GIB: A Novel Unidirectional Interconnection Architecture for FPGA

Kaichuang Shi, Hao Zhou, Xuegong Zhou*, Lingli Wang
State Key Laboratory of ASIC and System
Fudan University, Shanghai, China
*zhouxg@fudan.edu.cn

***Abstract*—Field Programmable Gate Arrays (FPGAs) are widely used because of the superiority in flexibility and low non-recurring engineering cost. The routing architecture has a large impact on the FPGA area, delay and routability. Hence, it is important to optimize the routing architecture. In academia, the routing architecture is mainly based on the connection blocks (CBs) and switch blocks (SBs), while most researches have focused on the SB architectures, such as Wilton, Universal and Disjoint SB patterns. In this paper, we propose a novel unidirectional routing architecture, general interconnection block (GIB) to improve the routability and performance. With GIB architecture, logic block (LB) pins can directly connect with the adjacent GIBs without programmable switches. Inside a GIB, LB pins and wire segments can connect with each other flexibly. LB pins can connect to the routing channel tracks on the four sides of a GIB. In particular, the logic pins from different neighboring LBs that connect to the same GIB can connect with each other with only one programmable switch which is impossible in the CB-SB architecture. We evaluate the GIB architecture on VTR 8 with the provided benchmark circuits. The experimental results show that the GIB architecture with all length-4 wires, which can offer the best area-delay tradeoff among the single wire types, achieves 8.3% improvement on the critical path delay and 9.9% improvement on the area-delay product on average compared to the VTR CB-SB architecture with the equivalent CB flexibility *(fc)* and SB flexibility *(fs)* values. In addition, it can achieve 9.5% improvement on the critical path delay and 11.1% improvement on the area-delay product after exploring different *fc* and *fs* values with all length-4 wires.**

I. INTRODUCTION

FPGAs are widely used due to low non-recurring engineering cost, fast time-to-market and their superiority in flexibility. However, the flexibility heavily relies on the programmable routing architectures, which consist of wire segments and programmable switches. The routing architecture has a great impact on the area, delay and routability [1][2]. In academic researches, the routing architecture is mainly based on the CBs and SBs [3][4], where CBs are used to connect LB pins with channel tracks while SBs are used to connect horizonal and vertical tracks. In the early years, most researches focused on the SB design to improve the performance and routability, such as Wilton [5], Universal [6] and Disjoint which is also known as Subset SB pattern [7]. References [8][9] propose CS-Box and GSB architectures respectively which rely on bidirectional wires to improve the FPGA performance instead of the CB-SB architecture. In this paper, we propose a novel unidirectional interconnection to explore the routing architecture efficiency. Our contributions include:

- We propose a novel unidirectional interconnection architecture, GIB. An LB can connect to four adjacent GIBs directly and a GIB has four adjacent LBs. Both LB input and output pins can directly connect to the adjacent GIB without programmable switches. Inside a GIB, LB pins and wire segments can connect with each other flexibly. An LB pin can connect to the routing channel tracks on the four sides of a GIB, while an LB pin can connect to one adjacent channel only in the CB-SB architecture. In addition, the pins from different neighboring LBs that connect to the same GIB can connect with each other with only one programmable switch which is impossible in the CB-SB architecture.
- In order to evaluate the performance of GIB architecture, we enhance the architecture description format and the Routing Resource Graph (RRG) generator in the latest VTR 8 [10]. We evaluate the performance of GIB architecture based on the area and delay parameters extracted from COFFE 2 [11] with VTR benchmarks [12]. Experimental results show that GIB architecture with all length-4 wires can improve the critical path delay by 8.3% and the area-delay product by 9.9% on average compared to CB-SB architecture with equivalent *fc* and *fs* values. After exploring different *fc* and *fs* values, GIB architecture can improve the critical path delay by 9.5% and achieve area-delay product savings by 11.1% on average. In addition, we evaluate GIB and CB-SB architectures with more single wire types of length-{2,3,4,6} respectively, results show that the GIB architecture with length-6 wires can achieve the most area-delay product improvement by 13.5% than CB-SB architecture. Besides, Wilton SB pattern can improve the critical path delay better than Subset and Universal SB patterns.

The rest of this paper is organized as follows. Section II introduces the academic routing architecture and the related work. The GIB architecture is proposed in Section III. Section IV presents the experimental results compared with the CB-SB architecture. Section V concludes this paper with the future work.

