Efficient FPGA Routing Architecture Exploration Based on Two-Stage MUXes

Jide Zhang, Kaixiang Zhu, Kaichuang Shi, Lingli Wang*, Hao Zhou
State Key Laboratory of ASIC and System
Fudan University, Shanghai, China
*Corresponding Author's Email: llwang@fudan.edu.cn

***Abstract*—Employing large routing multiplexers (MUXes) in FPGA results in significant area and delay overheads. Hence, the two-stage cascaded structure with small MUXes can reduce the area and delay effectively. However, the manual design of a two-stage MUX topology is challenging to find optimal architecture. An automatic switch pattern exploration framework is proposed based on a novel routing architecture, TSRB (Two-Stage MUX Routing Block), which consists of two-stage MUXes in switch box (SB) and local interconnection block (LIB). The framework can enable the router to select commonly used switches based on the usage of switch connections under the premise of solving congestion, leading to a minimal set of switch types used in the architecture. The TSRB switch pattern is generated aided by a pre-exploration method to avoid manual design. The pattern optimization is then followed by two-stage MUX exploration. Our exploration framework can generate a TSRB switch pattern and optimize the pattern by pruning more than half of the switch connections, achieving an average 7% shorter critical path delay with a 5% area reduction compared to the one-stage large MUX routing architecture.**

I. Background and Related Works

FPGA routing architecture design is always challenging since it significantly affects the area and delay. For the traditional island-FPGA [1], the routing architecture mainly includes global routing wires, switch box (SB) and local interconnection block (LIB), which consists of the connection box (CB) and the crossbar inside the cluster logic block (CLB). Recent studies [2] [3] implement large MUX in FPGA routing architecture, effectively improving routability. However, area and delay increase linearly with the MUX size, making the MUX topology design of FPGA require a trade-off between routability and area delay consumption.

To avoid the area and delay overheads of large MUX without affecting routability, two-stage MUX routing architecture is proposed in [4]. This manually designed two-stage MUX topology has a significant delay advantage over large MUX with a small cost of area. However, the challenge of designing a two-stage MUX topology lies in the design of the switch pattern. The manual design method and artificially imposed constraints limit the two-stage MUX design space, making it difficult to find an optimal architecture.

The automatic switch-block exploration method proposed in [5], abbreviated as $avalanche$, can select a minimum set of switch types through a series of VPR routing experiments. It allows the router to explore the design space without constraints. The router can select the switch connections with modified $avalanche\ cost$ during the routing. The more frequently used switch connections have a lower $avalanche\ cost$ so that they are more likely selected as the fixed switch pattern, leading to the minimum set of switch types under the premise of eliminating congestion. However, that paper only explores the SB pattern without covering the CB and crossbar.

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

Inspired by $avalanche$, two-stage MUX topology can also be designed by automatic exploration. The contributions of this paper are as follows:

- The TSRB architecture proposed in this paper utilizes two-stage MUXes to replace large MUXes in both the SB and LIB. The two-stage MUXes in the SB and LIB are divided into multiple groups, and the interconnections only exist within the group, which optimizes the routing area and delay effects.
- An automatic TSRB exploration framework is proposed based on $avalanche$. To avoid designing a two-stage MUX topology manually, a pre-exploration method is proposed to select a set of switch types in one-stage MUX $avalanche$, which are used for TSRB generation. Then two-stage MUX exploration is followed for TSRB pattern optimization.
- Compared with the one-stage MUX architecture, the optimized TSRB architecture can achieve a 5.21% reduction in area and an average 7.7% decrease in critical path delay with the proposed exploration framework, which can provide new insights for the two-stage MUX automatic design.

II. TSRB Architecture Introduction

Fig.1 describes a two-stage MUX topology of SB/LIB. The first stage MUX (FSM) accepts inputs from global wires and CLB outputs and drives the second stage MUX (SSM) via a crossbar, which can be either fully or custom connected. The SSM receives inputs from FSM and drives either the global wires (SB) or directly connects to LUT input pins (LIB).

