An Enhanced Packing Algorithm for FPGA Architectures without Local Crossbar

Yuanqi Wang, Kaichuang Shi, Lingli Wang*
State Key Laboratory of ASIC and System
Fudan University
Shanghai, China
*llwang@fudan.edu.cn

***Abstract*—In the EDA process of FPGA, VTR (Verilog-to-Routing) is a commonly used open-source CAD tool in the academic community, and VPR (Versatile Place and Route) is the back-end process of VTR. The packing algorithm in VPR is a seed-based architecture-aware algorithm, which has strong versatility. However, the packing density is low for CLB architectures without local crossbar interconnect in the commercial FPGAs, which results in long critical path delays in circuits. To solve this problem, this paper optimizes the packing algorithm in VPR by adding a virtual crossbar in CLB, aiming to improve the internal utilization of each CLB and decrease the critical path delay of the circuit. The experimental result shows that it can reduce the average number of packed CLBs by 56.51% and the average critical path delay by 9.14% after the packing algorithm enhancement.**

Keywords—FPGA，Packing Algorithm，Crossbar，CLB

I. Introduction

In the back-end process of the entire EDA flow for FPGA, packing is the first stage, followed by placement and routing [1]. A packer reads in an architecture file of the FPGA and a netlist produced by logic synthesis. Then it assigns the netlist blocks to proper configurable logic blocks (CLBs), before entering the placement process.

There is a local crossbar interconnect [2] in each CLB, which provides rich routing resource to connect signals from CLB input pins to LUT input pins in the traditional FPGA architecture in academia. However, the advantage is at the cost of larger area, which is increasingly unbearable in commercial FPGAs. Hence there are no crossbars in recent commercial FPGAs [3]. As a consequence, the EDA algorithm needs to be improved for the adaption to changes in commercial FPGA architectures.

In this paper, a packing algorithm based on AAPack in VPR is enhanced to improve the performance of the packer. This paper is organized as follow: Section 2 briefly introduces the background of AAPack algorithm and the structure of crossbars in CLBs. Section 3 analyses the reason why AAPack performs badly for CLBs without crossbars and how to improve the algorithm. Section 4 gives the experiment results. Section 5 concludes this paper.

II. Background

In this section, we first briefly introduce AAPack algorithm. Then the structure of the crossbar in CLB is presented.

A. AAPack Algorithm

AAPack [4] is the packing algorithm in VPR where inputs to the packer are a technology mapped netlist of unpacked netlist blocks and a description file of FPGA architecture. The output is a netlist of packed CLBs.

The first step of the algorithm is pre-packing [5], aiming to group some netlist atoms that are not able to pack in different clusters because of inflexibility in interconnect into molecules. Secondly, a molecule is selected as a seed, and then the packer creates a new cluster containing the seed molecule. In the third step, it continuously finds a candidate molecule according to an affinity function and attempts to pack the molecule into the current cluster. This step continues until the cluster is full or no more molecules can be packed into it. Every time when this step is over, it means the packing for current cluster is done and the packer jumps back to the second step to establish a new cluster. The whole process stops when all molecules are packed.

Attempting to pack a molecule into the current cluster in the third step mainly consists of three parts: 1) it finds a candidate location in the cluster for the molecule; 2) it checks whether packing on the chosen position is feasible for routing resource; 3) through the cluster's interconnect, it routes the molecule's nets in the cluster. These parts are similar to placement and routing process within a cluster.

