VIB: A Versatile Interconnection Block for FPGA Routing Architecture

Kaichuang Shi, Hao Zhou, Lingli Wang*
State Key Laboratory of ASIC and System
Fudan University, Shanghai, China
*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). There are also input crossbars inside the logic blocks (LBs). Muxes with high fanins are used to implement the intra- and inter-cluster connections. In the previous researches, CBs, SBs and input crossbars are designed separately which may miss some potential optimization spaces to trade off area, delay and routability. Besides, it is hard to model the complex routing architecture in commercial FPGAs. In this paper, we propose a novel tile-based routing architecture, versatile interconnection block (VIB) which replaces the CBs, SBs and input crossbars to alleviate this problem. All the routing resources are included in the VIBs which are based on two level mux topology used in many commercial FPGAs. Six parameters are proposed to describe the VIB architecture. In addition, VTR 8 is enhanced to support the proposed VIB architecture. Then, we compare the proposed architecture with the latest work of two level mux design. Experimental results show that the proposed VIB architecture can achieve 18.9% improvement on the routing area, 2% improvement on the routability and 16.6% improvement on the area-delay product with VTR benchmarks.**

***Index Terms*—FPGA, routing architecture, tileable, route, VTR enhancement**

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]. However, it needs more area, delay and power when the circuits are implemented in the FPGAs. Researches show that the routing architecture has a large impact on the FPGA area and delay [2]. Besides, wire delay has become a critical challenge with the process scaling [3]. Although wire distances shrink each generation, wire cross-sectional area shrinks quadratically which leads to an increase to wire delay.

The FPGA routing architecture includes the routing wires and the programmable switches among them. The most common routing architecture in academia is mainly based on the CBs and SBs which is also used in VTR 8 [4]. CBs are used to connect wire segments with logic block (LB) pins while SBs provide programmable switches to connect with different wire segments. Many researches have been done to optimize the routing architecture based on the CB-SB modeling, such as the distributions of wire segment lengths [5], the flexibility of CBs and SBs [1], architecture exploring methodology [6] and so on.

However, CB-SB architecture is too restricted to describe the complex routing architecture in commercial FPGAs. The detailed routing architecture is generated by several abstract parameters in VTR and only one level mux topology is supported. In addition, the architecture is not tileable [7] and two level mux topology [8] can not be realized in the VTR CB-SB routing architecture. F4PGA is an open source solution for HDL to bitstream FPGA synthesis targeting commercial FPGAs [9] [10]. But only Xilinx 7-Series FPGAs which have been over ten years are supported.

In this paper, we propose a tile-based VIB routing architecture to narrow the gap between the academic and commercial FPGAs and carry out a new architectural research. Our contributions include:

- We propose a tile-based VIB routing architecture where the basic tile is composed of an LB and a VIB. Tiles communicate with each other through routing wires. Two level mux topology is used in the VIB architecture.
- We define six parameters to explore the VIB architecture and the FPGA architecture description file format is extended to describe the complex routing architecture. In order to evaluate the performance of the VIB architecture, we enhance the Routing Resource Graph (RRG) generator and the packing algorithm in the latest VTR 8 [4].
- We evaluate the performance of the VIB architecture whose area and delay parameters are extracted from COFFE 2 [11] with VTR benchmarks [12]. Experimental results show that the VIB architecture can achieve 18.9% improvement on the routing area, 2% on the routability and 16.6% on the area-delay product compared to the two level mux architecture in [8].

The rest of this paper is organized as follows. Section II introduces the academic CB-SB routing architecture and the related work about FPGA routing architecture. Section III proposes the VIB architecture and introduces six parameters to describe the VIB architecture. Section IV gives the enhancements in VTR 8 and COFFE 2 to support the proposed VIB architecture. Section V presents the experimental methodology and experimental results compared with the two level mux architecture in [8]. Section VI concludes this paper with future work.

II. Background and Related Work

A. Routing Architecture of Island-Style FPGAs

Fig. 1 shows an example of island-style FPGA architecture which mainly contains LBs, CBs and SBs. And they are interconnected by vertical and horizontal routing wires. Fig. 2 is the detailed CB-SB routing architecture. The wire length L means the number of LBs that the wire segment spans. The routing channel width is described by W. The SB flexibility F_s is the number of other wires connected by an incoming wire inside the SBs. Input and output CB flexibility, $F_{c,in}$ & $F_{c,out}$ mean the fraction of wire segments in the routing channels that an LB input and output pin can connect to respectively.

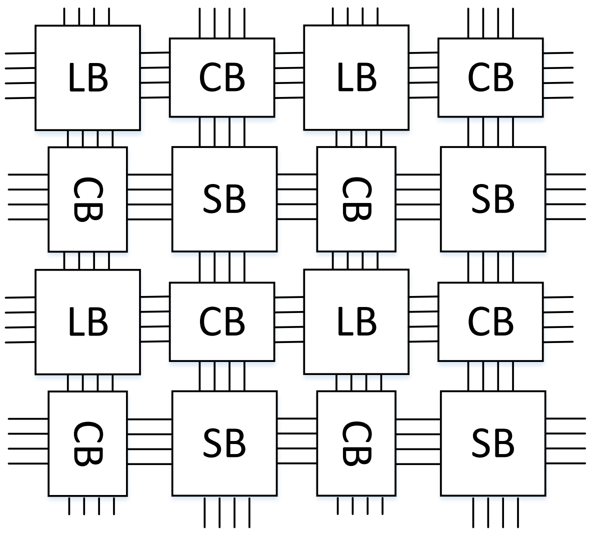


Fig. 1. Island-style FPGA architecture.

