On Maximizing the Compound Yield for 3D Wafer-to-Wafer Stacked ICs

Mottaqiallah Taouil[1] Said Hamdioui[1]
[1]Delft University of Technology
Faculty of EE, Mathematics and CS
Mekelweg 4, 2628 CD Delft, The Netherlands
{M.Taouil, S.Hamdioui}@tudelft.nl

Jouke Verbree[1,2] Erik Jan Marinissen[2]
[2]IMEC vzw
3D Integration Program
Kapeldreef 75, 3001 Leuven, Belgium
{jouke.verbree, erik.jan.marinissen}@imec.be

Abstract

Three-Dimensional Stacked IC (3D-SIC) is an emerging technology that provides heterogeneous integration, higher performance, and lower power consumption compared to planar ICs. Fabricating these 3D-SICs using Wafer-to-Wafer (W2W) stacking has several advantages including: high throughput, thin wafer and small die handling, and high TSV density. However, W2W stacking suffers from low compound yield. This paper investigates various matching processes by using different wafer matching criteria in order to maximize the compound yield. It first establishes a framework covering different matching processes and wafer matching criteria for both replenished and non-replenished wafer repositories. Thereafter, a subset of the framework is analyzed. The simulation results show that the compound yield not only depends on the number of stacked dies, die yield, and repository size, but it also strongly depends on the used matching process and the wafer matching criteria. Moreover, by choosing an appropriate wafer matching scenario (e.g., wafer matching process, criterion etc.), the compound yield can be improved up to 13.4% relative to random W2W stacking.

Keywords: *3D integration, wafer matching, matching criteria, compound yield, wafer-to-wafer stacking*

I. Introduction

The ability to create *Three-Dimensional Stacked Integrated Circuits (3D-SICs)* alleviates or even eliminates various existing problems in planar ICs. A 3D-SIC consists of multiple stacked planar dies, fabricated in a conventional process augmented by new Through Silicon Via (TSV) process steps, which electrically connect the planar wafers in the vertical direction. An efficient partitioning of IP cores among the stacked dies reduces the need for long wires and is thus able to reduce the wire delay, as well as the power dissipation [1], [2]. Heterogeneous integration is a promising concept for 3D-SICs, since each layer can be manufactured with different technology and optimized for speed or area. This affects the yield, performance, and lithography cost positively. Furthermore, miniaturization of the physical sizes of stacked dies reduces the footprint size and volume area, which in turn increases the package density. Examples of 3D-SICs include 3D CMOS sensors [3], 3D FPGAs [3], 3D processors [4], 3D cache and memory [5], [6], and combined stacks of memories and processors [3], [7].

Tiers are stacked at the die or the wafer level and can be stacked based on Wafer-to-Wafer (W2W), Die-to-Wafer (D2W) or Die-to-Die (D2D) bonding. In W2W bonding, complete wafers are stacked and bonded together. One of the benefits of W2W stacking is the high manufacturing throughput due to single wafer alignment [8]. High alignment accuracy can also be applied to D2W and D2D, but it negatively affects the throughput due to many dies that have to be aligned [8]. However, the yield loss for 3D-SICs is one of the major bottlenecks that must be overcome for 3D technology to make it a lucrative business [9]. The major limitations of W2W stacking is the rapid compound yield decrease, as the number of layers in the stack increases. The compound yield can be improved by *wafer matching*, initially introduced by Smith et al. [10]. In wafer matching, a software algorithm keeps track of the fault map of each wafer. The algorithm matches wafer pairs that contain the same or similar fault maps. This increases the 3D compound yield over randomly stacked wafers. More elaborated studies of wafer matching are presented in [11], [12]. Nevertheless, all the published work considered wafer matching with *static* repositories, i.e., after wafer selection, the repositories are not replenished unless they are empty. In addition, these papers focused *only* on matching of the *good dies* from the bottom layer with the *good dies* from the top layer. However, this could also be the matching of the *faulty* dies instead of the good dies.

In this paper, the impact of *replenished* repositories on the compound yield by using different wafer matching criteria is investigated. In this case, when a wafer is selected from a repository, its empty spot is directly replenished with a new one. This keeps the size of the *running* repository constant over time. The main contributions of this paper are:

- A new framework that covers all matching processes and wafer matching criteria for both *static* and *running* repositories.
- The illustration of the impact of several matching processes and wafer matching criteria on the compound yield of 3D-SICs.
- The demonstration of the impact of running repositories on the 3D-SIC compound yield.
- A comparison between the yield benefits gained from static and running repositories over random stacking.

The remainder of the paper is organized as follows. Section II provides an overview of the prior work in the area of wafer matching. Section III introduces the framework for wafer matching and defines the focus of this paper. Section IV describes the wafer matching scenarios to be experimented with in this work. Section V presents the simulation results and the comparison to the related work. Section VI concludes the paper.

II. Related Prior Work

Improving the yield for 3D circuits based on wafer matching was initially introduced by Smith et al. [10], where the authors compared the yield improvement of a single die SoC, by mapping it into a 3D-SIC with two equal sized layers. The yield improvement is both simulated for D2W and W2W stacking. In the W2W stacking case, a software matching algorithm is used to select pair-wise the best wafers from two repositories with a size of 25 each. The wafer fault map is based on a random generation.

The concept of W2W matching introduced by Smith [10] is further generalized by Reda et al. [11]. The paper formulates the W2W matching problem and proves it to be $\mathcal{NP}$-hard. Several matching processes and wafer matching algorithms are investigated, including the optimal hard one. In [12], Verbree et al. define a mathematical model for wafer matching; the model has some practical limitations, but nevertheless it gives a good indication of the yield improvements. The authors include wafer matching simulations for a greedy algorithm that address the limitations. In addition, the authors justify their pre-bond test cost required for wafer matching.

