Test-Cost Optimization and Test-Flow Selection for 3D-Stacked ICs*

Mukesh Agrawal
Electrical & Computer Engineering
Duke University, Durham, NC 27708

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

Bill Eklow
Cisco Systems, Inc.
San Jose, CA 95134

***Abstract*—Three-dimensional (3D) integration is an attractive technology platform for next-generation ICs. Despite the benefits offered by 3D integration, test cost remains a major concern, and analysis and tools are needed to understand test flows and minimize test cost. We propose a generic cost model to account for various test costs involved in 3D integration and present a heuristic solution to minimize the overall manufacturing cost. In contrast to prior work, which is based on explicit enumeration of test flows, we adopt a formal optimization approach, which allows us to select an effective test flow by systematically exploring an exponentially large number of candidate test flows. Experimental results highlight the effectiveness of the proposed heuristic approach, which is compared to an exact approach for small test cases (three dies) and to a random-selection baseline methods for large test cases (up to 10 dies).**

I. Introduction

Three dimensional (3D) stacking involves the integration of multiple silicon dies in a vertical stack using short through-silicon vias (TSVs) [1]. Compared to traditional core-integration technologies, 3D stacking offers several benefits, such as reduced wire length, reduction in interconnect delays and power consumption, and higher interconnect bandwidth with improved performance. 3D-stacked memory chips are already in production [2]–[4] and the semiconductor industry is headed towards further exploitation of the benefits provided by 3D integration in a variety of product lines, such as 3D NOC [5], [6], 3D memory-on-processor [7], and 3D FPGA [8]. The emergence of 3D logic-logic stacks has also been predicted for the near future [9]. Motivated by advances in design and technology, researchers have started investigating test and design-for-testability techniques for 3D ICs [10]–[16].

Test cost has emerged as a potential showstopper for further advances in the adoption of 3D integration. The choice of test flow, i.e., what tests are used and when they are applied during 3D integration ("what to test", "when to test") affects test cost. 3D stacking involves many possible test insertions. Due to multiple yield and test cost parameters corresponding to different dies and tests, such as for pre-bond, post-bond, and partial stack, an exponentially large number of test flows must be evaluated. Therefore, analysis methods and tools are needed for test-cost optimization and automated test-flow selection.

*This research was supported in part by the National Science Foundation under grant no. CCF-1017391, and a gift from Cisco System through the Silicon Valley Community Foundation.

Several papers have been published recently on various aspects of test-cost modeling and optimization for 3D ICs [17]–[20]. These papers include attempts to hand-pick a test flow or select a test flow based on explicit enumeration of a few candidate test flows. However, prior work does not provide any means for systematically exploring (e.g., through implicit enumeration) the solution space of all possible test flows and reporting the test flow that minimizes test cost.

In this paper, we address the test-cost optimization problem for 3D ICs by developing a cost model that takes into account various test costs at each step of the stacking process. The model is generic and flexible in that it provides placeholders for different test costs that are typically incurred during 3D integration. The proposed model can be adapted for wafer-to-wafer (W2W), die-to-wafer (D2W), and die-to-die (D2D) stacking. We describe a heuristic procedure that is guided by a matrix-partitioning problem. Results are presented for a range of yield values and test costs to highlight the impact of parameters on test cost and test-flow selection. We also compare our results to an optimal (exhaustive-search) solution for a small problem instance of three dies and a baseline method for larger problem instances of up to 10 dies.

The rest of the paper is organized as follows. Section II details the motivation for this work and lists paper contributions. The notation used in the paper and the problem formulation are presented in Section III. Section IV models the optimization problem in terms of matrix partitioning, which forms the basis for the heuristic described in Section V. Results are shown in Section VI and conclusions are drawn in Section VII.

II. Motivation and Paper Contributions

For today's 2D ICs, tests can be applied at two stages: (i) at the wafer level (wafer sort); (ii) after the chip is packaged. Depending on the yield, wafer sort can be a significant cost saver by alleviating the need for packaging defective dies.

In the manufacturing of a 3D-stacked, tests can be applied at multiple stages—individual wafers or dies can be tested prior to bonding (pre-bond test) and testing can be carried out again after partial stacks are created (mid-bond test). A post-bond stack test can be applied to the complete stack with all the dies. During testing of a partial stack, dies at different layers in the stack can be tested (or re-tested), or their testing can be omitted.

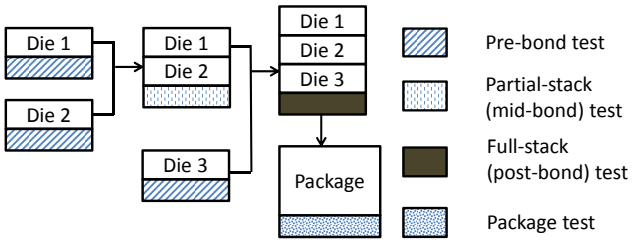

Fig. 1. 3D integration of a stack and the associated test insertions.

Fig. 1 sketches the typical integration of three dies to form a stack. This process involves incremental stacking of one die at a time. During testing of a stack, tests targeting faults in either of the dies present in the stack can be applied. Hence, in addition to pre-bond testing, a die can be tested at multiple stages of stacking as well. Moreover, there can be multiple types of tests, having different fault coverage and test application costs. If a test with lower cost is selected, that can save expenses upfront, but the lower fault coverage can result in the stacking of defective dies, thereby lowering the overall stack yield, and consequently increasing the overall cost of stack creation. Therefore, test-flow selection involves selection of the test insertions (also referred to as test moments in the literature [18]), and tests that provide the best trade-off between cost and quality. For this simple example, there exist $2^{3+3+2} = 256$ different combination of test insertions: tests for Die 1 and Die 2 can be applied at all the three stages and there are two stages in which tests for Die 3 can be applied. Moreover, for each selection of insertions, there are different ways in which tests can be selected out of a given test set. It can be easily shown that the test insertions can be selected in $O(2^{l^2})$ possible ways, where l is the total number of dies in a stack (refer to appendix). The optimization problem rapidly becomes intractable with the addition of dies to the stack, and because of added complexity associated with selection of tests. If we consider inter-die interconnect tests, even more test flows are possible. Our goal in this work is to minimize the total cost that includes test cost and the part of manufacturing cost that is affected by yield and test escapes. This work does not focus on how to calculate manufacturing cost or test cost at each stage; instead, it minimizes the total test-flow-driven cost given individual cost components.

The major contributions of this paper include the following:

- A generic cost model to incorporate different kinds of test costs involved in 3D integration;
- A reduction of the cost optimization problem to a better-understood matrix partitioning problem;
- A fast heuristic based on matrix partitioning for searching the solution space consisting of various test flows.

III. Problem Formulation

In this section, we formally define the test-cost optimization problem. In Table I, we highlight the notation for decision variables and the various parameters that constitute the optimization problem. The table entries include key parameters such as the pre-bond and stack tests that are available, test costs, yield, packaging cost, decision variables, etc. The definitions of these parameters are not repeated here for the sake of brevity. The reader is referred to Table I for details.

TABLE I
Table describing the notation used in the cost model.

