

**ENCS 211**

**Section #2**

**Experiments 3**

**Encoders, Decoders, Multiplexers and**

**Demultiplexers**

**Instructor: Hanna Bullata**

**Sara Sameeh #1130501**

**Partner name & id: duha #1130093**

* **Abstract :**

This experiment about Decoder, Encoder, Multiplexer & Demultiplexer.

\* aims of experiment:

To understands the operating principles of decoders, encoders, multiplexer & demultiplexer.

To use basic gates and IC to construct decoders, encoders, multiplexer & demultiplexer.

\* To achieve the previous aims we used:

1. KL-22001 Basic Electricity Circuit Lab

2. KL-26002 Combinational Logic Circuit Experiment Module (2)

3. KL-26003 Combinational Logic Circuit Experiment Module (3)

4. KL-26004 Combinational Logic Circuit Experiment Module (4)

* **contents :**

**Abstract \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2**

**Introduction \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 3**

**Procedure\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 5**

**Conclusion \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 14**

**Logic problems \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_14**

**Prelab\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 15**

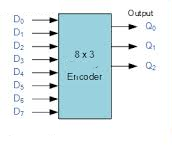
**References \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 18**

* **Introduction :**

**\*** Theory**:**

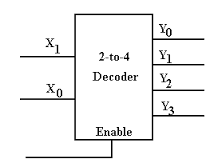
Encoder circuit: An encoder is a combinational logic gate that accepts one or multiple inputs and generates a specific output code. Only one input is triggered at a time. An encoder with n-bit inputs and m-bit.

Note: The outputs numbers ^ 2 = inputs numbers For example : there is 4x2 encoder, 8x2 encoder .. Etc.

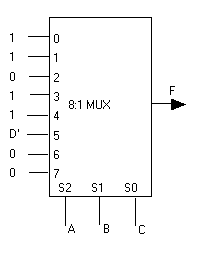


Decoder circuit: a logic circuit that takes a parallel binary number & the output is a binary signal that indicates the presence of that number.

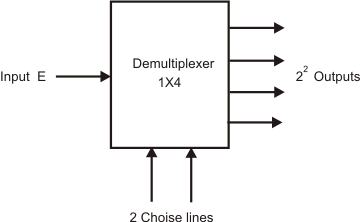
A common type of combinational decoder circuit is line decoder which takes an n-digit binary number and decodes it into 2n data lines, such as 1 to 2 line decoder, 2 to 4 line decoder.



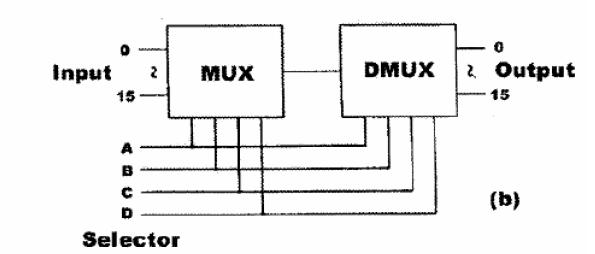
Multiplexer Circuits: it’s a logic circuits that select one of the input and pass it to the single output so it’s called “data selectors” (select one output among many inputs , one of the multiple inputs selected by selectors then goes to output, the capacity of multiplexer is depended on number of selectors, for example if we have 1 selector , then we have 2-1 line multiplexer , & if we have 3 selectors then we have 8-1 multiplexer , because 2^3 = 8 .



**Demultiplexer: (DMUX)** also called data distributor or data router is a logic circuit which simply the opposites of MUX circuit, it has a single input & multiple outputs, depending on the selectors , the input may be connected to any one of the outputs .

