International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue-17
December 2017

A Novel Design and Implementation of Hybrid Lut/Multiplexer For Fpga Logic Architectures

Y. Priyanka, A. Deepika, A. Vikas
y.priyankareddy@gmail.com[1] , adeepika.745@gmail.com[2] , vikas.ambekar@hotmail.com[3]
Assistant Professor, Dept of ECE, TKR college of Engineering and Technology, Medbowli, Hyderabad, Telangana, India.

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 nonfracturable 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.*

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

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 antifuses 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 [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 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. 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.

LOOKUP TABLES

The fundamental strategy used to build a combinational logic block (CLB) also called a logic element in a SRAM-based FPGA is the lookup table (LUT). As appeared in Figure, the lookup table is a SRAM that is utilized to implement a fact 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. A n-input function requires a SRAM with locations.

Since a fundamental SRAM isn't clocked, the lookup table logic element works much as some other logic gate as its inputs changes, its yield changes after some delay.

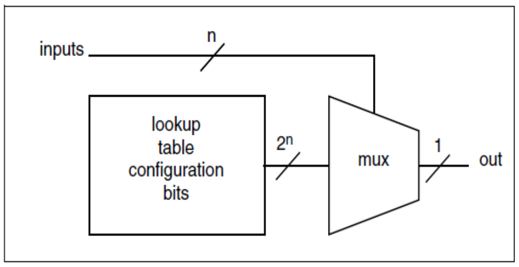


Fig .1: Lookup Tables

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).

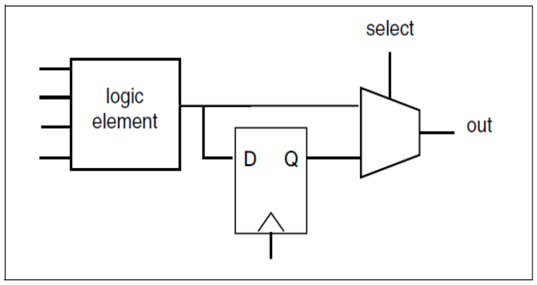


Fig.2: Programming a Lookup Table

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 wastefulness of LUTs implementing MUXs, think about that as a six information LUT (6-LUT) is essentially a 64-to-1 MUX (to select 1 of 64 truth-table lines) and 64-SRAM arrangement cells, yet it can only realize a 4-to-1 MUX (4 data+2 select=6 inputs). In this paper, we exhibit a six-input LE in view of a 4-to-1 MUX, MUX4, that can realize a subset of six-input Boolean logic capacities, and another hybrid complex logic block (CLB) that contains a blend of MUX4s and 6-LUTs. The proposed MUX4s are small contrasted and a 6-LUT (15% of 6-LUT area), and can efficiently delineate {2, 3}- input functions and some {4, 5, 6}- input functions. What's more, we explore fracturability of Les the ability to split the LEs into multiple smaller elements in both LUTs and MUX4s to build logic thickness. The proportion of LEs that should be LUTs versus MUX4s is also explored toward upgrading logic thickness for both nonfracturable and fracturable FPGA structures. To facilitate the engineering exploration, we developed a CAD flow for mapping into the proposed hybrid CLBs, made utilizing ABC and VPR, and portray technology mapping procedures that energize the selection of logic works that can be inserted into the MUX4 elements. The primary commitments 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 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.

Contrasted with the preliminary publication, we have performed transistor level modeling of the MUX4 LE, additionally examined the fracturable structures, and unified the open source tool-flow from C through LegUp-HLS to the VTR flow. Meager crossbars (versus full crossbars in the past work) have also been included in our CLBs, expanding modeling exactness. The new transistor-level modeling of the MUX4 also gives more exact results as contrasted and the past work. Results have also been expanded with the inclusion of timing results and larger architectural proportion clears.

II. LITERATURE REVIEW

Recent works have demonstrated that the heterogeneous designs and combination techniques can significantly affect enhancing logic density and delay, narrowing the ASIC– FPGA gap . Works by Anderson and Wang with "gated" LUTs, at that point with asymmetric LUT LEs, demonstrate that the LUT elements show in commercial FPGAs give superfluous flexibility. Toward enhanced delay and area, the macrocell-based FPGA designs have been proposed. These investigations portray huge changes to the traditional FPGA structures, though the progressions proposed here build on models utilized as a part of industry and the scholarly world. Similarly, and-inverter cones have been proposed as replacements for the LUTs, propelled by and-inverter diagrams (AIGs).

Purnaprajna and Ienne explored the possibility of repurposing the current MUXs contained inside the Xilinx Logic Slices. Similar to this work, they utilize the ABC need cut mapper and also VPR for pressing, place, and course. Be that as it may, their work is primarily delay-based demonstrating a normal speed up of 16% utilizing only ten of 19 VTR7 benchmarks.

