Restructuring and Logic Minimization for Testable PLA

Gwo-Haur Hwang, *Student Member, IEEE*, and Wen-Zen Shen, *Member, IEEE*

***Abstract*—Based on the testable design techniques proposed in [12], [13], we first derive a testability measure named UCP (Untestability Cube-number Product) which can accurately indicate the extra logic needed in testable PLA. Using UCP as a cost function, we have developed two new algorithms. The first one is a restructuring algorithm named REST, and the other is a logic minimizer for testable PLA named LMTPLA.**

REST can not only restructure a minimized PLA and improve its UCP, but also preserve the same logic function and almost the same chip area simultaneously. Thus, we can make the restructured PLA testable by taking less extra hardware. In order to get a minimum UCP, two new techniques are proposed: 1) Cube_Reduction, and 2) Cube_Partition. Using Cube_Reduction, we can make cubes far from the primes and increases the testability. By Cube_Partition, we may find out those hard-to-test cubes and partition them into smaller but easier-to-test cubes.

LMTPLA is principally based on ESPRESSO-II and REST. Different from other logic minimizers, it can consider the testability at the logic minimization process. In order to minimize UCP as well as the number of product terms, four strategies are developed: 1) deleting the cubes with poor testability and reserving the cubes with good testability, 2) giving up the primes, 3) if necessary, partitioning the more untestable cubes into smaller cubes, and 4) deleting the procedures which are useless in LMTPLA.

REST and LMTPLA have been implemented on SUN4/260 in C language. For 40 benchmark circuits, the hardware overheads required are reduced by about 30–40%.

I. Introduction

PROGRAMMABLE Logic Array (PLA) [1] is a good element for combinational and sequential logic circuits in VLSI for its simplicity, regularity and flexibility. However, besides the classical stuck-at and bridging faults, PLA has another important class of faults called crosspoint faults [2], [3]. In the past, many powerful algorithms were proposed for PLA test pattern generation [2]–[7]. Complete fault coverage (detecting all testable faults and identifying all redundant faults) can be guaranteed in PLATYPUS [6] and PLANET [7]. However, these algorithms are based on the single-fault assumption (i.e., single stuck-at or single extra/missing device). As for multiple faults such as multiple stuck-at faults, multiple extra and multiple missing faults, the testable design is a good choice. In this paper, testable design techniques are emphasized on the multiple faults.

In the past, many structured design-for-testability techniques have been suggested [8]–[15]. These techniques require extra hardware to increase the controllability and/or the observability of the product lines to make it fully testable. Some of these techniques [8]–[11] utilize shift-registers to control the product lines. They are considered as scan-type design. Since a shift register cell is wider than a product line, it causes a pitch mismatch problem. Although the mismatch problem can be solved by multiplexing, the area overhead is still too high. Other techniques as shown in Fig. 1 [12], [13] require extra inputs on the extended AND plane, and are considered as decoded-type design. Because there is no pitch mismatch problem and less area overhead is used, we choose decoded-type as our testable PLA design method. In this way, a PLA restructuring algorithm (REST) and a new logic minimizer (LMTPLA: Logic Minimization for Testable PLA) are developed to reduce the hardware overhead in decoded-type design. Recently, some techniques combining scan-type and decoded-type are proposed [14], [15]. We refer to them as mixed-type design. This technique partitions a PLA into several parts and use a shift-register to control each part. Thus a shorter shift register and fewer extra inputs are used to make each part fully testable. Our REST and LMTPLA can also be applied to mixed-type design.

In [12], a fully testable design technique was presented. If the Hamming distance between any two product terms is larger than or equals to 2, the PLA is fully testable, i.e., no redundant fault. Furthermore, using this technique, all multiple stuck-at faults, as well as all multiple extra and multiple missing device faults, are detected. The test set is a byproduct when the testable design is finished. Hence, test pattern generation is unnecessary and redundant faults do not exist. However, if the initial Hamming distance of any two product lines is smaller than 2, then we must add some extra inputs to increase the Hamming distance until all distances are larger than or equal to 2.

Fig. 2 shows the conventional approach of testable PLA design. The logic minimizers can reduce the chip area of PLA's to a minimum, but the testable design algorithm may enlarge it very much due to the poor testability of the minimized PLA's. Given a PLA specification, there may

Manuscript received December 12, 1990; revised April 20, 1992. This work was supported by the National Science Council, Republic of China, under Grant NSC79-0404-E009-24. This paper was recommended by Associate Editor R. K. Brayton.

The authors are with the Department of Electronics Engineering and Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan R.O.C.

IEEE Log Number 9202995.

0278-0070/93$03.00 © 1993 IEEE

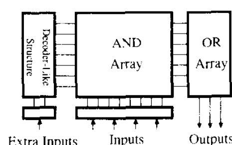

Fig. 1. Testable PLA using extra inputs and forming a decoder like structure on extended AND array.

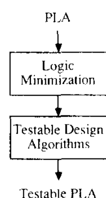

Fig. 2. Conventional approaches of testable PLA design.

