A METAL and VIA Maskset Programmable VLSI Design Methodology using PLAs

Nikhil Jayakumar Sunil P Khatri
Department of EE, Texas A&M University, College Station TX 77843.

Abstract
In recent times there has been a substantial increase in the cost and complexity of fabricating a VLSI chip. The lithography masks themselves can cost between $1M and $3M. It is conjectured that due to these increasing costs, the number of ASIC starts in the last few years has declined. In this paper, we address this problem by using an array of dynamic PLAs which require only METAL and VIA mask customization in order to implement a new design. This would allow several similar-sized designs to share the same base set of masks (right up to the metal layers) and only have different METAL and VIA masks. We have implemented our methodology for both combinational and sequential designs, and demonstrate that our approach strikes a reasonable compromise between ASIC and field programmable design methodologies in terms of placed-and-routed area and delay. Our method has a $2.89 \times (3.58 \times)$ delay overhead and a $4.96 \times (4.44 \times)$ area overhead compared to standard cells for combinational (sequential) designs.

1 Introduction
With the relentless reduction in the minimum feature sizes of modern Deep Sub-micron (DSM) VLSI fabrication processes, the complexity of fabrication is increasing at an alarmingly rate. Simultaneously, the number and cost of a full set of masks has been increasing rapidly. It is not uncommon for a full set of lithography masks to cost over $1-3M [1, 2]. This change has contributed to a roughly 25% reduction [2] in Application Specific Integrated Circuit (ASIC) design starts in the last 7 years. It is believed that cell based ASICs are becoming prohibitively expensive except for very high volume applications [2].

In this paper, we introduce a new VLSI design approach to address this problem and minimize the non-recurring expense (NRE) involved with IC design. Our approach utilizes an array of precharged Programmable Logic Arrays (PLAs) with flip-flops co-located at their outputs, as its underlying circuit structure. We envision that a manufacturer would stock such arrays (of varying sizes), pre-processed up until the metalization step. To create an ASIC for a given design, the manufacturer would technology map this design to the smallest available array. After technology mapping and routing of the design, the METAL and VIA masks (the only masks that require changes) would be generated and used to customize or personalize the array to implement the design. At this point, the manufacturer could process the remaining masks, to obtain the final design. Alternately, the manufacturer could perform all steps of processing, using old masks for all other layers and the new METAL and VIA masks for customization of the design. The latter option might be used by manufacturers who do not have experience in warehousing partially completed wafers.1

Since all other masks except METAL and VIA masks remain unaltered, the manufacturer can realize the design in a low cost manner, by amortizing the bulk of the NRE over a large number of designs. Further, the manufacturer could spend a considerable effort in optimizing these designs for maximum yield, and this effort would be amortized over a large number of designs that share the common masks. Additionally, such an approach could result in a reduced processing time for a new design. Processing for a modern IC can take anywhere from 3 weeks up to a few months [4]. This methodology can therefore help reduce design turnaround time by stockpiling wafers which have been processed up to the metalization step. Also, this methodology simplifies the task of engineering change. When a bug is discovered and the design needs to be modified, our methodology would reduce the cost and time to modify the design (since it requires only METAL and VIA mask changes).

After METAL and VIA mask customization, the design would be transformed into a network of precharged PLAs [5]. Such an implementation methodology was demonstrated to be fast and area-efficient compared to a standard cell approach. As shown in [5], for a network of PLAs there is a more direct relationship between the cost function being optimized for during logic synthesis (literal count), and the actual PLA implementation. In a standard cell based flow, there is an intervening technology mapping step, which often negates the benefits of technology-independent logic optimization. A network of PLAs on the other hand, allows us to carry forward the benefits of technology-independent multi-level logic synthesis.

We leverage this feature of the network of PLA design style in our work. In contrast to the work of [5], we are able to handle both combinational and sequential designs. Also, the mask programmability feature of our approach makes our PLAs design and layout quite different from those in [5].

In recent times, PLAs have experienced a renewed interest as a circuit implementation style for high-performance designs. The IBM Gigahertz processor [6] utilized PLAs2 to implement control logic, due to their high speed and because they provide the ability to quickly implement and modify the design.

The remainder of this paper is organized as follows. The next Section 2 talks about methods similar to our own. Section 3 describes our design flow, while Section 4 describes our experimental results. Finally, in Section 5, we make concluding comments and discuss further work that needs to be done in this area.

