On Test Scheduling for Core-Based SOCs

Sandeep Koranne*
ED&T/Test, Philips Research Labs, Eindhoven. The Netherlands.
sandeep.koranne@ieee.org

Abstract

We present a mathematical model for the problem of scheduling tests for core-based system-on-chip (SOC) VLSI designs. Given a set of tests for each core in the SOC and a set of test resources (e.g., test access mechanisms (TAM)), we determine the test plan for the application of the tests to the SOC. Test planning in this paper refers to the combined activities of test access architecture partitioning and test scheduling. These activities must be performed in conjunction as the choice of the test access architecture influences the test schedule. We justify the formulation of test scheduling w.r.t. minimum average completion time criterion as compared to minimum makespan. We show that then the problem of scheduling tests on TAMs can be mapped onto a graph theoretic problem which has a polynomial time optimal solution. We have implemented our algorithm as a test planner tool TPLAN. *We present the theoretical analysis of our approach in this paper, and compare our results against those published earlier using integer linear programming techniques with encouraging results.*

1 Introduction

The widening design productivity gap between VLSI system capabilities and design engineering capability, in a limited time to market scenario, has prompted many design groups to adopt a policy of core-based design reuse at the system level [11]. Typical cores include CPUs like MIPS and ARM, network controllers, embedded memories, DSP cores and associated peripherals like the IEEE 1394 Firewire and UARTs. But as has been noted in [18] reusability of the design alone is not sufficient as the verification and test generation effort now dominate the design cycle. Reusability of tests is crucial to reducing total design time. This raises the problem of test knowledge transfer and test application to embedded cores. The proposed IEEE P1500 (SECT) [9] standard provides facilities for test knowledge transfer using Core Test Language (CTL) and has suggested various test wrapper mechanisms for providing test access to embedded cores. In this paper we focus on a related problem, i.e., optimization of test access mechanism in conjunction with efficient scheduling of tests on system level.

Physical test application to embedded cores can be thought of as comprising of two sub-problems, i.e., the problem of providing test access to the core level test pins from the IC level pins and the problem of efficiently scheduling the tests so as to reduce the total test application time (which determines the test cost). In this paper we focus on the latter problem. Previous work on the first problem, i.e., *test access mechanisms (TAMs)*, is reported in [7], [14], [17] and [4]. In [7], Ghosh *et al.* describe a method of testing embedded cores by means of transparent access paths. The paper by Varma and Bhatia [17] describes a method using a dedicated test bus for the test access to embedded cores. In [14] a scalable and flexible test access mechanism (the TESTRAIL) is described by Marinissen *et al*. Chakrabarty [4] proposes TAM optimization using the integer linear programming approach. Bus-based TAM designs are currently popular as they are flexible and can (with minor modifications) support all the required functionality. In this paper we have assumed that a test bus similar to the one proposed in [14] will be used to transport the tests to embedded cores.

Recent work in the area of test scheduling can be found in [16], [5], [10] and [12]. In [16], Sugihara *et al.* address the problem of selecting a test set for each core from a set of precomputed tests provided by the core provider and scheduling these tests in order to minimize the test time of the complete system. The most restrictive assumption in this paper is that there is only one test access bus at the system level. Also the assumption that the core provider can provide multiple test sets is not realistic in light of the basic foundation of design reuse, i.e., reducing the design time.

In [5] Chakrabarty has analyzed the test scheduling problem by reducing it to open-shop scheduling with m-processors. The *finish time* of the schedule is the latest completion time of the individual processor schedule. The

*An abstract version of this paper appeared in the Conference of the European Chapter on Combinatorial Optimization, Bonn, May 2001.

length of the job is taken as the amount of time for the test to execute. The objective is to minimize the finish time and solutions using mixed integer linear programming (MILP) techniques have been proposed. The idea has been further extended by Iyengar, Chakrabarty and Marinissen in [10] to include TAM optimization with core-level wrapper optimizations. We compare our results to the one published in [10] in Section 5.1.

