2010 Asia Pacific Conference on Circuits and Systems (APCCAS 2010)
6 – 9 December 2010, Kuala Lumpur, Malaysia

Combining Unspecified Test Data Bit Filling Methods and Run Length Based Codes to Estimate Compression, Power and Area overhead

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

[#] *EC, PG VLSI Design, Institute of Technology,*
Nirma University, Ahmedabad, India
[1]usha.mehta@nirmauni.ac.in
[3]nmd@nirmauni.ac.in

[*] Space Application Center
Indian Space Research Organization, Ahmedabad, India
[2]ksd@sac.isro.gov.in

***Abstract*—For SoCs (Sea of Cores!) which contains a large amount of IP cores with pre computed test data, the code based test data compression scheme is more suitable as it does not require any knowledge of internal nodes of IP. The data compression of any partially specified test data depends upon how the unspecified bits are filled with 1s and 0s. In this paper, the five different approaches for don't care bit filling based on nature of runs are proposed. These methods are used here to predict the maximum compression based on entropy relevant to different run length based data compression code. These methods are also analyzed for test power and area overhead corresponding to run length based codes. The results are shown with various ISCAS circuits.**

***Index Terms*— Unspecified Test Data, Code Based Data Compression Methods, Compression Predicted by Entropy, EFDR, FDR, MFDR**

I. INTRODUCTION

The number of tests, corresponding test data volume and test time increase with each new fabrication process technology. Conventional external testing cannot proceed any faster than the amount of time required to transfer the test data: *Test time* ≥ (*amount of test data on tester*) / (*number of tester channels X tester clock rate*) [1]

As a result, some companies are looking for compression well beyond 100X tester cycle reduction. [13]. Test data compression involves adding some additional on-chip hardware before and after the scan chains. This additional hardware decompresses the test stimulus coming from the tester. This permits storing the test data in a compressed form on the tester. With test data compression, the tester still applies a precise deterministic (ATPG-generated) test set to the circuit under test (CUT).

The paper is organized as follows. Section II contains data compression techniques and existing run length based codes. Section III introduces the different methods of don't care bit filling for run length base code. Section IV introduces entropy. Section V present the experimental results of test data compression for various X filling methods compares the actual data compression for various methods claimed in literature with maximum possible compression predicted on basis of entropy. Section VI presents results of average and peak test power with different methods of X filling. Section VII compares the area overhead. Section VIII analyzes the nature of test data on basis of various experimental results. On-chip area overhead is described in section VIII. Conclusion is given in Section VIII.

II. CODE BASED DATA COMPRESSION TECHNIQUES

The quantity of test data rapidly increases, while, at the same time, the inner nodes of dense SoCs become less accessible from the external pins. The testing problem is further exacerbated by the use of *intellectual property* (IP) cores, for which 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. In this context, code based test data compression technique satisfies both the requirements [2].

The first data compression codes that researchers investigated for compressing scan vectors were encoding by runs of repeated values.

Modified run length coding [3] encodes runs of 0s. Golomb Code [4] encodes runs of 0s with variable-length code words that contains a group prefix and tail. FDR [5] is based on the observation that the frequency of runs of 0s with run length less than 20 is high. The inefficiency of FDR for compressing data streams that are composed of both runs of 0s and runs of 1s is improved by an extension of FDR [6]. By encoding both types of runs same way as FDR but adding

an extra bit at beginning of FDR codeword is added to indicate the run length type. The test set T is composed of alternating runs of zeros and run of ones. The alternating run-length [7] code is a variable-to-variable-length code with an additional parameter- the alternating bi*nary* variable *a.* In Shifted Alternating FDR Code [8], each symbol is made up of only 1s or only 0s. MFDR [9] is based on probability of 0s. The test patterns are reordered based on the Hamming distance between them in Run Based Reordering [10].

III. Don't Care Bit Filling for Run Length Based Codes

To get the maximum compression, the don't care bits should be filled to get the longer runs. How the runs are formed depends upon the type of run length code. The types of runs used by various run length codes can be classified as follows: 1. runs of 'zeros followed by one' 2. runs of 'ones followed by zero' 3. runs of 'ones followed by zero' and 'zeros followed by one' 4. alternating runs of 'zeros followed by one and ones followed by zero' 5. alternating runs of 'only zeros' and 'only ones'. To get the maximum data compression, the don't care bits should be filled with 1 or 0 as per the nature of code.

a. Runs of ' Zeroes Followed by One'

For codes like Golomb [3], FDR [5] or MFDR [9], the symbols are made of runs of 0s followed by bit '1'. So we have applied a simple technique of replacing all the Xs with 0s. So overall the runs of 0s will increase and data compression will increase.

b. Runs of 'Ones Followed by Zero'

