Research Article

FPGA Delay-Oriented Process Mapping Algorithm of Xiangxi Minority Based on LUT

Yun Xiao , Wei Zeng, and Huang Zhang

Institute of Art & Design, Changsha University of Science & Technology, Changsha 410000, Hunan, China

Correspondence should be addressed to Yun Xiao; xiaoy@csust.edu.cn

Received 28 October 2021; Revised 10 December 2021; Accepted 23 December 2021; Published 7 February 2022

Academic Editor: Sang-Bing Tsai

Copyright © 2022 Yun Xiao et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

At present, FPGA (field-programmable gate array) architecture has made great progress in the requirements of hardware volume, which can meet common needs. However, for the increasing number of resources, it is difficult to significantly reduce the delay of process mapping. Therefore, this paper proposes FDMAP (fit descending map) algorithm from the perspective of the LUT number to reduce the delay. This paper proposes a method of FPGA mapping and debugging for heterogeneous multicore high-performance processors based on isomorphic symmetric FPGA architecture, which effectively utilizes the architectural features of heterogeneous multicore processors and the symmetric features of isomorphic FPGA, divides FPGA functions from top to bottom in a hierarchical way, and constructs FPGA architecture from bottom to top. Using differential bridge and adaptive delay adjustment sampling technology, combined with the embedded virtual logic analyzer debugging tool, FPGA architecture can be lightened and deployed quickly. Multicore complementary core-to-core replacement simulation mapping methods such as debug shells can be used to effectively complete the mapping of the target's high-performance heterogeneous multicore processor to the entire SOC (system on-chip) chip system-level FPGA. In the aspect of algorithm, the fdmap algorithm is mainly implemented, and the low latency mapping of resources is realized with FPGA architecture. In order to verify the effectiveness of mapping the fdmap algorithm, this paper compares the fdmap algorithm with the vector VM algorithm. The research shows that when the wavelength resolution is 7 pm and the temperature error is less than 1°C, the shell is debugged, and 10 mapping examples are simulated with the fdmap algorithm. In the experiment, the LUT with the most critical 20% is selected, and the closed value of the LUT search type is set to 0.86. Compared with the original data, the number of LUTs increased by 15.2%, and the criticality decreased by 35.21%. Compared with the vector VM algorithm with the biggest gap, the number of LUTs decreased by 14.25%, the criticality improved by 14.21%, and the overall delay decreased by 65%. Therefore, the isomorphic symmetric FPGA architecture proposed in this paper can improve the structural criticality and significantly reduce the latency while reducing the number of LUTs.

1. Introduction

At present, FPGA process mapping and mapping methods have been successfully applied in processor mapping of different architectures. In order to make up for the deficiency of traditional analog mapping, Intel has adopted FPGA architecture in design mapping and performance analysis of each generation of processor. The domestic Loongson team has completed the FPGA mapping of Loongson-2g by using multi-FPGA process mapping. These FPGA process maps mainly focus on the prototype mapping of single architecture and single processor core on FPGA, but how to build FPGA architecture and debug FPGA

mapping methods for heterogeneous multicore high-performance processor chips with multigrain (die) is not involved.

At present, many scholars use intelligent algorithms such as particle swarm optimization to solve the problems of national process load balancing and delay optimization under multiresource constraints. For example, Luo R. defined the national process mapping problem as a multidimensional process mapping problem, proposed a reordering grouping genetic algorithm (RGGA), and used the real mapping to understand the advantages of the RGGA [1]. Wang P. abstracted the national process configuration problem as minimizing the total resource loss and compared

it with the traditional single objective process mapping algorithm, which proves the effectiveness of the algorithm [2]. Among them, Bowles deimproved the algorithm that depends on a single resource type, improved the national process integration problem to a multidimensional process mapping problem that supports multiple resource types, and solved the problem through the ACO algorithm [3]. Lewis D. proposed the vector VM algorithm that is based on the vector computing theory and proposed a static national process placement algorithm, dynamic national process placement algorithm, and load balancing algorithm [4]. Menasri W. depended on the analysis of the historical load data of the data center, combined with the real-time VM resource monitoring, and used the migration mechanism of national technology to prevent the overload of physical LUT-based FPGA [5].

In terms of algorithm, Jia J. proposed an enhanced FFD algorithm based on the first fit decreasing (FFD) algorithm. On the basis of dynamic monitoring of LUT-based FPGA resource utilization, a dynamic migration strategy is used to achieve a lower delay of LUT-based FPGA [6]. Kamali et al. proposed an algorithm that applies growing self-organizing feature graphs to reinforcement learning and realized the best representation of state space through two growing self-organizing maps [7]. Kumar et al., according to the characteristics of self-organizing growth map (GSOM) and binocular stereo vision parallax theory, solved the problem of determining the topological structure of the SOM network in advance by constructing the topological relationship of the spatial environment [8]. Venieris and Christos-Savvas proposed a synchronous positioning and process mapping construction model based on local view and pose recognition with a rat navigation LUT unit consisting of view cells and experience process mapping and a SLAM navigation strategy, significantly reducing FPGA communication latency [9]. The above research has made a great contribution to the process of process mapping and the improvement of hardware performance from the perspective of a static algorithm and delay reduction. However, there is little research on the topology of mapping at present, and the difficulty is that the consistency of topology and physical resource scheduling is difficult to be solved.

This paper presents a method of FPGA mapping and debugging for heterogeneous multicore high-performance processors based on a heterogeneous symmetric FPGA architecture. It effectively utilizes the architecture characteristics of heterogeneous multicore processors and the symmetry characteristics similar to FPGA. Speed bridge and adaptive delay are used to adjust the sampling technology combined with the debugging tool of the embedded virtual logic analyzer to quickly complete the construction and deployment of FPGA architecture. Multicore completion, intercore replacement, simulation debugging, shell, and other mapping methods can be used to effectively complete the system-level FPGA mapping of a high-performance heterogeneous multicore processor and the whole SOC chip. In the aspect of algorithm, we mainly implement the fdmap algorithm and combine it with FPGA architecture to achieve low latency resource mapping.

