April 2015

DATA COMPRESSION USING EFFICIENT DICTIONARY SELECTION METHOD

SRITULASI ADIGOPULA

Department of Electronics and Communication Engineering, Chirala Engineering College, Chirala.,
sritulasiadigopula@gmail.com

P. BALANAGU

Department of Electronics and Communication Engineering, Chirala Engineering College, Chirala.,
pbalanagu@gmail.com

N. SURESH BABU

Department of Electronics and Communication Engineering, Chirala Engineering College, Chirala.,
nsureshbabu@gmail.com

Follow this and additional works at: <https://www.interscience.in/ijcct>

Recommended Citation

ADIGOPULA, SRITULASI; BALANAGU, P.; and BABU, N. SURESH (2015) "DATA COMPRESSION USING EFFICIENT DICTIONARY SELECTION METHOD," *International Journal of Computer and Communication Technology*: Vol. 6 : Iss. 2 , Article 10.

DOI: 10.47893/IJCCT.2015.1287

Available at: <https://www.interscience.in/ijcct/vol6/iss2/10>

This Article is brought to you for free and open access by the Interscience Journals at Interscience Research Network. It has been accepted for inclusion in International Journal of Computer and Communication Technology by an authorized editor of Interscience Research Network. For more information, please contact sritampatnaik@gmail.com.

DATA COMPRESSION USING EFFICIENT DICTIONARY SELECTION METHOD

SRITULASI ADIGOPULA, P.BALANAGU & N.SURESH BABU

Department of Electronics and Communication Engineering,
Chirala Engineering College, Chirala.

Abstract: With the increase in silicon densities, it is becoming feasible for compression systems to be implemented in chip. A system with distributed memory architecture is based on having data compression and decompression engines working independently on different data at the same time. This data is stored in memory distributed to each processor. The objective of the project is to design a lossless data compression system which operates in high-speed to achieve high compression rate. By using the architecture of compressors, the data compression rates are significantly improved. Also inherent scalability of architecture is possible. The main parts of the system are the data compressors and the control blocks providing control signals for the Data compressors, allowing appropriate control of the routing of data into and from the system. Each Data compressor can process four bytes of data into and from a block of data in every clock cycle. The data entering the system needs to be clocked in at a rate of 4 bytes in every clock cycle. This is to ensure that adequate data is present for all compressors to process rather than being in an idle state.

Keywords: Dictionary Selection Method, decompression, test compression.

1. INTRODUCTION

Data Compression methodology is one of the most important techniques for testing large and complex systems. The efficiency of implementation is characterized by the test length and the hardware overhead required to achieve complete or sufficiently high fault coverage. It is as shown in figure 1

An on-chip decoder decodes the compressed test data from the memory and delivers the original uncompressed set of test vectors to the design-under-test (DUT).

Fig 1. Data Compression Testing Methodology

In general, combinational circuits are not pseudo exhaustively testable, and deterministic test sets have to be applied if the circuit is not allowed to be segmented by test points for timing or area reasons. A deterministic "test per clock" scheme may be implemented by designing an appropriate feedback function of a non-linear feedback shift register [DaMu8 11, or by including additional circuitry between an LFSR and the CUT which maps random patterns to deterministic test patterns [AkJa89, DUFA95, ToMc95b, ChPr951. The first solution is only feasible for small circuits and test sets, and the

second one slows down performance, as the additional test circuitry is part of the data path. Moreover, some effort is required to show that the test circuitry is fault free, too.

Our approach solves these problems by selecting efficient dictionary selection algorithms, to improve the compression efficiency without introducing any additional decompression penalty. Our experimental results demonstrate that our approach produces up to 30% better compression compared to existing dictionary-based approaches.

2. RELATED WORK

Dictionary-based compression techniques have been recently used to reduce the test data volume in SOCs. Li et al. and Reddy et al. used fixed length dictionary entries to reduce test data volume. Dynamic dictionaries along with LZ77 technique has been used by Wolff et al. Wurtenberger et al. have proposed a test data compression method by remembering the mismatches with the dictionary entries.

A detailed comparison between their approach and ours is provided at the end of this section. We have proposed a bitmask-based compression technique, which renders significantly better results than Li et al. [1], as demonstrated in Section VI. Bitmask-based compression was developed by Seong et al. [6] for code compression in embedded systems. We have employed a modified version of the bitmask-based compression technique in our test data compression. Our results have demonstrated significant improvement in compression efficiency compared to existing bitmask based compression. This section briefly describes bitmask-based code compression [6], and highlight the challenges in employing bitmaskbased technique for test data compression.

