Implementation of Data Compression Algorithms on FPGA using Soft-core Processor

Vijay G. Savani*, Piyush M. Bhatasana, Akash I. Mecwan
Assistant Professor, Institute of Technology, Nirma University, India.
Corresponding Author Email: vijay.savani@nirmauni.ac.in

Abstract

With the increase in the requirement of online real time data, data compression algorithms are to be implemented for increasing throughput. This paper describes the methods of creating dedicated hardware which can receive uncompressed data as input and transmit compressed data at the output terminal. This method uses FPGA for the same, wherein the hardware part has been created using Xilinx Embedded Development Kit (EDK) and data compression algorithms have then been implemented on the same hardware. The EDK helps creating a Soft Core Processor on the FPGA with desired specifications. The data compression algorithm can be implemented on this processor. The advantage of this kind of a system is that, without changing the hardware, the FPGA can be reprogrammed with a new algorithm whenever a better technique is discovered. For the proof of concept the Huffman coding technique has been implemented. The Soft Core Processor uses serial port and for direct input the GPIO of the processor were used. The user enters text data through this port, and the soft core processor using Huffman's data compression algorithm gives compressed data as the output.

Keywords: Data Compression algorithms, EDK, Micro Blaze Soft Core Processor, FPGA.

1. Introduction

As the demand for increased data throughput increases day by day, it becomes vital for us to compress data and thereby reduce its size. For this, we require implementing various data compression algorithms (Lossless and Lossy). A dedicated hardware developed for the same can be developed using FPGA (Field Programmable Gate Array). Here the dedicated hardware was developed along with the softcore processor. The data source transmits data which is received by the Processor and any of the available data compression algorithms is implemented in the processor. Compressed data is received at the output port of the processor. The following block diagram explains this process.

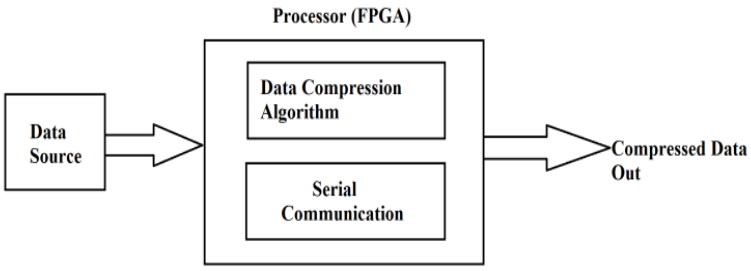

Fig. 1: Project Introduction

The data source could be any user transmitting data like text, audio or video etc. The uncompressed data is received by the processor block where any of the various data compression algorithms (discussed briefly in the next sections) is implemented on it. Finally, the compressed data is transmitted through the GPIO of the processor or via the serial port for testing the results.

2. Data Compression Algorithms

Generally data compression is also referred to as coding, where coding is a very general term encompassing any special representation of data which satisfies a given need. Information theory is defined to be the study of efficient coding and its consequences, in the form of speed of transmission and probability of error. Different Coding Technique [1] are described the following section.

2.1 *Shannon Fano Technique*

For the Shannon Fano coding, the symbols are arranged in order from most probable to least probable, and then divided into two sets whose total probabilities are as close as possible to being equal. All symbols then have the first digits of their codes assigned; symbols in the first set receive "0" and symbols in the second set receive "1". As long as any sets with more than one member remain, the same process is repeated on those sets, to determine successive digits of their codes. When a set has been reduced to one symbol, of course, this means the symbol's code is complete and will not form the prefix of any other symbol's code.

2.2 *Huffman Coding Technique*

Huffman coding basically uses a specific method for choosing the representation for each symbol, resulting in a prefix code (sometimes called "prefix-free codes", that is, the bit string representing some particular symbol is never a prefix of the bit string representing any other symbol) that expresses the most common source symbols using shorter strings of bits than are used for less common source symbols. This algorithm is most efficient compression method and no other mapping of individual source symbols to unique strings of bits will produce a smaller average output size when the actual symbol frequencies agree with those used to create the code. A method was later found to design a Huffman code in linear time if input probabilities (weights) are sorted.

2.3 *LZ77 &LZ78 Techniques*

There are many different types of LZ algorithms, they all work by parsing the input sequence into distinct phrases and then each phrase is encoded by reference to some previous phrase and perhaps some additional information, which will exploits redundancy in the input sequence. Particularly simple variant described below does its parsing in a greedy way. At any point in its execution, it is at some point along the input sequence. From here, it searches ahead for the shortest phrase that has not been seen before. Because of this new phrase can be encoded by reference to a previously seen phrase plus one character compression can be achieved. This process repeats after new phrase is entered in the dictionary.

3. Implementation Methods

For implementing the hardware for data compression, the various different methods for the same are to be explored. The hardware should be capable of meeting the requirements as well as be flexible for reprogramming. The following are the few options to be considered for realizing the hardware for the data compression algorithm.

3.1 Using FPGA as a Hardware

One of the methods to implementation hardware for the data compression algorithm is to use FPGA and algorithm both using HDL language. This is like a programming system where the hardware is defined once with the implementation of the code.

