applied sciences

MDPI

Article

Research on an Intelligent Test Method for Interconnect Resources in an FPGA

Weikun Xie [1], Wenjing Qi [2], Xiaohui Lin [3] and Houjun Wang [1,*]

[1] School of Automation Engineer, University of Electronic and Technology of China, Chengdu 611731, China; chinagrass@163.com
[2] Shenzhen Institute for Advanced Study, University of Electronic and Technology of China, Shenzhen 518110, China; wenjing11qi@163.com
[3] The 58th Research Institute of China Electronics Technology Group Corporation, Wuxi 214072, China; hebut_ck12_lxh@163.com
* Correspondence: hjwang@uestc.edu.cn; Tel.: +86-158-9536-3327

Abstract: With the rapid development of integrated circuit production technology, the scale of FPGA circuits has expanded to billions of gates. The complexity of the internal resource structures in the FPGAs (field programmable gate arrays) is continually increasing, and there is an increasing possibility of various faults in these circuits, especially in interconnect resources. These occupy more than 80% of a chip's area and have the highest fault rate. To ensure the reliability of the FPGAs, it is very important to perform high-coverage testing on the interconnect resources within them. This article uses AMD Xilinx's Kintex-7 series FPGA as the research object and proposes a deep-priority algorithm based on graph-based models and improved priority algorithms to intelligently wire the FPGA interconnected resources. The routing results were produced using a configuration script written in the XDL language, and the FPGA configuration and testing were conducted accordingly. This approach achieved a high coverage and intelligent testing for the interconnect resources in the FPGAs.

Keywords: FPGA; interconnect resources; high coverage; intelligent test

Citation: Xie, W.; Qi, W.; Lin, X.; Wang, H. Research on an Intelligent Test Method for Interconnect Resources in an FPGA. *Appl. Sci.* **2023**, *13*, 7951. https://doi.org/10.3390/app13137951

Academic Editor: Alexander Barkalov

Received: 15 June 2023
Revised: 4 July 2023
Accepted: 6 July 2023
Published: 7 July 2023

1. Introduction

With the implementation of new applications such as artificial intelligence, big data, and the industrial Internet of Things, the market demand for chip performance continues to increase. Field programmable gate arrays (FPGA) have been widely adopted due to the advantages they provide, such as high performance, low power consumption, and programmability. As application scenarios become increasingly complex, the required size of the FPGAs is also increasing, with current scales reaching billions of gates. The area of interconnect resources within the FPGAs now account for more than 80% of the total chip area [1]. The probability of failure in the interconnect resources is much higher than that in other resources, making the testing of the interconnect resources in the FPGAs very important [2]. However, due to the increasing complexity of the interconnect resource structures, the testing of the switch matrices and interconnect lines has become a very complex problem. Traditional routing testing methods can no longer meet the testing requirements of these ultra-large-scale FPGA interconnect resources, and there is an urgent need to research new intelligent testing methods to achieve automatic routing and testing.

As FPGA chips do not have any functional configurations when they leave the factory, the initial step in testing them is to test the resource configuration. Configuration-related testing methods are then combined to analyze the test results and locate any faults. To ensure complete test coverage for all resources within an FPGA chip, multiple configurations are often used, with the number of configurations directly impacting the efficiency and cost

of the test. Therefore, the key to efficient FPGA testing lies in generating the least number of configurations necessary to achieve full coverage.

While earlier research on interconnect resources testing within FPGAs focused mainly on testing the configurable logic blocks (CLBs) and random access memory (RAM) [3,4], as well as developing testing methods for these components, such methods are no longer sufficient for testing today's larger and more complex FPGAs. In recent years, research has emerged that focused on testing interconnect resources, utilizing modeling methods based on directed graph theory, wiring algorithms, and built-in self-tests (BISTs) [4–6]. These studies abstract the metal interconnects and switch matrices into directed graph models, using path search algorithms based on directed graphs such as network maximum flow algorithms, depth-first search algorithms, and breadth-first search algorithms to achieve automatic testing of the switch matrices [7,8].

However, most of this research has yet to establish a general testing model, and there are few effective methods for determining and locating fault types. While machine learning methods have been employed in some testing studies, these approaches require a large number of training sets and can be hindered by long code running times and uncontrollable test costs. Therefore, we utilized an improved depth-first algorithm to validate the Kintex-7 FPGA. After nine configurations, we achieved coverage of 100%.

Before testing an FPGA, it needs to be configured, which involves interconnecting resources to generate test paths. Manually configuring FPGA is time-consuming and increases testing costs due to the vast and complex interconnect resources. Therefore, in this paper, we propose an improved depth-first algorithm for wire routing search. Figure 1 is a structure diagram of the FPGA interconnect resource test process. Prior to the search, we need to extract and process the interconnect resources of the FPGA and convert them into models. Once the routing configuration file is generated, it is loaded onto the DUT (device under test) FPGA using an ATE (Automated Test Equipment) to complete the configuration. Then, the testing phase begins with the ATE applying stimuli to the DUT FPGA and collecting the corresponding responses to compare them with the expected results. By analyzing the comparison results, we can determine if the DUT FPGA has any faults. In this paper, the resource extraction and modeling process will be introduced in Section 3, the automatic wire routing search will be discussed in Section 4, and the utilization of ATE for configuration and testing will be explained in Section 5. Table 1 provides the abbreviations used in this paper along with their respective explanations.

