Digital Logic

COSC 2329
Sam Houston State University

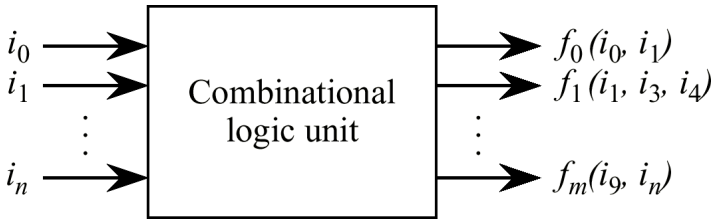
Slides based on a set supplied by Miles Murdocca

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

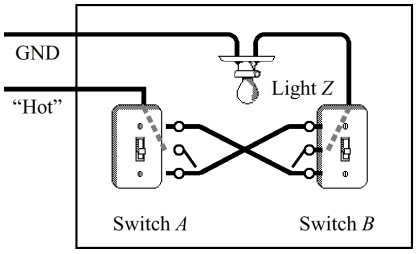
- Translates a set of inputs into a set of outputs according to one or more mapping functions.
- Inputs and outputs for a CLU normally have two distinct (binary) values: high and low, 1 and 0, 0 and 1, or 5 v. and 0 v. for example.
- The outputs of a CLU are strictly functions of the inputs, and the outputs are updated immediately after the inputs change. A set of inputs i0 – in are presented to the CLU, which produces a set of outputs according to mapping functions f0 – fm



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?



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

Truth Tables Showing All Possible Functions of Two Binary Variables

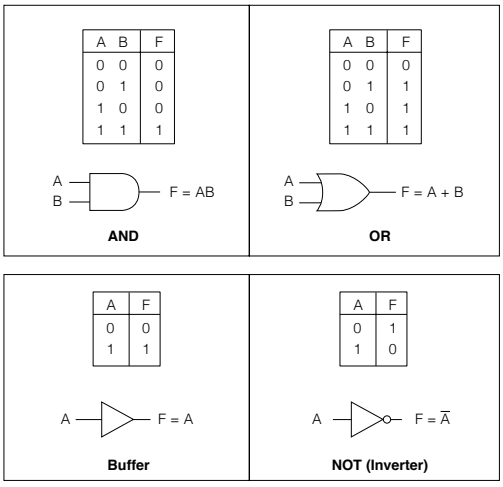
A	B	*False*	*AND*	$A\overline{B}$	A	$\overline{A}B$	B	*XOR*	*OR*
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

A	B	*NOR*	*XNOR*	$\overline{B}$	$A + \overline{B}$	$\overline{A}$	$\overline{A} + B$	*NAND*	*True*
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, XOR, and NAND. (Always use upper case spelling.)

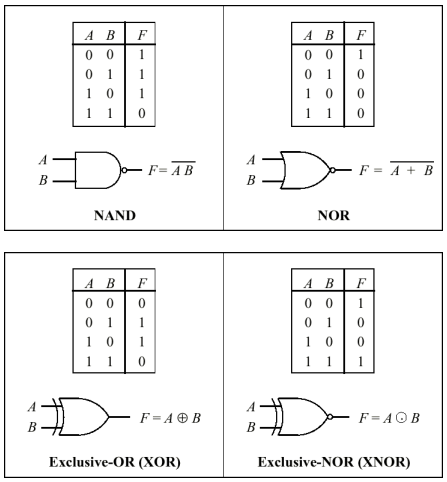
Logic Gates and Their Symbols

Logic symbols for AND, OR, buffer, and NOT Boolean functions

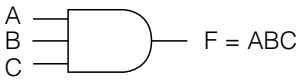


- Note the use of the "inversion bubble."
- (Be careful about the "nose" of the gate when drawing AND vs. OR.)

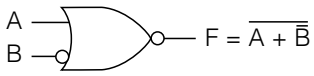
Logic symbols for NAND, NOR, XOR, and XNOR Boolean functions



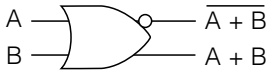
Variations of Basic Logic Gate Symbols



(a)



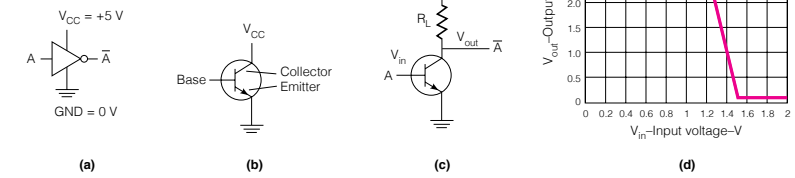
(b)



