Compression of FPGA Bit Stream using Modified Decode Aware Placement Algorithm

R.Saranya
A.S.L.Pauls College of Engineering and Technology
Coimbatore

S.Kousalya Devi
A.S.L.Pauls College of Engineering and Technology
Coimbatore

V.Lakshmi Prabha, Ph.D
Government College of Technology
Coimbatore

ABSTRACT
FPGA uses a promising technology for developing high-performance embedded systems. Reconfiguration systems widely uses Field Programmable Gate Arrays and configured using bitstream often loaded from memory. The Bitstream Compression and Decompression technique reduce the size of the bitstream and also limits the memory constraint. The Compression mechanism improves the access bandwidth for communication and thereby decreases the reconfiguration time. The Existing Approach implements the combination of techniques Dictionary Selection, Bitmask Selection and Run Length Encoding with Decode-aware Placement technique. The drawback of this approach is the extent of continuous variation of bitstream in the Run Length Encoding.

The proposed work of this paper is Golomb Encoding in place of Runlength encoding known as modified Decode Aware Compression method. Golomb Encoding is a compression technique which is capable of compressing larger size data into smaller size data. In addition, the achieved Compression Ratio is independent of the decompression hardware. It depends only on the entropy of the configuration bitstream. Finally, a time to configure FPGA depends only on the data rate of the configuration mechanism. The speed of a memory stores the configuration data, and the size of the configuration bit-stream.

Index Terms: Bitmask based Compression, decompression hardware, Golomb coding, Decode aware placement algorithm.

1. INTRODUCTION
FPGA-based embedded system uses novel compression technique to reduce the memory requirements for storing configuration bitstream which limits the capacity and bandwidth. The reconfigurable systems and application specific integrated circuits are commonly uses FPGAs, to reduce the delay in reconfiguration. There are few algorithms that offer both efficient compression ratio and fast decompression mechanisms. Figure 1 show the Standard Code Compression Methodology. The Bitmask based code compression encodes the original instruction into compressed instruction. The decompression hardware decodes the compressed bitstream from the memory to the outside buffer.

To measure the efficiency of bitstream compression, Compression Ratio (CR) is wide used as a metric.

$$Compression\ Ratio = \frac{Original\ bitstream}{Compressed\ bitstream}$$

The compression ratio should be reduced for better compression techniques.

In the work of this paper, Golomb coding is one of the lossless data compression techniques. It is capable of compressing large sized data into small sized data while still allowing the original data to be reconstructed back after decompression.

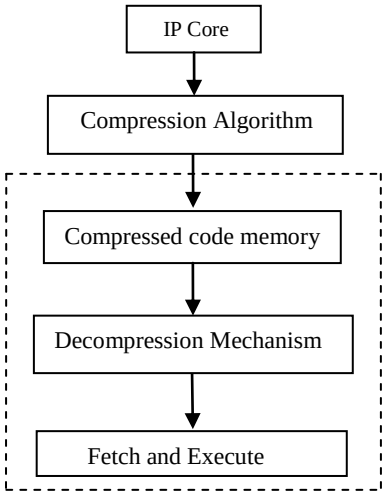

Figure1: Standard Code Compression Methodology

The rest of the paper arranges as follows. Section 2 describes the related work. Section 3 discusses the background motivation of Golomb coding. Section 4 describes the decode-aware compression techniques. Section 5 examines the experimental results. At last, Section 6 concludes the paper.

2. RELATED WORK
Wolfe and Chanin et al. proposed the first compression techniques for embedded processor using FPGA. [1], [4], [7] reports the effort related to FPGA configuration compression. The coding of compression technique in hardware description language is synthesize into schematic diagram as a logic gates in the design. The schematic diagram interconnects these gates to fit into the Look up Tables in a Field Programmable Gate Array board. To connect the gates together while opening or closing the switches in the routing matrices, the Place and Route tool assign the gate collections to a Configuration Logic Blocks. The bitstream generated is downloaded into Xilinx FPGA.

In contrast, the compressed bitstream techniques only access the configuration memory linearly during decompression. In FPGA, split the given bitstream sequence into a small word of

8-bit. Then compress the bitstream with statistical methods such as Dictionary, Bitmask and Golomb coding techniques. The compression ratio improves, by combining the techniques together. During decompression, the compressed bitstream can be decoded capably without complex hardware.