Bitmask-based compression is an enhancement on the dictionary-based compression scheme, that helps us to get more matching patterns. In dictionary-based compression, each vector is compressed only if it completely matches with a dictionary entry. As seen in Fig. 2, we can compress up to six data entries using bitmask based compression.

Fig 2. Bitmask-based test Data Compression

The compressed data is represented as follows. Those vectors that match directly are compressed with 3 bits. The first bit represents whether it is compressed (using 0) or not (using 1). The second bit indicates whether it is compressed using bitmask (using 0) or not (using 1). The last bit indicates the dictionary index. Data that are compressed using bitmask requires 7 bits. The first two bits, as before, represent if the data is compressed, and whether the data is compressed using bitmasks.

The next two bits indicate the bitmask position and followed by two bits that indicate the bitmask pattern. For example, the last data vector in Fig. 2 is compressed using a bitmask. The bitmask position is 11, which indicates the fourth even bit position from left. For this case, we have assumed fixed bitmasks, which are always employed on even-bit positions and hence only 2 bits are sufficient to represent the four positions in a 8-bit data. The last bit gives the dictionary index. The bitmask XORed with the dictionary entry produces the original data. More details on the types and positions of bitmasks will be described in Section IV-B in the context of test data compression. In this example, the compression efficiency is 27.5%, based on the following formula expressed as percentage:

$$\text{Compression Efficiency} = \frac{\text{Original Size} - \text{Compressed Size}}{\text{Original Size}}$$

3. ALGORITHM

The algorithm uses a fixed width dictionary of previously seen data and attempts to match the current data element with a match in the dictionary. It works by taking a 4-byte word and trying to match

this word with past data. This past data is stored in a dictionary, which is constructed from a content addressable memory. Initially all the entries in the dictionary are empty & 4-bytes are added to the front of the dictionary, while the rest move one position down if a full match has not occurred. The larger the dictionary, the greater the number of address bits needed to identify each memory location, reducing compression performance. Since the number of bits needed to code each location address is a function of the dictionary size greater compression is obtained in comparison to the case where a fixed size dictionary uses fixed address codes for a partially full dictionary. In the system, the data stream to be compressed enters the compression system, which is then partitioned and routed to the compressors.

Fig 3 Conceptual View of CAM

The Main Component- Content Addressable Memory Dictionary based schemes copy repetitive or redundant data into a lookup table (such as CAM) and output the dictionary address as a code to replace the data. The compression architecture is based around a block of CAM to realize the dictionary. This is necessary since the search operation must be done in parallel in all the entries in the dictionary to allow high and data-independent throughput.

Flow Graph

```

clear the dictionary;
set the next free location (NFL) to 0;
Do
{
    read in a tuple T from the data stream;
    search the dictionary for tuple T;
    IF (full or partial hit)
    {
        determine the best match location ML and match type MT;
        output "B";
        output any required literal characters of T;
    }
    ELSE
    {
        output "1";
        output tuple T;
    }
    IF (full hit)
    {
        move dictionary entries 0 to ML-1 down by one location;
    }
    ELSE
    {
        move all dictionary entries down by one location;
        increment NFL (if dictionary is not full);
    }
    copy tuple T to dictionary location 0;
}
WHILE (more data is to be compressed);

```

The number of bits in a CAM word is usually large, with existing implementations ranging from 36 to 144 bits. A typical CAM employs a table size ranging between a few hundred entries to 32K entries, corresponding to an address space ranging from 7 bits to 15 bits. The length of the CAM varies with three possible values of 16, 32 or 64 tuples trading complexity for compression.

The no. of tuples present in the dictionary has an important effect on compression. In principle, the larger the dictionary the higher the probability of having a match and improving compression. On the other hand, a bigger dictionary uses more bits to code its

locations degrading compression when processing small data blocks that only use a fraction of the dictionary length available. The width of the CAM is fixed with 4bytes/word.

Content Addressable Memory (CAM) compares input search data against a table of stored data, and returns the address of the matching data.

CAMs have a single clock cycle throughput making them faster than other hardware and software-based search systems. The input to the system is the search word that is broadcast onto the search lines to the table of stored data. Each stored word has a matchline that indicates whether the search word and stored word are identical (the match case) or are different (a mismatch case, or miss).

Fig 4 Architecture of Compressor and decompressor

The matchlines are fed to an encoder that generates a binary match location corresponding to the match line that is in the match state. An encoder is used in systems where only a single match is expected. The overall function of a CAM is to take a search word and return the matching memory location.

