Exploring Logic Block Granularity for Regular Fabrics

A. Koorapaty, V. Kheterpal, P. Gopalakrishnan, M. Fu, L. Pileggi
{aneeshk, vkheterp, pgopalak, mfu, pileggi}@ece.cmu.edu
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213, USA

Abstract

Driven by the economics of design and manufacturing nanoscale integrated circuits, an emphasis is being placed on developing new, regular logic fabrics that leverage the regularity and programmability of FPGAs, yet deliver a level of performance and density close to ASICs. One example of such a fabric is a Via-Patterned Gate Array (VPGA) [9], which employs ASIC style global routing on top of an array of patternable logic blocks (PLBs). Previous work [8], [6], [10] showed that by employing even limited heterogeneity for the VPGA logic blocks, namely combining a 3-LUT with two 3-input Nand gates, one can achieve performance comparable to that provided by standard cells. Since the area cost for such heterogenity is far less for a VPGA than for SRAM programmed fabrics such as FPGAs, we can explore new configurations of via-configurable logic blocks that offer greater heterogenity and granularity to achieve even higher performance. In this paper, we present a new, more granular, via-patterned heterogeneous logic block architecture and compare it to a less granular LUT-based heterogeneous PLB. Our results show higher performance and more effective packing of the logic functions due to increased granularity.

1. Introduction

Traditionally, digital systems have been implemented either as Application Specific Integrated Circuits (ASICs) or using standard parts such as Field Programmable Gate Arrays (FPGAs). ASICs consist of pre-designed logic cells and up to seven layers or more of metal wiring. The high degree of flexibility in placement and routing of the cells necessitates unique, customized masks for all fabrication layers. With shrinking feature sizes and increasing design complexity, mask costs and design costs for re-spins are becoming prohibitively expensive. As a result, FPGAs are becoming increasingly attractive. Unlike ASICs, FPGAs amortize design costs across several applications and enhance manufacturability via greater layout regularity. However, FPGAs can be three times slower and require ten times more die area than an equivalent standard cell design.

Driven by the economics of design and manufacturing of deep sub-micron integrated circuits, an emphasis is being placed on developing new regular fabrics that deliver a combined cost and performance advantage. A Via-Patterned Gate Array [9] is one example of such a fabric. Like an FPGA, a VPGA consists of an array of PLBs. However, there are two key differences: First, the customization of the logic is done by the placement or removal of vias at the potential via locations, as opposed to configuring SRAM bits. Second, global routing is on top of, instead of adjacent to the PLB array, resulting in a significant reduction in die area.

The choice of PLB architecture for the logic array is crucial. This problem has been well studied for FPGAs in [4] and [3]. For VPGAs, previous work in [7] and [6] showed that LUT-mapped designs are dominated by simple logic functions like two and three input AND, NAND, OR, NOR, and AOI (And-Or-Invert) functions, which are not implemented efficiently by LUTs. Based on these results, [7] and [8] explored four heterogeneous logic block architectures with a combination of LUTs, MUXes, and logic gates, and showed that such architectures offer significant performance and density benefits over conventional homogeneous LUT-based PLBs. In particular, [8] explored heterogeneous logic blocks for the VPGA fabric and showed that a heterogeneous PLB with a combination of 3-LUTs and three input Nand gates with programmable inversion (ND3WI gates), offers the best performance and density for both logic and datapath applications. Figure 1 illustrates the PLB proposed in [8] for the VPGA fabric.

The key objective of the VPGA fabric is to leverage the regularity benefits of FPGAs, while delivering a level of performance close to ASICs. In [10], the authors showed that the VPGA fabric with the PLB shown in Figure 1 is quite competitive in performance with standard cell designs. To further improve performance and density, it is necessary to employ logic blocks with higher granularity.

