The advancement of cluster based FPGA place & route technic

Zhimei Zhou[1], *Yong* Wan[1,*], *Yin* Liu[2], *Xiaoyan* Guo[2], *Qilin* Yin[2], and *Chen* Feng[1]

[1]Smart Shine Microelectronics Technology CO. Ltd, Beijing 10089, China
[2]Information & Telecommunications Company. State Grid Shandong Electric Power Company, Jinan 25010, China

Keywords: FPGA, Place and route, Cluster structure.

Abstract. As one of the core components of electronic hardware systems, Field Programmable Logic Array (FPGA) device design technology continues to advance under the guidance of electronic information technology policies, and has made information technology applications. huge contribution. However, with the advancement of chip technology and the continuous upgrading of information technology, the functions that FPGAs need to perform are more and more complicated. How to efficiently perform layout design and make full use of chip resources has become an important technology to be solved and optimized in FPGA design. The FPGA itself is not limited to a specific function. It contains internal functions such as memory, protocol module, clock module, high-speed interface module and digital signal processing. It can be programmed through logic modules such as programmable logic unit modules and interconnects. Blank FPGA devices are designed to be high performance system applications with complex functions. The layout and routing technology based on cluster logic unit blocks can combine the above resources to give full play to its performance advantages, and its importance is self-evident. Based on the traditional FPGA implementation, this paper analyzes several advantages based on cluster logic block layout and routing technology, and generalizes the design method and flow based on cluster logic block layout and routing technology.

1 Introduction

Currently, most SRAM-based FPGA logic cell blocks use a look-up table (LUT) structure [1,2]. The complexity of the logical functions that the LUT can achieve is directly proportional to the number of inputs. At the same time, the complexity of the internal structure of the LUT increases with the number of inputs. On the one hand, we need to enter more LUTs to reduce the wiring between logic blocks and the interconnection area. On the

* Corresponding author: yaoliang@sgitg.sgcc.com.cn

other hand, we do not want to reduce the efficiency of placement and routing because the complexity of the LUT is too high.

When implementing circuits with FPGAs, place and route is often the most time-consuming steps [3,4], To improve the speed of place and route and optimize the use efficiency of FPGA resources, we need to combine some LUTs first, and then use local interconnects to form blocks. Place and route on the basis of these logical unit blocks, and such logical unit blocks are called logical cluster [5~8]. Using cluster-based logic unit blocks can significantly reduce the compilation time of a design. As the scale of FPGAs increases, it becomes very important to control compilation time, otherwise the key advantages of FPGAs will no longer exist [9~12].

The effect of placing multiple LUTs into a logical cluster on placement and routing is very complex. On the one hand, merging related LUTs into a single logical unit block can reduce the number of interconnecting networks between logical unit blocks, thereby saving interconnection area. On the other hand, for the logic cluster placement and routing technology, the area occupied by local interconnects increased quadratically with the size of the logic cluster. In this way, for a sufficiently large logical cluster, the area occupied by the local interconnect will exceed the area saved by the global interconnect [13~17].

This paper explored the following four aspects of cluster structure logic block placement and routing technology:

① The optimal number of inputs that the FPGA interconnect should assign to each LUT of the logical cluster;

② The change in the connectivity of the logical block and the interconnect interface when the number of LUTs inside a logical unit block changes;

③ To achieve the best routing speed and area utilization for an FPGA, the number of LUTs should be included in a logic cluster;

④ The time required to compile the circuit is affected by the size of the logical cluster.

2 Cluster-based placement and routing process

The goal of this paper is to optimize FPGA routing speed and increase area utilization based on the cluster structure. Due to the lack of detailed analysis models of FPGA structures and circuits, experimental methods are adopted here to evaluate various structures. Use 20 reference circuits for various FPGA structure implementations to be evaluated, and record the routing speed and required area of the circuit.

The experiment use a typical FPGA automated CAD to implement the circuit: mapping, placement, and routing.

First, use a synthesis tool to optimize each circuit for process-independent logic. Then, the circuit obtains a 4-input LUT and trigger via the FlowMap process mapping tool, and then the FlowPack post-processing algorithm optimizes the mapping result. The timing driver packs a netlist containing 4-input LUTS and triggers into logical clusters corresponding to specific I and N values. At this point, the circuit is completely described in the form of interconnected logic cell blocks in the target FPGA. Finally, place and route the circuit (including global and detailed routing) with CAD.

