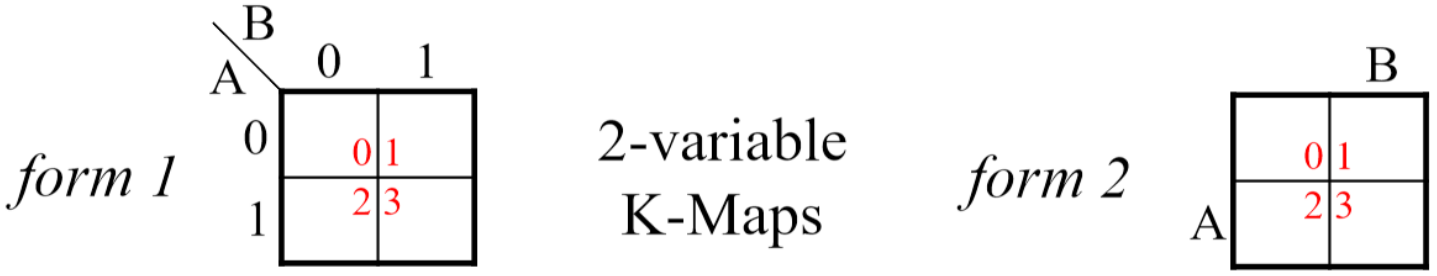
Karnaugh Maps (K-map)

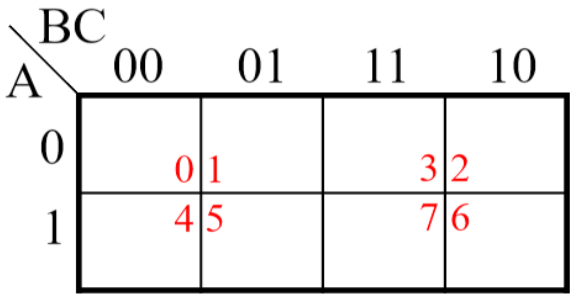
- Alternate representation of a truth table
  - ➢ Red decimal = minterm value
    - Note that A is the MSB for this minterm numbering
  - ➢ Adjacent squares have distance = 1
- Valuable tool for logic minimization
  - ➢ Applies most Boolean theorems & postulates automatically (when procedure is followed)

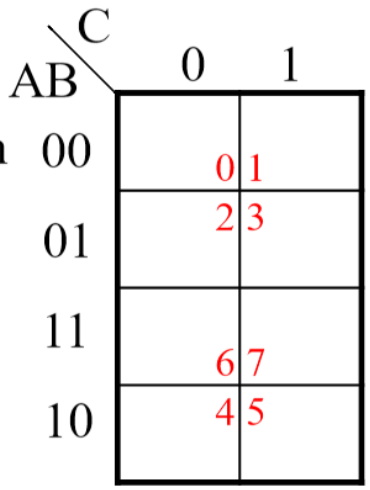


Karnaugh Maps (K-map)

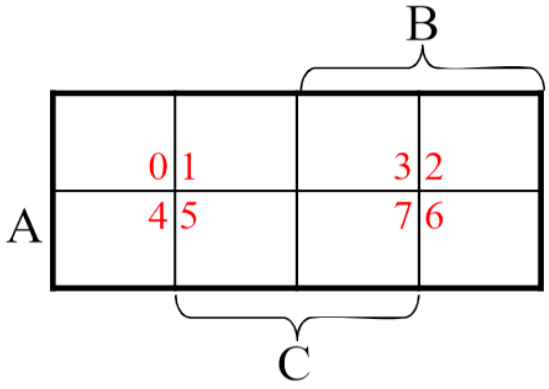
- Alternate forms of 3-variable K-maps
  - ➢ Note end-around adjacency
    - Distance = 1
    - Note: A is MSB, C is LSB for minterm numbering