In this article, we contemplate the technology mapping problem for a novel field-programmable gate exhibit (FPGA) design that is based onk-input single-yield programmable logic cluster (PLA-) like cells, or, k/m-macrocells. Every cell in this design can implement a single yield functions of up to k inputs and up to m item terms. We develop an exceptionally effective technology mapping algorithm, km flow, for this new sort of engineering. The experimental results demonstrate that our algorithm can accomplish profundity optimality on almost all the experiments in an arrangement of 16 Microelectronics Center of North Carolina (MCNC) benchmarks. Moreover it is demonstrated that on this arrangement of benchmarks, with only a relatively small number of item terms (m≤k+3), the k/m-full scale cell-based FPGAs can accomplish the same or similar mapping profundity contrasted and the traditional k-input single-yield lookup table-(k-LUT-) based FPGAs. We also investigate the total area and delay of k/m-large scale cell-based FPGAs and contrast them and those of the commonly utilized 4-LUT-based FPGAs. The experimental results demonstrate that k/m-full scale cell-based FPGAs can outflank 4-LUT-based FPGAs as far as both delay and area after placement and steering by VPR on this arrangement of benchmarks.

This paper presents experimental estimations of the contrasts between a 90-nm CMOS field programmable gate exhibit (FPGA) and 90-nm CMOS standard-cell application particular coordinated circuits (ASICs) as far as logic thickness, circuit speed, and power consumption for center logic. We are roused to make these estimations to enable framework fashioners to settle on better educated decisions between these two media and to offer knowledge to FPGA creators on the lacks to assault and, in this manner, enhance FPGAs. We depict the methodology by which the estimations were gotten and demonstrate that, for circuits containing only look-up table-based logic and flip-flops, the proportion of silicon area required to implement them in FPGAs and ASICs is by and large 35. Current FPGAs also contain "hard" blocks, for example, multiplier/accumulators and block recollections. We find that these blocks decrease this normal area gap significantly to as meager as 18 for our benchmarks, and we evaluate that broad utilization of these hard blocks could potentially lower the gap to below five. The proportion of critical-way delay, from FPGA to ASIC, is roughly three to four with less influence from block memory and hard multipliers. The dynamic power consumption proportion is approximately 14 times and, with hard blocks, this gap generally ends up noticeably smaller.

In this paper the new architectural proposals are routinely created in both scholarly community and industry. For FPGA's to keep on growing, it is essential that these new architectural thoughts are fairly and accurately evaluated, so those commendable thoughts can be included in future chips. Typically, this evaluation is finished utilizing experimentation. In any case, the utilization of experimentation is perilous, since it requires making assumptions in regards to the tools and design of the gadget being referred to. In the event that these assumptions are not precise, the conclusions from the trials may not be meaningful. In this paper, we investigate the affectability of FPGA architectural conclusions to experimental varieties. To influence our examination to solid, we evaluate the affectability of four previously published and well-known FPGA architectural results: lookup-table size, switch block topology, cluster size, and memory measure. It is demonstrated that these examinations are significantly influenced by the assumptions, tools, and procedures utilized as a part of the trials.

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 inputpin count of a 6-LUT, allowing for fair comparisons with respect 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.

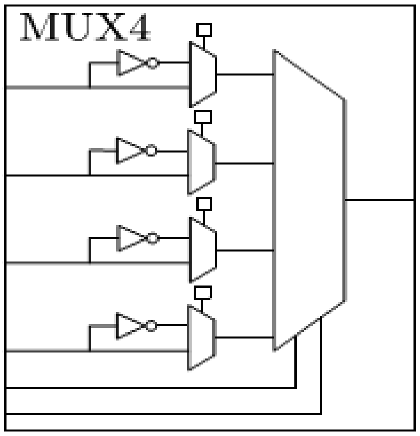


Fig.3: MUX4 LE depicting optional data input inversions

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

Two families of architectures were created:

1) Without fracturable LEs 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. 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.

Fracturable 6-LUT

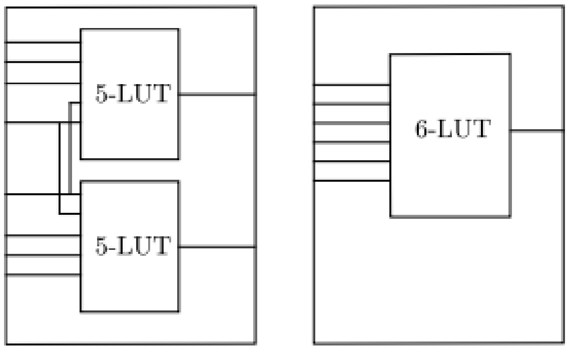


Fig. 4: Fracturable 6-LUT that can be fractured into two 5-LUTs with two shared inputs.

A 6-LUT that can be fractured into two 5-LUTs using eight inputs is shown in Fig. 4. 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. 5, 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.

