ABSTRACT

The Braille language, which is a tactile writing system used by people who are suffering from visual impairments, has been a blessing for this fast-developing world. However, it can be a great cause of burden and irk to blind people when they have to process numbers through Braille, especially when using services like ATMs and elevators.

With the Braille-to-ASCII Converter, a software tool that is designed to help these individuals, we aim to help bridge the gap between the blind or visually impaired people and the world of written content, and hence improve their quality of life by processing the Braille Numbers and producing an ASCII output. This ASCII output can then be read by standard computer devices and software, making it possible for these individuals to access written information in a format that is more accessible to them.

Furthermore, the Braille-to-ASCII converter can also be useful for people who are not blind but may need to use Braille in certain situations, such as in fields like music notation or mathematics. This tool can be used to convert Braille notation into ASCII text, making it possible for sighted individuals to read and understand Braille notation.

Each number from 0 to 9 in Braille is represented by a 3x2 matrix, where each cell may either be ‘bumped’ or ‘flat’. The input is provided through the means of a 6-bit input (arranged in a 3x2 matrix), with each 1 representing a 'bump' and each 0 representing a ‘flat’. A sequential circuit keeps track of each bump/flat entry given by the user. The circuit processes these 6-bit inputs into the corresponding ASCII output using a Finite State Machine (FSM) model. There is also a flag which toggles if the user has entered an invalid sequence during the entry itself, hence clearing the input stream and preparing the circuit to take in the next set of entries. This helps further increase its accessibility and user engagement. The main circuit can also be paired up with any other feedback system thus expanding the purpose of the device, like in elevators and ATMs, which is beyond the scope of the current project.

CONTENTS

INTRODUCTION

1. What is Braille?

Braille is a system of raised dots that can be read with the fingers by people who are blind or who have low vision. Braille is not a language to be precise, it is a code through which many languages such as English, Hindi, Kannada and many other languages are understood. Braille has been a lifesaver to the blind, enabling them to read and write, removing the barrier of the written media for the blind. It has empowered them to get educated and become more expressive of their thoughts and opinions.

The Braille system was invented in 1824 by Louis Braille, a French educator who was himself blind. He developed the system as a way to help blind students to read and write, and it quickly gained widespread acceptance as a means of literacy for blind people.

Braille is based on a 6-dot cell, and each dot in the cell can be ‘bumped’ (indicating a 1) or ‘flat’ (indicating a 0). Each letter of the alphabet and each number is represented by a unique combination of dots in the 6-dot cell. For example, the letter "A" is represented by the combination of dots 1-2, the letter "B" is represented by the combination of dots 1-2-3, and so on. The combination of dots can be used to represent contractions and punctuations as well.

Braille is written using a special type of device called a Braille writer, which is similar to a typewriter. The writer has six keys, each of which corresponds to one of the dots in the Braille cell. To write a letter, the user raises the dots that correspond to that letter and then presses the paper against the keys to make an impression of the dots. Braille can also be produced using a Braille embosser, which is a device that can print Braille on to special paper. This is useful for creating Braille documents in large quantities, such as Braille books and other materials.

2. Drawbacks of Braille :

Braille has aided millions of blind souls world wide. However, it’s not without its own challenges.

(a) Cost: Braille devices, such as Braille writers and Braille embossers, can be expensive, making them cost-prohibitive for some individuals and organisations.
(b) Space: Braille takes up significantly more space than print, which can be a problem when trying to store or transport large amounts of Braille materials.
(c) Limited availability: Braille materials are often difficult to find, and there is a limited selection of Braille books, newspapers, and other materials available.
(d) Time-consuming: Transcribing materials into Braille can be time-consuming and labor-intensive, which can be a barrier for organisations and individuals who want to make written materials available in Braille.
(e) Stigma: some people still view Braille as something that is only for the blind and not needed by the sighted, so they may not be as likely to use it.
(f) Limited access to technology: Not all blind people have access to technology such as Braille displays, Braille note-takers, and Braille printers.
(g) Limited accessibility to certain subjects: some subjects like mathematics and music have notations that are not easily represented in Braille, making it difficult for blind students to access certain subjects.

It's worth noting that despite these disadvantages, Braille remains an important tool for many individuals who are blind or visually impaired, and technology advancements in recent years have helped to mitigate some of these disadvantages.

