Hybrid LUT/Multiplexer FPGA Logic Architectures

Stephen Alexander Chin, *Student Member, IEEE*, Jason Luu, Safeen Huda, and Jason H. Anderson, *Member, IEEE*

***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.**

***Index Terms*— Field-programmable gate array (FPGA), hybrid complex logic block, multiplexer (MUX).**

I. 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 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 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 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

Manuscript received January 19, 2015; revised May 8, 2015; accepted June 9, 2015. Date of publication September 17, 2015; date of current version March 18, 2016.

S. A. Chin, S. Huda, and J. H. Anderson are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G8, Canada (e-mail: xan@ece.utoronto.ca; safeen@ece.utoronto.ca; janders@ece.utoronto.ca).

J. Luu is with Altera Inc., Toronto, ON M5S 2X9, Canada (e-mail: jluu@altera.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2015.2451658

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 non-fracturable 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.

II. Related Work

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.

III. Proposed Architectures

A number of FPGA architecture variants were evaluated and all are based on the basic MUX4 element described in Section III-A. The design of LEs and considerations for fracturability are addressed in Section III-B followed by hybrid-CLB design in Section III-C. The areas of all proposed architectures are discussed in Section III-D.

A. 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

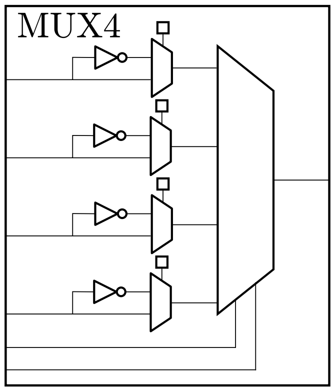


Fig. 1. MUX4 LE depicting optional data input inversions.

pin 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.

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. 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.

For the fracturable architecture, we consider an eight-input LE, closely matched with the adaptive logic

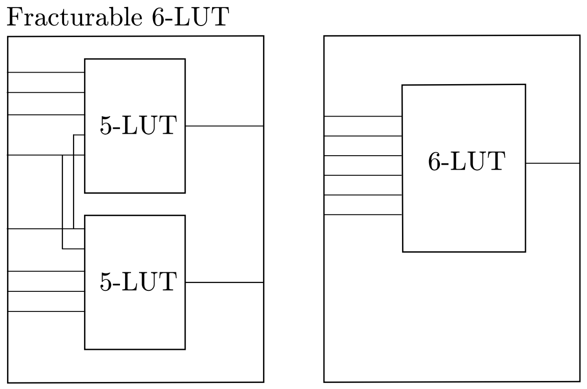


Fig. 2. Fracturable 6-LUT that can be fractured into two 5-LUTs with two shared inputs.

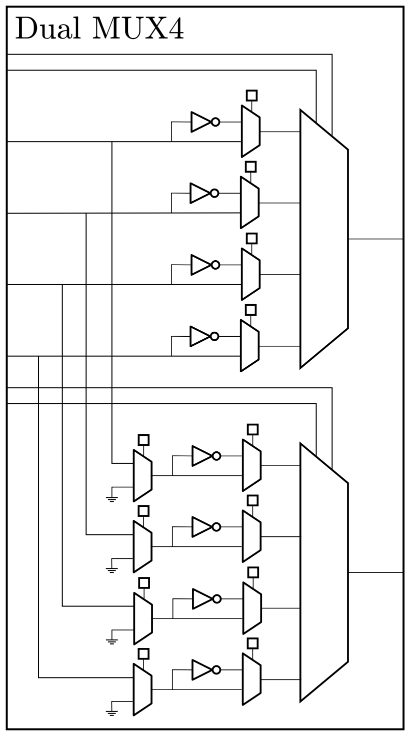


Fig. 3. Dual MUX4 LE that utilizes dedicated select inputs and shared data inputs.

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.

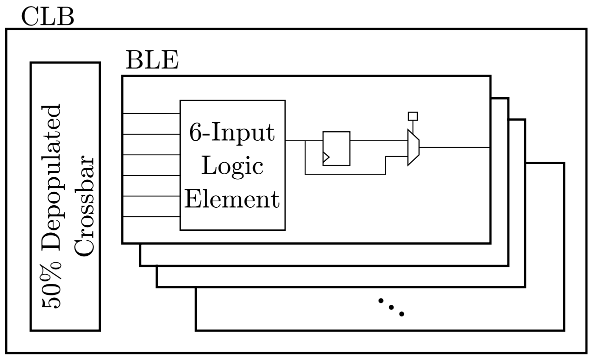


Fig. 4. Hybrid CLB with a 50% depopulated intra-CLB crossbar depicting BLE internals for a nonfracturable (one optional register and one output) architecture.

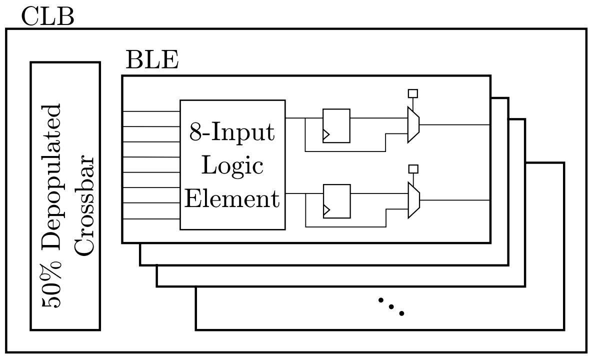


Fig. 5. Hybrid CLB with a 50% depopulated intra-CLB crossbar depicting BLE internals for a fracturable (two optional registers and two outputs) architecture.

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. 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.

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

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

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. Extended discussion on architecture modelling follows in Section V.

D. Area Modelling

1) MUX4 Logic Element: Initial estimates of the MUX4 element showed 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. Hence, the MUX4 element contains seven 2-to-1 MUXs, four SRAM cells, and four inverters in total (see Fig. 1). The optional inversion uses the four SRAM cells, whereas the rest of the LE configuration is performed through routing. In addition, the depth of the MUX tree is halved compared with the 6-LUT, which has six 2-to-1 MUXs on its longest paths. Conservatively, assuming constant pass transistor sizing 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 minimum-width 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.

2) 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, following

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

Anderson 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.

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 $LogicChange\% = (3 \times 0.116 + 7)/10$ (3 MUX4s each at 0.116 the area of a 6-LUT and 7 6-LUTs), and multiplying $LogicChange\% \times 30\% = 22\%$ of total FPGA area. If routing and miscellaneous area were held constant, our overall architecture area is $Area_{3:7} = 50\% + 20\% + 22\% = 92\%$ 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 ($CLBChange\%$) and routing demand may increase postplacement and routing ($RoutingChange\%$). In general, the model used to estimate area relative to the baseline 6-LUT only architecture (nonfracturable or fracturable) is as follows:

$$\begin{aligned} Area\% = CLBChange\% \times (&50\% \times RoutingChange\% \\ &+ 30\% \times LogicChange\% + 20\%). \end{aligned} \quad (1)$$

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

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

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.

