# MSI Design Examples

In this lesson, you will see some design examples using MSI devices. These examples are:

- Designing a circuit that adds three 4-bit numbers.
- Design of a 4-to-16 Decoder using five 2-to-4 Decoders with enable inputs.
- Design of a circuit that takes 2 unsigned 4-bit numbers and outputs the larger of both.
- Designing a 16-bit adder using four 4-bit adders.
- Designing a 3-bit excess-3 code converter using a Decoder and an Encoder.

## Designing a circuit that adds three 4-bit numbers

Recall that a 4-bit binary adder adds two binary numbers, where each number is of 4 bits. For adding three 4-bit numbers we have:

Inputs

- First *4*-bit number X = $X_3X_2X_1X_0$
- Second *4*-bit number Y = $Y_3Y_2Y_1Y_0$
- Third *4*-bit number Z = $Z_3Z_2Z_1Z_0$

Outputs

The summation of X, Y, and Z. How many output lines are exactly needed will be discussed as we proceed.

To design a circuit using MSI devices that adds three 4-bit numbers, we first have to understand how the addition is done. In this case, the addition will take place in two steps, that is, we will first add the first two numbers, and the resulting sum will be added to the third number, thus giving us the complete addition.
Apparently it seems that we will have to use two 4-bit adders, and probably some extra hardware as well. Let us analyze the steps involved in adding three 4-bit numbers.

### Step 1: Addition of X and Y

A 4-bit adder is required. This addition will result in a sum and a possible carry, as follows:

$X_3X_2X_1X_0$
$Y_3Y_2Y_1Y_0$
---
$C_4$ $S_3S_2S_1S_0$

Note that the input carry $C_{in}$ = 0 in this 4-bit adder

### Step 2: Addition of S and Z

This resulting partial sum (i.e. $S_3S_2S_1S_0$) will be added to the third 4-bit number $Z_3Z_2Z_1Z_0$ by using another 4-bit adder as follows, resulting in a final sum and a possible carry:

$S_3S_2S_1S_0$
$Z_3Z_2Z_1Z_0$
---
$D_4$ $F_3F_2F_1F_0$

where $F_3F_2F_1F_0$ represents the final sum of the three inputs X, Y, and Z. Again, in this step, the input carry to this second adder will also be zero.

Notice that in Step 1, a carry $C_4$ was generated in bit position 4, while in Step 2, another carry $D_4$ was generated also in bit position 4. These two carries must be added together to generate the final Sum bits of positions 4 and 5 ($F_4$ and $F_5$).
Adding $C_4$ and $D_4$ requires a half adder. Thus, the output from this circuit will be six bits, namely $F_5$ $F_4$ $F_3F_2F_1F_0$ (See Figure 1)