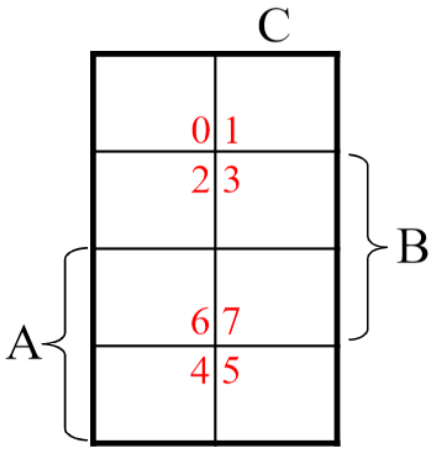
form 1





form 2





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

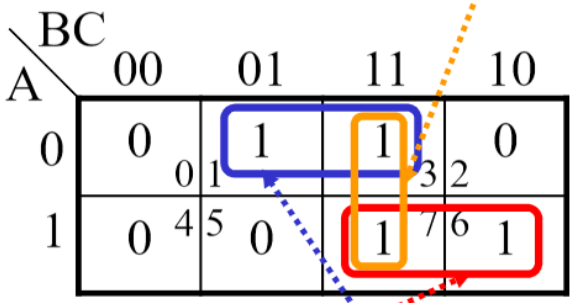
- All groups (aka *prime implicants*) must be rectangular and contain a "power-of-2" number of 1s
  - 1, 2, 4, 8, 16, 32, …
- An essential group (aka *essential prime implicant*) contains at least 1 minterm not included in any other groups
  - A given minterm may be included in multiple groups

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

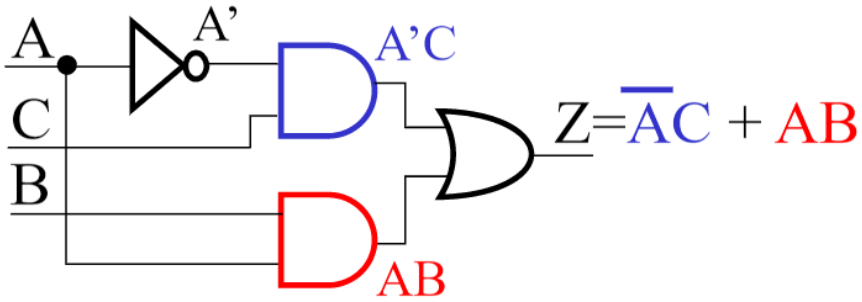
- $Z=\Sigma_{A,B,C}(1,3,6,7)$
  - Recall SOP minterm implementation
    - 8 gates
    - 27 gate I/O
  - K-map results
    - 4 gates
    - 11 gate I/O



