

Principles of Computer Architecture

Miles Murdocca and Vincent Heuring

Appendix B: Reduction of Digital Logic

Reduction (Simplification) of Boolean Expressions

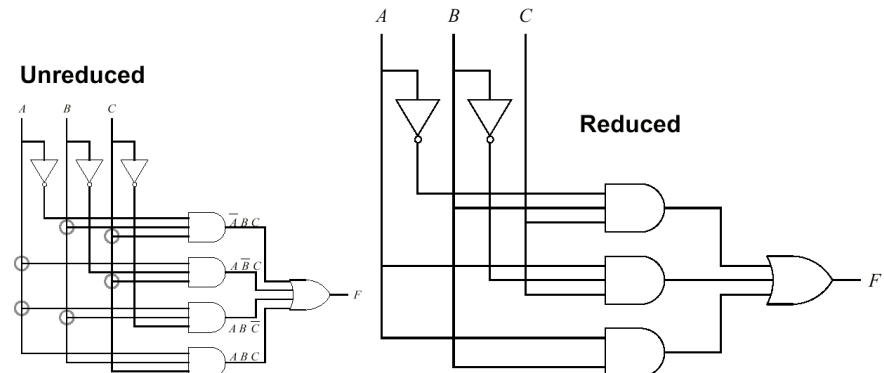
- It is usually possible to simplify the canonical SOP (or POS) forms.
- A smaller Boolean equation generally translates to a lower gate count in the target circuit.
- We cover three methods: algebraic reduction, Karnaugh map reduction, and tabular (Quine-McCluskey) reduction.

Chapter Contents

- B.1 Reduction of Combinational Logic and Sequential Logic
- B.2 Reduction of Two-Level Expressions
- B.3 State Reduction

Reduced Majority Function Circuit

- Compared with the AND-OR circuit for the unreduced majority function, the inverter for C has been eliminated, one AND gate has been eliminated, and one AND gate has only two inputs instead of three inputs. Can the function be reduced further? How do we go about it?



The Algebraic Method

- Consider the majority function, F . We apply the algebraic method to reduce F to its minimal two-level form:

$$F = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

$$F = \overline{A}BC + A\overline{B}C + AB(\overline{C} + C) \quad \text{Distributive Property}$$

$$F = \overline{A}BC + A\overline{B}C + AB(1) \quad \text{Complement Property}$$

$$F = \overline{A}BC + A\overline{B}C + AB \quad \text{Identity Property}$$

$$F = \overline{A}BC + A\overline{B}C + AB + ABC \quad \text{Idempotence}$$

$$F = \overline{A}BC + AC(\overline{B} + B) + AB \quad \text{Identity Property}$$

$$F = \overline{A}BC + AC + AB \quad \text{Complement and Identity}$$

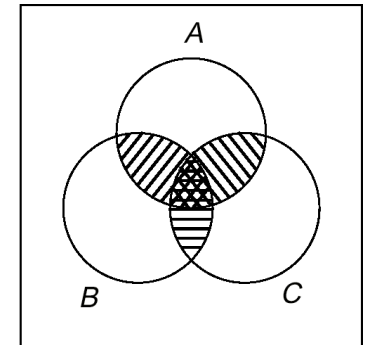
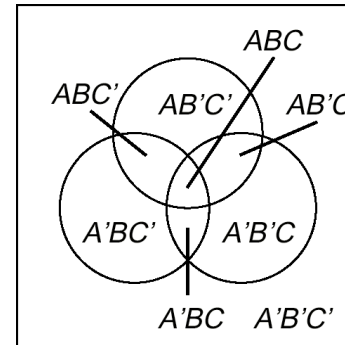
$$F = \overline{A}BC + AC + AB + ABC \quad \text{Idempotence}$$

