International Journal of Computer and Communication Technology

Volume 7 | Issue 2 Article 10

April 2016

A Novel Decode-Aware Compression Technique for Improved Compression and Decompression

J. Suresh Babu
Department Of Electronics & Comm. Engineering, Nimra College of Engineering & Technology, Ibrahimpatnam, Vijayawada, sureshbabujakkula@gmail.com

K. Tirumala Rao
Department Of Electronics & Comm. Engineering, Nimra College of Engineering & Technology, Ibrahimpatnam, Vijayawada, ktirumalarao@gmail.com

P. Srinivas
Department Of Electronics & Comm. Engineering, Nimra College of Engineering & Technology, Ibrahimpatnam, Vijayawada, psrinivas@gmail.com

A Novel Decode-Aware Compression Technique for Improved Compression and Decompression

J. Suresh Babu, K. Tirumala Rao & P. Srinivas

Department Of Electronics & Comm. Engineering,
Nimra College of Engineering & Technology, Ibrahimpatnam, Vijayawada
E-mail : sureshbabujakkula@gmail.com

Abstract – With compressed bit streams, more configuration information can be stored using the same memory. The access delay is also reduced, because less bits need to be transferred through the memory interface. To measure the efficiency of bit stream compression, compression ratio (CR) is widely used as a metric. it is a major challenge to develop an efficient compression technique that can significantly reduce the bit stream size without sacrificing the decompression performance. Our approach combines the advantages of previous compression techniques with good compression ratio and those with fast decompression. This paper makes three important contributions. First, it performs smart placement of compressed bit streams to enable fast decompression of variable-length coding. Next, it selects bitmask-based compression parameters suitable for bit stream compression. Finally, it efficiently combines run length encoding and bitmask-based compression to obtain better compression and faster decompression.

Keywords *- Field-programmable gate array (FPGA), bit stream compression, Bitmask-based compression, decompression hardware.*

I. INTRODUCTION

Since the configuration information for FPGA has to be stored in internal or external memory as bit streams, the limited memory size, and access bandwidth become the key factors in determining the different functionalities that a system can be configured and how quickly the configuration can be performed. While it is quite costly to employ memory with more capacity and access bandwidth, bit stream compression technique alleviates the memory constraint by reducing the size of the bit streams.

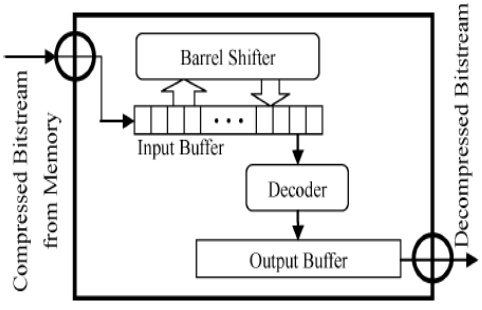


Fig. 1 Decompression module

With compressed bit streams, more configuration information can be stored using the same memory. There are two major challenges in bit stream compression: 1) how to compress the bit stream as much as possible and 2) how to efficiently decompress the bit stream without affecting the reconfiguration time. We can classify the existing bit stream compression techniques into two categories. The techniques in the first category have good compression ratio due to complex and variable- length coding (VLC) [1]–[3]. However, they also need expensive decompression hardware, which may not be acceptable for practical implementation. The other category of compression approaches accelerate decompression using simple or fixed-length coding (FLC) [4] and therefore have very efficient decompression hardware. The only concern is that their compression ratios are usually compromised. Among various compression techniques that has been proposed in recent years, application of bitmask-based compression [5] seems to be attractive for bit stream compression, because of its good compression ratio and relatively simple decompression scheme. However, the original algorithm is proposed for instruction compression and not suitable for FPGA bit stream compression. Moreover, the use of variable-

length coding is challenging for the design of decompression hardware because it requires expensive buffering circuitry as described in Section III. Hence, it is a major challenge to develop an efficient compression technique that can significantly reduce the bit stream size without sacrificing the decompression performance. Our approach combines the advantages of previous compression techniques with good compression ratio and those with fast decompression.

