CMPE 212
Principles of Digital Design

Lecture 12

Tabular Method for Logic Optimization

March 2, 2016

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

Lecture's Overview

❑ ***Previous Lecture:***

- ➔ Extended K-map procedure
  (Minterm covering, map-entered variable)
- ➔ Optimization of incompletely specified functions
  (Exploiting don't care entries)
- ➔ Scalability limitation of the K-map method
  (5 and 6 variables k-maps)

❑ ***This Lecture***

- ➔ Other circuit performance considerations
- ➔ The Quine-McCluskey algorithm
- ➔ Petrick's algorithm

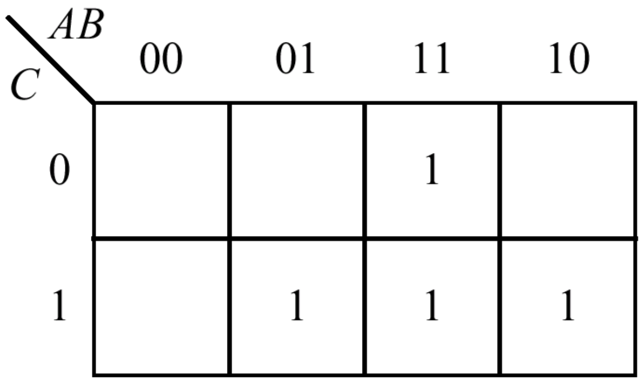
Design Optimization

- 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)

Forming K-Map

- Labeling along top and left side is arranged in a "Gray Code", in which exactly one variable changes between adjacent cells along each dimension
- Place a "1" in each cell that corresponds to that minterm (remaining cells should have a zero but omitted for clarity)
- Adjacent 1's satisfy the condition needed to apply complement property of Boolean algebra
- Grouping of adjacent cells are made into rectangles in sizes that corresponds to powers of 2 (called prime implicants)
- The larger the prime implicants gets, the higher the number of eliminated variables becomes
- Cells on the outer edge of the map "wrap around"
- All ones has to be covered in at least one prime implicant (multiple coverage is actually encouraged if it leads to further simplification)

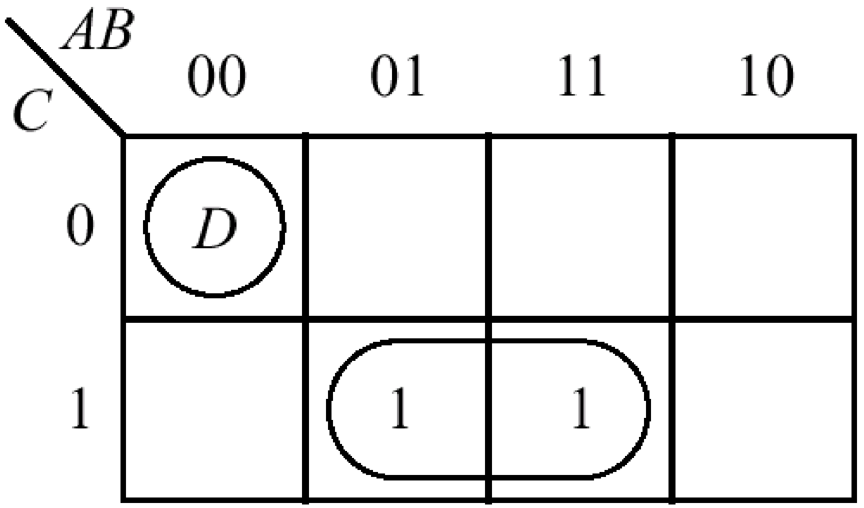
Minterm Index	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1



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

Map-Entered Variables

- ❑ Allowing variables to be entered in the K-map simplifies the representation of some functions (less map size)



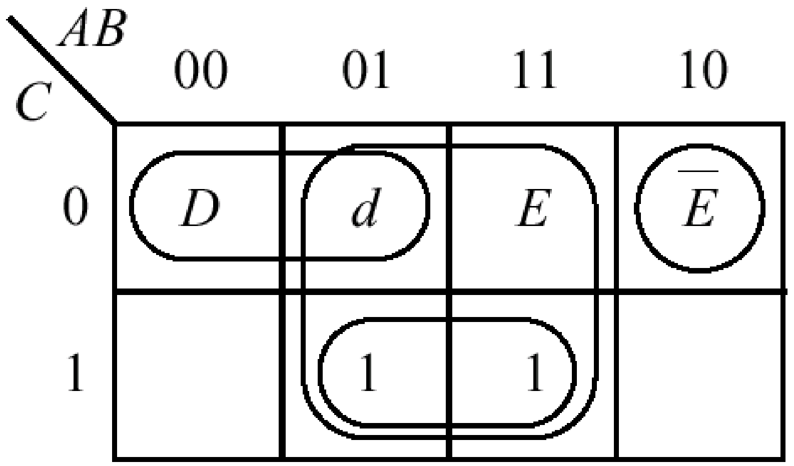
$$F = B\,C + \overline{A}\,\overline{B}\,\overline{C}\,D$$

- ❑ The map-entered variable D is treated as a 1 for the purpose of grouping, which in this case results in a 1-group since there are no adjacent 1's to the D cell
- ❑ Notice that the variable D appears in the final expression

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

General Procedure

- The general procedure for reducing an expression from K-map with map-entered variables:
  1. Obtain an expression for 1-cells while treating the map-entered variables as 0's
  2. Minterms are then added for each variable while treating 1's as don't care since the 1's have already been covered
  3. Repeat step 2 until all variables are covered



$$F = BC + \overline{A}\,\overline{C}D + BE + A\overline{B}\,\overline{C}\,\overline{E}$$

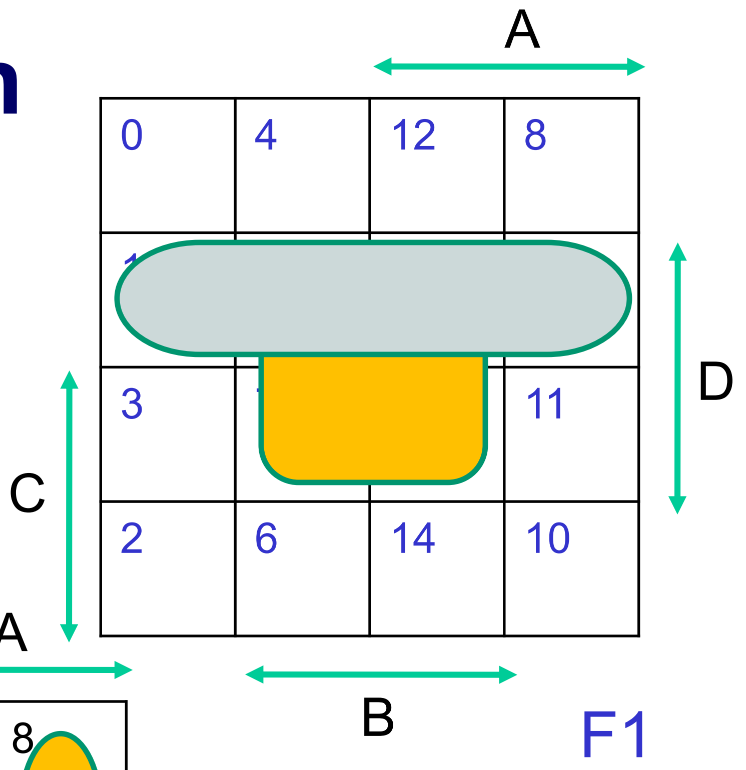
➢ Note that the E and its complement are considered distinct variables

