Optimizing SOC Test Resources using Dual Sequences

Abstract

In this paper, we propose a new data structure called *dual sequences* to represent SOC test schedules. Dual sequences are used together with a simulated annealing based procedure to optimize the SOC test application time and tester resources. The problems we consider are generation of optimal test schedules for SOCs and minimizing tester memory and test channels. Results of experiments conducted on ITC'02 benchmark SOCs show the effectiveness of the proposed method.

1. Introduction

In the last few years, SOC (system on a chip) design has become the trend in integrated circuit design. Designing a SOC usually involves using pre-defined IP blocks, potentially from different sources, and then adding user defined logic to create a design for a specific application. By taking advantage of the re-usability of the IP blocks, SOC design eliminates the need to design an entire chip from scratch and accelerates time-to-market. As SOC design moves toward mainstream use, the problem of effectively testing the IP blocks (called cores) within the SOC needs to be addressed. Generally speaking, SOC test requires considering the following issues: test access mechanism (TAM) design, core wrapper design, test scheduling, tester memory and tester channels.

TAM is the hardware infrastructure, which transports the test data between the SOC pins and the core wrappers. The core wrapper is a logic block consisting primarily of scan chains placed around the core to isolate the core from its surrounding logic and serve as an interface between the TAM and the core. A number of approaches have been proposed for the core wrapper design [1-6, 11].

SOC test scheduling is the procedure of deciding the test start time of every core so as to obtain a minimum test application time for the SOC under certain constraints, such as TAM width (i.e. the number of SOC pins), power dissipation during test, etc.. Since test scheduling depends on the SOC TAM design and the core wrapper design, SOC test requires co-optimization of the TAM, the core wrapper design and the test schedule. Recently a number of works proposed solutions to this problem. In [3] Marinissen et. al. presented several methods to design TAMs. In [7, 8] Larsson and Peng considered co-optimization of SOC test time and the number of SOC pins under the assumption that a wrapper for each core is given. In [9], Chakrabarty developed an integer linear programming model for minimizing test application time by co-optimization of bandwidth distribution and test bus assignment. Huang et al. [10] formulated the co-optimization problem as a two-dimensional bin packing or rectangle packing problem and solved it by using a best-fit heuristic algorithm. In [16], a SOC test schedule representation called k-tuples was introduced and test scheduling was realized using a greedy algorithm. A similar test schedule representation known as sequence pair was used together with simulated annealing to solve the co-optimization problem [6]. Other works, such as [11-15] have investigated the same problem using specialized heuristic procedures.

In [6] it was shown that test application time for benchmark SOCs using a simulated annealing algorithm were most often shorter than all earlier proposed heuristic solutions and also shorter than an ILP based procedure when the run time of the ILP procedure was limited (to several hours). The SOC schedules were represented in [6] by what are called sequence pair [19].

In this paper, we introduce a simple and effective data structure called *Dual Sequences* (DS) to represent SOC test schedules and use this to obtain optimal SOC test schedules using simulated annealing. Experimental results show that test schedules obtained using DS with simulated annealing are as good as or better than those obtained using sequence pair [6] while the run time of the simulated annealing procedure is greatly reduced. Another problem we consider is minimization of tester memory and tester channels, again using DS to represent test schedules together with simulated annealing.

The paper is organized as follows. In Section 2, we briefly review the features of SOC core test time. SOC test scheduling is introduced in Section 3. In Section 4, the new test schedule representation by dual sequences is presented. In section 5, the simulated annealing algorithm used to obtain optimal SOC test schedules is presented. The problem of minimizing tester memory and tester channels is discussed in Section 6. Experimental results are given in Section 7. Section 8 concludes the paper.

2. The Features of SOC Core Test Time

The core wrapper is the interface between the core and the SOC TAM. It provides several kinds of operation modes, such as normal function, interconnect test, bypass test, etc.. The test time for a core is derived by the following formula [1].

$$T=\{1+\max(Si, So)\} \bullet P + \min(Si, So) \qquad (1)$$

where P is the number of test patterns, and Si (So) denotes the length of the longest wrapper scan input (output) chain for the core. The core test time T is decided by the length of the longest wrapper scan chain. So one goal of the core wrapper design is to shorten the longest wrapper scan chain. For this purpose, balanced wrapper design was proposed [1, 11], which partitions the wrapper scan elements among the wrapper chains to make the length of the wrapper chains as equal as possible. In our method, balanced wrapper design proposed in [11] is used.

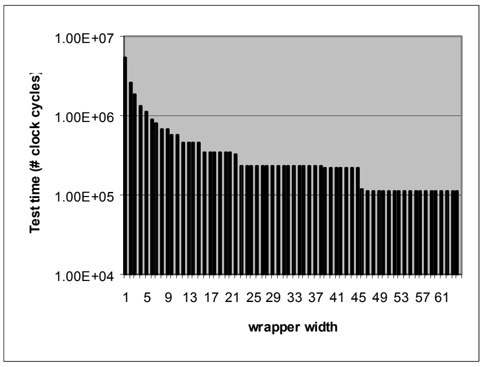


Fig.1 The testing time for the core 6 of p93791

Next we consider the relationship between the core test time and the core wrapper width. Figure 1 shows the test time for core 6 of the ITC'02 benchmark SOC p93791 for different wrapper widths. It can be seen that the test time for the core is a staircase function, which means that there are only some wrapper width values where the core test time changes. These points are called pareto-optimal points [11]. This feature of core test time enables us to restrict consideration of candidate core wrapper widths to the pareto-optimal points as the permissible wrapper widths. If the core wrapper is represented by a rectangle with the width representing the wrapper width and the height representing the core test time, there is a set of candidate rectangles for every core corresponding to the pareto-optimal core wrapper widths. In co-optimizing wrapper design and SOC test time, one of these rectangles is chosen for each core.

