How Effective are Compression Codes for Reducing Test Data Volume?[1]

Anshuman Chandra[†], Krishnendu Chakrabarty[†] and Rafael A. Medina[§][⋆]

[†] Dept. Electrical & Computer Engineering   [§] Dept. Electrical Engineering & Computer Science

Duke University, Durham, NC 27708   Massachusetts Institute of Technology, Cambridge, MA 02139

Abstract

Run-length codes and their variants have recently been shown to be very effective for compressing system-on-a-chip (SOC) test data. In this paper, we analyze the Golomb code, the conventional run-length code and the FDR code for a binary memoryless data source, and compare the compression obtained in each case to fundamental entropy bounds. We show analytically that the FDR code outperforms both the conventional run-length code and the Golomb code for test resource partitioning (TRP) based on data compression. We also present a modified compression/decompression architecture for obtaining even higher compression. We demonstrate the effectiveness of these compression codes using the larger ISCAS-89 benchmark circuits and two representative circuits from industry. Finally, we show that the FDR code is almost as effective as Unix utilities gzip and compress, even though it uses a much simpler decompression algorithm.

1 Introduction

Test automation remains a major bottleneck in the plug-and-play design of system-on-a-chip (SOC). New techniques are needed to test complex SOCs consisting of several intellectual property (IP) cores. Each of these cores must be exercised with a large number of precomputed test patterns. The volume of test data and the testing time for an SOC are growing rapidly as IP cores become more complex and an increasing number of cores are integrated on a chip.

Test resource partitioning (TRP), in which some of the test resources are moved from the automatic test equipment (ATE) to the chip, offers a promising solution to the problem of rising test data volume and testing time [10]. One approach to TRP is based on test data compression [2]. In this approach, a precomputed test set T_D for an IP core is compressed (encoded) to a much smaller test set T_E, which is stored in ATE memory. An on-chip decoder is used for pattern decompression to obtain T_D from T_E during test application [3, 4, 5, 6]. Compression codes have recently been shown to be very effective in reducing test data volume and testing time [4, 6]. Here, we address the following important questions related to TRP based on test data compression:

- What are the fundamental compression limits of the codes proposed for TRP?
- How effective are these codes for representative industrial circuits?
- What techniques and variants can be used to increase the compression further?
- How do these codes compare to standard Unix file compression utilities such as *gzip* and *compress*?

We systematically address each of these issues and show that compression codes are especially suitable for TRP.

In order to efficiently encode SOC test data, we can use conventional run-length codes to map variable-length blocks of data to fixed-length codewords. These codes are however less efficient than more general variable-to-variable-length codes [7, 8], especially if hardware-based decoding is necessary. Instead of using a run-length code with a fixed block size b, we can achieve greater compression by using Golomb coding, which maps variable-length runs of 0s in the data sequence to variable-length codewords [7]. Yet another approach to run-length coding is based on the use of the frequency-directed run-length (FDR) code [6], which takes into account the specific properties of the data source. The on-chip decompression hardware is extremely small for the Golomb and the FDR code [4, 6].

In this paper, we examine the fundamental limits of compression achieved by the Golomb code, the conventional run-length code and the FDR code. We compare the compression achieved with the three codes for a binary memoryless data source, and show that the FDR code outperforms both the conventional run-length code and the Golomb code. We also show that the average codeword size for the FDR code is smaller than that for the Golomb code. This analysis is necessary to explain the high percentage compression reported in [4, 6] for the ISCAS-89 benchmark circuits. Such a rigorous analysis has not been attempted before for these codes. In particular, the FDR code was developed only recently [6] and hence has not been studied by the information theory community.

