

Properties of a Boolean Algebra

1. Operations are *commutative*.

$$A \bullet B = B \bullet A$$

$$A + B = B + A$$

2. Operations are *associative*.

$$(A \bullet B) \bullet C = A \bullet (B \bullet C)$$

$$(A + B) + C = A + (B + C)$$

3. Each operation is *distributive* over the other.

$$A \bullet (B + C) = (A \bullet B) + (A \bullet C)$$

$$A + (B \bullet C) = (A + B) \bullet (A + C)$$

4. There exists an *identity* element for each operation.

$$+ \text{ Identity} = 0 \quad A + 0 = A$$

$$\bullet \text{ Identity} = 1 \quad A \bullet 1 = A$$

5. There exists a *complement* for each element.

$$\text{Complement of } A = \bar{A}$$

$$\text{Complement of } 1 = 0$$

$$\text{Complement of } 0 = 1$$

6. There exists an *inverse* for each operation.

$$A \bullet \bar{A} = 0$$

$$A + \bar{A} = 1$$

7. Each element is *idempotent*.

$$A \bullet A = A$$

$$A + A = A$$

8. The *absorption* property holds for each element.

$$A \bullet (A + B) = A$$

$$A + (A \bullet B) = A$$

Logic Circuits

Combinational or Memoryless Logic Circuits
Function of Current Input Only

Sequential or Memory Logic Circuits
Function of Current Input plus Past Inputs

State Table (Outputs & Next State)
Next State = Present State + Current Input

Moore & Mealy Machines

Moore Machine
Outputs Function of State Only

Mealy Machine
Outputs Function of State and Input

Sequential Machines

Synchronous Sequential Machines
Defined only at discrete times
Controlled by external clock
Uses Flip-Flops to hold
state variables between clock pulses

Asynchronous Sequential Machines
Defined for all times
No need for explicit memory
Simpler - Two Implementation Restrictions

Asynchronous Machines

No more than one input variable may change at any one time. State variable must be assigned in such a way that no more than one state variable changes for any possible state changes.

"Simultaneous" Signal Changes

Finite Propagation Times

If $00 > 11$ may happen in several ways

$00 > 01 > 11$ or

$00 > 10 > 11$

Depends on "who wins the race"

May have "Don't Care" States

Mealy & Moore Machines

Moore Machine is a finite-state machine whose output values are determined solely by its current state and can be defined as six elements $(S, S_0, \Sigma, \Lambda, T, G)$, consisting of the following:

a finite set of states (S)

a start state (also called initial state) S_0 which is an element of (S)

a finite set called the input alphabet (Σ)

a finite set called the output alphabet (Λ)

a transition function $(T : S \times \Sigma \rightarrow S)$ mapping a state and the input alphabet to the next state

an output function $(G : S \rightarrow \Lambda)$ mapping each state to the output alphabet.

Mealy Machine output values are determined both by its current state and by the values of its inputs and can be defined as six elements $(S, S_0, \Sigma, \Lambda, T, G)$, consisting of the following:

a finite set of states (S)

a start state (also called initial state) S_0 which is an element of (S)

a finite set called the input alphabet (Σ)

a finite set called the output alphabet (Λ)

a transition function $(T : S \times \Sigma \rightarrow S)$ mapping a state and the input alphabet to the next state

an output function $(G : S \times \Sigma \rightarrow \Lambda)$ mapping pairs of a state and an input symbol to the corresponding output symbol.

http://en.wikipedia.org/wiki/Theory_of_Computation

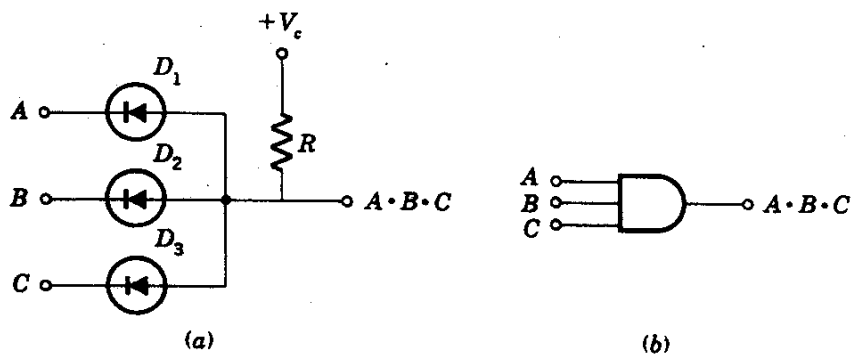


Figure 9-2 (a) Diode AND gate and (b) circuit symbol.

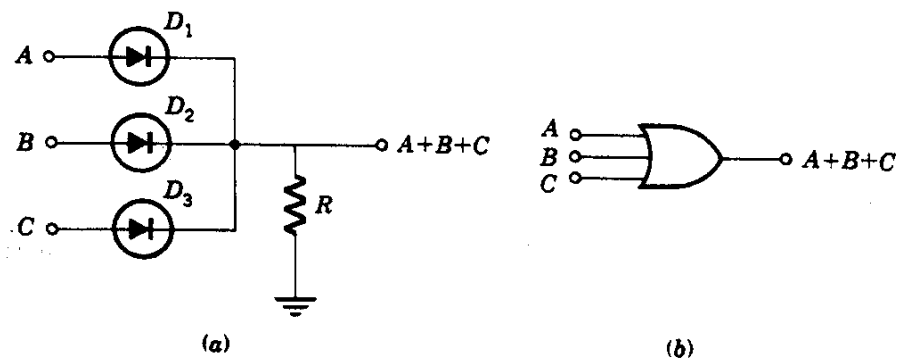


Figure 9-3 (a) Diode OR gate and (b) circuit symbol.

TRANSISTOR SWITCH

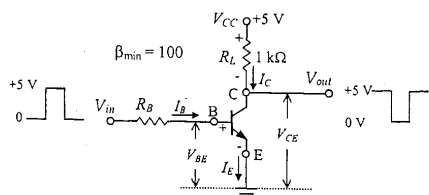


Figure 3-32: The BJT Inverter

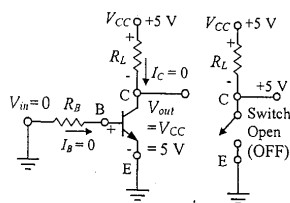


Figure 3-32(a): Transistor OFF

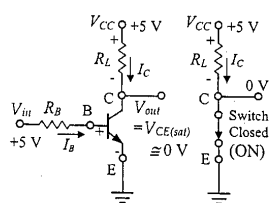
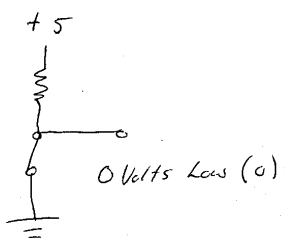
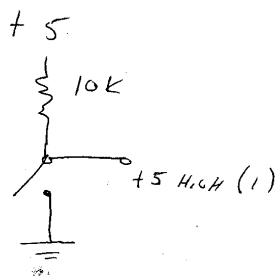


