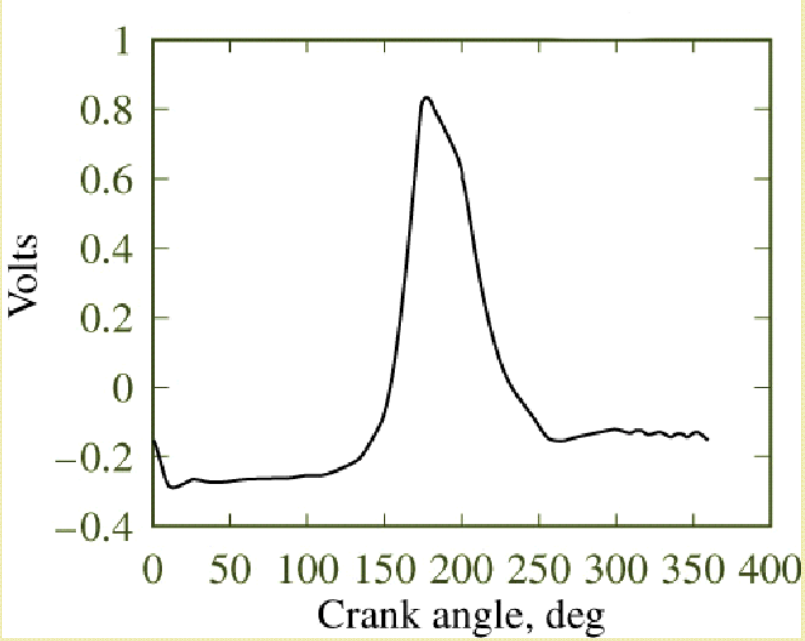
Digital Logic Circuits

- Let's look at the essential features of digital logic circuits, which are at the heart of digital computers.
- Learning Objectives
  - Understand the concepts of analog and digital signals and quantization
  - Know the differences between combinational and sequential logic
  - Write truth tables and realize logic functions from truth tables by using logic gates

- Be able to design logic circuits
- Be able to find a Boolean expression given a truth table
- Be able to use a variety of flip-flops

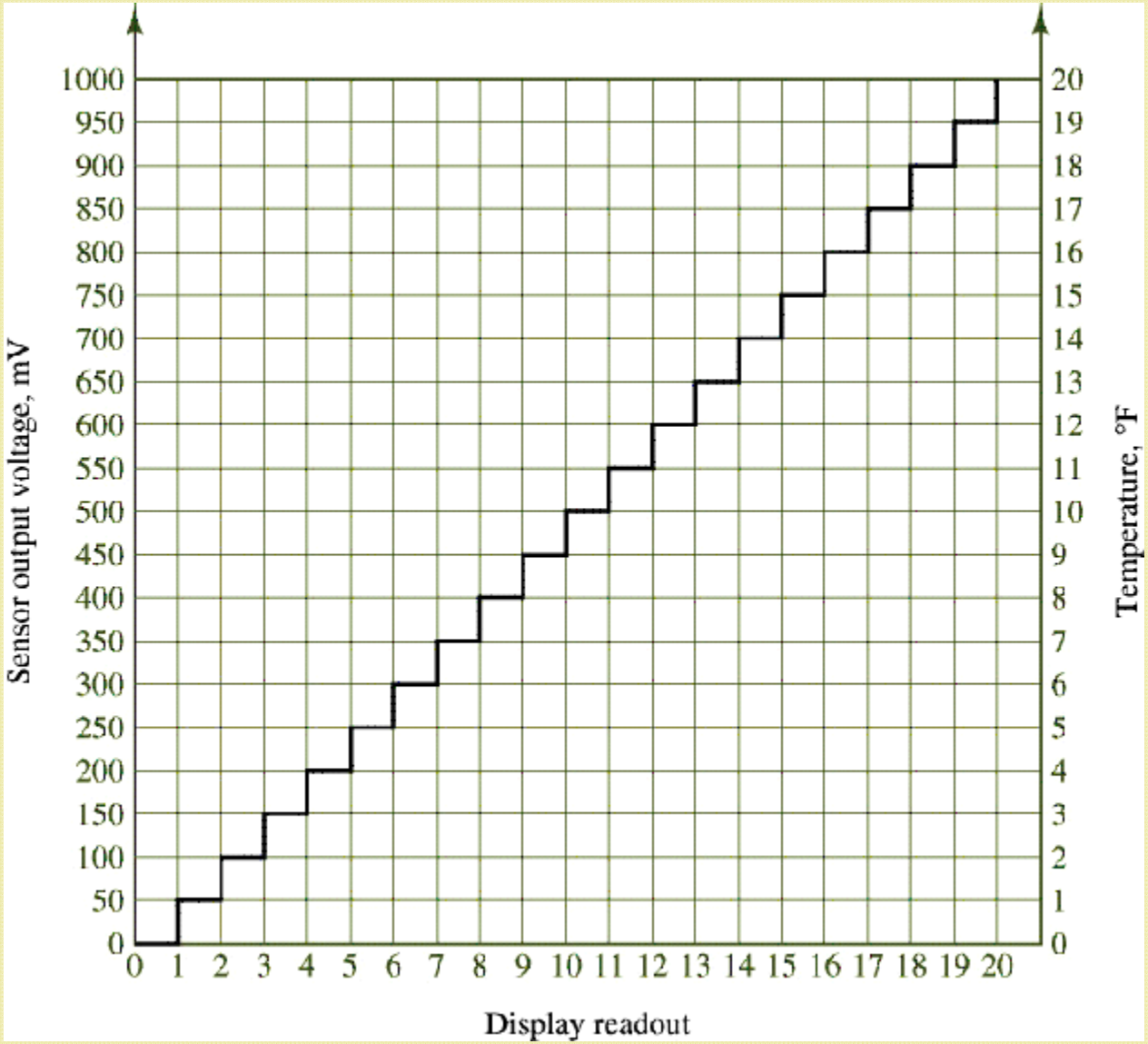
- Analog and Digital Signals
  - An analog signal is an electric signal whose value varies in analogy with a physical quantity, e.g., temperature, force, acceleration, etc.
    - For example, a voltage, v(t), proportional to a measured variable pressure, p(t), naturally varies in an analog fashion.

