An Open-Source Tool to Model and Explore Complex Routing Architecture for FPGA

Kaichuang Shi, Lingli Wang*
State Key Laboratory of Integrated Chips and Systems
Fudan University, Shanghai, China
{19112020058, llwang}@fudan.edu.cn

***Abstract*—Routing architecture has a large impact on the FPGA performance and area. In academia, the routing architecture is mainly based on the connection blocks (CBs) and switch blocks (SBs) which is used in VPR. And there are input crossbars inside the logic blocks (LBs). The routing architecture in VPR is not tileable. Besides, it is hard to model the complex routing architecture as in commercial FPGAs. In this paper, we model a tile-based VRB (Versatile Routing Block) architecture which replaces the CBs, SBs and input crossbars to alleviate this problem. All the routing resources are included in the VRBs and many routing features which are used in commercial FPGAs are supported, such as bent wires, nearest neighbor interconnects and two-level mux topology. In addition, VTR 8 is enhanced to support the VRB architecture and we make it open publicly[1]. Experimental results show that the proposed VRB architecture can achieve 8.2% improvement on the critical path delay and 8.4% improvement on the area-delay product compared to the latest two-level mux architecture.**

***Index Terms*—FPGA, routing architecture, route, VPR, open-source**

I. Introduction

FPGAs are widely used due to the fact that they have superiority in time to market, non-recurring engineering (NRE) cost and flexibility [1]. Routing architecture has a large impact on the FPGA area and performance. VPR (Versatile Place and Route) is an open-source academic CAD tool which is widely used to carry out the architecture exploration and CAD researches for FPGA. The most common routing architecture in academia is mainly based on the CBs and SBs which is used in VPR [2]. CBs are used to connect wire segments with LB pins while SBs provide programmable switches to connect with different wire segments. Inside the LB, there is a local crossbar to distribute the LB inputs and the local feedback signals to the LUTs of the LB. Many researches have been done to optimize the CB-SB routing architecture.

However, the CB-SB architecture is different from the routing architecture in commercial FPGAs. The GRM routing architecture in Xilinx FPGAs is widely used [3]. The GRM architecture is tile-based and it contains complex multi-level mux topology [4]. F4GA [5] is an open-source flow for hardware description language to FPGA bitstream targeting some commercial FPGAs. However, it is hard to do architecture exploration as a routing resource graph (RRG) file is needed and multi-level mux topology is not supported. Recently, there are some papers proposing new routing architectures to replace the CB-SB architecture. For example, GIB [6], INTB [7], GRB [8], VIB [9]. But all of them are not open-source and they are not versatile enough to model the commercial FPGAs.

In this paper, we propose a VRB routing architecture and make the enhanced VPR tool open to narrow the gap between academic research and industry development. Our contributions include:

- We enhance the FPGA architecture description format to describe the VRB architecture. In order to evaluate the performance of the architecture, we enhance the Routing Resource Graph (RRG) generator in the latest VTR 8 [2].
- We evaluate the performance of the VRB architecture whose area and delay parameters are extracted from COFFE 2 [10] with VTR benchmarks. Experimental results show that the VRB architecture can achieve 8.2% improvement on the critical path delay and 8.4% on the area-delay product compared to the two-level mux architecture in [11].

The rest of this paper is organized as follows. Section II introduces the academic CB-SB routing architecture and the related work. Section III presents the VRB architecture and the new features added in the routing architecture. Section IV gives the enhancements in VTR 8 to support the proposed architecture. Section V presents the baseline architecture and experimental results. Section VI concludes this paper with future work.

II. Background and related work

A. Routing Architecture of Island-Style FPGAs

Island-style FPGA architecture mainly contains LBs, CBs and SBs. And they are interconnected by vertical and horizontal routing wires. Fig. 1 shows the CB-SB routing architecture. W is the routing channel width and the wire length L defines the number of LBs that the wire segment spans. CB flexibility $F_{c,in}$ & $F_{c,out}$ represent the fraction of wire segments in routing channel that an LB input and output pin can connect to through CB respectively. SB flexibility F_s defines the number of other wire segments that an incoming wire segment can connect through SB.

