

# An LPGA with Foldable PLA-Style Logic Blocks

Jason Helge Anderson and Stephen Dean Brown

Department of Electrical and Computer Engineering

University of Toronto

10 King's College Road

Toronto, Ontario, Canada M5S 3G4

{janderslbrown}@eecg.toronto.edu

## ABSTRACT

*Laser-programmed gate arrays (LPGAs) represent a new approach to application specific integrated circuit prototyping and implementation. This paper proposes a new LPGA logic block architecture called a foldable PLA-style logic block. The proposed logic block architecture is similar to that found in commercially available CPLDs. The term foldable means that the granularity of the logic block can be varied. This is achieved using the LPGA laser disconnect methodology. A custom CAD tool has been developed to map circuits into the new logic block architecture. An experimental study shows that LPGAs with foldable logic blocks are more area-efficient than those based on normal unfoldable logic blocks.*

## 1. INTRODUCTION

Laser-programmed gate arrays (LPGAs) are becoming a popular choice for ASIC prototyping and implementation because of their short programming time, high speed, and high logic density. An LPGA is a VLSI chip consisting of a two-dimensional array of logic blocks and a configurable interconnection network. Similar to field-programmable gate arrays (FPGAs) and complex programmable logic devices (CPLDs), all of the mask layers in an LPGA are pre-defined by the manufacturer. An unprogrammed LPGA has all possible metal connections between logic blocks. The LPGA is programmed by using a laser to permanently cut away some of the pre-defined metal links according to a user's design specifications. LPGA laser programmers may eventually be available to ASIC designers, thus allowing LPGAs to be labelled as "field-programmable devices."

In recent years, the issue of what type or size of logic block provides the best area-efficiency has been an active area of research and development for a number of different types of chips. For MPGAs, logic blocks have traditionally been relatively fine-grained [29][12][16][13]. For FPGAs, logic blocks are usually based on look-up-tables (LUTs) [28][2][19], or multiplexers [1]. In CPLDs, PLA-style blocks are used [3][21][17][9]. These blocks have relatively high granularity, having a large number of inputs per block.

Permission to make digital/hard copies of all or part of this material for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copyright is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires specific permission and/or fee.

FPGA 98 Monterey CA USA

Copyright 1998 ACM 0-89791-978-5/98/ 01..\$5.00

Existing LPGAs have fine-grained logic blocks. For example, the Chip Express CX2001 LPGA [8] has small, multiplexer-based blocks. It is an open research question as to what type of logic block is best for an LPGA. The architectural issues for LPGAs are similar to those for FPGAs and CPLDs, since in all of these technologies a fixed amount of interconnect and logic is pre-fabricated on a chip. However, the capability in LPGA technology to cut metal lines with little area overhead introduces new architectural possibilities. The focus of this paper is to investigate the benefits of using coarse-grained logic blocks in LPGAs in a way that leverages the ability to cut metal lines. We introduce a new logic block architecture called a foldable PLA-style logic block. It is a variation on the logic blocks found in CPLDs. The term *foldable* refers to the fact that the granularity of the logic block can be varied, by using the laser disconnect methodology.

A significant advantage of variable logic block granularity is that it facilitates "packing" additional logic into each logic block. This reduces the number of logic blocks needed to implement circuits and can increase area-efficiency. Coarse-grained blocks often suffer from under-utilization. For example, when circuits are mapped into traditional PLA-style logic blocks, a portion of some of the logic blocks is left unused and, therefore, wasted. In the proposed foldable logic blocks, the unused portion of a block may be separated from the used portion, and logic may then be implemented in the unused portion. Figure 1 is an abstract illustration of how additional logic is packed into foldable logic blocks. The figure shows two implementations of an arbitrary digital circuit. The left side of the figure depicts the circuit after it has been mapped into normal unfoldable logic blocks. The shaded portion of each logic block represents the used portion of the block, while the unshaded portion represents wasted area. The right side of the figure shows the same circuit after it has been mapped into foldable logic blocks. In the folded implementation, the logic blocks are better utilized. Furthermore, fewer logic blocks are needed in the folded implementation. Section 4 shows the effect of folding on reducing the overall silicon area needed for the circuit.

This paper is organized as follows: Section 2 elaborates on the architecture of the proposed logic blocks. Section 3 discusses a custom CAD tool that has been developed to map circuits into LPGA architectures with foldable PLA-style logic blocks. Section 4 presents the results of an experimental study on the area-efficiency of the proposed architectures. Final remarks and suggestions for future work are offered in Section 5.

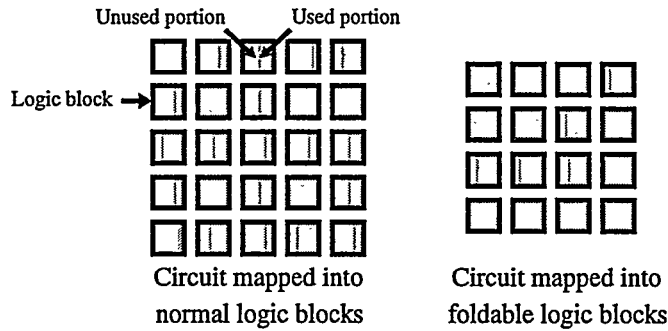


Figure 1. Packing additional logic into foldable logic blocks.

## 2. ARCHITECTURE OF FOLDABLE PLA-STYLE LOGIC BLOCKS

A PLA is characterized by its number of inputs, product terms, and outputs. The logic function implemented by a PLA can be described using a personality matrix [27]. A personality matrix for two combinational functions is shown in Figure 2. The rows of the personality matrix correspond to product terms, while the columns correspond to inputs and outputs. A '1' in an input column indicates that an input is present in its 'true' form in a product term; a '0' indicates that an input is present in its complemented form; a '-' represents a "don't care" and indicates that an input is not used in a product term. The '1', '0', and '-' have similar meanings when used in an output column, indicating whether or not a product term is present in the sum-of-products form of the function corresponding to the output.