For each value of t, v(t) can take one value among any of the values in a given range.

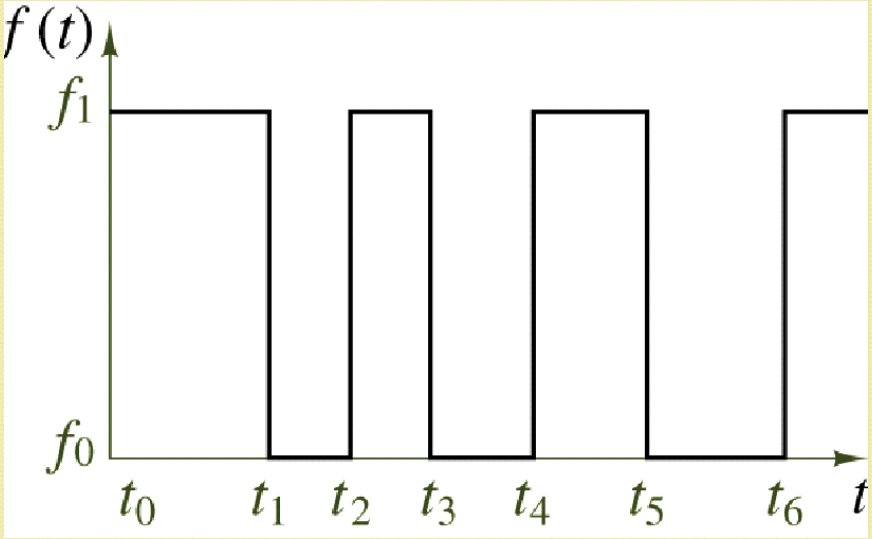


- A digital signal can take only a finite number of values.
  - An example is a signal that allows display of a temperature measurement on a digital readout.
  - Suppose that the digital readout is three digits long and can display numbers from 0 to 100. Assume that the temperature sensor is calibrated to measure temperatures from 0 to 100ºC and that the output of the sensor ranges from 0 to 5V, i.e., 20ºC per volt.
  - The sensor output is an analog signal, but the digital display can take a value from a discrete set of states, the integers from 0 to 100.
  - Each digit on the display represents 1/100 of the 5V range, or 0.05V = 50 mV.
  - Note the staircase function relationship between the analog voltage and the digital readout – the quantization of the sensor output voltage.

Digital Representation on an Analog Signal



– A binary signal, the most common digital signal, is a signal that can take only one of two discrete values and is therefore characterized by transitions between two states.



– In binary arithmetic, the two discrete values f_1 and f_0 are represented by the numbers 1 and 0, respectively.

- In binary voltage waveforms, these values are represented by two voltage levels.
  - In TTL convention, these values are nominally 5V and 0V, respectively.
- Note that in a binary waveform, knowledge of the transition between one state and another is equivalent to knowledge of the state. Thus, digital logic circuits can operate by detecting transitions between voltage levels. The transitions are called edges and can be positive (f_0 to f_1) or negative (f_1 to f_0).

- Combinational and Sequential Logic
  - Sequential Logic Devices
    - The timing, or sequencing history, of the input signals plays a role in determining the output.
  - Combinational Logic Devices
    - The outputs depend only on the instantaneous values of the inputs.
    - These devices convert binary inputs into binary outputs based on the rules of mathematical logic.

- Boolean Algebra
  - The mathematics associated with the binary number system (and with the more general field of logic) is called boolean (George Boole, English Mathematician, circa 1850).
  - The variables in a boolean, or logic, expression can take only one of two values, 0 (false) and 1 (true).
  - Analysis of logic functions (functions of boolean variables) can be carried out in terms of truth tables.
    - A truth table is a listing of all possible values that each of the boolean variables can take, and of the corresponding value of the desired function.
  - Logic gates are physical devices that can be used to implement logic functions. They control the flow of signals from the inputs to the single output.

- The basis of boolean algebra lies in the operations of logical addition, or the OR operation, and logical multiplication, or the AND operation.
- OR Gate
  - If either X or Y is true (1), then Z is true (1)

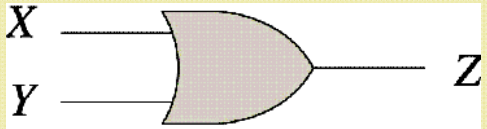
$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 1$$

- AND Gate
  - If both X and Y are true (1), then Z is true (1)

$$0 \cdot 0 = 0$$
$$0 \cdot 1 = 0$$
$$1 \cdot 0 = 0$$
$$1 \cdot 1 = 1$$

- Logic gates can have an arbitrary number of inputs.

- The rules that define a logic function are often represented in tabular form by means of a truth table, i.e., a tabular summary of all possible outputs of a logic gate, given all possible input values.
- Truth tables are very useful in defining logic functions.



OR gate

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Truth table

AND gate

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

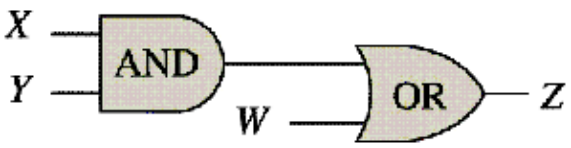
Truth table

– Logic Design Example

- Determine the combination of logic gates that exactly implements the required logic function.
- Statement: The output Z shall be logic 1 only when condition (X =1 AND Y =1) OR (W = 1) occurs and shall be logic 0 otherwise.

