An Efficient Approach to SiP Design Integration

Meng-Syue Chan, Chun-Yao Wang and Yung-Chih Chen
Dept. of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
g9562640@oz.nthu.edu.tw, {wcyao, ycchen}@cs.nthu.edu.tw

Abstract

System-in-Package (SiP) design methodology integrates multiple dies which come from different vendors into a package. It is more advantageous than Printed Circuit Board (PCB) and System-on-a-Chip (SoC) design methodologies from the aspects of development cost, power consumption, time-to-market, and miniaturization. Since these integrated dies can be manufactured and tested separately, the best-quality SiP design can be achieved by using these best-quality dies. However, with considering cost constraint, the best-quality dies are not always qualified due to high cost. Thus, this paper proposes an algorithm to optimize the combination of dies with different yields and test coverages in an SiP such that a minimal cost is achieved subject to a quality constraint, or a least defect level is reached under the cost constraint. The proposed method significantly prunes the solution space, and efficiently determines the combination of dies.

Keywords

System-in-Package (SiP), optimization

1. Introduction

With the advances of semiconductor technologies, the number of transistors in a chip increases exponentially. This evolution leads a large system to be realized in a single chip, which is named System-on-a-Chip (SoC). An SoC design typically integrates various cores or intellectual properties (IPs), which are developed in-house or purchased from IP vendors, on a chip. Although the SoC design methodology has a great integration of versatile cores, it suffers a lower yield from the large die size and heterogeneous integration as compared with Application Specific ICs (ASICs).

To balance the yield and miniaturization of system designs, the System-in-Package (SiP) design methodology is proposed. An SiP consists of several bare dies which are placed on a common substrate horizontally or vertically within the same package. Designers can integrate a variety of dies which are manufactured and tested separately with the most advanced process technologies and test methods, respectively. Thus, the yield of an SiP design may be elevated without sacrificing time-to-market and chip size. Furthermore, SiP also provides a solution to protect the intellectual properties of the integrated dies, since the manufactured dies, instead of design cores are delivered to the system integrators.

In order to produce a high-quality and cost-effective SiP design, designers must explore the relationship between cost and quality among the individual die and the entire SiP design [1][4][5]. SiP designs are similar to multichip module (MCM) design from the aspect of integration of manufactured and tested dies. But an SiP design is realized in a single package rather than on a PCB. Previous works [1][4][5] on MCM designs and ASIC design which are related to this work are summarized here. In [1], Abadir explores the tradeoff between various testing and rework strategies for MCM designs. The work compared the cost and the defect level (DL) of various testing strategies such as die-level testing, full DFT, and partial DFT. The disadvantages and advantages of each testing strategy are also discussed in that paper. However, the paper does not discuss the algorithm of obtaining a cost-effective and high quality MCM design. [4] proposes a test cost prediction model that estimates chip test cost and test quality. It focuses on predicting the cost and the quality of the test method to produce a high quality ASIC in a manufacturing environment. [5] proposes a system named Profit Evaluation System (PES) to determine the yield and the test plan when the specified quality level is given. Fabric type and raw manufacturing data (e.g., wafer size, wafer cost, defect density, and defect distribution) are given for the PES. The outputs are the yield and the fault coverage that come out a maximal profit. However, those models are not suitable for SiP designs, since dies of an SiP design are provided by third parties, and the raw manufacturing data are not available to SiP integration.

Since an SiP design integrates many manufactured and tested dies in one package, the qualities of these dies directly affect the quality of the SiP design. If the best-quality dies are used, the quality of the SiP design is also the best. However, achieving high quality without considering cost is impractical. In this work, we propose a novel and efficient algorithm to determine the combination of dies in an SiP design such that the objectives are achieved under the given constraints in the proposed two quality-cost evaluation models. The objective of the first model is the cost minimization, its corresponding constraint is the DL. Thus, this model minimizes the cost under the DL constraint. The second model is to minimize the DL under the cost constraint.

The rest of this paper is organized as follows. Section II introduces the background. Section III proposes our algorithm and models for integrating an SiP design. The experimental results are shown in Section IV. Section V concludes this work.

2. Background

An SiP design may consist of several components, such as digital block, analog block, substrate, memory block and some passive components. The quality of the design is measured by DL. DL is a measurement that indicates the percentage of components passing test process but are still defective. The lowest DL can be obtained by using the highest quality component and using test methods with the highest fault

This work was supported in part by the National Science Council of R.O.C. under Grant NSC 97-2220-E-007-042 and NSC 97-2220-E-007-034.

coverage. From another viewpoint of the quality requirement, however, the quality would be acceptable if the DL is smaller than the given DL constraint.

