Robust Optimization of Test-Architecture Designs for Core-Based SoCs*

Sergej Deutsch and Krishnendu Chakrabarty
Department of Electrical and Computer Engineering
Duke University, Durham, NC 27708, USA

***Abstract*—Today's technology allows for the integration of many cores in a single die, for instance, in core-based SoCs, and an even larger number of cores are likely to be integrated over multiple layers in a 3D stack. In order to minimize test cost, the test architecture in a core-based SOC is optimized for minimum test time. Optimization methods in use today assume that all relevant input parameters, such as core test time and power consumption during test, are known at the design stage. However, these parameters can change after manufacturing and, in that scenario, the originally designed test architecture may no longer be optimal. Moreover, conventional optimization methods have to consider worst-case estimates for all input parameters to ensure feasibility, which can result in conservative and hence expensive solutions. We propose the use of robust optimization for test-architecture design and test scheduling. This goal of this approach is to find a solution that remains close to optimal in the presence of parameter variations. Experimental results for the ITC'02 SoC benchmarks show that, compared to optimization methods that target only a single point in the input-parameter space, robust optimization can better optimize test time in the presence of parameter variations.**

I. Introduction

The continuous progress in semiconductor technology has enabled integration of complex systems comprising a large number of different modules, such as embedded cores, in a single die or package. Examples include system-on-chip (SoC) and three-dimensional stacked ICs (3D ICs) using through-silicon via (TSV) technology [1]. Due to the increasing number of cores that can be integrated on a single die in an SoC or in a 3D IC stack, optimization of the test architecture and test schedule require more attention in order to minimize test cost.

In the past, many publications have focused on optimization of test architecture and test schedule for conventional 2D SoCs [2]–[8] as well as for 3D ICs [9]. The proposed optimization methods have included, for instance, integer linear programming (ILP) and heuristics such as rectangle packing [9]. These conventional optimization methods assume that all input parameters, such as core test times and the compatibility of core tests are known constants. However, these data may change after the design has been fixed, such that the originally obtained solution is no longer optimal.

Input parameter variations can have multiple origins. One source of uncertainty in SoC test planing is the application of adaptive testing [10]. In adaptive testing, test limits and test content are dynamically adjusted during production testing based on the analysis of test responses. These adjustment can result, for instance, in variation of core test time, which is used as input parameters for optimization of test architecture and test schedule at the design stage. With the updated input parameters, the given test architecture may not be optimal in terms of total test time. However, we cannot change the design at the production-test stage hence additional constraints must be added to the test schedule. By assuming extreme-case values for the input parameters, the design may not be optimized for other cases, resulting in potential loss of benefits of adaptive testing.

Another example of uncertainty in parameter values is the power consumption of cores during test, which is an important consideration in IC test. In particular, for emerging 3D ICs that can integrate a large number of modules, the total power consumption during test must be limited in order to prevent overheating and potential overkill. Moreover, voltage droop due to abnormally high power consumption can cause over-testing. Even though it is possible to generate low-power tests with commercial tools, not all core tests are optimized for low power. This has to be taken into account during test architecture optimization and test scheduling to prevent violations of power constraints due to parallel core test. Therefore, accurate estimates of power consumption during test are required in the design stage. However, these estimates, e.g., obtained from simulations, are often inaccurate and the actual power consumption can significantly deviate from the predicted values due to process variations.

Neglecting variations in these parameters and assuming worst-case values will result in an overly conservative solution, unnecessarily increasing the product cost. A better approach is to anticipate potential variations in input parameters and make the solution insensitive to these variations. This type of a solution method is called "robust optimization" [11]. The main idea of robust optimization is to find a solution that may not be optimal for the nominal values of the input parameters but that remains close to the optimum in the presence of parameter variations. Even though this approach was originally developed for linear programming (LP), we can also apply it for robust optimization with ILP models. However, the complexity of the model grows fast with the number of cores and the number of parameters that can vary, hence solving it using exact methods for large problem sizes is infeasible. Therefore, efficient heuristics are also needed.

The key contributions of this paper are the following.

- We formulate, for the first time, a robust optimization

*The work of S. Deutsch and K. Chakrabarty was supported in part by the National Science Foundation under grants CCF-1017391 and CCF-0903392, and by the Semiconductor Research Corporation (SRC) under contract no. 2118.001.

978-1-4673-6377-8/13/$31.00 ©2013 IEEE

problem for SoC testing.