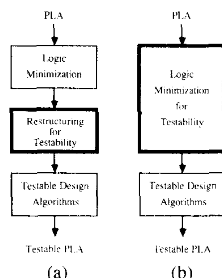

Fig. 3. New approaches of testable PLA design. (a) Using restructuring for testability. (b) Using logic minimization for testability.

exist many different PLA logic structures that are equivalent in logic function and chip area. But, the extra overheads are different when applying testable design techniques. Therefore, how to choose a PLA with the least extra overhead becomes an important issue in testable design. In Fig. 3, we suggest two new approaches to design the testable PLA. One new idea as shown in Fig. 3(a) is to enhance the testability of the minimized PLA's by restructuring them. Here, restructuring means changing the structure of PLA's while keeping the same logic functions. The structure means the position of devices or contacts in AND plane and OR plane. Then we can map the restructured PLA into a testable PLA by adding less extra hardware, i.e., less extra inputs as shown in Fig. 1. The other approach as shown in Fig. 3(b) is to combine the restructuring algorithm and the logic minimization algorithm to form a new logic minimization algorithm which considers both the testability and the number of product terms. The objective too, is to reduce the extra inputs of testable design.

In Section II, some basic definitions and techniques of testable PLA are presented [12], [13]. Section III describes the testability analysis of PLA's. The cost function, UCP (Untestability Cube-number Product), is also defined in this section. In Section IV, the restructuring algorithm, REST, is proposed. In Section V, we combine REST and ESPRESSO-II to be LMTPLA. In Section VI, we show the simulation results of REST and LMTPLA, and compare them to that of conventional approach without restructuring. Finally, we offer some concluding remarks in Section VII.

II. Preliminaries

First, we define some terms that will be used later. Then the techniques of testable PLA which had been proposed by Bozorgui-Nesbat [12] and Khakbaz [13] are demonstrated by an example.

2.1. Basic Definitions

Fig. 4(a) shows a small NOR-NOR PLA. Now, we assume that a PLA has n input lines, p product lines, and m output lines. For simplicity, we use the characteristic matrix to describe a PLA.

Definition—Characteristic Matrix of a PLA: The characteristic matrix of a PLA is a p-by-$(n + m)$ matrix denoted by M whose entries are defined as follows: $M(i, j) = 0$, if a device exists on the intersection of the ith product line and the true bit line of the jth input. $M(i, j) = 1$, if a device exists on the intersection of the ith product line and the complement bit line of the jth input. $M(i, j) = 2$, if no device exists on the intersection of the ith product line with either the true or the complement bit line of jth input. $M(i, j) = 3$, if no device exists on the intersection of the ith product line and the $(j-n)$th output line. $M(i, j) = 4$, if a device exists on the intersection of the ith product line and the $(j-n)$th output line.

The characteristic matrix of the PLA of Fig. 4(a) is shown in Fig. 4(b).

Definition—Cube: The ith cube of a PLA is a row vector $c^i = [c_1^i, c_2^i, \cdots, c_n^i, c_{n+1}^i, \cdots, c_{n+m}^i]$, where $c_j^i = M(i, j)$.

Definition—Input Part of c^i: The input part of c^i is a row vector $I(c^i) = [c_1^i, c_2^i, \cdots, c_n^i]$

Definition—Select Set of P_i [12]: The select set of the ith product line P_i is S_i, which is a set of input vectors. Any member of S_i, when applied to the inputs of PLA, activates P_i. In fact, the description of S_i in cubic form is the same as $I(c^i)$.

Definition—Test Set of P_i [12]: Any nonempty subset of S_i is called a test set of P_i. A test set of P_i is denoted as T_i.

Definition—Distance Matrix of a PLA [12]: Given the select set S_i and a test set T_i for each product line P_i, the distance matrix D is a p by p matrix whose entries are defined as follows.

If $i = j$, then $D(i, j) = 2$.

If $i \neq j$, then $D(i, j)$ = the minimal Hamming distance between any member of T_i and any member of S_j.

Definition—MD_matrix of a PLA: The MD_matrix of a PLA is a p by p matrix whose entries are derived from the D matrix and are modified by two procedures: 1) Test set selection, 2) OR plane consideration. The two procedures are proposed by [12] and [13], respectively, and are demonstrated in Section 2.2. The entries of MD_matrix larger than 2 should be set to 2.

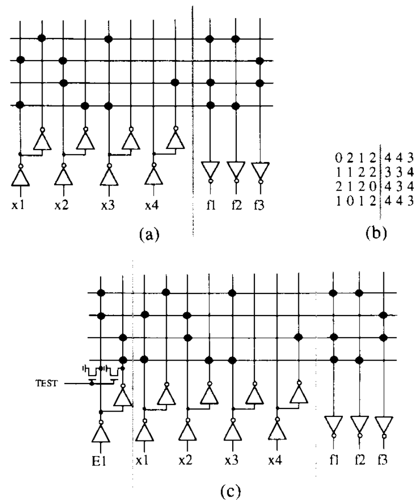

Fig. 4. (a) A small NOR-NOR PLA. (b) Characteristic matrix of a PLA. (c) Testable structure of a PLA.

2.2. Testable PLA Technique

The testable PLA design method is based on [12] and [13]. If all the entries of D matrix are larger than or equal to 2, the PLA is fully testable. Otherwise, it is not fully testable. There are three procedures to increase the entries of D matrix: 1) by judiciously selecting among many input patterns available in T_i [12], 2) by considering the output part of the characteristic matrix, i.e., OR plane characteristic [13], and 3) by assigning additional test inputs to the PLA to change its characteristic matrix. Here, we use an example to demonstrate the techniques.

Example 1: Modify the PLA in Fig. 4(a) to be fully testable.

