Design and Implementation of FPGA Logic Architectures using Hybrid LUT/Multiplexer

Krosuri Rajyalakshmi[1] **J.Narashima Rao**[2]
rajyalakshmi.krosuri@gmail.com[1] jnarasimharao09@gmail.com[2]
[1]PG Scholar, VLSI, NARASARAOPETA ENGINEERING COLLEGE,YALLAMANDA, GUNTUR, ANDHRA PRADESH.
[2]Associate Professor, Dept of ECE, NARASARAOPETA ENGINEERING COLLEGE, YALLAMANDA, GUNTUR, ANDHRA PRADESH.

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. This paper proposes Multiple hybrid configurable logic block architectures, both nonfracturable and fracturable with varying MUX: LUT logic element ratios. The designed multiplier Logic element is used for the operation of multiplier. Experimental Results are observed using Xilinx ISE 13.2 and are compared with the proposed and the existing techniques.

***Keywords*— FPGA, Multiplexer logic element, Complex logic block, mapping technologies, FIR Fliter.**

I. INTRODUCTION

A field-programmable gate array (FPGA) is a block of programmable logic that can implement multi-level logic functions. FPGAs are most commonly used as separate commodity chips that can be programmed to implement large functions. However, small blocks of FPGA logic can be useful components on-chip to allow the user of the chip to customize part of the chip's logical function. An FPGA block must implement both combinational logic functions and interconnect to be able to construct multi-level logic functions. There are several different technologies for programming FPGAs, but most logic processes are unlikely to implement anti-fuses or similar hard programming technologies.

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. Recently, a number of other works have explored alternative FPGA LE architectures for performance improvement to close the large gap between FPGAs and application-specific integrated circuits (ASICs)

A. LOOKUP TABLES

The basic method used to build a combinational logic block (CLB) also called a logic element in an SRAM-based FPGA is the lookup table (LUT). As shown in Figure, the lookup table is an SRAM that is used to implement a truth table. Each address in the SRAM represents a combination of inputs to the logic element. The value stored at that address represents the value of the function for that input combination. An n-input function requires an SRAM with locations.

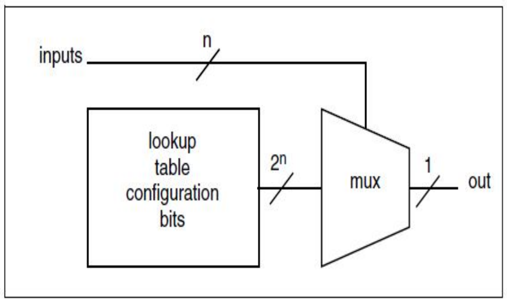


Fig -1 Lookup Tables

Because a basic SRAM is not clocked, the lookup table logic element operates much as any other logic gate as its inputs changes, its output changes after some delay.

B. PROGRAMMING A LOOKUP TABLE

Unlike a typical logic gate, the function represented by the logic element can be changed by changing the values of the bits stored in the SRAM. As a result, the n-input logic element can represent functions (though some of these functions are permutations of each other).

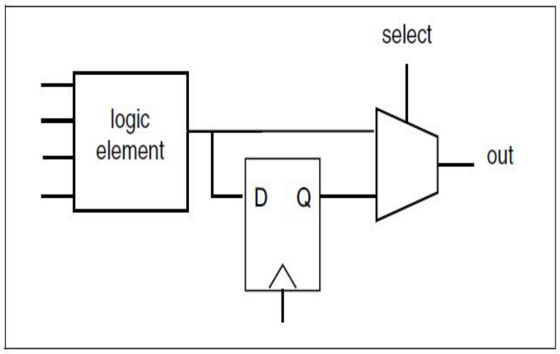


Fig-2 Programming A Lookup Table

A typical logic element has four inputs. The delay through the lookup table is independent of the bits stored in the SRAM, so the delay through the logic element is the same for all functions. This means that, for example, a lookup table-based logic element will exhibit the same delay for a 4-input XOR and a 4-input NAND. In contrast, a 4-input XOR built with static CMOS logic is considerably slower than a 4-input NAND. Of course, the static logic gate is generally faster than the logic element. Logic elements generally contain registers flip-flops and latches as well as combinational logic. A flip-flop or latch is small compared to the combinational logic element (in sharp contrast to the situation in custom VLSI), so it makes sense to add it to the combinational logic element. Using a separate cell for the memory element would simply take up routing resources. The memory element is connected to the output; whether it stores a given value is controlled by its clock and enable inputs.