This is a hypothetical case introduced to analyze the compression results for VLSI test data. The symbols are made of runs of 1s followed by bit '0'. Here all Xs are replaced by 1s. So the runs of 1s will increase.

c. Runs of 'Zeros Followed by One' and ' Runs of Ones Followed by Zero'

The code Extended FDR [6] is a case which accepts runs of 0s as well as runs of 1s. Here each symbol is a run of 0s followed by bit '1' or run of 1s followed by the bit '0'. If the last symbol is a run of 0s (1s) without any follower bit '1' (0), in that case, it would be counted as a symbol of the run length equal to number of 0s (1s). The X filling is done in such a way that it should maximize the run length as well as it should not introduce any new symbol. While filling the X, the logic is that if the symbol has ended just before the position of X, the X should be filled with reference to next symbol. But if there is a continuous symbol going on at position of X, X should be filled such that it increases the run length of current symbol. Proposed algorithm needs to do back tracking as well as forth tracking.

d. Alternating Runs of 'only Zeros' and Runs of 'only Ones'

For code like Shifted Alternate FDR [8], the symbols are made of alternate runs of 0s and runs of 1s without any follower bit. So we have applied a simple technique of replacing the Xs with last non-X value i.e don't cares following a run of zeros are mapped to zeros, and don't cares following a run of ones were mapped to ones to minimize the number of runs.

e. Alternating Runs of 'Zeros followed by One' and 'Runs of Ones followed by Zero'

For code like Alternate FDR [7], the symbols are made of alternate runs of 0s followed by one and runs of 1s followed by zero. Here the first run must be of zero type. So if there is any X at first bit position, it is replaced by 0. If first bit is '1', then the first run is of 0 length and then it starts with 1. After than all the X bits are filled with last non-X value.

Considering above cases, we can divide these methods in two categories: 1. considering the runs of one type only i.e. either runs of 0s or runs of 1s. 2. Considering runs of both the type i.e. runs of 0s as well as 1s. For second category, after X filling, there may be runs of same run length but of different run type. While identifying the unique symbols, such runs are taken as two different symbols in this paper.

IV. Entropy

Entropy is an important concept to data compression. The entropy of a symbol E(s) is the minimum number of bits needed to encode that symbol. The entropy of the test set is calculated from the probabilities of occurrence of unique symbols using the formula $E(s) = \sum_{i=1}^{k} pi \log_2 \frac{1}{pi}$ where *pi* is the probability of occurrence of symbol *xi* in the test set and k is the total number of unique symbols. In case of fixed symbol length, the formula for maximum compression that can be achieved is given by $\frac{symbol\ length - entropy}{symbol\ length}$ and in case of variable symbol length, the maximum compression is equal to $\frac{avg\ symbol\ length - entropy}{avg\ symbol\ length}$.

The average symbol length is computed as $\sum_{i=1}^{n} pi\ |Xi|$ where *pi* is the probability of occurrence of symbol *xi*, |*xi* |is the length of symbol *xi*, and *n* is the total number of unique symbols [11]. Mathematically it can be proved that the following formula for maximum compression is valid for fixed symbol length as well as variable symbol length. $\%\ compression = \frac{[T - (S\ X\ E)]}{T} X\ 100$ where T is the total number of bits in original uncoded test data, S is the total number of symbols needed to be encoded and E is the entropy. For all further discussion, the above formula of % compression is used.

V. Experimental Results For Test Data Compression

We have implemented all the X filling techniques using MATLAB7.0 language. The experiments are conducted on a workstation with a 3.0 GHz Pentium IV processor and 1GB of memory. Experiments were performed for X filling to calculate the theoretical limit on test data compression for the dynamically compacted test cubes generated by MINTEST for the largest ISCAS'89 benchmark circuits. These are the same test sets used for experiments in [3], [5], [6] and [7]. The compression values in table I are predicted from the exact values of entropy that were generated after the X filling. In table I and successive all tables in this paper, the X filling methods A, B, C, D and E indicates the same five methods of X filling methods described section III respectively. As can be seen from the table I, the percentage compression that can be achieved is maximum where the runs are considered of both the types i.e. runs of 0s and runs of 1s. Note, however, that these entropy bounds would be different for a different test set for these circuits. If the reordering or any such other method is used to change the location or number of don't cares, the entropy can be different. However, given any test set, the proposed method can be used to determine the corresponding entropy bound for it.