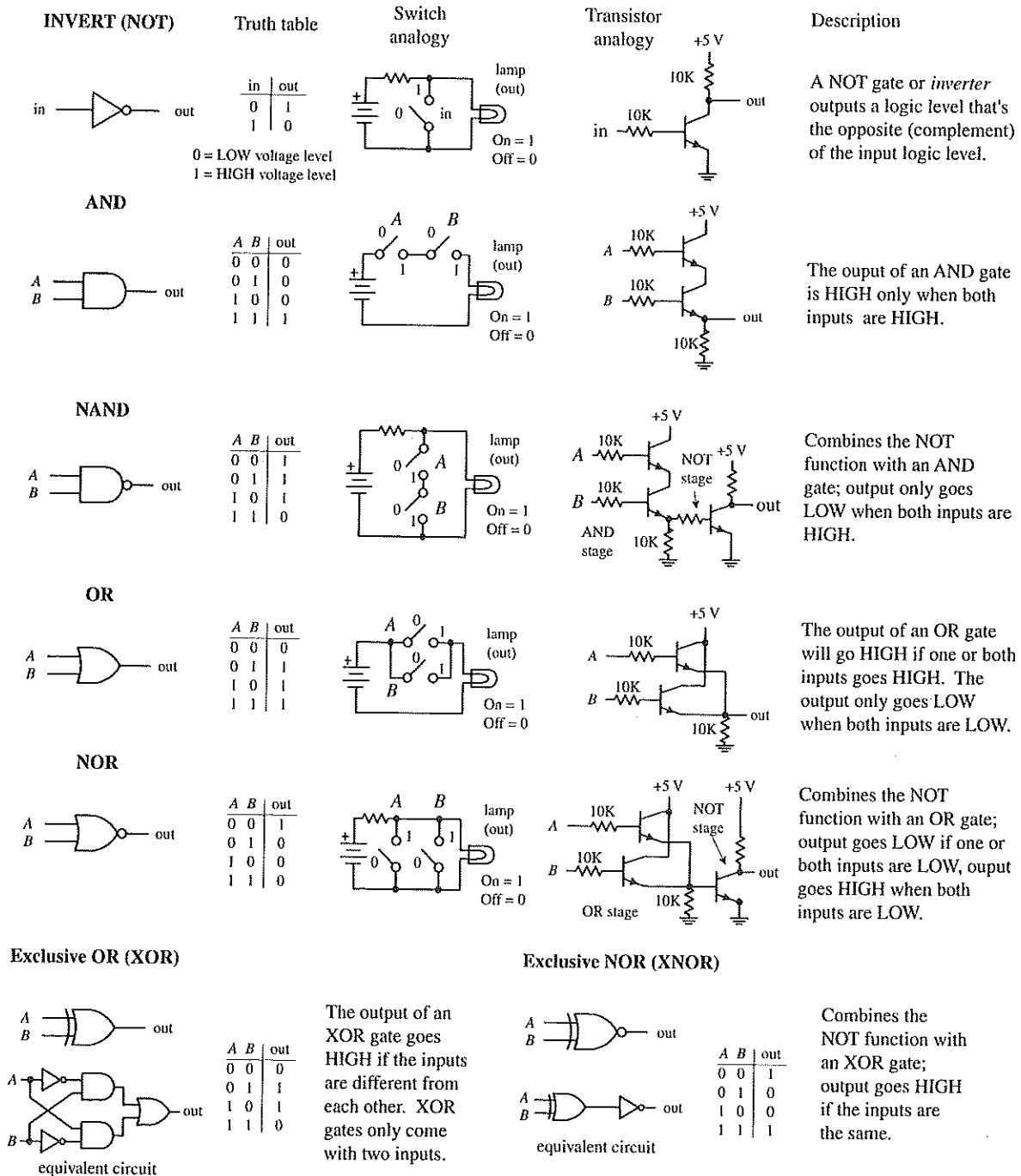
Figure 3-32(b): Transistor ON



12.2 Logic Gates

Logic gates are the building blocks of digital electronics. The fundamental logic gates include the INVERT (NOT), AND, NAND, OR, NOR, exclusive OR (XOR), and exclusive NOR (XNOR) gates. Each of these gates performs a different logical operation. Figure 12.10 provides a description of what each logic gate does and gives a switch and transistor analogy for each gate.

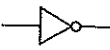

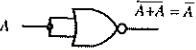

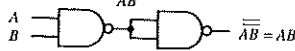

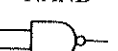
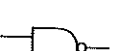




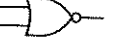

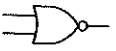
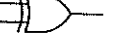

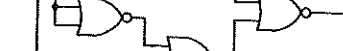
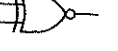

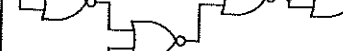
FIGURE 12.10



Universal Capability of NAND and NOR Gates

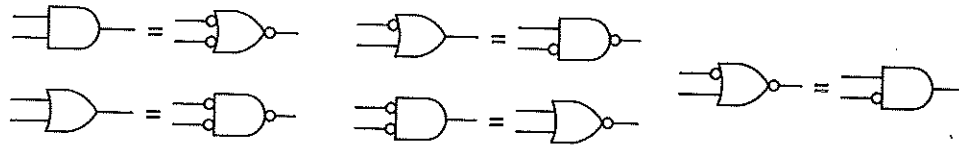
NAND and NOR gates are referred to as *universal gates* because each alone can be combined together with itself to form all other possible logic gates. The ability to create any logic gate from NAND or NOR gates is obviously a handy feature. For example, if you do not have an XOR IC handy, you can use a single multigate NAND gate (e.g., 74HC00) instead. The figure below shows how to wire NAND or NOR gates together to create equivalent circuits of the various logic gates.

FIGURE 12.24

Logic gate	NAND equivalent circuit	NOR equivalent circuit
	 $\overline{AA} = \bar{A}$	 $\overline{A+A} = \bar{A}$
	 $\overline{\overline{AB}} = AB$	 $\overline{\overline{A+B}} = AB$
		 $\overline{A+B}$
	 $\overline{\overline{A} \overline{B}} = A+B$	 $\overline{\overline{A+B}} = A+B$
	 $\overline{\overline{A} \overline{B}} = A+B$	
		
		

Bubble Pushing

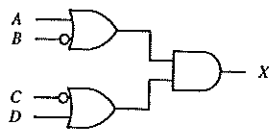
A shortcut method for forming equivalent logic circuits, based on De Morgan's theorem, is to use what's called *bubble pushing*.