In [13], Ferri et al. used wafer matching to increase the *parametric yield* of a two layered D2W stacked 3D-SIC. Only functional dies are considered in this case to produce an optimal binning; i.e., maximize the fastest speed bins and minimize the slowest ones. Wafer matching is then used to combine and improve the 3D parametric yield by including the process variation of both layers in a D2W stacking approach. The authors were able to increase the number of 3D-SICs in the fastest speed bins as well as simultaneously reducing the number of slow 3D-SICs.

All the related previous work considered *static* repositories and used a *single* wafer matching criterion.

III. Wafer Matching Framework

As it has already been mentioned, W2W stacking provides the highest manufacturing throughput and is suitable for wafers with identical die sizes and/or small die sizes. However, it suffers from lower compound yield, as the stacking of bad dies on good dies and vice versa can not be avoided. *Wafer matching* can be performed on repositories of wafers in order to find out the best wafer combinations that would result in higher yield, given that the wafers were tested before the bonding. This section defines a framework for all possible *wafer matching scenarios* for 3D W2W stacked ICs; a wafer matching scenario combines different aspects at a time: (a) *Static or running repositories*, (b) *Wafer matching process*; e.g., how many wafer and/or layers are considered at each step, and (c) *Wafer matching criterion*; e.g., select the matching based on the good matched dies.

In the rest of the section, first the problem of W2W 3D-SICs is defined. Then, the aspects of wafer matching scenarios are addressed. Thereafter, the wafer matching framework is given.

A. W2W 3D-SIC Problem

The problem of W2W 3D-SICs can be defined as follows: Given, (a) n number of repositories each with k wafers, (b) fault maps for all the wafers (based on pre-bond testing), and (c) a production size of m 3D-SICs, the purpose is to maximize the overall compound yield for all m 3D-SICs, by selecting appropriate wafers for the n-layer 3D-SICs from the repositories. Figure 1 shows two freedom degrees to create 3D stacks. The vertical direction considers the wafers and the selection freedom here is the number of wafers that are selected to be stacked simultaneously; this can be either one wafer at a time (*Wafer-by-Wafer*) or k wafers at a time (*All-Wafers*). The horizontal direction shows the freedom selection from the number of layers that are considered simultaneously for stacking; this can be either two layers at a time (*Layer-by-Layer*) or n layers at a time (*All-Layers*).

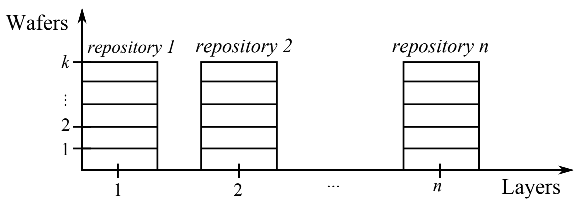


Fig. 1. Wafer vs layers in 3D stacking

B. Static Versus Running Repositories

Wafer matching can be considered as a time consuming process when the objective is to obtain the global compound yield for a production size m; m can be in the order of thousands or millions. To split up and divide the problem, a fixed number of k (usually $k << m$) wafers per repository can be considered and matched at a time. Depending on either a repository is replenished immediately (after a wafer is removed from it for matching and stacking) or not, two classes can be defined:

- *Static repositories*: From each repository k wafers are selected and processed before considering the next group of k wafers. The procedure stops after m/k steps.
- *Running repositories*. Each repository is immediately replenished with a new wafer each time a wafer is selected. The procedure stops after m wafers are processed.

The freedom to select wafers from static repositories reduces over time, since the repositories become more and more empty. Running repositories, however, provide always the full repository (of size k) to select from; this improves the effectiveness of wafer matching as compared with static repositories. The downside of running repositories is that unattractive wafers may remain in the repository for many iterations, occupying space, and in effect reducing the size of the repository in the long run. We call this effect, the repository *pollution*.

Another difference between static and running repositories is the actual implementation. Static repositories map fairly well onto a production line, where basically the repositories are the wafer containers that move from one machine in the production line to the next. With running repositories, a container would need to go back and forth between the bonding machine and the wafer production line to be replenished, before a new selection is made. Clearly, this is impractical, and therefore we suggest using two containers. One to select from, and one acts as a wafer source to replenish the first one at the bonding machine. This, however, reduces the effective capacity of the bonding machine as both containers are in the machine, yet only one is used to select a wafer from.

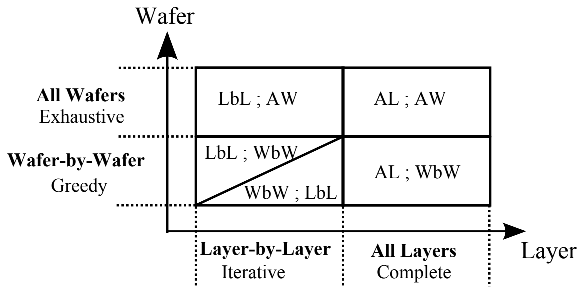


Fig. 2. Framework of matching processes

C. Matching Process

The matching process defines the step-by-step process to be followed in order to realize wafer matching. The matching process, therefore, determines the *number of repositories* and the *number of wafers* that are considered at a time.

Depending on the number of involved repositories (see also Figure 1), two cases are distinguished:

- *Layer-by-Layer (LbL)*: Initially, the first two repositories are selected for wafer matching. In each additional step, only one additional repository is used during matching. Hence, this is an *iterative* process in terms of the number of involved layers.
- *All-Layers (AL)*: In each step of the wafer matching process, all repositories are used at once. As every wafer in every repository is taken into account, this process is labeled *complete*.

In a similar way, depending on the number of wafers involved in each step of the matching process, two cases can be distinguished:

- *Wafer-by-Wafer (WbW)*: In each step of the wafer matching process, the best wafers contributing to the possible match are selected. Only one wafer from each repository is involved in the matching process, with no regard to the remaining wafers in those repositories. Thus, this process is regarded as *greedy*.
- *All-Wafers (AW)*: In each step of the wafer matching process, all wafers from all involved repositories are matched. As the process considers all possible outcomes for all k wafers to be matched, this process is considered to be *exhaustive*.