B. Related Work

There are many open-source frameworks which target FPGA CAD algorithm and architecture exploration. Open-

[1]https://github.com/shikc/VTR_Complex_Interconnect

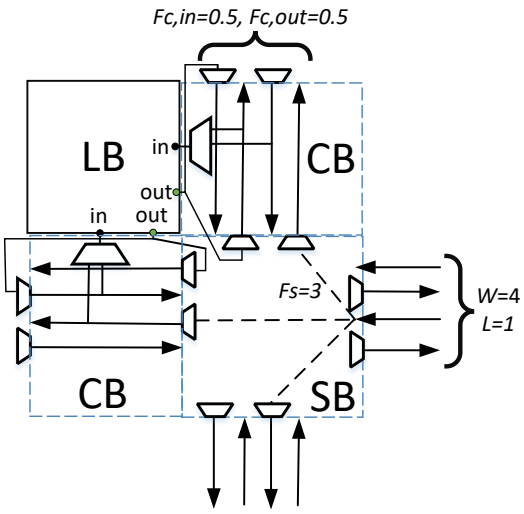


Fig. 1. The detailed CB-SB routing architecture.

PARF [12] is an academic placement and routing (PAR) engine for heterogeneous FPGAs which is implemented with deep learning toolkit. However, it targets to the Ultrascale FPGAs [13] and no timing information is supported which means it is not timing-driven. It is not impractical to carry out architecture research. RapidSmith [14] is an open-source framework to enable researchers to directly experiment with CAD tools for Xilinx FPGAs. Similarly, RapidSmith only targets to the commercial Xilinx FPGAs which makes it impossible to perform architectural exploration.

VTR has been commonly used as the infrastructure to conduct FPGA CAD and architecture research and development [2]. VTR takes an XML-based architecture description file and a digital circuit described in Verilog as inputs. There are three components: Odin-II [15], ABC [16] and VPR [17]. The Odin-II elaborates and synthesizes the Verilog design into a flatted netlist containing logical gates, flip-flops and blackboxes. Then, ABC is used to perform logic optimization and technology mapping. Finally, VPR packs the netlist into the logic blocks, places and routes the circuit in the FPGA. In addition, VPR will perform the result analysis and report the statistics, such as the circuit performance, area, power and total routed wirelength. Besides, VTR provides many heterogeneous benchmark circuits which are suitable for regression tests [18].

Many researches about FPGA architecture have been performed. For example, the exploration of LB size [19], LUT size [20], LB local connectivity [21], optimizing the DSP blocks for complex arithmetic operations [22] and developing the routing architecture including SB patterns [23] [24] [25], CB flexibility [26] and wire lengths [27] [28]. In addition, many VTR enhancements are proposed to explore novel FPGA routing architecture. In [29], X. Sun et al. proposed a bent routing pattern which can span in vertical and horizontal wires without passing through programmable switches and it can improve area-delay product by 11% compared to the architecture with length-4 straight wires. K. Shi et al. proposed GIB routing architecture to replace the traditional CB-SB architecture [6] [30], and experimental results show that the GIB architecture can achieve 11.1% improvement on the area-delay product compared to the CB-SB architecture. In [31], K. Shi et al. proposed hexagon-based honeycomb routing architecture and experimental results show that it can achieve 9.9% improvement on the routed wirelength, 11.5% on the critical path delay and 12.4% on the area-delay product compared to the traditional rectangular architecture. In [32], X. Tang et al. explored tileable routing architecture, and experimental results show that the tileable architectures can improve the minimum routable channel width by 13% and area-delay product by 2% compared to the well-optimized non-tileable architectures in VTR. Tile-based architecture has also widely used in commercial FPGAs [3] [33]. Because the full fabric can be built with a small number of repeatable tiles which can simplify the development of FPGA layouts. In [8], J. Qian et al. proposed GRB routing architecture to model two-level muxes with output sharing and bent wires which are not supported in VTR. Then, K. Shi et al. explored the feedback interconnects in GRB architecture [34]. VTR is very powerful in the support of logical block. It can support the latest Intel Stratix 10 [35] and Agilex [33] devices including the complex DSP and embedded memory block [36]. However, the support of routing architecture is not powerful enough to model the commercial FPGAs. For example, the VPR architecture file format and routing resource graph generator do not support multistage routing switches [36]. The source of LB input muxes (LIMs) can only from wire segments, no LB output feedbacks. The feedback interconnects only exist in the local interconnect inside the LB. In addition, the detailed routing architecture can only be generated automatically through the abstract parameters as introduced in Section II-A and it is hard to adjust the routing switches in detail. These deficiencies in routing architecture inspire us to carry out the work. In this paper, we enhance VTR to support complex FPGA routing architecture and make it open publicly.