3. Problem Formulation

The problem of SOC test scheduling we are considering is stated below.

Given are a SOC with N pins and N_c cores. Each core C_i ($1 \leq i \leq N_c$) has a set of N_i permissible wrapper configurations. Each wrapper configuration is represented by a pair (W_{ij}, $T(W_{ij})$), where W_{ij} stands for the width of the j-th wrapper configuration for core C_i and $T(W_{ij})$ stands for the test time of core C_i with wrapper width W_{ij}. The objective is to pick one wrapper design for each core, determine the mapping from the SOC pins to the core wrapper pins, and set the test start time for each core such that the SOC test application time is minimized.

This problem can be transformed into the well-known two-dimensional bin packing problem, in which the SOC is represented by a bin with width N and the set of N_i SOC wrappers for every core is represented by a set of Ui rectangles with width W_{ij} and height $T(W_{ij})$ [10]. The objective is to choose a rectangle for every core C_i and pack all the rectangles in the bin, such that height of the bin is minimum.

4. Representation of SOC Test Schedules by Dual Sequences

In Figure 2, we illustrate a test schedule for a SOC with six cores. The vertical axis is time and the horizontal axis represents SOC pins. In this schedule, testing of Core 1, Core 2 and Core 3 starts simultaneously at time t = 0. Testing of Core 3 is completed at time $t = t_3$ at which time testing of Core 4 is initiated. Testing of Core 4 is completed at $t = t_4$ at which time testing of Cores 5 and 6 is initiated. The two parts of Core 5, denoted 5_1 and 5_2, indicate that Core 5 is tested through two non-consecutive subsets of TAM pins. Testing of Core 2 is completed at $t = t_2$ and testing of Core 6 is completed at $t = t_6$. At this time, testing of the SOC is completed. As seen from Figure 2, every test schedule corresponds to a rectangle placement in the bin representing the SOC. So the problem of SOC test scheduling can be transformed to the rectangle placement problem.

In this section, a new representation called *Dual Sequences* (DS) is introduced to express the rectangle placement. Earlier, sequence pair was used to represent SOC test schedules in [6, 16]. As shown in the section on experimental results, using DS representation of SOC test schedules reduces the run time and improves the quality of the solutions obtained by using simulated annealing.

The DS for a placement of a set of n rectangles (cores) is a pair of sequences ($\mathcal{R}$, $\mathcal{W}$), in which $\mathcal{R}$ is a sequence of the names of the n rectangles and $\mathcal{W}$ is a sequence of the widths of the n rectangles listed in $\mathcal{R}$. For example, (< R3 R1 R2 R4 >, < 4 2 1 5 >) is a DS from which we can see that the placement is composed of four rectangles with widths 4,2,1 and 5, respectively. Next we discuss how to represent a rectangle placement by a DS and how to obtain the rectangle placement corresponding to a DS.

4.1 DS Extraction from Rectangle Placement

Given a placement of rectangles, the corresponding DS can be obtained by visiting every rectangle in the placement from bottom to top and from left to right. During the visitation, the rectangles we encounter are recorded in R in the order of visiting them and the width corresponding to the discovered rectangles are recorded in $\mathcal{W}$. If a rectangle is split into several sub-rectangles in the placement, the sub-rectangles are merged into one rectangle for representation in ($\mathcal{R}$, $\mathcal{W}$) and its position in $\mathcal{R}$ is decided by the first sub rectangle and the width in $\mathcal{W}$ is the sum of the widths of the sub-rectangles. A rectangle placement corresponding to a SOC test schedule may have split a rectangle corresponding to a core since its wrapper pins are connected to non-consecutive SOC pins. For example consider the rectangle placement in Figure 2, which has six rectangles R1,R2,R3,R4,R5,R6 corresponding to the six cores with widths, say Φ_1, Φ_2, Φ_3, Φ_4, Φ_5, and Φ_6, respectively. The rectangle R5 is divided into two sub-rectangles R5_1 and R5_2. As explained next, by visiting each rectangle within the placement and merging the sub-rectangles, we obtain the Dual Sequences (<R1 R2 R3 R4 R5 R6>, < Φ_1 Φ_2 Φ_3 Φ_4 Φ_5 Φ_6 >). Since testing of R1, R2 and R3 are all scheduled at time zero, they are visited before R4 which is scheduled for testing at t_3. Within the set of rectangles R1, R2, and R3, R1 is visited first since it is left of R2, followed by R2 and then R3. Next R4 is visited. After visiting R4, R5 and R6 are visited but R5 is visited before R6.

