

Appendix A: Digital Logic

**Miles Murdocca
Rutgers, The State University
of New Jersey**

Topics

- A.1 Combinational Logic**
- A.2 Truth Tables**
- A.3 Logic Gates**
- A.4 Properties of Boolean Algebra**
- A.5 The Sum-of-Products Form and Logic Diagrams**
- A.6 The Product-of-Sums Form**
- A.7 Positive versus Negative Logic**
- A.8 The Data Sheet**
- A.9 Digital Components**
- A.10 Reduction of Two-Level Expressions**
- A.11 Speed and Performance**
- A.12 Sequential Logic**
- A.13 J-K and T Flip-Flops**
- A.14 Design of Finite State Machines**
- A.15 Mealy versus Moore Machines**
- A.16 Registers**
- A.17 Counters**

Some Definitions

- **Combinational logic:** a digital logic circuit in which logical decisions are made based only on combinations of the inputs (e.g., an adder).
- **Sequential logic:** a circuit in which decisions are made based on combinations of the current inputs as well as the past history of inputs (e.g., a memory unit).
- **Finite state machine:** a circuit which has an internal state, and whose outputs are functions of both current inputs and its internal state (e.g., a vending machine controller).

The Combinational Logic Unit

- Translates a set of inputs into a set of outputs according to one or more mapping functions.
- Inputs and outputs for a CLU normally have two distinct (binary) values: high and low, 1 and 0, 0 and 1, or 5 v and 0 v, for example.
- The outputs of a CLU are strictly functions of the inputs, and the outputs are updated immediately after the inputs change. A set of inputs i_0 – i_n are presented to the CLU, which produces a set of outputs according to mapping functions f_0 – f_m .

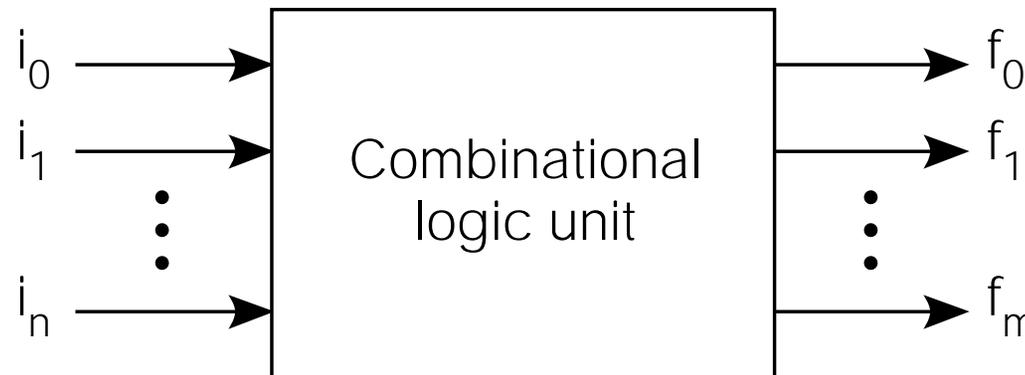


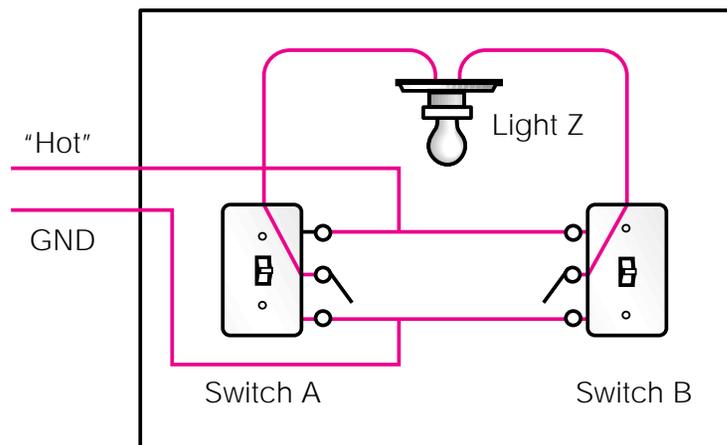
Fig A.1

Truth Tables

- Developed in 1854 by George Boole
- Further developed by Claude Shannon (Bell Labs)
- Outputs are computed for all possible input combinations (how many input combinations are there?)

Consider a room with two light switches. How must they work†?

Fig. A.2



Inputs		Output
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

†Don't show this to your electrician, or wire your house this way. This circuit definitely violates the electric code. The practical circuit never leaves the lines to the light "hot" when the light is turned off. Can you figure how?

Fig A.4 Truth Tables Showing All Possible Functions of Two Binary Variables

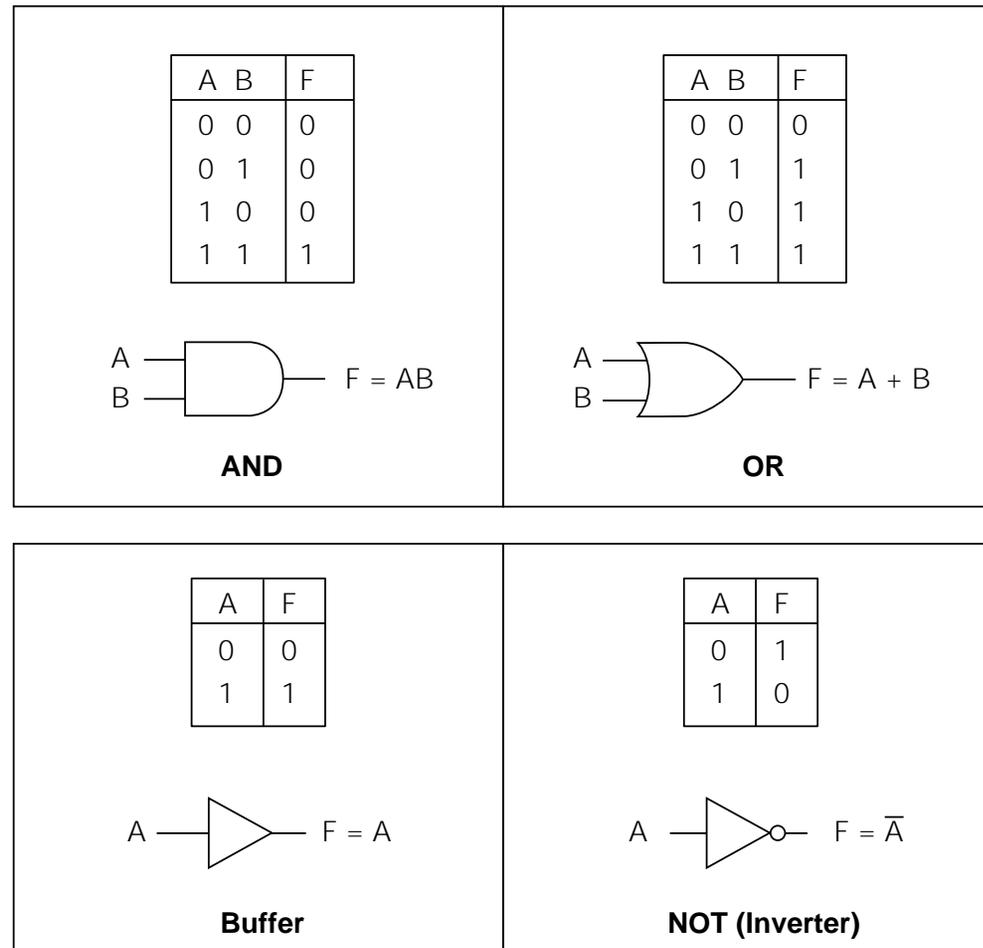
<i>A</i>	<i>B</i>	<i>False</i>	<i>AND</i>	$\overline{A\overline{B}}$	<i>A</i>	$\overline{A}B$	<i>B</i>	<i>XOR</i>	<i>OR</i>
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

<i>A</i>	<i>B</i>	<i>NOR</i>	<i>XNOR</i>	\overline{B}	$A + \overline{B}$	\overline{A}	$\overline{A} + B$	<i>NAND</i>	<i>True</i>
0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

- The more frequently used functions have names: AND, XOR, OR, NOR, XNOR, and NAND. (Always use upper-case spelling.)

Logic Gates and Their Symbols

Fig. A.5 Logic Gate Symbols for *AND*, *OR*, Buffer, and *NOT* Boolean functions



- Note the use of the “inversion bubble.”
- Be careful about the “nose” of the gate when drawing **AND** vs. **OR**.

Fig A.6 Logic Gate Symbols for *NAND*, *NOR*, *XOR*, and *XNOR* Boolean functions

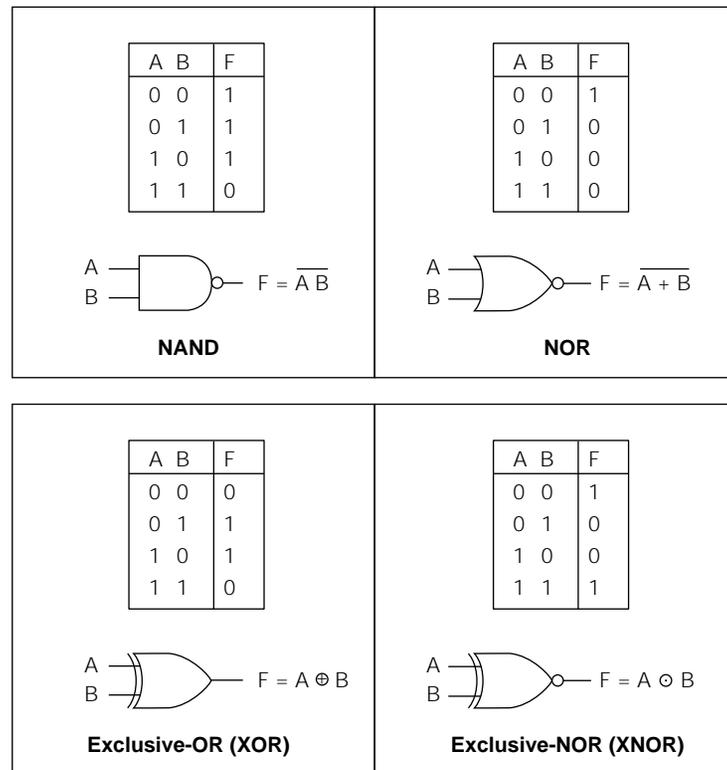
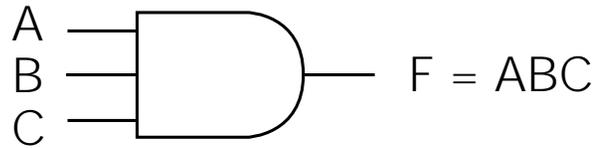
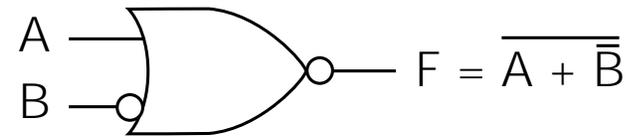


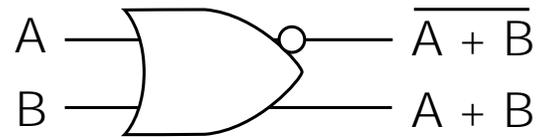
Fig A.7 Variations of Basic Logic Gate Symbols



(a)



(b)



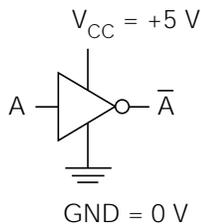
(c)

(a) 3 inputs

(b) A negated input

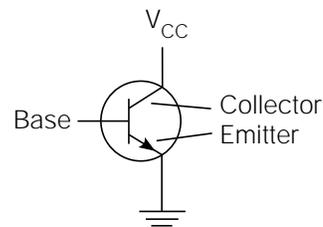
(c) Complementary outputs

Fig A.8 The Inverter at the Transistor Level



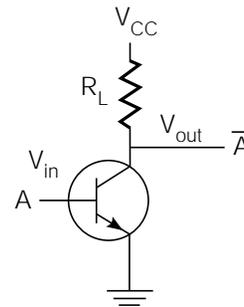
(a)

**Power
terminals**



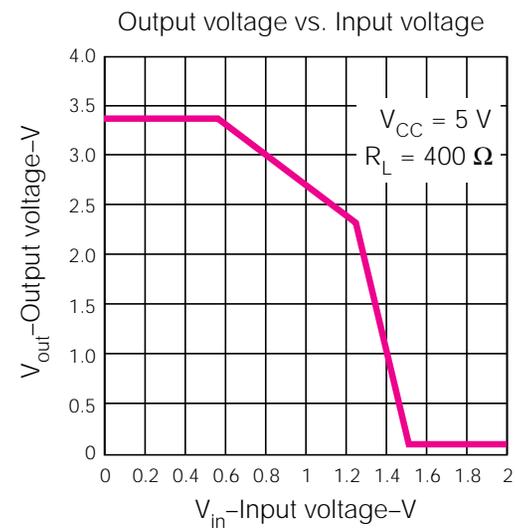
(b)

**Transistor
symbol**



(c)

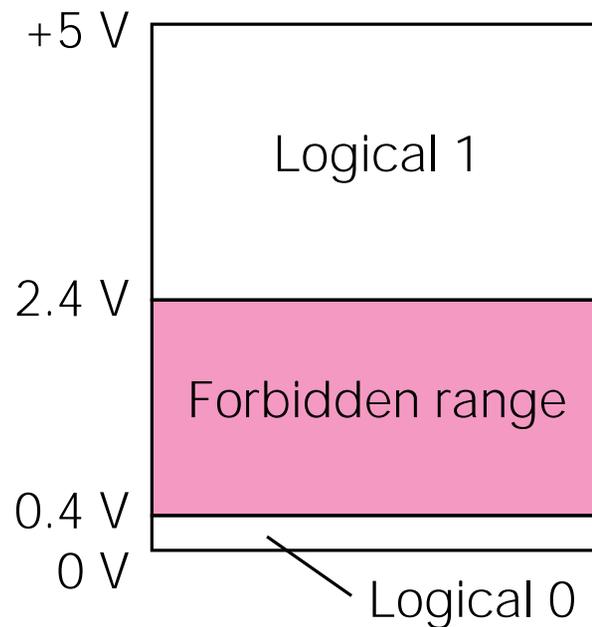
**A transistor used
as an inverter**



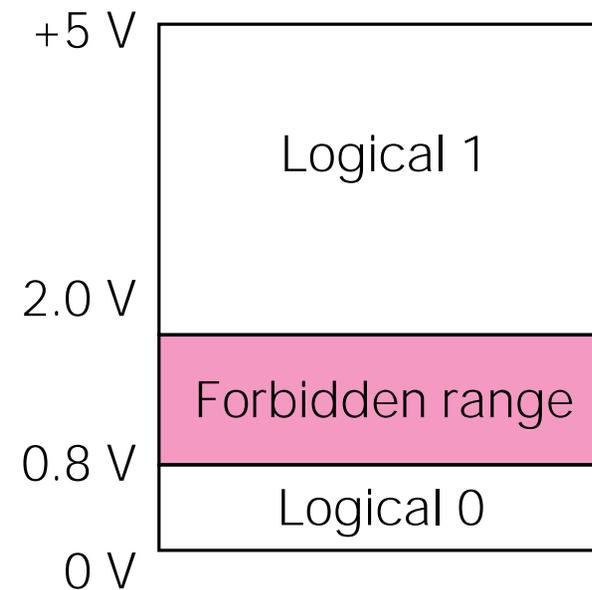
(d)

**Inverter transfer
function**

Fig A.9 Assignments of Logical 0 and Logical 1 to Voltage Ranges

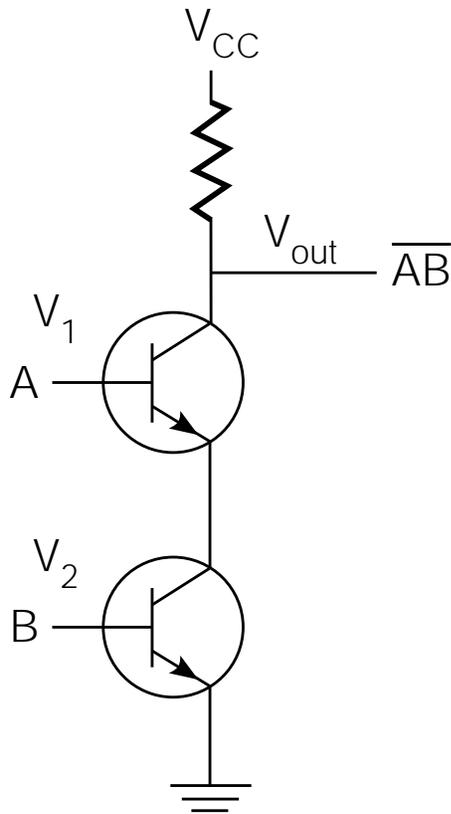


(a) At the output of a logic gate

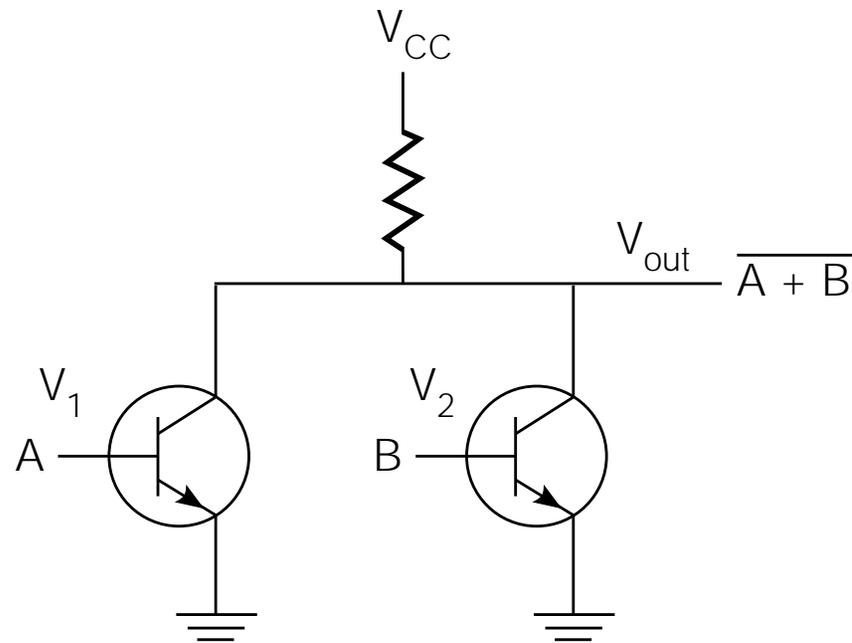


(b) At the input to a logic gate

A.10 Transistor Circuits



(a) A two-input NAND gate



(b) A two-input NOR gate

Tbl A.1 The Basic Properties of Boolean Algebra

Principle of duality: The dual of a Boolean function is gotten by replacing AND with OR and OR with AND, constant 1s by 0s, and 0s by 1s

Relationship	Dual	Property
$A B = B A$	$A + B = B + A$	Commutative
$A (B + C) = A B + A C$	$A + B C = (A + B) (A + C)$	Distributive
$1 A = A$	$0 + A = A$	Identity
$A \bar{A} = 0$	$A + \bar{A} = 1$	Inverse
$0 A = 0$	$1 + A = 1$	Null
$A A = A$	$A + A = A$	Idempotence
$A (B C) = (A B) C$	$A + (B + C) = (A + B) + C$	Associative
$\overline{\bar{A}} = A$		Complement
$\overline{A B} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A} \bar{B}$	DeMorgan's Theorem
$AB + \bar{A}C + BC = AB + \bar{A}C$	$(A + B)(\bar{A} + C)(B + C) = (A + B)(\bar{A} + C)$	Consensus Theorem

Postulates



Theorems

A, B, etc. are Literals; 0 and 1 are constants.

