Test Scheduling for Core-Based Systems[1]

Krishnendu Chakrabarty
Department of Electrical & Computer Engineering
Duke University, Durham, NC 27708

Abstract

We present optimal solutions to the test scheduling problem for core-based systems. We show that test scheduling is equivalent to the m-processor open-shop scheduling problem and is therefore NP-complete. However, a commonly-encountered instance of this problem ($m = 2$) can be solved in polynomial time. For the general case ($m > 2$), we present a mixed-integer linear programming (MILP) model for optimal scheduling and apply it to a representative core-based system using an MILP solver. We also extend the MILP model to allow optimal test set selection from a set of alternatives. Finally, we present an efficient heuristic algorithm for handling larger systems for which the MILP model may be infeasible.

1 Introduction

Embedded cores are being increasingly used in the design of large systems-on-a-chip [6]. In order to reduce cost and time-to-market, the testing time for a core-based system must be minimized by scheduling tests for the cores, and by designing an appropriate test access architecture. Most of the previous research on core testing at the system level has focussed only on the design of efficient test access architectures [3, 6]. Test scheduling for core-based systems has received much less attention. Given a set of tasks (test sets for the cores), a set of test resources and a test access architecture, test scheduling refers to the problem of determining start times for the tasks such that the total test application time is minimized.

A number of test scenarios are possible for core testing at the system level. The embedded cores may be tested using built-in self test (BIST), external testing, or a combination of both. For external testing, the test buses that are used for test access may be shared among multiple cores. If BIST is used, then a core may either be "BIST-ed", in which case it has dedicated BIST logic, or it may simply be "BIST-ready" without containing BIST pattern generators and response monitors. In the latter case, the system integrator may design BIST logic that is shared by multiple cores. In order to minimize the testing time, the test resources in the system (test buses, BIST logic) should be carefully allocated cores, and the tests for the cores should be optimally scheduled.

Sugihara et al. [5] recently addressed the problem of selecting a test set for each core from a set of test sets provided by the core vendor and scheduling these tests in order to minimize the testing time. Each test set consists of a subset of patterns for BIST and a subset of patterns for external testing. This approach requires the core vendor to provide multiple test sets for each core, with the test sets containing varying proportions of patterns for BIST and external testing. The scheduling problem is formulated as a combinatorial optimization problem and solved heuristically. The authors make two restrictive assumptions (i) every core has its own BIST logic, i.e. the BIST components of the test sets for any two cores can be assigned identical starting times, and (ii) external testing can be carried out for only one core at a time, i.e. there is only one test access bus at the system level.

We formulate a generalized test scheduling problem that includes the problem addressed in [5] as a special case. The main contributions of the paper are summarized below.

- We relate the general problem of test scheduling to the NP-complete open-shop scheduling problem [2].
- We relate a special case of the scheduling problem to the problem of open-shop scheduling with two processors, and present a polynomial-time algorithm for it.
- Even though the scheduling problem is NP-complete, we show that it can be solved exactly for realistic core-based systems using mixed-integer linear programming (MILP).
- In order to handle larger systems, we present a heuristic "shortest-task-first" algorithm.

A generic example of core-based systems that we consider is shown in Figure 1. For core i, we assume that external test application takes e_i cycles and BIST takes b_i cycles. Note that in this generic example, Cores 1 and 2 are BIST-ed while Cores 3 and 4 share BIST logic. Also, Core 5 is tested entirely using BIST, while Core 6 is tested entirely using external patterns.

2 Polynomial-time algorithm

In this section, we consider a special case in which the core-based system has only one external bus and BIST logic is shared by all the cores. The test set for every core includes BIST and external test components. Figure 2 illustrates a system with four cores; the test lengths for BIST and external testing are also shown. For example, Core 1 requires 125 cycles for external testing and 100 cycles for BIST.