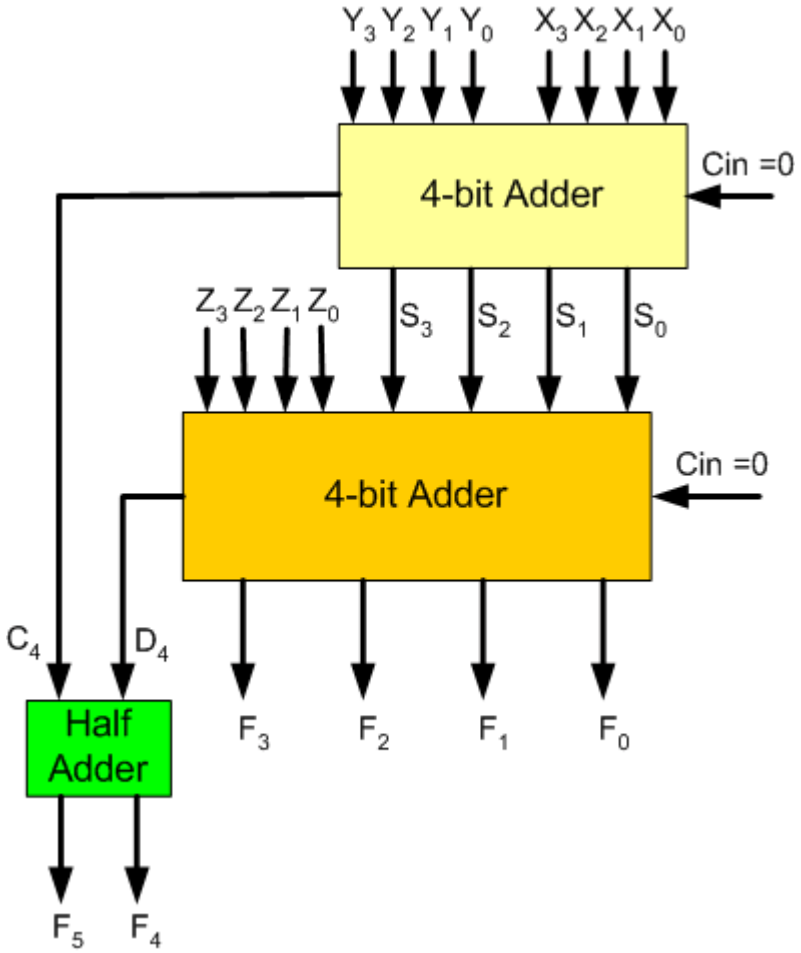


Figure 1: Circuit for adding three 4-bit numbers

## Design a 4-to-16 Decoder using five 2-to-4 Decoders with enable inputs

We have seen how can we construct a bigger decoder using smaller decoders, by taking the specific example of designing a 3-to-8 decoder using two 2-to-4 decoders. Now we will design a 4-to-16 decoder using five 2-to-4 decoders.

There are a total of sixteen possible input combinations, as shown in the table (Figure 2). These sixteen combinations can be divided into four groups, each group containing four combinations. Within each group, $A_3$ and $A_2$ remain constant, while $A_1$ and $A_0$ change their values. Also, in each group, same combination is repeated for $A_1$ and $A_0$ (i.e. 00→01→10→11)

| $A_3$ | $A_2$ | $A_1$ | $A_0$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 |

Figure 2: Combinations with 4 variables

Thus we can use a 2-to-4 decoder for each of the groups, giving us a total of four decoders (since we have sixteen outputs; each decoder would give four outputs). To each decoder, $A_1$ and $A_0$ will go as the input.

A fifth decoder will be used to select which of the four other decoders should be activated. The inputs to this fifth decoder will be $A_3$ and $A_2$. Each of the four outputs of this decoder will go to each enable of the other four decoders in the "proper order".

This means that line 0 (representing $A_3A_2 = 00$) of decoder '5' will go to the enable of decoder '1'. Line 1 (representing $A_3A_2 = 01$) of decoder '5' will go to the enable of decoder '2' and so on.

Thus a combination of $A_3$ and $A_2$ will decide which "group" (decoder) to select, while the combination of $A_1$ and $A_0$ will decide which output line of that particular decoder is to be selected.

Moreover, the enable input of decoder '5' will be connected to logic switch, which will provide logic 1 value to activate the decoder.