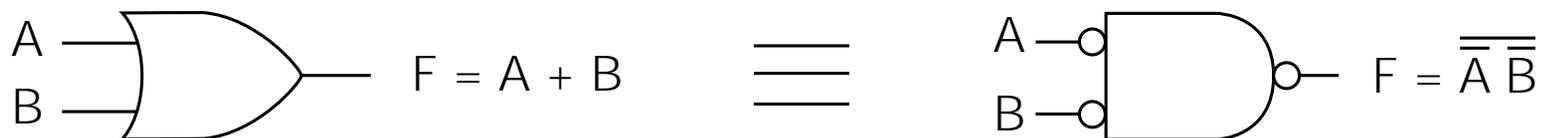
DeMorgan's Theorem

Fig A.11

A	B	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A} \overline{B}$
0	0	1	1
0	1	1	0
1	0	1	0
1	1	0	0

DeMorgan's theorem: $A + B = \overline{\overline{A + B}} = \overline{\overline{A} \overline{B}}$

Fig A.12

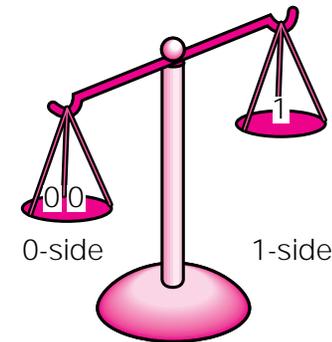


Discuss: Applying DeMorgan's theorem by “pushing the bubbles” and “bubble tricks.”

The Sum-of-Products (SOP) Form

Fig. A.14 Truth Table for the Majority Function

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



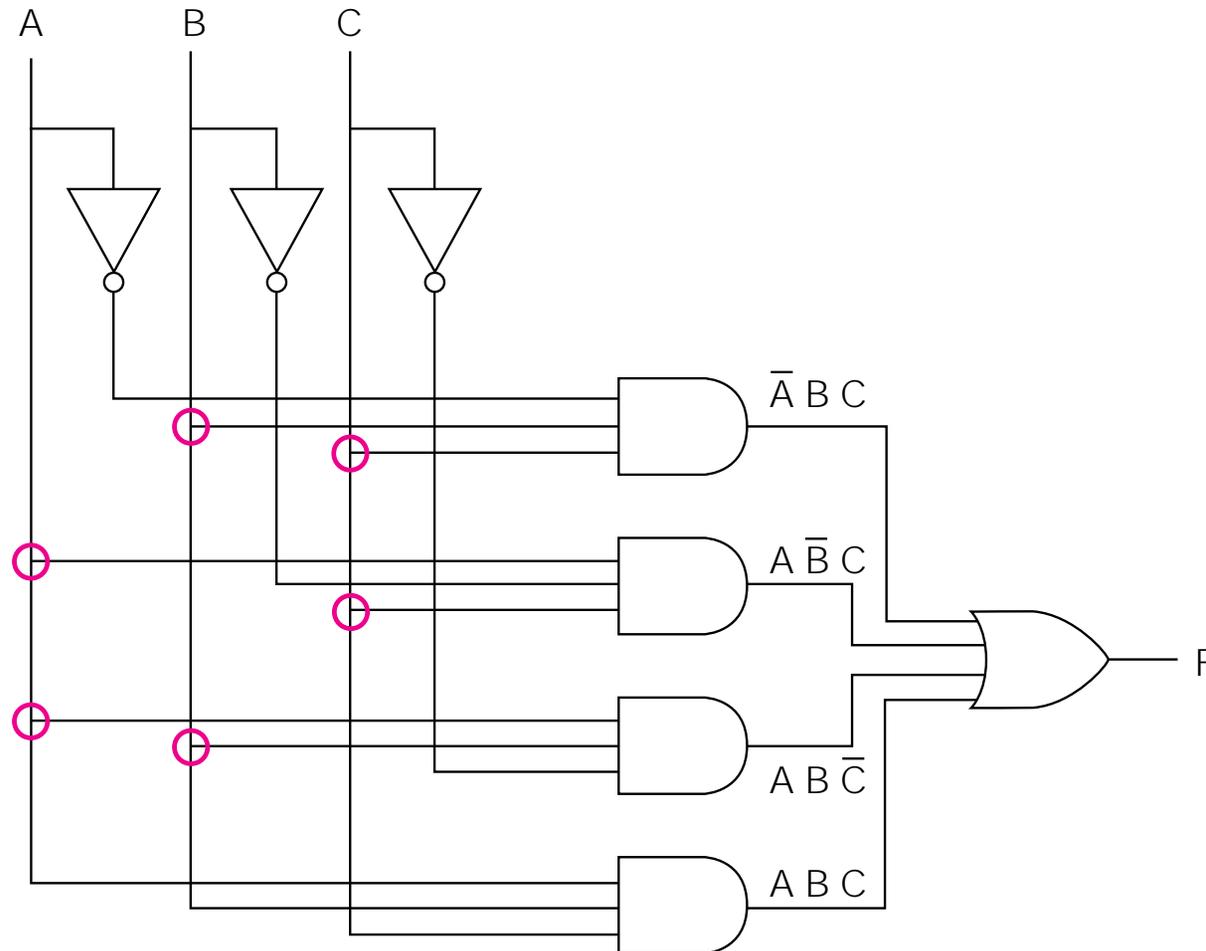
A balance tips to the left or right depending on whether there are more 0's or 1's.

- Transform the function into a two-level AND-OR equation
- Implement the function with an arrangement of logic gates from the set {AND, OR, NOT}
- M is true when $A = 0$, $B = 1$, and $C = 1$, or when $A = 1$, $B = 0$, and $C = 1$, and so on for the remaining cases.
- Represent logic equations by using the sum-of-products (SOP) form

The SOP Form of the Majority Gate

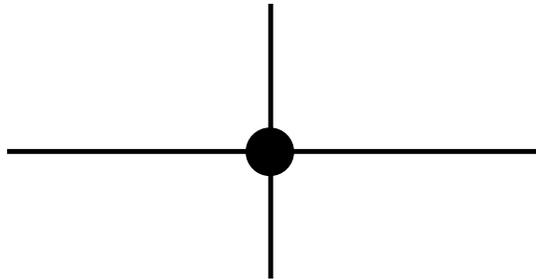
- The SOP form for the 3-input majority gate is:
- $M = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC = m_3 + m_5 + m_6 + m_7 = \Sigma(3, 5, 6, 7)$
- Each of the 2^n terms are called minterms, running from 0 to $2^n - 1$
- Note the relationship between minterm number and Boolean value.
- Discuss: common-sense interpretation of equation.

Fig A.15 A Two-Level AND-OR Circuit Implements the Majority Function

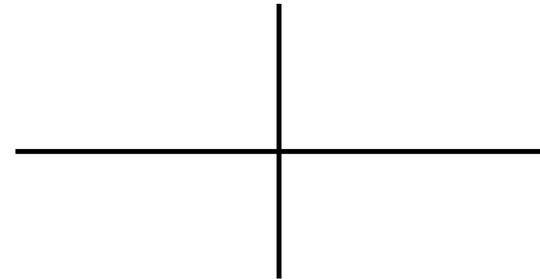


Discuss: what is the gate count?

Fig A.16 Four Notations Used at Circuit Intersections



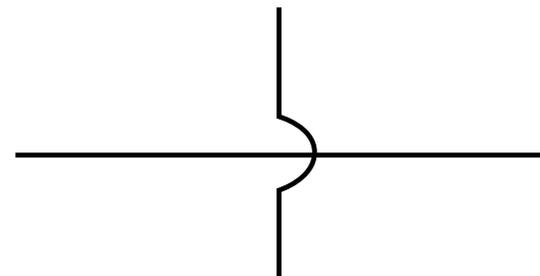
Connection



No connection

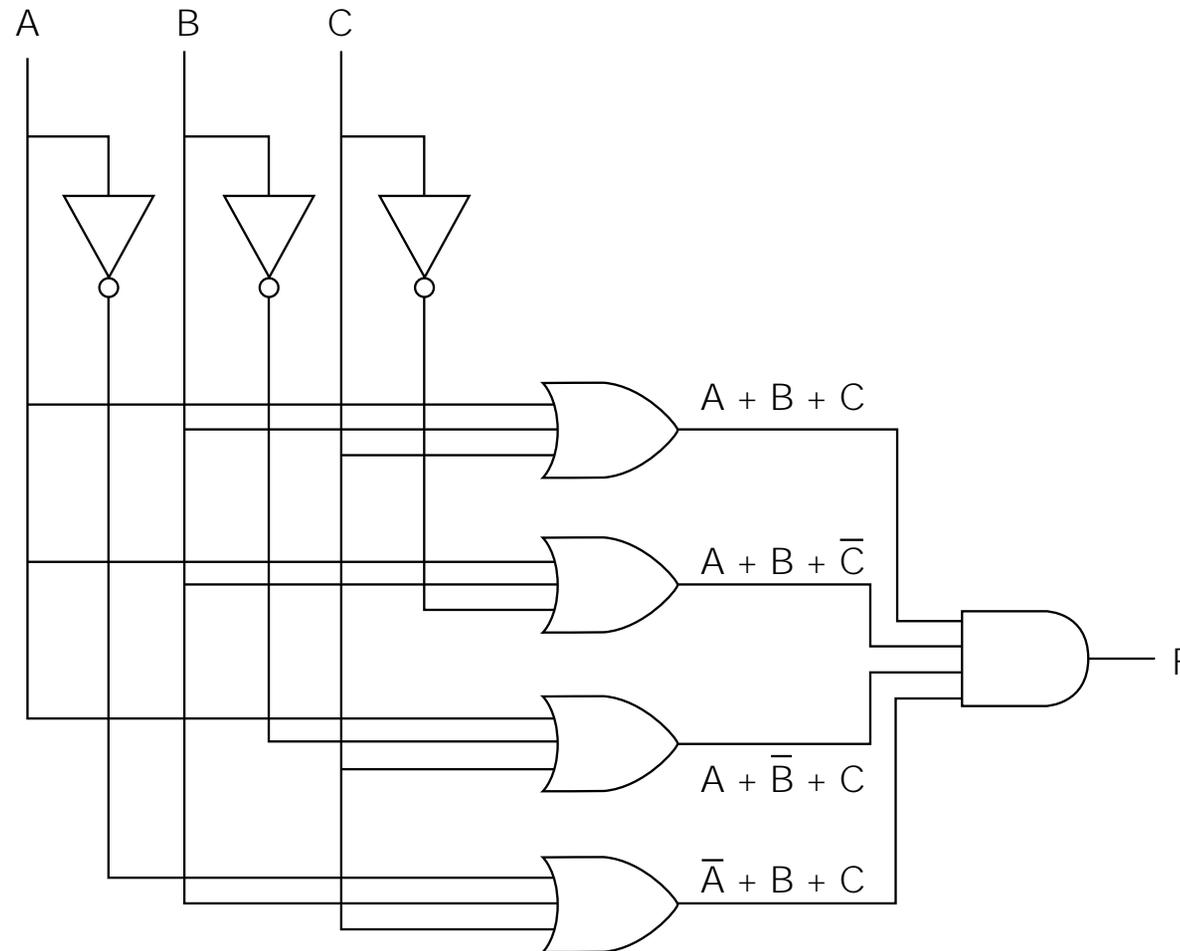


Connection



No connection

Fig A.17 A Two-Level OR-AND Circuit that Implements the Majority Function



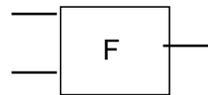
Positive versus Negative Logic

- Positive logic: truth, or assertion is represented by logic 1, higher voltage; falsity, de- or unassertion, logic 0, is represented by lower voltage.
- Negative logic: truth, or assertion is represented by logic 0, lower voltage; falsity, de- or unassertion, logic 1, is represented by higher voltage

Gate Logic: Positive vs. Negative Logic

Normal Convention: Positive Logic/Active High
Low Voltage = 0; High Voltage = 1

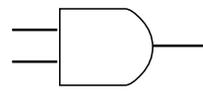
Alternative Convention sometimes used: Negative Logic/Active Low



Voltage Truth Table

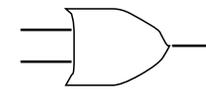
A	B	F
low	low	low
low	high	low
high	low	low
high	high	high

Behavior in terms
of Electrical Levels



Positive Logic

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1



Negative Logic

A	B	F
1	1	1
1	0	1
0	1	1
0	0	0

Two Alternative Interpretations
Positive Logic AND
Negative Logic OR

Dual Operations

Fig A.18 Positive and Negative Logic Assignments

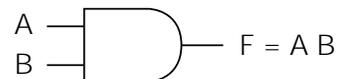
Voltage levels

A	B	F
low	low	low
low	high	low
high	low	low
high	high	high



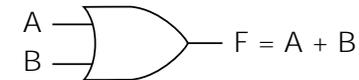
Positive logic levels

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1



Negative logic levels

A	B	F
1	1	1
1	0	1
0	1	1
0	0	0



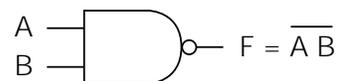
Voltage levels

A	B	F
low	low	high
low	high	high
high	low	high
high	high	low



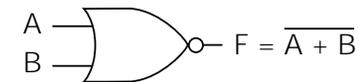
Positive logic levels

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0



Negative logic levels

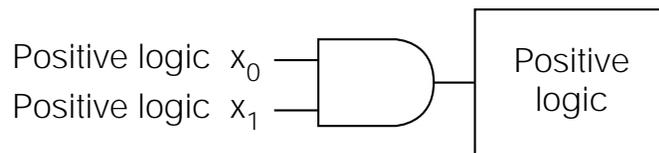
A	B	F
1	1	0
1	0	0
0	1	0
0	0	1



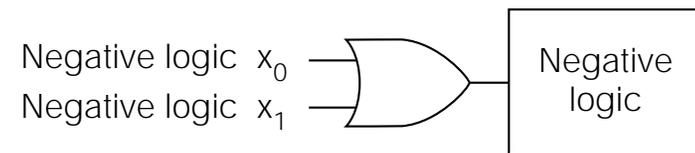
Bubble Matching

- Active-low signals are signified by a prime or overbar or /.
- Active high: enable
- Active low: enable', $\overline{\text{enable}}$, enable/
- Discuss microwave oven control:
 - Active high: Heat = DoorClosed • Start
 - Active low: ? (Hint: begin with AND gate as before)

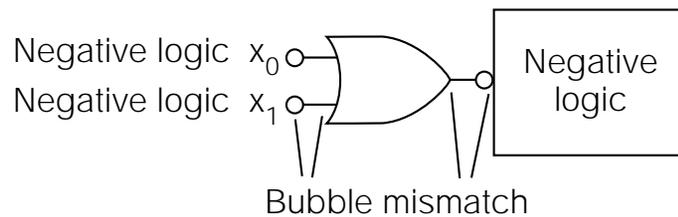
Fig. A.19 The Process of Bubble Matching



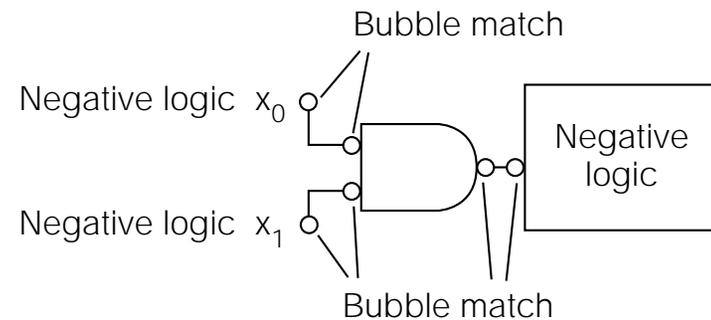
(a)



(b)



(c)



(d)

Digital Components

- **High-level digital circuit designs are normally made using collections of logic gates referred to as components, rather than using individual logic gates. The majority function can be viewed as a component.**
- **Levels of integration (numbers of gates) in an integrated circuit (IC):**
 - **Small-scale integration (SSI): 10–100 gates.**
 - **Medium-scale integration (MSI): 100–1000 gates.**
 - **Large-scale integration (LSI): 1000–10,000 logic gates.**
 - **Very large scale integration (VLSI): 10,000–upward.**
- **These levels are approximate, but the distinctions are useful in comparing the relative complexity of circuits.**
- **Let us consider several useful MSI components.**

Fig A.20 Simplified Data Sheet

SN7400 QUADRUPLE 2-INPUT POSITIVE-NAND GATES

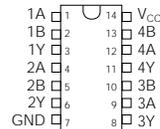
description

These devices contain four independent 2-input NAND gates.

function table (each gate)

INPUTS		OUTPUT
A	B	Y
H	H	L
L	X	H
X	L	H

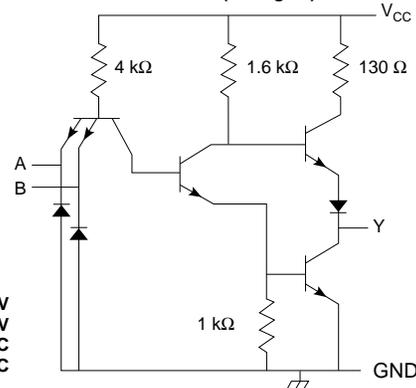
package (top view)



absolute maximum ratings

Supply voltage, V_{CC} 7 V
 Input voltage: 5.5 V
 Operating free-air temperature range: 0°C to 70°C
 Storage temperature range: -65°C to 150°C

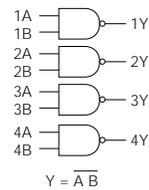
schematic (each gate)



recommended operating conditions

	MIN	NOM	MAX	UNIT
V_{CC} Supply voltage	4.75	5	5.25	V
V_{IH} High-level input voltage	2			V
V_{IL} Low-level input voltage			0.8	V
I_{OH} High-level output current			-0.4	mA
I_{OL} Low-level output current			16	mA
T_A Operating free-air temperature	0		70	°C

logic diagram (positive logic)



electrical characteristics over recommended operating free-air temperature range

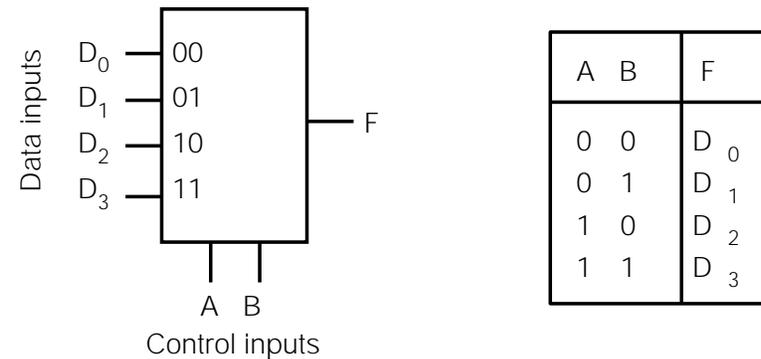
VALUE	OPERATING CONDITIONS	MIN	TYP	MAX	UNIT
V_{OH}	$V_{CC} = \text{MIN}, V_{IL} = 0.8 \text{ V}, I_{OH} = -0.4 \text{ mA}$	2.4	3.4		V
V_{OL}	$V_{CC} = \text{MIN}, V_{IH} = 2 \text{ V}, I_{OL} = 16 \text{ mA}$		0.2	0.4	V
I_{IH}	$V_{CC} = \text{MAX}, V_I = 2.4 \text{ V}$			40	μA
I_{IL}	$V_{CC} = \text{MAX}, V_I = 0.4 \text{ V}$			-1.6	mA
I_{CCH}	$V_{CC} = \text{MAX}, V_I = 0 \text{ V}$		4	8	mA
I_{CCL}	$V_{CC} = \text{MAX}, V_I = 4.5 \text{ V}$		12	22	mA

switching characteristics, $V_{CC} = 5 \text{ V}, T_A = 25^\circ \text{ C}$

PARAMETER	FROM (input)	TO (output)	TEST CONDITIONS	MIN	NOM	MAX	UNIT
t_{PLH}	A or B	Y	$R_L = 400 \Omega$ $C_L = 15 \text{ pF}$		11	22	ns
t_{PHL}					7	15	ns