Notably, not all FSMs are connected to all SSMs but are divided into several groups called Sub-SBs or Sub-LIBs. The FSMs within each group only connect to the SSMs within that group, reducing the complexity of the crossbar between the two stages of MUXes.

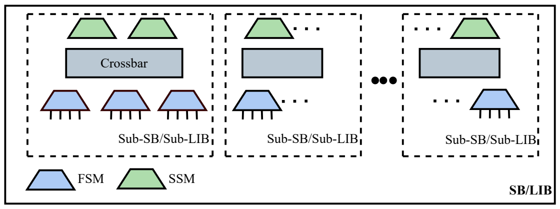


Fig. 1: Two-stage MUX topology

III. Extended Avalanche exploration

Avalanche in [5] generates a routing resource graph (RRG) with full connectivity among global routing wires. These switches are called candidate connections, each with an *avalanche cost*. The router freely chooses the path and candidate connections during the routing iteration. The *avalanche cost* of candidate connections can be adjusted based on their usage after each routing iteration. This negotiation mechanism cooperates with *congestion negotiation* [6], leading the router to select commonly used switch types while eliminating congestion. After each iteration, the candidate connections with high usage can be selected into a fixed switch pattern. The exploration iteration ends when the router can route all nets successfully with no candidate connections.

To conduct TSRB exploration, we make a series of extensions based on *avalanche*, as outlined below:

1) Larger Design Space Exploration: The work in [5] only optimizes the switch pattern in the SB, but the local interconnections can also be optimized. The framework for SB and LIB exploration is constructed, which supports more extensive design space exploration.

2) Delay Modeling: HSPICE is adopted to measure the delay of two-stage MUX topology. We fix the transistor sizes and assume that every circuit block(e.g. a MUX, a CLB, etc.) is a square with an area equal to the sum of all circuit element areas within the block and the length of wire across the block is equal to the width of that block.

3) Multi-Threading Exploration: Multiple circuits are merged on a single FPGA to explore the switch pattern in [5]. However, the larger exploration space of TSRB leads to an even greater complexity of routing resources, which causes a long routing time. Therefore, the framework routes test circuits separately with multiple threads and combines the candidate connection usages at the end of each iteration, reducing the iteration time significantly.

IV. TSRB generation and exploration

Fig.2 shows the design framework for TSRB generation and exploration. We define a candidate connection from in element (wire or pin) to out element as $cc(in, out)$. Pre-exploration of one-stage MUX $avalanche$ is used to select a set of switch types, $\{cc(in, out)\}$, which is used to generate the TSRB switch pattern. The TSRB pattern is further optimized by two-stage MUX $avalanche$. VPR [7] is used to analyze the usage of candidate connections. Based on the VPR experiment, the framework selects eligible candidate connections, generates a new architecture file and RRG, and starts a new iterative exploration.

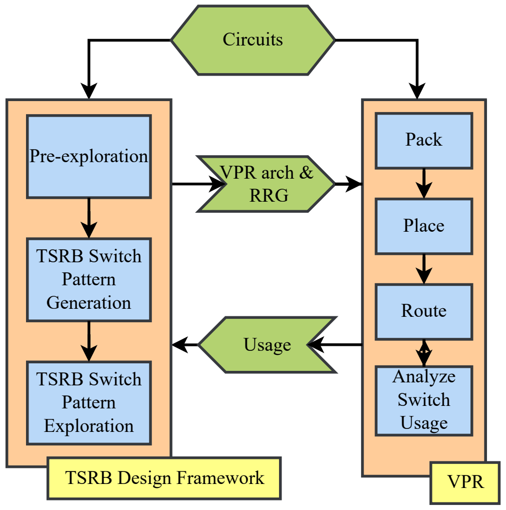


Fig. 2: TSRB exploration framework

A. TSRB pre-exploration

Designing a two-stage MUX topology unrestrictedly is too general for analysis. Pre-exploration is proposed, which uses $avalanche$ based on a fully-connected one-stage MUX architecture, which contains all possible switch types that may be used. Pre-exploration is discontinued at a specific iteration considering runtime and a set of candidate connections, $\{cc(in, out)\}$, is obtained for designing a two-stage MUX topology.