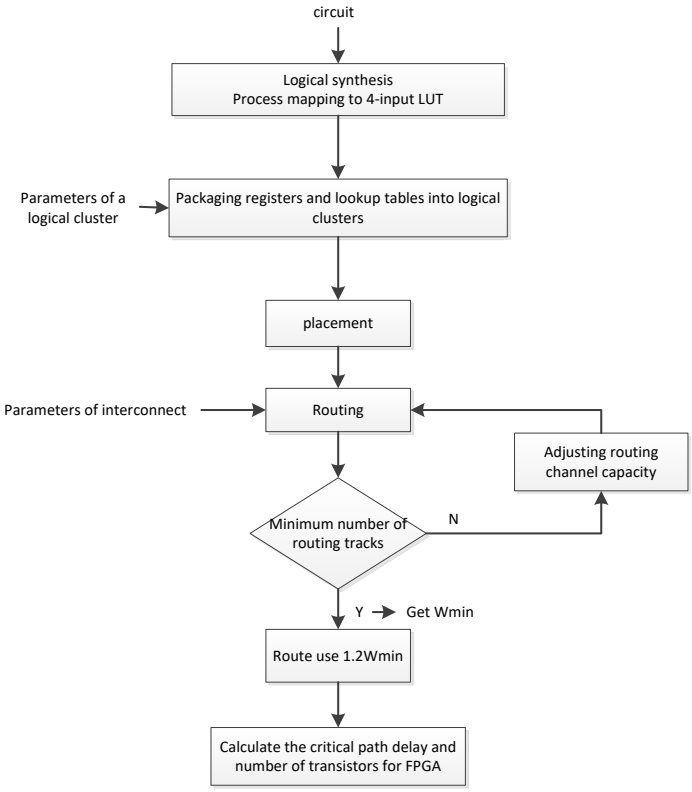

Fig. 1. Placement and routing process based on cluster structure.

As shown in Figure 1, the circuit is routed with different channel widths until the placement and routing device finds the minimum channel width required to successfully route the circuit, that is, Wmin. When manufacturing FPGAs, usually sufficient interconnection resources are built in the chip, so that the FPGA will have some redundancy for "typical" circuits. For this reason, set the channel width to 1. 2Wmin, and then perform "loose" routing for each circuit. Then use the delay model to estimate the critical path of the circuit, and use the area model to estimate the total area of the transistor needed for the FPGA layout. After this process, there is enough information to compare the speed and area utilization of different logical block structures.

In the above, if the FPGA width of the implemented circuit is fixed, then it can be known whether the circuit routing is successful, but it is difficult to draw conclusions about the FPGA structure from this "binary" result. Our improvement is to allow the channel width to vary with the needs of the circuit. By adjusting the channel width and finding the minimum value, minor improvements that can be overlooked in FPGA structures or CAD algorithms can be detected.

3 Cluster-based logical element block

As shown in Figure 2, the Basic Logic Element (BLE) contains a LUT whose output can be selected from the first-level register output, or the bypass register is directly output and controlled by an output selector. The use of lookup tables and registers in combinational logic and sequential logic structures is a common method for commercial FPGAs.

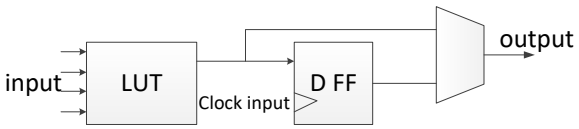

Fig. 2. Structure of Basic Logic Element (BLE).

As shown in Figure 2, the Basic Logic Element (BLE) contains a LUT whose output can be selected from the first-level register output, or the bypass register is directly output and controlled by an output selector. The use of lookup tables and registers in combinational logic and sequential logic structures is a common method for commercial FPGAs.

Based on the BLE, we can build a complex logic block which is the cluster-based logic element block- logic clusters.