DL is used to evaluate the quality of a design and can be formulated as Equation (1), where Y is the yield of the component, FC is the fault coverage of the test method.

$$DL = 1 - Y^{(1-FC)} \tag{1}$$

However, Equation (1) is used for single chip or single core design. For multiple-component designs, Equation (1) is modified as Equation (2) where m is the number of components within a design. For example, assume there are three components in an SiP design and their yields are 90%, 95%, and 99%, respectively. The corresponding fault coverages are 80%, 85%, and 88%, respectively. Then, the DL of this design is 29527 parts per million (ppm) as shown in Equation (3).

$$DL = 1 - \prod_{i=1}^{m} \left\{ Y_i^{(1-FC_i)} \right\} \tag{2}$$

$$\begin{aligned} DL &= 1 - \quad (0.9^{1-0.8}) \times (0.95^{1-0.85}) \times (0.99^{1-0.88}) \\ &= 29527ppm \end{aligned} \tag{3}$$

3. The proposed approach

The quality of an SiP design is affected by the quality of the integrated components and the fault coverage of the test strategies. Integrating with higher quality dies makes the quality of an SiP design better as well. Thus, Known Good Die (KGD)[10] acquisition is a critical issue to SiP integration. However, with economic concerns, the development and manufacturing cost must be considered in the SiP integration. Thus, in this paper, we propose two quality-cost evaluation models to optimize the SiP integration. In our models, the best solution can be found under the cost constraint or the quality constraint. The first model reduces the manufacturing and development cost as much as possible, but keeps its quality. Thus, this model is to minimize the cost under the DL constraint. On the other hand, if the cost is a more important factor, the second model, which minimizes the DL under the cost constraint can be used. Here, we assume that components with different yields but the same functionality, and test methods with different fault coverages are both available to integrating an SiP design.

3.1. Cost Minimization under the Defect Level Constraint

In this model, we want to minimize the cost under the DL constraint. This evaluation model is summarized as follows. *Minimize:*

$$TC = \sum_{i=1}^{m} X_i C_i + P \qquad X_i \in \{0, 1\} \tag{4}$$

Subject to:

$$P = Fine \times DL \tag{5}$$

$$\begin{aligned} &(X_1 + X_2 + \ldots + X_{n_1})(X_{n_1+1} + \ldots + X_{n_1+n_2}) \\ &\ldots(X_{n_1+n_2+\ldots+n_{m-1}+1} + \ldots + X_{n_1+n_2+\ldots+n_m}) = 1 \end{aligned} \tag{6}$$

$$DL \leq Con_{DL} \tag{7}$$

The total cost (TC) of an SiP design can be formulated as Equation (4) where C_i is the cost of the component i or the cost of the test method i, X_i is a binary number, and P is the $average\,penalty$ of delivering a defective product to customers. P can be formulated as Equation (5) where $Fine$ is the payment to customers when delivering a defective product. Equation (6) indicates that only one item, either a component or a test method, can be chosen in one clause. Equation (7) indicates that the actual DL must meet the DL constraint, denoted as Con_{DL}.

The optimal solution of this model can be found after all combinations of components and test methods have been exhaustively tried. However, the number of combinations is enormous if the SiP design consists of many components, and has many test choices. Hence, we have to filter out some combinations to speed up the evaluation process. Note that the most important thing is that the best combination must exist in the pruned solution space.

First, we focus on pruning the combinations within the same component group. For example, assume that an SiP design consists of a component A, and other components. The component A has five choices of different yields and three test choices of different fault coverages. Thus, there are 15 (3 × 5) combinations for the component A. One of 15 combinations will be a part of the final result. We then generate these combinations, and calculate the cost (the sum of component cost and test cost) and a parameter w for each combination which is calculated as Equation (8). Since the DL of a single component design can be expressed as $(1-w)$, a combination with a larger w can result in a lower DL.

$$w = Y^{(1-FC)} \tag{8}$$

Then these combinations are sorted by their cost in ascending order. Fig. 1(a) is a sample result of the sorting. An element in Fig. 1(a) that has a higher cost but a lower w as compared with its left neighbor will be pruned out. We can see that the element-2 is the case that has a higher cost but a lower w as compared with the element-1. By pruning out the element-2 in Fig. 1(a), the elements with higher costs also have higher w as shown in Fig. 1(b). This is the pruning process within the same component group.

