

amount of clashing is needed to get the maximum benefit from ODPs, which increases as the density of nonzero digits increases, which comes from increasing k . However, as the bit width increases, an increment in k increases the density of nonzero digits to a lesser extent. $H(1) + \text{ODP}$ outperforms $H(4)$ up to 19 b, after which it outperforms $H(3)$ up to 28 b. Above 32 b, it no longer outperforms $H(2)$. $H(2) + \text{ODP}$ outperforms $H(4)$, as $H(4)$ at 48 b will likely need more adders than $H(2) + \text{ODP}$. $H(3) + \text{ODP}$ always outperforms $H(2) + \text{ODP}$ but provides less improvement in run time over $H(4)$.

VIII. CONCLUSION

We proposed and integrated ODPs into $H(k)$, the best existing heuristic SCM algorithm, which significantly reduced its run time while still improving its performance. Consider a custom hardware design with 100 32-b SCMs. $H(4)$ needs 35 min whereas $H(2) + \text{ODP}$ generally produces better solutions in 2 min. ODPs enable the Hartley algorithm to more effectively search for patterns, thus we can search fewer SD forms (reducing the run time) and still produce solutions with fewer adders.

REFERENCES

- [1] A. G. Dempster and M. D. Macleod, "Using all signed-digit representations to design single integer multipliers using subexpression elimination," in *IEEE Proc. Int. Symp. Circuits Syst.*, 2004, pp. III165–III168.
- [2] R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 43, no. 10, pp. 677–688, Oct. 1996.
- [3] O. Gustafsson, A. G. Dempster, and L. Wanhammar, "Extended results for minimum-adder constant integer multipliers," in *IEEE Proc. Int. Symp. Circuits Syst.*, 2002, pp. 173–176.
- [4] A. Avizienis, "Signed-digit representation for fast parallel arithmetic," *IRE Trans. Electr. Comput.*, vol. 10, pp. 389–400, Sep. 1961.
- [5] I. C. Park and H. J. Kang, "Digital filter synthesis based on an algorithm to generate all minimal signed digit representations," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 12, pp. 1525–1529, Dec. 2002.
- [6] A. G. Dempster and M. D. Macleod, "Generation of signed-digit representations for integer multiplication," *IEEE Signal Process. Lett.*, vol. 11, no. 8, pp. 663–665, Aug. 2004.
- [7] O. Gustafsson, A. G. Dempster, K. Johansson, M. D. Macleod, and L. Wanhammar, "Simplified design of constant coefficient multipliers," *Circuits Syst. Signal Process.*, vol. 25, no. 2, pp. 225–251, 2006.
- [8] Y. Voronenko and M. Püschel, "Multiplierless multiple constant multiplication," *ACM Trans. Algorithms*, vol. 3, no. 2, pp. 1–38, May 2007, article 11.

Maximizing the Functional Yield of Wafer-to-Wafer 3-D Integration

Sherief Reda, Gregory Smith, and Larry Smith

Abstract—Three-dimensional integrated circuit technology with through-silicon vias offers many advantages, including improved form factor, increased circuit performance, robust heterogenous integration, and reduced costs. Wafer-to-wafer integration supports the highest possible density of through-silicon vias and highest throughput; however, in contrast to die-to-wafer integration, it does not benefit from the ability to bond only tested and diced good die. In wafer-to-wafer integration, wafers are entirely bonded together, which can unintentionally integrate a bad die from one wafer to a good die from another wafer reducing the yield. In this paper, we propose solutions that maximize the yield of wafer-to-wafer 3-D integration, assuming that the individual die can be tested on the wafers before bonding. We exploit some of the available flexibility in the integration process, and propose wafer assignment algorithms that maximize the number of good 3-D ICs. Our algorithms range from scalable, fast heuristics to optimal methods that exactly maximize the yield of wafer-to-wafer 3-D integration. Using realistic defect models and yield simulations, we demonstrate the effectiveness of our methods up to large numbers of wafer stacks. Our results demonstrate that it is possible to significantly improve the yield in comparison to yield-oblivious wafer assignment methods.

Index Terms—3-D IC, wafer to wafer integration, yield.

I. INTRODUCTION

Three-dimensional integrated circuit (IC) technology with through silicon vias (TSVs) is a new technology that allows the vertical stacking and interconnecting of multiple die into one 3-D IC [10], [14]. There are a number of benefits and motivations for developing 3-D ICs, including 1) a better form factor realized from the increased density from vertical integration [3]; 2) increased performance due to the improvement in interconnect delay because of short TSV length; 3) heterogenous integration where different functional die, such as memory, logic and sensors, are fabricated separately and then integrated together; and, finally, 4) cost, as 3-D technology might offer an alternative cheaper path to increase semiconductor integration without the need to resort to prohibitively expensive 2-D lithographic geometric shrinking. Examples of 3-D ICs include 3-D sensors, 3-D memory (Flash or DRAM), 3-D processors and 3-D FPGAs.

There are a number of integration methods used in 3-D IC fabrication: *wafer-to-wafer* (WTW), *die-to-wafer* (DTW), and *die-to-die* (DTD). These methods play an important role in determining the final yield of 3-D ICs [1], [2], [10], [14]. In wafer-to-wafer integration, entire wafers are directly bonded together. WTW offers the highest throughput, and allows for the thinnest wafers. Since the minimum TSV diameter is limited by the via's aspect ratio, WTW supports TSVs with the smallest via diameters, as it has the thinnest wafers, which in turn allows for greater TSV density. However, WTW can incur a serious yield loss as there is no way to separate the good die in advance. With WTW integration, a bad die from one wafer can end up integrated with a good die in another wafer yielding an overall bad

Manuscript received March 23, 2008; revised June 30, 2008. First published March 10, 2009; current version published August 19, 2009. S. Reda was supported by Qualcomm Corporation.