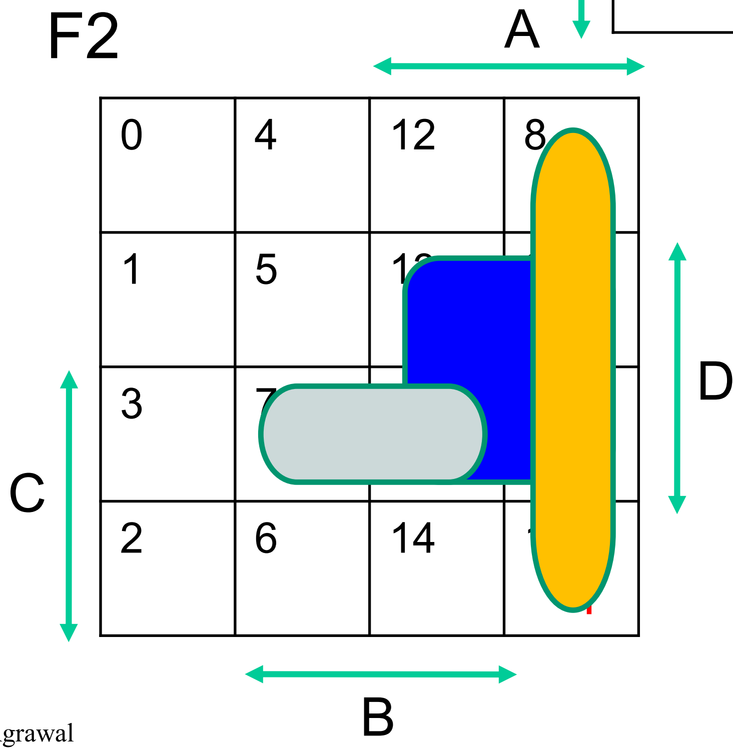
* Slide is courtesy of M. Murdocca and V. Heuring

Multiple-Output Function

Inputs				Outputs	
A	B	C	D	F1	F2
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	1	1
1	0	0	0	0	1
1	0	0	1	1	1
1	0	1	0	0	1
1	0	1	1	0	1
1	1	0	0	0	0
1	1	0	1	1	1
1	1	1	0	0	0
1	1	1	1	1	1

Need five products



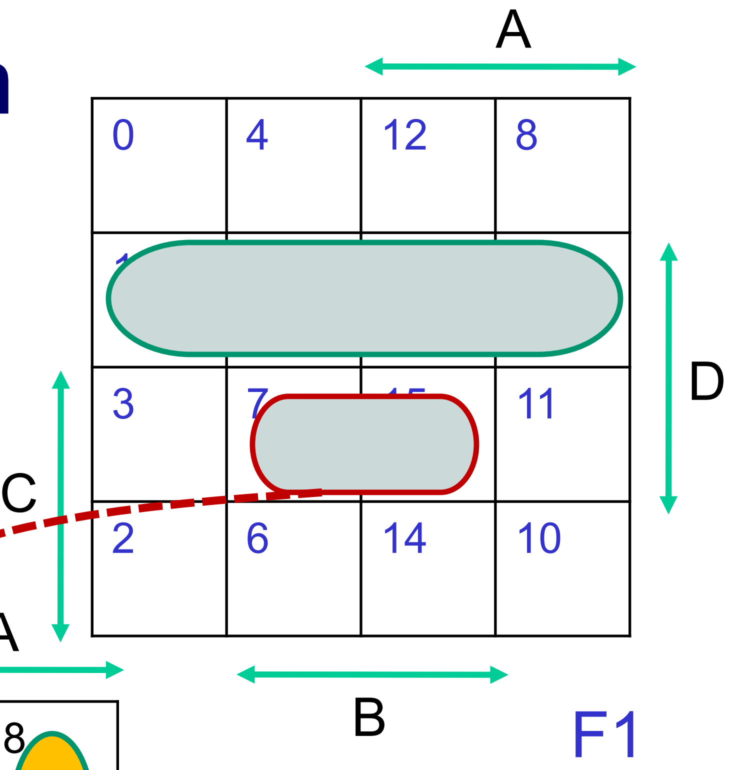


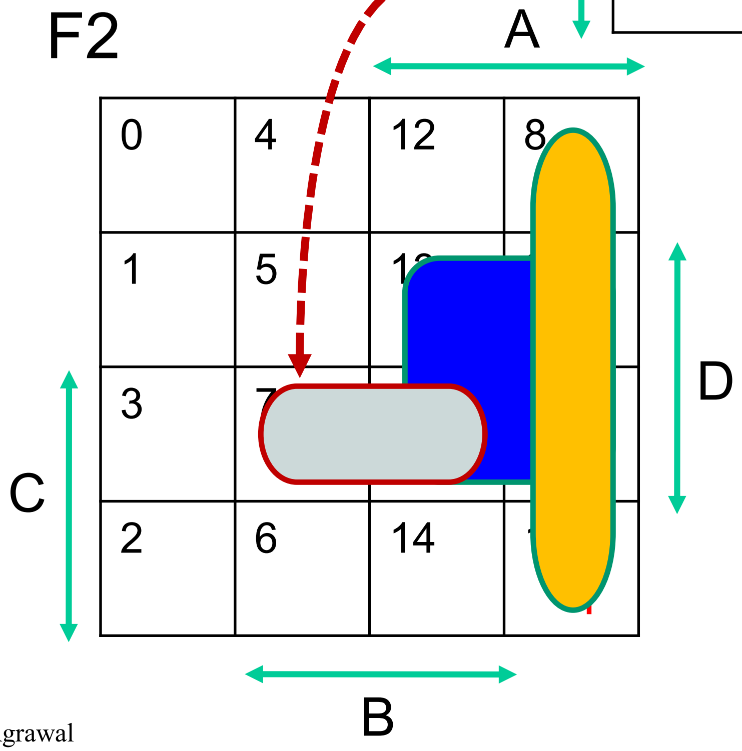
Individual Output Minimization

Multiple-Output Function

Inputs				Outputs	
A	B	C	D	F1	F2
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	1	1
1	0	0	0	0	1
1	0	0	1	1	1
1	0	1	0	0	1
1	0	1	1	0	1
1	1	0	0	0	0
1	1	0	1	1	1
1	1	1	0	0	0
1	1	1	1	1	1