Next, another parameter $Cost_{new}^{i}$ as shown in Equation (9) is calculated for each remaining element i within the same component group. $Cost_{new}^{i}$ is used to evaluate the effect of selecting the element i in the final SiP integration while considering other components and the penalty issue. This evaluation equation is derived from Equation (2) and Equation (5), but it assumes that except the selected element i, the other elements are defect-free (i.e. yield is 100%) and their costs are zero. The cost evaluation of selecting the element i based on this assumption can highlight the impact of element i on the integration. $Cost_{new}^{i}$ consists of the cost of element i (c_i) and the penalty P where P is minimal since other elements are assumed defect-free. In other words, Equation (9) shows the cost of integration when defects only occur at the element

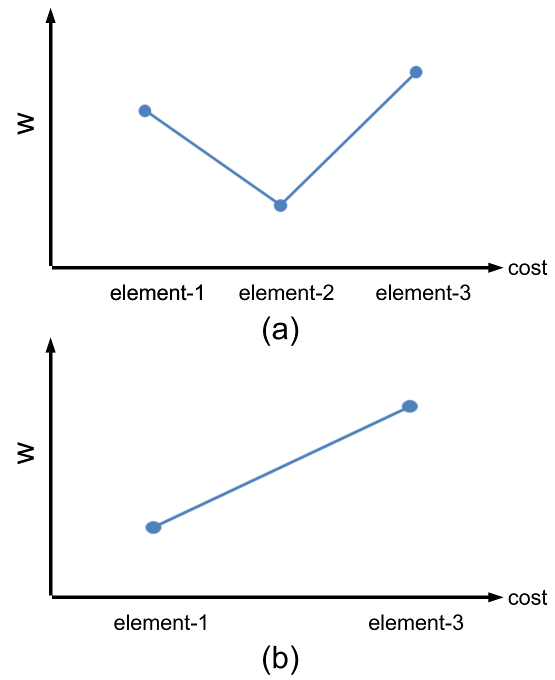

Fig. 1. (a) An element with a higher cost but a lower w (element-2) is in the original data (b) The remaining elements after the pruning process.

i. Thus, $Cost^i_{new}$ is the minimal cost of selecting the element i in the SiP integration.

$$\begin{aligned} Cost^i_{new} &= P + c_i \\ &= Fine \times (1 - w_i \times 1 \times 1 \times \ldots) + c_i \end{aligned} \tag{9}$$

When all remaining elements get their $Cost^i_{new}$, these elements are sorted again by their $Cost^i_{new}$ in ascending order. The element that has a higher $Cost^i_{new}$ but a lower w will then be pruned out in this second pruning process. The element i which has a higher $Cost^i_{new}$ but a lower w under our assumption indicates that including element i in the SiP integration will cause a higher cost. Thus, we can prune out this element i.

$$\begin{aligned} & DL \leq Con_{DL} \\ & (1 - \prod_{i=1}^{m} w_i) \leq Con_{DL} \\ & \log{(1 - Con_{DL})} \leq \log{(\prod_{i=1}^{m} w_i)} \\ & -\log{(1 - Con_{DL})} \geq -[\log(w_1) + \ldots + \log(w_m)] \\ & S(-\log{(1 - Con_{DL})}) \geq S\{-[\log{(w_1)} + \ldots + \log{(w_m)}]\} \\ & Con_{scaled} \geq (sw_1 + sw_2 + \ldots + sw_m) \end{aligned} \tag{10}$$

After these pruning processes, the elements with a higher cost within the same component group (or a higher $Cost^i_{new}$ in the whole SiP), but a lower w are eliminated. However, we still need an efficient approach to select the proper elements as the part of our answer from these remaining elements. In this work, we transform our picking up problem into the one that is similar to 0-1 knapsack problem. Thus, the data of these remaining elements have to be transformed for accommodating to the modified 0-1 knapsack problem.

Now, we introduce the process of data transformation where the DL constraint of this model is focused on. Equation (10) shows the transformation process. In the row 3, the inequality is taken a logarithmic operation. Since $\log(1 - Con)$ and $\log(w_i)$ are negative numbers, we turn them to positive ones by multiplying -1. Then, we multiply a positive constant S to the inequality such that both sides of the inequality are integers. This step is for our picking up process. These transformed data are named scaled constraint (Con_{scaled}) and scaled weight (sw_i), respectively, as shown in the last row of Equation (10) where $Con_{scaled} = -S \times \log(1 - Con_{DL})$ and $sw_i = -S \times \log(w_i)$. This data transformation process can speed up our approach, since the multiplication operations in the constraint equation are currently replaced by the addition operations. Another step of the transformation process is to make the $Cost^i_{new}$ of each element be a negative number by multiplying -1, and it is named scaled cost ($Cost^i_{scaled}$). This step is also for the picking up process since it always picks up an element with a larger value. Thus, a lower $Cost^i_{new}$ will become a larger value after multiplying -1 and will be selected.

