Lecture 11, Feb. 12, 07

Magnitude Comparator

Another example of a basic circuit we consider at this point is a Comparator.

Assume we would like to Compare numbers:

$$A = A_3 A_2 A_1 A_0$$

and

$$B = B_3 B_2 B_1 B_0 .$$

That is, we want to know whether:

$$A = B \qquad A > B \quad \text{or} \quad A < B ,$$

It is clear that $A = B$ if and only if

$$A_3 = B_3 \; , \; A_2 = B_2 \; , \; A_1 = B_1 \; , \; A_0 = B_0 .$$

For A_3 being equal to B_3 we need to have the function

$$x_3 = A_3 B_3 + A_3' B_3'$$

be true. Similarly, for other bits. Let's define

$$x_i = A_i B_i + A_i' B_i' \qquad i = 0, 1, 2, 3$$

Then $(A=B) = x_3 x_2 x_1 x_0$.

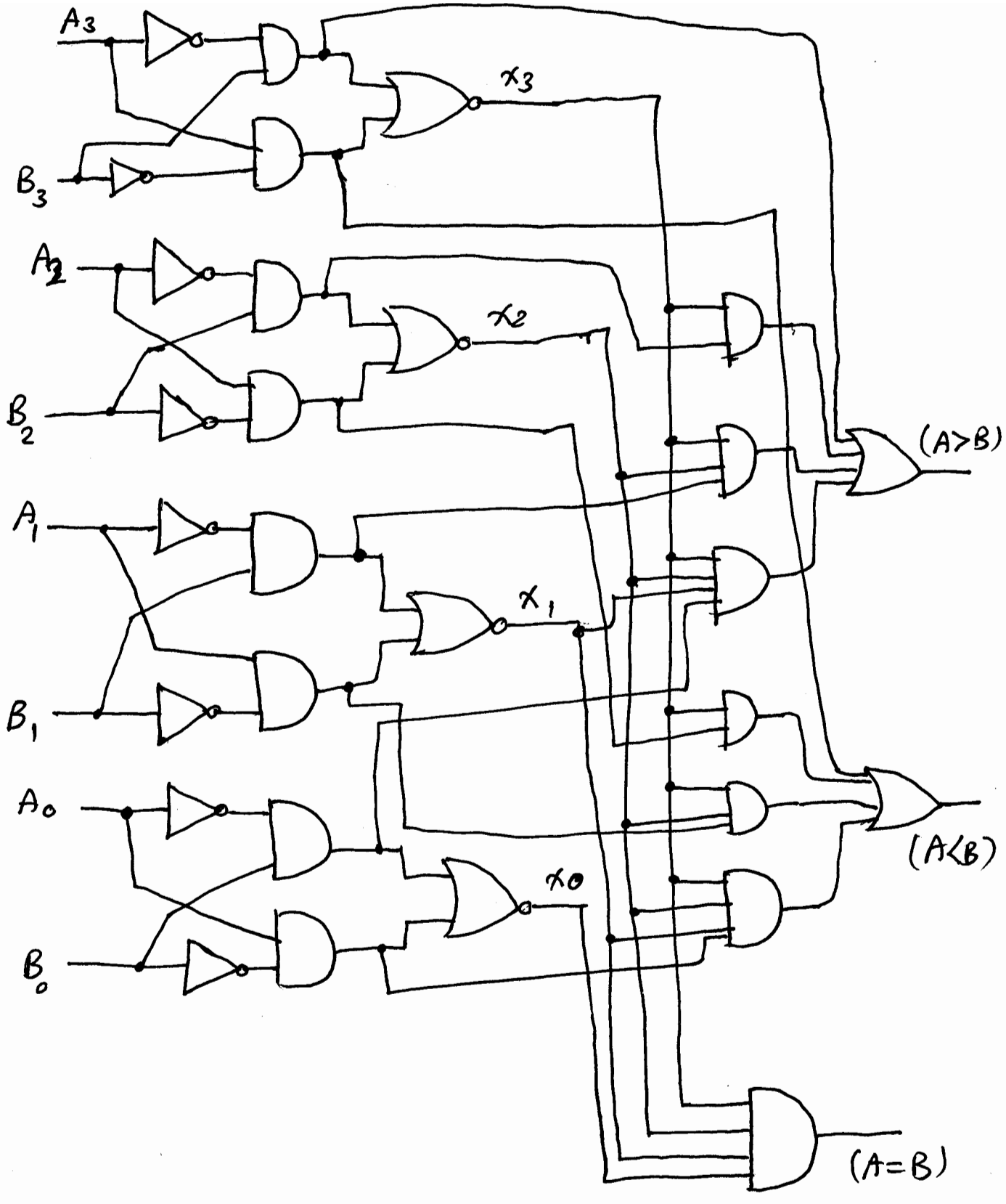
So, we need to implement x_0, x_1, x_2, x_3 and then AND them to get an output showing that $A=B$. What about $A>B$. If A_3 is equal to 1 and $B_3 = 0$ then $A>B$. This means that if $A_3 B_3' = 1$ then $A>B$. But if $A_3 = B_3$, i.e., if $x_3 = 1$ and $A_2 = 1$ and $B_2 = 0$, i.e., if $x_3 A_2 B_2' = 1$ then again $A>B$. The same is true if $x_3 x_2 A_1 B_1' = 1$ or $x_3 x_2 x_1 A_0 B_0' = 1$. So,

