EE 2720

Handout # 18.

- Decoders continued.
- Decoders with an Enable Input.

Usually decoders have one or more additional input lines that are referred to as enable inputs. When the enable input(s) is (are) active, the decoder acts like a decoder, otherwise not. Fig. 1 below shows a 2-to-4 decoder with an active high enable input named E; (active high E means that if E=1, then the decoder is enabled; if E=0, then the decoder is not enabled).

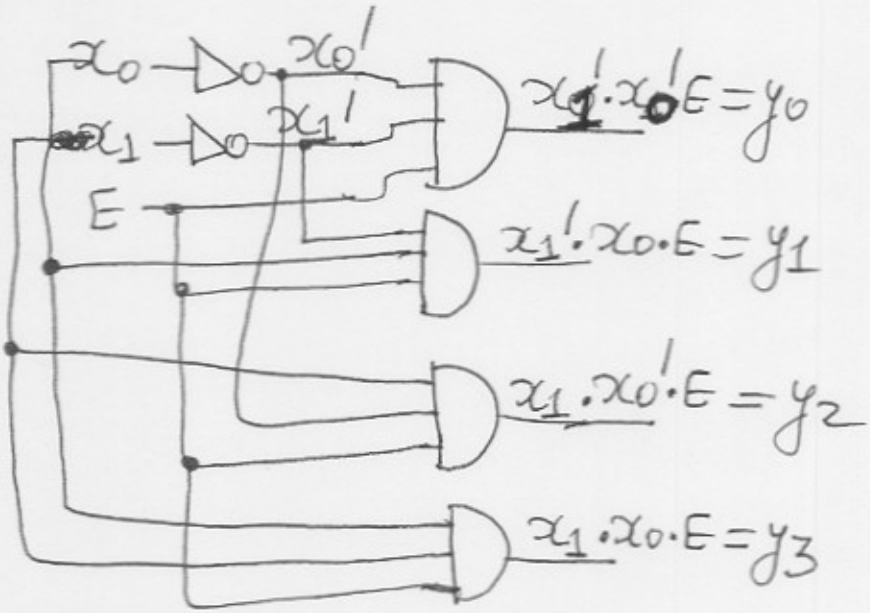

Fig. 1(a). Detailed diagram for a 2-to-4 decoder with an active high enable input E

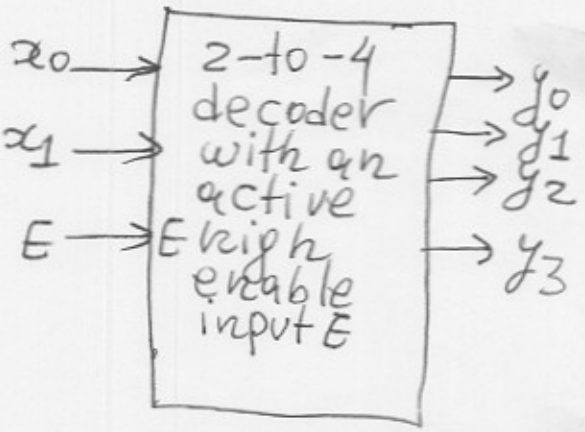

Fig. 1(b). Symbol for a 2-to-4 decoder with an active high enable input E.

The truth table of the above decoder of Fig. 1 is shown in table 1.

E	x_1	x_0	z_0	z_1	z_2	z_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

Table 1: Truth table of a 2-to-4 decoder with an active high enable input E.

- How to read the above truth table 1:

This is a compressed truth table. Although we have three inputs E, x_1, x_0, instead of having $2^3=8$ rows in the truth table, we only have five rows. An X in the inputs means that the corresponding inputs can take any values. For example, the top row of the truth table that corresponds to Ex_1x_0 = 0xx means $E=0$ and x_1, x_0 taking any values which means 00, 01, 10, 11.

Let me now show you how to use five decoders of figure 1 (2-to-4 decoders with active high enable input) to implement a 4-to-16 decoder. This 4-to-16 decoder is shown in figure 2 on next page.