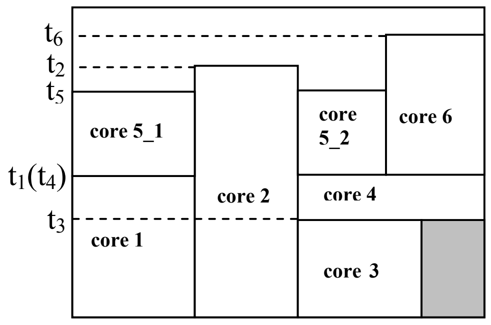


Fig. 2 A SOC test schedule

4.2 Mapping from DS to Placement of Rectangles

To obtain a placement from a DS, a greedy algorithm based on two dimensional bin packing is used. The basic idea of this algorithm is that given the packing sequence $\mathcal{R}$ and the width sequence $\mathcal{W}$ of the packed rectangles, we pick rectangles from $\mathcal{R}$ one at a time in the order of their appearance in $\mathcal{R}$ and pack a selected rectangle at a position which is as low as possible (i.e., we schedule the start of the test of the core corresponding to the rectangle as early as possible). It is important to point the distinction between dual sequences used here and sequence pair used in [6, 16] to represent a bin packing. Given a sequence pair the corresponding bin packing is uniquely defined and is obtained by a longest path procedure run over two graphs derived from the sequence pair [6]. The bin packing corresponding to a given DS is not unique. The sequence R determines the order in which the rectangles are considered and $\mathcal{W}$ restricts the choice of the width (i.e. wrappers) of the rectangle corresponding to the core being packed. Any procedure to pack the rectangles in the order given by $\mathcal{R}$ can be used.

In the proposed method to obtain a rectangle placement from a given DS, a data structure called a *layer* is used, which corresponds to a position where a yet unplaced rectangle can be placed. A layer has two attributes: starting time and width. The starting time is the height of the layer in the bin and the width indicates the space available at this height. For example, in Fig.3 (a), we have four layers, layer 0 to layer 3, which are indicated by the thick dark lines. The start time and the width of the layer can be seen from Fig.3 (a). For example, the start time of layer1 is H1 and the width is (W2-W1).

Before we describe the procedure to obtain the rectangle placement from a DS, we show how the layers change when a new rectangle is added into an existing partial placement.

Given the partial placement in Fig 3(a), suppose a new rectangle, say R4 with width (W2+W4-W3) is placed on layer 2. As shown in Fig 3 (b), R4 occupies layer 1, layer 2 and part of layer 0. Widths of layer 1 and layer 2 are changed to 0 and the width of layer 0 is changed to (W5-W4). A new layer 4 with width (W2+W4-W3) is added. Layer 4 is split into two sub-layers as can be seen in Fig. 3(b).

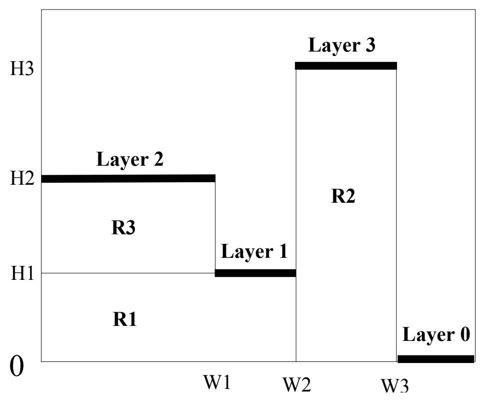
Next we introduce the greedy algorithm for obtaining a rectangle placement from a DS. At the start, there is only one layer whose width is equal to the total TAM width (the number of SOC pins). We pick a rectangle from sequnce $\mathcal{R}$ from left to right , whose width is decided by the corresponding entry in sequence $\mathcal{W}$, and place that rectangle on a layer L, which satisfies the following requirment.

1. The sum of the width of L and the widths of the layers with non-zero widths whose start time is less than or equal to the start time of layer L is greater than or equal to the width of the rectangle being placed.
2. The start time of layer L is the lowest among all layers that satisfy requirment 1.
3. If there is a power constraint, the core placed at that layer will not violate this constraint.

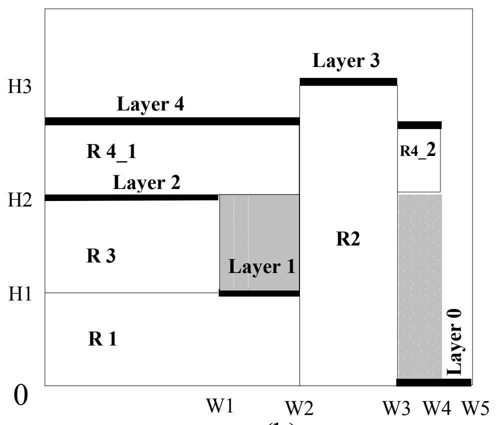
Following the palcement of the rectangle, a new layer with width equal to the width of the just placed rectangle is added to the placement and the widths of the other layers are updated. This procedure is repeated until all the rectangles in sequence $\mathcal{R}$ are packed. It should be pointed out that the procedure proposed above is sub-optimal. One reason for this sub-optimality is that when the width of a layer is reduced it may preclude the use of some packing space. For example the shaded area in Figure 3(b) is not availble for future packing after rectangle R4 is placed.

