Frequency Dependant Bit Appending: An Enhancement to Statistical Codes for Test Data Compression

Usha S. Mehta[#1], Kankar S. Dasgupta[*2], Niranjan M. Devashrayee[#3]

[#]*Institute of Technology, Nirma University, Ahmedabad, India*

[1]usha.mehta@nirmauni.ac.in

[3]nmd@nirmauni.ac.in

[*]*Space Application Centre, ISRO, Ahmedabad, India*

[2]ksd@sac.isro.gov.in

Abstract—**Test data compression is a basic necessity for today's test methodology with reference to test cost and test time. This paper presents a compression/decompression scheme based on Frequency Dependant Bit Appending of test vector used with statistical codes. In the proposed scheme, the emphasis is not only on data compression but it aims the data compression with a smaller amount of silicon area overhead for on chip decoder. We have observed that when the number of bits per test vector is prime number or multiplication of prime number (particularly multiplied by 2 or 3), statistical codes gives a large area overhead. The proposed scheme of Frequency Dependant Bit Appending (FDBA) shows that in such cases, if we append few bits at the end of test vector before compression, it improves % compression with very less area overhead. With ISCAS benchmark circuits, it has been shown that when the proposed scheme is applied with statistical coding method, it not only improves % compression, but the area overhead is reduced a lot compared to the base statistical method.**

Keywords— **Test Data Compression, Area Overhead, Frequency Dependant Bit Appending (FDBA), Optimal Selective Huffman Code**

I. INTRODUCTION

One of the major parameter of testing cost of an SoC is cost of Automatic Test Equipment (ATE). How fast the chip can be tested depends on the number of ATE channels, clock rate of channels and required test data.

Test time ≥ (*amount of test data on tester*) / (*number of tester channels X tester clock rate*) [1].

So to reduce the test time and hence test cost, the amount of test data needs to be compressed.

Code based scheme has been widely used for *intellectual property* (IP) cores, since their structure is often hidden from the system integrator [1, 2]. In such cases, no modifications can be applied to the cores or their scan chains, whereas neither automatic test pattern generation nor fault simulation tools can be used. Only precomputed test sets are provided by the core vendors, which should be applied to the cores during testing. The Statistical code is a very popular code for test data compression. There are many variants of Statistical codes like selective Huffman code [4, 5], optimal selective Huffman code [5, 6], RL Huffman Code [7], Multilevel Huffman Code [8] etc. During the analysis of results for % compressions and silicon area overheads for various circuits in case of different statistical codes, we have observed that in case of test vectors with prime numbers or small multiple of prime number (mostly multiplied by 2 or 3), the area overhead was quite high. The scheme proposed in this paper aims with the same good reasonable compression with less area overhead with any statistical coding method used for test data compression.

The paper is organized as follow. Section II describes existing statistical codes. Section III presents the on-chip area overhead in case of test vectors with prime number bits. Section IV introduces the concept of Bit Appending. Section V proposes Frequency Dependant Bit Appending. Section VI describes one example of the on-chip decoder in case of optimal selective Huffman code. Section VII contains experimental results for data compression on ISCAS circuits. Finally conclusions and future work discussion are given in Section VIII.

II. BACKGROUND

The Huffman code is proven to provide the shortest average codeword length requires a large amount of area overhead because its size grows exponentially with symbol size. For the *selective* coding approach, a very simple decoder can be constructed [4]. Jas et al. [5, 6] described a scan vector compression scheme based on selective Huffman coding which codes the most frequently occurring blocks instead of all blocks.

The inefficiency of the Selective Huffman encoding stems from the fact that the required extra bit equally lengthens all data in the compressed test set (encoded or not), irrespective of their occurrence frequency. The optimum selective Huffman code [6] uses an additional Huffman codeword in front of only the unencoded data blocks, relieving the frequently occurring codewords from the extra-bit overhead. The following example illustrates the comparison. The test set T is: 1010 0000 1010 1111; 1111 0000 1010 0001;1010 0000 0010 1010; 0000 1010 1010 0000;1010 1111 1010 0001.

