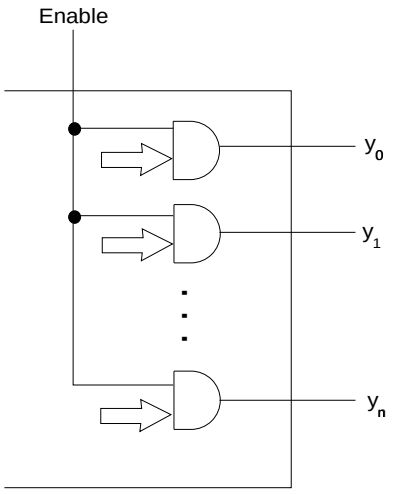
Lecture Notes for ENEE244, Fall 1998

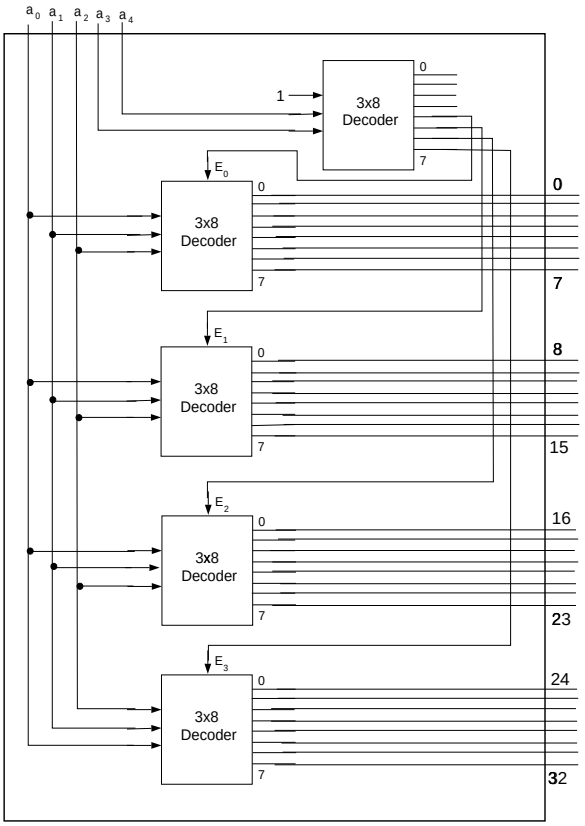
October 21, 1998

If you think back to the previous lecture you'll recall that we are trying to build a 5x32 decoder using only 3x8 decoders. We introduced the idea of an enable input on the 3x8 decoders that will allow us to do this. How would one actually implement the enable? For an active high decoder the inside is simply a bunch of AND gates which tie together the correct lines to produce the desired output. We can bring the enable line into each of these AND gates as well. If the enable is set to 0 all of the AND gates will be forced to 0. If the enable is set to 1 then the decoder will function as before. Thus the inside of the decoder will look something like:



So how are we going to build a 5x32 from the 3x8 blocks? We obviously need four of them to get the 32 output lines but we still need to figure out how

we are going to select when to enable a particular 3x8 decoder. If we run three of the five input lines in parallel to each of the four 3x8 decoders we are still left with two input lines. How many combinations can we get from the remaining two lines? That's easy, it's simply $2^2 = 4$. But that's exactly how many different decoders we have to enable, right? So how do we get from two inputs to four outputs? Sounds like a decoder... Let's see what the circuit looks like:



5 x 32 Decoder

The inputs a_3, a_4 are run into the enable decoder and select which of the lower decoders will be enabled. If they are (0,0) then the zeroeth decoder will be enabled, if (0,1) then the first will be enabled, and so on. The other three inputs, a_2, a_1, a_0 select which of the output lines on the enabled decoder is set to true. So if the input (01000) is placed on the input lines then the

zeroeth line of the first decoder will be enabled and all the other lines will be zero. Thus the eigth output line is turned on- just as we expected!

Generalizing...

So can we extend this idea? That is can we come up with an algorithm to build a decoder with α inputs and 2^α outputs from decoders with β inputs and 2^β outputs? We will assume $\alpha > \beta$ because otherwise there's no problem. First let's ask how many of the β decoders we need to get the output lines. In our previous example (a 5x32 from 3x8's) we divided 32 by 8 to find that we needed four. For the generic case, then, we need to divide 2^α by 2^β. That is:

$$\begin{aligned} \text{Number of } \beta \text{ decoders for output} &= \frac{2^\alpha}{2^\beta} \\ &= 2^{\alpha-\beta} \end{aligned}$$

Each of these decoders will need to be enabled so we need $2^{\alpha-\beta}$ enable lines which requires a decoder with $\ln(2^{\alpha-\beta}) = \alpha - \beta$ inputs. If $\alpha - \beta \leq \beta$ then we're ok and one more encoder will take care of it. If not then we're asking the same question again: how do we build an encoder with $\alpha - \beta$ inputs from encoders with β inputs? So we just need to repeat the process. If that encoder needs more enable lines than a single β encoder has then you repeat the process again. You keep repeating until you only need β encoders.

