A Case for Heterogeneous Technology-Mapping: Soft versus Hard Multiplexers

Madhura Purnaprajna and Paolo Ienne
Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland
School of Computer and Communication Sciences
CH-1015 Lausanne, Switzerland
madhura.purnaprajna@epfl.ch and paolo.ienne@epfl.ch

***Abstract*—Lookup table-based FPGAs offer flexibility but compromise on performance, as compared to custom CMOS implementations. This paper explores the idea of minimising this performance gap by using fixed, fine-grained, non-programmable logic structures in place of *lookup tables* (LUTs). Functions previously mapped onto LUTs can now be diverted to these structures, resulting in reduced LUT usage and higher operating speed. This paper presents a generic heterogeneous technology-mapping scheme for segregating LUTs and hard logic blocks. For the proof-of-concept, we choose to isolate multiplexers present in most general-purpose circuits. These multiplexers are mapped onto hard blocks of multiplexers that are present in existing commercial FPGA fabrics, but often unused. Since the hard multiplexers are already present, there is no additional performance or area penalty. Using this approach, an average reduction in LUT usage of 16% and an average speedup of 8% has been observed for the VTR benchmarks as compared to the LUTs-only implementation.**

I. Introduction and Related Work

FPGAs have emerged as popular programmable devices, being used in a wide range of applications—embedded systems, space applications, high-performance computing machines, etc. The versatility of the FPGA lies in its basic programmable element, which is the *lookup table* (LUT). A K-input LUT can implement any K-input function, although at the cost of area and reduced performance in comparison to custom CMOS implementations. The relative area and delay ratios for FPGAs as compared to custom CMOS implementations is 18× and about 3–4× respectively [1]. As alternatives to LUTs, logic structures that improve the area-efficiency of the FPGA fabric have been studied [2], [3], [4]. Anderson et al. [2] replace a K-input LUT with an extended (K-1)-input LUT, which requires much smaller area than the implementation with K-input LUT and improves logic density with a minimal impact in circuit speed. Another study uses programmable macro-gates alongside LUTs [3]. These programmable macro-gates are composed of the most frequently occurring functions, which are extracted by analysing a set of benchmarks. Cong et al. [4] present another alternative to LUTs with PLA-like cells, or k/m-macro-cells. Overall, a common issue in all these approaches is that introducing such macro-gates or macro-cells may necessitate a complete redesign in the organisation of logic and routing in the FPGA fabric.

Although very versatile, LUTs can be very inefficient for certain very commonly used circuit components. For instance, multiplexers are generic structures found widely in most logic circuits [5], but are very inefficient on LUTs. A detailed comparison between FPGA-based multiplexers and an analytical model of a switch-based tree multiplexers reveals a delay ratio in the range of 40–75× for small isolated multiplexer [6]. In terms of area of the multiplexer, the cost of programmability is nearly 50× the equivalent fixed structure. Our focus is on exploring the role of multiplexers in general-purpose circuits—a choice between soft and hard logic.

In this paper, a heterogeneous technology-mapping scheme is presented that generates a netlist with the optimal combination of programmable LUTs and non-programmable hard blocks. As a proof-of-concept, we treat multiplexers as fixed logic structures and map them onto dedicated blocks present in most commercial FPGA fabrics (typically unused). The scheme has been applied to the general-purpose circuits in the VTR benchmarks [7]. The organisation of the paper is as follows. Section II presents the new design flow of extracting non-programmable fixed structures alongside LUTs to generate a heterogeneous netlist. Section III discusses the experimental framework and analyses the performance benefits using the heterogeneous technology-mapping flow and Section IV concludes the paper.

II. Heterogeneous Technology Mapping

In a typical FPGA design flow, the functionality of the application is decomposed into logic functions that are mapped onto LUTs. In our design methodology, the objective is to use dedicated hard structures to implement logic functions that are expensive when implemented using LUTs. This necessitates transforming a technology-independent netlist into a heterogeneous netlist comprising programmable LUTs and hard structures. In this context, the challenge is to find the right choice of resource (LUT or hard logic block) that reduces the overall area and also improve speed. For the proof-of-concept, multiplexers are chosen to be extracted as fixed, non-programmable structures in this paper. The design flow is generic and can be applied to identify other candidate structures by computing the truth-table for the structure.

978-0-7695-4969-9/13 $26.00 © 2013 IEEE
DOI 10.1109/FCCM.2013.19

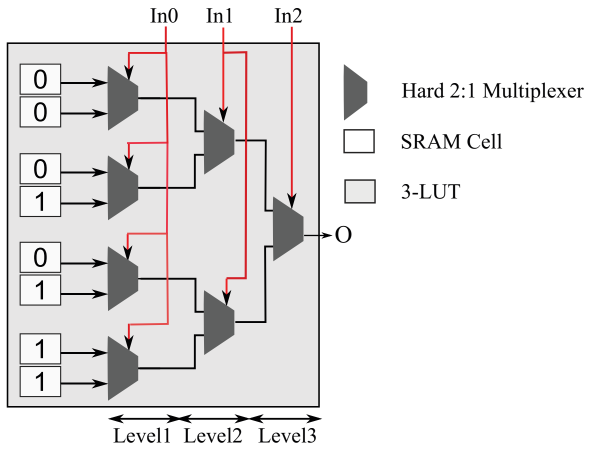


Figure 2: A 3-LUT can implement a 2:1 multiplexer, but by itself has three levels of 2:1 multiplexers to address the eight configuration bits. Hence, if the 2:1 multiplexer accounts to one logic level, the 3-LUT has three such logic levels. The ratio of 1:3 holds even for higher order multiplexers

The technology-mapping algorithm is typically cut-based, where a cost function prioritises feasible cuts that can be accommodated in a K-input LUT [8]. In the case of heterogeneous technology-mapping, a choice has to be made between the soft (the LUT) or the hard logic (dedicated multiplexer). For every cut with a known functionality (or known truth-table) a special cost function decides the choice of the right resource. As shown in Figure 1, the cost associated with the heterogeneous logic structures introduces a significant change in the composition of the target netlist.

The open source synthesis and technology-mapping tool ABC [9] had to be modified to generate a heterogeneous netlist comprising LUTs and multiplexers. ABC supports only two targets for technology-mapping, viz., the LUTs-only version and standard-cell mapping. The scheme of heterogeneous technology-mapping presented in this paper, is a combination of the two—it extends the LUTs-only version by introducing standard-cell-like fixed structures. All modifications have been made to the command *if* that identifies priority-based candidate cuts. A special option within the *if* command (called *(if -M)*) isolates multiplexers using Boolean matching on all candidate cuts. On identifying a match, it also becomes necessary to identify the corresponding input pin permutation for the truth-table to ensure the logic equivalence with the original netlist. The logical equivalence checks of the pre- and post-technology mapped netlists confirms that the functionality of the transformed netlist is not altered.

A. Delay Model: LUT versus Hard multiplexer

The relative delay of a LUT to a multiplexer was modelled using the technology-independent representation of a LUT. A LUT by itself is composed of SRAM cells followed by cascaded stages of 2:1 multiplexer. Figure 2 shows a 3-LUT configured as a 2:1 multiplexer. As can be seen, the 3-LUT has 3 stages of 2:1 multiplexers. As a result, the multiplexer implemented using a LUT has a delay that is 3× the delay of the dedicated 2:1 multiplexer. The same ratio of 3× holds even for higher order multiplexers. For example, a 4:1 multiplexer can be implemented using a 6-LUT, which has 6-levels of 2:1 multiplexers. The dedicated 4:1 multiplexer itself has two levels of 2:1 multiplexers. Although simplistic, our delay assumptions can be considered very conservative, since typical delay ratios for multiplexers are generally significantly higher. Rose et al. [10] infer a delay ratio 7–11× for LUTs versus dedicated multiplexers. Further, small isolated multiplexers have even higher delay ratios, being in the range of 40-75× [6].

B. Dedicated Multiplexers in the FPGA fabric

Figure 3 shows the slice architecture in the Xilinx Virtex-6 family of FPGAs. Dedicated multiplexers merge two K-input LUTs to build a larger K+1-input LUT. In addition, these multiplexers also improve the efficiency of implementing 8:1 and 16:1 multiplexers, provided the first level of LUTs that feed the dedicated multiplexers to be configured as 4:1 multiplexers [11]. However, 4:1 multiplexers are still mapped onto LUTs. In general-purpose circuits, the number of 4:1 multiplexers exceeds the number of 8:1 and 16:1 multiplexers and they do not benefit from the existing dedicated multiplexers within the Xilinx tool chain. For the case of heterogeneous technology-mapping, our objective is to make use of these multiplexers to map 4:1 multiplexers present in most applications. Since there is no possibility to alter the scheme within the Xilinx design tools, it was impossible to introduce heterogeneous technology-mapping within Xilinx design flow. However, the architecture was well suited for mapping netlists comprising LUTs and multiplexers. Consequently, we modelled the architecture in VPR and used the modified design flow to pack, place and route the heterogeneous netlist generated.

III. Experiments and Results

Modifications were introduced in ABC to generate the heterogeneous netlist comprising LUTs and multiplexers. This netlist was forwarded to VPR [7] to perform packing, placement and routing. To adhere to timing and area specifications, the Virtex-6 architecture was modelled using the VPR architectural specification in xml with the timing specification for the 40nm technology (for LUTs) and the technology-independent model described in Section II-A (for multiplexers). Experiments were conducted on the VTR benchmarks that comprise a diverse set of general-purpose circuits. In this case, an average reduction in the LUT requirement of 16% was observed (up to 45% for stereovision3), as shown in Figure 4. The average speedup was about 8% (up to 22% for stereovision1), as shown

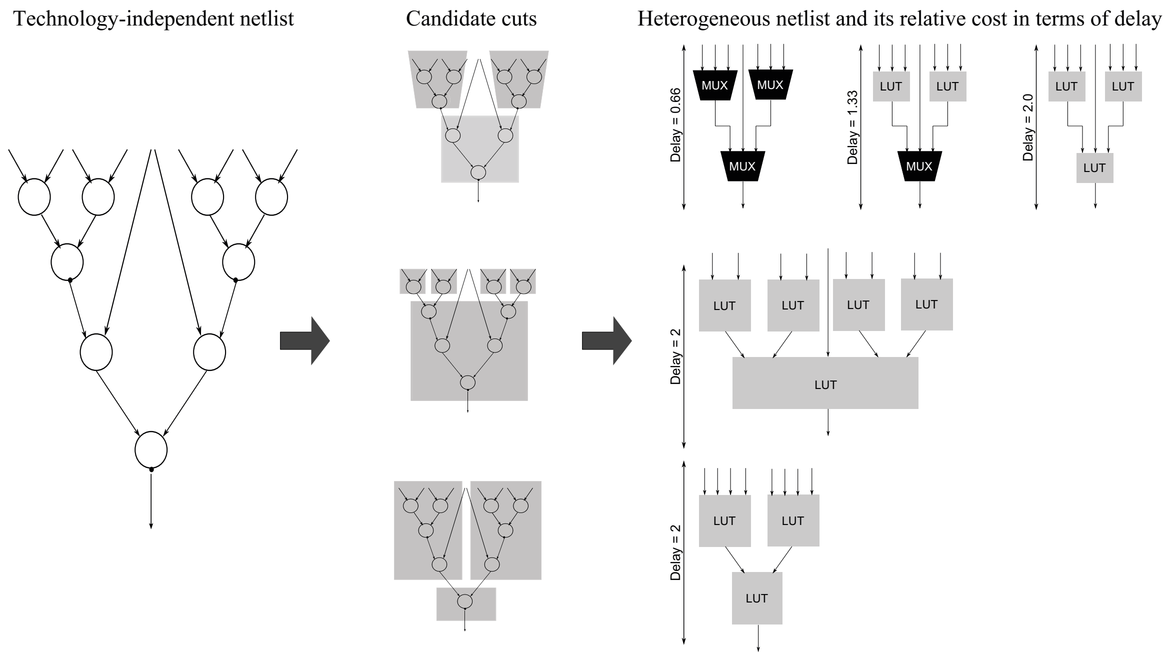


Figure 1: Heterogeneous technology-mapping flow extends the existing LUT synthesis and generates a target netlist composed of LUTs and multiplexers using a cost function for area and time.

