General Routing Architecture Modelling and Exploration for Modern FPGAs

Jiadong Qian, Yuhang Shen, Kaichuang Shi, Hao Zhou*, Lingli Wang*
State Key Laboratory of ASIC and System.
Fudan University, Shanghai, China
*zhouhao@fudan.edu.cn, llwang@fudan.edu.cn

***Abstract*—Routing architecture has a significant impact on the area, critical path delay and power consumption of modern FPGAs. The most common routing architecture of island-style FPGAs in academia is the CB-SB model, which is not effective to model complex routing architectures in modern FPGAs. To improve the routability and performance of the existing routing model, we propose a new routing model called General Routing Block (GRB) to model complex commercial FPGAs. In the proposed model, all routing resources can be divided into three modules: general switch block (GSB), input connection block (ICB) and output connection block (OCB). The GSB and ICB are extended from the SB and CB with more flexible and richer connections. The OCB is a new module that provides novel connections for the LB output pins. We support bent wire architecture to reduce the delay, and two-level MUXes with output sharing to achieve a better trade-off between the area and flexibility. Moreover, to explore the trade-offs of different design spaces and find better architectures, an architecture exploration platform based on the simulated annealing algorithm is proposed to efficiently explore the enormous design space specified by a set of parameters. The results of global design space exploration show that the architecture with the proposed GRB model reduces the critical path delay by 15.5% and area-delay product by 14.8% compared to the length-4 CB-SB architecture based on the VTR benchmarks. After further local subspace explorations, the best architecture can achieve an 18.7% improvement on the critical path delay and a 23.8% improvement on the area-delay product, which represents a significant improvement over other routing architectures.**

I. INTRODUCTION

Field Programmable Gate Array (FPGA) is widely used in many applications, such as artificial intelligence, automotive, and communication, etc., because of its great flexibility, parallelism, low non-recurring engineering(NRE) cost, and fast time-to-market[1]. Studies have shown that routing architecture has a great influence on the FPGA area, delay, and power consumption [1][2]. The most common routing architecture of island-style FPGA in academia is the CB-SB model, which includes connection blocks (CBs), switch blocks (SBs) and routing channels. Based on the CB-SB model, there have been many studies about routing architectures, such as routing segments [3]-[5], routing patterns [6]-[9] and fast interconnections [10][11].

However, the routing architectures of modern commercial FPGAs are too complex to be modeled by the CB-SB architecture in VTR 8.0 [12]. Some researchers proposed different architectures, such as CS-Box [13], GSB [14], GIB [15], INTB [16] among others [17][18]. But most of them are still difficult to support commercial FPGA architectures. Symbiflow [19] solves this problem, but it only supports some old commercial FPGAs and we need to provide the RR-graph if we want to extend new commercial FPGA architectures in Symbiflow. To narrow the gap between academic research and industrial development, reference [20] developed an open-source tool to facilitate the analysis of commercial FPGA routing architectures and applied it to the 7-Series architecture family. According to reference [20], some architectures in commercial FPGAs, such as bent wires [9] and two-level MUXes with output sharing [21], have not yet been supported in a unified academic model. In addition, there have been few researches explored the trade-offs of these architectures. In this paper, we propose a unified FPGA routing model, GRB, to support these architectures and improve the routability and performance of existing routing models. Based on the GRB model, an exploration platform is proposed to explore the design space combined by these architectures and find the optimized routing architecture. The key contributions of our work are as follows:

(1) A routing architecture model called GRB is proposed to model modern FPGAs. All routing resources in a GRB can be divided into three modules: general switch block (GSB), input connection block (ICB) and output connection block (OCB). Bent patterns [9] and two-level MUXes with output sharing [21] are supported in the GRB model. To evaluate the performance of the proposed model, the FPGA architecture description file and routing resource graph (RRG) generator in the VTR 8.0 release [12] are extended. Two sets of parameters, coarse-grained and fine-grained descriptions, are proposed to explore the design space.

(2) An exploration platform inspired by TORCH [22][23] is developed to efficiently explore the enormous design space of FPGA routing architectures specified by a set of parameters. The platform which is based on the simulated annealing algorithm can search for an optimized combination of different segments and driving relationships in the architecture automatically. It will significantly reduce the time required to design the architectures manually. The area and delay parameters are determined by the transistor sizing tool COFFE 2 [24], which has been updated to support the GRB model.

(3) The area and delay of the best architecture reported by the platform are analyzed with the VTR benchmarks. The results show that the GRB architecture has an average improvement of 15.5% on the critical path delay and 14.8% on the area-delay product compared to the length-4 CB-SB architecture. Besides, in further subspace explorations, we search the segment distribution, bent pattern, and driving relationship separately based on the iterative optimization method. The contribution of the above subspaces to critical path delay is analyzed, and the best architecture can achieve 18.7% improvement on the critical path delay and 23.8% improvement on the area-delay product, which shows a significant improvement over other routing architectures.

In the rest of this paper, Section II introduces the common FPGA routing architecture and some related work. The details about the GRB model are shown in Section III. Section IV explains the architecture exploration platform and its implementation. Finally, the experimental results and conclusion are shown in Section V and Section VI.

II. Background And Related Work

A. FPGA Routing Architecture

The routing architecture of island-style FPGAs based on the CB-SB model is shown in Fig.1. The basic routing resources include wire segments and programmable switches. LBs are surrounded by wire segments on all four sides. W is the channel width which specifies the number of wire segments in one channel. The input or output pins of LBs can connect to some wire segments in the adjacent channel via CBs. The parameters $F_{c,in}$ *and* $F_{c,out}$ are the input and output CB flexibilities which specify the fraction of wires in a routing channel connected by an LB pin. SBs are placed in every intersection of a horizontal channel and a vertical channel. It includes some programmable switches which allow wire segments adjacent to the SB to be connected to others. The parameter F_s is the SB flexibilities which specifies the number of wires to which each incoming wire can connect.

