Test Cost Reduction for SoC Using a Combined Approach to Test Data Compression and Test Scheduling

Quming Zhou
Dept. of Electrical and Computer Engg.
Rice University, Houston, TX, USA
quming@rice.edu

Kedarnath J. Balakrishnan
NEC Laboratories America
4 Independence Way, Princeton, NJ, USA
bala@nec-labs.com

Abstract

A combined approach for implementing system level test compression and core test scheduling to reduce SoC test costs is proposed in this paper. A broadcast scan based test compression algorithm for parallel testing of cores with multiple scan chains is used to reduce the test data of the SoC. Unlike other test compression schemes, the proposed algorithm doesn't require specialized test generation or fault simulation and is applicable with intellectual property (IP) cores. The core testing schedule with compression enabled is decided using a generalized strip packing algorithm. The hardware architecture to implement the proposed scheme is very simple. By using the combined approach, the total test data volume and test application time of the SoC is reduced to a level comparable with the test data volume and test application time of the largest core in the SoC.

1. Introduction

Recent advances in technology have made it possible to integrate entire systems on a single chip. These systems-on-a-chip (SoC) consists of varied components (also called cores) like embedded processors, digital logic, memories, and analog/RF components, all in the same chip. With the new paradigm of core based design, intellectual property (IP) cores and design reuse, the time to market new SoCs reduces considerably. However, because of the same factors, the SoC integrator has new challenges for manufacturing test of the SoC.

With the increasing complexity and the number of cores in SoC, the volume of test data required to guarantee good manufacturing test quality also increase. In addition, the number and varied types of cores require different test methodologies for SoC designs as compared to conventional integrated circuits. Due to the heterogeneous nature of the cores, some may require BIST (memories) while others (analog cores) need to be isolated during conventional digital testing. For certain cores which are IP protected, the structural information is not available and core internals like layouts and scan configurations may be fixed. Conventional state-of-the-art techniques for test data compression like Mentor Graphics EDT [10], Synopsys DFT Compiler Maxand other EDA vendor tools utilize structural information for scan chain synthesis, fault simulation and test generation with compression. They are not applicable at the system level, especially if the SoC has an IP core. Other techniques for test data compression like selective Huffman coding [6] or dictionary coding [9] where structural information is not required have usually focussed on core level testing. The decoder for these techniques is designed based on the test patterns for each core separately. Hence, at the system level multiple decoders are necessary for testing all the cores in the SoC which leads to high area overhead as well as synchronization and timing issues.

Another important issue in SoC testing is the restricted access to internal signals of cores. Recently, the IEEE P1500 standard [1] has proposed a test architecture consisting of test wrappers for each core and test access mechanisms (TAMs). There has been considerable work in the area of TAM/wrapper design. Most of the previous works have concentrated on optimizing wrapper designs [7, 11] and core test scheduling [4, 5] to reduce the test application time for the SoC. However, these techniques haven't considered the impact of test data compression in their design considerations. The focus of this paper is on system level test of SoCs. We present a combined algorithm for implementing system level test compression and core test scheduling that is applicable to IP cores. The proposed algorithm utilizes a broadcast scan [8] based test compression method across multiple cores and a generalized strip packing algorithm to decide the core test schedule with compression enabled. We assume that the test wrapper for each core is decided by the core vendor and hence the core characteristics like number of scan chains (internal and boundary) etc cannot be modified.

It is well known that the percentage of unspecified (or don't care) bits in automated test pattern generator (ATPG) generated test patterns (without random filling) is high even for compacted tests. This high percentage of don't cares provides an opportunity to find scan chain sharing from different cores, so that the corresponding test sets can be merged and then broadcasted to multiple chains in parallel testing. By sharing scan chain inputs among several cores it is possible to reduce test data volume and shorten test application time significantly, since cores that share chains are tested concurrently. The major contribution of this paper is a scan chain sharing strategy to reduce test volume without fault simulation. Our strategy exploits don't care bits in test sets and combines test scheduling with scan chain sharing. To minimize architecture support, we do merging at scan chain level so that the structure is same for all test patterns. Another advantage of the proposed test strategy is that it does not require fault simulation to verify the fault coverage.