After several one-stage MUX $avalanche$ iterations, we choose the candidate connections selected into the fixed pattern or used in the discontinued iteration to design the TSRB switch pattern, and their fan-in weights formulas are shown in the table. I. In the formula, $usage$, max_usage means the candidate connection usage and maximum usage in discontinued iteration. The cc_1 is the most important, with the weight set to 1. The weight represents the dependent relationship between in and out within the candidate connections, which produces an effect in generating the TSRB switch pattern.

TABLE I: Candidate connections in pre-exploration

Type	Description	Weight
cc_1	Been selected into fixed pattern	1
cc_2	Been used in discontinued iteration but have not been selected	$usage/max\_usage$

B. TSRB Generation and Exploration

The TSRB switch pattern generation method is proposed and illustrated in Algorithm 1. The output bandwidth of each group—OB, the maximum size of FSM—FSM_{max} are required to specify. Based on the $\{cc(in, out)\}$

obtained in pre-exploration and these two parameters, the algorithm 1 generates the TSRB switch pattern, $\{cc(in, medium), (medium, out)\}$, where $medium$ means medium node between two stages of MUXes or output of FSM. Grouping out elements that contain the most common in elements can effectively limit the total number of fan-ins within the group and save FSM resources. After grouping out elements, all in elements and their weights are obtained within a group. Then the in elements are distributed into FSMs in turn. The algorithm iterates over all existing FSMs within the group that have not reached the maximum size, compares the cost of inserting the in element to the eligible FSMs with creating a new FSM, and finally generates the least costly connection. The FSM cost function is shown in Formula 1, which is related to the number of existing FSMs, FSM_num, the minimum size of the existing FSMs, min_size and the sum of in element weights. The cost function tends to create new FSM at the beginning of allocation as it helps to create adequate FSMs to increase input bandwidth. However, as more in elements are inserted, the number of FSMs tends to stabilize, and the function favors allocating in elements to existing FSMs. A fully populated crossbar connects the two stages of MUXes since there are only several SSMs in each group.

$$cost_{FSM} = \begin{cases} e^{FSM\_num-min\_size^2}, & new\ FSM \\ sum_{weight}, & existing\ FSM \end{cases} \quad (1)$$

Algorithm 1 TSRB Generation

```
Input: $Usage_{pre}$ , $OB$ , $FSM_{max}$
Output: $FSMs$, $TSRB\_pattern$
 1: $FSMs$ = {}, $TSRB\_pattern$ = {};
 2: Get $cc(in, out)$ and assign weights from $Usage_{pre}$;
 3: $Groups$ = making_Groups($OB$);
 4: for Each group do
 5:     calculate $in$ element weights
 6:     for Each $in$ element within the group do
 7:         $min\_cost$, $min\_cost\_mux$=find_mux($FSM_{max}$);
 8:         $new\_mux\_cost$ = new_mux_cost();
 9:         if $min\_cost > new\_mux\_cost$ then
10:             $min\_cost\_mux$ = build_new_mux();
11:         end if
12:         insert_$in$_element_to_mux($min\_cost\_mux, weight$);
13:         cc = generate_cc($in$,$min\_cost\_mux$);
14:         $FSMs$,$TSRB\_pattern$ = update_info();
15:     end for
16: end for
17: return $FSMs$,$TSRB\_pattern$;
```

TSRB pattern exploration is conducted after switch pattern generation, which can bring TSRB sparser connectivity and introduce less load for routing wires.

