DIGITAL ELECTRONICS: LOGIC AND CLOCKS

LAB 9 INTRO: INTRODUCTION TO DISCRETE DIGITAL LOGIC, MEMORY, AND CLOCKS

GOALS

In this experiment, we will learn about the most basic elements of digital electronics, from which more complex circuits, including computers, can be constructed.

Proficiency with new equipment and approaches:

- Logic gates, memory circuits, digital clocks
- Combining components & Boolean logic

DEFINITIONS

Duty cycle – percentage of time during one cycle that a system is active (+5V in the case of digital logic)
Truth table – table that shows all possible input combinations and the resulting outputs of digital logic components
Flip-flop - a circuit that has two stable states and can be used to store state information.
Logic gates – a physical device that implements some Boolean logic operation

DIGITAL CIRCUITS - GENERAL

In almost all experiments in the physical sciences, the signals that represent physical quantities start out as *analog* waveforms. To display and analyze the information contained in these signals, they most often are converted into *digital* data. Often this is done inside a commercial instrument such as an oscilloscope or a lock-in amplifier, which is then connected to a computer through a digital interface. In other cases, data acquisition cards are added to a computer chassis, allowing analog signals to be input directly to the computer. Scientists usually buy their data acquisition equipment rather than build it, so they usually don't have to know too much about the digital circuitry that makes it work. Almost all data are eventually analyzed digitally with a computer.

Analog information can be translated into digital form by a device called an Analog-to-Digital Converter (A/D converter or ADC). A set of N bits has 2^N possible different values, as you might recall from Lab #5. If you try to represent an analog voltage by 7 bits, your minimum uncertainty will be about 1%, since there are $2^7 = 128$ possible combinations of 7 bits. For higher accuracy you need more bits. The corresponding device that can convert digital data back into an analog waveform is called a Digital-to-Analog Converter (D/A converter or DAC), which we built in Lab #5.

Logic gates alone can be used to construct arbitrary combinatorial logic (they can generate any *truth-table*), but to create a machine that steps through a sequence of instructions like a computer does, we also need *memory* and a *clock*. The fundamental single-bit memory element of digital electronics is called a *flip-flop*. We will study two types, called SR (or RS) and JK. The flip-flops we have chosen are from the TTL (Transistor-transistor logic) family. A *digital clock* is a repeating digital waveform used to step a digital circuit through a sequence of states. We will introduce the 555 timer chip and use it to generate a clock signal. Digital circuits that are able to step through a sequence of states with the aid of flip-flops and a clock are called sequential logic.

DIGITAL LOGIC STATES

The voltage in a digital circuit is allowed to be in only one of two states: HIGH or LOW. HIGH is taken to mean logical (1) or logical TRUE. LOW is taken to mean logical (0) or logical FALSE. In the TTL logic family (see Figure 1), the "ideal" HIGH and LOW voltage levels are 5 V and 0 V but any input voltage in the range 2 to 5.0 V is interpreted as HIGH, and any input voltage in the range 0 to 0.8 V as LOW. Voltages outside this range are undefined, and therefore "illegal," except if they occur briefly during transitions. If the input to a TTL circuit is a voltage in this undefined range, the response is unpredictable, with the circuit sometimes interpreting it as a "1" and sometimes as a "0." Avoid sending voltage in the undefined range into a TTL components.

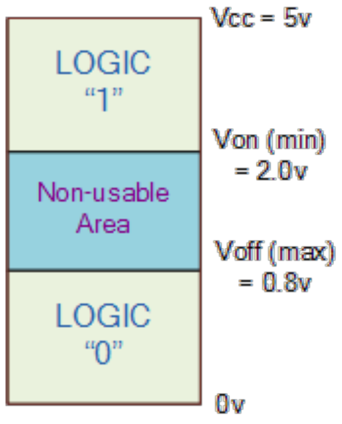


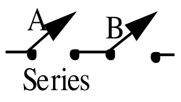
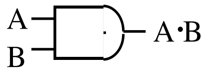
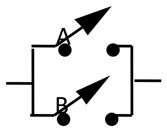
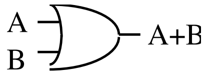
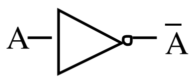
Figure 1: TTL Input Voltage Levels