978-1-4244-4859-3/09/$25.00 ©2009

Here the original test data bits are 80. Here the symbol size(m) is 4 and number of symbols coded (n) is 3.

A. Data Compression

Table 1 describes the distinct symbols, the frequency of distinct symbols and codeword for each symbol in case of selective Huffman and Optimal Selective Huffman.

TABLE I
EXAMPLE FOR SELECTIVE HUFFMAN CODE AND OPTIMAL SELECTIVE HUFFMAN CODE

Distinct Patterns	Freq	Selective Huffman Code n=3	Optimal Selective Huffman Code n=3
S0- 1010	9/20	**1**0	**1**
S1- 0000	5/20	**1**10	**01**
S2- 1111	3/20	**1**11	**000**
0001	2/20	**0**0001	**001**0001
0010	1/20	**0**0010	**001**0010

B. Area Overhead

Xrysovalantis Kavousianos et al. has describe the optimal selective Huffman code [6]. In this paper, we propose the FSM for the on chip decoder as shown in fig. 1.

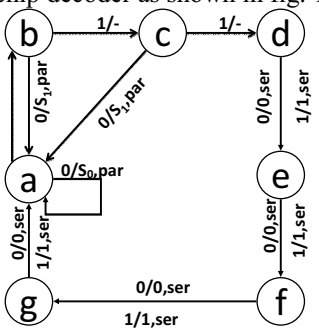

Fig. 1 State Transition Diagram of the FSM Decoder for Optimal Selective Huffman Code.

Fig. 1 describes the example of an implementation of the decoder using simple FSM for a specific case of 3 bits per symbols and 4 symbols to be encoded. There are two inputs to the decoder, one is the tester clock and the other is the serial input from the tester channel. For a symbol size of m, the decoder has m data outputs and two control outputs. The two control outputs are: parallel_load (**Par)** and serial_load (**Ser)**. These two signals control the buffering and loading of data into the serializer when the data has been decoded.

On arrival of the first bit of the codeword, the decoder branches on each bit one at a time until it reaches the end of the codeword at which point it does a parallel loading of the appropriate m-bit block into the register. The codeword "111" indicates that the next m bits are original uncoded data bits. So *ser* signal will be high for next m cycles and will load next m bits serially from coded data.

As silicon area overhead for on chip decoder is directly proportion to the states of decoder FSM and the memory requirement to store the symbol words. To make the discussion technology independent, the bits required to implement the FSM and memory will be compared throughout this paper. For the selective Huffman code or optimal selective Huffman code, $number\ of\ states\ in\ FSM = umber\ of\ symbols\ coded + number\ of\ bits\ /symbol$. Here the hardware requirement to store the distinct symbols which are coded is ROM / RAM of size n X m bits. The hardware requirement to implement the FSM is fsm = $\lceil \log_2(n+m) \rceil$ bits.

C. Comparison of Compression and Area Overhead

TABLE II
AREA OVERHEAD AND COMPRESSION COMPARISON FOR STATISTICAL CODES

Huffman Code	% Compression	Area Overhead for Decoder		
		FSM Bits	Memory Bits	Total
Selective	28.80	3	12	15
Optimal Selective	38.75	3	12	15

As shown in table II, Optimal Selective Huffman coding gives the approximately same area overhead (a little difference because of controlling signals) as Selective Huffman coding but it gives better compression than Selective Huffman. So for our further discussion, optimal selective Huffman Code is selected.

III. TEST VECTORS WITH PRIME NUMBER BITS

Our observation says that the above schemes, gives a very large area overhead for the cases where the total no. of bits (t) per vector is a prime number. In such cases, the possible symbol sizes are either m =1 or m = t. Taking m = 1 for any coding style, gives no compression. So only possible option is m = t. It is to be noted here that as for the given test set, none of the test vector will be repeated within the test set, when we select symbol size equals to the length of test vector, each test vector itself becomes a distinct symbol which needs to be coded. So for m = t, it is same as Full Huffman coding and gives very high area overhead. In the example in table III, the test set T_1 is similar to test set T but one extra bit in each test vector. The only possible symbol size is m = 17.

