7 PRACTICAL EXERCISES MANUAL

7.1 INTRODUCTION

This equipment is made up of a practice board and a practice handbook. This will teach you the basic concepts of combinational circuits of digital electronics, while, at the same time, we will ask you to carry out a series of assemblies on the mentioned practice board to reaffirm your studies.

7.1.1 Use of this Handbook

This handbook describes how to use the board M-12 correctly to carry out a series of exercises. Furthermore, each chapter contains a theoretical compendium on the subject being studied.

In addition, each chapter will ask you to answer several questions on the circuit built.

These questions are the same as those that appear in the computer and which the student will have to answer in the *software* version. You will recognise them by the symbol that appears beside each question.

Nevertheless, these questions can be answered on the photocopies from the last pages of this handbook appendix.

7.1.2 Use of the Practice Board M-12

To use this board correctly it is necessary to use the following accessories:

Set of connecting leads (Supplied with the equipment).

Set of links (Supplied with the equipment. The *bridge* is a piece with two pins covered in plastic and joined internally).

Multimeter.

Base Unit EBC-100 or power supply of 5v.

The board M-12 can be used with the box belonging to the complete equipment, or by itself, just like all the EBC modules.

When doing the exercises, be careful with the leads when handling them, since their terminals may provoke involuntary short circuits (remember that they are conductors) in the components on the board. An example would be if you touch two pins of an integrated circuit with the same end of a lead.

7.1.3 Power Supply

7.1.3.1 Operation with the Base Unit EBC-100

To make the practice board M-12 operates independently of the base unit you will need a 5 volts power supply and two leads with 2 mm. Pins.

Make sure that the power supply switch is turned off, and that there are no links or leads between the test points on the board.

Put the pins of the power supply leads in the two holes in the upper right hand part of the board. These two connection points are labelled as *5v* and *GND*. It is very important that the polarity is observed.

In all the exercises it will be assumed that you have the base unit EBC-100. If this is not the case, when you read "feed the board by pressing on the switch of the base board", do the same with the power supply that you are using.

7.1.4 Faults Study

In most of the circuits to be studied, faults can be simulated. The student must investigate what is happening in the circuit and why it does not operate correctly. These failure simulations can be of several types: from faulty components to an incorrect circuit assembly.

These faults are simulated through some switches at the side of each circuit. There are two positions: one position is for normal operation (an *N* printed on

the board) and the other is the fault one (an *F* printed on the board).

In the board M-12, the normal operation in all the circuits is with the switches downward, therefore, when you start to work, check that all the switches are downwards.

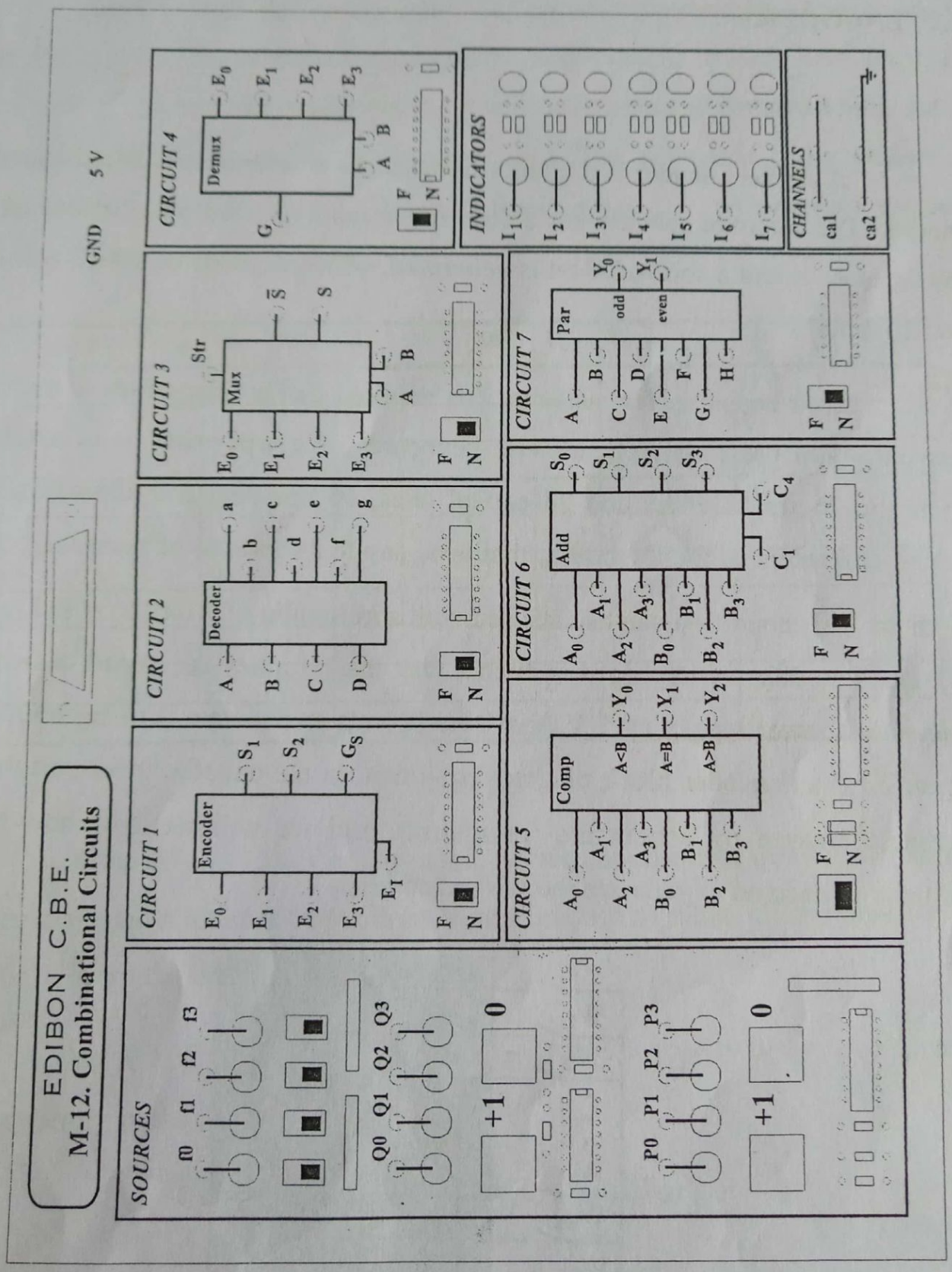


Figure 1.4.1: Aspect of the board M-12

7.2 ENCODERS

7.2.1 The Encoder

A digital encoder is a circuit that changes a determinate digital signal into another. The common encoder has a determined input number, and just one of them has the state 1, and a code of *N* bit is *generated*, which depends on which will be the excited input.

Let's consider, for instance, it is required that a binary code is transmitted each time you press an alphanumeric keyboard (of a typewriter or of a teletype). There are 26 small letters and 26 capital ones, 10 numbers and approximately 22 special characters in the keyboard, that is to say that the total of necessary codes is about 84. This condition can be satisfied with a minimum of 7 bit ($2^7 = 128$, but $2^5 = 64$). Let's modify the keyboard in such a way that, if a key is pressed, a switch that connects a power supply of 5 V (that corresponds to the state l) to an input line. In figure 2.1.1 an encoder block diagram is shown. In the encoder inner part there is a rectangle crossed by conductors (or matrix), and we will see now how they are mutually connected to generate the wished codes.

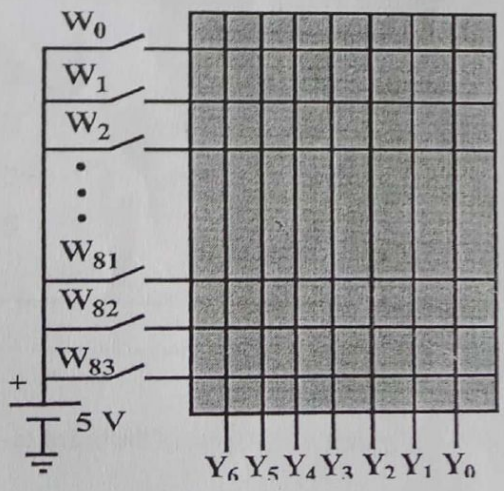


Figure 2.1.1.

To illustrate the procedure of an encoder construction design, we will simplify the previous example limiting the keyboard just to 10 keys, numbers 0, 1, 2, ...9. In this case, an output code with 4 bit will be enough and because of this we will choose the BCD system for the output codes. The table that really defines this encoder can be seen in table 1.1. The input W_n (n = 0, 1, . . . , 9) represents the key n.

W_9	W_8	W_7	W_6	W_5	W_4	W_3	W_2	W_1	W_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

Table 2.1.1.

When $W_n = 1$, key n is pressed. As we suppose that there is no more than an key activated simultaneously, then at any line of the table, all the inputs but one will be 0. From the table it is deduced that

$Y_0 = 1$ if $W_1 = 1$ or if $W_3 = 1$ or if $W_5 = 1$ or if $W_7 = 1$ or if $W_9 = 1$.

Consequently in the Boolean notation:

$$Y_0 = W_1 + W_3 + W_5 + W_7 + W_9$$

In an similar way,

$$Y_1 = W_2 + W_3 + W_6 + W_7$$

$$Y_2 = W_4 + W_5 + W_6 + W_7$$

$$Y_3 = W_8 + W_9$$

The gates OR of the previous equations are made up with diodes, as it is shown in figure 2.1.2. An encoder arrangement as the one in figure 2.1.2 is called *rectangular matrix of diodes.*

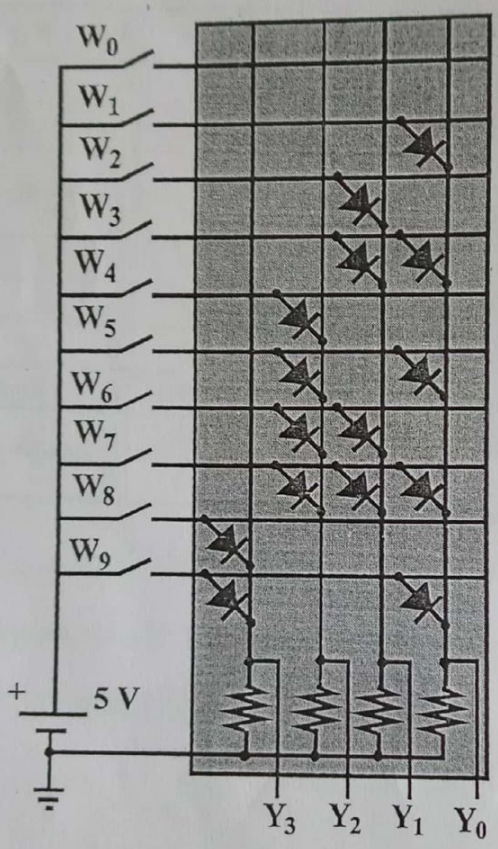


Figure 2.1.2.

Each encoder diode in figure 2.1.2 can be substituted by a transistor base-emitter diode. If the collector is joined to the supply voltage V_{cc}, then a gate to emitter follower is obtained. This structure is drawn in figure 2.1.3 for the output Y_2. Notice that if any of W_4, W_5, W_6, or W_7 is in a high state, the emitter follower output will be also high, and this fulfils that:

$$Y_2 = W_4 + W_5 + W_6 + W_7$$

as the previous equation demands.

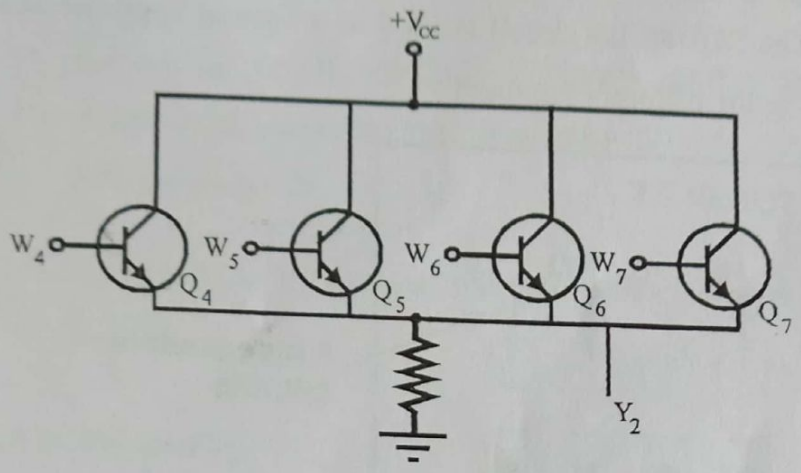


Figure 2.1.3.

It is required just a transistor (with multiple emitter) by each encoder input. The base is joined to the input line, and each emitter is connected to a different output line, according to the encoder logic. For instance, as the line W7 in figure 2.1.2 is joined to three diodes whose cathodes go to Y_0, Y_1 and Y_2, this combination can be substituted by a transistor with three emitters Q7 connected as in figure 2.1.4. The maximum number of emitters that can be needed is equal to the number of bit of the output code. For the particular encoder drawn in figure 2.1.2., *Q1, Q2, Q4* and *Q8* have an emitter each one, *Q3, Q5, Q6* and *Q9* have two emitters and *Q7* has three.

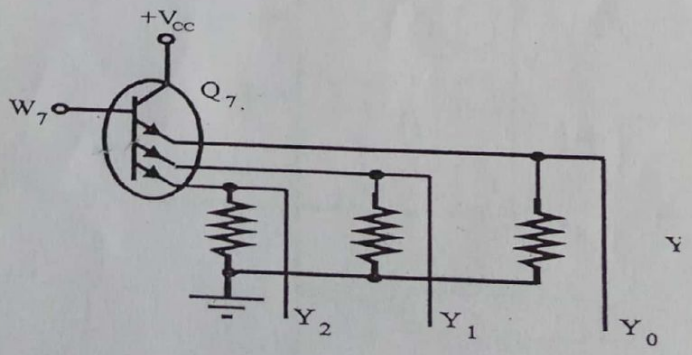


Figure 2.1.4.

7.2.2 Practice 1: Study of an Encoder

7.2.2.1 SOURCES Circuit

The SOURCES circuit is used as a base to apply its signals to the rest of the circuits in the different practices.

