Unified 3D Test Architecture for Variable Test Data Bandwidth Across Pre-Bond, Partial Stack, and Post-Bond Test

Yu-Wei Lee and Nur A. Touba

Computer Engineering Research Center
Department of Electrical and Computer Engineering
University of Texas, Austin, TX 78712-1084
Email: ywlee@utexas.edu, touba@ece.utexas.edu

***Abstract*—Conventional approaches for test architecture optimization are based on designing test access mechanisms (TAMs) and core wrappers for a particular test data bandwidth available from the tester. However, constructing three dimensional integrated circuits (3D-ICs) using known-good dies (KGD) and known-good stacks (KGS) requires pre-bond testing of die and optionally partial stack testing in addition to the final post-bond test. In each of these different test periods: pre-bond, partial stack, and final test, the test data bandwidth available for a particular die may be different. A test architecture optimized for one particular test data bandwidth may be very inefficient when the bandwidth changes. Previously proposed test optimization techniques for handling this involve designing different TAM architectures for pre-bond and post-bond test in order to minimize the test time for each different test data bandwidth. This paper describes an approach for designing a single TAM architecture with a "bandwidth adapter" on each die that can be used efficiently for multiple test data bandwidths. Experimental results are presented which show that this approach allows efficient test in all phases from pre-bond, multiple partial stack configurations, and post-bond.**

Keywords: -test access mechanism, test scheduling, 3D test

1. Introduction

Three-dimensional integrated circuits (3D-IC) using through-silicon vias (TSVs) are an important new technology that provide a number of significant advantages including increased functional density, shorter interconnect, higher performance, and lower power. Stacking in a 3D-IC can be done wafer-to-wafer (W2W), die-to-wafer (D2W), or die-to-die (D2D). W2W allows higher manufacturing throughput, but achieving a good compound yield is difficult. In D2W and D2D, pre-bond testing can be used to screen out defective die and use only *known-good die (KGD)* when constructing the stack. Furthermore, during construction of the stack, additional *known-good stack (KGS)* testing can be done after each die is added to the stack which adds additional test cost, but has been shown to payoff in reducing overall costs by avoid additional processing in some cases [Taouil 10].

In order to do pre-bond testing on the non-bottom layers, it is necessary to add probe pads for test purposes. This is because the TSV tips as well as the microbumps are too small to be probed and are sensitive to scrub marks [Marinissen 09]. These probe pads are a test overhead that take up a lot of space and limit the locations where TSVs can be placed which puts constraints on the design and floorplan of a die. Minimizing the probe pads is very important. In post-bond test, the bottom layer is accessed through the normal functional pins, and test data is transported to the upper layers via test elevator TSVs.

In 2D testing, test access mechanisms (TAMs) are used to deliver test data to core wrappers which interface with the scan chains in the cores. Procedures for optimizing the test architecture (i.e., selecting the width of TAMs, which cores are connected to each TAM, and optimizing the test wrappers) to minimize test time and satisfy power constraints have been well studied [Iyengar 02], [Goel 02], [Xu 05], [Larsson 05]. These algorithms are based on having a fixed test data bandwidth available from the tester. Based on this bandwidth they optimize the test architecture to minimize test time.

For 3D testing, procedures for optimizing the test architecture for post-bond test under a constraint on the number of TSVs used as test elevators to bring test data from the bottom layer to the non-bottom layers has been proposed in [Wu 08] and [Noia 10a]. In [Noia 10b], an optimization procedure is described for generating a test schedule for each stage of KGS testing (as each die is added to the stack) followed by the final test.