V. Experimental Results

A. FPGA Architecture

The baseline parameters used to compare with TSRB are shown in the table. II. Eight 6-input non-fracturable LUTs are included within CLB, and the crossbar is 50% populated. Uni-directional, multiple wire types are utilized in TSRB architecture. The length of routing wires is set to 1,2,4,6 with a proportion of 1:1:1:1, and the channel width is fixed to 160 [8]. For baseline large MUX topology, each wire is required to drive at least one wire of each type (length and direction) and to be driven by at least one wire of each type, except wires that come from the direction of the driving wire.

TABLE II: FPGA architecture parameters

Paramemter	Value
Logic Block	Eight 6-input Non-fracturable LUTs
CLB Input Crossbar	50% populated
DSP Block	36*36Fracturable Multipliers
Memory Block	32Kb Block RAMs
Segment Type	1, 2, 4, 6
Segment Proportion	1:1:1:1
Channel Width	160

B. Experiment Setup

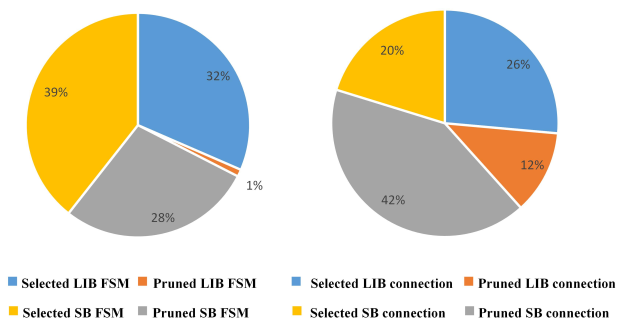
$Alu4$, $misex3$, $ch\_intrinsics$ and $diffeq2$ are employed as the test circuits for automatic exploration. Delay modeling is based on the 22nm PTM model [9]. OB_{SB} and OB_{LIB} are set to 6 and 8, respectively while FSM_{max} is set to 8, which are referred to [4]. Since our focus is routing architecture, the logic block delays are from $k6\_N8\_gate\_boost\_0.2V\_22nm$ architecture distributed with VTR 8.0 [7].

C. Pre-exploration

A fully-connected one-stage MUX architecture is generated for pre-exploration, which consists of 21,216 candidate connections. The pre-exploration is set to end after 30 iterations. There are 247, 5,817 candidate connections of cc_1, cc_2, respectively, which together accounted for 28.5% of the total.

D. TSRB Generation and Exploration

TSRB switch pattern is generated based on pre-exploration and algorithm 1, which contains 95 IB-FSMs, 197 SB-FSMs and 4012 candidate connections. TSRB switch pattern exploration is conducted to optimize two-stage MUX topology further. After 133 iterations, the optimized TSRB pattern is obtained, including 207 FSMs and 1,872 candidate connections, accounting for 70.89% of the FSMs and 46.66% of the total candidate connections, respectively. The detailed data is shown in Fig.3. It proves that TSRB exploration effectively prunes candidate connections related to SB; even 28% SB-FSMs that have never been used are pruned, reducing tile area. In contrast, candidate connections related to LIB are pruned by 12%, with only 1% of LIB-FSMs pruned. It indicates that the connection relationship in LIB is hard to prune, which is consistent with expectations.



(a) FSM **(b) Candidate connection**

Fig. 3: Results of TSRB switch pattern exploration.

A comparison of MUX delays between optimized TSRB and baseline is shown in Fig.4. The delay across two-stage MUXes is better than a single large MUX delay. It also can be observed that the longer the wire length, the better the optimization of delay, which is reasonable due to wire loads.

