Multiplexers and Demultiplexers: Description and Design Issues

Edward L. Bosworth, Ph.D.
TSYS School of Computer Science
Columbus State University
Columbus, GA 31907

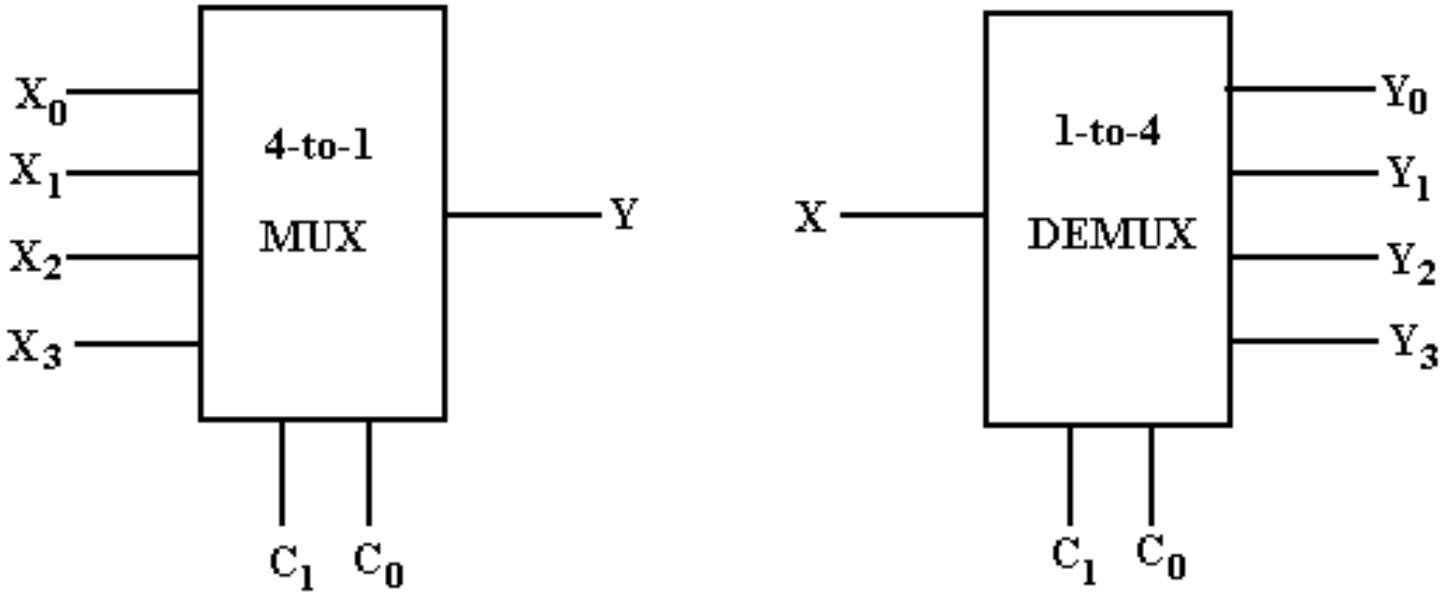
Multiplexers and Demultiplexers

Multiplexer – MUX
Associates One of Many Inputs to a Single Output

Demultiplexer – DEMUX
Associates One Input with One of Many Outputs

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

Sample: 4–to–1 MUX and 1–to–4 DEMUX



My Notation: X for Input
C for Control Signals
Y for Output

The Multiplexer Equation Illustrated for a 4–to–1 MUX

