HOMEWORK 4 Solution

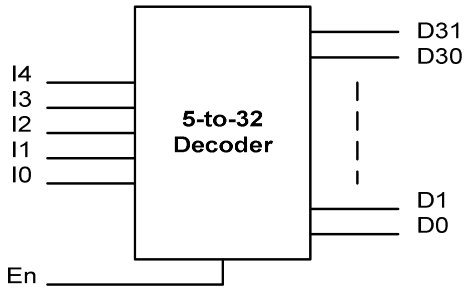
ICS 151 – Digital Logic Design

Spring 2004

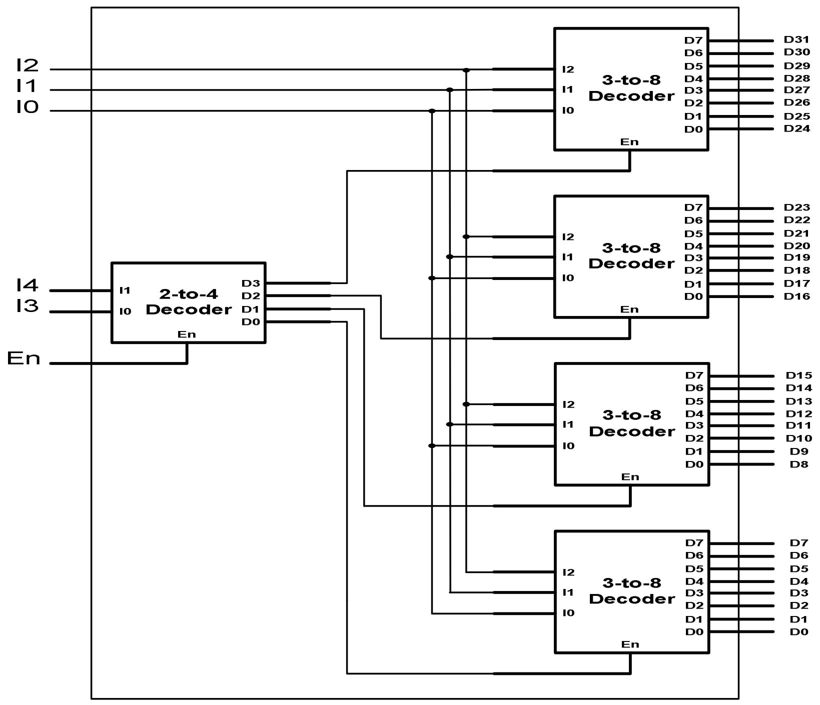
1. **Decoder/Multiplexer combining**
   a. Construct a 5-to-32 decoder using only 2-to-4 decoders and 3-to-8 decoders (with enable).
   b. Design a 32-to-1 multiplexer using only 8-to-1 multiplexer.

Use block diagram for the components.

