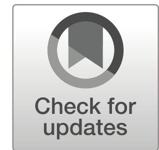
Journal of Electronic Testing (2019) 35:425–440
https://doi.org/10.1007/s10836-019-05813-z



Test Flow Selection for Stacked Integrated Circuits

Breeta SenGupta[1] · **Dimitar Nikolov**[1] · **Assmitra Dash**[2] · **Erik Larsson**[1]

Received: 22 June 2018 / Accepted: 16 May 2019 / Published online: 14 August 2019
© The Author(s) 2019

Abstract
Integrated circuits (ICs) with a single chip (die) are typically tested with a test flow consisting of two test instances: (1) wafer sort for the bare chip and (2) package test for the packaged IC. For ICs with stacked chips - 3D Stacked ICs - there are many possible test instances, even more test flows, and no commonly used test flow. In this paper, we propose a test flow selection algorithm (TFSA) to obtain a test flow for a given 3D Stacked IC. The TFSA results in a test flow for a given 3D Stacked IC, such that the expected total test time to produce each good package is minimized. We implemented the TFSA, three straightforward test flow schemes and an exhaustive search, and experimentally compared the test flow schemes on three different test architecture design approaches. The results demonstrate the importance to have methods both to select the test flow and design the test architecture.

Keywords 3D IC · Stacked integrated circuits · Test flow · Test time · Yield · Test plan · IEEE 1500 · Test architecture · Expected time · Effective yield · Quantity

1 Introduction

The constant development in semiconductor technologies enables increasingly advanced integrated circuits (ICs). Today, it is possible to manufacture wafers where each individual chip (die) contains billions of transistors. After manufacturing, the chips are first cut from the wafer and then wire bonded to connect the chip to the package. Finally, the chips are packaged. The most recent advancement in semiconductor technologies is to stack several chips on top of each other and package them in one IC − 3D Stacked IC. The chips in such a 3D Stacked IC are connected by through silicon vias (TSVs) [8].

IC manufacturing is extremely complex, which increases the risk of defects. To detect manufacturing defects, each and every IC is carefully tested. ICs with a single chip are commonly tested with a test flow consisting of two test instances; wafer sort and package test. The bare chip is tested at wafer sort to avoid packaging of defective chips. If no defects are found at wafer sort, the chip is wire-bonded, packaged, and re-tested during package test as defects may be introduced during wire-bonding and packaging. For 3D Stacked ICs there are many more test instances. It is possible to test each individual chip during wafer sort instances, at intermediate test instances where the partially complete stacks can be tested, and at package test instance where the complete 3D Stacked IC is tested. For a 3D Stacked IC with N chips there are $2N$ instances when a test may be performed, N during wafer sort, $N-1$ for intermediate stacks and 1 for package test. Hence, there are 2^{2N} possible variations of test flows [8].

The test cost of 3D Stacked ICs depend on a large number of factors, such as the hardware manufacturing cost that includes wafer fabrication, stacking and packaging, DfT (design for test), fault coverage, test resource and test equipment, the test time and yield. In this paper we reduce the test cost per good 3D Stacked IC produced, by selecting

Responsible Editor: E. J. Marinissen

✉ Erik Larsson
erik.larsson@eit.lth.se

Breeta SenGupta
breetasg@gmail.com

Dimitar Nikolov
dimitar.nikolov@altran.com

Assmitra Dash
asmitra.dash@gmail.com

[1] Lund University, Lund, Sweden

[2] Linköping University, Linköping, Sweden

a test flow optimizing two of the major contributing factors: test time and yield. Eventually, we address minimization of the expected test time for each good 3D Stacked IC produced, and implement that along with test architecture optimization as described in previous articles [12, 13].

The expected test time may vary both with the choice of test flow as well as the applied test schedule. The time spent on testing defective ICs also contributes to the test cost. Hence, the manufacturing yield needs to be taken into account. For 3D Stacked ICs, where it is possible to stack a number of different chips, it is of interest to know how many chips of different types are needed in the manufacturing process to obtain a fixed number of good packages of 3D Stacked IC.

The test cost can be reduced by improving the yield and/or by reducing the time spent on testing. Improving yield implies reducing defects in the manufacturing process, for example, by upgrading production technologies. Yield improvement is not in the scope of this paper. In this paper, we focus on minimizing the time spent on testing.

In this paper, we assume a 3D Stacked IC, where the test time and yield at each test instance is known. With the goal of minimizing the expected test time per good 3D Stacked IC package, we propose a method to compute the effective yield, the number of chips that need to be tested at each test instance, and the expected test time, depending on the selected test flow. To find the most suitable test flow, we propose the Test Flow Selection Algorithm (TFSA). We performed experiments on several 3D Stacked ICs to compare the test flow obtained using TFSA against test flows obtained by exhaustive search and three straightforward test flows – wafer sort of each individual chip followed by package test (WSPT), test at all possible instances (TA) and test performed only at package test (PT). The results demonstrate that (1) TFSA produces results that are better than the three straightforward test flow schemes, (2) TFSA produces optimal test flow in most cases, and (3) the straightforward test flow where wafer sort of each chip is followed by package test of the complete 3D Stacked IC gives the best result among the three straightforward test flow schemes. We also integrated test flow selection with test architecture design, adopting the test planning scheme proposed in [12]. While [12] optimizes the test architecture for a single test flow that consists of wafer sorts of the individual chips followed by package test of the complete 3D Stacked IC, in this paper we generalize the approach for any given test flow. The experimental results validate that it is important to have methods to find the test flow as well as methods to design the test architecture. The test flow model and TFSA complies with all die orientations – face-to-face, face-to-back and back-to-back; as well as all wafer bonding technologies – wafer-to-wafer, die-to-die and die-to wafer.

The limitations of this work are as follows. First, we assume that all test flows include package test. This is motivated by the fact that if the final test instance (package test) is not performed, all defects in the last test instance are not checked. Hence it is self-evident to assume package test to be mandatory. Second, in our experiments we calculate the test time of a intermediate stage as the sum of the test times of each individual die in the partial stack and that of the interconnects. Thus, each die would undergo the same test during wafer sort and all successive intermediate stages. The yields at all intermediate stages are also assumed to be equal during the experiments. Third, in this paper we address reduction of a part of the test cost by minimizing the test time associated with test flow selection. Optimization of all factors contributing to the test cost, like manufacturing cost or fault coverage, would lead to even higher complexity to the problem of selecting the most suitable test flow, and have therefore not considered in this paper. The expressions are however scalable to accommodate the trade-off with additional contributing factors, which is addressed in our future work.

The rest of the paper is organized as follows. In Section 2 we discuss related research. The test architecture is elaborated in Section 3. In Section 4, we illustrate with an example at three different yield sets the need of finding a suitable test flow. In Section 5, we introduce notations and formulae, while in Section 6 we present the TFSA. In Section 7 we report the results from the experiments. The paper is concluded in Section 8.

2 Related Work

Several works have addressed test planning for core-based ICs having a single chip with the aim of optimizing the test cost [2, 6, 7]. Design and optimization of test architecture for non-stacked ICs with IEEE 1500 is described in [4, 5, 11, 15]. In [5], Iyengar et al. address optimization of test access mechanisms (TAMs) for System-on-Chips (SoCs) to reduce core-test time by balancing core scan chains. Mullane et al. in [11] propose a hybrid scan for non-stacked ICs provided with IEEE 1500 core wrappers, by combining the serial and the parallel ports of the wrapper, resulting in an efficient test access that reduces the test time. However, for 3D Stacked ICs, test architecture optimized for each chip in the stack during wafer sort may not lead to an optimized test architecture when all the chips are tested jointly during package test.

We have proposed methods to reduce the test time for core-based 3D Stacked ICs, by optimizing the test architecture and the test plan [12, 13]. We used a straightforward test flow, where each individual chip is tested at wafer sort and the complete stack at package test.

The test planning approaches were not adapted for arbitrary test flows.

Taouil et al. propose test cost models to predict the impact of test flows on the product quality and overall stack cost at an early design stage, which is important for a trade-off between quality and cost [14]. They present a model that predicts the product quality, defined in terms of Defective Parts Per Million (DPPM) for different test flows. A framework is provided for covering different test flows and cost models to identify the most cost effective test flow [3]. Simulation results show that test flows that include wafer sort generally reduce the overall cost and that the most cost-effective test flow strongly depends on the stack yield. They concluded, after experiments on different test flows, that adapting the tests according to the stack yield is a good approach. The paper analyzes several test flows for given 3D Stacked ICs, but do not provide a method for the selection of the most suitable test flow. Agrawal et al. in [1] have proposed a low complexity test flow selection scheme for 3D Stacked ICs to achieve a low test cost. It is shown that the test flow selection method takes considerably lower computation time as compared to exhaustive methods that completely explore the exponentially growing search space for 3D Stacked ICs. However, an estimate of the margin of the increase in test cost by using the proposed method against exhaustive search is lacking. Both [1, 14] are built on the compromise between yield and test cost. However, the trade-off among two major contributing factors to the test cost, namely, test time and manufacturing cost of each component has been overlooked. The work is not verified against any test architecture design and test planning scheme. To the best of our knowledge, no work has previously proposed a test flow selection algorithm and verified it against any test architecture and test planning scheme.

