A Study on Switch Block Patterns for Tileable FPGA Routing Architectures

Xifan Tang, Edouard Giacomin, Aurélien Alacchi and Pierre-Emmanuel Gaillardon
University of Utah
Email: xifan.tang@utah.edu

***Abstract*—Following the rapid growth of *Field Programmable Gate Arrays* (FPGAs) sizes, the regularity of architectures has become a critical feature, leading to the development of million-of-LUT devices. While the routing architecture plays a dominant role in the area, delay and power of modern FPGAs, most of previously published works focus on improving the routability and performance of FPGAs while very few studied tileable (highly-regular) routing architectures. In this paper, we provide a detailed analysis between tileable and popular non-tileable FPGAs considering modern routing architectures. First, we upgrade VPR to generate tileable routing architecture, which can support different switch block patterns for (1) the routing tracks that start/end in a tile and (2) the routing tracks that pass through a tile. Then, we evaluate the performance of mixed switch blocks patterns in the context of a Stratix IV-like FPGA architecture, by considering the most representative patterns, i.e., Subset, Universal and Wilton. Experimental results show that averaged over the MCNC and VTR benchmarks, when compared to the well-optimized non-tileable architectures, the tileable architectures can improve the minimum routable channel width by 13% and area-delay product by 2%. In particular, our results showed that in the context of tileable FPGA, a mix of Universal and Wilton switch block patterns lead to the best trade-off in area, delay and routability, while Wilton switch block was the best choice in non-tileable FPGAs.**

I. Introduction

The routing architecture implementation has a dominant impact on the area, delay and power of modern *Field Programmable Gate Arrays* (FPGAs) [1], [2]. As FPGA sizes grew rapidly, highly-regular architectures, especially tile-based FPGAs, have become mainstream in commercial products [3], [4]. Tileable architectures remarkably simplify the development of FPGA layouts, as the full fabric can be built with a small number of repeatable tiles. For full-custom designed FPGAs, manual layouts are only required for a small number of tiles, reducing engineering efforts as well as development cost [2]–[9]. For semi-custom designed fabrics [10]–[14], the runtime and memory usage of backend flows can be potentially reduced by exploiting the hierarchical P&R provided by cutting-edge tools. Nevertheless, very limited published works studied the specifics of tileable FPGAs, in particular the routing architecture [2], [5]–[7]. Previous works mainly focus on improving the routability and performance of FPGAs using VPR, which produces non-tileable unidirectional fabrics [15]–[23]. Moreover, switch block patterns have a crucial impact on the routing architecture, while previous works assume simple conditions, e.g., length-1 wires and bidirectional routing tracks, which are rarely used in modern FPGAs [19]–[22]. Note that major works only considered uniform switch block patterns even for various lengths of routing tracks [19]–[21], while very limited work has been done for mixed switch block patterns [15], [22].

In this paper, we provide a detailed analysis between fully tileable and non-tileable FPGAs considering modern routing architectures. Our contributions are as follows:
(1) We upgrade VPR with a tileable *Routing Resource Graph* (RRG) generator, which can generate regular routing architecture for both homogeneous and heterogeneous FPGAs. Experimental results show that compared to VPR, our RRG generator can reduce the number of unique tiles by $8.8\times$ and $5.5\times$ for homogeneous and heterogeneous FPGAs respectively, even considering 128×128 array size.
(2) More than tileable FPGA, our RRG generator also supports different switch block patterns for (a) the routing tracks that start/end in a tile and (b) the routing tracks that pass a tile. We evaluate the performance of mixed switch blocks patterns in the context of a Stratix IV-like FPGA architecture, by considering the most representative patterns, i.e., Subset [20], Universal [19], Wilton [21] and Imran [22]. Experimental results show that averaged over the MCNC and VTR benchmarks, when compared to the well-optimized non-tileable architectures, the tileable architectures can improve the minimum routable channel width by 13% and area-delay product by 2%. In particular, our results showed that in the context of tileable FPGA, a mix of Universal and Wilton switch block patterns leads to the best trade-off in area, delay and routability, while Wilton switch block was the best choice in non-tileable FPGAs.

The rest of this paper is organized as follows. Section II reviews the modern FPGA architectures and related works. Section III introduces the tileable routing architecture generator. Section IV presents experimental results. Section V concludes this paper.

II. Background

This section will first introduce the tile-based modern FPGA architectures and then briefly review previous works on routing architectures.