2 Previous Work
In the past, gate arrays [7, 8] have been used as an implementation method in which a design can be personalized via and metal customization. This approach was popular until standard cell based design became the dominant means to design ICs. The speed of our approach is based on the fact that wiring is embedded inside our PLAs. This is not true for gate arrays. Also, the P and N diffusions in any row of the gate array need to be separated, resulting in larger area

---

1 Also, as the industry starts to move toward the highly absorptive and fragile low-k dielectric materials in the metal stack, the shelf life and the nature of contamination risks are not well known [3]

2 Note that single PLAs used as opposed to a network of PLAs
The transistor level design of a PLA is shown in Figure 1. We can explain the operation of the PLA based on this figure:

- In this figure, there is a transistor at the intersection of each bitline (or input line) and wordline (in the AND plane) as well as at the intersection of each wordline and output line (in the OR plane). These devices can be connected to the wordline and output line respectively as shown on the right side of Figure 1 (Subfigures b)).
- Subfigures b) on the right side of Figure 1 indicate how a device may be disconnected from the wordline or output line.
- Precharge devices pull up each wordline during the low (precharge) phase of the clock.

3 Our Approach

In this section, we describe our design methodology, starting with the PLAs we utilize (Section 3.1). In Section 3.2, we report results on characterization of these PLAs to determine the specific size (number of inputs, outputs and cubes) of PLAs we utilize in our approach. Section 3.3 describes our synthesis approach.

3.1 PLA design

In order to achieve complete programmability of the IC using only METAL and VIA masks, we use Programmable Logic Arrays (PLAs) as the implementation structure for our designs.

Consider a PLA consisting of \( n \) input variables \( x_1, x_2, \ldots, x_n \), and \( m \) output variables \( y_1, y_2, \ldots, y_m \). Let \( k \) be the number of rows in the PLA. A literal \( l_i \) is defined as an input variable or its complement.

Suppose we want to implement a function \( f \) represented as a sum of cubes \( f = c_1 + c_2 + \cdots + c_k \), where each cube \( c_i = l_{i1} \cdot l_{i2} \cdot \cdots \cdot l_{ir} \). We consider PLAs which are of the NOR-NOR form. This means that we actually implement \( f \) as

\[
\bar{f} = \sum_{i=1}^{k} \overline{c_i} = \sum_{i=1}^{k} (\overline{l_{i1}} + \overline{l_{i2}} + \cdots + \overline{l_{ir}})
\]

The PLA output \( \bar{f} \) is a logical NOR of a series of expressions, each corresponding to the NOR of the complement of the literals present in the cubes of \( f \). In the PLA, each such expression is implemented by word lines, in what is called the AND plane. Assume that these word lines run horizontally. Literals of the PLA are implemented by vertical-running bit-lines. For each input variable, there are two bit-lines, one for each of its literals. The outputs of the PLA are implemented by output lines, which also run vertically. This portion of the PLA is called the OR plane.

In the precharged NOR-NOR PLAs that we use in our methodology, the outputs are co-located with D flip-flops (one for each output).

Figure 2: Layout of the PLA core
- Static pullups are required on wordlines (in addition to precharge devices) since there is a possibility that a wordline gets capacitively pulled low if its neighbor switches low.
- Also, for each PLA, we have one wordline which we design to always switch low in the evaluate phase of the clock, and effectively act as a timing reference for the PLA. In Figure 1, the first wordline performs this function. Its signal is inverted to produce a delayed clock signal (DCLK) which is used to gate the GND signal for the OR plane, thereby ensuring that all outputs fire at the same time. This delayed clock signal is also connected to PMOS pullups at each output line which serve to precharge (pullup) the output lines during the precharge phase. We use a GND line as a shield between the reference wordline and the next wordline to ensure that there is no delay variation on this reference wordline due to crosstalk. As a result, the PLA structure has timing characteristics that are precisely estimable at the time of logic synthesis.
- Each PLA has a single output (the completion signal) which always switches in each cycle, signaling the completion of the PLA computation. This output is used to gate the clock signal of other PLAs in the network which depend on the output value of the given PLA, so that they do not begin their computation too early.
- Also, each bitline must be pulled down during precharge (low phase of CLK) to eliminate a crowbar path between VDD and GND. To do this, we included pass transistors and a pulldown device controlled by the clock signal for each input line. Note that each input is available in its complemented and uncomplemented form to the PLA core in the figure shown. In our implementation the input drivers and the inverter to generate the complemented input are included in the PLA cell. The devices for this circuit are not shown on Figure 1 for the sake of brevity.
- Finally, each output is co-located with a negative edge triggered D flip-flop (DFF), to allow for sequential circuit support. The DFFs are not shown in the Figure 1. Since the PLAs evaluate in the high phase of the clock signal, the DFFs are negative edge triggered. In the worst case, each output may need to be connected to its co-located DFF.