PLA folding was first introduced as a method for reducing the silicon area consumed by PLAs in custom VLSI. A PLA's area is proportional to the number of columns in its personality matrix multiplied by the number of rows in its personality matrix [10][20]. Folding leverages the high percentage of "don't cares" in personality matrices and reduces area by allowing two columns of a personality matrix to reside on a single physical column (*column folding*), or by allowing two rows of a personality matrix to reside on a single physical row (*row folding*). Column folding is illustrated in Figure 3. A normal unfolded PLA is shown in part *a* of the figure. To keep the figure simple, a single column is used to represent both the true and complemented versions of each input signal. A folded PLA in which three column pairs are folded onto single physical columns is shown in part *b* of the figure. Notice the "breaks" that occur on the folded columns. Column folding eliminates columns from a PLA; row folding eliminates product term rows from a PLA. It is also possible to combine row and column folding. *Combined* folding can be applied to eliminate *both* rows and columns from a PLA; a combined folded PLA has breaks on both its columns and its rows. The amount of folding in a PLA is quantified by a parameter called the *size* of the folding [10]. This parameter is equal to the number of columns or rows eliminated from the original PLA. The size of the column folding in Figure 3 is three.

Inputs		Outputs	
abcdghij		fk	
11-----		10	$f = a \cdot b + c + d$
--1-----		10	$k = g \oplus h + i \cdot j$
---1-----		10	
----01--		01	
----10--		01	
-----11		01	

Figure 2. An example of a PLA personality matrix.

PLAs can be folded by cutting either physical input columns or physical product term rows. These structures can be implemented using the metalization layers in a VLSI chip. Since metal lines can be cut in LPGA technology, it is possible to build an LPGA with foldable PLA-style logic blocks. In such an architecture, each PLA-style logic block in the array has a fixed number of physical input columns, product term rows, and outputs. Folding is applied to facilitate packing additional logic into each logic block. As an example, consider the PLA and logic block shown in the top portion of Figure 4. The logic block has five input columns, five product term rows, and two outputs. Each  $\times$  in the figure represents a programmable switch; switches are programmed to realize product terms and logical sums of product terms. Clearly, the PLA shown in the figure does not fit into the logic block, because it needs six product term rows and seven input columns. However, it is easy to fit the PLA into the block by using folding. The bottom part of Figure 4 shows how two columns and one row can be folded to accommodate the PLA. The notion of an array of foldable PLA-style logic blocks in an LPGA represents an entirely new application for PLA folding, since it has previously been applied only for a single custom-fabricated PLA.

The architecture of a foldable PLA-style logic block is summarized in Figure 5. It is characterized by the number of input columns ( $I$ ), product term rows ( $P$ ), and outputs ( $O$ ), along with the type of folding that is permitted for the block. A PLA-style logic block may be unfoldable, row foldable, column foldable, or combined foldable. The parameters of a PLA-style block are expressed using the tuple, ( $I, P, O$ ).

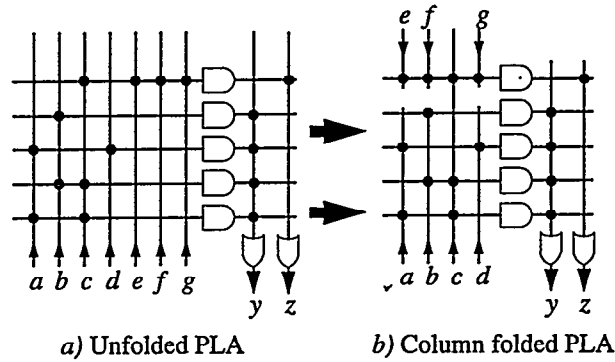


Figure 3. PLA column folding.

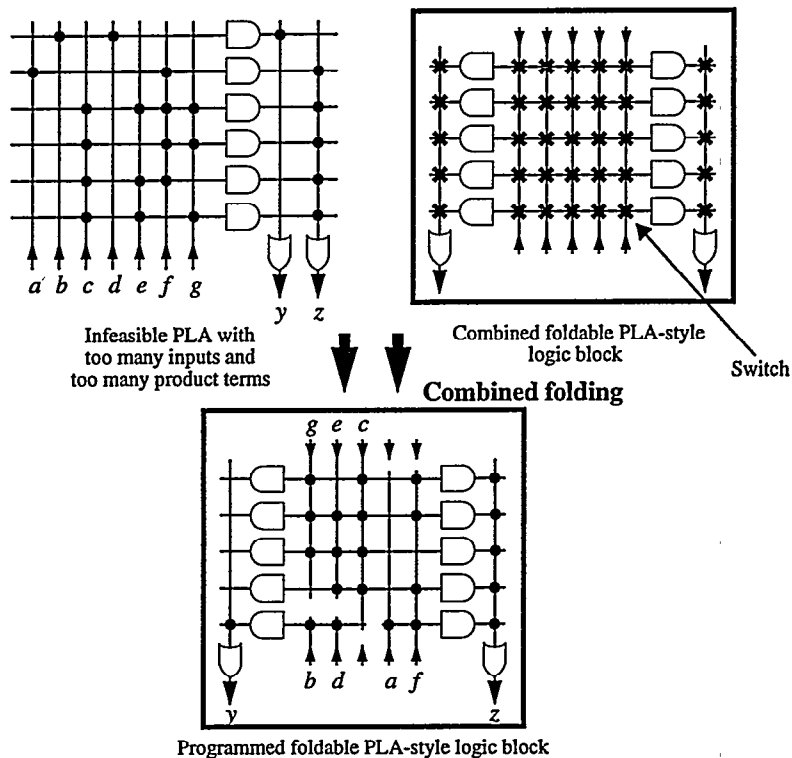


Figure 4. Combined folding to pack additional logic into a foldable PLA-style logic block.

