Quinn Roemer

Engineering – 303

Lab 3

2/10/2017

Introduction/Description

The purpose of this lab was to learn how to design and build slightly more complex circuits by creating the algebraic expressions yourself using truth tables and Kmaps. I was supposed to learn how build Kmaps, use those Kmaps to build algebraic circuit expressions, and implement them as actual circuits in Quartus.

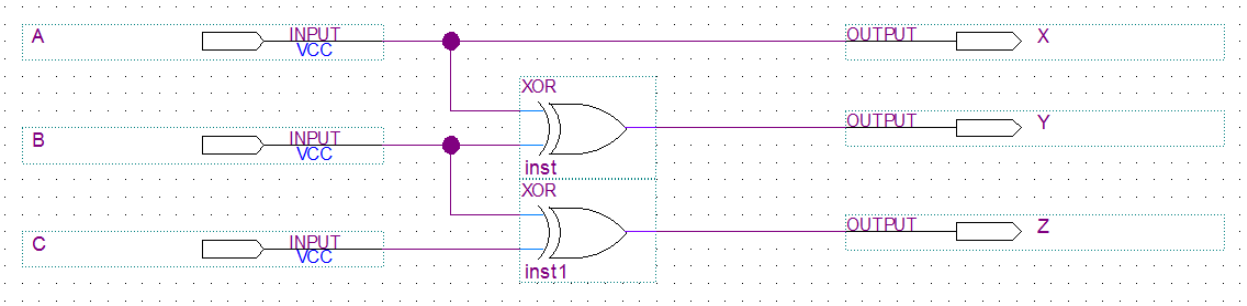
Design

In the first part of the lab I was supposed to follow the given instructions to create a circuit that would convert a three byte binary number into its equivalent three byte grey code number. I was supposed to implement this circuit as a Block-Diagram and a Verilog Program.

Here is the Verilog program for the circuit.

```
1   //A circuit that converts a three byte binary number into a
2   //three byte grey code number.
3   module BinaryGreyVerilog (a,b,c,x,y,z);
4
5   input a,b,c;
6   output x,y,z;
7
8   assign x = a;
9   assign y = a ^ b;
10  assign z = b ^ c;
11
12  endmodule
```

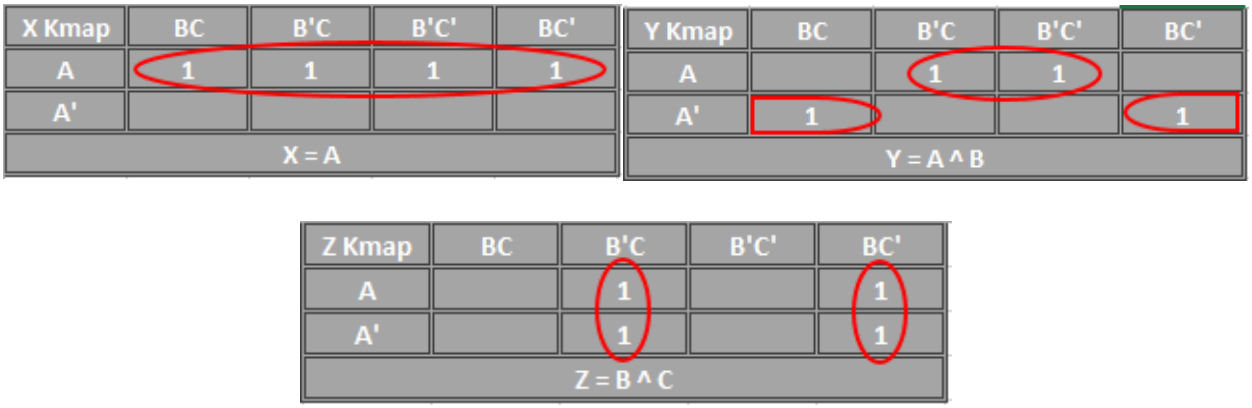
Here is the Block-Diagram for the binary to grey code converter circuit.