The above combinations result into five possible wafer matching processes, as shown in Figure 2.

- LbL;WbW: The matching process steps are iterative over the repositories. In each iteration step, only two repositories are considered. First, the best wafer pair for the first two repositories (each with k wafers) is selected. Then, the step is repeated $(k-1)$ times on these two repositories. Thereafter, the process is repeated on the rest of the repositories one by one.

Note that in each step, the size of the repositories are reduced by one.

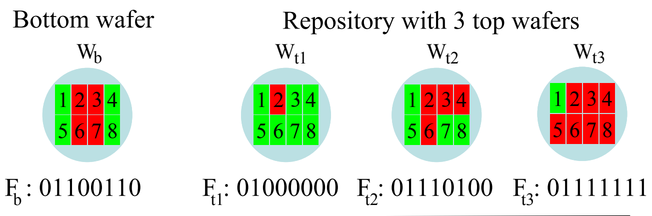
- WbW;LbL: The matching process steps are iterative over the wafers. In each step, a single wafer is selected iteratively from each repository to form the 3D-SIC. The difference between LbL;WbW and WbW;LbL is the reversed loop order of visiting the repositories and the wafer selections within a repository.
- LbL;AW: Similar to LbL;WbW, the matching process iteratively considers two repositories at a time, but in this case, all wafers from the two repositories under consideration are matched. Note that, this matching process is only applicable to static repositories, since running repositories are replenished, each time a wafer is selected from them. The difference between LbL;WbW and LbL;AW is that LbL;AW provides an *exhaustive* solution within the LbL process, while LbL;WbW selects the wafers one by one in a *greedy* way.
- AL;WbW: The matching process considers all repositories simultaneously in each matching step, and selects the best matching combination of n wafers along the repositories. The same step is repeated over time. In the case of static repositories, the matching of n wafers along the repositories is performed, first with k wafers and in the second step with $k-1$ wafers, etc. In case of running repositories, the matching considers always k wafers from each repository.
- AL;AW: This is similar to AL;WbW, but here, all k wafers from each repository are matched simultaneously. Note that, this matching process is only applicable to static repositories.

It is worth noting that for the LbL processes, an additional freedom can be defined for the traversal order for the repositories. The number of freedom possibilities to step over the repositories equals to $\binom{n}{2} \cdot (n-2)! = \frac{n!}{2}$; the first term of the equation represents the number of possibilities to select the first two repositories out of n, while the second term $(n-2)!$ presents the number of combinations of the remaining repositories.

D. Matching Criteria

The matching processes select wafers based on certain criteria; e.g., best good dies. Each criterion is orthogonal with respect to the process. Based on the fact that each wafer consists of both good and bad dies and that the purpose of the wafer matching is to maximize the compound yield, one can define three possible criteria: (a) maximize the matching good dies, (b) maximize the matching faulty dies, and (c) minimize the matching between good and bad dies. The criteria are defined as follows:

- *Max(MG)*. This criterion selects the best wafer pair combinations based on the maximum *Matched Good (MG)* dies. All the published work so far regarding wafer matching considers only this criterion.
- *Max(MF)*. The best wafer pair combinations is selected based on maximum *Matched Faulty (MF)* dies.
- *Min(UF)*. This criterion selects the best wafer pair combinations based on minimum *Unmatched Faulty (UF)* dies. The objective is to increase the compound yield by minimizing faulty dies that land on good dies and vice versa.



MG: matched goods	MG = 4	MG = 3	MG = 1
MF: matched faults	MF = 1	MF = 3	MF = 4
UF: unmatched faults	UF = 3	UF = 2	UF = 3

Fig. 3. Wafer matching criteria

All the above criteria produce the same result in terms of compound yield, in case the wafer matching process is exhaustive (AW process) for static repositories. For the greedy wafer matching processes (WbW), it is evident that different criteria lead to different results due to the greediness of the algorithm. For running repositories, the criteria lead to different compound yields, as will be explained next.

In order to provide more insight into the impact of the above criteria on wafer selection, refer to the example shown in Figure 3, which considers a bottom wafer W_b and three potential top wafers (W_{t1}, W_{t2}, W_{t3}), each with its own fault map. The fault map of each wafer is denoted by F and contains a sequence of 0s (good dies) and 1s (bad dies) ordered according to the indices of the dies on the wafer; e.g., the bottom wafer has $F_b = 01100110$, since the dies 2, 3, 6 and 7 are faulty. The bottom table in the figure lists the value of the different criteria for the three matching possibilities; e.g., for matching W_b-W_{t1}, the number of matched good dies is MG = 4 (which are dies 1, 4, 5, 8). The figure clearly shows that depending on the criterion, different top wafers will be selected; e.g., if max(MG) is considered, then W_{t1} will be selected. However, if the max(MF) is the criterion, then W_{t3} is the best match. The criteria can be mathematically formulated. Let the function $G(F_i)$ be the number of faulty dies in the wafer with fault map F_i. Then,

$$\begin{aligned} Max(MG) &= max(\forall_{i,j}, G(\bar{F_i} \& \bar{F_j})) & (1) \\ Max(UF) &= min(\forall_{i,j}, G(F_i \oplus F_j)) & (2) \\ Max(MF) &= max(\forall_{i,j}, G(F_i \& F_j)) & (3) \end{aligned}$$

Here, $0 \leq i, j \leq k$, where k the repository size.