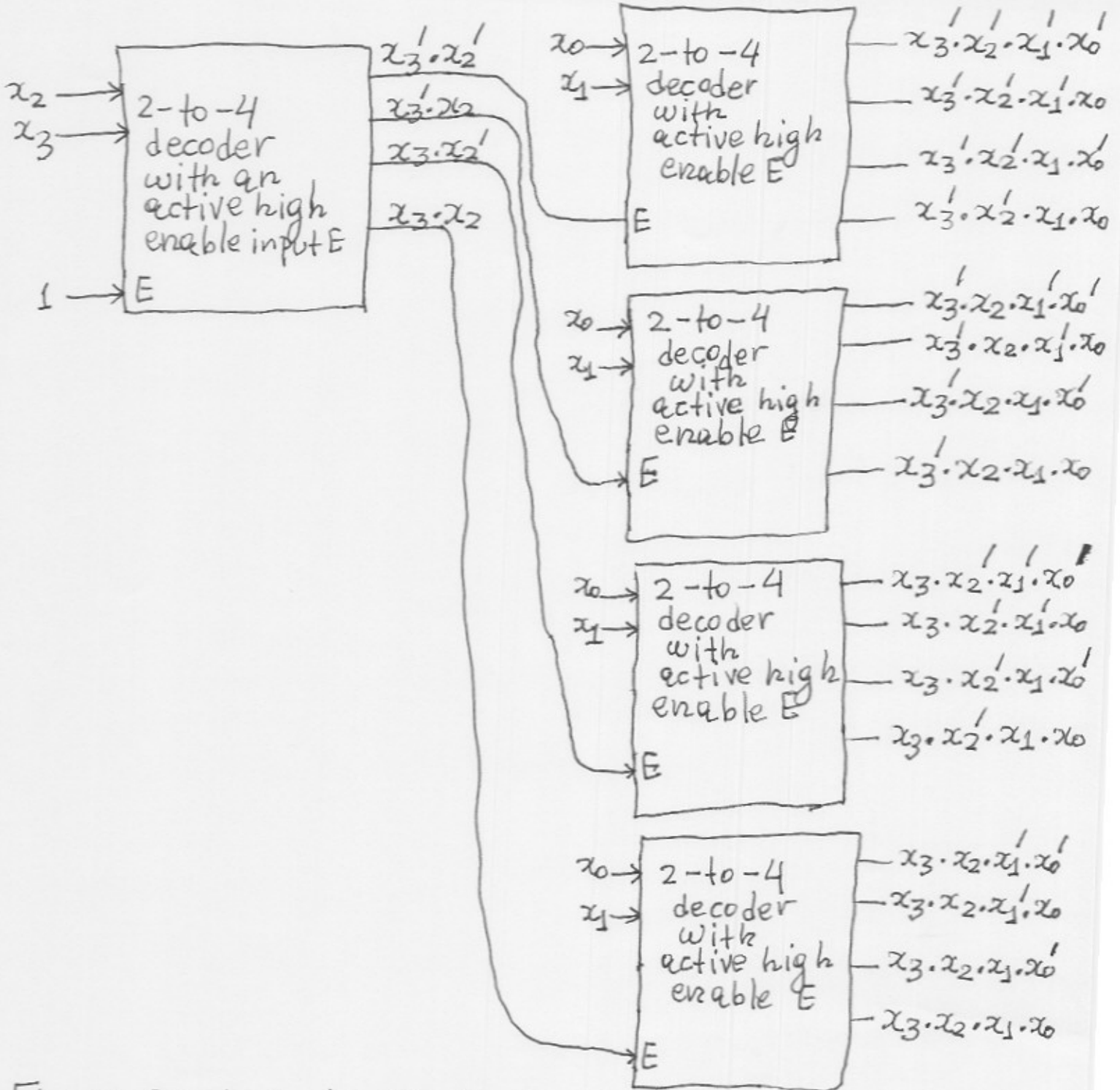

Figure 2: A 4-to-16 decoder constructed from five 2-to-4 decoders with enables. Here, x_3 is the MSB of the vector to be decoded, while x_0 is the LSB of the vector to be decoded.

