A Study on a Multidimensional Configurable Processor Array in Hardware and Software Complex Architecture

Author: Jiang Li

Supervisor: Prof. Masatoshi Sekine

Department of Electrical and Electronic Engineering

Tokyo University of Agriculture and Technology

A Dissertation submitted for degree of Doctor of Engineering

March 2015

Abstract

High performance computing (HPC) usually solves complex science, engineering, and business problems that require huge computation capabilities. The main trend HPC solutions are implemented by supercomputers which are composed of huge amount of general purpose processors as computing nodes on a network, to meet the demand of most high performance computing applications. As the ordinary HPC systems composed of CPUs is limited by power and heat constraints, the system had to be comprised of much larger number of lower-power, lower-performance cores. The high-performance with low power consumption is required. Recently, GPGPU which composes of thousands cores is commonly used to accelerate HPC in many studies, but the actual achieved performance changes greatly for each application relative to its peak performance. In addition, data communication bottleneck among computing nodes also can be solved by through various approaches such as optical communication.

FPGA (Field Programmable Gate Array) is a LSI that can implement most suitable specific processor circuit on particular applications. With the development of FPGA technology, many HPC applications can be accelerated by using FPGAs to deliver enormous performance. The configurable HPC systems which accumulated a lot of FPGAs are able to be widely utilized on HPC to implement high performance on low power consumption.

We constructed a configurable processor array with multidimensional FPGA array, that named as Virtual Object by Configurable Array of Little Scalable Engine(Vocalise). The proposed system has following features:

1. The design and development is high-efficiency and easy-to-use for various applica-

tions through combining software and logic circuits.

2. The scalable multidimensional FPGA array enables implementation of 3-D interconnections. And the network topology of FPGAs can suits to solve different dimensional problems.

In general, applications implemented on system with FPGAs are considered difficult to develop, because hardware design knowledge of application designer and particular development tools are required. Moreover, large-scale FPGA array management is also difficult to be implemented. In order to compensate this problem, hardware object(hwObject) has been proposed in previous studies of our laboratory. The logic circuit as an hwObject can be utilized to implement applications like software object. The application system which using the method called hw/sw complex. Through a hw/sw complex, the application designer can easily realizes configuration/access/control on all FPGA array. A lots of standardized peripheral and control circuit can be hidden in hw/sw complex units, the programmers are only requires to focus on processor circuit design for a new application, and does not require development of Therefore, the development cycle and difficulties reduces.

In the meantime, on almost all of previous configurable computing studies, the interconnection among FPGAs were only connected by 1D or 2D network. For 3D computational problem, when 3D mesh grids are mapped on 2D or 1D processor network, it certainly will arises the data communication loss to cause of system's performance degradation For probing and solving the problem, we implemented a multidimensional FPGA array (maximum dimension is 3D) which composed of many small FPGA cards in our system. Each FPGA card can interconnect using six I/O (top, down, left, right, front, and back) terminals. The multidimensional FPGA array are scalable design that can increase and decrease on a scale freely, and it is easy to control with a host PC through the hw/sw complex method. Meanwhile, the communication network among FPGAs is scalable according to user design. When the system operates multidimensional applications, transmission efficiency among FPGA can be improved through user-adjusted dimensionality and network topologies for different applications.

In the study, we aims to explore merit and demerit of the approach in a real system

with a 3D FPGA array. In order to realize a 3D FPGA array on Vocalise system, we developed a solution to realize fast and flexible circuit configuration on multidimensional FPGA array, and implemented data communications among host and FPGAs. Furthermore, to demonstrate the effectiveness of the proposed methods, we solved numerical calculation problems: CIP method and 3D Poisson equation with Vocalise system running at 66 Mhz. We also evaluated performance, communications overhead among FPGAs and power consumption. As results, one FPGA can performs 1.916 GFlops on 3D CIP method which achieves 99% peak performance (1.93 GFlops). Moreover, 2×2×2 3D FPGA array performs 12.46 GFlops on 3D Poisson equation problem. The maximum scale FPGA array can realizes 199.36 GFlops with consuming 435 W. Although the utilized FPGA is a low performance FPGA in Xilinx products, it is also possible to achieve high-speed operation at 500 MHz. The effective performance of system (8×4×4 FPGAs) can be expected to realize close to 2 TFlops running at 500 Mhz. Based on the above results, we analyzed the specific efficiency and capacity of Vocalise.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Almost all High Performance Computing(HPC) studies aim to realize higher peak performance, higher bandwidth and lower latency among computing nodes and lower power consumption through various approaches. Throughout the development process, there has been a long-standing rivalry between hardware solutions and software solutions for solving the computational problem. Hardware solutions realize the specific computing with massive parallel arithmetic logic array on hardware level such as application-specific integrated circuits (ASIC). It is high-performance, high-efficiency and low power consumption, but is difficult to use and very high cost. Software solutions realize the computing with general-purpose processor array which implement algorithms variable through instruction streams from operating system (software). They are flexible and very easy-to-use, but low performance and power efficiency. In decades past, software solutions became the main trend in HPC. Because of fast-developing of software and general-purpose processor, the performance can meet the many applications requirements. But there are still no substantive changes in its disadvantages. With the development of semiconductor technology, especially very flexible high performance programmable logic hardware arise, like field-programmable gate arrays (FPGAs). It becomes possible to realize a kind of reconfigurable computing architecture which combing the flexibility of software with high performance of hardware. Thus computing architecture not only has advantage between hardware solution and software solution, but

also blurs the borders between hardware solution and software solution. It could hopefully becomes the leading HPC area in new period over the next several decades. In this thesis, we build a HPC system by using the reconfigurable computing approach.

1.1 Background

Today more supercomputers are designed to achieve higher performance with lower power consumption. The conventional supercomputers which are composed of general-processors are main trend for High Performance Computing (HPC) applications. It is easy for general-purpose CPUs to provide an optimal solution for the broad spectrum of HPC applications. The conventional super-computers mainly rely on increasing implemented cores to achieve higher peak-performance. However, there is a demand to the conventional supercomputers for improving the parallel-processing efficiency. It depends mainly on the overhead in synchronization or data transfer among the shared memories or message transposing, most of the computational power has been wasted. In many cases, the bandwidth among the parallel microprocessors causes a limitation of the overall performance.

For example, Fujitsu K supercomputer is the fastest supercomputer in 2011, uses 68,544 2.0GHz 8-core SPARC64 VIII-fx processors packed in 672 cabinets, total of 548,352 cores, manufactured by Fujitsu with 45nm CMOS process technology. Its performance is 8.162 PFlops on LINPACK benchmark[5]. However, by increasing number of the composed processor, the power consumption and running costs were rapidly increased. The highest total power consumption of Fujitsu K supercomputer was estimated to be 9.89 MW, while the average power consumption of a Top 10 system is 4.3MW. Thus the power consumption roughly equals to that of 10,000 houses, and its annual running cost is 10 million US dollars. It is important to reduce the power consumption and cost. And the power dissipation and heat has become one of the major drawbacks to limit to achieve higher peak performance of supercomputer. Therefore, it becomes an important trend to achieve higher computation efficiency and lower power consumption on most supercomputer researches in recent years [6]. The Green500 List [7] was paid

more and more attentions, which supercomputers can be compared by performance-per-watt.

Recent years, GPGPU(general-purpose computing on graphics process units) originally designed for computer video cards have emerged as the most powerful chip to accelerating high performance computing. Different to multi-core CPU architecture which currently ship with two or four cores, GPU architectures running thousands of threads in parallel with hundreds of cores. There are several approaches to programming GPUs such as NVIDIA CUDA(Compute Unified Device Architecture), AMD stream, OpenACC and OpenCL ect.. Many new generation supercomputers implemented with GPGPUs to realized higher performance and power, such as TSUBAME-KFC [8], Tianhe-I [9] ect..

Meanwhile, interest has arisen in augmenting these clusters with programmable logic devices, such as Field Programmable Gate Array (FPGAs). The FPGA is a reconfigurable hardware. The FPGA's hardware can creates custom specific processing units which are optimized to meet the particular requirements of each HPC application.

In many cases, FPGAs are effective for the high-performance computing (HPC) applications, and this solution can potentially deliver enormous performance. And many HPC applications can be accelerated by incorporating specialized processing capabilities to handle particular tasks.

Although the number of usable gates in FPGA has increased up to several million, parallel applications will require even larger numbers. Because of FPGA capacity has terms for each of the available hardware resources, including hard multipliers and BRAMs as well as general-purpose logic elements. Depending on the application, any of the resources can become the limiting one. For solving many practical applications, computational capability of an FPGA is often not enough. For example, on our previous research [19],The system needed at least 10 FPGAs to implement simultaneously brain process applications such as move motion, voice recognition, image recognition, voice synthesis.

Therefore, a scalable FPGA array structure is also interesting. The FPGA's computing power comes from the parallel it uses to handle a problem. Special purpose processors

built with FPGAs are becoming popular in super computing. Many parallel computing systems with multi-FPGAs have been developed, such as Maxwell [41], the Berkeley Emulation Engine (BEE) [13], Cube [14], programmable active memory (PAM) [15], and the systolic computational-memory array (SCMA) [16].

BEE3 The third-generation BEE (BEE3) comprises modules with four Virtex-5 FPGAs connected by ring interconnection. BEE3 is a production multi-FPGA system with up to 64 GB of dynamic random-access memory (DRAM) and several I/O subsystems that can be used to enable faster, larger and higher fidelity computer architecture or other systems research.

Maxwell Maxwell is a 64 FPGA supercomputer with an IBM Blade Centre Cluster and FPGA acceleration. It has 32 Blade servers, each with one Intel Xeon CPU and two Xilinx Virtex-4 FPGAs. The CPUs are connected to the FPGAs by a standard IBM PCI-X expansion module. The FPGAs are connected by a dedicated 2D torus network.

Cube Cube is a massively parallel FPGA cluster that contains 512 Xilinx Spartan 3 FPGAs on 64 boards. The FPGAs on each board are connected in a chain and are suited to pipeline and systolic architectures.

PAM The FPGA-based PAM comprises a 2D array of FPGAs, and external local-memory behaves as memory for a host machine while processing the stored data.

SCMA The systolic computational-memory array(SCMA) is extensible over a 1D or 2D array of FPGAs connected by a mesh network. It is designed for extensibility with multiple devices for high and scalable performance of floating-point computation.

Above researches show that the multi-FPGAs HPC system can usually provide higher utilize the peak performance of hardware, higher computation performance and better power efficiency. For example, a SCMA system, which equips two ALTERA Stratix II FPGA, The implemented system, which implements 192 processing elements run-

ning at 106MHz; it has 40.7 GFlops the peak performance in single precision computation. The SCMA achieves high utilization of peak performance, about 32.8 to 35.7 GFlops performances can be implemented for the simple benchmark computations, which consists of red black-SOR (successive over-relaxation) method, Fractional-step method (FRAC) method, finite-difference time-domain(FDTD) method computation. The SCMA can provides 29 times and 10.13 times faster computation than 3.4-GHz Pentium4 processor for Fractional-step method (FRAC) computation and finite-difference time-domain(FDTD) method computation. Consequently, double- FPGA SCMA consumes 90.21 W to 109.81 W for the simple benchmark computations . While software computation with the Pentium4 processor which has the average power consumption of 125.99 W. It means that the FPGA system consumes the 69% to 87% power of software computation with the Pentium4 processor, and requires only 2.8% to 7.0% of the total energy for the same computations with Pentium4 processor to realize the computation speedup.

Because of FPGA has high efficiency, low power consumption and low-heating characteristics, but CPU and GPU performance/watt are hitting the wall in recent years. Recent studies have shown that FPGA-based application can achieve more than 10 times better performance per watt and latency improvement compared to CPU/GPU implementation[17][18]. FPGAs can provide the heart of what's needed for power-efficient hardware application acceleration on one chip while providing solutions that are below the 25W per board targets. For instance, Baidu (Chinese web services leader) presented their research at the 2014 Hot Chips Symposium,[18] , they achieved 375 GFlops dissipating less than 20W in DNN(Deep Neural Network) prediction computation with mid-end FPGAs, the FPGAs can deliver more performance than CPUs or GPUs .

On the other hand, although there are lots of researches which implement applications with FPGA. But the reconfigurable computing system with FPGAs is still difficult to be widely used by users in many practical HPC applications. Because of the difficulty of the hardware design and the lack of the particular design tools in building system. If developers want to utilize the heterogeneous HPC system with FPGA, they often have to

study and master VHDL/Verilog HDL language to design the whole circuits on FPGAs. The developing cost is very high and the hardware development cycle is very long; the designer have to spend an amount of time to design the whole circuits implemented on FPGA, corresponding interface circuit and driver for every specific application. Therefore it is important to provide an easy-to-use development environment on a HPC system with FPGAs.

A kind of hardware/software complex architecture was proposed on a lot of previous works of our laboratory.[19]-[23] The proposed system composed of host pc and a PCI FPGA device. An object-oriented application operations were able to be divided into Objects processed by CPU and Objects processed by FPGA on PCI device. The user can easily utilize the objects operated by FPGA like software objects. By using this approach, we had realized amount of accelerated application processes such as the image recognition process, half-negation web application process etc. on previous of proposed system. Furthermore, the application developing cost and cycle also were effectively reduced [20].

In order to implement a high-efficiency and easy-to-use development environment for multi-FPGAs HPC systems. We utilize hw/sw complex concept to implement our HPC system. The controls/data communications of FPGA array and computation of application are implemented by hw/sw complex units. Meanwhile, the data communication circuit, control circuit among host PC and FPGA array have been standardized and achieved as peripheral circuits, which spend very much time and energy in the design. The hardware developers only need to design the processing elements which implemented on FPGA array for a new specific application, and focus on execution efficiency of circuit and parallel efficiency among processing elements on multi-FPGAs. It enables to effectively reduce the difficulty of system utilization and development cycle.

1.2 Objectives of the study

Many HPC applications such as partial differential equation (PDE) solvers [24], climate/ocean-modeling systems[25], and molecular dynamics simulations[26] use

Cartesian grids of different dimensions and structure. A distributed computing system operates these applications with a processor array in parallel through grid processes mapped on a computing network. However, the communication pattern might be completely different for different applications. Process mapping on the network significantly affects application performance. Moreover, communication efficiency might also be changed when operating applications with different network topologies.

For example, for 3D PDE problems, each Cartesian grid must perform nearest-neighbor communication along the edges. The 3D computational domain is arranged to each parallel computing node/processor of a 1D or 2D computing network. Each node/processor communicates not only with its physically nearest neighbors; it is forced to share network links with other communication. Such sharing results in significant communication contention and performance loss.

While FPGAs are connected by 2D direct interconnection in some multi-FPGAs system, such as Maxwell, PAM and SCAM. Cube comprises multiple FPGAs in 3D space; however communication among FPGAs on each board is still achieved with a chain interconnection. When operating 3D numerical calculation problems, the 1D or 2D physical layout of application processes might not match the communication characteristics of the application to result in performance and communication efficiency loss. Thus, link bandwidth often needs to be doubled and redoubled because data communication among nodes might cause system bottleneck. With a 3D FPGA array, the communication efficiency is able to be improved.

For improving this issue, we propose an multidimensional FPGA array designed with configurable circuits. We designed a reconfigurable parallel computing platform with a multidimensional array of FPGAs. This custom computing system has been named the "Virtual Object by Configurable Array of Little Scalable Engine (Vocalise)".

The purpose of the system is to study the feasibility of an application-specific multidimensional configuration of the FPGA array. The personal HPC can be configured to customize it for specific problems. In Vocalise, each FPGA card has six-way 3D I/Os that enable implementation of 3-D interconnection, FPGA array can be configured in a cubic form or a plane form for each specific problem, like LEGO block. By using appli-

cation specific and scalable multidimensional interconnection, the FPGA array is easy to create various network topologies of different dimension (1D, 2D, or 3D) and information. For different applications, we can configure the physical layout of the system to match communication patterns of different applications and optimally map processes to the network to achieve improved communication efficiency.

Considering a 3D FPGA array easily achieves higher transmission efficiency than a 1D or 2D arrays of FPGAs for a 3D PDE problem. In our approach, the multidimensional computational domain is arranged to each parallel computing node/processor which is in a same dimensional computing network, For a 3D problem, the computation is operated by 3D FPGA array.

The study aims to evaluate and explore the merit and demerit of the approach in a real system. To clarify the capacity, we implemented Vocalise system with a 3D FPGA array firstly. With sw/hw complex, we achieved the communication/control of a 3D FPGA array. In order to realize the high efficiency circuit configuration, we developed a flexible and high-speed circuit configuration solution in parallel for a large-scale 3D FPGA array. Through developing a configuration circuits implemented on FPGA, bitstream data of circuits can be effectively written into any connected FPGAs in parallel with the implemented configuration circuit via the existing network connections among multidimensional FPGAs.

In addition, we implement the computations of multidimensional CIP method on 1 FPGA, and 3D Poisson equation on 1 FPGA, 2×2 FPGAs, and 3D(2×2×2 FPGAs) FPGA array, and examined the following features in the study: 1. Performance of the processing element implemented on FPGAs. 2. The overheads of data communication between adjacent FPGAs on 2D/3D network. 3. Power consumption and power efficiency of the FPGA array. 4. Capability of multidimensional FPGA array.

We demonstrated that the Vocalise system has enough computation power to implement HPC applications. Through analyzing above measured results, we also estimated the computation performance and communication condition of a RHPC system which was built with large-scale multidimensional FPGA array.

1.3 Dissertation constitutes

This Chapter introduces background and study objectives.

The remainder of the thesis is organized as follows.

Chapter 2 the previous works of our laboratory is introduced. It consists of hw/sw complex system hardware and software compositions. The hw/sw complex and FPGA board: hwModule series and implementation method for solving specific applications with hw/sw complex architecture will be presented. Moreover, in the chapter, we also introduce how to utilize the hw/sw complex on our Vocalise system, a multidimensional FPGA array system to implement applications; the design method and throughout the development process will be described and discussed.

Chapter 3 describes hardware architecture of Vocalise system. The system composed of amounts of FPGAs through VC Bus network and VI Bus network. Through the two networks, the configuration method on multidimensional FPGA array, the data transfer mechanism and parallel computing implementations among FPGAs are realized. Moreover, the hardware design on FPGAs will be described. Discusses features of VC Bus network and VI Bus network.

Chapter 4 demonstrates sample applications which have been realized on our system to evaluate the system performance. Implementations of numerical simulation: Advection Equation with CIP method and Poisson Equation with Jacobi method are described in detail. Then, evaluates the performances of system for solving advection equation with CIP method and Poisson equation, data communication overhead among FPGAs and power consumption.

Chapter 5 summarizes conclusions and suggestions for future work.

Chapter 2

Previous Work - Hardware and Software Complex Architecture

2.1 Hw/sw complex

The Vocalise system is based on a hardware and software (hw/sw) complex that have been previously proposed in [20] by our laboratory. Using the hw/sw complex design style, we can design a system with swObjects only at the beginning, and then replacing the swObjects with hwObjects, the design and implementation are completed at the end. The concept of hw/sw complex unit is shown in Figure 2.1. The hw/sw complex consists of a Host PC which is a standard X86 architecture computer, and hwModule board which equip numbers of FPGAs and SDRAM.

The HwModule is attached to the PCI bus of the standard Windows PC, and has some FPGAs for the applications. The Device Driver of hwModule Board is composed according to WDM (Windows Driver Model) specifications and operates under Windows XP OS.

By using the hw/sw complex unit, execution of application can be separated by two parts, the object processed by Host Processor is named ”swObject”, the object processed by FPGA is named ”hwObject”. The computing core on FPGA that processes hwObject

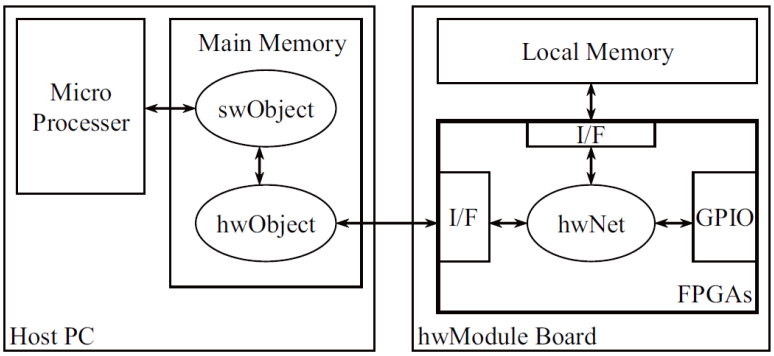

Fig. 2.1 A conceptual model of hw/sw complex system

is named ”hwNet”.

2.1.1 SwObject and hwObject

SwObject means an object operated by software, it is an ordinary object described by C++ Language Class. The operating data is implemented on main memory of Host PC, and the swObject operation is implemented on member function which computed with general purpose processor. Based on this character, we currently utilize swObject to execute the complicated operation such as complicated analysis and logical judgment.

HwObject means an object operated by hardware; it is one kind of special object that its operation is executed by a specific process circuit on FPGA. The hardware object(HwObject) is implemented on FPGAs of hwModule as HwNets, and its data, functions, and interface from the host are implemented in main memory. In sw/hw complex unit, we can utilize a hwObject like a software object. We can generate a corresponding hwObject can when the operation is require, and also delete the hwObject when the operation. The cooperation of the software and the hardware is easily realized. When application has large-scale bit operations, or signal stream operations, and parallel computing, these parts of application is usually suited to be operated by a application specific processors efficiently, relative to software operated by CPU. It is possible to achieve high efficiency operation and reducing the whole computation time by utilizing a hwObject

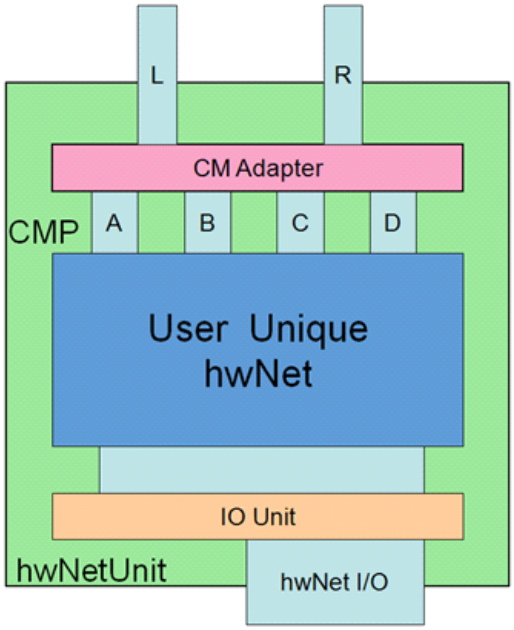

Fig. 2.2 A typical hwNet unit

to process the bit operation, signal operation, and massive parallel computing.

Meanwhile, with the development of FPGA technology because of the reconfigurability of FPGA, it has been widely utilized as a replacement for ASICs in the application-specific domain, it is suitable for rapid prototyping, and quick time-to-market. In a hw/sw complex unit, parallel computing parts usually are suited to operated by hwObject, complicated analyze operations are operated by swObject. Through the method, system simultaneously has flexibility of software and high-speed of hardware.

The Figure 2.2 shows a typical Block diagram of hwNet on an FPGA.

2.1.2 HwNet

The hwNet is a virtual circuit. The Figure 2.2 shows a typical hNet Unit. The hwNet is separated from peripheral circuits through corresponding Interface circuits: CM Adapter module and hwNet I/O module. Through the CM Adapter module, a hwNet can accesses

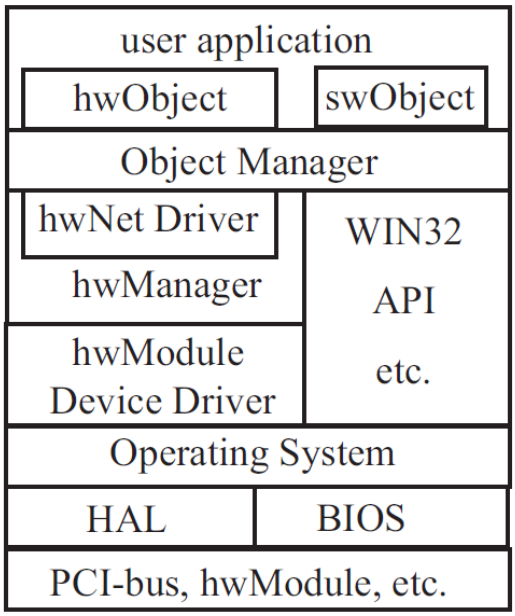

Fig. 2.3 A layers model from hwObject to hwModule Board

SDRAM on Board. Through hwNet I/O module, the hwNet can receive signals from host PC, so that users also directly control the hwNet from the software with hwObject. The hwNet is also possible to connect to external devices through GPIO (General Purpose Input Output Interface).

The Figure 2.3 shows a layer model from hwObject to hwModule Board. In the hw/sw complex unit, the virtual hardware circuit that is being configured on the FPGA is installed as library in an existing software development environment as hwObject. Though object manager (ObjectManager) enables to host PC can access and control the hwModule. By using the ObjectManager, the circuit (HwNet) on hwModule is easily accessed from the standard C++ compiler and can be used as an object in applications.

For the applications, the hwObject is just an object same as swObject. Thus enabling composing, computing and deleting at any time, which also interlocks the writing, computing, and the users can replace hwNet on the FPGA according the requirement at any time. The hw/sw complex architecture enables software programmer to reutilize the

prepared hwObject and thus implementing hw/sw complex system as an extension of Object-Orientated Programming.

As shown in Figure 2.1, the controlling the peripheral circuits, communication between swObject, the corresponding interface module by concealing the process regarding dynamic composition of hwNet. Consequently, the developed hwNet operates on FPGA boards. Since the control circuit of PCI bus and SDRAM; which overloads the development, is being concealed at the external of hwNet, hardware designer need only to concentrate on the designing of hwNet. Through the method as such explained above, we are making attempt to realize the virtual circuit for hwNet and development cost reduction.

Using the hw/sw complex design method, we can design a system that temporarily includes swObjects only at the beginning of the design process, then replace the swObjects with hwObjects when completing hardware design and implementation.

From the viewpoint of reutilizing hwObject, the programmer don't need to re-architecture the whole circuit every time on corresponding FPGA board, just redesign processing core as a hwNet. We established a standard bus inner FPGA named FIB (FPGA Internal Bus) and connect hwNets via FIB.

■**FPGA Internal Bus(FIB)** When an FPGA implements multiple Modules or hwNets has been implemented FPGA, the communications between the modules or hwNets usually have been achieved by lots of master and slave. It is necessary to establish a efficient bus standard to meet the communication demands. We proposed a standard bus: FPGA Internal Bus(FIB) to users reduce the development difficulty and cycle.

FIB is a typical Master/Slave transaction mechanism; and FIB consists of following signals.

Global signals The Global signals of FIB composed of CLK signal and nRst signal(shown in Table 2.1). CLK signal: provide clock signal for transaction, and rising edges of signals is effective. nRst signals: Asynchronous reset signal for system; negative logic operation is effective.

Data line divides Read Data(**DW** and Write Data. **BE** bus transaction mode signal.

Table 2.1 Global Signals

Signals	Source	Description
CLK	System	system clock signal
nRST	System	For bus initialization

WE Wirte/Read signal. **SEL** send from master, declare connection status between slaves **MRDY**, **SRDY** declare status of data transfer.

Arbitration signals Masters transfer REQ signal to arbiter to gain access authorization of bus. The bit-width of SSA signal can be changing with the number of slaves. When the circuit has multiple masters and salvers, the Arbiter Module arbitrates one of masters to gain access authorization of bus, and send signals to the specified slave. Likewise, the Arbiter Module also is able to achieve management of signals from slaves, and send signals to the corresponding master.

Transfer mechanism When a master transfers data to a slave that there are multiple masters on an FPGA, following operations need to be completed. The Figure 2.4 shows The waveform in FIB (FPGA inner Bus). Firstly, the master module send a REQ signals to the arbiter to gain access privilege of bus. When has multiple slave, the master outputs a SSA signal to specify a slave module at the same time. When receiving a ACK signal, the master send address signal and SEL signal. See from bus line, when SEL signal

Table 2.2 Bus Transaction Signals

Signals	Source	Description
A[31:0]	Master	address signals .DWORD (32 [bit]) addressing.
DW[31:0]	Master	Writing Data signals
DR[31:0]	Slave	Reading Data signals
BE[31:0]	Master	Byte enabling
WE	Master	Judge Write/Read operation
SEL	Master	Transaction from master
MRDY	Master	Transaction Ready signal from master
SRDY	Slave	Transaction Ready signal from slave

Table 2.3 Arbitration Signals

Signals	Source	Description
SSA	Master	Specified signal for slave
REQ	Master	Request signal for utilizing Bus
ACK	Arbiter	acknowledge signals for utilizing Bus

becomes high means there are data transfers via Bus line. When master and slave receive read signals(MRDY and SRDY) from each other, master send data to slave. The SEL signal is de-asserted when the data communication has been completed, and then REQ signal is de-asserted, the master release the bus permission. In our design, FIB can achieve single transmission and burst transmission.

2.2 HwModule series

2.2.1 HwModule V2

The hwModule V2 is a PCI device, which implements FPGAs (reconfigurable LSI) and independent memories. This can be easily mounted to a general-purpose PC, and is used as a platform for the virtual circuit.

In Vocalise system, the “hwModule V2” FPGA board is mainly a device that connects extensible multi-FPGAs, i. e., an scalable array of FPGAs. This PCI device is used to configure application specific circuits and implement data communication between the host PC and the extensible multi-FPGAs. Use of the hw/sw complex reduces development cycle time and design difficulties of complex applications. The appearance of hwModule V2 is shown in the Figure 2.5.

The Table 2.4 lists the specifications of hwModule V2 (shown in Figure 2.2). The board implements 4 FPGAs. There is one FPGA to implement a bridge circuit to achieve communications between host PC and other 3 FPGAs, and other 3 FPGAs are utilized to implement hwNet (virtual circuit) for the users. It is also equipped with four (16MB) SDRAM to store the process data as the Local Memory. In addition, the hwModule

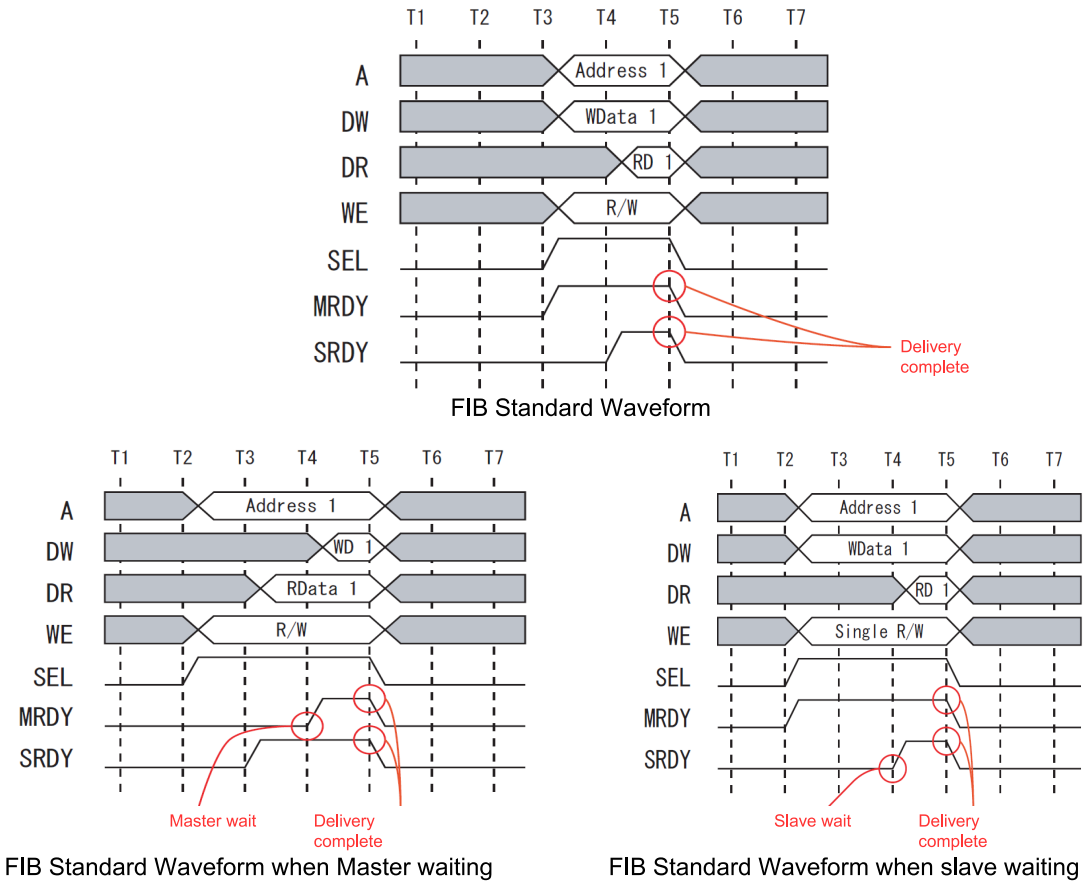

Fig. 2.4 FIB-Bus waveform of FIB

V2 implements three General Purpose Interfaces (GPIF) to achieve connections with external devices.

The Figure2.6 shows the circuits in HwModule V2.

■**Inner bus line** Inner bus lines and circuits of HwModule V2 are shown in Table 2.5. LM-Bus: The Data line for local memory, implement to transfer large amounts of data between LM and user-FPGA. HN-Bus: the control command lines for User-FPGA, implement management of users-FPGA. Configuration-Bus achieve to configure circuits of hwNet on User-FPGAs. And PCI-Bus is implemented to achieve communication between PCI-FPGA and host PC.

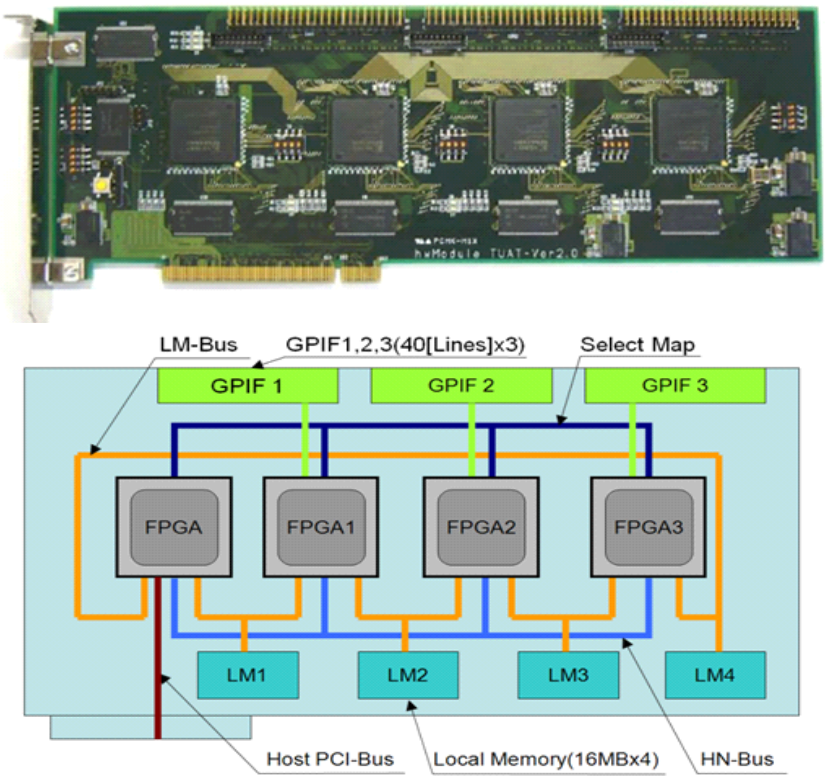

Fig. 2.5 The appearance of hwModuleV2

■**PCI-FPGA** We used 1 FPGA(XC3S1000) as PCI-FPGA to achieve communication and controls from host PC, and circuit configurations on hwModule V2.