III. VRB Architecture

In this section, we propose tile-based VRB architecture. Then, we introduce the new features we added in VPR to model the complex interconnect architecture in the commercial FPGAs.

A. Tile-based VRB Architecture

Fig. 2 shows the proposed VRB architecture which is tile-based. Each tile is composed of an LB and a VRB. Each LB can interact with the corresponding VRB which contains all the routing programmable switches in one tile. Arbitrary level muxes are supported in VRB. Fig. 3 shows an example of the detailed interconnect architecture in VRB, the node 6 can not only get connections from node 0 through a two-level mux architecture (mux-0 and mux-2), but also can get connections from node 4 through a one-level mux architecture

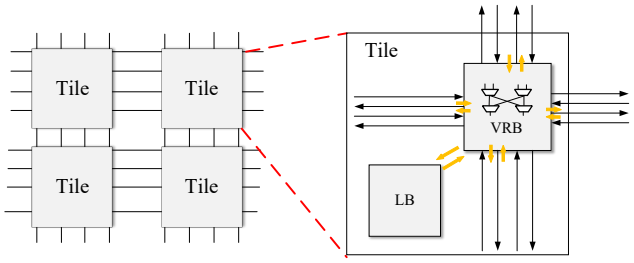


Fig. 2. The VRB architecture.

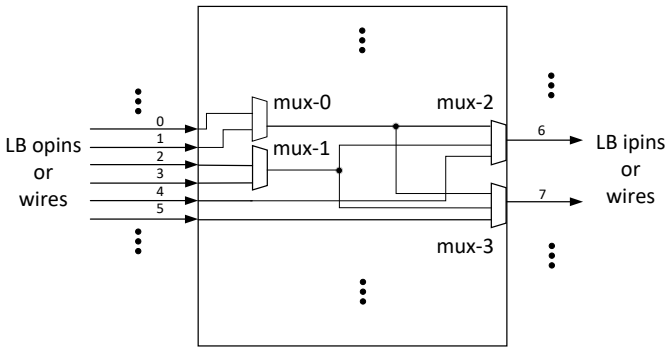
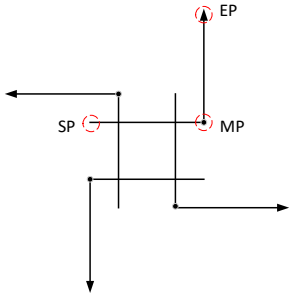


Fig. 3. An example of the detailed interconnect architecture in VRB.

(mux-2). For convenience, all the nodes go into the VRB are drawn in the left of this figure and all the nodes leave the VRB are drawn in the right of this figure. In VRB, we can customize the connections of arbitrary level muxes as we need. The intermediate nodes between different muxes are modeled by length-1 wires without timing cost.

B. Bent Wires

Bent (Diagonal) wires are widely used in many commercial FPGAs. For example, length-5 bent wires in Xilinx Virtex-5 FPGAs [3], length-2 and length-6 bent wires in Xilinx 7-series FPGAs [4]. Each length-L wire segment contains one start point (SP), $(L-1)$ middle points (MPs) and one end point (EP). Fig. 4 shows one example of length-2 bent wires. Each wire segment can be driven by a buffered mux at the SP and connect to the other wire segments at the MPs and EP. There are two optional bent types $\{CC, CW\}$ for each MP. CC is the counterclockwise type, and CW represents the clockwise type. We enhance the VTR architecture description file format to describe the bent wires which is similar to [29]. We add the $<bend>$ tag to represent the bent type at each MP. In addition to the bent type (CC and CW), straight (ST) type is also supported. Fig. 4 shows one example of length-2 bent wires and the corresponding architecture description. The length-2 bent wire has one MP whose bent type is CC. The other tags in the architecture description file are the same as that in the original VTR.