essential prime implicants

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

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

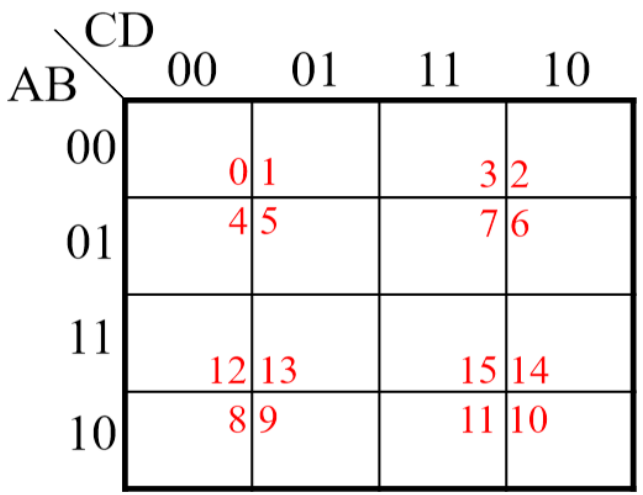


K-map Minimization Goals

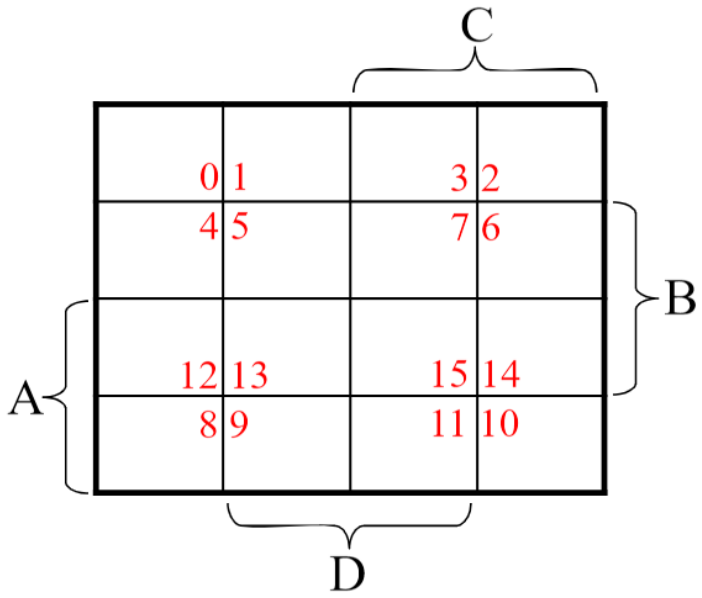
- Larger groups:
  - ➢ Smaller product terms
    - Fewer variables in common
  - ➢ Smaller AND gates
    - In terms of number of inputs
- Fewer groups:
  - ➢ Fewer product terms
    - Fewer AND gates
    - Smaller OR gate
      - ✓ In terms of number of inputs
- Alternate method:
  - ➢ Group 0s
    -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

4-variable K-maps

- Note adjacency of 4 corners as well as sides
- Variable ordering for this minterm numbering: ABCD