TABLE III
EXAMPLE FOR TEST SET WITH PRIME NUMBER OF BITS

Distinct Symbols	Occur. Freq.	Optimal Selective Huffman Code n=3
1010 0000 1010 1111 1	1/5	**01**
1111 0000 1010 0001 0	1/5	**000**
1010 0000 0010 1010 1	1/5	**001**
0000 1010 1010 0000 0	1/5	**1**00001010101000000
1010 1111 1010 0001 1	1/5	**1**10101111101000010

% compression = (85 – 44) /85 = 48.24
The area overhead: fsm bits=5; ROM / RAM bits= 51

TABLE IV
AREA OVERHEAD AND COMPRESSION COMPARISON FOR TEST SET T AND T_1

Test Set	% Compression	Area Overhead for Decoder		
		FSM Bits	ROM/RAM Bits	Total Bits
R_T	38.75	3	12	15
R_{T1}	48.24	5	51	56
% Increase in	24.49	66.67	325	273.3

Table IV describes the comparison for both test set. Here the original test data for both the test set are nearly similar, the number of symbols selected for encoding is also same. But when the % compression and area overhead are compared, the result says that the area overhead has been increased drastically compared to % compression.

IV. Appending Zeroes to Test Vector

The main reason for excessive area overhead in case of test vectors with prime numbers of bits is that the test vector could not be divided into small symbols and hence the symbol size is too large which in turn requires large FSM and larger memory storage. Our approach here is to convert this undividable test vector in to divisible vectors so it can be cut it in to smaller symbols. The scheme proposed here is to add some extra bits at the end of test vector such that the total number of bits per test vector is divisible by a preferred number (i.e. in our case, bits per symbols (n)). The % compression and area overhead are dependent on two variables: 1. the bits per symbol (n) and 2. the encoded number of symbols out of total distinct symbols (m). To decide these two numbers is a computational sensitive task.

V. Appending Frequency Dependant Bits

When the zeroes are appended to test vectors with prime number of bits, the following observations are noteworthy: 1. The total no of distinct symbols are increased. So frequency distribution for distinct symbols was affected negatively as it was broadening which had affected the selection criteria for number of symbols to be encoded and finally the overall quality index I. The main concern here is that the bits should be appended to test vectors but it should not affect the frequency distribution negatively. For the solution of this problem, we propose a scheme based on frequency dependant bit appending. The concept here is to append the bits in such a way that it does not add any new symbol but it creates the symbol same as symbol with the highest possible frequency. In this case, each new symbol created by bit appending will strengthen the frequency distribution of symbols and improves the overall quality index. The following steps describe the proposed scheme.

1. Input the given test set file in text mode.
2. Calculate the total number of test vectors (t) and total number of bits per each test vector (b).
3. Is the bits/ test vector a prime number? If no, go to the end
4. For n = 2 to $\left\lfloor \frac{b}{2} \right\rfloor$, do the following steps
   a. Divide each vector into symbols of n bits and separate the last r = rem (b, n) bits from each vector.
   b. Find out the total number of distinct symbols (m) and frequency corresponding to each symbol.
   c. For each test vector, now take the remainder bits r and find out a match for it with highest possible frequency from distinct symbols and append the remaining n-r bits of test vector same as last n-r bits of that highest frequency symbol. If the remainder is not matching with any of the distinct symbols than append the necessary zeroes. In such case, a new symbol will be formed and the modified number of distinct symbols will be m_1.
   d. For j = 1 to m_1, Apply the optimal selective Huffman Code for each case where the number of symbols to be coded is j and calculate the % compression C.
   e. For the minimum % compression, Calculate the area overhead S = (n X m) + $\lceil \log_2(n+m+1) \rceil$ bits.

