Uncertainty-Aware Robust Optimization of Test-Access Architectures for 3D Stacked ICs*

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

Erik Jan Marinissen
IMEC
B-3001 Leuven, Belgium

***Abstract*—3D integration using through-silicon vias offers many benefits, such as high bandwidth, low power, and small footprint. However, test complexity and test cost are major concerns for 3D-SICs. Recent work on the optimization of 3D test architectures to reduce test cost suffer from the drawback that they ignore potential uncertainties in input parameters; they consider only a single point in the input-parameter space. In realistic scenarios, the assumed values for parameters such as test power and pattern count of logic cores, which are used for optimizing the test architecture for a die, may differ from the actual values that are known only after the design stage. In a 3D setting, a die can be used in multiple stacks each with different properties. As a result, the originally designed test architecture might no longer be optimal, which leads to an undesirable increase in the test cost. We propose an optimization approach that takes uncertainties in input parameters into account and provides a solution that is efficient in the presence of input-parameter variations. We use integer linear programming (ILP) to formulate the robust test-architecture optimization problem, and the resulting ILP model serves as the basis for a heuristic solution that scales well for large designs. The proposed optimization framework is evaluated using the ITC'02 SoC benchmarks and we show that robust solutions are superior to single-point solutions in terms of average test time when there are uncertainties in the values of input parameters.**

I. Introduction

Three-dimensional stacking using through-silicon vias (TSVs) has enabled a new way to integrate even more devices in a single package and sustain Moore's law. TSVs are metal vias that provide electrical connections from one die to another through the silicon substrate [1]. TSVs outperform wire bonds used in conventional 3D stacking in interconnect density, performance and power consumption.

As is the case for conventional (2D) semiconductor products, 3D-SICs need to be tested for manufacturing defects. Due to the potential of 3D stacks to integrate a number of large SoCs, the complexity and the cost of test is increased. Therefore, 3D test requires careful attention in order to optimize test cost. Recent work on 3D test strategies have addressed this issue and presented several methods for test architecture optimization and test scheduling. These methods are based on exact optimization techniques such as integer linear programming (ILP) and heuristics such as rectangle packing [2]–[4].

A drawback of the above methods is that they consider known (constant) values for input parameters, which may differ from the actual values. This can lead to non-optimal decisions made at the design stage, increasing the test time. Variations in input parameters have several sources.

- At the design stage, some input parameters such as power consumption and test pattern count, are not known exactly and hence we need to rely on estimates, which can be inaccurate. This is valid for both 2D and 3D scenarios.
- In a 3D scenario, dies can be used as "off-the-shelf" items in multiple 3D stacks with different constraints on power consumption and available bandwidth for test data. A die that is optimized for a particular 3D stack can result in non-optimal test times in other stacks.

In Section II, we list examples of uncertainties in input parameters for 3D test architecture optimization and test scheduling. Neglecting these uncertainties and assuming a single point in the input-parameter space results in solutions that are not optimized for scenarios when input parameters change, leading to increased cost. Therefore, despite the large body of work on SOC test scheduling [5]–[7], the test scheduling and test-architecture problems in realistic scenarios for core-based 3D-SICs need more scrutiny.

To deal with the issue of uncertainties in input parameters in optimization, a method called "robust optimization" has been proposed in the past [8]. This approach takes variations in input parameters into account and finds a solution that does not deviate much from the optimum in case the parameters change. In [8], robust optimization is used for linear programming (LP); however, the same approach can also be applied to problems modeled by integer linear programming (ILP). Not surprisingly, we found that the complexity of robust ILP models for test-architecture design grows extremely fast with the number of cores and the number of parameter variations, such that solving these problems exactly becomes intractable. Therefore, in this paper, we apply an efficient heuristic method based on simulated annealing. Even though ILP is impractical for solving the complete robust optimization problem, we exploit it for solving sub-problems in our larger framework in order to optimize the test schedule for a given test architecture and a given set of input parameters.

The main contributions of this paper are as follows.

- We formulate the problem of robust TAM optimization and test scheduling for 3D-SICs in the presence of variations in input parameters. A robust solution may be inferior to a solution optimized for certain values of the input parameters in case their estimates were accurate; however, a robust

*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.

978-1-4799-0859-2/13/$31.00 ©2013 IEEE

solution performs better than non-robust solutions by staying closer to the optimum in the presence of variations.
- We develop an ILP model for robust optimization in the presence of variations in input parameters for the test architecture, such as power and available bit-width.
- For dies in a 3D stack that have a large number of embedded cores, we propose an efficient heuristic for robust optimization. The heuristic makes use of the simulated annealing algorithm [9] to quickly explore the search space and find reasonably good solutions, and uses ILP for solving smaller sub-problems.
- We show that robust solutions result in lower average test time compared to non-robust single-point solutions.

The remainder of this paper is organized as follows. Section II gives examples of uncertain parameters in optimization of 3D test architecture and test scheduling. Section III presents an overview of related prior work on test architecture design, robust optimization, and simulated annealing. In Section IV, we formulate an ILP model for robust optimization of 3D test architecture and present our heuristic method for robust optimization for systems containing a large number of cores. In Section V, we provide experimental results for various designs using ITC'02 test benchmarks [10]. Finally, Section VI concludes the paper.

II. Uncertain Parameters in Optimization of 3D Test Architecture and Test Scheduling

This section presents examples of uncertainty in parameters, including the test architecture, test power, test time, and compatibility of tests.

In a 3D setting, the test architecture can be designed to be flexible in order to enable compatibility between dies that will be stacked in different stacks. Even though the die manufacturer and the stack integrator have to agree on the physical interface of the dies to be integrated in a stack, the die-level test-access mechanism (TAM) may be reconfigurable, for instance, in order to increase the bandwidth and make maximum use of the channels provided by the test equipment. In this case, the width of the TAM becomes an uncertain parameter at the design stage, since it is not known *a priori* in which configuration the die will be tested. Figure 1 highlights an example of such a reconfigurable architecture with a variable number of test inputs. The figure shows the partitioning of cores at die level into four groups and the routing within groups as well as between groups. In this example, there are three possible configurations with different TAM bandwidths:

- An n-bit interface. In this case, only TI1 and TO1 are used as test inputs and test outputs, respectively, such that all groups are concatenated. Two cores can only be tested in parallel if they are assigned to different TAM wires, irrespective of their group assignment. For example, $c1$ and $c12$ cannot be tested in parallel in this configuration because they share wire 3.
- A $2n$-bit interface. TI1-TI2 and TO1-TO2 are used as test inputs and test outputs, respectively. Now, Group 1 and Group 2 are on separate test I/Os than Group 3 and Group 4.