The FPGA implements 4 circuits that consist of PCI Controller, LM-Bus Bridge, hwNet I/O Bridge, User-FPGA Configurator. PCI Controller achieve communication based on PCI signal standard with Host PC. Host PC enable to access the hwModule V2 with the PCI Controller. LM-Bus Bridge: achieve data communication between Local Memory and PCI-FPGA. When multi-FPGA are implemented, the LM-Bus is shared by all FPGA. HwNetI/OBridge is the controller circuit for HN-Bus. User-FPGA configurator implemented other circuits of other 3 FPGAs. Through PCI-FPGA, users can easily

Table 2.4 hwModule V2 Specification

Implemented FPGA	Purpose
Xilinx Spartan3 XC3S1000 × 1	For PCI control circuit (PCI FPGA)
Xilinx Spartan3 XC3S1000 × 4	For hwNet circuit(User FPGA)

Implemented SDRAM	Max Frequency	Purpose
16[MB] × 1	133[MHz]	For Configuration Cache Memory
16[MB] × 4	133[MHz]	For Local Memory

Implemented Connectors	Pin
GPIF	Data(40 Pin), 3 Ports , total 120 Pin
LVDS	Data(16 pin), 3 Ports, total 48 pin

implement circuit configuration of other 3FPGAs on hwModule V2 by host PC, and data

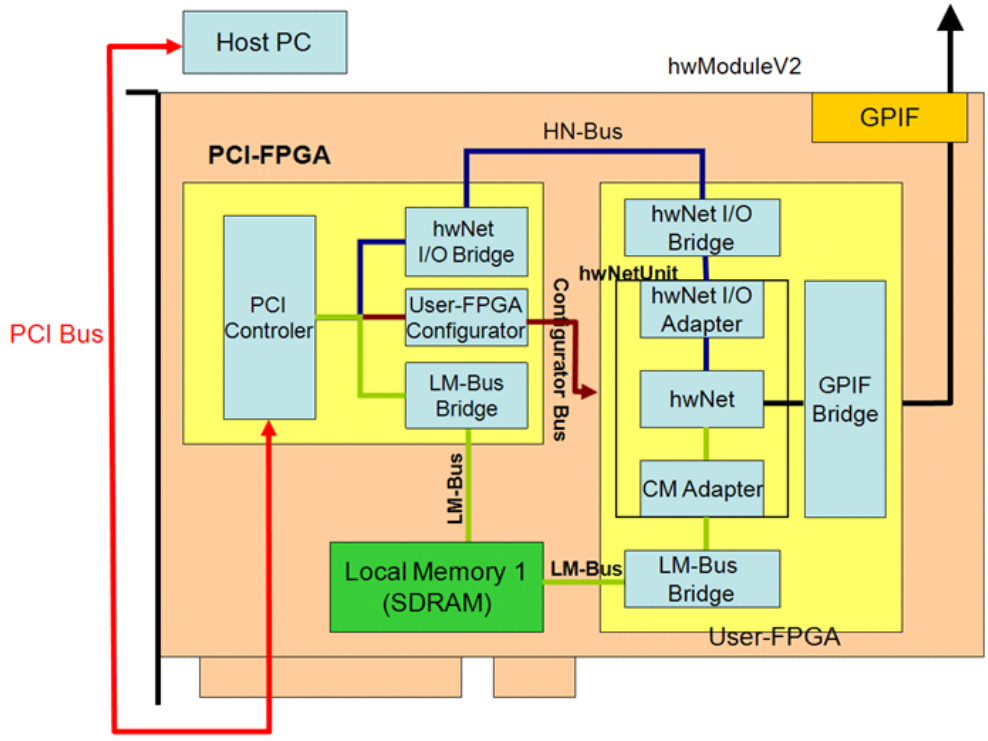

Fig. 2.6 Inner Block circuits on hwModule V2

Table 2.5 Inner Bus of HwModlue V2

Bus Type	Description
LM-Bus	For Data stream
HN-Bus	For communication of hwNets
Configuration-Bus	For Configurations of hwNet in user FPGA

communication with hwModule V2.

■**User-FPGA** The other 3 FPGA is named User-FPGA. The user can implement the various hwNets(virtual circuits) on these 3 User-FPGAs according to different requirements.

The implemented circuits mainly consist of 4 type:LM-Bus Bridge・hwNet I/O Bridge・hwNet Unit・GPIF Bridge. The hwNet I/O Bridge acheive data communication and control signals from host PC. The hwNetUnit consists of hwNet(virtual circuits), CMAdapter(hwNet connect to LM-Bus Bridge), hwNetI/O Adapter (hwNet connect to hwNet I/O Bridge). Through the User-FPGA, we can not only achieve the application computing on hwModule V2, but also access lots of external computing device to elastically expand computing power of the whole system.

■**Local Memory（SDRAM)** The hwModuleV2 equips 4 SDRAM(16MB) to store the computing results of hwNets. Because of memory host PC is named as Main Memory. Correspondingly these SDRAMs are named Local Memory. The available space of Local Memory by users are total 64 MB on a hwModule V2.

■**General purpose interfaces (GPIF)** The hwModule V2 equips 3 GPIF connectors. Each GPIF corresponding a user FPGA, and is 50 pin connector, data pin 40 pin, total 120 pin on 3 GPIFs.

Through the GPIFs, the HwModule V2 can accesses extern FPGA devices , implements to control the hwNets or communicate data. The computing power of system is extendible by connected extern FPGAs.

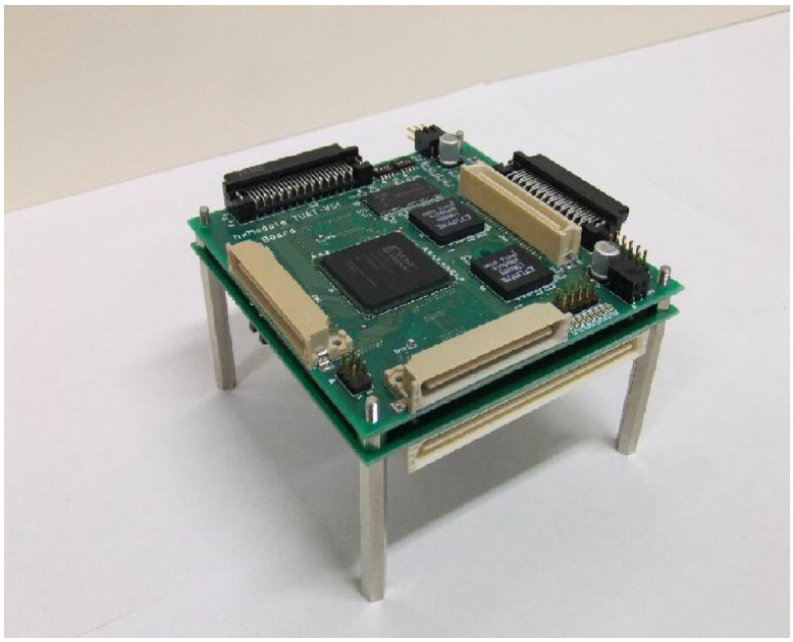

Fig. 2.7 Appearance of hwModule VS

2.2.2 HwMoudle VS

The hwModule V2 can connect numbers of small FPGA boards to extend systems computing performance. The small FPGA boards, named “hwModule VS”.(shown in Figure 2.7)

The hwMoudle VS board comprises a single Sub Board and a single Processing Element (PE) Board. Figure shows a PE Board of hwMoudle VS.

Figure 2.8 shows a Sub Board of hwMoudle VS.

The Sub Board is equipped with one Xilinx Spartan-3 XC3S700A FPGA. The Sub Board equips 4 GPIF IO. Through front and back GPIF IOs, a sub Board can connect two other sub Board. And other GPIF connects its own PE Board. The Table 2.6 has shown the specific of Sub Board. In our system, sub board has mainly two functions.

1.) Sub Board supports the implementation of special circuits into its own PE board

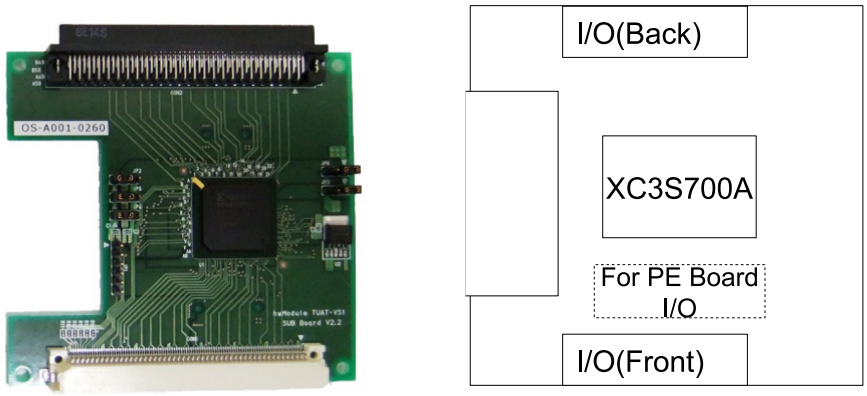

Fig. 2.8 Sub board Appearance and block diagram

or the next Sub Board.

2.) Sub Board also transfers computing data and command signals between the host PC and the PE Board. For many applications, the Sub Board as selector circuit to be implemented on FPGA of Sub Board, Therefore Sub Board is not equipped with any local memory such as SDR SDRAM.

Figure 2.9 shows a PE Board of hwMoudle VS.

A PE Board equips one Xilinx Spartan-3 XC3S4000 FPGA. And the PE Board also equips two 32bit-16MB SDR SDRAM as Local Memory on Board. The PE board has eight-way General Purpose Interface (GPIF) I/O ports. There are two GPIF I/Os to be used to connect to sub Board, and other six-way GPIF I/Os are used to connect with adjacent PE Board; thus, it is possible to achieve three dimensional data transfer in the same time. Based on the characteristic, the distributed system is easily extensible to 1D, 2D, or 3D FPGA arrays. It is primarily used to implement the custom arithmetic circuit for different types of applications. The computational mesh is homogeneously partitioned into each hwModule VS.

Table 2.6 The specific of Sub Board

LSI	
FPGA	Xilinx Spartan3 XC3S700A
Connector	Description
Front/Back connection between Sub Board	1 Port: 78-pin) Front/Back Total: 156-pin
SIMDATA line Connector	Data line: 32-pin, Control line: 26-pin, Total 58-pin
Configuration Connector	23-pin

2.3 Application implementation of Vocalise system

In previous sections, we mainly described the concept of the hw/sw complex. In practical applications, we utilize hw/sw complex units to implement the computations of application, circuit configuration and access/control on FPGA array. In this section, we will introduce how to implement applications on Vocalise system by using hw/sw complex unit. We illustrate the concepts and detail design of swObject and hwObject on Vocalise system, process flow, design flow of hwObject model.

Table 2.7 The specific of PE Board

LSI	
FPGA	Xilinx Spartan3 XC3S4000
SDRAM	16[MB]×2
Connection	Description
Connector between PE Boards	DATA line: 32-pin, Control line 17-pin CLK lin :1-pin for each Port, Total: 50-pin 6 Ports(Up/Down・Front/Back・Left/Right) Total:300-pin
SIMDATA Connector	Data (32-pin) , Control (26-pin) ,Total 58-pin
Configuration Connector	23 pin

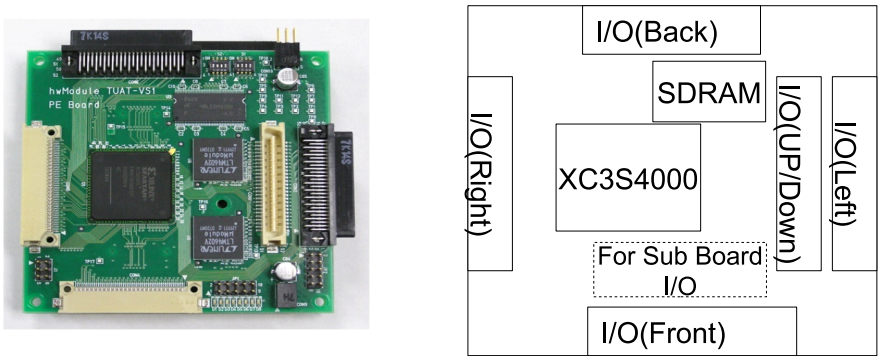

Fig. 2.9 PE board appearance and block diagram

2.3.1 SwObject in Vocalise system

The swObjects which are a ordinary object described by C++ Language Class. Computing data is implemented on main memory of Host PC, and the swObject's operation is implemented on member function which computed with Host PC's general purpose processor. The essence of a swObject is achieved by allocating memory area on main memory of Host PC. And swObject operations are implemented by member functions operated by CPU. These member functions are subroutines of loaded program in main memory. When having functions calls, CPU achieves decode and operate the subroutine from code area. Actually, the operations of swObject are same to ordinary software operations.

The conceptual swObject on Vocalise system is shown in Figure 2.10.

When Vocalise system implements most HPC applications, the main processes of applications are normally operated by hwObject, the swObjects are usually to operate pre-process or post-process of application.

Pre-process: The swObject decomposes computational data domain into many sub-computational domain in main memory as requirements of application. Each sub-computational data domain corresponds to a local memory on hw Module VS. And swObject can accesses the hwObject to map sub-computational data in main memory to

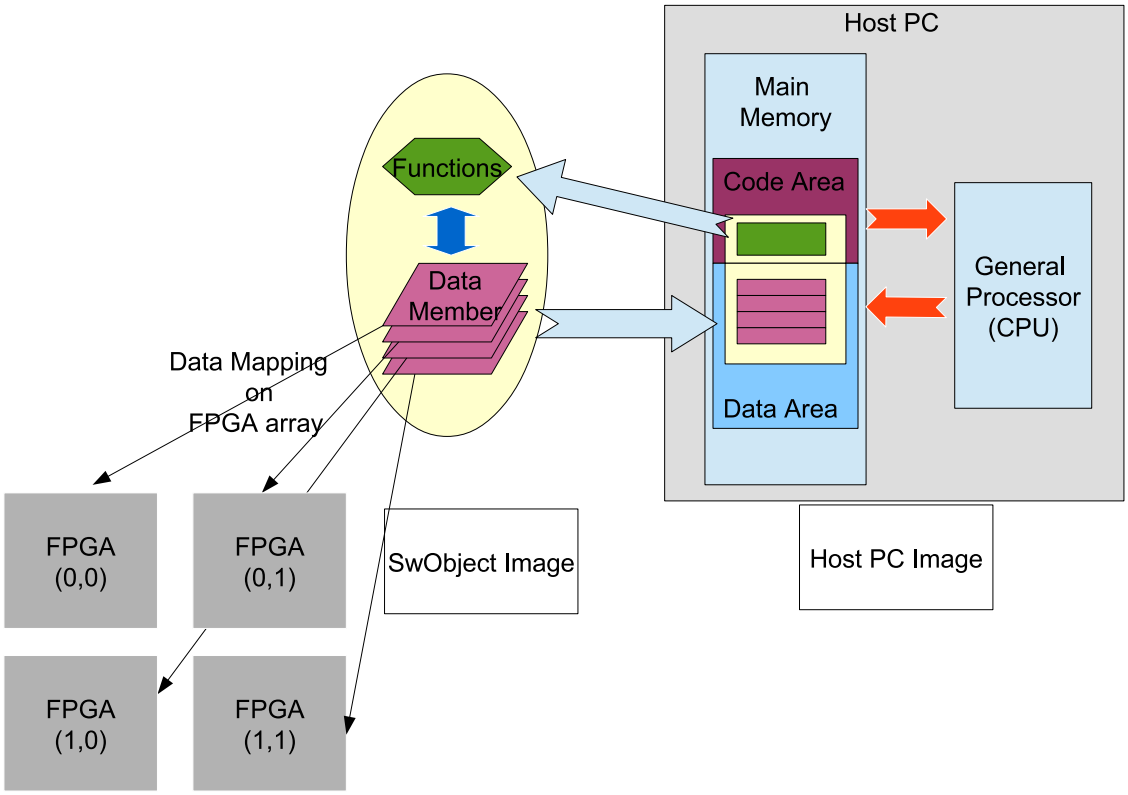

Fig. 2.10 SwObeject concept on Vocalise.

each local memory on the designated VS.

Post-process: When FPGA array completes the calculations, the swObject accesses a hwObject to read the result data on each VS, the result data is mapped on data area in main memory of Host PC. Then the swObject outputs the result data to display program or others post-processing programs.

2.3.2 HwObject in Vocalise system

As well as swObject's operation, the hwObject's operation also can be implemented through calling member functions of hwObject. The member function calling on swObject is implemented by ordinary subroutine which processed by CPU. While the member function calling on hwObject is just a starting procedure to hwNet. The process can be

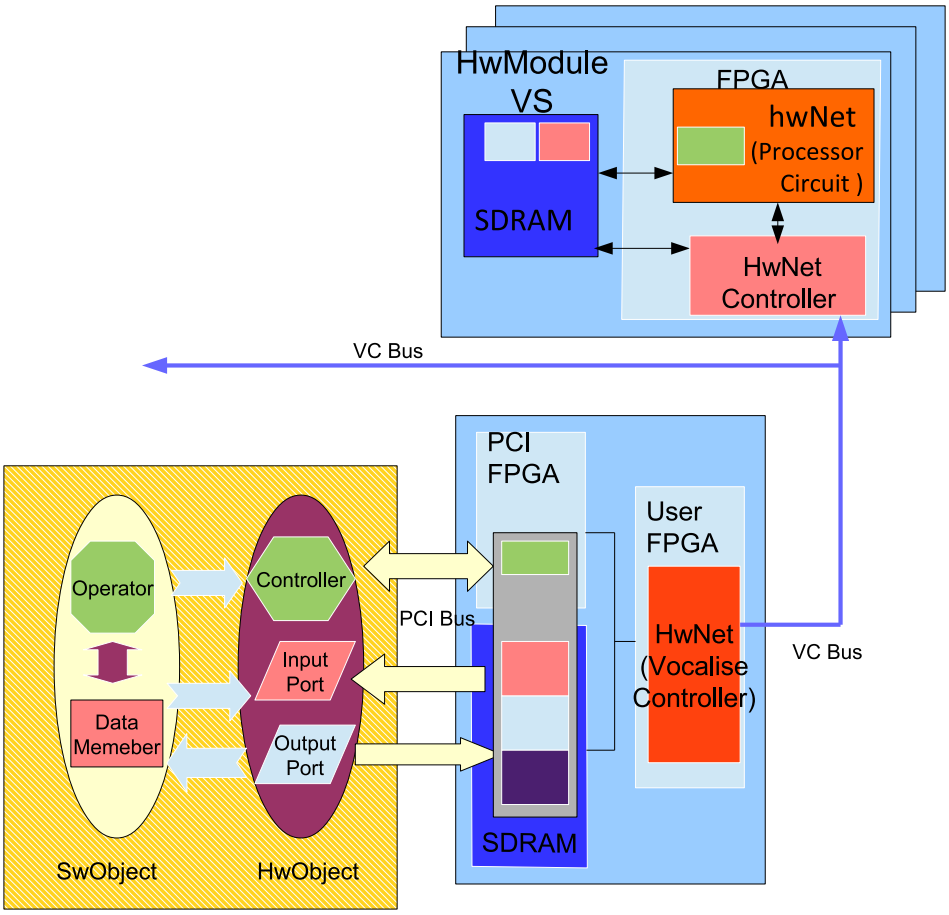

Fig. 2.11 hwObject concept on Vocalise.

seen as that a hwNet receives a start signal from host, then starts up the operations to access LM and compute application which are controlled by its own state machine.

The Figure 2.11 shows the data process flow on Volcailse system. Data communication among host and FPGA array are implemented on following two steps.

1st step, the data in main memory are send to local memory on hwModule V2 by using Intput/Ouptut port of hwObjects on host via PCI Bus.

2nd step, a controller circuit as a hwNet, which is configured on hwModule V2, receives commands from host and switches to the corresponding execution status for data communication. Then, the data commutations among hwModule V2 and VSs are implemented on hardware layer, the controller circuit on V2 access the data on local memory

on hwModule V2, and communicate data/command with hwNet on VSs).

Through above two steps operations, data in main memory can be physically mapped on distributed local memories on FPGA array. When host starts to operate applications, it sends start commands to hwModule V2 via PCI Bus. Then hwNets on hwModule V2 decodes the commands and outputs start signals to each VS on FPGA array on VC Bus protocol. The hwNet on hwModule VS receives the starting signal, then implemented process circuit works on operation state to complete computation controlled by its state machine. The VS send a end signals to V2 when completed computation via VC Bus. Until hwModule V2 received all end signal from FPGA array, a end signal is send to inform Host via PCI Bus.

2.3.3 HwObject interface in Vocalise system

HwObject interface is hwObject's class · object on Host PC. It is allocated on main memory of host PC. Software programs can call hwObject class in C++ language to implement to access the substance of hwobject: hwNet on hwModule. Therefore, this class of hwobject is a interface of hwObject on hwModule(hwNet).

On Vocalise system, the hwObject interface assess method is show in Figure 2.12. To implement circuits configuration, access, control and application operation on Vocalise system, there are two layers of hwObject concept on system (shown in Figure 2.12).

■ *Vocalise_Conrtol_hwObject* This is an hwObject for solving control on Vocalise. There, we designed a Vocalise controller module circuit as a hwNet implemented on hwModule V2. Thus hwNet correspond to a hwObject model named as *Vocalise_Conrtol_hwObject*. In other words, the management functions of FPGA array as applications are implemented by *Vocalise_Conrtol_hwObject*. The yellow area in Figure 2.12 is *hwObject_Vocalise_Conrtol* model.

There, functions of *Vocalise_Conrtol_hwObject* call hwNetDriver to execute requests of instruction and acquisitions of status. On FPGA side, the hwNetDriver as I/O space registers forms, can generates formats for appropriate command, requests to access hwNet. HwNetManger can grasps the status of Object hwModules, avoids access con-

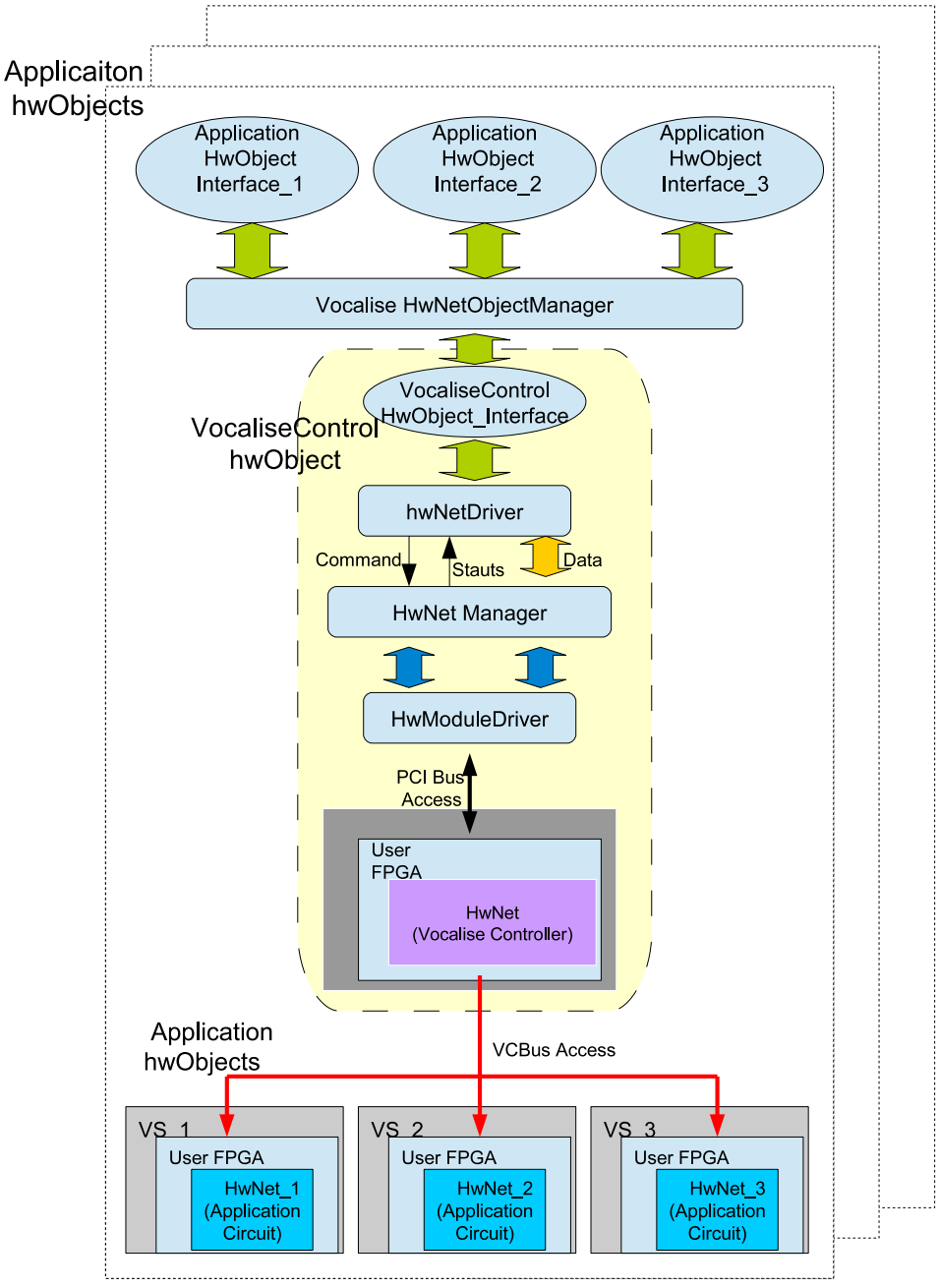

Fig. 2.12 hwObject access method on Vocalise.

flict among multiple hwObject. By using hwNetMnager, request of hwNetDriver can be executed to hwModule V2 through hwModuleDriver via PCI Bus . Once Vocalise controller circuit receive request of commands/status from host, the circuit will start up to corresponding work mode. Then Vocalise controller executes circuit configuration, data/command/status communication with FPGA array on VC Bus protocol, and these operations are controlled by hardware state machine. It means that the data communication/control among hwModule V2 and FPGA array are hidden on hardware level. When software users want to implement access/control with each hwNet on FPGA array, they just call functions of *Vocalise_Conrtol_hwObject* ,and don't need to learn communication mechanism on hardware level.

■ *Application_hwObject* See from the application calculation level, the virtual circuits for solving application computation are implemented on hwNets(Application processing circuits) on VSs. Each virtual circuit hwNet on FPGA array corresponds to an *Application_hwObject*. The developers can utilize hwNetObjectManagement to manage multiple *Application_hwObject*. Through call functions of *Vocalise_Conrtol_hwObject* model, *Application_hwObject_Interface* can easily implements to access/control corresponding hwNet(applications circuits) and local memory on each VS. The amounts of hwNet(processing circuits) on VS and *Vocalise_Conrtol_hwObject* model form multiple *Application_hwObjects*.

While software users just require to call *Application_hwObjects* , the calculation of application will be operated by corresponding hwNets on FPGA array.

For example, the Figure 2.13 shows application hwOjbects for solving Poisson equation problem with a 2×2 VSs FPGA array. There, each hwObject has ID uniquely corresponding to a hwNet(Process circuit) on VS.

Here, HwObject(0,0,0) to VS(0,0,0), hwObject(0,0,1) to VS(0,0,1),hwObject(0,1,0) to VS(0,1,0), hwObject(0,1,1) to VS(0,1,1). For solving Poisson equation, boundary data among adjacent computing nodes needs to be exchanged at each iteration.

Because development cycle of hwNet often is much longer than software developing cycle. To reduce development difficulty and cost, the design and verification are realized

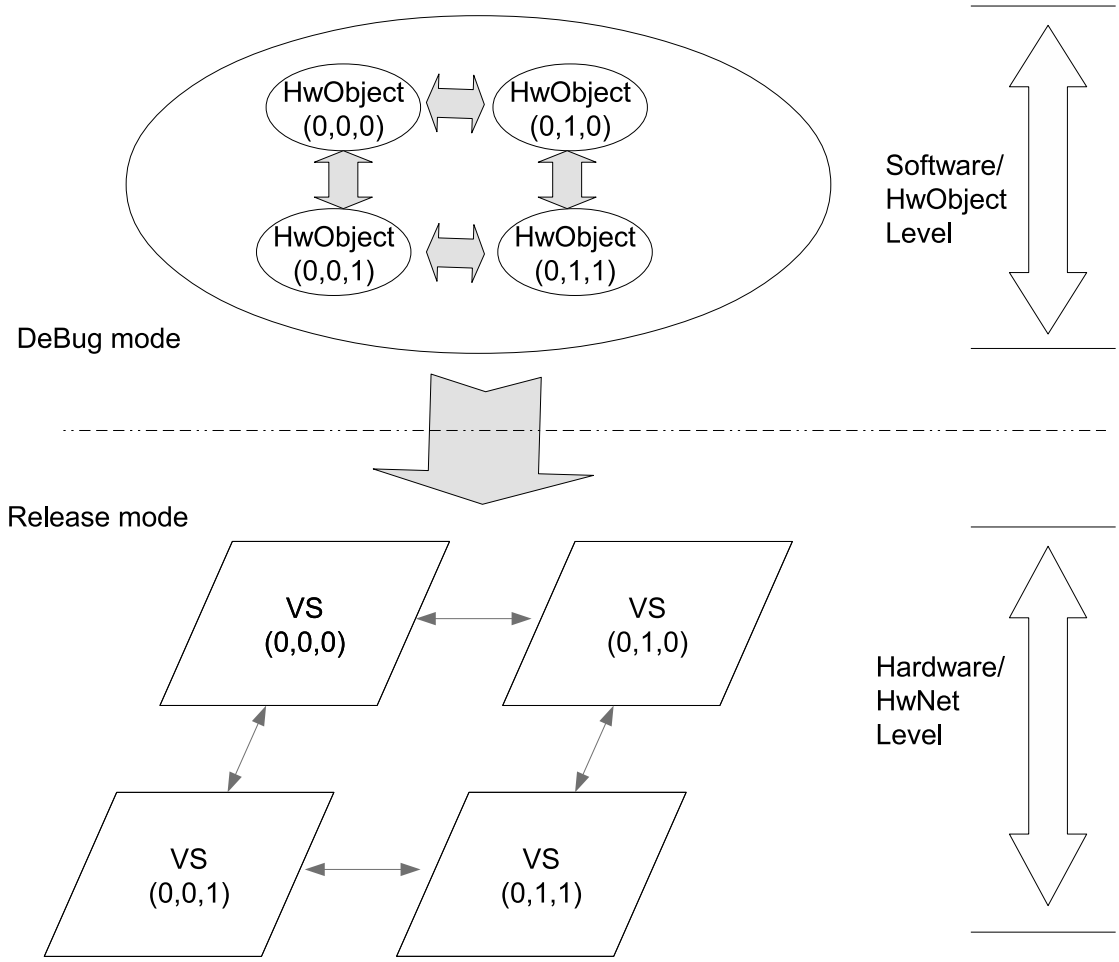

Fig. 2.13 Multiple hwObjects for applications on Vocalise system.

on two stages: debug mode stage and release mode stage. On debug mode stage, the data communication among computing nodes are realized by accesses among corresponding adjacent hwObjects. The hardware developers only require to design main processing circuits on hwNet, and do not redesign the development of data communications, which occupied a great deal of time on the hardware design, among adjacent VSs on debug mode. The host reads result data on each VS to main memory at each iteration, then implements boundary data communication among hwObjects in main memory. The exchanged data was written back to local memory of corresponding VS. Through this method, application debugging can be implemented on hwObject level(software level),

to a certain extent, the hardware's debugging works can be realized as simple as debugging software to a certain extent. The programmers are able to monitor the execution of FPGA, stop it, restart it, set breakpoints etc, not just for using inserts logic analyzer tools, such as Xilinx ChipScope that allows you to probe the internal signals of your design inside an FPGA. While your design is running on the FPGA, you can trigger when certain events take place and view any of your design's internal signals.

Since computations on debug mode need to frequent exchanges data between host and FPGA array, the execution efficiency is very low, the bandwidth of PCI Bus and VC Bus become the bottlenecks. In practical application, the data communications among computing nodes are implemented on hwNet on each VS on release mode. On the development stage, the designers completed the all functions of hwNet. And the debugging of application implemented on hwNet level(hardware level), the debug method is same to ordinary hardware design methods by using inserts logic analyzer tools, such as Xilinx ChipScope.

The Figure 2.14 shows a header of hwObject class on one of a apppication hwNets on a VS.

The implementation of hwObject's class consists of following factors.

1) Implementation of hwNet: implementation virtual circuits on hwModule V2, Bridge VS and Process VS. The member function $Simplelogic$ is used to configure a controller hwNet on hwModule V2, and member function $WriteExec$ is used to implement a hwNet(process circuit) on designated VS. When hwNets are download on VS, the initialization of hwNet can be achieved by the member function $INIT$ and $ResetVS$.

2)Write input data on Local memory, and read output data when completed the computation. By using member functions of $WriteVSSDRAM$ and $ReadVSSDRAM$, the host can achieve write/read local memory on any VS. The member functions of $StartVSWBRAM$ and $StartVSRBRAM$ can implement directly write/read the BRAMs on any selected hwNet on FPGA array. $ExecWrite$ and $ExecRead$ implements to write/read initial data/ results data to local memory on selected VS for application of Poisson equation.

```
1
2  #ifndef _SIMPLELOGIC1_H
3  #define _SIMPLELOGIC1_H
4
5  #include <VirtualObject.h>
6
7  using namespace ObjMan;
8
9
10 class Simplelogic : public VirtualObject
11 {
12 public:
13     Simplelogic();      // コンストラクタ
14     // 物理実装指定
15     Simplelogic(string hwModuleBoard,int iFpgaNo,int ihwNetNo);
16     // 位置指定なし
17     Simplelogic(string hwNetBase);
18     ~Simplelogic();     // デストラクタ
19
20 public:
21    // Initialization
22     bool INIT();
23     bool ResetVS();
24    // Configuration VS
25     bool WriteExec(char* cBitFileName, int Writemode, unsigned int &Output, BYTE NumOfVS);
26
27     // VCBus Data Communication
28     bool WriteVSCTRL(BYTE ADRS, BYTE SELECT, int DATA); //send control command
29     int ReadVSSTTS(BYTE ADRS, BYTE SELECT);  //read VS's status
30     bool WriteVSSDRAM(BYTE VSADRS, int SDRAMADRS, int ByteNum, int* Data); //write data VS's SDRAM
31     bool ReadVSSDRAM(BYTE VSADRS, int SDRAMADRS, int ByteNum, int* Data); //read data VS's SDRAM
32
33     // for execution processing(computation)
34     bool StartVSPE_Debug( int STAPE); // for debug mode
35    bool StartVSPE_Release( int STAPE); // for release mode
36     // VIBus Data Communication
37     bool ExecVIBus_Debug(BYTE AddrTx, BYTE AddrRx);  // for debug mode
38
39 private:
40     bool m_bLMAllocFlg;
41
42
43 };
44
45
46 #endif
47
```

Fig. 2.14 HwObject concept on hwModule.

