

Parameter Sensitivity in Virtual FPGA Architectures

Peter Figuli¹✉, Weiqiao Ding¹, Shalina Figuli¹, Kostas Siozios²,
Dimitrios Soudris³, and Jürgen Becker¹

¹ Institute for Information Processing Technology, Karlsruhe Institute of Technology,
Karlsruhe, Germany

{peter.figuli,shalina.ford,becker}@kit.edu, weiqiao.ding@student.kit.edu

² Department of Physics, Aristotle University of Thessaloniki, Thessaloniki, Greece
ksiop@auth.gr

³ School of Electrical and Computer Engineering,
National Technical University of Athens, Athens, Greece
dsoudris@microlab.ntua.gr

Abstract. Virtual FPGAs add the benefits of increased flexibility and application portability on bitstream level across any underlying commercial off-the-shelf FPGAs at the expense of additional area and delay overhead. Hence it becomes a priority to tune the architecture parameters of the virtual layer. Thereby, the adoption of parameter recommendations intended for physical FPGAs can be misleading, as they are based on transistor level models. This paper presents an extensive study of architectural parameters and their effects on area and performance by introducing an extended parameterizable virtual FPGA architecture and deriving suitable area and delay models. Furthermore, a design space exploration methodology based on these models is carried out. An analysis of over 1400 benchmark-runs with various combinations of cluster and LUT size reveals high parameter sensitivity with variances up to $\pm 95.9\%$ in area and $\pm 78.1\%$ in performance and a discrepancy to the studies on physical FPGAs.

Keywords: FPGA · Virtualization · Cluster size · LUT size · Efficiency

1 Introduction

During the last three decades, Field Programmable Gate Arrays (FPGAs) have evolved from less competitive and prototyping devices with as little as 64 logic cells towards complex System on Chip (SoC) and massive parallel digital signal processing architectures. The functional density alone, however, is not the unique selling point and there is still a considerable gap to ASICs in this regard. Moreover, it is the flexibility and the comparably short design times along with low NRE costs and low risks that make FPGAs so attractive. Currently, we are witnessing a new movement towards general purpose computing. The signs are conspicuous considering the facts that (1) there is a trend towards heterogeneous

reconfigurable SoCs, (2) recently Intel as a major General-Purpose Processor (GPP) company acquired Altera and (3) there are serious efforts to employ FPGAs in data centers and cloud services, e.g. Intel Xeon+FPGA Integrated Platform [8] or the Microsoft Catapult project [13]. At this rate, FPGAs will become mainstream in the future and indispensable in our day-to-day systems and applications such as entertainment, communication, assistance, automation, cyber-physical systems, cloud services, monitoring, controlling, and many more. There will be the situation that FPGA based devices and applications change more often than how it is today, thereby making it necessary to loosen the bond between application and the execution platform.

Virtualization can be a key for instant **portability and migration** of applications even on bitstream level without redesigning or recompiling. Thereby, an optimized reconfigurable architecture as a virtual layer can be mapped onto an existing Commercial Off-The-Shelf (COTS) FPGA, while the application itself will be executed on the virtual layer, thus being independent of the underlying physical platform. We call this technique *virtual FPGA*. The eminent advantage is that the specification of the virtual architecture can persist, while the hosting physical platform can be exchanged by another one. Furthermore, virtual FPGAs can be utilized to (1) enable independent reconfiguration mechanisms, (2) prototype novel FPGA architectures without physical implementation and (3) emulate custom reconfigurable architectures. Despite a few related works [5,6,9,10,12], the field of virtual FPGAs is still considered unexplored. The design space gets extended by a new dimension as the virtual FPGA has the added flexibility to alter not only the application circuit but also the executing architecture. In this regard, the mapping efficiency of applications which is highly related to architectural parameters, is getting very important, especially as the additional layer adds a considerable overhead to the underlying platform. The practice of adopting parameter recommendations intended for physical FPGAs to the virtual domain is questionable as explained in this paper. Yet, due to lack of separate and detailed studies, they have been predominantly followed more or less blindly, accepting that it might not be the optimum solution.