Figure 2 shows the layout of our PLAs. Note that we performed our layout, timing simulations and place / route experiments using a 0.1µm process. We used the bsim100 [16] model cards.

We can observe the following from Figure 2:
- Our PLA utilizes only METAL1 and METAL2 in the layout, leaving other layers free for over-the-cell routing.
- We have left in the ability to personalize the design using METAL and VIA mask changes only. Each wordline and output line device has this capability which is shown schematically in Figure 1. Also, outputs can be programmably driven to a co-located DFF, or can be left as combinational signals and driven out using output buffers.
- The layout described does not allow PLA input or output lines to be folded, since this could not be performed in a METAL and VIA mask programmable fashion. As a result, the decomposition of a design into a network of PLAs in our case requires more PLAs than [5] (where folding of PLA inputs was permitted).
- Also, unlike the case in [5], where all output buffers were implemented in the routing region (due to the presence of a wiring fabric in the die), we have to implement output drivers near the OR plane (bottom right of Figure 2), resulting in a larger area penalty than [5].
- DFFs in Figure 2 are implemented on the top right. Note that the pitch of DFFs is larger than the pitch of the outputs (shown below the DFFs). As a result, the output pitch is increased to match the DFF pitch.
- Also, compared to [5], we require an extra free vertical track for every input (for complete AND plane programmability) and an extra free horizontal track for each pair of word-lines (for full OR plane programmability). As a result, the PLA core we use is less dense than [5]. Also, as a result of this, we only consider even numbers of wordlines in our PLAs during decomposition.
- The top left portion of Figure 2 consists of clock drivers for the PLA. VDD is supplied at the top and bottom of the PLA, and GND signals are gridded through the AND and OR plane cores.

3.2 Choosing the values of n, m, and k for PLAs

The values of n, m and k for each PLA in the array are found based on synthesis considerations as well as individual PLA delay considerations. For the PLA size we choose, we should be able to implement reasonably complex functions, with a reasonable delay as well.

We first characterized PLA delay as a function of n, m and k. We used the PLA device sizes per our final transistor level design (in which AND plane devices are minimum-sized). Then we performed a 3-D parasitic extraction for the wiring configuration of the PLA (obtained from the layout that we used) using Space3D [17]. We used the capacitance values from this extraction in our PLA spice decks. Our results are shown in Table 1.

<table>
<thead>
<tr>
<th>n</th>
<th>m</th>
<th>k</th>
<th>Delay (ps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>10</td>
<td>12</td>
<td>148.13</td>
</tr>
<tr>
<td>8</td>
<td>17</td>
<td>12</td>
<td>199.77</td>
</tr>
<tr>
<td>10</td>
<td>17</td>
<td>10</td>
<td>203.66</td>
</tr>
<tr>
<td>12</td>
<td>17</td>
<td>8</td>
<td>226.24</td>
</tr>
<tr>
<td>14</td>
<td>17</td>
<td>6</td>
<td>260.83</td>
</tr>
</tbody>
</table>

Table 1: Delay of a single PLA in ps (k = 12 rows)

Table 1 indicates that the delay of a PLA is quite heavily dependent on m, increasing by roughly 40ps for every 2 outputs. The increase in delay as a function of n is more gradual (a few ps for every 2 inputs). Finally, the dependence on k (which is not shown in Table 1 for which k = 12) is very slow (a few ps for every 2 rows added).

Synthesis constraints also play an important role in determining the size of the PLA. We performed several experiments on 7 designs to choose the best values of n, m and k. We decomposed each multi-level design into a array of PLAs using varying values of n, m and k as a constraint to the decomposition tool. Table 2 illustrates the results via the representative example s1488COMB (combinational part of the benchmark s1488). For this table, n = 12.

