**TEXT TO BRAILLE CONVERTER**

**Team Members:**

1. 221CS206, Alen Basil Thelappilly, alenbasilthelappilly.221cs206@nitk.edu.in, 6282343708

2. 221CS225, Hemanth Kumar P L, hemanthkumarpl.221cs225@nitk.edu.in, 9380256422

3. 221CS238, Rathod Smit Amitkumar, smit.221cs238@nitk.edu.in, 9148719006

**Abstract:**

This project aims to create a Text to Braille Converter using logic gates, integrated circuits (ICs), and advanced design techniques, emphasizing creating a foundational component for various Braille equipment. While the primary motivation for this project is to improve accessibility for the visually impaired by enabling text-to-Braille conversion, we see this component as a versatile building block for various Braille devices and systems, such as notetakers, refreshable Braille displays, and embossers.

**Background**

When accessing printed information, visually impaired people frequently face formidable obstacles, and Braille, with its raised-dot tactile system, provides an essential means of communication.

**Our unique contribution**

Existing assistive technologies have primarily addressed converting Binary-Coded Decimal (BCD) numbers to Braille using logic gates and ICs, leaving a significant gap in text conversion. We plan to fill this void by creating a Text to Braille Converter that uses logic gates, integrated circuits, and advanced design techniques.

**Motivation**

The primary goal of our project is to create a robust and adaptable component capable of efficiently converting textual content into Braille patterns so that visually impaired people have more access to written text.

**Summary**

Our project is developing a Text to Braille Converter using logic gates, integrated circuits, and advanced design techniques to create a versatile foundation for diverse Braille equipment. By focusing on this fundamental component, we hope to improve accessibility, independence, and inclusivity for people with visual impairments in the future, paving the way for many innovative Braille solutions.

**Brief Description:**

**Introduction:**

In our ever-evolving world, technology continues to be a powerful force in bridging gaps and fostering inclusivity. In line with this ethos, we present a project that can potentially transform the lives of individuals with visual impairments by providing them with a more accessible means of understanding and interacting with the written word. The English-to-Braille conversion system, fueled by integrated circuits (ICs), embodies innovation with a purpose - to make information available to all.

**Key Components:**

1. Keyboard Input Interface:

The foundation of our project rests on a user-friendly keyboard input interface. We recognize the importance of simplifying the process for individuals with visual impairments to input English characters for conversion into Braille. This interface stands as a beacon of accessibility, ensuring that they can easily communicate their thoughts and needs. To make the keyboard, we make use of tactile push button switches. Each button represents a character on the keyboard.

2. Encoder:

At the heart of our system lies the encoder, a crucial element that bridges English and Braille. It is designed to perform the intricate task of translating the entered English characters into a compact 5-bit representation. This encoding scheme is carefully crafted to ensure that it effectively captures the essence of the English alphabet, numbers, and various symbols in a format that can be readily transformed into Braille. To make this, we use 5 OR gates.

3. Converter to Braille Output:

Following the encoding process, the converter stage takes the 5-bit representation and performs a remarkable transformation, rendering the information in 6-bit Braille output. This stage is where the magic happens, as it translates the encoded data into tactile patterns that correspond to Braille characters. The converter thus empowers individuals with visual impairments by giving them access to a form of communication that is both efficient and universally recognized. The converter uses a combination of AND, OR, and NOT gates.

**Purpose:**

Our project has a profound and noble purpose: to empower individuals with visual impairments, granting them the independence to access and comprehend written information in Braille. This endeavor aligns seamlessly with the broader mission of making technology more inclusive and accessible, emphasizing equal opportunities for all. It is a testament to our commitment to fostering a more equitable and just society where everyone can engage with the world around them, regardless of their abilities.

**Conclusion:**

In conclusion, the English-to-Braille converter using integrated circuits is not just a technological advancement but a significant endeavor with a noble purpose. As technology advances, we should never forget that its true power lies in its capacity to make the world a better place for everyone, irrespective of their abilities or disabilities.

