SoC Test Scheduling Using the B*-Tree Based Floorplanning Technique *

Jen-Yi Wuu[1], Tung-Chieh Chen[2], and Yao-Wen Chang[3]

[1]Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan
[2]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan
[3]Department of Electronic Engineering & Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan
jywuu@eda.ee.ntu.edu.tw, tungchieh@ntu.edu.tw, ywchang@cc.ee.ntu.edu.tw

Abstract— We present in this paper a new algorithm to co-optimize the problems of test scheduling and core wrapper design under power constraints for core-based SoC (System on Chip) designs. The problem of test scheduling is first transformed into a floorplanning problem with a given maximum height (test access mechanism width) constraint. Then, we apply the B*-tree based floorplanning technique to solve the SoC test scheduling problem. Experimental results based on the ITC'02 benchmarks show that our method is very effective and efficient---our method obtains the best results ever reported for SoC test scheduling with power constraint in every efficient running time. Compared with recent works, our method achieves average improvements of 4.7% to 20.1%.

I. INTRODUCTION

In order to shorten the time-to-market, SoC designs with embedded cores have become more and more popular. As a result, SoC testing has become an important research problem. The major objective of SoC testing is to reduce the test time. During the testing of the embedded cores, test access mechanisms (TAMs) are used to transport test vectors between SoC pins and the core wrappers. The core wrapper forms an interface between a core and the TAM. Since the TAM wires are limited to a certain number and the cores usually have more pins than the TAM wires, the wrapper is also designed to match the TAM width to the number of core pins when they do not match. After the design of core wrappers, the problem of test scheduling arises. Test scheduling is a process that determines the start and finish times for each core test and the assignment of the TAM wires to the tests so that the overall test time is minimized. Therefore, given the number of SoC pins, the specific parameters of the cores under test, and the maximum allowable power consumption during test intervals, it is desired to solve the co-optimization problem of core wrapper design and test scheduling such that the overall test time is minimized.

A. Previous Work

Many recent works considered various aspects of the SoC test scheduling problem. Most of the earlier works considered only some particular objectives of the problem (i.e., solve some sub-problems). Since co-optimization of the SoC test problem such as core wrapper design, test scheduling, and peak power dissipation, is essential for solving the real problem, several recent works have started to consider the co-optimization issues.

Chakrabarty in [2] proposed an integer linear programming model for minimizing test time by co-optimizing the bandwidth distribution and test bus assignment. Iyenger et al. in [3] discussed the assignment of TAM wires to partitions. Huang et al. in [4] modeled the SoC test problem with power constraints as a 3-D bin-packing problem and provided a heuristic to handle the problem. Iyengar et al. in [1] presented an approach for the co-optimization of the core wrapper design and TAMs by using rectangle packing and considering the assignment of non-consecutive TAM pins to core wrappers. Zou et al. proposed in [5] a method using simulated annealing (SA) and the sequence pair representation to handle the problem. Xia et al. in [6] presented an algorithm for co-optimizing test scheduling and wrapper design under power constraints by using an evolutionary algorithm and the sequence pair representation. Xia et al. also proposed an algorithm for assigning non-consecutive TAM wires to core tests.

B. Our Contribution

In this paper, we present a new approach to the problem of the co-optimization of core wrapper design and test scheduling so that the SoC test time is minimized. In our approach, we first extend the wrapper design method presented in [5]. Then, we transform the test scheduling problem to a floorplanning problem. The height of the floorplan corresponds to the TAM width, and the width of the floorplan corresponds to the total test time. Since the TAM width is given, the maximum height of the floorplan is determined. Therefore, we find the floorplan with the minimum width, corresponding to a test scheduling with the minimum test time.

We develop a two-stage, low-temperature simulated annealing algorithm based on the B*-tree floorplanning representation to find a desired floorplan (and thus a desired test schedule). In the first stage, we only deal with the tests that have longer average test times. In the second stage, we add the remaining tests into the final configuration of the first stage to obtain an initial solution for simulated annealing. In particular, we use a low-temperature simulated annealing scheme to preserve the main structure of the scheduling result obtained from the first stage. Experimental results (with and without power constraints) show that our algorithm can obtain the best average test times among all published works. Further, our method is very efficient.

