2010 10th IEEE International Conference on Computer and Information Technology (CIT 2010)

Design and Implementation of a Heterogeneous High-performance Computing Framework using Dynamic and Partial Reconfigurable FPGAs

Xingjun Zhang, Yanfei Ding, Yiyuan Huang and Xiaoshe Dong
School of Electronic and Information Engineering
Xi'an Jiaotong University, Xi'an 710049, China
Email: {xjzhang, xsdong}@mail.xjtu.edu.cn

***Abstract*—Integrating reconfigurable computing with high-performance computing, exploiting reconfigurable hardware with their advantages to make up for the inadequacy of the existing high-performance computers had gradually become the high-performance computing solutions and trends. Based on comprehensively investigating the reconfigurable technologies, the paper presented a high-performance computing scheme in which the general-purpose processing nodes are connected to the dynamic partial reconfigurable computing nodes through the high-speed network. Using module-based partial reconfiguration design method, a FPGA based dynamic and partial reconfigurable computing node is designed. This node has the ability to do dynamic and partial reconfiguration and can load different computing units according to the different requirements. Dynamic partial reconfigurable computing node integrated microprocessor, memory, network interface, reconfigurable computing module, interface module are designed and implemented. The experimental results show that the system can achieve more functions with fewer resources; and the reconfigurable computing node can nicely complete the task and the system performance is effectively improved.**

I. Introduction

With the high performance computing application expanding, the traditional computing mode using the general purpose processors can not meet the challenges of high availability. The multi-framework integrated heterogeneous computing becomes a choice. Reconfigurable computing based on FPGA (Field Programmable Gate Array) can accelerate the high performance computing [1][2], those systems have the potential to exploit coarse-grain functional parallelism through conventional parallel processing, while exploiting fine-grain parallelism through direct hardware execution on the FPGA. And also, due to the hardware programmability, which allows changing the hardware to fit the underlying problem, HPRC (High-Performance Reconfigurable Computers) systems have shown orders of magnitude improvement in performance, power, size and cost over conventional High-Performance Computers (HPCs) [3]. However, the HPRC systems [4][5] have not yet been universally used; they are still the costly devices for the most users. Integrating FPGA-based dynamic and partial reconfigurable computing with the traditional high-performance computing nodes to construct the high performance computing framework is the effective method to resolve this problem.

Usually a familiar high-performance reconfigurable computing system is a dynamic and global reconfigurable computing system, aim at one or several integrated reconfigurable logic devices each time, configuration file need to cover entire logic devices, it cost a lot of time, moreover, it need peripheral equipment to control the execution of reconfiguration, increase the complicacy of control. This paper designs dynamic partial reconfigurable computing nodes that can integrate with the traditional high-performance computing nodes, based on FPGA and reconfigurable computing technology. The dynamic partial reconfigurable computing nodes have ability of dynamic partial reconfiguration, which is for partial logic, the configuration file is small, the time of reconfiguration is short, moreover, it's self-reconfigurable, it need not the control of peripheral equipment. Reconfigurable computing nodes that is designed integrate many function modules as microprocessor, memory, network interface, partial reconfigurable computing module, reconfigurable computing module interface and so on, it can be used to do the general-purpose processing, as well as load different computing units according to the different requirements.

II. System Architecture

The system consists of general-purpose processing nodes and dynamic partial reconfigurable nodes, which communicate with each other through the high speed interconnection network and support cooperative work with the master-slave mode. General-purpose processing nodes which mainly process some tasks that are not suitable for reconfigurable hardware dealing with, assign computing tasks on dynamic partial reconfigurable nodes. And dynamic partial reconfigurable nodes can dynamic load different hardware configuration data to change the hardware structure on the basic of different computing needs of general-purpose processing nodes to mainly provide dynamic allocation. Its main idea is to use the same hardware resources to provide different computing functions under the time-sharing; it develops advantage of hardware acceleration well while makes full use of limited hardware resources.