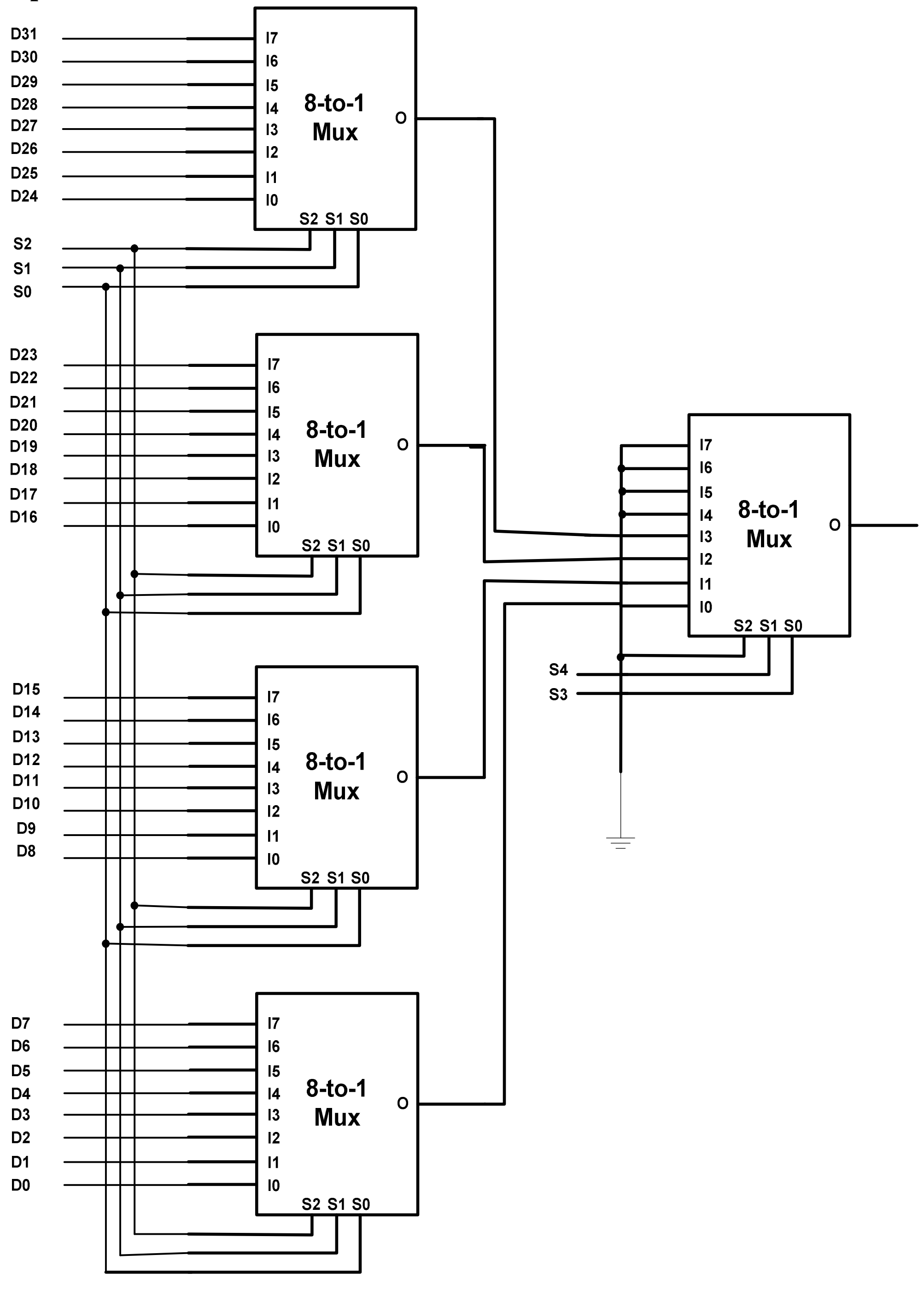
a. We are going to make 5-to-32 decoder like the one shown below:



We need four 3-to-8 decoder for the last stage and one 2-to-4 decoder for selecting each of them at the first stage:



b. We use four 8-to-1 in the first stage and another one to select between the output of the others:



2. Function implementation using Multiplexer

Realize the following functions:

$f_1(x,y,z) = \sum m(0,2,3,5,7)$

$f_2(x,y,z) = y'+z$

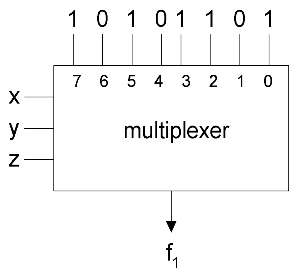
We need to have the minterm lists for both f1 and f2. For f1 we have it, for f2 we can reach to it by usning truth table or minterm expansion, we will use truth table for now:

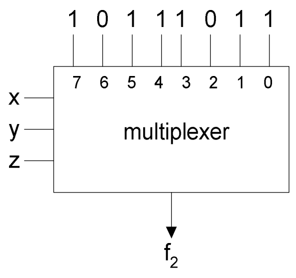
x	y	z	f_2
0	**0**	**0**	**1**
0	**0**	**1**	**1**
0	**1**	**0**	**0**
0	**1**	**1**	**1**
1	**0**	**0**	**1**
1	**0**	**1**	**1**
1	**1**	**0**	**0**
1	**1**	**1**	**1**