A. Tile-based FPGA Architecture

Modern FPGA architectures are commonly organized in a high-granularity island style, which consists of an array of repeatable tiles. As shown in Fig. 1, each tile consists of a *Configurable Logic Block* (CLB), two *Connection Blocks* (CBs) which connect CLB pins to X-direction and Y-direction routing

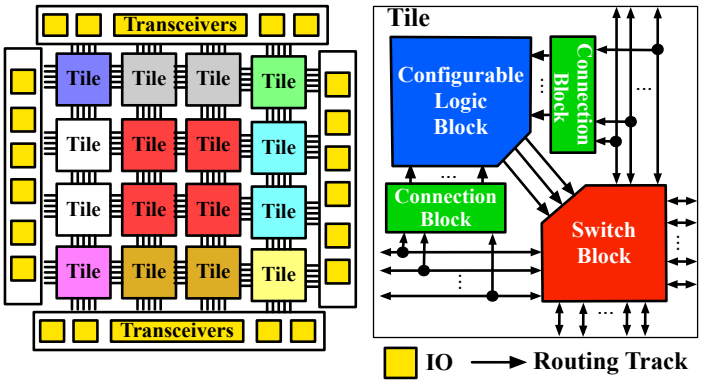

Fig. 1: Tile-based FPGA architecture: tiles in the same color are repeatable.

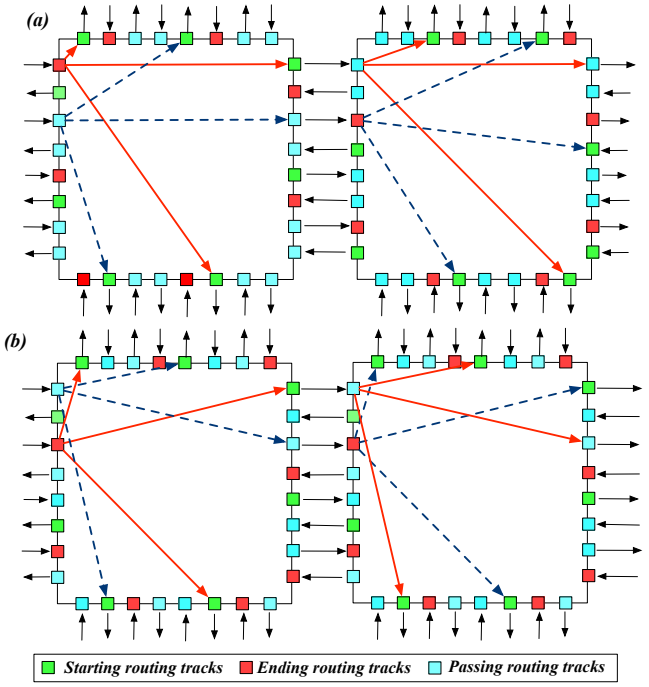

Fig. 2: An illustrative example of subset switch block pattern interconnecting uni-directional Length-2 wires in : (a) non-tileable (untileable) architecture and (b) tileable architecture.

tracks respectively, and a *Switch Block* (SB) that interconnects routing tracks. An FPGA is considered tileable when all the tiles except the periphery tiles share exactly the same internal structures and layouts. Fig. 1 shows an example of tileable FPGAs, where the replicated tiles are highlighted with the same color. The main advantage of tileable architecture is that only one tile will be laid out and can be assembled to create any sizes of FPGAs. However, the majority of previous works are based on the unidirectional FPGAs modeled by VPR [23], which is not tileable, while very limited works studied tileable FPGA [2], [6], [7]. For instance, I. Kuon *et al.* proposed a tileable FPGA generator, GILES [2], which can generate any FPGA fabric using 9 unique tiles. Nevertheless, none of these works provided a detailed performance analysis comparing tileable and non-tileable FPGAs.

B. Previous works on Switch Block Patterns

Switch blocks have a crucial impact on most critical paths in FPGAs, and may contribute up to 50% of the delays [1]. Through twenty-year of intensive research, there are three most popular switch block patterns standing out for best area-delay trade-off: Subset [20], Universal [19], Wilton [21]. Fig. 2(a) depicts an example of subset switch block pattern in the context of uni-directional routing architecture using length-2 wire segments. Note that when length-X ($X > 1$) wire segments are used in a switch block, there are a larger number of routing tracks that just pass through than those that start/end. Most previous works applied uniform switch block patterns to the starting, ending and passing tracks [15]–[23], while very limited works tried separated patterns [15], [22], [23]. For example, the popular architecture exploration tool VPR assigns passing tracks to starting tracks following a round-robin scheme [23]. This indeed balances the multiplexer sizes of SBs, but it does make the architecture not tileable. Also, it remains an open question if the round-robin scheme is superior to other interconnection patterns. Moreover, early studies assume simple conditions in routing architectures, e.g., length-1 wires and bidirectional routing tracks, which are rarely used in modern FPGAs [19]–[22]. Only a few recent works performed practical analysis considering multi-length wires and uni-directional routing [15], [23]. Considering these facts, the switch block patterns do require a revisit in the context of modern tileable FPGA architectures.

