Optimal Stacking of SOCs in a 3D-SIC for Post-Bond Testing

Manjari Pradhan*, Chandan Giri†, Hafizur Rahaman†, Debesh K. Das*
*Dept. of Computer Science, Jadavpur University, Jadavpur, Kolkata-32, India
pradhanmanjari@gmail.com , debeshd@hotmail.com
†Dept of Information Technology, Bengal Engineering and Science University Shibpur, Howrah, India
chandangiri@gmail.com , rahaman_h@yahoo.co.in

***Abstract*—3D IC testing is one of the major concern in the semiconductor industry today. Multiple subsequent testing of partial stack during 3D assembly is required due to the die stacking steps of thinning, alignment and bonding. In this paper we address the problem of minimizing the total time of partial stack and complete stack testing. We analyze how the stacking sequence of different System-on-Chips (SOCs) in a 3D Stacked Integrated Circuit (SIC) affects the total test time. We propose an algorithm to find this stacking sequence to achieve the minimum test time. Our algorithm is run on ITC'02 benchmarks and the results are shown.**

***Index Terms*—Three-dimensional integration, through silicon via, test access mechanism, system-on-a-chip (SOC)**

I. Introduction

As scientists and engineers are working hard to continue the trend of Moore's law [1] in designing the integrated circuits, some inherent problems are becoming inevitable in these designs. Increasing interconnect cost between several modules is an important issue in this respect [2], [3]. One alternative promising option to avoid such problems is the building of integrated circuits in 3D architecture [4]. In 3D architecture, a stack of multiple device layers, with direct vertical tunnelling through them, are put together on the same chip. Several methods exist for this vertical interconnection such as, wire bonding, micro-bump, contactless approaches and through silicon vias (TSVs) [5]. Among them, the most reliable way to achieve the 3D stacked integrated circuits (SIC) is TSVs, though they require more manufacturing steps [6]. TSVs are vertical interconnects going through the chip silicon substrate filled with a conducting material. They enable short interconnections in 3D SICs and are used to provide power, clock and signal lines. Besides that, a limited number of TSVs are reserved to provide test access to logic blocks on different layers. Number of instances of 3D integration have been reported (IBM [7], IMEC [8], MIT [9]). Introduction of 3D architectures are also offering us several advantages, such as (i) higher packing density,(ii) higher performance (iii) lower interconnect power consumption (iv) support for realization of mixed technology chip, etc [2], [10], [11].

While designing 3D ICs, all the advantages possessed by it must be translated into cost effectiveness. Testing is a major component of the total production cost of any IC design. Thus, optimization in test architecture along with test time is the goal of the researchers to ultimately reduce the cost of an IC. Two important components of test infrastructure in a core based SOC are test wrapper and test access mechanism (TAM). The test wrapper establishes the link between the different SOCs and between a SOC and TAM [12]. TAM is a special kind of test infrastructure that transports test data to the SOCs. While IEEE 1500 standard has addressed the issue of wrapper design, the TAM optimization is left to system integrators [13]. The problem of optimization of test access architecture for a SOC is proved to be NP-hard [14] in [15]. In comparison to testing of 2D ICs, 3D SICs introduce many new challenges for testing. The testing of 3D ICs has been considered as the number one challenge in research in 2009 [16]. Several researchers are still concentrating to make a breakthrough in this area [17]–[27]. In 3D design TAM allows transferring of test data from input/output pins to different dies.