3) Control/status of VS's hwNet. The member functions of $WriteVSCTRL$ and $ReadVSSTTS$ implement to control/status on selected VS's hwNet. For application, the member function of $StartVSPE_Debug$ are used on debug mode to start the execution of hwNets. And $ExecVIBus\_Debug$ function implements the data communication among VS with hwObject on debug mode. On debug mode, when completed each iteration, the hwNet executes a stall state, and send result data in BRAM to host. Until the receive exchanged data from host, the hwNet execute the next iteration operation. In practice, the member function of $StartVSPE_Release$ is used to start up computation of Poisson equation on FPGA array on release mode. On release mode, all the operation and data communication among VSs are executed by state machine on hardware layer. When operations on VSs are completed, the hwModule V2 received all end signals from hwNet.

hwNet on hwMoudel VS

Since our system can be applied to numerical simulations based on finite difference methods. Application circuits(Processing Circuits) are implemented on VSs, as shown in Figure 2.15. The hwNet accesses local memory (32 bit-16 MB SDRAM) via an FPGA Inner Bus (FIB). The FIB comprises a 32-bit address/data bus and control signals (SEL, MRDY, SRDY, and WR). The application program can manage up to seven hwNet modules via FIB with hwNet manager module. This makes it easy to control distributed and parallel computing with VSs.

When an FPGA array is used to compute different applications, users only need to re-design the hwNet and hwObject interface, which encapsulate complex FPGA-software interface protocols. All standard peripheral modules are reusable in different applications, which decreases the number of development steps.

These simulations numerically solve the PDEs to model physics-based numerical simulations. By utilizing scalable multidimensional design, different spatial domain problems can be solved by structuring FPGAs of different scale. Depending on calculation requirements, the scale of an FPGA array can be changed to corresponding different dimensionalities and computational scales. For example, we can construct a 2D scalable

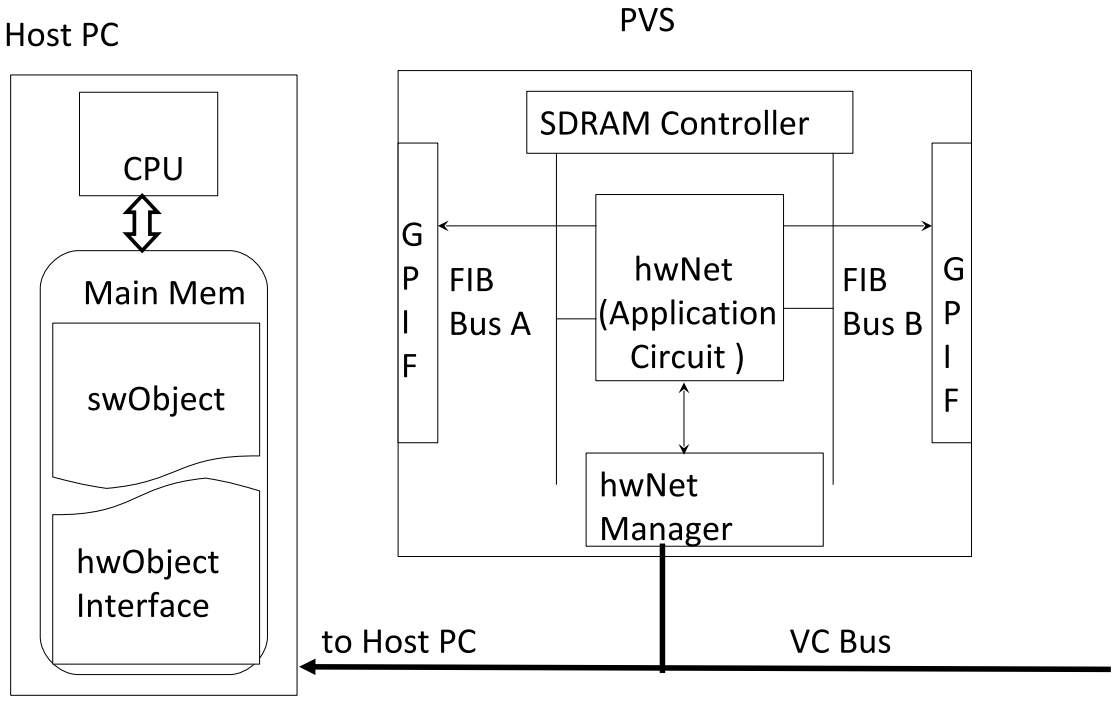

Fig. 2.15 HwNet implemented on 1 VS.

FPGA array for 2D problems or a scalable 3D FPGA array for 3D simulations. The design of the scalable FPGA array enables effective numerical simulation for a particular computational domain.

2.3.4 Design flow of application

The Figure 2.16.a shwos design folw of application on Vocalise system.

On our design method, the development process flow has following design steps.

1st step design; hw/sw complex architecture are considered in the initial stage of application design. The executions of the application are divided into multiple swObjects and hwObjects in parallel. There, the hwObjects without hwNet are actually also operated by CPU.

2nd step design; the developers implement applications on debug mode. The hwObject interfaces are designed in C++ language, and the main processing circuits of hwNet

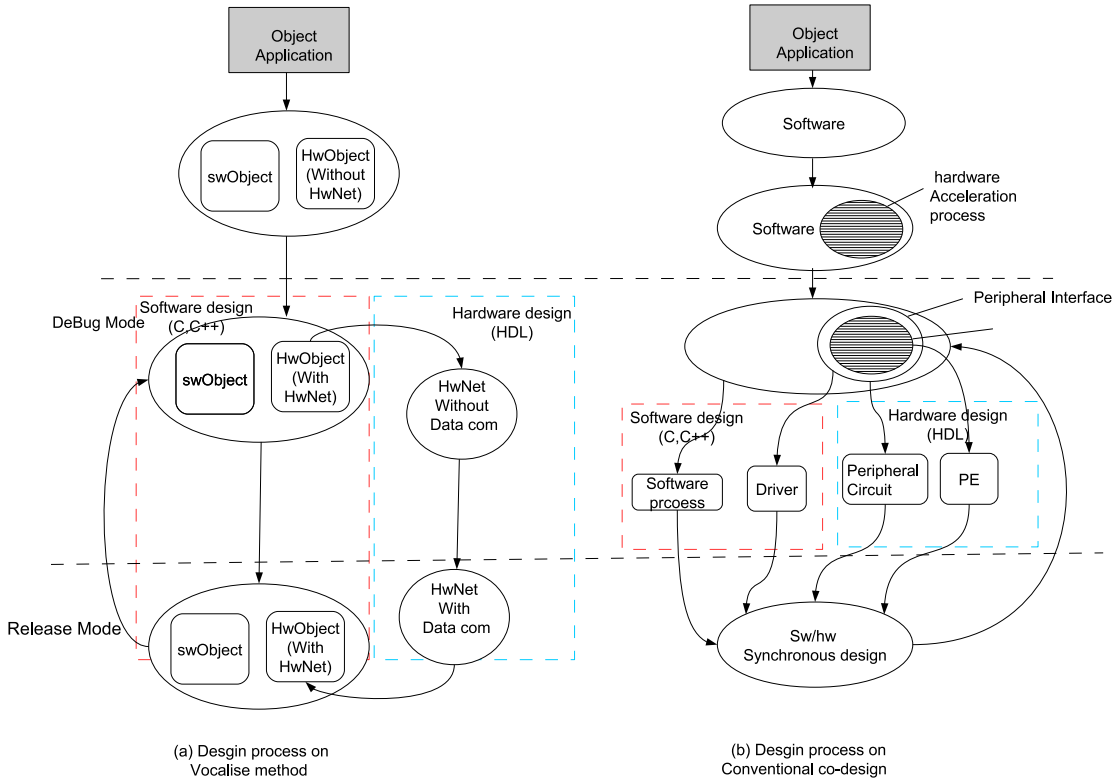

Fig. 2.16 Desing process of hw/sw complex systems.

are completed on hardware RTL design level. On this step, the data communications among hwNets can be realized by software, and execution of hwNet can be implemented to logging an monitoring on software debugging environment. Therefore, it means that debugging of hwNet can be implemented on hwObject level(software).

3rd step design; the developer complete the final design and verification on release mode, when completed debugging on 3rd step and all function of hwNet on hardware register-transfer level (RTL) design level. On the step, the developer realize the hwNet design and debugging by using inserts logic analyzer tools, such as Xilinx ChipScope.

Compared to our approach with a common hw/sw cooperative design(co-desgin) flow on reconfigurable computing systems(shown in Figure 2.16).

In a common hw/sw co-design technology, the applications are realized by software in the initial design stage. Then, the developer analyzes the multiple software objects

in parallel, the processing parts on higher computation overhead of CPU are replaced with optimized specific processing circuits on FPGAs. And developers enquires to design corresponding interface circuits and driver to match the specific processing circuits on FPGAs. For each particular application, peripheral interface circuits and drivers are different. Once solving a new application, the design process need to return to initial steps on hardware design and software design. The interface circuits and driver need to redesign to match the new application specific processing circuits, which will occupy much time and cost on the process of development. And developers implements software/hardware synthesis design on final step of development.

While using our method, all peripheral interface circuits and data communication protocols among host and FPGA array were standardized, and control/access (driver) of FPGA array were hidden in hw/sw complex units. The hwObject interface and hwNet for applications only are required to redesign when change to solve new applications. It is able to effectively simple design and reduce the amount of redesign and the turnaround time.

The developers also can realize hardware design and debugging steps by steps on debug mode and release mode. don't need to return to initial design stage. Development difficulty on debugging of software and hardware can be reduced.

Meanwhile, on the common co-design method, the developers need spend much time on cooperative works among software and hardware on final design step. But on our approach, the parallel operations among objects are implemented on initial steps(1st step). In the following steps, the software developers can focuses on execution efficiency of software design, and hardware developers also can concentrate on architecture design on hardware design and debug steps. The overall development costs and development cycle can be reduced through these improvements.

Design flow of hwObject

The Figure 2.17 shows design flow of hwObject models for solving hwObject- oriented applications.

SwObject and hwObject interface are developed on a integrated software design envi-

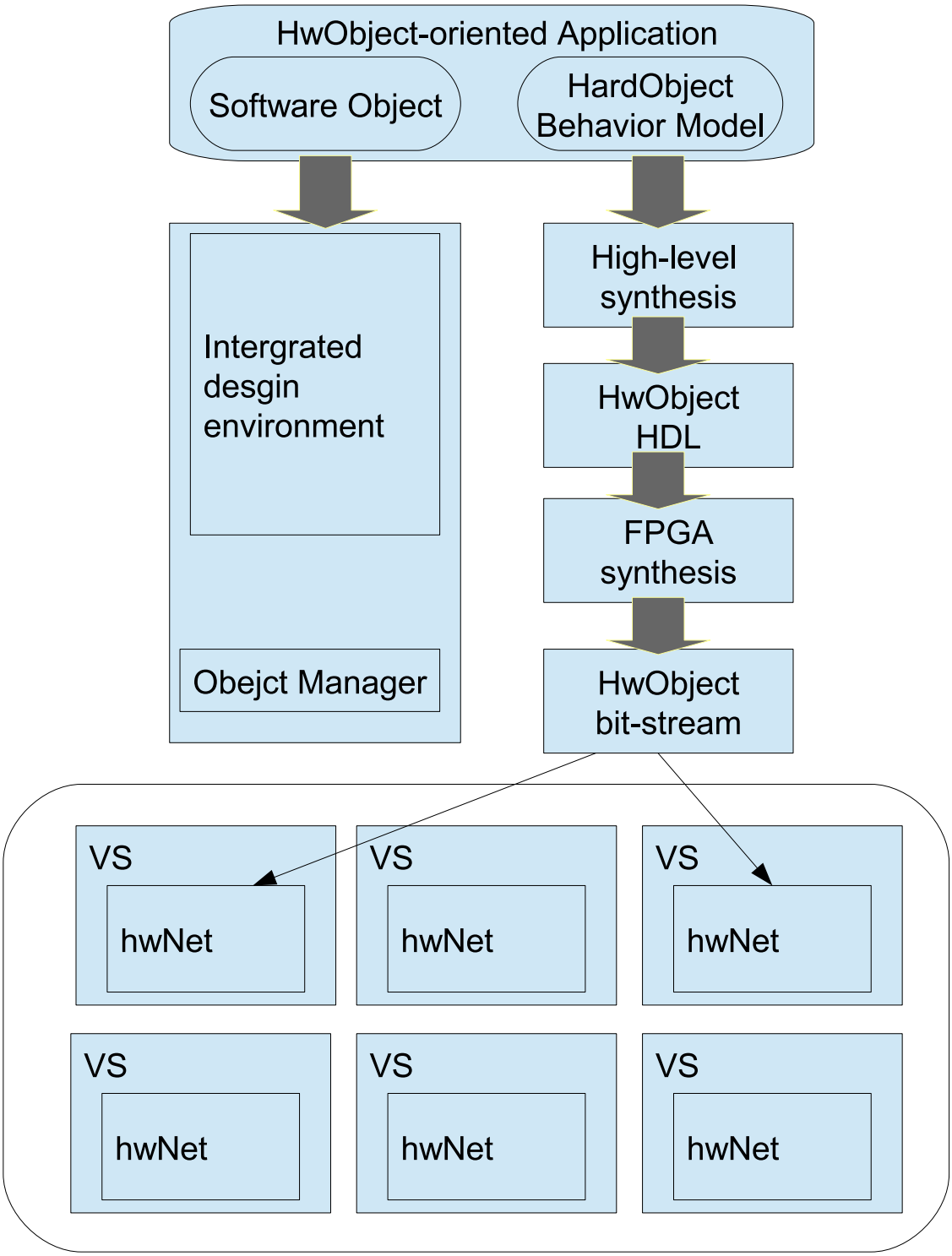

Fig. 2.17 Desing flow of hwObject Model.

```
module HwNet_Poisson3D(
// Port for  read/write Local Memory
oA_LM, iD_LM , oD_LM, oBE_LM, oWE_LM, oREQ_LM, iACK_A, iMODE_LM,
// Port for  read/write via VC Bus
oA_VC, iD_VC , oD_VC, oBE_VC, oWE_VC, oREQ_VC, iACK_VC, iMODE_VC,
// Port for read/write via VIBus
oA_VI_U, iD_VI_U , oD_VI_U, oBE_VI_U, oWE_VI_U, oREQ_VI_U, iACK_VI_U, iMODE_VI_U, //VIBus Up way
oA_VI_D, iD_VI_D , oD_VI_D, oBE_VI_D, oWE_VI_D, oREQ_VI_D, iACK_VI_D, iMODE_VI_D, //VIBus Down way
oA_VI_F, iD_VI_F , oD_VI_F, oBE_VI_F, oWE_VI_F, oREQ_VI_F, iACK_VI_F, iMODE_VI_F, //VIBus Front way
oA_VI_B, iD_VI_B , oD_VI_B, oBE_VI_B, oWE_VI_B, oREQ_VI_B, iACK_VI_B, iMODE_VI_B, //VIBus Back way
oA_VI_L, iD_VI_L , oD_VI_L, oBE_VI_L, oWE_VI_L, oREQ_VI_L, iACK_VI_L, iMODE_VI_L, //VIBus Left way
oA_VI_R, iD_VI_R , oD_VI_R, oBE_VI_R, oWE_VI_R, oREQ_VI_R, iACK_VI_R, iMODE_VI_R, //VIBus Right way
// Synchronizing signal of VI Bus
oEnd_U, oEnd_D, oEnd_F,oEnd_B,oEnd_L,oEnd_R,
iEnd_U, iEnd_D,iEnd_F,iEnd_B,iEnd_L,iEnd_R,
//Control
[31:0]iCTRLA, iCTRLB, iCTRLC, iCTRLD,
//Status
[31:0]oSTTSA, oSTTSB, oSTTSC, oSTTSD,
//system line
iCLK, inRST
);
```

Fig. 2.18 I/O interface on HwNet(Verilog HDL).

ronment, such as Borland C++.

To hardware developer, only hwNet circuits on VS need to redesign for a new specific application problem. In hardware development process, hardware developer usually use register-transfer level (RTL) description to design a hwNet with EDA tools (such as Xilinx ISE) in VHDL/Verilog language. Recently, high-level synthesis (HLS) design have developed. High-level synthesis works at a higher level of abstraction, starting with an algorithmic description in a high-level language such as System C and Ansi C/C++. The designer typically develops the module functionality and the interconnect protocol. The high-level synthesis tools handle the micro-architecture and transform untimed or partially timed functional code into fully timed RTL implementations, automatically creating cycle-by-cycle detail for hardware implementation. The HLS method lets hardware designers efficiently build and verify hardware, by giving them better control over optimization of their design architecture, and through the nature of allowing the designer to describe the design at a higher level of tools while the tool does the RTL implementation.

In order to simplify hwNet's development, we provide a standardized design on hwNet's I/O interface. The Figure 2.18 shows declarations of a hwNet's inputs/outputs interface for solving 3D Poisson equation. The hwNet communicates data with local memory, other hwNets on an VS, other VSs, Host PC through the Memory Type Ports

(MTP) on FIBus protocol. There, a hwNet has a MTP LM for accessing to LM(line 3), a MTP VC for access via VC Bus (line 5), 6 MTPs for data communication via VI Bus(line 7 to line 12). And the signals of iEnd and oEnd (line 14,15)is I/O interface for synchronizing multiple VSs via VI Bus. The signals shown in (line17,line19) are control/status I/O interface for host which can be defined by user. The signals shown in line 21 are system line. According to the such standardized interface, the designer can esaliy to implement a new application through the simple application hwNet. They don't need to put a lot of effort on data communications among hwNets, just needs to redesign the process elements for solving application, and concentrate on algorithm and architecture of hwNets. Meanwhile, because of VS and hwModule V2 equip the same Xilinx Spartan-3 4000 FPGA, and the hwNet on hwModule V2 and hwNet on VS adopt similar standardized interface design based on FIB protocol. Amount of hwNets of applications implemented on hwModule V2 on previous works can be easily ported to VSs on Vocalise platform. Through the design method of hwObject, the development cost and cycle also can be effectively reduced on hardware level design.

Chapter 3

Architecture and Implementation of Vocalise System

Based on Hw/Sw complex architecture, we designed a HPC system with multi- dimensional FPGA array: Vocalise . The system consists of an FPGA array as the core component for operating applications in the proposed system and connect Host PC through hwModule V2 (shown in Figure 3.1). The proposed FPGA array adopts distributed and scalable design and consist of massive small FPGA cards.

3.1 3D FPGA array

Figure 3.2-3 shows a photograph of the 3D FPGA array, comprising 64 FPGA Boards ($4 \times 4 \times 4$). This is a promising approach that provides bandwidth-aware structures and easy to achieve high-efficiency data communication between multi-FPGAs for multidimensional computational problems.

For a distributed computing system. The bandwidth between massive processors is very important, the communication between chips usually become bottleneck to affect

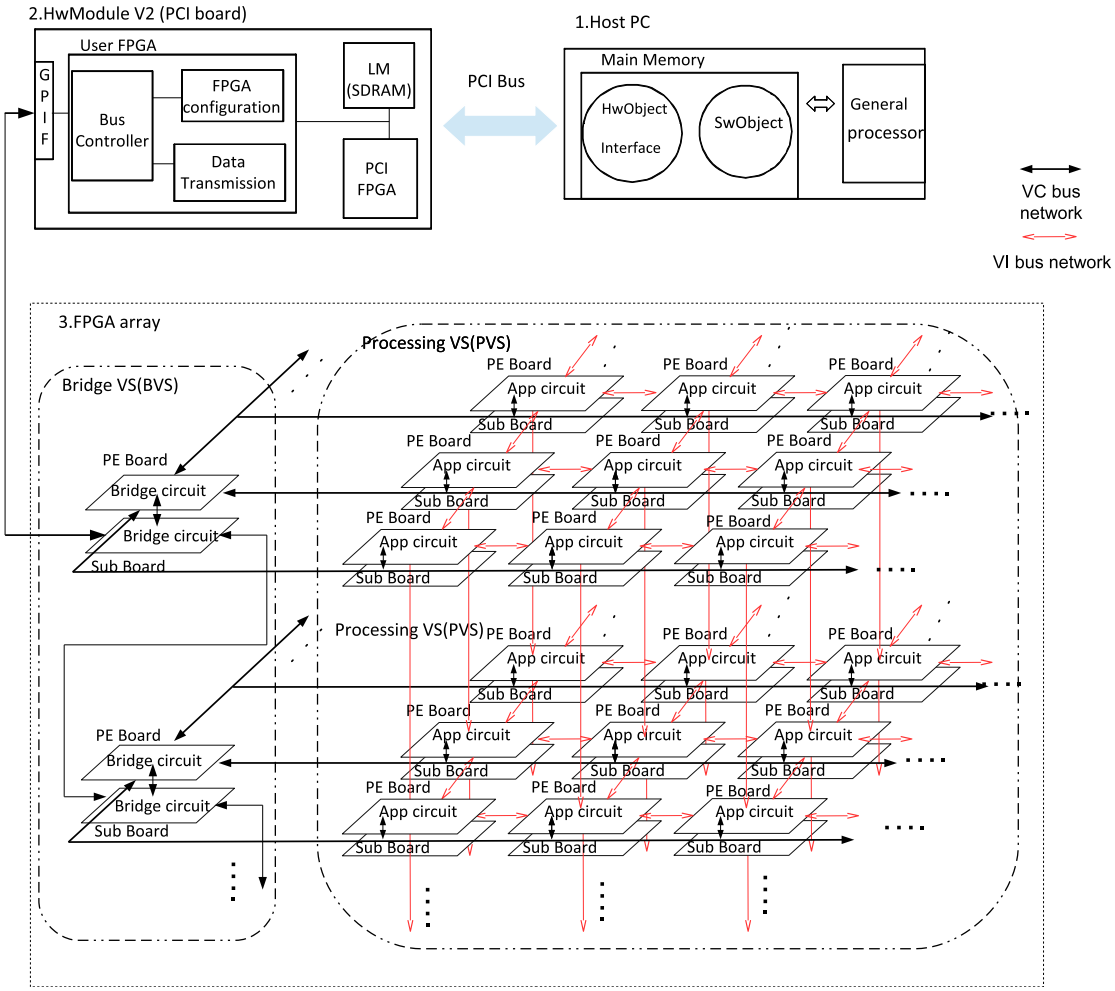

Fig. 3.1 Overview of Vocalise

system performance. In our 3 dimensional FPGA array, as the dimensions of the FPGA array grow, the off-chip bandwidth of FPGAs is boosted. For instance, a single 32-bit GPIF I/O provides 532 MB/s bandwidth at 133 MHz. Thus, the off-chip bandwidth of VS achieves 3,192 MB/s via six-way channels with 3D connections. This is higher than Maxwell [41], where in the FPGA Board connects to the CPU using a PCI/PCI-X bridge that is capable of 64-bit, 133 MHz operation in PCI-X mode. The configuration has a peak bandwidth of 600 MB/s, which is a potential performance bottleneck for Maxwell.

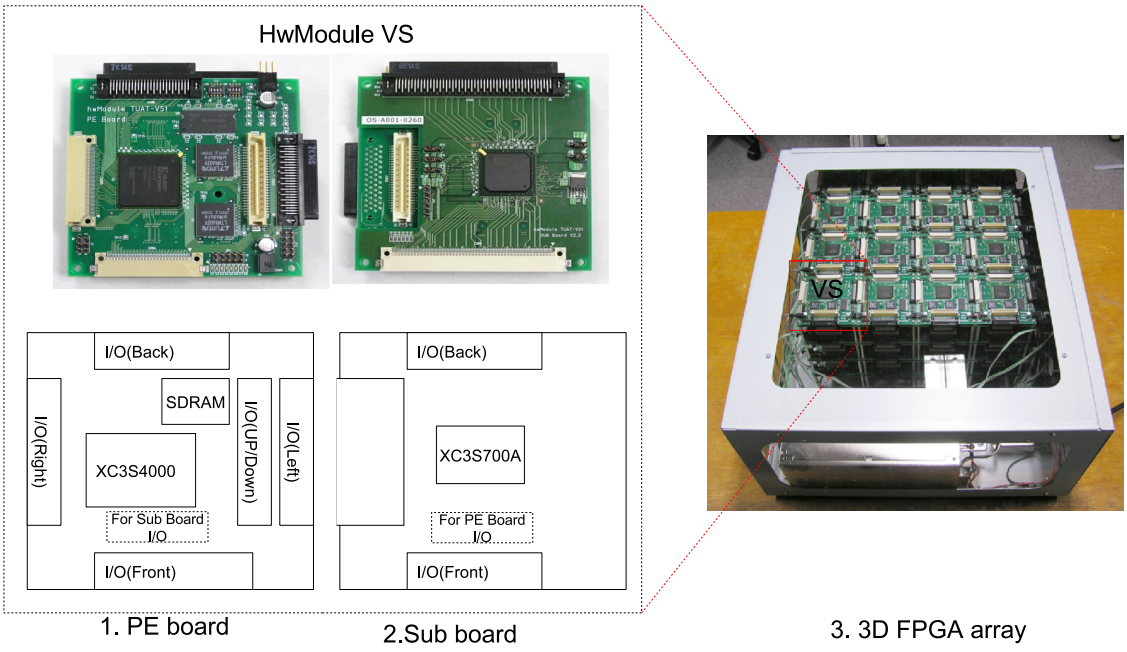

Fig. 3.2 A HwModule VS (Left) and a ($4 \times 4 \times 4$ VSs) 3D FPGA array (Right)

3.1.1 Bridge VS (BVS) and Process VS (PVS)

The overall Vocalise system structure is shown in Figure 3.1. The hwModule VSs were divided into two types depending on their implemented function. We exploit the hwModule VS for extension of V2's I/O channels to achieve implementation and data transmission of the multidimensional FPGA array,. These VSs which implement bridge circuits, is named "Bridge VS" (BVS) Board (Figure 3.1- 3). These Bridge VSs are used to connect massive VSs via the multidimensional GPIF I/O. The VS that implements the application circuit is named the "Processing VS" (PVS) Board (Figure 3.1- 3). The host PC can be connected to as many as 32 PVSs (4 rows $\times$ 8 VSs) through a single Bridge VS. There are massive FPGAs in Vocalise system. Through two kinds of network: Vocalise connection Bus (VC Bus) network and Vocalise inner Bus (VI Bus)network, the system connects all FPGAs and implements circuit configurations, managements, com-

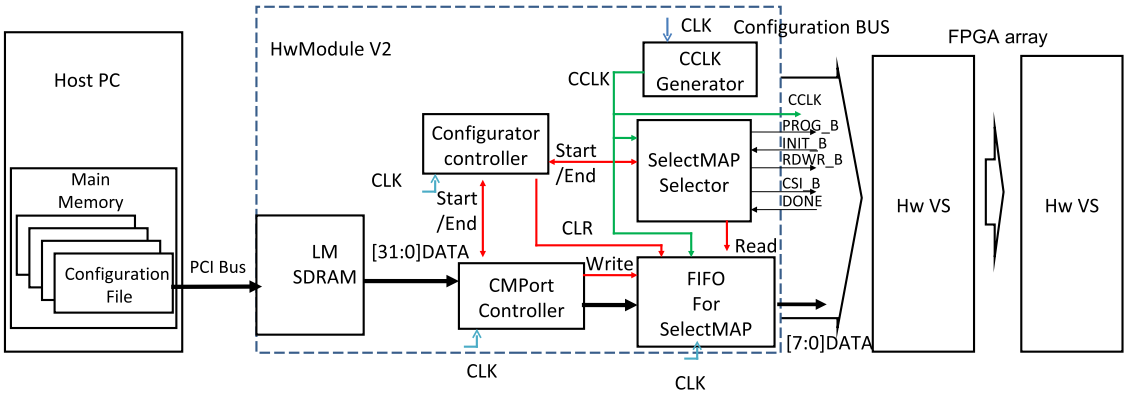

Fig. 3.3 SelectMap configuration element.

munications and applications on FPGA array.

In the following sections, we describe the function and implementation of the two networks: VC Bus network and VI Bus network.

3.2 Vocalise connection bus (VC Bus) network

In our design, the implemented host PC circuit configuration, data communication, and management of FPGA array is performed via the Vocalise Connection Bus (VC Bus) network. Since off-chip I/O bandwidth is significantly limited compared to the internal wires, the single GPIF I/O equipped on the hwModule V2 is only 58-bit width; thus, we enable a VC Bus line to work in two switchable modes: configuration bus mode and data/command bus mode.

3.2.1 Circuit configuration

In the Vocalise platform, a configuration solution based on SelectMAP configuration schemes is provided to minimize configuration time and maximize flexibility. Multiple FPGAs can be configured using the SelectMAP mode and can be made to start-up simultaneously.

The SelectMap(Slave Parallel Mode) configuration is a slave parallel mode; it supports the fastest configuration of Xilinx Spartan-3 FPGA Family. [32], [33] It is able to users to program multiple FPGA devices through an external host, such as a microprocessor or microcontroller, writes byte-wide configuration data into the FPGA.

By using the implemented configuration circuit, we can achieve to configure a multi-dimensional FPGA array via VC Bus, the configuration bus requires an 8-bit data-line and a 14-bit control-line. The Figure 3.4 shows signals of SelectMAP configuration.

CLK Clock line.

BusMode Swicth signal of Bus Mode(High: Configuration Bus Mode, Low: Data/Command Bus Mode.)

REQ Request signals from master side.

ACK The acknowledge signal from bus arbitration organization.

SEL Select signal, master side outputs active-high when master gains access authorization of VC Bus to objective boards.

A/D Address and Data signals.

FRAME FRAME is active-high, A/D line output objective VS's address. FRMAE is active-low, A/D line output configuration data.

The circuit configuration technique for a 3D FPGA array is achieved by the following steps, which is shown in Figure 3.5.

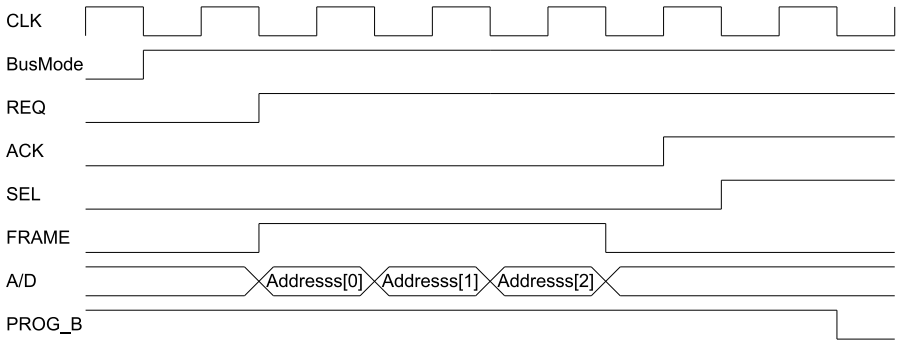

Fig. 3.4 Waveform of SelectMAP configuration .

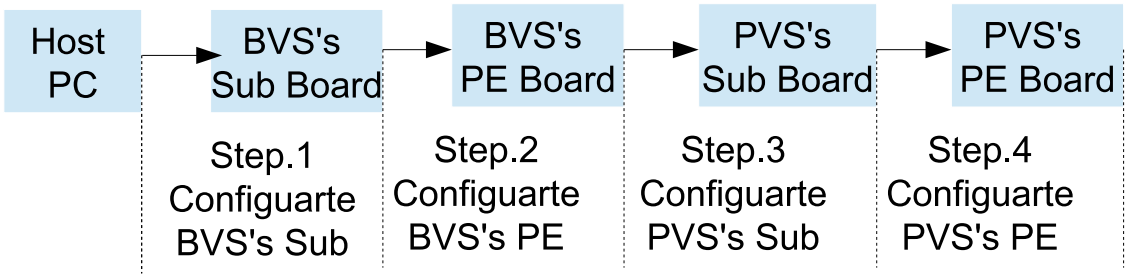

Fig. 3.5 The operation steps for configuration of a 3D FPGA array.

1). Configure BVS's Sub Board: The hwModule V2 configures a bridge circuit (shown in Figure 3.12a) in the nearest connected Sub board of BVS , and then the host PC configures the same bridge circuit on the next Sub Board of BVS through the former configured bridge circuit on the configured Sub Board. The system must complete circuit configuration n times to install n BVSs.

2). Configure BVS's PE Board: The host PC can configure a bridge circuit (shown in Figure 3.12b) on PE boards of BVS in parallel with its Sub Board.

3). Configure PVS's Sub2 Board: The host PC can select any connected Sub Board of PVS and configure the selector circuit on a target FPGA through the BVS bridge circuits. Using the same method to configure the BVS's Sub Board, the host PC can configure the selector circuit on any row of Sub Board.

4). Configure PVS's PE Board: The host PC can easily download specific application circuits in parallel with any selected PE Board of PVS via the VC Bus. Each PVS has a unique ID; thus, through ID signals, the selector circuits on Sub Boards of PVS can determine whether to configure their associated PE Boards. Consequently, it is possible to program different configurations for any FPGA in any row. This provides additional flexibility and enables the user to program different configurations in the FPGA array. The bridge and selector circuits, which are implemented on the BVS and the PVS's Sub board, are intrinsic peripheral circuits of the system. Circuit configuration is only required initially. Users do not need to repeat Steps 1-3. Only the application circuits on

the PVS's PE Board need to be configured when operating other applications.

3.2.2 Configuration circuits

The system uses a SelectMap configuration control circuit (Figure 3.3) on hwModule V2 to write byte-wide configuration data to all FPGAs via VC Bus, which works in the configuration bus mode. In the SelectMap configuration control circuit consists of following unit.

- CCLK Generator
- CMportController
- Configurator controller unit
- SelectMAP selector
- FIFOforSelctMap

CCLK generator provides the clock signals for configuration. The circuits bit stream is transferred to stored in Local Memory by PCI FPGA on hwModule V2. When completed the initialization of objective FPGA, the CMportController read the bit stream data from Local Memory to FIFOforSelectMap. Finally The circuit's bit stream data is transmitted to connected FPGA via GPIF IO from hwModule V2. The Configurator controller unit is a main control circuit for SelectMAP configuration. It is used to achieve status identification and control of objective FPGA. SelectMAP selector unit is used to input/output the control signals of selectMAP configuration.

3.2.3 Data transmission and FPGA array management

The VC Bus works in the data bus mode until the system has completed circuit configuration of the FPGA array. The host PC is able to transmit data/command to any objective PVS and read data/status from objective PVS.

