A TESTABLE DESIGN OF PROGRAMMABLE LOGIC ARRAYS WITH UNIVERSAL CONTROL AND MINIMAL OVERHEAD

K. K. SALUJA, H. FUJIWARA and K. KINOSHITA

1Department of Electrical and Computer Engineering, University of Wisconsin-Madison, Madison, WI 53706, U.S.A.
2Department of Electronics and Communications, Meiji University, Kawasaki, 214 Japan
3Department of Information and Behavioral Sciences, Hiroshima University, Hiroshima, 730 Japan

Abstract—In this paper we propose a new technique for designing easily testable PLAs. Our design is a hybrid of the many existing testable designs of PLAs and therefore has almost all features of existing designs. These are (1) simple design, (2) high fault coverage, (3) easily implemented on existing design automation systems, (4) little or no degradation of PLA performance in normal operations, and (5) elimination of need for test pattern generation and fault simulation. In addition to these, we define the silicon area overhead, \( g \), associated with a PLA as an objective function. We then find a solution such that \( g \) is minimized in our design. Thus the additional feature our PLA possesses is (6) "minimal" overhead. The technique consists of addition of some bit lines as well as a shift register for control of product-lines during test mode. The extra logic is added in such a manner that all the easily testable features are maintained whereas the overall area of the extra logic is minimized. Using this design all multiple stuck-at faults, as well as all multiple extra and missing cross-point faults are detected.

1. INTRODUCTION

With the growth in the complexity of VLSI circuits, the only way to carry out a circuit design to completion is by not only enlisting the help of design automation (DA) tools but by making use of regular structures in the design process. Although, with the use of DA tools almost any regular structure can be designed without much difficulty, yet some regular structures have become integral parts of DA systems and of integrated circuits (IC). Most commonly used regular structures being RAMs, ROMs and PLAs (programmable logic arrays). Use of PLAs in ICs is gaining increasing popularity for the following reasons:

(1) It is a powerful structure to realized arbitrary combinational as well as sequential circuits. Therefore, its use reduces the overall complexity of the chip design.
(2) PLAs can often be implemented as testable structures [1-11] thus making the otherwise complex problem of testing VLSI circuits of manageable proportions.
(3) It is easy to include engineering design changes in ICs designed using PLAs.
(4) Hardware and silicon area required to implement a PLA can often be further reduced by using PLA minimization methods [12-13].

In most testable designs of PLAs, enhancement in testability is achieved through use of additional logic to control individual product lines in test mode, though some designs [17,18] are based on the ability to individually control bit lines. Typically, such a control over product lines is achieved by one of the following two methods:

(1) Using a shift register (SR) or shift register with multiplexer [1,3-7].
(2) Using extra bit lines to form a decoder or decoder like structure [2,8,9].

As both the methods provide almost equal fault coverage, the superiority of a method depends on the amount of silicon area overhead. Bozorgui-Nesbat and McCluskey[9] argue that it is difficult to construct register cells with the same pitch as PLA pitch, thus a design employing SR is likely to be inferior than a design which uses extra bit lines. On the other hand Hua et al.[7] and others [14] argue the use of SR with multiplexers and SR with extra inputs or SR wrapped around a PLA to obtain a low silicon area overhead.

In general, depending on the function realized by a PLA, number of inputs, number of product lines and number of outputs; any of the two methods can result into a testable design with lower silicon area overhead.
In this paper we marry the two approaches and show that the resulting design not only has the required fault coverage but results into an area overhead lower than the either approach.

This paper is organised as follows. In Section 2 of this paper we give preliminaries and describe the required notation and the details of the methods which are to be integrated in this paper. In Section 3 we propose the design of a universal control and discuss its properties. In Section 4 we described as to how partitioning can be employed to merge the universal control and the use of SR concepts. In Section 5 we find the optimal length of SR and size of control, analytically, such that overhead is minimized. Asymptotic bounds on overhead and further reduction of overhead for non-universal control are discussed in Sections 6 and 7, respectively.

2. PRELIMINARIES

In this section we present the notation and known testable designs. We also include some of the known results for the sake of completeness of this paper.

From logical description point of view, PLAs are two level sum-of-product realizations of combinational logic functions. Although in a given technology their implementation may not be AND/OR. For example nMOS PLAs are NOR–NOR implementations. However, for our analysis and presentation of results we choose AND/OR realization. Conversion of the tests derived in this paper to suit NOR–NOR or other forms is a simple matter. Similarly, fault coverage results also apply to other forms of PLAs.

