Focus on What is Needed: Area and Power Efficient FPGAs Using Turn-Restricted Switch Boxes

Fatemeh Serajeh-hassani*, Mohammad Sadrosadati†, Sebastian Pointner‡, Robert Wille§ and Hamid Sarbazi-azad¶
*†¶Sharif University of Technology, ‡§Johannes Kepler University, ¶IPM
Email: *†{serajeh,sadrosadati}@ce.sharif.edu, ‡{sebastian.pointner,robert.wille}@jku.at, ¶azad@ipm.ir

***Abstract*—Field-Programmable Gate Arrays (FPGAs) employ a significant amount of SRAM cells in order to provide a flexible routing architecture. While this flexibility allows for a rather easy realization of arbitrary functionality, the respectively required cells significantly increase the area and power consumption of the FPGA. At the same time, it can be observed that full routing flexibility is frequently not needed in order to efficiently realize the desired functionality. In this work, we are proposing an FPGA realization which focuses on what is needed and realizes only a subset of the possible routing options using what we call *Turn-Restricted Switch-Boxes*. While this may yield a slight decrease in the run-time performance of the realized functionality, it allows for substantial improvements with respect to area and power consumption. In fact, experimental evaluations confirm that area and power can be reduced by more than 40% and 60%, respectively, in the best cases. The performance overhead is negligible (up to 3%), on average.**

I. Introduction

FPGAs [11] are programmable pre-fabricated silicon devices that can implement almost any kind of circuit functionality. They have become a popular design platform over the last decade, because they (1) allow for an early prototyping and fast time-to-market due to their much faster realization process, (2) offer low non-recurring-engineering costs due to their flexibility and re-programmability, (3) are much cheaper than ASICs when the number of required devices is not that large, and (4) allow for frequent and remote (re-)programming that makes FPGAs very suitable for applications that involve on-line modifications [1], [4], [9], [25].

An FPGA consists of a two-dimensional array of building blocks to provide the required logic and routing capabilities. Horizontal and vertical routing channels connect these building blocks for implementing larger functions [2]. To this end, FPGAs employ SRAM cells to configure the routing blocks and to eventually realize the desired functionality. More precisely, FPGAs realize so-called *switch-boxes* [6] which allow a routing to take *turns* [5]. In order to ensure full routing flexibility (i.e., turns in all possible directions) and, by this, a rather easy realization of the desired functionality, common FPGA structures employ a significant amount of SRAM cells. However, these routing resources significantly increase the area and the power consumption of the FPGA. In fact, because of them, modern FPGAs consume about 60%-80% of the transistors, just to realize the full routing flexibility [12], [19]. We additionally observe that *switch-boxes* contribute to the total FPGA area and power consumption by 33% and 45%, on average, respectively.

At the same time, switch-boxes are often not completely utilized. In fact, Figure 1 shows the utilization rate of the resources in the switch-box (such as SRAM cells and multiplexers) for different standard circuits from the MCNC benchmark suite [24]. This shows that, on average, 50% of the resources in the switch box are not used, i.e. many SRAM cells and multiplexers in FPGAs which contribute to the area and power consumption are actually not required. Hence, the full routing flexibility which is realized by such significant amount of resources is often not needed, i.e., in many cases the desired functionality can be realized using only a fraction of the turns available through the switch-boxes.

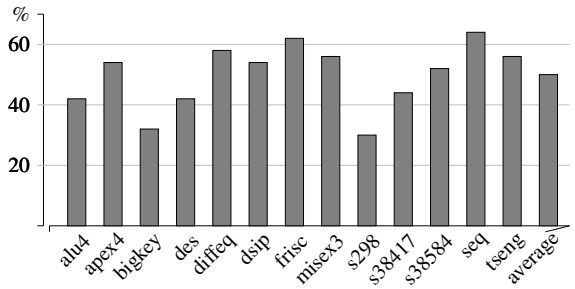

Fig. 1: Utilization of the switch-boxes

In this work, we exploit this observation. More precisely, we focus on what is needed and introduce a revised switch-box structure called *Turn-Restricted Switch-Box* (TRSB), which does *not* realize *all* possible turns anymore, but only a subset of them. While this restricts the flexibility and makes routing failures more likely, it significantly improves the utilization rate and allows for a much more efficient use of resources. The restrictions can thereby be applied to improve the utilization rate of the switch-box *and* to satisfy the required routing flexibility at the same time. Overall, TRSBs may make it slightly harder to realize the desired functionality due to decreased routing flexibility, but the corresponding realization of TRSBs require a significantly smaller amount of SRAM cells. Consequently, the area and power consumption are significantly reduced.

