Cross-talk Immune VLSI Design using a Network of PLAs Embedded in a Regular Layout Fabric

Sunil P. Khatri (spkhatri@colorado.edu) Robert K. Brayton (brayton@ic.eecs.berkeley.edu)
Alberto Sangiovanni-Vincentelli (alberto@ic.eecs.berkeley.edu)

Abstract

We present a VLSI design methodology to address the cross-talk problem, which is becoming increasingly important in Deep Sub-Micron (DSM) IC design. In our approach, we implement the logic netlist in the form of a network of medium sized PLAs. We utilize two regular layout "fabrics" in our methodology, one for areas where PLA logic is implemented, and another for routing regions between such logic blocks. We show that a single PLA implemented in the first fabric style is not only cross-talk immune, but also about 2$\times$ smaller and faster than a traditional standard cell based implementation of the same logic. The second fabric, utilized in the routing region between individual PLAs, is also highly cross-talk immune. Additionally, in this fabric, power and ground signals are essentially "pre-routed" all over the die.

Our synthesis flow involves decomposing the design into a network of PLAs, each of which has a bounded width and height. The number of inputs and outputs of each PLA are flexible as long as the resulting PLA width is bounded. We perform folding of PLAs to achieve better logic density. Routing is performed using 2, 3, 4, 5 and 6 routing layers. State-of-the-art commercial routing tools are utilized for the experiments involving the use of 3, 4, 5 and 6 routing layers.

We have implemented the entire design flow using these ideas. Our scheme results in a reduction in the cross-talk between signal wires of between one and two orders of magnitude. As a result, for a 0.1μm process, the delay variation due to cross-talk dramatically drops from 2.47:1 to 1.02:1. Additionally, our methodology results in circuits that are extremely fast and dense, with a timing improvement of about 15% and an overall area penalty of about 3% compared to standard cells. The regular arrangement of metal conductors in our scheme results in low and highly predictable inductive and capacitive parasitics, resulting in highly predictable designs. The crosstalk immunity, high speed, low area overhead and high predictability of our methodology indicate that it is a strong candidate as the preferred design methodology in the DSM era.

1 Introduction and Previous Work

As the minimum feature size of VLSI fabrication processes decreases to deep sub-micron (DSM) levels, several new challenges are being faced. Certain electrical problems like cross-talk, electromigration, self-heat and statistical process variations are becoming increasingly important. Until recently, IC designers were able to cleanly partition the design task into a logical and a physical one, with no interaction between the two subtasks. The increasing importance of the above electrical effects requires that designers consider the interaction between logical and physical design at the same time. This makes the design task more complex and time-consuming.

The *cross-talk* problem is perhaps the most important effect which jeopardizes the ability of designers to abstract the logical and physical aspects of design. Cross-talk typically occurs between adjacent wires on the same metal layer, when the cross-coupling capacitance between these wires is large enough for them to affect each other's electrical characteristics. As the minimum feature size of VLSI fabrication processes reaches the 0.1μm range, process engineers are forced to increase the height of wires in relationship to their width, in order to keep their sheet resistivity from increasing quadratically. This in turn increases the cross-coupling capacitance between a wire and its neighbors as a fraction of its total capacitance, resulting in cross-talk problems. In particular, cross-talk can cause a significant delay variation in a wire depending on the electrical state of neighboring wires. Also, it can cause the logic value of a wire to be incorrectly interpreted depending on the state of neighboring aggressor wires, resulting in a loss of signal integrity. With the decreasing minimum feature size of VLSI fabrication processes, these problems are becoming increasingly common [1].

In this paper, we present a design flow to alleviate the cross-talk problem. Our scheme is motivated by the fabric concept introduced in [2]. In both approaches, the cross-talk problem is eliminated by design. This is done by imposing a fixed pattern of wires on the IC die, on all metal layers. In particular, this repeating pattern, henceforth referred to as the *Dense Wiring Fabric* (*DWF*) is $\cdots$ *VSGSVSGS* $\cdots$, where *V* represents a *VDD* wire, *G* represents a *GND* wire, and *S* represents a signal wire. Wider wires can be implemented by discretely widening wires, in steps of $2 \cdot P$, where P is the wiring pitch. This ensures that adjacent signal wires are always capacitively shielded from each other. Metal wires on any layer run perpendicular to those on layers above and below it. This layout arrangement is shown in Figure 1.

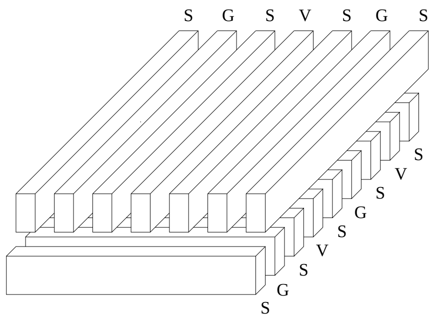

Figure 1: Arrangement of conductors as in [2].