A general PLA structure is shown in Fig. 1. Input decoders, $D_i$, shown in this figure will be assumed to be one-bit decoders providing true and complement bit lines to the AND plane. For simplicity of presentation other details of a PLA structure (e.g. pull-ups, grounds etc.) will now be shown. Program points of a PLA are intersection of bit lines and product lines in the AND-plane and product lines and output lines in the OR-plane. A PLA can be completely described by its personality matrices for AND and OR planes. We shall denote the size of a PLA by an ordered triple $(n, m, l)$ having $n =$ number of inputs, $m =$ number of product lines and $l =$ number of

![Fig. 1. A general PLA structure.](image-url)
outputs. The personality matrix $A$ for AND plane is an $n \times m$ matrix whose entries are defined as follows:

$$a_{i,j} = \begin{cases} 
1 & \text{if } i\text{th input and } j\text{th product line have a cross-point present} \\
0 & \text{if complement of } i\text{th input and } j\text{th product line have a cross-point present} \\
- & \text{otherwise.}
\end{cases}$$

Similarly, the personality matrix $Q$ for the OR plane is a $l \times m$ matrix whose entries are as follows:

$$q_{i,j} = \begin{cases} 
1 & \text{if } i\text{th output and } j\text{th product line have a cross-point present} \\
- & \text{otherwise.}
\end{cases}$$

Typically, all testable designs incorporate extra logic to enhance testability. A general structure of a testable PLA is shown in Fig. 2. While discussing different testable design we shall often refer to this figure.

2.1. Fault model

We shall assume that only the following faults can be present in a PLA: (1) any number of $s$-$a$ faults; (2) any number of extra devices; and (3) any number of missing devices.
Though we do not assume the presence of adjacent line bridging faults, yet they can be detected with only a minor modifications in some of the cases discussed in this paper. Also, as Saluja et al. [11] have shown that s-a faults in a PLA are equivalent to missing device or output s-a-0 faults, therefore, while presenting the proofs we shall only consider the fault set modified accordingly.

2.2. Bozorgui-Nesbat and McCluskey's (BM) approach [9]
In this approach \( L_p \) (Fig. 2) consists of a decoder-like structure. Extra inputs are added such that the resulting \( A \) matrix has certain distance properties. \( L_i \) essentially consists of a circuit which can be enabled only during test mode (in nMOS it will consist of pull down transistors on extra lines). We shall call the PLA obtained with this approach a BM-PLA. In our notation, \( P_i \) denotes \( i \)th product line as well as function realized by \( i \)th product line. An \( n \)-bit input vector will be denoted as \( X = (c_1, c_2, \ldots, c_n) \).

**DEFINITION 1**
A set of inputs, \( S_i \), is called a select set of \( P_i \) if \( S_i = \{X/P_i(X) = 1\} \).

**DEFINITION 2**
An input vector \( t_{i,0} \) is called main test pattern of \( P_i \) if (1) \( t_{i,0} \in S_i \) and (2) \( d_H(t_{i,0}, X) \geq 2 \) for all \( X \) and \( S_j \) such that \( X \in S_j \) and \( j \neq i \), where \( d_H(a, b) \) is Hamming distance between vectors \( a \) and \( b \).

**DEFINITION 3**
For a main test pattern \( t_{i,0} = (c_1, c_2, \ldots, c_n) \) for \( P_i \), we define \( n \) auxiliary test patterns of \( P_i \) as \( t_{ij} = (c_2, c_3, \ldots, ?j, c_j+1, \ldots, c_n); j = 1, 2, \ldots, n \).

**DEFINITION 4**
Test patterns for \( P_i \), \( TP_i \), is the set defined as: \( TP_i = \{t_{i,0}, t_{i,1}, \ldots, t_{i,n}\} \).

**DEFINITION 5**
Test set for BM-PLA, \( T_{BM} \) is defined as \( T_{BM} = TP_1 \cup TP_2 \cup \cdots \cup TP_m \).

**THEOREM 1** [9]
A BM-PLA can be tested for all faults by the test set \( T_{BM} \).
In [9] is given a procedure for adding extra inputs such that the resulting PLA is easily testable. Number of extra inputs depend on the \( A \) matrix of the original PLA. Computational complexity of the procedure which changes a PLA to BM-PLA is \( O(m^3) \).