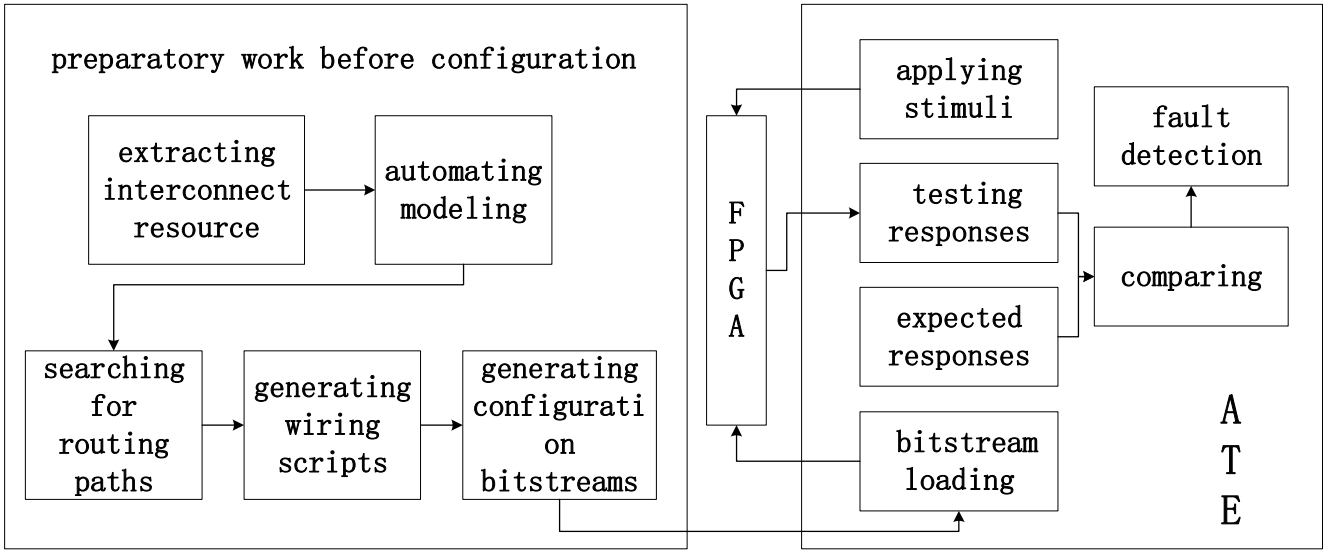

Figure 1. FPGA interconnect resource testing process.

Table 1. Abbreviation table.

Abbreviation	Full Name
FPGA	Field Programmable Gate Arrays
CLB	Configurable Logic Blocks
RAM	Random Access Memory
BIST	Built-in Self Tests
DUT	Device Under Test
ATE	Automated Test Equipment
IOB	Input-output Blocks
IR	Interconnect Resources
BRAM	Block Random Access Memory
LUT	Look Up Tables
EDA	Electronic Design Automation
PIP	Programmable Interconnect Points
BEG	Begin
SRAM	Static Random-access Memory
NMOS	N-Metal-Oxide-Semiconductor

2. FPGA Interconnect Resource Structure and Fault Model Analysis

2.1. FPGA Interconnect Structure Analysis

FPGAs are composed primarily of modules, such as configurable logic blocks (CLBs), input-output blocks (IOBs), interconnect resources (IRs), and block random access memory (BRAM) [9,10]. The CLB is the core resource of the entire FPGA, consisting mainly of programmable resources, such as look-up tables (LUTs), multiplexers, carry chains, and flip-flops, to implement different logical functions. The IOBs form an interface between the chip and the external circuits, providing suitable drive signals to match the differences in the electrical characteristics between the internal and external environments. Using electronic design automation (EDA) software, different electrical standards and physical information can be configured according to certain requirements, such as adjusting the size of the driving currents and changing the resistance values of the pull-up and pull-down resistors. In general, several IOBs are grouped together (banked) for ease of design and management, with the number of IOBs per group varying across different series of FPGA chips.

Interconnect resources can be divided into metal line segments and switch matrix resources based on their connection methods. Metal line segments are distributed mainly between each tile, connecting various logic blocks and hard-core resources. Switch matrix resources include the switch matrices distributed in the wiring channels and those that guide the signals into logic blocks or hard-core resources.

The schematic diagram in Figure 2 shows the switch matrix and CLB structure of an FPGA, which is an important research object with respect to interconnect resources. The metal interconnect lines form the outer perimeter of the switch matrix and are divided into X-fold lines based on the number of switch matrices they traverse. The most common types of metal interconnect lines are single-fold lines, double-fold lines, and long lines. These lines typically start at the BEG port and end at the END port, with a unique mapping relationship between the starting and ending points.