II. RELATED WORK

The difference between consecutive frames (difference vector) is encoded using either Huffman-based run length encoding or LZSS-based compression. Such sophisticated encoding schemes can produce excellent compression. However, they did not address the decompression overhead in [1], which is a major bottleneck in reconfigurable systems. In contrast, many bit stream compression techniques only access the configuration memory linearly during decompression, and therefore can be applied to virtually all FPGAs. The basic idea behind most of these techniques is to divide the entire bit stream into many small words, then compress them with common algorithms such as Huffman coding [7], arithmetic coding [8], or dictionary-based compression.

Among them, LZSS-based algorithms have received special interest because the compressed stream can be decoded efficiently without complex hardware. For instance, Xilinx [9] introduced a bit stream compression algorithm based on LZ77 which is integrated in the System ACE controller. Huebner et al. [10] proposed an LZSS-based technique for Xilinx Virtex XCV2000E FPGA. The decompression engine is designed carefully to achieve fast decompression. Stefan et al. [11] observed that simpler algorithms like LZSS successfully maintains decompression overhead in an acceptable range but compromises on compression efficiency. On the other hand, compression techniques using complex algorithms can achieve significant compression but incurs considerable hardware overhead during decompression.

Unfortunately, the authors did not model the buffering circuitry of the decompression engine in their work. Hence the hardware overhead presented for some variable-length coding techniques may be inaccurate. To increase the decompression throughput of complex compression algorithms, parallel decompression can be used. Nikara et al. [12] improved the throughput employing speculative parallel decoders. Qin et al. [13] introduced a placement technique of compressed bit streams to enable parallel decompression. However, since the structure of each decoder and buffering circuitry are not changed, the area overhead is also multiplied. Most importantly, this approach does not reduce the speed overhead introduced by the buffering circuitry for VLC bit stream. In contrast, our proposed approach will significantly improve the maximum operating frequency by effectively addressing the buffering circuitry problem.

III. BIT STREAM COMPRESSION ALGORITHMS

Fig. 2 shows our decode-aware bit stream compression framework. On the compression side, FPGA configuration bit stream is analyzed for selection of profitable dictionary entries and bitmask patterns. The compressed bit stream is then generated using bitmask-based compression and run length encoding (RLE).

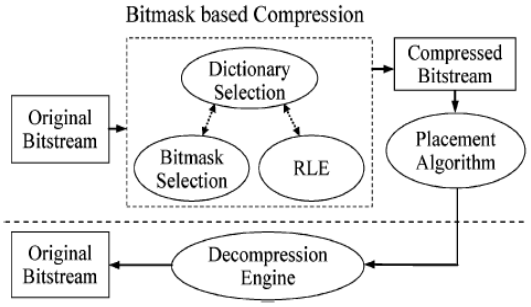
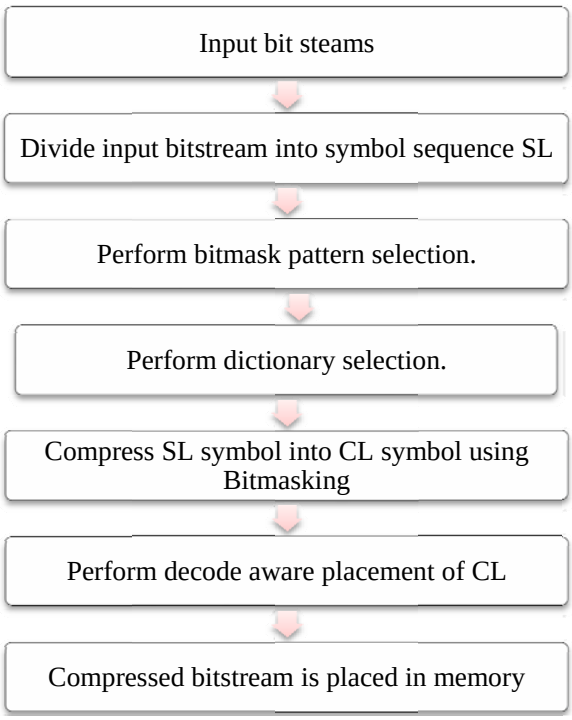