So f2 = $\sum$m (0, 1, 2, 3, 4, 5, 7)

a. **Using only 8-to-1 multiplexer**

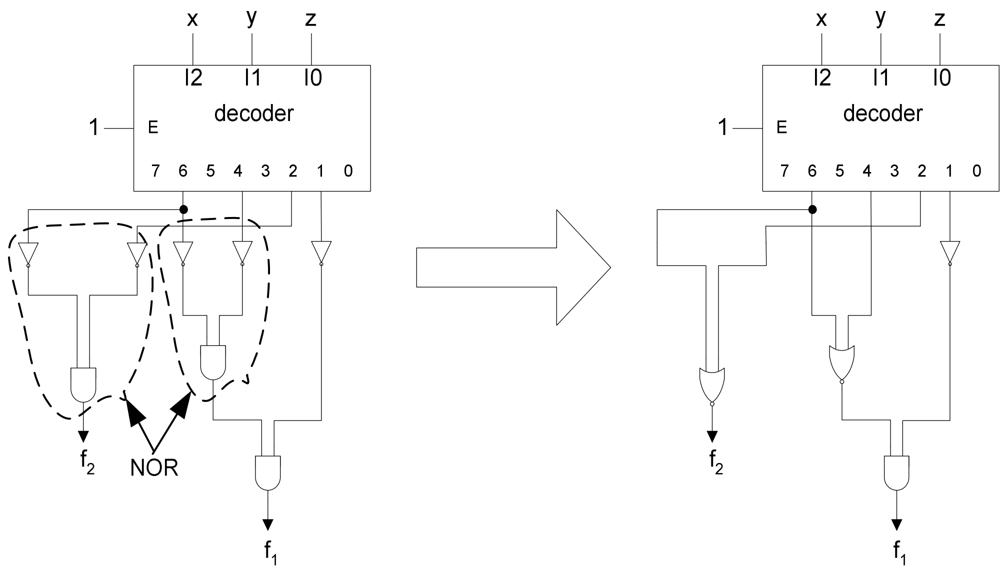
For each of f1 and f2 we need 1 8-to-1 multiplexer:





b. Using one 3-to-8 decoder and two-input gates (INV, AND, OR, NAND, NOR, XOR)

For implementing f1and f2 using decoder we need only one decoder, because we can share the decoder outputs.
And also because the number of 1's are more than the number of 0's for both of the functions, it's better to use INV-AND to make the functions, so:



We first derive the left diagram and then from there we will reach to the right digram by replacing INV-ANDs with NORs.