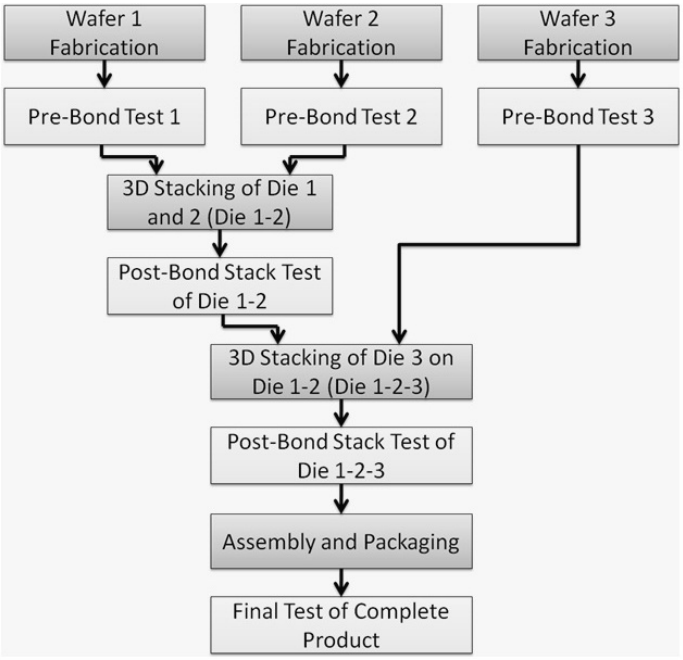

Fig. 1: 3D-SIC manufacturing and test flow with multiple test insertion.

While two-dimensional ICs typically require two test insertions, namely wafer test and package test, when it comes to 3-D stacking, a number of natural test insertions are intro-

978-1-4673-6484-3/13/$31.00 ©2013 IEEE

duced [28]. Testing multiple subsequent (partial) stacks during assembly becomes necessary due to the defects that may be introduced during the die-stacking steps of thinning, alignment and bonding. Figure 1 shows the example of manufacturing and test flow for 3D stack [29]. Initially, wafer or pre-bond test is done to test each die prior to stacking. Next, Die 1 and Die 2 are stacked, and then tested again. This is also the first time the TSV between Die 1 and Die 2 are tested. Defects in the stack due to additional 3D manufacturing steps such as alignment and bonding can also be tested in this step. The third die is added to the stack and all dies in the stack, including the TSV connections are retested. Finally, the "known good stack" is packaged and the final product is tested. Optimization methods are needed to minimize test time not only for the final stack test, but also to minimize the total test time of the final stack and the partial stacks that are tested during bonding.

This paper deals with finding the sequences in which the die should be tested such the the total time of the partial and final time is minimized.

The organization of this paper is as follows. Section II gives a motivational example of the problem. Section III states the problem. Section IV, V and VI provide the algorithm, its time complexity and an illustrative example respectively. Section VII depicts the experimental results and section VIII reports the conclusion.

II. Motivational Example

Die	d695	f2126	p22810	p93791	p34392
	1	2	3	4	5
Test Length	106391	700665	1333098	2608870	2743317
Test Pins	10	20	25	30	25

TABLE I: Test lengths and number of test pins for hard dies used in optimisation.

The test times and TAM available for the five benchmark SOCs are given in Table I. Consider two sequences of these SOCs, SIC 1 and SIC 2, given in Figure 2. In SIC 1, the SOCs are arranged in decreasing order of their test time i.e., the SOC having the highest test time is stacked and tested first. In SIC 2, the SOCs are tested in reverse order, i.e., the SOC with lowest test time is tested first. A scheduling algorithm given in [30] minimizes the test time when either complete stack or complete stack and partial stacks are tested. Following this algorithm, Table II shows the scheduling for the two SICs when a TAM width 40 is available. Here "$\|$" refers to parallel and "," refers to series scheduling. For example 1$\|$(2,3) means SOCs 2 and 3 are scheduled in series in one layer and SOC 1 in parallel with the two in another layer. The total test time of SIC 1 is 31603976 while that for SIC 2 is 14763011. Clearly, SIC 2 is much better option for scheduling than SIC 1. From the table we can infer that the sequence in which the SOCs are tested affects the total test time. So, for a given available TAM width, there should exist an optimal sequence in which the SOCs can be tested in order to get the minimum test time.