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 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 six input 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 factorability 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 and VPR, 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 fracturable) that contain a mixture of MUX4 LEs and the traditional LUTs yielding up to 8% area savings.

2) Mapping techniques called Natural Mux and Mux Map 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, and CHStone benchmarks facilitated by LegUp-HLS, the Verilog-to-Routing project showing impact on both area and delay.

Compared with the preliminary publication, we have performed transistor level modeling 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 modeling accuracy. The new transistor-level modeling 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.

II. LITERATURE REVIEW

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, then with asymmetric LUT LEs, 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. These studies describe significant changes to the traditional FPGA architectures, whereas the changes proposed here build on architectures used in industry and academia. Similarly, and-inverter cones have been proposed as replacements for the LUTs, inspired by and-inverter graphs (AIGs).

Purnaprajna and Ienne explored the possibility of repurposing the existing MUXs contained within the Xilinx Logic Slices. Similar to this work, they use the ABC priority cut mapped as well as VPR for packing, place, and route. However, their work is primarily delay-based showing an average speed up of 16% using only ten of 19 VTR7 benchmarks.

In this article, we study the technology mapping problem for a novel field-programmable gate array (FPGA) architecture that is based on k-input single-output programmable logic array- (PLA) like cells, or, k/m-macro cells. Each cell in this architecture can implement a single output function of up to k inputs and up to m product terms. We develop a very efficient technology mapping algorithm, km flow, for this new type of architecture. The experimental results show that our algorithm can achieve depth-optimality on almost all the test cases in a set of 16 Microelectronics Center of North Carolina (MCNC) benchmarks. Furthermore it is shown that on this set of benchmarks, with only a relatively small number of product terms (m≤k+3), the k/m-macro cell-based FPGAs can achieve the same or similar mapping depth compared with the traditional k-input single-output lookup table- (k-LUT-) based FPGAs. We also investigate the total area and delay of k/m-macro cell-based FPGAs and compare them with those of the commonly used 4-LUT-based FPGAs. The experimental results show that k/m-macro cell-based FPGAs can outperform 4-LUT-based FPGAs in terms of both delay and area after placement and routing by VPR on this set of benchmarks. This paper presents experimental measurements of the differences between a 90-nm CMOS field programmable gate array (FPGA) and 90-nm CMOS standard-cell application specific integrated circuits (ASICs) in terms of logic density, circuit speed, and power consumption for core logic. We are motivated to make these measurements to enable system designers to make better informed choices between these two media and to give insight to FPGA makers on the deficiencies to attack and, thereby, improve FPGAs. We describe the methodology by which the measurements were obtained and show that, for circuits containing only look-up table-based logic and flip-flops, the ratio of silicon area required to implement them in FPGAs and ASICs is on average 35. Modern FPGAs also contain "hard" blocks such as multiplier/ accumulators and block memories. We find that these blocks reduce this average area gap significantly to as little as 18 for our benchmarks, and we estimate that extensive use of these hard blocks could potentially lower the gap to below five. The ratio of critical-path delay, from FPGA to ASIC, is roughly three to four with less influence from block memory and hard multipliers. The dynamic power consumption ratio is approximately 14 times and, with hard blocks, this gap generally becomes smaller.

In this paper the new architectural proposals are routinely generated in both academia and industry. For FPGA's to continue to grow, it is important that these new architectural ideas are fairly and accurately evaluated, so that those worthy ideas can be included in future chips. Typically, this evaluation is done using experimentation. However, the use of experimentation is dangerous, since it requires making assumptions regarding the tools and architecture of the device in question. If these assumptions are not accurate, the conclusions from the experiments may not be meaningful. In this paper, we investigate the sensitivity of FPGA architectural conclusions to experimental variations. To make our study concrete, we evaluate the sensitivity of four previously published and well-known FPGA architectural results: lookup-table size, switch block topology, cluster size, and memory size. It is shown that these experiments are significantly affected by the assumptions, tools, and techniques used in the experiments.