S. Reda is with the Division of Engineering, Brown University, Providence, RI 02912 USA (e-mail: sherief_reda@brown.edu).

G. Smith and L. Smith are with SEMATECH, Austin, TX 78714 USA (e-mail: gregory.smith@sematech.org; larry.smith@sematech.org).

Digital Object Identifier 10.1109/TVLSI.2008.2003513

3-D IC, die-to-wafer and die-to-die integration can improve the yield of 3-D ICs as they allow the die to be diced and tested in advance and use only the good ones during the 3-D integration process. DTD and DTW also allow the use of different wafer and die sizes. This flexibility, however, comes at additional test and bonding costs [11], lower throughput and lower TSV density. Yield loss can be mitigated through the use of redundancy as in the case of 3-D DRAM ICs, or 3-D multicore processors [10]. Furthermore, some applications, especially in high-end systems, require a small pitch that is only attainable through WTW, irrespective of the yield.

The objective of this paper is to develop techniques that improve the yield of WTW integration. As a wafer lot typically contains many wafers (typically 25), one way to improve the yield of WTW integration is to first test the wafers in the different wafer lots, and then match the wafers together during integration so as to increase the number of good 3-D ICs. Fundamentally, we should match wafers from different lots to reduce (or avoid at best) the chance that a good die from one wafer ends up integrated with a bad die from another wafer. In this paper, we thoroughly investigate this flexibility and develop optimal methods that maximize the yield of WTW 3-D integration. The contributions of this paper are as follows.

- We formulate the yield maximization problem in wafer-to-wafer 3-D integration technology. We provide hardness results for this problem and show special cases where it can be solved optimally in polynomial time.
- We propose a number of effective heuristic and optimal solutions to solve the problem. Our algorithms offer a graceful tradeoff in terms of quality of results as measured by yield and scalability as measured by runtime and memory requirements.
- Using realistic defect models and yield analysis simulations, and we provide comprehensive experimental results that demonstrate the effectiveness of our proposed algorithms in improving the yield of wafer-to-wafer 3-D integration for large numbers of wafer stacks.
- Our results demonstrate that our proposed optimal integration techniques can improve the yield (reaching up to 25%) in comparison to yield-oblivious integration strategies.

The organization of this paper is as follows. Section II provides a brief overview of the related research. In Section III, we formulate the main problem of maximizing the yield of wafer-to-wafer integration and propose a number of solutions. Section IV provides a comprehensive set of experimental results and conclusions that demonstrate the effectiveness of our proposed approaches.

II. PREVIOUS WORK

Despite the importance of the yield on the cost-effectiveness of 3-D technology [1], [10], there are few works that directly address the yield problem [1], [5], [10], [11], [12]. Yield loss in WTW integration can happen either due to defects in the individual wafers that constitute the stack, or defects that result from the 3-D integraton process (e.g., during TSV creation or bonding). The defects that impact the individual wafers result from typical random defect mechanisms that impact 2-D ICs. Generally, the larger the die area, the larger the chance it includes one or more defects; thus, wafers with large die printed on them will have a lower die yield than wafers with small die. If two types of wafers are made in the same fabrication process then they are subject to the same defect density. If the wafers are made with different fabrication processes, a possibility with 3-D ICs, then they are likely to have different defect density. Defects impacting different wafers are typically uncorrelated, and the modeling of such defects have been researched to maturity in the past [7], [8], [13]. For example, the negative binomial distribution [13] is typically used as a good model for the distribution of defects on semiconductor wafers.

To address yield loss in 3-D ICs, a few techniques have been so far proposed. Patti [10] suggests incorporating redundant resources into the 3-D IC to make potential stacked devices (such as memories and FPGAs) repairable in the presence of defects. More recently, Ferri *et al.* [5] suggest improving the parametric yield of DTW and DTD integration by carefully matching the speed of the die that are integrated in the 3-D stack, and Smith *et al.* [12] suggest matching the wafers in WTW integration to improve the yield. Finally, Smith *et al.* [11] investigate the implications of 3-D IC yield on the cost of WTW, DTW, and DTD integration methodologies.

III. PROBLEM FORMULATION AND PROPOSED SOLUTIONS

The defect wafer map of some wafer W_i can be represented as a string of 0s and 1s where a 1 indicates a good die and a 0 indicates a bad die. Let $G(\cdot)$ be a function that returns the number of good die in a given wafer map. $G(W_i)$ basically counts the number of 1s in the wafer map string W_i . If two wafers with maps W_i and W_j take part in a 3-D integration stack, then the wafer map of the resultant stack is $W_i \cdot W_j$, which is formed by bitwise ANDing of the strings W_i and W_j . A wafer lot is a batch of a number of wafers. Let L_i denotes the set of wafer maps that belong to wafer lot i . The problem of yield maximization in wafer-to-wafer 3-D integration can be formulated as follows.