General-purpose processing nodes are connected with reconfigurable computing nodes through the high-speed interconnection network, visit reconfigurable computing nodes

978-0-7695-4108-2/10 $26.00 © 2010 IEEE
DOI 10.1109/CIT.2010.401

through remote access interface library, and distribute tasks to reconfigurable computing nodes. Reconfigurable computing nodes perform calculations and return results. Remote Access Interface library provide uniform access interface for the application, including the request of computing tasks, the sending of the calculated data, the reception of computing result and so on. Basic system function module is shown in figure 1.

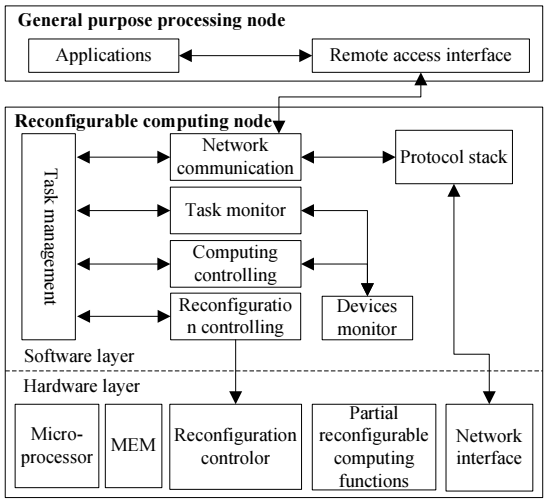

Fig. 1. System function module architecture

In reconfigurable computing nodes, partial reconfiguration computing (PRC) is one of the main functions provided by the system, and different computing services can be dynamically configured according to application requirements. This process is mainly realized by hardware logic so that the speed of execution is very high. Moreover, reconfigurable computing nodes must also have some capacity to complete the general processing operations, such as task management, task monitoring, network communications, computing control, reconfigurable control and so on. Therefore, the PRC nodes need also provide some devices including micro-processing, memory, network interface etc to run some simple software applications. The task management module responds in terms of the computing tasks delivered by the network communication module and monitors the local configuration of the computing function, then determines whether or not to call the reconfigurable module to update the configure.

III. General Processing Node Software

System software consists of two parts: first, the reconfiguration and control service programs which are running in the reconfigurable computing nodes provide computing services for the general-purpose processing nodes; second, the application running on the general processing nodes call the library of interface, through which application can access to the reconfigurable computing nodes. If you want to achieve hardware acceleration for some calculation function in application, only need to use a particular function call, which completes calculating function accelerated by accessing reconfigurable computing nodes based on the remote access interface library.

The general-purpose processing nodes can access the reconfigurable computing nodes through the remote access interface library which mainly transforms the user's specific computing tasks into the tasks that can be computed in reconfigurable computing node and interacts with reconfigurable computing nodes through the network, interface library modules.

Access library provides uniform access interfaces for the user application program which can call the computing functions from the uniform access interface by the instructions and the necessary parameters. For example, the following library function is for the fixed-point matrix multiplication:

$int\ mmm\_fix(int\ dimension,\ void * addr\_a,\ void * addr\_b,\ void * addr\_c)$

The first parameter is the number of matrix dimensions, the rest three parameters are the addresses of the two original matrixes and the address of the result matrix respectively.

IV. Reconfigurable Computing Node Design

A. Hardware Structure of the Reconfigurable Computing Node

Module-based partial configurable design method [6] is used in the paper, and the hardware structure of the dynamic reconfigurable computing node need to solve five problems: partitioning the system static and dynamic module, selecting the static module function units, designing the reconfigurable interface module, designing the basic structure of the system, and designing the reconfigurable computing unit. Reconfigurable computing node function modules partition is shown in figure 2.