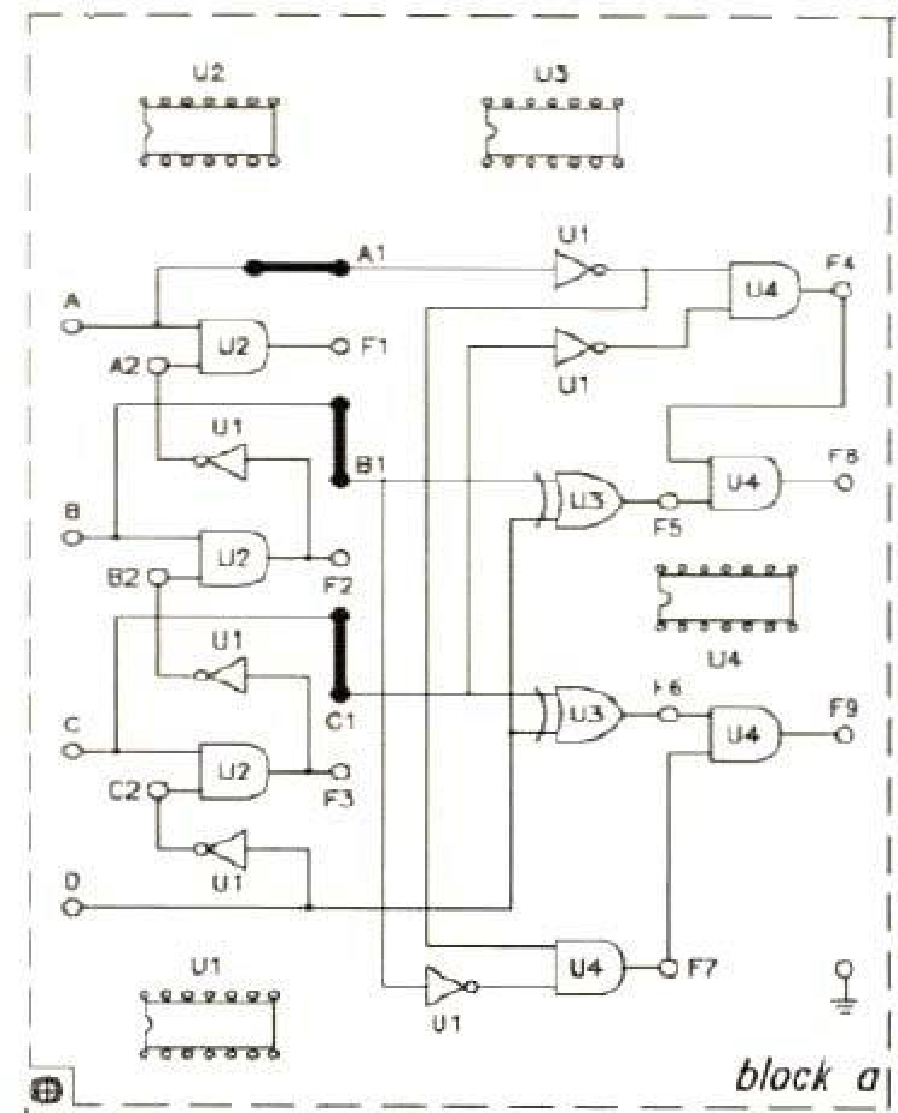
****

**\*\*\*Note: by combining MUX & DMUX long distance system transmission will set up, increasing the efficiency of transmission lines as following:**

****

* **Procedure discussion :**

We used basic gates to construct 4 - to - 2 line encoder as following:



And the result is:

| Inputs | | | | Outputs | |
| --- | --- | --- | --- | --- | --- |
| D | C | B | A | F9 | F8 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 |

We notice that this encoder work only when one of the inputs is 1, or the output will be 0s, it is work as following:

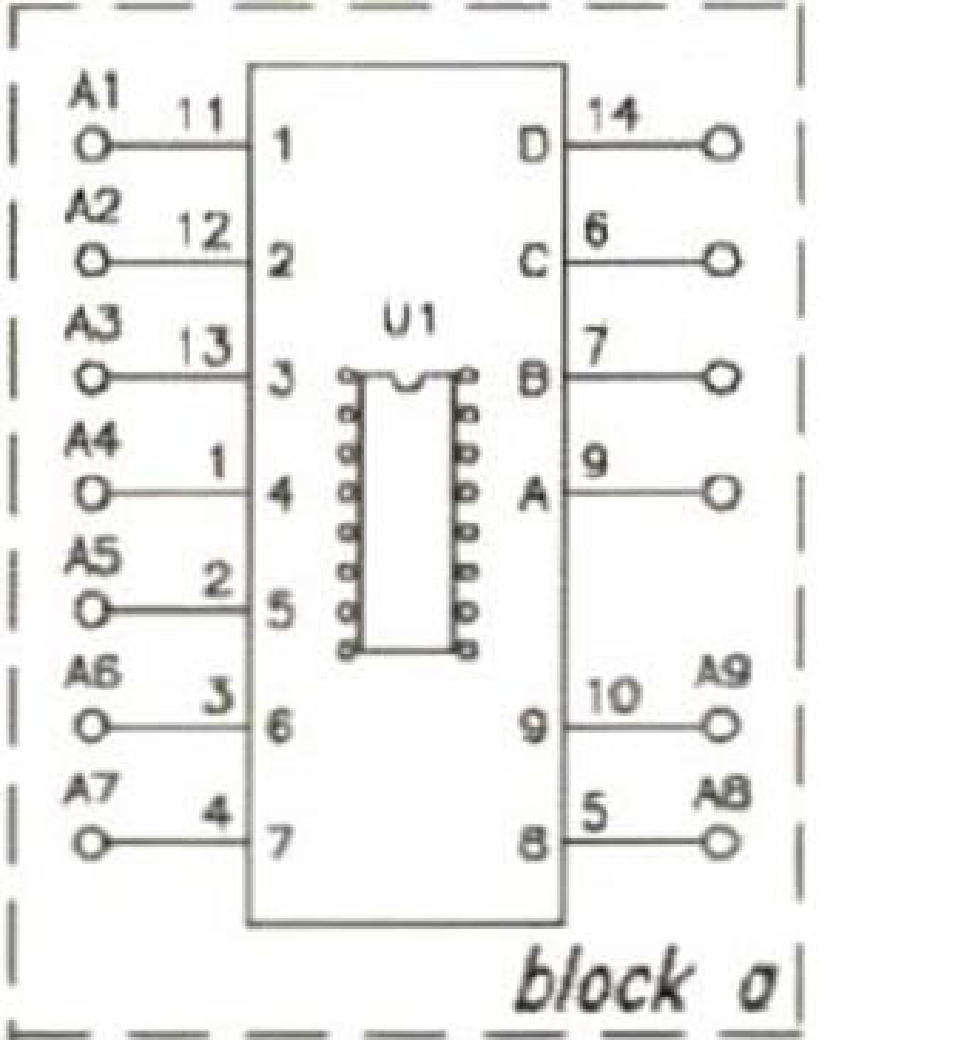
Input is: 0001, the output is 00

Input is: 0001, the output is 01

Input is: 0001, the output is 10

Input is: 0001, the output is 11

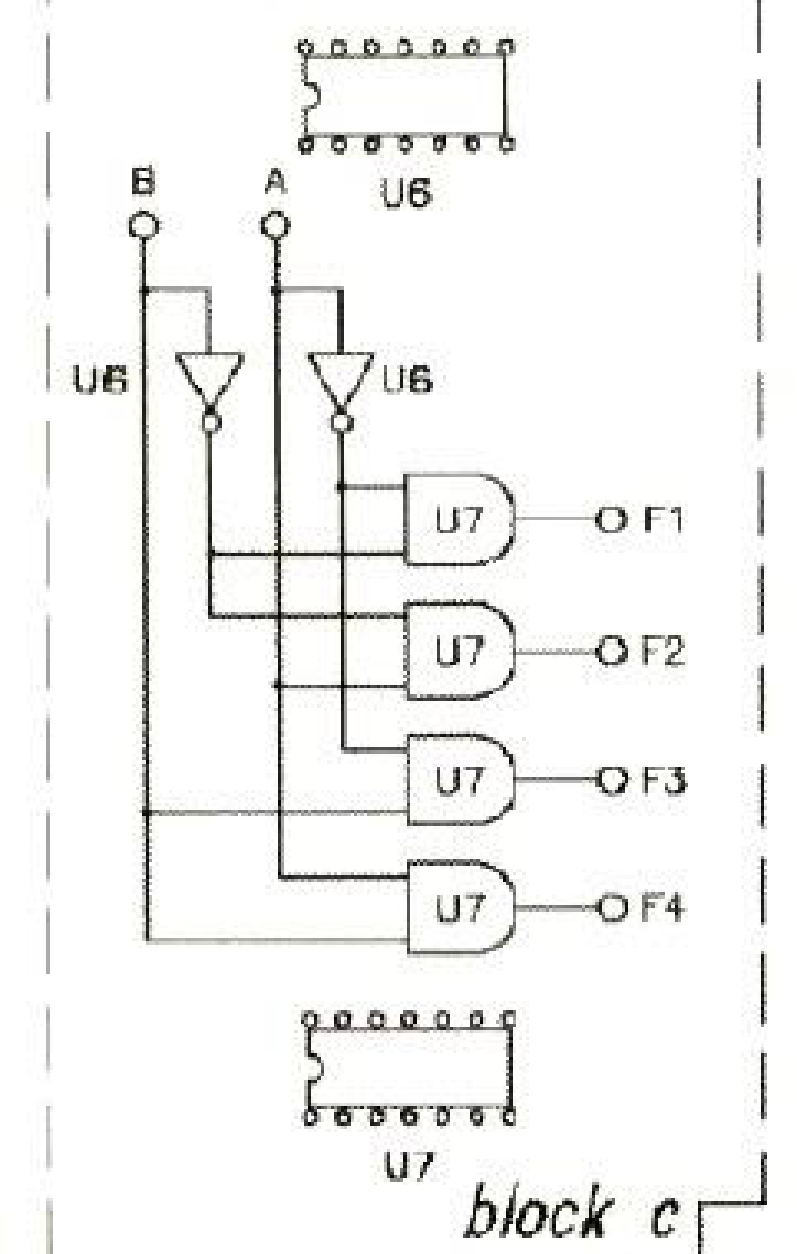
We construct 10 - to - 4 line encoder with TTL IC as following:



The result is:

| Inputs | | | | | | | | | Outputs | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | D | C | B | A |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |

We use basic gates to construct 2-to-4 decoder as following:

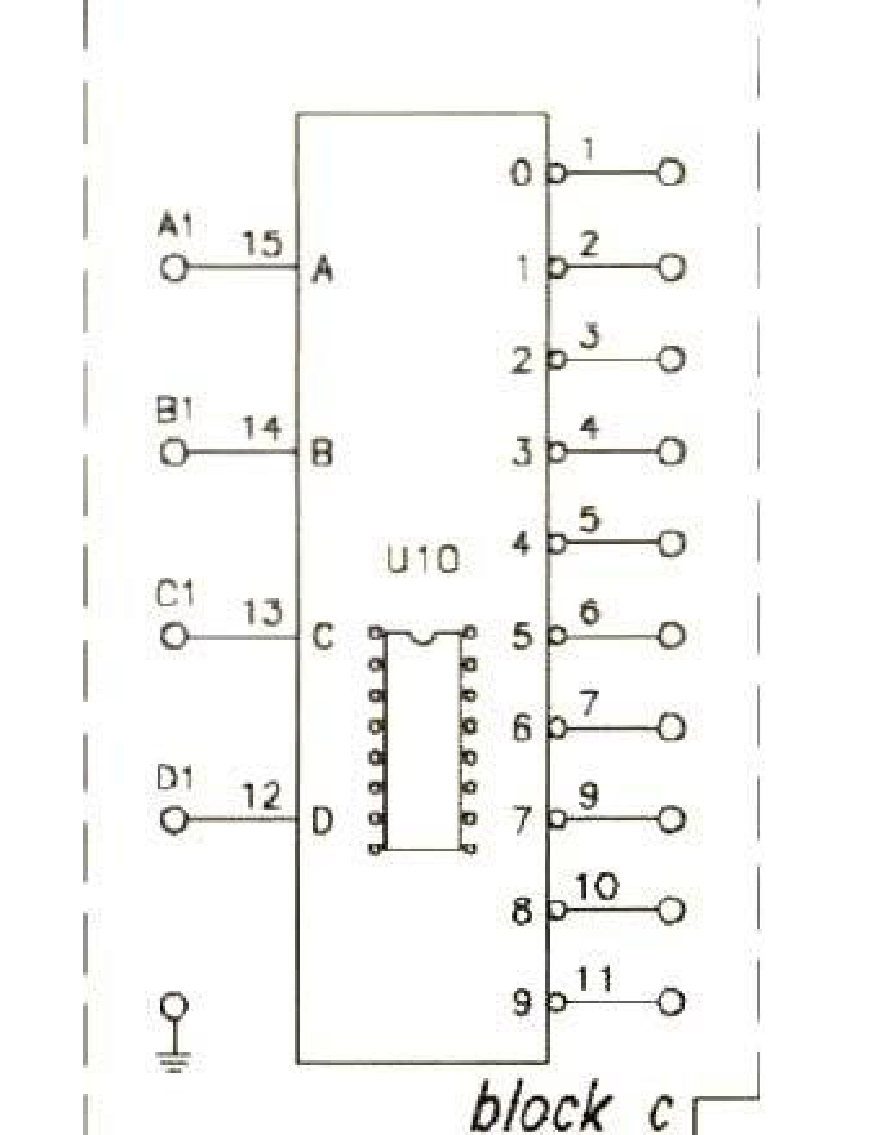


The result as following:

| Inputs | | Outputs | | | |
| --- | --- | --- | --- | --- | --- |
| B | A | F1 | F2 | F3 | F4 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

We conclude that BA decides which one from (D0, D1, D2, D3) will be the output, for example BA = 11 then the output will be F4 (D3).

We construct 4-to-10 decoder with TTL IC as following:

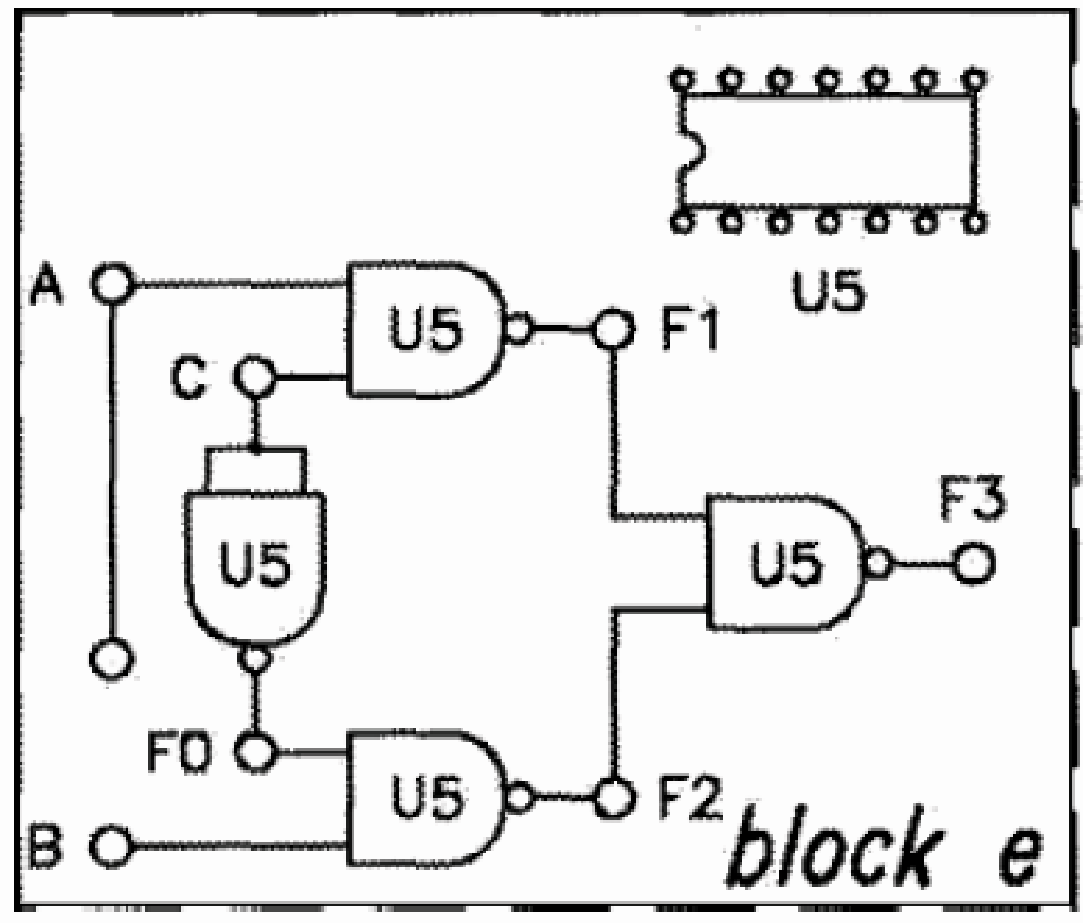


& the result is:

|  | Inputs | | | | Outputs | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BCD | D | C | B | A | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 3 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 9 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

We conclude that inputs active high, but the outputs active low, if we take 4 (0100) as example, we notice that just 4 in the output is 0, and that’s mean that output active low.

We construct 2-to-1 line multiplexer with basic gates as following:



The result as following:

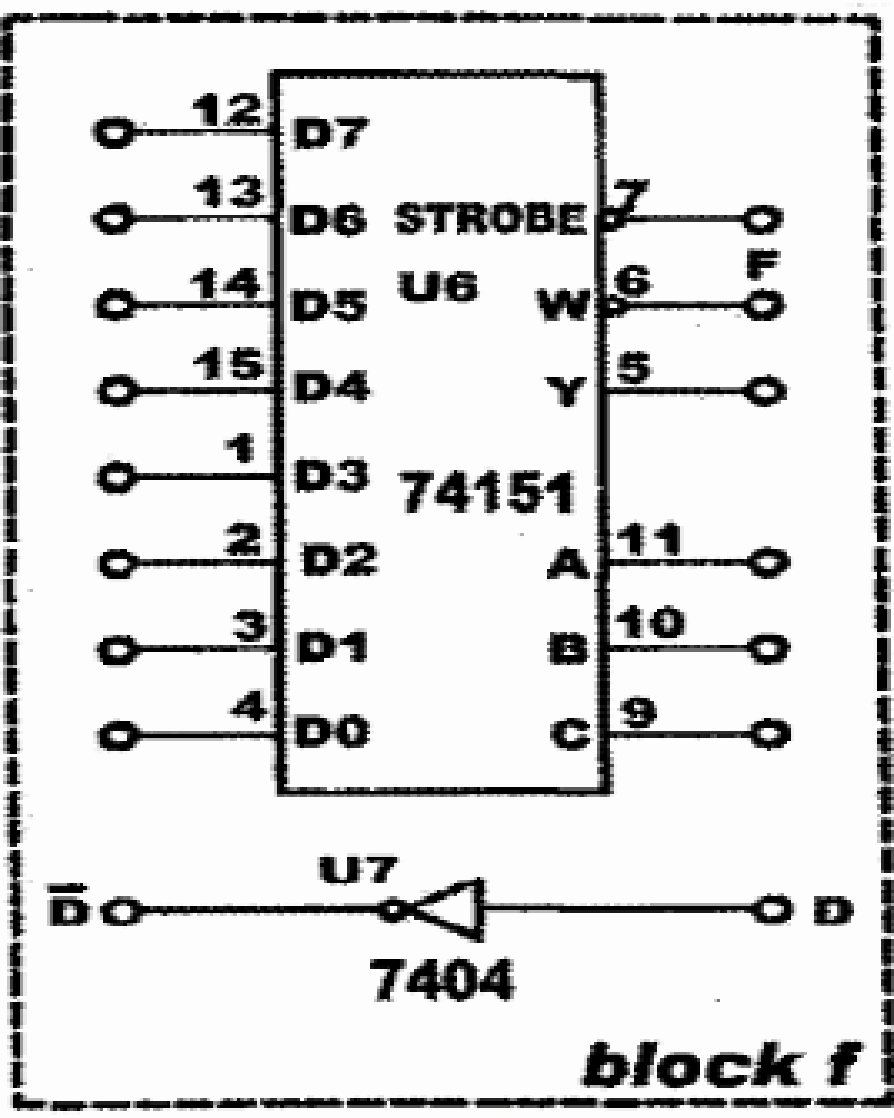
| Inputs | | | Output |
| --- | --- | --- | --- |
| C | B | A | F3 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

Which input (A or B) determines the output when C=0? B

Which input (A or B) determines the output when C=1? A

We notice that when C =1, the output values equal A values, which means that A values goes to output, & when C=0, B values goes to output.

