Digital Logic Layer

Your PC ran into a problem and needs to restart. This BSOD (blue screen of death) was faked for the sake of this presentation.

Computer System Organization with Assembly Language: CS_212

Gates and Boolean Algebra

Many combinations can be made from a limited number of basic components to create digital circuits.

Gates

A circuit with only two logical values is called a digital circuit. Generally, one value is represented by a signal between 0 and 0.5 volts, and the other value is represented by a signal between 1 and 1.5 volts.

Gates

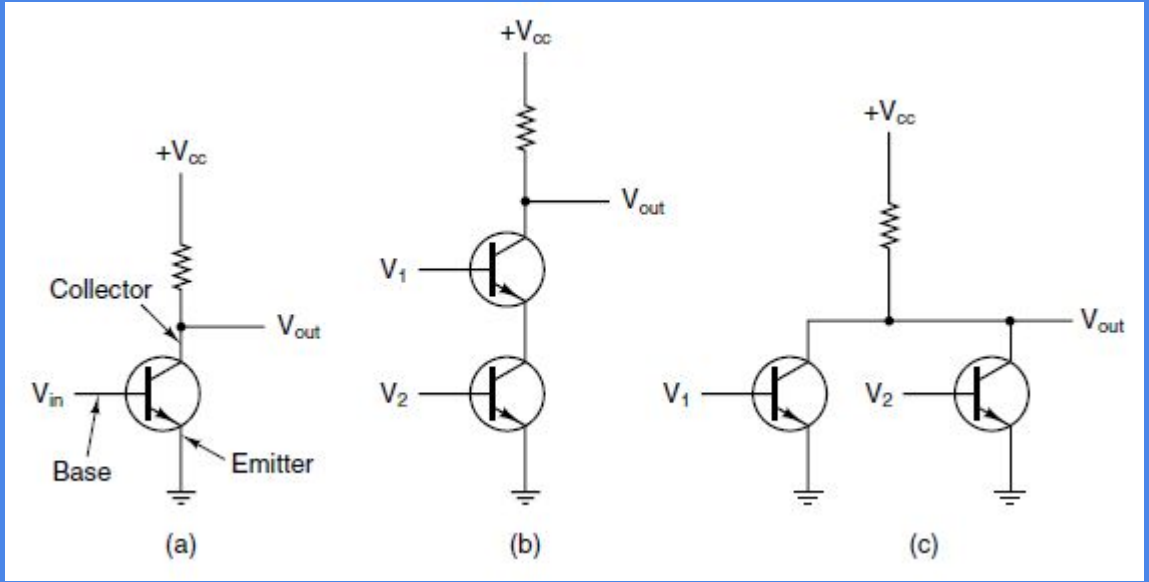
These two-valued signals can be used to compute a variety of functions via tiny electrical devices known as *gates*. These gates comprise the hardware basis on which all digital computers are created.

Gates

The foundation of all contemporary digital logic is the ability to use a transistor to function as an extremely quick binary switch. The *collector*, *base*, and *emitter* of this transistor are its three external connections.