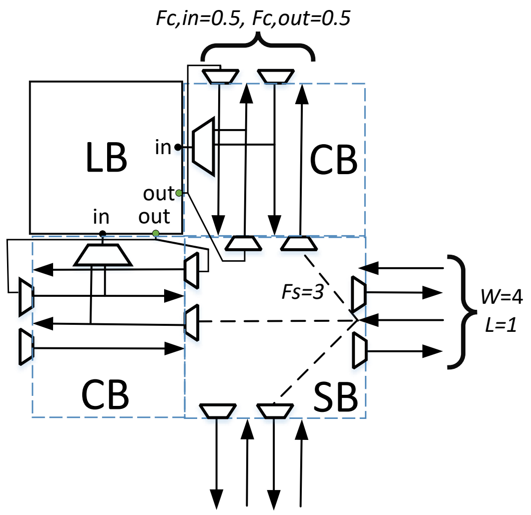


Fig. 2. The detailed CB-SB routing architecture.

B. Related Work

In the traditional CB-SB architecture, global routing architecture is composed of CBs and SBs. CBs are used to connect wire segments to LB pins and SBs are used to connect different wire segments. Besides, there are input crossbars in LBs to connect LB pins or feedbacks to LUT pins. Generally, these three components are designed separately in academic researches. For example, many SB patterns have been designed to improve the FPGA performance including Wilton [13], Universal [14], Disjoint [15] and so on. G. Lemieux et al. proposed an approach to generate the highly routable sparse crossbars in CBs [16]. In [17], G. Lemieux et al. also designed the sparse crossbars within LBs later. In addition, G. Zgheib et al. evaluated the effect of the local interconnect density in LBs on the FPGA performance and area [18].

W. Feng et al. proposed input interconnect blocks (IIBs) which route signals from wires and LB feedbacks to LUT inputs [19]. Experimental results show that IIB with two level muxes can achieve great area savings with no routability decreasing. K. Shi et al. proposed GIB architecture to replace traditional CB-SB architecture [20] [21]. However, the LB connections and the wire connections are designed separately, which are represented by *fc* and *fs* respectively. In addition, GIB architecture is not tileable. Tileable FPGA routing architecture has been proved that it can achieve better trade-off in area, delay and routability than non-tileable FPGAs [7]. Qian et al. proposed tile-based GRB architecture to model complex commercial FPGAs [22]. GRB is composed of three relatively independent modules: GSB, ICB and OCB, and the connections among them are restricted. In [8], Yu. Shen et al. explored two level mux topology in FPGA routing architecture, and experimental results show that it can achieve great delay improvement with small area overhead. Two level mux topology has been also used in the real commercial FPGAs. In Intel Agilex FPGA, The LB input pins can select the signals from LB output pins and wire segments through two level muxes which are called *LIM* and *LEIM* respectively [23]. In Xilinx FPGAs, two level muxes with different sizes are also widely used and some inputs can act as fast inputs from the global routing [24].

This paper is largely inspired by [7] [22] [8]. In this paper, we present a tile-based VIB routing architecture to replace the CBs, SBs and input crossbars. We design the two level mux (L1-mux and L2-mux) topology in VIB architecture to trade off the area and delay.

III. VIB Architecture

In this section, the VIB routing architecture is introduced. In addition, we design six parameters to describe the two level mux topology.

A. VIB Architecture

Fig. 3 shows the proposed VIB routing architecture which is tile-based. Each tile is composed of an LB and a VIB. Tiles communicate with each other through wire segments. There is no input crossbar inside the LB which is similar to the

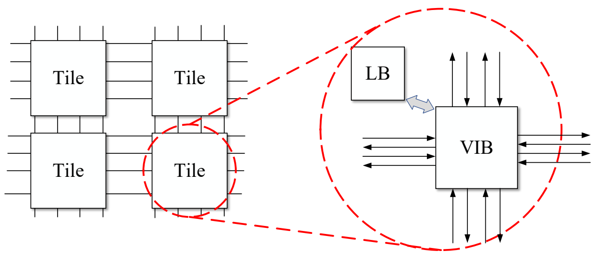


Fig. 3. The proposed VIB architecture.

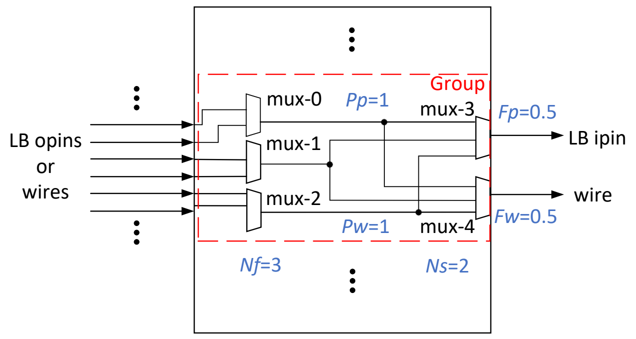


Fig. 4. An example of VIB architecture.