DIGITAL LOGIC GATES

The flow of digital signals is controlled by transistors in various configurations depending on the logic family (see H&H 8.09 for details). For most purposes, we can imagine that the logic gates are composed of several ideal switches with just two states: OPEN and CLOSED. The state of a switch is controlled by a digital signal. The switch remains closed so long as a logical (1) signal is applied. A logical (0) control signal keeps it open.

Logic signals interact by means of gates. The three fundamental gates, AND, OR, and NOT, are named after the three fundamental operations of logic that they carry out. The AND and OR gates each have two inputs and one output. The output state is determined by the states of the two inputs. The NOT gate has one input and one output.

The function of each gate is defined by a truth table, which specifies the output state for every possible combination of input states. The output values of the truth tables can be understood in terms of two switches. If the switches are in series, you get the AND function. Parallel switches perform the OR operation. The most common gates are shown in Fig. 2. A small circle after a gate or at an input on the schematic symbol indicates negation (NOT).

Operation	Switches	Condition that circuit is closed	Boolean Notation	Symbol	Truth Table
AND		(A AND B are closed)	$A \bullet B$		$\begin{array}{cc\|c} A & B & A\cdot B \\ \hline 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{array}$
OR	 Parallel	(A OR B is closed)	$A + B$		$\begin{array}{cc\|c} A & B & A+B \\ \hline 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{array}$
NOT (same as invert)	Different kind of switch	1 means open 0 means closed	$\text{NOT}\,(A) \equiv \overline{A}$		$\begin{array}{c\|c} A & \overline{A} \\ \hline 0 & 1 \\ 1 & 0 \end{array}$

Compound Gates

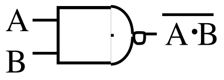
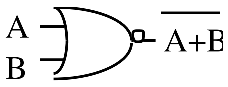
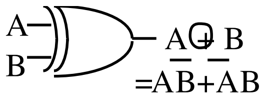
NAND	
NOR	
XOR	

Figure 2: Digital Logic gates

MEMORY ELEMENTS AND FLIP-FLOPS

In sequential logic circuits, the output depends upon previous values of the input signals as well as their present-time values. Such circuits necessarily include memory elements that store the logic values of the earlier signals. The fundamental memory circuit is the RS memory element. The JK flip-flop has an RS flip-flop at its core, but it adds circuitry that synchronizes output transitions to a clock signal. Timing control by a clock is essential to most complex sequential circuits

RS Memory Circuit

The truth table for the RS memory element shows how the circuit remembers. Suppose that it is originally in a state with Q=0 and R=S=0. A positive pulse S at the input sets it into the state Q=1, where it remains after S returns to zero. A later pulse R on the other input resets the circuit to Q=0, where it remains until the next S pulse.

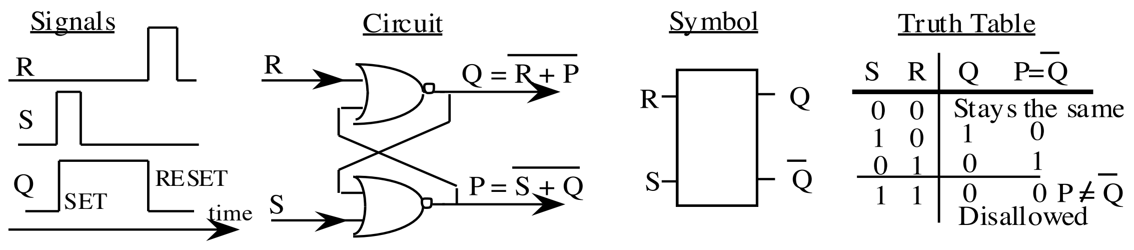


Figure 3: RS memory element.