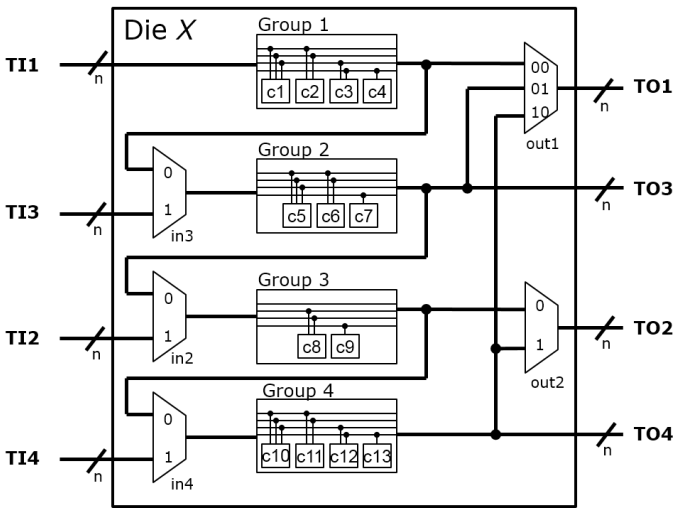

Fig. 1. Die-level architecture with reconfigurable multiple-width TAM.

Therefore, any core of Group 1 or Group 2 can be tested in parallel with all cores of Group 3 and Group 4 irrespective of the TAM assignment within the group, for instance $c1$ and $c12$.
- A $4n$-bit interface. In this configuration, all test inputs and test outputs are used, such that all groups can be tested independently from each other. However, the TAM-assignment within each group still limits the concurrent test of cores in the same group.

With this flexible 3D test architecture, the die can be integrated in different stacks that use n, $2n$, or $4n$ test inputs. Hence the die-level TAM width in test-architecture optimization for the stack becomes a distributed input parameter at the design stage.

This multiple-width die-level architecture is compatible with IEEE Std 1500, which is considered as basis for the proposed standard for a 3D-DfT architecture, IEEE Std P1838 [11]. Even though IEEE Std 1500 uses a fixed-width parallel TAM (WPI/O), we can introduce additional parallel modes that use a subset of the available test data signals. In the example shown in Figure 1, we can match the full $4n$-bit wide test-data interface to the wrapper-level WPI. The instruction set of the wrapper instruction register (WIR) can be extended by two extra instructions to enable $2n$-bit and $1n$-bit configurations. With this extension, the die can be integrated in stacks with different stack-level WPI width, which is determined by the amount of package-level pins available for testing or by the number of available channels of the test equipment. In case only a subset of the test I/Os is used, the unused inputs will be tied to ground and the unused outputs will be floating.

Another example of uncertainty in input parameters is power consumption during test. As 3D stacking offers integration of many dies in a single package, the problem of high power consumption (and resulting heat dissipation) becomes even more critical than in 2D SoCs. Recent papers have identified this problem and several methods for enhanced cooling in 3D-SICs have been proposed [12]–[14]. Nevertheless, power consumption during 3D-SIC test must be limited to prevent over-heating and over-testing. This limit may not be known before the 3D-SIC

978-1-4799-0859-2/13/$31.00 ©2013 IEEE

has been manufactured; hence, we must rely on estimates of core power consumption obtained through simulations. These estimates can be inaccurate, e.g., due to process variations, and the actual power limit can be different from the value assumed at the design stage. In this case, tests may need to be rescheduled to satisfy the power constraint and optimize the test time. As the test architecture cannot be changed anymore, the new schedule may result in high test times. Moreover, the same die may be integrated in different stacks, each with a different package with its unique thermal properties and hence different power limits.

Variations in test time can result due to test-set adaptation, which can be done in several ways:

- Additional types of tests, such as functional tests, might be added and because of limited test time budget, some scan tests that were considered in test architecture design might have to be dropped. If some types of tests are dropped, additional scan patterns can be included due to the available test time budget.
- Sometimes test patterns might have to be dropped due to need for truncation, which is typically motivated by limited tester depth or test time limitations. This scenario can arise if tests are developed in a distributed manner and they are combined later for manufacturing test.

As the test content is dynamically adjusted during test application, the test time will change and hence should be considered as an uncertain parameter at the design stage. A test architecture designed assuming nominal values for input parameters may not benefit from the updated test times, potentially limiting the advantages of test-set adaptation.

III. Related Prior Work

In this section, we present prior work on optimization of test architectures, robust optimization for LP problems, and simulated annealing.

A. Optimization of Test Architectures

Optimization of 2D SoC test architectures has been explored well in the past [5], [7], [15]–[17]. The proposed methods use ILP as well as heuristics such as rectangle packing in order to optimize the test architecture for minimum test time. The authors of [18] have addressed the issue of robustness of test architectures against variations in input parameters and proposed to solve the problem of robust optimization by using a daisy-chain architecture, matching the width of the core-level TAM to that of the system-level TAM, and testing one core at a time. The requirement of reconfigurable core wrappers is difficult to meet in practice because of complex scan-chain routing. In addition, some cores do not benefit from extra TAM width, wasting the bandwidth that could be used for testing other cores in parallel. In our work, we relax the requirement of flexible core wrappers, allowing for concurrent core testing at die level, and solve the robust test-architecture optimization problem by taking variations in input parameters into account.

Recent publications have been focusing on optimization of test architectures for 3D-SICs [2]–[4]. However, these methods assume known values for all input parameters, as well as a fixed stack-level TAM width.

A wrapper-based daisy-chain architecture has been proposed for 3D-SICs [19], [20], which is under consideration for standardization by the IEEE P1838 Working Group [21]. The proposed wrapper is based on IEEE Std 1500 and allows for serial and parallel configurations. The width of the TAMs in parallel configurations can be different in pre-bond and post-bond modes; however, the post-bond parallel TAM width is fixed. Nevertheless, we can implement additional parallel configurations with variable TAM width and still ensure compatibility with the originally proposed wrapper. The flow for 3D-wrapper insertion has been automated in [22] and this framework can be easily extended to support extra parallel post-bond tests with different TAM widths. In our work, we assume a daisy-chain architecture at stack-level.

Our motivation for a reconfigurable die-level TAM is based on the reconfigurable core wrapper approaches presented in [23], [24]. The proposed core wrappers allow for an adjustable core-level TAM. The scan-chains are partitioned into groups and interconnected such that the total test time is optimal for every configuration. We extend this idea to 3D wrappers and propose a reconfigurable TAM at die-level, as shown in Figure 1.

B. Overview of Robust Optimization

The main idea of the robust-optimization method proposed in [8] is to consider a number of scenarios in each of which variable input parameters take certain values and to find a solution that stays near optimum in all scenarios. Even though a robust solution sacrifices optimality in some scenarios, it performs better on average than a single-point solution.