3.2 Using Soft Core Processor on FPGA

One can develop a softcore processor hardware having required specifications using the Xilinx EDK tool kit and then implement the software algorithm on this processor. This feature allows us to change the software algorithm whenever wanted without changing the hardware description. This gives an additional power to the user to be able to change the compression technique without making any difference to the existing hardware.

3.3 Using a available hardcore processor

This task can also be achieved by using already available processors like the ARM processor. In this case, we are required to implement the algorithm using any programming tool like C or C++ on the processor hardware.

Of all the options listed above, the paper describe the second option, because it gives the power of selecting processor of our requirement using the tool like EDK (Embedded Development Kit) for hardware description. The algorithm for data compression can be implemented on this processor by writing the source code in the C code, using the toll like SDK (Software Development Kit). One can change the data compression algorithm whenever we want just by changing the code.

4. Introduction to FPGA

FPGA is an integrated circuit designed to be configured by the customer or designer after manufacturing. The FPGA configuration is generally specified using a hardware description language (HDL). FPGAs can be used to implement any logical function. FPGA has ability to update the functionality of the hardware by doing the partial reconfiguration of a portion of the design in the hardware. FPGAs contain programmable logic components called generally called logic blocks, and a hierarchy of reconfigurable interconnects that allow the blocks to be connected together. Many changeable logic gates that can be interred wired in many different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates.

5. Features of VIRTEX 5 family FPGA

The Virtex-5 family from the Xilinx vendor provides the newest most powerful features in the FPGA. In addition to the most advanced, high performance logic fabric, Virtex-5 FPGAs contain many hard IP system level blocks Additional platform dependant features include power optimized high speed serial transceiver blocks for enhanced serial connectivity These features allow advanced logic designers to build the highest levels of performance and functionality into their FPGA based systems.

6. Soft Core Processor

6.1 *Introduction*

The processor whose architecture and behavior are fully described using a synthesizable subset of a hardware description language is called soft core processor. They can be compiled for any application specific Integrated Circuit or FPGA devices; hence it provides designers with a very good amount of flexibility in terms of hardware and feature requirement. The soft core processor provides many advantages to the designer like

a) Soft core processors are flexible and can be customized for a specific application with relative ease.

b) Since soft-core processors are technology independent and can be synthesized for any given target technology, they are therefore more immune to becoming obsolete when compared with circuit or logic level descriptions of a processor.

c) A softcore processor's architecture and behavior are described at a higher abstraction level using an HDL; it becomes much easier to understand the overall design.

d) User level customization of the design.

e) Obsolescence mitigation.

f) Component and cost reduction

g) Hardware acceleration

6.2 *Peripherals and memory controllers*

To built embedded soft core processor on FPGA both Xilinx and Altera offer extensive libraries of intellectual property (IP) in the form of peripherals and memory controllers. The IP is included in the toolsets provided by these manufacturers long with the tool. To increase the versatility and flexibility few of the IP are like Peripherals & peripheral controllers which includes the General purpose I/O, UART, Timer, Debug, DMA Controller etc. and memory controllers like SRAM, Flash controller, Compact Flash controller are available inbuilt.

6.3 *Microblaze Softcore Processor*

This microprocessor is available for use in Xilinx's Field Programmable Gate Arrays with EDK software tools. The MicroBlaze is a virtual microprocessor that is built by combining blocks of code called cores inside a Xilinx FPGA. The beauty to this approach is that we only end up with as much microprocessor as we need. MicroBlaze processor is a 32-bit Harvard RISC architecture optimized for implementation in Xilinx FPGAs which having separate 32-bit

instruction and data buses. This flexibility of choosing the require hardware in the final design allows the user to balance the required performance of the target application.

7. Embedded Hardware Platform

7.1 Embedded Development Kit (EDK)

The Xilinx Company provides the suite of tool called EDK as well as IP that enables us to design a complete embedded processor system for implementation in a Xilinx FPGA device. This tool is used to create the hardware system using the Microblaze soft core processor along with require features depends upon the application of the hardware. Also have so many IP, which can be added in the hardware as when required.

7.2 Software Development Kit (SDK)

Once the hardware is ready, we require to write the software as well as algorithm which will be run on processor, for this purpose we use the tool named as SDK (part of EDK tool suit). SDK is built on the Eclipse open source framework, so this software development tool might appear familiar to you or members of your design team. The figure 2 Shows the entire the hardware and software flow for developing the application.