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 all mapper variants, cuts that are MUX4-embeddable need to be identified, meaning that we must determine whether the logic function implied by such cuts can be implemented in a MUX4 LE. For LUTs that are identified as MUX4-embeddable, we tag them in the mapped network written out by ABC so that the VPR is able to pack these elements into the hybrid-CLB architectures. The identification function essentially performs a 2-level Shannon decomposition for all combinations of select inputs—a maximum of C_2^6 times for a cut of size 6. For a logic function f, let $Inputs(f)$ represent its variable set, $\{x_0, \ldots, x_i\}$, and let $f_{x_i x_j}$ represent the Shannon cofactor of f with respect to its variables x_i and x_j. Logic function f can be implemented in a MUX4 if and only if

$$|Inputs(f)| \leq 3 \tag{3}$$

or

$$\begin{aligned} &\exists x_i, x_j \in Inputs(f) \text{ such that} \\ &\qquad |Inputs(f_{x_i x_j})| \leq 1 \\ &\qquad |Inputs(f_{x_i \overline{x}_j})| \leq 1 \\ &\qquad |Inputs(f_{\overline{x}_i x_j})| \leq 1 \\ &\qquad |Inputs(f_{\overline{x}_i \overline{x}_j})| \leq 1. \end{aligned} \tag{4}$$

That is, any function up to three inputs can be implemented in a MUX4, and for functions with four or more inputs, there must exist two variables such that the Shannon cofactors with respect to such variables have one or fewer inputs.

Likewise, conditions must hold true for MUX2-embeddability using a 1-level Shannon decomposition. Note that a 1-level Shannon decomposition technique has previously been leveraged for mapping into asymmetric-LUT architectures [8]. Any function up to two inputs can be implemented in a MUX2. The only three-input function that can be implemented in a MUX2 is a 2-to-1 MUX with optional data input inversion. This limited flexibility was quickly seen with the implementation of a MUX4 that was fracturable into two MUX2s. These MUX2s do not provide ample function diversity to justify the overheads of getting the signals into and out of the hybrid CLB and led to the creation of the Dual MUX4.

Compared with the preliminary publication [15], we have also fully modeled all MUX inputs for wholly accurate MUX4 input mapping. Previously, all the MUX data-inputs were optimistically modeled to be equivalent and interchangeable. Likewise for MUX select-inputs. 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. NaturalMux

NaturalMux 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. MuxMap

In default ABC technology mapping, each LUT has a unit area of 1. In our MuxMap 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 routing is 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.

```
<pb_type name=‘mux4hard’ blif_model=‘.subckt mux4’ num_pb
     =‘1’>
 <input name=‘s’  num_pins=‘2’/>
 <input name=‘in’ num_pins=‘4’/>
 <output name=‘out’ num_pins=‘1’/>
</pb_type>
```

Fig. 6. MUX4 LE model.

```
<!–– Within LUT Element definition ––>
<mode name=‘6lut’>
 <pb_type name=‘lut6’ blif_model=‘.names’ num_pb=‘1’ class
      =‘lut’>
  <input name=‘in’ num_pins=‘6’ port_class=‘lut_in’/>
  <output name=‘out’ num_pins=‘1’ port_class=‘lut_out’/>
 </pb_type>
<mode name=‘mux4’>
 <pb_type name=‘mux4lut’ blif_model=‘.subckt mux4’ num_pb
      =‘1’>
  <input name=‘s’ num_pins=‘2’/>
  <input name=‘in’ num_pins=‘4’/>
  <output name=‘out’ num_pins=‘1’/>
 </pb_type>
```

Fig. 7. MUX4 mode in the 6-LUT element model.

C. Select Mapping

Depending on the circuit, NaturalMux or MuxMap 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 MuxMap 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. Modelling Using VPR

VPR was used to perform architectural evaluation. The standard ten 6-LUT CLB architecture in 40-nm included with the VPR distribution was used for baseline modelling [1]. The hybrid CLBs shown in Figs. 4 and 5 were modeled using the XML-based VPR architectural language [1]. The snippet from the architecture file for the physical block hardened MUX4 element is shown in Fig. 6—this code specifies a MUX4 as a six-input one-output blackbox to the VPR. In addition, since all MUX4s can also be mapped to the 6-LUTs, an additional mode was added to the 6-LUT physical block, shown in Fig. 7. The mode concept allows the VPR packer to pack LUTs into LUTs (as usual), but also enables MUX4s to be packed into the LUTs. 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 MuxMap mapping schemes. After both mappings were performed, the select mapping was performed as described in Section IV-C.

While the VTR7 benchmarks are input to our flow as Verilog, the CHStone benchmarks are input as C. LegUp 3.0 [3] was used for C-to-Verilog HLS for these CHStone benchmarks. Differing from our preliminary work [15] (that had two separate tool flows), modifications were made to both the LegUp and ODIN-II tools so that CHStone benchmarks could be carried through the typical full VTR flow (ODIN-II, ABC, VPR) using ODIN-II for front-end synthesis. In the preliminary work [15], Altera Quartus II is used for the front-end synthesis thus producing an Altera-specific mapping for DSP blocks, Memory blocks, and Carry Chains—all of which are not supported by the standard VPR architectures. This restricted us only to be able to make post-mapping estimates. In this new tool-flow, ODIN-II is able to synthesize the Verilog output from LegUp-HLS. ODIN-II, thereby produces the compatible DSP and Memory blocks for our modified standard FPGA architectures, allowing for post-place-and-route results (previously nonexisting).

Lastly, the *mcml* and *LU32PEEng* benchmarks were omitted from the VTR7 benchmark suite due to excessive runtime.

1) Mapping: Using ABC, we performed technology-independent optimization using the standard *resyn2* script included with the ABC distribution [13]. Then, we performed technology mapping with the priority cuts mapper (*if* command), targeting an LUT size of 6. Our modified priority cuts mapper was invoked using the *if* command with a cut set size of 32 also limiting the maximum cut size to 6. The initial unaltered baseline mapper was run without any mapping to MUX4s, and then NaturalMux mapping was performed for the identification of MUX4-embeddable functions. Our second mapper, MuxMap, seeks to improve the mapping by reducing the area cost of a MUX4-embeddable function within priority cuts mapping.

After mapping with both mappers, we computed the ratio of the number of MUX4-embeddable LEs to the total number of LEs, i.e., the MUX4 percentage. Based on this ratio for each circuit, we projected the area benefits of hybrid architectures 1:9 to 5:5 MUX4:LUT ratios in the nonfracturable architectures. The area projections were made assuming complete (full) CLB packing, and the tile area breakdown described in Section III-D (assuming no routing area change). The results are shown in Table III and demonstrate the maximum area savings to be had for each benchmark, since these projections ignore the effects of packing, placement, and routing constraints. For the fracturable architectures, this prediction is made difficult, since pairing of functions to be packed into fractured LEs is largely determined by pin-sharing on the LE inputs—a packing constraint. No predictions are made for the fracturable architectures, however, packed, placed, and routed results for the fracturable architectures are discussed in Section VI-B.

The MuxMap improves mapping by increasing the number of MUX4-embeddable logic functions by reducing the area cost of a MUX4-embeddable function, thereby encouraging the mapper to create more MUX4-embeddable functions, increasing the MUX4-embeddable ratio of each circuit overall. However, the improvements in the ratio may come at the cost of a higher total number of LEs. If there is a significant increase in LEs (greater than the limits shown in Table II), no area savings may be seen. Fig. 8 shows a sweep of the weighting of MUX4-embeddable LEs from 0% cost to 100% cost of a 6-LUT over the VTR7 suite; and the projected area is shown on the vertical axis (normalized to a traditional LUT-based architecture). Area is minimized for more aggressive architectures (5:5 and 4:6) around a weighting of 50%–60% whereas less aggressive architectures (3:7, 2:8, 1:9) do not benefit at all from a reduced weighting. This can be seen as the minimum area for a weighting of 100%.