The remainder of this paper is organized as follows. Section 2 formulates the SoC test scheduling problems and gives an overview of our algorithm. The methods of the wrapper design for the test cores are presented in Section 3. Section 4 shows how to transfer a test scheduling problem into a floorplanning problem. Section 5 describes the B*-tree representation. Section 6 presents the heuristic of two-stage annealing schedule. Section 7 considers the power constraints. We present our experimental results based on the ITC'02 benchmark in Section 8 and give concluding remarks in Section 9.

II. PROBLEM FORMULATION

Let an SoC design consist of N cores, and each core C_i, where $1 \le i \le N$, has n_i data inputs, m_i data outputs, b_i bidirectional I/O's, sin_i scan inputs, and $sout_i$ scan outputs. Let K be the total width of the TAMs. For each core C_i, there are P_i test patterns. Also, the maximum peak power consumption during testing is given.

Given the aforementioned inputs, the SoC test problem determines the TAM architecture, the wrapper design for all the wrapper-based cores, and the schedule for the core tests so that the overall test time is minimized and the power consumption constraint is satisfied.

The SoC test problem can be divided into two parts: (1) the core wrapper design: Optimize the core wrapper design for each TAM width under the maximum TAM wire constraint and generate a set of wrapper designs for each core; (2) test scheduling: Optimize the schedule for the core tests. It will be clear later that we develop a two-stage, low-temperature simulated annealing algorithm based on the B*-tree floorplanning technique to solve the test scheduling problem.

III. SoC CORE WRAPPER DESIGN

The SoC core wrapper is the interface between the TAM and the core. Since large cores usually have more pins than the number of TAM wires used to perform the test of the core, the core wrapper may often need to perform width adoption when the TAM width is not equal to the number of core pins. Also, while different wrapper designs greatly affect the test time of the core, optimizing wrapper design can improve the TAM efficiency and reduce the test time. To calculate the test time, T, for a wrapper we use the following expression [8]:

$$T = \{1 + \max(S_i, S_o)\}P + \min(S_i, S_o), \qquad (1)$$

where S_i is the length of the wrapper's input scan chain, S_o is the length of the wrapper's output scan chain, and P is the number of test patterns.

For cores with internal scan chains, we use the wrapper design algorithm based on the Best Fit Decreasing (BFD) heuristic [3]: The internal scan chains are first sorted in decreasing order of length and then assigned to a wrapper scan chain whose length after the assignment is the closest to, but not exceeding the length of the current longest wrapper scan chain. If there is no such wrapper scan chain available, the internal scan chain is assigned to the current shortest wrapper scan chain. This process is then repeated for the functional inputs and outputs.

For the cores without internal scan chains, we use the unbalanced design introduced in [5] and allow different numbers of SoC pins to be assigned to scan-in and to scan-out to achieve the optimal wrapper designs for the core. Besides, we found that the previous works [4] and [5] considered one special wrapper design for cores without internal scan chains. It was claimed that if given enough TAM wires to the inputs and outputs of the core, the test can be

* This work was partially supported by the SpringSoft, Inc. and the National Science Council of Taiwan under Grant No's. NSC 92-2220-E-002-013 and. NSC-93-2752-E-002-008-PAE

completed without using a clock-cycle to scan in/out. Therefore, the test time for this core equals the patters needed to be tested. For example, for a core with 17 inputs, 3 outputs, and 2666 patterns to test, if we assign 17 pins to inputs and 3 pins to outputs, that is, a total of 10 TAM wires, the test time will be (1+0)*2666+0 = 2666 clock cycles.

Using the wrapper design methods discussed above, for each core we can generate a set of wrapper configurations with the TAM wire usage from 1 to K, where K is the maximum TAM wires allocated for the SoC test application. Each wrapper configuration can be represented by a rectangle with width equal to the test time of the core and height equal to the TAM wire usage. Thus, each configuration can be represented by a two-tuple (W_{ij}, $T(W_{ij})$), where W_{ij} is the number of TAM wires of the *j-th* wrapper configuration for core i, and $T(W_{ij})$ is the corresponding test time. It can be shown that the relationship between the test time and the wrapper width for a given core is a "staircase" function. As defined in [1], the designs that represent the smallest TAM width for a specific test time are known as Pareto-optimal designs. Since the Pareto-optimal designs use fewer TAM wires than those with the same test time, only these wrapper designs would be considered in the test scheduling.