AND-OR-INVERT Gates (AOIs)

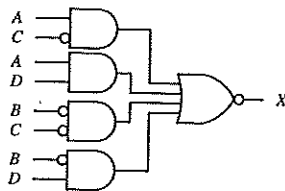
When a Boolean expression is reduced, the equation that is left over typically will be of one of the following two forms: *product-of-sums* (POS) or *sum-of-products* (SOP). A POS expression appears as two or more ORed variables ANDed together with two or more additional ORed variables. An SOP expression appears as two or more ANDed variables ORed together with additional ANDed variables. The figure below shows two circuits that provide the same logic function (they are equivalent), but the circuit to the left is designed to yield a POS expression, while the circuit to the right is designed to yield a SOP expression.

Logic circuit for POS expression



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

Logic circuit for SOP expression



$$X = A\bar{C} + AD + \bar{B}C + \bar{B}D$$

Table made using SOP expression
(it's easier than POS)

A	B	C	D	$A\bar{C}$	AD	$\bar{B}C$	$\bar{B}D$	X
0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	1	1	1
0	0	1	0	0	0	0	0	0
0	0	1	1	0	0	0	1	1
0	1	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0
1	0	0	0	1	0	1	0	1
1	0	0	1	1	1	1	1	1
1	0	1	0	0	0	0	0	0
1	0	1	1	0	1	0	1	1
1	1	0	0	1	0	0	0	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	0	0	0
1	1	1	1	0	1	0	0	1

FIGURE 12.25

Bubble pushing involves the following tricks: First, change an AND gate to an OR gate or change an OR gate to an AND gate. Second, add inversion bubbles to the inputs and outputs where there were none, while removing the original bubbles. That's it. You can prove to yourself that this works by examining the corresponding truth tables for the original gate and the bubble-pushed gate, or you can work out the Boolean expressions using De Morgan's theorem. Figure 12.23 shows examples of bubble pushing.

LOGIC IDENTITIES

- 1) $A + B = B + A$
- 2) $AB = BA$
- 3) $A + (B + C) = (A + B) + C$
- 4) $A(BC) = (AB)C$
- 5) $A(B + C) = AB + AC$
- 6) $(A + B)(C + D) = AC + AD + BC + BD$
- 7) $\bar{1} = 0$
- 8) $\bar{0} = 1$
- 9) $A \cdot 0 = 0$
- 10) $A \cdot 1 = A$
- 11) $A + 0 = A$
- 12) $A + 1 = 1$
- 13) $A + A = A$
- 14) $\overline{\overline{A}} = A$
- 15) $\overline{\overline{A}} = A$
- 16) $A + \bar{A} = 1$
- 17) $\overline{\overline{A}} = 0$
- 18) $\overline{A + B} = \bar{A}\bar{B}$
- 19) $\overline{AB} = \bar{A} + \bar{B}$
- 20) $A + \bar{A}B = A + B$
- 21) $\bar{A} + AB = \bar{A} + B$
- 22) $A \oplus B = \bar{A}B + A\bar{B} = (A + B)(\bar{A}\bar{B})$
- 23) $A \odot B = AB + \bar{A}\bar{B}$

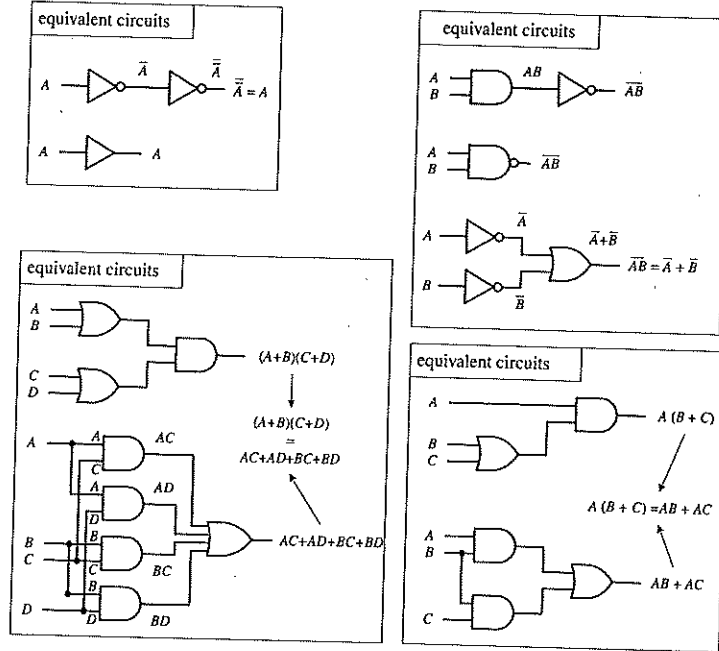
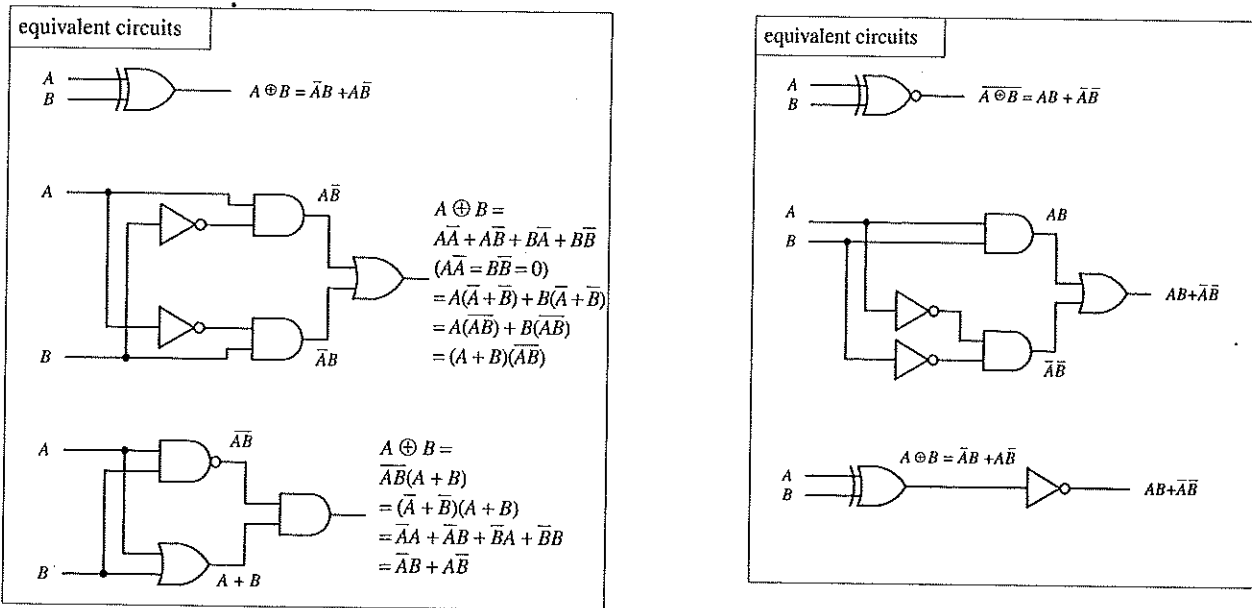


FIGURE 12.19

FIGURE 12.21



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

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

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

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

FIGURE 12.22

Digital Logic Signal Levels and State Variables

Simple Positive Logic

Define "Lo" = State "0" = 0; i.e., "near 0 volts, or maybe +0.7V, or less than +2.1V, etc."

