CMPE 212

Principles of Digital Design

Lecture 13

Quine-McCluskey Algorithm

March 7, 2016

www.csee.umbc.edu/~younis/CMPE212/CMPE212.htm

Lecture's Overview

- ***Previous Lecture:***
  - ➔ Extended K-map procedure
    (Multi-output optimization, map-entered variable)
  - ➔ Other circuit performance considerations
    (fan-in limitations, timing hazards issues and countermeasures)
- ***This Lecture***
  - ➔ The Quine-McCluskey algorithm
  - ➔ Tabular multi-output optimization
  - ➔ Petrick's algorithm

Conclusion

- ***Summary***
  - ➔ Extended K-map procedure
    (Multi-output optimization, map-entered variable)
  - ➔ Other circuit performance considerations
    (fan-in limitations, timing hazards issues and countermeasures)
  - ➔ The Quine-McCluskey algorithm
    (successive reduction, table of choices, Coverage process)
  - ➔ Petrick's algorithm
    (Coverage expression, prime implicants selection)
- ***Next Lecture***
  - ➔ Modular Combinational Logic

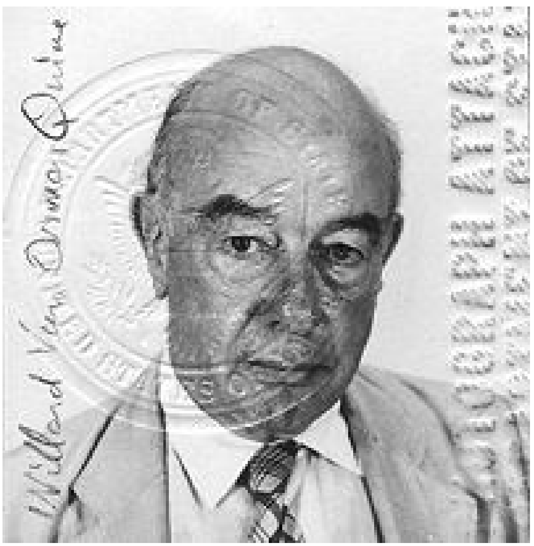
Reading assignment: Sections 3.9 – 3.10 in the textbook

Design Optimization

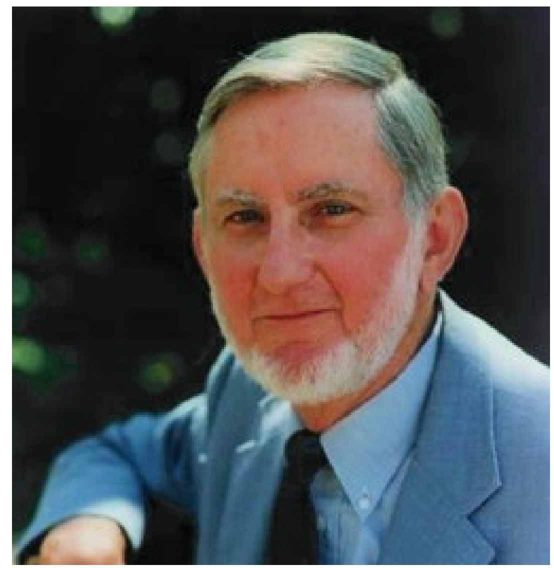
- Quality of combinational circuit design is measured using following metrics:
  - *Gate counts*: fewer gates require smaller area and cost less
  - *Propagation delay*: time for the output to become available after applying input. This time depends on transistor-level gate implementation
  - *Gate fan-in*: large gate fan-in can lead to increased gate counts and propagation delay (by using multi-level of gates)
  - *Gate fan-out*: large gate fan-out may mandate logic replication
- In many cases the canonical sum-of-products or product-of-sums forms are not minimal in terms in their number and size
- Since a smaller Boolean equation translates to a lower gate input count in the target circuit, reduction of the equation is an important consideration when circuit complexity is an issue
- Three methods for reducing Boolean equations are considered:
  - Algebraic reduction
  - Karnaugh map (K-map) reduction
  - Tabular reduction (Quine-McCluskey)

Quine-McCluskey Tabular Minimization Method

- W. V. Quine, “The Problem of Simplifying Truth Functions,” *American Mathematical Monthly*, vol. 59, no. 10, pp. 521-531, October 1952.
- E. J. McCluskey, “Minimization of Boolean Functions,” *Bell System Technical Journa*l, vol. 35, no. 11, pp. 1417-1444, November 1956.