As reported in [2], the use of this fabric has some compelling advantages. First, with this choice of layout fabric, the cross-coupling capacitance between signal wires drops by one to two orders of magnitude, thereby all but eliminating the delay variation and signal integrity problems due to cross-talk[1]. It was experimentally shown in [2] that the delay variation due to cross-talk drops from 2.47:1 to 1.02:1, if the fabric is utilized. This experiment simulates wires of length 200μm, driven by 10$\times$ minimum inverters implemented in a 0.1μm process technology. At the same time, the total wire capacitance increases by a mere 5% compared to the traditional routing style. Also, on-chip signal inductance, which is increasingly becoming a concern for high-speed designs, drops by 35%

[1]Merely removing the VDD and GND wires in the fabric (i.e. routing signal wires at twice minimum pitch) was shown to result in a 3$\times$ reduction in cross-coupling capacitance.

as well, since the current return path for any signal wire is always adjacent to it. The uniformity of inductive and capacitive parasitics which results from the regularity of the DWF is a feature that CAD tools can exploit [3]. Also, by suitably introducing vias whenever *VDD* or *GND* wires intersect on adjacent metal layers, a power and ground distribution network of low and highly uniform resistance is created. The DWF also results in tighter tolerances on the inter-layer dielectric thicknesses due to the fact that metal is maximally gridded all over the IC die. This in turn results in a tighter control on inter-layer wiring capacitances. Finally, the DWF enables us to easily generate a low-skew global clocking network due to the low and predictable parasitics. For a detailed quantification of these benefits, the reader is referred to [2].

The main disadvantage of the scheme of [2] was an increased area requirement compared to the standard cell methodology. Over a series of examples, an area increase of about 65% was reported.

This paper introduces a new design flow which retains the best features of the scheme of [2], with an extremely low area penalty, and a significant delay improvement compared to a standard cell implementation. In our scheme, a logic network is implemented as a network of *Programmable Logic Arrays* (PLAs). The routing area between PLAs utilizes the DWF pattern. We choose a layout implementation of PLAs which is naturally cross-talk immune, and extremely dense. We show that the delay and area of a single PLA are about 50% compared to that of a standard cell based layout. We introduce algorithms to decompose a logic netlist into a network of PLAs in this design style, such that each of the resulting PLAs has a bounded width and height. The output of this synthesis program is then placed and routed, using between 2 and 6 metal layers for routing. We utilized commercial placement and routing tools for the experiments that utilized greater than 3 routing layers. For a series of examples, the average area penalty using this PLA implementation style is shown to be about 3% compared to the standard cell approach. This is in spite of the fact that the DWF is used in the routing area between PLAs. The DWF is not used in the standard cell approach. For the same examples, the timing of our approach was on average 15% better than the standard cell approach.

With a network of PLAs, there is a more direct relationship between the cost function being optimized for during synthesis, and the actual PLA implementation, since there is no intervening technology mapping step. As a result, multi-level logic synthesis is tightly coupled with logic implementation in our design flow.

PLAs have recently experienced a renewed interest as a logic implementation style for high-performance designs. The IBM Gigahertz processor [4] utilized PLAs to implement control logic. The stated reasons for this choice were high speed and the ability to quickly implement and modify the design. We note that the IBM design did not utilize a network of PLAs as we are proposing; rather, single PLAs were used.

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

2 Our Approach

For this paper, we assume a 0.1 μm processing technology, with copper interconnect, and a low-K dielectric. We based the experimental results in this paper on the "strawman" process technology reported in [2].

In our scheme, we implement the circuit as a network of PLAs. Each PLA is a multi-output structure, laid out in a crosstalk-immune manner. Each PLA implements its logic functionality with high density and speed, as we will show Section 2.2. The routing region between PLAs is organized using the DWF, giving rise to highly predictable, crosstalk-immune routes. Metal layers that are not utilized in the layout of the PLA are gridded maximally throughout the die, using the technique of [2]. Also, intersecting *VDD* or *GND* wires on adjacent layers are connected by vias. This gives rise to a highly efficient power and ground distribution network throughout the die. When PLAs are placed, local breaks occur in the power and ground gridding structure of Metal1 and Metal2. This condition was simulated, and determined to cause a negligible change in the power and ground resistance, because the contribution of higher metal layers to the resistance of the power and ground network far outweighs that of the lower metal layers.

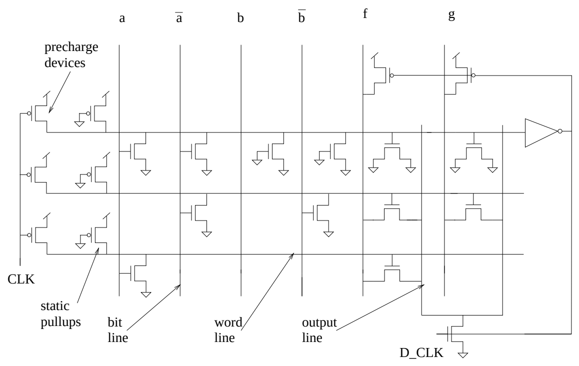

Figure 2: Schematic view of the PLA core

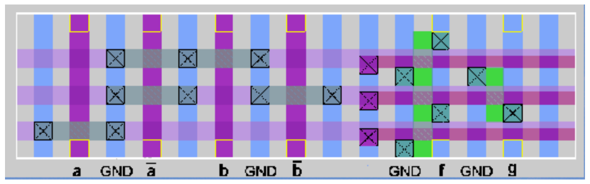

Figure 3: Layout of the PLA core

The remainder of this section is organized as follows. Section 2.1 describes the structure of the PLAs used in our design style, while the results of characterization experiments for individual PLAs are reported in Section 2.2. In Section 2.3 we discuss the construction of a network of PLAs. Our synthesis algorithm, which decomposes a design into a network of PLAs, is described in Section 2.4. Finally, Section 2.5 describes the placement and routing flow we used.

