International Journal on Recent Researches in Science, Engineering & Technology (IJRRSET)
A Journal Established in early 2000 as National journal and upgraded to International journal in 2013 and is in existence for the last 10 years. It is run by Retired Professors from NIT, Trichy. Journal Indexed in JIR, DIIF and SJIF.
Research Paper
Available online at: www.jrrset.com
UGC Approved Journal No: 45483
Received on: 02.02.2018 Published on: 28.02.2018 Pages : 67 - 79

ISSN (Print) : 2347-6729
ISSN (Online) : 2348-3105
Volume 6, Issue 2
February 2018.
JIR IF : 2.54
DIIF IF : 1.46
SJIF IF : 1.329

MIXED BIT SAVING DICTIONARY SELECTION ALGORITHM FOR CODE COMPRESSION RATIO TO FIELD PROGRAMMABLE GATE ARRAY

B. Jareena Banu[1a*], D. Sobya[2b], S. Nallusamy[3c]

[1]Lecturer, Department of Instrumentation and Control Engineering, Government Polytechnic College, Udhagamandalam, Nilgiri-641014, Tamilnadu, India
[2]Research Scholar, Department of Adult and Continuing Education and Extension, Jadavpur University, Kolkata-700032, India
[3]Professor, Department of Mechanical Engineering, Dr. M G R Educational and Research Institute, Chennai-600095, Tamilnadu, India

[a]e-mail: jareenabanu67@gmail.com, [b]e-mail: sobyadevaraj@gmail.com,
[c]e-mail: ksnallu@gmail.com

Abstract

The aim of this research is to obtain an efficient code compression ratio with less hardware overhead using a multiple lookup table's access with Index number. It can help to reduce the memory usage in field programmable gate array architecture. Dictionary based compression algorithm is used to design a lookup tables. The large lookup tables are used to compress single instructions and small lookup tables is used to compress the extremely high-frequency instructions. The Tag bits are used to identify whether the code is compressed with direct dictionary access or Bit-mask approach. Mixed bit saving dictionary selection algorithm is used to build a dictionary based on the frequencies of the instructions. Codeword length constrained bitmask code compression is an algorithm used for compressing the code. The decompression engine was designed to retrieve the original instructions. The code compression algorithm is simulated and synthesized in Quartus II 10.0 tool. The code compression ratio is measured for large number of instructions. The Decompression engine is simulated and synthesized in Xilinx 14.2 tool. From the results it was observed that the proposed dictionary based code compression provides high code compression ratio with less overhead of hardware architecture for decompression.

Keywords: Code Compression Ratio, FPGA, MBSDSA, Lookup Table, Xilinx 14.2

1. Introduction

Memory consumption is a key factor for designing a Field Programmable Gate Array (FPGA). An FPGA design plays a vital role in real time applications, and is widely used worldwide. Literatures survey reveals that earlier works are focusing on to reduce memory for large number of instructions using Bitmask method with high hardware overhead. Memory is one of the key factors that affect cost, power consumption and memory consumption [1-5]. Code compression is a technique used in embedded systems to reduce the memory usage. Bit-mask based code compression is a customized version of dictionary based code compression. The basic of Bit mask is to trace the divergence values and their positions to reduce a greater number of instructions; it can be used exclusively or incorporated with the reference instructions to decode the codeword's [6-11]. Code compression to reduce the program size to reduce the memory usage was proposed. In recent research not only the memory consumption but also the power consumption plays an important factor [12-19]. The compression ratio (CR) is defined ratio between the sum of compressed program size and decoding table size to the original program size. It is used to determine amount of memory conserved.

$$\mathrm{CR} = \frac{\text{Compressed Program} + \text{Decoding Table Size}}{\text{Original Program Size}}$$

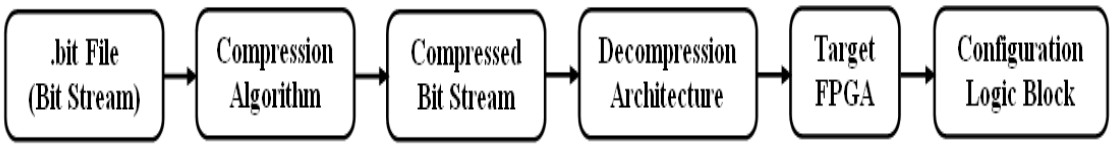


Figure 1Traditinal FPGA Reconfiguration with Compression

Therefore, larger compression ratio betters the compression technique. Dictionary based code compression techniques are idely used for better compression ratio and fast decompression mechanism. The basic idea is to exploit repeating instructions sequence by using a dictionary. In all the existing code compression technique does not provide better compression ratio for all benchmarks [20-23]. In this article all the code compression technique are combined to make a new code compression algorithm which can yield better compression ratio using distributed dictionaries. Figure 1 shows the traditional approach for FPGA reconfiguration with compression. The bit stream from the .bit file is processed in the compression algorithm. The compressed bit streams are placed in the memory and they are decompressed using decompression hardware before targeting the FPGA hardware.

The direct approach of this application is not suitable in choosing a word length or number, size and type of bitmasks or dictionary size. Therefore it is more challenging to introduce the better compression technique which significant improvement in compression ratio. In the earlier work application of specific bitmask selection and bitmask-aware dictionary selection but it became more challenge-able. Improved standard dictionary-based compression by considering mismatches was proposed [24-27]. A new method of code compression for embedded systems called as Compressed Code using Huffman-Based Multi-Level Dictionary (CCMLD) is proposed [28-32]. The Codeword Length Constrained Bitmask Code Compression (CLCBCC) with MBSDS algorithm is proposed which can leads to achieve efficient compression and saving ratio by deploying separate dictionary [33-36]. Our proposed work reduce the number of tag bit to one bit for the compressed and uncompressed instructions which can further provide the high saving ratio when compared to all the other compression technique.