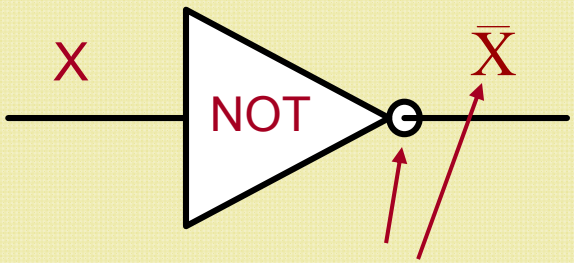
X	Y	W	Z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Truth table



Solution using logic gates

– NOT Gate (inverter)



X	$\bar{X}$
1	0
0	1

Truth Table

Note: small circle and overbar denotes signal inversion

– We make frequent use of truth tables to evaluate logic expressions. A set of rules will facilitate this task. The following set of rules and identities can be used to simplify logic expressions.

$$0 + X = X \qquad 0 \bullet X = 0 \qquad \bar{\bar{X}} = X$$

$$1 + X = 1 \qquad 1 \bullet X = X \qquad X + Y = Y + X$$

$$X + X = X \qquad X \bullet X = X \qquad X \bullet Y = Y \bullet X$$

$$X + \bar{X} = 1 \qquad X \bullet \bar{X} = 0 \qquad X + (Y + Z) = (X + Y) + Z$$

$$X \bullet (Y \bullet Z) = (X \bullet Y) \bullet Z$$

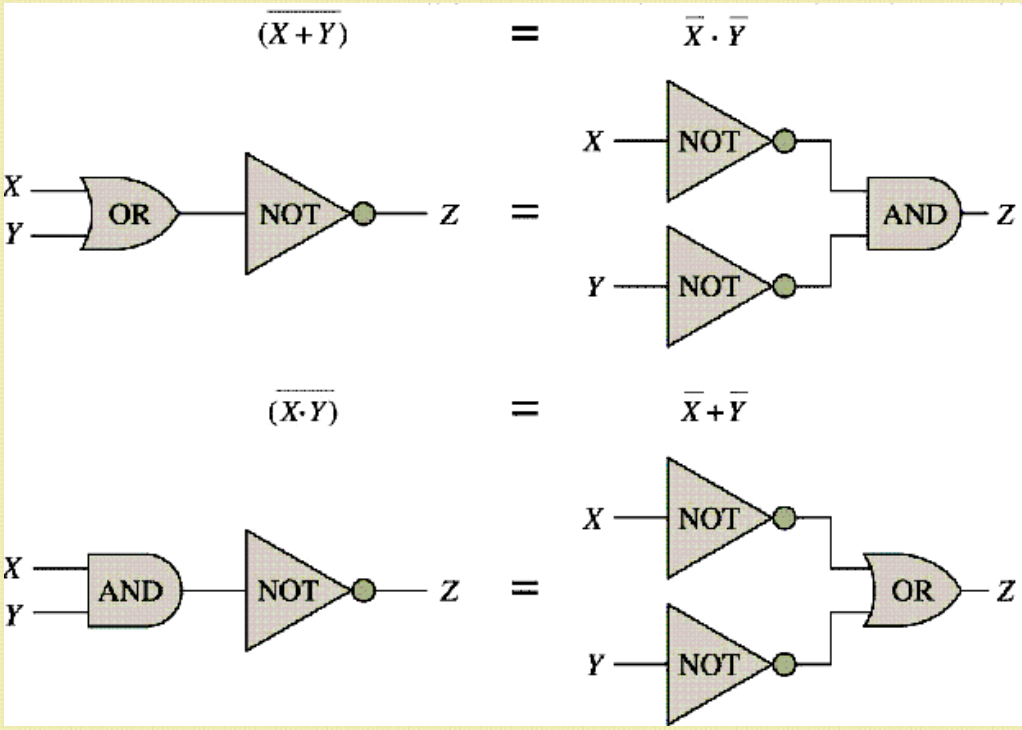
$$X \bullet (Y + Z) = X \bullet Y + X \bullet Z$$

$$X + (Y \bullet Z) = (X + Y) \bullet (X + Z) \qquad (X + Y) \bullet (X + Z) = X + (Y \bullet Z)$$

$$X + (X \bullet Z) = X \qquad X + (\bar{X} \bullet Y) = X + Y$$

$$X \bullet (X + Y) = X \qquad (X \bullet Y) + (Y \bullet Z) + (\bar{X} \bullet Z) = (X \bullet Y) + (\bar{X} \bullet Z)$$