As the number of LEs grow, 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 MuxMap 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	%	Arch. M:L Ratio					% LE	%	Arch. M:L Ratio				
			MUX4	1:9	2:8	3:7	4:6	5:5	Change	*MUX4*	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							

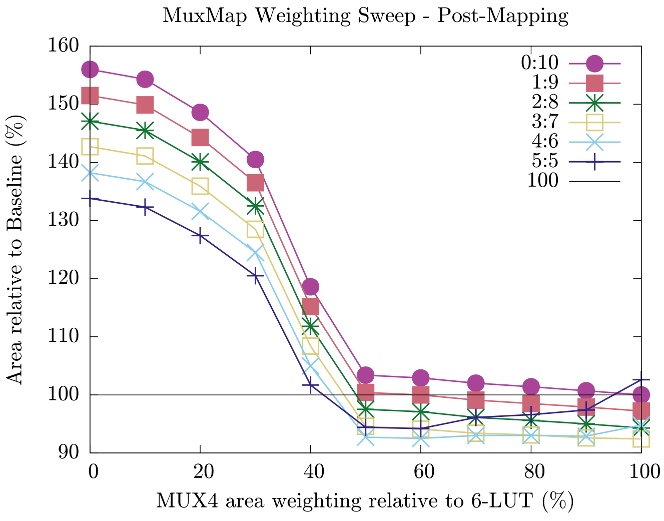


Fig. 8. MuxMap varying the weight from zero area to equal 6-LUT area for different MUX4:LUT architectures. A weight equal to the LUT area is equivalent to the NaturalMux mapping. As the MUX4 area weighting is increased for some architecture ratios (e.g., 5:5 and 4:6), an increase in relative area is seen due to unbalanced circuit and architecture MUX4:LUT ratios.

The right-hand side shows projected area results for the MuxMap mapping with a weighting of 0.8. For both of the mappers, we show projected results for the architectures with ratios ranging from 1:9 to 5:5. Again, Table II shows the lower bound architectural areas relative to a traditional 6-LUT-based architecture for the configurations tested and this lower bound can only be achieved when the natural MUX4-embeddable ratio exceeds the architecture ratio. When the natural ratio is lower than the architecture ratio, more CLBs are required, increasing area. Table III shows the lower bound on area for each benchmark and nonfracturable architecture, since this projection assumes perfect packing, placement, and routing of each benchmark on each architecture. Looking at the left half of Table III, the baseline number of LEs along with the natural MUX4-embeddable ratio is given along with projected areas for each architecture for NaturalMux.

The VTR7 benchmarks are projected to perform well, with a high natural MUX4-embeddable ratio of 40% over all benchmarks. For the VTR7 benchmarks, a 3:7 architecture seems to be the best architecture for NaturalMux mapping, yielding ~7% projected area reduction postmapping. MuxMap shows an increase in MUX4-embeddable ratio to 46% over all benchmarks; however, the increase in LE count associated with MuxMap leads to worse projected area results per architecture versus NaturalMux, except for the 4:6 and 5:5 architectures. Employing the Select mapping scheme yields marginally better area reduction for the best architecture ratio, 3:7 (~1% above the architectural minimum).

The CHStone benchmarks all show large percentages of MUX4-embeddable functions using the NaturalMux mapper, 55%. In fact, the percentage of MUX4-embeddable functions is so large that in nearly all cases, the architectural minimum area is reached. In this case, the MuxMap mapper

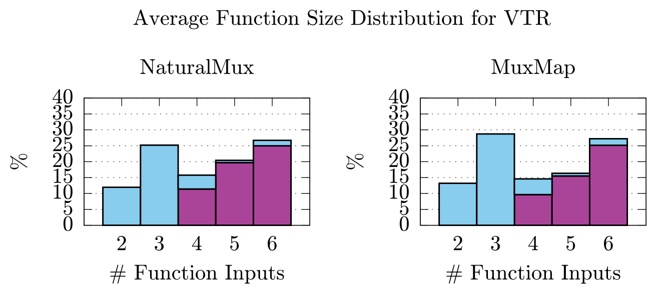


Fig. 9. Input-size distribution of VTR7 benchmarks with the MUX4-embeddable LEs in blue (light gray) and LUTs in purple (dark gray).

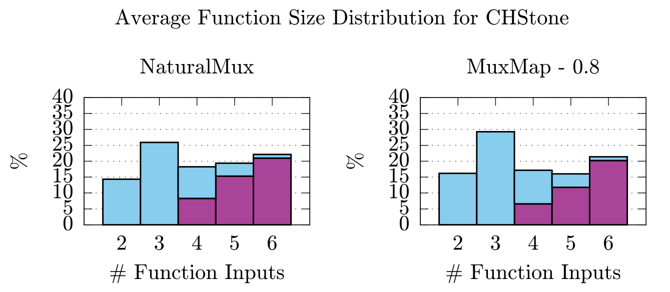


Fig. 10. Input-size distribution for CHStone benchmarks with the MUX4-embeddable cuts in blue (light gray) and LUTs in purple (dark gray).

and Select schemes have no further improvements. From these postmapping estimates, it seems as though even a very aggressive 5:5 architectural ratio for the CHStone suite may be the best as we reach the architectural minimum area for the nonfracturable 5:5 architecture. This variation between benchmark suites is interesting and the CHStone benchmarks suiting more aggressive architectures could be a function of how LegUp-HLS transforms the CHStone benchmarks, written in C, to hardware. Again, we see that different architectural conclusions can be made based on the benchmark circuits employed in an architectural study [24].

We can also examine how the weighting within the MuxMap mapper affects the used input-size distribution for the LUTs in the circuits versus the NaturalMux mapper. The input-size distributions for the NaturalMux and MuxMap mappings are shown in Figs. 9 and 10 for VTR7 and CHStone benchmarks, respectively. Each bar in the distribution shows the portion of LEs (with a given number of used inputs) that are MUX4-embeddable (in light-blue). The VTR suite has only a small percentage of functions with more than three inputs that are naturally MUX4-embeddable ($\sim$10%). With MuxMap, the percentage of larger five- and six-input functions is somewhat reduced with these functions being implemented mostly as three- or four-input MUX4-embeddable functions. Looking at the six-input functions specifically, we also observe that very few are MUX4-embeddable. Recall that the only six-input logic function that is MUX4-embeddable is a 4-to-1 MUX (with possible data input inversions). Two- and three-input MUX4-embeddable functions are also quite abundant in both the VTR7 and CHStone benchmarks, comprising the majority of all the MUX4-embeddable functions. Concerning these small functions, note that over 35% of functions in the VTR7 circuits and 40% in CHStone have three or fewer inputs (Fig. 9), which bodes well for the proposed hybrid architectures. In addition, in both benchmark suites, MuxMap encourages a decrease in five-input elements and an increase in MUX4-embeddable three-input elements.

