An Improved "Soft" eFPGA Design and Implementation Strategy

Victor Aken'Ova, Guy Lemieux, Resve Saleh
Department of Electrical and Computer Engineering, University of British Columbia
Vancouver, British Columbia, Canada
(vaken, lemieux, res)@ece.ubc.ca

Abstract

A recently proposed "soft" eFPGA methodology was used to create small amounts of programmable logic using the ASIC flow, but it incurs significant overhead. In this paper, it is shown that architecture-specific tactical standard cells can reduce the area and delay overhead of the previous approach by 58% and 40% respectively. It is also shown that by imposing a structured design and layout approach, the logic capacity and quality of standard-cell-based eFPGAs can be significantly improved. Finally, it is shown that our improved ASIC flow approach can create layouts that are competitive with another approach called GILES that uses custom FPGA CAD tools and non-standard cells for tile layout purposes.

1. Introduction

Growing design complexity and cost has forced designers to build programmability into System-on-Chip (SoC) designs to reduce the number of costly chip re-spins, and amortize chip development costs over several design derivatives. Programmability in the form of embedded field programmable gate array (eFPGA) cores is one of a handful of design solutions that has emerged to meet this challenge. Typically, an eFPGA would be used to implement small and medium sized logic functions such as a local accelerator for a processor or I/O interface protocols [05] that can be updated as needed.

Despite the potential benefits of such an approach in SoC design, its widespread use has been hampered by many unresolved difficulties. Not surprisingly, the overhead costs typically associated with programmable logic architectures are of concern. Also related to this problem is the fact that eFPGA cores are currently only available as a library of hard macros or fixed size cores. This presents an additional challenge, because it is possible that in a hard eFPGA library, the most suitable core for a target application is larger than required, and possibly slower and less power efficient. Although these hard cores are designed using full-custom techniques, and offer the best possible transistor density, speed and power for a given eFPGA core size, potential mismatches between the SoC application domain and the size and composition (architecture) of cores available can very quickly erode these benefits [01].

A "soft" eFPGA approach [02] that uses the ASIC flow was developed to address some limitations of hard cores, but it incurs significant overhead due to its use of generic standard cells. In this paper, we show that the use of tactical cells in the ASIC flow greatly reduces the area and delay overhead of standard cell eFPGAs, and that the flexibility afforded by such an approach can result in fabrics that incur significantly less overhead compared to hard eFPGAs. We also show that our approach is competitive with other recent approaches [06,09].

2. Background

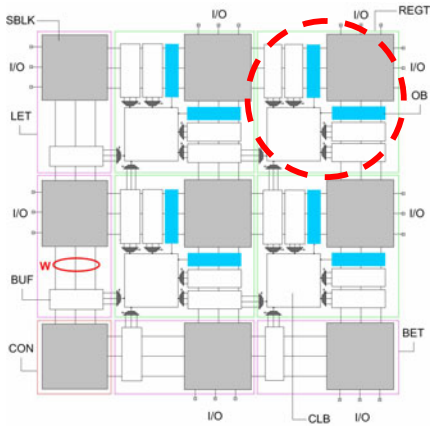
The use of "crafted" cells to improve the quality of generic standard cell ASICs is documented in [04,08]. However, much of this work focused on datapath and arithmetic ASICs not programmable logic. A similar approach was first applied to programmable logic in [01] to implement a one-dimensional reprogrammable fabric for datapath circuits. This work resulted in only 19% and 9% improvements in area and delay respectively. In [06,09] a customized suite of layout CAD tools and non-standard cells was used to generate physical tile layouts for standalone FPGAs. Using this approach, a Virtex-E [06] tile layout estimated to be 36% larger than a full-custom equivalent was generated. A standard cell based Virtex-E tile using some crafted cells had significantly worse area overhead.