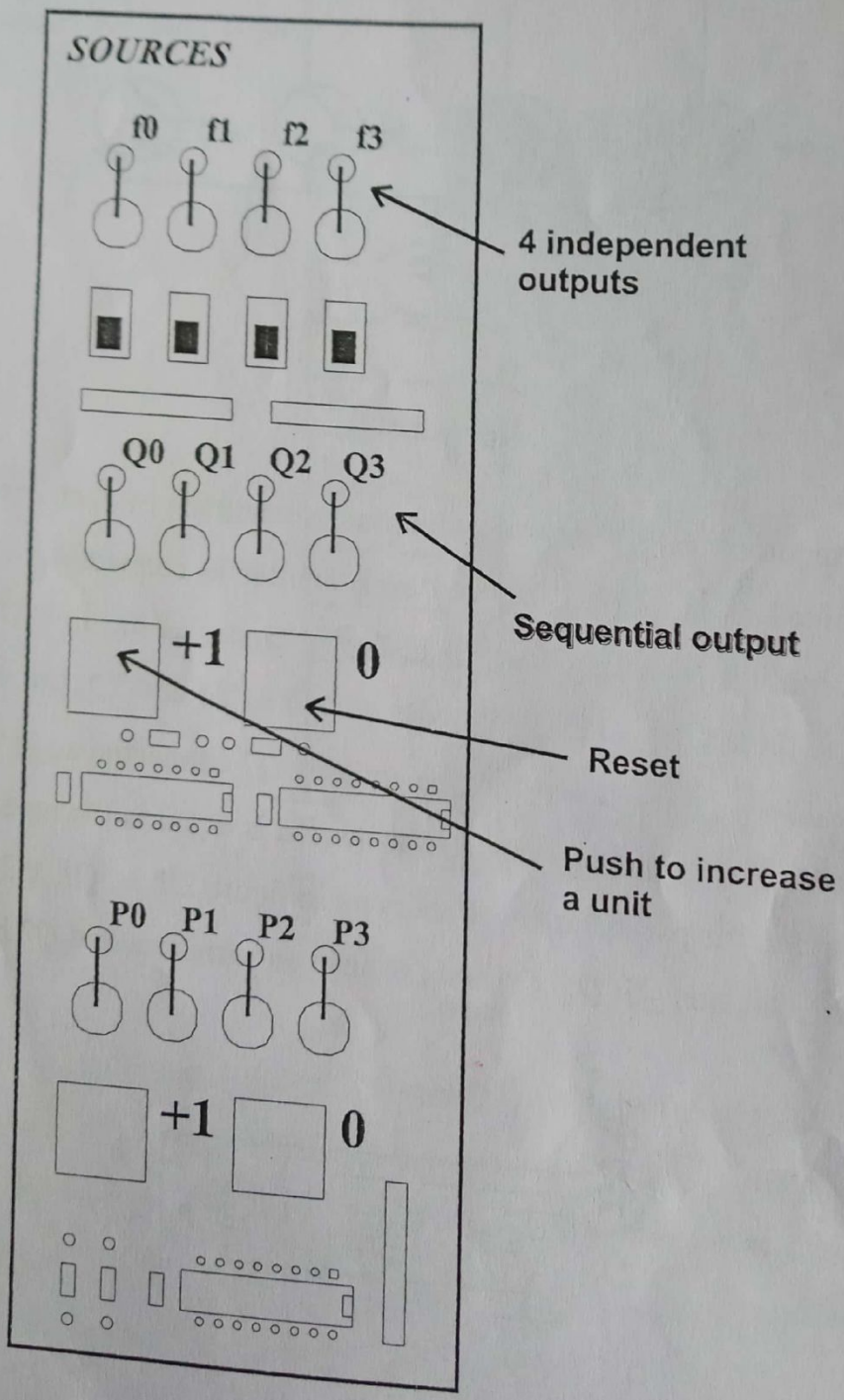


Figure 2.2.1

It consist of three groups of signals:

- Signals from f_0 to f_3 can be put to the logic value 1 or logic value 0 through the switches placed under the output females.
- Signals Q_0 to Q_3 represent a binary number that it can be increased in 1 with the push labelled with <+1>, or be set to zero with the push <0>.
- Signals P_0 to P_3 have an operation just the same as the group $Q_0...Q_3$.

7.2.2.2 INDICATORS Circuit

The INDICATORS circuit is used to verify the logical state of a signal. To do it, touch with a lead connected to the signal to check any of the points I1 to I7. If the LED diode is lighted, that signal state is a logic 1; otherwise it is a logic 0.

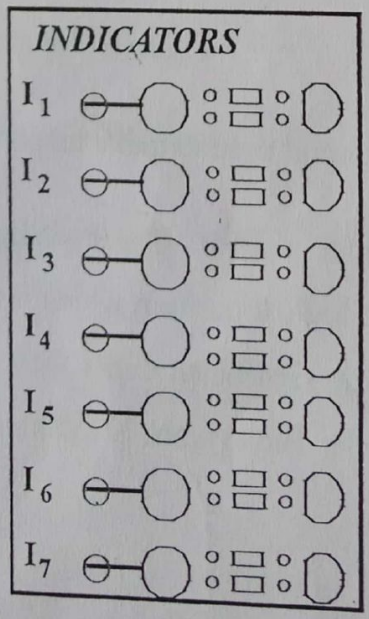


Figure 2.2.2

7.2.2.3 Circuit 1

The circuit to study is the one silk-printed as *Circuit 1*, which is in figure 2.2.3.

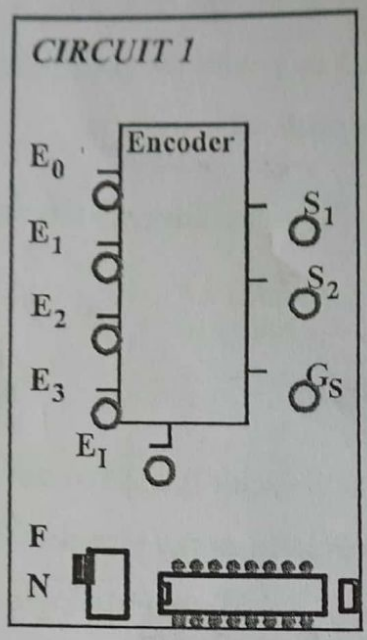


Figure 2.2.3

The integrated circuit used to make an encoder is 74LS148.

It is not a "normal" encoder. An input passes to a priority binary number. Its purpose is to detect the logic 0 position which it is in the most significant position, in such a way that the most significant digits than it are at "1". Its truth table is the . The symbol X indicates that it can take any value: 0 or 1.

E_I	E_3	E_2	E_1	E_0	S_2	S_1	G_S
0	0	X	X	X			
0	1	0	X	X			
0	1	1	0	X			
0	1	1	1	0			
0	1	1	1	1			
1	X	X	X	X		1	1

Table 2.2.1

The integrated circuit 74LS148 has 8 inputs and 5 outputs. For its study, the ones that are seen on the board have been selected.

The signal EI serves to enable the chip operation. This is a signal that is in most of the integrated circuits. In this case, the chip operates correctly when this signal is at a low level.

7.2.2.4 Practice Carrying out

Remember that for all the circuits to operate correctly, the fault switches should be at their position *N*, that is to say, downward.

To accomplish the practice, make the following assembly (figure 2.2.4).

Connect the four independent signals from SOURCES circuit to the inputs E0 to E3.

Connect the input EI to "0" using any of the other SOURCES circuit outputs.

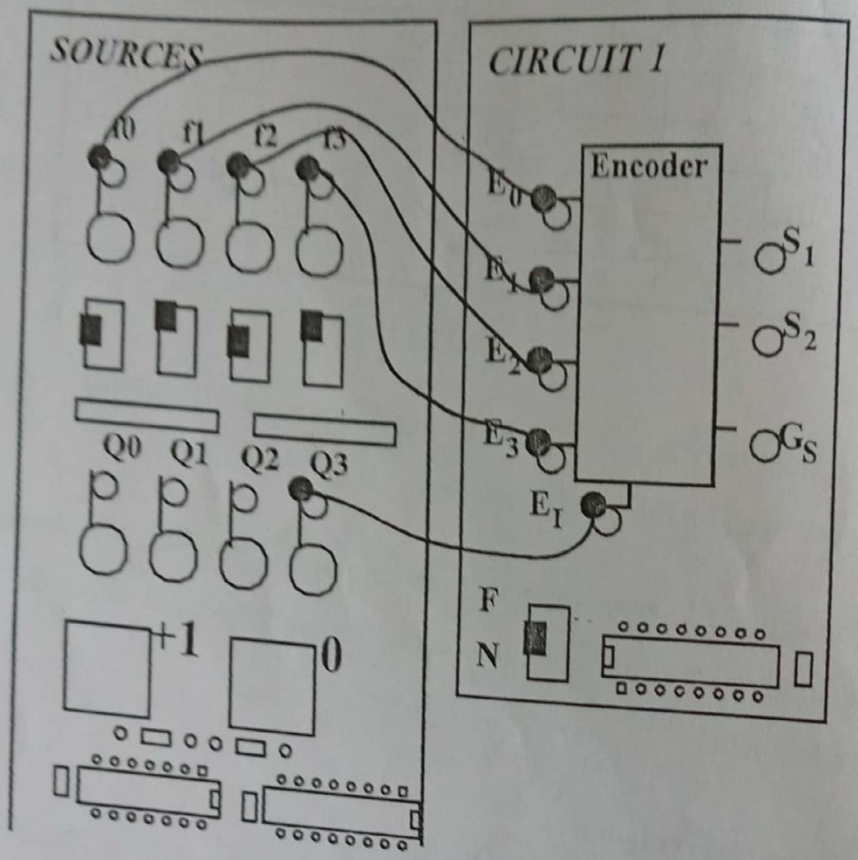


Figure 2.2.4.

Fill in table 2.2.1 getting the results S2, S1 and GS from the practice board.

E_I	E_3	E_2	E_1	E_0	S_2	S_1	G_S
0	0	X	X	X			
0	1	0	X	X			
0	1	1	0	X			
0	1	1	1	0			
1	0	X	X	X			
1	1	0	X	X			
1	1	1	0	X			
1	1	1	1	0			
1	1	1	1	1			

Table 2.2.1

C1.1. Which are the outputs value the following case?

E_I	E_3	E_2	E_1	E_0
0	1	1	1	1

C1.2. Which are the outputs value in the following case?

E_I	E_3	E_2	E_1	E_0
1	1	1	1	1

C1.3. Which are the outputs value in the following case?

E_I	E_3	E_2	E_1	E_0
0	1	0	0	1

C1.4. What means that the output GS signal is at 1?

7.2.3 Practice 2: Study of the Fault in the Encoder

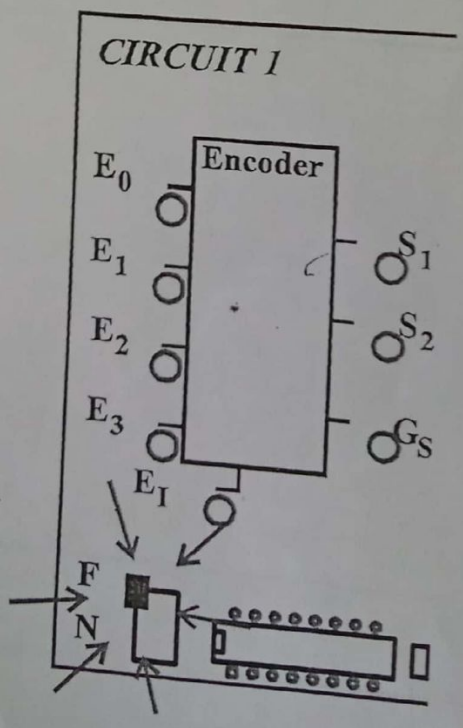


Figure 2.3.1

Set the fault switch in fault position (F).

In CIRCUIT 1 there is now a fault that you have to find. Notice the diagram drawn on the board and try to search what does not correspond with it.

To discover what does not operate correctly, we suggest filling in table 2.3.1, and comparing it with 2.2.1.

E_I	E_3	E_2	E_1	E_0	S_2	S_1	G_S
0	0	X	X	X			
0	1	0	X	X			
0	1	1	0	X			
0	1	1	1	0			
0	1	1	1	1			
1	0	X	X	X			
1	1	0	X	X			
1	1	1	0	X			
1	1	1	1	0			
1	1	1	1	1			

Table 2.3.1


C1.5. What do you think the anomalous operation is due to?

7.2.4 Practice 3: Exercises

These exercises can only be made with *software* version included with this board.

7.3 DECODERS

7.3.1 The Decoder

In a digital system, both instructions and numbers can be transmitted through binary levels or impulse trains. If, for instance, the 4 bits of a message are prepared to transmit orders, 16 different instructions can be obtained. This information is *codified* in **binary** system. Many times, switches are needed having several positions that could operate according to this code. In other words, for each one of 16 codes, it is very important that just one line has to be excited. This process of particular code identification is called *decoding*.

7.3.1.1 Decimal binary coding system (BCD)

This code translates a decimal number substituting each figure by a 4 binary figure combination. As there are 16 different possibilities for 4 binary figures, 10 combinations will serve to represent the decimal figures from 0 to 9. Therefore, we have a wide availability of BCD codes. One of them, called "natural decimal binary code", is the code 8421 indicated by 10 first inputs of table 3.1.1. This is a weighted code because its decimal digits are equal to the sum of the bit in series products codified by the successive powers of two, beginning from the right (LSB). We need 4 N bit to represent a number of N decimal figures in BCD notation. The first 4 bit, set on the right, represents the units, the second one represents the tens, the third one, the hundreds, and so on. For instance, the decimal number 264 require 3 groups of 4 bit, as table 3.1.1 indicates. Take into account that these three figures from the BCD code can represent any number between 0 and 999; therefore, it has a resolution of 1 part by 1 000, that is to say, 0'1 %. This requires 12 bit, what in a normal binary code can solve 1 part in $2^{12} = 4096$, it is 0'025 %.

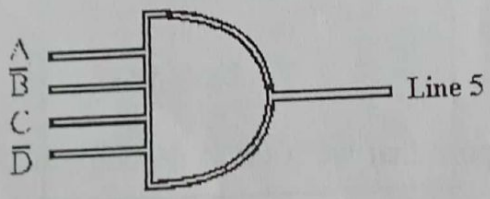
Weighted factor	800	400	200	100	80	40	20	10	8	4	2	1
BCD Code	0	0	1	1	0	1	1	1	1	0	0	1
decimal digits	3				7				9			

Table 3.1.1

7.3.2 Decoder from BCD to Decimal

Let's suppose that we want to decode a BCD instruction that represents a decimal number of a figure, for instance 5. This operation can be carried out with a gate Y with four inputs, which is excited by the four BCD bit. For instance, the output of the gate AND in figure 3.2.1 is exclusively 1 if the BCD inputs are A = 1 (LSB), B = 0, C = 1 and D = 0. Since this code represents the decimal number 5, the output is indicated "line 5".

Figure 3.2.1



A decoder BCD to decimal would be made with 10 circuits similar to the one in figure 3.2.1, one for each output line. Each one of these circuits would have four inputs, *A, B, C and D* and an output. Furthermore, with the ground and feed connections, a 16-pin encapsulation will be needed. The complementary inputs $\overline{A}$, $\overline{B}$, $\overline{C}$, y $\overline{D}$ can be obtained with inverters in the own chip. The system is also called "decoder 4 to 10", indicating that a 4 bit input selects 1 of the 10 output lines. In other words, the decoder acts as a switch of 10 positions corresponding with the instructions of a BCD input.

7.3.2.1 BCD decoder to 7 segments code

It is usual to make a digital instrument reading visible (frequency meter, digital voltmeter, etc.) through a numerical indicator of seven segments as the one indicated in figure 3.2.2. A static indicator whose segments obtain the lighting from some luminescent diodes works at small voltage and with small power, and consequently it can be excited directly from integrated logic gates.

The first ten images in figure 3.2.2 are the figures from 0 to 9, which in the digital instrument, are represented as BCD. A 4-bit code has 16 possible states, and the images from 10 to 15 are symbols used to identify a not valid BCD condition.

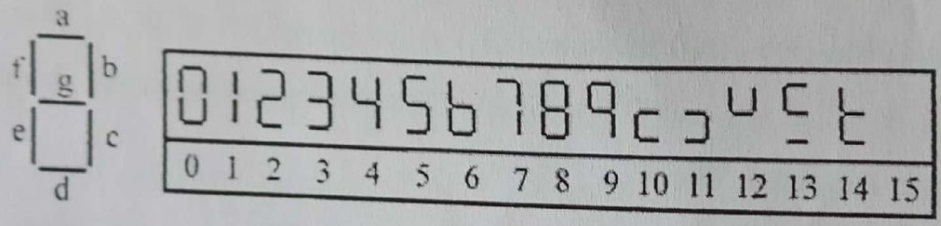


Figure 3.2.2.