JK Flip-Flop (TTL74107)

There are three kinds of inputs to the JK flip-flop

1) data inputs J and K
2) the clock C
3) the direct input CLR (clear)

There are two outputs: *Q* and its compliment.

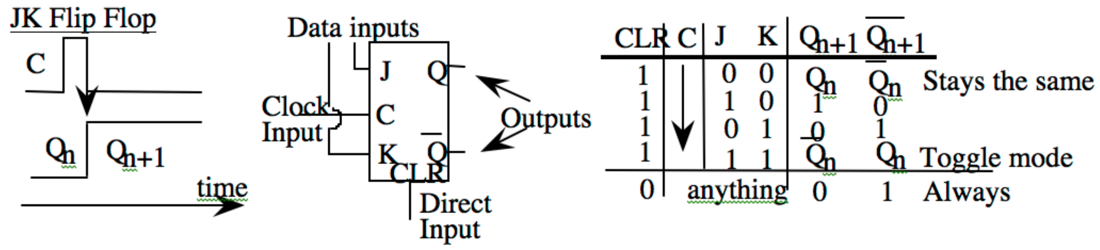

JK Flip Flop
C
Qn
Qn+1
time
Data inputs
Clock Input
J
C
K
Q
$\overline{Q}$
CLR
Outputs
Direct Input
CLR C | J K | Qn+1 $\overline{Qn+1}$
1 ↓ | 0 0 | Qn $\overline{Qn}$ Stays the same
1 | 1 0 | 1 0
1 | 0 1 | 0 1
1 | 1 1 | $\overline{Qn}$ Qn Toggle mode
0 | anything | 0 1 Always

Figure 4: JK Flip-Flop

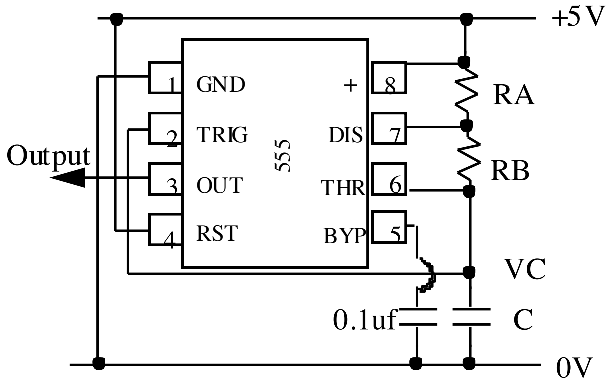
n counts the number of clock pulses since the start of the experiment. In the absence of a clock pulse, the output remains unchanged at the previously acquired value, Q_n, which is independent of the present-time data inputs J and K. Only on the arrival of a clock pulse, C, can the output change to a new value, Q_{n+1}. The value of Q_n depends on the J and K inputs in the way specified in the truth table. The change occurs at the falling (trailing) edge of the clock pulse, indicated by a downward arrow in the truth table in Fig. 4.

The direct input, CLR, overrides the clock and data inputs. During normal operation, CLR = 1. At the moment CLR goes to zero, the output goes to zero and remains there as long as CLR = 0.

555 Timer and Digital Clock

See FC section 11.14 for a description of the guts of the 555 timer chip. Figure 9.7 shows the circuit for generating a clock with the 555 and summarizes the formulas relating the resistor and capacitor values to the output low time T1 and the output high time T2

(a) Astable circuit (Digital Clock)



(b) Component values

Output High (charge time):

$T2 = (RA+RB)C \ln 2$

Output Low (discharge):