It was shown in [6] that the FDR code is very effective for compressing deterministic test sequences. We now present a novel technique to further increase the amount of compression for precomputed test sets. We show that the hardware overhead for the proposed scheme is small and requires only a simple redesign of the decompression logic for the basic FDR code. In addition, we apply the Golomb

---

Run-length	Golomb code ($m = 4$)				FDR code			
	Group	Group prefix	Tail	Code-word	Group	Group prefix	Tail	Code-word
0			00	000		0	0	00
1	A_1	0	01	001	A_1		1	01
2			10	010			00	1000
3			11	011	A_2	10	01	1001
4			00	1000			10	1010
5	A_2	10	01	1001			11	1011
6			10	1010			000	110000
7			11	1011			001	110001
8			00	11000			010	110010
9	A_3	110	01	11001	A_3	110	011	110011
10			10	11010			100	110100
11			11	11011			100	110100
...	...	...	...	...	...	...	...	...

Figure 1. An example of Golomb and FDR coding.

and FDR codes to the scan test data for two real-life microprocessor designs from IBM. We demonstrate that the compression results obtained for these circuits are not only exceptionally high, but they are extremely close to fundamental entropy bounds. Finally, we show that the FDR code is almost as effective as the Unix file compression utilities *gzip* and *compress*. This is particularly striking since *gzip* and *compress* employ sophisticated compression algorithms, and the corresponding decompression utilities (*gunzip* and *uncompress*) are implemented in software.

The rest of the paper is organized as follows. We present a rigorous analysis of the conventional run-length code, the Golomb code and the FDR code for a memoryless binary data source in Section 2. In Section 3, we present the modified test architecture for on-chip pattern decompression. Experimental results for the large ISCAS-89 benchmark circuits and two industrial circuits from IBM are presented in Section 4. We also present entropy bounds and show that the FDR code provides almost as much compression as the entropy bounds for the benchmarks and for the industrial circuits.

2 Analysis for a memoryless data source

In this section, we analyze the Golomb code, the conventional run-length code and the FDR codes for a memoryless data source that produces 0s and 1s with probabilities p and $(1 - p)$, respectively. The purpose of this analysis is to examine the fundamental limits of the three codes, and to demonstrate the effectiveness of the FDR code for all values of p, $0 < p < 1$. The entropy $H(p)$ of the data generated by this memoryless source is given by the following equation [8]:

$$H(p) = -p \log_2 p - (1 - p) \log_2(1 - p).$$

We first analyze the Golomb code with parameter m. This is necessary to determine a baseline for evaluating FDR codes. (The reader is referred to [4] for a review of Golomb codes. Figure 1 presents an overview of the Golomb code for $m = 4$.) The smallest and longest run-lengths that belong to group A_k are $(k-1)m$ and $(km-1)$, respectively. Therefore, the probability that an arbitrarily-
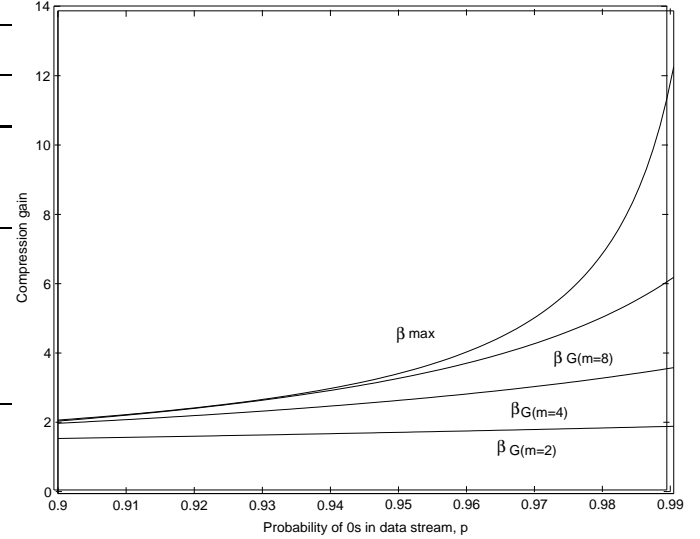

Figure 2. Compression gain for the Golomb code.

