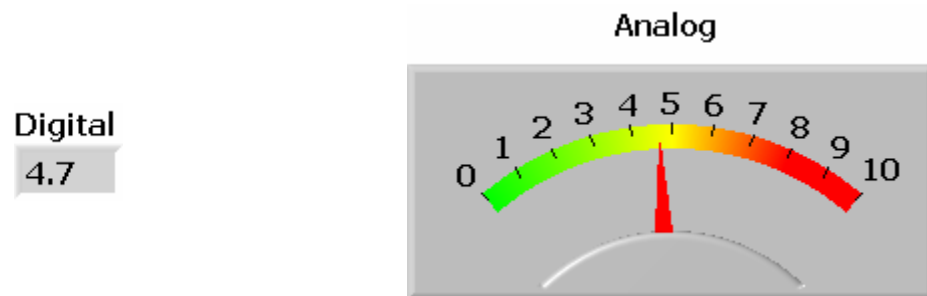


## ***Introduction to Digital:***

### ***Combinational Logic and Systems Design***

So far we have been discussing the generation, transmission and processing of signals whose amplitude (voltage, current) varies continuously in time and can in principle take any value.

At a certain instant of time we may represent a signal by displaying its amplitude in an analog form or in a digital format. The graphics below demonstrate the familiar representation of the two forms. Both displays are asked to display the number 4.7.



In this case the digital display on the left has the required resolution to represent the number exactly. If we try to display the number 4.76, this digital display, with its ability to display only 2 digits will have to round off the number, representing it either as 4.7 or 4.8, depending on how the system processes the numerics.

Our reading of the number off the analog display requires some interpolation of the value but in principle the resolution is only limited by our ability to identify the position of the measuring needle.

In general the characteristics of the digital display correspond to the digital signal which is required to generate the digital display in the first place.

If we now consider an analog signal which varies continuously in time, see Figure 2a, then if we sample the signal at discrete times ( $\tau$ ,  $2\tau$ ,  $3\tau$ ,  $k\lambda\tau$ ) we will obtain the values indicated by the solid circles on Figure 2b. Furthermore, if we consider the **quantization** of the signal at these **discrete** sampling times we obtain the signal indicated on Figure 2c which is a **digital signal**. The analog signal is also shown on Figure 2c to emphasize the relationship between it and its digital representation.

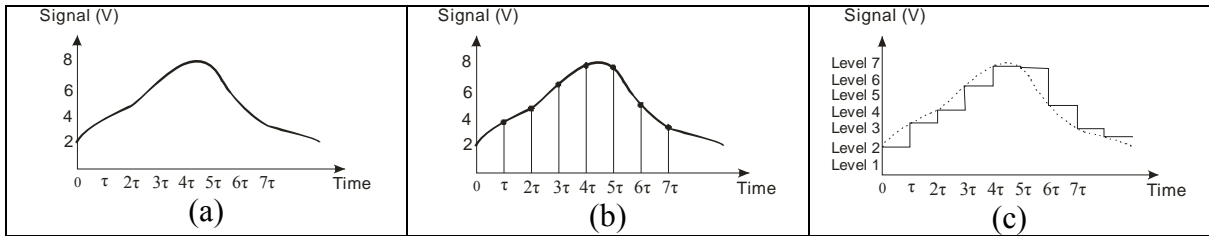


Figure 2. Schematic representation of analog and digital signals.

This review should have motivated us to ask a few questions about these signals and in particular about the digital signal shown on Figure 2c.

Some of these questions are:

1. How is the information embodied by the digital signal represented?
2. How is the signal generated?
  - a. How is the sampling frequency selected and how is it related to the “quality” of signal representation?
  - b. How is the amplitude quantization achieved?
3. What are the advantages and disadvantages of generating the digital signal? For example, how does it perform in
  - a. Accuracy
  - b. Transmission
  - c. Noise immunity
  - d. Information storage
  - e. Computation

In the next few classes we will answer these questions and explore the fundamental issues associated with the design of digital circuits.

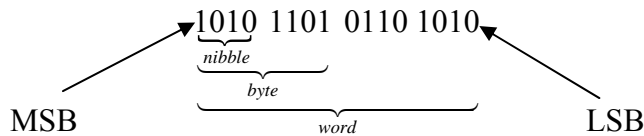
## Numbering Systems

### Binary Code.

In digital electronics the signals are formed with only two voltage values, HI and LOW, or level **1** and level **0** and it is called binary digital signal.<sup>1</sup> Therefore, the information contained in the digital signal is represented by the numbers **1** and **0**. In most digital systems the state **1** corresponds to a voltage range from 2V to 5V while the state **0** corresponds to a voltage range from a fraction of a volt to 1 volts.

Digital operations are performed by creating and operating on binary numbers. Binary numbers are comprised of the digits 0 and 1 and are based on powers of 2.

Each digit of a binary number, 0 or 1, is called a bit, an abbreviation for **binary digit**. Four bits together is a nibble, 8 bits is called a byte. (8, 16, 32, 64 bit arrangements are also called words) The rightmost bit is called the Least Significant Bit (LSB) while the leftmost bit is called the Most Significant Bit (MSB). The schematic below illustrates the general structure of a binary number and the associated labels.



---

<sup>1</sup> In addition to binary digital systems and its associated binary logic, multivalued logic also exists but we will not consider it in our discussion.

## Binary to Decimal Conversion.

The conversion of a binary number to a decimal number may be accomplished by taking the successive powers of 2 and summing for the result.

For example let's consider the four bit binary number 0101. The conversion to a decimal number (base 10) is illustrated below.

$$\begin{array}{ccccccccccc}
 0 & & 1 & & 0 & & 1 & & & & \\
 \downarrow & & \downarrow & & \downarrow & & \downarrow & & & & \\
 \underbrace{0 \times 2^3} & + & \underbrace{1 \times 2^2} & + & \underbrace{0 \times 2^1} & + & \underbrace{1 \times 2^0} & & & & \\
 \downarrow & & \downarrow & & \downarrow & & \downarrow & & & & \\
 0 & + & 4 & + & 0 & + & 1 & = & 5_{10} & & 
 \end{array}$$

For this four bit binary number the range of powers of 2 goes from 0, corresponding to the LSB, to 3, corresponding to the MSB. The number 5 is shown as  $5_{10}$  to indicate that it is a decimal number (power of 10).