- We formulate an ILP model for robust optimization of test architecture design and test scheduling. This model is practical for systems, such as SoCs or 3D ICs, that contain a relatively small number of cores.
- We show that robust optimization provides solutions that are on average better than non-robust solutions in terms of test time.
- For larger systems, for which the exact solution of the ILP model becomes intractable, we propose a "randomized divide-and-conquer" heuristic for robust optimization that uses the exact ILP method to solve sub-problems. The proposed heuristic scales linearly with the number of cores and is hence practical for large systems. In addition, since the iterations during the randomized divide-and-conquer procedure are independent from each other, the performance of our heuristic can be increased by running multiple instances (or threads) in parallel.

The remainder of this paper is organized as follows. Section II gives an overview of robust optimization. In Section III, we formulate an ILP model for robust optimization of test architecture and present our heuristic method for robust optimization for systems containing a large number of cores. In Section IV, we provide experimental results for various SoCs using ITC'02 test benchmarks [12]. In Section V, we discuss limitations of the current method and directions for future work. Finally, Section VI concludes the paper.

II. Overview of Robust Optimization

Conventional (non-robust) optimization of test architectures for SoCs as well as for 3D ICs has been studied in the past [2]–[9], [13]. The proposed methods assume that all input parameters are known and will remain constant after the design has been fixed. As explained in Section I, this may not be true in practice, resulting in non-optimal solutions.

The robust optimization method proposed in [11] takes uncertainties of design parameters into account. This work introduces two types of variables:

- $x \in \mathbb{R}^{n_1}$, which denotes the vector of *design* variables. Their optimal values do not depend on the uncertain parameters and they cannot be adjusted once the values of the uncertain parameters are known.
- $y \in \mathbb{R}^{n_2}$, which denotes the vector of *control* variables. These parameters can be adjusted once the values of the uncertain parameters are known.

The LP model using design and control variables is structured as follows.

Objective:

$$\text{Minimize} \quad c^T x + d^T y$$

Subject to:

$$Ax = b,$$
$$Bx + Cy = e,$$
$$x, y \geq 0,$$

where A and b are noise-free constraint parameters and d, B, C, and e are subject to noise. For the robust optimization problem, a set of scenarios $\Omega = \{1, 2, 3, \ldots, S\}$ is defined. Each scenario $s \in \Omega$ occurs with the probability p_s ($\sum_{s\in\Omega} p_s = 1$), where the noisy parameters take the values $\{d_s, B_s, C_s, e_s\}$. A solution to the above program is robust if it remains "close" to the optimum for any scenario $s \in \Omega$, which means that the deviation of the cost function from the optimal value is minimized.

The set $\{y_1, y_2, \ldots, y_s\}$ introduces control variables for each scenario. The set $\{z_1, z_2, \ldots, z_s\}$ contains error vectors to measure the allowed infeasibility. The mathematical model for robust optimization is defined as follows.

Objective:

$$\text{Minimize} \quad \sigma(x, y_1, \ldots, y_s) + \omega\rho(z_1, \ldots, z_s)$$

Subject to:

$$\begin{aligned} &Ax = b, \\ &B_s x + C_s y_s + z_s = e_s, \qquad \forall\, s \in \Omega, \\ &x,\ y_s \geq 0, \qquad \forall\, s \in \Omega, \end{aligned} \tag{1}$$

where $\sigma(\cdot)$ can be either the mean value of the cost function for stochastic analysis $\sigma(\cdot) = \sum_{s\in\Omega} p_s(c^T x + d_s^T y_s)$ or the maximum value of the cost function for worst-case analysis $\sigma(\cdot) = \max_{s\in\Omega}(c^T x + d_s^T y_s)$. The function $\rho(z_1, \ldots, z_s)$ serves as a penalty function to restrain violations of the control constraints by using the weight ω. In our work, we set this function to zero.

The benefit of robust optimization can be demonstrated using a simple minimization problem $F = \min\{c^T x \mid Ax \geq b,\ x \geq 0\}$, where $c^T = [1\ 1]$, $b^T = [1\ 1]$, and A can take the values $A_{s_1} = \begin{bmatrix} 0.7 & 0.5 \\ -1.2 & 1.3 \end{bmatrix}$ in Scenario s_1 or $A_{s_2} = \begin{bmatrix} 0.65 & 0.7 \\ -1 & 1 \end{bmatrix}$ in Scenario s_2 with equal probabilities. Suppose that x_1 is a design variable, i.e. it is the same for both s_1 and s_2, and x_2 is a control variable that can be adjusted in each scenario. The table below shows the solutions of the LP problem (a) for $s1$, (b) for $s2$, and (c) taking both s_1 and s_2 into account. The non-robust solutions show relatively large increase of the cost F if the input parameters change. Even though the robust solution moves away from the optimum for both scenarios, it has the best expectation of the cost.