Fig.2 Bit stream Compression

Next, our decode-aware placement algorithm is employed to place the compressed bit stream in the memory for efficient decompression. During run-time, the compressed bit stream is transmitted from the memory to the decompression engine, and the original configuration bit stream is produced by decompression. Algorithm 1 outlines four important steps in our decode-aware compression framework (shown in Fig. 2): 1) bitmask selection; 2) dictionary selection; 3) integrated RLE compression; and 4) decode-aware placement. The input bit stream is first divided into a sequence of symbols with length of . Then bitmask patterns and dictionary entries used for bitmask-based compression are selected as described Next, the symbol sequence is compressed using bitmask and RLE. We use the same algorithm in [5] to perform the bitmask-based compression. The RLE compression in our algorithm is discussed. Finally, we place the compressed bit stream into a decode friendly layout within the memory using placement algorithm.

Since memory and communication bus are designed in multiple of bytes (8 bits), storing dictionaries or transmitting data other than multiple of byte size is not

efficient. Thus, we restrict the symbol length to be multiples of eight in our current implementation. Since the dictionary for bit stream compression is smaller compared to the size of the bit stream itself, we use d=2i to fully utilize the bits for dictionary indexing, where is the number of indexing bits.



Algorithm-1

1. *Bitmask Selection:*

Our bitmask-based compression is similar to [5], where three types of encoding formats are used. Fig. 3 shows the formats in these cases: no compression, compression using dictionary, and compression using bitmask. The selection of bitmask plays an important role in bitmask-based compression. Generally, there are two types of bitmask patterns. One is "fixed" bitmask, which can only be applied on fixed positions in a symbol. The other one is "sliding" bitmask, which can be applied at any position. For example, a 2-bit fixed bitmask ("2f" bitmask) is restricted to be used on even locations, but a 2-bit sliding bitmask ("2s" bitmask) can be used anywhere. Clearly, fixed bitmasks require less bits to encode its location, but they can only match bit changes at fixed positions. On the other hand, sliding bitmasks are more flexible, but consume more bits to encode. In other words, only a few number of bitmask patterns or their combinations are profitable for compression. Similar to [5], in our study of bit stream compression, we only use profitable bitmask patterns(1s,2s,2f,3s,3f,4s,4f).

2. *Dictionary Selection:*

Algorithm 2 shows our dictionary selection algorithm. Compared to the dictionary selection approach proposed in [5] for instruction compression, we made an important optimization at Step 5). In the original algorithm [5], any node adjacent to the most profitable node is removed, if its profit is less than certain threshold. This mechanism is designed to reduce the dictionary size. However, if the threshold is not chosen properly, some high frequency symbols may be incorrectly removed. Since the dictionary size in bit stream compression is usually negligible compared with the size of the bit stream, it is not beneficial to reduce the dictionary size by scarifying the compression ratio.

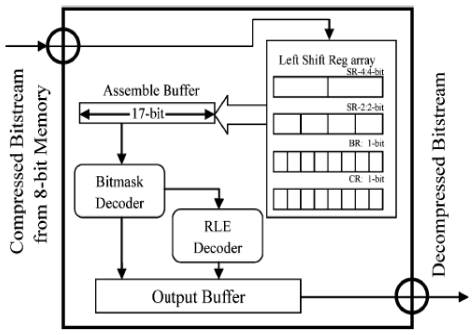


Fig.3 Decompression Mechanism

Therefore, our algorithm used new heuristics in Step 5), which carefully removes edges instead of nodes. Experimental results in Section V-A show that our approach is more suitable for bit stream compression, because we ensure better dictionary coverage.