Here is the truth table that was given to me in the lab to create the circuit.

Inputs			Outputs			Minterms		
Binary			Grey					
A	B	C	X	Y	Z	mX	mY	mZ
0	0	0	0	0	0			
0	0	1	0	0	1			A'B'C
0	1	0	0	1	1		A'BC'	A'BC'
0	1	1	0	1	0		A'BC	
1	0	0	1	1	0	AB'C'	AB'C'	
1	0	1	1	1	1	AB'C	AB'C	AB'C
1	1	0	1	0	1	ABC'		ABC'
1	1	1	1	0	0	ABC		

Here are the Kmaps that I was given in the lab to create the circuit.



In the second part of the lab I was supposed to create a circuit called a Constant Adder. This circuit was designed to count up in binary starting from three all the way to ten. Unlike the previous circuit I had to create the truth table, kmaps, and circuit without being given them.

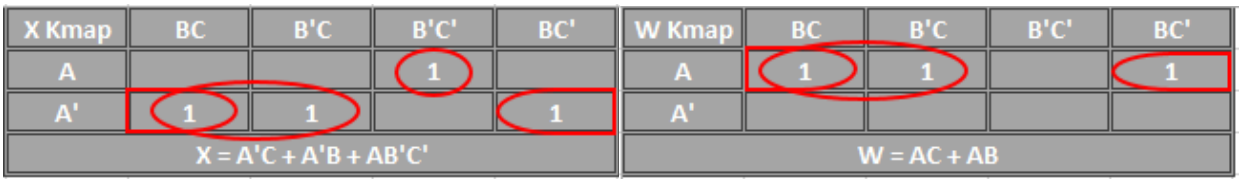
Here is the Verilog design that I came up with.

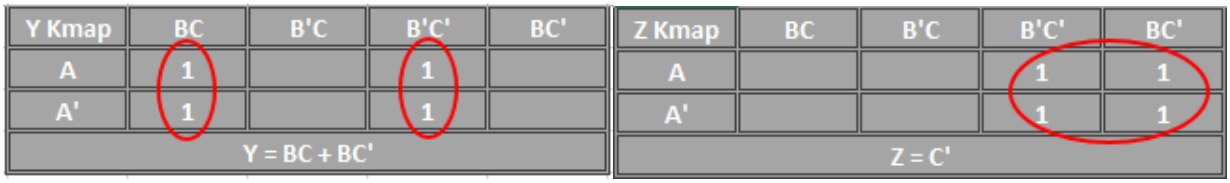
```
1   //A constant adder circuit.
2   module ConstantAdder (a,b,c,w,x,y,z);
3
4   input a,b,c;
5   output w,x,y,z;
6
7   assign w = a & c | a & b;
8   assign x = ~a & c | ~a & b | a & ~b & ~c;
9   assign y = b & c | ~b & ~c;
10  assign z = ~c;
11
12  endmodule
```

Here is the truth table filled with all the desired outputs.

Inputs			Outputs				Minterms			
A	B	C	W	X	Y	Z	mW	mX	mY	mZ
0	0	0	0	0	1	1			A'B'C'	A'B'C'
0	0	1	0	1	0	0		A'B'C		
0	1	0	0	1	0	1		A'BC'		A'BC'
0	1	1	0	1	1	0		A'BC	A'BC	
1	0	0	0	1	1	1		AB'C'	AB'C'	AB'C'
1	0	1	1	0	0	0	AB'C			
1	1	0	1	0	0	1	ABC'			ABC'
1	1	1	1	0	1	0	ABC		ABC	

Here are the Kmaps that I created to figure out the algebraic expressions for the circuit.





In the next part of the lab I was instructed to design a circuit with random outputs. This circuit was called an Arbitrary Combo circuit. I was given a truth table filled with all of the desired outputs and was told to create a circuit that would perform in such a manner.