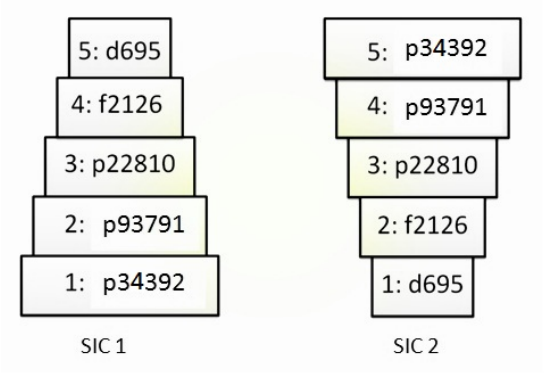

Fig. 2: Two possible 3D SIC sequence.

III. Problem Statement

Given a set of SOCs, their corresponding test times and TAM width available, the way in which the order of stacking the SOCs affects the total test time is analyzed. The goal is to find the order in which the SOCs should be stacked such that the total of partial and complete stack test times is minimum. Also, the partial and final schedule is done within the constraint of maximum TAM width available.

IV. Proposed Algorithm

The algorithm for generating the stacking sequence of SOCs in a SIC for testing so as to reduce the total test time of partial stack tests and the complete stack test is described below. It gives the sequence in which the SOCs should be stacked. It uses a subroutine called Schedule() which takes as input a set of SOCs that are chosen for stacking and finds their optimal scheduling and return the minimum time required for the scheduling. If all the given SOCs can be scheduled in parallel, the function Schedule() does so and returns the maximum test time among all the SOCs. Otherwise, some of the SOCs are scheduled in series in some of the layers, and the test time of the schedule is returned. The algorithm is as follows:

Input:

1) A set {N} of n SOCs with their test times T_i (test time of the i^{th} SOC) and TAM width TAM_i (TAM width of i^{th} SOC, SOC_i)
2) The maximum TAM width available TAM_{max}

Output:

1) The sequence in which the SOCs should be stacked
2) The total test time

begin

1) Initially all the SOCs in the set N are unstacked
2) Sort the SOCs such that $T_i < T_{i+1}$
3) Stack SOC_1, stack SOC_2
4) {N} $\leftarrow$ {N} - {SOC_1, SOC_2}
5) t $\leftarrow$ Schedule ({SOC_1, SOC_2})
6) For each unstacked SOC, SOC_k in {N}
   Compute $t_k \leftarrow$ Schedule(set of stacked SOCs + SOC_k)

SIC	schedule 1	test length	schedule 2	test length	schedule 3	test length	schedule 4	test length	total time
SIC 1	1,2	6550838	1,2,3	7883936	1,2,3,4	8584601	(1‖2),3,4,5	8584601	31603976
SIC 2	1‖2	700665	1‖2,3	2033763	1‖2,3,4	4642633	1‖2,3,4,5	7385950	14763011

TABLE II: Scheduling for SIC 1 and SIC 2 for TAM width of 40

7) Choose an SOC_m for which the test time t_m is minimum among test times (t_k), computed in the above step
8) Stack SOC_m
9) {N} $\leftarrow$ {N} - SOC_m
10) t $\leftarrow t + t_m$
11) Repeat steps 6 to 10 until all the SOCs are stacked
12) return t

end

Schedule (Set of SOCs : {S})

Input:

1) A set of SOCs {S}, with their test times T_i (test time of the i^{th} SOC, SOC_i) and TAM width, TAM_i (TAM width of SOC_i)
2) The maximum TAM width available TAM_{max}

Output: The total test time

begin

/* Let {Q} be a set of SOCs which can be scheduled in parallel. Let Time[i] holds the time when the test of i^{th} SOC is complete. Initially {Q} $\leftarrow \phi$*/