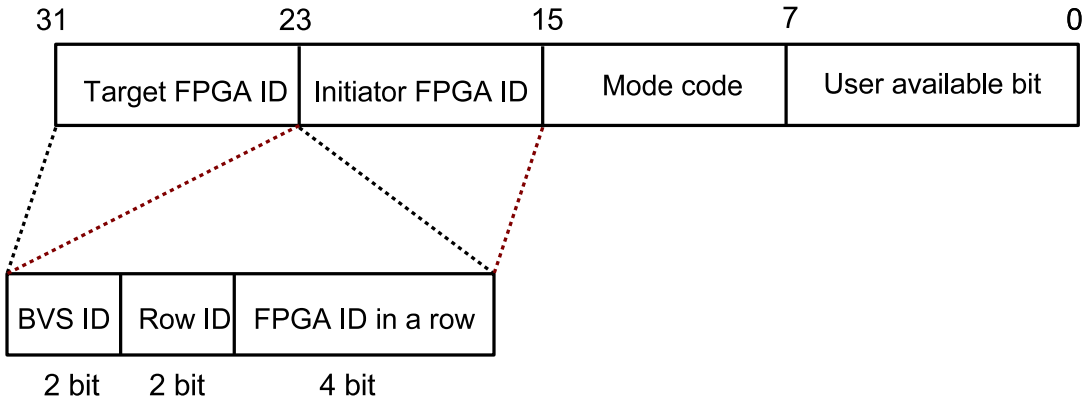

Fig. 3.6 PVS address signal line format in VC Bus network.

Address management of PVSs

Each PVS has a unique ID in the VC Bus network. The PVS address signal format is shown in Figure 3.6. The 32-bit PVS address signal comprises the target board ID(8-bit), initiator board ID (8-bit), 8-bit mode code line, and 8-bit user-available bit. The mode code can be identified by slave elements on the PVS in the VC Bus network. It comprises various types of operation such as read data, write data, and transfer commands. When multiple devices request the VC Bus, bus arbitration is realized by implemented bridge circuits on the Bridge VSs.

Communication mechanism

The VC Bus data communication protocol is based on a typical master-slave transmission mechanism and can achieve 32-bit burst transmission, which is analogous to FPGA inner Bus . The bus line is equipped with a 32-bit A/D line, a 6-bit control line, and a 1-bit clock line.

Figure 3.7 shows write and read operation signals among the hwModule and the FPGA array.

CLK Clock signal line.

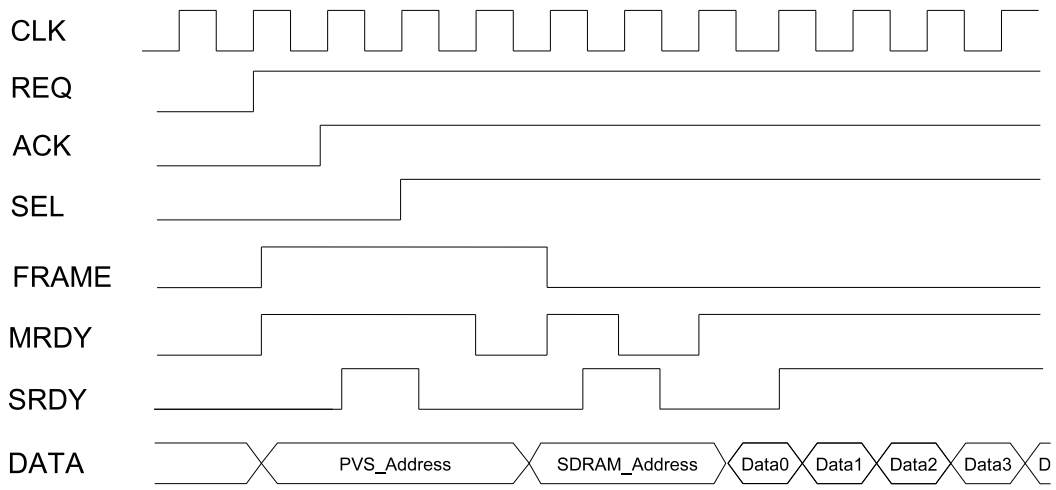

Fig. 3.7 The write operation between host PC and FPGA array via VC Bus.

BusMode Switch signals of Bus Mode(active-high: Configuration Bus Mode, active-low: Data/Command Bus Mode.

REQ Request signals from master side, output to arbitration to access to VC Bus.

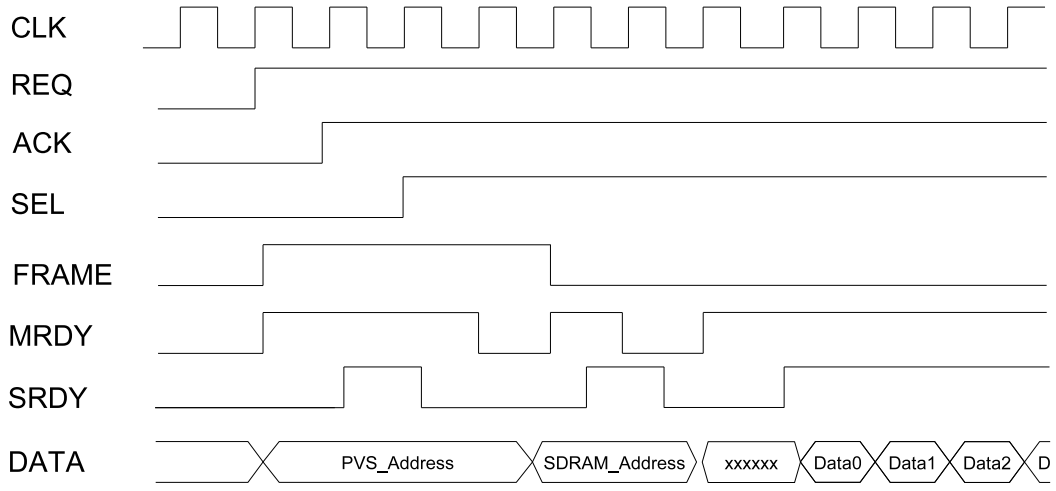

Fig. 3.8 The read operation between host PC and FPGA array via VC Bus.

ACK The acknowledge signal from arbitration.

SEL Master side output select signal to objective boards when Master gain access authorization of VC Bus.

A/D Address/data signals.

FRAME The identification signals of address/data line. (FRAME is active-high, outputs address of objective PVS; FRMAE is active-low, outputs communication data/command.

MRDY The ready signal from master. When master is ready, assert the signal to slave.

SRDY The ready signal from slave. When receiving master signal and slave is ready, assert the signal to master.

When host PC communicates with multi-PVSs, the hwModule V2 acts as a master, and sends SEL and MRDY signals to enable use of the VC Bus and broadcast valid PVS addresses to the PVSs as targets. The hwModule V2 waits until the SRDY signals from the target PVSs are received. Subsequently, the hwModule V2 performs a burst read/write operation after the negotiation. With the above approach, the host PC can execute write/read operations to each distributed SDRAM on the FPGA array. Moreover, the applications running on the host PC send specific commands to any PVS, and read status of each PVS.

3.2.4 Implementation circuits for data communication on VC Bus

Data communication modules are implemented on hwModule V2 and each PVS PE board (Figure 3.9) .

The modules consist of following units.

- VCBusConnector
- VCBusMater
- VCBusSlave
- HNController

The Table 3.1 shows the circuit scale of controller circuit on hwModule V2.

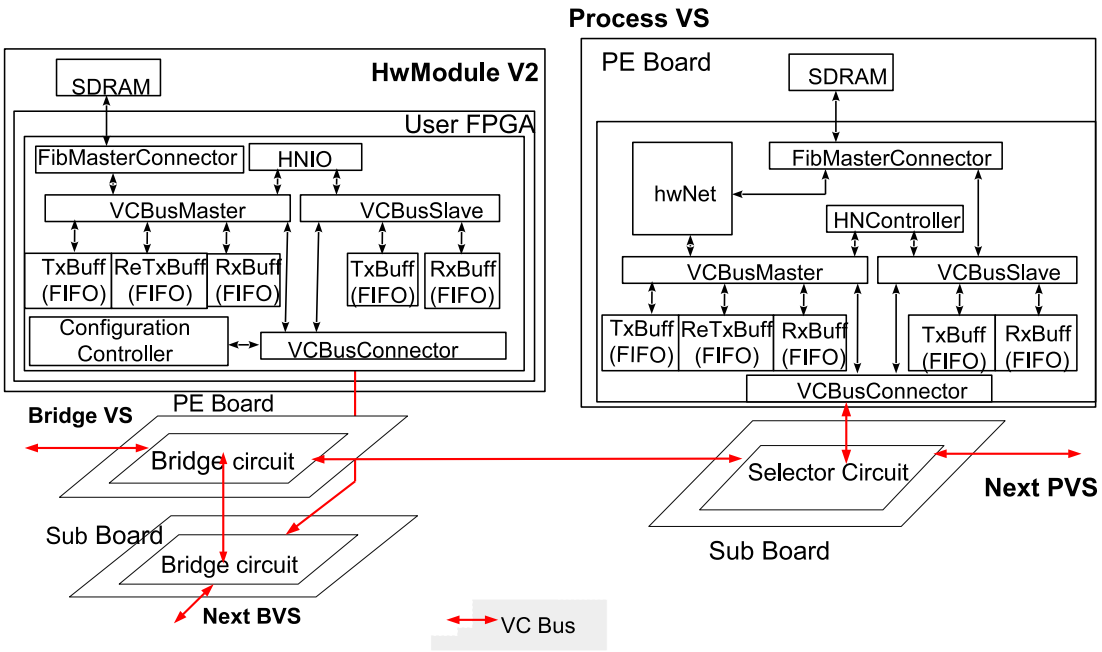

Fig. 3.9 VC Bus data communication elements.

VCBusConnector

This module connects external BVS's Sub board and PVS's Sub board. It has lots of IO buffers and registers to convert VC Bus signals from inner FPGA and outside FPGA. Since clock source VC bus is from host PC, we utilize a Digital Clock Manager（DCM）to achieve driver and modulation of VC bus clock signals from external Sub Board, and

Table 3.1 The circuit scale of hwModuleV2

	Used	Available	Utilization[%]
Nubmer of Slice Flip Flops	4,520	55,296	8
Nubmer of 4 input LUTs	5,891	55,296	10
Nubmer of bonded IOB	277	489	56
Number of DCMs	3	4	75
RAMB16s	8	96	8

output the clock signals to other unit: VCBusMater unit, VCBusSlave unit and HNIO unit. The DCM primitive in Xilinx FPGA parts is used to implement delay locked loop, digital frequency synthesizer, digital phase shifter, or a digital spread spectrum.[34]

VCBusMaster

VCBusMaster as a master port of VC bus; the unit receives data from hwNet with FIB protocol and sends the data to connected FPGAs; it converts FIB protocol and VC Bus protocol; and it has the slave function of FIB, and the master function of VCBus. The control signals of hwNet on hwModule VS are also transmitted via VCBus through the interrupt action of HNController. Since the unit is an asynchronous module, which has double clock source; FIB clock is from hwNet (inner FPGA), and VC Bus clock signal is from hwModule V2(host PC). The unit consists of a FIB controller, a VCBus controller, two asynchronous FIFOs for writing/reading, and a synchronous FIFO for resending data.(shown in Figure 3.10)

The FIBController receives a request from hwNet or HNController unit, outputs the request to VCBusController; The VCBusController can distinguishes the request is control or data communication (read/ write operation), then send commands or data based on VCBus Protocol method via VC Bus links.

VCBusSlave

VCBusSlave as VC bus Slave port; it's operation is opposite to VCBusMaster unit, which implements a slave function of VC Bus, and a FIB master function. It can receives signals from external FPGAs with VC protocol as a slave, then converts the signals and outputs to hwNet with FIB protocol. Moreover, control commands for hwNet on PVS are also send to HNIO unit via VC bus. The clock source of the unit is same as VCBusMaster unit, FIB clock source is from hwNet, and VC Bus clock source is from hwModule V2 on Host PC. We implements the unit with a FIB controller (clock signal source: FPGA inner clock), a VCBus controller (clock signal source: VC Bus clock from host), two asynchronous FIFOs for writing/reading data.

The unit is driven by VC Bus signals. When a request come from external FPGA, the

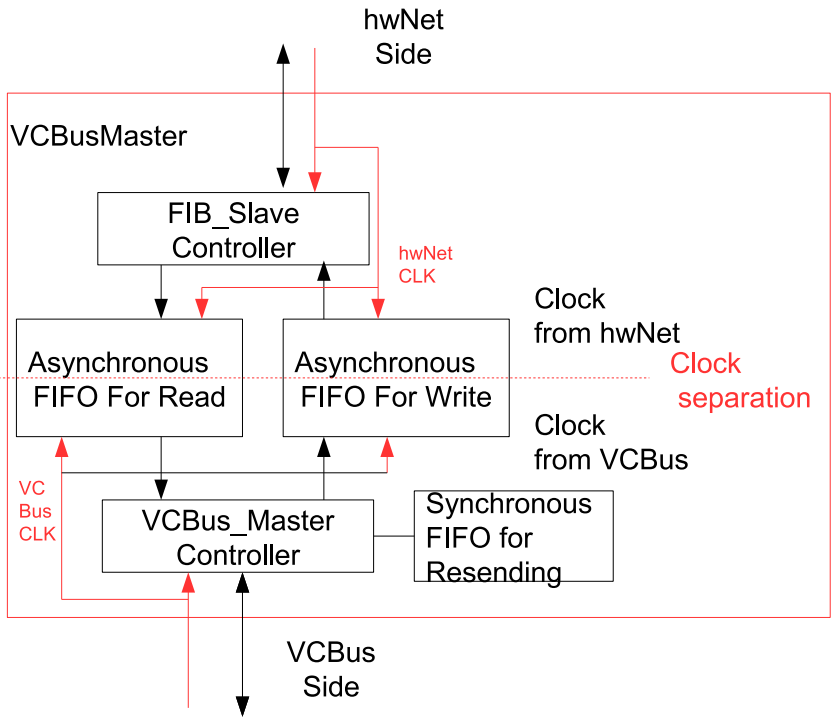

Fig. 3.10 VCBusMaster Module.

unit firstly match its Board address to the address signal from VC Bus. If the match result is true($requestaddress = Boradaddress$), the VCBusController decodes the command signals and outputs it to HNController, When host operates data read/write executions, the data is transmitted to PVS on VC Bus Protocol, and finally send data to hwNet with FIBController on FIB protocol method.

HNController

This unit is an controller module of hwNet IO; it is used to store commands (CTRL) from the host PC and the status (STTS) of hwNets. The applications on the host PC send specific commands to any PVS and read the status of each PVS through the implemented. Moreover, the HNController unit, which is implemented on HwModule V2, can stores any status (STTS) signals from hwNet on FPGA array; and these status (STTS) are

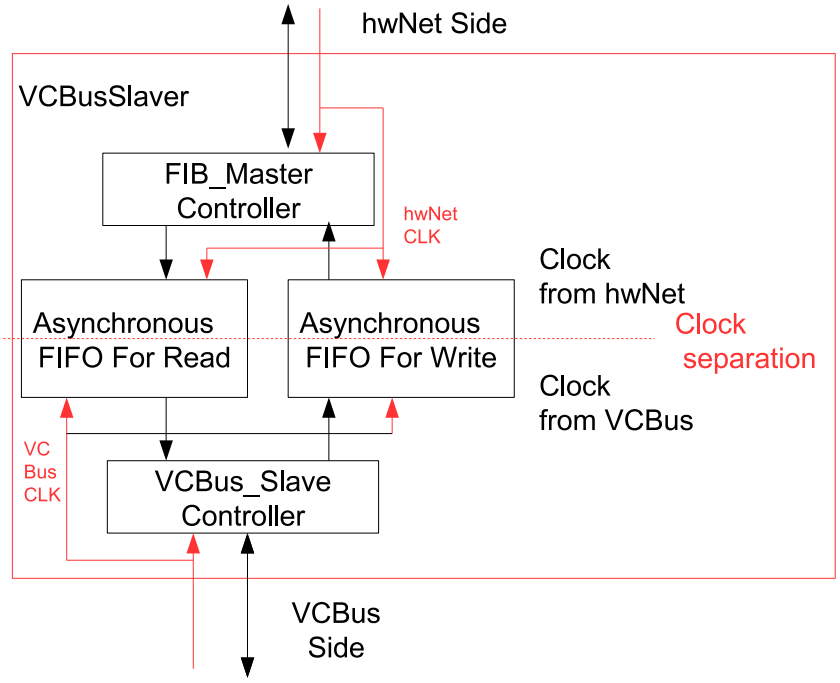

Fig. 3.11 VCBusSlave Module.

stored in BRAM for FPGA array management.

3.2.5 Selector circuits on sub board

Meanwhile, we implemented a selector circuits on Sub Board. The circuit consists of following units.

- IOBUF
- VCBusSwitch
- ConfigurationSwitch

The Table 3.2 shows the circuit scale of selector on Sub Board of PVS.

Table 3.2 The circuit scale of selector on Sub Board of PVS

	Used	Available	Utilization[%]
Number of Slice Flip Flops	255	11,776	2
Number of 4 input LUTs	673	11,776	1
Number of occupied Slices	249	5,888	4
Nubmer of bonded IOB	239	372	64
Number of DCMs	1	8	12

IOBUF

The module has lots of I/O Buffers and registers to achieve to convert the signals among external FPGAs and inner FPGAs; this modules are implemented on Sub Board, and consists of 3 IOBUFs to control I/Os in 3 directions.

1. The FDATA_IOBUF is a controller for IO interfaces on front port from Host PC(hwModule V2).

2. The BDATA_IOBUF realizes to control the IO interfaces on back port which connect to next Sub Board.

3. The SIMDATA_IBUF controls the IO interfaces on SIMDATA port which connect to own PE Board.

Moreover, we implement a digital clock manager (DCM) on the FDATA_IOBUF unit.

The clock signal from hwModule V2 inputs the DCM unit, and through the DCM unit outputs a clock signal as clock source to provide other modules inner FPGA.

SwitchVC

The module is a switch circuit for data/commands of VC Bus on DATA/Control mode; the module can identifies VC Bus address signals, and matches the target Board bit of address signals with its board address. Then, based on distinguishing results, the modules bridge the links among GPIOs in 3 directions (the Front side, the End Side and PE Board side). If the matching result of address is true, the module implements to connect FDATA I/Os with SIMDATA I/Os. If the matching result is false, the module

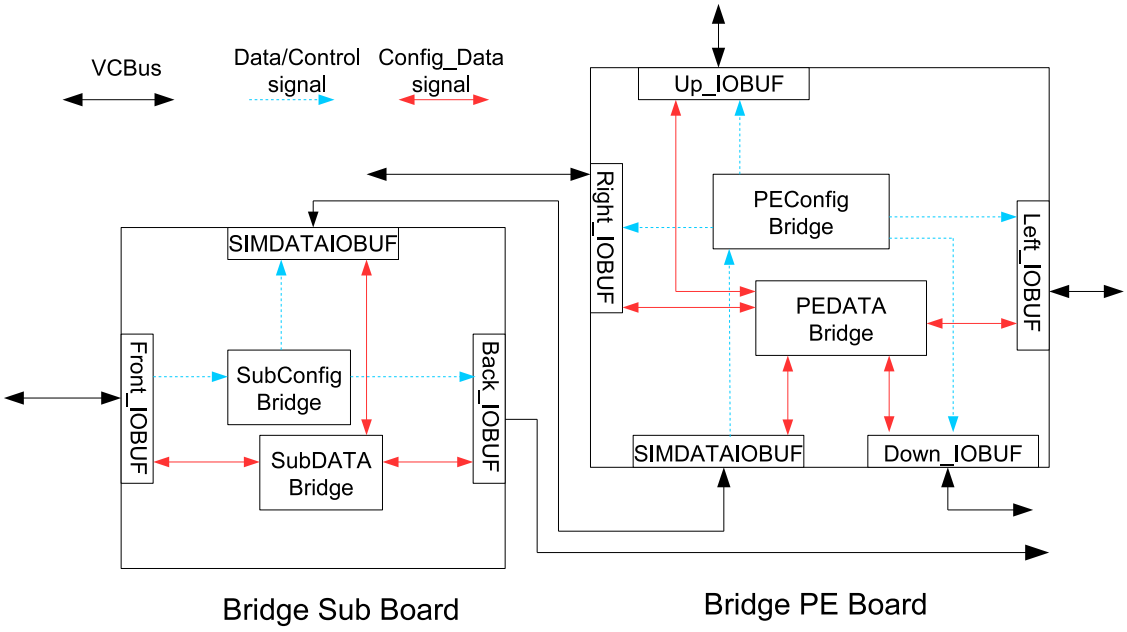

Fig. 3.12 Bridge circuit on Bridge VS. (a: Bridge Circuit on BVS's Sub Board, b: Bridge Circuit on BVS's PE Board.)

implements to link FDATA I/Os and BDATA I/Os. Moreover, we latch the input/output signals of VC Bus with registers, which is driven by clock signal of DCM, for stabilizing VC Bus signals inner FPGA.

ConfigurationSwitch

The ConfigurationSwitch module is a switch circuit for configuration data, works in configuration bus mode; It can identify the ID signal from Configuration Mode, and match the signal with its Board address. When ID signal is the board address, the module connects its PE Board, and accomplishes the circuit configuration for its own PE board.

3.2.6 Circuits design of Bridge VS

Host PC connects an 3D FPGA array through Bridge VSs, and realizes access/control/circuit configuration for the FPGA array. In our design, see from host PC,

a Bridge VS connects 4 rows Processing VSs as a sub-group PVS array (maximum scale: 4/times8 PVS). Each Bridge VS has a unique ID, we enable access any PVSs on a sub-group FPGA array through a Bridge VSs. Meanwhile, the Bridge VSs also are used to be realized to access arbitration of VC Bus; it guarantees the bus permission among host PC and FPGAs in connected sub-group FPGA array through a bridge VS.

The Figure 3.12 shows the block diagram of a Bridge VS. The VC Bus signals from the inner FPGA are divided into configuration signals and data/control signals. The implemented bridge circuits mainly comprises following units.

- Input/Output BUF Units(IO BUF units).
- Config Bridge Unit.
- DATA Bridge Unit.

We also utilize a DCM to realize management and synthesize of clock, phase shift and clock skew. And through the DCM, output clock signals to drive other units inner FPGA. The circuit scales of Bridge circuit on Bridge VS are shown Table 3.3 , 3.4.

Table 3.3 The circuit scale of Bridge circuit on Sub Board of BVS

	Used	Available	Utilization[%]
Number of Slice Flip Flops	197	11,776	1
Number of 4 input LUTs	222	11,776	1
Number of occupied Slices	249	5,888	4
Nubmer of bonded IOB	239	372	64
Number of DCMs	1	8	12

IO BUF units

The IO BUF units, which equip amount of IO buffers and registers, which are used to convert signals between off-chips and inner FPGA, and divide the off-chips VC Bus signals into configuration signals and data/control signals inner FPGA as the bus mode changes.

On the Bridge VS's Sub Board, we implement three IO BUF units: Front_IOBUF,

Table 3.4 The circuit scale of Bridge circuit on PE Board of BVS

	Used	Available	Utilization[%]
Number of Slice Flip Flops	672	55,296	1
Number of 4 input LUTs	895	55,296	1
Number of occupied Slices	762	27,648	2
Nubmer of bonded IOB	299	489	61
Number of DCMs	1	4	25

Back_IOBUF and SIMDATAIOBUF to link on ahead/rear Sub Boards and its own PE Board.

On the Bridge VS's PE Board, five IO BUF units are implemented; one IO BUF (SIMDATAIOBUF) unit links to its own Bridge Sub Board, and other four IO BUF units (Up_IOBUF, Down_IOBUF, Right_IOBUF and Left_IOBUF) are used to link to connected PVS's Sub Board in four directions through four (52 pin to 100 pin) convert boards.

ConfigBridge units

Configuration bridge unit is used to realize bridge function for configuration data; it works in circuit configuration bus mode. The module enables the host PC to implement circuit configuration on selected FPGAs. We implement a SubCongfig Bridge unit on Bridge Sub Board. The unit can identifies ID and configuration command signals from host PC. On Bridge VS's sub board configuration stage, the unit links Front_IOBUF and Back_IOBUF, and sends circuit configuration bit stream data to linked next sub board.

In the circuit configuration stages (Configuration of Bridge VS' PE board, Process VS's Sub board and PE board), if BVS's bit of configuration address is same to the BVS ID, the unit links Front_IOBUF and SIMDATAIOBUF, and sends circuit configuration bit stream data to PE Board. If it is not same to the BVS ID, the unit links to Front_IOBUF and Back_IOBUF, and transmits configuration signals to next Bridge VS' Sub board.

In the same way, we also implement a PEConfig Bridge unit on Bridge PE board. The

unit identifies configuration signals from Host PC through its Sub Board. On the basis of the object PVS address signals, the unit can links the corresponding I/O connections among host and PVSs along the four directions, and sends the configuration bit stream data to the corresponding PVSs.

DATA Bridge units

The module is a data/command bridge unit which works in data communication mode of VC Bus. The module enables the host PC to exchange data with any connected PVS and send control commands to any PVSs. On Bridge Sub Board, the communication data signals, command signals, status signals can be transmitted to SubDATABridge circuit to data among 3 GPIOs. Meanwhile, The SubDATABridge unit also can identifies the address signals and match the bits of Bridge VS ID on the address signals with its Board ID. If they are same, the module links the Bridge VS's PE Board and host PC. If not, the unit links the IO connections of FrontIO BUF and BackIO BUF.

We also implemented a PEDATA Bridge unit on PE Board. A 5-way arbiter of VC Bus is realized on the unit. The unit enables a Bridge VS to achieve to arbitrate the access requirements from host PC and connected 4 rows FPGA array. The host or PVS obtains ACK signal from the arbitrator, then send address signal to PEDATA Bridge unit and assert SEL signal. The unit identifies the target address bits and initiator address bits based on the address signal, and links to corresponding I/Os among target address and initiator address, until data communication and command/status transmit-receive among host PC and object FPGAs are completed.

3.3 Vocalise inner bus (VI Bus) network

In the Vocalise system, one PVS transfers data to other PVSs via the Vocalise inner Bus (VI Bus) network when operating applications (shown in Figure 3.13). The bus line consists of data line (32 bit) and control line (18 bit).

Figure 3.13 shows VIBus network among PVSs.

By utilizing GPIO connectors in six directions, each Process VS can connects adjacent

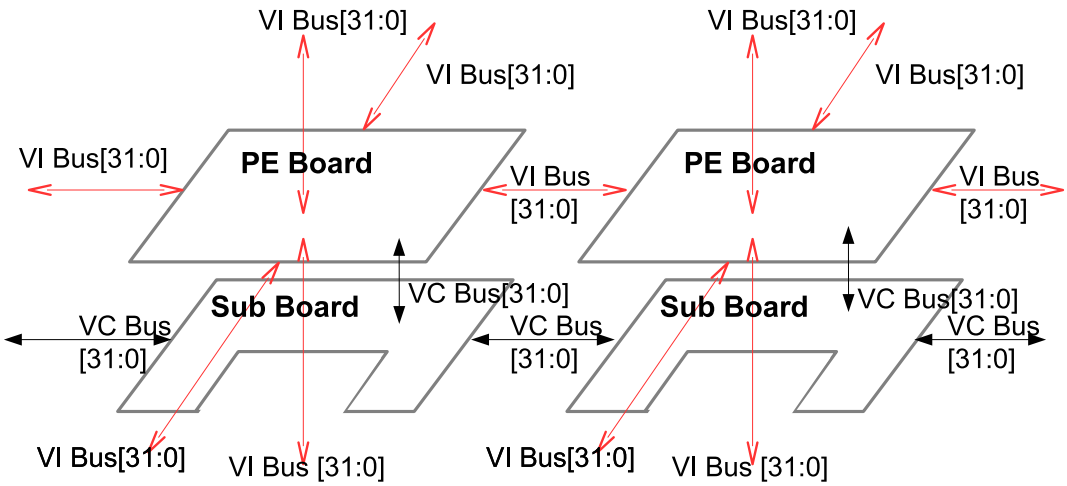

Fig. 3.13 VC Bus and VI Bus connections among Process VSs

6 PVS, and realizes the data communications at the same time with the connected PVSs. Moreover, the route of the VI Bus is variable; it can be changed with the corresponding communication requirements of different applications changes.

3.3.1 Telecommunication mechanisms

To implement the different transfer circuits on PVSs, three types of telecommunication mechanisms, simplex, half-duplex, and full-duplex transmissions can be realized. The telecommunication mechanisms and signals is shown in Figure 3.14

Depending on the requirements of applications, the VI Bus works in two modes. One is point-to-point communication mode, PVS only communicates with adjacent PVSs via the direct connections. The other one is distance data transmission mode; a PVS can communicates data with any PVSs on FPGA array via VIBus.

Point-to-point data transmission among adjacent PVSs

To reduce the circuit design development cycle, we have designed a foundational communication module: the VIBus module, to establish direct point-to-point connection to adjacent FPGAs in each six-way directions. The modules are implemented in each pro-

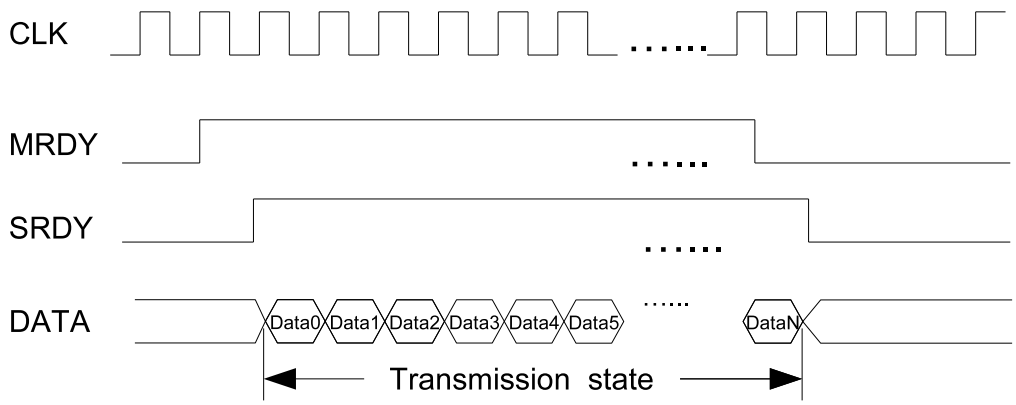

Fig. 3.14 Simplex transmission operation between adjacent FPGAs via VI Bus.

cessing FPGA, and data is transferred to the nearest adjacent FPGA using three types of telecommunication mechanisms, i.e., simplex, half-duplex and full-duplex transmissions. Table 3.5 shows the essential signals for implementing the three telecommunication modes.

In our experiments, the data communication between two FPGAs was implemented at 133 MHz, which is double that of the highest execution frequency of hwNet (66 MHz). The single connector I/O bandwidth was 4.26 Gbps ($\times$ 32 bits) at 133 MHz. Figure 3.14 shows the signal timing design of the simplex mode. In a fully connected network topology, each FPGA connects six FPGAs; thus, the maximum theoretical bandwidth is 25.56 Gbps (6 ways $\times$ 4.26 Gbps) at133 Mhz among interconnected FPGAs.

Table 3.5 Essential VI Bus signals for telecommunication

Telecommunication Mode	Simplex	Half-duplex	Full-duplex
Data line (bit)	32	32	Rx(16) + Tx(16)
control line (bit)	2	7	8
clock line (bit)	1	1	2
total (bit)	35	40	42

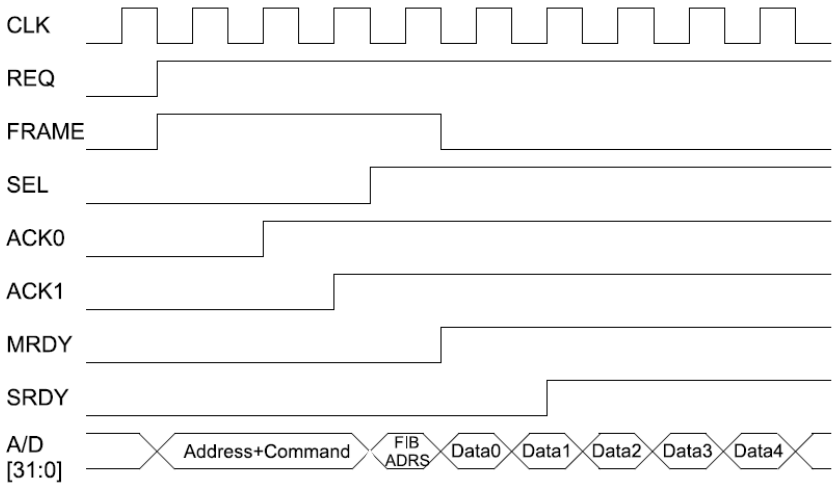

Fig. 3.15 Write operation signals via VIBus（half-duplex）in distance transmission mode.

High-speed distance data transmission of VI Bus

In massive applications, the parallel computing nodes (FPGAs) map on a complicated network.

When PVS communicates with other PVSs which are not adjacent, it works in distance communication mode. One PVS can send data to target PVS via intermediate PVSs. The intermediate PVSs as repeaters transmit data and keep the signal strength to target PVS.

Through utilizing the point-to-point data transmission and distance data transmission, various data communication network among multiple FPGA can be easily realized. We realized the distance data transmission with half-duplex and full-duplex transmission mechanisms. (The Figure 3.15 shows the data transmission waveform with half-duplex mechanisms.)

CLK Clock line of VI Bus.

REQ Request signal from initiating FPGA to objective FPGA.

FRAME Switch signal for A/D line; active-high: address/command, active-low: Data.

SEL Select signal from master, it is asserted when master get VI Bus authorization.

ACK0 Acknowledge signal from target FPGA when transmission route is determined.

ACK1 Acknowledge signal when FIB of objective FPGA can be utilized.

MRDY Data transmission ready (assert) signal from master.

SRDY Data transmission ready (assert) signal from slave.

A/D Address/data line.

Firstly, the initiating FPGA (master) asserts REQ, FRAME signal and send address/command signals (Read/write, half-duplex/ full-duplex, etc.) . When communication route from initiating FPGA to objective FPGA is valid, the slave on objective FPGA asserts ACK0 signals. Then, when FIB port inner objective FPGA can be utilized, the slave on objective asserts ACK1 signals. When master side receives the two acknowledge signals(ACK0 and ACK1), the master asserts SEL and MRDY signal, and transmit data when receiving SRDY from slave.

3.3.2 Implementation of VI Bus

We implemented a general VI Bus module as general peripheral circuit of hwNet which can realizes point-to-point transmission and distance data transmission on our system, to demonstrate the communication feasibility on our system. When PVS as a repeater receives and identifies data from adjacent PVS. If it is not target FPGA, the PVS can select a suitable channel and transmit data to adjacent PVS via the according channel. Because of a PVS can connects multiple PVSs through multiple channels. We realized a simplified routing function on hardware level on the module for 2D network connections. The block diagram is shown in Figure 3.16.