IV. Test Scheduling Problem

Given an SoC with K TAM wires, a set of N cores, and a set of Pareto-optimal wrapper designs for each core, we intend to find an assignment of TAM wires to each core test and to determine the test start time for each test so that the overall test time is minimized. We can transform the test scheduling problem into a floorplanning problem, where the height of the floorplan represents the fixed number of TAM wires, K, and the width of the floorplanning represents the SoC test time that has to be minimized. Each core test is represented by a rectangle with width equal to the test time and height equal to the TAM wire. Since at any time during the test, any TAM wire can only be assigned to one core to perform the test, the rectangles in the floorplanning problem transformed from the test scheduling problem cannot overlap. Figure 1 shows a floorplan corresponding to the test schedule of the ITC'02 benchmark circuit D695.

A number of floorplan representations have been proposed in recent years. Among them, the B*-trees [7] have been shown to be very efficient and effective for various floorplan problems. For most floorplanning problems, the B*-trees achieve the best results. Therefore, we shall solve the test scheduling problem using the B*-tree floorplan representation.

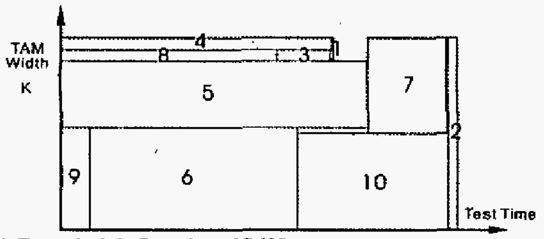

Fig. 1. Test-schedule floorplan of D695

V. B*-Tree Representation

A B*-tree is an ordered binary tree to represent a non-slicing or a slicing floorplan [7]. Given a compacted placement P in which no module can either move down or move left (called an *admissible* placement [11]), we can represent it by a unique B*-tree T. (See Figure 2(b) for the B*-tree representing the placement of Figure 2(a).) The root of a B*-tree corresponds to the module on the bottom-left corner. Using the depth-first-search (DFS) procedure, the B*-tree T for an admissible placement P can be constructed in a recursive fashion. Let R_i denotes the set of modules located on the right-hand side and adjacent to m_i. The left child of the node n_i corresponds to the lowest module in R_i that is unvisited. The right child of n_i represents the lowest module located above m_i, with its x-coordinate equal to that of m_i

The B*-tree keeps the geometric relationship between two modules as follows. If node n_j is the left child of node n_i, module m_j must be located on the right-hand side of m_i, with $x_j = x_i + w_i$. Besides, if node n_j is the right child of node n_i, module m_j must be located above m_i, with the x-coordinate of m_j equal to that of n_i; i.e.,$x_j = x_i$. Also, since the root of T represents the bottom-left module, the coordinate of the module is $(x_{root}, y_{root}) = (0, 0)$. Each y-coordinate can be computed by a *contour* data structure in amortized constant time [7] [11].

In a classical floorplanning problem, the x-coordinates and the y-coordinates of modules are determined to achieve the minimum enclosing area or wirelength. Transforming a test scheduling problem to a floorplanning problem, we model a test as a module. The x-coordinate of a test represents the start time of the test while the y-coordinate represents the TAM wire assignment of the test. In a test-schedule floorplan, we set an upper bound for the height of the floorplan (representing the TAM width) and aims at minimizing the width of the floorplan. Therefore, the transformation of test scheduling into a floorplanning problem makes the B*-tree based floorplanning algorithm very suitable for this problem. Moreover, since we need to perform perturbation moves to search for neighboring solutions frequently, the remarkable efficiency of the B*-tree representation for performing tree operations would be a very significant advantage for our problem.

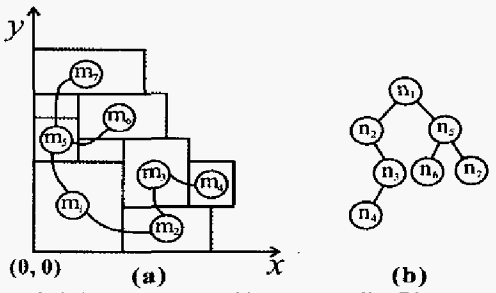

Fig. 2. An admissible placement and its corresponding B*-tree.