1) $\tau \leftarrow 0$, {P} $\leftarrow \phi$ /* {P} is a set of SOCs*/
2) $SOC_{max} \leftarrow$ an SOC from the set {S}, having the highest test time
3) *while* TAM width of $SOC_{max} < TAM_{max}$ *do*
   a) {Q} $\leftarrow$ {Q} + SOC_{max}
   b) $TAM_{max} \leftarrow TAM_{max}$ - TAM width of SOC_{max}
   c) {S} $\leftarrow$ {S} - SOC_{max}
   d) $SOC_{max} \leftarrow$ an SOC from the set {S}, having the highest test time

   end while
4) {P} $\leftarrow$ {P} + {Q}
5) For all $SOC_i \epsilon$ {Q}, Time[i] $\leftarrow \tau + T_i$,
6) Choose $SOC_j \epsilon$ P with Time[j] = min (Time[i], for each $SOC_i \epsilon$ {P})
7) $\{P\} \leftarrow \{P\}$ - SOC_j
8) $\tau \leftarrow \tau + Time[j]$
9) $TAM_{max} \leftarrow TAM_{max}$ + TAM_j
10) {Q} $\leftarrow \phi$
11) Repeat steps 2 to 10 until S = ϕ
12) Return maximum value of Time[i] among all the SOCs

end

Justification of the algorithm

The test time of the core stacked first is cumulated in the test times of all the partials test times after each stacking. The SOC stacked first affects the test time mostly. Due to this, we first stack the SOCs in the increasing order of their test times.

But, during stacking if the TAM requirement of an SOC is less, there is a possibility of the SOC being able to be scheduled in parallel with other SOCs and hence can reduce the test time further. In that case we may not have to take the available SOC with minimum test time. Now, after each new SOC is stacked, the partial testing of the SOCs is re-scheduled. Our aim is to choose the next SOC for stacking such that the partial test time becomes minimum . For each of unstacked SOCs, we use the Schedule() function to schedule it with the stacked cores and find which of them gives the minimum test time. The one giving the minimum test time is chosen for stacking.

In the Schedule() subroutine, first we schedule the SOCs having higher test time in parallel (while loop of step 3) until TAM_{max} is less than the TAM requirement of next unscheduled SOC having highest test time. As soon as a particular scheduled SOC (step 6) releases its TAM (step 9), the next SOC(s) is scheduled in series with it (step 3). If the available TAM, TAM_{max} is not sufficient to schedule the next SOC, it waits for the next iteration when the next scheduled SOC completes its testing and releases its TAM.

V. Time Complexity

Let the number of SOCs be n.

Complexity of Schedule() subroutine : Steps 3, 5 and 6 require $O(n)$ computations. Rest of the steps require constant time. Steps 2 to 10 are repeated for few iterations independent of n.

Complexity of algorithm: Sorting in step 2 requires $O(nlog(n))$ computations. In step 5, the Schedule() subroutine is called with two cores so time is $O(2)$. In step 6, Schedule() is called with r SOCs for $(n-(r-1))$ times and the step is repeated for $r \leftarrow 3$ to n. So step 6 is repeated $(n-2)$ times. Other steps are trivial with constant time. So the total test time is :

$O(nlog(n)) + 2 + 3(n-2) + 4(n-3) + ... + (n-1)(n-(n-2)) + n(n-(n-1))$

$= O(nlog(n)) + (2+3.n+4.n+....+n.n) - (3.2+4.3+5.4+...+(n-1)(n-2)+(n)(n-1) = O(nlog(n)) + O(n^3) - O(n^2) = O(n^3)$

VI. Illustrative Example

Consider the set of SOCs given in Table I. For illustration, let us implement our algorithm for TAM width 50.

Firstly, SOCs are already sorted with respect to their test times in Table I. Table III shows the result of various steps of the algorithm. Column 1 represents the SOCs that are inputs of the Schedule() subroutine during its successive calls. Columns 2 and 3 are the results of the Schedule() subroutine. Column 2 represents the schedule. Column 3 represents the test time calculated by the Schedule() subroutine. Column 4 represents the partial test times which is taken as the minimum test time

of all the possible stacking possible. Column 5 represents the stacking sequence and Column 6 represents the total test time.

