Braille Translator

Medasani Vineetha(19116041), Nitika Gupta(19116044), Pallavi Singh(19116045) and Tanya Rampal(19116080)

***Abstract*— The visually impaired form an integral part of our society. Braille, as a special written method of communication for the blind, has been globally accepted for years. It gives blind people another chance to learn and communicate more efficiently with the rest of the world. With the development of electronic technology, Braille turned out to be well suited to computer-aided production because of its coded forms.**

The project's objective is to design and develop a Braille System for the visually impaired individuals that enable them to interact and communicate. Different from most commercial software-based translators, the circuit presented in this paper is able to carry-out text-to-Braille translation in hardware. In this project, a real-time integrated solution of hardware and software is developed to help the visually impaired people all across the globe to support the communication and interaction of such individuals, thus fostering their independence.

The application of VERILOG is introduced to explain how the translating code can be transformed to hardware. Using a XILINX VIVADO development platform, the code for text-to-Braille translation is simulated and the structure of the translator is described hierarchically.

***Keywords*—Braille Translation, Code Reconfiguration, VERILOG Programming, ASCII numbers**

I. Introduction

Braille is a non-verbal language mainly used by blind or visually impaired people. Braille was invented by a blind Frenchman, Louis Braille, in 1829. Braille is comprised of a rectangular six dot cell on its end, with up to 63 possible combinations using one or more of the six dots. Braille language can be understood by analyzing and reading of different patterns by sensing different input signals happens in the hardware level.

According to WHO, 39 million people are blind. Braille literacy helps to get education who are suffering from blind issues or partial blind issues.

VERILOG is a hardware description language mainly used to design electronic systems. It is mainly used to design and verify digital circuits. It is architecture development type and focus on parallel execution. The processing power is time limited and has very high speed.

AIM: Our aim is to convert a given sentence (which is in lowercase form and has a fixed number of letters) into Braille format and vice-versa using VERILOG code.

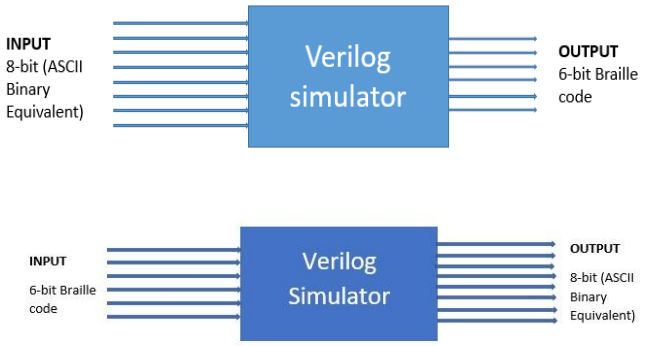


Fig: Overview of the Project

The code first converts the given sentence into binary form by associating each letter (in ASCII form) with its BRAILLE equivalent binary digit and storing it in an output variable, which is connected to a keypad. Then in the second part, braille code is inputted by typing on the keypad and is shown as text is the output, and the ASCII value is shown on 7 segment LED display.

MOTIVATION: Braille is a critical part of blind education and their culture. There are 39 million visually impaired people in the world and they are challenged in their daily lives by living in a sighted world.
The difficulties faced by the blind people, their inability to see the beautiful world around them. They also face many problems when it comes to accessing the digital data. When it comes to using basic features of a phone like text messages, emails etc, and the blind people have a tendency to become socially excluded. Hence in our project, we have focused on technology-based solutions to blind communications which are specifically designed to facilitate distant communications. We also did this project to increase our knowledge in hardware and software for the betterment of society.

RESEARCH PAPERS:

1. Murray Iain and Cesar Ortega- Sanchez, "Text-to-Braille Translator in a Chip" ICECE 2006.
2. K.Basavaraja, Pujari Apoorva, N.Karthika, Holla Ravishankar, "Braille Code Conversion Using VERILOG" May 2020.
3. X Zhang, "Hardware-Based Text-to-Braille Translation" 2007.