Functional Yield Maximization in Wafer-to-Wafer 3-D Integration. Given K wafer lots $\{L_1, \dots, L_K\}$, where each lot consists of $|L_i| = \{W_1^i, \dots, W_N^i\} = N$ wafer maps, find an assignment function that 1) assigns each wafer map to exactly one 3-D wafer stack (that is composed of K wafers) and 2) maximizes the functional yield as measured by the total number of good 3-D ICs resulting from the N 3-D wafer stacks.

Note that the list of wafer maps that compose a 3-D stack can be represented by a tuple $(W_{i_1}^1, W_{i_2}^2, \dots, W_{i_K}^K)$. There are N^K possible wafer tuples, and there are $(N!)^{K-1}$ ways to choose N tuples from the possible N^K tuples without repetition. Solving the functional yield maximization problem amounts to finding the N tuples that maximize the total functional yield such that each wafer participates in exactly one tuple.

It is easy to show that for the general case of $K \geq 3$, the classical NP-hard 3-D matching problem (one of the original six NP-hard problems considered by Garey and Johnson [6]) is reducible to the functional yield maximization problem. While this result diminishes the possibility of finding optimal solutions for increasing N and K in a feasible runtime, we will later show that it is possible to obtain optimal solutions for $K = 2$ in polynomial time, and we will demonstrate in the experimental results section (Section IV) optimal results for up to $K = 4$ wafer stacks. The hardness result also points out the importance of developing heuristic solutions that scale in performance, runtime and memory requirements for general values of N and K .

A. Greedy Heuristic

As discussed earlier, there are N^K possible different 3-D integration stacks. In an attempt to find the best N wafer stacks that maximize the total yield, it is possible to devise a greedy heuristic to solve the yield maximization problem. A greedy heuristic first forms a list of all possible N^K wafer stacks. Then, for every wafer stack, the heuristic calculates the number of resultant good 3-D ICs after taking into account the distribution of good die on each wafer as given by the wafers' defect maps. The heuristic then sorts the list in descending order according to the number of good 3-D ICs of each stack. The list is then traversed in order where a wafer stack is chosen as long as none of its constituent wafers participated in an earlier chosen wafer stack. Fig. 2 gives a summary of the greedy algorithm. Note that the runtime complexity of the algorithm is equal to $O(K N^K \log N)$, and the memory

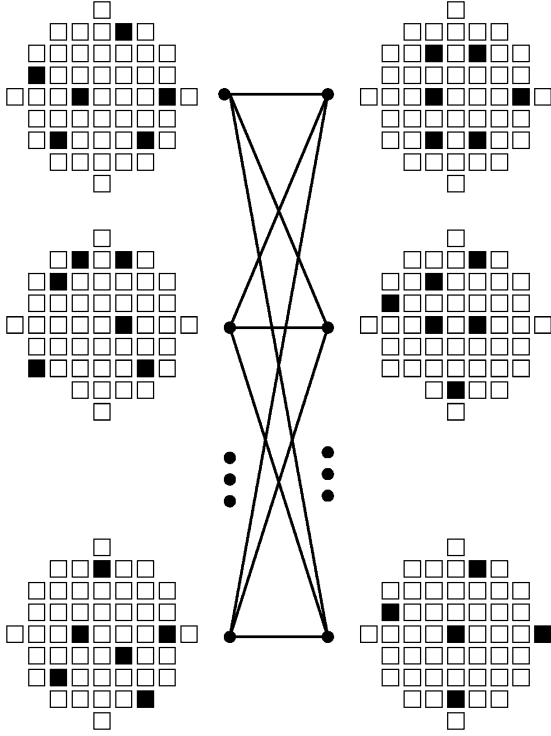


Fig. 1. Optimal integration of two wafer lots.

Input: $K \cdot N$ wafer maps corresponding to K lots each with N wafers.

Output: A mapping from wafers to 3D integration stacks.

1. “unlock” all wafers in all lots.
 2. for each stack in the N^K stacks: calculate the number of good 3D IC resulting from the stack.
 3. sort the N^K wafer stacks in descending order according to the number of good die they produce.
 4. for $i = 1$ to N^K :
 5. if all the wafers that constitute wafer stack i are unlocked then add wafer stack i and lock all its constitute wafers.
-

Fig. 2. Outline of the greedy heuristic algorithm.

requirement is equal to at least $(K + 1)N^K$ bytes needed to store the list of possible wafer stacks for sorting purposes. As our experimental results later show, the memory requirement turns out to be a limiter toward the application of the greedy algorithm for wafer stacks with large numbers of wafers $K > 5$. For example, for $N = 25$ (industrial lot size for 300 mm wafers) and $K = 6$, the algorithm would require at least 1.7 GB of memory and 48 GB of memory for $K = 7$.

B. Iterative Matching Heuristic (IMH)

To understand the proposed iterative matching heuristic, we first consider the special case where $K = 2$, i.e., where there are only two wafer lots $\{L_1, L_2\}$. This special case can be solved optimally using a graph-theoretical framework as follows. First, we construct a bipartite graph composed of $2N$ vertices and N^2 edges as shown in Fig. 1. The first set of N vertices corresponds to wafer maps of the first lot L_1 and the second set of N vertices corresponds to the set of wafer maps of the second lot L_2 . Each edge is labeled by the number of good die produced from integrating the wafers at its end points. In this case finding an optimal *matching* or assignment that maximizes the total yield as mea-

Input: K wafer lots maps $L = \{L_1, \dots, L_K\}$ each with N wafers.

