ECEN 429: Introduction to Digital Systems Design Laboratory

North Carolina Agricultural and Technical State University

Department of Electrical and Computer Engineering

Freddie Boadu (Reporter)

February 6, 2020

Pre Lab #3

**Introduction**

Full adders, and muxes are used in a lot of digital system. The full adder which we covered last week are great for arithmetic and the muxes is more known as a data selector. We learned about the power of full adders last week and that they can conduct carry in and carry outs and sums. The muxes are a new concept this week. The Mux has a selector on it. And a number of inputs, the selector will help determine if there is high impedance, zero, or one.

**Background, Design Solution and Results**

This prelab helps the student to further understand muxes and full adders and how they can all be connected in terms of digital systems.

Problem 1:

The first problem asks for a truth table of a full adder that accepts two 2-bit inputs A and B. This would mean that A and B would both be two bit. They will be represented with A0-A1 and B0-B1. The truth table would look like this:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A0 | A1 | B0 | B1 | CI | S0 | S1 | C0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Table 1.1 This is a truth table of a full adder with two two-bit inputs. And the outputs of CO and two sums are needed.

Problem 2:

The second problem of the lab deals with a 2:1 mux. The truth table for this would be:

|  |  |  |  |
| --- | --- | --- | --- |
| A(i0) | B(i1) | S | Z |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 |

Table 2.1 This is a table of the 2-1 mux. With two inputs (A and B), and select bit. And the Z would be the output of everything.

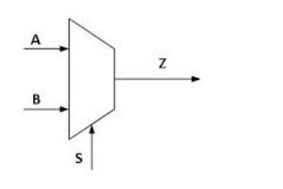


Figure 2.1 2-1 mux with 2 inputs 1 select and 1 output.

The VHDL expression for Z is Z<=(A\*X’)+(B\*X).

For the second part of this question we are asked to make a 4:1 mux. We were asked to draw a diagram for it. This is the diagram:

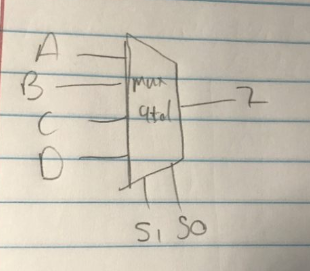


Figure 2.2 A 4:2 mux with 4 inputs and 2 selections and a 1 output.

The code for the 4:1 mux would be:

Entity fourtoone is

Port(A,B,C,D,S1,S0: in std\_logic;

Z: out std\_logic);

End;

Architecture bev of fourtoone is

begin

process(A,B,C,D,S1,S0) is

begin

|  |
| --- |
| (S0 ='0' and S1 = '0') then |

|  |
| --- |
| Z <= A; |

|  |
| --- |
| elsif (S0 ='1' and S1 = '0') then |

|  |
| --- |
| Z <= B; |

|  |
| --- |
| elsif (S0 ='0' and S1 = '1') then |

|  |
| --- |
| Z <= C; |

|  |
| --- |
| else |

|  |
| --- |
| Z <= D; |

|  |
| --- |
| end if; |

End ;

End;

Problem 3

For problem 3, we are looking a muxs and full adder together, with a seven segment display driver. Each light blue arrow is 1 bit from looking at the diagram. The BCD, sum, cout, A, B will all be one bit. The sel will be 2bits. The display will be around 7 bits to represent the type of numbers and letters created on a seven segment display. The 4-1 mux is only taking in 3 inputs. The 4 input that would have outputs would become a down care. So from the code that was previously done. We can just make the output dealing with the fourth input ZZ or a don’t care like: The code for the 4:1 mux would be:

Entity fourtoone is

Port(A,B,C,D,S1,S0: in std\_logic;

Z: out std\_logic);

End;

Architecture bev of fourtoone is

begin

process(A,B,C,D,S1,S0) is

begin

|  |
| --- |
| (S0 ='0' and S1 = '0') then |

|  |
| --- |
| Z <= A; |

|  |
| --- |
| elsif (S0 ='1' and S1 = '0') then |

|  |
| --- |
| Z <= B; |

|  |
| --- |
| elsif (S0 ='0' and S1 = '1') then |

|  |
| --- |
| Z <= C; |

|  |
| --- |
| else |

|  |
| --- |
| Z <= “zz”; |

|  |
| --- |
| end if; |

End ;

End;

The components will be declared like:

Component FA is

Port(a,b,sum: in std\_logic

Sum, co: out std\_logic)

End component;

Component MUX is

Port(sum, sum, cout, sel: in std\_logic

BCDnumber:out std\_logic)

End component;

Component seven\_segment is

Port(bcdnumber: in std\_logic;

Display: out std\_logic)

End component;

**Conclusion**

After doing the prelab, I understand how full adders and muxes work a lot better. I also understand how component play a part into Vhdl and how they can make the problems that we encounter a lot easier to deal with and break up.