3. BACKGROUND MOTIVATION

The existing bitstream compression technique called Run length encoding addressed a problem of continuous variation in terms of 1's and 0's. To overcome the problem, a new technique proposed called Golomb Coding. Assume a parameter value *s* for a given data set that decides variable length code structure in terms of compression efficiency. Split the group as G1, G2, G3, and so on according to the parameter fixed. The Prefix represents the (k-1) number of one's followed by zero (i.e.) attach the binary value 1 in front of the prefix value of G1=0. While each member of the group is given a tail, which is the binary representation of zero's until (*s*–1). Then grouping the prefix and tail to produce a codeword. Table 1 describes the run length encoding and Golomb coding technique. According to the table 1, Split the dataset which shows the runs of one's awaiting the code is finished with a zero is formed. So to encode the Run length by Golomb coding we used a codeword to replace it. Binary strings can be divided into subsets of binary strings and replacing the subsets with the equivalent codeword.

Table 1: Determination of Run Length and Golomb Coding

Dataset	10 1111110 1110 111110 110 0					
Subset	10	1111110	1110	111110	110	0
Runlength	1	6	3	5	2	0
Golomb	101	0110	111	0101	110	100

4. DECODE - AWARE BITSTREAM COMPRESSION

The five important steps in the decode-aware compression framework are:

1) Dictionary selection,
2) Bitmask selection,
3) Golomb Coding compression,
4) Decode-aware placement, and
5) Decompression Engine.

To compress the configuration bitstream, we introduced a new technique called Dictionary, Bitmask, and Golomb coding on the compression. After compression, the compressed bitstream in the memory are transferred into decode-aware placement technique. Decode aware placement algorithm is used to place the compressed bitstream into memory. The decompression engine gathers all the compressed bitstream from the memory and produces an original configuration bitstream, during Dynamic Encoding.

Modified Decode-aware Bitstream Compression Algorithm

Input: Input bitstream
Output: Compressed bitstream placed in Memory.
Step 1: Split the input bitstream into 8-bit bitstream.
Step 2: Perform dictionary selection.
Step 3: Perform bitmask pattern selection.
Step 4: Compress the 8-bit code into code sequence using Bitmask and Golomb coding.
Step 5: Perform decode aware placement technique.

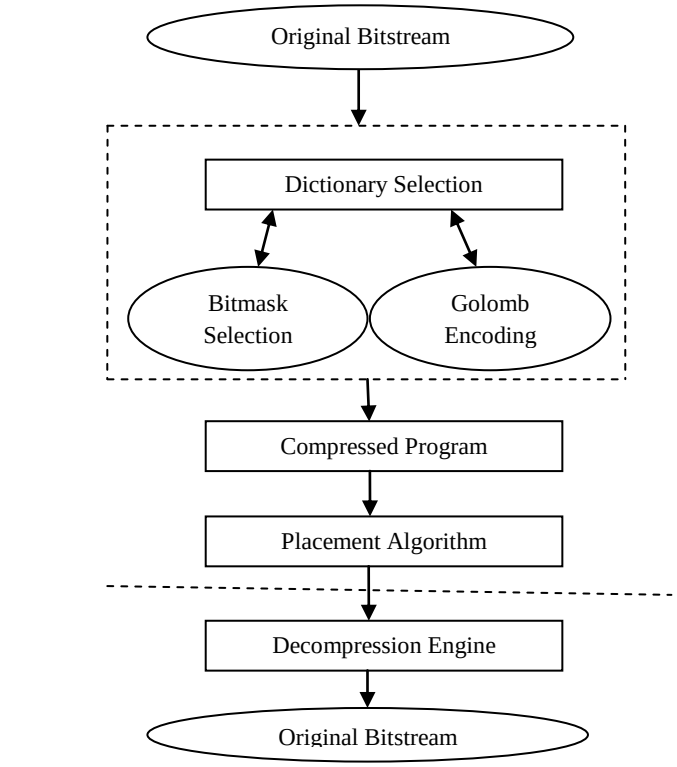

Figure2: Decode-aware bitstream compression framework

4.1 Dictionary Selection

Dictionary-based code compression techniques provide both good compression ratio and fast decompression mechanism.