The signal represented on Figure 2c has a value of 5 V at time= $6\tau$ . The binary representation of that value is 0101 and it is shown on Figure 3 replacing Level 4. We will see more of this later when we consider the fundamentals of the device which converts the analog signal to a digital signal.

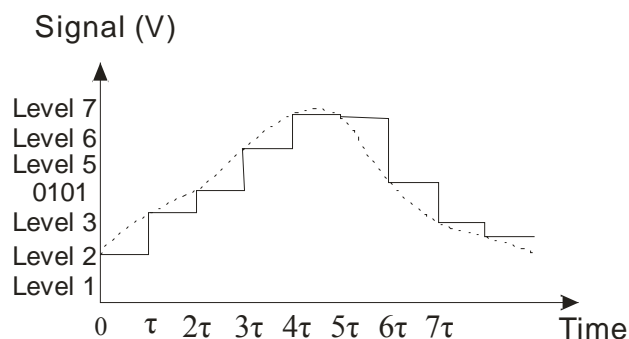


Figure 3.

In the next few examples we will use the subscript 2 to indicate a binary number but the subscripts will be omitted after that.

Examples:

Verify the Binary to Decimal conversion

$$1111_2 = 15_{10}$$

$$1111\ 0000_2 = 240_{10}$$

$$1111\ 1111_2 = 255_{10}$$

$$1101\ 1011_2 = 219_{10}$$

$$0001\ 0101\ 1011_2 = 347_{10}$$

$$1001\ 0101\ 1011_2 = 2395_{10}$$

### Decimal to Binary Conversion.

The conversion of a decimal number to a binary number is accomplished by successively dividing the decimal number by 2 and recording the remainder as 0 or 1. Here is an example of the conversion of decimal number 125 to binary.

$$\begin{array}{r} \frac{125}{2} = 62 + 1 \\ \frac{62}{2} = 31 + 0 \\ \frac{31}{2} = 15 + 1 \\ \frac{15}{2} = 7 + 1 \\ \frac{7}{2} = 3 + 1 \\ \frac{3}{2} = 1 + 1 \\ \frac{1}{2} = 0 + 1 \end{array} \left. \begin{array}{l} \text{LSB} \\ \\ \\ \\ \\ \\ \text{MSB} \end{array} \right\} \Rightarrow 0111\ 1101$$

Practice number conversion by verifying the conversions from decimal to binary:

Decimal	Binary
69	0100 0101
299	0001 0010 1011
756	0010 1111 0100

Representation of fractions and signed numbers.

A fractional number may be represented as a binary fraction by simply extending the procedure used in representing integer numbers. For example,

$$13.75_{10} = 1101.1100_2$$

The procedure is clearly visualized by considering the following mapping

$2^3$	$2^2$	$2^1$	$2^0$		$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$
8	4	2	1		0.5	0.25	0.125	0.0625
1	1	0	1	.	1	1	0	0
			13	.	75			

Signed binary numbers may be represented by assigning the MSB to indicate the sign. A 0 is used to indicate a positive number and a 1 is used to indicate a negative number.

For example, an 8 bit signed binary number represents the decimal numbers from -128 to +127.

Two's complement is used to represent negative numbers. The use of 2's complement simplifies the operation of subtraction since the circuit is only required to perform the operation of addition since  $X - Y = X + (-Y)$ .

The 2's complement of a binary number is obtained by subtracting each digit of the binary number from digit 1. This is equivalent to replacing all 1's by 0's and all 0's by 1's. Negative numbers of 2's complement can then be found by adding 1 to the complement of a positive number.

For example, the 2's complement of the 8 bit binary number 0000 1110 is  
 $1111\ 0001 = 10_{10}$

The negative number of this 2's complement representation is  
 $1111\ 0110 = -10_{10}$

The procedure is outlined in the following

0	0	0	0	1	0	1	0	binary number ( $10_{10}$ )
1	1	1	1	0	1	0	1	2's complement
							1	
1	1	1	1	0	1	1	0	$-10_{10}$

By adding the two numbers the result is zero as shown below.

$$\begin{array}{r}
 0000\ 1110\ (+10) \\
 +\ 1111\ 0110\ (-10) \\
 \hline
 0000\ 0000\ (0)
 \end{array}$$

The table below shows the 2's complement representation of a few numbers. Fill in the empty spaces.

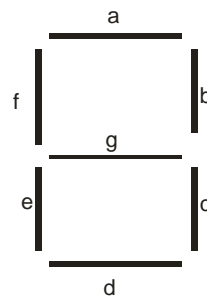
Decimal	2's complement
0	0000 0000
-1	1111 1111
-2	1111 1110
-3	1111 1101
-4	1111 1100
-10	1111 0110
-15	
-27	
-80	
-110	

## Binary Coded Decimal. BCD Code

BCD is a code used to represent each digit of a decimal number (0 to 9) as a 4 bit binary. For example, the decimal number 260 corresponds to the BDC number 0010 0110 0000.

$$\begin{array}{cccc}
 \underline{2} & \underline{6} & \underline{0} & \text{Decimal} \\
 \downarrow & \downarrow & \downarrow & \\
 0010 & 0110 & 0000 & \text{BDC}
 \end{array}$$

This code is used to drive the 7 segment led displays.



For example, the BCD number 0010 corresponds to decimal 2 and is used to drive the segments a,b,d,e,g of the display. Similarly the number 0110 corresponds to number 6 and it is used to drive segments a,c,d,e,f,g. BCD 0000 corresponds to decimal 0 and it drives

segments a,b,c,d,e,f. Special logic ICs are available for driving the led segments from a BCD number.

### **Numbers with other bases.**

The octal system, with base 8 and digits 0,1,2,3,4,5,6,7, and the hexadecimal system with base 16 and digits 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F are also used in digital electronics. Octal and Hex representation is more compact, consider the conversion of the decimal number 1132 in binary, Octal and Hex shown below, and it is used in assembly language programming of microcontrollers.

$$1132_{10} = 0100\ 0110\ 1100_2 = 2154_8 = 46C_{16}$$



## ***Fundamental Digital Devices: The inverter.***

The fundamental digital circuit for performing binary operations is the one which will convert from a logic **1** to a logic **0** and vice-versa. In our discussions we will use the positive logic convention which implies that the logic level 1 will correspond to the higher voltage level and the logic level 0 will correspond to the lower voltage level. Such a fundamental is shown on Figure 4.