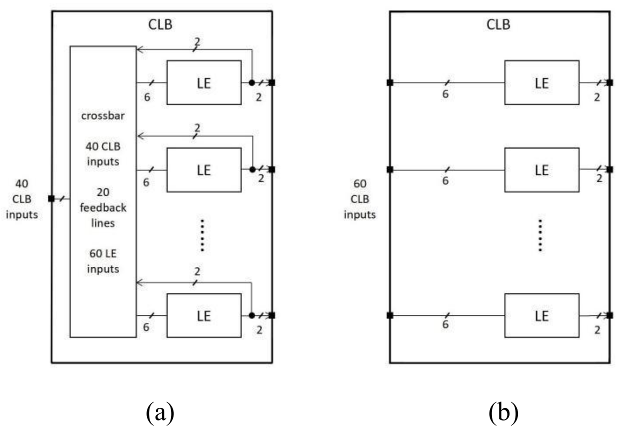

Fig. 1. (a) Structure of CLB with crossbar.[4] (b)Structure of CLB without crossbar.[7]

B. Crossbar in CLB

In the traditional FPGAs, CLBs consist of N logic elements (LEs) to implement diverse functions and each LE has K input pins. The local connections within a cluster can be presented as a crossbar. Each crossbar has I cluster inputs, $2*N$ feedbacks of LEs as inputs and $N*K$ outputs connected to LE inputs, as shown in Fig. 1.(a) in which the value of I, N and K is 40, 10 and 6 respectively. It provides abundant routing resource in CLBs.

However, there are no crossbars in CLBs of some recent commercial FPGA architectures, such as AMD/Xilinx UltraScale [6] and Intel Stratix-10 [7]. It means the input pins of CLBs are directly connected to LE input pins, and no feedbacks exist inside the CLB. A simplified architecture is illustrated in Fig. 1.(b). It will cause significant deterioration in the packing results like low utilization within a CLB.

Hence, it is important to optimize the packing algorithm targeting for the architecture without crossbars in CLBs.

III. Packing Algorithm Enhancement

The AAPack in VPR can result in significantly different packing results for CLBs without crossbars and CLBs with crossbars. It is obvious that the number of CLBs in architecture with crossbars is much less than that those without crossbars as a result of packing.

We first analyze the reasons for the difference, then a specific enhancement is proposed subsequently.

A. Analysis

Based on Section II-A, it can be found that it is almost the same in choosing a seed molecule and candidate molecules with or without crossbars. The difference occurs when attempting to pack the molecule into the current cluster. The three parts of this step are introduced briefly in Section II, and the main difference happens in the second part. In other words, after checking feasibility of the chosen positions, more molecules are considered to be infeasible in the exact position when there is no crossbar in CLBs compared to the CLBs with crossbars. Most of these unpacked molecules cannot be packed with others, so they are basically placed separately in different clusters, which leads to the low utilization inside each CLB.

A technique called pin counting is used to determine whether a placement is feasible or not. Pins in a CLB are divided into several classes as soon as the architecture file is read and each class has a pin number. If the number of pins required to be used is more than the number of practical pins in the class, the placement of the molecule is considered to be infeasible. The process of classification is influenced by the specific architecture. The number of pins in every pin class is smaller without crossbars in CLBs because of the strict rule of pin classification, leading to more illegal placements.

B. Enhancement

The classification rule in the pin counting is complex, and modifying it may result in extra negative influence. So adding a virtual crossbar in the CLB and reclassify pins is a better method to enhance the packing algorithm for architecture without crossbars.

After being read into VPR, the architecture file is converted into a graph and stored in a data structure. As Fig.1.(b) shows, an input pin of a CLB has only one edge connected to it if there is no crossbar, and this exact edge merely connects one input pin of an LE. No feedback connections are available in this situation.

The virtual crossbar is added before the feasibility checking stage. After positioning the molecule, if there is no crossbar in the data structure, we create new edges from all input pins of the CLB to all input pins of LEs in it. Besides, the feedbacks need to be added as well in the same way, by building new edges from all output pins to all input pins of LEs. Lastly, these new-built edges are bound to the corresponding pins data structures.

As soon as the virtual crossbar is added, the function of pin classification is called again to reclassify the pins in the CLB, and the number of pins in some pin classes increases so that many molecules can pass the feasibility checking in a relatively loose classification.

Some routing problems also need to be taken into consideration. The routing inside a cluster is supported by the routing resource graph (RRG) representing the internal connections of a CLB. A node in the routing graph represents a pin on the physical block and directed edges between nodes corresponds to the paths through CLB interconnects. The part of RRG generated from a CLB with the crossbar shown in Fig. 1.(a) is illustrated in Fig. 2, and the dash-dotted lines in the figure means the feedback nets. This step aims to find a feasible path from a source pin to the sink pin correspondingly on the RRG. A cluster is packed denser by the virtual crossbar. However, there is no enough internal routing resource to implement some netlist connections like feedback nets. So the feedback routing path should be moved outside the cluster and be implemented by interconnect nets between CLBs in the general routing stage. The packing algorithm does not need to be modified because the RRG generated by AAPack allows any CLB output pin to connect to any other input pin in the same CLB as the dashed line shown in Fig. 2. As a consequence, the router in clusters can find a path for feedback connections in the RRG though there are no feedback nets inside.

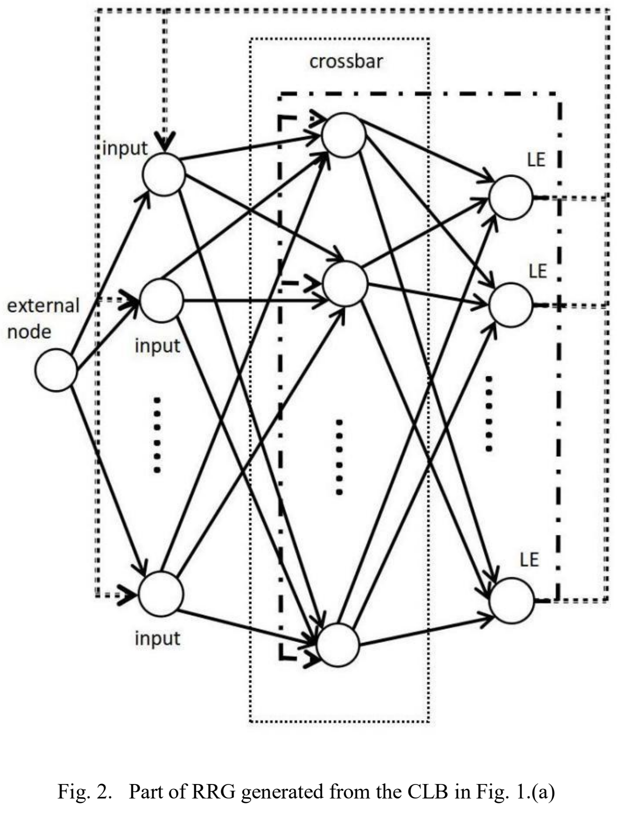

Fig. 2. Part of RRG generated from the CLB in Fig. 1.(a)

IV. Experimental Results

The complete flow of attempting to pack a molecule into a cluster is shown in Fig. 3. The experiment focuses on two results: 1) the number of CLBs after packing and 2) the critical path delay after the entire VPR flow. Twenty-one VPR benchmark designs and twenty Koios benchmark designs [8] are used for evaluation. The Odin II+ABC flow is first run to obtain the BLIF files, which are the input files to VPR. The architecture file is a 40nm CMOS architecture in VTR, k6_frac_N10_mem32K_40nm.xml [9], which has a crossbar and ten 6-input LEs in each CLB. Its internal structure of a CLB is briefly shown in Fig. 1.(a). The crossbar in this architecture is deleted to test the optimized algorithm. The original 40 input pins are not enough in this situation, so 20 inputs are added, implying that total 60 inputs of CLB directly one-to-one connect to 60 LE input pins. The routing channel width is set to 300 which is common in the previous work [10] [8].

