# Using Island-Style Bi-Directional Intra-CLB Routing in Low-Power FPGAs

Oluseyi Ayorinde, He Qi, Yu Huang, and Benton H. Calhoun
Charles L. Brown Department of Electrical Engineering
University of Virginia
Charlottesville, Virginia
<oaa4bj,hq5tj,yh3cf,bcalhoun>@virginia.edu

*Abstract*— **Increased clustering in Field Programmable Gate Arrays (FPGAs) has shifted a larger fraction of the overall routing load into the configurable logic blocks (CLBs), reducing usage of the costly global interconnect. However, increases in CLB size introduce additional overheads inside CLBs, which can limit the savings gained by minimizing the global interconnect use, motivating more efficient intra-CLB routing. This paper explores different topologies for the intra-CLB connectivity and identifies how the optimal local-CLB interconnect changes for different FPGA architecture and circuit parameters. This work compares area, delay, and energy for two intra-CLB topologies: multiplexer-based routing and island-style bi-directional routing, similar to the global FPGA interconnect, but used inside the CLB (which we call a mini-FPGA). The mini-FPGA style of local CLB interconnect prove to be favorable for minimum-energy operation, as they can reduce transistor count by as much as 62%, and consume as much as 77.9% less energy. Multipexer-based CLBs have performance benefits by reducing delays by almost 3x. Multiplexer-based CLBs can consume less energy at nominal voltages, but only if additional measures are taken to limit power consumption in the multiplexers. A 130-nm CMOS test chip confirms that simulation results track measured data for mini-FPGA CLBs.**

## I. INTRODUCTION

Field Programmable Gate Arrays (FPGAs) are attractive alternatives to Application Specific Integrated Circuits (ASICs) because of their reconfigurability. However, this flexibility comes at the price of increased overhead (in energy dissipation and area) and decreased performance. In a comparison between FPGA and ASIC implementations of the same benchmark circuits, the FPGA was on average 35x larger, 4.5x slower, and consumed 14x more dynamic power than the ASIC [9]. This degradation in efficiency comes from, among other things, a costly interconnect that can account for as much as 70% of the energy dissipation of an FPGA [10]. This overhead becomes even worse at extremely low voltages, making it infeasible to operate FPGAs in many low-power contexts. One way to reduce the dependency of an FPGA on its global interconnect is to increase the clustering, or the number of Basic Logic Elements (BLEs) inside of each Configurable Logic Block (CLB). Increasing the clustering of the CLBs in FPGAs allows logic blocks to communicate with each other (within the CLB), minimizing the use of the costly global interconnect, and thereby increasing the energy efficiency of the FPGA. Clustering has been shown to mitigate performance and area overheads incurred by implementing circuits using FPGAs [2], as well as energy consumption [3]. As a result, many commercial FPGA architectures ([5][6][7]) use cluster-based CLBs.

While increasing the size of FPGA clusters has clear benefits, there are penalties still associated with the larger CLBs. Increasing the cluster size of the CLBs increases their percentage of the energy and delay contributions. At a certain point, increasing the cluster size begins to introduce additional delay, energy, and area overheads without providing any benefits [3]. If then, the CLB can be redesigned to have better performance and be lower energy, designers can push the clustering size higher to get the same delay and energy as the previous design, which will result in fewer connections to the global interconnect, lowering overall energy consumption.

Aside from increasing logic block clustering, energy consumption can be reduced in FPGAs by reducing the supply voltage to a value near the threshold voltage of the transistors (near-threshold) or even lower (sub-threshold). Dynamic energy has a squared dependence on supply voltage, and leakage energy has a linear dependence. Therefore, voltage can be a strong knob for reducing energy consumption of FPGAs. In [4] and [11], researchers push supply voltages into the sub-threshold operating regime, and show very strong energy savings compared to the nominal supply voltage case. Therefore, in our efforts to increase clustering in FPGA logic blocks, it will be important to also characterize these clusters for low-voltage operation in order to understand how and when to use the different CLB topologies in all potential FPGA applications.

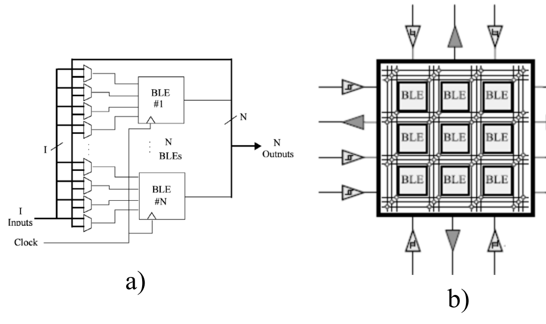


**Figure 1: a) multiplexer-based [1] and b) mini-FPGA [4] Configurable Logic Blocks (CLBs)**

In this paper, we address the dual problems of efficient clustering of CLBs and low-voltage operation by investigating both the widely-accepted multiplexer implementation of a clustered CLB, and using bi-directional island-style routing (similar to the global interconnect of an FPGA, with switch boxes and connection boxes) between the BLEs of a CLB, as introduced in [4] (hereafter referred to as a 'mini-FPGA' CLB). Both of these approaches are illustrated in Figure 1. In this work, we will compare and contrast these two approaches, based on area, delay, and energy consumption. In Section II, we the circuit topologies of the CLBs are discussed in detail. Section III provides a comparison of the area of the two approaches by analyzing transistor counts. Section IV discusses simulation results for delay and energy. In section V, we discuss our hardware implementation of the mini-FPGA CLB. The paper finishes in Section VI with conclusions.

