**COMP ENG 2DI4**
**Digital Logic**

# Lab 2: Combinational Logic

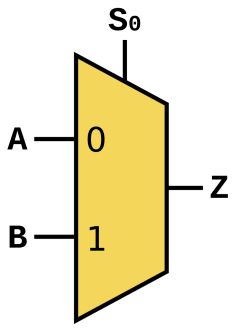
**Due date: Friday, October 22nd, 2021**
**By:**
**Jil Shah, shahj29, 400252316**
**Shiv Thakar, thakas4, 400247588**

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario.

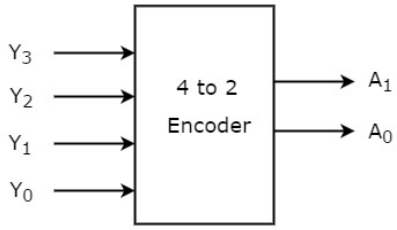
Submitted by: **Jil Shah, Shiv Thakar**

# Pre-Laboratory Preparation

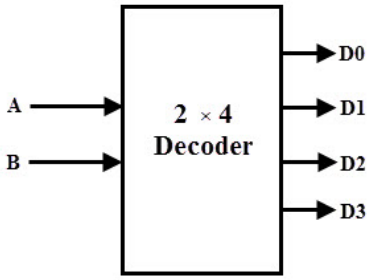
1. A multiplexer is a type of combinational circuit that has many different input lines that selects binary code from one input and directs it to one output line. An example for a multiplexer in a digital system is telephone networks, in this situation there are multiple audio signals that are integrated on a single line with the use of multiplexers. This allows the audio signals to be isolated and reach the recipient.




2. An encoder is a type of digital circuit that inverses the results of a decoder circuit. An encoder consists of *n* output lines and *2n* input lines. The encoder generates binary code depending on the input values on each line. An application in which encoders are used is large cranes, the encoders are mounted directly onto the motor shaft, this is so the position feedback can be returned and used to determine when the load can be released or picked up.




3. A decoder is a type of combinational circuit that consists of *n* inputs and *2n* outputs. Decoders are used as a means of translation coded information from one type input to another. An example of a decoder is found in the control unit of the central processing unit, in this case the decoder will decode the different instructions, this is so the specific control lines are activated; the different operations in the arithmetic logic unit (ALU) and the central processing unit (CPU).



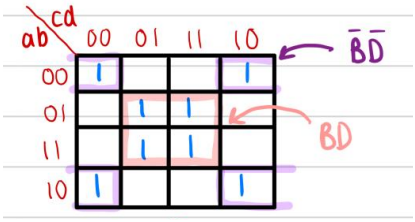

4. A Binary Coded Decimal (BCD) is an encoder that is used to convert decimal numbers into their binary equivalents. It is an easy way to represent decimal values, since each digit is represented

by its own four-bit binary value. An application for BCD is how the BIOS in some personal computers directly uses BCD's to store date and time. This is because the real-time clock chip in the motherboard provides the time encoded in BCD and this form can be easily converted to a display by using ASCII.

5. Minimize the following boolean function and draw the circuit diagram.

$$F(a, b, c, d) = \sum(0, 2, 5, 7, 8, 10, 13, 15)$$

a) Complete the correct K-map of function F(a,b,c,d)



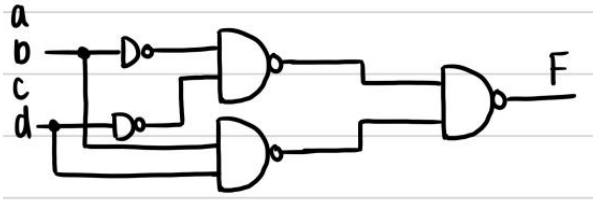

b) Write F(a,b,c,d) in standard form

$F_m = a'b'c'd' + a'b'cd' + a'b'c'd' + a'bc'd + a'bcd + ab'c'd' + ab'cd' + abc'd + abcd$

c) Write the minimized boolean F(a,b,c,d) function using K-map reduction techniques.

$$F_m = b'd' + bd$$

d) Write the minimized boolean F(a,b,c,d) function using only NAND gates. Show how you verified the minimized circuit.




$F_m = ((b * d)' * ((b * b)' * (d * d)')')'$ ; NAND Equation

$F_m = ((b * d)' * (b' * d')')'$ ; Theorem 1: x*x=x

$F_m = ((b' + d') * ((b + d)')')'$ ; Theorem 5, Demorgan: (x+y)' = x'y'

$F_m = ((b' + d') * (b + d))'$ ; Theorem 3, involution: (x')'

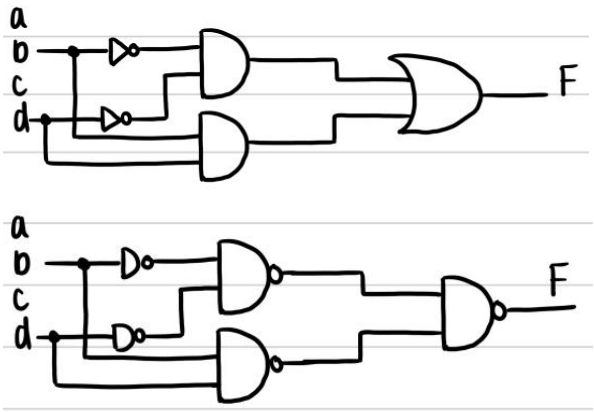
$F_m = (b' + d')' + (b + d)'$ ; Theorem 5, Demorgan b: (x*y)' = x'+y'