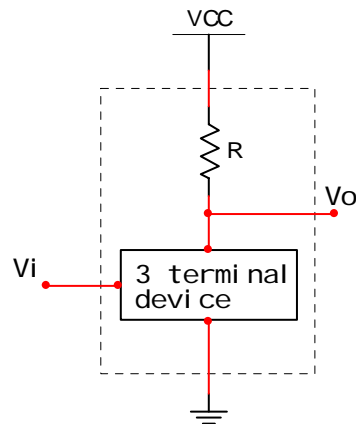


Figure 4. Fundamental inverter circuit.

The ideal voltage transfer characteristic of this circuit is shown on Figure 5.

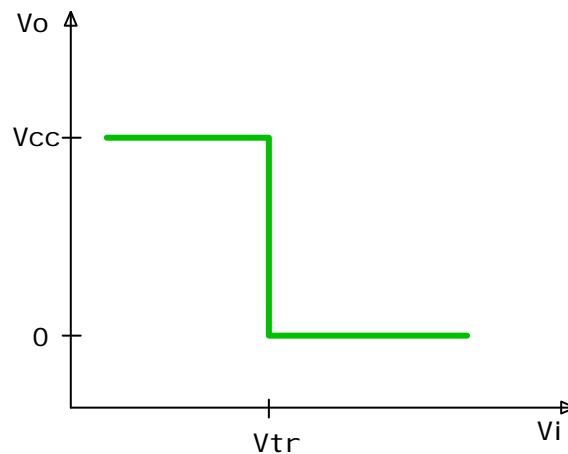


Figure 5. Ideal inverter voltage transfer characteristic

When the input voltage exceeds the transition value  $V_{tr}$ , the output switches.

It will become useful to familiarize ourselves with time evolution of digital signals. Such representation is called **timing diagram** which is used extensively in representing the operation of digital circuits. For our idealized inverter circuit, also called “inverter gate” a timing diagram would look like

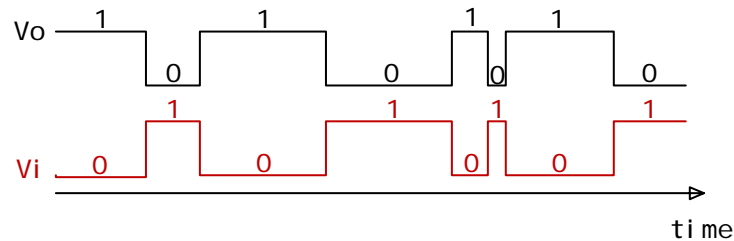


Figure 6. Ideal inverter timing diagram.

We thus see that the inverter is a voltage controlled digital switch. In practice the behavior of the inverter is not ideal. The output could assume a low value which will be in a range of voltages and a high value which also encompasses a range of voltage values. In addition the transition occurs with a time delay.

The inverter circuit may be constructed with active devices such as the Field Effect Transistor (FET) or the Bipolar Junction Transistor (BJT). The inverter circuit arrangements for these fundamental devices are shown on Figure 7.

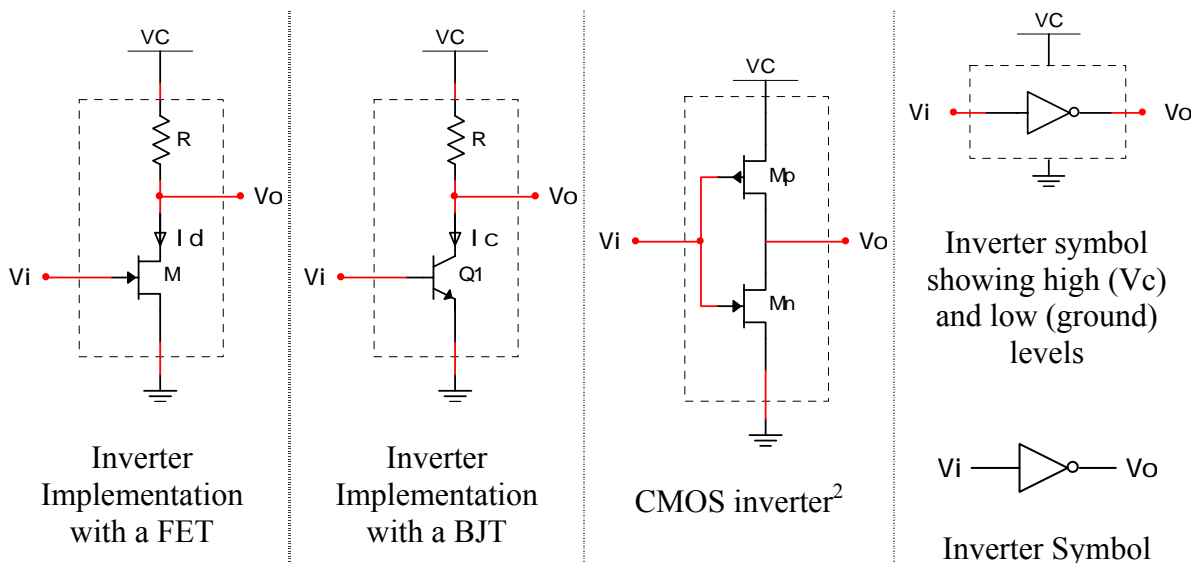
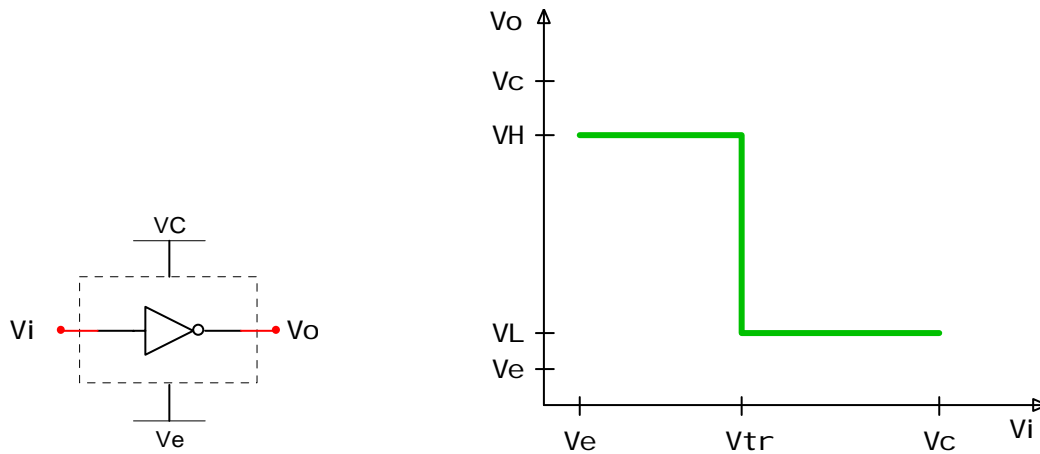