• Decoders with more than one enable inputs

In most decoders, (and actually most digital systems), enable inputs are active low. This means that a 0 does the job. For a decoder, for example, an active low enable input means that if Enable=0, then the decoder is enabled, (it acts like a decoder), otherwise if Enable=1 the decoder is not enabled. An active low enable is denoted by a bubble. Figure 3 below shows a 2-to-4 decoder with an active low enable input while its truth table is shown in table 2.

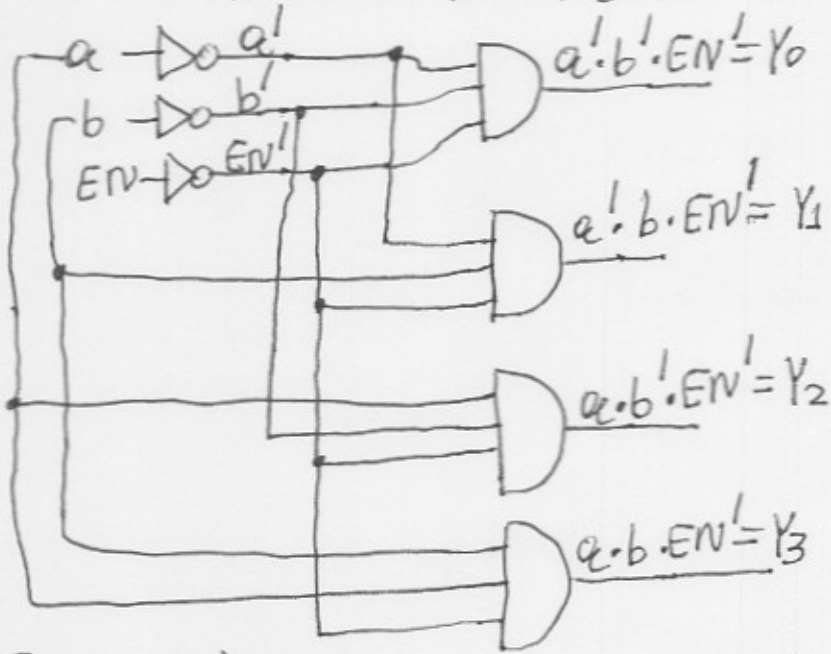

Fig. 3(a). Detailed diagram for a 2-to-4 decoder with an active low enable input EN.

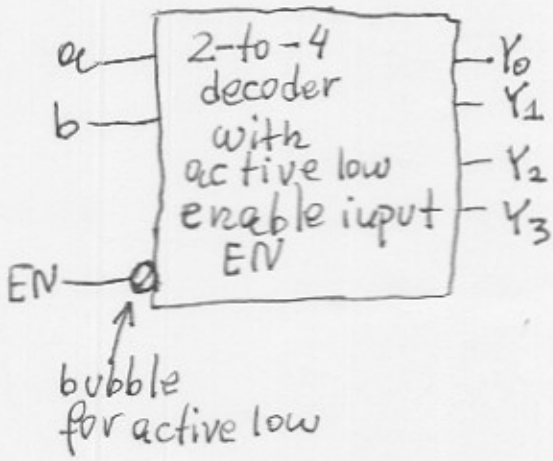

Fig. 3(b). Symbol for a 2-to-4 decoder with an active low enable input EN.

The truth table of the above decoder is shown in table 2 on next page.

EN	a	b	Y_0	Y_1	Y_2	Y_3
1	X	X	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1

Table 2: Truth table for a 2-to-4 decoder with an active low enable input EN.