The Multiplexer

Fig A.21 Block Diagram and Truth Table



$$F = \bar{A} \bar{B} D_0 + \bar{A} B D_1 + A \bar{B} D_2 + A B D_3$$

Fig A.22 AND-OR Circuit Implementation

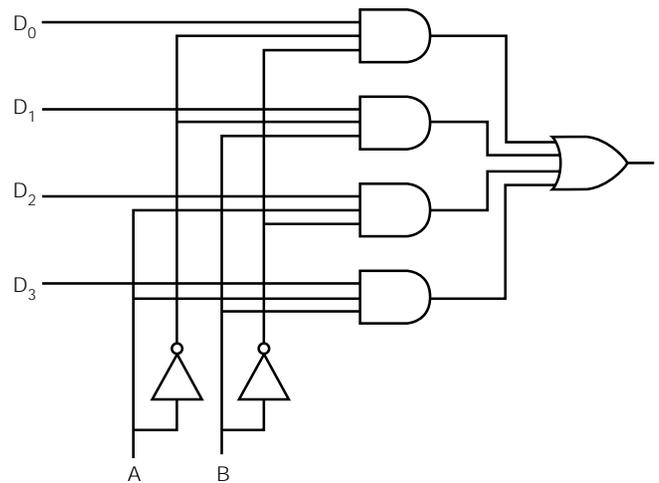
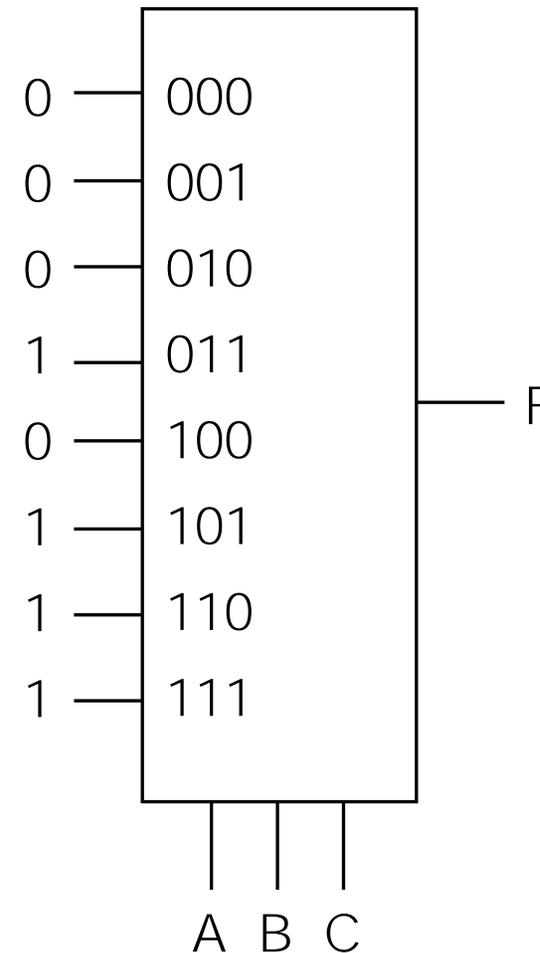


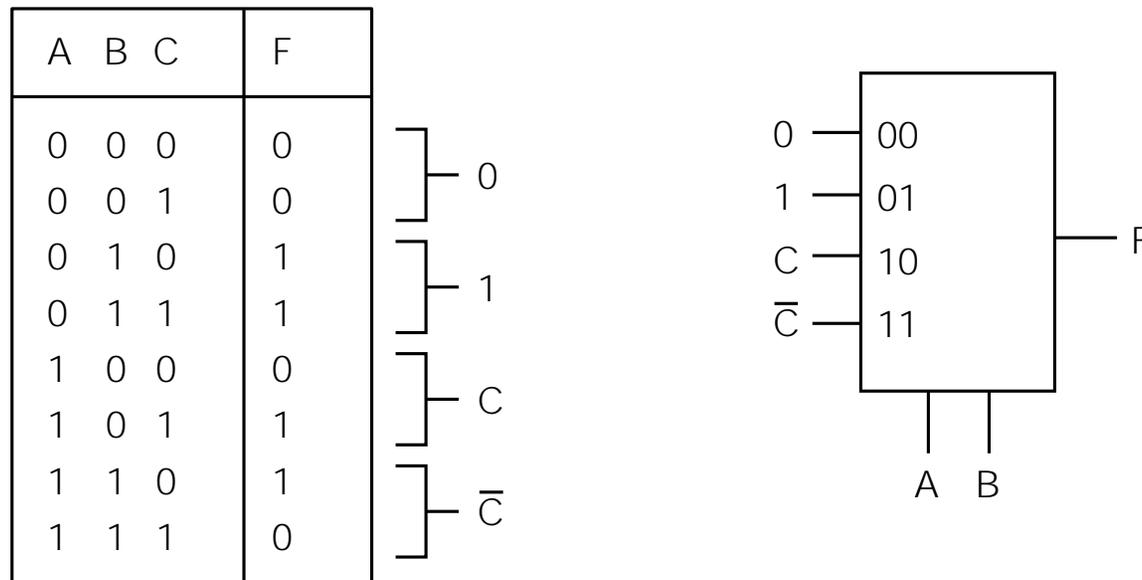
Fig A.23 An 8-1 MUX Implements the Majority Function

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Principle: Use the MUX select to pick out the selected minterms of the function.

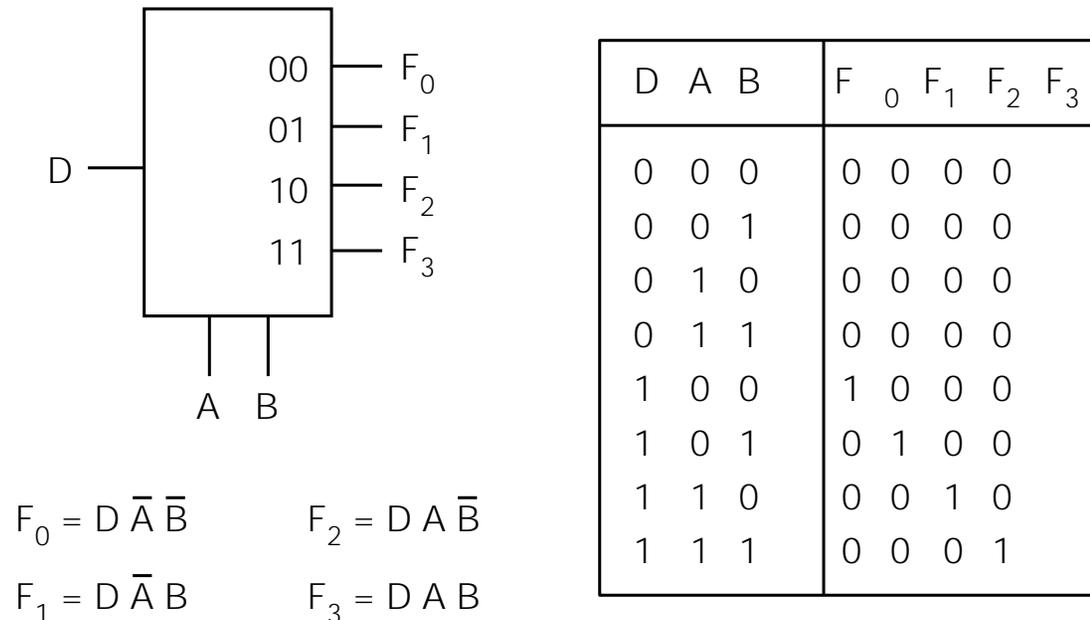
Fig A.24 A 4-1 MUX Implements a 3-Variable Function



Principle: Use the A and B inputs to select a pair of minterms. The value applied to the MUX input is selected from {0, 1, C, C} to pick the desired behavior of the minterm pair.

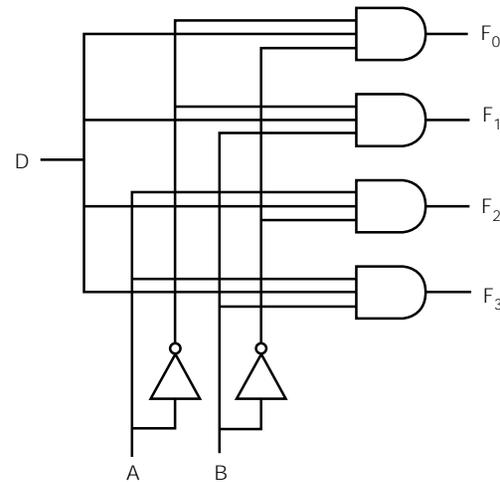
The Demultiplexer (DEMUX)

Fig A.25 Block Diagram and Truth Table



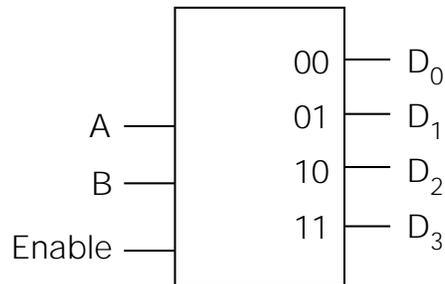
The Demultiplexer Is a Decoder with an Enable Input

Fig A.26 A Circuit for a 1-4 DEMUX



Compare to Fig A.28

Fig A.27 Block Diagram and Truth Table



		Enable = 1			
A	B	D ₀	D ₁	D ₂	D ₃
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

		Enable = 0			
A	B	D ₀	D ₁	D ₂	D ₃
0	0	0	0	0	0
0	1	0	0	0	0
1	0	0	0	0	0
1	1	0	0	0	0

$$D_0 = \bar{A} \bar{B}$$

$$D_1 = \bar{A} B$$

$$D_2 = A \bar{B}$$

$$D_3 = A B$$

Fig A.28 An AND Circuit for a 2-4 Decoder

Fig A.27

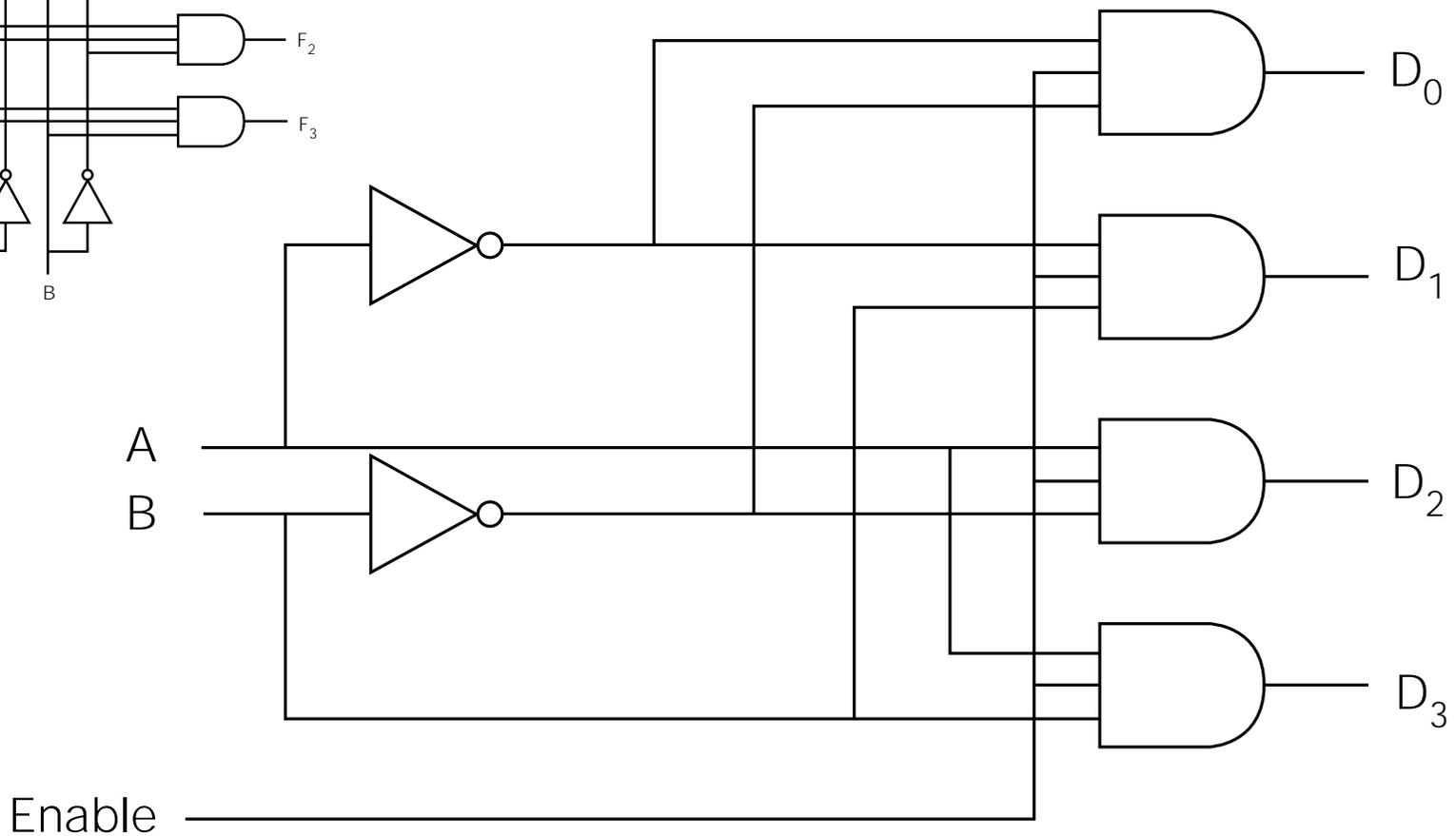
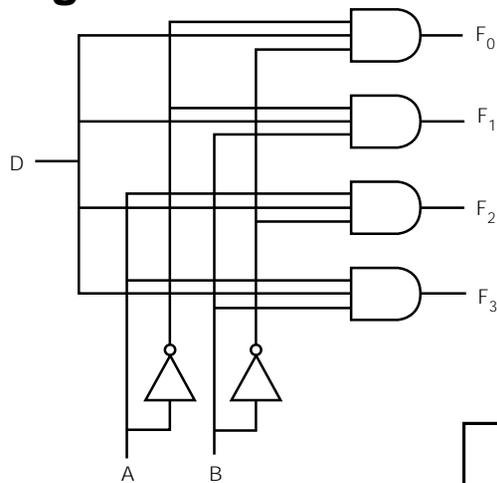
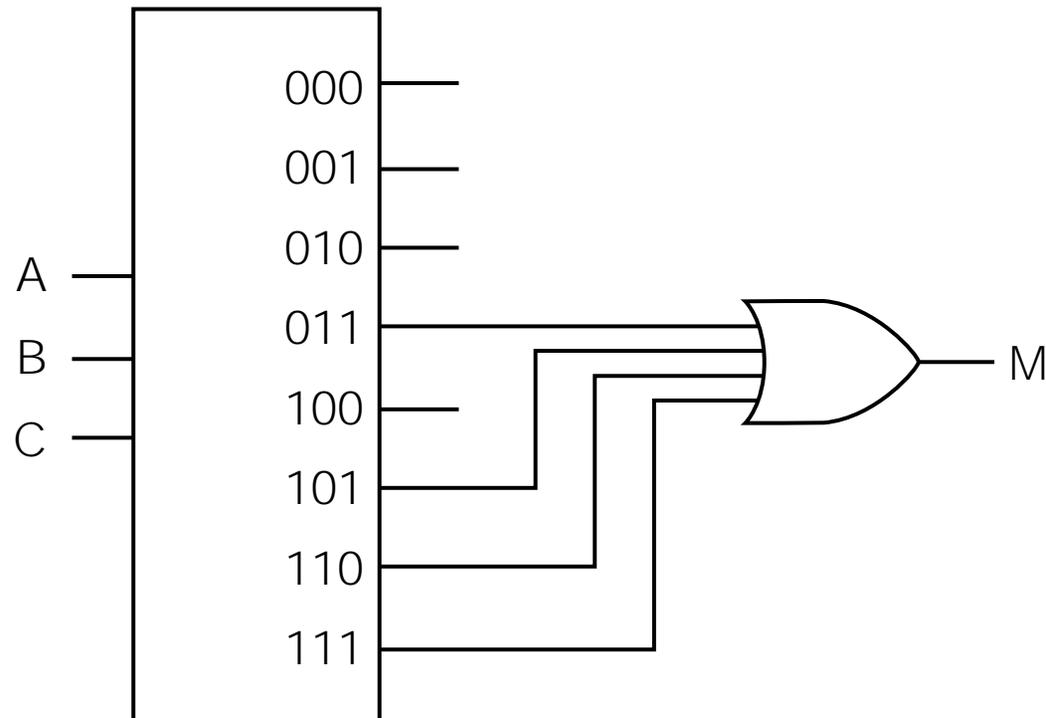


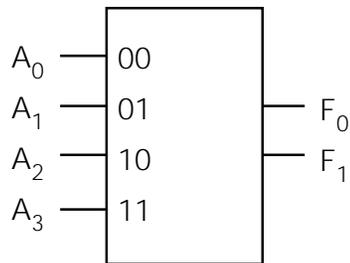
Fig A.29 A 3-to-8 Decoder Implements the Majority Function



The Priority Encoder

- An encoder translates a set of inputs into a binary encoding.
- Can be thought of as the converse of a decoder.
- A priority encoder imposes an order on the inputs.
- A_i has a higher priority than A_{i+1} .