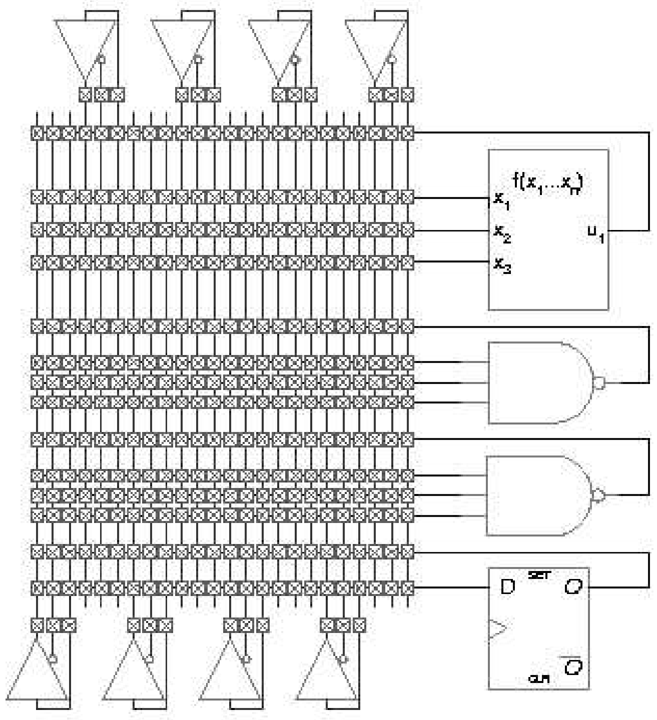


Figure 1. LUT based PLB for VPGA Fabric

Increasing granularity implies greater heterogeneity, which requires more configurability through the local interconnect architecture. Since greater configurability only results in an increase in potential via sites for via-patterned fabrics, the cost of higher granularity is significantly lower for the VPGA fabric, than for SRAM programmed FPGAs.

In this paper, we propose a new, more granular, via-patterned heterogeneous logic block architecture and compare its performance and density to the LUT-based VPGA PLB shown in Figure 1 and equivalent standard cell designs. Most importantly, as part of our design exploration process, we develop the set of simple logic circuit primitives that can be used to construct a high performance via-patternable CLB for almost any datapath or control logic application.

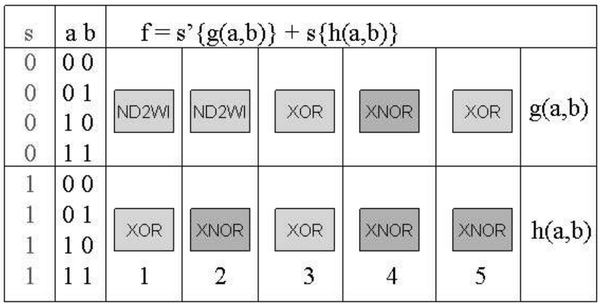
The remainder of this paper is organized as follows. Section 2 presents the new, granular, heterogeneous PLB architecture. Section 3 outlines our experimental methodology and presents results comparing the granular PLB to the LUT based PLB. Section 4 concludes the paper.

2. New, Granular, Via-Patterned PLB for VPGA Fabric

In [8], the authors explored various heterogeneous logic block architectures for the VPGA fabric and selected the PLB shown in Figure 1 with one 3-LUT and two ND3WI gates. Although this PLB achieves good performance and density for both logic and datapath applications, it has two key drawbacks. First, all 3-input functions that cannot be performed by the ND3WI gate must be implemented by the LUT. However, as shown in [10], the VPGA LUT is substantially inferior to an equivalent standard cell in terms of delay, power and area, when configured as a simple logic function. Second, a full adder cannot be implemented by a single PLB. To push the performance of the VPGA fabric further, it is necessary to leverage the benefits of heterogeneity by employing more granular architectures that overcome the performance penalty of the LUT, and enable a full adder to be packed into a single PLB.

