



De La Salle University - Manila

Second Term, A.Y. 2019-2020

Introduction to Computer Organization & Architecture 1 (CSARCH1)

**Digit to Code 128 Barcode Segment Converter**

BENEDICTOS, Bianca

TEJADA, Mikayla

TENG, Lance

Group 6 - S12

Submission Date: March 16, 2020

Presentation Date: March 17, 2020

## **INTRODUCTION**

### **A. Problem Description**

When manually dealing with large amounts of data, there will always be that undeniable possibility of human error. Companies, businesses, and other industries that keep track of inventory or pricing information require some of their employees to undergo training to gain familiarity with the entire procedure which is indeed costly and time consuming.

Nowadays, it is evident that technological development has flourished especially those that help people cut costs and save time. One of these technologies are barcodes, a pattern of parallel lines and spaces used to easily store, transfer, and process information in such a way that a machine could read. Barcodes are both cost-effective and reliable since these allow people to deal with a large amount of data and process faster than manual data entry with high accuracy.

CODE 128 is a computer friendly linear barcode that was developed in 1981 which can represent all 128 ASCII code characters. This is also considered as the most easily read barcode and the most flexible common linear symbology (barcode language). This symbology is used in various industries where large amounts of data must be encoded to a limited amount of space. Food industries in Australia and New Zealand, and the USA's apparel industry are some industries that utilize CODE 128.

The medium is composed of seven sections - leading quiet zone, start symbol, encoded data, check symbol, stop symbol, final bar, and trailing quiet zone. CODE 128 makes use of 4 types of bars which is relative to its thickness and color. Despite the name, it cannot encode all 128 distinct symbols, so it cannot record all 128 ASCII characters. Instead, the start symbol is

used to dictate which code set among three (A, B or C) is to be used. Code sets A and B feature all 128 characters, while set C encodes two digits with a single point, allowing for more information stored, if needed.

## **B. Application**

Barcodes allow stores to track and store information about their products. Their representation allows them to be easily scanned, and therefore serves an important role to automate entering information. This makes barcodes widely used and much more efficient in various industries such as shopping, shipping, and inventory control. There are several different standards used for barcodes, but one of the most common ones used is the CODE 128 barcode.

Through the years, the application of barcodes has greatly helped people who deal with data collection because barcodes are versatile. It could save and retrieve information rapidly which is why it could be used for any kind of necessary data collection. Businesses are one of the major beneficiaries of the invention of barcodes since it is inexpensive, user-friendly, and helped them save time and eliminate the possibility of human error when entering their data without breaking the bank. Furthermore, their inventory control improves since barcodes could also be customized to contain and precisely track other relevant information such as location, plus, learning how to use barcodes and scanners are easy to learn so training time for employees won't be necessary. This barcode system basically minimizes users' cost in terms of money, time, and effort.

### C. Definition of Input and Output Variables

#### a. Input Variables

The input is a number from 0 to 15 represented in 4 bits (binary), namely

$\{i_1, i_2, i_3, i_4\}$ . This will serve as the 4 input variables for the machine.

#### b. Output Variables





The output number is represented in 11 bits (binary), namely

$\{b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_{10}, b_{11}\}$ . Each bit represents whether the

block is shaded black or white for the bar according to the CODE 128 standard representation.

1 - Black

0 - White

Sample Input	Sample Output
0000 (0)	11011001100 
0001 (1)	11001101100 
0010 (2)	11001101100 
0011 (3)	10010011000 

**Table 1: Sample inputs and their respective outputs**

#### D. Definition of Logic Levels and Events

A 2-level logic will be implemented for all input and output variables.

The input is a 4-bit number represented by four variables  $\{i_1, i_2, i_3, i_4\}$ , with  $i_1$  being the most significant bit and  $i_4$  being the least significant bit. The output is represented by 11 variables, one for each module of the bar segment.

Variable Name	0	1
$b_1$	Module 1 is a white gap.	Module 1 is a black bar.
$b_2$	Module 2 is a white gap.	Module 2 is a black bar.
$b_3$	Module 3 is a white gap.	Module 3 is a black bar.
$b_4$	Module 4 is a white gap.	Module 4 is a black bar.
$b_5$	Module 5 is a white gap.	Module 5 is a black bar.
$b_6$	Module 6 is a white gap.	Module 6 is a black bar.
$b_7$	Module 7 is a white gap.	Module 7 is a black bar.
$b_8$	Module 8 is a white gap.	Module 8 is a black bar.
$b_9$	Module 9 is a white gap.	Module 9 is a black bar.
$b_{10}$	Module 10 is a white gap.	Module 10 is a black bar.
$b_{11}$	Module 11 is a white gap.	Module 11 is a black bar.

**Table 2. Output variables and their corresponding logic levels.**

#### E. Intended System Operations and Conditions

To simplify the conversion, it is assumed that the CODE C start character is used, and this is used to translate the encoded data only. In most cases, only the first 10 characters (0-9) are used to represent a code for products, thus only those characters will be used. The input will be

taken in from a numeric keypad after a number button is pressed. The output can be used by a printer to print out a barcode to use.