A. Nonfracturable Architectures: Packing, Place, and Route

Of course, the previous postmapping results do not consider the possible impact of the addition of MUX4 elements to the FPGA architecture on packing, placement, and routability. Therefore, the mapped netlists were packed, placed, and routed, using VPR, into the architectures discussed in Section V. The final area estimates in Tables IV and V show the area reduction relative to the baseline mapper and a traditional nonfracturable 6-LUT architecture. We use the area equation shown in Section III for these estimates. VPR was configured to find minimum channel width with timing driven routing. For all benchmarks, five different placement seeds were used to account for the random effects of placement on final routing. For $CLBChange\%$, we use the change in packed CLBs over the baseline. For $RoutingChange\%$, we use the change in routing area per logic tile (measured in minimum-width transistors) reported by VPR. Finally, we compute $LogicChange\%$ according to the chosen hybrid CLB architecture ratio—this is constant for a given architecture and represents the physical reduction in LE area.

Tables IV and V show the results for each benchmark using the NaturalMux and MuxMap mapping schemes respectively, over all nonfracturable architectures. In Tables IV and V, the baseline CLB count is shown with the natural MUX4-embeddable LE ratio. For each architecture, the percentage change in CLBs and routing with respect to the baseline 6-LUT-only architecture is shown with area based on the model in Section III-D. For each benchmark suite, we take the geomean area across all benchmarks in the suite. Each of these final mean relative area numbers is shown in bold. Table V has additionally a row showing the select geomean, which is the result of applying the select scheme between both mappers for each architecture.

Overall, we see increases in packed CLBs for both benchmark suites, but for most architectures, this gain is offset by the smaller hybrid CLB area. Looking at the minimum geomean areas in Table IV, we can see that the best architecture suited toward NaturalMux mapping is the 2:8 for VTR7 and 4:6 for CHStone. But for the MuxMap mapping scheme in Table V, the VTR7 performs best on a 3:7 architecture and CHStone performs the best on a 5:5 architecture. However, the best results overall are shown with Select. For Select mapping, the VTR7 on a 3:7 architecture is the best with $\sim$4% area reduction, but it is down four percentage points from the postmap estimate. CHStone performs best in a 4:6 architecture at $\sim$9% area reduction. The postmapping projections again suggested 3% points better area for the 5:5 architecture.

For the nonfracturable architectures, a ratio of 3:7 seems fitting as we see gains for both benchmark suites—though not full gains for the CHStone.

TABLE IV

NONFRACTURABLE ARCHITECTURE: POSTPLACE AND ROUTE RESULTS SHOWING ARCHITECTURAL SWEEPS BETWEEN ZERO (BASELINE) TO FIVE MUX4 LEs OUT OF TEN, FOR FRACTURABLE HYBRID-CLB ARCHITECTURES, USING THE NATURALMUX MAPPING SCHEME. ALL BENCHMARKS WERE AVERAGED OVER FIVE PLACEMENT SEEDS

		Baseline		**NaturalMux**														
		Nat.*MUX4* Ratio	# CLBs	Arch. 1:9			Arch. 2:8			Arch. 3:7			Arch. 4:6			Arch. 5:5		
				% CLBs	% Rtng	% Area	% CLBs	% Rtng	% Area	% CLBs	% Rtng	% Area	% CLBs	% Rtng	% Area	% CLBs	% Rtng	% Area
	bgm	0.25	3599	100.0	99.5	97.1	100.0	113.1	101.3	100.0	157.4	120.8	114.5	153.4	132.9	137.0	149.9	153.0
	blob_merge	0.22	604	100.0	100.6	97.6	101.5	109.6	101.0	113.1	109.2	109.3	130.8	104.2	119.7	156.3	92.8	129.9
	boundtop	0.43	300	100.0	98.9	96.8	100.0	100.0	94.7	100.0	103.9	94.0	100.7	109.1	94.6	112.7	110.4	103.6
	ch_intrinsics	0.29	39	100.0	100.0	97.3	100.0	100.4	94.9	102.6	103.1	96.0	117.9	102.7	107.0	141.0	94.9	118.7
	diffeq1	0.33	47	100.0	105.0	99.8	100.0	107.4	98.4	102.1	109.1	98.7	110.6	105.9	102.2	134.0	110.2	123.1
	diffeq2	0.32	32	100.0	99.1	96.9	100.0	102.4	95.9	100.0	107.1	95.6	115.6	99.1	102.9	137.5	99.6	119.0
	LU8PEEng	0.43	2579	100.1	101.8	98.4	100.0	108.9	99.2	100.0	118.1	101.1	100.0	130.2	104.5	104.7	144.9	114.3
	mkDelayWorker32B	0.43	550	100.0	102.4	98.6	100.0	102.3	95.9	100.4	103.4	94.1	106.4	108.2	99.5	119.1	109.8	109.1
vtr	mkPktMerge	0.74	24	100.0	99.5	97.1	100.0	95.4	92.4	100.0	98.0	91.0	100.0	98.0	88.4	100.0	99.0	86.2
	mkSMAdapter4B	0.32	204	100.0	102.4	98.6	100.0	101.8	95.6	100.5	106.5	95.8	112.7	108.8	105.7	134.3	107.9	121.8
	or1200	0.38	298	99.7	104.6	99.3	99.7	103.6	96.2	101.3	106.2	96.4	106.7	111.0	101.3	123.2	111.1	113.7
	raygentop	0.31	240	100.0	100.0	97.3	100.0	103.7	96.5	100.8	104.0	94.8	105.0	105.6	96.8	119.6	105.2	106.8
	sha	0.35	225	100.0	102.7	98.7	100.4	103.3	96.8	102.7	106.0	97.6	109.3	112.7	104.7	128.4	120.0	124.3
	stereovision0	0.76	1494	100.0	101.3	98.0	100.0	104.8	97.1	100.0	103.4	93.8	100.0	107.7	93.3	100.0	107.3	90.4
	stereovision1	0.69	1336	100.0	97.3	96.0	100.0	98.7	94.0	100.0	99.7	91.9	100.0	101.7	90.2	100.0	103.2	88.3
	stereovision2	0.52	3584	100.0	100.7	97.7	100.0	100.9	95.2	100.0	103.5	93.8	100.0	108.4	93.6	100.0	127.5	100.5
	stereovision3	0.51	19	100.0	101.2	98.0	100.0	100.6	95.0	94.7	101.8	88.1	100.0	106.8	92.8	100.0	110.5	92.0
	Geomean	-	-	-	-	**97.8**	-	-	**96.4**	-	-	**97.0**	-	-	**101.2**	-	-	**110.2**
	adpcm	0.63	1381	100.4	102.7	99.1	101.0	105.3	98.3	102.1	104.2	96.1	103.4	102.8	93.9	105.9	101.2	92.5
	aes	0.46	1430	100.1	98.4	96.6	100.3	99.5	94.8	100.6	100.2	92.7	101.1	98.5	89.7	104.1	104.7	92.7
	blowfish	0.49	1563	100.1	97.1	96.0	100.3	92.3	91.1	100.5	94.6	89.8	101.0	92.4	86.4	101.6	91.9	84.0
	dfadd	0.54	664	100.3	104.3	99.8	100.6	101.7	96.1	101.5	103.7	95.3	102.4	101.7	92.4	105.3	102.0	92.4
	dfdiv	0.52	754	100.1	101.0	98.0	100.5	102.9	96.7	101.1	102.5	94.3	102.1	101.9	92.3	105.8	106.6	95.3
CHStone	dfmul	0.46	479	100.6	101.3	98.6	101.5	100.6	96.4	102.9	102.7	96.1	105.0	99.7	93.7	110.2	103.9	97.8
	dfsin	0.60	2020	100.1	100.0	97.5	100.3	103.4	96.7	100.6	101.4	93.3	101.0	103.0	91.8	101.7	105.4	90.9
	gsm	0.58	1261	100.0	99.1	96.9	100.0	102.1	95.7	100.1	108.0	96.1	100.3	107.6	93.5	101.3	112.6	94.2
	jpeg	0.55	3665	100.2	101.0	98.1	100.5	102.5	96.5	101.0	101.5	93.7	102.1	99.1	90.8	107.3	102.4	94.4
	mips	0.53	475	100.2	93.0	94.0	100.2	98.4	94.1	100.2	94.8	89.6	100.4	97.3	88.4	104.0	101.1	90.8
	motion	0.49	520	100.0	97.9	96.3	100.2	100.4	95.1	100.2	100.9	92.7	101.2	103.9	92.4	109.0	104.7	97.1
	sha	0.52	1364	100.1	100.6	97.8	100.3	98.4	94.1	100.7	100.0	92.7	101.3	100.4	90.8	103.5	104.5	92.1
	Geomean	-	-	-	-	**97.4**	-	-	**95.4**	-	-	**93.5**	-	-	**91.3**	-	-	**92.8**