Row and combined foldable logic blocks have two OR-planes; hence, in these blocks, the outputs are divided such that there is an equal number in each OR-plane. Column and combined foldable logic blocks allow signals to enter the PLA from both the top and bottom of the AND-plane. Note that a column foldable PLA-style block with  $I$  input columns actually has  $2 \times I$  inputs, whereas an unfoldable PLA-style block with  $I$  input columns has  $I$  inputs. Figure 5 shows the laser cut points that are necessary to make the logic block row, column, or combined foldable. Although not shown in Figure 5, we assume that each output of the block has an associated flip-flop, which can either be used or by-passed.

There are many ways in which the block in Figure 5 can be folded. In this paper, we consider only simple bipartite folding. In simple PLA folding [27], each column or row of

a folded PLA is allowed to have at most one break. Bipartite folding [15] requires that all of the breaks in the folded PLA occur at the same level - same vertical level for column folding, same horizontal level for row folding.

### 3. SYNTHESIS TECHNIQUES FOR FOLDABLE PLA-STYLE LOGIC BLOCKS

The following CAD flow is used to implement circuits in foldable PLA-style blocks. First, the circuits are synthesized using the Synopsys Design Compiler. This results in an optimized netlist of gates from an intermediate library. The intermediate library consists of elements suitable for mapping circuits into PLA-based architectures. The netlist specifies the logic functions in the circuit, which may be two-level or multi-level functions. The netlist can be represented by a directed acyclic graph (DAG). Each node in the DAG represents a single logic function in the circuit.

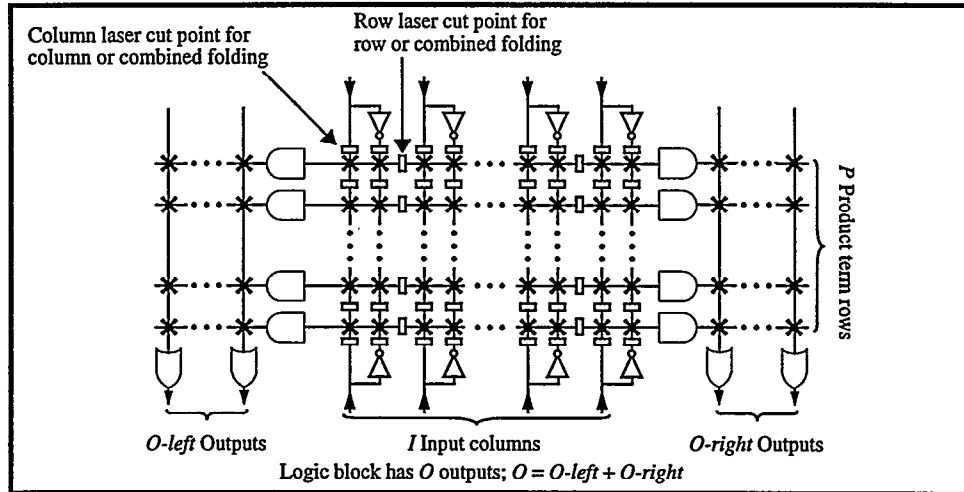


Figure 5. Foldable PLA-style logic block.

The netlist is technology mapped into the PLA-style logic blocks, using a custom-developed tool.

PLA-style blocks are referred to as coarse-grained logic blocks because they have a large number of inputs, and hence can realize logic functions of many inputs. Traditional technology mapping, in which a library represents all possible logic functions that are to be used, is not feasible for PLA-based architectures because the library would be too large to be repetitively searched during technology mapping. Several CAD tool vendors offer products that can perform technology mapping for PLA-style blocks. However, the tools use proprietary algorithms, which are not publicized. One of the few publicized approaches for mapping circuits into PLA-style blocks is found in [14]. It first uses a LUT-based technology mapper and sets the number of LUT-inputs to the number of PLA-style block inputs. Then, nodes whose number of product terms is too large for the PLA-style blocks are decomposed into smaller nodes. Another approach to technology mapping is to collapse circuits completely into two-level form, thus creating some nodes with more inputs or product terms than allowable in the PLA-style blocks. These large infeasible nodes can then be partitioned into smaller feasible ones and packed into multi-output logic blocks.

We have developed a new technology mapping CAD tool for PLA-based architectures. We believe that it provides excellent results in comparison to other approaches, as mentioned above. There is insufficient space in this paper to describe the technology mapping in detail, but we will give the reader an idea of the approach used. Our algorithm works in three phases. The first phase breaks up the DAG representing a circuit into a forest of trees and then maps each tree into a new tree possessing the minimum number of PLA-feasible nodes; PLA-feasible nodes possess a number of inputs and a number of product terms that allow them to fit into the targeted logic blocks. Phase II is a partial collapsing step that attempts to eliminate multi-fanout nodes from the network by collapsing them into their successors; that is, we attempt to collapse nodes across tree boundaries.

Phase III is a bin packing step in which the nodes in a circuit are packed into the targeted multi-output logic blocks. Our tool attempts to minimize the number of logic blocks in the final mapped circuit.