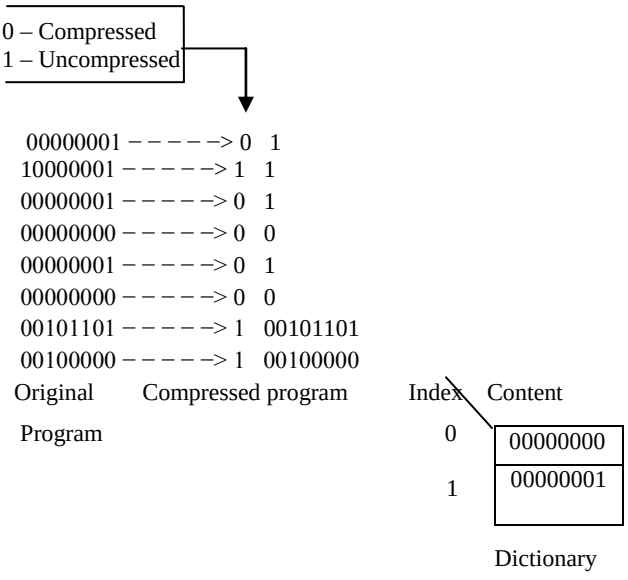

Figure 3: Dictionary based code compression

The Dictionary corresponds to configuration data and the index corresponds to the way the dictionary is read in order to reconstruct a configuration bit-stream. The Dictionary and Index are derived based on the LZW compression algorithm. Figure 3 shows the example of Dictionary based code

compression. Initially assign an instruction sequence as an input. Split the bitstream into 8-bits. If the instruction contains repeated occurrence, then replace the instruction by a codeword that points to index of the dictionary. The compressed program consists of both code words and uncompressed instructions.

4.2 Bitmask Selection

The Bitmask-based code compression technique improves the standard dictionary-based code compression technique by considering mismatches. Figure 4 shows the example of Bitmask-based code compression.

The three possible cases in bitmask are:

1) The first bit in the compressed program represents the compressed (0) or uncompressed bitstream (1). The second bit indicates the compressed using bitmask resolve (0) or not (1). The last bit in the program indicates the dictionary index.

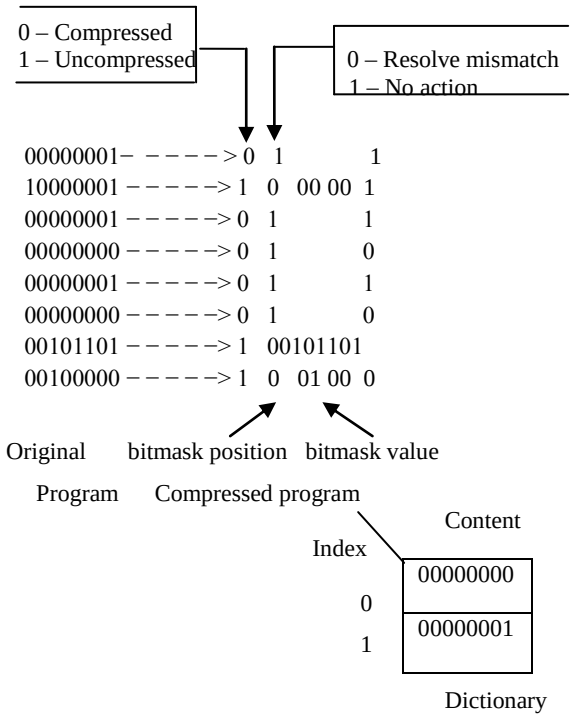

Figure 4: Bitmask Based Compression

2) While using the bitmask, the compressed bitstream requires only 7-bits. The first bit represents the compressed or uncompressed bitstream and Bitmask action represents the second bit. The next 4-bit describes the Bitmask Position and Bitmask Pattern. The last bit represents the Dictionary Index.
3) The Uncompressed bitstream represents that there is more than 1-bit changes.

4.3 Golomb Coding

The configuration bitstream usually contains step by step repeating bit sequences. Golomb Coding is a new technique used instead of Run Length encoding to avoid continuous variation during compression. Golomb Coding (GC) yields a better compression results than bitmask-based code compression.

Golomb coding [7] comes under lossless data compression technique and reduces the code redundancy. It is competent of compressing larger sized data into a smaller sized data while still allowing the original data to be reconstructed back after decompression.