Algorithm 2: Decode-Aware Dictionary Selection

Input: Input symbol sequence SL, Parameters $\{w, d\}$

Output: Dictionary D

Step 1: Construct graph $\mathbf{G} = (V, E)$ from SL.

Step 2: **Calculate bit savings** $S_{TS}(n)$ **of all** $n \in V$.

Step 3: Select the most profitable node N.

Step 4: Remove N from $\mathbf{G}$ and insert into D.

Step 5: **For each node** $v \in adj(N)$, **if the edge between the adjacent nodes and** N **is duplicated then remove that edge**.

Step 6: Repeat Steps 2 to 5 until D is full or $\mathbf{G}$ is empty.

return D

3. Performance Estimation

We used Xilinx Virtex-II family IP core benchmarks to analyze the results in this article. The same results are found applicable to other families and vendors too. In our experiments, Pan et al. [1] benchmarks are compressed with 32 bit symbols, 512 entry dictionary entries and two sliding 2- and 3-bit bitmasks for storing bitmask differences.Koch et al. [4] benchmarks are compressed using 16 bit symbols, with 16 entry dictionary and a 2-bit sliding bitmask.

Compression Efficiency:

We first compare our improved bitmask-compression technique with the original approach proposed in [5]. To avoid the bias caused by parameter selection, we use the same bitmask parameters for both of them.

Three different compression techniques are compared for compression efficiency: 1) bitmaskbased compression (BMC) [5]; 2) BMC with our dictionary selection technique (pBMC); and 3) BMC with our dictionary selection technique and run length encoding (pBMC+RLE). Fig. 4 shows the compression results on Pan et al. [1] and Koch et al. [4] benchmarks.

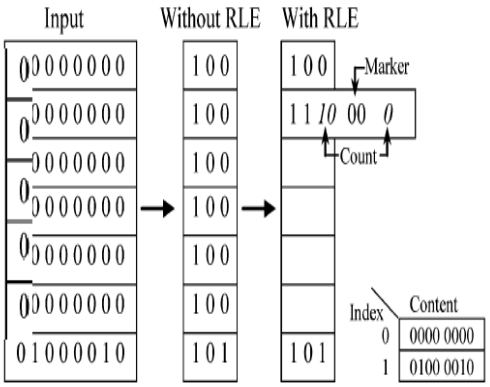
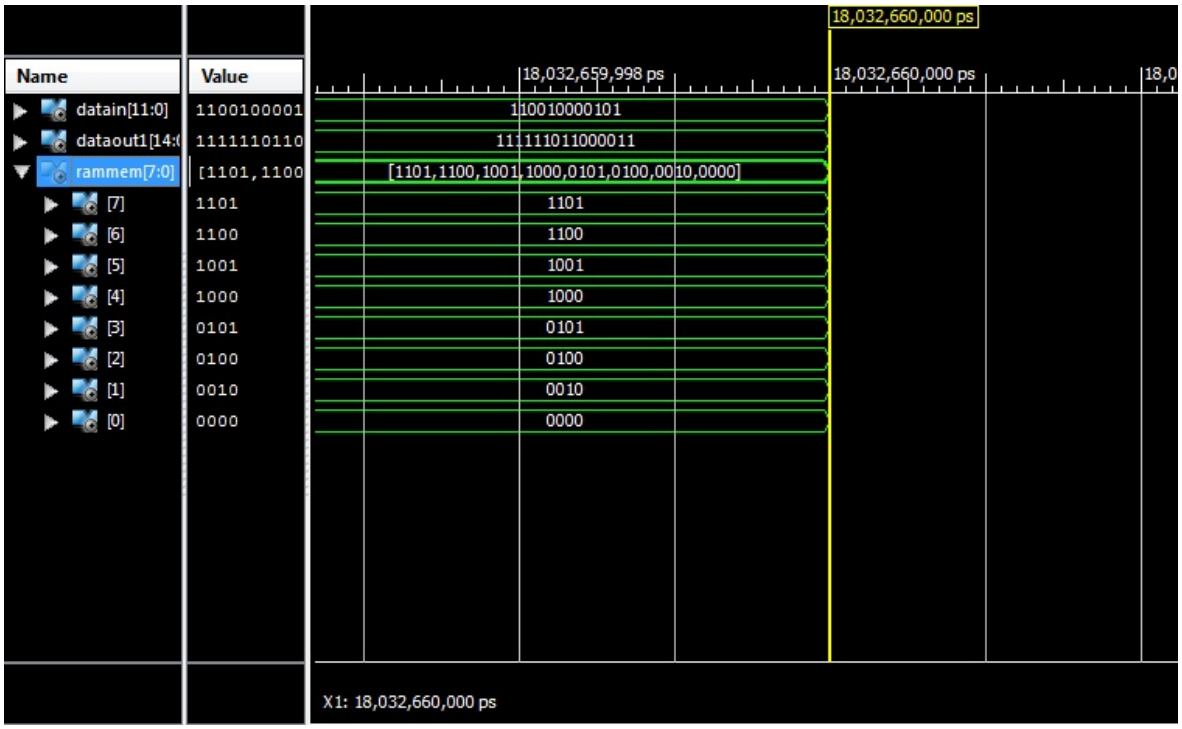