DESIGN

In this project, we have made use of the Mealy Finite State Machine Model to design the states of the machine, and its outputs. The circuit model accepts a stream of input bits, and after every 6 bits forming a valid Braille sequence, it displays the ASCII value corresponding to the input. If an invalid sequence is detected, the circuit model will abrupt the flow of the input stream and will reset to the initial state, toggling the invalid flag to 1.

To start using the converter, we first reset the machine to erase any bits stored from the previous input stream, and then start passing in the input stream. The circuit model designed shows NULL (ASCII value 0) until a valid input sequence is detected. If any other output has to be shown instead of NULL, it can be configured in the external display used.

The Braille input stream must be in the order as shown in Fig-1.

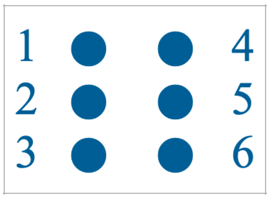


Fig-1: Order of the input stream in which data should be passed into the circuit

where, 1 represents the first bit which must be entered into the input stream, and 6 represents the last bit.

The Braille input stream for each of the numbers 0 to 9 (in decimal number system) is as shown in Table-1.

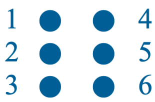
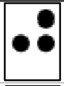
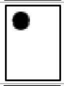
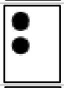
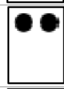
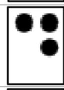
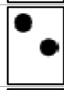
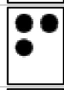
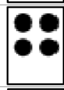
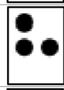
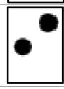
	B1	B2	B3	B4	B5	B6	Braille Configuration
0	0	1	0	1	1	0	
1	1	0	0	0	0	0	
2	1	1	0	0	0	0	
3	1	0	0	1	0	0	
4	1	0	0	1	1	0	
5	1	0	0	0	0	1	
6	1	1	0	1	0	0	
7	1	1	0	1	1	0	
8	1	1	0	0	1	0	
9	0	1	0	1	0	0	

Table-1: Braille Input Stream for each Decimal Number from 0 to 9

The ASCII values of numbers 0 to 9 in the Decimal Number System is shown in Table-2.

Decimal Number	ASCII Value
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57

Table-2: ASCII values for Decimal numbers from 0 to 9

ASM chart to generate output according to the above input stream is shown in Fig-2.