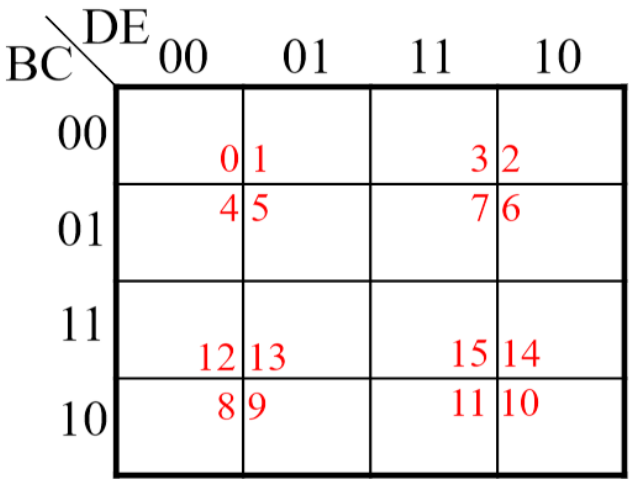
form 1



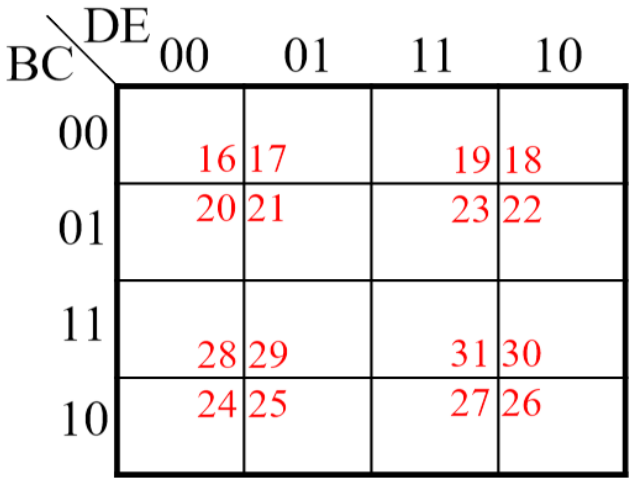
form 2

5-variable K-map

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



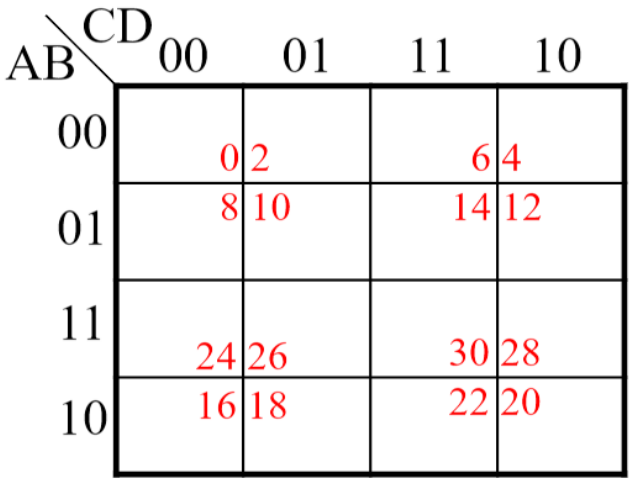
A=0



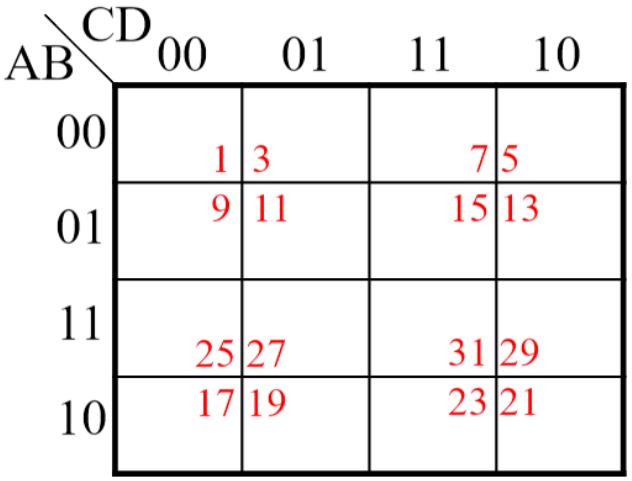
A=1

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)