(a) Lenth-2 bent wires.

```
<segment name="l2"
         freq="32"
         length="2"
         type="unidir"
         Rmetal="0.0"
         Cmetal="0.0">
    <mux name="2"/>
    <sb type="pattern">1 1 1</sb>
    <cb type="pattern">1 1</cb>
    <bend>CC</bend>
</segment>
```

(b) The corresponding description.

Fig. 4. One example of length-2 bent wires and the corresponding architecture description file.

```
<muxes>
    <mux name="mux-0">
        <from type="seg" name="l1" from_details="S0 E1" switchpoint="0"/>
    </mux>
    <mux name="mux-1">
        <from type="seg" name="l1" from_details="E2" switchpoint="0"/>
        <from type="seg" name="l2" from_details="E0" switchpoint="0"/>
    </mux>
    <mux name="mux-2" to_pin="i:0">
        <from type="mux" from_details="mux-0 mux-1"/>
        <from type="seg" name="l2" from_details="E1" switchpoint="0"/>
    </mux>
    <mux name="mux-3" to_seg_name="l1" to_track="E0">
        <from type="mux" from_details="mux-0 mux-1"/>
        <from type="pb" name="lb" from_details="o:0"/>
    </mux>
</muxes>
```

Fig. 5. An example of the detailed interconnects description format in VRB.

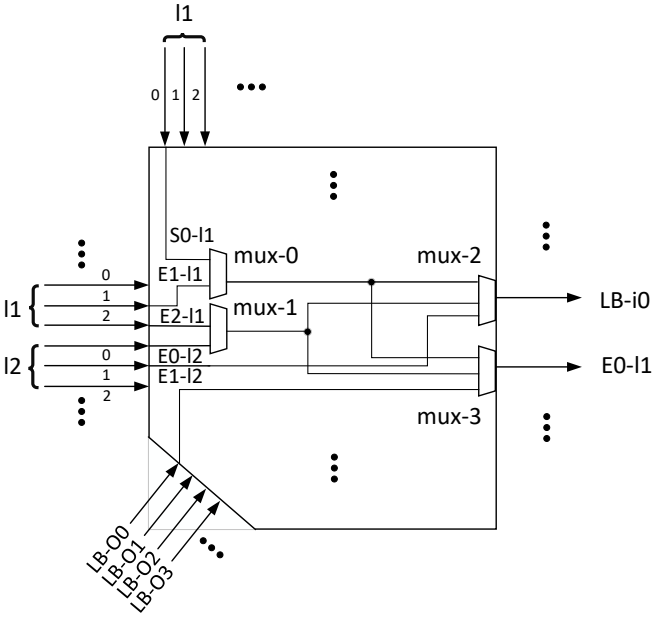


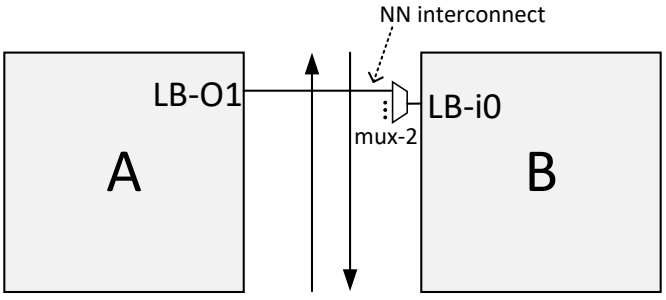
Fig. 6. The corresponding detailed architecture of Fig. 5.

C. Detailed Interconnects

To support the detailed interconnects in VRB, we enhance the architecture description file format in VTR as shown in Fig. 5. Fig. 6 shows the corresponding detailed architecture of Fig. 5. The *<muxes>* tag is added into the top level tags and it is composed of several *<mux>* tags which are used to describe the detailed interconnects for the muxes. The *name* attribute is a unique symbol for each mux. For the driving muxes of wire segments and LB input pins, additional attributes are added to recognize them as shown in Fig. 5. Attributes *to_seg_name* and *to_track* which correspond to the type name of this wire segment and the direction, index of this wire segment respectively are used to describe the driving muxes of wire segments. Attribute *to_pin* which corresponds to the name and index of the LB input pin is used to describe the driving muxes of LB input pins. The *<from>* tags are used to describe the information of mux fanins. The types of mux fanin include three options: mux, seg and pb which corresponding to other mux output, wire segment and LB output feedback respectively. The attribute $from\_details$ represents the detailed information of the fanins.