The characteristic matrix of the PLA is

$$c^1 = 0212443$$
$$c^2 = 1122334$$
$$c^3 = 2120434$$
$$c^4 = 1012443.$$

The select set can be derived from the characteristic matrix. In fact, the input part of each cube forms the select set of the corresponding cube as follows:

$$S_1 = I(c^1) = 0212$$
$$S_2 = I(c^2) = 1122$$
$$S_3 = I(c^3) = 2120$$
$$S_4 = I(c^4) = 1012.$$

For instance, it is obvious that four minterms can activate product line P_1, i.e., 0010, 0011, 0110, and 0111. Of course, each minterm can be chosen as a test. Initially, we hold all minterms in the test set so that

$$T_1 = 0212$$
$$T_2 = 1122$$
$$T_3 = 2120$$
$$T_4 = 1012.$$

From the initial test set T_i and the select set S_j, we can easily derive the initial D_matrix from the definition of D_{ij} as follows:

$$\begin{bmatrix} 2 & 1 & 0 & 1 \\ 1 & 2 & 1 & 1 \\ 0 & 0 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix}.$$

For example, D_{31} is 0. It means that there exists a test pattern 0110 in T_3 which may activate both P_3 and P_1 concurrently. So, 0110 is not a good test pattern for P_3. Now, the problem becomes how to assign "1" or "0" to all 2's in T_i's and select a test set such that the entries of the D_matrix are maximized. Bozorgui-Nesbat et al. [12] had proposed a heuristic algorithm to find out the optimal solution. The resultant test set is

$$T_1 = 0011$$
$$T_2 = 1101$$
$$T_3 = 0100$$
$$T_4 = 1011.$$

The corresponding D_matrix is

$$\begin{bmatrix} 2 & 2 & 2 & 1 \\ 2 & 2 & 1 & 2 \\ 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \end{bmatrix}.$$

We find that there still exists six entries less than 2. In [13], it is suggested that we can take the OR plane conditions into consideration. For example, originally $D_{31} = 1$ and, for the third output of the PLA, there exists a 4 on c^3 but a 3 on c^1. In other words, when we reverse any bit of T_3 to detect the crosspoint faults on P_3, the fault effect will not be masked by c^1. Hence, D_{31} can be increased to 2. Similarly, we have $D_{32} = 2$ and $D_{42} = 2$. So, after considering the OR plane characteristic, we obtain the MD_matrix as follows:

$$MD = \begin{bmatrix} 2 & 2 & 2 & 1 \\ 2 & 2 & 1 & 2 \\ 2 & 2 & 2 & 2 \\ 1 & 2 & 2 & 2 \end{bmatrix}.$$

Since there still exists three non-2's entries in the *MD*_matrix, so we need to add extra inputs to further increase the entries of the *MD*_matrix [12]. Here, the problem is "How to design a minimum number of extra inputs such that all the entries of the *MD*_matrix are 2?" In [12], some heuristics have been developed that use local optimization to assign these inputs in an effective manner. The solution is

Extra Input		Initial PLA		New PLA
1		0212 443		10212 443
1	+	1122 334	=	11122 334
0		2120 434		02120 434.
0		1012 443		01012 443

By adding this extra input, the final *MD*_matrix becomes

Increasing Distance		Initial *MD*_matrix		Final *MD*_matrix

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 2 & 2 & 2 & 1 \\ 2 & 2 & 1 & 2 \\ 2 & 2 & 2 & 2 \\ 1 & 2 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 \end{bmatrix}$$

Fig. 4(c) shows the testable structure of the addition of one extra input. The control signal, TEST, is set to be 0 in testing mode and 1 in normal mode. Since the extra area overhead introduced by the TEST is very little, so the total area overhead is proportional to the number of extra inputs (not including the TEST).

III. Testability Analysis of PLA

Given a PLA, we begin the testability analysis from the *MD*_matrix as defined in Section II. As demonstrated by Example 1, it is clear that the PLA with larger entries needs fewer extra inputs. In other words, the larger the entries, the higher the testability. However, testability analysis requires a testability measure by which the testability of a circuit can be quantified as accurately as possible. It seems that the summation of all entries of the *MD*_matrix can be viewed as a testability measure. From the definition of *MD*_matrix, the diagonal entries are always 2 and the other entries vary between 0 and 2. If we ignore the diagonal entries, the summation is between 0 and $2p(p - 1)$. To reflect the ease or the difficulty of applying testable design technique, this measure is normalized between 0 and 1. Now, we can define the testability (T) and untestability (U) of a PLA from the *MD*_matrix as follows:

Definition—Testability of a PLA: The testability of a PLA is denoted by T and defined as

$$T = \left(\sum_{i=1}^{p} \sum_{j=1}^{p} MD_{ij} - 2p \right) \Big/ [2p(p - 1)]$$

where p is the number of cubes.

Definition—Untestability of a PLA: The untestability of a PLA is denoted by U and defined as $U = 1 - T$.

From the above definitions, it is clear that both the testability and untestability of PLA's vary between 0 and 1. If $T = 1$, then the PLA is fully testable without any extra inputs. Otherwise, it's necessary to add some extra inputs. In general, PLA's with more cubes need more extra inputs to form a testable PLA. Similarly, PLAs with larger untestability need more extra inputs too. By combining these two facts, we define a value called UCP as follows:

Definition—Untestability-Cube_number Product: The untestability cube_number product is denoted by UCP and defined as $\text{UCP} = U * p$, where p is the number of cubes and U is the untestability of the PLA.

From this definition, we see that UCP can be used to evaluate approximately the number of extra inputs needed.

Definition—Estimated Number of Extra Inputs: The estimated number of extra inputs needed by testable PLA is denoted by E and defined as $E = \log_2(2 * \text{UCP} + 1)$.