2. FPGA Architecture and Process Mapping Algorithm

2.1. FPGA Architecture. With the development of processor architecture, high-performance heterogeneous multicore processors are emerging. Because the design of a high-performance heterogeneous multicore processor is very complex, in order to reduce the design risk, shorten the mapping cycle, carry out software development ahead of time, and reproduce the problems after silicon, it usually needs the prototype mapping architecture of FPGA (field-programmable gate array) and carries out a variety of hardware and software collaborative mapping and debugging works based on FPGA architecture [10]. The proposed FPGA debugging and mapping method for heterogeneous multicore high-performance processors based on isomorphic FPGA architecture effectively utilizes the architecture characteristics of heterogeneous multicore processors and the symmetry characteristics of isomorphic FPGA, divides FPGA from top to bottom in a hierarchical way, and constructs FPGA architecture from bottom to top [11]. Combined with differential bridge, adaptive delay adjustment, embedded virtual logic analyzer (VLA), and other technologies, the FPGA architecture can be quickly brought up and deployed [12]. The proposed methods, such as multicore complementation, intercore replacement, and simulation debugging shell, can quickly and completely map the target high-performance heterogeneous multicore processor to FPGA [13]. Through FPGAs, in recent years, with the continuous expansion of the processor application field, the functional requirements of processors to handle complex scenarios in real-world applications have increased. The traditional general-purpose processors for general-purpose computing scenarios cannot meet these requirements, so heterogeneous multicore processors for complex applications are emerging [14]. For high-performance heterogeneous multicore processors, the overall design scale is very large because of its many core groups, complex internal architecture, diverse intercore communication, and rich high-speed peripheral interfaces. As a result, the state space to be mapped has grown exponentially, and the complete function mapping has become a bottleneck in the design of heterogeneous multicore processors [15]. The traditional method based on software simulation mapping is flexible and easy to use. However, with the increase of the logic unit size of a heterogeneous multicore processor, the speed of full-chip system-level simulation mapping decreases significantly [16]. Although the transaction-based mapping method can improve the abstract level of the mapping object and accelerate the mapping speed, for the target design to be mapped in this paper, its running speed can only reach tens of hertz (Hz), or even a few hertz, and sometimes there are software simulation EDA (Electronic Design). It is impossible to run a large number of extensive system-level test programs, not to mention the complex mixed mapping of software and hardware [17]. The simulation mapping based on hardware simulation accelerator can run at several hundred kilohertz (kHz), even to megahertz (MHz) after optimization. However, it has high requirements for the

operating environment, high maintenance cost, and high price [18]. On the other hand, although its running speed is hundreds to thousands of times faster than the simulation speed, it still cannot meet the requirements of the operating system, compiler, system-level software application and performance, pressure test program, and so on. FPGA can implement most of the functions of the processor design, and other analog custom circuits cannot be implemented on FPGA, which can carry out high-speed prototype mapping [19]. Because its running speed can reach tens to hundreds of megabytes, it can run more and more real test programs and system software and often find many problems that are difficult to find in software simulation mapping [20]. In addition, simulation mapping and hardware simulation can only connect test models or virtual devices, which can make a difference between the mapping environment and the behavior of real physical devices. However, the architecture is widely used in system-level software development, testing, and processor development mapping, and performance analysis plays a very important role [21].

2.2. Process Mapping Transformation Algorithm. Firstly, graph $G(V, E)$ is defined as a directed acyclic Boolean graph. For the process mapping of Xiangxi Ethnic Minorities, this paper constructs a delay-oriented process mapping transformation algorithm and uses the Gaussian function to express the neighborhood constraint:

$$G(V, E) = \ln\left(\frac{V_{it}}{FI_{it} - 1}\right) \quad (1)$$

$$= \alpha + \beta \ln FI_{it} - 1 + v_i + \mathfrak{F}_i * S - X,$$

where V and FI are the positions of the output unit and the winning unit, respectively, and S is the neighborhood range. It represents the connection weight from the input unit to the output unit. The value of each connection weight is called weight. Here, the Euclidean distance is used to judge the weight of the corresponding connection weight. The calculation formula is as follows:

$$\psi = \sum_{x=1}^{\theta} Vx \quad (2)$$

$$= \sum_{x=1}^{\theta} \left(\frac{Wx}{\sum_{i=1}^n W_{\mathfrak{F}}} \frac{Sx}{G(V, E)} \right).$$

Here, W is the input of the unit and X is the weighted unit to unit connection weight. Using the minimum similarity measure to determine whether the winning unit is the best matching unit, update the connection weights of all units in the winning unit. The update rules are as follows:

$$F - G(V, E) = \frac{n \sum_{i=1}^n \sum_{j=1}^n w_{ij} (x_i - \bar{x})(x_j - \bar{x})}{\sum_{i=1}^n \sum_{j=1}^n w_{ij} (x_i - \bar{x})^2} \quad (3)$$

$$= \frac{n \sum_{i=1}^n \sum_{i \neq j}^n w_{ij} (x_i - \bar{x})(x_j - \bar{x})}{S^2 \sum_{i=1}^n \sum_{j=1}^n w_{ij}},$$

where F is the neighborhood and X is the learning factor. The update rule means that this method can generate typical connection weights represented by the winner and adjacent weights corresponding to the winner. When all the input samples are satisfied, the training is finished, which is the given error. Otherwise, repeat the steps until all the training samples are finished.

