ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 5, May 2015

Data Compression Using Bitmask and Dictionary Selection Methods

G.Meena Kumari, T.Sivarama Krishnan

Assistant Professor, Department of ECE Bharath University, Chennai, Tamil Nadu, India

Department of ECE, K.C.G. College of Technology, Chennai, Tamil Nadu, India

ABSTRACT: In system on chip (SOC) designs, higher circuit densities have larger volume of test data. So in this larger test data applying in a particular circuit, the occupied memory volume and testing time consumption is very large. Test data compression method is reducing the testing time and memory requirements. After test data compression, the data will be stored to memory. When we want to test a circuit, that time, compressed data is immediately decompressed and given to Design Under Test (DUT).This paper deals with the comparison of the Bitmask and Dictionary selection method test data compression, Run length compression(RL) and Golomb method test data compression and then best compression method output is stored to memory. The bitmask and dictionary selection method is a very efficient compression method and also fast compression, decompression mechanism. In this compression technique, input test data is compared with created dictionary. When the input data is matched with dictionary, no need for bitmask operation, otherwise, it needs. In this method of compression, data is stored to memory and then when we want to test a particular circuit that time immediately compressed data is decompressed and given to circuit (DUT).

KEYWORDS: Bitmasks, decompression, test compression.

I. INTRODUCTION

IN SYSTEM-ON-CHIP (SOC) designs, higher circuit densities have led to larger volume of test data, which demands larger memory requirement in addition to an increased testing time. Test data compression plays a crucial role, reducing the testing time and memory requirements. It also overcomes the automatic test equipment (ATE) bandwidth limitation. Alternatives to external testing include built-in self-test (BIST). However, BIST is not appropriate for logic testing because of its random-resistant fault and bus contention during test application [1], which leads to inadequate test coverage. Other alternatives like bit-flipping [2] and bit fixing [3] provide greater fault coverage, with the disadvantage that structural information has to be provided. Reduction of test data using structural methods like Illinois Scan Architecture (ILS) [4] demands modification of the design. Test data compression algorithms can reduce the test data to a larger degree without facing any of the aforementioned disadvantages.

The overview of a traditional test compression framework is shown in Fig. 1. The original test data is compressed and stored in the memory. Thus, the memory size is significantly reduced. 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).

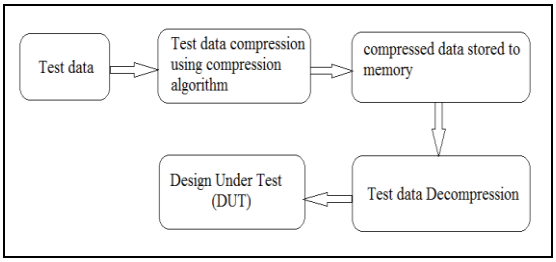

Fig. 1. Test data compression methodology.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 5, May 2015

Dictionary-based test data compression is a promising approach which has been used by Li *et al.* [1]. Dictionary-based compression techniques are also popular in embedded systems domain since they provide a dual advantage of good compression efficiency as well as fast decompression mechanism. Many recently proposed techniques [5] have tried to improve the dictionary-based compression techniques by considering mismatches. However, the efficiency of these techniques depends on the number of bits allowed to mismatch. It is obvious that if more number of bit changes is allowed, more matching patterns will be generated. However, remembering more bit positions may lead to unprofitable compression. Bitmask based compression [6] addresses this issue by creating more matching patterns with the aid of bitmasks, while taking care of the size of the compressed code. This paper tries to combine the advantages of dictionary based test compression [1] as well asbitmask-based code compression [6].Bitmask based compression of test data may seem attractive, but it presents various challenges. The primary concern is the presence of don't cares ("X") in the test data set. Since bitmask based compression technique [6] was not designed for data with don't care values, direct application of this technique on test data does not result in a good compression efficiency. We have to determine not only the effective bitmasks, but also a profitable dictionary that produces optimal results. We demonstrate that selection of bitmasks and dictionary using existing techniques [1], [6] are not appropriate in case of test data compression using bitmasks. Our approach solves these problems by selecting profitable bitmasks as well as proposing 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 [1].

II. RELATED WORK

Dictionary-based compression techniques have been recently used to reduce the test data volume in SOCs. Li *et al.* [1] and Reddy *et al.* [13] used fixed length dictionary entries to reduce test data volume. Dynamic dictionaries along with LZ77 technique has been used byWolff *et al.* [14].Wurtenberger *et al.* [15] 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 *etal.* [1], 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.Wurtenberger *et al.* [15] have proposed a test data compression technique which is somewhat similar to our approach to remember the mismatches from the dictionary entries.However, there are significant differences between the two approaches. While they [15] try to remember the positions of the bit changes, our method uses bitmasks to account for the mismatch. Due to considering only one bit-fix, [15] looses the opportunity of taking care of multiple mismatches.Our approach performs 10%–30% better compression while uses simpler decompression engine compared to them.

