



Jiangxi University of Science and Technology

DIGITAL DESIGN

Chapter 4 Combinational Logic_3 Review and Basics of Multiplexers:



Decoders and Encoders

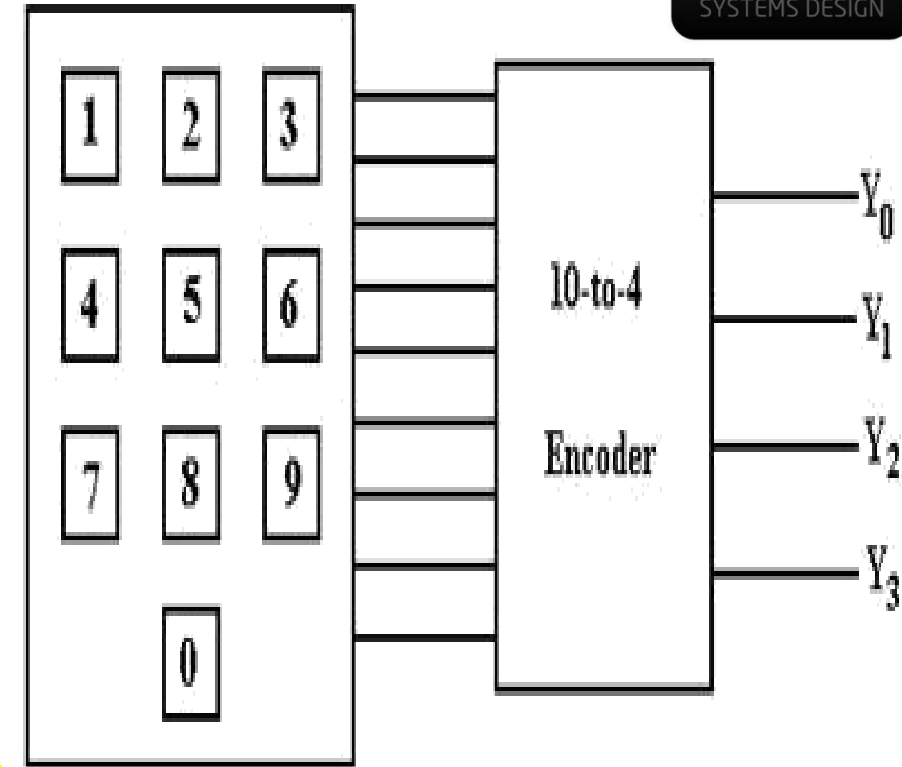
- These perform the functions suggested by the corresponding decimal-binary conversions. In conversion of a decimal number to binary, we obtain the binary equivalent of the number. An encoder has a number of inputs, usually a power of two, and a set of outputs giving the binary code for the “number” of the input.

Consider a classic 2^N -to- N encoder. The inputs are labeled I_0, I_1, \dots, I_K , where $K = 2^N - 1$. The assumption is that only one of the inputs is active; in our way of thinking only one of the inputs is 1 and the rest are 0. Suppose input J is 1 and the rest are 0. The output of the circuit is the binary code for J . Suppose a 32-to-5 encoder with input 18 active. The output Z is the binary code 10010; $Z_4 = 1, Z_3 = 0, Z_2 = 0, Z_1 = 1$, and $Z_0 = 0$.

Encoders

- Common encoders include 8-to-3, 16-to-4, and 32-to-5.
- One common exception to the rule of 2^N -to- N is a 10-to-4 encoder, which is used because decimal numbers are so common.
- Note that three binary bits are not sufficient to encode ten numbers, so we must use four bits and not produce the outputs 1010, 1011, 1100, 1101, 1110, or 1111.
- We now present a detailed discussion and a design of a 10-to-4 encoder. We begin with a diagram that might illustrate a possible use of an encoder.

In this example, the key pad has ten keys, one for each digit. When a key is pressed, the output line corresponding to that key goes to logic 1 (5 volts) and the other output lines stay at logic 0 (0 volts). Note that there are ten output lines from the key pad, one for each of the keys. These ten output lines form ten input lines into the 10-to-4 encoder.



Encoders

- The 10-to-4 encoder outputs a binary code indicating which of the keys has been pressed. In a complete design, we would require some way to indicate that no key has been pressed. For our discussion, it is sufficient to ignore this common case and assume that a key is active.
- We now present a table indicating the output of the encoder for each input. In this example, we assume that at any time exactly one input is active.

Encoders

Input	Y_3	Y_2	Y_1	Y_0
X_0	0	0	0	0
X_1	0	0	0	1
X_2	0	0	1	0
X_3	0	0	1	1
X_4	0	1	0	0
X_5	0	1	0	1
X_6	0	1	1	0
X_7	0	1	1	1
X_8	1	0	0	0
X_9	1	0	0	1

In the table at left, we label the inputs X_0 through X_9 , inclusive. To produce the equations for the outputs, we reason as follows.

Y_3 is 1 when either $X_8 = 1$ or $X_9 = 1$.

Y_2 is 1 when $X_4 = 1$ or $X_5 = 1$ or $X_6 = 1$ or $X_7 = 1$.

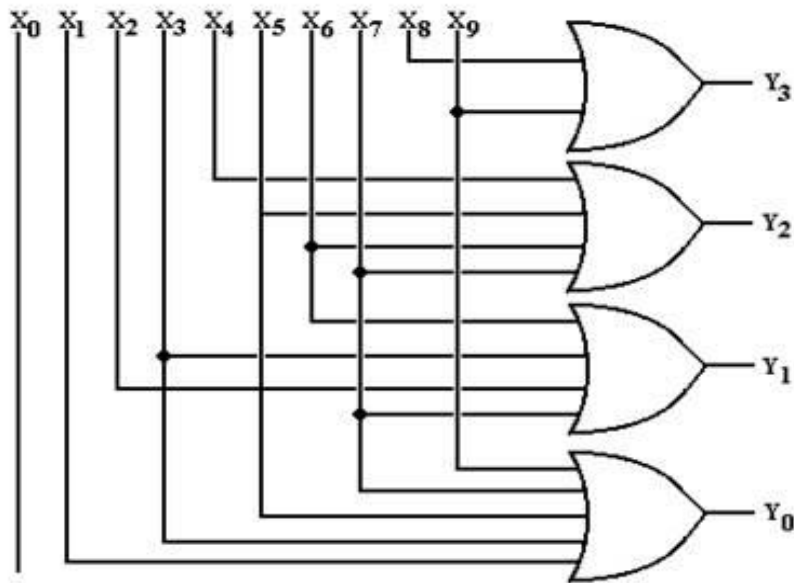
Y_1 is 1 when $X_2 = 1$, $X_3 = 1$, $X_6 = 1$, or $X_7 = 1$.

Y_0 is 1 when $X_1 = 1$, $X_3 = 1$, $X_5 = 1$, $X_7 = 1$, or $X_9 = 1$.

Encoders

- These observations lead to the following equations, used to design the encoder.

- $$\begin{aligned} Y_3 &= X_8 + X_9 \\ Y_2 &= X_4 + X_5 + X_6 + X_7 \\ Y_1 &= X_2 + X_3 + X_6 + X_7 \\ Y_0 &= X_1 + X_3 + X_5 + X_7 + X_9 \end{aligned}$$



The note that input X_0 is not connected to an output. This gives rise to the following problem for the circuit: how does one differentiate between X_0 being active and no input being active. That might be a problem for real encoder design.

The most straightforward modification of the circuit would be to create the logical OR of the ten inputs and pass that signal as a “key pressed” signal. We mention this only to show that some applications must handle this case; we shall not consider it further in this course.

Encoders

- Another issue with encoders is what to do if two or more inputs are active. For ‘plain” encoders the output is not always correct; for example, in the above circuit with inputs X_3 and X_5 active would output $Y_3 = 0$, $Y_2 = 1$, $Y_1 = 1$, and $Y_0 = 1$.
- Priority encoders are designed to avoid the problem of multiple inputs by implementing a priority order on the inputs and producing the output for the input that has priority.
For example, in a 32-to-5 priority encoder, having inputs 18 and 29 active would produce either the binary code 10010 (for 18) or 11101 (for 29), depending on the priority policy, and not the output 11111 that a plain encoder would produce.

Decoders



- An N -to- 2^N decoder does just the opposite, taking an N bit binary code and activating the output labeled with the corresponding number.
- Consider a 4-to-16 decoder with outputs labeled Z_0, Z_1, \dots, Z_{15} .
- Suppose the input is $I_3 = 1, I_2 = 0, I_1 = 0$, and $I_0 = 1$ for the binary code 1001.
- Then output Z_9 is active and the other outputs are not active.
- Again, the main exception to the N -to- 2^N rule for decoders is the 4-to-10 decoder, which is a common circuit.
- Note that it takes 4 bits to encode 10 items, as 3 bits will encode only 8.
- prefer be to use a 4-to-16 decoder and ignore some of the outputs, but this author does not establish commercial practice.
- The main advantage is that the 4-to-10 decoder chip would have 6 fewer pins than a 4-to-16 decoder; a 16-pin chip is standard and cheaper to manufacture than a 22-pin chip.

Decoders

- The decoder is based on the association of binary numbers to decimal numbers, as is shown in the figure at right.
- Since this is a 3-to-8 decoder, we have three inputs, labeled X_2 , X_1 , and X_0 ; and eight outputs, labeled Y_7 , Y_6 , Y_5 , Y_4 , Y_3 , Y_2 , Y_1 , and Y_0 .
- The observation that leads to the design of the decoder is the obvious one that the values of X_2 , X_1 , and X_0 determine the output selected. For this part of the discussion, I choose to ignore the enable input.

Decimal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

the 3-to-8 active-high decoder.

- The following Boolean equations determine the 3-to-8 decoder.

$$Y_0 = \overline{X_2} \cdot \overline{X_1} \cdot \overline{X_0}$$

$$Y_4 = X_2 \cdot \overline{X_1} \cdot \overline{X_0}$$

$$Y_1 = \overline{X_2} \cdot \overline{X_1} \cdot X_0$$

$$Y_5 = X_2 \cdot \overline{X_1} \cdot X_0$$

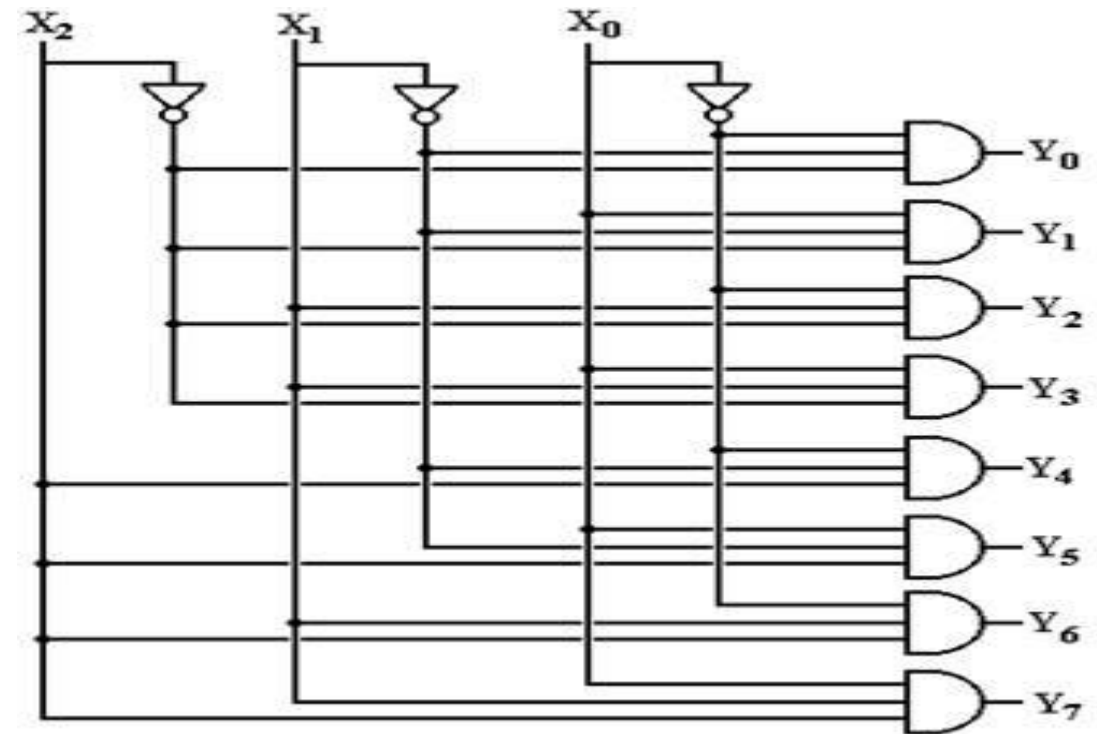
$$Y_2 = \overline{X_2} \cdot X_1 \cdot \overline{X_0}$$