2.3. Khakbaz’s (K) approach [5]
In this approach \( L_p \) consists of a SR to control the individual product lines. No extra logic is required at \( L_1 \). An extra observable output is added. We shall call the PLA obtained by this approach a K-PLA. In deriving the tests for K-PLA, a test vector will be denoted by a \( (m + n) \)-bit vector, \( (r_1, r_2, \ldots, r_m; c_1, c_2, \ldots, c_n) \). The first \( m \) bits denote the contents of SR, and the remaining \( n \) bits indicate the inputs. We shall often group them together and denote the test vector as \((R, X)\). The following \( R \) and \( X \) vectors are of special interest for deriving tests for K-PLA.

**DEFINITION 6**
Vectors \( R_0 \) and \( R_i \) are defined as follows:
\[
R_0 = (0, 0, \ldots, 0)
\]
\[
R_i = \left( 0, 0, \ldots, 0, 1, 0, \ldots, 0 \right).
\]

**DEFINITION 7**
An input \( X_i \) is called a test input for \( P_i \) if \( X_i \in S_i \) (see definition 1 for \( S_i \)).

**DEFINITION 8**
If \( X_{ij} \) (test input for \( P_i \)) is denoted as \( (c_1, c_2, \ldots, c_n) \), we define \( n \) auxiliary test inputs for \( P_i \) as \( X_{ij} = (c_1, c_2, \ldots, \bar{c}_j, \ldots, c_n); j = 1, 2, \ldots, n \).
A testable design of programmable logic arrays

Table 1. Comparison of different testable designs

<table>
<thead>
<tr>
<th>Extra logic L_p</th>
<th>BM-PLA</th>
<th>K-PLA</th>
<th>SKF-PLA and F-PLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Type</td>
<td>Decoder like</td>
<td>SR</td>
<td>SR</td>
</tr>
<tr>
<td>(2) Dependence on PLA</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>(3) Extra inputs</td>
<td>(PLA)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(4) Computation complexity</td>
<td>0(m³)</td>
<td>Nil</td>
<td>Nil</td>
</tr>
<tr>
<td>(5) Cost</td>
<td>(PLA)</td>
<td>f(m)</td>
<td>f(m)</td>
</tr>
<tr>
<td>Extra logic L_I</td>
<td>(1) Type</td>
<td>Controllable extra inputs</td>
<td>Nil</td>
</tr>
<tr>
<td>(2) Delay inserted</td>
<td>Nil</td>
<td>Nil</td>
<td>Yes</td>
</tr>
<tr>
<td>(3) Cost</td>
<td>Constant</td>
<td>Nil</td>
<td>f(n)</td>
</tr>
<tr>
<td>Test generation</td>
<td>(1) Required</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>(2) Length</td>
<td>(n, m)</td>
<td>f(n, m)</td>
<td>f(n, m)</td>
</tr>
<tr>
<td>(3) Fault Coverage</td>
<td>s-a and cross-points</td>
<td>s-a and cross-points</td>
<td>s-a, cross-points and bridging</td>
</tr>
<tr>
<td>(4) Stored test patterns</td>
<td>m</td>
<td>m</td>
<td>Constant (2)</td>
</tr>
</tbody>
</table>

f(x, y, ...) means a function of x, y, etc; f(PLA) means a function of personality of PLA.

DEFINITION 9

Test vectors for P_i, TV_i, is the set TV_i = {(R_i; X_1), (R_i; X_2), (R_i; X_{1+}), ..., (R_i; X_{n})}.

DEFINITION 10

Test set for K-PLA, T_K, is defined as

T_K = TV_0 ∪ TV_1 ∪ TV_2 ∪ ... ∪ TV_m,

where

TV_0 = {(R_0; X_1), (R_0; X_2), ..., (R_0; X_m)}.

THEOREM 2 [5]

A K-PLA can be tested for all faults by the test set T_K.

2.4. Saluja et al. [4] (SKF) and Fujiwara [6] (F) approaches

In these approaches L_o consists of a SR to control the individual product lines. L_i consists of additional controllable logic in the form of SR [4] or gates [6] to provide inputs to the AND plane of the PLA such that the resulting PLA can be tested by test sets independent of the personality of a PLA. We do not describe the necessary test sets here, as it will make this paper unduly long. However, the following theorem is a consequence of these works. In a later section in this paper we will apply the key idea used in SKF-PLA and F-PLA to arrive at a new design.

THEOREM 3[4,6]

SKF-PLA and F-PLA can be tested for all faults by test sets independent of the PLA. Furthermore, SKF-PLA and F-PLA can also be tested for adjacent line bridging faults.