We first show that the test scheduling problem is equivalent to open-shop scheduling [2]. In open-shop scheduling, we are given a *shop* consisting of m processors, a set J of jobs, each job $j \in J$ consisting of m tasks $t_1[j], t_2[j], \ldots, t_m[j]$, and a length $l(t) \geq 0$ for each task. A schedule for an m-shop is a set of m processor schedules, one for each processor in the shop. These schedules must be such that no job is processed simultaneouly on more than one processor. The *finish time* of a schedule is the latest completion time of the individual processor schedules. The objective in open-shop scheduling is to minimize the finish time.

In order to establish equivalence between test scheduling for core-based systems and open-shop scheduling, we view the test sets for the cores as jobs. Each job consists of two tasks, corresponding to the external test and BIST components of the test set, respectively. For the problem instance being considered in this section, $m = 2$, i.e. there are two processors in the system, corresponding to the external test bus and the BIST resource, respectively. An optimal schedule, i.e. one with the least finish time, guarantees the shortest testing time for the core-based system. We

[1] This research was supported in part by the National Science Foundation under grant no. CCR-9875324, by a contract from Delphi Delco Electronic Systems, and by an equipment grant from Sun Microsystems.

0-7803-5832-X /99/$10.00 ©1999 IEEE.

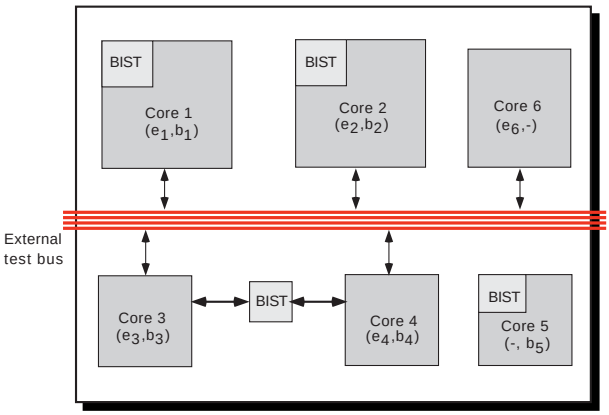


Figure 1. An example of a core-based system.

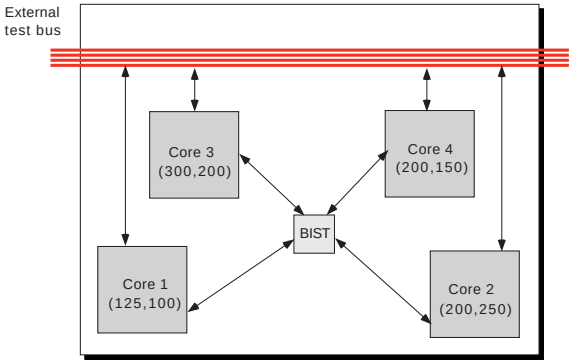


Figure 2. An example of a core-based system with one external test bus and BIST logic shared by all the cores.

can obtain efficient test schedules by exploiting the fact that the 2-shop scheduling problem can be solved efficiently using an $O(n)$ algorithm, where n is the number of jobs (cores) [2]. Figure 3 provides a pseudocode description of this algorithm. The external test schedule is denoted by the list S_1, while S_2 denotes the schedule for BIST. The symbol "||" is used to denote concatenation.

For the core-based system of Figure 2, optimal schedules for external test and BIST are given by S_1: 3124 and S_2: 4312, respectively, and the optimum testing time for this system is 825 cycles. An optimal schedule can be derived from S_1 and S_2.

As another example, we consider a representative core-based system $\mathcal{S}$ consisting of six ISCAS benchmarks (cores). These circuits are known to contain random-pattern-resistant faults, hence we use pseudorandom patterns for BIST as well as deterministic patterns (applied externally) for the hard faults. Table 1 presents the test data for each embedded core in this system. We assume that the s5378 circuit contains 4 internal scan chains, while each of the s1196 and s953 circuits contain a single internal scan chain. We also assume without loss of generality that a 32-bit external test bus is used. Finally, we use the parameter *test width* $\phi_i = \max\{n_i, m_i\}$, where n_i (m_i) equals the number of inputs (outputs) for core i.