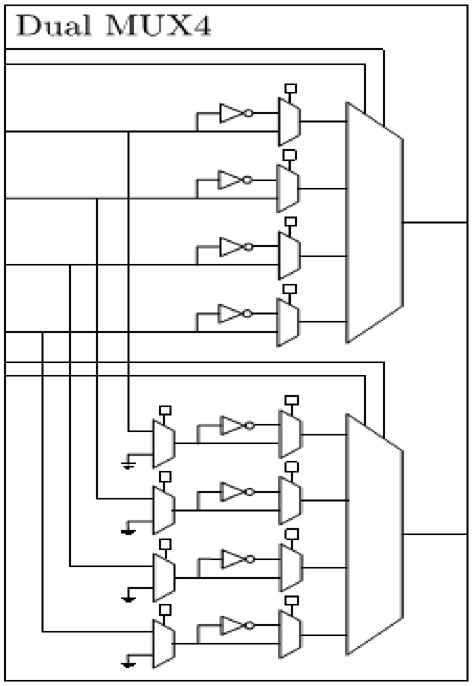


Fig.5: Dual MUX4 LE that utilizes dedicated select inputs and shared data Inputs

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 following empirical data in prior work [4]. Fig. 6 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. 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 [18]. The same sweep of MUX4 to LUT ratios was also performed.

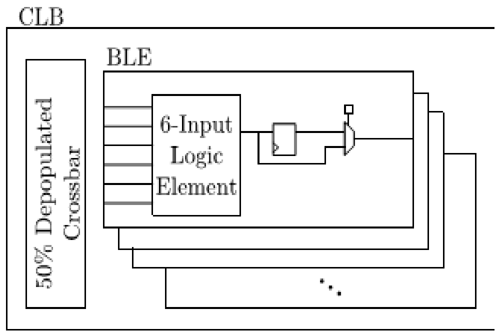


Fig. 6: Hybrid CLB with a 50% depopulated intra-CLB crossbar depicting BLE internals for nonfracturable (one optional register and one output) architecture.

Fig. 7 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 crossbars 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.

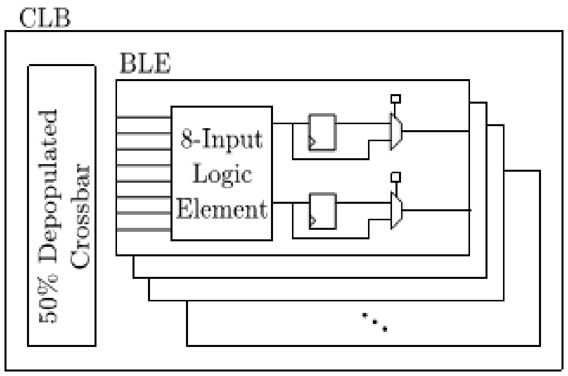


Fig.7: Hybrid CLB with a 50% depopulated intra-CLB crossbar depicting BLE internals for a fracturable (two optional registers and two outputs) architecture.

D. Area Modeling

1) MUX4 Logic Element: Initial estimates of the MUX4 element demonstrated that the MUX4 is~10% the area of a 6-LUT overall. A 4-to-1 MUX can be realized with three 2-to-1 MUXs. Consequently, the MUX4 element contains seven 2-to-1 MUXs, four SRAM cells, and four inverters altogether (see Fig. 3). The optional inversion utilizes the four SRAM cells, though whatever is left of the LE setup is performed through steering. Also, the profundity of the MUX tree is halved contrasted and the 6-LUT, which has six 2-to-1 MUXs on its longest ways. Conservatively, assuming consistent pass transistor measuring and that the area of a 2-to-1 MUX and six transistor SRAM cell are roughly equivalent, the MUX4 element has (1/16)th the SRAM area and(1/8)th the MUX area of a 6-LUT.

These estimates were revised using transistor level modelling of the circuit blocks. Transistor-level optimization of the constituent circuit blocks of an FPGA requires an understanding of the optimal area-delay tradeoffs for each individual circuit block. This requires extracting a representative critical path, which is a path whose composition of blocks and topology will be similar to the critical path of a specific design. Extracting the representative critical path allows us to judge to what extent each individual block is timing critical, which thus establishes an area-delay tradeoff goals for each block. This is in line with the transistor-level optimization tool developed previously [20]. We use the results of prior work [20] to establish the optimal area-delay tradeoff for 6-LUTs in a conventional island-style FPGA architecture with typical architectural parameters. The resulting 6-LUT delay serves as a point of reference for optimization for the circuits considered in this paper: in the interest of maximizing area reduction while allowing performance to be maintained (ignoring the differences in cell counts between mapping to a conventional LUT and the LEs proposed in this paper), we attempt to match the delay of a 6-LUT while minimizing the area of each of the variants of the MUX4 circuits.

