International Journal of Innovative Computing, Information and Control
Volume **16**, Number **1**, February **2020**

ICIC International ©2020 ISSN 1349-4198
pp. **395–403**

ROUTING RESOURCES REDUCTION OF VIRTUAL FPGA FOR SPECIFIC APPLICATION SETS

Theingi Myint[1], Ito Takanori[1], Motoki Amagasaki[1], Qian Zhao[2] and Masahiro Iida[1]

[1]Graduate School of Science and Technology
Kumamoto University
2-39-1 Kurokami, Chuo-ku, Kumamoto-shi, Kumamoto 860-8555, Japan
{ theingi; ito }@arch.cs.kumamoto-u.ac.jp; { amagasaki; iida }@cs.kumamoto-u.ac.jp

[2]School of Information Engineering
Kyushu Institute of Technology
680-4 Kawazu, Iizuka-shi, Fukuoka 820-8502, Japan
cho@ai.kyutech.ac.jp

Received June 2019; revised October 2019

Abstract. *A fine-grained FPGA overlay is a virtual device layer (vFPGA) implemented on a physical commercial FPGA. A vFPGA can provide advantages such as bitstream portability and design that works with third-party CAD tools. However, present vFPGAs impose large resource and performance overheads on the underlying FPGA. In this work, we focus on reducing these resource overheads by implementing an application-specific routing architecture for vFPGAs. Because the routing circuits occupy the majority of resources of typical FPGAs, removing routing channels not used for a given application set can significantly reduce the resources required for a vFPGA. By evaluating the proposed application-specific routing, using our previously proposed scalable-logic-module (SLM) architecture on 5 $\times$ 5 tile array of 7-input logic elements, our approach reduces the number of lookup tables and flip-flop gates required on the physical FPGA by 40% and 39%, respectively.*

Keywords: Reconfigurable computing, Overlay, Virtual FPGA, Routing

1. **Introduction.**

In recent years, field-programmable gate arrays (FPGAs) have been widely adopted in cloud computing and edge computing applications. However, overcoming some limitations of commercial FPGAs is becoming critical for the development and deployment of FPGA-enabled applications. First, a bitstream generated for one FPGA cannot be used on an FPGA of a different type because of hardware resource and architectural differences. Due to this, FPGA developers have to modify their designs and generate bitstreams for all target FPGAs. Second, the architecture details of commercial FPGAs are not open, so third-party CAD tools cannot be integrated with the FPGA design flow. Overlay-based virtual FPGAs (vFPGAs) have been introduced to address these problems. A vFPGA is a reconfigurable device synthesized and implemented on a commercial FPGA. Because the architecture of a vFPGA is opened, we can use third-party CAD tools to design circuit for one and then generate a bitstream that is portable to others.

Kristianti et al. [1] developed data encryption standard algorithm to optimize design by implementing with VHDL on XC3S1200E FPGA devices for security system. Coole and Stitt [2] addressed two problems with FPGAs – long placement and routing time,

DOI: 10.24507/ijicic.16.01.395

and lack of portability – by introducing intermediate fabrics, which are coarse-grained virtual reconfigurable architectures specialized for different application domains. The authors explored the use of intermediate fabric architectures with specialization techniques aimed at minimizing the area and performance overhead of the virtual fabric. Kulkarni et al. [3] proposed a virtual coarse-grained reconfigurable arrays (VCGRAs) architecture which is formed the group of processing elements (PEs) along with the interconnection network at a virtual higher abstraction level. Heyse et al. [4] mentioned that conventional implementations of VCGRAs cause a large overhead. They solved it by using tool flow for parametrized FPGA configurations. To address portability challenges in FPGA design productivity owing to lack of code portability, Kirchgessner et al. [5] introduced a framework for FPGA platform virtualization, called virtual reconfigurable computing (VirtualRC) that is enabling portability across any supported platform. Bollengier et al. [6, 7] designed fine-grained overlay implementing novel features in a cluster of heterogeneous commercial-of-the-shelf (COTS) FPGAs, and demonstrated the use of overlay in an FPGA cluster by performing a hardware application live migration between two nodes of a cluster. Koch et al. [8] presented a fine-grained FPGA-like overlay architecture which can be implemented in the user logic of various FPGA families from different vendors. Lysecky et al. [9] proposed a simple FPGA fabric as a virtual FPGA by using structural VHDL and synthesized the firm-core virtual FPGA onto Xilinx Spartan physical FPGAs, and then mapped 18 benchmark circuits onto the virtual FPGA.