Truth table Denote the multiplexer output by **M**

C_1	C_0	M
0	0	X_0
0	1	X_1
1	0	X_2
1	1	X_3

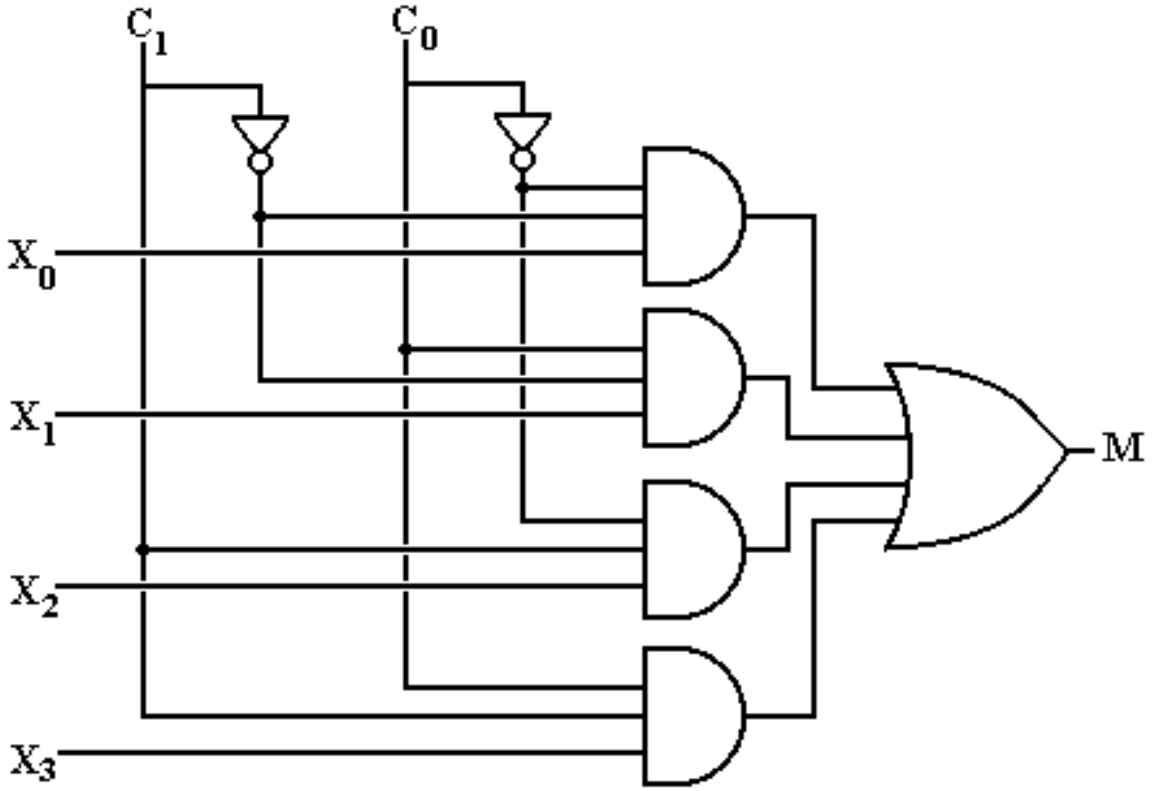
Equation Form

$$\mathbf{M = \overline{C}_1 \cdot \overline{C}_0 \cdot X_0 + \overline{C}_1 \cdot C_0 \cdot X_1 + C_1 \cdot \overline{C}_0 \cdot X_2 + C_1 \cdot C_0 \cdot X_3}$$

Here is another form of the equation that is better when X is used as an input.

$$\mathbf{M = \overline{C_0} \cdot \overline{C_1} \cdot I_0 + \overline{C_0} \cdot C_1 \cdot I_1 + C_0 \cdot \overline{C_1} \cdot I_2 + C_0 \cdot C_1 \cdot I_3}$$