$$F = BC(\overline{A} + A) + AC + AB \quad \text{Distributive}$$

$$F = BC + AC + AB \quad \text{Complement and Identity}$$

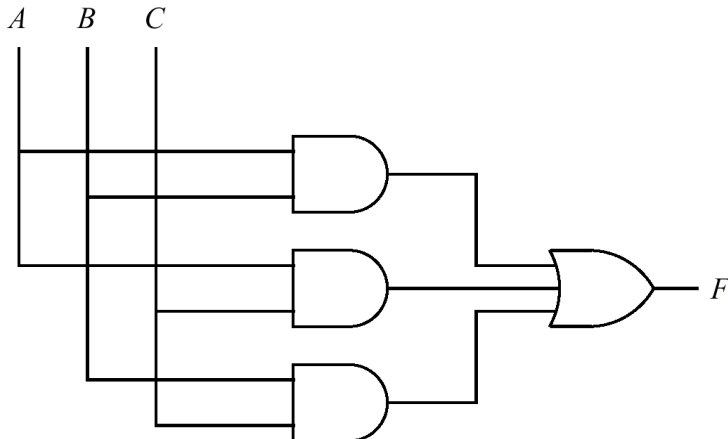
Karnaugh Maps: Venn Diagram Representation of Majority Function

- Each distinct region in the “Universe” represents a minterm.
- This diagram can be transformed into a *Karnaugh Map*.



The Algebraic Method

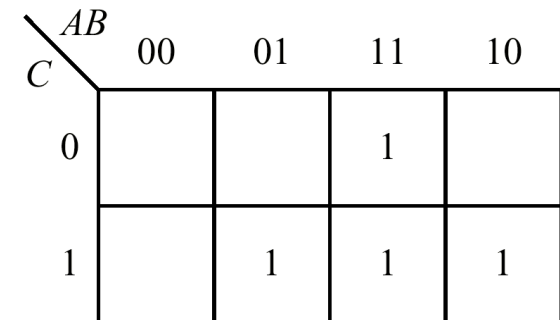
- This majority circuit is functionally equivalent to the previous majority circuit, but this one is in its minimal two-level form:



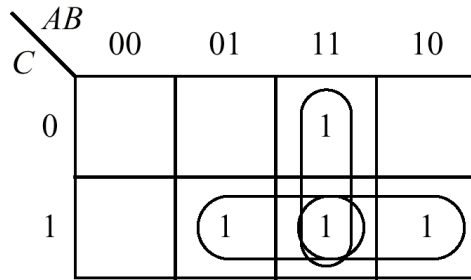
K-Map for Majority Function

- Place a “1” in each cell that corresponds to that minterm.
- Cells on the outer edge of the map “wrap around”

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



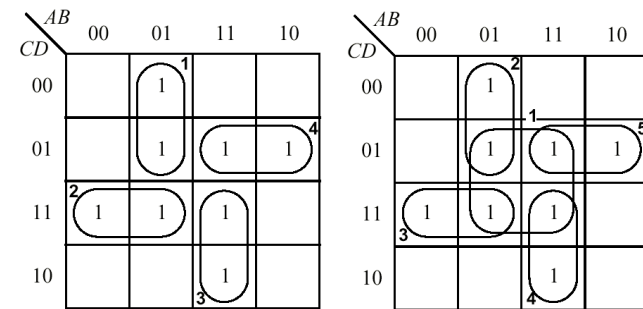
Adjacency Groupings for Majority Function



• $F = BC + AC + AB$

K-Map Groupings

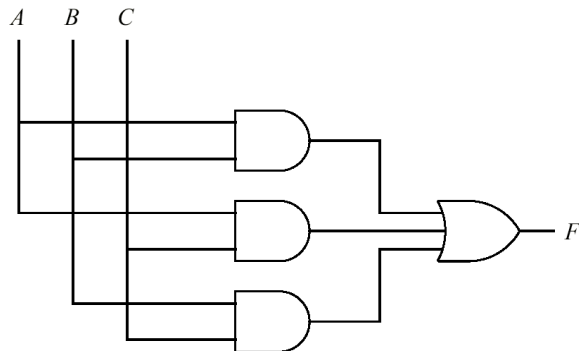
- Minimal grouping is on the left, non-minimal (but logically equivalent) grouping is on the right.
- To obtain minimal grouping, create *smallest* groups first.