So, initially, the first two cores are stacked, the Schedule() function gives the sequence of the test and the partial test time 700665 (Columns 2 and 3). Thereafter, each of the cores 3, 4 and 5 is rescheduled with the first two cores and the test times for each case is found. As given in Table III, we see that core 3 gives the minimum test time, so core 3 is stacked next with the partial test time 1333098. Next we see that core 5 gives the minimum test time among cores 4 and 5 when scheduled with 1, 2 and 3. This is because as shown in Column 2, core 5 can be scheduled in parallel with cores 1,2 and 3. The partial test time is becoming 2743317 after this step. The TAM requirement of core 4 is 30 so it can not be scheduled in parallel with core 3. Finally the last core 4 is stacked and the complete stack test time is calculated to be 5352187. The total test time is calculated as the sum of partial test times and test time of complete stack as 10129267.

The SIC for TAM width 50 is shown in Figure 3. As we can see, for TAM width 50, the stacking sequence 1-2-3-5-4 is not SIC 1 OR SIC 2 but a different sequence, SIC 3, as shown in the figure.

SOCs	Schedule	Test Time	Partial Test Time	Stacking Sequence	Total Test Time
{1,2}	1‖2	700665	700665	1,2	
{1,2,3}	3‖(1,2)	1333098			
{1,2,4}	4‖(1,2)	2608870	1333098	3	
{1,2,5}	5‖(1,2)	2743317			10129267
{1,2,3,4}	(2,1)‖(4,3)	3941968	2743317	5	
{1,2,3,5}	5‖(1,2,3)	2743317			
{1,2,3,4,5}	5,4‖(1,2,3)	5352187	5352187	4	

TABLE III: Implementation of the algorithm for TAM width 50

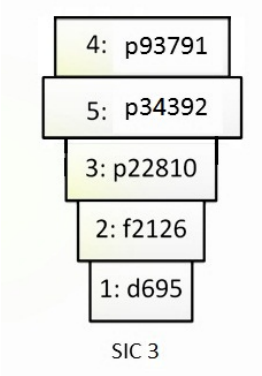

Fig. 3: The best stacking of the SOCs for TAM width 50 to achieve the minimum test time.

VII. Experimental Results

Given five SOCs as mentioned in Table I, we run our algorithm to find the stacking sequence of the SOCs so as to obtain the minimum test time. For our experiment, we have used Intel SOC Duo processor having 504 MB RAM. Programs are written in C++ programming language. We use five representative SOCs from ITC'02 SOC test benchmarks, namely d695, f2126, p22810, p34392 and p93791 [29]. Table I shows the test times(cycle) and TAM widths for different dies.

Table IV shows the overall experimental results for five SOCs for different TAM widths. Column 1 represents the TAM width. Columns 2,4 and 6 represent the partial schedules. Columns 3, 5 and 7 represent the corresponding partial test time. Columns 8 and 9 represent the complete schedule and complete test time respectively. Columns 10 and 11 represent the actual output of the algorithm, the stacking sequence and the total test time respectively. In Column 10 we can observe that, in most cases the stacking sequence is 1-2-3-4-5. But in the case of TAM widths 50 and 54, the stacking sequence becomes 1-2-3-5-4. We can see that given a TAM width, the sequence for stacking the SOCs is fixed to achieve the minimum post bond test time. The table shows the different arrangement of such stacking for different TAM widths.

VIII. Conclusion

We have analyzed how the sum of test times of the partial and complete stacks of SOCs during post bond test varies with the sequence in which the SOCs are stacked. We have presented an algorithm for finding the optimal sequence in which the SOCs should be stacked so that the total test time is minimum.

IX. Acknowledgement

The work was supported in part by UGC support for the Major Research Project (ref. no. 41-620/2012 (SR)).

References

[1] Liddle and E. David, "The wider impact of Moore's law," *IEEE Solid State Circuit Newsletter*, vol. 11, pp. 28–30, 2006.

[2] G. Loh, Y. Xie, and B. Black, "Processor design in 3d die stacking technologies," *IEEE Micro*, vol. 27, no. 3, pp. 31–48, 2007.

[3] Y. Xie, G. H. Loh, and K. Bernstein, "Design space exploration for 3D architecture," *J Emerg Technol Comput Syst*, vol. 2, no. 2, pp. 65–103, 2006.