Build a 4–to–1 MUX

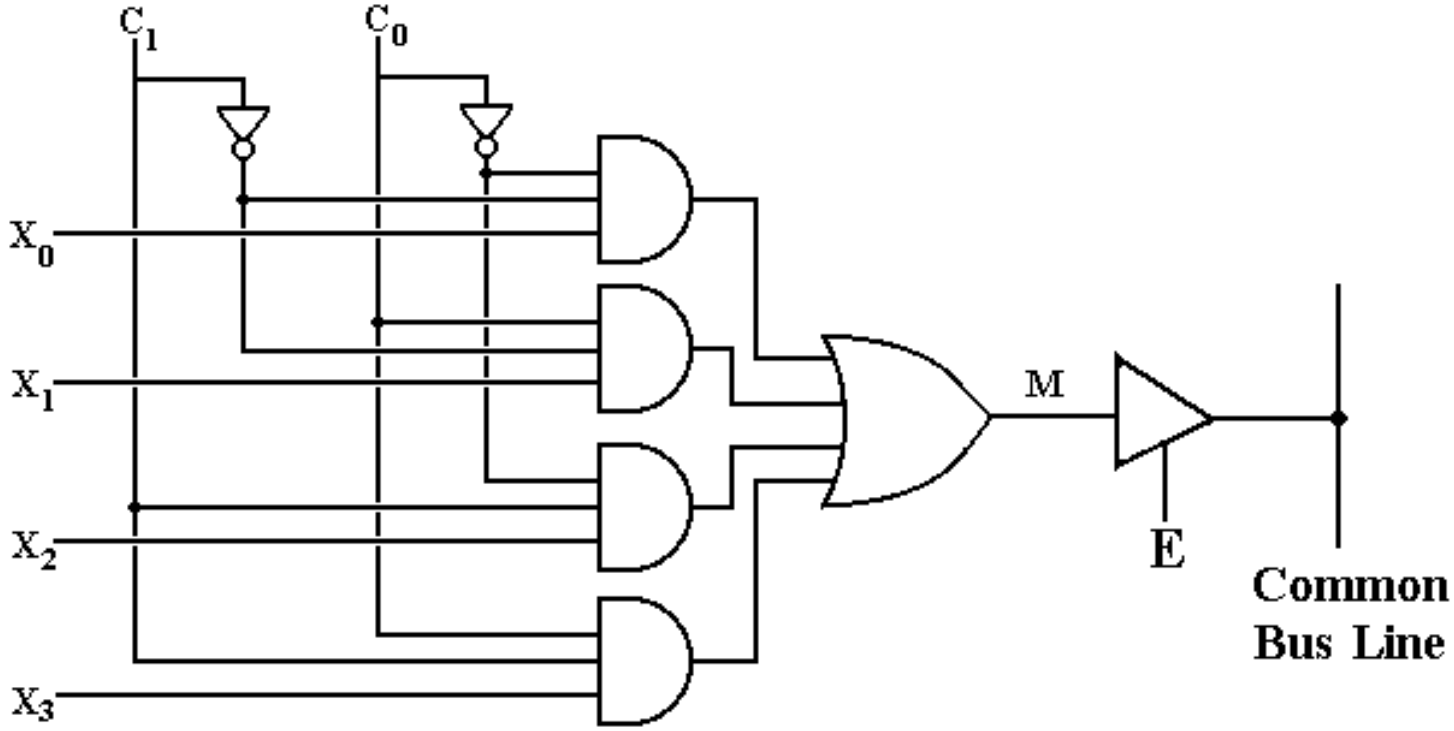


But what about an enable input for a multiplexer?

What does it mean for the output of the MUX to be 0?

Multiplexer Attached to a Bus Line

To control a multiplexer's connection to a common bus, we use a tri–state buffer and not an enable input to the MUX. Here I use "E" as the tri–state control.

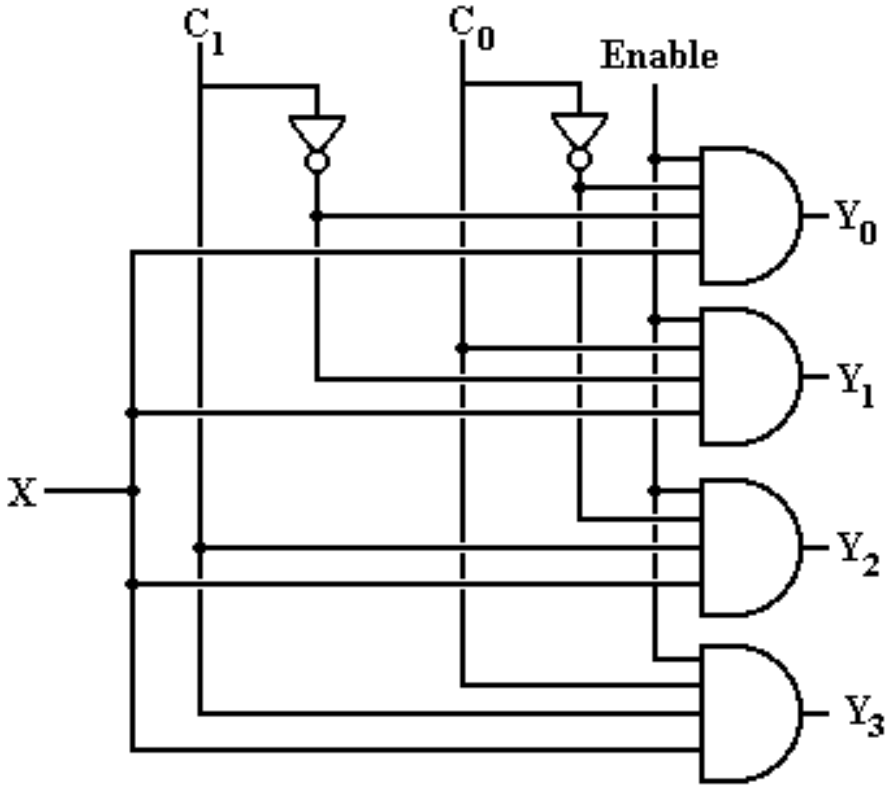


When E = 1, the selected MUX input is placed on the bus.
When E = 0, the MUX is detached from the bus; another source feeds the bus.

A 1–to–4 DEMUX

C_1	C_0	Selected Output
0	0	$Y_0 = X$ Other outputs 0
0	1	$Y_1 = X$ Other outputs 0
1	0	$Y_2 = X$ Other outputs 0
1	1	$Y_3 = X$ Other outputs 0

Build a 1–to–4 DEMUX With an Enable



If Enable = 0, all outputs are 0.