$$\frac{G(V, E)}{F} = \frac{\sum_{j=1}^k \sum_{h=1}^k \sum_{t=1}^{n_j} \sum_{r=1}^{n_h} |y_{ij} - y_{hr}|}{2n^2 u}. \quad (4)$$

The environment landmark is established in the process mapping, and the discharge model of the position LUT unit is mapped into the two-dimensional space of the process mapping. The discharge rate of the position LUT unit after position coding is obtained through landmark recognition. The discharge rate of the moving process map at the second position in the real space is as follows:

$$P(u_i, w_j) = P(d_i)P(w_j|d_i); P(w_j|d_i) \quad (5)$$

$$= \sum_{k=1}^K P(w_j|z_k)P(z_k|d_i),$$

$$LUT = \frac{P}{\pi}, \quad u_h \leq u_j \leq \dots \leq u_k, \quad (6)$$

where D is the real position of the position LUT unit in space and u is the reference position of the position LUT unit. The excitation degree of the LUT unit has a strong position selectivity, and in a specific position, the position LUT unit presents the maximum discharge rate [22]. The cognitive path of process mapping in space can be obtained by connecting multiple corresponding positions; that is, the position LUT unit can encode the position information of the current corresponding position, and the position information encoded by multiple position LUT units can encode the real space trajectory of process mapping. Based on this, this paper proposes a SLAM algorithm combining GSOM neural network and location LUT unit, as shown in Figure 1, where a is the LUT cell mapping relation and b is the LUT cell mapping transformation.

The environment information mapped by GSOM is associated with the location information represented by the location LUT unit, and the algorithm is applied to the VP-SLAM model to establish the FPGA physical model:

$$G_{jh} = \frac{\sum_{Z=1}^{h_j} \sum_{r=1}^{n_h} |y_{ji} - y_{hr}|}{n_j n_h (u_j + u_h)}. \quad (7)$$

Through the vision sensor mounted on the process mapping architecture, the environment image is captured in the process of process mapping roaming. After image processing, the position coordinate information is obtained and added to the GSOM network dataset. The training samples are input to the improved GSOM neural network for competitive learning, and the winning LUT unit associated with the discharge rate of position LUT unit is output

FIGURE 1: LUT mapping transformation and mapping relationship.

[23]. If the position error between the winner's LUT unit and the GSOM network dataset is less than a certain threshold, the position information encoded by the current LUT unit will match the process mapping position. Otherwise, the coordinates of the current process mapping position are recorded and added to the GSOM network dataset:

$$T_k = \frac{2k}{k+1} + \left[\frac{1}{2} + \frac{1}{2k} \right] \left[\frac{c_2 - c_1}{3} \right]^2 + \frac{2(c_2 - c_1)}{3}. \quad (8)$$

Let the number of all national processes in the data center be K , t denotes any virtual machine, and JH denotes the specific resource requirements of virtual machine VI . Let the number of physical LUT-based FPGAs in the data center be D , where C represents any physical LUT-based FPGA in the data center. For heterogeneous physical LUT-based FPGAs with different hardware parameters, the available resource vectors can be expressed as

$$d_{jh} = \frac{\int_0^\infty dF_j(y) \int_0^y (y-x) dF_h(x)}{VI}, \quad (9)$$

$$\sigma_t = \frac{\sqrt{(1/n) \sum_{i=1}^n (FI_{it} - FI_{it})^2}}{FI_{it}}, \quad (10)$$

where d is the number of resource types. When $d=0$, it means that the LUT-based FPGA is in the state of energy saving (such as sleep, shutdown, or low-frequency operation), and when $d=1$, it means that the LUT-based FPGA is in the state of operation [24]. Suppose that when a virtual machine is mapped to a physical LUT-based FPGA, the mapping relationship between all virtual machines in the data center and the physical LUT-based FPGA can be represented by a matrix R :

$$l_{ssim} = \frac{1}{\sqrt{R}} - \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \quad (11)$$

$$R = \frac{\alpha}{1 - \beta}. \quad (12)$$

For the virtual machine mapped on physical LUT-based FPGA, the column vector in the virtual machine mapping matrix D can be used as follows:

$$\begin{aligned} \frac{1}{\sqrt{R}} &= \theta \\ &= -\frac{1}{T} \ln(1 + \beta). \end{aligned} \quad (13)$$

The value of R is the number of virtual machines mapped on the LUT unit, and t indicates that the FPGA based on LUT is idle and has the condition to switch to the energy-saving state. The mapping matrix between the virtual machine and physical LUT-based FPGA is proved:

$$\ln\left(\frac{FI_{it}}{FI_{it} - 1}\right) = \alpha + \beta \ln FI_{it} - 1 + \phi X_{it} - 1 + v_i + \tau_t. \quad (14)$$

It represents the mapping of any virtual machine to all physical LUT-based FPGAs in the data center. According to the definition, VI represents being mapped to physical LUT-based FPGAs. Since any virtual machine can only be mapped to one physical LUT-based FPGAs at the same time, there are cases where equation (15) holds for all rows of each LUT cell:

$$\theta(p, q) = \arctan\left(\frac{L(p, q+1) - L(p, q-1)}{L(p+1, q) - L(p-1, q)}\right). \quad (15)$$