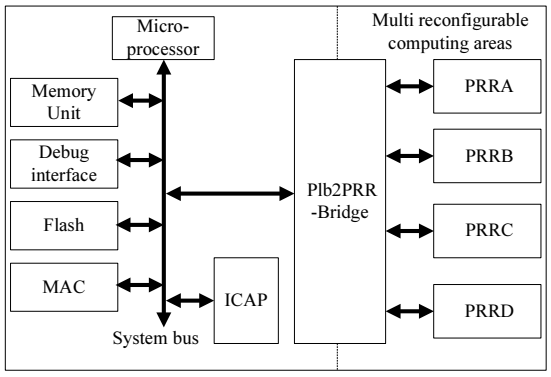

Fig. 2. Reconfigurable computing node function modules

Static logic part does not change during the execution of the mission and is used to implement the uninterrupted task. It is usually non-core and controls larger computing tasks which is not suitable for hardware implementation, such as network communication, task scheduling, etc. Therefore the hardware mainly include: 1) microprocessor unit, used to implement reconfigurable computing nodes which needed for a variety of software. The function of the software is protocol processing and various procedures controlling; 2)

Network Media Access Control (MAC) units, providing network communication interface, dealing with general-purpose computing nodes of the communication;3) memory cell, used the on-chip BRAM to achieve high-speed storage and on-board SDRAM, Flash memory cell, stored code execution, data and configuration files; 4) reconfiguration control unit can be embedded with the FPGA's internal configuration access port (ICAP) to achieve access to Virtex family chips, and access configuration memory of reconfigurable logic device through the ICAP.

Reconfigurable logic parts, also the dynamic part, can implement different computing functions for different hardware architecture according to computing different configurations of mission request. A variety of computing functions require the user to achieve independent design. This part is one of the main bodies of design and implementation of the node. Bus and reconfigurable computing unit bridge (e.g. Plb2PRR_Bridge) achieve the communication of static logic and reconfigurable logic also is one of the main bodies of node design. It mainly includes bus communication interface, device control logic, and user interface logic of three parts.

B. Reconfigurable Computing Node Software

Node software consists of two parts: the first one is the service program designed and runs on the reconfigurable computing nodes to provide computing services for general-purpose processing nodes. The second one is the interface library that runs on general-purpose processing nodes and provides the way that the application accesses to reconfigurable computing nodes. Both communicate each other through high speed network and are the client/server mode. Reconfigurable computing nodes are as service-side and general processing nodes as client-side. The design of on-chip system software on the reconfigurable computing nodes has two ways. The one way uses a real-time operating system and the other doesn't use any operating system. The former can provide multi-task scheduling, memory management, interrupt management, file systems, system functions and can be used to implement complex designs, however, it need a more intensive system resources. The latter is designed simply and uses fewer resources. In order to save the on-chip resources, this design features a non-operating mode [7]. In this mode, the system software on the reconfigurable computing nodes mainly consists of board support packages (BSP) and service procedures. The software hierarchy is shown in Figure 7. Board Support Package is located at the lowest level of software systems and provides support for the service program. Service program making use of the communication protocol stack and various functions provided by the board support package achieves specific functions, including the main control module, task monitoring and management module, configuration module and calculation control module.

BSP contains the necessary support system software, including boot code, device drivers and processor-related support library. Boot code guides the implementation of the application, device driver code includes various devices on the system operation functions and interrupt handler.

C. Implementation of Dynamic Partial Reconfiguration

Dynamic partial reconfigurable computing nodes can dynamically configure the computing functions according to the different computing demands. We designed three computing functions, namely AES (Advanced Encryption Standard) Encryption, AES decryption, and fixed-point matrix multiplication. We choose AES algorithm as an example to introduce the design and implementation of calculating function below.

AES is the Symmetric Key Block Cipher Algorithm Standard adopted by NIST. Belgium research achievement Rijndael is selected as implementation basis for AES in October 2000. The algorithm is so simple that it is easy to implement. The packet length and the key length for block cipher is alterable, the length of both a packet and a key can be independently set to 128 bits or 192 bits or 256 bits. In this paper, the packet and key length for AES are set 128-bit, and the 128-bit plaintext is divided into 16-byte packets which are organized into a 4x4 matrix by order. This matrix is called state, based on which running the AES. The AES algorithm is formed with iteratively transformation, each transformation includes four different basic operations in the composition, namely the byte substitution (SubByte), line shift (ShiftRows), the column confused (MixColumns), round keys plus (AddRoundKey).