(c)

a) 3 inputs **b) A Negated Input** **c) Complementary Outputs**

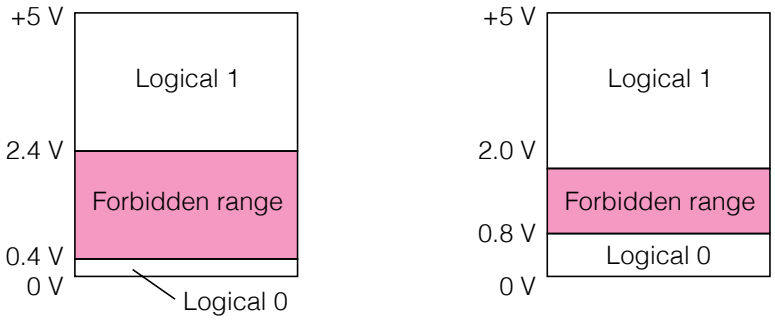
The Inverter at the Transistor Level



(a) (b) (c) (d)

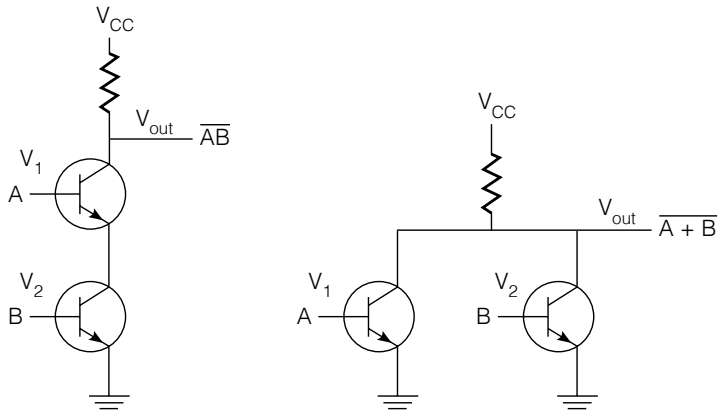
Power Terminals | **Transistor Symbol** | **A Transistor Used as an Inverter** | **Inverter Transfer Function**

Allowable Voltages in Transistor-Transistor-Logic (TTL)



(a) **(b)**

Transistor-Level Circuits For 2-Input a) NAND and b)NOR Gates

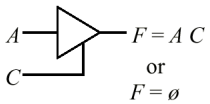


(a) **(b)**

Tri-State Buffers

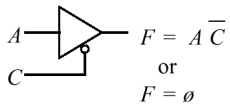
- Outputs can be 0, 1, or "electrically disconnected."

C	A	F
0	0	ø
0	1	ø
1	0	0
1	1	1



Tri-state buffer

C	A	F
0	0	0
0	1	1
1	0	ø
1	1	ø



Tri-state buffer, inverted control

The Basic Properties of Boolean Algebra

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
$A\,\overline{A} = 0$	$A + \overline{A} = 1$	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
$\overline{\overline{A}} = A$		Complement
$\overline{A\,B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\,\overline{B}$	DeMorgan's Theorem
$AB + \overline{A}C + BC$ $= AB + \overline{A}C$	$(A + B)(\overline{A} + C)(B + C)$ $= (A + B)(\overline{A} + C)$	Consensus Theorem

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

Postulates

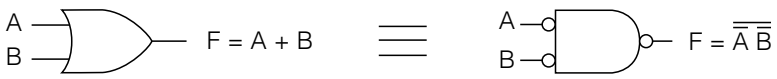
Theorems

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

DeMorgan's Theorem

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

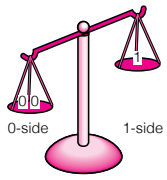
DeMorgan's theorem: $A + B = \overline{\overline{A + B}} = \overline{\overline{A}}\,\overline{\overline{B}}$



The Sum-of-Products (SOP) Form

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

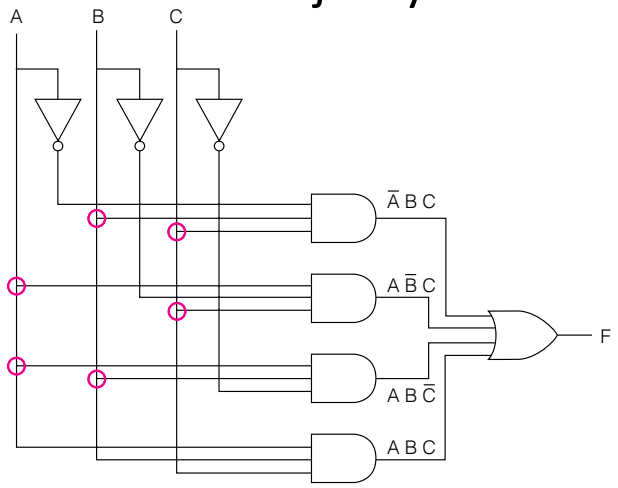
- The SOP form for the 3-input majority gate is:

$$M = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC = m_3 + m_5 + m_6 + m_7 = \Sigma(3, 5, 6, 7)$$

- Each of the 2^n terms are called minterms, running from 0 to $2^n - 1$

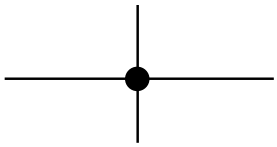
- Note the relationship between minterm number and boolean value.
- Discuss: common-sense interpretation of equation.