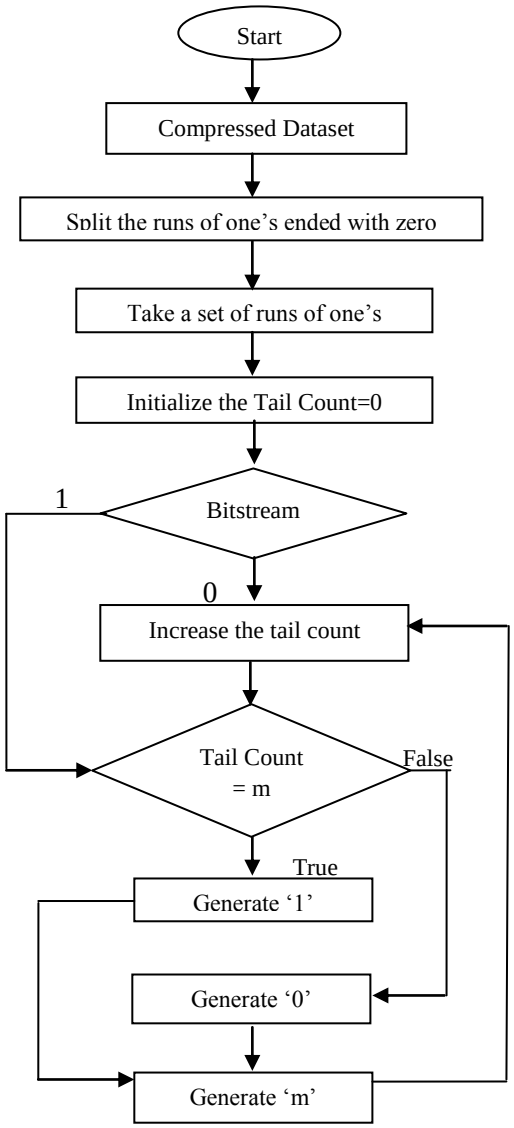

Figure 5: Golomb Encoder flowchart

The number of 1's in the data-set controls the tail count. As in figure 5, If the bitstream examine the zero, then increase the tail count proportionally until it reaches the parameter m and produce the output as '1'. If the input data is '1', the algorithm will generate a '0' which acts as the divider between the prefix and the tail, and output the current tail count as the tail of the encoded string. The algorithm will then reset the tail count and waits for the next input data.

4.4 Decode-aware Placement Algorithm

The placement algorithm places all bitmask codes in the memory so that they can be compressed using the efficient decompression hardware. First, split the original single Variable Length Coding bitstream into multiple Fixed Length Coding bitstream for storage. Bitmask decoding generates the configuration bitstream by buffering the FLC bitstream separately during decompression. When compared to VLC bitstream, the buffering circuitry for FLC bitstream is much

simpler. While adding multiple FLC buffers on the decompression engine, the performance level still increased.

Definition 1: Power-Two n-bit stream ("PT-n stream" for short) is FLC stream of n-bit codes, where n is a power of two such as 2^0, 2^1, 2^2, etc. While each code in a PT-n stream has the similar length. Subsequently the shift distances are stable when a PT-n stream is buffered.

4.5 Decompression Engine

The decompression engine is a hardware component used to decode the compressed configuration bitstream and feed the uncompressed bitstream to the configuration unit in FPGAs. A decompression engine divided into two parts.

1. Buffer and align the codes fetched from the memory by using buffering circuitry.
2. Using multiple decoders, decompression engine generates configuration bitstream.

The figure 6 shows the structure of decompression engine for 8-bit memory. The working of "Assemble Buffer with a Left Shifter Array" is to use an array of left shift registers to buffer the power-two bitstream separately. Bitmask-based compression represents the first two bits of a code as (is-compressed and is-bitmask flags) and checks the code length. The bitmask and Golomb decoder produce an uncompressed data by fetching the compressed data from an assemble buffer. When the shifter becomes empty, the decompression algorithm alerts the incoming memory lines to load the data correctly.

From the decompression algorithm, let the maximum code length and maximum bandwidth be *'l'* and *'b'*. Assemble buffer directly receive the PT-b stream and contains log2 (b) + 2 shift register, including two 1-bit Compressed shift register, Bitmask shift register used to buffer CS and BS. Log_2 *(b)* shift register *SR-1, SR-2, …, SR-b/2* used to buffer streams *PT-1, PT-2, PT-4, .., PT-b/2*. Each stream has the capacity of b bits same as the memory bandwidth. The size of the assemble buffer AB is l because AB only holds one code at a time.