TABLE I
Comparison of % Compression for various Schemes of X Filling

ISCAS Circuit	Test Data Bits	% Compression for Various X Filling Methods A, B, C, D, E				
		A	B	C	D	E
S5378	23754	52.36	54.33	**56.38**	40.73	38.94
S9234	39273	47.80	44.34	**53.37**	35.92	35.33
S13207	165200	83.65	81.12	**85.55**	79.72	79.59
S15850	76986	68.18	62.49	**71.90**	60.72	60.32
S38417	164736	54.50	57.44	**65.84**	55.77	55.14
S38584	199104	62.49	60.09	**66.67**	54.65	54.21

Table II compares the % compression claimed in literature for each of this category of coding with its theoretical upper limit predicted by entropy. It should be noted that the % compression claimed here as the upper bound predicted by entropy is achieved after filling all don't care bits with appropriate method of bit filling but without applying any technique like reordering of test vector or difference vector. Considering % compression, MFDR gives best compression for codes based on 'runs of zeros'. It can be seen that because of reordering and other techniques, in some of the cases of MFDR, the % compression achieved is even higher than the predicted by entropy. The EFDR gives the overall best performance.

VI. Experimental Results with X filling for Minimum Test Power

The goal of X filling was to reduce the number of runs. As the number of runs will decrease, the number of transitions should be reduced, which should lower test power. In this paper, a widely used weighted transition metric (WTM) introduced in [12] is used to estimate the average and peak power consumption. Test data $T = \{T_1, T_2, .., T_m\}$ have m patterns, and the length of the pattern is n bits. Each test pattern $Ti = \{t_{i1}, t_{i2}, ..t_{in}\}$, $1 \le i \le m$, $1 \le j \le n$ denotes the j-th bit in the i-th pattern. Weighted transitions metric WTM_j for T_j, the average test power P_{avg} and peak power P_{peak} are:

$$WTM_j = \sum_{i=1}^{n-1} (n-i) * (t_{j,i} \oplus t_{j,i+1})$$

,

$$P_{avg} = \frac{\sum_{j=1}^{m} WTM_j}{m}$$

and

$$P_{peak} = \max_{1 \le j \le m} WTM_j$$

Intuitively, the average power and peak power for test data should be minimum when there are long runs of ones as well as zeros. This is proved in table III. Peak power and average power is minimum when the X filling is done for alternating runs of 0s and 1s without any follower bit. represents the Table IV provides the comparison of total number of symbols needed to be encoded, entropy and total number of distinct symbols for five different case of don't care bit filling methods.

VII. On-Chip Decoder Area

The decoder used for run length based test data compression techniques are test data independent. For on-chip decoder area comparison, the decoder FSM described in literature for Golomb[3], FDR[5], EFDR[6], ASFDR[7], MFDR[9] are implemented using VHDL. The EDA tolls used are Mentor graphics HDL designer for design entry, Modelsim for simulation and Leonardo spectrum for synthesis. The library used for synthesis is TSMC 0.35u (Taiwan Semiconductor Machine Corporation). The table V shows the comparison in terms of No. of NAND gates and No. of Nets. It also represents the maximum clock frequency for decoder.

TABLE V
Comparison of On-Chip Decoder

Coding scheme	No of NAND gates	No of ports	No of nets	Maximum Clock Frequency (MHz)
GOLOMB	580	3	361	29.7 MHz
FDR	1104	3	797	24.6 MHz
EFDR	1078	3	757	13.9 MHz
ASFDR	1111	3	799	11.3 MHz
MFDR	2199	3	1493	11.0 MHz

From table V, it can be seen that, the area overhead for Golomb is minimum while the MFDR is maximum. The same way, the maximum clock frequency for Golomb code is quite high compared to MFDR.

VIII. Conclusion

In this paper, we have covered the wide variety of test data compression techniques based on run length scheme and their variants. Five different techniques of don't care bit filling based on nature of runs are used to increase the run length and hence % compression. The entropy based % compression for each of these five techniques is calculated and analysis proves that run length based code which includes

run of ones followed by zero as well as run of zeros followed by one gives best compression for VLSI test data. The same conclusion is further emphasized by comparison of actual compression claimed by literature where EFDR gives the maximum compression. The run based reordering and other techniques used to enhance the run length, further improves compression which is proven by Run Based Reordering with Extended FDR scheme. The area overhead for Golomb is very less compared to EFDR or MFDR. The researchers can start with this method and explore the possibilities of further compression with consideration of area overhead of on-chip decoder, over all test time and test power.