Compared to test schedule representation using sequence pair used in [6,16] for obtaining SOC test schedules, the search space for DS is of size $(Nc! \times \prod_{i=1}^{Nc} Ki)$ while for the sequence pair representation the search space is of size $((Nc!)^2 \times \prod_{i=1}^{Nc} Ki)$, where Nc is the number of cores in the SOC and Ki is the number of wrapper configurations for core i. The size or the search spaces given above is obtained by computing the number of distinct dual sequences and sequence pair, respectively.

Since the search space using DS representation of rectangle placement is smaller, it leads to a much lower run time for test schedule optimization using simulated annealing. As the experiments on ITC'02 benchmarks reported later show, the optimality of the obtained SOC test schedule is indeed not effected by using dual sequences instead of sequence pair.



(a)



(b)

Fig. 3 The changes in the layers during bin packing

5. Simulated Annealing

Simulated annealing (SA) is a global stochastic optimization algorithm that was first introduced by Kirkpatric et al. [17]. The algorithm begins with an initial solution, and then a neighboring solution is created by perturbing the current solution. If the cost of the neighboring solution is less than that of the current solution, the neighboring solution is accepted; else it is accepted or rejected with some probability. The probability of accepting an inferior solution is a function of a parameter called the temperature. The probability function used is:

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

where ΔC is the change in the cost between the neighboring solution and the current solution and T is the current temperature. At the beginning of the algorithm, the temperature T is large and an inferior solution has a high probability of being accepted. As the optimization progresses, the temperature decreases and there is a lower probability of accepting an inferior solution. The procedure we used to implement the simulated annealing algorithm for finding an optimal SOC test schedule that minimizes the expected test completion time is given below.

Objective: Find an optimal solution Sopt, which makes the cost function C(Sopt) minimum.

Procedure:

```
1. Construct an initial solution $S_{init}$;
2. Let the current solution be $S_{cur}$: = $S_{init}$;
3. Set the initial temperature to T: = $T_{init}$;
4. Set Counter:= 1;
5. While the stopping criteria are not met do begin
6. While T >$T_{final}$ do begin
7.  For i: = 1 to *Niter* do begin
8. Generate a neighboring solution $S_n$ from the
   current solution $S_{cur}$;
9. Compute the change in the cost function
        $\Delta C$ =C($S_n$) - C($S_{cur}$);
10. If $\Delta C \le 0$ then $S_{cur}$ : = $S_n$;
11. Else begin
12. q = random(0,1);
13. If q < $e^{-\frac{\Delta C}{T}}$ then $S_{cur}$ : = $S_n$ ;
14. End
15. End
16. Set new temperature T: = K * T;
17. End
18. Set T:= $T_{new}$;
19. Counter:=Counter+1;
20. End
```

We use the SA algorithm described above to implement the SOC test scheduling based on dual sequences by specifying the parameters of the SA algorithm as follows.

Cost function C: The objective of test scheduling is to reduce the test application time of the SOC. Therefore, the height of the bin where the rectangles are placed is defined as the cost function.

Neighboring solution S_n : The neighboring solution is defined by two types of moves over the dual sequences, given below.

M1: Exchange the position of two randomly chosen rectangles in the first sequence $\mathcal{R}$ (note that $\mathcal{W}$ is also changed to reflect the exchange in $\mathcal{R}$).

M2: Change the width (and hence the height) of a rectangle to another allowed width of the rectangle in the second sequence $\mathcal{W}$ (allowed widths are the pareto-optimal values as discussed earlier).

During the process of optimization, the probabilities of moves M1 and M2 are set to 0.5 each.

Initial solution: The initial solution S_{init} can be set randomly. In order to accelerate the convergence of SA, the test schedule obtained by the heuristic procedure in [13] is used as initial solution in the experiment reported later.

Initial temperature: The initial temperature T_{init} is set to 4000. At the end of each outer loop, temperature T is reset to T_{new} = 4000 + 1000 * Counter.

Other parameters: These parameters include the final temperature T_{final}, the number of iterations *Niter* at every temperature, the stopping criteria and the temperature reduction multiplier *K*. In our implementation, these parameters are set as follows.

(1) T_{final} = 10;

(2)The number of iterations *Niter* at each temperature is set to 400*N_c where N_c is the number of rectangles.

(3)The stopping criteria can be decided by the user. In our experiment, if Counter is larger than 10, the procedure is stopped.

(4) The temperature reduction multiplier K is set to 0.98 when T < 10000; otherwise K = 0.93.

6. Reducing ATE Resources

Automatic Test Equipment (ATE) used in SOC test provides the ability to perform multi-site testing, which allows several copies of a SOC to be tested concurrently. When the number of ATE channels is given, to test a maximum number of SOCs at the same time requires minimization of the TAM width of the SOC while not violating the ATE memory depth constraint (decreasing the TAM width of the SOC will increase the test application time and hence the ATE memory depth requirement). In this section we discuss how the proposed method using DS representation of the test schedules can be used to minimize SOC TAM width as well as the ATE buffer memory depth for a given SOC TAM width.