Test scheduling with power and test resource constraints is analyzed by [12], where Larsson presents an integrated SOC test framework by analyzing the problem of test access mechanism design alongwith test scheduling. It is one of the first papers that considers all the aspects of core-based SOC testing in an unified manner. The limitation of this approach is the use of constant test times for tests, which in practice would depend on the width of the TAM they are executed on. Also to simplify the problem, the author has assumed linear dependence between test time, test power consumption and the degree of parallelism, which is in fact the number of internal scan chains connected without serialization to the TAM.

In this paper we describe our experiments with an ILP (integer linear programming) formulation and improve upon previous results. We show that even using this improved formulation the time taken by the ILP solver to solve medium-sized is unacceptably large. The theoretical analysis of the reasons for the hardness of this problem are also discussed. Hence, we motivate the search for alternative optimality criteria which satisfy both the requirements i.e., of near-optimal performance bounds and modest computational complexity. There are several significant differences between our approach and previous art. Almost all previously published approaches on the test scheduling problem have assumed that the processing power of the test resource is uniform for all the TAMs. We depart from this assumption and do not apply this restriction; larger and more complex cores will in practice have a wider TAM. Secondly, we do not separate the test scheduling problem from the TAM partitioning problem. We argue that if we separate scheduling from TAM partitioning, then the test time calculation for a particular partition (which is used as a measure of the *fitness* of a partition) can be misleading, and hence constitute a sub-optimal solution. We devise a novel method of calculating the original partition of the total TAM width using a *projection*-based method. Our method subsumes the problem of constrained scheduling. In constrained scheduling some of the tests can only be executed on a subset of the test resources. A strong case can be made for constrained scheduling for mixed-signal test bus like the IEEE 1149.4 mixed-signal test bus. When using an ILP formulation additional constraints have to be added to the formulation increasing its complexity. We handle the constrained scheduling problem elegantly.

The rest of the paper is structured as follows. In the next section we describe our notation. In Section 3 we analyse the problem of partitioning available TAM bits efficiently to scan chains of the core. We also describe an efficient method to *meta-partition* the top-level TAM to arrive at good candidate sets for the scheduling step. In Section 4 we present our test planning algorithm by a reduction to minimum weight perfect bipartite graph matching. Our method runs in time $O(n^3)$ for a constant number of test access mechanisms, where n is the number of tests to be scheduled. In Section 5 we discuss our test plan model with the help of an example. We describe the tool we have developed for test planning TPLAN and give the results on some synthetic yet realistic SOCs. We compare the performance of our method with those of ILP based techniques as presented in [10] in Section 5.1. We conclude in Section 6.

2 Problem Statement and Notation

Problem 1 (Core-Based SOC Test Scheduling $\Sigma_{\pi W}$): *Given N_C cores each with a test T, where test T_i takes time t_{ij} on a TAM of width j bits, and a constraint W on the width of the top-level TAM; calculate the partition(πW) of W into m sub-TAMs $w_1, w_2, \ldots, w_m$ where $m \leq N_C$, and a schedule Σ of the N_C tests on πW, which has minimum completion time. The number of sub-TAMs m is not an input to the problem, and must be calculated. By a method of* restriction *it is easy to show that $\Sigma_{\pi W}$ is strongly $\mathcal{NP}$-hard.*

In the test application model the processing requirement for a test can be thought of as the demand of a p_i number bits at the test pin. We associate with T_i a *bitwidth* denoted ϕ_i, which is the number of core test ports involved in T_i. Formally, the width of a TAM over which there is no further decrease in the length of the longest scan chain of a core is called the *bitwidth* of a core. An example for a core `s38584` is shown in Fig. 1. Also associated with T_i is a set of TAMs μ_i. In our model μ_i is the set of TAMs responsible for transporting the test bits for the test T_i. We consider all μ_i as singleton sets or dedicated machines. We use m_i to refer to the *width* of the TAM μ_i. We associate each TAM with a job-dependent *speed* denoted s_{ij}. The processing time for the test T_i on machine j is denoted by t_{ij} and can be calculated as $\frac{p_i}{s_{ij}}$. Although t_{ij} is monotonically non-increasing w.r.t. to m_j, the relationship is not a linear one as assumed in [12]. In Section 3 we investigate the relationship between the width of the TAM μ_i (m_j) and its speed (s_{ij}). Along the lines of [2] we denote the test planning problem as $\Sigma = \alpha|\beta|\gamma$, where Σ denotes the schedule, α refers to the machine environment, β to the test characteristic and γ denotes the optimality criterion. We discuss these below:

1. Machine environment: We do not consider the machines to be identical to each other in terms of processing power. The width of the TAM μ can be thought of as *processing power* of machine μ. The machine environment is thus $\alpha = R$ (unrelated parallel machines).