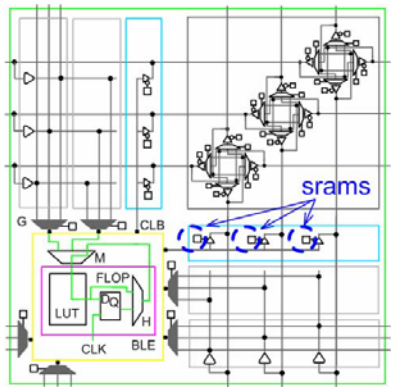
3. Our approach

The approach described in this paper addresses many of the issues that adversely impacted the quality of results achieved in previous approaches [01,02,06]. First, a highly-structured tile-based approach is used during logic synthesis and physical layout to significantly improve the logic capacity and quality of standard-cell-based eFPGAs. In addition, the eFPGA ASIC flow implementation is separated from that of the fixed logic (the rest of the SoC) so that tool optimizations are more focused. However, timing constraints from the top-level design are used to guide eFPGA implementation. In a previous approach [02] the eFPGA logic cells were allowed to "mix" with the fixed logic and this put stress on the tools [02]. Lastly, based on a careful analysis of area and performance bottlenecks due to the use of generic standard cells in eFPGA implementation, architecture-specific tactical standard cells are created to significantly reduce the area and delay overheads.

0-7803-9023-7/05/$20.00 ©2005 IEEE.

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on March 18,2023 at 11:50:42 UTC from IEEE Xplore. Restrictions apply.





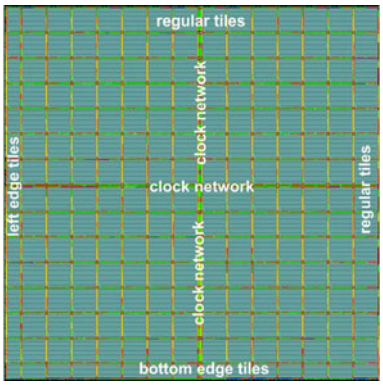


Figure 1: A 2x2 island-style array **Figure 2: Details of island-style tile** **Figure 3: ASIC layout of 14x14 array**

3.1 eFPGA Fabric Architecture

In order to demonstrate the benefits of our approach, a parameterized island-style eFPGA architecture was described in VHDL RTL. A parameterized configuration state machine, and row and column address decoders for programming were also implemented. Figure 1 shows this architecture as a 2 X 2 array of tiles (islands) surrounded by vertical and horizontal routing tracks. Figure 2 shows the logic and routing details of one tile. The island-style architecture offers adjustable parameters that can be changed as needed. This makes it a "configurable architecture". Some parameters of importance are: the array size Dx * Dy, the number of lookup table inputs K, the number of lookup tables per tile N, and the routing channel width W.

3.2 Imposing Layout Structure

A previous approach in [02] was limited in the size of fabrics it could generate, partly because synthesis tools ran out of memory when certain logic thresholds were exceeded. This happens primarily because programmable logic contains an unusually large number of *potential* timing paths, and this puts a strain on memory resources during synthesis. Using the structured island-style architecture above, it is possible to reduce demand on memory and improve runtime, because only a single tile is synthesized and then replicated. As a result, it was possible to increase the equivalent ASIC gate capacity of our island-style fabric from about 400 gates using "flat" synthesis to 4000 gates using a structured approach. It should be possible to create even larger fabrics using this approach.

Further, a structured approach makes it possible to create physical layouts of fabrics in significantly less time than using a "flat" approach. For example, the island-style eFPGA shown in Figure 3 with a logic capacity of 784 4-LUTs was created in about 3 hours using a structured automated layout approach. A flat layout of the same eFPGA took 18 hours to complete.

Finally, our results show an average reduction of 38.5% in inter-tile wire lengths compared to unstructured layouts. This is because a structured layout is "firm" on tile placements and so connections between tiles are reduced to short "hops". This approach resulted in an overall critical path delay (gates plus wire delay) reduction of 3% compared to an unstructured eFPGA. This difference is small because the architecture used here currently supports only "single length" [03] wires. This means channel routing tracks are short because they can span only one tile. As a result, wire delay accounts for only about 11.5% of overall delay. A 3% overall delay reduction is equivalent to a wire delay reduction of 26%. The delay impact of structure in architectures with longer wires will be greater.