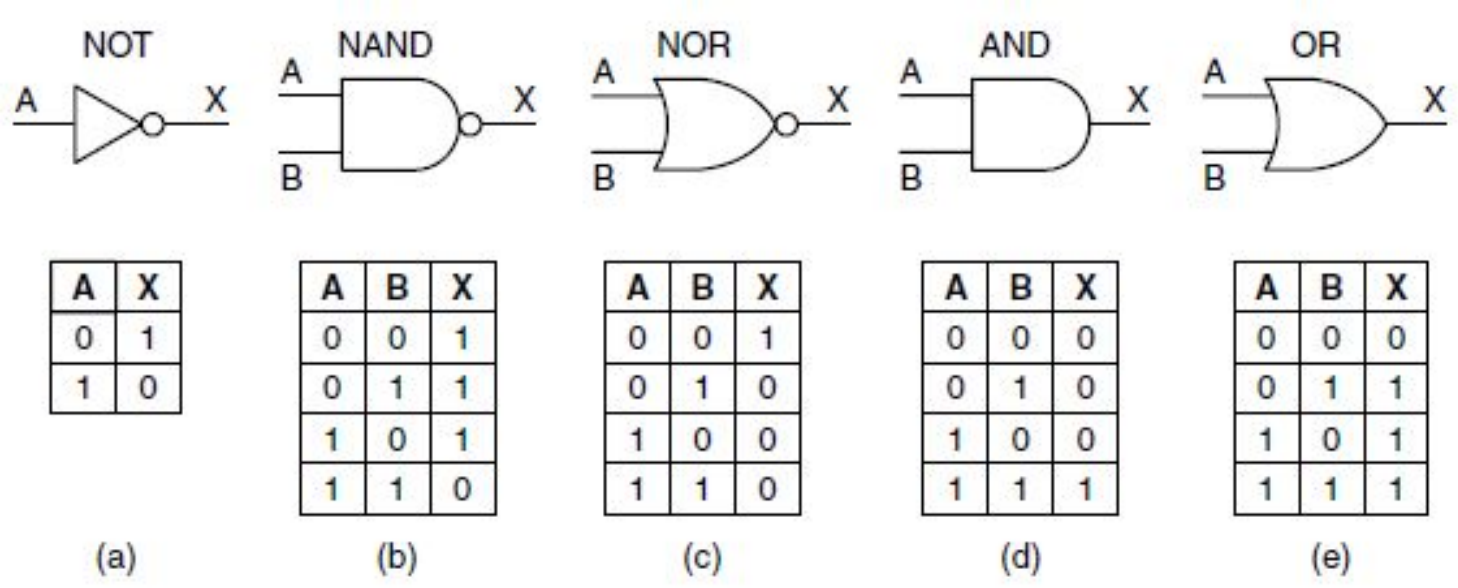
:(

Gates



(a) A transistor inverter. (b) A NAND gate. (c) A NOR gate

Gates



A	X
0	1
1	0

(a)

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

(b)

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

(c)

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(d)

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

(e)

The symbols and functional behavior for the five basic gates.

Boolean Algebra

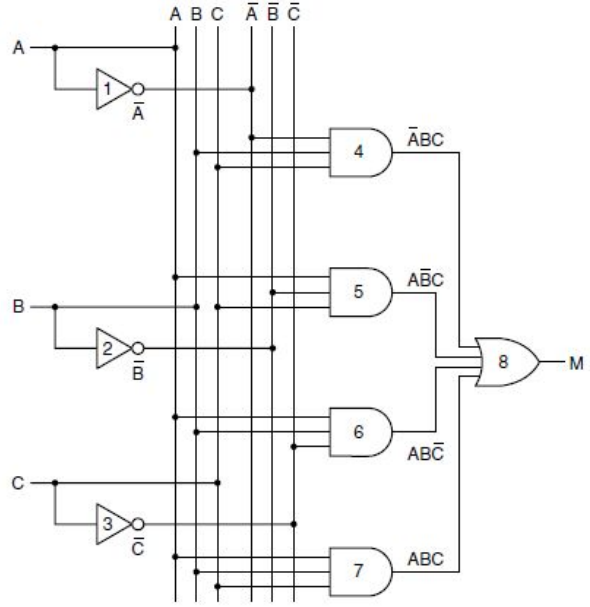
A new kind of algebra is required, one in which variables and functions can only have the values 0 and 1, in order to characterize the circuits that can be constructed by joining gates. Named for its discoverer, the English mathematician George Boole (1815–1864), such an algebra is known as a Boolean algebra.

:(

Boolean Algebra

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

(a)



(b)

(a) The truth table for the majority function of three variables. (b) A circuit for (a)

Boolean Algebra

In this example, ABC, for instance, only accepts the value 1 when A = 1, B = 0, and C = 1. Furthermore, only when (A = 1 and B = 0) or (B = 1 and C = 0) occur does AB + BC equal 1.

Boolean Algebra

The output's four rows that are yielding 1 bit are ABC, ABC, ABC, and ABC. If any one of these four conditions holds true, the function M is true (i.e., 1), thus we can write:

M = ABC + ABC + ABC + ABC

Implementation of Boolean Functions

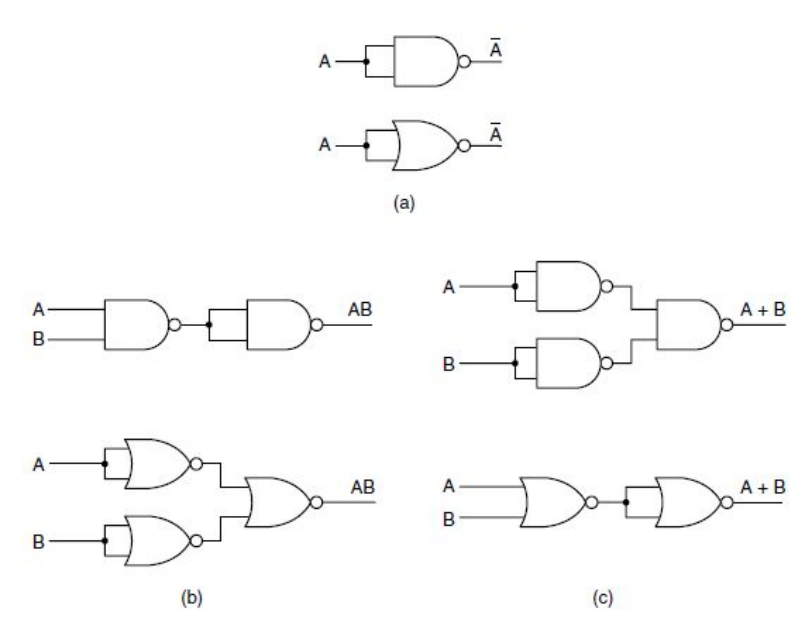
1. Write down the truth table for the function.
2. Provide inverters to generate the complement of each input.
3. Draw an AND gate for each term with a 1 in the result column.
4. Wire the AND gates to the appropriate inputs.
5. Feed the output of all the AND gates into an OR gate.