In pre-bond test, the test data bandwidth available from the tester for non-bottom dies is severely constrained because of the need to minimize the number of probe pads and thus will likely be different from the test data bandwidth used in post-bond test. This creates a challenge for optimizing the test architecture to minimize the overall test time corresponding to the sum of the test time for pre-bond test of each die plus the time for post-bond test of the final stack. In [Jiang 09], a test optimization procedure is proposed for this problem, but it assumed that the test elevators to the non-bottom layers could be probed in pre-bond test, however this would require a large number of probe pads to accomplish. In [Jiang 12], the problem is addressed by designing two TAM architectures, one optimized for the test data bandwidth available for pre-bond test and the other optimized for the test data bandwidth available for post-bond test. The key idea is to try to share the

978-1-4799-1585-9/13/$31.00 ©2013 IEEE

test wires in pre-bond and post-bond test as much as possible to minimize overhead.

In this paper, a methodology is proposed for designing a single TAM architecture on each die with a "bandwidth adapter" that allows it to be efficiently used for multiple different test data bandwidths. In this way, a single test architecture can be re-used for pre-bond, partial stack, and post-bond testing while minimizing test time across all phases of test. Unlike previous approaches, this methodology does not need multiple TAM architectures or reconfigurable wrappers in order to be efficient when the test data bandwidth changes. This helps to simplify the test architecture and minimize routing and overhead costs.

The paper is organized as follows: Section 2 describes the idea of using a bandwidth adapter to deal with the problem of variable bandwidth across test stages. Section 3 formulates the problem of selecting the optimal bandwidth for each stage of test as a dynamic programming problem.. Section 4 presents experimental results for the proposed scheme. Section 5 is a conclusion.

2. Proposed Scheme

The idea in the proposed scheme is to optimize the test architecture on a particular layer for the maximum test data bandwidth, n, that the layer will receive during any phase of test (pre-bond, partial stack, or post-bond). Then for any phase of test where the bandwidth to the layer is less than n, a bandwidth adapter is used to handle the bandwidth mismatch.

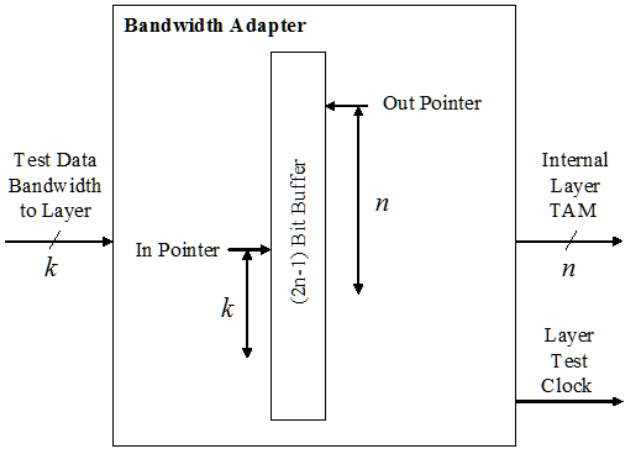

Figure 1. Input Bandwidth Adapter

An input bandwidth adapter is shown in Fig. 1. It takes as an input k bits of test data each clock cycle where k is less than or equal to n. It stores the k-bits received each clock cycle in a $2n-1$ bit buffer at the location pointed to by the *in_pointer* and then increments the pointer by k mod $2n-1$. When the difference between the *in_pointer* and *out_pointer* indicates that n or more bits are ready in the buffer, then the bandwidth adapter outputs the n bits pointed to by the *out_pointer* and increments the *out_pointer* by n mod $2n-1$. When the n bits are outputted, a pulse is generated on the *layer_test_clock* so that the internal layer TAM will transport the data. The bandwidth adapter can be designed in a general fashion to handle arbitrarily different test data bandwidths. Note that it is not necessary for n to be a multiple of k. Any value of k can be handled.

By the same token, an output bandwidth adapter can be designed to handle the output response. It is the inverse of the input bandwidth adapater. It takes as an input the n bits of the internal layer TAM and the *layer_test_clock*, and it outputs k bit test data every clock cycle.