In fact, we can estimate the number of extra inputs as above. There are three special cases matching the estimation as (for $p >> 1$):

1) If all the entries of *MD*_matrix are 2, it is obvious that the number of extra input is 0. In this special case, we find $T = 1$ and $U = 0$. This implies $\text{UCP} = 0$ and so the estimated number of extra inputs, $E = \log_2(1) = 0$.

2) If all the entries of the *MD*_matrix are 1, then extra inputs are assigned until the entries are increased to 2. This can be done by adding $\lceil \log_2(p) \rceil$ extra inputs and forming a binary code in the extended AND plane. In this special case, we find $T = U = 0.5$ and $\text{UCP} = p/2$, and thus $E = \log_2(p + 1)$. For $p >> 1$, E is close to the value of $\lceil \log_2(p) \rceil$.

3) If all the entries of the *MD*_matrix are 0, we can add $\lceil \log_2(p) \rceil$ extra inputs and form a binary code in the extended AND plane as that of special case (2). This makes all the entries of the *MD*_matrix 1 or 2. Then, we can easily add a new extra input and assign a parity code with those of $\lceil \log_2(p) \rceil$ inputs such that all the entries are increased to 2. Therefore, the number of extra inputs is $\lceil \log_2(p) \rceil + 1$. In this special case, we find $T = 0$ and $U = 1$. This implies $\text{UCP} = p$ and $E = \log_2(2p + 1)$. For $p >> 1$, E is close to the value of $\lceil \log_2(p) \rceil + 1$.

From the above testability analysis, we know that a PLA with smaller UCP needs less extra inputs. Hence, UCP is used as the cost function in REST and LMTPLA.

Example 2: For the PLA shown in Fig. 4, estimate the number of extra inputs.

From example 1, the *MD*_matrix is as follows:

$$MD = \begin{bmatrix} 2 & 2 & 2 & 1 \\ 2 & 2 & 1 & 2 \\ 2 & 2 & 2 & 2 \\ 1 & 2 & 2 & 2 \end{bmatrix}.$$

According to the definition, the values of T, U, UCP, and E are:

$$T = 21/24 = 0.875$$
$$U = 1 - T = 0.125$$
$$\text{UCP} = U * p = 0.125 * 4 = 0.5$$
$$E = \log_2(2 * \text{UCP} + 1) = 1.$$

The estimated number of extra inputs matches the real number of extra inputs in Example 1.

IV. Restructuring Algorithm

In this section, we describe a PLA restructuring algorithm REST. Fig. 5 shows the procedure for REST. It consists of two subroutines: Cube_Reduction and Cube_Partition. The objective is to optimize UCP of the minimized PLA's as shown in Fig. 3(a).

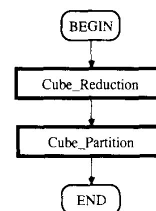

Fig. 5. Flowchart of REST.

4.1. Cube_Reduction

For a minimized PLA, if cubes are all primes, the entries of the *MD*_matrix are usually small. This implies that the PLA has poor testability. For this reason, we must make all cubes as small as possible. Using Cube_Reduction, we can transform a prime cover into a new (in general, not prime) cover, by replacing each cube with a (usually smaller) cube contained in it. This method makes cubes far from prime and thus increases the testability. Cube_Reduction is similar to the Reduce of ESPRESSO-II [16]. In order to enhance the effect of increasing testability, the order of cubes to be reduced must be considered carefully. In Reduce [16], it is suggested that cubes are reduced from the largest one to the smallest. In Cube_Reduction, the order is much the same as Reduce except when cubes are of equal size. In this case, the cube with poor testability will be reduced first. So we need a testability measure of a cube. Recall that if the entries of the ith row of the *MD*_matrix are all 2's, the ith cube is fully testable. It seems that the sum of the entries of the ith row in the *MD*_matrix can be considered as a measure of the testability for the ith cube. So, we define a value called distance weight as the testability of a cube.

Definition—The Distance Weight of the ith cube: The distance weight of the ith cube is denoted by $W(i)$ and defined as $W(i) = MD_{i1} + MD_{i2} + \cdots + MD_{ip}$, where p is the number of cubes.

If $W(i) = 2p$, then the ith cube is fully testable. Note that the minimal value of $W(i)$ is 2 since the diagonal of the *MD*_matrix is always 2.

Fig. 6 shows the flowchart of Cube_Reduction, the order of cubes to be reduced are as follows:

1) reduce the cubes from the largest one to the smallest;

2) if cubes are of equal size, the one with smaller distance weight is reduced first.

The following example demonstrates the algorithm of Cube_Reduction.

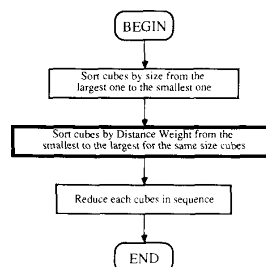

Fig. 6. Flowchart of CUBE_REDUCTION.

Example 3: For the PLA shown in Fig. 4, determine the order of cubes.

First, we compute the distance weight of each cube.

$$W(1) = 2 + 2 + 2 + 1 = 7$$
$$W(2) = 2 + 2 + 1 + 2 = 7$$
$$W(3) = 2 + 2 + 2 + 2 = 8$$
$$W(4) = 1 + 2 + 2 + 2 = 7.$$

The third cube (1122334) is fully testable because $W(3) = 2p = 8$.