[4] K. Banerjee et. al., "3D ICs: A novel chip design for improving deep sub-micrometer interconnect performance and system-on-chip integration," *Proc. IEEE*, vol. 89, pp. 602–633, 2011.

[5] W. R. Davis et. al., "Demystifying 3D ICs: The Pros and Cons of Going Vertical," *IEEE Design & Test on Computers*, vol. 22, no. 6, pp. 498–510, 2005.

[6] C. C. Liu, I. Ganusov, M. Burtscher, and S. Tiwari, "Bridging the Processor-Memory Performance Gap with 3D IC Technology," *IEEE Design & Test of Computers*, vol. 22, no. 6, pp. 556–564, November/December 2005.

[7] A. Topol et al., "Enabling SOI based assembly technology for three dimensional 3-D integrated circuits (ICs)," in *Proc. IEDM*, 2005, p. 363.

[8] B. Swinnen et al., "3D integration by Cu-Cu thermo-compression bonding of extremely thinned bulk Si die containing 10 m pitch through Si vias," in *Proc. IEDM*, December 2006, pp. 11–13.

[9] R. Reif et al., "Fabrication technologies for three-dimensional integrated circuits," in *Proc. ISQED*, 2002, p. 33.

[10] X. Wu, Y. Chen, K. Chakrabarty, and Y. Xie, "Test-access mechanism optimization for core-based three-dimensional socs," *Microelectronics Journal*, vol. 41, pp. 601–615, 2010.

TABLE IV: Stacking arrangement of the SICs to achieve the minimum test time.

TAM_{max}	schedule 1	test length	schedule 2	test length	schedule 3	test length	schedule 4	test length	Stacking Sequence	total time
30	1,2	807056	1,2,3	2140154	1,2,3,4	4749024	1,2,3,4,5	7492341	1-2-3-4-5	15188575
34	1,2	807056	1,2,3	2140154	1,2,3,4	4749024	1,2,3,4,5	7492341	1-2-3-4-5	15188575
35	1‖2	700665	1‖2,3	2033763	1‖2,3,4	4642633	1‖2,3,4,5	7385950	1-2-3-4-5	14763011
39	1‖2	700665	1‖2,3	2033763	1‖2,3,4	4642633	1‖2,3,4,5	7385950	1-2-3-4-5	14763011
40	1‖2	700665	1‖2,3	2033763	1‖2,3,4	4642633	1‖2,3,4,5	7385950	1-2-3-4-5	14763011
44	1‖2	700665	1‖2,3	2033763	1‖2,3,4	4642633	1‖2,3,4,5	7385950	1-2-3-4-5	14763011
45	1‖2	700665	3‖(1,2)	1333098	3‖(1,2),4	3941968	3‖(1,2),4,5	6685285	1-2-3-4-5	12661016
49	1‖2	700665	3‖(1,2)	1333098	1‖4,2‖3	3941968	1‖4,2‖3,5	6685285	1-2-3-4-5	12661016
50	1‖2	700665	3‖(1,2)	1333098	5‖(1,2,3)	2743317	5,4‖(1,2,3)	5352187	1-2-3-5-4	10129267
54	1‖2	700665	3‖(1,2)	1333098	5‖(1,2,3)	2743317	5,4‖(1,2,3)	5352187	1-2-3-5-4	10129267
55	1‖2	700665	3‖(1,2)	1333098	4‖(1,2,3)	2608870	(3‖4),(1‖2‖5)	3941968	1-2-3-4-5	8584601
69	1‖2	700665	1‖2‖3	1333098	4‖(1,2,3)	2608870	(3‖4),(1‖2‖5)	3941968	1-2-3-4-5	8584601
70	1‖2	700665	1‖2‖3	1333098	4‖(1,2,3)	2608870	(3‖4),(1‖2‖5)	3941968	1-2-3-4-5	8584601

[11] K. Puttaswamy and G. H. Loh, "3D-Integrated SRAM Components for High-Performance Microprocessors," *IEEE Transactions on Computers*, vol. 58, no. 10, pp. 1369–1381, 2009.