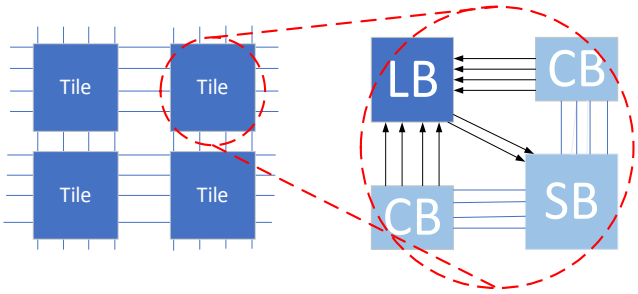


Fig. 1. Island-style FPGA architecture

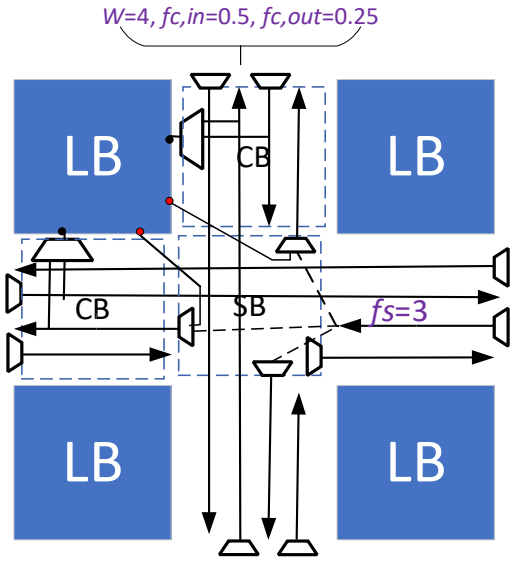


Fig. 2. The CB-SB routing architecture

II. Background and Related Work

A. FPGA routing architecture

The modern island-style FPGA is composed of an array of tiles which are connected with routing channels through programmable switches as show in Fig. 1. The routing channel width is described by *W*. Each tile consists of an LB (or memory, DSP block), two CBs and one SB [4], where CB flexibility and SB flexibility are described by *fc* and *fs* respectively. The value of *fc* defines the fraction of wires in routing channels that an LB pin can connect to, while *fs* defines the number of other wires to which an incoming wire can connect inside an SB [4] as shown in Fig. 2. These connections are all controlled by programmable switches which account for the delay and area. In modern FPGA, the programmable switches are isolating and configurable multiplexers [13]. Hence, reducing the number of programmable switches on the critical path can improve the FPGA performance.

B. VTR platform

In this paper, the platform to explore the GIB architecture is VTR [10], which is an open-source framework to conduct FPGA architecture and CAD algorithm research. The VTR flow needs two essential input files: a digital circuit described by Verilog and an FPGA architecture description file in XML format. There are three tools in VTR: Odin II [14], ABC [15] and VPR. Odin II is used for Verilog elaboration and the front-end hard-block synthesis. ABC is for logic optimization and technology mapping. VPR contains packing, placement, routing and timing analysis. VTR also provides medium-sized heterogeneous benchmarks which are suitable for FPGA architecture evaluation.

Currently VTR is based on CB-SB routing architecture as shown in Fig. 2. The input pins and output pins are represented by black and red circles respectively. The output pins can connect to the adjacent routing channel tracks through SB multiplexers directly. Horizonal and vertical tracks are connected with each other through SB multiplexers as well. The input pins can connect to the adjacent routing channel tracks through CB multiplexers.

C. Related work

The routing architecture has a great impact on FPGA routability and performance. Routing accounts for about 50% of the critical path delay [4]. Academic researchers have focused on the SB design to improve the routing architecture in the past several decades. There are several popular SB patterns such as Wilton, Universal and Disjoint. Reference [8] proposes Connection-Switch Box (CS-Box) where CB and SB are simply combined. In CS-Box, an LB pin can connect to the wire segments on the CS-Box sides except the side it belongs to. Results show that the number of programmable switches decrease by 11.81% at the cost of increasing the minimum channel width and the critical path delay. Reference [9] proposes a similar architecture, the general switch box (GSB), where an LB pin can connect to the wire segments on the four sides which can achieve better flexibility than CS-Box. The results show better improvement in delay with small reduction in routing switches. Both CS-Box and GSB rely on bidirectional wires with tristate drivers which are not efficient in modern FPGAs. Experimental results show the unidirectional wiring can achieve 32% area-delay product savings compared to the bidirectional wiring [13]. Reference [16] proposes a new switch box pattern for tileable FPGAs that achieves 12% improvement in the minimum routable channel width. Routing architecture is also well designed in commercial FPGAs, such as general routing matrix (GRM) in Xilinx Virtex-5 Family which provides an array of routing switches between each internal component [17]. Each programmable element is tied to one GRM to improve the performance.

There are several papers focusing on the wire segment patterns. To explore the routing architecture, references [18][19] propose a hard-wired routing pattern to reduce the number of programmable switches and achieves good improvement in delay. Recently, reference [20] proposes a bent routing pattern based on VTR to enhance the routing architecture which can achieve 11% area-delay product savings. In commercial FPGAs, Xilinx's Virtex-5 family implements a new diagonally-symmetrical architecture instead of the traditional wire segments [21]. A highly pipelined routing architecture is proposed in Intel's Agilex FPGAs to address the problem that the RC delay per physical distance increases as the process geometry shrinks [22].

This paper is largely inspired by [8][9]. LB pins and wire segments can connect with each other flexibly inside a GIB. To the best of knowledge, there is no academic paper applying similar ideas to the unidirectional routing architecture in modern FPGAs.

III. GIB Routing Architecture

In this section, we propose the GIB architecture as shown in Fig. 3, where an LB can connect to four adjacent GIBs

without programmable switches and a GIB has four adjacent LBs. Inside a GIB, LB pins and wire segments can connect with each other with significant flexibility improvement. The RRG generator in VTR is enhanced to support the GIB architecture.

A. GIB architecture

In GIB architecture, an LB pin can connect to the routing channel tracks on the four sides of a GIB while an LB pin can only connect to the one adjacent routing channel in CB-SB architecture. Besides, output pins and input pins can connect with each other inside a GIB with only one programmable switch. In CB-SB architecture, there are no such connections: output pins must connect to the wire segments through SB multiplexers firstly, then connect to input pins through CB multiplexers. These connections between output pins and input pins are called neighbor interconnects below. Comparison of CB-SB and GIB architecture is shown in Fig. 4. Wire segments are denoted by blue circles while input pins and output pins are denoted by black circles and red circles respectively. Neighbor interconnects are shown in red lines in Fig. 4 (b). It can be seen that GIB architecture can achieve much more routing flexibility than the CB-SB architecture. The wire segment connections between each other aren't shown because they have the same definitions in CB-SB architecture and GIB architecture.

The routing path generally starts from an LB output pin, and terminates at an LB input pin. In CB-SB architecture, an LB output pin passes through programmable switches in SBs, which eventually connects to LB input pins or IOs through CBs. With GIB architecture, LB pins can connect to the routing channel tracks on the four sides of a GIB through programmable switches and even connect to other LB pins with neighbor interconnects inside a GIB. Hence, GIB architecture can possibly reduce the number of programmable switches on the critical path. As shown in Fig. 5 (a), an LB output pin connects to a target LB input pin through two SB multiplexers and one CB multiplexer. With the GIB routing architecture as shown in Fig. 5 (b), one buffered multiplexer can be saved. Through reducing the number of the programmable switches on the critical path, delay can be reduced.