The scope of this paper is to close these gaps and to examine the impact of main architectural parameters of virtual FPGAs on area and performance. Therefore, we propose a suitable design space exploration methodology with area and delay models representing the virtual layer and its realization, which can differ from platform to platform. The contributions of this paper are:

- introduction of an extended highly parameterizable version of *V-FPGA*
- analytical area and delay models for virtual FPGA architectures
- parametric design space exploration
- analysis of parameter tuning and resulting area and performance variance

The rest of the paper is organized as follows: Sect. 2 summarizes the related works. In Sect. 3 we introduce the extended *V-FPGA* architecture. Section 4 derives the area and delay models while the methodology for parametric design space exploration is presented in Sect. 5. Experimental results are presented in Sect. 6 and the conclusions are summarized in Sect. 7.

2 Related Works

2.1 Virtual FPGA Architectures

In [10] Lagadec et al. present a toolset for generic implementation of virtual architectures. The main focus of their work lies on the generic tool flow for architectural representation and place & route of application netlists onto various abstract virtual architectures. The virtualization aspects from hardware perspective, the mapping onto the underlying platform, the programming mechanisms and configuration management remain predominantly unaddressed.

Lysecky et al. introduced in [12] a simple fine grain virtual FPGA that is specifically designed for fast place and route. The architecture has a mesh structure with fixed-size Configurable Logic Blocks (CLBs) being connected to Switch Matrixes (SMs) as opposed to architectures where logic blocks are connected directly to the routing channels. The *V-FPGA* architecture used in this paper has a similar granularity and architecture class as the work of Lysecky et al. but is generic and highly flexible, thus it can take over different shapes and be tailored towards the application needs.

In [6, 9] we introduced a scalable island style virtual FPGA architecture with the primary focus on adding new features to an underlying FPGA, that are not supported natively. More specifically, we achieved to enable partial and dynamic reconfiguration on a flash based Actel ProASIC3 device. The *V-FPGA* architecture presented in this paper builds upon this preliminary work, yet offering a higher flexibility with a rich parameter set.

The ZUMA architecture by Brant and Lemieux [5] is a clustered LUT based FPGA with island style topology and targets to reduce the area overhead of the virtualization layer by utilizing LUTRAMs of the underlying platform. *V-FPGA* follows a different ideology as it concentrates on portability and easy mapping even onto ASIC processes, thus renouncing platform exclusive element usage of ZUMA approach.

The major drawbacks of all virtual FPGA architectures, including the one used in this paper, are higher chip area and larger path delays compared to physical FPGAs. This is due to the fact that each virtual logic cell is realized by a multitude of programmable logic cells of the underlying physical platform. The area overhead of virtual FPGA mainly depends on the granularity of the underlying platform as well as how well the virtual resources can be matched by the physical resources. Thus, the same Virtual FPGA has a different area efficiency on one underlying platform than on another and a change in its design parameters can turn the game.

2.2 Effects of Architectural Parameters on Area and Performance

With respect to architectural parameter choice in physical FPGAs, [3, 14] indicate that a LUT size K between 3 and 4 provides the best area efficiency, while $K = 6$ gives the best performance. [7] shows similar results through a theoretical model, while [15] indicates that $K = 6$ is the best choice for area, delay

and area-delay product in nanometer technology. However, those results are an average and in virtual architectures the area efficiency and performance depend also on how efficient a K -input LUT can be realized by the underlying platform, thereby making their recommendations not highly applicable.

Betz and Rose studied in [4] the relationship between cluster size and required number of inputs per CLB and also the optimal cluster size for 4-input LUTs. Later, Ahmed and Rose extended this study in [3] by varying both LUT size K and cluster size N in the range of $K = 2..7$ and $N = 1..10$ and concluded that $K = 4..6$ and $N = 3..10$ provide the best trade-off between area and delay. The findings in [3] have become a widely accepted reference and guideline in parameter choice for many academic FPGA architectures. While the experiments mentioned above rely on area and delay models on transistor-level, and thus scale smoothly, virtual FPGAs are mostly platform independent and the base units are multiplexers and flip-flops realized by underlying logic blocks. Consequently, these differences can lead to unmatched proportions in logic area, local routing area and global routing area as well as in path delays.

