Minimization

Lecture Topics

- K-maps
- Minimization

Reading assignments

- Lumetta Set 2.1: Optimizing Logic Expressions

Karnaugh maps

- *Karnaugh map*, or *K-map*, is an alternative representation of truth table
  - Lists cells in *Gray code* order
  - Each cell corresponds to a minterm (row of the truth table)
- Two-variable Boolean function example:
  - four possible minterms, which can be arranged into a Karnaugh map

Conventional truth table for 2-variable function

x	**y**	**f(x,y)**
0	0	m_0
0	1	m_1
1	0	m_2
1	1	m_3

Corresponding K-map representation

		y	
		0	1
x	0	m_0	m_1
	1	m_2	m_3

f(x,y)

  - Now we can easily see which minterms contain common literals.
    - Minterms in column 0 and 1 contain y’ and y respectively.
    - Minterms in row 0 and 1 contain x’ and x respectively.
- Imagine a two-variable sum of minterms: x’y’ + x’y
  - Both of these minterms appear in the top row of a Karnaugh map, which means that they both contain the literal x’

		y	
		0	1
x	0	x’y’	x’y
	1	xy’	xy

f(x,y)

- What happens if you simplify this expression using Boolean algebra?
  - x’y’ + x’y = x’(y’ + y) = x’ • 1 = x’
- Another example expression is x’y + xy
  - Both minterms appear in the right side, where the literal y is common
  - Thus, we can reduce x’y + xy to just y

		y	
		0	1
x	0	x’y’	x’y
	1	xy’	xy

f(x,y)

- Another example x’y’ + x’y + xy
  - We have x’y’, x’y in the top row, combine along row to get x’
  - There is also x’y, xy in the right side, combine along column to y
  - This whole expression can be reduced to x’ + y

x \ y	0	1
0	x’y’	x’y
1	xy’	xy

f(x,y)

- Similarly, we can obtain K-maps for 3- and 4-variable Boolean functions

x \ yz	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

f(x,y,z)

wx \ yz	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

f(w,x,y,z)

- Some examples of 3-variable functions represented with K-maps

x \ yz	00	01	11	10
0	1	1	0	0
1	1	1	0	0

f(x,y,z)=y’

x \ yz	00	01	11	10
0	0	0	0	0
1	1	1	1	1

f(x,y,z)=x

x \ yz	00	01	11	10
0	0	1	0	0
1	0	0	0	0

f(x,y,z)=x’y’z

x \ yz	00	01	11	10
0	0	0	0	1
1	0	0	0	1

f(x,y,z)=yz’

x \ yz	00	01	11	10
0	0	0	0	0
1	1	0	0	1

f(x,y,z)=xz’

x \ yz	00	01	11	10
0	0	1	0	1
1	1	0	0	1

f(x,y,z)=x’y’z+yz’+xz’

- Observation: product terms correspond to rectangles
    - Rectangles / Cells / Literals in term

Rectangles	Cells	Literals in term
2x2 or 1x4	4	1
2x1 or 1x2	2	2
1x1	1	3

- Some examples of 4-variable functions represented with K-maps

wx \ yz	00	01	11	10
00		1		
01	1		1	1
11				1
10		1		

w’xz’+x’y’z+w’xy+xyz’

wx \ yz	00	01	11	10
00	1	1		1
01			1	1
11			1	1
10	1	1		1

x’z’+yz’+xy

wx \ yz	00	01	11	10
00	1			1
01				
11	1			1
10	1			1

wz’+x’z’

wx \ yz	00	01	11	10
00	1	1	1	1
01			1	1
11			1	1
10	1	1	1	1

x’+y

- Product terms correspond to rectangles
    - Rectangles / Cells / Literals in term

Rectangles	Cells	Literals in term
4x2 or 2x4	8	1
4x1 or 2x2 or 1x4	4	2
2x1 or 1x2	2	3
1x1	1	4

- Sum terms correspond to rectangles too:

wx \ yz	00	01	11	10
00	1	0	0	1
01	1	0	1	1
11	0	1	1	0
10	1	0	0	1

(w'+x'+z)(x+z')(w+y+z')

- Why *Grey code* ordering?
    - With this ordering, any group of 2, 4, 8, 16, … adjacent cells on the map contains common literals that can be factored out.
    - "Adjacency" includes wrapping around the left and right sides.

Function simplification

- K-maps is a great tool for simplifying Boolean expressions
- A product term is an *implicant* of a function if the function has the value 1 for all minterms of the product term
    - In terms of K-map, implicants correspond to *all* legal loops
- An implicant is a *prime implicant* if it is not contained within a larger implicant
    - In terms of K-map, prime implicants correspond to *all biggest* loops
- If a minterm is included in only one prime implicant, then it is an *essential prime implicant*
    - In other words, a prime implicant is essential if it covers some 1-cell for which no other prime implicants cover that cell
- Example:

Prime implicants

wx \ yz	00	01	11	10
00	1	1		
01	1	1		
11		1	1	1
10				1

f(w,x,y,z)

w'y', xy'z, wxz, wyz', wxy

Essential prime implicants

wx \ yz	00	01	11	10
00	1	1		
01	1	1		
11		1	1	1
10				1

f(w,x,y,z)

w'y', wyz'

- An SOP (or POS) expression is *minimal* if
    - It has the minimum number of product (sum) terms, and
    - Among expressions with minimum number of terms, it has fewest literals

- A minimal SOP expression is a sum of prime implicants. It consists of
  - All the essential prime implicants, and
  - As few as possible other prime implicants
- Procedure for finding minimal SOP representation
  - Find all essential prime implicants
    - For each 1 which has not yet been circled:
      - Is it covered by only one prime implicant? (i.e., there is no choice how to circle that 1?)
        - If yes, that prime implicant is essential and must be a term in any minimal SOP representation
  - Cover the remaining 1's using as few prime implicants as possible
  - In other words, find minimum number of rectangles to cover all 1's in K-map, each rectangle as large as possible

- Minimal SOP examples:

wx \ yz	00	01	11	10
00	1	1		
01	1	1	1	1
11	1	1		1
10	1	1		

y'+w'x+xz'

(all essential PIs)

wx \ yz	00	01	11	10
00		1		
01		1	1	1
11	1	1	1	
10			1	

wxy'+w'y'z+wyz+w'xy

(all essential PIs)

wx \ yz	00	01	11	10
00	1		1	1
01	1			1
11	1	1	1	
10			1	1

x'y+w'z'+wxy'+wxz

min SOP is not unique, there is another way to cover m_{12}, m_{13}, m_{15}:

x'y+w'z'+xy'z'+wxz

wx \ yz	00	01	11	10
00	1			1
01	1	1		
11		1	1	1
10	1			1

x'z'+w'xy'+wxz+wyz'

there are 4 min SOP solutions in this case

(x'z' is essential PI)

  - Note that min SOP may not be unique

- Minimal POS examples:

wx \ yz	00	01	11	10
00			0	0
01				
11			0	
10			0	0

(w'+y'+z')(x+y')

wx \ yz	00	01	11	10
00	0		0	0
01	0			
11				0
10	0	0		0

(w+y+z)(w+x+y') •
(w'+y'+z)(w'+x+y)

wx \ yz	00	01	11	10
00		0		
01		0	0	
11				0
10	0	0		

(w'+x'+y'+z)(w+x'+z') •
(w'+x+y)(w+y+z')
or last term can be (x+y+z')

wx \ yz	00	01	11	10
00		0	0	
01			0	0
11	0			
10		0	0	

(w'+x'+y+z)(w+x'+y')(x+z')