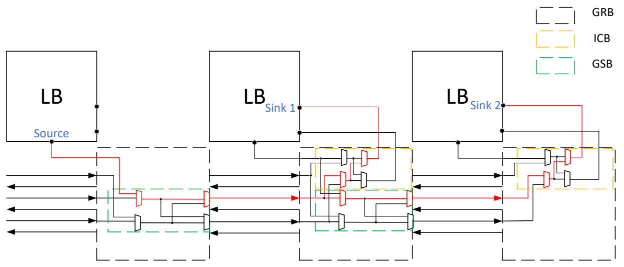
Xilinx FPGAs [25]. Inside the VIB, two level mux topology is implemented as shown in Fig. 4. For convenience, all nodes which go into the VIB are on the left side of the figure and all nodes which leave the VIB are on the right side. The VIB architecture is designed in groups which is described in detail in Section III-B. The LB input muxes and the driving muxes of wire segments can share the same fanins. For example, both mux-3 and mux-4 can achieve connections from mux-0, mux-1 and mux-2 as shown in Fig. 4.

In the previous papers about exploring the two level mux topology, the interconnections to the wires and the interconnections to the LB pins are designed separately. Such as IB and SB in [8], ICB and GSB in [22]. There is no communication between IBs (ICBs) and SBs (GSBs). It will lose some design spaces about saving routing area and it may lead to large FPGA routing area which has been proved [8]. Through the design method of two level mux topology in the VIB architecture, it can save FPGA routing area.

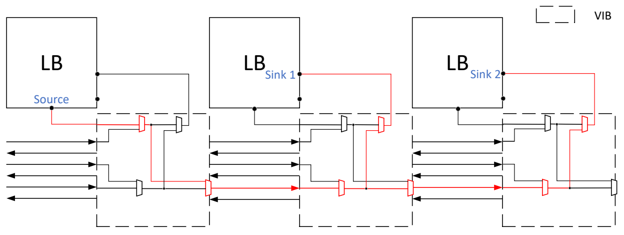
TABLE I shows the ratio of source-sink pairs in multi-sink nets to the total source-sink pairs in all nets. Experimental results show that about 57.77% source-sink pairs are in multi-sink nets. The implementation of multi-sink nets can achieve great area savings in the VIB architecture as the router try to re-use the wiring which has been used to connect the source and the other sinks [26]. Fig. 5 (a) shows a routing path of a two-sink net in the two level mux topology in [22]. ICB and GSB are designed separately and the two level muxes in ICB and the two level muxes in GSB have no communication. Fig. 5 (b) shows a routing path of the same net in the proposed

TABLE I
PERCENTAGE OF SOURCE-SINK PAIRS IN MULTI-SINK NETS

Circuits	source-sink pairs	source-sink pairs in multi-sink nets	Ratio
arm_core	27766	21228	76.45%
bgm	59131	50135	84.79%
blob_merge	13758	11777	85.60%
boundtop	268	38	14.18%
ch_intrinsics	238	68	28.57%
diffeq1	1181	710	60.12%
diffeq2	842	612	72.68%
LU32PEEng	202449	167305	82.64%
LU8PEEng	53851	44107	81.91%
mcml	92524	71229	76.98%
mkDelayWorker32B	1327	271	20.42%
mkPktMerge	1017	95	9.34%
mkSMAdapter4B	2259	1530	67.73%
or1200	6746	5310	78.71%
raygentop	2553	1396	54.68%
sha	4743	3986	84.04%
stereovision0	9545	4001	41.92%
stereovision1	14738	5878	39.88%
stereovision2	47550	24767	52.09%
stereovision3	145	62	42.76%
Average			**57.77%**



(a) The two level mux architecture in [22].



(b) The proposed VIB architecture.

Fig. 5. One routing path example to compare the two level mux architecture in [22] with the proposed VIB architecture.

VIB routing architecture. The LB input pins and wire segments can have the same fanins. It can be seen that no additional routing delay is added in the routing path. But the number of routing muxes is decreased which leads to less routing area. In addition, the routing resource utilization is also increased.

B. Modelling Parameters and The Enhanced Architecture Description File

To model the VIB architecture, six parameters are defined as shown in Fig. 6. We define N_f and N_s to represent the number of L1-muxes and L2-muxes in one group respectively. The L1-muxes can only connect to the L2-muxes in the same group through a crossbar which aims to achieve the regularity of detailed interconnections. We use P_p and P_w to represent the crossbar population density [27] of LB input muxes and wire driving muxes in L2-muxes respectively. F_p and F_w define the ratio of LB input muxes and the ratio of wire driving muxes to the total number of L2-muxes in one group (N_s) respectively. For example in Fig. 4, the value of N_f and N_s are 3 and 2 respectively. Both the values of P_p and P_w are 1 as both the LB input mux and wire driving mux can get connections from all the L1-muxes in the group. Both the values of F_p and F_w are 0.5 because half of the L2-muxes are LB input muxes and the other half are wire driving muxes in the group. The specific connections in L1-muxes follow a round-robin scheme [4] which tries to distribute the wire segments to the L1-muxes uniformly, based on the order of the segment index and direction. The number of L1-muxes and L2-muxes can be calculated by equation (1) and (2) respectively, where N_{lb_ipins} is the number of LB input pins in a tile and N_{wires} represents the number of wires in one direction in a tile. The handling of macro blocks is similar to that of LBs. The methods for generating detailed connections are the same. The difference of macro blocks and LBs is the number of output and input pins.