Define "Hi" = State "1" = 1; i.e., "near Vcc,

say +5V, or greater than +3.9V, etc. for TTL;
or +15V, or greater than +13.1V, etc. for CMOS."

Remember these are arbitrary definitions.

Notice however, that State "1" is more positive than State "0". With this in mind, we can even define

"Hi" = State "1" = 1 = 0 volts, and

"Lo" = State "0" = 0 = -5 volts.

We still have State "1" more positive than State "0".

And Boolean Algebra doesn't care!

Simple Negative Logic

Try reversing things, such that State "1" is more negative than State "0"; i.e.,

State "1" = 0 volts, and

State "0" = +5 volts, or even

State "1" = -5 volts, and

State "0" = 0 volts.

In both cases, State "1" is more negative than State "0".

Positive Logic Truth Tables

A	B	AND	OR
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Negative Logic Truth Tables

\bar{A}	\bar{B}	OR	AND
1	1	1	1
1	0	1	0
0	1	1	0
0	0	0	0

Note: **Positive AND Logic = Negative $\overline{\text{OR}}$ Logic**

Positive OR Logic = Negative $\overline{\text{AND}}$ Logic

DeMorgan's Law

$$\overline{A \bullet B} = \bar{A} + \bar{B}$$

$$\overline{A + B} = \bar{A} \bullet \bar{B}$$

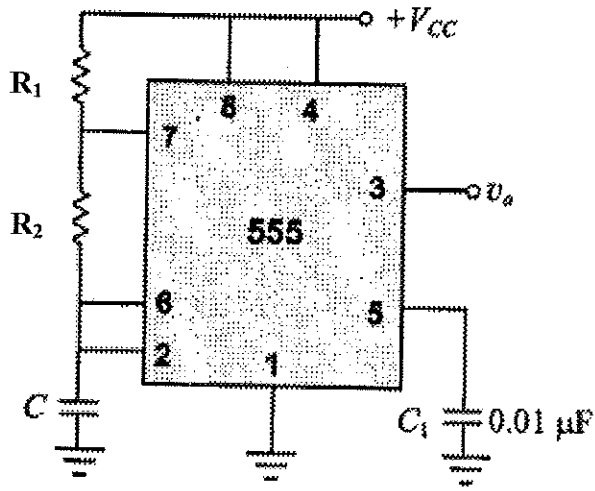
$$\overline{\overline{A \bullet B}} = \overline{\bar{A} + \bar{B}} = \overline{\bar{A}} \bullet \overline{\bar{B}} = A \bullet B$$

$$\overline{\overline{A + B}} = \overline{\bar{A} \bullet \bar{B}} = \overline{\bar{A}} + \overline{\bar{B}} = A + B$$

Positive Logic $A \bullet B =$ Negative Logic $\overline{\overline{A + B}}$

Positive Logic $A + B =$ Negative Logic $\overline{\overline{A \bullet B}}$

555 Astable Multivibrator Characteristics



The following computational formulas apply to the 555 configuration shown above.

$$\text{On-Time} = t_h = 0.69 (R_1 + R_2) C$$

$$\text{Off-Time} = t_l = 0.69 (R_2) C$$

$$\text{Period} = t_l + t_h = 0.69 (R_1 + 2R_2) C$$

$$\text{Frequency} = 1 / \text{Period} = 1.44 / (R_1 + 2R_2) C$$

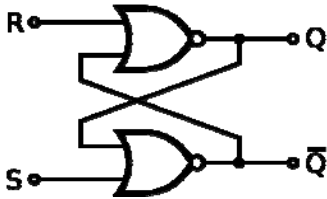
$$\text{Duty Cycle} = t_h / (t_l + t_h) = (R_1 + R_2) / (R_1 + 2R_2)$$

RS Flip Flop Truth Tables

In order to eliminate ambiguity and to achieve some sense of continuity, we will follow the convention:

Set implies $Q = 1$.

Reset implies $Q = 0$.



Set Reset

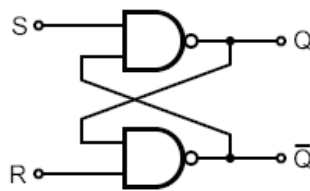
NOR Gates

S	R	Q
0	0	Q Hold
0	1	0 Reset
1	0	1 Set
1	1	X

X = Not Allowed

S=1 => Set (Q=1)

R=1 => Reset (Q=0)



Set Reset

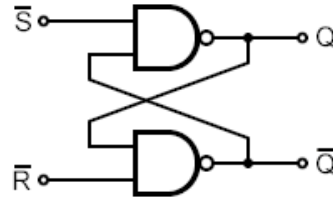
NAND Gates

S	R	Q
0	0	X
0	1	1 Set
1	0	0 Reset
1	1	Q Hold

X = Not Allowed

S=0 => Set (Q=1)

R=0 => Reset (Q=0)



Set Reset

NAND Gates with Inverted S & R inputs

S	R	S-bar	R-bar	Q
0	0	1	1	Q Hold
0	1	1	0	0 Reset
1	0	0	1	1 Set
1	1	0	0	X

X = Not Allowed

S=1 => S-bar=0 => Set (Q=1)

R=1 => R-bar=0 => Reset (Q=0)

As you can see, there is consistency for **Set** means $Q=1$ and **Reset** means $Q=0$; but there can be confusion trying to decide whether-or-not S & R are 0 or 1 depending on the type of gates (NOR or NAND).

If inverted S & R inputs are used with the NAND gates, then S=1 is the Set input and R=1 is the Reset input; which is the same as the NOR gates implementation.