Symbol	Meaning
	Knowns (The "Givens")
l	Total number of dies in the stack
$\mathcal{D}_i$	i^{th} die in the stack
$\mathcal{S}_k$	A stack of dies $\mathcal{D}_1$, $\mathcal{D}_2$, ... $\mathcal{D}_k$, $2 \leq k \leq l$. The stack $\mathcal{S}_k$ is partial for $k < l$.
$\mathcal{I}_i$	Interconnects between dies $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$, $1 \leq i < l$
B_i	Number of pre-bond tests of $\mathcal{D}_i$, $\forall i$.
PB_{ij}	j^{th} pre-bond test for $\mathcal{D}_i$, where $1 \leq j \leq B_i$.
b_{ij}	Cost associated with pre-bond test PB_{ij}.
$fb_i(j)$	A function that returns fault coverage of the pre-bond test PB_{ij}. If $j = 0$ (when no pre-bond test is chosen for $\mathcal{D}_i$), $fb_i(0) = 0$.
N_i	Number of different tests that can be applied to $\mathcal{D}_i$ during stack testing.
$\mathcal{T}_{ij}$	j^{th} stack test for $\mathcal{D}_i$, where $1 \leq j \leq N_i$.
t_{ij}	Cost of applying stack test $\mathcal{T}_{ij}$ (this remains the same regardless of the moment at which $\mathcal{T}_{ij}$ is applied).
$f_i(j)$	A function that returns fault coverage of the stack test $\mathcal{T}_{ij}$. If $j = 0$ (when no stack test is chosen for $\mathcal{D}_i$), $fb_i(0) = 0$.
a_{ijk}	A given binary 3-D matrix that indicates whether $\mathcal{T}_{ij}$ can be applied during stack testing of $\mathcal{S}_k$. The application of some stack test for a given die may require the presence of some other die(s) in the stack; therefore, not all tests are applicable at every stage.
ic_i	Cost of testing interconnect $\mathcal{I}_i$.
λ_i	Yield for die $\mathcal{D}_i$.
ω_{ik}	The component of bond yield of stack $\mathcal{S}_k$ related to defects induced in die $\mathcal{D}_i$ during the stacking of $\mathcal{S}_k$. The overall bond yield for stack $\mathcal{S}_k$ is given by $\prod_{i=1}^{k} \omega_{ik}$.
ρ_i	Interconnect yield of $\mathcal{I}_i$.
n_i	Number of instances of $\mathcal{D}_i$ manufactured.
DC_i	Manufacturing cost of each instance of $\mathcal{D}_i$
SC_k	Manufacturing cost of each instance of $\mathcal{S}_k$
TP_k	Fixed test cost for testing $\mathcal{S}_k$ (assuming some constant investment in setting up the infrastructure for testing $\mathcal{S}_k$, including logistic cost, regardless of the type or quality of the tests to be applied).
PC	Cost of packaging and testing a full stack
	Unknowns (To be determined)
n'_i	Number of instances of $\mathcal{D}_i$ that are available for creating $\mathcal{S}_i$.
m_k	Number of instances of $\mathcal{S}_k$ manufactured
m'_k	Number of instances of $\mathcal{S}_k$ that have been detected to be fault-free after stack testing.
x_{ij}	A binary variable that is assigned a value of 1 if pre-bond test PB_{ij} of $\mathcal{D}_i$ is carried out, otherwise it holds a zero.
y_{ijk}	A binary variable that is assigned a value of 1 if $\mathcal{D}_i$ is tested by applying the test $\mathcal{T}_{ij}$ during stack testing of $\mathcal{S}_k$. The variable y_{ijk} can be assigned 1 only if $a_{i,j,k} = 1$.
z_{ik}	A binary variable indicating whether interconnect layer $\mathcal{I}_i$ is tested during the testing of stack $\mathcal{S}_k$. The test time for the interconnects is small compared to test time for the dies [14], so we merge the testing of $\mathcal{I}_i$ with the stack test that first uses $\mathcal{I}_i$ for transporting test data; hence z_{ik} depends on the variables y_{ijk}.

We first quantify the total cost of manufacturing and pre-bond testing of the dies. The cost of manufacturing n_i instances of die $\mathcal{D}_i$ is $n_i \cdot DC_i$. The cost of testing n_i instances of $\mathcal{D}_i$ at a pre-bond stage is $n_i \cdot \sum_{j=1}^{B_i} x_{ij} \cdot b_{ij}$. Note that the x_{ij} decision variables model the fact that pre-bond tests are carried out only if they are part of the selected test flow.

The total cost of manufacturing and running pre-bond tests for all dies (C_1) is then given by the following equation.

$$C_1 = \sum_{i=1}^{l} n_i (DC_i + \sum_{j=1}^{B_i} x_{ij} \cdot b_{ij}) \quad (1)$$

Clearly, each die needs to be tested by at most one pre-bond test. In other words, $\sum_{j=1}^{B_i} x_{ij} \leq 1, \forall i$.

The total cost of manufacturing a stack of l dies ($C2$) is given by the equation: $C_2 = \sum_{k=2}^{l} m_k \cdot SC_k$, where the number of stacks of k dies (m_k) is determined by the number of dies and partial stacks that are available after defect screening.

The cost of testing the interconnects between the dies $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$ during the testing of S_k is $m_k \cdot z_{ik} \cdot ic_i$. The total cost of testing all the interconnect layers (C_3) is given by:

$$C_3 = \sum_{i=1}^{l-1} ic_i \cdot (\sum_{k=i+1}^{l} m_k \cdot z_{ik}) \quad (2)$$

As explained in Table I, an interconnect layer has to be tested not more than once. Hence, the constraint $\sum_{k=i+1}^{l} z_{ik} = 1$ is imposed on the variables z_{ik} for $i < l$.

If no stack test is applied on $\mathcal{S}_k$, the corresponding fixed test cost is zero. Therefore, the cumulative fixed test cost (C_4) can be written as follows:

$$C_4 = \sum_{k=2}^{l} I(\sum_{i=1}^{k} \sum_{j=1}^{N_i} a_{ijk} y_{ijk}).TP_k, \quad (3)$$

where $\sum_{i=1}^{k} \sum_{j=1}^{N_i} a_{ijk} y_{ijk}$ is the total number of stack tests that are run on $\mathcal{S}_k$ after it is created, and $I : \mathcal{Z} \rightarrow \{0, 1\}$ is a function that maps the set of non-negative integers to binary values. It is defined as follows: $I(x) = 0$ if $x = 0$ and $I(x) = 1$ if $x > 1$.

The total cost of stack testing can now be stated as:

$$C_5 = C_4 + \sum_{k=2}^{l} m_k (\sum_{i=1}^{k} \sum_{j=1}^{N_i} a_{ijk} y_{ijk}.t_{ij}). \quad (4)$$

Since we do not apply more than one stack test for a die at any given test stage, the following constraint on the variables y_{ijk} is applied: $\sum_{j=1}^{N_i} a_{ijk} y_{ijk} \leq 1$.

Finally, we account for the cost associated with packaging and final test, which is given by the equation $C_6 = m_l^{'} \cdot PC$.

The total cost for manufacturing and testing the 3D IC is $C_1 + C_2 + C_3 + C_5 + C_6$ and our objective is to minimize it by assigning appropriate values to x_{ij} and y_{ijk}. The variables z_{ik} depend on y_{ijk}, as explained in Table I.

In addition, if we assume that defects are induced only on top two dies during the stacking process, and only these dies are considered for testing at this stage, as in [18], we use constraint $\omega_{ik} = 1$ for $i \leq k - 2$.

There is one key constraint inherent in this optimization problem. Each type of test can be applied at most once, and the optimization method explores the solution space to find the best possible test stage at which a test should be applied (or not applied at all). A test can be ignored if the benefits offered by it are not commensurate with the added cost. This constraint can be captured using the relation: $\sum_{k=1}^{l} a_{ijk} y_{ijk} \leq 1, \forall i, j$.