The following text summarizes the method presented in [8]. It distinguishes between *design* and *control* variables:

- $x \in \mathbf{R}^{n_1}$, which denotes the vector of size n_1 containing the *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. In our work, design parameters describe the test architecture, which is fixed at the design stage.
- $y \in \mathbf{R}^{n_2}$, which denotes the vector of size n_2 containing the *control* variables. These parameters can be adjusted once the values of the uncertain parameters are known. In our work, control parameters represent the test schedule as it can be changed even after the design has been fixed.

With design and control variables, the LP model is formulated as:

Objective:

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

Subject to:

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

where A, b, and c are constant parameters and d, B, C, and e can be uncertain. In presence of parameter variations, a set

of scenarios $\Omega = \{1, 2, 3, \ldots, S\}$ is introduced. Each scenario $s \in \Omega$ occurs with the probability p_s ($\sum_{s\in\Omega} p_s = 1$), where the uncertain parameters take the values $\{d_s,\ B_s,\ C_s,\ e_s\}$. A robust solution of this LP is defined as a solution that does not move much from the optimum if a scenario $s \in \Omega$ occurs, i.e. the deviation of the cost function is minimized for the range of scenarios.

Using the set of control variables for each scenario $\{y_1, y_2, \ldots, y_s\}$ and the set of error vectors $\{z_1, z_2, \ldots, z_s\}$ that measure the allowed infeasibility, the mathematical model for robust optimization is formulated 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 the function to minimize $\sigma(\cdot)$ can be either the expectation of the cost function $\sigma(\cdot) = \sum_{s\in\Omega} p_s(c^T x + d_s^T y_s)$ 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 $\rho(z_1, \ldots, z_s)$ represents a penalty function to limit violations of the control constraints by assigning a weight ω. In this work, we set this function to zero.

As an example of robust optimization, let us consider 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. Table I 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 conventional (non-robust) solutions are optimal for the corresponding scenarios; however, they show relatively large increase of the cost F if the scenario and hence the input parameters change. In contrast, in case of the robust solution, we sacrifice optimality for both scenarios, but win in terms of the expectation of the cost, if $s1$ and $s2$ occur with equal probabilities. Recently, we have proposed a method for robust optimization of test architecture and test scheduling in 2D SoCs [25]. We have formulated an ILP model for the robust optimization problem which is suitable for small SoCs and a small number of scenarios. For larger SoCs, a simple randomized divide & conquer heuristic has been proposed. However, this approach does not consider a reconfigurable TAM, which is a likely scenario in 3D-SICs. Therefore, while [25] is an important first step towards robust optimization for SoCs, it is not adequate for the additional uncertainty in 3D designs of reconfigurable die-level TAMs that interface to other dies.

TABLE I
SOLUTIONS OF THE LP PROBLEM FOR (A) $s1$, (B) $s2$, AND (C) TAKING BOTH $s1$ AND $s2$ INTO ACCOUNT.

	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**

C. Simulated Annealing

Due to the high complexity of the test scheduling for SoCs and 3D-SICs with a large number of cores, efficient heuristics are required in order to find an acceptable solution within a reasonable time. We propose a heuristic using the simulating annealing algorithm [9]. In the past, a number of simulated annealing-based techniques for test scheduling as well as for 3D test architecture optimization has been introduced [26]–[28]. In this work, we also exploit the concept of simulated annealing in order to develop a heuristic for the robust optimization problem. The main idea of simulated annealing is to iteratively explore the search space by perturbing the solution at each step and accepting an inferior solution with the probability

$$Pr = \exp\left(\frac{-\Delta C}{T}\right),$$

where ΔC is the difference in the cost function and T is a parameter decreasing at each iteration step. The parameter T is analogous to the current temperature in a real annealing process. At each iteration step, the solution is perturbed and T decreased. In the beginning, when the T is high, a new inferior solution is likely to be accepted in order to prevent falling into a local optimum before a number of areas in the solution space have been explored. With decreasing T, this probability decays and eventually, we stay in one area and find a local optimum before the T reaches a specified threshold.

IV. ROBUST OPTIMIZATION OF 3D TEST ARCHITECTURE

The method proposed in [8] targets robust optimization problems that can be modeled as an LP problem and hence solved in polynomial time. However, many practical optimization problems are $\mathcal{NP}$-hard and cannot be solved in polynomial time using LP. Examples of such problems include test-architecture optimization and test scheduling.

In this section, we formulate a mathematical model for robust optimization of 3D test architecture and test scheduling and map it to an ILP problem in order to solve the problem using existing solvers. However, this approach is only practical for small problem instances. For larger instances, we propose an efficient heuristic based on simulated annealing. This heuristic solves the problem iteratively by finding TAM architecture candidates and evaluating them using ILP. In this sense, even though the ILP model is not used for the overall optimization problem, it is useful for solving sub-problems in the the heuristic approach.

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

In this work, we assume that the dies are stacked in a 3D-SIC with a daisy-chain stack-level test architecture, i.e., the stack is

Objective:

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

Subject to:

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

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

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

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

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

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

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

Fig. 2. Mathematical programming model for non-robust test architecture optimization.

tested die-by-die in a modular fashion such that each die gets the full test bandwidth when it is tested. Therefore, we exclude the option of concurrent test of embedded cores that are spread over multiple dies in the stack and focus on the test-time optimization at die level. Such an assumption is justified by power constraints that are likely to be important during test application in 3D-SICs.

Let us denote the maximum width of the die-level test access mechanism (TAM) W_{max}. The die under test has N cores that need to be tested. Each core i has the TAM width W_i, test time T_i, and peak power during test P_i. Cores that share a die-level TAM bit cannot be tested in parallel. In addition, the total power during test must not exceed P_{max} at any time in order to prevent voltage droop and high heat dissipation, resulting in potential yield loss, over-testing, or even permanent damage of the 3D-SIC. Power limit is the only constraint we use here as example; however, we can easily extend our framework with extra constraints, such as shared test resources that make test of certain groups of cores incompatible [29]. The objective is to assign cores to the die-level TAM and schedule the test such that the total test time is minimized.

Figure 2 shows our mathematical programming model for conventional (non-robust) co-optimization of test architecture and test scheduling. This model serves as a basis for robust optimization. In the following, we describe the details of the model.

The integer variables w_i, $1 \leq i \leq N$ indicate the lowest TAM wire to which core i is connected. The last bit of the core TAM is hence connected to $w_i + (W_i - 1)$. Line (3) adds the constraint the limit on maximum available TAM width.

Next, let the array of integer variables t_i, $1 \leq i \leq N$ denote the start times of core tests. The test of core i will therefore finish at $t_i + T_i$. The total test time is the maximum of all finish times $T_{tot} = \max_i\{t_i + T_i\}$, and the objective is to minimize it. Lines (2) and (8) represent this information.