Rectangular Wave & Square Wave Generators (Op-Amp Schmitt Triggers & 555 Timers)

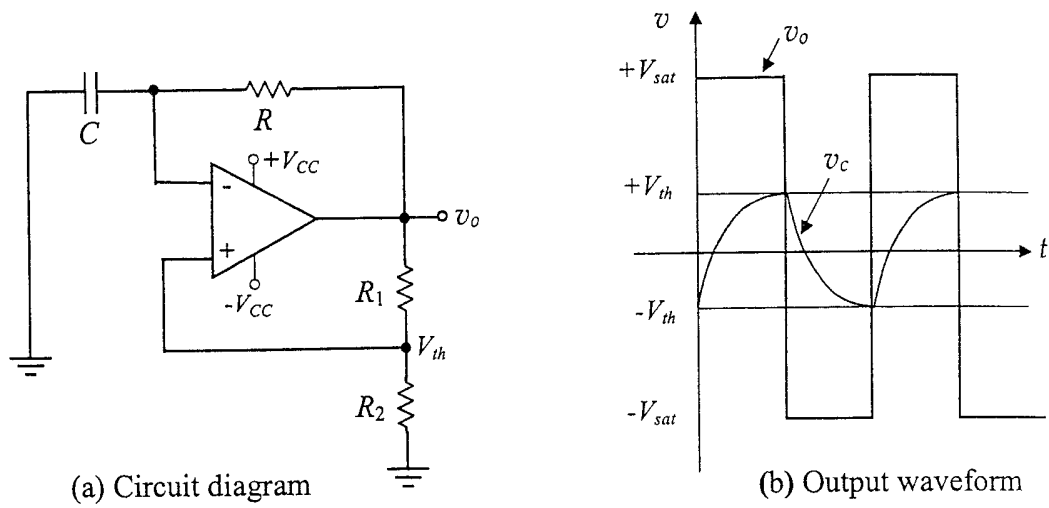


Figure 16-15: Square-wave generator

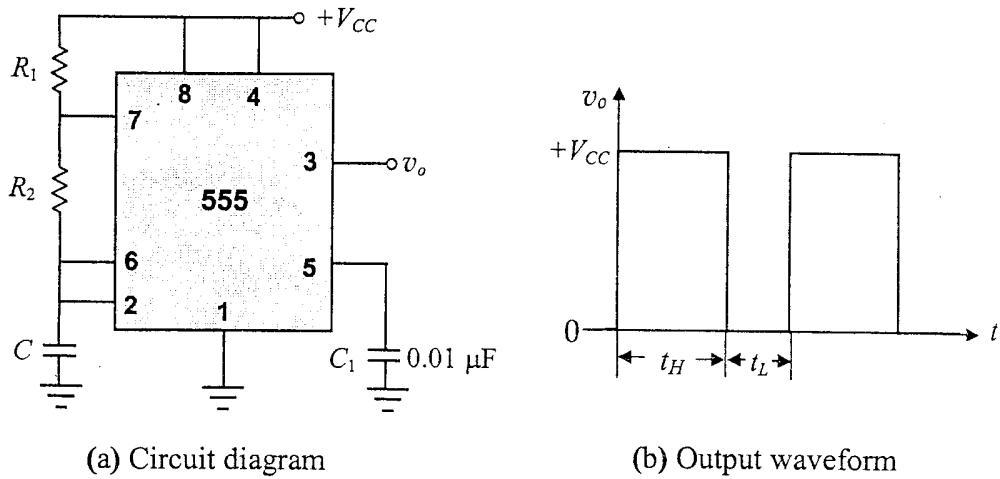


Figure 16-17: The 555 timer connected as a rectangular waveform generator

SQUARE-WAVE GENERATOR

Recall that the output of the Schmitt trigger, which was introduced in Chapter 11 as a bi-reference level comparator, is a square wave with $\pm v_{o(p)} = \pm V_{sat}$ of the op-amp. With the addition of a capacitor C and a feedback resistor R , as shown in Figure 16-15(a), the need for an input signal is eliminated and the output frequency can also be controlled by proper selection of the R and C .

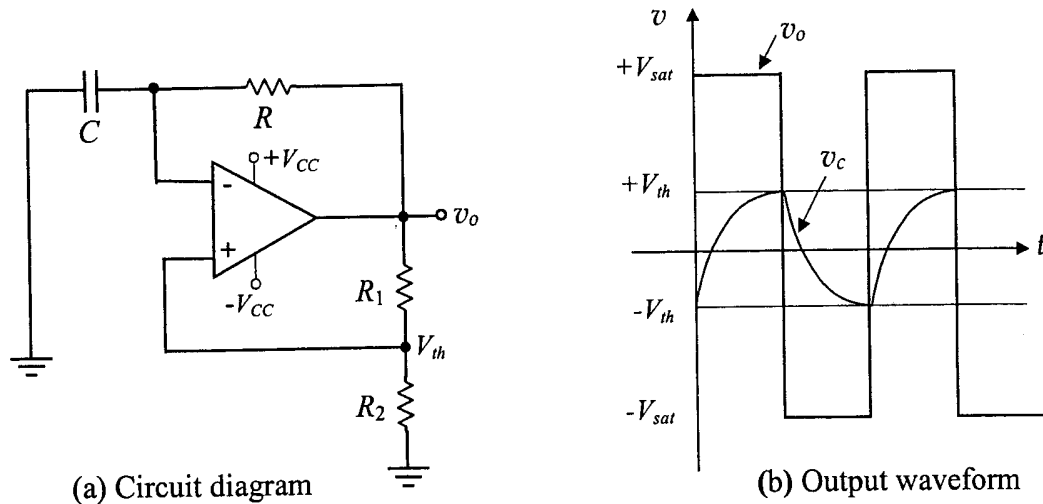


Figure 16-15: Square-wave generator

Referring to Equations 11-7 and 11-8, the upper and lower threshold voltages (V_{UT} & V_{LT}) or ($\pm V_{th}$) can be written in one equation as follows:

$$\pm V_{th} = \pm V_{sat} \frac{R_2}{R_1 + R_2} \quad (16-64)$$

It can be shown, with some considerable algebraic effort, that the period of the output waveform is as follows:

$$T = 2RC \ln \left(\frac{2R_2}{R_1} + 1 \right) \quad (16-65)$$

$$f_o = \frac{1}{T} = \frac{1}{2RC \ln(2R_2 / R_1 + 1)} \quad (16-66)$$

However, if we select R_1 and R_2 such that $(1 + 2R_2/R_1) = 2.178$ (the natural log base), then $\ln(1 + 2R_2/R_1)$ will equal unity.

$$\frac{2R_2}{R_1} + 1 = 2.178 \quad (16-67)$$

$$2R_2 = 1.178R_1 \quad (16-68)$$

$$R_2 = 0.589R_1 \quad (16-69)$$

Hence, the output frequency is a function of R and C only, and its equation simplifies as follows:

$$f_o = \frac{1}{2RC} \quad (16-70)$$

16.8 THE 555 TIMER

The 555 timer is a popular 8-pin integrated circuit (IC), which may be used in many applications including rectangular waveform generation. Figure 16-17 shows the common configuration of the 555 timer as it is connected to produce a rectangular waveform.

