Constraint Driven Pin Mapping for Concurrent SOC Testing

Yu Huang[1] Nilanjan Mukherjee[2] Chien-Chung Tsai[2] Omer Samman[2]
Yahya Zaidan[2] Yanping Zhang[2] Wu-Tung Cheng[2] Sudhakar M. Reddy[1]

[1] Department of Electrical & Computer Engineering, University of Iowa, Iowa City, IA 52242
[2] Mentor Graphics Corporation, 8005 S.W. Boeckman Rd., Wilsonville, OR 97070

Abstract

A solution for mapping core I/O pins to System-On-a-Chip (SOC) I/O pins in order to achieve cost-efficient concurrent test for core-based designs is presented in this paper. The problem of pin mapping is first formulated as two well-known NP-complete problems. A heuristic algorithm is then proposed to determine a solution. The objectives driving this solution are geared towards reducing the total number of SOC pins needed and satisfying the test constraints specified by core integrators. Experimental results demonstrate the efficiency of the proposed method.

1 Introduction

Core-based SOC design strategy is becoming more and more popular these days. The Semiconductor Industry Association's (SIA) Technology Roadmap [1] predicts the percentage of reusable cores in SOC to be rising to 80% in 2006, thereby resulting in a 50% reduction of time-to-market. However, conflicting design objectives such as increasing complexity and reduced design cycles have made the test application time a major bottleneck towards achieving aggressive marketing requirements. Concurrent SOC testing (i.e. testing more than one core simultaneously) is becoming an attractive solution to reduce the total test application time under such circumstances. In this paper, the pin mapping problem in concurrent SOC testing is addressed, and a solution is presented to optimize the total number of SOC pins needed.

The paper is organized in the following manner. In Section 2, related work in the SOC test scheduling area is reviewed. In Section 3, various formulations of the pin mapping problem are presented, and, in Section 4, a heuristic algorithm to achieve optimized concurrent SOC test is proposed. Experimental results are presented in Section 5, followed by the conclusion section.

2 Review of Related Work

The complexity of a SOC makes manufacturing test a much more difficult problem than before. Many new DFT techniques have been exploited to address this problem. However, considering test issues for individual cores and User Defined Logic (UDL) may not be enough. The SOC composite test requires adequate test scheduling given a number of chip level requirements such as total test time, power dissipation, pin limitations etc. Test scheduling is also necessary to run intra-core and inter-core tests in a certain order that does not impact the contents of individual cores. SOC test should be created to satisfy these scheduling constraints. The requirements for test scheduling will become more complex as the level of SOC integration increases. Previous research in the area is discussed in the following paragraphs.

Sugihara et al. [2] addressed the problem of selecting a test set for each core from a set of test sets provided by the core vendors, and then schedule the test sets in order to minimize the test application time. The authors assumed that: 1) each core has its own BIST logic, 2) external testing can be carried out for only one core at a time, and 3) core vendors provide multiple test sets comprising of both BIST and external test patterns for each core. The problem was modeled as a combinatorial optimization problem and solved using a heuristic method. Chakrabarty [3] generalized the test scheduling problem of [2]. He assumed that Test Access Mechanism (TAM) includes one external test bus and multiple BIST resources, and the cores have been assigned to a test bus. The problem was formulated as an m-processor open shop scheduling problem, and solved by using a Mixed-Integer Linear Programming (MILP) model in order to minimize the test time. Ravikumar et al. [4, 5] proposed a method to solve test scheduling problem under the constraint of power consumption. They assume that BIST is the only methodology for testing individual cores.

All the above mentioned test scheduling algorithms for core-based systems ([2]-[5]) assume a fixed TAM structure. However, recently, there have been a few works considering scheduling problem together with the test resource allocation [6]-[11].