In order to restrict parallel testing of two cores that "overlap" in die-level TAM, we introduce a matrix of binary variables q, such that

$$q_{i,j} = \begin{cases} 1 & \text{if Core } i \text{ and Core } j \text{ do not share a TAM wire} \\ 0 & \text{else} \end{cases} \tag{10}$$

The variables $q_{i,j}$ and w_i are therefore constrained as shown in Line (4). The start times t_i are constrained through (a) the TAM assignment and (b) test power. The constraint due to TAM assignment is modeled as follows. We introduce a matrix of binary variables r, such that $r_{i,j}$ is 1 if the test of Core i starts while test of Core j is in progress. This is modeled in Line (6). In case that $t_i = t_j + T_j$, then $r = 0$, which implies that the test of Core i starts right after the test of Core j and therefore the tests do not overlap. The constraint of the start times due to TAM overlap is formulated in Line (7): tests can only overlap if the cores do not share TAM wires. The constraints (4) and (6) are non-linear but can be linearized using standard linearization techniques in order to convert the model to ILP. The power constraint is modeled as follows. For each core, we check at the starting time of each core whether the total power of the cores being tested at this time exceeds P_{max}. This constraint is captured in Line (9).

Solving this problem provides an optimal core TAM assignment (w_i) and an optimal test schedule (t_i) for a given set of input parameters (one scenario).

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

The model in Figure 2 neglects potential variations of input parameters and assumes constant values instead. However, in practice, some of the input parameters may change. For instance, test power limits may vary for pre-bond, post-bond, and final test after packaging. In addition, a die can be manufactured to be stacked in different 3D-SICs that have different requirements on power consumption. The originally obtained test schedule (vector t_i) may not be optimal for all cases or it can even violate power constraints. However, the *design* variables related to the test architecture cannot be adjusted *a posteriori* for optimal scheduling. Therefore, for each scenario, the design variables, such as w_i become constraints, fixing all variables in lines (3)-(5) of the non-robust problem.

Another uncertain parameter that we take into account is the width of the die-level TAM. As highlighted in Section I, we can introduce a reconfigurable parallel TAM in order to enable extra bandwidth for test data in case the stack-level test architecture allows it. To model that, we introduce a vector of integer variables g_i that holds the group number for each Core i and an integer variable *conf* that encodes the configuration of the TAM. In our examples, we limit the number of groups to four, hence $g_i \in \{1,2,3,4\}$ and the number of configurations to three, hence $conf \in \{1,2,3\}$. Since g_i defines the partitioning of the cores and therefore the die-level test architecture, this is a design variable. In contrast, the value of *conf* depends on the stack in which the die will be integrated. Therefore, *conf* is a control variable and can be adjusted after the die has been manufactured.

In order to avoid non-robust solutions, we optimize the test architecture for all scenarios simultaneously and minimize the expectation of the total test time. For this, we use the approach described in [8] and extend the mathematical model described in the previous section.

In general, a number of uncertain input parameters for 3D test

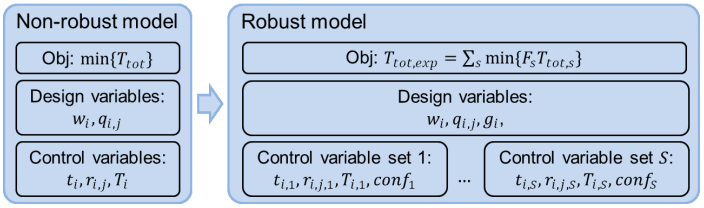

Fig. 3. Extension of non-robust ILP model to robust ILP model.

architecture optimization can be uncertain. In our work, we limit the set of variable parameters to P_{max} and *conf* for simplicity reasons; however, our framework is easily extendable with other uncertain parameters, for instance, core test time T_i. In addition, we assume known probability distributions for P_{max} and *conf* and take a number of discrete points (scenarios) in the distribution range. The parameter F_s denotes the probability that P_{max} and *conf* will take the values $P_{max,s}$ and $conf_s$ where s is one of the potential scenarios from the set $\Omega = \{1, 2, 3, \ldots, S\}$. In practice, these probabilities can be estimated using legacy knowledge from prior designs and manufactured chips. Even though it is difficult to obtain a precise distribution of uncertain parameters, an estimated distribution is likely to result in a more robust solution compared to a single-point approach.

Similar to the method proposed in [29], we classify all variables as either *design* or *control* variables. In case of 3D test architecture optimization, design variables are w_i, $q_{i,j}$, and g_i as they define the test architecture and hence cannot be changed after the design stage. In contrast, the test schedule can be adjusted 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 of the non-robust ILP model that include these variables 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. In Figure 3, we show an overview of the extension of the non-robust ILP model to the robust one. Due to multiple scenarios that can occur, the number of the control variables and the number of the constraints including these variables scales linearly with the number of scenarios S, drastically increasing the complexity of the problem and motivating the need for efficient heuristics. In case $S = 1$, the robust model reduces to the the non-robust model as a special case.

A solution to the robust optimization ILP problem gives us a "robust" core TAM assignment (w_i), core partitioning (g_i) and optimal test schedules ($t_{i,s}$) for each scenario. Since the number of variables and constraints grows rapidly with the number of cores and the number of possible scenarios, the exact ILP method becomes intractable for a die with many cores. In our experiments, the commercial ILP solver was not able to provide an optimal solution even for an 8-core die within 24 hours. Therefore, we focused in this work on developing an efficient heuristic to handle dies with more than 20 cores.

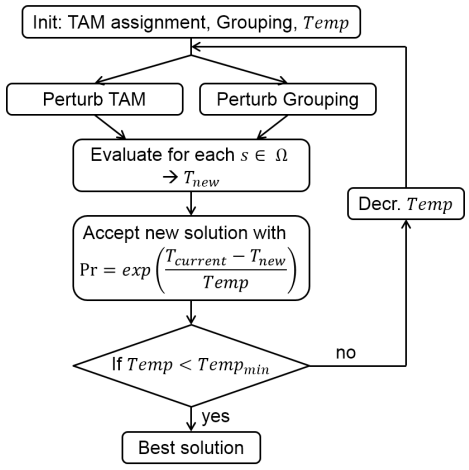

Fig. 4. SA-based Robust Optimization Flow.

C. Heuristic Method for Robust Optimization Based on Simulated Annealing

Due to the high complexity of the robust optimization problem for dies with even a small number of cores, we need alternative methods to handle realistic designs. In this work, we propose a heuristic method based on simulated annealing (SA).