Willard V. O. Quine
1908 – 2000



Edward J. McCluskey
born 1929, currently at Stanford

Tabular (Quine-McCluskey) Reduction

- The tabular method successively forms Boolean cross products among groups of terms that differ in one variable and then uses the smallest set of reduced terms
- Tabular reduction is systematic ➔ can be performed on a computer
- Tabular reduction begins by grouping minterms for which *F* is nonzero according to the number of 1's in each minterm
- Don't cares are considered to be nonzero

A	*B*	*C*	*D*	*F*
0	0	0	0	*d*
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	*d*
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	*d*

Initial setup

A	*B*	*C*	*D*
0	0	0	0
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	1	0
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	1

* Slide is courtesy of M. Murdocca and V. Heuring

Tabular Reduction (Cont.)

- The next step forms a consensus (the logical form of a cross product) between each pair of adjacent groups for all terms that differ in only one variable
- Common variables are removed between a couple of terms and replaced by a "_"
- A term can be used multiple times against the terms of the adjacent group
- Every term is included in the reduction is marked by a check
- Terms that are not covered are marked by '*' and correspond to prime implicants (may not be essential though)

Initial setup

A	B	C	D	
0	0	0	0	√
0	0	0	1	√
0	0	1	1	√
0	1	0	1	√
0	1	1	0	√
1	0	1	0	√
0	1	1	1	√
1	0	1	1	√
1	1	0	1	√
1	1	1	1	√

After first reduction

A	B	C	D
0	0	0	_
0	0	_	1
0	_	0	1
0	_	1	1
_	0	1	1
0	1	_	1
_	1	0	1
0	1	1	_
1	0	1	_
_	1	1	1
1	_	1	1
1	1	_	1

* Slide is courtesy of M. Murdocca and V. Heuring

Tabular Reduction (Cont.)

Initial setup

A	B	C	D	
0	0	0	0	√
0	0	0	1	√
0	0	1	1	√
0	1	0	1	√
0	1	1	0	√
1	0	1	0	√
0	1	1	1	√
1	0	1	1	√
1	1	0	1	√
1	1	1	1	√

After first reduction

A	B	C	D	
0	0	0	_	*
0	0	_	1	√
0	_	0	1	√
0	_	1	1	√
_	0	1	1	√
0	1	_	1	√
_	1	0	1	√
0	1	1	_	*
1	0	1	_	*
_	1	1	1	√
1	_	1	1	√
1	1	_	1	√

After second reduction

A	B	C	D	
0	_	_	1	*
_	_	1	1	*
_	1	_	1	*

- The consensus process is repeated using reduced tables
- The "_" has to be matched before a reduction can be made
- Process continue till no further reduction is possible

* Slide is courtesy of M. Murdocca and V. Heuring

Table of Choice

- The prime implicants form a set that completely covers the function, although not necessarily minimally.
- A table of choice is used to obtain a minimal cover set
- A single check in a column means that only one prime implicant covers the minterm ➔ becomes essential (must be picked)

True entries (no don't cares)

Prime Implicants	Minterms							
	0001	0011	0101	0110	0111	1010	1101	
0 0 0 _	√							
* 0 1 1 _				√	√			$\bar{A}BC$
* 1 0 1 _						√		$A\bar{B}C$
0 _ _ 1	√	√	√		√			
_ _ 1 1		√			√			
* _ 1 _ 1			√		√		√	BD

* Slide is courtesy of M. Murdocca and V. Heuring

Reduced Table of Choice

- In a reduced table of choice, the essential prime implicants and the minterms they cover are removed, producing the eligible set