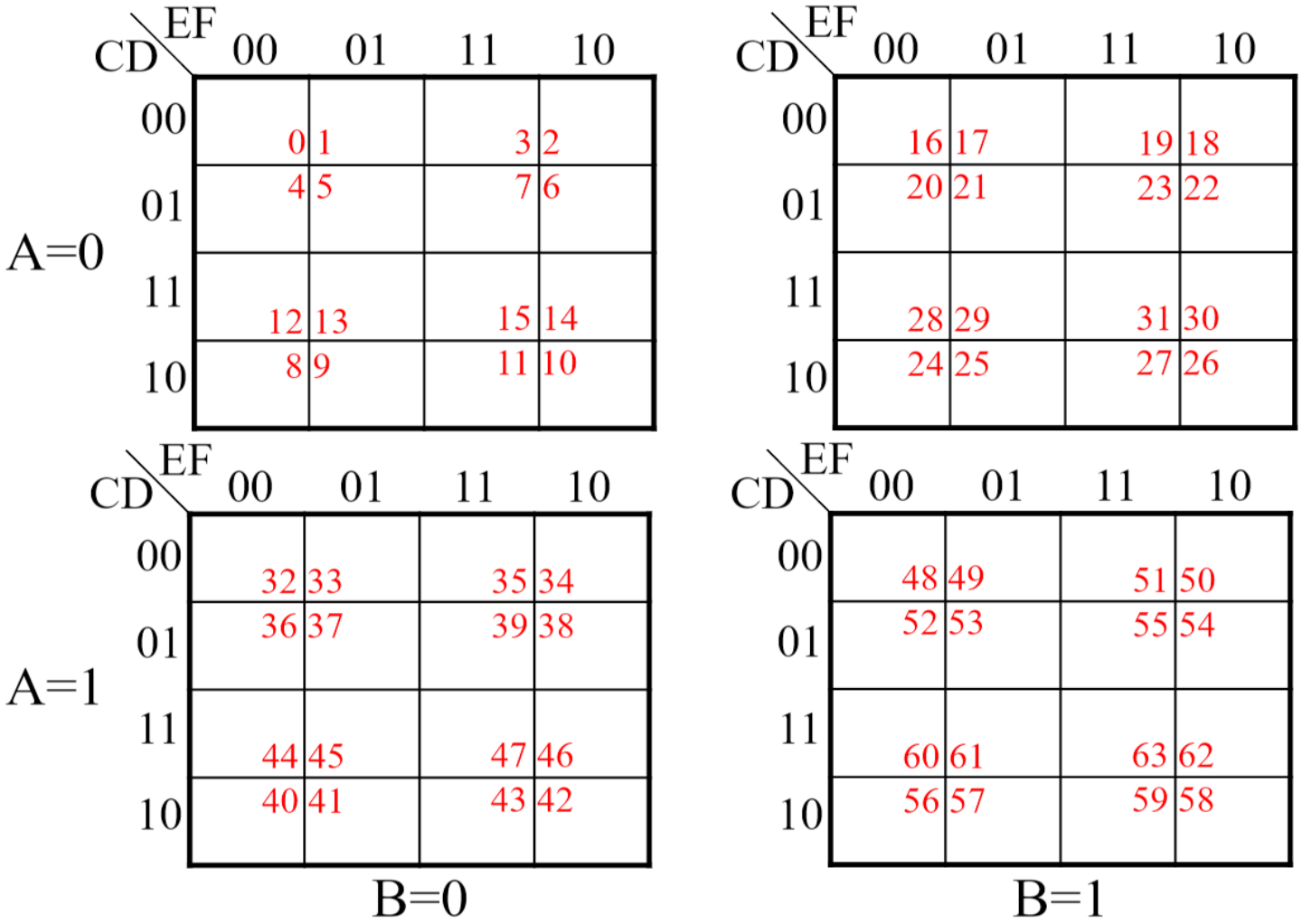
E=0



E=1

6-variable K-map

- Variable order for minterm numbers: ABCDEF

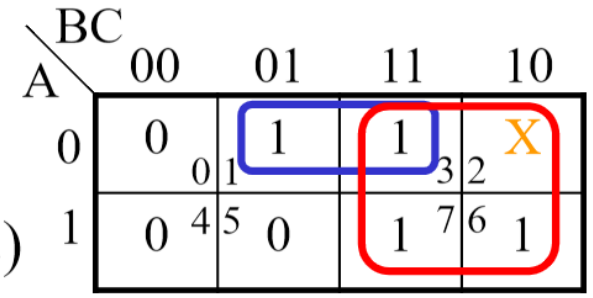


Don't Care Conditions

- 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

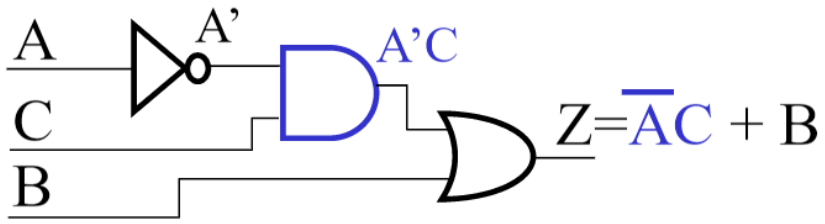
Don't Care Conditions

- Truth Table
- K-map
- Minterm
  - $Z=\Sigma_{A,B,C}(1,3,6,7)+d(2)$
- Maxterm
  - $Z=\Pi_{A,B,C}(0,4,5)+d(2)$
    - Notice Don't Cares are same for both minterm & maxterm



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

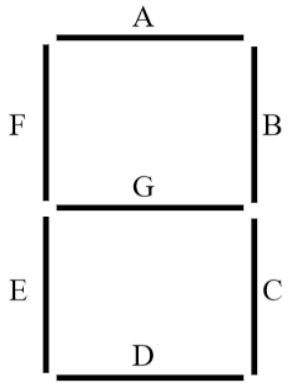
$Z=B+A'C$



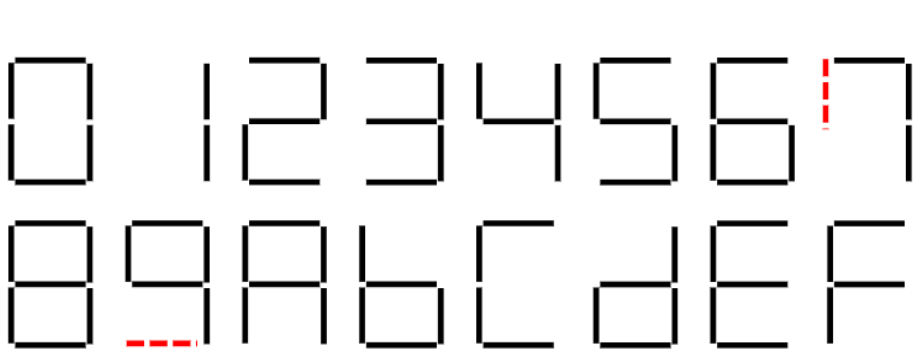
Circuit analysis:
G=3 G_{IO}=8
(compared to G=4 & G_{IO}=11 w/o don't care)