Figure 7. Inverter circuits and inverter symbol

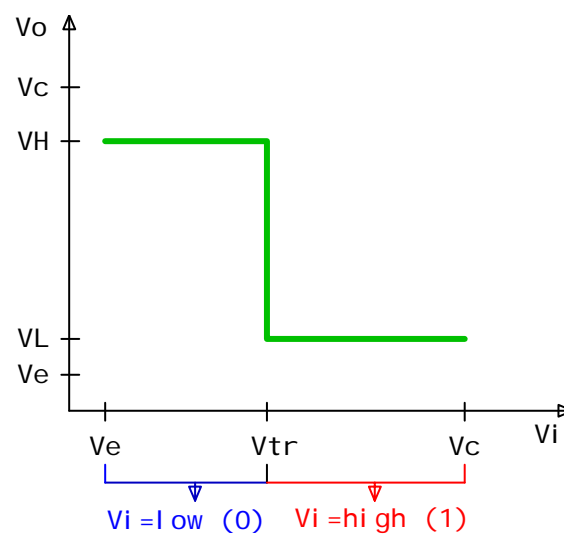
<sup>2</sup> The Complementary Metal-Oxide Semiconductor (CMOS) inverter incorporates an n-channel and a p-channel MOSFET.

The output voltage is the inverse of the input and the switching from one state to another state happens when the input voltage crosses a certain value  $V_{tr}$ . An inverter with supply voltages  $V_c$  and  $V_e$  and the corresponding voltage transfer characteristic is,



When  $V_i$  is less than the voltage  $V_{tr}$  (i.e. when  $V_i$  is low) the output is  $V_H$  (high). As the input voltage exceeds  $V_{tr}$ , ( $V_i$  is now high) the output switches to  $V_L$  (low). Note that the values for  $V_H$  and  $V_L$  are not the same as the supply rails of the device. The actual values of  $V_H$  and  $V_L$  depend on the particular technology used in the construction of the inverter circuit.

For each design and for the particular semiconductor technology used the supply voltages  $V_c$  and  $V_e$  are well defined. A familiar power supply for digital systems is the  $V_c = +5V$  and  $V_e = 0V$ . Supply systems with,  $V_c$ , voltage levels ranging from 1.5V to 5V are available.



In practical circuits the transition from one state to another does not happen abruptly. The transition is gradual and there is not a single value at which the transitions happen but

rather a range of values that correspond to the transition as well as to the high and low states. This is a desirable situation since it allows for the design of robust systems with considerable noise immunity.

In addition there is a range of values which correspond to a certain logic level. Figure 8 shows a generic representation of the logic levels.

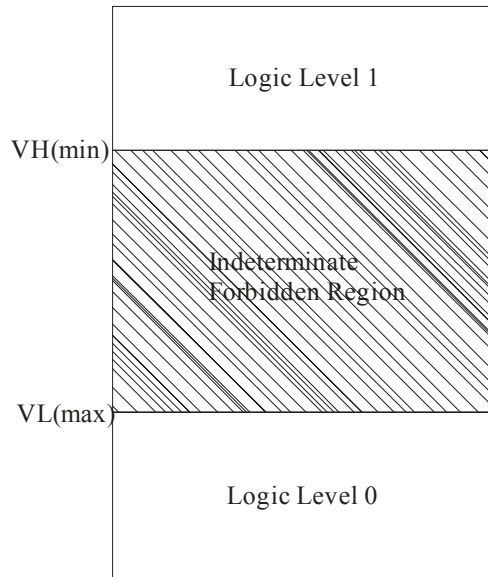


Figure 8. Logic gate levels

Any value less than  $VH(\min)$  and greater than  $VL(\max)$  falls in the forbidden region and its state is indeterminate. The values for  $VH(\min)$  and  $VL(\max)$  are different for the input and output of a gate. It is important to pay particular attention to these voltage levels since when one gate drives another, the input and output of each should fall within the specified level values.

When one gate drives another as in the graphic shown on Figure 9, the various voltage levels are defined as follows:

- $VH_{in}(\min)$ : The minimum voltage level required for a logic level 1 at the input of a gate.
- $VL_{in}(\max)$ : The maximum voltage level required for a logic 0 at the input of a gate.
- $VH_{out}(\min)$ : The minimum voltage level required for a logic level 1 at the output of a gate.
- $VL_{out}(\max)$ : The maximum voltage level required for a logic 0 at the output of a gate.

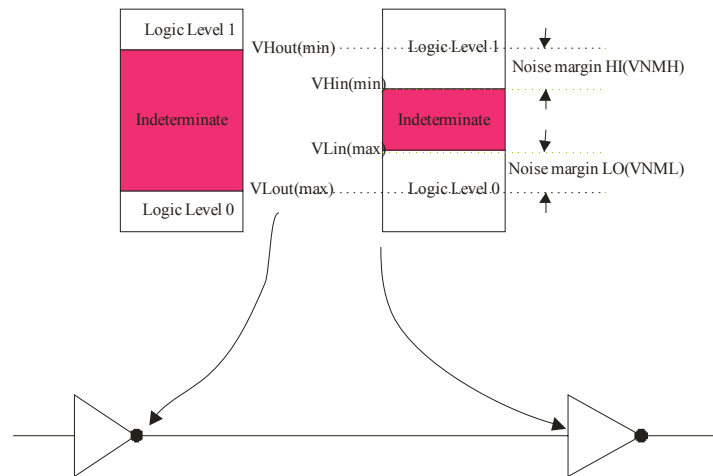


Figure 9. Logic gate input and output voltage levels

As can be seen from Figure 9, the presence of noise at the output of a certain gate may result in a voltage level which may be recognized as an appropriate state at the input of the following gate. The maximum voltage deviations due to noise that can be accepted by the logic gate inputs are defined as the HI and LO voltage noise margins,

$$V_{NMH} = V_{1out(min)} - V_{1in(min)}$$

$$V_{NML} = V_{0in(max)} - V_{0out(max)}$$

These voltage levels are summarized on Figure 10.