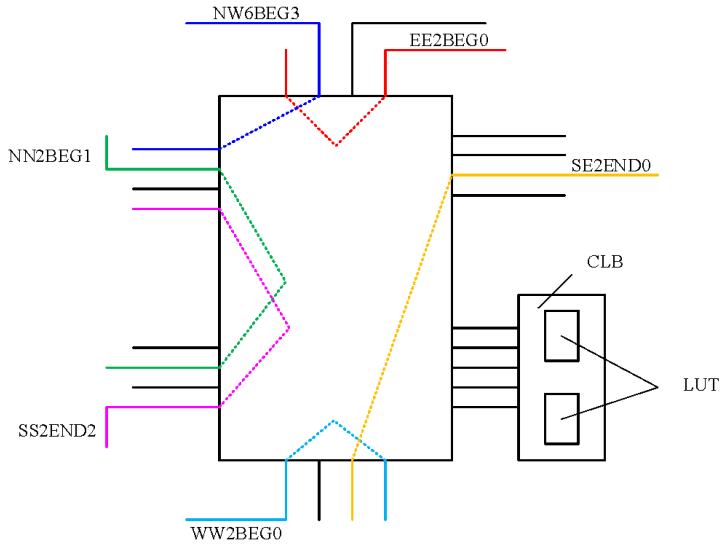

Figure 2. Schematic diagram of FPGA switch matrix and CLB structure.

Within an FPGA, the switch matrices distributed within each tile connect the metal line segments from different directions and wiring channels. The internal connections within the switch matrix are called PIPs (programmable interconnect points), which are controlled primarily by SRAM (static random-access memory) cells to implement transmission gates and buffers in different arrangements. A commonly used example is a switch matrix that uses one buffer and N-many transmission gates to achieve 1-N multiplexing, PIPs start at the END port and end at the BEG port. The mapping relationship of the PIPs is shown in Figure 3.

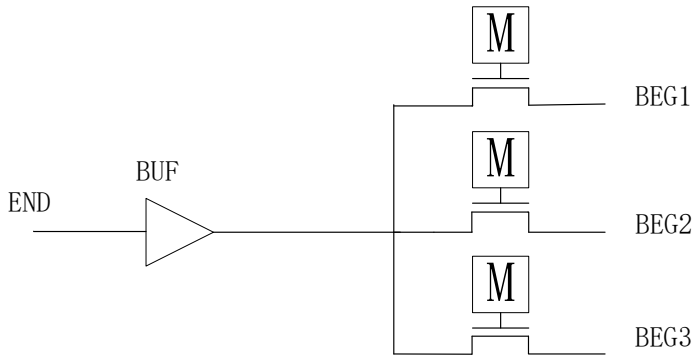

Figure 3. PIP structure and transmission schematic.

2.2. Fault Model Analysis

Interconnect wires within the FPGAs can be divided into two categories: metal segment lines and programmable interconnect points. Each of these interconnect types can experience different types of faults.

Metal line segments may experience open/short faults, bridging faults, and stuck-at faults. An open circuit fault may occur if the metal line segments disconnect during the manufacturing process due to deviations in the process. A short-circuit fault can cause interference to the shorted components during the transmission of the signal. Such faults can cause signal transmissions to the short-connected parts during transmission. Bridging failures may occur if the metal segment line is shorted together with the metal interconnect of an adjacent channel due to process deviations. This type of failure can cause coupling problems during the signal transmission. If the metal interconnect wire is short-circuited with adjacent components due to process deviations, this can result in the signal being fixed at a certain level, such as being shorted to VDD or GND, which is manifested as a fixed 0 fault (SA-0) or a fixed type 1 fault (SA-1).

As shown in Figure 3, the minimum unit of the programmable interconnect points consists of an SRAM and an NMOS (N-Metal-Oxide-Semiconductor) transmission gate, forming a switch that can be controlled by the SRAM. When the value stored by the SRAM is one, the switch is closed, and the NMOS and connection points, END and BEG, are on, enabling signal transmission. Conversely, when the value stored by the SRAM is zero, the switch is open, and NMOS gate is disconnected, and the END and BEG connection points are not conducting, thus preventing signal transmission. Since the probability of failure of the NMOS transmission gate is much less than that of the SRAM, we only discuss the failure of the SRAM.

If the programmed data cannot be written correctly when the SRAM fails, the data stored by the SRAM becomes fixed at 0 or 1. This is known as a stuck-at-0 or stuck-at-1 failure. A stuck-at-0 failure occurs when the programming data cannot be written correctly internally by the SRAM, maintaining a fixed value of 0 that cannot be modified. When this fault occurs, the PIP will normally be open and cannot be programmed to turn on. Conversely, a stuck-at-1 failure occurs when the SRAM cannot write the programming data correctly, resulting in a fixed value of one that cannot be modified. When this fault occurs, the PIP will normally be closed and the PIP will always be on [11–13].