2.1 Analysis of 3-input Functions

From the Shannon co-factoring property, it is known that any 3-input function can be written in terms of its cofactors as follows: f(a,b,s) $\rightarrow$ s'*(g(a,b)) + s*(h(a,b)), where s' is the inverted select signal, and the cofactors are g(a,b) and h(a,b). Based on this result, [7] showed that a 2-input MUX driven by two ND2WI gates can implement at least 196 of the 256 3-input functions. The functions that cannot be implemented by this structure, henceforth referred to as an S3 gate, are those that have one or more XOR or XNOR functions as cofactors. Figure 2 illustrates these classes of functions.



s	a b	f = s'{g(a,b)} + s{h(a,b)}					
0 0 0 0	0 0 0 1 1 0 1 1	ND2WI	ND2WI	XOR	XNOR	XOR	g(a,b)
1 1 1 1	0 0 0 1 1 0 1 1	XOR 1	XNOR 2	XOR 3	XNOR 4	XNOR 5	h(a,b)

Figure 2. Categories of S3 Infeasible Functions

Of the five categories of functions shown in Figure 2, the third and fourth simplify to a 2-input XOR and XNOR gate, respectively. Both of these functions can be implemented by a single 2:1 MUX. For the fifth category, we observe that one of the cofactors is the complement of the other. These functions, which correspond to a 3-input XOR or XNOR, can be implemented by two 2:1 MUXes and an inverter. This leaves the first two categories of functions in which one of the cofactors is implementable by a ND2WI gate, and the other is an XOR or XNOR function. Since a 2:1 MUX can implement *all* 2-input functions, including XOR and XNOR, we replace one of the ND2WI gates in the S3 structure with a 2:1 MUX. By adding a programmable inverter on the output of this MUX, we ensure that the modified S3 structure shown in Figure 3 can implement *all* 256 3-input functions.

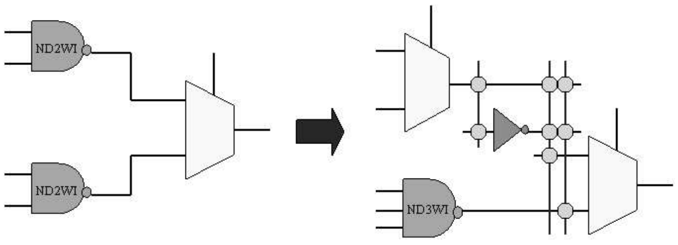


Figure 3. Modified S3 Cell

2.2 Implementing a Full Adder

To implement a full adder, the PLB must be able to generate both the sum and carry functions. To implement the sum function: Sum = (A $\bigoplus$ B $\bigoplus$ C_{in}), we only require the two MUXes in the cell shown in Figure 3. The first MUX implements the function (A $\bigoplus$ B), and the second implements the exclusive-or of C_{in} and the output of the first MUX. As a result, the Nand gate is still available for the carry function.

To take advantage of this, we express the carry function as: C_{out} = P*C_{in} + P'*G, where P = (A $\bigoplus$ B) is called the propogate function, and G = A*B is called the generate function. Since the first MUX already implements the carry propogate function (A $\bigoplus$ B), only one more MUX with P as the select signal is required to implement a full adder in a single PLB. Adding this MUX to the cell in Figure 3 results in the new, granular heterogeneous PLB shown in Figure 4.

This PLB consists of three 2:1 MUXes, one ND3WI gate, and programmable buffers. One of the MUXes is labelled XOA since it is primarily used as an XOR or a ND2WI gate, and because it is sized differently from the other two MUXes to minimize logic delay. Although not shown in Figure 4, the PLB also includes a D Flip Flop, and buffers that ensure that all primary inputs are available in both polarities.