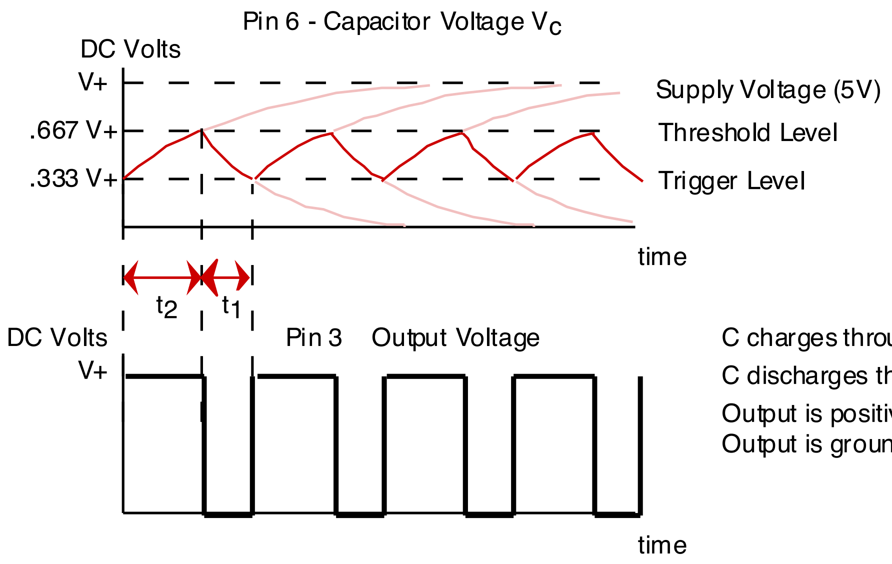
$T1 = RBC \ln 2$

Period: $T = T1 + T2$

(c) Limiting Values

Max RA, RB	3.3 MΩ
Min RA, RB	1 kΩ
Min. C	500pf

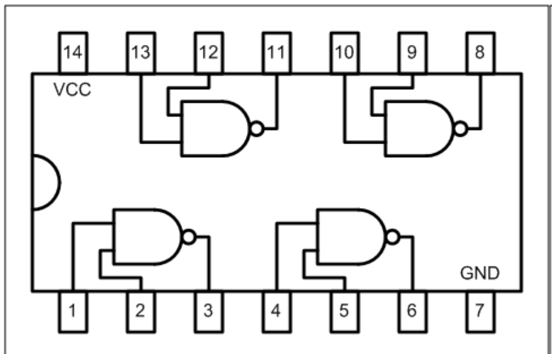
(d) Voltage outputs



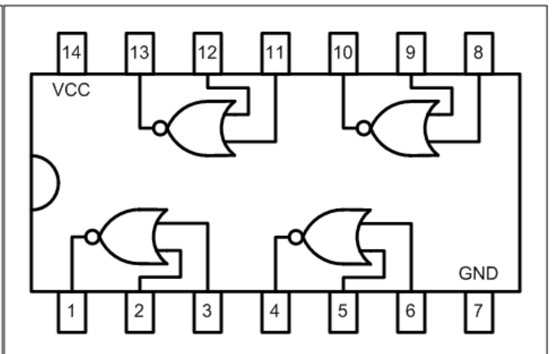
C charges through R_A and R_B in series
C discharges through R_B only
Output is positive while C is charging
Output is grounded while C is discharging

DIGITAL LOGIC CHIP PINOUTS

Each chip has a dot or notch to indicate the end where pins 1 and 14 are located. The pin numbers increase sequentially as you go counter-clockwise around the chip viewed from above. In 74xx family logic chips, pin 7 is always grounded (0 V) and pin 14 is always connected to the +5 V supply.

