Testing of 3D-Stacked ICs With Hard- and Soft-Dies – A Particle Swarm Optimization Based Approach

Rajit Karmakar, Aditya Agarwal and Santanu Chattopadhyay
Dept. of Electronics & Electrical Comm. Engineering
Indian Institute of Technology Kharagpur, India, Kharagpur, 721302
Email: {rajit, adityaagarwal, santanu}@ece.iitkgp.ernet.in

***Abstract*—This paper presents a test architecture optimization and test scheduling strategy for TSV based 3D-Stacked ICs (SICs). A test scheduling heuristic, that can fit in both session-based and session-less test environments, has been used to select the test concurrency between the dies of the stack. The proposed method minimizes the overall test time of the stack, without violating the system level resource and TSV limits. Particle Swarm Optimization (PSO) based meta search technique has been used to select the resource allocation of individual dies and also their internal test schedules. Incorporation of PSO in two stages of optimization produces a notable reduction in the overall test time of SIC. Experimental results show that upto 51% reduction in test time can be achieved using our strategy, over the existing techniques.**

***Keywords*—3D-SIC, TSV, Test scheduling, PSO, Optimization.**

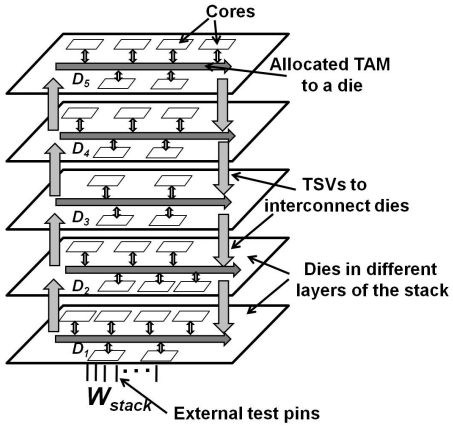

Figure 1. A typical example of 3D-SIC

I. Introduction

[1] With increasing demand for high performance and low-power chips, present day's semiconductor industry is heading towards smaller feature size with reduced chip area. Interconnects, which cannot be scaled down with transistors, are becoming main stumbling block in IC design. Long interconnects in 2D-ICs hamper circuit performance with its high delay and power consumption. Recently, 3D-IC has emerged to be a potential solution to this problem. Instead of designing 2D-IC with long global interconnects, interconnect lengths can be reduced significantly by designing circuit components into several layers and bonding them together. This helps to achieve high bandwidth, low latency circuit with higher packaging density and low footprint. Based on different stacking methodologies, 3D stacking can be categorized as wafer-to-wafer, die-to-wafer, and die-to-die stacking [1]. In 3D-SICs different dies are stacked and interconnected using through-silicon vias (TSVs) bonding. These TSVs are vertical metal interconnects that can be integrated into a substrate during manufacturing. TSVs are very important in 3D integration as they are used to provide functional signals, power/ground, clock, as well as test access to logic blocks of different layers of the device [2]. Figure 1 shows a typical example of 3D-SIC with dies at different layers of the stack.

Although 3D-SIC provides several advantages over 2D-IC, testing of 3D-SIC has become more challenging because of its high complexity. Individual dies need to be tested before stacking (pre-bond testing [3]) to ensure stacking of defect-free dies. Post-bond testing [3] is required after completion of stacking of all the dies, to ensure defect-free thinning, alignment, and bonding during stacking. Mid-bond testing [3] or partial-stack testing is also carried out optionally in the intermediate process of stacking of pre-bond tested dies. In post-bond testing, test data can only be fed through the pins of the lowest die of the stack. A 3D test access mechanism (3D-TAM) must be designed to transport the test data to the cores of the dies at different layers of the stack to facilitate both pre-bond and post-bond testing [4]. TSVs are used to route the 3-D TAM to different layers of the stack. However, total number of TSVs must be within a certain limit to reduce its area overhead as well as TSV defects and TSV-induced defects in devices [5].

Some initial works in 3D testing have proposed wrapper design for TSV based 3D-SICs [2] while some other works [6], [7] have considered test architecture design. However, test optimization method has not been discussed in these papers. Although, the paper [1] has proposed test optimization techniques for 3D SoC consisting of cores distributed over multiple layers, the claim of no maximum limit on TSVs may not be a valid assumption. Moreover, the proposed 3D test architecture may not be feasible in practical cases. An Integer Linear Programming (ILP) based optimization method has been proposed in [8] for complete stack testing as well as for multiple test insertion during bonding. Another work [3] has considered a matrix partitioning based cost model to account for various test costs incurred during 3D integration and a also test flow selection to optimize the test cost.