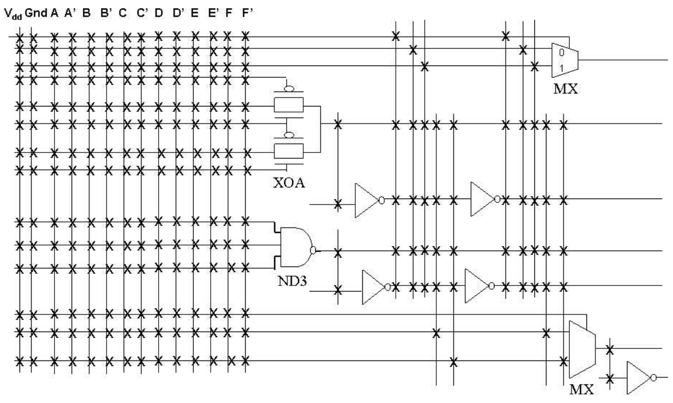


Figure 4. Granular PLB for the VPGA Fabric

2.3 Granularity Trade-offs

Although at first glance it appears that the PLB in Figure 4 is significantly different from the previously selected PLB shown in Figure 1, the new PLB actually contains similar logic elements. The key differences are that the new PLB has only one ND3WI gate instead of two. Also, the 3-LUT in the previous PLB is split into its component MUXes which are re-arranged in a manner that enables the new PLB to access intermediate outputs. Splitting the 3-LUT into three MUXes as shown in Figure 5 increases granularity and flexibility.

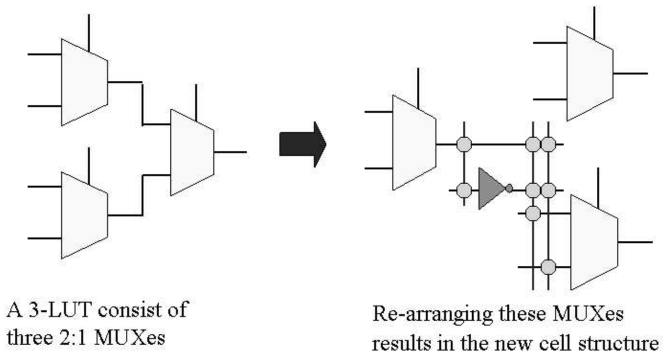

A 3-LUT consist of three 2:1 MUXes
Re-arranging these MUXes results in the new cell structure

Figure 5. Re-arrangement of the three 2:1 MUXes in a 3-LUT

One of the key advantages of higher granularity is that several 3-input functions can be implemented with logic configurations that are faster and denser than a 3-input LUT. For the PLB shown in Figure 4, these logic configurations are:

1. A single 2:1 MUX (MX)
2. A single ND3WI gate (ND3)

3. A 2:1 MUX driven by a single ND2WI gate (NDMX)
4. A 2:1 MUX driven by another 2:1 MUX (XOAMX)
5. A 2:1 MUX driven by a 2:1 MUX and a ND3WI gate (XOANDMX)

From these logic configurations, and the PLB diagram in Figure 4, we observe that the new VPGA cell can simultaneously implement three MX functions and one ND3 function, or one MX, one XOAMX, and one ND3 function. The PLB can also simultaneously implement a NDMX and XOAMX function. Since the XOA element also functions as a ND2WI element, two NDMX functions can be packed into a single PLB. In this configuration, one of the NDMX functions must be packed as an XOAMX function. With this degree of flexibility in packing, and the performance advantage of these logic structures compared to a 3-input LUT, we expect the higher granularity of this PLB to result in higher performance than the LUT-based PLB shown in Figure 1.

Increasing granularity also incurs an area penalty due to an increase in the number of configuration vias and total layout area. However, due to the faster logic configurations, we expect higher granularity to result in higher performance. Since the cost of potential vias is significantly less than SRAM programmable switches in an FPGA, this trade-off is worth considering for the VPGA fabric.

3. Experimental Methodology and Results

In this section, we describe the methodology that we use to implement designs using the proposed VPGA fabric. This design flow takes an RTL level description of the design as input and produces a GDSII description of the layout in the form of a regular array of PLBs with ASIC-style custom routing on the upper metal layers. We use commercial ASIC tools wherever possible to leverage the state-of-the-art in logic synthesis and physical design. We then use this design flow to compare the die-area and performance of different gate-arrays implementations using the more granular PLB shown in Figure 4 and the LUT-based PLB shown in Figure 1 [8] [11].