– DeMorgan's Theorems



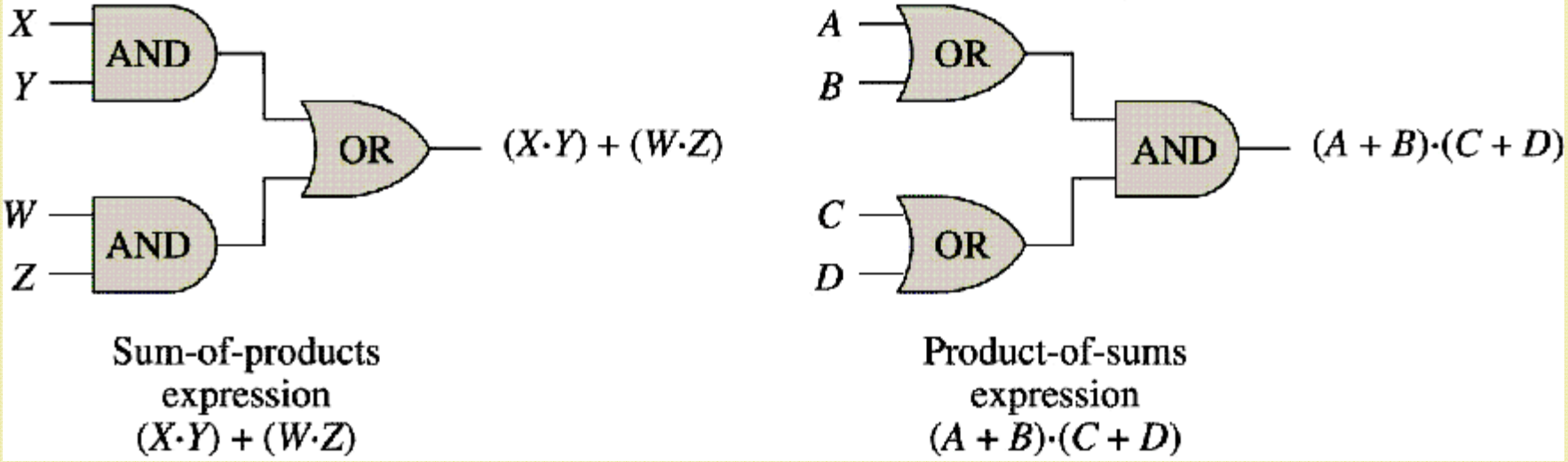
Truth table

X	Y	$Z = \overline{(X+Y)} = \bar{X} \cdot \bar{Y}$
0	0	1
0	1	0
1	0	0
1	1	0

Truth table

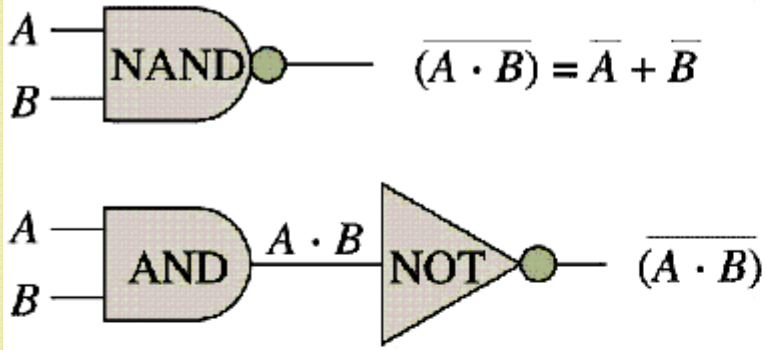
X	Y	$Z = \overline{(X+Y)} = \bar{X} + \bar{Y}$
0	0	1
0	1	1
1	0	1
1	1	0

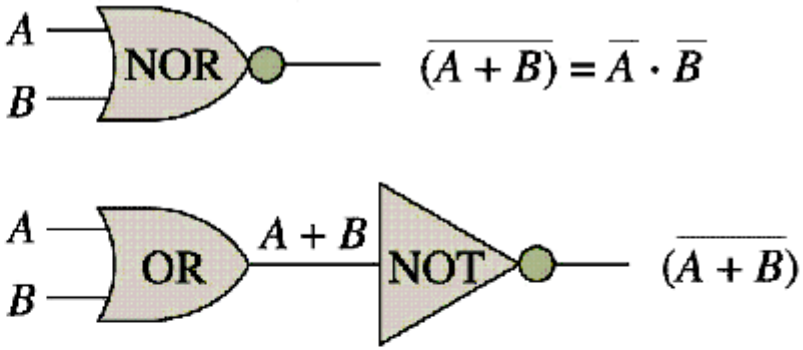
- DeMorgan's Theorems state a very important property of logic functions:
  - Any logic function can be implemented by using only OR and NOT gates, or only AND and NOT gates.
  - The importance of DeMorgan's Laws lies in the statement of the duality that exists between AND and OR operations: Any function can be realized by just one of the two basic operations, plus the compliment operation.
  - This gives rise to two families of logic functions:
    - Sums of Products
    - Products of Sums



– Any logical expression can be reduced to one of these two forms. Although the two forms are equivalent, it may well be true that one of the two forms has a simpler implementation (fewer gates).

- NAND and NOR Gates
  - In addition to the AND and OR gates, the complimentary forms of the gates, called NAND and NOR, are commonly used in practice.
  - It is important to note that, by DeMorgan's Laws, the NAND gate performs a logical addition on the compliments of the inputs, while the NOR gate performs a logical multiplication on the compliments of the inputs.
  - Functionally, then, any logic function could be implemented with either NOR or NAND gates only.





A	B	$\overline{A}$	$\overline{B}$	$\overline{(A \cdot B)}$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

NAND gate

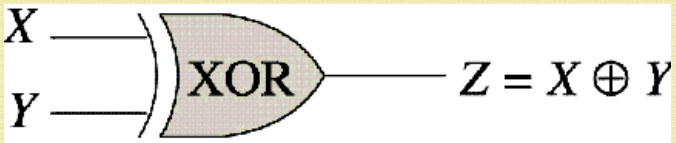
A	B	$\overline{A}$	$\overline{B}$	$\overline{(A + B)}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

NOR gate

Equivalence of NAND and NOR gates with AND and OR gates

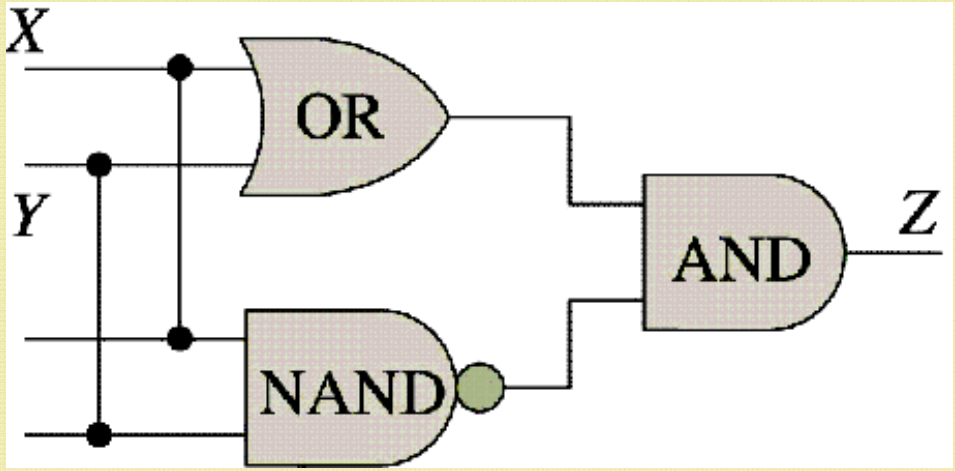
– XOR (exclusive OR) Gate

- Common combinations of logic circuits are often provided in a single integrated-circuit package. The XOR gate is an example.



X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

Truth table



Realization of an XOR Gate

- Design of Logic Networks
  - How do you apply combinational logic to a real engineering problem? Here is a sequence of steps one might follow.
    - Define the problem in words.
    - Write quasi-logic statements in English that can be translated into Boolean expressions.
    - Write the Boolean expressions.
    - Simplify and optimize the Boolean expressions, if possible.
    - Write an all-AND, all-NAND, all-OR, or all-NOR realization of the circuit to minimize the number of required logic IC gates.
    - Draw the logic schematic for the electronic realization.