Using A 2^N–to–1 MUX for a Boolean Function of N Boolean Variables

Theorem 1: Any Boolean function of N Boolean variables, N > 0, can be constructed by a multiplexer with 2^N inputs (usually labeled I_K, I_{K-1}, … I_1, I_0) and N control lines, labeled C_{N-1} … C_0.

Method: Express the Boolean function of N Boolean variables in Canonical Sum of Products and then match the desired function to the Multiplexer Equation for a 2^N–to–1 MUX.

Example: F2(X, Y, Z) = X•Y + X•Z + Y•Z

Step 1: This is a function of three Boolean variables. We must use a 2^3–to–1 MUX, also called a 8–to–1 MUX.

Using A 2^N–to–1 MUX (page 2)

Step 2: Convert F2(X, Y, Z) = X•Y + X•Z + Y•Z to Canonical SOP.
Every product term must have a literal for each variable.
A literal is either the variable or its complement.

$$
\begin{aligned}
F2 &= X \cdot Y + X \cdot Z + Y \cdot Z \\
&= X \cdot Y \cdot (\overline{Z} + Z) + X \cdot (\overline{Y} + Y) \cdot Z + (\overline{X} + X) \cdot Y \cdot Z \\
&= X \cdot Y \cdot \overline{Z} + X \cdot Y \cdot Z + X \cdot \overline{Y} \cdot Z + X \cdot Y \cdot Z + \overline{X} \cdot Y \cdot Z + X \cdot Y \cdot Z \\
&= \overline{X} \cdot Y \cdot Z + X \cdot \overline{Y} \cdot Z + X \cdot Y \cdot \overline{Z} + X \cdot Y \cdot Z
\end{aligned}
$$

Note that all four terms have a literal for each of the three variables X, Y, and Z.

Using A 2^N–to–1 MUX (page 3)

Step 3: Convert the function to a form with all 2^N product terms.
Here we convert F2 to have all eight possible product terms.

$$F(X, Y, Z) = \overline{X}\bullet Y\bullet Z + X\bullet\overline{Y}\bullet Z + X\bullet Y\bullet\overline{Z} + X\bullet Y\bullet Z$$

$$= \overline{X}\bullet\overline{Y}\bullet\overline{Z}\bullet 0 + \overline{X}\bullet\overline{Y}\bullet Z\bullet 0 + \overline{X}\bullet Y\bullet\overline{Z}\bullet 0 + \overline{X}\bullet Y\bullet Z\bullet 1$$

$$+ X\bullet\overline{Y}\bullet\overline{Z}\bullet 0 + X\bullet\overline{Y}\bullet Z\bullet 1 + X\bullet Y\bullet\overline{Z}\bullet 1 + X\bullet Y\bullet Z\bullet 1$$

Using A 2^N–to–1 MUX
(page 4)

Step 4: Write the Multiplexer Equation for an 8–to–1 MUX.

$$M = \overline{C}_2 \cdot \overline{C}_1 \cdot \overline{C}_0 \cdot I_0 + \overline{C}_2 \cdot \overline{C}_1 \cdot C_0 \cdot I_1 + \overline{C}_2 \cdot C_1 \cdot \overline{C}_0 \cdot I_2 + \overline{C}_2 \cdot C_1 \cdot C_0 \cdot I_3$$
$$+ C_2 \cdot \overline{C}_1 \cdot \overline{C}_0 \cdot I_4 + C_2 \cdot \overline{C}_1 \cdot C_0 \cdot I_5 + C_2 \cdot C_1 \cdot \overline{C}_0 \cdot I_6 + C_2 \cdot C_1 \cdot C_0 \cdot I_7$$

Step 5: Rewrite the equation with $C_2 = X$, $C_1 = Y$, and $C_0 = Z$.

$$M = \overline{X} \cdot \overline{Y} \cdot \overline{Z} \cdot I_0 + \overline{X} \cdot \overline{Y} \cdot Z \cdot I_1 + \overline{X} \cdot Y \cdot \overline{Z} \cdot I_2 + \overline{X} \cdot Y \cdot Z \cdot I_3$$
$$+ X \cdot \overline{Y} \cdot \overline{Z} \cdot I_4 + X \cdot \overline{Y} \cdot Z \cdot I_5 + X \cdot Y \cdot \overline{Z} \cdot I_6 + X \cdot Y \cdot Z \cdot I_7$$

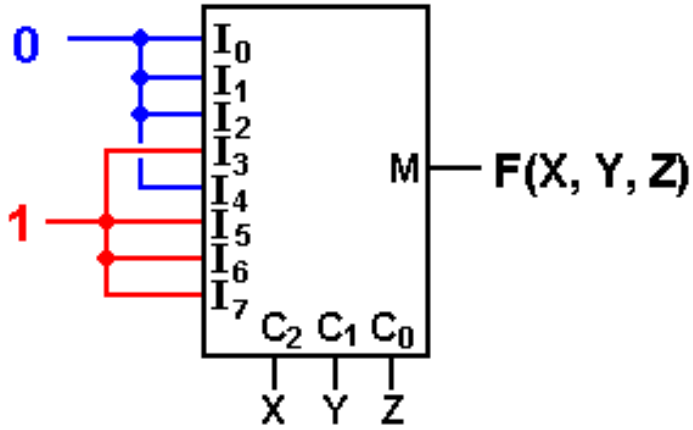
NOTE: Here I use I_0, I_1, …, I_7 as the MUX inputs because I am using X to denote one of the Boolean variables.