The proposed bandwidth adapter is totally independent to the underlying test architecture and thus is not limited to scan-based design. Note that some previous research has implemented the idea of variable bandwidth using reconfigurable wrappers which connect multiple scan chains to allow a smaller bandwidth to fill the scan chains [Khoche 01], [Koranne 03]. Serial/parallel conversion has also been used in [Noia 11] in the context of optimizing firm dies in post-bond test.

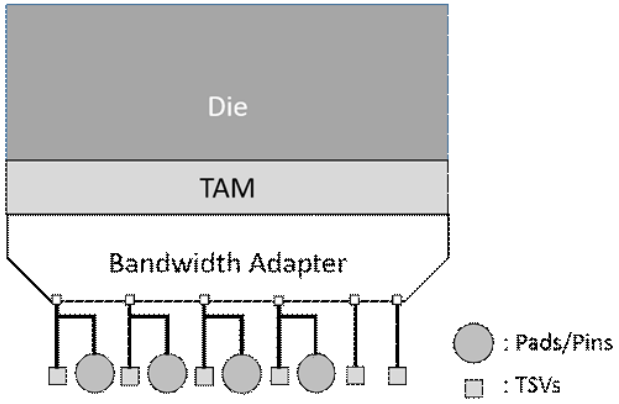

Figure 2. Block Diagram for Using Bandwidth Adapter

Figure 2 illustrates using a bandwidth adapter to unify the test architecture on a die for both pre-bond and post-bond test. The incoming bandwidth for pre-bond and post-bond tests are different because during pre-bond test, a smaller number of probe pads are used so the tester can directly probe the die, whereas during post-bond test, a larger number of test elevator TSVs are used to bring test data up from the pins in the bottom layer which are connected to the tester. The test architecture is optimized for the larger post-bond test data bandwidth, and then during pre-bond test, when the dies are tested separately, test vectors are brought in through the probe pads to the input bandwidth adapter which allows it to drive the larger number of TAM lines, and the output TAM lines go through an output bandwidth adapter to drive a smaller number of output probe pads.