2. Test (job) characteristics (β): Since for some tests (e.g., BIST) a preemptive test schedule would make the test control logic hardware expensive in terms of area we do not consider test preemption. In this paper we also do not consider precedence constraints, or power constraints. Hence, $\beta = \{\ \}$.

3. Optimality criterion (γ): Traditionally the optimality criterion for embedded core-based testing has been minimizing the finish time of the last test. This is referred to as the *makespan* of the schedule. The makespan is also the maximum time any TAM is active executing a test. Hence, the problem becomes a *min-max* type problem with integrality constraints. The problem of minimizing the makespan for parallel unrelated machines is denoted $R||C_{\max}$, and has been investigated in detail in the context of multiprocessor scheduling. In this paper we have used the average completion time optimality criterion. We justify the choice below.

In 1976, Horowitz and Sahni [8] gave algorithms for optimal solutions to $R||C_{\max}$ using dynamic programming, but the run time of their algorithm was exponential. For the case of fixed m they have shown that there exists a fully polynomial-time approximation scheme to solve $mR||C_{\max}$. They proved that for any $\epsilon > 0$, an ϵ-approximation algorithm can be computed in $O(nm(nm\epsilon)^{m-1})$ time, which is polynomial in both n and $1/\epsilon$ if m is a constant. But the algorithm is still expensive in term of runtime for practical cases. In another fundamental paper Lenstra, Shmoys and Tardos [13] proved that no $(\frac{3}{2} - \epsilon)$ approximation algorithm can exist for any $\epsilon > 0$, for $R||C_{\max}$ unless $\mathcal{P} = \mathcal{NP}$. Previous approaches to solving this problem in the context of test scheduling have not had much success. In [10] the ILP solver is not able to solve for values of $n = 10, m > 2$. In Section 5 we improve upon these results, but only marginally as the maximum problem instance that is solvable using our formulation, in a reasonable time is, $n = 10, m = 6$. State-of-art SOCs contain upto 40 cores and 64 bit wide test bus at IC level. Each of these 40 cores may have sub-cores which can constitute an schedulable entity in itself. Hence, an alternative optimality criterion, which is polynomial in the number of schedulable entities and the number of test resource is needed. We present such an optimality criterion in the form of minimum average completion time schedules, denoted $R||\Sigma C_j$. In addition to being an useful heuristic for the makespan problem, the minimum average completion time criterion can also be justified by noting that the makespan criterion optimizes the schedule assuming all tests will pass. For sequential testing of devices with an abort-on-fail strategy, $R||\Sigma C_j$ schedules will take less amortized time for a large number of devices.

3 Test Access Mechanism Partitioning

Test access mechanism planning is also referred to as TAM partitioning as mostly the top level TAM width of the system is constrained due to pin count limitations. In order to achieve good schedules the TAM partition must be chosen well. The test access mechanism partitioning problem can be divided into two sub-problems. The first one is the local scan chain partitioning problem per core, and the other is the global TAM meta-partitioning problem. We do not consider modification to the core to reduce the test time as this would imply regeneration of test vectors. Hence, in our model the length of the scan chains are fixed *a-priori*. We discuss the core level scan chain partitioning problem first.

3.1 Partitioning of Scan Chains at Core Level

In many cases the test bitwidth requirement of the core test is much larger than the available width of the TAM. In such cases some of the core test ports must be accessed in serial fashion. This problem of assigning the available TAM width to the core test ports has been referred to as the *scan chain partition problem (PSC)* in [15] and shown to be $\mathcal{NP}$-hard and several heuristic solutions for the problem have been described e.g., Best Fit Decreasing and MultiFit.