The network delay between any two virtual machines is defined in the virtual machine intranet communication matrix:

$$G_t = \sum_{j=2}^k \sum_{h=1}^{j-1} G_{jh} (p_j s_h + p_h s_j) D_{jh} (1 - D_{jh}), \quad (16)$$

where G is the total amount of data transmission of the virtual machine in the whole life cycle of the virtual machine and D is the total amount of data transmission of the virtual machine. However, for the physical data center, the optimization of LUT-based FPGA resource utilization or delay efficiency makes the number of LUT-based FPGAs to be activated minimum. That is, under the premise of meeting the resource constraints, the virtual machine mapping matrix M is satisfied

$$k_{t1}[i] = \sum_j \cos(w_i^1, w_j^2). \quad (17)$$

Minimize the number of rows for the condition. The state of LUT-based FPGA can be obtained by the physical LUT-based FPGA state vector k . The LUT-based FPGA delay optimization problem can be expressed as

$$h_t = \tanh(w_c x_t + u_c (r_t \ominus h_{t-1}) + b_c), \quad (18)$$

where h is the sum of all virtual machine resource requirements mapped to any physical LUT-based FPGA, which cannot be greater than the corresponding host LUT-based FPGA resources. In theory, the network delay optimization of the data center is to find out a set of optimized virtual machine mapping functions, that is, to reduce the network transmission within the data center by optimizing the mapping of the virtual machine, not only to reduce the data transmission between LUT-based FPGAs but also to optimize the network transmission path, that is, to reduce the number of switches or routers that need to pass in the process of network transmission number [25]. Based on the virtual machine delay matrix, find virtual machines that are in high demand for network communication and map these virtual machines to the same physical LUT-based FPGA as much as possible or between multiple LUT-based FPGAs in the same switch. In this way, most of the internal delay can be controlled by the same physical LUT-based FPGA, and the total amount of network transmission to the switch or router can be reduced as much as possible. When the network traffic inside the data center is reduced, the number of network devices (switches, routers, etc.) that need to be activated will be reduced, and the idle network devices will be adjusted to the state of energy saving so as to achieve the purpose of energy saving and emission reduction. The network optimization problem of data center can be expressed as

$$h_t = z_t \ominus h_{t-1} + (1 - z_t) \ominus h_t, \quad (19)$$

where Z represents the internal data center and the data transmission amount between the physical LUT-based FPGA nodes represents the total data transmission amount between the virtual machine and the external data center, which cannot be optimized by the mapping of the virtual machine. HT represents the resource constraints in the process of virtual machine mapping. There are many researches on how to achieve the purpose of energy saving by shutting down the network equipment in the data center. In

the vertical structure, the most commonly used method is to use the greedy algorithm to select a left link to meet the demand; in the horizontal structure, the previous random selection strategy is improved to the principle of from left to right. When the algorithm is completed, it can calculate an optimized subset of network devices that meet the current demand and have a small number of devices, and the network devices without delay, such as routers or switches, can be adjusted to the energy-saving state. On the basis of network optimization, that is, on the basis of equation (4), the delay optimization problem of network equipment can be expressed as

$$\ln HY_{it} = a_0 + a_1 du * dt + \sum_{i=1}^N b_j Xu + \varepsilon_u, \quad (20)$$

where HY represents the static delay of network equipment and the process mapping problem is a simple dynamic programming problem. However, in the process of mapping virtual machines to LUT-based FPGA, the resource constraints need to be considered; that is, the total resource requirements of all virtual machines mapped to any LUT-based FPGA cannot exceed the physical resource constraints of the LUT-based FPGA. Therefore, the mapping of the virtual machine to LUT-based FPGA in this paper belongs to the process mapping problem under multiresource constraints and the NP-hard problem. Through the mapping of the virtual machine to minimize the amount of data transmission in the data center network, this problem can be abstracted as a quadratic assignment QAP problem, and the solution of the QAP problem is also NP-hard. There is a great similarity between the shortest path problem and the combinatorial optimization problem, which makes it possible to solve combinatorial optimization problems based on the ACO algorithm.

In order to describe the fdmap algorithm conveniently, let $V(x)$ denote the probability that ants choose to map virtual machine VI to physical LUT-based FPGA. When using LUT-based FPGA delay optimization and network resource optimization, the heuristic information (visibility) of virtual machine VI is mapped to physical LUT-based FPGA. In the initial state of the algorithm, the ant K starts to construct the solution of the problem step by step. In this process, each iteration is represented by the time scale T . Because the ant K must first satisfy the physical resource h constraint in the process of constructing the solution, the rule that the ant chooses the virtual machine to put into the LUT-based FPGA can be expressed as follows:

$$V(x) = \frac{1}{Nh} \sum_{i=1}^N k \left(\frac{X_i - x}{h} \right), \quad (21)$$

where Nh is a random variable; when the generated random variable meets the conditions, the maximum value in the formula is obtained; and VI is mapped to DJ, where parameter a represents the enhancement factor of pheromone and G represents the enhanced visibility. In all unmapped virtual machine sets, all virtual machine sets can be put into (meet the resource constraints) LUT-based FPGA, that is, to meet the constraints of the conditions. Among them, the

updating of pheromone is determined by the following formula:

$$G = G_w + G_{nb} + G_t, \quad (22)$$

$$k(\delta) = \frac{(1/\sqrt{2\pi})G(V, E)}{(-(\pi^2/2))}. \quad (23)$$

The parameter k is the number of ants determined at the initialization of the algorithm, and E is the volatilization coefficient of pheromone. When the execution is constrained by physical resources, the delay of FPGA is the minimum.

3. Research and Design of FPGA Architecture

3.1. Methods. This paper proposes a method of FPGA mapping and debugging for heterogeneous multicore high-performance processors based on isomorphic symmetric FPGA architecture, which effectively utilizes the architectural features of heterogeneous multicore processors and the symmetric features of isomorphic FPGA, divides FPGA functions from top to bottom in a hierarchical way, and constructs FPGA architecture from bottom to top. The FPGA architecture can be lightened and deployed quickly by using speed bridge and adaptive delay adjustment sampling technology in combination with an embedded virtual logic analyzer (VLA) debugging tool. Using the mapping methods of multicore complementary, intercore replacement simulation, such as debug shell, can effectively complete the target high-performance heterogeneous multicore processor and the whole SOC chip system-level FPGA mapping. In the aspect of algorithm, the fdmap algorithm is mainly implemented, and the low latency mapping of resources is realized with FPGA architecture.