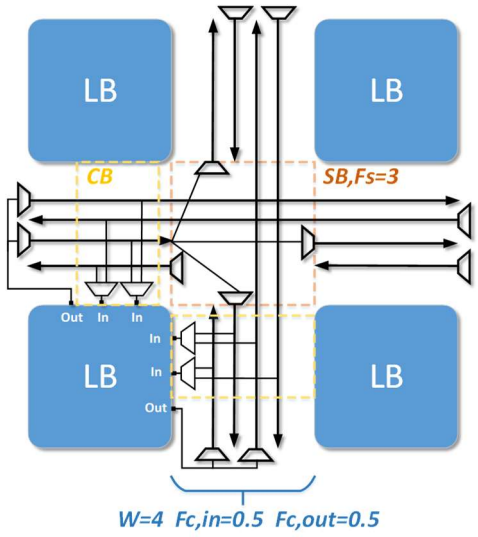


Fig.1 FPGA CB-SB architecture

B. Related Work

There are many studies about FPGA routing architectures. Reference [5] studies the influence of segment length distribution on the FPGA area and delay. The result shows the combination of length-4 wires and length-8 wires can achieve the optimal area-delay product. Reference [9] proposes the bent routing pattern, where the routing segments can span in both vertical and horizontal channels without passing through any turning switch. The result shows that bent routing topology achieves 9% shorter critical path delay and 11% area-delay product savings on average compared to the architecture with only straight wires. In addition to routing segments, there are also researches about neighbor interconnect, which is a fast connection between adjacent LBs that can achieve a lower delay. Reference [10] explores the topologies, quantities, and distances of neighbor interconnect and the best interconnect achieves a 6.4% improvement in the critical path delay. However, these researches are based on the CB-SB model, which is not effective to represent a large enough optimization space to be explored.

There are also some researches focusing on different routing architectures. Kejie Ma et al. [14] propose a GSB model used in bidirectional routing architecture. It has a larger exploration space compared to CS-Box [13] and is 24.3% better than the CB-SB model in terms of the product of the delay and channel width. Kaichuang Shi et al. [15] propose a GIB model in unidirectional routing architecture and achieves 8.3% improvement on the critical path delay and 9.9% improvement on the area-delay product on average compared to the CB-SB model. However, although these models have more flexible connections, they still have a similar topology as the CB-SB model. Reference [16] presents an INTB model with two-level local MUXes and curve wires to describe complex interconnect in modern FPGAs. But the architecture description is not flexible enough and they do not explore architecture. Reference [20] points out that there have been few researches explored the trade-offs in complex routing architectures, which has already existed in commercial FPGA. To promote the solution to this problem, we propose GRB to model and explore complex routing architectures. TABLE I shows the differences between the above architectures and our work, and other flexible connections mean more flexible and powerful interconnections than the CB-SB model.

TABLE I The differences between previous architectures and our work.

Arch. Feature	Bent Pattern [9]	GIB [15]	INTB [16]	Two-level MUXes [21]	Our Work
Bent wire	✓	×	✓	×	✓
Two-level MUXes with Output Sharing	×	×	✓	✓	✓
OCB	×	×	×	×	✓
Other Flexible Connections	×	✓	✓	×	✓

III. GRB Model

A. GRB Model Overview

The overview of the proposed GRB model is shown in Fig.2(a)&(b), which is based on tileable FPGAs. Inside each tile, there are an LB and a GRB which includes all routing resources in the tile. The LB is connected to the GRB module via the local interconnections while GRBs are connected by global wire segments. The input or output pins of LB are not equivalent. Both straight wires and bent wires [9], which can decrease the path delay for passing through fewer MUXes, are supported. Two-level MUXes with output sharing [20], as shown in Fig.2(d), which can provide more flexibility and reduce the load on the wire segments without increasing the area if designed properly [16], are also supported in the GRB module. Besides, the GRB support near local routing, which is a fast interconnection of neighboring tiles.

The GRB module can be further divided into the following three small modules which are composed of a large number of MUXes, as shown in Fig.2(c):

- ICB: It is used for the connections of the global segments to the LB input pins. It also includes partial feedback connections and neighboring connections [10]. The feedback connection means the OCB outputs and LB output pins of this tile can connect to the LB input pins through ICB. The neighboring connection means the OCB outputs of adjacent tiles can connect to LB input pins through ICB.
- OCB: It is used for the connection of the LB output pins to the global segments. Besides, the OCB outputs can also be used as feedback connection and neighbor connection.
- GSB: It contains networks of MUXes connecting its inputs to outputs, with the number of MUXes along the path limited to two. GSB is used for the connections between the global segments. Besides, It also includes the direct connections from the OCB outputs and LB output pins to the global segments.

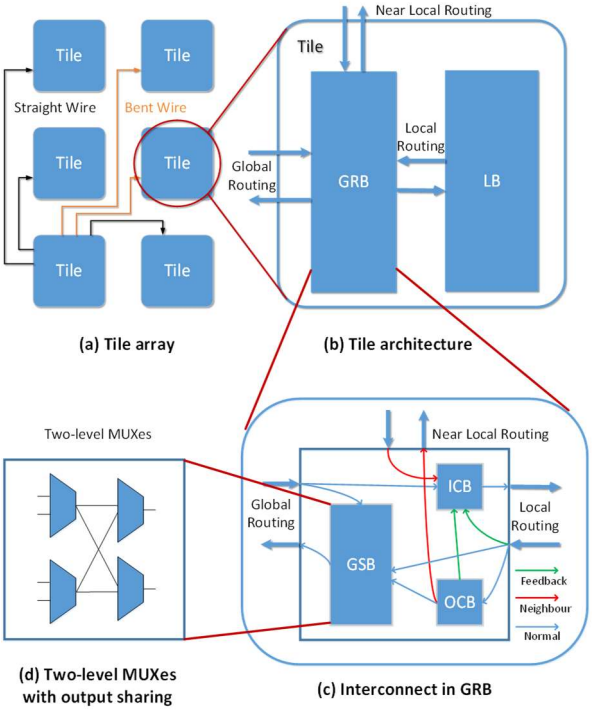
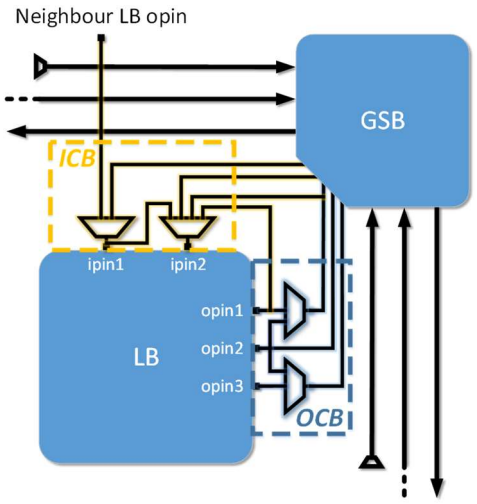