D. Nearest Neighbor Interconnects

Nearest Neighbor (NN) interconnects which provide fast connections between adjacent logic blocks are widely used in many commercial FPGAs. For example, the LAB's local interconnects can be driven by the left/right blocks through the direct link connections in Intel Stratix IV FPGAs [37]. The similar interconnects are also used in Xilinx FPGAs [4]. Fig. 7 shows an example of NN interconnect architecture in VRB. The $i0$ input pin of LB B can get connection from the $O1$ output pin of LB A without passing through any wire segment. To describe the NN interconnects, we add two attributes (x_offset and y_offset) in the *<from>* tag as shown in Fig. 7. Attributes x_offset and y_offset stand for the offsets of the two LBs with NN interconnects in x and y directions respectively. The default values of both x_offset and y_offset are 0 which means the connection from LB output pin is from the LB itself as shown in Fig. 5 and Fig. 6.




```
<mux name="mux-2" to_pin="i:0">
    <from type="pb" name="lb" from_details="o:1" x_offset="-1" y_offset="0"/>
    ...
</mux>
```

Fig. 7. An example of the NN interconnect architecture in VRB.

IV. Tool Enhancement

In this section, the tool enhancement is introduced to implement the VRB architecture modeling and exploration.

A. Enhanced RRG Generator

To support the VRB architecture, we enhance the RRG generator in VTR to model the FPGA routing resources. The routing resources are presented by a directed graph $G = (V, E)$. Vertices V correspond to the routing nodes which can be wire segments or LB pins, and edges E to the programmable switches between different vertices. For multi-level muxes, we use the similar modeling method in [8]. An intermediate node without timing cost is used to model the connection between the front level and the back level muxes. For the support of bent wires, one wire segment is divided into several parts according to its bent points and each part is represented as one vertex in the RRG. They are connected by non-configurable delayless switches which are treated as edges in the RRG.

V. Experimental Results

In this section, we describe the experimental methodology and the baseline architecture. Then, we use the enhanced VTR along with the provided benchmark set to evaluate the architecture in the area and delay.

TABLE I
Baseline Architecture Parameters

LB Size	Eight 6-input LUTs
DSP Elements	36 × 36 Fracturable Multipliers
Memories	32Kb Block RAMs
Routing channel width	160
Wire Length	4

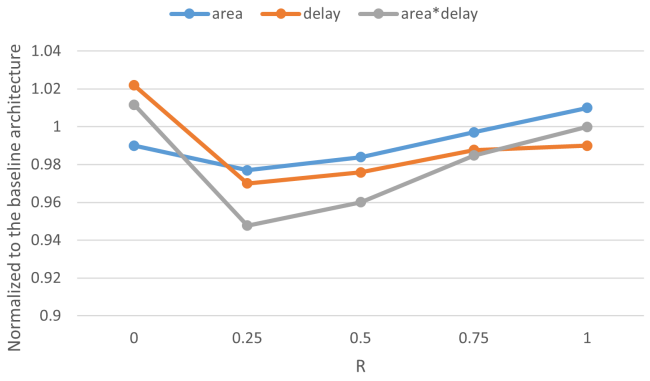


Fig. 8. VRB architectures with different R values.

A. Baseline Architecture

In this paper, we use the two-level mux architecture in [11] as the baseline architecture which is the latest work to explore the two-level mux topology. It has been proved that two-level