3.2. Design. In this paper, based on the first fit decreasing (FFD) algorithm, we propose an enhanced FFD algorithm of the first decreasing map (fdmap). On the basis of dynamic monitoring of LUT-based FPGA resource utilization, we use a dynamic migration strategy to achieve lower latency of LUT-based FPGA. In order to simplify and unify the experimental standard, it is assumed that all LUT-based FPGAs in the data center have the same configuration, and the resource requirements of VM proposed by users are expressed as the percentage of the total resource capacity of physical LUT-based FPGAs. That is, for the resource vector of the virtual machine, the percentage is used to represent the demand value of physical resources. In order to simulate the real data center operation scene, the value of the resource vector is set to be randomly generated in a certain value range.

3.2.1. Algorithm and Parameter Setting. In this experiment, the fdmap algorithm uses CloudSim as the simulation architecture, written in Java language. In order to map the effectiveness of the fdmap algorithm, this paper compares the fdmap algorithm with MDBP-ACO, vector VM, and PMOC algorithm. The virtual machine integration problem

is improved to a multidimensional process mapping problem supporting multiple resource types, and the problem is solved by the ACO algorithm. Compared with the vector VM algorithm, this paper proposes the static virtual machine placement algorithm, dynamic virtual machine placement algorithm, and load balancing algorithm based on vector computing theory. Based on the analysis of historical load data of the data center, combined with real-time VM resource monitoring, the migration mechanism of the virtual machine is used to prevent the overload of physical LUT-based FPGA. Under the condition of the same scale of virtual machines, when mapping to a smaller number of LUT-based FPGAs, it achieves more efficient resource and delay utilization; that is, the larger the number of virtual machines (me) in a single LUT-based FPGA, the higher the efficiency. Based on the experiment of process mapping, the influence of the LUT cell discharge model on the positioning accuracy of process mapping is studied by giving LUT cell interval r in different positions. The simulation parameters are set as follows: the motion space is $10\text{ m} \times 10\text{ m}$, the positioning period is 2 s, the positioning direction is arbitrary, the process mapping keeps uniform motion in each positioning period, and the operation speed of process mapping in different positioning periods is 0.5~2 M/s. Under the same training time, the influence of interval r on the positioning effect of process mapping is analyzed.

3.2.2. Delay Comparison Design under Different Algorithms. In this section, we do not consider the problem of network delay optimization, and we only consider the delay optimization of FPGA based on LUT. The setting of different algorithm parameters has a great influence on the calculation results and the convergence of the algorithm. In this experiment, the number of ants selected for each iteration is 20, and the maximum number of cycles to find the solution is NC_Max is 100, other ACO parameters are $a = 1$, 2, and volatile factor is 0.2. Let the size of LUT-based FPGA in the data center be 400, and the algorithm needs to map 1000 virtual machines. The experiment is divided into three groups: ABC Group and ACO Group. Among the three groups of experiments, the average requirements of LUT-based FPGA resources for virtual machines are set at 10%, 15%, and 20%, respectively. In order to adapt to multi-resource requirements, we simulate five resource requirements of CPU, memory, disk, I/O throughput, and network; that is, we set the dimension of the resource vector of the virtual machine as 5. For LUT-based FPGA delay, the idle LUT-based FPGA delay is 150W, and the full LUT-based FPGA delay is 220W. Each experiment was run 10 times, and the results of each experiment were averaged.

4. Results and Discussion

4.1. FPGA Simulation Features and Mapping Test Analysis. As shown in Figure 2, due to the comprehensive consideration of the needs of multidimensional resources, fdmap can achieve higher performance than MDBP-ACO,

FIGURE 2: FPGA process design simulation results.

vector VM, EFFD, and PMOC under the same conditions. That is, under the premise of meeting the resource constraints, more national processes can be mapped into the same LUT-based FPGA, thus reducing the number of activated LUT-based FPGAs and improving the utilization of delay.

The network traffic of the data center is randomly simulated by the communication matrix G and the communication vector g of the national process intranet, which unifies the life cycle of all national processes into one hour and reduces the delay of process mapping circuit structure. In terms of resource utilization, the average idle rate of the five algorithms in the three groups of ABC experiments is shown in Table 1. Compared with MDBP-ACO, vector VM, EFFD, and PMOC, fdmap performs better in terms of resource utilization. This is because fdmap considers more resource matching dimensions and can reduce resource waste in the case of more complementary resources.

As shown in Figure 3, the total network delay generated by the random algorithm is the largest, and the VPTCA algorithm achieves a better optimization effect than the fdmap algorithm; this is because the VPTCA algorithm only optimizes the network of the data center, while the fdmap algorithm not only considers the optimization of network resources but also integrates the comprehensive utilization of CPU, disk, memory, and other resources. Network optimization is not as effective as the VPTCA algorithm, but overall delay optimization is better than the VPTCA algorithm. Comparing the total average delays of the three LUT-based FPGAs, the overall performance of the VPTCA network optimization is better than the fdmap algorithm, but in the total average delay optimization of the LUT-based FPGAs, the fdmap algorithm is

VPTCA and random. You can see that it is better than the algorithm.