TABLE I
THE DATA OF THE EXAMPLE.

Component			*Test Method*		
Type	*Cost*	*Yield*	*Method*	*Cost*	*Fault Coverage*
A-1	5	85%	TA-1	10	90%
A-2	7	90%	TA-2	15	95%
A-3	8	93%	TB-1	10	93%
B-1	13	90%	TB-2	20	95%
B-2	15	95%	TB-3	50	99%
$Fine$: 10,000			$DL\,Constraint$: 9,000 ppm		

After performing the data transformation, the next step is the picking up process. Our picking up process is similar to 0-1 knapsack problem, but has a different property. This property is that we may have more than one choice for each SiP component, but just exactly one element can be selected. This property has been expressed in Equation (6) of this model.

We use an example to demonstrate our approach. Assume that there are three components to be integrated in one SiP design. One is with the component A; two are with the component B. Component A has three levels of quality with respect to different cost and yields, A-1, A-2, and A-3. Component B also has two levels of quality. Besides, several test choices are provided with different fault coverages and costs, respectively. Given that the DL constraint is 9,000 ppm, and the $Fine$ is \$10,000. The detailed data are listed in Table I.

Table II shows the data in the pruning process, and indicates if an element can be pruned out or not. The first pruning process prunes off the elements ComA4 and ComB3. Fig. 2 shows the curve of all elements of component B before and after the first pruning. After the first pruning process, we eliminate the element ComB3 of cost \$33 since its cost is higher than its left neighbor, but has a lower w as shown in Fig 2(a). These remaining elements then get their $Cost^i_{new}$. Fig. 3 shows the curve of those remaining elements of component B

TABLE II
THE DATA IN THE PRUNING PROCESSES.

$Element$	$Cost$	w	$member$	$1st. pruning$	$Cost^i_{new}$	$2nd. pruning$	$-\log(w)$	sw	$Cost^i_{scaled}$
ComA1	15	0.98388	A-1,TA-1	No	176.21	Yes	-	-	-
ComA2	17	0.98952	A-2,TA-1	No	121.81	Yes	-	-	-
ComA3	18	0.99277	A-3,TA-1	No	90.31	Yes	-	-	-
ComA4	20	0.99191	A-1,TA-2	Yes	-	-	-	-	-
ComA5	22	0.99475	A-2,TA-2	No	102.93	Yes	-	-	-
ComA6	23	0.99638	A-3,TA-2	No	75.54	No	0.0016	16	-75.54
ComB1	23	0.99265	B-1,TB-1	No	96.48	Yes	-	-	-
ComB2	25	0.99642	B-2,TB-1	No	60.84	Yes	-	-	-
ComB3	33	0.99475	B-1,TB-2	Yes	-	-	-	-	-
ComB4	35	0.99744	B-2,TB-2	No	60.61	No	0.0011	11	-60.61
ComB5	63	0.99895	B-1,TB-3	No	73.53	Yes	-	-	-
ComB6	65	0.99949	B-2,TB-3	No	70.13	No	0.0002	2	-70.13
$DL\,constraint$: 9,000 ppm					S:10,000			Con_{scaled}: 39	

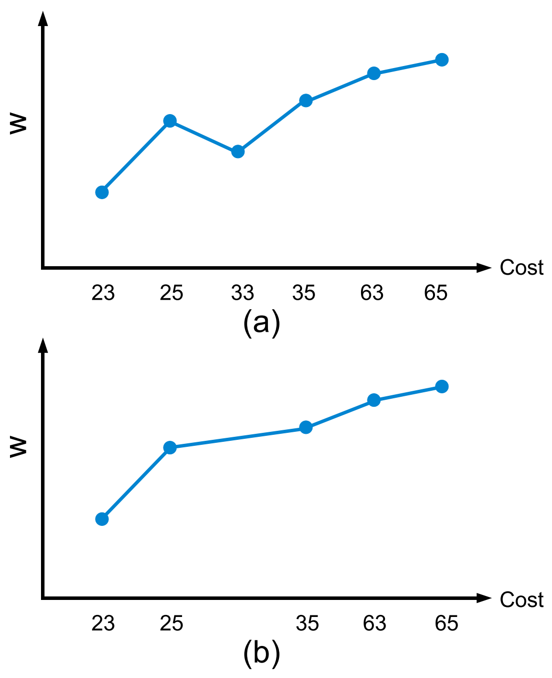

Fig. 2. The first pruning process of component B (a) Before the pruning process, it contains a redundant element. (b) After the pruning process, the remaining elements are the prospective candidates of the SiP design.