Fig A.30 Block Diagram and Truth Table



$$F_0 = \bar{A}_0 \bar{A}_1 A_3 + \bar{A}_0 \bar{A}_1 A_2$$

$$F_1 = \bar{A}_0 \bar{A}_2 A_3 + \bar{A}_0 A_1$$

A_0	A_1	A_2	A_3	F_0	F_1
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	1	0
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	0

Fig A.31 Logic Diagram for a 4-to-2 Priority Encoder

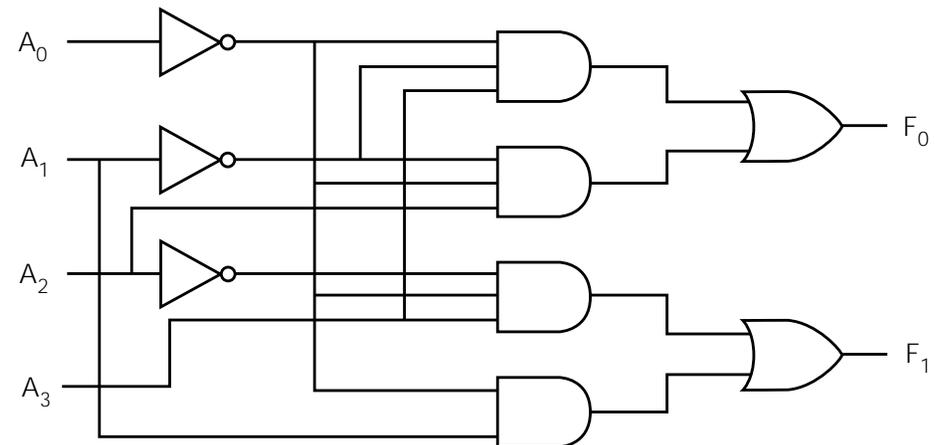


Fig A.32 A Programmable Logic Array

- A PLA is a customizable AND matrix followed by a customizable OR matrix

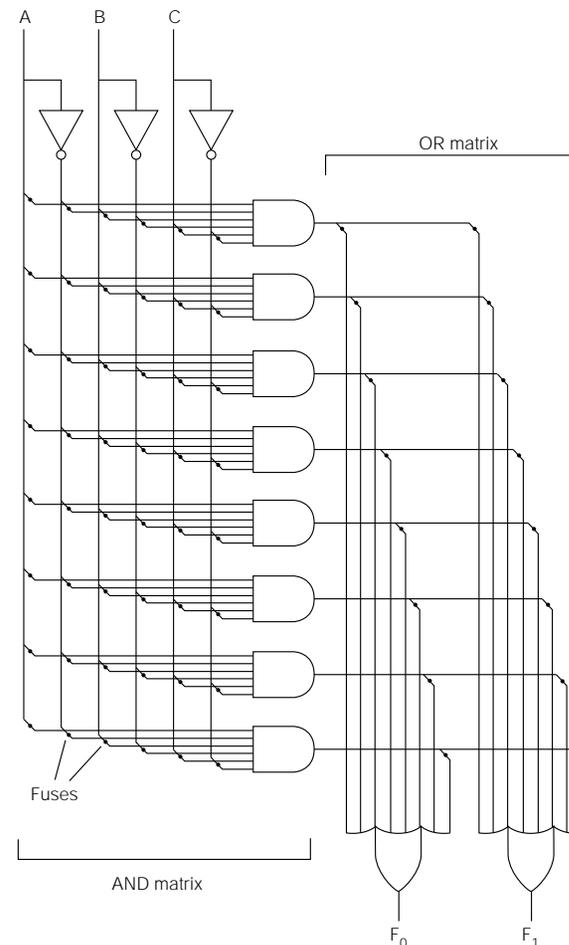
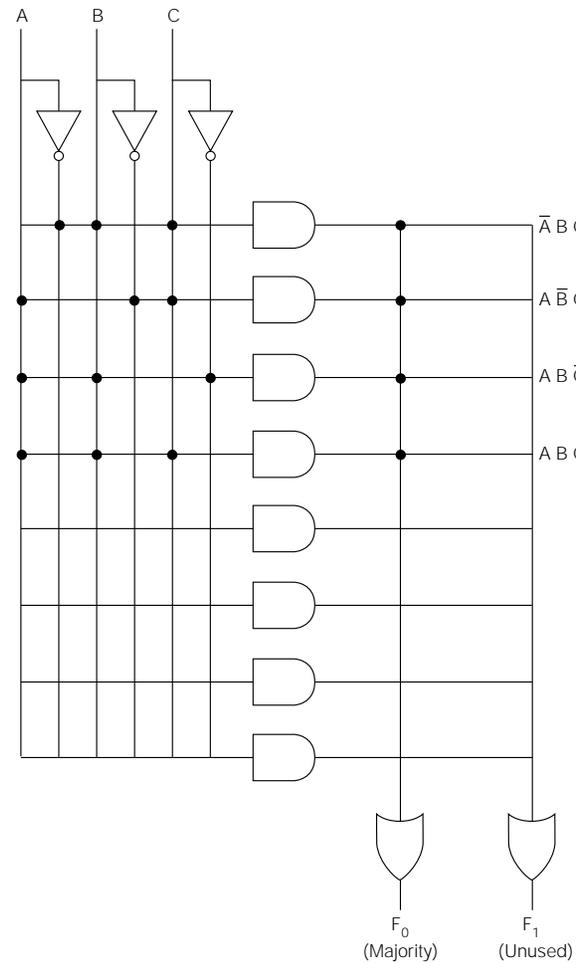
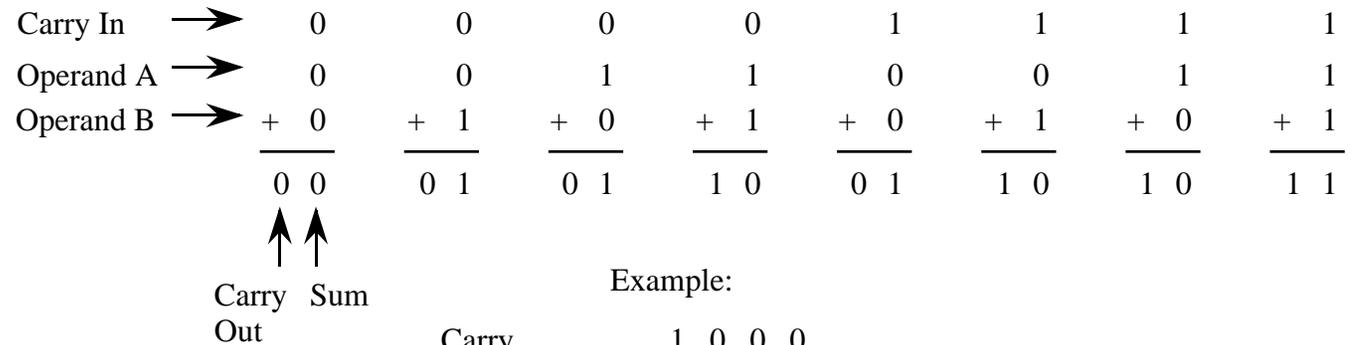


Fig A.33 Simplified Representation of a PLA



Using PLAs to Implement an Adder

Fig A.35 Addition for Two Unsigned Binary



Example:

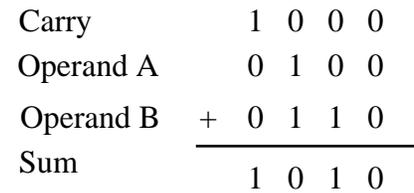


Fig A.34 Black Box Representation



Fig A.36 Truth Table for a Full Adder

A_i	B_i	C_i	S_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

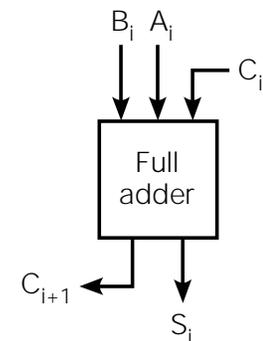


Fig A.37 A 4-Bit Adder Implemented with a Cascade of Full Adders

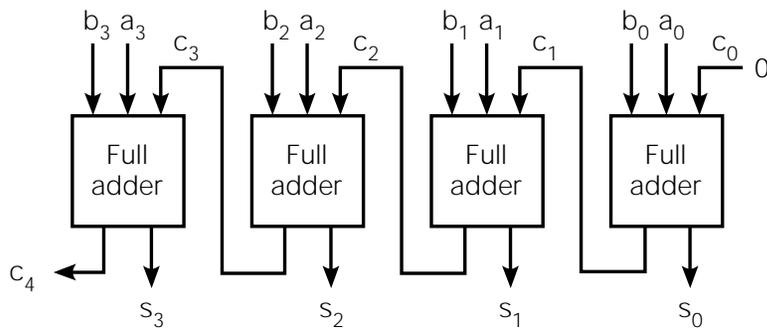
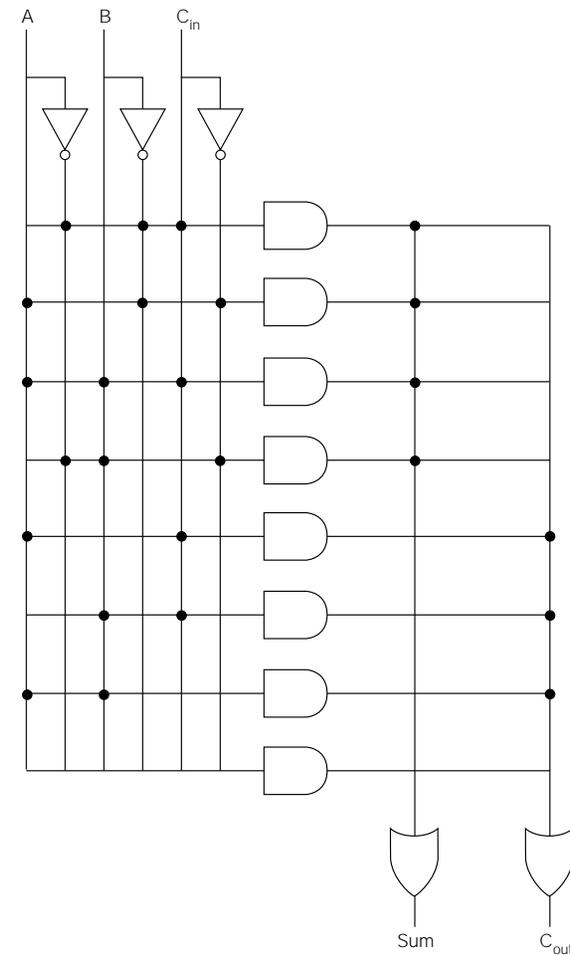


Fig A.38 PLA Realization of a Full Adder



Reduction (Simplification) of Boolean Expressions

- It may be possible to simplify the canonical SOP or POS forms.
- A smaller Boolean equation translates to a lower gate count in the target circuit.
- We discuss two methods: algebraic reduction and Karnaugh map reduction.

The Algebraic Method

Consider the majority function, F :

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

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

Distributive property

$$F = \bar{A}BC + A\bar{B}C + AB(1)$$

Complement property

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

Identity property

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

Idempotence property

$$F = \bar{A}BC + AC(\bar{B} + B) + AB$$

Identity property

$$F = \bar{A}BC + AC + AB$$

Complement and identity properties

$$F = \bar{A}BC + AC + AB + ABC$$

Idempotence property

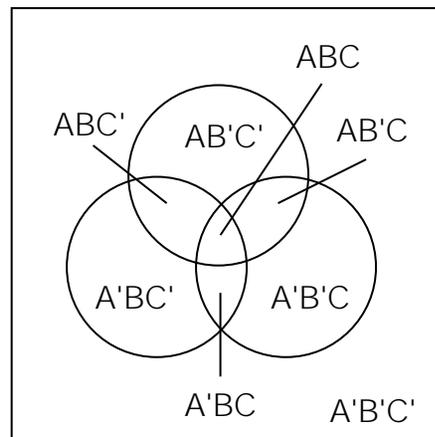
$$F = BC(\bar{A} + A) + AC + AB$$

Distributive property

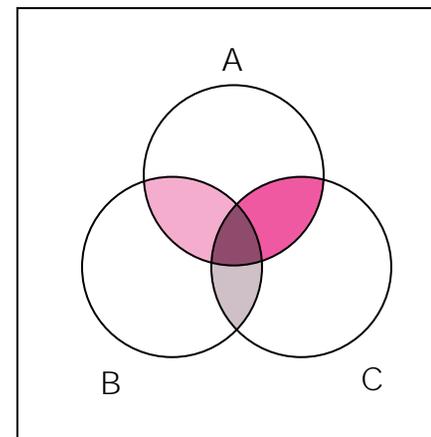
$$F = BC + AC + AB$$

Complement and identity properties

Fig A.40 Venn Diagrams



3 binary variables



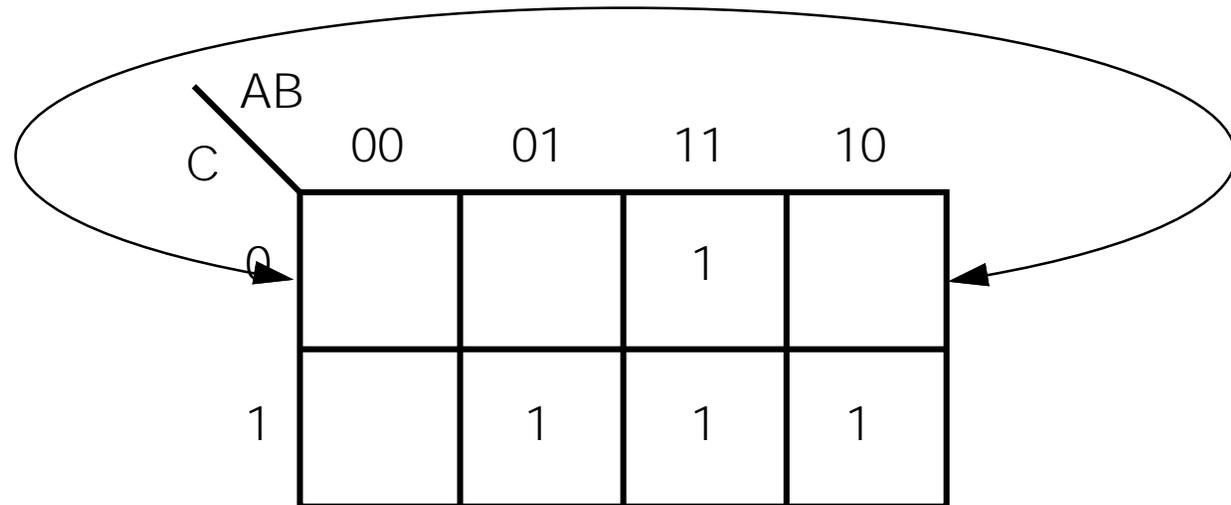
The majority function

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

Fig A.41 A K-Map of the Majority Function

Place a “1” in each cell that has a 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



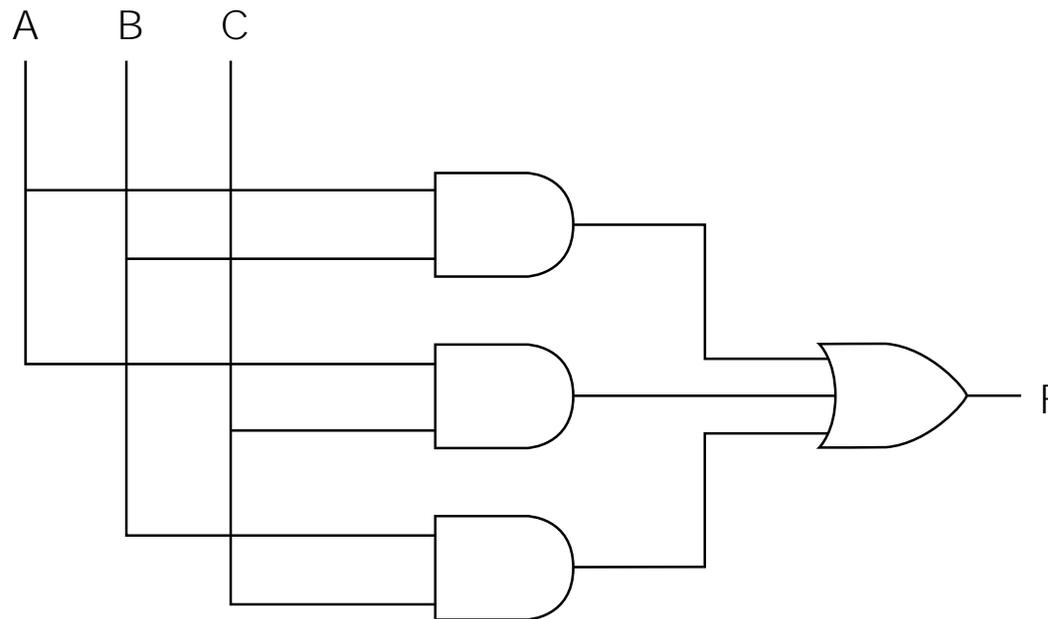
The map contains all the minterms. Adjacent 1's in the K-map satisfy the complement property of Boolean algebra.