Parallel or concurrent testing has been proposed before for SoCs, but using ATE capabilities [2]. The broadcast scan technique [8] where a single input is fed into multiple internal scan

978-3-9810801-2-4/DATE07 © 2007 EDAA

chains is a form of concurrent testing. However, this scheme and other techniques based on broadcast scan require a special ATPG methodology to ensure high fault coverage. In this paper, we apply the concept of broadcast scan for multiple cores after the test patterns are generated and hence our technique is applicable to IP cores with precomputed test sets. [12] proposes a method of sharing test sets for multiple cores with the assumption that each core has only one scan chain. An algorithm is presented to find the best position to merge each core test set into the core test set with the longest scan chain. A scan chain disable signal generator is used to retrieve the original test set for each core from the shared test set. The paper partitions test vectors into different segments and each partition is merged separately in the shared test set which requires a complex scan chain disable generator and high hardware area overhead. On the other hand, the proposed scheme is applicable to cores with multiple scan chains and doesn't require a complicated on-chip decoder. In this paper, we present a combined algorithm for concurrent testing using test set merging and core test scheduling.

The rest of the paper is organized as follows. Section 2 discusses the background and notation used in this paper. Section 3 describes the proposed architecture and the scan chain merging algorithm. SoC test scheduling taking into account scan chain merging is presented in section 4. Experimental results are reported in section 5 and conclusions are in 6.

2. Background and Notation

The basic idea behind the broadcast scan technique [8] was to utilize a single tester channel to load multiple scan chains simultaneously. However, because of the restrictions imposed by this architecture on the values that can be shifted into the scan chains, the fault coverage reduces. Hence, a *serial mode* is required [3] during which the scan chains can be loaded with values independent of each other to test the faults that became untestable in the *parallel* or broadcast scan mode. Hence, a special ATPG is required, which can handle both modes of operation to achieve high fault coverage. The compression obtained using this technique also has to take into account both the modes of operation.

In the proposed work, we also utilize a single tester channel to load multiple scan chains. However, there are several differences from earlier techniques. We are proposing a SoC testing approach using test data compression based on broadcast scan. We assume that the test patterns of all the cores are precomputed. Two scan chains from different cores are checked whether they can be merged, in which case both of them are connected to the same tester channel. This means the cores to which these scan chain belong need to be tested simultaneously. Hence a combined algorithm is required for test compression and core test scheduling.

The following assumptions are made in this paper: a) Each core has a full scan test architecture with multiple scan chains; b) Each core is a black box where fault simulations are unavailable; c) The test set with don't care bits in test patterns is provided for each core for full scan testing; d) Test cubes (test patterns with don't care bits) in a single test set can not be further merged with each other. A test set T consists of n test cubes $P_i \in T, i = 1, ..., n$, in which unspecified or don't care bits are left during test pattern generation. Each test cubes P_i is composed of m chain vectors from m chains C_j, $j = 1, ..., m$. Each chain C_j contains in turn L_j scan cells. We refer to L_j as the length of chain C_j.

3. Proposed Scheme

The proposed test architecture is shown in Fig. 1. Several external TAM lines and an external test clock signal are sent from the tester. The scheduler encodes the test schedule information *i.e.* when each core is tested. Each core may be tested during multiple time periods. A modulo counter is required for each core that generates the scan enable signal based on the maximum scan chain length for that core. The internal TAM lines (scan chain inputs) are tapped from the external TAM lines. Hence, to implement chain sharing, the corresponding scan chain inputs should be connected to the same TAM line.