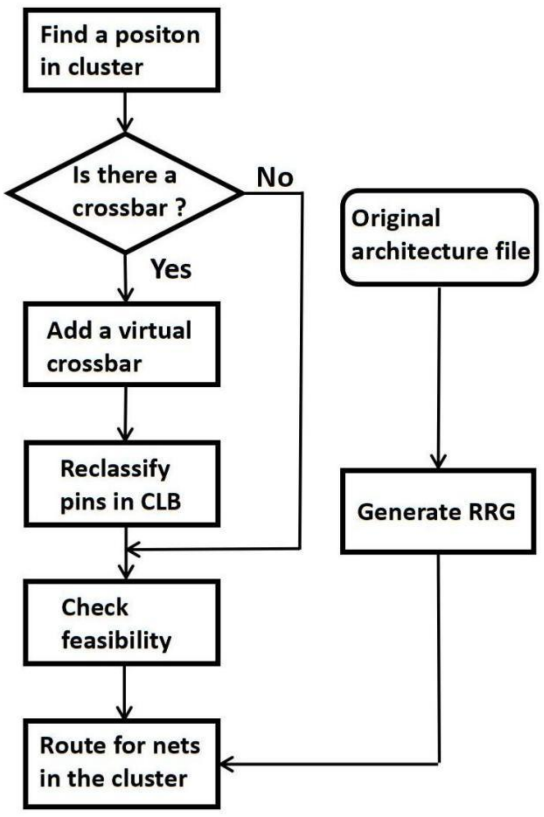

Fig. 3. Complete flow chart of packing a molecule into a cluster

The results are shown in TABLE I. It can be seen that the number of CLBs after packing in architecture without crossbars is large, which leads to long critical path delay.

Data in the column labeled as "-*Δ*" is calculated by the difference between the result of optimized VPR and baseline VPR without crossbar divided by the former number, representing for the reduced proportion of the number of CLBs, the critical path delay, total area and maximum routing channel utilization. It can be seen that after optimizing the packing algorithm, the number of CLBs and the critical path delay has a reduction of 53% and 9.14% respectively, compared to baseline VPR.

The optimization comes from the improvement of the packing algorithm. The new-added virtual crossbar loosen the pin classification. As a consequence, more molecules can pass the feasibility checking and then be packed into the current cluster when the packer tries to fill it. This contributes to dense clusters, in other words, the number of packed CLBs decreases. Naturally, total area has a reduction of 34.81% on average.

A drawback of the enhancement is nonnegligible. Because of high density in clusters and lack of internal routing resource of CLBs without crossbars, many connections need to be implemented by global routing wires between CLBs. This will increase routing channel utilization by 31.92% on average as shown in the last column of TABLE I, which means higher chance to be congested and to fail in routing stage.

V. Conclusion

In this paper, an enhancement for packing algorithm in FPGA back-end flow is developed, aiming to improve packing results for FPGA architecture without crossbars. This optimized packer can make every cluster denser and experimental results show that the number of CLBs after packing has a reduction of 53% on average compared to the baseline algorithm. Meanwhile, the critical path delay after the entire VPR flow reduces by 9.13%, which is a considerable improvement. As most of commercial FPGAs have no crossbars, this research can bring significant gain for recent commercial architectures.

References

[1] V. Betz, J. Rose, A. Marquardt, "Architecture and CAD for deep-submicron FPGAs." *Kluwer Academic Publishers Norwell, MA, USA,* 1999, pp.18-26.

[2] G. Lemieux, "Using sparse crossbars within LUT." *International Symposium on Field Programmable Gate Arrays*, 2001, pp.59-68.

[3] B. Gaide, D. Gaitonde, C. Ravishankar and T. Bauer, "Xilinx Adaptive Compute Acceleration Platform: Versal (TM) Architecture", *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 108-116, 2015.

[4] J. Luu, J. Rose, and J. Anderson, "Architecture Description and Packing for Logic Blocks with Hierarchy, Modes and Complex Interconnect."*Annual ACM International Symposium on Field-Programmable Gate Arrays*, 2011, pp.227-236.