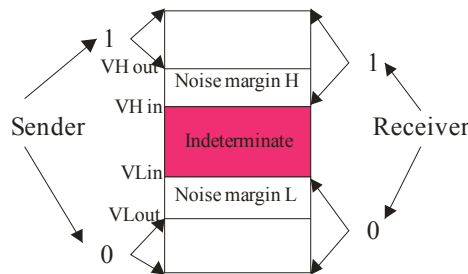


Figure 10. Voltage levels for sender and receiver.

The actual values of the minimum and maximum voltage values at the input and output of the logic gates depend on the type of device used to construct the logic gate. Currently there are two dominant logic families. They are the Transistor to Transistor Logic (TTL) based on the BJT inverter shown on Figure 7 and the Complementary Metal-Oxide Semiconductor (CMOS), based on the MOS families. Besides differences in speed and power consumption, these logic families are also different in the acceptable logic voltage levels for the gates. These differences should be taken into consideration when the design involves interfacing between TTL and CMOS circuits.

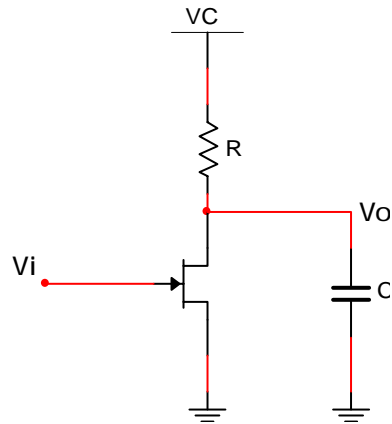
Table I provides a general comparison of the two families.

	TTL	CMOS
Supply voltage (V)	5	5
$V_{Iin(min)}$	2	3.5
$V_{Oin(max)}$	0.8	1.5
$V_{Iout(min)}$	2.4	4.5
$V_{Oout(max)}$	0.4	0.1
$V_{NMH}$	0.4	1.0
$V_{NML}$	0.4	1.4

Table I. Comparison of TTL and CMOS logic family parameters.

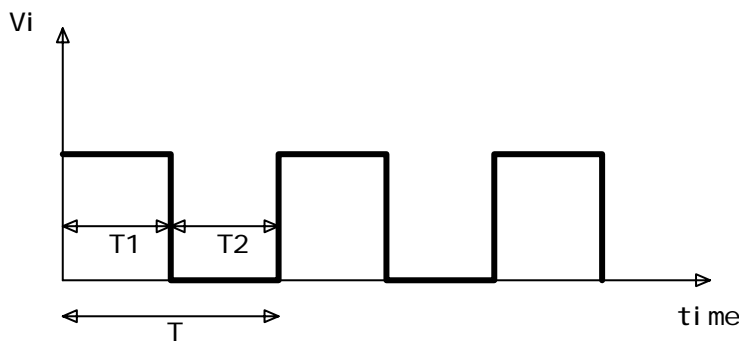
### A few words on power consumption.

Let's consider the inverting gate,

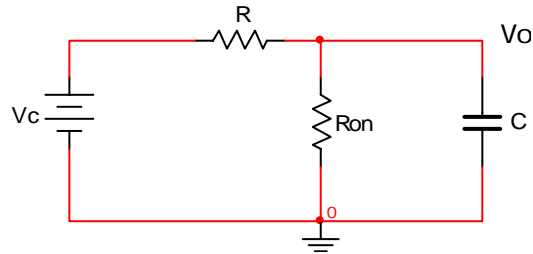


The capacitor C represents the wiring capacitance as well as the capacitance of the gate-source junction of the following gate.

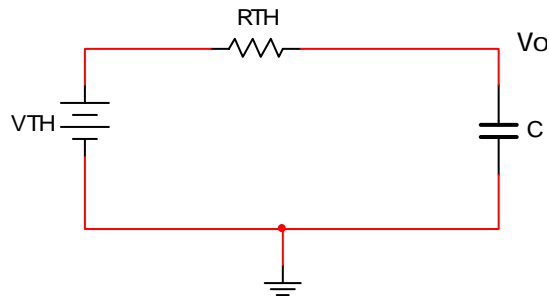
Let's calculate the average power for an input signal  $V_i$  has the form



During time T1 the FET is on and the equivalent circuit is



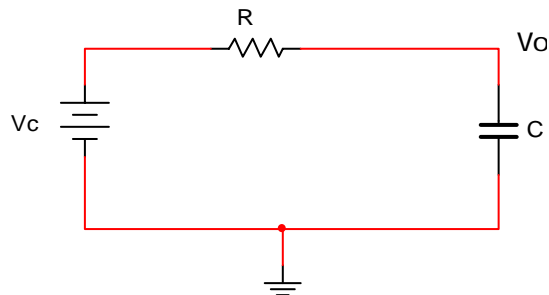
Where Ron is the on resistance of the FET. The Thevenin equivalent circuit is



Where  $V_{TH} = V_c \frac{R_{on}}{R+R_{on}}$  and  $R_{TH} = \frac{R(R_{on})}{R+R_{on}}$ . By assuming that time  $T1 \gg (R_{TH} C)$  the energy dissipated during T1 is (The details of the calculation are left as an exercise for the student)

$$E1 = \underbrace{\frac{V_c^2 T1}{R+R_{on}}}_{\text{Static}} + \underbrace{\frac{V_{TH}^2 C}{2}}_{\text{Dynamic}} = \underbrace{\frac{V_c^2 T1}{R+R_{on}}}_{\text{Static}} + \underbrace{\frac{V_c^2 R^2 C}{2(R+R_{on})^2}}_{\text{Dynamic}}$$

During time T2 the equivalent circuit is



During this time interval the capacitor will discharge starting from the voltage VTH reached during time T1. The energy dissipated during T2 is then (assuming  $T2 \gg RC$ )

$$E2 = \frac{Vc^2 R^2 C}{\underbrace{2(R+Ron)^2}_{\text{Dynamic}}}$$

The total power dissipated during one period (T), assuming that T1 = T2 is

$$P = \underbrace{\frac{Vc^2}{2(R+Ron)}}_{\text{Static}} + \underbrace{\frac{Vc^2 R^2 C}{(R+Ron)^2 T}}_{\text{Dynamic}}$$

Since  $R \gg Ron$  (usually) the total power is

$$P = \underbrace{\frac{Vc^2}{2R}}_{\text{Static}} + \underbrace{\frac{Vc^2 C}{T}}_{\text{Dynamic}} = \underbrace{\frac{Vc^2}{2R}}_{\text{Static}} + \underbrace{Vc^2 f C}_{\text{Dynamic}}$$

The static power is independent of frequency ( $f=1/T$ ). The dynamic power is proportional to frequency and the supply voltage  $Vc$ .