Using A 2^N–to–1 MUX (page 5)

Step 6: Match the two expressions

$$F(X, Y, Z) = \overline{X}\cdot\overline{Y}\cdot\overline{Z}\cdot 0 + \overline{X}\cdot\overline{Y}\cdot Z\cdot 0 + \overline{X}\cdot Y\cdot\overline{Z}\cdot 0 + \overline{X}\cdot Y\cdot Z\cdot 1$$
$$+ X\cdot\overline{Y}\cdot\overline{Z}\cdot 0 + X\cdot\overline{Y}\cdot Z\cdot 1 + X\cdot Y\cdot\overline{Z}\cdot 1 + X\cdot Y\cdot Z\cdot 1$$

$$M = \overline{X}\cdot\overline{Y}\cdot\overline{Z}\cdot I_0 + \overline{X}\cdot\overline{Y}\cdot Z\cdot I_1 + \overline{X}\cdot Y\cdot\overline{Z}\cdot I_2 + \overline{X}\cdot Y\cdot Z\cdot I_3$$
$$+ X\cdot\overline{Y}\cdot\overline{Z}\cdot I_4 + X\cdot\overline{Y}\cdot Z\cdot I_5 + X\cdot Y\cdot\overline{Z}\cdot I_6 + X\cdot Y\cdot Z\cdot I_7$$



$I_0 = 0$ $I_1 = 0$ $I_2 = 0$ $I_3 = 1$
$I_4 = 0$ $I_5 = 1$ $I_6 = 1$ $I_7 = 1$

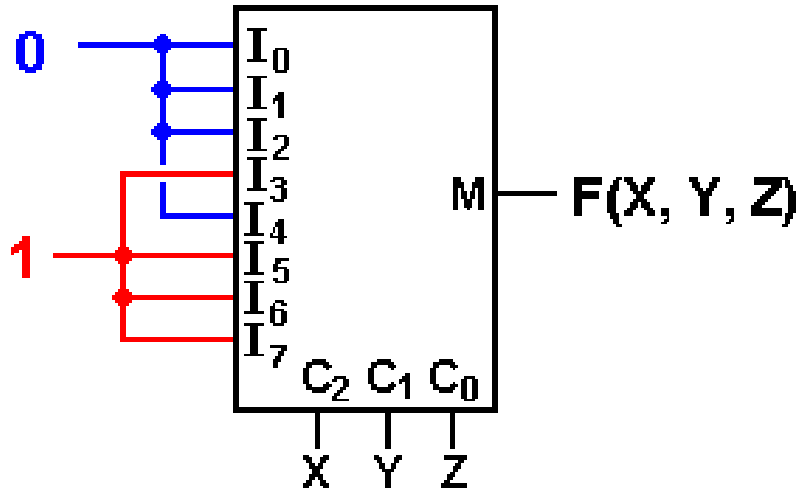
with $C_2 = X$, $C_1 = Y$, and $C_0 = Z$.

Using A 2^N–to–1 MUX
(Using either a Σ list or a Π list)

For a Σ list, connect the listed inputs to 1 and the others to 0.

For a Π list, connect the listed inputs to 0 and the others to 1.

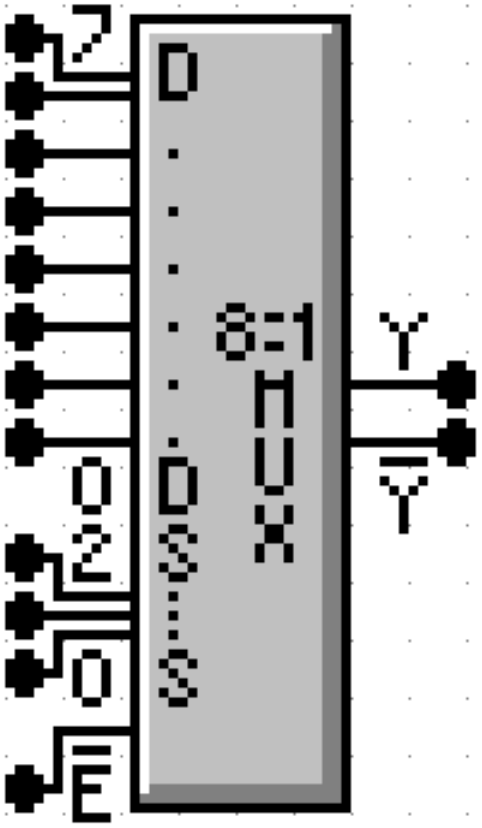
F(X, Y, Z) = Σ(3, 5, 6, 7) = Π(0, 1, 2, 4)



We try this with a common circuit emulator, such as Multi–Media Logic, and find that we need to think a bit more.