- VIBus module(point-to-point data communication)
- FIB_VIBus_Converter
- VIBus_FIB_Converter
- VIBusManager
- Route_Determiner

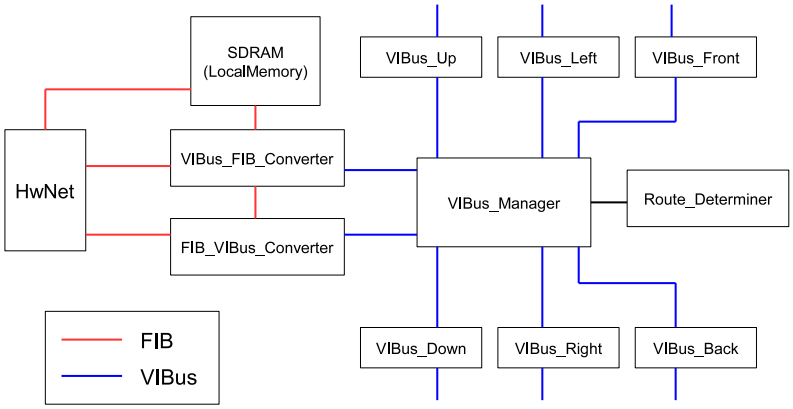

Fig. 3.16 Composition of VI Bus transmission module.

VIBus

The VIBus units can implement point-to-point data communication among adjacent PVSs. The units directly connect the according GPIF I/Os, and transmit data to adjacent PVS via the connection. The VIBus modules can achieve two transmission mechanisms (half-duplex and full-duplex transmission). Each VIBus module has two asynchronous FIFOs to send/receive data to/from adjacent FPGAs.

The Figure 3.17 shows the data flow of full-duplex transmission between PVS(1) data and PVS(3) through intermediate PVS(2) via VI Bus.

In the Figure 3.17, PVS(1) writes data to PVS(3). The data transfer is implemented on multiple clock signal source. PVS(1) firstly writes data to PVS(2) with inner clock signal from PVS(1), then PVS(2) transfers data to PVS(3) with inner clock signal of PVS(2) .

FIB_VIBus_Converter

The module can converts FIB protocol to VI Bus protocol; it realizes slave function of FIB and master function of VI Bus.

The number of FIB slave is variable as the required FIB port form hwNet. Inner FPGA, the hwNet can utilizes up to six FIB ports for data communications of VI Bus. Because of these FIBs are independent of each other, it is possible that a hwNet can uti-

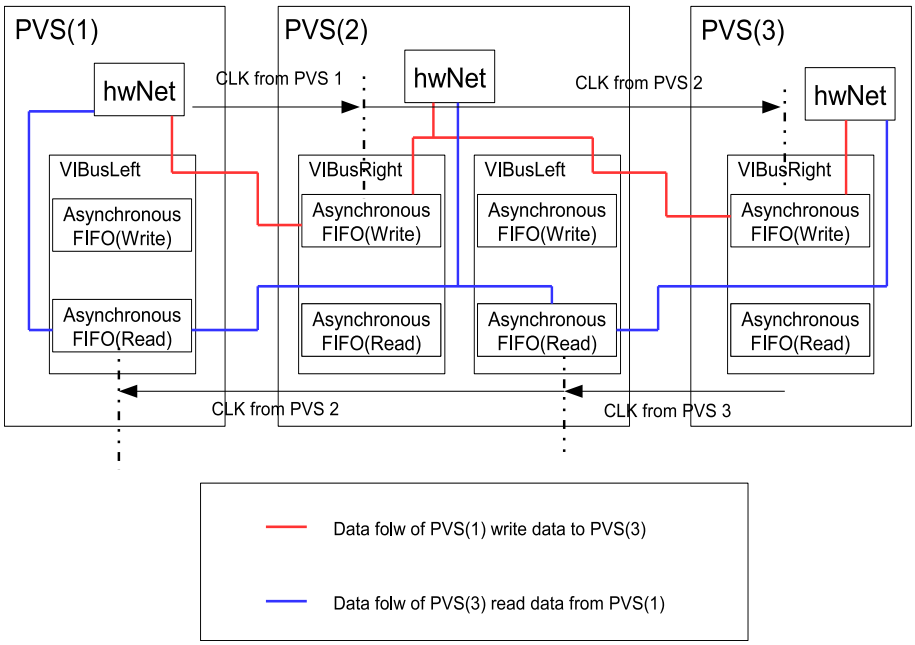

Fig. 3.17 Distance transmission among PVSs via VI Bus.

lizes 6 FIB at the same time to access 6 adjacent PVSs. Through FIB signals from hwNet, FIB_VIBus_Converter modules determine the transmission mechanism (half-duplex/full-duplex) and write/read operation; and send data to adjacent FPGAs with VIBus protocol.

VIBus_FIB_Converter

The module realizes opposite functions of FIB_VIBus_Converter. It implements slaves of VI Bus and masters of FIB, to convert VI Bus protocol to FIB protocol from external FPGA to hwNet.

The number of FIB masters is variable as the FIB ports of hwNet changes.

In the same way, a hwNet can utilizes up to 6 VIBus_FIB_Converter to realize receiving data through 6 VIBus modules with VIBus protocol.

VIBusManager

The module can implements bridge function of VI Bus, and monitors the status of VIBus modules. It has multiple arbiters and controllers for multiple VIBus modules. The module receives the bridge commands from Route_determiner module to arbiter, and confirms whether the transmission route is occupied. When path is valid, the module as a bridge circuit, links to VIBus_FIB_Converter and VIBus module on according adjacent to achieve the data transmit. Moreover, the VIBusManager also realizes to control the IO interface of VIBus alon initiating terminal. .

Route_Determiner

The module is used to propose data transmission route to objective FPGA, explore path and determine path. The Route_Determiner on initiating FPGA proposes multiple communication routes from initiating terminal to objective terminal.

The module sends REQ signals along multiple directions, and adopts the route which receives fastest ACK signal.

In our experiments, we realize distance data communication at 66 Mhz on a 2D FPGA array, which composed of 3×4 PVSs. (shown in Figure 3.18) The multiple PVSs can communicate data at the same time by using Route_Determiner. Because of the routes become surprisingly complex on a 3D network topology. We determine routes among PVSs on software, then send I/O command to control each PVS to determine routes through hw/sw complex units.

3.4 Discussion of VC Bus and VI Bus

3.4.1 Parallel circuit configuration

Many studies have demonstrated that circuit configuration on multi-FPGAs is often implemented with custom cables such as USB cables. As the scale and dimensions of FPGAs have increased, the configuration time and used cables also multiplied. There-

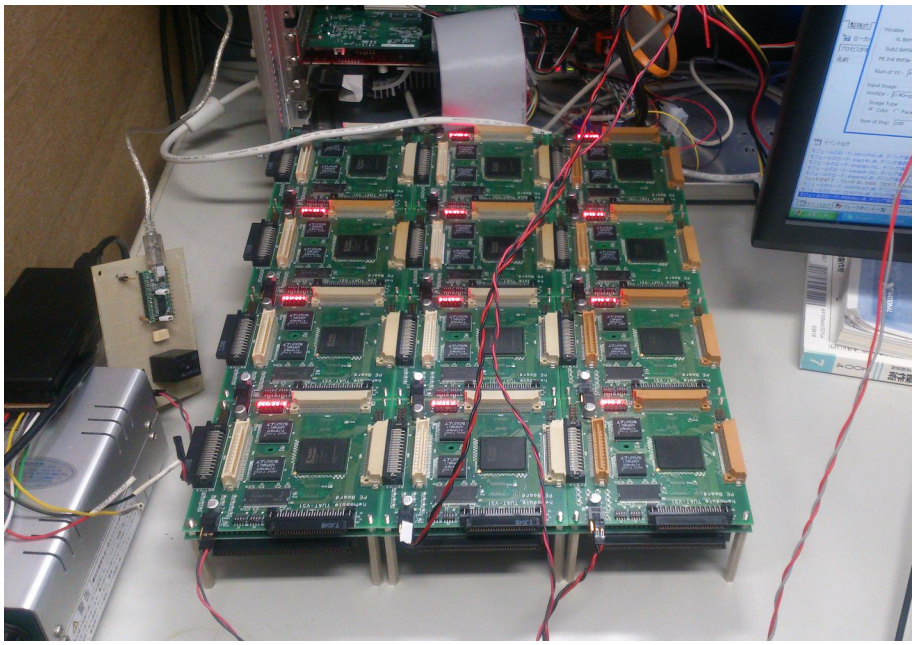

Fig. 3.18 Distance transmission on 2D VI Bus Network

fore, an effective approach is necessary to configure a flexible and scalable FPGA array quickly. In our system, we used an FPGA to configure a circuit on the next FPGA via VC Bus network; thus, extra custom cables are unnecessary.

Table 3.6 shows circuit configuration time for multiple PVSs in a row via the VC Bus network. The host PC required approximately 0.2s to configure the same application circuits to a row of multiple PVSs.

The approach provides parallel configuration of multidimensional FPGAs. For an application, the configuration mechanism enables the system to implement simultaneous circuit configuration on 32 PVSs (4 rows $\times$ 8 PVSs) through a BVS. Moreover, the configuration mechanism also allows the user to select any FPGA to configure different application circuits; the FPGA array can perform multiple applications simultaneously.

Table 3.6 Circuit configuration execution time of multi-FPGA

FPGA Number	1	2	3	4
Sub Board	0.051s	0.101s	0.152s	0.203s
PE Board	0.201s	0.201s	0.201s	0.202s

3.4.2 Scalable multi-FPGAs communication

In Vocalise, the data communication and synchronization among processing nodes (PVSs) is implemented via VI Bus network. The VI Bus network is an application network, i.e., it is a dedicated network; (preferably with high throughput and low latency). Since of each process node is implemented by FPGAs, which is a programmable logic device. The VI Bus is freely available for custom user designs. Users can arrange processes to each node (PVS) with different network topology according to the requirements of different applications.

For instance, Figure 3.19 shows three types of network topology that can be implemented by our Vocalise via the VI Bus network.

Network A is a common topology for many numerical calculations with multiple parallel computing nodes. The network comprises many sub-networks, which form a basic point-to-point network topology; each connection between FPGAs becomes a sub-bus. Each connector I/O bandwidth can be fully utilized when each FPGA sends data to the nearest adjacent FPGAs.

Network B is a typical ring bus topology; it is set up in a circular fashion where in data travels around the ring in one direction. Each processing node (FPGA) on the ring acts as a repeater to keep the signal strong as it travels. Each FPGA incorporates a receiver for the incoming signal and a transmitter is used to send the data to the next device in the ring. This network is dependent on the ability of the signal to travel around the ring. When an FPGA sends data, it must travel through each FPGA on the ring until it reaches its destination; thus, each node is a critical link. The ring bus network is commonly seen in multi-FPGA systems and general multi-core processor designs such as the BEE3, IBM

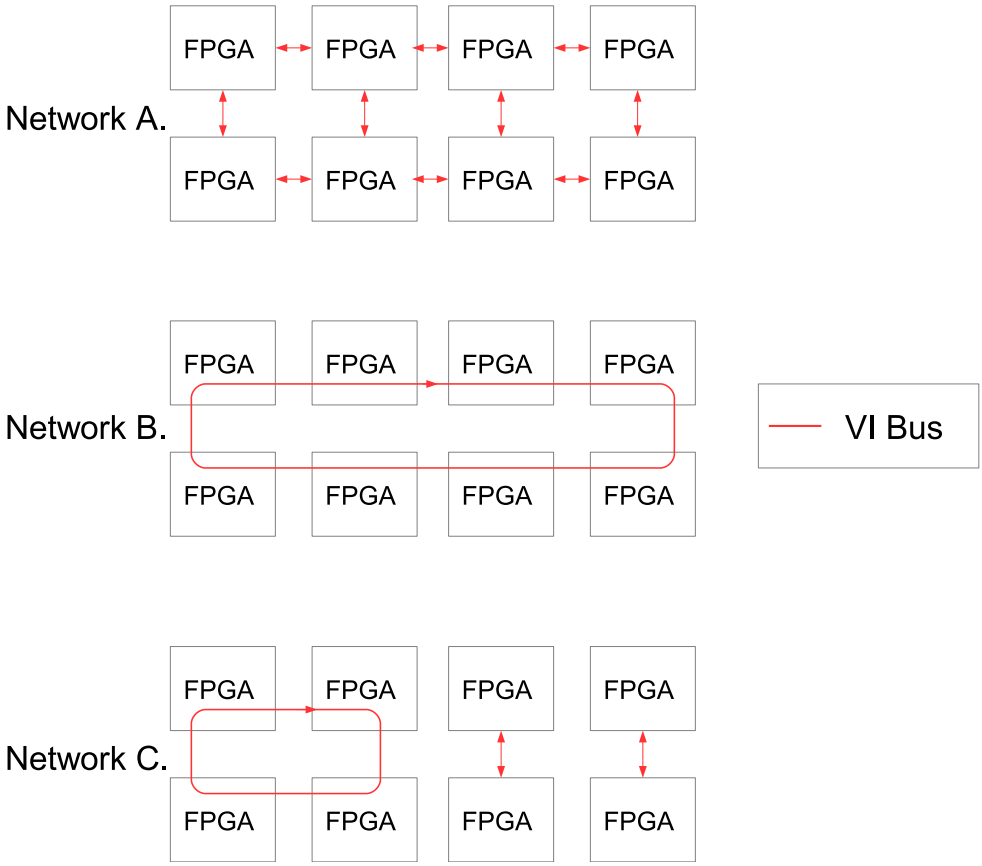

Fig. 3.19 Different types of VI Bus network topology

cell, and the Intel Haswell processor.

Network C is a hybrid network topology. The whole VI Bus network is distributed to three sub-networks. Four FPGAs on the left side are arranged in a ring bus network. The other four FPGAs are arranged in two point-to-point networks. A sub-network is a custom computing network for executing individual application simultaneously. This allows that our system do not only to achieve single application/multiple data stream computing, but also simultaneously execute multiple applications with a distributed FPGA array. We have reported that an FPGA array can concurrently operate multiple individual brain process applications[28]. These brain process circuits such as voice recognition, voice synthesis, and image recognition were designed as hw/sw complex systems using the

Vocalise method.

3.4.3 Data communication efficiency on multidimensional interconnection

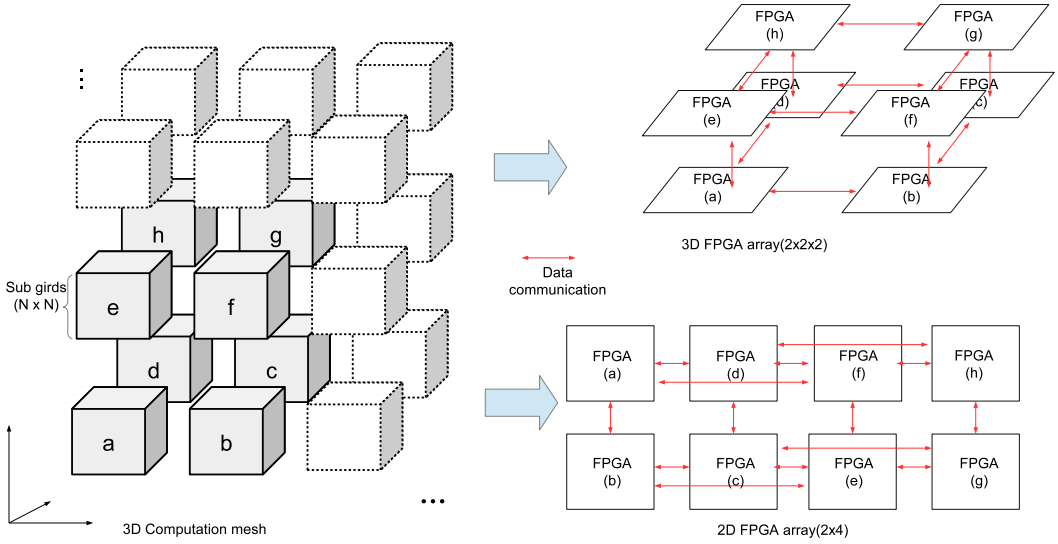

Fig. 3.20 Data communication in FPGA array with different dimensional interconnection

We discuss data communication overhead between FPGAs when operating a 3D domain problem with different dimensional FPGA arrays. In common finite difference numerical calculations, many exchanges of boundary data among FPGAs occur. For instance, Figure 3.20, shows a 3D computation mesh is decomposed to many sub-grids, each sub-grids comprises N $\times$ N $\times$ N grids. Thus, N $\times$ N boundary data must be exchanged with each nearest sub-grids between each iteration. We use a 4 $\times$ 2 2D FPGA array and a 2 $\times$ 2 $\times$ 2 3D FPGA array to distribute operations to 2 $\times$ 2 $\times$ 2 sub-grids. Each FPGA executes the calculation of relative sub-grids as a computing node, such as FPGA (a) operates sub-grids (a), and so on.

For a 3D FPGA array, each FPGA transfers boundary data only to the connected adjacent FPGAs via 3D interconnection connector I/O. Thus, it is relatively easy to imple-

ment high-speed and low latency point-to-point data exchange between two connected FPGAs.

In contrast, for a 2D FPGA array, because of 2D mesh interconnection between FPGAs, there are no direct interconnections between FPGAs when boundary data is exchanged between the top and bottom of sub-grids, for example, FPGA (a) must exchange data with FPGA (e). The FPGA must send data to connected FPGAs until the data reaches the destination FPGAs. To implement this approach, the system must build many paths and address identification modules for each FPGA. This will occupy more logic resources, which are very limited. To achieve highly efficient data exchanges, the users must take more work time to reduce transmission latency; which is often difficult when transferring data via multiple FPGAs. It may be necessary to provide wider link of data communication to achieve higher bandwidth.

Overall, the comparison shows that an FPGA array with 3D direct interconnection can provide better scalability, improve communication efficiency, and reduce design difficulty of data communication for 3D computing problems.

Chapter 4

System Evaluation

We implement a Vocalise system with multidimensional FPGA array in hw/sw complex, the system can implements various applications such as numerical simulations, brain processes, web applications and so on[23][28][29][50]. When a distributed system operates numerical simulation especially many PDE problems, the computing nodes normally want to keep frequent communication with the adjacent nodes. Therefore, we mainly evaluate the system through solving 3D PDE problems in the study.

4.1 Applications of numerical simulation

Numerical computing is an study of approximation techniques for numerically solving mathematical problems. It also is an interconnected combination of computer science and mathematics by using to develop and analyze algorithms for solving important problems in science, engineering, business, and medicine —for example, designing an aircraft, simulating atmospheric circulation, or detecting tumors in medical images.

By using most partial differential equation (PDEs), through obtain numerical solutions, physical phenomena such as can be simulated by computer. Since numerical simulation requires massive computational capabilities, such as big numerical simulation programs require supercomputers and a large amount of computer resources. Therefore it also is an important area of HPC study.

The numerical computation, especially partial differential equation (PEDs) solutions is a main application area of our Vocalise system. The proposed system realizes different applications with highly efficient and specific application circuits implemented on large scale FPGAs. We call these application circuits as hwNets. The most partial differential equation problem are solved with Cartesian-gird. The algorithm is usually suited to be solving our system. In the sections, we describe experimental implementations of 3D Poisson equation and CIP method.

4.1.1 Advection equation with CIP method

Advection equation

In physics, engineering, and earth sciences, advection is a transport mechanism of a substance or conserved property by a fluid due to the fluid's bulk motion. For example the transport of pollutants or silt in a river by bulk water flow downstream.

In general, any substance or conserved, extensive quantity can be advected by a fluid that can hold or contain the quantity or substance.

Advection is sometimes confused with the more encompassing process of convection which is the combination of advective transport and diffusive transport.

The advection equation is the partial differential equation that governs the motion of a conserved scalar field as it is advected by a known velocity vector field.

In 2D Cartesian coordinates the advection operator is shown in Equation 4.1.

$$\frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} + v\frac{\partial f}{\partial y} = 0 \tag{4.1}$$

Here, t is time, x, y is 2D Cartesian coordinates, f is wavefield of advection equation, u is velocity in x axis, and y is y axis.

Algorithm of CIP method

The Cubic Interpolated Profile (CIP) method is a complicated method for solving the advection equation, proposed as a stable and less dispersive method in computational

fluid dynamics (CFD) since the middle of the 1980s [30],[31]. It has been applied to simulations of various physical problems and proved to be well performing [35]-[40]. This method is based on a fact, that it is not only the wavefield but also its spatial derivatives propagate along the same characteristic curve derived from a hyperbolic differential equation [42]. The CIP method in combination with the method of characteristics, it was developed to simulate the Maxwell equation accurately compared with FDTD method [41].

In the thesis, we implemented the CIP method to simulate wave propagation through solving advection equation. The phenomenon of the wave propagation in one dimensional space can be expressed with the following first-order differential equation.

$$\frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} = 0 \tag{4.2}$$

This first-order advection equation shows that a wave packet on the wavefield f propagates along a curve $dx/dt = u$, which is a characteristic curve, in the phase space. Eq. 4.2 is a characteristic equation for solving the forward propagation of the wavefield. Although this equation is simple, it is difficult to evaluate numerically with high stability and less numerical dispersion. The CIP method can solves these problems through solving not only Eq.4.2 but also a differential equation for a spatial derivative of the wavefield f . When the propagation velocity u is constant, the Eq. 4.3 can be obtained through Eq.advecEq2.

$$\frac{\partial g}{\partial t} + u\frac{\partial g}{\partial x} = 0, g = \frac{\partial f}{\partial x} \tag{4.3}$$

Here, g is a spatial derivative of f. These two equations, Eq.4.2 and Eq.4.3, become the governing equations for the propagation of the wavefield f and its spatial derivatives g. This property can be utilized by the CIP method to solve a hyperbolic differential equation. The Figure 4.1 shows conceptual diagrams of the CIP method[42], [37].

In Figure 4.1(a), the solid line corresponds to an initial wave packet and dashed line

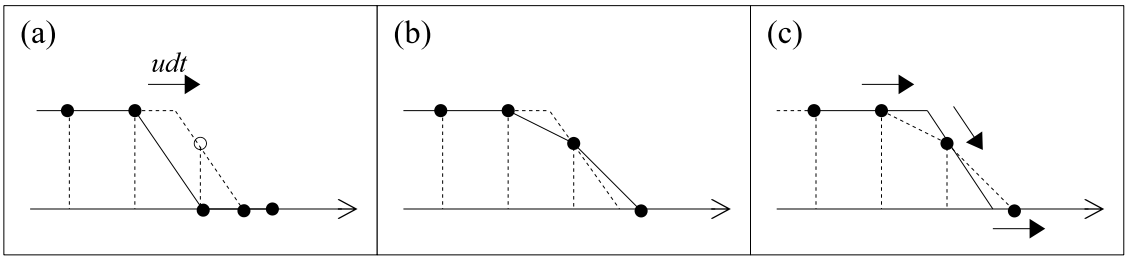

Fig. 4.1 Conceptual diagrams of the CIP method.

becomes an exact solution at one time-step ahead. Solving the wave equation numerically through the finite difference approximation,the white circle can be obtained after one time progressed (shown in Figure 4.1(a)). If the values of the wavefield between the grids are interpolated linearly through values at each grid, the numerical diffusion occurs shown in Figure 4.1(b). However, if the information of the spatial derivatives was used at each grid, the numerical dispersion problem and the original shape of the wave packet can be overcome and kept through the all simulation steps. This is the core idea of the CIP method, and the values at grids are interpolated using a cubic polynomial (shown in Figure 4.1(c)).

Solution of 1D CIP method

If values of wavefiled f and its derivative g are known at grids i and $i-1$ The profile between two grids can be interpolated using a cubic polynomial.

$$F_i = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \tag{4.4}$$

There are four variables $f_i^n, g_i^n, f_{i-1}^n, g_{i-1}^n$ between two adjacent grids; and these variables are determined by four coefficients a_i, b_i, c_i, d_i. By using g_i^n which is the differential operator of function $F_i(x)$. We can obtain the following equations.

$$F_i(x_i) = f_i^n \tag{4.5}$$

$$\begin{aligned} \frac{dF_i(x_i)}{dx} &= g_i^n \\ F_i(x_{i-1}) = -a_i\Delta x^3 + b_i\Delta x^2 - g_i^n\Delta x + f_i^n &= f_{i-1}^n \\ \frac{dF_i(x_{i-1})}{dx} = 3a_i\Delta x^2 - 2b_i\Delta x + g_i^n &= g_{i-1}^n \end{aligned}$$

When velocity u positive direction(forward propagation), value of i girds move to the profile$[i-1, i]$ Therefore, the coefficients of equation can be computed with following Equations. Here, $iup = i-1, D = -\Delta x$.

$$\begin{aligned} a_i &= \frac{g_i^n + g_{iup}^n}{D^2} + \frac{2(f_i^n - f_{iup}^n)}{D^3} \\ b_i &= \frac{3(f_{iup}^n - f_i^n)}{D^2} - \frac{2g_i^n + g_{iup}^n}{D} \\ c_i &= \frac{dF_i(x_i)}{dx} = g_i^n \\ di &= f_i^n \end{aligned} \tag{4.6}$$

When velocity u is negative direction(backward propagation), value of i girds move to the profile$[i, i+1]$. The coefficients can be operated with following Equations.

$$\begin{aligned} F_i(x_i) &= f_i^n \\ \frac{dF_i(x_i)}{dx} &= g_i^n \\ F_i(x_{i+1}) = a_i\Delta x^3 + b_i\Delta x^2 + g_i^n\Delta x + f_i^n &= f_{i+1}^n \\ \frac{dF_i(x_{i+1})}{dx} = 3a_i\Delta x^2 + 2b_i\Delta x + g_i^n &= g_{i+1}^n \end{aligned} \tag{4.7}$$

Here $iup = i+1, D = \Delta x$. When $u \geq 0$, $iup = i-1, D = -\Delta x$, $u \leq 0$, and when $u \leq 0$, $iup = i-1, D = -\Delta x$, $u \leq 0$. Therefore, We can operated the wavefield at next time step$(n+1)$ with following Equation, where the grids move along the profile at velocity $u\Delta t$. Here, $X = -u\Delta t$.

$$f_i^{n+1} = a_i X^3 + b_i X^2 + g_i^n X + f_i^n \tag{4.8}$$
$$g_i^{n+1} = 3a_i X^2 + 2b_i X + g_i^n$$

We summarize the CIP method as follows; the wavefield is interpolated by a cubic polynomial and it is shifted to the wave propagation direction by $u_i \Delta t$ at each time step. By computing the Eq 4.7 repeatedly, we can solve the 1D advection equation with CIP method.

Solution of 2D/3D CIP method

The 2D advection equation is expressed with following Equation.

$$\frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} + v\frac{\partial f}{\partial y} = 0 \tag{4.9}$$

Where, t is time, x, y is 2D Cartesian coordinates, f is wavefield of advection equation, and $\Delta x, \Delta y$ is constant. u is velocity on the x direction, v is velocity on y direction.

By using the method which described on previous section, the profile the wavefield between these two points can be interpolated using a cubic polynomial as follow.

$$F(X,Y) = [(a_1 X + c_1 Y + e_1)X + g_1 Y + f_x(i,j)]X + [(b_1 Y + d_1 X + f_1)Y + f_y(i,j)]Y + f(i,j) \tag{4.10}$$

Here, $X = x - x_i$,$Y = y - y_i$ There are 10 coefficients $a_1, b_1, c_1, d_1, e_1, f_1, g_1, f, f_x, f_y$ need to operate.

In the same way, the 3D advection equation is following Equation.

$$\frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} + v\frac{\partial f}{\partial y} + w\frac{\partial f}{\partial z} = 0 \tag{4.11}$$

By using the same method of 1D , the profile between these two points can be inter-

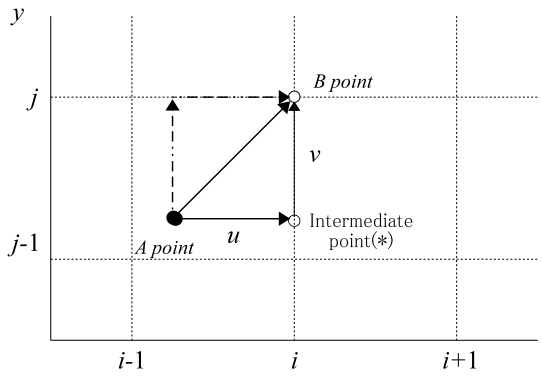

Fig. 4.2 The type M CIP method for sloving a 2D advection problem

polated by using a cubic polynomial as follow.

$$F_{i,j}(x,y) = \sum_{l=0}^{3}\sum_{m=0}^{3}\sum_{n=0}^{3} C_{l,m,n} X^l Y^m Z^n \tag{4.12}$$

There are 3^3 coefficients need to operate. We can see the numbers of coefficients increases as dimension increases.

When advection equation is n dimensions, it has 3^n number of coefficients, and needs huge amounts of calculation. For reducing the huge calculation burden, we adopt type M CIP method to operate multidimensional advection equation [43].

Type M CIP is a amplitude compensation method which is based on the diversion relation of 1D-CIP method, is applied to the multidimensional CIP calculation method with directional splitting technique [44]. The diagrammatic drawing of type M CIP solution is shown in Figure 4.2.

When wavefield moves from Point A to Point B With type M CIP method, the advection can be realized in two steps:

Step 1: Moving from origin point A to intermediate point $*$ at u speed in x direction.

Step 2: Moving from intermediate point $*$ to terminal point B at v speed in y direction.

By using CIP method which is highly precise with interpolating. Even if it passes along which course, the symmetry of a solution is held, and the same result is obtained.

Therefore, Eq 4.10 can be split into x, y direction, and solved with 1D CIP computation on according direction. We can gain the following Equations.

$$\frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} = 0, \qquad f^n \rightarrow f^* \tag{4.13}$$

$$\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial y} = 0, \qquad f^* \rightarrow f^{n+1} \tag{4.14}$$

Here, we set n time step is f^n, next time is f^{n+1}; and f^* is intermediate value. Therefore, we can solve a 2D advection equation as following steps.

Step1: By using 1D CIP method, we get the solution f^*, $\partial f^*/\partial x$ through f^n and $\partial f^n/\partial x$ in the x direction.

Step2: By using 1D CIP method, we get the solution f^{n+1},$\partial f^{n+1}/\partial y$ through f^* と $\partial f^*/\partial y$ in the y direction.

Because of $\partial f^*/\partial y$, which is necessary in step 2 , has not been solved in step 1. Meanwhile, at next time (n+1), the step 1 computation also need $\partial f^n + 1/\partial x$, which has not been solved in step at n time(step 2) .

For the solution in next step, we solve the $\partial f^*/\partial y$ and $\partial f^{n+1}/\partial x$ with upwind method which is a numerical discretization method for solving hyperbolic PDEs. According to such a scheme, the spatial differences are skewed in the " upwind" direction, i.e., the direction from which the advecting flow originates. The origin of the method can be traced back to the work of Courant, Isaacson et al who proposed the CIR method [45].

The $\partial f^*/\partial y$ can be solved the follow equation.

$$\begin{aligned} \partial_y f^*_{i,j} &= \partial_y f^n_{i,j} - \frac{\partial_y f^n_{i,j} - \partial_y f^n_{i-1,j}}{\Delta x}, \qquad u > 0 \\ &= \partial_y f^n_{i,j} + \frac{\partial_y f^n_{i,j} - \partial_y f^n_{i+1,j}}{\Delta x}, \qquad u < 0 \end{aligned} \tag{4.15}$$

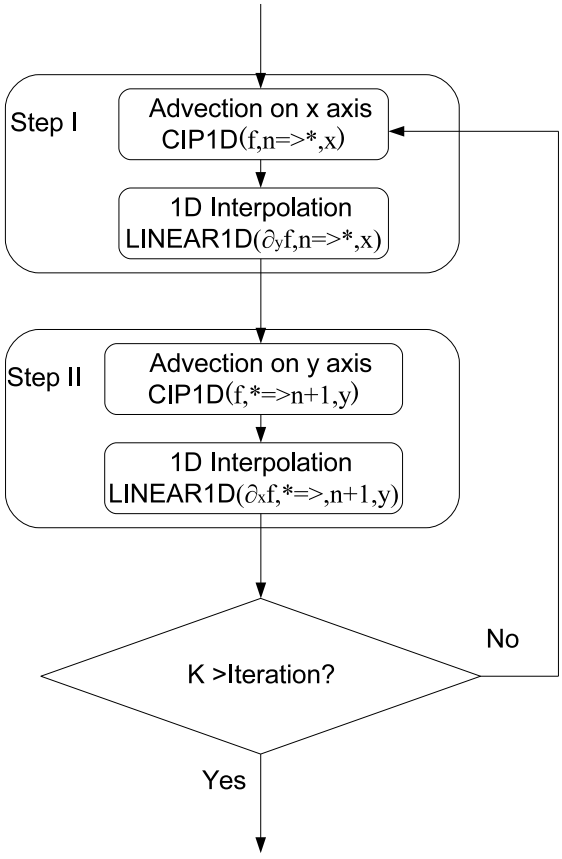

Fig. 4.3 The process flow chart of type-M 2D CIP method

The solution of $\partial f^{n+1}/\partial x$ also can be completed in the same way. The operation flow of 2D CIP method is shown in Figure 4.3.

Process element(PE) for 1D CIP method

To implement the Process element circuit , we set $\Delta x = \Delta y = 1$ as constant value, and $\Delta t = 1$. The Equation with 1D CIP method becomes following Equations.

$$a = g_i^n - g_{iup}^n - 2(f_i^n - f_{iup}^n) \tag{4.16}$$

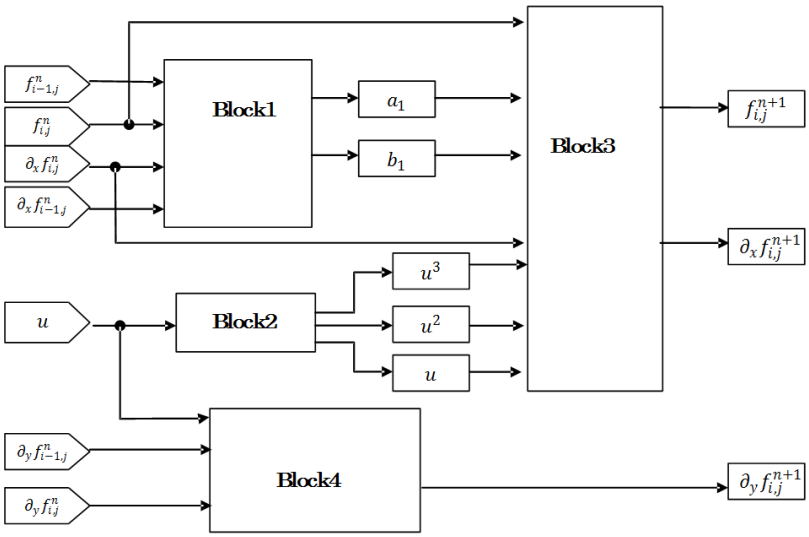

Fig. 4.4 Block diagram of PE for CIP method

$$b = 3(f^n_{iup} - f^n_i) + 2g^n_i + g^n_{iup} \tag{4.17}$$

$$f^{n+1}_i = -au^3 + bu^2 - g^n_i u + f^n_i \tag{4.18}$$

$$g^{n+1}_i = 3au^2 - 2bu + g^n_i \tag{4.19}$$

We utilize adders, multipliers and bit-shift operators to realize the operation of Eq (4.16)- Eq (4.19). All ALUs are 32-bit floating-point arithmetic units on IEEE754 standard. The PE circuit is shown in Figure 4.4.