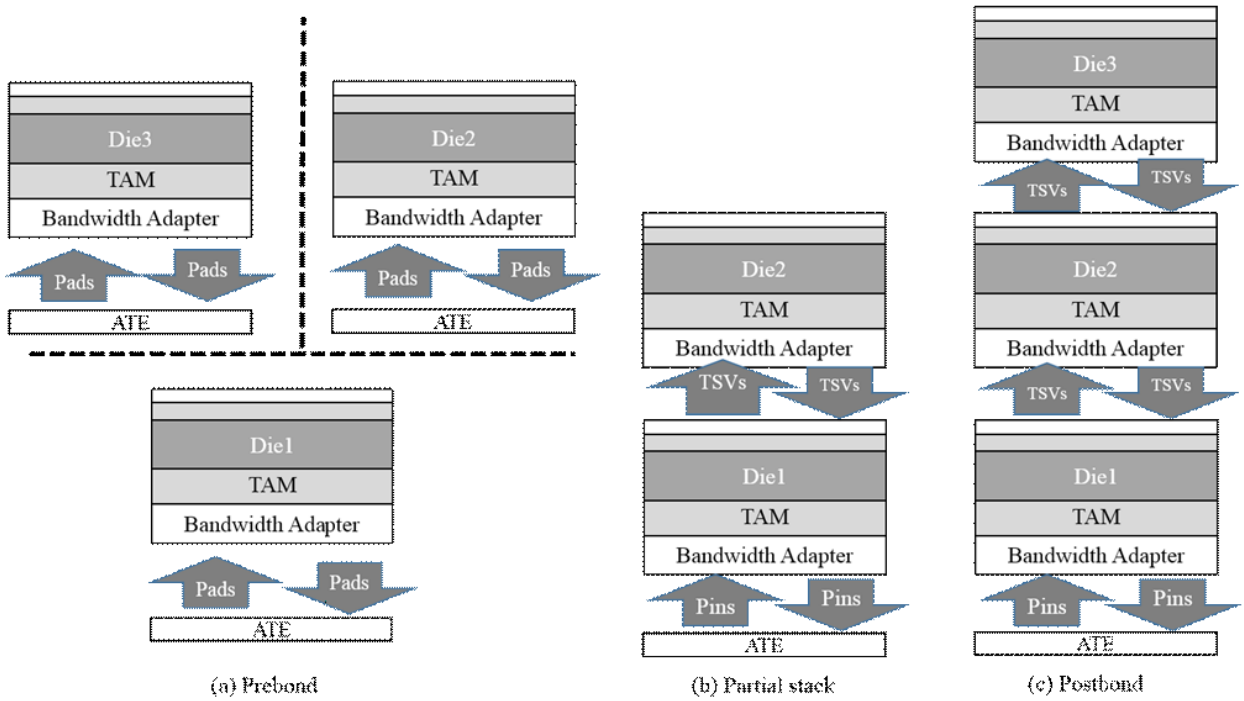

Figure 3. Unified TAM, Bandwidth Adapter and the Corresponding Test Flow

Figure 3 illustrates all the different phases of test using the bandwidth adapter. As mentioned previously, partial stack test may be optional. With the help of a bandwidth adapter, the insertion of a partial stack utilizes the fact that only a subset of dies from the final stack test is tested in this stage. With the bandwidth adapter, the underlying test architecture does not need to be redesigned for this additional test. That is, the partial stack test can be viewed as an intermediate stage of a post-bond test. The dies in these intermediate stages can use any bandwidth different from the final stack test within the limitation on the number of test elevators and test pins.

3. Selecting Bandwidth for Each Stage of Test

This section describes how the test bandwidth is allocated at each stage of test to minimize the overall test time under the assumption of a unified test architecture with the proposed bandwidth adapter. There are two different scenarios considered in this section. The first scenario is only a pre-bond and post-bond test. The second scenario also includes partial stack tests, which can be viewed as a subset of the final stack test.

If only pre-bond and post-bond test are performed, the bandwidth used for pre-bond test of each die is equal to whatever maximum bandwidth is available from the tester to that particular die. That will depend on how many probe pads are available on the die. For post-bond test, the layers are tested in parallel under the constraint that the total amount of bandwidth for all layers is equal to the bandwidth between the tester and the pins of the bottom layer which is then distributed to the non-bottom layers via test elevator TSVs. The final post-bond test time is the largest test time among all layers in the design. The test data bandwidth that should be allocated to each layer to minimize the total test time needs to be selected.

The problem of determining the optimal post-bond bandwidth allocation to each layer can be formulated and solved with dynamic programming. The dynamic programming formulation can be written as follows:

$$\begin{cases} T(D_{i\ldots k}, B) = \min\{\max[time_i(b_i), T(D_{i+1\ldots k}, B - b_i)]\} \\ \qquad\qquad if\ i < k \\ T(D_{i\ldots k}, B) = time_i(B) \\ \qquad\qquad if\ i = k \end{cases}$$

Where $D_{i\ldots k}$ represents set of dies from i through k, $T(D_{i\ldots k}, B)$ means the test time for dies from i through k with bandwidth B, b_i must be greater than 0 and represents the bandwidth allocated to core i, and $time_i(B)$ represents the test time for *die* i with bandwidth B.

The dynamic programming formulation above suggests that the search for an optimal bandwidth allocation can be divided into recursive subproblems. In addition, instead of examining all possible combinations exhaustively, the algorithm efficiently stores the results of overlapping subproblems into a table to avoid repeating unnecessary computation.

Function $time_i(B)$ is a design-dependent function which queries what the test time for a layer would be using a given bandwidth B. To get this number, one of the many 2D algorithms for optimizing the test architecture for a given test

data bandwidth can be used. For example, the TR-Architect algorithm [Goel 03] could be used as was done in the experiments reported in Sec. 4, but there are many others in the literature [Xu 05], [Larsson 05]. The proposed approach is applicable regardless of how the test architecture is optimized on each layer.