## DESIGN METHODOLOGY

### A. Truth Tables

Input and Output table:

$i_1$	$i_2$	$i_3$	$i_4$	$b_1$	$b_2$	$b_3$	$b_4$	$b_5$	$b_6$	$b_7$	$b_8$	$b_9$	$b_{10}$	$b_{11}$
0	0	0	0	1	1	0	1	1	0	0	1	1	0	0
0	0	0	1	1	1	0	0	1	1	0	1	1	0	0
0	0	1	0	1	1	0	0	1	1	0	0	1	1	0
0	0	1	1	1	0	0	1	0	0	1	1	0	0	0
0	1	0	0	1	0	0	1	0	0	0	1	1	0	0
0	1	0	1	1	0	0	0	1	0	0	1	1	0	0
0	1	1	0	1	0	0	1	1	0	0	1	0	0	0
0	1	1	1	1	0	0	1	1	0	0	0	1	0	0
1	0	0	0	1	0	0	0	1	1	0	0	1	0	0
1	0	0	1	1	1	0	0	1	0	0	1	0	0	0
1	0	1	0	1	1	0	0	1	0	0	0	1	0	0
1	0	1	1	1	1	0	0	0	1	0	0	1	0	0
1	1	0	0	1	0	1	1	0	0	1	1	1	0	0
1	1	0	1	1	0	0	1	1	0	1	1	1	0	0
1	1	1	0	1	0	0	1	1	0	0	1	1	1	0
1	1	1	1	1	0	1	1	1	0	0	1	1	0	0

Table 3. Truth table for Code 128 Barcode

### B. Boolean Functions

$$b_1 = 1$$

$$b_2 = \overline{i_1} \overline{i_2} \overline{i_3} + \overline{i_2} i_3 \overline{i_4} + i_1 \overline{i_2} i_4$$

$$b_3 = i_1 i_2 \overline{i_3} \overline{i_4} + i_1 i_2 i_3 i_4$$

$$b_4 = \overline{i_1} \overline{i_3} \overline{i_4} + \overline{i_1} i_3 i_4 + i_2 \overline{i_4} + i_1 i_2$$

$$b_5 = \overline{i_2} \overline{i_3} + i_3 \overline{i_4} + i_2 i_4$$

$$b_6 = \overline{i_1} \overline{i_2} \overline{i_3} i_4 + \overline{i_1} \overline{i_2} i_3 \overline{i_4} + i_1 \overline{i_2} \overline{i_3} \overline{i_4} + i_1 \overline{i_2} i_3 i_4$$

$$b_7 = \overline{i_1} \overline{i_2} i_3 i_4 + i_1 i_2 \overline{i_3}$$

$$b_8 = \overline{i_1} \overline{i_3} + \overline{i_1} \overline{i_2} i_4 + \overline{i_3} i_4 + i_2 \overline{i_4} + i_1 i_2$$

$$b_9 = \overline{i_1} \overline{i_3} + \overline{i_2} \overline{i_4} + \overline{i_3} \overline{i_4} + i_2 i_4 + i_1 i_3$$

$$b_{10} = \overline{i_1} \overline{i_2} i_3 \overline{i_4} + i_1 i_2 i_3 \overline{i_4}$$

$$b_{11} = 0$$

### C. Minimization/Simplification using Boolean Algebra or K-maps

Simplification will be done using K-maps. This allows for minimal delay when the circuit is implemented, allowing for faster usage.

a.  $b_1 = 1$

		i3, i4			
		00	01	11	10
i1, i2	00	1	1	1	1
	01	1	1	1	1
	11	1	1	1	1
	10	1	1	1	1

**Table 4. K-Mapping for  $b_1$**

b.  $b_2 = \bar{i}_1 \bar{i}_2 \bar{i}_3 + \bar{i}_2 i_3 \bar{i}_4 + i_1 \bar{i}_2 i_4$

		i3, i4			
		00	01	11	10
i1, i2	00	1	1	0	1
	01	0	0	0	0
	11	0	0	0	0
	10	0	1	1	1

Table 5. K-Mapping for  $b_2$

c.  $b_3 = i_1 i_2 \bar{i}_3 \bar{i}_4 + i_1 i_2 i_3 i_4$

		i3, i4			
		00	01	11	10
i1, i2	00	0	0	0	0
	01	0	0	0	0
	11	1	0	1	0
	10	0	0	0	0

Table 6. K-Mapping for  $b_3$

d.  $b_4 = \bar{i}_1 \bar{i}_3 \bar{i}_4 + \bar{i}_1 i_3 i_4 + i_2 \bar{i}_4 + i_1 i_2$