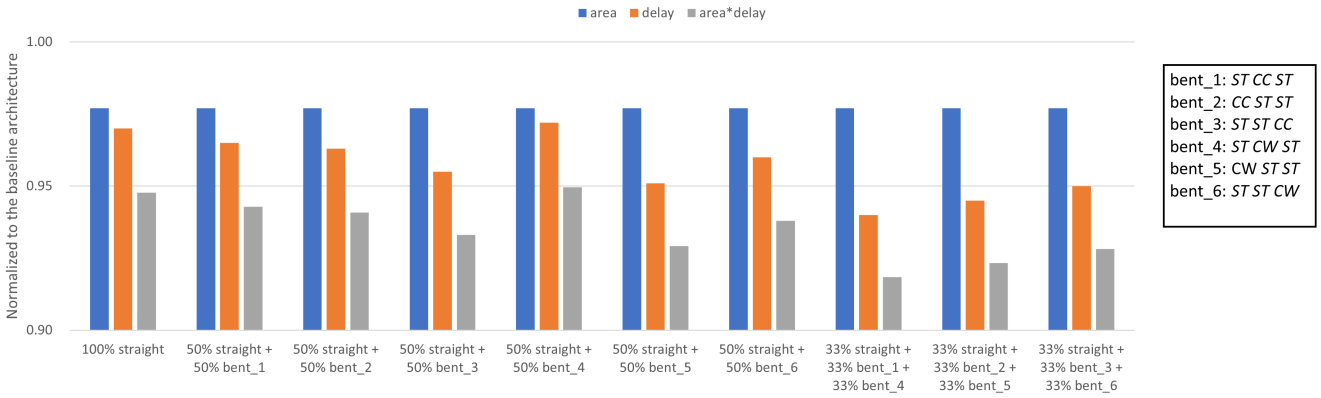


Fig. 9. VRB architectures with different bent wires.

mux topology can achieve better area-delay tradeoff compared to the one-level mux topology in CB-SB architecture. TABLE I shows the baseline architecture parameters. The delay and area parameters which are used in VTR architecture files are extracted from COFFE 2 [10] at the 22nm technology node. COFFE 2 is a fully-automated transistor sizing tool for FPGAs which relies on HSPICE simulation.

B. Architecture with Different features

In this section, we explore the VRB architecture with different features which are introduced in Section III.

Firstly, we explore the ratio of one-level mux and two-level mux connections in VRB architecture. Parameter R is defined to describe the ratio of fanins from wire segments and feedbacks directly in the driving muxes of wire segments and LB input muxes. For example, the value of R is 1 in CB-SB architecture as it is one-level mux topology and all fanins of the driving muxes of wire segments and LB input muxes are from wire segments and feedbacks directly. The value of R is 0 means that it is two-level mux topology and all fanins of the driving muxes of wire segments and LB input muxes are from level-1 muxes. Fig. 8 shows the results with different R values and the results are normalized to the two-level mux architecture in [11]. When the value of R is set to 0.25, the VRB architecture can achieve the best area and delay tradeoff and it can improve the critical path delay by 4.0% and area by 2.3%. One-level mux topology usually needs bigger mux size to keep the routability. Two-level mux topology leads to pass through two muxes to achieve connections. Experimental results show that mixed one-level and two-level mux topology is a better choice.

Then, bent wires are explored based on the VRB architecture with $R = 0.25$. We design several VRB architectures with different length-4 bent wires which is similar to [30]. The wire length is restricted to 4 which aims to eliminate the performance improvements due to different wire lengths. Experimental results show that VRB architecture with bent wires can achieve 6.0% improvement on the critical path delay and 2.3% improvement on area compared to the baseline architecture as shown in Fig. 9. There is no significant change in area as the wire length is kept fixed.

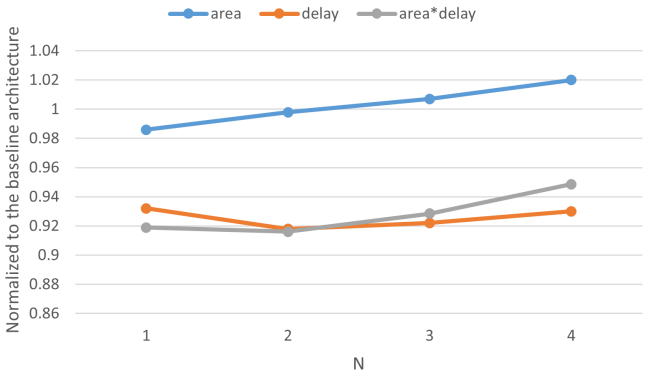


Fig. 10. VRB architectures with different N values.

Then, NN interconnects are explored based on the VRB architecture which can achieve the best area and delay tradeoff in Fig. 9. $Radius$ is defined to represent the distance of two LBs. Especially, the $Radius$ between two diagonal LBs is also defined as 1. Evidence shows that the net connections with a $Radius$ of 1 account for 28.7% of the whole net connections on average [30]. So, we focus on exploring the NN interconnects with $Radius = 1$. We use N to represent the number of NN interconnects for each LB output pin. Fig. 10 shows the results with different N values. Experimental results show that the VRB architecture can achieve the best area and delay tradeoff when N is set to 2 and it can achieve 8.2% improvement on the critical path delay and 8.4% improvement on area-delay product compared to the baseline architecture. Bigger N will lead to bigger routing area and larger mux size.