$$Y_6 = X_2 \cdot X_1 \cdot \overline{X_0}$$

$$Y_3 = \overline{X_2} \cdot X_1 \cdot X_0$$

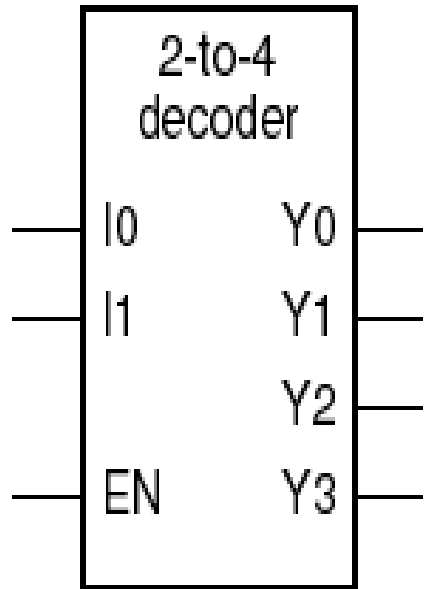
$$Y_7 = X_2 \cdot X_1 \cdot X_0$$

the circuit diagram for the 3-to-8 active-high decoder.



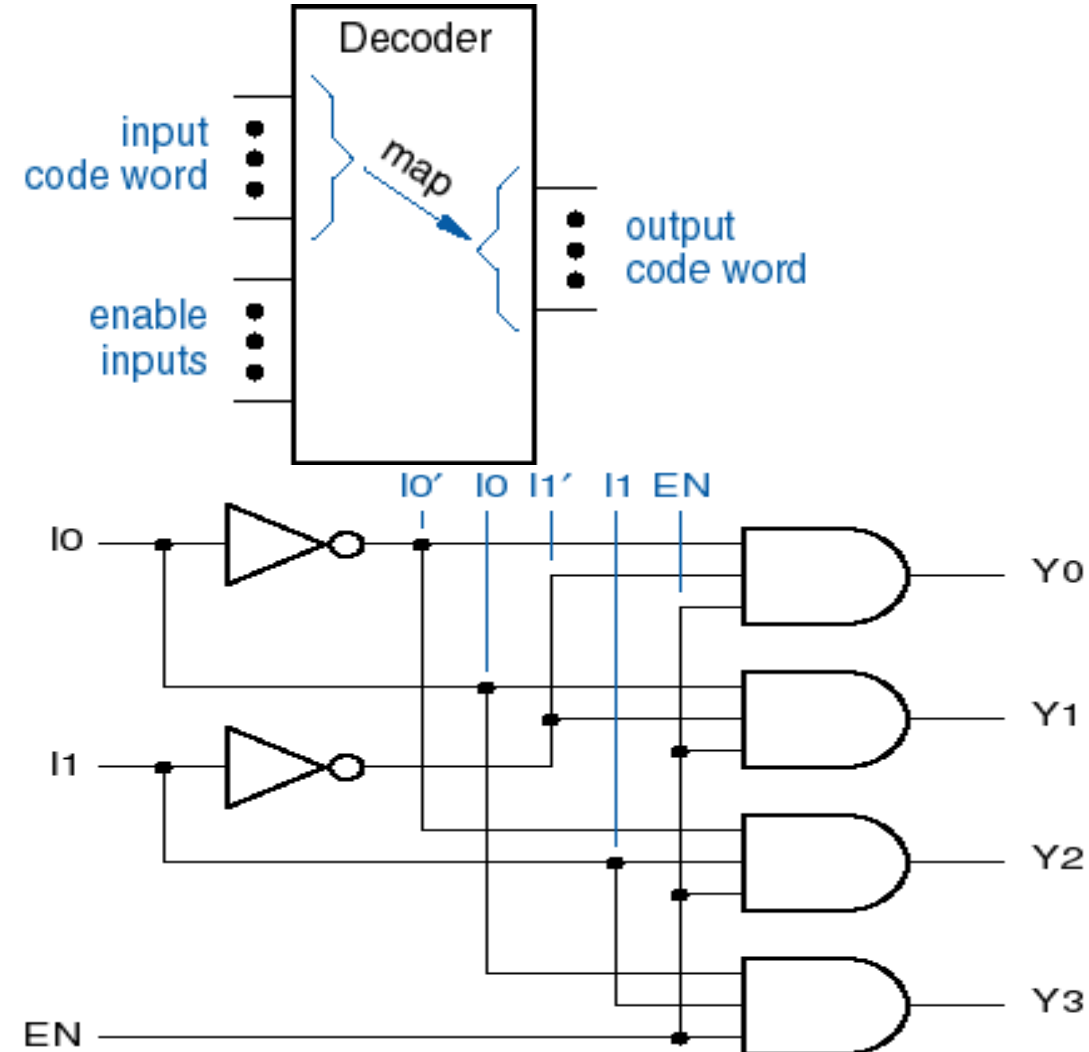
Decoders

Typically n inputs, 2^n outputs: 2-to-4, 3-to-8, 4-to-16, etc.

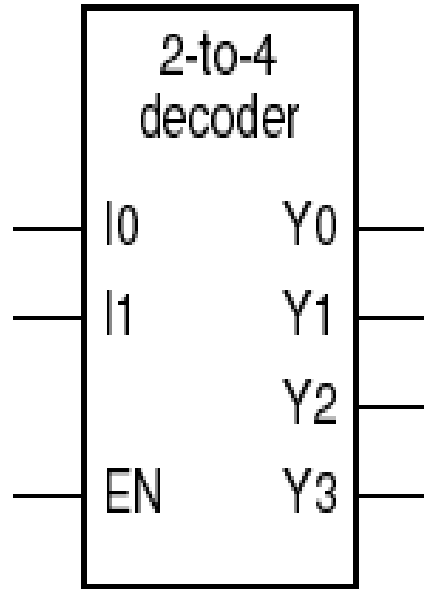


Inputs			Outputs			
EN	I1	I0	Y3	Y2	Y1	Y0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Note “x” (don’t care) notation.

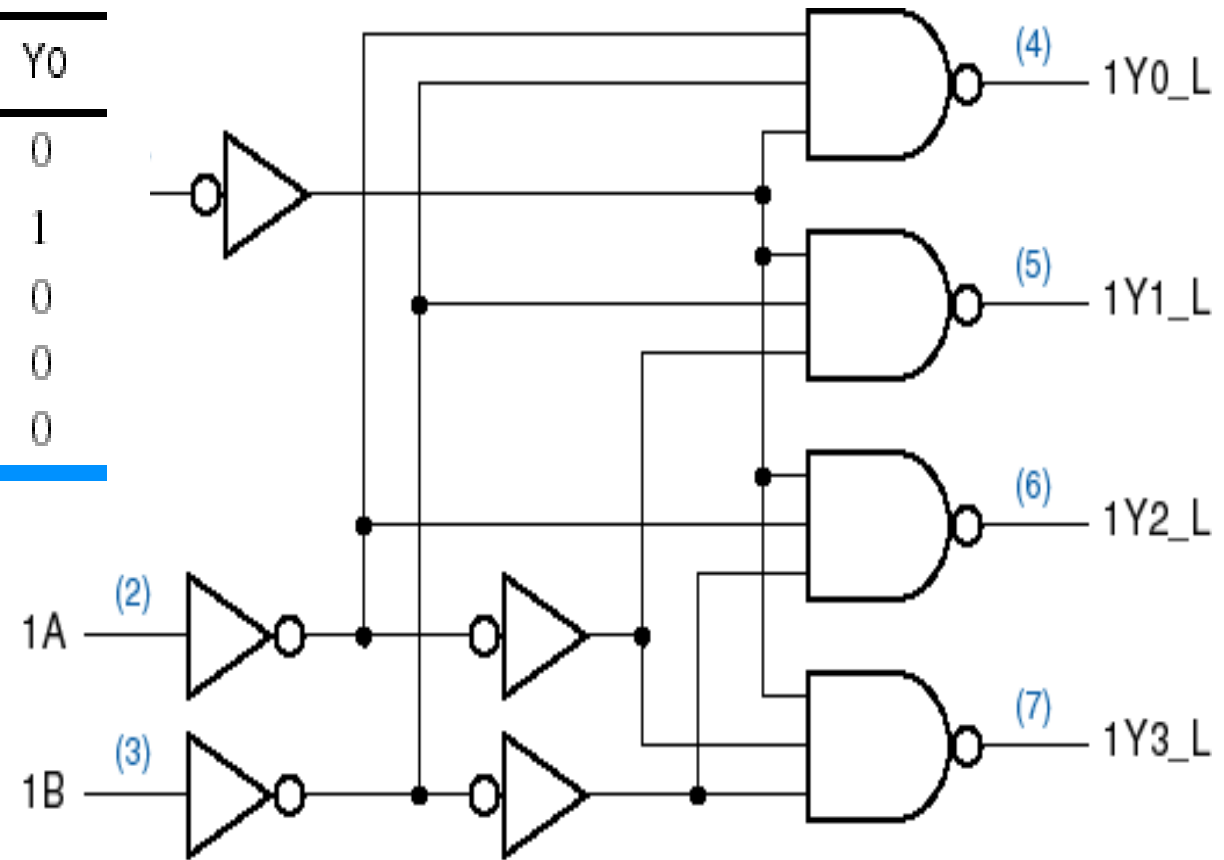


2-to-4 decoder



Inputs			Outputs			
EN	I1	I0	Y3	Y2	Y1	Y0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Note “x” (don’t care) notation.