Field Programmable Gate Array implements the Assemble Buffer Left Shift Array requires less area and shorter critical path length.

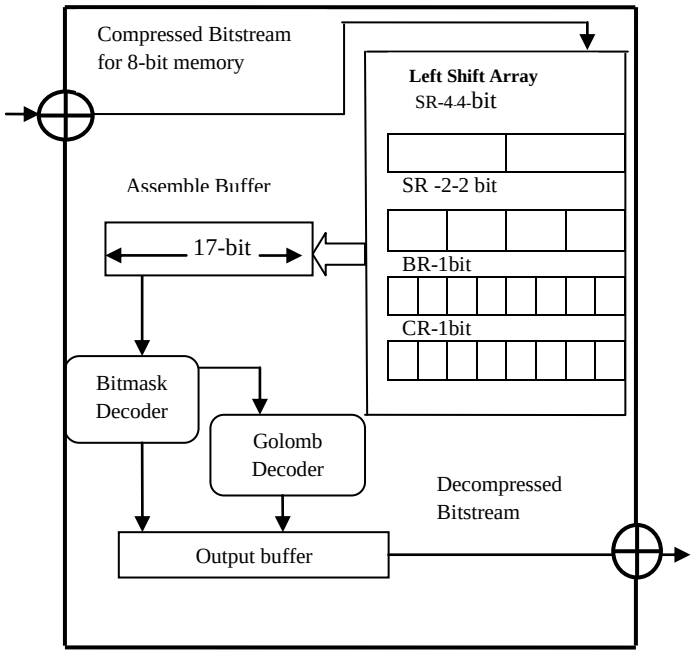

Figure 6: Decompression Engine.

5. EXPERIMENTAL RESULTS

In this section, experimental results are presented. Input bitstream is compressed using Dictionary, bitmask based compression and Golomb Coding technique. The compressed bitstream is stored in the main memory and retrieved back as an original input using decompression hardware. Here, these techniques were simulated in Modelsim and synthesized using Xilinx. To get hold of a better Compression Ratio, we club the compression performance with one another. Table 2 describes the Compression Efficiency for Different Bitstream. The compression ratio is obtained based on the difference between original and compressed bitstream.