As an example let's consider a digital device (chip) which incorporates  $10^9$  gates operating at a frequency of 1GHz. Typical values are:

$$R = 10 \text{ k}\Omega$$

$$C = 0.1 \text{ fF}$$

$$Vc = 5V$$

Total power is,

$$Pt = 10^9 \left( \frac{25}{20000} + 25 \times 10^{-16} \times 10^9 \right)$$

$$= \underbrace{1250 \text{ kW}}_{\text{Static power}} + \underbrace{2.5 \text{ kW}}_{\text{Dynamic power}}$$

Well... something has to be done.

Indeed the static power can be practically reduced to zero by the CMOS design which basically eliminates the path to ground through resistor  $Ron$ .

The Dynamic power can be reduced by decreasing  $Vc$ . By going from  $Vc=5 \text{ V}$  to  $Vc=1 \text{ V}$  the dynamic power is reduced from 2500 W to 100 W. Not bad.



## Signal Conversion.

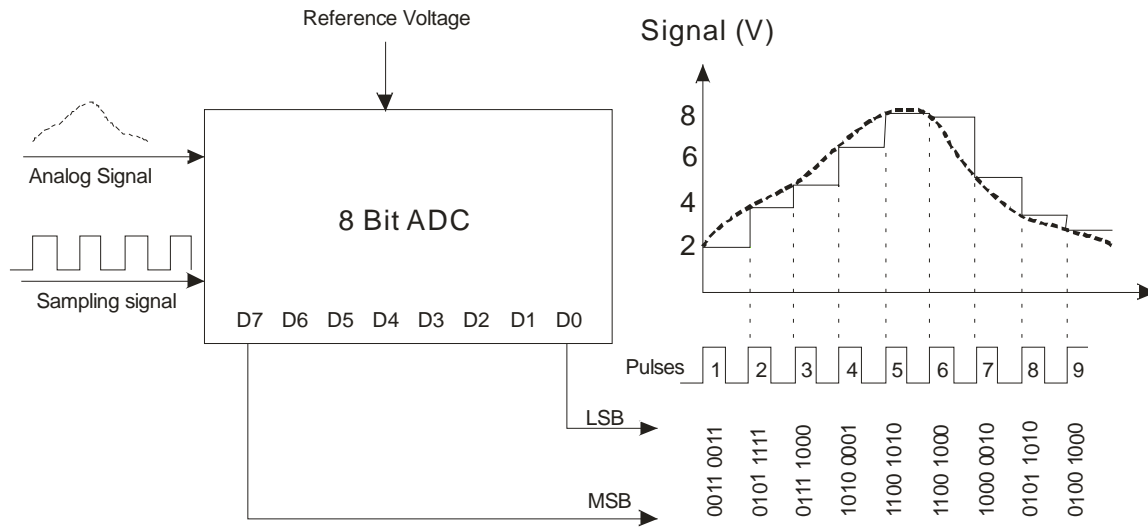
### Analog to Digital Conversion

The electrical signals (voltage or current) generated by a transducer is an analog signal. The amplitude of the signal corresponds to the value of the physical phenomenon that the transducer detects. The signal values are continuous in time.

The processing of the signal by a digital system requires the conversion of the analog signal to a digital signal. The analog to digital conversion is not a continuous process but it happens at discrete time intervals. Furthermore the magnitude of the digital signal at the time of conversion should correspond to the magnitude of the analog signal.

The analog to digital converter (ADC) is a device that receives as its input the analog signal along with instructions regarding the sampling rate (how often is a conversion going to be performed) and scaling parameters corresponding to the desired resolution of the system. The output of the ADC is a binary number at each sampling time.

The following schematic shows the basic structure of an 8 bit ADC.



The selection of an 8 bit ADC sets the resolution of our conversion and the selection of the scale for the analog signal determines the measurement resolution for our ADC. In our example the 8 bit ADC implies  $2^8 = 256$  different levels within the maximum signal range.

Since we are measuring a voltage with possible values between 0V and 10V, our 8 bit ADC is not able to resolve voltages smaller than  $\frac{10}{2^8} \text{ mV} = 39 \text{ mV}$ .

If this ADC has a resolution of 16 bits, like the one that you have in your laboratory, the resolution, for the same measurement range, would be  $\frac{10}{2^{16}} \text{ mV} = 0.15 \text{ mV}$ .

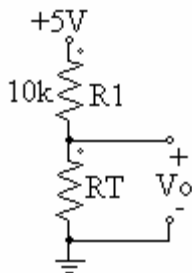
The table below summarizes the conversion process

Pulse	Signal Value	Level	Binary number
1	2	$\frac{2}{10}256 = 51$	0011 0011
2	3.7	$\frac{3.7}{10}256 = 95$	0101 1111
3	4.7	$\frac{4.7}{10}256 = 120$	0111 1000
4	6.3	$\frac{6.3}{10}256 = 161$	1010 0001
5	7.9	$\frac{7.9}{10}256 = 202$	1100 1010
6	7.8	$\frac{7.8}{10}256 = 200$	1100 1000
7	5.1	$\frac{5.1}{10}256 = 130$	1000 0010
8	3.5	$\frac{3.5}{10}256 = 90$	0101 1010

The sampling frequency must be larger than the highest frequency of the analog signal to be converted. In fact as stated by the “Sampling Theorem” ***The sampling frequency must be at greater than 2 times the bandwidth of the input signal.***

Problem:

You would like to design a thermistor based thermometer with a resolution of 0.1° Celcius in the range of 0 to 100 degrees Celsius. Let’s assume that your thermistor (RT) has an accuracy of 0.001 Degrees Celsius and that the analog signal is generated by the standard voltage divider network,



Design your instrument by appropriately selecting resistor R1 and the resolution of the ADC.