II. ARCHITECTURE OF THE DESIGN

To convert a given sentence(which is in lowercase form and has a fixed number of letters) into braille format, The code first converts the given sentence into binary form by associating each letter(in ascii form) with its braille equivalent binary digit and storing it in an output variable, which is connected to a keypad(fig 2). This is done according to table no 1. When it converts a given braille code to a sentence, braille is input through the keypad and is converted back to each letter using table 1.

S. No.	Alphabets	ASCII Values	Braille code
1	a	97	100000
2	b	98	110000
3	c	99	100001
4	d	100	100011
5	e	101	100010
6	f	102	110001
7	g	103	110011
8	h	104	110010
9	i	105	010100
10	j	106	010001
11	k	107	010011
12	l	108	101000
13	m	109	111000
14	n	110	101001
15	o	111	101010
16	p	112	101011
17	q	113	111011
18	r	114	111010
19	s	115	011001
20	t	116	011011
21	u	117	101100
22	v	118	111100
23	w	119	010111
24	x	120	101101
25	y	121	101111
26	z	122	101110

Table 1: ASCII value of letters and their Braille binary number

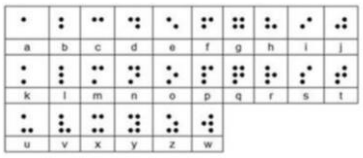


Fig 1: Braille format

Fig 2: Output Keypad

Since each input letter in the sentence corresponds to 8 bits, we consider an input port of n*8 bits vector(sen) to store the sentence. Each braille letter has 6 bits, so we consider an output port of n*6 bits vector(out1) as a translation of all the n characters in the input sentence. Similarly, for braille to text translation, we have a braille input of n*6 bits vector(br) and sentence output vector of n*8 bits(out2). Out3 is a vector that stores the 7 bit led display of each letter of out2.

When converting from text to braille, the output reg type vector(out1) will have 6 bits corresponding to each translated letter from the input, thus a storage element(latch) is formed to pass on these values through a buffer. The translation is done with the help of 'case' statement in Verilog which allows us to assign a 6 bit braille letter to one output segment based on the ascii value of the each letter from the input sentence(blank spaces are given 0 value, ie a pause on the keypad). The case statement is mapped into multiple multiplexers as shown in the Synthesis design diagram. Similarly, when converting from braille to text, a case statement is used to store the letters in the output variable, and out3 stores the led 7 bit values through a case statement.

In the hardware form, the latch(fig 3) allows the current 6 bit braille letter to be shown on the keypad, while the next letter in the sentence is being processed by the circuit. The given keypad consists of pins numbered 1-6. Each pin depicts a button that will pop up when it's high(its value is 1) and remain at surface level when it's low(it's value is 0), thus enabling the visually able person to understand the translated text. Through this keypad, he/she can input the braille code which is then converted in to text