A 2-Level AND-OR Circuit that Implements the Majority Function

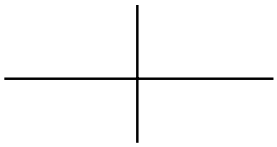


Discuss: What is the Gate Count?

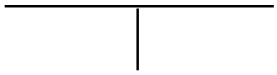
Notation Used at Circuit Intersections



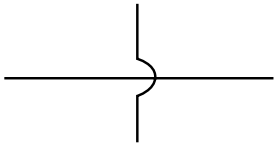
Connection



No connection

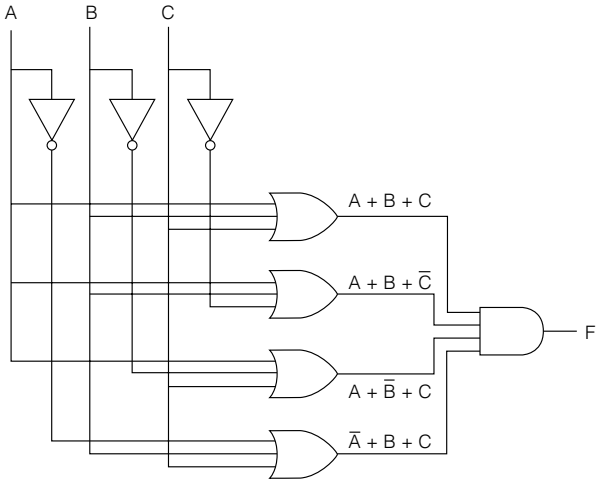


Connection



No connection

A 2-Level OR-AND Circuit that Implements the Majority Function



Positive vs. Negative Logic

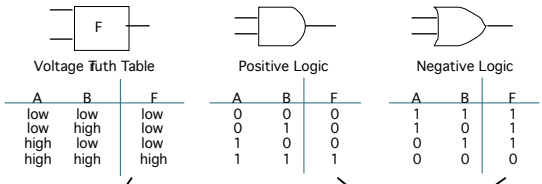
•Positive logic: truth, or assertion is represented by logic 1, higher voltage; falsity, de- or unassertion, logic 0, is represented by lower voltage.

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

Gate Logic: Positive vs. Negative Logic

Normal Convention: Postive 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

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

Behavior in terms of Electrical Levels

Two Alternative Interpretations
Positive Logic AND
Negative Logic OR

Dual Operations

Positive and Negative Logic (Cont'd.)

Voltage Levels

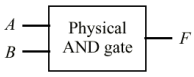
A	B	F
low	low	low
low	high	low
high	low	low
high	high	high

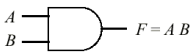
Positive Logic Levels

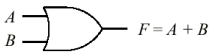
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

Negative Logic Levels

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







Voltage Levels

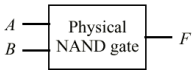
A	B	F
low	low	high
low	high	high
high	low	high
high	high	low

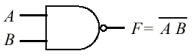
Positive Logic Levels

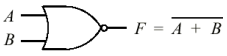
A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

Negative Logic Levels

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







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 to 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:

The Data Sheet

SN7400 QUADRUPLE 2-INPUT POSITIVE-NAND GATES

description

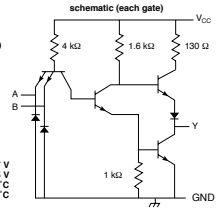
These devices contain four independent 2-input NAND gates.

function table (each gate)

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

package (top view)

schematic (each gate)



absolute maximum ratings

Supply voltage, VCC	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

logic diagram (positive logic)

$Y = \overline{A\,B}$

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

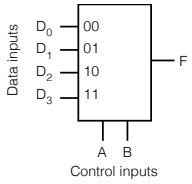
electrical characteristics over recommended operating free-air temperature range

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

switching characteristics, V_{CC} = 5 V, T_A = 25° C

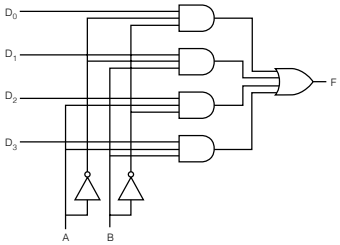
PARAMETER	FROM (input)	TO (output)	TEST CONDITIONS	MIN	NOM	MAX	UNIT
t_{PLH}	A or B	Y	R_L = 400 Ω, C_L = 15 pF		11	22	ns
t_{PHL}					7	15	ns

