

# Mixed RL-Huffman Encoding for Power Reduction and Data Compression in Scan Test

M. H. Tehranipour<sup>1</sup>, M. Nourani<sup>1</sup>, K. Arabi<sup>2</sup>, A. Afzali-Kusha<sup>3</sup>

<sup>1</sup>The Univ. of Texas at Dallas, Richardson, TX 75083, {mht021000,nourani}@utdallas.edu

<sup>2</sup>PMC Sierra, Inc., British Columbia, Canada, V5A4V7, karim\_arabi@pmc-sierra.com

<sup>3</sup>The Univ. of Tehran, Tehran, 14399, Iran, afzali@ut.ac.ir

**Abstract**—This paper mixes two encoding techniques to reduce test data volume, test pattern delivery time and power dissipation in scan test applications. This is achieved by using the Run-Length (RL) encoding followed by Huffman encoding. This combination is especially effective when the ratio of don't cares in a test set is high which is a common case in today's large SoCs. Our analytical analysis and the experimental results on ISCAS89 benchmarks confirm that achieving 32 to 85% compression ratio and 55 to 93% power reduction is possible for scan testable SoCs.

## I. INTRODUCTION

Design for testability (DFT) based on scan and automatic test pattern generation (ATPG) is a reliable technique to achieve high fault coverage. Unfortunately, for large circuits the number of scan cells and test patterns become very large. This growth can be especially seen in today's system-on-chips (SoCs) for which the test data volume increases which in turn increases test cost due to the rise in test time and memory requirement.

Power dissipation is another important factor in today's chip design. Power dissipation in CMOS circuits is proportional to the switching activity in the circuit. During normal operation of a circuit, a small number of flip flops change values in each clock cycle. In the case of test operation, large number of flip flops switch when test patterns are scanned into the scan chain. The switching activity increases when a test set is compacted [1]. Often, there are plenty don't cares in test pattern generated for scan which means opportunity for compression and power reduction.

The compression techniques are used to speed up ATE-SoC interaction during test. A variable-length compression coding using FDR codes presented in [2] which uses the distributions of 0's and 1's in typical test sequences. Variable-length input Huffman compression (VIHC) method [3] uses a maximum acceptable length to improve compression ratio, area overhead and test application time. Authors in [5] show a compression/decompression based on the statistical coding of fixed-length block to reduce test data. An embedded deterministic test technology for low cost test by reducing scan test data volume and scan test time is presented in [6].

An ATPG often assigns '0' or '1' to the unspecified bits randomly. There are techniques that utilize large number of don't cares in the test patterns generated by ATPG. Reducing the number of scan chain input pins fed by ATE by using an on-chip decoder is proposed in [7]. Reseeding the LFSR is another method [8] in which for every test vector one or multiple seeds for on-chip Linear Feedback Shift Register (LFSR) is found. The generated bit sequence for LFSR matches the test vector

at the specified bit positions. The size of the LFSR is significantly smaller than the scan chain. Reduction of scan test data in designs with multiple scan chains by using a combinational decoder is presented in [9]. The basic approach proposed in [10] is a fixed packet-based compression in which don't cares are filled appropriately in the test vectors such that they need not be stored in the ATE memory. Reordering test patterns is the proposed method in [11].

The excessive switching activity during scan test may cause larger peak and average power dissipation than in the normal operation. Therefore, the power constraint of the circuit requires to be considered in test mode. The authors in [12] proposed a test generation procedure that finds a set of test vectors to minimize the average power dissipated during scan testing. Applying scan test vectors generated by an ATPG is very time consuming and thus compacting test vectors is unavoidable. However, compacting the test vectors intensively increases power dissipation of the elements in the scan chain. A static compaction process to find a reduced test power dissipation is presented in [1] but it does not achieve high test data volume reduction. A mixed compression/decompression and low power test application technique is discussed in [4].

### A. Main Contribution

In this paper we present a compression/decompression technique to reduce the test data volume, test application time and scan test power. The proposed technique combines two well known Run-Length (RL) and Huffman encodings. Essentially, RL encoding performs a variable-length grouping to utilize the don't cares for: 1) minimizing bit-transition and power and 2) compressing data by sending the length of running (similar) bits. Huffman encoding further enhances the compression. The compressed test data is scanned in and decompressed by an inexpensive on-chip decoder to generate the exact test pattern which is finally scanned into the scan chain.