Chakrabarty [6] assumed a TestRAIL TAM structure and formulated the problem with an ILP model to find an optimum solution for allocating N test lines to a fixed number (N_B) of test buses, and assign each core to a test bus in order to minimize the total test time. The place-and-route and power constraints were also considered in [7]. In [8], a technique to determine optimal SOC test schedules with precedence constraints, i.e. schedules that preserve desired ordering among tests were introduced. An algorithm was proposed to solve

the problem in polynomial time by using preemption. Bagchi et al. [9] addressed the same problem as in [6], but considered clustering some cores into a module, and schedule testing of modules rather than individual cores in order to reduce the total test time.

Marinissen et al. [10] broke down a test into a test protocol and a list of test patterns. The test protocols, originally defined at the core terminals, are translated to the IC pins by Test Protocol Expansion. Subsequently, the expanded test protocol together with the test pattern list are used to assemble an IC-level test. The test scheduling problem was formalized as a No-Wait Job Shop scheduling problem. Huang et al. [11] proposed a method to allocate test resources and schedule test sets in order to achieve optimal concurrent SOC test. The objective was to minimize the test application time, while offering full scan / partial scan / functional tests for different TAMs under the constraint of peak power consumption.

Compared with the above-mentioned methods, the method proposed in this paper does not target the test scheduling problem, instead, it uses the test scheduling information as a constraint to optimize the necessary SOC resources.

The concurrent test scheduling information is specified usually by core integrators. Because core integrators have access to the information such as test application time and power dissipation for each core, they prefer specifying groups of cores that should be tested concurrently. In practice, this scenario is more typical when core integrators happen to be core providers themselves.

In this paper, the problem of mapping core pins to SOC pins to access cores from chip-level I/Os directly under the test scheduling constraints is addressed. There are several major techniques for direct access of embedded core, such as multiplexed access [12, 13], Test Bus [14], and TestRAIL [15]. Our methodology is a general solution to fit all the above test access mechanisms. Given a fixed set of concurrent test groups and the total number of SOC pins, the method proposed in this paper can find the optimal resource requirement and allocation that can be used to satisfy the specified concurrent test constraints and save valuable chip-level I/Os as much as possible.

3 Problem Formulations

3.1 Assumptions

Before the description of the problem, some of the assumptions are listed as follows.

1. It is assumed that only a subset of the SOC I/Os are available for mapping (either via MUXes or through test buses) based on the functionality and user preferences. Specifically, the dedicated SOC pins such as clocks, scan inputs, scan outputs, scan enables and test enables are excluded from consideration. A total of N SOC I/Os are considered ready for mapping.

2. User-Defined Logic (UDL) is treated as a core, and for each core, there are clearly modes available to place it into isolation or subject it to test. The isolated core remains free from any harmful effects of input changes at the core boundary and test patterns can be applied in an unconstrained fashion to other core(s) under test. The cores that are specified for serial test or internal BIST will use the dedicated resources, and are not considered in the problem formulation. It is assumed that there are a total of K cores waiting for resource allocation.

3. For each core C_i ($0<i\leq K$), the number of I/O pins that need mapping to SOC pins is W_i. It is assumed that the SOC pins can be bi-directional in nature. Therefore the pin direction is not a constraint in the problem we address, i.e. any pin of a core can be mapped to any pin of SOC.

4. All core pins are assumed to have direct access from the SOC pins.

3.2 Problem Statement

The problem is formally stated as follows. Given N SOC pins and K cores, for each core C_i ($0<i\leq K$), its total number of I/O pins W_i is recorded as a weight for core i. Let Ω indicate the set of groups. In each group, there are a set of cores which have to be tested concurrently. A core can appear in different groups. The objective is to determine a one-to-one mapping from pins of each core C_i to the SOC pins. The realization of the concurrent SOC test needs to satisfy the following conditions.