Need Four products

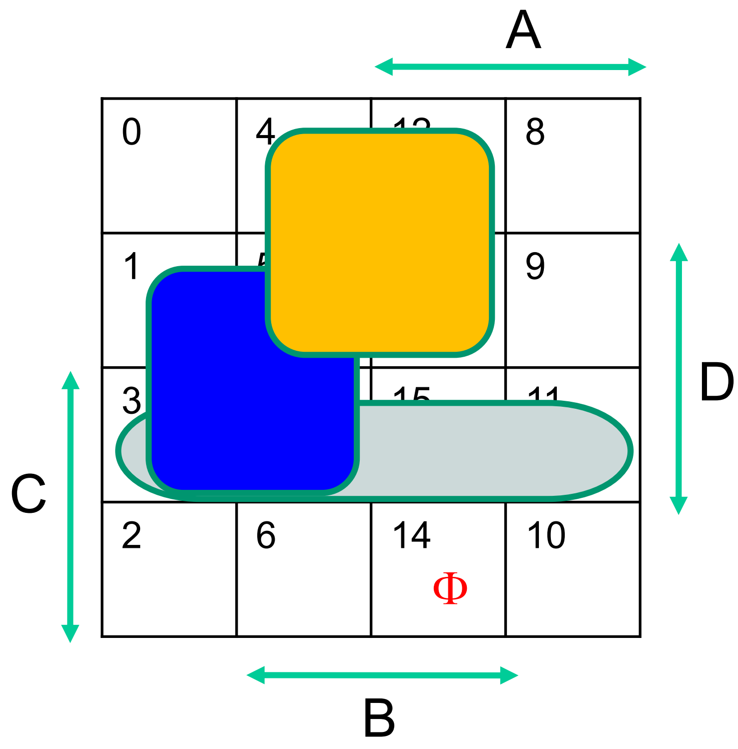


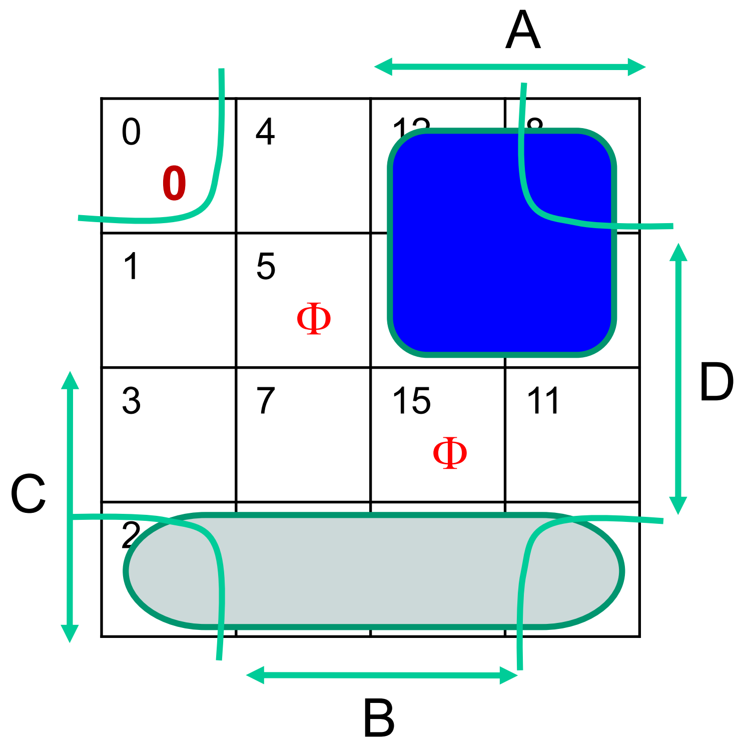


Global Minimization

Minimized SOP and POS

- $F(A,B,C,D) = \sum m(1,3,4,7,11) + d(5,12,13,14,15)$
  $= \prod M(0,2,6,8,9,10)\ D(5,12,13,14,15)$





$F = B\,\overline{C} + \overline{A}D + C\,D$

$\overline{F} = \overline{B}\ \overline{D} + C\ \overline{D} + A\ \overline{C}$

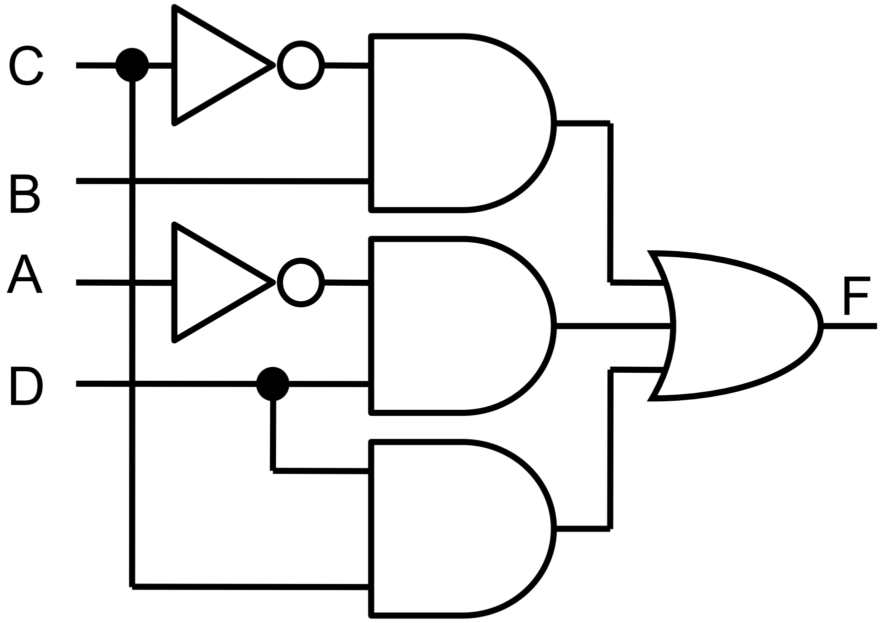
$F = (B + D)(\overline{C} + D)(\overline{A} + C)$

Slide contents are courtesy of Vishwani D. Agrawal

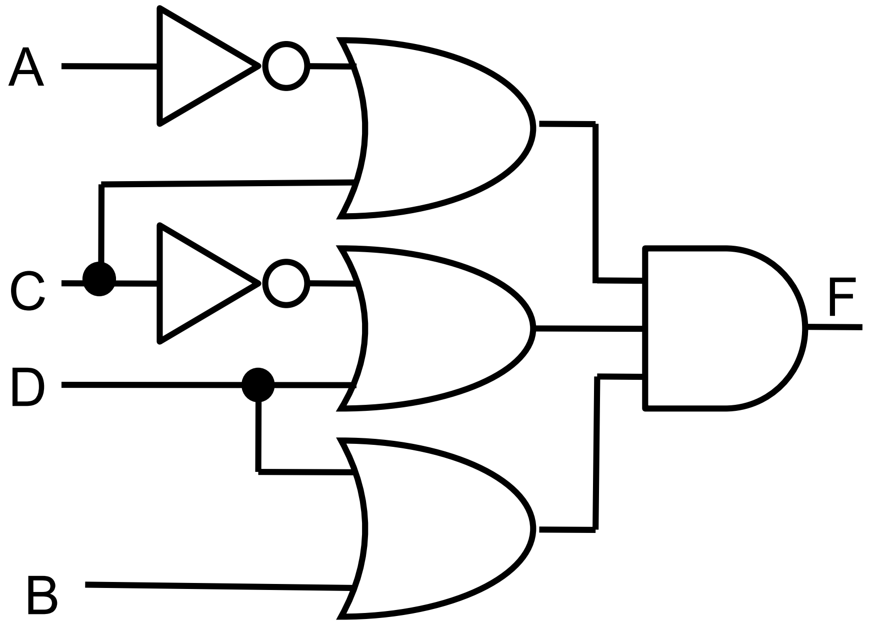
SOP and POS Circuits

- $F(A,B,C,D) = \sum m(1,3,4,7,11)+d(5,12,13,14,15)$
  $= \Pi\ M(0,2,6,8,9,10)\ D(5,12,13,14,15)$