Let's suppose that we want to decode a BCD instruction that represents a decimal number of one figure in the corresponding 7 segments code. This operation can be carried out in several ways. One of them consists in making 7 logic circuits (one for each indicator bar) with AND gates with four input excited by the four BCD bit.

For instance, to make the circuit in charge of illuminating the indicator bar <e>, we see that this has to *be lit* when the input number is 0, 2, 6, 8, 10 or 14. This means that the logic function has to be the following:

$$e = \overline{D}\,\overline{C}\,\overline{B}\,\overline{A} + \overline{D}\,\overline{C}B\overline{A} + \overline{D}C\overline{B}\,\overline{A} + D\overline{C}\,\overline{B}\,\overline{A} + D\overline{C}B\overline{A} + DC\overline{B}\,\overline{A}$$

Of course, this expression can be simplified with some logic algebraic methods in order to minimise the number of gates that should be used.

7.3.3 Practice 4: Study of a decoder

7.3.3.1 Circuit 2

The circuit to study is printed as *Circuit 2*, represented in figure 3.3.1

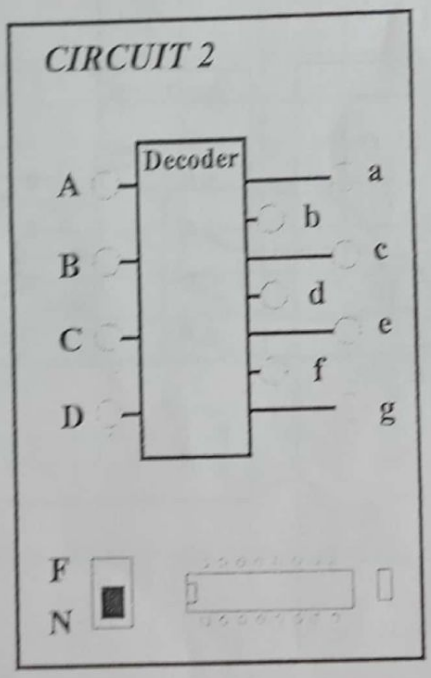


Figure 3.3.1

The integrated circuit used to make the decoder is 74LS48.

7.3.3.2 Practice development

Remember that for all the circuits to operate correctly, all the fault switches should be in their position *N*, that is to say, downward.

To carry out this practice make the following assembly (figure 3.3.2):

Connect the four signals from the counter circuit (Q0 to Q3) to the inputs A, B, C and D.

Connect the output signals to the seven indicator channels.

NOTE: remember that the digital circuits should have all the inputs connected to a logic state of 1 or 0 so that the output is reliable.

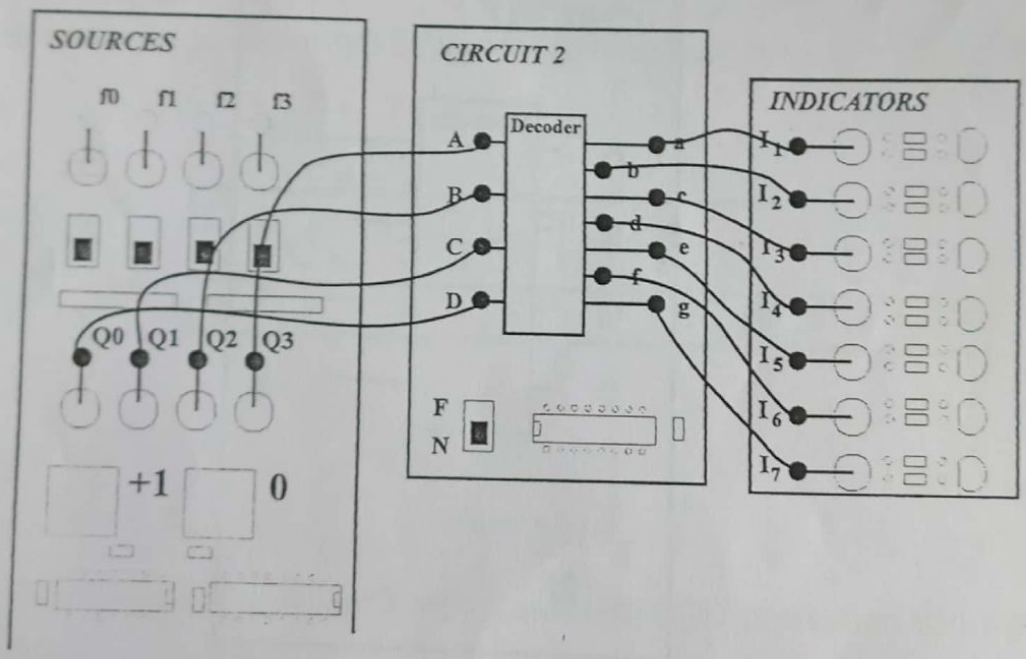


Figure 3.3.2

Connect the circuit by pressing on the switch of the base unit EBC-100.

Fill in table 3.3.1, obtaining the results from the practice board.

D	C	B	A	a	b	C	d	e	f	g
0	0	0	0							
0	0	0	1							
0	0	1	0							
0	0	1	1							
0	1	0	0							
0	1	0	1							
0	1	1	0							
0	1	1	1							
1	0	0	0							
1	0	0	1							
1	0	1	0							
1	0	1	1							
1	1	0	0							
1	1	0	1							
1	1	1	0							
1	1	1	1							

Table 3.3.1.

C2.1. What have you noticed in the previous table that shows a non-anticipated result?

The cause of the previous result is the integrated circuit 74LS48, that has a pin called RBI in such way that if the voltage connected to it is 0V, a zero number

input in BCD will not produce the lighting of any LED. If in the input RBI we set a "1", then the output with a zero input in BCD will be the one shown in figure 3.2.2.

C2.2. What is the value of the bar <f> output in the following case?

D	C	B	A
0	1	1	1

C2.3. What is the value of the bar <a> output in the following case?

D	C	B	A
0	1	0	1

C2.4. Which input is diode <b> lighted with?

C2.5. Which input is diode <g> lighted with?

7.3.4 Practice 5: Study of the Fault in a Decoder

Set the fault switch in failure position (F).

In CIRCUIT 2 there is now a failure that you have to find. Observe the diagram drawn on the board and try to see what does not correspond with it.

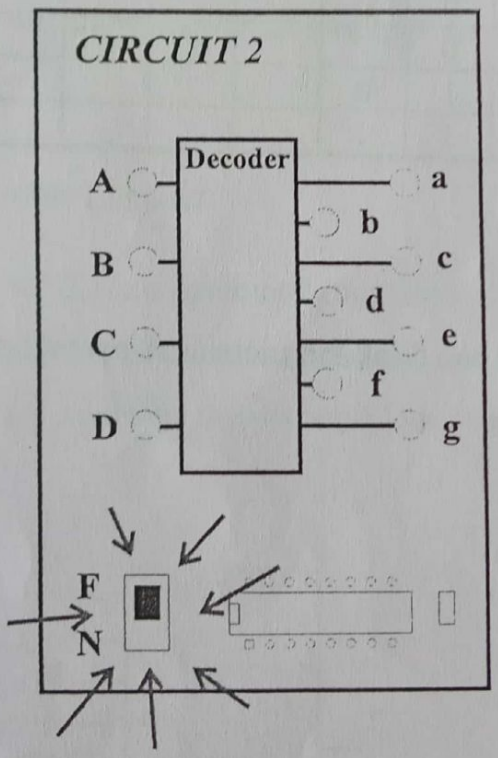


Figure 3.4.1

To find what does not operate correctly, we suggest to you to fill in table 2.3.1, and compare it with 3.3.1.

D	C	B	A	a	b	c	d	e	f	g
0	0	0	0							
0	0	0	1							
0	0	1	0							
0	0	1	1							
0	1	0	0							
0	1	0	1							
0	1	1	0							
0	1	1	1							
1	0	0	0							
1	0	0	1							
1	0	1	0							
1	0	1	1							
1	1	0	0							
1	1	0	1							
1	1	1	0							
1	1	1	1							

Table 2.3.1

C2.6. What do you think the anomalous operation is due to?

7.3.5 Practice 6: Exercises

These exercises can only be made with *software* version included with this board.

7.4 MULTIPLEXERS

7.4.1 The Multiplexer

The multiplexers (or data selectors) are combined circuits that select one of several possible input lines and direct the datum located in that line (a 1 or a 0) to a single output line. In this sense, a multiplexer operates in a similar way to a switch with several positions.

Multiplexers are used in data transmission systems, event sequencers, logic function generators and other applications. In this chapter we shall study their basic aspects, paying special attention to the description of the most common MSI multiplexers.

7.4.1.1 What is a Multiplexer?

A multiplexer (MUX) is a combined logic circuit with a certain number of input lines (M), a certain number of selection lines (N) and an output line that directs information from one of the inputs to the output in accordance with code present in the selection lines.

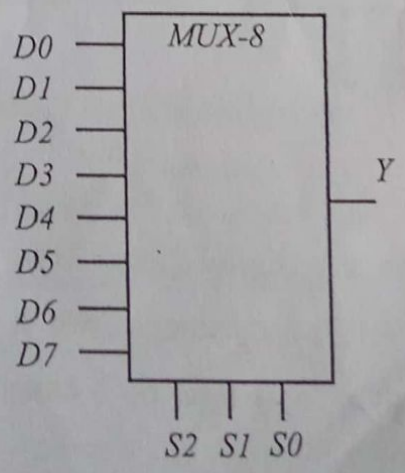


Figure 4.1.1

For example, if the code $S_2 S_1 S_0$ =011 (that is to say, 3 in decimal figures) is applied to the selection lines of the multiplexer of figure 4.1.1, the information available in the input D3 will appear in the output Y. If the code 110 is applied, the datum of the line D6 will be reflected, and successively. The input information can be a 1, a 0, a pulse chain or any digital signal.

The input lines of a multiplexer are usually called channels. There are multiplexers of 2, 4, 8, 16 and more channels. As a rule, with N selection lines, it is possible to handle or direct up to $M=2^n$ channels. For example, if N=4, then M=24=16 channels.

The operation of a multiplexer is similar to that of a switch with several positions. This concept is shown in figure 4.1.2 with the electromechanical equivalent of an MUX of 4 channels. Depending on the position of the axis, the output terminal is connected to any of the input terminals and transfers the information present in that point.

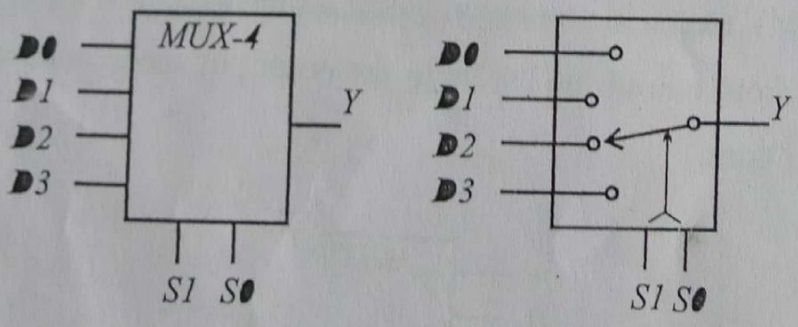


Figure 4.1.2

A multiplexer, like any digital circuit, can be implemented using small-scale gates (SSI). In figure 4.1.3, for example, we show the truth table, the design equation and the logic circuit of an MUX of 2 channels built with AND, OR and NOT gates.

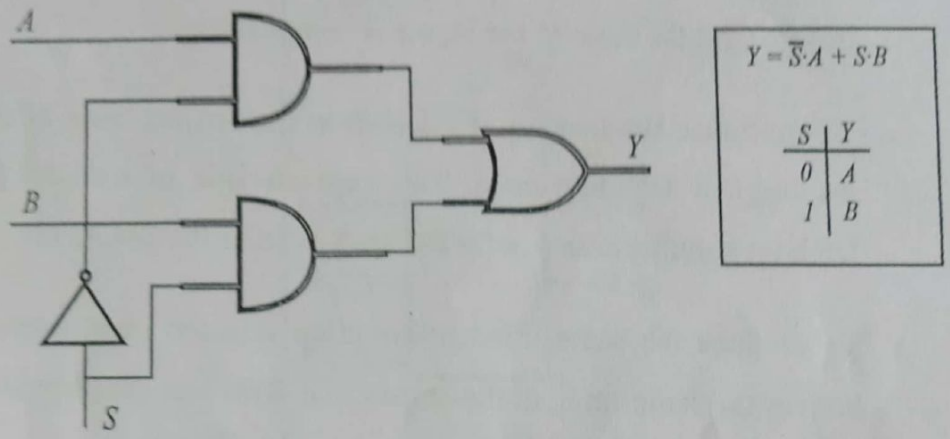


Figure 4.1.3

When the number of inputs is increased or additional characteristics are introduced (for example, qualification lines), the SSI multiplexers become increasingly complex to design. For this reason, in most cases, it is preferable to use integrated circuits of medium scale integration (MSI), designed specifically for this function.

In these circuits, as also happens with the other integrated MSI circuits, the manufacturer has already done all the work of design and of gate interconnections. Using MSI multiplexers saves time, space, money and effort and is reliable, versatile, saves power and many other advantages.

7.4.1.2 Medium Scale Integrated Multiplexers

As we shall see, several of these devices, in addition to their basic function (directing one of the inputs toward a digitally selected output), provide other auxiliary functions, such as:

- **To digitally permit or inhibit data selection**. This characteristic belongs to multiplexers with a qualification line, such as the 74LS157.

In the inhibition state, all the outputs are high or low, depending on the design, and the state of the inputs is insignificant.

- **To maintain the last input selected in the output**, even after changing the original selection code. This characteristic of memory belongs to latch type multiplexers (with storage), such as the 74LS298.
- To produce the same input information selected or its complement or both at the same time, in the output line. This last characteristic belongs to multiplexers with two outputs, such as the 74LS151.
- To locate the outputs in a high impedance state according to the state of a special control line intended for this purpose. This characteristic belongs to the three-state multiplexers, such as the 4512.

7.4.2 Practice 7: Study of a Multiplexer

7.4.2.1 Circuit 3

The circuit to be studied is the one that has been silk-screen printed as *Circuit 3*, and which is represented in figure 4.2.1.

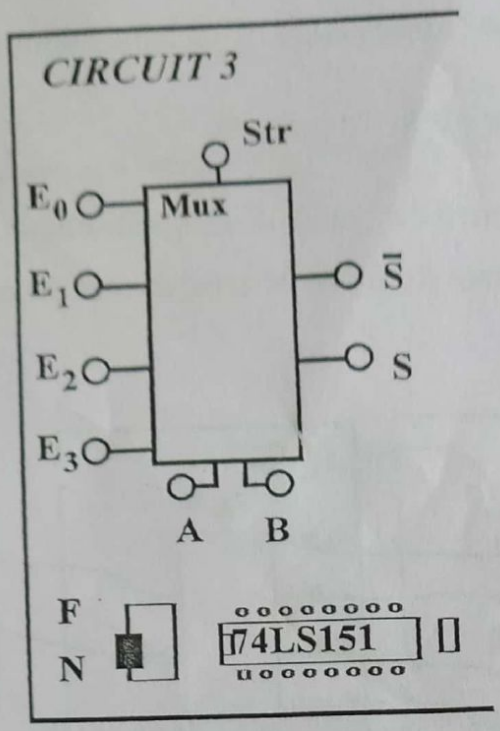


Figure 4.2.1

The integrated circuit used to make the multiplexer is 74LS151. Although this integrated circuit is a multiplexer of 8 channels, we shall work with it as if it had 4.