chosen run of length i belongs to group A_k is given by:

$$P(i, k) = \sum_{i=(k-1)m}^{(km-1)} p^i(1 - p) = (1 - p^m)p^{(k-1)m}.$$

The codewords in group A_k consist of $(\log_2 m + k)$ bits each [4]. Therefore, the average codeword length $\bar{G}$ for the Golomb code is given by:

$$\bar{G} = \sum_{k=1}^{\infty}(1 - p^m)p^{(k-1)m}(\log_2 m + k)$$
$$= \log_2 m + 1/(1 - p^m).$$

We next determine λ, the average number of bits in any run generated by the data source. It can be easily shown that:

$$\lambda = 1 + \sum_{i=1}^{\infty} ip^i(1 - p) = \frac{1}{1 - p}.$$

The effectiveness of compression is measured by the compression gain β_G, which is defined as the ratio of the average number of bits in any run to the average codeword size, i.e., $\beta_G = \frac{\lambda}{\bar{G}}$. This yields

$$\beta_G = \frac{1}{(1 - p)(\log_2 m + \frac{1}{1-p^m})}.$$

For example, for $m = 4$ and $p = 0.95$, $\beta_G = 2.7059$.

An upper bound on the compression gain is obtained from the entropy $H(p)$ of the source using the equation $\beta_{max} = 1/H(p)$. For example, $\beta_{max} = 3.4916$ for $p = 0.95$. Figure 2 shows the relationship between β_G and p for three values of m. The upper bound β_{max} is also shown in the figure. We note that while the compression gain for the Golomb code is significant, especially for large values of p, there is a significant difference between β_G and the upper bound β_{max}. This motivates the need for the FDR code.

We next analyze the effectiveness of conventional run-length codes for a memoryless data source. Let group A_k
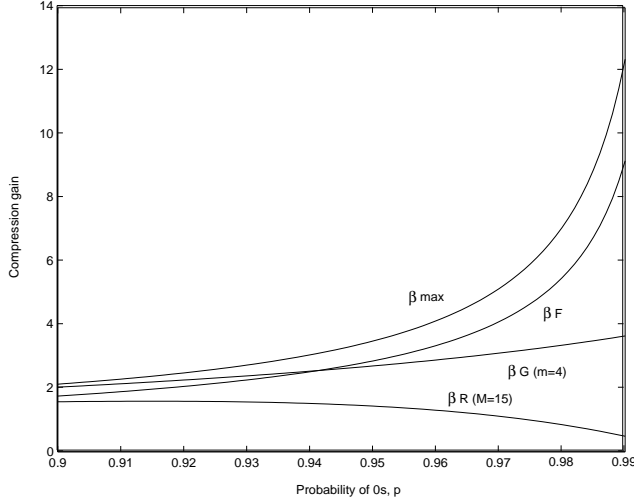
Figure 3. Compression gain for the Golomb code, the run-length code and the FDR code for $0.9 \leq p \leq 0.99$.

for run-length codes contain $(M + 1)$ members such that $M = 2^N - 1$ for some positive number N. The parameter M must be kept small, e.g. $M = 15$, in order to keep the decoder simple. The smallest and the longest run-length that belong to group A_k are given by $M(k-1)$ and $Mk-1$, respectively. Therefore, the probability that an arbitrarily-chosen run of length i belongs to group A_k is given by:

$$P(i,k) = \sum_{i=(k-1)M}^{(kM-1)} p^i(1-p) + p^{kM} = \frac{p^{kM}}{p^M}.$$

The codewords in group A_k consist of $k \log_2(M + 1)$ bits. Therefore, the average codeword length $\bar{R}$ for run-length codes is given by:

$$\bar{R} = \sum_{k=1}^{\infty} \frac{p^{kM}}{p^M} k \log_2(M+1) = \frac{\log_2(M+1)}{(1-p^M)^2}.$$

The compression gain β_R for run-length codes is given by:

$$\beta_R = \frac{(1-p^M)^2}{(1-p)\log_2(M+1)}.$$

For $p = 0.95$ and $M = 15$, we get $\beta_R = 1.4403$.