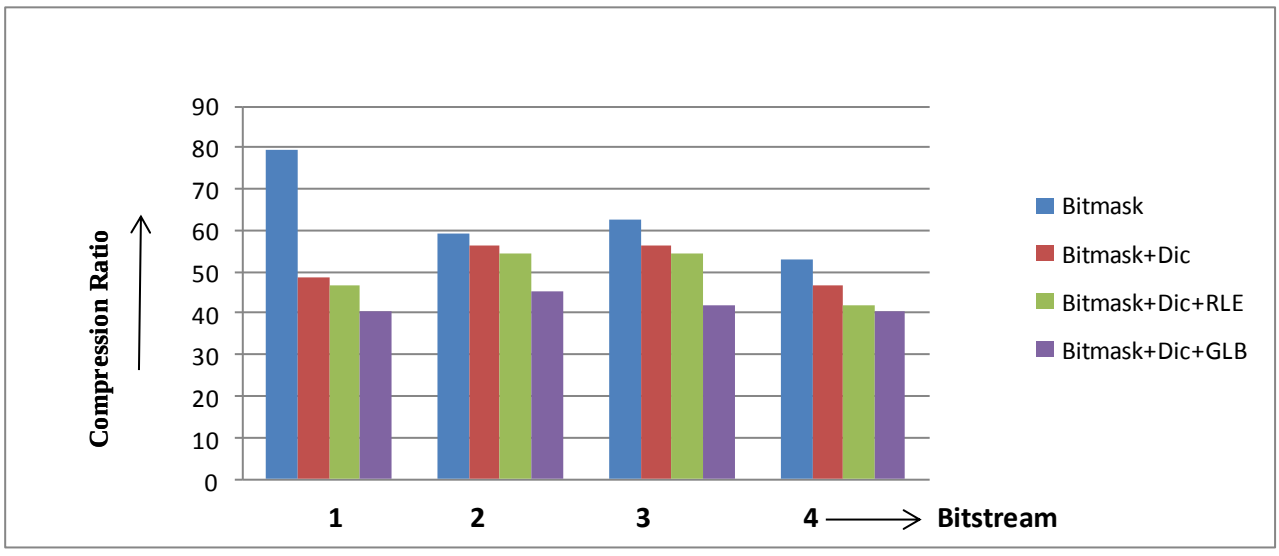

Figure 7: Compression Ratio for different bitstream

Table 2: Compression Efficiency for Different Bitstream

Compression Techniques	Bitstream 1	Bitstream 2	Bitstream 3	Bitstream 4
Bitmask	20.3	40.6	37.5	46.8
Dictionary+ bitmask	51.6	43.7	43.7	53.1
Dictionary+ bitmask+ RLE	53.1	45.3	45.3	57.8
Dictionary+ bitmask+ Golomb	59.4	54.6	57.8	59.3

Figure 7 shows that the existing bitstream compression techniques outcomes with 0 - 55% work of the compression ratio. While the proposed method outfit that the compression ratio improves from 0 - 45% compared to the existing work. Thus, a smaller compression ratio implies a better compression technique.

5. CONCLUSION

The work of this paper includes a novel configuration compression technique. The purpose is to reduce memory compulsorily to store configuration bitstream in FPGA-based embedded systems. The existing configuration bitstream techniques provide good compression ratio with slow or fast decompression efficiency. In the proposed work, we comprise Golomb coding technique instead of Run Length encoding. Golomb coding of consecutive repetitive patterns is efficiently combined with bitmask and dictionary based code compression, to improve the compression ratio and decompression efficiency. During decompression, to reduce the hardware overhead we introduced a technique called placement algorithm. It enables the compressed variable length coding bitstream to be stored and buffered separately in the form of multiple fixed length coding streams. The buffering circuitry for fixed length coding will reduce the area and delay of the decompression engine. As a result, the compression technique improves the compression ratio up to 45%, while the decompression engine operates at closest to the Field Programmable Gate Array.

6. ACKNOWLEDGMENTS

I cordially thank my co-authors to complete my study of the work.

7. REFERENCES

[1] Drisya. M. K and Senoj Joseph (2012), 'Compression of FPGA *Bitstreams – A Comparsion,' Bonfring International Journal of Power Systems and Integrated Circuits.*

[2] Chetan Muthry, Prabhat Mishra, and Xiaoke Qin March (2011), 'Decoding – Aware Compression of FPGA Bitstreams,' *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*.

[3] Chetan Murthy and Prabhat Mishra (2009), 'Lossless Compression using Efficient Encoding of Bitmask,' *IEEE Computer Society Annual Symposium on VLSI*.

[4] Beckhoff. C, Koch. D and Teich. J (2007), 'Bitstream Decompression for High Speed FPGA Configuration from Slow Memories,' *in Proc. ICFPT*.

[5] Cotofana. J and Stefan. R (2008), 'Bitstream compression techniques for Virtex 4 FPGAs,' in *Proc. Int. Conf. Field Program. Logic Appl.*

[6] Dandalis. A, Prasanna. V. K (2005), 'Configuration compression for FPGA-based embedded systems,' *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*

[7] H'ng. G. H, Halim. Z. A and Salleh. M. F. M (2008), 'Golomb Coding Implementation in FPGA,' *http://fke.utm.my/elektrika*.

[8] Hauck. S and Wilson. W (1999), 'Runlength compression techniques for fpga configuration,' *in Proc. FCCM*.

[9] Kanad Basu and Prabhat Mishra (2008), 'A Novel Test-Data Compression Technique using Application-Aware Bitmask and Dictionary Selection Methods,' *Published by ACM*.

[10] Kanad Basu and Prabhat Mishra (2010), 'Test Data Compression Using Efficient Bitmask and Dictionary Selection Methods,' *IEEE Trans. VLSI Syst*.

[11] Li. L and Touba. N. A (2003), 'Test data compression using dictionaries with selective entries and fixed-length indices,' *ACM Trans. Des. Autom. Electron. Syst.*

[12] Mitra. T, Pan. J. H and W. F. Wong (2004), 'Configuration bitstream compression for dynamically reconfigurable FPGAs,' in *Proc. Int. Conf. Comput.-Aided Design.*

[13] Mishra. P and Seong. S. W (2006), 'A Bitmask-Based Code Compression Technique for Embedded Systems,' *in Proc. ICCAD*.