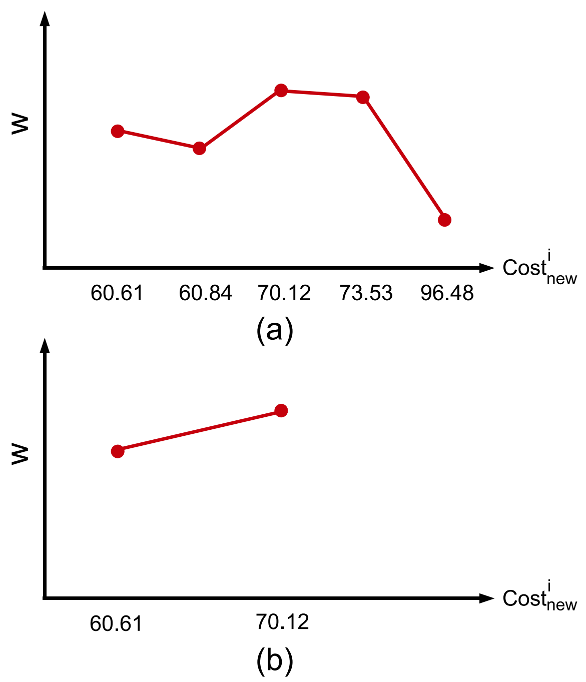

Fig. 3. The second pruning process of component B (a) The elements left in the first pruning process will be sorted by their $Cost^i_{new}$. (b) The remaining elements after the second pruning process.

before and after the second pruning process. Those remaining elements are sorted by their $Cost^i_{new}$ in ascending order. The elements which have higher $Cost^i_{new}$ but lower w would be pruned out again in the second pruning process. The detailed data can be seen in Table II. In summary, there are three components in the design of this example, and the size of the original solution space are 216 $(6 \times 6 \times 6)$. However, only 125 $(5 \times 5 \times 5)$ elements need to be examined after the first pruning process. Furthermore, the number of candidates of this design under the constraint remains 4 $(1 \times 2 \times 2)$ elements after the second pruning process. This indicates that the pruning processes are effective to speed up our algorithm.

After the pruning process, the picking up process proceeds. Fig. 4 shows the picking up process of the example. The process of filling out the table is row by row. In Fig. 4(a), the entries of the row A are all zeros until the capacity is greater than 15, since sw of ComA6 is 16. Thus, we select ComA6 as part of our solution, and the summation of $Cost^i_{new}$ of this selection is -75.54. After finishing the picking up process for the element A, we consider the row B1 as shown in Fig. 4(b). When the capacity is 2, ComB6 can be selected at this entry, since the sw of ComB6 is not greater than the capacity. While the capacity is 11, ComB6 can be replaced with ComB4 at this entry. This is because the $Cost_{scaled}$ of ComB4 (-60.61) is greater than that of ComB6. The $Cost_{scaled}$ is then updated to -145.67 and ComB6 replaces the ComB4 while the capacity is 18. This is because the selection with ComB6 leaves the space for accommodating to ComA6. But when the capacity is 27, the selection with ComB4 is better than that with ComB6. The picking up process of B2 is shown in Fig. 4(c). The 4th entry of the row B2 is -140.26, this selection indicates that both B1 and B2 select the ComB6. Then the $Cost_{scaled}$ is -215.80 for the 20th entry of the row B2, since the selection contains ComB6, ComB6, and ComA6. But the $Cost_{scaled}$ is updated to -206.28 when the capacity of the row B2 is 29. Finally, the $Cost_{scaled}$ of the last (39th) entry is -196.76, due to the larger value of sw.

capacity	1	...	15	16	...	38	39
A	0(NULL)	...	0(NULL)	-75.54(ComA6)	...	-75.54(ComA6)	-75.54(ComA6)