**Working:**

|  |  |  |  |
| --- | --- | --- | --- |
| Alphabets/Numerals | Input Format | Encoded | Braille  ○ = 0  ● = 1 |
| A | 0000000000000000000000000**1** | 00001 | ●○  ○○  ○○ |
| B | 000000000000000000000000**1**0 | 00010 | ●○  ●○  ○○ |
| C | 00000000000000000000000**1**00 | 00011 | ●●  ○○  ○○ |
| D | 0000000000000000000000**1**000 | 00100 | ●●  ○●  ○○ |
| E | 000000000000000000000**1**0000 | 00101 | ●○  ○●  ○○ |
| F | 00000000000000000000**1**00000 | 00110 | ●●  ●○  ○○ |
| G | 0000000000000000000**1**000000 | 00111 | ●●  ●●  ○○ |
| H | 000000000000000000**1**0000000 | 01000 | ●○  ●●  ○○ |
| I | 00000000000000000**1**00000000 | 01001 | ○●  ●○  ○○ |
| J | 0000000000000000**1**000000000 | 01010 | ○●  ●●  ○○ |
| K | 000000000000000**1**0000000000 | 01011 | ●○  ○○  ●○ |
| L | 00000000000000**1**00000000000 | 01100 | ●○  ●○  ●○ |
| M | 0000000000000**1**000000000000 | 01101 | ●●  ○○  ●○ |
| N | 000000000000**1**0000000000000 | 01110 | ●●  ○●  ●○ |
| O | 00000000000**1**00000000000000 | 01111 | ●○  ○●  ●○ |
| P | 0000000000**1**000000000000000 | 10000 | ●●  ●○  ●○ |
| Q | 000000000**1**0000000000000000 | 10001 | ●●  ●●  ●○ |
| R | 00000000**1**00000000000000000 | 10010 | ●○  ●●  ●○ |
| S | 0000000**1**000000000000000000 | 10011 | ○●  ●○  ●○ |
| T | 000000**1**0000000000000000000 | 10100 | ○●  ●●  ●○ |
| U | 00000**1**00000000000000000000 | 10101 | ●○  ○○  ●● |
| V | 0000**1**000000000000000000000 | 10110 | ●○  ●○  ●● |
| W | 000**1**0000000000000000000000 | 10111 | ○●  ●●  ○● |
| X | 00**1**00000000000000000000000 | 11000 | ●●  ○○  ●● |
| Y | 0**1**000000000000000000000000 | 11001 | ●●  ○●  ●● |
| Z | **1**0000000000000000000000000 | 11010 | ●○  ○●  ●● |

**FLOWCHART**

KEYBOARD INPUT

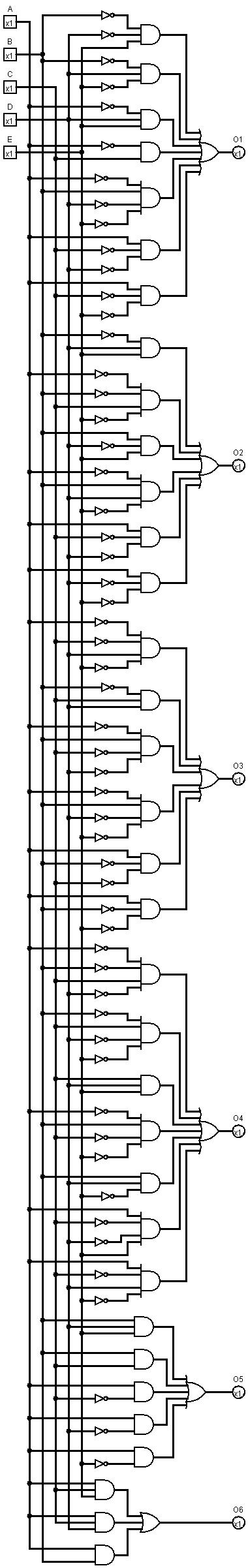
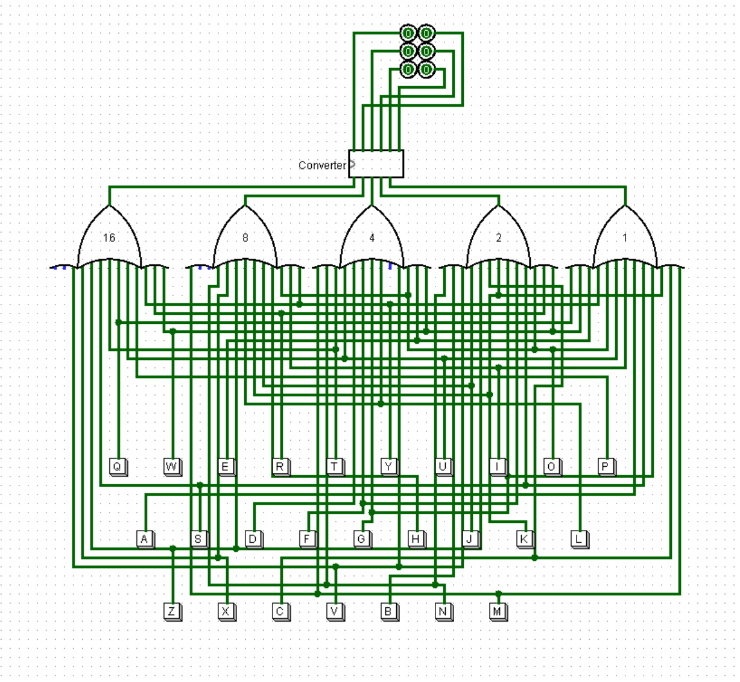
ENCODER