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



$F = (B + D)(\bar{C} + D)(\bar{A} + C)$



Are two circuits functionally Identical?

Slide contents are courtesy of Vishwani D. Agrawal

Speed and Performance

Design Area

- Gate counts and interconnects derive the required design area

Circuit Depth

- Circuit depth is the number of logic gates on the longest path between the input and output within the circuit
- Fan-in and Fan-out of logic gates affects the circuit depth
- Circuit depth can be reduced using function decomposition

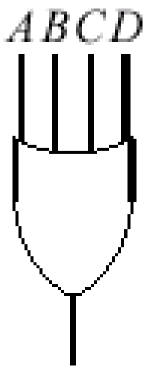
Circuit Speed

- The speed of a digital system is governed by:
  - the propagation delay through the logic gates
  - the propagation delay across interconnections

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

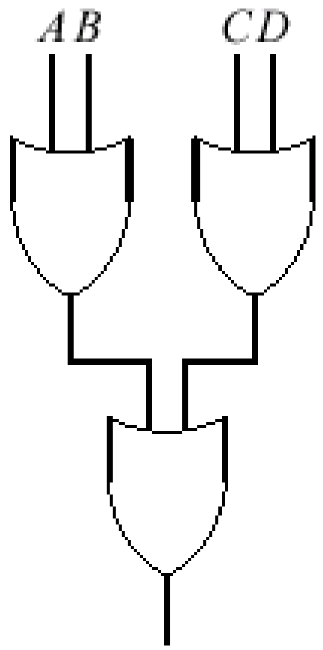
OR-Gate Decomposition

- Fan-in affects circuit depth.



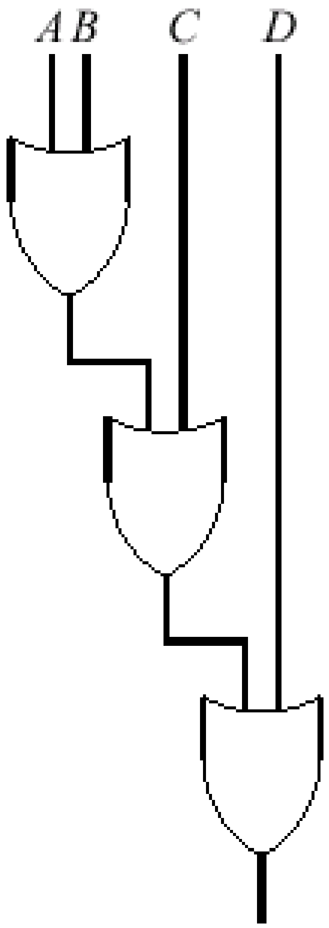
$A + B + C + D$

Initial high fan-in gate



$(A + B) + (C + D)$

Balanced tree



Degenerate trees increase circuit depth but facilitate circuit splitting on multiple stages

Associative law of Boolean algebra:

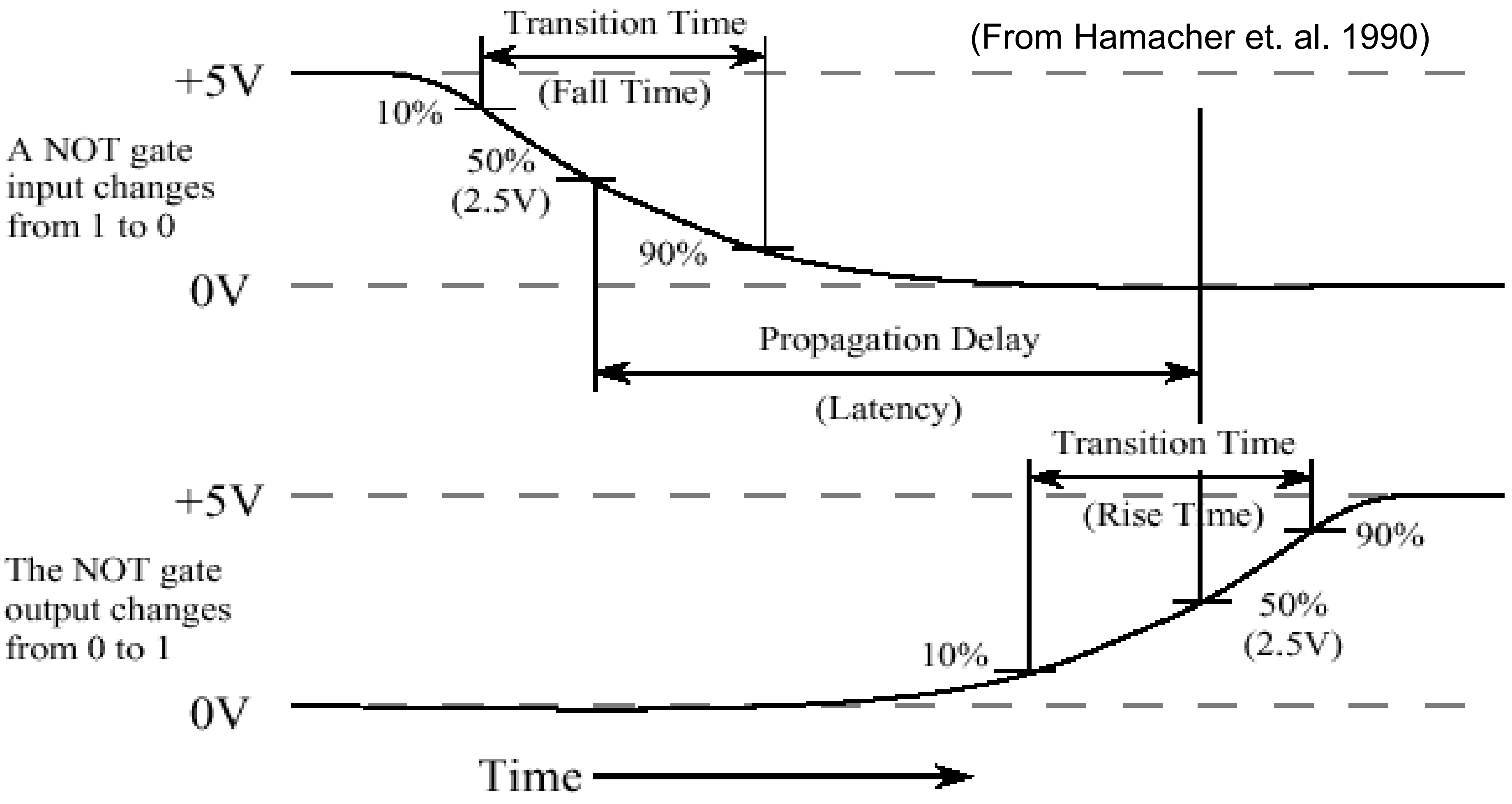
$A + B + C + D = (A + B) + (C + D)$

$((A + B) + C) + D$

Degenerate tree

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

Propagation Delay

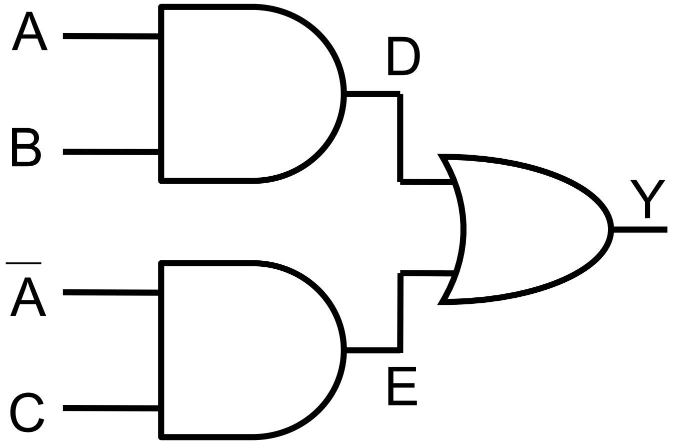


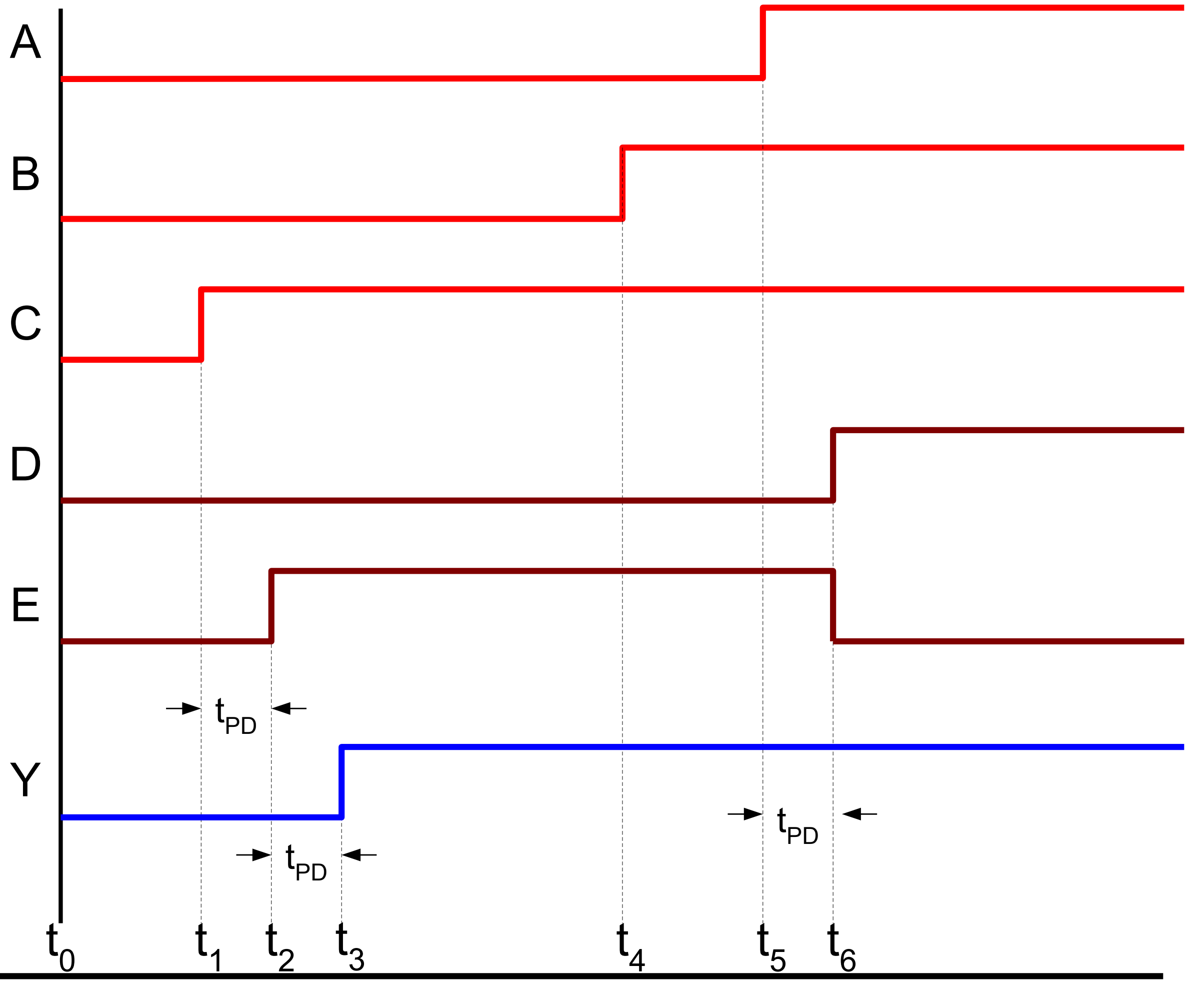
- ❑ Propagation delay depends on the technology used in the transistor's circuit
- ❑ As power consumption increases, propagation delay decreases
- ❑ High power consumption increases heat dissipation and requires circuit cooling

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

Timing Hazard

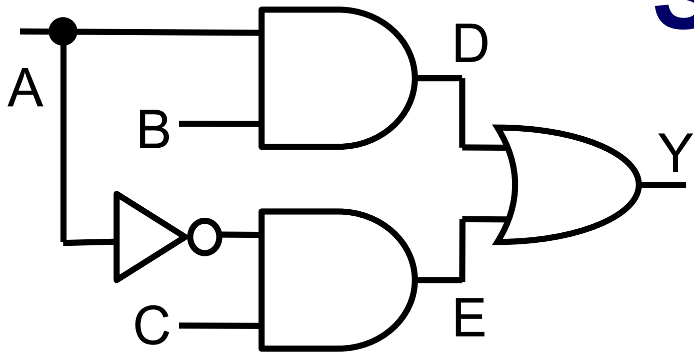
- Gate delay denotes the response time for an output of a gate to reflect the result of a new input
- The gate delay is in the order of nanosecond, yet may not be the same for all devices