Najem et al. [10] proposed a fine-grained overlay-based vFPGA approach to improve portability, speed up reconfiguration, and promote resources abstraction. By implementing the same vFPGA on different commercial FPGAs in a cluster, the same application bitstream can be used on heterogeneous instances. Using the proposed platform as a base, cluster-scale hardware management capabilities such as node-to-node application migration, scheduling and load balancing, can be implemented. Brant and Lemieux [11] implemented a fine-grained overlay as user logics on top of commercial FPGAs, calling this ZUMA. ZUMA was designed as an open vFPGA architecture that reduces the implementation cost by mapping overlay lookup tables (LUTs) and overlay multiplexers onto a LUT configuration of the physical FPGA fabric. They designed a Clos-style input interconnect block (IIB) network to improve the efficiency of the area for the internal crossbar of the cluster, a resource efficient configuration controller, and a modeling architecture to determine the most efficient parameters for mapping to a given architecture. Configurable LUT random-access memories (LUTRAMs) are used for implementing both programmable LUTs and routing multiplexers. The ZUMA overlay architecture as compiled on Xilinx and Altera FPGAs required two thirds less LUTs when LUTRAMs were used. However, the ZUMA architecture is heavily reliant on the physical FPGA structure (e.g., LUT size), which limits bitstream portability. Wiersema et al. [12, 13] extended an embedding of a ZUMA-based virtual FPGA fabric into a complete configurable system-on-chip. The authors presented an open tool flow to synthesize configurations for the virtual FPGA with the extension of ZUMA and its embedding into the ReconOS/Linux system running on a Xilinx Zynq in order to analyze the area and delay overheads.

Between these coarse- and fine-grained vFPGAs, coarse-grained overlays have small overheads but limited flexibility in logic expression while fine-grained overlays can implement any application circuit but have higher resource and performance overhead because their large number of configuration memories have to be implemented with limited flip-flop gates (FFs) on the physical FPGA. In this work, we focus on reducing the resource overhead by implementing a vFPGA on an application-specific routing architecture for the overlay. Because the routing circuits occupy the majority of resources in typical FPGAs, removing routing channels not needed for a given application set can significantly reduce the resource requirements of a vFPGA. By implementing a vFPGA with the proposed application-specific routing and our previously proposed scalable-logic module (SLM), which requires less configuration memory, we can achieve a vFPGA architecture with a better balance of overhead and flexibility.

The remainder of this paper is organized as follows. Section 2 describes fine-grained overlay vFPGAs, including the strategy of virtual synthesis. The evaluation condition of a vFPGA based on the SLM architecture and the results of evaluating the architecture with a CAD tool are described in Section 3. The conclusion of this paper is given in Section 4.

2. **Fine-Grained vFPGA.**

2.1. **Overview.** A fine-grained vFPGA architecture is cognate with a traditional FPGA architecture because the reconfigurable elements are made up of logic blocks as fine-grained reconfigurable elements. Logic Blocks, which comprise N clustered basic logic elements (BLEs) include a full crossbar that connects both the logic block inputs (I) and feedback from BLE outputs (N). A BLE includes a k-inputs LUT and one FF. A LUT is used as fine-grained logic cell. It corresponds to a set of reconfigurable elements available to the application. When we explore vFPGA architecture, we must consider nine important objectives: multi-tenancy, resource management, flexibility, isolation, scalability, performance, security, resilience, and programmer productivity [14]. These are the differences in the point of design for between conventional discrete FPGAs and vFPGAs.