2.5. Comparison

Comparative advantages and disadvantages of the different designs discussed above are listed in Table 1.

3. UNIVERSAL CONTROL

In this section we propose a design of L_p which makes use of extra inputs and bit lines similar to BM-PLAs. However, the design proposed will be universal in nature, i.e. independent of the personality of PLA. The algorithm described in [9] for adding extra inputs to the PLA has a
computational complexity of $O(m^3)$. For large PLAs, use of such an algorithm can be prohibitively expensive, thus making the use of this method applicable only to small PLAs.

We shall first study some properties of a special AND-array, called a decoder–parity–AND–array (DPAA), defined below.

**DEFINITION 11**

A DPAA is an AND-Array with $m$ product lines, $\lceil \log_2 m \rceil + 1$ inputs and with the $A$ matrix $A_D$ as follows: if $m$ columns of $A_D$ matrix are numbered from $A_0$ to $A_{m-1}$, and

\[
\begin{bmatrix}
    a_{0,i} \\
    a_{1,i} \\
    \vdots \\
    a_{n,i} \\
    a_{n+1,i}
\end{bmatrix}
\]

then $(a_{0,i}, \ldots, a_{n,i})$ is the binary representation of $i$ and $a_{n+1,i} = a_{n,i} \oplus \cdots \oplus a_{2,i} \oplus a_{1,i}$.

The following lemmas describe some properties of a DPAA.

**LEMMA 1**

$d_H(A_i, A_j) \geq 2$; $0 \leq i, j < m; i \neq j$.

**LEMMA 2**

For the DPAA, the select set $S_i$ of $P_i$ is uniquely determined and consists of a single pattern. Notationally, $|S_i| = 1$ and $S_i = A_i^T$.

The following definition is a minor variation of the definition in [9] and is stated here for the sake of completeness of this paper:

**DEFINITION 12**

The distance matrix, $D$, of a PLA is a $m \times m$ matrix whose entries are defined as follows:

\[
D_{ij} = \min\{d_H(X, Y) | X \in S_i, Y \in S_j\}; \quad i \neq j
\]

\[
= 2 \quad \text{for} \quad i = j.
\]

**LEMMA 3**

Every element of the distance matrix for the DPAA is greater than or equal to 2.

Proof. It follows from Lemmas 1 and 2. □

**DESIGN 1**

Augment a given $(n, m, l)$ PLA by concatenating a DPAA to it as shown in Fig. 3b. We shall call this PLA as D1-PLA. In Fig. 3 we have not shown the extra logic which can be used to disable the DPAA part of the PLA during normal operation of the PLA. In nMOS PLAs [16] such a circuit will be of the form shown in Fig. 3c. Note that $k = \lceil \log_2 m \rceil$ and total number of extra inputs to D1-PLA are $(1 + k)$.

We now derive a test set for D1-PLA.

Let $S_i$ be the select set of $P_i$ for the original PLA. Let $\{u_i\}$ by the select set of $P_i$ for the decoder–parity PLA. Then, the select set of $P_i$ for the augmented PLA is

$\{u_i, X | X \in S_i\}$.

**LEMMA 4**

For all $X \in S_i$ and $Y \in S_j$, $d_H([u_i, X], [u_i, Y]) \geq 2$.

This lemma guarantees that any element of the select set of $P_i$ for the augmented PLA can be a main test pattern of $P_i$.  

\[\uparrow|x| \text{ means cardinality of a set } x; \ Y^T \text{ means transpose of a vector } Y.\]
Let $t_{i,0}$ be any one of the elements of $u_i \cdot S_i$. For a $t_{i,0}$, we can obtain the auxiliary test patterns of $P_i$. Let these be $t_{i,1}, t_{i,2}, \ldots, t_{i,n + \lceil \log_2 m \rceil + 1}$.

Then, the test patterns for the D1-PLA are:

$$T_{D1} = \{t_{i,0}, t_{i,1}, \ldots, t_{i,n + \lceil \log_2 m \rceil + 1},$$

$$t_{j,0}, t_{j,1}, \ldots, t_{j,n + \lceil \log_2 m \rceil + 1},$$

$$t_{m,0}, t_{m,1}, \ldots, t_{m,n + \lceil \log_2 m \rceil + 1}\}.$$

**Theorem 4**

The D1-PLA of Fig. 3b can be tested for all faults by the test set $T_{D1}$ mentioned above.

*Proof.* It follows from Theorem 1 and Lemma 4.