The article is structured in the following sections. In second section block diagram, the existing Dictionary Based Code Compression (DCC) is discussed in third section, fourth section provides Bit Mask Code Compression (BCC), fifth section discuss about Mixed Bit Saving Dictionary Selection (MBSDS) and details about CLCBCC. The Proposed modified CLCBCC with MBSDS are discussed in sixth section. The architecture for decompressed engine is discussed in seventh section. Eighth section VIII provides the results of the proposed algorithm and they are discussed and finally section nine reveals the conclusion of this article with its future scope.

2. Block Diagram

The proposed code compression is based on the frequency distribution between various instructions. The Figure 2 shows the outside and inside FPGA process, the original bit stream are fetched from the cache memory and compressed with the help of distributed dictionary. Inside the FPGA a specially designed decompression engine is designed and original instruction are retrieved from the original instruction.

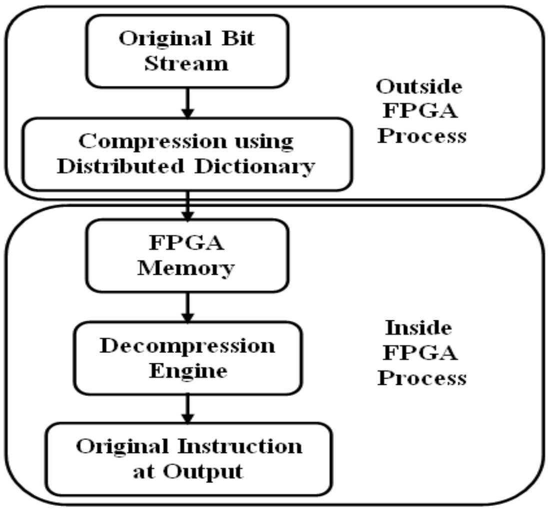


Figure 2 Block Diagram for Code Compression

The generated bit file is 8-bit size in general, so we fixed the instruction size as 12-bit. The 2-bit is used as a tag bit for identifying the compression technique. Therefore the Distributed dictionary based code compression with bitmask code compression algorithm was introduced for both small and large bench marks so we can restrict the length as a multiple of 8-bit in this proposed compression technique. Therefore the dictionary size is small when compared with bit stream compression. The data are processed from the cache memory which can provide faster access instead of in-built memory which reduces the speed.

3. Dictionary based Code Compression

Dictionary based code compression is the method in which the Lookup Tables (LUT) are constructed based on their frequent data occurrence. The most frequently occurred data are stored in small LUT with index number. Similarly the data are inserted in large LUT with frequent data repetition until the LUT is full. The tag bits are used to identify whether the data are compressed with LUT access or not. The dictionary based code compression provides efficient code compression with less hardware overhead. The following examples in Figure 3 shows the dictionary based code compression which provide clear knowledge how it works. The following example has a compression ratio of 2.5%. The original program has 8-bit code size and they are compressed with the help of value stored in LUT and access by index number. The next section shows a bit mask code compression.

4. Bit Mask Code Compression

Bitmask based compression an enhancement on the dictionary based compression scheme, which helps us to get more matching patterns as shown in Figure 4. In dictionary based compression, each data is compressed only if it completely matches with a dictionary entry. The data that area directly coordinated are compressed with 3 bits. The first bit represents whether it is uncompressed (using1) or compressed (using0). These condbit shows whether it is compressed using bit mask (using0) or not (using1). The last bit indicates the dictionary index. Data that are compressed using bit mask requires 7 bits. The data compressed by using bitmask approach, are indicated by initial two bits. The next two bits indicate position of the mask and followed by two bits that indicate the value of bit in that position. The data which is different for more than 1 bit is left uncompressed. Dictionary-based code compression is commonly used in FPGA systems, because it can reach a proficient CR, possess a fairly straight forward decoding. A new algorithm is to develop the compression concert (including the CR) with a lesser hardware operating cost. Based on the Bit Mask code compression algorithm, a tiny divided dictionary is projected to limit the code word length of high-frequency instructions, and a original dictionary selection algorithm is proposed to achieve more satisfactory instruction selection, which in turn may reduce the average CR. The corresponding examples show how it works with Bitmask approach.