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

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

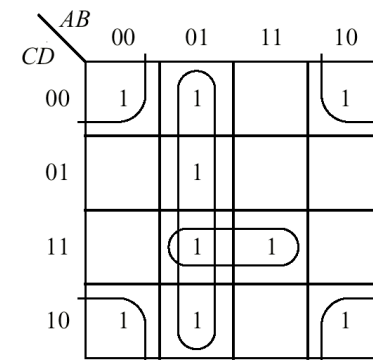
Minimized AND-OR Majority Circuit



• $F = BC + AC + AB$

- The K-map approach yields the same minimal two-level form as the algebraic approach.

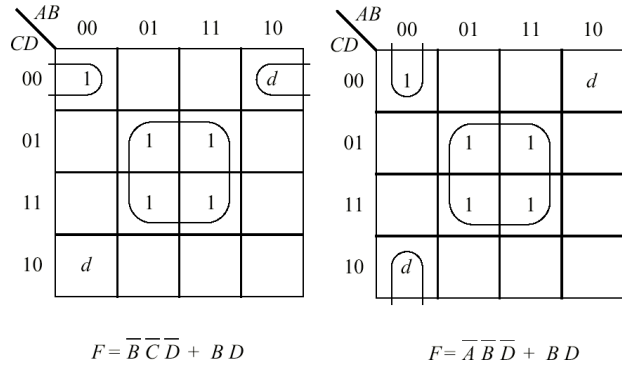
K-Map Corners are Logically Adjacent



$$F = BCD + \bar{B}\bar{D} + \bar{A}B$$

K-Maps and Don't Cares

- There can be more than one minimal grouping, as a result of don't cares.



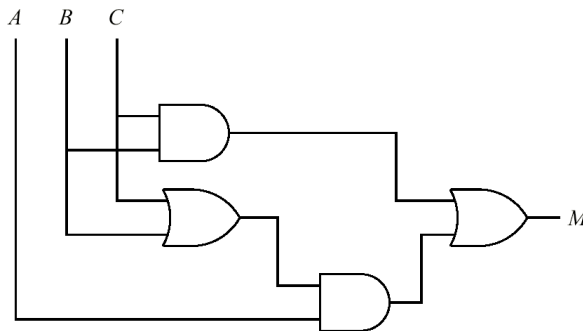
Truth Table with Don't Cares

- A truth table representation of a single function with don't cares.

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

3-Level Majority Circuit

- K-Map Reduction results in a reduced two-level circuit (that is, AND followed by OR. Inverters are not included in the two-level count). Algebraic reduction can result in multi-level circuits with even fewer logic gates and fewer inputs to the logic gates.



Tabular (Quine-McCluskey) Reduction

- 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.
- 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.

Initial setup	After first reduction	After second reduction																																																																																																																																									
<table border="1" style="border-collapse: collapse; margin: auto;"> <thead> <tr><th>A</th><th>B</th><th>C</th><th>D</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>√</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>1</td><td>√</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td><td>√</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td><td>√</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td><td>√</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>0</td><td>√</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>1</td><td>√</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>1</td><td>√</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td><td>√</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>1</td><td>√</td></tr> </tbody> </table> <p>(a)</p>	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	√	<table border="1" style="border-collapse: collapse; margin: auto;"> <thead> <tr><th>A</th><th>B</th><th>C</th><th>D</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>-</td><td>*</td></tr> <tr><td>0</td><td>0</td><td>-</td><td>1</td><td>√</td></tr> <tr><td>0</td><td>-</td><td>0</td><td>1</td><td>√</td></tr> <tr><td>0</td><td>-</td><td>1</td><td>1</td><td>√</td></tr> <tr><td>-</td><td>0</td><td>1</td><td>1</td><td>√</td></tr> <tr><td>0</td><td>1</td><td>-</td><td>1</td><td>√</td></tr> <tr><td>-</td><td>1</td><td>0</td><td>1</td><td>√</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>-</td><td>*</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>-</td><td>*</td></tr> <tr><td>-</td><td>1</td><td>1</td><td>1</td><td>√</td></tr> <tr><td>1</td><td>-</td><td>1</td><td>1</td><td>√</td></tr> <tr><td>1</td><td>1</td><td>-</td><td>1</td><td>√</td></tr> </tbody> </table> <p>(b)</p>	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	√	1	0	1	-	*	1	0	1	-	*	-	1	1	1	√	1	-	1	1	√	1	1	-	1	√	<table border="1" style="border-collapse: collapse; margin: auto;"> <thead> <tr><th>A</th><th>B</th><th>C</th><th>D</th></tr> </thead> <tbody> <tr><td>0</td><td>-</td><td>-</td><td>1</td><td>*</td></tr> <tr><td>-</td><td>-</td><td>1</td><td>1</td><td>*</td></tr> <tr><td>-</td><td>1</td><td>-</td><td>1</td><td>*</td></tr> </tbody> </table> <p>(c)</p>	A	B	C	D	0	-	-	1	*	-	-	1	1	*	-	1	-	1	*
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	√																																																																																																																																							
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	√																																																																																																																																							
1	0	1	-	*																																																																																																																																							
1	0	1	-	*																																																																																																																																							
-	1	1	1	√																																																																																																																																							
1	-	1	1	√																																																																																																																																							
1	1	-	1	√																																																																																																																																							
A	B	C	D																																																																																																																																								
0	-	-	1	*																																																																																																																																							
-	-	1	1	*																																																																																																																																							
-	1	-	1	*																																																																																																																																							