[5] J. Luu, J. Rose, and J. Anderson, "Towards Interconnect-Adaptive Packing for FPGAs." *Annual ACM International Symposium on Field-Programmable Gate Arrays*, 2014, pp.21-30.

[6] Xilinx Corporation, "Versal ACAP Configurable Logic Block Architecture Manual." 2023.

[7] Intel® Stratix® 10 FPGAs and SoC FPGAs Support, "Programming, Reference & Implementation Guides for Developers." 2018.

[8] A. Arora et al, "Koios: A Deep Learning Benchmark Suite for FPGA Architecture and CAD Research." *31st International Conference on Field Programmable Logic and Applications (FPL)*, 2021, pp.355–362.

[9] J. Luu, J. Goeders, V.Betz. http://github.com/verilog-to-routing/vtr-verilog-to-routing/blob/master/vtr_flow/arch/timing/k6_frac_N10_mem32K_40nm.xml

[10] K. Shi, X. Zhou, H. Zhou, L. Wang, "An Optimized GIB Routing Architecture with Bent Wires for FPGA." *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 2022, pp.1-28.

[11] K. Shi, H. Zhou, L. Wang, "A Hexagon-Based Honeycomb Routing Architecture for FPGA." *20th International Conference on Field-Programmable Technology (ICFPT)*, 2021, pp.179-184