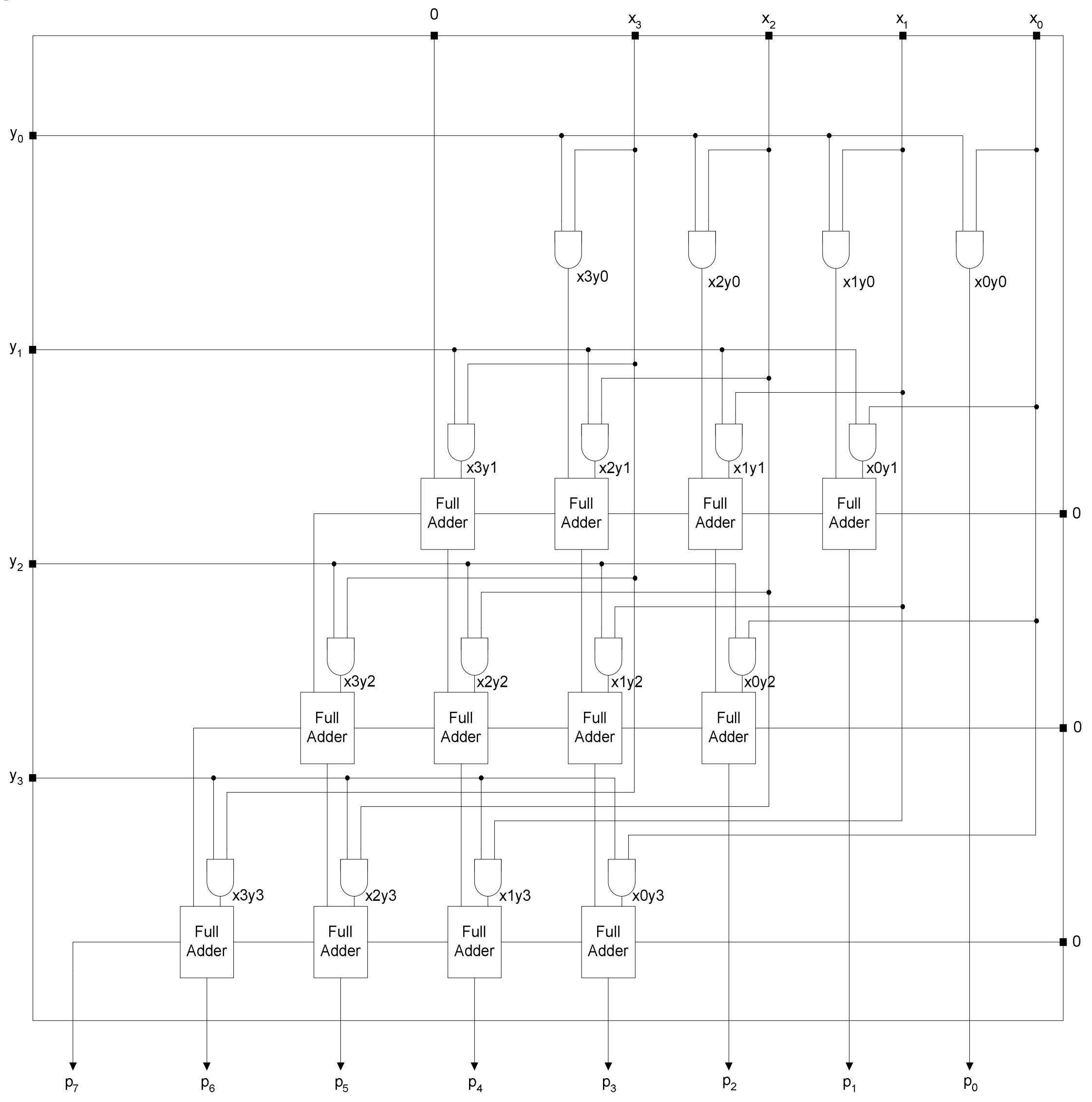
3. Combinational Multiplier Design

Design a binary multiplier that multiplies two 4-bit unsigned binary numbers. Use AND gates and full adder only.

Imagine that we want to multiply x3x2x1x0 by y3y2y1y0, for this multiplier we need to make a circuit that do the multiplication like below:

				x3y0	**x2y0**	**x1y0**	**x0y0**
			x3y1	**x2y1**	**x1y1**	**x0y1**	
		x3y2	**x2y2**	**x1y2**	**x0y2**		
	x3y3	**x2y3**	**x1y3**	**x0y3**			
P7	**P6**	**P5**	**P4**	**P3**	**P2**	**P1**	**P0**

All the ANDed sentences in each column should be added. So We need 16 2 input AND gates and 12 FAs to add all the columns. So, this is the final circuit:

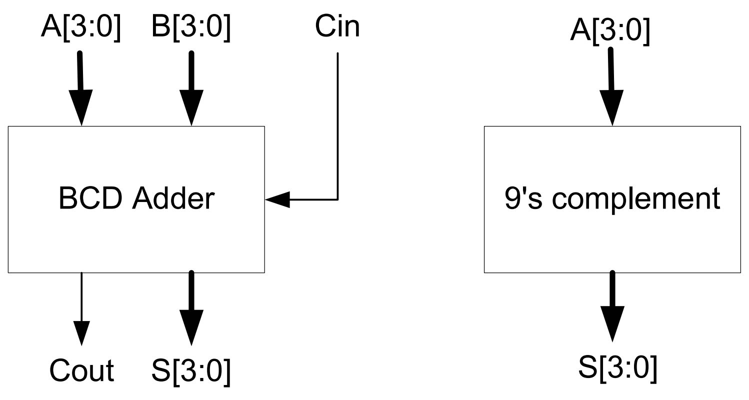


Module-based Design

Imagine that we have these two modules ready to use:

1) BCD adder
2) 9's complement circuit

The block diagrams for these modules are shown below: (A, B, S are all BCD digits)



For example if A=”1001” (BCD of decimal 9) and B=”0101” (BCD of decimal 5) then the outputs from BCD Adder will be C_{out}=1 and S = “0100” (“1 0100” is BCD of decimal 14).
And if A=”0111” then the output from 9's complement will be S=”0010”
Now we want to use the aforementioned modules to:

Design a 4-digit BCD adder-subtractor using four BCD adders (as shown above) and four 9's complement circuit (as shown above) and multiplexers. Your circuit has 3 set of inputs:

1) $X_3X_2X_1X_0$ (4-digit BCD input; for example 4896 or “0100 1000 1001 0110”)
2) $Y_3Y_2Y_1Y_0$ (4-digit BCD input; for example 1267 or “0001 0010 0110 0111”)
3) M (mode input; M=0 means add and M=1 means subtract)