Test architecture optimization for 3D-SIC can be divided into two parts, *Die level optimization* to be followed by *stack level optimization*. In *stack level optimization*, a 3D test architecture and test schedule is developed for all the dies of the stack, under the resource and TSV constraints. Given an allocated TAM to a die, *die level optimization*

[1]This work is partially supported by the research project No. 9(5)/ 2010-MDD dt. 23/1/2011, sponsored by the DeitY, Govt. of India

978-1-4799-7597-6/15/$31.00 ©2015 IEEE

deals with resource allocation to individual cores and selection of core ordering during testing to minimize the test time of the die. The authors in [4] have classified 3D-SIC into three categories, "$hard\ die$", "$soft\ die$" and "$firm\ die$". 2D test architectures of the dies are fixed for 3D-SIC with $hard\ die$, while for $soft\ die$, test engineers have the flexibility to decide how much test resources should be allocated to each die, $die\ level\ optimization$ is also possible. In $firm\ dies$, test resources and TSV requirements of the dies can be reduced at the expense of an added serial/parallel conversion hardware. The authors in [4] have proposed an ILP based test architecture optimization, where they have considered session-based testing for all three categories of 3D-SICs. However, the schedules generated using this method, fail to satisfy the maximum TSV limit for hard dies in some cases (as discussed in Section IV).

In this paper, we have presented test architecture optimization and test schedule generation strategies for 3D-SIC with hard dies as well as soft dies. A TSV constrained test scheduling heuristic has been used to select the test concurrency between the dies without violating the resource and TSV constraints. Both session-based and session-less testings have been considered. Particle Swarm Optimization (PSO) based search procedure is used to take the decision about the test resource allocation for individual dies for 3D-SIC with soft die. This helps to evolve towards better test schedule of the stack. Although, $Die\ level\ optimization$, which is similar to test architecture optimization of 2D-SoC, has been well researched in the past few years [9], [10], none of the reported approaches could produce better solution for all the test cases. This justifies the search for newer techniques. So, unlike other 3D test architecture optimization approaches [4], which consider one of the previously reported 2D test architecture optimization methods [10], we have developed a new PSO guided 2D test architecture optimization technique to generate test schedule of individual dies. It gives us the flexibility of both $die\ level$ and $stack\ level\ optimization$. Experimental results show that our strategy can achieve upto 51% improvement in test time over the works reported in the literature.

The rest of the paper is organized as follows. Test architecture optimization strategy for 3D-SIC with hard die has been described in Section II. Section III presents the PSO guided test architecture optimization technique for 3D-SIC with soft die. Results of our experimentation and related discussions have been presented in Section IV. Section V draws the conclusion of the paper.

II. Test-Architecture Optimization For 3D-SIC With Hard Die

For 3D-SIC with hard dies, the vendors provide fabricated dies with fixed test bandwidth and test time to the 3D-integrator. Hard dies offer less flexibility for optimization, as the test engineers are limited to selection of test concurrency between dies. The test architecture and test schedule problem of 3D-SIC with hard dies can be described as follows.

A. Problem Formulation

Suppose a stack consisting of M dies D_i $(1 \leq i \leq M)$ is to be tested with a maximum available test pins, W_{stack} and maximum number of allowable TSVs, TSV_{stack} to route the 3D-TAM. Each die D_i has a predefined test pin value of WD_i ($WD_i \leq W_{stack}$) and associated test time, TD_i. Determine an optimal TAM design and corresponding test schedule for the stack such that the total test time TT for the stack is minimized without violating resource and TSV constraints.

B. Scheduling Strategy

As all the dies have to be tested using predefined number of test resources and test architecture of each die is fixed, $die\ level\ optimization$ is not applicable for 3D-SIC with hard die. We can only decide the test concurrency and ordering of the dies in the schedule. In this section, we present a test scheduling heuristic considering both session-based testing and session-less testing.

*1) **Session-based testing**:* For session-based testing, maximum test time of all the dies tested concurrently in k^{th} test session (TS_k), is the test time (TST_k) of that session. It may be noted that, if all the dies are tested serially, a maximum number of M test sessions are possible. Total test time (TT) of the stack is the summation of all the test session's time. Our test scheduling heuristic $3D\_Test\_Schedule$ (Algorithm 1) starts with sorting the dies D_i $(1 \leq i \leq M)$ in descending order of test time TD_i. We choose a die D_i in sorted order and check its schedulability in a test session without violating resource (W_{stack}) and TSV (TSV_{stack}) constraints. A $TSV\_Checker$ (Procedure 2)checks TSV requirement between different layers of the stack to facilitate concurrent testing of D_i with other dies tested in the same session TS_k. The number of TSVs required between layer j and $j-1$ is determined by two factors- (i) the maximum number of test pins required by a layer at or above j and (ii) the sum of test pins for parallel tested dies at or above layer j in the same test session. TSVs between layers j and $j-1$ must be equal to the maximum of these two quantities. If D_i does not satisfy resource and TSV constraints, we move to the next die or try to schedule D_i in next test session TS_{k+1}. We repeat this process until all the dies get scheduled. However, it must be kept in mind that, TSVs cannot be allocated dynamically. It may happen that the TSV requirement between layers j and $j-1$ (TSV_j) in test session TS_{k+1} is less than that in TS_k. Still we have to use TSV_{j_k} number of TSVs for TS_{k+1}, otherwise it will hamper the test schedule of TS_k. So, we only update the TSV_j if some later test session requires more TSVs between layers j and $j-1$. In that case, earlier test session will have some unused TSVs. However, total TSV requirement at any test session must not violate the limit of TSV_{stack}.