$$(A>B) = A_3 B_3' + x_3 A_2 B_2' + x_3 x_2 A_1 B_1' + x_3 x_2 x_1 A_0 B_0'$$

Similarly

$$(A<B) = A_3' B_3 + x_3 A_2' B_2 + x_3 x_2 A_1' B_1 + x_3 x_2 x_1 A_0' B_0$$

The comparator circuit's implementation is shown on the next page.


A3
B3
x3
A2
B2
x2
A1
B1
x1
A0
B0
x0
(A>B)
(A<B)
(A=B)

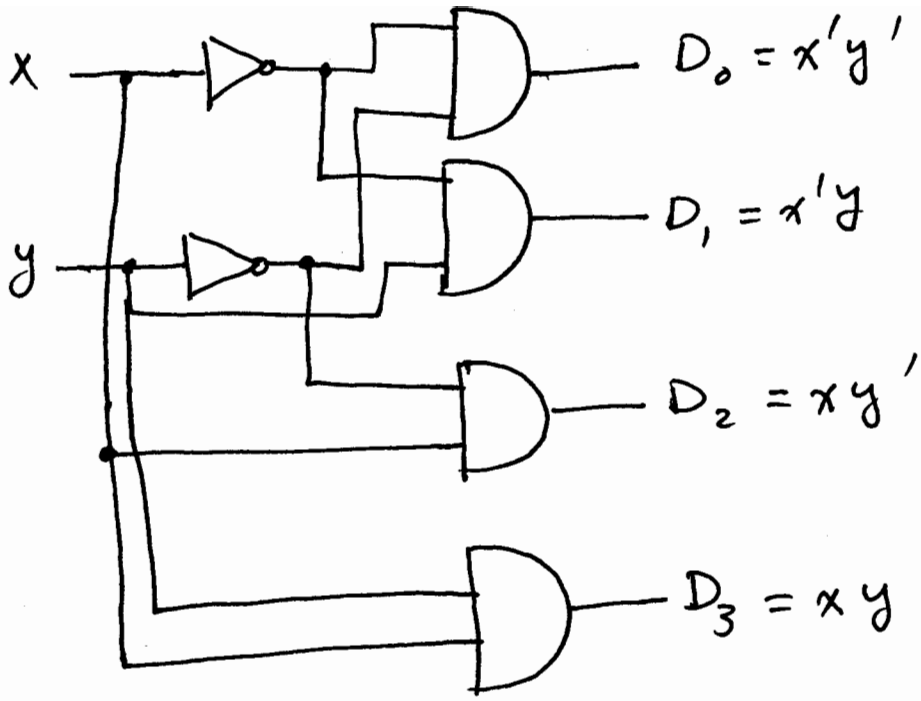
Decoders, Encoders, Multiplexers:

A decoder takes in n inputs and outputs $m \leq 2^n$ bits.

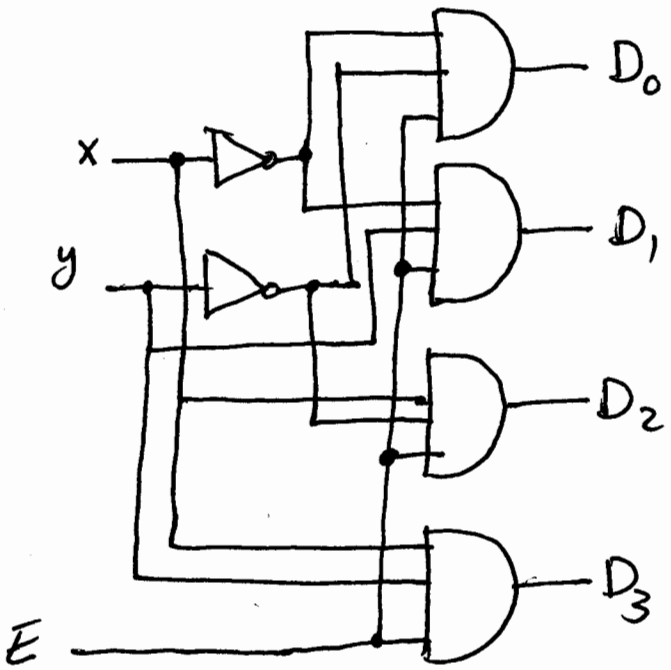
An encoder does the inverse function of a decoder. It has 2^n inputs and n outputs.