TABLE I. EXPERIMENTAL RESULTS

Benchmark	**Number of CLBs**			**Critical Path Delay/ns**			**Total Area/(10^8)**			**Maximum Routing Channel Utilization**		
	Baseline VPR	***Optimized VPR***	***-Δ***	***Baseline VPR***	***Optimized VPR***	***-Δ***	***Baseline VPR***	***Optimized VPR***	***-Δ***	***Baseline VPR***	***Optimized VPR***	***-Δ***
VTR Benchmarks												
arm_core	3220	959	70.22%	23.94	20.96	12.45%	3.70	1.11	70.09%	0.63	0.83	-31.75%
bgm	10548	2504	76.26%	24.78	20.00	19.29%	11.92	2.91	75.60%	0.61	0.7	-14.75%
blob_merge	1975	577	70.78%	14.28	13.23	7.40%	2.28	0.66	71.19%	0.47	0.66	-40.43%
boundtop	85	82	3.53%	1.99	2.08	-4.75%	0.09	0.09	0.00%	0.26	0.26	-0.00%
ch_intrinsics	72	65	9.72%	2.91	2.76	5.04%	0.09	0.08	15.36%	0.17	0.25	-47.06%
diffeq1	159	36	77.36%	24.51	24.19	1.34%	0.18	0.16	10.72%	0.28	0.52	-85.71%
diffeq2	108	29	73.15%	19.53	19.74	-1.07%	0.21	0.21	0.00%	0.23	0.49	-113.04%
LU8PEEng	6980	1979	71.65%	119.21	100.24	15.92%	8.00	2.28	71.41%	0.58	0.73	-25.86%
LU32PEEng	25227	6959	72.41%	111.58	99.04	11.24%	28.66	7.96	72.21%	0.7	0.83	-18.57%
mcml	13438	5449	59.45%	102.59	92.56	9.78%	15.38	6.88	55.31%	0.54	0.72	-33.33%
mkDelayWorker32B	458	459	-0.22%	8.37	8.00	4.48%	1.97	1.97	0.00%	0.19	0.21	-10.53%
mkPktMerge	29	24	17.24%	3.94	4.05	-2.87%	0.57	0.57	0.00%	0.17	0.19	-11.76%
mkSMAdapter4B	346	166	52.02%	6.76	6.15	9.08%	0.40	0.28	30.99%	0.37	0.63	-70.27%
or1200	873	250	71.36%	20.93	19.86	5.13%	1.01	0.53	47.83%	0.5	0.69	-38.00%
raygentop	207	104	49.76%	6.42	5.98	6.88%	0.34	0.34	0.00%	0.52	0.57	-9.62%
sha	1359	209	84.62%	16.12	16.09	0.18%	1.56	0.24	84.82%	0.44	0.59	-34.09%
spree	170	66	61.18%	13.18	12.59	4.49%	0.21	0.13	41.58%	0.59	0.62	-5.08%
stereovision0	1101	743	32.52%	4.30	4.28	0.35%	1.27	0.87	31.74%	0.49	0.56	-14.29%
stereovision1	1430	760	46.85%	5.77	5.49	4.74%	1.64	1.22	25.52%	0.53	0.66	-24.53%
stereovision2	6659	2288	65.64%	21.31	20.26	4.90%	8.83	8.83	0.00%	0.51	0.66	-29.41%
stereovision3	34	13	61.76%	3.18	2.90	8.82%	0.04	0.02	56.94%	0.23	0.24	-4.35%
Koios Benchmarks												
attention_layer	5059	1491	70.53%	12.49	10.36	17.05%	5.78	3.70	35.96%	0.52	0.71	-36.54%
bnn	24626	7861	68.08%	14.58	13.66	6.36%	28.06	8.96	68.06%	0.59	0.84	-42.37%
conv_layer	2909	1410	51.53%	11.92	10.99	7.82%	3.34	2.48	25.69%	0.57	0.76	-33.33%
conv_layer_hls	2874	1873	34.83%	11.99	11.94	0.43%	9.17	9.17	0.00%	0.47	0.65	-38.30%
eltwise_layer	2212	1024	53.71%	6.18	5.32	13.92%	2.83	2.83	0.00%	0.56	0.77	-37.50%
gemm_layer	69810	22398	67.92%	23.15	14.85	35.87%	78.75	25.26	67.93%	0.63	0.88	-39.68%
lstm	16830	5981	64.46%	15.29	12.96	15.25%	19.29	9.17	52.48%	0.56	0.87	-55.36%
reduction_layer	4426	967	78.15%	8.95	7.01	21.67%	4.98	1.50	69.84%	0.41	0.78	-90.24%
robot_rl	5711	1434	74.89%	15.57	14.50	6.90%	6.54	1.97	69.96%	0.55	0.72	-30.91%
softmax	4247	1410	66.80%	10.54	7.81	25.88%	4.87	1.64	66.36%	0.54	0.62	-14.81%
spmv	1631	771	52.73%	7.12	6.38	10.41%	8.13	8.13	0.00%	0.44	0.64	-45.45%
clstm_like.small	25302	10178	59.77%	16.63	11.14	33.01%	28.66	11.55	59.68%	0.58	0.75	-29.31%
clstm_like.medium	49585	19374	60.93%	16.77	13.17	21.43%	55.91	21.87	60.88%	0.61	0.8	-31.15%
clstm_like.large	74143	28557	61.48%	20.61	15.20	26.22%	84.04	32.52	61.30%	0.58	0.83	-43.10%
dla_like.small	14310	6569	54.10%	14.68	14.63	0.37%	18.04	18.04	0.00%	0.68	0.74	-8.82%
dla_like.medium	33554	16096	52.03%	17.59	17.10	2.78%	38.11	27.50	27.85%	0.65	0.75	-15.38%
tiny_darknet_like.small	9794	4924	49.72%	19.16	19.87	-3.69%	92.42	92.42	0.00%	0.54	0.59	-9.26%
tiny_darknet_like.medium	39537	16310	58.75%	18.43	18.30	0.69%	111.11	111.11	0.00%	0.62	0.69	-11.29%
tpu_like.small	4639	1676	63.87%	8.11	7.80	3.79%	7.46	7.46	0.00%	0.52	0.61	-17.31%
tpu_like.medium	10850	5934	45.31%	14.08	13.25	5.88%	29.96	29.96	0.00%	0.62	0.72	-16.13%
Average			**56.51%**			**9.14%**			**34.81%**			**-31.92%**