		i3, i4			
		00	01	11	10
i1, i2	00	1	0	1	0
	01	1	0	1	1
	11	1	1	1	1
	10	0	0	0	0

Table 7. K-Mapping for  $b_4$



e.  $b_5 = \bar{i}_2 \bar{i}_3 + i_3 \bar{i}_4 + i_2 i_4$

		i3, i4			
		00	01	11	10
i1, i2	00	1	1	0	1
	01	0	1	1	1
	11	0	1	1	1
	10	1	1	0	1

**Table 8. K-Mapping for  $b_5$**

f.  $b_6 = \bar{i}_1 \bar{i}_2 \bar{i}_3 i_4 + \bar{i}_1 \bar{i}_2 i_3 \bar{i}_4 + i_1 \bar{i}_2 \bar{i}_3 \bar{i}_4 + i_1 \bar{i}_2 i_3 i_4$

		i3, i4			
		00	01	11	10
i1, i2	00	0	1	0	1
	01	0	0	0	0
	11	0	0	0	0
	10	1	0	1	0

**Table 9. K-Mapping for  $b_6$**

g.  $b_7 = \overline{i_1} \overline{i_2} i_3 i_4 + i_1 i_2 \overline{i_3}$

		i3, i4			
		00	01	11	10
i1, i2	00	0	0	1	0
	01	0	0	0	0
	11	1	1	0	0
	10	0	0	0	0

Table 10. K-Mapping for  $b_7$

h.  $b_8 = \overline{i_1} \overline{i_3} + \overline{i_1} \overline{i_2} i_4 + \overline{i_3} i_4 + i_2 \overline{i_4} + i_1 i_2$

		i3, i4			
		00	01	11	10
i1, i2	00	1	1	1	0
	01	1	1	0	1
	11	1	1	1	1
	10	0	1	0	0

Table 8. K-Mapping for  $b_8$

i.  $b_9 = \bar{i}_1 \bar{i}_3 + \bar{i}_2 \bar{i}_4 + \bar{i}_3 \bar{i}_4 + i_2 i_4 + i_1 i_3$

