Hardware-Based Text-To-Braille Translator

Xuan Zhang, Cesar Ortega-Sanchez and Iain Murray

Electrical and Computer Engineering Department
Curtin University of Technology
Kent Street, Bentley 6102, Western Australia
i.murray@ece.curtin.edu.au

ABSTRACT

This paper describes the hardware implementation of a text to Braille Translator using Field-Programmable Gate Arrays (FPGAs). Different from most commercial software-based translators, the circuit presented is able to carry out text-to-Braille translation in hardware. The translator is based on the translating algorithm, proposed by Paul Blenkhorn [1]. The Very high speed Hardware Description Language (VHDL) was used to describe the chip in a hierarchical way. The test results indicate that the hardware-based translator achieves the same results as software-based commercial translators.

Categories and Subject Descriptors

B.7.1 [Integrated Circuits]: Types and Design Styles − advanced technologies, algorithms implemented in hardware, gate arrays, input/output circuits, memory technologies, VLSI (very large scale integration).

General Terms: Algorithms, Design, Languages, Verification, FPGA design.

Keywords: Braille translation, FPGAs, VHDL.

1. INTRODUCTION

In 1829 Louis Braille developed a system based on a 6-dot cell, which allowed the blind to read and write. Useful as it was, the original Braille system suffered from low throughput because text had to be spelled letter by letter. To solve this problem, English and other languages introduced the use of contractions [1, 3]. When contractions are used, Braille is usually called "grade 2", in contrast to "grade 1" transcriptions [2].

Since Braille became one of the most important ways for the blind to learn and obtain information, translating normal text into Braille became a necessity. Today, most Braille translators are computer-based and use the American Standard Code for Information Interchange (ASCII).

Paul Blenkhorn's proposed a system to convert text into Standard English Braille [1]. This method uses a decision table with input lasses and states and a rule table with all rules for translation. The format of each row in the table is:
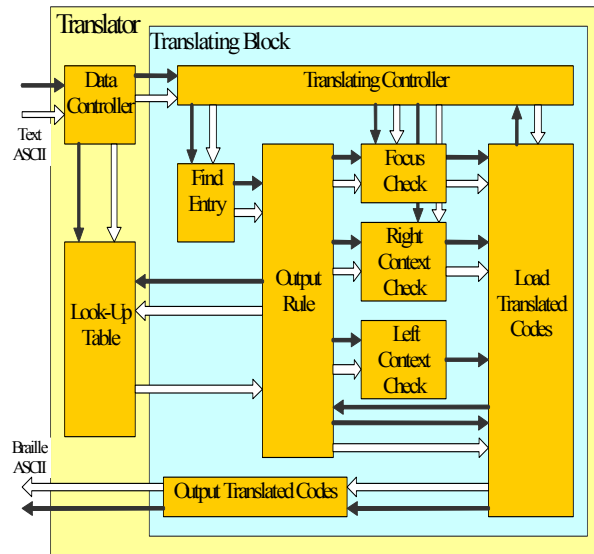
Input class <TAB> Rule <TAB> New state

Figure 1. Block diagram of text to Braille translator

The system presented in this paper only considers grade 2 Braille translations; hence the table can be simplified by ignoring input classes and states. Only rules are considered.

Every rule has the following format:

Left context [focus] Right context = input text

All rules are listed in ASCII alphabetical order. For rules whose focuses start with the same character(s), the order in which they appear in the table is related to their priority. Therefore, the rules have to be checked in order.

2. ARCHITECTURE OF THE SYSTEM

Figure 1 shows a block diagram of the text to Braille translator implemented in an FPGA.

Figure 1 shows that the translating block consists of 8 sub-blocks. The translating-controller block gets feedback from the load-translated-codes block and also receives and stores the text data in registers. In this particular implementation, the translator carries out the conversion five words at a time.

In Figure 1 the find-entry block receives one entry character from the translating-controller block and outputs a particular address to the output-rule block. The entry character is the first un-translated character in the input text string. In this block, there is a look-up table that stores all the entry addresses. If an address corresponds to a particular entry character, it is sent to the output-rule block. However, if no entry address can be found for a particular character, it means the entry character is not in the list. Therefore, a fail signal is issued and the character will be output without translation.

Two operations keep running in the output-rule block. One is reading rules from the look-up-table block, and the other is sending every single rule to focus-check, right-context-check, left-context-check, and load-translated-codes blocks. The output-rule block receives signals from the find-entry block obtaining addresses, and signals from the load-translated-codes block that indicate if the output rule can be used.