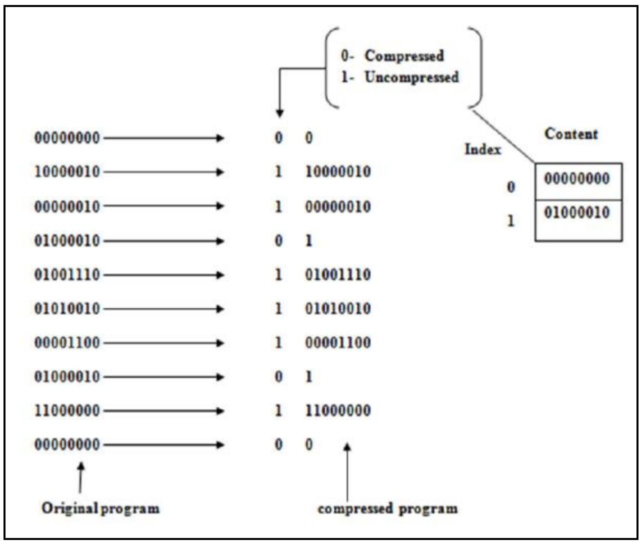


Figure 3 Dictionary Code Compression

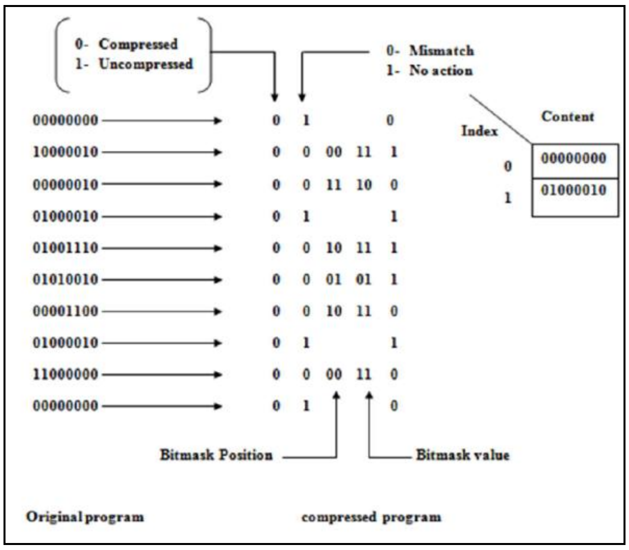


Figure 4 Bit Mask Code Compression

5. Mixed Bit-Saving Dictionary Selection

Frequency dictionary selection cannot achieve efficient CR in bitmask code compression; because it cannot guarantee that the matched rate is maximized. The mixed bit saving dictionary selection is used to built a dictionary based on the frequently occurred data and it frequency distributions among each node and the following cases to build a dictionary.

Case 1: A high frequency node mostly connects to a high frequency node.
Case 2: A high frequency node mostly connects to low frequency nodes.
Case 3: A low frequency node mostly connects to high frequency nodes.
Case 4: A low frequency node mostly connects to low frequency nodes.
Case 5: A low frequency node with few connections.

The case 1, 2 and 4 are better choices for the improvement for CR and case 2 can provide better saving. The case 3 are limited because they are not suitable for dictionary selection. The case 5 never can be used in dictionary selection because of its low frequency and few connections results in poor saving. The proposed algorithm for MBSDS is shown below by which the dictionary is selected.

Inputs:
1. 32-bit unique instruction vector.
2. Dictionary size
3. Mask types

Output: Optimized dictionary

Begin
Step 1: Transform every unique instruction to a graph, G =(V, E). If two nodes can be matched by Bit Mask, use directional edges to connect them.
Step 2: Allocate bit saving to the node and edge**.**
Node saving = (original instruction size-compressed codeword size) * (frequency of the instruction - 32 bits Overhead)
Edge saving (Wij) = (original instruction size –compressed Code word size) * frequency of the matched instruction
Step 3: Calculate the total bit saving distribution of all Nodes.
Step 4: Select the most profitable node.
Step 5: Remove the most profitable node from G and insert it into the dictionary.
Step 6: For each node connects with the most profitable node, all its edges are removed.
Step 7: Repeat Steps 3 - 6 until the dictionary is full.
Step 8: Return Dictionary

The above algorithm first transforms every unique instruction into individual nodes and they are connected with the help of above cases like 1, 2, and 4. The following example shows how to proposed algorithm process takes place. The Figure 5 shows an example of selection of dictionary using MBSDS. All the nodes represented here are 32-bit wide, and they have their own frequency distribution. The highest frequency node are removed from the graph using node saving and edge saving equations and they are updated simultaneously. The node saving for every node equaled to 32-12 code word length 10 + tag width 2, which was multiplied by its frequency. The edge saving equaled to 32-18 code word length10 + tag width 2 + bitmask 6, which was multiplied by the frequency of the matched node.

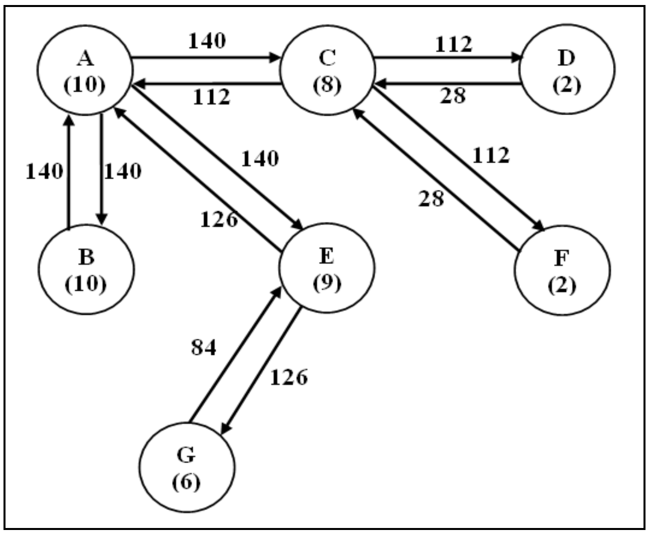


Figure 5 Mixed Bit Saving Dictionary Selection

Bit Savings:

Sa = 200+140+126+112=578
Sb= 200+140=340
Sc = 160+28+28=216
Sd= 40+112=162
Se = 180+140+84=404

Sf = 40+112=152
Sg = 120+126=246

The bit saving for all nodes are calculated, from that we can clearly find node A having highest frequency distribution among the all the others nodes. Then the node A is first inserted in the LUT and it is removed from the graph. Once the node A is removed all the other nodes are updates by using node saving equation. The update graph is shown Figure 6.