3 The Generic *V-FPGA* Architecture

The V-FPGA is a generic LUT-based FPGA architecture that can be mapped onto existing commercial off-the-shelf (COTS) FPGAs, such as Xilinx, Altera, Microsemi, etc. This extensively scalable and parameterizable architecture is implemented in a fully synthesizable HDL code, utilizing hierarchy, modularity and generics. As illustrated in Fig. 1, the applications will be mapped and executed on the virtual layer rather than on the logic layer of the underlying COTS FPGA. The merit of this approach is that the specification of the virtual FPGA stays unchanged independent to the underlying hardware and adds new features such as dynamic reconfigurability which is for example not available with all COTS FPGAs. It also entitles the re-use of hardware blocks on other physical FPGA devices and enables portability of unaltered bitstreams among different FPGA manufacturers and device families, e.g. in order to overcome the problem of device discontinuation. In the following subsections the structure of the architecture and its tuneable parameters are presented.

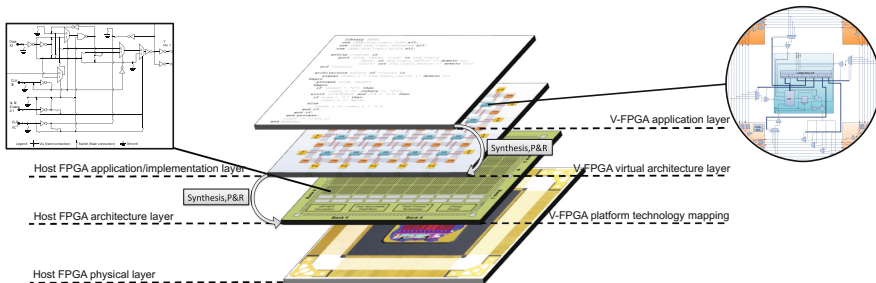


Fig. 1. Layer model of the *V-FPGA* approach

3.3 Routing Infrastructure

Connection boxes around the CLBs consist mainly of multiplexers and their select signals are controlled by configuration registers. At the same time only one routing track can be connected to the input through CB_r , whereas several tracks can be connected to the same output through CB_w . PSMs realize the global routing of the signal paths by connecting tracks from different channels at the intersections. Therefore a 4:1 MUX is located at each output of a PSM as shown in Fig. 3(a). A PSM has on each side W in- and outputs, whereby W is the channel width. On the left and bottom sides, the first position of the MUX is the logic level ‘1’, which is the defined idle value of the routing infrastructure, i.e. if there is no routing intended in this direction. The three remaining positions are each associated with an input from one of the three adjacent sides. The two select lines of the MUX are controlled by configuration registers set by the configuration unit during programming. On the top and right sides of the PSM, the inputs can be fed back to the outputs of the same sides by selecting the first position of the respective multiplexers. This technique, which we call *loopback propagation* enables emulation of bi-directional tracks using uni-directional tracks.

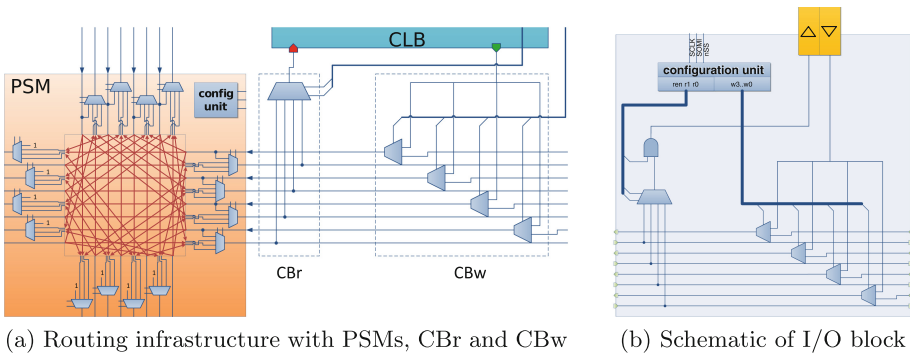


Fig. 3. Interconnects in *V-FPGA*

3.4 I/O Blocks

IOBs on the perimeter of the array have exactly one in- and one output and work in a similar way like the connection boxes of the CLBs. As shown in Fig. 3 (b), a MUX connects one of the tracks from the routing channel to the output pad. When an output is not assigned, logic ‘0’ is issued by an AND gate connected between the MUX and the configuration register bit ren . In favour of higher routability, the input pad can be connected to several tracks in parallel through respective 2:1 MUXs. All the MUXs are controlled by configuration registers.