Finally, we analyze the effectiveness of FDR codes for a memoryless data source. Figure 1 presents the FDR coding procedure (details are discussed in [6]). The smallest and longest run-lengths that belong to group A_k are $(2^k - 2)$ and $(2^{k+1} - 3)$, respectively. Therefore, the probability $P(i,k)$ that an arbitrarily-chosen run of length i belongs to group A_k is given by:

$$P(i,k) = \sum_{i=2^k-2}^{2^{k+1}-3} p^i(1-p) = p^{2^k-2}(1-p^{2^k}).$$

The codeword in group A_k consists of $2k$ bits. Therefore, the average codeword length $\bar{F}$ for FDR codes is given by:

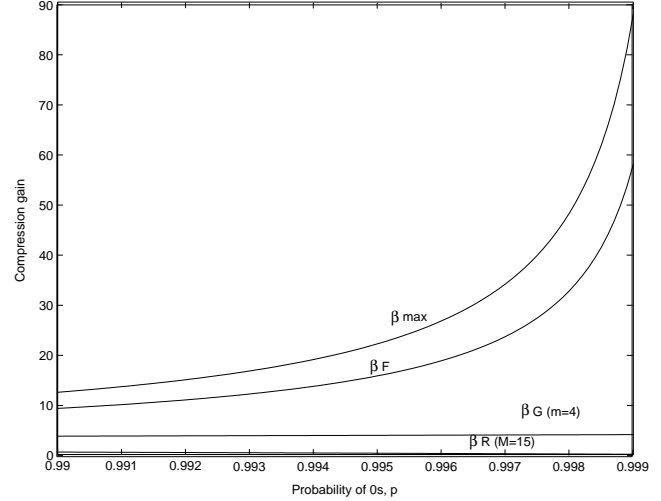$$\bar{F} = \sum_{k=1}^{\infty} 2kp^{2^k-2}(1-p^{2^k}) = 2\sum_{k=1}^{\infty} p^{2^k-2}.$$

Figure 4. Compression gain for the Golomb code, the run-length codes and the FDR code for $0.99 \leq p \leq 0.999$.

Even though we do not have a closed-form expression for $\bar{F}$, the series summation converges when k exceeds 100. Hence the above equation can be used to evaluate the effectiveness of FDR codes. The compression gain β_F for FDR codes is given by

$$\beta_F = \frac{1}{2(1-p)\sum_{k=1}^{\infty} p^{2^k-2}}.$$

For $p = 0.95$, we have $\beta_F = 2.9559$. Figure 3 shows a comparison between β_{max} and the compression gains β_F, β_G and β_R. The upper bound β_{max} is also shown in the figure. We note that compression gain for FDR codes is always higher than that for Golomb codes for $p > 0.942$. Figure 4 shows that for large values of p, there is a significant difference between β_F and β_G. The figures also show how closely the FDR gain curve follows the upper bound β_{max}. Finally, it can be easily shown that the average codeword size for FDR codes is less than for that for Golomb codes. Hence these results show that FDR codes are inherently superior to Golomb codes and run-length codes, and they allow us to approach fundamental entropy limits.

3 Improved test data compression

In this section, we present an enhancement to the basic on-chip decompression architecture of [4] to increase the amount of compression achieved with variable-to-variable-length codes. We first present a theoretical basis for the proposed scheme and then present the architecture for achieving higher compression.