(a)

capacity	1	2	...	10	11	...	18
A	0(NULL)	0(NULL)	...	0(NULL)	0(NULL)	...	-75.54(ComA6)
B1	0(NULL)	-70.13(ComB6)	...	-70.13(ComB6)	-60.61(ComB4)	...	-145.67(ComB4)
capacity	19	...	26	27	...	38	39
A	-75.54(ComA6)	...	-75.54(ComA6)	-75.54(ComA6)	...	-75.54(ComA6)	-75.54(ComA6)
B1	-145.67(ComB6)	...	-145.67(ComB6)	-136.15(ComB4)	...	-136.15(ComB4)	-136.15(ComB4)

(b)

capacity	0	1	2	3	4	...	20
A	0(NULL)	0(NULL)	0(NULL)	0(NULL)	0(NULL)	...	-75.54(ComA6)
B1	0(NULL)	0(NULL)	-70.13(ComB6)	-70.13(ComB6)	-70.13(ComB6)	...	-145.67(ComB6)
B2	0(NULL)	0(NULL)	-70.13(ComB6)	-70.13(ComB6)	-140.26(ComB6)	...	-215.80(ComB6)
capacity	21	...	29	...	37	38	39
A	-75.54(ComA6)	...	-75.54(ComA6)	...	-75.54(ComA6)	-75.54(ComA6)	-75.54(ComA6)
B1	-145.67(ComB6)	...	-136.15(ComB4)	...	-136.15(ComB4)	-136.15(ComB4)	-136.15(ComB4)
B2	-215.80(ComB6)	...	-206.28(ComB6)	...	-206.28(ComB6)	-196.76(ComB4)	-196.76(ComB4)

(c)

capacity	0	...	17	...	28	...	39
A	0(NULL)	...	-75.54(ComA6)	...	-75.54(ComA6)	...	-75.54(ComA6)
B1	0(NULL)	...	-60.61(ComB4)	...	-136.15(ComB4)	...	-136.15(ComB4)
B2	0(NULL)	...	-140.26(ComB6)	...	-206.28(ComB6)	...	-196.76(ComB4)

(d)

Fig. 4. The picking up process of the example in the first model (a) Pick up the element for component A (b) Pick up the element for component B1 (c) Pick up the element for component B2 (d) Trace the table back to find the result of this example.

$$\begin{aligned}
TC &= ComA6 + ComB4 + ComB4 + P \\
&= (8+15) + (15+20) + (15+20) + P \\
&= 93 + 87.14 \\
&= 180.14 \\
\\
P &= 10,000 \times (1 - w_{ComA6} \times w_{ComB4} \times w_{ComB4}) \\
&= 10,000 \times 0.008714 \\
&= 87.14 \qquad (11)
\end{aligned}$$

The final selection can be found by tracing back in Fig. 4(d). We select ComB4 while the capacity is 39 in the row B2. Then, we select ComB4 while the capacity is 28 ($39 - sw_{ComB4} = 39 - 11$), and select ComA6 while the capacity is 17 ($28 - sw_{ComB4} = 28 - 11$). Thus, the answer for this example consists of ComA6, ComB4, and ComB4. In other words, the solution is composed of A-3, TA-2, B-2, TB-2, B-2, and TB-2. The summation of the sw of these selected elements is 38 ($16+11+11$) and it is less than the Con_{scaled}. Therefore, this selection meets the constraint. Equation (11) shows the minimum cost of this example with the quality constraint. Our algorithm is summarized in Fig. 5.

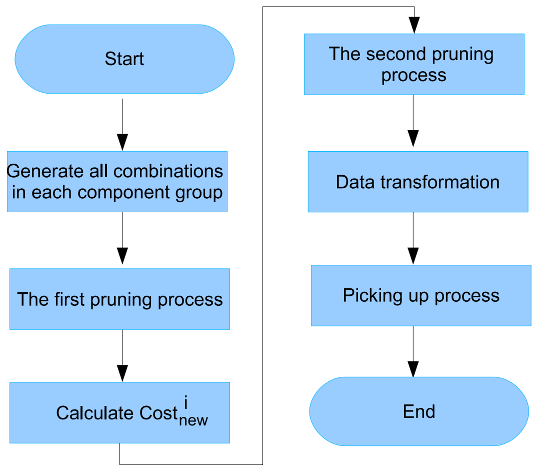
Start
Generate all combinations in each component group
The first pruning process
Calculate Cost$^{i}_{new}$
The second pruning process
Data transformation
Picking up process
End

Fig. 5. The flowchart of quality-cost evaluation model with the quality constraint.