2.2. **Strategy.** Figure 1 shows vFPGA implementation flow. A target application for vFPGA is described by hardware design language (HDL) such as Verilog, SystemC for high-level synthesis (HLS) design, or a domain-specific language (DSL). This design is synthesized for the vFPGA (i.e., virtual synthesis), and the generated bitstream does not depend on the specific FPGA. However, a traditional LUT-based vFPGA has the following key features: a huge amount of memory cell bits are used for configuration. An n-input

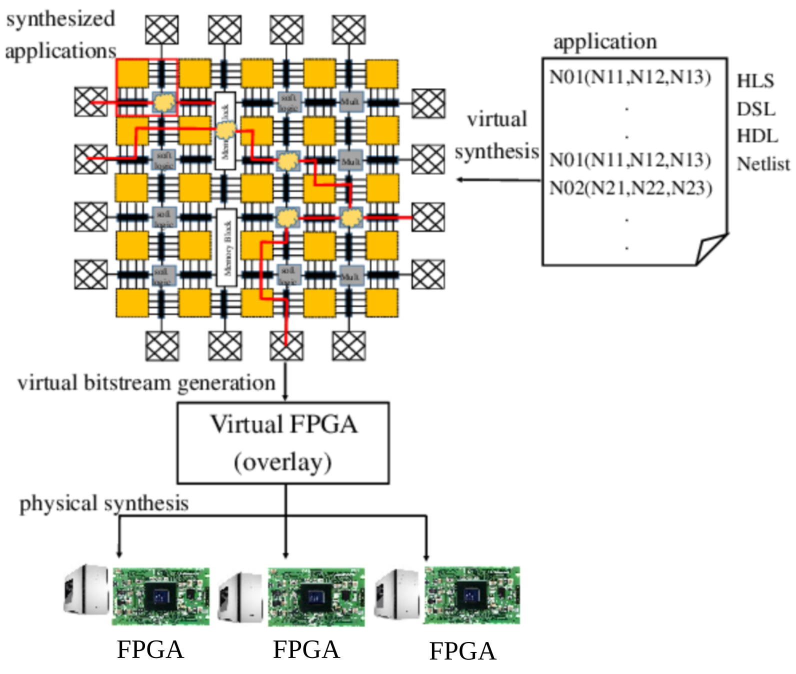

FIGURE 1. Example of virtual synthesis flow

LUT requires 2^n bits of configuration memory. Configuration memory cells are assigned to FFs in vFPGA design. As the number of bits needed increases, so does the amount of hardware resources required. In this paper, we proposed a vFPGA overlay architecture based on SLM logic cells to mitigate the amount of configuration memory needed. A primary feature of SLMs is that they require less configuration memory than traditional LUTs do. The resulting vFPGA overlay can be implemented on various FPGAs, as shown in Figure 1.

Although vFPGAs offer portable design, resource abstraction, and faster configuration times, traditional vFPGAs are also high cost because of the necessary integration scale, speed, and power. As the scale integration of such vFPGAs increases, the routing part becomes both more complex and larger. Here, it is assumed that vFPGA is used as hardware macro block or hardware library with a limited application set. This means that we can eliminate many unused resources, depending on application-specific routing. In our approach, the amount of hardware resources of the logic part required are reduced by using SLM as logic cells. In addition, the area devoted to wiring portion is reduced by removing unused wiring. Then, we detect the unused wiring in the vFPGA with our CAD tool and generate modified HDL files that account for the wiring information so as to efficiently reduce the used area.

2.3. **LUT-based vs SLM-based logic cells.** Most FPGAs use LUTs as their logic cells. The area requirement of LUTs is increased with the input size k, which means the area of FPGAs increases when input size does. As an example, we consider a LUT with an input size of 7. In LUT architecture, the input of LUT (k) is 7, the cluster size (N) is 4. The flexibility of the switch block ($F_s = 3$) indicates that an output multipliexer in the switch block can select an input from three directions for its output. The flexibility of inputs and outputs of the connection block ($F_c = 0.5$) means that 50% of tracks in the routing channel are connected to the logic cluster. Figure 2 shows the logic block structure with 7-input LUTs. The general input (I) of the logic block is 17, found by the calculation $I = k(N + 1)/2$, rounded down. One BLE consists of a LUT, an FF, and a selector [15]. The selector's function determines whether the value of the LUT is output or the value of FF is the output. The resource and area overheads needed for a k-input LUT-based fine-grained overlay includes 2^k configuration memory bits, which are implemented with limited FFs on the physical FPGA.