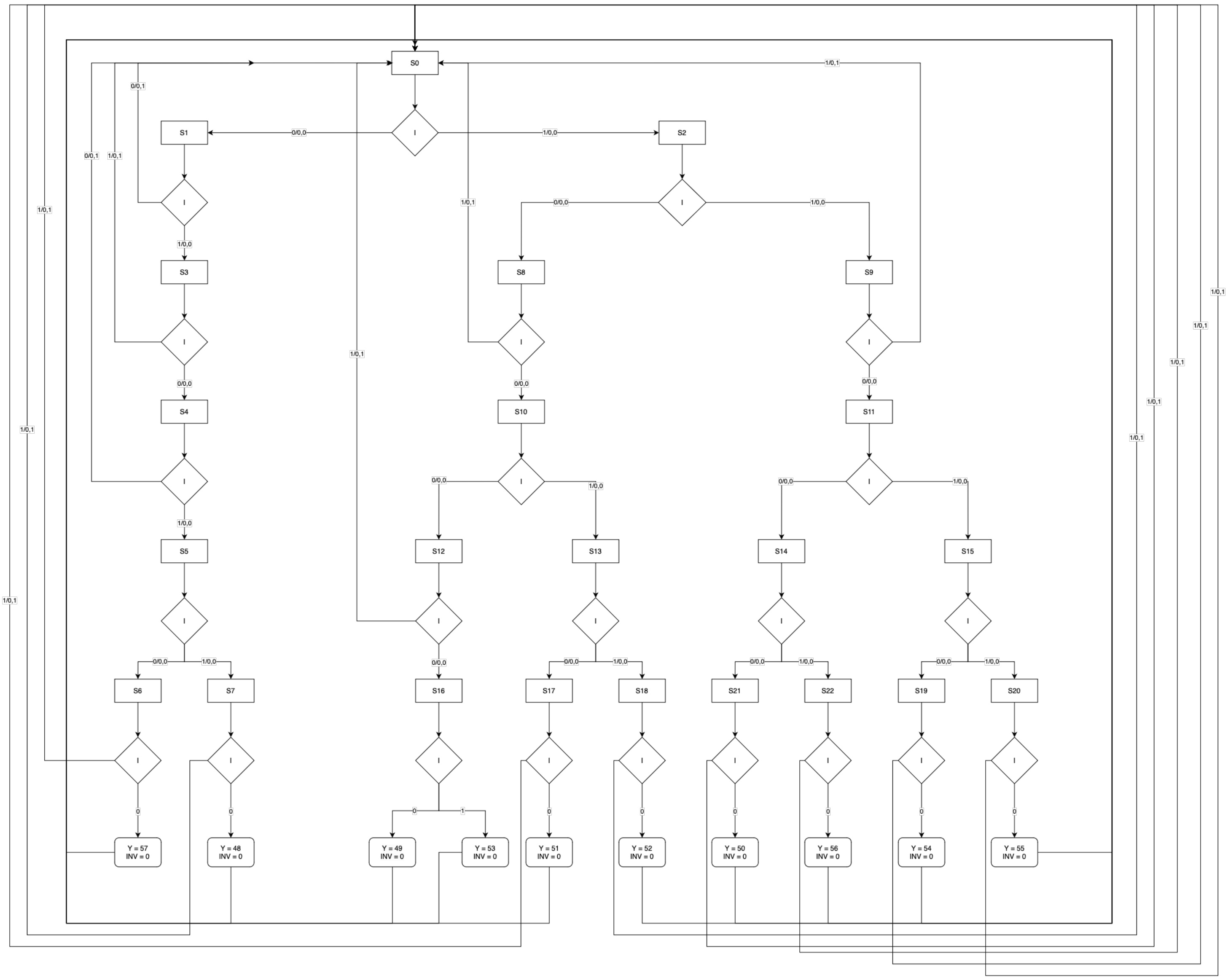


Fig-2: ASM Chart for the Circuit Model

The corresponding State Table is shown in Table-3. (The ASCII value of NULL is 0)

Present State	Next State		Output			
			ASCII Value		Invalid Flag	
	x=0	x=1	x=0	x=1	x=0	x=1
S0	S1	S2	NULL (0)	NULL (0)	0	0
S1	S0	S3	NULL (0)	NULL (0)	1	0
S2	S8	S9	NULL (0)	NULL (0)	0	0
S3	S4	S0	NULL (0)	NULL (0)	0	1
S4	S0	S5	NULL (0)	NULL (0)	1	0
S5	S6	S7	NULL (0)	NULL (0)	0	0
S6	S0	S0	9 (57)	NULL (0)	0	1
S7	S0	S0	0 (48)	NULL (0)	0	1
S8	S10	S0	NULL (0)	NULL (0)	0	1
S9	S11	S0	NULL (0)	NULL (0)	0	1
S10	S12	S13	NULL (0)	NULL (0)	0	0
S11	S14	S15	NULL (0)	NULL (0)	0	0
S12	S16	S0	NULL (0)	NULL (0)	0	1
S13	S17	S18	NULL (0)	NULL (0)	0	0
S14	S21	S22	NULL (0)	NULL (0)	0	0
S15	S19	S20	NULL (0)	NULL (0)	0	0
S16	S0	S0	1 (49)	5 (53)	0	0
S17	S0	S0	3 (51)	NULL (0)	0	1
S18	S0	S0	4 (52)	NULL (0)	0	1
S19	S0	S0	6 (54)	NULL (0)	0	1
S20	S0	S0	7 (55)	NULL (0)	0	1
S21	S0	S0	2 (50)	NULL (0)	0	1
S22	S0	S0	8 (56)	NULL (0)	0	1

Table-3: State Table for the above designed ASM chart

IMPLEMENTATION

We have designed the Braille-to-ASCII converter using Verilog, which is a Hardware Description Language (HDL) used to model, design, simulate, and document electronic systems. Attached below is the code along with test-benches and corresponding timing diagrams to demonstrate the working of the modelled circuit by using sample inputs.