Next we show how the variables n_i', m_i and m_i' are computed. The Williams and Brown model [21] postulates the following relationship between defect-level (DL), fault-coverage (fc), and yield (PY): $DL = 1 - (PY)^{1-fc}$. If n is the total number of chips manufactured using a process having production yield (PY), the number of non-faulty chips are $n \cdot PY$. If a test having fault coverage (fc) is applied, the number of chips that are detected to be non-faulty is given by the expression:

$$\frac{n \cdot PY}{1 - (1 - PY^{1-fc})} = n \cdot PY^{fc}.$$

Without loss of generality, we use the above relationship for calculating n_i'. Other yield and defect escape models can be easily incorporated without requiring any changes to our model. If for die $\mathcal{D}_i$, α_i is the index of the pre-bond test selected from the set of B_i pre-bond tests, n_i' is obtained as follows: $n_i' = n_i \cdot \lambda_i^{fb_i(\alpha_i)}$, where $fb_i(\alpha_i)$ is the fault coverage of the selected test. If none of the tests are selected, then we use the equations $\alpha_i = 0$ and $fb_i(0) = 0$ to get $n_i' = n_i$.

The number of instances of $\mathcal{S}_k$, referred to as m_k, formed after $\mathcal{S}_{k-1}$ and $\mathcal{D}_k$ are stacked, is given by the expression: $m_k = \min\{m_{k-1}^{'}, n_k^{'}\}$ for $k > 2$. Since m_1' is undefined, we use the equation $m_2 = \min\{n_1^{'}, n_2^{'}\}$ for $k = 2$.

The number of copies of $\mathcal{S}_k$ that are determined to be fault-free ($m_k^{'}$) depends on what tests are applied during testing of $\mathcal{S}_k$ and during all the previous test insertions. For ease of understanding, let us define β_{ik} as the index of the stack test selected from the set of the N_i stack tests for testing the die $\mathcal{D}_i$ during the testing of stack $\mathcal{S}_k$. The corresponding fault coverage is given by $f_i(\beta_{ik})$. If no stack test for $\mathcal{D}_i$ is selected during this stage, then $\beta_{ik} = 0$ and $f_i(\beta_{ik}) = 0$. Assumptions made for simplifying the calculation of m_k' are listed below:

Assumption 1 The fault coverage of each interconnect test is 100%. The cost-optimization model can be extended for imperfect interconnect tests—a subject of future research.

Assumption 2 We assume that failure of dies in a stack during mid-bond testing and post-bond testing are uncorrelated. Since dies in a heterogeneous stack are likely to be dissimilar, the above independence assumption can be justified in practice. This assumption has two implications as discussed below:

1) If $\mathcal{D}_i$ is tested during stack testing of $\mathcal{S}_k$ using the test $\mathcal{T}_{ip}$ ($\beta_{ik} = p$ in this case), and no other tests for other dies are applied during this test stage, then the number of stacks obtained after testing is given by the expression

$$m_k' = \begin{cases} m_k \cdot (\omega_{kk} \cdot \lambda_k^{1-I(\alpha_k)})^{f_k(p)}, & \text{if } i = k \\ m_k \cdot (\omega_{ik} \cdot Y_{i,k-1})^{f_i(p)}, & \text{if } i < k, \end{cases}$$

where $Y_{i,k-1}$ incorporates bond-yield components $\omega_{i,k'}$ (for $i \leq k' \leq k - 1$) and λ_i. The factor $Y_{i,k-1}$ is recursively defined as below:

$$Y_{i,k-1} = (\omega_{i,k-1} \cdot Y_{i,k-2})^{1-I(\beta_{i,k-1})},$$

$$Y_{i,k-2} = (\omega_{i,k-2} \cdot Y_{i,k-3})^{1-I(\beta_{i,k-2})},$$

$$\cdots$$

$$Y_{i,i} = \begin{cases} (\omega_{i,i} \cdot \lambda_i^{1-I(\alpha_i)})^{1-I(\beta_{i,i})}, & \text{if } i > 1 \\ (\lambda_i^{1-I(\alpha_i)}), & \text{if } i = 1. \end{cases}$$

This means that $\omega_{i,k-1}$ is included in the expression for m'_k only if no stack test for $\mathcal{D}_i$ was applied during the testing of $\mathcal{S}_{k-1}$. Also, if $\mathcal{D}_i$ remains untested after the stacks $\mathcal{S}_{k-1}$ and $\mathcal{S}_{k-2}$ are created, $\omega_{i,k-2}$ is included in the expression for m'_k. Similarly, it can be concluded that λ_i is included in the expression for m'_k only if $\mathcal{D}_i$ is never tested after it is manufactured, neither using a pre-bond test nor any of the stack tests. For $i = 1$, the expression for the base case ($Y_{i,i}$) is different from the general case because no stack is created using a single die $\mathcal{D}_1$ ($\omega_{1,1}$ and $\beta_{1,1}$ are undefined).

Since the stack test $\mathcal{T}_{ip}$ for $\mathcal{D}_i$ does not capture defects in other dies, $m_k^{'}$ does not depend on parameters such as $\omega_{jk'}$ or λ_j for $j \neq i$ and $j \leq k' \leq k$. The interconnect yields do not feature in the expression for the same reason.

2) Next, along with the test $\mathcal{T}_{ip}$, we apply the stack test $\mathcal{T}_{jq}$ ($\beta_{jk} = q$) on $\mathcal{D}_j$ ($i \neq j$) during the testing of $\mathcal{S}_k$. The number of stacks obtained after applying these tests is

$$m_k^{'} = m_k \cdot (\omega_{ik} \cdot Y_{i,k-1})^{f_i(p)} \cdot (\omega_{jk} \cdot Y_{j,k-1})^{f_j(q)}.$$

Due to the assumption of the independence of die failures to tests, yield components from different dies are multiplied together. Also the independence of fail events does not enforce a particular order in which the tests are to be applied at a given test insertion.

Using the above two assumptions, a more general equation for $m_k^{'}$ can be written in a compact manner as follows:

$$m_k^{'} = m_k \cdot (\omega_{kk} \cdot \lambda_k^{1-I(\alpha_k)})^{f_k(\beta_{k,k})} \cdot \prod_{i=1}^{k-1} (\omega_{ik} \cdot Y_{i,k-1})^{f_i(\beta_{i,k})} \cdot \prod_{i=1}^{k-1} \rho_i^{z_{ik}}$$

The above equation also accounts for the interconnect yield ρ_i of the interconnect layer $\mathcal{I}_i$.

As stated in Section I, the cost model described above can be adopted for any of the integration approaches, viz., W2W, D2D, and D2W. In the case of W2W integration, the pre-bond tests may be skipped due to difficulty associated with this step. Hence the x_{ij} variables can be constrained to zero in our model. Some mid-bond tests can be skipped (and the decision variables fixed appropriately) due to constraints on test access. For D2W stacking, the cost model can consider pre-bond tests as decision variables, but certain mid-bond tests may be omitted and corresponding decision variables fixed. Note that D2D stacking offers the most flexibility in terms of test flows, hence it leads to the largest number of decision variables among the stacking scenarios. The model is illustrated with the help of a simple example in the appendix.