3.1 Design Flow

For a given PLB architecture, Figure 6 outlines the flow that we use to map an RTL-level description of a design onto a regular array of PLBs. First we use a restricted library of standard cells to obtain an ASIC-style detailed placement of the design. This part of the flow uses commercial CAD tools with the exception of a logic-compaction step. Next we legalize this placement by 'packing' the standard-cells into an array of PLBs. Our legalization algorithm works with a cost function that minimizes perturbation of the ASIC-stye placement, and thus minimizes any loss of performance or increase in area at this stage. Finally we perform ASIC-style global and detailed routing on this regular array of PLBs. In the rest of this section, we describe each stage of this flow in greater detail.

The restricted library of standard-cells used in this flow consists of the component cells of the given PLB architecture - for example MUX, XOA, ND3WI, 3-LUT buffers and inverters. The library is further restricted in the sense that each component cell has a fixed size which is chosen to give a good power-delay tradeoff. This corresponds to the size of that component cell in the PLB. The timing information for this library is generated by characterizing these cells using a commercial tool called CellRater from Silicon Metrics [1]. We use Design Compiler from Synopsys to do logic optimization and technology-mapping to this restricted library.

Technology-mapping is followed by a compaction algorithm that reduces the area of the netlist by better utilizing the given PLB architecture. Our algorithm first finds clusters of logic or 'supernodes' corresponding to functions with 3 or less than 3 inputs. This is done using a maxflow-mincut algorithm similar to Flowmap [5]. It then matches these computed supernodes to the appropriate combination of PLB components. This allows more logic to be collapsed into PLBs. For both the PLB architectures that we considered, this compaction step resulted in a significant reduction in total gate area of about 15% on the average. We then use

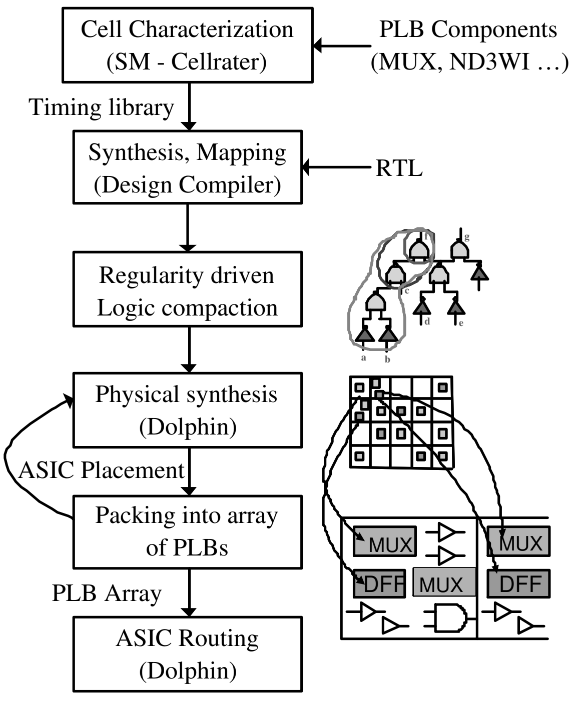

Cell Characterization (SM - Cellrater)
PLB Components (MUX, ND3WI …)
Timing library
Synthesis, Mapping (Design Compiler)
RTL
Regularity driven Logic compaction
Physical synthesis (Dolphin)
ASIC Placement
Packing into array of PLBs
PLB Array
ASIC Routing (Dolphin)
MUX
MUX
DFF
MUX
DFF

Figure 6. Design Flow

a commercial tool called Dolphin from Monterey Design

Systems [2] to perform physical synthesis and placement. The result of this stage is a detailed ASIC-style placement that has been optimized for performance, area and routability based on physical information. The resulting netlist includes logic changes and buffer insertion to meet timing constraints and area specifications.