VI. On Chip Decoder for Frequency Dependant Bit Appending

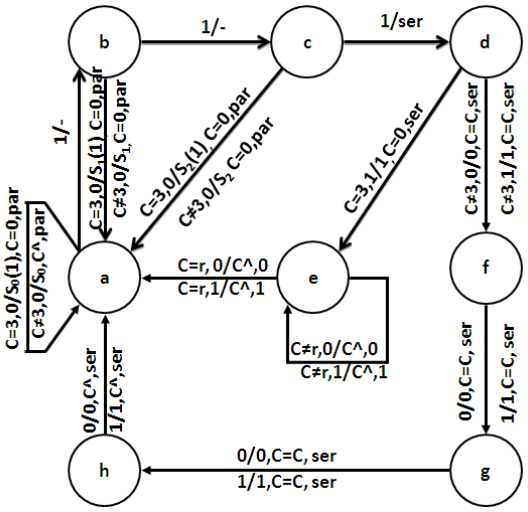

Fig. 2 State Transition Diagram of the FSM Decoder for Frequency Dependant Bit Appending with Optimal Selective Huffman Code.

Fig. 2 describes an implementation of the decoder using simple FSM for a specific case of 3 bits per symbols and 4 symbols to be encoded. The inputs and outputs are same as in FSM for optimal selective Huffman decoder. Only one signal *count* (c) is added to indicate the index of symbol being decoded. Counter counts up to total symbols per vector. The concept is divided in to two parts.

A. If c is less than 3, it means the code word is not corresponding to last symbol of test vector. On arrival of the first bit of the codeword, the decoder branches on each bit one at a time until it reaches the end of the codeword at which point it does a parallel load of the appropriate m-bit block into the parallel and c is incremented by 1 to indicate that one codeword is successfully decoded and corresponding symbol is loaded. The codeword "111" indicates that the next m bits are original uncoded data bits. So *ser* signal will be high for next m cycles and will load next m bits serially from coded data.

B. If c is equal to 3, the code word is corresponding to last symbol of test vector. On arrival of the first bit of the codeword, the decoder branches on each bit one at a time until it reaches the end of the codeword at which point it does a parallel load of the appropriate r bits from n bits symbols into the parallel, c is set to 0 to indicate that one complete test vector is successfully decoded and corresponding symbol is loaded.. The codeword "111" indicates that the next n bits are original uncoded data bits. Only r bits (1 bit in this case) should be passed to dut

through serial load. Remaining n-r bits should be discarded. The same counter is used to count up to r and when it reaches to r, FSM moves to initial state.

Here the extra hardware requirement is one counter of $count = \left\lceil \frac{b}{m} \right\rceil$ bits.

VII. Experimental Results for Data Compression and Area Overhead for On Chip Decoder

TABLE V
% compression for ISCAS 74L85 Circuit

m	# Bits per symbol (n)					
	2	3	4	5	6	11*
1	12.6	12.6	18.6	13.0	16.2	-1.2
2	16.6	24.9	24.1	21.3	24.1	2.0
3	21.3	27.7	27.7	26.5	28.5	5.1
4		29.6	30.8	27.3	31.6	7.9
5		30.8	31.2	27.7	35.2	10.7
6		32.0	31.6	28.1	35.6	13.4
7			32.0	28.1	36.8	16.2
8			32.4	28.5	37.5	18.6
9			33.2	29.2	38.7	20.9
10			34.8	30.0	39.9	23.3
11				30.8	41.5	25.7
12				32.0	**43.9**	28.1
13				34.0		30.4
14						32.8
15						35.6
16						38.3
17						41.1
18						**43.9**

In table V, the % compression for various values of number of bits per symbol (n) and number of symbols coded (m) in case of optimal selective Huffman Code for the ISCAS 85 benchmark circuit 74L85 is presented. Column 7 of table V indicates the % compression when the symbol size is equal to no. of bits per vector. i.e. no appending is done. While column 2 to column 6 is the % compression for different symbol size i.e. appending is done here. For the case of bit appending, the maximum possible data compression is 43.9% for which the number of bits per symbol (n) is 6 and number of symbols encoded (m) is 13. Now for the nearly same % (coincidently in this case, both are exactly same) compression in case where no bit appending is done (i.e. in column 7), the number of bits per symbol (n) is 11 and number of symbols encoded(m) is 19.