- Finding a Boolean Expression Given a Truth Table
  - Rather than defining a logic problem in words and then writing quasi-logic statements, sometimes it is more convenient to express the complete input/output combinations with a truth table.
  - In these situations, there are two methods for directly obtaining the Boolean expression that performs the logic specific in the truth table.
  - Sum-of-Products Method
    - We can represent an output as a sum of products containing combinations of the inputs. If we have 3 inputs and 1 output X, the sum of the products would be the following Boolean expression: $X = (\overline{A} \bullet B \bullet C) + (\overline{A} \bullet \overline{B} \bullet C) + (A \bullet B \bullet \overline{C})$

- If we form a product for every row in the truth table that results in an output of 1 and take the sum of the products, we can represent the complete logic of the table.
- For rows whose output values are 1, we must ensure that the product representing that row is 1. In order to do this, any input whose value is 0 in the row must be inverted in the product.
- By expressing a product for every input combination whose value is 1, we have completely modeled the logic of the truth table since every other combination will result in a 0.
- Example:

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

$$X = (\bar{A} \bullet B) + (A \bullet \bar{B})$$

– Product-of-Sums Method

- This is based on the fact that we can represent an output as a product of sums containing combinations of the inputs. If we have 3 inputs and 1 output X, the product of the sums would be the following Boolean expression:

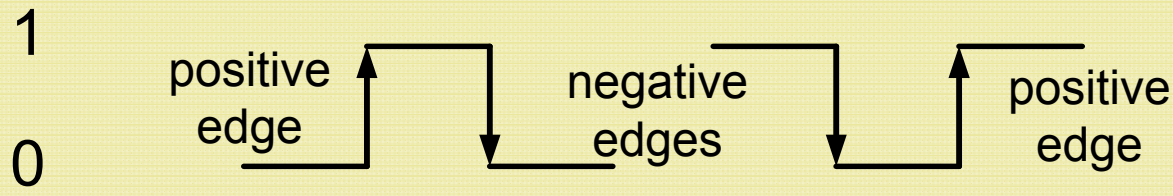
$$X = (\overline{A} + B + C) \bullet (\overline{A} + \overline{B} + C) \bullet (A + B + \overline{C})$$

- If we form a sum for every row in the truth table that results in an output of 0 and take the product of the sums, we can represent the complete logic of the table.
- For rows whose output values are 0, we must ensure that the sum representing that row is 0. In order to do this, any input whose value is 1 in the row must be inverted in the sum.
- By expressing a sum for every input combination (row) whose value is 0, we have completely modeled the logic of the truth table since every other combination will result in a 1.
- Example: For the previous truth table

$$X = (A + B) \bullet (\overline{A} + \overline{B})$$

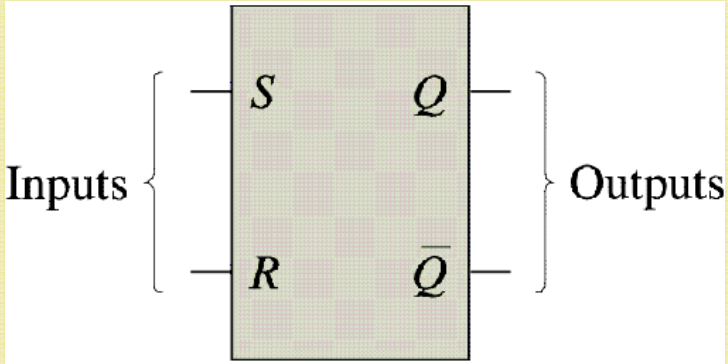
- Sequential Logic
  - Combinational logic devices generate an output based on the input values, independent of the input timing.
  - With sequential logic devices, the timing or sequencing of the input signals is important. Devices in this class include flip-flops, counters, monostables, latches, and more complex devices such as microprocessors.
  - Sequential logic devices usually respond to inputs when a separate trigger signal transitions from one level to another. The trigger signal is usually refereed to as the clock (CK) signal and can be a periodic square wave or an aperiodic collection of pulses.

– Positive edge-triggered devices respond to a low-to-high (0 to 1) transition, and negative edge-triggered devices respond to a high-to-low (1 to 0) transition.



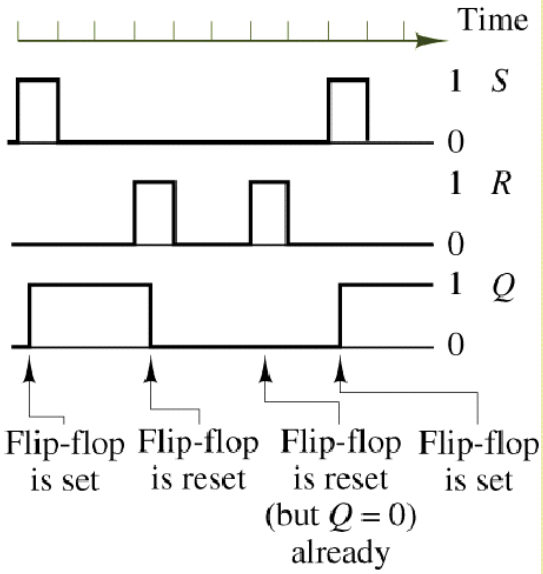
- Flip-Flops
  - A flip-flop is a sequential device that can store and switch between the two binary states.
  - It is called a bistable device since it has two and only two possible output states: 1 (high) and 0 (low).
  - It has the capability of remaining in a particular state (i.e., storing a bit) until input signals cause it to change state.
  - Let's consider a fundamental flip-flop: the RS Flip-Flop
    - S is the set input
    - R is the rest input
    - Q and $\bar{Q}$ are the complimentary outputs.