I will now present the topic of decoders with more than one enable inputs; (some of them will be active high and some active low).
In figure 4 below you can see a 3-to-8 decoder with three enable inputs; (one active high and two active low). The enable inputs are EN1, EN2, EN3

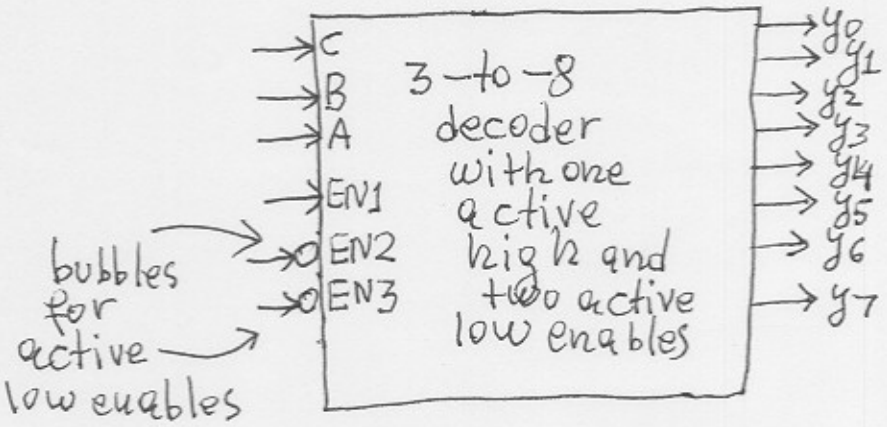

Fig. 4. A 3-to-8 decoder with one active high and two active low enable inputs.

The truth table of the above 3-to-8 decoder of figure 4 is shown in table 3 on the next page.

disabled

EN1	EN2	EN3	C	B	A	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
0	X	X	X	X	X	0	0	0	0	0	0	0	0
X	1	X	X	X	X	0	0	0	0	0	0	0	0
X	X	1	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	1	0	1	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	0	0	0	0	0
1	0	0	0	1	1	0	0	0	1	0	0	0	0
1	0	0	1	0	0	0	0	0	0	1	0	0	0
1	0	0	1	0	1	0	0	0	0	0	1	0	0
1	0	0	1	1	0	0	0	0	0	0	0	1	0
1	0	0	1	1	1	0	0	0	0	0	0	0	1

Table 3: Truth table of the 3-to-8 decoder with one active high and two active low enable inputs. The enable inputs are EN1, EN2, EN3.

I will now show you how to cascade (connect) four decoders of figure 4, (3-to-8 decoders), to implement a 5-to-32 decoder. This is shown in figure 5 on the next page.