Output: A mapping from wafer to 3D integration stack.

- ```

// start with the first lot as the “seed” lot
1. let $L_s = L_1$
2. let $L = L - \{L_1\}$
3. while $|L| > 1$:
 // g_{best} holds the number of good die
 4. let $g_{best} = 0$
 5. for each $L_j \in L$:
 // calculate the number of good die from
 // optimally matching lots L_s and L_j
 6. let $g_j = G(L_j \odot L_s)$
 7. if $g_j \geq g_{best}$ then
 // b_j is the index of the best lot
 8. let $g_{best} = g_j$
 9. let $b_j = j$
 10. let $L_s = L_s \odot L_{b_j}$
 11. let $L = L - \{L_{b_j}\}$

```
- 

Fig. 3. Iterative matching heuristic.

sured by the edge labels can be achieved in polynomial time in  $O(N^3)$  using the Hungarian algorithm [9]. We will use a left-precedence operator  $\odot$  to denote the optimal matching operation on two wafer map lots; thus, the set of wafer maps resulting from optimally integrating lots  $L_1$  and  $L_2$  can be expressed by  $L_1 \odot L_2$ .

We propose to extend the matching algorithm heuristically by applying it iteratively. Given a set of wafer lots  $L = \{L_1, L_2, \dots, L_K\}$ , the final wafer map can be iteratively calculated as follows:  $L_{i_1} \odot L_{i_2} \odot \dots \odot L_{i_K}$ . One issue that needs to be considered is to find a good iteration order, i.e., the values of  $i_1, \dots, i_K$ , to carry out the matching iteratively. To resolve this issue, at any iteration  $j$  our algorithm picks the lot  $L_{i_j}$  that gives the largest number of good die when optimally matched to the wafer maps  $L_1 \odot \dots \odot L_{i_{j-1}}$  resulting from the previous  $j - 1$  iterations. The first wafer lot  $L_{i_1}$  can be chosen either randomly or according to the number of good die. The algorithm description is formally described in Fig. 3. The runtime of the algorithm is  $O(K^2 N^3)$  (assuming the Hungarian algorithm is used for pair-wise lot matching) and the memory requirement is  $O(N^2)$ . We stress that IMH is guaranteed to be optimal for only two wafer lots ( $K = 2$ ). For more than two lots, IMH is no longer guaranteed to be optimal and is only a heuristic. Our experimental results in Section IV show that it provides very close to optimal results. Note that the order of wafer lot integration in the algorithm has no relationship whatsoever with the order of integration of the actual wafers during fabrication. The final output of the algorithm is the assignment of each wafer to a wafer stack. The integration of the wafers that belong to a wafer stack will be carried out in order during 3-D fabrication.

### C. Optimal Integration Using ILP

To find the optimal integration strategy for general values of  $K$ , we propose an integer linear program (ILP) that maximizes the number of good 3-D ICs yielded from 3-D wafer-to-wafer integration. Let  $x_{i_1, i_2, \dots, i_K}$  denote a binary variable that is true when wafer  $i_1 \in \{1, \dots, N\}$  from lot 1, wafer  $i_2 \in \{1, \dots, N\}$  from lot 2,  $\dots$ , and wafer  $i_K \in \{1, \dots, N\}$  from lot  $K$  are integrated into a 3-D wafer stack. Let  $Y_{i_1, i_2, \dots, i_K} = G(W_{i_1}^1 \dots W_{i_K}^K)$  denote the number of good die resulting from integrating the  $i_1, i_2, \dots$ , and  $i_K$  wafers. Given  $K$  wafers each with  $N$  die, the functional yield maximization problem can be formulated as follows:

$$\max \sum_{i_1=1}^N \dots \sum_{i_K=1}^N Y_{i_1, i_2, \dots, i_K} \times x_{i_1, \dots, i_K} \quad (1)$$

such that there are exactly  $N$  produced wafer stacks

$$\sum_{i_1=1}^N \cdots \sum_{i_K=1}^N x_{i_1, \dots, i_K} = N \quad (2)$$

and each wafer in any lot participates in exactly one 3-D wafer stack

$$\forall i_1 \in \{1, \dots, N\} : \sum_{i_2=1}^N \cdots \sum_{i_K=1}^N x_{i_1, \dots, i_K} = 1 \quad (3)$$

$$\forall i_j \in \{1, \dots, N\} : \sum_{i_1=1}^N \cdots \sum_{i_{j-1}=1}^N \sum_{i_{j+1}=1}^N \cdots \sum_{i_K=1}^N x_{i_1, \dots, i_K} = 1 \quad (4)$$

$$\forall i_K \in \{1, \dots, N\} : \sum_{i_1=1}^N \cdots \sum_{i_{K-1}=1}^N x_{i_1, \dots, i_K} = 1. \quad (5)$$

The ILP requires  $N^K$  variables with a sparse constraint matrix of  $(K+1)N^K$  non-zero (essentially 1) entries out of a total of  $(K \times N + 1)N^K$  entries and an objective function vector of  $N^K$  entries. While the computational runtime complexity incurred from using ILP solvers can be significant, memory will turn out to be the real limiter as specifying the the indices and values of the non-zero entries of the sparse constraint matrix requires  $3 \times (K+1)N^K$  bytes.