2.1 PLAs in DSM VLSI Design

Consider a PLA consisting of n input variables $x_1, x_2, \cdots, x_n$, and m output variables $y_1, y_2, \cdots, y_m$. 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 $c_i = l_i^1 \cdot l_i^2 \cdots l_i^{r_i}$. We consider PLAs which are of the *NOR-NOR* form. This means that we actually implement f as

$$\overline{f} = \overline{\sum_{i=1}^{k}(c_i)} = \overline{\sum_{i=1}^{k}(\overline{\overline{c_i}})} = \overline{\sum_{i=1}^{k}(\overline{\overline{l_i^1} + \overline{l_i^2} + \cdots + \overline{l_i^{r_i}}})} \qquad (1)$$

The PLA output $\overline{f}$ 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*.

We use a pre-charged NOR-NOR style of PLAs in our design. The schematic view of the PLA core is shown in Figure 2. Several observations can be made from this figure:

- In a pre-charged NOR-NOR PLA, each word-line of the PLA switches from high to low at the end of any computation, if it switches at all. As a result, there is no delay deterioration effect due to crosstalk with neighboring word-lines.
- However, there is a possibility that two "aggressor" word-lines on either side of a "victim" word-line may switch low during the evaluate phase of the clock, while the victim attempts to stay pre-charged. In this situation, it is possible that the switching of the aggressor word-lines will cause the victim word-line to pull low. We simulated this for the relevant sizes of PLAs, and determined that a long channel (i.e. "weak") static pull-up device suffices to avoid this situation. Hence word lines may be safely routed at minimum pitch. These static pull-up devices are highly resistive and their introduction increases the power consumption of a PLA by less than 10%.
- The PLAs are implemented using metal layers 1 and 2. It could be argued that a bus on metal layer 3, routed just above a PLA, could be a source of cross-talk, with the pre-charged word-lines (on metal layer 2) as the victims. This is not a problem in practice, because the static pull-up devices on the word-lines ensure that this cross-talk effect is negligible.
- In the vertical direction, we shield an input and its complement by a GND wire, which is required by the devices in the AND plane anyway.
- One maximally loaded[2] word-line is designed to switch low in the evaluate phase of every clock. It effectively generates a delayed clock, D_CLK, which delays the evaluation of the other word-lines until they have switched to their final values.

  In general, the delay of this word-line may be much higher than the slowest of the remaining word-lines. By suitably selecting the loads on this word-line, the overall delay and power consumption of the PLAs could be improved. This is not currently implemented, and will be addressed in the future.
- Each bit-line is pre-charged low in the pre-charge phase. The corresponding devices are not shown in in Figure 2.

By using a pre-charged NOR-NOR PLA as the layout building block in our methodology, we incur no extra area penalty, either in the horizontal or vertical direction. At the same time, the PLA structure is crosstalk immune, which makes it an ideal choice.

Figure 3 shows the layout of the PLA core (implemented using two metal layers). The horizontal word lines are implemented in METAL2. The width of the PLA core is $4 \cdot n + 2 \cdot m$ tracks, since the each input requires 4 vertical tracks, and each output requires 2.

We implement the input and output drivers outside the footprint of the PLA[3] (i.e. in the routing channel). This gives rise to a much lower area overhead for our PLAs, and also allows us significant flexibility in sizing the drivers. The only effect it has on the routing channel is the introduction of one via per driver. Figure 7 illustrates the location of output drivers relative to the PLA core. We were able to complete the layout of all control signals with an additional cost of only 4 horizontal tracks. 4 extra vertical tracks are required per PLA for the implementation of the pre-charge and static pull-up devices. Figure 4 shows the relative orientation of pre-charge devices, muxes and drivers in the layout of each PLA. In all the simulations we report in this paper, these overheads are accounted for. Also, in the electrical simulation of the PLA characteristics, the transistor sizes utilized are as shown in Figure 3.

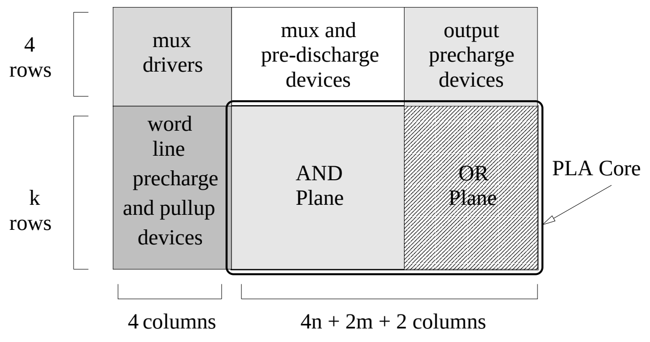

Figure 4: Layout Floorplan of PLA

Due to the regularity of the PLA structure, a simple delay formula can be used to estimate the worst-case delay of a PLA. As we will see, this formula is utilized in the synthesis step.

2.2 PLA Characterization

Figure 5 shows the pattern of wires occurring within the core of our PLAs. Capacitive parasitics for the wires within our PLA core were extracted using a 3-dimensional parasitic extractor called *Space3D* [5]. The input to *Space3D* is a 3-dimensional circuit layout (dimensions are as in the 0.1μm "strawman" process reported in [2]), and the output is the value of the capacitive parasitics between different features of that layout. *Space3D* uses a boundary element method to compute interconnect capacitances.

The results of these extractions are shown in Table 1. In this table $C^1_{i,i}$ refers to the capacitance between two metal conductors on the same level i, which are separated by minimum spacing. $C^2_{i,i}$ refers to the capacitance between two such conductors separated by twice the minimum spacing. $C_{i,0}$ is the capacitance of a conductor to the substrate, and $C_{i,i+1}$ is the capacitance of a conductor on level i to other conductors on level $i+1$.