Let t_i be the number of (scan) patterns for core i. The number of external test cycles required by core i, $T_i = t_i$ if $\phi_i \leq 32$, and $T_i = (\phi_i - 31)t_i$ if $\phi_i > 32$. If $\phi_i > 32$ then serialization of the test data is necessary at the inputs and/or outputs of core i. If the test bandwidth is adequate, i.e. $\phi_i \leq 32$, then no serialization is necessary and core i can be tested in exactly t_i cycles.

Procedure SCHEDULE($\{e_i, b_i\}$)
begin
  $T_1 := 0$; $T_2 := 0$; $e_0 := 0$; $b_0 := 0$; $l := 0$; $r := 0$; $S := \phi$;
  /* e_0 and b_0 are dummy variables */
  for i:=1 **to** n **do** /* system contains n cores */
   begin
    $T_1 := T_1 + e_i$; $T_2 := T_2 + b_i$
    if $e_i \geq b_i$ **then**
      if $e_i \geq b_r$ **then**
        $S := S \parallel r$; $r := i$;
      else
        $S := S \parallel i$;
    else
      if $b_i \geq e_l$ **then**
        $S := l \parallel S$; $l := i$;
      else $S := i \parallel S$;
   end
if $T_1 - a_l < T_2 - b_r$ **then**
  $S_1 := S \parallel r \parallel l$; $S_2 := l \parallel S \parallel r$;
else
  $S_1 := l \parallel S \parallel r$; $S_2 := r \parallel l \parallel S$;
delete_zeros(); /* Delete all occurrences of zeros from S_1 and S_2 */
end

Figure 3. An optimal test scheduling algorithm.

Circuit (core)	i	p_i	No. of scan test patterns t_i	No. of external test cycles	No. of BIST patterns	No. of BIST cycles
c880	1	13	13	377	4096	4096
c2670	2	79	79	15958	64000	64000
c7552	3	48	48	8448	64000	64000
s953	4	45	1379	28959	4096	217140
s5378	5	59	2759	60698	4096	389214
s1196	6	40	778	778	4096	135200

Table 1. Test data for the cores in System $\mathcal{S}$.

For the combinational cores, $1 \leq i \leq 3$, the number of scan test cycles t_i is equal to the number of test patterns p_i. However, for the remaining three cores with internal scan, $t_i = (p_i + 1)\lceil f_i/N_i \rceil + p_i$, where core i contains f_i flip-flops and N_i internal scan chains.

For this example, the two schedule lists are S_1: 162345 and S_2: 516234, respectively. This yields an optimum testing time of 1152180 cycles. Once again, an optimal schedule can be easily derived from S_1 and S_2.

3 Test scheduling: General case

While the special case of the scheduling problem discussed in Section 2 can be easily solved using a linear-time algorithm, the general case of $m > 2$ corresponding to more than one BIST resource in the system is NP-complete [2]. In this section, we develop a mixed-integer linear programming (MILP) model to solve the test scheduling problem.

A typical MILP model is described as follows [4]: *minimize* $\mathbf{Ax} + \mathbf{By}$ *subject to* $\mathbf{Cx} + \mathbf{Dy} \leq \mathbf{E}$, $\mathbf{x} \geq 0$, $\mathbf{y} \geq 0$, where $\mathbf{A}$ and $\mathbf{B}$ are cost vectors, $\mathbf{C}$ and $\mathbf{D}$ are constraint matrices, $\mathbf{E}$ is a column vector of constants, $\mathbf{x}$ is a vector of integer variables, and $\mathbf{y}$ is a vector of real variables. Eficient MILP solvers are now readily available, both commercially and in the public domain. For our experiments, we used the *lpsolve* package [1].

Let $T = \{t_1, t_2, \ldots, t_{2N_C}\}$ denote the start times of the set of test patterns (external and (BIST) that must be applied to the N_C cores in the system. The start time of the external test set e_i for core i is denoted by t_{2i-1} while the start time of the BIST test set b_i for core i is denoted by t_{2i}. For notational convenience, we will interchangeably use e_i (b_i) and $2i - 1$ ($2i$) to refer to the external (BIST) component of the test set of core i.