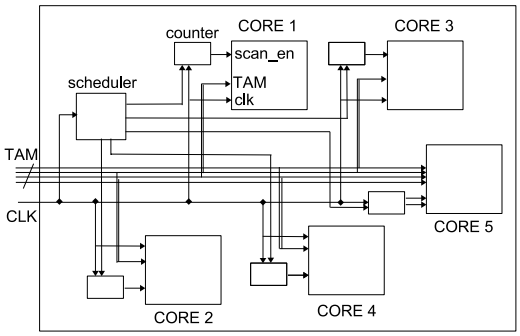

Figure 1. Test Architecture

3.1. Chain Merging

Two chain vectors v_i and v_j are compatible if they do not specify complementary values in any bit position. If any two chain vectors, say v_i and v_j are compatible, they are replaced by the new chain vector v_k that has all the binary values of both v_i and v_j as shown in Fig. 2. The length of the resultant chain vector is the maximum length of v_i and v_j. In Fig. 2, the left most bit of chain vector v_i is aligned with the third bit of chain vector v_j, and the right most bit is aligned with the last second bit of chain vector v_j. The parameter $shift\_bit$ is used to denote the shifted bits of v_i while merging with v_j. The range of $shift\_bit$ is between 0 and the length difference of two merging vectors, $shift\_bit = 2$ in Fig. 2. The opportunity of merging two vectors is increased by varying $shift\_bit$. The $shift\_bit$ can be implemented by disabling the scan shifting clock to hold the bit values in scan cells. Negative $shift\_bit$ is not allowed since it will lead to a longer merged vector than the original chain vectors.

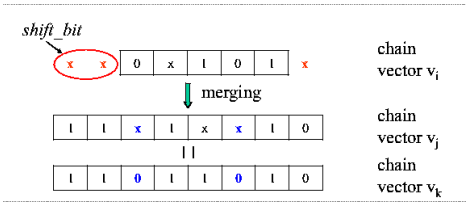

Figure 2. Merging two chain vectors

The above definition of chain vector compatibility can be extended to compatible scan chains. Two scan chains (say c_i and

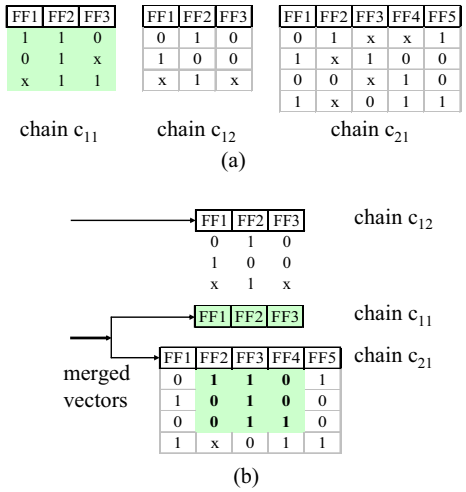

Figure 3. Merging Scan Chains

c_j) are compatible when all the chain vectors of the first chain (corresponding to every test cube) are compatible with chain vectors of the other chain. After merging, both chain c_i and chain c_j are applied with same vectors denoted by c_k. There are several parameters that affect the chances of merging between two chains. However, in this paper, we utilize two of them, $shift\_bit$ and $vector\_order$, which are easy to implement in hardware. A merged chain is assigned as an unique $shift\_bit$ (it remains constant across all test vectors) to indicate how many clocks should be disabled when it is connected together with another chain. This optimum value of $shift\_bit$ is determined from all possible values in its range using two factors. The first factor is the number of test vectors in chain c_i that can be merged into chain c_j which needs to be maximized and the second factor is the number of additional don't care bits to be specified when the vectors are merged (which needs to be minimized).

The second parameter utilizes the fact that the order of test vectors is irrelevant for most fault models (stuck-at etc). Without any loss of generality, we assume that the number of test vectors corresponding to the core to which the merging chain belongs to is not greater than the number of test vectors corresponding to the core of the target chain. In the proposed scheme, we use a simple heuristic to decide the vector order. The test vectors of the target core are sorted in ascending order of specified bits with the least specified test vectors in the beginning and the most specified in the end. The test vectors of the merging core are sorted in the opposite direction. This sorting allows test cubes with fewest don't cares in the merging core to be merged with test cubes with most don't cares in the target core.