3.2. Defect Level Minimization under the Cost Constraint

Another model that minimizes the DL under the cost constraint is proposed. The formulation is as follows.
Minimize:

$$DL = 1 - \prod_{i=1}^{m} \sum X_i Y_i^{(1-\sum X_i FC_i)} \qquad (12)$$

Subject to:

$$P = Fine \times DL \tag{13}$$

$$(X_1 + X_2 + \ldots + X_{n_1})(X_{n_1+1} + \ldots + X_{n_1+n_2})$$
$$\ldots(X_{n_1+n_2+\ldots+n_{m-1}+1} + \ldots + X_{n_1+n_2+\ldots+n_m}) = 1 \tag{14}$$

$$TC = \sum(X_i C_i) + P \le Con_{cost} \tag{15}$$

First, we also generate all combinations for each component group. Then, we eliminate some redundant elements by the first pruning process. Those remaining elements would be filtered out again by the second pruning process. After the pruning processes, we would pick up the elements from those remaining elements. The picking up process is similar to the one in the previous model, but it has something different. We will find the maximum product of w, since the higher w can bring the lower DL. Moreover, the TC of these selected elements must meet Equation (15). Thus, we need to calculate the updated TC to determine whether the selection meets the constraint at each entry. The optimal solution also can be found at the last entry of the last row.

We use the same example to demonstrate this model. The Con_{cost} is set to 200 in this example. Due to the same pruning processes and data, the remaining elements are the same as shown in Table II. The TC is the capacity of this model, and we record the product of w. In Fig. 6(a), ComA6 is selected at the entry of capacity 60, and the TC of this selection which is calculated by Equation (15) is 59.2. After selecting the element A, we would consider the selection of the element B1 in Fig. 6(b). When the capacity is 61 of the row B1, the TC of the selection with ComB4 is 60.6. Then, ComB6 is selected at the 71th entry of the row B1. However, ComB6 is replaced with ComB4 at the 120th entry of the row B1, since the selection with ComB4 leaves the space for accommodating to ComA6. But when the capacity is 130, the selection with ComB6 is better than that with ComB4. The picking up process of B2 is shown in Fig. 6(c). We select ComB4 at the 181th entry of the row B2, and the product of w is 0.99129 until the 189th entry of the row B2. However, we select ComB6 at the last (200th) entry of the row B2, due to larger w.

When we trace the table back, we need to consider the P_{diff} which is the difference of the penalty when selecting the element i or not as shown in Equation (16). The final selection can be found by tracing back in Fig. 6(d). We select ComB6 when the capacity is 200 in the row B2. Then, we select ComB6 when the capacity is 130 ($200 - Cost_{comB6} - P_{diff} = 200 - 65 - 5$), and select ComA6 when the capacity is 60 ($130 - Cost_{comB6} - P_{diff} = 130 - 65 - 5$). Thus, the result of this example consists of ComB4, ComB4, and ComA6. The DL of this selection is 4640 ppm and the TC is 199.36 as shown in Equation (17).

$$\begin{aligned} P_{diff} &= Fine \times \{(1 - w_{new}) - (1 - w_{old})\} \\ &= Fine \times (w_{old} - w_{new}) \end{aligned} \tag{16}$$

$$\begin{aligned} &TC = ComA6 + ComB6 + ComB6 + P \\ &= (8 + 15) + (15 + 50) + (15 + 50) + P \\ &= 153 + 46.36 \\ &= 199.36 \\ \\ &P = 10,000 \times (1 - w_{ComA6} \times w_{ComB6} \times w_{ComB6}) \\ &= 10,000 \times 0.004636 \\ &= 46.36 \end{aligned} \tag{17}$$

4. Experimental Results

The experiment was implemented on an INTEL® Xeon® 3.0GHz GUN/Linux workstation with 32GBytes memory. The experimental results are shown in Table III. In Table III, we show the CPU time with respect to the problem size.

TABLE III
THE CPU TIME COMPARISON BETWEEN BASIC APPROACH AND OUR APPROACH.

	$Time(Sec.)$		
		$Our\ Approach$	
$\|Var.\|$	$Basic$	$The\ First\ Model$	$The\ Second\ Model$
10	$\sim$ 0.00	$\sim$ 0.00	$\sim$ 0.00
20	$\sim$ 0.00	$\sim$ 0.00	$\sim$ 0.00
50	0.03	$\sim$ 0.00	$\sim$ 0.00
100	2767.52	$\sim$ 0.00	$\sim$ 0.00
300	-	0.02	0.02
500	-	0.03	0.03
700	-	0.05	0.04
800	-	2.87	0.06
900	-	3.26	0.08
1,000	-	366.78	0.08

In this experiment, we compare the CPU time of our approach with the basic approach. The basic approach is an exhaustive one that tries all combinations in the benchmark and record the solution for comparison. Furthermore, the basic approach simultaneously calculates the results for these two models. Ten benchmarks ranged from 10 to 1,000 variables are used in this experiment. In the benchmark with 100 variables, our approach is more efficient than the basic approach. Furthermore, the basic approach cannot solve the case with 300 variables. For the 1,000 variables problem, however, our approach still can solve it.

