Lecture 12, Feb. 26, 2007

Encoders

In the last lecture, we introduced the decoders. An encoder does the invers function of a decoder. That is, an encoder converts a symbol (say, one of 2^n possible symbols) into a bit stream of n bits. A familiar example of an encoder is an ASCII encoder. An ASCII encoder transforms an alphaneumetric symbol such as a computer key-board's keys into a 7-bit output suitable for the computer. When you type the letter A, for example, the output will be 1000001, and letter W will correspond to 1010111. The symbol > (greater than) is represented by 0111110, etc.

In general, an encoder will have 2^n (or less) inputs denoted by $D_0, D_1, \ldots$ and up to n outputs.

Let's consider the example of the octal (8-input) encoder. This encoder need to have 8 inputs $D_0, D_1, \ldots D_7$ and three outputs x, y and z. The value of the output represent the input that is active for example if $D_0 = 1$ and other inputs are 0. Then $xyz = 000$.

The truth table for the octal encoder is:

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	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

You may notice that the implementation of the encoder is quite simple. It can be implemented using OR gates:

$$z = D_1 + D_3 + D_5 + D_7$$

$$y = D_2 + D_3 + D_6 + D_7$$

$$x = D_4 + D_5 + D_6 + D_7$$

Note that all is well as long as <u>one</u> <u>and</u> <u>only</u> <u>one</u> input is selected.

As soon as two inputs become 1, the output becomes ambiguous. It is like pressing two keys on the key-board at the same time. Say, if we press 3 and 6 at the same time. Then $D_3 = D_6 = 1$ and $z = y = x = 1$. But, 111 represents D_7.

So, we need to put in some logic so that when two inputs are high (equal to 1) only one of them be given <u>priority</u> and be accepted. Also note that when there is no input, the output is 000 which is the same as the code for $D_0 = 1$.

To avoid these problems, we need:

- An extra output indicating whether any of the inputs is on or not.
- A <u>priority</u> logic deciding to

choose one of the inputs when more than one input is high.

Example: As an example, we take a 4-input encoder. Here, instead of two outputs x and y, we have three outputs x, y and V, where $V=1$ signifies the presence of input. $V=0$ means that no input is selected. We also add priority by giving priority to the input with the largest index. For example if D_0 and D_2 are both 1, we assume that the input is $D_2=1$. The truth table for this priority encoder is:

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

Note that $V = D_0 + D_1 + D_2 + D_3$.

The K-maps for x and y are:

D_0D_1 \ D_2D_3	00	01	11	10
00	X	1	1	1
01	0	1	1	1
11	0	1	1	1
10	0	1	1	1

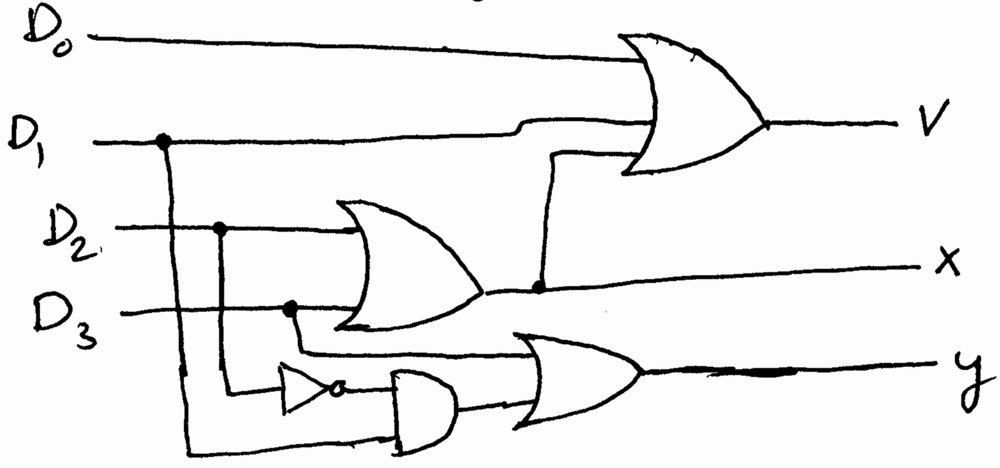
D_2

D_3

$$X = D_2 + D_3$$

D_0D_1 \ D_2D_3	00	01	11	10
00	X	1	1	0
01	1	1	1	0
11	1	1	1	0
10	0	1	1	0

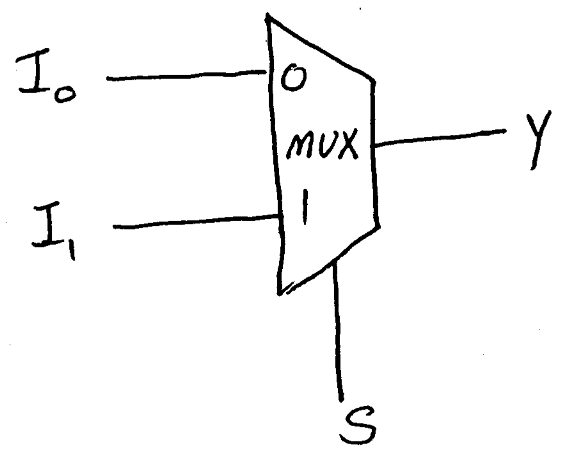
$$y = D_3 + D_1 D_2'$$



Multiplexers

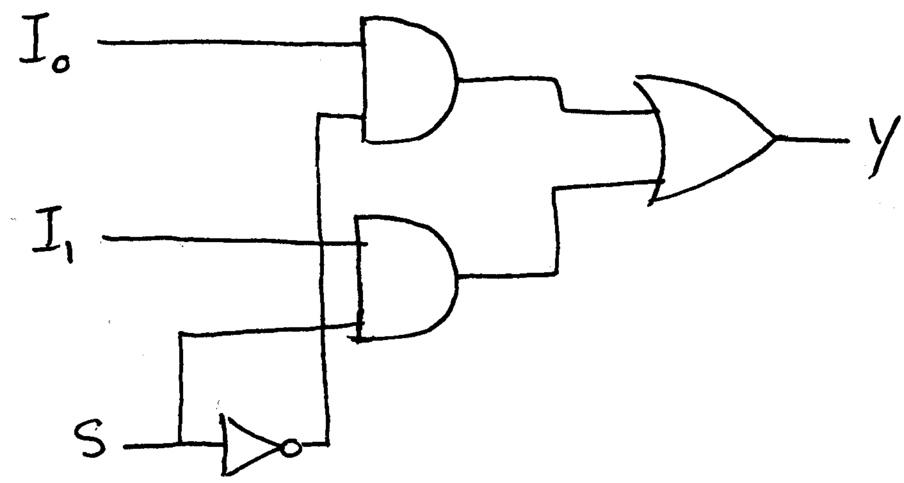
A *multiplexer* has several (usually 2^n) inputs and one output. It works like many-to-one switch. If the multiplexer has 2^n input, it needs n control (or select) inputs to say which input should be selected, i.e., connected to the output.

Example: A two-input multiplexer:



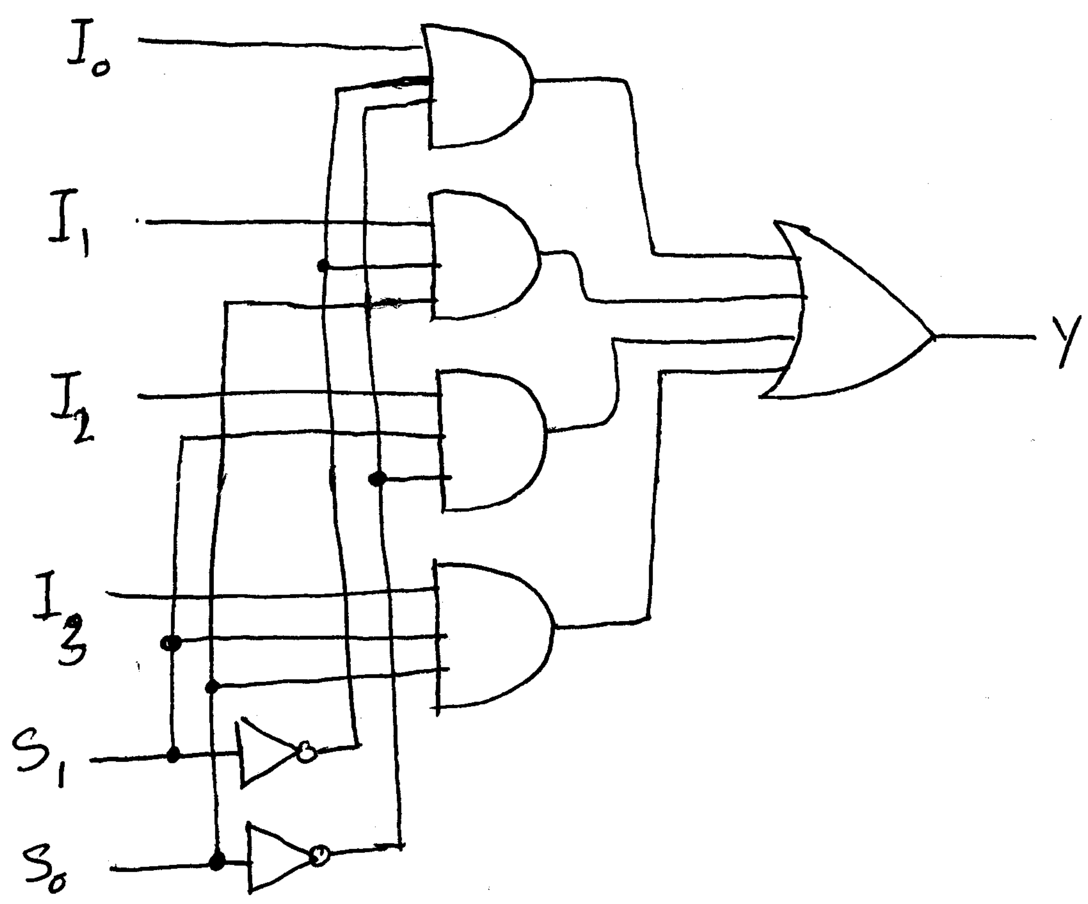
Here, the select bit decides which input should be connected to the output. If $S=0$, then $Y=I_0$ and when $S=1$, we have $Y=I_1$.

