International Journal for Modern Trends in Science and Technology
Volume: 02, Issue No: 10, October 2016
http://www.ijmtst.com
ISSN: 2455-3778

Improvement in Error Resilience in BIST using hamming code

Radharapu Ravivarma[1] | M. Sanjay [2]

[1]PG Scholar, Vaagdevi College of Engineering, Telangana.
[2]Assistant Professor, Vaagdevi College of Engineering, Telangana.

To Cite this Article
Radharapu Ravivarma and M.Sanjay, Improvement in Error Resilience in BIST using hamming code , International Journal for Modern Trends in Science and Technology, Vol. 02, Issue 10, 2016, pp. 82-87.

ABSTRACT

In the current scenario of IP core based SoC, to test the CUT we need to communication link between Circuit Under Test and ATPG, so before applying to actual DUT. If there is a problem with this link, there may be a lip in bit of test data. Compared to original test data, if there is a bit lip in the original data, the codeword may change and hence the decompressed data will have a large number of bit deviation. This deviation in bits can severely degrade the test quality and overall fault coverage which may affect yield. The error resilience is the capability of the test data to resist against such bit lips. Here in this paper, the earlier methods of error resilience is compared and a Hamming code based error resilience technique is proposed to improve the error resilience capacity of compressed test data. This method is applied on Huffman code based compressed test data of widely used ISCAS benchmark circuits. The fault coverage measurement results show the effectiveness of the proposed method. The basic goal here is to survey the effect of bit lips on fault coverage and prepare a platform for further development in this avenue.

KEYWORDS: *Automatic Test Equipment (ATE), bit-lip; Compression; Fault Tolerance; Hamming code; Fault Coverage; Area Overhead;, Bits Overhead*

I. INTRODUCTION

Considering the today's scenario of fast time-to-market and increasing test cost compared to manufacturing cost, the use of Intellectual Property (IP) course for gathering complex functionality has been common. [1]. IP core based system on chip(SoC) is becoming new paradigm in the electronics based industry for its reusability and ability to execute rich set of functionality in a very short time. Multiple core on a SoC adds complexity to the design, so it's become challenging to test each core, as all core have different 1I0S, different test pattens and different scan chain lengths, etc. In general, SoC test challenge includes the dificulty of distributing design inside the SoC and its test development, the difficulty of accessing the test in embedded core and finally optimization of the SoC level test.

A test set for a SoC test includes test vectors for each core. The total volume of these data can be very high, can be in the scale of a gigabits which need high storage facility, but on another way, it increases the entire cost of the test. To highlight these problems, test data compression techniques are introduced. [2] [3].

Recently, many test data compression techniques are developed to reduce test data volume. For example, statistical test data compression techniques like run length code, frequency directed run length code [4][5][6], Golomb code[5][7] Extended FDR [5], Modified

Extended EFDR [5], VIHC coding[8] etc. The comparison of these techniques is presented in [9 HDR]. Among such various test data compression method like broadcast scan based methods, linear decompression based methods and the code based methods, the code based compression method inds the most suitable for IP cores where the internal architecture of core is hidden rom system Integrator. [9].

When the compressed data is being transferred rom ATE to SoC, if there is any distortion or noise in the communication link, it may happen that the test data bit may lip. This bit lip even though may not be so serious for uncompressed data, but gives serious results in case of compressed data because it may change one or many codewords in compressed data and correspondingly regenerated-decompressed data will be different from original data. When such deviated data will be applied to DUT for testing, definitely will not give the test quality which was intended. The fault coverage may be affected and hence the yield will be also affected.

The capability of test data to resist the effects of such bit lips is called error resilience. Fault endurance, reliability and security are the major concern for resilience of test data. So the error resilience can be defined as the ability of a design under test (DUT) to execute proper unction in the presence of bit lip error, bit deletion, burst error and missing fragmentations and to recover from any service degradation. These stuffs are essential in a testing where the validity of data depends on the reliable, secure and correct function of the device under test. Considered systems include, but are not limited to, infrastructures, computer networks, including adhoc and mesh networks, web-based systems, or service- oriented architectures, embedded systems, manufacturing systems, control systems and more. Much work has been done already on evaluation, analysis and enhancement of error resilience of test data, but a lot remains to be done. The combination of performance and dependability, performability, has been widely studied. But aspect, like, improvement in error resilience is still ongoing work. As the systems are changing day by day as per its characteristics, it's a high need to expand work in this area device under test. Considered systems include, but are not limited to, inrastructures, computer networks, including adhoc and mesh networks, web-based systems, or service- oriented architectures, embedded systems, manufacturing systems, control systems and more. Much work has been done already on evaluation, analysis and enhancement of error resilience of test data, but a lot remains to be done. The combination of performance and dependability, performability, has been widely studied. But aspect, like, improvement in error resilience is still ongoing work. As the systems are changing day by day as per its characteristics, it's a high need to expand work in this area The paper has addressed methods and tools to describe, measure, evaluate benchmark and improve resilience concening resilience of test data. Here in this paper, we have calculated the effect of bit lips on fault coverage of widely cited ISCAS benchmark circuits for both the case : I. when normal (uncompressed) data is used through serial link and 2. When the compressed data is used and the comparisons are analyzed. Further we have proposed the Hamming code base approach to improve the error resilience and its effectivity is analyzed by calculating the bit overhead and improvement in fault coverage. The basic goal here is to prepare a platform for further development to improve error resilience capability in case of compressed test data being used for IP core based SoC