		i3, i4			
		00	01	11	10
i1, i2	00	1	1	0	1
	01	1	1	1	0
	11	1	1	1	1
	10	1	0	1	1

Table 12. K-Mapping for  $b_9$

j.  $b_{10} = \bar{i}_1 \bar{i}_2 \bar{i}_3 \bar{i}_4 + i_1 i_2 i_3 \bar{i}_4$

		i3, i4			
		00	01	11	10
i1, i2	00	0	0	0	1
	01	0	0	0	0
	11	0	0	0	1
	10	0	0	0	0

Table 13. K-Mapping for  $b_{10}$

k.  $b_{11} = 0$

		i3, i4			
		00	01	11	10
i1, i2	00	0	0	0	0
	01	0	0	0	0
	11	0	0	0	0
	10	0	0	0	0

Table 14. K-Mapping for  $b_{11}$



## D. Logic Gate Implementation

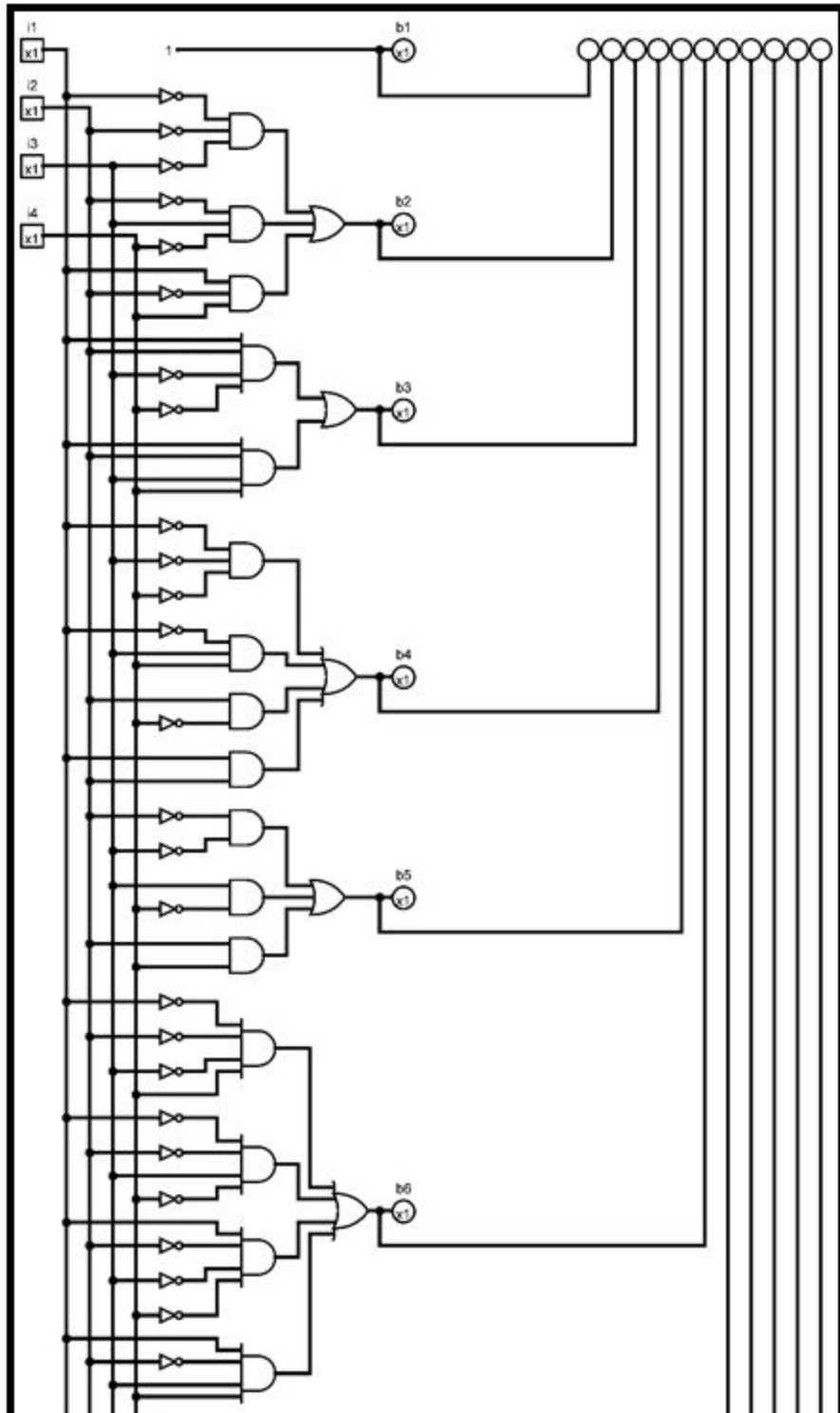


Figure 1. Logic Gate Implementation

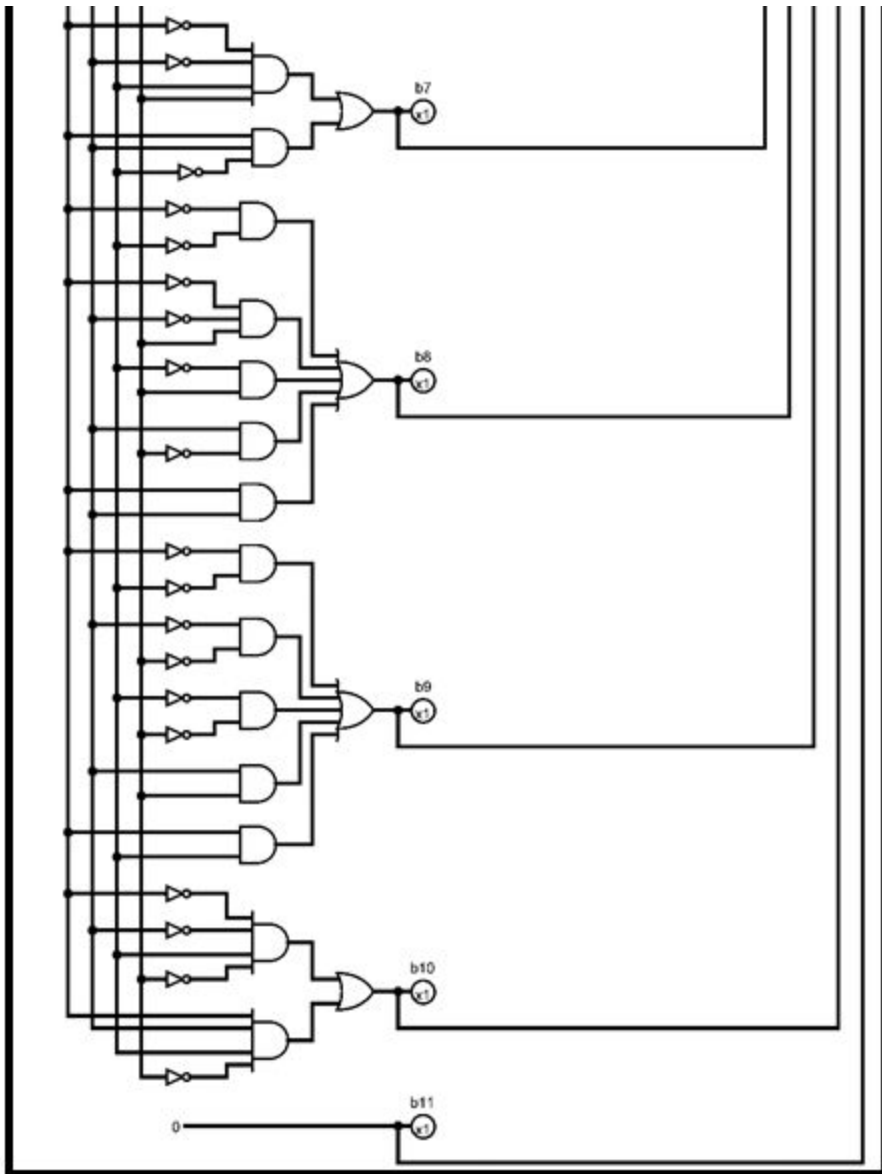


Figure 1.1. Logic Gate Implementation (continuation)