The 2-input multiplexer can be implemented as:



<u>Example</u>: A 4-to-1 multiplexer.

Here, we need $\log_2 4 = 2$ select inputs:



Boolean function implementation using multiplexers

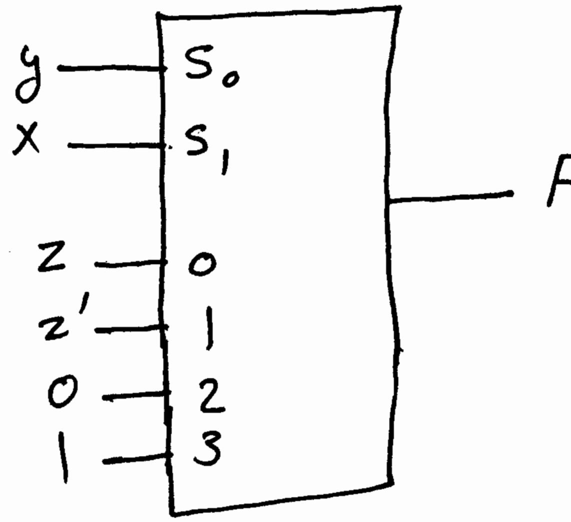
A Boolean function with n variables, can be implemented using a multiplexer with 2^{n-1} inputs. Such a multiplexer has n-1 selection lines. The first n-1 variables of the function are connected to the n-1 selection inputs of the multiplexer. The remaing variable, say, z, will be used for data inputs. Depending on the function, the inputs will receive z, z', 1 or 0.

Example: Implement

$$F(x,y,z) = \sum(1,2,6,7)$$

x	y	z	F	
0	0	0	0	F=z
0	0	1	1	
0	1	0	1	F=z'
0	1	1	0	
1	0	0	0	F=0
1	0	1	0	
1	1	0	1	F=1
1	1	1	1	

we connecte x and y to the 2 selection inputs of 4-to-1 MUX.



when $xy = 00$, $F = D_0$. From the truth table, we see that for $xy = 0$, $F = z$. So to D_0 we should connect z

When $xy = 01$, $F = D_1$. But, from the truth table $xy = 01 \Rightarrow F = z'$. So, connect z' to D_1.

similarly, connect 0 to D_2 and 1 to D_3.

Example: Implement

$$F(A,B,C,D) = \sum(1,3,4,11,12,13,14,15)$$

using a multiplexer.

since there are 4 variables, we need an 8-to-1 multiplexer.

A	B	C	D	F	
0	0	0	0	0	F=D
0	0	0	1	1	
0	0	1	0	0	F=D
0	0	1	1	1	
0	1	0	0	1	F=D'
0	1	0	1	0	
0	1	1	0	0	F=0
0	1	1	1	0	
1	0	0	0	0	F=0
1	0	0	1	0	
1	0	1	0	0	F=D
1	0	1	1	1	
1	1	0	0	1	F=1
1	1	0	1	1	
1	1	1	0	1	F=1
1	1	1	1	1