III. PROPOSED ARCHITECTURES

A. MUX4: 4-TO-1 MULTIPLEXER LOGIC ELEMENT

The MUX4 LE shown in Fig. 3 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 intra cluster routing. 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-0or logic-1) can be routed to the data inputs accordingly. For three-input functions; consider that 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 produces cofactors with at most one input.

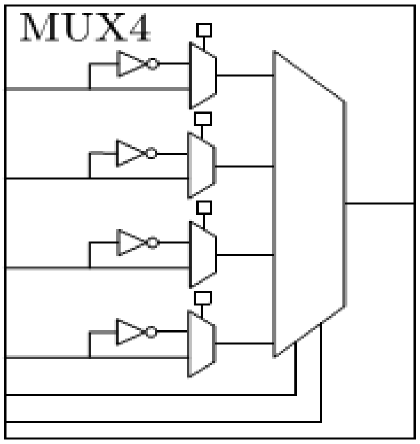


Fig.3. MUX4 LE depicting optional data input inversions

B. Logic Elements, Fracturability, and MUX4-Based Variants.

Two families of architectures were created:

1) Without fracturable LEs

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. 3 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. For the fracturable architecture, we consider an eight-input LE, closely matched with the adaptive logic module in recent Altera Stratix FPGA families. For the MUX4 variant, Dual MUX4, we use two MUX4s within a single eight-input LE. In the configuration, shown in Fig. 4, 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 considered.

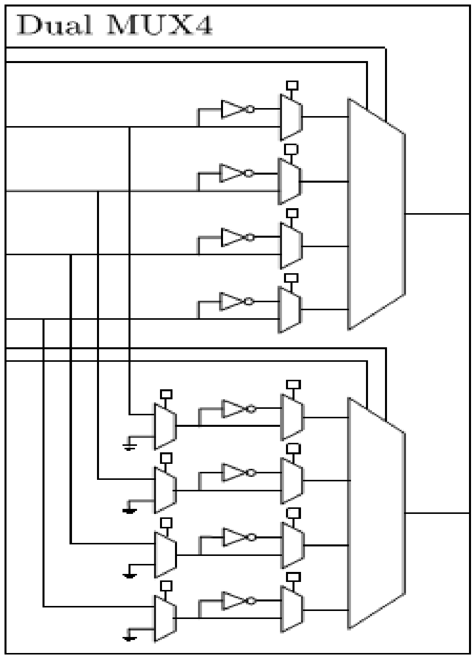


Fig.4. Dual MUX4 LE that utilizes dedicated select inputs and shared data Inputs

C. 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. Fig.5 shows this nonfracturable CLB architecture with BLEs that contain an optional register. We vary the ratio of MUX4s to LUTs within the ten elements 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).

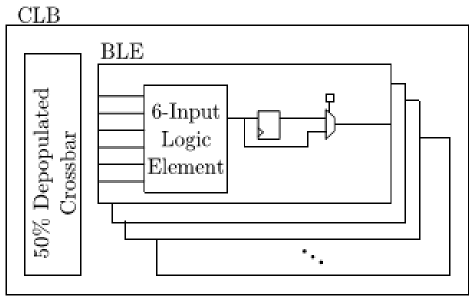


Fig. 5. Hybrid CLB with a 50% depopulated intra-CLB crossbar depicting BLE internals for nonfracturable (one optional register and one output) architecture.

Fig. 6 shows the organization of our CLB and internal BLEs. 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. The same sweep of MUX4 to LUT ratios was also performed. Fig. 4 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.