An Eight–to–One MUX in Multi–Media

Here is the circuit element selected in the Multi–Media Logic tool.



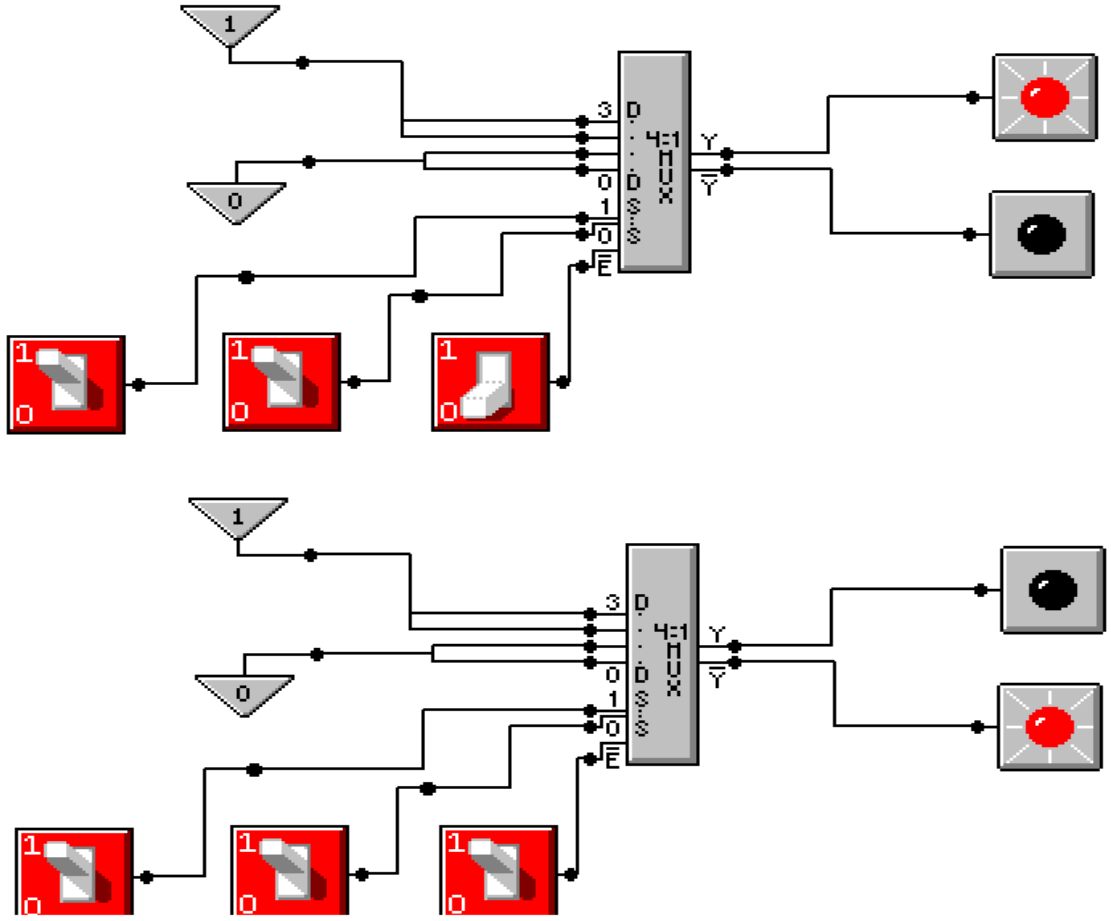
This is an 8–to–1 MUX with inputs labeled 7 through 0, or equivalently X_7 through X_0. This is expected.

The selector (control) lines are as expected; 2 through 0.

In my notes, I use M for the output of the Multiplexer. This figure uses the symbol Y (not a problem) and notes that real multiplexers also output the complement.

The only issue here is the enable. Note that the MUX is enabled low; this signal must be set to ground in order for the multiplexer to function as advertised.