Egan and Liu showed that the problem of finding an optimal size bipartite folding is NP-complete [10]. Furthermore, previous work has focused on using folding to reduce the area of a single PLA, rather than using folding to pack additional logic into PLA-style logic blocks. We have integrated a heuristic folding algorithm into phases II and III of our technology mapper. When the target logic blocks are column or combined foldable, folding is used in phase II to allow more nodes to be collapsed into their successors and eliminated from the network. In this case, a node can be collapsed into its successors as long as the resultant nodes after collapsing have less than or equal to  $P$  product terms, and less than or equal to  $F$  inputs.  $F$  may be larger than  $I$  as long as a column folding can be found such that  $F$  minus the size of the folding is less than or equal to  $I$ . In phase III of our algorithm, we use folding to pack additional nodes into each multi-output block in an attempt to utilize all of a logic block's outputs. The folding algorithm we use is similar to that developed by Liu and Wei [18], which involves transforming the folding problem into an equivalent min-cut graph partitioning problem. Specifically, we have adapted the algorithm in [18] to be able to perform combined folding. Our algorithm is described in detail in [4].

#### 4. EXPERIMENTAL STUDY OF LPGAs WITH FOLDABLE PLA-STYLE LOGIC BLOCKS

An empirical approach is used to study the benefits of the proposed foldable logic block architectures. Experiments consist of mapping a set of 30 benchmark circuits into experimental LPGa architectures. Architectural parameters are varied to study the effect they have on the mapping solutions. The parameters  $I$ ,  $P$ , and  $O$  are varied, and the effects of row folding, column folding, and combined folding are investigated. A total of 19 of the 30 circuits used

in this study are large MCNC benchmarks [30]. Ten circuits are HDL specifications developed at the University of Toronto. The last circuit is a processor benchmark from the PREP synthesis suite [22].

#### 4.1 Area Models

To determine the relative area-efficiencies of the foldable architectures, area models are used. The models assume that silicon area is consumed by a combination of logic and routing, with the possibility that some routing may be placed in metalization layers directly on top of active logic. Routing on top of active logic is feasible in LPGA technology because the routing circuitry present in LPGAs is entirely metal.

Since the amount of routing resources that may be placed on top of active logic is limited by the area of each logic block and the laser cut points needed to configure the logic circuitry, two area models are used: one pessimistic, the other optimistic. These two models are depicted in Figure 6. The pessimistic model assumes that only vertical routing tracks may be placed on top of active logic. The optimistic model assumes that it is possible for both horizontal and vertical routing resources to be located on top of logic.

A *basic tile* is defined to be the area of a single logic block and its adjacent routing circuitry. The basic tile structure is shown in Figure 6. Using the pessimistic area model, the area of a basic tile is:

$$\begin{aligned} \text{TileArea}_{\text{pess}} &= \text{height} \times \text{width} = \\ &(\sqrt{LA} + W \cdot R_p) \times \max(\sqrt{LA}, W \cdot R_p) \end{aligned} \quad (1)$$

where  $W$  is the number of tracks in a routing channel,  $R_p$  is the routing pitch, and  $LA$  is the area of a logic block. For simplicity, logic blocks are assumed to be square. The *max* function reflects the notion that a vertical routing channel may be wider than a logic block. If the optimistic model is applied, then the area of a basic tile is:

$$\text{TileArea}_{\text{opt}} = \max(LA, (W \cdot R_p)^2) \quad (2)$$

The total area needed to implement a circuit is equal to the number of logic blocks needed multiplied by the tile area. The models assume that there are equal amounts of horizontal and vertical routing resources. An empirical study by Betz showed that this routing architecture leads to

the smallest possible routing resource area in FPGAs [5]. We measure both routing and logic area in terms of the technology independent parameter  $\lambda$  [20], which is equal to half the minimum feature size in a given technology. In typical LPGA technology, accounting for the overhead needed for laser cut-points,  $R_p$  is approximately equal to  $11\lambda$ .

##### 4.1.1 Chip Area of Foldable PLA-Style Logic Blocks

A chip area model for foldable PLA-style logic blocks was developed using a layout automatically generated by the PLA-layout-generation program MPLA [24]. The PLA layout is for a dynamic NOR/NOR PLA [20][25]. The logic block area models in this section were produced using the generated MPLA layout, and by estimating how the layout would need to be modified so that it could be configured using the laser disconnect methodology.

The area of an unfoldable PLA-style logic block is estimated as:

$$\begin{aligned} LA &= (24 \cdot I + 19 \cdot O + 58) \cdot (16 \cdot P + 44) + \\ &1000 \cdot I + 13500 \cdot O + 1000 \lambda^2 \end{aligned} \quad (3)$$

The term  $(24 \cdot I + 19 \cdot O + 58)$  is the combined width of the PLA's AND and OR-planes. The next term,  $(16 \cdot P + 44)$ , is the height of the PLA's AND-plane. The  $1000 \cdot I$  term accounts for the area consumed by the input buffers. The  $13500 \cdot O$  term includes the area consumed by the latch ( $\sim 3500 \lambda^2$ ), flip-flop ( $\sim 8000 \lambda^2$  [23]), and output driver ( $\sim 2000 \lambda^2$ ) that are present for each logic block output. The latch is needed for the PLA-style block to have zero stand-by power [26]. The final constant, 1000, represents the area needed to buffer and invert the signal used to clock the pull-up transistors in the AND and OR-planes. Row foldable PLAs have an OR-AND-OR structure and thus, require pull-up transistors on both sides of the AND-plane. The logic area of a row foldable logic block is estimated as:

$$\begin{aligned} LA &= (24 \cdot I + 19 \cdot O + 84) \cdot (16 \cdot P + 44) + \\ &1000 \cdot I + 13500 \cdot O + 1000 \lambda^2 \end{aligned} \quad (4)$$

Column foldable PLAs have extra inputs, and therefore, require additional input buffers. The logic area of a column foldable PLA-style logic block is estimated as:

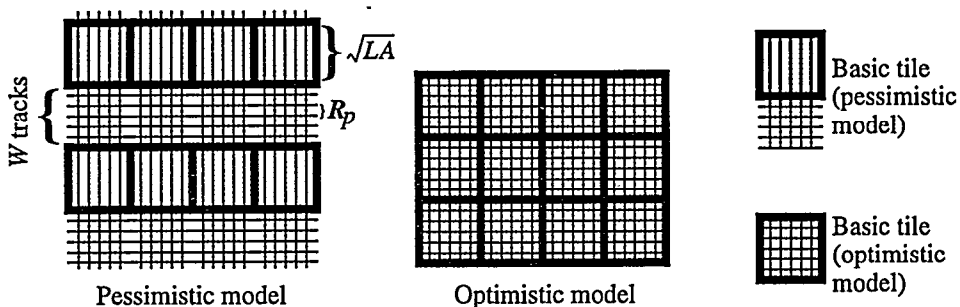


Figure 6. Pessimistic and optimistic area models.

$$LA = (24 \cdot I + 19 \cdot O + 58) \cdot (16 \cdot P + 44) + 1000 \cdot I \cdot 2 + 13500 \cdot O + 1000 \lambda^2 \quad (5)$$

Finally, the area of a combined foldable PLA-style logic block is:

$$LA = (24 \cdot I + 19 \cdot O + 84) \cdot (16 \cdot P + 44) + 1000 \cdot I \cdot 2 + 13500 \cdot O + 1000 \lambda^2 \quad (6)$$

Low power is an important consideration in many LPGA applications. The PLA that was used to generate the models above is very similar to the zero stand-by power PLA described in [26], with the main difference being that it contains no input transition detection circuitry. However, this circuitry is relatively small in comparison with the other circuit structures internal to the PLA.

#### 4.1.2 Estimation of Routing Resource Area

We do not currently have a routing tool that can use the pins on foldable logic blocks effectively, and therefore we use theoretical results to predict routing resource area. In [11], El Gamal showed that the average number of used tracks in any channel of a gate array with equal amounts of horizontal and vertical routing resources is given by:

$$W_{avg} = \frac{\lambda_{pins} \bar{R}}{2} \quad (7)$$

where  $\lambda_{pins}$  is the average number of connected input pins per logic block for circuits implemented in the gate array, and  $\bar{R}$  is the average manhattan length of two-point routing connections (measured in the number of blocks). The parameter  $\lambda_{pins}$  is known after technology mapping is complete; however, placement and routing must be completed to ascertain  $\bar{R}$ . In this study, the value of  $W_{avg}$  for a circuit implemented in a foldable architecture is estimated using equation (7) above, and the  $\bar{R}$  value that results from performing placement and global routing for the circuit in the unfoldable architecture with the same parameters, ( $I, P, O$ ). Placement and routing is done using the CAD system, VPR [6]. We assume that the clock, set, and reset signals feeding the flip-flops in each logic block are routed on dedicated tracks, and that the pins on logic blocks are distributed, with some pins being accessible to horizontal routing resources, and some being accessible to vertical routing resources.

It is assumed that the maximum number of used tracks in any routing channel,  $W$ , is greater than the average number of used tracks,  $W_{avg}$ . When each circuit is placed and routed in an unfoldable architecture, the ratio of  $W$  to  $W_{avg}$  can be computed. This ratio is then used to compute the number of tracks,  $W$ , needed to route the circuit in a foldable architecture; the ratio is used by multiplying it by the value of  $W_{avg}$  that is computed using equation (7) after the circuit is mapped into the foldable architecture.

## 4.2 Experimental Results

The results of this experimental study are presented in two ways: 1) as a reduction in the number of blocks needed to implement circuits, and 2) an area is presented for each experimental architecture; the area is computed using the area models described in the previous sections.

The left-hand column of plots in Figure 7 depicts the results for row folding. The top, centre, and bottom plots in the column give the results for architectures with 3, 4, and 5 outputs, respectively. The vertical axis indicates the average percentage reduction in the number of logic blocks needed to implement a circuit when logic blocks are row foldable in comparison with the number of logic blocks needed when logic blocks are unfoldable. This was computed by determining a percentage reduction for each of the benchmark circuits and then averaging these percentages. Thus, each benchmark (whether small or large) was treated equally. The figure indicates that row folding holds the most benefit for blocks that have very few product term rows and large numbers of input columns and outputs. Recall that row folding allows two product terms to be placed onto the same physical product term row. This sharing of physical product term rows is tantamount to increasing the number of product terms that may be placed into a logic block, which helps alleviate the effect of having relatively few product term rows.

The centre column of Figure 7 illustrates the benefits of column folding. It shows that in comparison to row folding, column folding is superior at reducing the number of blocks needed to implement circuits. Opposite to the results observed for row folding, column folding performs best when logic blocks have relatively few input columns in comparison with product term rows and outputs; that is, column folding is most useful when inputs are scarce. The maximum reduction of 43.1% occurs when column folding is permitted in architectures with 8 input columns, 24 product term rows, and 5 outputs.

One reason why column folding provides a greater reduction in the number of blocks than row folding is related to how choices are made with regard to which inputs are permitted to share a single physical input column, and which product terms are permitted to share a single physical product term row. In row folding, physical product term rows can only be shared by product terms belonging to outputs in different OR-planes. This is due to the OR-AND-OR structure of row foldable PLA-style logic blocks as shown in Figure 5. This restricts the number of pairs of product terms that may share a physical product term row. On the other hand, in column folding, no restrictions are placed on which inputs may share a physical column. Thus, it is even possible for two inputs to the same function to share a physical input column. Clearly, there are more degrees of freedom available in column folding than row folding, accounting for the more significant gains of column folding.