REFERENCES

[1] N. Tauba, "Survey of Test Vector Compression Techniques" *IEEE transaction Design & Test of Computers , 2006*

[2] U. 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] A. Chandra and K. Chakrabarty, "Test Data Compression for System-on-a-Chip Using Golomb Codes" ***VTS '00:*** *Proceedings of the 18th IEEE VLSI Test Symposium, 2000*

[4] L. Li and K. Chakrabarty, "On using exponential – Golomb codes and subexponential codes for System-on-Chip Test data compression" *Journal of Electronic Testing: Theory and Applications, Volume 20 Issue 6,* December 2004

[5] A. Chandra and K. Chakrabarty, "Frequency-Directed Run-Length (FDR) Codes with Application to System-on-a-Chip Test Data Compression" in the *Proceedings of the 19th IEEE VLSI Test Symposium,* March 2001

[6] A. El-Maleh and R. Al-Abaji, "Extended frequency-directed run-length code with improved application to system-on-a-chip test data compression" *Proc. Int. Conf: on Electronics, Circuits and Systems, 2:449-452,*Sep 2002

[7] A. Chandra and K. Chakrabarty, "Reduction of SOC test data volume, scan power and testing time using alternating run-length codes", *DAC '02: Proceedings of the 39th conference on Design automation,* June 2002

[8] S. Hellebrand and A. Würtenberger, "Alternating Run-Length Coding –A Technique for Improved Test Data Compression", *Handouts 3rd IEEE International Workshop on Test Resource Partitioning, Baltimore, MD, USA*, October 10 – 11, 2002

[9] J. Feng and G. Li, "A Test Data Compression Method for System-on-a-Chip" *4th IEEE International Symposium on Electronic Design, Test and Applications, 2008. DELTA 2008*

[10] H. Fang and et. al., "RunBasedReordering: A Novel Approach for Test Data Compression and Scan Power" ***ASP-DAC '07:*** *Proceedings of the 2007 conference on Asia South Pacific design automation*

[11] K. Balakrishnan and N. Touba, "Relating Entropy Theory to Test Data Compression" *Proceedings of the European Test Symposium,* May 2004

[12] R. Sankaralingam et. al., "Static compaction techniques to control scan vector power dissipation" *Proc. IEEE VLSI Test Symp., pages 35-40,* 2000.

[13] http://www.reed-ectronics.com/tmworld

TABLE II
COMPARISON OF % COMPRESSION PREDICTED BY ENTROPY WITH CORRESPONDING ACTUAL COMPRESSION CLAIMED IN LITERATURE

ISCAS Circuits	Original Test Data Bits	Codes Based on Run Length of Zeroes				Code Based on Run Length of Zeros and Ones		Code Based on Alternate Runs of Zeros and Ones	
		Golomb Code[3]	FDR [5]	MFDR [9]	% Compression Predicted by Entropy	EFDR [6]	% compression Predicted by Entropy	Alternating FDR [7]	% Compression Predicted by Entropy
S5378	23754	40.70	48.02	51.47	52.36	51.93	56.32	-NA-	51.44
S9234	39273	43.34	43.59	**57.74**	47.80	45.89	53.37	44.96	47.26
S13207	165200	74.78	81.30	83.42	83.65	81.85	85.54	80.23	82.45
S15850	76986	47.11	66.22	66.93	68.18	67.99	71.90	65.83	67.25
S38417	164736	44.12	43.26	**57.95**	54.50	60.57	65.84	60.55	62.93
S38584	199104	47.71	60.91	59.32	62.49	62.91	66.67	61.13	62.23

TABLE III
COMPARISON OF PEAK POWER AND AVERAGE POWER FOR VARIOUS METHODS OF DON'T CARE BIT FILLING

ISCAS Circuit	Peak Power				Average Power			
	A	B	C	D	A	B	C	D
S5378	12085	12375	**11732**	11522	4300	4087	**3524**	3526
S9234	15395	15640	**14092**	14103	6706	6521	**4002**	4022
S13207	110129	126820	**94879**	94886	12318	1453	**8073**	7887
S15850	84360	88794	**70875**	70894	19448	25636	**13611**	13659
S38417	514716	539019	**437884**	437935	194843	193140	**118100**	118080
S38584	530464	533975	**481158**	481171	133320	142220	**86135**	86305

TABLE IV
COMPARISON OF TOTAL NO. OF SYMBOLS NEEDED TO BE ENCODED, ENTROPY AND TOTAL NUMBER OF UNIQUE SYMBOLS

ISCAS Circuits	Total No. of Symbols Needed to be Encoded					Entropy					Total No. of Unique Symbols				
	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
S5378	3538	2969	**2094**	3129	3130	3.20	3.65	**4.95**	4.50	4.63	79	87	**141**	131	136
S9234	4817	5786	**3216**	4904	4905	4.26	3.78	**5.69**	5.13	5.17	74	86	**141**	133	134
S13207	5021	6294	**3550**	5427	5428	5.38	4.95	**6.72**	6.17	6.21	211	178	**331**	314	314
S15850	5330	7329	**3647**	5628	5629	4.60	3.94	**5.93**	5.37	5.42	173	145	**234**	230	229
S38417	29473	23110	**9548**	13751	13752	2.54	3.03	**5.89**	5.29	5.37	121	154	**236**	233	238
S38584	16814	18474	**10771**	16275	16275	4.44	4.30	**6.16**	5.54	5.60	214	191	**345**	333	333