Eligible Set		Minterms	
		0001	0011
X	0 0 0 _	√	
Y	0 _ _ 1	√	√
Z	_ _ 1 1		√

Set 1	Set 2
0 0 0 _	0 _ _ 1 ✓
_ _ 1 1	

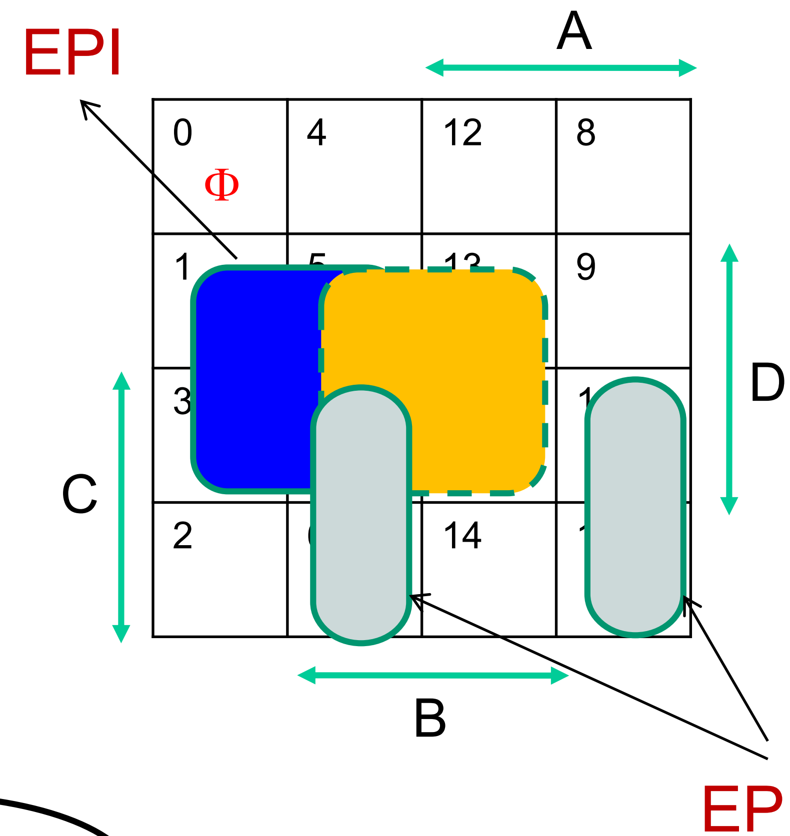
$$F = \bar{A}BC + A\bar{B}C + BD + \bar{A}D$$

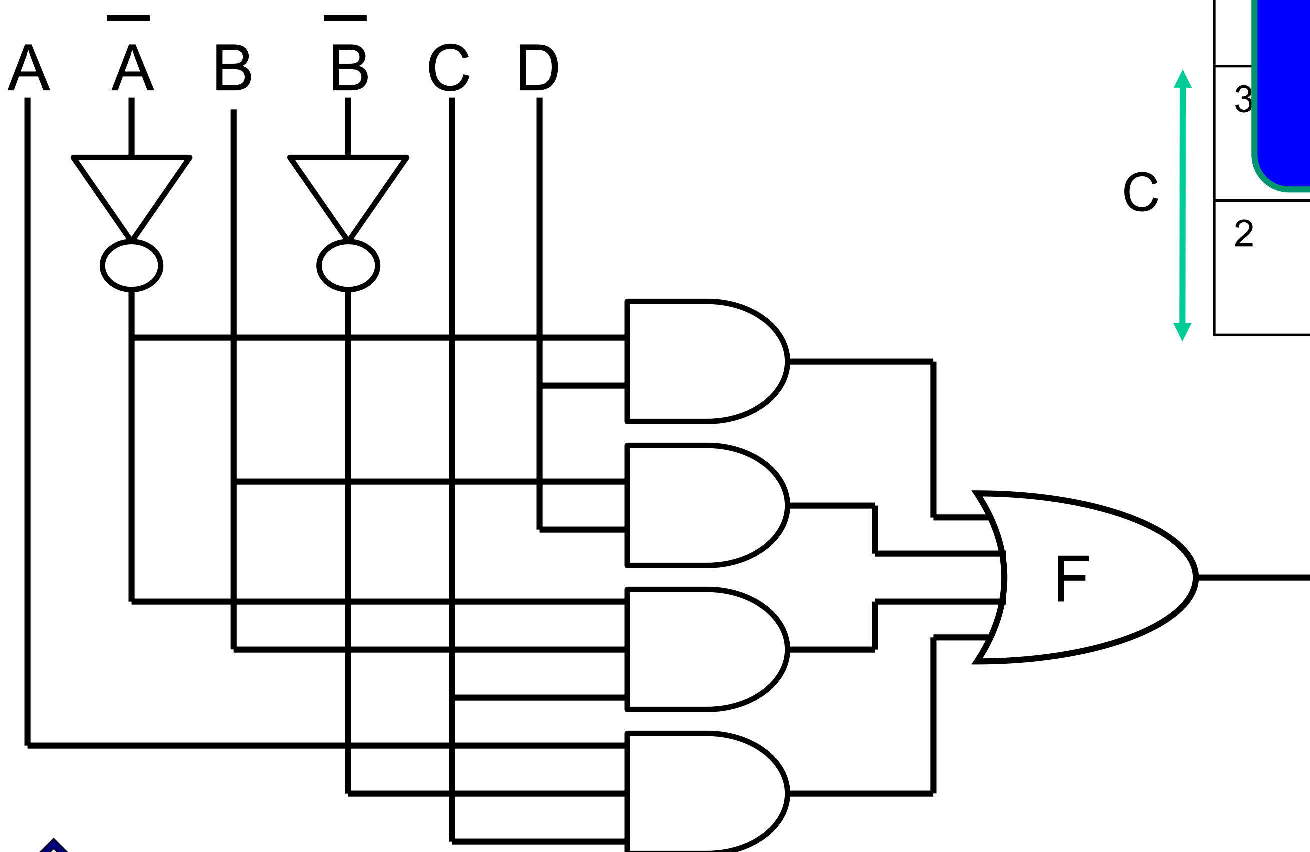
* Slide is courtesy of M. Murdocca and V. Heuring