Figure 4 provides on overview of our SA algorithm. The following text describes the main idea of the algorithm. Later, we will focus on the detailed description of the important blocks. In the beginning, we set the initial temperature, and find an initial grouping of cores and a TAM assignment of each core to the bus of n wires. Next, we randomly perturb the TAM assignment by rearranging some of the cores. The new test architecture is evaluated by finding an near-optimal test schedule for each scenario $s \in \Omega$. The total test times for each scenario are weighted with the probability F_s, resulting in the new expectation of the test time T_{new}. If the new solution is superior to the current solution, the new test architecture is accepted. Otherwise it is accepted with the probability

$$Pr = \exp\left(\frac{T_{current} - T_{new}}{Temp}\right). \quad (11)$$

If the temperature is above a specified threshold $Temp_{min}$, the temperature is decreased by a specified factor and the algorithm goes back to the perturbation phase. This time, the grouping of the cores is slightly perturbed and the new test architecture is evaluated again. Perturbation of TAM and the grouping alternate between each iteration. Once the temperature has fallen under $Temp_{min}$, the algorithm stops and the best solution so far is provided as the result. In the following, we describe each step in detail.

The die-level TAM assignment is defined by the vector $w = [w_1, w_2, \ldots, w_N]$, as described in the mathematical model in Figure 2. The number of all feasible variations of w can be extremely large and hence completely random TAM perturbations would require an impractically large number of iterations. In our algorithm, we try to improve our solution by using the following properties of w.

First, there is a number of TAM assignments that correspond

to the same overlap matrix q, which is defined in Equation (10). Since only q directly limits which cores can be tested in parallel (Equation 7), all these TAM assignments will result in the same optimal schedule and hence the same minimum test time. Therefore, it is sufficient to perturb the matrix q and to find which one gives the best schedule. The actual solution is then any TAM assignment that maps to this q.

Second, assume a TAM assignment in which Core i and Core j overlap in TAM and therefore cannot be tested in parallel ($q_{i,j} = q_{j,i} = 0$). Let us change $q_{i,j} = q_{j,i}$ to 1, which means Core i and Core j are moved away from each other such that they do not overlap in TAM anymore. All test schedules that were feasible for the previous TAM assignment will remain feasible. Potentially, we can improve the test time by testing Core i and Core j in parallel. Therefore, it is enough to consider the new matrix as a candidate and discard the old (inferior) matrix. We exploit this property to find an initial TAM candidate and to perturb it. First, we start with an identity matrix (all cores share TAM wires). This is the most inferior solution as it only allows for serial testing. However, it is guaranteed that this TAM assignment is feasible because we require that $W_i \leq W_{max}$ for all i. Next, we try to improve the candidate by setting some randomly chosen entries to 1. By doing this, we force certain cores to be assigned to distinct TAM wires and therefore increase the minimum required group-level TAM width for the new q. Once this width exceeds the given maximum TAM width W_{max}, q becomes infeasible, hence we need to undo the change and try to change another entries of q to 1. After we have tried setting all entries to 1, we stop and receive a TAM candidate that is evaluated in the next step.

The checking of q is done by mapping it to the maximum-weight clique problem and solving it using an existing algorithm [30]. The mapping is done as follows. Each Core i is represented by a vertex v_i with the weight W_i. An edge between two vertices v_i and v_j exists if $q_{i,j} = 1$. The objective of the algorithm is to find a clique (a set of vertices in which there is an edge between every pair of two vertices) with the maximum total weight. In our original problem this clique is a set of cores that can be tested simultaneously and "span" the maximum width. If the weight of the maximum-weight clique exceeds W_{max}, the given TAM assignment is infeasible, as it requires more TAM wires than available. If the maximum weight does not exceed W_{max}, q is feasible and we can find a corresponding TAM assignment. The maximum-weight clique problem is $\mathcal{NP}$-hard; however, we can solve it within microseconds for realistically sized dies with $N < 100$. Even though we call the check function $O(N^2)$ times during each perturbation, the CPU time for this step is negligible compared to the evaluation part.

Perturbation of a given q is done as follows. We "reset" a number of elements back to 0 and try to randomly fill the matrix with 1 until no zero-element can be set to 1. If the number of elements that is reset is large enough, we will receive a different TAM candidate with high probability. In our implementation, we randomly choose a row and a column and reset their elements. However, our framework can be easily adjusted to perform a different type of perturbation.

The grouping of cores is stored in the vector $g = [g_1, g_2, \ldots, g_N]$. The initial grouping is assigned randomly. In order to perturb g, we perform either of the following operations with equal probability: (a) we swap two randomly chosen cores from different groups, (b) we move a randomly chosen core to a different group.

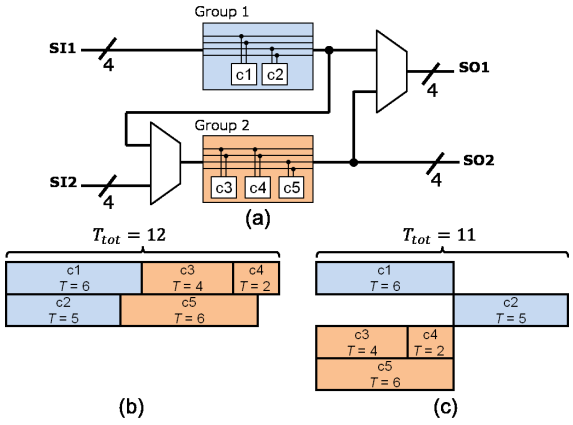

Fig. 5. Test Architecture 1: (a) TAM assignment and partitioning, (b) schedule for 4-bit configuration, (c) schedule for 8-bit configuration.

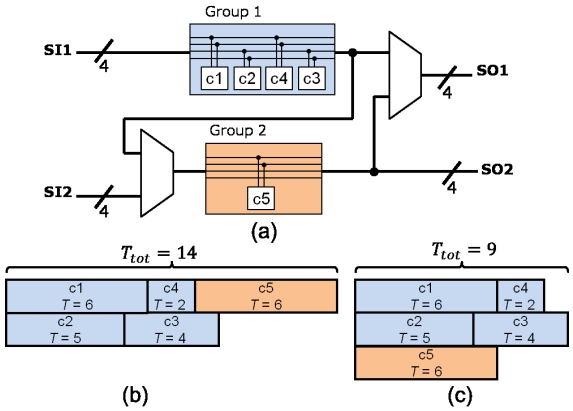

Fig. 6. Test Architecture 2: (a) TAM assignment and partitioning, (b) schedule for 4-bit configuration, (c) schedule for 8-bit configuration.

In the evaluation phase, we calculate an expectation of the total test time for the given TAM assignment and grouping. Since all design variables are fixed, the solutions (schedules) for each scenario $s \in \Omega$ are not "coupled" between each other. This allows us to partition the evaluation problem in S sub-problems, where each sub-problem is a non-robust scheduling problem with the constraint set s and the given TAM assignment q and grouping g. Since this partitioning greatly reduces the complexity of the evaluation problem, we use ILP for it. Once the test schedules for each scenario have been generated, we calculate the expectation of the total test time and accept the new candidate with the probability as shown in Equation (11).

During SA iterations, we keep track of the best solution in terms of the expectation of the test time. Once the temperature falls below a threshold, the best combination of a TAM assignment and grouping is provided as a result.