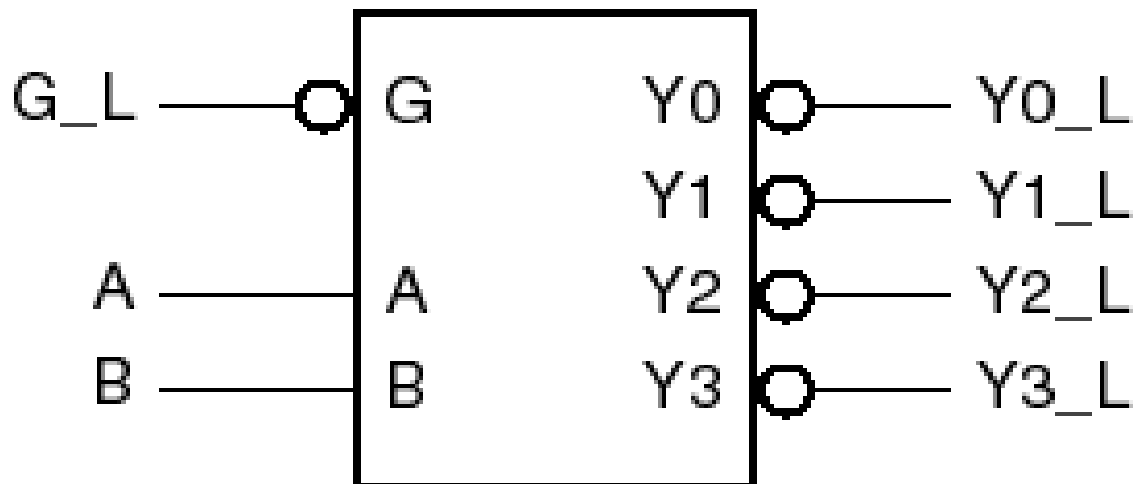


- Input buffering (less load)
- NAND gates (faster)

Decoder Symbol

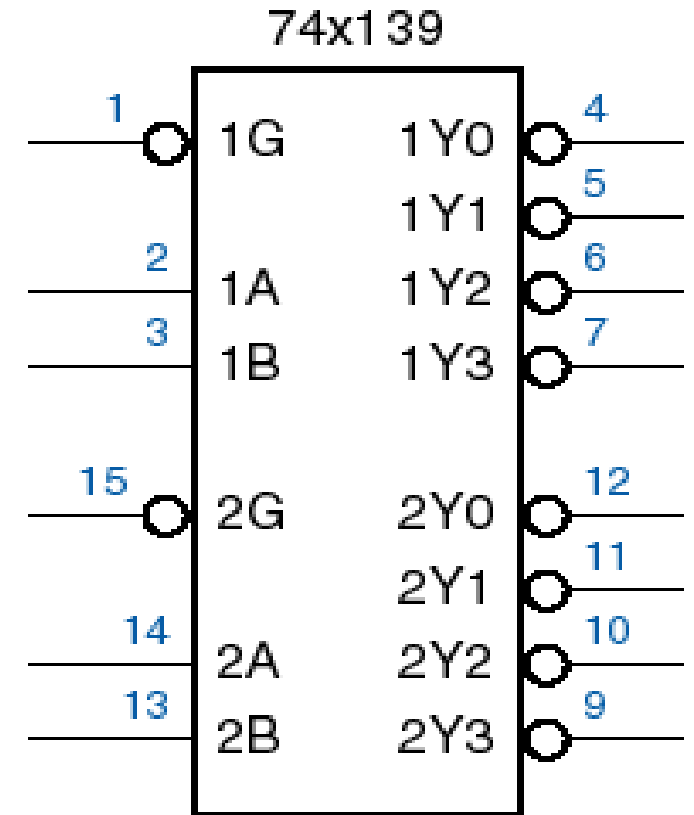
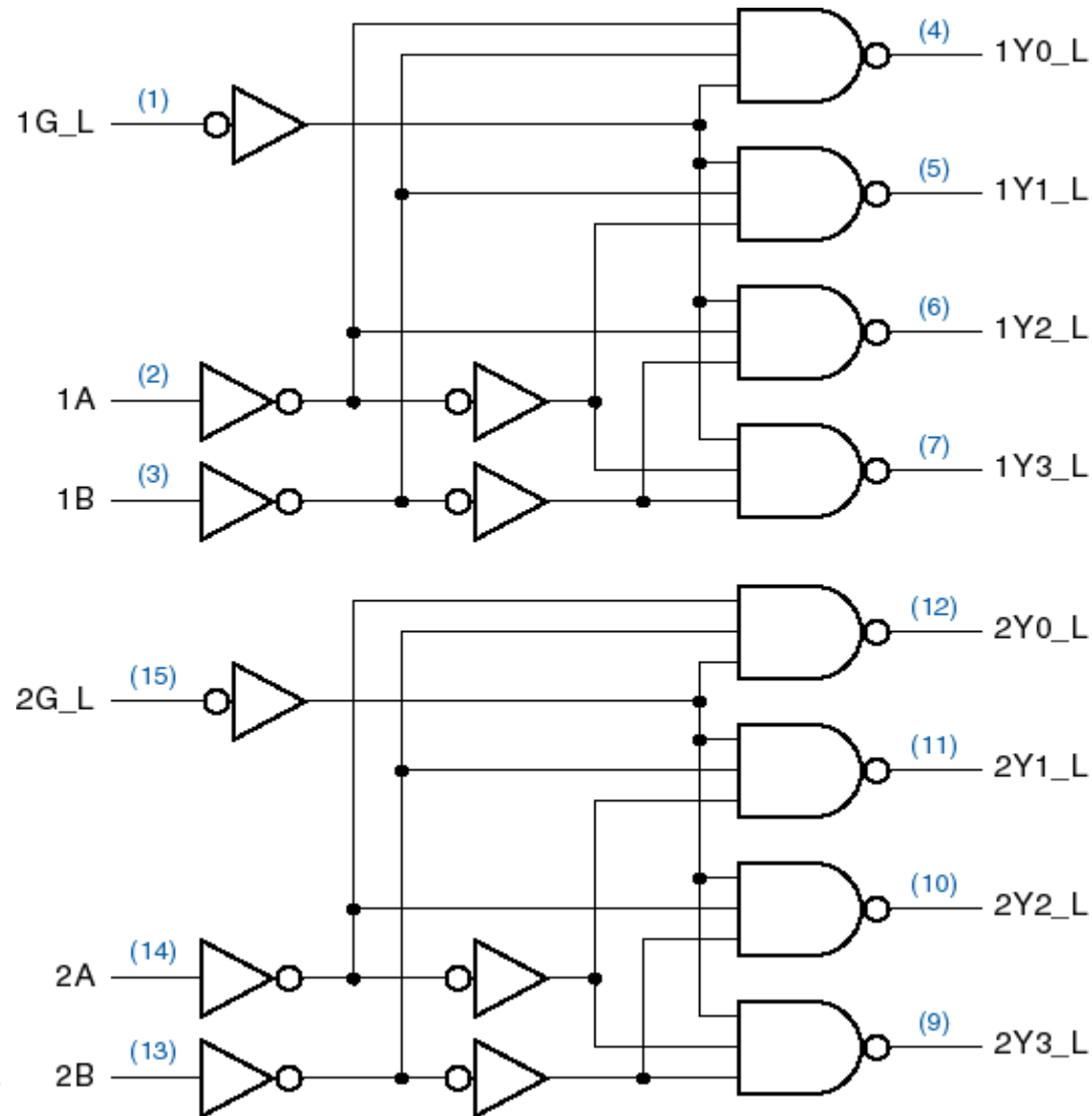


1/2 74x139

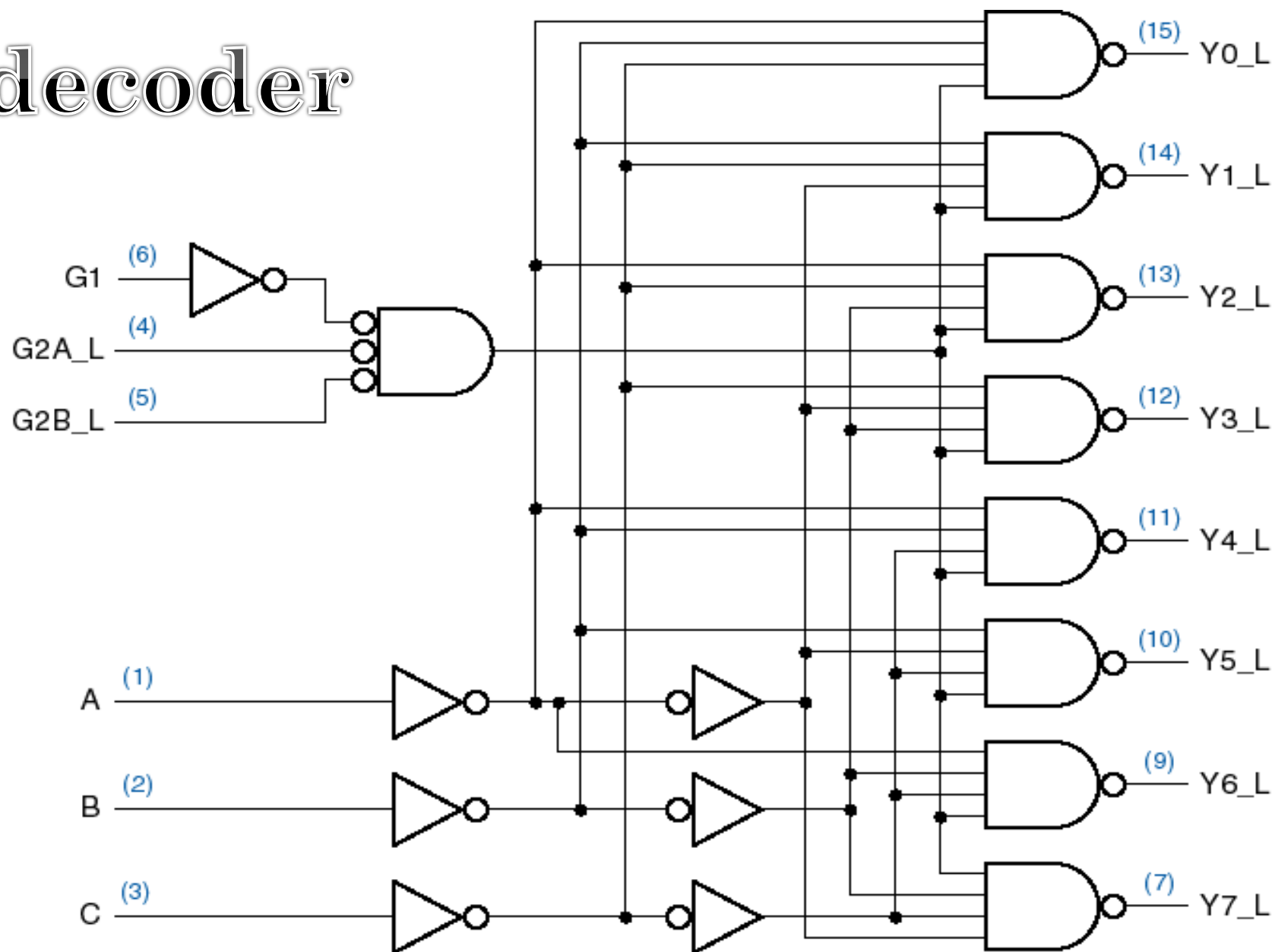


A	B	Y0	Y1	Y2	Y3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

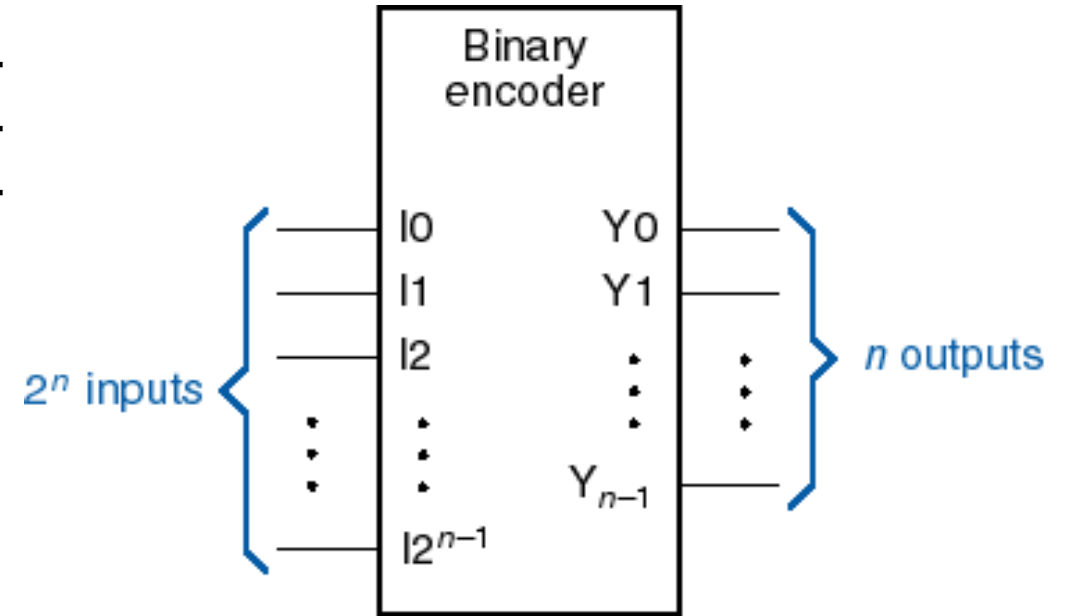
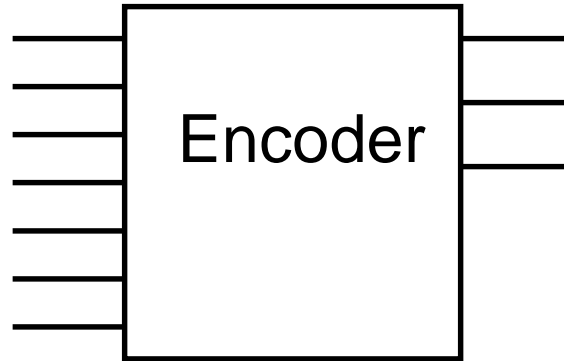
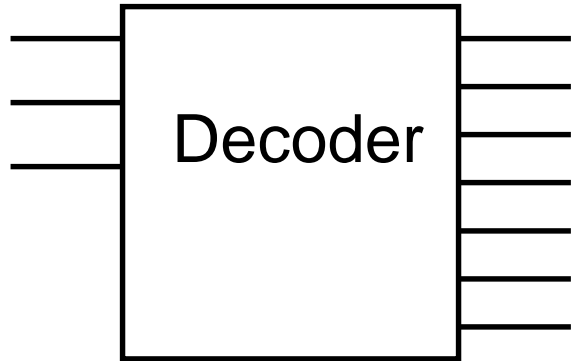
Complete 74x139 Decoder