7.4.2.2 Carrying out the Practice

Remember that for all the circuits to operate correctly, the fault switch must be in the position N, that is to say downward.

To carry out the practice make the following assembly (figure 4.2.2):

Connect the four signals of the counter circuit (Q0 to Q3) in the inputs E0 to E3.

Connect the two output signals in the indicator channels.

Connect the two selection lines in the points f0 and f1.

Join the line Str to the point f3.

NOTE: remember that the digital circuit should have all its inputs connected to a logic state from one or zero, so that the output is reliable.

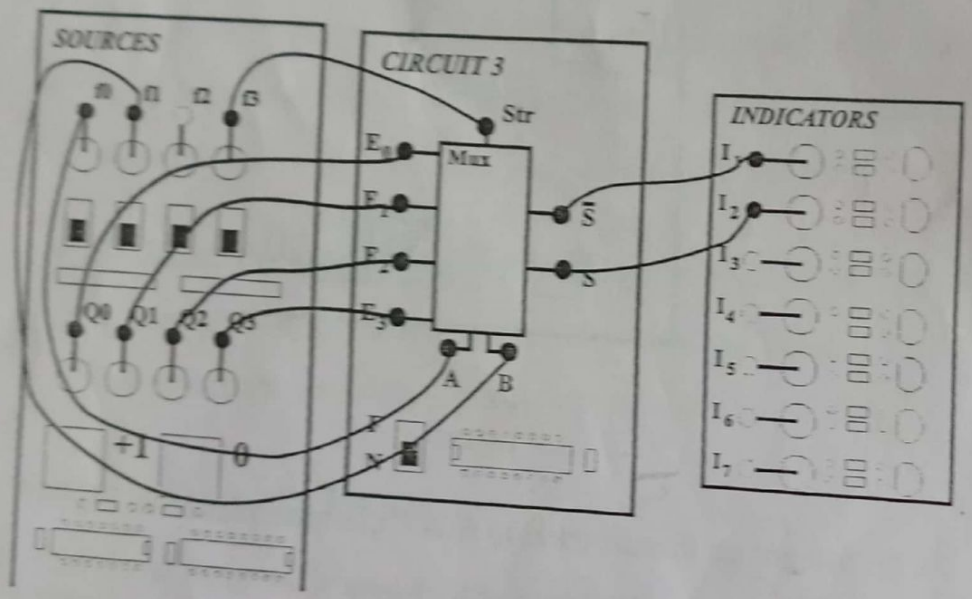


Figure 4.2.2

Feed the circuit by pressing the switch of the base unit EBC-100.

C3.1. Which are the outputs when Str=1, A=0 and B=0?

 C3.2. Which are the outputs when Str=0, A=0 and B=0?

Fill in table 3.1 with the results obtained in the practice board.

B	A	Str	S	S´
X	X	1		
0	0	0		‘
0	1	0		‘
1	0	0		‘
1	1	0		‘

Table 4.2.1

 C3.3. Which outputs are obtained when B=0, A=1, Str=0?

C3.4. If we want to obtain the value of input E3 in the output, what values should the lines A, B and Str have?

7.4.3 Practice 8: Study of the Error in the Multiplexers

Set the fault switch in the fault position (F).

There is now a fault that must be discovered in CIRCUIT 3.

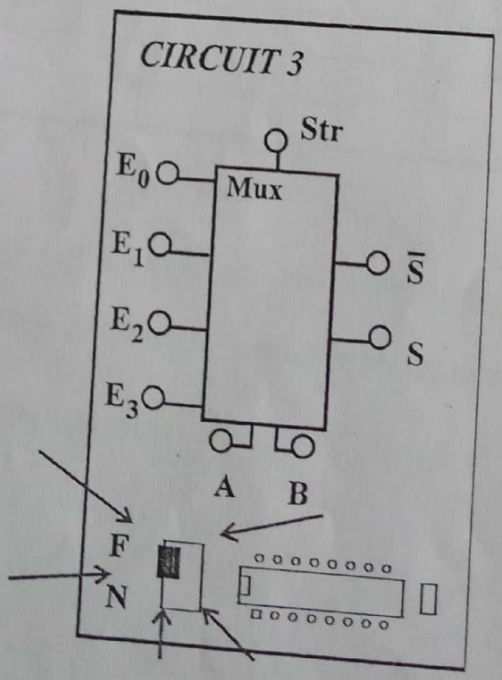


Figure 4.3.1

B	A	Str	S	S´
X	X	1		
0	0	0		‘
0	1	0		‘
1	0	0		‘
1	1	0		‘

Table 4.3.1.

To discover what does not operate correctly, we suggest that you fill in table 4.3.1 and compare it with 4.2.1.

 C3.5. What do you think is the cause of the fault?

7.4.4 Practice 9: Exercises

These exercises can only be made with *software* version included with this board.

7.5 DEMULTIPLEXERS

7.5.1 The Demultiplexer

A demultiplexer (DEMUX) is a combinational logic circuit (that is to say, its output only depends on the inputs) with an input line (G), a certain number of selection lines (N) and a certain number of output lines (M) or ways. According to a code applied to the selection lines, it transfers the datum available in the input to one of the outputs.

In other words, a demultiplexer carries out the opposite function to a multiplexer. For example, if the code CBA=011 (that is to say, 3 in decimal numbers) is applied to the selection lines of the DEMUX of figure 5.1.1, the datum (a 0 or a 1) will appear in the output Y3.

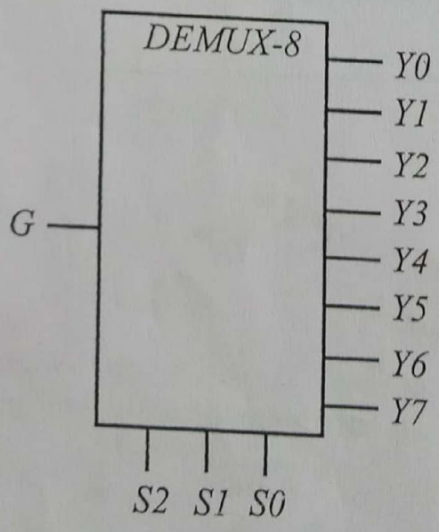


Figure 5.1.1

A demultiplexer can also be used as a decoder, sending the input line at a high or low level, depending on the design, and using the selection lines to supply the

input codes. In figure 5.1.2, the way to use the above DEMUX as a "1 of 8" decoder with high active outputs is shown.

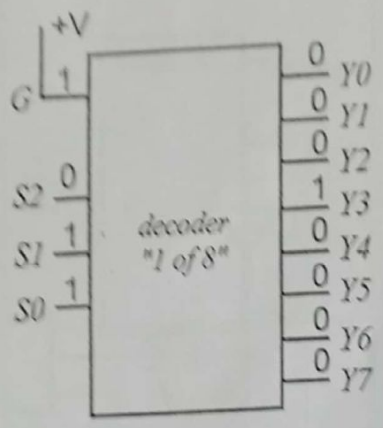


Figure 5.1.2

For example, if the code S_2 S_1 S_0 =011 is applied to the selection lines of the demultiplexer of figure 5.1.2, the information present in the input G (1) will appear in the output Y3.

In the same way, a decoder can be used as a demultiplexer by using the code inputs as selection lines and the qualification line as data input.

The operation of a multiplexer is similar to that of a rotary switch with several positions. Depending on the position of the selector axis, the common input terminal can be connected to any of the output terminals and transfer the information.

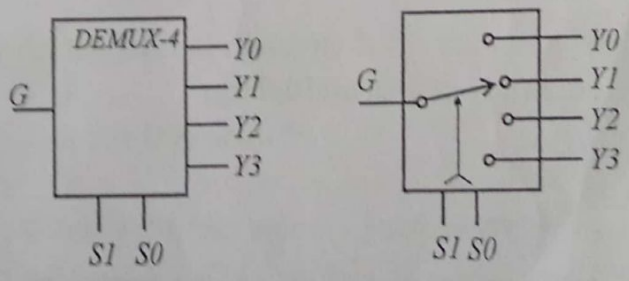


Figure 5.1.3

The demultiplexers can be built using SSI or small-scale gates. In figure 5.1.4, for example, the logic circuit of a multiplexer is shown with 4 outputs made with AND and NOT gates. The input G is transferred without inversion to the selected output.

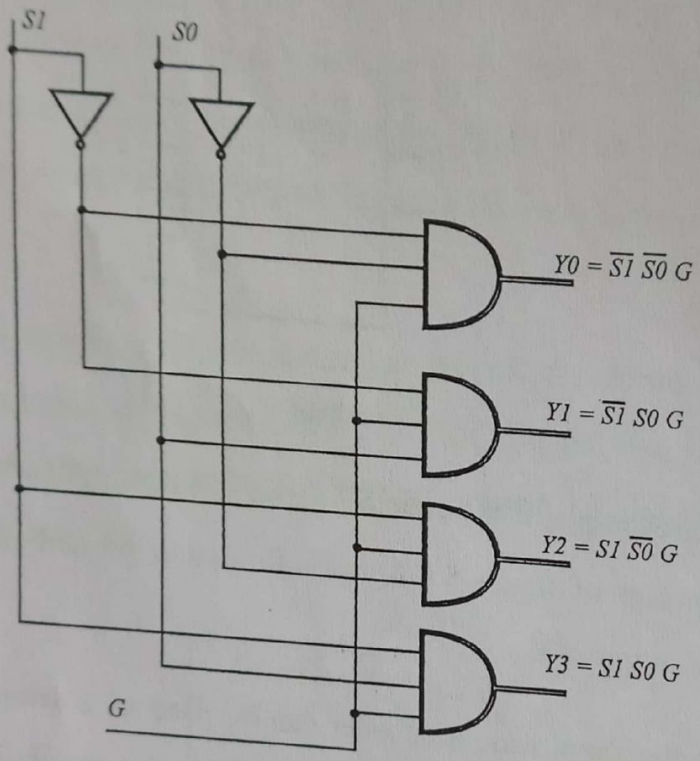


Figure 5.1.4

Although the SSI demultiplexers are very simple to design, in most cases, the medium scale integrated circuits (MSI) are preferred, as they are designed specifically for this mission.

7.5.2 Practice 10: Study of a Demultiplexer

7.5.2.1 Circuit 4

The circuit being studied is the one printed as *Circuit 4*, and which is

represented in figure 5.2.1.

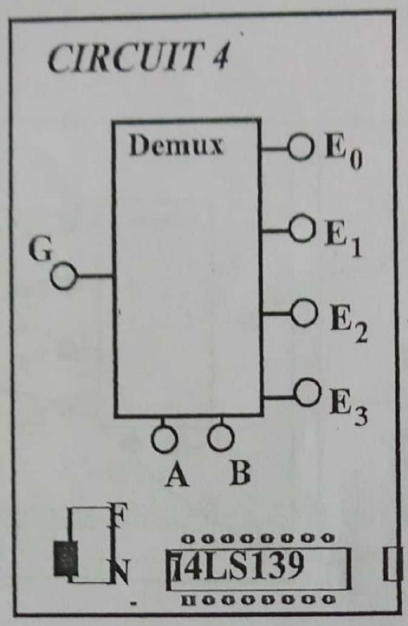


Figure 5.2.1

The integrated circuit used to make the multiplexer is 74LS139. This integrated circuit contains two multiplexers of 4 channels, though we shall only work with one.

7.5.2.2 Carrying out the Practice

Remember that for a proper operation of all the circuits, the fault switch must be in the position N, that is to say downward.

To carry out the practice make the following assembly (figure 5.2.2).

- Join line G to point f3.
- Connect the four output signals to the indicator channels.
- Connect the two selection lines to the points f0 and f1.

NOTE: remember that the digital circuits should all have the inputs connected to a logic state of one or zero, this way the output will be reliable.

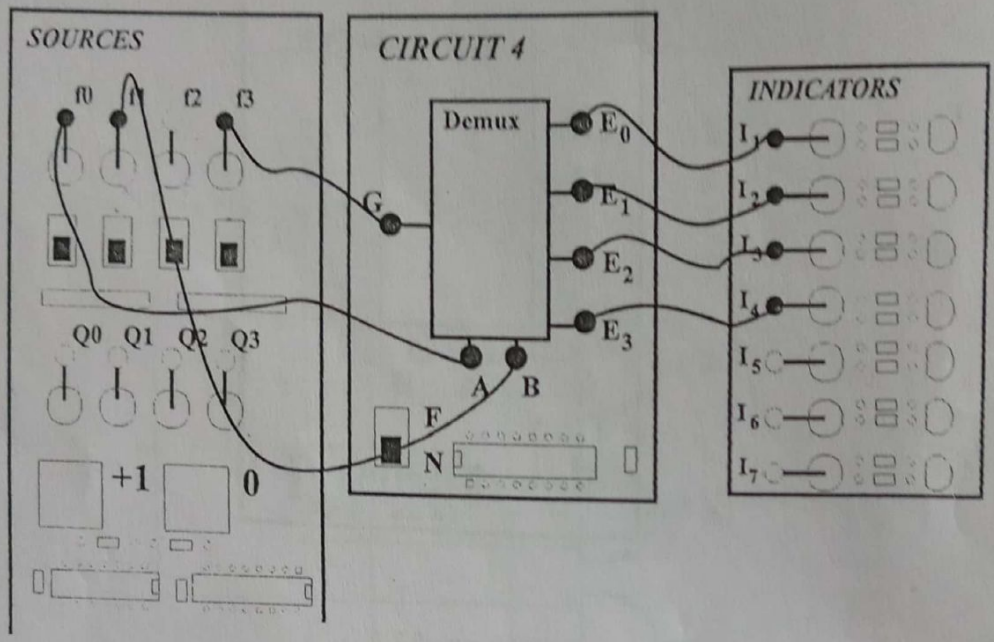


Figure 5.2.2

Feed the circuit by pressing the switch of the base unit EBC-100.

Fill in table 5.2.1 with the results obtained in the practice board.

B	A	G	E0	E1	E2	E3
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Table 5.2.1

 C4.1. Which are the outputs when G=1, A=0 and B=1?

 C4.2. Which are the outputs when G=0, A=0 and B=1?

 C4.3. What inputs must be there to obtain the following values at the output: E0=1, E1=1, E2=1, E3=1?

C4.4. If we want to use the demultiplexer as a decoder, what inputs must there be?

7.5.3 Practice 11: Study of the Error in Demultiplexers

Set the fault switch in the fault position (F).

There is now a fault that must be discovered in CIRCUIT 4.

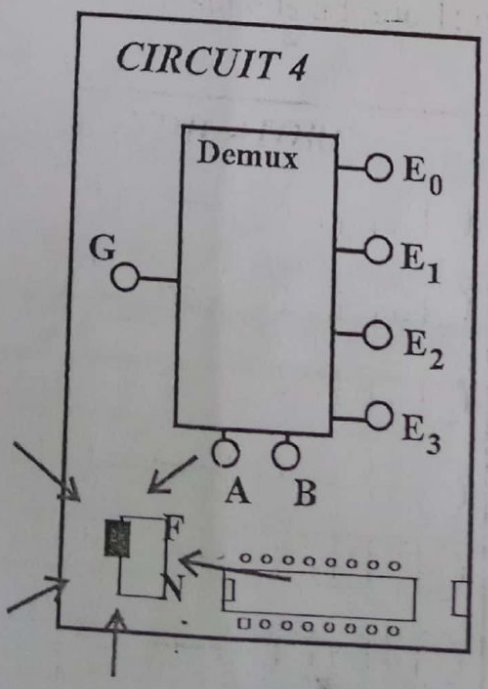


Figure 5.3.1

B	A	G	E0	E1	E2	E3
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Table 5.3.1

To discover what does not operate correctly, we suggest that you fill in table 5.3.1 and compare it with 5.2.1.