(1) The total number of SOC pins has to be minimized. If the determined number of SOC pins M is less than the maximum allowed pins (N), then one can use the additional available SOC pins to balance the load for each pin or to alleviate the routing congestion. If $M>N$, the program reports back to the core integrator about pin shortage. It is then up to the core integrator to decide whether to add more SOC pins or change the grouping constraints.

(2) At any given time t, a group / set of cores (C_{t1}, C_{t2}…C_{tr}) included in Ω can be tested simultaneously.

3.3 Problem Formulation I

The problem can be transformed to a well-known chromatic number problem as follows.

A graph G(V,E) corresponding to the problem given in the last subsection is built. Each vertex represents a pin on a core. An edge exists between two vertices if and only if one of the following condition holds.

(1) The two pins represented by these two vertices belong to the same core.

(2) The two pins represented by these two vertices belong to two different cores, but those two cores are specified in one concurrent group in Ω.

Now, the problem is easily transformed to a chromatic number problem. Each vertex is assigned a color, which means that the core pin represented by the vertex is to be mapped to a SOC pin indicated by that color. The original problem is transformed to find minimum number of colors to achieve a proper coloring of G. A proper coloring of G occurs when no two adjacent vertices have the same color. Chromatic number problem is a notoriously well-known NP-complete problem [16].

3.4 Problem Formulation II

The same problem could also be formulated as a dependency matrix partitioning problem used in pseudoexhaustive test [17]. The dependency matrix corresponding to the problem given in subsection 3.2 has m rows and n columns, where m is the cardinality of the concurrent test set Ω and n is the total number of the pins for all cores. An entry a_{ij} is "1" if and only if the corresponding pin of the j^{th} column belongs to a core which is included in the i^{th} concurrent test group. All other entries are "0". The dependency matrix partitioning problem is formed by partitioning the columns of the dependency matrix into sets such that each row of a set has at most one 1-entry and the total number of sets is a minimum.

It was shown that the above problem is also NP-complete [18].

4 Proposed Algorithm

In this section, a clique partitioning based heuristic method is proposed to solve the concurrent SOC testing problem. Note that both the above formulations are pin-based. Although there are several heuristics for solving the chromatic number problem, one needs to be careful while using them in this scenario because the total number of pins of cores in a SOC could be very large (could be up to ~10K), thereby leading to very expensive computational cost. In addition, the pin based heuristic solutions are typically 10% off, and as much as 100% off from the exact solutions [19]. In contrast, the number of cores in a SOC is limited (typically, less than 100 in practical designs). Therefore, in this paper a heuristic solution based on cores rather than pins is proposed.

The basic idea of the algorithm is to find a lower bound for this problem, and increase the lower bound gradually until a solution is found. The algorithm is described in Fig. 1.

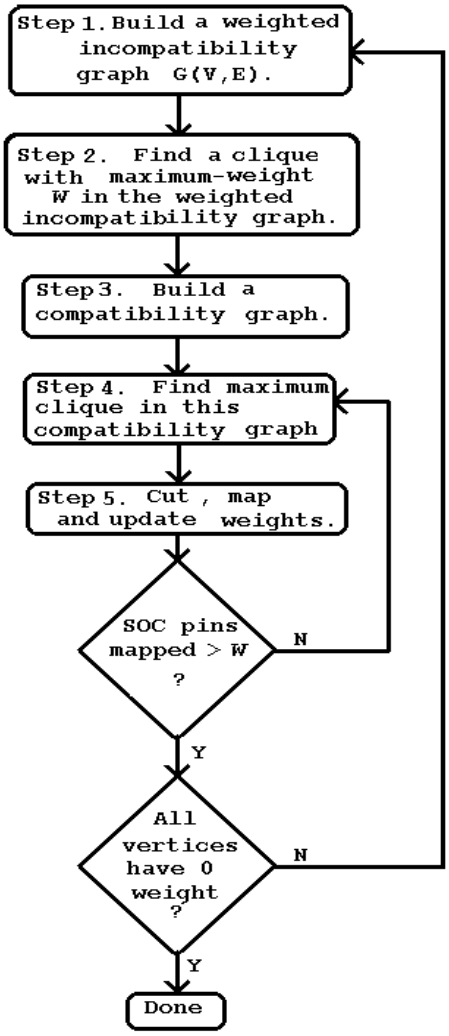

Fig. 1. Flow Chart of the Proposed Algorithm.