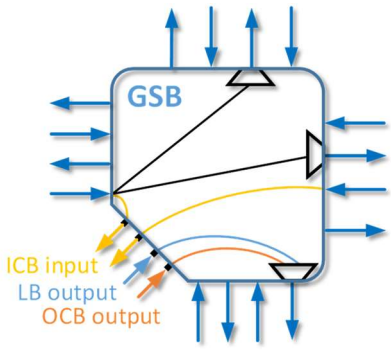
Fig.2 The proposed GRB model

The detailed routing architecture of the GRB model is shown in Fig.3. The ICB has similar functions as CB, but with a richer connection. The routing tracks on four sides of a GSB can connect to ICB in the GRB model while only one side of an LB can connect to CB in the CB-SB model. Besides, the driving source of ICB also includes OCB outputs in this tile, the output pins of LB in this tile or adjacent tiles, and ICB outputs in this tile. These connections greatly increase the flexibility of routing architecture, reduce the number of programmable switches on the paths and improve the routability. The OCB enables the LB output pins to be connected to the GSB either directly or via a MUX. It can connect more different LB output pins to global wire segments, which also provides more choices for routing. While the LB output pins can only connect to SB directly in the CB-SB model.

Fig.3(b) shows the detailed connections in GSB. The connections between wire segments are the same as the connections in SB. Besides, the wire segments on four sides are equal for this tile. The output pins of LB and OCB outputs can connect to segments on all four sides and segments can connect to ICB input directly. All these connections need at most one programmable switch while two are required in the CB-SB model if LB pins and wire segments are not on the same side of LB. So the GRB model can achieve a better performance in delay.



(a) Detail routing architecture of ICB/OCB



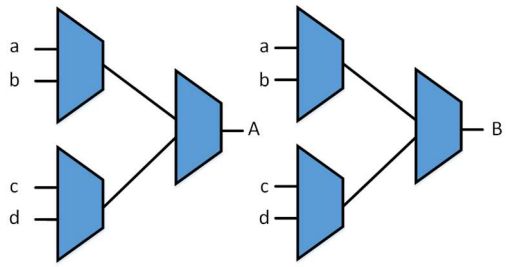
(b) Detailed routing architecture of GSB

Fig.3 Detailed routing architecture of GRB model

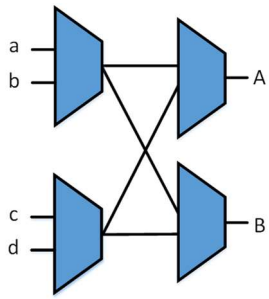
B. *Two-level MUXes*

In the GRB model, we present the two-level MUXes with output sharing [20], which means the outputs of the first-level MUXes are used as the inputs of multiple second-level MUXes. Fig.4 shows schematic diagrams of the two-level MUXes. The advantage of MUXes with output sharing is that it can reduce the area, which is very important in modern FPGA design. In Fig.4, outputs *A* and *B* both have four sources, but only four 2:1 MUXes are required while six 2:1 MUXes are required if there is no output sharing. In addition, it is mentioned in reference [16] that the two-level MUXes can be used to reduce the MUX load of the global segments, and in

reference that it can reduce the number of SRAM cells. In summary, the output sharing of MUXes results in a better trade-off between the area and architecture flexibility. Reasonably designed two-level MUXes can provide higher flexibility without increasing the area so that global segments could have more paths in GRB to swap or connect to more LB input pins.



(a) Two-level MUXes without output sharing



(b) Two-level MUXes with output sharing

Fig.4 Example of two-level MUXes

C. VTR Architecture File Enhancement

The architecture file in VTR 8.0 is extended to support the GRB model. We propose two sets of parameters with different granularity to describe the design space:

1) Coarse-grained description.

An XML tag called <*grb_arch*> with the detailed structure is designed as shown in Fig.5(a), and the related comments explain the meaning of the attributes. Tag <*ocb*>, <*icb*> and <*gsb*> describe the connection in OCB, ICB and GSB. Tag <*from*> describes the different driving sources within each module. The attribute *type* specifies the type of driving sources. It can be four types: global segments, OCB outputs, ICB outputs and LB output pins. The attribute *num_foreach* specifies the connection number of these driving sources. The attribute *reuse* decides whether the source is connected to the sink with output sharing or not. Besides, there is a tag <*seg_group*> which specifies the driven segment types.