	x_1	$x_2(s_1)$	$x_2(s_2)$	$F(s_1)$	$F(s_2)$	F_{exp}
a) s_1-optimized	0.53	1.26	1.53	1.79	2.06	**1.92**
b) s_2-optimized	0	2	1.43	2	1.43	**1.74**
c) robust	0.22	1.69	1.22	1.91	1.44	**1.67**

III. Robust Optimization and Application to SoC Testing

The robust optimization method proposed in [11] is scalable if the problem can be modeled as LP, since LP problems are solvable in polynomial time. However, many real-world optimization problems, such as test scheduling, are $\mathcal{NP}$-hard and cannot be solved in polynomial time. To solve these problems optimally, we can use ILP for reasonably-sized problem instances.

In this section, we formulate an ILP model for co-optimization of the test architecture and the test schedule for core-based systems, such as SoCs and 3D ICs. Subsequently, we extend this model for robust optimization in order to take uncertainties in input parameters into account.

Objective:

Minimize $\{T_{tot}\}$ (2)

Subject to:

$w_i + (W_i - 1) \leq W_{max} \quad \forall i$ (3)

$q_{i,j} \iff (w_i \geq w_j + W_j \text{ OR } w_j \geq w_i + W_i) \quad \forall i,j$ (4)

$q_{i,i} = 1 \quad \forall i$ (5)

$r_{i,j} \iff t_j \leq t_i < t_j + T_j \quad \forall i,j$ (6)

$r_{i,j} \leq q_{j,i} \quad \forall i,j$ (7)

$t_i + T_i \leq T_{tot} \quad \forall\, i$ (8)

$$\sum_{j=1}^{N} P_j r_{i,j} \leq P_{max} \quad \forall\, i \tag{9}$$

Fig. 1. Mathematical programming model for non-robust test architecture optimization.

A. ILP Model for Non-Robust Co-Optimization of Test Architecture and Test Scheduling

Consider a system, e.g., an SoC or 3D IC, containing N cores that need to be tested. The objective is to minimize the total test time. Each core $i \in \{1, \ldots, N\}$ has a test access mechanism (TAM) with a width of W_i wires. A total of T_i clock cycles are required to test core i. The system provides a test bus (system-level TAM) to which all cores need to be connected. The total width of the system-level TAM is W_{max}. Multiple modules can share the same system TAM wire; in that case, at most one of these modules can be tested at a time. If two modules do not share system TAM wires, they can be tested in parallel.

In addition to the maximum system-level TAM width constraint, test scheduling can be further constrained due to incompatibility of tests for certain cores. One example is power during test. Power during scan test is significantly higher that power during functional operation. This can result in overkill or even in permanent damage of the system under test if the total power during test exceeds a certain threshold. We model this constraint as follows. Each core has a known peak power P_i during test. At any point of time, the sum of P_i of cores currently being tested should not exceed a given parameter P_{max}. Another example of scheduling constraint can be a shared resource. For instance, if multiple cores share a built-in self-test (BIST) resource, they cannot be tested in parallel [13]. In this work, we only include the power constraint into our ILP model. The model, however, can be easily extended with other constraints.

Figure 1 shows our mathematical programming model for non-robust test architecture optimization which is later extended to solve the robust optimization problem. The following text describes the model in detail. We define an array of integer variables w_i, $1 \leq i \leq N$ that indicate the lowest TAM wire to which core i is connected. We assume that each core i connects to the continuous part of the test bus starting at w_i and ending at $w_i + (W_i - 1)$. Line (3) expresses the limit on maximum available TAM wires.

Next, we introduce an an array of integer variables t_i, $1 \leq i \leq N$ that holds the starting times of core tests. The time slot for testing core i will end at $t_i + T_i$. Therefore, the total test time is $T_{tot} = \max_i\{t_i + T_i\}$, and the objective is to minimize it. This is captured in lines (2) and (8).