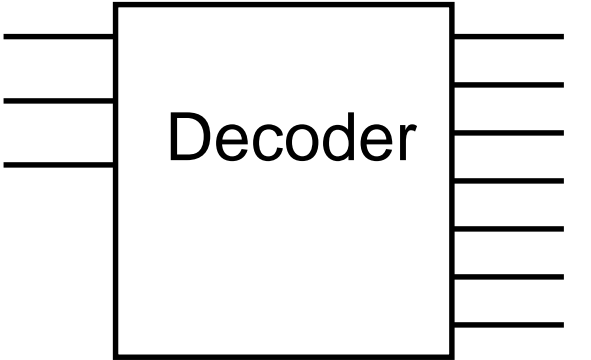
3-to-8 decoder



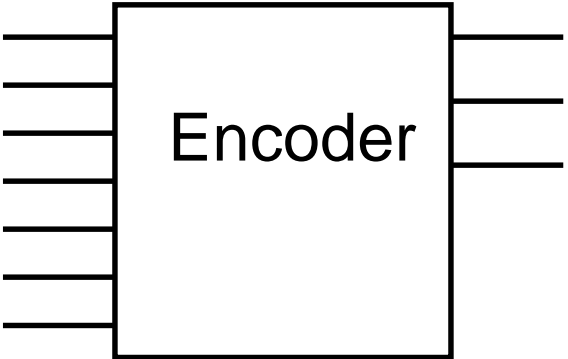
Encoders vs. Decoders



Input			Output							Decoder	
X	Y	Z	D0	D1	D2	D3	D4	D5	D6	D7	
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	1	0	0	0	0	0	0	0	1	



Input								Output			Encoder	
D0	D1	D2	D3	D4	D5	D6	D7	X	Y	Z		
1	0	0	0	0	0	0	0	0	0	0		
0	1	0	0	0	0	0	0	0	0	1		
0	0	1	0	0	0	0	0	0	1	0		
0	0	0	1	0	0	0	0	0	1	1		
0	0	0	0	1	0	0	0	1	0	0		
0	0	0	0	0	1	0	0	1	0	1		
0	0	0	0	0	0	1	0	1	1	0		
0	0	0	0	0	0	0	1	1	1	1		



Multiplexers and Demultiplexers

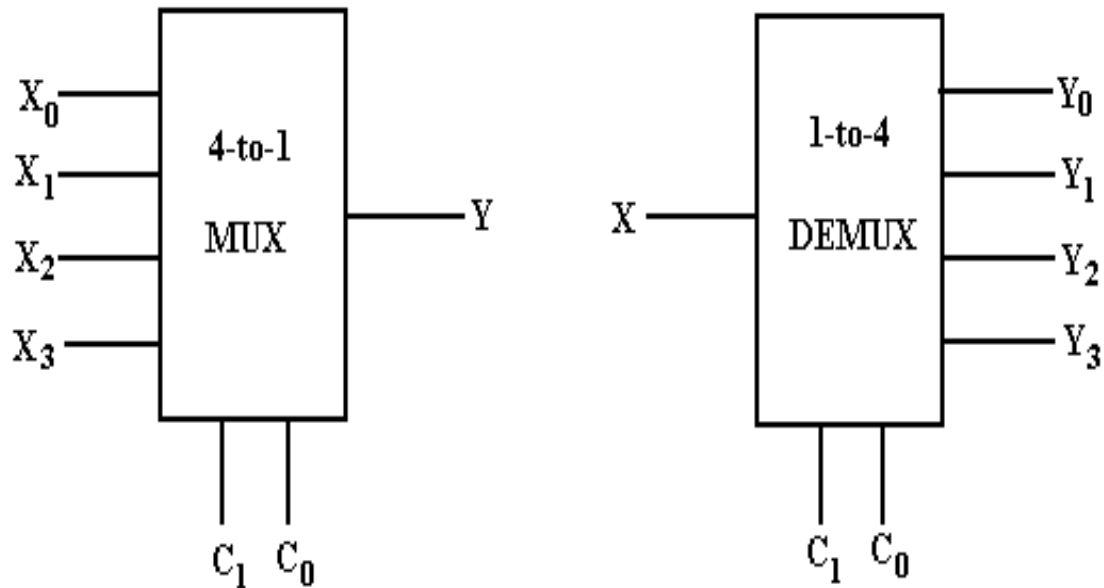
- A multiplexer has a number of inputs (usually a power of two), a number of control signals, and one output.
- A demultiplexer has one input signal, a number of control signals, and a number of outputs, also usually a power of two.
- We consider here a 2^N –to–1 multiplexer and a 1–to– 2^N demultiplexer.

Circuit	Inputs	Control Signals	Outputs
Multiplexer	2^N	N	1
Demultiplexer	1	N	2^N

The action of each of these circuits is determined by the control signals. For a multiplexer, the output is the selected input. In a demultiplexer, the input is routed to the selected output.

Multiplexers and Demultiplexers

- As examples, we show the diagrams for both a four-to-one multiplexer (MUX) and a one-to-four demultiplexer (DEMUX).



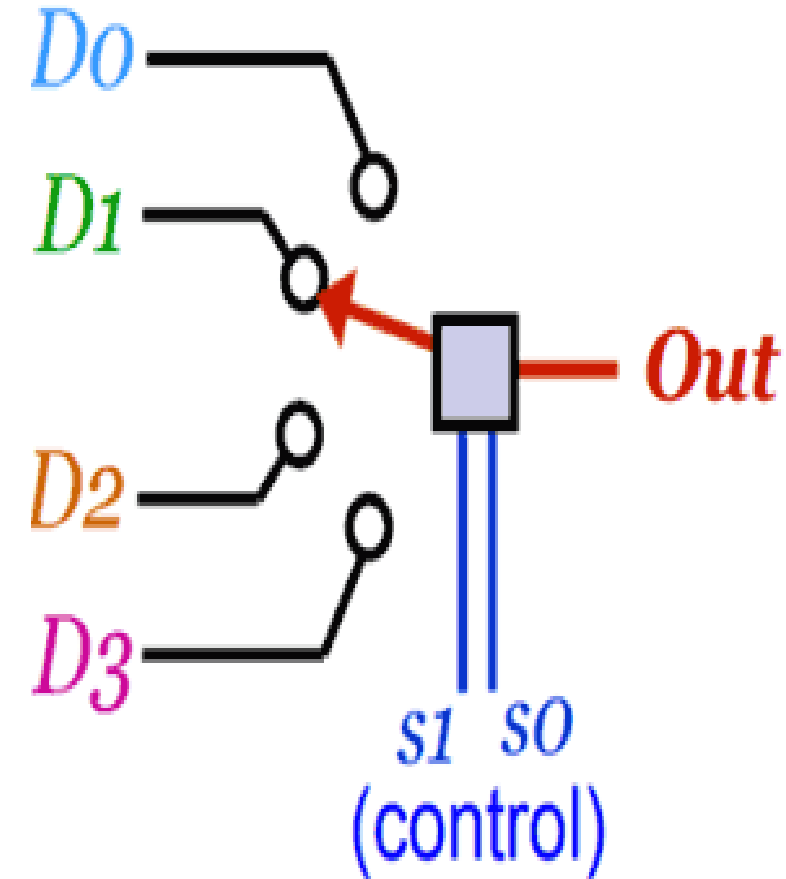
Note that each of the circuits has two control signals.

For a multiplexer, the N control signals select which of the 2^N inputs will be passed to the output.

For a demultiplexer, the N control signals select which of the 2^N outputs will be connected to the input.

Basics of Multiplexers:

- The best way to understand Multiplexers is by looking at a single pole multi-positioned as shown below.
- **Here the switch has multiple inputs D0, D1, D2 and D3 but it has only one Output (Out) pin.**
- The Control knob is used to select one of the four available data and this data will be reflected on the output side. This way the user can select the required the signal among many available signals.



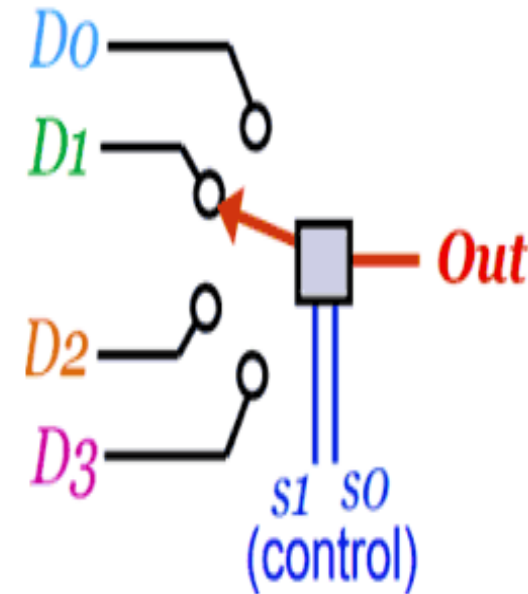
Basics of Multiplexers:

- This is a plain example of a mechanical Multiplexer. But in electronic circuit which involves high speed switching and data transfers we should be able to select the required input very fast using digital circuits. The Control signals (S1 and S0) does exactly the same, they select one input of the many available ones base on the signal provided to them. So the three basic and bare minimum terms on any Multiplexer will be Input Input Pins, Output Pin and Control Signal
- **Input Pins:** These are the available signal pins from which one has to be selected. These signals can either be a digital signal or an analog signal.
- **Output Pin:** A multiplexer will always have only one output pin. The selected input pin signal will be provided by the output pin.
- **Control/Selection Pin:** The Control Pins are used to select the input pin signal. The number of Control pins on a Multiplexer depends on the number of input pins. For example a 4-input multiplexer will have 2 signal pins.
- This is a plain example of a mechanical Multiplexer. But in electronic circuit which involves high speed switching and data transfers we should be able to select the required input very fast using digital circuits. The Control signals (S1 and S0) does exactly the same, they select one input of the many available ones base on the signal provided to them. So the three basic and bare minimum terms on any Multiplexer will be Input Input Pins, Output Pin and Control Signal

Basics of Multiplexers:

- For understanding purpose, let us consider a 4-input multiplexer that is shown above. It has two control signal using which we can select one of the available four input lines. The truth Table below illustrates the status of Control pins (S0 and S1) for selecting the required Input pin.