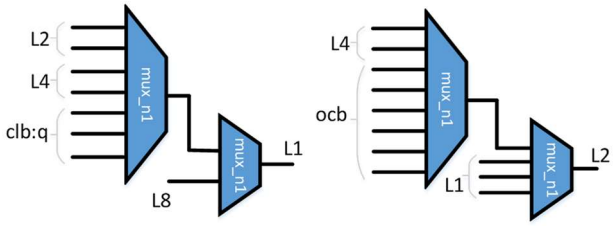
Fig.5(b) shows the connection schematics of tag <gsb> in Fig.5(a). There are eight L1 tracks where each L1 track is driven by two L1 tracks, two L2 tracks, one L8 track, and three clb:q output pins. The first three are connected by two-level MUXes, and the last one, L8, is connected to L1 directly. There are six L2 tracks and each L2 track is driven by three L2 tracks, two L4 tracks, and six OCB outputs. The first one is connected directly, and the latter two are connected by two-level MUXes.

```
<grb_arch>
  <!--arch_gsb_switch: the switch type used in GSB;-->
  <gsb arch_gsb_switch="mux_n1">
    <!--
      name: the name of the segment to be driven;
      track_nums: the number of the segment to be driven;
    -->
    <seg_group name="L1" track_nums="8">
      <!--
        type: the type of the driving source;
        name: the name of the driving source;
        num_foreach: the connection number of this driving source;
        reuse: whether two-level MUXes with output sharing or not;
        pin_types: the name of pins if the driving source is clb output;
      -->
      <from type="seg" name="L2" num_foreach="2" reuse="1"/>
      <from type="seg" name="L4" num_foreach="2" reuse="1"/>
      <from type="seg" name="L8" num_foreach="1" reuse="0"/>
      <from type="clb" name="clb" num_foreach="3" pin_types="q" reuse="1"/>
      ........
    </seg_group>
    <seg_group name="L2" track_nums="6">
      <from type="seg" name="L1" num_foreach="3" reuse="0"/>
      <from type="seg" name="L4" num_foreach="2" reuse="1"/>
      <from type="ocb" name="ocb" num_foreach="6" reuse="1"/>
      ........
    </seg_group>
    ........
  </gsb>

  <!--arch_icb_switch: the switch type used in ICB;-->
  <icb arch_icb_switch="mux_n2">
    <from type="seg" name="L1" num_foreach="2" reuse="1"/>
    ......
  </icb>

  <!--
    arch_ocb_switch: the switch type used in OCB;
    mux_nums: the mux number in OCB;
  -->
  <ocb arch_ocb_switch="mux_n3" mux_nums="16">
    <from num_foreach="6"/>
  </ocb>
</grb_arch>
```

(a) Example of tag <*gsb_arch*>



(b) Connection schematic of tag <*gsb* >

Fig.5 Coarse-grained description example

2) Fine-grained description.

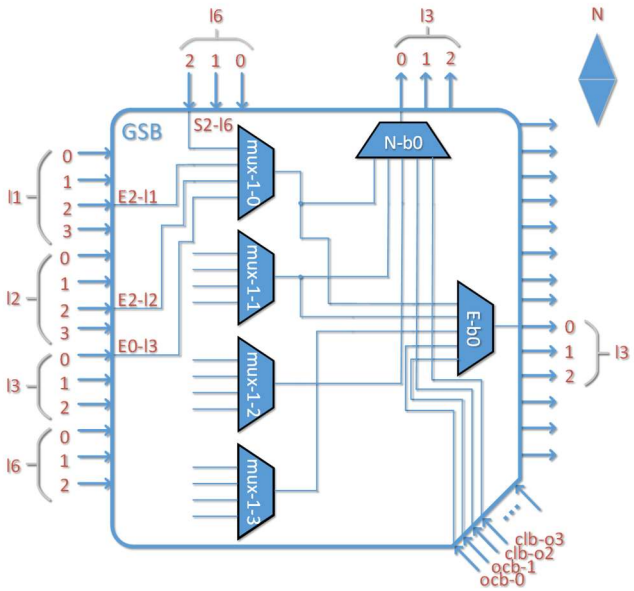
To support two-level MUXes in ICB and GSB. A detailed description tag <*multistage_muxs*> which can describe the detailed connection of each MUX in one tile is designed as shown in Fig.6(a). Tag <*first_stage*> and <*second_stage*> describe the first-level MUXes and second-level MUXes. Tag <*mux*> is used to describe the detailed connection of one MUX. Each <*from*> tag specifies one or more inputs. The attribute *type* and *name* specify the type and name of the driving source. The attribute *from_detail* specifies the detailed driving source with an index number: it consists of direction and track numbers for segments; while for other types, it consists of index numbers only. The attribute *mux_name* is only used in <*second_stage*>. It specifies the name of first-level MUXes which connect to the second-level MUXes.

Fig.6(b) shows an example of tag <*multistage_muxs*> and the corresponding connection schematic, mux-1-0 is the first-level MUX that has four inputs. E-b0 and N-b0 are the second-level MUXes. They have five inputs, three of which are the outputs of the first-level MUXes. The fine-grained

description can be generated by the information described in Fig.6 or written manually.

```
<multistage_muxs>
  <first_stage switch_name="only_mux">
    <mux name="mux-1-0">
      <from from_detail="E2" name="L1" type="seg"/>
      <from from_detail="E2" name="L2" type="seg"/>
      <from from_detail="E0" name="L3" type="seg"/>
      <from from_detail="S2" name="L6" type="seg"/>
    </mux>
    ......
  </first_stage>
  <second_stage>
    <mux name="E-b0" to_seg_name="L3" to_track="E0">
      <from mux_name="mux-1-0 mux-1-1 mux-1-3"/>
      <from from_detail="0 1" type="ocb"/>
    </mux>
    <mux name="N-b0" to_seg_name="L3" to_track="N0">
      <from mux_name="mux-1-0 mux-1-1 mux-1-2"/>
      <from from_detail="2 3" name="o" type="clb"/>
    </mux>
    ......
  </second_stage>
</multistage_muxs>
```