Here is the finished Verilog design for the circuit.

```
1   //An arbitrary circuit.
2   module ArbitraryCombo (a,b,c,d,f1,f2,f3,f4);
3
4   input a,b,c,d;
5   output f1,f2,f3,f4;
6
7   assign f1 = ~a & (d | ~b | c & ~d) | ~b & ~c & ~d;
8   assign f2 = b & c & d | ~a & ~c & d | a & ~b & c | a & c & ~d | ~a & b & ~c;
9   assign f3 = a & b & d | a & b & ~c | ~b & ~c & ~d | a & ~b & ~d | ~a & ~b & c & d | ~a & b & c & ~d;
10  assign f4 = ~a & b & ~d | ~a & ~b & ~c | ~b & ~c & ~d | a & b & ~c & d;
11
12  endmodule
```

Here is the truth table that I was given to design the circuit in question.

Inputs				Outputs				Minterms			
A	B	C	D	F1	F2	F3	F4	mF1	mF2	mF3	mF4
0	0	0	0	1	0	1	1	A'B'C'D'		A'B'C'D'	A'B'C'D'
0	0	0	1	1	1	0	1	A'B'C'D	A'B'C'D		A'B'C'D
0	0	1	0	1	0	0	0	A'B'CD'			
0	0	1	1	1	0	1	0	A'B'CD		A'B'CD	
0	1	0	0	0	1	0	1		A'BC'D'		A'BC'D'
0	1	0	1	1	1	0	0	A'BC'D	A'BC'D		
0	1	1	0	1	0	1	1	A'BCD'		A'BCD'	A'BCD'
0	1	1	1	1	1	0	0	A'BCD	A'BCD		
1	0	0	0	1	0	1	1	AB'C'D'		AB'C'D'	AB'C'D'
1	0	0	1	0	0	0	0				
1	0	1	0	0	1	1	0		AB'CD'	AB'CD'	
1	0	1	1	0	1	0	0		AB'CD		
1	1	0	0	0	0	1	0			ABC'D'	
1	1	0	1	0	0	1	1			ABC'D	ABC'D
1	1	1	0	0	1	0	0		ABCD'		
1	1	1	1	0	1	1	0		ABCD	ABCD	

Here are the Kmaps that I created to find the algebraic expressions that I would use in the circuit's design.

F1 Kmap	CD	C'D	C'D'	CD'
AB				
A'B	1	1		1
A'B'	1	1	1	1
AB'			1	
F1 = A'(D+B'+CD')+B'C'D'				

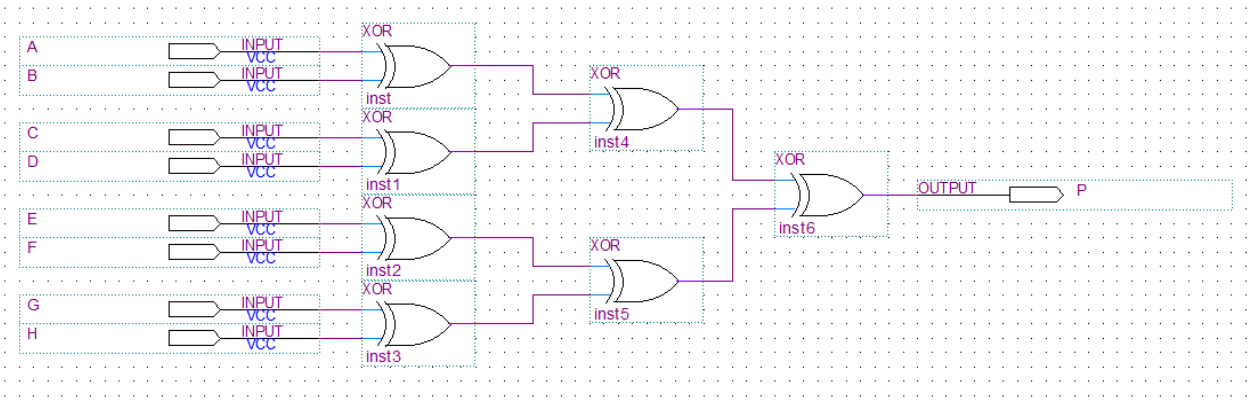
F2 Kmap	CD	C'D	C'D'	CD'
AB	1			1
A'B	1	1	1	
A'B'		1		
AB'	1			1
F2 = BCD+A'C'D+AB'C+ACD'+A'BC'				