Fig. 3 illustrates an example of sharing chains across two cores. Core 1 has two chains, chains c_{11} and c_{12}, and three test vectors (one on each row), and core 2 has one chain c_{21} and four vectors as shown in Fig. 3(a). The test set of chain c_{11} can be merged with c_{21}, resulting in the merged test set in Fig. 3(b). In this example, $shift\_bit$ is 1 for both c_{11} and c_{12}, and the test volume is reduced from 38 bits to 29 bits.

The above definition for sharing chains from different cores has the requirement that all the patterns in one core (the merging core) should be mergeable with the patterns of the second core (the target core) which is a hard constraint to satisfy. This constraint can be removed if the core testing is done in two different modes. First, the combined test cubes for both the cores are applied in parallel or concurrently. The remaining unmergeable test cubes for each core are applied separately in serial mode. We call this as partial compatibility, and two chains are partially compatible if the number of unmergeable cubes is not more than a threshold. The threshold is decided by the number of test patterns and the number of scan chains in the core.

3.2. Sharing Chains between Cores

The previous subsection described merging between two chains. In this subsection, we develop our algorithm for sharing scan chains between two different cores, each with multiple chains. A graph model is used to represent the compatibility between scan chains. Let $G(V, E)$ be a graph with a set of vertices V and a set of edges E where each vertex represents a scan chain. An edge exists between two vertices if and only if the corresponding scan chains are mergeable. Let us consider merging chains between two cores, $Core_I$ and $Core_{II}$. Since we assume that the scan chains within a core cannot be merged with each other, the compatibility graph is basically a bipartite graph. The graph vertices can be decomposed into two disjoint sets containing the vertices corresponding to the scan chains of each core with the property that no two vertices within the same set are adjacent.

Consider the following coloring scheme for the vertices of the graph. Each vertex in the graph is initially assigned an indistinct color. If there is an edge between two vertices, *i.e.* the chains corresponding to the two vertices can be merged together, then they can have the same color. The problem of maximum chain sharing amongst the two cores is similar to minimum coloring of the graph. In fact, this problem can be mapped into the minimum coloring problem of a dual graph which is NP-Hard. The final number of colors should be less than the available ATE channels for the two cores to be mergeable. Note that merging one chain into another chain will specify the don't care bits of vectors in chains, and this may invalidate other edges of the vertices. Since the problem is NP-Hard, a greedy heuristic is used to find a solution. The idea is to maximize the number of remaining graph edges after each chain merging since each chain merging will cut off edges from the graph.

1. Build a compatibility graph G by checking the compatibility of each scan chain in $Core_I$ with all chains in $Core_{II}$. Each edge is assigned an edge distance that indicates the number of don't care bits specified during the merging. Each vertex in $Core_{II}$ is assigned a distinct color.
2. Build a list L_I for vertices in $Core_I$.
3. Sort the list in ascending order of vertex degrees.
4. Process a vertex with the minimum degree in L_I, say V_i^I. If its degree is zero, assign V_i^I a new color, remove it from graph G and list L_I, and go to step 8.
5. Find all vertices in $Core_{II}$ that are connected with vertex V_i^I. Choose the vertex that has the minimum degree among them. If the degrees are same, pick the vertex that has the least edge distance. Lets name this vertex V_j^{II}, chosen to merge with V_i^I.
6. Merge V_i^I with V_j^{II}. Assign V_i^I with the same color as V_j^{II}, delete all edges associated with V_i^I, and remove V_i^I from graph G and list L_I.

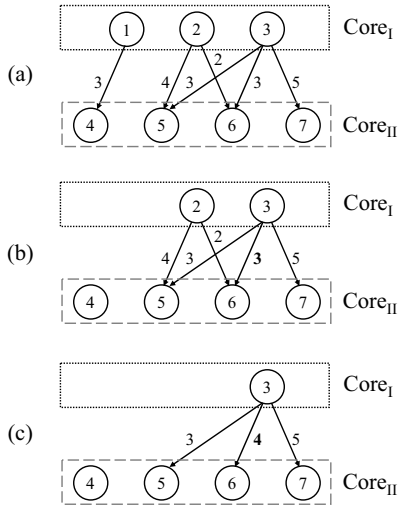

Figure 4. Selecting Chains to Merge

7. Update edges associated with V_j^{II} in graph G. Some edges may not exist and some edge distances may be updated as well.
8. Repeat step 3, until L_I is empty.