The steps of the algorithm as presented in the flow chart are explained in detail with the following example.

Example 1: Let there be 7 cores C_1, C_2, … C_7 in a design and the number of pins for these cores be 200, 200, 100, 150, 160, 100, and 80 respectively. Let the concurrent test groups be the following:

Ω = { (C_1, C_2), (C_1, C_4), (C_1, C_6, C_7), (C_2, C_5), (C_3, C_4), (C_3, C_5), (C_3, C_6, C_7) }.

To apply the proposed algorithm in order to get an optimal number of SOC pins under the given constraints, the following steps are necessary.

Step1 To start, a weighted incompatibility graph G(V,E) is built. A weighted incompatibility graph is

defined as a graph where each vertex in V represents a core and an edge between two vertices exists iff these two vertices appear in the same group at least once (The vertices are incompatible since they can not share any SOC pins). A weight attached to each vertex is the number of pins of the corresponding core.

The incompatibility graph for the given example is given in Fig. 2.

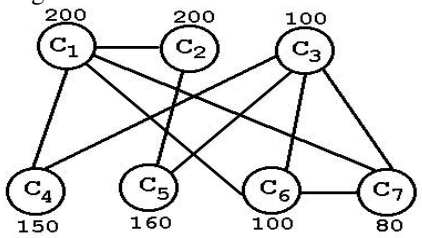

Fig. 2. Incompatibility Graph for Example 1.

Step2 A maximum-weight clique in the weighted incompatibility graph is obtained with the total weight *W*. A clique of G is a complete subgraph of G. Hence, no two cores corresponding to a pair of vertices in the clique can share SOC pins. Therefore, a lower bound on the number of required SOC pins is the total weight of this clique. Since the number of cores in practice is limited (<100), it is computationally feasible to solve the maximum weight clique problem to get an exact solution. In the proposed algorithm, we use a branch and bound method proposed in [20] to solve the maximum weight clique problem.

In Example 1, the maximum weight-clique is (C_1, C_2). The total weight *W* is 400.

Step3 Based on the incompatibility graph, a compatibility graph is built. The compatibility graph is the complement of the incompatibility graph. For Example 1, the compatibility graph is shown in Fig. 3.

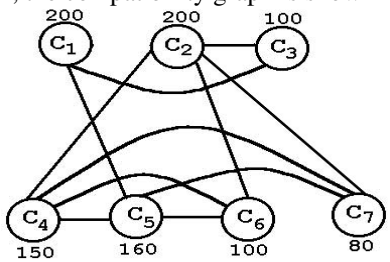

Fig. 3. Compatibility Graph for Example 1.

Step4 Next, a maximum clique in the compatibility graph is determined that satisfies the following conditions.

(1) At least one core is selected from the maximum-weight clique as determined in Step 2.

(2) The maximum clique is determined without considering their weights.

(3) If there are more than one candidate groups, the group that has the largest total weight is picked. If there is still a tie, one of the groups is picked randomly.

In Example 1, there are 2 maximum cliques that satisfy the above-mentioned conditions, (C_2, C_4, C_6) and (C_2, C_4, C_7). Group (C_2, C_4, C_6) is picked as it has a larger weight.

Step5 Let w_i be the weight of core i that is included in the maximum clique selected in Step 4. Min(w_i) is the minimum weight of a core in this clique. Min(w_i) pins are selected out of each core in the clique, and are mapped to the same SOC pins. For each core selected in the clique, the weight is updated to (w_i - Min(w_i)). If a core has weight 0, the corresponding vertex and all the edges incident to it are deleted.