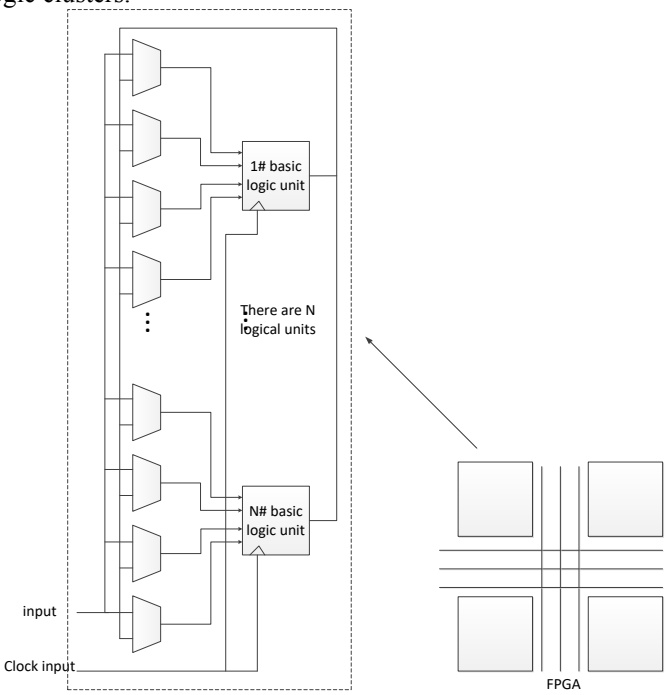

Fig. 3. Structure of the logical cluster.

There are four parameters that can be used to describe a logical cluster:

① the number of input (I);
② the number of BLE (N);
③ the number of input clock (MCLK);
④ the number of inputs to a LUT (K).

As shown in figure 3, the interface of the logical cluster includes clock input, I input, and N logical cluster outputs. I input is not connected directly to the BLE, but through a multi-channel selector array, select the LUT after input to the BLE, and the output of the each BLEt is connected to the multi-channel selector array of input, through multi-channel selector control signal, the output of the cluster within any BLE input can be connected to any BLE. At the same time, the input/output of other logical clusters can also be connected to the input/output of the current logical cluster through FPGA routing resources, which can be designed flexibly.

Since the input of BLE can be connected to any cluster input or output of BLE, the logical cluster is fully connected. This is a useful feature that can significantly simplify CAD tools. For example, if we want to determine whether all appropriate signals can be connected to the BLE input, just calculate the number of input signals required by each BLE within the cluster (excluding the signals generated within the cluster), and then compare this number with the input number I of the cluster.

4 Cluster based logic unit block packaging algorithm VPack

This section describes the BLE block packaging tool VPack. The packaging process of VPack is divided into two stages: first, VPack can input K according to the LUT we defined, and package LUT and register as BLE; Then, multiple BLEs are packaged into logical clusters based on the defined target parameters (I, N, MCLK). The input to VPack is the network table containing LUT and registers, and the output is the network table of the logical cluster.

The first phase of the algorithm adopts the pattern matching algorithm [18], that is, package a register and a LUT into a BLE as easily and efficiently as possible. As shown in figure 4, when the output of a LUT is faned out to a single register, VPack packages the LUT and register as a BLE first. The LUT and the register need to be packaged separately when the output of the LUT is connected to other logical resources in addition to the fan out to the register.

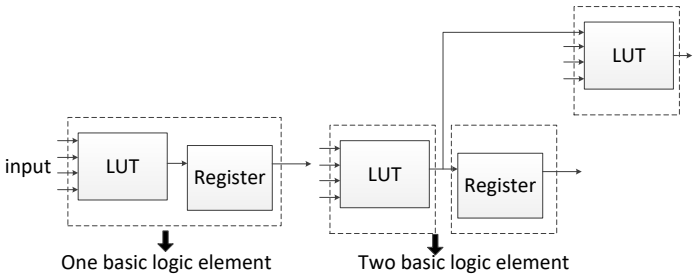

Fig. 4. Packages LUT and registers into basic logical unit blocks.

In the second stage of the algorithm, the BLE packaged in the first stage needs to be repackaged into logical clusters. The optimization objective of this stage is complicated and consists of two parts. First, the number of logical unit blocks needs to be optimized according to its capacity definition N. Secondly, in order to improve the routing rate, it is necessary to minimize the number of wiring between logical clusters, that is, the less the input number of each logical cluster, the better. A set of BLEs that can be packaged into a single logical cluster needs to satisfy the following conditions:

① The number of BLE must be less than the size of the cluster N;
② The number of input signals of the cluster must be less than or equal to I;
③ The number of clocks used by each BLE must be less than MCLK.