VI. Discussion and Conclusion

In this paper, we propose a tile-based VRB routing architecture for FPGAs and make it open publicly to narrow the gap between academic research and industry development. Several new features are extended in VTR 8. Based on the VRB

architecture, we explore the effects of different features on the FPGA area and delay. Reference [30] proposed a searching framework based on the SA algorithm to explore the FPGA routing architecture and reference [38] explored general routing architecture via Bayesian optimization algorithm. In the future, we will try to develop a similar automated architecture exploration tool to find the near-optimal architecture in area-delay tradeoff.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under grant 62174035.

REFERENCES

[1] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for deep-submicron FPGAs*. Kluwer Academic Publishers, 1999.

[2] K. E. Murray *et al.*, "VTR 8: High Performance CAD and Customizable FPGA Architecture Modelling," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 13, no. 2, pp. 1–55, 2020.

[3] P. B. Minev and V. S. Kukenska, "The Virtex-5 routing and logic architecture," *Annual Journal of Electronics, Technical University of Sofia*, vol. 3, pp. 107–110, 2009.

[4] M. B. Petersen, S. Nikolić, and M. Stojilović, "NetCracker: A peek into the routing architecture of Xilinx 7-Series FPGAs," in *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 11–22, 2021.

[5] K. E. Murray, M. A. Elgammal, V. Betz, T. Ansell, K. Rothman, and A. Comodi, "SymbiFlow and VPR: An open-source design flow for commercial and novel FPGAs," *IEEE Micro*, vol. 40, no. 4, pp. 49–57, 2020.

[6] K. Shi, H. Zhou, X. Zhou, and L. Wang, "GIB: A Novel Unidirectional Interconnection Architecture for FPGA," in *2020 International Conference on Field-Programmable Technology (ICFPT)*, pp. 174–181, 2020.

[7] C. Hu, Q. Duan, P. Lu, W. Liu, J. Wang, and J. Lai, "A Tile-based Interconnect Model for FPGA Architecture Exploration," in *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, pp. 113–118, 2020.

[8] J. Qian, Y. Shen, K. Shi, H. Zhou, and L. Wang, "General routing architecture modelling and exploration for modern FPGAs," in *2021 International Conference on Field-Programmable Technology (ICFPT)*, pp. 1–9, IEEE, 2021.

[9] K. Shi, H. Zhou, and L. Wang, "VIB: A Versatile Interconnection Block for FPGA Routing Architecture," in *2023 International Conference on Field-Programmable Technology (ICFPT)*, pp. 79–87, IEEE, 2023.

[10] S. Yazdanshenas and V. Betz, "COFFE 2: Automatic modelling and optimization of complex and heterogeneous FPGA architectures," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 12, no. 1, pp. 1–27, 2019.

[11] Y. Shen, J. Qian, K. Shi, L. Wang, and H. Zhou, "Two-level MUX Design and Exploration in FPGA Routing Architecture," in *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 234–241, IEEE, 2021.

[12] J. Mai, J. Wang, Z. Di, G. Luo, Y. Liang, and Y. Lin, "OpenPARF: An Open-Source Placement and Routing Framework for Large-Scale Heterogeneous FPGAs with Deep Learning Toolkit," in *2023 IEEE 15th International Conference on ASIC (ASICON)*, pp. 1–4, 2023.

[13] S. Chandrakar, D. Gaitonde, and T. Bauer, "Enhancements in UltraScale CLB architecture," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 108–116, 2015.

[14] T. Haroldsen, B. Nelson, and B. Hutchings, "RapidSmith 2: A framework for BEL-level CAD exploration on Xilinx FPGAs," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 66–69, 2015.

[15] P. Jamieson, K. B. Kent, F. Gharibian, and L. Shannon, "Odin II-an open-source verilog HDL synthesis tool for CAD research," in *2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, pp. 149–156, IEEE, 2010.

[16] A. Mishchenko *et al.*, "ABC: A system for sequential synthesis and verification," *URL: http://www.eecs.berkeley.edu/alanmi/abc*, 2007.