The example in Fig. 4 is used to illustrate the above proposed merging procedure. $Core_I$ has three scan chains and $Core_{II}$ has five scan chains represented as vertices in the bipartite graph. Assume the compatibility edges and the edge distances as shown in Fig. 4(a). Since vertex 1 of $Core_I$ has the smallest degree, it is chosen as the first candidate. It can only be merged with vertex corresponding to chain 4 of $Core_{II}$ and the graph after this merging is shown in Fig. 4(b). The next candidate is vertex 2 of $Core_I$ which can be either merged with vertex 5 or vertex 6 of $Core_{II}$, both with degree two. However, since the edge distance from vertex 2 of $Core_I$ to vertex 6 of $Core_{II}$ is smaller than the edge distance between vertex 2 of $Core_I$ and vertex 5 of $Core_{II}$, chain 2 of $Core_I$ is merged with chain 6 of $Core_{II}$. Assume that after merging with chain 2 of $Core_I$, chain 6 of $Core_{II}$ has a new edge distance from chain 3 of $Core_I$ as shown in Fig. 4(c). Therefore, the last vertex of $Core_I$ which corresponds to chain 3 is merged with vertex 5 of $Core_{II}$ due to the smallest corresponding edge distance. As a result of this chain sharing algorithm, the seven scan chains of the two cores are reduced to four chains, each corresponding to one chain of $Core_{II}$.

4. Test Scheduling

The algorithm presented in the previous section can be used to determine the merging between scan chains of two cores. However, current SoCs contain several cores and determining the optimal order of the cores for merging is necessary to obtain the highest data compression and lowest test application time. Even when chain sharing is not used, test scheduling of the different cores is required to ensure that the limited number of ATE channels are utilized optimally thereby to reduce the total test application time.

Previous work on chain sharing between cores [12] assumed a single scan chain for each core and hence didn't consider the requirements imposed by the limited number of ATE channels etc. Such a constraint is obvious from the following example. Suppose there are three cores in a SoC such that core 1 has 2 scan chains, core 2 has 4 scan chains and core 3 has 3 scan chains for a total of 9 scan chains. Further, assume the scan chain merging compatibility is such that both core 1 and core 2 have one chain that can be merged with scan chains of core 3. To test all the three cores in parallel using chain sharing, the number of ATE channels should be atleast 7. However, if the number of ATE channels is 4, only core 1 and core 3 can be tested in parallel by sharing one chain. Core 2 has to be tested separately after testing the other two cores.

We formulate the test scheduling as a generalized strip packing problem, and then present a solution in conjunction with scan chain sharing. For test scheduling, each core can be considered as a rectangle, with the width representing the number of scan chains in the core and the height representing the test time for all the test patterns of the core [5]. Consider the two-dimensional strip packing problem which can described as the following. Given a set of rectangles and an open-end bin of fixed width, pack the rectangles into the bin in a way that minimizes the overall height of the bin. All rectangles must be packed with their width parallel to the bottom of the bin (they cannot be rotated). For the test scheduling problem, width of the bin corresponds to the available number of ATE channels, and the overall height of the bin is the time required to test all the cores in the SoC. Hence, the aim is to schedule the tests so that they are completed in the least possible time. Finding an optimal solution for the strip packing problem is NP-complete, so we use a level-oriented heuristic.

Initially, all the rectangles are pre-sorted in order of decreasing height and the sorted rectangles are packed to the lowest level that they can fit. The first level is the bottom of the bin and subsequent levels are defined by the height of the tallest rectangle of the previous level. Testing for cores at the same level begins at the same instant and finishes no later than the instant denoted by the height of the level. The level-oriented algorithm allows cores at the same level to be tested in parallel and to share the same output response analyzer. Furthermore, the architecture of sharing ATE channels amongst multiple scan chains is simplified since all tests in a level are synchronized. The dynamic test scheduling algorithm with scan chain sharing is described below.