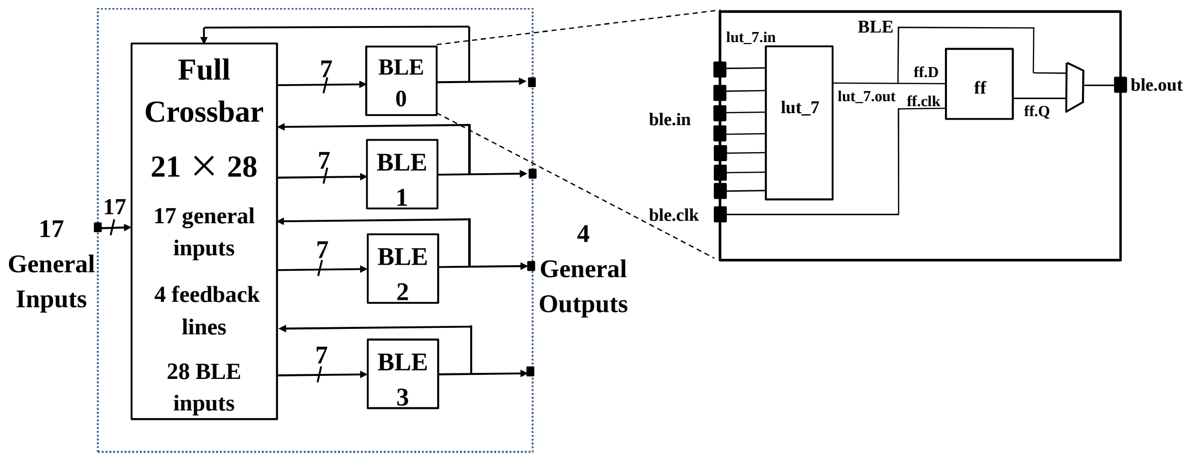

Figure 2. Logic block structure with 7-input LUT

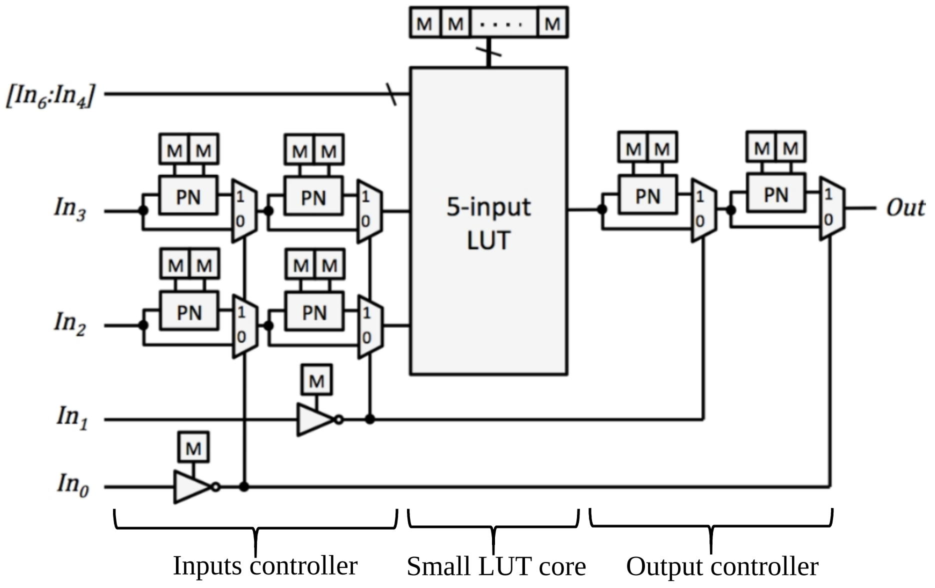

FIGURE 3. The 7-SLM (5,2) structure

An SLM includes an input controller, a small LUT core and an output controller as shown in Figure 3, where PN is the programmable NAND and M is the configuration memory [16]. The minimum number of configuration memory bits is 2^{k-1} for k-input SLM in this architecture. The programmable NAND (PN) circuit can negate inputs or fix the output to 0 or 1 with two configuration memory cells. This structure has k-inputs, a $(k-1)$-LUT core, and one output that can be defined as k-SLM $(k-1,\omega)$, where $0 \leq \omega \leq k-1$. For multilevel SLMs, the sizes for a k-SLM $(k-m,\omega)$ are the following: a k-input SLM with a $(k-m)$-LUT core, and ω controllable inputs with m-level controllers, where $0 \leq \omega \leq k-m$ and $1 \leq m \leq k-2$. If $k = 7$, then $m = 2 - level$, the number of controllable inputs. Here, $\omega = 2$, for a k-SLM $(k-m,\omega) =$ 7-SLM (5,2), that is, a 7-input SLM with 5-input LUT core, as shown in Figure 3.