C4.5. What do you think is the cause of the fault?

7.5.4 Practice 12: Exercises

These exercises can only be made with the *software* version included with this board.

7.6 DIGITAL COMPARATORS

7.6.1 The Digital Comparator

First of all, we should point out that, although the digital and analogical comparators have the same mission, while one evaluates magnitudes of analogical voltage, the other (which we shall study now) works with digital signals.

It is sometimes necessary to know if a binary number A is bigger, smaller or equal to another number B. The system to determine this is called *digital value* (or binary) *comparator*. We shall first of all consider only one-bit numbers. As we already know, the NOR-EXCLUSIVE gate is an equality detector since:

$$E = \overline{A\overline{B} + \overline{A}B} = \begin{cases} 1 & A = B \\ 0 & A \neq B \end{cases}$$

The condition A>B is given by:

$$C = A\overline{B} = 1$$

since, if A>B, then A=1 and B=0, whereby C=1. On the other hand, if A=B or A<B (A=0, B=1), then C=0.

Similarly, the restriction A<B is determined by:

$$D = \overline{A}B = 1$$

The logic block diagram for the n-bit, drawn in figure 6.1.1, has the three necessary outputs C_n, D_n and E_n.

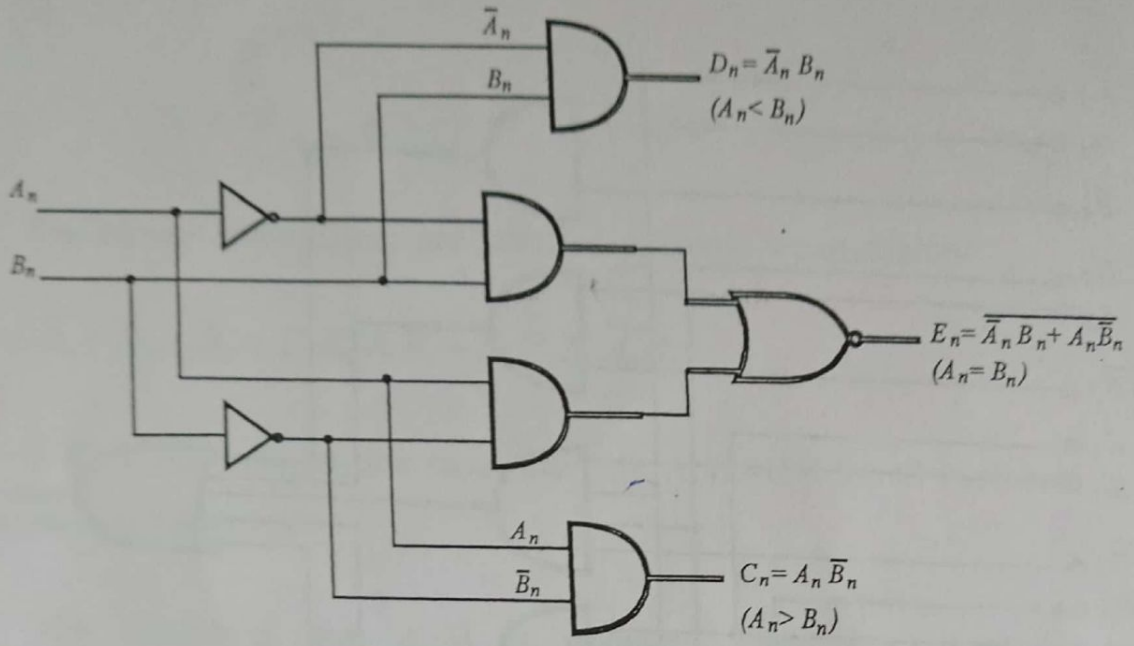


Figure 6.1.1

We shall now consider a comparator of 4 bit. A=B means that

$$A_3 = B_3 \quad A_2 = B_2 \quad A_1 = B_1 \quad A_0 = B_0$$

Therefore, the AND gate shown as E in figure 6.1.2, is described by

$$E = E_3 E_2 E_1 E_0 \qquad [1]$$

which implies that A=B if E=1 and A is not equal to B if E=0 (supposing that the input E' stays at the high level; E′=1).

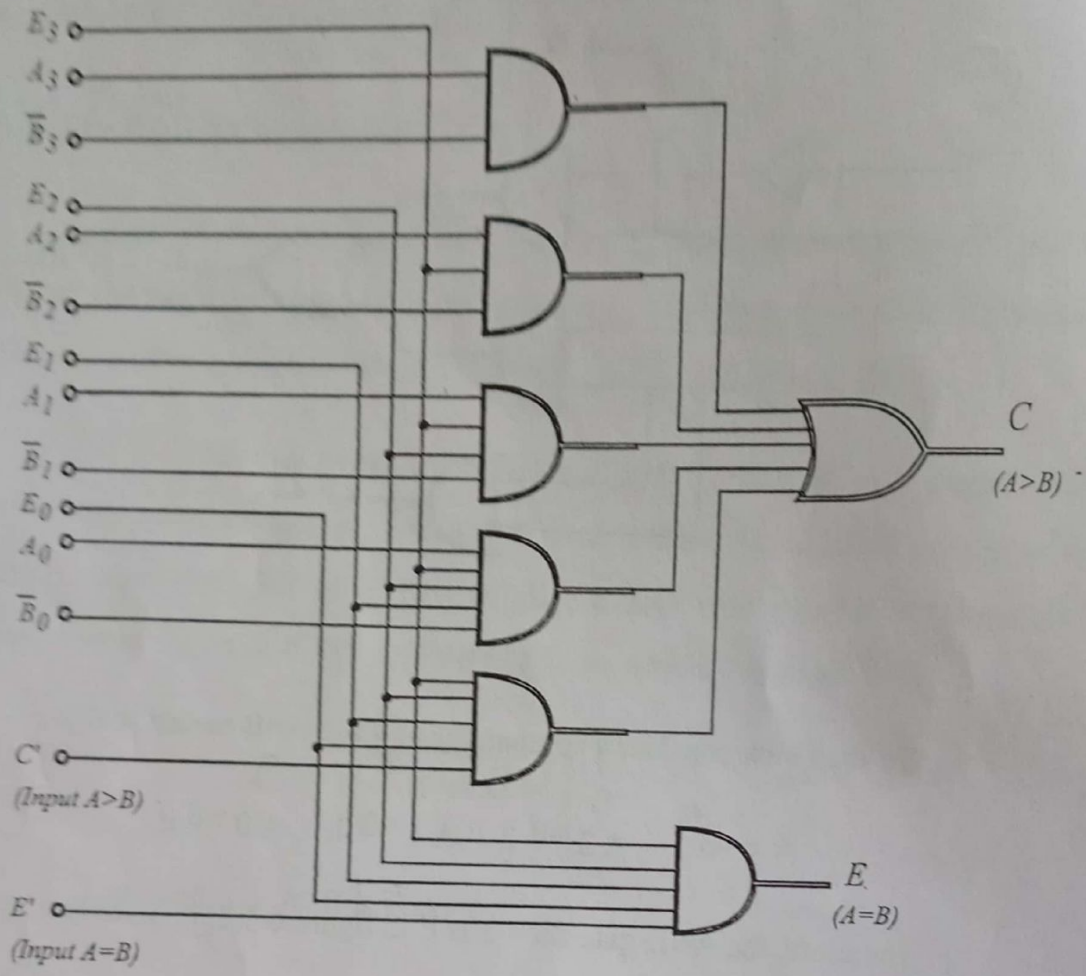


Figure 6.1.2

The inequality A>B means that

$$A_3 > B_3 \quad \text{(MSB)}$$

or

$$A_3 = B_3 \quad A_2 > B_2$$

or

$$A_3 = B_3 \quad A_2 = B_2 \quad A_1 > B_1$$

or

$$A_3 = B_3 \quad A_2 = B_2 \quad A_1 = B_1 \quad A_0 > B_0$$

The previous conditions are satisfied in Boole's expression:

$$C = A_3\overline{B}_3 + E_3A_2\overline{B}_2 + E_3E_2A_1\overline{B}_1 + E_3E_2E_1A_0\overline{B}_0 \qquad [2]$$

If C=1 and only in this case, the gate AND-OR for C is shown in figure 6.1.2 (supposing that C'=0)

The condition that A<B is obtained from the previous equation, exchanging A and B. That is to say:

$$D = \overline{A}_3B_3 + E_3\overline{A}_2B_2 + E_3E_2\overline{A}_1B_1 + E_3E_2E_1\overline{A}_0B_0 \qquad [3]$$

This implies that A<B only if D=1. This part of the system is obtained from figure 6.1.2 by changing A for B, B for A, and C for D. On the other hand, D can be obtained from **EC** since, if AB (E=0) and if A>B (C=0), then A<B (D=1). However, this use of D introduces the additional delay of the propagation of an inverter and an AND gate. From there, the logic indicated in the equation [3] for D is manufactured in the same chip as for C of the equation [2] and E of the equation [1].

The C.I. 54L85 is an encapsulated medium scale MSI that serves to compare 4 bits values. If it is necessary to compare numbers with more figures, more similar units can be used in parallel.

7.6.2 Practice 13: Study of a Comparator

7.6.2.1 Circuit 5

The circuit being studied is the one printed as *Circuit 5*, and which is represented in figure 6.2.1.

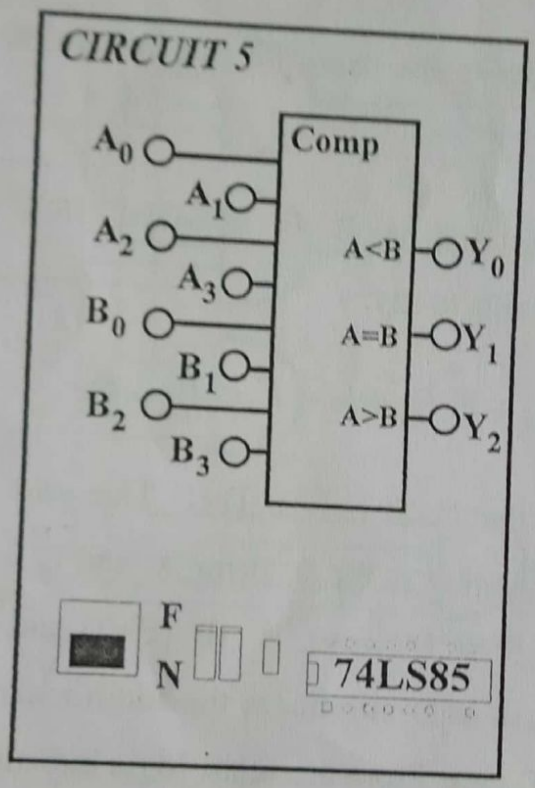


Figure 6.2.1

The integrated circuit used is 74LS85. It is a digital comparator of 4 bit that gives us information about the three possibilities: bigger, equal or smaller.

7.6.2.2 Practice Development

Remember that for a proper operation of all the circuits, the fault switch must be in the position *N*, that is to say downward.

To carry out the practice make the following assembly (figure 6.2.2):

Connect the four output signals of the group Q0..Q3 to the input channels A0..A3.

Connect the four output signals of the group P0..P3 to the input channels B0..B3.

Connect the comparator circuit outputs to three indicator channels.

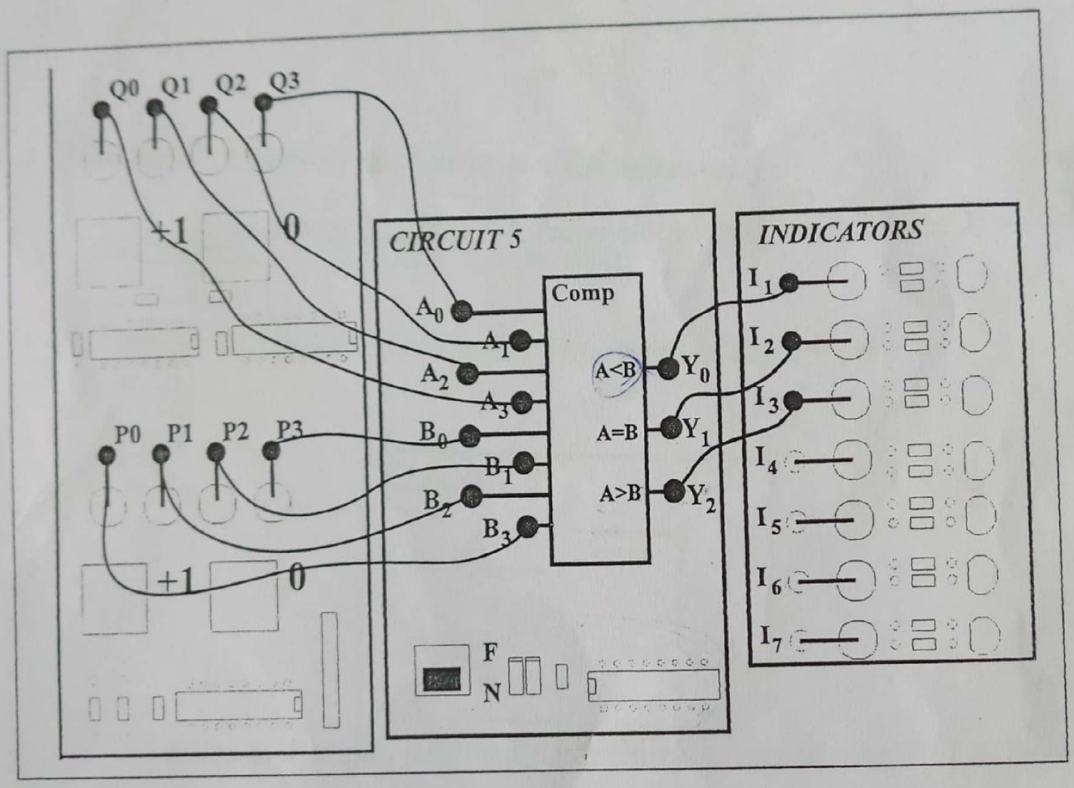


Figure 6.2.2

Feed the circuit by pressing the switch of the base unit EBC-100.

Fill in table 6.2.1 with the results obtained from the practice board.

A3	A2	A1	A0	B3	B2	B1	B0	A<B	A=B	A>B
1	0	1	0	1	0	1	0			
1	0	1	0	1	0	1	1			
1	0	1	1	1	0	1	0			

Table 6.2.1

 C5.1. The outputs are active ...

 C5.2. Which are the outputs when A=0000 and B=0000?

 C5.3. Are there any cases in which the three outputs are active?

 C4.4. If A=0110, and we can only connect a lead with a logic 1 in the input B,

where would we connect it to be sure that B>A is fulfilled?

7.6.3 Practice 14: Study of the Error in a Comparator

Set the fault switch in the fault position (F).

There is now a fault that must be discovered in CIRCUIT 5.