## II. CLB Topologies

The configurable logic block (CLB) is the building block of the island-style FPGA. Clustered CLBs include more than one basic logic element (BLE). Each BLE includes one Look-Up Table (LUT), which is implemented as a collection of $2^k$ SRAM bits and a $2^k$-to-1 multiplexer, where $k$ is the number of inputs to the LUT. The SRAM bits can be configured to represent the truth table of any $k$-input Boolean function. The SRAM values are fed through the $2^k$-to-1 multiplexer, whose select signals are the inputs to the LUT. The output of the LUT is multiplexed with a register, so the BLE can be either combinational or sequential in nature.

A group of these BLEs are clustered together to create the CLB. The standard implementation of a CLB, which is assumed in the VPR tool [14], uses multiplexers to route all of the inputs and outputs of the CLB to each input of each BLE (illustrated in Figure 1a). In our multiplexer-based CLB implementation, we build the BLE input multiplexers out of 2:1 multiplexers. The multiplexers are implemented with transmission gates with a buffer at the output.

Figure 1b illustrates the mini-FPGA CLB, where BLEs are treated like CLBs in an FPGA, connected by a network of local switch boxes, connection boxes, and routing channels. The switch box topology is the 'subset' implementation (as named by VPR [1], which is comparable to the Xilinx XC4000 switch box topology [12]. While this is not the most efficient switch box topology, moving to another topology would require the same or more transistors and would not affect area or energy consumption, despite an increase in routeability. For this topology, there are $W$ switch points per switch box, where $W$ is the channel width (or the number of routing tracks). The connection boxes connect different pins of BLEs to the routing resources around it. Here, we use the same programmable switch as we do in the switch box, and use it to connect each BLE pin to each track in the adjacent routing channel. Thus, we have $W$ programmable switches in each connection box, where $W$ is the width of the routing channel.

## III. Area Comparison

An important metric for evaluating the options for CLB topology is total area. Minimizing the area of the CLB minimizes the lengths of the global interconnect wires, which contribute to most of the delay, area, and energy overhead of an FPGA. For this comparison, we will use the transistor count in the CLB interconnect as a proxy for area, since the BLEs are identical. Transistor count provides an estimate for total area, since full layouts are not available for the large number of designs under consideration and automatic layout generation is not available for both implementation styles. Using transistor count is equivalent to minimum-width transistor estimates (MTEs) [1] because each of the CLB topologies are implemented with minimum-width transistors (which minimize capacitance, and therefore power consumption) resulting in MTE = 1. Sizing for minimal energy could result in varying transistor widths, which would require using MTE calculations in order to have an accurate area estimate. For the purposes of this work, however, transistor counts are sufficient.

The local interconnect of the multiplexer-based CLBs consists purely of the input multiplexers and the configuration bits controlling them. The transistor count for the local interconnect of mini-FPGA CLBs is attributed to the transistors and configuration bits in the switch boxes and the connection boxes of the mini-FPGA. Figure 2 plots the total number of transistors attributed to the local interconnects of both multiplexer-based and mini-FPGA CLBs against clustering. The transistor counts for both fully buffered (e.g. buffers between every transmission gate pair) and unbuffered multiplexers are plotted to show the bounds of possible transistor counts for multiplexer-based CLBs.

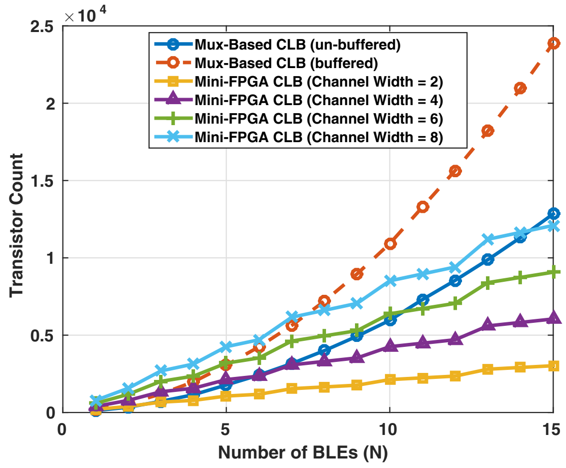


**Figure 2: Total number of transistors vs. # of BLEs for different CLB types. In this comparison, *k* = 4. Multiplexer-based CLBs become very large with high clustering values. The channel width of Mini-FPGAs contributes heavily to the transistor count.**

The number of transistors in the mini-FPGA style CLBs increases linearly with the number of BLEs, whereas the count for the mux-based CLB increases quadratically. The buffered mux-based CLB has more transistors than mini-FPGA CLBs with a channel width of 8 at a cluster width of 8, which is smaller than the clusters of many commercial FPGAs ([5][6][7]) and the best clustering value for CLBs as determined by [2] and [3]. We empirically determined that a channel width of 6 is sufficient for mapping the entirety of the MCNC "Golden 20" benchmarks, which are often used to benchmark FPGAs [8]. At this channel width, the transistor count of the mini-FPGA CLB with 15 BLEs (9072 transistors) is approximately 38% of the buffered multiplexer-based CLB

(23880 transistors) and 87% of the unbuffered multiplexer-based CLB (12840 transistors).

Many current FPGAs employ *depopulation* to reduce the overhead of multiplexer-based CLBs. Instead of connecting the input multiplexers for the BLEs to every input and output of the CLB, the multiplexers only connect to a percentage of those signals, reducing the size of the input multiplexers. [13] and [15] illustrate varying levels of depopulation that reduce area with minimal effect on the critical path delay of the FPGA as a whole. Additional depopulation further reduces area, but starts to affect FPGA performance. Thus, it is important to also address depopulation when comparing multiplexer-based CLBs to the mini-FPGA variety.

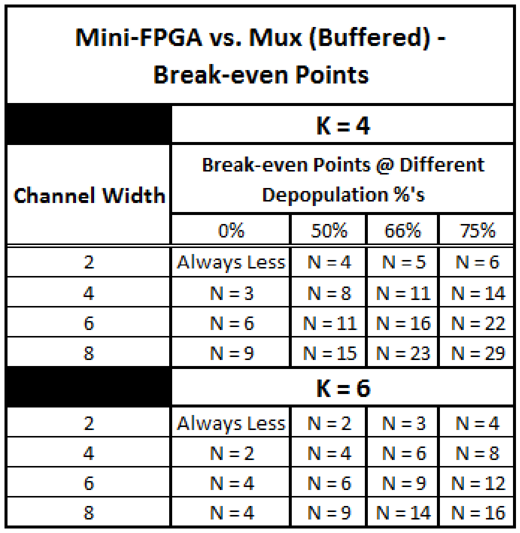
Figure 3 gives the cluster sizes (*N*) at which the multiplexer-based design and the mini-FPGA design have approximately the same transistor count, which will be referred to as "break-even points." Along with taking into account the number of inputs to each BLE (*k*), we vary multiplexer depopulation and mini-FPGA channel width. The depopulation percentages given show the reduction in the number of signals routed (75% depopulation connects to 25% of the total CLB I/Os). At cluster sizes higher than these break-even points, there are smaller area penalties when increasing the area of the mini-FPGA style CLB versus the multiplexer-based design. As the channel width increases, the break-even point happens at larger cluster sizes, as number of switch boxes and connection boxes increases. With 50% depopulation (suggested in [13]), channel width of 6 (suggested by [4]) and $k = 4$, the areas of the two CLB types are equal at a clustering value (*N*) of 11. As depopulation increases beyond 50%, multiplexer-based CLBs decrease in area and the break-even point moves to higher clustering values. The same is true for decreasing mini-FPGA channel width. However, decreasing both parameters reduces routeability in the CLBs. We will discuss this issue further in Section IV.

The multiplexers in this work are built from 2-1 multiplexers, each with a buffer at the output. Thus, the design includes more buffers than are necessary, and removing some of the buffers would reduce the number of transistors for the multiplexers. The mini-FPGA CLB design, however, also introduces additional overhead that is not needed for implementation of most circuit benchmarks. Research into more efficient multiplexer and mini-FPGA interconnect topologies are left for future work.

## IV. Simulation Comparison

While discussion of transistor area is important, it is also important to characterize the effect of CLB implementation strategies on the overall functionality of the FPGA. For the purposes of this comparison, we will observe delay for a notion of CLB performance and energy as a metric for CLB efficiency. To effectively explore the simulation design space, we will simulate at each of the break-even points to determine which CLB topology has better performance and efficiency at equal transistor areas.

Figure 4 shows a flow-chart that describes the process for setting up the simulations. In order to simulate the CLBs we need two things: 1) a simulation netlist for each CLB and 2) the configuration for the CLB netlist in order to represent a functionality that can be observed and tested. First, scripts written in SKILL, the scripting language used by Cadence, create test benches for CLBs. The SKILL script is parameterized by architecture parameters (such as *k*, *N*, etc.), allowing multiple test benches to be generated rapidly. Next, we use the Verilog-To-Routing (VTR) tool-flow to virtually map each CLB with *k*-input chessboard patterns (odd input values return a 1, even input values return a 0) for each BLE. The two inputs necessary for the VTR flow are 1) an architecture file representing the target FPGA fabric and 2) a



| Mini-FPGA vs. Mux (Buffered) - Break-even Points | | | | |
|---|---|---|---|---|
| | K = 4 | | | |
| Channel Width | Break-even Points @ Different Depopulation %'s | | | |
| | 0% | 50% | 66% | 75% |
| 2 | Always Less | N = 4 | N = 5 | N = 6 |
| 4 | N = 3 | N = 8 | N = 11 | N = 14 |
| 6 | N = 6 | N = 11 | N = 16 | N = 22 |
| 8 | N = 9 | N = 15 | N = 23 | N = 29 |
| | K = 6 | | | |
| 2 | Always Less | N = 2 | N = 3 | N = 4 |
| 4 | N = 2 | N = 4 | N = 6 | N = 8 |
| 6 | N = 4 | N = 6 | N = 9 | N = 12 |
| 8 | N = 4 | N = 9 | N = 14 | N = 16 |

**Figure 3: Break-Even points between multiplexer-based and mini-FPGA CLB topologies in terms of transistor count. The break-even points change as a function of the depopulation of the multiplexer-based CLB and the channel widths of the mini-FPGA CLBs. At cluster values (N) higher than those given, mini-FPGAs have smaller transistor counts (hence smaller area) than multiplexer-based CLBs. The break-even points occur at lower *N* values with higher #'s of BLE inputs (*k*).**

Verilog representation of the circuit that the FPGA should implement. In order to stream-line the CLB simulation exploration, we developed Perl scripts to generate both architecture files and Verilog for each CLB we are simulating. For each combination of *k* and *N*, we generate Verilog files that are simply one *k*-input AND gate. That AND gate mapping is changed to the chessboard pattern mentioned above through post-processing with Perl scripts. Each architecture file is an FPGA with a single CLB. That single CLB has architectural parameters (*k* and *N*) that match the target CLB to be tested. Depopulation of multiplexer-based CLBs is addressed in the architecture file, which (among other things) describes the connectivity of the BLE I/Os with the CLB I/Os. For the mini-FPGA style CLB, we leave the architecture file unchanged, maintaining a complete interconnect between BLE and CLB I/Os, but address channel width in the parsing of VTR outputs.

After mapping the CLBs to the two topologies, a custom scripted flow parses the output files of the VTR flow, which provide high-level descriptions of the interconnect and logical mapping of algorithms, and creates initial conditions that can be used to set the configuration bits in the schematics of our CLBs to the proper values to perform the algorithm.

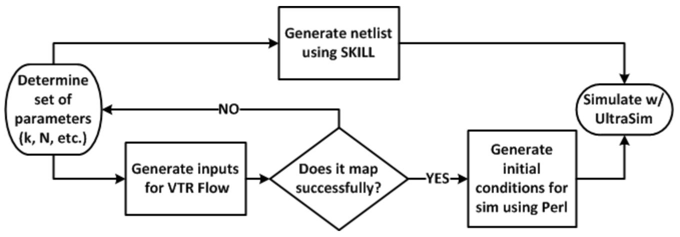

Generate netlist using SKILL
Determine set of parameters (k, N, etc.)
NO
Simulate w/ UltraSim
Generate inputs for VTR Flow
Does it map successfully?
YES
Generate initial conditions for sim using Perl

**Figure 4: Flow for CLB simulations. A custom script extracts information about the logic inside the BLEs of the CLB, their relative location to each other, and the inputs/outputs of the CLB that are used, and creates a set of initial conditions for proper simulation.**

The VTR flow has four output files of interest. The placement file gives a description of the physical location of each block inside the FPGA, including I/Os and logic blocks. The routing file gives high level descriptions of the interconnect channels and pins that are used in the global interconnect of the FPGA. The netlist shows the intra-CLB connectivity between the different BLEs in the CLB, as well as the connectivity between CLBs. Finally, the BLIF (Berkeley Logic Interface Format) file provides the logic internal to the LUTs in the BLE. The scripts can generate the configurations for both multiplexer-style and mini-FPGA CLBs.

In order to create a fair comparison between most of the break-even points, we configured one BLE in each CLB with a chessboard pattern. We then drive the inputs of that BLE with $k$ signals that serve as a binary counter from 0 to $2^k$-1. This ensures that our simulation exercises the worst case delay path through the CLB, because there is a transition in the output at every change of the inputs. We observe the worst-case delay through each CLB, and then calculate energy/op by computing the average power during the worst-case delay, and multiplying the two. We repeat this calculation at a range of voltages from sub-threshold (0.3 V) up to higher voltages (0.8 V). We don't sweep any higher, because our focus is on low energy consumption as opposed to maximum performance.

Large depopulation in the input multiplexers limits the number of inputs that can be routed to each BLE. In this exploration, if the depopulation ever results in a multiplexer with less than $k$ signals, then none of the BLEs can be mapped with a $k$-input function. In the case of the mini-FPGA CLBs, having too few routing tracks prevents multiple signals from connecting to the pins on a given side of any BLE. If there are more used pins on any side of any BLE than there are routing tracks, the Verilog circuit cannot be mapped to a mini-FPGA CLB. Additionally, routing for outputs and inputs often need to span multiple BLEs, further limiting routing resources. As the number of used BLEs increases, more output signals need to be routed to pins, and the available routing quickly becomes insufficient for low channel widths. As we will see later in this paper, only break-even points with N > 4 were successfully routed through the mini-FPGA CLB, due to the restricted routeability. Configuring only one BLE in each CLB ensures that the circuit will be successfully routed to most of the break-even points found in section III, including those with limited routeability. Additionally, we do not simulate the break-even points with no depopulation, as most FPGAs today employ at least some form of multiplexer depopulation. Because the routing switches in the interconnect of the mini-FPGA CLBs have no buffers, they are tied to a higher configuration voltage rail (VDDc), so as to increase the speed of the interconnect with minimal leakage overhead, as suggested in [4]. VDDc is tied to a voltage equal to VDD+0.2 for all simulations.

Figure 5 compares the best energy savings and associated delay savings of both multiplexer- and mini-FPGA CLBs at each of the 24 break-even points discussed previously. 3 of the CLBs (the smallest with k = 6) were unrouteable for both the multiplexer-based and mini-FPGA topologies. Because we are primarily focused on power- and energy-constrained applications, we first find the voltage at which energy consumption is minimized, then determine the delay of the CLB at that voltage for an additional comparison point. For

| K | N | Depopulation (mux-based) | Channel Width (mini-FPGA) | Energy savings w/ mini-FPGA | Delay savings w/ mini-FPGA |
|---|---|---|---|---|---|
| 4 | 4 | 50% | 2 | 20.6% | 3.8% |
| 4 | 5 | 66% | 2 | 61.6% | 81.6% |
| 4 | 6 | 75% | 2 | -10.1% | 4.0% |
| 4 | 8 | 50% | 4 | 4.4% | -232.2% |
| 4 | 11 | 66% | 4 | 0.8% | -233.0% |
| 4 | 11 | 50% | 6 | 6.7% | -228.7% |
| 4 | 14 | 75% | 4 | 6.6% | -226.0% |
| 4 | 15 | 50% | 8 | 5.5% | -235.4% |
| 4 | 16 | 66% | 6 | 16.0% | -247.3% |
| 4 | 22 | 75% | 6 | 5.0% | 6.1% |
| 4 | 23 | 66% | 8 | 5.3% | 5.0% |
| 4 | 29 | 75% | 8 | -11.4% | -10.0% |
| 6 | 2 | 50% | 2 | Unrouteable | |
| 6 | 3 | 66% | 2 | | |
| 6 | 4 | 75% | 2 | | |
| 6 | 4 | 50% | 4 | 36.4% | 78.7% |
| 6 | 6 | 66% | 4 | -17.5% | -9.1% |
| 6 | 6 | 50% | 6 | -33.8% | -9.1% |
| 6 | 8 | 75% | 4 | 64.6% | 76.9% |
| 6 | 9 | 50% | 8 | 10.4% | -6.2% |
| 6 | 9 | 66% | 6 | 7.1% | -7.1% |
| 6 | 12 | 75% | 6 | 7.3% | -6.0% |
| 6 | 14 | 66% | 8 | 24.5% | -256.5% |
| 6 | 16 | 75% | 8 | 77.9% | 0.2% |

**Figure 5: Energy savings from using mini-FPGA CLBs instead of multiplexer CLBs across the different break-even points in CLB area. Overall, mini-FPGA CLBs consume less energy, saving as much as 77.9%. When looking at the delays that correspond to the lowest energy operation for each CLB, multiplexer-based CLBs are faster. Large discrepancies in delay (>200%) are due to the fact that the minimum energy point for multiplexer-based CLBs is sometimes at a higher voltage.**

the most part, using mini-FPGA CLBs minimize energy consumption, reducing energy by as much as 80% compared to the multiplexer-based CLBs. This is due to the exclusion of buffers throughout the interconnect of the mini-FPGA CLB, which greatly reduces the current drawn from the voltage supply. A break from this trends occurs with high depopulation for the multiplexer-based CLB (best-case for energy consumption), but larger channel width for the mini-FPGA CLB (worst-case for energy consumption). One example of this is when $k$ = 4, $N$ = 29, depopulation is 75%, and the channel width is 8. In this case, the multiplexer-based CLB decreases energy consumption by about 11%.

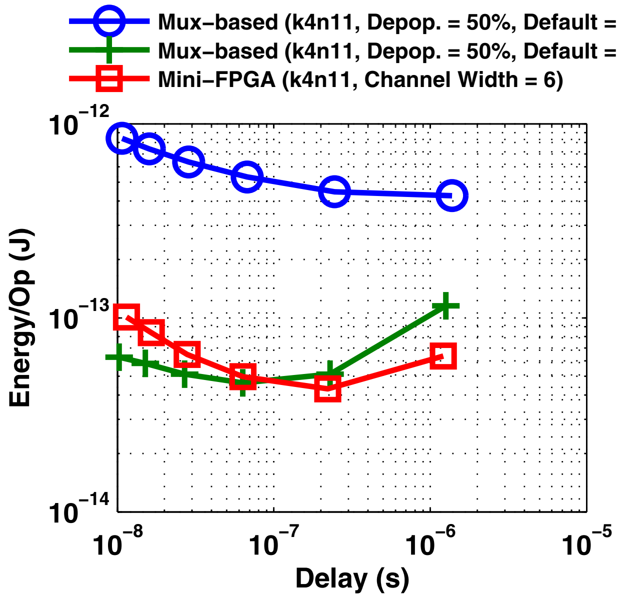


**Figure 6: Energy-Delay (ED) curves for different CLB topologies with $k$ = 4, $N$ = 11, depopulation of 50% for the multiplexer-based CLB and a channel width of 6 for the mini-FPGA CLB. Changing the default configuration for unused multiplexers in the multiplexer-based CLB has a significant effect on energy consumption.**

To observe the mini-FPGA CLB functionality, we fabricated a test chip in a 130-nm process. The chip includes mini-FPGA CLBs, as well as other FPGA test circuits, including routing resources and configuration bits. There are 6 total mini-FPGA CLBs on the chip. We use two switch topologies (pass gate and transmission gate) and vary N between 1, 4, and 9 for six total combinations.

When delay is compared between the two topologies, we find that multiplexer-based CLBs are faster than mini-FPGA CLBs when observing the lowest-energy options. For some of the delays there is a large discrepancy, with delays being faster by over 200%. This is because for these particular architectural parameters, the multiplexer-based CLBs have minimum energy operation at a higher voltage. Thus, the circuit is much faster in these cases, as delay has a strong dependence on operating voltage.

We observe from these simulations that the power consumption of multiplexer-based CLBs has a strong dependence on the default configuration of unused multiplexers. Even though only one BLE inside the multiplexer-based CLB is used, it is still necessary to configure all of the unused blocks as well, to make sure those extraneous blocks have minimal effect on the CLB. For multiplexers, the configuration bits that determine the input are set either to all 0's or all 1's. This choice has a large impact on the energy consumption of the block. Configuring the multiplexer with all 0's chooses CLB input 0 for the multiplexer, whereas all 1's chooses CLB input $X$ (where $X$ is the number of inputs to the multiplexer). If, for example, input 0 is a high-activity net, then every multiplexer not in use will toggle at that high activity, causing the CLB to consume energy as if all $N$ BLEs in the CLB are in use, as opposed to just one. This issue is illustrated in Figure 6 clearly.

When $k$ = 4, $N$ = 23, and the depopulation is 66%, the energy consumption is only 60 pJ when the default configuration for off-multiplexers is all 0's, and goes all the way up to 945 pJ when 1 is the default configuration bit value. The opposite is true when $k$ = 4, $N$ = 22, and the depopulation is 75%. In this case, the high energy consumption (923 pJ) happens when the default configuration is 0's, and the low energy consumption (64 pJ) occurs when the multiplexers are configured with 1's. Thus, in order to use multiplexer-based CLBs efficiently, this issue needs to be addressed, either by power gating the multiplexers, or decoupling the inputs from the multiplexers when the BLE is not in use. As a result of this observation, the table in Figure 5 reports the energy savings based on the lower of the two multiplexer-based CLB energy consumptions.

| K | N | Depopulation (mux-based) | Channel Width (mux-based) | Energy (mux-based) | Energy (mini-FPGA) | Energy Savings (w mini-FPGA) |
|---|---|---|---|---|---|---|
| 4 | 11 | 50% | 6 | 236 pJ | 206 pJ | 13% |
| 4 | 15 | 50% | 8 | 309 pJ | 293 pJ | 5% |
| 4 | 16 | 66% | 6 | 338 pJ | 304 pJ | 10% |
| 4 | 23 | 66% | 8 | 525 pJ | 484 pJ | 8% |
| 4 | 29 | 75% | 8 | 596 pJ | 655 pJ | -10% |

**Figure 7: Comparison of energy consumption of fully-utilized CLBs, where each BLE in the CLB is configured and active. Mini-FPGA CLBs provide lowest overall energy consumption.**

While looking at a single BLE configured in a CLB provides a comparison point that is valid for all of the break-even points observed in the area comparison study, it is not necessarily representative of FPGA implementations generally. More often than not, multiple BLEs inside of a CLB are utilized. Figure 7 shows a table similar to the one in Figure 3, but now only includes break-even points at which both CLB topologies are capable or routing the necessary inputs and outputs when all N BLEs are configured.

We observe that the best-case energy consumptions for the two CLB topologies at break-even points are much closer for the fully utilized CLBs than those with only one BLE in use. At most, mini-FPGA CLBs only save 13% in energy consumption, compared to the 80% observed when utilizing only one BLE in each CLB. Multiplexer-based CLBs look even better compared to mini-FPGA CLBs when compared across voltages. Figure 8 shows the energy-delay (ED) curves for the two CLB topologies at the break-even point where k = 4, N =

11, the depopulation is 50%, and the channel width (W) = 6. While the mini-FPGA CLB has the minimum energy consumption, every other point on that pareto surface (minimizing both energy and delay) belongs to the multiplexer-based design. Thus, for a small relative increase in delay, CLB performance and efficiency are better when using multiplexer-based CLBs.

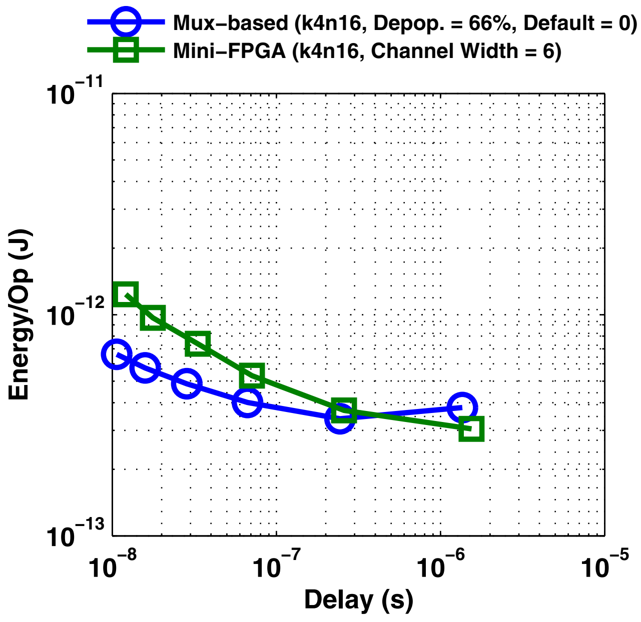


**Figure 8: ED Curves for fully-utilized CLBs at one of the break-even points, with *k* = 4 and *N* = 11. Other fully-utilized CLBs with different k's and N's follow a similar trend. While the mini-FPGA CLB has the lowest energy consumption, the multiplexer-based CLB has better energy efficiency over multiple voltages.**

## V. Hardware Implementation

To observe the mini-FPGA CLB functionality, we fabricated a test chip in a 130-nm process. The chip includes mini-FPGA CLBs, as well as other FPGA test circuits, including routing resources and configuration bits. There are 6 total mini-FPGA CLBs on the chip. We use two switch topologies (pass gate and transmission gate) and vary N between 1, 4, and 9 for six total combinations.

Figure 9 shows the comparison between simulation and measured data for the mini-FPGA CLBs we fabricated. In this plot, we measure the ED curves of 9 BLEs, each configured with a chessboard pattern in the LUTs. The shape of this plot differs from the simulations described earlier because here, the simulation and measurement are for the full chip system, including all of the other blocks and peripheral circuitry. The shape of the simulation and measurement data is very similar, suggesting that conclusions made from simulation results about trends will track hardware implementations accurately. The magnitudes of delay and energy differ, but because the trends are similar, shifting the process corner away from TT (typical NMOS, typical PMOS) will shift the delay and energy consumption for simulations unilaterally. The chip measurements support our SPICE level simulations for illustrating which intra-CLB topologies are best across architectural parameters (*k*, *N*, etc.). Measurements also suggest that the chip was fabricated at a process corner closer to SF (slow NMOS, fast PMOS) then TT, which accounts for the difference in the absolute values.

## VI. Conclusions

By using an FPGA-style local interconnect for the CLB instead of the conventional, multiplexer-based CLB design, transistor counts for the CLBs can be reduced at large cluster values due to a linear dependence on clustering value (*N*), as opposed to the exponential relationship present in the multiplexer design. With larger LUTs (*k*), mini-FPGA CLBs become smaller than multiplexer-based CLBs at smaller *N*. At places in the design space where the area between the two CLB topologies is equal (which we have called *break-even points*) energy is reduced by as much as 80% when using mini-FPGA CLBs. Multiplexer-based CLBs, however, are generally faster than mini-FPGA CLBs at these break-even points, reducing delay by as much as 200% for low-voltage operation. At full CLB utilization, mini-FPGA CLBs still minimize energy consumption at each break-even point when operating in sub-threshold, but multiplexer-based CLBs balance performance and energy consumption better across a larger range of operating voltages. Thus, using mini-FPGA CLBs reduces the area and energy penalties attributed to increased CLB clustering, and makes further clustering (and thus minimal global interconnect routing) for low-power applications more feasible. Hardware fabrication of a test chip shows that trends in CLB simulation match the trend of physical measurements.

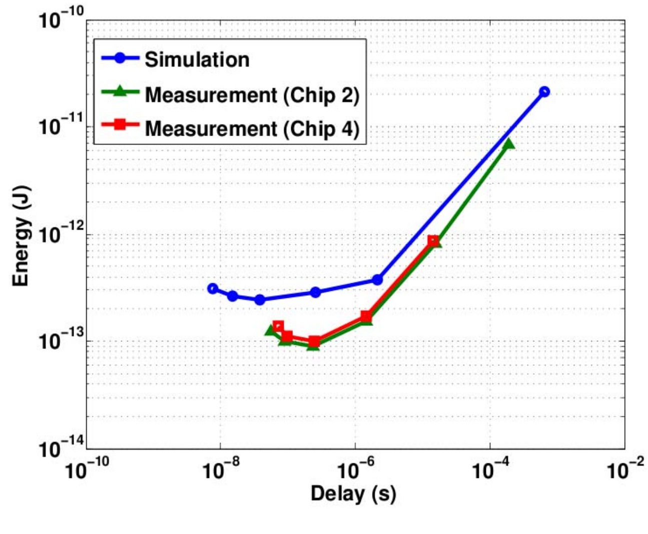


**Figure 9: Comparison between simulated and measured CLB data with similar delay and energy trends. The offset in delay and energy is largely due to the silicon falling at a different process corner.**

## Acknowledgements

This work was partially sponsored by Defense Advanced Research Projects Agency (DARPA) under agreement [HR0011-13-2-0004]. The content, views and conclusions presented in this document do not necessarily reflect the position or the policy of DARPA or the U.S. Government, no official endorsement should be inferred.

## REFERENCES

[1] V.Betz, J.Rose and A.Marquardt. Architecture and CAD for Deep-submicron FPGAs. Kluwer Academic Publishers, February, 1999.

[2] Ahmed, E.; Rose, J., "The effect of LUT and cluster size on deep-submicron FPGA performance and density," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on , vol.12, no.3, pp.288,298, March 2004

[3] Fei Li, Deming Chen, Lei He, and Jason Cong. 2003. "Architecture evaluation for power-efficient FPGAs." In Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field programmable gate arrays (FPGA '03). ACM, New York, NY, USA, 175-184.

[4] Ryan, J. F., and B. H. Calhoun. "A sub-threshold FPGA with low-swing dual-VDD interconnect in 90nm CMOS." CICC. 2010.

[5] Altera Corp. http://www.altera.com

[6] Lattice Semiconductor Corp. http://www.latticesemi.com

[7] Xilinx Corp. http://www.xilinx.com.

[8] S. Yang. Logic synthesis and optimization benchmarks user guide: version 3.0. Microelectronics Center of North Carolina (MCNC), 1991.

[9] Ian Kuon; Rose, J., "Measuring the Gap Between FPGAs and ASICs," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , vol.26, no.2, pp.203,215, Feb. 2007

[10] Fei Li, Yan Lin, and Lei He. 2004. Vdd programmability to reduce FPGA interconnect power. In Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design (ICCAD '04). IEEE Computer Society, Washington, DC, USA, 760-765.

[11] Grossmann, P.J.; Leeser, M.E.; Onabajo, M., "Minimum Energy Analysis and Experimental Verification of a Latch-Based Subthreshold FPGA," Circuits and Systems II: Express Briefs, IEEE Transactions on , vol.59, no.12, pp.942,946, Dec. 2012

[12] Stephen Trimberger. 1995. Effects of FPGA architecture on FPGA routing. In Proceedings of the 32nd annual ACM/IEEE Design Automation Conference (DAC '95). ACM, New York, NY, USA, 574-578.

[13] Guy Lemieux and David Lewis. 2001. Using sparse crossbars within LUT. In Proceedings of the 2001 ACM/SIGDA ninth international symposium on Field programmable gate arrays (FPGA '01), Martine Schlag and Russell Tessier (Eds.). ACM, New York, NY, USA, 59-68.

[14] Jonathan Rose, Jason Luu, Chi Wai Yu, Opal Densmore, Jeffrey Goeders, Andrew Somerville, Kenneth B. Kent, Peter Jamieson, and Jason Anderson. 2012. The VTR project: architecture and CAD for FPGAs from verilog to routing. In Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays (FPGA '12). ACM, New York, NY, USA

[15] Jonathan Greene, Sinan Kaptanoglu, Wenyi Feng, Volker Hecht, Joel Landry, Fei Li, Anton Krouglyanskiy, Mihai Morosan, and Val Pevzner. 2011. A 65nm flash-based FPGA fabric optimized for low cost and power. In *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays* (FPGA '11). ACM, New York, NY, USA, 87-96.