F3 Kmap	CD	C'D	C'D'	CD'
AB	1	1	1	
A'B				1
A'B'	1		1	
AB'			1	1
F3 = ABD + ABC'+B'C'D'+AB'D'+A'B'CD+ A'BCD'				

F4 Kmap	CD	C'D	C'D'	CD'
AB		1		
A'B			1	1
A'B'		1	1	
AB'			1	
F4 = A'BD'+A'B'C'+B'C'D'+ABC'D				

In the last part of the lab I was given a design for an odd parity bit generator circuit. I was instructed to implement the Block-Diagram design I was given in Quartus. In addition I was supposed to create a large truth table that would encompass all the possible inputs and outputs of the circuit.

Here is the Block-Diagram design.



Here is the truth that encompasses all 255 possible inputs.

Inputs								Outputs
A	B	C	D	E	F	G	H	P
0	0	0	0	0	0	0	0	FALSE
0	0	0	0	0	0	0	1	TRUE
0	0	0	0	0	0	1	0	TRUE
0	0	0	0	0	0	1	1	FALSE
0	0	0	0	0	1	0	0	TRUE
0	0	0	0	0	1	0	1	FALSE
0	0	0	0	0	1	1	0	FALSE
0	0	0	0	0	1	1	1	TRUE
0	0	0	0	1	0	0	0	TRUE
0	0	0	0	1	0	0	1	FALSE
0	0	0	0	1	0	1	0	FALSE
0	0	0	0	1	0	1	1	TRUE
0	0	0	0	1	1	0	0	FALSE
0	0	0	0	1	1	0	1	TRUE
0	0	0	0	1	1	1	0	TRUE
0	0	0	0	1	1	1	1	FALSE
0	0	0	1	0	0	0	0	TRUE
0	0	0	1	0	0	0	1	FALSE
0	0	0	1	0	0	1	0	FALSE
0	0	0	1	0	0	1	1	TRUE
0	0	0	1	0	1	0	0	FALSE
0	0	0	1	0	1	0	1	TRUE
0	0	0	1	0	1	1	0	TRUE
0	0	0	1	0	1	1	1	FALSE
0	0	0	1	1	0	0	0	FALSE
0	0	0	1	1	0	0	1	TRUE
0	0	0	1	1	0	1	0	TRUE
0	0	0	1	1	0	1	1	FALSE
0	0	0	1	1	1	0	0	TRUE
0	0	0	1	1	1	0	1	FALSE
0	0	0	1	1	1	1	0	FALSE
0	0	0	1	1	1	1	1	TRUE
0	0	1	0	0	0	0	0	TRUE
0	0	1	0	0	0	0	1	FALSE
0	0	1	0	0	0	1	0	FALSE
0	0	1	0	0	0	1	1	TRUE
0	0	1	0	0	1	0	0	FALSE
0	0	1	0	0	1	0	1	TRUE
0	0	1	0	0	1	1	0	TRUE
0	0	1	0	0	1	1	1	FALSE
0	0	1	0	1	0	0	0	FALSE
0	0	1	0	1	0	0	1	TRUE