**Theorem 5**

The length of the test set to test D1-PLA is $m(2 + n + \log_2 m)$. All these test patterns can be generated from $m$ main test patterns.

Comparison of this method with BM-PLA is given in Table 2 for 10 PLAs taken from [9]. The number of extra inputs in D1-PLA is not much larger than BM-PLA, but the computational complexity for generating D1-PLA is nil. Minimization of overhead (number of inputs as well as silicon area) with the use of DPAA is the subject of the next two sections.

4. PARTITIONING

In this section we propose a partitioning procedure which helps reduce the number of extra inputs without sacrificing the testability properties of the PLA. The basic idea being a PLA is partitioned into $k$ blocks, each block containing $m/k$ product lines. A DPAA is appended to each
<table>
<thead>
<tr>
<th>PLA name</th>
<th>Master</th>
<th>New ALU</th>
<th>Bas New</th>
<th>Recur</th>
<th>Traffic</th>
<th>ALU test</th>
<th>CERBERUS</th>
<th>COND</th>
<th>BAR</th>
<th>RIMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of extra inputs for:</td>
<td>BM-PLA</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>D1-PLA</td>
<td>6</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Computation complexity of generating:</td>
<td>BM-PLA</td>
<td>$O(27^2)$</td>
<td>$O(26^2)$</td>
<td>$O(33^3)$</td>
<td>$O(9^3)$</td>
<td>$O(36^3)$</td>
<td>$O(50^3)$</td>
<td>$O(24^3)$</td>
<td>$O(29^3)$</td>
<td>$O(39^3)$</td>
</tr>
<tr>
<td></td>
<td>D1-PLA</td>
<td>nil</td>
<td>nil</td>
<td>nil</td>
<td>nil</td>
<td>nil</td>
<td>nil</td>
<td>nil</td>
<td>nil</td>
<td>nil</td>
</tr>
</tbody>
</table>

Fig. 4. Partitioning of a PLA (D2-PLA).
block as shown in Fig. 4. Furthermore, an SR of length \( k \) is introduced to control these blocks. The formal design description and the testability properties of the resulting PLA are given below.

**DESIGN 2**

Product lines of a PLA are divided into \( k \) blocks with each block containing \( h \) product lines, i.e. \( h = \lceil m/k \rceil \). Note that all but the \( k \)th block have \( h \) product lines, whereas number of product lines in the \( k \)th block are \( m - (k - 1)h \). A DPAA is appended to each block. All DPAAAs receive the same inputs thus total number of extra inputs to the PLA are \( 1 + \lceil \log_2 h \rceil \). An SR of length \( k \) is appended. The resulting PLA is called D2-PLA (Fig. 4).

While discussing testing of D2-PLA we must not lose sight of the actual realization and physical failures. Keeping this in mind we make the following observations.

**OBSERVATION 1**

Saluja et al.\[11\] have shown that in a PLA all s-a faults can be reduced to multiple missing cross-point or outputs s-a-0 faults. However, s-a-1 faults at \( r_i, r_i0, e_i, e_{i1}, \ldots \) etc, for block \( i \) shown in Fig. 5 must be tested explicitly. Although s-a-1-faults at these locations will not interfere in the normal operation of a PLA yet their presence may invalidate other tests. Thus our fault set is (1) multiple cross-points (extra and/or missing) and output(s) s-a-0 faults and (2) s-a-1 faults at location marked in Fig. 5. Notice that this fault set includes the fault set given in Section 2.1.

**OBSERVATION 2**

In Fig. 4, the number of extra inputs is (1) 1 for SR and (2) \( 1 + \lceil \log_2 h \rceil \) for DPAAAs. However, if we take \( h = 1 \) then Design 2 should be degenerated to K-PLA requiring no DPAA. Similarly, for \( h = m \) Design 2 should be degenerated to D1-PLA requiring no SR.

Further, in the presence of SR we would need to observe the output \( r_{ki} \) to test SR.

**OBSERVATION 3**

We can introduce an extra output, \( z^* \), in the OR-plane similar to [4–6], in which case the AND plane can be completely tested by observing only \( z^* \). However, to test the OR-plane, as well as outputs s-a-0, all outputs need to be observed, therefore it is not necessary to introduce \( z^* \) outputs and complete testing can be carried through by observing the normal outputs of D2-PLA.

We now describe the testing of D2-PLA. Testing is carried in a number of conceptual steps described below, though in practice these steps can be merged to obtain a reduced test set.