When determining $time_i(B)$, the existance of the bandwidth adapter needs to be taken into consideration. If B is smaller than the largest bandwidth used for die i in any phase of test, which will be referred to as $B_{i,max}$ then for bandwidth B, the bandwidth adapter will be used to convert bandwidth B into the larger bandwidth $B_{i,max}$. The test time in this case will be equal to $time_i(B_{i,max})$ scaled for the delay through the bandwidth adapter:

$$time_i(B) = [time_i(B_{i,\max}) - P](\frac{B_{i,\max}}{B_i}) + P \quad when\ B < B_{i,\max}$$

where P is the number of test patterns applied by the bottleneck TAM. The capture cycles for these patterns should not be scaled, so they are subtracted out before multiplying by the scaling factor so that only the shift time is scaled, and then the capture cycles are added back in unscaled.

The following is an illustrative example of deriving the optimal bandwidth allocation for a post-bond test. There are four dies in this example, *D0, D1, D2* and *D3* to be placed at layer 0, layer 1, layer 2, and layer 3. The post-bond bandwidth is 6 in this example. Table 1 shows the relationship between bandwidth and test length for each die, namely the value of $time_i(B)$. Note that each die may come from very different designs, so the relationship between test time and bandwidth may not be consistent.

Table 1. Example of $time_i(B)$

B_i	D0	D1	D2	D3
1	40	8	20	5
2	30	7	20	5
3	20	6	10	5
4	10	5	10	5

The algorithm begins with checking $T(D_{0...3}, 6)$, which creates subproblems $T(D_{1...3}, 5)$, $T(D_{1...3}, 4)$, $T(D_{1...3}, 3)$, $T(D_{1...3}, 2)$ and $T(D_{1...3}, 1)$. An example of an overlapping subproblem is that both $T(D_{1...3}, 3)$ and $T(D_{1...3}, 2)$ produce subproblem $T(D_{2...3}, 1)$. Only necessary computation will be performed and will only be performed once. In this example, the optimal allocation is $\{b0, b1, b2, b3\} = \{3, 1, 1, 1\}$. The post-bond test time with the derived bandwidth allocation is 20.

Now consider partial stack tests. The bandwidth allocation problem can be solved in the same way as it is for final stack except that there are fewer layers. The same dynamic programming approach can be used. The key is to order the processing from smallest stack to largest stack because the bandwidth allocation for smaller stacks will be larger than for larger stacks. In this manner, $B_{i,max}$ will be determined right away and then in all subsequent steps, the use of the bandwidth adapter will be known up front and can be factored in when determining $time_i(B)$.

So the overall procedure is to first determine the bandwidth for each die during pre-bond test. This is a given based on the number of probe pads available on the die. Next, if partial stack testing is to be done, then the bandwidth allocation for the smallest partial stack is computed first (using dynamic programming) since it will use the largest bandwidths, followed by the next smallest partial stack, and so forth until finally doing the full stack. The test architecture for each die is then designed and optimized based on the maximum bandwidth, $B_{i,max}$, that it receives in any phase of test. An input and output bandwidth adapter is then added to handle all other bandwidths that the die will see in any phase of test.

4. Experimental Results

Experiments were performed on the ITC'02 benchmarks. [Marinissen 02] provides detailed descriptions of the ITC'02 benchmarks. Each 3D benchmark has four layers and each layer contains a design from the ITC'02 benchmark set. Table 2 lists the components of each 3D benchmark.