2.4. **Application-specific routing.** Our vFPGA is similar to an FPGA homogeneous tile based on a mesh structure, in which each tile includes one logic block, one switch block, and two connection blocks. The logic block structure is used with the SLM structure; the switch blocks located at the intersection of horizontal and vertical wiring are Wilton type [17] ($F_s = 3$), and the flexibility of inputs and outputs of the connection blocks connected to the wiring and logic blocks are half path wire ($F_c = 0.5$). In the routing part, we eliminate the unused wiring of the input and output part between the logic tiles to reduce the channel widths, which reduces hardware resources required. Figure 4 shows a developed CAD flow. Figure 4(a) shows an architecture exploration flow that matches that of a traditional FPGA. Then Figure 4(b) shows an evaluation flow. We modified EasyRouter to detect unused wiring by combining P&R information from Figure 4(a). Based on unused wiring, the final HDLs of each tile are generated.

3. Evaluation.

3.1. **Evaluation condition and environment.** For this evaluation, 7-LUT and 7-SLM (5,2) architectures are used. The main objective is to use fine-grained overlay vFPGA architecture in a hardware library implementation with our CAD tool in order to reduce the amount of hardware resources required by removing unused wiring. We use ABC [18]

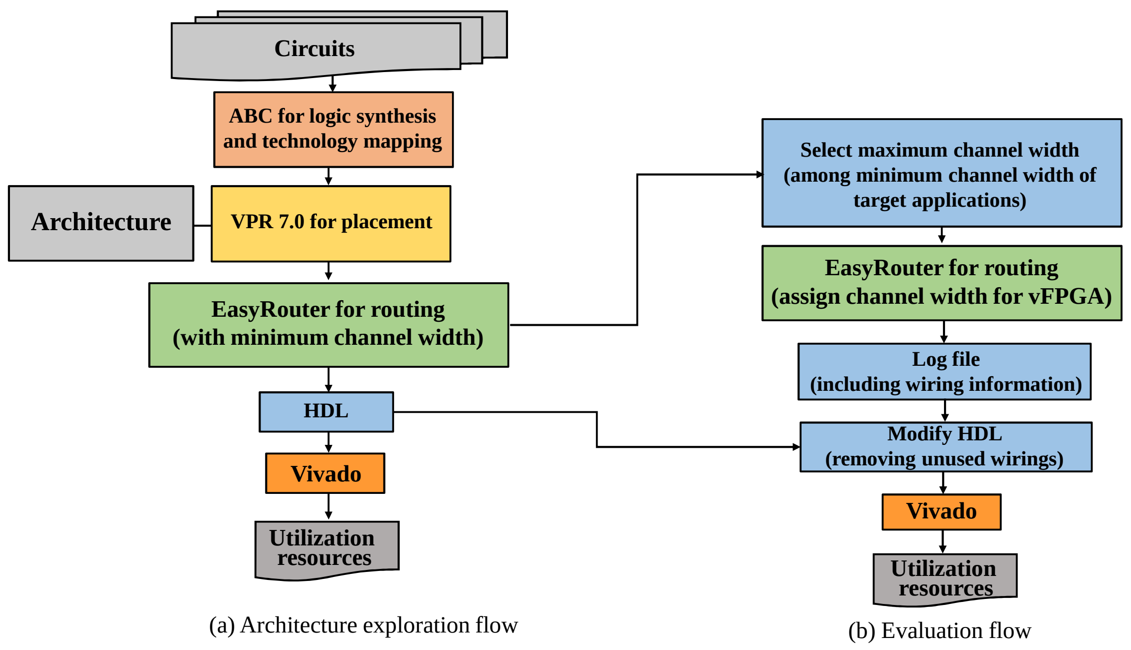
Circuits
ABC for logic synthesis and technology mapping
Architecture
VPR 7.0 for placement
EasyRouter for routing (with minimum channel width)
HDL
Vivado
Utilization resources
(a) Architecture exploration flow
Select maximum channel width (among minimum channel width of target applications)
EasyRouter for routing (assign channel width for vFPGA)
Log file (including wiring information)
Modify HDL (removing unused wirings)
Vivado
Utilization resources
(b) Evaluation flow