Eventually, experimental evaluations show that the proposed FPGA realization indeed may lead to a slightly slower realization of the desired functionality. But in contrast, substantial improvements with respect to area and power consumption are gained. In the best cases, the area and power consumption can be reduced by more than 40% and 60%, respectively. In contrast, the increase in critical path delay remains tolerable (up to 3%).

The remainder of this work is structured as follows. The next section briefly reviews the architecture of FPGAs as well as the considered cost metrics used in this paper. Afterwards, Section III provides a summary of the conducted investigations which eventually motivated the proposed approach and led to the general idea of this work. The realization of this idea is then described in Section IV. Finally, Section V summarizes and discusses the obtained results before we discuss related work in Section VI and conclude the paper in Section VII.

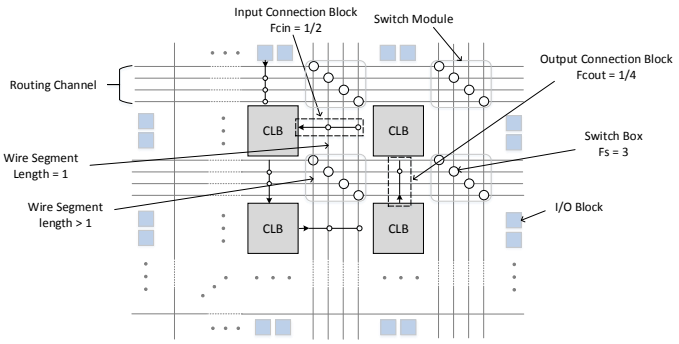

Fig. 2: Architecture of an island-style FPGA.

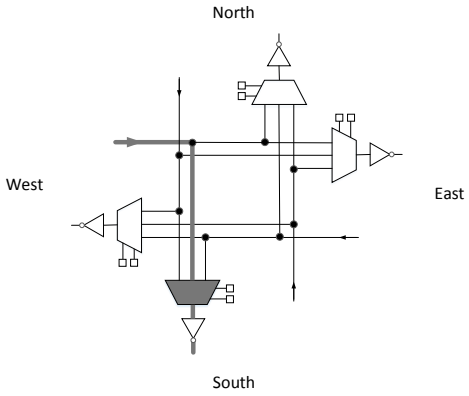

Fig. 3: Unidirectional mux-based switch-box.

II. Background

In order to keep this work self-contained, this section briefly reviews the concept of configurable logic in general before the considered island-style routing architecture for FPGAs is considered in detail. Afterwards, a brief discussion on the considered cost metrics for FPGAs (namely area, power, and critical path delay) is given.

A. Configurable Logic Block

Modern FPGAs can realize almost all logic designs. Therefore, FPGAs are built up using different kinds of basic blocks as illustrated in Figure 2. The most generic basic block is called the *Configurable Logic Block* (CLB) out of which the desired functionality can be built. Every CLB involved in the realization is to realize a part of the target circuit (i.e., a particular sub-function). CLBs can be built in various fashions (e.g., processor based, LUT-based, etc.). Modern CLBs, as used by Altera/Intel or Xilinx, are based on *Lookup Tables* (LUTs) [8], [21]–[23]. Here, a CLB is composed of an arbitrary number of so-called *Basic Logic Elements* (BLEs), where each BLE consists of an LUT and a storage element such as a flip-flop. A BLE, based on a k-input LUT, is capable to realize any Boolean function of k input bits. The combination of those LUTs eventually realizes the desired target functionality. However, to connect those building blocks with their sub-functionalities, a proper routing needs to be determined as described next.

B. Island-Style Routing

Modern FPGAs are most commonly built up using the so-called *island-style* architecture [5], [6], [18], [25]. To this end, the FPGA is built up using a two-dimensional grid. CLBs are placed on this grid together with other basic blocks for establishing connections between the single CLBs as well as with the outside-world forming a routing network. To this end, the following further building blocks are utilized:

Routing Channel: As can be seen in Figure 2, routing channels are basically the wire segments between the other basic-blocks. The number of logic blocks that a wire segment spans identifies its length. FPGA wire segment lengths can be either the same or a combination of different sizes. In terms of the island-style routing architecture, routing is based on horizontal and vertical routing channels.