3.3 Architecture-Specific Cells

In full-custom "hard" eFPGA designs, circuits are optimized for the architecture being implemented. On the other hand, standard cells used in the ASIC flow are typically generic circuits designed to implement a wide range of logic. As a result, ASICs incur some overhead relative to full-custom designs. For standard cell eFPGAs [01,02], the overhead could be much worse depending on the architecture being implemented. For example, in a LUT-based architecture such as Figure 2, wide-fan-in multiplexers built from generic cells can incur a significant area overhead [02]. In full-custom hard eFPGAs, multiplexers are implemented with NMOS pass-trees to save area. Furthermore, the use of flip-flops rather than SRAM cells as configuration memory [02] increases area

Figure 4 shows area distribution results based on generic standard cells for our island-style eFPGA. As expected configuration memory and multiplexing logic dominate the area and buffers account for the rest. Figure 4(b) shows area distribution for multiplexing logic only. Clearly, the configuration flip-flops that comprise the configuration memory as well as LUTs and LUT input multiplexers (M, in Figure 2) are leading candidates for tactical cell substitution. However, static timing results showed that switch multiplexers [13] account for over 60% of the delay overhead in generic standard cell eFPGAs. This would also makes them very strong candidates for tactical standard cell substitution.

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on March 18,2023 at 11:50:42 UTC from IEEE Xplore. Restrictions apply.

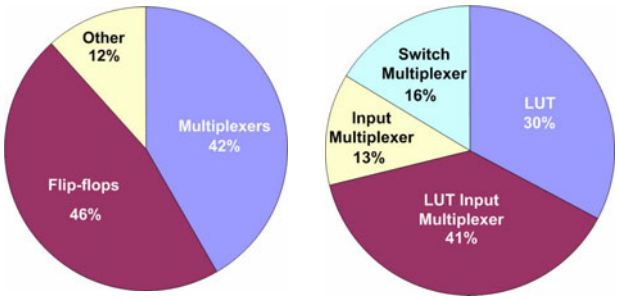


Figure 4: Cell area distribution in island-style eFPGA

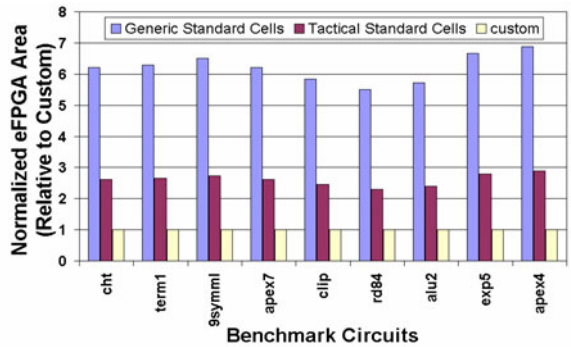


Figure 5: Area savings due to tactical standard cells

Custom layouts were created for 3, 4, and 5-input LUTs, SRAMs, and multiplexers. Table 1 compares these with their generic standard cell equivalents (LUT area includes SRAMs). The results show area savings of 2.5X from SRAMs and area savings of between 3.5X to 7.6X for multiplexers and LUTs.

The layout density achieved for multiplexers is significant given the restrictions within standard cells. In particular, standard cells require PMOS transistors to occupy the upper half of a cell and NMOS transistors occupy the lower half. However, in circuits with many NMOS transistors this leads to low utilization of cell area. To achieve high densities, n-well cutouts in PMOS regions allow more NMOS pass transistors to be accommodated. The n-well "guardband" that remains around PMOS regions ensures these modified cells appear as "normal" cells and do not violate design rules after abutment.

Furthermore, inputs of the multiplexer cell layouts are not buffered, but left as diffusion inputs. This saves area because a single external buffer chain can drive multiplexer inputs that share a source. This is possible for the island-style architecture in Figure 2, because it is known exactly where such sharing occurs it is easy to place suitably sized buffers in the eFPGA tile. The buffer sizes used were calculated using a special case of the Elmore delay equation for distributed RC networks [12].