Minimize C subject to:

1. $C \geq t_i + l_i, 1 \leq i \leq 2N_C$
2. $x_{ij}(s_{ij1} - s_{ij2} - l_j\delta_{ij1}) + x_{ij}(s_{ij3} - s_{ij4} - l_i\delta_{ij2}) \geq 0$, $1 \leq i, j \leq 2N_C$
3. $\delta_{ij1} + \delta_{ij2} = 1, 1 \leq i, j \leq 2N_C$
4. $s_{ij1} - M\delta_{ij1} \geq 0, M = \sum_{i=1}^{N_C} l_i, 1 \leq i, j \leq 2N_C$
5. $-t_i + s_{ij1} \leq 0, 1 \leq i, j \leq 2N_C$
6. $t_i - s_{ij1} + M\delta_{ij1} \leq M, 1 \leq i, j \leq 2N_C$
7. $\delta_{ij1}, \delta_{ij2} = 0$ or $1, 1 \leq i, j \leq 2N_C$.

Figure 4. MILP model for $\mathcal{P}1$.

Let $L = \{l_1, l_2, \ldots, l_{2N_C}\}$ denote the corresponding test lengths (number of cycles) for the test sets. Note that if the test set for core i has no external test (BIST) component, then $t_{2i-1} = 0$ and $l_{2i-1} = 0$ ($t_{2i} = 0$ and $l_{2i} = 0$). Two test sets i and j *overlap* if either (i) $t_i < t_j + l_j$ and $t_i + l_i > t_j$, or (ii) $t_j < t_i + l_i$ and $t_j + l_j > t_i$. If there is only one external test bus, the external test components for the cores in any valid test schedule must not overlap. Note also that test sets i and j do not overlap if and only if either (i) $t_i - t_j - l_j \geq 0$, or (ii) $t_j - t_i - l_i \geq 0$.

Two test sets are *conflicting* if they cannot be applied to the system at the same time. Test sets can be conflicting if (i) they share an external test bus, (ii) they are BIST test sets for cores that share a BIST resource, or (ii) they are the external and BIST components of a core's test set. Clearly, there cannot be any overlap between conflicting test sets, and conflicts are reduced if the cores are assigned to multiple test buses.

The optimization problem ($\mathcal{P}1$) that we address is the minimization of system testing time by optimally determining the start times $t_1, t_2, \ldots, t_{2N_C}$ for the various test sets. Let x_{ij}, $1 \leq i, j \leq 2N_C$, be a 0-1 variable defined as follows: $x_{ij} = 1$ if the test sets i and j are conflicting, and $x_{ij} = 0$ otherwise. We first formulate the model for $\mathcal{P}1$ in terms of nonlinear constraints, and then linearize it using standard techniques.

Objective: Minimize the cost function $C = \max\{t_i + l_i\}$ **subject to**: $x_{ij}(t_i - t_j - l_j) \geq 0$ **or** $x_{ij}(t_j - t_i - l_i) \geq 0$.

The above minmax nonlinear cost function can easily be linearized [4] by minimizing the (real) variable C and adding the constraints $C \geq t_i + l_i, 1 \leq i \leq 2N_C$. To linearize the nonlinear constraints, we introduce 0-1 "indicator" variables δ_{ij1} and δ_{ij2}. The optimization problem is now restated as:

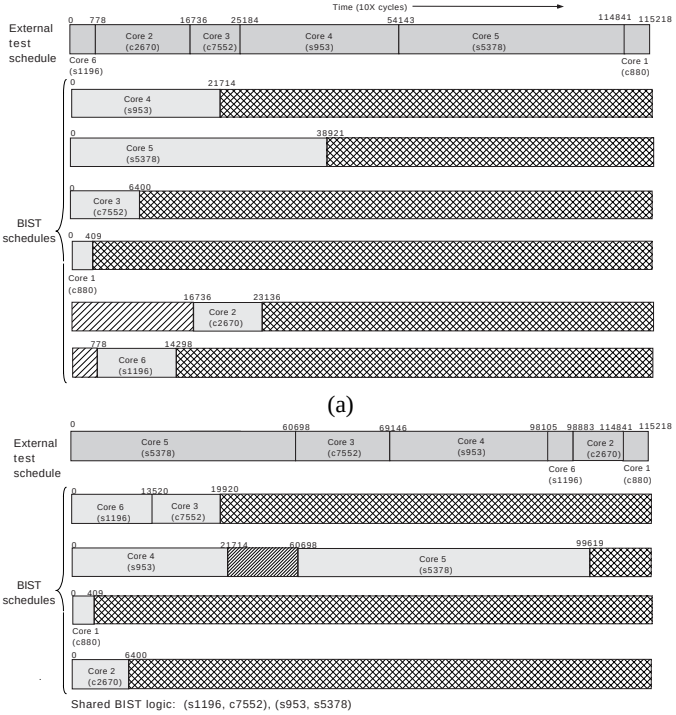
Objective: Minimize the cost function C **subject to**:

1) $C \geq t_i + l_i, 1 \leq i \leq 2N_C$.
2) $x_{ij}\delta_{ij1}(t_i - t_j - l_j) + x_{ij}\delta_{ij2}(t_j - t_i - l_i) \geq 0$.
3) $\delta_{ij1} + \delta_{ij2} = 1, 1 \leq i, j \leq 2N_C$.
4) $\delta_{ij1}, \delta_{ij2} = 0$ or $1, 1 \leq i, j \leq 2N_C$.

The constraint 2) above is still nonlinear. We linearize it by replacing $t_i\delta_{ij1}$ by the (real) variable s_{ij1} and $t_j\delta_{ij1}$ by the (real) variable s_{ij2}. Similarly, we replace $t_j\delta_{ij2}$ by s_{ij3} and $t_i x_{ij2}$ by s_{ij4}. For each such substitution, we add three additional constraints. For example, for the substitution of $t_i\delta_{ij1}$ by s_{ij1}, we add: (i) $s_{ij1} - M\delta_{ij1} \geq 0$, (ii) $-t_i + s_{ij1} \leq 0$, and (iii) $t_i - s_{ij1} + M\delta_{ij1} \leq M$, where $M = \sum_{i=1}^{N_C} l_i$ is an upper bound on t_i. The resulting MILP model is shown in Figure 4.

We applied the MILP model of Figure 4 to the core-based system described in Table 1 for several test scenarios corresponding to varying amount of on-chip BIST resources. Note that the scheduling complexity does not depend on the sizes of the cores—it depends only on the number of cores in the system, and on the number of resource conflicts.

We assumed that the application of a BIST pattern takes one clock cycle and external test application is ten time slower. We solved the ILP models using a Sun Ultra 10 workstation with a 333 MHz processor and 128 MB memory. We were unable to obtain actual CPU times from *lpsolve*; however, the user time was less than one minute in each case. The optimum testing time for this system is 1152180 cycles, e.g. see Figure 5.



(a)

(b)

Figure 5. Optimal test schedules: (a) cores have dedicated BIST; (b) cores share BIST resources.

Since test scheduling is NP-complete, the amount of time required for larger systems may be excessive. In order to handle such systems, we present a shortest-task-first heuristic scheduling algorithm (Figure 6). It initializes the start times of the test sets (tasks) to zero, and then iteratively updates the start times until all test conflicts are eliminated. For m tasks, its worst-case time complexity is $O(m^3)$.

The shortest-task-first algorithm yields a testing time of 1204630 cycles for the system of Table 1 when each core has its dedicated BIST circuit. This is only 4.5% greater than the optimum testing time of 1152810 cycles.