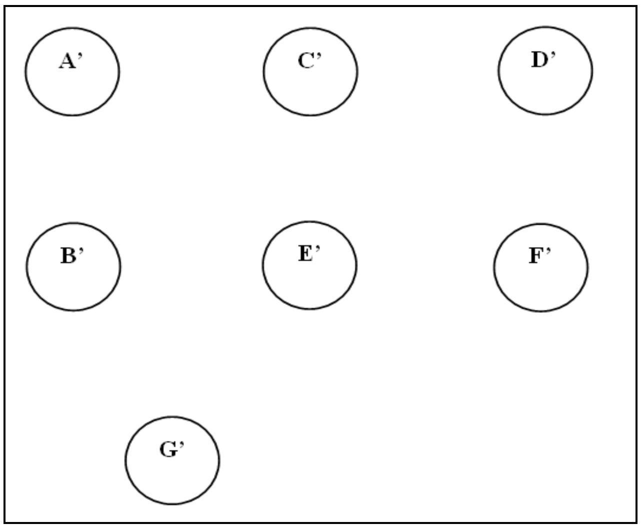


Figure 6 Update Node after Node A Selected

After Selection Node A:

Sb' = Nb'= Nb - Wab = 200-140 = 60
Sc' = Nc'= Nc - Wac = 160-112 = 48
Sd' = Nd= 40
Se' = Ne' = Ne - Wae = 180-126 = 54
Sf' = Nf=40
Sg' = Ng'= 120

The 'S' shown the updated node saving value from this the next highest frequency node is inserted in LUT and the process is repeated until the LUT is full.

5.1 CLCBCC with MBSDS

The codeword constraint length bit mask algorithm and the mixed bit saving algorithm was combined to perform the compression process. The small LUT and large LUT are designed with respect to the frequency distributions among the instructions. The algorithm for CLCBCC with MBSDS is shown below

Input:
1.8 bit FPGA bit streams
2. Small dictionary size
3. Big dictionary size
4. Mask types

Output: CR and compressed codeword

Begin
step1: calculate the frequency distribution of all instructions symbols
Step2: select the highest frequency symbols into the small LUT based on step1.
Step3: for every unique instruction symbols which are not selected into the small dictionary, use the MBSDS to construct big dictionary.

Step4: Use the bitmask based method to compress all instructions based on the current dictionaries and masks setting.
Step5: Calculate the CR.
Step6: Return the compressed code words and CR.
End

The large LUT are used to compress the single instructions and the small LUT are used to compress the extremely high frequency instructions. The number of tag bits was increased with 1-bit with slightly hardware overhead. The index number are used for the defining the LUT contains more than one dictionaries entry.

5.2 **Architecture of CLCBCC**

The architecture for the CLCBCC for the distributed dictionary is shown below in the Figure 7. The selector is used for selecting the whether the instructions should be entered into big LUT or small LUT.

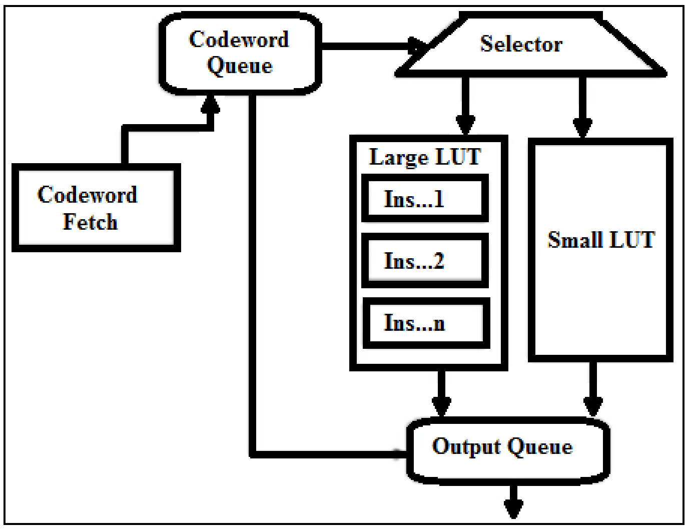


Figure 7 CLBCC Architecture

The code word is fetch from the memory and all the instructions are placed in the queue. There are three different instructions are at the output one is from big LUT, next is from small LUT and last is an uncompressed instructions.

5.3Encoding Format for CLCBCC with MBSDS

The encoding format for the CLCBCC with MBSDS is 4f mask and the 4f-2f masks are combined, and 1-bit is used to indentify whether the codeword uses one or two masks. The encoding format is shown in Figure 8, which contains four situations, such as uncompressed, matched with small dictionary, matched with large dictionary, and matched using a variable number of masks.

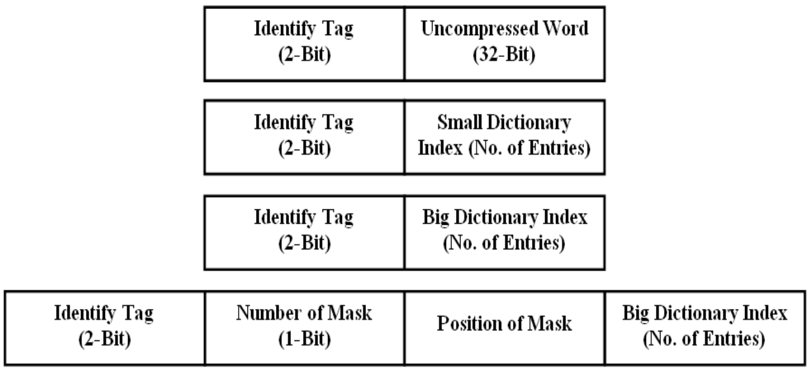

Identify Tag (2-Bit)
Uncompressed Word (32-Bit)
Identify Tag (2-Bit)
Small Dictionary Index (No. of Entries)
Identify Tag (2-Bit)
Big Dictionary Index (No. of Entries)
Identify Tag (2-Bit)
Number of Mask (1-Bit)
Position of Mask
Big Dictionary Index (No. of Entries)

Figure 8 Encoding Format

The example for CLCBCC with MBSDS is shown Figure 9, in which the highest frequency node are inserted in the small LUT and he next highest frequency is inserted in the large LUT until the LUT is empty. The CLCBCC with MBSDS provide a proficient compression ratio when compared with all other compression algorithm. Then the code is compressed with comparing the content with in the dictionary and with the input vectors. Once the input vectors are compressed with the small LUT, it is indicated by the tag bit 01. If the input vectors are compressed with the Large LUT, it indicated by the tag bit 10 with the index number either 0 or 1. The input vectors are compressed with either dictionary access or with Bitmask approach; it is indicated by the tag bit 00. The input vectors are compressed using bitmask approach; the first two bit 11 indicates using bit mask approach; next two bits indicates the mask position 4f (00,01,10,11), the next two bits indicates mask value using XOR operation and last bit indicates the index value.