A PE mainly consists of 4 blocks. Block1 is used to achieve operation of Eq (4.16) and Eq (4.17) to solve a and b. The circuit is shown in Figure 4.4.

The Block1 consists of six adders/subtractors, one multiplier and two bit-shift operator. All operators are multi-pipeline architecture. There, the adder is 13 stage pipeline,

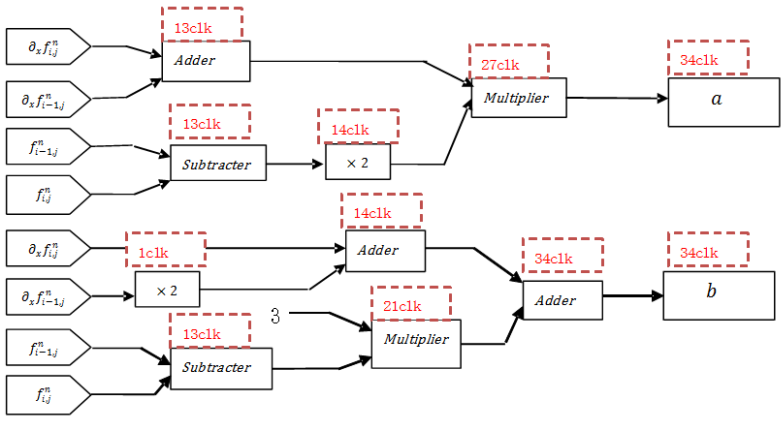

Fig. 4.5 The circuit of Block1

multiplier is 8 stage pipeline. Since of sub-unit for solving a has 27 stage pipeline length, and the sub unit for b has 34 stage pipeline length To implement pipelined architecture of Block1, we use 7 stage shift registers to delay output of a.

The Block2 is a sub process unit for solving u^2 and u^3 with u(shown in 4.6). The unit utilizes two multipliers to constitute a 16 stage pipelined unit.

The Block3 solves Eq (4.18)and Eq (4.19). It consist of five adders/subtractors, six multipliers and one shift register, and the pipeline length is 35 stage.

We implement parallel computing with Block 1 (which solves a and b) and Block2 (which solves u^2 and u^3), the pipeline lengths are 34 stages. The blocks 1-3 compose a pipelined process unit, which has 69 stage pipeline, and consist of 11 adder/subtractors, 9 multipliers, 3 bit shift operators.

We also set $\Delta x = \Delta y = 1$ and $\Delta t = 1$. Eq(4.15) becomes the follow Equation.

$$g_i^* = g_i^n + (g_i^n - g_{iup}^n)u \tag{4.20}$$

Block4 is a processing unit for solving interpolated equation with upwind method (Eq

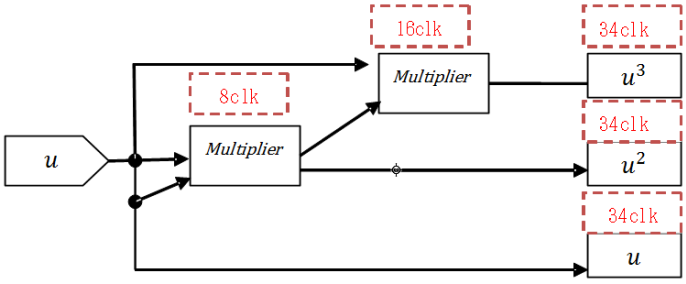

Fig. 4.6 The circuit of Block2

(4.8)); It has 34 stage pipeline, and consist of 2 adder/subtractors, 1 multipliers.

Due to type-M CIP method, multidimensional advection can be split into n (n is dimension)steps computation which realizes computation of 1D CIP method in each step. The 2D or 3D wave propagation simulation can be realized through changing input data to the PE circuit for solving 1D CIP method. Meanwhile, for solving n dimension advection equation, n-1 numbers of Block 4 need to be implemented on PE. For 2D CIP method, the PE circuit scale is shown in Table 4.1.

Architecture of hwNet

We realized a hwNet to solve 2D/3D advection equation in type-M CIP method. The hwNet architecture is shown in Figure 4.9.

- Cache
- Processing elements
- PE Controller

Table 4.1 Circuit scale of 1 PE for 2D CIP

Circuit scale	9,940 [Slices], (35[%] of XC3S4000)
Maximum frequency	165.3 [MHz]

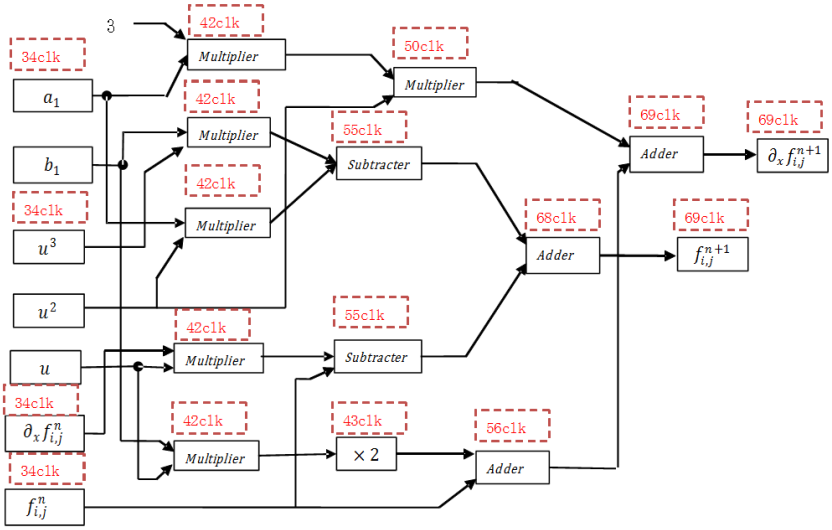

Fig. 4.7 The circuit of Block3

- Buffer Controller
- BRAM & LM controller

The data store unit composes of local memory and cache. The initial data and result data are stored on local memory. We implemented a high-speed cache with registers and BRAMs in FPGA to satisfy data inputs/outputs of PE at every clock cycle. One PE of 2D CIP method running at 66 Mhz, needs 1.596 GB/s input bandwidth and 779 MB/S output bandwidth .

Table 4.2 The circuit scale of hwNets of 2D/3D CIP method

Logic Utilization	2D CIP		3D CIP		Available
	Used	Utilization	Used	Utilization	
Flip-Fliops	14,126	25%	16,813	30%	55,296
LUTs	13,664	24%	16,134	29%	55,296
Occupied Slices	10,877	39 %	12,395	44%	27,648
RAM16	3	3%	4	4%	96

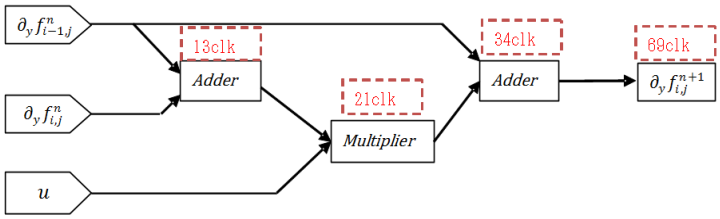

Fig. 4.8 The circuit of Block4

The controller module of hwNet composes of PE controller, BRAM & LM controller, Cache controller. BRAM & LM controller module is used to realize read/write access to initial value data/result data among host PC and local memories on PVSs. The Cache controller is used to read the initial value data in local memory to cache before cal-

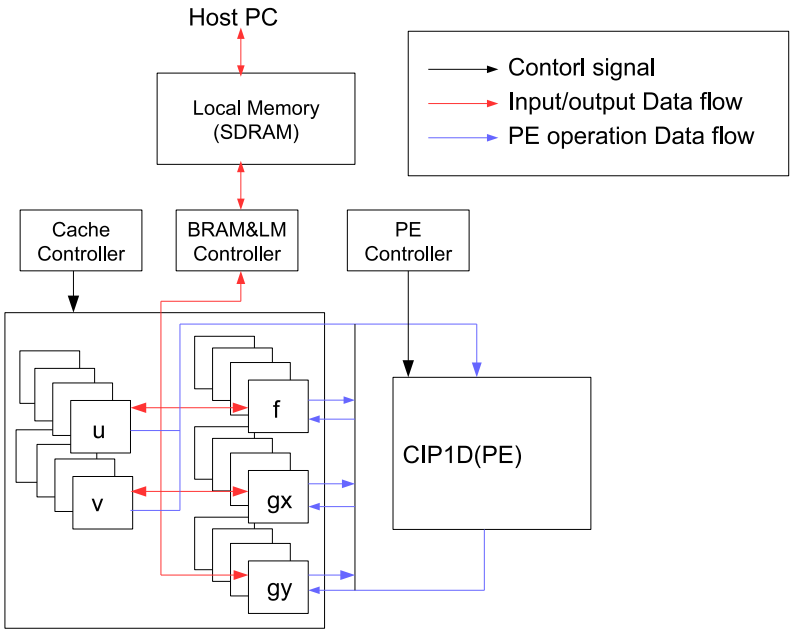

Fig. 4.9 Block diagram of hwNet for sloving 2D CIP method

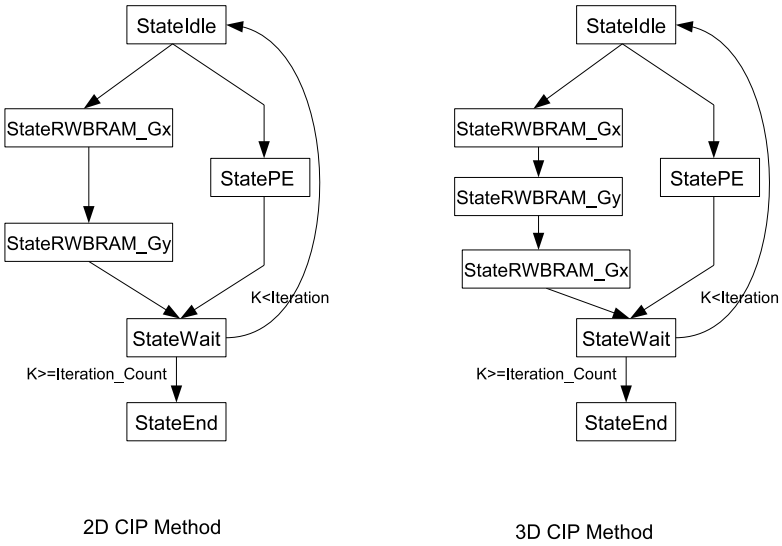

Fig. 4.10 The processing flow chart of main controller for 2D/3D CIP method

culating, and write the result data in cache to local memory after the computation is completed.

The PE controller is a main controller module of hwNet for managing the status of PE and data throughputs between PE and cache. The Figure 4.10 shows the processing flow chart of PE controller on 2D/3D CIP methods. The circuit scales of hwNets on hwModule V2 are shown in Table 4.2.

4.1.2 Poisson equation with Jacobi method

A Poisson equation is an elliptic PDE that has broad utility in electrostatics, mechanical engineering and theoretical physics [46], [47]. One of the principle cornerstones of electrostatics is the formulation and resolution of problems described by Poisson equation. Eq.(4.21) is Poisson equation apply to electrostatics.

$$\nabla^2 \phi = -\rho/\epsilon_0 \tag{4.21}$$

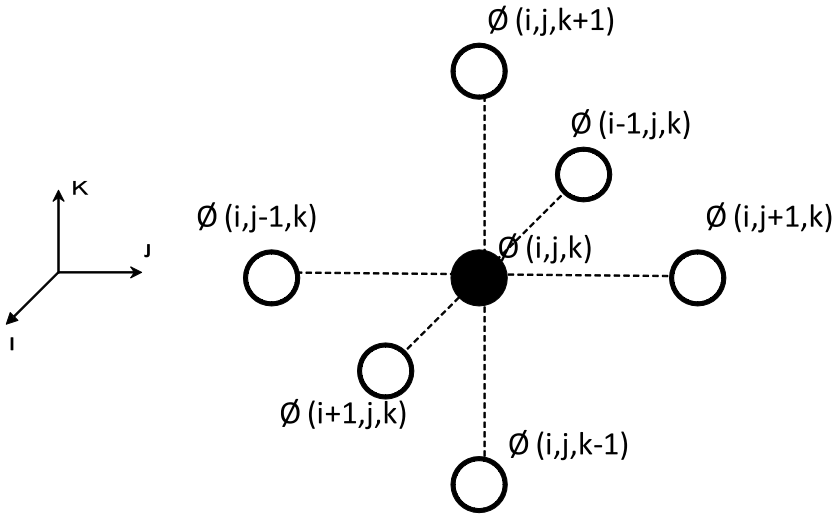

Fig. 4.11 3D collocated grids and computational image.

Here, ∇^2 is a Laplace operator, ρ is charge density, ϕ is electric potential, and ϵ_0 is the vacuum permittivity.

Solution of Poisson equation

In a case of 3D space, central-difference methods with second order accuracy give the approximations for the 3D collocated grids in Figure 4.11.

The Poisson equation can be solved by using Jacobi method(or Jacobi iterative method) with approximate operations[54], [55]. A 3D Poisson equation can be expressed in following common form.

$$\phi^{new}_{i,j,k} = -h^2\rho + (\phi^{old}_{i-1,j,k} + \phi^{old}_{i+1,j,k} + \phi^{old}_{i,j-1,k} + \phi^{old}_{i,j+1,k} + \phi^{old}_{i,j,k+1} + \phi^{old}_{i,j,k-1})/6. \quad (4.22)$$

Here, $\phi_{i,j,k}$ is a certain value at grid point (i, j, k). We refer to the operation as neighboring accumulations. In Eq. (4.22) all grid points only require the accumulation com-

putations using the adjacent grid point data. All grid-point operations are independent at one computation time; consequently, this computation is suitable for parallel execution. We can use an array of parallel processing elements (PEs) to execute Eq. (4.22) to exploit these locality and parallelism properties. The PEs are the core components of the implemented application circuits.

By multiplying 6 and then adding $2\phi^{new}_{i,j,k}$ to both sides of Eq.(4.22), we obtain the following equation.

$$\begin{aligned}\phi^{new}_{i,j,k} = (&\phi^{old}_{i-1,j,k} + \phi^{old}_{i+1,j,k} + \phi^{old}_{i,j-1,k} \\ &+\phi^{old}_{i,j+1,k} + \phi^{old}_{i,j,k+1} + \phi^{old}_{i,j,k-1} + 2\phi^{new}_{i,j,k} - 6h^2\rho)/8. \end{aligned} \tag{4.23}$$

Next, we replace $2\phi^{new}_{i,j,k} = 2\phi^{old}_{i,j,k} + \delta_{i,j,k}$, to achieve the final form of the equation.

$$\begin{aligned}\phi^{new}_{i,j,k} = (&\phi^{old}_{i-1,j,k} + \phi^{old}_{i+1,j,k} + \phi^{old}_{i,j-1,k} + \phi^{old}_{i,j+1,k} \\ &+\phi^{old}_{i,j,k+1} + \phi^{old}_{i,j,k-1} + 2\phi^{old}_{i,j,k} + \delta_{i,j,k} - 6h^2\rho)/8. \end{aligned} \tag{4.24}$$

In our numerical experiments, the errors $\delta_{i,j,k}/8$ decreased rapidly as expected, and we obtain the experimental results that are less than 10^{-4}%. Thus, we can transform Equation (4.22) to Equation (4.24) easily to simplify an arithmetic circuit design.

Architecture of hwNet

We developed a hwNet to solve 3D Poisson problems. Figure 4.12 shows the architecture of application circuits (hwNet) for a Poisson equation on a single PVS. The whole circuit consists of four major components: PE unit, data storage unit, control unit, and data communication unit.

We implemented a PE for a 3D Poisson equation shown in Figure 4.13. The PE contains seven adders, one divider and one multiplier to operate one grid-point with Eq.(4.23) for single-precision floating-point numbers that comply with IEEE754, the

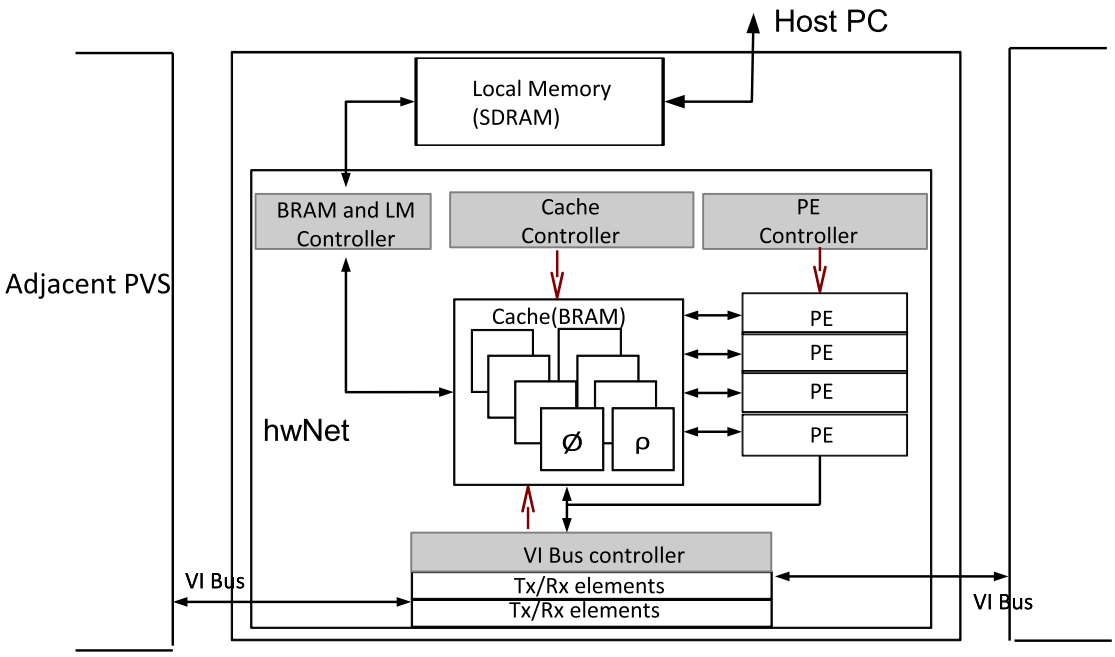

Fig. 4.12 Architecture of hwNet for 3D Poisson Equation.

standard for floating-point arithmetic. For additional simplification, the arithmetic circuits, divider, and multiplier are implemented by bit shifting. In addition, the PE is pipelined. The pipeline length of the PE is 41 stage, which enhances operational efficiency and achieves high utilization of arithmetic unit. A PE accounts for approximately 7% on a Xilinx Spartan 3 xc3s4000 FPGA equipped on a PVS.

Due to the circuit scale limitation of the Spartan-3 XC3S4000 FPGA (shown in Ta-

Table 4.3 Circuit scale for 3D Poisson equation

Logic Utilization	A Process element		3D Poisson(8PE)		Available
	Used	Utilization	Used	Utilization	
Flip-Fliops	4,383	7%	36,317	65%	55,296
LUTs	4,107	7%	36,027	59%	55,296
Occupied Slices	3,381	12%	27,646	99%	27,648
RAM16	10	11%	62	64%	96

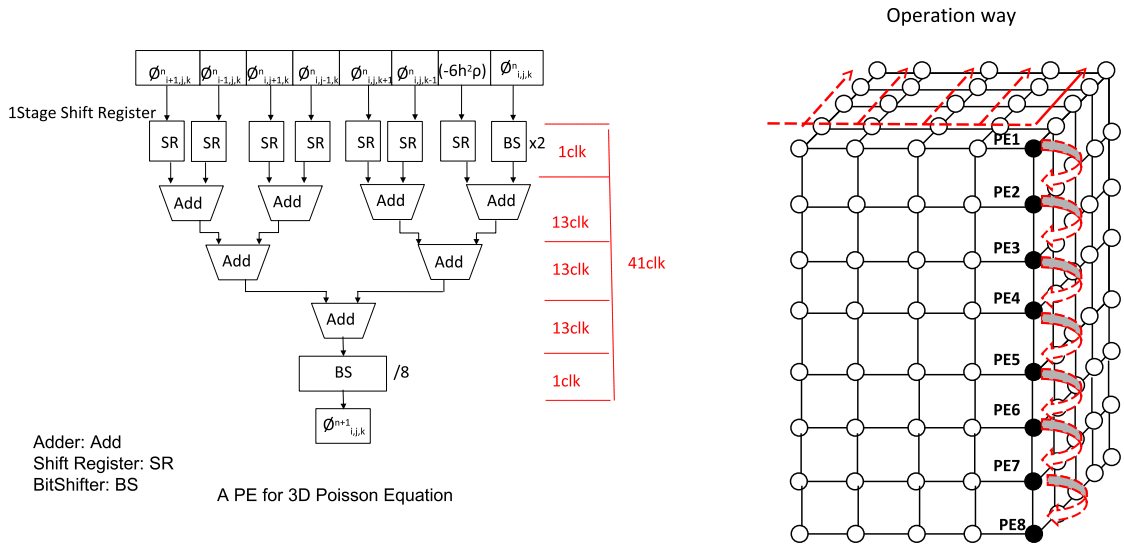

Fig. 4.13 A processing element (PE) of 3D Poisson equation (Left) and parallel operation organization of 8 PEs(Right).

ble 4.3), eight parallel PEs are implemented on a single PVS to solve the 3D Poisson Equation. The eight PEs are implemented by homogeneously partitioning the entire grids array as shown in Figure 4.13. Each PE operates sub-grids which are distributed on a plane. Therefore, the eight PEs are able to process eight sub-girds on eight parallel planes synchronously. We utilized 32 high-speed block RAM (BRAM) modules as cache to provide sufficient inner bandwidth for eight parallel PEs. We implemented a choice of 66 MHz for an operating frequency on 8 PEs.

The hwNet's data storage unit consists of SDRAM and BRAM. SDRAM is local memory used to store the initial data and the result data, and it allows hwNets access with a direct memory access (DMA) module via the FIB, which supports burst transmission. There are frequent data exchanges among parallel PEs and memory modules. Therefore we utilize a significant number of high-speed BRAM modules as cache to satisfy many PEs. Twelve 32-bit data inputs to 10 PEs at 1 clk provide sufficient inner bandwidth to enable parallel computing on pipelined PEs. The control circuit for the Poisson equation contains four circuit units: A BRAMandLMController is used to

control read-write operations for data stored in SDRAM via the FIB; a cache controller enables high-speed PE cache access. A PE controller enables multiple PEs to operate application synchronously. The VIBusController: controls data transmission and synchronization of the PVSs.

Data transmission and synchronization among multi-PVSs

In finite difference numerical calculation, data transmission between adjacent PVSs is necessary when PEs compute the boundary mesh girds. In order to achieve high-speed data transmission between nearest adjacent FPGAs, we design a transfer data circuit which is connected via GPIF I/O, 32-bit width, and the data transfer of each way is independent.

The FPGAs achieve data transmission by using the VI Bus Controller module that installed multiple VI Bus connectors which implements data transmit elements (Tx elements) and data receive elements (Rx elements). Each Tx/Rx element comprises multiple Tx/Rx FIFOs. For 3D interconnection, each PVS implements six VI Bus connectors for six-way data transmission when performing a 3D numerical calculation, as shown in Figure 4.14.

On the other hand, a synchronization problem with different clock sources arises from multiple-device implementation. In our design different clock sources are utilized for different FPGA cards. To address clock signal synchronization problem among FPGAs, we have utilized delay locked loop digital clock managers (DCM) in each FPGA, the clock skew and phase divergence among FPGAs can be effectively improved through DCMs to achieve synchronization of clock signals.

To implement synchronous operations of multiple PVSs for distributed computing, we utilize the VIBusController module to achieve a stall mechanism to enable multiple PVSs to attain synchronous operation of each iteration process. The module causes a local stall to inner PEs after each iteration, and it outputs a 1-bit End signal (oEnd) to all adjacent PVSs via GPIF I/Os. Until all input End signals (iEnd) from adjacent PVSs become high and off-chip data transmission is completed, inner PEs cannot execute the next iteration and send a start signal to announce the each boundary PVSs.

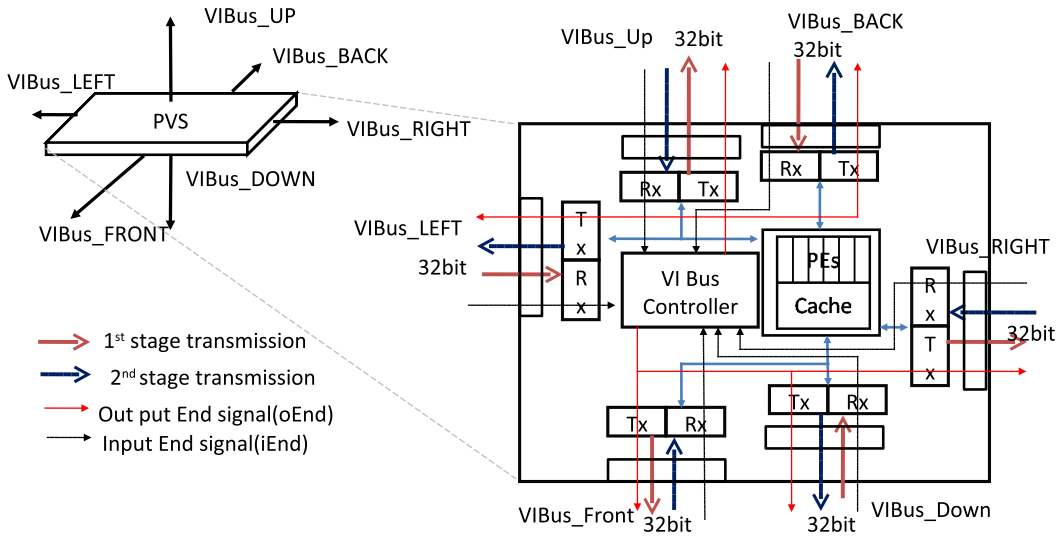

Fig. 4.14 Data transmission and synchronization between adjacent FPGAs

By considering the synchronous operation of a large scale multidimensional FPGA array, while the data transmission is limited by a 52-bit GPIF I/O port. Therefore, our system employs a two-stage data transfer mechanism. For example, when PEs operate a 3D Poisson equation, data transmission works via first stage operation where in the boundary grids on *VIBus_FRONT*, *VIBus_RIGHT*, and *VIBus_UP* sides are transferred to adjacent PVSs with corresponding Tx elements. Rx elements on *VIBus_BACK*, *VIBus_LEFT*, and *VIBus_DOWN* sides receive data from adjacent PVSs via VI Bus. When a transmission module completes the above operations, the data transmission state proceeds to stage, which is a backward operation stage. The data flow is opposite to the forward operation stage. If inner PEs have finished all calculations for all grids in an iteration process, the VIBusController module enables all PEs to be stall state until all boundary data has been transferred to adjacent PVSs.

Since the 6-way VIBusController modules require lots of logic sources which consist of Flip-Flops, LUTs, Slices, especial for BRAMs, there are only up-to 6 PEs can be realized on a Xilinx Spartan-3 XC3S4000 FPGA which implements on a PVS. The Table 4.4 shows the he circuit scale hwNet with 6-way data communication for solving 3D

Table 4.4 The circuit scale of hwNet with data communication for 3D Poisson equation

Logic Utilization	3D Poisson(6PE)		Available
	Used	Utilization	
Flip-Flops	30,237	54%	55,296
LUTs	31,636	57%	55,296
Occupied Slices	25,953	93%	27,648
RAM16	84	87%	96

Poisson equation.

HwNet design on Xilinx Virtex-7 XC7V2000t

Since Xilinx Spartan-3 XC3S4000 FPGA was released in 2008, was only equivalent to four million ASIC gates for a 90-nm process. In our design, only 6 PEs can be implemented on a Spartan-3 XC3S4000 FPGA for solving 3D Poisson Equation. The peak performance of a circuit can be calculated in follow equation.

$$F_{peak} = N \times F \times OF. \tag{4.25}$$

Where, F_{peak} is the peak performance of an FPGA, N is number of PEs, F is number of floating point arithmetic units in a PE, OF is PEs' operating frequency. When using a Spartan-3 XC3S4000 FPGA, one FPGA can achieve 6×9×66 MHz = 3.56 GFlops.

With the development of semiconductor technology, circuit scale of new generation high-end FPGA was increased readily. For instance, Xilinx Virtex-7 XC7V2000t FPGA, a 28-nm process high-end FPGA which was released in 2012. The Table 4.5 shows specifications of Xilinx Virtex-7 XC7V2000T and Spartan-3 XC3S4000 FPGA. Compared to used Spartan-3 XC3S4000 FPGA, the flip-flops (FFs) of Xilinx Virtex-7 XC7V2000T improved 44 times, and Block RAM improved 26.9 times. When using the new generation high-end FPGA, more PEs can be implemented on single high-end FPGA.

We also design a hwNet which realizes 192 PEs on a Xilinx Virtex-7 XC7V2000t FPGA to solve 3D Poisson equation with the similar design method. The parallel op-

Table 4.5 The specifications of Xilinx Virtex-7 XC7V2000T and Spartan-3 XC3S4000 FPGA

FPGA	Spartan-3 XC3S4000 FPGA	Virtex-7 XC7V2000T
Slices	27,648	305,400
Logic cells	62,208	1,954,560
CLB Flip-Flops	55,296	2,443,200
Maximum Distributed RAM (Kb)	432	21,550
Total Block RAM (Kb)	1,728	46,512

eration organization of 192 PEs are shown in Figure 4.15. In order to control 192 PEs, we use 6 PEs as a Block processor(BP), and each Block processor operates 10×10×6 grids. We implemented 4×4×2 BPs, which can operates 40×40×12 grids, on one Xilinx Virtex-7 XC7V2000T FPGA.

The Figure 4.16 shows the hwNet architecture on Xilinx Virtex-7 FPGA. In the design, a logical 3D mesh network interconnections among BPs was mapped on 2D layout of FPGA. Each BP consists of a PE controller, 6 PEs and a corresponding custom cache

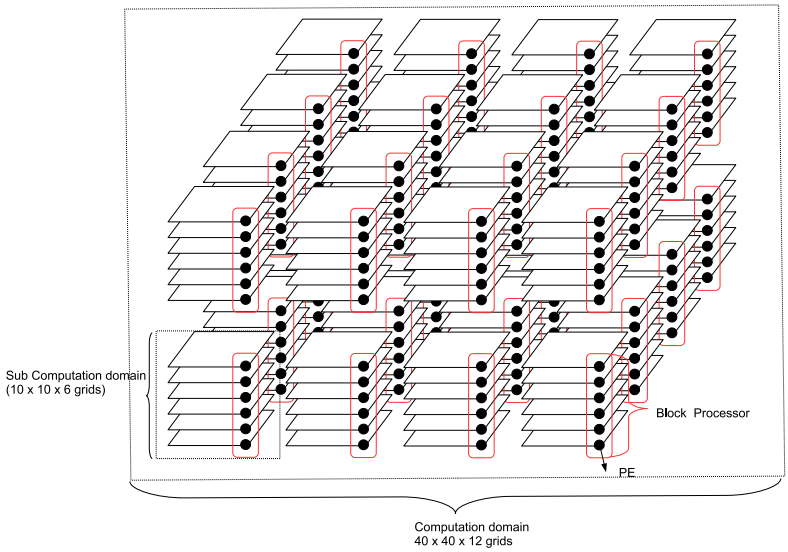

Fig. 4.15 Parallel operation organization of 192 PEs.

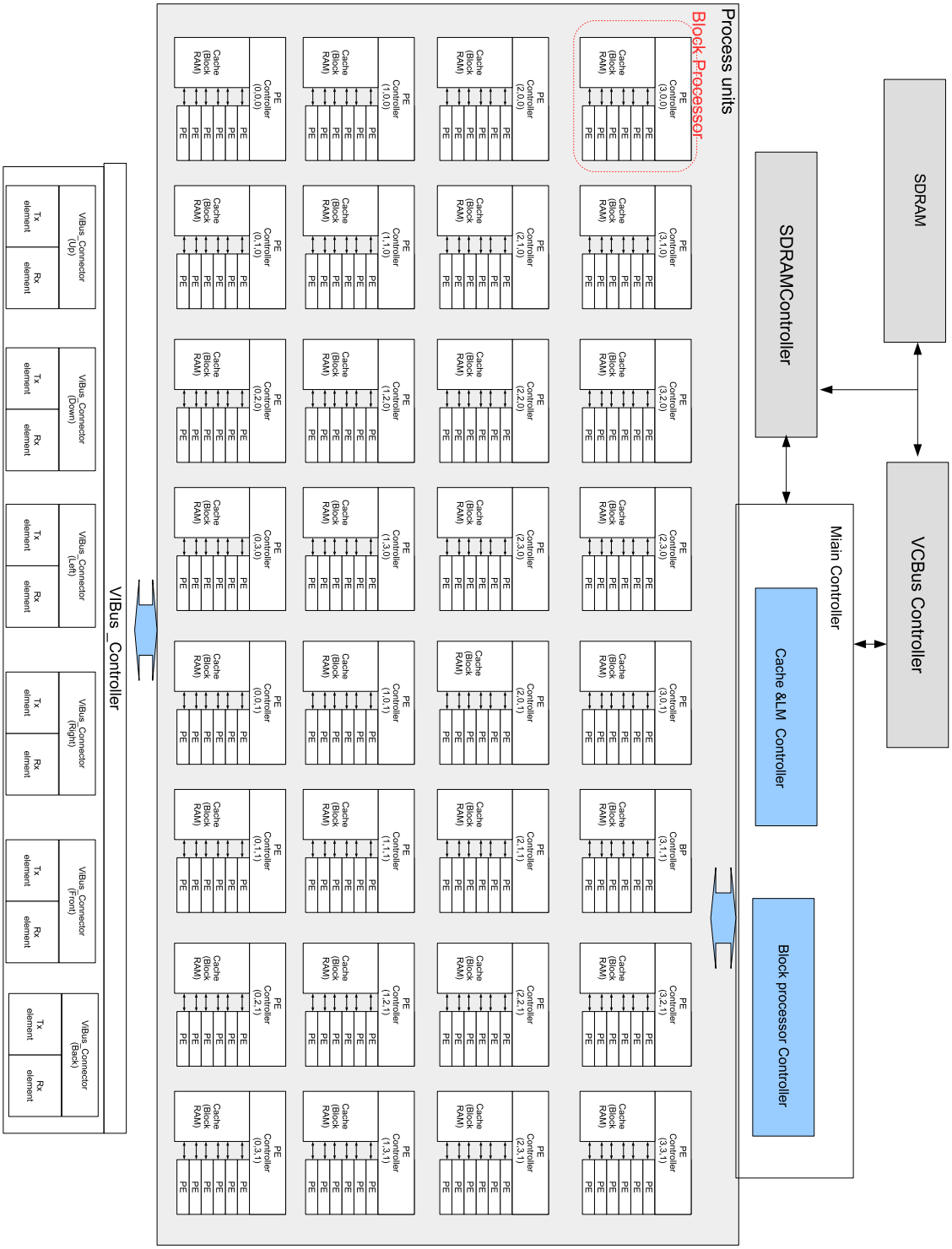

Fig. 4.16 The architecture of hwNet on Xilinx Virtex-7 XC7V2000T FPGA; 3D(logical network) to 2D(layout); 1Block Processor = 6PEs.