Now, we can determine the order of cubes:

Step 1: Sort all cubes from the largest one to the smallest one. (The largest one is defined as the cube that has the greatest number of 2's or 4's)

$c^1 = 0212443$ The number of 2's or 4's is 4.

$c^3 = 2120434$ The number of 2's or 4's is 4.

$c^2 = 1122334$ The number of 2's or 4's is 3.

$c^4 = 1012443$ The number of 2's or 4's is 3.

Step 2: For cubes of equal size, compare the distance weight. c^1 and c^3 are of equal size. But, $W(3)$ is larger than $W(1)$. Thus c^1 must be reduced first.

4.2. Cube_Partition

In PLA's, there may exist some hard-to-test cubes. These cubes usually produce many 0 entries in the *MD*_matrix. Cube_Partition can find out the hardest-to-test cube, called *P*_cube, which is defined as the cube that adjoins the most of other cubes. Then, we can partition the *P*_cube into two half cubes by a *P*_variable. In this

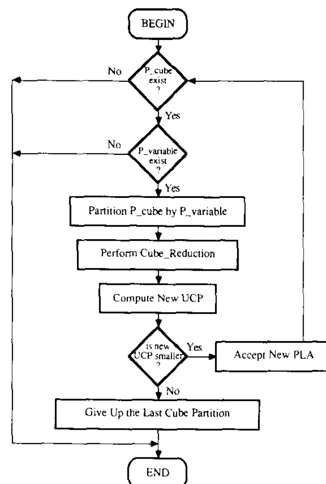

Fig. 7. Flowchart of CUBE_PARTITION.

way, the untestability U is decreased, but the number of cubes is increased by one. Usually, this may generate a smaller UCP. Thus we have an opportunity to trade one extra input to one extra product line. The flowchart of Cube_Partition is shown in Fig. 7 and described as follows:

1) Find a row with the greatest number of zeros in the *MD*_matrix. The corresponding cube is called the *P*_cube.

2) For each don't care variable in the *P*_cube, say the ith variable, count the number of cubes that satisfy the following two conditions

   1) The distance between the cube and the *P*_cube is 0.
   2) The ith variable of the cube is not don't-care.

   Then, find the variable with maximal counting. This variable is called the *P*_variable.

3) Partition the *P*_cube into two cubes with respect to the *P*_variable.

As an example, consider a *P*_cube 2210 and its *P*_variable (the first variable), the *P*_cube will be partitioned into two half cubes, 0210 and 1210. Note that any don't-care variable in the *P*_cube used for partition will keep the total number of zeros the same. However, after this step, the number of cubes will be increase by one. So, the testability is improved.

4) Perform Cube_Reduction on this PLA.

A PLA may be reduced again when a cube is partitioned into two half cubes. Hence, running Cube_Reduction after partition is necessary in order to keep cubes of the PLA as small as possible.

5) If the UCP of the new PLA is smaller than the old one, then we accept it and go to Step 1). Otherwise, give up the last partition and stop.

As mentioned previously, the cost function, UCP, may be increased or decreased. If the new UCP is smaller than the old one, it seems that the partition is worthy. Otherwise, we give up the last partition and terminate the Cube_Partition procedure.

V. Logic Minimization for Testable PLA

In this section, we combine the logic minimization algorithm and restructured algorithm into a new algorithm called LMTPLA (Logic Minimization for Testable PLA). The main body of logic minimization algorithm is based on ESPRESSO-II which is described in [16] in detail. Fig. 8 shows the flowchart of ESPRESSO-II. As compared to ESPRESSO-II, the flowchart of LMTPLA is shown in Fig. 9. The objective of LMTPLA is to minimize UCP during the logic minimization process. In order to get a highly testable PLA, the following four strategies are used in LMTPLA.

1) Delete the cubes with poor testability and reserve the cubes with good testability.

The results of Reduce and Expand in ESPRESSO-II depend on the order in which the cubes are processed. The cube which has the largest size should be reduced or expanded first in the Reduce and Expand, respectively. As discussed in Section IV, when cubes are of equal size, a cube with the smallest distance weight will be reduced first. However, on the other hand, a cube with the largest distance weight is expanded first in the Expand procedure. The reason behind this strategy is that the cube reduced first would be deleted most easily and the cube expanded first would be conserved most easily. This strategy is embodied in the Reduce and Expand, the resultant procedures are called Cube_Reduction and Cube_Expand, respectively. The other modified procedure based on this strategy is the Irredundant procedure, which consists of three subroutines: Redundant, Partially_Redundant, and Minimal_Irredundant. The Redundant subroutine can find out all the redundant cubes. The redundant cubes can be partitioned into the totally redundant cubes and the partially redundant cubes. The totally redundant cubes are covered by the union of the non-redundant cubes and the don't-care set, so they are deleted. The Partially_Redundant subroutine can delete the totally redundant cubes and get the partially redundant cubes. From the partially redundant cubes, the Minimal_Irredundant subroutine finds the minimal irredundant cover by deleting nearly the greatest number of cubes [16]. In LMTPLA, we rewrite the Minimal_Irredundant to delete the cube one by one with the least distance weight first until the cover becomes irredundant. In other words, the cubes with poor testability are deleted first. We call the new Irredundant procedure the Cube_Irredundant.

2) Give up the primes.