We need to include constraints to forbid parallel testing of two cores that share system TAM wires. For this purpose, we introduce a new binary variable $q_{i,j}$, which is 1 if Core i and Core j can be tested in parallel and 0 otherwise. Two cores can only be tested in parallel if their TAM assignments do not overlap. Therefore $q_{i,j}$ and w_i have the relationship as described in Line (4). In addition to overlap in TAM, we define a new binary variable $r_{i,j}$, which is 1 if the test of Core i starts while testing Core j, as modeled in Line (6). Note that if $t_i = t_j + T_j$, then $r = 0$, which means that the test of module i starts right after the test of module j has finished and the tests do not overlap. The test of Core i can start while Core j is currently being tested only if these cores do not share TAM wires. This constraint is captured in Line (7). The constraints (4) and (6) are non-linear and hence need to be linearized to obtain an ILP model, e.g., using standard linearization techniques.

To satisfy the maximum power constraint, it is sufficient to look at the starting point of each core i, sum the power consumption of all cores that are being tested at this point (including P_i), and verify that the total power is within the allowed limit. Line (9) includes this constraint.

By solving the problem described above, we get an optimal core TAM assignment (w_i) and an optimal test schedule (t_i) for a given set of input parameters.

B. ILP Model Extension for Robust Co-Optimization of Test Architecture and Test Scheduling

The ILP model in Figure 1 assumes that all input parameters are known in the design stage of the system. After the product has been manufactured, some of these parameters may change. For instance, the test of Core i may need to be augmented to increase test quality, resulting in a different core test time T_i and power consumption P_i. The originally obtained solution, w_i and t_i, may no longer be optimal and can even be infeasible if the power constraint is violated. We can solve the optimization problem again using the new parameter values; however, we cannot change variables related to the test architecture since it has already been fixed. At this stage, we can only adjust the test schedule, taking the TAM assignment as a constraint. Therefore, the vector w_i becomes an input parameter, which fixes all variables in lines (3)-(5). The new schedule will be optimal for the given architecture but may not be optimal if both test architecture and test scheduling had been co-optimized using the new parameter values.

A better approach is to anticipate potential variations in input parameters at the design stage and find a test architecture that is *robust* to these variations, which means that the value of the objective function (total test time T_{tot}) remains close to the optimum. For this, we use the approach described in [11] and extend the ILP model described in the previous section.

We assume that the probability distribution of the uncertain parameters T_i and P_i is known and that they can only take values from a finite set of elements. Let F_s denote the probability that these parameters will take the values $\{T_{i,s}, P_{i,s}\}$ where s is one of the S potential scenarios from the set $\Omega = \{1, 2, 3, \ldots, S\}$.

We distinguish between *design* and *control* variables. Design variables cannot be adjusted after the design has been fixed. In our robust ILP model, the design variables are w_i and $q_{i,j}$, and they remain the same for each constraint scenario s, as well as the constraints (3)-(5). In contrast, the test schedule can be adjusted

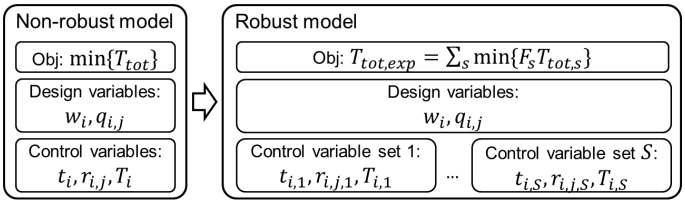

Fig. 2. Extension of non-robust ILP model to robust ILP model.

depending on which scenario occurs. For each scenario s, we introduce an independent set of control variables $t_{i,s}$, $r_{i,j,s}$, and $T_{tot,s}$. The constraints (4)-(9) of the ILP model are replaced by S sets of constraints with the new control variables.

In this work, we minimize the expectation of the total test time $T_{tot,exp} = \sum_{s\in\Omega} F_s T_{tot,s}$. Alternatively, the same ILP model can be used for worst-case robust optimization where the worst test time is minimized. Figure 2 shows a summary of the conversion of the non-robust ILP model to the robust one. Note that the number of the control variables and the number of the constraints associated with test scheduling scale linearly with the number of scenarios S. Furthermore, the non-robust model is a special case of the robust model ($S = 1$).

A solution to the robust optimization ILP problem gives us a "robust" core TAM assignment (w_i) and optimal test schedules ($t_{i,s}$) for each scenario. The number of constraints and the number of variables in the robust ILP model is $O(SN^2)$. Note that the robust ILP model becomes intractable for systems with many cores. Our experiments show that a commercial ILP solver was not able to find an optimal solution for a 12-core system within 24 hours. This motivates the need for a heuristic method to tackle this problem.