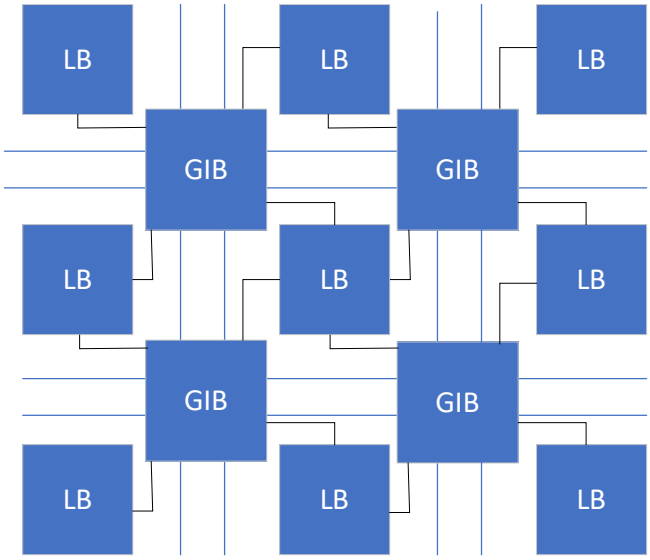
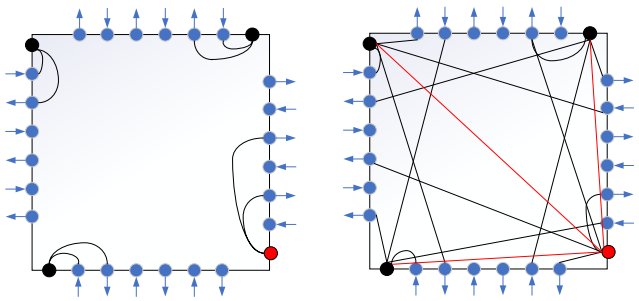
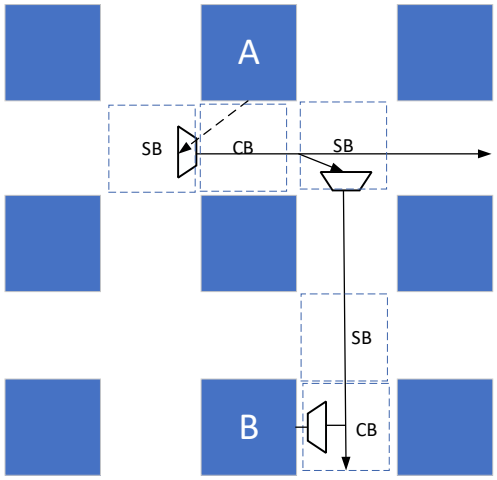


Fig. 3. GIB architecture

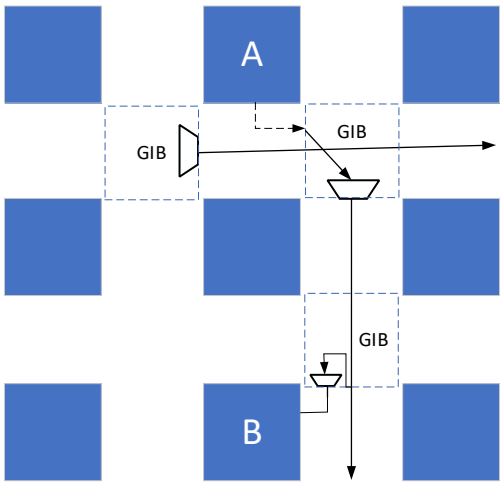


(a) (b)

Fig. 4. Comparison of CB-SB and GIB architecture, (a) CB-SB architecture and (b) GIB architecture



(a)



(b)

Fig. 5. The routing path compared CB-SB with GIB architecture, (a) CB-SB architecture and (b) GIB architecture

B. GIB architecture enhanced in VTR

To model GIB architecture in VTR [10], we enhance the FPGA architecture description format. The connections in GIB can be divided into three types:

1. The connections between LB pins and wire segments,

2. The connections between different wire segments,

3. The connections between LB output pins and LB input pins.

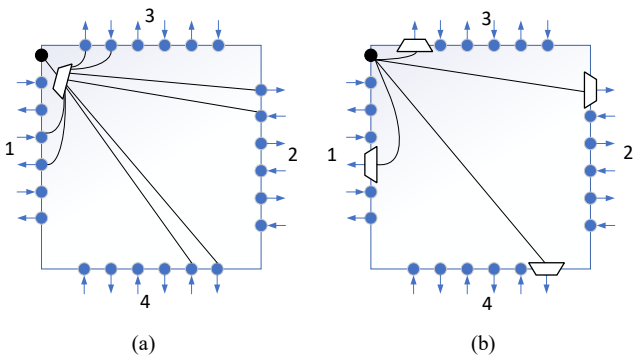
To describe the above GIB connections, three parameters are defined accordingly, *fc*, *fs* and *fn*: *fc* defines the fraction of wire segments in routing channels that an LB pin can connect to; *fs* defines the number of other wires that a wire can connect to; *fn* defines the number of input pins that an output pin can connect to through neighbor interconnects inside a GIB.