Cell	Equivalent Standard Cell Area um^2	Custom Tactical Cell Area um^2	Improvement Factor
1-SRAM	61	24	2.5
8:1 Mux	267	77	3.5
16:1 Mux	899	146	6.1
32:1 Mux	2,228	293	7.6
4-LUT	1,875	530	3.5
5-LUT	4,180	1,061	3.9

Table 1: Tactical vs. generic standard cell layouts

4. Experimental Results

VPR 4.30 [03] was used to place & route 9 MCNC circuits and to determine the characteristics of an eFPGA tailored to implement just that circuit. Seven of these circuits range in size from 56 to 200 LUTs, and the two largest occupy 1112 and 1340 LUTs. As explained earlier, we focus on small benchmarks because we assume only small amounts of embedded programmable logic are desired. All results are produced using TSMC's 180nm process.

4.1 Area Results

As shown in Figure 5, tactical cells reduced eFPGA area by 58% on average compared to using generic standard cells. To compare against GILES (custom tools + custom cells), we note that GILES improves area by 33% compared to generic standard cells with their custom SRAMs [06]. Our approach (standard tools + tactical cells) improved area by 42% when compared to generic standard cells with our custom SRAMs. Area results for a "full-custom eFPGA" were also obtained using the area model built in to VPR. This model indicates tactical cell eFPGAs are 2X to 2.8X larger than full-custom eFPGAs. We believe VPR area results are optimistic and may not be attainable, so this can be viewed as a lower bound.

Figure 6 compares our approach to the hard eFPGA cores used in industry. The same hard eFPGA core sizes in [10] are assumed and their area is estimated using data from [11] (after correcting for differences in process technology). We tailor an eFPGA to each benchmark and note that only 2/3 to 1/20 of the area of a single, fixed hard eFPGA is required. This savings is possible because the logic and routing overcapacity present in hard eFPGAs can be avoided by tailoring the architecture to the target function(s). The ASIC flow and tactical cells make this build-to-order approach feasible.

One drawback of this approach is that the tailored eFPGA may not be big enough to accommodate future design changes or completely new circuits. This problem is not easy to solve automatically. Instead, the SoC designer must ensure the eFPGA has enough capacity for the circuits expected by choosing the "largest" one and adding a small margin for error. For example, **apex4** is the largest of the nine benchmarks. A tailored eFPGA which holds **apex4** is 30% smaller than the smallest available hard eFPGA, and it is sufficiently large to implement each of the eight other benchmarks individually. A method to determine robust architecture parameters is needed.

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on March 18,2023 at 11:50:42 UTC from IEEE Xplore. Restrictions apply.