[17] J. Luu *et al.*, "VPR 5.0: FPGA CAD and architecture exploration tools with single-driver routing, heterogeneity and process scaling," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 4, no. 4, pp. 1–23, 2011.

[18] J. Luu *et al.*, "VTR 7.0: Next generation architecture and CAD system for FPGAs," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 7, no. 2, pp. 1–30, 2014.

[19] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deep-submicron FPGA performance and density," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 3, pp. 288–298, 2004.

[20] G. Zgheib and P. Ienne, "Evaluating FPGA clusters under wide ranges of design parameters," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 1–8, IEEE, 2017.

[21] G. Lemieux and D. Lewis, "Using sparse crossbars within LUT," in *Proceedings of the 2001 ACM/SIGDA ninth international symposium on Field programmable gate arrays*, pp. 59–68, 2001.

[22] A. Boutros, S. Yazdanshenas, and V. Betz, "Embracing Diversity: Enhanced DSP Blocks for Low-Precision Deep Learning on FPGAs," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 35–357, 2018.

[23] G. G. Lemieux, S. D. Brown, and D. Vranesic, "On two-step routing for FPGAs," in *Proceedings of the 1997 International Symposium on Physical Design*, pp. 60–66, 1997.

[24] G.-M. Wu, M. Shyu, and Y.-W. Chang, "Universal switch blocks for three-dimensional FPGA design," *IEE Proceedings-Circuits, Devices and Systems*, vol. 151, no. 1, pp. 49–57, 2004.

[25] S. J. Wilton *et al.*, *Architectures and algorithms for field-programmable gate arrays with embedded memory*. PhD thesis, University of Toronto, 1997.

[26] J. Rose and S. Brown, "Flexibility of interconnection structures for field-programmable gate arrays," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 3, pp. 277–282, 1991.

[27] V. Betz and J. Rose, "Fpga routing architecture: Segmentation and buffering to optimize speed and density," in *Proceedings of the ACM/SIGDA seventh international symposium on Field programmable gate arrays*, pp. 59–68, 1999.

[28] M. Lin, J. Wawrzynek, and A. El Gamal, "Exploring FPGA routing architecture stochastically," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 29, no. 10, pp. 1509–1522, 2010.

[29] X. Sun, H. Zhou, and L. Wang, "Bent Routing Pattern for FPGA," in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 9–16, 2019.

[30] K. Shi, X. Zhou, H. Zhou, and L. Wang, "An Optimized GIB Routing Architecture with Bent Wires for FPGA," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 16, no. 1, pp. 1–28, 2022.

[31] K. Shi, H. Zhou, and L. Wang, "A hexagon-based honeycomb routing architecture for FPGA," in *2021 International Conference on Field-Programmable Technology (ICFPT)*, pp. 1–6, IEEE, 2021.

[32] X. Tang, E. Giacomin, A. Alacchi, and P.-E. Gaillardon, "A study on switch block patterns for tileable FPGA routing architectures," in *2019 International Conference on Field-Programmable Technology (ICFPT)*, pp. 247–250, IEEE, 2019.

[33] J. Chromczak *et al.*, "Architectural Enhancements in Intel® Agilex™ FPGAs," in *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 140–149, 2020.

[34] K. Shi, H. Zhou, and L. Wang, "Explore the Feedback Interconnects in Intra-Cluster Routing for FPGAs," in *2023 International Conference on Field Programmable Technology (ICFPT)*, pp. 250–253, IEEE, 2023.

[35] D. Lewis *et al.*, "The Stratix™ 10 highly pipelined FPGA architecture," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 159–168, 2016.

[36] K. T. Khoozani, A. A. Dehkordi, and V. Betz, "Titan 2.0: Enabling Open-Source CAD Evaluation with a Modern Architecture Capture," in *2023 33rd International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 57–64, IEEE, 2023.

[37] D. Lewis *et al.*, "Architectural enhancements in Stratix-III™ and Stratix-IV™," in *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 33–42, 2009.

[38] S. Zheng, J. Qian, H. Zhou, and L. Wang, "GRAEBO: FPGA General Routing Architecture Exploration via Bayesian Optimization," in *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 282–286, 2022.