The Multiplexer

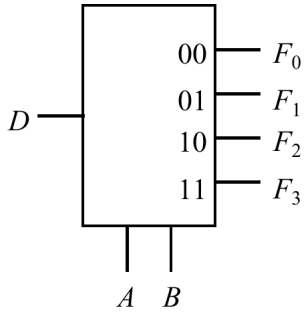


A	B	F
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

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



The Demultiplexer (DEMUX)

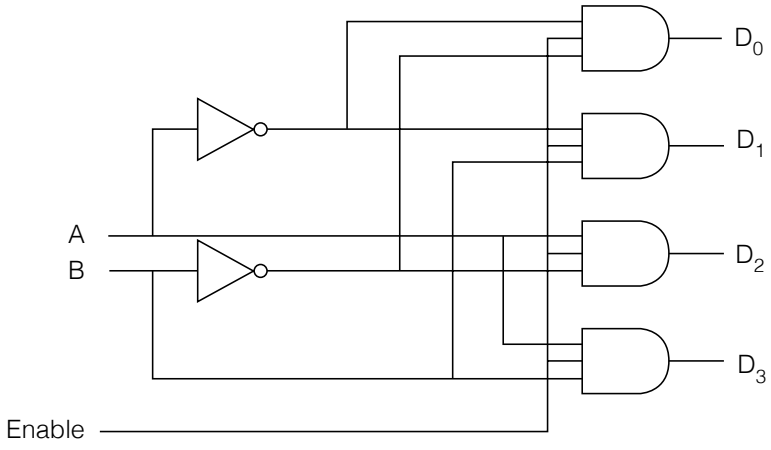


D	A	B	F_0	F_1	F_2	F_3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

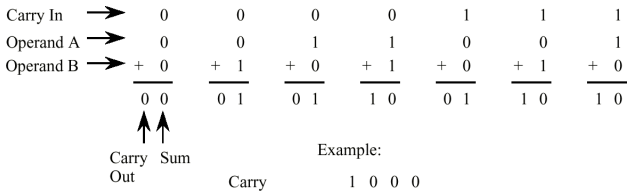
$F_0 = D\overline{A}\,\overline{B}$ $\quad F_2 = DA\overline{B}$

$F_1 = D\overline{A}B$ $\quad F_3 = DAB$

A 2-4 Decoder



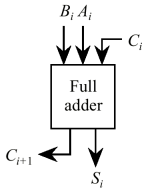
Implementing an Adder



Example:

Carry	1	0	0	0
Operand A	0	1	0	0
Operand B +	0	1	1	0
Sum	1	0	1	0

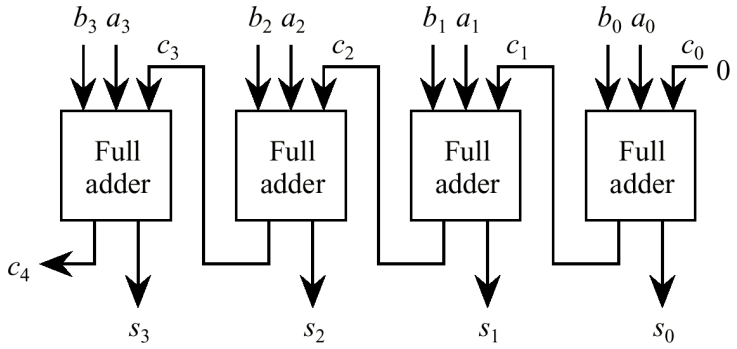
A_i	B_i	C_i	S_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



In-Class Exercise

- What are the logic functions used in a 1-bit adder?
- What is the difference between a half adder and a full adder?

A Multi-Bit Ripple-Carry Adder



In-Class Exercise

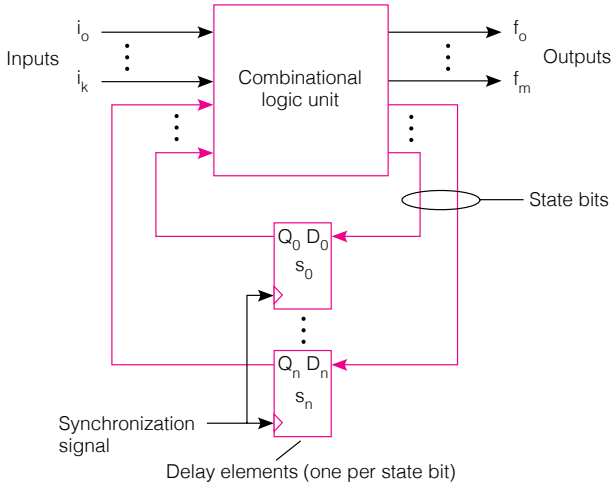
- How would you make a subtractor circuit?