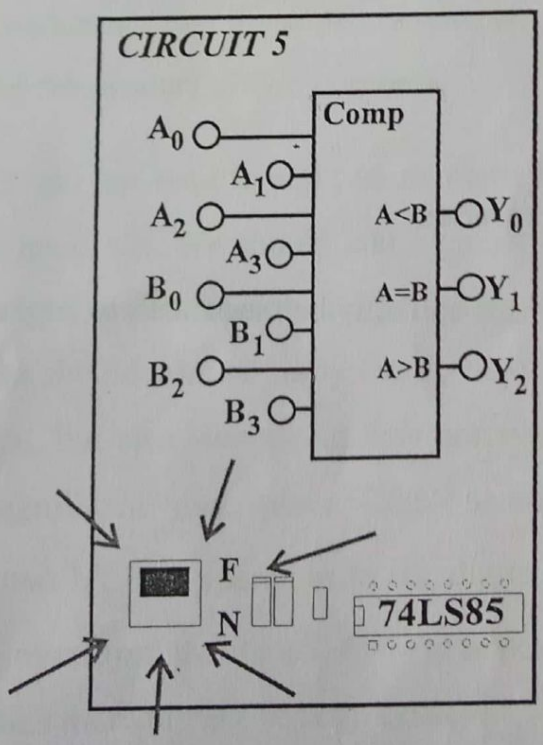


Figure 6.3.1

To discover what does not operate correctly, we suggest that you fill in

table 6.3.1 and compare it with 6.2.1.

A3	A2	A1	A0	B3	B2	B1	B0	A<B	A=B	A>B
1	0	1	0	1	0	1	0			
1	0	1	0	1	0	1	1			
1	0	1	1	1	0	1	0			

Table 6.3.1

 C5.5. What do you think is the cause of the fault?

7.6.4 Practice 15: Exercises

These exercises can only be made with *software* version included with this board.

7.7 Arithmetic and Logic Operations

In this chapter, as an example, we shall look at two integrated circuits that carry out an arithmetic function and a logic one (though in fact, both functions are logic).

7.7.1 The adder

A digital calculator must, of course, contain circuits that carry out arithmetic operations, for example, addition, subtraction, multiplication and division. The basic operations are addition and subtraction, since multiplication is, mainly, a repetition of adding, and division a repetition of subtracting. It is perfectly possible to build a calculator whose only arithmetic unit is an adder. Multiplication, for example, can be done using a programmer that gives instructions and indicates how to use the adder repetitively to find the product of two numbers.

Let's suppose that we want to add two numbers in decimal arithmetic and obtain the digits of the hundreds. We should add together not only the digits of the hundreds, but also the digits drawn from the tens (if there are any). In a similar way, in binary arithmetic, you should add, not only the digits of each significant place of the numbers to be added, but also (supposing they are available) the digits that are drawn from the next significant lower place. This operation is carried out in two parts: first, to add the two bit corresponding to the digits 2^n, and then to add the result of what we carry over from the digit 2^{n-1}. An adder of two inputs is called a *semi-adder,* since the full sum requires two semi-adders.

First, we shall indicate how a *semi-adder* is built with basic logic gates, and then we shall see how the *total or full adder* is formed. A semi-adder has two inputs (A and B) that represent the bit to be added, and two outputs: D (for the digits

with the same meaning as A and *B)* and C (for the bit that is carried over).

7.7.2 The Semi-adder

In figure 7.2.1 the symbol for the semi-adder and the truth table is shown. Notice that the column D gives the sum of A and B, as long as it can be represented by a single digit. If the sum were greater than what can be represented by a single digit, then column D would represent the digit of the sum corresponding to the same significant place as the addends. That is to say, that according to what we can see in the first three columns of the table, D directly gives the sum of A *and* B. When we add using the decimal base system, we say "l plus 1 equals 2", but, translated to the binary system this must be expressed as: "01 plus 01 equals 10", thus, in the last column, $D = 0$. The 1 will then have to be taken into account in the immediately superior significant figure, in this case the remainder column C, or rather C = 1.

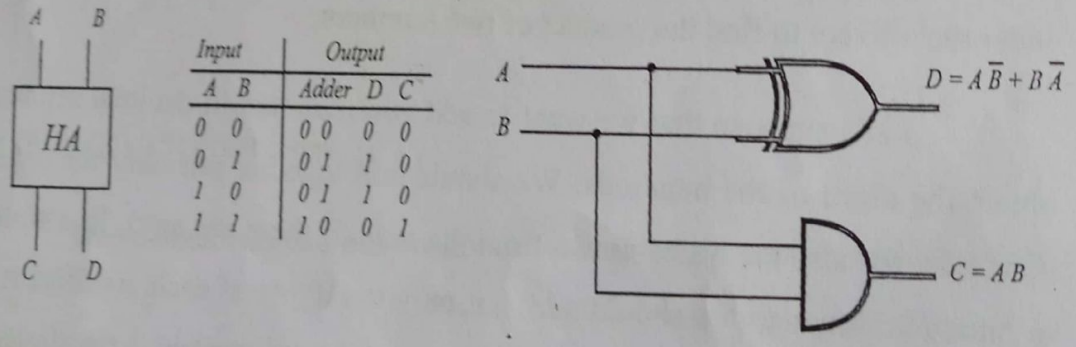


Figure 7.2.1

For figure 7.2.1 we can see that D corresponds to the function OR EXCLUSIVE and *C* to a logic AND gate. These functions are indicated in figure 7.2.1, and can be carried out in many different ways.

7.7.2.1 Operation in Parallel

Two multidigit numbers can be added in series (one column every time) or

in parallel (all the columns simultaneously). We shall first consider operation in parallel. For a binary number with N digits, there are (in addition to a common ground) N signal connections for each number. The nth line of the number A (or *B)* is activated by A_n (or B_n), that corresponds to the bit of the digit 2^n ($n = 0, 1, \ldots, N-1$). Figure 7.2.2 represents a parallel binary adder. Each digit, except for the first significant figure (2^0), requires a full adder that consists of two semi-adders in cascade. The sum digit (the result) of the bit 2^0 is $S_0 = D_0$ of a semi-adder, since it is not necessary to add any remainder to the sum $A_0 + B_0$. The sum S_n *(n 0)* of A_n plus B_n is done in two parts. First the digit Dn is obtained using a semi-adder, and then the remainder C_n, that results from the immediately lower place, is added to it. As an example, we shall consider n = 2 in figure 7.2.2. The remainder bit C_1 can be considered as the result of the direct sum of A_1 and B_1 if each one of them is 1. We shall call this first remainder C_{11} in figure 7.2.2. The second possibility is that $A_1=1$ *and* $B_1=0$ or conversely, in such a way that $D_1 = 1$, but that there is a remainder C_0 from the previous significant figure. The sum of $D_1 = 1$ and $C_0 = 1$ produces a remainder bit called C_{12}. It is obvious that C_{11} and C_{12} cannot both be 1 at the same time, although they can be 0 if $A_1=0$ and $B_1=0$. As either of the two, C_{11} or C_{12}, must be transmitted to the next stage, an OR gate must be placed between both stages, as it is indicated in figure 7.2.2.

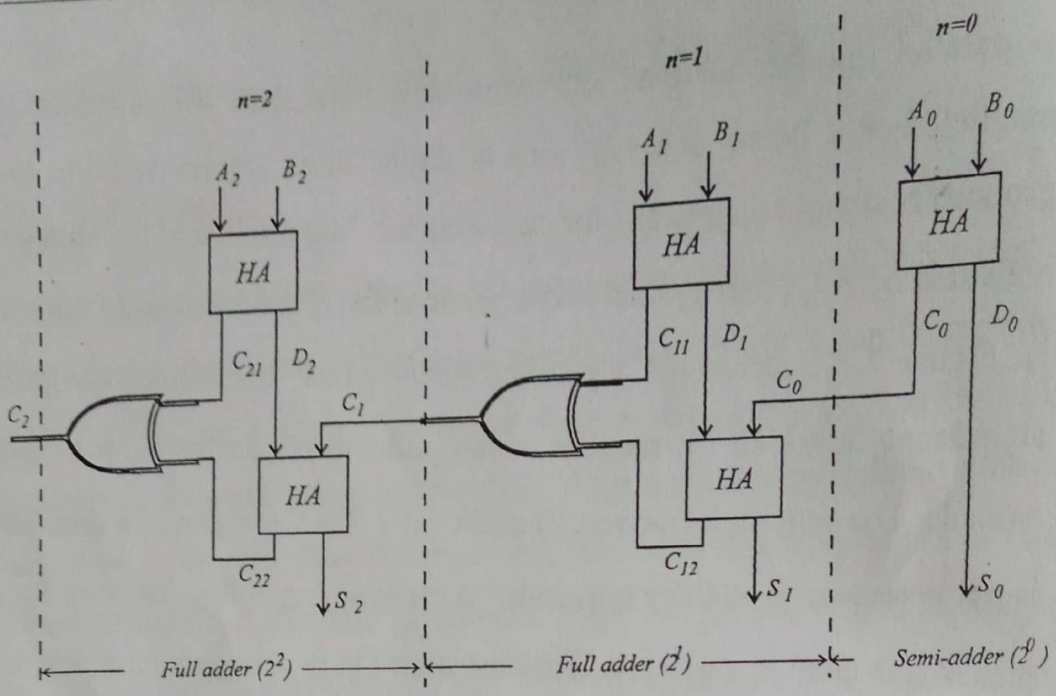


Figure 7.2.2

7.7.2.2 Full Adder

In the construction with integrated circuits, the sum is carried out using a full adder, and (for reasons of economy) not with two semi-adders. The nth full adder (FA) has three inputs: the addends A_n and B_n and remainder C_{n-1} (of the immediately previous bit). The outputs are the sum S_n and remainder C_n. In figure 7.2.3 a parallel adder of 4 bit is shown. The first significant bit is represented in the figure by FA0 (LSB) and it has no remainder input because there is no preceding stage.

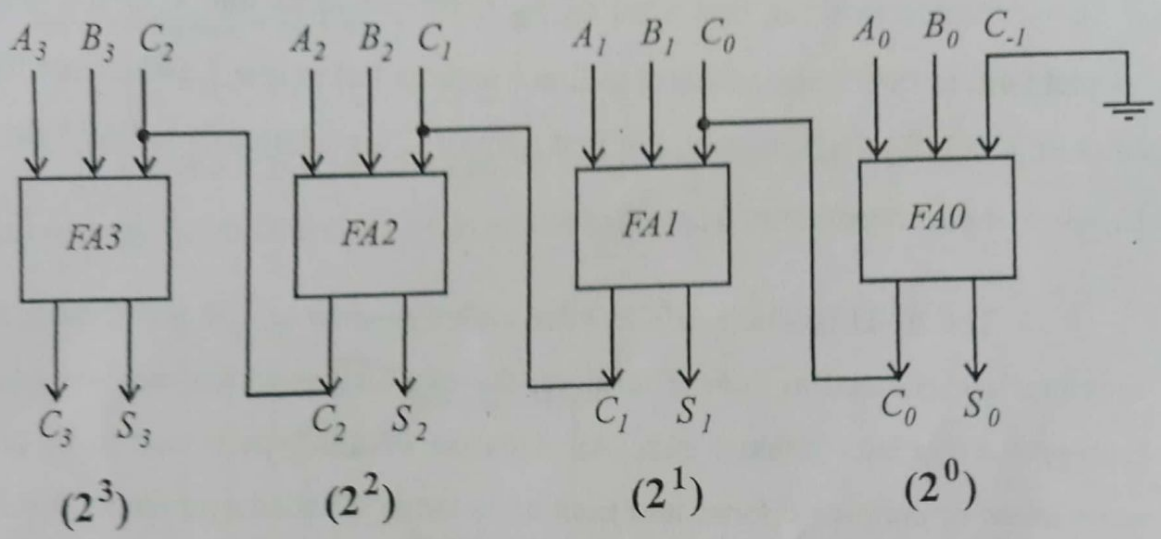


Figure 7.2.3

From table 7.2.1, it is possible to determine the interior circuit corresponding to the FA block. This chart is in fact the truth table of adding three binary bits. With this table it can be proven that the Boole expressions of S_n and C_n will be:

$$S_n = \overline{A}_n \overline{B}_n C_{n-1} + \overline{A}_n B_n \overline{C}_{n-1} + A_n \overline{B}_n \overline{C}_{n-1} + A_n B_n C_{n-1} \quad [6.1]$$

$$C_n = \overline{A}_n B_n C_{n-1} + A_n \overline{B}_n C_{n-1} + A_n B_n \overline{C}_{n-1} + A_n B_n C_{n-1} \quad [6.2]$$

Line	A_n	B_n	C_{n-1}	S_n	C_n
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

Table 7.2.1

Notice that the first term of S_n corresponds to line 1 of the table, the second term to line 2, the mediator to line 4 and the last to line 7, (which are the only lines in which S_n=l). Similarly, the first term of C_n corresponds to line 3 (in which C_n=l), and the second term to line 5, etc.

The AND function *ABC* is often called *product of A, B* and *C* Also, the OR function is expressed as *sum.* Therefore, the expressions of the previous equations represent a Boolean product sum. An equation of this type is said to be given *in normalised or canonical form,* and each of its terms is called a miniterm. A miniterm contains the product of all the variables of Boole, or their complements.

The expression of C_n can be simplified considerably in the following way: since $Y + Y + Y = Y$, then the equation 6.2 with $Y = A_n B_n C_{n-1}$, becomes

$$C_n = \left(\overline{A}_n B_n C_{n-1} + A_n B_n C_{n-1}\right) + \left(A_n \overline{B}_n C_{n-1} + A_n B_n C_{n-1}\right) + \left(A_n B_n \overline{C}_{n-1} + A_n B_n C_{n-1}\right) \quad [6.3]$$

As $\overline{X} + X = 1$, when $X = A_n$ in the first bracket, $X = B_n$ in the second bracket, and $X = C_n$ in the third bracket, the equation. 6.3 is reduced to

$$C_n = B_n C_{n-1} + A_n C_{n-1} + A_n B_n \quad [6.4]$$

This expression could be written directly from table 7.2.1, taking into account that C_n=1 if at least two of the three inputs are l.

It is interesting to notice that, if all the 1 are changed to 0 and all the 0 to 1, then the lines 0 and 7 exchange places, as do 1 and 6, 2 and 5, and also 3 and 4. Since the swap of 1 for 0 does not change the table, whatever the logic represented by table 7.2.1 is, it is equally valid if all the inputs and outputs are complementary. Thus, the equation 6.4 is true if all the variables are denied, or rather

$$\overline{C}_n = \overline{B}_n \overline{C}_{n-1} + \overline{A}_n \overline{C}_{n-1} + \overline{A}_n \overline{B}_n \quad [6.5]$$

Finding $D_n = (A_n + B_n + C_{n-1})C_n$ and comparing the results with the equation.

6.1, we find that $S_n = D_n + A_n B_n C_{n-1}$, or rather

$$S_n = A_n \overline{C}_n + B_n \overline{C}_n + C_{n-1} \overline{C}_n + A_n B_n C_{n-1} \quad [6.6]$$

Equations 6.4 and 6.6 can be completed using simple logic gates, so it is easy to carry out the nth stage of an adder, as it is shown in figure 7.2.4.

