stroud

Combinational Logic Minimization

(1/06)

**Circuit Analysis**  
 $G = 38$      $G_{IO} = 142$   
 $G_{del} = 3$      $P_{del} = 30$   
 worst case path:  
 $In_0 \rightarrow In_0' \rightarrow G1 \rightarrow A$

# Don't Care Conditions

- Sometimes input combinations are of no concern
  - Because they may not exist
    - Example: BCD uses only 10 of possible 16 input combinations
  - Since we “don't care” what the output, we can use these “don't care” conditions for logic minimization
    - The output for a don't care condition can be either 0 or 1
      - ✓ WE DON'T CARE!!!
- Don't Care conditions denoted by:
  - X, -, d, 2
    - X is probably the most often used
- Can also be used to denote inputs
  - Example:  $ABC = 1X1 = AC$ 
    - B can be a 0 or a 1

# Don't Care Conditions

- Truth Table

- K-map

- Minterm

➤  $Z = \sum_{A,B,C}(1,3,6,7) + d(2)$

		BC			
		00	01	11	10
A	0	0	1	1	X
	1	0	0	1	1

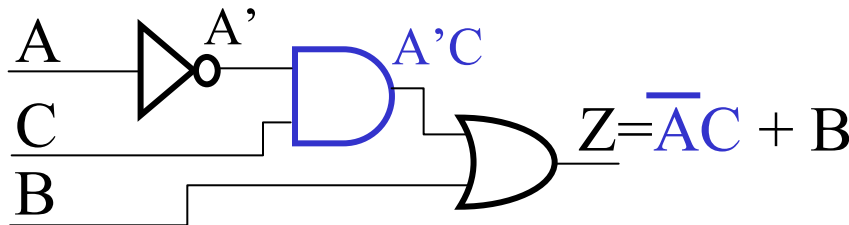
A	B	C	Z
0	0	0	0
0	0	1	1
0	1	0	X
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

- Maxterm

➤  $Z = \prod_{A,B,C}(0,4,5) + d(2)$

$Z = B + A'C$

- Notice Don't Cares are same for both minterm & maxterm



**Circuit analysis:**

$G=3 \quad G_{IO}=8$

(compared to  $G=4$  &  $G_{IO}=11$  w/o don't care)