Fig. 5 : RLE based Compression

It can be seen that our dictionary selection algorithm outperform the original technique.



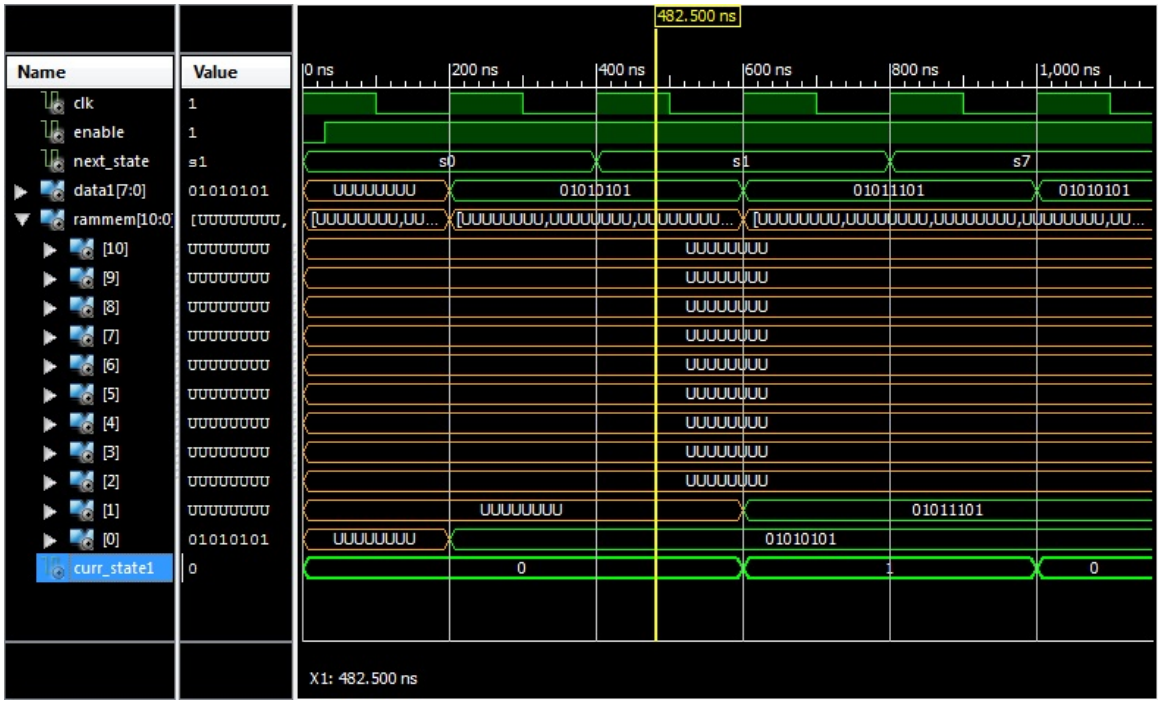


Fig.4 Simulation Results

The dictionary generated by our algorithm improves the compression ratio by 4% to 5%. Since in our approach we do not have to find the threshold value manually for each bit stream, our algorithm adaptively finds the most suitable dictionary entries for each bit stream. On the other hand, our method has the same performance.