TABLE I. W2W Matching Framework

Matching process	Static repository	Running repository
LbL;WbW	yes (Greedy [12])	n.a.
WbW;LbL	yes	yes
LbL;AW	yes (IMH [11])	n.a.
AL;WbW	yes (Greedy [11])	yes
AL;AW	yes (ILP/UB [11])	n.a.

n.a. denotes not applicable

E. The W2W matching framework

The wafer matching scenario aspects discussed in the previous section can be integrated into a complete framework that covers all wafer matching scenarios as shown in Table I. The table shows the possible combinations of matching processes and repository types (e.g., static and running repositories). Each combination results in a wafer matching scenario when combined with a matching criteria. The matching scenarios considered in the previous published work are represented by their references in the table. The criteria are left out, since they are independent of the matching processes. The table shows whether for each combination between the processes and the repository types, a valid combination exists ("yes" in the table) or not ("n.a"). Going vertically down in the entries of the table, more advanced algorithms are used which in general lead to a higher compound yield at the cost of higher computational effort. Putting the previous work in the context of the framework defined in Section III, the following can be concluded. The greedy algorithm in [12] is a LbL;WbW process. It creates a sorted list based on the compound yield of all wafer combinations between two repositories. From this list, valid pairs are selected starting from the highest yield. A combination is considered invalid when at least one of the wafers of the current compound has already been taken in a previous selection. After the repositories are empty, the repository of the next layer is matched with the current temporary stacks. In [11], three matching scenarios are described. In the first scenario, a greedy algorithm is used to create a sorted list of all k^n wafer combinations; this is in fact AL;WbW process. The difference with the greedy algorithm in [12] is that in this scenario all layers are considered at the same time. The second scenario, referred to as the Iterative Heuristic Matching (IHM) algorithm, considers two repositories at a time and optimally matches them by the Hungarian algorithm. These steps are iteratively repeated by including one additional repository in each iteration. The IHM algorithm is an LbL;AW process. In the third scenario, a global optimal algorithm based on Inter Linear Programming (ILP) is used to explore the exhaustive search space and obtain the global maximum yield. The execution time reduction of ILP scenario is realized by relaxing the ILP and allowing the program variables to take fractional values; this resulted into Upper Bound (UB) scenario. The ILP and UB scenarios are both AL;AW processes.

From Table I we conclude that several scenarios are not explored yet, mainly the ones for running repositories. This paper explores part of this space as will be explained in the next section.

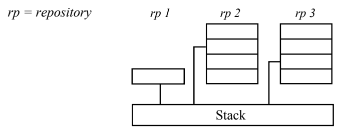


Fig. 4. Matching scenario FIFO1

IV. Scenarios for Running Repositories

The paper focuses on the impact of running repositories on the compound yield. Different wafer matching scenarios are considered based on the WbW;LbL matching process and different matching criteria. Due to space limitation and its low time and memory complexity, WbW;LbL is the only wafer matching processes considered in this paper.

As already explained, WbW;LbL process considers only two repositories at a time; in addition, only a single wafer pair selection is performed. Based on the wafer pairs selection order, three LbL;WbW matching processes can be defined:

- FIFO1-based WbW;LbL matching process.
- FIFOn-based WbW;LbL matching process.
- Best Pair-based WbW;LbL matching process.

Note that there are 9 matching scenarios, where 9 = 1 (running repository) · 3 (matching processes) · 3 (matching criteria). These are explained next.

A. FIFO1

In the FIFO1-based matching process the wafers from the first repository are selected based on a FIFO approach, as depicted in Figure 4 for $n = 3$. The wafers from repository 1 (rp1) are selected without any freedom and matched with the best wafer from the second repository. The process iterates over all the repositories. The size of the first repository is actually irrelevant, and can be changed to one. The order in which the repositories are traversed is linear starting at repository 1 and ending at repository n. Note that for FIFO1, the pollution is not critical for the first repository, since wafers are forced to get out. The runtime complexity of FIFO1 is $O(m \cdot k \cdot (n-1)) = O(m \cdot k \cdot n)$. The worst case memory complexity is $O(n)$; this is the memory required to store the list of indices holding the positions of the selected wafers from each repository.

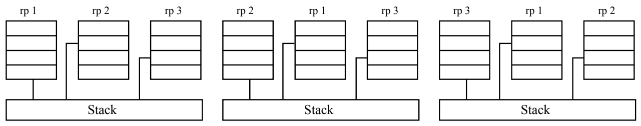


Fig. 5. Matching scenario FIFOn

B. FIFOn

In the FIFOn-based matching process, we generalize the concept of FIFO1. This is performed by moving the FIFO-repository in a round robin fashion among all repositories as shown in Figure 5 for $n = 3$. At the left side of the figure, repository one (rp1) is used as FIFO. After an n-compound stack is created, the repository belonging to the next layer is considered to be the FIFO as shown in the middle of the figure. Here, the algorithm starts from rp2 and proceeds next with rp1 and rp3; the traversal order is written in the top part of Figure 5. For the next compound, rp3 is used as FIFO. These steps are repeated until the production size is reached. The first traversed repository is the repository that is considered as FIFO, the remaining repositories are traversed in monolithic increasing order starting at repository 1 and ending at repository n. FIFOn is able to control the pollution since it forces wafers to stay maximally $n \cdot k$ cycles in a repository. In this way, the repositories are not contaminated with bad wafers that stay for a long time in the repositories without being selected. The memory and runtime complexity for this scenario are the same as in the case for FIFO1, since it only changes the position of the FIFO-repository.

C. Best Pair (BP)

In the BP-based matching process, the wafers from the first two repositories are matched in pairs without any selection restrictions; see Figure 6 for $n = 3$. The process iteratively proceeds along the repositories until a single n-compound match is determined. Then, this process is repeated until the production size m is met. The BP matching process has more freedom in wafer selection, as compared to FIFOn, but it lacks controlling the repository pollution. The runtime complexity equals to $O(m \cdot k \cdot n + k^2) = O(mkn)$. Initially, k^2 comparisons are performed on the initial set of the first two repositories. The best pair is selected and used to search for the best matching with the rest of repositories (one by one); this requires $(n-2)*k$ comparisons. Note that after replenishing, the process will be repeated; however, now the first two repositories require only $2*k-1$ comparisons rather than k^2 since the results of the previous comparison can be reused. The memory complexity is $O(k^2+n)$, required to store all k^2 compound yield combinations between the first repositories, and to hold a list of n numbers identifying the indices of the selected wafers of each repository.