III. BACKGROUND AND MOTIVATION

This section briefly describes bitmask-based code compression[6], and highlight the challenges in employing bitmaskbased technique for test data compression.

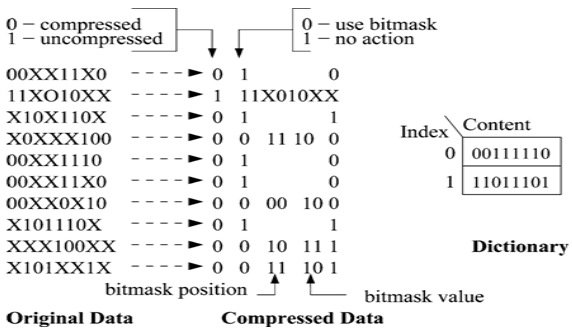

Fig. 2. Bitmask-based test data compression.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 5, May 2015

A. Bitmask-Based 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. 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. In this example, the compression efficiency is 27.5%, based on the following formula expressed as percentage:

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

Since existing approach does not handle don't cares ("X"), in this example we have replaced all don't cares by 1. Note that we could have replaced all don't cares with 0's as well. In that case, it will result in worse compression efficiency of 2.5%. A better compression efficiency can be achieved by selectively replacing the don't cares with "0" or "1" instead of replacing all by 0's (or 1's). It is a major challenge to identify the selective replacement to generate the best possible compression efficiency.

B. Challenges in Bitmask Based Compression

This section outlines various challenges in using bitmask based approach in test data compression. There are three major challenges in bitmask-based test data compression. The problem is further aggravated since each of these challenges are interdependent and cannot be solved independently.

1) Dictionary Selection: A profitable dictionary is to be selected which takes into account the bit savings due to frequency matching as well as bitmasks.

2) Bitmask Selection: Appropriate number and type of bitmasks are to be selected for compression.

3) Don't Care Resolution: It is necessary to selectively replaceeach don't care with "0" or "1".

Another challenge for test data compression is that selective don't care resolution is difficult and it further complicates the dictionary and bitmask selection. Consider the two vectors:

"00XX1X10" and "X0X110X0". Although these two vectors have lot of dissimilarities, it is obvious that they will match. This is because a don't care can be matched with a "0" or a "1" according to the matching pattern. Thus a lot more matching can be obtained in case of vectors with don't cares. Therefore, the dictionary selection and bitmask selection algorithms have to be modified in order to incorporate this factor.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 5, May 2015

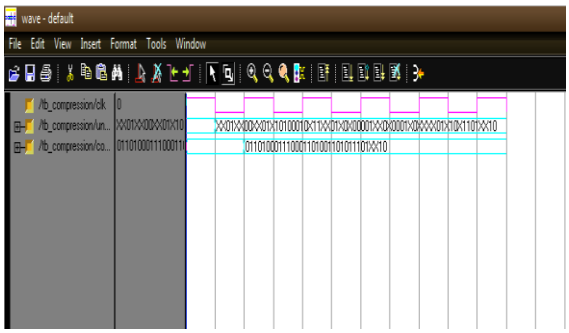

Fig.3. Dictionary Selection and Bitmask Method Test Data Compressed output.

IV. TEST DATA COMPRESSION USING BITMASKS.

We have developed an efficient test data compression algorithm using bitmasks. Fig. 3 presents our bitmask-based test compression methodology. It has four major steps: 1) divide the input test data based on number of scan chains; 2) dictionary selection; 3) bitmask selection; and 4) test compression using don't care resolution, dictionary and bitmask selection.