TABLE II
DESIGN PARAMETERS OF SOME OF THE BENCHMARK SOCS USED IN EXPERIMENTS.

	h953			d695			p93791			p22810		
core	W	T	% FF	W	T	% FF	W	T	% FF	W	T	% FF
1	4	118	24.2	16	94	10.0	46	113	26.5	10	1021	5
2	2	3	14.0	4	56	3.3	44	75	8.3	29	432	9
3	2	1	1.4	32	50	22.5	46	68	7.6	24	214	9
4	4	1	1.8	4	44	2.8	46	62	3.4	4	38	1
5	4	13	10.4	32	35	25.5	46	42	10.6	8	83	2
6	4	33	15.8	16	31	8.3	46	42	10.6	11	76	3
7	8	57	32.4	1	24	0.5	46	40	8.5	4	1	1
8				32	6	27.0	46	36	4.7	3	79	1
9							35	32	7.3	6	30	1
10							43	32	7.1	1	9	1
11							44	21	4.8	4	22	1
12							11	15	0.6	5	438	1
13										3	25	1
14										4	95	1
15										10	848	4
16										3	45	1
17										7	46	4
18										5	27	1
19										18	389	9
20										31	724	46
21										1	1	1
22										5	26	1

V. EXPERIMENTAL RESULTS

We present experimental results to evaluate the proposed heuristic method for robust optimization. The framework is implemented in C++ using BCL libraries of the Xpress-MP tool [31] to solve ILP problems.

First, we demonstrate the effect of robust optimization using a simple example. Next, we show experimental results obtained with publicly available benchmarks.

Consider a die containing 5 cores with equal TAM widths $W_i = 2$ and the test times {6,5,4,2,6} in some arbitrarily chosen units. The die-level TAM can be reconfigured in 4-bit (n) and 8-bit (n) modes. Due to the power constraints, only three cores can be tested at a time. Figure 5 and Figure 6 show two test architectures (core TAM assignment and core partitioning) with the corresponding optimal schedules for both configurations, resulting in test times of {12,11} and {14,9}, respectively. If the die will be used in both configurations with equal probabilities, the expectation of the test time is 11.5, regardless of the architecture. However, if one scenario is more likely than the other, one architecture will perform better on average than the other. Therefore, this probability must be considered during optimization of the test architecture.

Since there are no publicly available 3D-SIC benchmarks, we applied our methods to SoCs from the ITC'02 SoC Test Benchmark set [10] that we use as dies in a 3D stack in our experiments. Table II summarizes the relevant design data of the benchmarks, h953, d695, p93791, and p22810, including the core TAM width W, the core test time T in units of 1000 clock cycles, and the percentage of the scan flip-flop count in relation to the total scan flip-flop count. Cores without scan chains were excluded from the set; therefore, the dies have 7, 8, 12, and 22 cores to be connected to the system TAM, respectively. In addition, we used the combination of the SoCs h953 and d695 as the fifth die. We approximated the test time of each core by multiplying the length the longest scan chain with the test pattern count.

TABLE III
INPUT PARAMETERS AND RESULTING TEST TIMES FOR THE SOC BENCHMARKS.

Benchmark	$W_{m,n}$	$P_{m,n}$	Prob: 0.2-0.6-0.2			Prob: 0.1-0.8-0.1		
			T_{nom}	T_{rob}	impr	T_{nom}	T_{rob}	impr
h953	10	60	156.7	145.6	7%	176.0	142.8	19.4%
d695	32	50	144.0	131.0	9%	159.7	125.8	21.2%
p93791	100	45	223.5	207.9	7%	231.3	203.9	11.8%
d695_h953	32	60	191.7	161.0	16%	189.3	179.0	5.4%
p22810	50	60	1844.0	1634.0	11.4%	2364.6	1847.3	21.8%

We assume that the peak power during test depends linearly on the amount of logic that is switching, which is proportional to the amount of scan flip-flops. Therefore, we set the constraint on power during test by limiting the fraction of active scan flip-flops. For instance, if we set $P_{max} = 0.5$ for h953 then Core 1 cannot be tested in parallel with Core 7 because the power would exceed the limit and the schedule would become infeasible. In a real design flow, P_{max} and P_i will be estimated using simulations. Due to inaccuracies of these simulations, these data will be subject to variations that can be handled through robust optimization. In addition, P_{max} may vary significantly in case a die is integrated in different 3D stacks with different thermal properties. In practice, each of these parameters can vary independently from each other, which increases the complexity of the robust optimization problem. Out of these parameters, we only consider P_{max} as an uncertain parameter in this paper. We assume three discrete points for P_{max}, such that the nominal value occurs with the probability of 0.6 and the other two points $P_{max} \pm \Delta P$ with the probability of 0.2 each. We repeated the experiments for different probabilities of P_{max}: 0.1, 0.8, 0.1.

An additional input parameter that is subject to variations in our framework is the configuration of the TAM. We limit the possible configurations to n-bit, $2n$-bit, and $4n$-bit configurations, as shown in Figure 1, and assign equal probabilities to them. As both *conf* and P_{max} can take three different values independently from each other, the total number of scenarios we consider is $S = 9$. In contrast to the exact ILP model, the complexity of our heuristic algorithm grows only linearly with the number of scenarios, as we solve the scheduling problem for each scenario independently. Therefore, we can easily extend our framework to handle more uncertain parameters while maintaining reasonable CPU-runtimes.

Due to the high complexity of the ILP model for dies with many cores, the ILP solver did not terminate within reasonable time even for SoCs with seven cores. Therefore, we focus on experiments using the proposed heuristic method based on simulated annealing. We limited the number of SA iterations to 200 in order to experiment with the setting and collect the results in a reasonable time. In a real application, this number of SA iterations can be increased in order to improve the quality of the solution. The CPU time in our experiments varied from 20 minutes for small benchmarks to a few hours for the 22-core benchmark.