We have adopted a modular design approach to divide the calculating functions into several sub-modules, and the basic module structure of the basic Encryption calculation module unit is shown in figure 8. Memory access module achieve access control logic of system memory by encryption calculation function units; cache module achieve local high-speed memory space of computing function unit, and provide high-speed cache for calculation module; and encryption calculation modules achieve encryption calculation. Master control module control the entire computing process and turn a large number of encryption calculations into multiple basic encryption calculations.

The master control module controls the implementation of the entire encryption calculation tasks through three basic functions, one is to control memory access module reading and writing system memory, the second is to control encryption calculation module implementing encryption calculation and the last one is to control interrupt handling.

The encryption calculation sub-module of the encryption computing function unit can finish 512 pieces of operand at most at once. The specific operand is controlled by the signal C2P_N, so the main working of the master control module is dividing a mass of encryption calculation assignment to several encryption calculation assignment which is 512 pieces and controlling the computing process of the encryption calculation module, at the same time, reading and writing in memory before and after computing, reading the calculation data into cache and writing the calculation results back to memory.

The basic structure of decryption calculation function module is similar to encryption calculation function module; the different is key expansion and the scheduling unit. Because

of the key that needed in first round decrypting is the key that created in the last round by key expansion unit, so it required executing key expansion in advance and it save the key into register. Then executing decryption computing after all the keys is created.

The hardware implementation of reconfigurable computing nodes includes the implementation of top-level module, generation of partial reconfigurable configurable data and save of partial reconfigurable configurable data. We use PlanAhead development platform to generate partial reconfigurable configurable data, including the following four steps: the introduction of netlist files, the layout, and the detection of design rules and the generation of configurable data.

The system can enter the implementation phase after the layout and test of design rules and the system logic module and partial reconfigurable module need to implement respectively then generate corresponding. Used file and . Ncd file. The static logic module needs to be implemented first, and then each reconfigurable module can be implemented. The system can integrate to generate the global and partial .ncd file as well as the corresponding configurable data file (.bit) when the static module and all reconfigurable modules have been implemented. The global configurable data file named static_full.bit and three partial configurable data files named $partial\_plb\_prr\_enrypt$, $partial\_plb\_prr\_decrypt$, $partial\_plb\_prr\_mmm$ are generated respectively for encryption, decryption, and matrix multiplication operation. The FPGA function mapping design view is shown in figure 9, partial reconfigurable area is framed by a box.

V. Experiment and Analysis

A. Analysis of On-chip Resource Utilization

We analyze the impact that the partial reconfiguration bring to the on-chip resource utilization through the statistical analysis of the on-chip resource utilization which is used by partial reconfigurable computing module plb_prr and interface module plb2prr_bridge. Programmable on-chip resources include programmable logic block, block storage unit, high-speed DSP processors, input and output interfaces, connections, etc. Programmable logic block composed mainly by the slice is the basic programming unit, and the use of block storage units and high-speed DSP processor can save a lot of programmable logic blocks. Therefore we identify the on-chip resource utilization through them.

We divide reconfigurable area in hardware logic to implement a variety of computing functions time-sharing. Reconfigurable area is the plb_prr physical block which is divided when the partial data generate, and each reconfigurable computing is time-loaded into the reconfigurable area. The resource utilization of reconfigurable area is showed in figure 3.