A new test standard, IEEE P1838 is being developed to enable efficient modular testing of SICs [10]. The standard involves a die level wrapper on each chip in the stack. In addition, an IEEE 1149.1 based TAP controller is provided in the bottom die that controls the WIRs of the die wrappers. [12, 13] addresses test scheduling for SICs to minimize the test cost. In [12] a scalable test architecture is assumed where each chip is provided with a IEEE 1500 based wrapper, in accordance with the developing IEEE P1838 standard. Similarly, [13] assumes a IEEE 1149.1 based test architecture for the SICs for test planning.

3 IEEE 1500 Based Test Architecture

In this section we discuss the test architecture for a 3D Stacked IC, where each chip of the stack is supported by a IEEE 1500 based infrastructure as proposed by [8].

A number of chips are stacked to construct a 3D Stacked IC. Figure 1 shows a 3D Stacked IC with two chips, where Chip2 is stacked on top of Chip1, where each chip comprises of several cores. Chip1 contains core1, core2 and core3, while Chip2 hosts core4 and core5. To enable testing, each core consists of a number of scan-chains that are concatenated over a set of TAM lines to form wrapper-chains which are connected to the wrapper. The Wrapper Parallel Port (WPP) and the Wrapper Serial Port (WSP) constitute the test terminals for the 3D Stacked IC. The WPP includes the Wrapper Parallel Input (WPI) and the Wrapper Parallel Output (WPO), each with a width W, that is decided by the design er. The WSP, comprises of Wrapper Serial Input (WSI), Wrapper Serial Output (WSO), and Wrapper Serial Control (WSC) terminals, and supports serial test mode. The instruction to be executed in a chip is stored in the corresponding Wrapper Instruction Register (WIR).

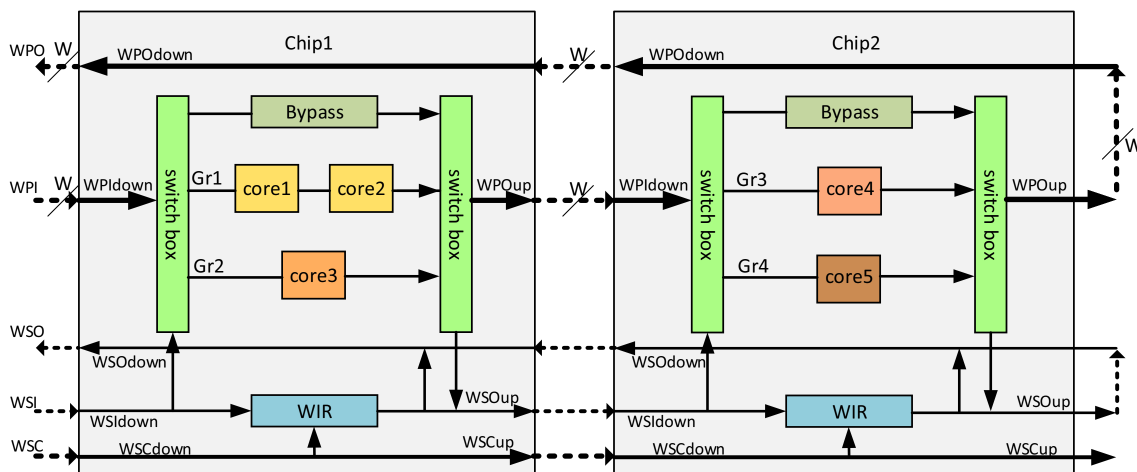


Fig. 1 Test architecture of a 3D Stacked IC, with two chips in the stack: Chip2 stacked on Chip1, where each chip is supported by the IEEE 1500 based test architecture

Within each chip, the TAM width, WPI, is split at the switch box, to several groups of TAMs. Each group of TAM is used to access one or more cores of the chip, connected in series. The TAM width W in Fig. 1, for Chip1 is split among Gr_1 connecting core1 and core2 in series, and Gr_2 to core3, while for Chip2 Gr_3 and Gr_4 accesses core4 and core5, respectively. Wrapper-chains are constructed by allocating one or more scan-chains to a single TAM line. In Fig. 2, core1 of Chip1 is illustrated, that has three scan-chains: $sc1$, $sc2$ and $sc3$. TAM group $Gr1$ is used to access core1, such that $sc1$ forms a wrapper-chain on $TAM1$, while $sc2$ and $sc3$ in series is a wrapper-chain on $TAM2$. As illustrated in Fig. 1. the WPOup, WSOup and WSCup of the lower chip, Chip1, is connected to the WPIdown, WSIdown and WSCdown of the chip on top, Chip2, respectively. The WPOup, WSOup and WSCup of the topmost chip, Chip2, are directed out via the WPOdown, WSOdown and WSCdown, respectively, of the lowermost chip in the stack, Chip1. The WPOdown, WPIdown, WSOdown, WSOdown and WSCdown of the lowermost chip, Chip1, serve as the package test interface for the 3D Stacked IC. As in [9], in this paper we assume equal width of WPIs and WPOs for each chip.

The TSV interconnect between chips may be tested using the boundary scan registers, which connects all input/output via TSVs. Boundary scan registers are implemented on both chips and are used in TSV interconnect test. Test stimuli are applied on out-going TSVs and test responses are captured on in-coming TSVs. Since the boundary scan register is a separate register, testing of TSVs cannot be performed concurrently with core tests.

The TSV interconnect tests contribute with a constant term to the overall test time and could not be scheduled with any core tests. Therefore, the time required to perform TSV interconnect tests are overseen while addressing the total test time in the remainder of the paper.

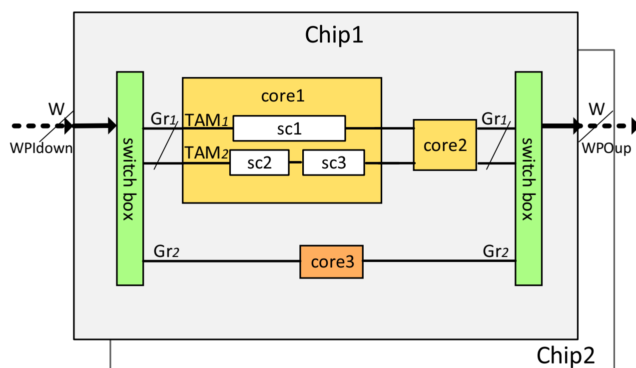


Fig. 2 Scan-chains configured into wrapper-chains in core1 of Chip1

4 Impact of Test Flow on the Expected Test Time

In this section we show with an example the effect of test flow on the expected total test time. The example considers a 3D Stacked IC with two chips. Four test instances exist for the 3D Stacked IC, *viz*, wafer sort of each individual chip (WS1 and WS2, respectively), intermediate test (IT) of the two chips, and package test (PT) of the final 3D Stacked IC.

Table 1 details the given testing times for each instance. To demonstrate the impact of test flows, we consider three sets of yield values for each test instance, as shown in Table 1. For example, for yield Case 1, the yield at the wafer sort of Chip 1 is 0.90.

We use three straightforward test flow schemes:

- Test all (TA): tests are applied at every possible test instance;
- Wafer sort and package test (WSPT): each individual chip is tested at wafer sort and the complete 3D Stacked IC is tested at package test;
- Package test (PT): testing is only applied to the complete 3D Stacked IC at the final test instance, package test;

We compare the expected total test time required to obtain each good 3D Stacked IC, by assuming TA, WSPT and PT as the test flows. Figure 3 illustrates the expected total test time required by the three test flows with the three sets of yield values. Computation details of the expected test time required by the test flows are elaborated in the following Section.

The results show that for Case 1 PT has the lowest expected test time, while in Case 2, the test flow with the lowest expected test time is WSPT, and that in Case 3 TA results in the lowest expected test time. Thus, it can be concluded from the results that a straightforward test flow may not provide the lowest expected test time for any given 3D Stacked IC. In this paper we present a method to obtain a test flow for any given 3D Stacked IC, such that the expected total test time is minimized.

Table 1 SIC with three different sets of yield

Test instance I_{ij}	WS1 I_{11}	WS2 I_{21}	IT I_{22}	PT I_{32}
Test time T_{ij}	10	10	30	70
Yield y_{ij}(Case 1)	0.90	0.91	0.92	0.93
Yield y_{ij}(Case 2)	0.70	0.71	0.72	0.73
Yield y_{ij}(Case 3)	0.50	0.51	0.52	0.53

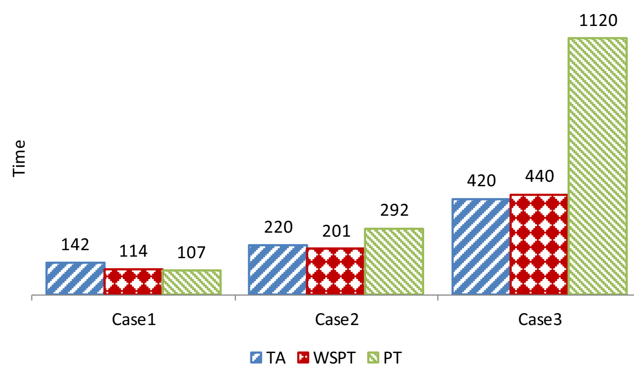


Fig. 3 Comparison of expected total test times for three test flows – TA, WSPT and PT respectively – on one design with three sets of yield values: Case 1, Case 2, and Case 3. PT, WSPT and TA require the lowest expected test times for Case1, Case2 and Case3 respectively

5 Expected Total Test Time Estimation

In this section we derive an expression to calculate the expected total test time required to produce each fault-free 3D Stacked IC for any assumed test flow. For reference, we assume the design provided in Table 1 with yield Case 1.

