International Journal of Research in Advent Technology, Vol.3, No.1, January 2015
E-ISSN: 2321-9637

An Optimum Algorithm for Data Compression using VHDL: A Review

Mr. Pralhad Sarkar[1], Prof. Ravindra Kadam[2], Prof. Prashant Indurkar[3]
[1]*M.Tech(VLSI) Scholar, B.D.C.E, Sewagarm, Wardha. Email: Sarkar619@rediffmail.com*
[2]*Assistant Professor, B.D.C.E, Sewagarm, Wardha. Email: Rdk_arvi@rediffmail.coms*
[3]*Associate Professor, B.D.C.E, Sewagarm, Wardha. Email: Prashantindurkar@rediffmail.com*

Abstract-Today's embedded system has a numerous application, but it has a limited internal memory. This paper review and discuss an optimum algorithm for data compression using VHDL so that we can access wide range of program in a limited internal memory. The main objective of data compression is to find out the redundancy and eliminate them through different efficient methodology, so that the data which is reduced require less memory as well as the size of the data decreases hence the cost of transmission is also reduce. This paper gives comparative study of different compression method.

Index terms –code compression; embedded system; pattern block;lossy; multilevel dictionary.

1. INTRODUCTION

With the increase in the requirement of online real time data, data compression algorithms are to be implemented for increasing throughput. Compression is the art of representing information in a compact form rather than its original or uncompressed form. The compression code is place in main memory and/or instruction cache memory, thereby increasing the number of stored instruction, then increasing the cache hit rate and decreasing the search into the main memory, thus increasing the system performance and reducing power consumption. The original file which is to be compressed is first coded which is then known as encrypted file. For any efficient compression algorithm file size must be less than the original file. To get back the original file we need to 'decrypt' the encoded file. Data compression methods are sometime very difficult as it require hardware for its implementation ants of maintains.

Data compression can either can be lossless or lossy. Lossless data compression recreates the exact original data from the compressed data while lossy data compression cannot regenerate the perfect original data from the compressed data. The data source transmits data which is received by the processor. Compressed data is received at the output port of the processor. Fig. 1explains this process.

A good compression rate can be achieved by compressing the instruction which occurs most

frequently into the code, and that can be obtained from analysis conducted in static or dynamic form.

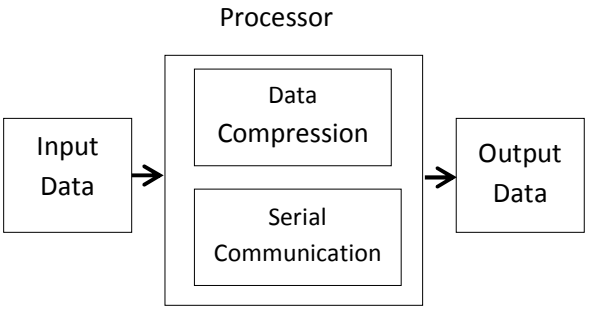

Fig. 1 Data Compression Logic

The goal of data compression is to eliminate the redundancy in a file's code in order to reduce its size. It is useful in reducing the data storage space and in reducing the time needed to transmit the data. The evolution of computing has created a rapid expansion in the volume of data to be stored on hard disk and sent over the internet. This growth has led to a need for data compression (i.e. the ability to reduce the amount of storage or internet bandwidth required to handle data). 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.

The main purpose of data compression is to find out redundancy and eliminate them through different efficient methodology. The code compression helps

make better use of the resources of embedded system. As the evolution of computing has created a rapid expansion in the volume of data to be stored on hard disks and sent over the internet. This growth has led to need for data compression.

2. LITURATURE SURVEY

Wander Roger Azervedo Dias, Edward David Moreno and IssancPalmeria give a new method of code compression for embedded systems by them as CC-MLD (Compressed Code using Huffman-Based Multi-Level Dictionary). This method applies two compression techniques and it uses the Huffman code compression algorithms. A single dictionary is divided into two levels and it is shared by both techniques. They performed simulations using application from MiBench and they had used four embedded processor (ARM, MIPS, PowerPC and SPARC). Their method reduces code size up to 30.6% (including all extra costs for these four platforms). They had implemented the decompressor using VHDL and FPGA and they had obtained one clock from decompression process [1].