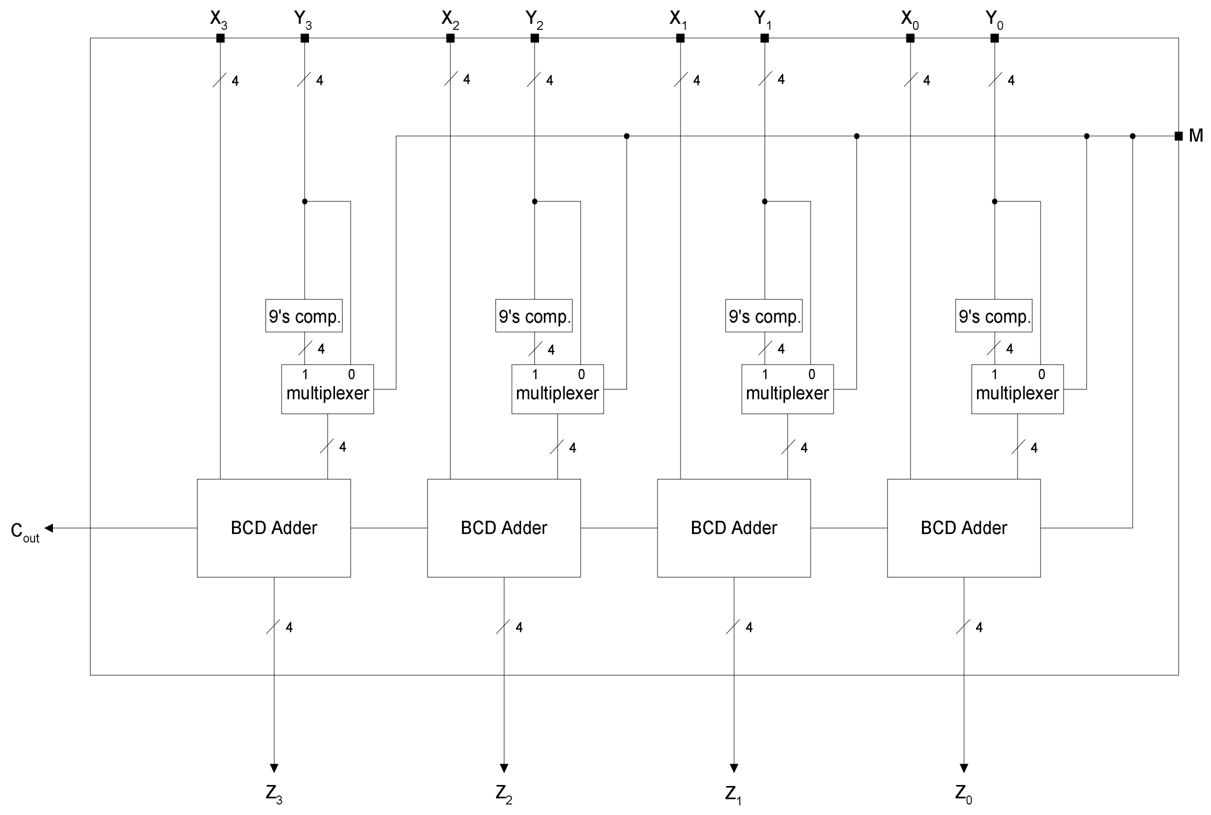
And it has two sets of outputs:

1) $Z_3Z_2Z_1Z_0$ (4-digit BCD output showing the result of the add or subtract; for example the result of the add will be 6163 or “0110 0001 0110 0011” and the result of the subtract will be 3629 or “0011 0110 0010 1001” for the above examples)
2) C_{out} (Carry output)

Use block diagrams for each component, showing only inputs and outputs.

The design of 4-digit BCD adder/subtractor is almost the same as the design of 4 bit adder/subtractor. In case of the adder we should just add the 4 BCD digits using the 4 BCD adders that we have. For the subtraction we need to add the first number and the 9's complement and 1. Four 9's complements modules used for generating the 9's complement of the 4 BCD digits and the carry in of the first stage is used to feed the 1 that we need during the subtraction, so we can connect it to mode (M) input.

And finally we need 4 quadruple 2-to-1 multiplexers to select between the second variable and its 9's complement for addition and subtraction. The select line will be connected to mode (M). Each of these quadruple 2-to-1 multiplexers is consisting of four 2-to-1 multiplexers that have a shared select line. The final design block diagram is given below:



X3
Y3
X2
Y2
X1
Y1
X0
Y0
M
4
9's comp.
1 0
multiplexer
BCD Adder
Cout
Z3
Z2
Z1
Z0