TABLE V

NONFRACTURABLE ARCHITECTURE: POSTPLACE AND ROUTE RESULTS SHOWING ARCHITECTURAL SWEEPS BETWEEN ZERO (BASELINE) TO FIVE MUX4 LEs OUT OF TEN, FOR FRACTURABLE HYBRID-CLB ARCHITECTURES USING THE MUXMAP MAPPING SCHEME. ALL BENCHMARKS WERE AVERAGED OVER FIVE PLACEMENT SEEDS

		Baseline		**MuxMap - 0.8**														
		Nat.*MUX4* Ratio	# CLBs	Arch. 1:9			Arch. 2:8			Arch. 3:7			Arch. 4:6			Arch. 5:5		
				% CLBs	% Rtng	% Area	% CLBs	% Rtng	% Area	% CLBs	% Rtng	% Area	% CLBs	% Rtng	% Area	% CLBs	% Rtng	% Area
	bgm	0.25	3599	101.2	100.4	98.7	101.3	108.8	100.3	101.3	142.8	114.8	110.3	150.0	126.1	132.0	152.7	149.3
	blob_merge	0.22	604	101.0	101.8	99.2	101.0	108.9	100.1	107.6	106.3	102.4	121.7	103.7	111.0	142.1	100.8	123.8
	boundtop	0.43	300	100.3	91.0	93.2	100.3	93.8	91.9	100.3	100.4	92.6	100.3	109.1	94.2	101.0	115.4	95.4
	ch_intrinsics	0.29	39	100.0	100.0	97.3	100.0	100.4	94.9	102.6	103.1	96.0	117.9	102.7	107.0	141.0	94.9	118.7
	diffeq1	0.33	47	102.1	106.3	102.6	102.1	107.3	100.5	102.1	105.9	97.0	102.1	105.0	93.8	114.9	104.2	102.0
	diffeq2	0.32	32	103.1	96.3	98.5	103.1	97.8	96.5	103.1	102.5	96.2	112.5	103.3	102.4	134.4	115.4	126.9
	LU8PEEng	0.43	2579	100.6	101.8	98.8	100.6	103.1	96.8	100.5	110.3	97.7	100.6	124.8	102.4	100.6	143.3	109.0
	mkDelayWorker32B	0.43	550	101.3	100.4	98.8	101.3	103.4	97.6	101.3	103.6	95.1	104.7	113.2	100.5	115.6	109.5	105.8
vtr	mkPktMerge	0.74	24	100.0	95.9	95.3	100.0	98.5	93.9	100.0	98.0	91.0	100.0	100.9	89.8	100.0	96.9	85.2
	mkSMAdapter4B	0.32	204	101.5	103.0	100.3	101.5	103.0	97.6	101.5	108.1	97.5	106.4	110.6	100.7	125.5	106.5	112.9
	or1200	0.38	298	102.3	101.4	100.4	102.3	102.2	98.1	102.3	100.8	94.6	103.7	110.4	98.1	109.7	106.2	98.6
	raygentop	0.31	240	102.5	100.7	100.1	102.5	102.0	98.1	102.5	99.3	94.0	103.8	100.0	92.7	105.0	106.5	94.5
	sha	0.35	225	100.4	104.4	100.0	100.4	104.4	97.3	103.6	107.9	99.4	107.6	113.1	103.2	126.7	119.3	122.1
	stereovision0	0.76	1494	101.7	104.4	101.2	101.7	102.6	97.6	101.7	103.0	95.1	101.7	105.9	93.9	101.7	104.8	90.6
	stereovision1	0.69	1336	100.1	100.0	97.5	100.1	102.7	96.2	100.1	103.2	93.7	100.1	105.0	92.0	100.1	105.0	89.3
	stereovision2	0.52	3584	101.1	98.1	97.4	101.0	100.0	95.7	101.0	97.9	92.0	101.0	103.6	92.2	101.1	112.3	93.9
	stereovision3	0.51	19	100.0	100.0	97.3	100.0	100.0	94.7	100.0	103.6	93.9	100.0	104.9	91.8	100.0	109.3	91.4
	Geomean	-	-	-	-	**98.6**	-	-	**96.9**	-	-	**96.5**	-	-	**99.2**	-	-	**105.2**
Select Geomean		-	-	-	-	**97.5**	-	-	**96.0**	-	-	**95.7**	-	-	**98.9**	-	-	**104.7**
	adpcm	0.63	1381	101.5	104.7	101.2	102.2	105.8	99.8	103.4	103.1	96.8	104.7	100.3	93.8	106.9	101.3	93.4
	aes	0.46	1430	102.8	100.4	100.3	103.1	101.1	98.2	103.2	102.5	96.3	103.6	98.8	92.0	104.3	98.1	89.5
	blowfish	0.49	1563	100.0	96.0	95.4	100.1	95.8	92.7	100.2	93.8	89.1	100.3	91.6	85.4	100.8	92.3	83.6
	dfadd	0.54	664	101.5	102.0	99.8	101.7	101.8	97.2	102.3	101.2	94.8	103.0	100.8	92.5	104.4	103.6	92.4
	dfdiv	0.52	754	102.8	97.2	98.6	103.2	94.7	95.0	103.7	95.9	93.3	104.4	99.6	93.1	106.1	102.1	93.1
CHStone	dfmul	0.46	479	102.3	100.0	99.6	103.1	101.0	98.2	104.4	99.6	95.9	105.8	100.3	94.8	110.4	106.7	99.5
	dfsin	0.60	2020	102.3	95.3	97.1	102.5	97.5	95.7	102.7	97.3	93.1	103.1	98.3	91.3	103.6	102.6	91.2
	gsm	0.58	1261	102.1	98.4	98.5	102.1	101.7	97.5	102.1	102.2	95.2	102.2	102.1	92.5	102.5	106.0	91.9
	jpeg	0.55	3665	102.9	100.4	100.4	103.3	100.7	98.2	103.8	100.6	95.8	104.6	98.2	92.6	106.7	98.4	91.7
	mips	0.53	475	102.1	94.1	96.4	101.9	95.1	94.0	102.1	96.1	92.0	102.3	94.6	88.7	103.6	95.3	87.4
	motion	0.49	520	102.7	99.8	99.8	102.9	102.0	98.5	103.1	101.1	95.5	103.3	102.9	93.8	106.2	103.1	93.7
	sha	0.52	1364	103.6	99.8	100.7	103.7	99.8	98.1	104.0	99.6	95.5	104.5	98.3	92.5	105.5	98.4	90.6
	Geomean	-	-	-	-	**99.0**	-	-	**96.9**	-	-	**94.4**	-	-	**91.9**	-	-	**91.4**
Select Geomean		-	-	-	-	**97.3**	-	-	**95.2**	-	-	**93.2**	-	-	**91.1**	-	-	**91.2**