4 Area and Delay Models

Typically, the overall area requirement and performance of an application mapped onto a virtual FPGA is revealed after the synthesis, place and route (P&R) steps of both, the application and the virtual FPGA. Since the P&R steps are area and timing driven, area and delay models are required in order to find an optimized solution. For instance, our initial *V-FPGA* [9] used the physical parameters of the architecture file templates contained in MEANDER toolflow [1], which are based on 180 nm technology. Similarly, the technology parameters of ZUMA architecture in [2] are almost the same as that of the ones used in the 90 nm architecture templates of the VTR toolflow package [11]. While the application mapping will be still valid and the circuits operable, this practice might be deceitful for the purpose of design space exploration and optimization as the ratios of e.g. logic area to routing area or local routing to global routing will suffer accuracy. This might lead to non-optimal parameter choices and reduced mapping efficiency. To overcome this situation, area and delay models for the *V-FPGA* are derived based on the utilized resource types of the underlying platform. The idea is to decompose the *V-FPGA* into basic elements of minimum size, characterize these elements and in a bottom up approach derive area and delay models that are dependent on the parameters K , N , W , I , and O described in Sect. 3. The programmable units of *V-FPGA* are BLEs, CLBs (including connection boxes), PSMs and IOBs. A BLE is composed of 2^K :1 MUX (for the LUT), 2:1 MUX (for the bypass) and flip-flops (2^K for the configuration unit and one for bypass at the LUT output). The remaining CLB circuitry requires $(N + \lceil I/K \rceil)$:1 MUXs for BLE inputs, optionally N :1 MUXs at the outputs and D-FFs for the configuration unit. Additionally, the connection boxes require 2:1 MUXs for CBw and W :1 MUXs for CBr . Each PSM is composed of 4:1 MUXs for routing and D-FFs for configuration. An IOB needs a W :1 MUX and an AND gate for the output, 2:1 MUXs for the input and D-FFs for the configuration unit. Hence the Minimum Size Basic Elements (MSBEs) are 2:1 MUX, 2-input AND gate and D-FF with their corresponding areas and delays as A_{MUX2} , A_{AND2} , A_{FF} , T_{MUX2} , T_{AND2} , T_{FF_setup} , $T_{FF_clock.to.Q}$ and T_{net} respectively. All the other elements are composed of these MSBEs and are derived as follows:

$$A_{BLE} = (2^K + 1) \cdot A_{FF} + ((2^K - 1) + 1) \cdot A_{MUX2} \quad (1)$$

$$A_{BLE.inMUX} = \left(N + \left\lceil \frac{I}{K} \right\rceil - 1 \right) \cdot A_{MUX2} + \left\lceil \log_2 \left(N + \left\lceil \frac{I}{K} \right\rceil \right) \right\rceil \cdot A_{FF} \quad (2)$$

$$A_{BLE.outMUX} = (N - 1) \cdot A_{MUX2} + \lceil \log_2(N) \rceil \cdot A_{FF} \quad (3)$$

$$A_{CLB} = N \cdot K \cdot A_{BLE.inMUX} + N \cdot A_{BLE} + O \cdot A_{BLE.outMUX} \quad (4)$$

$$A_{CBr} = (W - 1) \cdot A_{MUX2} + \lceil \log_2(W) \rceil \cdot A_{FF} \quad (5)$$

$$A_{CBw} = W \cdot (A_{MUX2} + A_{FF}) \quad (6)$$

$$A_{PSM} = 4 \cdot W \cdot (3 \cdot A_{MUX2} + 2 \cdot A_{FF}) \quad (7)$$

$$A_{IOB} = (2W - 1) \cdot A_{MUX2} + A_{AND2} + (W + \lceil \log_2(W) \rceil + 1) \cdot A_{FF} \quad (8)$$