Layer	$C^1_{i,i}$	$C^2_{i,i}$	$C_{i,0}$	$C_{i,i+1}$
1	47.17	14.57	13.72	15.78
2	48.37	-	0.77	5.96

Table 1: 3-Dimensional Parasitics for Figure 5 (10^{-18}F per μ)

To compare a *single* PLA implemented in our layout style against the standard cell layout style, we took four examples and implemented them in both styles. Delay and power results were obtained utilizing SPICE [6]. The area comparison was done using actual layout for both styles (using two routing layers).

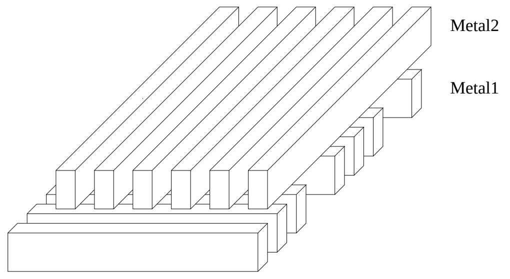

Figure 5: Arrangement of conductors in the PLA core

[2]This word line has the maximum number of diffusion and gate loads possible in the PLA. Figure 2 illustrates this schematically

[3]These devices are not shown in Figure 2 and Figure 3.

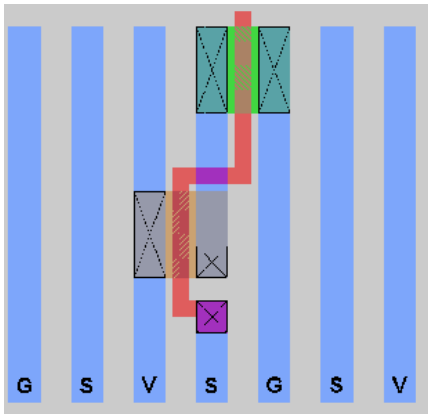

Figure 6: Buffer insertion in the DWF

For the standard cell style of layout, we performed technology-independent optimizations in SIS [7], after which we mapped the circuit using a library of 11 standard cells. These cells were optimized for low power consumption. We use the *wolfe* tool within *OCT* [8] to do the placement and routing. *Wolfe* in turn calls *TimberWolfSC-4.2* [9] to do the placement and global routing, and YACR [10] to do the detailed routing.

For the PLA layout style, we flatten the examples, and then generate the *magic* [11] layout for the resulting PLA using a *perl* script. To compute the delay, we simulate a maximally loaded output line pulled down by a single output pull-down device. Parasitics from Table 1 were utilized to model the interconnect within a PLA.

The results of this comparison are listed in Table 2. For each layout style, D refers to the delay in picoseconds, A refers to the layout area of the resulting implementation in square grids, and P refers to the power consumption. Note that for the standard cell layout style, D and P values are the maximum values obtained after simulating about 20 input vectors. Also, we don't account for wire capacitances in the standard cell implementation, which would only increase its delay and power. In the case of the PLA layout style, however, the D and P are worst-case values. Despite this, the PLA layout style shows impressive improvements over the standard cell layout style. The PLA layout requires between 0.33 and 0.81 times the area of the standard cell layout. The average area requirement of the PLAs is 0.46 times that of the standard cell layout style, which is an impressive reduction. The delay value for the PLA is on average 0.48 times that of the standard cell implementation, which is partially attributed to the fact that the PLA is dynamic, while the standard cell implementation is static.

The power consumption of the PLA is usually larger than that of the standard cell implementation, mainly because the bit-line capacitances are charged and discharged on every cycle. PLAs with large values of k exhibit a much higher power consumption. For this reason, we use a value of $k \leq 25$ while decomposing a circuit into a network of PLAs. This power consumption can also be curbed by gating the clocks of PLAs which are not utilized during a given computation. Also, power and delay can be effectively traded off in our PLA design style (results of this experiment are excluded for brevity). Finally alternative fabrics utilizing unate PLAs can be used to reduce the power consumption.

To estimate the effect of crosstalk between literals of neighboring variables in the PLA, we simulated a PLA with $k = 40$. Let l_i be a literal of variable x_i, and l_{i+1} be a literal of variable x_{i+1}. Assume l_i and l_{i+1} are separated by a blank track. In this situation, there is a 1:1.0156 delay variation for l_i, depending on whether l_{i+1} switches in the opposite or similar direction. This delay variation is small enough to be disregarded.

The reason why PLAs result in very favorable area and delay characteristics compared to a standard cell layout are the following:

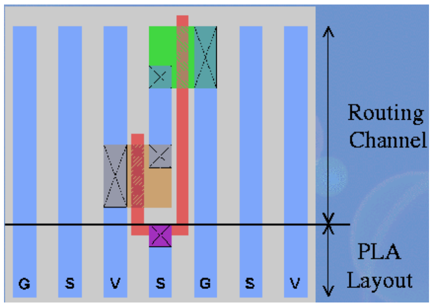

Figure 7: PLA output driver layout

• First, PLAs implement their logic function in 2-level form, which results in superior delay characteristics as long as k is bounded. On the other hand, in a standard cell implementation, considerable delay is incurred in traversing the different levels (i.e. gates) of the design.
• In the PLA implementation scheme, local wiring is collapsed into a compact 2-level core, which is naturally crosstalk-immune. Hence local wiring delays are reduced.
• In DSM processes, it is often stated that a large part of a signal's delay is attributable to global wiring. Methods to tackle this include up-sizing of drivers, and buffer insertion.
— Sizing of PLA output drivers is easily done with no PLA area penalty. This is because output drivers are placed in the routing area, as illustrated in Figure 7.
— A signal can be buffered in the DWF with no area penalty, since the VDD and GND signals required to construct a buffer are available on either side of the signal. Figure 6 illustrates the insertion of a buffer in the routing region (which utilizes the DWF).
• Devices in the PLA core are minimum-sized, giving rise to extremely compact layouts. Such is not the case for standard cell layouts.
• In our PLA core, NMOS devices are used exclusively. As a result, devices can be placed extremely close together. However, in a standard cell layout, both PMOS and NMOS devices are present in each cell, and the PMOS-to-NMOS diffusion spacing requirement results in a loss of layout density.