As shown in Fig. 6, we enhance the *fc* XML tag in the architecture description format. For every tile, we set four *fc* values for LB input pins and four *fc* values for LB output pins which correspond to four sides in a GIB. For an LB input pin, each of four *fc* values corresponds to the routing channel tracks that a pin can connect to one side of the GIB model. It can connect to different unidirectional wire pairs modelled by VTR as shown in Fig. 7 (a), where each pair corresponds to two wire segments with opposite directions. The 1st side is defined as the side that the pin belongs to. And the 2nd, 3rd and 4th sides correspond to the opposite side, the left side and the right side respectively as shown in Fig. 7. For LB output pins, they can only connect to those wires that go out of the GIB because unidirectional wires can only be driven in the start point as shown in Fig. 7 (b). So, the connected routing channel tracks are the half of *fc* values. For symmetry, the first and the second *fc* values have to be the same. So are the third and the fourth *fc* values. Supposing the value of *fc* is set to 0.1 for both input pins and output pins in CB-SB architecture, *fc* values from four sides should add up to 0.1 for input pins and 0.2 for output pins in GIB architecture to achieve the equivalent pins flexibility. The *fs* value defines the number of other wires that a wire can connect to inside the GIBs which is similar to the *fs* in CB-SB architecture. Each wire segment can connect to three other wire segments when the value of *fs* is set to 3. The value of *fn* is set to 3 which means an LB output pin can connect to three adjacent LB input pins inside a GIB through neighbor interconnects. It will be described in detail in Section III-C.

```
<!-- fc value is divided into four parts, which correspond to four different GIB sides respectively -->
<fc in_type="frac" in_val="fc_in1 fc_in2 fc_in3 fc_in4" out_type="frac" out_val="fc_out1 fc_out2 fc_out3 fc_out4"/>

<!--fn value defines the connections between output pins and input pins through neighbor interconnect -->
<neighbor_interconnect fn=3 />
```

Fig. 6. Example of enhanced XML tags for *fc* and *fn* values



(a) (b)

Fig. 7. An example of LB pins connections in GIB, (a) for input pins and (b) for output pins

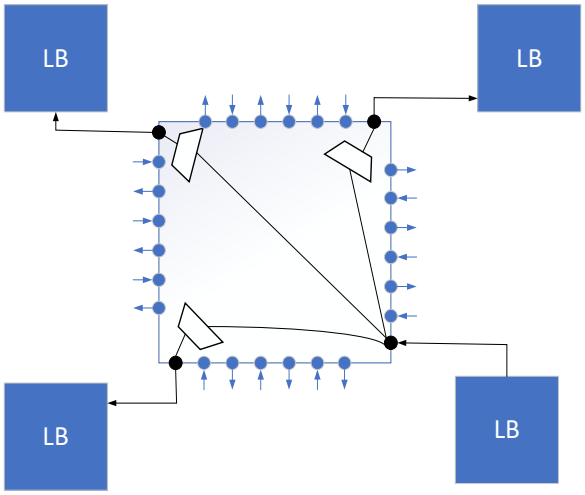


Fig. 8. Neighbor connects in GIB architecture

TABLE I. PERCENTAGE OF NET CONNECTIONS WITH RADIUS = 1

Circuit	Δx=1, Δy=0	Δx=0, Δy=1	Δx=1, Δy=1	Sum
arm_core	2.6%	3.5%	3.5%	9.7%
bgm	3.6%	5.3%	5.0%	13.9%
blob_merge	3.7%	5.4%	4.0%	13.2%
boundtop	19.4%	18.2%	29.8%	67.4%
ch_intrinsics	15.0%	22.5%	33.6%	71.1%
diffeq1	5.5%	12.0%	5.7%	23.2%
diffeq2	6.7%	13.9%	5.0%	25.7%
LU8PEEng	3.5%	5.1%	4.6%	13.1%
LU32PEEng	3.4%	4.8%	4.7%	13.0%
mcml	6.7%	9.7%	7.4%	23.8%
mkDelayWorker32B	11.1%	0.7%	6.0%	17.8%
mkPktMerge	5.9%	2.3%	9.8%	18.0%
mkSMAdapter4B	7.4%	9.0%	8.3%	24.7%
or1200	3.0%	3.7%	4.2%	10.9%
raygentop	8.4%	9.2%	6.6%	24.2%
sha	4.0%	7.1%	5.0%	16.1%
stereovision0	13.6%	19.6%	8.8%	42.0%
stereovision1	9.0%	12.8%	6.5%	28.3%
stereovision2	8.3%	11.0%	8.6%	27.9%
stereovision3	37.9%	33.7%	17.9%	89.5%
Average	**8.9%**	**10.5%**	**9.3%**	**28.7%**

TABLE II. PERCENTAGE OF INTER-CLUSTER DELAY WITH RADIUS = 1

(Δx, Δy)	Percentage
(1, 0)	2.8%
(0, 1)	4.2%
(1, 1)	4.4%
Sum	**11.4%**

C. Neighbor interconnect

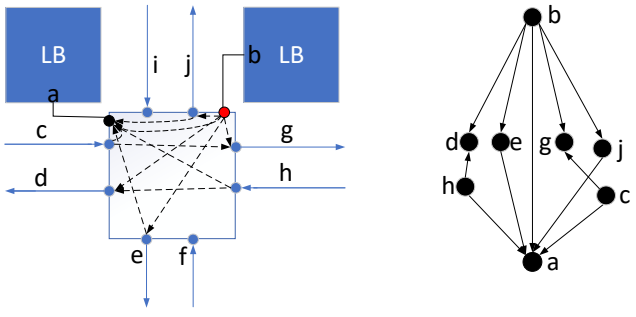
To improve GIB architecture, the neighbor interconnects [23] are added which can be described by *fn*. The radius parameter [23] is defined to represent the distance of two LBs that connect with each other through neighbor interconnects. Notice that the radius between two diagonal LBs is also defined as 1. For example, the radius between the LB in the

upper left corner and the LB in the bottom right corner in Fig. 8 is 1. To illustrate the advantage of neighbor interconnects, we run the VTR flow with the provided Stratix IV- like FPGA architecture and benchmarks. Then, the percentage of net connections with a radius of 1 is counted as shown in TABLE I where Δx and Δy stand for the distances between source node and sink node in the x and y directions respectively. Experimental results show that the net connections with a radius of 1 account for 28.7% of the whole net connections on average. These nets connections can be connected through neighbor interconnects in GIB architecture instead of wire segments. Besides, the percentage of the inter-cluster delay that the radius is 1 in the total critical path delay is counted as shown in TABLE II. Experimental results show that the inter-cluster delay that the radius is 1 account for 11.4% of the critical path delay on average. With neighbor interconnects, the critical path delay can be improved by reducing the programmable switch number.