When an SOC is tested by an ATE, the test channel memory depth required for the SOC is decided by the test data volume. The depth of the test channel memory can be approximated by the SOC test application time (the number of clock cycles) [20]. Therefore, the problem of multi-site SOC test under ATE memory depth constraints can be considered as a problem of reducing the SOC TAM width while the total test application time is fixed. This allows testing of a maximum number of SOCs using a given number of test channels and their buffer memory depth. The SOC multi-site test problem can be solved using the two dimensional bin packing procedure with the width of the bin representing the memory depth constraint and the height of the bin representing the TAM width. We should point out that in the bin packing problem for multi-site testing, a rectangle cannot be divided into several sub-rectangles, which is different from the bin packing problem we discussed before. Dividing rectangles was permitted in the earlier problem since it is not necessary to connect the wrapper pins of a core to adjacent SOC pins. However, in multi-site testing, breaking a rectangle represents interruption of the test of a core, which may not be permitted. A simple way to accommodate the requirement that core tests cannot be interrupted is to require that the new rectangle to be packed must occupy contiguous layers only, thus avoiding division of rectangles.

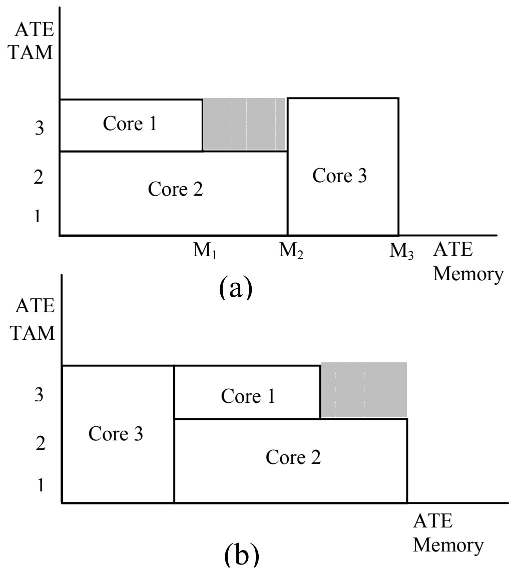


Figure 4 Rectangle packing to optimize ATE resources

Another issue that needs to be considered is illustrated by the rectangle packing shown in Figure 4(a). Figure 4(a) shows the case where the ATE test channels 1 and 2 are used to test core 2 and ATE test channel 3 is used to test core 1. Tests for core 1 occupy M1 bits of memory buffer for tester channel 3 and tests for core 2 occupy M2 bits of memory buffer for channels 1 and 2. Tests for core 3 use all three test channels and hence can only be started after completing the test of core 2. It should be pointed out that the tests are loaded into the buffer and shifted out to the inputs of the device under test. If the tester architecture is such that all test channel buffers are shifted at the same time and each channel has dedicated memory buffer then the buffer bits of channel 3 are don't cares from M1 to M2. However if the tester architecture is such that each test channel is individually controlled, then test channel 3 can be idled after testing core 1 until core 3 test is initiated. In this case the size of the buffer for test channel 3 need only be (M3-M2+M1). The packing shown in Figure 4(b) for the same cores as in Figure 4(a) illustrates the situation where the memory buffer contents for test channel 3 is such that don't cares occur only at the end. In this case after the testing of core 1 is complete test channel 3 can be idled. In general, if the packing is such that all the test channel buffers have don't cares only at the end the ATE memory management is simpler [20,21]. Finally, in some ATE architectures the entire buffer memory can be configured as a single pool of memory that can be dynamically assigned to test channels [21, 22]. For such architectures the total memory requirements for a SOC test is important.

For finding SOC test schedules to minimize the number of ATE test channels given the maximum depth of test channel buffers, we used two different procedures to obtain rectangle packings from dual sequences. The first one is a modified version of the procedure in [20] to obtain rectangle packings such that all the don't cares in the memory buffers are at the end. The second procedure is the one described in the last section with the additional constraint that rectangles are not divided during packing.

7. Experimental Results

The proposed simulated annealing based algorithm is implemented in C++ and executed on a PC with a Pentium IV 1.4GHZ processor and a 512 MB memory. The implemented procedure was applied to ITC'02 benchmark SOCs [18] under the assumption of no power constraint.

The results of applying the proposed method to SOC test scheduling together with the results reported by earlier methods are reported in Table 1. The proposed simulated annealing based procedure was run for ten iterations and the best schedule obtained is reported. The method used is indicated in column 2, where DS indicates the proposed method and the other methods are indicated by the number of the corresponding reference. The remaining columns give the SOC test application time for the number of SOC pins shown as the heading for the column. The entry for the method(s) achieving the best test application time is shown in bold. The method in [6] also used a simulated annealing algorithm with test schedules represented by sequence pair and had achieved better schedules than a heuristic method that also used sequence pair [16]. For this method also we report the schedule obtained from ten iterations of the procedure. It can be seen that for all the benchmark SOCs, the proposed method achieves better or equal SOC test application time compared to [6]. It can also be observed that the proposed method achieves the same or better test application time than all other methods, except in the cases of P93791 with 80 SOC pins and A586710 with 32 SOC pins.

The run times for the proposed simulated annealing based procedure and the earlier procedure using simulated annealing together with sequence pair [6] are given in Table 2. The run times reported for both procedures are for a total of ten iterations of the procedures. From Table 2 it can be seen that using dual sequences instead of sequence pair to represent rectangle placements improves the run time of simulated annealing based procedures. A 2X to 3X improvement in run time is obtained for most designs.