3.2 Top Level TAM Partitioning

Problem 2 (Top-level TAM Partitioning πW)*: Given a SOC with N_C cores and total M TAM pins; divide M into N_C bits $m_1, m_2, \ldots, m_{N_C}$, such that a schedule on the tests using these sets of TAM bits is close to optimal.*

The idea is not to assign core i a TAM of width m_i (static assignment), but to create a sufficiently *varied* TAM partition, such that a test scheduler can choose a good solution. As an example consider $M = 4$ and $N_C = 3$, then there can exist only 1 partition, i.e., {1,1,2}. Clearly, the number of such potential partitions is exponential (even considering only unique partition sets removing cyclicly isomorphic partitions).The problem is hard as the *fitness* of a particular partition depends on the schedule which in turn depends on the partition. Estimating the fitness of a particular partition without taking into account scheduling can lead to suboptimal solutions. But this approach is popular as it is simple to implement both in software and in hardware. We need a method to arrive at a good partition looking at static information. This partition will then be used by the scheduling algorithm to assign a TAM of certain width to a core for

a test. We present an algorithm for calculating the initial partition based on the total number of test bits that need to be transported for each core. This initial partition will be used by the scheduler to arrive at an optimal schedule for the complete test application.

Proposed Method for non-Bitwidth Limited Tests:

Let the expected bitwidth for each core be $o_1, o_2, \ldots, o_{N_C}$, where

$$o_i \quad = \quad \forall\, tests\ i \ :\ \max \quad \forall\, operations\ j \in i\ \ \phi_{ij} \quad (1)$$

The partition of M is given by:

$$m_i = 1 + \frac{o_i}{\sum_{j=1}^{N_C} o_j} \cdot (M - N_C) \quad (2)$$

In effect we are *projecting* the ratio of the bitwidth requirement for core i to the total bitwidth of the SOC, onto M. In Eqn. 2 we have assigned each core at least 1 bit. This works well in principal for non-bitwidth limited tests (e.g, BIST).

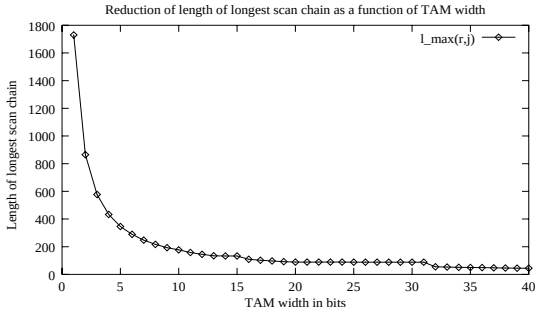

Figure 1. Plot of $l_{\max}(j, r)$ for j=s58384

4 An Optimal Test Scheduling Formulation

In the previous sections we have seen how to partition the available TAM width optimally at core level and how to arrive at a partition set for the top level TAM bits. In this section we describe an algorithm for deriving the optimal test plan for all the tests on the available set of TAMs. We denote by p_{ij} the number of bits flowing for the j^{th} operation of test i (we uniquely rename all tests so that tests per core are combined into a larger test set). We do not derive the starting time for each test, but rather derive the order in which each TAM executes the tests (this is useful in our discussion and no information is lost as the starting time for each test can be calculated from this ordering).

Let J_{ki} be the k-th from last test to run on TAM i. Let there be a total of l_i tests on TAM i. Let C_j denote the completion time for test j. The total test time for the complete system is given by:

$$\sum_j C_j = \sum_{i=1}^{N_C} \sum_{k=1}^{l_i} C_{J_{ik}} \quad (3)$$

Since the completion time for a test is the sum of the processing time of all tests that are executed before it, and since J_{ki} denotes the k-th from *last* test to run, we get:

$$\begin{aligned} \sum_j C_j &= \sum_{i=1}^{N_C} \sum_{k=1}^{l_i} \sum_{x=k}^{l_i} t_{i,J_{xi}} \\ &= \sum_{i=1}^{N_C} \sum_{k=1}^{l_i} k t_{i,J_{ik}} \end{aligned} \quad (4)$$