Connection block: In order to create a connection between the CLBs/IO-pins and the routing network, connection boxes are used. Connection-boxes can be classified in terms of their flexibility. As can be seen in Figure 2, the flexibility F_{cin} of the input connection block highlighted in the top-left of the figure is $F_{cin} = \frac{1}{2}$ since two wire segments in the routing channel are connected to the logic block input pin. The output connection block flexibility F_{cout} of the output connection block highlighted in the bottom-right of the figure is $F_{cout} = \frac{1}{4}$, that is the fraction of wire segments in a routing channel which the output logic block pin is connected to.

Switch Box: Compared to connection-blocks, switch-boxes are used to configure the routing-network itself, i.e. to establish a connection between the horizontal and vertical wires of the routing network. To this end, a switch-box is capable to establish routing from any direction to any other direction.

Modern SRAM-based FPGAs employ a unidirectional routing architecture with MUX-based switch-boxes [7], [14], [17], [22], [23]. Figure 3 shows a realization of such a switch-box where the routing from West to South is highlighted. This routing is realized by setting the select signals of the respective multiplexers as indicated in Figure 3. Because of these select signals, it is only possible to realize one routing scenario at once. Switch-box flexibility, which is known as F_s, is defined to be the number of possible connections a wire segment can make. The classification in terms of flexibility as introduced for connection-boxes can also be applied to switch-boxes, e.g., the switch-boxes as shown on the right-hand side of Figure 2 as well as the switch-box shown in Figure 3 command over a flexibility of $F_s = 3$.

Switch Module: Since the routing network of an FPGA usually commands over a certain bit-width, switch-boxes designed for a single bit are grouped into so called switch-module as illustrated in Figure 2 for a switch-module with a bit-width of 4 bit. The switch module topology describes how each wire segment on one side of the switch module is connected to the wire segments on the other 3 sides. We employed disjoint switch-module which has been used in industrial FPGAs [3]. In a disjoint switch module, a wire segment can only connect to other wire segments with the same numerical designation.

C. Considered Optimization Metrics

In this work, we aim for the optimization of FPGAs with respect to area and power-consumption (at a possible expense of performance). To this end, we consider the following cost definitions (which are common in the domain; see e.g., [5], [6], [18], [25]):

Critical Path Delay (Performance): Almost all digital circuits are based on a synchronous-sequential design. To this end, a clock reference signal is used as timing reference. In order to estimate the maximum clock speed which can be applied to the circuit, the critical path of the design is getting utilized (i.e., the longest possible trace through the design). In this work, we aim

for modifications at the architectural level. Therefore, we also have to analyze if these modifications have a positive or negative impact on the critical path (e.g., can we keep the same clock rate or do we have to adapt the clock rate after the architectural modifications).

Area Consumption: Modern FPGAs are based on a flexible routing network. However, this routing network consumes more than 60% of the entire used transistors of the FPGA. For analyzing the optimization characteristics, not only the critical path is getting analyzed, also the area consumption. By utilizing both characteristics, the impact of area optimization towards execution performance can be analyzed.

Power Consumption: Altering the area consumption (i.e., the number of utilized transistors) of the FPGA has a direct impact to the power consumption. The power consumption of FPGAs depends on their actual configuration (e.g., which traces are routed). We can divide the power-consumption of FPGAs into dynamic and static power consumption. Since the routing-network is configured normally only once, when the FPGA loads the configuration e.g., from a flash memory, the dynamic power consumption of the routing network for the configuration of CMOS-based SRAM cells is negligible. However, static power consumption is getting more and more important, since even when the transistor is inactive, it wastes energy due to e.g., leak-current.

III. Motivation and General Idea

In this work, we aim for the reduction of area and power consumption of generic FPGA architectures. Here, the discussed switch-boxes are one of the main reasons for the resulting area and power costs since the corresponding switch-boxes needed to establish a new routing includes SRAM cells as a main contributor for the cost. Hence, an obvious way to reduce the cost is to simplify the switch-box's reconfiguration circuit.

In this work, we propose such a simplification. The general idea of this rests thereby on the following two observations.

Observation 1. *Consider again Figure 3 which shows the realization of a switch-box. The figure shows the components (i.e., the multiplexers and SRAM cells) which contribute to the highlighted West-South turn (e.g., note that each multiplexer needs 2 SRAM cells in order to select among its 3 inputs). As can clearly be seen, only a fraction of the entire switch-box is actually needed to realize this turn (more precisely, 3 of the 4 multiplexers as well as inverters are not needed in this particular scenario).*

Hence, rather than realizing a switch-box supporting *all* turns, the area and power consumption can be reduced if only a limited number of turns are realized. At the same time, this would limit the flexibility of the resulting FPGA. In fact, a restricted number of turns may make it harder to realize the desired functionality. In the worst case, certain functionalities could only be realized at higher costs. However, as shown by the second observation, many turns are not that often needed by the generic routing algorithms.