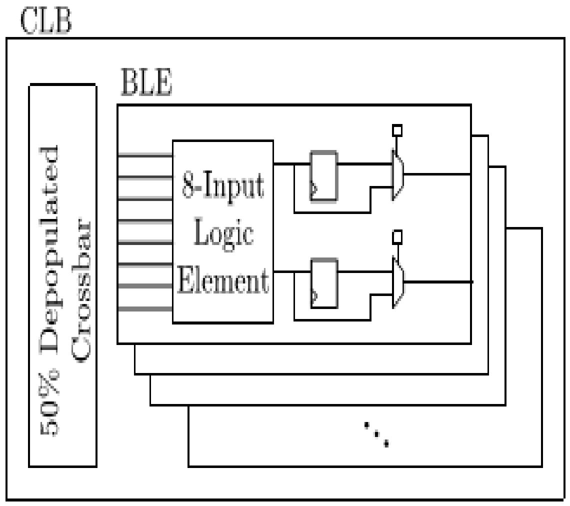


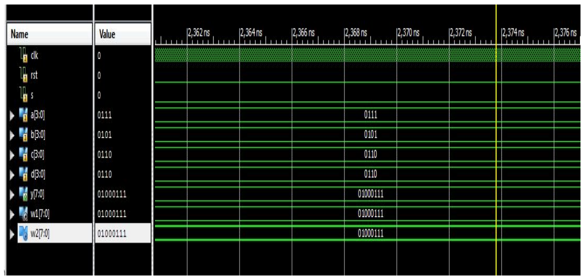
Fig.6. Hybrid CLB with a 50% depopulated intra-CLB crossbar

depicting BLE internals for a fracturable (two optional registers and two outputs) architecture.

IV. RESULTS

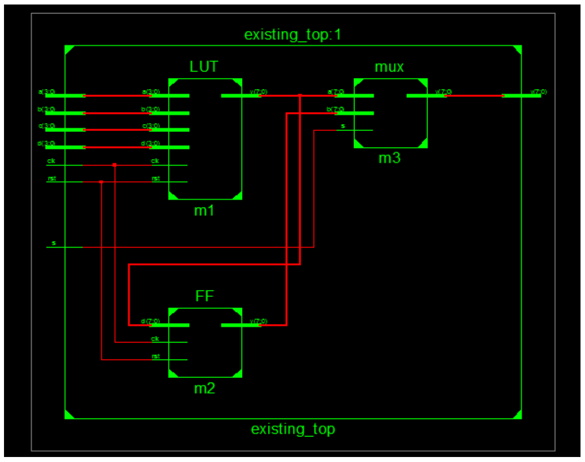
Proposed Result:

SIMULATION

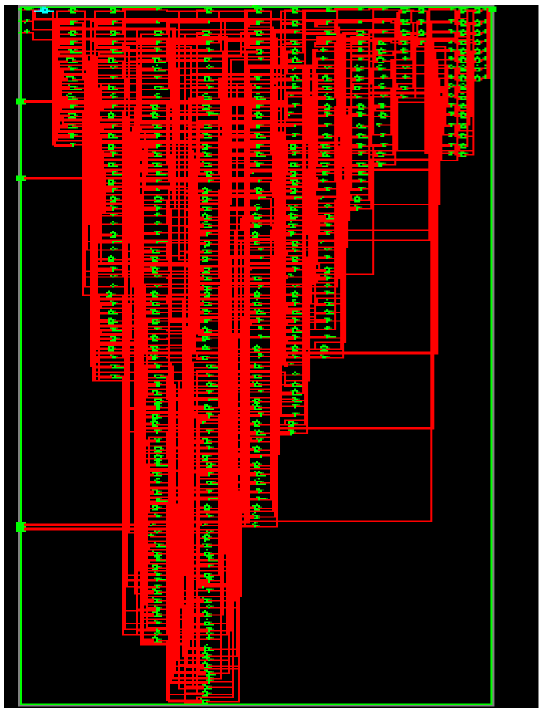


```
Name      Value      |2,362 ns |2,364 ns |2,366 ns |2,368 ns |2,370 ns |2,372 ns |2,374 ns |2,376 ns
clk       0
rst       0
s         0
a[3:0]    0111       0111
b[3:0]    0101       0101
c[3:0]    0110       0110
d[3:0]    0110       0110
y[7:0]    01000111   01000111
w1[7:0]   01000111   01000111
w2[7:0]   01000111   01000111
```