The experimental results also illustrate the improvement of compression ratio due to the run length encoding used in our technique.

Decompression Efficiency:

We measured the decompression efficiency using the time required to reconfigure a compressed bit stream, the resource usage and maximum operating frequency of the decompression engine. The reconfiguration time is calculated using the product of number of cycles required to decode the compressed bit stream and operating clock speed. We have synthesized decompression units for variable-length bitmask-based compression, difference vector-based compression (DV RLE RB), LZSS (8 bit symbols6), and our proposed approach on Xilinx Virtex II family XC2v40 device FG356 package using ISE 9.2.04i to measure the decompression efficiency.

IV. CONCLUSIONS

The existing compression algorithms either provide good compression with slow decompression or fast decompression at the cost of compression efficiency. In this paper, we proposed a decoding-aware compression technique that tries to obtain both best possible compression and fast decompression performance.
The proposed compression technique analyzes the effect of parameters on compression ratio and chooses the optimal ones automatically. We also exploit run length encoding of consecutive repetitive patterns efficiently combined with bitmask-based compression to further improve both compression ratio and decompression efficiency. We proposed a smart placement algorithm which enables the compressed variable-length coding bit stream to be stored Our experimental results demonstrated that our technique improves the compression ratio by 10% to 15% while the decompression engine is capable of operating at 200 MHz in Virtex II FPGAs. The configuration time is reduced by 15% to 20% compared to the best known decompression accelerator [4].

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] J. H. Pan, T. Mitra, and W. F. Wong, "Configuration bitstream compression for dynamically reconfigurable FPGAs," in Proc. Int. Conf. Comput.-Aided Des., 2004, pp. 766–773.

[2] D. Koch, C. Beckhoff, and J. Teich, "Bitstream decompression for high speed FPGA configuration from slow memories," in Proc. Int. Conf. Field-Program. Technol., 2007, pp. 161–168.

[3] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to ddos attack detection and response," in DISCEX, 2003.

[4] L. Spitzner, Honeypots: Tracking Attackers. Addison-Wesley, 2002.

[5] C.Morrow http://www.secsup.org/Tracking. BlackHole Route Server and Tracking Traffic on an IP Network.

[6] http://www.snort.org. SNORT: Open-Source Network IDS/IPS.

[7] A. V. Aho and M. J. Corasick, "Efficient string matching: an aid to bibliographic search," Commun. ACM, vol. 18, no. 6, pp. 333–340, 1975.

[8] D. E. Knuth, J. H. M. Jr., and V. R. Pratt, "Fast pattern matching in strings," SIAM J. Comput., vol. 6, no. 2, pp. 323–350, June 1977.

[9] R. S. Boyer and J. S. Moore, "A fast string matching algorithm," Commun. ACM, vol. 20, no. 10, pp. 762–772, October 1977.

Authors Profile:

J.Suresh Babu

J.Suresh Babu is pursuing his masters degree in VLSI system design. He is very much interested towards digital electronics.

K.Tirumala Rao
M.Tech

K.Tirumala Rao is an Assistant professor in electronics and communication engineering in Nimra College of Engg & Tech. He got his M.Tech from JNTUK.

P.Srinivas
M.Tech

P.Srinivas Is A young and dynamic Associate Professor of electronics and communication Engineering.. He Got his M.Tech From Acharya Nagarjuna University. He Worked in various reputed Engineering Colleges as an Assoc Professor and Head Of The Department.

◈◈◈