As shown in Figure 4, the generated FPGA simulation uses 10 LUT units to represent 1100 environment sample points, which is very close to the distribution structure of samples while maintaining the topological order of sample data. These core groups are connected to the internal data and control bus of the chip, so they can simulate and replace each other between heterogeneous cores. At an earlier stage, cross-mapping between other high-speed peripheral modules or processor cores can be carried out with fewer resources, which not only saves FPGA resources but also simplifies the architecture and greatly speeds up the prototype mapping test process. In addition, the coordination of size and core is also very helpful to the location of the problem elimination method. When the silicon chip is debugged, it can also be used to reproduce the problems of various modules, especially those of high-speed peripherals.

As shown in Figure 5, in the same scene, the improved model can achieve closed-loop detection faster, correct the process mapping pose, and map the robustness and rapidity of the improved VP-SLAM model in the experimental scene. Compared with the original VP-SLAM model, the improved VP-SLAM model is closer to the actual path of process mapping, which effectively improves the accuracy of process mapping.

As shown in Figure 6, the maximum error of the FBG high-temperature sensor demodulation system in this paper is 0.071% in the high-temperature environment measurement at 800°C. The wavelength resolution of the demodulation system is less than 7 pm, and the temperature resolution is less than 1°C. The adaptive delay circuit is

TABLE 1: Virtual machine intranet communication delay.

Item	VM	VP-SLAM	Resources	MDBP-ACO	Vector VM	Delay
LUT	3.43	1.76	1.75	1.42	3.39	2.78
Random	3.74	2.83	4.16	2.83	4.73	4.46
NC_MAX	1.64	2.9	2.45	4.81	3.14	2.96
FPGA	1.07	2.66	4.13	4.24	1.69	3.37
EFFD	2.02	1.16	6.53	6.38	1.51	4
Process mapping	5.7	2.28	4.74	1.73	2.86	6.8

FIGURE 3: Process mapping circuit structure for reducing delay.

FIGURE 4: Generated FPGA simulation environment sample points.

embedded in the receiver (Rx), and the received signal delay is corrected by the register. In this design, according to the delay demand, 800 MHz high-frequency clock is used to control the sampling delay, each delay step is 1.25 ns, a total of 4B control, and the delay interval is 0–20 ns, which can cover the whole period of the interface transmission signal, as shown in Table 2.

Compare the measurement results of the sensor demodulation system at different temperatures with the theoretical analysis results, as shown in Table 1. In the environment of wavelength resolution of 7 pm and temperature error less than 1°C, the shell is debugged. Ten mapping examples are simulated by the fdmap algorithm. In the experiment, the LUT with the most critical degree of 20%

FIGURE 5: Closed-loop inspection process mapping pose.

FIGURE 6: High-temperature sensor demodulation system.

TABLE 2: Measurement results and theoretical analysis results.

Item	MDBP-ACO	Vector VM	EFFD	PMOC	FBG	InGaAs
FPGA	3.26	5.53	4.02	3.66	5.69	3.28
Delay	4.24	3.3	3.05	2.26	4.36	4.85
Process mapping	4.98	1.35	3.12	4.46	4.09	3.24
Virtual machine	5.97	2.24	5.2	2	5.12	1.06
VP-SLAM	2.74	2.22	5.15	4.15	6.83	5.03
Resources	5.22	2.16	6.33	3.82	2.39	5.37

is selected, and the closed value of the LUT search type is set to 0.86. Compared with the original data, the number of LUTs increased by 15.2%, and the criticality decreased by 35.21%. Compared with the vector VM algorithm with the

biggest gap, the number of LUTs decreased by 14.25%, the criticality improved by 14.21%, and the overall delay decreased by 65%. As a part of the engineering development mode, after the RISC processor boot program is completed,

you can choose whether to enter the debugging shell. Once the debugging shell is turned on, you can test other functional modules or processor cores in a more flexible and intuitive way. In addition, debugging is a very important shell that can be embedded in any kind of processor core because it does not depend on lib and OS, which is very flexible to use. In addition to managing test routines, it also provides a bare chip running environment for test routines.

The simulation characteristics of FPGA are shown in Figure 7. Heterogeneous multicore chips adopt multigrain packaging architecture. It is necessary to combine the multigrains in Figure 1 for multigrain interconnection mapping and performance analysis. On the other hand, the efficiency is low. In this white paper, taking the 4-grain interconnect as an example, you need to complete the entire process of partitioning, compositing, laying out, wiring, producing, and downloading files for up to 48 FPGAs. It takes more than a week to iterate once. In view of this, if we can take advantage of the functional symmetry between grains and the regular interconnection of FPGAs, this work can be greatly simplified.

As shown in Figure 8, the scale of the processor to be mapped has been reduced by a large part, but it is still very large, with complex internal structure and a wide variety of high-speed interfaces, so it is necessary to implement FPGA mapping and hardware/software code development and debugging are still facing many challenges. The main challenges are that the target heterogeneous multicore processor architecture has a complex internal interconnection, large logic scale, and many high-speed IO. How to map to multiple FPGAs and meet the requirements of target chips in different scale configurations, such as module level/system level, single core/multicore, single module/multimodule, single crystal/multigrain, and full chip, is a great challenge to light up and implement effective mapping quickly.

As shown in Figure 8, the scale of the processor to be mapped has been reduced by a large part, but it is still very large, with a complex internal structure and a wide variety of high-speed interfaces, so it is necessary to implement FPGA mapping and hardware/software code development and debugging are still facing many challenges. The main challenges are that the target heterogeneous multicore processor architecture has a complex internal interconnection, large logic scale, and many high-speed IO. How to map to multiple FPGAs and meet the requirements of target chips in different scale configurations, such as module level/system level, single core/multicore, single module/multimodule, single crystal/multigrain, and full chip, is a great challenge to light up and implement effective mapping quickly. As shown in Figure 9, compared with traditional analog mapping and hardware accelerator mapping, FPGA process mapping has worse signal observability and controllability, difficult debugging, and long iteration cycle. How to design a modular, flexible, and easy-to-use virtual logic analyzer (VLA) is an urgent issue to solve. The target heterogeneous multicore processor is different from the traditional homologous multicore processor. How to

perform heterogeneous cores, software task partitions, load balancing, and coordination between system scheduling algorithms and how to effectively use heterogeneous multicore architectures for more effective and faster mapping are also controversial. Finally, the FPGA architecture construction, debugging, hardware and software comapping of the heterogeneous multicore processor, and the recurrence of postsilicon problems on the FPGA architecture are completed.