4 Test scheduling with multiple test sets

In this section, we select test sets for the cores from a set of alternatives, and optimally determine their start times. While Sugihara et al. [5] provide a heuristic solution and make the restrictive assumption that the cores do not share BIST logic, our general MILP model allows sharing of BIST resources among cores and provides an exact solution.

```
Procedure SHORTEST-TASK-FIRST({t_i})
begin
for i:=1 to m do /* there are m tasks */
   start_time_i := 0;
while flag = 1 do
   begin
   flag = 0;
   for i:=1 to m do
      for j:=i+ 1 to m do
         if x_ij = 1 then /* x_ij = 1 if i and j are conflicting */
            if OVERLAP(i, j) then
               begin
               if start_time_i + l_i ≥ start_time_j + l_j then
                  start_time_i := start_time_j + l_j;
               else start_time_j := start_time_i + l_i;
               flag = 1; end
   end
end
```

Figure 6. The shortest-task-first procedure.

Suppose we have N_i alternative test sets for core i in the system. These test sets may contain varying proportion of BIST patterns. We denote the test lengths for the BIST patterns for core i by $l_{2i,1}, l_{2i,2}, \ldots l_{2i,N_i}$. Similarly, we denote the test lengths for the external patterns for core i by $l_{2i-1,1}, l_{2i-1,2}, \ldots, l_{2i-1,N_i}$. If the kth test set is chosen for core i, then it consists of $l_{2i,k}$ cycles for BIST and $l_{2i-1,k}$ cycles for external testing.

We use the parameter x_{ij} and the variables t_i, δ_{ij1}, and δ_{ij2} ($1 \leq i, j, \leq 2N_C$) as defined for $\mathcal{P}2$. In addition, we use a 0-1 indicator variable λ_{ik} ($1 \leq i \leq 2N_C$, $1 \leq k \leq N_i$), which is set to 1 if the kth test set (consisting of BIST and external test patterns) is selected for core i. We now develop the MILP model for the optimization problem $\mathcal{P}2$.

Objective: Minimize $C = \max\{t_i + \sum_{k=1}^{N_i} \lambda_{ik} l_{ik}\}$ **subject to**:

1) $\sum_{k=1}^{N_i} \lambda_{ik} = 1, 1 \leq i \leq 2N_C$
2) If i and j refer to the same core ($i - j = 1$ and i is even), then $\lambda_{ik} - \lambda_{jk} = 0, 1 \leq k \leq N_i$
3) $\sum_{k=1}^{N_i} \lambda_{ik} = 1, 1 \leq i \leq 2N_C$
4) $x_{ij}\delta_{ij1}(t_i - t_j - \sum_{k=1}^{N_j} \lambda_{jk} l_{j,k}) + x_{ij}\delta_{ij2}(t_j - t_i - \sum_{k=1}^{N_i} \lambda_{ik} l_{i,k}) \leq 0$
5) $\delta_{ij1}, \delta_{ij2} = 0$ or 1, $1 \leq i, j \leq 2N_C$.

Once again, the minmax nonlinear cost function can easily be linearized [4] by minimizing the (real) variable C and adding the constraints $C \geq t_i + \sum_{k=1}^{N_i} \lambda_{ik} l_{i,k}$, $1 \leq i \leq 2N_C$. In order to linearize constraint 4) above, we replace the product of 0-1 variables $\delta_{ij1}\lambda_{jk}$ by u_{ij1k} and $\delta_{ij2}\lambda_{ik}$ by u_{ij2k}, $1 \leq i, j \leq 2N_C$. We also need six constraints for each such substitution [4]. The resulting MILP model for $\mathcal{P}2$ shown in Figure 7.

We applied the MILP model to the example of four 16-bit multiplier cores used in [5]. Figure 8 shows the test lengths and an optimal selection of test sets (highlighted) for these cores; an optimal test schedule requires 188 cycles.

5 Conclusions

We have presented optimal solutions to the test scheduling problem for core-based systems. We have shown that test scheduling is equivalent to m-processor open-shop scheduling, and is therefore NP-complete. However, it can be solved in polynomial time if all the cores in the system share BIST logic. For the general case, we have presented a mixed integer linear programming (MILP) model for optimal scheduling. We have also extended the MILP model to allow optimal test set selection from a set of alternatives. Finally, we have presented an efficient heuristic scheduling algorithm for handling larger systems.