We elaborate the notations given in Table 2 using Figs. 4 and 5. First, the given notations corresponding to given data are discussed, followed by the notations that refer to values that need to be calculated for a selected test flow. Finally, we discuss the notations that represent values that hold true for all 3D Stacked ICs considered in this paper. We assume a given 3D Stacked IC with N chips in the stack, where each chip is denoted by i, $1 \leq i \leq N$. A test instance is denoted by I_{ij}, illustrated in Fig. 5. Each test instance, I_{ij}, requires test time T_{ij} and has a yield y_{ij}.

The wafer sort instances are illustrated by the boxes in the upper row, where $j = 1$, which means I_{11} to I_{N1} are the instances for wafer sort. The wafer sort instance of a chip i is indicated with I_{i1} to the left in Fig. 4. We have $j = 2$ for intermediate tests of partial stacks with i chips, and for package test instances, I_{i2}. In Fig. 5, the bottom row indicates the intermediate test instances and the package test. For an intermediate test instance I_{i2}, the intermediate stack consists of all chips from 1 to i. For example, if $i = 3$, the intermediate test instance I_{32} consists of the partial stack of chip 1, 2 and 3. The intermediate test instances and package test are illustrated to the right in Fig. 4.

A partial stack, with i chips, is tested during the intermediate instance, I_{i2}, which comprises of components from two previous instances – a partial stack with $i - 1$ chips, and chip i. This is illustrated by arrows connecting instance $I_{i-12} \rightarrow I_{i2}$, and $I_{i1} \rightarrow I_{i2}$, in Fig. 5.

We now discuss the values that need to be computed to obtain the desired test flow. A test flow is represented by the vector $X = (x_{11}...x_{N1}), (x_{22}...x_{N2}), (x_{N+12})$, with $2N$ elements, that include N wafer sorts, $N - 1$ intermediate tests and 1 package test. Each element is denoted by the binary decision variables x_{ij}, for each box in Fig. 5, we set $x_{ij} = 1$ when a test is performed at instance I_{ij}, and $x_{ij} = 0$ when no tests are performed at instance I_{ij}. Let us consider the example of a 3D Stacked IC with 2 chips in the stack, as illustrated in Section 4. The vectors for TA, WSPT and PT would be represented as: $X = (1, 1), (1), (1)$, $X = (1, 1), (0), (1)$ and $X = (0, 0), (0), (1)$ respectively. For convenience, we also define $\bar{x}_{ij} = 1 - x_{ij}$.

τ is the total expected time taken by the test flow given by X. The total expected time depends on what tests are applied in a test flow. At each instance, the effective yield is computed depending on the tests that have been previously performed. To enable computation of test time, for a test instance I_{ij}, we let Q_{ij} denote the number of good units that need to be produced at instance I_{ij}, $T_{eff}(ij)$ denote the expected time taken to produce each good unit at instance I_{ij}, and Y_{ij} the effective yield. The objective of this paper is to minimize the expected total test time τ.

Finally, it is given for all 3D Stacked ICs that, instances I_{12} and I_{N+11} do not exist. Instance I_{12} corresponds to the box at the bottom left of Figure 5, for the intermediate test of only chip 1, which does not exist as at least two stacked chips are tested at any intermediate test. Again, for a 3D Stacked IC comprising of N chips in the stack, instance I_{N+11} corresponding to the box at the top right corner of Fig. 5, refers to wafer sort of chip $N+1$, also does not exist. Therefore, it may be assumed that these instances require no test time, *i.e.*, $T_{12} = T_{N+11} = 0$, and also have a perfect yield, $y_{12} = y_{N+11} = 1$, for the generic expressions.

In the following Sections 5.1, 5.2, and 5.3 respectively, we elaborate each component of the expression, *viz*, yield Y_{ij}, quantity Q_{ij} and time $T_{eff}(ij)$.

5.1 Yield

The effective yield at each test instance is presented here, which is given as a function of the given yield values of the preceding test instances, depending on whether a test was performed during that instance. We will discuss effective yield first for wafer sort and then for intermediate and package tests.

Let us consider an arbitrary wafer sort test instance I_{i1}, which has the given yield of y_{i1}. As there are no prior tests of the chip at wafer sort, the effective yield depends only on the yield at test instance I_{i1}.

For wafer sort instances the effective yield Y_{i1} at test instance I_{i1} is given as:

$$Y_{i1} = y_{i1} \tag{1}$$

Table 2 List of notations

Input data	
N	Number of chips constituting the stack
I_{ij}	Test instance corresponding to i^{th} chip
T_{ij}	Time taken to test a single unit at instance I_{ij}
y_{ij}	Yield of the manufacturing stage I_{ij}
i	i^{th} chip in the stack, $1 \leq i \leq N$
	index $N+1$ is used for package tests
j	$j = 1 \implies$ wafer sort
	$j = 2 \implies$ intermediate and package test
Known data	
I_{12}	Does not exist
	$T_{12} = 0$
	$y_{12} = 1$
I_{N+11}	Does not exist
	$T_{N+11} = 0$
	$y_{N+11} = 1$
Q_{N+12}	$= 1$
Q_{i2}	$= Q_{i-12}$
Calculated	
x_{ij}	Binary decision variable
	1; if test is performed at instance I_{ij}
	0; otherwise
$\bar{x}_{ij}$	$1 - x_{ij}$
X	Test flow vector composed of x_{ij}
	$X = (x_{i1} \mid 1 \leq i \leq N), (x_{i2} \mid 2 \leq i \leq N), (x_{N+12} = 1)$
Q_{ij}	Number of good units that need to be produced at instance I_{ij}
$T_{eff}(ij)$	Expected time taken to produce each good unit at instance I_{ij}
Y_{ij}	Effective yield at instance I_{ij}
τ	Expected total test time taken by the test flow given by X

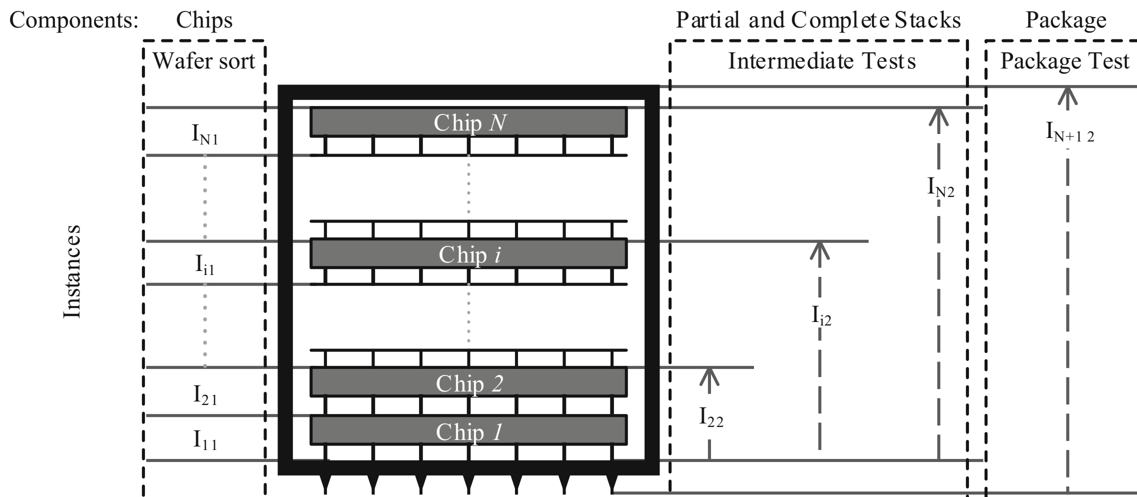


Fig. 4 Components of the 3D Stacked IC that are tested at instance I_{ij}. Wafer sort of individual chips are marked on the left, from I_{11} to I_{N1}. Intermediate tests, I_{22} to I_{N2} of the 3D Stacked IC with 2 to N chips are to the right. Package test I_{N+12} is at the right

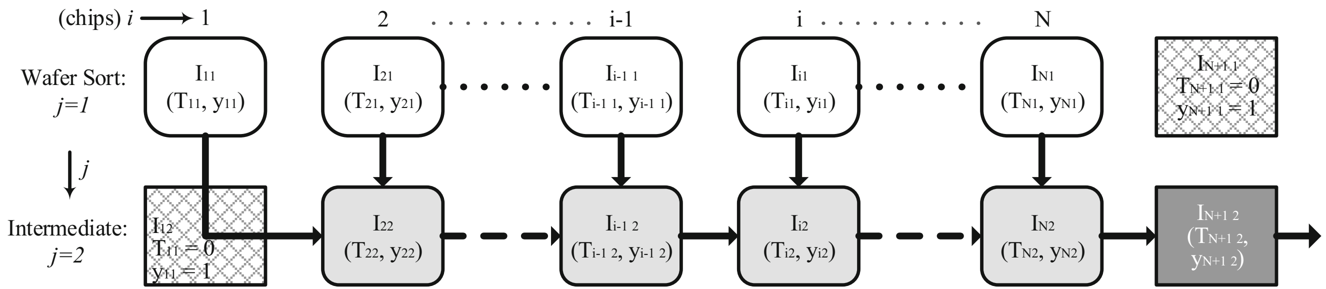


Fig. 5 Test instances I_{ij} of a 3D Stacked IC with N chips in the stack. Upper row indicates wafer sort instances of chips $i = 1$ to N; from I_{11} to I_{N1}. Lower row indicates intermediate test instances, after stacking chips $i = 2$ to N to the incomplete stack, and package test instances; I_{22} to I_{N2}. Connecting arrows between instances (boxes) indicate transfer of components. Boxes representing instances I_{12} and I_{N+11} are represented for the sake of convenience