Table 2. 3D Benchmarks

Design	Layer 1	Layer 2	Layer 3	Layer 4
1	h953	d695	u226	d281
2	f2126	g1023	d695	u226
3	p93791	p34392	p22810	g1023
4	q12710	p34392	p22810	f2126
5	p93791	q12710	p34392	p22810
6	t512505	p93791	q12710	u226
7	a586710	p93791	q12710	p22810
8	a586710	t512505	p93791	p34392

To derive the test time for each design for different test data bandwidths, TR-Architect [Goel 03] was used in two scenarios. One is where the scan architecture is assumed fixed in the design as would be the case for hard cores (results shown in Table 3), and the second is where the scan architecture is assumed to be flexible as may be the case for soft cores (results shown in Table 4). The pre-bond bandwidth was assumed to be 8 in these experiments for the non-bottom layers, and the bottom layer was assumed to have a test data bandwidth of 32, 48, or 64. All possible partial stack tests are tested. The first partial stack test, which test layer 0 and layer 1 is referred to as intermediate test 1, and the second partial test is referred to as intermediate test 2. The final stack test involves testing all dies together.

In Table 3 and 4, results are shown for 3D designs 1 through 8 which are constructed with one ITC'02 benchmark circuit on each layer corresponding to what is shown in Table 2. The test time (in clock cycles) for each phase of test is shown comparing two cases. One is where a separate test architecture is designed for each different bandwidth (i.e., no bandwith adapter is used), and the other is where a single unified test architecture is designed using a bandwidth adapter.

Table 3. Experimental Result for Non-Flexible Designs

	Pre-bond Bandwidth	Post-Bond Bandwidth	Pre-bond		Intermediate Test 1		Intermediate Test 2		Final Stack		Overall		
			Separately Optimized	Unified w/Adapter	Separately Optimized	Unified w/Adapter	Separately Optimized	Unified w/Adapter	Separately Optimized	Unified w/Adapter	Separately Optimized	Unified w/Adapter	**Diff.**
1	8	32	267163	267163	132343	132343	132343	132343	132343	132343	664192	664192	100.00%
	8	48	267163	267163	132343	132343	132343	132343	132343	132343	664192	664192	100.00%
	8	64	267163	267163	132343	132343	132343	132343	132343	132343	664192	664192	100.00%
2	8	32	537887	537887	357757	357757	357757	357757	357757	357757	1611158	1611158	100.00%
	8	48	537887	537887	357757	357757	357757	357757	357757	357757	1611158	1611158	100.00%
	8	64	537887	537887	357757	357757	357757	357757	357757	357757	1611158	1611158	100.00%
3	8	32	3961050	3851545	1314769	1314769	1522806	1526525	1538204	1533345	8336829	8226184	98.67%
	8	48	3567417	3670068	921136	921136	1042640	1024827	1068036	1062649	6599229	6678680	101.20%
	8	64	3340565	3477831	695066	695066	812318	795918	833549	812143	5681498	5780958	101.75%
4	8	32	6763984	6763984	3466238	3466238	3466238	3466238	3466238	3466238	17162698	17162698	100.00%
	8	48	6763984	6763984	3466238	3466238	3466238	3466238	3466238	3466238	17162698	17162698	100.00%
	8	64	6763984	6763984	3466238	3466238	3466238	3466238	3466238	3466238	17162698	17162698	100.00%
5	8	32	9233954	9578467	3466238	3466238	3466238	3466238	3466238	3466238	19632668	19977181	101.75%
	8	48	9233954	9578467	3466238	3466238	3466238	3466238	3466238	3466238	19632668	19977181	101.75%
	8	64	9233954	9578467	3466238	3466238	3466238	3466238	3466238	3466238	19632668	19977181	101.75%
6	8	32	30388711	30388711	22973206	22973206	23162552	23699371	24703920	23699371	101228389	100760659	99.54%
	8	48	22627954	22627954	15212449	15212449	17126882	15808671	17229905	15883686	72197190	69532760	96.31%
	8	64	20351239	20351239	12935734	12935734	13024248	13486929	13024248	13486929	59335469	60260831	101.56%
Avg													100.95%