Managing Dictionary entries Since the initialization of a compression CAM sets all words to zero, a possible input word formed by zeros will generate multiple full matches in different locations. The compression system simply selects the full match closer to the top. This operational mode initializes the dictionary to a state where all the words with location address bigger than zero are declared invalid without the need for extra logic.

4. SIMULATION RESULTS

In this section, we give the compression performance

and decompression of our approach.

Fig 5 Compression Waveform

Fig 6 Decompression Waveform

5. CONCLUSION

The various modules are designed and coded using VHDL. The source codes are simulated and the various waveforms are obtained for all the modules. Since the Compression/Decompression system uses Dictionary algorithm, speed of compression[4], throughput is high. The Improved Compression ratio is achieved in Compression architecture with least increase in latency. The High speed throughput is achieved. The total time required to transmit compressed data is less than that of transmitting uncompressed data.

Fig 7. RTL Schematic

This can lead to a performance benefit, as the bandwidth[9] of a link appears greater when transmitting compressed data and hence more data can be transmitted in a given amount of time.

Fig 8. Technology Schematic

Fig 9 Design Summary

There is a potential of doubling the performance of storage / communication system by increasing the available transmission bandwidth and data capacity with minimum investment. It can be applied in Computer systems, High performance storage devices. The architecture provides inherent scalability in future.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their comments which were very helpful in improving the quality and presentation of this paper.

REFERENCES:

- [1] F. Hsu, K. Butler, and J. Patel, "A case study on the implementation of Illinois scan architecture," in Proc. Int. Test Conf., 2001, pp. 538–547.
- [2] M. Ros and P. Sutton, "A hamming distance based VLIW/EPIC code compression technique," in Proc. Compilers, Arch., Synth. Embed.Syst., 2004, pp. 132–139.
- [3] S. Seong and P. Mishra, "Bitmask-based code compression for embedded systems," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 27, no. 4, pp. 673–685, Apr. 2008.
- [4] M.-E. N. A. Jas, J. Ghosh-Dastidar, and N. Touba, "An efficient testvector compression scheme using selective Huffman coding," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 22, no. 6, pp. 797–806, Jun. 2003.
- [5] A. Jas and N. Touba, "Test vector decompression using cyclical scan chains and its application to testing core based design," in Proc. Int. Test Conf., 1998, pp. 458–464.
- [6] A. Chandra and K. Chakrabarty, "System on a chip test data compression and decompression architectures based on Golomb codes," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 20, no. 3, pp.355–368, Mar. 2001.

- [7] X. Kavousianos, E. Kalligeros, and D. Nikolos, "Optimal selective Huffman coding for test-data compression," IEEE Trans. Computers, vol. 56, no. 8, pp. 1146–1152, Aug. 2007.
- [8] M. Nourani and M. Tehranipour, "RL-Huffman encoding for test compression and power reduction in scan applications," ACM Trans. Des. Autom. Electron. Syst., vol. 10, no. 1, pp. 91–115, 2005.
- [9] H. Hashempour, L. Schiano, and F. Lombardi, "Error-resilient test data compression using Tunstall codes," in Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst., 2004, pp. 316–323.
- [10] M. Knieser, F. Wolff, C. Papachristou, D. Weyer, and D. McIntyre, "A technique for high ratio LZW compression," in Proc. Des., Autom., Test Eur., 2003, p. 10116.
- [11] M. Tehranipour, M. Nourani, and K. Chakrabarty, "Nine-coded compression technique for testing embedded cores in SOCs," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 13, pp. 719–731, Jun. 2005.
- [12] E. Volkerink, A. Khoche, and S. Mitra, "Packet-based input test data compression techniques," in Proc. Int. Test Conf., 2002, pp. 154–163.
- [13] S. Reddy, K. Miyase, S. Kajihara, and I. Pomeranz, "On test data volume reduction for multiple scan chain design," in Proc. VLSI Test Symp., 2002, pp. 103–108.

AUTHORS PROFILE:

Sritulasi Adigopula is pursuing M.Tech in VLSI & ES at Chirala Engineering College, Chirala.

P. Balanagu, M.Tech in Micro electronics from VTU, Belgaum. Working as Assoc. prof in ECE Dept. CEC, Chirala. He has 9 years of teaching Experience.

Prof. N. Suresh Babu is Vice- Principal & HOD of ECE Dept in CEC, Chirala. He got his M.Tech in Microwave Engineering from Birla Institute of technology, Ranchi. He has 13 years of teaching Experience and 2 Years of Industrial Experience in various organisations.