In the example the yield of chip 2 for case 1 at wafer sort is $y_{21} = 0.91$, which means that the effective yield Y_{21}, which is wafer sort test of chip 2, is:

$$Y_{21} = y_{21} = 0.91 \tag{2}$$

Next, we discuss the effective yield at intermediate and package test instances. Let us consider an arbitrary intermediate or package test instance I_{i2}. For the intermediate test instances the given yield y_{i2}, represents the yield of producing an intermediate stack with i chips, assuming that both components, *i.e*., the chip i and the intermediate stack with $i-1$ chips, are tested and are defect free. Contrary to wafer sort instances, as seen from Fig. 5, any intermediate test instance, $I_{i2} \forall i > 2$, receives components from the preceding partial stack with $i-1$ and wafer manufacturing stage of chip i. Now, if no tests were performed at the wafer sort instance I_{i1}, the defective components will be passed on to the corresponding intermediate test I_{i2}, which would decrease the effective yield at the intermediate test instance I_{i2}. This would result in the effective yield to be the product of the yield of the individual test instances I_{i1} and I_{i2}, *i.e.*, $(y_{i2} \cdot y_{i1})$. For example, with the design in Section 4, the yield at the intermediate test instance of stacking chip 1 and chip 2, I_{22} is $0.90 \cdot 0.92 = 0.828$, assuming $X = (1, 0), (1), (1)$, as wafer sort has not been performed for chip 2 at instance I_{21}. Similarly, the yield at any intermediate instance I_{i2} depends on tests performed during previous instances $I_{i'2}$, where, $i' < i$. Therefore, to calculate the effective yield at any intermediate test instance I_{ij} as a effect of the yield of the instance itself and that of the corresponding wafer sort instance I_{i1}, we use the expression $y_{ij}(y_{i1}^{\bar{x}_{i1}})$. If $x_{ij} = 0$ we get $y_{i1}^{\bar{x}_{i1}} = y_{i1}^1 = y_{i1}$ and $x_{ij} = 1$ gives $y_{i1}^{\bar{x}_{i1}} = y_{i1}^0 = 1$. Therefore, the expression gives y_{ij} when $x_{i1} = 1$ *i.e*., when test was performed at wafer sort, and $y_{ij} \cdot y_{i1}$ when $x_{i1} = 0$ *i.e*., when no tests were performed at wafer sort. However, the effective yield at any intermediate instance I_{i2} will also depend on the yield of the preceding intermediate instance I_{i-12} when test was not performed, which in turn will depend on the yield of I_{i-22} and so on, unless a test had been performed at any of these prior instances. In other words, the yield at any intermediate test instance I_{ij} is a consequence of of the preceding intermediate test instances up to $I_{i'j} \quad \forall i' < i$, which is the latest intermediate test instance when a test has been performed. Therefore, the first intermediate test instance I_{22} depends only on the preceding wafer sort instances I_{11} and I_{21}, as there are no previous intermediate tests possible. Thus we can derive the effective yield at instance I_{22} as:

$$Y_{22} = y_{22} \cdot y_{11}^{\bar{x}_{11}} \cdot y_{21}^{\bar{x}_{21}} \tag{3}$$

For the 3D Stacked IC in Section 4 with yield Case 1 and test flow $X = (0, 1), (1), (1)$, we have $x_{11} = 0$, $x_{21} = 1$, $x_{22} = 1$ and $x_{32} = 1$. Hence we can compute Y_{22} as:

$$\begin{aligned} Y_{22} &= y_{22} \cdot y_{11}^{\bar{x}_{11}} \cdot y_{21}^{\bar{x}_{21}} \\ &= 0.92 \cdot 0.90^1 \cdot 0.91^0 \\ &= 0.92 \cdot 0.90 \cdot 1 = 0.8280 \end{aligned} \tag{4}$$

Similarly, the effective yield at the following intermediate or package test instance I_{32} depends on all previous tests performed, and can be given as:

$$\begin{aligned} Y_{32} &= y_{32} \cdot \left(y_{31}^{\bar{x}_{31}} \cdot \left(y_{22} \cdot \left(y_{11}^{\bar{x}_{11}} \cdot y_{21}^{\bar{x}_{21}} \right) \right)^{\bar{x}_{22}} \right. \\ &= y_{32} \cdot Y_{22}^{\bar{x}_{22}} \end{aligned} \tag{5}$$

As seen in Fig. 5, all intermediate test instances, depicted by the bottom row, receive components from preceding wafer sort and intermediate test instances. The yield, depending on the preceding instances, is:

$$Y_{i2} = \begin{cases} y_{i2} \cdot y_{i1}^{\bar{x}_{i1}} \cdot Y_{i-12}^{\bar{x}_{i-12}}, & \text{for } 2 < i \leq N+1 \\ y_{22} \cdot y_{11}^{\bar{x}_{11}} \cdot y_{21}^{\bar{x}_{21}}, & \text{for } i = 2 \end{cases} \tag{6}$$

Therefore, in case of package tests, we will have:

$$Y_{N+12} = y_{N+12} \cdot y_{N+11}^{\bar{x}_{N+11}} \cdot Y_{N2}^{\bar{x}_{N2}} \tag{7}$$

Where, it is given that $y_{N+11} = 1$, as noted in Table 2 and we set $\bar{x}_{N+11} = 1$.

For the given package test instance I_{32}, in this example, the preceding wafer sorts were performed, while the intermediate test instance was avoided; such that $X = (1, 1), (0), (1)$. Therefore, the yield of only the intermediate test instance, I_{22}, affects the package test instance:

$$\begin{aligned} Y_{32} &= y_{32} \cdot y_{31}^{\bar{x}_{31}} \cdot Y_{22}^{\bar{x}_{22}} \\ &= y_{32} \cdot y_{31}^{\bar{x}_{31}} \cdot \left\{ y_{22}^{\bar{x}_{22}} \cdot \left(y_{11}^{\bar{x}_{11}} \cdot y_{21}^{\bar{x}_{21}} \right)^{\bar{x}_{22}} \right\} \\ &= 0.93 \cdot 1^1 \cdot \{0.92^1 \cdot (0.90^0 \cdot 0.91^0)^1\} \\ &= 0.93 \cdot 1 \cdot 0.92 \cdot 1 \cdot 1 = 0.8556 \end{aligned} \tag{8}$$

5.2 Quantity

At any test instance, the number of units tested is greater than the number of good units obtained, due to imperfect (< 1) yield. Therefore, we calculate the expected quantity of good units required at the end of each instance such that a fixed number of good units are obtained from a succeeding manufacturing stage.

Let us start with the package test instance, illustrated by the rightmost box in Fig. 5. With a yield of $y_{N+12}(< 1)$, to obtain $Q_{N+12} = 1$ good packages we need to test Q_{N+12}/y_{N+12} packages. Therefore, at the preceding instance I_{N2}, we need to produce $Q_{N2} = Q_{N+12}/y_{N+12}$ good units. Now, to produce Q_{N2} good units at the intermediate test instance I_{N2}, we need to test $Q_{N2}/(y_{N2} \cdot y_{N1}^{\bar{x}_{N1}})$ units. It is useful to note here that the number of instances that need to be tested during the intermediate test instance I_{N2} increases by $1/y_{N1}$ times if test was not performed at the wafer sort instance I_{N1}. This is due to the share of the defective wafers that pass on to the intermediate stack. Consequently, we need to produce $Q_{N1} = Q_{N2}/(y_{N2} \cdot y_{N1}^{\bar{x}_{N1}})$ good intermediate stacks and back calculate the number of good units that need to be produced after each test instance up to Q_{11}. The quantity of good units required after each test instance Q_{ij} is formulated as follows.