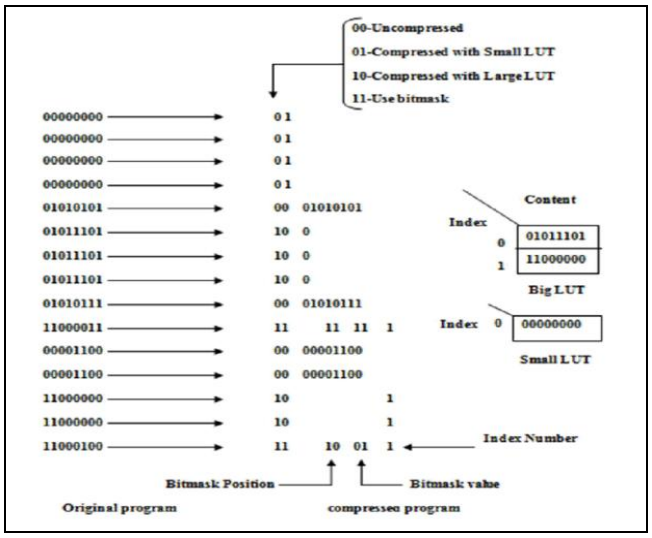


Figure 9 CLCBCC with MBSDS

6. Modified CLCBCC WITH MBSDS

In this article a modified version of CLCBCC with mixed bit saving dictionary selection is proposed. The number of tag bits are reduced further which can leads to more efficient code compression ratio. The 1-bit comparator and 3-bit comparator are used in decompression side to retrieve the original instructions. The bit 0 denotes compression of data with small LUT. The bit 1 denotes uncompressed data. The bit 10 and 11 denotes compression with large LUT and using bitmask approach respectively. But in this case to avoid bit flip error we can used an extra one bit to reduce the error. Overall compression ratio is reduced greatly when compared to other compression techniques using dictionary method. The following Figure 10 shows how the modified CLCBCC with MBSDS works.

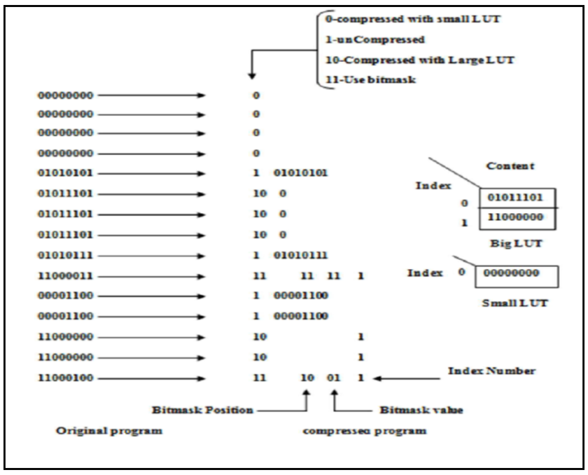


Figure 10 Modified CLBCC with MBSDS

From the above proposed technique the overall compression ratio for the provided example is 75% when compared to all other technique. The Figure 9 and 10 shows how the memory can be reduced using the proposed algorithm CLCBCC and MBSDS and its improved version. It shows that the data are compressed and decompressed in a lossless manner, so it can be used in many application regions such as in digital image processing for image compression, for high security process like encryption and decryption, in lower power consumption circuits etc.

7. Decompression Engine

The proposed decompression engine was implemented by Verilog and Xilinx. Decompression units consist of a control unit, LUT's de-multiplexer, shift buffers, and the bitmask unit as shown in Figure 11. The control units assign the work to all the units and its control the decompression operation. The frequency instructions are placed in the small LUT to provide the fast decoding with less codeword length. The randomly used instructions are placed in the large LUT. The output queue
stores the decompressed instructions and delivering them to the processor or cache.

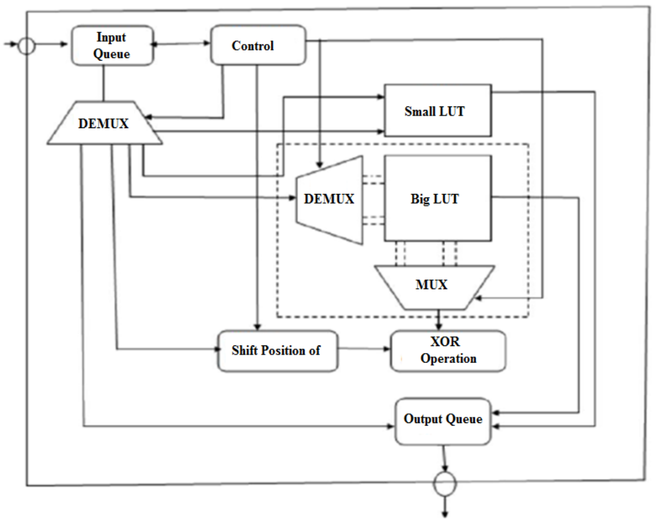
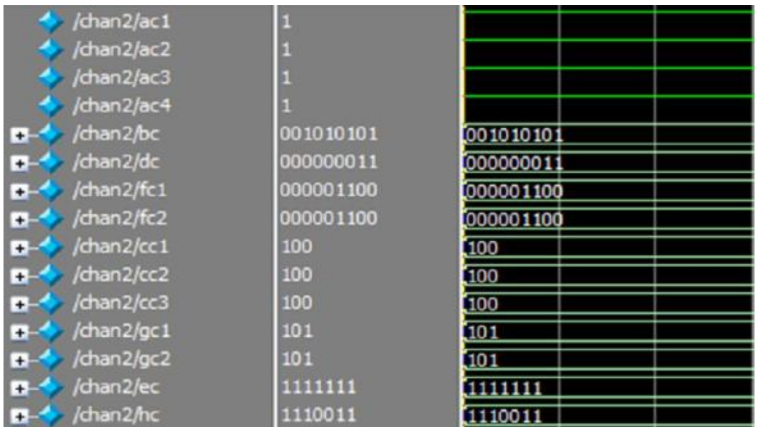