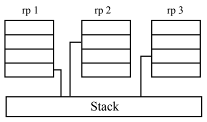


Fig. 6. Matching scenario BP

V. Simulation Results

This section presents the simulation results and analyzes the impact of the 9 wafer matching scenarios discussed in the previous section of the compound yield (i.e., FIFO1-based, FIFOn-based and BP-Based scenarios). Section V-A describes the experimental setup. Section V-B provides the impact of the running repositories, while V-C presents the impact on repository 'pollution'. Finally, the best wafer matching scenario will be selected and compared to related work in Section V-D.

A. Experimental Setup

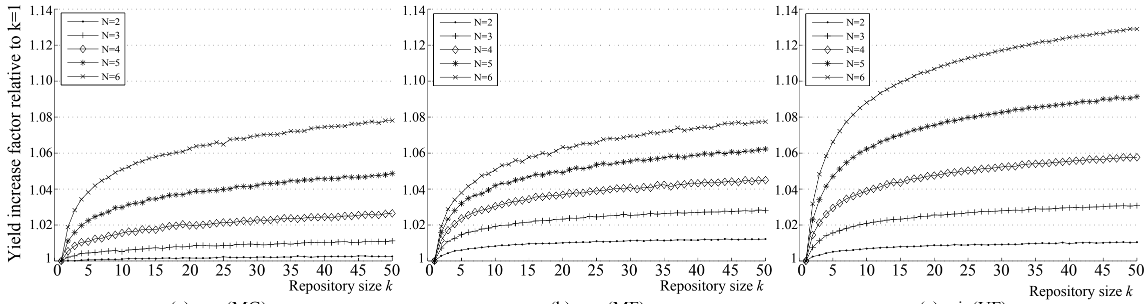
The experiments are based on the reference process in [12]. A standard 300 mm diameter wafer is selected with an edge clearance of 3 mm. The defect density is considered to be d_0 = 0.5 defects/cm^2 and the defect clustering parameter $\alpha = 0.5$. For the reference design, the die area is assumed to be A=$50mm^2$. For this die area and wafer size, the number of Gross Dies per Wafer (GDW) approximately equals to 1278 [15]. The expected yield of the wafers can be estimated by the negative binomial formula as: $y = (1 + \frac{A \cdot d_0}{\alpha})^{-\alpha}$= 81.65% [16].

In our experiments, we simulate a production size $m = 25000$. Here, m is the number of produced 3D-SICs. Initially, each repository is filled up with k wafers and after selecting and stacking m-compound wafers, the wafers that are left in the repository are discarded and not included in the simulation results for two reasons.

1) First, we want to observe the impact of the running repository only.
2) Second, even if the wafers would be thrown away, their impact on the compound yield is minimal (k/m), due to a high production volume m. Actually, the matching scenarios presented in [11] and [12] for static repositories could be used to match these last k unconsidered wafers.

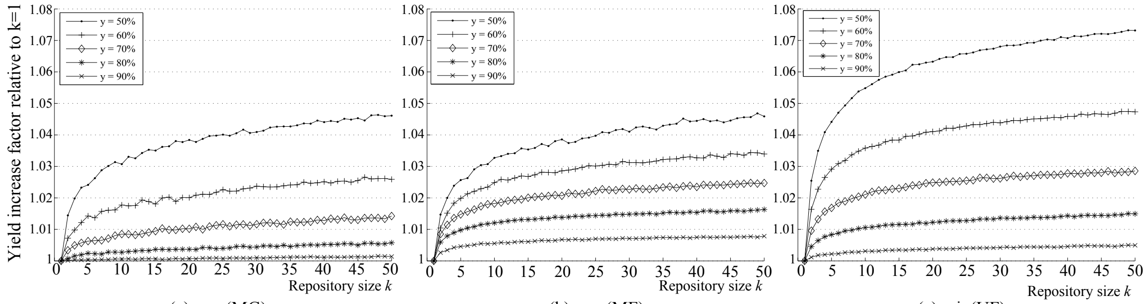
To measure the impact of running repositories on the compound yield (while considering repository size, wafer yield and matching criteria) as well as repository pollution, three experiments are performed:

- In experiment 1, the impact on the compound yield for different stacked number of layers n ($2 \leq n \leq 6$) for various repository sizes is examined. The reference



(a) max(MG) (b) max(MF) (c) min(UF)

Fig. 7. Impact of n and k on compound yield for FIFO1 using the reference process



(a) max(MG) (b) max(MF) (c) min(UF)

Fig. 8. Impact of wafer yield and k on the compound yield for FIFO1

process is considered and all criteria are simulated for each scenario.

- In experiment 2, we adjust the wafer yield of the reference process over a wide range to simulate the impact of the compound yield on stacked 3D-SICs. We consider a stack of two layers and vary the repository size.
- The last experiment consists of indirect measurement of the repository pollution. By plotting the compound yield for different stack sizes versus different production sizes m, we can indirectly measure the pollution that takes place and observe the effect on the compound yield. Moreover, we look at the compound yield differences between FIFO1 and FIFOn.

B. Impact of Running Repositories

Figure 7 plots the relative compound yield increase with respect to random stacking (i.e., $k = 1$) for different stacked layers n and repository sizes k for each criteria, while Figure 8 plots the relative compound yield increase with respect to wafer yield. Due to space limitation, only the simulation results for FIFO1-based matching processes are presented here. The figures clearly show that the relative compound yield increases with larger repositories and lower wafer yield, but the obtained gain stabilizes as the size of k becomes larger; the trends are similar for all criteria. It is worth noting that FIFOn-based and BP-based show similar trends as FIFO1-based wafer matching processes.

Let's now examine the impact of the matching criteria on the compound yield. Figure 7 shows that the criteria min(UF) outperforms the other two criteria for $n \geq 3$. On the other hand, Figure 8 indicates that min(UF) performs the best for wafer yields in the range of 50%-70%, while the criteria max(MF) performs the best for higher wafer yield (80% and above).