CONVERTER

OUTPUT

**Logisim Circuit diagram:**

**CONVERTER MAIN**



**Verilog code:**

**Testbench:**

module main\_tb;

    reg [25:0]alp;

    wire Of1, Of2, Of3, Of4, Of5, Of6;

    top t (alp, Of1, Of2, Of3, Of4, Of5, Of6);

    initial

    begin

        $dumpfile("Project.vcd");

        $dumpvars(0, main\_tb);

    end

    initial

    begin

        $display("|                   Encoded Aplhabets                     |        Braille        |");

        $display("-----------------------------------------------------------------------------------");

        #10 alp = 26'b00000000000000000000000001;

        $monitor("|               %b                | %b | %b | %b | %b | %b | %b |", alp, Of1, Of2, Of3, Of4, Of5, Of6);

        repeat(25)

        #10 alp = alp \* 26'b00000000000000000000000010;

    end

    initial #300 $finish;

endmodule

**Main:**

module key(alp, O16, O8, O4, O2, O1);

    input [25:0] alp;

    output O16, O8, O4, O2, O1;

    assign O16 = (alp[15] | alp[16] | alp[17] | alp[18] | alp[19] | alp[20] | alp[21] | alp[22] | alp[23] | alp[24] | alp[25]);

    assign O8 = (alp[7] | alp[8] | alp[9] | alp[10] | alp[11] | alp[12] | alp[13] | alp[14] | alp[23] | alp[24] | alp[25]);

    assign O4 = (alp[3] | alp[4] | alp[5] | alp[6] | alp[11] | alp[12] | alp[13] | alp[14] | alp[19] | alp[20] | alp[21] | alp[22]);

    assign O2 = (alp[1] | alp[2] | alp[5] | alp[6] | alp[9] | alp[10] | alp[13] | alp[14] | alp[17] | alp[18] | alp[21] | alp[22] | alp[25]);

    assign O1 = (alp[0] | alp[2] | alp[4] | alp[6] | alp[8] | alp[10] | alp[12] | alp[14] | alp[16] | alp[18] | alp[20] | alp[22] | alp[24]);

endmodule

module converter(A, B, C, D, E, Of1, Of2, Of3, Of4, Of5, Of6);

    input A, B, C, D, E;

    output Of1, Of2, Of3, Of4, Of5, Of6;

    assign Of1 = ((~B) & (~D) & E) | ((~B) & D & (~E)) | ((~A) & D & E) | ((~A) & C) | (B & (~C) & (~D) & ~(E))| (A & (~C) & (~D)) | (A & (~C) & (~E));

    assign Of2 = ((~B) & D & E) | ((~A) & (~B) & C & (~E)) | (B & (~D) & E) | ((~A) & B & D & (~E)) | (A & (~C) & (~D)) | (A & (~D) & (~E));