It was shown in [4] that the compression achieved is directly proportional to the number of 1s in the test set. Hence, for the test cubes that have more 1s than 0s, inverting the 0s to 1 (and vice versa) is expected to yield higher compression. Let T_D (T_D') be the set of original (inverted) test cubes. T_D' is obtained from T_D by flipping all the 1s to 0 and all the 0s to 1. Let r_1 and r_0 be the total number of 1s and 0s in T_D, respectively. If $r_0 < r_1$, we expect T_D' to
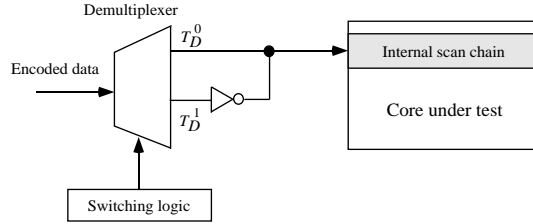
Figure 5. Application of T_D^0 and T_D^1.

yield a higher compression using either the Golomb or the FDR code. Note that all the don't cares are mapped to 0 in T_D and T_D'. If T_D' is used for compression instead of T_D, an inverter must be placed between the internal scan chain and the decoder.

Upper and lower bounds on the amount of compression achieved using Golomb codes were presented in [4]. A lower bound on the number of bits in the encoded test set for T_D with n bits and r_1 1s is given by: $L_{T_D} = n/m + r_1 \log_2 m$, where m is the Golomb code parameter. Similarly, an upper bound on the number of bits in the encoded test set for T_D' with n bits and r_0 1s is given by: $U_{T_D'} = n/m + r_0 \log_2 m + r_0(1 - 1/m)$.

It is obvious that T_D' will result in higher compression compared to T_D if $U_{T_D'} < L_{T_D}$, which implies that $n/m + r_0 \log_2 m + r_0(1 - 1/m) < n/m + r_1 \log_2 m$. This yields $r_1/r_0 > 1 + 1/\log_2 m(1 - 1/m)$, where the right-hand side of the inequality serves as a lower bound on the ratio r_1/r_0. This ratio can be used to determine whether T_D' should be used for compression. A high value of r_1/r_0 is sufficient to ensure greater compression with T_D'; however this not a necessary condition.

An alternative approach is to selectively invert only those patterns that have more specified 1s than 0s. T_D is now divided into two parts—T_D^0 and T_D^1. For all the patterns in T_D^0, the number of 0s is greater than number of 1s and for all the patterns in T_D^1, the number of 1s is greater than number of 0s. T_D^0 is encoded directly but T_D^1 is encoded after complementation. Thus T_D^0 and T_D^1 are encoded separately and this provides higher compression than that obtained if T_D is encoded directly. Pattern application is also carried out in two stages. First T_D^0 is applied to the core under test and than T_D^1 is applied to it. A small amount of logic is required to switch the pattern application from T_D^0 to T_D^1. The "switch" signal can either be provided by the ATE or by a small on-chip finite-state machine. Figure 5 shows a test architecture for the proposed scheme. Our experiments show that the proposed test architecture based on selective inversion of patterns provides significant improvements over the amount of compression obtained using simple test pattern decompression.

4 Experimental results

We evaluate the proposed compression techniques for the ISCAS-89 benchmarks and two representative industrial circuits. The experiments were conducted on a Sun Ultra 10 workstation with a 333 MHz processor and 256 MB of memory. For ISCAS-89 benchmarks, we considered only the large full-scan circuits with a single scan chain each. The test vectors for these circuits were reordered to increase compression.

Table 1 presents the experimental results for test cubes T_D obtained from the Mintest ATPG program with dynamic compaction [11]. The table lists the sizes of the pre-computed (uncompacted) test sets, the sizes of the encoded test sets obtained using the difference vector test sets derived from T_D and T_D' and the percentage compression for Golomb and FDR codes. We note that the T_D' yields better compression in all cases. This increase in compression is obtained at negligible additional cost since only an inverter needs to be added to the decompression circuit.

We now present compression results when test pattern compression/decompression is performed in two stages. Table 2 shows the results for the scheme based on selective inversion of patterns. We note that there is significant increase in compression in almost all cases. For example, there is 24.51% increase in compression for s38417 using the FDR code. The results show that the proposed two-stage compression/decompression scheme makes test data compression even more attractive for TRP.