## RESULTS AND CONCLUSION

### A. Screenshots of Simulation

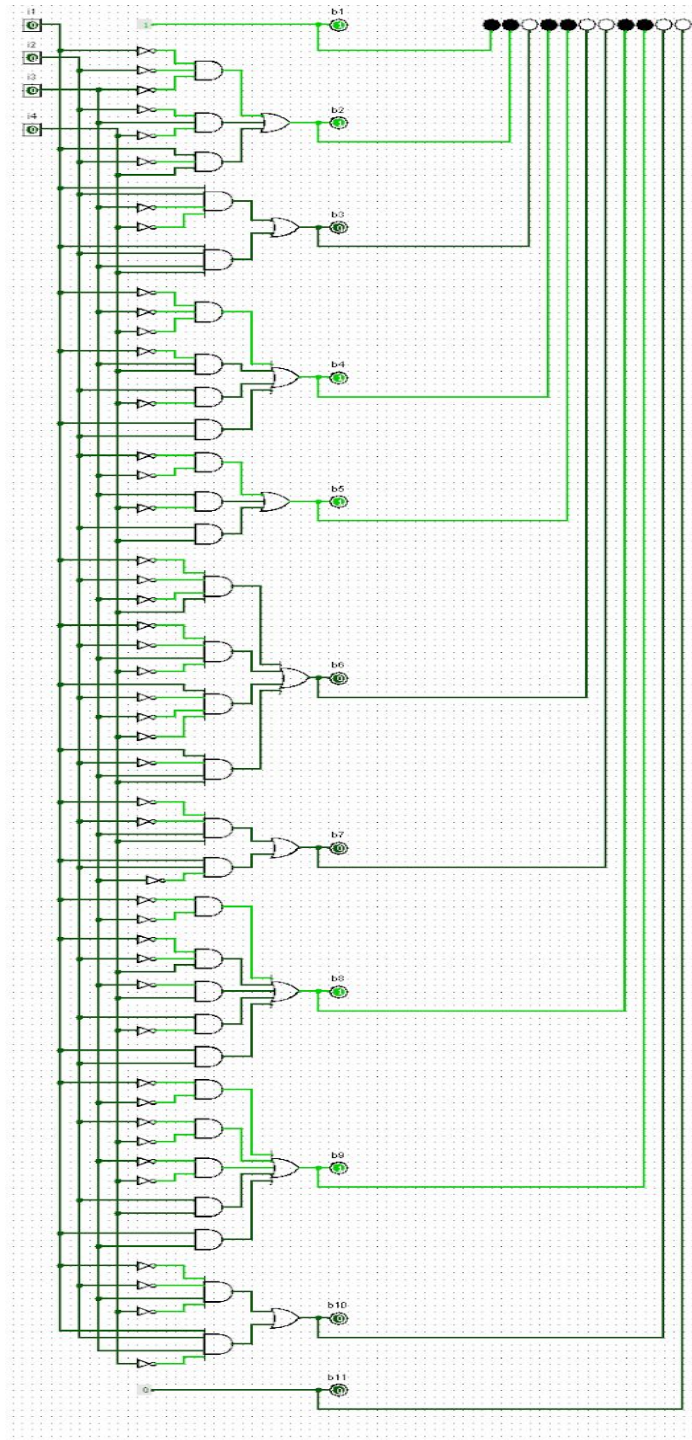


Figure 2.1. Simulation for input 0000.