C. Randomized Divide and Conquer Heuristic

Even though solving the robust optimization problem for large N using ILP takes extremely long time, we can obtain an optimal solution for sufficiently small N within seconds. Due this property, we can use the ILP model to create a heuristic method for the robust optimization problem.

The algorithm for our *randomized divide and conquer* heuristic is shown in Figure 3. First, the algorithm randomly assigns the cores to M subgroups of a sufficiently small size, such that the subproblems can be solved within a few seconds. Then each subgroup g_m is solved exactly using the robust ILP model. Finally, the results are combined such that the test between different groups do not overlap. The total test time is then the sum of sub-solutions $T_{tot,exp} = \sum_{m=1}^{M} T_{tot,exp,m}$. Figure 4 shows an example of a solution for an eight-core SoC divided into two subgroups. The TAM allocation and test schedule for each subgroup is obtained by solving the robust ILP model. The test architecture for the entire SOC is the combination of the TAM assignments in Subgroup 1 and Subgroup 2, and the starting point of the test schedule for Subgroup 2 is shifted by $T_{tot,1}$.

We repeat the entire procedure with a new (random) assignment of the subgroups until we reach a certain number of iterations. The best solution is then selected. Since the iterations of the heuristic algorithm are independent from each other, we can run multiple instances of the optimization tool on different machines and reduce the time needed for optimization.

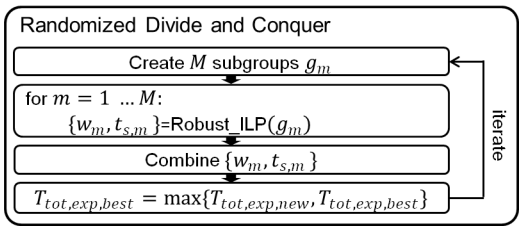

Fig. 3. Randomized divide and conquer flow.

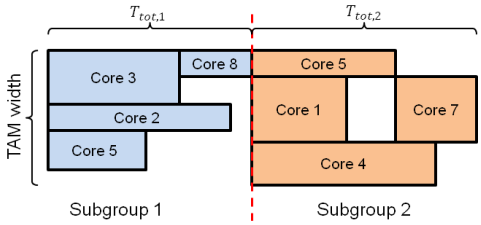

Fig. 4. Combining solutions obtained for two subgroups using robust ILP method.

IV. Experimental Results

In this section, we present experimental results to evaluate the ILP model for robust optimization described in the previous section. To solve the formulated ILP models, we used the Xpress-MP tool [14]. We applied our methods to the SoCs q12710, d695, h953, p93791, and p22810 from the ITC'02 SoC Test Benchmark set [12]. The relevant design data of these benchmarks are summarized in Table I, which shows the core TAM width W, the core test time T, and the percentage of the scan flip-flop count in relation to the total scan flip-flop count (p22810 is not shown due to the large number of cores). In these SoCs, we only considered cores that have scan chains; hence the SoCs have 4, 7, 8, 12, and 22 cores to be connected to the system TAM, respectively. We assume that the cores are wrapped, e.g., by IEEE 1500 wrappers, and the test time for each core is calculated *a priori* based on the wrapper design. However, our approach can be applied to any wrapper design. The core test time was estimated as the product of the length the longest scan chain and the test pattern count. For convenience, we normalized the test times for q12710 by 100 clock cycles, and for the other three SoCs by 1000 clock cycles.

We assume that the power during test is proportional to the amount of logic switching during test which, in turn, is proportional to the number of scannable flip-flops that are switching. To set the limit on total power during test, we allow only a certain amount of logic to be active at a time during test. For instance, if we set the maximum amount of active logic during test to 40% for q12710 then Core 1 can only be tested in parallel to either Core 3 or Core 4 ($16.0\% + 23.5\% \leq 40\%$); otherwise the power constraint would be violated. In real applications, the core test power and maximum allowed power will be estimated using simulation. These data will be subject to variations because of simulation inaccuracy and lack of correlation between simulated data and measured data for nanoscale ICs.