For wafer sort instances:

$$Q_{i1} = \begin{cases} \dfrac{Q_{i2}}{y_{i2} \cdot y_{i+11}^{\bar{x}_{i+11}}}, & 2 < i \leq N \\ \dfrac{Q_{22}}{y_{22} \cdot y_{11}^{\bar{x}_{11}} \cdot y_{21}^{\bar{x}_{21}}}, & i \leq 2 \end{cases} \tag{9}$$

For the 3D Stacked IC in Section 4 with yield Case 1 and $X = (0, 0), (0), (1)$, we utilize the second part of Eq. 9, since $i = 1 \leq 2$. Therefore, the number of units required at the end of instance I_{11} is:

$$\begin{aligned} Q_{11} &= \frac{1}{y_{11}^{\bar{x}_{11}} \cdot y_{21}^{\bar{x}_{21}} \cdot y_{22}^{\bar{x}_{22}} \cdot y_{32}^{\bar{x}_{32}}} \\ &= \frac{1}{0.90^1 \cdot 0.91^0 \cdot 0.92^1 \cdot 0.93^1} = 1.299 \end{aligned} \tag{10}$$

For intermediate test instances:

$$Q_{i2} = \begin{cases} \dfrac{Q_{i+12}}{y_{i+12} \cdot y_{i+11}^{\bar{x}_{i+11}}}, & 2 \leq i \leq N-1 \\ \dfrac{Q_{N+12}}{y_{N+12}}, & i = N \end{cases} \tag{11}$$

In the example, the quantity required for package test, when only the wafer sorts have been performed, such that $X = (1, 1), (0), (1)$ is:

$$Q_{22} = \frac{Q_{32}}{y_{32}} = \frac{1}{0.93} = 1.075 \tag{12}$$

It should be noted that at any intermediate test instance I_{i2}, for each intermediate stack comprising of chips 1 to $i-1$ obtained from instance I_{i-12}, per chip i from instance I_{i1} is stacked. Therefore, we need equal number of units from preceding instances I_{i1} and I_{i-12}, giving $Q_{i1} = Q_{i-12}$.

5.3 Time

An expression to calculate the expected total time taken by any test flow to produce each fault-free packaged 3D Stacked IC is formulated here. The time expected at each test instance depends on the time taken to test each unit at the instance, the effective yield, as well as the number of units required at successive test instances to produce the desired number of good packages. Table 3 is used to list the expected test time at each instance for different test flows required by the 3D Stacked ICs mentioned in Table 1.

The effective test time, $T_{eff}(ij)$, spent at any instance, I_{ij}, depends on the given test time, T_{ij}, effective yield, Y_{ij}, at the instance and the number of defect-free units that need to be obtained, Q_{ij}, and the binary decision variable, x_{ij}, is given by:

$$T_{eff}(ij) = \frac{T_{ij}}{Y_{ij}} \cdot Q_{ij} \cdot x_{ij} \tag{13}$$

Table 3 Effective test time at each test instance Table 1

Test Flow	Teff(11)	Teff(21)	Teff(22)	Teff(32)	τ
Case 1					
TA	12.99	12.84	35.06	75.27	136.16
WSPT	12.99	12.84	0.00	81.81	107.64
PT	0.00	0.00	0.00	99.89	99.89
Case 2					
TA	27.18	26.80	57.08	95.89	206.94
WSPT	27.18	26.80	0.00	133.18	187.16
PT	0.00	0.00	0.00	267.97	267.97
Case 3					
TA	72.57	71.15	108.85	132.08	384.64
WSPT	72.57	71.15	0.00	253.99	397.71
PT	0.00	0.00	0.00	996.04	996.04

For example, the effective time spent at the intermediate test instance I_{22} when all tests are performed, as seen in the first row of Case 1; such that $X = (1,1), (1), (1)$ is:

$$T_{eff}(22) = \frac{T_{22}}{Y_{22}} \cdot Q_{22} \cdot x_{22} = \frac{30}{0.92 \cdot 0.90^0 \cdot 0.91^0} \cdot 1.075 \cdot 1 = 35.06 \tag{14}$$

The sum of the effective test times, $T_{eff}(ij)$, at each instance, I_{ij}, gives the expected total test time required, τ, by any selected test flow, X, to produce each fault-free packaged 3D Stacked IC as shown below.

$$\tau = \sum_{i=1}^{N+1} T_{eff}(ij) \qquad \forall j = 1, 2 \tag{15}$$

The expected total test time, assuming a test flow when all instances are tested, as seen in the topmost row of Case 1, gives:

$$\tau = T_{eff}(11) + T_{eff}(21) + T_{eff}(22) + T_{eff}(32) = 12.99 + 12.84 + 35.06 + 75.27 = 136.16 \tag{16}$$

The objective is to find a suitable test flow for any given 3D Stacked IC, such that the expected total test time τ is minimized.

6 Test Flow Selection Algorithm (TFSA)

In this section, we first detail the Test Flow Selection Algorithm (TFSA) and then we detail the computational complexity of the algorithm.

Given the test time T_{ij} and yield y_{ij} at all test instances I_{ij}, the TFSA generates a test flow, X, by iteratively trying to reduce the expected total test time τ. At each iteration, the test instance that contributes to most reduction in τ is selected. As discussed in the previous section, we represent a test flow with the vector $X = (x_{11}...x_{N1}), (x_{22}...x_{N2}), (x_{N+12})$, where $(x_{N+12}) = 1$, since package test is always performed.

TFSA, which is detailed in Algorithm 1, in line 1, takes as input N chips where for each test instance I_{ij} $(1 \leq i \leq N, 1 \leq j \leq 2)$ the test time T_{ij} and y_{ij} are given. We use the 3D Stacked IC described in Section 4, with yield Case 3 in Table 1 for illustration. In the example, there are 2 chips ($N = 2$); hence, there are $2 \cdot N = 4$ possible test instances.

Algorithm 1 Test Flow Selection Algorithm (TFSA).

```
1:  Input: A 3D Stacked IC with N chips
        for each test instance I_ij: test time T_ij and yield y_ij
2:  Initialize:
        X as x_ij = 0 for i ∈ 1..N; j ∈ [1, 2] and
    x_N+12 = 1
        Compute expected total test time τ using Eq. 15
3:  for Counter = 1 to (2N − 1) do
4:      Set: í = 0 and j́ = 0
5:      for i = 1 to N do
6:          for j = 1 to 2 do
7:              if x_ij = 0 then
8:                  Set: x_ij = 1
9:                  Compute: τ́
10:                 if τ́ < τ then
11:                     Set: τ = τ́
12:                     Set: í = i and j́ = j
13:                 end if
14:                 Revert: x_ij = 0
15:             end if
16:         end for
17:     end for
18:     Set: x_íj́ = 1
19: end for
20: Output: X, τ
```

The test flow vector X, and the corresponding test cost τ are initialized in line 2. All binary decision variables, x_{ij}, are initialized such that only package test is applied. For the example, we set $X = (0,0), (0), (1)$. After initialization, the expected time is computed with Eq. 15 to:

$$\begin{aligned}\tau &= \sum_{\forall i,j} T_{eff}(ij) \\ &= T_{eff}(11) + T_{eff}(21) + T_{eff}(22) + T_{eff}(32) \\ &= 0 + 0 + 0 + \frac{T_{32}}{Y_{32}} \cdot Q_{32} \cdot x_{32} \\ &= \frac{T_{32}}{y_{32} \cdot (y_{22} \cdot (y_{11}^{\bar{x}_{11}} \cdot y_{21}^{\bar{x}_{21}}))^{\bar{x}_{22}}} \cdot 1 \cdot 1 \\ &= \frac{70}{0.53 \cdot (0.52 \cdot (0.50^1 \cdot 0.51^1))^1} \cdot 1 \cdot 1 \qquad = 996\end{aligned} \tag{17}$$

As noted in Table 2, $T_{eff}(12) = 0$ and $T_{eff}(31) = 0$.

When only package test is applied, the actual yield at package test takes the yield at all instances into account.

A variable, $Counter$, is active between lines 3 → 19, to ascertain $2N - 1$ iterations. In this example, $Counter$ iterates from 1 → 3. Variables $\acute{i}$ and $\acute{j}$ are reset for the iteration, in line 4.

To scan through all $2N - 1$ test instances I_{ij}, variables i and j are defined between lines 5 → 17 and lines 6 → 16, respectively, where $1 \leq i \leq N$ and $1 \leq j \leq 2$.

During an iteration, each inactive test instance $x_{ij} = 0$ is set to $x_{ij} = 1$, in line 7 → 8. The corresponding test cost $\acute{\tau}$ is computed in line 9, as a result of the modified test flow, to evaluate if there is a benefit to include the test instance in the test flow. In the first iteration, the matrix is updated to $(1, 0), (0), (1)$.

At line 9, the effective total test time $\acute{\tau}$ for the current test flow $X = (1, 0), (0), (1)$ is computed as:

$$\begin{aligned}
\acute{\tau} &= \sum_{\forall i,j} T_{eff}(ij) \\
&= T_{eff}(11) + T_{eff}(21) + T_{eff}(22) + T_{eff}(32) \\
&= \frac{T_{11}}{Y_{11}} \cdot Q_{11} \cdot x_{11} + 0 + 0 + \frac{T_{32}}{Y_{32}} \cdot Q_{32} \cdot x_{32} \\
&= \frac{T_{11}}{y_{11}} \cdot \frac{1}{y_{12} \cdot y_{22} \cdot y_{32}} \cdot 1 + 0 + 0 \\
&\quad + \frac{T_{32}}{y_{32} \cdot (y_{22} \cdot (y_{11}^{\bar{x}_{11}} \cdot y_{21}^{\bar{x}_{21}}))^{\bar{x}_{22}}} \cdot 1 \cdot 1 \\
&= \frac{10}{0.50} \cdot \frac{1}{0.51 \cdot 0.52 \cdot 0.53} \cdot 1 \\
&\quad + \frac{70}{0.53 \cdot (0.52 \cdot (0.50^{0} \cdot 0.51^{1}))^{1}} \cdot 1 \cdot 1 \qquad = 640
\end{aligned} \tag{18}$$

Note, in this case, wafer sort is applied to chip 1, which means y_{11} is used at test instance I_{11}, and not in test instance I_{32}.