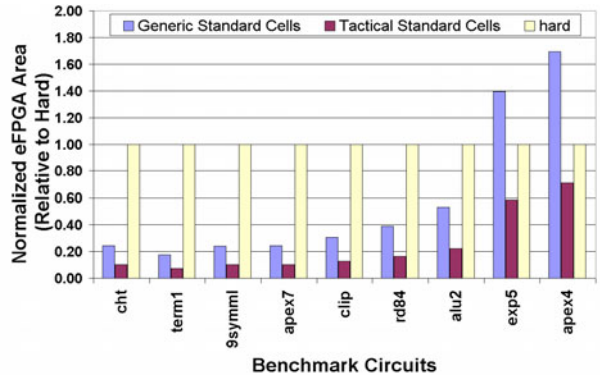


Figure 6: Area comparison to hard eFPGA cores

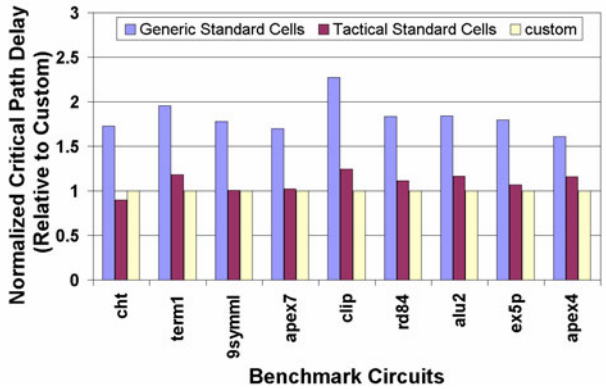


Figure 7: Critical path delay comparisons

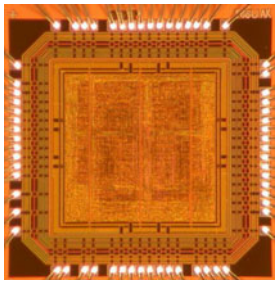


Figure 8: Fabricated eFPGA, 2 x 2 array in 180nm

4.2 Delay Results

In order to measure critical path delay, each eFPGA netlist was "programmed" in PrimeTime using timing path exceptions that are generated based on the program bit stream extracted from VPR. A consistency check has shown that critical paths reported by PrimeTime are often identical to VPR timing reports (custom). For tactical cell eFPGAs, characterized cell delays were used to re-annotate and calculate critical path delays in PrimeTime. Figure 7 shows an average delay improvement of 40% compared to generic standard cells. On average, this is 10% slower than a full-custom eFPGA (custom delays are based on timing estimates obtained from VPR).

Finally, prior work [02] showed a strong timing correlation between PrimeTime and a fabricated eFPGA in the same technology.

4.3 Status

Figure 8 shows a photomicrograph of a fabricated eFPGA chip using standard cells. The chip is currently being tested.

5. Summary

In this paper, it was shown that the "soft" eFGPA approach can be significantly improved by imposing structure and applying architecture-specific tactical standard cells. As a result of imposing structure, the logic capacity achievable with this approach was increased to over 1000 4-LUTs. Tactical standard cells resulted in area and delay reductions of 58% and 40% respectively. For larger circuits (over 1000 4-LUTs), our approach resulted in fabrics that were up to 40% smaller than commercial hard cores, because logic and routing overcapacity can be avoided. Finally, it was also shown that our approach is competitive with the GILES approach used in [06, 09].

Acknowledgments

This work is sponsored by PMC-Sierra and NSERC (including Grant No. STPGP 257684). The authors also thank Dr. Steve Wilton, James Wu, Zion Kwok, and Brad Quinton for comments.

Bibliography

[01] S. Phillips, S. Hauck "Automatic Layout of Domain specific Reconfigurable Subsystems for System-on-Chip" FPGA 2002.

[02] S. Wilton, et. al "Design Considerations for Soft Embedded Programmable Logic Cores", IEEE Journal of Solid-State Circuits, vol. 40, no. 2, pp.1–9, Feb. 2005.

[03] V. Betz, J. Rose, A. Marquardt, *Architecture and CAD for Deep Submicron FPGAs*, Kluwer Academic Publishers, 1999.

[04] D. Chinnery, K. Keutzer,, *Closing the Gap between ASIC and Custom*, Kluwer Academic Publishers, 2002.

[05] A Cappelli et. al "XiSystem: a XiRISC-Based SoC with a Reconfigurable IO module" IEEE ISSCC, February 2005.

[06] I. Kuon, "Automated FPGA Design, Verification and Layout" Master of Applied Science Thesis, University of Toronto 2004.

[07] S. Yang, "Logic Synthesis and Optimization Benchmarks, Version 3.0", MCNC, 1991.

[08] W. Dally, A. Chang, "The Role of Custom Design in ASIC Chips", DAC 2000.

[09] K. Padalia, R. Fung, M. Bourgeault, A. Egier, J. Rose "Automatic Transistor and Physical Design of FPGA Tiles From An Architectural Specification", FPGA 2003.

[10] Actel Corp., "Varicore™ Embedded Programmable Gate Array Core 0.18um Family", Datasheet Rel. 2.2, Dec. 2001.

[11] P. Zuchowsky et. al "A Hybrid ASIC and FPGA architecture", ICCAD 2002, pp187–194.

[12] J. Rubenstien, P. Penfield, M. Horowitz "Signal Delay in RC Networks" IEEE Transactions on Computer Aided Design, vol. CAD-2, pp202-211, July 1983

[13] G. Lemieux, D. Lewis, *Design of Interconnection Networks for Programmable logic*, Kluwer Academic Publishers, 2003

Authorized licensed use limited to: TUBITAK ULAKBIM. Downloaded on March 18,2023 at 11:50:42 UTC from IEEE Xplore. Restrictions apply.