The steps for VPack to construct the cluster consist of two phases:

The initial phase USES the greedy algorithm [19~21], In the greedy phase, VPack first selects a seed BLE for the current cluster, then selects the BLE with the highest attraction factor for the seed BLE, and adds it to the cluster. The formula for calculating the attraction factor is as follows:

$$Attraction(B) = \left| Nets(B) \cap Nets(C) \right| \tag{1}$$

In the formula, B represents the target BLE, and C represents the current cluster. VPack gives preference to the most input but unpackaged BLE as the seed, because such a unit can take advantage of the most cluster input. From formula (1), we can see that the attraction factor of a BLE B and the current cluster C is the number of inputs/outputs they share.

The attraction factor function makes VPack tend to add the most relevant BLE to the current cluster and minimize the number of added inputs to the packaged logical cluster. However, in even if a BLE has a high attraction factor to the current cluster, its addition will result in an illegal cluster (the parameters of the cluster violate the input number I or clock number MCLK defined by the target), then the unit cannot be added.

The greedy phase of packaging BLEs into the current cluster using the greedy algorithm will continue until the cluster is full or illegal becase the addition of any unpacked cell. In the first case, VPack chooses a new seed BLE to start packaging greedily again. In the second case, if the number of BLE packages within the cluster is less than N, but the input number or clock number of the cluster is not enough to add any BLE, VPack will call the second climbing phase.

Packing the clusters into the second phase is a difficult task. VPack selects the BLE that causes the smallest increase in input in the cluster to be added to the current cluster. The increase in the number of cluster inputs caused by the addition of a basic logical unit block B is simply expressed as:

$$\Delta_{clusterinputs}(B) = \left|Fanin(B)\right| - \left|Nets(B) \cap Nets(C)\right| \tag{2}$$

During this climbing phase, VPack allows BLE to be added to the cluster that causes the cluster input to exceed I (but does not allow the clock count to exceed MCLK). When a BLE is added to a cluster, if all the inputs to the BLE are the inputs to the cluster, and its output is used by other BLE in the cluster, it will reduce one input in the cluster.

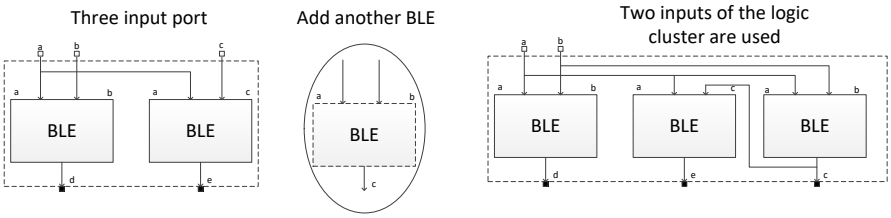

Fig. 5. Adding a BLE reduces the number of cluster inputs required.

The situation illustrated in figure 5 is the key to the climbing phase: if adding BLE makes the cluster illegal, adding BLE may make the cluster legal. This climbing phase stops when the cluster is filled;If the cluster is still illegal, VPack returns the result of the last valid cluster. VPack then proceeds to select the seed BLE for the next cluster, recalling the first stage as described earlier.

The climbing phase resulted in only a small increase in the number of packageable BLE (called logical utilization) in the cluster, compared to the greedy method alone. For many circuits, there was no effect, while for others, logic utilization increased by only 1% to 2%. The overall average logic utilization was improved by less than 1%. It is possible that VPack has been very effective for this type of packaging problem during the greed phase, or that the allowed type of climb is too restricted. If the repetitive logic can be moved naturally during the climbing phase, it is possible to increase the utilization of the climbing.

Finally, we can calculate the logical complexity of the logical cluster packaging algorithm by the following formula:

$$O = k_{\max} Kn \tag{3}$$

where, kmax represents the maximum number of network terminals, K represents the input number of LUT, and n is the sum of the number of LUT and register in the circuit.

Where, kmax represents the maximum number of network terminals, K represents the input number of LUT, and n is the sum of the number of LUT and register in the circuit.