#### D. Upper Bounds to the Optimal Solution

An upper bound to the optimal solution can be found by relaxing the ILP and allowing the program variables  $x_{i_1, i_2, \dots, i_K}$  to take fractional values. In this case the  $0 \leq x_{i_1, i_2, \dots, i_K} \leq 1$  constraint is added for each variable in the program, and then the program is solved using standard linear programming techniques (e.g., the simplex method or interior point methods). Standard linear programming solvers are typically quite fast; however, in our case, the main bottleneck will be the memory needed to specify the constrain sparse matrix, especially as  $K$  and  $N$  increase in value and as explained in the previous subsection.

## IV. EXPERIMENTAL RESULTS

In this section, we demonstrate the effectiveness of the proposed algorithms in maximizing the functional yield of wafer-to-wafer 3-D integration through a set of comprehensive experiments. The following settings apply to all of our experiments.

- The classical negative binomial distribution [13] is used to generate defect wafer maps, where the yield of an individual wafer is given by  $(1 + (AD_0/\alpha))^{-\alpha}$ , where  $\alpha$  is the defect clustering ratio,  $D_0$  is the defect density and  $A$  is the area of the die. We use an  $\alpha = 4$  for the defect clustering ratio in all experiments. We assume 300-mm wafers with 3-mm edge exclusion on the periphery. The gross number of die per wafer is given by  $(\pi R_{\text{eff}}^2/A) - 2\pi(R_{\text{eff}}/\sqrt{A}) + \pi$  [4], where  $R_{\text{eff}}$  is the effective wafer radius. For all experiments but one, we assume a standard wafer lot size of 25 wafers. We vary the die area, defect density and number of wafers in the 3-D stack depending on the experiment.
- All proposed algorithms are implemented in C++ and compiled with  $-O3$  optimizations. The basic Hungarian algorithm is implemented to compute the optimal matching of wafers in two wafer lots, and the GNU linear programming kit (LPK) is used to compute the solution to the integer linear program together with the solution to the relaxed linear program.<sup>1</sup>

<sup>1</sup>It is likely that using a commercial ILP solver like CPLEX will speed our calculations. The GNU manual mentions that GNU LPK is slower by 10–100× compared to CPLEX.

TABLE I  
IMPACT OF DEFECT DENSITY PER WAFER ON THE YIELD

| method | Yield per wafer |             |             |             |             |             |             |       |
|--------|-----------------|-------------|-------------|-------------|-------------|-------------|-------------|-------|
|        | 30%             |             | 50%         |             | 70%         |             | 90%         |       |
|        | 3D IC yield     | 3D IC yield | 3D IC yield | 3D IC yield | 3D IC yield | 3D IC yield | 3D IC yield |       |
| random | 2.54%           | 1.000       | 12.37%      | 1.000       | 34.24%      | 1.000       | 72.88%      | 1.000 |
| greedy | 4.07%           | 1.600       | 14.75%      | 1.192       | 36.61%      | 1.069       | 73.90%      | 1.014 |
| ILM    | 4.07%           | 1.600       | 14.75%      | 1.192       | 36.95%      | 1.079       | 74.41%      | 1.021 |
| ILP    | 4.24%           | 1.667       | 15.08%      | 1.219       | 37.29%      | 1.089       | 74.41%      | 1.021 |
| UB     | 4.24%           | 1.667       | 15.08%      | 1.219       | 37.29%      | 1.089       | 74.41%      | 1.021 |

- All experiments are carried out on a workstation equipped with an Intel Core 2 Duo Extreme edition processor running at 2.93 GHz with 2 GB of dynamic memory. All reported results are an average of five random seeds.
- For comparison purposes, we have implemented a yield oblivious assignment strategy where wafers from different lots are randomly integrated together. Such assignment is oblivious to the flexibility offered by having individual wafer test information. The final 3-D IC yield in this case is expected to be equal to the multiplication of the yield of the individual wafers. For example, if the yield per wafer is 90% then the expected yield of a 3-D wafer stack composed of three wafers is equal to  $0.9 \times 0.9 \times 0.9 = 73\%$ .

**Impact of Defect Density.** In the first set of experiments, we investigate the impact of the defect density per wafer on the final yield of the produced 3-D ICs. We compare the performance of the proposed integration algorithms at different defect densities. The die area is assumed to be  $1 \text{ cm}^2$ , which gives about  $N = 590$  die per wafer for a 300-mm wafer. We set the number of wafers in the 3-D stack to be equal to  $K = 3$  and vary the defect density to result in yields from 30% to 90% per wafer. In Table I, we report two values for each integration algorithm: 1) the overall yield of 3-D ICs, and 2) the number of produced good 3-D ICs normalized to the number of 3-D ICs produced from random assignment. The latter value gives the advantage of deploying our techniques over a yield-oblivious random assignment integration. Furthermore, the normalized value gives the direct increase in revenue from using our algorithms.

The results show that the proposed integration algorithms consistently lead to an improved overall yield compared to a random yield-oblivious assignment. The defect density and, hence, the yield per wafer is a factor of the design, the process technology and the fabrication facility. Thus, for a given wafer yield dictated by these factors, the proposed techniques result in quite significant improvements. For example, at 50% yield per wafer, the optimal technique (ILP) gives a 21.9% improvement over random assignment, i.e., the revenues will be multiplied by 1.219. The results also show that the upper bounds calculated through relaxing the ILP are quite close to the optimal solution.

One may wonder if the random assignment technique might give comparable results to the proposed algorithms if different random assignments are simulated and the best one is picked and applied during integration. To test that possibility we executed 10000 different random integration assignments for the case at yield = 50%. The different simulations give results around the reported average of 12.37% with a standard deviation of 0.206 and a maximum of 13.02%; these results are far from the optimal yield value 15.08%.