In order to illustrate the mapping process, let C[*a*,*b*] indicate the pins in the range from *a* to *b* in a core or the SOC. For the given example, C_2[1,100], C_4[1,100], and C_6[1,100] are mapped to SOC[1,100], and the updated compatibility graph is shown in Fig. 4.

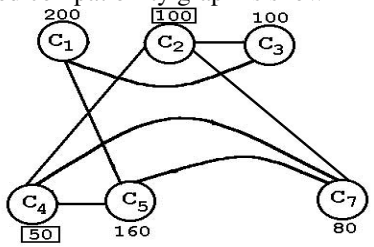

Fig. 4. Updated Compatibility Graph

Steps 4 and 5 are repeated until the number of mapped SOC pins reach the maximum limit *W*, which is determined in Step 2. The process is shown in figure 5.

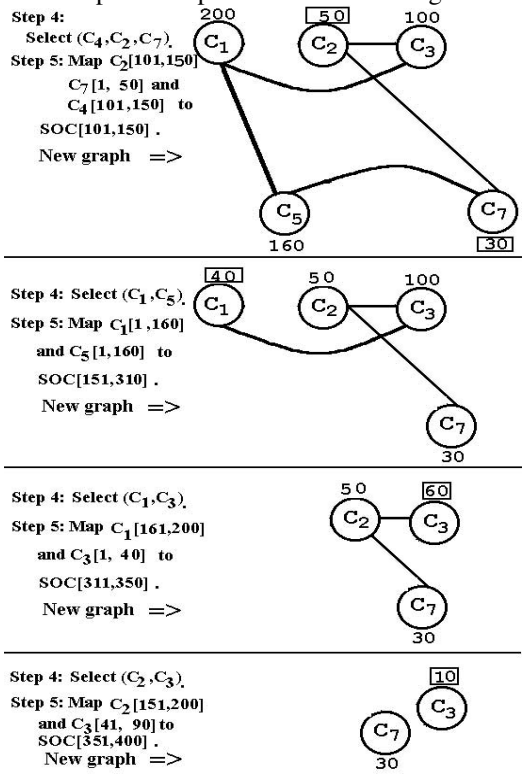

Fig. 5. Updating Compatibility Graph for Example 1.

At this point, there are still 2 vertices left, for which one needs to go back to Step 1 to repeat the above steps iteratively until there is no vertex left in the graph. At the end, pins C_7[51,80] are mapped to SOC[401,430], and pins C_3[91,100] are mapped to SOC[431,440]. Therefore, a total 440 SOC pins are needed to concurrently test the cores under the specified constraints.

5 Experimental Results

In order to evaluate the proposed algorithm, we compare the results with two greedy algorithms – namely Heuristic 2 (H2) and Heuristic 3 (H3). H2 is based on the information of a core and its immediate neighbors. In this case, a degree is associated to each core, which is the sum of the weights of all the neighbors including the core itself. A core with the maximum degree is selected first for mapping. In heuristic H3, the algorithm is the same except that the degree for each core is now calculated based on just its neighbors (without considering the weights). In other words, both these algorithms try to map the most "hard-to-map" cores first based on some locally optimal decisions. In addition, the proposed method is also compared with another greedy algorithm where the cores are ordered randomly for pin mapping. All these algorithms were implemented to provide a fair comparison of the proposed method as no such work in this area is reported in the literature.

These algorithms are run on 9 hypothetical but nontrivial SOCs (S1 to S9) and 2 real industrial designs (IND1, IND2). The number of pins for each core is randomly generated between 30 and 300 for S1 to S9. The number of cores and number of pins for each core in SOCs IND1 and IND2 were fixed when designed for commercial use. However, since the core grouping information is not available, the concurrent test groups for all the circuits were generated randomly. The algorithms were implemented in C running on a SUNBlade1000 workstation. The CPU time for each benchmark is less than 1 second for the proposed algorithm (H1) and up to ~10s for H2 and H3. That's why the CPU time is not included in the tabulated results.