3. Building a Test Model

The focus of modeling is to construct mathematical graph models of the interconnect resources within the FPGAs. In Section 2, we learned that metal interconnect lines are divided into different types of fold lines based on the number of switch matrices they traverse. By observing the driver relationship of the metal interconnect lines in different series of FPGAs, we found that different types of metal interconnect lines exist in different FPGAs, and their drive mapping relationships also differ. Sections of the same type of metal interconnect line and its corresponding port within the same series of FPGAs are grouped together as an X-fold line layer. Thus, a mapping relationship between different fold line layers exists.

By treating each fold line layer as a point on a mathematical graph and the directional driver relationship as edges in the graph, we can abstractly convert the interconnect resources of an FPGA into a directed graph. For example, in Figure 4, the different fold line layers can be interpreted as being self-driven; single layers and double layers or single layers and hex layers can drive each other; double layers can drive long layers; hex layers and long-line layers can drive hex layers; and single layers can drive long layers.

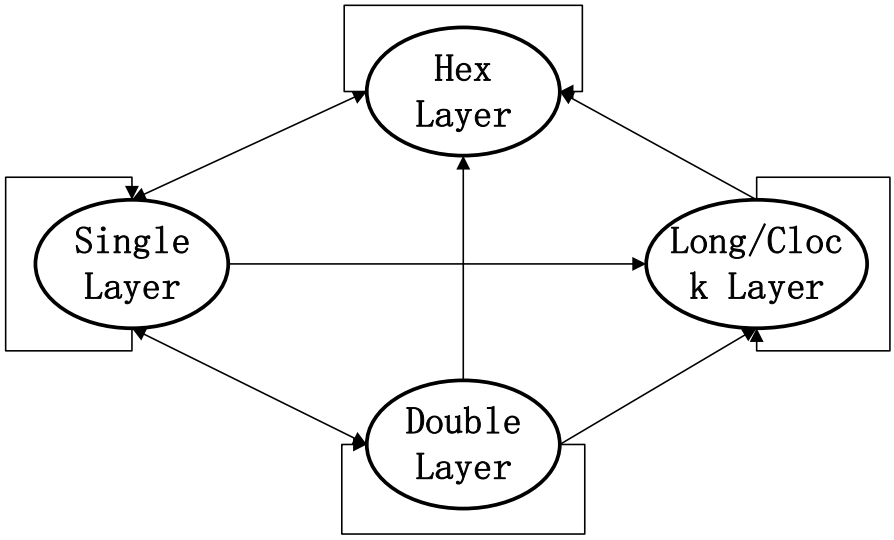

Figure 4. Schematic diagram of line layer driver.

Within each fold line layer, the points on the switch matrix ports, metal interconnect lines, and programmable interconnect lines within the interconnect resources of the FPGA can be treated as points and edges on a mathematical graph. Given the properties of the metal interconnect lines starting at the BEG port and ending at the END port, and the programmable interconnect lines starting at the END port and ending at the BEG port, they can be abstracted into a mathematical graph: G<V,E>. This is demonstrated in Figure 5, where the solid lines represent the driver relationship of the metal line segments, and the dotted lines represent the driver relationship of the programmable interconnect lines.

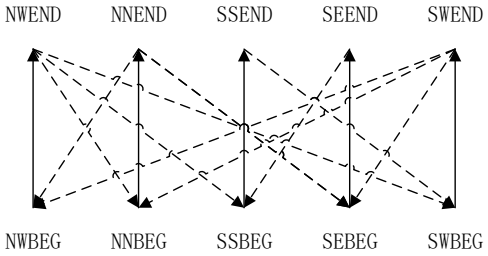

Figure 5. Abstract schematic diagram of interconnecting lines.

The graph G<V,E> can be divided into two subsets, V_{BEG} and V_{END}, where $V_{BEG} \cap V_{END} = \emptyset$ and $V_{BEG} \cup V_{END} = V$. All edge endpoints in graph G belong to either V_{BEG} or V_{END}, and graph G satisfies the definition of a bipartite graph. In this study, we analyze the interconnect relationships of the FPGA's underlying layers by converting them into a bipartite graph. As the connection from the BEG port to the END port is unique and certain, we need to find similar mapping relationships from the END port to the BEG port that form the maximum path during the configuration. The metal interconnect line, M, is a subset of the edges in graph G (excluding the loops), where any two edges in M do not have common vertices. M is a match for graph G, and it represents the maximum number of edges contained in M, saturating all vertices in graph G. Therefore, the metal interconnect line M is a perfect match (maximum match) for graph G. When searching for the maximum path in one configuration, we are also searching for the longest alternating path in graph G.

As the connectivity of the metal interconnect lines is certain, to reduce the search information in the graph model during the actual processing, we treat a metal interconnect line from the BEG port to the END port as a single point on the mathematical graph. We also consider the programmable interconnect lines between the connecting metal interconnect lines as edges in the mathematical graph, as shown in Figure 6a, which represents the actual connection information of the ports, wires, and PIPs in the FPGA interconnect resources. Figure 6b shows the point-edge information that has been batch processed for ease of the wire routing searches.