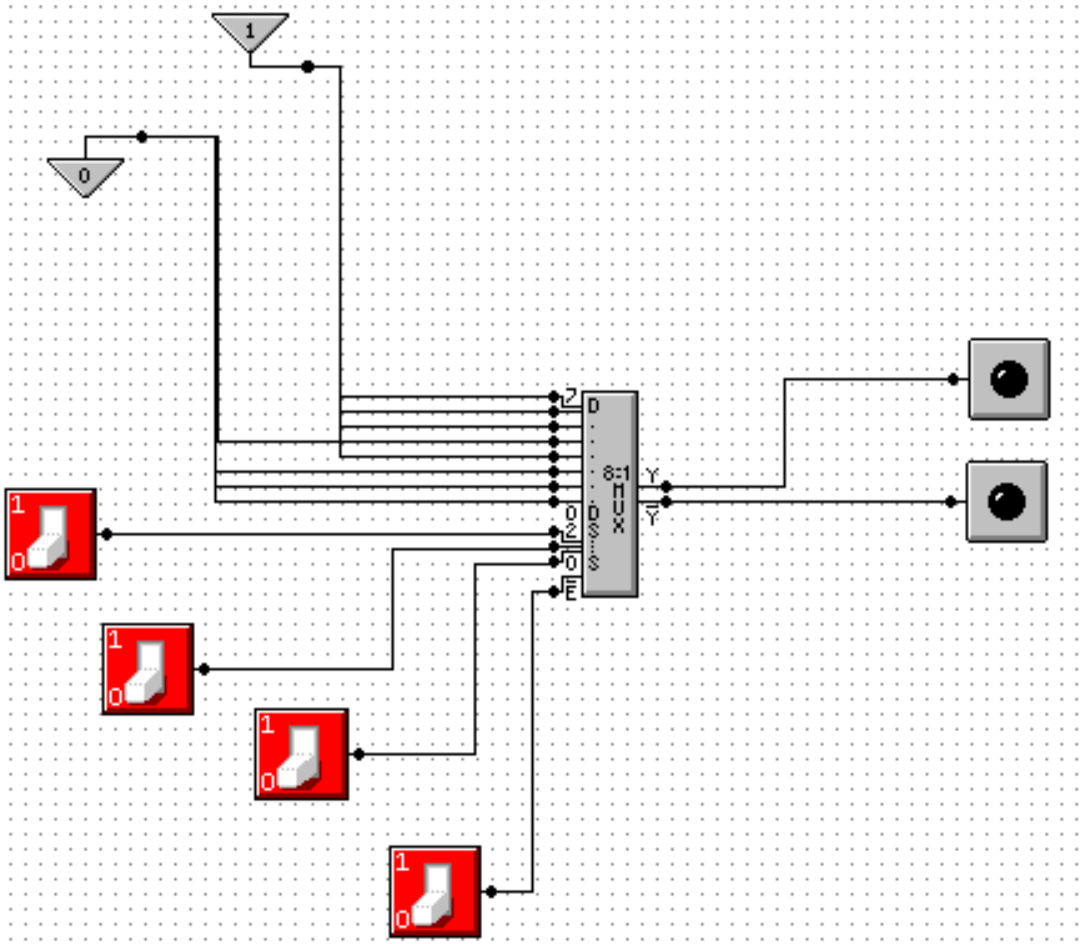
Commercial Multiplexer: Enabled and Not Enabled



At top, the output is X_3. At bottom, the output is 0.

Carry–Out of a Full Adder

Here is a screen shot of my implementation of F(X, Y, Z) = Σ(3, 5, 6, 7).



NOTE: Show simulation here.

Gray Codes: Minimal Effort Testing

Consider the above circuit with three basic inputs S_2, S_1, S_0.
How can one test all possible inputs with minimum switching?

One good answer is to use Gray Codes for input. Here are the 2–bit and 3–bit codes.

00	000
01	001
11	011
10	010
	110
	111
	101
	100

To generate an (N + 1)–bit code set from an N–bit code set.

1. Write out the N–bit codes with 0 as a prefix, then
2. Write out the N–bit codes in reverse with 1 as a prefix.

00, 01, 11, 10 becomes 000, 001, 011, 010, 110, 111, 101, and 100

Testing the Carry–Out Circuit

If the Enable switch is set to 1, the output is always 0. Y' = 1.

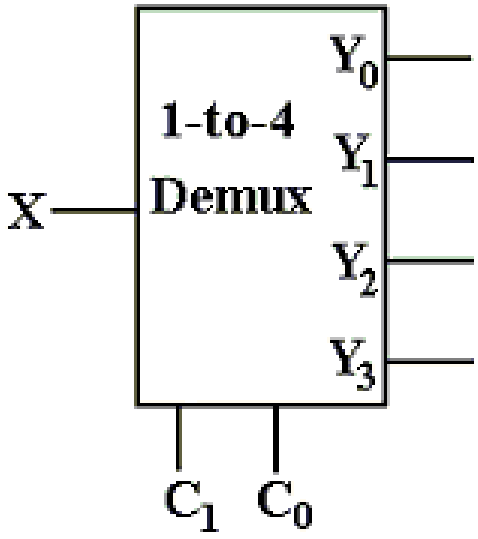
Set the Enable switch to 0 and generate the following sequence.