5. Conclusion

In this work, we have proposed two models with different requirements for SiP integrators. The first model is the cost minimization with the quality constraint, and the other one is the quality enhancement with the cost constraint. In addition to the models, the corresponding algorithms for solving the problems are also proposed. The proposed algorithms are more efficient than the basic approach according to the experimental results. These models can be used to find the optimal combination for an SiP design.

REFERENCES

[1] M. Abadir, "Economics Modeling of Multichip Modules Testing Strategies," *IEEE Transactions on Components, Packaging, and Manufacturing Technology*, vol. 21, no. 4, pp. 360-370, Nov, 1998.

[2] D. Appello, P. Bernardi, M. Grosso, and M. S. Reorda, "System-in-Package Testing: Problems and Solutions," *IEEE Design and Test of Computers*, vol. 23, no. 3, pp. 203-211, May/Jun, 2006.

capacity	0	...	23	...	60	...	200
A	0(NULL)	...	0(NULL)	...	0.99638(ComA6)	...	0.99638(ComA6)

(a)

capacity	61	...	71	...	120	...	130
A	0.99638(ComA6)	...	0.99638(ComA6)	...	0.99638(ComA6)	...	0.99638(ComA6)
B1	0.99744(ComB4)	...	0.99949(ComB6)	...	0.99382(ComB4)	...	0.99587(ComB6)

(b)

capacity	181	...	189	190	...	199	200
A	0.99638(ComA6)	...	0.99638(ComA6)	0.99638(ComA6)	...	0.99638(ComA6)	0.99638(ComA6)
B1	0.99587(ComB6)	...	0.99587(ComB6)	0.99587(ComB6)	...	0.99587(ComB6)	0.99587(ComB6)
B2	0.99129(ComB4)	...	0.99129(ComB4)	0.99332(ComB4)	...	0.99332(ComB4)	0.99536(ComB6)

(c)

capacity	60	...	129	130	...	199	200
A	0.99638(ComA6)	...	0.99638(ComA6)	0.99638(ComA6)	...	0.99638(ComA6)	0.99638(ComA6)
B1	...	...	0.99587(ComB6)	0.99587(ComB6)	...	0.99587(ComB6)	0.99587(ComB6)
B2	...	...	...	...	...	0.99129(ComB4)	0.99536(ComB6)

(d)

Fig. 6. The picking up process of the example in the second model (a) Pick up the element for component A (b) Pick up the element for component B1 (c) Pick up the element for component B2 (d) Trace the table back to find the result of this example.

[3] F. Corsi, S. Martino, and T. W. Williams, “Defect Level As a Function of Fault Coverage and Yield,” in *Proc. European Test Conference*, pp. 507-508, 1993.

[4] V. K. Kim, T. Chen, and M. Tegethoff, “ASIC Manufacturing Test Cost Prediction at Early Design Stage,” in *Proc. International Test Conference*, pp. 356-361, 1997.

[5] S.-K. Lu, T.-Y. Lee, and C.-W. Wu, “A Profit Evaluation System (PES) for Logic Cores at Early Design Stage,” in *Proc. International Conference on Electronics, Circuits and Systems*, vol. 3, pp. 1491-1494, 2001.

[6] B. McCaffrey, “Exploring the Challenges in Creating a High-quality Mainstream Design Solution for System-in-Package (SiP) Design,” in *Proc. International Symposium Quality of Electronic Design*, pp. 556-561, 2005.

[7] A. A. Setty, H. L. Martin, “BIST and Interconnect Testing with Boundary Scan,” in *Proc. Southeastcon*, vol. 1, pp. 12-15, Apr, 1991.

[8] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli, “SIS: A System for sequential circuit synthesis, Technical Report UCB/ERL M92/41, Electronics Research Lab, Univ. of California, Berkeley, CA 94720, 1992.

[9] K. L. Tai, ”System-In-Package (SiP): Challenges and Opportunities,” in *Proc. Asia and South Pacific Design Automation Conference*, pp. 191-196, 2000.

[10] E. J. Vardaman, “Is a Known Good Die Hard to Find?,” in *Proc. IEEE Multi-Chip Module Conference*, pp. 8, 1996.