TABLE VI

FRACTURABLE ARCHITECTURE: POSTPLACE AND ROUTE RESULTS SHOWING ARCHITECTURAL SWEEPS BETWEEN ZERO (BASELINE) TO FIVE MUX4 LEs OUT OF TEN, FOR FRACTURABLE HYBRID-CLB ARCHITECTURES, USING THE NATURALMUX MAPPING SCHEME. ALL THE BENCHMARKS WERE AVERAGED OVER FIVE PLACEMENT SEEDS

		Baseline		**NaturalMux**														
		Nat.*MUX4* Ratio	# CLBs	Arch. 1:9			Arch. 2:8			Arch. 3:7			Arch. 4:6			Arch. 5:5		
				% CLBs	% Rtng	% Area	% CLBs	% Rtng	% Area	% CLBs	% Rtng	% Area	% CLBs	% Rtng	% Area	% CLBs	% Rtng	% Area
vtr	bgm	0.25	2462	97.8	109.2	100.2	96.1	151.6	116.8	107.7	145.5	125.2	124.3	138.0	137.1	148.6	127.5	152.9
	blob_merge	0.22	424	101.9	107.3	103.4	111.8	107.4	111.0	125.7	101.0	118.2	145.3	95.7	129.5	173.6	92.4	148.0
	boundtop	0.43	187	100.0	102.8	99.2	100.5	114.8	103.6	101.6	118.5	104.4	113.9	123.1	117.1	133.7	109.9	125.7
	ch_intrinsics	0.29	27	100.0	98.4	97.0	103.7	98.8	98.5	114.8	99.2	106.8	133.3	97.5	120.0	159.3	97.1	139.6
	diffeq1	0.33	31	100.0	100.5	98.1	103.2	98.4	97.9	116.1	103.2	110.4	132.3	98.7	119.9	158.1	95.8	137.5
	diffeq2	0.32	22	100.0	102.8	99.2	104.5	104.8	102.5	118.2	100.8	110.9	136.4	100.0	124.5	163.6	100.9	146.5
	LU8PEEng	0.43	1746	98.9	105.6	99.4	98.2	115.5	101.6	97.9	125.9	104.2	104.4	137.1	114.6	120.6	142.6	133.1
	mkDelayWorker32B	0.43	354	100.3	99.1	97.7	101.7	107.5	101.1	106.5	118.6	109.5	117.5	117.9	117.8	138.4	119.9	137.1
	mkPktMerge	0.74	13	100.0	105.0	100.3	100.0	101.9	96.6	107.7	102.7	102.1	100.0	105.0	93.8	100.0	102.7	90.5
	mkSMAdapter4B	0.32	127	100.0	102.1	98.9	102.4	105.9	100.9	111.0	107.4	107.9	127.6	108.0	121.6	152.0	106.4	140.3
	or1200	0.38	196	100.5	99.7	98.2	103.1	104.7	101.0	110.7	106.1	106.9	125.0	104.9	117.2	147.4	106.4	136.1
	raygentop	0.31	167	96.4	101.3	94.9	96.4	103.5	93.9	97.0	109.7	95.4	109.0	113.1	106.6	128.7	114.1	123.8
	sha	0.35	162	100.6	103.6	100.2	101.2	115.5	104.7	112.3	123.3	118.1	129.6	120.1	131.3	154.9	106.1	142.8
	stereovision0	0.76	899	96.4	102.2	95.4	94.2	106.3	93.1	92.2	108.2	90.0	91.1	112.0	88.6	89.9	114.5	86.6
	stereovision1	0.69	842	96.2	100.8	94.5	93.7	103.0	91.0	90.7	104.3	86.8	92.5	102.7	85.7	93.5	112.7	89.2
	stereovision2	0.52	2304	97.3	105.0	97.6	95.8	111.7	97.2	94.0	124.2	99.2	92.8	136.7	101.8	101.4	141.0	111.1
	stereovision3	0.51	12	100.0	103.2	99.4	100.0	106.6	98.9	100.0	122.3	104.6	108.3	99.7	98.7	116.7	101.9	105.0
	Geomean	-	-	-	-	**98.4**	-	-	**100.4**	-	-	**105.5**	-	-	**112.3**	-	-	**124.3**
CHStone	adpcm	0.63	803	100.0	102.5	99.1	101.4	103.1	98.5	103.4	104.1	98.7	108.8	105.8	102.5	118.6	105.6	108.9
	aes	0.46	904	99.8	103.6	99.4	100.7	101.2	96.9	102.0	109.2	100.0	111.6	115.7	110.6	131.5	110.8	124.3
	blowfish	0.49	966	100.2	101.2	98.6	100.5	97.5	94.9	101.3	100.3	94.9	103.5	105.2	97.2	116.5	110.2	109.7
	dfadd	0.54	413	100.0	101.4	98.5	100.7	101.7	97.2	102.7	104.4	98.2	107.5	109.0	103.0	120.1	103.5	109.1
	dfdiv	0.52	455	100.4	102.9	99.7	100.9	103.1	98.0	102.2	107.7	99.4	110.1	113.6	108.0	126.8	112.6	121.0
	dfmul	0.46	293	100.7	100.0	98.5	101.7	103.0	98.8	103.4	107.4	100.5	112.6	111.1	109.1	131.1	111.9	124.6
	dfsin	0.60	1230	100.5	101.7	99.2	101.0	102.8	98.0	101.5	105.6	97.8	103.5	110.5	99.9	115.0	115.4	111.4
	gsm	0.58	761	100.0	103.5	99.6	100.3	104.0	97.9	100.9	103.8	96.2	106.7	114.1	104.9	121.0	112.4	115.4
	jpeg	0.55	2201	100.4	99.7	98.0	101.4	101.4	97.7	103.7	102.4	98.2	110.0	106.5	103.9	124.8	104.1	113.7
	mips	0.53	301	100.0	98.1	96.9	100.3	99.7	95.8	101.0	104.1	96.5	110.3	105.9	103.9	126.2	104.0	115.0
	motion	0.49	320	100.3	103.4	99.8	101.6	104.9	99.6	105.3	105.2	101.2	114.4	104.9	107.2	133.1	101.1	119.4
	sha	0.52	844	100.4	100.5	98.4	101.5	100.7	97.5	103.6	105.0	99.4	111.0	108.5	106.0	128.9	108.1	120.1
	Geomean	-	-	-	-	**98.8**	-	-	**97.6**	-	-	**98.4**	-	-	**104.6**	-	-	**115.9**