$F_m = ((bd)')' + b'd'$ ; Theorem 5, Demorgan a: (x+y)' +x'y'

$F_m = bd + b'd'$ ; Theorem 3, involution: (x')'

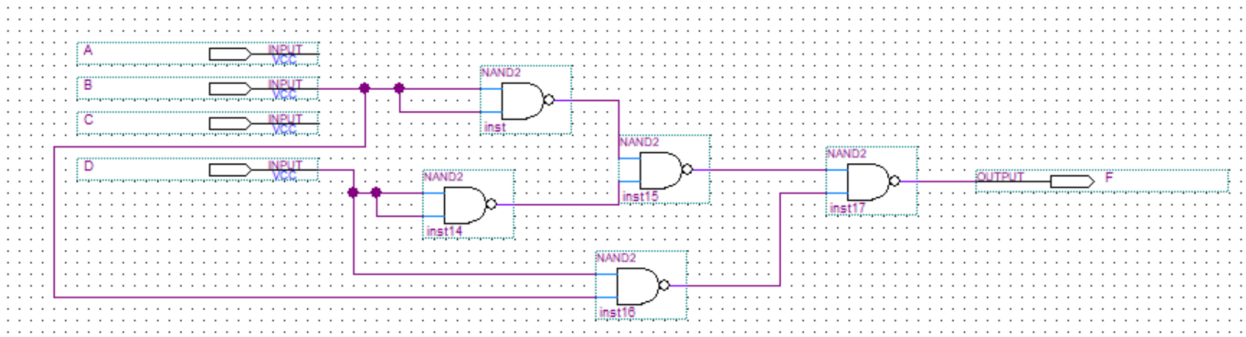
$F_m = b'd' + bd$ ; rearrange

e) Use logic symbols to draw the minimized boolean F(a,b,c,d) function using only NAND gates.




# 2.4.1 The Karnaugh-Map for Boolean Minimization

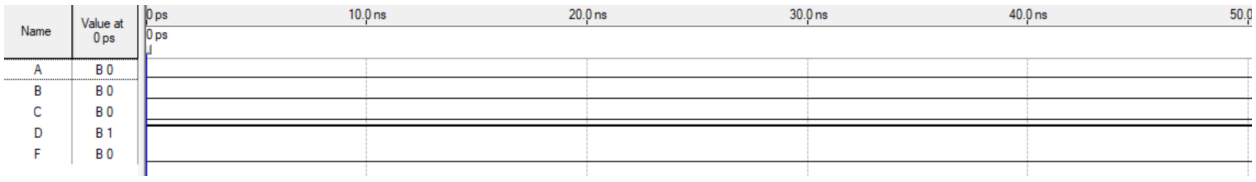
**Circuit:**



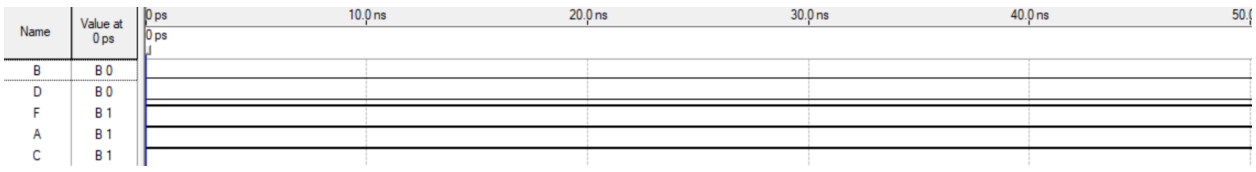

**Simulation Results:**

| Combination # | A | B | C | D | F |
|---|---|---|---|---|---|
| **1** | 0 | 0 | 0 | 1 | 0 |
| **2** | 1 | 0 | 1 | 0 | 1 |
| **3** | 1 | 1 | 0 | 0 | 0 |
| **4** | 0 | 1 | 1 | 1 | 1 |

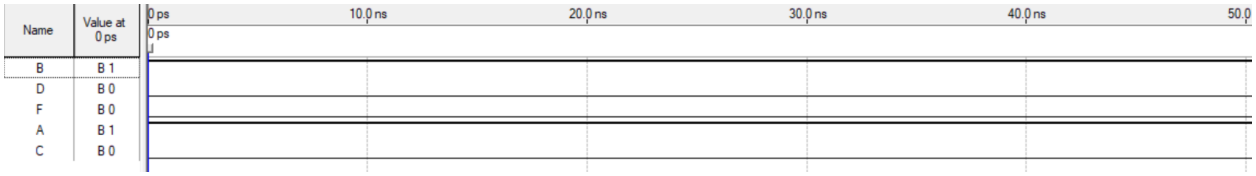
Combination 1:



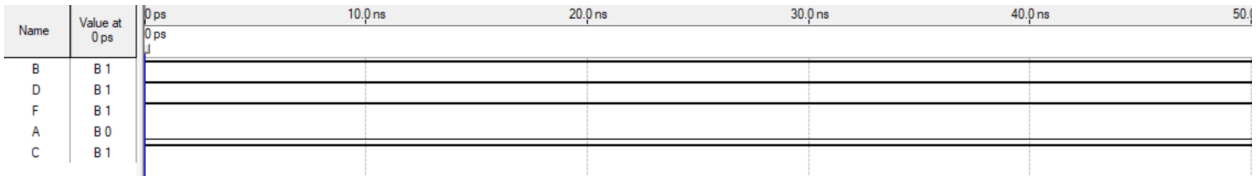

Combination 2:




Combination 3:

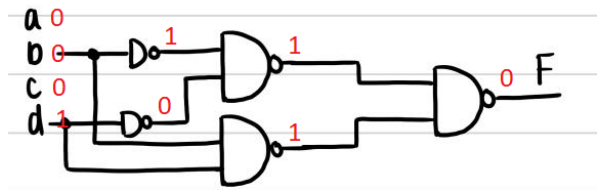
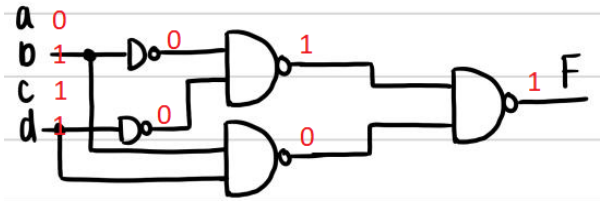



Combination 4:



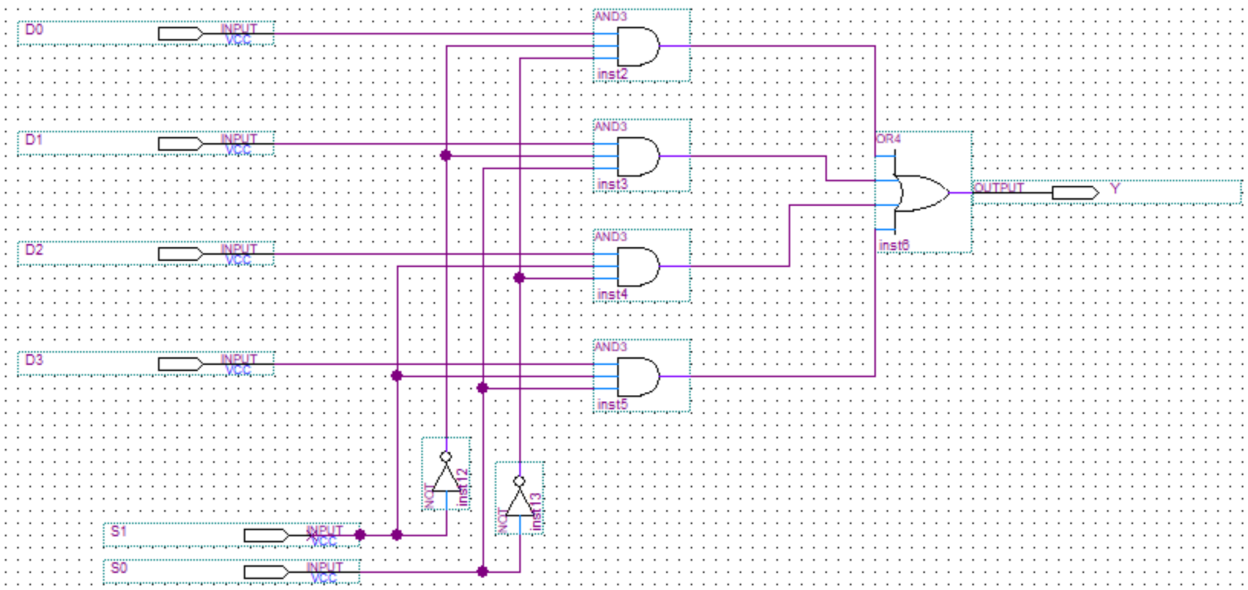

**Description**

Through the various combinations that we tested, 0001,1010,1100, and 0111, we can see that the output matches the ones calculated in the truth table and given to us in the pre-lab. Due to this, we can see that the circuits are built currently and the nand-gate logic circuit was simplified correctly.

| | |
|---|---|
| $a = 0,\ b = 0,\ c = 0,\ d = 1$   $F_m = b'd' + bd = (0' * 1') + (0 * 1) = 0$ | $a = 0,\ b = 1,\ c = 1,\ d = 1$   $F_m = b'd' + bd = (1' * 1') + (1 * 1) = 1$ |