*2) **Session-less testing**:* In a session-based testing, the dies tested in a test session, may not have the same test times. Session time depends on the test time of the die with the largest test time, scheduled in that session. As no new die can be scheduled in the middle of a test session, the resources occupied by the die, which finishes its testing earlier in the session, have to remain unutilized till the session ends. As a result, total test time increases unnecessarily. This shortcoming of session-based testing can be overcome using session-less testing, which allows scheduling of a die at any point in the schedule depending upon resource availability. In session-less testing, instead of considering different test sessions, we consider two important data-structures throughout the schedule. $Break\_Point\_List(BP_{stack})$ note the points in the schedule,

Algorithm 1: ***3D_Test_Schedule***

Input : List of dies $D_i (1 \leq i \leq M)$ to be scheduled with assigned test pins WD_i and test time TD_i; W_{stack}: maximum test pins of the stack; TSV_{stack}: maximum TSV limit;

Var : TS_k**:** k^{th} test session ($k \leq M$); TST**:** Test session time; TSW**:** Test session width; TT**:** Total time; TSV_j**:** number of TSVs between layer j and $j-1$ ($2 \leq j \leq M$);

begin

- $TT \longleftarrow 0; k \longleftarrow 1$;
- **for** $j \leftarrow 2$ **to** M **do**
  - $TSV_j \longleftarrow 0$;
- Sort the dies in descending order of TD_i;
- Mark all dies as unscheduled;
- **while** there exists unscheduled die **do**
  - $TSW_k \longleftarrow W_{stack}$;
  - $TST_k \longleftarrow 0$;
  - **while** there exists unscheduled and unchecked die with $WD_i \leq TSW_k$ **do**
    - Select an unscheduled die D_i in sorted order;
    - **if** *($WD_i \leq TSW_k$)* **then**
      - $TSV\_Checker()$;
      - **if** yes **then**
        - Schedule D_i; Mark D_i as scheduled;
        - $TSW_k = TSW_k - WD_i$;
        - **if** *($TD_i > TST_k$)* **then**
          - $TST_k = TD_i$;
      - **else**
        - Check next die of the sorted list;
  - $TT = TT + TST_k$; k++;
- Return TT as total schedule generation time;

Procedure 2: ***TSV_Checker***

$TSV_{total} \longleftarrow 0$;
for $j \leftarrow 2$ **to** M **do**
- $TSV_{parallel} \longleftarrow 0$;
- $WD_{max} \longleftarrow 0$;
- **for** $l \leftarrow j$ **to** M **do**
  - **if** *($WD_l > WD_{max}$)* **then**
    - $WD_{max} = WD_l$;
  - **if** D_l is being tested **then**
    - $TSV_{parallel} = TSV_{parallel} + WD_l$;
- TSV_{j_k} = *max* $(WD_{max}, TSV_{parallel}, TSV_{j_{k-1}})$;
- $TSV_{total} = TSV_{total} + TSV_{j_k}$;

if *($TSV_{total} < TSV_{stack}$)* **then**
- *Return yes;*

else
- *Return no;*

where a die finishes its test and releases occupied test pins. An unscheduled die can be scheduled at any of the break-points, $bp_{stack_k} \in BP$. $Available\_Test\_Pins(ATP_{stack})$ keeps track of the available test resources at each bp_{stack_k}. We choose a die D_i in descending order of test time and check its schedulability in the minimum break-point bp_{stack_k}. The same $TSV\_Checker$ mentioned in session-based testing, is used to check TSV constraint at each bp_{stack_k}, where an unscheduled die can be scheduled. BP_{stack} and ATP_{stack} get updated with the schedule of each die. If sufficient resources are not available to schedule any unscheduled die at any break-point, we shift to the next break-point. This process continues till all the dies get scheduled. Finally maximum test finish time among all the dies is reported as the test time TT of the stack.