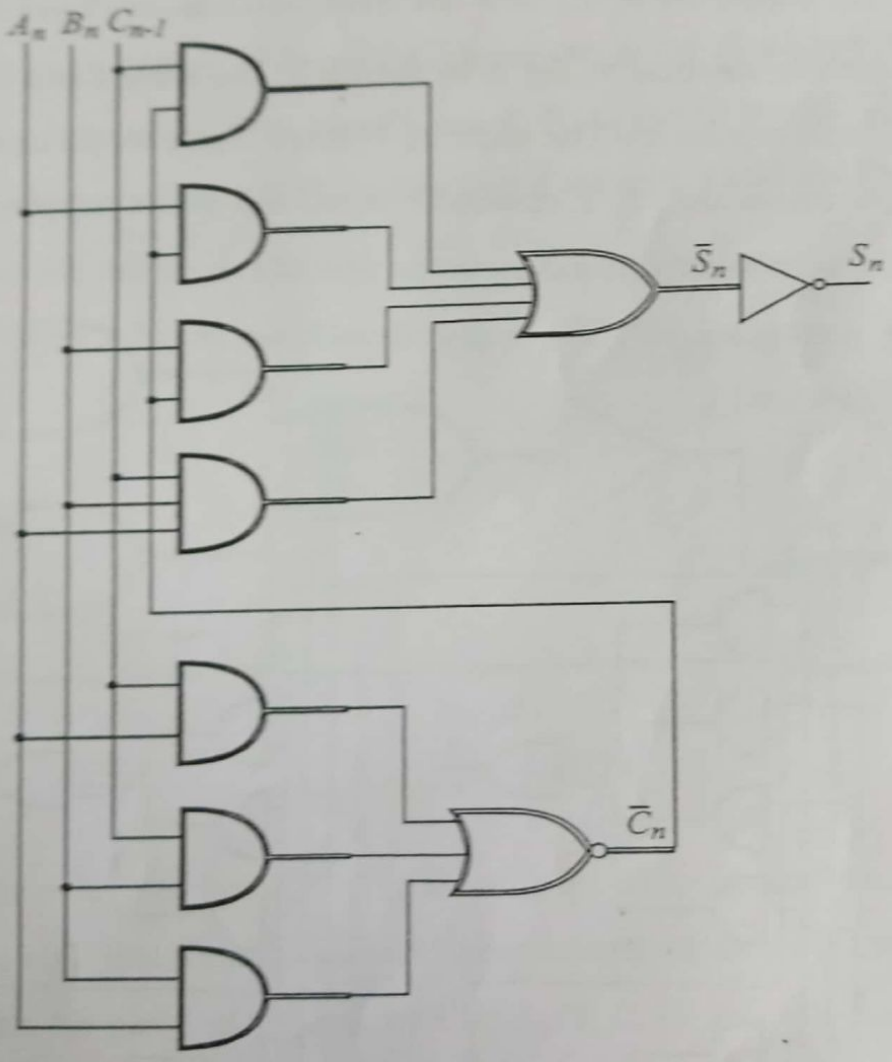


Figure 7.2.4

7.7.2.3 MSI Adders

There are full adders on the market of 1 bit, 2 bit and 4 bit, each in a single case. In figure 7.2.5, the logic topology for the sum of two bits is shown. The inputs

of the first stage are A_0 and A_1, and the input of the remainder C_{-1} is connected to ground. The output is the sum S_0. The remainder C_0 is connected inside and no output pin is necessary. This stage (LSB) 2 0 is identical to that of figure 7.2.4 with n=0. (The abbreviator LSB means Least Significant Bit).

In a 4-bit adder, C_1 does not leave, but it is connected inside to the third stage, which is identical to the first. Similarly, the second and fourth stages have identical logic topologies. The adder of 4 bit requires a 16-pin case: 8 inputs, 4 sum outputs, a drawn output, a remainder input, the power supply and ground. The remainder input is only necessary in the case of two arithmetic units in cascade; for example, putting a 2-bit adder in cascade with another of 4, gives us the sum of two numbers of 6 bits.

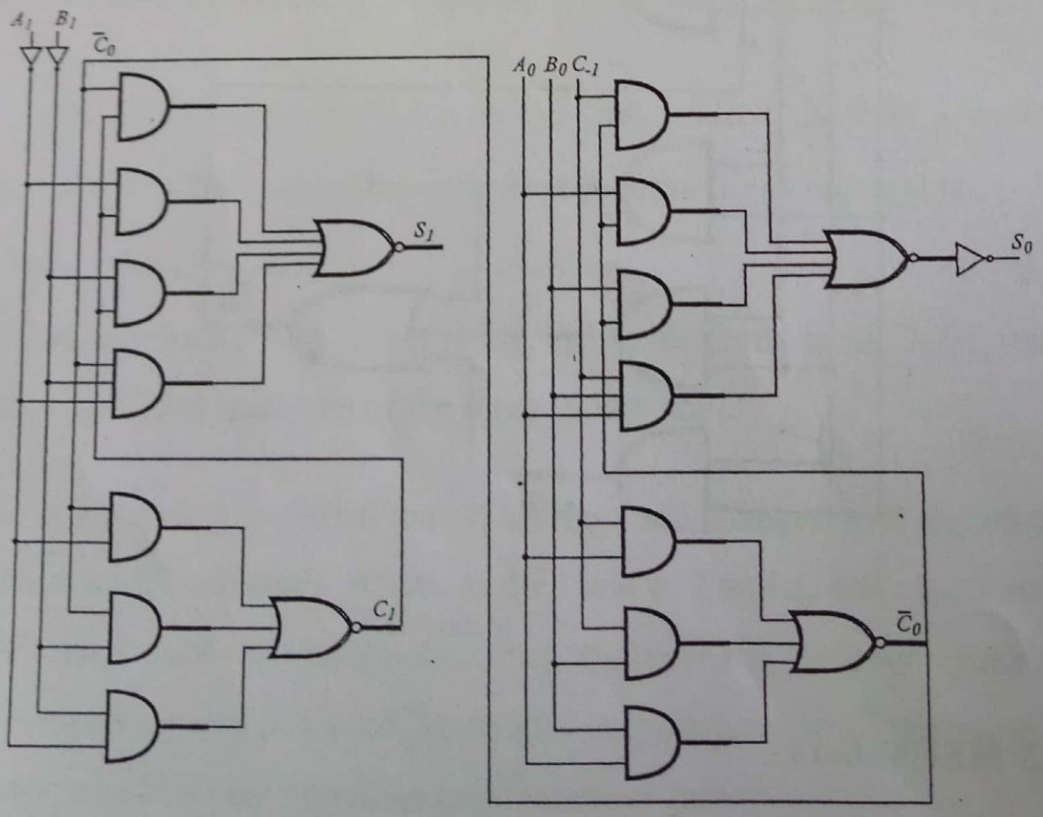


Figure 7.2.5

7.7.3 The Parity Checker

An arithmetic operation that is often used in digital systems is determining whether the sum of binary bit in a piece of information is even or odd. The output of an OR-EXCLUSIVE gate is 1 if one input is exclusively 1 and the other 0. In other words, the output is 1 if the sum of the digits is 1. An extension of this concept to the multiple OR-EXCLUSIVE of figure 7.3.1 brings us to the conclusion of the fact that Z=1 (or Y=0) if the sum of the bit of input A, B, C and D is odd. Therefore, if the input P is connected to ground (P'=0) then P=0 if the sum is odd and P=1 if it is even.

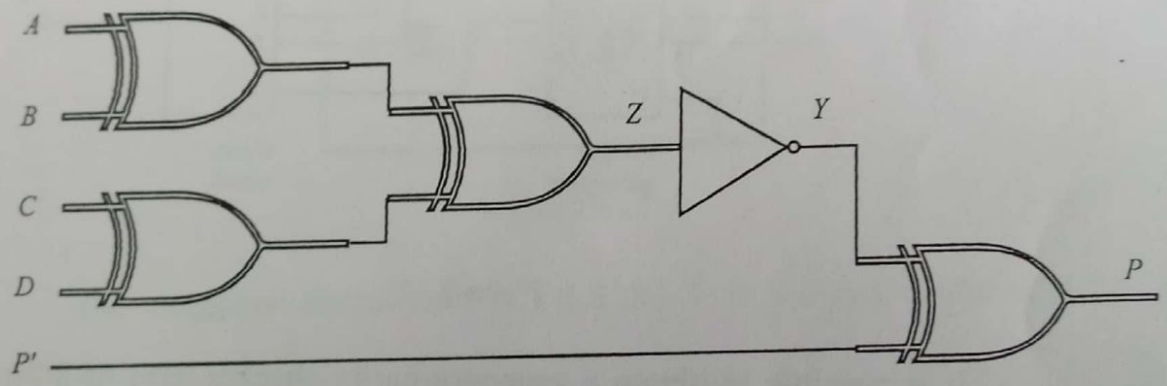


Figure 7.3.1

The system of figure 7.3.1 is not only a parity checker, it can also be used to generate a bit P that fixes the parities. Independently from the parities of the input signal of 4 bit, the parity of the code of 5 bit A, B, C, D, and *P* is odd. This premise comes from the fact that, if the sum of A, B, C and D is odd (even), then P is 0 (1), and consequently, the sum A, B, C, D and P is always odd.

The use of a parity code increases the transmission safety for binary information. As it is indicated in figure 7.3.2, a parity bit Pl is generated that is transmitted along with the input signals of *N* bit. In the receiver, the parity of the increased signal (N + l) bit is checked, and if the output P2 of the checker is 0, it can

be supposed that there is no mistake in the transmission of the message, while, if P2=1 indicates that there is a mistake in the reception (for example due to noise). Take into account that with only one parity checker, only mistakes in an odd number of digits can be detected.

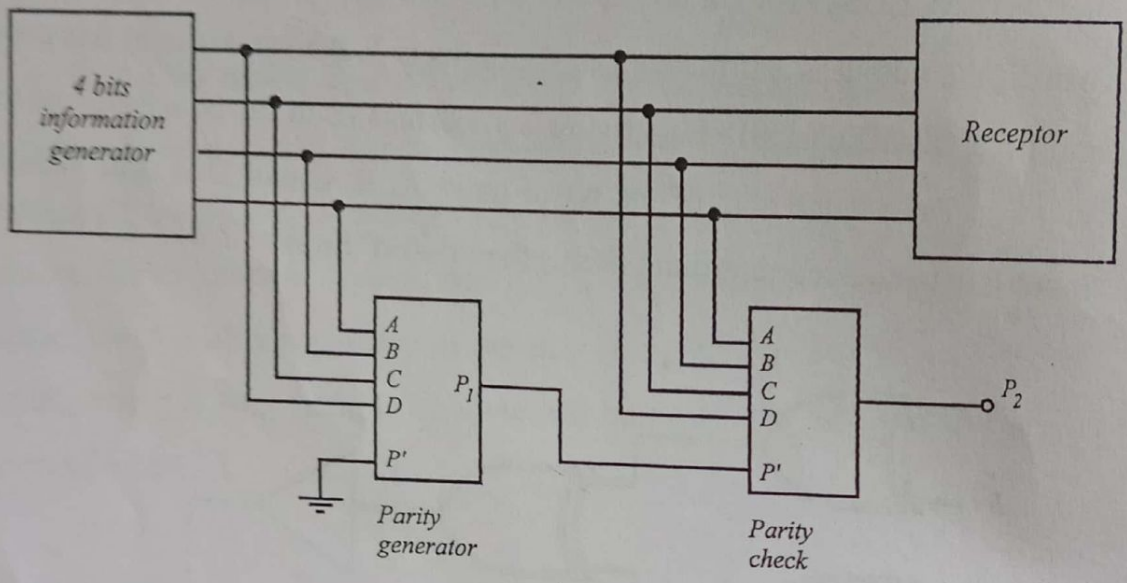


Figure 7.3.2

It is possible to obtain a generator/parity checker MSI of 8 bit (74LS180) with controlled inputs, which can be applied to the two types of parity: even or odd. For signals greater than 8 bit, several units will be used in cascade.

7.7.4 Practice 16: Study of an Adder

7.7.4.1 Circuit 6

The circuit being studied is printed as *Circuit 6* and is represented in figure 7.4.1.

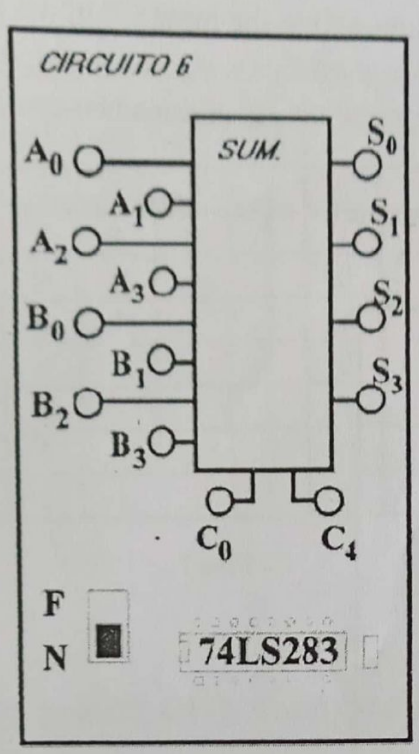


Figure 7.4.1

The integrated circuit used is 74LS283. It is a digital adder of 4 bit, plus the drawing.

7.7.4.2 Carrying out the Practice

Remember that for a proper circuit operation, the fault switch must be in the position *N*.

To carry out the practice, make the following assembly (figure 7.4.2):

- Connect the four output signals of the group Q0..Q3 to the input channels A0..A3.
- Connect the four output signals of the group P0..P3 to the input channels B0..B3.

- Connect the source f0 to the input C0 of the adder circuit.
- Connect the outputs S0..S3 of the adder circuit to the indicator channels.
- Connect the output C4 of the adder circuit to the other indicator channel.

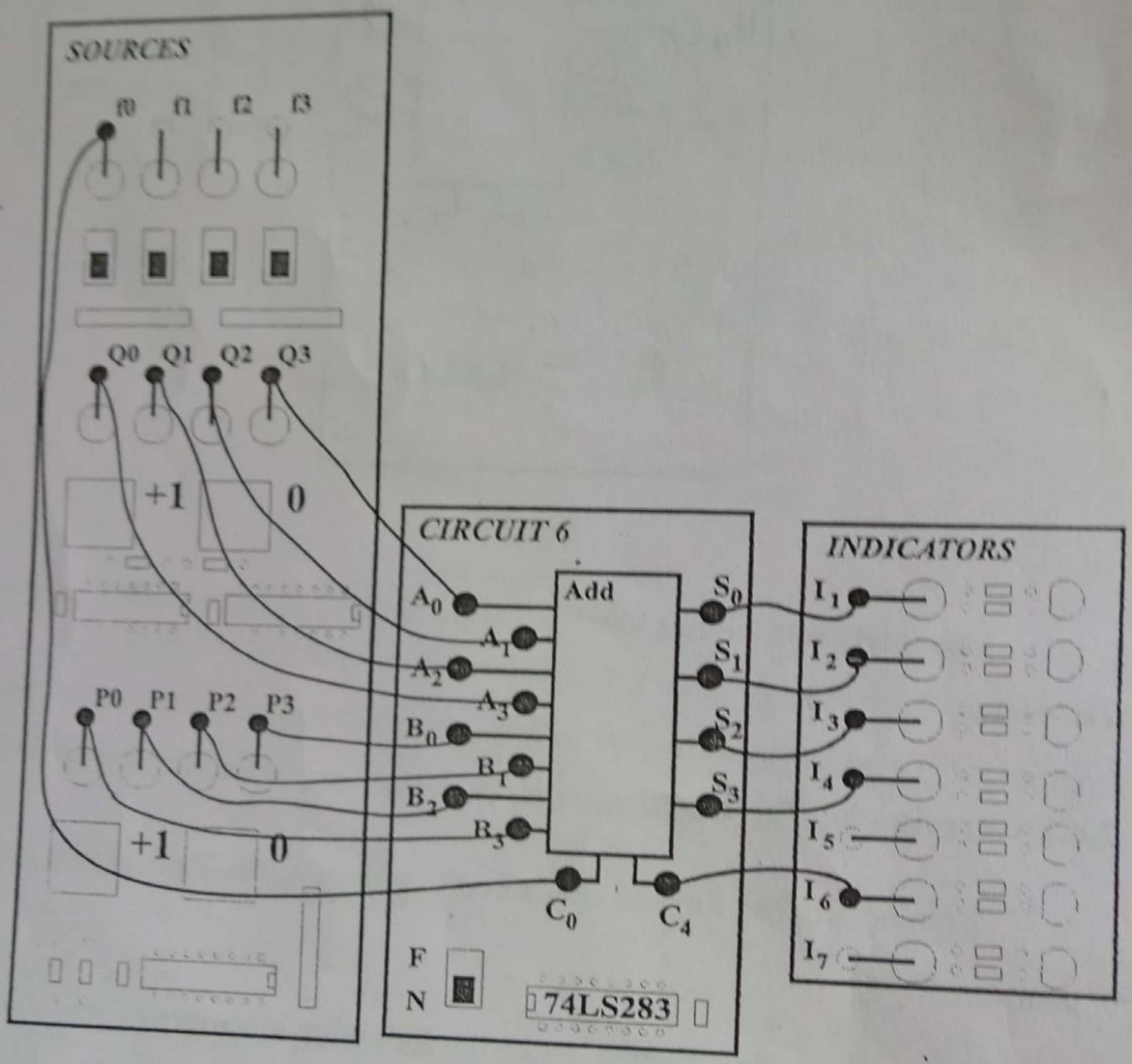


Figure 7.4.2

Feed the circuit by pressing the switch of the base unit EBC-100.