In practice, parameters of each core can vary independently from each other. To take this into account, we need to create a large set of scenarios to cover all possible combinations. This, however, drastically increases the complexity of the problem, since the number of control variables and the corresponding constraints scales linearly with the number of scenarios. In this

TABLE I
DESIGN PARAMETERS OF THE TEST SOCS.

	q12710			h953			d695			p93791		
	W	T	% FF	W	T	% FF	W	T	% FF	W	T	% FF
Core 1	3	8	16.0	4	118	24.2	16	94	10.0	46	113	26.5
Core 2	4	22	37.0	2	3	14.0	4	56	3.3	44	75	8.3
Core 3	3	16	23.5	2	1	1.4	32	50	22.5	46	68	7.6
Core 4	3	16	23.5	4	1	1.8	4	44	2.8	46	62	3.4
Core 5				4	13	10.4	32	35	25.5	46	42	10.6
Core 6				4	33	15.8	16	31	8.3	46	42	10.6
Core 7				8	57	32.4	1	24	0.5	46	40	8.5
Core 8							32	6	27.0	46	36	4.7
Core 9										35	32	7.3
Core 10										43	32	7.1
Core 11										44	21	4.8
Core 12										11	15	0.6

work, we limit the number of scenarios by assuming that only the power consumption P_i of two cores can be uncertain. We assume that P_i takes its nominal value with the probability 0.6, and deviates from it by $\pm 20\%$ with the probability 0.2.

We were able to exactly solve the ILP model for robust optimization for the first three SoCs. For the largest SoC, p93791, the solver did not terminate within 24 hours, which will be discussed later. In the following, we explain using the simplest of our benchmarks, why a non-robust approach can result in unnecessarily costly solutions.

Figure 5(a) shows the TAM assignment for q12710 that we obtained with the robust optimization method, assuming variations in power of Core 3 and Core 4, and the constraints $W_{max} = 6$ and $P_{max} = 40$. We used three scenarios that result in three different test schedules. In Scenario 1 (best case), Core 3 and Core 4 have 80% of their nominal power consumption, $P_3 = P_4 = 18.8$. In case this scenario occurs after the test architecture has been fixed, Core 3 and Core 4 can be tested in parallel, and Schedule s_1 in Figure 5(c) is optimal with $T_{tot,1} = 46$. If the input parameters take the nominal values (nominal case), Core 3 and Core 4 cannot be tested in parallel anymore because of the power constraint; however Core 1 and Core 4 can be scheduled for the same time slot such that $T_{tot,2} = 54$, resulting in Schedule s_2. In the third scenario $P_3 = P_4 = 28.2$ (worst case); therefore no two cores can be tested in parallel, which results in a sequential schedule (Schedule s_3) with $T_{tot,3} = 62$. Assuming that the scenarios occur with the probability 20%-60%-20%, the expectation of the total test time for a robust solution is $T_{tot,exp} = 54$.

If we solve the ILP model assuming the nominal case or the worst case, the solver provides the TAM wire assignment shown in Figure 5(b) as solution. Figure Figure 5(d) shows optimal test schedules for this architecture for the three cases. For both nominal and worst case scenarios, the test time is the same as that of the robust solution. However, the solver did not have the information that P_3 and P_4 can take the values such that parallel test of Core 3 and Core 4 becomes feasible. With this test architecture, we cannot schedule them in parallel for the best case and have to use s_2, losing the option to improve the test time in the best case. Assuming the same parameter distribution, the expectation of the total test time is $T_{tot,exp} = 55.6$ which is higher than than that of the robust solution.

In this simple example, we could have taken a non-robust solution for a single scenario, identified the potential for improvement by inspection and re-assigned Core 3 and Core 4 to different TAM

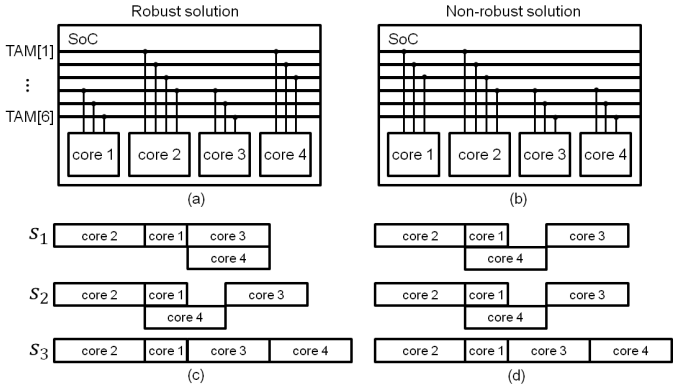

Fig. 5. Robust and non-robust test architectures for q12710 and the corresponding schedule for three different scenarios.

wires in order to have the option to test them in parallel. For SoCs with a large number of cores, this manual approach is no longer practical due to the large number of all possibilities. Therefore, we have to take the solution provided by the solver.