# 2.4.2 The Multiplexer (MUX)

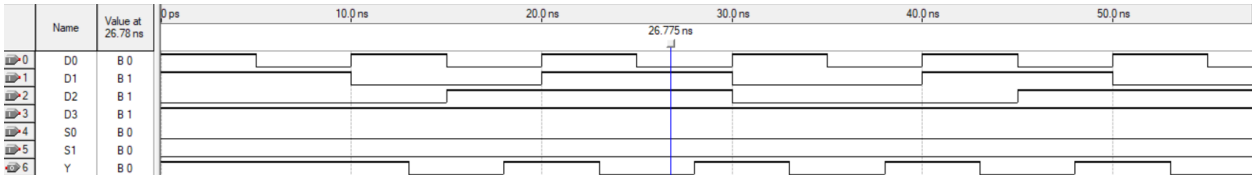
**Circuit:**



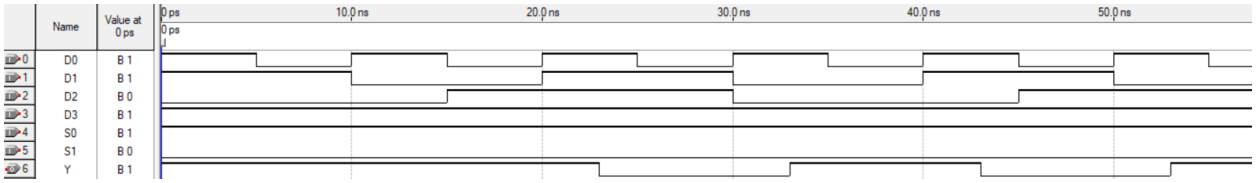

| S0 | S1 | Y |
|---|---|---|
| 0 | 0 | D0 |
| 0 | 1 | D1 |
| 1 | 0 | D2 |
| 1 | 1 | D3 |

**Simulation Results:**

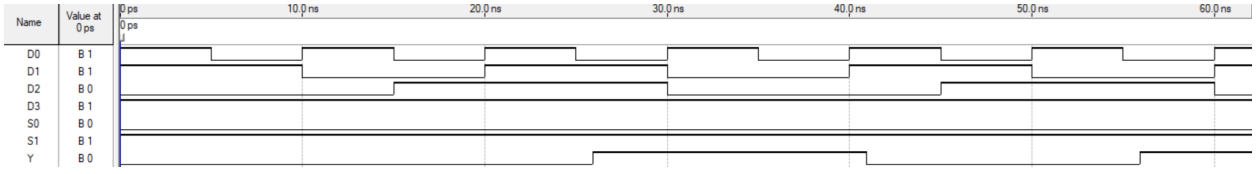
D0 outputted



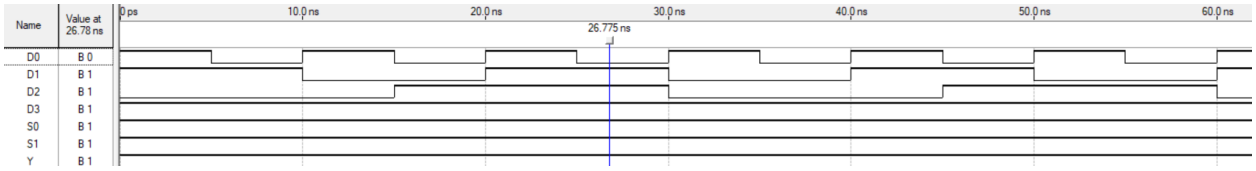

D1 Outputted




D2 Outputted




D3 Outputted




**Description:**

For the multiplexer circuit, we verified that our built circuit was correct by testing different inputs and comparing the outputs to the function table. We tested for different combinations of $S_0$ *and* $S_1$ that would result in $D_0, D_1, D_2,$ *and* $D_3$.

$D_0 = 0, D_1 = 1, D_2 = 1, \ D_3 = 1$

$S_0 = 0$ *and* $S_1 = 0$

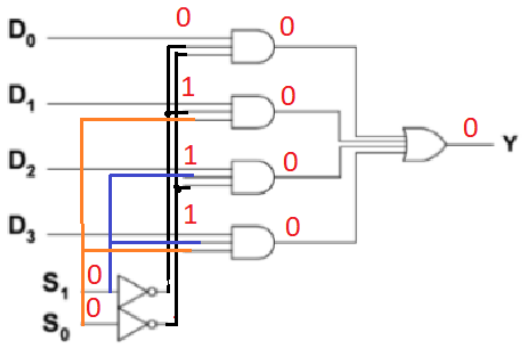



Figure 2.1: 4:1 MUX

$D_0 = 1, D_1 = 1, D_2 = 0, \ D_3 = 1$

$S_0 = 0$ *and* $S_1 = 1$

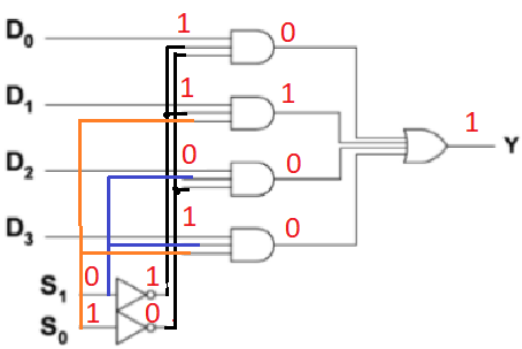
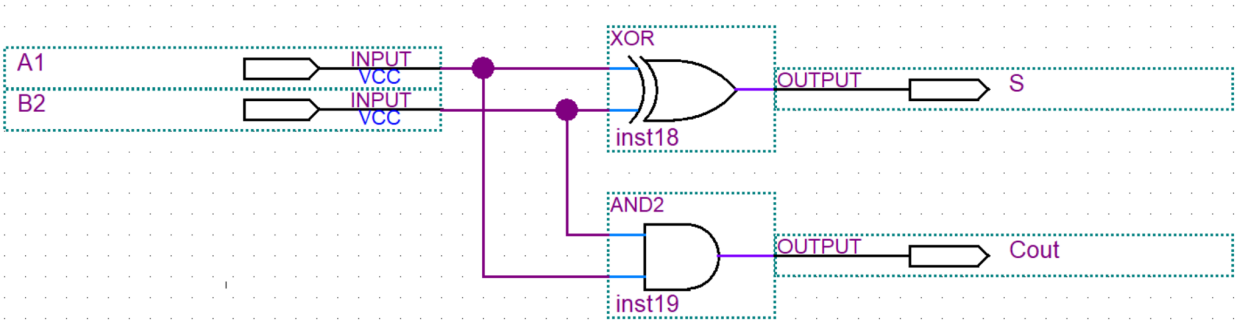