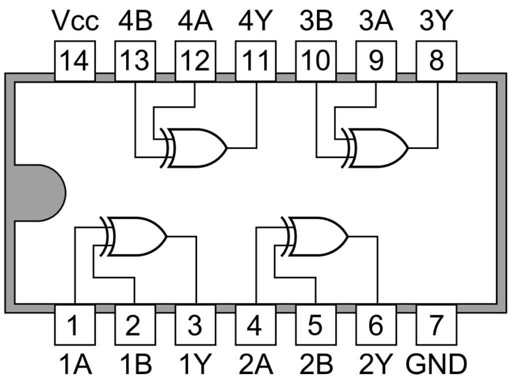


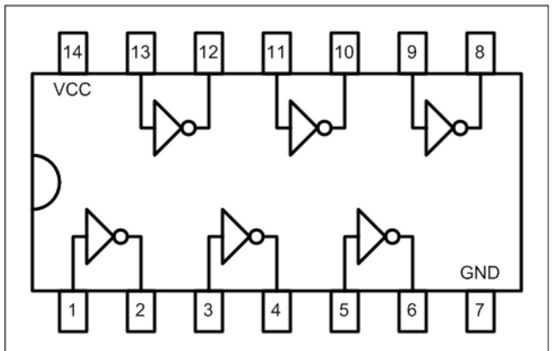
7400 Quad 2 Input NAND



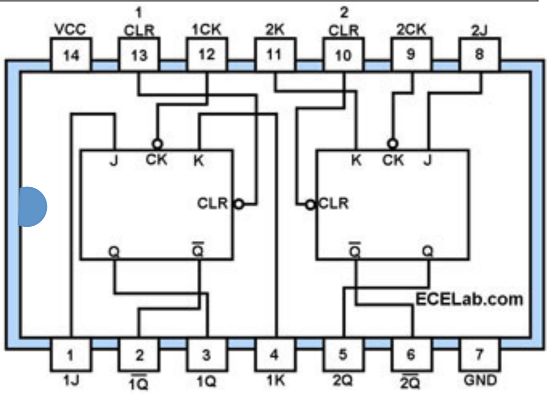
7402 Quad 2 Input NOR

7486 Quad 2-input ExOR Gates



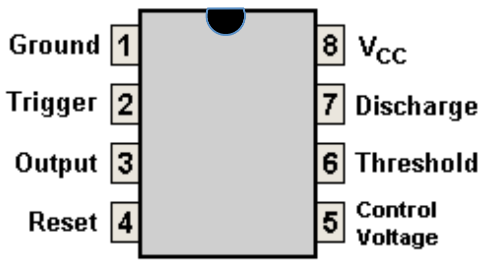


7404 Hex Inverter



74107 JK flip-flop

555 Timer



8-Pin DIP

USEFUL READINGS

1. FC Chapter 11 (digital electronics)
2. H&H Chapter 8. Everything in this chapter is good to know about but sections 8.01, 8.02, 8.04, 8.07-8.10, 8.12, 8.16 are most relevant. Also have a look at section 5.14 on the 555 timer chip.

LAB PREP ACTIVITIES

Answer the following questions using Mathematica or do them by hand in your lab book.

Question 1	**Basic Digital Logic** a. Read the lab thoroughly and enter in your lab book the circuit diagrams and truth tables of all the circuits you will test. These include the NAND, NOR, and INVERT/NOT. b. Design a circuit to perform the EXCLUSIVE OR (XOR) function using only NAND and/or NOR gates. Simplify the circuit so that you use the smallest possible number of NAND and/or NOR gates c. Check the circuit does perform the EXLUSIVE OR using truth tables or Boolean algebra.
Question 2	**555 Timer** a. Design a 4 kHz clock using the 555-timer chip. Make the low level 1/4 of the output period (a 75% duty cycle: 25% low, 75% high). b. How large a capacitor would you need to substitute in order to modify your clock to run at 1 Hz (e.g. for visual observation of LEDs), keeping all other components fixed?
Question 3	**JK Flip-flop** a. A JK flip-flop with J=K=1 and CLR=1 is driven at the clock input by 1 kHz pulses. Draw the waveforms for the clock and the Q output vs. time using the same time scale. Make sure to include enough periods of the clock signal to see all the behavior of the flip-flop's output.
Question 4	**Lab activities** a. Read through all of the lab steps and identify the step (or sub-step) that you think will be the most challenging. b. List at least one question you have about the lab activity.