We compared a robust test architecture obtained for a larger SoC, d695, with three non-robust architectures that were created assuming the following input parameters: the best-case value for power of Core 3 $P_3 = 0.8P_{3,nom}$ (BC solution), the nominal value $P_3 = P_{3,nom}$ (NC solution), and the worst-case value $P_3 = 1.2P_{3,nom}$ (WC solution). We calculated optimal test schedules for the four solutions for a range of values of P_3. Figure 6(a) shows the results obtained for the case $W_{max} = 64$ and $P_{max} = 35$. In this case, the WC solution has no potential for improvement in terms of test time for $P_3 \leq 1.1P_{3,nom}$, which means that this solution will not benefit from a decrease in test power of Core 3.

It is not only the WC scenario that can result in non-optimal architectures. The BC and NC can also lead to non-optimal solutions, as shows in Figure 6(b). In this case, we changed the constraints to $W_{max} = 50$ and $P_{max} = 37$. Even though the BC and NC solutions are slightly below the robust one under BC and NC conditions, they result in significantly increased test times for $P_3 \geq 1.2$. This analysis demonstrates that optimization methods assuming only one scenario of the input parameters, e.g., the nominal case, worst case, or best case, can result in ineffective test architectures in the presence of parameter variations.

Next, we summarize the results obtained for our benchmarks. For all examples, we assumed that at most two input parameters can vary and that they distribution is approximated by three points with 20%, 60%, and 20% probability. The SoCs q12710, d695, and h953 were solved exactly by modeling the entire problem as ILP and solving it with Xpress-MP within reasonable time (several minutes). This approach is too expensive in terms of computation time for larger SoCs, such as p93791 and p22810 because of the large number of constraints (2652, and 9042, respectively). Therefore, for these SOCs, we applied the heuristic method described in the previous section. We randomly partitioned each SoC into three core groups, optimized each group using exact ILP method, and then combined the results. After 1000 iterations, we selected a solution with the best total test time. The total runtime was under 30 minutes on a single machine.

For many input parameter combinations (W_{max}, P_{max} and uncertain parameters), the solutions assuming worst-case result in the same test time as robust solutions. The reason for it is that

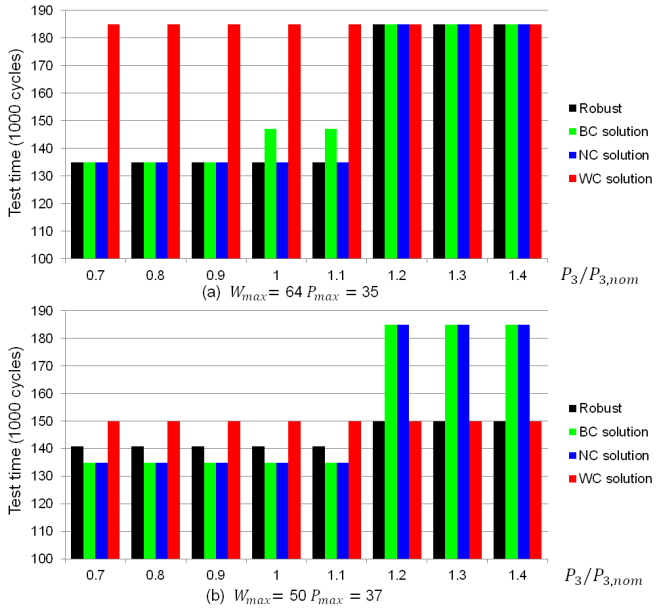

Fig. 6. Test time T of d695 as a function of input parameter P_3 for different architectures.

in these cases, variations in parameters are not severe enough to have an impact on compatibility of test of different cores. For some input parameter combinations, however, robust solutions are better than conventional optimization assuming worst-case values.

In Table II, we summarize the input parameters (W_{max}, P_{max}, and variable parameter names) that we used for our benchmarks, as well as the test times: the expected total test time assuming worst-case input parameters $T_{tot,exp,wc}$, and the expected total test time obtained by robust optimization $T_{tot,exp,rob}$. As our results show, depending on the design parameters, robust optimization can significantly improve the average test time. Even though we limited the number of parameters that can vary in our experiments, we already see an impact of uncertainty on the robustness of the solution. In practice, parameters of every core can vary. This implies that non-robust solutions will deviate from an optimal solution at least as much, or, most likely, even more than our experimental data show.

V. Limitations and Future Work

Our ILP-based robust optimization approach is scalable up to several cores and several input parameter scenarios. SoCs with more than ten cores require a heuristic approach because of the complexity of the ILP model. We have proposed a randomized divide-and-conquer heuristic to cope with this issue. However, this approach makes little use of the results obtained in previous iterations; therefore it can take a long time to find a good solution close to the optimum with high confidence. For future work, we will consider improving our randomized divide-and-conquer heuristic to accelerate the search for a good solution by taking previous results into account.