The fact that the IBM Gigahertz processor[4] utilizes two-level PLAs to implement control logic is further evidence that PLAs are an effective logic implementation style for high-performance designs.

2.3 Network of PLAs

Having discussed the characteristics of a single PLA, we now discuss how a *network* of PLAs is constructed.

Since the PLAs in our design are pre-charged, we need to ensure that the inputs to any PLA settle before its evaluation begins. A network of PLAs is *correct* iff each PLA in this network satisfies this constraint.

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

- $V = \{v_1, v_2, \cdots, v_r\}$*, 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.

It is easy to see that if the PLA dependency graph has a cycle, then the corresponding network of PLAs is not correct. For a correct network, we need to ensure that the PLA dependency graph is acyclic, and also that the evaluation of PLA p begins only after the evaluation of the slowest PLA q, such that $(q, p) \in E$. This suggests a *self-timed* [12] design style. For this reason, each PLA q generates an additional *completion* signal,

				PLA implementation			Standard Cell			Ratios		
Example	*n*	*m*	*k*	*D*	*A*	*P*	*D*	*A*	*P*	*D*	*A*	*P*
cmb	16	4	15	160.3	53.3k	5.32	300	159.8k	6.15	0.534	0.334	0.864
cu	14	11	19	189.1	69.5k	4.84	420	186.5k	4.24	0.450	0.373	1.140
x2	10	7	17	164.8	45.3k	4.23	290	136.8k	1.82	0.568	0.331	2.324
z4ml	7	4	59	200.5	95.2k	10.28	575	118.3k	3.17	0.349	0.805	3.243

Table 2: Comparison of Standard Cell and PLA implementation styles

which gates the evaluation clock of the appropriate PLA *p*. Given the regularity of the PLA structure, the worst case delay of each PLA is easily known, and corresponds to the delay of a single output device discharging a maximally loaded output line. This completion signal is generated with an overhead of one additional word line and one additional output line in each PLA. Additional timing margin is obtained by downsizing the output driver of the completion signal.

2.4 Synthesis of a Network of PLAs

Problem Definition: Given an arbitrary combinational logic circuit *C*, find a decomposition of *C* into a network *N* of PLAs, subject to :
• the network *N* is correct.
• each PLA has a height no larger than a specified maximum, H.
• each PLA has a width no larger than a specified maximum, W.

Algorithm 1 outlines our decomposition strategy. We begin by performing technology independent optimizations on *C* . Next, we decompose *C* into a network C^* of nodes with at most *p* inputs. Now C^* is sorted in depth-first manner. The resulting array of nodes is sorted in *levelization*[4] order, and placed into an array *L*.

Now we greedily construct the logic in each PLA, by successively grouping nodes from *L* such that the resulting PLA implementation of the grouped nodes N^* does not violate the constraints of PLA width and height. This check is performed in the *check_PLA* routine, which first flattens N^* into a two-level form, *P*. It then calls *espresso* [13] on the result to minimize the number of cubes in *P*. Next, *check_PLA* calls a *PLA folding* routine which attempts to fold the inputs of *P* so as to implement a more complex PLA in the same area. Finally *check_PLA* ensures that the final PLA, after folding and simplification using *espresso*, satisfies the maximum width and height constraints respectively. If so, we attempt to include another node into N^*, otherwise we append the last PLA satisfying the height and width constraints to the result.

The *get_next_element* routine returns nodes in the fanout of the nodes in N^* (in an attempt to reduce the wiring between PLAs), provided that the inclusion of such a node into N^* would not result in a cyclic PLA dependency graph. If such nodes are not available, the first un-mapped node from *L* is returned.

Note that this algorithm does not attempt to ensure that the maximum delay between any PI-PO pair is bounded. As a result, it sometimes returns a network with delays larger than the corresponding standard cell implementation. However, on average, the delay of the network it returns is much better than a standard cell implementation (see Section 3).

We implemented our algorithm in SIS [7], and performed extensive benchmarking of the PLA network decomposition code. We found that a good choice of parameters was $p = 5$, W = 50 to 70, and H = 15 to 25. Increasing H beyond 30 did not usually result in a reduction in the total number of PLAs generated. Folding the PLAs resulted in a decrease of between 20% and 50% in the total number of PLAs required for a network. Our PLA Folding algorithm folds only PLA inputs. It constructs a list of candidates to fold, and then assigns a heuristic figure of merit to each candidate. This figure of merit awards folds that allow subsequent folds to proceed without hindrance.