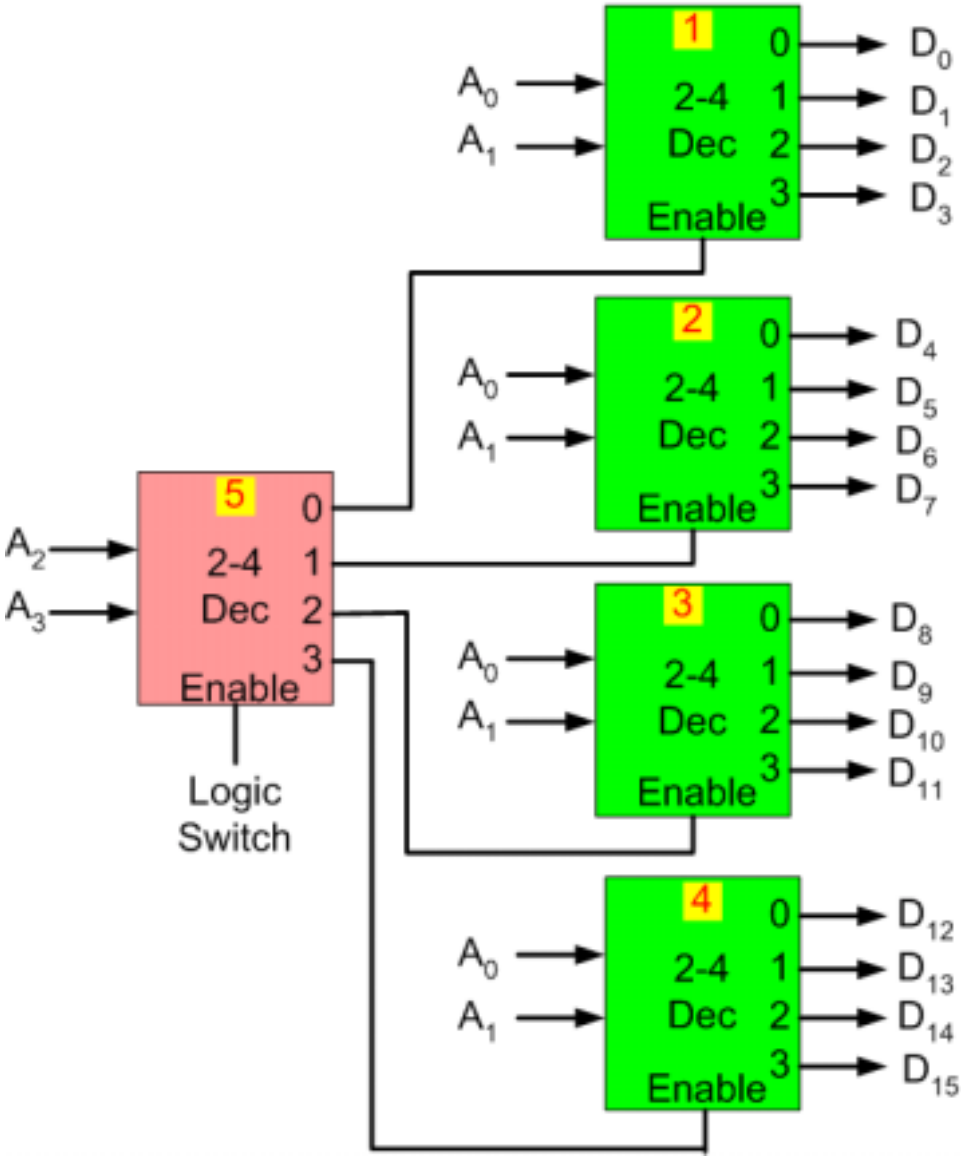


Figure 3: Constructing 4-to-16 decoder using 2-to-4 decoders

Decoder example: "Activate" line $D_2$. The corresponding input combination that would activate this line is 0010. Now apply 00 at input of decoder '5'. This activates line '0' connected to enable of decoder '1'. Once decoder '1' is activated, inputs at $A_1A_0$ = 10 activate line $D_2$.
Thus we get the effect of a 4-16 decoder using this design, by applying input combinations in two steps.

As another example, to "activate" the line $D_{10}$: The corresponding input combination is 1010. Apply 10 at the input of decoder '5'. This activates line '2' connected to enable of decoder '3'. Once decoder '3' is activated, the inputs at $A_1A_0$ = 10 activate line $D_{10}$.

## Given two 4-bit unsigned numbers A and B, design a circuit which outputs the larger of the 2 numbers.

Here we will use Quad 2-1 Mux, and a 4-bit magnitude comparator. Both of these devices have been discussed earlier. The circuit is given in the figure

Since we are to select one of the two 4-bit numbers A ($A_3A_2A_1A_0$) and B ($B_3B_2B_1B_0$), it is obvious that we will need a quad 2-1 Mux.

The inputs to this Mux are the two 4-bit numbers A and B.

The select input of the Mux must be a signal which indicates the relative magnitude of the two numbers A and B. This signal may be True if A<B or if A>B.

Such signal is easily obtained from a 4-bit magnitude comparator.