1. Sort all rectangles (cores) in decreasing order of the height and process in this order.
2. Add the current rectangle into the lowest level that it can fit and shift it to the left most. Each rectangle is adjacent to the rectangle just processed or to the left wall of the bin.
3. Start a new level if the current rectangle can not fit the previous level *i.e.*, the width goes beyond the available TAM width. The height of a level is determined by the first rectangle in the level. All the subsequent rectangles in the level will have smaller heights.
4. To check whether a rectangle will fit into the current level, explore all possible chain sharing between the scan chains of the current core with all other cores in the current level. If scan chain sharing is possible, the current rectangle will have common area with other rectangles and more free space available in the current level for other rectangles.
5. Loop back to step 2 until all rectangles are packed into the bin.

The example in Fig. 5 is used to illustrate the test scheduling algorithm and show the chain sharing reduces the total test volume as well as total test application time. Fig. 5 (a) shows the

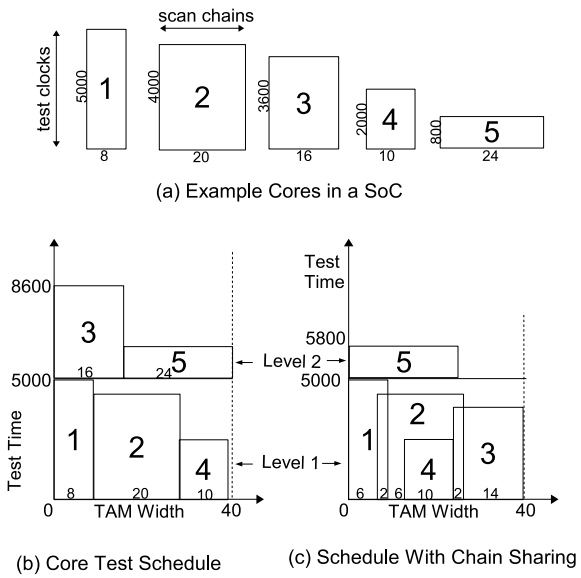

Figure 5. Test Scheduling With Chain Sharing

characteristics of the different cores arranged in descending order of height (testing time for all the test patterns). The width of each core represents the number of scan chains. Suppose the number of ATE channels available (TAM Width) is 40. Fig. 5 (b) shows the core test schedule without any merging. The first two cores will be packed into the first level since the required width $(8 + 20 = 28)$ is less than the available TAM width. However, core 3 has 16 scan chains and cannot be packed into the first level. Hence it is placed at level 2, above core 1 which has the most height in level 1. Core 4 has only 10 scan chains and can be accommodated easily in level 1. Core 5 is added to level 2. The total test time for this schedule is equal to the sum of testing times for core 1 and core 3, since they represent the height of each level. Hence $5000 + 3600 = 8600$ clock cycles are required, as shown in Fig. 5 (b).

The results of test scheduling with scan chain sharing is illustrated in Fig. 5 (c). The procedure starts by packing core 1 as the first core in level 1. When core 2 is added to level 1, possible chain sharing is explored between the two cores. As a result, two chains can be shared between core 1 and core 2, shown as the portion common to rectangles 1 and 2 in Fig. 5 (c). Core 3 has 16 scan chains and cannot be directly added to level 1, since the required TAM width will exceed 40. However, while checking scan chain compatibility, two chains between cores 2 and 3 can be shared. With this sharing, core 3 can be accommodated in level 1. If the scan chain compatibilities of cores 2 and 4 are such that all the scan chains of core 4 can be shared with core 2, the test schedule rectangle for core 4 will be fully contained in the rectangle for core 2. Core 5 is then added to the next level. The total test time for this schedule is $5000 + 800 = 5800$ as shown in Fig. 5 (c).