Circuit Equivalence

In order to decrease power consumption, boost speed, and reduce the amount of chip area required to implement the gates, circuit designers frequently aim to limit the number of gates in their devices. To minimize the complexity of a circuit, the designer must identify another circuit that computes the same function as the original but does it with fewer gates

:(

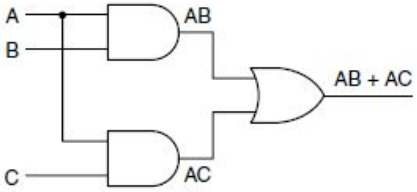
Circuit Equivalence



Construction of (a) NOT, (b) AND, and (c) OR gates using only NAND gates or only NOR gates.

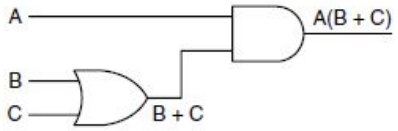
:(

Circuit Equivalence



A	B	C	AB	AC	AB + AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

(a)



A	B	C	A	B + C	A(B + C)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

(b)

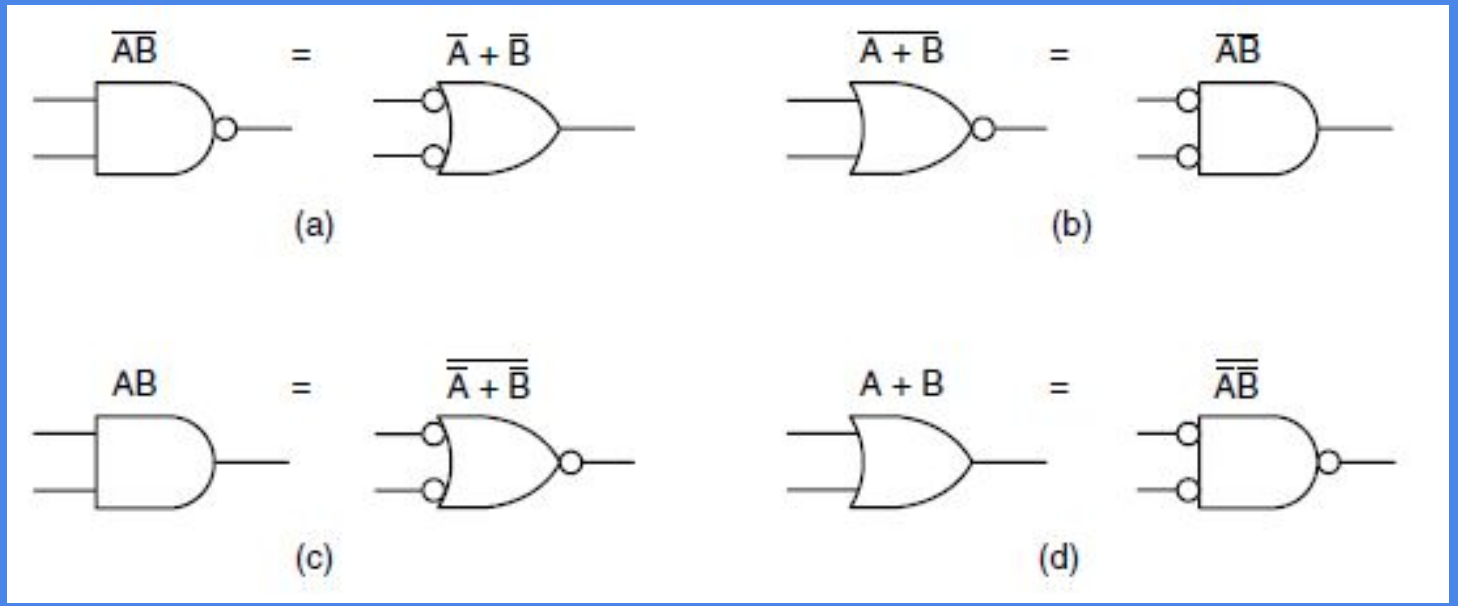
Two equivalent functions. (a) AB + AC. (b) A(B + C)

Circuit Equivalence

Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\overline{A} = 0$	$A + \overline{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\,\overline{B}$