Figure 2.1: 4:1 MUX

# 2.4.3 Half Adder

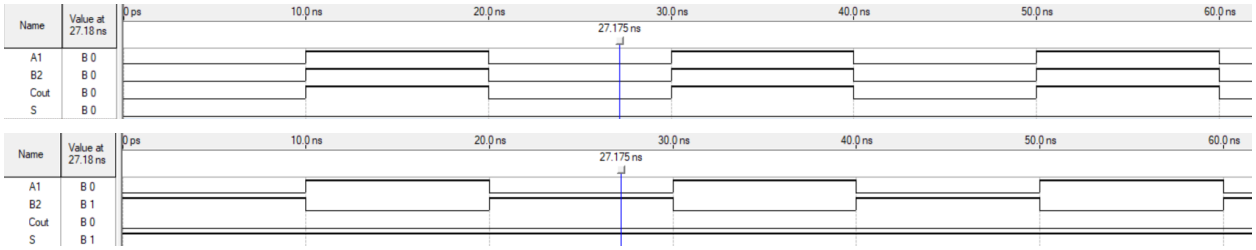
**Circuit:**




| A | B | Cout | S |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |

| | | | |
|---|---|---|---|
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

**Simulation Results:**




**Description:**

For the half adder circuit that we built, we verified this circuit was correct by validating our outputs with the outputs that were determined by the truth table using the boolean function and by using the graphical calculations.

A=0, B=0

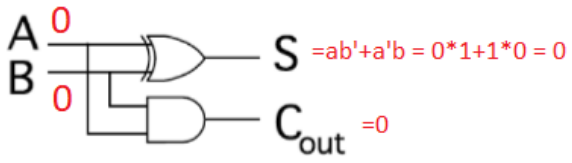



Figure 2.2: The half-adder

A=0, B=1

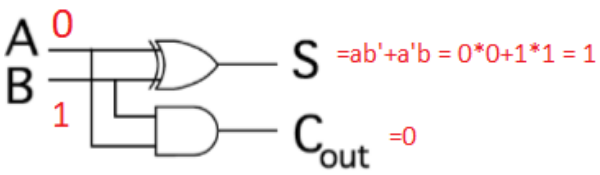
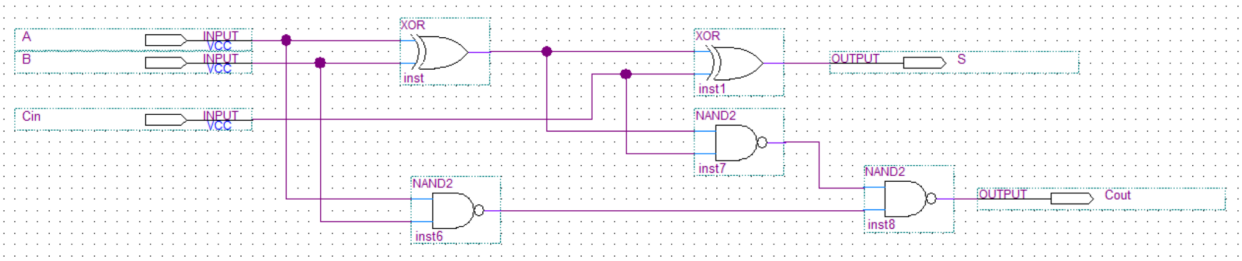



Figure 2.2: The half-adder

# 2.4.4 Full Adder

**Circuit:**

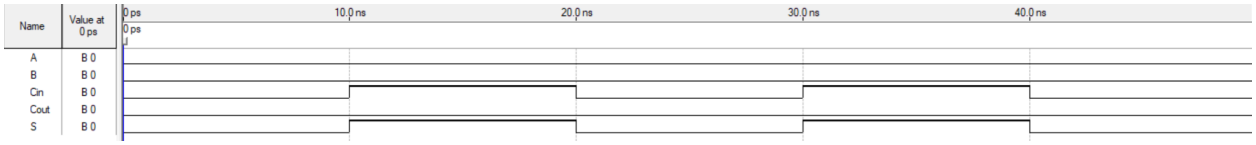



| A | B | Cin | Cout | S |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |

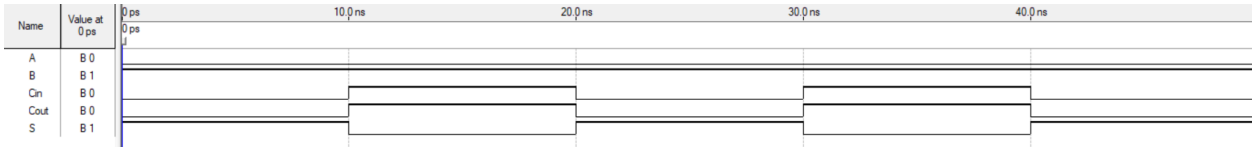
| | | | | |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