Table 4.6 The circuit scale of hwNet (4×4×2 Block processors(BPs) =192PEs) for 3D Poisson equation on a Virtex-7 XC7V2000t

Logic Utilization	3D Poisson(192PEs)		
	Used	Utilization	Available
Number of Slice registers	942,427	38%	2,443,200
Number of slice LUTs	886,868	72%	1,221,600
Number of bonded IOBs	452	37%	1,200
Number of Block RAM	799	61 %	1292

(Block RAMs) simultaneously. The PE controller can manage computing status of 6 PEs and data communication among inner PEs and cache on each BP. The BRAM&LM controller was used to write initial data, which stored in SDRAM, to each block processor, or read result data on each cache of block processor to SDRAM. The Block processor controller was mainly used to start up or stall the block processor array. The data communication among inner Block processors in a 3D mesh network are realized via directed links. We realized same data communication mechanism among adjacent FPGAs to hwNet on Spartan-3 XC3S4000 FPGA with a VIBus_Controller module. Through the synthesis process optimization by ISE's XST (Xilinx Synthesis Technology), the design hierarchical of the modules are flatten. The circuits scales of the hwNet is shown in Table 4.6.

We implements a hwNet composes of 4×4×2 BPs = 192 PEs on a Xilinx Virtex-7 xc7v2000t FPGA, to realize 863 GFlops at 500 Mhz(which can be realized on some examples design). Meanwhile, when 4×4×3 BPs = 288 PEs cannot be implemented on a Virtex-7 xc7v2000t FPGA , since its circuit scale limitation, slice LUTs utilization will be 112%, over 100%.

In order to implement more PEs on a chip, we improved the hwNet design, to reduce utilized circuits scale. We tested a improvement approach, to combined two intrinsic block processors on the z axis way as a new Block processor, which consists of 12 PEs and operates 10×10×12 grids as a decomposed sub-computation domain. For instance, BP(0,0,0) and BP(0,0,1) was combined as a new BP(0,0). It can be deem that a logical

Table 4.7 The circuit scale of hwNet (4×4 Block processors=192PEs) for 3D Poisson equation on a Virtex-7 xc7v2000T FPGA

Logic Utilization	Top module	BP(12PEs)	hwNet(192PEs)		
	used			Utilization	Available
Number of Slice registers	3,609	53,564	860,633	35%	2,443,200
Number of slice LUTs	695,596	43,108	695,596	57%	1,221,600
Number of bonded IOBs	452	0	452	37%	1,200
Number of Block RAM	31	48	799	61 %	1292

2D mesh computing network was mapped on the physical 2D network. Because of ISE's XST optimizes circuit to flatten the design hierarchical of BP module. We improved the hwNet design through the black box design method of FPGA, to maintain the BP module's boundaries design and avoid to be flatten design hierarchical of the BP module by XST(Synthesis Technology) optimization.

In the improved design, the BP module was designed as a black box model, so we can directly utilized the generated net list (NGC) file of the BP module to avoid the BP module was optimized by synthesis process. The hierarchical of BP module can be kept in the design. The overall hwNet' circuit scale can be calculated based on the circuit scale of top module of hwNet and BP module. The improved hwNet architecture was shown in Figure4.17, and the BP module (Black Box), the Top module of hwNet and the improved hwNet's circuit scale were shown in Table 4.7.

For conveniently describing two designs, we named the former hwNet design as 3*Dto*2*D* hwNet, the latter as 2*Dto*2*D* hwNet. Compared to the circuits scale of the two hwNet, the utilization of slice registers was down 3% (from 38%(3*Dto*2*D*) to 35%(2*Dto*2*D*)), the utilization of the slice LUTs was down 15% (from 72%(3*Dto*2*D*) to 57%(2*Dto*2*D*)). We can see the resource usage of slice has been effectively reduced, and more PEs are able to be implemented on the Virtex-7 XC7V2000T FPGA.

Meanwhile, see from the results, in the 3*Dto*2*D* hwNet design, a logical 3D mesh network was implemented on a physical 2D layout, thus implementation loss and rout-

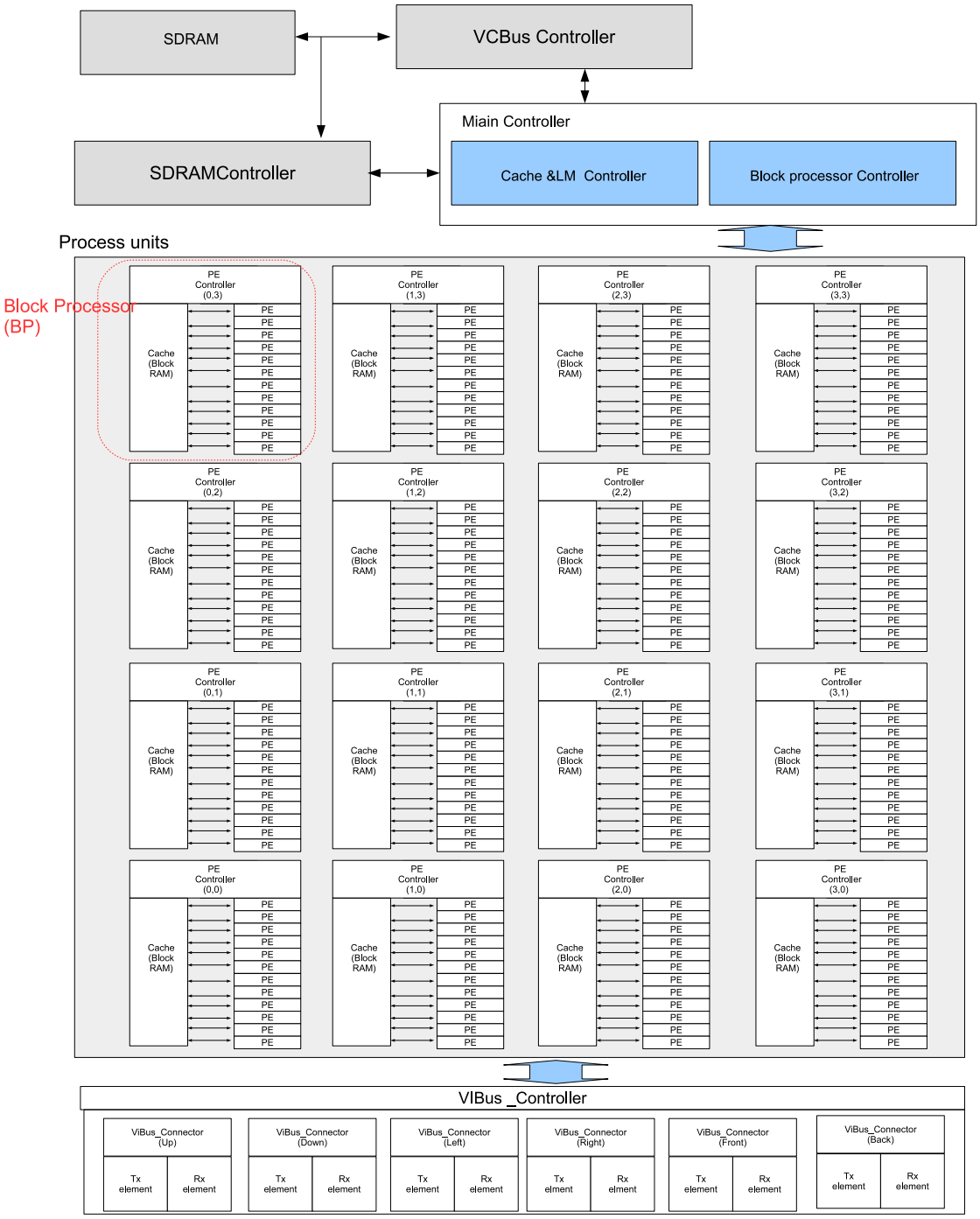

Fig. 4.17 The hwNet architecture on Xilinx Virtex-7 XC7V2000T FPGA.(1Block Processor = 12PEs)

ing problems has been seen in the case, compared to the 2*Dto*2*D* hwNet design . We speculate the cause is that the synthesis process of ISE is incapable of optimizing the 3D network algorithm, led to the optimized synthesis cannot do anything to minimize the interconnection among BPs on a 2D layout FPGA.

4.2 Performance evaluation and discussion

4.2.1 Experimental environment and comparison object

We implemented 1D, 2D and 3D advection equation with CIP method, and 3D Poisson equation with Jacobi method on our HPC system. As one of comparison objects, we also operate the same calculation with general processor. The experimental environment is shown in Table 4.8.

Table 4.8 Desktop computer Specifications

CPU	Intel Core i5 750 (2.53[GHz])
Mian Mem	DDR3 SDRAM PC3-10700 4[GB]
Mother board	Gigabye GA-P55M-UD2
OS	Windows XP Professional SP3
Compiler	Borland C++Builder 2006

4.2.2 Evaluation of advection equation with CIP method

We implemented a operation circuit for sloving 1D, 2D and 3D advection equation with CIP method with an FPGA on hwModule V2. Because of hwModule V2 and PVS equip the same Xilinx Spatran-3 XC3S4000 FPGA. The hwNet on hwMoule V2 can be easily ported to PVSs on FPGA array. As comparison objects, we also solving the same problem with Core i5 CPU. Through measure the computation time, we can evaluated the practical floating-point performance as following equation.

Table 4.9 Performance of 1D CIP(GFLOPS)

Iteration	10^3	10^4	10^5	10^6
FPGA (1PE)	0.197	0.817	1.353	1.392
CPU (1Core)	0.203	0.202	0.200	0.200

$$F = \frac{N \times P \times I}{T} \tag{4.26}$$

There, F is practical floating-point performance, N is computational domain size, P is number of operator units, I is iteration number.

■1D CIP method We solved 1D wave a propagation problem with CIP method by using a processing circuit on FPGA at 66 MHz. Meanwhile, we also solved the same computation with CPU as the comparisons. The computational domains of FPGA, CPU are a 1D space which composed of 512 girds. Through measuring the calculation time, the performances were evaluated at 10^3 to 10^6 Iterations respectively, shown in Table 4.9 and Figure 4.18.

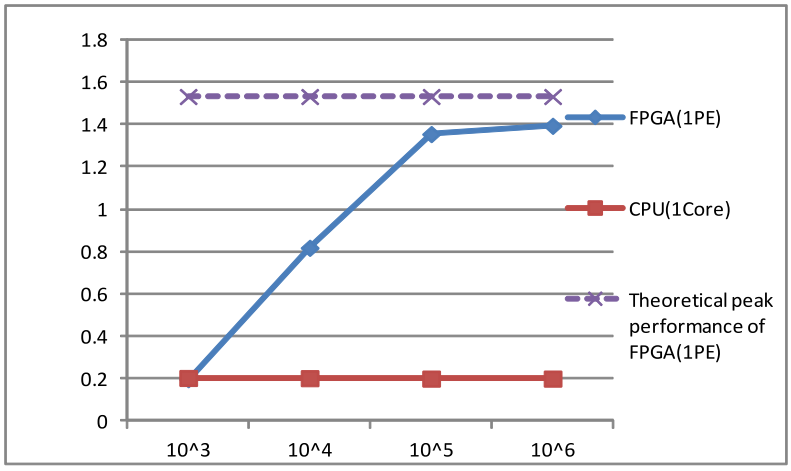

Fig. 4.18 Performance of 1D M-type CIP method

Table 4.10 Performance of 2D CIP(GFLOPS)

Iteration	10^3	10^4	10^5	10^6
FPGA (1PE)	0.222	1.057	1.548	1.627
CPU (1Core)	0.494	0.495	0.493	0.495

As iteration increased, the performance enhanced, and was asymptotically stable at 1.9 GFlops performance. This is because data communication and control cost among FPGA and host can be hidden in computing time as iteration increased. The performance of FPGA can achieves 95% of the processing circuit's peak performance (1.53 GFlops). While CPU performs about 200 MFlops of performance, almost have not unchanged over iterations.

■**2D CIP method** We solved the 2D advection equation with CIP method by using an FPGA. The computational mesh of FPGA composes of $16 \times 16 = 256$ grids. Meanwhile, we also solved the same computation with Core i5 CPU. The computational domain of CPU is same to FPGA, is $16 \times 16 = 256$ grids, The evaluated performances are shown in Figure 4.19 and Table 4.10. We evaluated the total computing time of FPGA, CPU

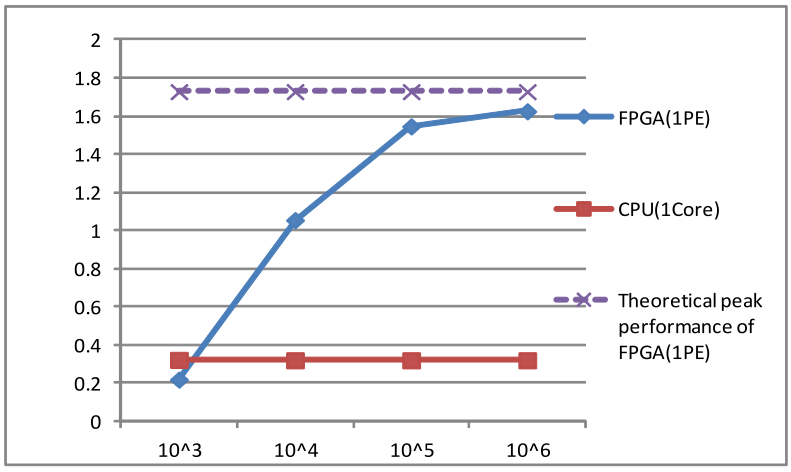

Fig. 4.19 Performance of 2D M-type CIP method

Table 4.11 Performance of 3D CIP(GFLOPS)

Iteration	10^3	10^4	10^5	10^6
FPGA (1PE)	0.278	1.218	1.827	1.916
CPU (1 core)	0.596	0.583	0.579	0.590

at different iterations. Figure 4.19 and Table 4.10 shows that results calculating the advection equation in two-dimensional mesh. As iterations increased, the control and transfer cost can be hidden, the performance of FPGA enhanced. The FPGA performs 1.627 GFlops performance, achieved 94% of peak performance (1.732 GFlops). While one core of CPU performs steadily about 490 MFlops at different iterations.

■**3D CIP method** The results of calculating the advection equation in three-dimensional cuboids space were shown the Figure 4.20 and table 4.11. The computational domains of FPGA and CPU were a 3D cuboids which composed of $7 \times 6 \times 6$. As well as 1D, 2D problem, the 1PE of FPGA can performs 1.916 GFolps of actual performance after $10\hat{6}$ iterations or so. The effective performance achieved 99 % of 1.93 GFolps of peak performance. One core of CPU (Intel Core i5) performs about 590

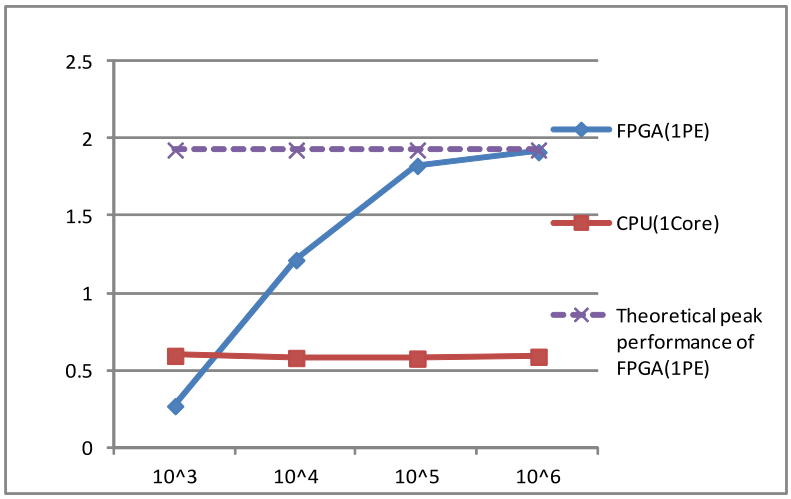

Fig. 4.20 Performance of 3D M-type CIP method

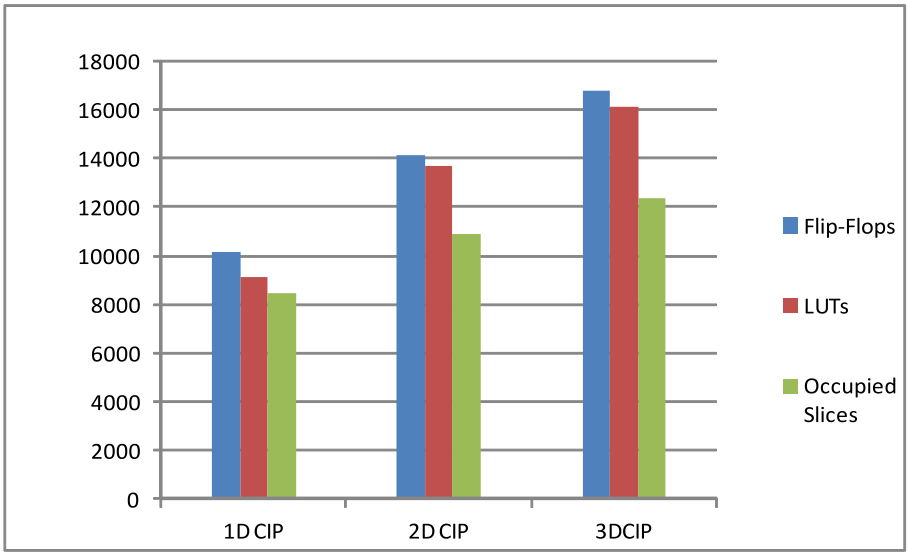

Fig. 4.21 Change of circuit capacity as the dimensions of operation domain increases

MFlops of performance for computing 3D CIP method.

Through the results of multidimensional advection equation with CIP method. The results shows that the processing circuits on an FPGA are high-efficiency, can achieve more than 90% of peak performances. One PE on FPGA running at 66 Mhz can achieves higher performance than 1 core of Core i5.

From Figure 4.18 - 4.20, when computational domain ascended each dimension, the arithmetic circuit performance enhanced about 18% in the case for solving advection equation with type-M CIP method. The arithmetic circuits on FPGAs are composed

Table 4.12 The required number of input variable, ALU and BRAM on a PE on different operation domain dimensions

	1D CIP PE	2D CIP PE	3D CIP PE
Number of variable	5	9	13
Number of ALU	22	26	29
Number of BRAM	2	3	4

of amounts of FFs(flip-flops), LUTs, Slices and BRAMs. As computational complexity increased, the arithmetic circuits scale was bound to increased to meet the computational requirements.

Meanwhile, Figure 4.21 also shows the changes of arithmetic circuit scale for solving the advection equation with CIP method as each dimension of computational domains increases. And Table 4.12 shows change of number of ALUs on one PE as input variable increases. Whenever ascending a dimension of computational domain, the utilized circuits resources (Flip-flops, LUTs, Slices) enhanced about 35% to satisfy the computing requirements.

Comparing the both changes between performance and circuits scale, we can see the rates of circuits scale increase are higher than rates of performance enhance when computational complexity increased. There are several main factors to cause the condition. As the computational complexity increased, the arithmetic-logic units(ALUs) on process element(PE) and PE's pipeline stage increased. Meanwhile, to keep the pipelined architecture of the PE, we need to utilize more FIFOs to adjust timing among inputs/outputs of ALUs.

Moreover, when dimensions of computational domain increased, there are more variables need to be inputted to the Processing element. This implies that data communication between data buffers and main arithmetic circuits becomes more complicated, and causes cache-control and other peripheral control circuits become enlarged.

4.2.3 Evaluation of Poisson equation with Jacobi method

Performance of 1PVS

The example circuit for a 3D Poisson equation was designed a sample benchmark to evaluate the performance of an FPGA array. We solved 3D Poisson equation problem with an FPGA Board (PVS); Thus, eight PEs can compute a cubical space of 10×10×8 grids at 66 MHz. The same computation was operated by Core i5 CPU for comparison.

In comparison, six PEs on 1PVS to compute cubical space of 10×10×6 grids at 66 MHz. The floating-point performance of a PVS can be calculated to measure the exe-

Table 4.13 Execution time and performance for 3D Poisson Equation on 1PVS

	Core i5(1 Core)		1 PVS (6PE)		1 PVS (8PE)	
Iterations	10^6	10^8	10^6	10^8	10^6	10^8
Execution Time [s]	21.9	2,187.39	1.60	155.34	1.57	152.34
Performance [GFlops]	0.33	0.33	3.37	3.47	4.57	4.72

cution time of different iterations. Table 4.13 and Figure 4.22 show the floating-point performance for the 3D Poisson equation with six PEs and eight PEs. As the number of iterations increased, the real performance (4.72 GFlops) of 1 PVS approached the its peak performance, which was 4.79 GFlops at 66 MHz. The results shows that 1 PVS (8PEs) can realized up to 14.3 times speedup than 1 core of CPU.

Performance of FPGA array

In many parallel and distributed systems, the time cost of communication among computing nodes is sizeable fraction of the total time needed to solve a problem. To develop insight into our experimental results, we made reference to [52], [53], and develop a simple analytical model of application sensitivity to communication overhead. We can think of the execution efficiency as the ratio:

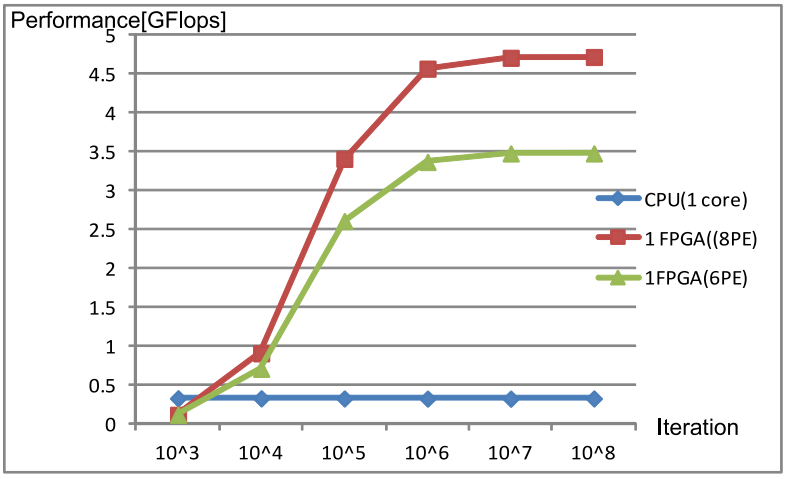

Fig. 4.22 Performance of 1 PVS for 3D Poisson equation

$$EF = \frac{T_{COMP}}{T_{TOTAL}} \tag{4.27}$$

where T_{TOTAL} is the time required by the algorithm to solve the given problem, and T_{COMP} is the corresponding time that can be attributed just to computation, that is, the time would be required if all communication were instantaneous. The time analysis is significant for a distributed system because of it is directly related to the efficiency of the system. In general, the computation time of a distributed system includes the evaluation time, and the communication time as following equation.

$$T_{TOTAL} = T_{COMP} + T_{OH} = T_{COMP} + T_{COMU} - T_{OL} + T_d \tag{4.28}$$

Where, T_{COMU} is the data communication time among computing nodes, T_{OL} is the overlap time among computation and data communication, T_d is delay for synchronous operation and control. The communication overhead $T_{OH} = T_{COMU} - T_{OL} = T_{Total} - T_{COMP} - T_d$. In our system, the synchronous time T_d can be negligible. Assuming that the environment is homogenous, then

$$T_{COMP} = \frac{Dt_p}{P} \tag{4.29}$$

where t_p is the time to evaluate one individual (a grid) and D is the computational domain for an FPGA, P is the number of PEs on one FPGA.

The communication time can be divided into three parts: queuing time, transmission time, propagation time. To analyze communication issues, it is helpful to view the distributed computing system as a network of computing nodes connected by communication links. We thus arrive at the following equation for the data communication time in a link:

$$T_{COMU} = P + At_c + Q, \tag{4.30}$$

where P is the processing and propagation time, A is the amount of transfer data (or number of bytes) and t_c is the time cost of single data, and Q is the queuing time.

In our system, we can reasonably assume that the processing and propagation time on a given link is constant, and the transmission time At_c is much larger than the processing and propagation time. The transmission time is proportional to the number of length of the packet, propagation time and queuing time can be negligible, we can roughly consider $T_{COMU} = At_c$.

The Eq 4.27 can be changed as

$$EF = \frac{T_{COMP}}{T_{COMP} + T_{COMU} - T_{OL} + T_d} = \frac{Dt_p/P}{Dt_p/P + At_c - T_{OL} + T_d}, \tag{4.31}$$

The communication overhead is the key factor to affect the distributed system's performance. Therefore, we designed a test vehicle with six PEs and six-way data communication circuits in each PVS for multidimensional connection FPGAs. The six PEs execute the operations on the 10×10 ×6 sub-grids for which the data communications of 10×10 and 10× 6 planes are required. The execution and data communication are also processed at 66 MHz. The synchronous data transmission among FPGAs is described in previous chapter. The exchange data quantity in all six directions varied. To verify the effect of exchange data quantity on the data communication overhead in 3D-connection FPGAs, we used a 1D FPGA array to evaluate the data communication overhead time among I/O connections in each direction (X, Y, Z-axis). In each 1D FPGA array, 3D Poisson equation is calculated with one data transmission from three directions at 10^7 iterations, as shown in Table 4. The execution times were measured with and without the data communications, and the communication overhead time along X-axis was $26.73 - 16.43 = 10.3sec$, $10.17sec$ for the Y-axis, and $18.44sec$ for the Z-axis. The results in Table 4 indicate that the data communication overhead is approximately linear with respect to the size of the data communication plane.

Table 4.14 shows the required data communication overhead between adjacent PVSs, where the 3D Poisson equation with 2×2 PVSs was evaluated (Figure4.23). We implemented 2×2 PVSs, i.e., 24($2 \times 2 \times 6$) PEs, to calculate the 3D Poisson equation. We also evaluated execution time for 2×2 PVSs without communications, when the 3D Poisson operations on each PVS are independent. Table 4.15 shows the communication

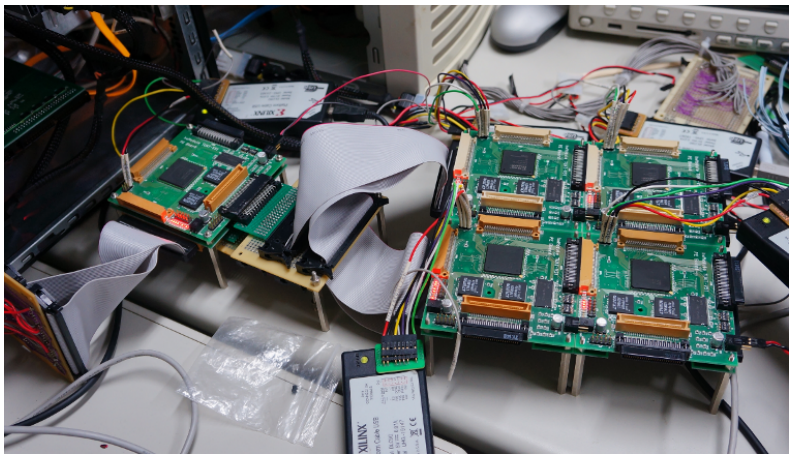

Fig. 4.23 A 2D FPGA array (2×2) PVSs for 3D Poisson equation.

overhead time 264.06 − 165.34 = 98.72*sec* for full 2D data communications with adjacent PVSs. The performance of 2×2 PVSs without and with data communication was 13.05 GFlops and 8.18 GFlops respectively. When calculating the 3D Poisson equation with a 3D FPGA array, adjacent PVSs achieve the data communications of 10×10 plane via the Z-axis (Up↔Down) connection. Therefore, we roughly estimated the 3D data communication overhead as $98.72 \times (10 \times 10)/(10 \times 6) = 164.53sec$. We estimated it in this manner because the 10×10 plane replaces the 10×6 plane as a bottleneck for the calculation interval.

We also calculated the 3D Poisson equation with a 3D FPGA array(2×2×2 PVSs implemented at 2×2×2×6 = 48 PEs) and evaluated the communication overhead(show in Figure 4.24).

The results are shown in Table 5. The 2×2×2 PVSs performs 25.63 GFlops without communication and 12.46 GFlops with communication. The communication overhead time among $2 \times 2 \times 2$ PVSs is 346.46 − 168.54 = 177.92*sec*. The measured value (177.92*sec*) is approximately the same as the estimated value for the communication overhead (164.53*sec*), with an 8% error. Moreover, for the Poisson calculation, when the internal grids were increased, the data communication ratio decreased. Based on these

Table 4.14 Execution time and data communication among 3D connection for 3D Poisson Equation.(Ex Time = Execution Time, Iteration=10^7)

Transmission direction	X-axis way (Front↔Back)	Y-axis way (Left↔Right)	Z-axis way (Up↔Down)
Exchange Data Plane	10×6	10×6	10×10
Ex Time with Communication [s]	26.73	26.52	35.34
Ex Time without Communication [s]	16.43	16.35	16.9
Communication Overhead [s]	10.03	10.17	18.44

results, the performance of 8×4 × 4 FPGAs can be estimated; 128 FPGAs implemented at 8×4×4×6 = 768 PEs with communications can achieve 199.36 GFlops. It is possible to work at 500 MHz for high-speed operation. The effective performance of system (8×4×4 FPGAs) can be expected to realize 1.5 TFlops performance running at 500 Mhz.

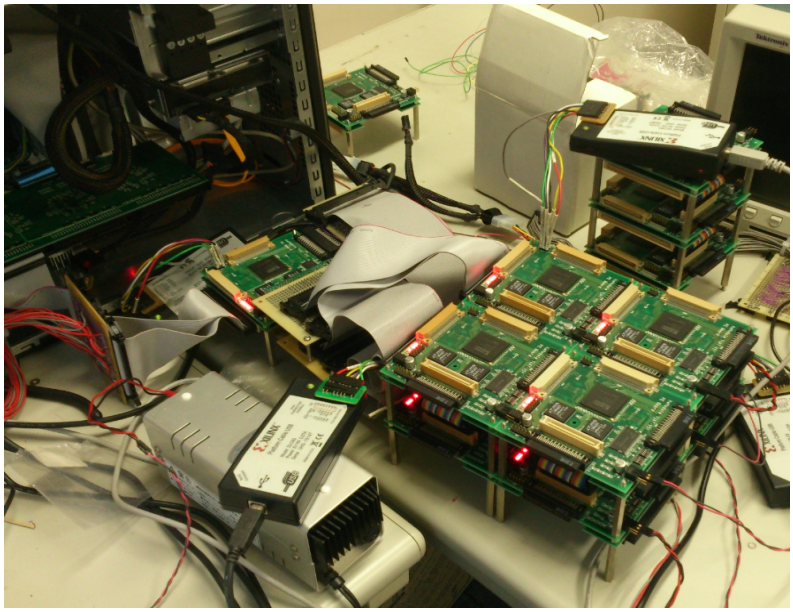

Fig. 4.24 A 3D FPGA array (2×2×2 PVSs) for 3D Poisson equation.

Table 4.15 Execution time and performance for 3D Poisson Equation on 2D(2×2 PVSs) and 3D(2×2×2 PVSs) FPGA array(× : without communication ◯ : with communication)

	2×2 PVSs		$2 \times 2 \times 2$ PVSs	
Communication Available	×	◯	×	◯
Iterations	10^8	10^8	10^8	10^8
Execution time [s]	165.34	264.06	168.54	346.46
Performance [GFlops]	3.26	8.18	25.63	12.46

4.2.4 Discussion of the distributed system

In our experiments, 3D Poisson equation was solved with a 3D FPGA array which consists of 2×2×2 PVSs.

We used a fundamental transfer and synchronization mechanism to implemented data communication among FPGA via 3D directed connection. In this case, the data communication among FPGAs was realized via direct six-way connections , the figure 4.25 is the simulated time chart which shows computation and data communication on one chip of 3D FPGA array in the prototype design.

And data transmission and data reception in each link was completed respectively with two stages at each iteration computation. The data communication time in each link can be consider to $T_{COMU} = T_{S1} + T_{S2}$, where T_{S1} and T_{S2} was required process time at two communication stage. The execution efficiency Eq. 4.31 can be expressed as following:

$$EF = \frac{T_{COMP}}{T_{COMP} + T_{S1} + T_{S2} - T_{OL} + T_d}, \tag{4.32}$$

In a time step, considering the pipeline length can be hidden as iteration increases, and each PE operates a sub-block which consists of 10×10×1 grids, so T_{COMP} could be considered 10×10 clk. The data communication on 1st stage can be overlapped to computation, the overlap time T_{OL} can be roughly considered to $T_{OL} = T_{S1}$. And the data communication in links on z-axis way became the bottleneck of data transfer, so the data communication overhead was $T_{OH} = T_{S1} + T_{S2} - T_{OL} = T_{S2} = 100clk$. Based on

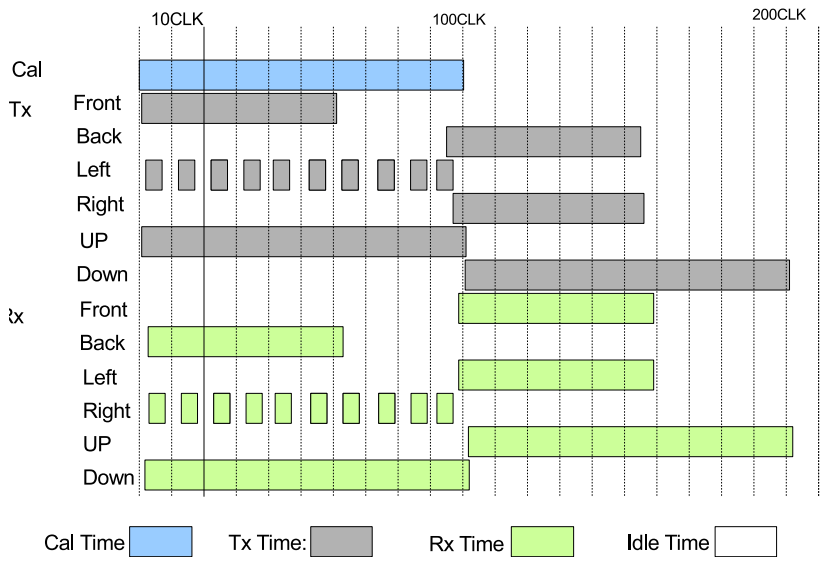

Fig. 4.25 Time chart of one FPGA on 3D FPGA array in a time step

the analysis of the existing design, we can predict the execution efficiency with Eq 4.32. The evaluated value of efficiency is about 50%. Meanwhile, based on measured results shown in Table 4.15; the practical execution efficiency $EF = 168.54s/346s = 48.7\%$, with an 5.2% error. The results shows that design value of the system was realized.