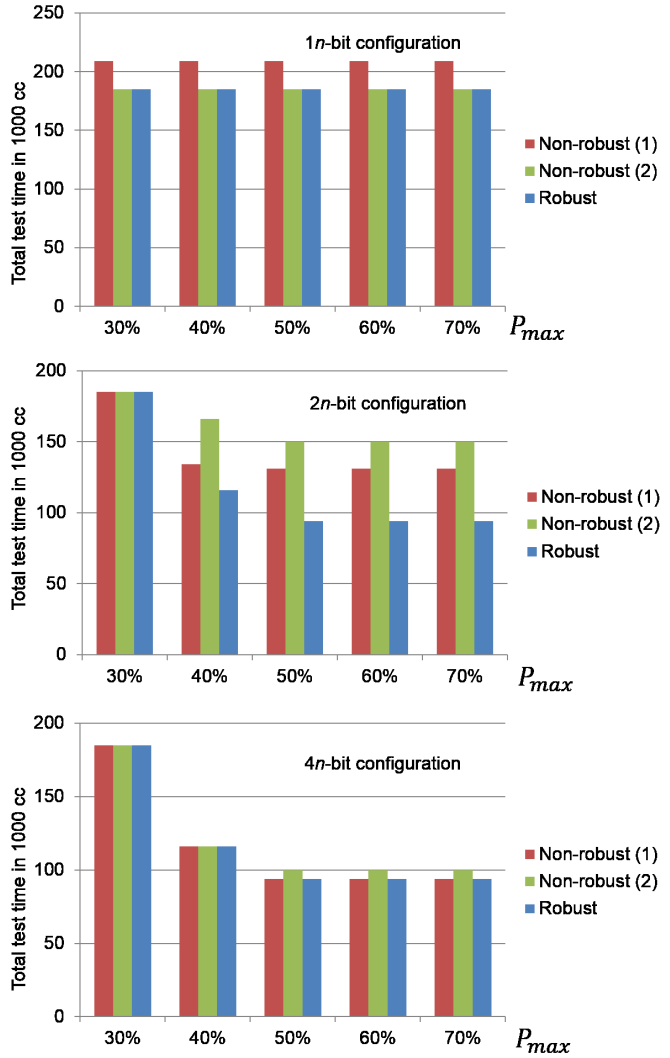

Fig. 7. Evaluation of the robust and non-robust solutions for d695 in $1n$-bit, $2n$-bit, and $4n$-bit configuration for different P_{max}.

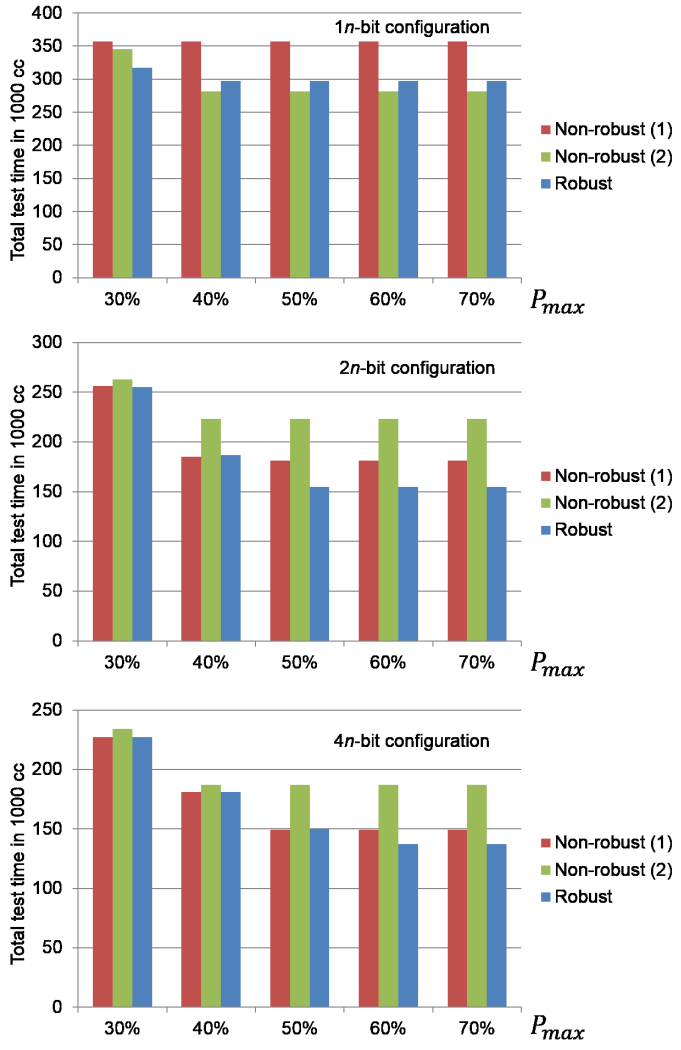

Fig. 8. Evaluation of the robust and non-robust solutions for p93791 in $1n$-bit, $2n$-bit, and $4n$-bit configuration for different P_{max}.

In order to demonstrate the advantage of robust solutions we did the following experiment. For each SoC, we obtained a non-robust solution assuming a single points in the input parameter space (the nominal value of P_{max} and the $4n$ configuration). For the robust solutions, we considered nine scenarios for variable P_{max} and *conf*. The non-robust solutions were evaluated for the same scenarios as the robust solutions in order to compare their performance in the presence of uncertainties in the input parameters. Table III shows the input parameters $W_{m,n}$ (TAM width for the case of $1n$-bit configuration), $P_{m,n}$ (nominal value of P_{max}), and the reduction in the expectation of test time for the robust solutions compared to the non-robust case. As the results indicate, test architectures that were optimized using the proposed robust optimization method outperform the solutions obtained assumed known (constant) input parameters in terms of the expectation of test time.

We compared robust and non-robust solutions for d695. We obtained two non-robust solutions using the $1n$-bit and $4n$-bit configuration, respectively, and $P_{m,n} = 50\%$. The robust and non-robust solutions were evaluated for a number of scenarios with different configurations and different power limits. We also included values of P_{max} that are outside of the range used for robust optimization in order to see how both robust and non-robust solutions perform in extreme cases.

Figure 7, shows the evaluation results for all configurations. During robust optimization, the algorithm exploits the relaxed constraints on P_{max} and W_{max} and adjusts the test schedule for shorter test times. As the algorithm for the non-robust optimization ignores potential variations in input parameters, the obtained non-robust test architecture is less effective than the robust one. The non-robust solution (1) was optimized for the $4n$-bit configuration and therefore performs well in that case. Similarly, as the non-robust solution (2) was optimized for the $1n$-bit configuration, it is optimal in this case. The robust solution results in the same total test time for $1n$ and $4n$ as the single-point solutions optimized for these configurations. In the $2n$-bit configuration, the robust solution outperforms other solutions. We performed a similar experiment with p93791 and d695_h953 benchmarks and observed similar behavior of robust and non-robust solutions, as shown in Figure 8 and Figure 9, respectively.