A. *Bitmask Selection*

The generic encoding formats of bitmask-based compression technique for various number of bitmasks. [18]A compressed
data stores information regarding the bitmask type, bitmask location, and the mask pattern itself. The bitmask can be applied on different places on a vector and the number of bits required for indicating the position varies depending on the bitmask type. For instance, if we consider a 32-bit vector, an 8-bit mask applied on only byte boundaries requires 2-bits, since it can be applied on four locations. If we do not restrict the placement of the bitmask, it will require 5 bits to indicate any starting position on a 32-bit vector.[19]

Bitmasks may be sliding or fixed. Fixed bitmasks are referred with the letter *F* while sliding bitmasks are referred with the letter S . For example *2S*, refers to a sliding bitmask of length 2 while *2F* refers to a fixed bitmask of length 2. A fixed bitmask is one which can be applied to fixed locations, such as byte boundaries. However, sliding bitmasks can be applied anywhere in the test vector. Since the fixed bitmasks can be applied only to fixed locations, the number of positions where they can be applied is significantly less compared to sliding bitmasks. Hence, the number of bits needed to represent them are less than sliding
bitmasks.Seong *et al.* [6] has shown that the profitable bitmasks to be selected for code compression are 2s, 2f, 4s, and 4f. However, in case of test data compression, the last two bitmasks are not profitable. This is because the probability that four corresponding contiguous bits will differ in a pair of test data is only 0.2% [16], which can easily be neglected. Therefore, it is not profitable to use bitmasks of length 4 or higher. Thus, we perform our compression by using 1-bit, 2s or 2f bitmasks. The number of bitmasks selected, which depends on both the test vector length and the dictionary entries, can be determined using the following lemma.
Lemma: The number of bitmasks is dependent on vector length and dictionary entries.[20]

B.Dictionary Selection

The dictionary selection algorithm is a critical part in bitmask-based test data compression. Similar to Li et al. [1], we used the classical clique partitioning algorithm of graph
theory [17]. The clique-partitioning problem basically refers to the breaking up of a graph into several cliques, such that the nodes within one clique are all interconnected. This problem is NP-hard [18], so heuristic approaches are used to solve it in real time.
A graph G is drawn with *n*l* nodes, where each node signifies *m* a -bit test vector. An edge is drawn between two nodes when they are compatible. Two nodes are said to be compatible if they meet any one of the following two

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 5, May 2015

requirements: 1) for all positions, the corresponding characters in the two vectors are either equal or one of them is a don't care; or 2) two vectors can be matched by predetermined profitable bitmasks. Each edge also contains weight information. The weight is determined based on the number of bits that can be saved by using that edge (direct or bitmask-based matching). Based on this graph model, we developed three dictionary selection
techniques: 1) two-step method (TSM); 2) method using compatible edges (MCE) without edge weights; 3) MCE with edge weights (MEW). Each of these techniques uses a variant
of well-known clique partitioning algorithm. The remainder of this section describes these three techniques in detail.

1) Two-Step Method: In TSM, we consider only edges that are formed by direct matching. In other words, the graph will not have any edges corresponding to bitmask-based matching.Then a clique partitioning algorithm [1] is performed on the graph. This is a heuristics-based procedure that selects the node with the largest connectivity and is entered as the first entry to a clique. Now, the nodes connected with it are analyzed, and the
node having the largest connectivity among these (and not in the entire graph) is selected. This process is repeated until no node remains to be selected. The entries of the clique are deleted from the graph. The algorithm is repeated until the graph becomes empty. The clique partitioning algorithm is used in MCE and MEW as well. Since we have a predefined number of dictionary entries, two possibilities may arise. The number of cliques selected may be greater than the predefined number of entries or vice versa.[21]

2) Method Using Compatible Edges (MCE) Without Edge Weights: In MCE, weight of all the edges (direct or bitmask- based match) are considered equal. A clique selection algorithm is then performed in the same way.

3) MCE With Edge Weights (MEW): MEW is same as MCE except that we consider edge weights. As indicated earlier, the edge weight is determined based on the number of bits saved if
that edge is used for direct or bitmask-based matching. During clique partitioning, instead of connectivity, the total savings by each node is taken into account. The total savings by each node is obtained as the sum of weights of edges originating from that node.

V. DECOMPRESSION MECHANISM

We have proposed the design of a decompression engine (DCE), shown in Fig. 4, that can easily handle bitmasks and provide fast decompression. [22]The design of our engine is based on the one cycle decompression engine proposed by Seong *et al.* [6]. The most important feature is the introduction of XOR gate in addition to the decompression scheme for dictionary based compression. The decompression engine generates a test data length bitmask, which is then XORed with the dictionary entry. The test data length bitmask is created by applying the bitmask on the specified position in the encoding. The generation of bitmask is done in parallel with dictionary access, thus reducing additional penalty. The DCE can decode more than one compressed data in one cycle.