Verilog Code:

```verilog
module brailleToAscii(I,CLK,R,Y,INV);
    input CLK;
    input R;
    input I;

    output [7:0] Y;
    output reg INV;

    reg [4:0] ps;
    reg [4:0] ns;
    reg [7:0] op;

    parameter [4:0] s0 = 0;
    parameter [4:0] s1 = 1;
    parameter [4:0] s2 = 2;
    parameter [4:0] s3 = 3;
    parameter [4:0] s4 = 4;
    parameter [4:0] s5 = 5;
    parameter [4:0] s6 = 6;
    parameter [4:0] s7 = 7;
    parameter [4:0] s8 = 8;
    parameter [4:0] s9 = 9;
    parameter [4:0] s10 = 10;
    parameter [4:0] s11 = 11;
    parameter [4:0] s12 = 12;
    parameter [4:0] s13 = 13;
    parameter [4:0] s14 = 14;
    parameter [4:0] s15 = 15;
    parameter [4:0] s16 = 16;
    parameter [4:0] s17 = 17;
    parameter [4:0] s18 = 18;
    parameter [4:0] s19 = 19;
    parameter [4:0] s20 = 20;
    parameter [4:0] s21 = 21;
    parameter [4:0] s22 = 22;

```

```
always @(posedge CLK, R, I)
    begin
        INV = 0;

        if(R == 0) ps = s0;

        else begin
            case(ps)
                s0: begin
                    if(I) begin
                        ns = s2;
                        op = "";
                    end
                    else begin
                        ns = s1;
                        op = "";
                    end
                end
                s1: begin
                    if(I) begin
                        ns = s3;
                        op = "";
                    end
                    else begin
                        ns = s0;
                        op = "";
                        INV = 1;
                    end
                end
                s2: begin
                    if(I) begin
                        ns = s9;
                        op = "";
                    end
                    else begin
                        ns = s8;
                        op = "";
                    end
                end
                s3: begin
                    if(I) begin
                        ns = s0;
                        op = "";
                        INV = 1;
                    end
                    else begin
                        ns = s4;
                        op = "";
                    end
                end
```

```
                s4: begin
                    if(I) begin
                        ns = s5;
                        op = "";
                    end
                    else begin
                        ns = s0;
                        op = "";
                        INV = 1;
                    end
                end
                s5: begin
                    if(I) begin
                        ns = s7;
                        op = "";
                    end
                    else begin
                        ns = s6;
                        op = "";
                    end
                end
                s6: begin
                    if(I) begin
                        ns = s0;
                        op = "";
                        INV = 1;
                    end
                    else begin
                        ns = s0;
                        op = "9";
                    end
                end
                s7: begin
                    if(I) begin
                        ns = s0;
                        op = "";
                        INV = 1;
                    end
                    else begin
                        ns = s0;
                        op = "0";
                    end
                end
                s8: begin
                    if(I) begin
                        ns = s0;
                        op = "";
                        INV = 1;
                    end
                    else begin
                        ns = s10;
                        op = "";
                    end
                end

```

```
s9: begin
    if(I) begin
        ns = s0;
        op = "";
        INV = 1;
    end
    else begin
        ns = s11;
        op = "";
    end
end
s10: begin
    if(I) begin
        ns = s13;
        op = "";
    end
    else begin
        ns = s12;
        op = "";
    end
end
s11: begin
    if(I) begin
        ns = s15;
        op = "";
    end
    else begin
        ns = s14;
        op = "";
    end
end
s12: begin
    if(I) begin
        ns = s0;
        op = "";
        INV = 1;
    end
    else begin
        ns = s16;
        op = "";
    end
end
s13: begin
    if(I) begin
        ns = s18;
        op = "";
    end
    else begin
        ns = s17;
        op = "";
    end
end
```

```
s14: begin
    if(I) begin
        ns = s22;
        op = "";
    end
    else begin
        ns = s21;
        op = "";
    end
end
s15: begin
    if(I) begin
        ns = s20;
        op = "";
    end
    else begin
        ns = s19;
        op = "";
    end
end
s16: begin
    if(I) begin
        ns = s0;
        op = "5";
    end
    else begin
        ns = s0;
        op = "1";
    end
end
s17: begin
    if(I) begin
        ns = s0;
        op = "";
        INV = 1;
    end
    else begin
        ns = s0;
        op = "3";
    end
end
s18: begin
    if(I) begin
        ns = s0;
        op = "";
        INV = 1;
    end
    else begin
        ns = s0;
        op = "4";
    end
end
```

```
                s19: begin
                    if(I) begin
                        ns = s0;
                        op = "";
                        INV = 1;
                    end
                    else begin
                        ns = s0;
                        op = "6";
                    end
                end
                s20: begin
                    if(I) begin
                        ns = s0;
                        op = "";
                        INV = 1;
                    end
                    else begin
                        ns = s0;
                        op = "7";
                    end
                end
                s21: begin
                    if(I) begin
                        ns = s0;
                        op = "";
                        INV = 1;
                    end
                    else begin
                        ns = s0;
                        op = "2";
                    end
                end
                s22: begin
                    if(I) begin
                        ns = s0;
                        op = "";
                        INV = 1;
                    end
                    else begin
                        ns = s0;
                        op = "8";
                    end
                end
            endcase

            ps = ns;
        end
    end

    assign Y = op;
endmodule
```