IV. Yield Matrix and Partitioning

In this section, we construct a yield matrix that provides insights into the optimization method. All yield parameters are written in a matrix form as shown below

$$YM = \begin{pmatrix} \omega_{l,l} & \underline{\lambda_l} & 1 & 1 & 1 & 1 \\ \omega_{l-1,l} & \omega_{l-1,l-1} & \underline{\lambda_{l-1}} & 1 & 1 & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \omega_{1,l} & \omega_{1,l-1} & \cdot & \cdot & \omega_{1,2} & \underline{\lambda_1} \end{pmatrix}$$

Each row of this lxl matrix corresponds to a die. The first row (counting from bottom) corresponds to die $\mathcal{D}_1$, row 2 to $\mathcal{D}_2$, and so on. Each column from column 2 to column l (counting from right) of the matrix corresponds to a test insertion In addition, the underlined cells in the matrix corresponds to the pre-bond test insertions. Hence, there are a total of $l-1+l = 2l-1$ possible test insertions. We study how the test-flow selection problem is modeled as a yield matrix-partitioning problem and we use this model later to develop a heuristic solution. If $\mathcal{D}_1$ is tested at insertions corresponding to columns k_1 and k_2, and a pre-bond test is also applied, then the first row is partitioned as below:

$$\omega_{1,l} \cdot \cdot \, \omega_{1,k_2+1} \mid \omega_{1,k_2} \cdot \cdot \, \omega_{1,k_1+1} \mid \omega_{1,k_1} \cdot \cdot \, \omega_{1,2} \mid \lambda_1,$$

where $l \geq k_2 \geq k_1 \geq 2$.
The symbol '|' is a *divider* that is used for showing the partitions created due to selection of a test flow. Suppose test $\mathcal{T}_{1p}$ is applied at the test insertion corresponding to column k_1, and $\mathcal{T}_{1q}$ corresponding to column k_2, and PB_{1r} at the pre-bond test insertion of $\mathcal{D}_1$. Using the equation derived for n'_i in the previous section, we obtain $n_1^{'} = n_1 \cdot \lambda_1^{fb_1(r)}$. It can be verified that the variable m'_{k_1} will have a factor of $(\omega_{1,k_1} \cdot \omega_{1,k_1-1} \ \cdot \cdot \ \omega_{1,2})^{f_1(p)}$, and m_{k_2} will have a factor of $(\omega_{1,k_2} \cdot \omega_{1,k_2-1} \ \cdot \cdot \ \omega_{1,k_1+1})^{f_1(q)}$. Such partitions are created in every row depending on what test insertions are selected for the corresponding die. The test insertions at which the tests are applied creates the partition. A different selection of test insertions results in a different partitioning of the matrix YM. With the above partitioning scheme, it becomes very easy to visualize how different yield parameters affect several m'_k. With each test-flow selected, we have a unique combination of partitions of all rows of the yield matrix, and an assignment of tests to each partition. An optimal solution consists of an optimal partitioning of each row of the matrix and an optimal selection of tests that are assigned to test insertions. In this way, we are able to determine an effective test flow in a constructive manner that reduces overall cost, unlike prior work such as [18] that simply enumerates a small number of test flows and evaluates their cost.

V. Heuristic Procedure

In this section, we describe the heuristic procedure that we have implemented to solve the optimization problem.

On running an exhaustive-search approach to small and moderate-sized problem instances, the optimal solution obtained was found to consist of sparse assignments of binary

variables, i.e., a small number of these binary variables took a value of 1 in the optimal solution. Therefore, we start by assuming that no tests are applied for any die and try to reduce the cost by running the optimization flow on one die at a time. Hence a large optimization problem is reduced to l smaller optimization problems that are solved sequentially. Equivalently, the first row of the yield matrix is partitioned first and a solution is obtained. The heuristic then partitions all other rows based on the assignments of the variables made in previous rows.

We create the partition of a row of the yield matrix incrementally; for row i, a partition of cells from column 1 to k is created first, where k is varied from 1 to l. Partitioning is carried out using dividers. An optimal partition is an optimal placement of such dividers. Note that the proposed heuristic solution does not guarantee optimality for what we conjecture to be an NP-hard problem. If no divider is used for a row, the corresponding die is never tested during the stacking process. Let us denote the problem of partitioning cells of row i from columns 1 to k by $\mathcal{P}(i,k)$. The solutions corresponding to the set of problems $\mathcal{P}(i,k')$, where $1 \leq k' < k$, are used for solving the problem $\mathcal{P}(i,k)$.

The heuristic is outlined in Fig. 7 in the appendix. For each pair of i and k, a call to procedure $find\_partition$ is made for solving the problem $\mathcal{P}(i,k)$. The partition for the row i is given by the solution for $\mathcal{P}(i,l)$. First, the cost is computed without using a divider, which is later improved upon using the $find\_partition$ procedure (see Fig. 8 in the appendix). A partition can be created using multiple dividers. In order to solve $\mathcal{P}(i,k)$, solutions are stored for each possible number of dividers, and the best among them is chosen as the solution for $P(i,k)$. We use a three-dimensional array M to store the candidate solutions, where $M[i,k,d]$ stores the solution for $\mathcal{P}(i,k)$ using d dividers. The d^{th} divider is swept from column 1 to column k and best selection of tests is obtained based on the choices already made for the smaller subproblems; since the d^{th} divider is placed after the $(d-1)^{th}$ divider, $M[i,k,d]$ is constructed using solution of $M[i,k',d-1]$, where $1 \leq k' < k$. There are two fields of each element of the array M. The field div_pos stores the position of the corresponding divider, and the the field $selected\_test$ stores the test selected (using the $select\_test$ procedure—Fig. 9 in the appendix) for the partition created on the right of the divider. The global variable $mincost$ maintains the best cost found so far, and it is updated whenever an improvement is found.

The procedure $find\_partition$ is called l^2 times, which in turn, calls the procedure $select\_test$ O(l) times. The for loop in the procedure $select\_test$ gets executed a maximum of l times, each of which takes O(l) time to compute cost. On finding a cost better than the $mincost$, updating the set of x and y variables takes O(l) time, but it is dominated by the execution time of $select\_test$. Therefore, the overall computational complexity of the proposed heuristic method is O(l^5).

TABLE II
MODEL PARAMETERS FOR A THREE-DIE STACK.

Parameters	Values	Remarks
(n_1, n_2, n_3)	$(600, 550, 500)$	Number of dies.
(DC_1, DC_2, DC_3)	$(4, 6, 5)$	Die manufacturing cost.
(B_1, B_2, B_3)	$(2, 2, 2)$	Two pre-bond tests for each die.
$(b_{11}, fb_1(1))$ $(b_{12}, fb_1(2))$ $(b_{21}, fb_2(1))$ $(b_{21}, fb_2(2))$ $(b_{31}, fb_3(1))$ $(b_{32}, fb_3(2))$	$(1.8, 0.80)$ $(2.3, 0.93)$ $(2.5, 0.90)$ $(3.5, 0.95)$ $(2, 0.82)$ $(4, 0.87)$	Test cost and corresponding fault coverage for each pre-bond test.
(SC_2, SC_3)	$(10, 15)$	Stacking cost
(N_1, N_2, N_3)	$(2, 2, 1)$	Two tests each for $\mathcal{D}_1$ and $\mathcal{D}_2$, and one for $\mathcal{D}_3$.
$(a_{111}, a_{112}, a_{113})$ $(a_{121}, a_{122}, a_{123})$ $(a_{211}, a_{212}, a_{213})$ $(a_{221}, a_{222}, a_{223})$ $(a_{311}, a_{312}, a_{313})$	$(0, 1, 1)$ $(0, 1, 1)$ $(0, 1, 1)$ $(0, 1, 1)$ $(0, 0, 1)$	The tests $\mathcal{T}_{11}$, $\mathcal{T}_{12}$, $\mathcal{T}_{21}$, $\mathcal{T}_{22}$ can be applied to both $\mathcal{S}_2$ and $\mathcal{S}_3$, but the test $\mathcal{T}_{31}$ can only be applied to full stack $\mathcal{S}_3$.
$(t_{11}, f_1(1))$ $(t_{12}, f_1(2))$ $(t_{21}, f_2(1))$ $(t_{22}, f_2(2))$ $(t_{31}, f_3(1))$	$(2.4, 0.93)$ $(1.8, 0.82)$ $(3.4, 0.89)$ $(4, 0.94)$ $(3.5, 0.82)$	Test cost and corresponding fault coverage for each stack test.
$(\lambda_1, \lambda_2, \lambda_3)$	$(0.50, 0.83, 0.93)$	
$(\omega_{12}, \omega_{13})$ $(\omega_{22}, \omega_{23})$ ω_{33}	$(0.89, 0.93)$ $(0.81, 0.85)$ 0.91	
(TP_2, TP_3)	$(500, 1300)$	Fixed test cost.
PC	50	Package cost.