From Figure 3 we can see that on-chip resources possessed by reconfigurable area are greater than the resources that each reconfigurable computing function need, therefore we can time-load the reconfigurable computing functions to reconfigurable area. However, the total resources that all reconfigurable computing functions need are far greater than the resources

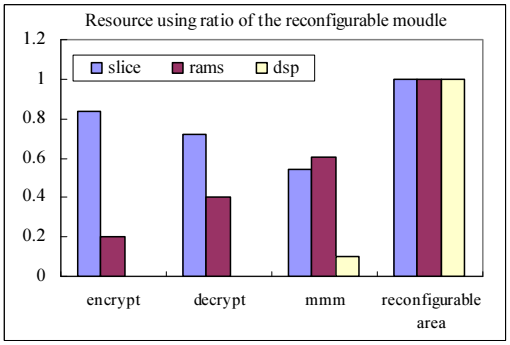

Fig. 3. Used resource rates of partial reconfigurable computing

which are possessed by reconfigurable area, thus the on-chip resource utilization is greatly increased. At the same time all resources inside the reconfigurable area belong to the reconfigurable module in partial reconfigurable design, but not all resources will be effectively used, for example DSP in reconfigurable area in this paper only can be used in the matrix multiplication unit, and the utilization is low.

B. System Performance Analysis

We analyze execution time of three basic computing functions provided by reconfigurable computing nodes, reconfigurable computing nodes are based on the Virtex-4 Development Board of Avnet in test environment, the clock frequency of function unit is 100MHz, general-purpose processing nodes use the Core 2 Duo 1.83GHz processor computer, and general-purpose processing nodes are interconnected with reconfigurable computing nodes through 100M Ethernet.

In the following results:

Speedup ratio of computing unit = software execution time of computing function / hardware execution time of computing unit.

Speedup ratio of partial reconfiguration mode = Software execution time of applications / execution time based on reconfiguration mode.

In those formulae, software execution time of computing function is that the accelerated computing functions execute with software mode on the general-purpose processing nodes. Hardware execution time of computing unit is that that the accelerated computing functions execute with hardware mode on the reconfigurable computing nodes. Software execution time of applications is that applications execute with software mode on the general-purpose processing nodes. Execution time based on reconfigurable mode is that applications execute on the general-purpose processing nodes and applications execute accessing the reconfigurable computing nodes through network realize the accelerated computing functions.

Figure 4, 5 show the Speedup ratio of encryption computing unit with partial reconfiguration computing way. From Figure 4, the speedup ratio of FPGA encryption computing is larger with the software executing way and keeps stable with the large computing scale. The reason is that complexity

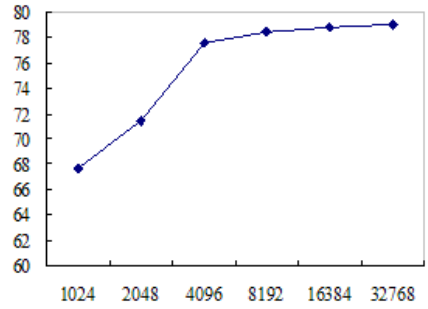

Fig. 4. Speedup ratio of encryption computing unit

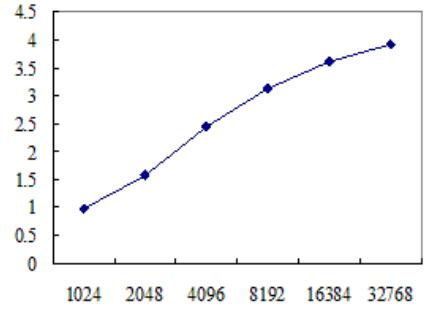

Fig. 5. Whole speedup ratio of encryption computing

of Hardware processing logic may increase as the scale of encryption computing increase. From Figure 5, the speedup ratio of encryption computing on the partial reconfiguration computing way is smaller and increases as the increase of computing scale. There is part which cannot be accelerated in the applications, the network communication overhead and Hardware-accelerated execution time is relatively small.