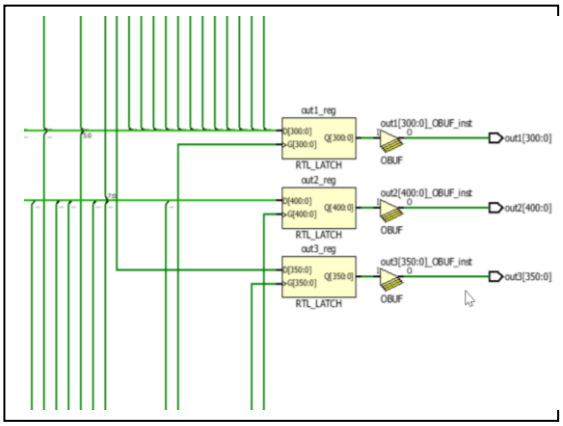
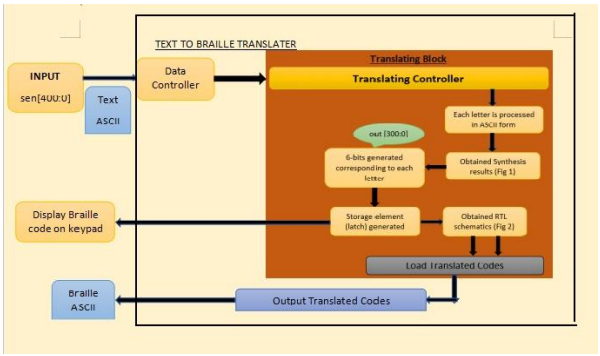
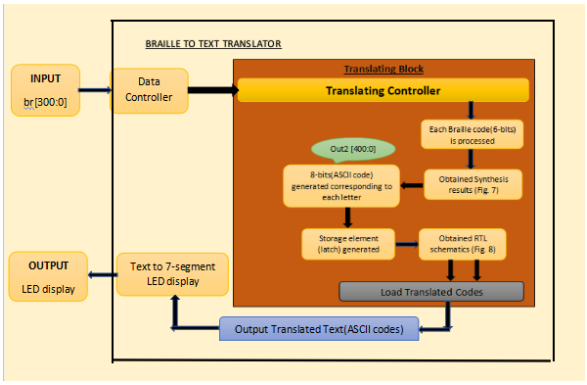


Fig 3: Output vectors(out1 and out2 taken from RTL schem.)

for our convenience.

BLOCK DIAGRAM:





III. CHALLENGES FACED DURING IMPLEMENTATION

1. Lack of hardware to implement our project and see physical output.
2. Since ports cannot be unpacked arrays, vectors were used to store input/output results, which can cause storage problems for very long words.
3. Correct output was not obtained until after several tries.
4. Simulation and synthesis results were sketchy at first and were showing errors.
5. There was lack of communication between team members due to lockdown.
6. Global pandemic created distractions and interference in our work.

IV. SIMULATION RESULTS

SIMULATION: Simulation is the execution of a model in the software environment. It is basically the process of using a simulation software (simulator) to verify the functional correctness of a digital design that is modelled using a HDL (hardware description language) like Verilog.

The test bench is used to simulate our design by specifying the inputs into the system.

Given below is the test bench of our Braille translator Verilog code(fig 4).

```
//testbench for braille translator

`timescale 10ns/1ns
module t_braille_translator;
    parameter n=50;
    reg[n*7:0] sen; //given sentence to be converted to braille(in lowercase, no special characters)
    reg [n*6:0] br; //given braille to be converted to text
    wire[n*6:0] out1;   //braille output
    wire[n*8:0] out2;   //sentence output
    wire[n*7:0] out3;   //7 segment display
    integer i;

    braille_translator b1(sen, out1, br, out2,out3);

    initial
        begin

            sen=0;
            br=0;

            #5  sen="pauls laptop is silver in colour but small in size";
                $display("%s",sen);

            #5  $display(" Braille translation(in binary form) : ");
                for(i=n-1;i>=0;i=i-1)
                    $display(" %b ",out1[i*6+:6]);

            #5  $display("\n");
                sen="he sells sea shells on the sea shore to earn money";
                $display("%s",sen);

            #5  $display(" Braille translation(in binary form) : ");
                for(i=n-1;i>=0;i=i-1)
                    $display(" %b ",out1[i*6+:6]);

            #5  $display("\n");
                sen=0;
                br=301'b110010100010000000011001100010101000101000011001000000011001100010100000000000110011100101000101

            #5  $display("\n");
                $finish;

        end
endmodule
```

Fig 4: Testbench code

Fig. 5 shows the output that the simulator provides after the execution of the above test bench

The output shows 6-bit braille code corresponding to each translated letter in the input sentence (for text to braille translator part of the code) and it shows the text (in lower-case) corresponding to the input braille code(for braille to text translator part of the code).