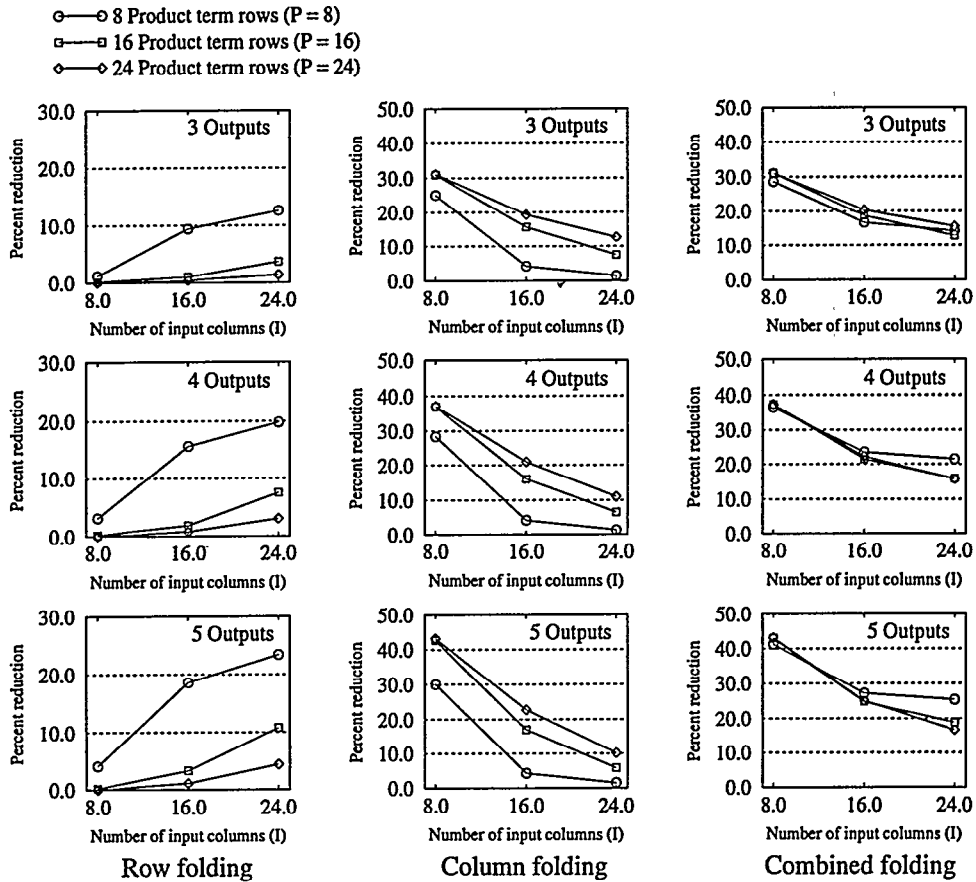


Figure 7. The benefits of PLA folding - percentage reduction in number of logic blocks.

The right-hand column of plots in Figure 7 depicts the gains of combined folding. The architecture with the largest gain is the same as that for the case of column folding. The main difference between the results for column and combined folding is that combined folding provides more significant gains for architectures with 16 and 24 input columns and architectures with 8 product term rows. For example, column folding alone provides only small benefits for architectures with 16 input columns, 8 product term rows, and 3 outputs; however, combined folding allows the number of blocks to be reduced by 16.6%. The figure shows that combined folding provides the benefits of both row and column folding.

Figure 8 shows the normalized area results for architectures with unfoldable PLA-style logic blocks. The three graphs give the results for architectures with 3, 4, and 5 outputs, respectively. The area consumed by each circuit in each architecture was normalized to the area consumed by the same circuit in an architecture containing combined foldable logic blocks with 8 input columns, 8 product term rows, and 4 outputs. These normalized area values for each circuit were then averaged and the results are shown in the figure. The combined foldable (8, 8, 4) architecture was determined to be the most area-efficient of all the foldable and unfoldable architectures considered. Figure 8 shows that the unfoldable architecture with the best area-efficiency

has the parameters (8, 8, 3). Other good architectures include the (16, 8, 3) architecture, the (16, 8, 4) architecture, and the (16, 16, 5) architecture. These architectures are reasonably similar to the architecture with 10-12 inputs, 12-13 product terms, and 3-4 outputs that was identified as the most area-efficient in [14].

Figure 9 depicts the normalized area results for architectures with foldable PLA-style logic blocks. The data points in the figure reflect the best folded area for each combination of parameters ( $I, P, O$ ). Thus, some of the values shown are the areas of combined foldable architectures, whereas others are area values for column or row foldable architectures. The data points in Figure 9 correspond with the data points in Figure 8. For example, the unfoldable architecture with the parameters (8, 24, 3) has an area of 2.20, and the corresponding foldable architecture has an area of 1.55. Notice that for most of the architectures considered, the folded area is significantly less than the unfolded area. Folding is particularly helpful at reducing area in architectures with 8 input columns and in architectures with 5 outputs.

Table 1 summarizes the area results for all of the architectures considered. It shows the best folded area achievable for each of the architectures, the type of folding used to achieve that area, and the unfolded area for each of

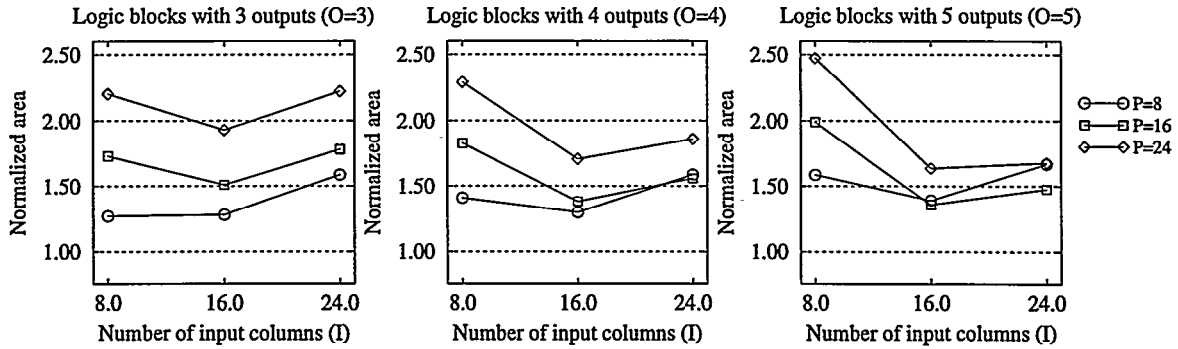


Figure 8. Area results for unfoldable PLA-style logic block architectures (Optimistic Model).