Table 2 shows that as k increases, the number of PLAs decreases. After k has increased beyond some threshold value, the number of PLAs stays constant. This threshold is around k = 12 for all seven designs. Beyond this threshold, we observe that the average height of a PLA saturates. In other words, even if we were to add PLA rows beyond this threshold, many of them would stay unfilled, since the number of inputs or outputs of the PLA have been used up. For very small values of m, the outputs get used up rapidly, even while inputs are about 50% utilized. This results in a large number of PLAs in the decomposition. After the number of outputs increased beyond a threshold, the number of PLAs remained relatively constant. A good trade-off appears to be m = 6, k = 12. For these values, the total number of PLAs is small and non-decreasing with increasing k, and the input and output utilization is high. Choosing m > 6 would increase the delay of each PLA by about 40ps per output pair, even though the
number of PLAs traversed in the topologically longest path from any circuit PI to any circuit PO remains unchanged. As a result, the circuit delay increases with \( m > 6 \). Also, increasing \( n \) beyond 12 does not change the topological depth of the design, and therefore results in a larger total circuit delay. As a result, we choose \( n = 12 \), \( k = 12 \) and \( m = 6 \) in our experiments. Similar trends were observed with the other six test examples as well. The total number of PLAs (and the topological depth of the PLA network) we obtained are larger that those reported in [5], since in our design approach, PLA folding is not performed. Further, the decomposition process in [5] was less constrained since there was only a maximum PLA width limit (as opposed to our situation, which requires a separate constraint for maximum \( n \) and \( m \)). Finally, we use multiple instances of a single PLA in our approach, in contrast to [5].

### 3.3 Synthesis for Metal/Via Programmable PLAs

Given a pre-fabricated (up to the metal layers) array of PLAs, we need to decompose the multi-level network \( \eta \) into a network of PLAs \( \mathcal{N} \) such that \( \mathcal{N} \) can be implemented using the array of PLAs by METAL and VIA mask customization. The wiring that is required is written to a mask. Using this mask, the pre-fabricated PLA array is customized or personalized to perform the required functionality of \( \eta \).

**Definition 1** The PLA dependency graph \( G(V, E) \) of a network of PLAs is a directed graph such that

- \( V = \{v_1, v_2, \ldots, v_f\} \), where each vertex \( v_i \) corresponds to a unique PLA in the network.
- \((v_i, v_j) \in E \) iff an output of PLA \( p_i \) is an input to PLA \( p_j \).

Our PLAs are precharged, and so we must ensure that the inputs to any PLA \( p \) must have evaluated before \( p \) evaluates. For this to be true, the PLA dependency graph should be acyclic. In addition, the evaluation of some PLA \( p \) must begin after the evaluation of the slowest PLA \( q \) such that \((q, p) \in E \). This is achieved by gating the evaluation signal of PLA \( p \) with the slowest completion signal among all PLAs that drive some input of \( p \). Since each PLA has a fixed delay, finding the appropriate completion signal to gate the evaluation of \( p \) is easy.

The problem of synthesis in this context is therefore defined as:

**Problem Statement 1** Given a multi-level logic network \( \eta \), decompose this network into a network \( \mathcal{N} \) of PLAs such that:

- This network \( \mathcal{N} \) is acyclic
- Each PLA in \( \mathcal{N} \) has at most \( n \) inputs
- Each PLA in \( \mathcal{N} \) has at most \( m \) outputs
- Each PLA in \( \mathcal{N} \) has at most \( k \) product terms

Our algorithm to perform the decomposition first levelizes the multi-level network. We then generate a DFS of network nodes and sort them in increasing order of their levels. We then greedily assign nodes of the multi-level network to a PLA until any PLA constraint is violated. A new node \( n \) is attempted to be included in the current PLA if \( n \in F0(m) \) (where \( m \) is included in the current PLA) and the inclusion of \( n \) in the PLA would not result in a cyclic PLA dependency graph. This reduces the wiring between PLAs. If such a node \( n \) is not available, then the next unmatched node in levelization order is selected. This is continued until all nodes of the multi-level network are matched.

For sequential designs, we first extract the combinational part of the design and decompose it into a network of PLAs. The latches are then re-instated into the design before placement and routing.

### 4 Experimental Results

In order to validate our design methodology, we first determined the values of \( n, m \) and \( k \) as discussed in Section 3. Using the values of \( n = 12, m = 6 \) and \( k = 12 \), we next decomposed several combinational and sequential circuits into a network of PLA with the above constraints on \( n, m \) and \( k \). The result was then placed and routed using the CADENCE SEDSM [19] tool-set. We wrote Perl [20] scripts to convert the synthesis output into a def file (for routing connectivity) and a lef file (for layout footprint, blockages and pin information of PLAs). We utilized 4 metal layers to perform the placement and routing. Placement of the individual PLAs was performed using QPLACE and routing between these PLAs was handled by WROUTE.

The aspect ratio of the final design was made as square as possible. Our PLAs were 21.5 \( \mu \)m wide and 12.25 \( \mu \)m high. Adding an additional 2.0 \( \mu \)m space above the PLA for inter-PLA routing, we obtained a overall PLA site 21.5 \( \mu \)m wide and 14.25 \( \mu \)m high. Given \( N \) PLAs in the decomposed design, suppose we require \( p \) rows and \( q \) columns to ensure a square aspect ratio. In that case, 21.5 \( \times \) 14.25 \( \times \) \( p \). This gives us a relationship between \( p \) and \( q \). Also, \( pq = N \). From these equations, we find the optimal \( p \) and \( q \) values to ensure a square aspect ratio of the final design.