FIGURE 4. CAD tool flow

TABLE 1. Comparison of array size and channel width

Circuits	Array size (7-LUT)	Array size (7-SLM (5,2))	CW (7-LUT)	CW (7-SLM (5,2))
9sym	(4×4)	(5×5)	16	12
example2	(5×5)	(5×5)	16	16
i4	(5×5)	(5×5)	22	22
mult32a	(5×5)	(5×5)	12	14
rd84	(5×5)	(5×5)	16	18
s832	(5×5)	(5×5)	18	16
x1	(5×5)	(5×5)	22	22
C1355	(5×5)	(5×5)	24	20
C499	(5×5)	(5×5)	20	20

for technology mapping, VPR 7.0 [19, 20] is used in placement, and EasyRouter [21] in the routing part, as shown in Figure 4(a). By running the routing part with EasyRouter, we find the minimum viable channel width (setting fixed channel width), array size, and HDL files. The target devices are Xilinx Artix-7 xc7a200tffg1156-1, Kintex-7 xc7k420tffv1156-1, and Kintex UltraScale+ KCU116 xcku5p-ffvb676-2-e, using Xilinx Vivado 2018.2. After placement and routing, we have the hardware usage of both architectures for each logic tile for the target applications set (nine circuits). Table 1 shows the comparison of channel width (CW) before evaluation and the array size for both LUT-based and SLM-based logic tiles. In most cases, the array size of the SLM architecture is similar to that of the LUT architecture. We can directly calculate the hardware resources required for one logic tile with our CAD tool, as shown in Figure 4(a). Before removing the unused wiring, the comparison results of hardware resources in one logic tile with Artix-7, Kintex-7, and Kintex UltraScale+ are found; these are shown in Table 2.

TABLE 2. Hardware resources in one logic tile before removing unused wiring

Structure	LUTs	FFs
7-LUT with Artix-7	1,378	927
7-SLM (5,2) with Artix-7	832	570
7-LUT with Kintex-7	1,374	927
7-SLM (5,2) with Kintex-7	836	570
7-LUT with Kintex UltraScale+	1,346	927
7-SLM (5,2) with Kintex UltraScale+	819	570

3.2. **Evaluation result.** Our aim is to reduce the routing area for multiple tiles (5×5) with an SLM architecture for a fine-grained overlay vFPGA. After routing with EasyRouter, we know the minimum channel width for each circuit. The least amount of routing resources that run parallel and allow the successful routing of each circuit is called the minimum channel width. We choose the largest minimum channel width of all circuits as the maximum channel width to fix the minimum necessary FPGA scale to be able to implement all desired circuits. After calculation in this way, we use 24 (CW) for 7-LUT and 22 (CW) for 7-SLM (5,2), as shown in Table 1.

For the target vFPGA, the channel width can be assigned as the maximum channel width multiplied by 1.2 to provide sufficient resources for target applications. We calculate (24×1.2) = 30 (CW) for 7-LUT and (22×1.2) = 28 (CW) for 7-SLM (5,2). Applying the corresponding channel width, we use EasyRouter to generate a log file that includes the wiring information. By calculating how many wirings are used or unused within the (5×5) logic tiles for a fine-grained overlay vFPGA, we modify the HDL files by using these wiring information results, and evaluate the hardware resources used in the vFPGA by removing unused wiring, as shown in Figure 4(b). As a result, we can remove 7.5% (by routing area) of wiring as unused wiring in 7-LUT and 4.6% in 7-SLM (5,2).