III. Tileable Routing Resource Graph Generation

In VPR, FPGA routing architectures are modeled by the *Routing Resource Graph* (RRG), where each node represents routing tracks and CLB inputs/outputs, while edges denote interconnections. In this section, we propose a tileable RRG generator, which (1) can automatically arrange the distribution of wire segments for routing channels in a tileable style; and (2) can apply different switch block patterns for passing tracks than starting/ending tracks. In addition, the proposed tileable RRG generator is compatible with the wide set of routing architecture parameters that VPR supported, including connectivity parameters $F_{c,in}$, $F_{c,out}$, F_s, CB/SB population and various wire segments. The RRG generator is implemented in VPR, available at [25].

A. Auto-arrange Tileable Routing Channels

A tileable routing architecture forces a few restrictions on the width of routing channels as well as array sizes. First, for each segment group, the number of routing tracks should be the multiple of its length. Take the example of length-2 wires in Fig. 2(b); the number of length-2 routing tracks should be an even number. Otherwise, due to the wire twisting, the last few tracks in the group will be length-1 wires instead of length-2. Therefore, considering various lengths of wire segments, the width of a tileable routing channel (W) is constrained by:

$$W = \sum_{L} f_L \cdot L, \tag{1}$$

where L denotes the length of each type of wire segment and f_L is the frequency of wire segments in a routing channel. This requires designers to carefully craft the W w.r.t. L and f_L. To avoid such effort, we adapted the channel builder of VPR to arrange routing channels by adding groups of wire segments according to their lengths. As detailed in Algorithm 1, the builder keep adding a set of tracks whose size equals to the length of wires, until the channel width is satisfied. Note that *assigned_W* may not be the same as *required_W* but is the closest number which is tileable. To guarantee tileable architectures even in search of best routing channel width, the

wire_types: *a list of wire types.*
required_W: *Routing channel width specified by users.*
Function `chan_builder(`*required_W, wire_types*`)`**:**
  foreach $L \in$ *wire_types* **do**
    `compute_num_tracks_per_type();`
  end
  while *assigned_W* $<$ *required_W* **do**
    max_{f_L} = `max_num _tracks(`*wire_types*`);`
    assigned_W += `L_of_wire(`max_{f_L}`);`
    max_{f_L} -= `L_of_wire(`max_{f_L}`);`
  end
end

Algorithm 1: Tileable routing channel arranging algorithm (pseudo-code).

```
<!-- Switch block with a mix of Subset and Universal
patterns -->
<switch_block type="subset" fs="3" sub_type="universal"
sub_fs="3"/>
```

Fig. 3: Examples of extended XML syntax for switch block patterns.

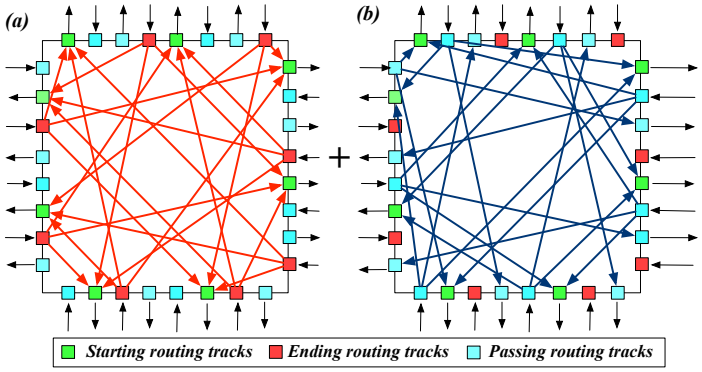

Fig. 4: An illustrative example of mixed switch block patterns: (left) starting/ending tracks follow the Subset pattern; (right) passing tracks follow the Universal pattern and are bent in a staggered style [6]; both are combined in the actual switch block.

binary-search placement & routing engine is adapted to use the tileable W provided by Algorithm 1. Similarly, the array size should be the multiple of all the lengths of wires, in order to guarantee regular wires even at the fringes of FPGAs. Consider the example in Fig. 2(b), when the array width is an odd number, there would be length-1 wires at the left/right borders of FPGAs. In this paper, to guarantee the fine-granularity, we consider both restrictions in sizing the FPGAs.