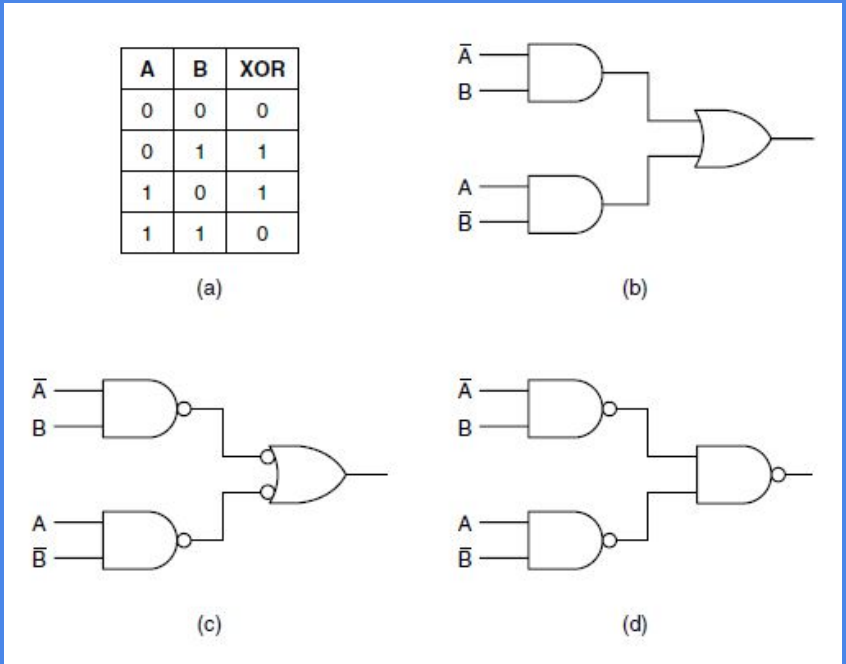
Some identities of Boolean algebra

Circuit Equivalence



Alternative symbols for some gates: (a) NAND (b) NOR (c) AND (d) OR

:(Circuit Equivalence



A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

(a)

(b)

(c)

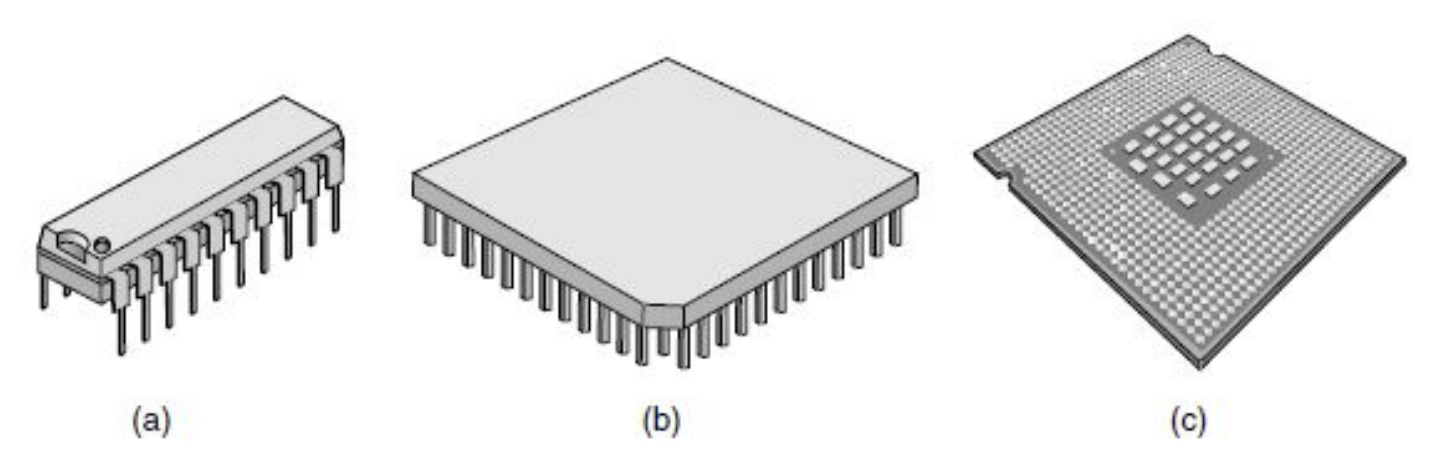
(d)

(a) The truth table for the XOR function. (b)–(d) Three circuits for computing it

Basic Digital Circuits

Computer System Organization with Assembly Language: CS_212

Integrated Circuits



Common types of integrated-circuit packages, including a dual-in-line package (a), pin grid array (b), and land grid array (c).