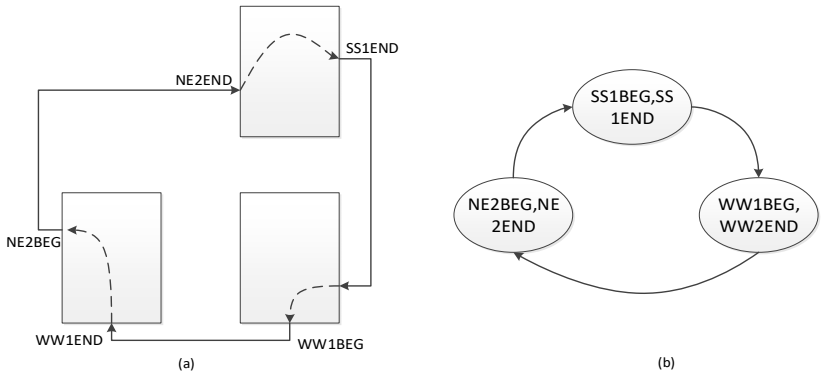

Figure 6. Schematic diagram of a switch matrix model. (**a**) Description of the actual connection information of ports, segment lines, and PIPs in the FPGA interconnect resources; (**b**) description of the dot and edge information after batch processing to facilitate the routing search.

4. Intelligent Routing Method

The purpose of the FPGA test configuration is to find the path through which a test stimulus signal passes. Using the modeling process described above, we converted the physical connection relationships in the FPGA interconnect resources into vertex-edge relationships in a mathematical graph. Intelligent wiring is the process of finding a test path. To satisfy the considerations of high test efficiency and low test cost, we need to search for the longest wiring path in the mathematical graph. In this study, an improved depth-first algorithm was used as the intelligent wiring method, with the goal of reaching the leaf node of the searched structure.

The traversal process of the depth-first algorithm starts at an unvisited vertex as the starting vertex and walks along the edges of the current vertex to another unvisited vertex. When there are no unvisited vertices, we return to the previous vertex and continue to explore other vertices until all vertices have been visited. This algorithm is simple to execute, but its search strategy is incomplete and has some randomness, which may prevent it from producing the optimal solution. Additionally, in a directed graph, when traversing child nodes, it is necessary to backtrack to the parent node, which may require multiple traversals, thus affecting the traversal efficiency. Furthermore, when processing a large number of data, duplicate paths may also exist.

Therefore, this study improves the depth-first algorithm to improve the efficiency of intelligent wiring. The improved algorithm uses memorization during the graph traversal process to record the visited vertices and maintain the longest path value for each vertex. When a previously visited vertex is encountered, the current path is compared with the previously saved longest path value, and the larger value is taken as the new longest path. This improved depth-first algorithm not only reduces the occurrence of duplicate paths but also effectively avoids backtracking in the directed graphs, thereby improving both traversal efficiency and the accuracy of finding the longest wiring path.

The algorithm used in this article retains the recursive idea in the depth-first algorithm. When modeling and parameterizing each point, the out-degree (the number of outgoing edges from a vertex) and the traversal flags are added to the original graph dictionary. In the same type of FPGA, the out-degree of each point may be different, but we set it to 1 for all points. When using this improved depth-first algorithm to find the next traversal point, we choose a point with a higher probability of being routable based on a probability model. If the current state is denoted as s and the next state is denoted as s', at a certain moment in time when the state is s, the probability of transitioning to the subsequent state s' after taking action a is:

$$P^{a}_{ss'} = P[s(t+1) = s' | s(t) = s, A(t) = a] \tag{1}$$

In this model, s represents a point in a mathematical graph, s' represents the adjacent points connected to that point, and action a represents the edges connecting them. In practice, there can be multiple s' corresponding to a given s, meaning that the current state can be connected to many other states. We estimate the probability P by utilizing historical trajectory configurations and employing the maximum likelihood estimation method. The probability P is defined as follows:

$$P^{a}_{ss'} = \frac{\#\text{Selectable path in any state}}{\#\text{Selectable path to all states } S'} \tag{2}$$

where i represents the number of available paths in the current state, that is, the out-degree W of the current point. Therefore, it follows from the properties of the maximum likelihood probability that

$$\sum_{i=1}^{w} P^{a}_{ss'_i} = 1 \tag{3}$$

Comparing the above calculation results, we can find that the point with a higher probability of being routable is the one with a larger out-degree in the next state. This can

also avoid the occurrence of small out-degree points being occupied near the starting point, resulting in missing or disconnected wires in subsequent wiring paths. The improved depth-first algorithm process is shown in Figure 7, where the starting point and ending point are set first. The current state point is set to start, and its adjacent points are the next state points. The point with the largest out-degree is searched and added to the wiring path, and it becomes the new starting point until the ending point is found. When a point is added to the traversal path, its new out-degree becomes