VI. RESULTS

First, we compare the results from our heuristic with that from exhaustive enumeration for a stack consisting of three dies. Note that exhaustive enumeration leads to optimal results (minimum cost), hence we can compare the heuristic solution in terms of optimality for this small test case. The problem instance is small and the exhaustive approach terminates quickly. Next, we compare our results with that obtained using a random test-flow selection approach for larger designs, and intermediate results derived from the enumeration approach after letting it run for several hours.

There are five possible test insertions for a stack of three dies: (i) Pre-bond testing of $\mathcal{D}_1$; (ii) Pre-bond testing of $\mathcal{D}_2$; (iii) Pre-bond testing of $\mathcal{D}_3$; (iv) Stack testing of $\mathcal{S}_2$; (v) Stack testing of $\mathcal{S}_3$. A package is thoroughly tested before shipping, hence it is not a decision variable in our model. Other parameters of the model are listed below in Table II. The unit of cost is chosen arbitrarily, hence we refer to cost in terms of arbitrary units (A.U.), as done in [19].

An exhaustive search evaluates 32768 different possible selection of test flows possible for this example, where pre-bond and stack tests can be chosen in different combinations. Exhaustive enumeration approach reports an optimum value of 27864 for the objective function. Fig. 2 captures the test cost, normalized to maximum test cost, for a large number of test flows. It was observed that the minimum cost obtained is 0.54 times the reported maximum cost for these test flows. These results highlight the importance of selecting cost-effective test flows. If selective enumeration is used to pick only a few test

TABLE III
EXPERIMENTS ON STACK OF THREE DIES BY VARYING SOME PARAMETERS.

varied	Value range	Overall Cost (A.U.) Exhaustive	Overall Cost (A.U.) Heuristic	Remarks
TP_2	low	27364	27364	For "high" values of TP_2, no tests are applied to $\mathcal{S}_2$.
	medium	27864	27864	
	high	28547	28547	
SC_3	low	23824	23824	When the stacking cost for $\mathcal{S}_3$ is "high", tests are applied during pre-bond and $\mathcal{S}_2$ test insertions to weed out bad dies and partial stacks.
	medium	27864	27864	
	high	33727	33727	
t_{21}, t_{22}	low	25922	25922	When the stack test cost for $\mathcal{D}_2$ are "high", only pre-bond test is applied to $\mathcal{D}_2$.
	medium	27864	27864	
	high	30011	30011	

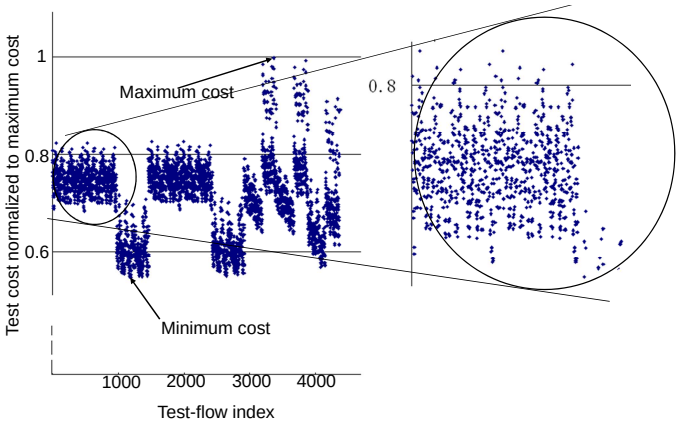

Fig. 2. Showing overall cost (normalized to maximum cost) of various flows.

flows as in [18], many cost-effective test flows are likely to be missed for larger stacks and more test insertions. It is simply not practical to explicitly enumerate all possible test flows. The values of the decision variables (sets x and y) corresponding to an optimal solution were found to be the following:

$$\begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} y_{111} & y_{112} & y_{113} \\ y_{121} & y_{122} & y_{123} \\ y_{211} & y_{212} & y_{213} \\ y_{221} & y_{222} & y_{223} \\ y_{311} & y_{322} & y_{333} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

From the above values, we infer that in the selected cost-optimal test flow, the pre-bond test for $\mathcal{D}_1$ is applied, but no pre-bond tests are applied to the other dies. Since the yield of $\mathcal{D}_1$ is low, the optimizer made the decision to run pre-bond tests for it. Moreover, the stack tests $\mathcal{T}_{12}$ and $\mathcal{T}_{22}$ are selected to be applied to partial stack $\mathcal{S}_2$. All the remaining tests are discarded. Our heuristic provided the same result.

Next, we sweep λ_1 from 0.5 to to 0.95 to see its effect on the decision about including pre-bond test for die $\mathcal{D}_1$ in the chosen test flow. Other parameters, including λ_2 and λ_3, were not changed. For lower values of yield, pre-bond test of $\mathcal{D}_1$ is selected, but optimal solutions do not include this pre-bond test after the yield exceeds a threshold (Fig. 3(a)). The above

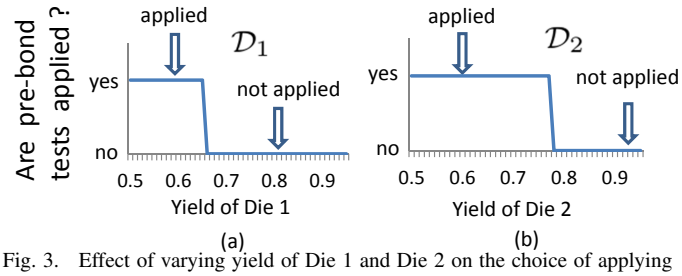

Fig. 3. Effect of varying yield of Die 1 and Die 2 on the choice of applying pre-bond tests to the corresponding dies.