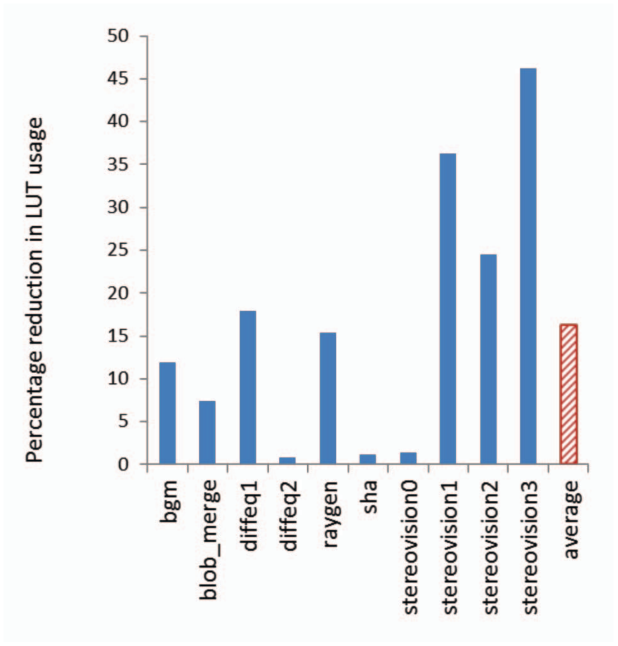


Figure 4: The VTR Benchmarks show an average reduction of 16% in the LUT requirement by using hard multiplexers in comparison to the LUTs-only implementation.

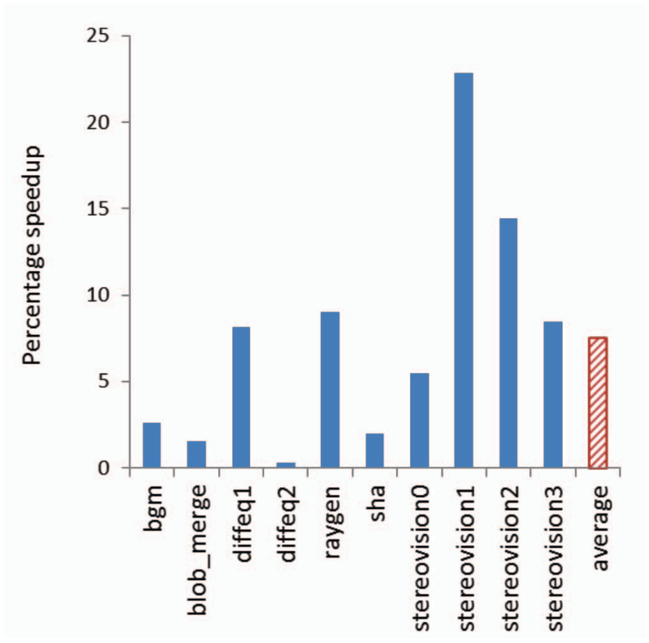
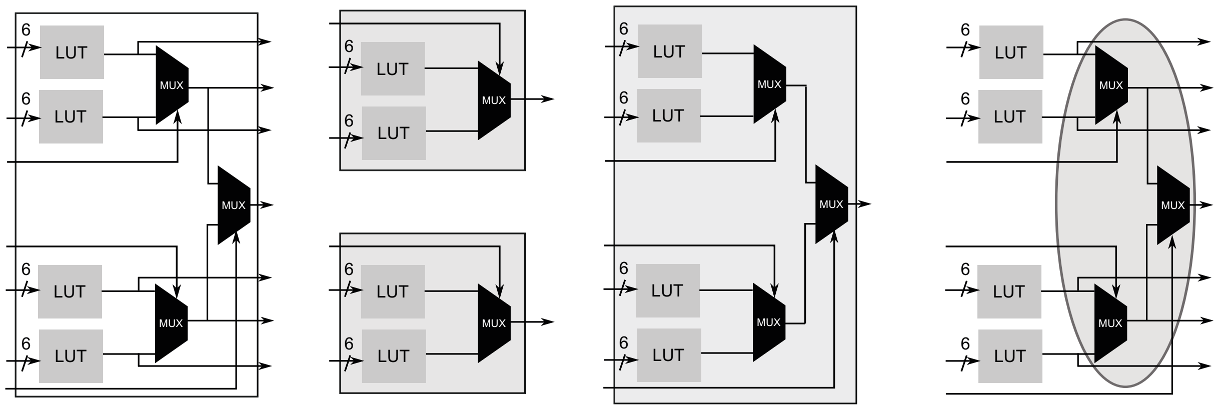