$$W_i' = W_i - 1 \tag{4}$$

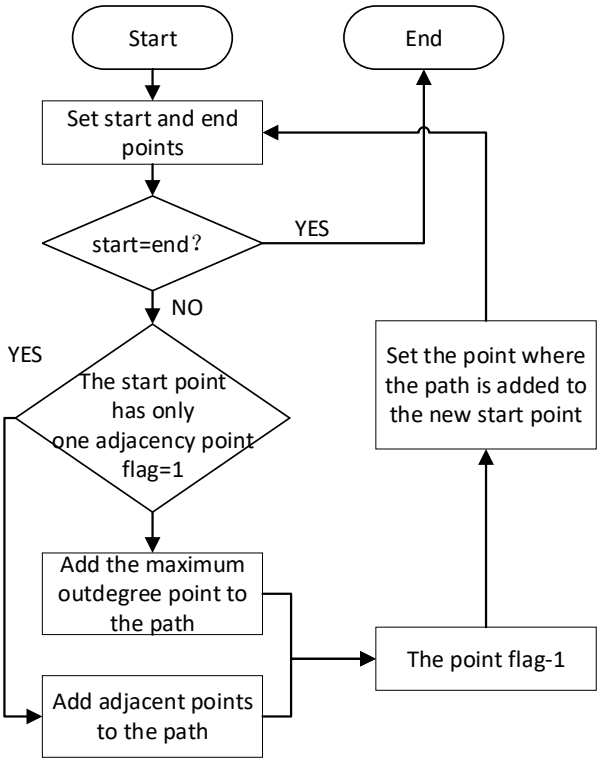

Figure 7. Algorithm process.

During traversal, the principle that any points with an out-degree of 0 cannot be added to traversal paths is always observed. Additionally, the method sets the $flag_i$ of all points to 1 for marking because each point can only be traversed once in a single routing process. The traversed points' $flag_i$ values are reduced by 1 to become 0, while $flag_i$ = 1 marks the untraversed points. Only the untraversed points can be selected as the next passing point. In summary, the following principles should be followed when implementing intelligent wiring based on the idea of depth-first algorithms: (1) points that have already been traversed cannot be accessed; (2) points with an out-degree of 0 cannot be traversed; (3) if there are multiple adjacencies, the point with the largest out-degree is prioritized; and (4) the process continues until the ending point is reached.

5. FPGA Configuration Method and Test Implementation

5.1. FPGA Configuration

In the earlier discussion, we described how we layered the internal wiring resources of the FPGA into a directed graph and used graph theory to batch the different types of connections to achieve higher coverage. We selected the shortest subgraph in the directed graph, G, as a type of configuration circuit, G_i, in the Euler diagram. We then selected the new configuration type diagram, G-G_i, as the new graph G until no such subgraph existed in Figure G, recording the number of configuration types as C. In each configuration type, C_i, all its port points were converted into a two-part diagram. Due to the limitations of the

metal cables, to achieve for 100% coverage in this configuration type, it must reach W_i a number of times, which is the out-degree of the largest point of the out-degree in all ports. Therefore, the total number of configurations is determined as follows:

$$Configuration\ times = \mathrm{C} \times \max(W_i) \tag{5}$$

In addition, if 100% coverage is pursued, a new configuration type can be generated by adding edges to the last graph, G-G_i, to form an Euler graph.

5.2. CLB Configuration Method

The CLB is the main logic resource for implementing the sequential and combinational circuits, and each CLB in the Xilinx's Kintex-7 series FPGA contains two SLICEs. Each SLICE contains a look-up table, registers, carry chains, flip-flops, multiplexers, and other components. In interconnection resource testing, we mainly use the look-up table and the flip-flop in the SLICE [14,15].

In the Kintex-7 series FPGA, the logic function generator is implemented as a six-input look-up table, with inputs A1~A6 and outputs O5 and O6. During wiring, the LUTs in the CLBs are connected in series between the interconnection lines to detect faults such as open circuits and bridges. By configuring the LUT to perform logic operations between input signals, such as O5 = A1&A2&A3&A4&A5&A6, the output port of the LUT can be used to monitor its output by connecting it to the trigger in the CLB where the LUT is located. The response value of the trigger is obtained through bitstream readback, and then the output response of the LUT can be obtained. In this way, by configuring the logic operation of the LUT and entering the different excitations into it, the faults can be located, and their types can be analyzed. By setting the LUT to six inputs and one output, the port of the switch matrix can be fully used to improve the coverage of the first configuration.