(a) Example of tag <*multistage_muxs*>

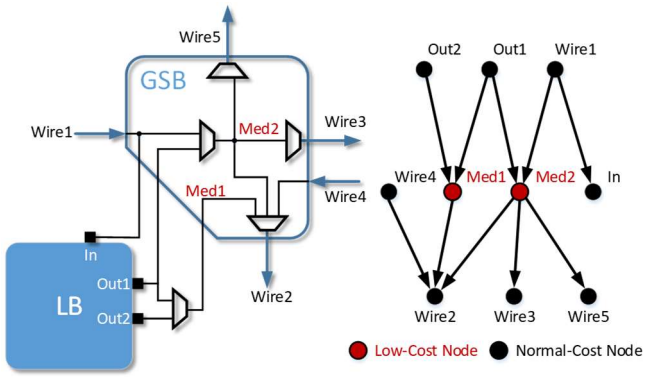


(b) Connection schematic

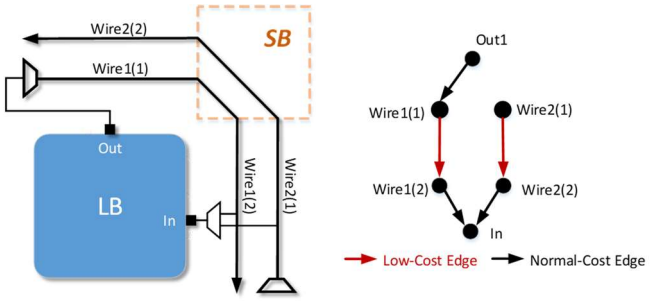
Fig.6 Two-level MUXes connection example

D. RRG generator

To support the GRB model, we rewrite the Routing Resource Graph (RRG) generator in VPR. The RRG is a directed graph $G = (V, E)$. V is a set of nodes, each $v_i \in V$ represents a routing track or an LB pin. E is a set of edges, each $e_{i,j} \in E$ represents a connection of a programmable switch between two nodes v_i *and* v_j. Our RRG generator generates concrete connections per tile according to the driving relationship or detailed MUX connections specified by the arch. For two-level MUXes, a low-cost node that includes no timing cost is used to represent the connection between first-level MUXes and second-level MUXes. And we also reserve extension interfaces for the more multilevel MUXes. Fig.7 shows an example of RRG for two-level MUXes. The method to support bent segments is the same as the reference [9]. Each part of bent wires is represented as a vertex in the RRG. A non-programmable and low-cost edge is used as the connection between adjacent parts of bent wires, as shown in Fig. 7(b).



(a) Two-level MUXes example



(b) Bent segment example

Fig.7 Example of RRG

IV. Architecture Exploration Platform

The design space for the architecture based on our model is enormous because there are so many subspaces that we can specify in the architecture, such as the distribution of segments, driving relationship in GRB and so on. It is impractical to search these spaces manually. So we propose an architecture exploration platform that is inspired by TORCH [22][23] to efficiently explore the design space of routing architectures. The simulated annealing algorithm is adopted as the overall framework. Algorithm 1 shows the pseudocode of our platform.

The platform takes baseline architecture *A* and benchmark sets *B* as inputs. Then it runs the *RunBaseline()* which calls VTR to evaluate the area and delay of baseline *A* with benchmark sets *B*. *InitArch()* generates an architecture randomly as the initial architecture to be optimized. We can also specify an architecture artificially. *EvaluateCost()* makes multiple calls to VTR in parallel to evaluate the delay and area of the architecture *M* on each benchmark and compare with the baseline, calculating the cost required by simulated annealing algorithm to evaluate the performance of the architecture *M* relative to baseline *A*. The expression of the cost function is as follows:

$$\text{cost(M)} = \frac{1}{k}\sum_{b=1}^{k}\left(\frac{a_{b,M}}{a_{b,base}}\right)^{\alpha} \times \left(\frac{d_{b,M}}{d_{b,base}}\right)^{\beta} \quad (1)$$

where $a_{b,M}, a_{b,base}, d_{b,M}$ and $d_{b,base}$ is the area and delay of architecture M and baseline A on benchmark sets B. The exponent $\alpha, \beta > 0$, control the trade-off between area and delay, k is the number of benchmarks.

The *while* loop at line 5 performs the simulated annealing process to explore architecture. *NewSegments()* updates the segments in four dimensions: length, bent pattern, number of each segment and number of segment types. *NewGRB()* updates the driving relationship in ICB and GSB and the MUX number of OCB. *NewArch()* generates a new architecture according to new segments and GRB. Then *EvaluateCost()* is called to evaluate the performance of the new architecture.

The possibility of architecture updating is controlled by the temperature *T*, the higher the *T*, the higher the possibility of updating and the greater the difference between the two architectures. As *T* decreases, the changes of the architecture will become smaller and smaller, and eventually converge to a new optimized architecture. Besides, if we only run part of *NewSegments()* or *NewGRB()* in the platform, we can explore a subspace.

Algorithm 1 Architecture Exploration Algorithm

Input: *Baseline architectureh A, Benchmark sets B*

Output: *Optimized architecture*

1: $(a^{base}, d^{base}) \leftarrow$ RunBaseline(B, A)
2: $M \leftarrow$ InitArch()
3: $c \leftarrow$ EvaluateCost$(M, B, a^{base}, d^{base}, A)$
4: $T \leftarrow$ InitTemperature()
5: **while** OuterLoopCriterion(T) is **false do**
6: $AcceptCount \leftarrow 0$
7: **while** InnerLoopCriterion() is **false do**
8: $Seg_{new} \leftarrow$ NewSegments(M, T)
9: $GRB_{new} \leftarrow$ NewGRB(M, Seg_{new}, T)
10: $M_{new} \leftarrow$ NewArch(Seg_{new}, GRB_{new})
11: $c^* \leftarrow$ EvaluateCost$(M_{new}, B, a^{base}, d^{base}, A)$
12: $\triangle c \leftarrow c^* - c$
13: **if** $\triangle c >= 0$ **then**
14: $r \leftarrow random(0, 1)$
15: **if** $r < e^{-\frac{\triangle c}{T}}$ **then**
16: $M \leftarrow M_{new}, c \leftarrow c^*$,$AcceptCount \leftarrow AcceptCount + 1$
17: **end if**
18: **else**
19: $M \leftarrow M_{new}, c \leftarrow c^*$,$AcceptCount \leftarrow AcceptCount + 1$
20: **end if**
21: **end while**
22: $T \leftarrow$ UpdateTemperature$(T, AcceptCount)$
23: **end while**

V. Experimental Results

A. Baseline Architecture

The parameters of the baseline architecture we used to compare are shown in TABLE II. The LB includes eight 6-input non-fracturable LUTs, and the connectivity of the input crossbar is set to 50% which is reasonable. The channel width is fixed. Because in the GRB model, the channel width is uniquely defined according to the number of each segment and it's reasonable in real FPGAs. The expression to calculate the channel width is shown in formula (2), where n is the number of segment types, N_k represents the number of this segment type, L_k represents the length of this segment type. We multiply it by 2 because we use unidirectional segments, each of which has two opposite directions.

