TutorTube: Multiplexers and Decoders

Spring 2021

Introduction

Hello and welcome to TutorTube, where The Learning Center's Lead Tutors help you understand challenging course concepts with easy to understand videos. My name is Jeff, Lead Tutor for Electrical Engineering. In today's video, we will explore analyzing sequential circuits. Let's get started!

Define our Devices

So first off, what are multiplexers and decoders? Multiplexers, or Mux for short, and decoders are modular logic devices, meaning they can be integrated into a circuit whenever you need them. Their functions are very distinct from each other, but it's easy to get the two confused since they look similar, and they can both be used in similar places. However, both of these devices are used to filter or process multiple inputs.

Multiplexers

Multiplexers receive multiple different logical inputs, but only one is sent to the final output. Think about it like a bouncer at a door who decides who goes through the door at a time. Another example would be multiple computers connected to a printer. The printer can only print from one computer at a time, otherwise documents could get mixed up. So, the mux helps the printer decide which computer to listen to.

The multiplexer is controlled by a second set of inputs called the select line. The multiplexer interprets the number sent across the select line and sends the corresponding input to the output. This can be shown by this truth table. If the select line spells 0, then input 0 is passed. If the select line spells 1 then input 1 is passed, and so on. From this table you could even come up with a Boolean expression for the multiplexer, which will be helpful when solving circuits that use a multiplexer. The expression is just every minterm ANDed with the corresponding input. It's a very generic Sum of Product expression that we've seen before.

S1	S0	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

Table 1 - 4x1 Multiplexer Truth Table

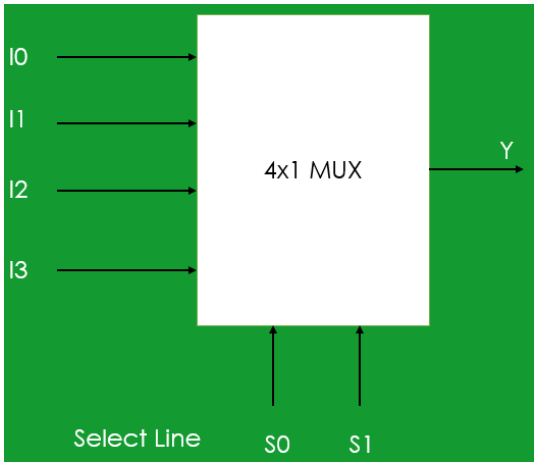

Figure 1 - 4x1 Multiplexer

Y = m0*I0 + m1*I1 + m2*I2 + m3*I3

Decoders

The decoder truth table isn't so straight forward so I won't list the whole thing out. What's more important is knowing its function qualitatively. The decoder reads the inputs as a single binary number, with A, B, C etc. making up the digits. Whatever that number spells corresponds to one of the outputs. That output is set to high while all others are set to low.

However, we do still need some way to relate this to a Boolean equation without a truth table, and that's by setting each output equivalent to the same numbered minterm. This table shows what that looks like. So, input 0 is equivalent to minterm 0, input 1 to minterm 1, etc.

Y	Minterm	Binary Equivalent
Y0	A'B'C'	000
Y1	A'B'C	001
Y2	A'BC'	010
Y3	A'BC	011
Y4	AB'C'	100
Y5	AB'C	101
Y6	ABC'	110
Y7	ABC	111

Table 2 - 3x8 Decoder Output List

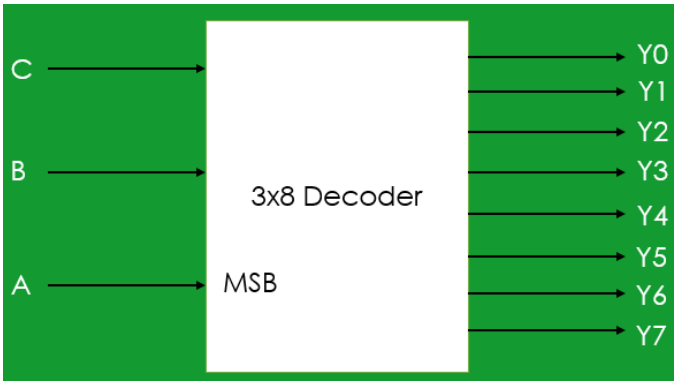

Figure 2 - 3x8 Decoder

Equation Overview

So, what equations do you need to know? The first is how to relate a multiplexer to its inputs. This is your more traditional Boolean function with inputs and outputs. The decoder is broken up into its minterms where each output corresponds to only one minterm. This will become clearer once we do some example problems.

It's also worth noting that these formulas are not limited by dimension sizes. If you have a 64 by 1 multiplexer, the equation follows the same pattern that a 4 by 1 does.