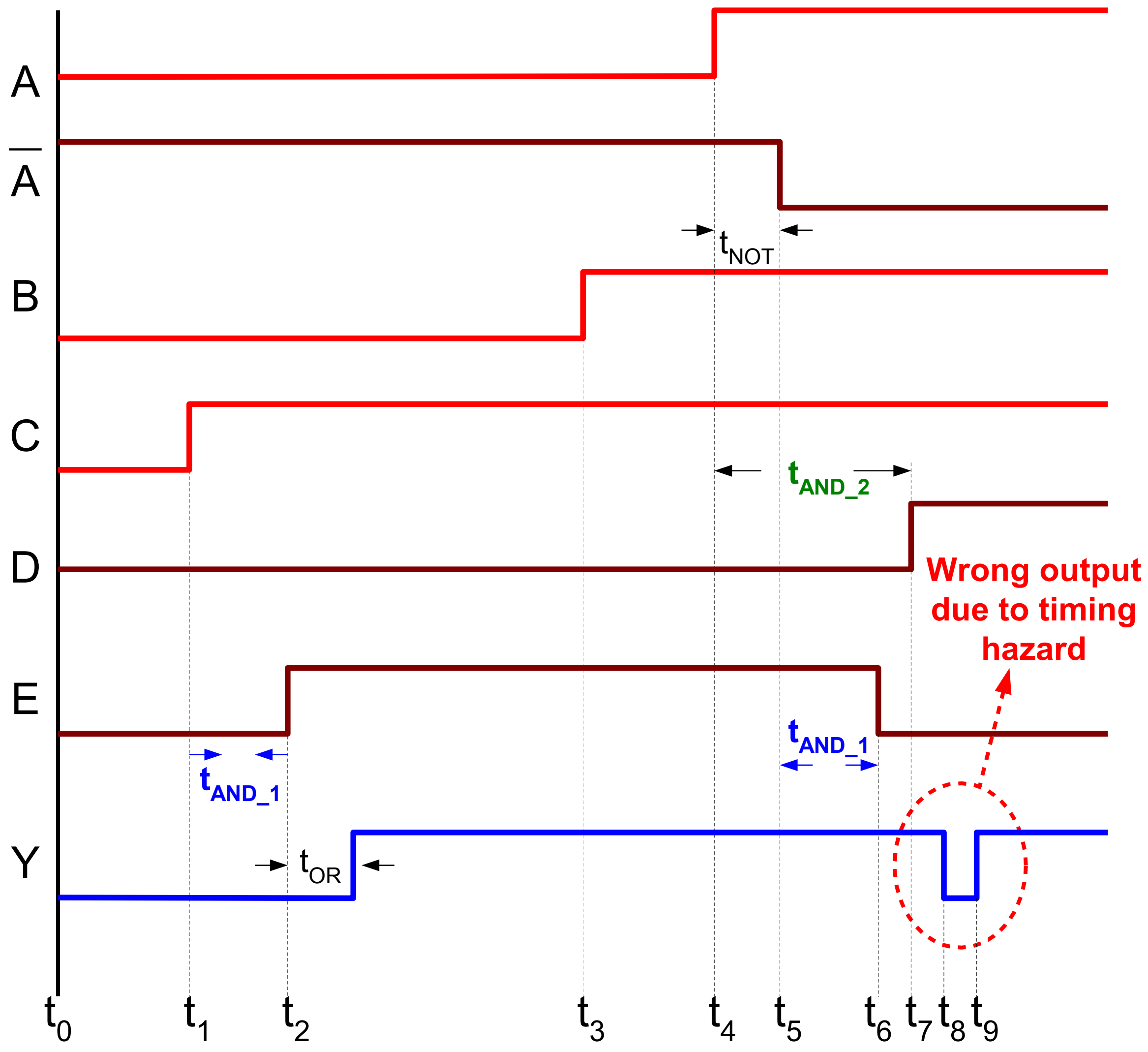
When all gates have the same response time, race conditions are avoided, and output "D" and "E" change simultaneously

Static Hazard



What if $\bar{A}$ was not readily available?

Gets more complicated if gate delay is not equal

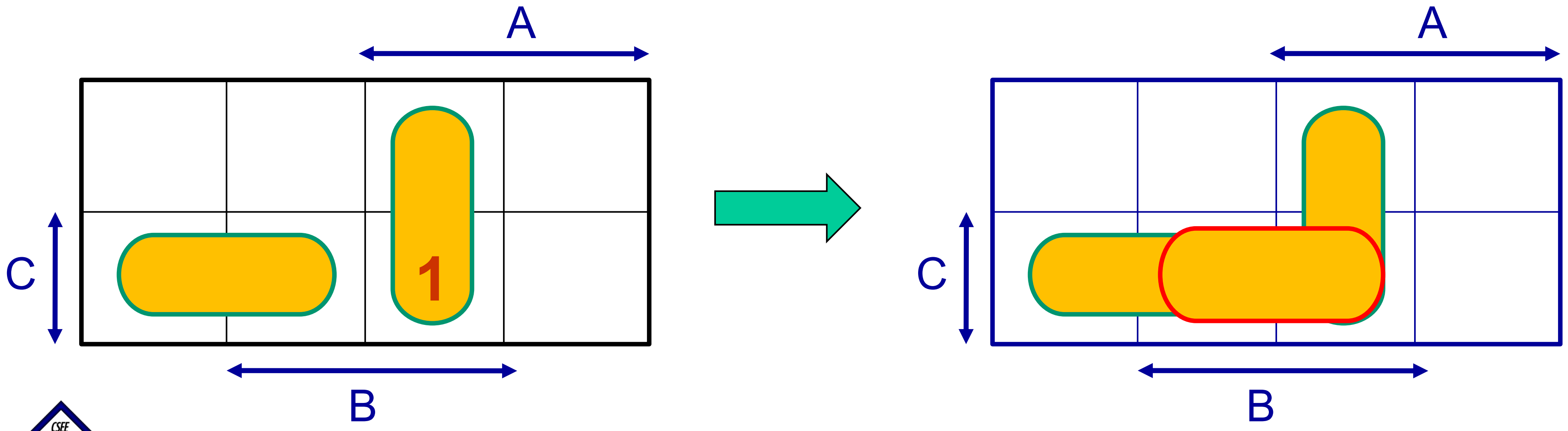


- In some periods, the output will reflect incorrect values and may cause problems to other circuits or the application
- Can be serious if the output is connected to an important device
- Called dynamic hazard if it happens more than once until the output settles