5.3. FPGA Interconnect Resource Test Implementation

This study employs ATE (Automatic Test Equipment) V93000 to test the XC7K325T as the verification object. After the automatic routing process is completed, but before ATE testing, the XDL script generated by the code auto-routing must be converted into an NCD test graphic file [16]. Figure 8 shows an example of an NCD test graphic, where the black rectangular boxes in the background represent the switch matrix, and the blue lines indicate the paths after automatic routing. The test graphic requires the generation of a binary configuration file in the FPGA development environment for sorting and then transforms it into a pattern that the ATE machine can recognize. The pattern file is subsequently loaded onto the ATE machine for online configuration. Figure 9 depicts the DUT (device under test) board, where the central part is a socket for inserting the FPGA chip. The pins of the FPGA chip are connected one-to-one with the testing channels of the ATE through fan-out. The ATE machine is capable of providing power to the chip and applying test vectors to each I/O pin. The device under test generates corresponding signals, which are then transmitted to the ATE. By comparing the test response with the test graphic, the quality of the chip can be determined.

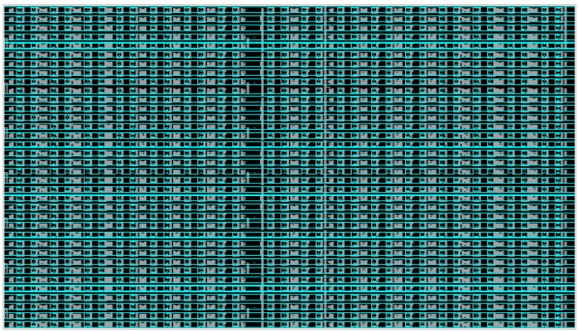

Figure 8. NCD test graphical interface.

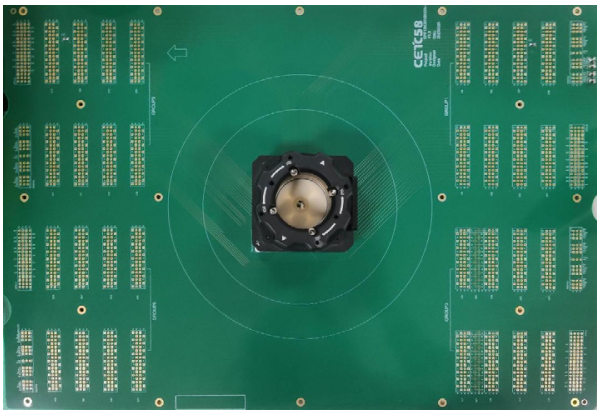

Figure 9. DUT test board.

6. Discussion and Conclusions

The automated modeling and intelligent wiring methods proposed in this article were successfully applied to the connection resource test development of AMD Xilinx's XC7K325T FPGA, achieving the automatic generation of interconnect resource tests. Table 2 compares the modeling method of the interconnecting metal segment lines as a point on a mathematical graph with the traditional modeling method. The original point information is successfully compressed by nearly half, as shown in Equation (6), where 363 represents the number of points to be processed for a single switch matrix in traditional modeling, and 202 is the number of points to be processed for a single switch matrix after modeling optimization. N represents the total number of switch matrices inside the FPGA. Compared with the traditional modeling method, this method compresses the modeling space and reduces the search points when searching the routing path. Consequently, this method is faster and does not require a separate calculation of the next traversal coordinates of the current traversal.

$$\text{Compression ratio} = \frac{(363 - 202) * N}{363 * N} = 44.35\% \tag{6}$$

Table 2. Modeling optimization comparison.

	Before Optimization	After Optimization
Single switch matrix math graph scale (points)	363	202

Compared with the theory used in [17], the work reported in this paper adopts an improved depth optimization algorithm for searching the wiring path that is more intelligent and automatic. It does not need to observe the direction of the path in each switch matrix in advance or plan the wiring according to the usual laws. Table 3 shows the comparison results for the wiring of the sextuple line using the theory used in [9,18]. Due to the vast and complex nature of FPGA interconnect resources, multiple path searches and configurations are required to achieve high coverage. The "Times" column in Table 3 represents the number of configurations, while the "Pre-planned" column indicates whether advance planning is needed before searching the path, including finding wiring patterns or planning the direction of wiring according to a predetermined plan. There are various types of linear wires between the interconnection relationships of the six-fold line. Although the coverage rate of the running results can reach 100%, there are too many operations. In summary, the improved depth-first algorithm discussed in this paper achieves a high coverage without requiring advanced observation and planning, making it suitable for FPGA interconnect resource testing.

Table 3. Comparison of hex line routing results.

Configuration Type	Times	Coverage	Pre-Planned
[1]	7	100%	Y
[4]	29	100%	N
This paper	9	100%	N

Author Contributions: W.X. proposed the research topic, designed the research plan and specific implementation process, participated in the writing of the paper, and proposed the revision plan; W.Q. participated in the discussion and specific implementation of the research plan and the writing of the papers and participated in revisions; X.L. is an experienced FPGA test engineer who guided specific cabling operations and ATE machine operations; and H.W. provided the data and technical support and guidance for some modules during the implementation of the project and proposed revisions to the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The code is available at https://github.com/wenjingqi7/resources (github.com) (accessed on 1 June 2023).