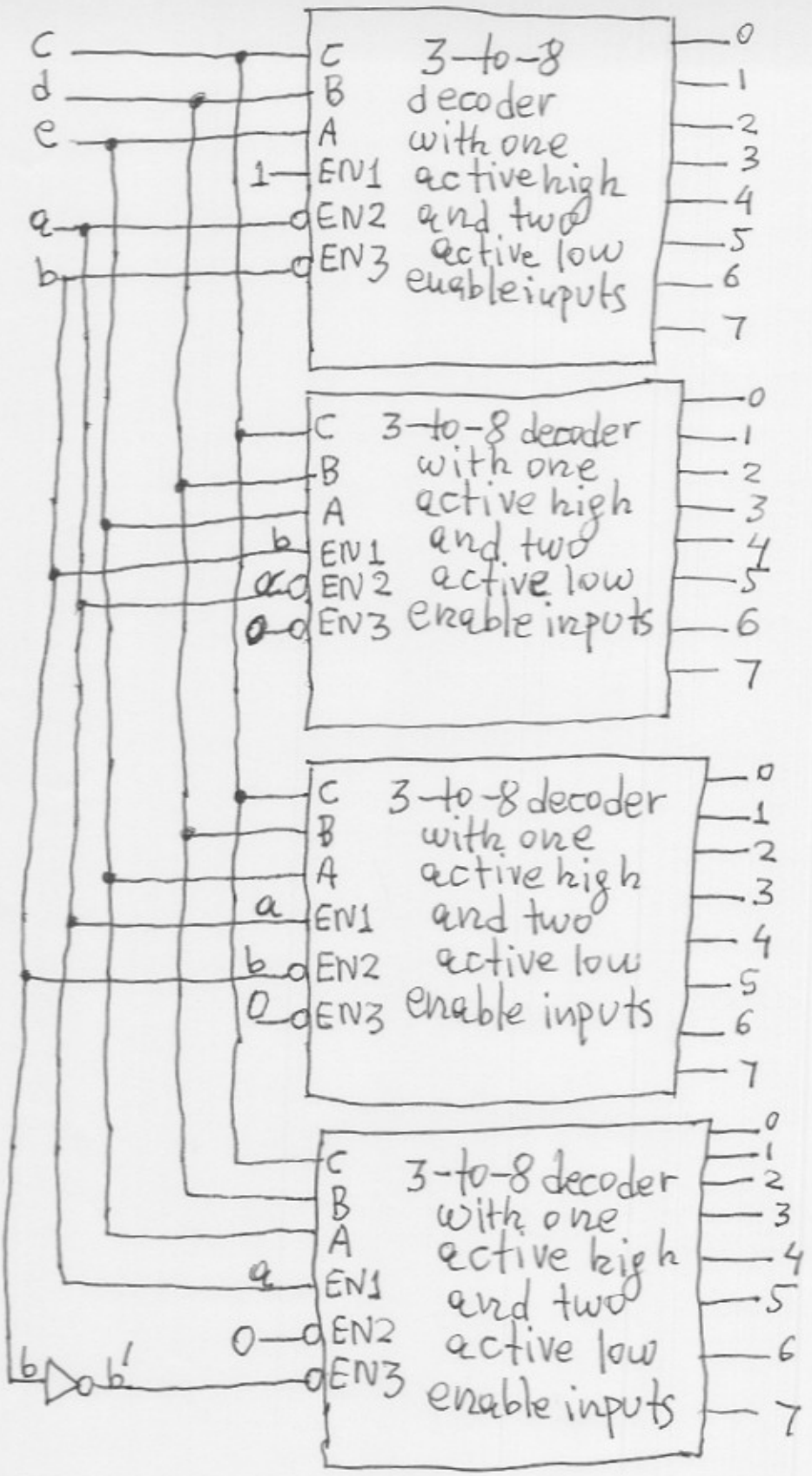

Figure 5: A 5-to-32 decoder constructed from 3-to-8 decoders with enable inputs.

Explanations about the decoder of figure 5

Although the decoder of figure 5 is self explanatory, I will still provide some explanations. At first, observe that the inputs cde are inputs to all four decoders (3-to-8 decoders of fig. 5 I mean). Then observe that when ab=00, then the top most decoder is enabled and all other 3-to-8 decoders are disabled. Thus, the top most 3-to-8 decoder gives as outputs 10000000, 01000000, 00100000, ..., 00000001, while all the bottom most 3-to-8 decoders give as outputs 0's. When ab=01, then the second from the top 3-to-8 decoder is enabled and all other 3-to-8 decoders are disabled. Thus, the second from the top 3-to-8 decoder gives as outputs 10000000, 01000000, ..., 00000001, while all other decoders give as outputs 0's. When ab=10, then the third from the top decoder is enabled and all other decoders are disabled. Finally, when ab=11, the bottom most decoder is enabled and all other decoders are disabled. This way, exactly one output of the 5-to-32 decoder is 1 and all the others 0. I hope that fig. 5 is clear by now. For the 5-to-32 decoder of fig. 5, a is the MSB of the decoded vector, while e is the LSB of the decoded vector. Also, if the 3-to-8 decoders had all enable inputs active high, we would need more than one inverters.

• Encoders:

An encoder performs the inverse function of a decoder. The simplest encoder is the binary encoder or the 2^n-to-n encoder. Table 4 below shows the truth table of an 8-to-3 encoder. Its inputs are $I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7$ and its outputs are Y_2, Y_1, Y_0.