Observation 2. *In order to evaluate how often turns are actually needed, we applied several examples taken from an established benchmark library (namely MCNC benchmark suite [24]). Then, we applied the* Versatile-Place-and-Route *(VPR) tool [15] in order to emulate the behavior of an FPGA as reviewed in Section II. Finally, we extracted how often certain switch-box directions are actually used by the output generated after the routing has been performed by VPR. Figure 4 summarizes the respectively obtained utilization rates for all turns selected by the routing algorithm. We use WE, EW, NS, SN, WN, WS, NW, NE, EN, ES, SW, and SE as the abbreviations of West-East, East-West, North-South, South-North, West-North, West-South, North-West, North-East, East-North, East-South, South-West, and South-East, respectively. As can be seen in the figure, just a fraction of the turns are actually needed.*

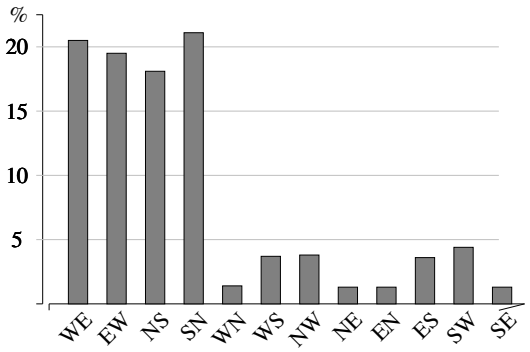

Fig. 4: Utilization of switch-box turns.

Based on these observations, the general idea of reducing area/power consumption is to realize the switch-box not in a fashion which supports *all* turns, but just a restricted set of turns. Although this might affect the performance of some functional realizations (since not all turns can be utilized anymore which may require "detours"), it definitely will allow for improvements in area and power consumption. Furthermore, since the number of SRAM cells is reduced, also benefits with respect to reliability of the FPGA can be expected. As shown later in the experimental evaluation summarized in Section V, this yields a very useful trade-off in most cases.

IV. Realization

In this section, we are describing the realization of the proposed idea sketched above. We first introduce the design flow which has been used for the realization of the baseline implementation (i.e., without optimizations). Afterwards, we discuss constraints which have to be considered when restricting the turns. Finally and based on these prerequisites, we describe the implementation of the proposed idea on top of a tool established for FPGA research.

A. Baseline FPGA Flow

In order to compare the results eventually obtained using the proposed idea, we first have to define a proper baseline, i.e., a flow for the realization of FPGA designs without any modifications. To this end, we chose the *Versatile-Place-and-Route* (VPR) tool [15] widely used for FPGA research. The VPR tool is configured by using two input files. For the configuration of the target FPGA architecture, the tool uses an XML input file. The design to be realized (which requires a dedicated placement and routing within the FPGA architecture) is handed over to the tool in BLIF format.[1] Based on these input files, VPR can now perform the place-and-route step. In order to evaluate the results generated by VPR, the tool performs, e.g., a static timing analysis to determine the critical path or the power consumption of the realized functionality. Results obtained from the tool assuming no modifications of the architecture and the routing method whatsoever represent the baseline to which we are comparing the proposed idea. This

[1]Please note that, in this work, we are assuming a BLIF file as input available. For the synthesis from, e.g., Verilog to BLIF see, e.g., ODIN II [10].

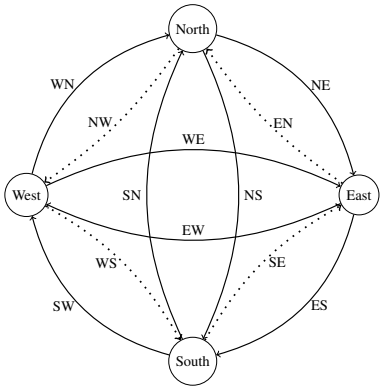

Fig. 5: Possible turns realized by a switch-box.