Fig A.42 Adjacency Groupings for the Majority Function

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

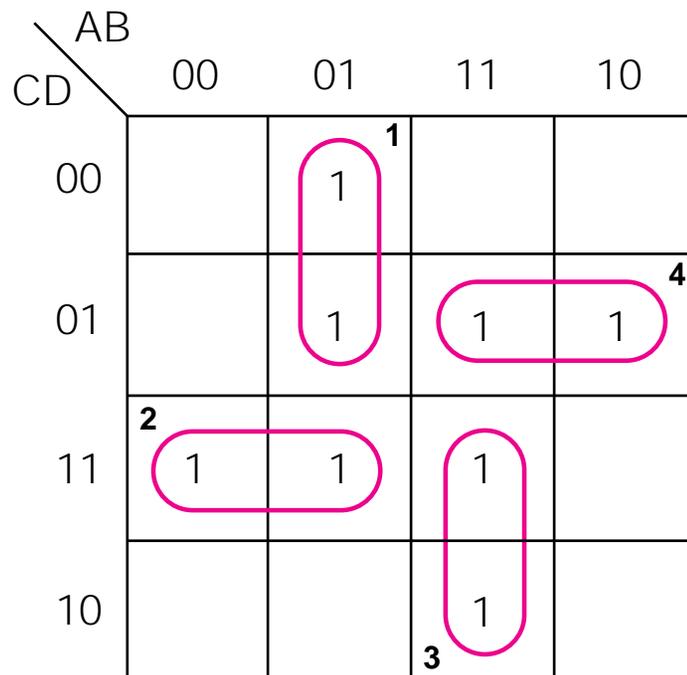
$$M = BC + AC + AB$$

A.43 Minimized AND-OR Circuit for the Majority Function

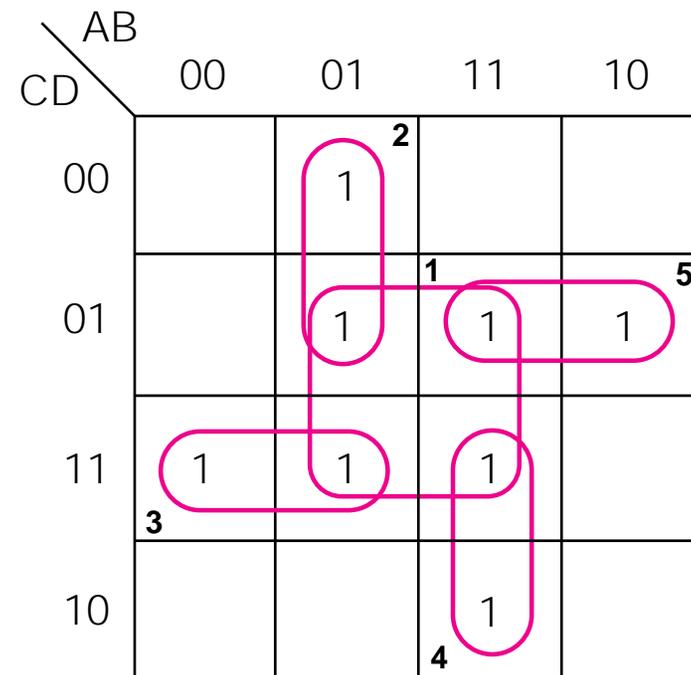


$$M = BC + AC + AB$$

Fig A.44 Minimal and Not-Minimal K-Map Groupings

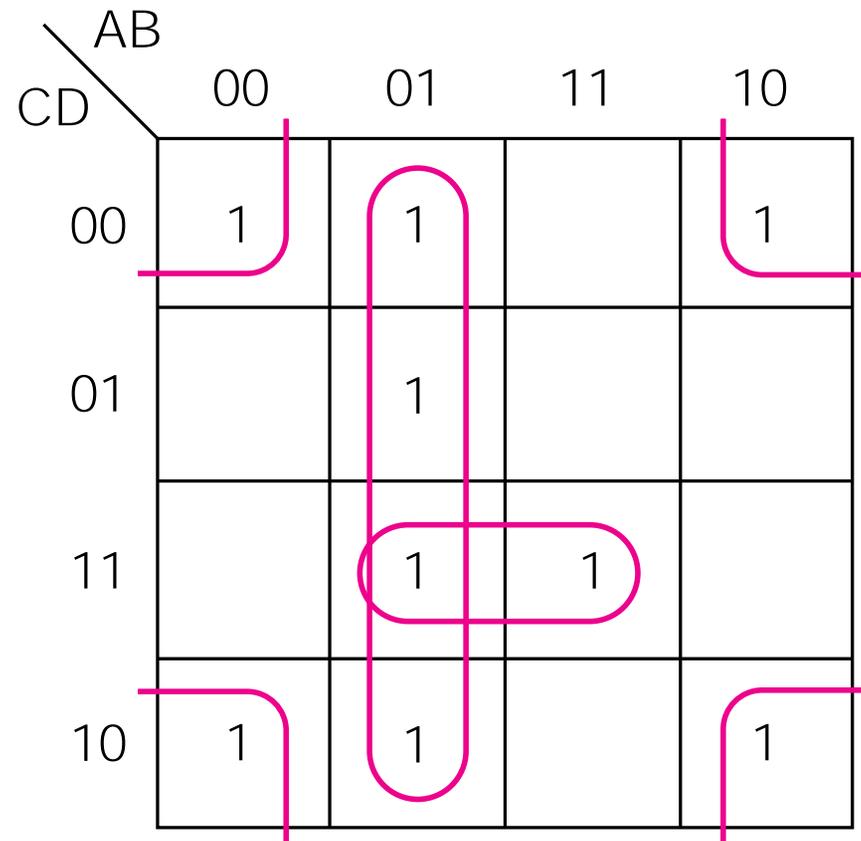


$$F = \bar{A}\bar{B}\bar{C} + \bar{A}CD + ABC + A\bar{C}\bar{D}$$



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

Fig A.45 The Corners of a K-Map Are Logically Adjacent



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

A.46 Two Different Minimized Equations Are Produced from the Same K-Map

		AB			
		00	01	11	10
CD	00	1			d
	01		1	1	
	11		1	1	
	10	d			

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

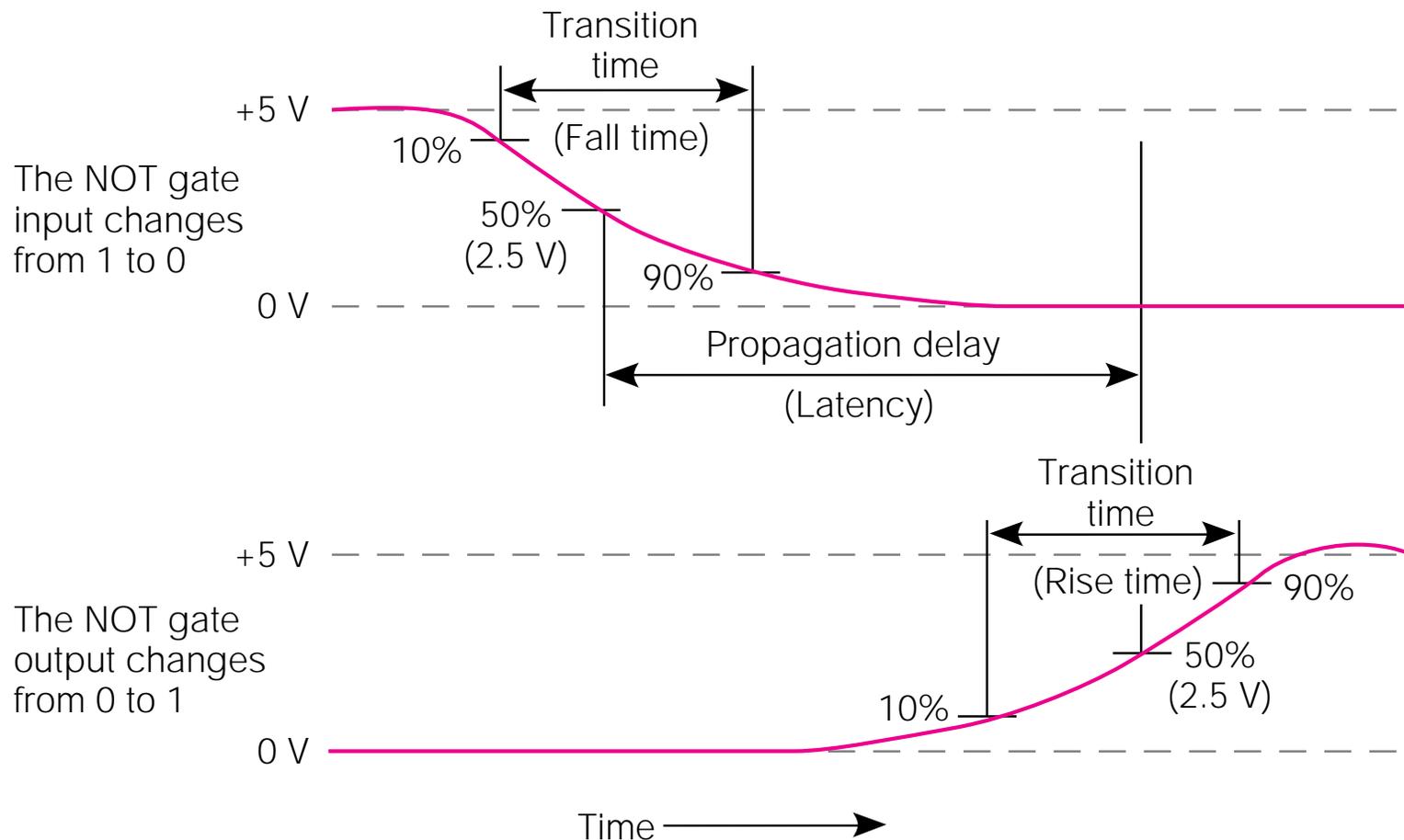
		AB			
		00	01	11	10
CD	00	1			d
	01		1	1	
	11		1	1	
	10	d			

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

Speed and Performance

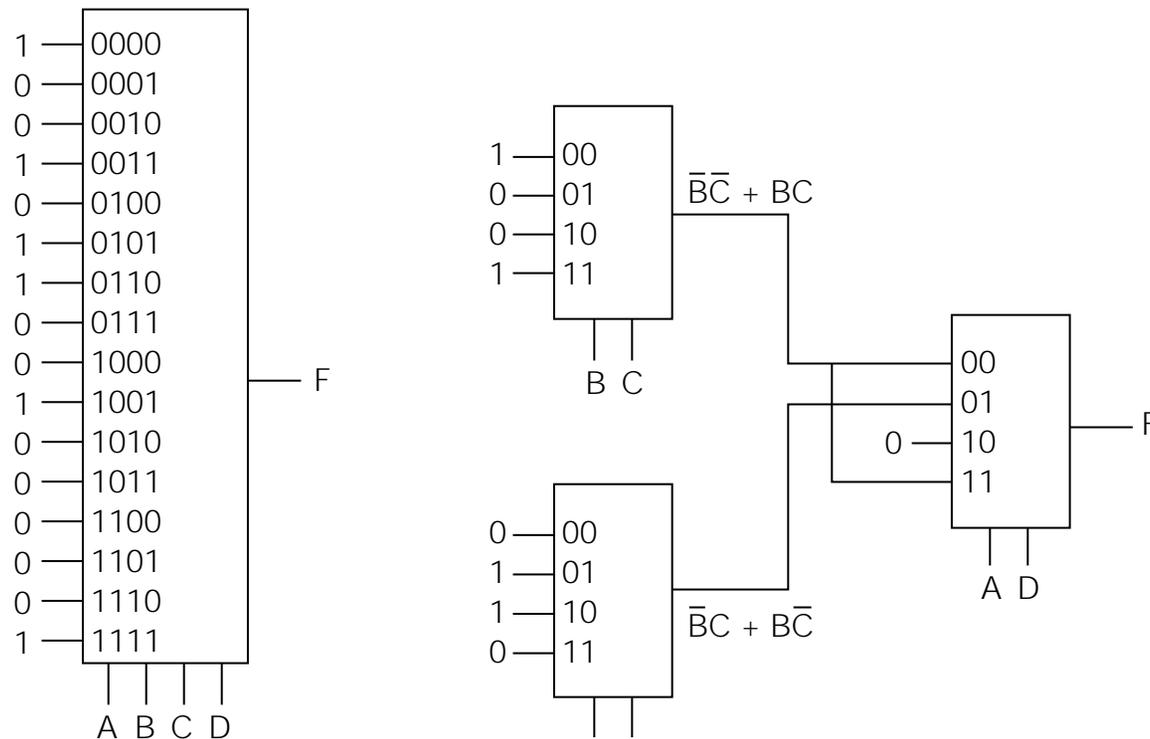
- **The speed of a digital system is governed by**
 - **the propagation delay through the logic gates and**
 - **the propagation across interconnections.**

Fig A.47 Propagation Delay for a NOT Gate (Adapted from Hamacher et al., 1990)



Circuit Depth Affects Propagation Delay

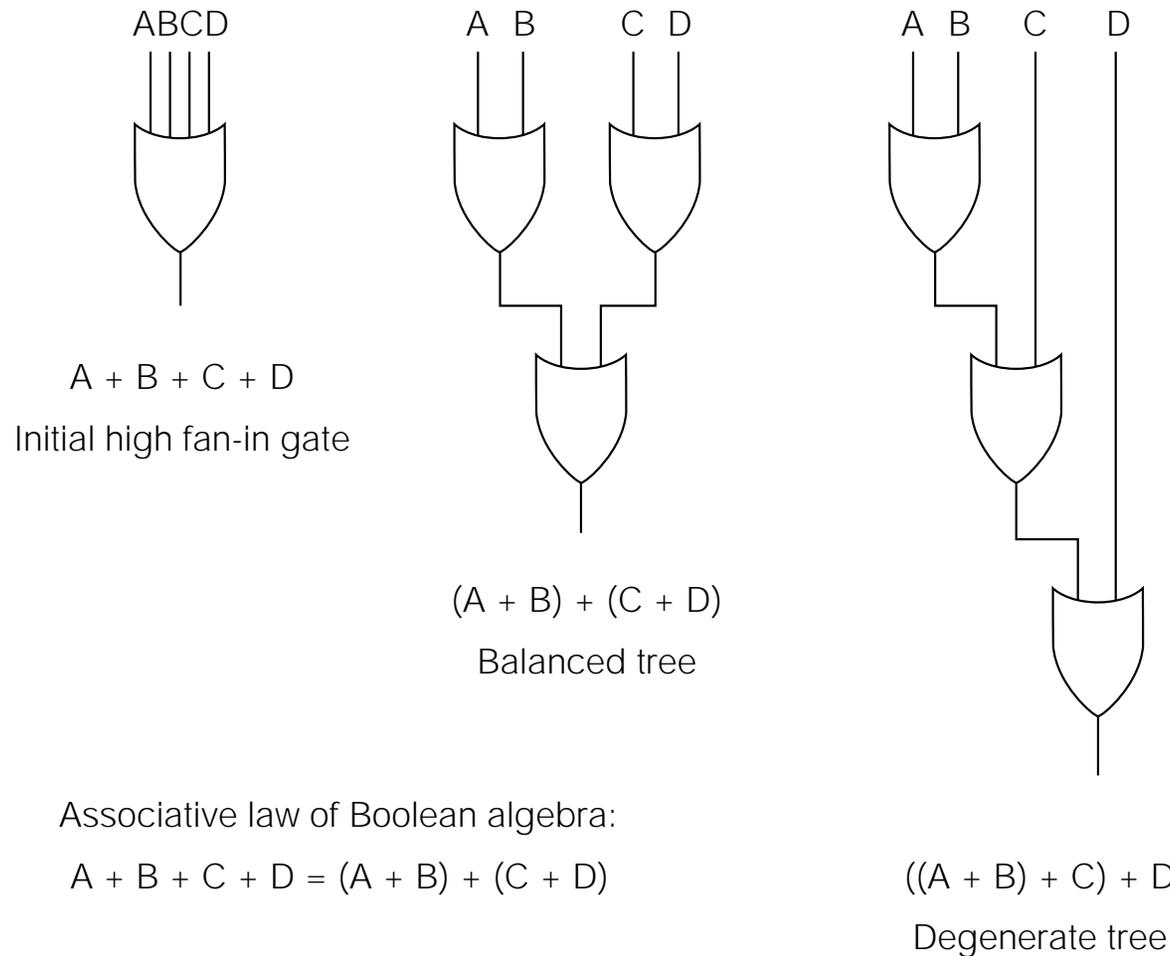
Fig A.48 A 4-Variable Function Implemented with a 16-to-1 MUX



$$\begin{aligned}
 F(ABCD) &= \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}\bar{D} + \bar{A}BC\bar{D} + A\bar{B}\bar{C}D + ABCD \\
 &= (\bar{B}\bar{C} + BC)AD + (\bar{B}C + B\bar{C})\bar{A}D + (\bar{B}\bar{C} + BC)
 \end{aligned}$$

Fan-in May Affect Circuit Depth

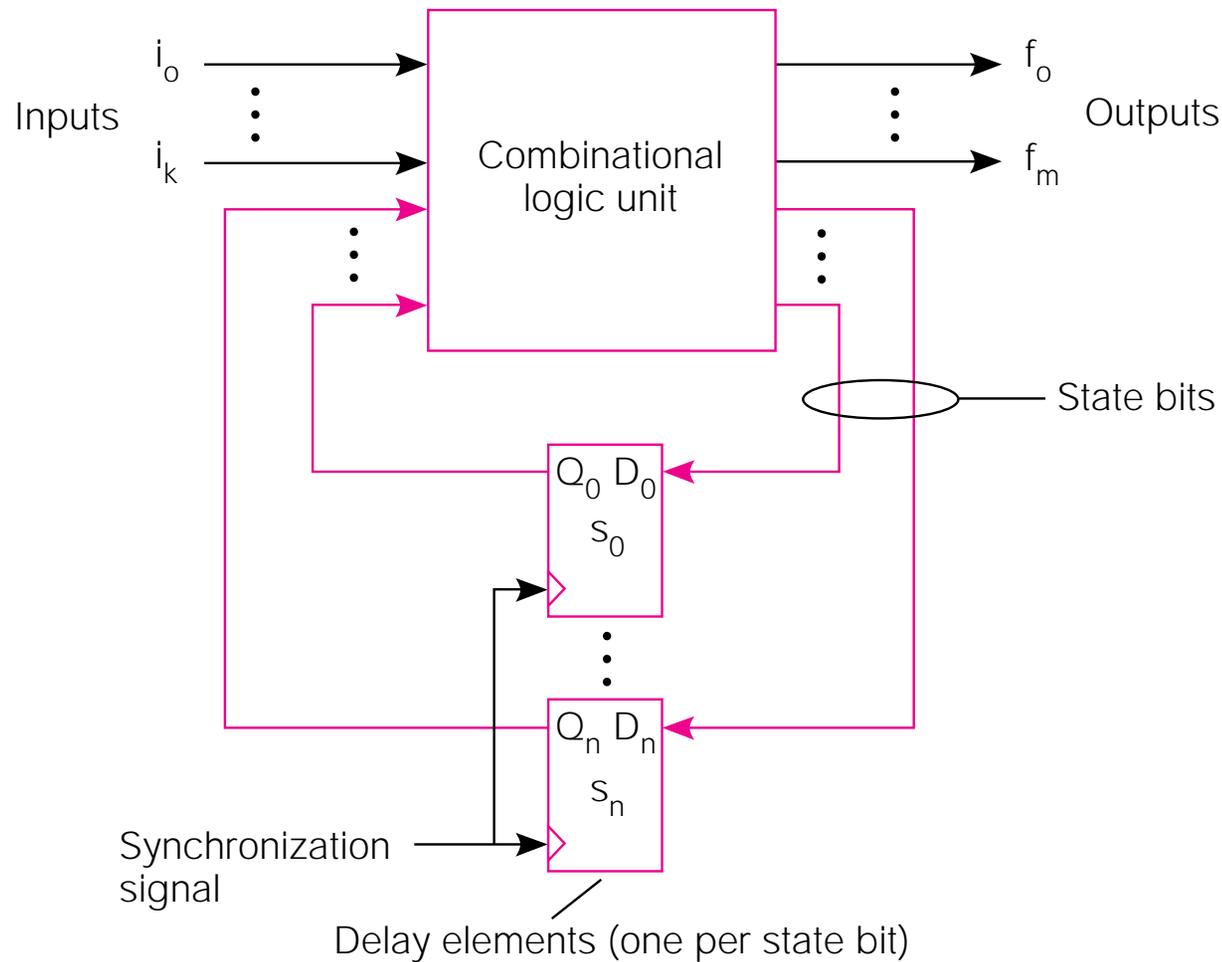
Fig A.49 A Logic Gate



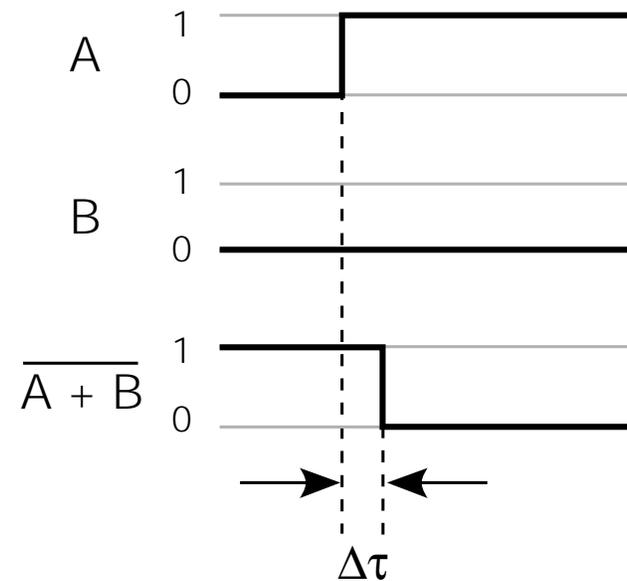
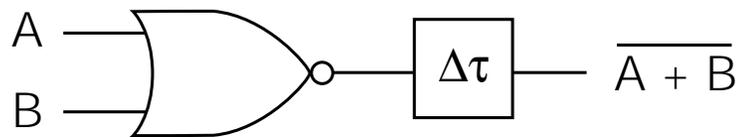
Sequential Logic

- The combinational logic circuits we have been studying so far have no memory. The outputs always follow the inputs.
- There is a need for circuits with a memory, which behave differently *depending upon their previous state*.
- An example is the vending machine, which must remember how many and what kinds of coins have been inserted, and which behave according to not only the current coin inserted, but also upon how many and what kind of coins have been deposited previously.
- These are referred to as *finite state machines*, because they can have at most a finite number of states.

Fig A.50 Classical Model of a Finite State Machine



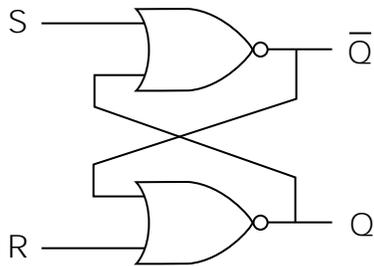
A.51 A NOR Gate with a Lumped Delay



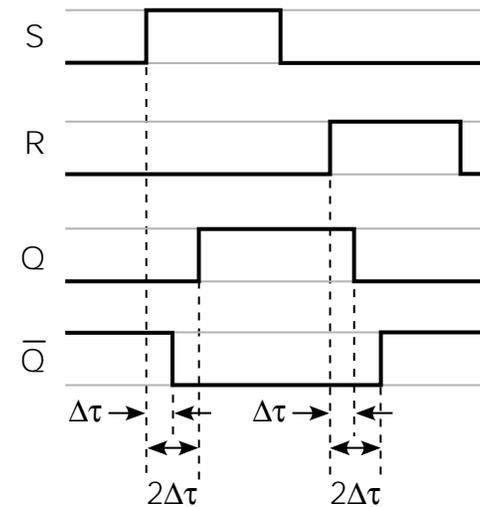
Timing behavior

This delay between input and output is at the basis of the functioning of an important memory element, the *flip-flop*.

A.52 An S-R Flip-Flop



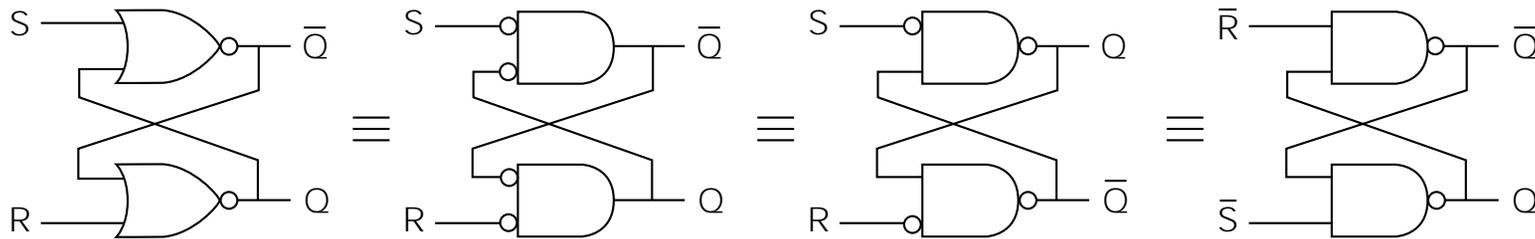
Q_t	S_t	R_t	Q_{i+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	(disallowed)
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	(disallowed)



Timing behavior

The S-R flip-flop is an active-high (positive logic) device.