This paper is organized as follows. Section II describes the proposed RL-Huffman technique for power reduction and test data compression. The decompression and test time analysis are discussed in Section III. The experimental results are shown in Section IV. The paper ends with conclusion in Section V.

## II. MIXED RL-HUFFMAN ENCODING TECHNIQUE

This paper mixes two encoding techniques, called RL-Huffman, to reduce test data volume, test application time and scan test power dissipation. This is achieved in two steps. The first step applies the Run-Length (RL) to the test vector set. Specifically, it minimizes the transitions and eventually power

dissipation during test and also compresses data by grouping 0's and 1's in blocks. The second step produces more compression on top of the first step using Huffman encoding.

### A. Step 1: Applying RL Encoding

In the first step, the RL encoding is used for both power reduction and first round of data compression. In RL encoding, instead of sending the exact bits of a test vector, the length of the created blocks are sent. Hence, a test vector is partitioned into some variable length of 0-blocks or 1-blocks. Having don't care bits in the test vectors, there are different ways of creating 0-blocks and 1-blocks because of replacing the don't cares with either '0' or '1'. Therefore, replacing the don't care bits is an important issue which may cause different results of coding such as more compression and power reduction.

In this paper the basic idea in the RL encoding is careful replacement of the don't cares with '0' or '1' to find the minimum number of  $0 \rightarrow 1$  and  $1 \rightarrow 0$  transitions. Simultaneously, minimizing such transitions has also a significant impact in reducing dynamic switching activities of components to which the vectors are sent. While keeping the minimum transitions in the test patterns, don't care bits can be replaced with '0' or '1' in different ways.

We have used the power estimation relation proposed in [4] and [1]. Note that since the core responses are not known, these analytical relations only estimate the bit transitions (proportional to power consumption) due to the input test patterns traveling through the chain. Suppose  $n$  and  $m$  are the number of test patterns and the scan chain length, respectively. Assume  $T_i=(b_{i1} b_{i2} \dots b_{im})$  is the  $i$ th test pattern ( $1 \leq i \leq n$ ), where  $b_{i1}$  is the first bit scanned into the chain. The power dissipated in the scan cell elements (due to input test patterns) is estimated by counting the number of weighted transitions ( $WT$ ) in the pattern as presented in [1]. The transition weight for a bit means how many times this bit replaces with the complemented value when it is scanned into the scan chain. In [4] the authors showed an analytical formula for scanning in pattern  $T_i$ ,  $WT_i(sip)$ :

$$WT_i(sip) = \sum_{j=1}^{m-1} (m-j)(b_{ij} \oplus b_{i(j+1)}) \quad (1)$$

The RL coding in our technique builds the minimum transitions in each test vector. Minimum transition for a test vector does not necessarily mean minimum weighted transition ( $WT$ ) in that test vector.  $WT$  depends on the position of the transitions. As shown in (1),  $m-j$  is the weight for  $j$ th transition position in  $T_i$ . To minimize  $WT_i$  we need to minimize  $m-j$ . This means that if we push the transition position toward the least significant bit, the weight transition is reduced in each transition in the test vector and eventually the overall  $WT_i$ .

For example, assume that  $T_1=(b_{11}, b_{12}, \dots, b_{18})=(0xx11x0x)$  when  $m=8$  and obviously minimum number of transitions is two. Suppose that  $b_{11}$  is the first bit scanned into the scan chain. Table I shows comparison between all possible replacement of don't cares while keeping minimum number of transitions for test vector  $T_1$ . The last row in the table shows minimum  $WT$ . In this case ( $T_1=(00011100)$ ) the transitions are pushed toward the least significant bit. Any other different filling way with the

TABLE I  
COMPARING VARIOUS FILLINGS OF DON'T CARES.