Design Example

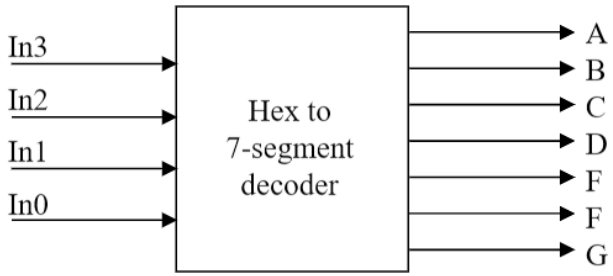
- Hexadecimal to 7-segment display decoder
  - A common circuit in calculators
  - 7-segments (A-G) to represent digits (0-9 & A-F)
    - A logic 1 turns on given segment



7 segments



active (on) segments for a given HEX value

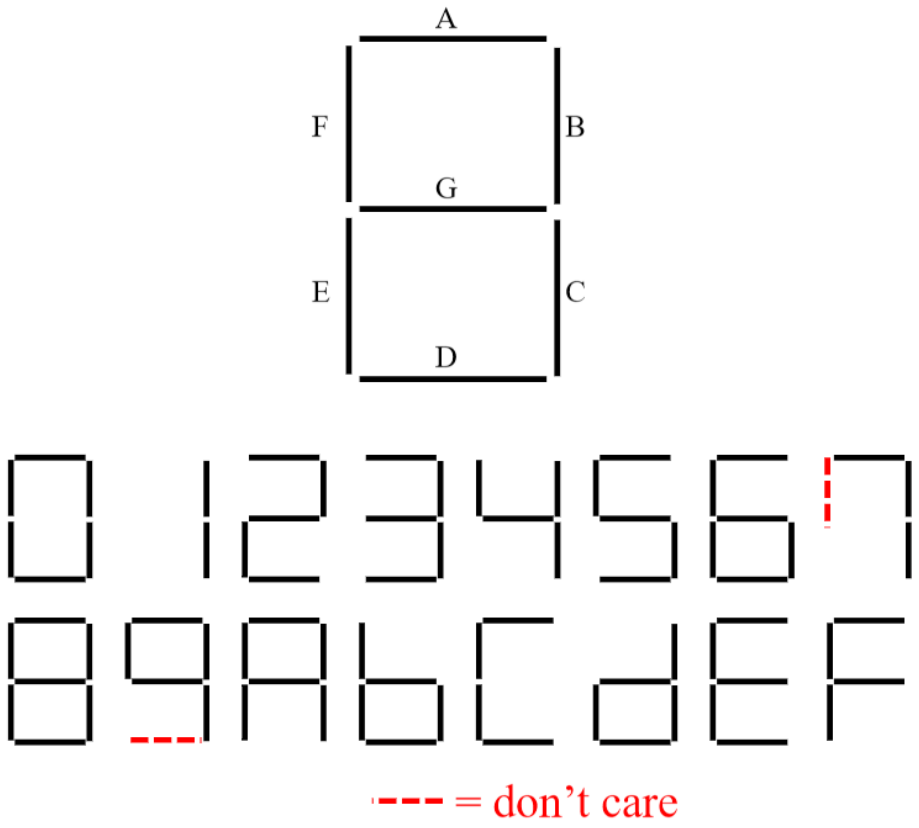


= don't care

Circuit block diagram

HEX to 7-seg Design Example

- Create truth table from specification

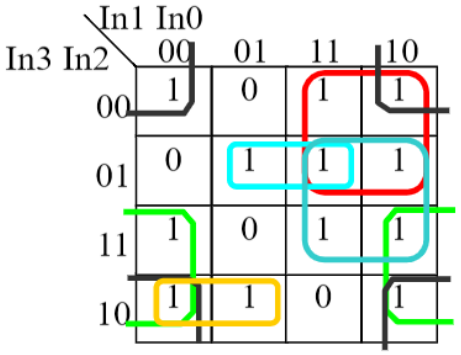


In3	In2	In1	In0	A	B	C	D	E	F	G
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	X	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	X	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0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

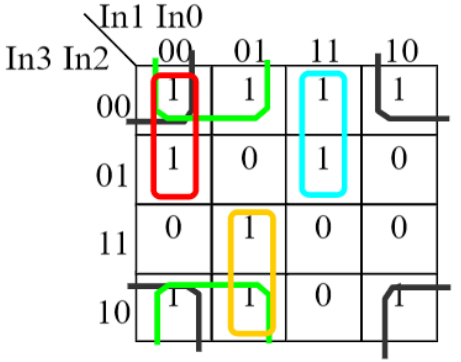
- Generate K-maps & obtain logic equations