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

The segment used in baseline architecture is length-4 straight wires with full switchpoints which is proved to provide the best performance in the unidirectional CB-SB model [25]. The area and delay parameters for segments and MUXes in the baseline are extracted from the transistor sizing tool COFFE 2 [24]. Besides, because COFFE 2 only supports the CB-SB model, we modified it to get the parameters for the GRB model. All of the parameters are determined at the 22 *nm* technology node. The trade-off between area and delay for COFFE 2 is 3:7 which achieves a great trade-off between the area and delay in our experiment.

TABLE II PARAMETERS OF BASELINE ARCHITECTURE

LB Size	Eight 6-input Non-fracturable LUTs
LB Input Crossbar	50%
DSP Elements	36×36 Fracturable Multipliers
Memories	32Kb Block RAMs
Channel Width	Same as the W in the GRB model to be compared
$F_{c,in}$	0.1
$F_{c,out}$	0.1
F_s	3
Switch Type	Wilton
Segments	Length-4 straight wires

B. Global Design Space Exploration

We use the platform discussed in Section-IV to search for architectures with great performance. The global design space includes the following parts: 1) segment type and distribution; 2) bent patterns; 3) driving relationship in GRB. It takes about 10 days on one machine with Intel(R) Xeon(R) Gold 6126 CPU @ 2.60GHz to finish the exploration while one iteration takes about one hour. The segment distribution and driving relationship of the best architecture reported by the platform are shown in TABLE III and TABLE IV. The definition of the bent pattern in TABLE III is the same as [9], while *ST* represents the straight type, (*n, CC/CW)* represents counter-clockwise or clockwise bend types in *n*-switchpoint. The architecture includes 7 length-1 straight wires, 9 length-2 straight wires, 5 length-6 straight wires, and 2 length-8 straight wires.

According to formula (2), the routing channel width is 142. L1 and L8 can be driven by all other segments to provide flexibility for both short segments and long segments. L1 does not drive L2 and L8 does not drive L6, one possible reason is that the length is approximately the same and they can drive ICB input directly. There is no need to add extra flexibility which may increase the delay.

The improvements of the architecture compared to the baseline are shown in TABLE V. The routing area reported by VTR 8.0 depends on the benchmarks because different benchmarks require different array sizes. For the CB-SB model, the total routing area includes both the global routing area and local routing area which equals the area of the crossbar in LBs. The GRB model achieves local routing in GSB so there is no crossbar in LBs. The results show that the architecture based on the GRB model has an average improvement of 15.6% in the critical path delay and 8.0% in

the area-delay product compared to the CB-SB baseline architecture.

TABLE III SEGMENT DISTRIBUTION OF THE BEST ARCHITECTURE

Name	Length	Number	Bent Pattern
L1	1	7	ST
L2	2	9	ST
L6	6	5	ST
L8	8	2	ST

TABLE IV DRIVING RELATIONSHIP OF THE BEST ARCHITECTURE

Sink	Source
L1	L1, L2, L6, L8, OCB output, LB output
L2	L2, L6, L8, OCB output, LB output
L6	L1, L2, L6, OCB output, LB output
L8	L1, L2, L6, L8, OCB output
ICB Input	L1, L2, L8, OCB output, LB output

TABLE V COMPARISON OF ARCHITECTURE WITH GRB MODEL AND CB-SB MODEL

Circuit	Total Routing Area(10^6)			Critical Path Delay(*ns*)			Area-Delay Product(10^6)		
	CB-SB	GRB	Ratio	CB-SB	GRB	Ratio	CB-SB	GRB	Ratio
arm_core	33.68	32.96	-2.1%	12.64	10.08	-20.2%	425.70	332.27	-21.9%
bgm	64.74	62.57	-3.4%	12.68	10.19	-19.7%	821.06	637.56	-22.3%
blob_merge	17.68	17.46	-1.3%	6.00	4.36	-27.3%	106.03	76.08	-28.2%
boundtop	2.17	2.34	7.7%	1.50	1.07	-28.6%	3.25	2.50	-23.1%
ch_intrinsics	1.80	1.98	10.0%	1.93	1.63	-15.4%	3.47	3.23	-7.0%
diffeq1	3.84	4.04	5.0%	18.91	17.36	-8.2%	72.66	70.10	-3.5%
diffeq2	3.84	4.04	5.0%	14.28	12.47	-12.7%	54.90	50.36	-8.3%
LU8PEEng	70.61	68.36	-3.2%	55.28	49.17	-11.0%	3903.43	3361.44	-13.9%
LU32PEEng	235.91	225.32	-4.5%	55.56	47.65	-14.2%	13108.01	10736.19	-18.1%
mcml	217.18	207.61	-4.4%	49.76	40.56	-18.5%	10806.61	8420.49	-22.1%
mkDelayWorker32B	36.66	35.78	-2.4%	5.51	4.31	-21.8%	202.16	154.30	-23.7%
mkPktMerge	11.02	11.10	0.8%	3.62	3.27	-9.6%	39.89	36.34	-8.9%
mkSMAdapter4B	5.41	5.61	3.7%	3.90	3.25	-16.7%	21.08	18.22	-13.6%
or1200	10.26	9.57	-6.8%	9.85	8.43	-14.4%	101.05	80.63	-20.2%
raygentop	8.01	8.14	1.5%	4.43	4.18	-5.5%	35.47	34.05	-4.0%
sha	7.37	7.47	1.4%	8.76	7.66	-12.5%	64.52	57.25	-11.3%
stereovision0	35.23	34.37	-2.4%	2.29	2.09	-8.6%	80.53	71.83	-10.8%
stereovision1	33.68	32.96	-2.1%	4.74	4.51	-4.9%	159.67	148.63	-6.9%
stereovision2	138.33	132.71	-4.1%	12.00	10.43	-13.1%	1659.74	1383.77	-16.6%
stereovision3	0.54	0.65	20.8%	1.72	1.27	-26.5%	0.93	0.83	-11.2%
Average	**1.0%**			**-15.5%**			**-14.8%**		