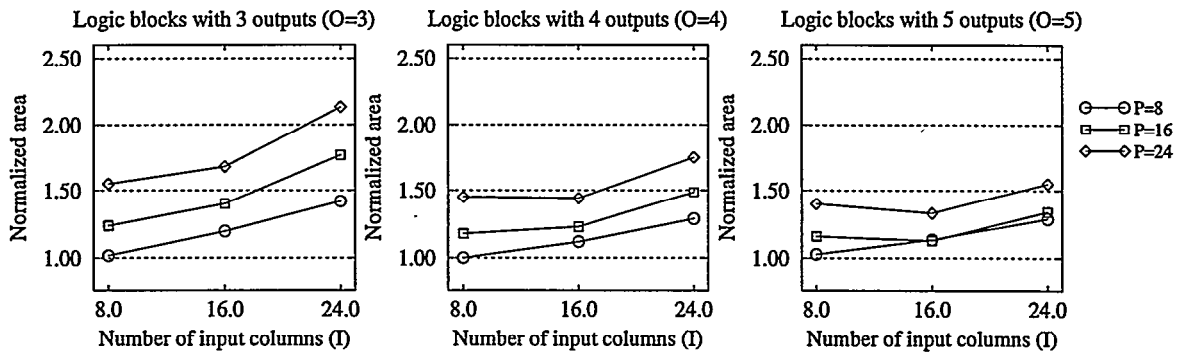


Figure 9. Area results for foldable PLA-style logic block architectures (Optimistic Model).

the architectures. Each cell of the table gives the area that results from applying the optimistic area model, and the area that results from applying the pessimistic area model (shown in parentheses). In the table cells describing folded area, RF is used to indicate row folding, CF is used to indicate column folding, and BF is used to indicate combined folding. As already mentioned, the combined foldable (8, 8, 4) architecture is the most area-efficient architecture. Other architectures with good area-efficiencies include the combined foldable architecture with the parameters (8, 8, 3), and the combined foldable architecture with the parameters (8, 8, 5). It is interesting to note that no single type of folding is best for all architectures; the best type of folding can be any of row, column, or combined folding depending on the parameters, ( $I, P, O$ ). The best unfoldable logic block architecture, with 8 input columns, 8 product term rows, and 3 outputs consumes 27% more area than the (8, 8, 4) combined foldable architecture (or 19% more area using the pessimistic area model). Thus, the results show that foldable PLA-style logic block architectures use significantly less area than the most area-efficient unfoldable architectures.

## 5. FINAL REMARKS AND FUTURE WORK

The experimental study described in the previous section showed that architectures based on the proposed foldable PLA-style logic blocks are significantly more area-efficient than normal PLA-based architectures. The notion of an array of foldable PLA-style logic blocks represents a new application for PLA folding, which has previously been used only in custom VLSI. Another advantage of the

proposed logic blocks is that they are coarse-grained in comparison with the logic blocks in traditional MPGAs or existing LPGAs. Designs implemented using coarse-grained blocks have fewer logic levels on their critical paths. This is advantageous because logic blocks on the critical path are connected using the programmable interconnection network, and as feature sizes shrink in VLSI technology, interconnect delay is becoming a more significant portion of total delay. Hence, using coarse-grained logic blocks may improve speed-performance, without reducing area-efficiency.

In the future, we will investigate whether the proposed logic blocks are more area-efficient than the logic blocks in existing state-of-the-art LPGAs, like the Chip Express CX2001.

## 6. ACKNOWLEDGEMENTS

The authors gratefully acknowledge the financial support of Chip Express Corporation and the Government of Ontario.

## 7. REFERENCES

- [1] *ACT 1 Series FPGAs Data Sheet*, Actel Corporation, 1996.
- [2] *The Altera Data Book*, Altera Corporation, 1996.
- [3] *The MACH 5 Family Data Sheet*, Advanced Micro Devices, 1996.
- [4] J. H. Anderson, "Architectures and Algorithms for Laser-Programmed Gate Arrays with Foldable Logic Blocks", *M.A.Sc. Thesis*, Department of Electrical and Computer Engineering, University of Toronto, 1997.
- [5] V. Betz and J. Rose, "Directional Bias and Non-Uniformity in FPGA Global Routing Architectures", *IEEE/ACM International Conference on Computer-Aided Design*, 1996, pp. 652-659.