S1	S0	Out
0	0	D0
0	1	D1
1	0	D2
1	1	D3



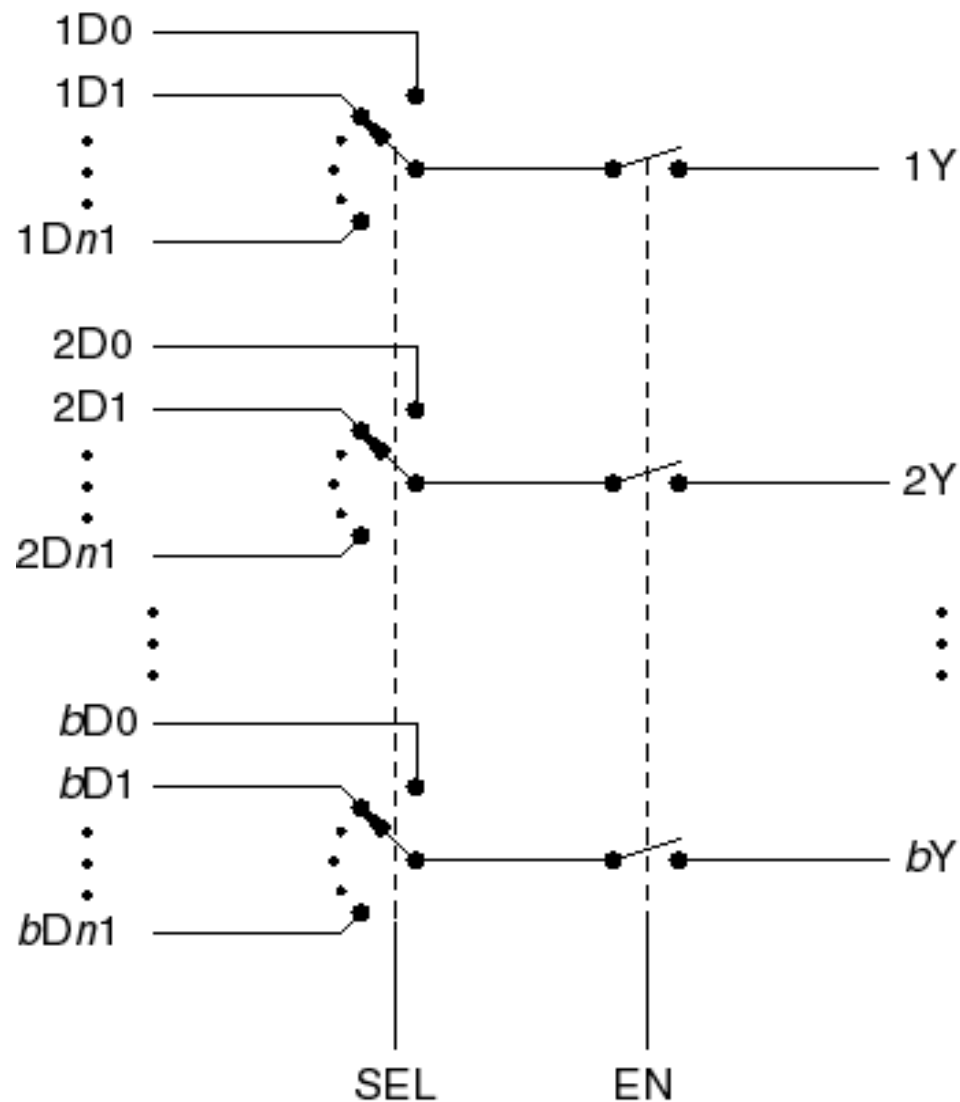
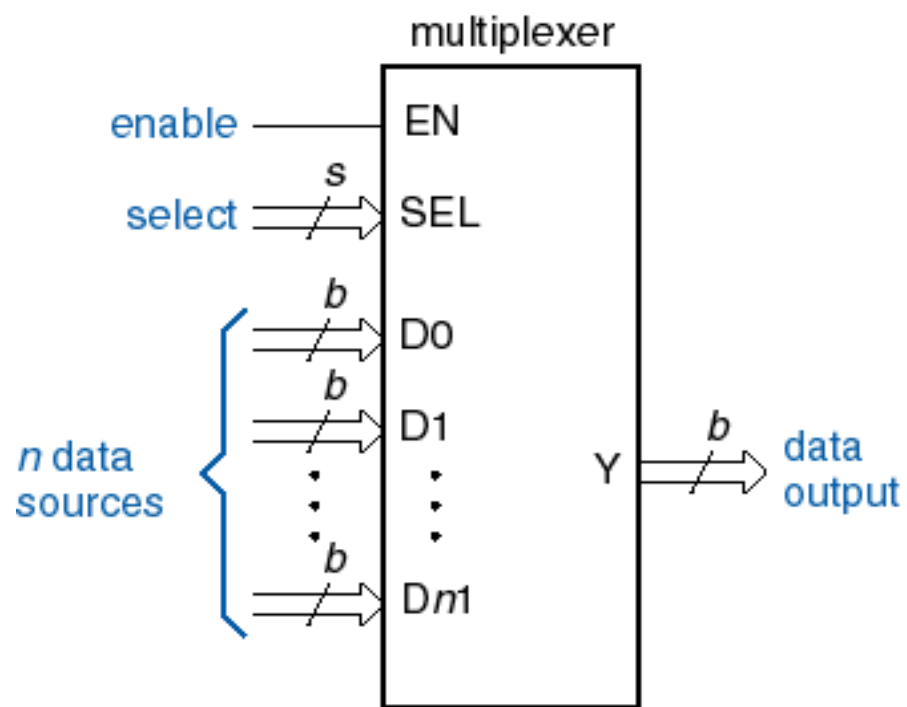
Multiplexers and Demultiplexers

- The output for a multiplexer can be represented as a Boolean function of the inputs and the control signals.
- As an example, we consider a 4-input multiplexer, with control signals labeled C_0 and C_1 and inputs labeled I_0 , I_1 , I_2 , and I_3 . The output can be described as a truth table or algebraically.
- Note that each of the truth tables and algebraic expression shows the input that is passed to the output.
- The truth table is an abbreviated form of the full version, which as a table for independent variables C_0 , C_1 , I_0 , I_1 , I_2 , and I_3 would have 64 rows.

C_1	C_0	M
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

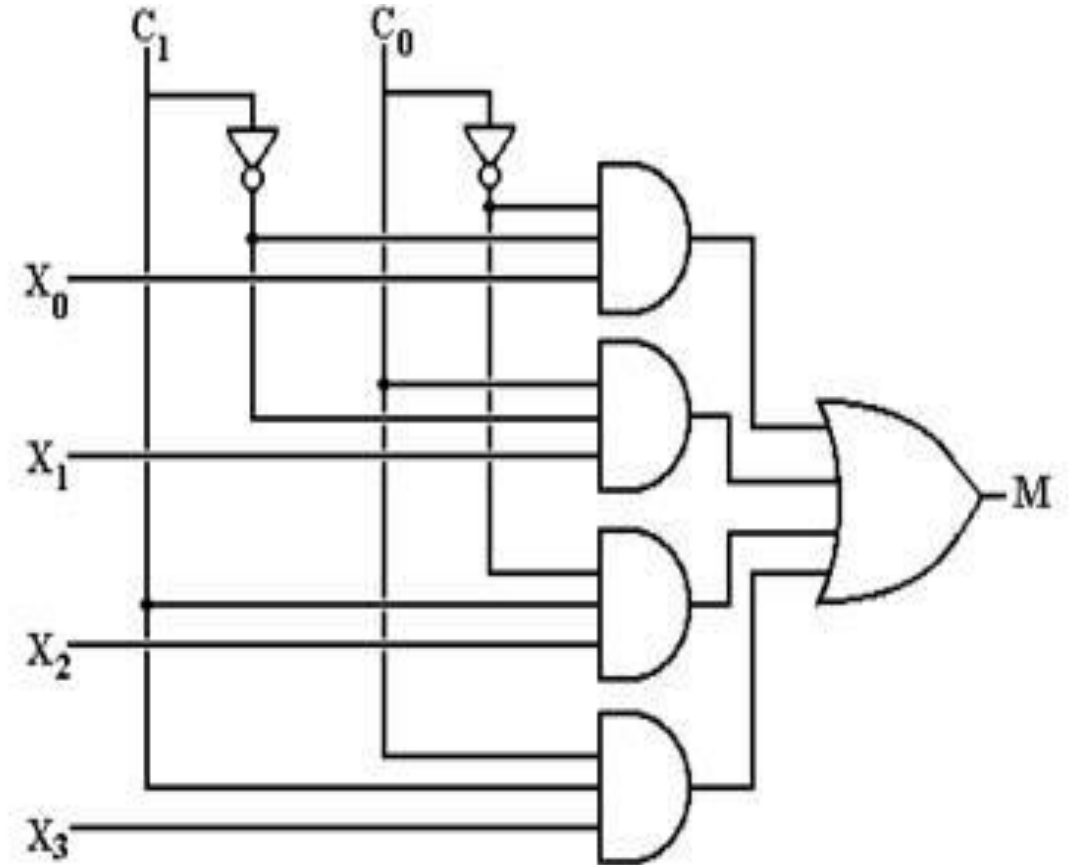
$$M = C_1' \bullet C_0' \bullet I_0 + C_1' \bullet C_0 \bullet I_1 + C_1 \bullet C_0' \bullet I_2 + C_1 \bullet C_0 \bullet I_3$$

Multiplexers

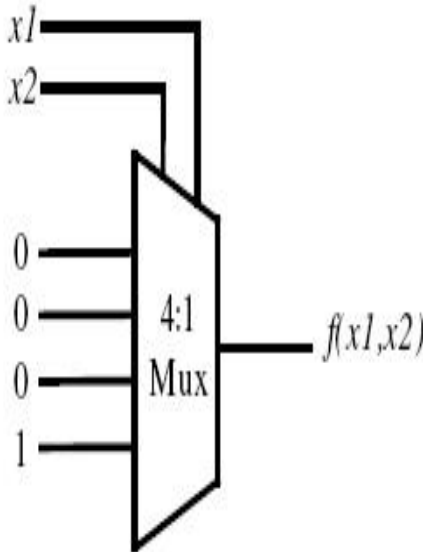


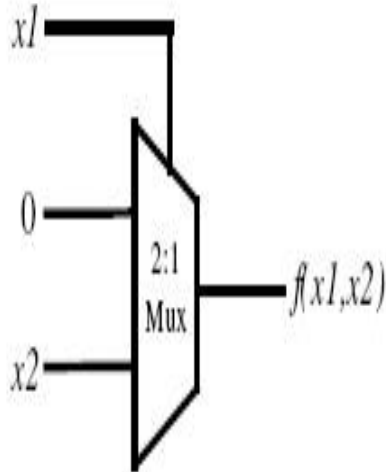
Multiplexers

- Multiplexers are generally described as 2^N -to-1 devices.
- These multiplexers have 2^N inputs, one of which is connected to the single output line. The N control lines determine which of the inputs is connected to the output.
- Here is a circuit for a 4-to-1 multiplexer. Note that the inputs are labeled X_3 , X_2 , X_1 , and X_0 here and I_3 , I_2 , I_1 , and I_0 in the multiplexer equation.