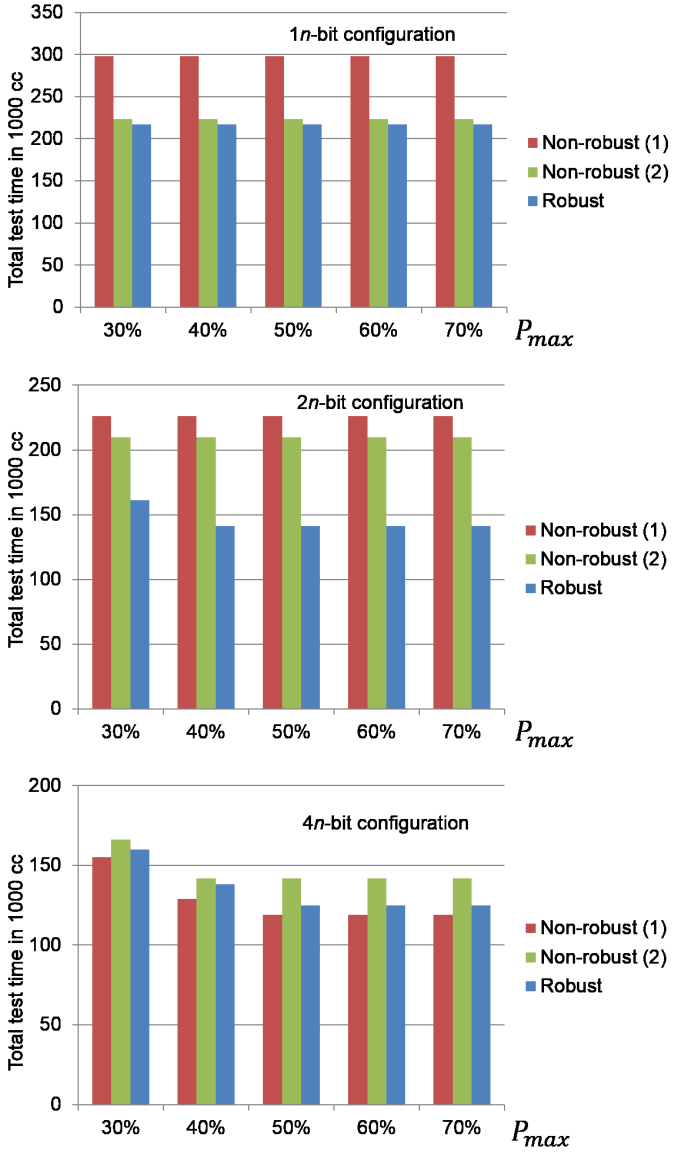

Fig. 9. Evaluation of the robust and non-robust solutions for d695_h953 in $1n$-bit, $2n$-bit, and $4n$-bit configuration for different P_{max}.

VI. Conclusion

We have presented a method for robust optimization of 3D test architecture and test scheduling in the presence of input parameter variations. We have formulated a mathematical model for the robust optimization problem using ILP. Even though the ILP model for the complete problem requires long CPU times for realistic dies with many cores, we can use this model in a heuristic method in order to solve small sub-problems. Our heuristic is based on the widely used simulated-annealing technique and optimizes the TAM assignment of the cores by perturbation and evaluation of the solution for the given scenarios of uncertain parameter values. Results show that neglecting variations in input parameters will result in inefficient single-point solutions with increased test time. The proposed method scales well with the complexity of the die and with the number of scenarios. The designed framework can easily be extended with additional constraints and uncertain parameters.

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] Li Jiang et al. Layout-Driven Test-Architecture Design and Optimization for 3D SoCs under Pre-Bond Test-Pin-Count Constraint. In *ICCAD*, pages 191–196, November 2009.

[3] B. Noia et al. Test-Architecture Optimization for TSV-Based 3D Stacked ICs. In *ETS*, pages 24–29, 2010.

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

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

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

[7] 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.

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

[9] P.J. van Laarhoven and E.H.L. Aarts. *Simulated Annealing: Theory and Applications*, volume 37. Springer, 1987.

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

[11] IEEE Standard Assosiation, http://standards.ieee.org/develop/project/1838.html.

[12] J.H. Lau and T.G. Yue. Thermal Management of 3D IC Integration with TSV (Through Silicon Via). In *Electronic Components and Technology Conference (ECTC)*, pages 635–640, 2009.

[13] Y.K. Joshi, A.G. Fedorov, Y.J. Lee, and S.K. Lim. Thermal Characterization of Interlayer Microfluidic Cooling of Three-Dimensional Integrated Circuits with Nonuniform Heat Flux. *Journal of Heat Transfer*, 132, 2010.

[14] A. Dembla, Y. Zhang, and M.S. Bakir. Fine Pitch TSV Integration in Silicon Micropin-Fin Heat Sinks for 3D ICs. In *Proceedings IEEE International Interconnect Technology Conference (IITC)*, pages 1–3, 2012.

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

[16] E. Larsson, K. Arvidsson, H. Fujiwara, and Z. Peng. Efficient Test Solutions for Core-Based Designs. *TCAD*, 23(5):758–775, 2004.

[17] G. Giles, J. Wang, A. Sehgal, K.J. Balakrishnan, and J. Wingfield. Test Access Mechanism for Multiple Identical Cores. In *ITC*, pages 1–10, 2009.

[18] T. Waayers, R. Morren, and R. Grandi. Definition of a Robust Modular SOC Test Architecture; Resurrection of the Single TAM Daisy-Chain. In *ITC*, pages 10–pp, 2005.

[19] E.J. Marinissen, J. Verbree, and M. Konijnenburg. A Structured and Scalable Test Access Architecture for TSV-based 3D Stacked ICs. In *VTS*, pages 269–274, 2010.

[20] E.J. Marinissen, C.C. Chi, J. Verbree, and M. Konijnenburg. 3D DfT Architecture for Pre-bond and Post-bond Testing. In *3DIC*, pages 1–8, 2010.

[21] IEEE 3D-Test P1838 Working Group. http://grouper.ieee.org/groups/3Dtest/.

[22] S. Deutsch et al. Automation of 3D-DfT Insertion. In *ATS*, pages 395–400, 2011.

[23] S. Koranne. A Novel Reconfigurable Wrapper for Testing of Embedded Core-Based SOCs and Its Associated Scheduling Algorithm. *Journal of Electronic Testing*, pages 415–434, 2002.

[24] E. Larsson and Z. Peng. A Reconfigurable Power-Conscious Core Wrapper and Its Application to SOC Test Scheduling. In *ITC*, pages 1135–1144, 2003.

[25] S. Deutsch and K. Chakrabarty. Robust Optimization of Test-Architecture Designs for Core-Based SoCs. In *ETS*, 2013.

[26] W. Zou et al. SOC Test Scheduling Using Simulated Annealing. In *Proceedings IEEE VLSI Symposium on Circuits (VLSI)*, pages 325–330, 2003.

[27] L. Jiang, L. Huang, and Q. Xu. Test Architecture Design and Optimization for Three-Dimensional SoCs. In *DATE*, pages 220–225, 2009.

[28] C.Y. Hsu et al. 3D IC Test Scheduling Using Simulated Annealing. In *VLSI Design, Automation, and Test (VLSI-DAT), 2012 International Symposium on*, 2012.

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

[30] P.R.J. Östergård. A New Algorithm for the Maximum-Weight Clique Problem. *Nordic Journal of Computing*, 8(4):424–436, 2001.

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