RS Flip-Flop: Symbol, Truth Table, and Timing Diagram



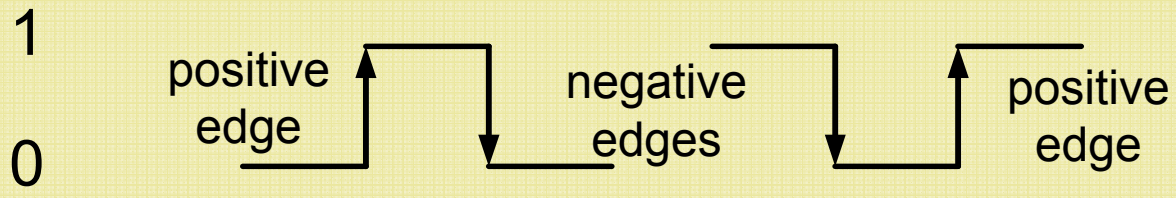
S	R	Q
0	0	Present state
0	1	Reset Q = 0
1	0	Set Q = 1
1	1	Disallowed

S	R	Q
1	0	1
0	0	1
0	0	1
0	1	0
0	0	0
0	0	0
0	1	0
0	0	0
1	0	1
0	0	1



– Triggering of Flip-Flops

- Flip-flops are usually clocked, i.e., a master signal in the circuit coordinates or synchronizes the changes of the output states of the device. This is called synchronous operation since changes in state are coordinated by the clock pulses.
- The outputs of different types of clocked flip-flops can change on either a positive edge or negative edge of a clock pulse. These flip-flops are called edge-triggered flip flops.

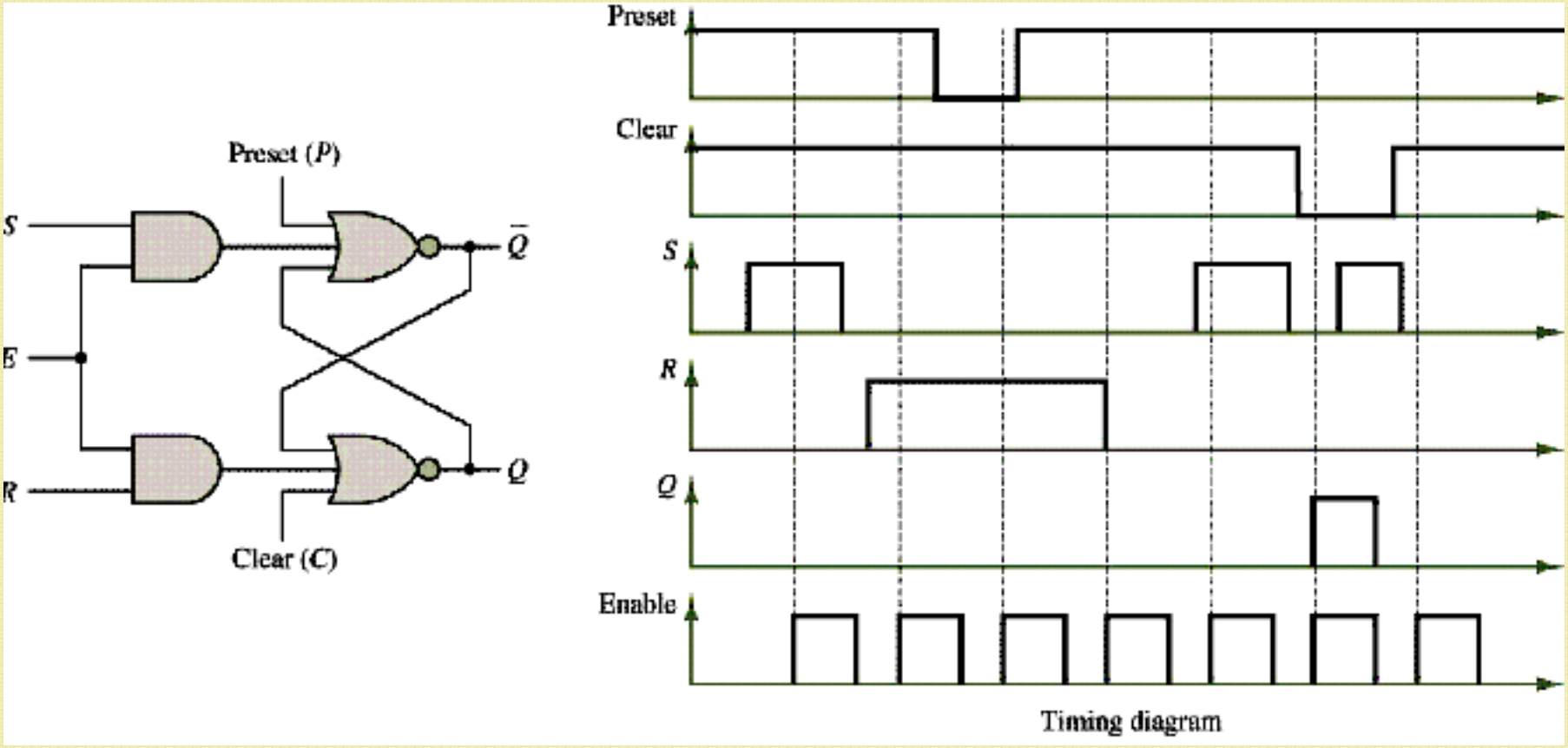


- Rules:
  - If S and R are both 0 when the clock edge is encountered, the output state remains unchanged.
  - If S = 1 and R = 0 when the clock signal is encountered, the output is set to 1. If the output is 1 already, there is no change.
  - If S = 0 and R = 1 when the clock signal is encountered, the output is reset to 0. If the output is 0 already, there is no change.