Fig 5: Output of testbench

SIMULATION WAVEFORM: Simulation also allows us to view the timing diagram of related signals to understand how the design description in Verilog actually behaves.

It allows us to dump design and test bench signals into a waveform (Fig. 2) that can be graphically represented to analyze and debug functionality of the RTL design.

The waveform below shows the progress of each signal with respect to time. They are ordered according to the simulation (as mentioned in the test bench). The time values shown are of the order of nanoseconds.

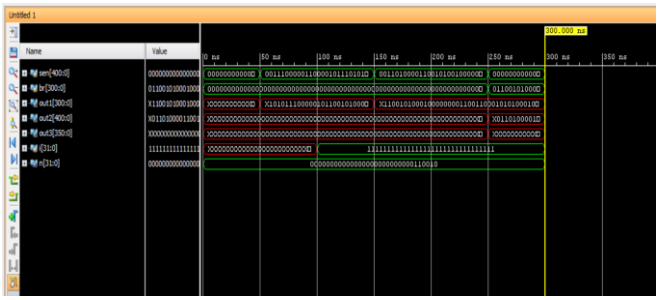


Fig 6: Waveform

SYNTHESIS: Synthesis is a process by which an abstract specification of desired circuit behaviour, typically at register transfer level (RTL), is turned into a design implementation in terms of logic gates, typically by a computer program called a synthesis tool (Synthesizer). Synthesizer performs architectural optimizations, then creates an internal representation of the design. The output of a synthesizer is a gate-level Verilog description.

Fig.7 shows the gate-level diagram produced by the Synthesis tool.

RTL SCHEMATIC: RTL (Register Transfer Level) schematic is generated after the HDL synthesis phase of the synthesis process. It describes how data is transformed as it is passed from register to register. The transforming of the data is performed by the combinational logic that exists between the registers. Fig.8 shows the RTL schematic diagram which shows three **latches**.

The first latch(text to braille) is generated to pass the 6 bit values corresponding to each translated later in the input sentence through a buffer. The latch basically shows the translated letter on the keypad while the next letter is being processed. The second latch(braile to text) is generated to pass 8 bit values(ASCII values of text letters) corresponding to each 6 bit braille code in the input through a buffer. The third latch has the values of the 7 bit segment values of all letters of out 2.