TABLE VII

FRACTURABLE ARCHITECTURE: POSTPLACE AND ROUTE RESULTS SHOWING ARCHITECTURAL SWEEPS BETWEEN ZERO (BASELINE) TO FIVE MUX4 LEs OUT OF TEN, FOR FRACTURABLE HYBRID-CLB ARCHITECTURES, USING MUXMAP MAPPING SCHEME. ALL THE BENCHMARKS WERE AVERAGED OVER FIVE PLACEMENT SEEDS

		Baseline		**MuxMap - 0.8**														
		Nat.*MUX4* Ratio	# CLBs	Arch. 1:9			Arch. 2:8			Arch. 3:7			Arch. 4:6			Arch. 5:5		
				% CLBs	% Rtng	% Area	% CLBs	% Rtng	% Area	% CLBs	% Rtng	% Area	% CLBs	% Rtng	% Area	% CLBs	% Rtng	% Area
vtr	bgm	0.25	2462	99.8	104.7	99.9	97.6	144.1	114.9	106.5	143.9	122.9	123.3	139.8	137.1	147.0	132.6	154.9
	blob_merge	0.22	424	103.3	97.5	99.7	110.8	103.5	107.9	122.2	100.4	114.4	140.6	96.8	126.1	167.5	92.2	142.7
	boundtop	0.43	187	98.9	103.6	98.6	99.5	111.9	101.1	99.5	118.2	102.0	104.3	122.0	106.7	120.9	116.1	117.4
	ch_intrinsics	0.29	27	100.0	98.4	97.0	103.7	98.8	98.5	114.8	99.2	106.8	133.3	97.5	120.0	159.3	97.1	139.6
	diffeq1	0.33	31	106.5	92.4	100.1	106.5	107.7	105.9	112.9	98.0	104.4	129.0	98.7	116.9	154.8	104.3	141.3
	diffeq2	0.32	22	100.0	103.3	99.5	104.5	104.0	102.1	118.2	107.9	115.2	136.4	99.2	123.9	163.6	103.6	148.8
	LU8PEEng	0.43	1746	99.4	102.5	98.5	98.9	110.8	100.0	98.6	125.4	104.6	101.6	142.7	114.4	117.6	140.3	128.5
	mkDelayWorker32B	0.43	354	99.4	97.1	95.8	100.6	104.3	98.4	104.5	117.7	106.9	114.1	116.9	113.8	132.2	118.6	130.1
	mkPktMerge	0.74	13	100.0	104.2	99.9	100.0	106.1	98.7	100.0	107.3	97.1	100.0	101.1	91.9	100.0	100.0	89.1
	mkSMAdapter4B	0.32	127	103.9	103.1	103.3	103.9	107.7	103.4	111.8	110.3	110.3	128.3	107.1	121.7	151.2	106.1	139.3
	or1200	0.38	196	103.1	98.4	100.0	103.6	100.5	99.3	106.6	103.4	101.5	115.8	103.5	107.8	135.7	106.5	125.3
	raygentop	0.31	167	98.8	101.3	97.3	98.2	103.0	95.4	98.8	102.3	93.5	99.4	114.9	98.1	109.0	116.3	106.0
	sha	0.35	162	98.8	105.4	99.3	100.0	108.2	99.7	109.3	122.4	114.3	126.5	122.2	129.5	151.2	108.9	141.5
	stereovision0	0.76	899	101.2	97.5	97.8	99.6	100.7	95.6	97.7	106.2	94.3	95.6	106.9	90.5	95.3	108.6	89.1
	stereovision1	0.69	842	95.8	102.7	95.1	92.8	101.8	89.6	89.8	106.4	86.8	93.7	106.2	88.4	93.3	110.9	88.3
	stereovision2	0.52	2304	99.5	103.9	99.3	97.7	105.4	96.1	96.8	115.2	97.8	94.6	129.6	100.3	99.9	140.3	109.1
	stereovision3	0.51	12	100.0	102.8	99.2	100.0	106.1	98.7	100.0	118.5	102.7	108.3	99.7	98.7	116.7	104.1	106.4
	Geomean	-	-	-	-	**98.8**	-	-	**100.2**	-	-	**104.1**	-	-	**110.0**	-	-	**121.4**
Select Geomean		-	-	-	-	**97.9**	-	-	**99.2**	-	-	**103.4**	-	-	**109.7**	-	-	**120.7**
CHStone	adpcm	0.63	803	101.0	102.8	100.2	102.4	99.9	97.9	104.0	98.9	96.6	107.5	104.3	100.4	114.9	105.8	105.8
	aes	0.46	904	99.9	99.9	97.7	100.2	102.0	96.9	100.9	104.6	96.6	103.8	109.2	99.5	118.8	109.5	111.5
	blowfish	0.49	966	98.8	104.2	98.7	99.1	101.5	95.5	99.6	102.8	94.5	101.0	102.0	93.3	109.6	105.1	100.5
	dfadd	0.54	413	101.5	101.2	99.9	101.9	101.8	98.4	102.2	102.9	97.0	105.1	104.0	98.0	115.3	103.2	104.5
	dfdiv	0.52	455	102.2	101.5	100.7	103.1	99.8	98.5	103.5	105.9	99.8	106.2	107.5	100.9	117.4	111.9	111.6
	dfmul	0.46	293	102.7	99.8	100.4	103.4	103.2	100.6	104.8	106.6	101.4	111.3	111.6	108.0	127.6	111.9	121.3
	dfsin	0.60	1230	101.2	99.8	98.9	101.8	100.4	97.6	102.5	102.3	97.0	103.3	108.9	98.8	108.8	109.6	102.2
	gsm	0.58	761	101.6	97.9	98.3	101.3	99.7	96.7	102.0	102.0	96.3	104.7	108.3	100.0	115.2	113.0	110.2
	jpeg	0.55	2201	101.6	100.6	99.7	102.3	100.8	98.3	103.7	99.7	96.7	107.1	102.7	99.2	115.8	108.0	107.8
	mips	0.53	301	102.0	99.6	99.6	101.7	98.5	96.5	102.0	99.5	95.1	109.0	102.2	100.7	124.3	103.6	113.0
	motion	0.49	320	101.9	102.3	100.8	102.5	104.2	100.2	104.4	105.9	100.6	110.9	103.1	103.0	128.4	100.0	114.5
	sha	0.52	844	103.1	100.3	101.0	103.3	96.9	97.2	104.4	99.8	97.4	107.6	104.4	100.6	122.5	108.6	114.4
	Geomean	-	-	-	-	**99.6**	-	-	**97.8**	-	-	**97.4**	-	-	**100.1**	-	-	**109.6**
Select Geomean		-	-	-	-	**98.5**	-	-	**97.4**	-	-	**97.3**	-	-	**100.1**	-	-	**109.6**

TABLE VIII
GEOMEAN FMAX CHANGE FOR EACH ARCHITECTURE FOR SELECT MAPPING RELATIVE TO THE STANDARD TEN 6-LUT ARCHITECTURE FOR EACH BENCHMARK SUITE

		Arch. 1:9	Arch. 2:8	Arch. 3:7	Arch. 4:6	Arch. 5:5
Non-Fracturable	VTR	100.8%	100.7%	100.3%	97.5%	96.5%
	CHStone	101.0%	102.0%	101.2%	102.5%	100.1%
Fracturable	VTR	99.0%	96.5%	95.9%	94.3%	93.0%
	CHStone	99.7%	98.5%	96.5%	95.6%	94.6%

B. Fracturable Architectures: Packing, Place, and Route

The final area estimates in Tables VI and VII show the area reduction relative to the baseline mapper and a traditional fracturable 6-LUT architecture. The same experimental methodology used for nonfracturable architectures is used for the fracturable architectures.