## Digital Logic: Boolean Algebra

Boolean algebra is a special symbolic mathematical language used to carry out logic operations. Boolean algebra was developed in the mid 1800s by George Boole (1815-1864) an English mathematician, for the purpose of performing “Logic and Probability Calculations”. In Boolean algebra variables have only two possible values, 1 or 0 and it is the language used for performing binary logic operations.

The basic logic operations and their corresponding Boolean representation and associated symbols are given on Table II. The variables A and B are the inputs to the functions (Gates) and can assume the value 0 or 1. The variable Y represents the output of the logic operation or equivalently the output of the logic gate and its value is also either 0 or 1.




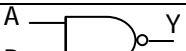

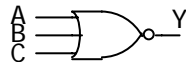
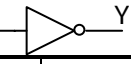
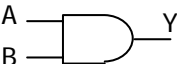



Logic Operation	Boolean Representation	Logic Symbol (Gates)
NOT	$Y = \bar{A}$	
AND	$Y = A \cdot B = AB$	
OR	$Y = A + B$	
NAND	$Y = \overline{A \cdot B} = \overline{AB}$	
NOR	$Y = \overline{A + B}$	

Table II. Basic logic operations and their Boolean representation

The NOT operation (inversion) is performed by the inverter discussed earlier and it has one input and one output. The number of inputs to the AND, OR, NOR, NAND logic gates can be greater than two. For example the Boolean representation of the 3 input NOR gate is  $Y = \overline{A + B + C}$  and the corresponding logic gate is,



The rules associated with each logic operation (function) may be represented in a useful tabular form by the Truth Table. The Truth Tables for the basic logic operations listed on Table II are:

NOT		AND			OR			NAND			NOR		
													
A	Y	A	B	Y	A	B	Y	A	B	Y	A	B	Y
0	1	0	0	0	0	0	0	0	0	1	0	0	1
1	0	0	1	0	0	1	1	0	1	1	0	1	0
		1	0	0	1	0	1	1	0	1	1	0	0
		1	1	1	1	1	1	1	1	0	1	1	0

The designer of logic circuits and systems must optimize the design for maximum performance (i.e low power, speed, etc) and for the lowest cost. The simplification of logic expressions results in a simplified digital logic circuit. Logic expressions may be simplified by making use of the following Boolean identities.

Boolean Identities	
$A + 0 = A$	$A \cdot 1 = A$
$A + B = B + A$	$AB = BA$
$A + (B + C) = (A + B) + C$	$A(BC) = (AB)C$
$A + BC = (A + B)(A + C)$	$A(B+C) = AB + AC$
$A + A = A$	$A \cdot \bar{A} = 0$
$A + 1 = 1$	$A \cdot A = A$
$A + \bar{A} = 1$	$A \cdot 0 = 0$
$A + AB = A$	$A(A + B) = A$
$A + \bar{A}B = A + B$	$(A + B)(A + C) = A + BC$
$\overline{AB} = \bar{A} + \bar{B}$	$\overline{A+B} = \bar{A} \cdot \bar{B}$

The identities  $\overline{AB} = \bar{A} + \bar{B}$  and  $\overline{A+B} = \bar{A} \cdot \bar{B}$  are also called DeMorgan's Theorems and are of particular importance in logic analysis and design. The fundamental consequence of DeMorgan's theorems is that:

**Any logic function may be implemented by using only OR and NOT gates or only AND and NOT gates.**

A very important consequence of Boolean algebra is the **sum of products** or the **product of sums** representation of any Boolean expression. To illustrate the usefulness of the procedure let's consider the logic function represented by the following truth table,

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

The information in this truth table might represent the logic associated with a certain digital process that we would like to model, or it might represent the operational characteristics of a certain digital circuit that we would like to construct. The Boolean representation of the logic function is the first step in achieving a robust and efficient design.

In order to construct the **sum of products** we proceed as follows.

1. Identify the rows for which the result (Y in our case) is 1 (TRUE). For our example these are rows 2 and 3.

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

2. Use these cases to construct the sum of products by using the uncomplemented representation of the variable that has the value of 1 and the complemented representation of the variable that has the value of 0

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

$$\bar{A} \cdot B$$

$$A \cdot \bar{B}$$

And by taking the “sum of the products” we obtain:  $Y = (\bar{A} \cdot B) + (A \cdot \bar{B})$

3. The resulting Boolean function represents the entire logic sequence as can be verified by constructing the corresponding truth table.

The **product of sums** method may also be used in order to realize the Boolean representation of the logic presented by the previous truth table.

The construction of **products of sums** proceeds as follows.

1. Identify the rows for which the result (Y in our case) is 0 (FALSE). For our example these are rows 1 and 4.

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

2. Use these cases to construct the products of sums by using the uncomplemented representation of the variable that is 0 and the complemented representation of the variable that is 1

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

A + B

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

$$Y = (\overline{A} + \overline{B}) \cdot (A + B)$$

3. The resulting Boolean function represents the entire logic sequence as can be verified by constructing the corresponding truth table. Also by manipulating the expression obtained from the application of the product of sums we obtain:

$$(\overline{A} + \overline{B}) \cdot (A + B) = \quad \leftarrow \text{(product of sums)}$$

$$(\overline{A}\overline{B}) \cdot (A + B) =$$

$$(\overline{A}\overline{B}) \cdot A + (\overline{A}\overline{B}) \cdot B =$$

$$(\overline{A} + \overline{B}) \cdot A + (\overline{A} + \overline{B}) \cdot B =$$

$$(\overline{A} + \overline{B}) \cdot A + (\overline{A} + \overline{B}) \cdot B =$$

$$(\overline{A} \cdot A + \overline{B} \cdot A) + (\overline{A} \cdot B + \overline{B} \cdot B) =$$

$$(0 + \overline{B} \cdot A) + (\overline{A} \cdot B + 0) =$$

$$(\overline{B} \cdot A) + (\overline{A} \cdot B) = (A \cdot \overline{B}) + (\overline{A} \cdot B) \quad \leftarrow \text{sum of products}$$

The two methods are equivalent. The choice of which method to use is based on the details of the logic function to be implemented. For example, if we are considering a 3 input logic sequence for which the result is true (1) for 2 out of the eight ( $2^3$ ) possible cases then the reduced algebraic complexity associated with the use of the sum of products approach will be advantageous.