Table 4. Experimental Result for Flexible Designs

	Pre-bond Bandwidth	Post-Bond Bandwidth	Pre-bond		Intermediate Test 1		Intermediate Test 2		Final Stack		Overall		
			Separately Optimized	Unified w/Adapter	Separately Optimized	Unified w/Adapter	Separately Optimized	Unified w/Adapter	Separately Optimized	Unified w/Adapter	Separately Optimized	Unified w/Adapter	**Diff.**
1	8	32	185043	193729	58383	58383	66955	66656	68623	68512	379004	387280	102.18%
	8	48	165924	175251	39264	39264	44455	45225	47749	48264	297392	308004	103.57%
	8	64	155826	162168	29165	29165	33957	33794	35351	35554	254299	260681	102.51%
2	8	32	348445	348445	179417	179417	200231	197433	216881	200047	944974	925342	97.92%
	8	48	287954	287954	118926	118926	134129	134104	141317	137619	682326	678603	99.45%
	8	64	259052	259052	90024	90024	99066	99066	104303	102295	552445	550437	99.64%
3	8	32	3818236	3875858	1290998	1290998	1514161	1514161	1531433	1570649	8154828	8251666	101.19%
	8	48	3416782	3535277	885071	885071	1006965	1016655	1037496	1042895	6346314	6479898	102.10%
	8	64	3223401	3428306	691690	691690	777193	787404	787323	809828	5479607	5717228	104.34%
4	8	32	4348203	4416227	1227133	1227133	1443224	1443023	1633317	1578602	8651877	8664985	100.15%
	8	48	3940850	4129643	842227	842227	971816	982984	1068114	1082651	6823007	7037505	103.14%
	8	64	3753565	4028200	632495	632495	747585	746656	819780	831156	5953425	6238507	104.79%
5	8	32	7106302	7105983	1633317	1633317	2040990	2057327	2227321	2228377	13007930	13025004	100.13%
	8	48	6616992	6665707	1085050	1085050	1406248	1406248	1529900	1533397	10638190	10690402	100.49%
	8	64	6365179	6463831	833237	833237	1047365	1068134	1169749	1179100	9415530	9544302	101.37%
6	8	32	29922077	29922077	22973206	22973206	24263686	24437740	24333128	24477405	101492097	101810428	100.31%
	8	48	21973150	21973150	15024279	15024279	16249811	15620048	16320517	15938714	69567757	68556191	98.55%
	8	64	18695162	18695162	11746291	11746291	12532429	12138131	12553054	12138131	55526936	54717715	98.54%
Avg													101.13%

As can be seen from the results in Tables 3 and 4, the overall test time of the proposed approach with a unified test architecture is very close to that of having separately optimized test architectures. In fact, it is even better in a few cases, but that is likely due to variations in the efficiency of different bandwidths for a design. The result is very good because it demonstrates that a unified test architecture which is simpler to design and requires less hardware overhead is able to do almost as well as if one custom designed a test architecture for each bandwidth that a layer sees.

Note that even though the tester bandwidth is the same for all cases in pre-bond test, some cases reports different pre-bond test times. This arises due to the bandwidth adapter having to convert from a smaller tester bandwidth to a larger internal TAM bandwidth during pre-bond tests. It can be harder to efficiently schedule cores for larger TAM bandwidth. However, the test time in post-bond test is limited by the layer that takes the most time since post-bond test is performed in parallel. Therefore, the dynamic programming algorithm will try to use a larger TAM bandwidth for the bottleneck layer which may slightly increase pre-bond test time due to the inefficiency mentioned above. However, the net effect is improved overall test time because the reduction in post-bond test time is significantly greater than the increase in pre-bond test time.

5. Conclusions

The proposed approach for designing a unified test architecture across all test stages in testing 3D-ICs allows efficient hardware utilization without significant overhead in test time. It is simple to design and implement, and can be used together with any of the existing 2D test architecture optimization schemes.

Acknowledgements

This research was supported in part by the National Science Foundation under Grant No. CCF-0916837

References

[Goel 03] Goel, S.K., and E.J. Marinissen, "Effective and Efficient Test Architecture Design for SOCs," *Proc. of International Test Conference*, pp. 529-538, 2002.