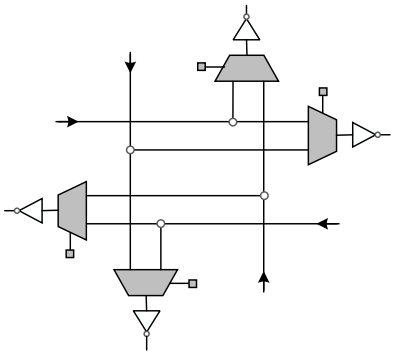

Fig. 6: Scenario-{EN, NW, WS, SE} switch-box architecture.

baseline approach allows all possible turns as sketched in Figure 5 (note that, here, e.g., North-West is abbreviated as NW). That is, a total of 12 turns are possible in this baseline approach.

B. Constrains to be considered

Having the baseline, a naive realization of the proposed idea could involve an arbitrary removal of a set of turns (and their respective hardware elements which become obsolete because of the removal). However, such an arbitrary removal is not advisable in general because not all possible turns at a fully flexible switch-box can be removed without destroying the connectivity of the switch-box. More precisely, the following constraints need to be considered when selecting the turns to be removed

1) As the utilization rate of 180° turns is rather high (see Figure 4), none of the 180° turns (e.g., EW and SN) should be eliminated.
2) There should be at least one incoming and one outgoing connection for every side of the switch-box. This is because the connections, which have to be routed through them, would be imposed to the 180° turns which already have a considerable utilization rate.
3) The immediate connections between neighboring sides should not be discarded completely. This is because the routing algorithm would not be able to make circular connections anymore which imposes a huge limit on the placement algorithm.

Example 1. *Neither NN turn nor WN and EN turns nor WS and SW turns can be eliminated at the same time according to the first, second and, third constraint, respectively. However, since it is possible to remove up to 4 out of the 12 turns, the used SRAM-cells per switch-box can be reduced from 8 cells down to 4 cells. Figure 6 shows the realization of the switch-box where the EN, NW, WS and SE turns are omitted. As can be seen the flexibility of this switch-box is* $F_s = 2$.

Overall, respecting these constraints yields a total of 34 possible scenarios, i.e., sets of allowed turns in which either one, 2, 3 or 4 turns are eliminated. In order to determine all possible scenarios, we recursively traversed all possible combinations and selected those which satisfied the constraints stated above. Depending on the respectively chosen scenario, SRAM cells which are not required anymore can be removed. This yields to a new kind of switch boxes called *Turn-Restricted Switch-Box* (TRSB) in the following.

As discussed above, this leads to a trade-off between area and power consumption and run-time performance. Turn eliminations simplify the switch-box architecture and results in area and power gain for each switch-box. The more turns are omitted from switch-boxes, the more reduction in the number of switch-boxes SRAM cells and more area and power gain. On the other hand, reducing the number of turns in the switch-box decreases the number of possible paths and, thus, the flexibility of the FPGA routing is decreased. As a result, the routing algorithm may need to employ more switch-boxes to route the paths – increasing the critical path delay. It also may increase the channel width which results in more switch-boxes to be used. As the area and power consumption depends on both the number of SRAM cells in each switch-box and the total number of FPGA switch-boxes, the area and power consumption can be reduced if the reduction of the number of SRAM cells in each switch-box dominates the increase in the total number of FPGA switch-boxes.

C. Resulting Implementation

Finally, the considerations from above can eventually be used to implement and evaluate different instances of the proposed idea (i.e., FPGA architectures with turn-aware switch boxes realizing different versions of turn eliminations/scenarios). To this end, we again utilize the VPR tool. But in contrast to the baseline realization reviewed in Section IV-A, we adapted the VPR tool to consider the adjusted switch-box realization as well as to still determine proper routing even in the absence of certain turns. For the latter, we have flagged the possible turns which cannot be used by the routing algorithm anymore. By this, the needed modifications to the routing algorithm remains rather moderate. In fact, the routing algorithm itself as well as the architecture file remain unchanged and only a setting that flagged turns are not allowed anymore is required. This makes the implementation suitable for basically any routing algorithm and FPGA architecture.

Example 2. *Consider again Figure 5 and assume that turns NW, EN, WS, and SE are eliminated. Accordingly, those turns are dotted lines in Figure 5). These flags show the correspondingly applied routing algorithm (e.g., VPR) where those turns cannot be used anymore when determining a routing in order to realize the desired functionality. Since these turns are ignored, the realization of the switch-box can be replaced with the corresponding TRSBs. The respective resulting costs in terms of area and power consumption are accordingly adjusted by the tool (c.f. Example 1).*

V. Experimental Evaluation

The proposed idea has been implemented as described above and experimentally evaluated. This section provides a summary and discussion of the respectively obtained results. To this end,

TABLE I: Baseline Results.