$$N_{L1-muxes} = N_f \times \left\lceil \frac{N_{lb\_ipins} + 4 \times N_{wires}}{N_s} \right\rceil \tag{1}$$

$$N_{L2-muxes} = N_{lb\_ipins} + 4 \times N_{wires} \tag{2}$$

In addition, we enhance the XML tags in the architecture description file to extend some detailed mux interconnections in additional to the interconnections generated by the six parameters as shown in Fig. 6. 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 interconnections for the muxes. The *name* attribute is a unique symbol for each mux. The *<from>* tags contain the fanins information of the mux. The *type* attribute has three optional values which stand for the output of other muxes, the output of LB and wire segment respectively as shown in Fig. 6. For the output of other muxes, the *name* attribute is the name of the mux. For the output of LB, the *name* attribute corresponds to the output pin of the LB. And for the wire segment, the *name* attribute contains the direction and the index of the wire segment. Besides, the additional *switchpoint* attribute means the switch point [28] of the wire segment (only for wire length $>$ 1). The switch point at the start of the wire is given an index of 0 and is incremented by 1 at each subsequent VIB. The last switch point has an index of 0 because it is shared between the end of the current segment and the start of the next one. With the enhanced architecture description file, arbitrary interconnect rules can be implemented.

```
<!--Several parameters to describe the VIB architecture-->
<VIB Nf="8" Ns="8" Pp="1" Pw="1" Fp="0.5" Fw="0.5"/>
<!--Detailed mux connections desciption-->
<muxes>
    <mux name="mux-1">
        <from type="mux" name="mux-0"/>
        <from type="lb" name="o[0]"/>
        <from type="L4" name="W[0]" switchpoint="0"/>
    </mux>
</muxes>
```

Fig. 6. The enhanced XML tags in the architecture description file.

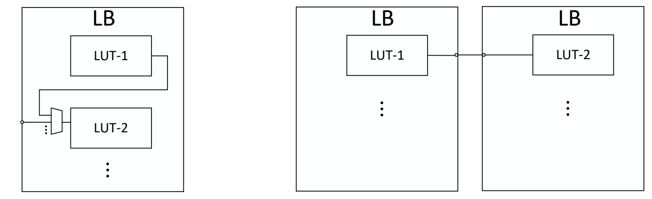
IV. Tool Enhancement

In this section, the tool enhancement is introduced to implement the VIB architecture modeling and exploring.

A. Enhanced Packer

VPR uses a packing algorithm which is based on the AAPack greedy algorithm [29]. One of the conditions of a legal packing solution is that all nets within a logical block can be routed. The netlist block packed into a logical block may have some connections with the netlist blocks which have been packed into the same logical block. If there is no intra-interconnects for such connections inside the logical block, the solution will be traded as an infeasible packing and the netlist block cannot be packed into the logical block. Fig. 7 is an example which shows the situation. The output of LUT-1 is also an input of LUT-2. In classical CB-SB FPGAs, the two LUTs can be packed into one LBs. However, there is no intra-connection available between the two LUTs in the proposed VIB architecture. So, LUT-1 and LUT-2 cannot be packed into the same logical block. The packer in VPR uses the PathFinder negotiated routing algorithm to determine whether a feasible routing solution can be found. Since we remove the local interconnects outside the logical blocks, the input pins of the logical block are connected with the LUT input pins directly. The packer needs to be enhanced to support the architecture [30] [31]. To illustrate the reason for enhancing the packing algorithm, we run VTR flow with two different LB architectures. The difference of the two LB architectures is that one has local interconnects with full crossbar, and the other has no local interconnect. TABLE II shows the numbers of LBs needed to implement the benchmark circuits. Experimental results show that it only needs 57.73% on the numbers of LBs after adding the local interconnects with full crossbar. The packing results of LBs without local interconnect are not efficient and the LB density is low.

To achieve the efficient packing results, we enhance the packer algorithm in VTR. A virtual full-connected crossbar is built inside the LB before the AAPack greedy algorithm is performed which is similar to [30]. It can simplify validating the routability of clusters. After the packing stage, the virtual full-connected crossbar is removed. And the signals of the CLB input pins are marked as the same as the input pins of internal LUTs. So the connections between CLB output pins



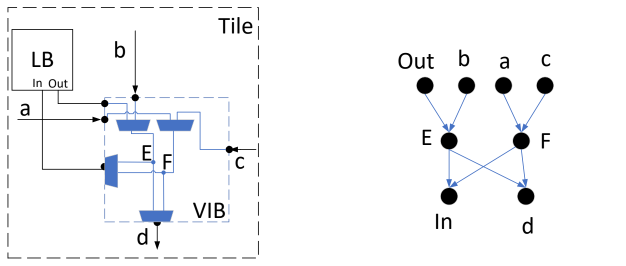
(a) LB with local interconnects. (b) LB without local interconnect.

Fig. 7. Comparison of the packing results with different LB architecture.