5. Experimental Results

Experiments to evaluate the proposed technique were performed on three industrial SoC designs. Test patterns for the stuck-at faults of each core (assuming full scan) were generated using a proprietary ATPG tool with static and dynamic compaction but no random filling of unspecified bits (Xs). The results are shown in Table 1. The first five rows of Table 1 lists the number of cores, number of scan chain, the total number of flip-flops, the total test volume (in bits) and the total test application time for each SoC. The total test volume is calculated as the sum of test data volume of each core. Similarly, total test application time shown in Table 1 is just the sum of test application times of all the cores, *i.e.,* assuming all the cores are tested one by one serially. The next five rows of Table 1 show the characteristic of the core with the largest test data volume. The number of flip-flops, number of chains, data volume and test application time of the largest core in each SoC are listed. "Vol. Ratio" shows the ratio of the test data volume of the largest core with the total test data volume of the SoC. Note that since the proposed scheme doesn't compress the data of each core separately, the test data volume of the largest core is the limit to which total test data volume of the SoC can be reduced. For the same reason, test time of the largest core will be the minimum possible test time for the SoC.

Table 1. SoC Results

		SoC 1	SoC 2	SoC 3
Total	No. Cores	7	5	22
	No. Chains	76	123	266
	Scan FF	6988	20241	264226
	Data Vol.	4215616	10994900	354526708
	Test Time	326096	382795	20685808
Max.	Scan FF	1744	6800	50937
	No. Chains	16	34	48
	Data Vol.	1784112	6337600	93489984
	Vol. Ratio	42.3%	57.6%	26.3%
	Test Time	112530	187332	3124157
Test Sch.	TAM Width	22	35	80
	Test Time	255512	382795	6473275
Prop.	TAM Width	22	35	80
	Data Vol.	2598030	6920800	212109250
	Vol. Ratio	61.6%	62.9%	59.8%
	Test Time	135168	214552	3499835
	Time Ratio	52.9 %	56.0 %	54.0 %
	Runtime (s)	2.4	29.2	611.6

Table 2. Details of Cores in SoC1

Core	Chains	Length	Patterns	Volume	Time
Dct	8	48	607	233088	29136
Idct	8	84	185	124320	15540
Ispq	16	109	1023	1784120	111500
Mc	12	97	414	481896	40158
Mv	12	97	332	386448	32204
Vld	16	97	713	1106576	69161
Rbit	4	77	322	99176	24794

The two rows of Table 1 under "Test Sch." report the test application time using test scheduling (without scan chain sharing) for the given ATE channels (reported as "TAM Width"). The test data volume, test application time for the TAM Width using the

Table 3. SoC1 Test Schedule for TAM Width 22

Level	With Chain Sharing				Without Sharing			
	Start	End	TAM	Cores	Start	End	TAM	Cores
Level 1	0	112530	22	ALL	0	112530	20	Ispq,Rbit
Level 2	112530	124486	20	Vld,Rbit	112530	182404	16	Vld
Level 3	124486	131150	20	Mv,Dct	182404	222976	20	Mc,Dct
Level 4	131150	**135168**	20	Mc,Idct	222976	**255512**	20	Mv,Idct

proposed scheme are shown in the last five rows of Table 1. The row "Vol. Ratio" reports the ratio of the new test data volume with the total test data volume. The row "Time Ratio" reports the ratio of the test application time using the proposed scheme with the test application time using normal test scheduling (without scan chain sharing). As can be seen from the table, proposed scheme achieves very good test data volume and test application time reduction. An important point to note from the results in Table 1 is that the test data volume and test application time of the SoC using the proposed scheme is comparable to that of the largest core in the SoC. In the case of SoC 2, where the largest core dominates the SoC, the total test data volume and test application time is only 10% more than that of the largest core. For the other two SoCs, there are other cores which are similar in size to the largest core and hence the results are not that spectacular. However, the results are still significantly less than the original test data volume and test application time. The proposed algorithm is very fast; the runtime for the largest SoC is around 11 minutes.