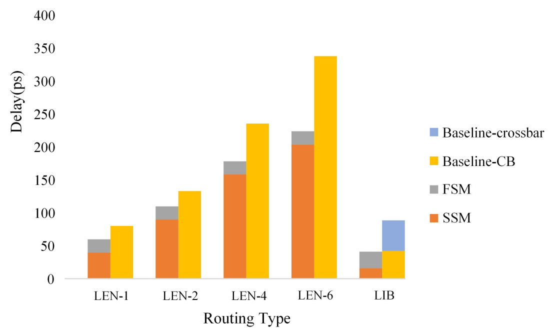


Fig. 4: The comparison of MUX delays between optimized TSRB and baseline.

Table.III compares the optimized TSRB architecture and baseline in terms of tile area and critical path delay on VTR benchmarks. An average 7.70% improvement in critical path delay with a 5.21% reduction in tile area can be achieved. The experiment results prove that our automatic exploration framework can effectively generate, and optimize the TSRB switch pattern and the optimized TSRB architecture can achieve good area and delay optimizations without affecting routability.

VI. Conclusion

This paper proposes a novel TSRB architecture using an automatic switch pattern exploration framework. The TSRB switch pattern is generated aided by pre-exploration, and TSRB pattern exploration is performed for further optimization. The exploration framework can prune more than half of the candidate connections. The optimized TSRB obtains an average 7.7% critical path delay improvement with 5.2% area reduction. However, the exploration framework only supports the routing wires without switchpoints, resulting in the inability to compare with the existing two-stage MUX architecture in [4]. Our future work will further improve the framework to support different FPGA architectures for more convincing comparisons.

TABLE III: Comparison of optimized TSRB and baseline

	Tile Area(um^2)		
	TSRB	Baseline	Ratio(%)
	426.93	**449.17**	**-5.21**

Benchmark	Critical Path Delay(ns)		
	TSRB	Baseline	Ratio(%)
arm_core	14.9624	17.1509	-12.76
blob_merge	6.33072	7.1648	-11.64
bgm	14.7179	16.3911	-10.21
boundtop	1.52871	1.84177	-17.00
ch_intrinsics	1.94491	2.43777	-20.22
diffeq1	18.7879	20.8397	-9.85
diffeq2	14.422	15.5853	-7.46
LU8PEEng	67.4494	75.1254	-10.22
LU32PEEng	67.8522	69.3052	-2.10
mcml	59.4959	61.9227	-3.92
mkDelayWorker	9.21438	9.24904	-0.37
mkPktMerge	4.33836	4.54874	-4.63
mkSMAdapter4B	4.27376	4.70139	-9.10
or1200	11.4957	11.6071	-0.96
raygentop	4.64625	5.1368	-9.55
stereovision0	3.59072	3.45334	3.98
stereovision1	5.74508	5.74066	0.08
stereovision2	15.7157	18.7217	-16.06
stereovision3	1.65728	1.71909	-3.60
Average			**-7.70**

References

[1] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for deep-submicron FPGAs*, vol. 497. Springer Science & Business Media, 2012.

[2] D. Lewis, G. Chiu, J. Chromczak, D. Galloway, B. Gamsa, V. Manohararajah, I. Milton, T. Vanderhoek, and J. Van Dyken, "The stratix™ 10 highly pipelined FPGA architecture," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 159–168, 2016.

[3] Xilinx, "Ultrascale architecture and product overview," 2015.

[4] 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.

[5] S. Nikolić and P. Ienne, "Turning PathFinder upside-down: Exploring FPGA switch-blocks by negotiating switch presence," in *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 225–233, IEEE, 2021.

[6] L. McMurchie and C. Ebeling, "PathFinder: A negotiation-based performance-driven router for FPGAs, 1995," *Google Scholar Google Scholar Digital Library Digital Library*.

[7] K. E. Murray, O. Petelin, S. Zhong, J. M. Wang, M. Eldafrawy, J.-P. Legault, E. Sha, A. G. Graham, J. Wu, M. J. Walker, *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.

[8] 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.

[9] PTM, "PTM - predictive technology model." https://ptm.asu.edu/, 2018.