VI. Simulated Annealing

Simulated annealing was first introduced by Kirkpatric et al. in 1983 [9]. In [5], Zou et al. applied simulated annealing to SoC test scheduling. Simulated annealing begins with an initial solution, and a neighboring solution is generated by performing a perturbation on the current solution. If the cost of the neighboring solution is lower than that of the current solution, the neighboring solution is accepted. If the cost of the neighboring solution is higher than that of the current solution, it is either rejected or accepted with some probability. The probability of accepting an inferior solution is a function of a parameter called temperature. The probability function is defined as follows:

$$p = e^{\frac{-\Delta C}{T}} \tag{2}$$

where ΔC is the change in cost between the neighboring solution and the current solution, and T is the temperature parameter which decreases as the search time increases. At the beginning of the algorithm, the temperature parameter is large; therefore, there is a high probability that inferior solutions would be accepted. This feature of accepting inferior solutions makes simulated annealing possible to escape from a local optimum and to reach another region of the solution space that may contain the global optimum. The temperature of each iteration decreases so the probability of accepting an inferior solution is getting lower, and finally an optimal solution can be obtained.

The solution space of the test scheduling problem consists of all test-schedule floorplans that satisfy the following two constraints: (1) the height of the floorplan cannot be larger than the TAM width, K; (2) no two rectangles in the floorplan may overlap.

For SoC test scheduling, the minimization of the test time is our primary concern. Thus, the cost function of the simulated annealing algorithm is set to the width of the test-schedule floorplan (i.e., the overall test time) obtained from the B*-tree representation discussed in the previous section.

We perturb a B*-tree to another by the following operations:

- **Op1**: swapping two nodes.
- **Op2**: move a node to another place.
- **Op3**: change the size of a module.

Op1 deletes two nodes and inserts them into the corresponding positions in the B*-tree. Op2 deletes a node from a B*-tree and inserts it to another position. Op3 changes the size of a module. The probabilities of the three perturbation moves are all set to 1/3.

In order to reduce the running time and to get an optimal solution more efficiently, we propose a two-stage simulated annealing algorithm. In the first stage, we choose from all core tests that are larger in average test area (average test time multiplied by the TAM width). We deal with the larger tests that comprise 85% to 95% of the total rectangular area of the test cores in the first run. The remaining smaller tests are set aside temporarily at the first run. After the first run, we add all other smaller tests set aside to the left-most branch of the B*-tree. This floorplan is taken as the initial solution of the second-stage simulated annealing. In the second stage, we use low-temperature simulated annealing to further improve the floorplanning result to obtain better solutions. This heuristic is illustrated in Figure 3. Figure 3(a) shows a solution found after the first stage while Figure 3(b) shows a final solution slightly adjusted from the previous solution. Table 1 gives the experimental results on the ITC'02 benchmark P22810 using one-stage and two-stage simulated annealing. As shown in the table, the two-stage simulated annealing is more effective and efficient. Figure 4 shows the pseudo-code of our algorithm.

VII. Power Constraint

For the SoC test scheduling problem with power constraints, the peak power dissipation at any time during the test cannot exceed a certain limit, Q, also known as the SoC power budget. In order to compare with [6], we made the same assumptions and ran our experiments for two cases: (1) the maximum peak power is constant for a given core; (2) the maximum peak power is a linear function of the width of the test data.

To deal with the test-scheduling problem with power constraints, we calculate the maximum peak power consumption by building a power histogram after a test-schedule floorplan solution is generated. In the cost function of the simulated annealing algorithm, we added a penalty value that is in proportion to the difference of the power budget and the peak power consumption of the test schedule. We made the same assumptions as in [6] and used the ITC'02 benchmark for the experiments; that is, for case (1) we assume that the power numbers given in the benchmark represent the maximum peak power dissipated during core testing; for case (2) we assume that the given power numbers represent the maximum peak power when test is performed using only one TAM wire; that is, for example, if three TAM wires are used for a test, the given number of power dissipation for that core will be multiplied by three.

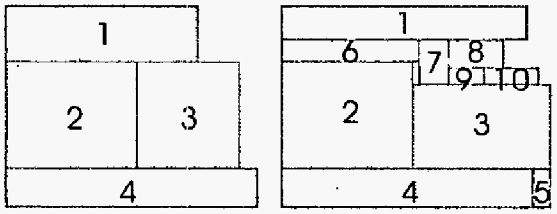

Fig. 3. (a) A stage-1 solution. (b) A resulting final solution obtained at stage 2.

```
Algorithm: SoC Test Scheduling
Input: C – set of SoC cores;
       K – total width of the TAMs;
       Q – power budget;
Output: Test schedule for the SoC tests
begin
1    For each SoC core
2        Generate a set of wrapper design with TAM width from
             1 to K
3        Find the Pareto-optimal designs
4    Sort the cores by average test area (test time * TAM width)
         in decreasing order
5    Make a set of cores C_L consisting of the 85%~90% of total
         test area so that for all cores in (C - C_L) are smaller than
         those in C_L
6    Randomly build a binary tree with these cores in C_L
7    Simulated_Annealing();
8    Insert set (C - C_L) to the left edge of the tree
9    Simulated_Annealing();      //low initial temperature
10   Output_Result();
end
```