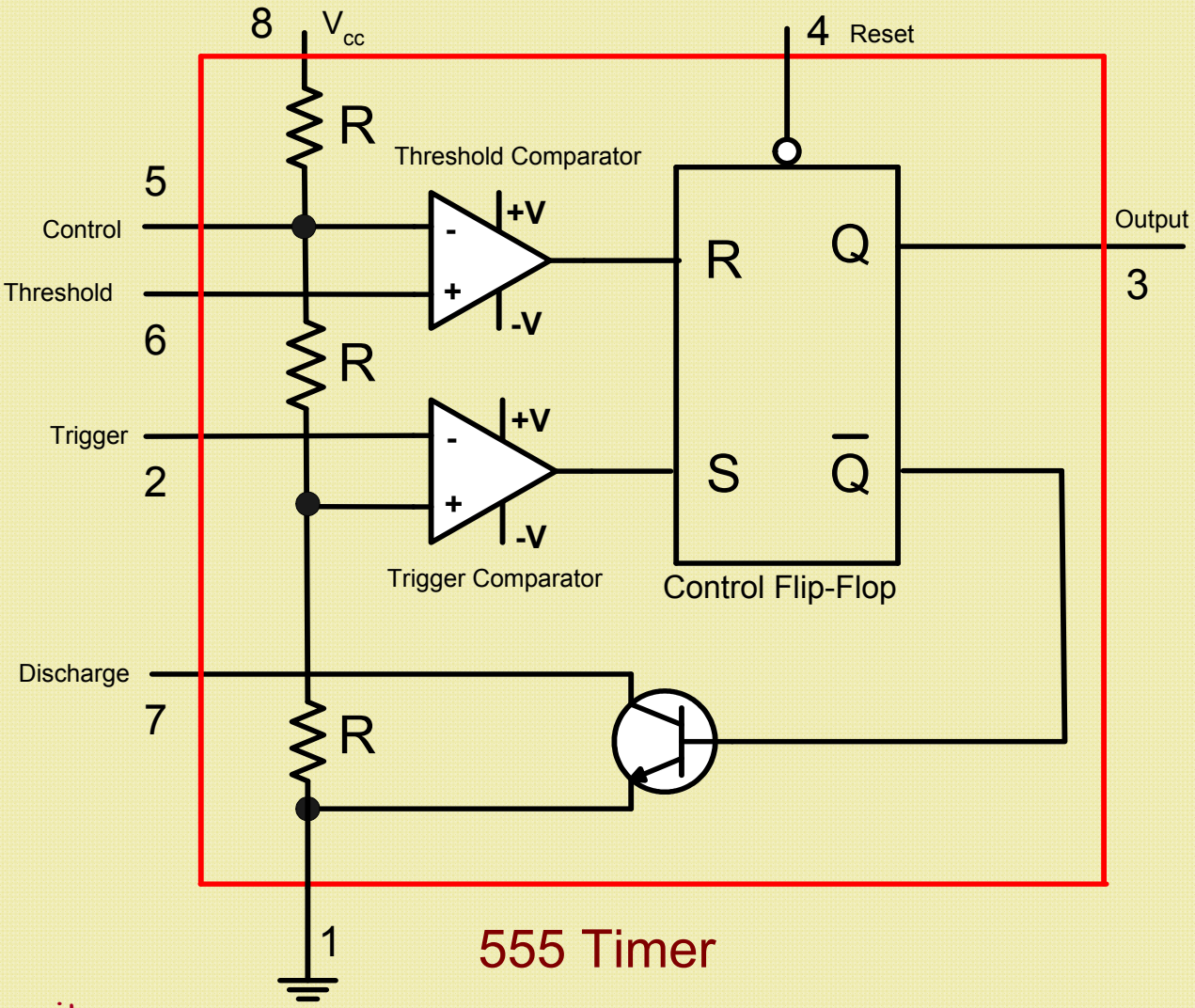
- Asynchronous Inputs
  - Flip-flops may have preset and clear functions that instantaneously override any other inputs. These are called asynchronous inputs, because their effect may be asserted at any time. They are not triggered by a clock signal.
  - The preset input is used to set or initialize the output Q of the flip-flop to 1 or high.
  - The clear input is used to clear or reset the output Q of the flip-flop to 0 or low.

- The small inversion symbol (open circle) shown at an asynchronous input implies that the function is asserted when the asynchronous input signal is low. This is referred to as an active low input.
- Both the preset and clear should not be asserted simultaneously.
- Either of these inputs can be used to define the state of a flip-flop after power-up; otherwise, at power-up the output of a flip-flop is uncertain.

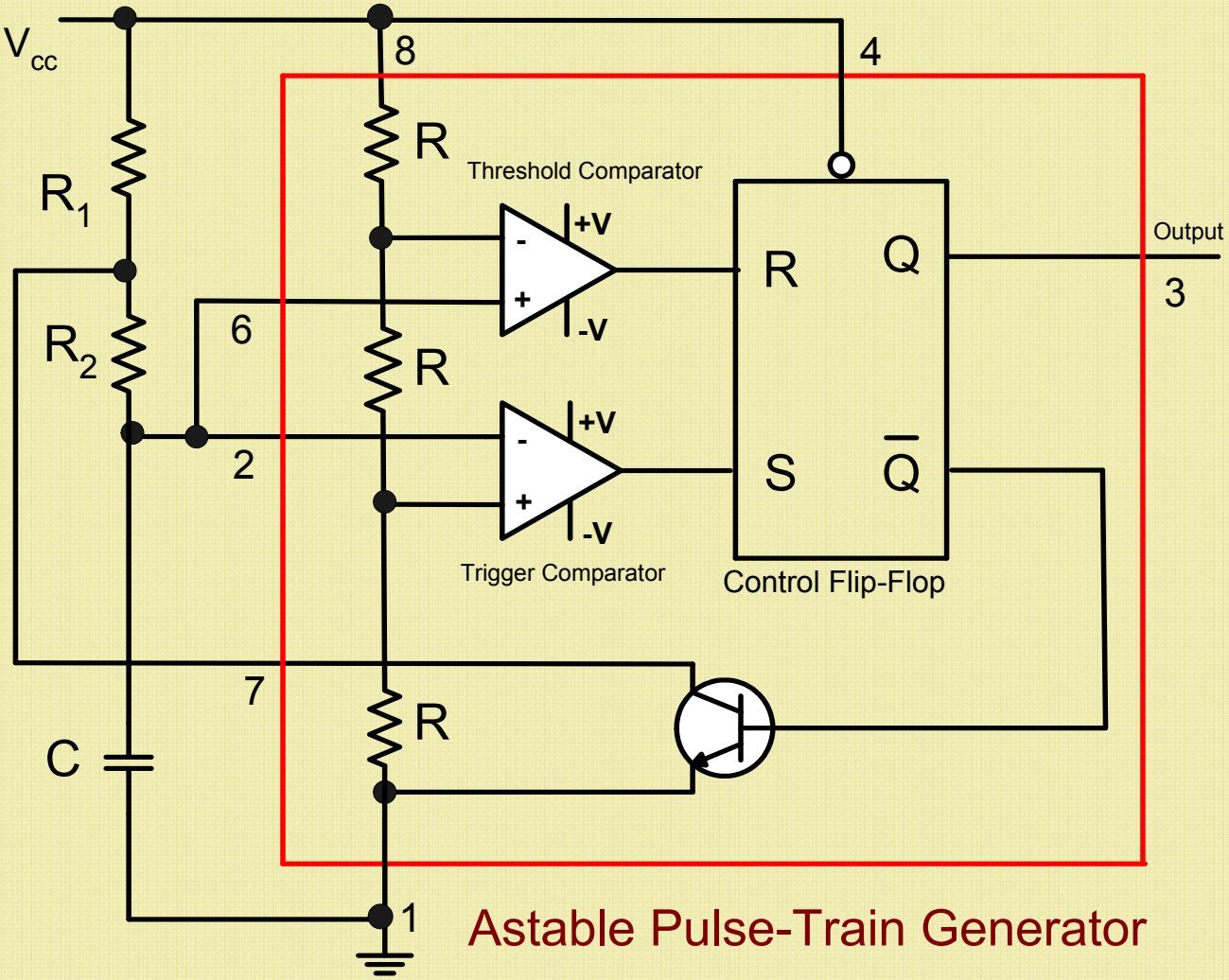
RS flip-flop with enable, preset, and clear lines: logic diagram and timing diagram