III. Test-Architecture Optimization For 3D-SIC With Soft Die

3D-SIC with soft dies provide better opportunity of optimization of test time than 3D-SICs with hard dies. Unlike 3D-SIC with hard dies, the test architecture of individual dies is not fixed for soft dies. Any number of test resources within W_{stack} can be allocated to any die and 2D test architecture of individual dies can also be developed. The test architecture optimization and test schedule generation problem for 3D-SIC with soft dies can be formulated as follows.

A. *Problem Formulation*

Suppose a stack consisting of M dies D_i ($1 \leq i \leq M$) is to be tested with a maximum available test pins W_{stack}. Assume that the maximum number of allowable TSVs to route the 3D-TAM is TSV_{stack}. Each die D_i consists of N_i number of cores C_1, $C_2 \ldots C_{N_i}$. The number of test patterns required to test core C_j ($1 \leq j \leq N_i$) is p_j. Determine an optimal TAM design and corresponding test schedule for the stack, as well as for each die, such that the total test time TT for the stack is minimized without violating resource and TSV constraints.

B. *Scheduling Strategy*

Although the flexibility of choosing TAM width for individual dies and *die level optimization* for each die offer better opportunity to optimize overall test time of the stack, test architecture design and scheduling of 3D-SIC with soft dies becomes more complex than hard die cases. Both the problems of (i) 3D-TAM distribution among dies of the different layers and (ii) determining an optimized test schedule for cores in a die, are NP-hard [4]. We have approached these problems in two steps. First, we have carried out *die level optimization*. The results of *die level optimization* are used in later part of *stack level optimization*, where we have carried out the 3D-TAM design. Particle Swarm Optimization (PSO) guided heuristics have been used in both levels of optimization.

1) ***Die level optimization****:* Suppose a die with N cores C_1, $C_2 \ldots C_N$ is to be tested with a maximum of W_{die} test pins. The die level test scheduling problem is to allocate test resources and test times to the cores so that, the test time (TAT) of the die is minimized.

We have used PSO guided Rectangular 2D-bin packing approach to generate optimized test schedule. Each core C_i ($1 \leq i \leq N$) is represented by a set of wrapper configurations R_i. The test resource of core C_i with j^{th} wrapper configuration

can be represented by a rectangle whose height and width represent allocated test pins (w_{ij}) and the corresponding test time ($T(w_{ij})$) respectively. To get a schedule for the full die, the rectangles are to be packed into a bin of fixed height (W_{die}) so that TAT (width of the bin) is minimized. Selection of one test rectangle per core has been performed using PSO.

- *Particle Swarm Optimization Formulation:*

PSO is a population based evolutionary technique designed by Eberhart and Kennedy [11]. It starts with an initial population of particles. Each particle corresponds to a solution to the optimization problem being solved. Each particle has its fitness value. Particles evolve over generations guided by three factors – its own intelligence ($pbest$), global (swarm) intelligence ($gbest$), and the inertia factor.

Each core has a set of test rectangles and the maximum number of rectangles for any core be R. Let $B = \lceil \log_2^R \rceil$. A particle consists of $N \times B$ number of bits. First B bits identify the test rectangle selected for the first core, second B bits for the second core, and so on. Figure 2 shows a sample particle with N = 4 and B = 4. Fitness of a particle is equal to the total test time (TAT) of the SoC after scheduling the test rectangles using the $2D\_Test\_Schedule$ procedure (Algorithm 3). For the initial generation, particles are generated randomly. In the successive generations, new particles are created using a $replace$ operator, which attempts to align a particle with its $pbest$ and the $gbest$ particles, with some probability. For the sake of this alignment, the $replace$ operator is applied at each bit position of a particle. For bit position i of a particle, the bit is replaced by the corresponding bit of $pbest$ particle with probability α. After the operator has been applied for $pbest$, the same is done with respect to $gbest$ with probability of replacement, β. In our experimentation, we have kept both α and β values at 0.1.

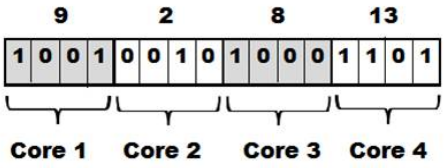

Figure 2. Sample particle structure of 4 cores with W_{die} =16 (B = 4)

- *Scheduling of cores in a die:*

The algorithm takes as input the rectangle set corresponding to a particle, the maximum test pins W_{die}. BP_{die} and ATP_{die} keep track of the scheduling points and corresponding resource availability at those points. As the still unscheduled cores get scheduled, the list BP_{die} and ATP_{die} get updated. The rectangles are sorted on their area values (test pins (w) $\times$ test time (T)) in a descending order. The break-point list BP_{die} is scanned from the minimum to the maximum value. For the break-point bp_{die_k}, the algorithm scans the unscheduled rectangle list to check for the largest rectangle that can be scheduled at bp_{die_k}. If none are feasible, the algorithm advances to the next break-point. When rectangles corresponding to all cores have been scheduled, the maximum end time of testing of all cores gives the total test application time for the SoC. The $2D\_Test\_Schedule$ algorithm to produce the schedule is presented next.