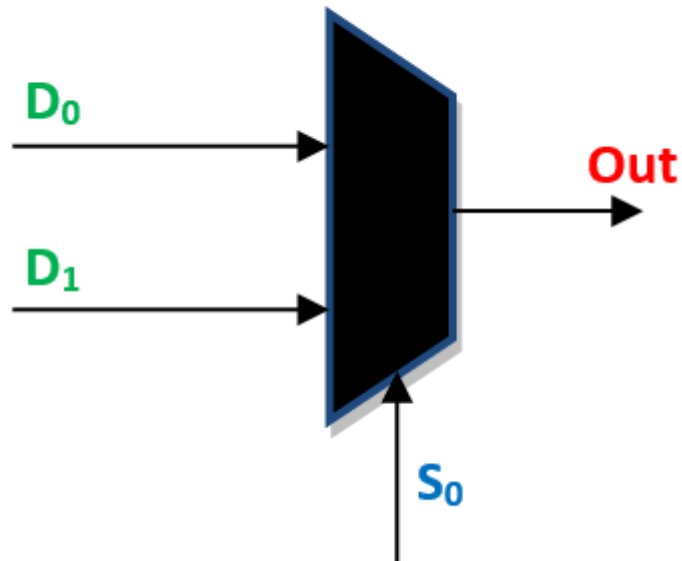
Basics of Multiplexers:

Truth Table	Logic Function Implementation															
<table><tr><th>x_1</th><th>x_2</th><th>$f(x_1,x_2)$</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x_1	x_2	$f(x_1,x_2)$	0	0	0	0	1	0	1	0	0	1	1	1	 <p>A 4:1 Multiplexer implementation of the function $f(x_1, x_2) = x_1x_2$. The two select inputs are x_1 and x_2. The four data inputs are 0, 0, 0, and 1. The output is $f(x_1, x_2)$.</p>
x_1	x_2	$f(x_1,x_2)$														
0	0	0														
0	1	0														
1	0	0														
1	1	1														

Truth Table	Logic Function Implementation						
<table> <tr> <th>x_1</th><th>$f(x_1,x_2)$</th></tr> <tr><td>0</td><td>0</td></tr> <tr><td>1</td><td>x_2</td></tr> </table>	x_1	$f(x_1,x_2)$	0	0	1	x_2	 <p>A 2:1 Multiplexer implementation of the function $f(x_1, x_2) = x_1x_2$. The select input is x_1. The two data inputs are 0 and x_2. The output is $f(x_1, x_2)$.</p>
x_1	$f(x_1,x_2)$						
0	0						
1	x_2						

2-Input Multiplexers:

2-Channel Multiplexer

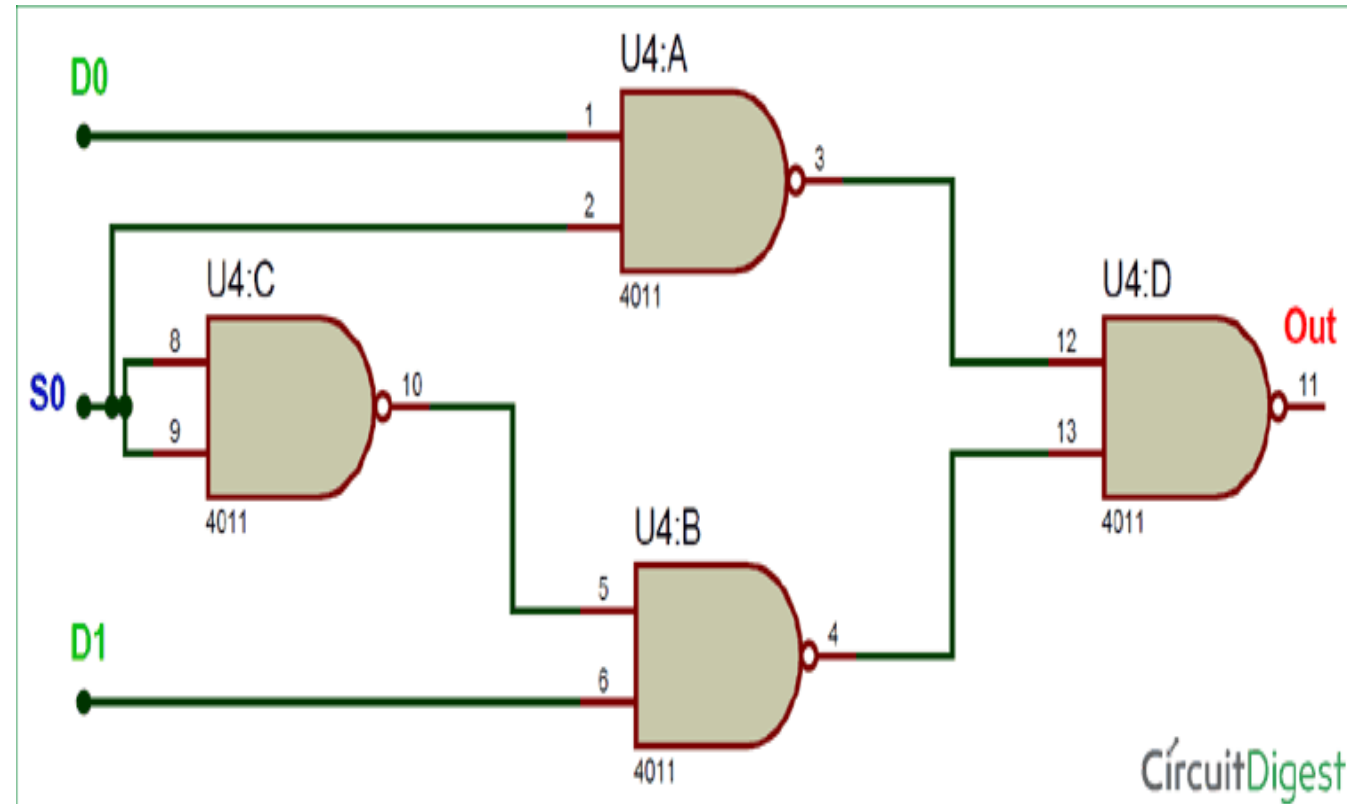


Here the input pins are named as D0 and D1 and the Output pin is named as out. The user can select one of the inputs that is either D0 or D1 by using the Control Pin S0. If S0 is kept low (logic 0) then the Input D0 will be reflected on the output pin and if the Input S0 is kept high (logic 1) then the Input D1 will be reflected on the output pin. The truth table representing the same is shown

S0	D0	D1	Out
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

2-Input Multiplexers:

- As you can see from the table above, when the control signal S_0 is 0 the Output reflects the signal values of D_0 (highlighted in blue) and similarly when the control signal S_0 is 1 the Output reflects the signal values of D_1 (highlighted in red). There are few dedicated IC packages which will work as multiplexers straight out of the package, but since we are trying to understand the combinational logic designs, let us build the above 2-input multiplexer by using logic gates. The Logic circuit diagram for the same is shown below



2-Input Multiplexers:

- The logic diagram utilises only the NAND gates and hence can be easily build on a perf board or even on a breadboard. The Boolean expression for the Logic diagram can be given by

$$\text{Out} = S_0' \cdot D_0' \cdot D_1 + S_0' \cdot D_0 \cdot D_1 + S_0 \cdot D_0 \cdot D_1' + S_0 \cdot D_0 \cdot D_1$$

- We can further simplify this Boolean expression by using the cancelling out the common terms, so that the logic diagram gets much more simple and easy to construct. The simplified Boolean expression is given below.

$$\text{Out} = S_0' \cdot D_0 + S_0 \cdot D_1$$

Higher Order Multiplexers (4:1 Multiplexer):

- Once you understand the working of a 2:1 Multiplexer, it should be easy to also understand the 4:1 Multiplexer. **It is just that it will have 4 input pins and 1 output pins with two control lines.** These two control lines can form 4 different combinational logic signals and for each signal one particular input will be selected.
- The number of control lines for any Multiplexer can be found using the below formulae

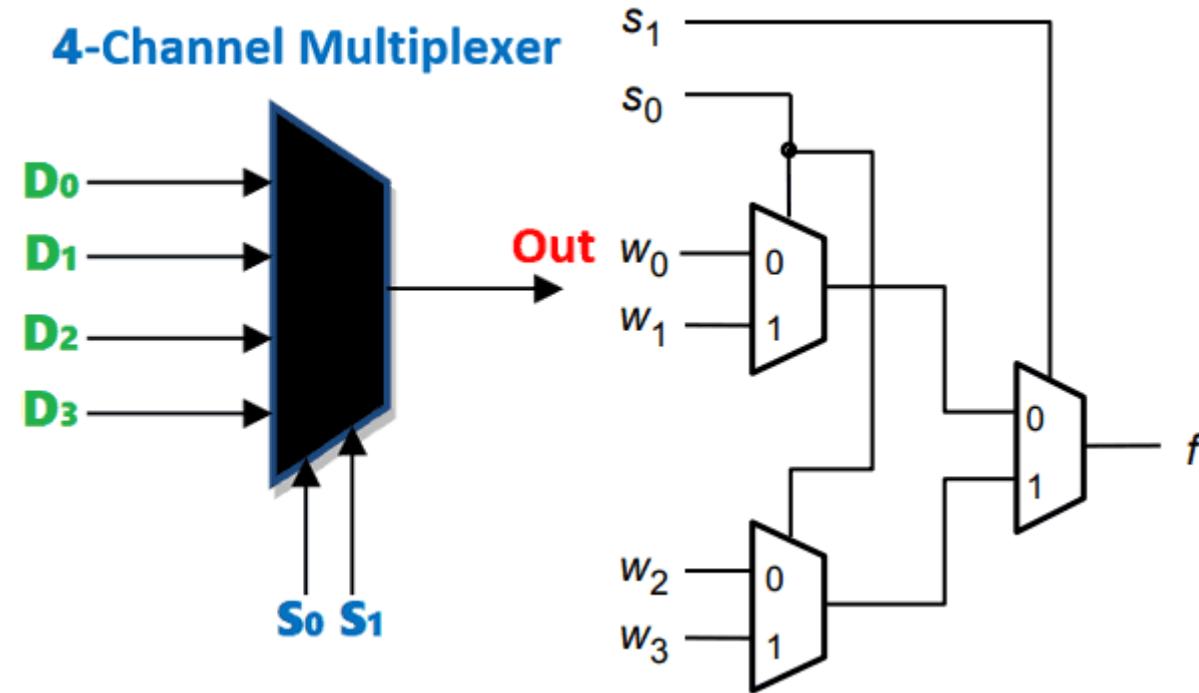
$$2^{\text{Number of Control lines}} = \text{Number of Input lines}$$

So, for instance a 2:1 Multiplexer will have 1 control line because $2^1 = 2$ and a 4:1 Multiplexer will have 2 control lines because $2^2 = 4$. Similarly you can calculate for any higher order Multiplexers.

It is also common to combine to lower order multiplexers like 2:1 and 4:1 MUX to form higher order MUX like 8:1 Multiplexer. Now, for example let us try to implement a 4:1 Multiplexer using a 2:1 Multiplexer. To construct a 4:1 MUX using a 2:1 MUX, we will have to combine three 2:1 MUX together.

Higher Order Multiplexers (4:1 Multiplexer):

- The end result should give us 4 Input pins, 2 Control/Select Pins and one output pin.
- To achieve the first two MUX is connected in parallel and then the output of those two are feeded as input to the 3rd MUX as shown below.



Higher Order Multiplexers (4:1 Multiplexer):

- The control/select line of the first two MUX is connected together to form a single line (S_0) and then the control line of the 3rd MUX is used as the second control/select signal.
- Thus finally we get a multiplexer with four inputs (W_0 , W_1 , W_2 and W_3) and only one output (f).
- The **truth table for a 4:1 Multiplexer** is shown below.

S_0	S_0	Out (f)
0	0	W_0
0	1	W_1
1	0	W_2
1	1	W_3

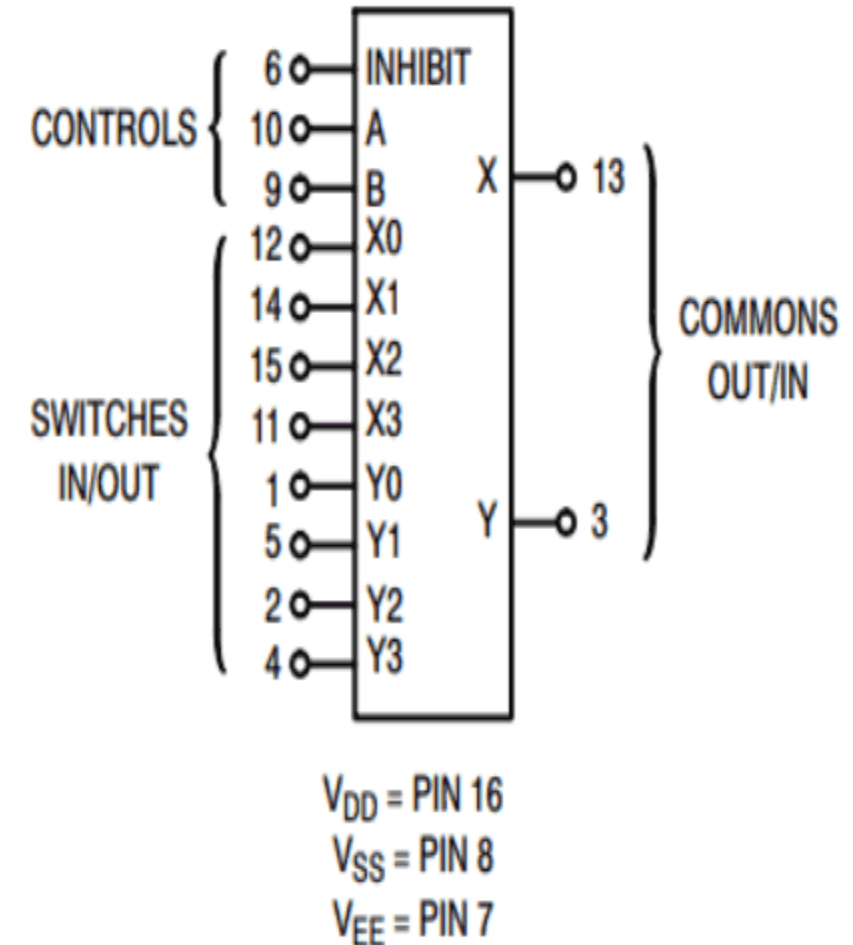
Higher Order Multiplexers (4:1 Multiplexer):

- As you can see in the table above, for each set of value provided to the Control signal pins (S0 and S1) we get a different Output from the input pins on our output pin.
- This way we can use the MUX to select one among the available four input pins to work with. Normally these Control pins (S0 and S1) will be controlled automatically using a digital circuit. There are certain dedicated IC which can act as MUX and make the job easy for us, so let us take a look at them.