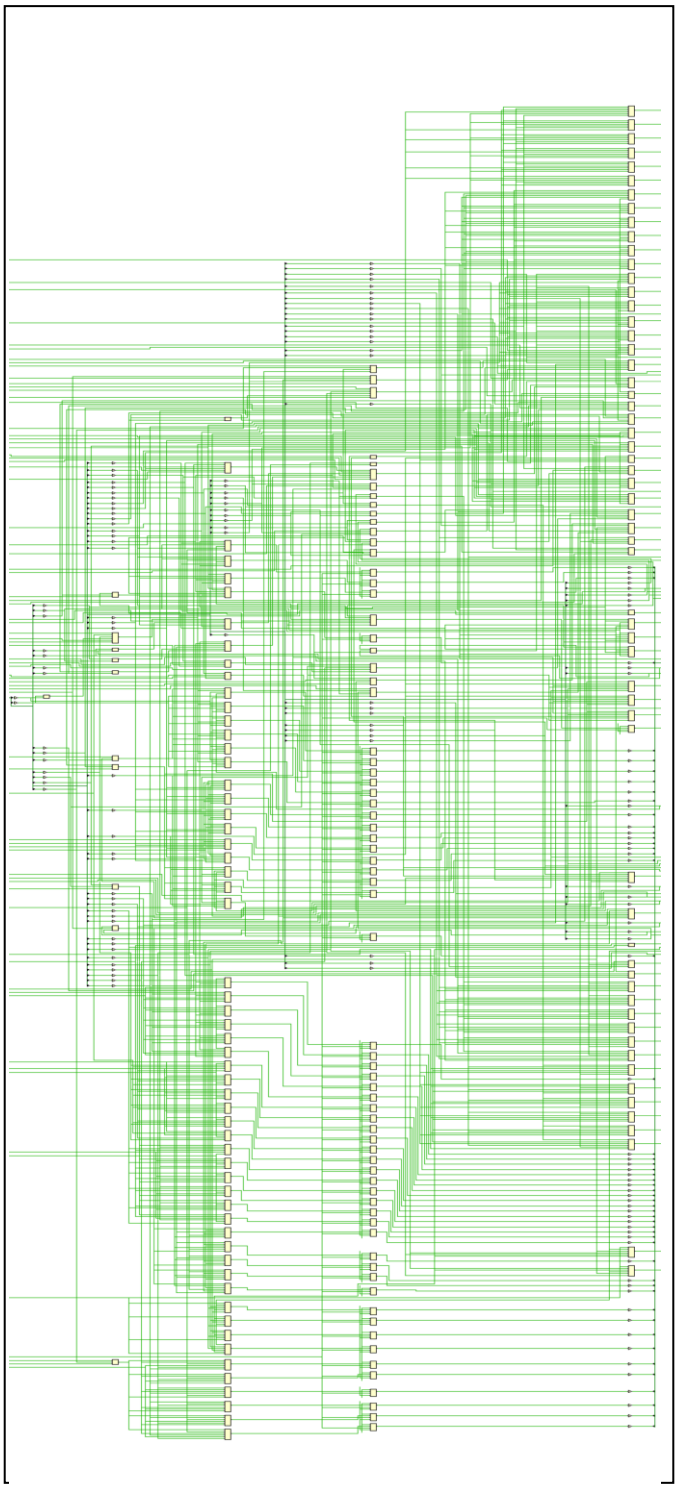


Fig 7: Synthesis schematic

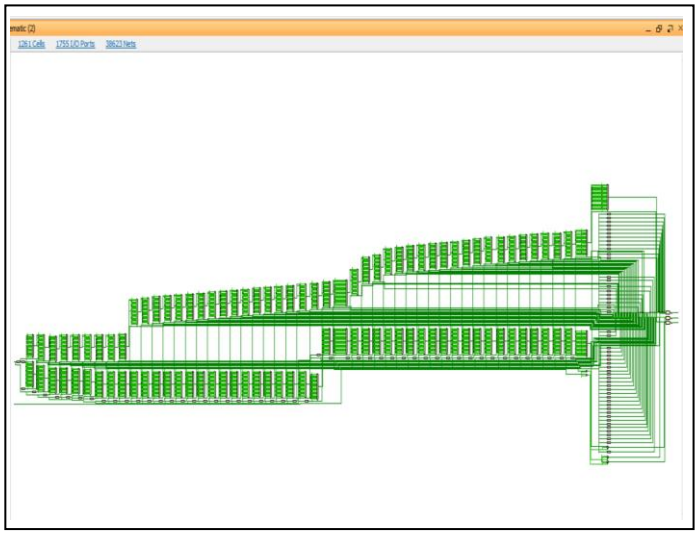


Fig8: RTL schematic

V. CONCLUSION

The design and implementation of text to braille translator has been presented in this project. A Text-to-braille translator has been successfully designed which converts a given sentence into braille code format consisting of cells of raised dot pattern.

The Verilog code has been successfully simulated using Xilinx Vivado development platform. The aim of this translator is to help people who are blind and partially able to read and write. The ability to read and write in braille has opened door to literacy, equal opportunity and personal security and through this braille translator we are able to achieve this.

In the current version, the system can be used in high performance applications. The software version now can be further improved by translations in many languages [Multi language translator].

Standards for Braille translation are much higher than for print. This level of accuracy is necessary because Braille uses the same ASCII code for different purposes according to the context. The results obtained show that the system is able to implement text-to-Braille translation with high accuracy.

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, “Title of paper if known,” unpublished.

[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.