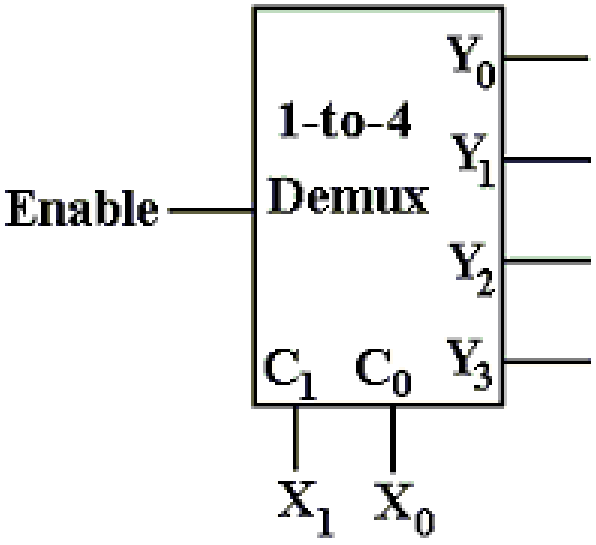
Start with $S_2 = 0$, $S_1 = 0$, $S_0 = 0$.	0 0 0
Click S_0 to get	0 0 1
Click S_1 to get	0 1 1
Click S_0 to get	0 1 0
Click S_2 to get	1 1 0
Click S_0 to get	1 1 1
Click S_1 to get	1 0 1
Click S_0 to get	1 0 0

Where are the Decoders?

One will note that the Multi–Media Logic tool does not provide a decoder circuit.

Fortunately, a 1–to–2^N demultiplexer can be made into an N–to–2^N decoder.





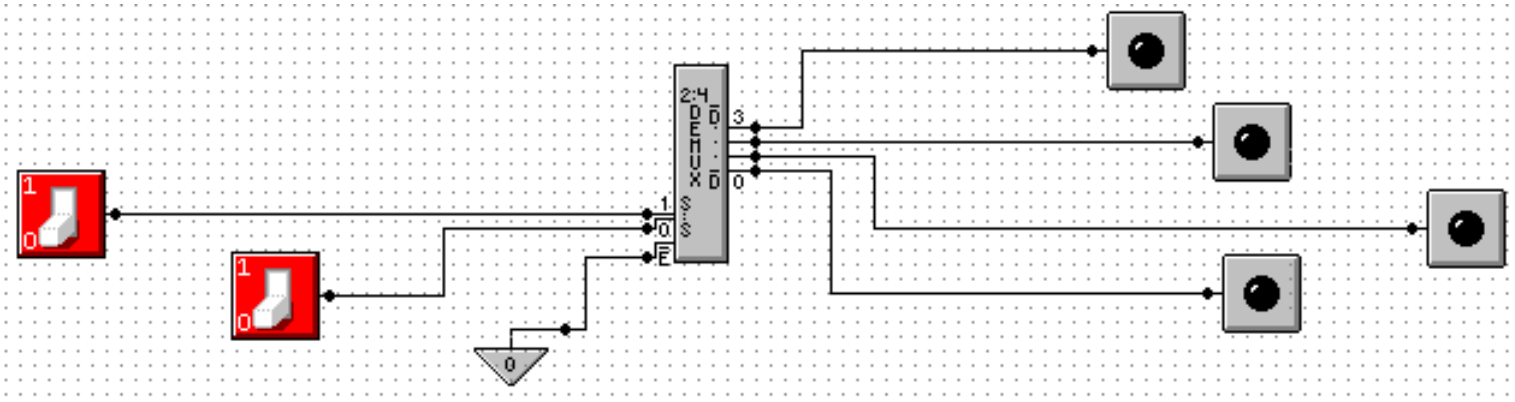
Look at the circuit to the left. The control signals C_1,C_0 select the output to receive the input X. This is exactly equivalent to a decoder.

In the circuit at right, the selected output gets the input, now called “Enable”.
For the demultiplexers we use, the other outputs get a logic 1.

We can fabricate an active low decoder.

The MUX as an Active–Low Decoder

Here is the 2–to–4 Demultiplexer as an 2–to–4 active low decoder.



Here is an answer to one of the homework problems: use a 2–to–4 decoder for XOR. The function is either $\Sigma(1, 2)$ or $\Pi(0, 3)$.