I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	Y_2	Y_1	Y_0
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

Table 4: Truth table for an 8-to-3 encoder.

Notice that only one of the inputs of the encoder is 1 and all the others 0. If input $I_i = 1$, then the output $Y_2Y_1Y_0$ represents the number i.

Figure 6 below shows the symbol for a 2^n-to-n encoder.

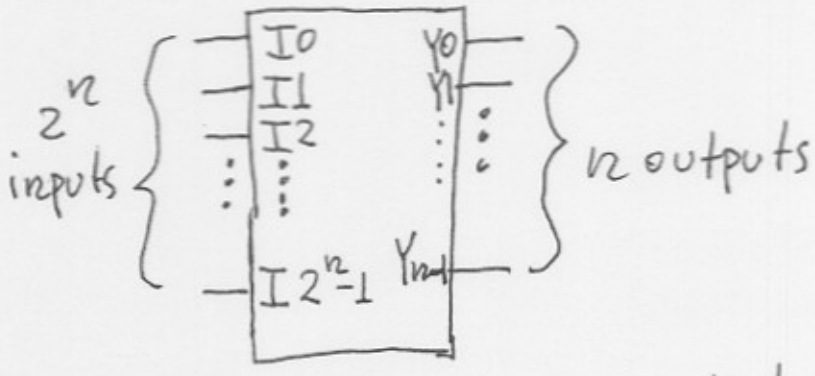

Figure 6: The symbol of a 2^n-to-n encoder.

We can now easily derive the logic equations for the 8-to-3 encoder described in table 4. We have:

$$Y2 = I4 + I5 + I6 + I7 \quad (1)$$
$$Y1 = I2 + I3 + I6 + I7 \quad (2)$$
$$Y0 = I1 + I3 + I5 + I7 \quad (3)$$

From the above equations (1), (2), (3) we can easily get the circuit realization for the 8-to-3 encoder which is shown in figure 7 below:

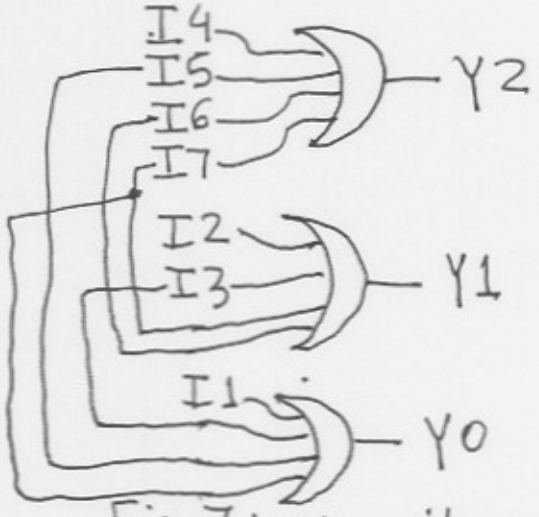

Fig. 7: Circuit realization for the 8-to-3 encoder.

• Priority encoders:

Priority encoders assign priority to the input lines, so that when multiple lines become 1 at the same time, the encoder will produce the highest priority requestor. The truth table of a priority encoder with eight inputs named I0, I1, I2, I3, I4, I5, I6, I7 and three outputs named Y2, Y1, Y0 is shown in table 5 below:

I0	I1	I2	I3	I4	I5	I6	I7	Y2	Y1	Y0
1	0	0	0	0	0	0	0	0	0	0
X	1	0	0	0	0	0	0	0	0	1
X	X	1	0	0	0	0	0	0	1	0
X	X	X	1	0	0	0	0	0	1	1
X	X	X	X	1	0	0	0	1	0	0
X	X	X	X	X	1	0	0	1	0	1
X	X	X	X	X	X	1	0	1	1	0
X	X	X	X	X	X	X	1	1	1	1

Table 5: The truth table of a priority encoder with eight inputs and three outputs.