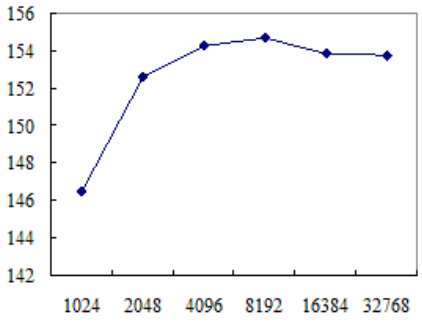

Fig. 6. Speedup ratio of decryption computing unit

Figure 6, 7 show Speedup ratio of decryption computing with partial reconfiguration computing way. From Figure 6, the speedup ratio of FPGA decryption computing is larger with the software executing way and keeps stable with the large computing scale. The reason is that complexity of Hardware processing logic may increase as the scale of encryption computing increase. From Figure 7, the speedup ratio of encryption computing on the partial reconfiguration computing

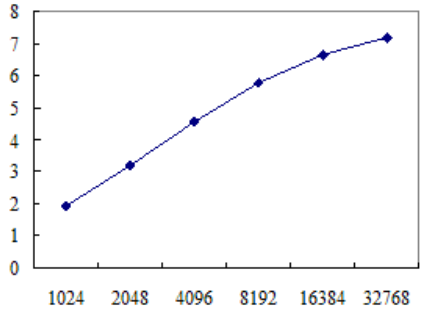

Fig. 7. Whole speedup ratio of decryption computing

way is smaller and increases as the increase of computing scale. The reason is same with the reason of the encryption operations.

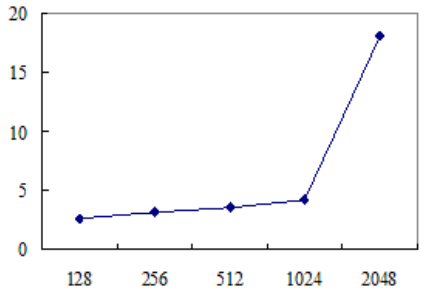

Fig. 8. Speedup ratio of matrix multiplication

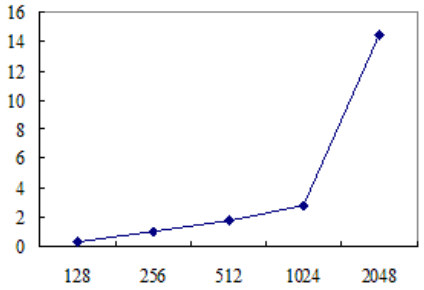

Fig. 9. Whole speedup ratio of matrix multiplication

Figure 8, 9 shows the Matrix operation performance with partial reconfiguration computing way. From Figure 8, the speedup ratio of FPGA matrix multiplication is larger with software executing way and the increase is apparent with large computing scale. The rules of matrix operations are simple, that is different With encryption and decryption, and complexity of hardware processing logic keeps stable as the scale of matrix operations increase. From Figure 9, the speedup ratio of matrix multiplication is relatively small with partial reconfiguration computing mode. The reason is same with the reason of the encryption operations. But the speedup ratio increase as the increase of computing scale. The speedup ratio increase obviously with the larger matrix

scale. Compared with the matrix application of encryption and decryption computation, the part which cannot be accelerated and the network communication overhead is relatively small.

From the analysis of speed-up ratio we can see:

1) About the computing speed, the hardware unit calculating way is fastest, followed by the calculating method of partial reconfiguration, the software implementation is the slowest. From above analysis, the main reason of speed difference is that the hardware calculating unit is parallel and unreconfiguration, it doesn't has the problem of declining of the memory access performance, so running speed is faster. The method of partial reconfiguration calculation, the execution time is not only including execution time of hardware computing unit, but also including the related I/O time, reconfigurable control time, so the calculation speed is relatively slow compared with the simple hardware, but computing model of the partial reconfiguration are more flexible than the hardware unit and utilizes fewer on-chip resource.

2) On the software computing mode, as the computing scale increasing, the speed-up ratio increases and the rate may be less than 1. Compared with this mode, on the partial reconfiguration mode, there is fixed part in the computing time, that's called Treconfig. It doesn't change as the computing scale changes. Therefore, when the computing scale is smaller, the time of software computation will be far less than the time of reconfiguration computation and the speed-up ratio is less than 1. While the computing scale is bigger, I/O time and computing time will increase and the percentage of reconfiguration time will reduce, then the speedup rate will increase. When the computing scale reaches the size of a certain level, then I/O time and computing time will be far more than the time of reconfiguration computation, the speed-up ratio will be basic stability of the same.