RTL SCHEMATIC

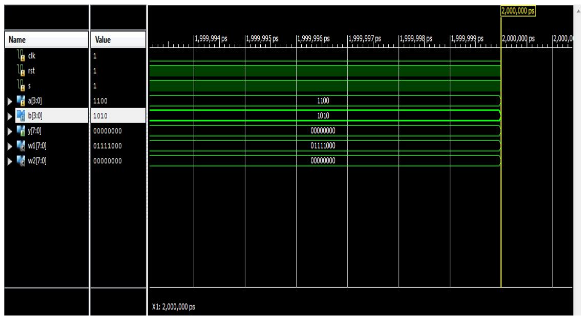


TECHNOLOGY



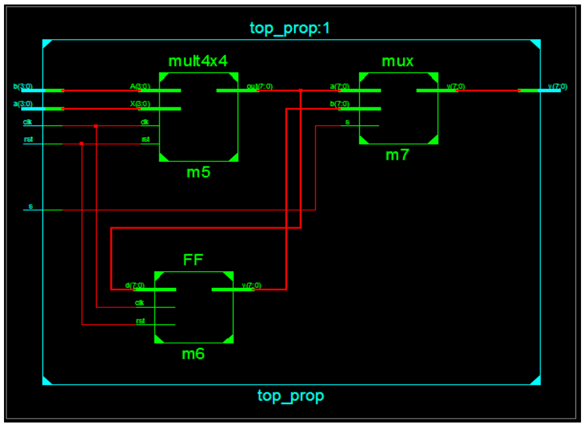
Proposed Result:

SIMULATION

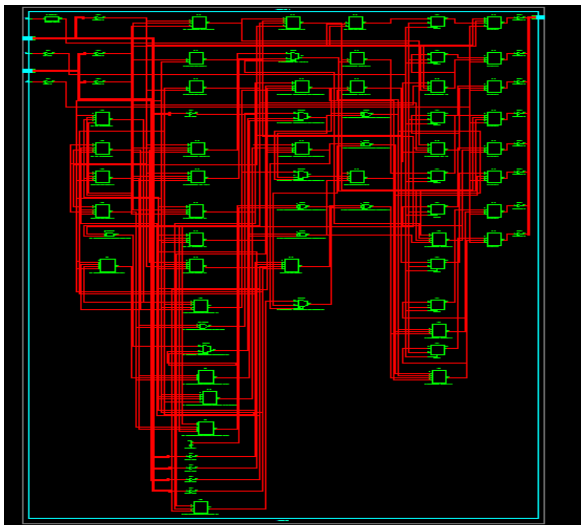


```
Name      Value      |1,999,994 ps |1,999,995 ps |1,999,996 ps |1,999,997 ps |1,999,998 ps |1,999,999 ps |2,000,000 ps
clk       1
rst       1
s         1
a[3:0]    1100       1100
b[3:0]    1010       1010
y[7:0]    00000000   00000000
w1[7:0]   01111000   01111000
w2[7:0]   00000000   00000000
X1: 2,000,000 ps
```

RTL SCHEMATIC



TECHNOLOGY



Comparison Table.

SYSTEM	AREA			DELAY(ns)
	SLICES	LUTS	IOB's	
EXISTING	112	202	27	20.11ns
PROPOSED	2	4	8	5.895ns

V. CONCLUSION

In this paper we proposed a new hybrid CLB architecture containing MUX4 hard MUX elements and shown techniques for efficiently mapping to these architectures. We also provided analysis of the benchmark suites post mapping, discussing the distribution of functions within each benchmark suite.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 logic density in fracturable architecture.

REFERENCES

[1] J. Rose et al., “The VTR project: Architecture and CAD for FPGAs from verilog to routing,” inProc. 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 deep submicron FPGA performance and density,” IEEE Trans. Very Large Scale Integer. (VLSI), vol. 12, no. 3, pp. 288–298, Mar. 2004.

[5] J. Rose, R. Francis, D. Lewis, and P. Chow, “Architecture of field programmable 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 synthesis inspired architecture,” inProc. IEEE FPL, Aug./Sep. 2009, pp. 105–111.

[8] J. Anderson and Q. Wang, “Area-efficient FPGA logic elements: Architecture and synthesis,” inProc. ASP DAC, 2011, pp. 369–375.

[9] J. Cong, H. Huang, and X. Yuan, “Technology mapping and architecture evaluation for k/m-macrocell-based FPGAs,”ACM Trans. Design Autom. Electron. Syst., vol. 10, no. 1, pp. 3–23, Jan. 2005