Sequential Logic

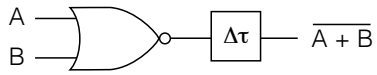
- The combinational logic circuits we have been studying so far have no memory. The outputs always follow the inputs.
- There is a need for circuits with a memory, which behave differently *depending upon their previous state.*
- An example is the vending machine, which must remember how many and what 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.

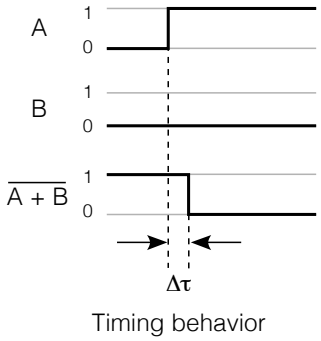
Classical Model of a Finite State Machine (FSM)

An FSM is composed of a combinational logic unit and delay elements (called *flip-flops*) in a feedback path, which maintains state information.



A NOR Gate with a Lumped Delay

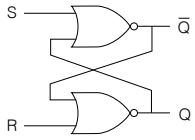




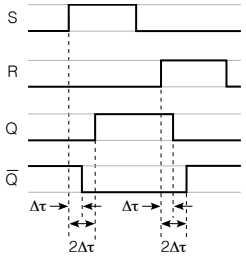
Timing behavior

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

The S-R (Set-Reset) Flip-Flop



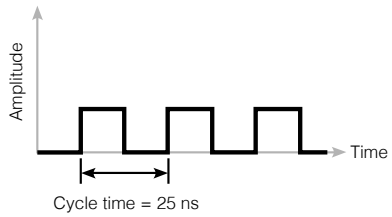
Q_t	S_t	R_t	Q_{t+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	(disallowed)
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	(disallowed)



Timing behavior

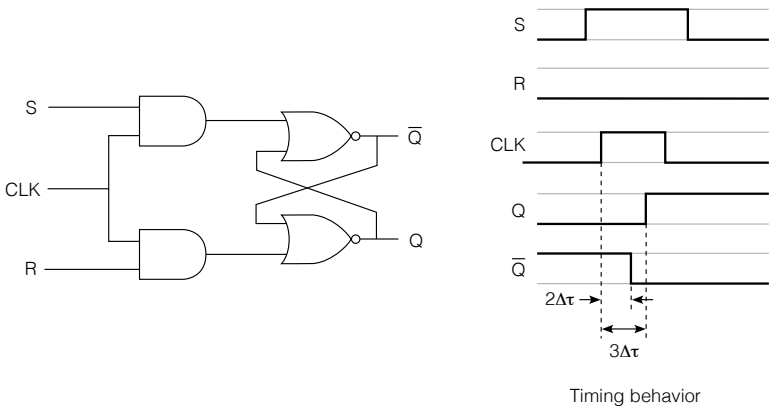
The S-R flip-flop is an active high (positive logic) device.

The Clock Paces the System



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

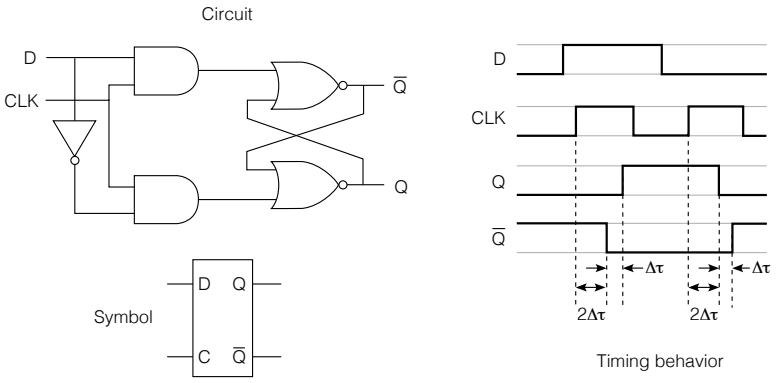
The clock signal, CLK, turns on the inputs to the flip-flop.