Combinational Circuits

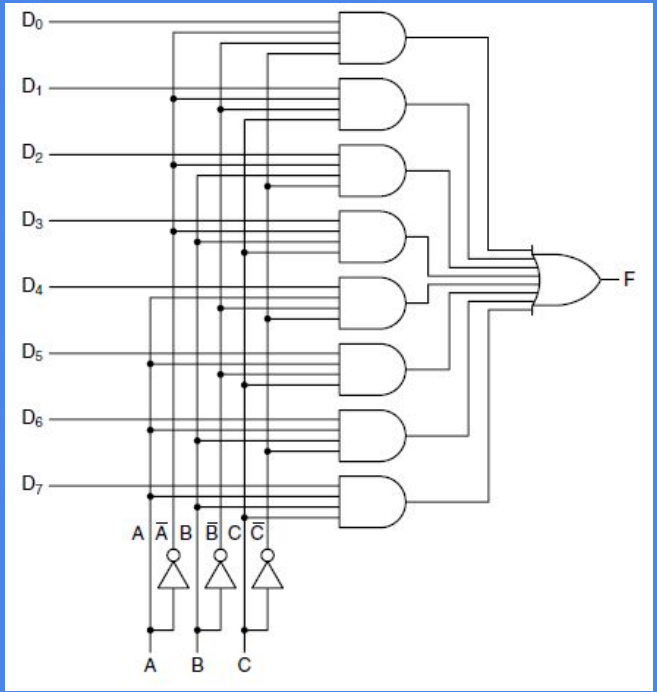
A circuit with numerous inputs and outputs, where each output is uniquely determined by the current input values, is necessary for many applications of digital logic. We refer to this type of circuit as a *combinational circuit*.

Multiplexers

A *multiplexer* is a circuit with two *n* data inputs, one data output, and n control inputs that pick one of the data inputs at the digital logic level. The chosen data input is "gated," or transmitted, to the output.

:(

Multiplexers



An eight-input multiplexer circuit.

Multiplexers

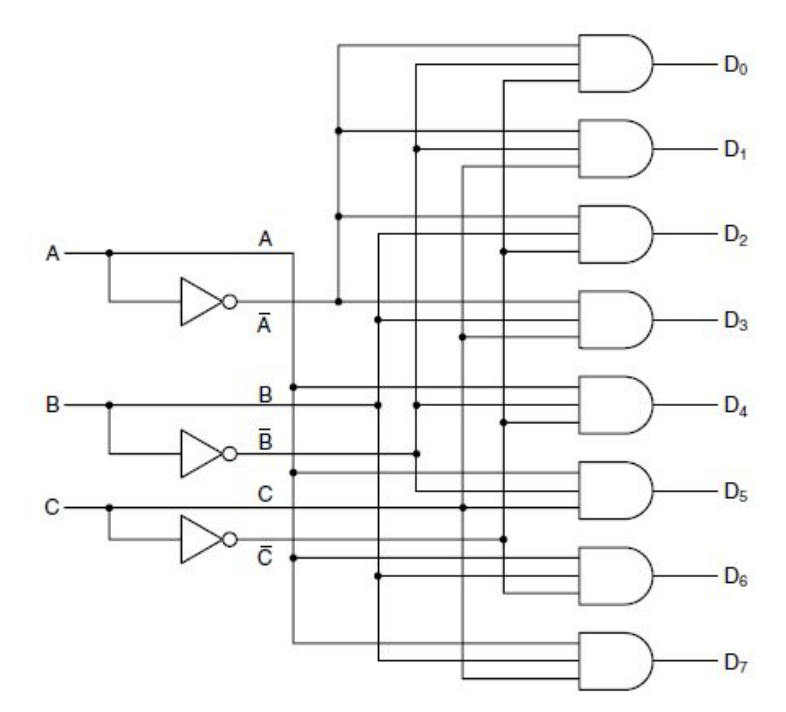
The opposite of a multiplexer is a *demultiplexer*, which, based on the values of the *n* control lines, directs its single input signal to one of *n* outputs.

Decoder

A *decoder* is a circuit that takes an n-bit number as input and uses it to select (i.e., set to 1) exactly one of the 2*n* output lines.