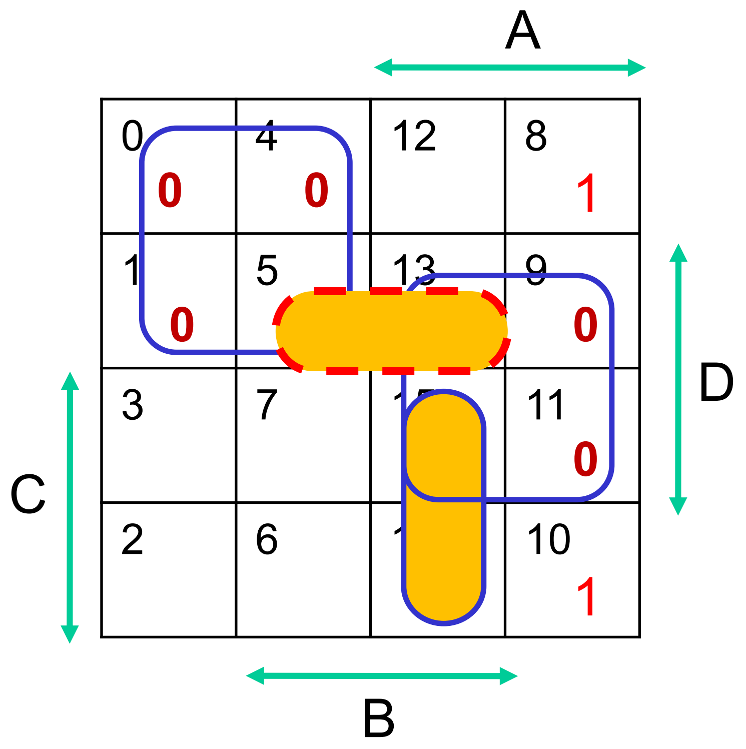
1 and 0 Glitches

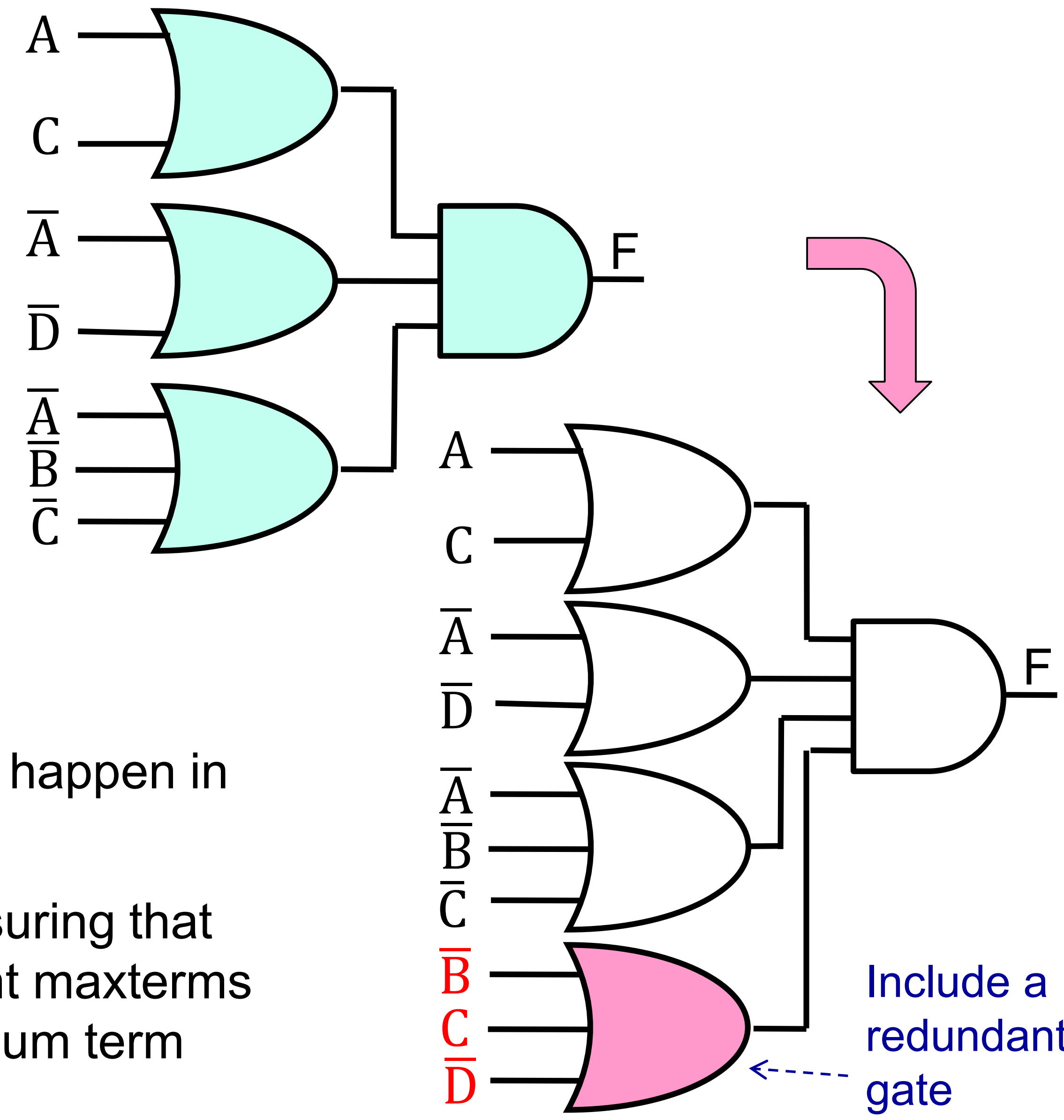
- Glitches refer to the case when momentary changes in the output takes place when no change should occur
- Static 1 hazard (1 glitch) occurs primarily in AND-OR circuits, while static 0 hazard (0 glitch) may happen in OR-AND circuits
- Static hazards can be prevented by ensuring that every pair of logically adjacent minterms (maxterms) are covered with a common product (sum) term

➔ Hazard can be eliminated with the inclusion of redundant gates



Example: 0 Glitch Avoidance

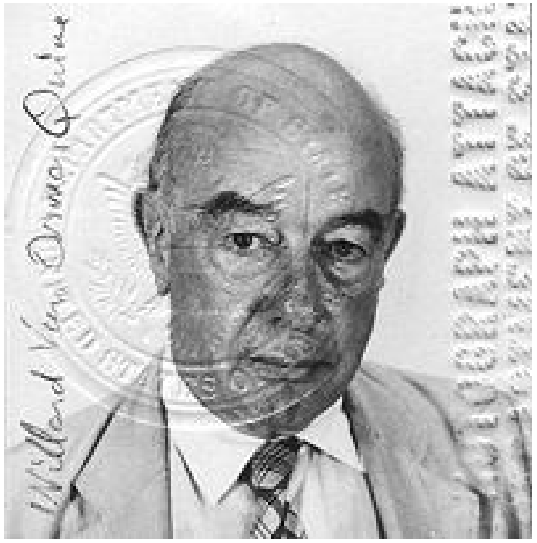


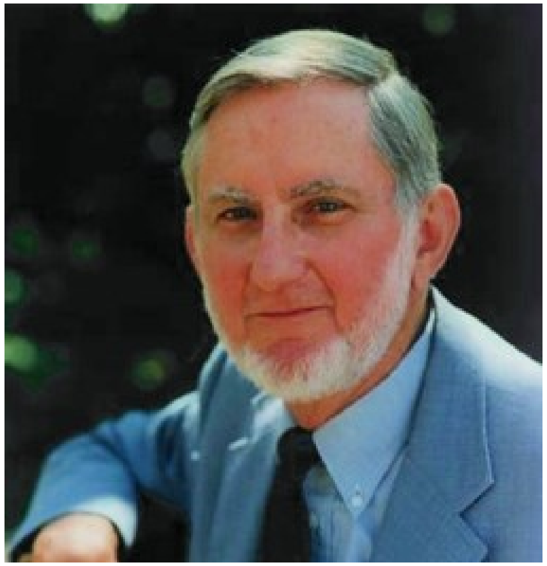


- Static 0 hazard (0 glitch) may happen in OR-AND circuits
- Prevent static hazards by ensuring that every pair of logically adjacent maxterms are covered with a common sum term