To obtain a better picture of the impact of the criteria on the compound yield, simulation for all scenarios (FIFO1, FIFOn and BP) and different criteria for a fixed repository size of $k = 50$ has been performed. Figure 9 and 10 show the results. One can conclude the following:

- In general, a higher improvement can be gained for larger stack sizes and lower wafer yield. Note that, when the stack size increases, the compound yield decreases.
- FIFOn always performs better than FIFO1 for the same conditions, especially for the criteria that relatively perform poor. This difference in performance is minimal for min(UF) criterion; this means that in this case a small pollution is taking place (see next section).
- Overall, BP scenario scores the best in terms of compound yield. Depending on the value of the wafer yield y, BP has to be combined with appropriate criterion. For wafer yield $50 \leq y \leq 70$, BP combined with Min(UF) scores the best, while for $y \geq 80$ BP combined with Max(MF) scores the best.

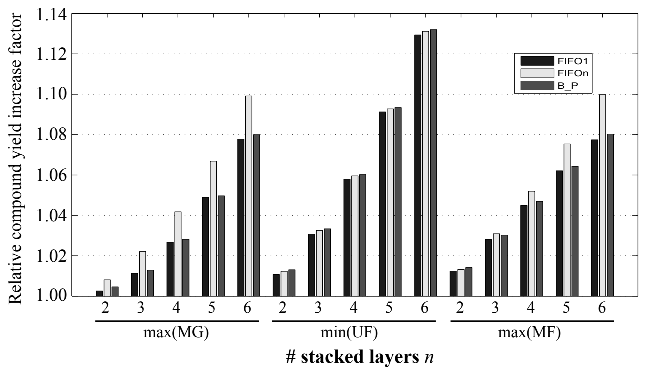


Fig. 9. Yield increase with variable n

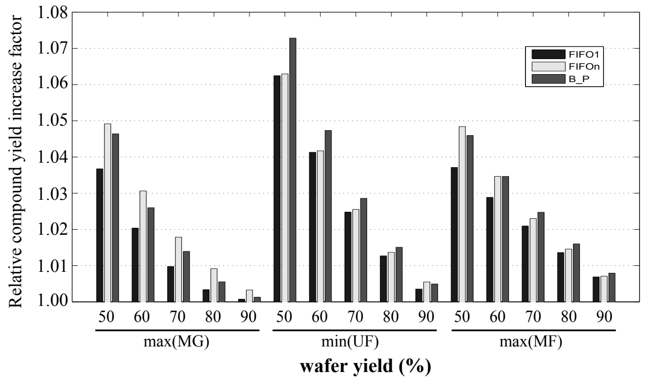


Fig. 10. Yield versus wafer yield

The above results clearly show that BP scenario outperforms both FIFO1 and FIFOn. The question is now which matching criterion has to be combined with -for a certain process- to maximize the compound yield. Table II answers this question. The table shows the criteria for different top wafer yield y_t and bottom wafer yield y_b that have be selected to achieve the highest compound yield. From the table one can conclude the following:

- When the wafer yield is low, the Max(MG) criterion should be selected. Max(MG) tries to match the good dies only and since these are in minority, the choice to select the best matching is relatively easy.
- For wafer yield in midrange values, the Min(UF) criterion performs the best. In this case, the probability of the presence of good and bad dies is similar.
- For very high wafer yield, it is most advantageous to select the Max(MF) criterion. In this case, the matching is based on faulty dies. As the faulty dies are in minority due to a high wafer yield, an overall highest compound yield is obtained if the matching of the minority dies is maximized.

The above clearly shows that an *adaptive* BP-based wafer matching is the best approach to realize the maximal overall compound yield. Table II can be used as a decision rule for the matching criterion selection. Each time a new wafer has to be selected for stacking, the table determines

TABLE II. Yield Based Criterion Selection

$y_t \backslash y_b$	10	20	30	40	50	60	70	80	90
10	MG	MG	MG	UF	UF	UF	MG	MG	MG
20	MG	MG	UF	UF	UF	UF	UF	MF	MF
30	MG	UF	UF	UF	UF	UF	UF	UF	MF
40	UF	UF	UF	UF	UF	UF	UF	UF	UF
50	UF	UF	UF	UF	UF	UF	UF	UF	UF
60	UF	UF	UF	UF	UF	UF	UF	UF	UF
70	MG	UF	UF	UF	UF	UF	UF	UF	MF
80	MG	MG	UF	UF	UF	UF	UF	MF	MF
90	MF	MF	MF	UF	UF	UF	MF	MF	MF

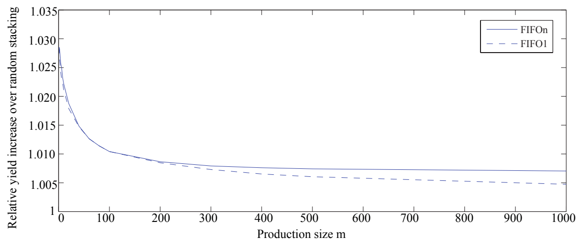


Fig. 11. Yield versus production sizes.

the matching criterion to be used. As an example, consider a three layered stack with equal wafer yield of 80%. According to Table II, the matching of the bottom and middle wafers is performed best using the Max(MF) criterion. If we assume now that the compound yield of this two-stacked IC is 70%, then the matching with the third layer can be best performed based on the min(UF) criterion. This adaptive BP scenario always results in the highest yield for all simulation parameters. From now on, we refer to adaptive BP as the matching scenario that adapts itself with respect to the criterion selection. In the next section, this adaptive scenario is used for comparison with the related work.

C. Repository pollution

In order to estimate the repository pollution, the compound yield for different production sizes is simulated. FIFO1 and FIFOn scenarios are considered for this experiment because: (a) they have the same complexity and (b) FIFOn forces the wafers to leave the repositories while FIFO1 does this only for one repository. Comparing these two scenarios will provide us with an idea about the impact of repository pollution on the overall compound yield.