Benchmark	Delay(ns)	Area(#transistors)	Power(W)
alu4	4.20	970772	0.070
apex4	5.42	909E3	0.0063
bigkey	2.22	122E4	0.0091
des	3.99	132E4	0.0098
diffeq	6.52	820252	0.0047
dsip	2.44	863E3	0.0060
frisc	11.9	254E4	0.0164
misex3	5.41	814E3	0.0061
s298	8.04	129E4	0.0086
s38417	6.39	430E5	0.0305
s38584	5.10	373E4	0.0266
seq	5.16	109E4	0.0080
tseng	6.19	526208	0.0033

we first review the environment utilized for the experimental evaluation (including the used benchmarks). Afterwards, the obtained results are presented and discussed.

A. Experimental Set Up

For performing the experiments, we used VPR 7.0 [15] as a well-known tool for placement and routing in the FPGA research community. In order to model a certain FPGA behavior, we have utilized a model of a Xilinx Virtex architecture as the baseline model [21]. Based on this setting, we have realized the proposed idea for TRSB within VPR as described in Section IV.

In order to evaluate the performance of the proposed approach, we used the *Micro-electronics Center of North Carolina benchmark collection* (MCNC) [24]. The collection offers a wide range of varieties of benchmark circuits available in BLIF format which could directly been used within VPR. For our experiments, we have picked a collection of circuits with a wide design variety as stated in [5], [6], [18], [25]. By utilizing these benchmark circuits, we are able to obtain insights about the performance and the capabilities of our optimization approach based on TRSB. We used a timing-driven placement and routing algorithm and ran each benchmark with 50 different seeds per scenario. Although we did not see considerable changes across our runs, we reported the average results for each benchmark.

As cost metrics, we utilized the definitions reviewed in Section II-C which have been realized within the VPR tool as follows:

Critical Path Delay (Performance): We have obtained the critical path delay from the static timing analysis feature included in VPR [15].

Area Consumption: We have counted the transistors used for switch-boxes as well as for connection-blocks in order to measure the area.

Power Consumption: To measure the power consumption, we have used the FPGA power model, which is embedded into VPR tool. We analyzed the power using 22 nm technology. Since we focus on the optimization of the static power consumption caused by routing resources, we will later on only report the static power consumption caused by the routing resources.

B. Obtained Results

We have evaluated the performance of the proposed FPGA realization (compared to the baseline realization) for all 34 possible scenarios. However, due to page limitations, we cannot provide results for all evaluations. Instead, we present the results for the scenarios which turned out best, namely scenarios which eliminate {NE, ES, SW}-turns, {ES, SW, WN}-turns, {NW, WS, SE, EN}-turns, and {NE, ES, SW, WN}-turns.

Table I reports the resulting critical path delay (*Delay*, in ns), area (*Area*, in number of transistors), and static power consumption (*Power*, in W) of the baseline for each benchmark. Figure 7 shows the delay, area, and power consumption normalized to the baseline for the four scenarios.

The results clearly show that, using the proposed idea, area and power consumption can be substantially reduced for almost all benchmarks. In the best case, improvements of more than 40% in area and more than 60% in power consumption are reported. At the same time, the results also show the trade-off to the performance.

As discussed above, the limited routing flexibility may lead to more complex routing decisions which may increase the critical path delay. While this is partially confirmed for benchmarks in Figure 7, this increase often remains moderate. Moreover, in some of the cases even an improvement in critical path delay can be observed. This is mainly due to the fact that the VPR tool (and also other synthesizing tools) uses some heuristic approaches for place and route process, and hence, does not always offer the best place and route for a circuit. However, TRSB cannot improve the performance with respect to the baseline FPGA if the synthesizing tool outputs the best place and route for a circuit. Overall, this leads to a perfect trade-off: Area and power consumption can substantially be reduced while the critical path delay might be affected by the limited flexibility.

VI. Related Work

The idea proposed in the paper is not the first that aims for reducing the number of SRAM cells in switch-boxes of FPGAs. In this section, we briefly review related work in this area.

Several pieces of related work studied the effect of limiting the flexibility of switch-boxes (i.e., F_s) on FPGA performance and area [13], [16], [20]. In particular, these works showed routing is still possible for $F_s = 3$. However, *none* of these works have considered unidirectional MUX-based FPGAs, and these works have *not* thoroughly studied different routing patterns in FPGA switch-boxes.