[Iyengar 02] Iyengar, V., K. Chakrabarty, and E.J. Marinissen, "Test Wrapper and Test Access Mechanism Co-Optimization for System-on-a-Chip," *Journal of Electronic Test: Theory and Applications (JETTA)*, Vol. 18, p. 213-230, Apr. 2002.

[Jiang 09] Jiang, L., L. Huang, Q. Xu, "Test Architecture Design and Optimization for Three-Dimensional SoCs", *Proc. of Design Automation and Test in Europe*, 2009.

[Jiang 12] Jiang, L., Q. Xu, K. Chakrabarty, T.M. Mak, "Integrated Test-Architecture Optimization and Thermal-Aware Test Scheduling for 3-D SoCs Under Pre-Bond Test-Pin-Count Constraint," *IEEE Trans. on VLSI*, Vol. 20, No. 9, pp. 1621-1633, Sep. 2012.

[Khoche 01] Khoche, A., R. Kapur, D. Armstrong, T.W. Williams, M. Tegethoff,, and J. Rivoir, "A new methodology for improved tester utilization." *Proc. of Int. Test Conference*, pp. 916-923, 2001.

[Koranne 03] Koranne, S., "Design of Reconfigurable Access Wrappers for Embedded Core Based SoC Test," *Trans. on VLSI Systems*, Vol. 11, No. 5, pp. 955-960, Oct. 2003.

[Larsson 05] Larsson, E., *Introduction to Advanced System-on-Chip Test Design and Optimization*, Springer, ISBN 978-1402032073, 2005.

[Marinissen 00] Marinissen, E.J., S.K. Goel, and M. Lousberg, "Wrapper design for embedded core test," in *Test Conference, 2000. Proceedings. International*, 2000, pp. 911–920.

[Marinissen 02] Marinissen, E.J., V. Iyengar, and K. Chakrabarty, "A set of benchmarks for modular testing of SOCs," in *Test Conference, 2002. Proceedings. International*, 2002, pp. 519–528.

[Marinissen 09] Marinissen, E.J., and Y. Zorian, "Testing 3D Chips Containing Through-Silicon Vias," *Proc. of Int. Test Conference*, Paper ET1.1, 2009.

[Noia 10a] Noia, B., S.K. Goel, K. Chakrabarty, E.J. Marinissen, and J. Verbree, "Test-Architecture Optimizations for TSV-Based 3D Stacked ICs", *Proc. of European Test Symposium*, pp. 24-29, 2010.

[Noia 10b] Noia, B., K. Chakrabarty, and E.J. Marinissen, "Optimization Methods for Post-Bond Die-Internal/External Testing in 3D Stacked ICs", *Proc. of Int. Test Conference*, Paper 6.3, 2010.

[Noia 11] Noia, B., K. Chakrabarty, S.K. Goel, and E.J. Marinissen, "Test-Architecture Optimization and Test Scheduling for TSV-Based 3-D Stacked ICs", *IEEE Trans. on Computer-Aided Design*, Vol. 30, No. 11, Nov. 2011.

[Taouil 10] Taouil, M., S. Hamdioui, K. Beenakker, and E.J. Marinissen, "Test Cost Analysis for 3D Die-to-Wafer Stacking," *Proc. of Asian Test Symposium*, pp. 435-441, 2010.

[Wu 08] Wu, X., Y. Chen, K. Chakrabarty, and Y. Xie, "Test-access Mechanism Optimization for Core-based Three-Dimensional SOCs", *Proc. of Int. Conf. on Computer Design*, pp. 212-218, 2008.

[Xu 05] Xu, Q., N. Nicolici, "Resource-Constrained System-on-a-Chip Test: A Survey," *IEE Proc. on Computers and Digital Techniques*, Vol. 152, Iss. 1, pp. 67-81, Jan. 2005.