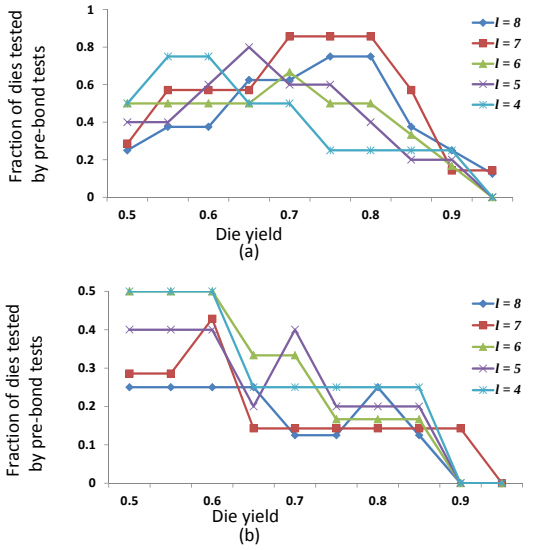

Fig. 4. Effect of varying die yield on pre-bond test selection under (a) low pre-bond test cost, and (b) high pre-bond test costs.

threshold is specific to the design and parameter values, such as yield and test costs. We repeated the above experiment with $\mathcal{D}_2$ by fixing λ_1 to a value of 0.73 and sweeping λ_2 from 0.5 to 0.95. A similar "phase transition" behavior was observed; see Fig. 3(b).

We next vary some other parameters to show their impact on test-flow selection. Table III lists the results of experiments that were carried out by varying some other parameters of the model. Observations are recorded in the "Remarks" column of the table. Note that the heuristic solution matched the results obtained by the exact approach based on exhaustive enumeration.

We also evaluated the impact of package cost (inclusive of package test cost) on the test-flow selection. Since package cost is incurred only after the stack is created, cost due to packaging will tend to dominate the overall cost if a large number of stacks is created, or the package cost per stack is high. In order to minimize this component of the overall cost when package cost per stack is high, bad dies and stacks should be screened with more certainty such that packaging of faulty stacks can be avoided. On sweeping package cost from low to high values, we find that more tests are applied upfront for higher package costs, hence the test component of overall cost increases.

/Fig. 4(a) depicts the effect of varying die yields on the fraction of dies chosen for pre-bond test application. The die

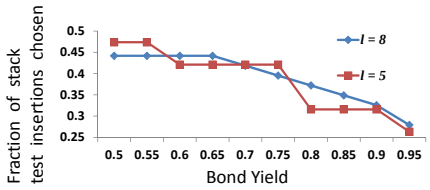

Fig. 5. Effect of varying bond yield on stack test selection.

yields for all dies are swept from a low value of 0.5 to a value as high as 0.95. The yield parameters denoted by ω_{ij} are assigned random values between 80% to 95%. For low values of die yield, the number of dies tested by their respective pre-bond test is usually higher. Note that a cost-effective solution to a test-flow selection problem depends on the problem instance; therefore, a generic rule cannot be established that all dies with a low die yield must always be tested before stacking. This is also highlighted by the profile of the curve in Fig. 4(a). We observed that those dies that are not tested before stacking at low values of die yield, are tested later after stacking. Since the expression for m_k is given by $\min\{m'_{k-1}, n'_k\}$, we can achieve some cost saving by skipping the pre-bond test for $\mathcal{D}_k$, if $m'_{k-1} < n_k$, and if a cheaper stack test for $\mathcal{D}_k$ is present downstream the stacking process. We repeated the same experiment by doubling the cost of pre-bond tests. The results are shown in Fig. 4(b). The fraction of dies chosen is less in this case, further highlighting the importance of a cost-analysis tool.

In an another experiment, we vary the bond yields from 0.5 to 0.95 after fixing the die yields to high values (in a range of 0.9 to 0.95). We plot the fraction of stack test insertions selected for stack with eight dies and five dies in Fig. 5. As the bond yield increases, the number of selected stack test insertions decreases. Note that for a stack consisting of eight dies, the number of possible test insertions is $\frac{8^2+3\times8-2}{2} = 43$ (refer to appendix), and that for a five-die stack is 19.

We ran our heuristic procedure on larger stacks with up to 10 dies; see Table IV. The table reports the cost obtained using three methods. Exhaustive enumeration is clearly not feasible for these large designs, so we allowed it to run for 3 hours before terminating the search process; therefore, the results obtained in this manner are not optimal. The heuristic procedure completed within a second of CPU time to report a much better result. We also implemented a random test-flow selection approach, and report the maximum, minimum, and average value of the cost over 500 iterations. The results in Table IV clearly highlight the benefit of cost optimization compared to the enumeration of a few selected test flows, as in [18].

TABLE IV
EXPERIMENTAL RESULTS FOR LARGER STACKS.

No. of dies	Cost of the reported test flow (A.U.)				
	Exhaustive enumeration (up to 3 h of CPU time)	Random test-flow selection			Proposed method
		Min.	Avg.	Max.	
6	115215	265918	277297	341767	103615
7	145324	420645	432454	538187	122287
8	188465	541867	555969	671057	144363
9	204798	614195	626623	746857	150398
10	213033	652188	667377	775237	156658

VII. CONCLUSION

We have studied the test-flow selection problem for achieving cost minimization, and proposed a generic and flexible cost model to account for various test costs incurred during 3D integration. Solutions to the test-flow selection problem depends on the problem instance, and it is difficult to specify a generic set of rules for cost minimization. We have modeled the problem of selecting a cost-effective test flow in terms of matrix-partitioning problem, and solved using a fast heuristic method. Experimental results have highlighted the effectiveness of the proposed heuristic approach, which is compared to an exact approach for small test cases (three dies) and to a random-selection baseline methods for large test cases (up to 10 dies). The cost-optimization method provides interesting insights into relationships between yield, test cost, and the selection various pre-bond and stack tests.

REFERENCES

[1] K. Banerjee, S. Souri, P. Kapur, and K. Saraswat, "3-D ICs: A Novel Chip Design for Improving Deep-Submicrometer Interconnect Performance and Systems-on-Chip Integration," *Proceedings of the IEEE*, vol. 89, no. 5, pp. 602–633, May 2001.

[2] U. Kang *et al.*, "8 Gb 3-D DDR3 DRAM Using Through-Silicon-Via Technology," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 1, pp. 111–119, Jan. 2010.

[3] T. Mitsuhashi *et al.*, "Development of 3D-Packaging Process Technology for Stacked Memory Chips," in *MRS Proceedings*, 2006, pp. 155–162.

[4] M. Kawano *et al.*, "A 3D Packaging Technology for 4 Gbit Stacked DRAM with 3 Gbps Data Transfer," in *Proceedings of the Electron Devices Meeting (IEDM)*, Dec. 2006, pp. 1–4.

[5] B. Feero and P. Pande, "Networks-on-Chip in a Three-Dimensional Environment: A Performance Evaluation," *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 32–45, Jan. 2009.

[6] J. Kim *et al.*, "A Novel Dimensionally-Decomposed Router for On-Chip Communication in 3D Architectures," *SIGARCH Comput. Archit. News*, vol. 35, no. 2, pp. 138–149, June 2007.

[7] G. H. Loh, Y. Xie, and B. Black, "Processor Design in 3D Die-Stacking Technologies," *IEEE Micro*, vol. 27, no. 3, pp. 31–48, May-June 2007.

[8] C. Ababei, P. Maidee, and K. Bazargan, "Exploring Potential Benefits of 3D FPGA Integration." in *Field Programmable Logic and Application*, vol. 3203. Springer, 2004, pp. 874–880.

[9] T. Thorolfsson, G. Luo, J. Cong, and P. Franzon, "Logic-on-Logic 3D Integration and Placement," in *Proceeding of the 3D Systems Integration Conference (3DIC)*, Nov. 2010, pp. 1–4.

[10] B. Noia and K. Chakrabarty, "Pre-Bond Probing of TSVs in 3D Stacked ICs," in *Proceedings of the IEEE International Test Conference (ITC)*, Sept. 2011, pp. 1 –10.