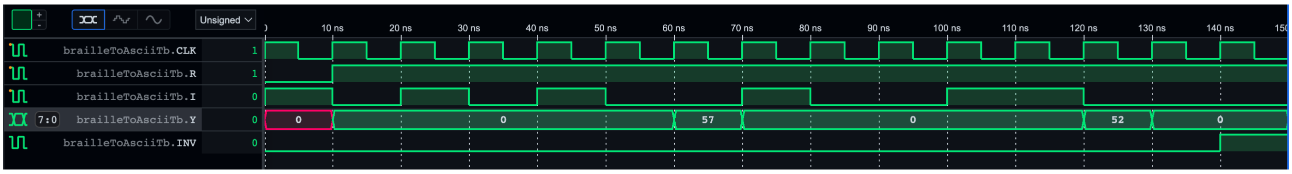
Case - 1:

Input Stream given to get ASCII values of decimal numbers 9 and 4 :

Test-bench code :

```verilog
`timescale 1ns/1ns
`include "braille_to_ascii.v"

module brailleToAsciiTb();
    reg CLK;
    reg R;
    reg I;

    wire [7:0] Y;
    wire INV;

    brailleToAscii BTA (I,CLK,R,Y,INV);

    initial
    begin
        $dumpfile("braille_to_ascii.vcd");
        $dumpvars();
    end

    always
        #5 CLK = ~CLK;

    initial
    begin
        CLK = 1;

        R = 0; I = 1; #10;

        //Input Stream for 9 (57)
        R = 1; I = 0; #10;
        R = 1; I = 1; #10;
        R = 1; I = 0; #10;
        R = 1; I = 1; #10;
        R = 1; I = 0; #10;
        R = 1; I = 0; #10;

        //Input Stream for 4 (52)
        R = 1; I = 1; #10;
        R = 1; I = 0; #10;
        R = 1; I = 0; #10;
        R = 1; I = 1; #10;
        R = 1; I = 1; #10;
        R = 1; I = 0; #10;
    end

    initial
    begin
        $display("Test Completed!!");
        #150 $finish;
    end
endmodule
```

Corresponding Timing Diagram :



Explanation :

1. At 0ns, at the first positive edge of the input clock, the circuit is reset, and hence the output (Y) is at don't care state (indicated with RED colour in the timing diagram).

2. From the next positive edge, till 5 more positive edges of the input clock, input stream is given in the sequence '010100'. As soon as the 6th bit is input, at 60ns, the output (Y) shows '57' which is the ASCII value of 9. The machine goes back to the initial state after showing 57, and accepts the next stream of input bits.

3. Similarly, after feeding the input bits as '100110' in the input stream, at 120ns, the output (Y) shows '52' which is the ASCII value of 4.

The invalid flag (INV) never becomes '1' because there was no invalid sequence of bits detected.

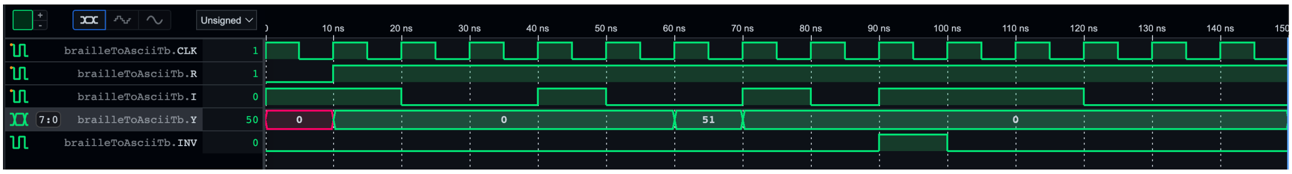
Case - 2 :

Input Stream contains one valid sequence to give ASCII value of decimal number 3 and one invalid stream