$i$	$T_1=0xx11x0x$	$N_{Transition}$	$WT$
1	0xx11x01	$\geq 2$	$\geq 8$
2	00011000	2	8
3	00111000	2	9
4	00111100	2	8
5	01011x00	4	$\geq 15$
6	01111000	2	10
7	01111100	2	9
8	<b>00011100</b>	2	7

xxx1 0xx00xx 1xxxxx 0xxxxx 1xxx 0xxx0xxx 1xxx 0 1xx1xxxx1 0x0xx 1x1xx1xxx  
L4=4 L7=7 L6=6 L7=7 L4=4 L8=8 L4=4 L1=1 L9=9 L5=5 L9=9  
(a)

Final Sequence ( $L_i[v]$ ): 4[1] 7[0] 6[1] 7[0] 4[1] 8[0] 4[1] 1[0] 9[1] 5[0] 9[1]  
(b)

Figure 1. Applying the RL algorithm to a small example.

same number of transitions cause larger  $WT$ . The other rows either show more  $WT$  or indicate more transitions which eventually result in more  $WT$ .

Based on the above discussion we show another example of replacing the don't care bits with '0' or '1' to clarify our RL encoding technique (see Figure 1). As shown, the replacing process can be started from the first bit of vector sequence. The don't cares are replaced with '1' until we reach a '0'. At the next step, the don't cares are filled with '0' until the next '1'. This process continues until it replaces all don't care bits with '0' or '1'. In this example, the number of transitions is ten and the test set is partitioned into eleven blocks ( $N_b=11$ ) each of which is filled with only 1's or 0's.  $L_i$  shows the length of the blocks in the test set which is named *character*. The corresponding block lengths are shown in Figure 1(b). The format of showing the lengths is like  $L_i[v]$ , where  $L_i$  is the block length and  $[v]$  is the bit value '0' or '1' that is filled in that block. As shown, the number of distinct characters ( $N_c$ ) is seven. Figure 2 summarizes the occurrence frequency ( $f_i$ ) of these characters. In this work, we simply replace don't cares with 0/1 values to maximally enlarge a block and achieve minimum switching activities in scan chain when test application. More sophisticated approaches can be devised for other objectives such as minimizing  $N_c$  or increasing  $f_i$  values.

### B. Step 2: Applying Huffman Encoding

The proposed RL encoding not only reduces the test power dissipation but also performs the compression. To get the best compression rate we also apply Huffman coding (a variable-length encoding by nature) [13] to encode block length values (characters). The idea is to assign smaller number of bits to the codewords that occur most frequently and larger number of bits to those that occur less frequently. A Huffman code is obtained by constructing a Huffman tree. The generated Huffman code is shown in Figure 2. Clearly, the blocks with higher occurrence frequencies will get the smaller length codeword. For this example, instead of sending 64 bits, ATE sends only 64-34=30 bits. This is 53.12% overall saving in test data volume and also transfer time. The last column shows the saving of data sent into the chip. Instead of sending  $L$  bit 0's or 1's, the  $l$  bit codeword is sent. The saving for the  $i$ th codeword is  $L_i - l_i$ , where  $l_i$  is the length of the  $i$ th codeword ( $C_i$ ). The saving for sending

Characters (Li)	Occurrence Frequency (fi)	Huffman Code (Ci)	Block Length (li)	Saving (Si)
4	3	00	2	+6
9	2	010	3	+12
7	2	011	3	+8
5	1	100	3	+2
8	1	101	3	+5
6	1	110	3	+3
1	1	111	3	-2
				S=+34

Figure 2. Huffman code for the example in Figure 1.

the  $i$ th codeword with frequency  $f_i$  is  $S_i = f_i(L_i - l_i)$ . The total saving for all codewords is  $S = \sum_{i=1}^{N_c} f_i(L_i - l_i)$ . Then, the compression ratio (percentage of data reduction) for  $n$  vectors sent to a scan chain of  $m$  cells (overall  $n \cdot m$  bits) will be:

$$CR = \frac{S}{n \cdot m} = \frac{\sum_{i=1}^{N_c} f_i(L_i - l_i)}{\sum_{i=1}^{N_b} L_i}$$

In the next section we will show the positive impact of the RL encoding on minimizing bit transitions, and hence power consumptions, of the scan cells. Note that Huffman encoding provides further reduction and does not deteriorate the minimum transition property achieved by RL encoding.

### III. DECOMPRESSION

Huffman codes are prefix-free. This is an important property compared to other coding techniques used for compression and significantly simplifies the decoding process. In Huffman code, the decoder easily recognizes the end of each codeword. An on-chip decoder at the serial input of the SoC under test is used to decompress the test vectors. The decoder decodes the input codes to recognize how many 1's or 0's need to be shifted into the scan chain. The test vectors can be shifted into the scan chain with a higher rate, e.g. at the SoC clock speed. The ATE sends the compressed data and clock for synchronization to the decoder. In our method the compressed data is a stream of variable-length (Huffman) codewords corresponding to the block lengths.