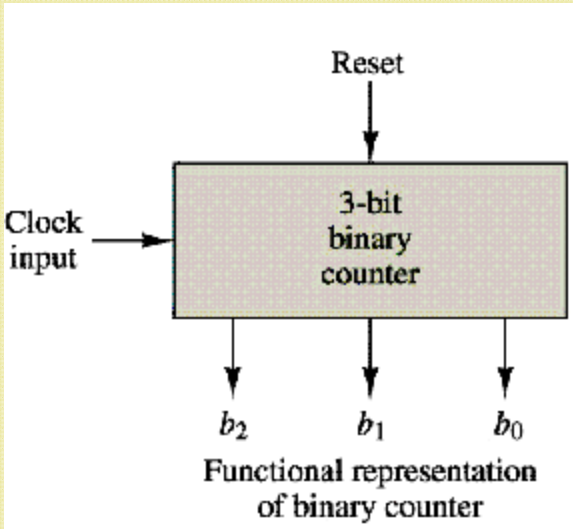


- Application of RS Flip-Flop: 555 Timer



- Astable Pulse-train Generator

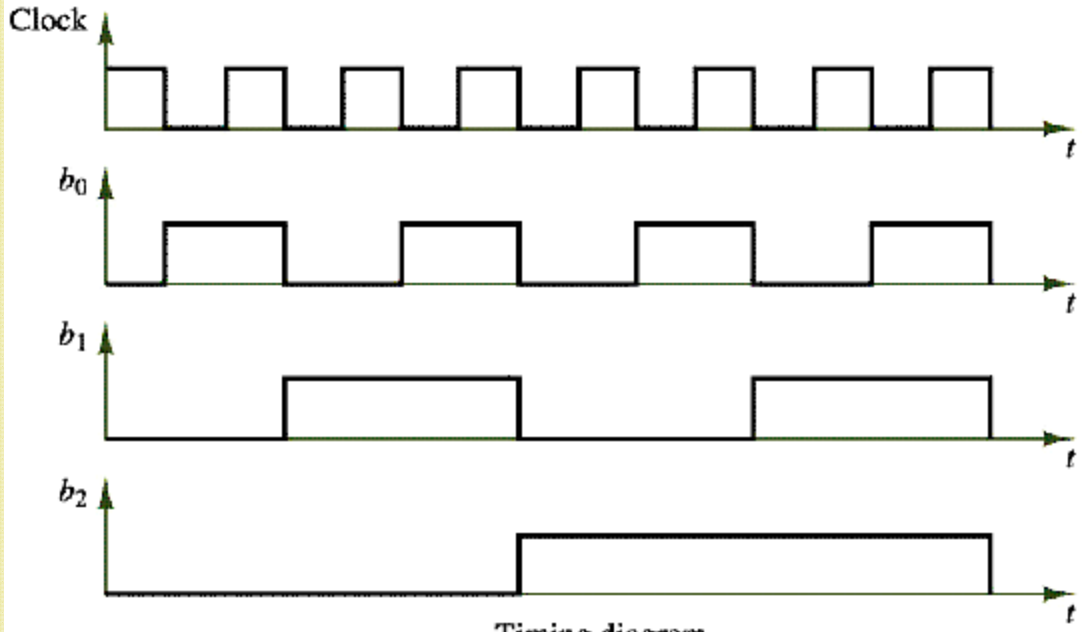




Functional representation of binary counter

Input pulses	State b_2	b_1	b_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Timing table



Timing diagram

- Karnaugh Maps and Logic Design
  - More than one solution is usually available for the implementation of a given logic expression.
  - Some combinations of gates can implement a given function more efficiently than others.
  - How can we be assured of having chosen the most efficient realization?
  - A Karnaugh Map describes all possible combinations of the variables present in the logic function of interest.
  - A Karnaugh Map consists of 2^N cells, where N is the number of logic variables. Row and column assignments are arranged so that all adjacent terms change by only one bit.

– For example:

The Karnaugh Map provides an immediate view of the values of the function in graphical form.

	$\bar{X}\bar{Y}$	$\bar{X}Y$	XY	$X\bar{Y}$
$\bar{Z}$	0	1	1	0
Z	0	1	1	0

Karnaugh map

X	Y	Z	Desired Function
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Truth table

– Let's use a four-variable logic function to explain how Karnaugh Maps can be used directly to implement a logic function.

X	Y	Y	Z	Desired Function
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Truth table for four-variable expression

	$\overline{W}\cdot\overline{X}$	$\overline{W}\cdot X$	$W\cdot X$	$W\cdot\overline{X}$
$\overline{Y}\cdot\overline{Z}$	1	0	0	0
$\overline{Y}\cdot Z$	1	1	0	1
$Y\cdot Z$	0	0	1	0
$Y\cdot\overline{Z}$	0	1	0	1