Fig. 4. Pseudo-code of our algorithm

TABLE 1

COMPARISON OF THE ONE-STAGE AND TWO-STAGE SIMULATED ANNEALING BASED ON THE **P22810** BENCHMARK.

TAM wires	One-Stage SA		Two-Stage SA	
	Test Time (Clock Cycle)	**CPU Time (sec)**	**Test Time (Clock Cycle)**	**CPU Time (sec)**
16	438619	11	438619	7
24	292356	45	287999	8
32	222385	41	216747	24
40	179643	39	178223	28
48	150007	49	149592	17
56	131874	74	129624	39
64	117231	22	115622	9
Comp.	*1.012X*	*2.50X*	*1X*	*1X*

VIII. EXPERIMENTAL RESULTS

In this section, we present the experimental results for our method based on the ITC'02 SOC benchmarks [10]. The proposed method was implemented in C++ and executed on a PC with a Pentium 4 processor and 640 MB memory.

Table 2 shows the comparison of our results and those of the previous works. The numbers in bold face represent the best results ever reported. B*-SA denotes our proposed algorithm. EA(C) and EA(nC) denote the results reported from [6] for assignments of consecutive and non-consecutive pins of test cores to TAM wires, respectively. It can be seen that our results are generally better than the best previous works. Figure 5 shows the normalized average test time of the previous works listed in Table 2. Our improvements over the previous works range from 4.7% to 20.1%. In addition, our method is very efficient; the CPU times were less than 30 seconds for all benchmarks. One thing to note is that for some benchmarks we listed our results in two rows and named our methods B*-SA(1) and B*-SA(2) separately. In B*-SA(2) we considered the special core wrapper design mentioned in Section 3. With this special wrapper design, it is possible to complete a test using less time. B*-SA(1) denotes the method without using this special core wrapper configuration. For the benchmarks with only one row of data for B*-SA, we obtained the same results for B*-SA(1) and B*-SA(2).

Table 3 shows the comparison of our results and those from [6] for H953 with the power constraint. This is the only ITC'02 benchmark that includes the power data for each core. In this benchmark, each core is assigned a non-negative integer (with no unit specified) representing the peak power consumption. We worked on two assumptions, which are the same as those in [6], in order to make fair comparisons with [6]: (1) the maximum peak power for a given core is constant throughout testing time, and (2) the peak power dissipation for a given core is a linear function of the number of TAM wires used. For the cases with the first assumption, our results are exactly the same as those in [6]. This may be attributed to the fact that the power budges are not so tight that the optimal solution can be obtained easily. However, for the cases with the second assumption, we consistently obtain a better result under each power limit. Table 3 gives the comparison. In Figure 6, we show one of our results under the power constraint of 5,753,800,192. This result is based on the assumption that the peak power consumption is a linear function of the number of TAM wires used, which was the same as in [6]. It should be noted that we can obtain better results even using only three TAM wires; this phenomenon reveals the fact that the TAM width does not affect the resulting test times for the H953 benchmark, which conforms to the observation made in [6]. Figure 7 shows the resulting test schedule of P22810 of 24 TAM width.

IX. CONCLUSIONS

We have presented the B*-tree based floorplanning technique for the co-optimization of core wrapper design and test scheduling. The B*-tree representation is used to represent the test schedule. Simulated annealing is used to search for desired solutions. We also have proposed a two-stage simulated annealing scheme, with each stage given a different set of tests. Experimental results have shown that our method is very effective and efficient. Our test scheduling results for power and test time optimization are better than the most recent works.