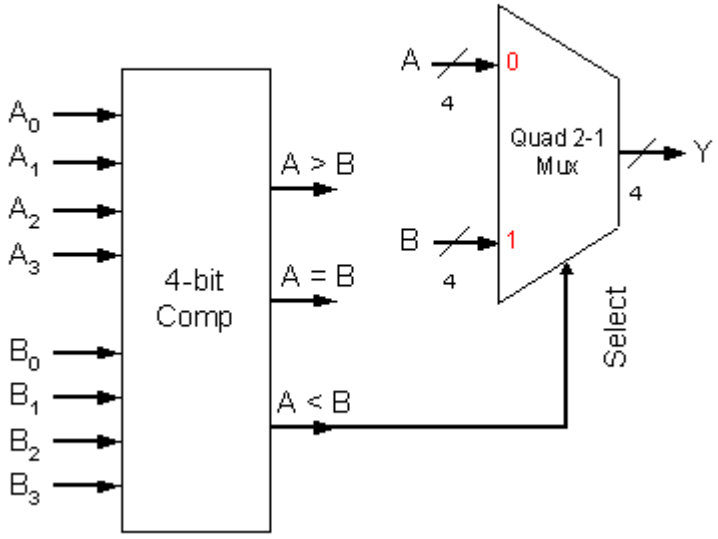


Figure 4: Circuit that outputs the larger of two numbers

By connecting the select input to the A<B output of the magnitude comparator, we must connect A to the 0 input of the Mux and B to the 1 input of the Mux . Alternatively, if we connect the select input to the A>B output of the magnitude comparator, we must connect A to the 1 input of the Mux and B the 0 input of the Mux . In either case, the Mux output will be the larger of the two numbers

## Designing a 16-bit adder using four 4-bit adders

Adds two 16-bit numbers X ($X_0$ to $X_{15}$), and Y ($Y_0$ to $Y_{15}$) producing a 16-bit Sum S ($S_0$ to $S_{15}$) and a carry out $C_{16}$ as the most significant position. Thus, four 4-bit adders are connected in cascade.

Each adder takes four bits of each input (X and Y) and generates a 4-bit sum and a carry that is fed into the next 4-bit adder as shown in Figure 5.