After obtaining this ASIC-style detailed placement, our next step is to 'legalize' this by packing the component cells into a regular array of PLBs. Our packing algorithm does this by recursive quadrisection. At each quadrisection level, the component cells are relocated to other regions of the chip depending on the availability of the corresponding resource. For example, if there are more 3-LUTs in a region of the chip compared to the resources available in the PLBs in that region, some 3-LUTs will be moved to the nearest region of the chip that has unused 3-LUTs available. The cost function used in this algorithm takes into consideration the criticality of the cells being moved and also tries to minimize perturbation of the ASIC-style placement.

In order to further minimize the loss in performance due to the motion of the component cells, we use the packing algorithm in an iterative loop with the physical synthesis tool Dolphin. In each iteration, the packing algorithm restricts the locations of some of the components to regions of the chip that have unused resources available. Physical synthesis is repeated with these restrictions to give new locations for the remaining components, and also redo buffer insertion or logic restructuring where necessary to meet performance and area constraints. This iteration loop is repeated until all the components have been alloted legal locations in the PLB array. It ensures that the performance degradation due to legalizing the ASIC-style placement is minimal.

After legalization, we use Dolphin to perform ASIC-style custom global and detailed routing on the regular array of PLBs. We measure the final performance of the design by running static timing analysis in Dolphin with data from post-layout extraction.

3.2 Experimental Results

We used the flow described in Section 3.1 to implement different designs onto an gate-array of regular PLBs. In this section we compare the area and performance of gate-arrays using the granular heterogeneous architecture shown in Figure 4 and the heterogenous LUT-based architecture shown in Figure 1. We present comparisons for four different designs. The designs *ALU*, *FPU*, and *Network switch* are dominated by datapath, while the design *Firewire* is a small controller that is dominated by control logic. Our results show that the proposed granular PLB offers both a performance and density improvement.

Table 1 and Table 2 show the comparison of the final die-area and timing respectively, for each of the following design flows:

Flow a is obtained if we skip the Packing step for the design flow described in Section 3.1. Essentially, it is the standard cell ASIC flow using a library which comprises of cells that make up each PLB.

Flow b is the design flow described in Section 3.1.This produces a regular PLB array with ASIC-style custom routing.

The gate count for each design is given in units of equivalent 2-input Nand gates. With the proposed granular PLB, the die-area of the three datapath dominated designs is reduced by 32% on average. For the design *FPU*, it is reduced by as much as 40%. This improvement is despite the area of the proposed granular PLB being 20% larger than the LUT-based PLB. The primary reason for the smaller die-area with the granular PLB architecture is that majority of the functions that are mapped to a 3-LUT in the LUT-based PLB are mapped to a NDMX or XOAMX configuration in the proposed granular PLB. These configurations occupy a smaller part of the PLB than the LUT, which leaves more resources available in that PLB for packing in additional logic leading to a superior packing efficiency. The packing efficiency can be measured by looking at the die-area for *Flow b* when compared with the die-area for *Flow a*. This overhead is due to the additional packing step in *Flow b*. On an average, there is 48.37% less die-area overhead for designs that employ the granular PLB architecture. It is upto 88.6% for the design *Network Switch*.

In addition, with the granular PLB architecture, certain functions can be implemented in multiple ways in a PLB. This flexibility also results in greater packing efficiency. For example, a 2-input Nand function on a non-critical path can be mapped into a MUX without affecting performance if the ND3WI gate in the PLB is already used up, allowing an extra function to packed in the PLB.

For the design *Firewire* , however, we see that the proposed granular PLB architecture results in a larger die-area. This is due to the fact that the design is dominated by sequential rather than combinational logic. As a result most of the combinational logic in the PLB array is not utilized. Since the proposed granular PLB has 26.6% more combinational logic area than the LUT-based PLB, this results in the area overhead. This overhead can be avoided by using a PLB with a greater ratio of Flip Flops to combinational logic elements. These results also suggest the optimal PLB architecture depends on the application domain under consideration. Table 2 compares the performance after post-layout extraction with the two PLB architectures using each of the flows described above. The cycle time for all the designs is .5 ns. We compare the average slack over the top 10 critical paths in the design. Our results show that the average improvement in the slack is about 18%. For the design *FPU*, it is as much 40%. Also, there is about 68% less performance degradation from *Flow a* to *Flow b* for designs employing the granular PLB. The improvement in the performance is largely due to the fact that the 3-input functions performed by the LUT in the LUT-based PLB are performed by faster NDMX or XOAMX combinations in the granular PLB. Another factor contributing to the performance improvement is the superior packing efficiency with the granular PLB.