On considère la fonction = 1 pour tous les mintermes

Implicants premier	Mintermes									
	0000	0001	0011	0101	0110	0111	1010	1011	1101	1111
000-	√	√								
011-					√	√				
101-							√	√		
0--1		√	√	√		√				
-11			√			√		√		√
-1-1				√		√			√	√

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*.

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

Eligible Set	Minterms		
		0001	0011
X 000_	√		
Y 0__1	√	√	
Z __11			√

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

Table of Choice

On élimine les don't care

- 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.

Prime Implicants	Minterms						
	0001	0011	0101	0110	0111	1010	1101
0 0 0 _	√						
* 0 1 1 _				√	√		
* 1 0 1 _						√	
0 _ _ 1	√	√	√		√		
_ _ 1 1		√			√		
* _ 1 _ 1			√		√		√

Multiple Output Truth Table

- The power of tabular reduction comes into play for multiple functions, in which minterms can be shared among the functions.

Minterm	A	B	C	F_0	F_1	F_2
m_0	0	0	0	1	0	0
m_1	0	0	1	0	1	0
m_2	0	1	0	0	0	1
m_3	0	1	1	1	1	1
m_4	1	0	0	0	1	0
m_5	1	0	1	0	0	0
m_6	1	1	0	0	1	1
m_7	1	1	1	1	1	1

Multiple Output Table of Choice

$$F_0(A,B,C) = ABC + BC$$

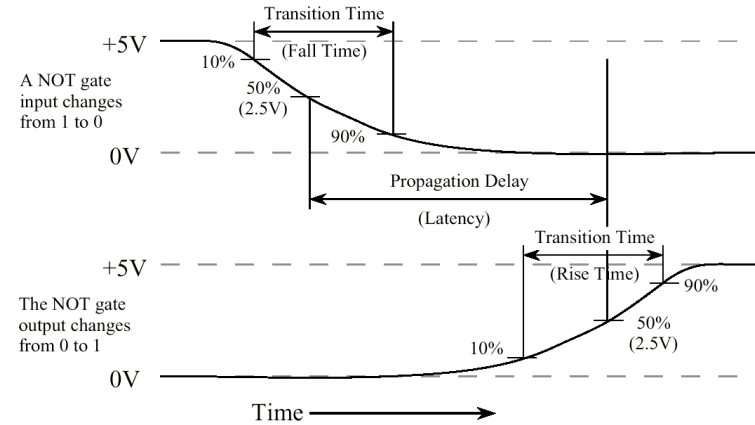
$$F_1(A,B,C) = AC + AC + BC$$

$$F_2(A,B,C) = B$$

Prime Implicants	Min-terms	$F_0(A,B,C)$			$F_1(A,B,C)$				$F_2(A,B,C)$			
	m_0 m_3 m_7	m_1 m_3 m_4 m_6 m_7	m_2 m_3 m_6 m_7									
F_0	* 0 0 0	√										
F_1	* 0 _ 1			√	√							
F_1	* 1 _ 0					√	√					
F_2	* _ 1 _							√	√	√	√	
$F_{1,2}$	1 1 _							√	√		√	√
$F_{0,1,2}$	* _ 1 1		√	√	√			√		√		√