Scientific Prefixes

- **For computer memory, 1K = 2^{10} = 1024. For everything else, like clock speeds, 1K = 1000, and likewise for 1M, 1G, *etc.***

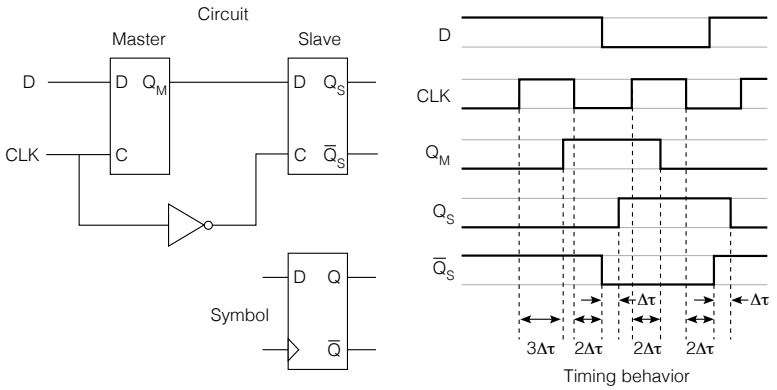
Prefix	Abbrev.	Quantity	Prefix	Abbrev.	Quantity
milli	m	10^{-3}	Kilo	K	10^{3}
micro	μ	10^{-6}	Mega	M	10^{6}
nano	n	10^{-9}	Giga	G	10^{9}
pico	p	10^{-12}	Tera	T	10^{12}
femto	f	10^{-15}	Peta	P	10^{15}
atto	a	10^{-18}	Exa	E	10^{18}

The Clocked D (Data) Flip-Flop



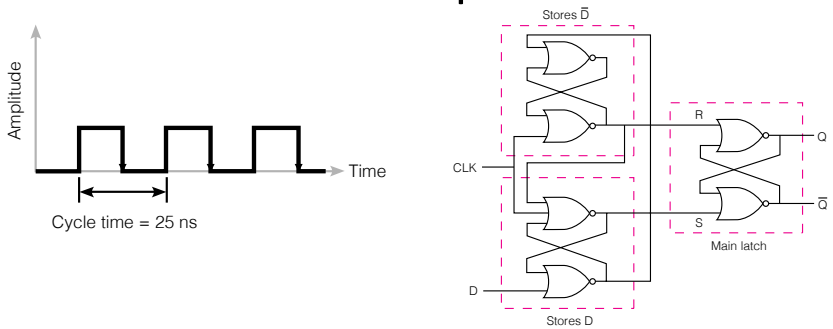
The clocked D flip-flop, sometimes called a latch, has a potential problem: If D changes while the clock is high, the output will also change. The Master-Slave flip-flop solves this problem:

The 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.

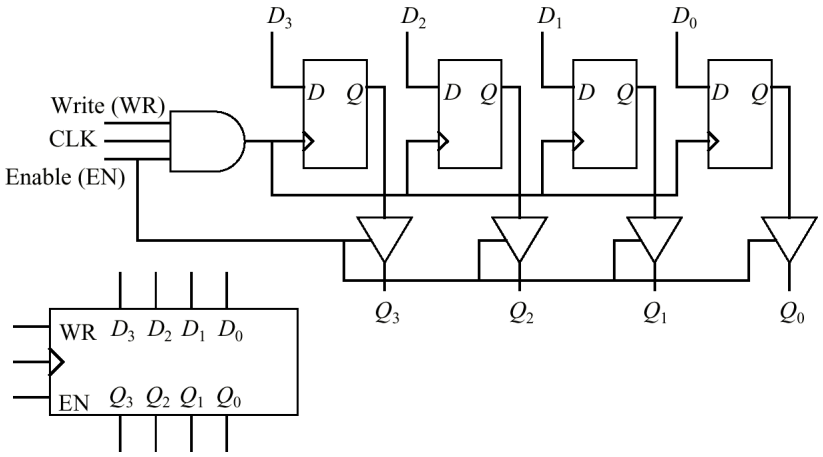
The Negative Edge-Triggered D Flip-Flop



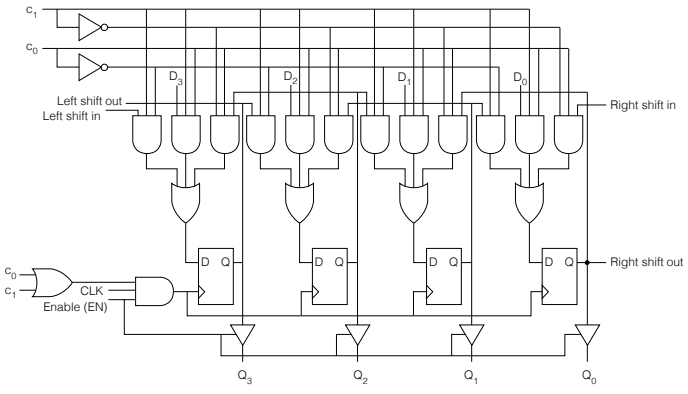
- When the clock is high, the two input latches output 0, so the Main latch remains in its previous state, regardless of changes in D.
- When the clock goes high-low, values in the two input latches will affect the state of the Main latch.
- While the clock is low, D cannot affect the Main latch.