The results are explained in more detail using SoC 1 as an example. SoC1 a media-processing SoC with seven cores. Table 2 lists the details of each core. The core **Ispq** is the largest core with 42.3% of the total test data volume and with the maximum number of test patterns. The schedule for SoC 1 with and without scan chain sharing for TAM Width 22 is shown in Table 3. Since the proposed algorithm is level oriented, the results are listed level wise with each subsequent level starting after testing for all the cores in the previous level is finished. In Table 3, the columns "Start" and "End" represent the start and end clock cycles of each level, while the column "TAM" shows the number of TAM lines used in that level. The column "Cores" lists the cores that are tested in that level. Both techniques have the test schedule with four levels. The schedule without chain sharing is able to utilize only 20 TAM lines at the maximum. When scan chain sharing is enabled, initially all cores are tested concurrently (Level 1). The remaining test patterns of each core are tested in the higher levels without any sharing. Even though there are 22 TAM lines available, only 20 are used in levels 2-4. The test schedule finishes at 135168 cycles, which is 52.9% of finish time without any chain sharing.

6. Conclusions

In this paper, we proposed a combined approach for implementing system level test compression and core test scheduling to reduce SoC test costs. A broadcast scan based test compression algorithm for parallel testing of cores with multiple scan chains is used to reduce the test data of the SoC. Unlike other test compression schemes, the proposed algorithm doesn't require specialized test generation or fault simulation and is applicable with intellectual property (IP) cores. A generalized strip packing based core testing schedule algorithm with compression is presented. The hardware architecture to implement the proposed scheme is very simple. By using the combined approach, very good test data volume and test application time reduction can be obtained. Experiments using the proposed scheme on three industrial SoCs are reported. Experimental results show that the test data volume and test application time of the SoC using the proposed scheme is comparable with the test data volume and test application time of the largest core in the SoC.

References

[1] F. DaSilva, Y. Zorian, and et. al. Overview of the IEEE P1500 Standard. In *Proc. Intl. Test Conference*, pages 988–997, 2003.

[2] R. Dorsch, R. H. Rivera, and et. al. Adapting an SOC to ATE Concurrent Test Capabilities. In *Proc. Intl. Test Conference*, pages 1169–1175, 2002.

[3] I. Hamzaoglu and J. H. Patel. Reducing Test Application Time for Full Scan Embedded Cores. In *Proc. of Int. Symposium on Fault Tolerant Computing*, pages 260–267, 1999.

[4] Y. Huang, S. M. Reddy, and et. al. Optimal Core Wrapper Width Selection and SOC Test Scheduling Based On 3-D Bin Packing Algorithm. In *Proc. Intl. Test Conference*, pages 74–82, 2002.

[5] V. Iyengar, K. Chakrabarty, and E. J. Marinissen. On Using Rectangle Packing for SOC Wrapper/TAM Co-optimization. In *Proc. VLSI Test Symposium*, pages 253–258, 2002.

[6] A. Jas, J. G. Dastidar, M.-E. Ng, and N. Touba. An Efficient Test Vector Compression Scheme Using Selective Huffman Coding. *IEEE Trans. Computer-aided Design*, 22(6):797–806, 2003.

[7] S. Koranne. Design of Reconfigurable Access Wrappers for Embedded Core Based SoC Test. *IEEE Trans. VLSI Systems*, 11(5):955–960, 2003.

[8] K.-J. Lee, J. Chen, and C. Huang. Using a Single Input to Support Multiple Scan Chains. In *Proc. International Conference on Computer Aided Design*, pages 74–78, 1998.

[9] L. Li, K. Charabarty, and N. A. Touba. Test Data Compression using Dictionaries with Selective Entries and Fixed-Length Indices. *ACM Transactions on Design Automation of Electronic Systems*, 8(4):470–490, Oct. 2003.

[10] J. Rajski and et. al. Embedded Deterministic Test. *IEEE Trans. Computer-aided Design*, 23(5), May 2004.

[11] A. Sehgal, S. K. Goel, E. J. Marinissen, and K. Chakrabarty. IEEE P1500-Compliant Test Wrapper Design for Hierarchical Cores. In *Proc. Intl. Test Conference*, pages 1203–1212, 2004.

[12] G. Zeng and H. Ito. Concurrent Core Test for SOC using Shared Test Set and Scan Chain Disable. In *Proc. Design Automation and Test in Europe*, pages 1045–1050, 2006.