TABLE II
COMPARISON OF THE NUMBER OF LBS WITHOUT LOCAL INTERCONNECT AND THE NUMBER OF LBS WITH FULL CROSSBAR.

Circuits	# LBs without local interconnect	# LBs with full crossbar	Ratio
arm_core	3759	1587	42.22%
bgm	8873	3120	35.16%
blob_merge	2123	828	39.00%
boundtop	95	92	96.84%
ch_intrinsics	80	73	91.25%
diffeq1	195	63	32.31%
diffeq2	123	44	35.77%
LU32PEEng	7641	3261	42.68%
LU8PEEng	28050	11341	40.43%
mcml	17299	10471	60.53%
mkDelayWorker32B	552	547	99.09%
mkPktMerge	43	37	86.05%
mkSMAdapter4B	410	222	54.15%
or1200	941	372	39.53%
raygentop	305	205	67.21%
sha	1421	317	22.31%
stereovision0	1954	1632	83.52%
stereovision1	2252	1544	68.56%
stereovision2	7224	4367	60.45%
stereovision3	40	23	57.50%
Average			**57.73%**

[a] Ratio is the number of LBs with full crossbar divided by the number of LBs without local interconnect.



(a) The VIB architecture. (b) The corresponding RRG.

Fig. 8. RRG example.

and CLB input pins still need to be routed through global routing resources in the routing stage. The packing results become efficient and the LB density get improved. In order to get rid of the impact of software modification, we run the baseline architecture and the VIB architectures with the same packer in the following experiments.

B. Enhanced RRG Generator

To support the VIB 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 wires or LB pins, and edges E to the programmable switches. The enhanced RRG generator can automatically generate the routing nodes and edges according to the parameters described in Section III-B. For two level muxes, we use the similar modeling method in [22]. An intermediate node without timing cost is used to model the connection between the first level and the second level muxes. Fig. 8 shows an example of VIB architecture and the corresponding RRG. Node E and F are two intermediate nodes in Fig. 8.

C. Area and Delay Modelling

We use VTR to estimate the whole FPGA area and the critical path delay which measures the area in *minimum-width transistor areas* (MWTAs) [32] and uses Elmore delay model to estimate the delay. We enhance the COFFE 2 [11] to get the area and delay parameters needed by the VTR architecture description file. COFFE 2 is a fully automated transistor sizing tool for FPGAs which measures the area and delay by relying on HSPICE simulation. The input file is changed to an RRG file which contains all the routing nodes and programmable switches of one FPGA tile. The FPGA circuitry can be constructed by parsing the RRG file. Then, the area and delay parameters of each cell can be obtained by HSPICE simulation.

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 compare the two level mux architecture in [8] which is the latest paper to explore two level mux architecture with the VIB architecture in the area, delay and routability.

A. Experimental methodology

TABLE III shows the baseline architecture parameters. The delay and area parameters which are used in VTR architecture files are extracted from COFFE 2 at the 22nm technology node. Each LB contains eight 6-input LUTs which is similar to commercial FPGAs [25]. Unidirectional length-4 wires with full switchpoints are used which have been proved to achieve the best area-delay tradeoff [33]. The routing channel width is fixed at 160 which is reasonable in prior work [8]. The two level mux architecture in [8] is used as the baseline routing architecture which can achieve great delay improvement compared to CB-SB architecture. Finally, we will compare the VIB architecture with the two level mux architecture in the routability with W_{min}, the minimum routing channel width of each circuit.

TABLE III
BASELINE ARCHITECTURE PARAMETERS

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

B. VIB Architecture with Different Parameters

In this section, we explore the VIB architectures with different parameters which are introduced in Section III-B. The fanin size of L1-muxes is set to 5 to get the signals from wire segments in four directions and LB outputs which is the same as [8]. When one parameter is explored, the other parameters are kept fixed. Fig. 9 shows the results with different parameters and the results are normalized to the two level mux architecture in [8]. When the value of N_f is set to 8, the VIB architecture can achieve the best area and delay tradeoff as shown in Fig. 9 (a). N_f means the number of L1-muxes in a group. Larger N_f value leads to larger routing area while smaller value may result in worse delay. About the exploration of N_s, experimental results show that the VIB architecture can achieve the best area and delay tradeoff when N_s is set to 8 as shown in Fig. 9 (b). The area decreases and the delay increases as the N_s increases. Fig. 9 (c) shows the results of VIB architectures with different P_p and P_w. Smaller P_p and P_w values can decrease the routing area, but may lead to poor routability and long delay. Results show that the VIB architecture can achieve the best area and delay tradeoff when P_p = 1 and P_w = 0.75. Fig. 9 (d) shows the results of VIB architectures with different F_p and F_w. As the number of muxes is not correlated with F_p and F_w which can be seen in equation (1) and (2), the routing area keep constant. It can be seen that the VIB architecture can achieve the best area and delay tradeoff when F_p = 0.5 and F_w = 0.5. When N_f = 8, N_s = 8, P_p = 1, P_w = 0.75, F_p = 0.5 and F_w = 0.5, the VIB architecture can achieve the best area and delay tradeoff.