**Simulation Results:**

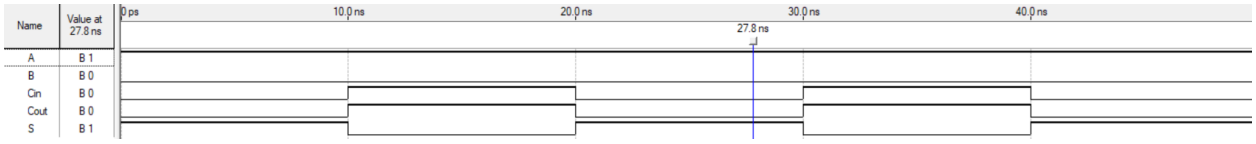
A = 0 B = 0 Cin = Alternates Between 0 and 1



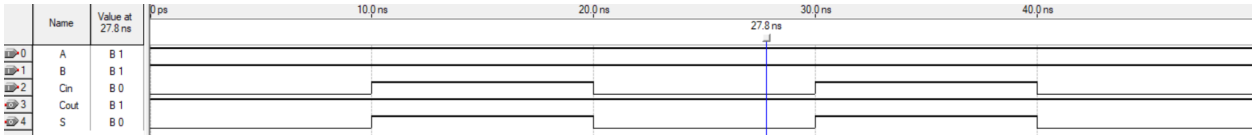

A = 0 B = 1 Cin = Alternates Between 0 and 1




A = 1 B = 0 Cin = Alternates Between 0 and 1




A = 1 B = 1 Cin = Alternates Between 0 and 1




**Description:**

The full adder circuit was verified by testing the unique combinations with the truth table outputs that were determined by the boolean functional equation. The simulation results above shows the output for different combinations of A and B, and a waveform to alternate between 0 and 1 for Cin.

A=0 B=0 Cin =0

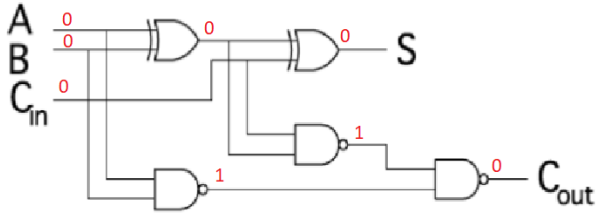



Figure 2.3: The full-adder

A=0 B=0 Cin =1

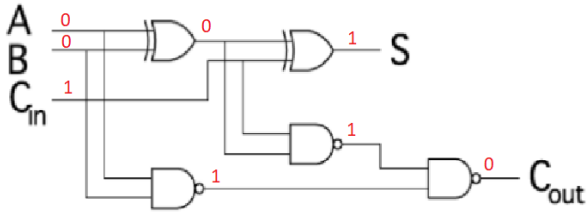



Figure 2.3: The full-adder

# 2.4.5 Binary Code Decimal and the 7-Segment Display Encoder

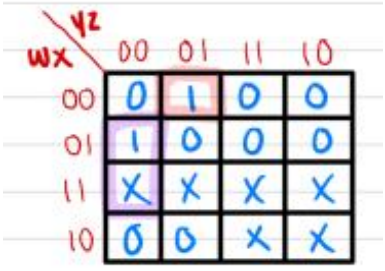
**Truth Table:**

| Decimal | w | x | y | z | a | b | c | d | e | f | g |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 6 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

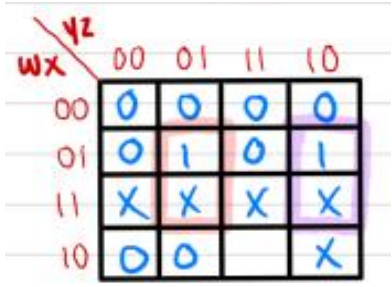
**K-map Representation:**

- Assume 10-15 can be either 0 and 1, and we don't care which one it is.