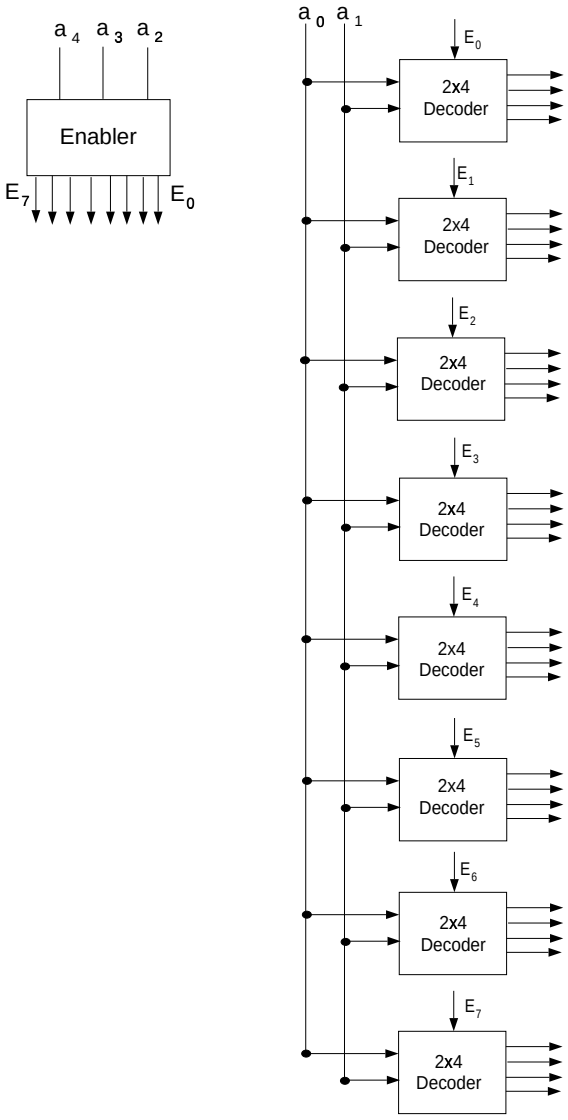
Let's look at an example. Let $\alpha = 5$ and $\beta = 2$. We need:

$$\begin{aligned} 2^{\alpha-\beta} &= 2^{5-2} \\ &= 8 \end{aligned}$$

of the 2x4 encoders to get the output lines. How many enablers do we need?

$$\alpha - \beta = 5 - 2 = 3 > 2$$

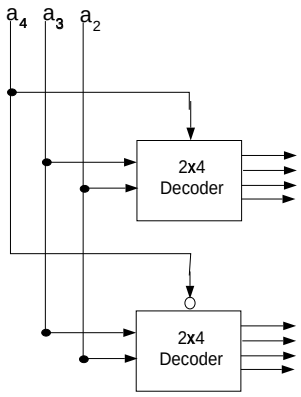
So we know we need more than one enabling decoder. The block diagram so far looks like:



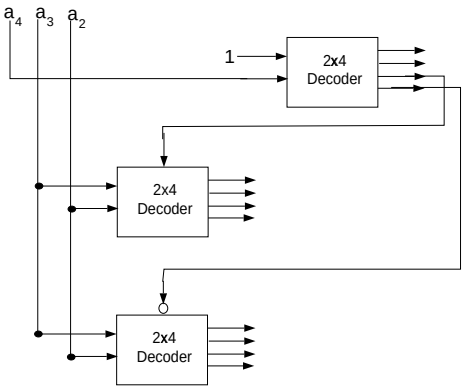
Now we need to build the enabler. We need:

$$\frac{2^{\alpha-\beta}}{2^{\beta}} \quad = \quad 2^1 = 2$$

of the 2x4 decoders to build it. The enabler will look like:



If you don't like the fact that we added a bubble to the enable input of one of the decoders then you could simply add one more encoder to get rid of it. The enabler would then look like:



Summary

If you're given a bag of β by 2^β decoders and you need to design an α by 2^α deocder:

- If $\alpha \leq \beta$ then use a single βx2^β
- Else use $2^{\alpha-\beta}$ of the βx2^β deocders and one $(\alpha-\beta)$x$2^{\alpha-\beta}$ decoder for enabling.

To implement the enabler go through the process again.