Another limitation of the current method is that a large number of parameter scenarios increases the complexity of the problem as well, such that it becomes intractable even for small SoCs. Overcoming this computational limitation is part of future work as well.

TABLE II
Input parameters and resulting test times for the SoC benchmarks.

	W_{max}	P_{max}	var. parameter	$T_{tot,exp,wc}$	$T_{tot,exp,rob}$	% impr.
q12710	6	40	P_3, P_4	54.6	54	2.9 %
d695	64	35	P_3, P_8	183.8	143.8	21.7 %
h953	10	40	P_1	209	204.4	3.2 %
p93791	200	35	P_1, P_5	237	234.2	1.2 %
p22810	60	60	P_{20}	1959	1731.8	11.6 %

VI. Conclusion

We have studied the problem of parameter uncertainty in the optimization of test architecture and test scheduling in core-based SoCs. Previous optimization solutions have been targeted at specific values of the inputs parameters. However, such "point solution" are not effective in the presence of uncertainties in input parameters, such as core test time and power consumption in test mode. These uncertainties are likely to arise in practical scenarios. We have therefore formulated a test-architecture and test-scheduling optimization framework based on robust optimization. We have described an ILP model for robust optimization that takes input parameter variations into account and provides a solution which minimizes the expectation of the test time of the SoC. We have also presented a heuristic method for targeting large designs. We have solved the robust optimization problem for five SoCs from the ITC'02 SoC Test Benchmarks and we have shown that robust solutions reduce test time more effectively compared to techniques that consider only one point in the input parameter space. In our experiments, we have only considered variations in test power; however, the proposed methodology can take variations in any parameter into account. This work is likely to lead to renewed interest in SOC test-architecture design and test-scheduling problems for more realistic scenarios.

References

[1] K. Banerjee et al. 3-D ICs: A Novel Chip Design for Improving Deep-Submicrometer Interconnect Performance and Systems-on-Chip Integration. *Proceedings of the IEEE*, 89(5):602–633, May 2001.

[2] E. J. Marinissen, S. K. Goel, and M. Lousberg. Wrapper Design for Embedded Core Test. In *ITC*, pages 911–920, 2000.

[3] E. Larsson and Z. Peng. An Integrated Framework for the Design and Optimization of SOC Test Solutions. *JETTA*, 18(4):385–400, 2002.

[4] V. Iyengar, K. Chakrabarty, and E.J. Marinissen. Test Wrapper and Test Access Mechanism Co-optimization for System-on-Chip. In *ITC*, pages 1023–1032, 2001.

[5] Y. Huang et al. Resource Allocation and Test Scheduling for Concurrent Test of Core-Based SOC Design. In *ATS*, pages 265–270, 2001.

[6] H. S. Hsu et al. Test Scheduling and Test Access Architecture Optimization for System-on-Chip. In *ATS*, pages 411–416, 2002.

[7] S. K. Goel and E. J. Marinissen. SOC Test Architecture Design for Efficient Utilization of Test Bandwidth. *TODAES*, 8(4):399–429, October 2003.

[8] E. Larsson, K. Arvidsson, H. Fujiwara, and Z. Peng. Efficient test solutions for core-based designs. *TCAD*, 23(5):758–775, 2004.

[9] B. Noia et al. Test-Architecture Optimization and Test Scheduling for TSV-Based 3-D Stacked ICs. *TCAD*, 30(11):1705 –1718, November 2011.

[10] P. Maxwell. Adaptive Testing: Dealing with Process Variability. *D&T*, 28(6):41–49, 2011.

[11] J. M. Mulvey, R. J. Vanderbei, and S. A. Zenios. Robust Optimization of Large-Scale Systems. *Operations Research*, 43(2):264–281, 1995.

[12] E. J. Marinissen, V. Iyengar, and K. Chakrabarty. A Set of Benchmarks for Modular Testing of SoCs. In *ITC*, pages 519–528, October 2002.

[13] K. Chakrabarty. Test Scheduling for Core-Based Systems Using Mixed-Integer Linear Programming. *TCAD*, pages 1163–1174, 2000.

[14] FICO Xpress-MP, http://www.fico.com/en/Products/DMTools/xpress-overview/Pages/Xpress-and-Optimization.aspx.