- **Multiplexer (4x1)**
- ***Y = (S0'*S1')I0 + (S0'*S1)I1 + (S0*S1')I2 + (S0*S1)I3***
- ***Y = m0*I0 + m1*I1 + m2*I2 + m3*I3***

- **Decoder (3x8)**
- ***Y1 = A'B'C', Y2 = A'B'C, Y3 = A'BC',…,Y8 = ABC***

Example 1

Let's take a look at this example circuit. It's a 4 by 1 mux with some gates preceding the inputs to the mux. We're asked to find the minterms of the function created by the circuit. In other words, what combinations of A, B, and C, will make Y a 1?

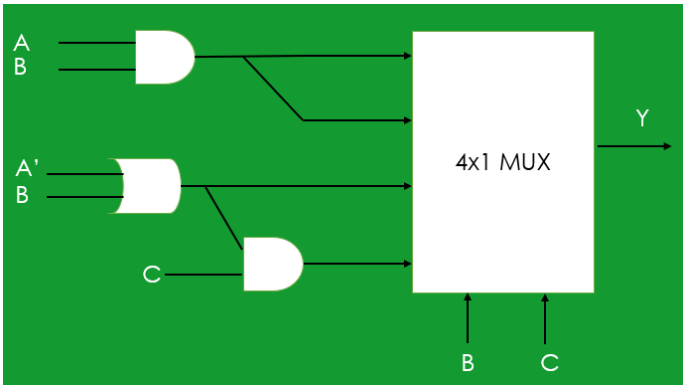

Figure 3 - Example 1 Circuit

Let's work our way backwards, starting from the end. We know from the multiplexer equation that the output Y is equal to the sum of products of the inputs and the select line minterms. Right now, we just have the inputs written as I0, I1, etc. so let's substitute those out. I0 and I1 are the same: A AND B. I2 is A' OR B. I3 takes I2 and ANDs it with C giving (A'+ B)C.

$$Y = (B'C')I0 + (B'C)I1 + (BC')I2 + (BC)I3$$

$$Y = (B'C')AB + (B'C)AB + (BC')(A'+B) + (BC)(A'+B)C$$

Now let's apply distributive law to clean this equation up. We'll end up with a lot of repeated variables but using our Boolean postulates we know that repeated letters just vanish, and anything ANDed with its complement becomes 0. Now anything ANDed with 0 is also 0 which means 3 of our terms simply vanish.

$$Y = (B'C')AB + (B'C)AB + (BC')(A'+B) + (BC)(A'+B)C$$

$$Y = ABB'C' + ABB'C + A'BC' + BB'C + A'BCC + BBCC$$

$$Y = A(0)C' + A(0)C + A'BC' + (0)C + A'BC + BC$$

$$Y = 0 + 0 + A'BC' + 0 + A'BC + BC$$

$$Y = A'BC' + A'BC + BC$$

Now we almost have everything in canonical form, but the last term BC is missing an A value. So, we're going to AND it with 1 and then substitute that 1 with A' + A. You can verify with your Boolean postulates that I haven't actually changed the value of the final expression, just how it appears. In doing so, I see I have a duplicate minterm, which I can ignore. And finally, we can rewrite those minterms as a summation, and our problem is done!

$$Y = A'BC' + A'BC + BC = A'BC' + A'BC + BC(1)$$

$$Y = A'BC' + A'BC + BC(A + A')$$

$$Y = A'BC' + A'BC + ABC + A'BC$$

$$Y = \sum m(2,3,7)$$

Example 2

Let's look at another example. We have the same goal as before, but now we're dealing with a decoder.

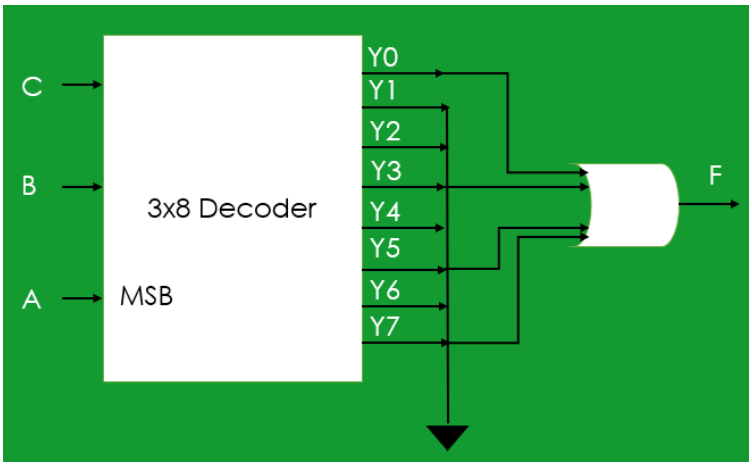

Figure 4 - Example 2 Circuit

We can notice that some of the outputs have been grounded and don't contribute to the final output, so we'll ignore them. What we can see is that the output is the result of four of the outputs OR'd together. Since we know that each output is equivalent to a single corresponding minterm, this expression is really just the canonical form of minterms 0, 3, 5 and 7.

$$F = Y0 + Y3 + Y5 + Y7$$

$$F = m0 + m3 + m5 + m7$$

$$F = \sum m(0,3,5,7)$$

$$F = A'B'C' + A'BC + AB'C + ABC$$

This arrangement: connecting outputs of a decoder to an OR gate, is a very simple way to express any logic function, though it's probably not as efficient as it could be.