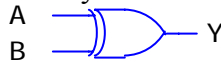
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr><th>A</th><th>B</th><th>C</th><th>Y1</th><th></th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td></td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td><td><math>\bar{A}\cdot\bar{B}\cdot C</math></td></tr> <tr><td>0</td><td>1</td><td>0</td><td>0</td><td></td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td><td></td></tr> <tr><td>1</td><td>0</td><td>0</td><td>0</td><td></td></tr> <tr><td>1</td><td>0</td><td>1</td><td>1</td><td><math>A\cdot\bar{B}\cdot C</math></td></tr> <tr><td>1</td><td>1</td><td>0</td><td>0</td><td></td></tr> <tr><td>1</td><td>1</td><td>1</td><td>0</td><td></td></tr> </tbody> </table> <p style="text-align: center;">Work with Sum of Products  <math>Y1 = (\bar{A}\cdot\bar{B}\cdot C) + (A\cdot\bar{B}\cdot C)</math></p>	A	B	C	Y1		0	0	0	0		0	0	1	1	$\bar{A}\cdot\bar{B}\cdot C$	0	1	0	0		0	1	1	0		1	0	0	0		1	0	1	1	$A\cdot\bar{B}\cdot C$	1	1	0	0		1	1	1	0		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr><th>A</th><th>B</th><th>C</th><th>Y2</th><th></th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td><math>A+B+C</math></td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td><td></td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td><td></td></tr> <tr><td>0</td><td>1</td><td>1</td><td>1</td><td></td></tr> <tr><td>1</td><td>0</td><td>0</td><td>0</td><td><math>\bar{A}+\bar{B}+\bar{C}</math></td></tr> <tr><td>1</td><td>0</td><td>1</td><td>1</td><td></td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td><td></td></tr> <tr><td>1</td><td>1</td><td>1</td><td>1</td><td></td></tr> </tbody> </table> <p style="text-align: center;">Work with Products of Sums  <math>Y2 = (A+B+C)\cdot(\bar{A}+\bar{B}+\bar{C})</math></p>	A	B	C	Y2		0	0	0	0	$A+B+C$	0	0	1	1		0	1	0	1		0	1	1	1		1	0	0	0	$\bar{A}+\bar{B}+\bar{C}$	1	0	1	1		1	1	0	1		1	1	1	1	
A	B	C	Y1																																																																																								
0	0	0	0																																																																																								
0	0	1	1	$\bar{A}\cdot\bar{B}\cdot C$																																																																																							
0	1	0	0																																																																																								
0	1	1	0																																																																																								
1	0	0	0																																																																																								
1	0	1	1	$A\cdot\bar{B}\cdot C$																																																																																							
1	1	0	0																																																																																								
1	1	1	0																																																																																								
A	B	C	Y2																																																																																								
0	0	0	0	$A+B+C$																																																																																							
0	0	1	1																																																																																								
0	1	0	1																																																																																								
0	1	1	1																																																																																								
1	0	0	0	$\bar{A}+\bar{B}+\bar{C}$																																																																																							
1	0	1	1																																																																																								
1	1	0	1																																																																																								
1	1	1	1																																																																																								

### Two additional useful gates: XOR and XNOR

The gate represented by the truth table

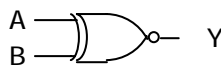
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Is very useful and it is called Exclusive OR (XOR). The symbol of this gate is



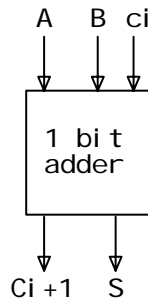
Another useful gate is the Exclusive NOR gate (XNOR). The truth table and the symbol for this gate are

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1



For the laboratory

As an example lets design an adder which is able to add two 1 bit digital numbers. This is called a 1 bit adder and schematically has the form



A and B are the digits to be added. Ci is the carry from the previous addition and Ci+1 is the carry after the addition. The basic rules for binary addition are:

- 1 + 1 = 0, Carry = 1
- 1 + 0 = 1, Carry = 0
- 0 + 1 = 1, Carry = 0
- 0 + 0 = 0, Carry = 0

The first thing to do is construct the truth table for this procedure.

A	B	Ci	S	Ci+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

As part of your laboratory you will design and then built this 1 bit full adder circuit using fundamental gates.

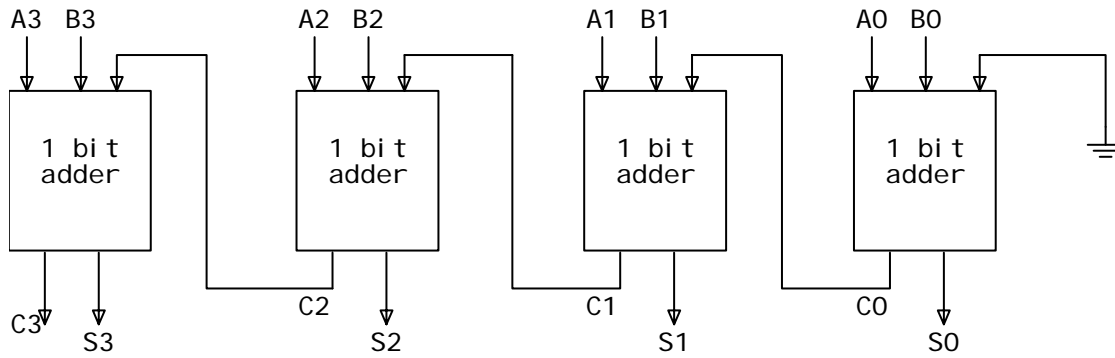
Download the instrument called **adder.vi** from the class web site. Take a minute to familiarize yourself with the instrument. Your goal is to construct the circuit by using the



gates provided and then run it to see the results. You may also view the results on the ELVIS leds by wiring DO0 - DO4 to LED0 - LED4.

Try simplifying your circuit by using the XOR or the XNOR gates.

With this as your basic 1 bit adder circuit you will then built and test an adder circuit for two 4 bit binary numbers. The structure of the circuit is



This circuit will add numbers  $A_0A_1A_2A_3$  and  $B_0B_1B_2B_3$  by propagating the carry bit