**Step 1: Testing SR.** SR is tested by applying a sequence of zeros, followed by a one, followed by zeros and observing the \( r_{ki} \) output. This will also detect s-a faults at \( r_i \)'s in any block.

**Step 2: Testing s-a-1 faults at locations marked in Fig. 5.** Set \( r_i = r_{i2} = \cdots = r_k = 0 \). Now apply \( m \) main test patterns. All outputs must stay zero during this test for a fault free PLA.

![Fig. 5. Certain fault sites in block i.](image-url)
Step 3: Testing PLA. Each block of the PLA is now tested independently. To test block \( i \) set \( r_i = 1 \) and \( r_j = 0 \) for \( j \neq i \). Apply \( h(2 + n + \lceil \log_2 h \rceil) \) test patterns required to test block \( i \).

This completes the testing procedure.

The following theorem states this result formally though the proof has not been included as it is straightforward.

**Theorem 6**

The D2-PLA of Fig. 4 can be tested for all faults by tests stated in above three steps.

**Theorem 7**

The length of the test set to test D2-PLA is \( m(3 + n + \lceil \log_2 h \rceil) \).

Proof. This is total number of tests in Steps 2 and 3 above. Test for SR is included in these steps. \( \square \)

5. OPTIMAL PARTITIONING

In the previous section we discussed how using partitioning one can reduce the number of extra inputs without sacrificing the fault coverage. Of course, if reduction of number of extra inputs is the only objective than SKF-PLA, F-PLA or K-PLA are optimal (in general) by choosing \( h = 1 \). Here we set out objective as follows:

**Objective:** Find optimal partitioning such that the area of \( L_p \) in D2-PLA is minimized. Notice that in D2-PLA the only added logic is in the form of \( L_p \) (other than disabling transistors for extra inputs—these are not shown in the figure to keep the figure simple). Therefore, minimizing the areas of \( L_p \) will result in a minimal overhead PLA.

We now define the cost function. Let \( w_1 \) be the area of a PLA cell and \( w_2 \) to be the area of a SR cell.

It is difficult, if not impossible, to have the pitch of SR cells the same as PLA pitch. However, in D2-PLA we need the pitch of SR cells to be \( h \times \) PLA pitch, to avoid any interconnection overheads. In the following discussion we assume such to be the case, i.e. \( h \) will be large enough to make interconnection overhead as zero. Notice interconnection overhead can also be reduced to zero by having SR wrap around PLA [14] or by use of SR with multiplexers [7].

In a D2-PLA increase in area of PLA by adding an extra bit line is \( mw_1 \). Therefore an extra input increases the area of PLA by \( 2mw_1 \). Thus in D2-PLA the overhead

\[
g = 2mw_1 (1 + \lceil \log_2 h \rceil) + kw_2 \tag{1}
\]

where

\[
k = \lceil m/h \rceil.
\]

Our objective is to find \( h = f(m, w_1, w_2) \) such that \( g \) is minimized.

**Note:** (i) There are two special cases which are not included in equation (1) to keep the presentation simple. These are (a) for \( h = 1 \), \( g = mw_2 \); (b) for \( h = m \), \( g = 2mw_1 (1 + \lceil \log_2 m \rceil) \).

(ii) The solution for \( h \) must be found in the set of integers.

The following theorem simplifies the solution space for \( h \) considerably.

**Theorem 8**

If \( g \) is minimized for some \( h \) such that \( 2^{x-1} < h < 2^x \) then \( g \) is also minimized for \( h = 2^x \).

Proof. If \( g \) is minimized for \( 2^{x-1} < h < 2^x \) then

\[
g_{\text{min}} = 2mw_1 (1 + x) + \lceil m/h \rceil w_2
\]

\[
g_{h=2^x} = 2mw_1 (1 + x) + \lceil m/2^x \rceil w_2
\]

but

\[
\lceil m/2^x \rceil \leq \lceil m/h \rceil \quad \text{for} \quad h < 2^x
\]

therefore

\[
g_{h=2^x} \leq g_{\text{min}}
\]
but $g_{\min}$ is minimum, therefore

$$g_{s=2^x} = g_{\min}. \quad \Box$$

The implication of Theorem 8 is that while finding the minimum value of $g$ we need only consider those $h$ which are some powers of 2. Therefore a solution can easily be found either by hand computations or by using a computer. The problem can also be solved analytically by solving the equation

$$\frac{dg}{dh} = 0. \quad (2)$$

To gain some insight into the solution method we first consider the case where $2^x ( = h)$ divides $m$. Note a sufficient conditions for this case is $m$ to be a power of two.