In Tables 3-6 we report the results on ATE tester channels and buffer memory for four circuits for which data of the earlier work [20] is available. In the first column we show the maximum memory allowed per test channel. In the next three columns we show the number of tester channels required to deliver the tests using the two procedures described in the last section and the method of [20], respectively. Procedure DS_1 is the proposed simulated annealing based procedure when the don't care bits in the buffer memory of the test channels are all at the end and DS is the procedure where the don't care bits are allowed to be anywhere in the buffer memory. In the next four columns we give the total ATE memory required to store the test input data. For method DS we report two entries. Under DS_g we report total memory including the don't cares portion and under DS we report the total memory ignoring the don't care portion. For the other two procedures the don't care portions are not included in the totals reported.

From Tables 3-6 it can be seen that the simulated annealing based procedures require the same or smaller number of test channels compared to the heuristic procedure of [20] for all the SOCs considered. It can also be seen that the total memory required is also smaller for the simulated annealing based procedures.

As an example of how reducing the number of ATE test channels helps reduce the cost of SOC test using multi-site testers, consider a tester with 128 test channels. Note that all SOCs under test can receive the test data simultaneously from the same tester channels. However the test responses from each SOC under test require separate test channels. From Table 4 for SOC p22810 with test channel buffer size limited to 640K, we note that using methods DS_1, DS and of [20], the number of test channels needed to apply tests to all SOCs under test is 12, 11 and 13, respectively. However each tested SOC needs the same number of separate test channels to obtain test responses. Thus in this case, the number of SOCs that can be simultaneously tested using a tester with 128 test channels will be 8 using the procedure of [20], 9 using procedure DS_1 and 10 if procedure DS is used. Thus by using procedure DS the number of SOCs tested per unit of time increases by 25% over the number tested if the procedure from [20] is used.

8. Conclusions

A new data structure called *dual sequence* to represent rectangle packings is introduced. Using dual sequences together with simulated annealing procedures to obtain optimal SOC test schedules and to reduce ATE test resources were presented. Experimental results on ITC '02 SOC benchmarks showed that the proposed procedures yield better results than procedures proposed earlier.

References

[1] E. J. Marinissen, S. K. Goel and M. Lousberg, "Wrapper Design for Embedded Core test," pp.911-920, ITC ,2000.

[2] P. Varma and S. Bhatia, “A Structured Test Re-Use Methodology for Core-Based System Chips,” pp. 294 –302, ITC, 1998.
[3] 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. 284-293, ITC, 1998.
[4]P. T. Gonciari, B. M. Al-Hashimi and N.Nicolici , “Addressing Useless Memory in Core-Based System-on-a-Chip Test,” pp. 423-430, VTS, 2002.
[5] E. J. Marinissen, R. Kapur, and Y. Zorian, “On Using IEEE P1500 SECT for Test Plug-n-Play,” pp. 770-777, ITC, 2000.
[6] W. Zou, S. M. Reddy, I. Pomeranz and Y. Huang, “SOC Test Scheduling Using Simulated Annealing,” pp. 325-330 VTS, 2003.
[7] E. Larsson and Z. Peng, “An Integrated System-On-Chip Test Framework,” pp. 138-144, DATE, 2001.
[8] E. Larsson, Z. Peng and G. Carlsson, “The Design and Optimization of SOC Test Solutions,” pp. 523-530, ICCAD, 2001.
[9] K. Chakrabarty, “Design of System-on-Chip Test Access Architectures using Integer Linear Programming,” pp. 127-134, VTS, 2000.
[10] Y. Huang et. al., “Resource Allocation and Test Scheduling for Concurrent Test of Core -Based SOC Design,” pp. 265-270, ATS, 2001.
[11] V. Iyenger, K. Chakrabarty and E. J. Marinssen, “Test Wrapper and Test Access Mechanism Co-Optimization for System-on-Chip,” pp. 1023-1032, ITC, 2001.
[12] V. Iyenger, K. Chakrabarty and E. J. Marinssen, “On Using Rectangle Packing for SOC Wrapper/TAM Co-Optimization,” pp. 253-258, VTS, 2002.
[13] V. Iyengar and K. Chakrabarty and E. J. Marinssen, “Integrated Wrapper/TAM Co-Optimization, Constraint-Driven Test Scheduling, and Tester Data Volume reduction for SOCs,” pp. 685-690, DAC, 2002.
[14] Y. Huang et al., “Optimal Core Wrapper Width Selection and SOC Test Scheduling Based on3-D Bin Packing Algorithm,” pp. 74-82, ITC, 2002.
[15] S. K. Goel and E. J. Marinissen, “Effective and Efficient Test Architecture Design for SOCs,” pp. 529-538, ITC, 2002 .
[16] S. Koranne and V. Iyengar, “On the Use of k-tuples for SoC Test Schedule Representation,” pp. 539-548, ITC, 2002 .
[17] S. Kirkpatrick et al., “Optimization by Simulated Annealing,” pp.671-680, Science, Vol.220, No.4598, 1983.
[18] E. J. Marinissen, V. Iyengar and K. Chakrabarty. ITC2002 SOC Benchmarking initiative, http://www.extra.research.philips.com/itc02socbenchm.
[19] H. Murata, et. al., “VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair,” pp. 1518-1524, IEEE, TCAD, 1996.
[20]V. Iyengar et. al., “Test resource Optimization for multi-Site testing of SOCs under ATE memory Depth constrains, ” pp. 1159-1168, ITC, 2002.
[21] P. T. Gonciari and B. M. Al-Hashimi, “Useless Memory Allocation in System-on-Chip test: Problems and Solutions,” pp. 423-429, VTS, 2002.
[22] J. Bedsole, R. Raina, A. Crouch and M. S. Abadir, “Very Low Cost Tester: Opportunities and Challenges,” pp.738-747, ITC, 2001.