where $t_{m,J_{ijk}}$ denotes the completion time for operation O_{ij} with bits p_{ij} if scheduled on the k-th from last position on TAM m (of width m). As can be seen from Eqn. 4, the k-th from the last test contributes exactly k times its processing time to the total cost. Using the result obtained above we can formulate the planning problem as a graph theoretic one. To solve the problem we reduce it to an assignment problem. We assign operation O_{ij} (operation j of test i) to position k on TAM m. The cost of assigning test O_{ij} to (k, m) is kp_{ijm} where $p_{ijm} = p_{ij}/s_m$, p_{ij} is the number of bits to be transported for operation O_{ij} and s_m is the effective speed (taking into account the slowdown because of serialization) of TAM m. A polynomial time solution for this problem was presented in [3] with time complexity $O(mn^3)$, where m is the number of machines and n is the number of jobs; we present the idea of the algorithm below.

4.1 Reduction to Minimum Weight Perfect Bipartite Graph Matching

Let $G = (V, E)$ be a bipartite graph in which $V = n \times nm$, where $n = set\ of\ tests$ and $m = set\ of\ TAMs$. There is an edge e_{ijk} between each element of set $A = n$ and each element of set $B = nm$ of weight kp_{ij}, which can be read as the cost of scheduling test i on the j-th TAM at the k-th position from end. See Fig. 2 for an example (not all edges are drawn for clarity). For brevity we omit the division of tests into operations.

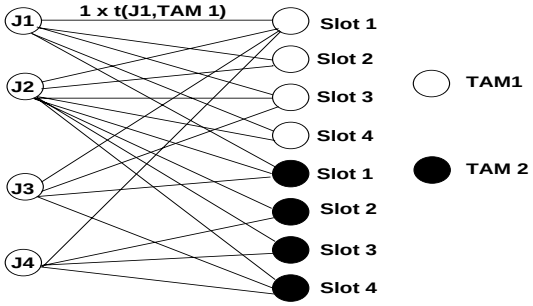

Figure 2. Reduction to Bipartite Matching

A feasible schedule is one that assigns each test to some position of a TAM. This corresponds to the *perfect matching* problem. An optimal schedule corresponds to a matching with minimum weight. Hence, the minimum weight perfect

matching corresponds to the optimal test plan for the test application problem.

The input to the algorithm are the number of tests, the number of TAMs, the tables containing the number of patterns per test, and the length of the longest scan chain per test per TAM width. We first create two partitions A and B, which can witness the bipartite property of the graph G. The calculation of the weight of each edge between a vertex pair $(u, v) : u \in A, v \in B$ is done as explained above. For a constant number of test access mechanisms (most common in practice) the time complexity is $O(n^3)$. The bipartite graph matching formulation has several advantages over conventional integer programming approaches besides the computational tractability. The bipartite formulation subsumes the resource conflict for TAMs (for any test resource considered for scheduling). As an example consider a constraint graph G'. A resource conflict graph G'' can be constructed as $G'' = G^* - G'$, where G^* denotes the complete graph. Given the bipartite graph $G = (V, E)$, we modify the weight as follows:

$$\forall e \in E : w(e) = \infty | \text{ if } e \in G'' \tag{5}$$

No other change is required to the formulation. If there exists a feasible assignment then a perfect matching of weight less than ∞ will be found by the algorithm.

5 Proposed SOC Test Planning Flow

We now present out test planning flow with an example. To present our approach in a simplified manner we present a didactic example. Given a system with N_C cores and T tests we have to devise a schedule for each of the tests on the TAMs, and at the same time come up with an optimal partition of the TAMs. We assume that the maximum width of the TAM at IC level is fixed at W. We then have :$\forall i \sum_{\mu_{ijm}} m \leq W$. *Note:* this is an inequality as compared to the strict equality that has been used in some of the previous papers. When solving the mathematical program with integrality constraints, this inequality can speed the computation substantially. Each core gets a *slice* of the total available TAM width (W). The speed of the TAM slice μ_j (read as speed of the TAM j responsible for transporting the test bits of test i) can be calculated from [1] as :

$$s_{ij} = \frac{p_i}{l_{\max}(j, m_j)(p_i + 1) + p_i} \tag{6}$$

where m_x denotes the width of TAM μ_x, p_x denotes the number of patterns for operation x. In the sequel of this paper we assume that a table containing information about all cores in the SOC is given. This table contains information regarding the number of patterns per core, the number of functional ports of a core, the number of scan chains, flip-flops etc.