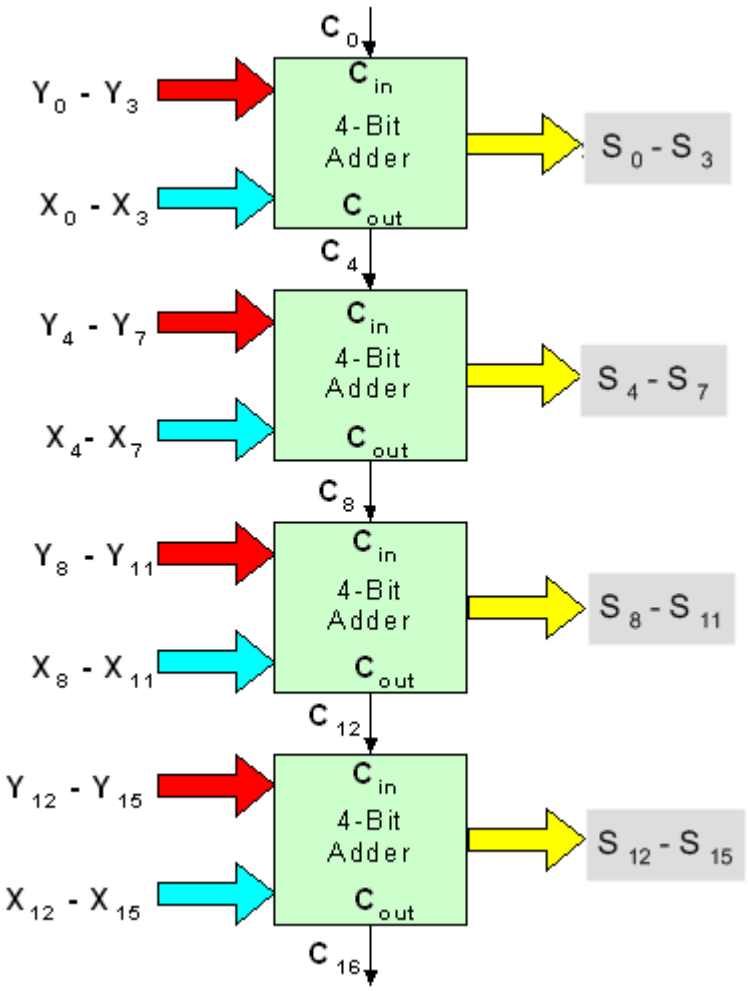


Figure 5: A 16-bit adder

## Designing an Excess-3 code converter using a Decoder and an Encoder

In this example, the circuit takes a BCD number as input and generates the corresponding Ex-3 code. The truth table for this circuit is given in figure 6.
The outputs 0000, 0001, 0010, 1101, 1110, and 1111 are never generated (Why?)

To design this circuit, a 4-to-16 decoder and a 16-to-4 encoder are required. The design is given in figure 7. In this circuit, the decoder takes 4 bits as inputs, represented by variables w, x, y, and z. Based on these four bits, the corresponding minterm output is activated. This decoder output then goes to the input of encoder which is three greater than the value generated by the decoder.

The encoder then encodes the value and sends the output bits at A, B, C, and D. For example, suppose 0011 is sent as input. This will activate minterm 3 of the decoder. This

output is connected to input 6 of encoder. Thus the encoder will generate the corresponding bit combination, which is 0110.

| w | x | y | z | A | B | C | D |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

Figure 6: table for BCD to Ex-3 conversion

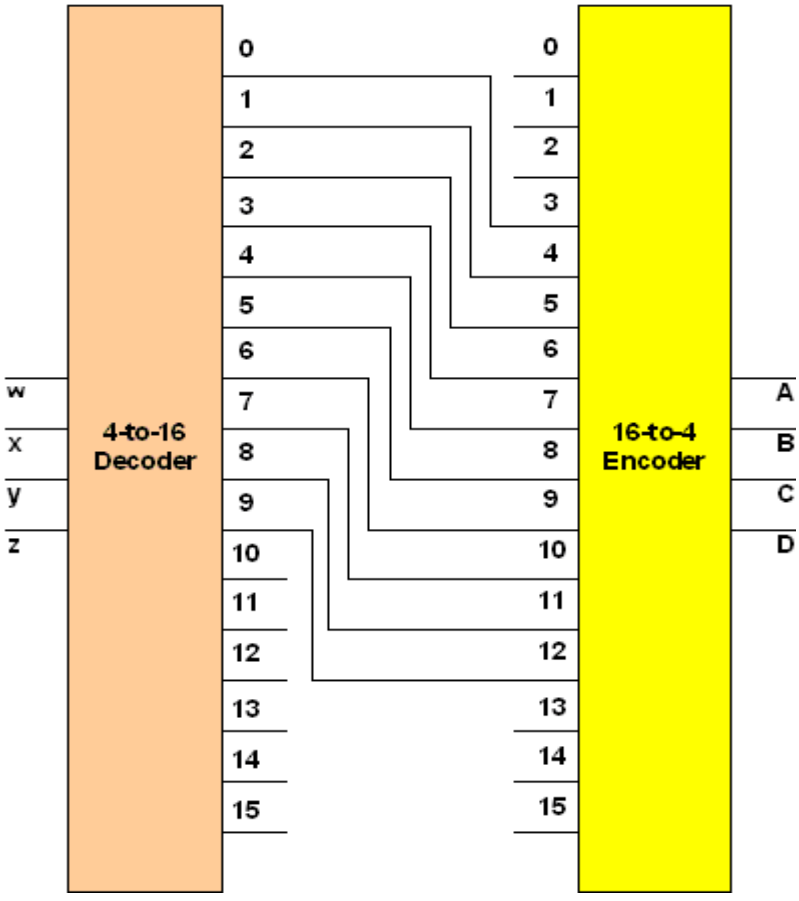


Figure 7: Circuit for BCD to Ex-3 conversion