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



K-map for A output

A = In2'In0' + In3'In1 + In2 In1 + In3 In0' + In3'In2 In0 + In3 In2'In1'

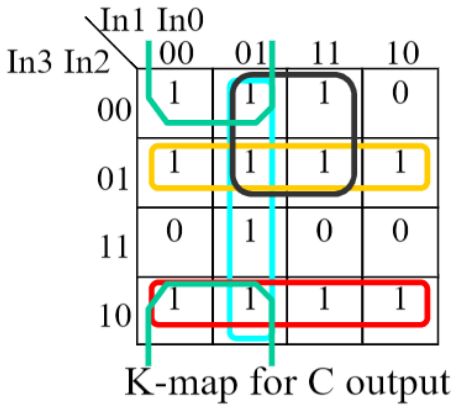


K-map for B output

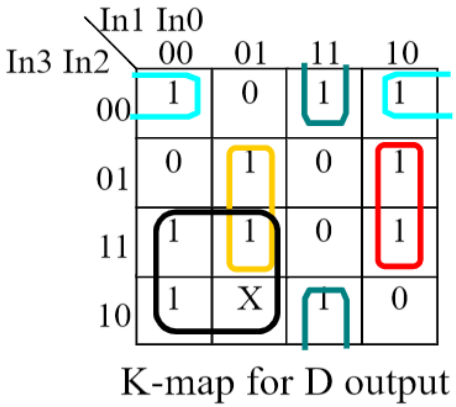
B = In2'In0' + In2'In1' + In3'In1'In0' + In3 In1'In0 + In3'In1 In0

HEX to 7-seg Design Example

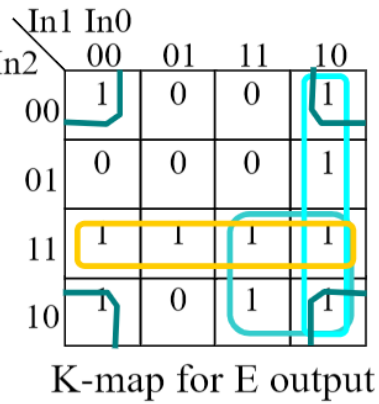
- K-maps & logic equations for outputs C-G



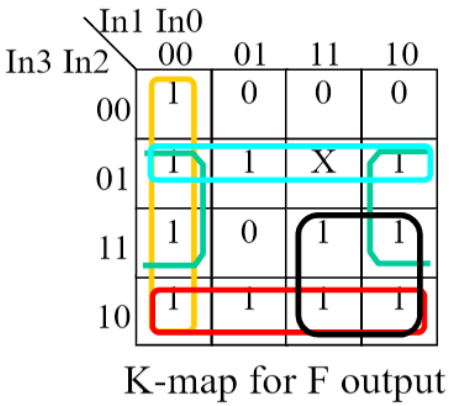
K-map for C output



K-map for D output



K-map for E output



K-map for F output

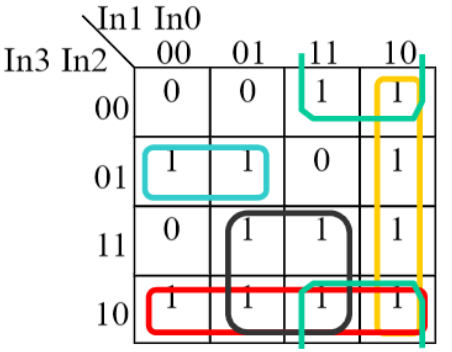
C = In3 In2' + In1'In0 + In2'In1' + In3'In0 + In3'In2

D = In3'In2'In0' + In2'In1 In0 + In2 In1'In0
+ In3 In1' + In2 In1 In0'