We construct 8-to-1 line multiplexer using IC as following:



The previous circuit is 8x1 multiplexer, we connected as experiment instructions, so we connect D0 – D7 to data switches D0 – D7, and A,B,C to sw0,sw1,sw2, and we start to change the selectors (A,B,C) with changing data switches (D0-D7) and notice the outputs , the result was :

| Inputs | | | | Outputs | |
| --- | --- | --- | --- | --- | --- |
|  | C | B | A | Y | F |
| D0 | 0 | 0 | 0 | D0 | D0’ |
| D1 | 0 | 0 | 1 | D1 | D1’ |
| D2 | 0 | 1 | 0 | D2 | D2’ |
| D3 | 0 | 1 | 1 | D3 | D3’ |
| D4 | 1 | 0 | 0 | D4 | D4’ |
| D5 | 1 | 0 | 1 | D5 | D5’ |
| D6 | 1 | 1 | 0 | D6 | D6’ |
| D7 | 1 | 1 | 1 | D7 | D7’ |

NOTE: When CBA ="010", data at D2 is send to output Y. When CBA ="111", data at D7 is sent to output Y. The IC will function properly only when STROBE ="0". Y will remain “0” when STOROBE=”1”.

We conclude that if we change the CBA with the D values the output change depending on input value (1 or 0) if we take CBA = D5 = 101, if we send input D5 =1, then Y = 1(D5), and F = 0(D5’).

We use multiplexer to create logic function as following:



The function is: F = ( 0,2,4,5,7,8,10,11,15)

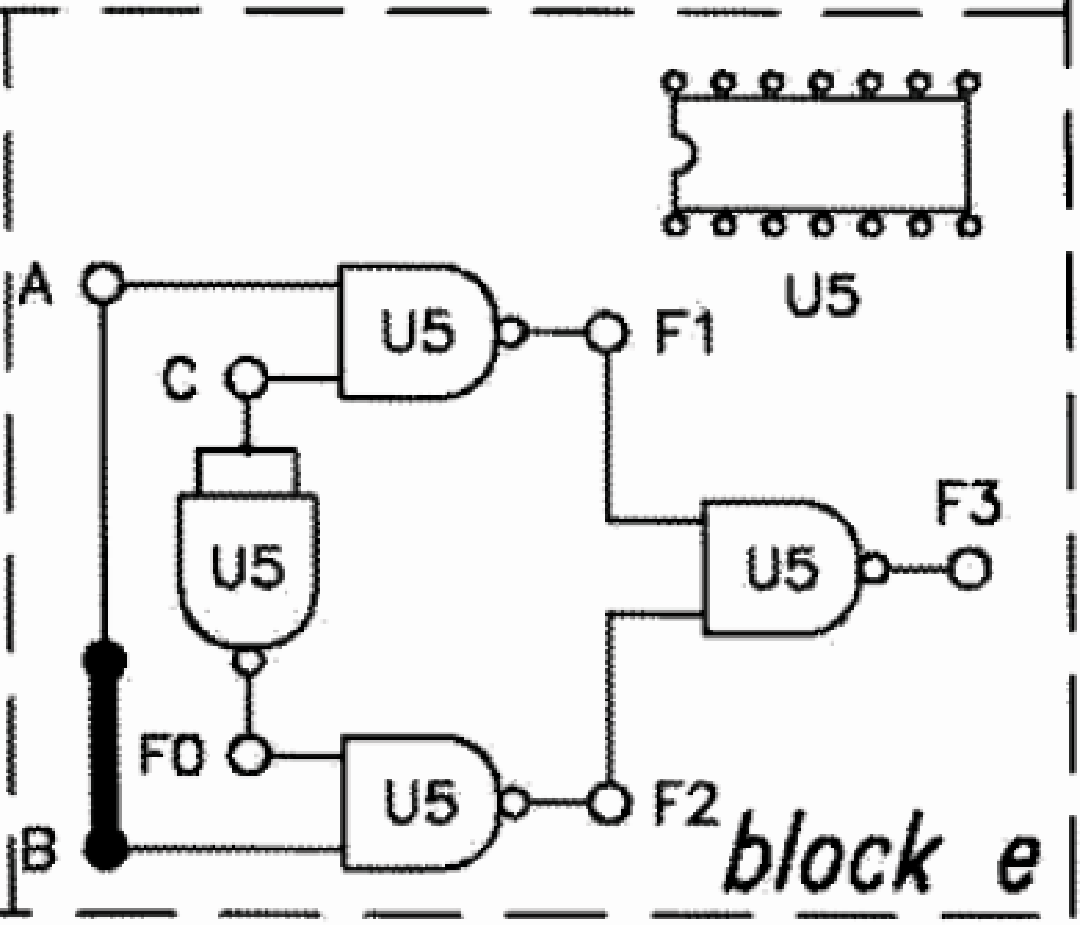
First of all, truth table for the function is:

| A | B | C | D | F |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

Then we connect D0, D1, D4 with D’, D2, D5 with 5v (1), D3, D7 with D, & D6 with 0.

We observe that we can reduce the number of selection lines to use smaller multiplexer (4x1), to create logic function.

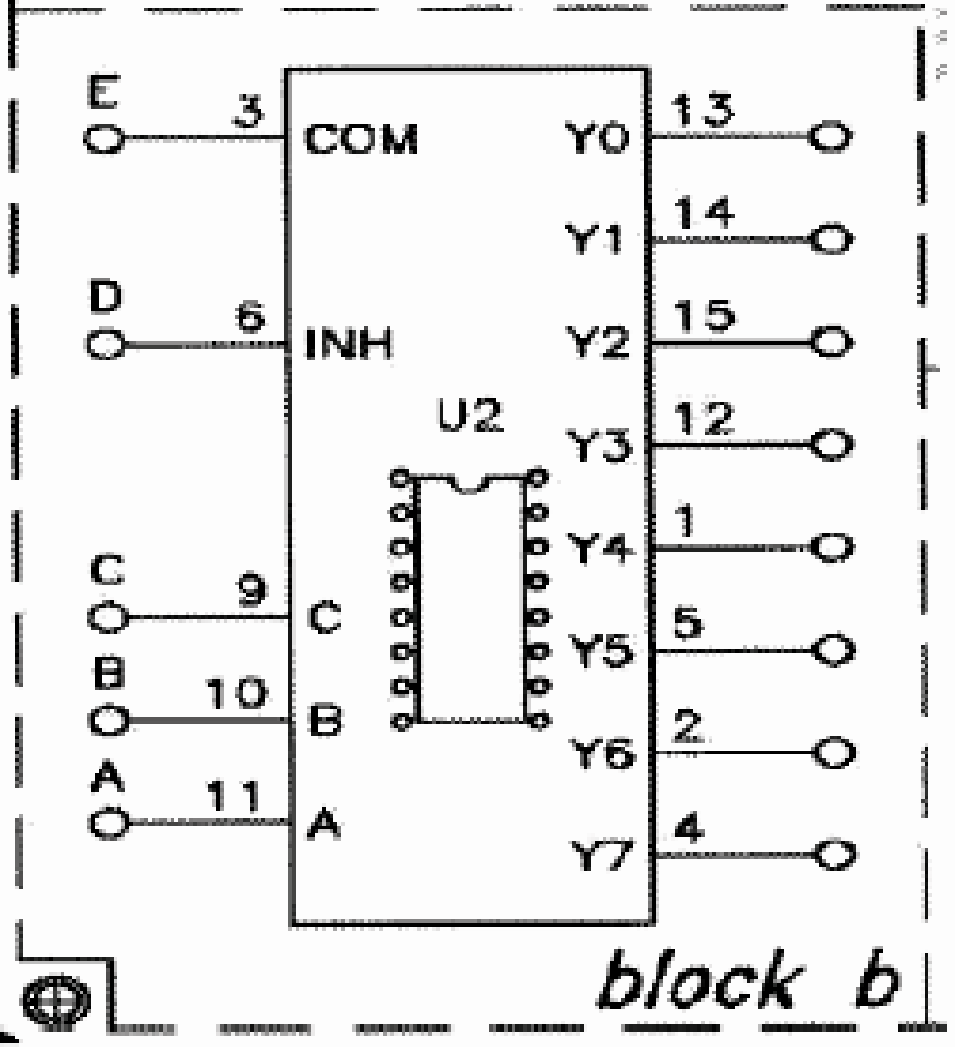
\* We use basic gates to construct demultiplexer as following:



Then we got the result:

| Inputs | | Outputs | |
| --- | --- | --- | --- |
| C | A | F1 | F2 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |

\* We construct demultiplexer using COMS IC as following:



\* Set D=”0”, apply the sequence 1-0-1-0 to the Common Input E and observe output Y0~Y7.Do the outputs change as the input sequence is? Yes

\* Set D="1", apply the input sequence 1-0-1-0 to the Common Input E and observe outputs Y0~Y7.