The fracturable architectures, overall, show vastly differing performance compared with the nonfracturable architectures. Using NaturalMux mapping shown in Table VI, the best architectures are the 1:9 architecture for VTR7 and the 2:8 architecture for CHStone. Only up to ~2% area savings are seen for NaturalMux mapping for fracturable architectures. Using MuxMap mapping (Table VII), the best architectures are the 1:9 architecture for VTR7 but now the 3:7 architecture for CHStone. Again, only up to ~2% area savings are seen for MuxMap mapping for fracturable architectures. For Select mapping, we see marginal gains and show the best overall result for both the VTR7 and CHStone benchmark suites. A 1:9 architecture looks best for VTR7 at an overall 2.1% area reduction and a 2:8 or 3:7 looks best for the CHStone suite at 2.6% and 2.7% reduction, respectively.

Recall that the fracturable 6-LUT architecture presented is very flexible—it can map two functions with up to four disjoint inputs each. This flexibility is quite powerful as seen with the reduced performance of the fracturable architectures versus the nonfracturable architectures. The Dual MUX4 element can only map some combinations of three and four input functions, since four inputs to the LE are shared (see Fig. 3). The constraints (pin sharing) on the input pins of the Dual MUX4 restrict many pairs of functions, usually two to four inputs, to be mapped, whereas the fracturable LUT is able to map any pair of functions up to four-inputs each.

Looking again at the function input distributions in Figs. 9 and 10, we can see that over 50% of the functions in each benchmark suite is four-input or smaller. Thus, these four-input or smaller functions are ideal candidates for mapping into the fracturable LUTs, allowing the fracturable LUT to perform very well in terms of logic density. Also from the distributions, there are not many five- or six-input function that can be mapped to MUX4 elements. Thus, most of these elements also require LUTs leaving not many functions left to be mapped to Dual MUX4 elements.

C. Performance

The worst-case delay for both the 6-LUT and MUX4 variants was used to determine the longest delay and FMax. Table VIII shows the geomean FMax change for Select mapping for nonfracturable and fracturable architectures for each benchmark suite. Using the Select mapping results, we are taking the best mapping scheme based on area and then measuring performance. Over all tested architectures, we observed between a 2% improvement to a 7% slowdown. The general trend seems to be that the more aggressive architectures see more slowdown especially for fracturable architectures.

For the nonfracturable architecture, the best area savings were seen in 3:7 and 4:6 ratios. The performance is relatively constant for the 3:7 architecture, while the 4:6 architecture the VTR7 benchmarks have a 2.5% hit and the CHStone benchmarks have a 2.5% gain.

For fracturable architectures, we saw the best area savings in a 1:9 architecture for VTR7 and for performance, we now see a 1% slowdown. For CHStone, based on area, the preferable architecture was a toss-up between a 2:8 and 3:7 architecture but, here, we see that a 2:8 architecture is preferable, since we take a smaller performance hit of 1.5%.

Overall, the performance impact on the architectures that yielded the best area savings were minimal for fracturable architectures, while some gains were seen for nonfracturable architectures.

D. Discussion

The results show that relative to baseline nonfracturable LUT-based LE architectures, the inclusion of hardened MUX4s to form a hybrid LUT/MUX4-based CLB provides an area benefit of up to 8.9% and a 2.5% increase in FMax (for the case of the CHStone circuits, 4:6 architecture and Select mapping). In essence, this result reaffirms conventional wisdom that LUTs are in a sense over-engineered for many logic functions that regularly occur in application circuits. For such functions, one can get away with using a LE with reduced flexibility.

With respect to a baseline of fracturable LUT-based elements, the incorporation of MUX4s offers at most a modest area benefit of 2.6% with a small performance hit of 1.5% (CHStone, with a 2:8 ratio). The notion of making LUTs fracturable is, in fact, a feature whose purpose is rooted in the overengineered nature of nonfracturable LUTs: fracturable LUTs permit area recovery for implementing small logic functions. Thus, the underlying purpose of making LUTs fracturable is not orthogonal to the purpose of incorporating MUX4s, and hence, it is perhaps unsurprising that MUX4s provide a small benefit in this case. Nevertheless, we consider the results encouraging in that they point out that the LUT, while being the long-standing lynchpin of FPGA logic, may not be the only circuit structure capable of delivering high logic density. Other structures, such as MUXs, are worthy of consideration.

VII. 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 logic-density in fracturable architectures.

References

[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 deep-submicron 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 synthesis-inspired 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 field-programmable 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 technology-mapping: 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.

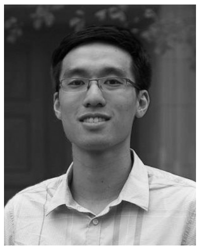
[20] C. Chiasson and V. Betz, "COFFE: Fully-automated transistor sizing for FPGAs," in *Proc. Int. Conf. FPT*, Dec. 2013, pp. 34–41.

[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.



Stephen Alexander Chin received the B.A.Sc. degree in electrical engineering and the M.A.Sc. degree in computer engineering from the University of Toronto, Toronto, ON, Canada, in 2009 and 2013, respectively, where he is currently pursuing the Ph.D. degree in computer engineering.

His current research interests include reconfigurable architectures and related computer-aided design for silicon efficiency.

Jason Luu received the B.A.Sc. degree in computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2007, and the M.A.Sc. and Ph.D. degrees from the University of Toronto, Toronto, ON, Canada, in 2010 and 2014, respectively.

He has contributed eight years to the open source VTR project for field-programmable gate array computer-aided design and architecture research. He is currently a Senior Software Engineer with Altera Inc., Toronto.

Safeen Huda received the B.A.Sc. and M.A.Sc. degrees in electrical engineering from the University of Toronto, Toronto, ON, Canada, in 2009 and 2012, respectively, where he is currently pursuing the Ph.D. degree in computer engineering.

He has been involved in the development of STT-MRAM at the device and circuit level. His current research interests include the development of circuits, architectures, and computer-aided design flows for field-programmable gate arrays, with an emphasis on increasing energy efficiency.

Mr. Huda has held the Natural Sciences and Engineering Research Council of Canada Canada Graduate Scholarship and the University of Toronto Fellowship.

Jason H. Anderson (S'96–M'05) received the B.Sc. degree in computer engineering from the University of Manitoba, Winnipeg, MB, Canada, and the M.A.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Toronto (U of T), Toronto, ON, Canada.

He joined the Field-Programmable Gate Array (FPGA) Implementation Tools Group, Xilinx, Inc., San Jose, CA, USA, in 1997, where he was involved in placement, routing, and synthesis. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, U of T, and holds the Jeffrey Skoll Chair in Software Engineering. He has authored over 60 papers in refereed conference proceedings and journals, and holds 26 issued U.S. patents. His current research interests include computer-aided design, architecture, and circuits for FPGAs.