C. Area and Delay Comparison

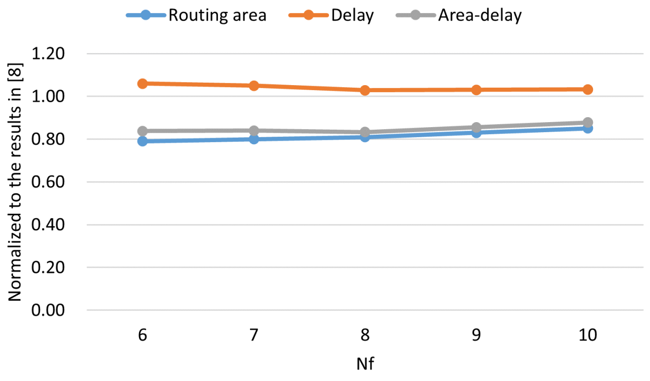
In this section, we compare the VIB architecture in Section V-B with the two level mux architecture in [8] in the area and delay. Experimental results show the VIB architecture can achieve 18.9% area savings at the cost of 2.8% increase in delay as shown in TABLE IV which is predictable. The L1-muxes can only connect to the L2-muxes of LB pins or the L2-muxes of wire segments in [8], but the L1-muxes can connect to the L2-muxes of LB pins and wire segments simultaneously which will increase the muxes loads and delay. In addition, the less muxes may lead to the routing congestion. Especially, the *mkDelayWorker32B* circuit has a 53.9% increase in critical path delay. We found that most of the nets in the *mkDelay-Worker32B* circuit are single-fanout nets as shown in TABLE I which can not take advantage of the VIB architecture. The less muxes in VIB architecture cause the routing congestion which leads to the longer critical path delay. The reduction of routing area is because the L2-muxes of LB pins and wire segments can share the output of L1-muxes which decreases in the number of L1-muxes. TABLE V lists the number of switches in one tile for the VIB architecture and the two level mux architecture in [8]. It can be seen that the VIB architecture has fewer L1-muxes and fewer total programmable switches. The channel utilization increases by 3% because the VIB architecture has fewer programmable switches which leads to more channel utilization.

D. Routability Comparison

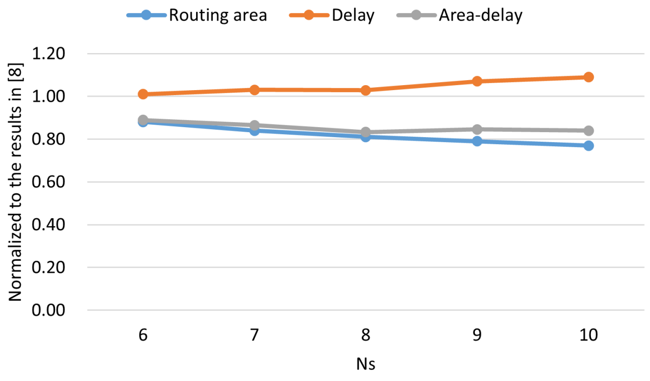
In this section, we compare the VIB architecture in Section V-B with the two level mux architecture in [8] in the routability. As the VIB architecture is tileable which has been described in Section III-A, the routing channel width can be calculated by equation (3), where n is the number of wire types, N_k represents the number of this wire type, L_k means the length of this wire type. Since only length-4 wires are used in the architecture, equation (3) can be simplified to equation (4). We use a binary search algorithm to find the minimum channel width which is similar to VPR. The router is run repeatedly to find the smallest number of tracks to route the circuit successfully. Experimental results show that the VIB architecture can improve W_{min} by 2% compared to the baseline two level mux architecture as shown in TABLE VI. Although there are fewer programmable connections in the VIB architecture, routability can be improved by increasing the interconnect utilization which is shown in Section V-C. The small improvement in routability of VIB architecture comes from the lack of fast LUT output feedbacks which connect LUT output pins with LUT input pins. All the LUT output feedbacks are connected to the L1-muxes and L2-muxes can only get connections from L1-muxes, no fast LUT output feedback. Fig. 10 shows the comparison of VIB architecture with and without fast LUT output feedback and the red line represents the fast LUT output feedback. This design approach of VIB architecture without fast LUT output feedback can improve the routability of the VIB architecture. However, it may lead to longer delay without fast LUT output feedback which has been prove in the Section V-C. In the future, fast LUT output feedback in VIB architecture will be explored to achieve better area-delay tradeoff.