Minimize C subject to:

1. $C \geq t_i + \sum_{k=1}^{N_i} \lambda_{ik} l_{i,k}$
2. $x_{ij}(s_{ij1} - s_{ij2} - \sum_{k=1}^{N_j} l_k u_{ijk1}) + x_{ij}(s_{ij3} - s_{ij4} - \sum_{k=1}^{N_i} l_k u_{ijk2}) \geq 0, 1 \leq i, j \leq 2N_C$
3. $s_{ij1} - M\delta_{ij1} \geq 0, M = \sum_{i=1}^{N_C} l_i, 1 \leq i, j \leq 2N_C$
4. $-t_i + s_{ij1} \leq 0, 1 \leq i, j \leq 2N_C$
5. $t_i - s_{ij1} + M\delta_{ij1} \leq M, 1 \leq i, j \leq 2N_C$
6. $\delta_{ij1}, \delta_{ij2} = 0$ or 1, $1 \leq i, j \leq 2N_C$
7. $\delta_{ij1} + \delta_{ij2} = 1, 1 \leq i, j \leq 2N_C$
8. If i and j refer to the same core ($i - j = 1$ and i is even), then $\lambda_{ik} - \lambda_{jk} = 0, 1 \leq k \leq N_i$
9. $u_{ijk1} = 0$ or 1, $1 \leq i, j \leq 2N_C, 1 \leq k \leq N_j$
10. $u_{ijk2} = 0$ or 1, $1 \leq i, j \leq 2N_C, 1 \leq k \leq N_i$
11. $-\lambda_{jk} + u_{ij1k} \leq 0, 1 \leq i, j \leq 2N_C, 1 \leq k \leq N_j$
12. $-\delta_{ij1} + u_{ij1k} \leq 0, 1 \leq i, j \leq 2N_C, 1 \leq k \leq N_j$
13. $\lambda_{jk} + \delta_{ij1} - u_{ij1k} \leq 1, 1 \leq i, j \leq 2N_C, 1 \leq k \leq N_j$
14. $-\lambda_{ik} + u_{ij2k} \leq 0, 1 \leq i, j \leq 2N_C, 1 \leq k \leq N_i$
15. $-\delta_{ij2} + u_{ij2k} \leq 0, 1 \leq i, j \leq 2N_C, 1 \leq k \leq N_i$
16. $\lambda_{ik} + \delta_{ij2} - u_{ij2k} \leq 1, 1 \leq i, j \leq 2N_C, 1 \leq k \leq N_i$

Figure 7. MILP model for $\mathcal{P}2$.

	Test lengths (BIST, external)		
Core 1	(235, 9)	(10, 58)	(55,30)
Core 2	(120, 27)	(140, 19)	(270, 10)
Core 3	(20, 46)	(60, 28)	(360, 13)
Core 4	(55, 84)	(120, 68)	(195, 53)

Figure 8. An optimal test selection for the multiplier cores.

Acknowledgements

The author thanks M. Sugihara of Kyushu Univ., and H. Date of the Inst. Sys. & Inf. Tech., Kyushu, Japan for providing test data for the multiplier cores.

References

[1] M. Berkelaar. *lpsolve*, version 2.0. Eindhoven Univ. Tech., The Netherlands. E-mail: michel@es.ele.tue.nl.

[2] T. Gonzales and S. Sahni. Open shop scheduling to minimize finish time. *Journal of the ACM*, vol. 23, pp. 665–679, Oct 1976.

[3] E. J. Marinissen et al. A structured and scalable mechanism for test access to embedded reusable cores. *Proc. Int. Test Conf.*, pp. 130–143, 1998.

[4] H. P. Williams. *Model Building in Mathematical Programming*, 2nd ed., John Wiley, New York, 1985.

[5] M. Sugihara, H. Date and H. Yasuura. A novel test methodology for core-based system LSIs and a testing time minimization problem. *Proc. Int. Test Conf.*, pp. 465–472, 1998.

[6] Y. Zorian, E. J. Marinissen and S. Dey. Testing embedded-core based system chips. *Proc. Int. Test Conf.*, pp. 130–143, 1998.