#### A. FSM-Based Decoder

Several Huffman decoders have been proposed in recent years which are independent of the circuit and data set [16] [17] [?]. The drawback, however, is that these decoders are expensive. In this paper we propose using an inexpensive decoder that is test set-dependent. Dependency of decoder to the test data is not desirable in general but can be well justified by its low cost and generic architecture. Empirical results are shown in Section IV.

The block diagram of one possible implementation of decoder is shown in Figure 3. The compressed data (input codeword) come from the ATE to Huffman FSM where the input codewords are decoded. The decoded code addresses a small lookup table to find the block length  $L_i$  that indicates the number of 1's or 0's required to be shifted into the scan chain. The other outputs of the FSM are  $sel$  (to shift appropriate 0 or 1 into the scan chain),  $load$  (to enable of the counter) and  $stop$  (to stop the ATE when the counter has not finished its job while the FSM has generated a new address).

When the FSM generates address pointer (AP), it enables  $load$  and the counter starts counting. At the same time  $sel$  ('0' or '1') is shifted into the scan chain through an open buffer. The

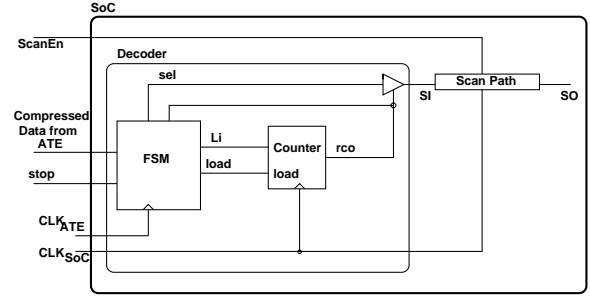


Figure 3. The decoder architecture.

FSM receives the new codeword for decoding. When the value of counter and output of the lookup table  $L_i$  become equal, output of the counter (ripple carry out (rco)) becomes 1 and shifting  $sel$  value is stopped. It also signals the FSM to put next address pointer on the input line of lookup table. If the FSM generates the address before receiving the  $rco$  from counter, it stops the ATE with  $stop$  signal and waits for  $rco$ . When it receives  $rco=1$  signal, it puts the new address on the output and deactivates the  $stop$  signal ( $stop=0$ ). Every time that a new AP is provided, a T flip-flop inside the FSM toggles to generate alternating blocks of 0's and 1's on the  $sel$  line.

#### B. Test Time Reduction

Reducing the overall test application time is the ultimate goal of test pattern compression. In general, the amount of time reduction depends on compression rate and decompression method. In this section we estimate the overall time reduction ratio (TR).

Suppose the ATE and SoC under test work with frequencies  $f_{ATE}$  and  $f_{SoC}$ , respectively. The one-cycle capture time compared to overall scan time is negligible. When there is no compression the test time is the same as transfer time, that is:

$$T_{no\_comp} = \left( \sum_{i=1}^{N_b} L_i \right) \cdot \left( \frac{1}{f_{ATE}} \right)$$

When we compress test data, such as in our method, the overall time is made of three portions:

$$T_{comp} = T_{transfer} + T_{decode} + T_{idle}$$

In our method, the transfer and decode part is quite straightforward. Essentially, the codewords ( $C_i$ ) are transferred with speed of  $f_{ATE}$  and counting toward  $L_i$  is done with speed of  $f_{SoC}$ . In other words:

$$T_{transfer} = \left( \sum_{i=1}^{N_b} L_i \right) \cdot \left( \frac{1}{f_{ATE}} \right) \quad T_{decode} = \left( \sum_{i=1}^{N_b} L_i \right) \cdot \left( \frac{1}{f_{SoC}} \right)$$

$$T_{idle} = \sum_{i=1}^{N_b} \left( \frac{L_i}{f_{SoC}} - \frac{l_i}{f_{ATE}} \right) \quad \forall i \ni \frac{L_i}{f_{SoC}} > \frac{l_i}{f_{ATE}}$$

According to the above equations, the upper bound for  $T_{comp}$  will be:

$$T_{comp} \leq \sum_{i=1}^{N_b} \left( \frac{l_i}{f_{ATE}} + 2 \cdot \frac{L_i}{f_{SoC}} \right)$$

Finally, the time-reduction ratio can be expressed as:

$$TR = \frac{T_{no\_comp} - T_{comp}}{T_{no\_comp}} \geq CR - 2 \frac{f_{ATE}}{f_{SoC}}$$

TABLE II  
COMPRESSION RATIO (CR) FOR DIFFERENT BENCHMARKS.