II. THE SIGNIFICANCE OF BIT FLIP N TEST DATA

A. Why Bit Flp Occurs?

Bit lips can occur in many ways during the transmission of test data from ATE to OUT. Noise can affect the interfacing line between the ATE and design under test, which can raise the errors because of bit lip. Moreover, Bit lips can also be generated when test inputs are generated as a pin waveform. So it can be said that bit lip can be reduced if the ATE test program and the DUT is fully debugged. However, as testing is done at very high speed, eror in few bits(bit lips) must be considered as unavoidable in ATE environments [10].

B. fect of Bit Flip on Fault Coverage

During the development of the test program, Bit-lips can produce diverse effects over its entire testing environment. Throughout the manufacturing test, a bit-lip affects the test data in such a way that the coverage of inaccurate test set can be reduced (as affected by bit-lips). As a result, there is a hike in cost incorporated with debugging and development, while decreases productivity.

C. Experimental Set-up to measure the efect ofBit-lip on Fault Coverage

Here the full scan version of the ISCAS'89 circuits is used as a platform to evaluate different bit lip effects and associated fundamentals. The experiments were performed on test data generated by MINTEST tools. In the given test cubes, all unspecified bits are filled with MT ill technique and then re-ordered using the Artificial Intelligence (AI) based algorithm[II][12]. Generally the occurrence of bit lips is considered in two ways i.) A bit lip probability ii.) Bit lip count. In this paper, we are focusing mainly on bit lip count. The bit lip locations are uniformly spread over the entire test vector. So the bit lips are uniformly distributed to the entire bit stream and can be evaluated in all conditions. Here, for simplicity bit lip count of 1,2 5 and 10 are used for experimental framework. Here each bit lip count is repeated for 25 times and average fault coverage is calculated. The Huffman compression technique is used for data compression with symbol size of 4 bits [13].

D. fect of Bit Flip on Uncompressed Data

As we know, bits in compressed test data carries more information as compared to original bit stream, So bit lip is important for test data compression circuits. To understand the minimum effect of bit lip, it is understand that fault coverage loss with original test data due to bit lip represents lower bound on the eror resilience. For the proper understanding, bit lips are uniformly injected in the test data and corresponding fault coverage is measured. Tabla I shows the fault coverage due to bit lip in uncompressed test data. The average coverage loss is 0.1 to 0.32% for 1 bit, 2 bit,S bit and 10 bit lips.

S820	99.14	98.92	98.92	98.92
S832	97.17	97.17	97.17	97.17
S838	99.67	99.67	99.67	99.67

III. EFFECT OF BIT FLIP ON COMPRESSED TEST DATA

Assume that there are V test vectors which are the source of combinational and sequential test data. In the compression technique discussed above, test vectors are partitioned into given symbol size ((sill) S I S S) where S is the total number of different symbols. Ouring compression, each symbol Sj is mapped to its associated codeword. The compressed test data can be written as an ordering n of the codeword (C,ll), Cn(2),....C,lN)), where N is the total number of symbols per codeword in the test set. The different effects with bit lips are described below.

A. Propagation and sht efect

As it is discussed, bit lip can change the compressed sequence in a complicated manner, which depends on a number of symbols, compression ratio, length of symbol or codeword and ordering technique used. In general, bit lip can affect codeword in two ways.

Codeword which is affected is partitioned into(K ◆ 1) valid codeword.
Codeword which is affected is partitioned into(K ◆ 0) valid codeword and so called dangling suix, which is not a valid code word.

Example: Assume thatcodeword for the Huffman compression is given as 0,10,110,I11.