Next, we compare the experimental results to the theoretical upper bounds on the compression predicted by the "entropy" of the test data. Let S be a data sequence with patterns $s_1, s_2, s_3, \ldots, s_k$, and let $p_1, p_2, p_3, \ldots, p_k$ be the relative frequencies of the patterns in S, respectively. An entropy measure of S is given by[1]: $E(S) = \sum_{i=1}^{k} p_i \log_2(1/p)$. Intuitively, $E(S)$ provides a lower bound on the average number of bits required to encode each pattern in S [9]. If b is the sum of the relative frequencies of $s_1, s_2, s_3, \ldots, s_k$, a lower bound on the encoded data stream for S is given by $bE(S)$.

Table 3 compares the entropy bounds to the compression obtained with the FDR code. We find that in almost all cases, the percentage compression obtained is very close to the entropy bound (less than 3% in all cases).

We next present experimental results for two real test sets from industry. The test set for the first circuit (CKT1) from IBM consists 32 statically-compacted scan vectors (a total of 362922 bits of test data per vector). This microprocessor design consists of 3.6 million gates and 726000 latches. The compression results using the Golomb and the FDR code, and the entropy bounds for the 32 scan vectors are shown in Table 4. Note that we obtain a staggering 97.10% compression on average. Table 4 also shows the entropy bounds for the test vectors. The difference between the entropy-based lower bound on the size of the encoded data and the size of

---

[1]An explicit distinction is being made here between the formal notion of entropy for a probabilistic data source, and the entropy measure for a deterministic test set with relative frequencies of test patterns.

Circuit	Size of T_D (bits)	Golomb code				FDR code			
		Based on T_D (bits)	Based on T_D (%)	Based on T_D' (bits)	Based on T_D' (%)	Based on T_D (bits)	Based on T_D (%)	Based on T_D' (bits)	Based on T_D' (%)
s5378	23754	10989	53.73	10881	54.19	9188	61.32	9200	61.26
s9234	39273	15767	59.85	15357	60.89	15460	60.63	15324	60.98
s13207	165200	25873	84.33	24268	85.30	20368	87.67	20284	87.72
s15850	76986	25748	66.55	23835	69.03	21590	71.95	20904	72.84
s38417	164736	69047	58.08	67889	58.78	57066	65.35	56634	65.62
s38584	199104	80404	59.61	75058	62.30	70328	64.67	67814	65.94

Table 1. Compression obtained using Golomb and FDR code for difference vector sequences derived using T_D and T_D'.

Circuit	Golomb code			FDR code		
	Percentage compression (based on T_D only)	Two stage compression		Percentage compression (based on T_D only)	Two stage compression	
		Size of T_E (bits)	Percentage compression		Size of T_E (bits)	Percentage compression
s5378	53.73	5330	77.56	61.32	2942	87.61
s9234	59.85	14231	63.76	60.63	12613	67.86
s13207	84.33	25217	84.73	87.67	18118	89.03
s15850	66.55	25904	66.35	71.95	20938	72.80
s38417	58.08	34585	79.00	65.35	16698	89.86
s38584	59.61	67546	66.07	64.67	52458	73.65

Table 2. Compression obtained for test data with T_D and selective inversion of test patterns.

Circuit	Size of T_D (bits)	Entropy bounds		FDR code		Circuit	Size of T_D (bits)	Entropy bounds		FDR code	
		Percentage compression	Size of encoded data (bits)	Percentage compression	Size of T_E (bits)			Percentage compression	Size of encoded data (bits)	Percentage compression	Size of T_E (bits)
s5378	23754	63.21	8738.4	61.32	9188	s15850	76986	73.51	20387.8	71.95	21590
s9234	39273	63.06	14505.9	60.63	15460	s38417	164736	67.88	52903.9	65.35	57066
s13207	165200	88.77	18543.9	87.67	20368	s38584	199104	65.98	67733.1	64.67	70328

Table 3. Comparison of compression predicted by entropy of test data and obtained using the FDR code.