As shown in Figure 10, since the logic functions inside the grains are completely consistent, if it is not necessary to consider the power-on sequence and signal clock synchronization between grains (how to solve the power-on sequence and clock sampling synchronization will be described later), it is only necessary to change the FPGA. During the system startup, the firmware (FW) will select different initialization execution paths according to the master-slave status of the current grain. After the master-slave grain completes the data path establishment, the firmware of the master grain can complete the initialization of other slave grains' configuration.

As shown in Table 3, the traditional method of grabbing signals based on FPGA's own tools is often limited by FPGA's internal memory capacity, so it is impossible to observe many signals for a long time (signal depth). Another method is to connect the signal to the pin, with the help of a logic analyzer and other debugging equipment, but also limited by the available pin limit and sampling depth; only some key signals can be selected. For the target mapping chip, its internal data bus has a large bit width and needs to observe a lot of signals, so the number of debugging pins using FPGA IO pins is often insufficient. On the other hand, in the process of multicore processor chip debugging, due to the heterogeneous multiprocessor architecture superimposing multicore and multithread, branch prediction, speculative execution, and other complex technologies, the problem shows certain randomness. Sometimes the test can find the problem within a few hours; sometimes, it takes dozens of hours to encounter the problem, which requires recording the multicore data of multiple processor cores a long-running sequence of instructions (PC values) for each thread to further analyze and locate the problem.

4.2. Discussion. The FPGA architecture of this paper selects a unified architecture and the same type of FPGA so that all grains can use the same FPGA configuration file. In this way, only 12 single grains of FPGA architecture can be completed, and then multigrains of interconnected deployment can be supported at the same time, which greatly improves the work efficiency and shortens the debugging time. How to synchronize the power-on sequence of multigrain (multi-FPGA architecture) and the interconnected signal and clock is also a problem to be solved. According to the requirements of interconnection protocol and sampling between dies, an adaptive and software configurable delay control mechanism is introduced to

FIGURE 7: FPGA process design simulation features.

FIGURE 8: Processor size to be mapped.

realize the accurate interconnection and signal synchronization of interface sampling. When the received signal meets the expectation, the control state machine finishes the delay correction and informs the sender to stop sending the training sequence. In the boot sequence, the firmware polls the control register values for software and hardware synchronization to ensure that the data sampling synchronization between the FPGA architectures is complete. In the next step, the software controls the normal data transfer between the FPGA architecture interfaces. Due to the target mapping, the multicore processor of multigrain

FPGA integrates DDR4 memory controller, pcie4.0, sata3.0, usb3.1, 10-gigabit Ethernet, and many other high-speed IO peripheral interfaces. In order to map their logical correctness, these high-speed IO interfaces need to be implemented on the FPGA architecture. According to the standard protocol, the real speed of high-speed interfaces such as pcie4.0 and sata3.0 is fast, and the working frequency of FPGA is very high. After the interconnection of the target chip, the internal control logic of the chip needs to span multiple FPGA boards and multiple FPGA chips, and its frequency has been difficult to meet the full rate

FIGURE 9: Traditional analog mapping and hardware accelerator mapping.

TABLE 3: Pin limit and sampling depth.

Item	FPGA	Delay	Process mapping	Virtual machine	VP-SLAM
Vector VM	3.8	1.73	2.79	1	1.06
EFFD	4.22	4.95	3.14	2.99	2.95
PMOC	2.03	1.48	5.11	2.14	4.78
FBG	2.07	4.5	1.9	2.81	2.71
InGaAs	6.79	5.15	6.07	6.31	5.6
Resources	5.61	6.3	5.87	1.24	5.61

FIGURE 10: Internal logic function of the die.

requirements such as pcie4.0, which requires a differential bridge to match the rate between FPGA and real interface, so that pcie4.0, sata3.0, and other high-speed interfaces can be realized in the FPGA architecture in the form of low speed.

5. Conclusions

This paper proposes a fdmap algorithm based on the FFD algorithm and, on the basis of dynamic monitoring of LUT-based FPGA resource utilization, uses a dynamic migration strategy to

achieve lower latency of LUT-based FPGA. In order to simplify and unify the experimental standard, it is assumed that all LUT-based FPGAs in the data center have the same configuration, and the resource requirements of VM proposed by users are expressed as the percentage of the total resource capacity of physical LUT-based FPGAs. That is, for the resource vector of the virtual machine, the percentage is used to represent the demand value of physical resources. In order to simulate the real data center operation scene, the value of the resource vector is set to be randomly generated in a certain value range. Compared with vector VM, which has the biggest gap, this algorithm can reduce the number of LUTs by 14.25%, improve the criticality by 14.21%, and reduce the overall delay by 65%. Therefore, the isomorphic symmetric FPGA architecture proposed in this paper can improve the structural criticality and significantly reduce the latency while reducing the number of LUTs.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors state that this paper has no conflicts of interest.

Acknowledgments

This work was supported by the 13th Five-Year Plan of Educational Science in Hunan Province, "Research on Art Design Education in Colleges and Universities from the Perspective of Fender," ND206234.