The experimental results are shown in Table 1. The number of cores for each test case is shown in the second column. The lower bound of SOC pins given by the proposed clique partitioning algorithm are in Column 3. The SOC pins obtained in the proposed algorithm (H1) are listed in Column 4. The incremental number of SOC pins obtained by algorithms H2 and H3 over H1 are shown in Column 5 and Column 6 respectively. The next 6 columns show the results obtained by 6 runs of the greedy algorithm based on random selection. From the results, it is evident that:

(1) The proposed algorithm H1 results in the best solution for all the test cases used. Note that the results are quite close to the lower bound.
(2) The order of mapping is very critical. A bad ordering of cores could lead to results that are far from the optimal solution.
(3) The greedy algorithm H2 is better than H3.
(4) The greedy algorithm based on random core selection usually can not guarantee good solutions.

On average, the total number of SOC pins computed by greedy algorithms H2 and H3 are about 3% and 5.5% more than the proposed method H1 respectively. The random algorithm, on the other hand, may use upto 13% more SOC pins.

In order to compare the heuristics with an exact solution, a pin-based exact coloring algorithm is implemented using the method proposed in [19]. However, for the test cases used in Table 1, the computational time is exponentially large and far from completion. For some of the large test cases in Table 1, the exact algorithm could not come up with a solution after running for even 2 days. Therefore, small test cases were used to compare the results. The results are shown in Table 2. In Column 2, the total numbers of pins on all cores are given. This is followed by the optimum number of SOC pins necessary in each case. In the last 3 columns, the numbers of SOC pins obtained by H1, H2, and H3 are presented.

The computational time for all heuristics are less than 0.01s for all the test cases in Table 2, whereas the exact solution suffers from exponential computational time. Even for these small test cases, the CPU time for the exact solution is upto 6386.3s. The results in Table 2 show that the proposed heuristics can come up with solutions that are very close to the exact solution, and would therefore provide fairly good results for all practical scenarios.

6 Conclusions

In this paper, an algorithm to allocate test resources efficiently for achieving concurrent SOC test under specific test constraints is presented. The objective was to minimize the total number of SOC pins required for a design. The problem can be formulated as a chromatic number or a clique partitioning problem. A heuristic algorithm is proposed to solve the problem. Experimental results show that the proposed method produces much better results when compared with several greedy approaches and is cost efficient when compared with an exact solution.

References

[1] Semiconductor Industry Association, The National Technology Roadmap for Semiconductors, Sematech, Inc. pp24, 1997.

