An Architectural Exploration of Via Patterned Gate Arrays

Chetan Patel, Anthony Cozzie, Herman Schmit, Larry Pileggi
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
{cpatel, cozzie, herman, pileggi}@ece.cmu.edu

ABSTRACT
In this work we investigate the architecture of a Via Patterned Gate Array (VPGA) [1], focusing primarily on: 1) the optimal lookup table (LUT) size; and 2) a comparison the crossbar and switch block routing architectures. Unlike FPGAs, the routing architectures in a VPGA do not dominate the total area of the circuit. Therefore our results suggest that using smaller LUTs results in a much faster and smaller design. In the routing architecture comparison, our results also show that the switch block architecture is inferior to the crossbar architecture in terms of area utilization. As the number of routing tracks grows, the switch block architecture begins to dominate the total area of the design as in the case of the FPGAs.

Categories and Subject Descriptors
B.7.1 [**Integrated Circuits**]: Types and Design Styles – *gate array, VLSI.*

General Terms
Performance, Design, Experimentation.

Keywords
VPGA, interconnect architectures, Lookup Table, gate array.

1. INTRODUCTION
Cell-based ASIC designs are becoming more difficult to produce affordably as the technologies go below 100 nm [2]. The problems initially begin with the ever increasing mask cost. The manufacturing process is becoming so complicated that it is difficult to construct a set of simple, local design rules that would guarantee manufacturability. Parasitic extraction, as well as clock and power distribution are also challenging in a cell-based ASIC. All of these challenges are compounded further by shrinking design and product life cycles.

Regular, user-programmable devices, such as FPGAs, avoid many of these problems. Because silicon designs have high volume, mask costs and complex design optimization for manufacturability is amortized. Because an FPGA is a highly regular device, clock and power distribution can be done in a highly symmetrical manner. Unfortunately, FPGAs require more power and silicon area than a logically equivalent cell-based ASIC.

A Via Patterned Gate Array, or VPGA, is a device where all transistor and metal layers are generic. Customization to a particular application-specific function is done by defining whether the set of "potential via" locations is occupied by an actual via. This device leverages the regularity and design cost amortization benefits of an FPGA, while having the silicon area and power consumption of a cell-based ASIC. VPGAs use well-characterized, regular logic blocks and regular, fixed interconnect structures to combat manufacturability and parasitic extraction problems. Thus by this approach, VPGA offers new forms of regularity to increase the predictability for top-down designs, analogous to what is possible for FPGAs.

Our initial VPGA architecture has a structure that closely resembles an FPGA in that it consists of repetitive logic blocks with a fixed interconnect structure. There are, however, some key differences between an FPGA and our VPGA, with the biggest difference being the way the routing and logic is programmed. In an FPGA, programming of the logic cells and routing tracks are done by using SRAM cells. In the VPGA, however, the "programming" is done by configuring custom via masks during BEOL (back end of line) manufacturing. With the lack of SRAM cells, the routing architectures are placed above the CLB, thus not occupying any silicon as in the case of the FPGA.

The via configurability allows the VPGA wafer to be pre-fabricated up to the metal 2 layer. We chose metal 2 as the boundary between fully regular and via-customized because both metal and via masks below metal 2 are very complex and expensive to manufacture. Because the VPGA has a regular interconnect architecture, the masks for the upper metal layer are also pre-fabricated. One needs to only manufacture the via masks above metal 2 to personalize the VPGA. The question that remains to be answered is what should comprise the logic blocks and interconnect architecture of a VPGA. Since the VPGA closely resembles an FPGA, reconstructing certain experiments may shed some light on this question.

In an FPGA architecture, the choice of the logic cell employed can affect the area and speed of circuits that are mapped onto it. Using a small logic cell may require too many routing tracks [3]. Further, most of the delay and area is associated with the routing architecture, thus increasing the number of tracks would lead to a larger and slower FPGA [4]. Using a very large logic cell may also have the same affect. Many resources could potentially go unused and the resulting FPGA would not be very area-efficient

ISPD '03, April 6-9, 2003, Monterey, California, USA.

[3]. Therefore choosing the appropriate size of the logic cell is very crucial to an FPGA.

Besides choosing the size of the logic cell, one must also choose an appropriate routing architecture. As we have already stated, the routing architecture is one of the most important factors in determining the area, delay, and routability of the FPGA. Thus, one needs to choose the routing architecture, which enables designs to be mapped to an FPGA with the best area and performance possible.

These two arguments are very important to a design for a FPGA that is very area-efficient as well as one that that minimizes the overall delay. We believe these two arguments apply for our VPGA because it closely resembles an FPGA. In our VPGA, we also have clusters of logic cells that perform any programmed boolean operation. We also have a fixed routing architecture that routes the signals between the various logic cells.

2. EXPERIMENTAL METHODOLOGY

In order to decide upon which architecture would best suite our VPGA, this paper considers the issue of the size of the logic cell as well as what routing architecture to use. As for the size of the logic cell, we are most interested in what size of a look-up table should be use because this will affect the size of the CLB. As for the routing architecture, we are interested in comparing two distinct architectures, the crossbar and the switch point architectures. The details of these architectures will be discussed when we outline the area model for the architectures.

We explore the LUT sizes by implementing MCNC benchmark circuits. For the LUT size experiment, we first used SIS and FlowMap [5] to optimize and map each of the circuits. We then used T-Vpack [6] to group the circuits into logic blocks that contained varying LUT sizes. To place and route the circuits, we used the Versatile Place and Route tool (VPR) [6]. VPR places each of the circuits and then determines the minimum number of tracks needed to route the circuit. Using our area models and VPR's placement and routing information, we have the information to compare different LUT sizes.

To compare the different routing architectures, we first outline the area models for each architecture in Section 4. These area models give us an understanding of how the routing architectures grow as the number of tracks increases. By analyzing the area models, we can obtain an understanding about which architecture offers the most flexibility in routing as well as with routing architecture is the most area efficient.

3. ARCHITECTURE AREA MODEL

3.1 Look up Table Area Model

In order to calculate the area model for the CLB, we have to characterize the area for different LUT sizes. In our VPGA cell, the LUT is a $k-1$ level tree with a complimentary pull up and pull down network as seen in Figure 1 [7]. Each of the leaf nodes in the look-up table can contain directly to V_{DD}, ground, or another k^{th} input or its compliment. One approach would be to calculate the area of a single transistor and multiply by the total number of transistors needed to implement the LUT. In [8] and [9], the area model of a LUT is shown to be proportional to the number of transistors. Though this metric is simple and easy to

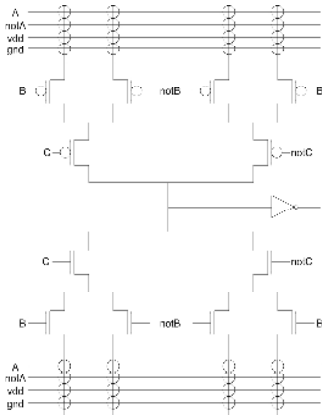

Figure 1. Schematic for a 3-input Lookup Table.

calculate, it is at best a lower bound and may not be entirely accurate for a fair comparison. When designing the layout of the LUT, one also has to consider how the LUT is configured. In our VPGA architecture, CLB configuration is done by specifying via locations between the metal 2 and metal 3 layers. Therefore all local routing internal to the CLB must be done on the metal 1 layer. The local routing would include the power and ground rails as well as the wiring required to directly connect components together in the cell. We must consider the local routing because this would add significant area as the CLB grows in size.

A more accurate approach would be to build the LUT out of 2:1 muxes as shown in Figure 2. Using this approach, we could figure out the actual area of a minimum sized mux with the power and ground rails included. To compensate for the internal routing, we could add into the mux enough physical space to account for these routing wires. For instance, a 2 input LUT would require a single 2:1 mux and would require no internal routing as in Figure 2. A 3 input LUT, however, would require 3 2:1 muxes. Therefore, we would need to add enough area to the mux to allow for an additional metal wire as in Figure 2.

Building upon this approach, the total area for a given LUT of k inputs would be

$$Area = \left(2^{k-1} - 1\right)\left(Area_{mux+(k-2)wire\_pitch}\right)$$

Table 1 shows the comparison of the area of the different LUT sizes. As in the previous case, this approach is not entirely accurate because custom designed layouts might differ in size. However, by adding in these factors, this model tries to closely depict the actual layout area, which leads to a more accurate

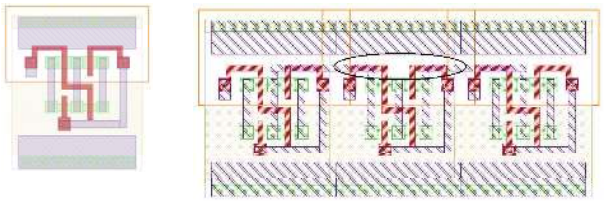

Figure 2. The layout of a 2:1 mux and a 3-input Lookup Table with the additional area highlighted.

Table 1. LUT size vs. area, delay, and CLB area.

	3 LUT	4 LUT	5 LUT
LUT Area (μm^2)	45.02	113.36	260.70
LUT delay (ps)	88.70	118.60	152.60
CLB area (μm^2)	125.18	207.04	369.45

comparison of area for different LUT sizes.

3.2 Look-Up Table Delay Model

To justify our choice for the correct LUT size, we must also have a way to characterize the delay of a given LUT. As seen in Figure 1, we constructed the SPICE file for LUT sizes ranging from 2 to 5 inputs. To keep our model consistent with the area model and to avoid making the experiment too complex, we kept all the transistors minimum sized. Using HSPICE, we simulated the LUTS using the ST 0.13 µm technology library with an output load comparable to an output buffer. Each LUT tested was configured to perform the NAND function because of ease of testing. The results in Table 1 show the delay of each different LUT size averaged across all inputs. They delay values are not entirely accurate because each LUT is not optimally sized. However this approach is still fair because we tested each LUT rather than create a model to estimate the delay for the LUTs.

3.3 CLB Area

To complete the area model for the CLB, we have to include all the components in our CLB. For our experiment, each CLB would include a Look-Up Table, a DFF, and a mux to select the desired output. What also have to include in the area model the inverters needed to generate the complemented signals for the LUT. To give a fair comparison for each LUT, we could use a model similar to the one used in calculating the area of the LUT. As in the layout, one would try to keep components the same size to create a more compact layout. Therefore, one choice would be to make the inverter and buffer the same height as the mux. By using this approach, which is similar to a standard cell approach, the power and ground rails would align and allow for a compact layout. Using the same data for the area model for the LUT, Table 1 shows the estimated CLB area for different LUT sizes.

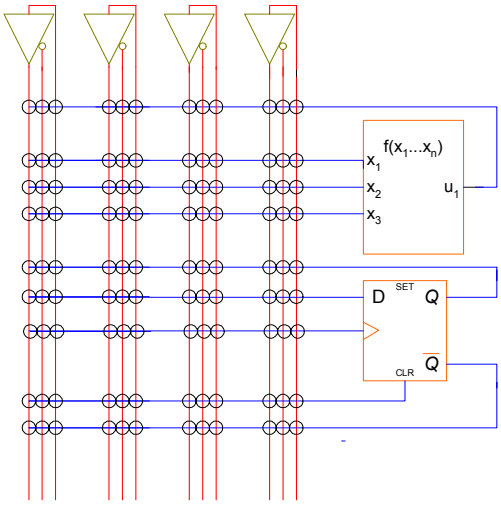

Figure 3. Depiction of the CLB showing local connectivity.

4. ROUTING ARCHITECTURE

4.1 CLB Connectivity

Figure 3 shows the connectivity of our CLB without a routing architecture above. The local connectivity of the CLB consists of a crossbar of metal 3 and metal 2 wires. The primary input pins sit atop the CLB on metal 3. Each input to the different components within the CLB are connected to metal 2 wires. To connect a primary input to either the LUT or the DFF, a via must be inserted to connect the appropriate metal 2 and metal 3 line. Therefore the configuration of the CLB occurs by specifying the via locations between the metal 2 and metal 3 layers. This approach allows the area of the CLB to be as small as possible and also allows the interconnect to connect easily to the inputs of the CLB. We have investigated two interconnect architectures, the crossbar and switch block architectures, which would both be able to connect to the CLB in this manner.

4.2 Crossbar Architecture

The crossbar architecture includes vertical and horizontal routing tracks, on two adjacent metal layers, that span the entire CLB as in Figure 4. Each horizontal track can connect directly to any vertical track by simply having a via inserted at the appropriate location. Each vertical track can also connect to any horizontal track by the same means. Figure 5 shows a close-up of the crossbar architecture showing the possible locations of vias and how the metal tracks connect to each other. The simplicity of this architecture allows us to easily calculate the area regardless of the size of the architecture. Since the tracks are just metal lines and we are assuming the there are the same number of horizontal tracks as there are vertical tracks, the area for the crossbar would simply be

$$Area_{crossbar} = (nw)^2$$

where n is the number of tracks and w is the minimum width plus the minimum pitch of a metal line.

Since our routing architecture sits directly atop the CLB in our VPGA, we include "jumpers" to allow connections between neighboring CLBs. Because all of our programming is done by introducing a via in the appropriate location, the jumper consists

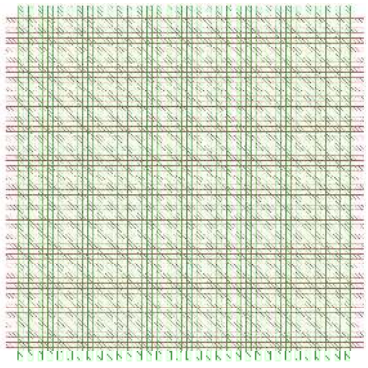

Figure 4. The layout of the crossbar routing

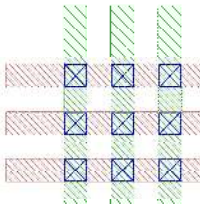

Figure 5. Possible via locations in the crossbar architecture.

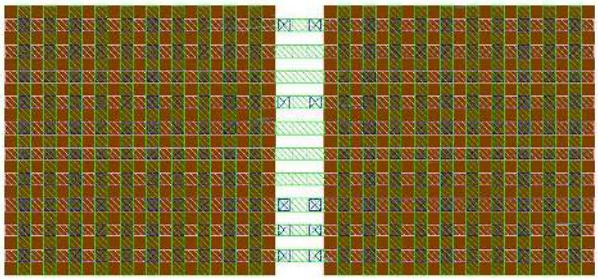

Figure 6. Jumpers connecting tracks of neighboring CLBs.

of a metal wire with two possible via locations. When a connection is to be made from one track to the same track in the neighboring routing block, the two vias are inserted to allow the two tracks to connect to each other, as shown in Figure 6. The jumpers are included in all sides of the cell to allow connections to each of the neighboring cells.

4.3 Switch block Architecture

The switch block architecture we are employing is similar to the island style switch block used in FPGAs. Each tracks requires a switch point to connect to the same track in a neighboring switch block. Figure 7 shows an example of a 16x16 VPGA switch block [10]. The difference, however, is that there are no pass transistors used to determine the connectivity of the switch points. As in the case of the crossbar, we use vias to designate a connection. As seen in Figure 8, the subset switch block is a 3x3 arrangement of metal lines with the vias inserted to show the possible connectivity points. Using this approach, each track can connect to any of the 3 other tracks in the neighboring channels. The calculation of the area of the switch block, however, is much more complicated than the crossbar architecture. For a given

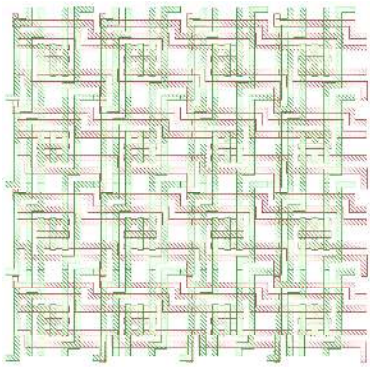

Figure 7. The layout of the switch block architecture.

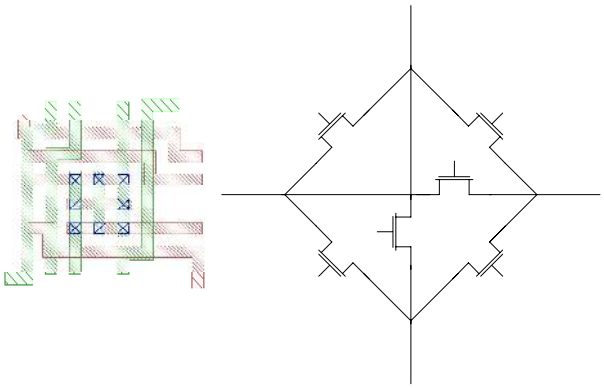

Figure 8. VPGA switch point derived from FPGA pass-transistor switch point.

number of tracks, n, a switch block contains n switch points. The area of the switch point, however, grows as n increases because there are more tracks to route. Using a very simplistic model, the switch point requires 3 metal wires in the horizontal and vertical directions and $\sqrt{n}$ horizontal and vertical metal wires as seen if Figure 8. Therefore, the total area for the switch block would be

$$Area_{switchblock} = n\left[\left(3+\sqrt{n}\right)w\right]^2$$

where w is the minimum width plus minimum pitch of a metal line. Unlike the crossbar architecture, each switch point connects directly to the same switch point in its neighboring cells. Hence there is no need for jumpers in the switch block architecture to connect neighboring routing cells.

5. EXPERIMENTAL RESULTS

Using VPR we placed and routed some of the MCNC benchmarks using the switch block architecture. Table 2 shows the results of our simulations. Using VPR, we were able to obtain the critical path delay and the number of CLBs used to place each benchmark. Using our area models discussed above, we were then able to compare which LUT size is best suited for our VPGA. Since our routing architecture sits atop our CLB, for each benchmark, we computed the area as the max of the switch block area and the CLB area.

From Table 2, one can see that the most area efficient LUT size is 4. Using a LUT size of 3 does not incur much of an area penalty but using sizes larger than 4 are not very area efficient. A 4 LUT is more area efficient than a 3 input LUT because the 4 LUT does a good job of decreasing the depth of the nets, hence requiring less CLBs for any design. At the same time, the area for larger LUT sizes becomes too large and destroys this benefit.

One can also see from Table 2 that using the 4 LUT also produces the fastest design. Again this results from decreasing the depth of the nets. Larger LUT sizes would also accomplish the same task, however. The difference however is that the area of the larger LUTs causes the routing tracks to be much longer, hence more resistive and capacitive.

In the comparison of the routing architectures, we first tried to analyze which architecture would possess more flexibility in routing. Given a certain area, the crossbar architecture would always possess many more routing tracks than the switch block

Table 2. Critical path delay and Total area for LUT size vs. benchmark.

	3lut	4lut	5lut
benchmark	crit.path (ns)	crit. Path (ns)	crit. Path (ns)
alu4	13.71	8.11	10.55
apex2	12.50	10.33	11.00
apex4	15.42	10.96	14.17
bigkey	29.86	29.81	22.72
des	26.04	34.07	20.00
diffeq	16.85	8.30	9.56
dsip	23.29	18.18	21.12
elliptic	28.74	17.27	34.72
ex1010	54.83	21.42	50.84
ex5p	12.43	8.15	12.17
misex3	10.82	7.53	9.21
s298	26.32	28.01	17.38
seq	14.60	9.64	14.75
spla	48.76	30.28	45.15
tseng	16.00	6.76	8.69
geo mean	**20.60**	**14.12**	**17.05**
benchmark	Total area (μm^2)	Total area (μm^2)	Total area (μm^2)
alu4	349878.10	332920.32	563041.80
apex2	420855.16	438510.72	780278.40
apex4	264503.33	267288.64	527205.15
bigkey	554422.22	607662.40	585947.70
des	753458.42	449690.88	932122.35
diffeq	431620.64	400208.32	670551.75
dsip	368404.74	335197.76	672029.55
elliptic	983163.72	866462.40	1670283.45
ex1010	1606572.88	1188127.17	1889693.56
ex5p	352006.16	249069.12	477329.40
misex3	331476.64	310767.04	556391.70
s298	870515.61	527537.92	706388.40
seq	404706.94	404349.12	740008.35
spla	1490400.84	815116.48	1512528.30
tseng	352006.16	268944.96	458487.45
geo mean	**536628.66**	**445090.54**	**763087.35**

architecture. We felt the increase in the number of available tracks would make the crossbar architecture a more flexible routing architecture.

In these benchmarks tested, only a few cases occurred when the routing area was larger than the size of the CLB. Most of the benchmarks were logic dominated, meaning that the switch block architecture fit directly over the CLB. Because the benchmarks were logic dominated, preliminary results showed very little benefit from using the crossbar architecture for any area gain.