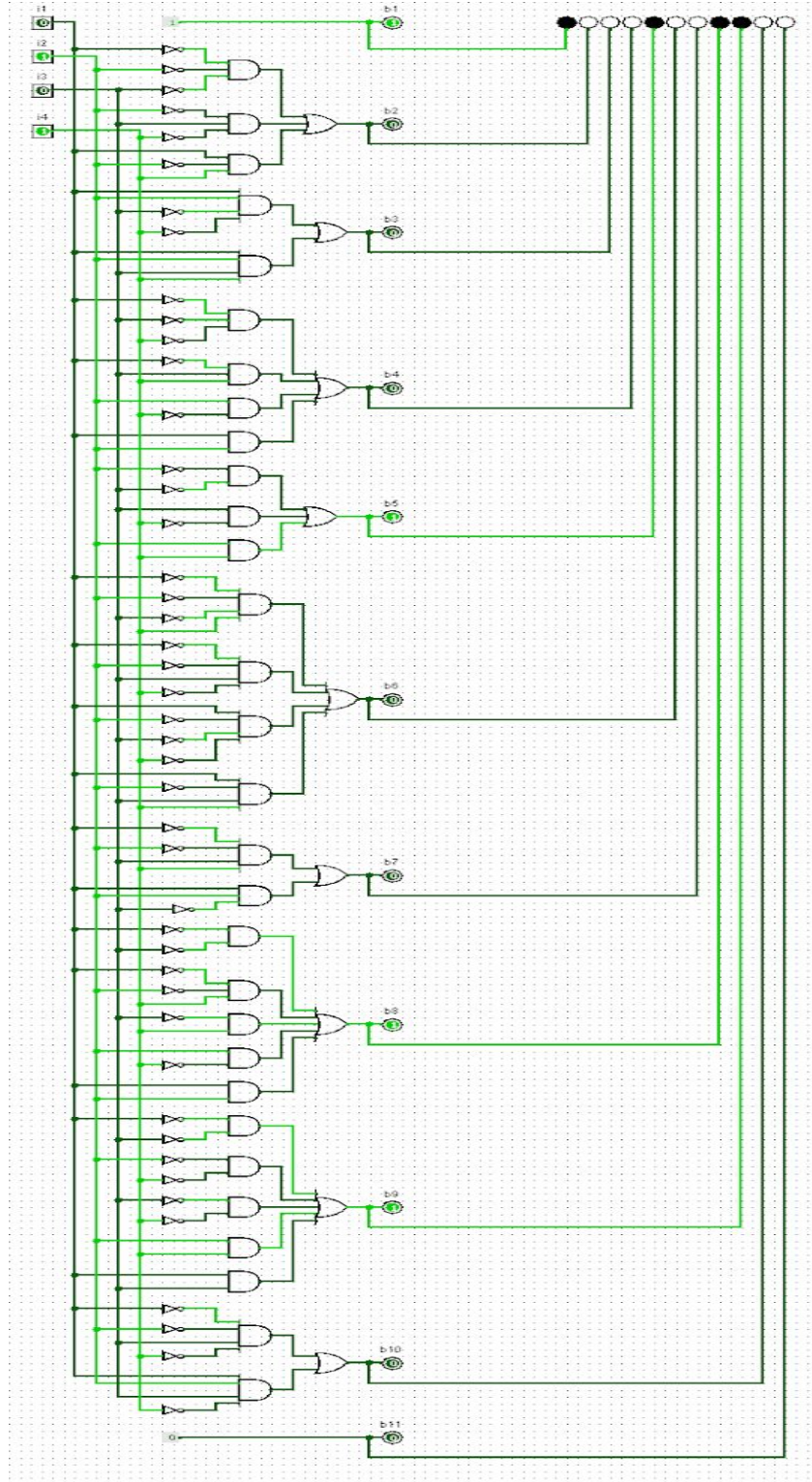
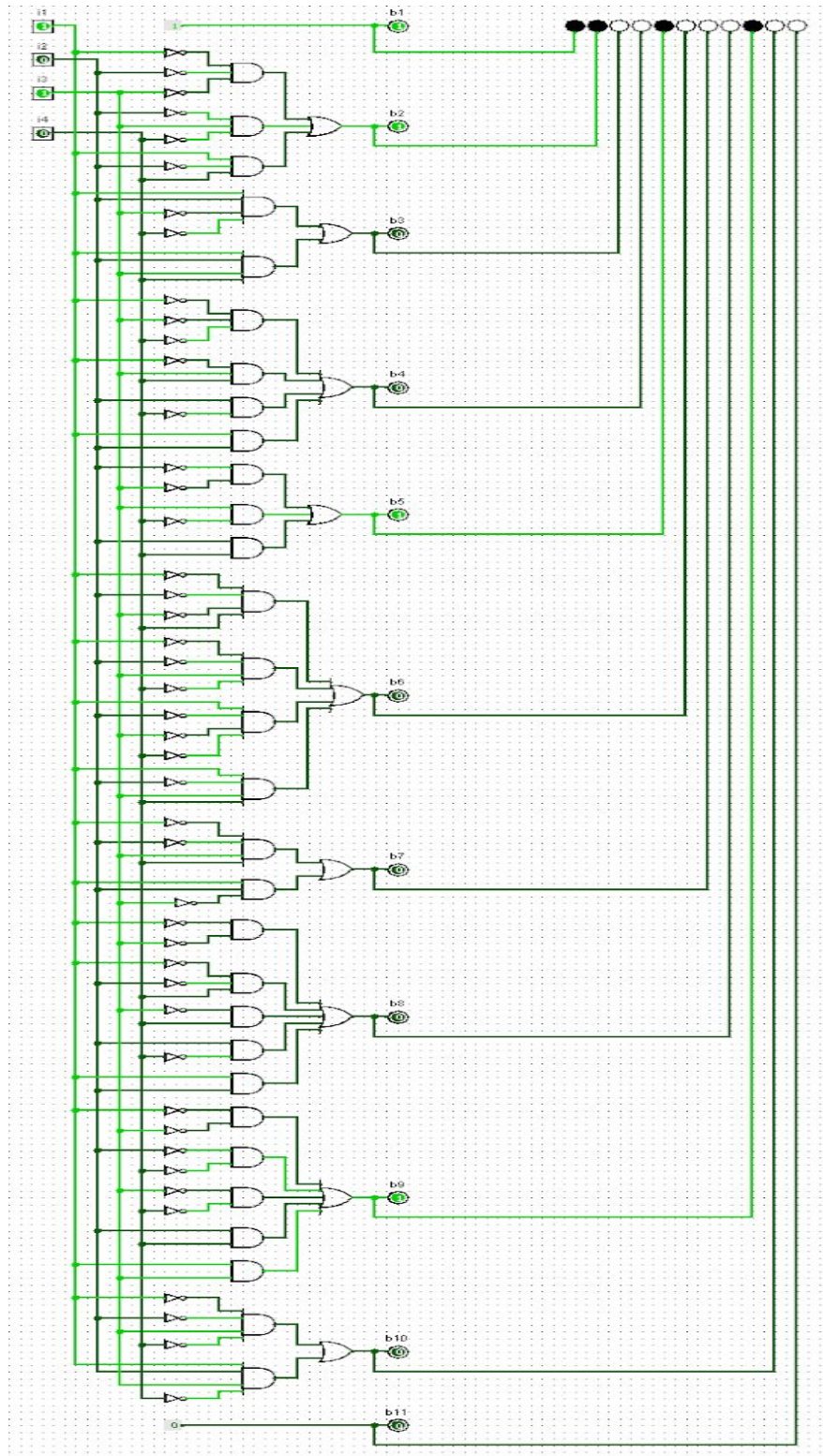