Figure 11 Decompression Engine

The Bitmask unit provides the XOR and shift operation based on the instructions from the large LUT. The input to the decompression engine is fetched from the cache memory where the compressed data are stored. With respect to the data's in the small and big LUT are decompressed with the help tag bits and index number. The decompression engine provides the lossless data recovery. It is specially designed hardware architecture placed in the FPGA architecture without increasing any area overhead. The data are decompressed with help of direct dictionary access or with bit-mask based approach. The data are matched with dictionary are processed directly. If it is matched using bitmask approach the data are shifted to its appropriate position using shift of mask operation and recovered using XOR operation. If the number of instructions are high then the decompression engine deployed into the fully distributed dictionary access. Where the number of LUT indexes are varied in accordance with the size of the processed instructions. The fully distributed dictionary access leads to faster decompression of matched instructions and it also increase the matching rate.

8. Results and Discussion

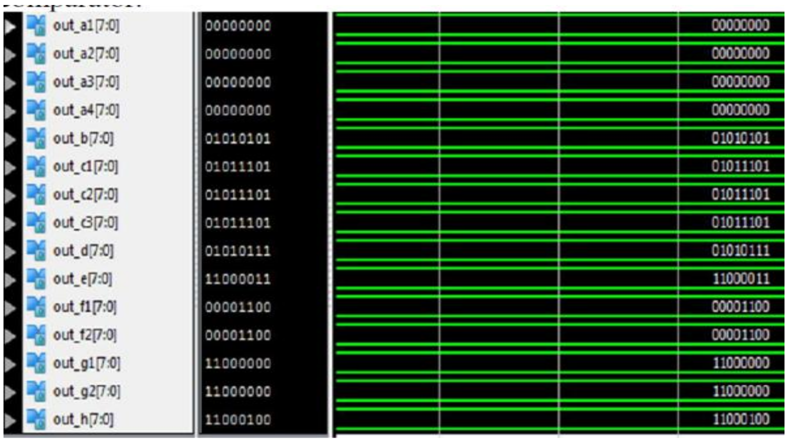
The following compression algorithm was performed using Quartus-II 10.0 tool and the decompression architecture was designed using Xilinx 14.1 tool. The following Figure 12 had shown the respective results for example taken in Figure 10 and its corresponding compression and decompression process. The input a, b, c, i, e, f, g and h are the different input data and results shows it corresponding compressed output. The Figure 12 shows the simulation results of compressed output using Quartus –II tool, which are compressed in a lossless manner. The Figure 13 shows the decompressed output using Xilinx 14.2 tool and it can retrieve the original instructions with the use of less hardware architecture. The overall memory consumption and timing is reduced which are shown in Figure 14 and 15.



```
/chan2/ac1    1            
/chan2/ac2    1            
/chan2/ac3    1            
/chan2/ac4    1            
/chan2/bc     001010101    001010101
/chan2/dc     000000011    000000011
/chan2/fc1    000001100    000001100
/chan2/fc2    000001100    000001100
/chan2/cc1    100          100
/chan2/cc2    100          100
/chan2/cc3    100          100
/chan2/gc1    101          101
/chan2/gc2    101          101
/chan2/ec     1111111      1111111
/chan2/hc     1110011      1110011
```

Figure 12 Compressed Output

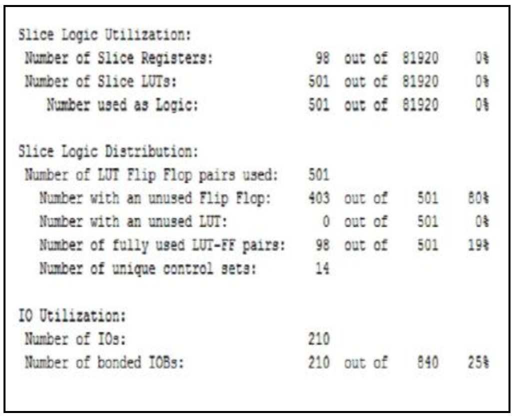
The bit 01 represented the data are compressed with small LUT access, 01 represented the data are compressed with large LUT access, 00 represented uncompressed data and 11 represented the data are compressed using bit mask technique. The decompression engine output for the following compressed output is shown following Figure 13 by comparing initial tag bit with the data stored in the dictionary by using comparator.



```
out_a1[7:0]   00000000    00000000
out_a2[7:0]   00000000    00000000
out_a3[7:0]   00000000    00000000
out_a4[7:0]   00000000    00000000
out_b[7:0]    01010101    01010101
out_c1[7:0]   01011101    01011101
out_c2[7:0]   01011101    01011101
out_c3[7:0]   01011101    01011101
out_d[7:0]    01010111    01010111
out_e[7:0]    11000011    11000011
out_f1[7:0]   00001100    00001100
out_f2[7:0]   00001100    00001100
out_g1[7:0]   11000000    11000000
out_g2[7:0]   11000000    11000000
out_h[7:0]    11000100    11000100
```