Transistor level modelling and optimizations were based on a predictive 22-nm high performance process [21], while the area model presented in prior work [20] was used to estimate the area of various circuit structures. With this methodology, we determined an area-delay optimal 6-LUT has an area of 930 minimum-width transistors, and a worst-case delay of 261 ps. For the MUX4 cell and Dual MUX4 cell, a minimum area and minimum delay cell was created. The minimum area MUX4 cell has an area of 95 minimumwidth transistors and a delay of 204 ps; all transistors were minimum-width in this case, and as the minimum area solution for this circuit was able to meet (and improve upon) the worst-case delay target of a 6-LUT. Similarly, the Dual MUX4 cell has an area of 249 minimum-width transistors while meeting the worst-case delay requirement. However, we chose to use the minimum delay design for both the MUX4 and Dual MUX4 elements for the rest of the study as there is not a significant increase in area over the minimum area design. Although the modelling was performed in the 22-nm process, the standard VPR architecture we use has all parameters (routing delays, crossbar delays, and so on) scaled to a 40-nm process.

In this standard VPR architecture, parameters are compounded from a multitude of sources, some also in other lithographic processes, and subsequently scaled to 40- nm. Likewise, we linearly scale our delays by comparing the delays of our 22-nm 6-LUT (261 ps) and the 6-LUT in the standard architecture (398 ps). The delays for each design after scaling to 40-nm are shown in Table I.

TABLE I
LE TRANSISTOR MODELS WITH AREA GIVEN IN MINIMUM-WIDTH TRANSISTOR AREA AND DELAYS SCALED FOR A 40-nm PROCESS

Logic Element Design	Area (MWTA)	% 6-LUT Area	Scaled Max. Delay (ps)
MUX4 Min. Area	95	10.2%	311
MUX4 Min. Delay	108	11.6%	248
Dual *MUX4* Min. Area	249	26.7%	398
Dual *MUX4* Min. Delay	255	27.4%	375
6-LUT	930	100.0%	398

FPGA Area Model:

Although determining the area of a MUX4 element relative to a 6-LUT is important, we need to also examine global FPGA area considering the number of CLB tiles, area overheads within the CLB and routing area per CLB. Throughout this paper, global FPGA area was estimated assuming that, per tile, 50% of the area is intercluster and intracluster routing, 30% of the area is used for LUTs, and 20% for registers and other miscellaneous logic, followingAnderson and Wang [7] and a private communication [22]. It is important to note that this 50%–30%–20% model is an estimate based on a traditional full FPGA design where-by the routing and internal CLB crossbars are optimized toward 6-LUTs. Production of an optimized FPGA utilizing our new MUX4 elements would surely change said model. However, optimizing the entire routing architecture toward our MUX4 variants, measuring the routing architecture, and closing the loop by creating a more accurate model is out of the scope of this work.

Area calculation:

Using this model, we can make some observations about the hybrid CLB architecture. The 30% that normally would account for ten 6-LUT LEs within the tile is now split between the smaller MUX4 elements and 6-LUTs. For example, in a 3 MUX4:7 6-LUT architecture, the area relative to the reference area model can be estimated by deducing the Logic Change% = (3 × 0.116 + 7)/10 (3 MUX4s each at 0.116 the area of a 6-LUT and 7 6-LUTs), and multiplying Logic Change% × 30% = 22% of total FPGA area. If routing and miscellaneous area were held constant, our overall architecture area is Area3:7 = 50% + 20% + 22% = 92% of the reference area—8% area savings. However, this is the maximum area savings and it can only be realized by circuits that have a natural (i.e., inherent) MUX4:LUT ratio greater than or equal to the architecture ratio. In addition, since any function that can be mapped to a MUX4 element can also be mapped into a 6-LUT, all excess MUX4 functions can be mapped to 6-LUTs. If the natural MUX4: LUT ratio of the circuit is less than the architecture ratio, additional CLBs will be required to supply more LUTs. In addition, the number of CLBs may also increase during CLB packing (CLB Change%) and routing demand may increase post placement and routing (Routing Change%). In general, the model used to estimate area relative to the baseline 6-LUT only architecture (nonfracturable or fracturable) is as follows:

$$Area\% = CLBChange\% \times (50\% \times RoutingChange\% + 30\% \times LogicChange\% + 20\%). \quad (1)$$

Using this model, it is useful to calculate how many additional CLBs can be tolerated for our new architectures. Again, consider a 3:7 MUX4:LUT architecture. Disregarding packing, placement, and routing effects

$$\begin{aligned} NumCLB_{3:7} &\leq 1/Area_{3:7} \times NumCLB_{LUT} \\ &\leq 1.08 \times NumCLB_{LUT}. \end{aligned} \tag{2}$$

This means that an area win can only be achieved if the number of CLBs needed to implement circuits in a hybrid 3:7 architecture is less than 1.08× the number needed for a traditional LUT-only architecture. Similarly, the calculation is performed for the fracturable architecture with the larger Dual MUX4 element. A full table for all architectures showing the architectural minimum area and tolerable CLB increase is shown in Table II.