Fig A.53 Converting a NOR S-R to an NAND S-R



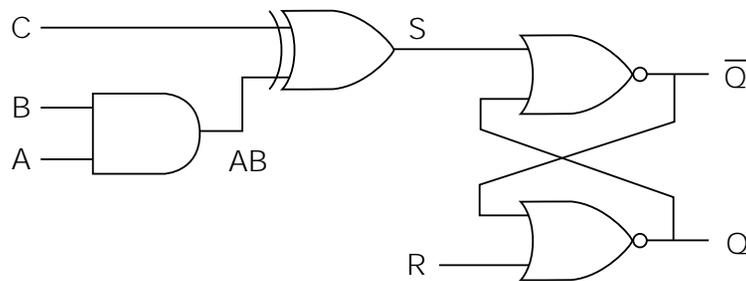
Active-high
NOR
Implementation

Push bubbles
(DeMorgan's)

Rearrange
bubbles

Convert
from bubbles
to active-low
signal names

Fig A.54 A Circuit with a Hazard



It is desirable to be able to “turn off” the flip-flop so it does not respond to such hazards.

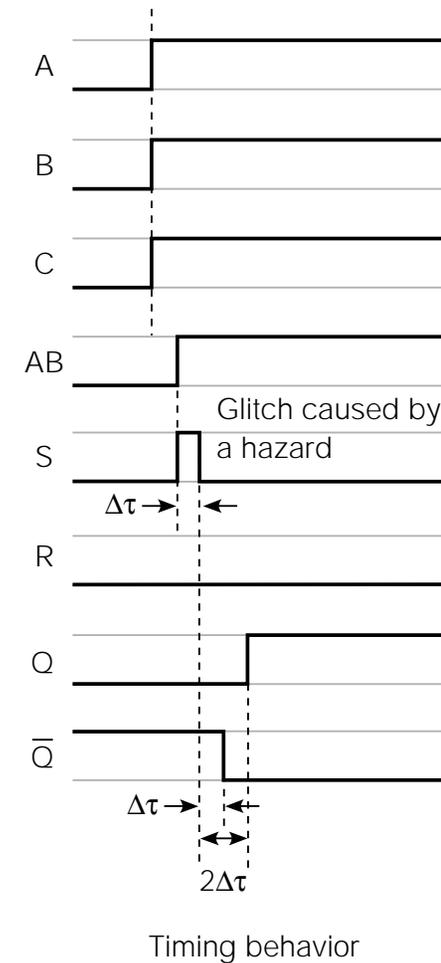
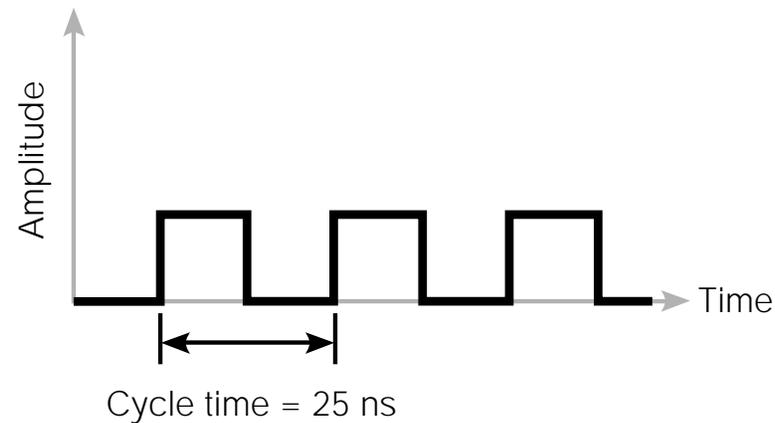
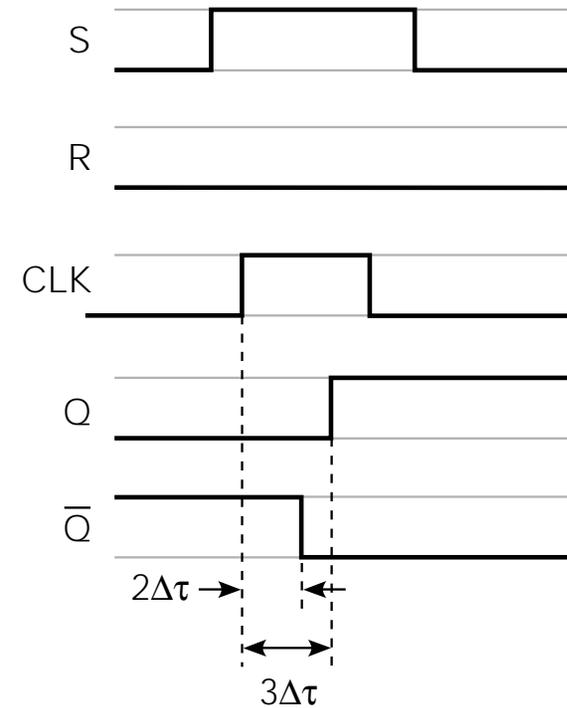
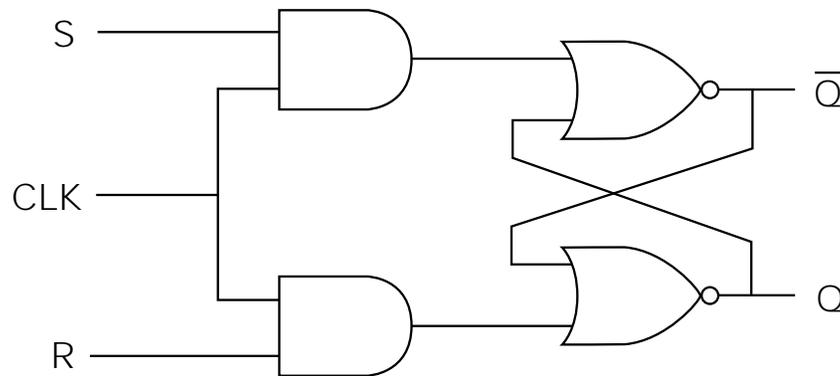


Fig A.55 A Clock Waveform



In a positive logic system, the “action” happens when the clock is high, or positive. The low part of the clock cycle allows propagation between subcircuits, so their inputs are stable at the correct value when the clock next goes high.

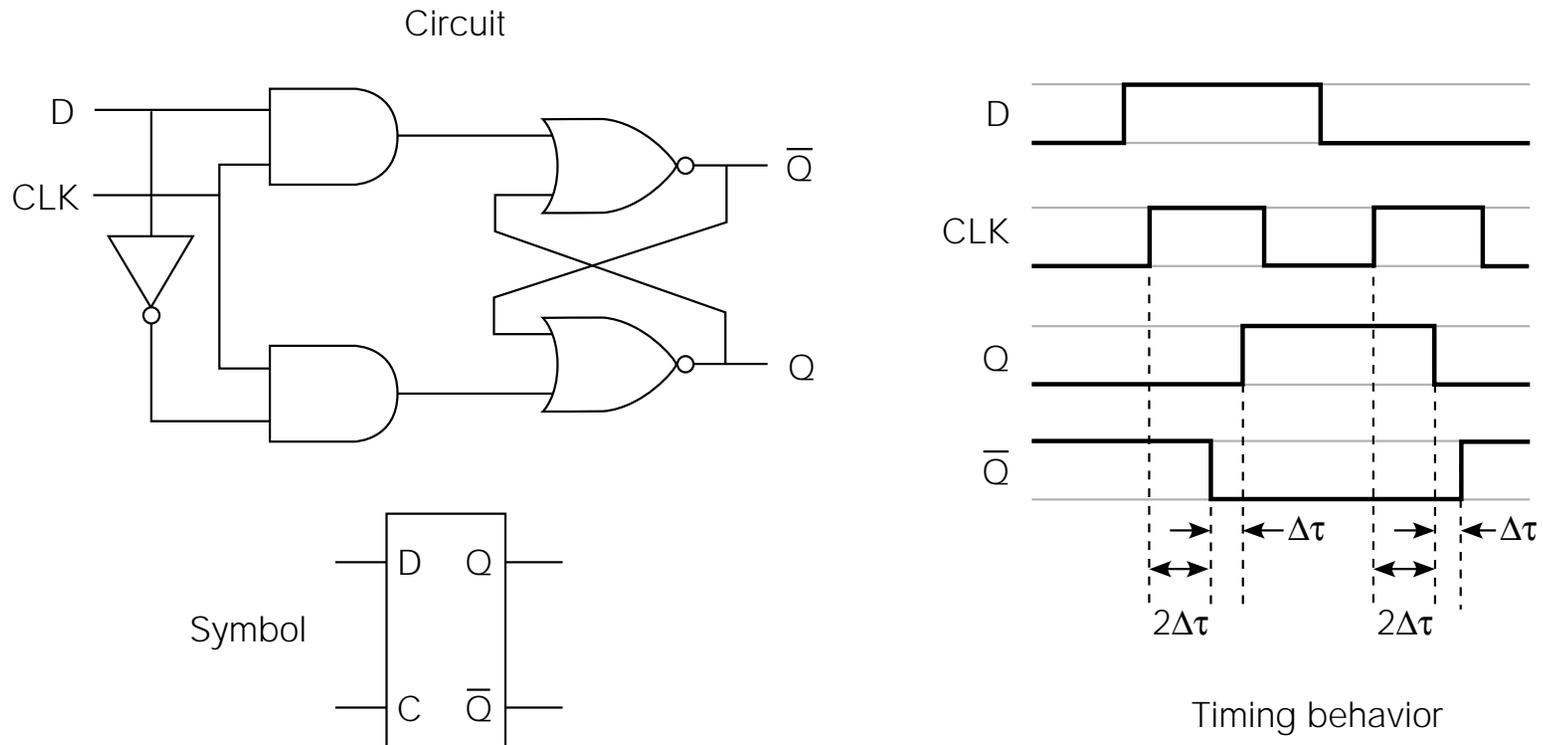
A.56 A Clocked S-R Flip-Flop



Timing behavior

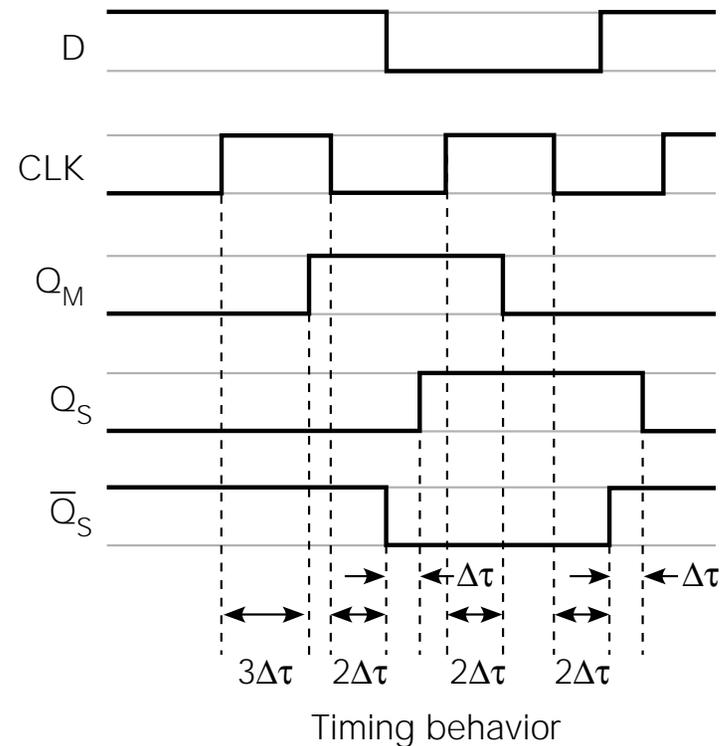
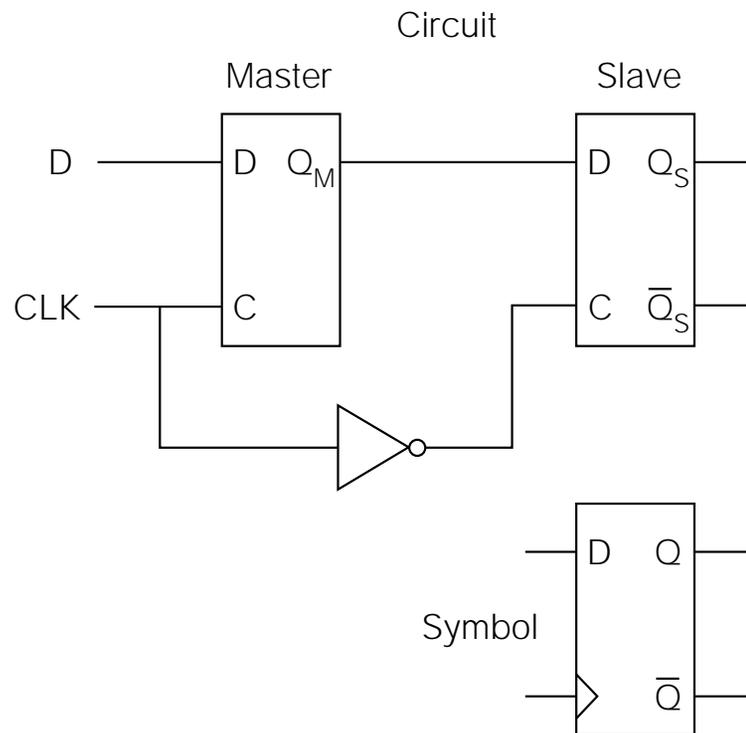
The clock signal, CLK, turns on the inputs to the flip-flop.

Fig A.57 A Clocked D (Data) Flip-Flop



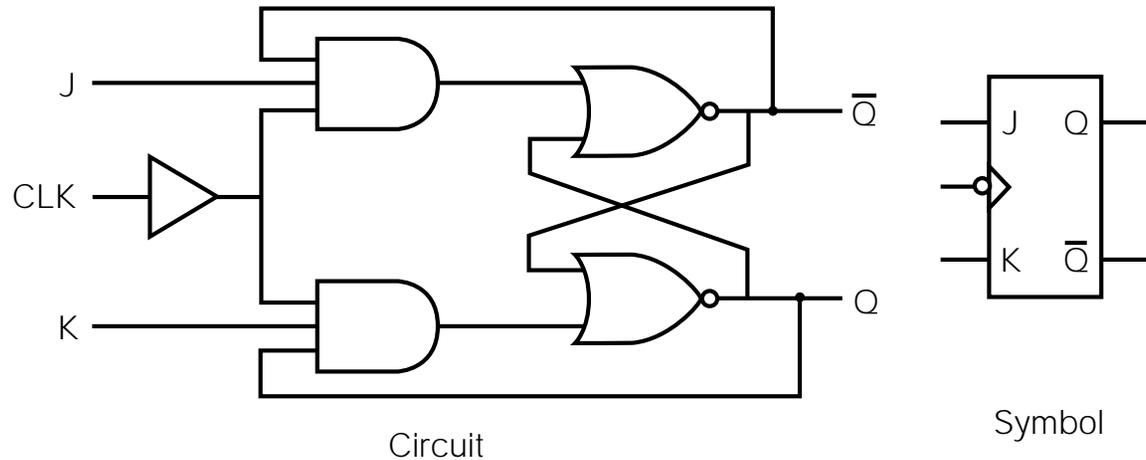
The clocked D flip-flop, sometimes called a latch, has a potential problem: If D changes while the clock is high, the output will also change. The Master-Slave flip-flop solves this problem.

A.58 A Master-Slave Flip-Flop



The rising edge of the clock clocks new data into the master, while the slave holds previous data. The falling edge clocks the new master data into the slave.

Fig A.59 The Basic J-K Flip-Flop



- The J-K flip-flop eliminates the $S = R = 1$ problem of the S-R flip-flop, because Q enables J while Q' disables K, and vice versa.
- However there is still a problem. If J goes momentarily to 1 and then back to 0 while the flip-flop is active and in the reset, the flip-flop will “catch” the 1.
- This is referred to as “1’s catching.”
- The J-K master-slave flip-flop solves this problem.

Fig A.61 A Master-Slave J-K Flip-Flop

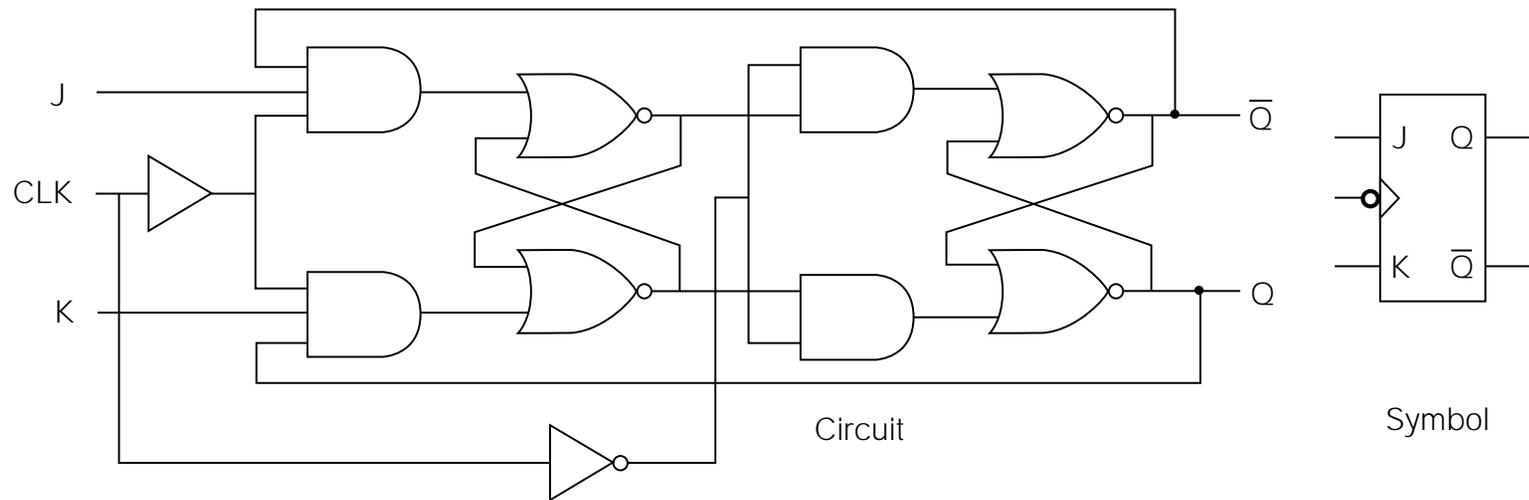
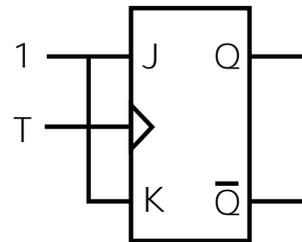
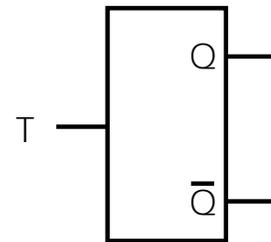


Fig A.60 A T Flip-Flop



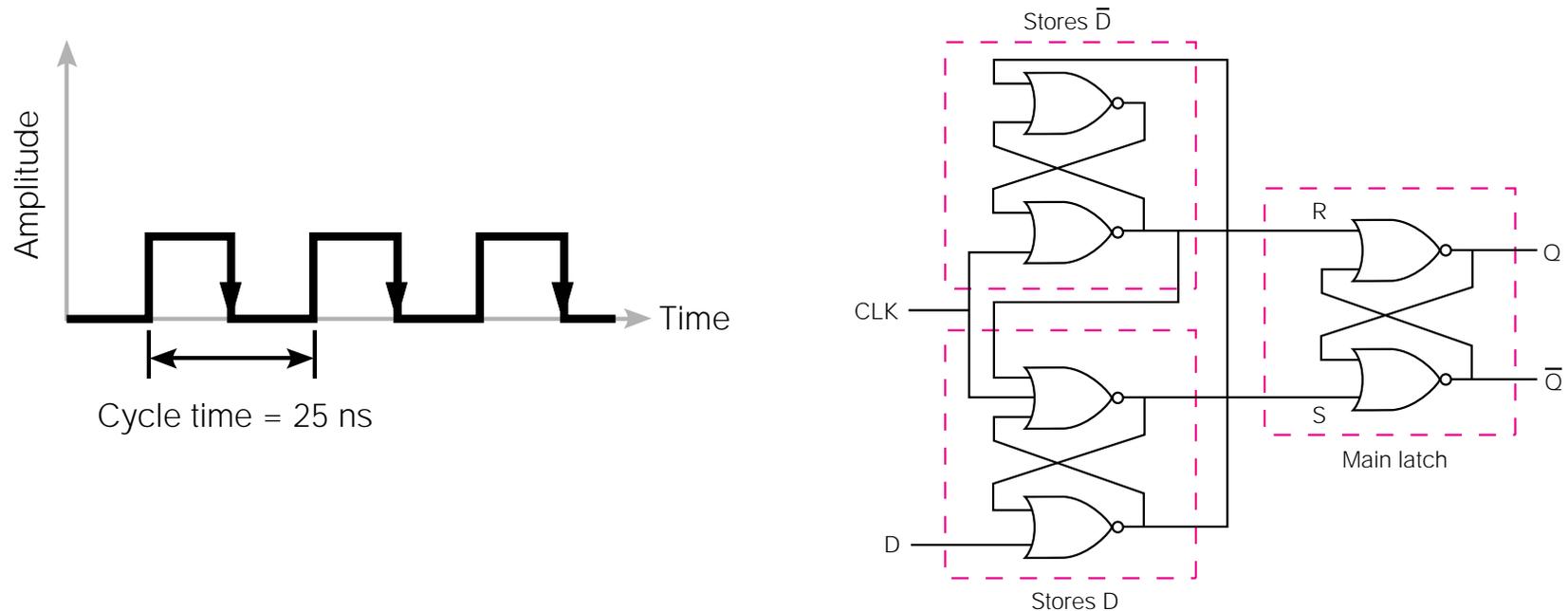
Circuit



Symbol

- The presence of a constant 1 at J and K means that the flip-flop will change its state from 0-1 or 1-0 each time it is clocked by the T (toggle) input.