The area increases a little because of the difference in routing resources of IOs. In the GRB model, each IO has a GRB which is the same as LBs. But in the CB-SB model, the IOs on the left side and bottom side have a simpler SB compared to LBs, while the IOs on the right side and top side have no corresponding SBs. So the routing area of IOs in the CB-SB model is smaller than the GSB model. For small circuits, a large proportion of IOs may result in a larger routing area. We can see that in TABLE V, the circuits that increase in the area are a few small circuits. And the larger the circuit, the greater the area reduction. So the GRB model is better for large circuits. Besides, the use of two-level MUXes has a great influence on the area, as we discuss in Section B. The output sharing of two-level MUXes are adjusted during the architecture exploration and achieve a great trade-off between flexibility and area according to the area and delay weight we set in formula (1). Therefore, all circuits can be routed successfully and the routing area does not increase too much. The improvement of delay is mainly due to the richer connections which have been discussed in Section A. The two-level MUXes also has some contribution because it can reduce the MUX size.

We also compare our best architecture with length-4 architecture at the best area-delay product channel width. Different channel widths are tried on a step of 10 and we find that the area-delay product of the baseline is minimized when W=130. Compared to the CB-SB baseline with W=130, our architecture still reduces the critical path delay by 15.6% and area-delay product by 8.0%.

C. Local Subspace Exploration

The best architecture in global space exploration does not include any bent segment. The reason is that the global design space is too large to explore and it's difficult to tell which point has a bigger performance boost. The global exploration results can be fine-tuned for better results so we explore the following subspaces based on the iterative optimization method: a) segment distribution without bent pattern; b) segment distribution with bent pattern; c) driving relationship; d) firstly segment distribution with bent pattern and then driving relationship. All these explorations start from the same baseline which is generated randomly. Fig.8 shows the best improvement of these subspaces exploration compared to the CB-SB model with the same channel width as the GRB model. The results show that segment distribution without bent pattern has a 7.8% improvement in the delay and the improvement can be doubled with bent pattern. The segment distribution with bent type has a better performance in the area but is slightly weaker in delay compared to the driving relationship. Fig.9 shows the contribution of different parts to the delay improvement. The effect of the driving relationship on the delay is as great as the combination of the segment distribution and bent pattern. This seems to contradict the conclusions of Lin et al. in Figs. 10 and 11 [23]. It is because Lin et al. obtain their conclusions by iterating over different subspaces, such as the comparison of subspaces b) and d). The

improvement of the driving relationship shown in d) is based on the improvement of segment distribution with bent pattern. While our conclusion about the delay contribution is based on subspaces a), b) and c), which explore different subspaces separately.

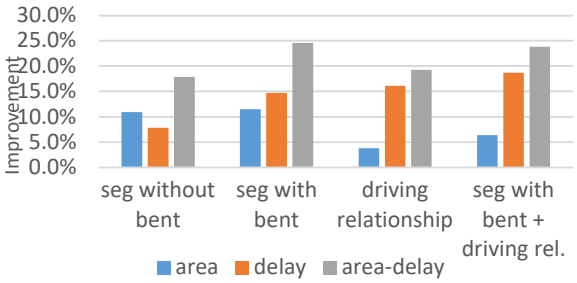


Fig.8 Best improvement of subspace exploration

Fig.9 Contribution of different factors to delay

The driving relationship has more influence on the area because it can change the number and size of MUXes in GRB. It tends to reduce delay at the expense of area because we focus more on delay and set the coefficient β to be much larger than α in formula (1). So the exploration of the latter two subspaces tends to be more about the extremity of delay and less about the area. The driving relationship exploration based on the best segments can further improve the delay, but not by much, which indicates that the exploration has fallen into the optimal solution.

The best segment distribution reported in b) is shown in TABLE VI. The channel width for the best architecture chosen according to equation (2), W = 2*(1*8+2*8+3*5+3*1 +5*1+8*2+11*2) = 170. The number of segment types increases and there are two kinds of bent segments. The best driving relationship for this segment distribution reported in d) is shown in 0. L1, L2 and L3-a can connect to ICB inputs and they can drive many other segments. Bent segments and long segments can only drive themselves or some short segments. This is similar to the conclusion mentioned in [25]. We also compare the average improvements of different benchmarks between our model and some other routing architectures based on similar baselines with the same routing architecture, the result is shown in TABLE VIII. Our architecture has improved significantly in terms of both area and delay.

TABLE VI THE BEST SEGMENT DISTRIBUTION

Name	Length	Number	Bent Pattern
L1	1	8	ST
L2	2	8	ST
L3-a	3	5	ST
L3-b	3	1	(2, CW)
L5	5	1	(3, CC)
L8	8	2	ST
L11	11	2	ST

TABLE VII THE BEST DRIVING RELATIONSHIP OF SUBSPACE EXPLORATION

Sink	Source
L1	L1, L3-a, L8, OCB output, LB output
L2	L1, L2, L8, OCB output, LB output
L3-a	L1, L2, L3-a, L5, OCB output, LB output
L3-b	L1, L2, L3-a, L3-b, L5, OCB output
L5	L1, L2, L3-a, L5, OCB output
L8	L1, L2, L8, L11
L11	L1, L2, L3-a, L11
ICB Input	L1, L2, L3-a, L5, OCB output, LB output