E = In2'In0' + In3 In2 + In1 In0' + In3 In1

F = In1'In0' + In3 In2' + In2 In0' + In3 In1 + In3'In2

G = In3 In2' + In1 In0' + In3 In0 + In3'In2 In1' + In2'In1

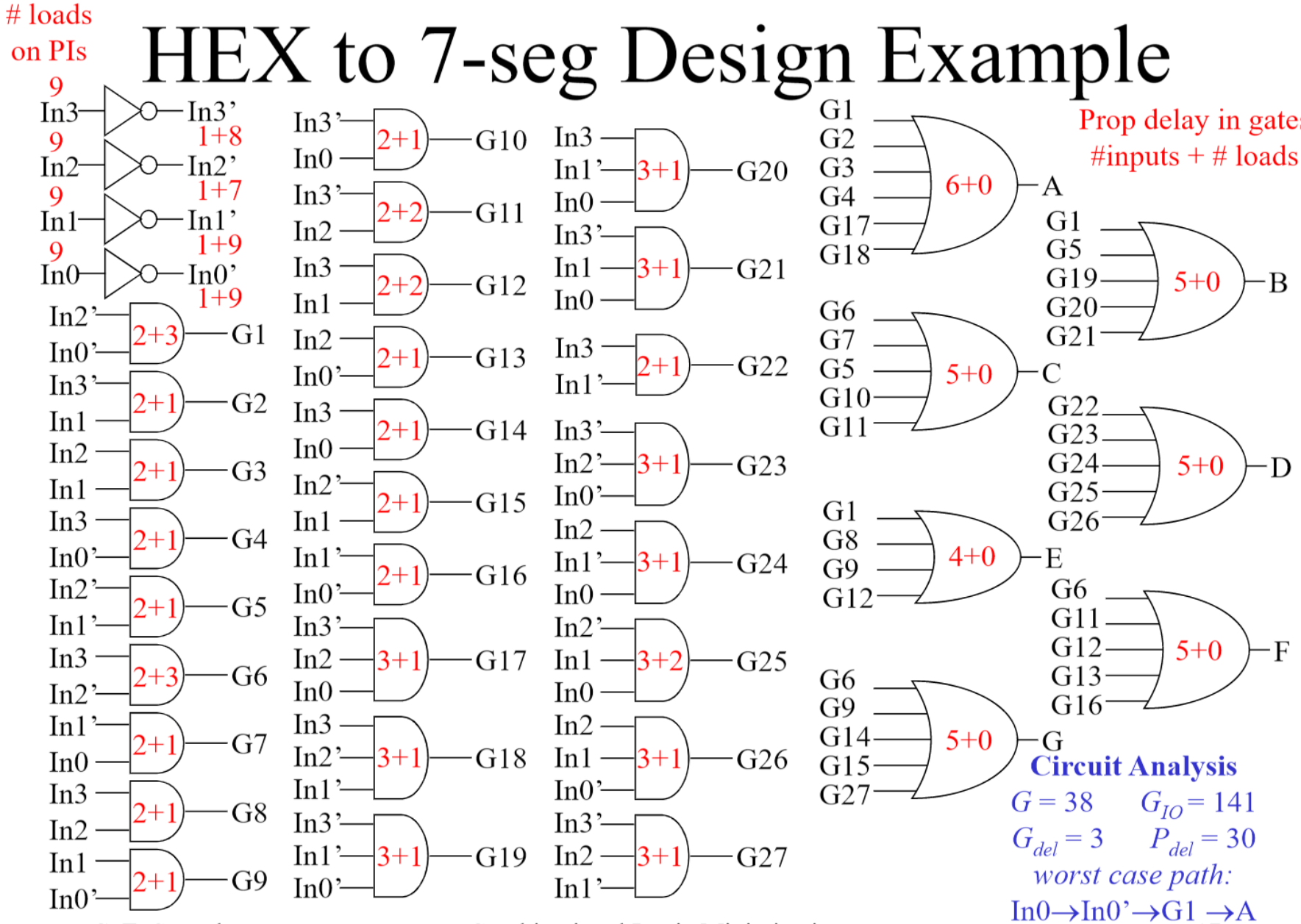


K-map for G output

HEX to 7-seg Design Example

- Remaining steps to complete design:
  - ➢ Draw logic diagram (sharing common gates)
    - Analyze for optimization metirc: 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



Circuit Analysis

$G = 38$ $G_{IO} = 141$

$G_{del} = 3$ $P_{del} = 30$

worst case path:

In0→In0'→G1→A