Lecture 9

- Logistics
  - HW3 posted, due on Friday
  - Lab3 this week
- Last lecture
  - Verilog
- Last last lecture
  - K-maps examples
  - K-map high dimension example
  - Don't cares
- Today
  - Don't care (review)
  - POS minimization with K-map
  - Design examples with K-map

The "WHY" slide

- Don't cares
  - Sometimes the logic output doesn't matter. When we don't care if the output is 0 or 1, rather than assigning random outputs, it is best to denote it as "Don't care." If you learn how to use the "don't care's", you will be able to build even more efficient circuits than without them.
- Design examples with K-map
  - Doing K-map is fun, but when it is combined with an actual design problem you will see how k-map fits into the whole scheme of logic design.

Revisit Don't cares example: Truth table for a BCD increment-by-1

INPUTS				OUTPUTS			
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

- Function F computes the next number in a BCD sequence

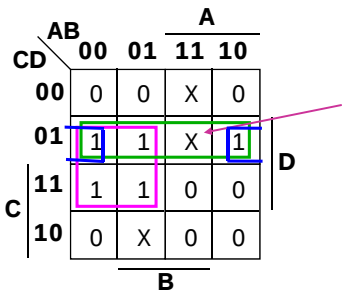
  If the input is 0010_2, the output is 0011_2

- BCD encodes decimal digits 0–9 as 0000_2–1001_2

  Don't care about binary numbers 1010_2–1111_2

Example: with don't cares

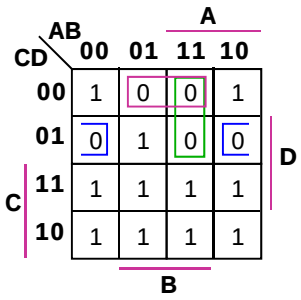
- F(A,B,C,D) = Σm(1,3,5,7,9) + d(6,12,13)
  - F = A'D + B'C'D *without using* don't cares
  - F = A'D + C'D *using* don't cares



Assign X == "1"
⇒ allows a 2-cube rather than a 1-cube

POS minimization using k-maps

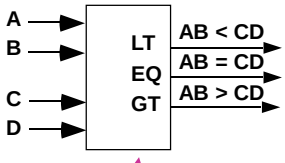
- Using k-maps for POS minimization
  - Encircle the zeros in the map
  - Interpret indices complementary to SOP form



F = (B'+C+D)(B+C+D')(A'+B'+C)

Same idea as the Truth Table

Design example: a two-bit comparator



block diagram

truth table

A	B	C	D	LT	EQ	GT
0	0	0	0	0	1	0
		0	1	1	0	0
		1	0	1	0	0
		1	1	1	0	0
0	1	0	0	0	0	1
		0	1	0	1	0
		1	0	1	0	0
		1	1	1	0	0
1	0	0	0	0	0	1
		0	1	0	0	1
		1	0	0	1	0
		1	1	1	0	0
1	1	0	0	0	0	1
		0	1	0	0	1
		1	0	0	0	1
		1	1	0	1	0

Need a 4 -variable Karnaugh map for each of the 3 output functions

Design example: a two-bit comparator (con't)

K-map for LT

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

C, D, A, B

K-map for EQ

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

C, D, A, B

K-map for GT

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

C, D, A, B

LT = A'B'D+A'C+B'CD

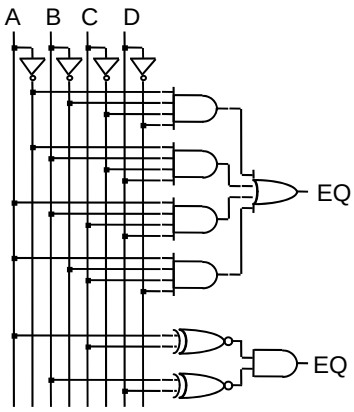
GT = BC'D'+AC'+ABD'

EQ = A'B'C'D'+A'BC'D+ABCD+AB'CD'

= (A xnor C)•(B xnor D)

Design example: a two-bit comparator (con't)

- Two ways to implement EQ:

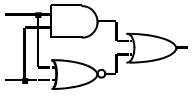


Option 1:
EQ = A'B'C'D'+A'BC'D+ABCD+AB'CD'

5 gates but they require lots of inputs

Option 2
EQ = (A xnor C) •(B xnor D)