Several pieces of related work have attempted to propose new switch-boxes with less switches [5], [6], [18], [25]. Sivaswamy et al. [18] proposed *HARP*, as a new group of hard-wired switch-boxes and replaced some of programmable switch-boxes with these hard-wired switch-boxes in the FPGA. Zarandi et al. [25] proposed a group of switch-boxes each with hybrid hard-wired and reconfigurable connections. Ebrahimi et al. [5] proposed the so-called *SW4-components*, that include two new bidirectional switch-box architectures, in which, two out of four switches that connect vertical and horizontal wire segments are eliminated. Ebrahimi et al. [6] proposed a new bidirectional switch-box that uses a 4×6 decoder to program the switches. Therefore, two out of six SRAM cells are omitted from all switch-boxes, while keeping the routing capacity almost unchanged. However, we are the first to (1) investigate the routing behavior of unidirectional FPGAs by focusing on turn occurrence in MUX-based switch-boxes, and based on that, (2) introduce one type of switch-box for the FPGA, rather than having various types of switch-boxes, in order to, at the same time, keep the overhead of fabrication process of the new FPGA, for realizing TRSB , as simple as the baseline FPGA and reduce the area and power consumption. Table II summarizes the key characteristics of the proposed FPGA realization in comparison to other realizations reported in the literature.

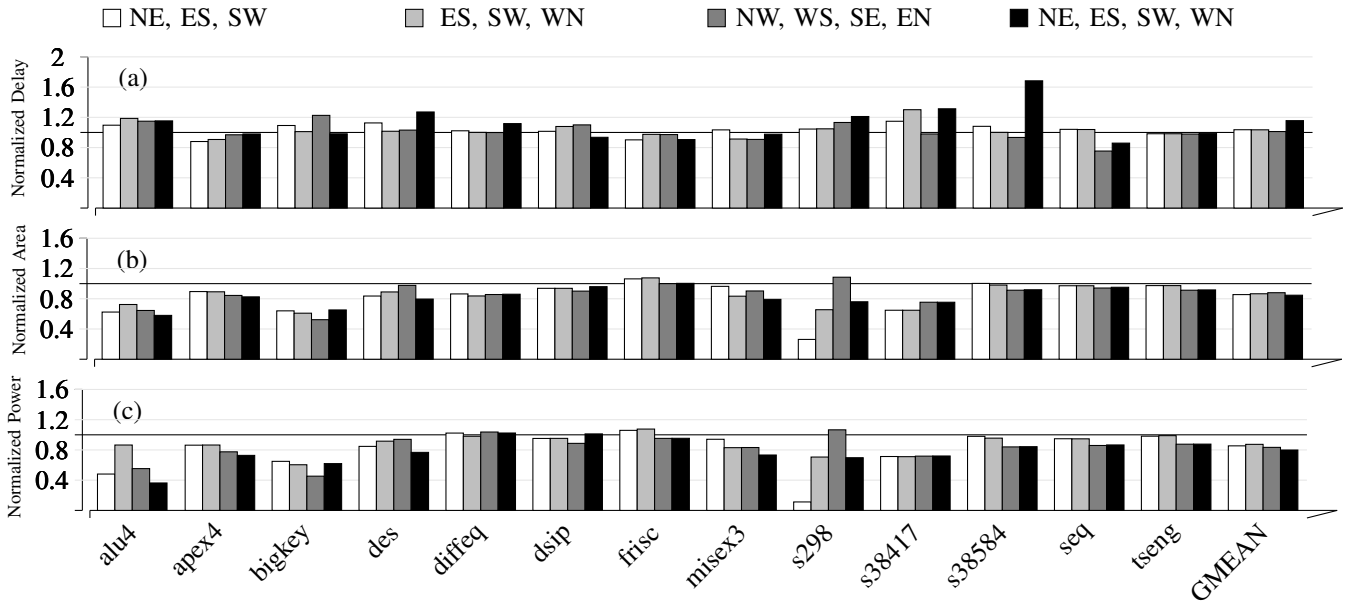

Fig. 7: Obtained results normalized to the baseline; (a) Normalized Delay, (b) Normalized Area, and (c) Normalized Power.

TABLE II: Key Characteristics of different realization techniques.