5 Conclusion

This paper introduces the speed and area advantages of logic cluster-based architecture in FPGA place and route technology, describes the flow of logic cluster-based place and route technology and the specific structure of logical cluster-based unit block, and tests the packaging algorithm of place and route based on cluster logical unit block. Experimental analysis and research show that the place and route technology based on logical cluster unit block can effectively reduce the influence of logic complexity on packaging speed, and reducing the time and area required by FPGA place and route, improving the efficiency of chip resource utilization.

This research was financially supported by State Grid Science and Technology Foundation.
Project Name: Research on key technologies and sample development of FPGA chip in relay protection field
Project Number: 5100-201941437A-0-0-00

References

1. Bauer, Trevor J. , Young, Steven P. FPGA architecture with deep look-up table RAMs[P]. United States Patent, 2001, 6288568.
2. K. J Muhonen, M. Kavehrad, R. Krishnamoorthy. Look-up table techniques for adaptive digital predistortion: a development and comparison[J]. IEEE Transactions on Vehicular Technology, 2000, 1995-2002.
3. V. Betz, J. Rose. Cluster-based logic blocks for FPGAs: area-efficiency vs. Input sharing and size[C]. Proceedings of CICC 97 - Custom Integrated Circuits Conference, 1997.
4. V. Betz, J. Rose. How much logic should go in an FPGA logic block[J]. IEEE Design & Test of Computers, 1998.
5. Scott Y. L. Chin, Steven J. E. Wilton. An analytical model relating FPGA architecture and place and route runtime[J]. 2009 International Conference on Field Programable Logic and Applications, 2009.
6. K. Roy. Power-dissipation driven FPGA place and route under timing constraints[J]. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 1999, 634-637.
7. Laurent Sauvage, Sylvain Guilley, Jean-Luc Danger, Yves Mathieu, Maxime Nassar. Successful attack on an FPGA-based WDDL DES cryptoprocessor without place and route constraints[C]. 2009 Design, Automation & Test in Europe Conference & Exhibition, 2009.
8. Christopher Lavin, Brent Nelson, Brad Hutchings. Impact of hard macro size on FPGA clock rate and place/route time[C]. 2013 23rd International Conference on Field programmable Logic and Applications, 2013.

9. Alexander Marquardt, Vaughn Betz, et al. Using Cluster-Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density[J]. engr. arizona. edu, 1999.
10. Marvin Tom, Guy Lemieux. Logic block clustering of large designs for channel-width constrained FPGAs[C]. Proceedings of the 42nd annual Design Automation Conference, 2005, 726-731.
11. Russell Tessier, Heather Giza. Balancing logic utilization and area efficiency in FPGAs[C]. Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing, 2002, 535-544.
12. Christian Gleichner, Tobias Koal, et al. Effective logic self repair based on extracted logic clusters[J]. Signal Processing Algorithms, Architectures, Arrangements, and Applications SPA 2010.
13. A. Marquardt, V. Betx, J. Rose. Speed and area trade-offs in cluster-based FPGA architectures[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2000, 84-93.
14. E. Ahmed, J. Rose. The effect of LUT and cluster size on deep-submicron FPGA performance and density[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2004, 288-298.
15. Amit Singh, Ganapathy Parthasarathy, Malgorzata Marek-Sadowska. Efficient circuit clustering for area and power reduction in FPGAs[J]. ACM Transactions on Design Automation of Electronic Systems (TODAES), 2002, 643-663.
16. Peter Andrew Jamieson, Jonathan Rose. Enhancing the area efficiency of FPGAs with hard circuits using shadow clusters[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2010, 1696-1709.
17. Vijay Lakamraju, Russell Tessier. Tolerating operational faults in cluster-based FPGAs[J]. FPGA '00 Proceedings of the 2000 ACM/SIGDA eighth international symposium on Field programmable gate arrays, 2000, 187-194.
18. Gong aihui. Research on FPGA software packing algorithm [D]. Shanghai: fudan university, 2011.
19. A. Aloisio, V. Izzo, S. Rampone. FPGA implementation of a greedy algorithm for set covering[C]. 14th IEEE-NPSS Real Time Conference, 2005.
20. K. Nachiket, M. Nikil, et al. Packet switched vs. Time multiplexed FPGA overlay networks[C]. 2006 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2006.
21. T. Taghavi, S. Ghiasi, M. Sarrafzadeh. Routing algorithms: architecture driven rerouting enhancement for FPGAs[C]. 2006 IEEE International Symposium on Circuits and Systems, 2006.