$$W = 2 \times \sum_{k=1}^{n}(N_k \times L_k) \tag{3}$$

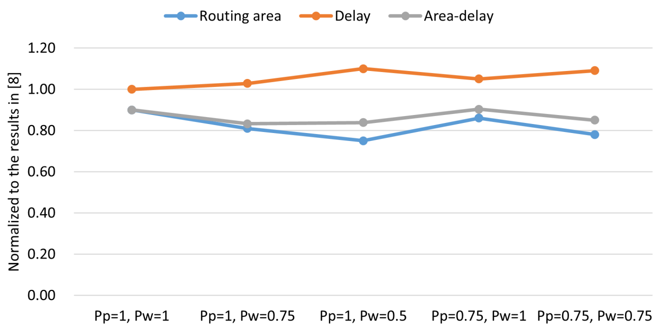
$$W = 8 \times N_k \tag{4}$$



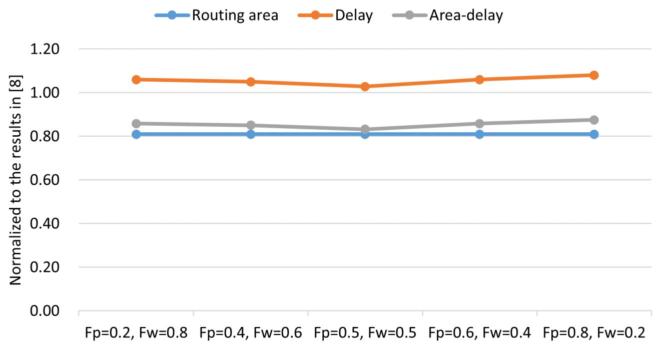
(a) Exploration of N_f.



(b) Exploration of N_s.



(c) Exploration of P_p and P_w.



(d) Exploration of F_p and F_w.

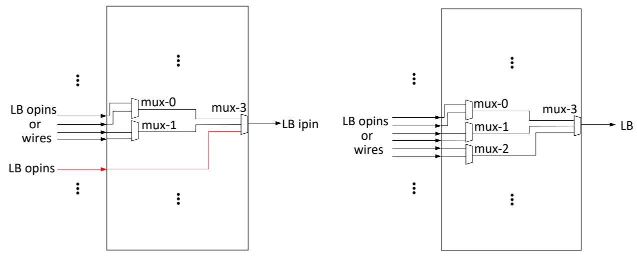
Fig. 9. Exploration of different parameters in VIB architecture.

TABLE IV
PERFORMANCE OF THE VIB ARCHITECTURE COMPARED WITH THE TWO LEVEL MUX ARCHITECTURE IN [8]

Circuit	Channel Utilization			Routing Area (10^6)			Critical Path Delay (ns)			Area-Delay (10^6)		
	VIB	[8]	Ratio	VIB	[8]	Ratio	VIB	[8]	Ratio	VIB	[8]	Ratio
arm_core	0.298	0.270	110.4%	37.88	46.69	81.1%	9.60	9.53	100.7%	363.65	444.96	81.7%
bgm	0.379	0.359	105.6%	47.45	58.47	81.1%	9.63	9.43	102.2%	457.06	551.11	82.9%
blob_merge	0.312	0.289	108.0%	13.27	16.36	81.1%	4.94	5.18	95.5%	65.60	84.66	77.5%
boundtop	0.014	0.014	100.7%	4.73	5.83	81.2%	1.09	1.25	87.5%	5.17	7.28	71.1%
ch_intrinsics	0.021	0.021	98.1%	2.74	3.37	81.2%	1.63	1.71	95.4%	4.46	5.75	77.5%
diffeq1	0.098	0.098	99.5%	3.47	4.27	81.2%	16.88	16.82	100.4%	58.52	71.85	81.5%
diffeq2	0.079	0.079	100.0%	3.08	3.80	81.2%	12.49	12.55	99.5%	38.52	47.70	80.8%
LU8PEEng	0.341	0.331	103.0%	170.72	210.41	81.1%	45.59	46.31	98.4%	7782.73	9744.72	79.9%
LU32PEEng	0.453	0.416	108.9%	50.36	62.07	81.1%	43.79	43.90	99.7%	2205.28	2724.91	80.9%
mcml	0.245	0.237	103.4%	157.30	193.87	81.1%	41.75	40.63	102.7%	6567.09	7877.60	83.4%
mkDelayWorker32B	0.015	0.013	120.0%	50.36	62.07	81.1%	8.65	5.62	153.9%	435.77	348.96	124.9%
mkPktMerge	0.031	0.032	99.0%	14.82	18.27	81.1%	3.66	3.51	104.4%	54.32	64.11	84.7%
mkSMAdapter4B	0.086	0.085	101.3%	7.29	8.98	81.2%	3.45	3.37	102.3%	25.12	30.26	83.0%
or1200	0.095	0.091	104.2%	27.16	33.48	81.1%	8.51	7.97	106.8%	231.27	266.82	86.7%
raygentop	0.075	0.074	101.5%	12.51	15.42	81.1%	3.88	3.88	100.0%	48.49	59.77	81.1%
sha	0.216	0.209	103.3%	5.69	7.01	81.2%	6.96	7.12	97.8%	39.63	49.93	79.4%
stereovision0	0.141	0.133	106.0%	26.08	32.15	81.1%	2.01	1.90	106.2%	52.54	61.00	86.1%
stereovision1	0.228	0.226	100.9%	25.01	30.83	81.1%	4.19	4.03	103.9%	104.72	124.29	84.3%
stereovision2	0.359	0.375	95.7%	73.33	90.38	81.1%	9.79	10.60	92.4%	718.26	958.01	75.0%
stereovision3	0.052	0.058	90.1%	0.66	0.82	81.2%	1.46	1.38	105.6%	0.97	1.13	85.8%
Av. Improvement		**-3.0%**			**18.9%**			**-2.8%**			**16.6%**	

TABLE V
THE NUMBER AND SIZE OF ROUTING MUXES IN ONE TILE OF THE VIB ARCHITECTURE AND THE ARCHITECTURE IN [8]

Architecture	MUX Name	Num	Size	Total Switch Count
[8]	IB L1-MUX	80	5	1820
	IB L2-MUX	48	13/14	
	SB L1-MUX	100	4	
	SB L2-MUX	80	4/5	
VIB	VIB L1-MUX	128	5	1504
	VIB L2-MUX	128	6/8	



(a) The VIB architecture with fast LUT output feedbacks.