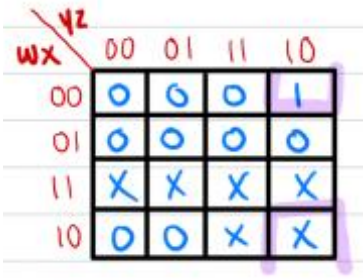
$Fa = xy'z' + w'x'y'z$



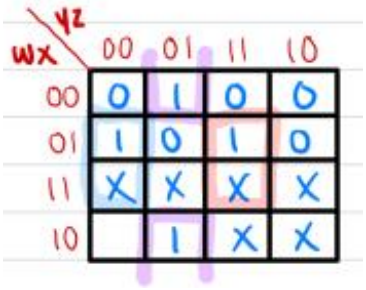

$Fb = xy'z + xyz'$



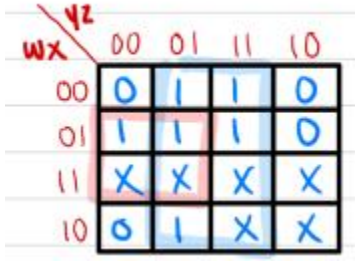

$Fc = x'yz'$



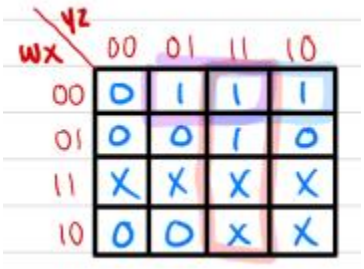

$Fd = xy'z' + wz + xyz + x'y'$

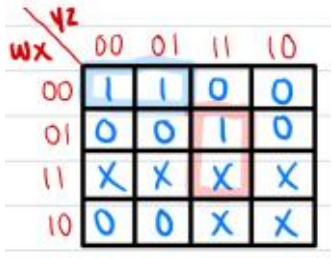



$Fe = z + xy'$

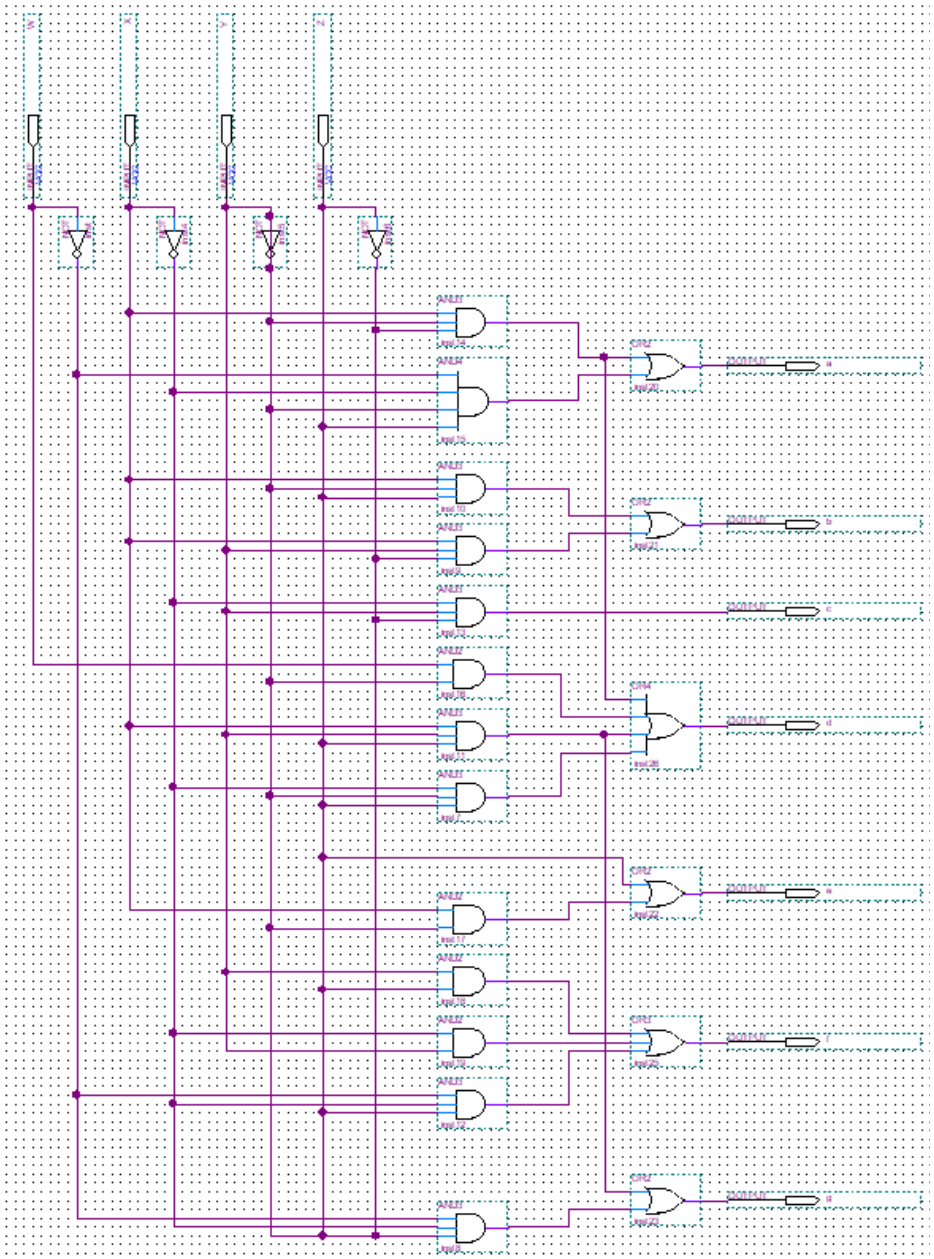



$Ff = yz + x'y + w'x'z$



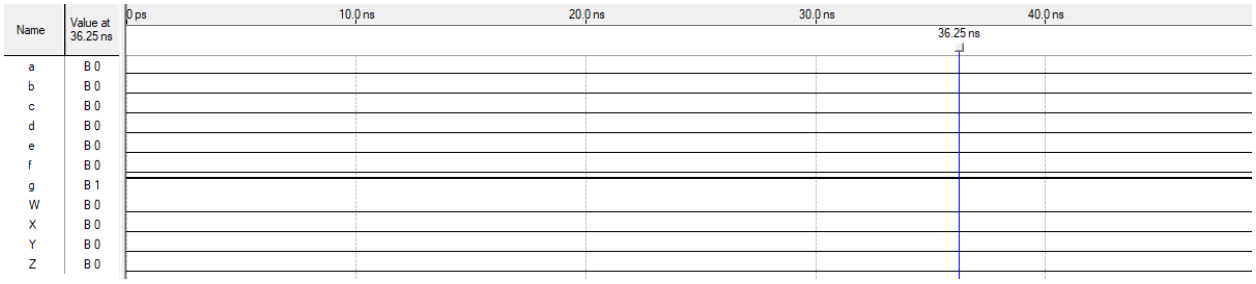
| | | |
|---|---|---|
| $Fg = w'x'y' + xyz$   | | |