We have implemented our method in the form of a Test Planner Tool TPLAN with the C++ language. The run-times correspond to the execution of TPLAN on a Pentium III machine running Linux with 512 MB of RAM. The software was compiled using GNU `g++` with the -O2 option.

5.1 Comparison to ILP Based Approaches

We formulated the makespan minimization problem as an ILP and solved it using CPLEX (available from `www.ilog.com`). Our results are better than those published earlier in [10]. The design considered is mentioned as SOC $S2$ in [6]. We formulated the problem using binary decision variables and a lookup table for implementing the function $l_{\max}(j, m)$. The input to CPLEX was W the number of top level test pins, and B, the number of TAMs. The ILP formulation computed the optimal partition of W into B partitions and calculated the optimal schedule. In [10] the solver (*lpsolve*) was halted after 180 minutes of runtime for $m = 3, W \geq 48$.We improve upon the schedules in all cases and as can be seen from Table 1 the runtime of our ILP formulation is significantly less than the one given in [10]. In most cases the runtime was few seconds. But in subsequent experiments even CPLEX was not able to solve instances for $W = 64, n = 32, m = 7$ in 15 hours of runtime. This justifies our use of minimum average completion time optimality criterion as a heuristic. We compared the makespan length for these two methods; the results are given in Table 1. The column ϵ denotes the approximation factor of the average completion time schedule over the optimum makespan schedule found using ILP.

Table 1. Comparison of schedules for $S2$

	Results of [10]		Our Experimental Results			
	lpsolve		CPLEX		TPLAN	
W	$R\|\|C_{\max}$ Sch.	Time (min)	$R\|\|C_{\max}$ Sch.	Time (min)	$R\|\|\Sigma C_j$ Sch.	ϵ ratio
16	42476	16.7	42293	0.1	43447	1.21
20	34493	26.0	33917	0.1	35760	1.23
24	28862	45.8	28305	0.5	30521	1.30
28	26964	57.6	24021	0.4	25583	1.33
32	22945	78.6	21474	0.6	22955	1.33
36	19858	118.0	19034	0.2	20646	1.27
40	19573	163.6	17612	0.1	18679	1.11
44	–	–	15811	1.2	17280	1.19
48	18999	$180^{\dagger}$	14317	0.6	15982	1.25
52	16928	$180^{\dagger}$	13482	1.0	15230	1.29
56	15694	$180^{\dagger}$	12466	0.8	13444	1.14
60	15694	$180^{\dagger}$	11432	$5.0^{\dagger}$	12902	1.23
64	15694	$180^{\dagger}$	10877	$5.0^{\dagger}$	11520	1.28

5.2 Analysis of results

From the results given in Table 1 we can see that the makespan obtained from solving the minimum average

completion time schedule is close to the ones obtained from the ILP formulation. In many cases the results are even better (e.g., the best previously published schedule for SOC $S2$ obtained using non-enumerative methods was 15694 [10], which was obtained by using *lpsolve* which had to be halted after 180 minutes, we obtain a schedule of 13444 using our meta-partition algorithm and bipartite matching in less than a second). Although we have obtained better schedules using CPLEX, even that method is not scalable and does not work for instance size as seen in state-of-art SOCs. It must be remembered that even though the values of ϵ for some W may seem high (e.g, 1.33), these values were calculated for instances where the ILP solver was able to produce a schedule. For larger instances like $W = 64, n = 32, m = 10$, the ILP solver is not able to compute the schedule (except for trivial feasible solutions) in a reasonable time. This is due to the fact that an ILP solver cannot continue to optimality because of the hardness of the problem.