Practical Implementation of Multiplexer using IC 4052:

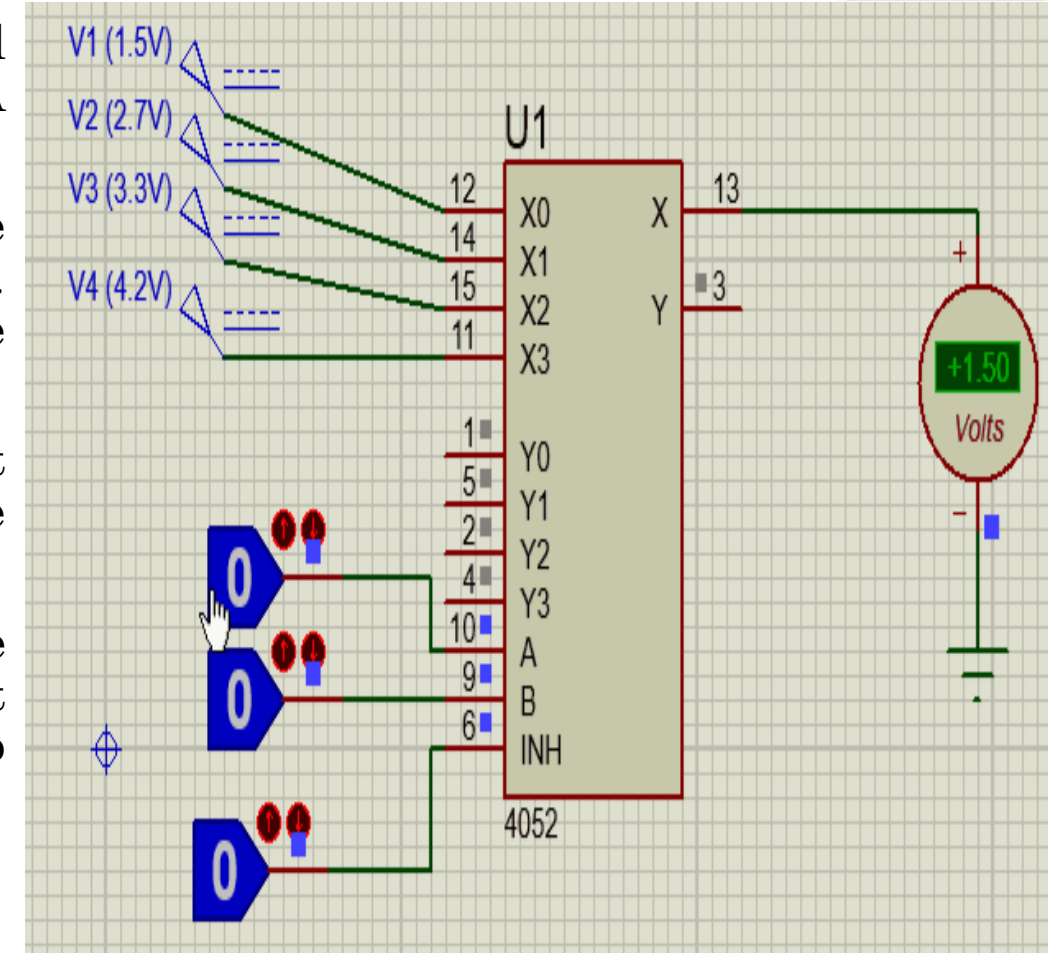


- It is always interesting to build and verify things practically, such that the theory we learn would make more sense. So let us build a 4:1 Multiplexer and check how it works.
- The IC that we are using here is **MC14052B** which has two 4:1 Multiplexers inside it. The pinouts of the IC is shown below



Practical Implementation of Multiplexer using IC 4052:

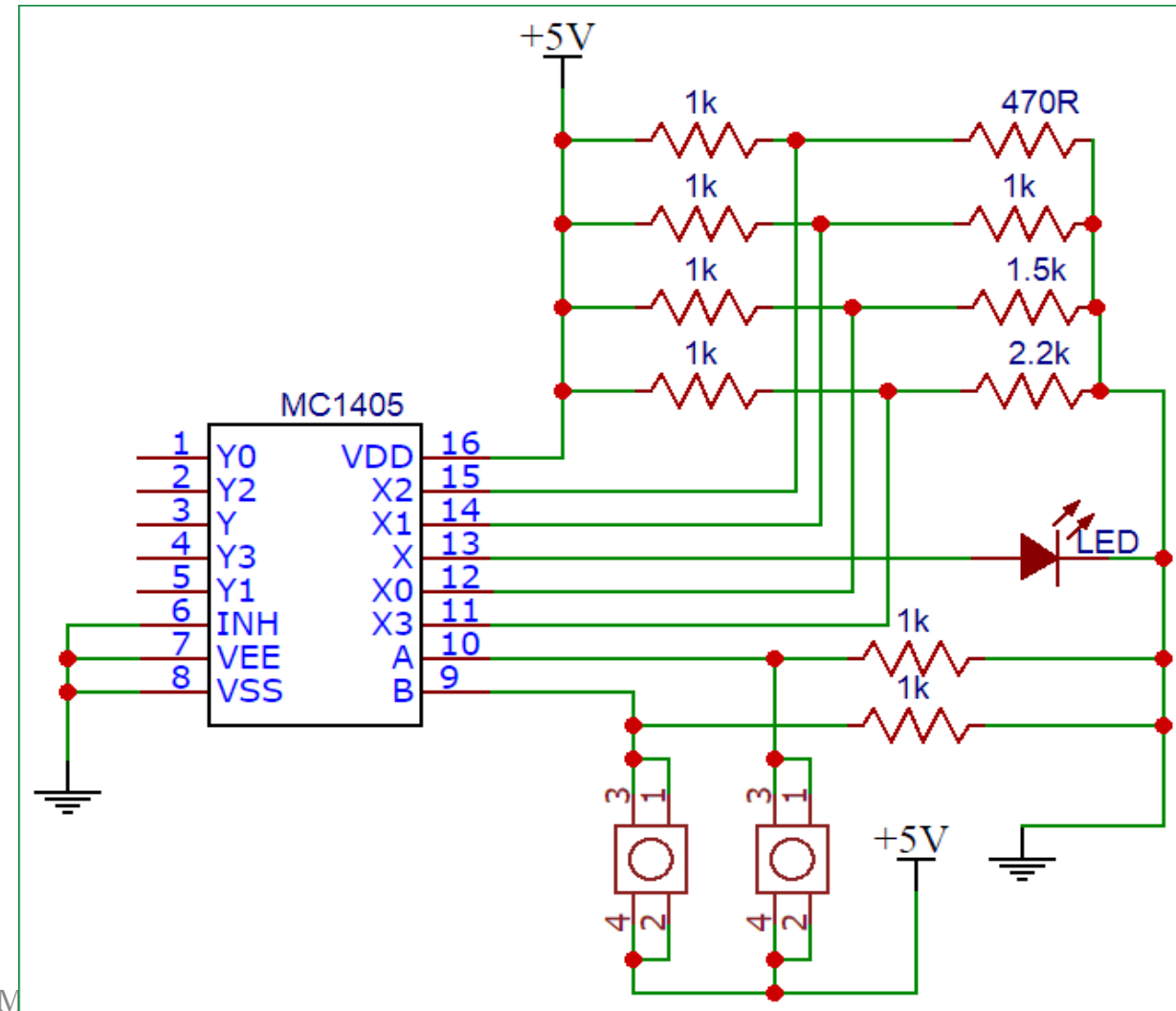
- Here the pins X0, X1, X2 and X3 are the four input pins and the pin X is its corresponding output pin. The control pins A and B are used to select the required input to the output pin.
- The Vdd pin (pin 16) has to connect to the supply voltage which is +5V and the Vss and Vee pin should be grounded. The Vee pin is for enable which is an active low pin so we have to ground it to enable this IC.
- The MC14052 is an Analog Multiplexer meaning the input pins can also be supplied with variable voltage and the same can be obtained through the output pins.
- The below GIF image shows how the IC outputs variable input voltage based in the control signals provided. The input pins has the voltage 1.5V, 2.7V, 3.3V and 4.8V which is also obtained on the Output pin based on the control signal given.