Propagation Delay for a NOT Gate

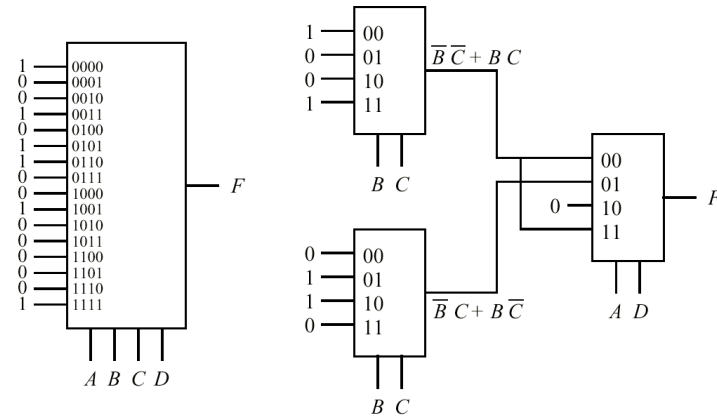
• (From Hamacher *et. al.* 1990)



Speed and Performance

- The speed of a digital system is governed by:
 - the propagation delay through the logic gates and
 - the propagation delay across interconnections.
- We will look at characterizing the delay for a logic gate, and a method of reducing circuit depth using function decomposition.

MUX Decomposition

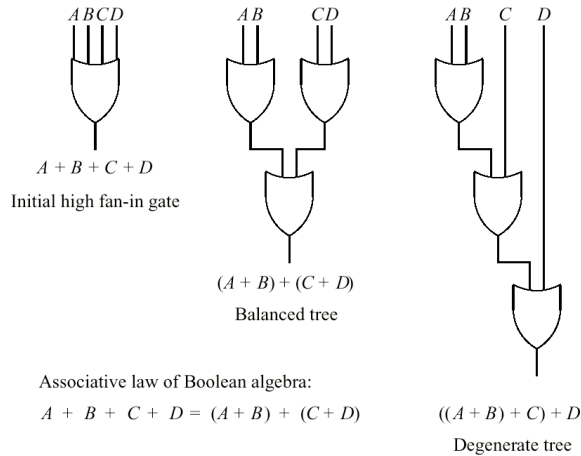


$$F(ABCD) = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}C\overline{D} + \overline{A}B\overline{C}\overline{D} + \overline{A}BC\overline{D} + A\overline{B}\overline{C}D + ABCD$$

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

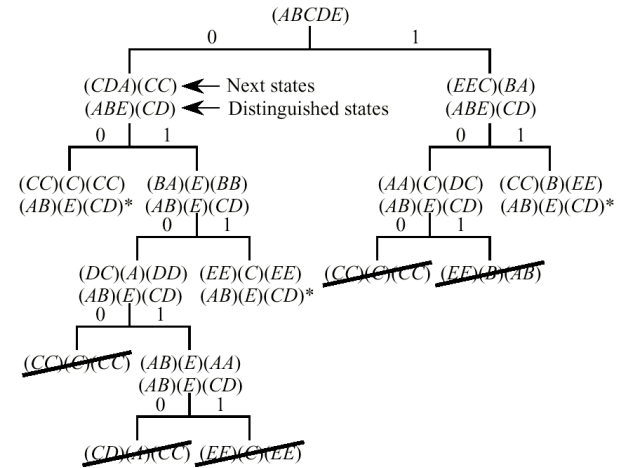
OR-Gate Decomposition

- Fanin affects circuit depth.



Distinguishing Tree

- A next state tree for M_0 .



State Reduction

- Description of state machine M_0 to be reduced.