The decompression engine takes the compressed vector as input. It checks the first bit to see whether the data is compressed. If the first bit is "1" (implies uncompressed), it directly
sends the uncompressed data to the output buffer. On the other hand, if the first bit is a "0", it implies this is a compressed data. Now, there are two possibilities in this scenario. The data may be compressed directly using dictionary entry or may have used bitmasks. The decompression engine will operate differently in
these two cases.[23]

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 5, May 2015

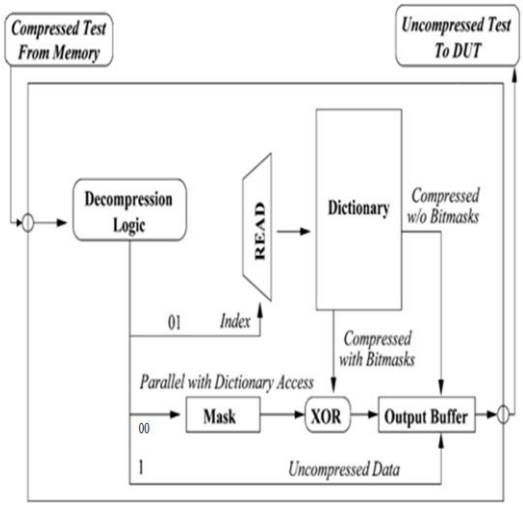

Fig. 4. Decompression engine for bitmask-based encoding.

In Decompression logic, the compressed data is checked whether the first two bit is '01' or '00'or '1' from memory.Here if it is first two bit is '01'- Data is compressed without bitmask operation. So read the index value. Directly taking the dictionary index corresponding 8 bit data is given to output buffer.Otherwise first two bit is '00' - Data is compressed with bitmask operation. So it refer next two bits (i.e. bitmask position) and go to next mentioned location. It EX-OR s the bitmask value with given corresponding dictionary indexed value to get the original uncompressed data. It is given to output buffer. Otherwise when the first bit is '1' - data is Uncompressed. so, directly it is given to output buffer. Finally we get the decompression methodology. Output is exactly equal to the original uncompressed data.

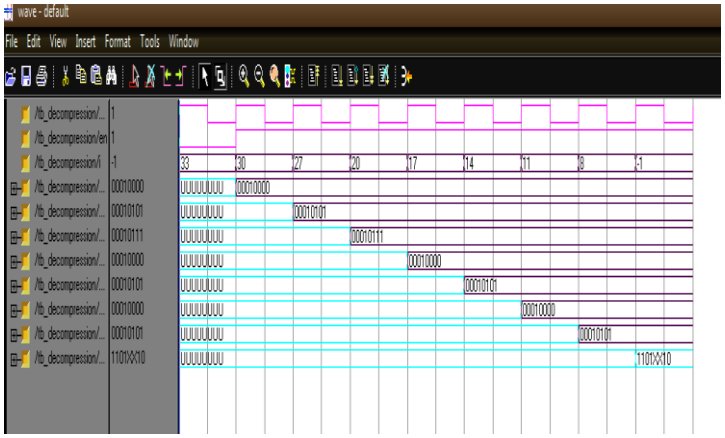

Fig.5.Decompression simulated output.

ACKNOWLEDGMENT

Our thanks to the experts of KCG College Of Technology- Chennai. Dr. RANGANATHAN, Dr. Ramarathinam and Mr. Thomas who have contributed towards development of my project work.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 5, May 2015

VI. CONCLUSION

A test data compression algorithm that combines the advantages of dictionary based compression and bitmask based compression. This paper developed efficient bitmask and dictionary selection techniques for test data compression in order to create maximum matching patterns in the presence of don't cares. Our test compression technique used the dictionary and bitmask selection methods to significantly reduce the testing time and memory requirements. The test data compressed using dictionary selection and Bitmask method is reduced the maximum bit volume that same time without affecting the test data overall performance and stored memory content decompressed at anytime immediately. The test data compression and decompression done is very fast. So this compression and decompression method reduces the higher order SOC circuit testing time consumption and occupied memory volume.