Algorithm 3: ***2D_Test_Schedule***

Input : List of rectangles to be scheduled; W_{die}, the maximum test pins;
Var : BP_{die}**:** A list of break points; ATP_{die}**:** List of available test pins at each break point $bp_{die} \in BP_{die}$;
begin
  Sort list of rectangles on decreasing area;
  Mark all rectangles as unscheduled;
  while there exists unscheduled rectangles **do**
    Check if any rectangle picked up in sorted order can be scheduled at next break point bp_{die_k} with available TAM resource atp_{die_k};
    if yes **then**
      Update BP_{die}, ATP_{die} and Rectangle List;
      Mark corresponding rectangle scheduled;
    else
      Continue with next $bp_{die_k} \in BP_{die}$;
  Return the maximum test end time of all rectangles;

2) ***Stack level optimization****:* In $stack\ level\ optimization$, total test resources W_{stack} are optimally allocated to all the dies and a test schedule of the dies is generated in such a way, so that the total test time TT to test the stack is minimized. The problem is similar to test architecture optimization of 3D-SIC with hard dies with an added complexity of selection of TAM allocation of each die. Again, it is an NP-hard problem [4], which we have solved by incorporating meta search technique like PSO with the $3D\_Test\_Schedule$ heuristic mentioned in Section II. Similar kind of PSO formulation used for $die\ level\ optimization$, is adopted here. Each die D_i is represented by a set of assigned test pin value WD_{i_k} ($1 \leq k \leq W_{stack}$) and associated test time TD_{i_k}. These values are obtained from $die\ level\ optimization$ by varying the W_{die} value from 1 to W_{stack} and noting the corresponding test times. Each particle selects a tuple of assigned test pins and associated test time (WD_{i_k}, TD_{i_k}) for each die. Fitness of a particle is evaluated by calculating the test time of the stack by using $3D\_Test\_Schedule$ heuristic. The $replace$ operator, guided by $pbest$ and $gbest$, evolves particles towards better solution. A similar $TSV\_Checker$ (as mentioned in Section II) is used to check TSV constraint. Both session-based and session-less testing have been performed. Figure 3 describes the total flow of the test architecture design and test scheduling procedure for 3D-SIC with soft dies.

IV. Experimental Results

In this section we present the results of our experimentation for both $hard\ die$ and $soft\ die$ cases. For the sake of comparison, we have considered the same 3D-SIC benchmarks presented in [4]. Figure 4 presents the SICs, which are formed using ITC'02 benchmarks as dies. For 3D-SIC with hard die, we have considered the same test architectures of individual dies reported in [4]. Table I reports the details of test architectures of each die.

Table II shows the comparison of our test scheduling approaches (both session-based and session-less) with the

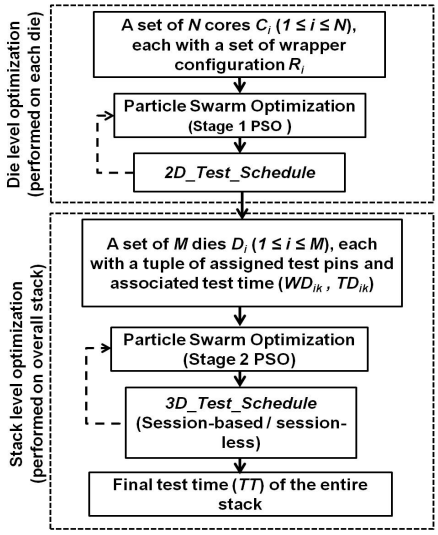

Figure 3. Test Flow for 3D-SIC with soft dies.

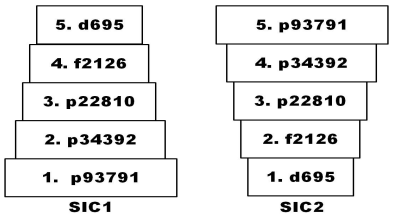

Figure 4. 3D-SIC benchmarks [4]