We specify that an LB output pin can connect to three adjacent LB input pins which come from three LBs as shown in Fig. 8. These three input pins are located in different sides of the GIB and they are from different LBs. These pins are connected through buffered multiplexers. With these neighbor interconnects, an output pin can connect to three adjacent LB input pins with only one multiplexer without passing through wire segments. It can save two programmable switches compared with the CB-SB architecture. Hence, the critical path delay can be well reduced. However, too many neighbor interconnects will bring extra area cost and redundancy because LB input pins are logically equivalent with crossbar. Hence, we only set *fn* = 3 for an LB output pin.

D. RRG generator

To support the GIB architecture, we enhance the RRG generator in VTR which is modelled to describe the FPGA routing architecture. FPGA routing resources are presented by a directed graph *G= (V, E)*, where *V* denotes routing nodes which can be LB pins or wire segments, and *E* stands for the programmable connections between different nodes. And the router tries to find routing paths to implement the connectivity of the circuit while minimizing the wirelength and the critical path delay. For the CB-SB architecture supported by VTR, one pin can only connect to the adjacent channel tracks. For the GIB architecture, a pin can connect to different channel tracks from four sides in GIB and there are neighbor interconnects between LB input pins and output pins which can achieve better flexibility as shown in Fig. 9. In addition, it still supports different wire segments.



(a) (b)

Fig. 9. Routing Resource Graph, (a) GIB architecture, (b) the corresponding RRG.

E. CB-SB modeling in GIB

GIB architecture can be simplified as CB-SB architecture. When the first *fc* value is set to non-zero and the other three *fc* values are set to zero in the enhanced XML tags as shown in Fig. 6, the pin can only connect to one side which it belongs to as shown in Fig. 10. Besides, the *fn* value is set to 0 which means there is no neighbor interconnect. Under this parameter setting, as far as the routing architecture connections concerned, GIB and CB-SB are equivalent.

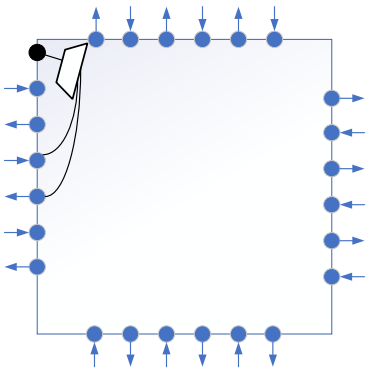


Fig. 10. CB-SB modelling in GIB

F. Area and delay modeling

VTR measures the whole area in *minimum-width transistor areas* [24]. One *minimum-width transistor area* is the area of the smallest possible contactable transistor plus the spacing to neighboring transistors for a specific process technology. The drive-strength of a transistor can be increased by either widening its diffusion region or by adding parallel diffusion regions. In other words, increasing the drive-strength of a transistor will increase its area. The delay is estimated with the Elmore delay model in VTR. The GIB architecture enhances the connections between LB pins and wire segments as well as the connections between LB pins. Hence, the default area and delay model in VTR are still suitable for GIB architecture.

IV. Experimental Results

In this section, we introduce the FPGA baseline architecture, and then compare GIB architecture with CB-SB architecture based on VTR with provided benchmarks.

A. Baseline architecture

In this paper, we use an island-based FPGA architecture whose area and delay parameters are extracted from COFFE 2 [11] at the 22 nm technology node which is the same as in [20]. COFFE 2 is a fully automated transistor sizing tool for FPGAs which measures delay and area by relying on HSPICE simulation. An LB is composed of ten 6-input fracturable LUTs and a local routing architecture with 50% connectivity. Memories are configurable 32K block RAMs which can operate in either single-port mode or dual-port mode. The memory has a configurable aspect ratio ranging from 32Kbits × 1 to 1K × 32 in dual-port mode and 32K × 1 to 512 × 64 in single-port mode. DSP modules are 36×36 fracturable multipliers which can operate as two 18×18 fracturable multipliers, and each 18×18 multiplier can be configured as two 9×9 multipliers. The IOs of this architecture are all on the perimeter and each IO contains 8 IO pins which can be configured to be input or output pins. Segments are all length-4 wires which can achieve the best area-delay tradeoff [25].

The *fc* value is set to 0.1 for input pins and output pins which can achieve good performance [20], and *fs* = 3. Experiments with single-driver routing architecture [13] have confirmed that *fs* = 3 is appropriate for this architecture. The SB pattern is Wilton, and the routing channel width is set to 300 which is reasonable in prior works [20][25].