Table 1: Test Application Times for ITC’02 SOC Benchmarks

Benchmark	Method	No. SOC pins						
		32	48	64	80	96	112	128
D695	DS	**41654**	**28161**	**21025**	**16962**	**14310**	**12134**	**10723**
	[6]	41899	28165	21258	17101	**14310**	**12134**	10760
	[11]	41949	28327	21423	17210	16403	13023	12327
	[12]	43723	30317	23021	18459	15698	13415	11604
	[15]	44307	28576	21518	17617	14608	12462	11033
P22810	DS	**433403**	**289332**	**219019**	**178402**	**147944**	**128887**	**110940**
	[6]	438619	293019	219923	180004	151886	132812	112515
	[12]	452639	307780	246150	197293	167256	145417	136941
	[15]	458068	299718	222471	190995	160221	145417	133405
P34392	DS	**960230**	**655607**	**544579**	**544579**	**544579**	**544579**	**544579**
	[6]	965252	657561	**544579**	**544579**	**544579**	**544579**	**544579**
	[12]	1023820	759427	**544579**	**544579**	**544579**	**544579**	**544579**
	[15]	1010821	680411	551778	**544579**	**544579**	**544579**	**544579**
P93791	DS	**1763528**	**1175756**	**887619**	710211	**594054**	**509845**	**445270**
	[6]	1765797	1178397	893892	718005	597182	510516	451472
	[11]	1775099	1192980	899807	**705164**	602613	521806	463707
	[12]	1851135	1248795	975016	794020	627934	568436	511286
	[15]	1791638	1185434	912233	718005	601450	528925	455738
G1023	DS	**30958**	**21233**	**16048**	**14794**	**14794**	**14794**	**14794**
	[6]	31398	21365	16067	**14794**	**14794**	**14794**	**14794**
	[15]	34459	22821	16855	**14794**	**14794**	**14794**	**14794**
U226	DS	**13416**	**10750**	**6746**	**5332**	**5332**	**4080**	**4080**
	[6]	**13416**	**10750**	**6746**	**5332**	**5332**	**4080**	**4080**
	[15]	18663	13331	10665	8084	7999	7999	7999
F2126	DS	**357088**	**335334**	**335334**	**335334**	**335334**	**335334**	**335334**
	[6]	**357088**	**335334**	**335334**	**335334**	**335334**	**335334**	**335334**
	[15]	372125	**335334**	**335334**	**335334**	**335334**	**335334**	**335334**
T512505	DS	**10530995**	**10453470**	**5268868**	**5228420**	**5228420**	**5228420**	**5228420**
	[6]	**10530995**	**10453470**	**5268868**	**5228420**	**5228420**	**5228420**	**5228420**
	[15]	**10530995**	**10453470**	**5268868**	**5228420**	**5228420**	**5228420**	**5228420**
A586710	DS	42198943	**27785885**	**21343768**	**19041307**	**15031300**	**13401034**	**11486601**
	[6]	42198943	**27785885**	21735555	**19041307**	15071700	14709449	12754585
	[15]	**41523868**	28716501	22475033	19048835	15315476	**13401034**	12700205
Q12710	DS	**2222349**	**2222349**	**2222349**	**2222349**	**2222349**	**2222349**	**2222349**
	[6]	**2222349**	**2222349**	**2222349**	**2222349**	**2222349**	**2222349**	**2222349**
	[15]	**2222349**	**2222349**	**2222349**	**2222349**	**2222349**	**2222349**	**2222349**
D281	DS	**7881**	**5329**	**4070**	**3926**	**3926**	**3926**	**3926**
	[6]	7946	5485	**4070**	**3926**	**3926**	**3926**	**3926**
	[15]	8444	6408	5084	3964	**3926**	**3926**	**3926**
H953	DS	**119357**	**119357**	**119357**	**119357**	**119357**	**119357**	**119357**
	[6]	**119357**	**119357**	**119357**	**119357**	**119357**	**119357**	**119357**
	[15]	**119357**	**119357**	**119357**	**119357**	**119357**	**119357**	**119357**

Table2. Run times for the proposed method and method of [6]