If the new test cost $\acute{\tau}$ is lower than all previously computed test costs τ (line 10), the test cost is updated as $\tau = \acute{\tau}$ and the indices i and j are recorded, at lines 11 and 12, respectively. In the example, $\acute{\tau} = 640$ and $\tau = 996$. Hence, a better solution is found, thus replacing τ by $\acute{\tau}$. The algorithm continues with the test flow $(0, 1), (0), (1)$ which gives $\acute{\tau} = 640$, same as the present value of $\tau = 640$. Hence, $\acute{i}$ and $\acute{j}$ are not updated. For the test flow $(0, 0), (1), (1)$, we get $\acute{\tau} = 558 < \tau = 640$, and update indices $\acute{i} = 2$ and $\acute{j} = 2$.

Hence, at the end of the first iteration, in line 18, we update the test flow to $X = (0, 0), (1), (1)$.

Similarly, at the end of the second iteration $Counter = 2$, we will have $X = (1, 0), (1), (1)$ and $\tau = 487$. Eventually, at the third and final iteration $Counter = 2N - 1 = 3$, we will have $X = (1, 1), (1), (1)$ and $\tau = 384$. Thus, for this example, $(1, 1), (1), (1)$, means wafer sort is applied to both chip 1 and chip 2, intermediate test is applied to the stack of chips $1 + 2$ and package test is applied to the complete stack. The expected total test time is 384.

6.1 Complexity Estimation

There are two nested iterations in algorithm 1. The outer iteration, *for* loops between lines 3 → 19, iterates $Counter$ from 1 to $2N - 1$, and the inner iteration, *for* loops between lines 5 → 17, iterates i from 1 to N. For each i, j takes two values 1 and 2. Thus, the complexity is the product of the number of iterations of each loop, *i.e.*, $(2N - 1) \cdot N \cdot 2 = 4N^2 - 2N$, which is of order $O(N^2)$.

7 Experiments

In this section we present two sets of experiments. First we compare the expected total test times obtained from TFSA with respect to three straightforward test flows and the test flow obtained by exhaustive search. Next, the TFSA is integrated with test planning of core-based 3D Stacked ICs with a IEEE 1500 based test architecture, to optimize the test time.

7.1 Test Flow Selection

The objective is to compare the expected total test time by applying TFSA and that required with the three straightforward test flow schemes (TA, WSPT, and PT) as well as against exhaustive search.

Experiments were performed on two sets of 3D Stacked IC designs with 2 to 10 chips in the stack. The 3D Stacked IC designs are detailed in Table 5. For example, in case of both Set 1 and Set 2, SIC_2 consists of three chips in the stack, chip 1, 2, and 3. The test time and yield values of each chip in the 3D Stacked IC designs at wafer sort are given for Set 1 and Set 2 in Table 4. In Set 1, chip 1 has a test time of 1000 time units and a yield of 0.62. The test times

Table 4 Test times and yields of Chips 1 to 10

Chip	1	2	3	4	5	6	7	8	9	10
Set 1										
Test time	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
Yield	0.62	0.66	0.70	0.74	0.78	0.82	0.86	0.90	0.94	0.98
Set 2										
Test time	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
Yield	0.98	0.94	0.90	0.86	0.82	0.78	0.74	0.70	0.66	0.62

Table 5 Designs

3D Stacked IC designs	Chips in the 3D Stacked ICs as detailed in Table 4. First chip is lowermost.
SIC_1	1, 2
SIC_2	1, 2, 3
SIC_3	1, 2, 3, 4
SIC_4	1, 2, 3, 4, 5
SIC_5	1, 2, 3, 4, 5, 6
SIC_6	1, 2, 3, 4, 5, 6, 7
SIC_7	1, 2, 3, 4, 5, 6, 7, 8
SIC_8	1, 2, 3, 4, 5, 6, 7, 8, 9
SIC_9	1, 2, 3, 4, 5, 6, 7, 8, 9, 10

are kept constant for all chips in the stack for both Set 1 and Set 2, to emphasize the difference among the expected total test times. However, the yield values for Set 1 and Set 2 are changed to emphasize the differences among the test flows obtained. For intermediate tests and package test, we assume an additional test time of 1000 units and the given yield at the test instance to be 0.70. For instance, the test time assumed during the package test of SIC_3 is the sum of the the test times of each individual chip − chips 1, 2 and 3 −, two layers of interconnects − between chips 1 and 2, and chips 2 and 3 − give an additional 2×1000, and finally 1000 time units for testing the package itself (Table 5).

The results of the comparison between TFSA, TA, PT, WSPT and exhaustive search are collated in Table 6. Table 6 is organized as follows. The leftmost column lists the 3D Stacked IC designs. The following group of five columns list the expected total test times required by each method. The rightmost group of four columns depicts for each method the overhead in expected total test time time compared to the optimal expected total test time obtained by exhaustive search. The most significant points that can be drawn from Table 6 are:

- TFSA generates test flows and corresponding test times very close to exhaustive search in most cases.
- PT has a low expected total test time for 3D Stacked ICs with up to three chips in the stack: for SIC_1 of Set 1, the result is only 1% away from optimum, whereas the optimal is obtained for SIC_1 and SIC_2 of Set 2. However, for all other cases, PT produces results that are far from optimum. As the number of chips in the 3D Stacked IC increases, the performance of PT deteriorates.
- TA results give expected total test times that are about 40% more than optimum for Set 1 and over 80% worse at an average for Set 2.
- WSPT is not as efficient as the TFSA. However, it is interesting to note that WSPT produces optimal results when the number of chips is less than 4, and WSPT is

Table 6 Comparison of expected total test times

3D Stacked ICs	Total test time (τ)					Difference(%)			
	Exhaustive	TFSA	TA	PT	WSPT	TFSA	TA	PT	WSPT
Set 1									
SIC_1	10874	10874	13812	10972	10874	0	27	1	0
SIC_2	21656	21656	29402	33588	21656	0	36	55	0
SIC_3	38277	38277	53088	86456	38277	0	39	126	0
SIC_4	63842	63842	88354	197930	63842	0	38	210	0
SIC_5	101631	101631	140179	413792	103042	0	36	302	1
SIC_6	159216	178426	215669	801922	162933	12	35	404	2
SIC_7	184017	184017	324978	1454734	254110	0	77	691	38
SIC_8	346572	346572	482610	2487197	392444	0	39	618	13
SIC_9	503730	503730	709280	4028502	601658	0	41	700	19
Set 2									
SIC_1	4874	4874	11682	4874	8743	0	140	0	79
SIC_2	11604	11604	25711	11604	17965	0	122	0	55
SIC_3	25702	25702	47430	25702	32619	0	85	10	27
SIC_4	45946	45946	80144	55971	55632	0	74	22	21
SIC_5	75558	75558	128577	123013	91441	0	70	63	21
SIC_6	121359	127854	199481	277057	146744	5	64	128	21
SIC_7	188408	188408	302500	646197	231631	0	61	243	23
SIC_8	284257	284257	451419	1573533	361253	0	59	454	27
SIC_9	421316	421316	665931	4028502	558309	0	58	856	33

only a few % away from optimum when the number of chips in the stack is less than eight.

Table 7 lists the test flows obtained by the exhaustive search and the TFSA, respectively. It is interesting to note, first of all, the test flows proposed by the TFSA are, in most cases, the same as the test flows obtained from exhaustive search. However, for exhaustive search the expected total test time for 2^{2N-1} test flows need to be evaluated, whereas TFSA only compares $(2N-1)^2$ test flows. Therefore, TFSA requires lower computation time, as compared to exhaustive search, to determine a test flow for 3D Stacked ICs with more than 2 chips in the stack. In case of 3D Stacked ICs with 9 chips in the stack, for TFSA, the test flow was determined in just over 2 minutes, whereas exhaustive search required longer than 2 days to arrive at the same result in Table 7. Secondly, the optimal test flow does not follow a regular pattern. For example, SIC_6 and SIC_7 of Set 1 differ by only one chip. But the test flows are very different. In addition, it is also observed that performing wafer sort pays off in most cases.

7.2 Test Architecture Design

In the second set of experiments, the goal is to compare the expected total test times obtained by integrating test architecture design and test planning schemes with different test flows. We evaluate (1) TFSA against three straightforward test flow schemes (TA, WSPT, and PT) against an exhaustive search of all possible test flows, and (2) the test flows on three test architecture designs and test planning schemes. The objective here is to integrate test flow selection and test architecture design to obtain the minimal test cost.