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

Slide contents are courtesy of Vishwani D. Agrawal

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 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	✓

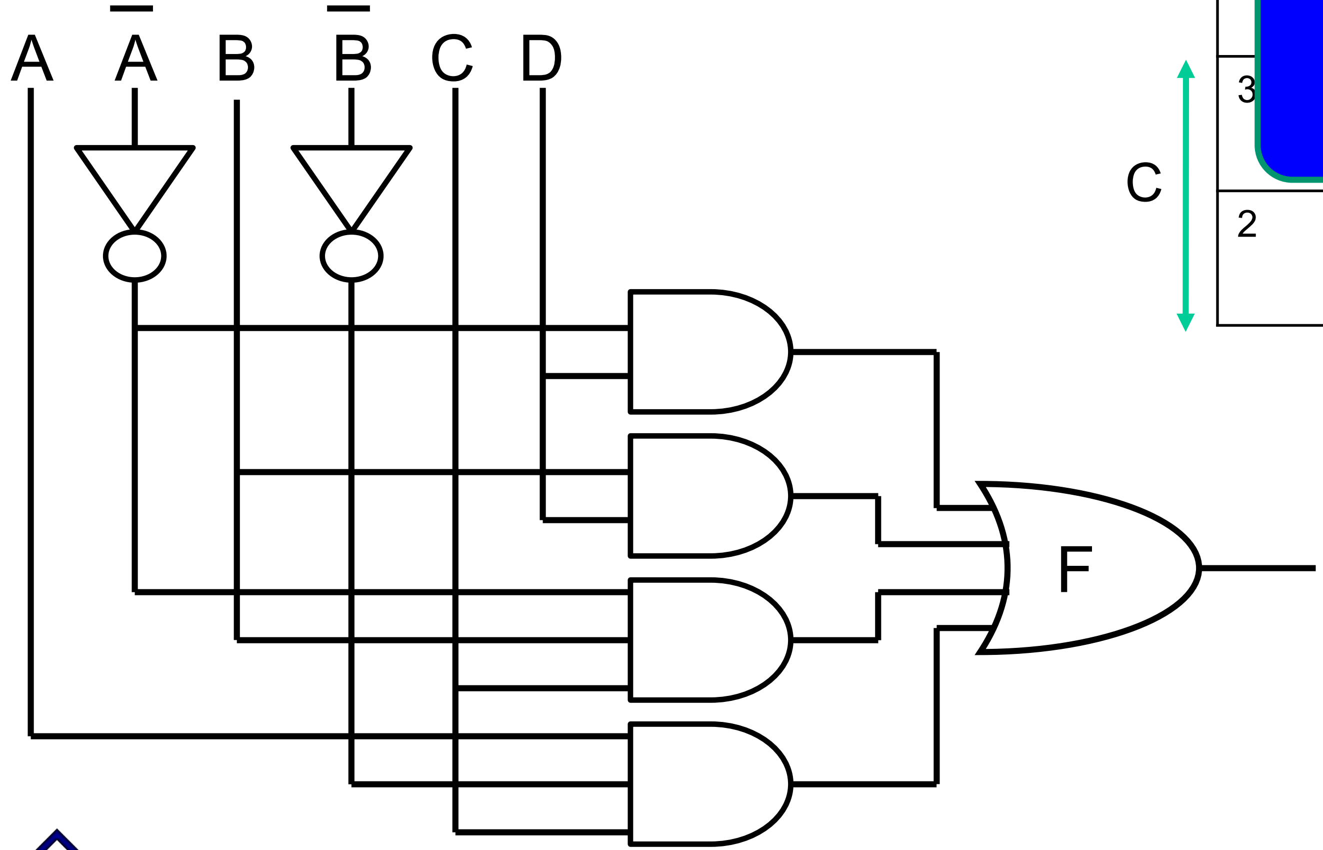
$$F = \bar{A}BC + A\bar{B}C + BD + \bar{A}D$$

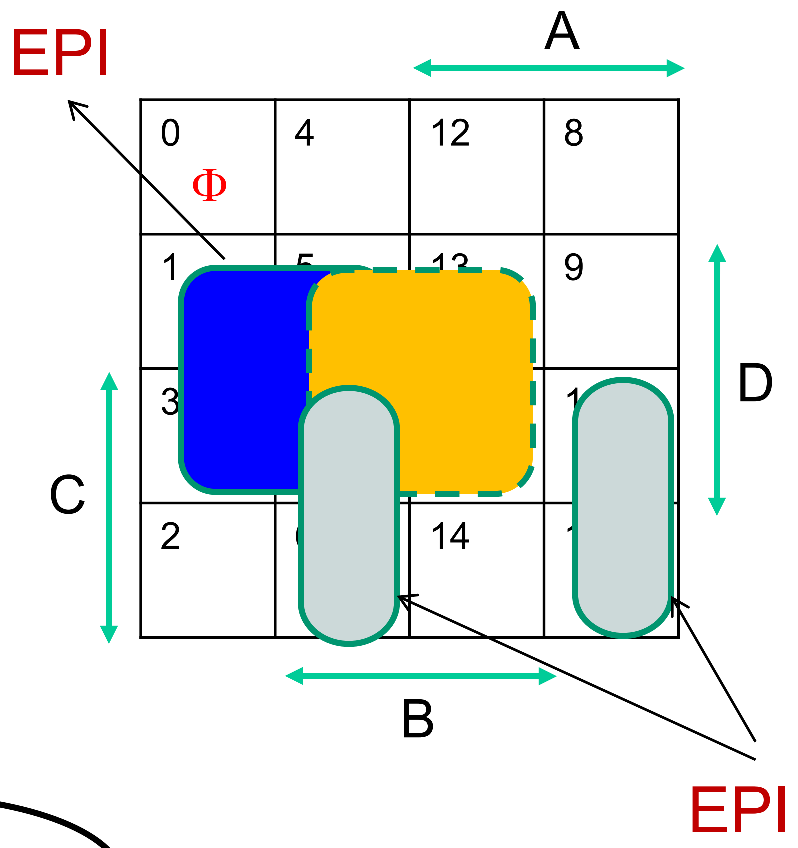
* Slide is courtesy of M. Murdocca and V. Heuring

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

Slide contents are c(partially) ourtesy of Vishwani D. Agrawal

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 a column "*i*" that *fully covers* another column "*j*" since any column 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* 000_~~	~~√~~	
Y 0__1	√	√
~~*Z* __11~~		~~√~~

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	

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

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

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

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