References

- [1] R. Luo, X. Chen, and Y. Ha, "A routing algorithm for fpgas with time-multiplexed interconnects," *Journal of Semiconductors*, vol. 41, no. 2, pp. 405–409, 2020.
- [2] P. Wang, X. Zhang, Z. Liu, L. Xu, J. He, and J. Xu, "FPGA implementation of adaptive time delay estimation for real-time near-field electromagnetic ranging," *International Journal of Circuit Theory and Applications*, vol. 46, no. 11, pp. 1940–1952, 2018.
- [3] D. E. Bowles and L. R. Gardiner, "Supporting process improvements with process mapping and system dynamics," *International Journal of Productivity and Performance Management*, vol. 67, no. 8, pp. 12–16, 2018.
- [4] D. Lewis and H. Schmit, "Spatial timing analysis with exact propagation of delay and application to Fpga performance," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 11, pp. 2153–2166, 2019.
- [5] W. Menasri, A. Skoudarli, and A. Belhadj, "FPGA implementation of variable-bins hevcc cabac decoder with path delay optimization," *IET Image Processing*, vol. 13, no. 6, pp. 56–59, 2019.
- [6] J. Jia, J. Wang, and J. Zhao, "Design of parallel wave-front restoration algorithm based on Fpga," *Optik*, vol. 176, no. 4, pp. 168–174, 2018.
- [7] H. M. Kamali, K. Z. Azar, and K. Gaj, "Lut-lock: a novel lut-based logic obfuscation for Fpga-bitstream and asic-hardware protection," in *Proceedings of the 2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, vol. 5, no. 1, pp. 405–410, Hong Kong, China, July 2018.
- [8] B. Kumar, N. Khare, and P. K. Chaturvedi, "FPGA-based design of advanced bms implementing soc/soh estimators," *Microelectronics Reliability*, vol. 84, no. 3, pp. 66–74, 2018.
- [9] S. I. Venieris and B. Christos-Savvas, "FPGAconvnet: mapping regular and irregular convolutional neural networks on FPGAs," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 4, pp. 326–342, 2018.
- [10] Z. Ghaderi, N. Bagherzadeh, and A. Albaqsmi, "Stable: stress-aware boolean matching to mitigate bti-induced snm reduction in sram-based FPGAs," *IEEE Transactions on Computers*, vol. 67, no. 1, pp. 102–114, 2018.
- [11] I. Ahmed, L. L. Shen, and V. Betz, "Optimizing FPGA logic circuitry for variable voltage supplies," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 99, pp. 1–14, 2020.
- [12] X. Sun, C. J. Xue, and J. Yu, "Accelerating data filtering for database using FPGA," *Journal of Systems Architecture*, vol. 114, no. 11, pp. 901–908, 2020.
- [13] O. Urso, F. Chiacchio, L. Compagno, and D. D'Urso, "An RFID application for the process mapping automation," *Procedia Manufacturing*, vol. 42, no. 4, pp. 8–15, 2020.
- [14] M. R. Khalilov and A. V. Timofeev, "Optimization of MPI-process mapping for clusters with angara interconnect," *Lobachevskii Journal of Mathematics*, vol. 39, no. 9, pp. 1188–1198, 2018.
- [15] K. Bonawitz, Z. Finer, and M. Heisler, "Workflow process mapping: a novel approach to promote implementation of evidence based practices," *Obstetrics & Gynecology*, vol. 135, no. 4, pp. 102–103, 2020.
- [16] Y. Kali, M. Saad, and K. Benjelloun, "Super-twisting algorithm with time delay estimation for uncertain robot manipulators," *Nonlinear Dynamics*, vol. 93, no. 10, pp. 84–89, 2018.
- [17] R. Arcentales-Carrion, E. C. Morles, and D. Sucozhanyay, "Process mapping and modeling: a theoretical tool analysis," *Test Engineering and Management*, vol. 83, Article ID 25914, 2020.
- [18] G. S. Slick, C. M. Davis Iii, J. C. Elfar, and L. E. Nikkel, "Process mapping total knee arthroplasty: a comparison of instrument designs," *The Journal of Arthroplasty*, vol. 36, no. 3, pp. 98–99, 2020.
- [19] C. Hu, Y. Liu, and F. Tao, "Minimum transmission delay algorithm for multi-hop nodes in Internet of Things based on symmetric algorithm," *Journal of Ambient Intelligence and Humanized Computing*, vol. 4, no. 12, pp. 1–10, 2020.
- [20] A. Atitallah, B. Saïda, and K. Abderrahim, "Joint parameter and time-delay identification algorithm and its convergence analysis for wiener time-delay systems," *Circuits, Systems, and Signal Processing*, vol. 39, no. 1, pp. 199–222, 2020.
- [21] J. Harris and I. C. Elliott, "Improving employee engagement and distributed leadership through lean systems process mapping," *British Journal of Healthcare Management*, vol. 26, no. 12, pp. 75–76, 2020.
- [22] B. Yan, L. Xiao, and G. Qin, "QTMS: a quadratic time complexity topology-aware process mapping method for large-scale parallel applications on shared hpc system," *Parallel Computing*, vol. 95, no. 4, pp. 634–637, 2020.
- [23] B. Aja and C. Mna, "Process mapping in industry - the self-centred phenomenon and how it effects continuous improvements," *Procedia CIRP*, vol. 93, no. 4, pp. 718–723, 2020.
- [24] S. Geng, S. Liu, Z. Fang, and S. Gao, "An optimal delay routing algorithm considering delay variation in the LEO satellite communication network," *Computer Networks*, vol. 173, no. 7, Article ID 107166, 2020.
- [25] H. Nashaat, O. Refaat, and F. W. Zaki, "Dragonfly-based joint delay/energy lte downlink scheduling algorithm," *IEEE Access*, vol. 7, no. 9, pp. 1–12, 2020.