[2] M. Sugihara, H.Date, and H. Yasuura, "A Novel Test Methodology for Core-Based System LSIs and a Testing Time Minimization Problem," pp. 465-472, ITC 1998.
[3] K. Chakrabarty, "Test Scheduling for Core-Based Systems Using Mixed-Integer Linear Programming," pp.1163-1174, IEEE TCAD, Vol. 19, Oct., 2000.
[4] C. P. Ravikumar, A. Verma, and G. Chandra, "A Polynomial-Time Algorithm for Power Constrained Testing of Core Based Systems," pp.107-112, ATS 1999.
[5] C. P. Ravikumar, G. Chandra, and A. Verma, "Simultaneous Module Selection and Scheduling for Power-Constrained Testing of Core Based Systems," VLSI Design 2000.
[6] K. Chakrabarty, "Design of System-on-a-Chip Test Access Architectures using Integer Linear Programming," pp. 127-134, VTS 2000.
[7] K. Chakrabarty, "Design of System-on-a-Chip Test Access Architectures under Place-and-Route and Power Constraints," DAC 2000, pp.432-437.
[8] V. Iyengar and K. Chakrabarty, "Predence-Based, Preemptive, and Power-Contrainted Test Scheduling for System-on-a-Chip," VTS 2001, pp.368-374.
[9] D. Bagchi, D.R.Chowdhury, and J. Mukherjee, "A Novel Strategy to Test Core Based Designs," pp.122-127, VLSI Design 2001.
[10] E. J. Marinissen, and J. Aerts, "Test Protocol Scheduling for Embedded-Core Based System ICs," IEEE International Workshop of Testing Embedded Core-based System-Chips, 1998.
[11] Y. Huang, W.-T. Cheng, C.-C. Tsai, N. Mukherjee, O. Samman, Z. Yahya, and S. M. Reddy, "Resource Allocation and Test Scheduling for Concurrent Test of Core-Based SOC Design," ATS 2001, accepted.
[12] V. Immaneni, and S. Raman, "Direct Access Test Scheme – Design of Block and Core Cells for Embedded ASICs," pp.488-492, ITC 1990.
[13] S. Bhatia, T. Gheewala, and P. Varma, "A Unifying Methodology for Intellectual Property and Custom Logic Testing," ITC 1996, pp.639-648.
[14] P. Varma and S. Bhatia, "A Structured Test Re-Use Methodology for Core-Based System Chips," pp.294-302, ITC 1998.
[15] E. J. Marinissen, R. Arendsen, G. Bos, H. Dingemanse, M. Lousberg, and C. Wouters, "A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores," pp.284-292, ITC 1998.
[16] M.R.Garey and D.S.Johnson, Computers and Intractability: A Guide to The Theory of NP-Completeness, W.H.Freeman Company.
[17] E. D. Mccluskey, "Verification Testing – A Pseudoexhaustive Test Technique," IEEE Trans. On Computers, vol. C-33, No.6, June, 1984.
[18] F. Hirose and V. Singh, "McDDP, A program for partitioning verification testing matrices," Center for Reliable Computing, Stanford Univ., Stanford, CA, Tech. Rep. 81-13, July, 1982.
[19] O. Coudert, "Exact Coloring of Real-Life Graphs is Easy," pp.121-126, DAC 1997.
[20] P.R.J.Ostergard, "A New Algorithm for the maximum-weight clique problem," Discrete Applied Mathematics, 2001.

Table 1: Comparison of the Proposed Concurrent SOC Pin Mapping with Random Method.

SOC Circuits	# of Cores	Lower Bound	SOC Pins achieved by the proposed algorithm H1	Additional SOC Pins Needed by H2	Additional SOC Pins Needed by H3	Additional SOC Pins needed by the Random Core Selection Method.					
						I	II	III	IV	V	VI
S1	10	338	338	0	0	44	0	28	15	36	79
S2	15	469	502	55	55	55	22	92	79	100	55
S3	20	505	547	3	3	96	85	49	34	94	36
S4	25	609	609	0	40	92	153	77	126	86	72
S5	30	482	482	37	40	51	105	99	120	58	5
S6	35	477	496	14	84	111	115	97	78	134	144
S7	40	704	717	32	76	73	71	37	166	32	111
S8	45	606	606	56	12	56	103	70	69	78	106
S9	50	507	507	2	0	66	25	59	0	42	70
IND1	23	814	814	0	0	38	0	94	38	41	38
IND2	34	811	822	6	48	48	147	173	269	217	88
Avg.	30	575	585	18	32	78					

Table 2: Comparison of the Heuristic Concurrent SOC Pin Mapping With the Exact Solution.

Test Case	Total # of Pins on All Cores	SOC Pins obtained by Exact Solution	SOC Pins obtained by H1	SOC Pins obtained by H2	SOC Pins obtained by H3
I	22	13	14	14	14
II	23	7	8	8	9
III	24	7	8	9	9
IV	25	13	13	13	13
V	26	14	14	14	14
VI	27	8	9	9	10
VII	28	8	9	9	10
VIII	29	15	15	15	15