**Impact of Die Area.** Increasing the die area decreases the number of produced die per wafer and also reduces the yield as a defect would destroy a larger portion of the wafer as the die are larger. To study the performance of the proposed algorithms under various die sizes, we choose a defect density of  $0.4 \text{ defects/cm}^2$  and vary the die sizes from  $50 \text{ mm}^2$  to  $250 \text{ mm}^2$ . We assume the number of wafers in the 3-D stack is equal to  $K = 3$ . The results are reported in Table II. As

TABLE II  
IMPACT OF DIE AREA ON THE FINAL YIELD FOR THE VARIOUS INTEGRATION STRATEGIES. A DEFECT DENSITY OF 0.4 DEFECTS PER  $\text{cm}^2$  IS ASSUMED

| method | area=50mm <sup>2</sup> |       | area=100mm <sup>2</sup> |       | area=150mm <sup>2</sup> |       | area=200mm <sup>2</sup> |       | area=250mm <sup>2</sup> |       |
|--------|------------------------|-------|-------------------------|-------|-------------------------|-------|-------------------------|-------|-------------------------|-------|
| random | 55.61%                 | 1.000 | 31.92%                  | 1.000 | 18.42%                  | 1.000 | 10.83%                  | 1.000 | 6.94%                   | 1.000 |
| greedy | 56.83%                 | 1.022 | 34.47%                  | 1.080 | 21.58%                  | 1.171 | 14.44%                  | 1.333 | 10.19%                  | 1.467 |
| IMH    | 57.15%                 | 1.028 | 34.63%                  | 1.085 | 21.84%                  | 1.186 | 14.44%                  | 1.333 | 10.19%                  | 1.467 |
| ILP    | 57.24%                 | 1.029 | 34.97%                  | 1.096 | 22.11%                  | 1.200 | 14.80%                  | 1.367 | 10.65%                  | 1.533 |
| UB     | 57.24%                 | 1.029 | 34.97%                  | 1.096 | 22.11%                  | 1.200 | 14.80%                  | 1.367 | 10.65%                  | 1.533 |

TABLE III  
IMPACT OF NUMBER OF WAFER STACKS FOR THE VARIOUS INTEGRATION STRATEGIES. THE INDIVIDUAL WAFER YIELD IS 80%. A “—” INDICATES THAT A SOLUTION WAS NOT FEASIBLE DUE TO MEMORY LIMITATIONS

| method | metric      | $K=2$<br>wafers in stack |       | $K=3$<br>wafers in stack |       | $K=4$<br>wafers in stack |       | $K=5$<br>wafers in stack |       | $K=6$<br>wafers in stack |       | $K=7$<br>wafers in stack |       |
|--------|-------------|--------------------------|-------|--------------------------|-------|--------------------------|-------|--------------------------|-------|--------------------------|-------|--------------------------|-------|
| random | yield       | 64.07%                   | 1.000 | 51.02%                   | 1.000 | 40.68%                   | 1.000 | 32.54%                   | 1.000 | 25.76%                   | 1.000 | 20.51%                   | 1.000 |
|        | runtime (s) | 0.00                     |       | 0.00                     |       | 0.00                     |       | 0.00                     |       | 0.00                     |       | 0.00                     |       |
| greedy | yield       | 64.92%                   | 1.013 | 52.71%                   | 1.033 | 43.73%                   | 1.075 | 36.44%                   | 1.120 | —                        | —     | —                        | —     |
|        | runtime (s) | 0.00                     |       | 0.09                     |       | 2.79                     |       | 79.94                    |       | —                        |       | —                        |       |
| IMH    | yield       | 65.25%                   | 1.019 | 53.22%                   | 1.043 | 44.07%                   | 1.083 | 36.61%                   | 1.125 | 30.68%                   | 1.191 | 25.76%                   | 1.256 |
|        | runtime (s) | 0.00                     |       | 0.00                     |       | 0.01                     |       | 0.02                     |       | 0.03                     |       | 0.04                     |       |
| ILP    | yield       | 65.25%                   | 1.019 | 53.56%                   | 1.050 | 44.41%                   | 1.092 | —                        | —     | —                        | —     | —                        | —     |
|        | runtime (s) | 0.00                     |       | 5.49                     |       | 15435.41                 |       | —                        |       | —                        |       | —                        |       |
| UB     | yield       | 65.25%                   | 1.019 | 53.56%                   | 1.050 | 44.58%                   | 1.096 | —                        | —     | —                        | —     | —                        | —     |
|        | runtime (s) | 0.00                     |       | 0.392                    |       | 40.64                    |       | —                        |       | —                        |       | —                        |       |

expected, the yield decreases as the die area increases; however, the improvements in yield from using the proposed integration strategies increase in magnitude as the die area increases.

**Impact of Number of Wafers in the 3-D Stack.** In this important experiment, we study the scalability of the proposed algorithms in quality and runtime as the number of wafers in the 3-D stack increases. We initially assume a defect rate resulting in 80% yield per wafer and a die area of  $1 \text{ cm}^2$ . The yield and runtime results are given in Table III. A “—” in the table indicates the algorithm failed because of memory allocation problems. The obvious part of the results is that yield generally degrades, as expected, as the number of wafers in the stack increases. In comparing the various algorithms, we find the following.