For standard cell based comparisons, we technology mapped our design to a library consisting of a total of 20 standard cells (including a DFF). Placement and routing for the standard cell based designs was performed using the same 0.1 \( \mu \)m process, using SEDSM [19]. 4 metal layers were used for routing in all cases.

To compute the delay of our design, we first found the longest topological path between any circuit PI and any circuit PO. This number was multiplied by the delay of each PLA of size \( n = 12 \), \( m = 6 \) and \( k = 12 \). A delay of 370ps is incurred in the precharging of PLAs. For delay estimates of the standard cell implementation, we used the sense [21] package. This package returns the longest sensitizable path in a design (as opposed to static timing analysis which returns potentially false paths).

We experimented with a large number of examples (mostly sequential, with some combinational).

Table 3 shows the areas and delays of the designs using the Standard Cells and our PLAs. The overheads shown are the ratio of the

<table>
<thead>
<tr>
<th></th>
<th>Average Number of Outputs (out of m)</th>
<th>Average Number of Inputs (out of n)</th>
<th>Average PLA Height (maximum is 9.14)</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>m</td>
<td>n-1</td>
<td>m-1</td>
</tr>
<tr>
<td>6</td>
<td>1.92</td>
<td>2.00</td>
<td>3.12</td>
</tr>
<tr>
<td>12</td>
<td>1.96</td>
<td>2.14</td>
<td>4.05</td>
</tr>
</tbody>
</table>

Table 2: Decomposition characteristics of s1488COMB (n = 12)
areas/delays of our PLA approach compared to the standard cell approach. Our approach had a reasonable area overhead of 4.96× (3.44×) over a standard cell based implementation for combinational (sequential) designs. In reality the notion of area overhead is a confusing one and probably not a fitting one in the context of our design style. In practice, all that matters is whether a design can fit into a certain sized pre-fabricated die. However, for a fair comparison of the area utilization of our scheme, we compare the area of the design using our method (using the minimum number of PLAs) with that of a standard cell based implementation. Our method has a delay overhead of 2.89× (3.58×) over standard cells for combinational (sequential) circuits.

Table 3: Comparison of our technique with standard cells

<table>
<thead>
<tr>
<th>Ckt</th>
<th>Stdcell</th>
<th>PLA</th>
<th>Ovh</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1002</td>
<td>272.66</td>
<td>517.24</td>
<td>2.89</td>
</tr>
<tr>
<td>C1004</td>
<td>272.66</td>
<td>517.24</td>
<td>2.89</td>
</tr>
<tr>
<td>C1006</td>
<td>272.66</td>
<td>517.24</td>
<td>2.89</td>
</tr>
<tr>
<td>C1008</td>
<td>272.66</td>
<td>517.24</td>
<td>2.89</td>
</tr>
</tbody>
</table>

These results show that our PLA based approach has reasonable overheads compared to standard cells, and achieves a good middle ground between the speed of ASICs and the flexibility of field programmable design approaches, while requiring only METAL and VIA mask customization.

5 Conclusions and Future Work

In order to deal with the increasing costs of fabricating a VLSI chip we suggest a method of implementing a design using a dynamic PLA based approach with the aim of reducing the NRE involved. Our approach utilizes an array of identical dynamic medium-sized PLAs, whose outputs can be connected to a co-located flip-flop. The array of PLAs can be connected in a network using METAL and VIA customization to implement an arbitrary design. We found the optimal values of the number of inputs, outputs and wordlines of each PLA in the array and implemented an algorithm to technology map a PLA to the design which can tremendously reduce the number of PLAs used, which can not only reduce the area overhead, but also reduce the delay penalty since the topological depth of the circuit will also be reduced. This would require a modification of the PLA cell layout to enable programmability in connecting the inputs to the NMOS transistors in the AND plane. It has been found that this will lead to an increase in the number of vertical tracks in the AND plane. In this new scheme, the vertical width for each input would increase by 1.5×. However, we could get back this area loss by including folding and by reducing the vertical width in the OR plane (on account of reducing the number of DFFs and hence the output wiring pitch). Rather than reduce the number of DFFs in the PLA, we could alternatively pull the DFFs out of the PLA cell and place them at regular intervals between perhaps pairs of PLAs. Also, experiments would have to be repeated to determine again what the optimal size of the PLA would be with folding implemented.

References