Figure 13 Decompression Output

The proposed compression and decompression technique provide the efficient LUT usage and area consumption when compared already existing technique as shown below

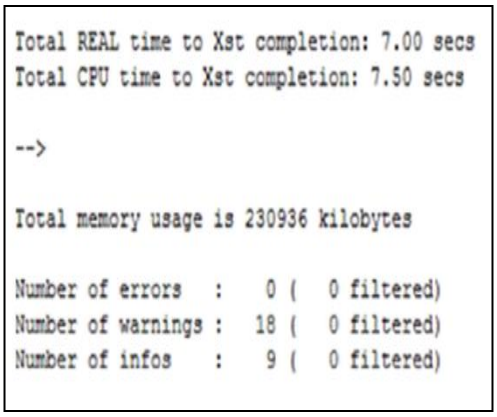


```
Slice Logic Utilization:
 Number of Slice Registers:            98  out of  81920    0%
 Number of Slice LUTs:                501  out of  81920    0%
    Number used as Logic:             501  out of  81920    0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:  501
   Number with an unused Flip Flop:   403  out of    501   80%
   Number with an unused LUT:           0  out of    501    0%
   Number of fully used LUT-FF pairs:  98  out of    501   19%
   Number of unique control sets:      14

IO Utilization:
 Number of IOs:                       210
 Number of bonded IOBs:               210  out of    840   25%
```

Figure 14 LUT usage in CLCBCC with MBSDS



```
Total REAL time to Xst completion: 7.00 secs
Total CPU time to Xst completion: 7.50 secs

-->

Total memory usage is 230936 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :   18 (   0 filtered)
Number of infos    :    9 (   0 filtered)
```

Figure15 Memory usage using CLCBCC with MBSDS

Table 1 Comparison for Different Decompression Technique

Compression Technique for (N=25)	
Compression Technique	Saving ratio = 100%-CR
DCC	18%
CLCBCC WITH MBSDS	21%
Improved CLCBCC with MBSDS	27.5%

The above Table 1 shows the saving ratio for the different technique with proposed technique of improved CLCBCC with MBSDS compression technique. It leads overall saving ratio of approximately 27.5% for number of inputs is 25. If the number of input is increased with the help of distributed dictionary access we can attain high compression ratio and also saving ratio. The bit flip is the biggest drawback with using the proposed technique but it can be overcome by extra one identification bit.

9. Conclusion

The objective of codeword length constrained bitmask code compression with mixed bit saving algorithm is proposed in this research. The fully separated dictionary architecture is designed to provide efficient code compression and also to improve the CR. The distributed dictionary increases the performance of the decompression engine with the less hardware overhead architecture. The decompression engine shows the original instructions that are retrieved from the compressed instructions without any loss. In future the same technique is applicable for multi-core architecture which requires higher communication bandwidth either between the processors and cache or between the cache and memory. In the multi-core system, design of decompression engine is to be investigated with the fully distributed lookup table access.

In future studies, not only the compression ratio but also performance and power consumption should be analyzed for single and multi-core architectures. The error correction and detection technique can also be considered which can yield high accuracy for code compression technique in large benchmarks.

10. References

[1]. X. Qin and P. Mishra, “A universal placement technique of compressed instructions for efficient parallel decompression,” IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, 28(8), 2009, 1224-1236

[2]. D. Sobya, Muruganandham, S. Nallusamy and Chakraborty, “Wireless ECG monitoring system using IoT based signal conditioning module for real time signal acquisition”, Indian Journal of Public Health Research and Development, 9(2), 2018, 296-301

[3]. S.K. Muruganandham, D. Sobya, S. Nallusamy, Dulal Krishna Mandal and P.S. Chakraborty, “Study on leaf segmentation using k-means and k-medoid clustering algorithm for identification of disease”, Indian Journal of Public Health Research and Development, 9(2), 2018, 291-295

[4]. D. Sobya, SK. Muruganantham, S. Nallusamy and P.S. Chakraborty, “A proposed model for public distribution system through IOT”, International Journal on Recent Researches in Science Engineering and Technology, 6(2), 2018, 67-74

[5]. J. Nikara, S. Vassiliadis, J. Takala and P. Liuha, “Multiple symbol parallel decoding for variable length codes,” IEEE Transactions on Very Large Scale Integration System, 12(7), 2004, 676-685

[6]. D. Sobya, “Discrete wavelet transform for image compression and reconstruction via VLSI”, International Research Journal in Advanced Engineering and Technology, 1(1), 2015, 31-35

[7]. SK. Muruganantham, D. Sobya, S. Nallusamy, Dulal Krishna Mandal and P.S. Chakraborty, “Development of policy based security application to enhance the security of software defined network”, International Journal on Recent Researches in Science Engineering and Technology, 6(2), 2018, 58-66

[8]. D. Sobya, SK. Muruganantham, S. Nallusamy and P.S. Chakraborty, "A proposed model for smart farming in rural areas using IoT advanced technologies", International Journal on Recent Researches in Science Engineering and Technology, 6(1), 2018, 61-67

[9]. M. Huebner, M. Ullmann, F. Weissel and J. Becker, "Real time configuration code decompression for dynamic FPGA self-reconfiguration," in Proceedings International Parallel Distributions Process Symposium, 2004, 138-143

[10]. D. Sobya, SK. Muruganantham, S. Nallusamy, Dulal Krishna Mandal and P.S. Chakraborty, "Study on mobile adhoc networks routing protocols to enhance the end-user experience", International Journal on Recent Researches in Science Engineering and Technology, 6(1), 2018, 54-60