(b) The VIB architecture without fast LUT output feedbacks.

Fig. 10. Comparison of VIB architecture with and without fast LUT output feedback.

TABLE VI
COMPARISON OF MINIMUM ROUTING CHANNEL WIDTH BETWEEN THE VIB ARCHITECTURE AND THE ARCHITECTURE IN [8]

Circuit	W_{min}		
	VIB	[8]	Ratio
arm_core	136	136	100.0%
bgm	96	96	100.0%
blob_merge	72	72	100.0%
boundtop	16	16	100.0%
ch_intrinsics	24	24	100.0%
diffeq1	40	40	100.0%
diffeq2	24	32	75.0%
LU8PEEng	96	96	100.0%
LU32PEEng	144	144	100.0%
mcml	136	136	100.0%
mkDelayWorker32B	32	32	100.0%
mkPktMerge	32	32	100.0%
mkSMAdapter4B	40	40	100.0%
or1200	80	72	111.1%
raygentop	40	40	100.0%
sha	72	64	112.5%
stereovision0	40	48	83.3%
stereovision1	80	72	111.1%
stereovision2	104	104	100.0%
stereovision3	16	24	66.7%
Av. Improvement		**2.0%**	

[a] Ratio is the minimum routing channel width of VIB architecture divided by the architecture in [8].

VI. Discussion and Conclusion

In this paper, we propose a tile-based VIB routing architecture. Six parameters are defined to describe the VIB architecture. Besides, the packing algorithm and RRG generator in VTR are enhanced to support the VIB architecture. Experimental results show that the VIB architecture can achieve 18.9% improvement on the routing area, 2% on the routability and 16.6% on the area-delay product compared to the two level muxes architecture in [8]. The VIB architecture eliminates the problem of the area increasing in the two level mux architecture [8].

In the future, there are still many researches to be carried out about the VIB architecture. We will focus on the routing channel segmentation with different lengths on the VIB architecture and try to find a near-optimal routing architecture in an automated way like in [22]. Besides, the interconnection rules also have potential to reduce the delay which we will explore in the future. For example, mixed one level mux and two level mux topology can be explored to trade off area and delay which has been discussed in the Section V-D.

Acknowledgment

This work is supported by the National Natural Science Foundation of China under grant 61971143.

References

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

[2] I. Kuon, R. Tessier, and J. Rose, *FPGA architecture: Survey and challenges*. Now Publishers, 2008.

[3] B. Gaide, D. Gaitonde, C. Ravishankar, and T. Bauer, "Xilinx adaptive compute acceleration platform: Versal™ architecture," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 84–93, 2019.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] "F4PGA - the GCC of FPGAs." http://www.f4pga.org.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] G. Lemieux, P. Leventis, and D. Lewis, "Generating highly-routable sparse crossbars for PLDs," in *Proceedings of the 2000 ACM/SIGDA eighth international symposium on Field programmable gate arrays*, pp. 155–164, 2000.

[17] 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.

[18] 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.

[19] W. Feng and S. Kaptanoglu, "Designing efficient input interconnect blocks for LUT clusters using counting and entropy," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 1, no. 1, pp. 1–28, 2008.

[20] 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.

[21] 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.

[22] 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.

[23] J. Chromczak, M. Wheeler, C. Chiasson, D. How, M. Langhammer, T. Vanderhoek, G. Zgheib, and I. Ganusov, "Architectural Enhancements in Intel® Agilex™ FPGAs," in *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 140–149, 2020.

[24] 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.

[25] 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.

[26] K. E. Murray, S. Zhong, and V. Betz, "AIR: A Fast but Lazy Timing-Driven FPGA Router," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 338–344, IEEE, 2020.

[27] Y. O. M. Moctar, G. G. Lemieux, and P. Brisk, "Fast and memory-efficient routing algorithms for field programmable gate arrays with sparse intracluster routing crossbars," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 12, pp. 1928–1941, 2015.

[28] VTR Developers, "Verilog-to-routing documentation." https://docs.verilogtorouting.org.

[29] J. Luu, J. Rose, and J. Anderson, "Towards interconnect-adaptive packing for FPGAs," in *Proceedings of the 2014 ACM/SIGDA international symposium on Field-programmable gate arrays*, pp. 21–30, 2014.

[30] T. D. Haroldsen, *Academic packing for commercial FPGA architectures*. PhD thesis, Brigham Young University, 2017.

[31] T. Haroldsen, B. Nelson, and B. Hutchings, "Packing a modern Xilinx FPGA using RapidSmith," in *2016 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pp. 1–6, IEEE, 2016.

[32] C. Chiasson, *Optimization and modeling of FPGA circuitry in advanced process technology*. PhD thesis, University of Toronto, 2013.

[33] O. Petelin and V. Betz, "The speed of diversity: Exploring complex FPGA routing topologies for the global metal layer," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 1–10, 2016.