Table 3 shows the hardware resources of 7-LUT and 7-SLM (5,2) configurations in (5×5) logic tiles by using an Artrix-7 device after removing the unused wiring. Table 4, shows a wiring reduction by 7.5% from removing unused wiring in the 7-LUT case, by 20.31% with LUTs and by 2.46% with FFs on Artix-7; for Kintex-7, these numbers are 20.13% for LUTs and 2.46% for FFs. For Kintex UltraScale+, these are 19.12% for LUTs and 2.46% for FFs. Eliminating 4.6% of the wiring as unused wiring with 7-SLMs (5,2), we can also reduce the LUTs by 40.22% and the FFs by 39.38% on Artix-7. For Kintex-7 (resp., Kintex UltraScale+), these numbers are 40.16% (resp., 40.32%) for LUTs, and 39.51% (resp., 39.38%) for FFs.

3.3. **Discussion.** A reduction of about 20% was achieved for LUTs and 3% for FFs reduction on three devices after removing an unused 7.5% of the writing for 7-LUT systems. In 7-SLM (5,2) systems, the reductions were about 40% for LUTs and 39% for FFs on three devices after removing an unused 4.6% of the wiring. For fine-grained vFPGAs of both architectures with the same (5×5) logic tiles for all target applications, we can reduce the area of routing resources on three devices by removing the unused wiring.

4. **Conclusions.** Overlay architectures, which are virtual configurable architectures that run on top of the physical architecture of FPGAs, allow design portability across FPGAs from many vendors. However, there can be area overhead problems because of the different hardware resources available. To fix this, we intend to reduce the hardware resource overhead by the implementation of application-specific routing architectures in vFPGAs. In this paper, we only focused on the reduction of the routing area of the vFPGA for Xilinx

Table 3. Hardware resources in (5×5) tiles after removing unused wiring in Artix-7 as an example

7-LUT	LUTs	FFs	7-SLM (5,2)	LUTs	FFs
TILE0101	946	900	TILE0101	772	552
TILE0102	920	889	TILE0102	798	558
TILE0103	966	910	TILE0103	816	564
TILE0104	959	913	TILE0104	847	561
TILE0105	919	880	TILE0105	839	561
TILE0201	1,290	882	TILE0201	756	546
TILE0202	936	895	TILE0202	840	564
TILE0203	959	912	TILE0203	844	567
TILE0204	956	913	TILE0204	846	561
TILE0205	918	886	TILE0205	835	564
TILE0301	1,372	912	TILE0301	766	555
TILE0302	1,357	906	TILE0302	833	567
TILE0303	969	916	TILE0303	837	567
TILE0304	1,357	916	TILE0304	865	567
TILE0305	1,375	896	TILE0305	820	570
TILE0401	1,353	921	TILE0401	807	555
TILE0402	972	907	TILE0402	831	555
TILE0403	1,372	913	TILE0403	850	567
TILE0404	946	909	TILE0404	819	567
TILE0405	949	895	TILE0405	837	564
TILE0501	1,362	909	TILE0501	799	552
TILE0502	971	906	TILE0502	839	567
TILE0503	966	912	TILE0503	835	564
TILE0504	1,381	905	TILE0504	821	567
TILE0505	982	901	TILE0505	843	567

Table 4. Reduction in hardware resources by removing unused wiring

Structure	Unused wiring reduction	LUTs reduction	FFs reduction
7-LUT with Artix-7	7.5%	20.31%	2.46%
7-SLM (5,2) with Artix-7	4.6%	40.22%	39.38%
7-LUT with Kintex-7	7.5%	20.13%	2.46%
7-SLM (5,2) with Kintex-7	4.6%	40.16%	39.51%
7-LUT with Kintex UltraScale+	7.5%	19.12%	2.46%
7-SLM (5,2) with Kintex UltraScale+	4.6%	40.32%	39.38%

FPGA devices. We can find how much the unused wiring can be removed, which reduces the required resources. In future works, we intend to generate bitstreams for vFPGA. And also, we will try to implement vFPGA on FPGA devices of different vendors.

REFERENCES

[1] V. E. Kristianti, E. P. Wibowo, A. Pertiwi, B. Soerowirdjo and H. Afandi, Implementation optimization of the DES algorithm on FPGA to support smartcard processors, *ICIC Express Letters, Part B: Applications*, vol.10, no.7, pp.565-570, 2019.