TABLE VIII THE IMPROVEMENT COMPARISON OF DIFFERENT ROUTING ARCHITECTURES

Name	Delay	Routing Area	Area-Delay Product
GRB	18.7%	6.4%	23.8%
GIB[15]	9.5%	1.8%	11.1%
Bent Pattern[9]	9%	3.2%	10.6%
Two-level MUX[21]	16.5%	-7.8%	14.2%

VI. CONCLUSION

In this paper, we propose a new routing architecture model, GRB for complex FPGAs based on VTR 8.0. Two sets of parameters, coarse-grained and fine-grained descriptions, are adopted to describe our model. To improve routing architecture performance, an exploration platform based on the simulated annealing algorithm is developed. The optimization result of the global design space exploration shows the GRB architecture can achieve an improvement of 15.5% on the critical path delay and 14.8% on the area-delay product on average compared to the CB-SB architecture. After some subspace exploration, results show driving relationship has a better performance in the delay than segment distribution with bent patterns, while the latter one is better in the area-delay product. The best architecture of subspaces exploration can reduce the critical path delay by 18.7% and the area-delay product by 23.8%. In the future, we plan to support more general commercial routing architectures, optimize the exploration platform and find FPGA architectures that are better than the 7-Series architecture family[20].

ACKNOWLEDGE

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, p.2, 1999.

[2] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *in Proc. ACM/SIGDA 10th Int. Symp. Field-Programmable Gate Arrays*, pp. 21–30, 2006.

[3] G. Lemieux, E. Lee, M. Tom, and A. Yu, "Directional and single-driver wires in FPGA interconnect," *Proc. - 2004 IEEE Int. Conf. Field-Programmable Technol. FPT '04*, pp. 41–48, 2004.

[4] F. S. Pourhashemi and M. Saheb Zamani, “Evaluation of FPGA routing architectures under process variation,” *Proc. ACM Gt. Lakes Symp. VLSI, GLSVLSI*, pp. 351–354, 2011.

[5] A. Mishra, N. Jayapalan, H. Rastogi, and T. Agrawal, “Impact of segmentation distribution on area and delay in FPGA routing architectures,” *Proc. 2013 3rd IEEE Int. Adv. Comput. Conf. IACC 2013*, pp. 1595–1599, 2013.

[6] Y. W. Chang, D. F. Wong, and C. K. Wong, “Universal switch modules for FPGA design,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 1, no. 1, pp. 80–101, 1996.

[7] X. Tang, E. Giacomin, A. Alacchi and P. Gaillardon, "A Study on Switch Block Patterns for Tileable FPGA Routing Architectures," *2019 International Conference on Field-Programmable Technology (ICFPT)*, Tianjin, China, pp. 247-250, 2019.

[8] S. Sivaswamy, G. Wang, C. Ababei, K. Bazargan, R. Kastner, and E. Bozorgzadeh, “HARP: Hard-wired routing pattern FPGAs,” *ACM/SIGDA Int. Symp. F. Program. Gate Arrays - FPGA*, pp. 21–29, 2005.

[9] X. Sun, H. Zhou, and L. Wang, “Bent routing pattern for FPGA,” *Proc. - 29th Int. Conf. Field-Programmable Log. Appl. FPL 2019*, pp. 9–16, 2019.

[10] A. Roopchansingh and J. Rose, “Nearest neighbour interconnect architecture in deep submicron FPGAs,” *Proc. Cust. Integr. Circuits Conf.*, pp. 59–62, 2002.

[11] S. Nikolić, G. Zgheib, and P. Ienne, “Straight to the point: Intra- and intercluster LUT connections to mitigate the delay of programmable routing,” *ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, pp. 150–160, 2020.

[12] K. E. Murray et al., “VTR 8: High-performance CAD and Customizable FPGA Architecture Modelling,” *ACM Trans. Reconfigurable Technol. Syst*., vol. 13, no. 2, 2020.

[13] C. L. Zhou, R. C. C. Cheung, and Y. L. Wu, “What if merging connection and switch boxes -an experimental revisit on FPGA architectures,” *Int. Conf. Commun. Circuits Syst., vol. 2, no. May 2014*, pp. 1295–1299, 2004.

[14] K. Ma, L. Wang, X. Zhou, S. X. D. Tan, and J. Tong, “General switch box modeling and optimization for FPGA routing architectures,” *Proc. - 2010 Int. Conf. Field-Programmable Technol. FPT’10*, pp. 320–323, 2010.

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

[16] C. Hu, Q. Duan, P. Lu, W. Liu, J. Wang, and J. Lai, “A tile-based interconnect model for FPGA architecture exploration,” *ACM Gt. Lakes Symp. VLSI, GLSVLSI*, no. c, pp. 113–118, 2020.

[17] Y. Ma and M. Lin, "Collaborative Routing Architecture for FPGA," *IEEE International Symposium on Circuits and Systems*, pp. 3700-3703, 2007.

[18] Z. Qi et al., "Timing Model For GRM FPGA Based Routing," *14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, pp. 1-3, 2018.

[19] 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," in *IEEE Micro*, vol. 40, no. 4, pp. 49-57, 1 July-Aug. 2020.

[20] Morten B. Petersen, Stefan Nikolić, and Mirjana Stojilović. “NetCracker: A Peek into the Routing Architecture of Xilinx 7-Series FPGAs,” *in Proc. ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp.11–22, 2021.

[21] Y. Shen, J. Qian, K. Shi, L. Wang, and H. Zhou, “Two-level MUX Design and Exploration in FPGA Routing Architecture.” *31th Int. Conf. Field-Programmable Log. Appl.*, pp. 234–241, *2021.*

[22] M. Lin and A. El Gamal, “TORCH: A design tool for routing channel segmentation in FPGAs,” *16th international ACM/SIGDA Symposium on Field Programmable Gate Arrays.*, pp. 131–138, 2008.

[23] Lin, J. Wawrzynek and A. E. 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.

[24] 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.

[25] O. Petelin and V. Betz, “The speed of diversity: Exploring complex FPGA routing topologies for the global metal layer,” *FPL 2016 - 26th Int. Conf. Field-Programmable Log. Appl*., pp. 1–10, 2016.