Conventional logic minimizers always make all cubes primes. This usually introduces larger U. As discussed in Section IV, the procedure Cube_Reduction is added after

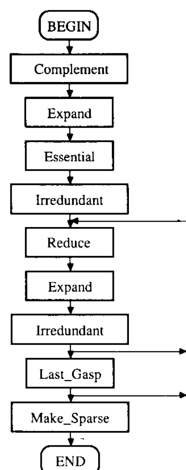

Fig. 8. Flowchart of ESPRESSO-II.

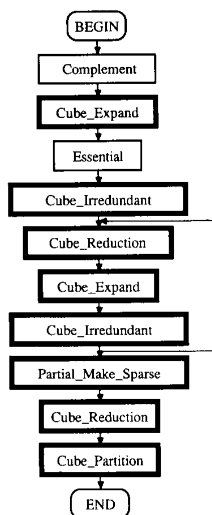

Fig. 9. Flowchart of LMTPLA.

the Make_Sparse procedure to reduce the size of the cubes and thus increase the testability of the PLA's.

3) If necessary, partition the more untestable cubes into smaller cubes.

Similar to REST, we add Cube_Partition followed by Cube_Reduction in LMTPLA to partition the hard-to-test cube into smaller cubes but easier to test.

4) Delete the procedures which are useless in LMTPLA.

Because of different cost functions between ESPRESSO-II and LMTPLA, some procedures in ESPRESSO-II become useless in LMTPLA. In ESPRESSO-II, there is a procedure called Last_Gasp which has very little effect on the minimization but spends much CPU time. So, we remove this procedure. In addition, ESPRESSO-II has an important procedure called Make_Sparse. As the name suggested, it attempts to make the PLA as sparse as possible. Make_Sparse consists of two subroutines: Lower_Outs and Rise_In. In general, Lower_Outs reduces the number of 4's in the OR plane. This allows us to make better use of the OR plane to improve T, i.e., convert more 0's and 1's to 2's in the MD_matrix. Hence, U is decreased. On the other hand, Rise_In increases the number of 2's and the AND plane. According to the definition of distance matrix, this introduces more 0's in the MD_matrix. So, U is increased. Moreover, its effect will be cancelled by Cube_Reduction. Therefore, Rise_In in Make_Sparse is eliminated. In LMTPLA, only Lower_Outs is included in the Partial_Make_Sparse procedure.

Now, we can compare ESPRESSO-II with LMTPLA as follows:

1) Complement and Essential are the same in ESPRESSO-II and LMTPLA.
2) Expand, Reduce, Irredundant and Make_Sparse in ESPRESSO-II are modified to be Cube_Expand, Cube_Reduction, Cube_Irredundant and Partial_Make_Sparse in LMTPLA, respectively.
3) Last_Gasp in ESPRESSO-II is removed in LMTPLA.
4) Cube_Reduction and Cube_Partition are added as the final steps in LMTPLA.

VI. Experimental Results and Comparisons

REST and LMTPLA have been implemented on SUN workstation in C language. We have applied REST to the 40 benchmark circuits [16] where ESPRESSO-MV is employed as the front end minimization tool. Table I summarizes the experimental results obtained with ESPRESSO-MV and REST. Columns 1-3 lists the names of the PLA's and the numbers of input and output variables. Column 4 shows the number of product terms obtained with ESPRESSO-MV. The total number of cubes is 2731. After the application of testable design technique as described in section II, extra inputs are added to obtain 100% testability of multiple struck-at faults, as well as multiple extra and multiple missing device faults. Column 6 indicates the number of extra inputs needed for each PLA. The total number of extra input is 168. However, if we run REST after ESPRESSO-MV, the total number of cubes is slightly increased from 2731 to 2751. The increasing rate is approximately 0.7%. The results are listed in column 5. Column 7 shows the number of extra inputs needed after running the REST algorithm. The total number of extra inputs is reduced dramatically from 168 to 105. The average reduction ratio is approximately 37%. The PLA "mish", for example, can be minimized to as 94 input variables, 43 output variables, 82 product terms, and 6 extra inputs with ESPRESSO-MV. After running REST, the number of product terms remains the same. However, the number of extra inputs is decreased to 3, which means a 50% reduction. The area overhead which is defined as the percentage of extra area over the original

TABLE I
REDUCTION OF EXTRA INPUTS BY REST

	I/O		Cubes		Extra Inputs	
PLA	in	out	ESP	REST	ESP	REST
adr4	8	5	75	75	8	2
alu1	12	8	19	19	3	2
alu2	10	8	68	68	7	4
alu3	10	8	66	66	7	3
apla	10	12	25	25	1	1
chkn	29	7	140	141	7	5
co14	14	1	14	14	0	0
dc1	4	7	9	9	2	2
dc2	8	7	39	43	3	2
dist	8	5	123	125	5	3
dk17	10	11	18	19	2	1
dk27	9	9	10	10	1	1
dk48	15	17	22	22	1	1
f51m	8	8	77	77	6	2
gary	15	11	107	108	4	3
in0	15	11	107	108	4	3
in1	16	17	106	107	4	3
in2	19	10	136	136	5	3
in3	35	29	74	74	4	4
in5	24	14	62	63	3	3
in6	33	23	54	55	3	2
in7	26	10	54	54	4	3
misg	56	23	69	70	8	5
mish	94	43	82	82	6	3
mlp4	8	8	128	129	5	3
radd	8	5	75	75	8	2
rd53	5	3	31	31	6	2
rd73	7	3	127	127	3	2
risc	8	31	29	30	2	1
root	8	5	57	57	4	3
sqn	7	3	38	39	3	3
sqn6	6	12	49	49	4	2
vg2	25	8	110	110	4	3
wim	4	7	9	9	3	3
x1dn	27	6	110	110	4	4
x6dn	39	5	82	82	5	4
x9dn	27	7	120	120	5	4
z4	7	4	59	59	4	2
5xp1	7	10	65	67	6	2
9sym	9	1	86	87	4	4
40			2731	2751	168	105