Benchmark	Method	No.SOC pins						
		32	48	64	80	96	112	128
D695	DS	43.76	43.51	42.99	42.82	41.82	41.7	40.74
	[6]	52.25	54.19	56.11	56.86	58.93	61.29	67.69
P22810	DS	320.68	320.92	320.29	322.83	322.85	323.13	323.87
	[6]	559.05	580.13	616.26	642.01	642.42	686.25	678.22
P34392	DS	148.59	148.02	146.15	142.15	139.95	137.65	136.1
	[6]	233.06	261.23	279.22	305.05	324.59	335.5	341.98
P93791	DS	433.21	438.96	439.15	441.91	443.44	444.15	446.22
	[6]	686.3	732.37	756.18	782.64	774.37	781.67	783.92
G1023	DS	77.63	76.79	76.02	73.67	71.8	70.3	69.07
	[6]	122.93	128	135.19	143.06	148.98	153.6	156.15
U226	DS	26.64	27.49	26.87	26.83	26.42	26.23	25.95
	[6]	33.34	32.16	33.25	33.68	34.43	35.26	35.44
F2126	DS	12.55	11.84	11.4	11.38	11.41	11.39	11.4
	[6]	16.08	17.24	17.47	17.41	17.42	17.41	17.42
T512505	DS	330.23	357.59	354.7	359.01	362.23	356.65	348.15
	[6]	743.52	943.89	782.03	963.37	1023.21	1038.73	1043.43
A586710	DS	24.08	23.58	23.52	23.31	23.87	22.84	23.56
	[6]	24.42	25.16	26.38	26.13	26.24	25.71	27.39
Q12710	DS	12.28	12.25	12.26	12.28	12.27	12.27	12.26
	[6]	17.49	17.49	17.47	17.51	17.47	17.48	17.49
D281	DS	27.85	27.36	26.77	26.28	25.88	25.37	25.15
	[6]	37.01	39.53	41.31	42.83	44.28	45.14	45.44
H953	DS	28.54	27.05	26.34	25.91	25.87	25.81	25.78
	[6]	49.42	52.46	53.3	53.55	53.51	53.18	53.25

Table 3. SOC g1023: TAM width and ATE memory

Memory	TAM width			Total ATE memory			
	DS_1	DS	[20]	DS_1	DSg	DS	[20]
32K	16	16	18	495679	496080	495639	511464
40K	13	13	15	495736	493047	492948	505107
48K	11	11	13	493358	494391	493717	507696
56K	9	9	11	492936	493713	492087	515956
64K	8	8	10	493475	493404	493124	514380
72K	7	7	9	490096	490086	488965	516732
80K	7	7	8	488759	488922	488910	514538
88K	6	6	7	489263	490291	490020	506660
96K	6	6	6	488639	490363	489495	507849
104K	5	5	5	490060	489592	489142	501840
112K	5	5	5	488244	490029	488886	500262
120K	5	4	5	487714	489018	487870	500209
128K	4	4	4	488271	488740	488740	497167

Table 4. SOC p22810: TAM width and ATE memory

Memory	TAM width			Total ATE memory			
	DS_1	DS	[20]	DS_1	DSg	DS	[20]
256K	30	28	30	7099492	7095609	6993356	7404961
320K	23	22	25	7027732	7003552	6967561	7134483
384K	19	19	21	6996904	7023324	6955179	7255983
448K	17	16	18	7086139	6994720	6913817	7326446
512K	14	14	16	6926695	7096052	7008133	7350768
576K	13	12	14	6862197	6955836	6935482	7390568
640K	12	11	13	6971280	6915493	6833044	7441084
704K	11	10	11	6804771	6886110	6820335	7234211
768K	10	9	11	6865817	6892560	6832015	7564350
832K	9	9	10	6839742	6917345	6835019	7601117
896K	9	8	10	6797761	6932763	6823407	7784192
960K	8	8	9	6837474	6940258	6843692	7642819
1M	7	7	8	6935563	6897983	6810674	7245774

Table 5. SOC p93791: TAM width and ATE memory

Memory	TAM width			Total ATE memory			
	DS_1	DS	[20]	DS_1	DSg	DS	[20]
1.00M	29	29	30	28801632	29165376	28701704	30569666
1.256M	23	23	23	28365553	28977417	28673994	28853177
1.512M	19	19	20	28416635	28749065	28485192	29587103
1.768M	16	16	17	28350143	28881393	28598386	30209460
2.000M	14	14	15	28448141	28576775	28221882	30570183
2.256M	13	13	13	28235620	28732954	28267254	29108758
2.512M	12	12	12	28304322	28902346	28223225	30385045
2.768M	11	11	11	28184680	28196995	28026911	29499548
3.000M	10	10	10	28157723	28665927	28160766	29635431
3.256M	9	9	9	28227717	28570811	28100678	29121214
3.512M	8	8	8	28056204	28417779	28194724	28853489
3.768M	8	8	8	28211308	28699383	28096145	29038354
4.000M	7	7	7	28301285	28420266	28189282	29096196

Table 6. SOC p34392: TAM width and ATE memory

Memory	TAM width			Total ATE memory			
	DS_1	DS	[20]	DS_1	DSg	DS	[20]
768K	21	21	23	15499573	15572458	15509657	15975513
896K	18	18	20	16037390	15381530	15348449	16676762
1.00M	16	15	16	15506240	15221266	15194202	15645989
1.128M	14	14	15	15399516	15267172	15213829	16227655
1.256M	13	12	14	15286752	15224186	15152249	15961051
1.384M	11	11	13	15336206	15345128	15268571	16713779
1.512M	11	10	12	15239519	15228804	15179270	15910317
1.640M	10	10	11	15177894	15304504	15165987	15474763
1.768M	9	9	10	15132399	15179765	15097375	15890652
1.896M	8	8	10	15124570	15176594	15127484	16330357
2.000M	8	8	9	15114833	15180006	15080645	16588577