B. GIB architecture with symmetry fc values

In this section, we compare GIB architecture with symmetry *fc* values with CB-SB architecture. At the same time, other parameters are fixed. We divide *fc* value in CB-SB into four equal values for four sides in GIB to achieve the same number of connections for pins. For example, *fc* = 0.1 in CB-SB architecture is equivalent to *fc* = (0.025 0.025 0.025 0.025) for input pins and *fc* = (0.05 0.05 0.05 0.05) for output pins in GIB architecture. Four groups of different *fc* values are chosen to evaluate GIB architecture respectively as shown in TABLE III. The experimental results show that GIB architecture can achieve about 8.3% improvement in the critical path delay and 9.9% improvement in area-delay product on average compared with CB-SB architecture as shown in TABLE III. Besides, there are small reduction in area (1.8%). One of the reasons for area reduction is that the IOs or LB pins in the perimeter of FPGA device can't connect to all four sides of GIBs. As shown in Fig. 11, any IO in the right boundary can only connect to three sides of the GIB, because there is no channel track on the right side of the GIB which costs less area. Another reason is that an input pin may connect to a wire segment repeatedly. For example, the value of *fc* is 0.68 for input pins in CB-SB architecture and the routing channel width is set to 6. That means an input pin can connect to 4 adjacent routing channel tracks since 6×0.68 = 4. Similarly, the value of *fc* is set to (0.17 0.17 0.17 0.17) for input pins in GIB architecture. That means that an input pin can connect to 4 routing channel tracks distributed in four sides of GIB since 6×0.17 + 6×0.17 + 6×0.17 + 6×0.17 = 4. However, the wire segments in opposite sides may be the same wire segment. As shown in Fig. 12, the input pin connects to one horizonal wire segment and two vertical wire segments. The size of multiplexer decreases because of the reduction in fan-ins which leads to the area reduction.

C. GIB architecture with asymmetry fc distribution

To find whether different *fc* value distributions affect FPGA performance, we set six groups of different *fc* values for GIB as shown in TABLE IV. With the constraints that the four *fc* values add up to 0.1 for input pins and 0.2 for output pins, we compare the results with CB-SB architecture. The *fc* distribution in group 1 is symmetry. As shown in Fig. 13, the experimental results show that the FPGA architecture with symmetry *fc* distribution can achieve the best improvement in the area-delay product.

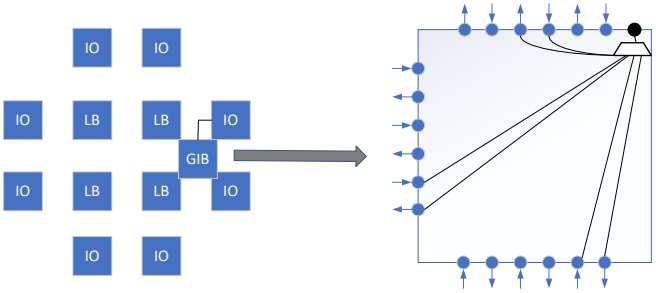


Fig. 11. An example of IO connects to GIB

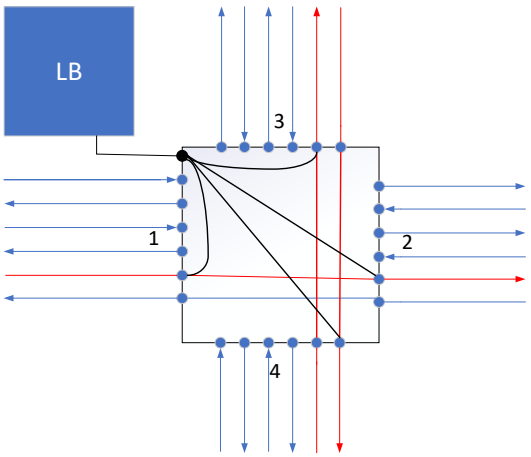


Fig. 12. Connections between input pin and wire segments

TABLE III. RESULT OF GIB COMPARED WITH CB-SB ARCHITECTURE WITH SYMMETRY *FC* VALUES

Fc	Area Ratio	Critical Path Delay Ratio	Area-Delay Ratio
0.1	98.2%	90.5%	88.9%
0.12	98.1%	92.7%	91.0%
0.16	98.0%	92.9%	91.0%
0.2	98.4%	90.8%	89.4%
Avg. improvement	**1.8%**	**8.3%**	**9.9%**

[a.] Ratios are the results that GIB are divided by CB-SB architecture.

TABLE IV. GROUPS OF GIB ARCHITECTURE WITH DIFFERENT *FC* DISTRIBUTIONS

Group	Fc, in_val	Fc, out_val
1	(0.025 0.025 0.025 0.025)	(0.05 0.05 0.05 0.05)
2	(0.04 0.01 0.04 0.01)	(0.05 0.05 0.05 0.05)
3	(0.03 0.02 0.025 0.025)	(0.08 0.08 0.02 0.02)
4	(0.025 0.025 0.025 0.025)	(0.07 0.07 0.03 0.03)
5	(0.03 0.02 0.025 0.025)	(0.05 0.05 0.05 0.05)
6	(0.03 0.02 0.03 0.02)	(0.05 0.05 0.05 0.05)

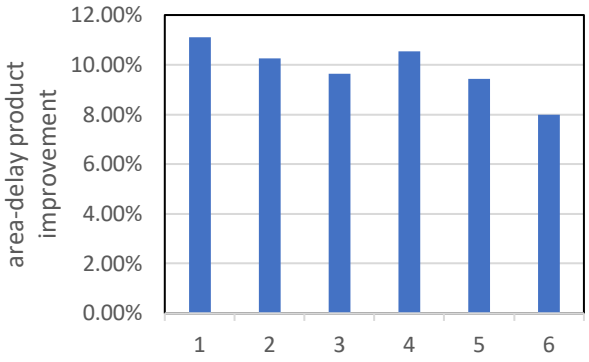


Fig. 13. The area-delay product improvement in GIB with different fc distributions compared to CB-SB architecture

D. GIB architecture with different fc values

In this section, we will explore more combinations of *fc* values to achieve better improvement in area-delay tradeoff. Because too large *fc* values will bring extra area cost, the upper bound of fc values is (0.05 0.05 0.05 0.05) for input pins and (0.10 0.10 0.10 0.10) for output pins. The experimental results are compared with the baseline FPGA architecture. As shown