The delays are obtained through characterizations of the MSBEs in a placed and routed design with the help of the timing analyzing tool. In addition to the MSBEs, the average delay of the short nets is also needed. The relevant delays of the other elements are estimated as follows:

$$T_{MUX4} = 2 \cdot T_{MUX2} + T_{net} \quad (9)$$

$$T_{LUT} = T_{net} + K \cdot (T_{MUX2} + T_{net}) \quad (10)$$

$$T_{BLE_outMUX} = \lceil \log_2(O) \rceil \cdot (T_{MUX2} + T_{net}) \quad (11)$$

$$T_{BLE_inMUX} = \left\lceil \log_2 \left(N + \left\lceil \frac{I}{K} \right\rceil \right) \right\rceil \cdot (T_{MUX2} + T_{net}) \quad (12)$$

$$T_{IOB_in} = T_{MUX2} + T_{net} \quad (13)$$

$$T_{IOB_out} = (\lceil \log_2(W) \rceil - 1) \cdot (T_{MUX2} + T_{net}) + T_{AND2} + T_{net} \quad (14)$$

These models target a fine grained underlying platform (e.g. the 3-input VersaTiles in Actel ProASIC3) and need to be slightly modified when the underlying platform changes. For instance, for an underlying platform with 6-input LUTs, a 4:1 MUX becomes an MSBE as it will have the same area and timing as a 2:1 MUX (both can be realized by 1 LUT). Note that the additive MSBE based models are pessimistic as they don't reflect possible LUT sharing techniques.

5 Methodology of Parametric Design Space Exploration

The architecture level design space exploration is performed with combinations of varying cluster size N and LUT size K . Parts of the VTR toolset [11] are used for this purpose and are complemented by our custom scripts and architecture file generators. However Fig. 4 illustrates a more general view of the CAD flow independent from the actual tools. Starting with the smallest $K = 2$, technology independent netlists of presynthesized benchmark circuits are translated into netlists of K -input LUTs. Proceeding this is the packing process, where N LUTs are clustered into one CLB with an initial value of $N = 1$. The hypergraph nodes of the resulting netlist are placed onto an array of CLBs, whose size is not known at the beginning. One of the optimization goals of this placing step is to determine the required number of CLB columns and rows with minimum area consumption. The placed nodes are then swapped to achieve timing driven optimizations, aiming for minimum distance between connected nodes. The next step is to route the signal paths between the placed nodes by considering the routing capabilities of the architecture (PSMs, connection boxes, in- and output multiplexers). The channel width W is estimated beforehand based on the parameters K and N . This is not an accurate estimation since the minimum W depends also on the application. However, this is good enough to start an initial routing attempt, followed by iterative bisection of the estimated W to converge towards the minimum channel width with a reduced number of routing attempts. Once the minimum channel width is found, the usual timing driven optimizations

follow. The steps of packing, placement and routing require information about the target virtual FPGA architecture and the parameters and constants related to area and delay models, which are provided through architecture files. Some of the area and delay model equations in Sect. 4 are dependent on W , which is known only after the routing process. Thus initially the estimated W is used. For an improved accuracy, a feedback is needed to update the architecture file with the actual channel width W and to re-run the area- and/or timing-driven P&R processes. The results in terms of array size, channel width, area, critical path delay are stored in a data base for assessing the figure of merit (FOM). Then the process is repeated with other combinations of N and K in a nested loop to span the design space of interest.

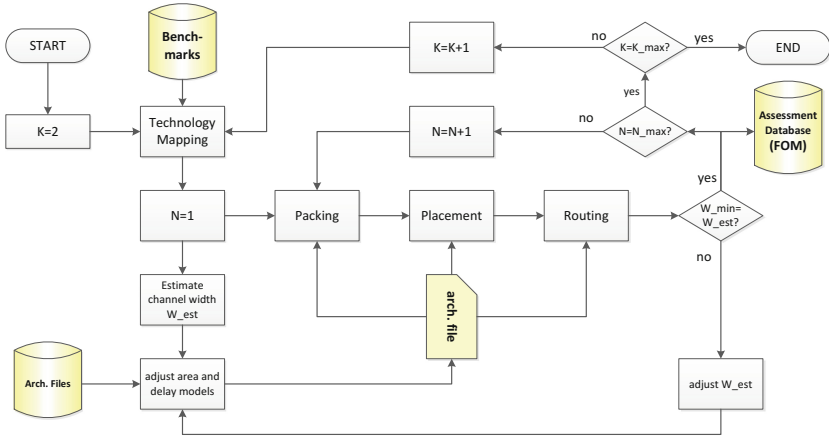


Fig. 4. Concept of parametric DSE Flow.

6 Experimental Results and Analysis

Design space explorations with the 20 largest MCNC benchmarks were performed to observe the parameter sensitivity. The effects of LUT size variation alone were examined in an unclustered architecture, followed by experiments with combinations of LUT size and cluster size. The analysis required 1400 benchmark-runs with the steps *logic optimization*, *LUT mapping*, *packing*, *placing* and *routing*. The first two steps were eased by reusing the pre-mapped versions of the MCNC benchmarks within the VTR package [11] and the latter steps were done with VPR (part of VTR) and with architecture files, that reflect the *V-FPGA* hosted by an Actel ProASIC3 device, including area and delay models from Sect. 4. The result graphs (Figs. 5, 6 and 7) are drawn according to the following scheme: For each benchmark, the variance is displayed with relative distances from the median centred between its best and worst result. This allows to compare the parameter sensitivity among all benchmarks, irrespective of their sizes and scales. The dotted curve represents the average over all benchmarks.

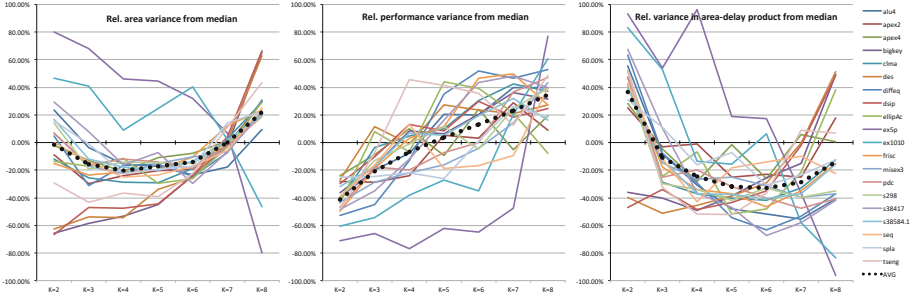


Fig. 5. Effects of LUT size K on (a) area (b) performance (c) area-delay product.

6.1 The Effects of LUT Size Tuning on Area and Performance in Unclustered Architectures

Figure 5 shows the variance of area, performance and area-delay product with different LUT sizes in the range of $K = 2..8$. The average curve (dotted line) of area variance has a smooth bathtub characteristic with a wide optimum for $K = 3..6$ and a degradation in area efficiency outside this range. Hence a LUT size of $K = 3..6$ will yield in average the best area efficiency for general purpose cases. However, the different benchmarks differ greatly in area variance over LUT size K and parameter sensitivity. This manifests not only by the variance range and the steepness but also by the smoothness of the curves. Some of the benchmarks are trending a higher area efficiency with rising K , while some show the opposite trend and others follow a bathtub characteristic. This shows that there is plenty of room for optimization through application specific customization and parameter tuning. Regarding performance variation, all benchmarks show an ascending performance trendline for increasing K with the average curve being almost linear from $K = 2$ to $K = 8$. Some of the benchmarks show relatively smooth curves that are oriented around the average curve, while others show a disturbed and inconsistent curve with alternating change of trend and few show a nearly exponential shape. Around 5% of the benchmarks have a performance maximum at $K = 5$, 10% at $K = 6$, 40% at $K = 7$ and 45% at $K = 8$, which indicates that $K = 7..8$ are in average the recommended choices for performance optimization in general purpose cases. LUT sizes between 5 and 7 show the best trade-off between area and performance with regard to area-delay product. These results differ from [3, 7, 14, 15] and show that their findings based on transistor-level modelling are not certainly transferable to virtual architectures.

6.2 The Effects of Combined Cluster Size N and LUT Size K Tuning on Area and Performance

Clustering can have a significant effect on area and performance depending on how well it is tuned in conjunction with the LUT size. For adequate parameter tuning, the 20 largest MCNC benchmarks are re-evaluated for all combinations

of cluster sizes with $N = 1..10$ and LUT inputs with $K = 2..8$. Figures 6, 7 present the resulting variances in area and performance. Interestingly, quite a few area curves have a sawtooth characteristic with minima at $N = 1$ for all K indicating that clustering is harmful for the respective benchmarks if area efficiency is the objective. For the average case starting with $N = 4$ and $K = 2$, N should decrease with increasing K for better area efficiency. On the whole, the performance increases with rising K and N . The evaluation also shows a strong parameter sensitivity with variances up to $\pm 95.9\%$ in area and $\pm 78.1\%$ in performance. Furthermore, the fluctuating benchmark curves confirm that application specific customization can yield high optimizations, rather than relying on average values for parameterization of the architecture.

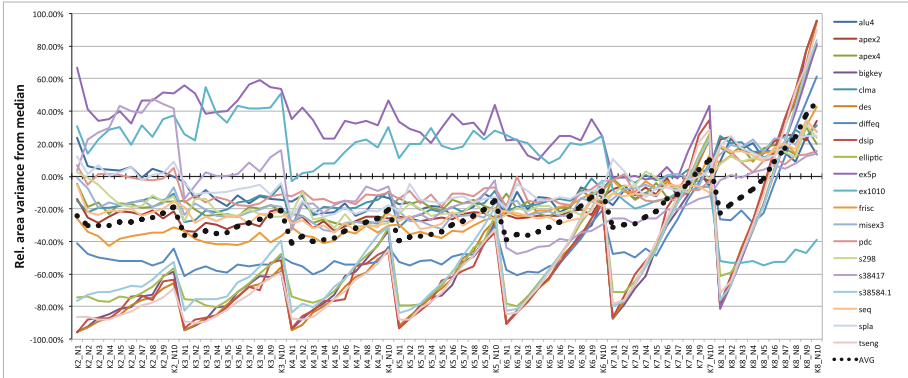


Fig. 6. Effects of LUT size K and cluster size N on area variance.

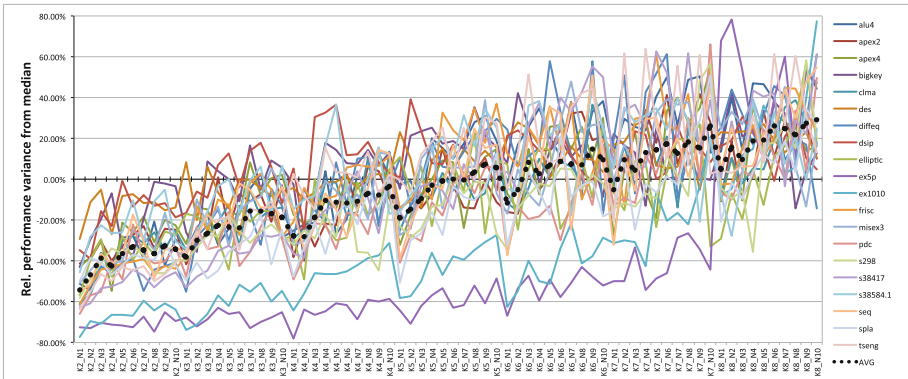


Fig. 7. Effects of LUT size K and cluster size N on performance variance.