A

0	0	1	0	1	0	1	0	TRUE
0	0	1	0	1	0	1	1	FALSE
0	0	1	0	1	1	0	0	TRUE
0	0	1	0	1	1	0	1	FALSE
0	0	1	0	1	1	1	0	FALSE
0	0	1	0	1	1	1	1	TRUE
0	0	1	1	0	0	0	0	FALSE
0	0	1	1	0	0	0	1	TRUE
0	0	1	1	0	0	1	0	TRUE
0	0	1	1	0	0	1	1	FALSE
0	0	1	1	0	1	0	0	TRUE
0	0	1	1	0	1	0	1	FALSE
0	0	1	1	0	1	1	0	FALSE
0	0	1	1	0	1	1	1	TRUE
0	0	1	1	1	0	0	0	TRUE
0	0	1	1	1	0	0	1	FALSE
0	0	1	1	1	0	1	0	FALSE
0	0	1	1	1	0	1	1	TRUE
0	0	1	1	1	1	0	0	FALSE
0	0	1	1	1	1	0	1	TRUE
0	0	1	1	1	1	1	0	TRUE
0	0	1	1	1	1	1	1	FALSE
0	1	0	0	0	0	0	0	TRUE
0	1	0	0	0	0	0	1	FALSE
0	1	0	0	0	0	1	0	FALSE
0	1	0	0	0	0	1	1	TRUE
0	1	0	0	0	1	0	0	FALSE
0	1	0	0	0	1	0	1	TRUE
0	1	0	0	0	1	1	0	TRUE
0	1	0	0	0	1	1	1	FALSE
0	1	0	0	1	0	0	0	FALSE
0	1	0	0	1	0	0	1	TRUE
0	1	0	0	1	0	1	0	TRUE
0	1	0	0	1	0	1	1	FALSE
0	1	0	0	1	1	0	0	TRUE
0	1	0	0	1	1	0	1	FALSE
0	1	0	0	1	1	1	0	FALSE
0	1	0	0	1	1	1	1	TRUE
0	1	0	1	0	0	0	0	FALSE
0	1	0	1	0	0	0	1	TRUE
0	1	0	1	0	0	1	0	TRUE
0	1	0	1	0	0	1	1	FALSE
0	1	0	1	0	1	0	0	TRUE
0	1	0	1	0	1	0	1	FALSE

A

B

B

‘0	1	0	1	0	1	1	0	FALSE
0	1	0	1	0	1	1	1	TRUE
0	1	0	1	1	0	0	0	TRUE
0	1	0	1	1	0	0	1	FALSE
0	1	0	1	1	0	1	0	FALSE
0	1	0	1	1	0	1	1	TRUE
0	1	0	1	1	1	0	0	FALSE
0	1	0	1	1	1	0	1	TRUE
0	1	0	1	1	1	1	0	TRUE
0	1	0	1	1	1	1	1	FALSE
0	1	1	0	0	0	0	0	FALSE
0	1	1	0	0	0	0	1	TRUE
0	1	1	0	0	0	1	0	TRUE
0	1	1	0	0	0	1	1	FALSE
0	1	1	0	0	1	0	0	TRUE
0	1	1	0	0	1	0	1	FALSE
0	1	1	0	0	1	1	0	FALSE
0	1	1	0	0	1	1	1	TRUE
0	1	1	0	1	0	0	0	TRUE
0	1	1	0	1	0	0	1	FALSE
0	1	1	0	1	0	1	0	FALSE
0	1	1	0	1	0	1	1	TRUE
0	1	1	0	1	1	0	0	FALSE
0	1	1	0	1	1	0	1	TRUE
0	1	1	0	1	1	1	0	TRUE
0	1	1	0	1	1	1	1	FALSE
0	1	1	1	0	0	0	0	TRUE
0	1	1	1	0	0	0	1	FALSE
0	1	1	1	0	0	1	0	FALSE
0	1	1	1	0	0	1	1	TRUE
0	1	1	1	0	1	0	0	FALSE
0	1	1	1	0	1	0	1	TRUE
0	1	1	1	0	1	1	0	TRUE
0	1	1	1	0	1	1	1	FALSE
0	1	1	1	1	0	0	0	FALSE
0	1	1	1	1	0	0	1	TRUE
0	1	1	1	1	0	1	0	TRUE
0	1	1	1	1	0	1	1	FALSE
0	1	1	1	1	1	0	0	TRUE
0	1	1	1	1	1	0	1	FALSE
0	1	1	1	1	1	1	0	FALSE
0	1	1	1	1	1	1	1	TRUE
1	0	0	0	0	0	0	0	TRUE
1	0	0	0	0	0	0	1	FALSE