:(

Decoders



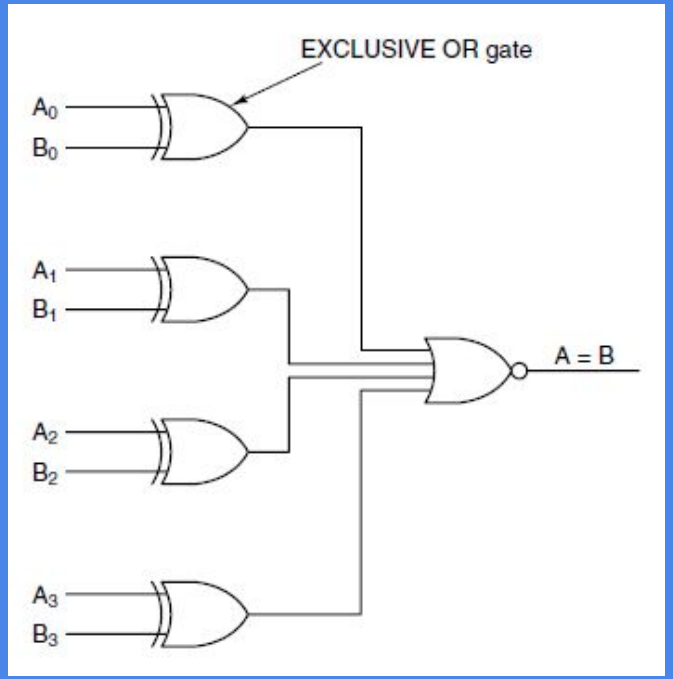
A three-to-eight decoder circuit

Comparator

The *comparator* circuit, which compares two input words. The basic comparator accepts two inputs, *A* and *B*, each of length *4* bits, and generates a *1* if they are equal and a *0* otherwise.

:(

Comparator

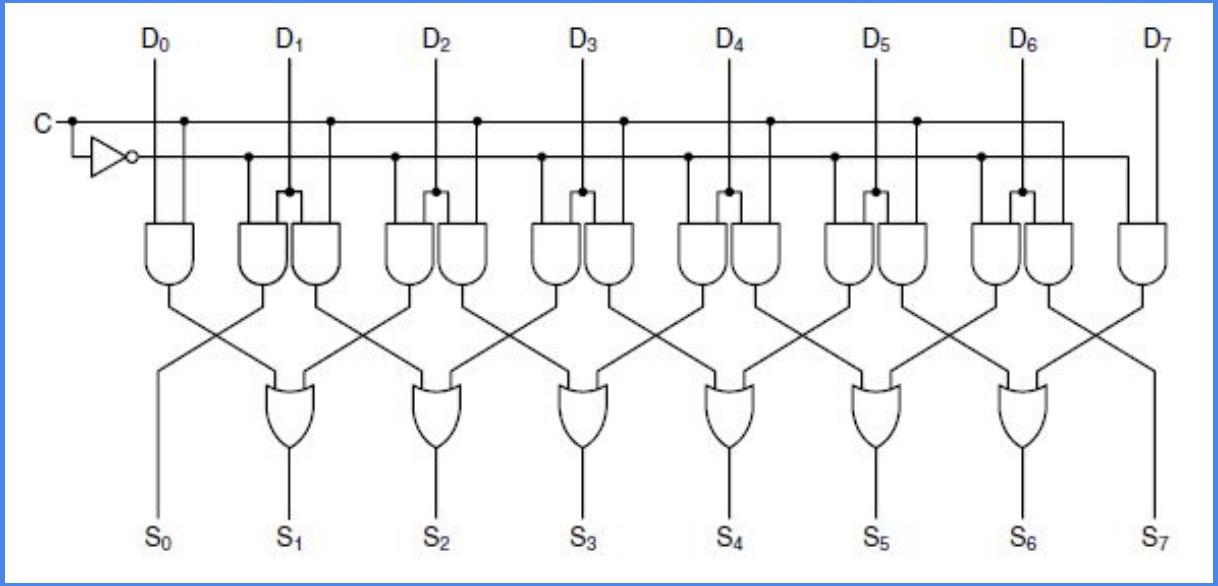


A simple 4-bit comparator.

Arithmetic Circuits

:(

Shifters



A 1-bit left/right shifter

Shifters

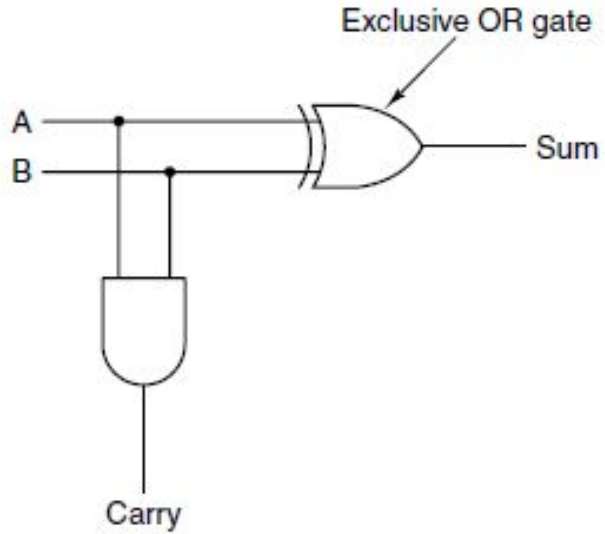
A shifter with eight inputs and eight outputs is our first arithmetic circuit. The input consists of eight bits on lines *D0,. ....D7*. Lines *S0*, provide the output, which is simply the input shifted by one bit. ....*S7*. The shift's direction is indicated by the control line, *C*, which reads *0* for left and *1* for right. Bit *7* receives a *0* when there is a shift to the left. In a same manner, bit *0* receives a *1* upon a right shift.