7 Conclusions

In this paper an extended version of the *V-FPGA* has been introduced and the area and delay models suitable for virtualization have been derived by decomposing the architecture into MSBEs. In contrast to the existing models which are based on transistor-level, the new models adopt the characterization of MSBEs that are mapped onto the desired underlying COTS FPGA. Thus they represent a more realistic view to the new design space exploration methodology and also to the CAD tools for application mapping. The analysis of over 1400 benchmark-runs with various combinations of LUT size and cluster size reveals a high parameter sensitivity with individual variances up to $\pm 95.9\%$ in area and $\pm 78.1\%$ in performance. This proves a remarkable potential for application specific optimizations through parameter tuning. For general purpose cases, an averaging of area-delay products over the examined benchmarks leads to recommendations of $K = 5.7$ for unclustered logic CLBs and combinations of $K = 4.7$ with $N = 2.5$ for clustered CLBs. However if the target application field is narrow, it is not recommended to rely on averaging as the individual benchmarks differ tremendously from the average values. Furthermore, our results show some discrepancy in the parameter recommendations of physical FPGAs and discourage a 1:1 adoption to virtual FPGAs.

Acknowledgments. This work was partially supported by the German Academic Exchange Service (DAAD).

References

1. MEANDER Design Framework (2016). <http://proteas.microlab.ntua.gr/meander/download/index.htm>. Accessed 25 Nov 2016
2. ZUMA Repository 2016. <https://github.com/adbrant/zuma-fpga/tree/master/source/templates>. Accessed 25 Nov 2016
3. Ahmed, E., Rose, J.: The effect of LUT and cluster size on deep-submicron FPGA performance and density. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **12**(3), 288–298 (2004)
4. Betz, V., Rose, J.: Cluster-based logic blocks for FPGAs: area-efficiency vs. input sharing and size. In: *Custom Integrated Circuits Conference, Proceedings of the IEEE 1997*, pp. 551–554 (1997)
5. Brant, A., Lemieux, G.G.F.: ZUMA: an open FPGA overlay architecture. In: *Field-Programmable Custom Computing Machines (FCCM)*, April 2012
6. Figuli, P., Huebner, M., et al.: A heterogeneous SoC architecture with embedded virtual FPGA cores and runtime core fusion. In: *NASA/ESA 6th Conference on Adaptive Hardware and Systems (AHS 2011)*, June 2011
7. Gao, H., Yang, Y., Ma, X., Dong, G.: Analysis of the effect of LUT size on FPGA area and delay using theoretical derivations. In: *Sixth International Symposium on Quality Electronic Design (ISQED 2005)*, pp. 370–374, March 2005
8. Gupta, P.K.: Accelerating datacenter workloads. In: *26th International Conference on Field Programmable Logic and Applications (FPL)*, August 2016

9. Huebner, M., Figuli, P., Girardey, R., Soudris, D., Siozos, K., Becker, J.: A heterogeneous multicore system on chip with run-time reconfigurable virtual FPGA architecture. In: 18th Reconfigurable Architectures Workshop, May 2011
10. Lagadec, L., Lavenier, D., Fabiani, E., Pottier, B.: Placing, routing, and editing virtual FPGAs. In: Brebner, G., Woods, R. (eds.) FPL 2001. LNCS, vol. 2147, pp. 357–366. Springer, Heidelberg (2001). doi:[10.1007/3-540-44687-7_37](https://doi.org/10.1007/3-540-44687-7_37)
11. Luu, J., Goeders, J., et al.: VTR 7.0: next generation architecture and CAD system for FPGAs. *ACM Trans. Reconfigurable Technol. Syst.* **7**(2), 6:1–6:30 (2014)
12. Lysecky, R., Miller, K., Vahid, F., Vissers, K.: Firm-core virtual FPGA for just-in-time FPGA compilation. In: Proceedings of the 2005 ACM/SIGDA 13th International Symposium on Field-programmable Gate Arrays, p. 271 (2005)
13. Putnam, A.: Large-scale reconfigurable computing in a Microsoft datacenter. In: Proceedings of the 26th IEEE Symposium on High-Performance Chips (2014)
14. Rose, J., Francis, R.J., Lewis, D., Chow, P.: Architecture of field-programmable gate arrays: the effect of logic block functionality on area efficiency. *IEEE J. Solid-State Circ.* **25**(5), 1217–1225 (1990)
15. Tang, X., Wang, L.: The effect of LUT size on nanometer FPGA architecture. In: 2012 IEEE 11th International Conference on Solid-State and Integrated Circuit Technology (ICSICT), pp. 1–4, October 2012