Fill in table 7.4.1with the results obtained from the practice board.

A3	A2	A1	A0	B3	B2	B1	B0	C0	S3	S2	S1	S0	C4
0	1	1	0	0	1	0	1	0					
0	1	1	0	0	1	0	1	1					
1	0	0	1	0	1	0	1	0					
1	0	0	1	0	1	0	1	1					
1	0	1	0	0	1	1	1	0					
1	0	1	0	0	1	1	1	1					
1	0	1	0	1	1	1	0	0					
1	0	1	1	1	1	1	0	1					

Table 7.4.1

C6.1. If the output is S=0000, C4=1, which are the inputs?

C6.2. Which is the output when A=0111 and B=1000?

C6.3. If A=8, B=7 and C0=1, which is the output?

C6.4. If A=15, B=15 and C0=1, which is the output?

7.7.5 Practice 17: Study of Errors in the Arithmetic and Logic Operations

Put the fault switch in the fault position (F).

There is now a fault that must be discovered in CIRCUIT 6.

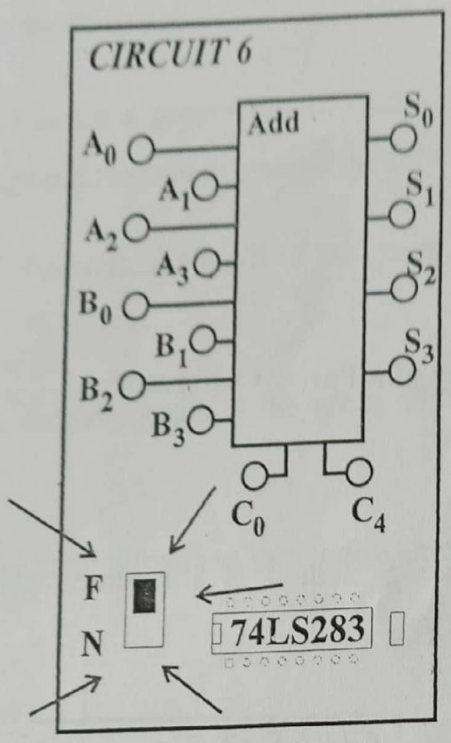


Figure 7.5.1

To discover what does not operate correctly, we suggest that you fill in table 7.5.1, and compare it with 7.4.1.

A3	A2	A1	A0	B3	B2	B1	B0	C0	S3	S2	S1	S0	C4
0	1	1	0	0	1	0	1	0					
0	1	1	0	0	1	0	1	1					
1	0	0	1	0	1	0	1	0					
1	0	0	1	0	1	0	1	1					
1	0	1	0	0	1	1	1	0					
1	0	1	0	0	1	1	1	1					
1	0	1	0	1	1	1	0	0					
1	0	1	1	1	1	1	0	1					

Table 7.5.1

C6.5. What do you think is the cause of the fault?

7.7.6 Practice 18: Study of a Parity Generator

7.7.6.1 Circuit 7

The circuit being studied is silk-screen printed as *Circuit 7*, and is represented in figure 7.6.1.

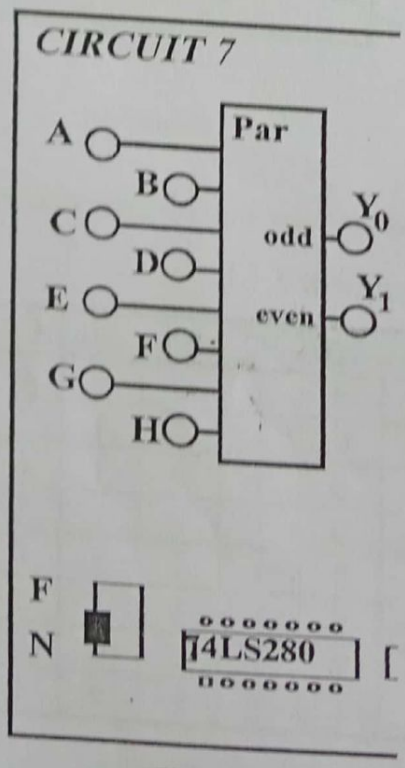


Figure 7.6.1

The integrated circuit used is 74LS280. It is a parity generator of 9 bits, which one of them is connected to ground, we shall therefore analyse only 8 bits.

7.7.6.2 Carrying out the Practice

Remember that for a proper circuit operation, the fault switch must be in the position *N*. To carry out the practice make the following assembly (figure 7.6.2):

- Connect the four output signals of the group Q0..Q3 to the input lines A, B, C and D.
- Connect the output signals of the group P0..P3 to the input channels E, F, G and H.
- Connect the circuit outputs to two indicator channels.

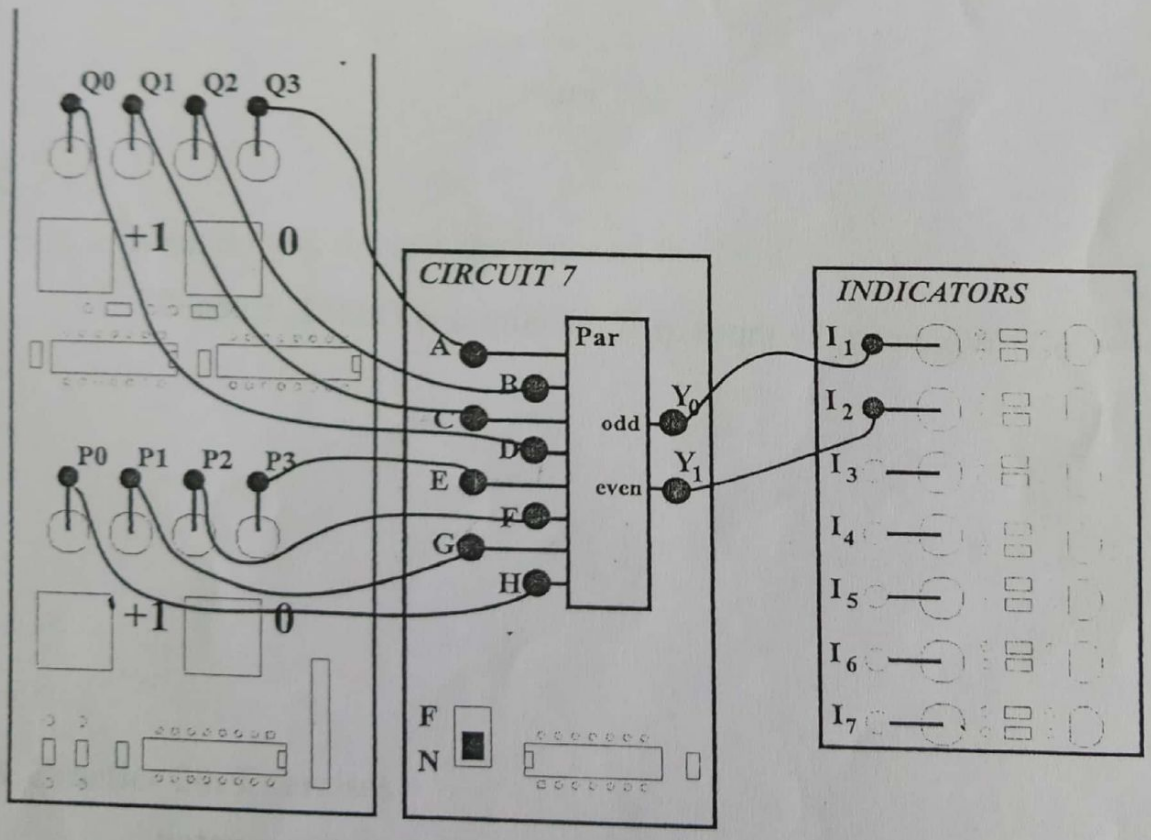


Figure 7.6.2

- Feed the circuit by pressing the switch of the base unit EBC-100.
- Fill in table 7.6.1with the results obtained in the practice board.

A	B	C	D	E	F	G	H	Odd	Even
0	1	1	0	0	1	0	0		
1	0	1	0	0	1	1	1		
1	0	1	0	0	1	1	0		
1	0	1	0	0	1	1	1		
1	0	1	1	1	1	1	1		

Table 7.6.1

 C6.6. Which is the output with the input 1011 1111?

 C6.7. What can the input be if the output is Odd=1, Even=0?

7.7.7 Practice 19: Study of the Error in the Parity Generator

Put the fault switch in the fault position (F).

There is now a fault that must be discovered in CIRCUIT 7.

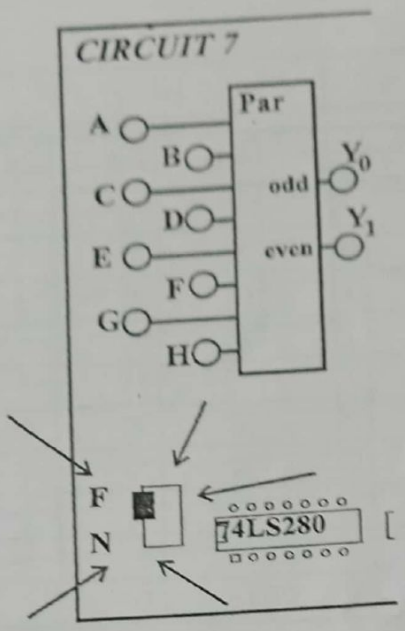


Figure 7.7.1

C6.8. What do you think is the cause of the fault?

7.7.8 Practice 20: Exercises

These exercises can only be made with software version included with this Board.

7.8.1 Chapter 1

E_1	E_3	E_2	E_1	E_0	S_2	S_1	G_S
0	0	X	X	X			
0	1	0	X	X			
0	1	1	0	X			
0	1	1	1	0			
0	1	1	1	1			
1	0	X	X	X			
1	1	0	X	X			
1	1	1	0	X			
1	1	1	1	0			
1	1	1	1	1			

Table 2.2.1

 C1.1.	
 C1.2.	
 C1.3.	
 C1.4.	

E_I	E_3	E_2	E_1	E_0	S_2	S_1	G_S
0	0	X	X	X			
0	1	0	X	X			
0	1	1	0	X			
0	1	1	1	0			
0	1	1	1	1			
1	0	X	X	X			
1	1	0	X	X			
1	1	1	0	X			
1	1	1	1	0			
1	1	1	1	1			

Table 2.3.1

7.8.2 Chapter 2

D	C	B	A	a	b	c	d	e	f	g
0	0	0	0							
0	0	0	1							
0	0	1	0							
0	0	1	1							
0	1	0	0							
0	1	0	1							
0	1	1	0							
0	1	1	1							
1	0	0	0							
1	0	0	1							
1	0	1	0							
1	0	1	1							
1	1	0	0							
1	1	0	1							
1	1	1	0							
1	1	1	1							

Table 3.3.1

C2.1.	
C2.2.	
C2.3.	
C2.4.	
C2.5.	

D	C	B	A	a	b	c	d	e	f	g
0	0	0	0							
0	0	0	1							
0	0	1	0							
0	0	1	1							
0	1	0	0							
0	1	0	1							
0	1	1	0							
0	1	1	1							
1	0	0	0							
1	0	0	1							
1	0	1	0							
1	0	1	1							
1	1	0	0							
1	1	0	1							
1	1	1	0							
1	1	1	1							

Table 2.3.1

C2.6.

7.8.3 Chapter 3

 C3.1.	
 C3.2.	

B	A	Str	S	S'
X	X	1		
0	0	0		
0	1	0		
1	0	0		
1	1	0		

Table 4.2.1

 C3.3.	
 C3.4.	

B	A	Str	S	S'
X	X	1		
0	0	0		
0	1	0		
1	0	0		
1	1	0		

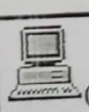
Table 4.3.1

 C3.5.	

7.8.4 Chapter 4

B	A	G	E0	E1	E2	E3
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Table 5.2.1

 C4.1.	
 C4.2.	
 C4.3.	
 C4.4.	

B	A	G	E0	E1	E2	E3
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Table 5.3.1

 C4.5.	

7.8.5 Chapter 5

A3	A2	A1	A0	B3	B2	B1	B0	A<B	A=B	A>B
1	0	1	0	1	0	1	0			
1	0	1	0	1	0	1	1			
1	0	1	1	1	0	1	1			

Table 6.2.1

C5.1.	
C5.2.	
C5.3.	
C5.4.	

A3	A2	A1	A0	B3	B2	B1	B0	A<B	A=B	A>B
1	0	1	0	1	0	1	0			
1	0	1	0	1	0	1	1			
1	0	1	1	1	0	1	0			

Table 6.3.1

C5.5.	

7.8.6 Chapter 6

A3	A2	A1	A0	B3	B2	B1	B0	C0	S3	S2	S1	S0	C4
0	1	1	0	0	1	0	1	0					
0	1	1	0	0	1	0	1	1					
1	0	0	1	0	1	0	1	0					
1	0	0	1	0	1	0	1	1					
1	0	1	0	0	1	1	1	0					
1	0	1	0	0	1	1	1	1					
1	0	1	0	1	1	1	0	0					
1	0	1	1	1	1	1	0	1					

Table 7.4.1

C6.1.	
C6.2.	
C6.3.	
C6.4.	

A3	A2	A1	A0	B3	B2	B1	B0	C0	S3	S2	S1	S0	C4
0	1	1	0	0	1	0	1	0					
0	1	1	0	0	1	0	1	1					
1	0	0	1	0	1	0	1	0					
1	0	0	1	0	1	0	1	1					
1	0	1	0	0	1	1	1	0					
1	0	1	0	0	1	1	1	1					
1	0	1	0	1	1	1	0	0					
1	0	1	1	1	1	1	0	1					

Table 7.5.1

C6.5.	

A	B	C	D	E	F	G	H	Odd	Even
0	1	1	0	0	1	0	0		
1	0	1	0	0	1	1	1		
1	0	1	0	0	1	1	0		
1	0	1	1	1	1	1	1		

Table 7.6.1

C6.6.	
C6.7.	
C6.8.	