TTL GATES

Step 1	**Truth Tables** a. Check your power supply before connecting to the circuit board. The Tektronix PS 280/3 has a fixed 5 V output that you should use to power digital circuits. The logic chips will burn out at around 6 V. If the supply voltage drops when you connect to the circuit, do not increase V. b. Input logical values can be set by connecting wires from the gate inputs to either 0 V (logical 0) or 5 V (logical 1). Use one long rail on your prototyping board for 0V and one for 5V. ***Note: Disconnecting an input from the 5 V rail is not the same as connecting it to 0V. If it is disconnected, the input can float up to 5 V on its own.*** c. The logic level of the output can be observed using a light emitting diode (LED), which is connected from the output to ground. The LED lights up when the output is +5 V and is off when the output is 0 V. To limit the amount of current though the diode, place a resistor in series with it. What value of resistor should you use to limit the current to 20 mA? Record your calculation. d. Record the measured truth tables for the NAND (7400), NOR (7402), and INVERT (7404) gates, using the LED indicators for your measurements.

Step 2	**Modifying basic gates** a. Connect a NAND gate so that it performs the INVERT function. Do this for a NOR gate also. This trick will be convenient in simplifying complex circuits. b. Record you circuit and measured truth table.
Step 3	**Exclusive OR** a. Verify the truth tables for an EXCLUSIVE OR chip (7486). b. Now build and test the XOR circuit of your own design using only NANDs and NORs.

MEMORY CIRCIUTS

Step 4	**RS memory circuit** a. Build an RS memory circuit from two NOR gates. Draw a schematic of your circuit. b. Demonstrate the memory property by going through a complete memory cycle: Set (R = 0, S = 1), Store (0, 0), Reset (1, 0), Store (0, 0), Set (0, 1). Record all inputs and outputs for each cycle. Does it agree with predictions? c. Examine the effect of the “illegal” input (R = 1, S = 1), for different initial states of the RS system. Describe the outcomes of the illegal operation.

TTL CLOCK

Step 5	**Digital Clock** a. Build the 4 kHz digital clock using a 555 Timer according to your design in Question 2 of the prelab. Measure the frequency, the pulse length (time the output is high), the duty cycle, and the nominal 5-volt amplitude. Do your measurements agree with your predictions using the measured values of your components? b. Check that a suitable large capacitor placed in parallel with the existing one converts the clock to 1 Hz.

JK FLIP-FLOP

Step 6	a. Construct a truth table for the JK flip-flop from your observations using the LED indicators. Since the output depends upon the previous state, Q_n, you will need to tabulate Q_{n+1} for both possible previous states, $Q_n=0$ and $Q_n=1$. We suggest that you add an additional column, Q_{n+2} , to get a better feel for the behavior of the flip-flop. b. Set CLR = 1 and J = K = 1. Now drive the clock input of the flip-flop with 4 kHz pulses from your clock circuit as shown in Fig. 5. Use the oscilloscope to measure the clock input (positive pulses out of the NAND gate), and the output, *Q*, of the flip-flop. Record your measurements. c. What happens when J = K = 0?