It is also important to note that given the fact that these benchmarks are mostly logic dominated and not routing dominated, timing results for the crossbar architecture would be slower than the switch block architecture. The reason we account for the extra delay lies in the dangling capacitance. In the switch block architecture, a signal can go in any direction at very little cost because it would always have to travel though a either 1 or 2 vias as seen in Figure 8. In the crossbar architecture, switching between some tracks might incur a greater penalty. The increase in penalty occurs when a signal changes direction. As seen in Figure 4, switching between one of the middle horizontal tracks and one of the middle vertical tracks would add some dangling capacitance. Though this may not be a significant problem for small nets, as the nets become larger, the dangling capacitance severely affects the delay for the net. Preliminary results show that the critical path for the crossbar architecture is much slower than that of the switch block architecture. Therefore when the routing resource is not heavily occupied and the area is mostly logic dominated, the switch block serves as the better interconnect structure.

6. CONCLUSIONS

In this paper, we have investigated two important questions for our VPGA architecture. First we showed that the optimal LUT size is 4. In most cases, circuits mapped to a 4 LUT are faster as well as more area efficient. Using a 3 LUT would not incur much of an area penalty but would affect the performance of circuits that are mapped to it. A LUT size of 5 is inferior in both terms of area and delay.

We have also compared the crossbar architecture to the switch block architecture. The crossbar architecture is more area efficient than the switch block architecture. The crossbar architecture also has to deal with the dangling capacitance issue while the switch block does not. More work needs to be done to see if there is any gain with the use of the crossbar architecture when the VPGA is not logic dominated, but routing dominated.

Though we have gained an understanding of how these factors would affect our design for VPGA, many questions still remain to be answered. Firstly, how would introducing heterogeneity affect which LUT size is better. Prior work in [11] has shown that introducing NAND gates in combination of LUTs produces designs that are much faster. More work also needs to be done with the interconnect structures to weigh their routability, their area, and most importantly, their delay.

7. ACKNOWLEDGMENTS

We would like to thank Aneesh Koorapty and Vikas Chandra for their help with the VPR simulations. We would also like to thank ST Microelectronics for allowing us to use their 0.13 μm technology. This project was supported by the Microelectronic Advanced Research Corporation (MARCO) and the Gigascale Silicon Research Center (GSRC).

8. REFERENCES

[1] L. Pileggi, H. Schmit, J. Shah, Y. Tong, C. Patel, V. Chandra, "Via Patterned Gate Array (VPGA)," *Technical Reports Series of the CMU Center for Silicon System Implementation*, No. CSSI 02-15, March 2002.

[2] W. Maly, "IC design in high-cost nanometer-technologies era," *IEEE Proc. of DAC*, June 2001, pp. 9-14.

[3] Marquardt, V. Betz and J. Rose, "Using Cluster-Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density," *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, CA, February 1999, pp. 37 – 46.

[4] V. Betz and J. Rose, "FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density," *ACM/SIGDA International Symposium of Field Programmable Gate Arrays, Monterey*, CA, February 1999, pp. 59 – 68.

[5] J. Cong and Y. Ding, "FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs," *IEEE Trans. on CAD*, Jan. 1994, pp 1 – 12.

[6] V. Betz and J. Rose, "VPR: A New Packing, Placement and Routing Tool for FPGA Research," *Int. Workshop on Field Programmable Logic and Applications*, 1997, pp. 213 – 222.

[7] K. Y. Tong, C. Patel, P. Gopalakrishnan, L. Pileggi, H. Schmit, R. Puri, "Lookup Tables for a Via Patterned Gate Array (VPGA)," *Technical Reports Series of the CMU Center for Silicon System Implementation*, No. CSSI 03-002, January 2002.

[8] J. Rose, R. J. Francis, P. Chow, D. Lewis, "The Effect of Logic Block Complexity on Area of Programmable Gate Arrays," *IEEE Custom Integrated Circuit Conference*, May 1989, pp. 5.3.1 – 5.3.5.

[9] J. Rose, R. J. Francis, D. Lewis, P. Chow, "Architecture of Field-programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency," *IEEE Journal of Solid-State Circuits*, Volume: 25 Issue: 5, Oct. 1990, pp. 1217 – 1225.

[10] H. Schmit and V. Chandra, "FPGA Switch Block Layout and Evaluation," *IEEE/ACM International Symposium on Field Programmable Gate Arrays (FPGA – 02)*, February 2002.

[11] A. Koorapty, V. Chandra, K. Y. Tong, C. Patel, L. Pileggi, H. Schmit, "Heterogeneous Programmable Logic Block Architectures," *Proceedings of Design Automation and Test in Europe*, March 2