[2] J. Coole and G. Stitt, Intermediate fabrics: Virtual architectures for circuit portability and fast placement and routing, *Proc. of the 8th International Conference on Hardware/Sofware Codesign and System Synthesis*, *CODES/ISSS'10*, Scottsdale, AZ, USA, pp.13-22, 2010.

[3] A. Kulkarni, E. Vansteenkiste, D. Stroobandt, A. Brokalakis and A. Nikitakis, A fully parameterized virtual coarse grained reconfigurable array for high performance computing applications, *IEEE International Parallel and Distributed Processing Symposium Workshops*, 2016.

[4] K. Heyse, T. Davidson, E. Vansteenkiste, K. Bruneel and D. Stroobandt, Efficient implementation of virtual coarse grained reconfigurable arrays on FPGAs, *Proc. of the 23rd International Conference on FPL*, Porto, Portugal, 2013.

[5] R. Kirchgessner, G. Stitt, A. George and H. Lam, VirtualRC: A virtual FPGA platform for applications and tools portability, *Proc. of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, New York, NY, USA, pp.205-208, 2012.

[6] T. Bollengier, M. Najem, J. C. L. Lann and L. Lagadec, Demo: Overlay architectures for heterogeneous FPGA cluster management, *2016 Conference on DASIP*, pp.239-240, 2016.

[7] T. Bollengier, M. Najem, J. C. L. Lann and L. Lagadec, Overlay architecture for FPGA resource virtualization, *GDR SOC SIP*, Nantes, France, 2016.

[8] D. Koch, C. Beckhoff and G. G. F. Lemieux, An efficient FPGA overlay for portable custom instruction set extensions, *Proc. of the 23rd International Conference on FPL*, Porto, Portugal, 2013.

[9] R. Lysecky, K. Miller, F. Vahid and K. Vissers, Firm-core virtual FPGA for just-in-time FPGA Compilation, *Proc. of the 13th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, 2005.

[10] M. Najem, T. Bollengier, J. Christophe Le Lann and L. Lagadec, Extended overlay architectures for heterogeneous FPGA cluster management, *Journal of Systems Architecture*, vol.78, pp.1-14, 2017.

[11] A. Brant and G. G. F. Lemieux, ZUMA: An open FPGA overlay architecture, *IEEE 20th International Symposium on FCMM*, INSPEC Accession Number: 12866207, 2012.

[12] T. Wiersema, A. Bockhorn and M. Platzner, An architecture and design tool flow for embedding a virtual FPGA into a reconfigurable system-on-chip, *Journal of Computers and Electrical Engineering*, vol.55, pp.112-122, 2016.

[13] T. Wiersema, A. Bockhorn and M. Platzne, Embedding FPGA overlays into configurable system-on-chip: ReconOS meets ZUMA, *Proc. of the International Conference on ReConFig*, pp.1-6, 2014.

[14] A. Vaishnav, K. Dang Pham and D. Koch, A survey on FPGA virtualization, *Proc. of the 28th International Conference on FPL*, Conference Number 28, 2018.

[15] M. Amagasaki and Y. Shibata, FPGA structure, in *Principles, and Structures of FPGAs*, H. Amano et al. (eds.), Springer, 2018.

[16] M. Amagasaki, R. Araki, M. Iida and T. Sueyoshi, SLM: A scalable logic module architecture with less configuration memory, *IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences*, vol.E99-A, no.12, pp.2500-2506, 2016.

[17] I. Kuon, R. Tessier and J. Rose, FPGA architecture: Survey and challenges, *Foundations and Trends in Electronic Design Automation*, vol.2, no.2, pp.135-253, 2008.

[18] Berkeley Logic Synthesis and Verification Group, *ABC: A System for Sequential Synthesis and Verification*, 2012.

[19] 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, *Proc. of the 20th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp.77-86, 2012.

[20] J. Anderson, J. Rose and V. Betz, VPR 7.0: Next generation architecture and CAD system for FPGAs, *ACM TRETS*, vol.7, no.2, article 6, 2014.

[21] Q. Zhao, K. Inoue, M. Amagasaki, M. Iida, M. Kuga and T. Sueyoshi, FPGA design framework combined with commercial VLSI CAD, *IEICE Transactions on Information and Systems*, vol.E96-D, no.8, pp.1602-1612, 2013.