C

1	0	0	0	0	0	1	0	FALSE
1	0	0	0	0	0	1	1	TRUE
1	0	0	0	0	1	0	0	FALSE
1	0	0	0	0	1	0	1	TRUE
1	0	0	0	0	1	1	0	TRUE
1	0	0	0	0	1	1	1	FALSE
1	0	0	0	1	0	0	0	FALSE
1	0	0	0	1	0	0	1	TRUE
1	0	0	0	1	0	1	0	TRUE
1	0	0	0	1	0	1	1	FALSE
1	0	0	0	1	1	0	0	TRUE
1	0	0	0	1	1	0	1	FALSE
1	0	0	0	1	1	1	0	FALSE
1	0	0	0	1	1	1	1	TRUE
1	0	0	1	0	0	0	0	FALSE
1	0	0	1	0	0	0	1	TRUE
1	0	0	1	0	0	1	0	TRUE
1	0	0	1	0	0	1	1	FALSE
1	0	0	1	0	1	0	0	TRUE
1	0	0	1	0	1	0	1	FALSE
1	0	0	1	0	1	1	0	FALSE
1	0	0	1	0	1	1	1	TRUE
1	0	0	1	1	0	0	0	TRUE
1	0	0	1	1	0	0	1	FALSE
1	0	0	1	1	0	1	0	FALSE
1	0	0	1	1	0	1	1	TRUE
1	0	0	1	1	1	0	0	FALSE
1	0	0	1	1	1	0	1	TRUE
1	0	0	1	1	1	1	0	TRUE
1	0	0	1	1	1	1	1	FALSE
1	0	1	0	0	0	0	0	FALSE
1	0	1	0	0	0	0	1	TRUE
1	0	1	0	0	0	1	0	TRUE
1	0	1	0	0	0	1	1	FALSE
1	0	1	0	0	1	0	0	TRUE
1	0	1	0	0	1	0	1	FALSE
1	0	1	0	0	1	1	0	FALSE
1	0	1	0	0	1	1	1	TRUE
1	0	1	0	1	0	0	0	TRUE
1	0	1	0	1	0	0	1	FALSE
1	0	1	0	1	0	1	0	FALSE
1	0	1	0	1	0	1	1	TRUE
1	0	1	0	1	1	0	0	FALSE
1	0	1	0	1	1	0	1	TRUE