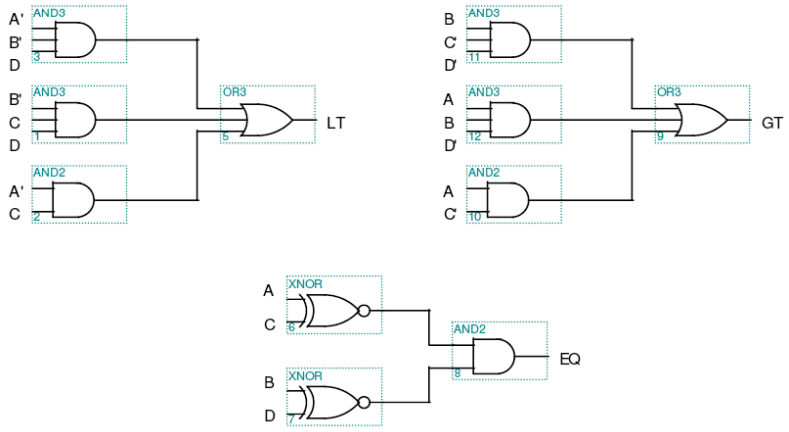
XNOR is constructed from 3 simple gates



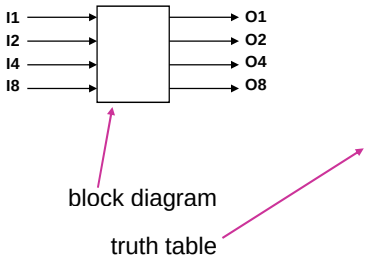
7 gates but they all have 2 inputs each

Design example: a two-bit comparator (con't)

Circuit schematics



Design example: BCD increment by 1



I8	I4	I2	I1	O8	O4	O2	O1
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

Need a 4 -variable Karnaugh map for each of the 4 output functions

Design example: BCD increment by 1 (con't)

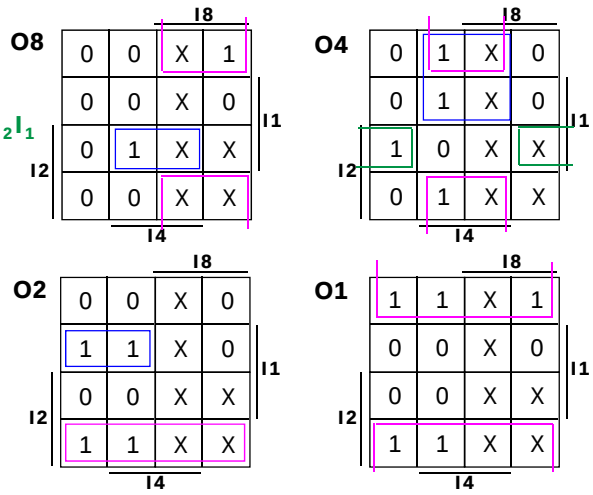
$O_8 = I_4I_2I_1 + I_8I_1$'

$O_4 = I_4I_2' + I_4I_1' + I_4'I_2I_1$

$O_2 = I_8'I_2'I_1 + I_2I_1$'

$O_1 = I_1$'

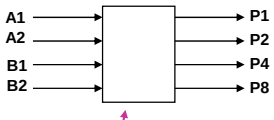
We greatly simplify the logic by using the don't cares



Design example: BCD increment by 1 (con't)

- Draw the circuit schematic
  - $O_8 = I_4I_2I_1 + I_8I_1$'
  - $O_4 = I_4I_2' + I_4I_1' + I_4'I_2I_1$
  - $O_2 = I_8'I_2'I_1 + I_2I_1$'
  - $O_1 = I_1$'

Design example: a two-bit multiplier



block diagram

truth table

A2	A1	B2	B1	P8	P4	P2	P1
0	0	0	0	0	0	0	0
		0	1	0	0	0	0
		1	0	0	0	0	0
		1	1	0	0	0	0
0	1	0	0	0	0	0	0
		0	1	0	0	0	1
		1	0	0	0	1	0
		1	1	0	0	1	1
1	0	0	0	0	0	0	0
		0	1	0	0	1	0
		1	0	0	1	0	0
		1	1	0	1	1	0
1	1	0	0	0	0	0	0
		0	1	0	0	1	1
		1	0	0	1	1	0
		1	1	1	0	0	1

Need a 4 -variable Karnaugh map for each of the 4 output functions

Two-bit multiplier (cont'd)

$P_8 = A_2A_1B_2B_1$

B2B1 \ A2A1	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	1	0
10	0	0	0	0

$P_4 = A_2B_2B_1' + A_2A_1'B_2$

B2B1 \ A2A1	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	1
10	0	0	1	1

$P_2 = A_2'A_1B_2 + A_1B_2B_1' + A_2B_2'B_1 + A_2A_1'B_1$

B2B1 \ A2A1	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	0	1	0	1
10	0	1	1	0

$P_1 = A_1B_1$

B2B1 \ A2A1	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0