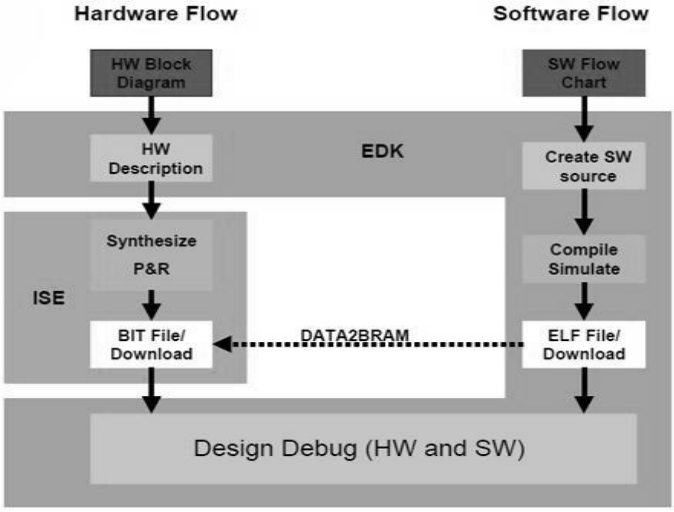

Fig. 2: Hardware/Software Flow for EDK/SDK

8. Results

The implementation of Huffman code for text data compression is done. Here, the user on the serial command window enters text to be compressed. The Huffman data compression code calculates the frequencies of all the characters used and calculates their probabilities of occurrence. Based on these probabilities, the optimum codes for achieving good compression are assigned to the characters and finally the compressed code is displayed on the output screen. The bits required before and after compression are also shown along with the code. The number of

bits required before compression are calculated based on the ASCII format requirement (8 bits per character). The result is shown in the figure 3.

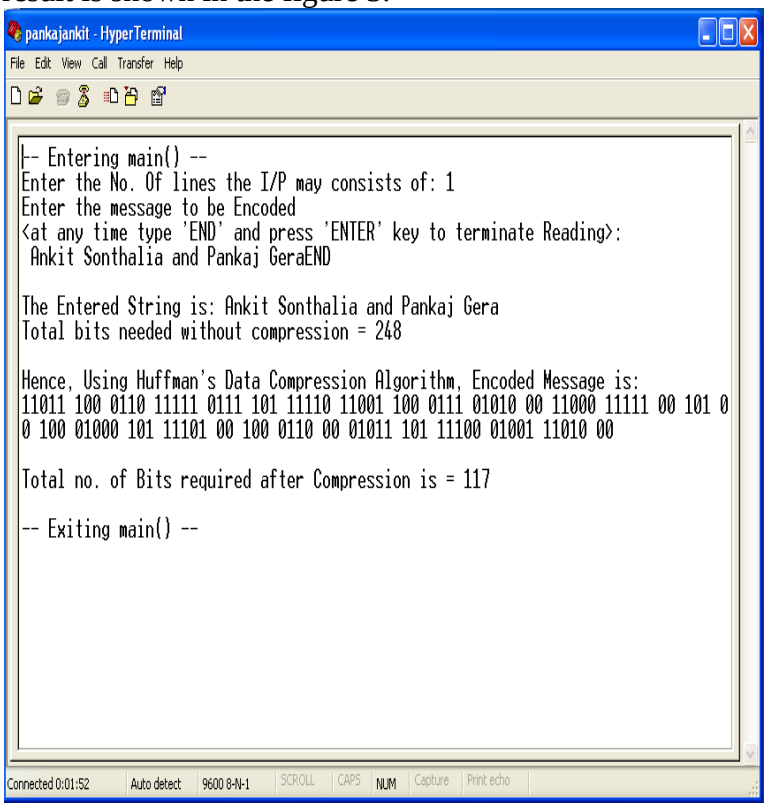

Fig. 3 Hyper Terminal Results

9.Conclusion

With Embedded systems being developed on a large scale in today's world, a dedicated system designed especially for data compression is a new development. A lot of work has been done on designing algorithms which help data compression; and further developments shall continue to come in this sector. This project shows how data from a source can be received by means of a serial port and processed for data compression on a Soft Core Processor. Here, Huffman data compression algorithm is used as the compression technique. Because of the flexibility of the MicroBlaze Soft Core processor any new algorithm can be implemented by reconfiguration of the device without changing the hardware design specifications. Here, the data being compressed was text entered by the user through the serial port. However, we may think of other modifications for further development as per the user requirement. For example, the data may be any bit stream representing any form of multimedia (audio, video or image). The compression algorithm suitable for that form of data can also be implemented on the processor. Moreover, other hardware interfaces and transmission media such as Ethernet or Fiber Optics may also be used for transmitting data to the processor for compression. Not only that, but the data reception and transmission may also be done in real time. Thus, the project does all the ground work necessary for developing an online real time data compression system.

References

[1] Debra A. Lelewer and Daniel S. Hirschberg, *Data Compression*., 1987.
[2] Xilinx. www.xilinx.com. [Online]. http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf
[3] *Ian D. L. Anderson and Mohammed A. S. Khalid Jason G. Tong, "Soft-Core Processors for Embedded Systems," in 18th International Conference on Microelectronics, 2006.*
[4] Xilinx.,http://www.xilinx.com/ise/embedded/emb_ref_guide.pdf
[5] www.xilinx.com. [Online]. http://www.xilinx.com/microblaze/
[6] www.xilinx.com. [Online]. http://www.xilinx.com/support/documentation/sw_manuals/edk92i_ctt.pdf
[7] Khalid Sayood, *Introduction to Data Compression*.