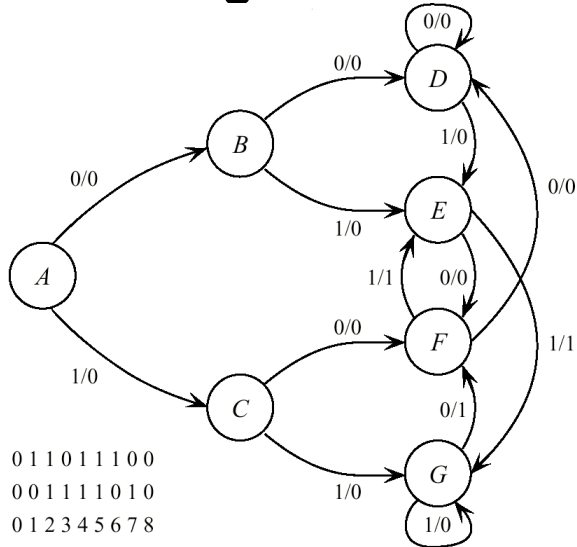
Present state \ Input	X	
	0	1
A	C/0	E/1
B	D/0	E/1
C	C/1	B/0
D	C/1	A/0
E	A/0	C/1

Reduced State Table

- A reduced state table for machine M_1 .

Current state \ Input	X	
	0	1
AB: A'	B'/0	C'/1
CD: B'	B'/1	A'/0
E: C'	A'/0	B'/1

Sequence Detector State Transition Diagram



Sequence Detector Reduced State Table

Present state \ Input	X	
	0	1
A: A'	B'/0	C'/0
BD: B'	B'/0	D'/0
C: C'	E'/0	F'/0
E: D'	E'/0	F'/1
F: E'	B'/0	D'/1
G: F'	E'/1	F'/0

Sequence Detector State Table

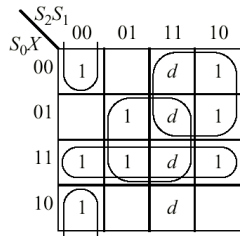
Present state \ Input	X	
	0	1
A	B/0	C/0
B	D/0	E/0
C	F/0	G/0
D	D/0	E/0
E	F/0	G/1
F	D/0	E/1
G	F/1	G/0

Sequence Detector State Assignment

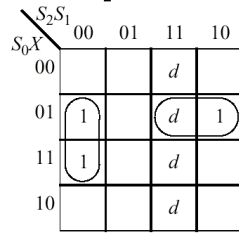
Present state \ Input	X	
	0	1
$S_2S_1S_0$	$S_2S_1S_0Z$	$S_2S_1S_0Z$
A': 000	001/0	010/0
B': 001	001/0	011/0
C': 010	100/0	101/0
D': 011	100/0	101/1
E': 100	001/0	011/1
F': 101	100/1	101/0

Sequence Detector K-Maps

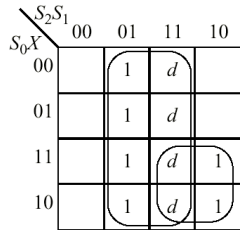
- K-map reduction of next state and output functions for sequence detector.



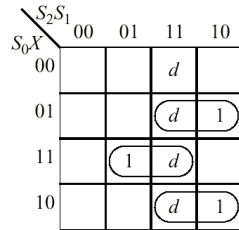
$$S_0 = \overline{S_2}S_1\overline{X} + S_0X + S_2S_0 + S_1X$$



$$S_1 = \overline{S_2}S_1X + S_2S_0X$$



$$S_2 = S_2S_0 + S_1$$



$$Z = S_2S_0X + S_1S_0X + S_2S_0\overline{X}$$

Excitation Tables

- Each table shows the settings that must be applied at the inputs at time t in order to change the outputs at time $t+1$.

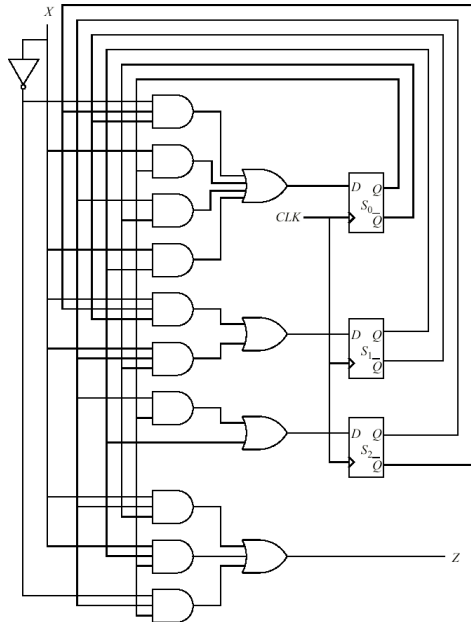
	Q_t	Q_{t+1}	S	R
<i>S-R</i> flip-flop	0	0	0	0
	0	1	1	0
	1	0	0	1
	1	1	0	0