Adders

An addition hardware circuit is a fundamental component of all CPUs. There are two outputs: the carry to the next (leftward) position and the sum of the inputs, *A* and *B*. Most people refer to this straightforward circuit as a *half adder*.

Adders

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



(a) Truth table for 1-bit addition. (b) A circuit for a half adder.

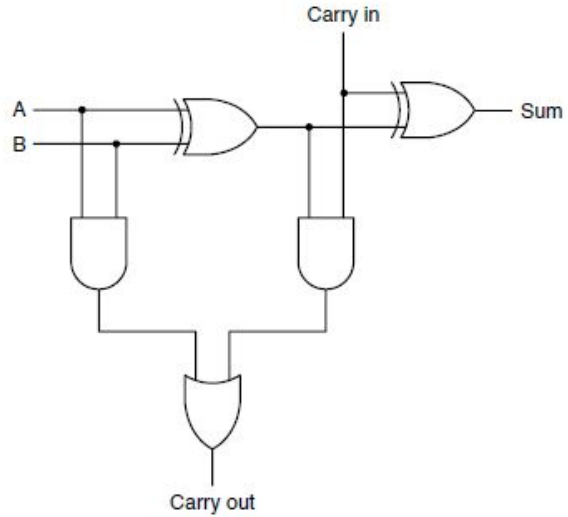
Adders

Two half adders are used to construct a *full adder*. If an odd number of *A*, *B*, and the carry in are *1*, then the *sum* output line is *1*. If both *A* and *B* (the left input to the OR gate) are *1* or if precisely one of them is *1* and the Carry in bit is also *1*, then the carry out is *1*. The carry and sum bits are produced by the two half adders working together.

:(Adders

A	B	Carry in	Sum	Carry out
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

(a)



(b)

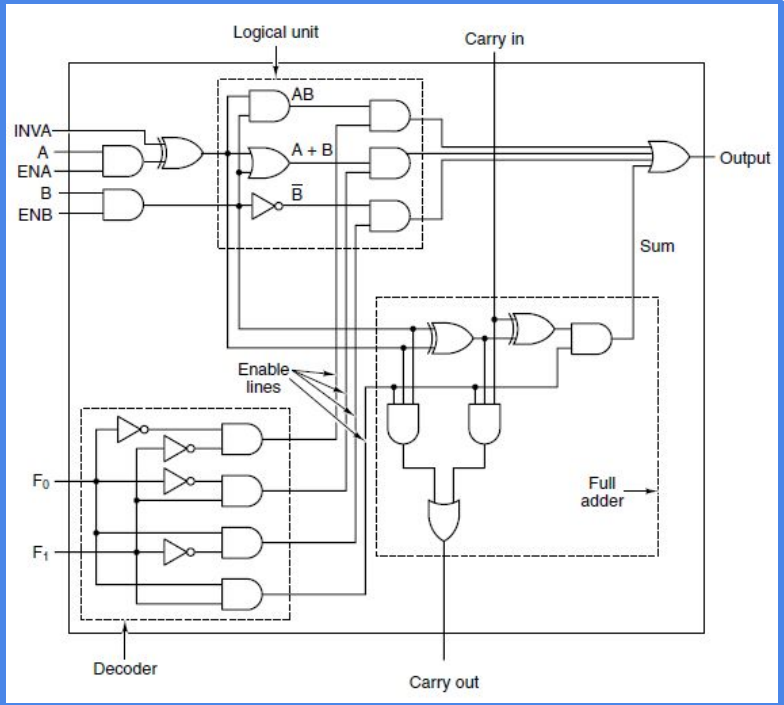
(a) Truth table for full adder. (b) Circuit for a full adder

Arithmetic Logic Units

The AND, OR, and sum of two machine words are performed by a single circuit found in the majority of computers. A circuit of this type for n-bit words consists of n circuits that are the same for each bit location. *Arithmetic Logic Units*, or *ALUs*, are a type of basic circuit example.

:(

Arithmetic and Logic Units



A 1-bit ALU.