Test-bench Code:

```
`timescale 1ns/1ns
`include "braille_to_ascii.v"

module brailleToAsciiTb();
    reg CLK;
    reg R;
    reg I;

    wire [7:0] Y;
    wire INV;

    brailleToAscii BTA (I,CLK,R,Y,INV);

    initial
    begin
        $dumpfile("braille_to_ascii.vcd");
        $dumpvars();
    end

    always
        #5 CLK = ~CLK;

    initial
    begin
        CLK = 1;

        R = 0; I = 1; #10;

        //Input Stream for 3 (51)
        R = 1; I = 1; #10;
        R = 1; I = 0; #10;
        R = 1; I = 0; #10;
        R = 1; I = 1; #10;
        R = 1; I = 0; #10;
        R = 1; I = 0; #10;

        //Input Stream for INV = 1
        R = 1; I = 1; #10;
        R = 1; I = 0; #10;
        R = 1; I = 1; #10;
        R = 1; I = 1; #10;
        R = 1; I = 1; #10;
        R = 1; I = 0; #10;
    end

    initial
    begin
        $display("Test Completed!!");
        #150 $finish;
    end
endmodule
```

Corresponding Timing Diagram :



Explanation :

1. At 0ns, at the first positive edge of the input clock, the circuit is reset, and hence the output (Y) is at don't care state (indicated with RED colour in the timing diagram).

2. From the next positive edge, till 5 more positive edges of the input clock, input stream is given in the sequence '100100'. As soon as the 6th bit is input, at 60ns, the output (Y) shows '51' which is the ASCII value of 3. The machine goes back to the initial state after showing 51, the invalid flag (INV) being 0 itself (as the input sequence was valid), and accepts the next stream of input bits.

3. Now, the input stream fed to the circuit is '101110'. This time, as soon as the sequence becomes '101', the invalid flag (INV) becomes 1, and the machine is reset to its initial state, as there are no valid sequences that have '101' as the first three bits (refer Table-1), and the output (Y) remains 0 (ASCII value of NULL).

RESULTS AND DISCUSSIONS

The results obtained on the completion of the Braille-to-ASCII converter model using Verilog are:

1. The circuit model was successfully able to interpret valid input data streams into their corresponding ASCII values.
2. The circuit model was also able to successfully detect invalid input data streams during the input itself, and was able to notify the user that the input stream sequence is invalid (by toggling the invalid flag to 1 as shown in the above timing diagram), hence resetting itself to the initial state and accepted the next stream of input data bits.
3. After testing the circuit model for various input data streams, we can safely conclude that the conversion is accurate and reliable for use by the users.

For now, the converter only supports conversion of decimal numbers 0 to 9, but we will put in effort into this converter to extend it's conversion capability to alphabets and other special characters, hence making it more versatile and useful for a wider range of users.

It could also then be considered to integrate the converter with other technologies such as smart devices, wearables, and other assistive technology. This would allow users to easily convert text on the go and make it more accessible for them. We will also try to improve the converter in-order to handle more complex braille codes such as mathematical equations and music notations.

In addition, the converter can be integrated with technologies like screen readers, speech synthesisers and so on, this will make it easier for students to read and write in Braille, which can in turn help to improve their literacy skills.

CONCLUSION

In conclusion, the Braille to ASCII converter project has successfully demonstrated the feasibility of using a software tool to increase the involvement of blind and visually impaired people in written assets of this world. We have tried our best to make the Braille-to-ASCII converter an effective tool for accurately converting Braille inputs into ASCII outputs, and tried to ensure that it is user-friendly and easy to use. The system's error detection also helped to further increase its accessibility and user-engagement.

The Braille-to-ASCII converter is a valuable tool that can play a key role in helping to bridge the digital divide and improve the quality of life for individuals who are blind or visually impaired. It can also be used in educational and professional settings to help blind students and employees to access written material and to make them more independent.

Overall, this small project has demonstrated the potential of the Braille to ASCII converter to improve accessibility and literacy for individuals who are blind or visually impaired and has highlighted the importance of continued research in this area.

REFERENCES

1. Books :
    (a) Speech, Text and Braille Conversion Technology | SpringerLink
    (b) https://www.afb.org/blindness-and-low-vision/braille/what-braille
    (c) Magnetic Tactile Sensor for Braille Reading | IEEE Journals & Magazine | IEEE Xplore

2. References :
    (a) Digital and Computer Design – M M Morris Mano