We assume that the given 3D Stacked ICs are core-based, and each core is provided with a IEEE 1500 based core test wrapper. The problem at system-level is given a TAM width (W) to find the most suitable number of TAM groups, their widths, and assign the cores to the TAM groups such that test time is minimized. The three test planning schemes that we assume are:

- Scheme 1, the TAM for each chip is optimized independently of all other chips in the 3D Stacked IC. It means that each chip gets the TAM that is most suitable for its wafer sort. Note that after the optimization additional TAM wires can be added to a chip. For example, if the top chip requires a wide TAM while all other chips only need a narrow TAM, the wide TAM is added to all chips to make testing of the top chip possible at package test.
- Scheme 2, the TAM for the lowest chip is optimized and that TAM architecture is used for all chips in the 3D Stacked IC. In this case, all chips use the TAM

Table 7 Comparison of test flows obtained with exhaustive search against TFSA

Design	Test flow for exhaustive search	Test flow for TFSA
Set 1		
SIC_1	(1, 1), (0), (1)	(1, 1), (0), (1)
SIC_2	(1, 1, 1), (0, 0), (1)	(1, 1, 1), (0, 0), (1)
SIC_3	(1, 1, 1, 1), (0, 0, 0), (1)	(1, 1, 1, 1), (0, 0, 0), (1)
SIC_4	(1, 1, 1, 1, 1), (0, 0, 0, 0), (1)	(1, 1, 1, 1, 1), (0, 0, 0, 0), (1)
SIC_5	(1, 1, 1, 1, 1, 1), (0, 1, 0, 0, 0), (1)	(1, 1, 1, 1, 1, 1), (0, 1, 0, 0, 0), (1)
SIC_6	(0, 0, 1, 1, 1, 1, 1), (1, 0, 1, 0, 0, 0), (1)	(1, 1, 1, 1, 1, 1, 1), (0, 1, 0, 0, 0, 0), (1)
SIC_7	(1, 1, 1, 1, 1, 1, 1, 1), (0, 0, 0, 0, 0, 1, 0), (1)	(1, 1, 1, 1, 1, 1, 1, 1), (0, 1, 0, 0, 0, 0, 0), (1)
SIC_8	(1, 1, 1, 1, 1, 1, 1, 1, 1), (0, 1, 0, 0, 1, 0, 0, 0), (1)	(1, 1, 1, 1, 1, 1, 1, 1, 1), (0, 1, 0, 0, 1, 0, 0, 0), (1)
SIC_9	(1, 1, 1, 1, 1, 1, 1, 1, 1, 1), (0, 1, 0, 0, 1, 0, 0, 0, 0), (1)	(1, 1, 1, 1, 1, 1, 1, 1, 1, 1), (0, 1, 0, 0, 1, 0, 0, 0, 0), (1)
Set 2		
SIC_1	(0, 0), (0), (1)	(0, 0), (0), (1)
SIC_2	(0, 0, 0), (0, 0), (1)	(0, 0, 0), (0, 0), (1)
SIC_3	(0, 0, 1, 1), (1, 0, 0), (1)	(0, 0, 1, 1), (0, 1, 0), (1)
SIC_4	(0, 0, 0, 1, 1), (0, 0, 1, 0), (1)	(0, 0, 0, 1, 1), (0, 0, 1, 0), (1)
SIC_5	(0, 0, 0, 1, 1, 1), (0, 0, 1, 1, 0), (1)	(0, 0, 0, 1, 1, 1), (0, 0, 1, 1, 0), (1)
SIC_6	(0, 0, 1, 1, 1, 1, 1), (0, 0, 1, 1, 0, 0), (1)	(0, 0, 0, 1, 1, 1, 1), (0, 0, 1, 1, 0, 0), (1)
SIC_7	(0, 0, 0, 1, 1, 1, 1, 1), (0, 0, 1, 1, 1, 0, 0), (1)	(0, 0, 0, 1, 1, 1, 1, 1), (0, 0, 1, 1, 1, 0, 0), (1)
SIC_8	(0, 0, 0, 1, 1, 1, 1, 1, 1), (0, 0, 1, 1, 1, 0, 0, 0), (1)	(0, 0, 0, 1, 1, 1, 1, 1, 1), (0, 0, 1, 1, 1, 0, 0, 0), (1)
SIC_9	(0, 0, 0, 1, 1, 1, 1, 1, 1, 1), (0, 0, 1, 1, 1, 0, 0, 0, 0), (1)	(0, 0, 0, 1, 1, 1, 1, 1, 1, 1), (0, 0, 1, 1, 1, 0, 0, 0, 0), (1)

Table 8 Experimental data

Label	Design	Cores	Time	Yield
D	d695	11	695828	0.65
G	g1023	15	731423	0.65
P	p34392	20	16372887	0.75
T	t512505	32	165324037	0.75
Intermediate test			10000	0.65
Package test			10000	0.75

optimized for wafer sort test of the lowest chip in the 3D Stacked IC.

- Scheme SIC, ILP is used to optimize the test architecture for a given test flow. Our ILP scheme [12] is extended from only accepting WSPT to allow an arbitrary test flow.

The four ITC'02 benchmarks used to construct the core-based 3D Stacked ICs are: d695 (D), g1023 (G), p34392 (P), and t512505 (T), as detailed in Table 8. To give an indication of the complexity of the designs, Table 8 details the number of cores in the third column. The test time for each design, when all scan chains within the design are concatenated to a single wrapper chain *i.e.* the TAM width is 1, is tabulated in the fourth column. The yield for each design, as assumed for experiments in this paper, are shown in the last column. For example, d695, represented as D, contains 11 cores, requires a test time of 695828 clock cycles and has a yield of 0.65. The two bottom rows indicate the additional test time incurred during each intermediate test, and during the package test respectively. At each intermediate test an additional 10000 time units are added to the total time at the test instance. For instance, the test time assumed during the package test of DGPT is the sum of the test times of each chip − D, G, P and T − three layers of interconnects in between correspond to 3×10000, and finally 10000 for the package itself. The yield at intermediate test is set to 0.65 and the yield at package test is set to 0.75.

Each ITC'02 benchmark in Table 8 represents a chip in a 3D Stacked IC. By combining the four benchmarks in various ways, we constructed 3D Stacked ICs with 2, 3 and 4 chips. In total we created 9 designs (DP, DT, GP, GT, DGP, DGT, DPT, GPT, and DGPT) where for example the DP design is a 3D Stacked IC with 2 chips consisting of d695 and p34392 where d695 is the lowest chip. The test time required to test each unit at any test instance is obtained from test architecture design and test planning using either Scheme 1, Scheme 2 or Scheme SIC, detailed in [12]. Note that the test time obtained from test architecture design and test planning of the 3D Stacked IC does not change with yield or the output quantity [12].

In the experiments, for each design we applied the four test flows and at each test flow we used the three test architecture design schemes. The results are collated in Table 9, which is organized as follows. There is a sub-table for each of the 9 designs (DP, DT, GP, GT, DGP, DGT, DPT, GPT, and DGPT). Each sub-table is organized in the same manner. As an example, we take the sub-table for DP at the upper left corner of Table 9. The four test flows are listed to the left and for each test flow the test time using the three test architecture schemes are reported. For example, the test time using Scheme SIC with a test flow obtained with TFSA is 5326513. Each test time is compared in two ways. First, at a given test flow the test times are compared against Scheme SIC. For example, the test time obtained with Scheme SIC is 10% lower than that of Scheme 1, where each test architecture scheme is applied on the corresponding test flow obtained with TFSA. Second, for a given test architecture the test times at different test flows are compared. For example, with the SIC Scheme, PT requires 56% higher test time as compared to TFSA.

The results indicate that Scheme SIC is best for all cases. In some cases, Scheme SIC versus Scheme 2 on design GP is 9% better, but in some cases, for example DGPT Scheme SIC is 56% better than Scheme 2, with each test architecture scheme using the test flow obtained by TFSA.

The results indicate that WSPT is as good as TFSA in many cases; however, overall TFSA is close to exhaustive search. In Table 10 the test flows with the lowest test times on any test architecture scheme produced by TFSA is compared against exhaustive search on the designs. For example, the test flow for design DP using exhaustive search is $X = (1,1),(0),(1)$, which means wafer sort of D and P and package test of DP (for details, refer to Section 5).

Table 9 Expected total test times required by 3D stacked ICs for different test flow and test architecture schemes