SoC/ Circuit	$N_{before}$	FSM	$N_{after}$	Compression Ratio (CR%)		
				RL-Huffman	[2]	[3]
PMC1	164800	391	24209	85.31	–	–
PMC2	4557	161	712	84.36	–	–
PMC3	16830	213	3519	79.09	–	–
PMC4	89154	337	17295	80.60	–	–
PMC5	13311	219	2678	79.89	–	–
s5378	23754	551	10986	<b>53.75</b>	50.77	51.52
s9234	39273	589	20582	<b>55.32</b>	44.96	54.84
s13207	165200	596	28893	82.51	80.23	<b>83.21</b>
s15850	76986	769	25143	<b>67.34</b>	65.83	60.68
s38417	164736	744	59024	<b>64.17</b>	60.55	54.51
s38584	199104	818	74863	<b>62.40</b>	61.13	56.97

TABLE III  
TEST TIME ANALYSIS

SoC/ Circuit	CR%	Time-Reduction Ratio (TR%)		
		$\frac{f_{ATE}}{f_{SoC}} = \frac{1}{20}$	$\frac{f_{ATE}}{f_{SoC}} = \frac{1}{10}$	$\frac{f_{ATE}}{f_{SoC}} = \frac{1}{5}$
s5378	53.75	46.95	40.15	26.55
s9234	55.32	51.11	44.60	31.56
s13207	82.51	75.26	68.01	53.51
s15850	67.34	60.34	53.34	39.34
s38417	64.17	57.46	50.88	37.05
s38584	62.40	55.90	49.40	36.40

It's obvious from the above formula,  $CR$  is the upper bound of the  $TR$ . For the example of Figure 1 if  $f_{ATE}=100\text{MHz}$  and  $f_{SoC}=1\text{GHz}$  and  $CR=53.12\%$  we estimate  $33.12\% \leq TR \leq 53.12\%$ . In Section IV we will show that this lower bound of  $TR$  is not tight and in practice  $CR - 1.5 \frac{f_{ATE}}{f_{SoC}} \leq TR \leq CR - 1.3 \frac{f_{ATE}}{f_{SoC}}$  for large test set.

#### IV. EXPERIMENTAL RESULTS

The proposed technique is used to compress test data for the ISCAS89 and PMC-Sierra's [14] benchmarks. Table II shows the compression results of five PMC Sierra's SoCs and six ISCAS89 benchmarks. Total number of bits before ( $N_{before}$ ) and after ( $N_{after}$ ) compression. The third column shows the overall cost of the FSM with lookup table included for the benchmarks based on the number of equivalent  $NAND$  gates reported by Synopsys synthesizer [15]. The compression ratios are obtained from both RL and Huffman encoding. Our results showed that Huffman (the second step of compression) increased 7 to 21% compression ratio on top of what RL achieves for the benchmarks. This table also compares our technique results with the best results presented in [2] (FDR coding) and [3] (VIHC method). Our RL-Huffman technique shows better or comparable results.

Table III shows the test time analysis of the ISCAS89 benchmarks for different frequencies of the system. It shows that when  $f_{SoC} \gg f_{ATE}$ , the test application time reduction ratio ( $TR$ ) is close to  $CR$ .

Table IV shows comparison between the power dissipated, due to the input test patterns, in two cases. First, we replace the don't care bits randomly for each ISCAS89 benchmark. This process is performed 50 times and the results are the average of the 50 times compilations. Second, the don't cares are replaced according to our technique. The reduction percentage for weighted transition average and peak ( $\Delta WT_{avg}$  and

TABLE IV  
COMPARING SWITCHING ACTIVITIES

Circuit	$N_{patterns}$	Power Reduction	
		$\Delta WT_{avg}\%$	$\Delta WT_{peak}\%$
s5378	23754	69.90	13.84
s9234	39273	73.76	23.47
s13207	165200	93.66	30.97
s15850	76986	85.34	31.06
s38417	164736	80.51	37.26
s38584	199104	83.77	15.95

$\Delta WT_{peak}$ ) are shown in the last two columns with respect to random filling.