- The upper bounds on the optimal solutions stay tight for up to  $K = 4$ ; however, both the ILP and the relaxed LP run out of memory for values of  $K \geq 5$ . Furthermore, the runtime of the ILP dramatically increases as the number of wafers  $K$  increase. The scalability of the optimal ILP algorithm can be improved by using more powerful workstations and better commercial ILP solvers.
- The greedy algorithm produces good results up to  $K = 5$ . For larger values of  $K$ , it runs into memory problems that prevent it from scaling gracefully.
- The iterative matching heuristic is the most scalable of all algorithms. All instances are solved in less than 1 second and furthermore the quality of the solution is close to the optimal. It also dominates the greedy algorithm in both yield and runtime. Compared to other methods, the iterative matching heuristic is the only technique that is scalable in memory requirements.
- Overall the yield loss due to wafer-to-wafer integration at large values of  $K$  will be unacceptable unless the yield per wafer is extremely high or the 3-D structure has redundant resources to cope with the defects (as is the case with error correction codes in memory stacks).

**Impact of Wafer Lot Size.** One possibility to improve the results of wafer-to-wafer integration is to *batch or aggregate* wafer lots to effectively increase the size of wafer lot. For example, it is possible to aggregate two wafer lots each with 25 wafers to produce a larger wafer lot of 50 wafers. The aggregated wafer lot will then be used with other aggregated wafer lots to derive the integration process. A random assignment will not benefit from such batching as the yield will stay the

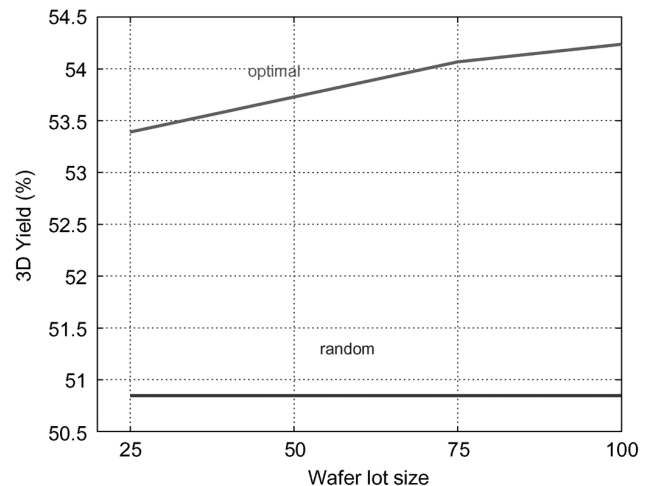


Fig. 4. Yield benefit from increasing the wafer lot size. Results are for wafer lot sizes of 25, 50, 75, and 100 wafers.

same on the average. However, the proposed algorithms can exploit the larger wafer lots to find better assignments that further maximize the functional yield. Towards testing this hypothesis, we carry out an experiment where we try four different wafer lot sizes  $N = 25, 50, 75,$  and  $100$  (we assume  $K = 3$ , individual wafer yield of 80%, and die area is  $1 \text{ cm}^2$ ). We plot the yield per wafer stack for both the random assignment and optimal assignment integration strategies in Fig. 4. As hypothesized, the yield random assignment strategy stays on the average constant; however, as the wafer lot size increases, the optimal strategy is able to exploit this flexibility and increase the yield.

**Cost Considerations.** Our proposed methods require wafer testing in comparison to randomly assigning wafers. The cost of testing should be evaluated in comparison to the improvement in revenues attained from the increased yield from our methods. Providing exact cost numbers requires many factors, but we consider here some hypothetical estimates for the purpose of illustration. Let's assume a 3-D processor based on the die of an Intel Core 2 Duo integrated with two DRAM die. Intel Core 2 Duo has a die area equal to  $143 \text{ mm}^2$ . Using our

die calculating formula in Section IV, the number of die per wafer is 418. The price of Core 2 Duo depends on the speed of the model. For the E6700 model, the price at launch time was \$530. We have no idea of the defect density at Intel fabrication facilities, but we assume the reasonable defect density of 0.4 defects/cm<sup>2</sup>. Then from Table II, we roughly expect that our technique will improve the yield from 18.42% to 22.11% which translates to extra 15 3-D ICs which are worth \$7950 (not including the price of the DRAM die). Thus, we can afford up to \$7950 in additional test costs.

## V. CONCLUSION

We have formulated the problem of yield maximization in wafer-to-wafer integration. We have proposed a optimal techniques and scalable heuristics with near optimal performance to maximize the yield. The proposed assignment techniques provide significant improvements to wafer-to-wafer integration yield, increasing the overall number of good die in many cases. Our proposed methods require wafer testing in comparison to randomly assigning wafers. The cost of testing should be evaluated in comparison to the improvement in revenues attained from the increased yield from our methods.