B. Support for Mixed Switch Block Patterns

To customize the switch block patterns for passing tracks, we added two new XML tags to the VPR architecture description language [23], as shown in Fig. 3. The XML property `sub_type` specifies the interconnection pattern for passing tracks, while `sub_fs` denotes the connectivity of passing tracks. Supporting the most popular patterns, `type` and `sub_type` could be any combination of Subset, Universal and Wilton. In addition, `sub_fs` can be different than the connectivity of starting/ending tracks `fs`, enabling a larger architecture exploration space than before. Fig. 4 depicts the corresponding switch block in an illustrative case, where uni-directional length-2 wires are interconnected in a routing channel width of $W = 4$. Note that when the length of wires is larger than 2, the number of passing tracks will be much larger than the number of start tracks in a switch block. Our RRG generator applies a round-robin scheme to routing tracks on each side of switch blocks (VPR does similarly but considering all the routing tracks as a group), in order to balance the multiplexer size. For each routing track, our RRG is larger in routing multiplexer sizes than VPR but more routable. To exploit the tileable RRG, we will study the best combination of switch block patterns in Section IV-C.

IV. Experimental Results

In this section, we first introduce the experimental methodology and then evaluate the performance of tileable architectures by comparing to non-tileable non-tileable ones.

A. Experimental Methodology

In this paper, we considered a Stratix-IV-like FPGA architecture using a commercial 40nm technology, modeled by the VTR project [23] [1]. Each CLB consists of 10 BLEs and a local routing architecture with 50% connectivity, and each BLE can operate at either 6-LUT or dual 5-LUT or arithmetic modes. DSP and memory blocks are repeated in every 8 columns. Global routing is uni-directional with 80% of length-4 routing tracks and 20% of length-16 routing tracks. CB connectivity is set to $F_{c,in} = 0.15$ while SB connectivity is set to $F_{c,out} = 0.1$. For a fair comparison, non-tileable FPGAs can use any routing channel width (W), while the tileable FPGAs is constrained by equation 1. Two sets of evaluation are performed with the MCNC big20 and VTR benchmarks [23], [24]: (1) the traditional architecture exploration flow [1], where area utilization and critical path delays are achieved by adding a 30% slack to the best routing channel width (W_{min}); (2) layout-level fabric comparison, where we employ a synthesizable FPGA design flow [14] and evaluate the layout area between two FPGAs. In this case, we set $W = 320$ for both FPGAs, similar to commercial products [8], [9].

B. Architecture Regularity and Layout Area

To validate the regularity of our tileable RRG, we use graph matching to identify the replicated CBs and SBs in both tileable and non-tileable architectures. For a fair comparison, we consider only tileable array sizes for both architectures, ranging from 16×16 to 128×128. Table I compares the number of unique tiles in both architectures. Even though the unidirectional architecture from VPR is not designed to be tileable [23], there are still many tiles that are repeatable when applying staggered routing segments [6]. However, the number of unique tiles remain too large, requiring complete manual layouts for a $\sim 9 \times 9$ array. Note that FPGA regularity could be much worse when longer wire segments (> 16) are introduced. In contrast, the tileable architecture has a constant number ($= 9$) of unique tiles, as colored in Fig. 1, whatever the array sizes are. The results prove the proposed RRG can minimize the number of tiles to be laid out, showing an improvement of $9.3\times$ compared to non-tileable architectures. To propose an apple-to-apple comparison, we compare the layout area of the two architectures using OpenFPGA, a fast FPGA prototyping tool [14]. The array sizes are fixed to 32×16, which is the minimum tileable array size that fits the largest MCNC big20 benchmark. The non-tileable FPGA consumes $14.016mm^2$ while the tileable FPGA requires $14.008mm^2$, showing the same area efficiency.

TABLE I: Number of unique tiles in FPGAs.