#### V. CONCLUSION

We presented a new compression and decompression technique based on the Run-Length and Huffman coding for the scan testing to reduce test data volume, test application time and scan test power. The proposed technique takes advantage of the don't cares generated by ATPG. The experimental results show the compression ratio up to 85% and power reduction up to 93% for the benchmarks that we tried so far.

#### ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation CAREER Award #CCR-0130513.

#### REFERENCES

- [1] R. Sankaralingam, R. Oruganti and N. A. Touba, "Static Compaction Techniques to Control Scan Vector Power Dissipation," in Proc. *VLSI Test Symp. (VTS'00)*, pp. 35-40, 2000.
- [2] A. Chandra and K. Chakrabarty, "Reduction of SOC Test Data Volume, Scan Power and Testing Time Using Alternating Run-Length Codes," *Design Automation Conf. (DAC'02)*, pp. 673-678, 2002.
- [3] P. Gonciari, B. Al-Hashimi and N. Nicolici, "Improving Compression Ratio, Area Overhead, and Test Application Time for System-on-a-chip Test Data Compression/Decompression," *Design, Automation and Test in Europe (DATE'02)*, pp. 604-611, 2002.
- [4] A. Chandra and K. Chakrabarty, "Low-Power Scan Testing and Test Data Compression for System-on-a-Chip," *IEEE Trans. On Computer-Aided Design (TCAD)*, vol. 21, no. 5, pp. 597-604, 2002.
- [5] A. Jas, J. Ghosh-Dastidar and N. Touba, "Scan Vector Compression/Decompression Using Statistical Coding," in Proc. *VLSI Test Symp. (VTS'99)*, pp. 114-120, 1999.
- [6] J. Rajski, et. al, "Embedded Deterministic Test for Low Cost Manufacturing Test," in Proc. *Int. Test Conf. (ITC'02)*, pp. 301-310, 2002.
- [7] I. Bayraktaroglu and A. Orailoglu, "Test Volume and Application Time Reduction Through Scan Chain Concealment," in Proc. *Design Automation Conf. (DAC'01)*, pp. 151-155, 2001.
- [8] C. Krishna, A. Jas and N. Touba, "Test Vector Encoding Using Partial LFSR Reseeding," in Proc. *Int. Test Conf. (ITC'01)*, pp. 885-893, 2001.
- [9] S. Reddy, K. Miyase, S. Kajihara and I. Pomeranz, "On Test Data Volume Reduction for Multiple Scan Chain Designs," in Proc. *VLSI Test Symp. (VTS'02)*, pp. 103-108, 2002.
- [10] A. Khoche, E. Volkerink, J. Rivoir and S. Mitra "Test Vector Compression Using EDA-ATE Synergies," in Proc. *VLSI Test Symp. (VTS'02)*, pp. 97-102, 2002.
- [11] M. H. Tehranipour, N. Ahmed, M. Nourani, "Testing SoC Interconnects for Signal Integrity Using Boundary Scan," to appear in *VLSI Test Symposium (VTS'03)*, 2003.
- [12] S. Wang, S. K. Gupta, "ATPG for Heat Dissipation Minimization During Scan Testing," in Proc. *Design Automation Conf. (DAC'97)*, pp. 614-619, 1997.
- [13] D. Huffman, "A Method for the Construction of Minimum Redundancy Codes," in Proc. *IRE*, vol. 40, no. 9, pp. 1098-1101, 1952.
- [14] PMC-Sierra, www.PMC-Sierra.com, 2002.
- [15] Synopsys Inc., "User Manuals for SYNOPSIS Toolset Version 2002.05," Synopsys, Inc., 2000.
- [16] L. Chia-Hsing and J. Chein-Wei, "Low Power Parallel Huffman Decoding," *Electronics Letters*, vol. 34, no. 3, pp. 240-241, 1998.
- [17] R. Benes, S. Nowick and A. Wolf, "A Fast Asynchronous Huffman Decoder for Compressed-Code Embedded Processor," *Int. Advanced Research in Asynchronous Circuits and Systems*, pp. 43-56, 1998.