**Theorem 9**

If $h$ divides $m$ then $g$ is minimized for $h = 2 \lfloor \log_2 (s/2 \ln 2) \rfloor$ or $2 \lfloor \log_2 (s/2 \ln 2) \rfloor$, where $s = w_2/w_1$.

*Proof.* By Theorem 8 we need only consider $h = 2^x$. Equation (1) in this case reduces to

$$g = 2m w_1 (1 + \alpha) + \frac{m}{2^x} w_2. \quad (3)$$

Setting $dg/dx = 0$ we have

$$\frac{d}{dx} (g) = 2m w_1 - \frac{m w_2}{2^x} \ln 2 = 0$$

i.e.

$$2^x = \frac{1}{2} \frac{w_2}{w_1} \ln 2$$

$$\alpha = \log_2 \left( \frac{s}{2} \ln 2 \right); \quad \text{where} \quad s = \frac{w_2}{w_1}.$$

However, $\alpha$ must be an integer. Therefore

$$\alpha = \left\lfloor \log_2 \left( \frac{1}{2} \frac{w_2}{w_1} \ln 2 \right) \right\rfloor \quad \text{or} \quad \left\lfloor \log_2 \left( \frac{1}{2} \frac{w_2}{w_1} \ln 2 \right) \right\rfloor. \quad \Box$$

The general solution is not as straightforward and leads to a recursive but approximate relation given by the following theorem:

**Theorem 10**

$g$ is minimized when

$$h = 2 \left\lfloor \log_2 \left( \frac{s}{2} \frac{m_1}{m} \ln 2 \right) \right\rfloor \quad \text{or} \quad 2 \left\lfloor \log_2 \left( \frac{s}{2} \frac{m_1}{m} \ln 2 \right) \right\rfloor$$

where

$$s = \frac{w_2}{w_1}; \quad m = qh + \gamma = m_1 + \gamma; \quad 0 \leq \gamma < h.$$

| Table 3. Optimal value of $s$ for different $m$ and $s$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $s$ | 16 | 20 | 30 | 32 | 40 | 50 | 60 | 64 | 70 | 80 | 90 | 100 | 128 |
| 10 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 20 | 4 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 30 | 4 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
Proof. Proof is similar to the proof for Theorem 9. We set
\[
\left\lfloor \frac{m}{h} \right\rfloor = \left\lfloor \frac{m_1 + \frac{r}{h}}{h} \right\rfloor = \left\lfloor \frac{m_1}{h} \right\rfloor + 1 = \frac{m_1}{h} + 1
\]
and \( h = 2^s \) (by Theorem 8) in the expression for \( g \) and set \( \frac{dg}{dh} = 0 \). \( \square \)

Note that although Theorem 10 gives a recursive solution it is very easy to solve by successive
approximation by first approximating the value of \( h \) using the result of Theorem 9 and then making
a correction in the solution. The solution never takes more than 2 steps of approximations. It is
interesting to note that the solution is almost independent of \( m \) and depends only on \( s \), i.e. the
ratio (area of SR cell)/(area of PLA cell).

Table 3 gives values of \( \alpha \) for different values of \( m \) and \( s \). It is, evident from this table that
BM-PLA are only likely to be optimal for small PLAs and for large \( s \). In other conditions
partitioning is likely to provide lower overhead. We don’t know the ratio \( s \) for the layout of PLA
used in [7], but we conjecture that \( s \) was approx. 10 and therefore, instead of one-bit multiplexer,
the use of a two-bit multiplexer would have resulted in a PLA with still lower overhead.

6. ASYMPTOTIC OVERHEAD

In this section we derive simplified expression of overhead for large PLAs. The total area of an
nMOS PLA is given by area of AND/OR planes and area of input inverters, pull-ups, etc. For
derivation of simple expression we shall consider only the area of AND/OR planes, i.e. PLA
area = \( m(2n + l)w_1 \).