#. of Unique Tiles	Homogeneous		Heterogeneous	
Array Size	VPR	Tileable	VPR	Tileable
16×16	80	9	148	27
32×32	84	9	174	27
64×64	84	9	178	27
128×128	84	9	174	27

C. Best Switch Block Patterns

We then investigate the best switch block patterns for tileable routing architecture, by enumerating all the possible combinations of Subset, Universal and Wilton patterns for switch blocks. Fig. 5 compares the area, delay and minimum routing channel width W_{min} between the non-tileable and tileable architectures. In non-tileable architecture, Wilton and Universal patterns are the best choices, being 2% smaller in W_{min} and 1% better in area-delay product than Subset. The conclusion is consistent with previous works and also explains

[1] https://github.com/verilog-to-routing/vtr-verilog-to-routing/blob/master/vtr_flow/arch/timing/k6_frac_N10_frac_chain_depop50_mem32K_40nm.xml

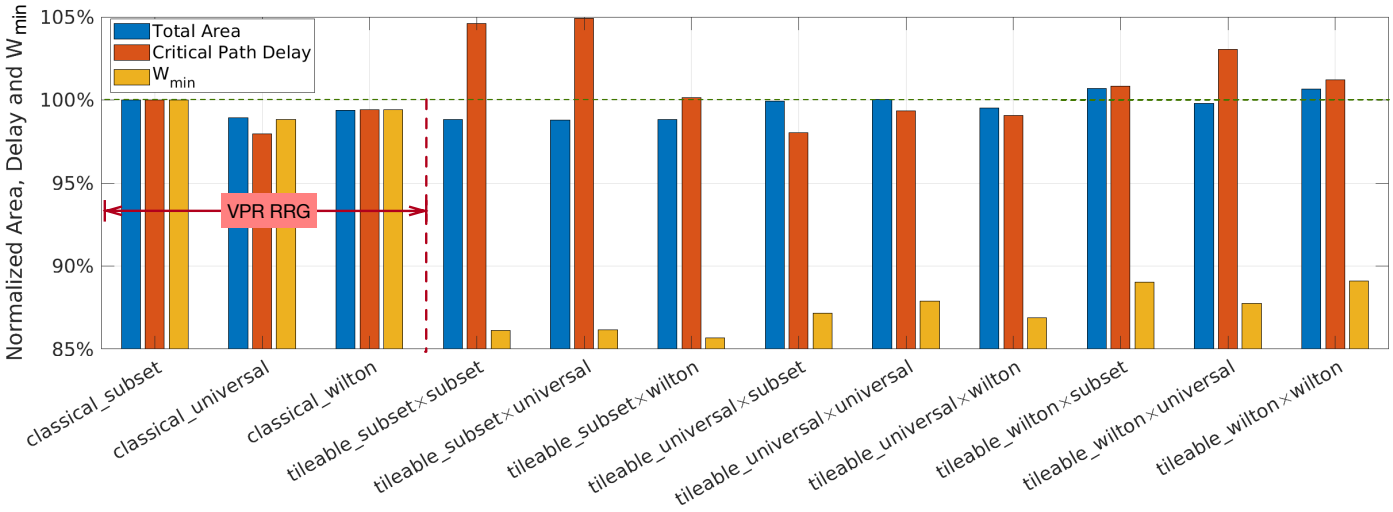

Fig. 5: Area, delay and channel width comparison between non-tileable and tileable FPGAs using different switch block patterns averaged over MCNC big-20 and VTR benchmarks (subset×universal denotes using subset pattern for start/ending tracks and universal for passing tracks).

well why Wilton SB is popular in many FPGA research papers. Differently, in tileable architecture, Subset patterns are the most routable but with a 3% overhead in area-delay product than the baseline. When considering mixed switch block patterns, using Universal for starting/ending tracks and Subset for passing tracks (Universal×Subset) leads to the best area-delay product. On average, the tileable FPGAs improve 13% in W_{min} than the non-tileable counterparts. This is due to that the passing tracks use Subset/Universal/Wilton are more routable than those using round-robin schemes. When compared to the best non-tileable architecture, the Universal×Subset tileable FPGA has a 2% area-delay product improvement, showing the promise of tileable architectures.

V. Conclusion

In this paper, we provide a detailed analysis of tileable and non-tileable FPGAs considering modern routing architectures. We propose a tileable *Routing Resource Graph* generator which can support different switch block patterns for (1) the routing tracks that start/end in a tile and (2) the routing tracks that pass a tile. Experimental results show that averaged over the MCNC and VTR benchmarks, when compared to the well-optimized non-tileable architectures, the tileable architectures can improve the minimum routable channel width by 13% and area-delay product by 2%. In particular, our results showed that in the context of tileable FPGA, a mix of Universal and Subset switch block patterns lead to the best trade-off in area, delay and routability.