	Q_t	Q_{t+1}	D
<i>D</i> flip-flop	0	0	0
	0	1	1
	1	0	0
	1	1	1

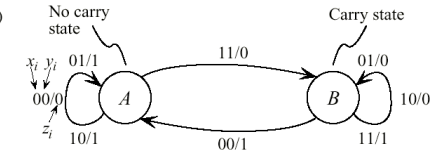
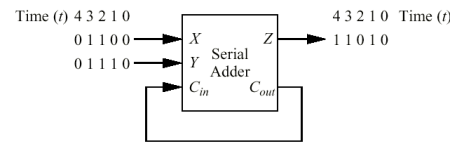
	Q_t	Q_{t+1}	J	K
<i>J-K</i> flip-flop	0	0	0	<i>d</i>
	0	1	1	<i>d</i>
	1	0	<i>d</i>	1
	1	1	<i>d</i>	0

	Q_t	Q_{t+1}	T
<i>T</i> flip-flop	0	0	0
	0	1	1
	1	0	1
	1	1	0

Sequence Detector Circuit



Serial Adder



- State transition diagram, state table, and state assignment for a serial adder.

Present state	Input XY			
	00	01	10	11
A	A/0	A/1	A/1	B/0
B	A/1	B/0	B/0	B/1

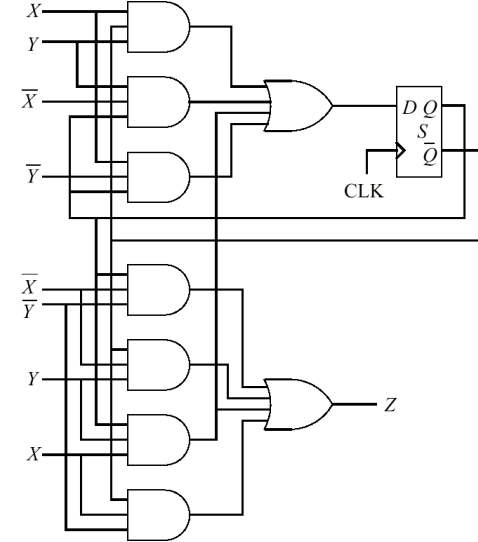
Present state (S_i)	Input XY			
	00	01	10	11
$A:0$	0/0	0/1	0/1	1/0
$B:1$	0/1	1/0	1/0	1/1

Serial Adder Next-State Functions

- Truth table showing next-state functions for a serial adder for D, S-R, T, and J-K flip-flops. Shaded functions are used in the example.

Present State			(Set) (Reset)						
X	Y	S _i	D	S	R	T	J	K	Z
0	0	0	0	0	0	0	0	d	0
0	0	1	0	0	1	1	d	1	1
0	1	0	0	0	0	0	0	d	1
0	1	1	1	0	0	0	d	0	0
1	0	0	0	0	0	0	0	d	1
1	0	1	1	0	0	0	d	0	0
1	1	0	1	1	0	1	1	d	0
1	1	1	1	0	0	0	d	0	1