approach presented in [4] for 3D-SIC with hard dies. The first two columns of Table II describe the maximum allowable TSV limit and number of allocated test pins to test the SIC. Columns 3, 4 and 8, 9 present the test time and corresponding test schedule reported in [4] for SIC1 and SIC2 respectively. The test times and schedules obtained from our $3D\_Test\_Schedule$ heuristic considering session-based (SB) testing for both SIC1 and SIC2 are reported in columns 5, 6 and 10, 11 respectively while columns 7 and 12 report our session-less (SL) test times for SIC1 and SIC2 respectively. It may be noted from Table II that the schedules reported in [4] violate maximum TSV limit TSV_{stack} in several cases. For example, the number of test pins of dies 3, 4 and 5 of SIC1 are 25, 20 and 15 respectively. To facilitate parallel testing of these three dies, 60 TSVs (25 + 20 + 15) are required between each of the layers 2 and 1 and layers 3 and 2. The TSV requirement between dies 4 and 3 is 35 (20 + 15) and finally 15 TSVs are required between layers 5 and 4. So, the schedule reported in [4], for TSV_{stack} = 160 and W_{stack} = 60, which shows parallel scheduling of dies 3, 4 and 5 requires a total of 170 (60 + 60 + 5 + 15 = 170) TSVs, which clearly violates the maximum TSV limit. Similarly, for SIC2, all the schedules (W_{stack} = 60, 70, 80, 90 and 100), which report parallel testing of dies 4 and 5, require a total of 195 TSVs. It again violates the maximum TSV limit. In contrast, no TSV limit violation can be noted in the schedules generated by our $3D\_Test\_Schedule$ heuristic. Moreover, our session-based test time results are same with the results reported in [4], for all the cases where [4] does not violate TSV_{stack}. Session-less testing further improves test time over session-based testing.

Next, we present the results of *stack level optimization* for soft dies. The *die level optimization* results of individual dies are fed to PSO guided $3D\_Test\_Schedule$ heuristic, which evolve over generations to explore a large search space of solution to find a near optimal test schedule of the stack. Table III presents the comparison between our scheduling strategy and the approach reported in [4] for different values of W_{stack} for SIC1. Column 3 of Table III reports the test time reported in [4] while columns 4 and 9 report our session-based and session-less test time results respectively. Column 6 reports the test schedule obtained from our session-based testing. Corresponding test pin allocation to each die and associated TSV requirements between all the layers are reported in columns 7 and 8 respectively. For example, to test SIC1 with 40 test pins, 26 test pins are allocated to die 1. Dies 2, 3, 4 and 5 are tested using 14, 40, 34 and 26 test pins respectively. Similarly 40, 40, 34 and 26 TSVs are required between dies 2 and 1, 3 and 2, 4 and 3, 5 and 4 respectively. It may be noted that, both of our session-based and session-less test architecture and scheduling strategies can reduce test time of SIC1 upto 51% compared to the results reported in [4]. Figure 5 presents a pictorial illustration of the session-based test schedule of SIC1 for W_{stack} = 70 and TSV_{stack} = 140, obtained from our PSO guided $3D\_Test\_Schedule$ heuristic. All the dies are tested in three test sessions in descending order of their individual test times and satisfying the resource and TSV constraints. The figure describes how $TSV_{Checker}$ updates the TSV requirements between different layers of the stack in successive test sessions. The number of TSVs used and the actual number of TSVs assigned between each two dies in each session is mentioned in the figure. It may be noted that, although 14 TSVs are required to send test data to die 3 in session 1, we have to assign 48 TSVs between die 2 and 1, as die 2 is allocated 48 test pins. The TSV value between die 2 and 1 gets updated to 66 in session 2, to facilitate parallel testing of dies 2 and 4. In the final session, only 24 out of 66 TSVs are used between die 2 and 1. Rest of the TSVs remain unused. Table IV reports the test time results of our techniques for SIC2. It may be noted from Table III and Table IV that session-less testing can reduce test time over session-based testing for both the SIC1 and SIC2.

Table I. TEST LENGTHS AND TEST PINS FOR HARD DIES [4]

Die	$d695$	$f2126$	$p22810$	$p34392$	$p93791$
Test length	96927	669329	651281	1384949	1947063
Test pin	15	20	25	25	30

V. CONCLUSION

In this paper we have presented a PSO guided heuristic for test architecture optimization and test scheduling for 3D-SIC with hard dies and soft dies. Both session-based and session-less testing have been considered. Our heuristic can produce optimized results without violating resources and TSV constrains. Incorporation of PSO in both 2D and 3D optimizations has given us the flexibility of two stage optimization, which has been instrumental in minimizing the overall test time of the stack to a large extent.

REFERENCES

[1] L. Jiang, L. Huang, and Q. Xu, "Test architecture design and optimization for three-dimensional socs," in *Proc. Conf. Design, Automation and Test in Europe*, 2009, pp. 220–225.