Note: For the above priority encoder described in table 5, I7 is the input with the highest priority, I6 is the second highest priority input, ..., etc, while I0 is the lowest priority input. That is to say that if I7 = 1, no matter what the values of the other inputs are, the output will be Y2Y1Y0 = 111 representing number 7; if I7 = 0 and I6 = 1, no matter what the values of the other

inputs are, the output will be $Y2Y1Y0 = 110$ representing number 6 etc.

In order to write logic equations for the outputs of the priority encoder described in table 5, we first define eight intermediate variables H0–H7, such that H_i is 1 if and only if input I_i is the highest priority 1 input.

$$H7 = I7$$
$$H6 = I6 \cdot I7'$$
$$H5 = I5 \cdot I6' \cdot I7'$$
$$H4 = I4 \cdot I5' \cdot I6' \cdot I7'$$
$$H3 = I3 \cdot I4' \cdot I5' \cdot I6' \cdot I7'$$
$$H2 = I2 \cdot I3' \cdot I4' \cdot I5' \cdot I6' \cdot I7'$$
$$H1 = I1 \cdot I2' \cdot I3' \cdot I4' \cdot I5' \cdot I6' \cdot I7'$$
$$H0 = I0 \cdot I1' \cdot I2' \cdot I3' \cdot I4' \cdot I5' \cdot I6' \cdot I7'$$

Using these H0–H7 signals, the equations for the outputs Y2, Y1, Y0 are similar to the ones for a simple binary encoder.

$$Y2 = H4 + H5 + H6 + H7$$
$$Y1 = H2 + H3 + H6 + H7$$
$$Y0 = H1 + H3 + H5 + H7$$

From the above equations, you can easily get the circuit realization for the priority encoder. Do it if you want.

• An interesting application of the priority encoder:

Consider the Central Processing Unit (CPU) of a Computer. The CPU is the main heart of the computer. Now consider, for example, eight peripheral devices called $d0, d1, d2, d3, d4, d5, d6, d7$. These devices are requesting service from the CPU of the computer. The device $d7$ has the highest priority, the device $d6$ has the second highest priority, ..., etc, while the device $d0$ has the lowest priority. That means that if device $d7$ requests service, no matter if the other seven devices request service or not, its request will be granted first; if device $d7$ is not requesting service and device $d6$ is requesting service no matter if the other six devices request service or not, its request will be granted etc. Each device di produces as an output a one-bit signal named ri. If $ri=1$ that means that device di is requesting service from the CPU of the computer. The CPU responds with eight signals named $g0, g1, g2, g3, g4, g5, g6, g7$. Signal gi is input to device di. If $gi=1$, that means that service is granted to device di. The CPU of the computer must grant service to the device with the highest priority. Figure 8 on next page

shows the implementation of the above scheme.