[4]Primary inputs are assigned a level 0, and other nodes are assigned a level which is one larger than the maximum level of all their fanins

Algorithm 1 Decomposition of a Circuit into a Network of PLAs

```
C = optimize_network(C)
C* = decompose_network(C, p)
L = dfs_and_levelize_nodes(C*)
N* = 0
RESULT = 0
while get_next_element(L) != NIL do
  N* = N* ∪ get_next_element(L)
  P = make_PLA(N*)
  if check_PLA(P, W, H) then
    continue
  else
    Q = remove_last_element(N*)
    RESULT = RESULT ∪ N*
    N* = Q
  end if
end while
```

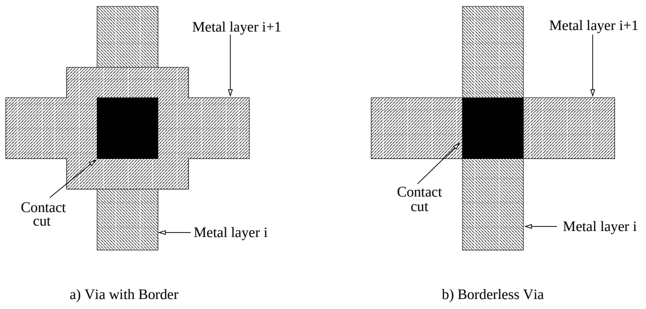

Figure 8: Vias with and without Borders

We verified functional correctness of the resulting network of PLAs, at the end of the decomposition step.

2.5 Placing and Routing a Network of PLAs

Placement and routing is performed using between 2 and 6 routing layers. For 2 routing layers, we utilize tools that are available in the public domain. For the other experiments, we use commercially available tools.

Routing within the constraints of the DWF is easily achieved by modifying the routing pitch value of the router to be twice that used for non-DWF routing.

Two Routing Layers: The synthesized network of PLAs was placed using a simulated annealing-based FPGA placement tool called *VPR* [14]. Since the PLAs in our design have approximately the same size, the problem of placing PLAs is similar to the FPGA placement problem. Hence *VPR* is a good choice. The placed result is routed using *wolfe*[5].

We also assume that all vias have *borders*. Figure 8 illustrates a borderless via in comparison to a via with borders.

The standard cell design flow was described in Section 2.2

More than Two Routing Layers:

The synthesized network of PLAs was placed and routed using the SEDSM-5.1 [15] toolset from CADENCE. Placement was performed using the QPLACE tool within SEDSM-5.1. Routing was performed using the WARP area router, which can use up to 6 metal layers for routing. Within this flow, we use between 3 and 6 metal layers to route the de-

[5]Global routing is performed using TimberWolfSC-4.0 [9], while detailed routing is performed using YACR [10]

signs. We assume that all vias are *borderless*. Modern fabrication process usually utilize borderless vias.

The standard cells used for these experiments were optimized to work with the CADENCE tools, and were provided with the SEDSM-5.1 package. There were 11 cells in the library we used, and these cells were logically equivalent to those used in the experiments with 2 routing layers. Each pin of these cells had several possible contact locations available, to enable easier routing. Also, rows of standard cells are flipped and abutted so that their power and ground buses were shared, resulting in reduced circuit area.

3 Experimental Results

We based the experimental results in this paper on the "strawman" process technology reported in [2]. This in turn used [16] and the Sematech process technology predictions [17] to come up with processing parameters for several upcoming generations. For this paper, we assume a 0.1 μm processing technology, with copper interconnect, and a low-K dielectric.

Tables 3 and 4 describe results using two metal layers to perform routing. The area and delay characteristics of our network of PLAs based design methodology are shown in Table 3. This table describes the results for a series of benchmarks, comparing the area of a standard cell implementation (column 2), and our approach (column 3). All areas are in units of square microns. Column 4 reports the ratio of column 3 to column 2. Two layers of metal are utilized to achieve the routing.

Also reported is the total delay (in picoseconds) of the standard cell implementation (column 5), and that of our approach (column 6). The ratio of these two delays is shown in column 7.

Delays for the standard cell implementation were obtained by running the *exact timing analysis* [18] technique on the mapped netlist. This technique logically eliminates false paths, so all delays are sensitizable. For our approach, we computed the worst case delays of each PLA in the network using Spice [6], as described in Section 2.2. Then we found the worst case delay path from any primary input to any primary output in the PLA network by traversing the network of PLAs in DFS order. It is possible that a delay quoted for our scheme is not sensitizable, and that the real delay is lower.

We note that over all these examples, the area overhead of our method was a mere 2.4%. This is in spite of the fact that the DWF is used in the routing area between PLAs, but not used in the standard cell case. Also, the delay of our approach is approximately 15% better than that of a standard cell implementation. Some examples result in much higher delays for the network of PLAs style. This is attributed to the fact that our decomposition routine does not attempt to control the delay of the network, but rather attempts to reduce wiring between PLAs.

Table 4 reports the area and timing results for the condition in which the routing of the network of PLAs is performed at minimum pitch (utilizing 2 metal layers for routing). In case cross-talk was not a consideration and wiring did not need to be performed using the DWF, then this would be the area and timing comparison of the Fabric3 methodology against the standard cell based methodology.

In Table 4 we notice that the average area requirement of the network of PLAs is significantly reduced compared to the standard cell based implementation. This is as expected, since the network of PLAs methodology implements logic in a very dense fashion, as detailed in Section 2.2. Also, the timing of the network of PLA implementation is on average about 16% better than that of the standard cell implementation. This improvement is very similar to that reported Table 3.

Example C3540 exhibits a large area penalty, and we conjecture that this is due to the fact that it implements an ALU with control. Arithmetic circuits typically do not have good PLA implementations, unless encoding of inputs [19] is performed. We plan to investigate encoding techniques in the future.