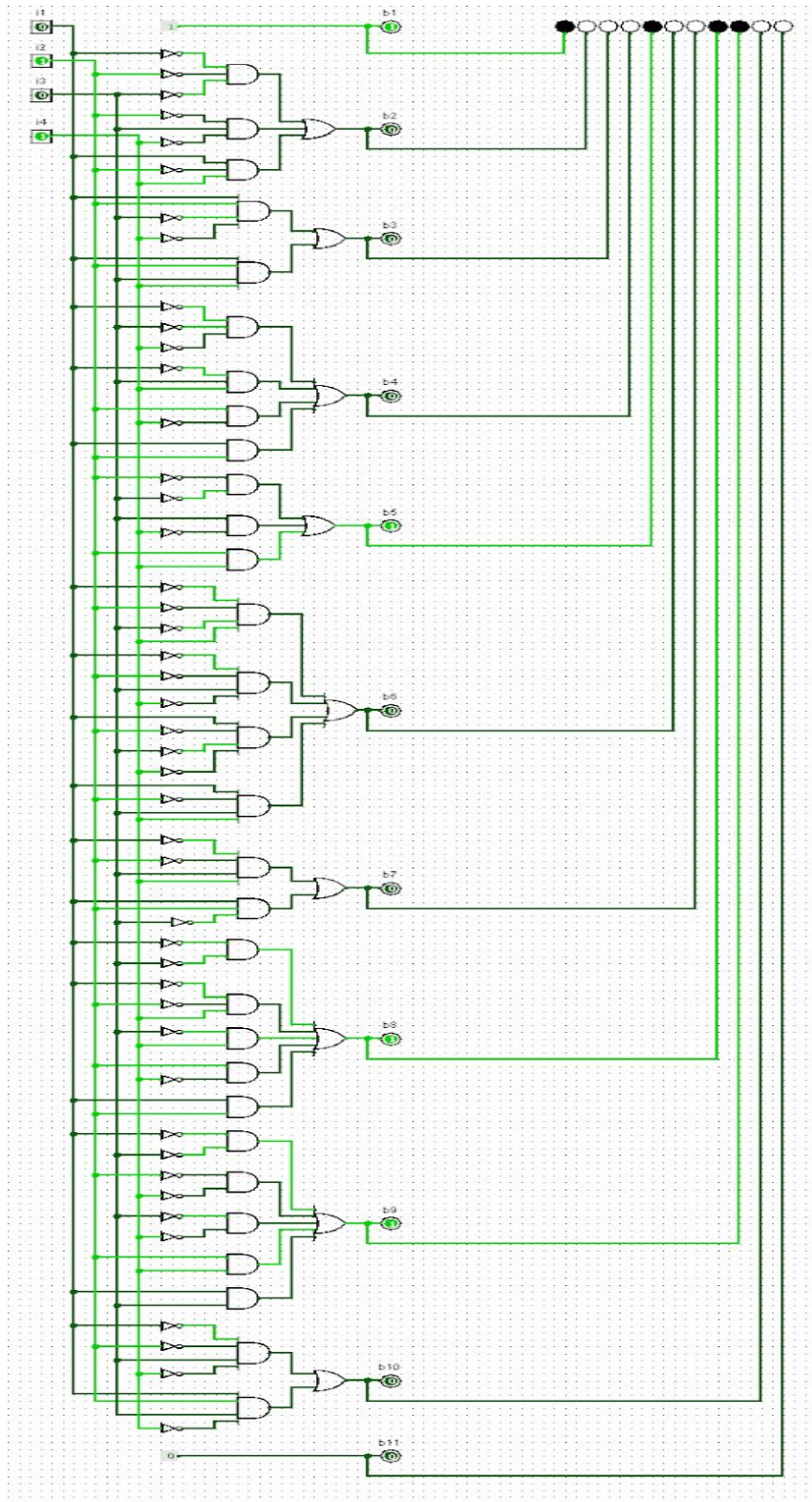