Now if there is a bit lip in 15t codeword as given in example, then it becomes'l' which is not a valid codeword. If there is a bit lip in the 1st bit of 2nd codeword then it becomes "00", Which gives two different codewords '0'-'0'. But if there is a bit lip in 2nd bit of 2nd codeword then it becomes "11" which is not valid codeword (only dangling sufix).

As shown in case I, if there is a bit lip in the codeword Cn(i), then codeword C[(i+I) to Cn(N) will remain unaffected. So the bit lip will generate K additional codeword C,li-I) and Cn(i+I). As a result, correct sequence of codeword will start rom C,li+1), shited by K additional codeword as shown in fig 3.

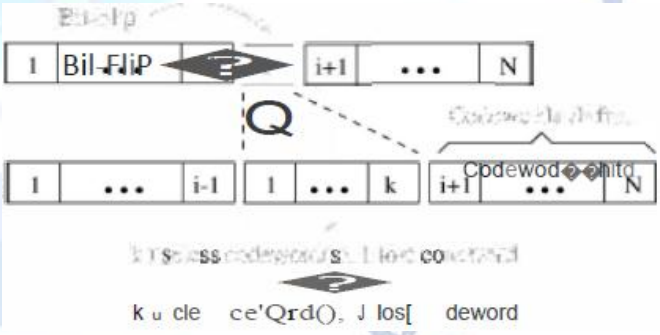

Fig 3. Conceptual view of the shit effect due to a bit lip

Now, as shown in case II, if there is a bit lip in the codeword C,li) and if bit lip creates a dangling suix, then that dangling sufix will form a valid codeword with bits of remaining codeword starting from Cn(i+I). This phenomena is called as propagation of bit lipto codeword C,li+1). Now, if codeword C,li+1) also let with dangling sufix, then that dangling suix will make codeword with bits of C,li+2) codeword. If Cn(i+j) doesn't have dangling sufix, then codewords C[(i+j+1) to C[(N) will be unaffected which is shown in fig 4.

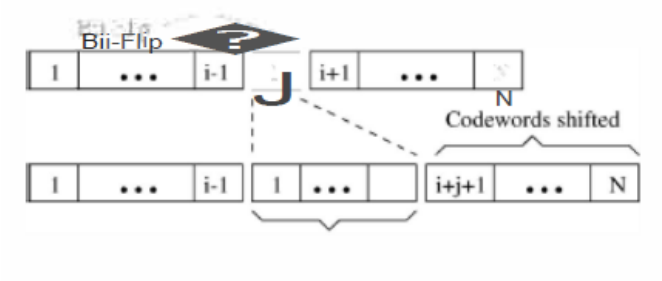

Fig 4. Conceptual view of the propagation effect due to a bit lip

Above two cases signifies the relationship between shit and propagation effect due to bit lip. In the 1st case, codeword Cn(i) is lost and forms another valid codeword which shits remaining code words by at least one codeword.

In the 2nd case, due to bit lip, codeword j+I are lost and dangling suffix will make a codeword by using bits of next codeword. So the remaining code words will shift by minimum one codeword. So it can be said that propagation and shit effect can be differentiated in terms of the number of lost codeword. As it can be seen in the diagram, shit is the special case of propagation where only one codeword is lost, propagation always comes with shit where number of lost codeword depends on the location of the bit lip.

B. Synchronization loss

After decompression, corrupted sequence of bits must appear in the same sequence as it would there in case of no bit flipping. In case of bit flipping, decompressed sequence will be disturbed by the corrupt sequence of bits. If these corrupt bit sequences are different than original one, then there is a loss of synchronization and hence decoding of the received code will be difficult.

C. Bit Flip in case Differentiated Vectors

In the compression technique described above, data is first of all, re-ordered and then differentiation is done to maximize compression.

Assume that 01, O2, 03 • . • . • . . • . . On is the difference vector which can be defined as

01=V1
O2= V2 XOR VI
O"=V,, XOR V,,_1
At the decompression side assigned vectors are retrieved in the following way.

VI = DI
Vz = Dz XOR VI
V" = 0" XOR 0,,-1 XOR .....02 XOR 01
Now in this process, if bit lip occurs in bit OJ, then it will affect on all subsequent vectors during decompression and all vector rom i to n will be affected.

D. Effect of Bit Flip on Fault Coverage of Compressed Data

Assume that the test data is compressed with the compression ratio of 30%. That means, there is 30% minimization in data volume. So it can be said that each 70 bits in the compressed test set can be expanded into 100 bits of the initial uncompressed test set. These bits will be scattered over entire test data, which can affect the multiple test vectors and degrade the fault coverage. Table II shows the average fault coverage with bit lip in Huffman compressed ISCAS circuit test data.