I, P	O = 3		O = 4		O = 5	
	Folded	Unfolded	Folded	Unfolded	Folded	Unfolded
8, 8	1.02 (1.01) BF	1.27 (1.19)	1.00 (1.00) BF	1.41 (1.31)	1.03 (1.02) BF	1.59 (1.44)
8, 16	1.24 (1.19) CF	1.73 (1.54)	1.18 (1.11) CF	1.83 (1.61)	1.16 (1.11) CF	1.99 (1.75)
8, 24	1.55 (1.42) CF	2.20 (1.89)	1.45 (1.32) CF	2.29 (1.96)	1.40 (1.30) CF	2.47 (2.10)
16, 8	1.20 (1.14) RF	1.28 (1.21)	1.12 (1.09) RF	1.30 (1.22)	1.14 BF (1.11 RF)	1.39 (1.29)
16, 16	1.40 (1.31) BF	1.51 (1.40)	1.23 (1.16) BF	1.38 (1.26)	1.13 (1.08) BF	1.36 (1.25)
16, 24	1.68 (1.51) CF	1.93 (1.68)	1.44 (1.31) CF	1.71 (1.51)	1.33 (1.23) CF	1.64 (1.46)
24, 8	1.42 (1.32) RF	1.59 (1.44)	1.29 (1.21) RF	1.59 (1.42)	1.29 (1.21) RF	1.67 (1.50)
24, 16	1.77 BF (1.57 RF)	1.78 (1.57)	1.49 (1.33) BF	1.56 (1.39)	1.34 (1.24) BF	1.48 (1.34)
24, 24	2.13 (1.83) CF	2.22 (1.90)	1.75 (1.53) BF	1.86 (1.60)	1.55 (1.39) BF	1.68 (1.49)

Table 1. Normalized area results for PLA-based LPGA architectures.

- [6] V. Betz and J. Rose, "On Biased and Non-Uniform Global Routing Architectures and CAD Tools for FPGAs", *CSRI Technical Report #358*, Department of Electrical and Computer Engineering, University of Toronto, 1996.
- [7] *Chip Express Technology Overview*, Chip Express Corporation, 1996.
- [8] *Chip Express Technology and CAD Tool Workshop Notes*, Chip Express Corporation, Santa Clara, California, July 1996.
- [9] *UltraLogic High-Performance CPLD Data Sheet*, Cypress Semiconductor, 1997.
- [10] J. R. Egan and C. L. Liu, "Bipartite Folding and Partitioning of a PLA", *IEEE Transactions on Computer-Aided Design*, Vol. CAD-3, No. 3, July 1984, pp. 191-199.
- [11] A. El Gamal, "Two-Dimensional Stochastic Model for Interconnections in Master Slice Integrated Circuits", *IEEE Transactions on Circuits and Systems*, Vol. CAS-28, No. 2, February 1981, pp. 127-138.
- [12] J. D. Gallia, R. J. Landers, C. Shaw, T. Blake and W. Banzhaf, "A Flexible Gate Array Architecture for High-Speed and High-Density Applications", *IEEE Journal of Solid-State Circuits*, Vol. 31, No. 3, March 1996, pp. 430-435.
- [13] M. Khatakhotan, "Interleaved Channeless Gate Array Architecture", *IEEE 1992 Custom Integrated Circuits Conference*, pp. 27.1.1-27.1.4.
- [14] J. L. Kouloheris, "Empirical Study of the Effect of Cell Granularity on FPGA Density and Performance", *Ph. D. Thesis*, Department of Electrical Engineering, Stanford University, 1993.
- [15] Y. S. Kuo, C. Chen and T. C. Hu, "A Heuristic Algorithm for PLA Block Folding", *22nd Design Automation Conference*, 1985, pp. 744-747.
- [16] M. Hashimoto, S. S. Mahant Shetti and J. D. Gallia, "New Base Cell for High Density Gate Array", *IEEE 1992 Custom Integrated Circuits Conference*, pp. 27.2.1-27.2.4.
- [17] *ispLSI and pLSI 6000, 3000 CPLD Datasheet*, Lattice Semiconductor, 1996.
- [18] B. Liu and K. Wei, "An Efficient Algorithm for Selecting Bipartite Row or Column Folding of Programmable Logic Arrays", *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications*, Vol. 41, No. 7, July 1994, pp. 494-498.
- [19] *ORCA OR3C/OR3T Series FPGA Product Brief*, Lucent Technologies, 1996.
- [20] C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley Publishing Company, Don Mills, Ontario, 1980.
- [21] *CoolRunner CPLD Data Sheet*, Philips Semiconductors, 1997.
- [22] *Programmable Electronics Performance Corporation Test Benches*, <http://www.prep.org>, 1996.
- [23] J. Rose, R. J. Francis, D. Lewis and P. Chow, "Architecture of Field-Programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency", *IEEE Journal of Solid-State Circuits*, Vol. 25, No. 5, October 1990, pp. 1217-1225.
- [24] W. S. Scott et al., "1986 VLSI Tools: Still More Works by the Original Artists", *Technical Report UCB/CSD 86/272*, University of California at Berkeley, 1985.
- [25] Neil H. E. Weste and Kamran Eshraghian, *Principles of CMOS VLSI Design*, Addison-Wesley Publishing Company, Don Mills Ontario, 1993.
- [26] S. Wong, H. So, C. Hung and J. Ou, "Novel Circuit Techniques for Zero-Power 25-ns CMOS Erasable Programmable Logic Devices (EPLD's)", *IEEE Journal of Solid-State Circuits*, Vol. SC-21, No. 5, October 1986, pp. 766-773.
- [27] D. F. Wong, H. W. Leong and C. L. Liu, "PLA Folding by Simulated Annealing", *IEEE Journal of Solid-State Circuits*, Vol. SC-22, No. 2, April 1987, pp. 208-215.
- [28] *The Programmable Logic Data Book*, Xilinx Corporation, 1994.
- [29] H. Veendrick, D. van den Elshout, D. Harberts and T. Brand, "An Efficient and Flexible Architecture for High-Density Gate Arrays", *1990 IEEE International Solid-State Circuits Conference*, pp. 86-87.
- [30] S. Yang, "Logic Synthesis and Optimization Benchmarks", *Technical Report, Microelectronics Center of North Carolina*, 1991.