Figure 5: The VTR Benchmarks shown an average speedup of 8% using multiplexers integrated as hard logic, in place of the the LUTs-only implementation

in Figure 5. Considerable speedup is observed by using dedicated multiplexers. The improvement in speed is on account of two factors. Firstly, a dedicated multiplexer is significantly faster than LUT (as discussed in Section II-A). Secondly, the dedicated multiplexer is an additional resource (previously unused) that is present at close proximity to the LUT. This also results in a compact design with reduced in the routing area.



(a) Slice with four 6-LUTs and three dedicated 2:1 multiplexers (b) Configured as two 7-LUTs (c) Configured as one 8-LUT (d) Heterogeneous Technology-Mapping: Configured as four 6-LUTs and one 4:1 multiplexer

Figure 3: Hard multiplexers in the Virtex-6 family of FPGAs are integrated alongside LUTs as shown in (a). Flip flops are not shown for simplicity. The current design tools solely use these multiplexers to extend 6-LUTs to build 7-LUTs (or 8:1 multiplexer) & 8-LUTs (or 16:1 multiplexers) in the user logic, as shown in (b) and (c) respectively. Shown in (d) is our approach to use the hard multiplexers exclusively for mapping 4:1 multiplexers segregated by the heterogeneous technology-mapping scheme.

IV. Conclusions

In this paper, our objective was to enhance performance and resource efficiency by diverting functions that were previously mapped on LUTs onto fixed, non-programmable structures. We present a heterogeneous technology-mapping phase that derives the optimal combination of LUTs and non-programmable structures. Since a non-programmable structure is significantly smaller and faster than the version implemented using LUTs, it improves the operating speed. For the proof-of-concept, we isolate multiplexers and map them onto hard multiplexers already present in current FPGAs. For general-purpose circuits the average speedup is 16% and the reduction is LUT usage is 8%. The scheme presented is generic (not limited to multiplexers) and was developed to exploit the idea of augmenting LUTs with frequently occurring generic logic functions.

References

[1] I. Kuon and J. Rose, "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.

[2] J. H. Anderson and Q. Wang, "Area-efficient FPGA logic elements: architecture and synthesis," in *Proceedings of the 16th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '11, 2011, pp. 369–375.

[3] Y. Hu, S. Das, S. Trimberger, and L. He, "Design, synthesis and evaluation of heterogeneous FPGA with mixed LUTs and macro-gates," in *Proceedings of the 2007 IEEE/ACM International Conference on Computer-aided design*, ser. ICCAD '07, 2007, pp. 188–193.

[4] J. Cong, H. Huang, and X. Yuan, "Technology mapping and architecture evalution for k/m-macrocell-based FPGAs," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 10, pp. 3–23, January 2005.

[5] P. Metzgen and D. Nancekievill, "Multiplexer restructuring for FPGA implementation cost reduction," in *Proceedings of the 42nd Annual Design Automation Conference*, ser. DAC '05, 2005, pp. 421–426.

[6] M. Alioto and G. Palumbo, "Interconnect-Aware Design of Fast Large Fan-In CMOS Multiplexers," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 54, no. 6, pp. 484–488, june 2007.

[7] J. Rose, J. Luu, C. W. Yu, O. Densmore, J. Goeders, A. Somerville, K. B. Kent, P. Jamieson, and J. Anderson, "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*, 2012, pp. 77–86.

[8] S. Cho, S. Chatterjee, A. Mishchenko, and R. Brayton, "Efficient FPGA mapping using priority cuts," in *poster, Proceedings International Symposium on Field programmable Gate Arrays*, 2007.

[9] A. Mishchenko *et al.*, "ABC: A System for Sequential Synthesis and Verification," http://www.eecs.berkeley.edu/~alanmi/abc, 2009.

[10] H. Wong, V. Betz, and J. Rose, "Comparing FPGA vs. custom CMOS and the impact on processor microarchitecture," in *Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, 2011, pp. 5–14.

[11] *Virtex-6 FPGA User Guide*, Xilinx, April 2011, available from http://www.xilinx.com.