A multiplexer has n inputs and one output. A multiplexer is like a switch that selects one of its inputs. The way that it selects the inputs is dictated by select (or control) inputs.

Example: A two-input, 4-output decoder. Denote the inputs by letters x and y. The outputs can be one of the four values $x'y'$, $x'y$, xy', and xy. The following block diagram shows the 2-input, 4-output decoder's implementation.



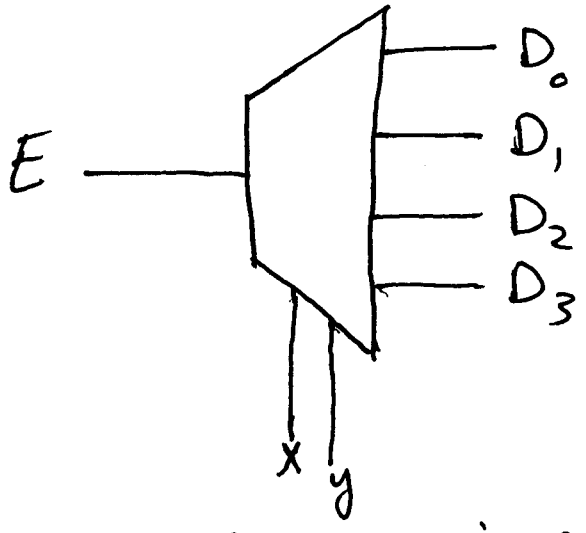
We can add an <u>enable</u> bit ~~to~~ the decoder. This input goes to all ANDs and lets the decoder to output the minterms only when it is equal to one. When enable bit is zero the output is all 0.



The truth table for the above decoder is

E	x	y	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Look carefully at the truth table. You observe that the inputs x and y cause the enable input to appear locations D_0, D_1, D_2, D_3 depending on the value of themselves. This is what is termed a <u>demultiplexer</u>



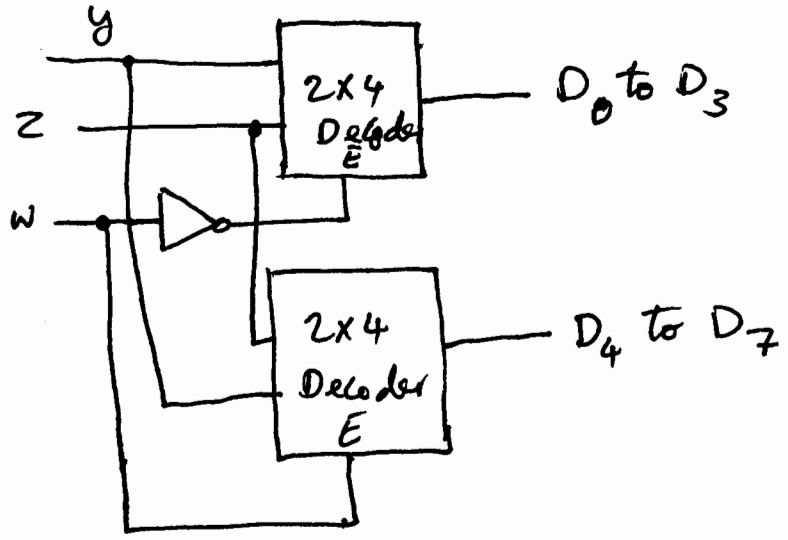
The <u>demultiplexer</u> is like a switch controlled by x and y.

Expanding the decoders

A decoder with n inputs can be implemented using two decoders each with n−1 inputs.

Example:

Use two 2-input decoders to implement a 3-input decoder.



When $w=0$ the first (top one) decoder is enabled and D_0 to D_3 represent $w'x'y'$, $w'x'y$ $w'xy'$ and $w'xy$.

When $w=1$ the lower decoder is enabled and D_4 to D_7 are high for $wx'y'$, $wx'y$, wxy' and wxy.

Combination logic implementation using decoders:

Note that the outputs of a decoder represent the minterms of the input. By ORing these outputs we can implement any function of the inputs.

Example:

Implement a full adder.

x	y	z	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$S = \sum(1, 2, 4, 7)$

$C = \sum(3, 5, 6, 7)$

So: