International Journal of Research
Available at https://edupediapublications.org/journals

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue 13
October 2017

Modelling And Analysis Of Hybrid Lut/Multiplexer Fpga Logic Architectures

Anitha Banoth
Department of ECE, CITS, Warangal, Telangana
Email id: anitha.banoth86@gmail.com

ABSTRACT: *Hybrid configurable logic block architectures for field-programmable gate arrays that contain a mixture of lookup tables and hardened multiplexers are evaluated toward the goal of higher logic density and area reduction. Multiple hybrid configurable logic block architectures, both nonfracturable and fracturable with varying MUX:LUT logic element ratios are evaluated across two benchmark suites (VTR and CHStone) using a custom tool flow consisting of LegUp-HLS, Odin-II front-end synthesis, ABC logic synthesis and technology mapping, and VPR for packing, placement, routing, and architecture exploration. Technology mapping optimizations that target the proposed architectures are also implemented within ABC. Experimentally, we show that for non fracturable architectures, without any mapper optimizations, we naturally save up to ~8% area postplace and route; both accounting for complex logic block and routing area while maintaining mapping depth. With architecture-aware technology mapper optimizations in ABC, additional area is saved, post-place-and-route. For fracturable architectures, experiments show that only marginal gains are seen after place-and-route up to ~2%. For both nonfracturable and fracturable architectures, we see minimal impact on timing performance for the architectures with best area-efficiency.*

INTRODUCTION:

THROUGHOUT the history of field-programmable gate arrays (FPGAs), lookup tables (LUTs) have been the primary logic element (LE) used to realize combinational logic. A K-input LUT is generic and very flexible—able to implement any K -input Boolean function. The use of LUTs simplifies technology mapping as the problem is reduced to a graph covering problem. However, an exponential area price is paid as larger LUTs are considered. The value of K between 4 and 6 is typically seen in industry and academia, and this range has been demonstrated to offer a good area/performance compromise [4], [5]. Recently, a number of other works have explored alternative FPGA LE architectures for performance improvement [6]–[10] to close the large gap between FPGAs and application-specific integrated circuits (ASICs) [11]. In this paper, we propose incorporating (some) hardened multiplexers (MUXs) in the FPGA logic blocks as a means of increasing silicon area efficiency and logic density. The MUX-based logic blocks for the FPGAs have seen success in early commercial architectures, such as the Actel ACT-1/2/3 architectures, and efficient mapping to these structures has been studied [12] in the early 1990s.However, their use in commercial chips has waned, perhaps partly due to the ease with which logic functions can be mapped into LUTs, simplifying the entire computer aided design (CAD) flow. Nevertheless, it is widely understood that the LUTs are inefficient at implementing MUXs, and that MUXs are frequently used in logic circuits. To underscore the inefficiency of LUTs implementing MUXs, consider that a sixinput LUT (6-LUT) is essentially a 64-to-1 MUX (to select 1 of 64 truth-table rows) and 64-SRAM configuration cells, yet it can only realize a 4-to-1 MUX (4 data + 2 select = 6 inputs). In this paper, we present a six-input LE based on a 4-to-1 MUX, MUX4, that can realize a subset of six-input Boolean logic functions, and a new hybrid complex logic block (CLB) that contains a mixture of MUX4s and 6-LUTs. The proposed MUX4s are small compared with a 6-LUT (15% of 6-LUT area), and can efficiently map all {2, 3}-input functions and some {4, 5, 6}-input functions. In addition, we explore fracturability of LEs—the ability to split the LEs into multiple smaller elements—in both LUTs and MUX4s to increase logic density. The ratio of LEs that should be LUTs versus MUX4s is also explored toward optimizing logic density for both nonfracturable and fracturable FPGA architectures.

To facilitate the architecture exploration, we developed a CAD flow for mapping into the proposed hybrid CLBs, created using ABC [13] and VPR [14], and describe technology mapping techniques that encourage the selection of logic functions that can be embedded into the MUX4 elements. The main contributions in this paper are as follows.

1) Two hybrid CLB architectures (nonfracturable and

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue 13
October 2017

fracturable) that contain a mixture of MUX4 LEs and the traditional LUTs yielding up to 8% area savings.
2) Mapping techniques called NaturalMux and MuxMap targeted toward the hybrid CLB architecture that optimize for area, while preserving the original mapping depth.
3) A full post-place-and-route architecture evaluation with VTR7 [1], and CHStone [2] benchmarks facilitated by LegUp-HLS [3], the Verilog-to-Routing project [1] showing impact on both area and delay.

Compared with the preliminary publication [15], we have performed transistor level modelling of the MUX4 LE, further studied the fracturable architectures, and unified the open source tool-flow from C through LegUp-HLS to the VTR flow. Sparse crossbars (versus full crossbars in the previous work) have also been included in our CLBs, increasing modelling accuracy. The new transistor-level modelling of the MUX4 also provides more accurate results as compared with the previous work. Results have also been expanded with the inclusion of timing results as well as larger architectural ratio sweeps. The remainder of this paper is organized as follows. Section II outlines related work. Section III discusses the proposed MUX4 LE, the variant used in the fracturable architecture and the design of the hybrid complex logic block. Section IV presents the technology mapping approaches to target the proposed hybrid architecture. Section V shows how we modeled the hybrid complex logic blocks for both the nonfracturable and fracturable architectures in VPR. Section VI discusses our evaluation methodology and provides the evaluation results. Finally, we conclude with final remarks in Section VII.

Recent works have shown that the heterogeneous architectures and synthesis methods can have a significant impact on improving logic density and delay, narrowing the ASIC–FPGA gap. Works by Anderson and Wang with “gated” LUTs [7], then with asymmetric LUT LEs [8], show that the LUT elements present in commercial FPGAs provide unnecessary flexibility. Toward improved delay and area, the macrocell-based FPGA architectures have been proposed [9], [10]. These studies describe significant changes to the traditional FPGA architectures, whereas the changes proposed here build on architectures used in industry and academia [4]. Similarly, and-inverter cones have been proposed as replacements for the LUTs, inspired by and-inverter graphs (AIGs) [6]. Purnaprajna and Ienne [16] explored the possibility of repurposing the existing MUXs contained within the Xilinx Logic Slices [17]. Similar to this work, they use the ABC priority cut mapper as well as VPR for packing, place, and route. However, their work is primarily delay-based showing an average speedup of 16% using only ten of 19 VTR7 benchmarks.

2 RELATED WORK:

In this section, the circuit scheme of conventional RCA, CLA, ETAII in [4] and ACSA in [5] will be analyzed in detail.

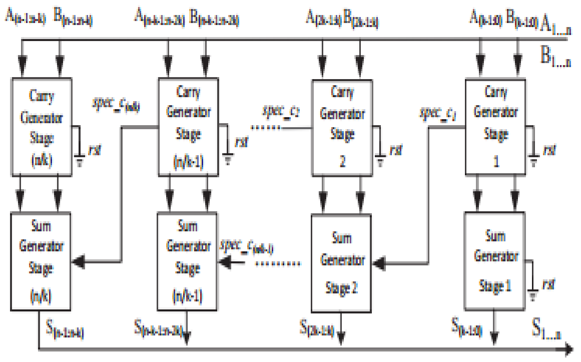
$A_{(n-1:n-k)}$ $B_{(n-1:n-k)}$
$A_{(n-k-1:n-2k)}$ $B_{(n-k-1:n-2k)}$
$A_{(2k-1:k)}$ $B_{(2k-1:k)}$
$A_{(k-1:0)}$ $B_{(k-1:0)}$ $A_{1...n}$
$B_{1...n}$
Carry Generator Stage (n/k)
spec_c(n/k)
Carry Generator Stage (n/k-1)
rst
spec_c2
Carry Generator Stage 2
spec_c1
Carry Generator Stage 1
Sum Generator Stage (n/k)
Sum Generator Stage (n/k-1)
spec_c(n/k-1)
Sum Generator Stage 2
Sum Generator Stage 1
$S_{(n-1:n-k)}$
$S_{(n-k-1:n-2k)}$
$S_{(2k-1:k)}$
$S_{(k-1:0)}$
$S_{1...n}$

Figure 2. Circuit Scheme of ETAII in [4]

As pointed out in [6], RCA is a common type of adder in digital circuit design, in which a series of one-bit full adders are connected in sequence and the higher output depends on lower carry signals. The delay of RCA is □(□) and the critical path starts from the lowest bit to the last one. However, the probability to activate this critical path is very small [6], [7], which provides the foundation to design speculative-based adder in following researches. In order to obtain the carry signal in advance, for CLA, the real value of carry signals for higher computation block is calculated using signal generation method. The critical path can be reduced efficiently. However, the process of carry signal generation is complicated in CLA, which could produce large logic area and will result in a big power consumption. Based on the idea of CLA, ETAII in [4] makes a full use of paralleling calculation and introduced approximation to reduce the power overhead as shown in Fig.2. The adder is divided into several stages. Each of the stages has a carry generator and a sum generator. The output of one stage comes from its sum generator with the previous carry signal. Thus, the critical path

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue 13
October 2017

is composed of one sum stage and carry signal generator. However, the carry signal generator involves only parts of the lower input data, which could cause large error when a wrong prediction happens in the upper stage of the adder. For 32-bits ETAII with 4 bits for each stage, the maximum error magnitude could be 228, which is too large that the adder have little practical value in real application.

In [5], an approximate adder with carry skip technique (ACSA) is proposed based on ETAII, in which a multiplexor is used to choose the carry signal for the sum generator in each stage. Different from ETAII, this adder detects the property of carry propagation in previous stage. When all the bits in previous (□ − 1)□ℎ-stage is in carry propagation mode, it will select the carry signal from (□ − 2)□ℎ-stage. In this way, the error rate of the adder will be decreased. Furthermore, a method for error compensation is also used. However, the error magnitude of this adder is still large. Take 32-bits adder with 4 bits for each stage, the maximum error magnitude could be 220. Meanwhile, the extra multiplexors could consume more area, energy and delay as well.

3 PROJECT DESCRIPTION

MUX4: 4-to-1 Multiplexer Logic Element The MUX4 LE shown in Fig. 1 consists of a 4-to-1 MUX with optional inversion on its inputs that allow the realization of any {2, 3}-input function, some {4, 5}-input functions, and one 6-input function—a 4-to-1 MUX itself with optional inversion on the data inputs. A 4-to-1 MUX matches the input pin count of a 6-LUT, allowing for fair comparisons with respect to the connectivity and intracluster routing.

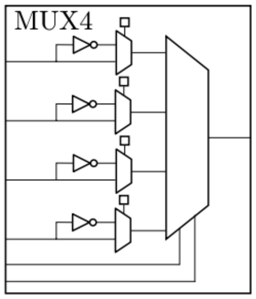

Fig. 1. MUX4 LE depicting optional data input inversions.

spect to the connectivity and intracluster routing. Naturally, any two-input Boolean function can be easily implemented in the MUX4: the two function inputs can be tied to the select lines and the truth table values (logic-0 or logic-1) can be routed to the data inputs accordingly. Or alternately, a Shannon decomposition can be performed about one of the two variables—the variable can then feed a select input. The Shannon cofactors will contain at most one variable and can, therefore, be fed to the data inputs (the optional inversion may be needed). For three-input functions, consider that a Shannon decomposition about one variable produces cofactors with at most two variables. A second decomposition of the cofactors about one of their two remaining variables produces cofactors with at most one variable. Such single-variable cofactors can be fed to the data inputs (the optional inversion may be needed), with the decomposition variables feeding the select inputs. Likewise, functions of more than four inputs can be implemented in the MUX4 as long as Shannon decomposition with respect to any two inputs produce cofactors with at most one input. Observe that input inversion on each select input is omitted as this would only serve to permute the four MUX data inputs. While this could help routability within the CLB's internal crossbar, additional inversions on the select inputs would not increase the number of Boolean functions that are able to map to the MUX4 LE.

Logic Elements, Fracturability, and MUX4-Based Variants

Two families of architectures were created: 1) without fracturable LEs and 2) with fracturable LEs. In this paper, the fracturable LEs refer to an architectural element on which one or more logic functions can be optionally mapped. Nonfracturable LEs refer to an architectural element on which only one logic function is mapped. In the nonfracturable architectures, the MUX4 element shown in Fig. 1 is used together with nonfracturable 6-LUTs. This element shares the same number of inputs as a 6-LUT lending for fair comparison with respect to the input connectivity.

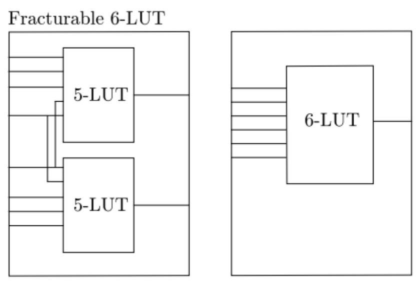

Fig. 2. Fracturable 6-LUT that can be fractured into two 5-LUTs with two shared inputs.

For the fracturable architecture, we consider an eight-input LE, closely matched with the adaptive logic module in recent Altera Stratix FPGA families. A 6-LUT that can be fractured into two 5-LUTs using eight inputs is shown in Fig. 2. Two five-input functions can be mapped into this LE if two inputs are shared between the two functions. If no inputs are shared, two four-input functions can be mapped to each 5-LUT. For the MUX4 variant, Dual MUX4, we use two MUX4s within a single eight-input LE. In the configuration, shown in Fig. 3, the two MUX4s are wired to have dedicated select inputs and shared data inputs. This configuration allows this structure to map two independent (no shared inputs) three-input functions, while larger functions may be mapped dependent on the shared inputs between both functions.

An architecture in which a 4-to-1 MUX (MUX4) is fractured into two smaller 2-to-1 MUXs was first considered. However, since a 2-to-1 MUX's mapping flexibility is quite limited (can only map two-input functions and the three-input 2-to-1 MUX itself), little benefit was added compared with the overheads of making the MUX4 fracturable and poor area results were observed.

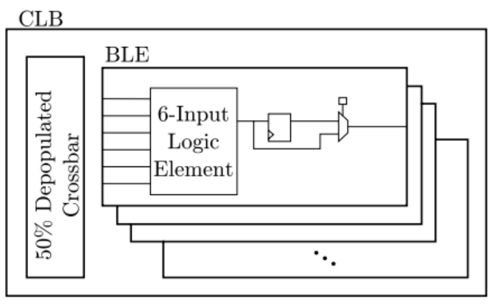

Fig. 4. Hybrid CLB with a 50% depopulated intra-CLB crossbar depicting BLE internals for a nonfracturable (one optional register and one output) architecture.

4 Hybrid Complex Logic Block

A variety of different architectures were considered—the first being a nonfracturable architecture. In the nonfracturable architecture, the CLB has 40 inputs and ten basic LEs (BLEs), with each BLE having six inputs and one output following empirical data in prior work [4]. Fig. 4 shows this nonfracturable CLB architecture with BLEs that contain an optional register. We vary the ratio of MUX4s to LUTs within the ten element CLB from 1:9 to 5:5 MUX4s:6-LUTs. The MUX4 element is proposed to work in conjunction with 6-LUTs, creating a hybrid CLB with a mixture of 6-LUTs and MUX4s (or MUX4 variants). Fig. 4 shows the organization of our CLB and internal BLEs.

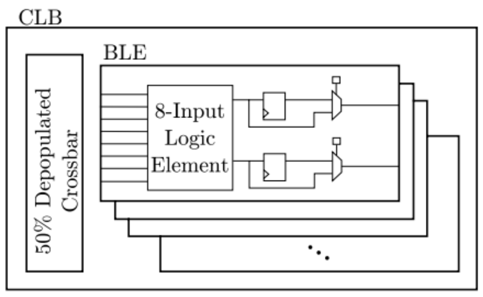

Fig. 5. Hybrid CLB with a 50% depopulated intra-CLB crossbar depicting BLE internals for a fracturable (two optional registers and two outputs) architecture.

For fracturable architectures, the CLB has 80 inputs and ten BLEs, with each BLE having eight inputs and two outputs emulating an Altera Stratix Adaptive-LUT [18]. The same sweep of MUX4 to LUT ratios was also performed. Fig. 5 shows the fracturable architecture with eight inputs to each BLE that contains two optional registers. We evaluate fracturability of LEs versus nonfracturable LEs in the context of MUX4 elements since fracturable LUTs are common in commercial architectures. For example, Altera Adaptive 6-LUTs in Stratix IV and Xilinx Virtex 5 6-LUTs can be fractured into two smaller LUTs with some limitations on inputs.

The crossbar for fracturable architectures are larger than the nonfracturable architectures for two reasons. Due to the virtual increase of LEs, a larger number of CLB inputs are required, which increases crossbar size. Since there are now twice as many outputs from the LEs, these additional outputs need to also be fed back into the crossbar, also increasing its size. Due to this disparity in

crossbar size, fair comparisons cannot be made between fracturable and nonfracturable architectures. Therefore, in this paper, we compare nonfracturable hybrid
CLB architectures to a baseline LUT only nonfracturable architecture and we compare fracturable hybrid CLB architectures to a baseline LUT-only fracturable architecture. Sparse crossbars have been previously studied [19] and in this paper, we model a 50% depopulated crossbar within the CLB for intracluster routing for both nonfracturable and fracturable architectures as compared with the preliminary publication [15] that only modeled a full input crossbar

5 CONCLUSION

We have proposed a new hybrid CLB architecture containing MUX4 hard MUX elements= and shown techniques for efficiently mapping to these architectures. Weighting of MUX4-embeddable functions with our MuxMap technique combined with a select mapping strategy provided aid to circuits with low natural MUX4-embeddable ratios. We also provided analysis of the benchmark suites postmapping, discussing the distribution of functions within each benchmark suite. From our first set of experiments with nonfracturable architectures, area reductions of up to 8% were seen for a 4:6 MUX4:LUT architecture in the CHStone suite with a 2:8 architecture most viable for the VTR suites with ~5% area savings. Our second set of experiments with fracturable architectures showed that the flexibility of a fracturable LUT is very powerful, reducing the impact of the MUX4 LEs, yielding smaller ~2%–3% area savings over the VTR7 and CHStone benchmark suites with less aggressive 2:8 and 1:9 architectures, respectively. Interestingly, we again found that different architectural conclusions can be made based on the benchmark circuits employed in an architecture study [24], since CHStone benchmarks generally preferred more aggressive MUX4:LUT architecture ratios. The CHStone benchmarks being high-level synthesized with LegUp-HLS also showed marginally better performance and this could be due to the way LegUp performs HLS on the CHStone benchmarks themselves. Overall, the addition of MUX4s to FPGA architectures minimally impact FMax and show potential for improving logic-density in nonfracturable architectures and modest potential for improving logicdensity in fracturable architectures.

REFERRENCES

[1] J. Rose et al., “The VTR project: Architecture and CAD for FPGAs from verilog to routing,” in Proc. ACM/SIGDA FPGA, 2012, pp. 77–86.
[2] Y. Hara, H. Tomiyama, S. Honda, and H. Takada, “Proposal and quantitative analysis of the CHStone benchmark program suite for practical C-based high-level synthesis,” J. Inf. Process., vol. 17, pp. 242–254, Oct. 2009.
[3] A. Canis et al., “LegUp: High-level synthesis for FPGA-based processor/accelerator systems,” in Proc. ACM/SIGDA FPGA, 2011, pp. 33–36.
[4] E. Ahmed and J. Rose, “The effect of LUT and cluster size on deepsubmicron FPGA performance and density,” IEEE Trans. Very Large Scale Integr. (VLSI), vol. 12, no. 3, pp. 288–298, Mar. 2004.
[5] J. Rose, R. Francis, D. Lewis, and P. Chow, “Architecture of fieldprogrammable gate arrays: The effect of logic block functionality on area efficiency,” IEEE J. Solid-State Circuits, vol. 25, no. 5, pp. 1217–1225, Oct. 1990.
[6] H. Parandeh-Afshar, H. Benbihi, D. Novo, and P. Ienne, “Rethinking FPGAs: Elude the flexibility excess of LUTs with and-inverter cones,” in Proc. ACM/SIGDA FPGA, 2012, pp. 119–128.
[7] J. Anderson and Q. Wang, “Improving logic density through synthesisinspired architecture,” in Proc. IEEE FPL, Aug./Sep. 2009, pp. 105–111.
[8] J. Anderson and Q. Wang, “Area-efficient FPGA logic elements: Architecture and synthesis,” in Proc. ASP DAC, 2011, pp. 369–375.
[9] J. Cong, H. Huang, and X. Yuan, “Technology mapping and architecture evalution for k/m-macrocell-based FPGAs,” ACM Trans. Design Autom. Electron. Syst., vol. 10, no. 1, pp. 3–23, Jan. 2005.
[10] Y. Hu, S. Das, S. Trimberger, and L. He, “Design, synthesis and evaluation of heterogeneous FPGA with mixed LUTs and macro-gates,” in Proc. IEEE ICCAD, Nov. 2007, pp. 188–193.
[11] I. Kuon and J. Rose, “Measuring the gap between FPGAs and ASICs,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 26, no. 2, pp. 203–215, Feb. 2007.
[12] K. Karplus, “Amap: A technology mapper for selector-based fieldprogrammable gate arrays,” in Proc. 28th ACM/IEE DAC, Jun. 1991, pp. 244–247.
[13] A. Mishchenko, S. Chatterjee, and R. Brayton, “DAG-aware AIG rewriting a fresh look at combinational logic synthesis,” in Proc. 43rd Annu. DAC, 2006, pp. 532–535.
[14] V. Betz and J. Rose, “VPR: A new packing,

placement and routing tool for FPGA research," in Proc. 7th Int. Workshop FPL, 1997, pp. 213–222.
[15] S. A. Chin and J. H. Anderson, "A case for hardened multiplexers in FPGAs," in Proc. FPT, Dec. 2013, pp. 42–49.
[16] M. Purnaprajna and P. Ienne, "A case for heterogeneous technologymapping: Soft versus hard multiplexers," in Proc. IEEE 21st Annu. Int. Symp. FCCM, Apr. 2013, pp. 53–56.

[17] (2011). Virtex-6 FPGA User Guide. [Online]. Available: http://www.xilinx.com
[18] (2011). Stratix IV Device Handbook. [Online]. Available: http://www.altera.com
[19] G. Lemieux and D. Lewis, "Using sparse crossbars within LUT," in Proc. 9th Int. Symp. ACM/SIGDA FPGA, 2001, pp. 59–68.
[20] C. Chiasson and V. Betz, "COFFE: Fully-automated transistor sizing for FPGAs," in Proc. Int. Conf. FPT, Dec. 2013, pp. 34–41.
[21] Predictive Technology Model. [Online]. Available: http://ptm.asu.edu/, accessed 2015.
[22] Altera, private communication, Mar. 2014.
[23] A. Mishchenko, S. Cho, S. Chatterjee, and R. Brayton, "Combinational and sequential mapping with priority cuts," in Proc. IEEE/ACM Int. Conf. ICCAD, Nov. 2007, pp. 354–361.
[24] A. Yan, R. Cheng, and S. J. E. Wilton, "On the sensitivity of FPGA architectural conclusions to experimental assumptions, tools, and techniques," in Proc. 10th Int. Symp. ACM/SIGDA FPGA, 2002, pp. 147–156.