TABLE II BIT FLIP WITH COMPESSION WITH HUFFMAN CODING

Benchmark circuit	%FanIt Coverage			
	t - Bit lip	2 - Bit lip	5 - Bit lip	10 - Bit lip
S27	98.00	95.00	97.00	95.00
S298	97.47	97.64	97.64	97.30
S344	99.00	99.00	99.00	99.00
S349	98.70	98.70	98.70	98.55
S382	97.81	97.81	97.81	97.81
S386	99.39	99.39	98.79	98.79
S400	97.34	97.34	97.34	97.34
S420	99.68	99.68	99.20	99.20
S444	97.33	97.33	97.33	97.33
S510	99.39	99.39	99.39	99.08
S526	97.52	97.52	97.52	97.52
S641	99.13	99.13	99.13	98.83
S713	99.23	99.23	99.04	98.85
S820	98.92	98.92	98.92	98.86
S832	98.34	98.34	98.34	98.28

IV. PROPOSED METHOD TOIMPROVEERROR RESILENCE N CASE OF BIT FLIP

In the previous section, importance of bit lip and its negative impact on fault coverage was described. As a solution of that, the new technique is developed to improve the fault coverage and hence error resilience with compressed test data. A parity bit based error resilience method is described in [14]. Here, in this paper, the advance method for error resilience is proposed based on Hamming distance.

A. Hamming Code based Technique

In communication, The Hamming code is called as linear error code which detects and corrects errors. At the most, it can detect two bits of error and corrects one bit. If the hamming distance between received and transmitted bit pattern is

less than one, then it is called as reliable communication. Hamming code is like a binary linear code. For each code m : 2, there will be m parity bits and 2m -m+1 message bits. Hamming code is the best example of the perfect code which means this code matches the theoretical upper bound on different code words for the given bits and its strength to correct the error bits. If more error correcting bits are added along with the message and if such bits are arranged to get different error results in different incorrect bits, then bad bits can be identified. For example, in (7, 3) Hamming code, there is a chance of error in 7 single bits. So in that 3 parity bits not only signifies the error in data, but also inds the position of the error.

B. General algorithm

1. Signiies the bit numbers in binary form. I.e. for 3 bit code, Bit I = 00I, Bit 2 = 01
2. Bit positions which are powers of 2 are considered as parity bits. e.g. 2° - 1st bit as parity bit
3. Consider all remaining bits as data bits.
4. Parity bits are calculated rom the different combination of data bits as per the binary structure of the bit position.
   Parity bit I includes the data bits which have least signiicant bit 'I' in their bit location. i.e bit 1,3,5,7....
   Parity bit 2 includes the data bits which have least signiicant bit '1' in their bit location. i.e bit 2,3,6,7,....
5. Calculate parity rom the message bits. For simplicity, (7,3) hamming code is taken as an example, where,out of 7 bits, 4 bits are message bits (0], O2, 03, 04) and 3 bits are parity bits (C], C2, C3). Assume that message sequence given here is "0111"

CI = Dlxor D2xor 04 C1 = 0 xor 1 xor 1
C1 = 0
C2 = 01Xor 03xor 04 C2 = 0 xor 1 xor 1
C2 = 0
C3 = 02Xor 03xor 04 C3 = I xor 1 xor I
C3 = I

6. After calculating the parity, again form a full sequence of hamming code. In our example it becomes "0001111".
7. Now suppose at the receiver side instead of "0001111", "0011111" is received. That indicates the bit flipping at some location
8. At the receiver side, decode the code in following mannerAl = Clxor 0lxor 02xor 04 AI = 0 xor I xor 1 xor I

Al = 1
A2 = C2xor 01Xor D3Xor D4 A2 = 0 xor 1 xor 1 xor 1
A2 = 1
A3 = C3xor D2xor D3Xor D4 A3 = 1 xor 1 xor 1 xor 1
A3 = 0
Where A), A2 and A3 bits are answer bits.

9. If A], A2 and A3 are zero, then there is no error in the message. But if any bit is '1' then there is an error.
10. Arranage A], A2 and A3 bits n reverse order. A3A2Al = 011. That indicates there is an error at the 3rd position in fmal hamming sequence. So now lip that bit and solve the error.

Now apply this algorithm to compress test data. Divide whole data set in to blocks of 4 bits. For each 4 message bits calculate the parity bits and detect and correct error.