Scan vector	Size of T_D (bits)	Size of encoded test set (bits)			Percentage compression		
		Golomb ($m = 128$)	FDR	Entropy bound	Golomb ($m = 128$)	FDR	Entropy bound
1	362922	27708	23800	20509.3	92.36	93.44	94.34
2	362922	20406	17410	14523.6	94.37	95.20	95.99
3	362922	20406	17410	14523.6	94.37	95.20	95.99
4	362922	20244	16366	13736	94.42	95.49	96.21
5	362922	18267	14802	12195.7	94.96	95.92	96.63
6	362922	14081	12152	9671.14	96.12	96.65	97.33
7	362922	14418	10432	8218.14	96.02	97.12	97.73
8	362922	14370	10058	7899.4	96.04	97.22	97.82
9	362922	12875	9718	7543.95	96.45	97.32	97.92
10	362922	10414	6866	5071.56	97.13	98.10	98.60
11	362922	9729	7622	5589.85	97.31	97.89	98.45
12	362922	11199	7908	5920.32	96.91	97.82	98.36
13	362922	8618	6240	4527.83	97.62	98.28	98.75
14	362922	8675	6380	4622.78	97.60	98.24	98.72
15	362922	8123	5326	3845.45	97.76	98.53	98.94
16	362922	6508	4000	2749.55	98.20	98.89	99.24
17	362922	7629	4982	3582.11	97.89	98.62	99.01
18	362922	7974	5996	4293.5	97.80	98.34	98.81
19	362922	7668	4842	3471.37	97.88	98.66	99.04
20	362922	6791	4256	2852.64	98.12	98.82	99.21
21	362922	8149	6022	4242.26	97.75	98.34	98.83
22	362922	7120	4892	3388.64	98.03	98.65	99.06
23	362922	6959	4266	3049.34	98.08	98.82	99.15
24	362922	5856	2836	1955.32	98.38	99.21	99.46
25	362922	5911	3330	2188.08	98.37	99.08	99.39
26	362922	6254	3994	2625.88	98.27	98.89	99.27
27	362922	6903	4258	2989.75	98.09	98.82	99.17
28	362922	9212	5220	3925.03	97.46	98.56	98.91
29	362922	6460	3982	2726.91	98.22	98.90	99.24
30	362922	5580	2386	1655.87	98.46	99.34	99.54
31	362922	5994	3418	2282.86	98.34	99.05	99.37
32	362922	6118	3208	2211.5	98.31	99.11	99.39

Table 4. Compression obtained for CKT1 from IBM.

Scan	Size of	Size of encoded test set (bits)			Percentage compression		
vector	T_D (bits)	Golomb ($m = 128$)	FDR	Entropy bound	Golomb ($m = 128$)	FDR	Entropy bound
1	1031072	82242	47998	41393.8	92.02	95.34	95.98
2	1031072	79927	49612	42938.4	92.24	95.18	95.83
3	1031072	74902	46986	40961.1	92.73	95.44	96.02
4	1031072	73013	43396	37650.4	92.91	95.79	96.34

Table 5. Compression obtained for CKT2 from IBM.

Circuit	Size of encoded test set (bytes)			Percentage compression		
	gzip	*comp-ress*	FDR code	*gzip*	*comp-ress*	FDR code
s9234	5913	6398	4910	46.19	41.77	55.31
s13207	8747	9279	20650	94.70	94.38	87.50
s15850	8135	8096	9624	89.43	89.48	87.49
s38417	10096	16833	20592	93.87	89.78	87.50
s38584	23948	21467	24888	87.97	89.21	87.50

Table 6. Comparison of compression obtained using *gzip*, *compress* and FDR code.

FDR-coded data is less than 1% in all cases.

Table 5 shows experimental results for a second microprocessor circuit (CKT2) from IBM. T_D for this consists of a set of 4 scan vectors (a total of 1031072 bits of test data per vector); this design contains 1.2 million gates and 32200 latches. Over 95% compression is obtained for the test cubes of CKT2. The compression results here are also within 1% of the entropy bounds.