Acknowledgments: We would like to thank Houjun Wang and Yijiu Zhao for their academic guidance and valuable comments on the paper. We also thank Wenjing Qi for the methods proposed for key technologies in the project and for her help in writing the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ruan, A.; Yang, J.; Wan, L.; Jie, B.; Tian, Z. Insight into a Generic Interconnect Resource Model for Xilinx Virtex and Spartan Series FPGAs. *IEEE Trans. Circuits Syst. II Express Briefs* **2013**, *60*, 801–805.
2. Toutounchi, S.; Lai, A. FPGA test and coverage. In Proceedings of the International Test Conference, Baltimore, MD, USA, 7–10 October 2002; pp. 599–607.
3. Liao, Y.; Ruan, A.; Wang, Y.; Xiang, C.; Wang, L.; Huang, H.; Zhu, J. Interconnect resources testing and faults diagnosis in field programmable gate arrays. In Proceedings of the IEEE 2011 10th International Conference on Electronic Measurement & Instruments, Chengdu, China, 16–19 August 2011; pp. 185–189.
4. Niamat, M.Y.; Nambiar, R.; Jamali, M.M. A BIST scheme for testing the interconnects of SRAM-based FPGAs. In Proceedings of the The 2002 45th Midwest Symposium on Circuits and Systems, 2002. MWSCAS-2002, Tulsa, OK, USA, 4–7 August 2002; p. II-41.
5. Stroud, C.; Nall, J.; Lashinsky, M.; Abramovici, M. BIST-based diagnosis of FPGA interconnect. In Proceedings of the International Test Conference, Baltimore, MD, USA, 7–10 October 2002; pp. 618–627.
6. Hsu, C.-L.; Chen, T.H. Built-in Self-Test Design for Fault Detection and Fault Diagnosis in SRAM-Based FPGA. *IEEE Trans. Instrum. Meas.* **2009**, *58*, 2300–2315.
7. Bai, R.; Guo, H.; Wang, Z.; Zhang, Y.; Zhang, F.; Chen, L. FPGA Interconnect Resources Test Based on A Improved Ford-Fulkerson Algorithm. In Proceedings of the 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 14–16 December 2018; pp. 251–258.
8. Dai, L.; Liu, Z.-B.; Liang, S.-C.; Yang, M.; Wang, L.-L. FPGA interconnect testing algorithm based on routing-resource graph. In Proceedings of the 2008 9th International Conference on Solid-State and Integrated-Circuit Technology, Beijing, Chian, 20–23 October 2008; pp. 2087–2090.
9. Zhang, F.; Guo, C.; Zhang, S.; Chen, L.; Li, X.; Sun, H.; Meng, Y.; Chen, Q. Research on Hex Programmable Interconnect Points Test in Island-Style FPGA. *Electronics* **2020**, *9*, 2177. [CrossRef]
10. Banik, S.; Roy, S.; Sen, B. An integrated framework for application independent testing of fpga interconnect. *J. Electron. Test.* **2019**, *35*, 729–740. [CrossRef]
11. Zhao, J.; Feng, J.; Lin, T.; Tong, Z. A novel FPGA manufacture oriented interconnect fault test. In Proceedings of the 9th International Conference on Solid-State and Integrated-Circuit Technology, Beijing, China, 20–23 October 2008; pp. 2091–2094.
12. Kumar, T.N.; Inn, C.S. An automated approach for the diagnosis of multiple faults in FPGA interconnects. In Proceedings of the 2009 1st Asia Symposium on Quality Electronic Design, Kuala Lumpur, Malaysia, 15–16 July 2009; pp. 391–395.
13. Ruan, A.; Tian, W.; Ni, B.; Wu, K. A hierarchical switch matrix and interconnect resources test in Virtex-5 FPGA. In Proceedings of the 2014 International Symposium on Integrated Circuits (ISIC), Singapore, 10–12 December 2014; pp. 111–114.

14. Sun, X.; Alimohammad, A.; Trouborst, P. Modeling of FPGA local/global interconnect resources and derivation of minimal test configurations. In Proceedings of the 17th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2002. DFT 2002. Proceedings., Vancouver, BC, Canada, 6–8 November 2002; pp. 284–292.
15. Xilinx. *The Programmable Logic Data Book*; Xilinx: San Jose, CA, USA, 2002.
16. Wang, X.-F.; Si, S.-H.; Gao, C.; Huang, J. A method of FPGA interconnect resources testing by using XDL-based configuration. In Proceedings of the 2014 Prognostics and System Health Management Conference (PHM-2014 Hunan), Zhangjiajie, China, 24–27 August 2014; pp. 203–207.
17. Banik, S.; Roy, S.; Sen, B. Test Configuration Generation for Different FPGA Architectures for Application Independent Testing. In Proceedings of the 2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID), Delhi, India, 5–9 January 2019; pp. 395–400.
18. Liu, P.; Xu, N.; Hui, F. An Automatic Method for Testing of FPGA Routing Resource. In Proceedings of the 2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Xi'an, China, 25–27 May 2018; pp. 918–923. [CrossRef]