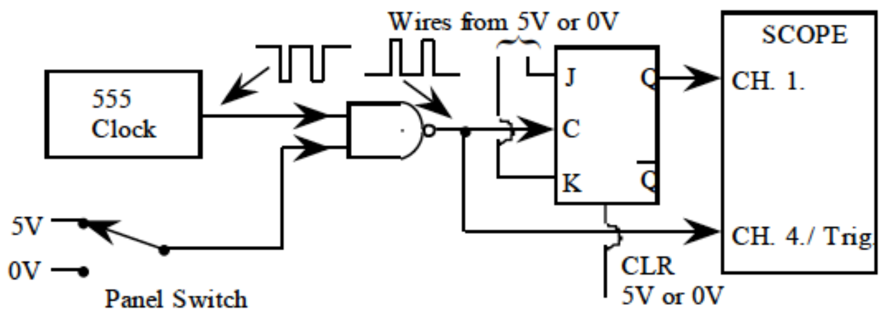


Figure 5: JK Flip-flop test set-up

APPENDIX: BOOLEAN ALGEBRA

Fundamental laws

We imagine a logical variable, *A*, that takes on the values 0 or 1. If *A = 0* then $\bar{A} = 1$ and if *A = 1* then $\bar{A} = 0$. Here are some obvious identities using the AND, OR and NOT operations. Looking at these identities you can see why the 'plus' symbol was chosen for OR and 'times' (•) for AND.

OR	AND	NOT
$A+0=A$	$A\bullet 0=0$	$A+\overline{A}=1$
$A+1=1$	$A\bullet 1=A$	$A\bullet\overline{A}=0$
$A+A=A$	$A\bullet A=A$	$\overline{\overline{A}}=A$
$A+\overline{A}=1$	$A\bullet\overline{A}=0$	

Equality

Two Boolean expressions are equal if and only if their truth tables are identical.

Associative Laws

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

Distributive Laws

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

Related identities:

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

DeMorgan's Theorems

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

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

Example of Proof

Each of the above equalities is a theorem that can be proved. Let's do an example by directly comparing the truth tables for the left and right sides. We take on DeMorgan's first theorem for two variables, $\overline{AB} = \bar{A} + \bar{B}$

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

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

The last columns of the truth tables are identical. Thus, the first theorem is proven for two variables.

Example of Simplification

Boolean algebra can be used to simplify logical expressions and reduce the number of gates required in a circuit. In Fig. 9.3 we show two ways to implement the expression, $Y = A + \bar{A}BC$.

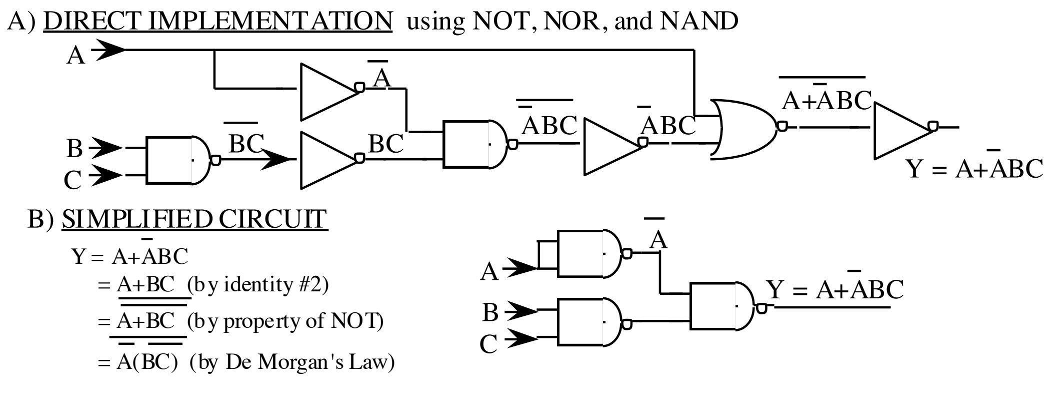

A) DIRECT IMPLEMENTATION using NOT, NOR, and NAND
A
B
C
$\overline{BC}$
$\bar{A}$
BC
$\overline{\bar{A}BC}$
$\bar{A}BC$
$\overline{A+\bar{A}BC}$
$Y = A+\bar{A}BC$
B) SIMPLIFIED CIRCUIT
$Y = A+\bar{A}BC$
$= A+BC$ (by identity #2)
$= \overline{\overline{A+BC}}$ (by property of NOT)
$= \overline{\bar{A}(\overline{BC})}$ (by De Morgan's Law)
A
B
C
$\bar{A}$
$Y = A+\bar{A}BC$

Fig. 9.3. Boolean simplification