Finally, we compare the compression obtained using the FDR code to the Unix file compression utilities *gzip* and *compress*. In order to carry out a fair comparison, we converted the encoded test sets obtained using the FDR code to a binary format. Table 6 shows the size of the encoded test set and the percentage compression obtained using *gzip*, *compress*, and the FDR code. We note that in almost all cases, the compression obtained using the FDR code is close to the compression obtained using the two Unix utilities. For s9234, the FDR code outperforms both *gzip* and *compress*. The *gzip* and *compress* utilities employ far more complex encoding algorithms than the FDR code. Hence it is inconceivable that they can be decoded using simple hardware techniques for TRP; the corresponding decompression utilities (*gunzip* and *uncompress*) are usually implemented in software. It is therefore particularly noteworthy that the simpler FDR code, which can be easily used for on-chip decompression, provides almost as much compression as *gzip* and *compress*.

5   Conclusion

Test data compression based on the FDR code is especially attractive for SOC test resource partitioning. Our rigorous analysis reveals that the FDR code outperforms both the conventional run-length code and the Golomb code. Moreover, the compression obtained using the FDR code is close to the entropy-based fundamental bound. We have also presented a modified test data compression/decompression architecture that provides even higher compression. The proposed test architecture requires a sim-ple redesign of the FDR decoder and the corresponding hardware overhead is kept small. We have also compared the FDR code to the standard Unix file compression utilities *gzip* and *compress*. Even though decompression for the FDR code is considerably simpler than the software-based decompression for *gzip* and *compress*, we obtain comparable compression using the FDR code for all benchmark circuits. Experimental results for the larger ISCAS-89 benchmarks and for two IBM production circuits demonstrate the effectiveness of test data compression for TRP.

Acknowledgments

References

[1] Y. Zorian, "Testing the monster chip", *IEEE Spectrum*, vol. 36, issue 7, pp. 54-70, July 1999.

[2] A. Chandra and K. Chakrabarty, "Test resource partitioning for SOCs", *IEEE Design & Test of Computers*, vol. 18, pp. 80-91, September-October 2001.

[3] A. Jas and N. A. Touba, "Test vector decompression via cyclical scan chains and its application to testing core-based designs", *Proc. International Test Conference*, pp. 458-464, 1998.

[4] A. Chandra and K. Chakrabarty, "System-on-a-chip test data compression and decompression architectures based on Golomb codes", *IEEE Trans. CAD*, vol. 20, no. 3, pp. 355-368, March 2001.

[5] A. Chandra and K. Chakrabarty, "Efficient test data compression and decompression for system-on-a-chip using internal scan chains and Golomb coding", *Proc. DATE Conf.*, pp. 145-149, 2001.

[6] A. Chandra and K. Chakrabarty, "Frequency-directed run-length (FDR) codes with application to system-on-a-chip test data compression", *Proc. IEEE VLSI Test Symposium*, pp. 42–47, 2001.

[7] S. W. Golomb, "Run-length encoding", *IEEE Transactions on Information Theory*, vol. IT-12, pp. 399-401, 1966.

[8] H. Kobayashi and L. R. Bahl, "Image data compression by predictive coding, Part I: Prediction Algorithm", *IBM Journal of Research and Development*, vol. 18, pp. 164, 1974.

[9] J. A. Storer, *Data Compression: Methods and Theory*, Computer Science Press, Rockville, MD, 1988.

[10] A. Khoche and J. Rivoir, "I/O Bandwidth Bottleneck for Test: Is it Real?", *Proc. Test Resource Partitioning Workshop*, pp. 2.3-1 - 2.3-6, 2000.

[11] I. Hamzaoglu and J. H. Patel, "Test set compaction algorithms for combinational circuits", *Proc. International Conference on CAD*, pp. 283-289, 1998.