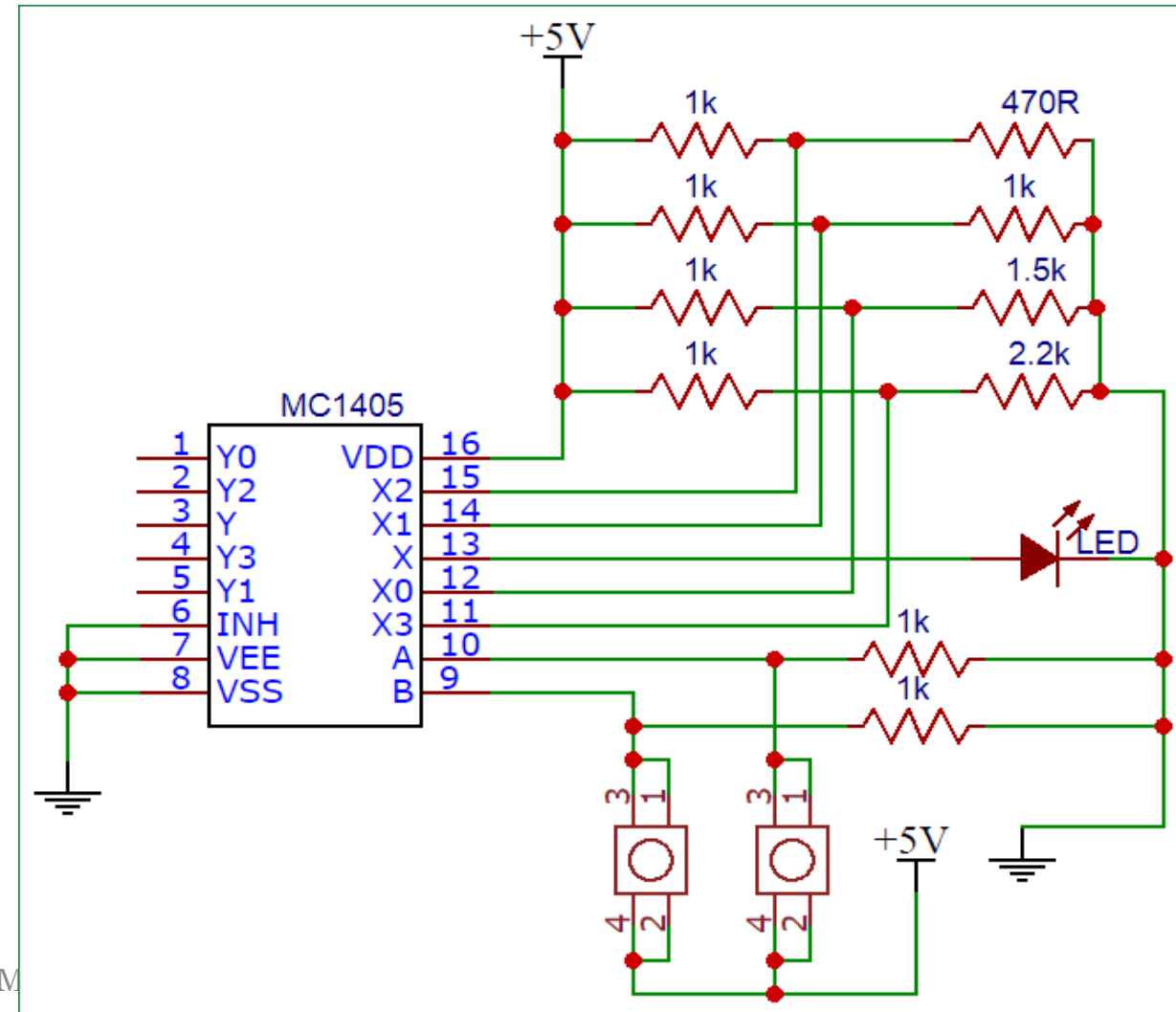
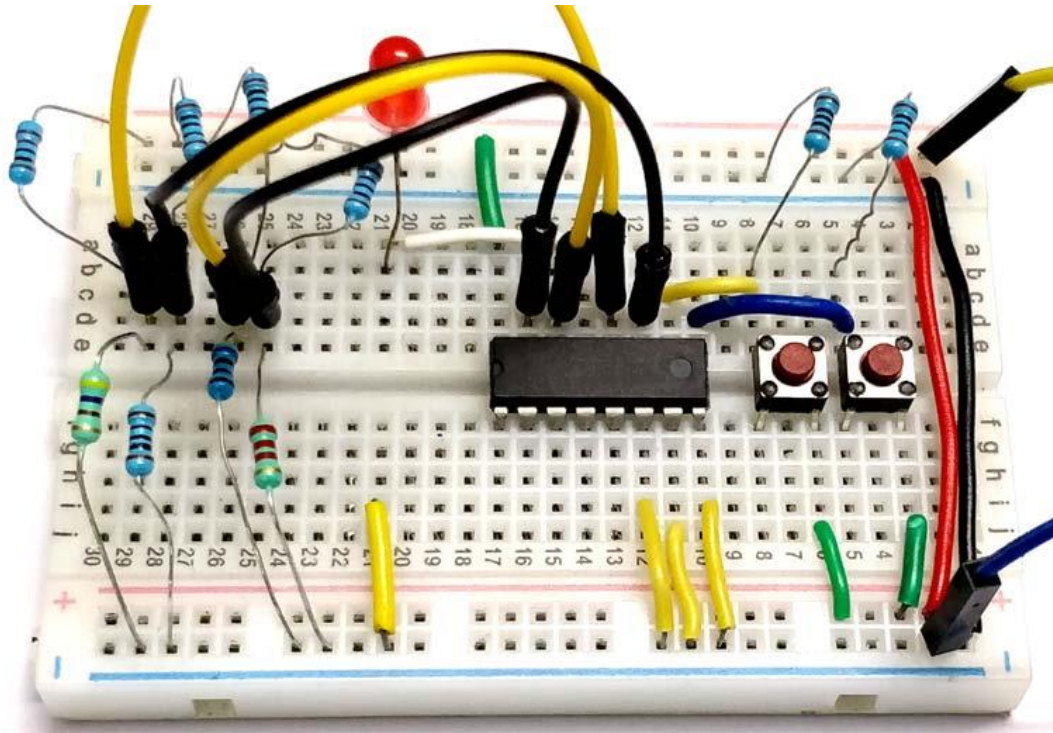
Practical Implementation of Multiplexer using IC 4052:



- We can also assemble this circuit over a breadboard and check if they are working. To do that I have used two push buttons are inputs for the control pins A and B. And used a series of potential divider combinations to provide variable voltages for the pins 12, 14, 15 and 11.
- The output pin 13 is connected to an LED. The variable voltages supplied to the LED will make it to vary the brightness based on the control signals.
- The circuit once build will look something like this below



Practical Implementation of Multiplexer using IC 4052:

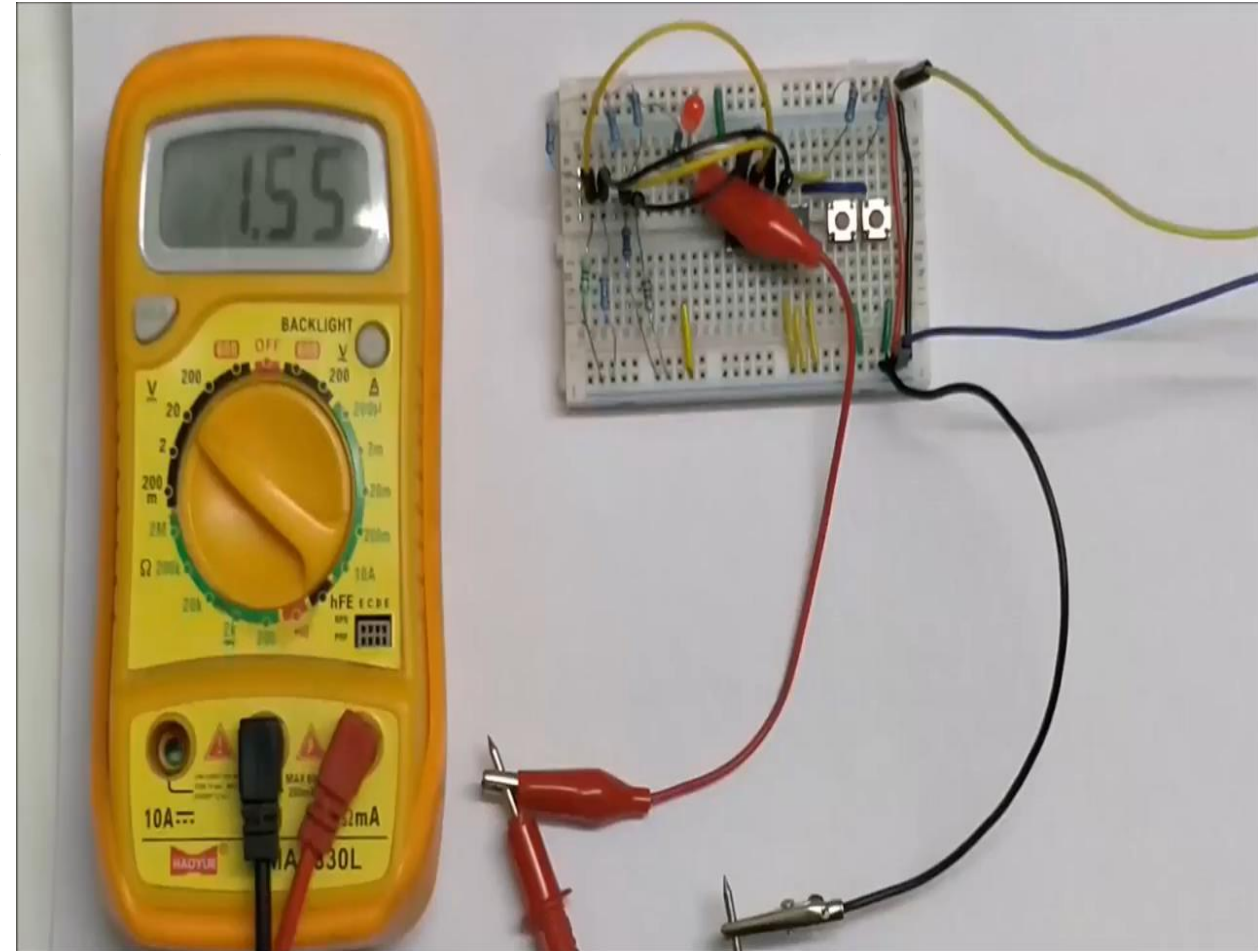


DR AJM

Practical Implementation of Multiplexer using IC 4052:

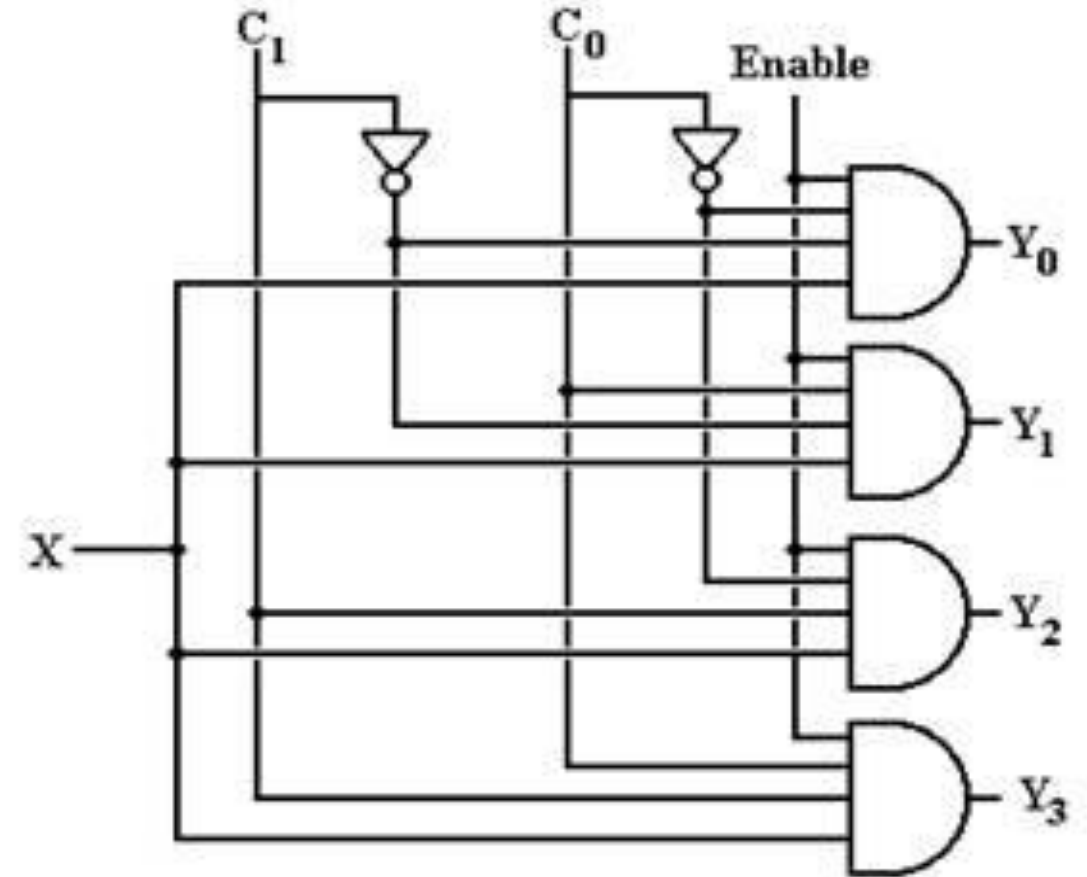


- The **complete working video of the circuit** can also be found at the bottom of this page. Hope you understood the working of Multiplexers and know where to use them in your projects. If you have any thoughts or doubts leave them in the comment section below and I will try my best to respond to them. You can also use the forums to resolve your technical doubts and share your knowledge among other members of this community.



Demultiplexer

- Demultiplexers are generally described as 1-to- 2^N devices. These multiplexers have one input, which is connected to one of the 2^N output lines.
- The N control lines determine which of the output line is connected to the input.
- Here is a circuit for a 1-to-4 demultiplexer, which might be called “active high” in that the outputs not selected are all set to 0.
- Note that multiplexers do not have the problem of unselected outputs; a MUX has only one output.



Reference

1. Mano book
2. https://www.tutorialspoint.com/computer_logical_organization/combinational_circuits.htm