The output-rule block sends an address to the look-up-table to read one rule at a time and sends it separately to focus-check, right-context-check and left-context-check blocks. If the rule does not find a match, then a signal is generated and the output-rule block gets the next rule and sends it. This process continues until a match is found and the focus is successfully translated.

The focus-check and right-context-check blocks receive not only the rule from output-rule block, but also the whole group of words to be translated from the translating controller because more than one letter of focus and right context might need to be checked. The left-context-check block checks one or two previously translated characters according to the look up table. These three blocks perform similar functions.

As shown in figure 1, the focus-check, right-context-check and left-context-check blocks work concurrently, providing better performance than sequential implementations. Each block generates signals for the load-translated-codes block indicating if the focus, the right context or the left context were successfully matched. If one of the three fails, then a signal is sent back to the output-rule block requesting the next rule. If the focus, right context and left context match one of the rules, then the load-translated-codes block sends the translated codes to the output-translated-codes block, and informs to the translating-controller block how many characters were translated. After one group of characters has been translated, the output-translated-codes block transmits the corresponding Braille ASCII characters one by one. Then the translation of a new set of characters begins.

3. IMPLEMENTATION AND TEST

The translator has been implemented using a Top-Down design methodology where high level functions are defined first, and the lower level implementation details are filled in later [4, p.90]. The system has been successfully implemented in a Xilinx's Virtex-4 FPGA evaluation board [5].

The texts to be translated, as well as the results of the translation were stored in a PC as text files and transmitted using an RS-232 serial connection. Figure 2 shows the setting used to test the translator. The system works as follows:

1. The text to be translated is sent to the FPGA through a serial link using Hyper Terminal.
2. Part of the FPGA implements a receiver that converts serial data into bytes that are loaded into the translator.
3. The translator takes the new character and stores it in a buffer. Characters are stored until a space is detected. At this point the translation process described in section 2 takes place.
4. The results of the translation are sent to a serial transmitter so that they can be received and stored in a text file by the computer.

For the implementation reported in this paper, the FPGA receives the text file to be translated at 4,800 bauds and sends the translated text back to the PC at 57,600 bauds.
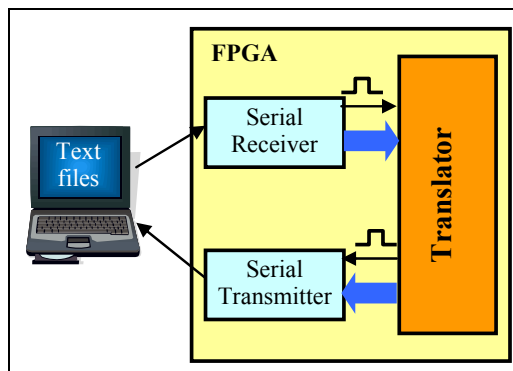

Figure 2. Test bench for Braille translator.

In this setting, the translator runs faster than the serial communication channel because, even though the translation process can be finished in the time required to transmit one bit of information to the PC, the current implementation works on words, hence; after a word has been translated it has to be sent to the computer before the next word is received. Parallel communications could be used to increase the throughput of the system.

During testing, outputs of the hardware translator were compared against the outputs of a commercial Braille translation program. The results show that the hardware translator is able to perform translations with the same accuracy as the commercial system.

4. CONCLUSIONS

The design and implementation of an FPGA-based, text-to-Braille translator has been presented. In its current version, the system can be used in embedded and high-performance applications. However, there are several improvements which will be incorporated in future versions of the hardware translator. For example, the current system is a stand-alone component. Its structure has to be changed for every individual application. An improved version will incorporate the hardware translator in a system on a chip for multifunctional text-Braille translation. The system will consist of a microcontroller for interface and control, and the text-Braille translator, all integrated in one single chip. For further improvement, a multi-language-Braille translator will be considered. Look-up tables for different languages could be stored in flash memory so that when translation of text in a particular language is required, the microcontroller loads the corresponding look-up table.

5. REFERENCES

[1] Blenkhorn, P. A System for Converting Print into Braille, *IEEE Transactions on Rehabilitation Engineering*, vol. 5, No. 2, pp. 121-129, 1997.

[2] Jonathen, A. Recent Improvement in Braille Transcription, *Proceedings of the ACM annual Conference*, vol. 1, Boston, pp.208-218, 1972.

[3] Blenkhorn, P. A System for Converting Braille into Print, *IEEE transactions on Rehabilitation Engineering*, vol. 3, no. 2, pp.215-221, 1995.

[4] Zeidman, B. Designing with FPGAs and CPLDs, *CMP books*, 2002, ISBN: 1-57820-112-8.

[5] Memec Inc. *Virtex-4(TM) FX12 LC Development Board User's Guide*, electronic documentation, version 1.0, 2005.