## REFERENCES

- [1] K. Banerjee, S. J. Souri, P. Kaput, and K. C. Saraswat, "3-D ICs: A novel chip design for deep-submicrometer interconnect performance and systems-on-chip integration," *Proc. IEEE*, vol. 89, no. 5, pp. 602–633, May 2001.
- [2] J. A. Burns, B. F. Aull, C. Chen, C.-L. Chen, C. L. Keast, J. Knecht, V. Suntharalingam, K. Warner, P. Wyatt, and D.-R. Yost, "A wafer-scale 3-D circuit integration technology," *IEEE Trans. Electron Devices*, vol. 53, no. 10, pp. 2507–2516, Oct. 2006.
- [3] S. Das, A. Fan, K.-N. Chen, C. S. Tan, N. Checka, and R. Reif, "Technology, performance, and computer-aided design of three-dimensional integrated circuits," in *Proc. Int. Symp. Physical Design*, 2004, pp. 108–115.
- [4] D. K. de Vries, "Investigation of gross die per wafer formula," *IEEE Trans. Semicond. Manufact.*, vol. 18, no. 1, pp. 136–139, Jan. 2005.
- [5] C. Ferri, S. Reda, and R. I. Bahar, "Strategies for improving the parametric yield and profits of 3-D ICs," in *Proc. Int. Conf. Computer Aided Design*, 2007, pp. 220–226.
- [6] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, 1st (23rd printing) ed. New York: W.H. Freeman, 1979.
- [7] R. J. Hess and L. H. Weiland, "Extraction of wafer-level defect density distributions to improve yield prediction," *IEEE Trans. Semicond. Manufact.*, vol. 12, no. 2, pp. 175–183, May 1999.
- [8] W. Maly and J. Deszczka, "Yield estimation model for VLSI artwork evaluation," *Electron. Lett.*, vol. 19, no. 6, pp. 226–227, 1983.
- [9] J. Munkres, "Algorithms for the assignment and transportation problems," *J. Soc. Ind. Appl. Math.*, vol. 5, no. 1, pp. 32–38, 1957.
- [10] R. S. Patti, "Three-dimensional integrated circuits and the future of systems-on-chip designs," *Proc. IEEE*, vol. 94, no. 6, pp. 1214–1224, Jun. 2006.
- [11] L. Smith, G. Smith, S. Hosali, and S. Arkalgud, "3-D: It all comes down to cost," presented at the 3-D Architectures for Semiconductor Integration and Packaging, 2007.
- [12] L. Smith, G. Smith, S. Hosali, and S. Arkalgud, "Yield considerations in the choice of 3D technology," in *Proc. IEEE Int. Symp. Semiconductor Manufacturing*, 2007, pp. 535–537.
- [13] C. H. Stapper and R. J. Rosner, "Integrated circuit yield management and yield analysis: Development and implementation," *IEEE Trans. Semicond. Manufact.*, vol. 8, no. 2, pp. 95–102, May 1995.
- [14] A. W. Topol, J. D. C. La Tulipe, L. Shi, D. J. Frank, K. Bernstein, S. E. Steen, A. Kumar, G. U. Singco, A. M. Young, K. W. Guarini, and M. Jeong, "Three-dimensional integrated circuits," *IBM J. Res. Develop.*, vol. 50, no. 4–5, pp. 491–506, 2006.

## Low-Power Snoop Architecture for Synchronized Producer-Consumer Embedded Multiprocessing

Chenjie Yu and Peter Petrov

**Abstract**—We introduce a cross-layer customization methodology where application knowledge regarding data sharing in producer-consumer relationships is used in order to aggressively eliminate unnecessary and predictable snoop-induced cache lookups even for references to shared data, thus, achieving significant power reductions with minimal hardware cost. The technique exploits application-specific information regarding the exact *producer-consumer relationships* between tasks as well as information regarding the *precise timing of synchronized accesses* to shared memory buffers by their corresponding producers and/or consumers. Snoop-induced cache lookups for accesses to the shared data are eliminated when it is ensured that such lookups will not result in extra knowledge regarding the cache state in respect to the other caches and the memory. Our experiments show average power reductions of more than 80% compared to a general-purpose snoop protocol.

**Index Terms**—Low-power cache coherence, low-power multiprocessor systems-on-a-chip (MPSoC), producer-consumer communication in MPSoC.

## I. INTRODUCTION

The abundance of wireless connectivity coupled with the ever growing increase in integration densities have resulted in a multitude of handheld and wearable embedded applications such as portable media players, mobile phones with aggregate data functions, personal organizers, etc. Battery life and power consumption has become one of the primary implementation constraints for these applications. Due to the integration of multiple functionalities and ever increasing demand for performance, it has become a natural design practice to utilize multiprocessor systems-on-a-chip (MPSoC) in embedded systems. Typically these systems feature several processor cores, possibly of heterogeneous natures, that access a shared memory. For reasons of low complexity and high speed, the most common approach is to use a shared system bus. In order to provide the required bandwidth to the shared memory, local caches at each processor node are usually employed. Local caching in multiprocessor systems, however, introduces the possibility of cache incoherence; a situation that occurs when a processor updates a data object after that same object is cached somewhere else. To resolve this issue, cache coherence protocols are used.

The snoop-based cache coherence protocols are the most widely deployed as they rely on the inherent broadcast nature of the common bus connecting the processor nodes to the memory. Each cache controller "snoops" the bus for memory transfers, for each of which a cache lookup is performed in order to determine whether a cache block state should be changed in the local cache. Easily extendable multiprocessor structures and software-transparent implementation have made snoop protocols easy to understand, deploy, and reuse, with minimal impact on the performance of memory subsystem [1]. Quite often, however, shared data are cached in just a few nodes. Snooping in the others leads to a waste of energy. It was shown in [2] that only around 10% of the application memory references actually require cache coherence tracking.

Manuscript received October 22, 2007; revised March 19, 2008. First published August 04, 2009; current version published August 19, 2009.

The authors are with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 USA (e-mail: purety@umd.edu; ppetrov@umd.edu).

Digital Object Identifier 10.1109/TVLSI.2009.2019414