For various ISCAS benchmark circuits, table VI shows that by applying FDBA method and proper selection of n and m, the % area overhead is reduced by a large amount and % compression is also improved by a small amount in most of the cases.

VIII. Conclusion

In this paper, with the example of Selective Huffman code and optimal selective Huffman code, it was shown that the statistical coding scheme is not suited for the case where the number of bits per test vector is a prime number or multiple of prime number in terms of area overhead and compression. A review was also demonstrated showing the inefficiency of above schemes. If the proposed scheme "Frequency Dependant Bit Appending (FDBA)" is applied on test set before applying the optimal selective Huffman coding with a properly selected symbol size, it guarantees a large amount of reduction in area overhead with improved or nearly same compression efficiency.

References

[1] Nur A. Tauba, "Survey of Test Vector Compression Techniques" *IEEE transaction Design & Test of Computers, Volume 23 Issue 4 , July 2006,*

[2] Usha Mehta, N. Devashrayee, K. Dasgupta, "Survey of Test Data Compression Techniques Emphasizing Code Based Scheme" *IEEE 12th Euromicro Conference on Digital System Design (DSD09)*, August 09

[3] Jas and N.A. Touba, "Test Vector Compression via Cyclical Scan Chains and Its Application to Testing Core-Based Designs," *Proc. Int'l Test Conf. (ITC 98), IEEE CS Press,* 1998, pp. 458-464.

[4] Jas, J. Ghosh-Dastidar, and N. A. Touba, "ScanVector compression/decompression using statistical coding," in *Proc. VLSI Test Symp.*, 1999, pp. 114–120

[5] Abhijit Jas, Jayabrata Ghosh-Dastidar, Mom-Eng Ng, and Nur A. Touba, "An Efficient Test Vector Compression Scheme Using Selective Huffman Coding" *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, Vol. 22, No. 6,* June 2003

[6] Xrysovalantis Kavousianos, Emmanouil Kalligeros, Dimitris Nikolos, "Optimal Selective Huffman Coding for Test-Data Compression" *IEEE Transactions on Computers , Volume 56 Issue 8 ,* August 2007

[7] Mehrdad Nourani, Mohammad H. Tehranipour, "RL-Huffman encoding for test compression and power reduction in scan applications" *Transactions on Design Automation of Electronic Systems (TODAES) , Volume 10 Issue 1* January 2005

[8] Xrysovalantis Kavousianos, Emmanouil Kalligeros, Dimitris Nikolos, "Multilevel-Huffman test-data compression for IP cores with multiple scan chains", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems , Volume 16 Issue 7* , July 2008

TABLE VI
Area Overhead Comparison for ISCAS 74L85 Circuit

ISCAS Bench mark Circuit	# of total bits/ test vector	# of total test vectors	No. of bits/symbol		No. test vectors coded		% Compression		Area Overhead in bits		% improvement with FDBA method	
			When FDBA not applied	When FDBA applied	When FDBA not applied	When FDBA applied	When FDBA not applied	When FDBA applied	When FDBA not applied	When FDBA applied	Compression	Area Overhead
74L85	11	23	11	6	19	13	43.9	43.9	214	83	0	61.2
C499	41	52	41	7	34	36	54.8	56.2	1471	258	2.6	82.5
C6288*	31	12	31	7	5	5	35.2	39.0	161	16	10.8	90