TABLE II
REDUCTION OF EXTRA INPUTS BY LMTPLA

	I/O		Cubes		Extra Inputs	
PLA	in	out	GHH	LMTPLA	GHH	LMTPLA
adr4	8	5	75	75	3	2
alu1	12	8	19	19	3	2
alu2	10	8	68	70	8	3
alu3	10	8	66	66	6	3
apla	10	12	26	25	2	1
chkn	29	7	144	144	7	4
co14	14	1	14	14	0	0
dc1	4	7	9	12	2	1
dc2	8	7	40	40	2	2
dist	8	5	126	125	4	3
dk17	10	11	18	18	2	1
dk27	9	9	10	10	1	0
dk48	15	17	22	22	1	1
f51m	8	8	78	78	5	3
gary	15	11	107	109	4	3
in0	15	11	107	108	4	3
in1	16	17	104	105	4	3
in2	19	10	137	137	4	3
in3	35	29	74	74	4	4
in5	24	14	62	65	3	2
in6	33	23	54	55	3	2
in7	26	10	54	54	4	4
misg	56	23	69	73	8	6
mish	94	43	82	82	6	3
mlp4	8	8	139	139	3	3
radd	8	5	75	75	3	2
rd53	5	3	31	31	4	1
rd73	7	3	127	127	3	2
risc	8	31	28	28	2	1
root	8	5	58	57	4	3
sqn	7	3	39	40	3	3
sqn6	6	12	60	60	2	2
vg2	25	8	110	110	4	3
wim	4	7	9	9	2	2
x1dn	27	6	110	110	5	4
x6dn	39	5	81	81	5	4
x9dn	27	7	120	120	5	4
z4	7	4	59	59	3	2
5xp1	7	10	81	80	3	2
9sym	9	1	96	97	5	4
40			2788	2803	146	101

area of the PLA is 5.2% (= [(2 * 6) * 82]/[(2 * 94 + 43) * 82]). The CPU time of REST is very little as compared to that of the testable design technique proposed by [12] and [13]. Since the CPU time for the testable design technique is heavily dependent on the number of extra inputs needed, so REST also reduces the CPU time of the testable design technique.

Based on the most of algorithms in ESPRESSO-II [16], we implement a logic minimization program named GHH. Then, followed by the strategies described in Section V, LMTPLA is implemented by combining GHH and restructuring algorithm. For the same 40 benchmark circuits, we have applied GHH and LMTPLA to these examples, and the results are summarized in Table II. Columns 1–3 are the same as that in Table I. Columns 4 shows the number of product terms obtained with GHH. The total number of cubes is 2788. Column 6 denotes the number of extra inputs required in testable design. The total number of extra inputs is 146. However, if we run LMTPLA instead of GHH, the total number of cubes is increased from 2788 to 2803, as shown in column 5. The increasing rate is approximately 0.5%. In this case, the total number of extra inputs is reduced dramatically from 146 to 101, as shown in column 7. The reduction ratio is approximately 30%. However, the total CPU time is slightly increased from 1063.6 s (GHH) to 1230.2 s (LMTPLA).

Tables I and II show that ESPRESSO-MV is better than GHH in logic minimization, i.e. 2731 cubes versus 2788 cubes. But GHH needs fewer extra inputs than ESPRESSO–MV, i.e. 146 inputs versus 168 inputs. Because it is based on GHH, the minimization effect of LMTPLA is not so good as compared with ESPRESSO-MV. If we modify ESPRESSO-MV to LMTPLA, the results should be better than the old LMTPLA in the total number of cubes. However, REST, can be used after logic minimization by any logic minimizer.

VII. CONCLUSIONS

Synthesis-for-testability is a popular research area recently. We categorize it into four different viewpoints. The first one is synthesis for full testability, i.e., removing all redundant faults in synthesis process. The second one is synthesis for easy generating tests, i.e., reducing the cost of test pattern generation. The third one is synthesis for shortening test length, i.e., reducing the cost of test application. The last one is synthesis for testable design, i.e., reducing the extra cost induced by testable design technique. In this paper, synthesis-for-testability is emphasized on the last viewpoint.

The fully testable PLA design technique proposed in [12], [13] can guarantee that all multiple stuck-at faults, as well as all multiple extra and multiple missing device faults, are detected. Furthermore, the test set is a byproduct when the testable design is finished. Hence, test pattern generation is unnecessary and redundant faults do not exist. To reflect the ease or the difficulty of applying testable design technique, an accurate method for measuring testability is important. In Section III, we first derive a simple testability measure called UCP. Based on UCP, we have developed two PLA synthesis algorithms. The objective of these algorithms is to optimize UCP such that the added extra logic is minimized. From experimental results, it is found that UCP is very accurate and effective. The first one is a restructuring algorithm named REST, and the other is a new logic minimizer named LMTPLA. For a minimized PLA, REST can enhance its testability and preserve the same logic function and almost the same

chip area for the PLA simultaneously. Thus we can make the restructured PLA testable by taking less extra hardware. In REST, with UCP as a cost function, two new procedures are proposed: 1) Cube_Reduction and 2) Cube_Partition. From experimental results, it is found that REST is very efficient not only on its performance but also the CPU time. LMTPLA is principally based on ESPRESSO-II and REST. In order to get a minimum UCP, four strategies are developed: 1) deleting the cubes with poor testability and reserving the cubes with good testability, 2) giving up the primes, 3) if necessary, partitioning the more untestable cubes into smaller cubes, and 4) deleting the procedures which are useless in LMTPLA. For 40 benchmark circuits, the hardware overheads required are reduced by about 30–40%. The CPU time of REST and LMTPLA is very little as compared to that of the testable design technique.