Table II. COMPARISON OF OUR TEST SCHEDULING APPROACH (SESSION-BASED AND SESSION-LESS) WITH [4] FOR 3D-SIC WITH HARD DIE

3D-SIC with hard die		SIC1					SIC2				
TSV_{stack}	W_{stack}	Test time [4]	Schedule [4]	Test time (SB)	Schedule (SB)	Test time (SL)	Test time [4]	Schedule [4]	Test time (SB)	Schedule (SB)	Test time (SL)
160	30	4748920	1, 2, 3, 4, 5	4748920	1, 2, 4, 3, 5	4748920	4748920	1, 2, 3, 4, 5	4748920	5, 4, 2, 3, 1	4748920
160	40	4652620	1, 2, 3, 4 ‖ 5	4652620	1, 2 ‖ 5, 4, 3	4652620	4652620	1 ‖ 3, 2, 4, 5	4652620	5, 4 ‖ 1, 2, 3	4652620
160	50	3428310	1 ‖ 4, 2 ‖ 3, 5	3428310	1 ‖ 4, 2 ‖ 3, 5	3332012	3428310	1, 2 ‖ 5, 3 ‖ 4	3428310	5 ‖ 2, 4 ‖ 3, 1	3332012
160	60	2616390	1 ‖ 2, 3 ‖ 4 ‖ 5	2712690	1 ‖ 2, 4 ‖ 3, 5	2598340	2616390	1 ‖ 2 ‖ 3, 4 ‖ 5	3428310	5 ‖ 2, 4 ‖ 3, 1	3332012
160	70	2616390	1 ‖ 2 ‖ 5, 3 ‖ 4	2616390	1 ‖ 2 ‖ 5, 4 ‖ 3	2598340	2616390	1 ‖ 2 ‖ 3, 4 ‖ 5	3332012	5 ‖ 2 ‖ 1, 4 ‖ 3	3332012
160	80	2598340	1 ‖ 2 ‖ 4, 3 ‖ 5	2598340	1 ‖ 2 ‖ 4, 3 ‖ 5	1947063	2616390	1 ‖ 2 ‖ 3, 4 ‖ 5	3332012	5 ‖ 2 ‖ 1, 4 ‖ 3	3332012
160	90	2598340	1 ‖ 2 ‖ 4, 3 ‖ 5	2598340	1 ‖ 2 ‖ 4 ‖ 5, 3	1947063	2616390	1 ‖ 2 ‖ 3, 4 ‖ 5	3332012	5 ‖ 2 ‖ 1, 4 ‖ 3	3332012
160	100	2043360	1 ‖ 2 ‖ 3 ‖ 4, 5	2043360	1 ‖ 2 ‖ 4 ‖ 3, 5	1947063	2616390	1 ‖ 2 ‖ 3, 4 ‖ 5	3332012	5 ‖ 2 ‖ 1, 4 ‖ 3	3332012

Table III. COMPARISON OF TEST SCHEDULING RESULTS BETWEEN [4] AND OUR APPROACHES (SESSION-BASED AND SESSION-LESS) FOR SIC1 (3D-SIC WITH SOFT DIE)

TSV_{stack}	W_{stack}	Test time [4]	Test time SB	Impv. over [4] %	Schedule (SB)	Test pins (SB)	TSV (SB)	Test time (SL)	Impv. over [4] %
140	30	4795930	3755885	**21.69**	1, 2, 3, 4, 5	30, 30, 30, 30, 30	30, 30, 30, 30	3724309	**22.34**
140	40	3841360	2881060	**25.0**	2 ‖ 1, 3, 4, 5	26, 14, 40, 34, 26	40, 40, 34, 26	2802280	**27.05**
140	50	3090720	2335908	**24.42**	1 ‖ 2, 4 ‖ 3 ‖ 5	32, 18, 22, 18, 10	50, 50, 28, 10	2265193	**26.71**
140	60	2873290	1915438	**33.34**	1 ‖ 3, 2 ‖ 4, 5	48, 44, 12, 16, 26	60, 26, 26, 26	1875655	**34.72**
140	70	2743320	1701271	**37.98**	1 ‖ 3, 2 ‖ 4, 5	56, 48, 14, 18, 24	66, 24, 24, 24	1623521	**40.82**
140	80	2439760	1481323	**39.28**	1 ‖ 4, 2 ‖ 3, 5	66, 56, 24, 14, 18	80, 24, 18, 18	1448635	**40.62**
140	90	2395760	1396718	**41.70**	1 ‖ 4, 2 ‖ 3, 5	74, 56, 24, 16, 18	80, 24, 18, 18	1267313	**47.10**
140	100	2369680	1153410	**51.32**	4 ‖ 1 ‖ 3 ‖ 2, 5	50, 28, 12, 10, 30	50, 30, 30, 30	1153169	**51.33**