TABLE II
PER ARCHITECTURE, MINIMUM RELATIVE ARCHITECTURAL PERCENTAGE AREA, AND TOLERABLE PERCENTAGE CLB INCREASE ASSUMING CONSTANT ROUTING DEMAND

	Arch. 1:9		Arch. 2:8		Arch. 3:7		Arch. 4:6		Arch. 5:5	
	% Area	% CLB	% Area	% CLB	% Area	% CLB	% Area	% CLB	% Area	% CLB
Non-Fracturable	97.3	102.7	94.7	105.6	92.0	108.6	89.4	111.9	86.7	115.3
Fracturable	97.8	102.2	95.6	104.6	93.5	107.0	91.3	109.5	89.1	112.2

IV. TECHNOLOGY MAPPING USING ABC

ABC [13] was used for technology mapping, with modifications that allow for MUX4-embeddable function identification and MUX2-embeddable function indentification in the case of fracturable MUX4s and custom mapping. The internal data structure used within the ABC is an AIG, where the logic circuit is represented using 2-input AND gates with inverters. Priority Cuts mapping in ABC (invoked with the if command) [23] was modified to perform our custom technology mapping. This mapper traverses the AIG from primary inputs to primary outputs finding intermediate mappings for internal nodes and finally the primary outputs, using a dynamic programming approach. The priority cuts mapper performs multiple passes on the AIG to find the best cut per node. For depth-oriented mapping, the mapper first prioritizes mapping depth then optimizes for area discarding cuts whose selection would increase the overall depth of the mapped network. Based on this standard mapper, two mapper variants were produced and evaluated. The first variant, NaturalMux, evaluates and identifies internal functions that are MUX4-embeddable, agnostic of the target architecture; i.e., this flow uses the default priority cuts mapping and performs a postprocessing step to identify MUX4-embeddable functions. From this mapping, we can evaluate what area savings are possible without any mapper changes. The second variant MuxMap, area-weights the MUX4-embeddable cuts relative to 6-LUT cuts, thereby establishing a preference for selection/creation of MUX4-embeddable solutions.

In this paper, each of the select input(s) and data inputs to the MUX4 element is classified by the mapper on a pin-by-pin basis, so that much more accurate packing can be performed in the VPR.

A. Natural Mux

Natural Mux mapping invokes the standard priority cuts mapper. Following mapping, we use the preceding approach to determine if the LUT logic functions in the mapping are MUX4-embeddable. This is needed so we can identify which LUTs are MUX4-embeddable in the subsequent packing stage.

B. Mux Map

In default ABC technology mapping, each LUT has a unit area of 1. In our Mux Map approach, we use a lower weight for the cases where logic functions are MUX4-embeddable. Following the area model where 50% of an FPGA tile area is routing, 30% is 6-LUTs and 20% is miscellaneous circuitry (FFs + other), we can derive the weight of a MUX4 element versus a 6-LUT. Dividing an FPGA tile into ten subtiles that contain a single 6-LUT plus the 6-LUT's associated routing and miscellaneous circuitry, the 6-LUT or logic portion of a subtile is 3% and the miscellaneous circuitry and routingis 7% of a complete tile. Recall from Section III-D that a MUX4 element consumes 11.6% of the area of a 6-LUT. Therefore, the area of a subtile with a MUX4 is 7.45% of the entire tile, i.e., 7% routing and miscellaneous circuitry area plus 11.6% × 3% logic area. The area ratio of a subtile with a MUX4 versus a subtile with a 6-LUT would be roughly 7.34%/10% = 0.734% (assuming the routing and other circuitry is held constant). Following this reasoning, we weight MUX4s conservatively at 80% of a 6-LUT during technology mapping. Experimental results shown in Section VI-A show that this is a reasonable choice.

C. Select Mapping

Depending on the circuit, Natural Mux or Mux Map may be preferred. In select mapping, the circuit is first mapped using NaturalMux. Following from the discussion in Section III-D, we know that if a circuit's MUX4:LUT ratio is higher than the architectural ratio, maximum area reductions are realized. Therefore, if the natural ratio of the circuit is higher than our target architectural ratio, we use this mapping. Otherwise, if the natural ratio is lower than the architectural ratio, we rerun the mapping with the Mux Map mapper to encourage the selection of more MUX4-embeddable LEs. Note that the technology mapping run-time is a small fraction of that required for placement and routing.