D Flip-Flop Serial Adder Circuit

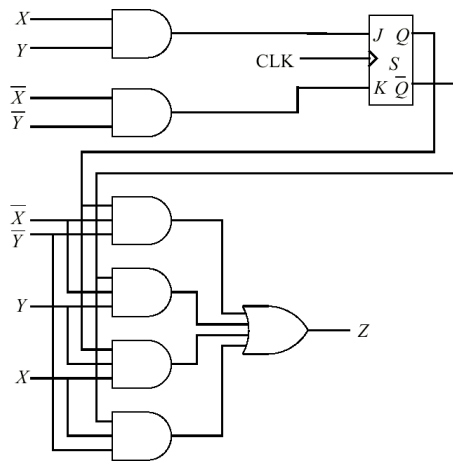


J-K Flip-Flop Serial Adder Circuit

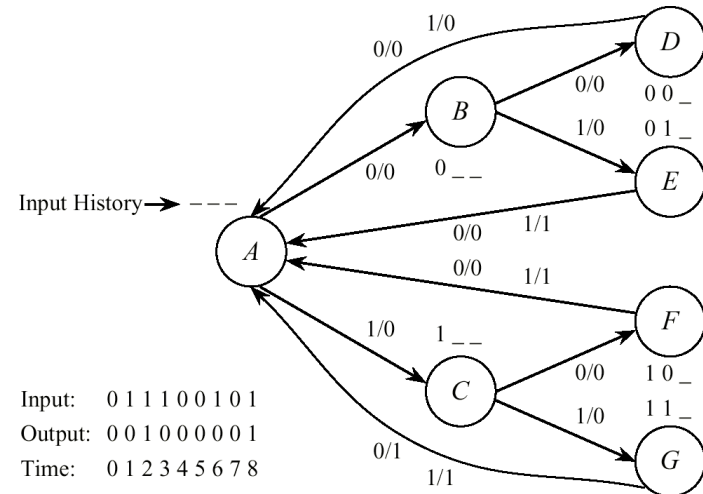
$$J = XY$$

$$K = \bar{X}\bar{Y}$$

$$Z = \bar{X}\bar{Y}S + \bar{X}Y\bar{S} + XY\bar{S} + X\bar{Y}S$$



Majority Finite State Machine



Input: 0 1 1 1 0 0 1 0 1
 Output: 0 0 1 0 0 0 0 0 1
 Time: 0 1 2 3 4 5 6 7 8

Majority FSM State Table

- (a) State table for majority FSM; (b) partitioning; (c) reduced state table.

Input P.S.	X	
	0	1
A	B/0 C/0	
B	D/0 E/0	
C	F/0 G/0	
D	A/0 A/0	
E	A/0 A/1	
F	A/0 A/1	
G	A/1 A/1	

$$P_0 = (ABCDEFGG)$$

$$P_1 = (ABCD)(EF)(G)$$

$$P_2 = (AD)(B)(C)(EF)(G)$$

$$P_3 = (A)(B)(C)(D)(EF)(G)$$

$$P_4 = (A)(B)(C)(D)(EF)(G) \checkmark$$

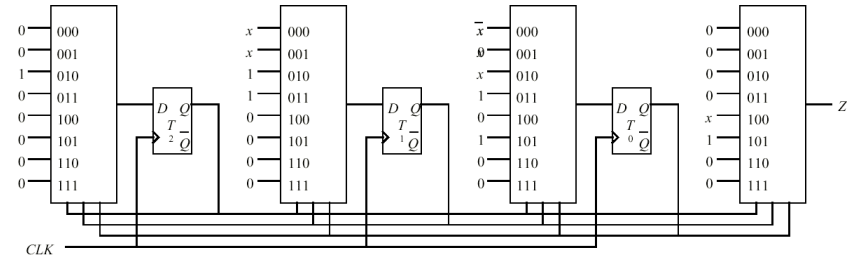
Input P.S.	X	
	0	1
A: A'	B'/0 C'/0	
B: B'	D'/0 E'/0	
C: C'	E'/0 F'/0	
D: D'	A'/0 A'/0	
EF: E'	A'/0 A'/1	
G: F'	A'/1 A'/1	

(a)

(b)

(c)

Majority FSM Circuit



Majority FSM State Assignment

- (a) State assignment for reduced majority FSM using D flip-flops; and (b) using T flip-flops.

Input P.S.	X	
	0	1
$S_2S_1S_0$	$S_2S_1S_0Z$	$S_2S_1S_0Z$
A': 000	001/0	010/0
B': 001	011/0	100/0
C': 010	100/0	101/0
D': 011	000/0	000/0
E': 100	000/0	000/1
F': 101	000/1	000/1

(a)

Input P.S.	X	
	0	1
$S_2S_1S_0$	$T_2T_1T_0Z$	$T_2T_1T_0Z$
A': 000	001/0	010/0
B': 001	000/0	010/0
C': 010	110/0	111/0
D': 011	011/0	011/0
E': 100	100/0	100/1
F': 101	101/1	101/1

(b)