V. EXPERIMENTAL RESULTS

So the main advantage of Hamming code is that it removes bit lip error, so no loss of fault coverage as shown in table III and hence it improves error resilience. But the biggest disadvantage of this method is bits overhead as shown in table IV

TABLE III. IMPROVED ERORRESILIENCEWITH HAMMINGCODE

Benchmark Circuit	Fault Coverage with bit lips	Improved Fault Coverage with Hamming Technique
S27	95.00	98.00
S298	97.30	97.64
S344	99.00	99.28
S349	98.55	99.28
S382	97.81	97.94
S386	98.79	99.70
S400	97.34	97.59
S420	99.20	99.68
S444	97.33	98.22
S510	99.08	99.85
S526	97.52	97.95
S641	98.83	99.22
S713	98.85	99.23
S820	98.86	99.14
S832	98.28	98.39

Table IV BITS Overhead with Hamming Technique

Bench mark Circuit	Bits Human Communcation	Total Bits Hamming Code	Bits Overhead
S27	18	35	17
	S298	406	714
	S344	390	680
	S349	420	735
	S382	714	1253
	S386	300	525
	S400	798	1400
	S420	928	1624

	S444	735	1288
	S510	402	707

VI. CONCLUSION

Through extensive simulation on the different benchmark circuit, it is shown that the bit lip in compressed test data impacts a lot on fault coverage as compared to uncompressed test data. But with the proposed Hamming code based technique, impact of bit flipping on compressed test data can be significantly nullify. So the proposed technique can improve the error resilience but at the cost of more bits overhead.

REFERENCES

[1] Y. Zorian, "Test requirements for embedded core-based systems and IEEE PI50 0 ," in Proc.IEEE Int. Test Coj, 1997, pp. 191-199.

[2] Touba NA,"Survey of Test Vector Compression Techniques", IEEE D esign and Test of Computers, 20 06, 3(4), pp. 294- 30 3

[3] D. Ha M. Ishida and T. Yamaguchi. Compact: A hybrid method for compressmg test data. IEEE VLSI Test Symposium, pp. 62-69, 1998.

[4] U Mehta, N D evashrayee, K S D as Gupta, "Combining Unspeciied Test D ata Bit Fil 1 ing Methods and Run Length Based Codes to Estimate Compression, Pow er nd Area overhead", IEEE Annual Symposium on VLSI,pp-448,449,20 10

[5] U Mehta, N D evashrayee, K S D as Gupta, "Run-Length-Based Test Data Compression Techniques: How Far from Entropy and Power Bounds?-A Survey", VLSI D esign, Hindawi Publ ication Corporation, PP-9,20 10

[6] A. Chandra nd K. Chkrabarty. "Frequency-D irected Run-Length (FD R) codes with appl ication to system-on-a-chip test data compression," in Proc.IEEE VLSI Test Symp., 20 0 1, pp. 42-47.

[7] A. Chandra and K. Chakrabarty, "System-on-a-chip test-data compression nd decompression architectures based on Gol omb codes," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol . 20 , no. 3, pp. 355-368, Mar. 20 01.

[8] P. Gonciari, B. AI-Hashimi, and N. Nicolici, "Variabl e-l ength input Huffman coding for system-on-a-chip test," IEEE Trans.Comput.-Aided Des.Inter.Circuits Syst., vol . 22, no. 6, pp. 783-796, Jun. 20 03.

[9] U Mehta, N D evashrayee, K S D as Gupta," Weighted Transition Based Reordering, Columnw ise Bit Fil l ing, and D ifference Vector: A Power- Aw re Test D ata Compression Method", VLSI D esign, Hindaw i Publ ication Corporation, PP-9,20 11.

[10] B. West, private communication, Credence Corp

[11] H Parmar, U Mehta, " A statistical test data compression technique with adaptive bit ll ing nd AI based reordering: optimization for compression and scan power", Intenational Jounal of VLSI and Sinal Processing Applications, Vol.I, Issue 2, May2011,pp -15-24

[12] H Parmar, U Mehta, K D asgupta, N D evashrayee," Test D ata Compression Technique for IP Core based SoC using Artiicial Intel l igence", ASDAT,Jan 20 11/

[13] M Sharma," Compression using Hufnan Coding", IJCSNS Intenational Jounal of Computer Science and Network Securiy, VOL. 10 No. 5, May 2010 .

[14] U Mehta, R Trivedi, N Thakkar, "Error Resil ience in Case of Test D ata Compression Techniques for IP Core Based SoC", Intenational Journal of Advance Research in Engineering and Technology, Vol ume 5, Issue I, January (2014), page:24-35.