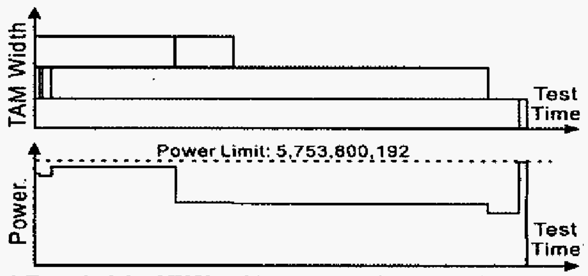

Fig. 6. Test schedule of H953 and its corresponding power histogram.

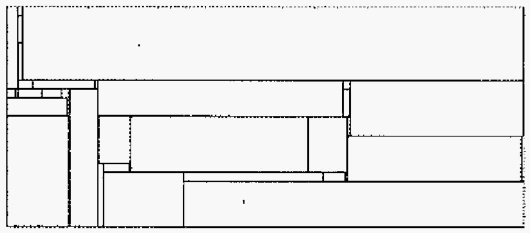

Fig. 7. Test schedule of P22810 of 24 TAM width. There are 28 cores, and the test time is 287,999 clock cycles.

TABLE 3

TEST SCHEDULING TIME UNDER THE POWER CONSTRAINTS GIVEN IN [6] (RESULTS FOR H953)

Power Limit	Number of TAM wires 16-64 (P = f(TAM))	
	B*-SA	**[6]**
5,753,800,192	**479285**	489945
$6*10^9$	**479285**	489945
$7*10^9$	**362733**	368191
$9*10^9$	**257141**	257151
$21*10^9$	**119357**	120861
$30*10^9$	**119357**	**119357**

TABLE 2

COMPARISON OF THE TEST SCHEDULING TIMES FOR THE ITC'02 BENCHMARKS. NOTE THAT THE PLATFORMS FOR B*-SA, EA[6], AND [3] ARE PC, SUN ULTRASPARC 5, SUN ULTRA 80, RESPECTIVELY, AND [5], [1], AND [4] DO NOT REPORT THEIR CPU TIMES. (THE NUMBERS IN BOLD FACE DENOTE THE BEST RESULTS EVER REPORTED.)