Minimized Circuit

$$F = 011\_ + 101\_ + 0\_\_1 + \_1\_1$$

$$= \bar{A}BC + A\bar{B}C + \bar{A}D \quad + BD$$





Q-M Tabular Minimization Algorithm

- Begin with minterms:
  - Step 1: Tabulate minterms in groups of increasing number of true variables (including don't care entries)
  - Step 2: Conduct linear searches to identify all prime implicants
  - Step 3: Tabulate PI's vs. minterms to identify EPI's.
  - Step 4: Tabulate non-essential PI's vs. minterms not covered by EPI's. *Select* minimum number of PI's to cover all minterms.
- MSOP contains all EPI's and *selected* non-EPI's.
- Step 4 can be performed by modeling the selection as integer linear program (solved by MATLAB or any other tool)
- Minimizes functions with many variables; however suffers exponential growth of complexity w.r.t the number of inputs
- Can be implemented in software ➔ tool based logic reduction

Coverage Process (Step 4)

- Rule #1: Identify the columns that have only one entry, which would correspond to EPI, then remove all columns covered by that row
- Rule #2: Remove any row "*i*" that *is fully covered* by another row "*j*" since "*j*" covers all the minterms covered by "*i*"
- Rule #3: Remove any column "*i*" that *fully cover* another column "*j*" since any row that covers the minterm "*j*" will cover "*i*"
- Rule #4: In case there is no EPI, one PI is picked at random to get the coverage process
- Coverage can be performed by modeling the PI selection as Integer linear program (and solved by MATLAB or any other tool)
  - Define integer {0,1} variables, x_k = 1, select PI_k;
  - Constraints are imposed to cover all minterms
  - Objective minimize $\sum_k x_k$

Eligible Set	Minterms	
	0001	0011
~~X 0 0 0 _~~	~~√~~	
Y 0 _ _ 1	√	√
~~Z _ _ 1 1~~		~~√~~

Example: Coverage Process

	m_1	m_2	m_3	m_4	m_5	m_6
✓ PI_1	X		X			
PI_2		X	X			
PI_3		X				X
PI_4				X		X
PI_5				X	X	
PI_6	X				X	

No EPI (cyclic) ➔ Pick one at random

	m_2	m_4	m_5	m_6
~~PI_2~~	X			
PI_3	X			X
PI_4		X		X
PI_5		X	X	
~~PI_6~~			X	

PI_2 covered by PI_3 and PI_6 by PI_5

	m_2	m_4	m_5	m_6
✓ PI_3	X			X
PI_4		X		X
✓ PI_5		X	X	

PI_3 and PI_5 are essential

Alternatively

	m_2	m_4	m_5	m_6
✓ PI_3	X			X
PI_4		X		X
✓ PI_5		X	X	

m_6 covers m_2 and m_4 covers m_5

Pertick's Algorithm

- Follow the steps 1-3 of QM algorithm without any change
- Identify all EPIs and remove the corresponding rows and columns (same like QM algorithm)
- Determine optimal set of non-essential PIs for full coverage and least cost:
  a) For each minterm (column) m_i write a sum (OR) of all PIs that cover m_i (indicating that any of these PIs can cover m_i)
  b) Form the product (AND) of all minterms in the table (modeling the coverage as a Boolean expression)
- Convert the formed POS to SOP using to distributive axiom and simplify the expression (recursion!!)
- Select the cover with the least cost, i.e., number of PIs and number of literals in the PIs