In D2-PLA, the area of extra logic
\[
g_{\text{min}} = 2m(1 + \alpha)w_1 + \left\lfloor \frac{m}{2^s} \right\rfloor w_2.
\]
If we simplify \( \left\lfloor \frac{m}{2^s} \right\rfloor \) as \( m/2^s \) then the extra area
\[
g = m \left( 2(1 + \alpha) + \frac{s}{2^s} \right) w_1.
\]
Therefore
\[
\%	ext{ \, overhead} = \frac{100m \left( 2(1 + \alpha) + \frac{s}{2^s} \right) w_1}{m(2n + l)w_1} = \frac{100 \left( 2(1 + \alpha) + \frac{s}{2^s} \right)}{2n + l} \tag{4}
\]

From Table 3 we observe that \( \alpha \) is 2, 3 and 3 for values of \( s \) of 10, 20 and 30 respectively for
large \( m \). On substitution of these values of \( \alpha \) and \( s \) in (4) we obtain
\[
s = 10; \quad \%	ext{ \, overhead} = \frac{850}{2n + l}
\]
\[
s = 20; \quad \%	ext{ \, overhead} = \frac{1050}{2n + l}
\]
\[
s = 30; \quad \%	ext{ \, overhead} = \frac{1175}{2n + l}
\]

These values are plotted in Fig. 6. Note that the above expressions are on the conservative side
as the actual area of original PLA will be more than the area of AND/OR planes only. Thus
percentage overhead in practice is likely to be less than the above bounds.

7. FURTHER REDUCTION IN OVERHEAD

In the previous sections we have described a way of reducing the overhead through the use of
a DPAA. However, the basic reason a DPAA was appended to each block of Fig. 4 was that each
A testable design of programmable logic arrays

Fig. 6. Percentage overhead for D2-PLAs.

element of the distance matrix for each block should be no less than 2. We now propose an alternate design and discuss some of its properties.

**DESIGN 3**

The product lines of a PLA are divided into \( k \) blocks. For a block \( i \) number of extra inputs, \( e_i \), and cross points are determined by using the algorithm given by Bozorgui-Nesbat and McCluskey[9]. Now each block of the PLA has desired distance property, the extra inputs are connected together along with an SR of length \( k \), as shown in Fig. 7. We call this design a D3-PLA. The total number of extra inputs to D3-PLA are \( e = \max\{e_1, e_2, \ldots, e_k\} \).

A D3-PLA can be tested in the same manner as a D2-PLA. We therefore can state the following theorem based on Theorems 6 and 7:

**THEOREM 11**

The D3-PLA of Fig. 7 can be tested for all faults by a test set of length \( m(2 + n + e) \).

To show as to how D3-PLA can result into an overhead lower than D2-PLA as well as BM-PLA we consider a simple example.

Let us consider CERBERUS PLA [9]. It is a \( 18 \times 50 \times 37 \) PLA. It is evident that D3-PLA can be no worse than D2-PLA because if we find that the number of extra inputs for any block are larger than \( 1 + \lceil \log_2 h \rceil \), we can realize such a block by appending a DPAA.

To show that we can actually improve on BM-PLA we note the following fact:

**FACT 1**

\( e \leq \text{number of extra inputs for BM-PLA} \).

Now let us say we divide the 50 lines of CERBERUS PLA into 4 blocks for realization of D3-PLA. It can be expected that the number of extra inputs will reduce by at least 1, i.e. from 4 for BM-PLA to 3 in D3-PLA, whereas an SR of length 4 will be added.
Thus the total change in the area from the BM-PLA is
\[ K = -2 \times 50 \times w_1 + 4w_2 \]
\[ = -100w_1 + 4w_2 \]

This change is negative as long as \( s < 25 \). In general for large PLAs we can expect a larger reduction in the number of extra inputs while changing a BM-PLA into D3-PLA.

Similar arguments hold for other larger examples in [9]. An additional benefit in adopting D3-PLA over BM-PLA is reduced computational complexity which is stated in the form of the following lemma.

**Lemma 5**

Computational complexity to generate D3-PL is \( O(m^3/k^2) \) as opposed to \( O(m^3) \) for BM-PLA.

**8. CONCLUSION**

In this paper we have proposed three testable designs of PLAs. These designs can be seen as methodologies to improve over the existing designs. We have set different goals at different stages of the design. While moving from D1-PLA to D2-PLA we retained all the properties of D1-PLAs yet reduced and overhead. While introducing D3-PLA, again we were able to reduce the overhead, complexity of generation of PLA, still maintaining the fault coverage. It is simple to incorporate many other variations of these designs. For example, it is possible to design PLAs which are testable by a universal test set and use the partitioning and distance concept to reduce overhead. Such a PLA is discussed in [15].

The methods proposed in this paper are straightforward to incorporate in design automation systems. In this paper we have also described how a number of designs can be merged to give rise to a design superior than the all known designs in the case of PLAs. These concepts can be included in expert systems and may possibly result into still improved designs and design methodologies.

**REFERENCES**