Ivan Scherbakov, Christian Weis and Norbert When had describe a design and develop a data compression engine on a single FPGA chip that is used as part as part of text-classification application. The implementation of the prediction by partial matching algorithm and arithmetic coding data compression is totally in hardware as well as in software code. Their design implements a dynamic data structure to store the symbol frequency counts up to maximal order of 2. The computation of the tag-interval that encodes the data sequence in arithmetic coding is done in parallel architecture that achieves a high speed up factor. Even with a relatively slow 50MHz clock their hardware engine performs more than 70 times faster than a software based implementation in C on a CPU running on a 3 Ghz clock. [2]

Joel Ratsaby and VadimSirota had presented a flexible high-performance implementation of the LZSS compression algorithm capable of processing up to 50 MB/s on a Virtex-5 FPGA chip. They exploit the independently addressable dual-port block RAMs inside the FPGA chip to achieve an average performance of 2 clock cycles per byte. To make the compressed stream compatible with the ZLib library they encode the LZSS algorithm output using a fixed Huffman table defined by the Deflate specification. They also demonstrate how changing the amount of memory allocated to various internal tables impacts the performance and compression ratio. Finally, they provide a cycle-accurate estimation tool that allows finding a trade-off between FPGA resource utilization, compression ratio and performance for a specific data sample.[3]

Vijay G. Savani, Piyush M. Bhatasana 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 [4].

ArohiAgarwal and V.S.Kulkarni had discussed about Data transmission, storage and processing are very necessary nowadays. Data can be represented in compact form using data compression for transmitting and storing a huge volume of data required large space which is an issue. In order to transmit and store such a large volume of data it requires large memory space and large bandwidth avability. Because of which the hardware increases as well as cost increases. Hence to solve this it is necessary to reduce the size of the data which is to be transmitted without any information loss .For this purpose they have taken the following algorithm. LZMA is a lossless dictionary based algorithm which is used in 7zip was proving to be effective in unknown byte stream compression for reliable lossless data compression. Here the algorithm LZMA is implemented on SPARTAN 3E FPGA to design both the encoder and the decoder which reduces the circuit size and its cost [5].

3. OVERALL ANALYSIS OF REPORTED WORK AND COMPARATIVE STUDY

[1]This paper presents a new code compression method (CC-MLD Compressed Code using Huffman – Based Multi-Level Dictionary) that was implemented in C language. Through the simulation

they can reached on average for the four processor a compression rate close to 31% and they detect that the decompression process was executed in just one-clock. This method (CC-MLD) method doesn't bring big overhead to the execution of programs in embedded processor.

[2]This paper describes a high performance flexible implementation of LZSS algorithm on vertix5 FPGA. The design compressor is flexible and allows tuning various parameters to achieve trade –offs between the speed, compression ratio and block RAM utilization. The drawback of this method is 8.4% of the time is spent on waiting for the headtable to read when the prefetched hash value is not useful (i.e when a valid match is found and several bytes are skipped).

[3] in this paper give a data compressor that use the PPM algorithm with arithmetic coding. It is implemented on a single FPGA chip on an inexpensive evaluation board (Xilinx Spartan 3A) with a 50 Mhz clock. This compressor executes the algorithmic stages in parallel. The PPM unit updates multiple data stuctures, some of which are dynamically and increntally allocated, that keep the symbol count statistics. The advantage of this compressor is more 70 times faster than a C based software implementation running on a 3Ghz PC.

[4]This project shows how data from the source can be received by means of a serial port and processed for data compression on Soft Core Processor. In this paper Huffman data compression algorithm is used as the compression technique. The data being compressed was text entered by the user through serial port. But this method is complicated for any bit stream representing any form of multimedia (audio, video or image).

[5]This paper present a portable and efficient LZMA compression algorithm will be implemented using VHDL that provides an excellent platform for Real time compression application. Encoding and decoding process are fully functional at 50MHz and compression ratio is comparable with that of real compression ratio.

Author	Platform	Bench-mark	Compression Rate
Wanderson Dias & Edward Moreno	ARM	MiBench	31.3%
	MIPS		30.7%
	PowerPc		30.3%
	SPARC		29.9%
Ivan sheherbakov	Vertax 5 FPGA Chip	Xilinx	50MB/s
Joel Ratsaby	FPGA	Xilinx	70% faster than C based program.
Vijay G. Savani	Soft core processor	Xlinx	-------
ArohiAgarwal	FPGA	Xilinx	Fully functional with 50MHz.

TABLE I SUMMERY OF RELATED WORK

4. OBJECTIVE

1. As we referred the previous method, in which the techniques used to obtain compressed data on MiBench benchmark using C language coding. So our first objective is to find out how we get compressed data on different benchmark using VHDL coding to vary the compression rate form previous one.
2. So this compression technique will also reduce delay.
3. This technique will be faster than C based compression technique.
4. It will optimize in Area/memory.
5. Vary the compression rate form pervious work

5. PROPOSED METHODLOGY

5.1 OPTIMUM DATA COMPRESSION METHOD.

Length are encoded separately and the bit mask is also encoded search buffer should have a suitable data structure which reduces the times required for long matching. The encoded process is more typical as it needs to encode the longest match and decode as it need to encode the longest match and the decoding is far easier. The fig, 2 gives basic steps required for the compression process.