Technique	Switch-Box Structure	Routing Directionality	Number of Switch-Box Types Employed by FPGA	Area Reduction	Static Power Reduction	Performance Overhead
HARP [18]	pass-based	bidirectional	6	low	considerable	no
semi-programable SM [25]	pass-based	bidirectional	4	no	-	no
SW4 [5]	pass-based	bidirectional	2	low	-	low
decoding SB [6]	pass-based	bidirectional	1	no	low	low
proposed TRSB	mux-based	unidirectional	1	considerable	considerable	low

VII. Conclusion

This paper proposed a new group of switch-boxes for unidirectional SRAM-based FPGAs to effectively reduce area and power consumption. Based on the low utilization rate of switch-box turns, we defined three rules that allowed to omit one to four turns and their supporting SRAM cells from FPGA switch-boxes. We evaluated possible scenarios using the MCNC benchmark suite and observed that the proposed structure allows for significant reduction in area and power consumption. As expected this came at the expense of a slightly degraded performance (up to 3%) for some scenarios. However, the degrade was rather small and might not be an issue for most applications and for some cases even improvements were reported. In fact, experimental evaluations confirmed that area and power consumption could be reduced by up to more than 40% and 60%, respectively, while the decrease in performance was moderate.

Acknowledgements

This work has partially been supported by the LIT Secure and Correct Systems Lab funded by the State of Upper Austria.

References

[1] G. Asadi and M. B. Tahoori, "Soft error mitigation for SRAM-based FPGAs," in *VLSI Test Symposium*. IEEE, 2005.
[2] C. Chiasson and V. Betz, "Should FPGAs abandon the pass-gate?" in *FPL*. IEEE, 2013.
[3] X. Datasheet, "Xc4000e and xc4000x series field programmable gate arrays, datasheet."
[4] M. De Alba and et al, "FPGA design of an efficient and low-cost smart phone interrupt controller," *Latin American applied research*, 2007.
[5] H. Ebrahimi and et al, "A switch box architecture to mitigate bridging and short faults in SRAM-based FPGAs," in *DFT*. IEEE, 2010.
[6] H. Ebrahimi and et al, "Mitigating soft errors in SRAM-based FPGAs by decoding configuration bits in switch boxes," *Microelectronics Journal*, 2011.
[7] H. Giesen and et al, "Quality-time tradeoffs in component-specific mapping: How to train your dynamically reconfigurable array of gates with outrageous network-delays," in *FPGA*. ACM, 2017.
[8] V. Handbook, "Xilinx," *Inc., www. xilinx. com*, 2002.
[9] J. Huang and et al, "Fault tolerance of switch blocks and switch block arrays in FPGA," *IEEE Transactions on VLSI Systems*, 2005.
[10] P. Jamieson and et al, "Odin ii-an open-source verilog hdl synthesis tool for cad research," in *FCCM*. IEEE, 2010.
[11] I. Kuon and et al, "FPGA architecture: Survey and challenges," *Foundations and Trends in Electronic Design Automation*, 2008.
[12] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2007.
[13] G. Lemieux and D. Lewis, *Design of interconnection networks for programmable logic*. Springer, 2004.
[14] D. Lewis and et al, "The stratix 10 highly pipelined fpga architecture," in *FPGA*. ACM, 2016.
[15] J. Luu and et al, "Vtr 7.0: Next generation architecture and CAD system for FPGAs," *ACM Transactions on Reconfigurable Technology and Systems*, 2014.
[16] J. Rose and S. Brown, "Flexibility of interconnection structures for field-programmable gate arrays," *IEEE Journal of Solid-State Circuits*, 1991.
[17] Z. Seifoori and et al, "A power gating switch box architecture in routing network of sram-based fpgas in dark silicon era," in *DATE*. IEEE, 2017.
[18] S. Sivaswamy and et al, "HARP: hard-wired routing pattern FPGAs," in *FPGA*. ACM, 2005.
[19] M. B. Tahoori and et al, "Fault grading FPGA interconnect test configurations," in *International Test Conference*. IEEE, 2002.
[20] B. Tseng and et al, "Improving fpga routing architectures using architecture and cad interactions," in *ICCD*. IEEE, 1992.
[21] I. Xilinx, "Virtex-4 family overview," *DS112 Version*, vol. 1.4, 2006.
[22] I. Xilinx, "Virtex-6 family overview," 2010.
[23] I. Xilinx, "Virtex-5 user guide," *UG190 Version*, vol. 2.1, October 2006.
[24] S. Yang, *Logic synthesis and optimization benchmarks user guide: version 3.0*. Microelectronics Center of North Carolina (MCNC), 1991.
[25] H. R. Zarandi and et al, "Soft error mitigation in switch modules of SRAM-based FPGAs," in *ISCAS*. IEEE, 2007.