ITC'02 Benchmark	Method	Number of TAM wires							
		16	24	32	40	48	56	64	
		Test Time (Clock Cycle)	Test Time (Clock Cycle)	Test Time (Clock Cycle)	Test Time (Clock Cycle)	Test Time (Clock Cycle)	Test Time (Clock Cycle)	Test Time (Clock Cycle)	CPU Time (sec)
D695	B*-SA	**39489**	**26203**	**19773**	**16149**	**13649**	**11285**	**9885**	2
	EA(C)[6]	41533	27982	21014	16908	14240	11988	10571	4
	EA(nC)[6]	41809	27989	21142	17015	14236	12134	10788	7
	[5]	41604	28064	21161	16993	14183	12085	10723	
	[1]	43723	30317	23021	18459	15698	13415	11604	
	[3]	41949	28327	21423	17210	16403	13023	12327	290
	[4]	42716	28639	21389	17366	15142	13208	11279	
P22810	B*-SA	**438619**	**287999**	**216747**	178223	149592	129624	115406	9
	EA(C)[6]	438783	292824	226545	**167792**	153260	133094	117638	13
	EA(nC)[6]	**438619**	289237	228732	183133	153525	130949	116625	39
	[5]	**438619**	289287	218855	175946	**147944**	**126947**	**109591**	
	[1]	452639	307780	246150	197293	167256	145417	136941	
	[3]	462240	361576	312662	278360	268474	266800	260639	120
	[4]	446684	300723	223462	184951	167858	145087	128512	
P34392	B*-SA	**935649**	635237	**544579**	**544579**	**544579**	**544579**	**544579**	4
	EA(C)[6]	939855	641514	**544579**	**544579**	**544579**	**544579**	**544579**	4
	EA(nC)[6]	939855	637263	**544579**	**544579**	**544579**	**544579**	**544579**	13
	[5]	944768	**628602**	**544579**	**544579**	**544579**	**544579**	**544579**	
	[1]	1023820	759427	**544579**	**544579**	**544579**	**544579**	**544579**	
	[3]	998733	720858	591027	**544579**	**544579**	**544579**	**544579**	1122
	[4]	1016640	681745	553713	**544579**	**544579**	**544579**	**544579**	
P93791	B*-SA	1782067	1190565	890092	707664	609580	517017	452245	26
	EA(C)[6]	**1754980**	1171190	886038	706820	600986	**501057**	**445748**	32
	EA(nC)[6]	**1754980**	1184630	900388	724758	611029	520868	458389	43
	[5]	1757452	**1169945**	**878493**	718005	**594575**	509041	447974	
	[1]	1851135	1248795	975016	794020	627934	568436	511286	
	[3]	1775099	1192980	899807	**705164**	602613	521806	463707	440
	[4]	1791860	1200157	900798	719880	607955	521168	459233	
U226	B*-SA(1)	**13333**	10666	8084	8000	5333	5333	5333	
	B*-SA(2)	**13333**	**8084**	**6746**	**5332**	**5332**	**4080**	**4080**	1
	EA(C)[6]	**13333**	10666	8084	8000	5333	5333	5333	2
	EA(nC)[6]	**13333**	10666	8084	8000	5333	5333	5333	5
	[5]	**13333**	**8084**	**6746**	**5332**	**5332**	**4080**	**4080**	
	[4]	13416	10750	**6746**	**5332**	**5332**	**4080**	**4080**	
A586710	B*-SA	32545376	**22498601**	17126879	14249746	12811089	11486602	9572169	1
	EA(C)[6]	**31877284**	22973206	**17126866**	**13522326**	**12700205**	**11486599**	**9572166**	3
	EA(nC)[6]	32626780	22973206	17126880	14249800	12811087	11486600	9572170	6
	[5]	32626782	23413604	18838663	14260261	12811087	12573448	10659014	
	[4]	42198943	27785885	21735586	19041307	15071730	14945057	12754584	
G1023	B*-SA(1)	**29765**	**20032**	15302	**14794**	**14794**	**14794**	**14794**	
	B*-SA(2)	**29765**	**20032**	**14913**	**14794**	**14794**	**14794**	**14794**	2
	[5]	31139	21024	15890	**14794**	**14794**	**14794**	**14794**	
F2126	B*-SA	**350030**	**335334**	**335334**	**335334**	**335334**	**335334**	**335334**	1
	[5]	357088	**335334**	**335334**	**335334**	**335334**	**335334**	**335334**	
D281	B*-SA(1)	7102	4870	**3926**	**3926**	**3926**	**3926**	**3926**	
	B*-SA(2)	**7097**	**4862**	**3926**	**3926**	**3926**	**3926**	**3926**	1
	[5]	7432	5003	**3926**	**3926**	**3926**	**3926**	**3926**	
T512505	B*-SA	**10504020**	**10453470**	**5255380**	**5228420**	**5228420**	**5228420**	**5228420**	9
	[5]	10530995	**10453470**	5268868	**5228420**	**5228420**	**5228420**	**5228420**	

REFERENCES

[1] V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "On using rectangle packing for SOC wrapper/TAM co-optimization," *VTS*, pp.253-258, 2002.

[2] K. Chakrabarty, "Test scheduling for core-based system using mixed-integer linear programming," *IEEE TCAD*, pp.1163-1174, 2000.

[3] V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "Test wrapper and test access mechanism co-optimization for System-on-Chip," *ITC*, pp.1023-1032, 2001.

[4] Y. Huang, S. M. Reddy, W.-T. Cheng, and P. Reuter, "Optimal core wrapper width selection and SOC test scheduling based on 3-D bin packing algorithm," *ITC*, pp.74-82, 2002.

[5] W. Zou, S. M. Reddy, I. Pomeranz, and Y. Huang, "SOC test scheduling using simulated annealing," *VTS*, 2003.

[6] Y. Xia, M. Chrzanowska-Jeske, B. Wang, and M. Jeske, "Using a distributed rectangle bin-packing approach for core-based SoC test scheduling with power constraints," *ICCAD*, pp.100-105, 2003.

[7] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, "B*-Trees: A new representation for non-slicing floorplans," *DAC*, pp.458-463, 2000.

[8] Y. Zorian, E. J. Marinissen, and S. Dey, "Testing embedded core-based system chips," *ITC*, pp.130-143, 1998.

[9] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," pp.671-680, *Science*, Vol.220, No.4598, 1983.

[10] E. J. Marinissen, V. Iyengar, and K. Chakrabarty. ITC'02 SoC Test Benchmarks, http://www.extra.research.philips.com/itc02socbenchm/

[11] P.-N. Guo, C.-K. Cheng, and T. Yoshimura, "An O-Tree representation of non-slicing floorplan and its application," *DAC*, pp.268-273, 1999.