Improving FPGA Performance and Area Using an Adaptive Logic Module

Mike Hutton[1], Jay Schleicher[1], David Lewis[2], Bruce Pedersen[1], Richard Yuan[1], Sinan Kaptanoglu[1], Gregg Baeckler[1], Boris Ratchev[1], Ketan Padalia[2], Mark Bourgeault[2], Andy Lee[1], Henry Kim[1] and Rahul Saini[1]

[1] Altera San Jose, 101 Innovation Dr., San Jose CA, USA, 95134.
mhutton@altera.com
[2] Altera Toronto, 151 Bloor St. W., Toronto, Ontario, Canada, M5S 1S4

Abstract. This paper proposes a new adaptable FPGA logic element based on fracturable 6-LUTs, which fundamentally alters the longstanding belief that a 4-LUT is the most efficient area/delay tradeoff. We will describe theory and benchmarking results showing a 15% performance increase with 12% area decrease vs. a standard BLE4. The ALM structure is one of a number of architectural improvements giving Altera's 90nm Stratix II architecture a 50% performance advantage over its 130nm Stratix predecessor.

1 Introduction

Previous research on LUT-size for FPGAs [12][14][15][10] has consistently shown that a 4-LUT provides the best area-delay product. This is based on the fact that larger LUTs can absorb more logic and decrease the critical path length, but require increasing resources for LUT-mask and input muxing. Mainstream Altera and Xilinx SRAM-based FPGAs use a 4-LUT, though this sometimes comes with additional hardware to compose base logic elements.

Here we show a new view of this tradeoff, illustrated in Figure 1. By novel use of input sharing and fracturability we are able to get the advantages of larger LUT sizes without paying the high price of the additional inputs required to build 5 or 6-LUTs.

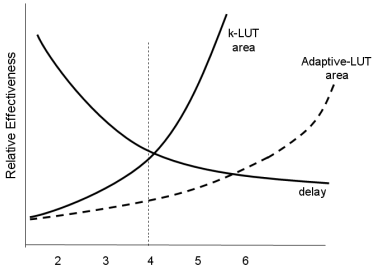

Fig. 1. Area/delay Tradeoff

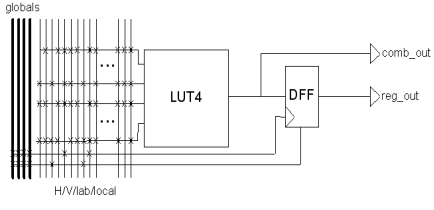

Fig. 2. BLE4 logic element

J. Becker, M. Platzner, S. Vernalde (Eds.): FPL 2004, LNCS 3203, pp. 135–144, 2004.

The adaptive logic module developed in this paper allows us to decrease the critical path depth by 20% on average, but because the structure can be used either as a 6-LUT, two 5-LUTs with sharing, or other combinations without the need for expensive additional input muxing, we are able to achieve this without area penalty. With further improvements built on the ALM we can actually show an area benefit.

2 Logic Element Architecture and Adaptive Logic Modules

In this paper the generic BLE4 of Figure 2 and it's BLE5 and BLE6 analogs form the base comparison for the new logic structure. Empirically as we map for larger values of k nominal area and delay decrease. For BLE5 an average netlist uses 15% fewer LUTs and has 25% shorter unit delay; for BLE6 this is (-22%, -36%) and for BLE7 (-28%, -46%). However, as we move from BLE4 to BLE5 we add 16-bits of SRAM for the LUT-mask, and a new input mux to sample the neighboring connection block (see [5] for terminology). The true chip-area metric of #LEs * sizeof(LE) is minimized at about k=4 as shown in Figure 1. Related previous work also involves the use of heterogeneous LUT sizes [8], and hybrid PTERM/LUT architectures [9].

An interesting property of tech-mapping for larger LUTs is a decrease in efficiency. In a 6-LUT mapping, for example, only about 1/4 of LUTs end up as 6-input functions, the rest are underutilized. A design mapping to 100 LUTs with k=4 will map to 78 6-LUTs with a distribution of {23,32,17,9,13} LUT-{6,5,4,3,2} functions, based on experiments with RASP/FlowMap [7][6] and confirmed with Altera tech-mappers.

The cost of larger k is not just the LUT mask, though that is significant. Most dominant is the input mux, which is roughly 30:1 in a Stratix LAB. Though CLB and LAB routing structures are rather different this is also roughly 30:1 for VirtexII, based on quotes in [2]. The register, adder and other logic is unchanged.

Two reasons why a 6-LUT might be more preferable for depth than previously seen are that both the area devoted to routing and the relative delay contribution of interconnect to the critical path have been increasing consistently with new process generations. Also, the concept of LAB hierarchy and routing flexibility introduced in Altera's FLEX8K architecture, discussed in [3][1] and improved upon with depopulation [11] has minimized the effect. All modern FPGAs utilize some degree of cluster-based hierarchy. However, to use a 6-LUT effectively, we need to deal efficiently with the increase in input muxing, and the wastage involved in building a 6-LUT which is often underutilized.

If LUT-mask were the only concern, than we could compose multiple base LEs into larger LUTs, as shown in Figure 3.

The drawback with this approach is the ensuing area cost. This structure has a total of 19 input muxes, and 4 registers. When used as a 6-LUT, multiple signals have to be routed repeatedly through the connection block, and 3 of the 4 registers and sets of output muxing are wasted. The netlist of 100 BLE4 LUTs quoted above will re-map on average into 78 6-LUTs with a distribution of 23 LUT6, 32 LUT5, and 39 other. To implement this directly in the structure of Figure 3 would cost well more than 100

BLE4s in equivalent area. Using BLE5 as the base (meaning you can't use the BLE4 outputs) would help, but would still be more expensive than the BLE4 base.

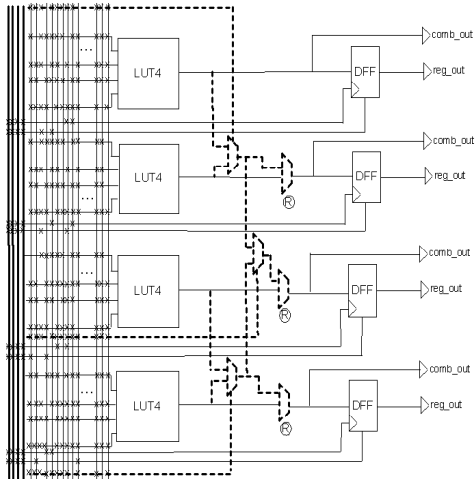

Fig. 3. Composing BLE4 to build a LUT6.

Our solution to this problem is the fracturable logic module, shown abstractly in Figure 4. The logic structure has a total of 8 input-muxes, and provides 4 functional outputs (2 comb, 2 reg), uses 64 bits of LUT-mask (6-input complete) and 2 bits of arithmetic and registers. We denote this a 6,2 fracturable LE, because it is a 6-LUT with 2 additional inputs for use when fracturing to smaller LUTs. These extra inputs are key to facilitating packing of non 6-input functions, without the overkill of adding a complete set of 10 or more inputs. Variants of this structure such as 4,2 are possible, but beyond the scope of this paper. Each of the two outputs (top and bottom) are denoted as an ALUT. This terminology is necessary in order to account for area later.

The ALM can implement one 6-LUT, two 5-LUTs which share 2 inputs, or two independent 4-LUTs, among other combinations.

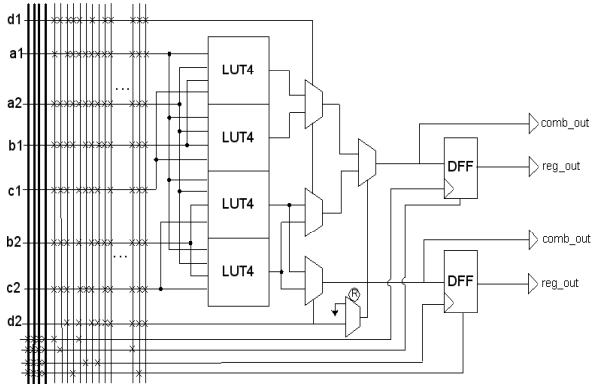

Fig. 4. Fracturable 6,2 adaptable logic module (first version).

Observe that the logic module of Figure 4 is more comparable in area with two BLE4 logic elements than with the four shown in Figure 3, because it has the same number of input mux, FF, output mux and arithmetic cost; only the proportion of the logic element devoted to the LUT-mask is increased. In functional terms, it is closer to the composition of two BLE5 logic elements.

Though the difference between composing two independent LEs and fracturing one compound structure is subtle, the most important issue to understand is the difference in the area cost of the two approaches.

2.1 Outputs, LUT-Mask Sharing, and 7-Input Functions

We can make a number of improvements to this first version of the ALM. One issue with Figure 4 is the number of outputs. We have partially addressed this by pushing the output merging back one stage in order to incur a speed hit only on the 6th stage input d2. When in fractured mode we set the SRAM-bit to 0 to disconnect the upper ALM-half from the lower.

However we can do better with the following transformation to Figure 5. First we duplicate the 2nd level muxes controlled by the 5th stage (d1 input), and add a new mux which choose c1 or GND on the top ALUT and c2 or VCC on the bottom. The effect of the transformation is to remove the additional output muxing from the critical path of the LE for all speed-paths, pushing it to the middle inputs only. Routing interfaces are identical to Figure 4 and are not shown.

As a further transformation, we introduce swap muxes controlled by R and T, for reasons which will become clear shortly.

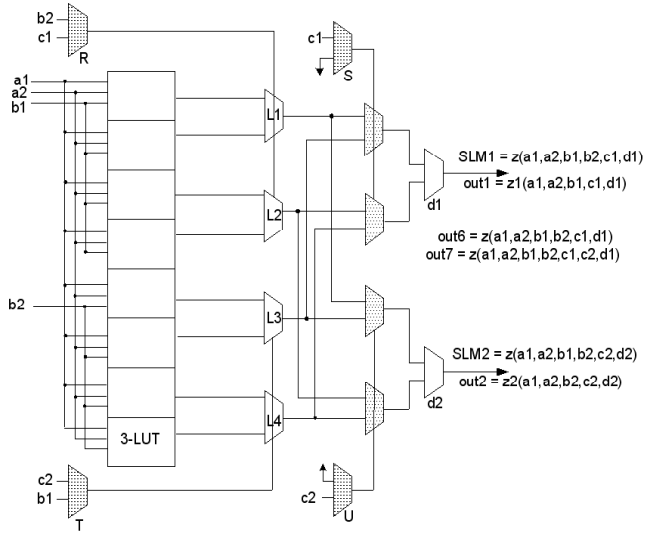

Fig. 5. 6,2 ALM with 2 outputs and shared LUT-mask.

The operation of the logic module of Figure 5 is now a bit more complex. We still have two outputs and 8 inputs, and all previous properties. However there is an additional benefit from the latter transformation that makes it particularly clever. Note that when a 6,2 ALM is used in 6-LUT mode, there are two outputs unused (wasted). With the additional circuitry, which we call "shared LUT-mask" or SLM, we are now able to configure the logic module to implement two 6-input functions that share 4 inputs as long as they share the identical LUT-mask function. By setting R=1 and T=1, S=0 and U=1, and reorganizing the LUT mask appropriately, SLM1 becomes a 6-LUT function of (a1,a2,b1,b2,c1,d1) and SLM2 becomes the same function, only of (a1,a2,b1,b2,c2,d2).

This seemingly obscure property is incredibly useful in practice. Designs which contain multiple barrel shifters and crossbars will synthesize into many 4:1 muxes with common data and different select lines, which fit perfectly into the SLM structure. For example, a benchmark SPI-4 (posphy level 4) core is able to implement about 12% of all ALMs as packed pairs of 6-LUTs implementing 4:1 muxes, meaning a 12% overall savings in ALM area.

A further side-effect of this transformation is that the ALM of Figure 5 can also implement a restricted set of 7-input functions.

Setting R=0 the upper two 4-LUTs are arbitrary functions of (a1,a2,b1,b2). Setting T=1 the bottom 4-LUTs are arbitrary functions of (a1,a2,b2,c2). Setting S=1 makes the upper shaded muxes controlled by c1 and the results of these controlled by d1. When c1=0 out7 is driven by the L1 and L3 outputs chosen by d1. When c1=0 out7 is driven by the L2 and L4 outputs chosen by d1. The result is that we can compute a class of 7-input functions using all the inputs except for d2.

Specifically, we can implement any 7-input function that can be expressed as

F1 = fn(a1,a2,**b1**,b2,d1), F2 = fn(a1,a2,b2,**c2**,d1), Out = mux(F1,F2,c1)

Thus we can compute the c1-controlled mux of two 5-input functions which share 4 of their inputs, differing only in b1 and c2. The reason that the output of the functional template is controlled by c1 rather than d1 as shown in the physical diagram comes from the LUT-mask changes performed by synthesis to rotate the c1 and d1 effects (this does require some thought to see completely).

Figure 6 shows how to build an 8:1 mux in 2 ALMs (4 ALUTs) using this property. We first compute sub-functions y0 and y1. Since y0 and y1 are 5 input functions with two shared inputs they pack into a single ALM and generate the two outputs y0 and y1. In the second ALM we compute the output of the 8:1 mux using F1=fn(s0,s1,d3,y0,y1) and F2=fn(s0,s1,d7,y0,y1) and the 2:1 mux controlled by s2. In both sub-functions one of the bridged inputs is unused, but nonetheless the result is a partial 7-input function matching the above template.

An 8:1 mux implemented with simple BLE4 requires 5 BLE4 logic elements vs. 2 ALMs, which saves roughly the area of a BLE4.

It is worth noting that though the composable BLE4 structure of Figure 3 is not efficient for making 6-LUTs or 5-LUTs, it is quite useful for building muxes – it can build an 8:1 mux in 4 composable BLE4s, which is comparable in area to the 2 ALM solution.

As a final comment on the choice of 6-LUTs as the basis for the ALM, we note that in addition to 4:1 muxes, 6-input functions are natural implementations for many other logical functions. One such class of functions is DES encryption.

The core operation of DES is an array of 8 sboxes or substitution tables. Each sbox has 6 inputs and produces 4 outputs. In a parallel implementation when targeting speed, it is usual to produce 128 SBOXes, each of which needs 6 BLE4 for each of 4 outputs (3072 LEs).

The sbox has a natural implementation in 4 6-LUTs. Due to the complex nature of the function each of the 6-LUTs would otherwise require the worst-case 6 4-LUTs shown in Figure 7. This behavior is typical of other encryption functions such as Rijndael also, and is a further justification that 6-LUTs are a "natural" building block for combinational functions.

For an area-optimized DES core, the ALM described in this paper uses 239 ALMs vs. 736 in BLE4. For the speed-optimized version, we use 1465 ALMs vs. 5352 BLE4 logic elements. This represents a roughly 35% and 45% overall area improvement, respectively.

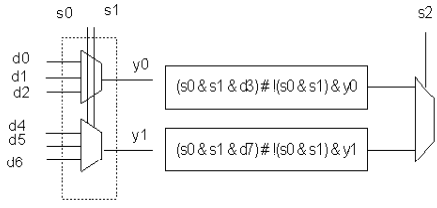

Fig. 6. 7-input function for 8:1 mux

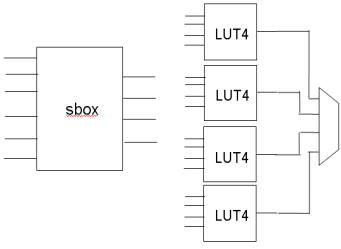

Fig. 7. DES sbox with BLE4

In a naïve implementation, the total area of the Figure 5 ALM is a little larger than two base BLE4s, roughly 15-20%. This comes from the additional LUT-mask SRAM bits and extra 2:1 muxes and configuration. However, since layout of the ALM is done as a pair, much of this can be clawed back with intelligent layout sharing. In overall chip area the physical implementation of Figure 5 is roughly area-neutral with the BLE4 architecture yet achieves the 36% decrease in logic depth.

3 Balanced Technology Mapping

It is critical to balance the distribution of LUT-sizes away from the natural distribution of tech-mapping to one which is more packable and facilitates SLM.

Consider Figure 8. The network on the left can be covered by three ALUTs – two 6-LUTs and one 4-LUT (upper solution), and this is the solution that FlowMap will generate. After packing, the solution has depth 2 and 2 ½ ALMs. The solution to the bottom, though it generates 4 ALUTs (all 5-LUTs), can be efficiently packed into just 2 ALMs while maintaining the depth-2 solution.

We refer to modifying the distribution of LUT-sizes to improve packing as balancing. The goal of balanced mapping is to maintain optimal critical path depth (unit delay) while producing a more packable LUT distribution.

The primary issue to tech-mapping for a good distribution is to modify cost functions to avoid 6-LUTs when they are not necessary for delay minimization. Further discussion is beyond the scope of this paper. However, Figure 9 shows empirical results from our prototype tools using default, balanced and aggressive balancing.

In the prototype we captured 7-LUT functions not by mapping to k=7, but rather by specifically recognizing 8:1 muxes in RTL synthesis, and then post-processing the netlist after tech-mapping. On average, we find that 7% of all ALMs are able to implement a 7-input function, a very significant area benefit.

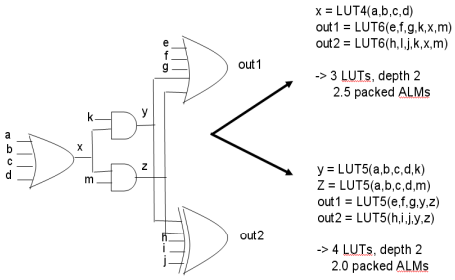

Fig. 8. Better with balanced mapping.

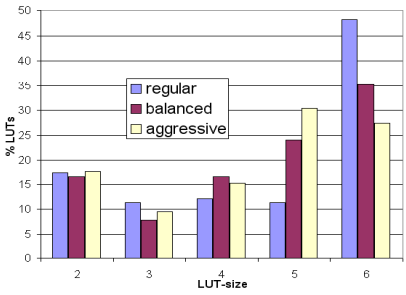

Fig. 9. LUT-size distribution

4 Experimental Results

This section shows results on 80 large VHDL/Verilog industrial designs, using prototype architecture development tools. Each design is synthesized and mapped by the Quartus II architecture evaluation flow, then clustered, placed and routed by our parameterized architecture evaluation toll PMT. Architectures are generated automatically by PMT based on the size of the design, to emulate as-full-as-possible chips.

For evaluating the ALM, the routing architecture of the LIM and global network outside the LAB is held fixed. However optimization sweeps for connectivity of the input mux is performed for both the BLE4 and ALM 6,2 cases, so that each receives its optimal layout and connectivity while maintaining routability. Timing delays for the different delays through the BLE4 and ALM are obtained by Spice simulation based on a preliminary layout on a common 130nm process. Routing delays are also obtained by Spice, and are common between the two architectures. Area models are computed using preliminary layout estimation, also using common 130nm process design rules. Layout optimizations and rough transistor sizing is done to optimize the BLE4 and ALM independently. As a caveat, design efforts were biased towards performance over area, so results could change slightly with different emphasis.

Figure 10 shows performance results (as a ratio), overall a 15% geomean improvement. Figure 11 shows chip area, with a 12% geomean improvement. For area,

we capture the effect of clustering into LABs and routing architecture, by using the metric "labsize * sizeof(lab)" as this is the fairest comparison. Both architectures route all designs.

The ALM structure introduced in this paper is one of a number of architectural changes introduced in the Stratix II family of FPGAs recently announced by Altera, further architectural changes were also made to the LAB and routing structure, and these are in general additive.

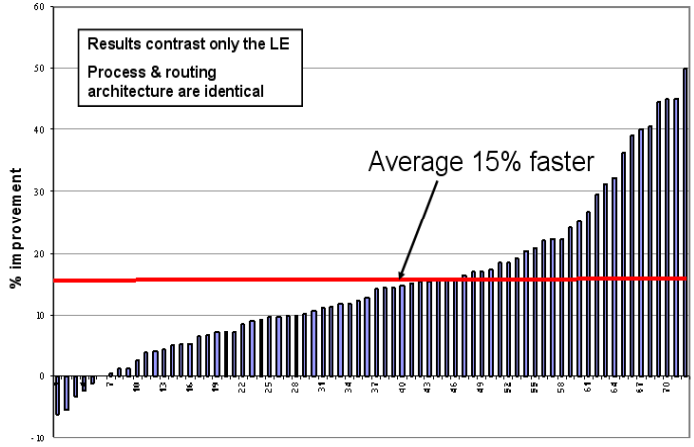

Fig. 10. Performance improvement of 6,2 ALM vs. BLE4

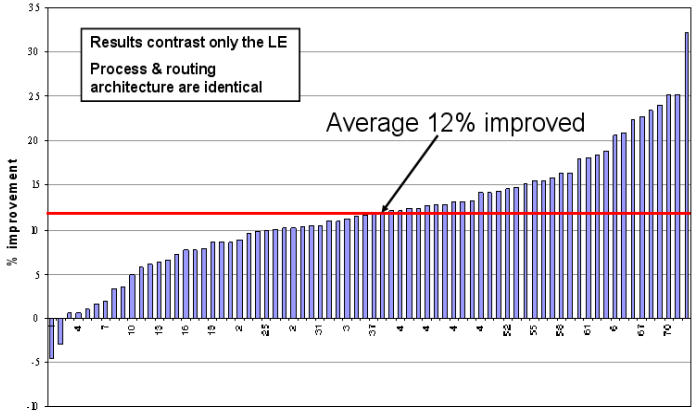

Fig. 11. Area improvement of 6,2 ALM vs. BLE4

Figure 12 shows the breakdown of overall performance gains in the Stratix II architecture over Stratix. In contrast with the earlier discussion, these are bottom-line results comparing production software and timing models in both cases and including the 90nm process gains for Stratix II.

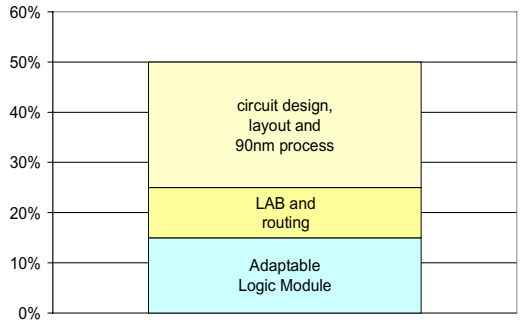

Fig. 12. Stratix II Silicon Performance Improvements vs. Stratix

5 Conclusions

This paper presents a new and novel adaptive logic module structure for FPGAs. The goals of the ALM is to allow technology mapping to 6-input functions in order to capture the depth benefits of wider functions without the unacceptable cost that would be incurred with a BLE6 based logic element.

We showed the sources of area cost of building logic elements with more than 4 inputs and how those costs break down into both the obvious costs (LUT-mask size) and the large but less apparent costs such as input and output muxing and appropriate number of FFs and outputs per block.

We presented a specific logic element based on a 6-LUT that is fracturable into 5 LUTs, but that using sharing and other optimizations can be implemented with area comparable to two BLE4 logic elements. Further extensions to the logic element improve propagation delay through the logic function, allow for partial functions of 7-inputs, and allow two 6-input functions that share 4 inputs and the same LUT mask to be implemented in the same logic element – a 2X area savings when used. Efficient balancing is achieved through improved software, and by choosing the right balance of extra inputs needed to achieve good packing results vs. the cost of providing them.

Overall comparisons between an FPGA architecture based on the 6,2 ALM and based on a 4-input LUT on the same process and with no routing architecture changes show an average performance gain of 15% and average decrease in chip area of 12%.

A version of the adaptable logic module described in this paper has been implemented as a key component of the Stratix II family of commercial FPGAs from Altera. The 15% performance improvement from the ALM along with further architectural changes and process migration results in a 50% average performance improvement between the 90nm Stratix II and it's predecessor Stratix on 130nm technology.

Acknowledgements. Thanks to Richard Cliff, Misha Burich, David Mendel and Paul Leventis for their reviews and comments improving the presentation of this paper.

References

[1] Ahmed and J. Rose, "The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density," in Proc. ACM Symp. FPGAs, pp. 3-12, 2000.

[2] J. Anderson, F. Najm and T. Tuan, "Active Leakage Power Optimization for FPGAs", in Proc. ACM Symp. FPGAs, pp. 33-41, 2004.

[3] V. Betz and J. Rose, "Cluster-Based Logic Blocks for FPGAs: Area-Efficiency vs. Input Sharing and Size", in Proc. Custom Integrated Circuits Conference 1997, pp. 551-554.

[4] V. Betz, J. Rose and A. Marquardt. "Architecture and CAD for Deep-Subicron FPGAs", Kluwer, 1999.

[5] S. Brown, R. Francis, J. Rose and Z. Vranesic, "Field-Programmable Gate Arrays", Kluwer, 1992.

[6] J. Cong and Y. Ding, "FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs", IEEE Trans. CAD Vol 13 No 1, pp. 1-12, 1994.

[7] J. Cong, J. Peck and Y. Ding, "RASP: A General Logic Synthesis System for SRAM-based FPGAs", in Proc. ACM Symp. FPGAs, pp. 137-143, 1996.

[8] J. He and J. Rose, "Advantages of Heterogeneous Logic Block Architecture for FPGAs", in Proc. IEEE Custom Integrated Circuits Conf. (CICC), pp. 7.4.1-7.4.5, 1993.

[9] Kaviani and S. Brown, "Hybrid FPGA Architecture", in Proc. ACM Synp. FPGAs, pp. 3-9, 1996.

[10] J. Kouloheris and A.El Gamal, "FPGA Performance vs. Cell Granularity", Proc. of Custom Integrated Circuits Conference, May 1991, pp. 6.2.1 - 6.2.4.

[11] G. Lemieux and D. Lewis, "Using Sparse Crossbars Within LUT Clusters", in Proc. ACM Symp. FPGAs, pp. 59-68 2001.

[12] J. Rose, R.J. Francis, D. Lewis and P. Chow, "Architecture of Field-Programmable Gate Arrays: The Effect of Logic Functionality on Area Efficiency", IEEE Journal of Solid-State Circuits, pp. 1217-1225, 1990.

[13] J. Rose, R.J. Francis, P. Chow and D. Lewis, "The Effect of Logic Block Complexity on Area of Programmable Gate Arrays", Proc. IEEE Custom Integrated Circuits Conference (CICC), pp. 5.3.1-5.3.5 1989.

[14] S. Singh, "The Effect of Logic Block Architecture on FPGA Performance", M.A.Sc. Thesis, University of Toronto, 1991.

[15] S. Singh, J. Rose, P. Chow and D. Lewis, "The Effect of Logic Block Architecture on FPGA Performance", IEEE Journal of Solid-State Circuits, Vol 27 No. 3, pp. 282-287, 1992.

[16] S. Trimberger, K.Duong, and B. Conn, "Architecture Issues and Solutions for a High-Capacity FPGA", Proc. ACM Synp. FPGAs, pp. 3-9, 1997.

[17] K. Veenstra, B. Pedersen, J. Schleicher and C. Sung, "Optimizations for a Highly Cost-Efficient Programmable Logic Architecture", in Proc. ACM Symp. FPGAs, pp. 20-24, 1998.