V. MODELING USING VPR

VPR was utilized to perform architectural evaluation. The standard ten 6-LUT CLB design in 40-nm included with the VPR dissemination was utilized for baseline modeling. The hybrid CLBs appeared in Figs. 3 and 4 were modeled utilizing the XML-based VPR architectural language. The piece from the design file for the physical block solidified MUX4 element, this code indicates a MUX4 as a six-input one-yield black box to the VPR. What's more, since all MUX4s can also be mapped to the 6-LUTs, an additional mode was added to the 6-LUT physical block.

The architectures with CLBs having MUX4:LUT ratios from 1:9 to 5:5 were created from the baseline 40-nm architectures with delays obtained through circuit simulations of the MUX4 variants. Importantly, we made minor modifications to the VPR packing algorithm [1] itself, so that the MUX4 netlist elements are preferred to be packed into the MUX4 LEs in the architecture (while limiting packing MUX4 netlist elements into LUTs). The modifications involved changing the attraction function during the CLB packing. One change was to ensure that the logic functions that were MUX4 embeddable were preferentially packed into a physical MUX4 element and not into an LUT. Another was to apply a negative weight on MUX4-embeddable functions when the current CLB's physical MUX4 elements are all occupied—also preventing MUX4-embeddable functions from being placed into the LUTs. Without this, the MUX4 netlist elements might needlessly consume LUTs, which should be reserved, where possible, for those netlist elements that demand their flexibility. This becomes doubly important for fracturable architectures, since their packing problem is more complex. Without this modification, a significant CLB usage increase was observed across all benchmark sets.

VI. EXPERIMENTAL EVALUATION

To determine the benefits of these new architectures, evaluation was performed for each architecture using multiple benchmark suites and mapping schemes. Two benchmarks suites were used to evaluate our hybrid architectures: 1) VTR7 [1] and 2) CHStone [2]. Over the nonfracturable and fracturable architecture families, two sets of experiments were performed using the NaturalMux and Mux Map mapping schemes.

As the number of LEs grows, packing, placement, and routing effects play a greater role in the final circuit area. In the remainder of this paper, a weighting of 80% was chosen for the Mux Map as this gave a good balance of additional MUX4-embeddable LEs. Lower weightings result in many additional LEs, exacerbating the losses due to packing, placement, and routing. The left-hand side of Table III shows the projected area results for NaturalMux mapping as well as the baseline statistics of each benchmark in the two benchmark suites.

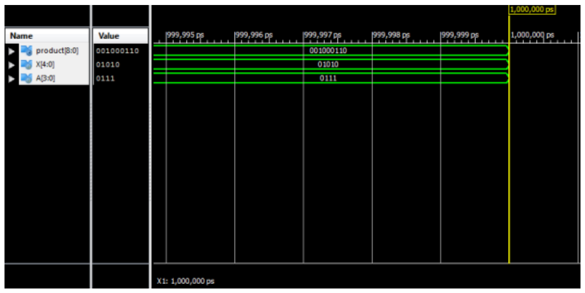
TABLE III
POSTMAPPING AREA ESTIMATE FOR VTR7 AND CHSTONE BENCHMARKS ASSUMING COMPLETE CLB PACKING AND NO INCREASE IN ROUTING DEMAND