6 Conclusion

We have presented a polynomial time method to solve the test planning problem for core-based SOCs by reducing it to the minimum weight perfect bipartite graph matching problem. We justify the use of minimum average completion time method as an optimality criterion as it is efficiently computable, its use can be justified at scheduling wafer level tests with an abort-on-fail strategy, and as we shown with experimental data, the approximate makespan schedule found using minimum average completion time methods can be better than the schedule obtained the computationally intractable min-max ILP for large SOCs. Theoretical analysis for the reduction as well as an efficient algorithm to compute such optimal test plans based only on information from the core test sheets is presented. Our method provides a global model by including the TAM partitioning problem within the search space. The top level meta-partition problem for creating candidate TAM sets for scheduling is also discussed and we present an efficient method of arriving at such sets. We have implemented our method as a test planner tool TPLAN. Comparison of our method (TPLAN) with integer linear programming techniques show that for large SOCs TPLAN can come up with good approximate schedules. In this paper we have considered the basic problem of test scheduling with no conflicts, precedence and power constraints. These are future research directions.

References

[1] J. Aerts and E. J. Marinissen. "Scan Chain Design for Test Time Reduction in Core-Based ICs". In *Proc. IEEE Intl. Test Conf. (ITC)*, pages 448–457, October 1998.

[2] P. Brucker. *Scheduling Algorithms*. Springer-Verlag, 2nd edition, 1997.

[3] J. L. Bruno, E. G. Coffman, and R. Sethi. "Scheduling independent tasks to reduce mean finishing time". *Comm. of the ACM*, 1974.

[4] K. Chakrabarty. "Design of System-on-a-Chip Test Access Architectures Using Integer Linear Programming". In *Proc. IEEE VLSI Test Symposium (VTS)*, pages 127–134, April 2000.

[5] K. Chakrabarty. "Test Scheduling for Core-Based Systems Using Mixed-Integer Linear Programming". *IEEE Trans. on CAD*, 19(10):1163–1174, October 2000.

[6] K. Chakrabarty. "Optimal Test Access Architectures for System-on-a-Chip". *ACM Tran. Design Automation of Electronic Systems*, January 2001.

[7] I. Ghosh, N. K. Jha, and S. Dey. "Low Overhead Design for Testability and Test Generation Technique for Core-Based Systems-on-a-Chip". *IEEE Trans. on CAD*, 18(11):1661, November 1999.

[8] E. Horowitz and S. Sahni. "Exact and Approximate Algorithms for Scheduling Nonidentical Procesors". *Journal of the ACM*, 23:317–327, 1976.

[9] IEEE P1500 Web Site. http://grouper.ieee.org/groups/1500/.

[10] V. Iyengar, K. Chakrabarty, and E. J. Marinissen. "Test Wrapper and Test Access Mechanism Co-Optimization for System-on-a-Chip". In *Proc. IEEE Intl. Test Conf. (ITC)*, pages –, October 2001.

[11] M. Keating and P. Bricaud. *Reuse Methodology Manual For System-on-Chip Designs*. Kluwer Academic Publishers, 1998.

[12] E. Larsson and Z. Peng. "An Integrated System-on-Chip Test Framework". In *Proc. Design, Automation, and Test in Europe (DATE)*, pages 138–144, March 2001.

[13] J. K. Lenstra, D. B. Shmoys, and E. Tardos. "Approximation algorithms for scheduling unrelated parallel machines". In *Proc. 28^{th} Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 217–224, 1987.

[14] 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". In *Proc. IEEE Intl. Test Conf. (ITC)*, pages 284–293, October 1998.

[15] E. J. Marinissen, S. K. Goel, and M. Lousberg. "Wrapper Design for Embedded Core Test". In *Proc. IEEE Intl. Test Conf. (ITC)*, pages 911–920, October 2000.

[16] M. Sugihara, H. Date, and H. Yasuura. "A Novel Test Methodology for Core-Based System LSIs and a Testing Time Minimization Problem". In *Proc. IEEE Intl. Test Conf. (ITC)*, pages 465–472, October 1998.

[17] P. Varma and S. Bhatia. "A Structured Test Re-Use Methodology for Systems on Silicon". In *Digest of Papers of IEEE Intl. Workshop on Testing Embedded Core-Based Systems (TECS)*, pages 3.1–1–8, November 1997.

[18] Y. Zorian, E. J. Marinissen, and S. Dey. "Testing Embedded-Core Based System Chips". In *Proc. IEEE Intl. Test Conf. (ITC)*, pages 130–143, October 1998.