Four-Bit Register

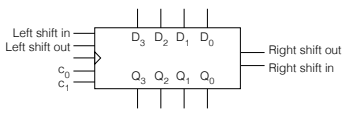
- **Makes use of tri-state buffers so that multiple registers can gang their outputs to common output lines.**



A Left-Right Shift Register with Parallel Read and Write



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



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:

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

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

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

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

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

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

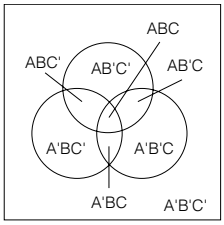
$F = \overline{A}BC + AC + AB$ Complement and Identity

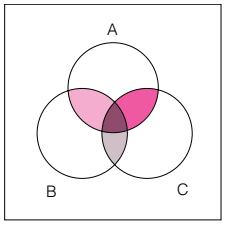
$F = \overline{A}BC + AC + AB + ABC$ Idempotence

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

$F = BC + AC + AB$ Complement and Identity

Venn Diagrams



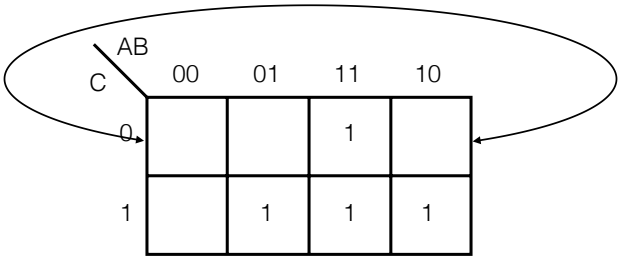


Each distinct region in the "Universe" represents a minterm.
This diagram can be transformed into a Karnaugh Map.

A K-Map of the Majority Function

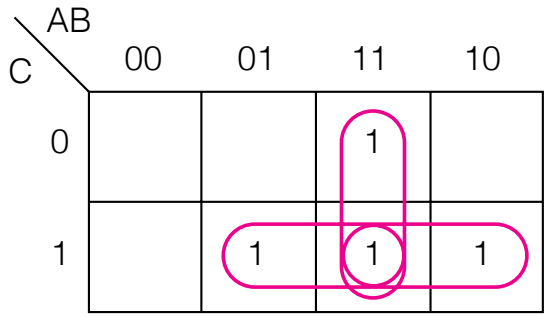
Place a "1" in each cell that has a that minterm.
Cells on the outer edge of the map "wrap around"

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



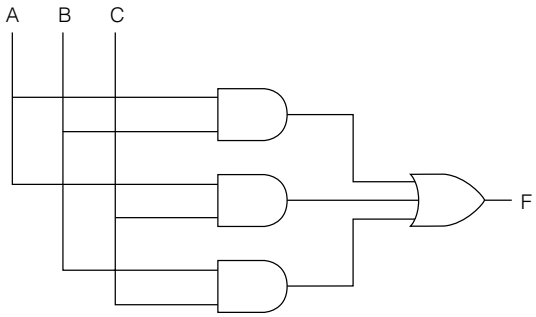
The map contains all the minterms. Adjacent 1's in the K-Map satisfy the Complement property of Boolean Algebra.