		Baseline & NaturalMux							MuxMap - 0.8						
		# LEs	% MUX4	Arch. M:L Ratio 1:9	2:8	3:7	4:6	5:5	% LE Change	% MUX4	Arch. M:L Ratio 1:9	2:8	3:7	4:6	5:5
vtr	bgm	31480	25	97.3	94.7	99.0	112.2	130.6	1.4	29	98.7	96.0	95.3	108.0	125.8
	blob_merge	6036	22	97.3	94.7	102.6	116.3	135.4	1.0	30	98.3	95.7	93.0	104.8	122.1
	boundtop	2932	43	97.3	94.7	92.0	89.4	98.5	0.3	50	97.6	95.0	92.3	89.6	87.0
	ch_intrinsics	382	29	97.3	94.7	94.0	106.5	124.0	0.0	29	97.3	94.7	94.0	106.5	124.0
	diffeq1	466	33	97.3	94.7	92.0	99.4	115.8	0.9	43	98.2	95.5	92.8	90.2	99.0
	diffeq2	317	32	97.3	94.7	92.0	101.0	117.7	1.3	34	98.6	95.9	93.2	99.2	115.5
	LU8PEEng	21927	43	97.3	94.7	92.0	89.4	99.5	0.6	46	97.9	95.3	92.6	89.9	93.8
	mkDelayWorker32B	5467	43	97.3	94.7	92.0	89.4	98.3	1.3	47	98.6	95.9	93.2	90.5	92.6
	mkPktMerge	232	74	97.3	94.7	92.0	89.4	86.7	0.4	76	97.8	95.1	92.4	89.8	87.1
	mkSMAdapter4B	1997	32	97.3	94.7	92.0	101.5	118.1	1.2	37	98.5	95.8	93.1	94.6	110.2
	or1200	2955	38	97.3	94.7	92.0	91.9	107.0	2.9	50	100.2	97.5	94.8	92.0	89.6
	raygentop	1995	31	97.3	94.7	92.0	102.8	119.7	2.9	51	100.2	97.4	94.7	92.0	89.3
	sha	2215	35	97.3	94.7	92.0	96.9	112.9	0.0	36	97.4	94.7	92.1	95.2	110.8
	stereovision0	11232	76	97.3	94.7	92.0	89.4	86.7	2.8	86	100.1	97.4	94.7	91.9	89.2
	stereovision1	10059	69	97.3	94.7	92.0	89.4	86.7	0.2	73	97.5	94.9	92.2	89.6	86.9
	stereovision2	28596	52	97.3	94.7	92.0	89.4	86.7	1.3	56	98.6	95.9	93.2	90.5	87.9
	stereovision3	173	51	97.3	94.7	92.0	89.4	86.7	0.6	55	97.9	95.2	92.6	89.9	87.2
	Geomean	-	40	97.3	94.7	93.1	96.3	105.3	1.1	46	98.4	95.8	93.3	94.2	98.9
Select Geomean		-	-	97.3	94.7	92.4	93.8	98.6							
CHStone	adpcm	13797	65	97.3	94.7	92.0	89.4	86.7	1.3	69	98.6	95.9	93.2	90.6	87.9
	aes	14265	52	97.3	94.7	92.0	89.4	86.7	2.7	61	99.9	97.2	94.5	91.8	89.1
	blowfish	15584	56	97.3	94.7	92.0	89.4	86.7	0.0	60	97.4	94.7	92.0	89.4	86.7
	dfadd	6523	56	97.3	94.7	92.0	89.4	86.7	1.3	60	98.6	95.9	93.2	90.5	87.8
	dfdiv	7292	54	97.3	94.7	92.0	89.4	86.7	2.6	62	99.9	97.2	94.5	91.7	89.0
	dfmul	4381	51	97.3	94.7	92.0	89.4	86.7	1.8	55	99.1	96.4	93.7	91.0	88.3
	dfsin	19714	61	97.3	94.7	92.0	89.4	86.7	2.2	68	99.4	96.7	94.0	91.3	88.6
	gsm	12580	59	97.3	94.7	92.0	89.4	86.7	2.1	65	99.4	96.7	94.0	91.3	88.5
	jpeg	36400	56	97.3	94.7	92.0	89.4	86.7	2.7	64	100.0	97.3	94.5	91.8	89.1
	mips	4736	55	97.3	94.7	92.0	89.4	86.7	1.9	59	99.2	96.5	93.8	91.1	88.4
	motion	5180	49	97.3	94.7	92.0	89.4	88.7	2.6	55	99.9	97.1	94.4	91.7	89.0
	sha	13607	52	97.3	94.7	92.0	89.4	86.7	3.4	60	100.7	98.0	95.2	92.5	89.7
	Geomean	-	55	97.3	94.7	92.0	89.4	86.9	2.0	61	99.3	96.6	93.9	91.2	88.5
Select Geomean		-	-	97.3	94.7	92.0	89.4	86.9							

VII. RESULTS

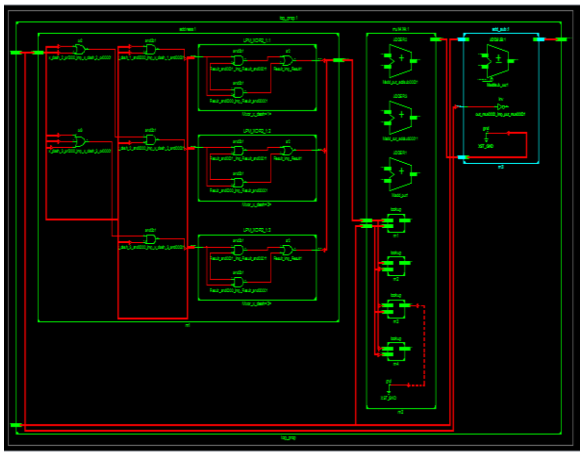
The composed Verilog HDL Modules have effectively recreated and confirmed utilizing Isim Simulator and orchestrated utilizing Xilinxise13.2.