TABLE V. COMPARISION OF GIB AND CB-SB BASELINE ARCHITECTURE

Fc, in_val	Fc, out_val	Area Ratio	Critical Path Delay Ratio	Area-Delay Ratio
(0.03 0.03 0.03 0.03)	(0.05 0.05 0.05 0.05)	99.3%	92.7%	92.1%
(0.025 0.025 0.025 0.025)	(0.04 0.04 0.04 0.04)	97.0%	92.6%	89.8%
(0.025 0.025 0.025 0.025)	(0.05 0.05 0.05 0.05)	98.2%	90.5%	88.9%
(0.025 0.025 0.025 0.025)	(0.06 0.06 0.06 0.06)	99.4%	91.5%	90.9%
(0.03 0.03 0.03 0.03)	(0.06 0.06 0.06 0.06)	100.5%	92.4%	92.8%
(0.04 0.04 0.04 0.04)	(0.05 0.05 0.05 0.05)	101.7%	89.8%	91.3%
(0.04 0.04 0.04 0.04)	(0.08 0.08 0.08 0.08)	105.3%	91.7%	96.6%
(0.05 0.05 0.05 0.05)	(0.10 0.10 0.10 0.10)	110.9%	90.6%	100.5%

TABLE VI. RESULT OF GIB COMPARED WITH CB-SB BASELINE ARCHITECTURE

Circuit	Total Area (10^6)			Critical Path Delay (*ns*)			Area-Delay (10^6)		
	GIB	CB-SB	Ratio	GIB	CB-SB	Ratio	GIB	CB-SB	Ratio
arm_core	52.65	53.58	98.3%	18.43	21.22	86.9%	970.33	1136.72	85.4%
bgm	107.15	109.08	98.2%	19.17	21.94	87.4%	2054.26	2392.73	85.9%
blob_merge	24.96	25.41	98.2%	9.86	11.28	87.4%	246.06	286.68	85.8%
boundtop	3.59	3.66	98.2%	2.01	2.30	87.6%	7.23	8.41	86.0%
ch_intrinsics	3.01	3.05	98.6%	2.38	2.57	92.8%	7.17	7.83	91.5%
diffeq1	4.51	4.59	98.3%	15.41	16.21	95.0%	69.478	74.36	93.4%
diffeq2	7.78	7.93	98.1%	13.09	13.52	96.8%	101.91	107.25	95.0%
LU8PEEng	83.02	84.53	98.2%	77.37	89.26	86.7%	6422.94	7544.76	85.1%
LU32PEEng	278.52	283.79	98.1%	77.02	87.19	88.3%	21451.51	24744.22	86.7%
mcml	239.76	244.16	98.2%	60.57	64.55	93.8%	14522.05	15760.76	92.1%
mkDelayWorker32B	68.89	70.17	98.2%	6.98	8.26	84.5%	480.58	579.56	82.9%
mkPktMerge	20.24	20.58	98.3%	4.29	4.00	107.1%	86.77	82.37	105.3%
mkSMAdapter4B	9.88	10.04	98.4%	5.38	6.47	83.1%	53.11	64.97	81.7%
or1200	18.90	19.25	98.2%	13.94	14.82	94.1%	263.51	285.37	92.3%
raygentop	12.11	12.32	98.3%	4.74	5.31	89.3%	57.40	65.39	87.8%
sha	8.72	8.88	98.1%	13.05	15.10	86.4%	113.78	134.15	84.8%
stereovision0	32.56	33.16	98.2%	4.38	4.73	92.6%	142.51	156.73	90.9%
stereovision1	38.91	39.60	98.3%	4.25	4.33	98.3%	165.38	171.32	96.5%
stereovision2	295.22	300.73	98.2%	16.26	18.05	90.1%	4800.99	5428.54	88.4%
stereovision3	0.75	0.76	98.1%	2.34	2.87	81.7%	1.75	2.19	80.1%
Average improvement	**1.8%**			**9.5%**			**11.1%**		

in TABLE V. When *fc* values are set to (0.025 0.025 0.025 0.025) for input pins and (0.05 0.05 0.05 0.05) for output pins, it can achieve the best improvement by 11.1% in area-delay product on average, particularly for those large benchmark circuits for which critical path delay is more than 20 *ns* such as *arm_core, bgm, LU8PEEng, LU32PEEng, mcml* as shown in TABLE VI. Results show that it can achieve 13% improvement in area-delay product on average. There are also several small circuits whose performances become worse as shown in TABLE VI. After analyzing the critical path of these circuits, we find that the placements are changed which affects the routing results. During the placement, the placer [26] in VTR will call for the router [27] to estimate the inter-cluster delay which relies on the routing architecture. After enhancing the RRG generator in VTR to support GIB architecture, the placer leads to different placements. To isolate the delay improvement without noise from changing placements, we evaluate the GIB architecture with locking down the original placements. Experimental results show that it can improve the critical path delay by approximately 9.5% and the area-delay product by 11.1% on average as shown in TABLE VII which is almost the same as the improvement without locking down the original placements. For these circuits whose performances become worse, there exist some regions getting congested which leads to longer critical path delay. In the future, we can explore the placement and routing algorithms in VTR to improve the performance of GIB architecture.

TABLE VII. RESULT OF GIB COMPARED WITH CB-SB BASELINE ARCHITECTURE WITH *LOCKING DOWN THE ORIGINAL PLACEMENTS*

	Area	Critical Path Delay	Area-Delay
Avg. improvement	**1.8%**	**9.5%**	**11.1%**

TABLE VIII. RESULT OF GIB COMPARED WITH CB-SB ARCHITECTURE WITH *WIRE SEGMENT* S OF DIFFERENT LENGTHS

Length	Area Improvement	Critical Path Delay Improvement	Area-Delay Improvement
2	0.7%	3.3%	4.0%
3	2.4%	6.6%	8.8%
4	1.8%	9.5%	11.1%
6	1.0%	12.6%	13.5%