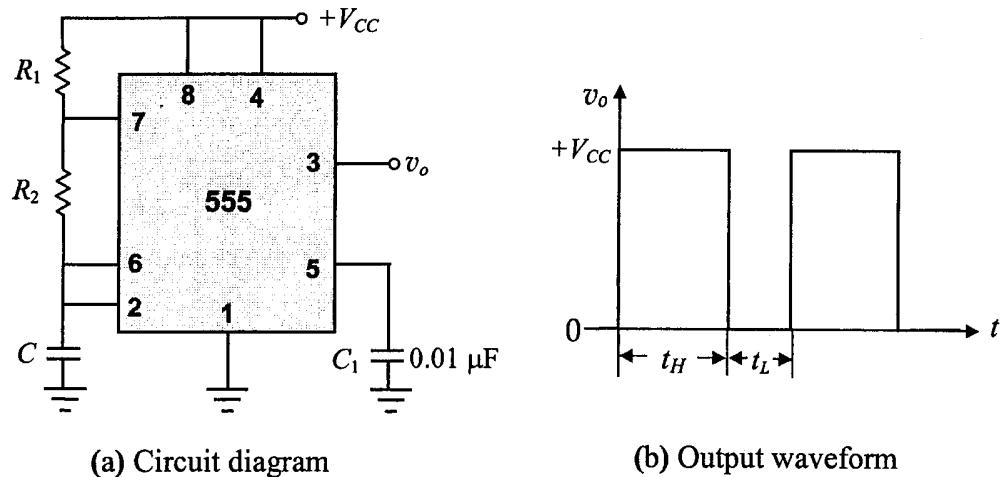


Figure 16-17: The 555 timer connected as a rectangular waveform generator

The time duration for which the output is high (t_H) is given by the following equation:

$$t_H = 0.69(R_1 + R_2)C \quad (16-71)$$

The time duration for which the output is low (t_L) is given by the following equation:

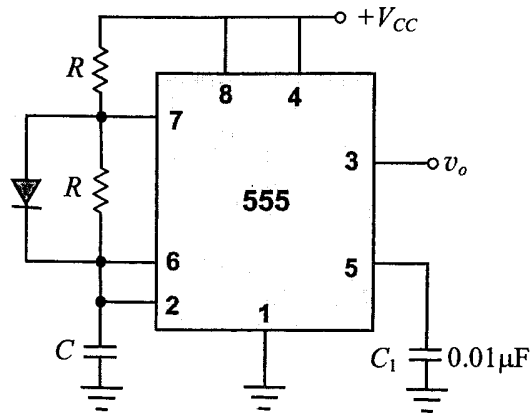
$$t_L = 0.69(R_2)C \quad (16-72)$$

Therefore, the period and frequency of the waveform are as follows:

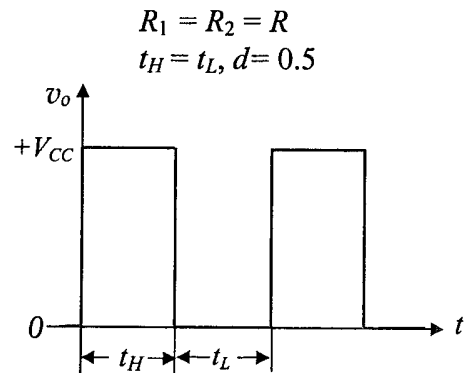
$$T = t_H + t_L = 0.69(R_1 + 2R_2)C \quad (16-73)$$

$$f_o = \frac{1}{T} = \frac{1}{0.69(R_1 + 2R_2)C} \quad (16-74)$$

For a rectangular waveform, the ratio of the pulse duration (t_H) to the period T is referred to as the *duty cycle* (d) of the waveform. A square wave is a rectangular waveform with $d = 0.5$ or 50% duty cycle. Examining the equations for t_H and t_L , we notice that it would not be possible to produce a square wave with the circuit of Figure 16-16. However, there is a simple solution for this problem, and that is to connect a diode across the R_2 and let $R_1 = R_2 = R$, as illustrated in Figure 16-18(a).



(a) Circuit diagram



(b) Output waveform

Figure 16-18: The 555 timer connected as a square-wave generator

When the output is high, the diode is forward-biased, shorting out R_2 ; hence,

$$t_H = 0.69(R_1)C = 0.69RC \quad (16-75)$$

When the output is low, the diode is unbiased, behaving like an open-circuit; hence,

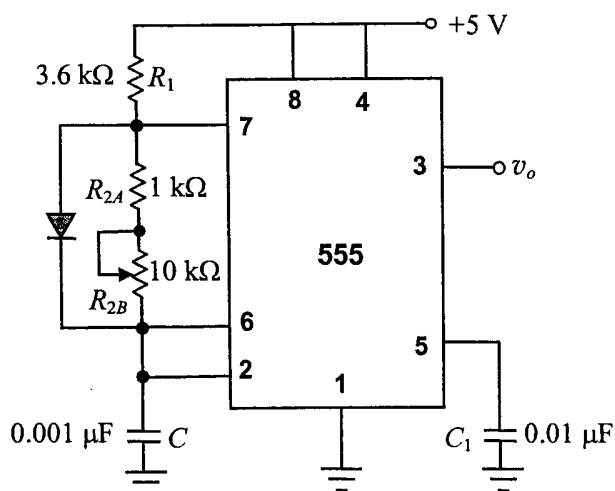
$$t_L = 0.69(R_2)C = 0.69RC \quad (16-76)$$

$$T = t_H + t_L = 0.69RC + 0.69RC = 1.38RC \quad (16-77)$$

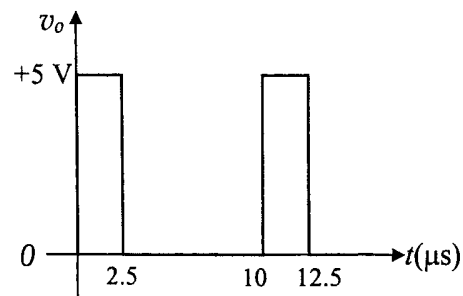
$$f_o = \frac{1}{T} = \frac{1}{1.38RC} \quad (16-78)$$

$$d = \frac{t_H}{T} = \frac{0.69RC}{1.38RC} = 0.5 \quad (16-79)$$

In order to produce a rectangular waveform with a duty cycle less than 50% ($t_H < t_L$), we can pick R_2 larger than R_1 , as required. However, the practical solution is to split R_2 into a series combination of a fixed resistor and a potentiometer, so that R_2 can be adjusted for a desired duty cycle.



(a) Circuit diagram



(b) Output waveform

Figure 16-19: Rectangular waveform generator of Design Example 16-6

NAND Gate Equivalent Circuits

Two-input (X & Y) with one-output (F) logic gate circuits can be defined by a total of 16 functions.