	Die-Area (μm^2)			
	Granular PLB		*LUT PLB*	
	flow a	flow b	flow a	flow b
ALU	6944	15147	7800	18225
Firewire	54720	68920	40944	56250
FPU	534375	662400	562500	1103625
Network switch	1822500	1960000	2088025	3294225

Table 1. Area comparison

		Path Slack 1-10 (ns)			
	No. of	*Granular PLB*		*LUT PLB*	
	gates	flow a	flow b	flow a	flow b
ALU	651	-0.301	-0.301	-0.302	-0.308
Firewire	4247	-1.03	-1.04	-1.454	-1.766
FPU	24k	-7.051	-7.089	-7.807	-9.474
Network Switch	80k	-2.778	-2.982	-2.671	-3.095

Table 2. Timing comparison

4. Conclusion

In this paper, we demonstrated that more granular heterogeneous PLBs offer further performance and density improvement over previously considered LUT-based heterogeneous PLBs, making the VPGA fabric even more competitive with standard cell designs in terms of performance. Furthermore, our results suggest that the logic block architecture should consist of some combination of Nand gates with programmable inversion, XOR gates, and MUXes. Finally, our results show that the optimal combination of these logic elements, and the optimal ratio of combinational to sequential logic elements varies with the application-domain. Accordingly, we propose to explore these issues in an application-domain specific manner in future work. Future work will also focus on exploring regular routing architectures for the VPGA fabric.

References

[1] *http://www.siliconmetrics.com/Products/CR.asp*.

[2] *http://www.mondes.com/products/dolphin.htm*.

[3] E. Ahmed and J. Rose. The effect of lut and cluster size on deep-submicron fpga performance and density. *ACM International Symposium on FPGAs*, 2000.

[4] V. Betz, J. Rose, and A. Marquardt. *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, 1999.

[5] J. Cong and Y. Ding. Flowmap: An optimal technology mapping algorithm for delay optimization in lookup-table based fpga designs. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, January 1994.

[6] A. Koorapaty. Modular, fabric-specific synthesis and heterogeneous logic block architectures for regular fabrics. *Ph.D. Thesis, Center for Silicon System Implementation, Carnegie Mellon University*, August 2003.

[7] A. Koorapaty, V. Chandra, K. Y. Tong, C. Patel, L. Pileggi, and H. Schmit. Heterogeneous programmable logic block architectures. *Proceedings of Design Automation and Test in Europe*, March 2003.

[8] A. Koorapaty, L. Pileggi, and H. Schmit. Heterogeneous logic block architectures for via-patterned programmable fabrics. *13th International Conference on Field Programmable Logic and Applications*, September 2003.

[9] L. Pileggi, H. Schmit, J. T. Shah, K. Y. Tong, C. Patel, and V. Chandra. A via patterned gate array (vpga). *Technical Report Series, Center for Silicon System Implementation, No. CSSI 02-15*, March 2002.

[10] L. Pileggi, H. Schmit, A. J. Strojwas, P. Gopalakrishnan, V. Kheterpal, A. Koorapaty, C. Patel, V. Rovner, and K. Y. Tong. Exploring regular fabrics to optimize the performance-cost trade-off. *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, June 2003.

[11] K. Y. Tong, V. Kheterpal, V. Rovner, L. Pileggi, H. Schmit, and R. Puri. Regular logic fabrics for via patterned gate array (vpga). *IEEE Custom Integrated Circuits Conference*.