	Area			Timing		
Example	Std Cell	Ntk of PLA	Ratio	Std Cell	Ntk of PLA	Ratio
C432	1971.67	2628.89	1.338	2326.4	2237.1	0.962
C499	4901.00	3868.22	0.789	1861.5	1575.8	0.847
C880	4337.67	5295.33	1.221	2007.5	1587.3	0.791
C1355	6346.89	7248.22	1.141	2688.4	1890.7	0.703
C1908	6835.11	10271.44	1.502	2203.9	3235.6	1.468
C2670	20974.78	15604.33	0.744	2388.3	2244.6	0.940
C3540	18101.78	39677.44	2.191	3324.3	4385.5	1.319
alu2	4093.56	2929.33	0.717	2528.3	1758.2	0.695
apex6	13613.89	8656.56	0.636	1332.4	1011.2	0.759
count	1220.56	694.78	0.564	2029.7	568.0	0.280
decod	375.56	225.33	0.567	330.4	184.3	0.558
pcle	507.00	469.44	0.925	1285.7	334.5	0.260
rot	13651.44	11454.44	0.838	2256.1	2110.8	0.936
pair	36147.22	41761.78	1.166	1951.9	2660.2	1.363
AVERAGE			1.024			0.848

Table 3: Layout Area and Timing (PLAs routed using DWF)

	Area			Timing		
Example	Std Cell	Ntk of PLA	Ratio	Std Cell	Ntk of PLA	Ratio
C432	1971.67	1765.11	0.895	2326.4	1791.4	0.770
C499	4901.00	2873.00	0.586	1861.5	1520.1	0.817
C880	4337.67	4037.22	0.931	2007.5	1761.4	0.877
C1355	6346.89	6853.89	1.080	2688.4	1919.1	0.714
C1908	6835.11	7548.67	1.104	2203.9	2785.3	1.264
C2670	20974.78	11923.89	0.568	2388.3	2361.7	0.989
C3540	18101.78	30326.11	1.675	3324.3	4529.8	1.363
alu2	4093.56	2234.56	0.546	2528.3	1235.3	0.489
apex6	13613.89	5933.78	0.436	1332.4	1236.9	0.928
count	1220.56	563.33	0.462	2029.7	568.0	0.280
decod	375.56	169.00	0.450	330.4	184.3	0.558
pcle	507.00	300.44	0.593	1285.7	319.0	0.248
rot	13651.44	8506.33	0.623	2256.1	2093.2	0.928
pair	36147.22	24918.11	0.689	1951.9	2900.2	1.486
AVERAGE			0.760			0.837

Table 4: Layout Area and Timing (PLAs routed at min. Pitch)

Layers	Penalty (DWF)	Penalty (no DWF)
2	1.024	0.760
3	1.081	0.787
4	1.018	0.794
5	1.015	0.801
6	1.005	0.811

Table 5: Average Layout Area Penalty using Multiple Routing Layers

Table 5 reports the average area penalty using our approach, as a function of the number of routing layers used. The results represent the average area penalty for the 14 benchmark circuits. The penalties reported in column 2 correspond to the case where the routing area between PLAs utilizes the DWF. In column 3, no DWF is utilized. The results for the row representing 2 routing layers correspond to the average area penalties reported in Tables 3 and 4. For all these cases, the average timing improvement was about 15%.

These results show that our approach works well regardless of the number of metal layers utilized to perform routing.

For the best standard cell implementation of C3540 across the experiments involving 2, 3, 4, 5, and 6 routing layers, the semi-perimeter of the layout was 240μm. For this length, the delay variation due to cross-talk was simulated using SPICE[6], and determined to be 3.45:1. The corresponding delay variation for the DWF was 1.07:1. This assumed that drivers are 3$\times$ minimum, implemented in a 0.1μm process. Even if the drivers were sized up[6] in a standard cell based design, the delay variation still remains significant. For example, using 60$\times$ minimum drivers still results in a delay variation of 1.22:1.

[6]Using a larger driver results in lower delay variation due to cross-talk

4 Conclusions and Future Work

We have presented a design methodology for use in DSM IC design. In this methodology, we decompose the original circuit into a network of PLAs of bounded width and height, which we place and route within a regular layout fabric as in [2]. The *advantages* of our method are as follows:

• High speed. Each PLA is shown to be on average 2.1× faster than its corresponding standard cell based circuit implementation. Also, the network of PLAs is about 15% faster than the standard cell implementation of the same netlist.

• Low area overhead. Over a series of examples, we show that our scheme has an area overhead of about 3%. This is in spite of the fact that the DWF is used in the routing area between PLAs, but not used in the standard cell case. Each individual PLA is shown to be 2.17× smaller than its corresponding standard cell based circuit implementation.

• Area overheads and timing improvements are shown to be largely independent of the number of routing layers utilized to route the design.

• Elimination of cross-talk and signal integrity problems that are common in DSM designs. The resulting implementation is highly reliable.

• Power and ground routing is done implicitly, and not in a separate step in the design methodology. Power and ground resistances are very low and vary much less compared to the power and ground distribution used in the standard cell methodology.

• Variations in delay of a signal wire due to switching activity on its neighboring signal wires is less than 1.02:1, compared to a 2.47:1 variation using conventional layout techniques (as described in [2]).

• Smaller and uniform inductances for all wires on the chip, compared to larger and unpredictable values using the existing layout styles.

• Rapid design turn-around time due to highly regular structures and regular parasitics.

• With a network of PLAs, there is a direct relationship between the cost function being optimized for during synthesis, and the PLA implementation, since there is no intervening technology mapping step. This helps ensure that benefits due to synthesis optimizations are not lost in the implementation step.

• Techniques for wire removal using multi-valued *Sets of Pairs of Functions to be Distinguished* (SPFDs) have been implemented [20]. Binary valued SPFDs were first introduced in [21], and adapted for use in logic networks in [22]. Multi-valued SPFDs can be used to perform wire removal in the PLA network. Preliminary wire removal results using binary SPFDs show up to 20% reduction in circuit area.