[11] B. Noia, K. Chakrabarty, S. Goel, E. J. Marinissen, and J. Verbree, "Test-Architecture Optimization and Test Scheduling for TSV-Based 3-D Stacked ICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 11, pp. 1705 –1718, Nov. 2011.

[12] H. Lee and K. Chakrabarty, "Test Challenges for 3D Integrated Circuits," *IEEE Design & Test of Computers*, vol. 26, pp. 26–35, Sept.-Oct. 2009.

[13] S. Panth and S. Lim, "Scan Chain and Power Delivery Network Synthesis for Pre-Bond test of 3D ICs," in *Proceedings of the 29th IEEE VLSI Test Symposium (VTS)*, may 2011, pp. 26 –31.

[14] E. J. Marinissen and Y. Zorian, "Testing 3D Chips Containing Through-Silicon Vias," in *Proceedings of the IEEE International Test Conference (ITC)*, nov. 2009, pp. 1 –11.

[15] E. J. Marinissen, J. Verbree, and M. Konijnenburg, "A Structured and Scalable Test Access Architecture for TSV-Based 3D Stacked ICs," in *Proceedings of the 28th IEEE VLSI Test Symposium (VTS)*, April 2010, pp. 269 –274.
[16] L. Jiang *et al.*, "Modeling TSV Open Defects in 3D-Stacked DRAM," in *Proceedings of the IEEE International Test Conference (ITC)*, 2010, pp. 174–182.
[17] M. Taouil and S. Hamdioui, "Layer redundancy based yield improvement for 3d wafer-to-wafer stacked memories," in *Proceedings of 16th IEEE European Test Symposium*, Trondheim, Norway, May 2011, pp. 45–50.
[18] M. Taouil, S. Hamdioui, K. Beenakker, and E. J. Marinissen, "Test Cost Analysis for 3D Die-to-Wafer Stacking," in *Proceedings of the 19th IEEE Asian Test Symposium (ATS)*, 2010, pp. 435–441.
[19] Y. Chen, D. Niu, Y. Xie, and K. Chakrabarty, "Cost-Effective Integration of Three-dimensional (3D) ICs Emphasizing Testing Cost Analysis," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, 2010, pp. 471–476.
[20] Y.-W. Chou, P.-Y. Chen, M. Lee, and C.-W. Wu, "Cost Modeling and Analysis for Interposer-Based Three-Dimensional IC," in *IEEE 30th VLSI Test Symposium (VTS)*, April 2012, pp. 108–113.
[21] T. Williams and N. Brown, "Defect Level as a Function of Fault Coverage," *IEEE Transactions on Computers*, vol. C-30, no. 12, pp. 987–988, Dec. 1981.

APPENDIX

PROOF FOR THE COMPLEXITY OF THE SELECTION OF TEST INSERTIONS

Given a stack of l dies, we show that the total number of ways in which test insertions can be selected is $O(2^{l^2})$. Corresponding to each pre-bond stage, there is one test insertion each. Since k^{th} die can potentially be tested at each subsequent stacking stages, this accounts for an additional $l-k+1$ test insertions for the k^{th} die ($l-1$ insertions for $k=1$). Therefore, the total number of possible test insertions is given by the expression:

$$l \text{ (for each possible pre-bond test)} + \sum_{k=2}^{l}(l-k+1) + l - 1$$

$$= 2l - 1 + \frac{l(l-1)}{2} = \frac{l^2+3l-2}{2}.$$

A test insertion can be selected or discarded independently of the selection (or omission) of other insertions, hence the number of ways in which test insertions can be selected is $2^{\frac{l^2+3l-2}{2}} = O(2^{l^2})$.

AN ILLUSTRATION OF THE PROPOSED COST MODEL USING A SMALL EXAMPLE

We illustrate the proposed cost model using a simple example consisting of three dies: $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}_3$. In the integration process, a partial stack $\mathcal{S}_2$ and a full stack $\mathcal{S}_3$ are formed. Tables V—VI show all the known (given) and unknown (to be determined) model parameters, respectively. The parameters for this small example are listed in the same order as in Table I. All manufacturing and test costs, and the constraints on the variables are shown in Table VII. Table VIII lists a set of values that can be assigned to the decision variables that satisfy the constraints, and shows how other variables depend on these chosen values. Fig. 6 shows the selected flow for this example.

TABLE V
TABLE OF KNOWN PARAMETERS FOR A SMALL EXAMPLE WITH THREE DIES.

$l = 3$ $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{D}_3$ $\mathcal{S}_2$ and $\mathcal{S}_3$ $\mathcal{I}_1$ and $\mathcal{I}_2$	
$B_1 = B_2 = B_3 = 1$	One pre-bond test available per die
$PB_{11}, PB_{21}, PB_{31}$	Pre-bond tests available for the dies
b_{11}, b_{21}, b_{31}	Cost associated with the above pre-bond tests
$fb_1(1), fb_2(1), fb_3(1)$	Fault coverage of the pre-bond tests, $fb_i(0) = 0$, for all i
$N_1 = N_2 = N_3 = 1$	One stack test available per die
$\mathcal{T}_{11}, \mathcal{T}_{21}, \mathcal{T}_{31}$	Stacks tests available for the dies
t_{11}, t_{21}, t_{31}	Cost associated with the above stack tests
$f_1(1), f_2(1), f_3(1)$	Fault coverage of the stack tests, $f_i(0) = 0, \forall i$
$a_{112}, a_{113}, a_{212}, a_{213}, a_{313}$	Only valid entries in the matrix because any stack test for $\mathcal{D}_1$ and $\mathcal{D}_2$ can only be applied after $\mathcal{S}_2$ or $\mathcal{S}_3$ are created, and a stack test for $\mathcal{D}_3$ can only be applied after $\mathcal{S}_3$ is manufactured
ic_1, ic_2	cost of testing interconnects
$\lambda_1, \lambda_2, \lambda_3$	Die yields
$\omega_{12}, \omega_{13}, \omega_{22}, \omega_{23}, \omega_{33}$	Yield parameters of stack
ρ_1, ρ_2	Interconnect yields
n_1, n_2, n_3	Number of instances of each die
DC_1, DC_2, DC_3	Manufacturing cost of each instance of respective dies
SC_2, SC_3	Manufacturing cost of each instance of respective stacks
TP_2, TP_3	Fixed test costs
PC	Cost of package and testing

TABLE VI
TABLE OF UNKNOWN PARAMETERS FOR THE SMALL EXAMPLE.

x_{11}, x_{21}, x_{31} $y_{112}, y_{113}, y_{212}, y_{213}, y_{213}$ z_{12}, z_{13}, z_{23}
$n'_1 = n_1 \cdot \lambda_1^{fb_1(\alpha_1)}$ $n'_2 = n_2 \cdot \lambda_2^{fb_2(\alpha_2)}$ $n'_3 = n_3 \cdot \lambda_3^{fb_3(\alpha_3)}$ $\alpha_1, \alpha_2, \alpha_3 \in \{0, 1\}$ (only one pre-bond test is available per die)
$m_2 = \min\{n'_1, n'_2\}$ $m'_2 = m_2 \cdot (\omega_{2,2}.(\lambda_2)^{1-I(\alpha_2)})^{f_2(\beta_{2,2})} \cdot (\omega_{1,2}.(\lambda_1)^{1-I(\alpha_1)})^{f_1(\beta_{1,2})} \cdot (\rho_1)^{z_{12}}$ $m_3 = \min\{m'_2, n'_3\}$ $m'_3 = m_3 \cdot (\omega_{3,3}.\lambda_3^{1-I(\alpha_3)})^{f_3(\beta_{3,3})}$ $\cdot(\omega_{2,3} \cdot (\omega_{2,2} \cdot \lambda_2^{1-I(\alpha_2)})^{1-I(\beta_{2,2})})^{f_2(\beta_{2,3})}$ $\cdot(\omega_{1,3} \cdot (\omega_{1,2} \cdot \lambda_1^{1-I(\alpha_1)})^{1-I(\beta_{1,2})})^{f_2(\beta_{1,3})}$ $\rho_1^{z_{13}} \cdot \rho_2^{z_{23}}$ $\beta_{12}, \beta_{13}, \beta_{22}, \beta_{23}, \beta_{33} \in \{0, 1\}$ (only one stack test available per die)

HEURISTIC PROCEDURES

We provide heuristic procedures used in the proposed method. Fig. 7 outlines the top-level heuristic that calls the procedure $find\_partition$, shown in Fig. 8, for solving the problem $\mathcal{P}(i,k)$. This procedure calls the procedure $select\_test$, which is shown in Fig. 9.