Fig A.62 Negative Edge-Triggered D Flip-Flop



- When the clock is high, the two input latches output 0, so the main latch remains in its previous state regardless of changes in D.
- When the clock goes high-low, values in the two input latches will affect the state of the main latch.
- While the clock is low, D cannot affect the main latch.

Finite State Machine Design

- Counter has a clock input, CLK, and a RESET input.
- Has two output lines, which must take values of 00, 01, 10, and 11 on subsequent clock cycles.

Fig A.63 A Modulo-4 Counter

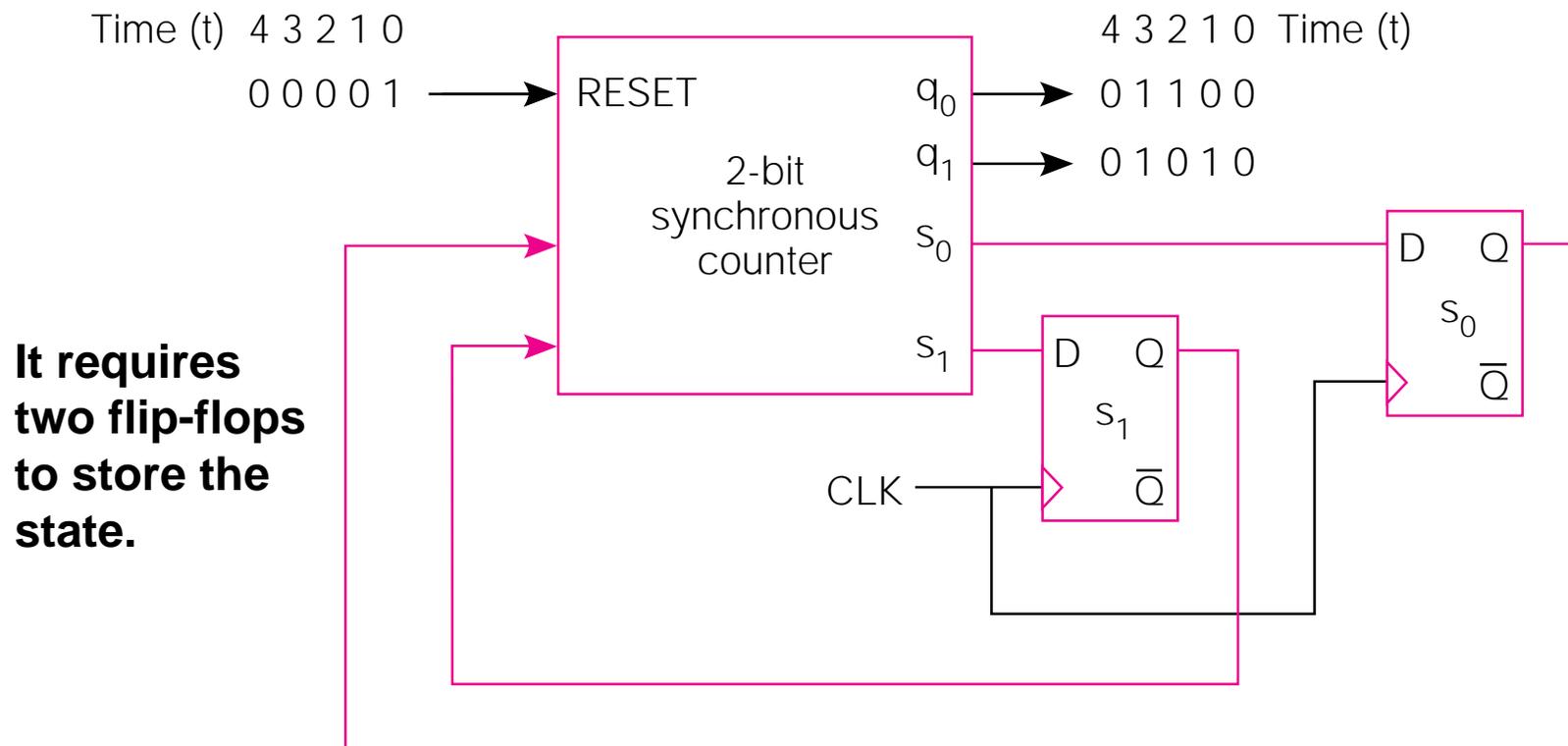
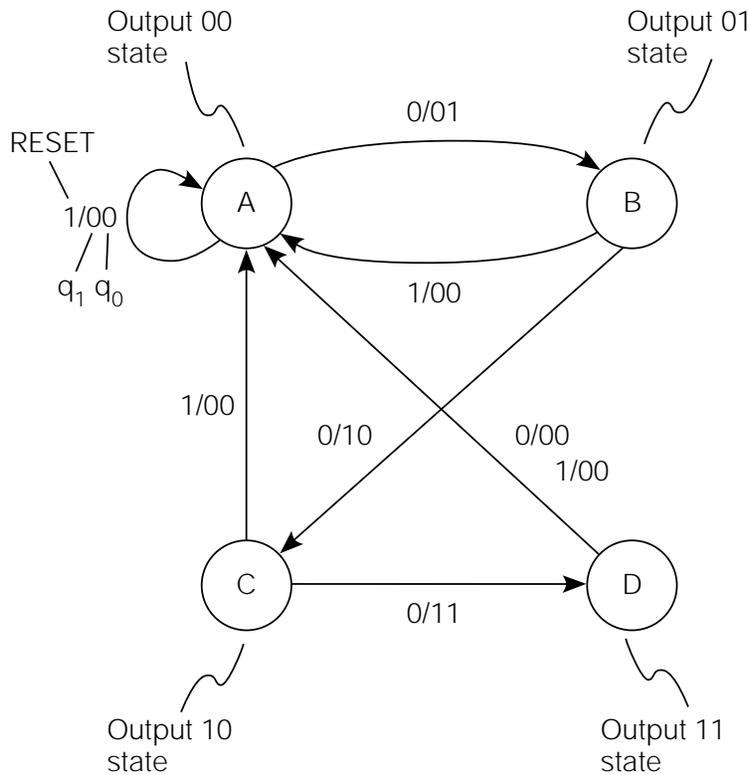


Fig A.64 State Transition Diagram for a Modulo(4) Counter



State Table	Present State	Next State	
		RESET 0	RESET 1
	A	B/01	A/00
	B	C/10	A/00
	C	D/11	A/00
	D	A/00	A/00

State Table With States Assigned	Present State	RESET	
		0	1
	A:00	01	00
	B:01	10	00
	C:10	11	00
	D:11	00	00

- The state diagram and state table tell “all there is to know” about the FSM, and are the basis for a provably correct design.

Fig A.67 Truth Table

$r(t)$	$s_1(t)s_0(t)$	$s_1s_0(t+1)$	$q_1q_0(t+1)$
0	00	01	01
0	01	10	10
0	10	11	11
0	11	00	00
1	00	00	00
1	01	00	00
1	10	00	00
1	11	00	00

- Develop equations from this truth table for $s_0(t+1)$, $s_1(t+1)$, $q_0(t+1)$, and $q_1(t+1)$ from inputs $r(t)$, $s_0(t)$ and $s_1(t)$

Fig A.67 Equations

$$s_0(t+1) = \overline{r(t)s_1(t)s_0(t)} + \overline{r(t)s_1(t)s_0(t)}$$

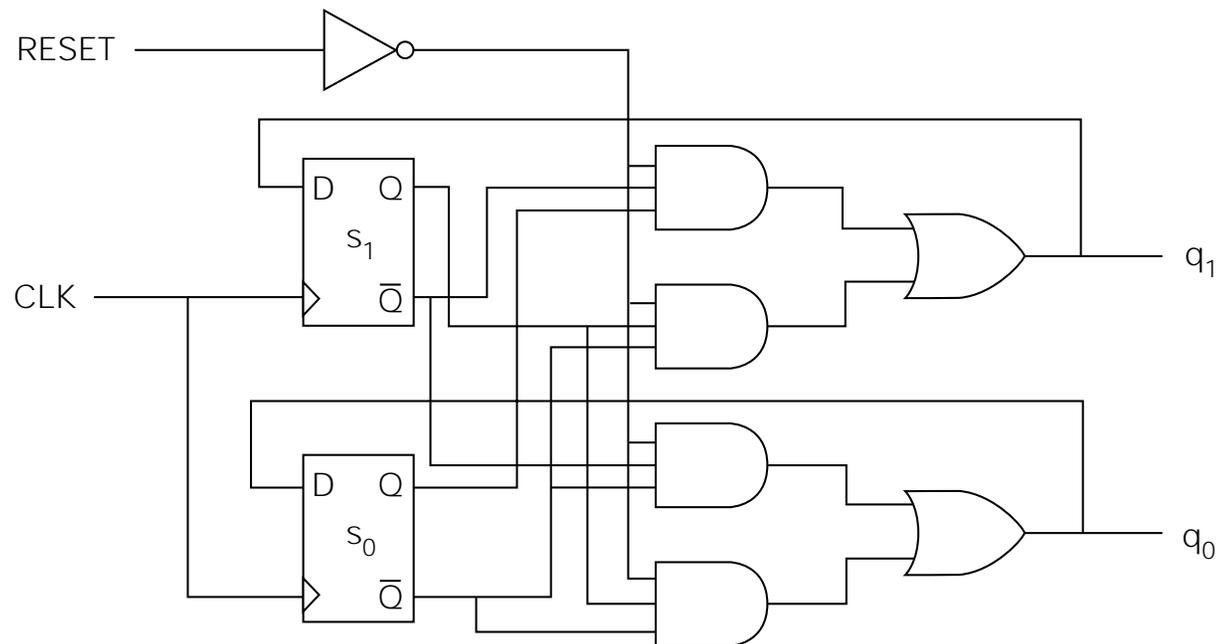
$$s_1(t+1) = \overline{r(t)s_1(t)s_0(t)} + \overline{r(t)s_1(t)s_0(t)}$$

$$q_0(t+1) = \overline{r(t)s_1(t)s_0(t)} + \overline{r(t)s_1(t)s_0(t)}$$

$$q_1(t+1) = \overline{r(t)s_1(t)s_0(t)} + \overline{r(t)s_1(t)s_0(t)}$$

Implement these equations

Fig A.68 Logic Design for a Modulo(4) Counter



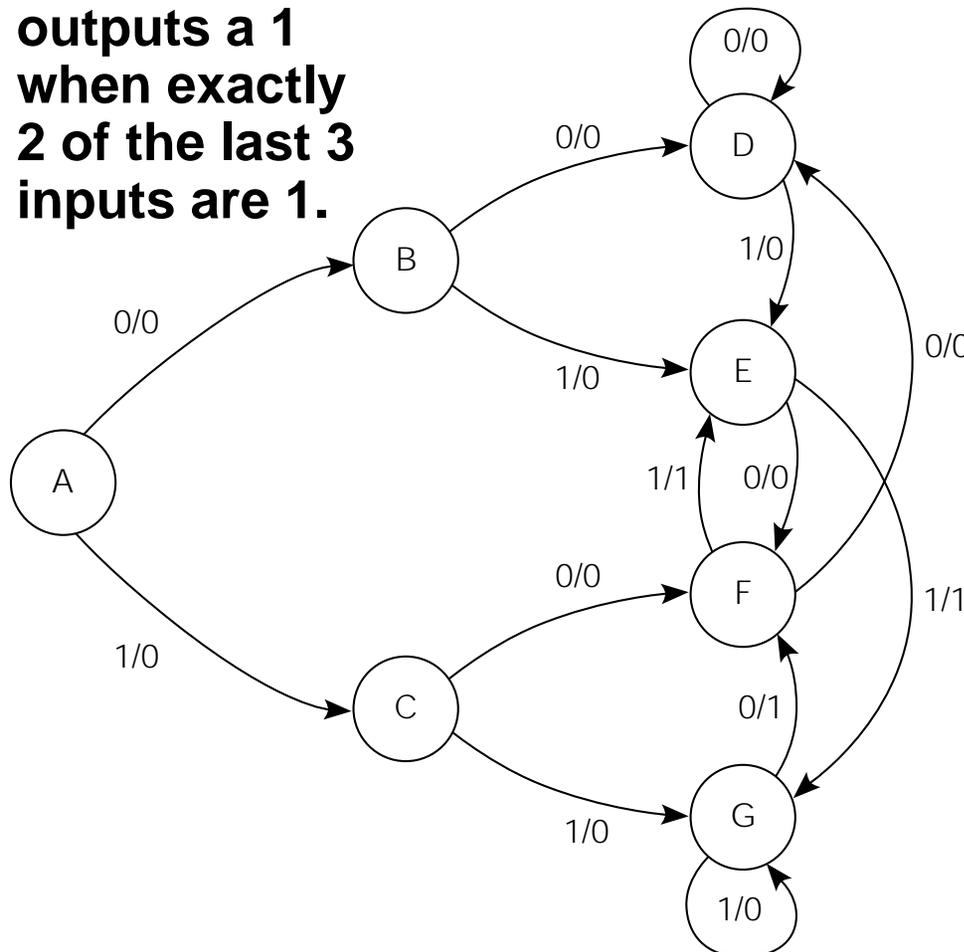
There are many simpler techniques for implementing counters.

Example A.2: A Sequence Detector

- Design a machine that outputs a 1 when exactly 2 of the last 3 inputs are 1.
- e.g. input sequence of **011011100** produces an output sequence of **001111010**
- Assume input is a 1-bit serial line.
- Use D flip-flops and 8-1 multiplexers.
- Begin by constructing a state transition diagram.

Fig A.69 State Transition Diagram for Sequence Detector

- Design a machine that outputs a 1 when exactly 2 of the last 3 inputs are 1.

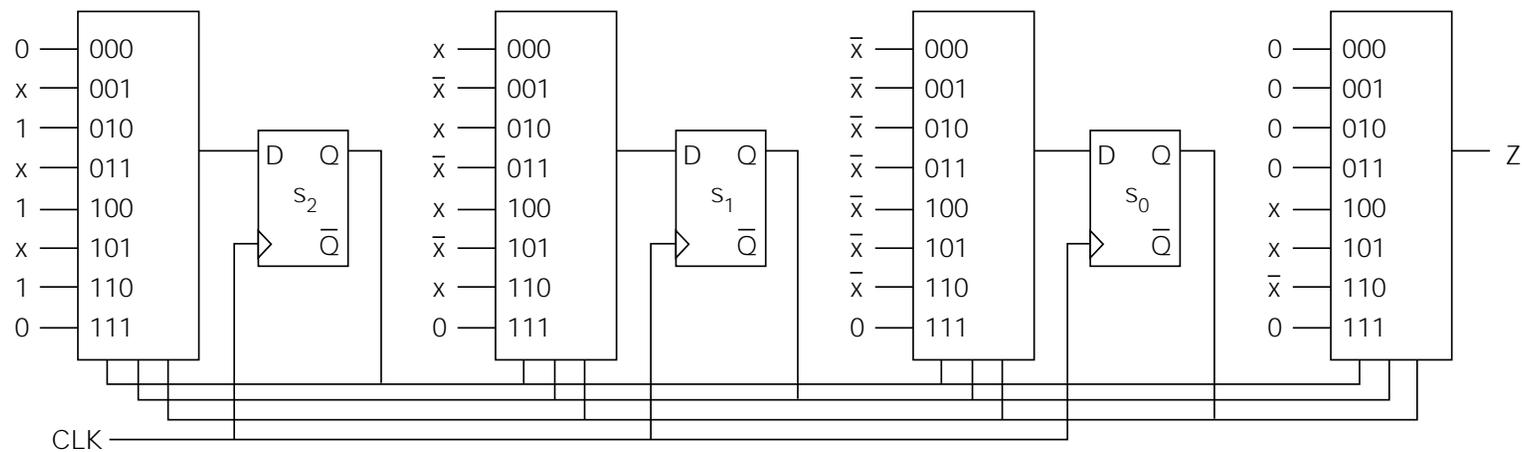


- Discuss: the “meaning” of each state.

Pres. State	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	101/0	110/0
D=011	011/0	100/0
E=100	101/0	110/1
F=101	011/0	100/1
G=110	101/1	110/0

- Convert table to truth table (how?).
- Solve for $s_2 s_1 s_0$ and Z.