In fact, the concept of restructuring-for-testability and logic-minimization-for-testability can also be applied to many other logic circuits. When we get a high testability circuit, we can reduce the cost of testable designs. REST has been used in LOPET (Low Overhead Pseudoexhaustive Testable PLA) [19] and gets a good result.

Acknowledgment

The authors wish to thank the anonymous reviewers for their helpful and constructive comments.

References

[1] H. Fleisher and L. I. Maissel, "An introduction to array logic," *IBM J. Res. Develop.*, vol. 19, pp. 98–109, Mar. 1975.

[2] C. W. Cha, "A testing strategy for PLA's," in *Proc. 15th Design Automation Conf.*, Las Vegas, NV, June 1978, pp. 83–89.

[3] S. J. Hong and D. L. Ostapko, "Fault analysis and test generation for programmable logic array (PLA's)," *IEEE Trans. Comput.*, vol. C-28, pp. 617–626, Sept. 1979.

[4] J. E. Smith, "Detection of faults in programmable logic arrays," *IEEE Trans. Comput.*, vol. C-28, pp. 845–853, Nov. 1979.

[5] P. Bose and J. A. Abraham, "Test generation for programmable logic arrays," in *Proc. 19th Design Automation Conf.*, Las Vegas, NV, June 14-16, 1982, pp. 574–580.

[6] R. S. Wei and A. Sangiovanni-Vincentelli, "PLATYPUS: A PLA test pattern generation tool," *IEEE Trans. Computer-Aided Design*, vol. CAD-5, pp. 633–644, Oct. 1986.

[7] M. Robinson and J. Rajski, "An algorithmic branch and bound method for PLA test pattern generation," in *Proc. IEEE 1985 Int. Test Conf.*, 1988, pp. 784–795.

[8] H. Fujiwara, and K. Kinoshita, "A design for programmable logic arrays with universal tests," *IEEE Trans. Comput.*, vol. C-30, pp. 823–828, Nov. 1981.

[9] J. Khakbaz, "A testable PLA design with low overhead and high fault coverage," in *Dig. 13th Ann. Int. Symp. Fault-Tolerant Comput. (FTCS-13)*, Turino, Italy, June 1983, pp. 426–429.

[10] K. K. Saluja, K. Kinoshita, and H. Fujiwara, "An easily testable design of programmable logic arrays for multiple faults," *IEEE Trans. Comput.*, vol. C-32, pp. 1038-1046, Nov. 1983.

[11] H. Fujiwara, "A new PLA design for universal testability," *IEEE Trans. Comput.*, vol. C-33, Aug. 1984.

[12] S. Bozorgui-Nesbat, and E. J. McCluskey, "Lower overhead design for testability of programmable logic arrays," *IEEE Trans. Comput.*, vol. C-35, pp. 379–383, Apr. 1986.

[13] J. Khakbaz, and S. Bozorgui-Nesbat, "Minimizing extra hardware for fully testable PLA design," in *IEEE Dig. Int. Conf. on Computer-Aided Design (ICCAD-85)*, Santa Clara, CA, Nov. 1985, pp. 102–104.

[14] K. K. Saluja, H. Fujiwara, and K. Kinoshita, "A testable design of programmable logic arrays with universal control and minimal overhead," in *Proc. IEEE 1985 Int. Test Conf.*, Philadelphia, Pa., Nov. 19–21, 1985, pp. 574–582.

[15] S. Bozorgui-Nesbat, "PLA partitioning for testability," in *Proc. IEEE 1987 Int. Test Conf.*, 1987, pp. 172–180.

[16] R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*. Hingham, MA: Kluwer Academic, 1984.

[17] R. L. Rudell, and A. Sangiovanni-Vincentelli, "Multiple-valued minimization for PLA optimization," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, pp. 727–751, Sept. 1987.

[18] G. H. Hwang and W. Z. Shen, "Restructure algorithm for testable PLA," in *Proc. 1990 Int. Symp. on Circuits and Systems* (ISCAS'90), May 1990, pp. 2740–2743.

[19] W. Z. Shen, G. H. Hwang, W. J. Hsu and Y. J. Jan, "Design of pseudoexhaustive testable PLA with low overhead," in *Proc. Int. Symp. on VLSI Technology, Systems, and Applications*, Taiwan, R.O.C., May 22–25, 1991, pp. 409–413.

Gwo-Haur Hwang (S'90) received the B.S. degree in 1987, and the M.S. in 1989, both in electronics engineering from the National Chiao Tung University, Taiwan, Republic of China. He is currently working toward the Ph.D. degree at National Chiao Tung University.

His research interests include VLSI Testing and synthesis for testability.

Wen-Zen Shen (S'78–M'90) received the Ph.D. degree in electronics from the National Chiao Tung University, Taiwan, Republic of China, in 1982.

He is currently a Professor in the Department of Electronics Engineering at National Chiao Tung University. His research interests include VLSI Design and Test, computer-aided-design, and VLSI for signal processing.

Dr. Shen is a member of Phi Tau Phi.