REFERENCES

[1] L. Li, K. Chakrabarty, and N. Touba, "Test data compression using dictionaries with selective entries and fixed-length indices," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 8, no. 4, pp. 470–490, 2003.
[2] Prabhakar J., Senthilkumar M., Priya M.S., Mahalakshmi K., Sehgal P.K., Sukumaran V.G., "Evaluation of Antimicrobial Efficacy of Herbal Alternatives (Triphala and Green Tea Polyphenols), MTAD, and 5% Sodium Hypochlorite against Enterococcus faecalis Biofilm Formed on Tooth Substrate: An In Vitro Study", Journal of Endodontics, ISSN : 0099-2399, 36(1) (2010) pp. 83-86
[3] H. Wunderlich and G. Kiefer, "Bit-flipping BIST," in *Proc. Int. Conf. Comput.-Aided Des.*, 1996, pp. 337–343.
[4] Ramkumar Prabhu, M., Reji, V., Sivabalan, A., "Improved radiation and bandwidth of triangular and star patch antenna", Research Journal of Applied Sciences, Engineering and Technology, ISSN : 2040-7459, 4(12) (2012) pp. 1740-1748
[5] N. Touba and E. McCluskey, "Altering a pseudo-random bit sequence for scan based bist," in *Proc. Int. Test Conf.*, 1996, pp. 167–175.
[6] Saravanan T., Saritha G., Udayakumar R., "A Robust H-infinity two degree of freedom control for electro magnetic suspension system", Middle - East Journal of Scientific Research, ISSN : 1990-9233, 18(12) (2013) pp. 1827-1831.
[7] 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.
[8] Saravanan, T., Saritha, G., "Buck converter with a variable number of predictive current distributing method", Indian Journal of Science and Technology, ISSN : 0974-6846, 6(S5) (2013) pp.4583-4588.
[9] 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.
[10] Saravanan, T., Srinivasan, V., Sandiya, V.P., "A two stage DC-DC converter with isolation for renewable energy applications", Indian Journal of Science and Technology, ISSN : 0974-6846, 6(S6) (2013) pp. 4824-4830.
[11] 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.
[12] M.-E. N. A. Jas, J. Ghosh-Dastidar, and N. Touba, "An efficient test vector compression scheme using selective Huffman coding," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 22, no. 6, pp.
797–806, Jun. 2003.
[13] 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.
[14] 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.
[15] 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.
[16] 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.
[17] L. Lingappan, S. Ravi, A. Raghunathan, N. K. Jha, and S. T.Chakradhar, "Test-volume reduction in systems-on-a-chip using heterogeneous and multilevel compression techniques," *IEEE Trans.Comput.-Aided Des. Integr. Circuits Syst.*, vol. 25, no. 10, pp.2193–2206, Oct. 2006.
[18] S. Reddy, K. Miyase, S. Kajihara, and I. Pomeranz, "On test datavolume reduction for multiple scan chain design," in *Proc. VLSI TestSymp.*, 2002, pp. 103–108.
[19] F. Wolff and C. Papachristou, "Multiscan-based test compression and
hardware decompression using LZ77," in Proc. Int. Test Conf., 2002,pp. 331–339.
[20] A. Wurtenberger, C. Tautermann, and S. Hellebrand, "Data compression for multiple scan chains using dictionaries with corrections," in
Proc. Int. Test Conf., 2004, pp. 926–935
[21] K. Basu and P. Mishra, "A novel test-data compression technique using application-aware bitmask and dictionary selection methods," in Proc.ACM Great Lakes Symp. VLSI, 2008, pp. 83–88
[22] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, Introduction to Algorithms. Boston, MA: MIT Press, 2001.
[23] M. Garey and D. Johnson, Computers and Intractability: A Guide tothe Theory of NP-Completeness. New York: Freeman, 1979.
[24] TVUK Kumar, B Karthik, EB Kumaran, Visual Secret Sharing Scheme for JPEG Compressed Images, Middle-East Journal of Scientific Research 12 (12), PP 1873-1880, 2012.
[25] B Karthik, TVUK Kumar, EMI Developed Test Methodologies for Short Duration Noises, Indian Journal of Science and Technology 6 (5S), PP 4615-4619, 2013.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 5, May 2015

[26] Mr.B.Karthik, Dr.T.V.U. Kiran Kumar, N. Gomathi, Immobilizer Implementation using Mutual Authentication with enhanced security using 802.15.4 protocol,International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering ,ISSN 2278 - 8875, pp 142-151, Vol. 1, Issue 3, September 2012.
[26] M.Jasmin,Bus Matrix Synthesis Based On Steiner Graphs For Power Efficient System On Chip Communications ,International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, ISSN 2278 - 8875 , pp 98-104,Vol. 1, Issue 2, August 2012.
[27] M. Jasmin, A Low Power Pipelined FFT/IFFT Processor for OFDM Applications,International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering,ISSN (Print) : 2320 – 3765, pp 4712-4721, Vol. 2, Issue 10, October 2013.
[28] M Jasmin, Low Power Design of the Processer Based on Architecture Modifications,International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering,ISSN: 2278 – 8875, pp 2922-2927, Vol. 2, Issue 7, July 2013.
[29] K.Subbulakshmi,Implementation for SMS4-GCM and High-Speed Architecture Design,International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, ISSN (Print) : 2320 – 3765, pp 10965-10970, Vol. 3, Issue 8, August 2014.