\*Do the outputs change as the input sequence is applied? No

\*Which state of D changes the outputs? When D =0

Because the output does not change when D=1, we set D=0, and apply the sequence of E (1-0-1-0) & the result was:

| Inputs | | | Outputs | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C | B | A | Y0 | Y1 | Y2 | Y3 | Y4 | Y5 | Y6 | Y7 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

* **Conclusion:**

In this experiment we understand how to use basic gates and IC to build decoder, encoder, multiplexer & demultiplexer and recognize how they work,

For example if we have decoder with n input, then we have 2^n output...etc.

Also we learned about some circuit that active low such as 4-to-10 (BCD) decoder, & understand why active low in some circuit is important (because it’s save power), in addition to that we learned how to implement function using decoder and multiplexer.

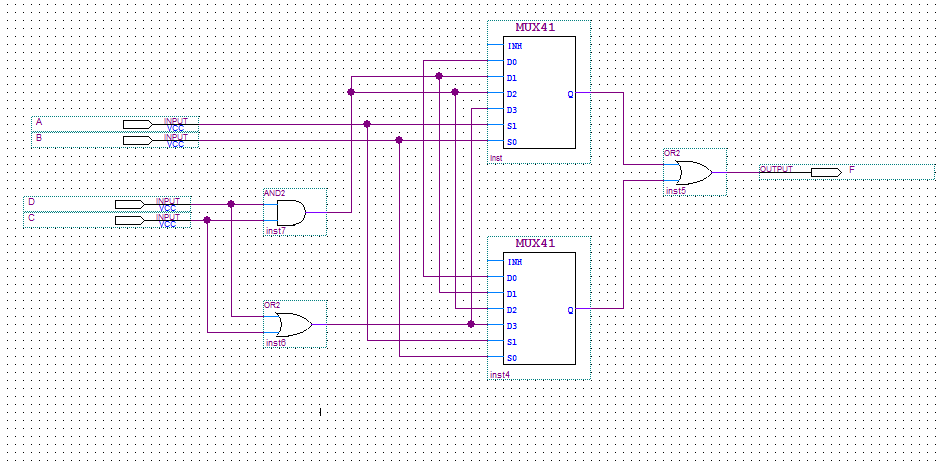
* **Logic problem:**

Design a Majority Circuit; a circuit that takes 4 inputs and 1 output, its output equals 1 when 3 or 4 of the inputs are 1. ***You can only use two 4×1 multiplexers.***

Step 1: truth table

| Inputs | | | | output |
| --- | --- | --- | --- | --- |
| A | B | C | D | F |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Then we can start design using two or gate, one and gate, & two multiplexer as following:



AB as selectors, D0 = 0, D1=D2 = CD, D3 = C+D

**\*\*PRELAB:**

**Design a circuit which uses an SN74151 to implement a sum-of-products expression, as follows:**

**(a) Convert the following expression into summation form (i.e. F(A,B,C)= Σ(…)):**

Y= AB’ +B’C

= AB’ (C+C’) + B’C (A+A’)

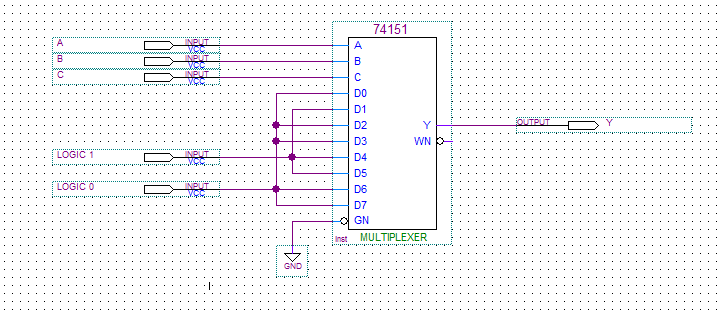
= AB’C+AB’C’+AB’C+A’B’C

= AB’C + AB’C’+A’B’C

Y = ∑ (1, 4, 5)

**(b) Sketch on figure.1 the input connections necessary to implement the function in (a).**

| **A** | **B** | **C** | **F** |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

****

**Design a circuit which uses an SN74138 demultiplexer to implement a sum- of-products expression, as follows:**

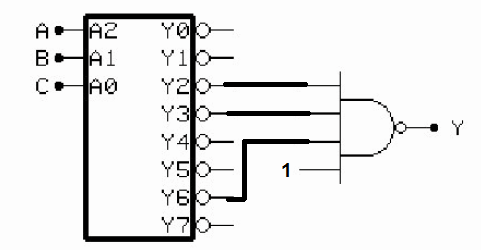
**(a) Convert the following expression into summation form (i.e. F(A,B,C)= Σ(…)):**

Y = A’BC + BC’

= A’BC + BC’ (A+A’)

= A’BC + ABC’+A’BC’

Y = ∑ (2, 3, 6)



References:

<http://www.allaboutcircuits.com/textbook/digital/chpt-9/decoder/>