**Circuit:**

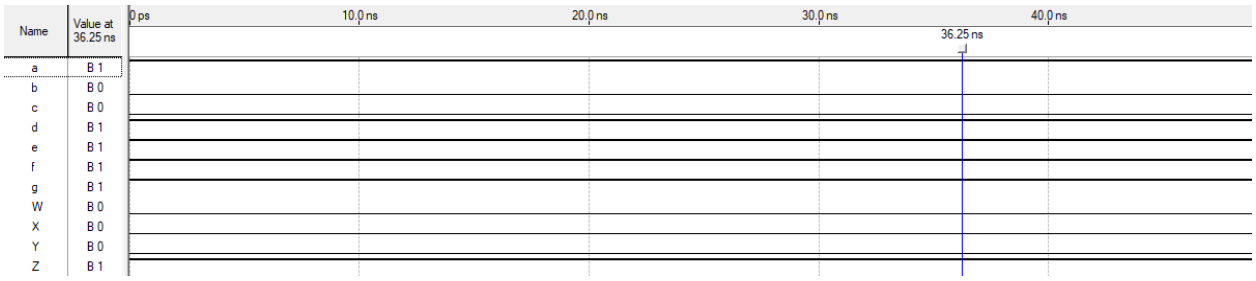


**Simulation:**

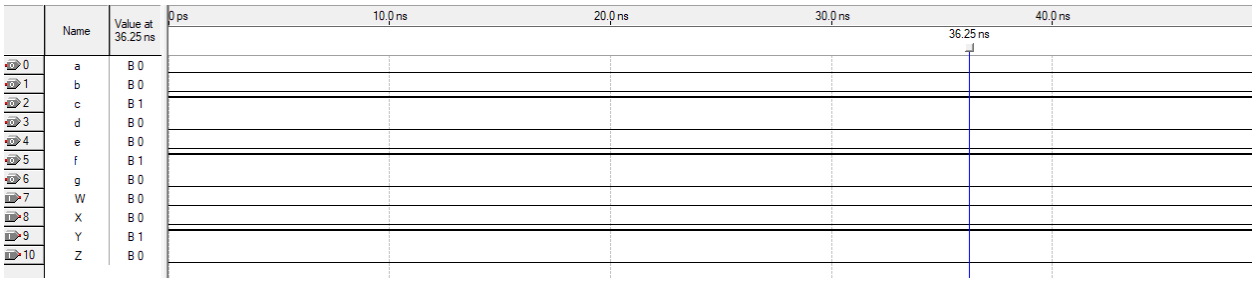
To Output 0:



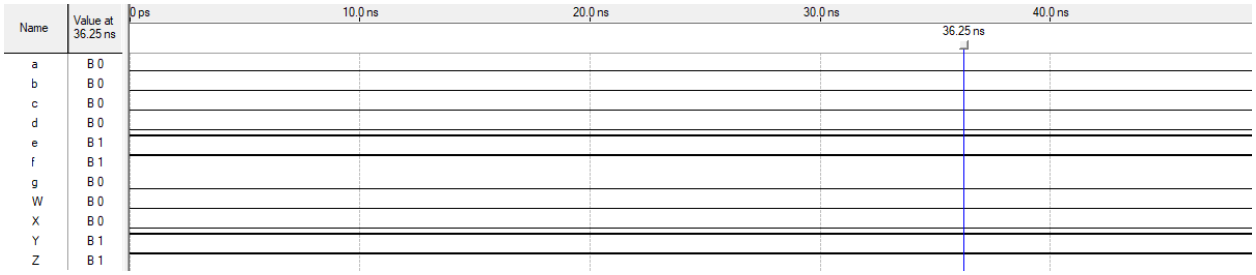

To Output 1:



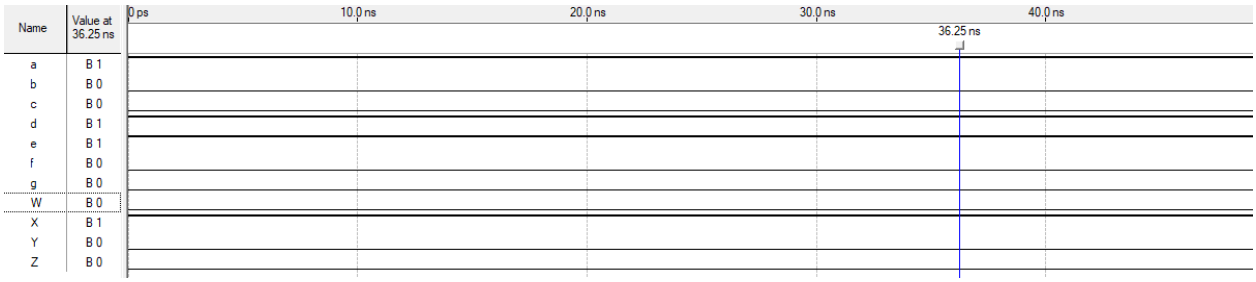

To Output 2:




To Output 3:




To Output 4:




**Description:**

We verified our circuit was working correctly by confirming the outputs that we determined in the simulation match the ones determined in the truth table. We tested for all 4 of the 9 numbers in the 7-segment display numbers and checked if the output a-g zeros and ones were correct.