Figure 2.2. Simulation for input 0101.





**Figure 2.3. Simulation for input 1010.**



**Figure 2.4. Simulation for input 1111.**

## **B. Results Analysis**

When representing the barcode output with the given input, a few patterns can be noticed for certain variables. One can see that in all cases, b1 will always have an output of 1 and b11 will always have an output of 0. This is mainly because of the patterning of the Code 128 barcode format. Since the format dictates that the pattern of a segment must always begin with a black bar and end with a white bar, it is a given that the beginning bar must always be at least one module long, and the ending space must also be at least one module wide. This is reflected in the outputs of the two extremes.

This is also reflected conversely. All of the other inputs have several more gates to them because of this fact. With the structure of the barcode segment, the middle segments have more variation in the possible values, thus tending to need more values to consider in order to properly represent them.

Applying k-mapping allowed us to optimize the implementation of the circuit since it does not need to have many levels of gates. This allows for less travel delay, making a faster time for the machine.

## **C. Challenges**

There were several challenges in constructing this project. With the many different applications of combinatorial logic machines already present, it becomes difficult to find a proper application that can still be currently relevant and relatively unique. A significant amount of time was spent researching about topics and combinatorial logic's applicability to them. There

are several kinds of applications that may be too contrived to justify automating using this method. The barcode segment has a definite format to be followed and is not subjectively decided, making its automation applicable in most cases.

Determining the proper input and output was also difficult. There was a major dilemma on how to properly transcribe the barcode output. A barcode alternates between gaps and blocks of varying lengths. While those lengths were different, the order of gaps and blocks were always the same as it starts with a block and alternates, ending with a gap. Thus, one possible way to format it that was considered was by implementing the lengths of the blocks and gaps as six 2-bit integers, with its order representing the size of the respective lengths. This would require a total of twelve outputs. This kind of implementation may be more intuitive to understand if properly displayed at the cost of more output variables and thus more complexity. In the end, the eleven output variable format was chosen because it required less output variables, and thus would be easier to implement.

#### **D. Conclusion**

Barcodes are a very convenient way of storing information in a physical medium for retrieval later. This convenience can be seen in many different places that use the system such as in shops and packaging services. Using combinatorial logic, we can develop a system to convert numbers to a segment, allowing for better automation. Barcode segments follow a pattern that allows us to implement a combinational circuit that can be implemented with minimal resulting delay in its operation.

## **E. Recommendation**

The researchers recommend implementing the full ASCII character set in Code 128 conversion. As this only shows the first sixteen codes, implementing the full 128 character set would prove to be more difficult. This application only works for a single segment. Eventually, a full character set can be used to convert a whole string of characters. Aside from that, implementing the inverse would also be a useful application. On the other hand, it is also recommended to try implementing a Code 128 decoder and also a decoder for other barcode types.

Aside from other possible applications stemming from this, other applications such as encoders, decoders, or multiplexers could be used to simplify the circuit layout. This implementation is the most simple using only AND gates, OR gates and inverters. Using multiplexers could simplify the process, making it easier to convert.

### CONTRIBUTION OF INDIVIDUAL GROUP MEMBER

Name	Contribution
Benedictos, Bianca Joy R	Minimization/Simplification using Boolean Algebra or K-maps, Truth Tables, Problem Formulation, Results Analysis, Recommendations, Presentation Design, Presentation
Tejada, Mikayla	Project Proposal, Problem Description & Formulation, Application, Intended System Operations and Conditions, Definition of Logic Levels and Events, Simulation, Presentation
Teng, Lance Ricco L.	Logisim Design, Idea Formulation, Results Analysis, Problem Description and Formulation, Application, Presentation, Definition of Logic Levels and Events, Challenges & Recommendations

**SOURCES:**

Azalea Software, Inc. (n.d.). Code 128 Barcode FAQ. Retrieved March 15, 2020, from <http://www.azaleabarcodes.com/faq/code-128/#BG2>

Barcode Information & Tips. (n.d.). CODE 128 and GS1-128. Retrieved March 15, 2020, from [https://www.keyence.com/ss/products/auto\\_id/barcode\\_lecture/basic/code128/](https://www.keyence.com/ss/products/auto_id/barcode_lecture/basic/code128/)

Verified Inc. (n.d.). Top Eight Benefits of Barcodes. Retrieved March 15, 2020, from [https://verifiedlabel.com/knowledgecenter/know\\_barcodes.aspx](https://verifiedlabel.com/knowledgecenter/know_barcodes.aspx)