Adjacency Groupings for the Majority Function



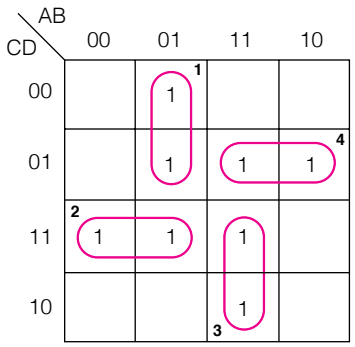
M= BC + AC + AB

Minimized AND OR Circuit for the Majority Function

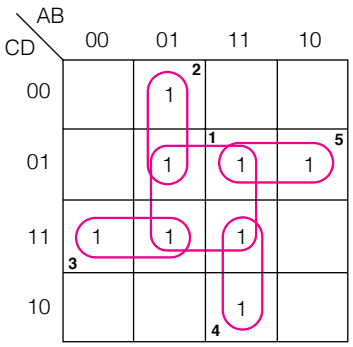


M= BC + AC + AB

Minimal and not Minimal Groupings

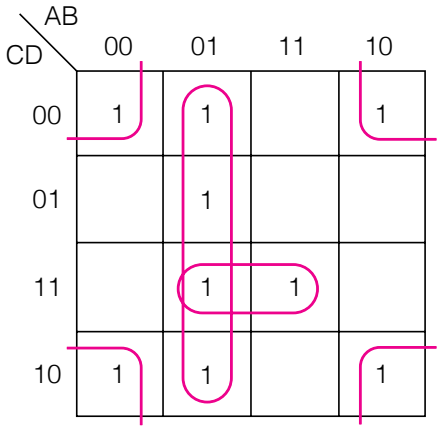


$F = \bar{A}B\bar{C} + \bar{A}CD + ABC + A\bar{C}D$



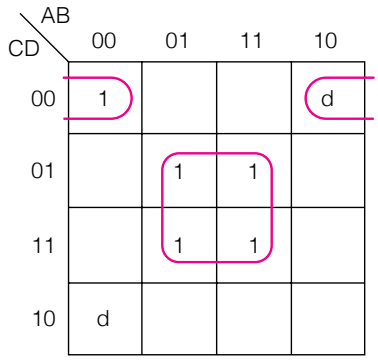
$F = BD + \bar{A}B\bar{C} + \bar{A}CD + ABC + A\bar{C}D$

The Corners are Logically Adjacent

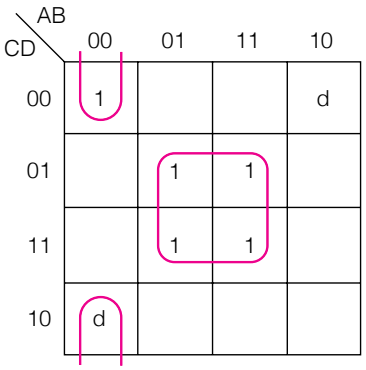


$F = BCD + \bar{B}\bar{D} + \bar{A}B$

Two Different Minimized Equations



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

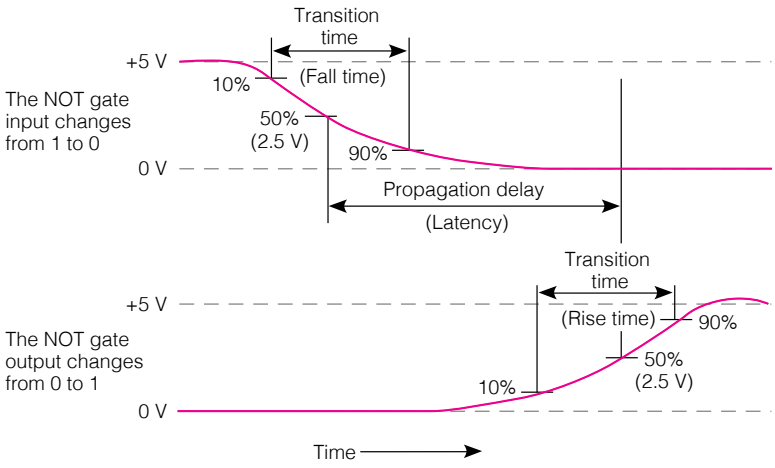


$F = \overline{A}\overline{B}\overline{D} + BD$

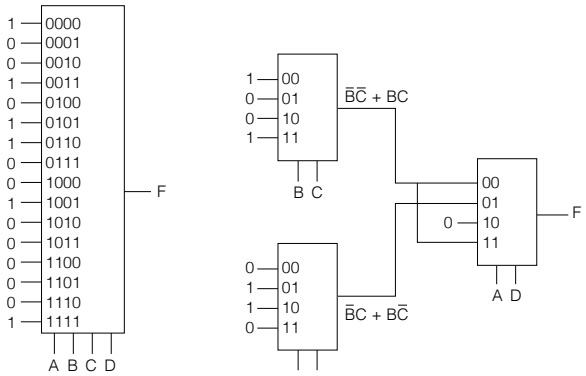
Speed and Performance

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

Propagation Delay for a NOT Gate (From Hamacher et. al. 1990)



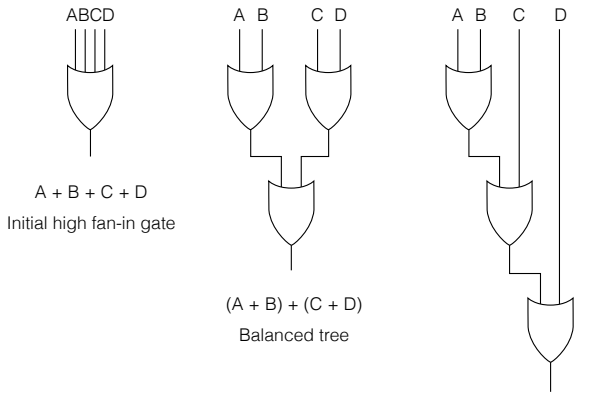
Circuit Depth Affects Propagation Delay



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

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

Fan-in may Affect Circuit Depth



A + B + C + D
Initial high fan-in gate

(A + B) + (C + D)
Balanced tree

((A + B) + C) + D
Degenerate tree

Associative law of Boolean algebra:
A + B + C + D = (A + B) + (C + D)