[11]. S. Nallusamy and Gautam Majumdar, "Enhancement of overall equipment effectiveness using total productive maintenance in a manufacturing industry", International Journal of Performability Engineering, 13(2), 2017, 01-16

[12]. D. Sobya, S.K. Muruganandham, S. Nallusamy and Partha Sarathi Chakraborty, "Optimization of bit error and data transmission rate for different modulation scheme using MIMO diversity technique", International Journal on Recent Researches in Science Engineering and Technology, 5(8), 2017, 06-13

[13]. Snehal S. Khartad and Pravin Matte, "Reduction of Bit stream Transfer Time in FPGA", IOSR Journal of Electronics and Communication Engineering, 9(2), 2014, 82-86

[14]. D. Sobya, S. Nallusamy, P.S. Chakraborty and S.K. Muruganandham, "Design of multi functional electronic energy meter enabled with zigbee protocol intended for industrial applications", International Journal of Current Advanced Research, 6(10), 2017, 7029-7033

[15]. S. Nallusamy, R. Balaji and S. Sundar, "Proposed model for inventory review policy through ABC analysis in an automotive manufacturing industry", International Journal of Engineering Research in Africa, 29, 2017, 165-174

[16]. Dandalis and V.K. Prasanna, "Configuration compression for FPGA based embedded systems," IEEE Transactions on Very Large Scale Integration and Systems, 13(12), 2005, 1394-1398

[17]. D. Sobya, S K. Muruganandham, S. Nallusamy and Partha Sarathi Chakraborty, "Development of bluetooth based smart meter reading system for residential power monitoring," International Journal on Recent Researches in Science Engineering and Technology, 5(12), 2017, 39-47

[18]. S K. Muruganandham, D. Sobya, S. Nallusamy, P.S. Chakraborty and D K Mandal, "Development of framework to enhance the lifetime of wireless network in mobile power sharing networks," International Journal on Recent Researches in Science Engineering and Technology, 5(12), 2017, 28-38

[19]. X. Qin, Muthry and P. Mishra, "Decoding aware compression of FPGA bit streams", IEEE Transactions on Very Large Scale Integration and Systems, 19, 2011, 411-419

[20]. S. Nallusamy, G.B. Dinagaraj, K. Balakannan and S. Satheesh, "Sustainable green lean manufacturing practices in small scale industries-A case study", International Journal of Applied Engineering Research, 10(62), 2015, 143-146

[21]. D. Sobya, S. Nallusamy and Partha Sarathi Chakraborty, "A proposed remote monitoring system by global system for mobile communication and internet technology," International Journal on Recent Researches in Science Engineering and Technology, 5(11), 2017, 07-14

[22]. S. Seong and P. Mishra, "Bitmask based code compression for embedded systems", IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, 27, 2008, 673-679

[23]. D. Sobya, Partha Sarathi Chakraborty and Dulal Krishna Mandal, "Design and development of IoT based residential automation security system with bluetooth technology", International Journal of Application or Innovation in Engineering & Management, 6(6), 2017, 62-72

[24]. Jian Yan et al., "Lossless compression decoders for bit streams and software binaries based on high level synthesis", IEEE Transactions on Very Large Scale Integration Systems, 25(10), 2017, 2842-2855

[25]. D. Sobya, Arvind Kumar and Vicky Kumar, “Smart IoT based energy monitoring and controlling household appliances”, International Innovative Research Journal of Engineering and Technology, 2, 2017, 94-97

[26]. Hemnath and Prabhu, “Compression of FPGA bit streams using improved RLE algorithm”, International Journal of Scientific and Engineering Research, 4(6), 2013, 1162-1169

[27]. D. Sobya, R. Varshni and P. Albinia, “MEMS based hand gesture wheel chair movement control with emergency alert”, International Innovative Research Journal of Engineering and Technology, 2, 2017, 90-93

[28]. S. Chattopadhyay and A.K. Turuk, “A Scheme for key revocation in wireless sensor networks”, International Journal of Advanced Computer Engineering and Communication Technology, 1, 2012, 16-20

[29]. D. Sobya, “Data compression analysis of rocket engines with vector quantization based on FCM algorithm”, Int. Journal of Engineering Research in Africa, 22, 2016, 135-140

[30]. L. Li, K. Chakrabarty and N.A. Touba, “Test data compression using dictionaries with selective entries and fixed-length indices,” ACM Transactions on Design Automation of Electronic Systems, 8(4), 2003, 470-490

[31]. D. Sobya, “Lab view based multi-input fuzzy logic controller of DC motor speed control”, International Journal of Research in Mechanical, Mechatronics and Automobile Engineering, 1(1), 2015, 55-60

[32]. M. Saravanakumar, D. Sobya and B. Sathis kumar “Design and development of new technique for testing of field programmable gate arrays”, International Journal of Research in Mechanical, Mechatronics and Automobile Engineering, 1(4), 2016, 139-147

[33]. K. Balakannan, S. Nallusamy, P.S. Chakraborty and Gautam Majumdar, “Selection and evaluation of supplier by decision model of hybrid data envelopment analysis”, International Journal of Applied Engineering Research, 10(62), 2015, 123-127

[34]. S. Nallusamy, Sri Lakshmana Kumar, K.Balakannan and P.S.Chakraborty, “MCDM tools application for selection of suppliers in manufacturing industries using AHP, Fuzzy Logic and ANN”, International Journal of Engineering Research in Africa, 19, 2015, 130-137

[35]. D. Sobya, “Design and implementation of FPGA based wave-pipelining for digital signal processing circuits”, International Research Journal in Advanced Engineering and Technology, 1(2), 2015, 36-42

[36]. S. Seong and P. Mishra, “Bitmask based code compression for embedded systems”, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, 27, 2008, 673-685