Figure 11 plots the relative compound yield for the FIFO1- and FIFOn-based matching processes over random stacking for different production wafer sizes m. Here, the reference process is used with $n = 2$, $k = 25$ and the matching criterion max(MG). Three observations can be made from the graph:

- The relative yield for both FIFO1 as well as FIFOn decreases with increasing production size. For low m (typically below 200), the yield for both scenarios are almost the same.

- As the size of m increases (hence probability of having bad wafers increases), the difference in yield between FIFOn and FIFO1 becomes more visible. The compound yield of FIFO1 decreases faster than that of FIFOn; FIFOn forces wafers to leave the repository at most after $k \cdot n$ cycles and this has a positive affect on the yield.
- The yield degradation due to pollution is stabilizing for larger m.

It can be concluded that in order to minimize the pollution and improve the overall compound yield, it is important to implement a mechanism to force the wafers to leave the repositories after a certain time period.

D. Comparison of Matching Scenarios

In this section, we compare our adaptive BP matching scenario with the scenarios of static repositories published in [11]. We reproduce the same experiments as in [11]; we compare the optimal UB scenario and when this scenario is inapplicable due to memory limitations, the IMH scenario is used [11]. It is worth noting that in case of the optimal UB, different wafer matching criteria will lead to the same compound yield and thus they are not able to enhance the compound yield further.

Table III and IV show these differences for n=3, k=25 with 590 dies per wafer. In each table, the first column provides the varied parameter of the simulation (i.e., stacked number of layers n and the wafer yield y); the second column reports the compound yield of the related work; the third column presents the compound yield of the adaptive BP scenario; the fourth column shows the relative improvement of the BP algorithm versus the obtained yield of the related work; finally, the last column shows the improvement of the BP scenario relative to random stacking. From the tables, we can clearly conclude that running repositories lead to a higher compound yield than static repositories. Although the yield improvement is small, the time complexity difference is huge as summarized in Table V; the table also gives an overview over the memory and runtime complexity cost for each wafer matching scenario. For example, the optimal static algorithm in [11] implemented in C++, requires 0.392 seconds to solve an instance for $n = 3$ and 40.64 seconds for $n = 4$ and runs out of memory for larger number of stacked layers [11]. For the same parameters, our adaptive BP scenario implemented in Matlab required only 0.0028 seconds while using a negligible amount of memory to match a single compound for $n = 7$.

It is important noting that using of wafer matching requires *pre-bond testing*. Hence, it is worth to examine the additional costs required for pre-bond testing. We compare

TABLE III. Yield Comparison with [11] for $n = 3$, $k = 25$, $d = 590$

yield	UB [11] (%)	BP (%)	$\frac{\text{BP}}{\text{UB [11]}}$ (%)	$\frac{\text{BP(k=25)}}{\text{random}}$ (%)
0.3	04.24	04.30	1.42	59.26
0.5	15.08	15.24	1.06	21.92
0.7	37.29	37.46	0.46	9.21
0.9	74.41	74.46	0.07	2.14

TABLE IV. Yield Comparison with [11] for $y = 80\%$, $k = 25$, $d = 590$

n	Alg. [11]	3D Yield [11] (%)	BP (%)	$\frac{\text{BP}}{\text{Alg. [11]}}$ (%)	$\frac{\text{BP(k=25)}}{\text{random}}$ (%)
2	UB	65.25	65.32	0.11	2.06
3	UB	53.56	53.76	0.37	5.00
4	UB	44.58	44.63	0.11	8.96
5	IMH	36.61	37.28	1.83	13.77
6	IMH	30.68	31.29	1.99	19.36
7	IMH	25.76	26.35	2.29	25.65

TABLE V. Wafer Matching Complexity

Ref	Scenario	Memory complexity	Runtime complexity
[11]	Greedy	$O((n+1) \cdot k^n)$	$O(m \cdot k^{n-1} \cdot log(k))$
[11]	IMH	$O(k^2)$	$O(m \cdot n^2 \cdot k^2)$
[11]	ILP/UB	$O((n+1) \cdot k^n)$	$O(\frac{m}{k} \cdot (k!)^{n-1})^*$
[12]	Greedy	$O(k^2)$	$O(m \cdot k^2 \cdot n)$
Ours	Fifo1	$O(n)$	$O(m \cdot k \cdot n)$
Ours	Fifon	$O(n)$	$O(m \cdot k \cdot n)$
Ours	Best Pair	$O(k^2 + n)$	$O(m \cdot k \cdot n)$

* denotes the complexity of the search space

three test flows depicted in Figure 12(a) [12]; they consist of three test moments: Die Test (i.e., pre-bond testing) on the wafers, Stack Tests to verify the stacked wafers before they are packaged and bonded; and Final Tests to ensure overall chip functionality.

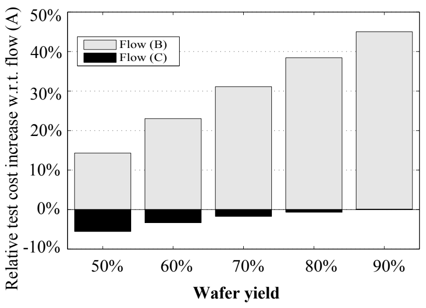
The work in [12] assumes a stack pass yield of 99% and an interconnect yield of 97%. Further, the wafer yield is assumed to be 81.65%, as for our reference process. The three test flows are:

- Flow (A) includes a stacking test and a final test, but has *no pre-bond* die tests. This flow is applicable for random W2W stacking.
- Flow (B) consists of pre-bond die tests (required for wafer matching), a stacking test that tests both dies and interconnects, and a final test.
- Flow (C) consists of pre-bond die tests, a stacking test for the interconnects only and a final test. The idea behind this flow is to optimize the wafer test flow (B) by not replicating the die test in the stacking test. As a consequence of faults introduced into the dies during stacking, a small percentage of faulty dies is still packed.