    assign Of3 = ((~A) & (~C) & D & (~E)) | ((~B) & C & D) | ((~A) & B & (~C) & (~D)) | ((~A) & B & (~D) & (~E)) | (A & (~B) & (~C)) | (A & (~B) & (~E));

    assign Of4 = ((~A) & (~B) & C & (~D)) | ((~B) & C & (~D) & (~E)) | (C & D & E) | ((~A) & B & (~C) & (~E)) | (B & D & (~E)) | (A & (~C) & (~D) & E) | (A & (~C) & D & (~E));

    assign Of5 = (B & D & E) | (B & C) | (A & (~C)) | (A & (~D)) | (A & (~E));

    assign Of6 = (A & C & E) | (A & C & D) | (A & B);

endmodule

module top(alp, Of1, Of2, Of3, Of4, Of5, Of6);

    input [25:0] alp;

    output Of1, Of2, Of3, Of4, Of5, Of6;

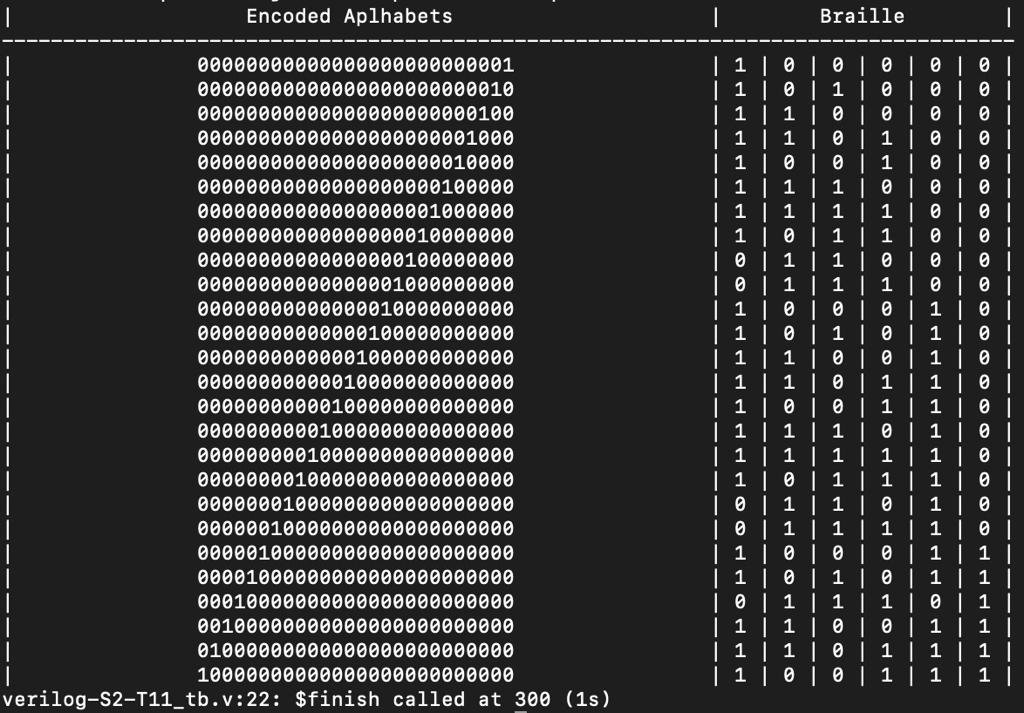
    wire O16, O8, O4, O2, O1;

    key K0(.alp(alp), .O16(O16), .O8(O8), .O4(O4), .O2(O2), .O1(O1));

    converter C0(.A(O16), .B(O8), .C(O4), .D(O2), .E(O1), .Of1(Of1), .Of2(Of2), .Of3(Of3), .Of4(Of4), .Of5(Of5), .Of6(Of6));

endmodule

**OUTPUT**

****

**References:**

Digital Design by Morris Mano

<https://funinnovate.wordpress.com/2014/12/17/bcd-to-braille/>

<https://www.ijresm.com/Vol.3_2020/Vol3_Iss5_May20/IJRESM_V3_I5_249.pdf>

<https://espace.curtin.edu.au/bitstream/handle/20.500.11937/1351/17220_thesis.pdf?sequence=2&isAllowed=y>

https://ieeexplore.ieee.org/abstract/document/4178521

**\*\*\* END \*\*\***