[12] Y. Zorian, E. J. Marinissen, and S. Dey, "Testing embedded-core-based system chips," vol. 32, no. 6, 1999.

[13] "IEEE Std. 1500: IEEE Standard Testability Method for Embedded Core-based Integrated Circuits," 2005.

[14] M. R. Gary and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman, 1979.

[15] K. Chakrabarty, "Optimal Test Access Architectures for System-on-a-Chip," *ACM Transactions on Design Automation of Electronic Systems*, vol. 6, pp. 26–49, Jan 2001.

[16] H. Lee and K. Chakrabarty, "Test Challenges for 3D Integrated Circuits," *IEEE Design & Test of Computers*, vol. 26, no. 5, pp. 26–35, September/October 2009.

[17] L. Jiang, Q. Xu, and B. Eklow, "On Effective TSV Repair for 3D-Stacked ICs," in *Proc. Design, Automation & Test in Europe (DATE)*, 2012, pp. 793–798.

[18] M. Ritcher and K. Chakrabarty, "Test Pin Count Reduction for NoC-based Test Delivery in Multicore SOCs," in *Proc. Design, Automation & Test in Europe (DATE)*, 2012, pp. 787–792.

[19] S. Hamdioui and M. Taouil, "Yield Improvement and Test Cost Optimization for 3D Stacked ICs," in *Proc. Asian Test Symposium*, 2011, pp. 480–485.

[20] B. Noia and K. Chakrabarty, "Testing and Design-for-Testability Techniques for 3D Integrated Circuits," in *Proc. Asian Test Symposium*, 2011, pp. 474–479.

[21] C. Pai, R. Cu, B. Cheng, L. Chen, and K. Li, "A Unified Interconnects Testing Scheme for 3D Integrated Circuits," in *Proc. Asian Test Symposium*, 2011, pp. 195–200.

[22] B. Noia and K. Chakrabarty, "Identification of Defective TSVs in Pre-Bond Testing of 3D ICs," in *Proc. Asian Test Symposium*, 2011, pp. 187–194.

[23] Y. Cheng, L. Zhang, Y. Han, J. Liu, and X. Li, "Wrapper Chain Design for Testing TSVs Minimization in Circuit-Partitioned 3D SoC," in *Proc. Asian Test Symposium*, 2011, pp. 181–186.

[24] E. J. Marinissen, C. Chi, M. Konijnenburg, and J. Verbree, "A Dft Architecture for 3D-SICs Based on a Standard Die Wrapper," *Jounal of Electronic Testing Theory and Applications*, vol. 28, no. 1, pp. 73–92, 2012.

[25] B. Noia, K. Chakrabarty, and E. J. Marinissen, "Optimization Methods for Post-Bond Testing of 3D Stacked ICs," *Jounal of Electronic Testing Theory and Applications*, vol. 28, no. 1, pp. 103–120, 2012.

[26] B. SenGupta, U. Ingelsson, and E. Larsson, "Scheduling Tests for 3D Stacked Chips under Power Constraints," *Jounal of Electronic Testing Theory and Applications*, vol. 28, no. 1, pp. 121–135, 2012.

[27] M. Buttrick and S. Kundu, "On Testing Prebond Die with Incomplete Clock Networks in a 3D IC Using DLLs," *Jounal of Electronic Testing Theory and Applications*, vol. 28, no. 1, pp. 93–101, 2012.

[28] E. Marinissen and Y. Zorian, "Testing technologies for three-dimensional integrated circuits," in *Proc. International Test Conference*, vol. E 1.1, 2009.

[29] B. Noia, K. Chakrabarty, and E. Marinissen, "Optimization methods for post-bond die-internal/external testing in 3d stacked ics," in *Proc. International Test Conference*, 2010, pp. 1–9.

[30] S. K. Roy, C. Giri, and H. Rahaman, "Optimization of Test Architecture in 3D Stacked ICs for Partial Stack/Complete Stack using Hard SOCs," in *Proc. International Design and Test Symposium, Doha*, December 2012.