X	Y	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16
0	0	0	1	0	0	0	1	1	1	0	0	0	1	1	1	0	1
0	1	0	0	1	0	0	1	0	0	1	1	0	1	1	0	1	1
1	0	0	0	0	1	0	0	1	0	1	0	1	1	0	1	1	1
1	1	0	0	0	0	1	0	0	1	0	1	1	0	1	1	1	1

Six of the functions can be represented merely as hard-wired configurations (inverters as required):
F1, F16, F11, F10, F6, F7.

X	Y	F1	F16	F11	F10	F6	F7
0	0	0	1	0	0	1	1
0	1	0	1	0	1	1	0
1	0	0	1	1	0	0	1
1	1	0	1	1	1	0	0
		0	1	X	Y	\bar{X}	\bar{Y}

Eight other functions can be implemented using only one NAND gate (inverters as required):
F2, F3, F4, F5, F15, F14, F13, F12

X	Y	F2	F3	F4	F5	F15	F14	F13	F12
0	0	1	0	0	0	0	1	1	1
0	1	0	1	0	0	1	0	1	1
1	0	0	0	1	0	1	1	0	1
1	1	0	0	0	1	1	1	1	0
		$\bar{X}\bar{Y}$	$\bar{X}Y$	$X\bar{Y}$	XY	NOT ($\bar{X}\bar{Y}$)	NOT ($\bar{X}Y$)	NOT ($X\bar{Y}$)	NOT (XY)

The remaining two functions can be implemented using three NAND gates (inverters as required):
F8, F9.

X	Y	F8	F9
0	0	1	0
0	1	0	1
1	0	0	1
1	1	1	0
		$\bar{X}\bar{Y} + XY$	$\bar{X}Y + X\bar{Y}$

Notes: A OR B = NAND ($\bar{A}\bar{B}$)

F6 = NOT F11 = \bar{X}

F7 = NOT F10 = \bar{Y}

F15 = NOT F2 = NAND ($\bar{X}\bar{Y}$)

F14 = NOT F3 = NAND ($\bar{X}Y$)

F13 = NOT F4 = NAND ($X\bar{Y}$)

F12 = NOT F5 = NAND (XY)

F8 = $\bar{X}\bar{Y} + XY$ = NAND [NAND ($\bar{X}\bar{Y}$) with NAND (XY)]

F9 = $\bar{X}Y + X\bar{Y}$ = NAND [NAND ($\bar{X}Y$) with NAND ($X\bar{Y}$)]

F8 = NOT F9

NAND Gate Equivalent Circuits

Sketch an equivalent circuit using only NAND gates to represent each of 16 different possible output functions from a combination of 2 inputs (X & Y).

Note:

Indicate inversion with either input bubbles or output bubbles; do not use additional inverter gates.

X	Y	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16
0	0	0	1	0	0	0	1	1	1	0	0	0	1	1	1	0	1
0	1	0	0	1	0	0	1	0	0	1	1	0	1	1	0	1	1
1	0	0	0	0	1	0	0	1	0	1	0	1	1	0	1	1	1
1	1	0	0	0	0	1	0	0	1	0	1	1	0	1	1	1	1

Example: See Function F8 above, use positive logic when the Output Function = 1

$$X = 0, Y = 0 \quad \bar{X} \text{ AND } \bar{Y} \quad \text{or}$$

$$X = 1, Y = 1 \quad X \text{ AND } Y$$

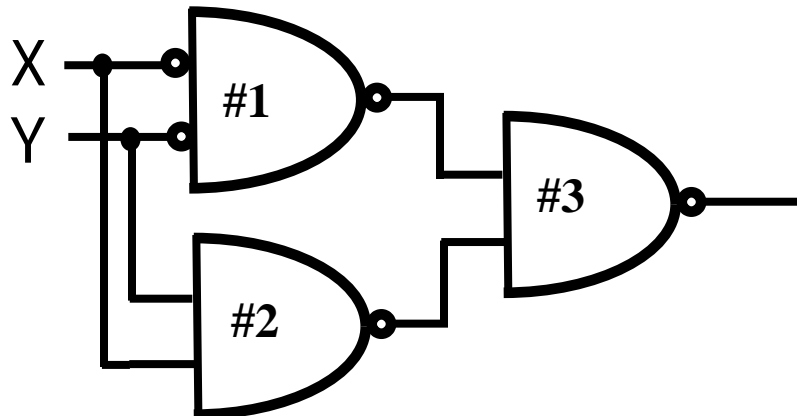
$$F8 = (\bar{X} \text{ AND } \bar{Y}) \text{ OR } (X \text{ AND } Y)$$

$$F8 = \overline{\overline{(\bar{X} \text{ AND } \bar{Y}) \text{ OR } (X \text{ AND } Y)}}$$

$$F8 = \overline{\overline{(\bar{X} \text{ AND } \bar{Y})} \text{ AND } \overline{\overline{(X \text{ AND } Y)}}} = \text{NAND}(\bar{X} \text{ with } \bar{Y}) \text{ AND } \text{NAND}(X \text{ with } Y)$$

$$F8 = \text{NAND}(\text{NAND}(\bar{X} \text{ with } \bar{Y}) \text{ with } \text{NAND}(X \text{ with } Y))$$

Sketch:



Logic Table Proof:

X	Y	\bar{X}	\bar{Y}	Gate #1		Gate #2		Gate #3	
				\bar{X}	NAND \bar{Y}	X	NAND Y	Gate#1	NAND Gate #2
0	0	1	1	0	1	1			
0	1	1	0	1	1	0			
1	0	0	1	1	1	0			
1	1	0	0	1	0	1			

See page 2 for a similar case using negative logic when the Output Function F8 = 0.

Negative logic when the Output Function $F8 = 0$

$$X = 0, Y = 1 \quad \overline{X} \text{ AND } Y \quad \text{or}$$

$$X = 1, Y = 1 \quad X \text{ AND } \overline{Y}$$

Note $F8 = 0$, is the same as $\overline{F8}$

$$\text{so } \overline{F8} = (\overline{X} \text{ AND } Y) \text{ OR } (X \text{ AND } \overline{Y})$$

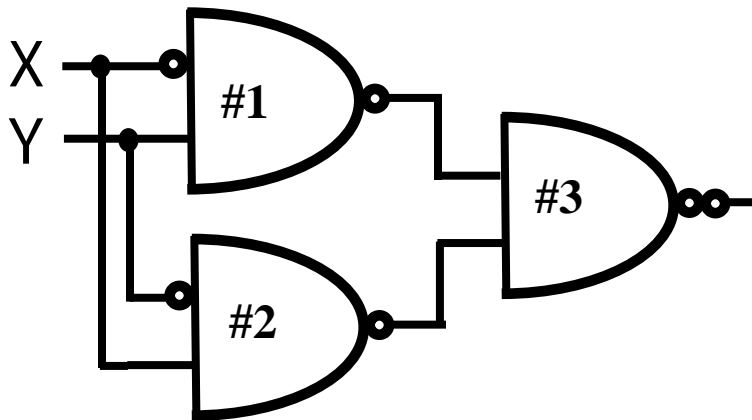
$$F8 = \overline{(\overline{X} \text{ AND } Y) \text{ OR } (X \text{ AND } \overline{Y})}$$

$$F8 = \overline{(\overline{X} \text{ AND } Y)} \text{ AND } \overline{(X \text{ AND } \overline{Y})}$$

$$F8 = \text{NAND}(\overline{X} \text{ with } Y) \text{ AND } \text{NAND}(X \text{ with } \overline{Y})$$

$$F8 = \text{NOT NAND}[(\text{NAND}(\overline{X} \text{ with } Y) \text{ with } \text{NAND}(X \text{ with } \overline{Y}))]$$