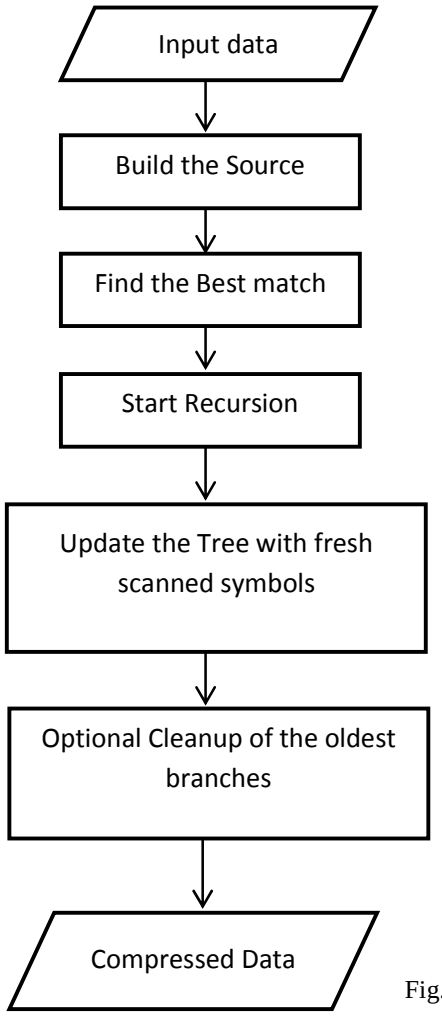

Fig. 2

Compression Process Flow

5.2 ENCODING AND DECODING PROCESS

The input data stream is selected properly by using delta filter and then its effective compression is done using the compression technique. The data is store in the sequential from rather than complete file. The data stream which is used as the input is the output of the first byte of delta encoding.

The difference between the current and its previous bytes is store as the subsequent byte. The data encoding is basically used to make the encoding more effectively by continuous varying of real time data.

5.3 EFFICIENT DELAY

This Data compression technique using VHDL gives an efficient delay in data encoding and decoding process which makes boost up the speed.

5.4 Area/ memory Optimization

The extremely fast growth of data that needs to be stored and transferred has given rise to the demand of better transmission and storage techniques. This data compression technique gives an optimum area or memory as required, which gives fastest processing. It is useful in reducing the data storage space and in reducing the time needed to transmit the data.

6. CONCULSION

The simple, portable and efficient compression algorithm will be implemented using VHDL that provides an excellent platform for Real time compression application. The proposed method will be tested with Altera modelsim software for various input data size and the different compression ratio will be calculated. Further this architecture can be extended to application specific integrated circuit so as to design a specific hardware ship for various data compression algorithm.

REFERENCE

[1] Wander Roger Azevedo Dias, Edward David Moreno, IssancPalmeria, " A New code compression Algorithm and its Decompressor in FPGA – Based Hardware" IEEE, 2013.

[2] Ivan Shcherbakov, Christian Weis, Norbert When, "A High-Performance FPGA-Based implementation of LZSS compression algorithm, IEEE, 2012.

[3] Joel Ratsaby, vadimSirota, " FPGA – based data compression based on Prediction by Partial Matching, IEEE, 2012.

[4] Vijay G. Savani, Piyush M. Bhatasana, AkashMecwan, "Implementation of Data Compression Algorithm on FPGA using soft core processor" ijict, Dec,2012.

[5] "FPGA based implementation of Data compression using Dicitionary based 'LZMA' Algorithm" by ArohiAgrawal, V.S.Kulkarni, IRF international Conference.

[6] "FPGA Based Lossless Data Compression using Huffman and LZ77 Algorithms", @2007, IEEE.

[7] Indian Journal of Computer Science and Engineering,"Comprarison of Lossless Data Compressional Algorithms for text Data", by S.R. KODITUWAKKU.

[8] "Data compression Methodologies for Lossless Data and Comparison between Algorithms", International Journal of Engineering Science

and Innovative Technology (IJESIT) issue 2, March 2013.

[9] " Analysis and compression of Algorithms for Lossless Data Compression", International Journal of Information and Computation Technology, ISSN 0974-2239 Volume 3,2013

[10] "FPGA-Based lossless Data Compression using GNU Zip", thesis presented to university of Waterloo, Ontario, Canada, 2007.