SIMULATION RESULTS:

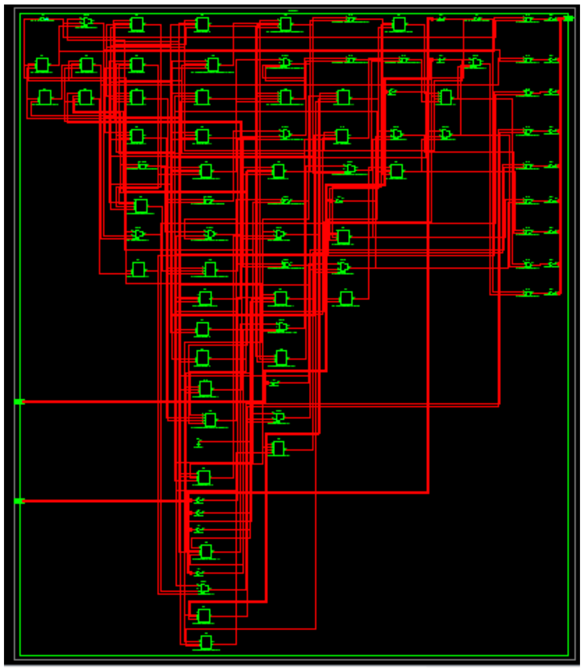


Name	Value
product[8:0]	001000110
X[4:0]	01010
A[3:0]	0111

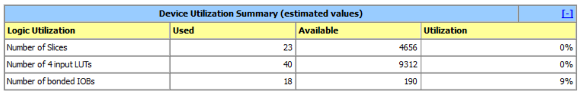
RTL schematic:



Technology Schematic:

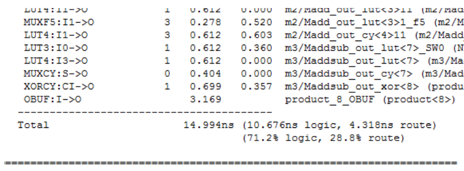


Design summary:



Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	23	4656	0%
Number of 4 input LUTs	40	9312	0%
Number of bonded IOBs	18	190	9%

Timing Report:



```
   LUT4:I1->O            1   0.612   0.000  m2/Madd_out_lut<3>11 (m2/Madd_out_lut<3>1)
   MUXF5:I1->O           3   0.278   0.520  m2/Madd_out_lut<3>1_f5 (m2/Madd_out_lut<3>)
   LUT4:I1->O            3   0.612   0.603  m2/Madd_out_cy<4>11 (m2/Madd_out_cy<4>)
   LUT3:I0->O            1   0.612   0.360  m3/Maddsub_out_lut<7>_SW0 (N27)
   LUT4:I3->O            1   0.612   0.000  m3/Maddsub_out_lut<7> (m3/Maddsub_out_lut<7>)
   MUXCY:S->O            0   0.404   0.000  m3/Maddsub_out_cy<7> (m3/Maddsub_out_cy<7>)
   XORCY:CI->O           1   0.699   0.357  m3/Maddsub_out_xor<8> (product_8_OBUF)
   OBUF:I->O                 3.169          product_8_OBUF (product<8>)
  ----------------------------------------
  Total                    14.994ns (10.676ns logic, 4.318ns route)
                                    (71.2% logic, 28.8% route)

=========================================================================
```

VIII. CONCLUSION

In this paper we proposed another hybrid CLB design containing MUX4 hard MUX elements and demonstrated procedures for efficiently mapping to these models. We also gave analysis of the benchmark suites post mapping, examining the conveyance of functions inside every benchmark suite. The area decreases for nonfracturable structures, is 8% and MUX4:LUT proportion is 4:6 and on account of fracturable engineering the area diminishments are 2%.The CHStone benchmarks being abnormal state blended with LegUp-HLS also demonstrated marginally better execution and this could be because of the way LegUp performs HLS on the CHStone benchmarks themselves. Overall, the expansion of MUX4s to FPGA models minimally affect FMax and show potential for enhancing logic-thickness in nonfracturable structures and unobtrusive potential for enhancing logic thickness in fracturable design.

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 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 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 synthesisinspired 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 evalution for k/m-macrocell-based FPGAs,"ACM Trans. Design Autom. Electron. Syst., vol. 10, no. 1, pp. 3–23, Jan. 2005

[10] (2011). Virtex-6 FPGA User Guide. [Online]. Available: http://www.xilinx.com

[11] (2011). Stratix IV Device Handbook. [Online]. Available: http://www.altera.com

[12] G. Lemieux and D. Lewis, "Using sparse crossbars within LUT," in Proc. 9th Int. Symp. ACM/SIGDA FPGA, 2001, pp. 59–68.

[13] C. Chiasson and V. Betz, "COFFE: Fully-automated transistor sizing for FPGAs," in Proc. Int. Conf. FPT, Dec. 2013, pp. 34–41.

[14] Predictive Technology Model. [Online]. Available: http://ptm.asu.edu/, accessed 2015.

[15] 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.

[16] 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.

[17] K. Karplus, "Amap: A technology mapper for selector-based fieldprogrammable gate arrays," in Proc. 28th ACM/IEE DAC, Jun. 1991, pp. 244–247.

[18] 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.

[19] 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.

[20] S. A. Chin and J. H. Anderson, "A case for hardened multiplexers in FPGAs," in Proc. FPT, Dec. 2013, pp. 42–49.

[21] 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.