Lecture 7

- Logistics
  - HW2 due now
  - Lab2 going on
  - 10/10 This Friday lecture: Prof. Paul Beame
- Last lecture
  - Logic simplification
    - Boolean cubes
    - Karnaugh maps
- Today
  - Continuing on K-maps (lots of examples)
  - Don't cares

The "WHY" slide

- Karnaugh map (K-map)
  - A visualization tool for logic simplification. It is a flatten version of Boolean cube that allows you to solve even difficult and complex Boolean expression into the minimized form. A convenient tool for you to know for up to 6 input variable expressions (e.g. X = f(A, B, C, D, E, F))
- 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.

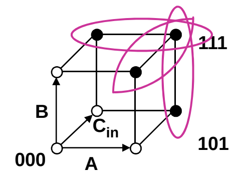
K-map minimization example: 3 variables

Find the least number of subcubes, each as large as possible, that cover the ON-set
Make sure subcubes contain 1, 2, 4, or 8 items (remember the Boolean cube)

A	B	Cin	Cout
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Cin \ AB	00	01	11 (A)	10 (A)
0	$_{0}$ 0	$_{2}$ 0	$_{6}$ 1	$_{4}$ 0
1	$_{1}$ 0	$_{3}$ 1	$_{7}$ 1	$_{5}$ 1
		B	B	

Cout = AB + BCin + ACin



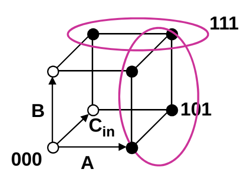
One more example: 3 variables

Find the least number of subcubes, each as large as possible, that cover the ON-set
Make sure subcubes contain 1, 2, 4, or 8 items (remember the Boolean cube)

A	B	Cin	Cout
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

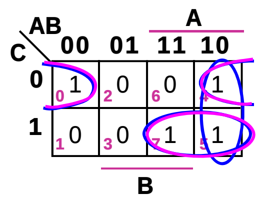
Cin \ AB	00	01	11 (A)	10 (A)
0	$_{0}$ 0	$_{2}$ 0	$_{6}$ 1	$_{4}$ 1
1	$_{1}$ 0	$_{3}$ 1	$_{7}$ 1	$_{5}$ 1
		B	B	

Cout = A + BCin

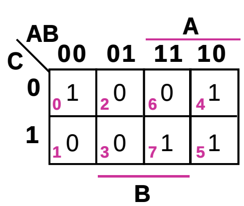


K-map minimization example: minterms

F(A,B,C) = Σm(0,4,5,7)
= B'C'+AC



K-map minimization example: complement

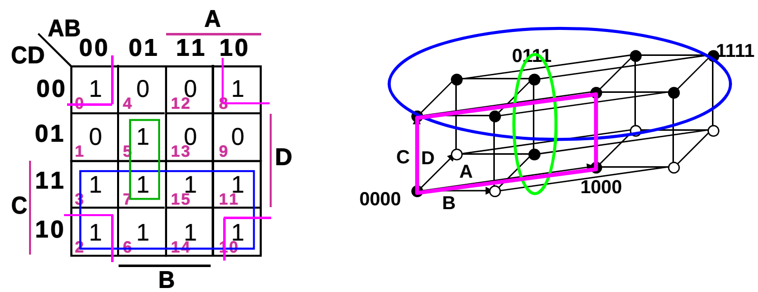


F(A,B,C) = Σm(0,4,5,7)
= B'C'+AC

F'(A,B,C) = Σm(1,2,3,6)
= A'C + BC'

K-map minimization example: 4 variables

- Minimize F(A,B,C,D) = Σm(0,2,3,5,6,7,8,10,11,14,15)
- Answer: F = C+A'BD+B'D'



Find the least number of subcubes, each as large as possible, that cover the ON-set

K-map minimization examples: on whiteboard

F(A,B,C) = Σm(0,3,6,7)

F(A,B,C) =
F'(A,B,C) =

C \ AB	00	01	11	10
0				
1				

F(A,B,C,D) = Σm(0,3,7,8,11,15)

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

CD \ AB	00	01	11	10
00				
01				
11				
10				

How about Karnaugh Maps, 6 dimensions

K-maps become 3D for 5 & 6 variables

AB = 00

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

AB = 01

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

AB = 11

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

AB = 10

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

OUTPUT =

A'BC'D'F' +

CF + BC'D'E

Incompletely specified functions: Don't cares

- Functions of n inputs have 2^n possible configurations
  - Some combinations may be unused
  - Call unused combinations "don't cares"
  - Exploit don't cares during logic minimization
  - Don't care ≠ no output
- Example: A BCD increment-by-1
  - 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

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

off-set for W: m0–m6, m9

on-set for W: m7 and m8

Don't care set for W:
We **don't care**
about the output values

Notation

- Don't cares in canonical forms
  - Three distinct logical sets: {on}, {off}, {don't care}
- Canonical representations of a BCD increment-by-1
  - Minterm expansion
    - W = m7+m8+d10+d11+d12+d13+d14+d15
      = Σm(7,8) + d(10,11,12,13,14,15)
  - Maxterm expansion
    - W = M0•M1•M2•M3•M4•M5•M6•M9•D10•D11•D12•D13•D14•D15
      = ΠM(0,1,2,3,4,5,6,9) • D(10,11,12,13,14,15)
- In K-maps, can treat 'don't cares' as 0s or 1s
  - Depending on which is more advantageous

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

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

A B C D

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