ECE380 Digital Logic

Optimized Implementation of
Logic Functions:
Strategy for Minimization,
Minimum Product-of-Sums Forms,
Incompletely Specified Functions

Terminology

- For a given term, each appearance of a variable (in true or complemented form) is called a ***literal***
  - xyz' => three literals
  - abc'd => four literals
- Any '1' or group of '1's that can be combined on a K-map represents an ***implicant*** of a function
- An implicant is a ***prime implicant*** if it cannot be combined with another implicant to remove a variable
- A collection of implicants that account of all valuations for which a given function is '1' is called a ***cover*** of that function
- ***Cost*** is the number of gates plus the total number of inputs to all gates in the circuit

Terminology example

cd \ ab	00	01	11	10
00	1	1	0	0
01	1	1	0	1
11	0	1	0	1
10	0	0	0	0

$f(a,b,c,d)=\Sigma m(0,1,4,5,7,9,11)$

Example Implicants: all single '1's,
a'c', a'b'c',
a'bd, ab'd

Prime Implicants: a'c', a'bd, ab'd, b'c'd

$f(a,b,c,d)_{min}$: a'c'+a'bd+ab'd

Thus, a minimum SOP form contains <u>only</u> (but not necessarily all) prime implicants.

Prime implicants distinctions

- **Essential**: needed to form a minimum solution
- **Nonessential**: not <u>necessarily</u> needed to form a minimum solution

cd \ ab	00	01	11	10
00	0	1	0	0
01	1	1	0	1
11	1	0	1	1
10	0	0	1	0

All prime implicants: b'd, a'bc', abc
a'c'd, acd

Essential primes: b'd, a'bc', abc

Nonessential primes: a'c'd, acd

$f(a,b,c,d)_{min}$: b'd+a'bc'+abc

Minimum contains <u>all</u> essential and <u>possibly</u> some nonessential primes

Prime implicants example

cd \ ab	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	1	1	0	0
10	1	1	0	0

Essential primes: a'c, ac'd

Nonessential primes: a'bd, bc'd

One of these must be included to form a minimum solution

$$f(a,b,c,d)_{min}: a'c + ac'd + \left\{ \begin{array}{c} a'bd \\ bc'd \end{array} \right\}$$

Prime implicants example

Identify all prime implicants for the given truth table. Which are essential and which are nonessential? What is a minimum SOP expression for this function?

cd \ ab	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	0	1	1	0
10	1	1	0	1

Minimization of POS expressions

- POS minimization using K-maps proceeds exactly as does SOP form except that groupings of '0's in the K-map are used to form POS terms.
- K-map can be constructed directly from ΠM expression for a function
- Place '0's in the K-map for every maxterm in the ΠM expression

Minimization of POS example

$f(a,b,c)=(a+b'+c')(a'+b+c')(a'+b'+c)(a'+b'+c')$

$f(a,b,c)=\Pi M(3,5,6,7)$

c \ ab	00	01	11	10
0	1	1	0	1
1	1	0	0	0

$f=(a'+b')(b'+c')(a'+c')$

Minimization of POS example

$f(a,b,c,d) = \Pi M(0,1,4,8,10\text{-}12,14,15)$

cd \ ab	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	1	1	0	0
10	1	1	0	0

(c+d)

(a+b+c)

(a'+c')

$f(a,b,c,d)_{min} = (a+b+c)(a'+c')(c+d)$

K-map groupings example

- Draw the K-map and give the minimized POS logic expression for the following.
  - $f(a,b,c) = \Pi M(0,2,3,5\text{-}7)$
- Show the groupings made in the K-map

Incompletely specified functions

- In digital systems it often happens that some input conditions (i.e. some input valuations) can never happen
- An input combination that can never happen is referred to as a ***don't care*** condition
- As a circuit is designed, a don't care condition can be ignored (i.e. the output for that condition can be treated as 0 or 1 in the truth table)
- A function that has don't care condition(s) is said to be ***incompletely specified***

Example function with don't cares

x	*y*	*z*	*f*
0	0	0	1
0	0	1	1
0	1	0	d
0	1	1	d
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Assume for a three variable function f(x,y,z) that the input combination xy=01 never occurs, otherwise the function is $\Sigma m(0,1,4,5)$

$f(x,y,z)= \Sigma m(0,1,4,5)+D(2,3)$

Or

$f(x,y,z)= \Pi M(6,7) \cdot D(2,3)$

Example function with don't cares

$f(x,y,z) = \Sigma m(0,1,4,5) + D(2,3)$

z \ xy	00	01	11	10
0	1	d	0	1
1	1	d	0	1

$f(x,y,z) = y'$

$f(x,y,z) = \Pi M(6,7) \cdot D(2,3)$

z \ xy	00	01	11	10
0	1	d	0	1
1	1	d	0	1

$f(x,y,z) = y'$

Minimum SOP form

1. Choose a minterm (a '1' in the K-map) which is not yet covered (don't consider d's).
2. Find all adjacent '1's and 'd's (check the n adjacent cells for an n-variable K-map).
3. If a single term (i.e. a single looping) covers the '1' and all adjacent '1's and 'd's then the looping forms an essential prime implicant. Loop the essential prime.
4. Repeat steps 1-3 until all essential prime implicants are located.
5. Find a minimum set of nonessential prime implicants to cover (loop) the remaining '1's. If more than 1 set is possible, choose the set with the minimum number of literals (the largest grouping).

Minimum POS form

1. Choose a maxterm (a '0' in the K-map) which is not yet covered (don't consider d's).
2. Find all adjacent '0's and 'd's (check the n adjacent cells for an n-variable K-map).
3. If a single term (i.e. a single looping) covers the '0' and all adjacent '0's and 'd's then the looping forms an essential prime implicant. Loop the essential prime.
4. Repeat steps 1-3 until all essential prime implicants are located.
5. Find a minimum set of nonessential prime implicants to cover (loop) the remaining '0's. If more than 1 set is possible, choose the set with the minimum number of literals (the largest grouping).