We believe that this technique will significantly simplify the design of chips with minimum feature sizes in the DSM range.

In the future, alternative unate fabrics will be pursued as a means to reduce the power consumption of the PLAs. Inductive and capacitive characteristics of such fabrics are being studied. We also plan to experiment with the idea of relaxing the fabric restriction on lower metal layers, while ensuring that routes on these layers are short. Efficient methods to decompose a logic netlist into a network of PLAs will also be pursued, such that the delay of the network of PLAs, as well as the wiring between PLAs is minimized. In general, PLAs may not be suitable for designs that have inherent multi-level logic structures. Arithmetic circuits are examples. To address this, [19] showed that encoding of PLA inputs was an effective technique to reduce the number of terms in PLAs, especially for arithmetic circuits. This will be investigated as well.

Currently, our approach addresses combinational circuits. It can be extended to handle sequential circuits as well, using an approach similar to [23, 24].

5 Acknowledgements

This research was supported partially by the SRC, the GSRC/Marco center at Berkeley, and the California micro program with our industrial sponsors, Motorola, Fujitsu, Synopsys, and Cadence.

References

[1] J. Grodstein, "Member, DEC Alpha microprocessor design team." Personal communication, 1998.

[2] S. Khatri, A. Mehrotra, R. Brayton, A. Sangiovanni-Vincentelli, and R. Otten, "A novel VLSI layout fabric for deep sub-micron applications," in *Proceedings of the Design Automation Conference*, (New Orleans), June 1999.

[3] R. Otten and R. Brayton, "Planning for performance," in *Proceedings of the Design Automation Conference*, pp. 122–127, Jun 1998.

[4] S. Posluszny, N. Aoki, D. Boerstler, J. Burns, S. Dhong, U. Ghoshal, P. Hofstee, D. LaPotin, K. Lee, D. Meltzer, H. Ngo, K. Nowka, J. Silberman, O. Takahashi, and I. Vo, "Design methodology for a 1.0 ghz microprocessor," in *Proceedings of the International Conference on Computer Design (ICCD)*, pp. 17–23, Oct 1998.

[5] "Physical Design Modelling and Verification Project (SPACE Project)." `http://cas.et.tudelft.nl/research/space/html`.

[6] L. Nagel, "Spice: A computer program to simulate computer circuits," in *University of California, Berkeley UCB/ERL Memo M520*, May 1995.

[7] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis," Tech. Rep. UCB/ERL M92/41, Electronics Research Laboratory, Univ. of California, Berkeley, CA 94720, May 1992.

[8] A. Casotto, ed., *Octtools-5.1 Manuals*, (Electronics Research Laboratory, College of Engineering, University of California, Berkeley, CA 94720), University of California at Berkeley, Sept. 1991.

[9] C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf Placement and Routing Package," *IEEE Journal of Solid-State Circuits*, 1985.

[10] J. Reed, M. Santomauro, and A. Sangiovanni-Vincentelli, "A new gridless channel router: Yet another channel router the second (YACR-II)," in *Digest of Technical Papers International Conference on Computer-Aided Design*, 1984.

[11] G. T. Hamachi, R. N. Mayo, and J. K. Ousterhout, "Magic: A VLSI Layout system," in *21st Design Automation Conference Proceedings*, 1984.

[12] J. Rabaey, *Digital Integrated Circuits: A Design Perspective*. Prentice Hall Electronics and VLSI Series, Prentice Hall, 1996.

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

[14] V. Betz and J. Rose, "VPR: A new packing, placement and routing tool for FPGA research," in *Proceedings of the International Workshop on Field Programmable Logic and Applications*, 1997.

[15] Cadence Design Systems, Inc., 555 River Oaks Parkway, San Jose, CA 95134, USA, *Envisia Silicon Ensemble Place-and-route Reference*, Nov 1999.

[16] "The National Tecnology Roadmap for Semiconductors." `http://notes.sematech.org/97melec.htm`, 1997.

[17] P. D. Fisher, "Clock Cycle Estimation for Future Microprocessor Generations," tech. rep., SEMATECH, 1997.

[18] P. McGeer, A. Saldanha, R. Brayton, and A. Sangiovanni-Vincentelli, *Logic Synthesis and Optimization*, ch. Delay Models and Exact Timing Analysis, pp. 167–189. Kluwer Academic Publishers, 1993.

[19] M. S. Schmookler, "Design of large ALUs using multiple PLA macros," *IBM Journal of Research and Development*, vol. 24, pp. 2–14, Jan 1980.

[20] S. Sinha, S. Khatri, R. Brayton, and A. Sangiovanni-Vincentelli, "Binary and multi-valued SPFD-based wire removal in PLA networks," in *Proceedings of the ICCD*, Sep 2000. To Appear.

[21] S. Yamashita, H. Sawada, and A. Nagoya, "A new method to express functional permissibilities for LUT based FPGAs and its applications," in *Proceedings of the International Conference on Computer-Aided Design*, pp. 254–61, Nov 1996.

[22] R. Brayton, "Understanding SPFDs: A new method for specifying flexibility," in *Workshop Notes, International Workshop on Logic Synthesis*, (Tahoe City, CA), May 1997.

[23] S. Patil and T. Welch, "A programmable logic approach for VLSI," *IEEE Transactions on Computers*, vol. c-28, pp. 594–601, Sep 1979.

[24] K. Smith, T. Carter, and C. Hunt, "Structured logic design of integrated circuits using the storage/logic array (SLA)," *IEEE Transactions on Electron Devices*, vol. ED-29, pp. 765–76, Apr 1982.