C

D

D

1	0	1	0	1	1	1	0	TRUE
1	0	1	0	1	1	1	1	FALSE
1	0	1	1	0	0	0	0	TRUE
1	0	1	1	0	0	0	1	FALSE
1	0	1	1	0	0	1	0	FALSE
1	0	1	1	0	0	1	1	TRUE
1	0	1	1	0	1	0	0	FALSE
1	0	1	1	0	1	0	1	TRUE
1	0	1	1	0	1	1	0	TRUE
1	0	1	1	0	1	1	1	FALSE
1	0	1	1	1	0	0	0	FALSE
1	0	1	1	1	0	0	1	TRUE
1	0	1	1	1	0	1	0	TRUE
1	0	1	1	1	0	1	1	FALSE
1	0	1	1	1	1	0	0	TRUE
1	0	1	1	1	1	0	1	FALSE
1	0	1	1	1	1	1	0	FALSE
1	0	1	1	1	1	1	1	TRUE
1	1	0	0	0	0	0	0	FALSE
1	1	0	0	0	0	0	1	TRUE
1	1	0	0	0	0	1	0	TRUE
1	1	0	0	0	0	1	1	FALSE
1	1	0	0	0	1	0	0	TRUE
1	1	0	0	0	1	0	1	FALSE
1	1	0	0	0	1	1	0	FALSE
1	1	0	0	0	1	1	1	TRUE
1	1	0	0	1	0	0	0	TRUE
1	1	0	0	1	0	0	1	FALSE
1	1	0	0	1	0	1	0	FALSE
1	1	0	0	1	0	1	1	TRUE
1	1	0	0	1	1	0	0	FALSE
1	1	0	0	1	1	0	1	TRUE
1	1	0	0	1	1	1	0	TRUE
1	1	0	0	1	1	1	1	FALSE
1	1	0	1	0	0	0	0	TRUE
1	1	0	1	0	0	0	1	FALSE
1	1	0	1	0	0	1	0	FALSE
1	1	0	1	0	0	1	1	TRUE
1	1	0	1	0	1	0	0	FALSE
1	1	0	1	0	1	0	1	TRUE
1	1	0	1	0	1	1	0	TRUE

E

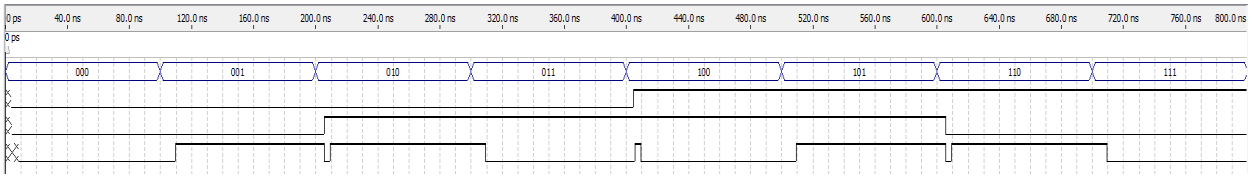
1	1	0	1	0	1	1	1	FALSE
1	1	0	1	1	0	0	0	FALSE
1	1	0	1	1	0	0	1	TRUE
1	1	0	1	1	0	1	0	TRUE
1	1	0	1	1	0	1	1	FALSE
1	1	0	1	1	1	0	0	TRUE
1	1	0	1	1	1	0	1	FALSE
1	1	0	1	1	1	1	0	FALSE
1	1	0	1	1	1	1	1	TRUE
1	1	1	0	0	0	0	0	TRUE
1	1	1	0	0	0	0	1	FALSE
1	1	1	0	0	0	1	0	FALSE
1	1	1	0	0	0	1	1	TRUE
1	1	1	0	0	1	0	0	FALSE
1	1	1	0	0	1	0	1	TRUE
1	1	1	0	0	1	1	0	TRUE
1	1	1	0	0	1	1	1	FALSE
1	1	1	0	1	0	0	0	FALSE
1	1	1	0	1	0	0	1	TRUE
1	1	1	0	1	0	1	0	TRUE
1	1	1	0	1	0	1	1	FALSE
1	1	1	0	1	1	0	0	TRUE
1	1	1	0	1	1	0	1	FALSE
1	1	1	0	1	1	1	0	FALSE
1	1	1	0	1	1	1	1	TRUE
1	1	1	1	0	0	0	0	FALSE
1	1	1	1	0	0	0	1	TRUE
1	1	1	1	0	0	1	0	TRUE
1	1	1	1	0	0	1	1	FALSE
1	1	1	1	0	1	0	0	TRUE
1	1	1	1	0	1	0	1	FALSE
1	1	1	1	0	1	1	0	FALSE
1	1	1	1	0	1	1	1	TRUE
1	1	1	1	1	0	0	0	TRUE
1	1	1	1	1	0	0	1	FALSE
1	1	1	1	1	0	1	0	FALSE
1	1	1	1	1	0	1	1	TRUE
1	1	1	1	1	1	0	0	FALSE
1	1	1	1	1	1	0	1	TRUE
1	1	1	1	1	1	1	0	TRUE
1	1	1	1	1	1	1	1	FALSE