	m_2	m_4	m_5	m_6
PI_2	X			
PI_3	X			X
PI_4		X		X
PI_5		X	X	
PI_6			X	

$$C= (PI_2 + PI_3)(PI_4 + PI_5)(PI_5 + PI_6)(PI_3 + PI_4)$$

$$C= (PI_3 + PI_2\, PI_4)(PI_5 + PI_4\, PI_6)$$

$$C= PI_3\, PI_5 + PI_3\, PI_4\, PI_6 + PI_2\, PI_4\, PI_5 + PI_2\, PI_4\, PI_6$$

System with Multiple Output

$f_{\alpha}(A,B,C,D) = \sum m(0,2,7,10) + d(12,15)$, $f_{\beta}(A,B,C,D) = \sum m(2,4,5) + d(6,7,8,10)$

$f_{\gamma}(A,B,C,D) = \sum m(2,7,8) + d(0,5,13)$

Minterm	List (ABCD)	Flags	mark
0	0000	αγ	
2	0010	αβγ	
4	0100	β	
8	1000	βγ	
5	0101	βγ	
6	0110	β	
10	1010	αβ	
12	1100	α	
7	0111	αβγ	
13	1101	γ	
15	1111	α	

List (ABCD)	Flags	mark

List (ABCD)	Flags	mark

1) Affix a flag to identify function
2) Combine 2 minterms if they have common flags (which will be kept to next stage)
3) Check off a minterm if all flags are kept in next stage

System with Multiple Output

$f_\alpha(A,B,C,D) = \sum m(0,2,7,10) + d(12,15)$, $f_\beta(A,B,C,D) = \sum m(2,4,5) + d(6,7,8,10)$

$f_\gamma(A,B,C,D) = \sum m(2,7,8) + d(0,5,13)$

Minterm	List (ABCD)	Flags	mark
0	0000	αγ	✓
2	0010	αβγ	PI_{10}
4	0100	β	✓
8	1000	βγ	PI_{11}
5	0101	βγ	✓
6	0110	β	✓
10	1010	αβ	✓
12	1100	α	PI_{12}
7	0111	αβγ	PI_{13}
13	1101	γ	✓
15	1111	α	✓

List (ABCD)	Flags	mark
00-0	αγ	PI_2
-000	γ	PI_3
0-10	β	PI_4
-010	αβ	PI_5
010-	β	✓
01-0	β	✓
10-0	β	PI_6
01-1	βγ	PI_7
-101	γ	PI_8
011-	β	✓
-111	α	PI_9

List (ABCD)	Flags	mark
01--	β	PI_1

- Identify all prime impicants
- Apply the coverage process

Coverage Process

		f_α				f_β			f_γ		
		0	2	7	10	2	4	5	2	7	8
PI_1	β						⊗	X			
PI_2	$\alpha\gamma$	⊗	X						X		
PI_3	γ										X
PI_4	β					X					
PI_5	$\alpha\beta$		X		⊗	X					
PI_6	β										
PI_7	$\beta\gamma$							X		X	
PI_8	γ										
PI_9	α			X							
PI_{10}	$\alpha\beta\gamma$		X			X			X		
PI_{11}	$\beta\gamma$										X
PI_{12}	α										
PI_{13}	$\alpha\beta\gamma$			X						X	

$$f_\alpha = \sum m(0,2,7,10) + d(12,15)$$

$$f_\beta = \sum m(2,4,5) + d(6,7,8,10)$$

$$f_\gamma = \sum m(2,7,8) + d(0,5,13)$$

		f_α	f_γ	
		7	7	8
PI_3	γ			X
~~PI_7~~	~~$\beta\gamma$~~		~~X~~	
~~PI_9~~	~~α~~	~~X~~		
PI_{11}	$\beta\gamma$			X
PI_{13}	$\alpha\beta\gamma$	X	X	

$$f_\alpha = PI_2 + PI_5 + PI_{13}$$

$$f_\beta = PI_1 + PI_5$$

$$f_\gamma = PI_2 + PI_3 + PI_{13}$$

Conclusion

- ***Summary***
  - ➔ The Quine-McCluskey algorithm
    (successive reduction, table of choices, Coverage process)
  - ➔ Petrick's algorithm
    (Coverage expression, prime implicants selection)
- ***Next Lecture***
  - ➔ Modular Combinational Logic

Reading assignment: Sections 3.9 – 3.10 in the textbook