Sketch:



Logic Table Proof:

X	Y	\overline{X}	\overline{Y}	Gate #1 \overline{X} NAND Y	Gate #2 X NAND \overline{Y}	Gate #3 Gate#1 NAND Gate #2	NOT (Gate#3)
0	0	1	1	1	1	0	1
0	1	1	0	0	1	1	0
1	0	0	1	1	0	1	0
1	1	0	0	1	1	0	1

NAND Gate Equivalent Circuits

Sketch an equivalent circuit using only NAND gates to represent each of 16 different possible output functions from a combination of 2 inputs (X & Y).

Note:

Indicate inversion with either input bubbles or output bubbles; do not use additional inverter gates.

X	Y	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16
0	0	0	1	0	0	0	1	1	1	0	0	0	1	1	1	0	1
0	1	0	0	1	0	0	1	0	0	1	1	0	1	1	0	1	1
1	0	0	0	0	1	0	0	1	0	1	0	1	1	0	1	1	1
1	1	0	0	0	0	1	0	0	1	0	1	1	0	1	1	1	1

Example: See Function F9 above, use positive logic when the Output Function = 1

$$X = 0, Y = 1 \quad \bar{X} \text{ AND } Y \quad \text{or}$$

$$X = 1, Y = 0 \quad X \text{ AND } \bar{Y}$$

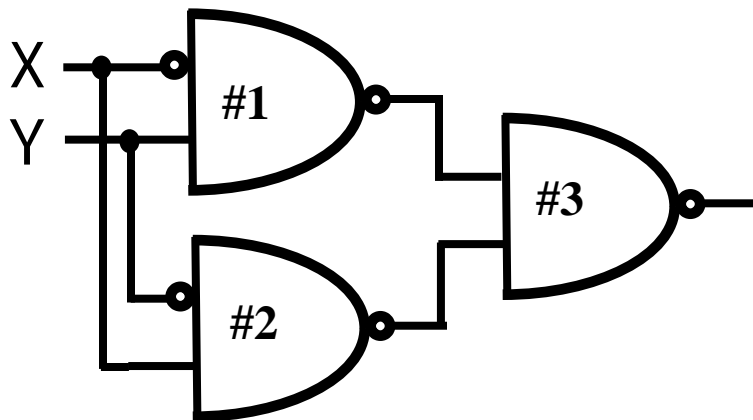
$$F9 = (\bar{X} \text{ AND } Y) \text{ OR } (X \text{ AND } \bar{Y})$$

$$F9 = \overline{\overline{(\bar{X} \text{ AND } Y) \text{ OR } (X \text{ AND } \bar{Y})}}$$

$$F9 = \overline{\overline{(\bar{X} \text{ AND } Y)} \text{ AND } \overline{\overline{(X \text{ AND } \bar{Y})}}} = \text{NAND}(\bar{X} \text{ with } Y) \text{ AND } \text{NAND}(X \text{ with } \bar{Y})$$

$$F9 = \text{NAND}(\text{NAND}(\bar{X} \text{ with } Y) \text{ with } \text{NAND}(X \text{ with } \bar{Y}))$$

Sketch:



Logic Table Proof:

X	Y	\bar{X}	\bar{Y}	Gate #1 \bar{X} NAND Y	Gate #2 X NAND \bar{Y}	Gate #3 Gate#1 NAND Gate #2
0	0	1	1	1	1	0
0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	1	0	0	1	1	0

See page 4 for a similar case using negative logic when the Output Function F9 = 0.

Negative logic when the Output Function $F9 = 0$

$$X = 0, Y = 0 \quad \overline{X} \text{ AND } \overline{Y} \quad \text{or}$$

$$X = 1, Y = 1 \quad X \text{ AND } Y$$

Note $F9 = 0$, is the same as $\overline{F9}$

$$\text{so } \overline{F9} = (\overline{X} \text{ AND } \overline{Y}) \text{ OR } (X \text{ AND } Y)$$

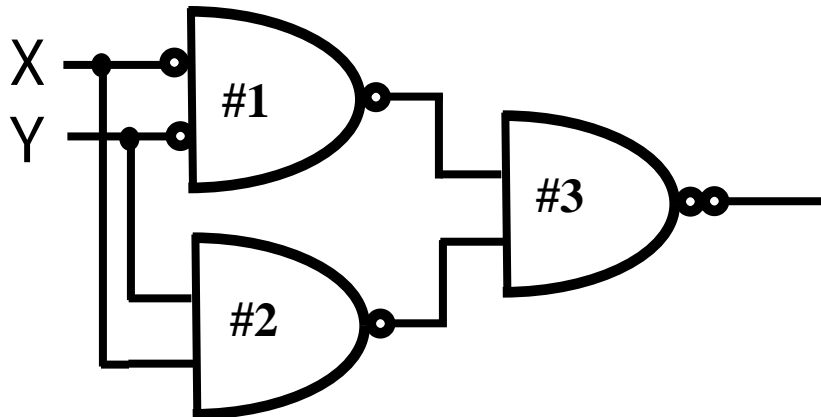
$$F9 = \overline{(\overline{X} \text{ AND } \overline{Y}) \text{ OR } (X \text{ AND } Y)}$$

$$F9 = \overline{(\overline{X} \text{ AND } \overline{Y})} \text{ AND } \overline{(X \text{ AND } Y)}$$

$$F9 = \text{NAND}(\overline{X} \text{ with } \overline{Y}) \text{ AND } \text{NAND}(X \text{ with } Y)$$

$$F9 = \text{NOT NAND} [(\text{NAND}(\overline{X} \text{ with } \overline{Y})) \text{ with } \text{NAND}(X \text{ with } Y)]$$

Sketch:



Logic Table Proof:

X	Y	\overline{X}	\overline{Y}	Gate #1 \overline{X} NAND \overline{Y}	Gate #2 X NAND Y	Gate #3 Gate#1 NAND Gate #2	NOT (Gate#3)
0	0	1	1	0	1	1	0
0	1	1	0	1	1	0	1
1	0	0	1	1	1	0	1
1	1	0	0	1	0	1	0