E

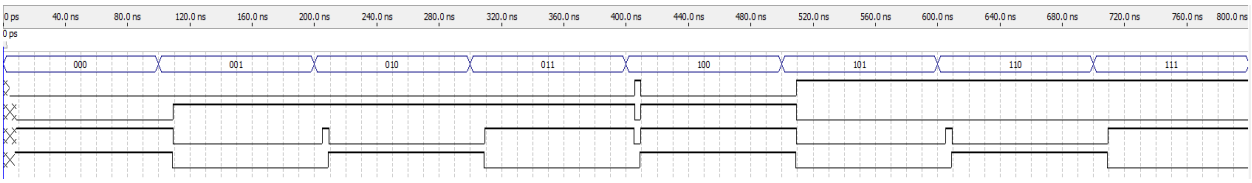
End

Testing

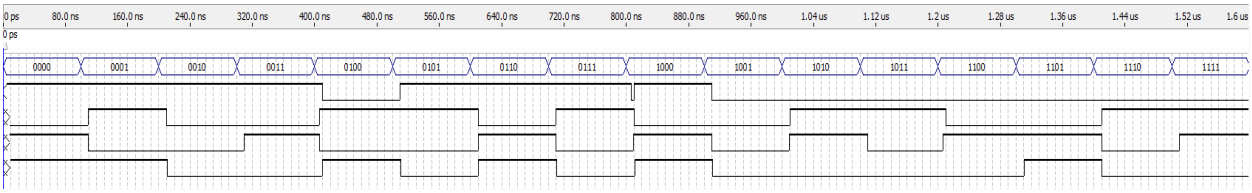
When testing the binary to grey code converter circuit I encountered no problems and it worked as expected for every single input combination. The following waveform applies to both the Verilog and Block-Diagram design since they essentially were the same circuit.



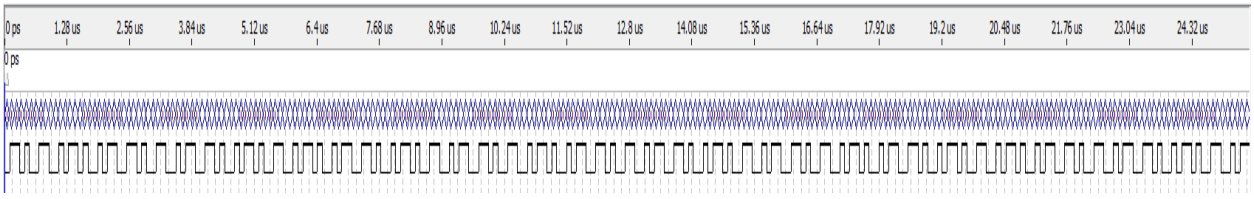
When testing the Constant Adder circuit I encountered no problems and it worked as expected for every single input combination.



When testing the Arbitrary Circuit I encountered one problem. The output of one combination was incorrect. This was fixed by reviewing the algebraic expressions that I created. By doing this I discovered that I had left out the last portion of an equation in my Verilog design. After correcting the equation the circuit worked as expected for every single input combination.



When testing the Parity Bit Generator I encountered no problems and it worked as expected for every single input combination.



Conclusion

In this lab I learned how to better create algebraic expressions from Kmaps, and then use those expressions in creating actual circuits that would perform as I wanted it to. This greatly increased my confidence in creating and using Kmaps. If I were to perform this lab a second time I believe I would try to simplify the equations that I got from the Kmaps to improve understandability and efficiency in the circuit design. Overall, the exercises in this lab proved to be enjoyable and were great fun to implement.