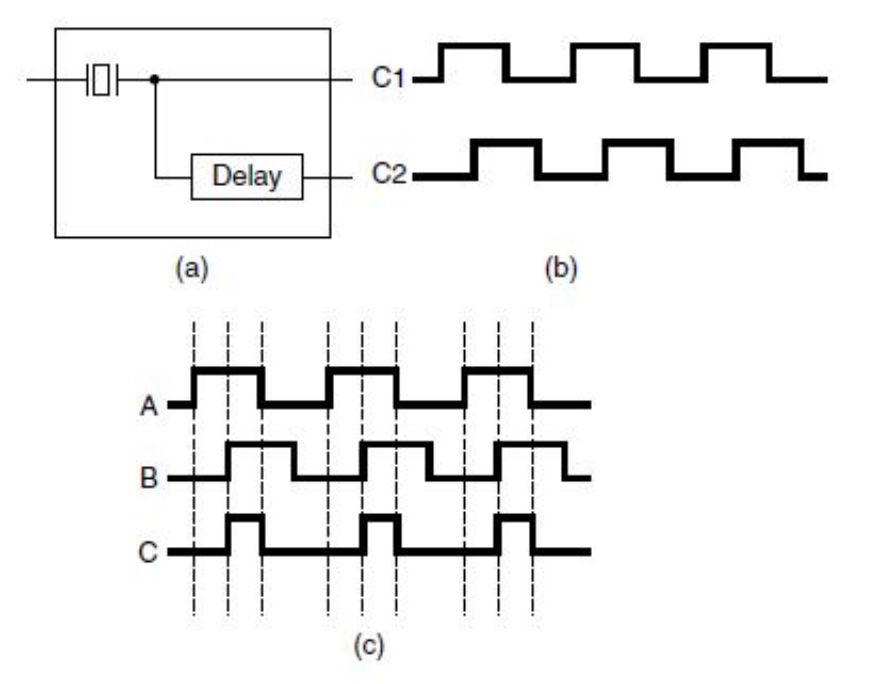
Clocks

Clocks are used in many digital circuits to provide synchronization, enabling designers to achieve the necessary time relations. In this usage, a clock is a circuit that pulses repeatedly at a specified pulse width and interval between pulses.

Clocks

The *clock cycle* time is the amount of time that separates the corresponding edges of two successive pulses. The typical range of pulse frequencies is 100 MHz to 4 GHz, which translates to clock cycles of 10 ns to 250 psec. A *crystal oscillator* is often used to regulate the clock frequency in order to get high precision.

:(Clocks



(a) A clock. (b) The timing diagram for the clock. (c) Generation of an asymmetric clock

Clocks

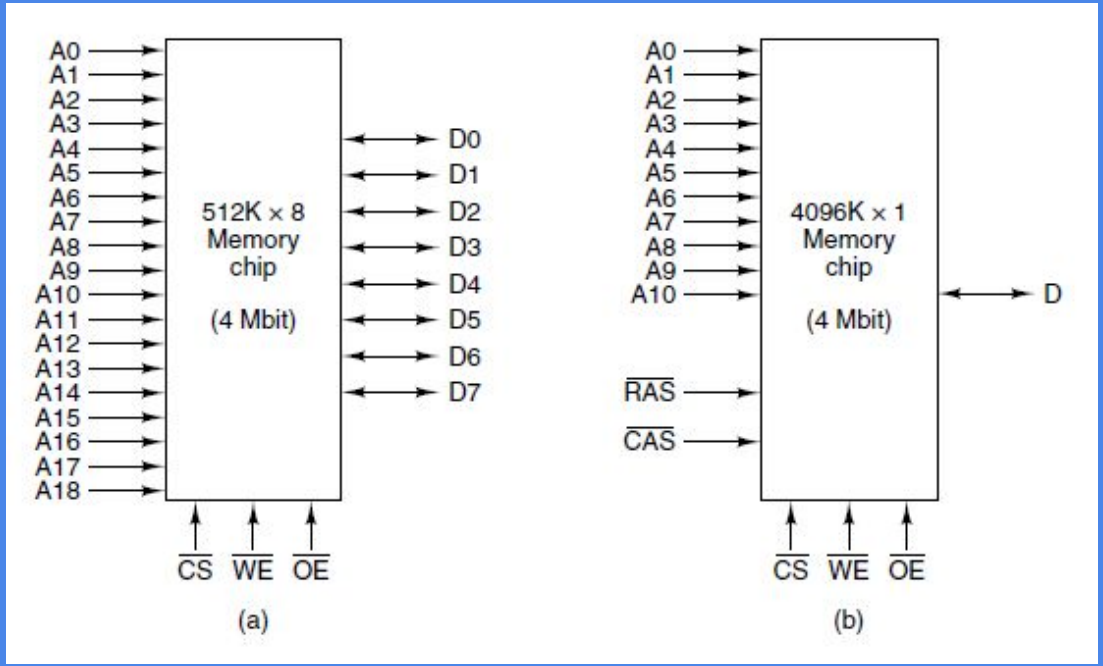
For discrete events, the timing diagram offers four time references:

1. C1's rising edge.
2. C1's falling edge.
3. Upward C2 edge.
4. C2's declining edge.

Memory

An vital component of every computer is its *memory*. There could be no computers as we know them today without memory. Data and instructions to be performed are both stored in memory. We will look at the fundamental parts of a memory system, starting at the gate level, in the following sections to see how they function and are combined to create enormous memories.

:(Memory



Two ways of organizing a 4-Mbit memory chip.

Thank You!

Your PC ran into a problem and needs to restart. This BSOD (blue screen of death) was faked for the sake of this presentation.