Fig A.72 Logic Diagram for Sequence Detector



Example A.3: A Vending Machine Controller

- **Accepts nickel, dime, and quarter. When value of money inserted equals or exceeds twenty cents, machine vends item and returns change if any, and waits for next transaction.**
- **Implement with PLA and D flip-flops.**

Fig A.73 State Transition Diagram for Vending Machine Controller

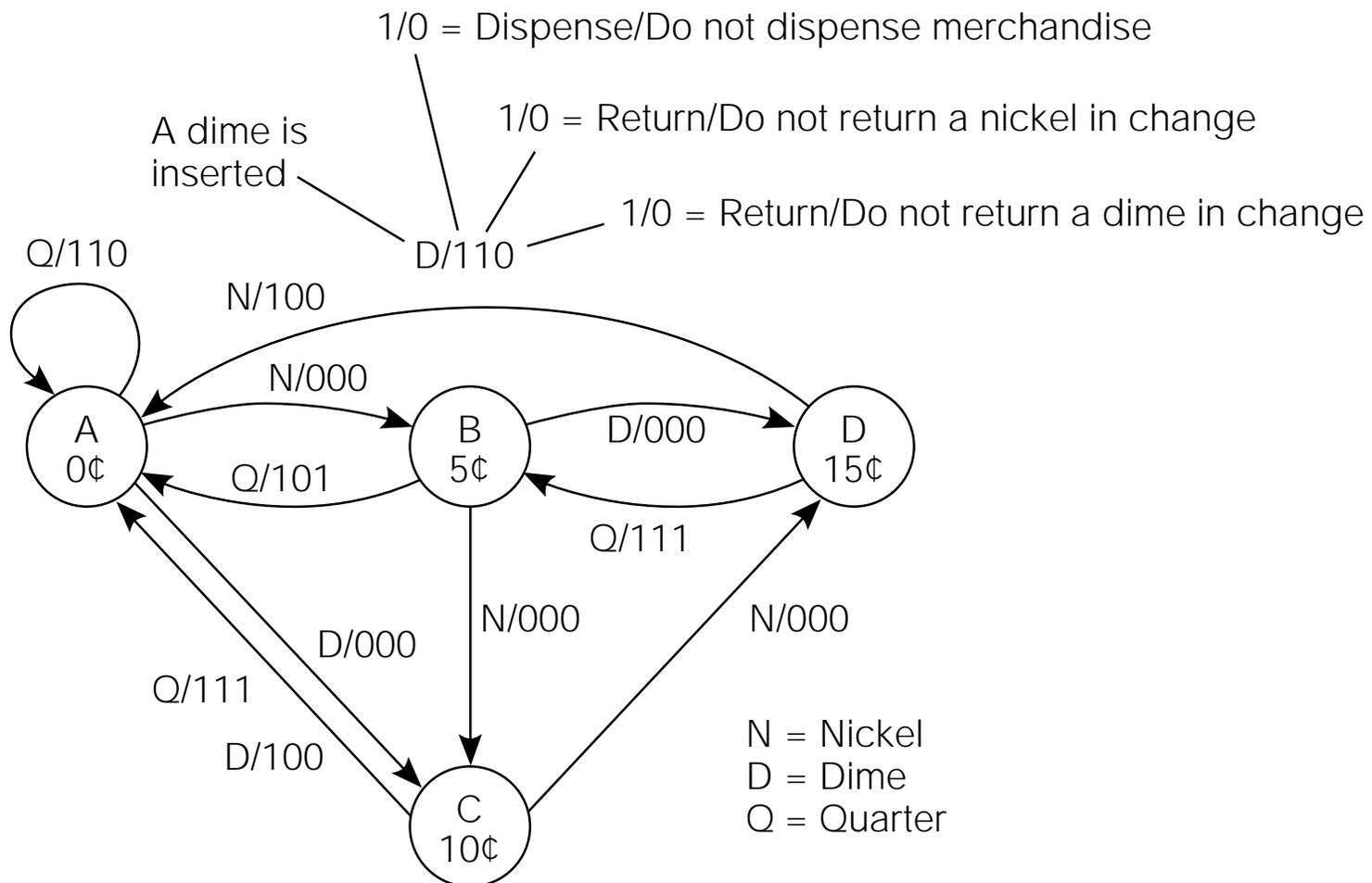
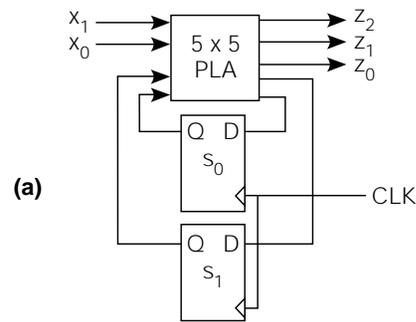


Fig A.75(b) Truth Table for Vending Machine Controller

Base 10 equivalent	Present state		Coin		Next state	Dispense			
	S_1	S_0	X_1	X_0		S_1	S_0	Z_2	Z_1
0	0	0	0	0	0	1	0	0	0
1	0	0	0	1	1	0	0	0	0
2	0	0	1	0	0	0	1	1	0
3	0	0	1	1	d	d	d	d	d
4	0	1	0	0	1	0	0	0	0
5	0	1	0	1	1	1	0	0	0
6	0	1	1	0	0	0	1	0	1
7	0	1	1	1	d	d	d	d	d
8	1	0	0	0	1	1	0	0	0
9	1	0	0	1	0	0	1	0	0
10	1	0	1	0	0	0	1	1	1
11	1	0	1	1	d	d	d	d	d
12	1	1	0	0	0	0	1	0	0
13	1	1	0	1	0	0	1	1	0
14	1	1	1	0	0	1	1	1	1
15	1	1	1	1	d	d	d	d	d

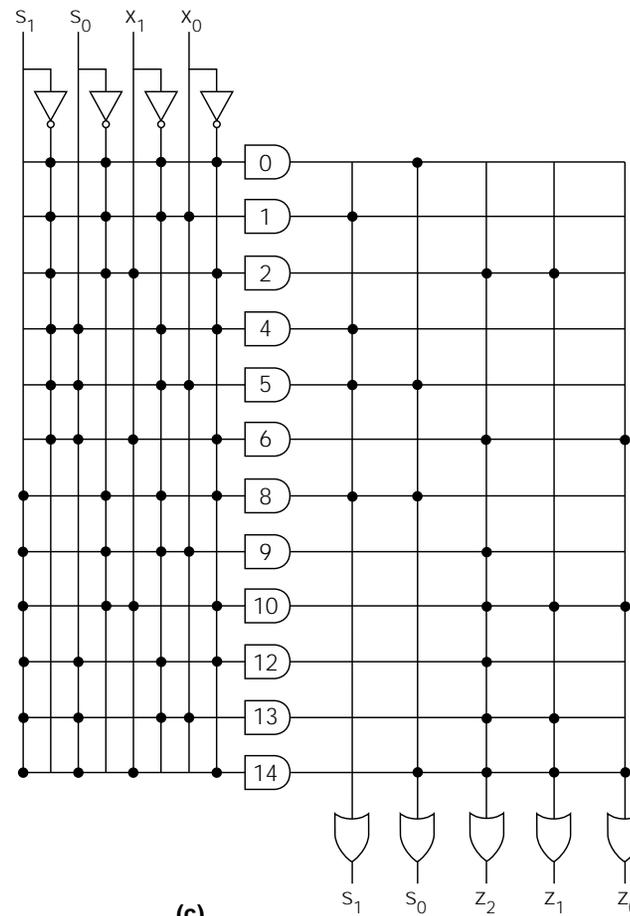
(b)

Fig A.75 (a)FSM Circuit, (b)Truth Table, and (c)PLA Realization for Vending Machine Controller



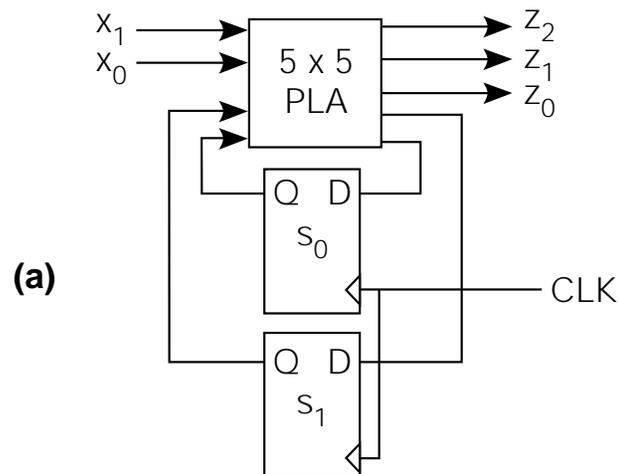
Base 10 equivalent	Present state				Coin	Next state				Dispense		
	s_1	s_0	x_1	x_0		s_1	s_0	z_2	z_1	z_0	Return nickel	Return dime
0	0	0	0	0	0	1	0	0	0			
1	0	0	0	1	1	0	0	0	0			
2	0	0	1	0	0	0	1	1	0			
3	0	0	1	1	d	d	d	d	d			
4	0	1	0	0	1	0	0	0	0			
5	0	1	0	1	1	1	0	0	0			
6	0	1	1	0	0	0	1	0	1			
7	0	1	1	1	d	d	d	d	d			
8	1	0	0	0	1	1	0	0	0			
9	1	0	0	1	0	0	1	0	0			
10	1	0	1	0	0	0	1	1	1			
11	1	0	1	1	d	d	d	d	d			
12	1	1	0	0	0	0	1	0	0			
13	1	1	0	1	0	0	1	1	0			
14	1	1	1	0	0	1	1	1	1			
15	1	1	1	1	d	d	d	d	d			

(b)

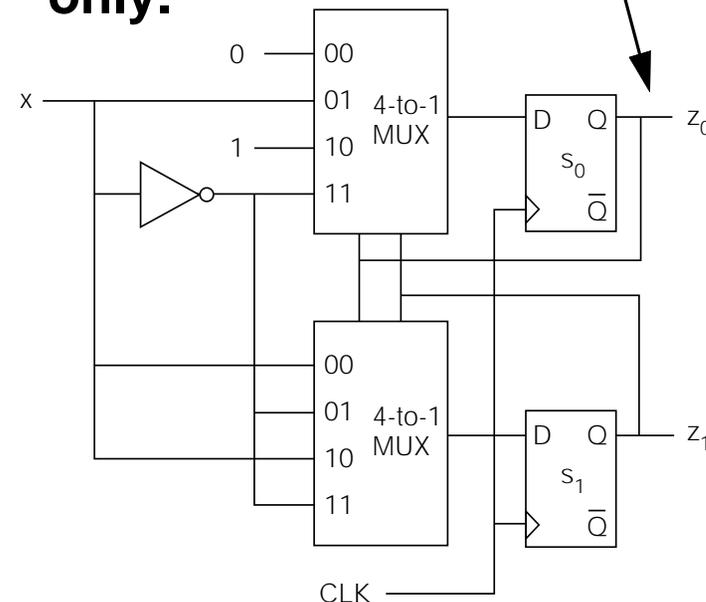


Mealy versus Moore Machines

- **Mealy model: Outputs are functions of inputs and present state.**
- **Previous FSM designs were Mealy machines, because next state was computed from present state and inputs.**



- **Moore model: Outputs are functions of present state only.**

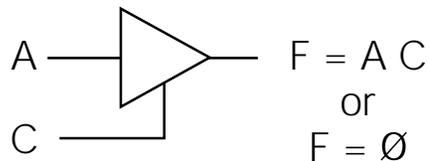


- **Both are equally powerful.**

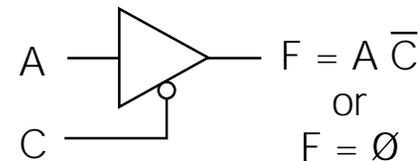
Fig A.77 Tri-State Buffers

C	A	F
0	0	∅
0	1	∅
1	0	0
1	1	1

C	A	F
0	0	0
0	1	1
1	0	∅
1	1	∅



Tri-state buffer



Tri-state buffer, inverted control

- There is a third state: high impedance. This means the gate output is essentially disconnected from the circuit.
- This state is indicated by \emptyset in the figure.

Fig A.78 A 4-Bit Register

Gate-Level View

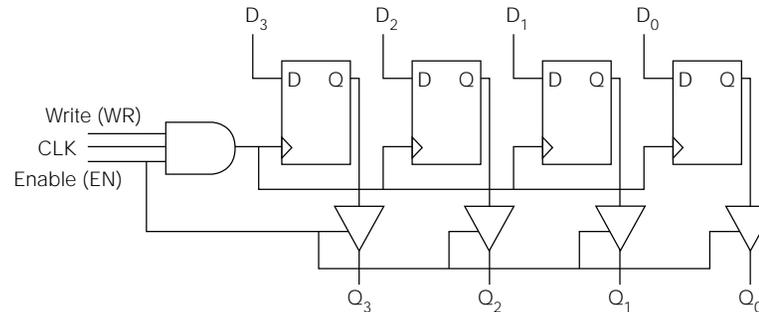


Fig A.79 Abstract Representation of a 4-Bit Register

Chip-Level View

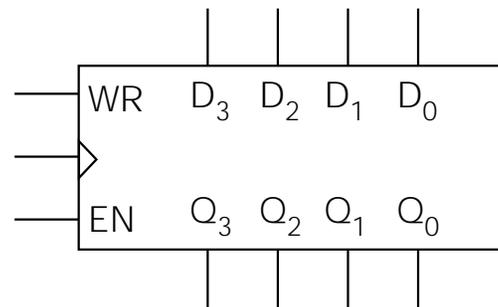
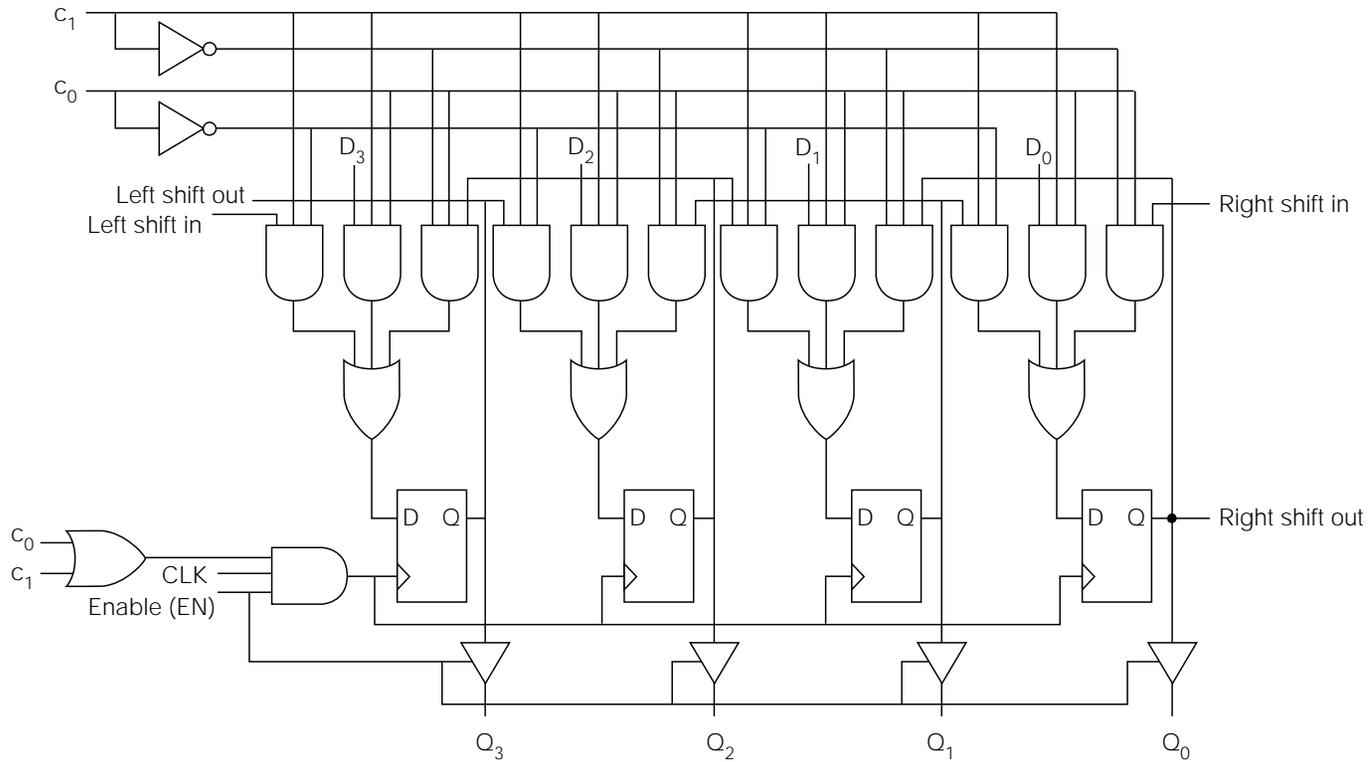


Fig A.80 Internal Layout and Block Diagram for Left-Right Shift with Parallel Read/Write Capabilities



Control	Function
$c_1 \ c_0$	
0 0	No change
0 1	Shift left
1 0	Shift right
1 1	Parallel load

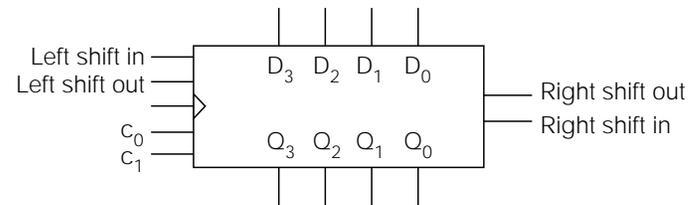
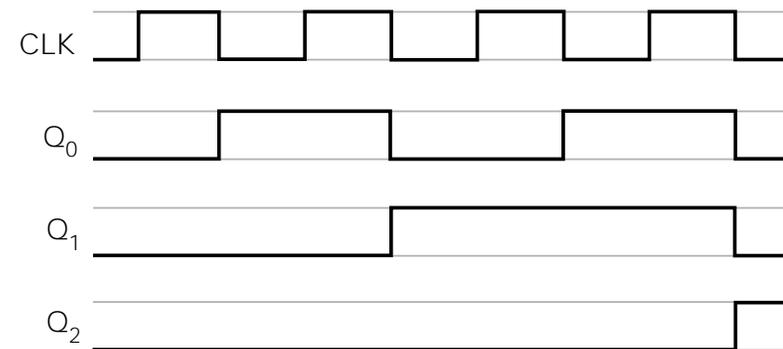
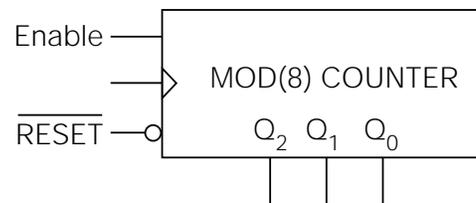
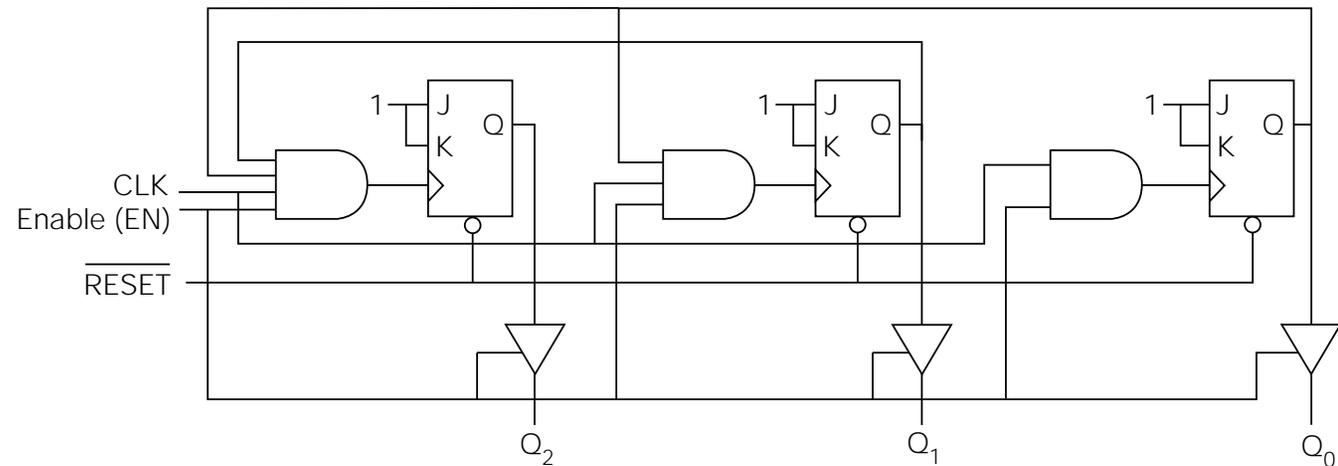


Fig A.81 A Modulo(8) Ripple Counter

Note the use of the T flip-flops. They are used to toggle the input of the next flip-flop when its output is 1.



Timing behavior