TABLE VII
COSTS AND CONSTRAINTS FOR THE SMALL EXAMPLE.

$C_1 = (DC_1 + x_{11} \cdot b_{11}) + (DC_2 + x_{21} \cdot b_{21}) + (DC_3 + x_{31} \cdot b_{31})$ $C_2 = m_2 \cdot SC_2 + m_3 \cdot SC_3$ $C_3 = ic_1 \cdot (m_2 \cdot z_{12} + m_3 \cdot z_{13}) + ic_2 \cdot (m_3 \cdot z_{23})$ $C_4 = (a_{112} \cdot y_{112} + a_{212} \cdot y_{212}) \cdot TP_2 + (a_{113} \cdot y_{113} + a_{213} \cdot y_{213} + a_{313} \cdot y_{313}) \cdot TP_3$ $C_5 = C_4 + m_2 \cdot (a_{112} \cdot y_{112} \cdot t_{112} + a_{212} \cdot y_{212} \cdot t_{212}) + m_3 \cdot (a_{113} \cdot y_{113} \cdot t_{113} + a_{213} \cdot y_{213} \cdot t_{213} + a_{313} \cdot y_{313} \cdot t_{313})$ $C_6 = m'_t \cdot PC$
Constraints
$z_{12} + z_{13} \leq 1$ $a_{113} \cdot y_{112} + a_{213} \cdot y_{113} \leq 1$ $a_{212} \cdot y_{212} + a_{213} \cdot y_{213} \leq 1$

TABLE VIII
A SOLUTION FOR THE DECISION VARIABLES FOR THE SMALL EXAMPLE.

$x_{11} = 1, x_{21} = 0, x_{31} = 0$: pre-bond test of only $\mathcal{D}_1$ is applied. $y_{112} = 1, y_{113} = 0, y_{212} = 0, y_{213} = 1, y_{313} = 1$: stack test of $\mathcal{D}_1$ is applied during $\mathcal{S}_2$ testing, and stack tests of $\mathcal{D}_2$ and $\mathcal{D}_3$ are applied during testing of $\mathcal{S}_3$. $z_{12} = 1, z_{13} = 0, z_{23} = 1$: interconnect test of $\mathcal{I}_i$ applied with the last applied stack test of $\mathcal{D}_i$.
Based on the above solution to the decision variables, we get $\alpha_1 = 1, \alpha_2 = 0, \alpha_3 = 0, \beta_{12} = 1, \beta_{13} = 0, \beta_{22} = 0, \beta_{23} = 1, \beta_{33} = 1$.
$n'_1 = n_1 \cdot \lambda_1^{fb_1(1)}$ $n'_2 = n_2$ $n'_3 = n_3$
$m_2 = \min\{n'_1, n'_2\}$ $m'_2 = m_2 \cdot (\omega_{1,2})^{f_1(1)} \cdot \rho_1$ $m_3 = \min\{m'_2, n'_3\}$ $m'_3 = m_3 \cdot (\omega_{3,3} . \lambda_3)^{f_3(1)} \cdot (\omega_{2,3} \cdot \omega_{2,2} \cdot \lambda_2)^{f_2(1)} \cdot \rho_2$

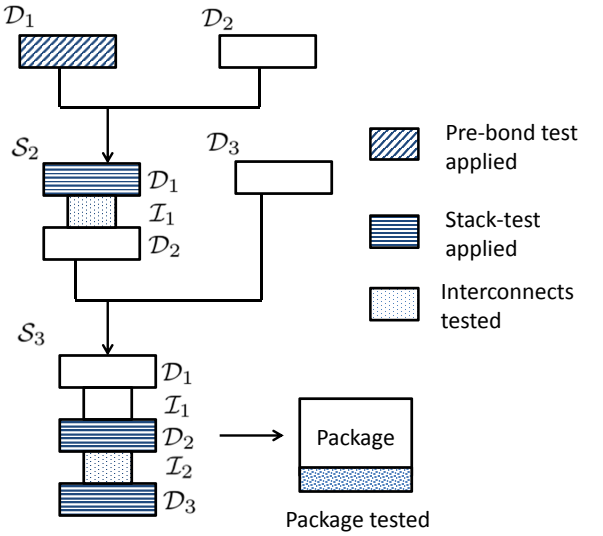

Fig. 6. A selected test insertions for a small example of 3D integration with three dies.

```
/*mincost is a global variable storing the minimum cost found.*/
mincost ← ∞
for i ← 1 to l do
    /*Computing cost without any dividers*/
    cost ← compute_cost()
    if cost < mincost then
        mincost ← cost
    end if
    for k ← 1 to l do
        /*Solving problem P(i,k) using d dividers*/
        find_partition(i,k)
    end for
end for
```

Fig. 7. The top-level heuristic procedure.

find_partition(i, k)

```
1: cost ← ∞;
2: d ← 1
3: while true do
4:     M[i,k,d].div_pos ← 0
5:     M[i,k,d].selected_test ← 0
6:     if d > k − i then
7:         return
8:     end if
9:     for pos ← k − 1 to 1 do
10:        (t, c) ← select_test(i, k, pos, d − 1)
11:        if t ≠ 0 then
12:            if c < cost then
13:                cost ← c
14:                M[i,k,d].div_pos ← k − 1
15:                M[i,k,d].selected_test ← t
16:            end if
17:            if c < mincost then
18:                mincost ← c
19:                Update the set of x and y variables corresponding to i, k
                   and test
20:            end if
21:        end if
22:    end for
23:    d++
24: end while
```

Fig. 8. Procedure find_partition

select_test(i, k, pos, d)

```
1: τ ← test set containing all tests applicable on D_i at the current stage
   (i.e. a set consisting of tests T_ij (or PB_ij, if pre-bond stage), such that
   a_ijk = 1).
2: t ← 0
3: //Remove tests that have already been applied on D_i using solutions
   already constructed for smaller subproblems P(i, k').
4: k' ← pos;
5: while true do
6:     τ ← τ − M[i, pos, d].selected_test
7:     k' = M[i, pos, d].div_pos
8:     d = d − 1
9:     if k' = 0 then
10:        break
11:    end if
12: end while
13: c ← ∞
14: //select best test from leftover tests in τ.
15: for each test ϵ τ do
16:    Assign test to the current stage
17:    cost ← compute_cost()
18:    if cost < c then
19:        c ← cost
20:        t ← test
21:    end if
22: end for
23: return (t, c)
```

Fig. 9. Procedure select_test