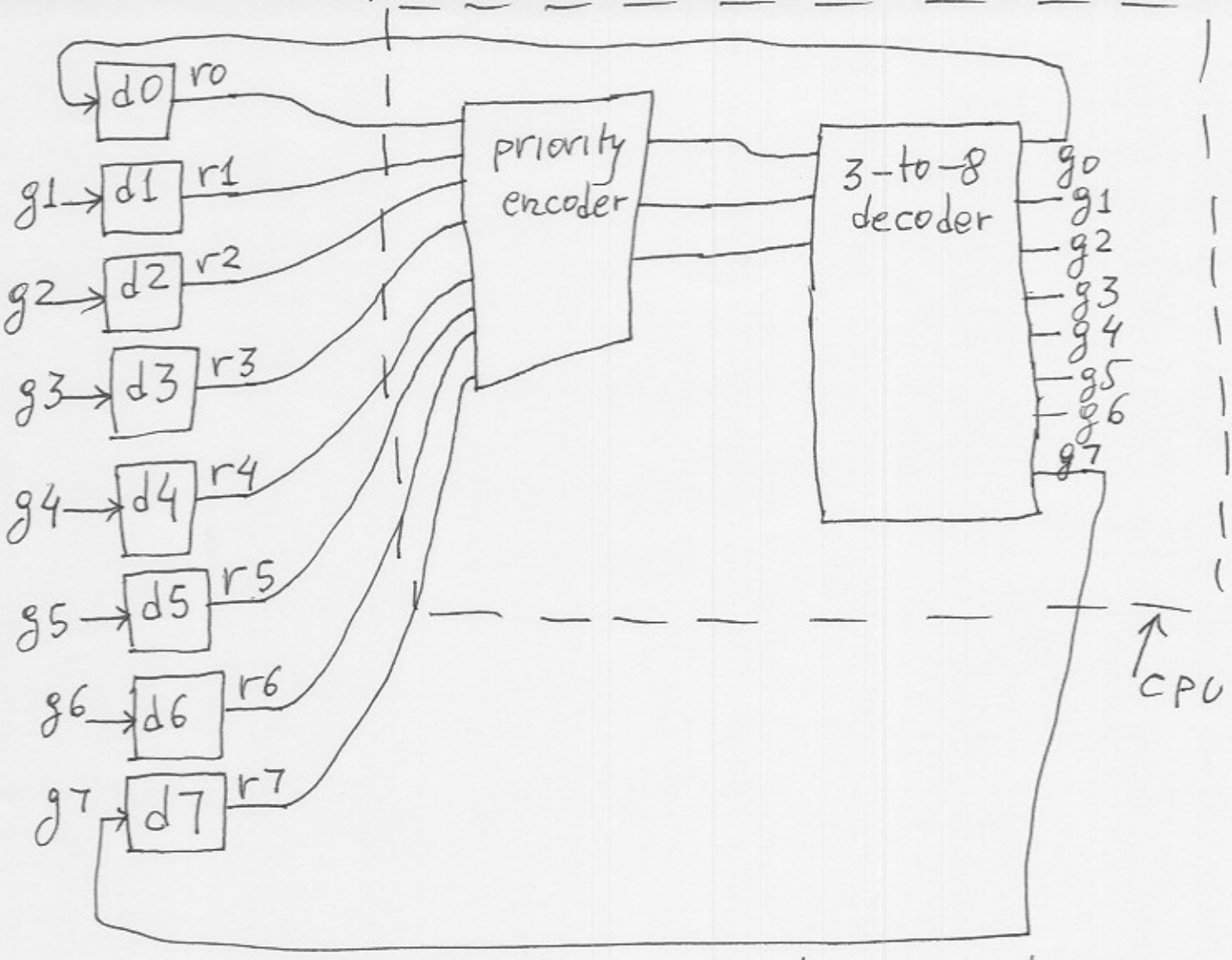

Fig 8: How to handle multiple requests in a priority system.

As you can see in the above figure 8, the signals $r_0, r_1, \ldots, r_7$ are encoded using a priority encoder. The three outputs of the priority encoder are inputs to a 3-to-8 decoder that generates the eight signals $g_0, g_1 \ldots g_7$. The only case that the above fig. 8 does not consider is the case where all r_i's are 0,

meaning that none of the devices d_i's requests service from the CPU. The way we can handle this case is the following: The priority encoder of fig. 8 will have one extra output; (call it a). The output a will be 1 if all inputs $r_0, r_1, \ldots, r_7$ to the priority encoder are 0. Obviously $a = r_0' \cdot r_1' \cdot r_2' \cdot r_3' \cdot r_4' \cdot r_5' \cdot r_6' \cdot r_7'$. This output a will be used as an input to the 3-to-8 decoder to disable it; (when $a = 1$, the 3-to-8 decoder is disabled). In this case, all the eight outputs of the 3-to-8 decoder namely $g_0, g_1, \ldots, g_7$ become 0, meaning that service is not granted to any of the devices $d_0, d_1, \ldots, d_7$.