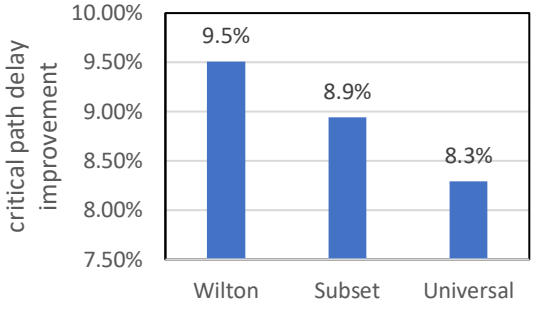


Fig. 14. The area-delay product improvement in GIB with different SB patterns

E. The effect of wire length and SB pattern

To explore the effect of wire length on GIB architecture, the single type of wire segments with different lengths are used. To obtain accurate and convincing experimental results, the timing and area parameters of wire segments and multiplexers are extracted from COFFE 2. Besides, different SB patterns are used to explore whether SB patterns influence the performance of GIB architecture. We use four kinds of wire segments with length- {2, 3, 4, 6}. Experimental results show that the longer wire segments can achieve more improvement as shown in TABLE VIII. The GIB architecture with length-6 wire segments can improve critical path delay by 12.6% and improve the area-delay product by 13.5% on average compared to CB-SB architecture with length-6 wire segments. Besides, we run the VTR flow with three different SB patterns where the value of *fs* is set to 3. Experimental results show that Wilton SB pattern can achieve the most improvement on critical path delay in GIB architecture as shown in Fig. 14. Especially, some circuits are unroutable which contain *arm_core, LU32PEEng, mcml, mkPktMerge* in Universal SB pattern which means Universal SB pattern is not always suitable for GIB architecture.

V. CONCLUSION

In this paper, we propose a novel unidirectional routing architecture for modern FPGAs. Compared with VTR CB-SB architecture, GIB architecture with the equivalent *fc* and *fs* values achieves 8.3% improvement on the critical path delay and 9.9% improvement on the area-delay product on average compared with CB-SB architecture. The optimized GIB architecture with *fc* = (0.025 0.025 0.025 0.025) for input pins and *fc* = (0.05 0.05 0.05 0.05) for output pins can improve the critical path delay by approximately 9.5% and the area-delay product by around 11% on average. Particularly for large circuits with long critical path delay, it can improve by 13% on the area-delay product. In future, we can run large circuits like Titan benchmarks [28] and explore different wire segments and bent wire pattern [20] to improve the GIB architecture further.

ACKNOWLEGEMENT

This work is supported in part by the National Science Foundation of China under Grant No. 61971143.

REFERENCES

[1] T. Karnik and S.-M. Kang, "An empirical model for accurate estimation of routing delay in FPGAs," in *Proceedings of the 1995 IEEE/ACM international conference on Computer-aided design.* IEEE Computer Society, 1995, pp. 328–331.

[2] M. Khellah et al., "Modelling routing delays in SRAM-based FPGAs," in *Canadian Conference* on VLSI. Citeseer, 1993, p. 6B.

[3] Eachempati, S. Nieuwoudt and A. Gayasen, "Assessing carbon nanotube bundle interconnect for future FPGA architectures," *Design, Automation & Test in Europe(DATE) Conference*, 2007, 23 -32

[4] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, 1999.

[5] S. Wilton et al., Architecture and algorithms for Field-Programmable Gate Arrays with embedded memory, PhD thesis, University of Toronto, 1997.

[6] 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.

[7] G.F. Lemieux, S.D. Brown and D. Vranesic, "On Two-step Routing for FPGAS," *Proceedings of the International Symposium on Physical Design (ISPD '97),* pp. 60-66, April 1997.

[8] C. Zhou , R. Cheung, and Y. Wu, "What if merging connection and switch boxes -- an experimental revisit on FPGA architectures," *IEEE International Conference on Communications, Circuits and Systems*, 2004, 1295 -1299

[9] K. Ma, L. Wang, X. Zhou, S. Tan and J. Tong, "General switch box modeling and optimization for FPGA routing architectures," *IEEE International Conference on Field-Programmable Technology (FPT)*, 2010, pp. 320–323.

[10] 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, p. 9, 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, p. 3, 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, p. 6, 2014.

[13] G. Lemieux et al., "Directional and single-driver wires in FPGA interconnect," *IEEE International Conference on Field-Programmable Technology (FPT)*, 2004, pp. 41–48.

[14] P. Jamieson, K. Kent, F. Gharibian, and L. Shannon. "Odin II-An Open-Source Verilog HDL Synthesis Tool for CAD Research," *In IEEE Annual Int'l Symp. on Field-Programmable Custom Computing Machines,* pp. 149–156. IEEE, 2010.

[15] A. Mishchenko et al. ABC: A System for Sequential Synthesis and Verification. *http://www.eecs.berkeley.edu/alanmi/abc,* 2009.

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

[17] Virtex-5 Family Overview, Xilinx, DS100 (v5.1) August 21, 2015

[18] S. Sivaswamy et al., "HARP: hard-wired routing pattern FPGAs," in *Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays.* ACM, 2005, pp. 21–29.

[19] G. Wang et al., "Statistical analysis and design of harp fpgas," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 2088–2102, 2006.

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

[21] 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.

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

[23] A. Roopchansingh and J. Rose, "Nearest neighbour interconnect architecture in deep submicron FPGAs," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, 2002, pp. 59–62.

[24] C. Chiasson, "Optimization and modeling of FPGA circuitry in advanced process technology," Master's thesis, University of Toronto, 2013.

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

[26] A. Marquardt, V. Betz, and J. Rose, "Timing-driven placement for FPGAs," in *Proceedings of the 2000 ACM/SIGDA 8th international symposium on Field-programmable gate arrays.* ACM, 2000, pp. 203–213.

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

[28] K. E. Murray, S. Whitty, S. Liu, J. Luu, and V. Betz, "Timing-driven Titan: Enabling large benchmarks and exploring the gap between academic and commercial CAD," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 8, no. 2 p. 10, 2015