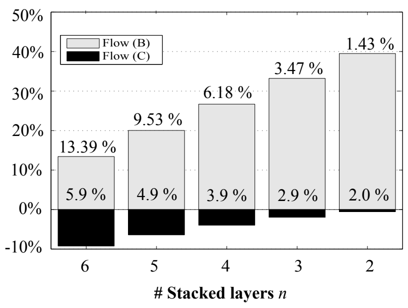
The test cost per functional good stack in terms of test time for the test flows (B) and (C) relative to flow(A) are shown in Figure 12(b) and 12(c); the heights of the bars present this relative cost. The absolute number variation from 2.0-5.9% within the gray bar in Figure 12(c) presents the percentage of faulty packaged 3D-SICs and

Test moment	Flow (A)	Flow (B)	Flow (C)
Die Test	—	✓	✓
Stack Test			
- TSV	✓	✓	✓
- Dies	✓	✓	—
Final Test	✓	✓	✓

(a) Test moments



(b) Test impact on yield



(c) Test impact on # layers

Fig. 12. Normalized cost of test flows (B) and (C) relative to the random W2W stack flow (A)

depends on the stack size n. For Figure 12(b), this package waste is equal to 2.0% for all different yields [12]. The numbers on top of the bars in Figure 12(c) present the yield gain relative to random stacking. Note that these are independent of the test flow.

Relatively to test cost of Test Flow (A), which has no pre-bond die tests, Test Flow (B) negatively effects the test cost, while test flow (C) is able to reduce the test cost per functional good stack. For example, in Figure 12(c), for a two-stacked 3D-SIC, the test time reduction is 0.55 %, the yield is increased with 1.43%, while the packaging cost is increased with 2%. For a six-layered stack, a test cost reduction of 9.23% is expected with a yield increase of 13.39%, but with a package cost increase of only 5.9%. Since the compound yield for running repositories is higher than that of static repositories (while the test costs are the same), we can conclude that the test time per functional working die is lower than in the case of static repositories.

VI. Conclusion

This paper investigates the impact of running repositories on the compound yield for 3D-ICs based on wafer-to-wafer (W2W) stacking. It first introduces a framework for 3D W2W matching, which consists of several wafer matching scenarios. Each scenario is a combination of a matching process, wafer matching criterion, and a repository type (e.g., running or static repositories). The framework shows several scenarios that are not explored yet and a subset of it was selected for further investigation.

Nine wafer matching scenarios have been analyzed based on running repositories. The simulation results showed that the compound yield not only depends on the wafer yield and the number of stacked layers, but also strongly depends on the selected wafer matching scenario. By merging the best performing criteria into the best wafer matching process, an adaptive matching scenario is created that provides the best solution at runtime. By using the adaptive wafer matching scenario, we were able to improve the compound yield up to 13.39% relative to random stacking for realistic wafer yield. Moreover, the adaptive approach outperforms the compound yield of all wafer matching scenarios based on static repositories at a lower cost in terms of the test time, the required memory and time complexity.

References

[1] W. R. Davis et al., "Demystifying 3D ICs: The Pros and Cons of Going Vertical", *IEEE Design Test on Computers*, Vol 22, Issue 8, pp. 498-510, Nov 2005.

[2] J. A. Davis et al. "Interconnect Limits on Gigascale Integration (GSI) in the 21st Century", *Proc. IEEE*, Vol 89, Issue 3, pp. 305-224, 2001.

[3] R. S. Patti, "Three-Dimensional Integrated Circuits and the Future of System-on-Chip Designs", *Proceedings of the IEEE*, Vol 94, Issue 6, June 2006.

[4] G. Loh et al. "Processor Design in 3D Die-Stacking Technologies", *IEEE Micro*, Vol 27, Issue 3, pp. 31-48, Aug. 2007.

[5] K. Puttaswamy et al. "3D-Integrated SRAM Components for High-Performance Microprocessors", *IEEE Transactions on Computers*, Vol 58, Issue 10, pp. 1369-1381, Aug. 2009.

[6] Y-F Tsai et al. "Design Space Exploration for 3-D Cache", *IEEE Transactions on Very Large Scale Integration Systems*, Vol 16, Issue 4, pp. 444-455, 2008.

[7] F. Li et al. "Design and Management of 3D Chip Multiprocessors Using Network-in-Memory", *International Symposium on Computer Architecture*, pp. 130-141, July 2006.

[8] P. Garrou, C. Bower and P. Ramm, "Handbook of 3D Integration", Wiley-VCH, 2008.

[9] J. Baliga, "Chips Go Vertical", *IEEE Spectrum*, Vol 41, Issue 3, pp. 43-47, March 2004.

[10] L. Smith, G. Smith, S. Hosali, and S. Arkalgud, "Yield Considerations in the Choice of 3D Technology", *In Proc. IEEE Int. Symp. Semiconductor Manufacturing*, pp. 535-537, 2007.

[11] S. Reda et al. "Maximizing the Functional Yield of Wafer-to-Wafer 3-D Integration", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol 17 , Issue: 9, pp. 1357 - 1362, 2010.

[12] J. Verbree et al. "On the Cost-Effectiveness of Matching Repositories of Pre-Tested Wafers for Wafer-to-wafer 3D Chip Stacking", Paper accepted in: *IEEE European Test Symposyum*, May 2010.

[13] C. Ferri et al. "Parametric Yield Management for 3D ICs: Models and Strategies for Improvement", *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, Vol 4, Issue 4, Oct. 2008.

[14] E. J. Marinissen and Y. Zorian, "Testing 3D Chips Containing Through-Silicon Vias", *International Test Conference, 2009*, Nov. 2009.

[15] D. K. de Vries, "Investigation of Gross Die Per Wafer Formulas", *IEEE Transactions on Semiconductor Manufacturing*, Vol 18, Issue 1, pp. 136-139, Feb. 2005.

[16] M. Bushnell and V. Agrawal, "Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits", Wiley-VCH, Weinheim, Germany, Aug. 2000.