Example 3

This one is a design question. We're given the function that we have to express but because of some constraint, we have to use a multiplexer, specifically an 8 by 1.

$$F(A,B,C,D) = m(0,1,3,5,7,10,11,14)$$

Let's start by making the assumption that the inputs, A, B, and C, go directly into the select lines of the mux, so we just need to figure out how D is going to play a role here.

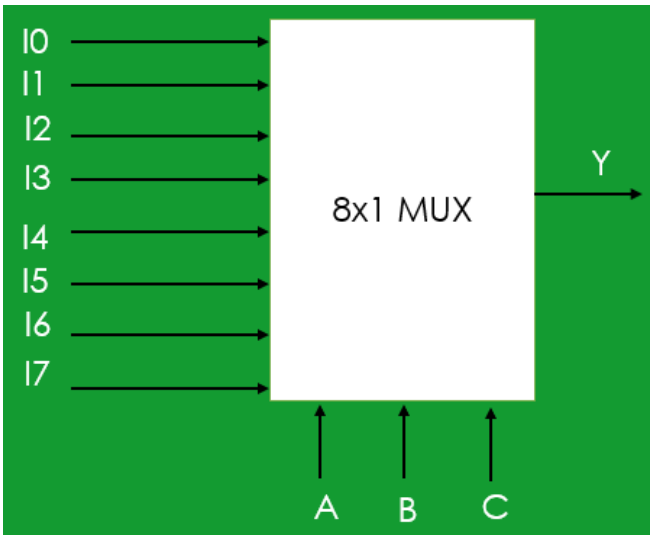

Figure 5 - 8x1 Multiplexer

I've drawn out the truth table for the entire function that we are asked to implement. The way we can analyze this, is recognize that every 2 rows, have the same combination of A, B, and C, meaning the mux will pull from the same input both times. So, the question becomes: for this combination of select lines, what should we make the input be, depending on what D is? It's a question that makes more sense once you see it done a few times.

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

Figure 6 - Example 3 Truth Table

Let's look at the first two rows, where the select inputs are all 0, corresponding to input 0. We can see that the output for both rows is 1, even though D is different. So, the output has no dependency on D for this input, so we can just set this input directly to 1, like I have on the drawing.

Let's move to the second pair, rows 3 and 4. Notice how the output and D match? This means the output is equal to D for this select combination, so the input becomes D. The third set, and also the fourth follow the same pattern and are equal to D.

The fifth set have both outputs as zeros, so there's no dependency on D, and so we can set the input to 0. And for the same reason, the sixth set is 1 regardless of D, so the input is 1.

The seventh set is just like the fifth and will have its input set to 0. And lastly, we can see that for the last two rows, the output is the complement of D, meaning this mux input is going to be D'.

Now if we redraw the circuit, connecting all the inputs that should be one to some power source, and all the inputs that should be 0 to ground, we've created our circuit that follows the given function by using an 8 by 1 mux!

This process works for other dimensions as well, and you don't have to pick A, B, and C as your select inputs. That was an arbitrary choice that just made the truth table a little easier to analyze. You could've used B, C, and D as your select lines too if you wanted, but it wouldn't look like this. There would be some other combination of 1s, 0s, and A values that make up your inputs to the mux. In these design questions you often have a little freedom in how you go about naming and setting up your design.

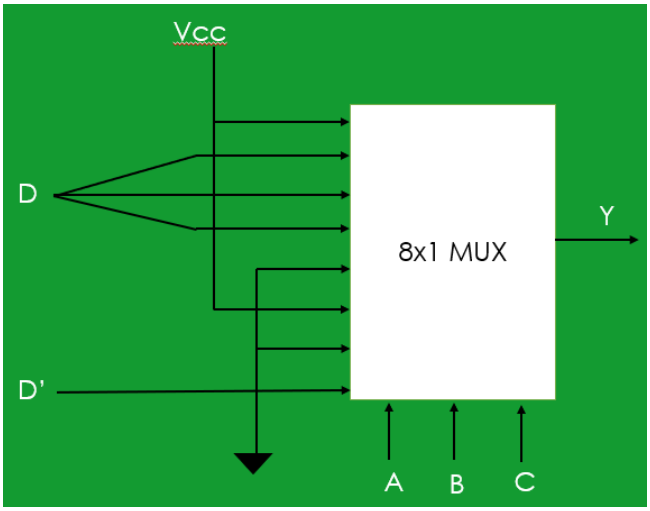

Figure 7 - Example 3 Final Circuit

Outro

Thank you for watching TutorTube! I hope you enjoyed this video. Please subscribe to our channel for more exciting videos. Check out the links in the description below for more information about The Learning Center and follow us on social media. See you next time!