3) Compared with AES algorithm, matrix speed-up ratio was big while the matrix scale was large and increased significantly with the matrix size. There are mainly two reasons. The first, the computing time on the mode of partial reconfiguration computation included three parts: I/O time, reconfigurable time, computing time. AES computing complexity is O(N), the calculation time were basically linear time of calculating scale, matrix multiplication complexity is O (N3), the computing time was expressed by the O(N3) of the computation scale. So as the increase of computing scale, the proportion of the computing time in the matrix multiplication calculation increased rapidly in the total time of the mode of partial reconfiguration computation. The second, when the scale of matrix calculation is large, on the mode of software computing, the rate of cache miss maybe increase and memory access performance maybe degrade, then ,the computing time will become longer. But on the mode of hardware unit computation, it doesn't have the problem of declining of memory access performance.

VI. Conclusion

We design a high-performance computing scheme in which the general-purpose processing nodes are connected to the dynamic partial reconfigurable computing nodes through the high-speed network. Using module-based partial reconfiguration design method, a FPGA-based dynamic and partial reconfigurable computing node is designed. This node has the ability to do dynamic and partial reconfiguration and can load different computing units according to the different requirements. System uses the module-based partial reconfigurable design method, divided the hardware system static and dynamic module, and the static and dynamic module of communication module unit bus to partial reconfigurable unit bridge, reconfigurable computing modules, as well as the general-purpose processing nodes and partial reconfigurable computing communication mode are designed. Based on the Virtex-4 Avnet development board, completed the basic hardware structure and the dynamic partial reconfigurable computing nodes in the hardware prototype implementation, on this basis, the global configuration data file is generated and each functional unit corresponds to a calculation of the part configuration data files to achieve the software system of the dynamic partial reconfigurable computing nodes and remote access interface library. Achieving results show that the design of resources to achieve more functions, the dynamic partial reconfigurable computing nodes to complete computing tasks primely, and can effectively improve the system performance.

Acknowledgment

This work was supported by the National High Technology Research and Development Program (863 Program) of China under grant No. 2009AA01Z108, No.2006AA01A109 and No. 2009AA01A135. Part of the work was supported by the XJTU multi-disciplinary project under grant No.2009xjtujc30.

References

[1] M. C. Herbordt, T. V. Court, Y. Gu, B. Sukhwani, A. Conti, J. Model, and D. DiSabello, "Achieving high performance with fpga-based computing," *Computer*, vol. 40, pp. 50–57, 2007.

[2] J. L. Rice, K. H. Abed, and G. R. Morris, "Design heuristics for mapping floating-point scientific computational kernels onto high performance reconfigurable computers," *Journal of Computers*, vol. 4, pp. 542–553, 2009.

[3] T. El-Ghazawi, D. Bennett, D. Poznanovic, A. Cantle, K. Underwood, R. Pennington, D. Buell, A. George, and V. Kindratenko, "Is high-performance reconfigurable computing the next supercomputing paradigm?" in *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, 2006.

[4] R. e. a. Baxter, "High-performance reconfigurable computing c the view from edinburgh," in *Proceedings of the second NASA/ESA Conference on Adaptive Hardware and Systems*, 2007.

[5] T. El-Ghazawi, E. El-Araby, M. Huang, K. Gaj, V. Kindratenko, and D. Buell, "The promise of high-performance reconfigurable computing," *IEEE Computer*, vol. 41, pp. 69–76, 2008.

[6] P. Sedcole, B. Blodget, T. Becker, J. Anderson, and P. Lysaght, "Modular dynamic reconfiguration in virtex fpgas," *IEE Proc.-Comput. Digit. Tech.*, vol. 153, pp. 157–164, 2006.

[7] Xilinx, "Standalone board support package." [Online]. Available: http://www.xilinx.com/support/documentation/index.htm.