Test flow	Test architecture scheme: Scheme SIC	Scheme 1	Scheme 2	vs SIC (%): Scheme 1	Scheme 2
	SIC: DP; TAM width = 10				
TFSA	5.3E+6	5.9E+6	6.7E+6	11	14
WSPT	5.3E+6	5.9E+6	6.7E+6	11	14
TA	1.0E+7	1.2E+7	1.4E+7	20	17
PT	8.3E+6	9.5E+6	1.1E+7	14	16
	vs TFSA (%)				
WSPT	0	0	0		
TA	94	102	104		
PT	56	62	65		
	SIC: DT; TAM width = 16				
TFSA	3.0E+7	3.7E+7	4.5E+7	23	22
WSPT	3.0E+7	3.7E+7	4.5E+7	23	22
TA	6.2E+7	7.6E+7	9.6E+7	23	26
PT	5.0E+7	5.9E+7	7.2E+7	18	22
	vs TFSA (%)				
WSPT	0	0	0		
TA	106	105	111		
PT	66	60	57		
	SIC: GP; TAM width = 22				
TFSA	2.5E+6	2.8E+6	3.1E+6	12	11
WSPT	2.5E+6	2.8E+6	3.1E+6	12	11
TA	4.7E+6	5.3E+6	5.6E+6	13	6
PT	3.8E+6	4.4E+6	4.9E+6	16	11
	vs TFSA (%)				
WSPT	0	0	0		
TA	92	88	82		
PT	56	56	60		
	SIC: GT; TAM width = 22				
TFSA	2.2E+7	2.6E+7	3.2E+7	18	23
WSPT	2.2E+7	2.6E+7	3.2E+7	18	23
TA	4.5E+7	5.1E+7	5.6E+7	13	10
PT	3.6E+7	4.2E+7	5.2E+7	17	24
	vs TFSA (%)				
WSPT	0	0	0		
TA	106	94	76		
PT	65	61	64		
	SIC: DGP; TAM width = 25				
TFSA	4.1E+6	4.5E+6	5.2E+6	10	16
WSPT	4.4E+6	5.0E+6	5.8E+6	14	16
TA	5.2E+6	6.0E+6	7.0E+6	15	17
PT	4.6E+6	5.3E+6	6.2E+6	15	17
	vs TFSA (%)				
WSPT	10	12	12		
TA	25	31	32		
PT	12	16	17		
	SIC: DGT; TAM width = 25				
TFSA	3.7E+7	4.1E+7	4.7E+7	11	15
WSPT	4.1E+7	4.6E+7	5.3E+7	12	15
TA	4.7E+7	5.4E+7	6.2E+7	15	15
PT	4.2E+7	4.8E+7	5.6E+7	14	17
	vs TFSA (%)				
WSPT	12	14	15		
TA	27	33	34		
PT	14	18	20		
	SIC: DPT; TAM width = 30				
TFSA	3.5E+7	3.8E+7	4.4E+7	9	16
WSPT	3.7E+7	4.2E+7	4.9E+7	14	17
TA	4.4E+7	5.1E+7	5.9E+7	16	16
PT	3.8E+7	4.3E+7	5.1E+7	13	19
	vs TFSA (%)				
WSPT	7	9	10		
TA	25	30	33		
PT	9	13	15		
	SIC: GPT; TAM width = 16				
TFSA	6.5E+7	7.2E+7	8.3E+7	11	15
WSPT	7.0E+7	7.9E+7	9.1E+7	13	15
TA	8.2E+7	9.5E+7	1.1E+8	16	16
PT	7.1E+7	8.1E+7	9.5E+7	14	17
	vs TFSA (%)				
WSPT	7	9	10		
TA	26	32	33		
PT	9	13	15		
	SIC: DGPT; TAM width = 30				
TFSA	3.5E+7	4.3E+7	6.3E+7	23	47
WSPT	5.0E+7	6.4E+7	9.9E+7	28	55
TA	5.2E+7	6.2E+7	9.2E+7	19	48
PT	6.8E+7	9.0E+7	1.3E+8	32	44
	vs TFSA (%)				
WSPT	45	48	57		
TA	45	45	45		
PT	97	108	110		

Table 10 Test flow used with SIC Scheme to minimize test time

Design	Exhaustive	TFSA
DP	(1, 1), (0), (1)	(1, 1), (0), (1)
DT	(1, 1), (0), (1)	(1, 1), (0), (1)
GP	(1, 1), (0), (1)	(1, 1), (0), (1)
GT	(1, 1), (0), (1)	(1, 1), (0), (1)
DGP	(1, 1, 0), (1, 0), (1)	(1, 1, 0), (1, 0), (1)
DGT	(1, 1, 0), (1, 0), (1)	(1, 1, 0), (1, 0), (1)
DPT	(1, 1, 0), (1, 0), (1)	(1, 1, 0), (1, 0), (1)
GPT	(1, 1, 0), (1, 0), (1)	(1, 1, 0), (1, 0), (1)
DGPT	(1, 1, 1, 0), (1, 1, 0), (1)	(1, 1, 1, 0), (1, 1, 0), (1)

8 Conclusion

In this paper, we illustrate the importance of test flow selection to reduce the expected total test time to produce each 3D Stacked IC. We propose a test flow selection algorithm (TFSA) to find the most suitable test flow for a given 3D Stacked IC. We evaluated the test flow obtained from TFSA, against three straightforward test flows and that obtained by exhaustive search. In the experiments we also compare the different test flows after integrating each with three test architecture design and test planning schemes. The experimental results demonstrate the importance to have methods both to find the test flow as well as test architecture design and test planning. It is observed that TFSA provides the optimal test flow, identical to exhaustive search, for all benchmarks. The test time can be further reduced by using both test architecture optimization and TFSA. For the benchmarks used in this paper, straightforward test flows like WSPT may perform as good as TFSA.

1) TFSA generates test flows and the corresponding test times very close to exhaustive search;
2) TFSA and a 3D Stacked IC optimized test architecture performs best with respect to test time, and
3) WSPT provides the minimum test time among the three straightforward test flow schemes and in many cases is equal to that with TFSA.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Agrawal M, Chakrabarty K (2015) Test-Cost Modeling and Optimal Test-Flow Selection of 3-D-Stacked ICs. In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp 1523–1536
2. Chou RM, Saluja KK, Agrawal VD (1997) Scheduling tests for VLSI systems under power constraints. IEEE Trans VLSI Syst 5(2):175–185
3. Hamdioui S, Taouil M (2011) Yield Improvement and Test Cost Optimization for 3D Stacked ICs. In: Asian Test Symposium (ATS), pp 480–485
4. Higgins M, MacNamee C, Mullane B (2010) Design and implementation challenges for adoption of the IEEE 1500 standard. IET Comput Digit Techn 4(1):38–49
5. Iyengar V, Chakrabarty K, Marinissen EJ (2002) Test Wrapper and Test Access Mechanism Co-Optimization for System-on-Chip. In: Journal of Electronic Testing: Theory and Applications, vol 18, pp 213–230
6. Iyengar V, Chakrabarty K, Marinissen EJ (2003) Test Access Mechanism Optimization, Test Scheduling, and Tester Data Volume Reduction for System-on-Chip. IEEE Trans Comput 52(12):1619–1632
7. Larsson E, Arvidsson K, Fujiwara H, Peng Z (2004) Efficient test solutions for core-based designs. IEEE Trans Comput-Aided Des Integr Circ Syst 23(5):758–775
8. Marinissen EJ, Zorian Y (2009) Testing 3D Chips Containing Through-Silicon Vias. In: IEEE International Test Conference (ITC), pp 1–11
9. Marinissen EJ, Verbree J, Konijnenburg M (2010) A Structured and Scalable Test Access Architecture for TSV-Based 3D Stacked ICs. In: IEEE VLSI Test Symposium (VTS), pp. 1–6
10. Marinissen EJ, McLaurin T, Jiao H (2016) Ieee std p1838: Dft standard-under-development for 2.5d-, 3d-, and 5.5d-sics. in: 2016 21th IEEE european test symposium (ETS), pp 1–10
11. Mullane B, Higgins M, MacNamee C (2008) IEEE 1500 Core wrapper optimization techniques and implementation. In: IEEE International test conference (ITC), no. 29.2, pp 1-10
12. SenGupta B, Larsson E (2014) Test Planning and Test Access Mechanism Design for Stacked Chips using ILP. In: IEEE VLSI Test Symposium (VTS), pp 1–6
13. SenGupta B, Ingelsson U, Larsson E (2012) Scheduling Tests for 3D Stacked Chips under Power Constraints. J Electron Test: Theory Appl (JETTA) 28(1):121–135
14. Taouil M, Hamdioui S (2012) Yield Improvement for 3D Wafer-to-Wafer Stacked Memories. In: Journal of Electronic Testing: Theory and Applications (JETTA), vol 28, pp 523–534
15. Yi H, Song J, Park S (2008) Low-Cost Scan Test for IEEE-1500-Based SoC. IEEE Trans Instrum Measur 57(5):1071–1078

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Breeta SenGupta received her M.Sc. degree from Indian Institute of Technology, Kharagpur, India, in 2009. She pursues her Ph.D. at the Department of Electrical and Information Technology, Lund University, Sweden. Her area of research includes 3D integration and testing. Breeta is currently engaged in development and verification of RF ASIC at Ericsson, Sweden.

Dimitar Nikolov received the diploma engineering degree from "SS. Cyril and Methodius" University, Skopje, Macedonia, in 2008, the licentiate degree from Linköping University, Linköping, Sweden, in 2012, and the PhD degree from Lund University, Lund, Sweden, in 2015. From 2015, he has been a postdoc fellow with Lund University. His current research interests include fault tolerance, design and test of digital systems, computer architectures, reconfigurable computing, and digital signal processing.

Assmitra Dash received his M.Sc. degree from the department of Computer and Information Science from Linköping University in 2013, and is currently working as a software engineer in SAP SE.

Erik Larsson received his M.Sc., Tech. Lic and Ph.D from Linköping University (LiU) in 1994, 1998, 2000, respectively. After a Post Doc (2001-2002) at Nara Institute of Science and Technology (NAIST), Japan, he was with LiU as an Assistant Professor (2002-2005) and Associate Professor (2006-2012). Since 2012 he is with Lund University, as Associate Professor (2012-2018) and since 2018 as Professor. His current research interests include test planning for manufacturing test, test during operation (in-situ), scan-chain diagnosis, silicon debug and validation, IJTAG/SJTAG, stacked 3D chip test, fault-tolerance for MPSoCs (Multi-Processor System-on-Chip), and property checking in distributed systems (MPSOcS with Network-on-Chip (NoC)). He has more than 160 publications in these areas. He received the Institution of Engineering and Technology (IET) Premium Award, 2009, and best paper awards at IEEE Asian Test Symposium (ATS), 2002 and at IEEE European Test Symposium (ETS), 2016.