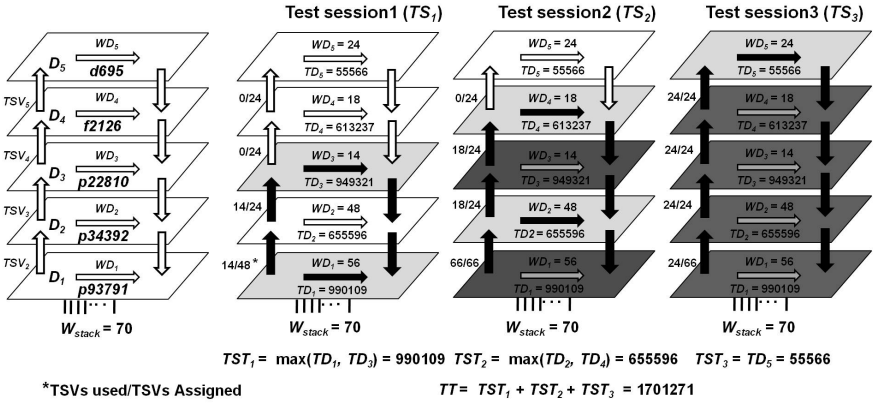

Figure 5. Session-based test schedule of SIC1 for W_{stack} = 70 and TSV_{stack} = 140 (considering 3D-SIC with soft dies)

Table IV. TEST SCHEDULING RESULTS OF OUR APPROACHES (SESSION-BASED AND SESSION-LESS) FOR SIC2 (3D-SIC WITH SOFT DIE)

TSV_{stack}	W_{stack}	Test time (SB)	Schedule (SB)	Test pins (SB)	TSVs (SB)	Test Time (SL)
140	30	3731707	5, 4, 3 ‖ 2, 1	30, 14, 16, 30, 30	30, 30, 30, 30	3695406
140	40	2960916	5 ‖ 3, 4 ‖ 2, 1	40, 12, 10, 28, 30	40, 40, 30, 30	2874935
140	50	2846974	4 ‖ 5 ‖ 2, 3 ‖ 1	10, 10, 40, 12, 22	44, 40, 34, 22	2735816
140	60	2846974	4 ‖ 5 ‖ 2 ‖ 1, 3	10, 10, 40, 12, 22	44, 40, 34,22	2735816
140	70	2846974	4 ‖ 5 ‖ 2 ‖ 1, 3	10, 10, 40, 12, 22	44, 40, 34, 22	2735816
140	80	2846974	4 ‖ 5 ‖ 2 ‖ 1, 3	28, 10, 40, 12, 22	44, 40, 34, 22	2735816
140	90	2846974	4 ‖ 5 ‖ 2 ‖ 1, 3	28, 10, 40, 12, 22	44, 40, 34, 22	2735816
140	100	2846974	4 ‖ 5 ‖ 2 ‖ 1, 3	46, 10, 40, 12, 22	22, 20, 17, 11	2735816

[2] B. Noia, K. Chakrabarty, and Y. Xie, "Test-wrapper optimization for embedded cores in tsv-based three-dimensional socs," in *Comput. Design, 2009. ICCD 2009. IEEE Int. Conf.* IEEE, 2009, pp. 70–77.

[3] M. Agrawal and K. Chakrabarty, "Test-cost optimization and test-flow selection for 3d-stacked ics," in *VLSI Test Symposium (VTS), 2013 IEEE 31st*, April 2013, pp. 1–6.

[4] B. Noia, K. Chakrabarty, S. K. Goel, E. J. Marinissen, and J. Verbree, "Test-architecture optimization and test scheduling for tsv-based 3-d stacked ics," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 30, no. 11, pp. 1705–1718, 2011.

[5] K. Chakrabarty, S. Deutsch, H. Thapliyal, and F. Ye, "Tsv defects and tsv-induced circuit failures: The third dimension in test and design-for-test," in *Rel. Physics Symp. (IRPS), 2012 IEEE Int.*, 2012, pp. 5F–1.

[6] C.-Y. Lo, Y.-T. Hsing, L.-M. Denq, and C.-W. Wu, "Soc test architecture and method for 3-d ics," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 29, no. 10, pp. 1645–1649, 2010.

[7] E. J. Marinissen, J. Verbree, and M. Konijnenburg, "A structured and scalable test access architecture for tsv-based 3d stacked ics," in *VLSI Test Symp. (VTS), 2010 28th*. IEEE, 2010, pp. 269–274.

[8] B. Noia, K. Chakrabarty, and E. J. Marinissen, "Optimization methods for post-bond die-internal/external testing in 3d stacked ics," in *Test Conf. (ITC), 2010 IEEE Int.* IEEE, 2010, pp. 1–9.

[9] C. Giri, S. Sarkar, and S. Chattopadhyay, "Test scheduling for core-based socs using genetic algorithm based heuristic approach," in *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*. Springer, 2007, pp. 1032–1041.

[10] S. K. Goel and E. J. Marinissen, "Control-aware test architecture design for modular soc testing," in *Test Workshop, 2003. Proc. The Eighth IEEE European*. IEEE, 2003, pp. 57–62.

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, Nov 1995, pp. 1942–1948 vol.4.