Acknowledgment

This material is based on research sponsored by Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under agreement number FA8650-18-2-7855. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) or the U.S. Government.

References

[1] V. Betz *et al.*, *Architecture and CAD for Deep-Submicron FPGAs*, Kluwer Academic Publishers, 1998.

[2] I. Kuon *et al.*, *Design, Layout and Verification of an FPGA Using Automated Tools*, ACM/SIGDA FPGA, 2005, pp. 215-226.

[3] S. Feng *et al.*, *Tileable Field-Programmable Gate Array Architecture*, U.S. Patent 7,015,719, 2006.

[4] D. Tavana *et al.*, *FPGA architecture with Repeatable Tiles Including Routing Matrices and Logic Matrices*, U.S. Patent 5,914,616, 1999.

[5] P. Chow *et al.*, *The Design of a SRAM-based Field-Programmable Gate Array-Part II: Circuit Design and Layout*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 7, no. 3, pp. 321-330, 1999.

[6] V. Betz *et al.*, *Automatic Generation of FPGA Routing Architectures from High-level Descriptions*, ACM/SIGDA FPGA, 2000, pp. 175-184.

[7] S. A. Razavi *et al.*, *A Tileable Switch Module Architecture for Homogeneous 3D FPGAs*, IEEE International Conference on 3D System Integration, 2009, pp. 1-4.

[8] Altera Corporation, *Stratix IV device handbook version SIV5V1-4.8*, January 2016, available online. https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/stratix-iv/stratix4_handbook.pdf

[9] Xilinx Corporation, *7 Series FPGAs Data Sheet: Overview DS180 (v2.6)*, February 2018, available online. https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf

[10] J. Kim *et al.*, *Synthesizable Standard Cell FPGA Fabrics Targetable by the Verilog-to-Routing (VTR) CAD Flow*, ACM Transactions on Reconfigurable Technology and Systems, Vol. 10, No. 2, 2017.

[11] B. Grady *et al.*, *Synthesizable Heterogeneous FPGA Fabrics*, IEEE FPT, 2018, pp. 1-8.

[12] H. Liu, *Archipelago - An Open Source FPGA with Toolflow Support*, Master Thesis, University of California, Berkeley, 2014.

[13] X. Tang *et al.*, *FPGA-SPICE: A Simulation-Based Architecture Evaluation Framework for FPGAs*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 27, No. 3, pp. 637-650, 2019.

[14] X. Tang *et al.*, *OpenFPGA: An Opensource Framework Enabling Rapid Prototyping of Customizable FPGAs*, IEEE FPL, 2019, pp. 1-8.

[15] O. Petelin *et al.*, *The Speed of Diversity: Exploring Complex FPGA Routing Topologies for the Global Metal Layer*, IEEE FPL, 2016, pp. 1–10.

[16] S. Sivaswamy *et al.*, *HARP: Hard-wired Routing Pattern FPGAs*, ACM/SIGDA FPGA, 2005, pp. 21–29.

[17] G. Lemieux *et al.*, *Directional and Single-driver Wires in FPGA Interconnect*, IEEE International Conference on Field-Programmable Technology(FPT), 2004, pp. 41–48.

[18] M. Lin *et al.*, *TORCH: A Design Tool for Routing Channel Segmentation in FPGAs*, ACM/SIGDA FPGA, 2008, pp. 131–138.

[19] Y.-W. Chang *et al.*, *Universal Switch Blocks for FPGA Design*, ACM Transactions Design Automation of Electronic Systems, Vol.1, No.1, pp. 80–101, 1996.

[20] G. Lemieux *et al.*, *On Two-step Routing for FPGAs*, International Symposium on Physical Design, 1997, pp. 60–66.

[21] S. Wilton *et al.*, *Architecture and Algorithms for Field-Programmable Gate Arrays with Embedded Memory*, PhD thesis, University of Toronto, 1997.

[22] M. Imran Masud *et al.*, *A New Switch Block for Segmented FPGAs*, IEEE FPL, 1999, pp. 274-281.

[23] J. Luu *et al.*, *VTR 7.0: Next Generation Architecture and CAD System for FPGAs*, ACM Transaction Reconfigurable Technology System, Vol. 7, No. 2, 2014.

[24] S. Yang, *Logic Synthesis and Optimization Benchmarks User Guide Version 3.0*, MCNC, Jan. 1991.

[25] https://github.com/LNIS-Projects/OpenFPGA