However, the implemented data communication mechanism in this case is inefficiency, but it can provide steady data communication in the prototype design stage. By analyzing the computation and data communication situation in the worse case, it is easily to find the ways to improve the system performance.

The results show the data commutation overhead among FPGAs became the performance bottleneck of FPGA array in the case. By reducing the communication overhead, the computational efficiency of FPGA array can improved. Several the most important factors that influence the communication delays are the following:

Table 4.16 The predicted computational efficiency as computational domain size increase

Computational domain size	5^3	10^3	20^3	40^3
Data communication size	5^2	10^2	20^2	40^2
Computational efficiency[%]	45.45	62.50	76.92	86.95
Computational domain size	80^3	160^3	320^3	640^3
Data communication size	80^2	160^2	320^2	640^2
Computational efficiency[%]	96.38	96	98	99

Computational domain size

In general, the computational domain size has a huge impact on execution efficiency of a distributed system. It is commonly known that communication overhead for domain-decomposition can be hidden through overlapping communication with computation when a problem size is sufficiently large. We assume that a decomposed sub-block of N×N×N grids has N×N×N complexity for computation while an exchange of boundary data has N×N complexity. Then parallel computation with a 3D direct connection can scale as long as the communication overhead is hidden. In the prototype design for solving poison equation, the computational efficiency EF can be calculated as following:

$$EF = \frac{T_{COMP}}{T_{COMP} + T_{OH}} = \frac{N^3 t_p / P}{N^3 t_p / P + N^2 t_c} \tag{4.33}$$

there, the number of PEs: $P = 6$, and $t_p = tc$. The Table 4.16 and Figure 4.26 shows the changed trend of computational efficiency as population size(N^3) changes.

By increasing the computational domain size, the data communication overhead can be effectively improved. Because of a PVS has two 16 MB SDRAM as local memory, which can stores 8×10^6 number of single precision floating-point data. It means that there are 4×10^6 number of ρ and ϕ can be sorted at local memory separately. The maximum computational domain sizes of single PVS were $158 \times 158 \times 158$ grids, the execution efficiency is 96.34%. It can be estimated that a PVS on a 3D FPGA array can realize the 3.33 GFlops equivalent performance.

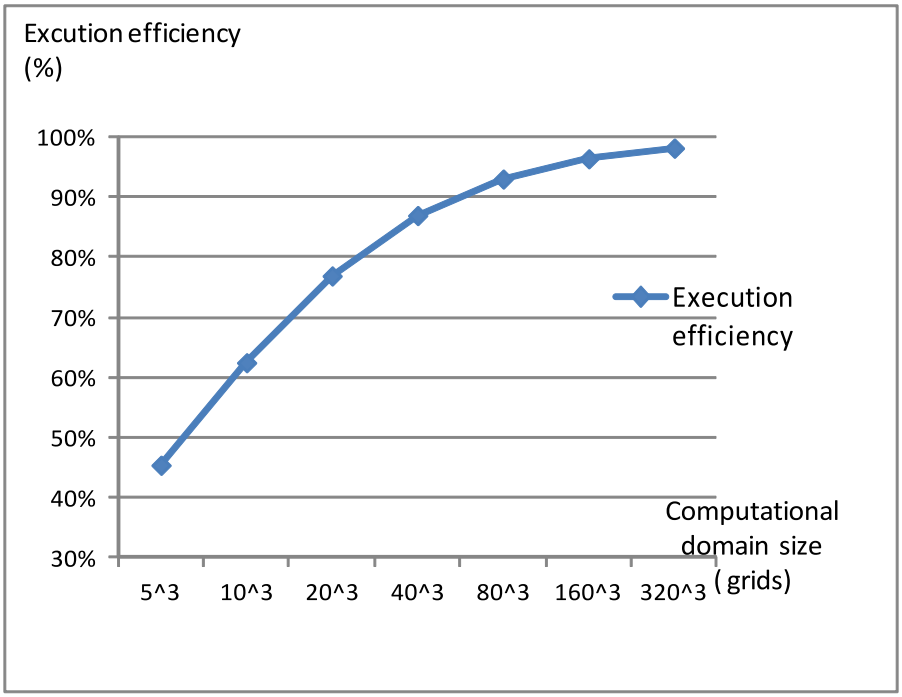

Fig. 4.26 The predicted computational efficiency as computational domain size increase

Operation and communication frequency

The communication frequency influences the data bandwidth in each link, and determines the communication time from nodes to other nodes. For the distributed system in general, the frequency is very important since the node rely very much on boundary information that is communicated among nodes.

We assume f_c is communication frequency, and f_p is operation frequency of PE, and the frequency ratio is $\alpha = f_c/f_p = t_p/t_c$; The Eq. 4.33 can be expressed in following form:

$$EF = \frac{T_{COMP}}{T_{COMP} + T_{OH}} = \frac{D/P}{D/P + A/\alpha} \tag{4.34}$$

In this case design for solving 3D Poisson equation, $D = 10\times10\times6, P = 6, A = 10\times10$;

Table 4.17 The predicted computational efficiency as frequency ratio α of communication/computation increases.(Computation domain of a PVS consists of $10\times10\times6$ grids)

Frequency ratio	5^3	10^3	20^3	40^3
Computational efficiency[%]	45.45	62.50	76.92	86.95
Frequency ratio	80^3	160^3	320^3	640^3
Computational efficiency[%]	96.38	96	98	99

and the operation and communication frequency were same at 66 Mhz, so $\alpha = 1$. Table 4.17 shows the variation trend of computational efficiency of an FPGA array when only enhance frequency ratio α of communication/computation.

The communication frequency is usually much faster than processor in practice implementation. Based on the ISE timing summary of synthesize reports, the specific VIBus_module can achieves the maximum frequency 355.36 Mhz. It means that the communication frequency among FPGAs can be realized at 330 Mhz, which is 5 times of computation frequency (66 Mhz), and the time chart simulation is shown in the Figure 4.27. Contrast to the time chart in Figure 4.25, when communication operation runs at 66 Mhz, the communication overhead has been greatly reduced, and the computational efficiency was enhanced to 83.3% when only raising the communication frequency.

Meanwhile, in general, we assume the computation domain size of one FPGA was N×N×N grids, It is obvious that a high communication frequency will improve the speed of convergence of execution efficiency. We can estimate a 3D FPGA array can achieve to 99.3% computational efficiency, when each PVS operates $158 \times 158 \times 158$ grids data at 66 Mhz, while transfers data in each interconnection link running at 330 Mhz.

Algorithm and data communication mechanism

We provided a prototype design of 3D FPGA array for solving 3D Poisson equation. The algorithm of PEs for solving 3D Poisson equation was not optimized for 3D data communication network. Through optimizing the computation algorithm and communication mechanism, that is an effective method to improve the execution efficiency of a distributed system. Because of the communication overhead: $T_{OH} = T_{COMU} - T_{OL}$. In

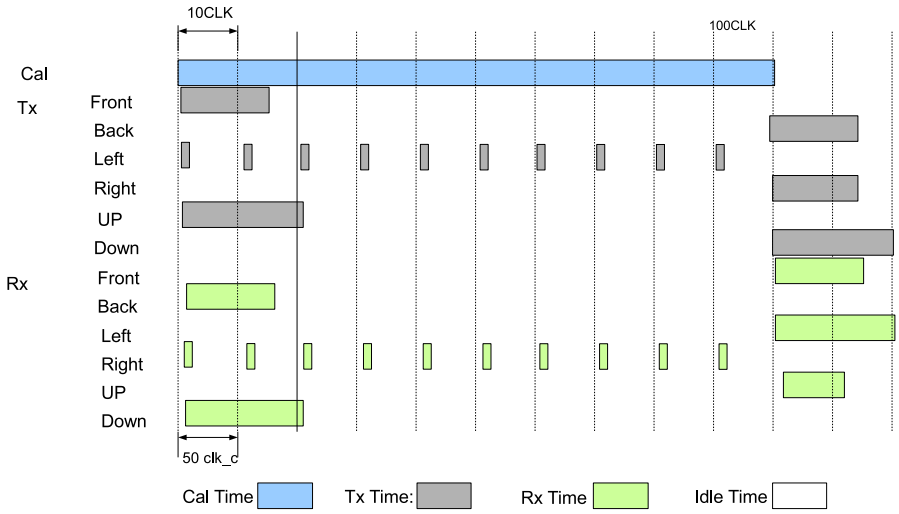

Fig. 4.27 Time chart of one FPGA on 3D FPGA array in a time step (communication frequency at 330 Mhz)

actual design, due to the computation domain size and communication frequency was generally limited by realistic conditions such as the hardware resources, chip manufacturing processes, memory sizes and so on. For a given problem solution with a hardware architecture, it is a effective way to reduce the required data communication and enhance overlapping time of computation through optimizing the computation and communication algorithm.

See from the time chart(Figure 4.25) of one FPGA on 3D FPGA array, the data communication the interconnection links in 6-way has to spent much idle time to wait the result data which output from PEs. By optimizing the algorithm and communication strategies, it is possible to realize more data communication overlap to computation. For instance, in the this case, in order to reduce the communication overhead, the algorithm and communication mechanism can be realized with the following approach. (shown in Figure 4.28)

There, each PVS implements *N* PEs to operate the N×N×N grids as a sub-block of computation domain, and each PE operates a sub-mesh which consists of N×N×1 grids.

In n time step, the PVS starts operation from the (0,0,0) gird to (N,N,N)grid along

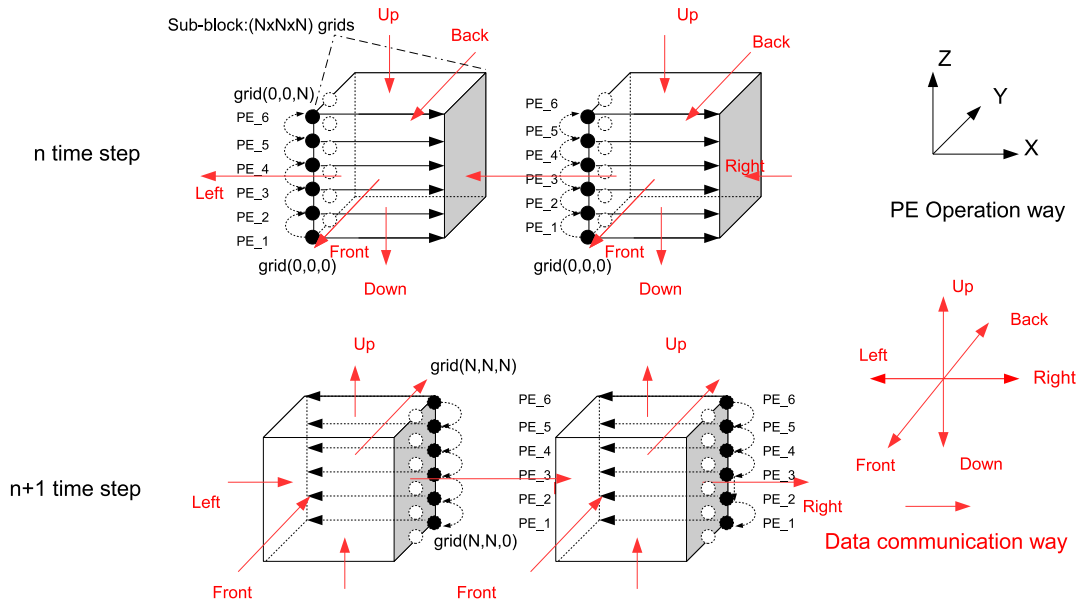

Fig. 4.28 A improved method to reduce communication overhead among FPGAs with optimizatizing algorithm and data communication mechanism

the positive direction of (X,Y,Z axis) in Cartesian coordinate (shown in Figure 4.28). For example, PE_1 firstly operates computations from grid(0,0,0) to grid(N,0,0). Then, PE_1 achieve computations from grid(0,1,0) to grid (N,1,0). Finally, PE_1 complete the computations from grid(N,N,0) to gird(N,N,1) at a iteration with the operation flow. Meanwhile, the PVS only transmits the operated boundaries data on Front, Left, Down sides via the corresponding VI Bus links, and receipts the boundaries data of Right, UP, Back sides from corresponding adjacent FPGAs for the next time step(n+1) computation.

In the next time step (n+1), the computation was operated from the grids(N,N,N) to (0,0,0)grid along the negative direction of (X,Y,Z) axis with the operation flow which is opposite to n time step. Similarly, the PVS only achieves the data communication in opposite directions of n time step, the operated boundaries data on Right, UP, Back sides were send to corresponding adjacent FPGAs, and boundaries data on Front, Left, Down sides were receipted from adjacent FPGAs for the next time step (n+2) computation. In the n+2 time step, the PVS executes the same process and communication as in n time

step, and so on, repeating the operation until the calculation is completed.

By using this approach, the communication data in each iteration can be reduced by half, and the data communication can almost overlap to data computation, although the improved method has error compared to previous one. In next works, we will implement the improved method on our system and measure the actual computation error.

By accounting for these factors, the data commutation overhead can be improved and the data communication almost completely be hidden. The data communication overhead is no longer the primary bottleneck, and the performance of 3D FPGA array can be improved considerably. It is possible to realized the nearly 100% execution efficiency. Because of the FPGAs used in the proposed system are Spartan-3 XC3S4000, which were released in 2008, only were equivalent to four million ASIC gates for a 90-nm process. The results do not show superior performance compared to near-term high-end processors. When the system utilizes the new generation high-end FPGA, the computing power enhanced as more parallel PEs have been implemented and operated at higher frequency. The system also can achieve higher off-chip communication bandwidth with new high speed I/O connector solution. In many related works have shown that Finite-Difference Time-Domain (FDTD) computation is suited to FPGAs; thus, a system with multi-FPGAs can deliver dozens of times computation acceleration compared to same term processor[46], [47].

We also designed a hwNet which can implements 192PEs on a Xilinx Virtex 7 XC7V2000T FPGA (described in Section 4.1.2). The one Virtex 7 XC7V2000T FPGA can performs 863 GFlops peak performance at 500 Mhz. When a large scale 3D FPGA array is built with $22 \times 22 \times 22 = 10,648$ Virtex 7 XC7V2000T FPGAs in Volicase method, the system can implements $22 \times 22 \times 22 \times 192 = 2,044,416$ PEs to realize 9.189 PFlops of performance for solving Poisson equation, more than K supercomputer peak performance (8.162 PFlops).

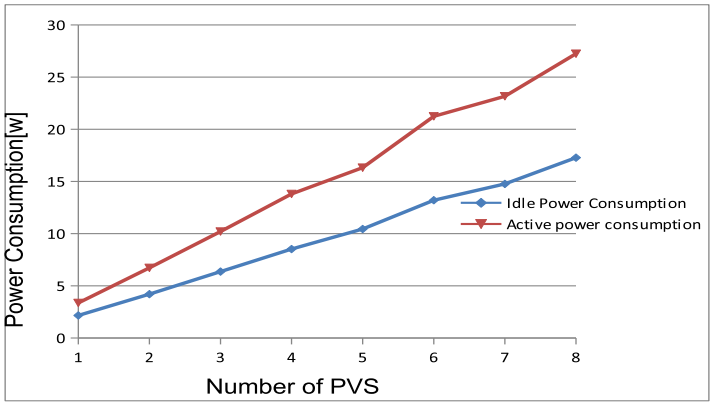

Fig. 4.29 Power consumption of FPGA array

4.3 Power consumption measurement

The FPGA array is powered by a 12V DC power supply. We acquired the power consumption by measuring the current value.

We measured respectively the actual power consumptions of 1 PVS to 8 PVS, when operating 3D Poisson Equation at 1×10^8 iterations. Meanwhile, we also measured the Idle power 1 PVS to 8 PVS, when the circuits are configured. The measured results were shown in Figure 4.29.

The power consumptions growth as numbers of PVS increase. The actual load power consumption of a single PVS was only 3.36 W, and the idle power of the PVS was 2.11 W, exceeded 50% of its actual power consumption. This is because of the utilized DC-DC converters on FPGA Boards only can provided 65% conversion efficiency. By using high-efficiency DC-DC converters or directly providing stable output voltage to FPGA, the idle power and actual power consumption can be reduced significantly. In contrast, we also measured the idle power consumption and active power consumption of CPU on host PC when operating the same calculation. The idle power consumption of CPU was

9.72 W, and actual power consumption was 31.8 W. The power consumption of FPGA is one-tenth than a general-processor so that without additional cooling devices design for the system. The power efficiency of single PVS was 1.40 GFlops/W when eight PEs are implemented on one PVS for solving 3D Poisson equation. We can estimate that a 3D FPGA array $4 \times 4 \times 8 = 128$ PVS consuming power of 435 W when operating 3D Poisson equation.

Meanwhile, we estimated the power consumption of calculation circuits which consists of 192 PEs implemented on a Xilinx Virtex-7 XC7V2000T with Xilinx Power Estimator(XPE) [56]. The Xilinx Power Estimator (XPE) spreadsheet is a power estimation tool typically used in the pre-design and pre-implementation phases of a project. XPE assists with architecture evaluation, device selection, appropriate power supply components, and thermal management components specific for your application.

The power consumption can be predicted through considering the design's resource usage, toggle rates, I/O loading, other factors which it combines the device models to calculated the estimated power distribution(show in Table 4.18).

Table 4.18 The estimated power consumption of hwNet on a Xilinx Virtex-7 XC7V2000T with Xilinx Power Estimator(XPE)

Resource	Power Consumption	
	Power value [Watt]	Percentage [%]
Leakage	1.75	4
Clock	12.32	31
Logic	17.88	44
BRAM	1.93	5
I/O	6.42	16
Total	40.30	100

The power consumption of a Virtex-7 XC7V2000T FPGA was 39.9 W running at 500 Mhz(Junction Temperature 58.4 °C). One Virtex-7 XC7V2000T can implements 192 PEs, that equivalents to 4×4×2 PVSs(Xilinx Spartan-3 XC3S4000). The power efficiency of efficiency of a Virtex-7 XC7V2000T was $863GFlops/40.30W = 21.4GFlops/W$, and it is 15.4 times more efficient than

PVS (Spartan-3 XC3S4000). We also estimate an FPGA array which composes of $22 \times 22 \times 22 = 10,648$ Virtex 7 XC7V2000T FPGAs can meets equal levels of K supercomputer's performance with consuming 429,114 W power consumption, only about 4% power consumption of the K supercomputer(9.89 MW).

Yet, the No.1 ranked machine in the Green 500 list(in November/2013): TSUBAME-KFC supercomputer, which combines the computing power of two Intel Xeon ES-2620 processors with four NVIDIA Tesla K20X graphics processing engines per node, can realized more than 150 TFLOPS of computation running the LINPACK benchmark and efficiency of 5.27 GFlops/W, it is less than one fourth times than the perfected efficiency of a Virtex-7 XC7V2000T FPGA.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In the work, we build a reconfigurable HPC system with a 3D FPGA array by using hw/sw complex architecture. The operations of the object-oriented applications have been partitioned into swObjects processed by CPU and hwObjects on initial stage, which are operated by processing circuits (hwNets) on FPGA array of design, and the user can utilize hwObjects like swObjects. The amounts of peripheral interface circuits of FPGA array were standardized, and circuit configuration, control/access of FPGA array can be hidden in hw/sw complex units. The designers just require to redesign the hwNet and hwObject interfaces to solve a new applications on distributed computing by using FPGA array.

Vocalise is a scalable multidimensional interconnection FPGA array computing platform. We developed the parallel and flexible circuit configuration solution for a large-scale multidimensional FPGA array. This solution enables easy implementation of circuit configuration for 2D/3D FPGA arrays . The host PC can concurrently configure most arithmetic circuits on 32 PVSs through each BVS through the configuration circuits implemented on FPGAs.

We designed specific circuits for solving CIP method and 3D Poisson equation. The results of the application specific circuits on Vocalise showed operational efficiency, scal-

ability, communication overhead and power consumption. One PVS which has 1PE was implemented can realize 1.91 GFlops at 66Mhz for solving 3D CIP method, about 99% of its peak performance (1.93 GFlops). These showed efficiency and high-utilization of the configurable computing system. The results of 1D,2D and 3D CIP method also show a realistic problem. With computational dimensions increases, hwNet's circuit scale growths are faster than the performance improvements; it shows the computational capabilities of one FPGA was very limited for solving most HPC applications, and the necessity of a multiple FPGAs system. The PVS implemented 8PEs to achieve 4.72 GFlops performance for the 3D Poisson equation, and 2×2 PVSs, which used 2×2×6 PEs with communication, achieved 8.18 GFlops. The 3D FPGA array (2×2×PVSs) which composed of 2×2×2×6 PEs, achieved 12.46 GFlops with communication. We estimated that a 8×4×4 FPGAs, which used 8×4 ×4×6=768 PEs with communications, achieve 199.54 GFlops by consuming power of 435W. Through designing the a hwNet on a high-end FPGA: Virtex XC7V2000T FPGA for solving 3D Poisson equation, we estimated a XC7V2000T FPGA implemented 192PEs can achieve 863 GFlops running at 500 Mhz, with 39.9 W power consumption, and a 3D FPGA array composes of $22 \times 22 \times 22 = 10,648$ Virtex 7 XC7V2000T FPGAs can meet equal levels of K supercomputer's performance, with consuming 429,114 W power consumption, only about 4% K supercomputer's power consumption(9.89 MW). The system can be as a prototype design of a multidimensional FPGA array to provide practical reference significance when required to build a new generation RHPC system with multidimensional connections among FPGAs.

Furthermore, in the hwNet designs on Virtex XC7V2000T, we separately developed two hwNets (2*Dto*2*DhwNet* and 3*Dto*2*DhwNet*) to map a logical 2D mesh network(4×4 BPs) and a logical 3D network(4×4×2 BPs) on FPGA's physical 2D layout for solving same 3D computation. Compared the two cases, we also found the 3*Dto*2*DhwNet* circuits scale which mapped a logical 3D mesh network on a physical 2D layout, required more 15% circuit resources(LUTs) as routing loss than 2*Dto*2*DhwNet* case.

The system can implement different network topologies using the multi-dimensional direct interconnection, and this scalability is critical to improve communication perfor-

mance. The network can be configured to provide application-specific data communication for each application.

In addition, numerous technologies for 3D integration are becoming available, such FPGA 3D packaging, the interconnect length among FPGAs can be reduced greatly. It means that more chips can be implemented on 3D FPGA array into a limited space, high density system level integration high density system level integration.

5.2 Related and future work

In the study, we demonstrated the capacity of Vocalise system with a 3D FPGA array for 3D PDE problems.

In future, we will continue the study as following aspects.

The improvement of Vocalise system

We will improve the data communication efficiency among FPGA array through a variety of approaches, such as optimizing commutation mechanism and parallel algorithm, enhancing the computational size on each computing node (FPGA) and enhancing the communication efficiency. We will improve stability system such data communication among host and FPGAs, circuit configuration on a large-scale 3D FPGA array. To realize a 8×4×4 FPGA array and evaluate the performance for solving 3D PDE problems in real system.

On the other hand, in order to explore merits and faults of our approach in various multidimensional application problems, not only PDE problems. We will use the multidimensional FPGA array to achieve a broad spectrum of applications such real time parallel brain processes or massively parallel web applications.

Parallel brain processes applications in real time

An artificial intelligent(AI) system requires huge computation for searching processes, recognitions, memorizing, or recalling. In previous related works, by using hw/sw complex, we have developed and several circuits of brain processes to realize artificial func-

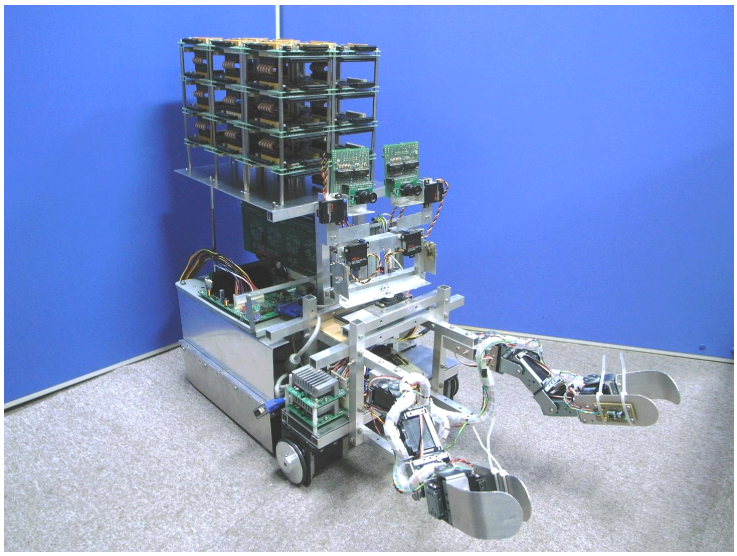

Fig. 5.1 A robot with a 3D FPGA array($2 \times 3 \times 4$ FPGAs) for brain processes

tions of voice recognition, voice synthesis, and image recognition, on hwModule V2 proposed in [28], [29]. The results shown the specific processing circuits, built with FPGA, for solving pre-processes or post-processes of brain processes such as voice recognition, image recognition and voice recognition, can effectively improve operational speed or reduce overload of CPU. By using the hw/sw complex architecture, real time process applications are able to be realized. After analyzing the results for brain processes such as voice recognition, voice synthesis and image recognition with self-organizing map, we roughly estimated gate count of the circuits for the brain processes. These distributed processes can be realized in real time with an FPGA array.

In next work, we will implement more software and circuits of the brain processes by with a 3D FPGA array, to realize an AI robot which can implements real-time brain processes in parallel(shown in Figure 5.1). Since robot is battery-powered, so the low power and performance/watt of the system is the key to implement real-time brain operation. The various brain processes were implemented on the complicated hybrid network. It is a challenge that optimal mapping and routing various brain processes on a suitable network dimensions and topologies with multidimensional FPGA array to effectively

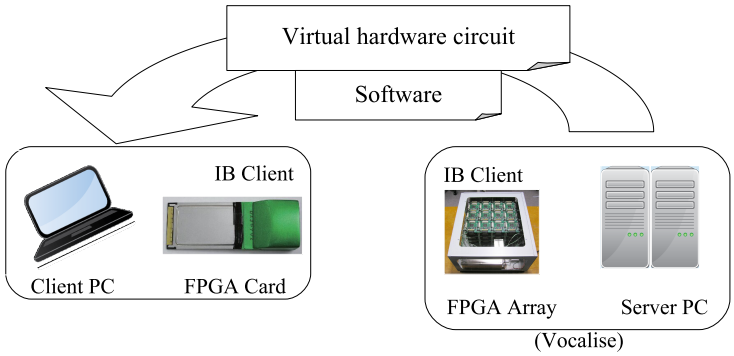

Fig. 5.2 The proposed implementation methodology of web application(Internet Booster and FPGA array server.)

execute multiple brain processes in parallel.

Web applications

The high performance and low power server is desired, to execute heavy load parallel applications from a large number clients. Recently, FPGAs were utilized to accelerating large-scale server in some studies. For example, Microsoft used medium-end FPGAs connected in a tours network to accelerate the Bing web search [17]. Meanwhile, in the related pervious work, we also proposed a hardware-accelerated web application platform for a power-efficient and high-performance computing system based on the FPGA array server and the mobile FPGA card in [23][50]. A new implementation methodology (Internet Boost) of web applications have been proposed, the concept of the proposed methodology is shown in Figure 5.2. We realize a networked hw/sw complex system and implemented Internet Booster to imposing it on a mobile FPGA board to accelerate web application. The experiments show that a hwNet implementation is 25 times faster than the software implementation especially for the Encoder component(a video streaming application), and can be used in real-time applications while the software implementation is not suitable for this purpose. Meanwhile, the CPU overload have been reduced vastly.

It is also full of challenges to realize distributed web applications with networked

hw/sw complex on a multidimensional FPGA array. In next work, we will introduce a hard-disk array to storage massive data into the FPGA array. To realize low power consumption and high performance WEB application server with multidimensional FPGA array with our method, to accelerate massive applications from lots of clients.

To demonstrate high-performance and low-power with multidimensional FPGA array on a broad spectrum of applications. As the more FPGA chips have been required, a lower price has become important. Our final goal is to realize an application specific personal computer including scalable multidimensional FPGA arrays.

Acknowledgments

My deepest gratitude goes first and foremost to Prof. Masatoshi Sekine, my supervisor, for his wonderful advice, invaluable guidance, understanding, patience enormous support and encouragement on my research work. His mentorship was paramount in providing a well rounded experience consistent my long-term career goals. His constructive criticism and dedication to academic research stimulate my research ideas. Without his consistent and illuminating instruction, this thesis could not have reached its present form.

At the same time, I have to express my sincere gratitude to Dr. Hakaru Tamukoh, the assistant professor of Sekine lab, he helped me to know hw/sw complex and hardware design technique and gave me many good advices for my study career. I would also like to thank the members of Sekine lab, especially for the HPC study group members for their input, vaulable discussions, friendships, encouragements, support and collaboration. They are: Yusuke Atusmari, Hiromasa Kubo, Yuichi Ogishima . Without their works and helps, the Vocalise system can't be built in past several years.

Lastly, I would like to thank my beloved family for their loving considerations. For my parents who raised me with a love of science and supported me in all my pursuits. For my wife Jun Lai, her support, encouragement, quiet patience and unwavering love were undeniably the bedrock upon which the past five years of my life have been built. Without them, I would be lost. They have supported my every endeavor and supported my choices in life, and for this, I am forever indebted. My family makes life both meaningful and fun.

This study is supported by program "Research and development of ultra-high-speed

hw / sw complex system with dynamic virtual circuit” (“1811 動的な仮想回路による超高速 hw/sw 複合システムの研究開発”) of Janpan Science and Technology Agency(JST).

Bibliography

[1] V. Kindratenko and D. Pointer, “A Case Study in Porting a Production Scientific Supercomputing Application to a Reconfigurable Computer,” Proc. of 14th Ann. IEEE Symp. Field-Programmable Custom Computing Machines, pp. 13-22, 2006.

[2] O. Mencer, “ASC: A Stream Compiler for Computing with FPGAs,” IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 25, no. 9, pp. 1603-1617, 2006.

[3] J. Makino, K. Hiraki, M. Inaba, “GRAPE-DR: 2-Pflops Massively-parallel Computer with 512-core, 512- Gflops Processor Chips for Scientific Computing,” Proc. of ACM/IEEE Conf. on Supercomputing, pp. 1-11, 2007.

[4] Y.S. Hwang, R. Das, J. H. Saltz, M. Hodoscek, and B. R. Brooks, “Parallelizing molecular dynamics programs for distributed-memory machines,” IEEE Computational Science & Engineering, vol. 2, no. 2, pp. 18-29, 1995.

[5] M. Yokokawa, F. Shoji, A. Uno, M. Kurokawa and T. Watanabe, “The K computer: Japanese next-generation supercomputer development project.” Proc. of the 17th IEEE/ACM international symposium on Low-power electronics and design, pp. 371-372, 2011.

[6] Rajovic N, Vilanova L, Villavieja C, et al., “The low power architecture approach towards exascale computing,” Journal of Computational Science, vol. 4, no. 6, pp 439-443, 2013.

[7] The Green500 list: http://www.green500.org.

[8] H. hamoto, K, Shirahata, A. Drozd, et al. Large-scale distributed sorting for GPU-based heterogeneous supercomputers, In Big Data, 2014 IEEE International Confer-

ence on. IEEE, pp. 510-518, 2014.

[9] P. Berczik, R. Spurzem, S. Zhong, L. Wang, et al., “Up to 700k GPU cores, Kepler, and the Exascale future for simulations of star clusters around black holes.” Supercomputing . Springer Berlin Heidelberg. pp. 13-25.

[10] M.C. Herboradt, T. VanCourt, Y. Gu, B. Sukhwani, A. Conti, J. Model, D. DiSabello, “Achieving High Performance with FPGA-based computing,” IEEE Computer, vol.40, no.3, pp. 50-57, 2007.

[11] H. Morishita, K. Inakagata, Y. Osana, N. Fujita, H. Amano, ”Implementation and Evaluation of an Arithmetic Pipeline on FLOPS-2D: Multi-FPGA System,” ACM SIGARCH Computer Architecture News, vol. 38, Issue 4, pp. 8-13, Sep 2010.

[12] R. Baxter, S. Booth, M. Bull, G. Gawood, J. Perry, M. Parsons, A. Simpson, A. Trew, A. McCormick, G. Smart, A. Cantle, R. Chamberlain, G. Genest. “Maxwell a 64 fpga supercomputer,” Proc. of NASA/SEA Conf. on Adaptive Hardware and Systems, pp. 287-294, Aug 2007.

[13] J. D. Davis, C. P. Thacker, C. Chang, “BEE3: Revitalizing computer, architecture research” Microsoft Research Technical Report, no. MSR-TR-2009-45, 2009.

[14] O. Mencer, K.H. Tsoi, S. Craimer, T. Todman, W. Luk, M.Y. Wong, P. Leong. “Cube: A 512-fpga cluster,” Proc. of IEEE 5th Southern Conference on Programable Logic, pp. 51-57, 2009.

[15] J. E. Vuillemin, P. Bertin, D. Roncin, M. Shand, H. H. Touati, and P. Boucard, “Programmable active memories: reconfigurable systems come of age, ” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 1, no. 4, pp. 56-69, Mar 1996.

[16] K. Sano, W. Luzhou, S. Yamamoto, “Prototype Implementation of Array-Processor Extensible over Multiple FPGAs for Scalable Stencil Computation,” ACM SIGARCH Computer Architecture News, vol. 38, no. 4, Sep 2010.

[17] A.Putnam, A. M Caulfield, E. S. Chung, et al., “A reconfigurable fabric for accelerating large-scale datacenter services,” Proc. of 2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA). pp13-24, June 2014.

[18] J. Ouyang, S. Lin, W. Qi, Y. Wang, B. Yu, S. Jiang, “SDA: Software-Defined Ac-

celerator for Large-Scale DNN Systems," Proc. of the HotChips26, Cupertino, CA, Aug 2014.

[19] M Sekine, T. Kanamaru, K. Kudoh, H. Imanaka, Y. Shiga H. Ito, Y. Myokan, "Hardware objects of the circuits for robotics," Proc. of 2003 IEEE internatiional sysmposium on Computational Intelligence in Robotics and Automation, pp. 1421-1426, 2003.

[20] K. Kudo, Y. Myokan, W.C. Than, S. Akimoto, T. Kanamaru, and M. Sekine, "Hardware object model and its application to the image processing," IEICE Trans. FUNDAMENTALS, vol.E87-A, no.3, pp.547-558, 2004.

[21] K. Kudo, H. Tamukoh, R. Sato and M. Sekine,"Half-negation pulse logic implemented on hardware/software complex system," Proc. of 5th International Symposium on Management Engineering (ISME2008), pp.397-404, Mar. 2008, Kitakyushu.

[22] K. Kudo, H. Tamukoh, T. Koga, R. Sato, M. Sekine and T. Yamakawa, "A novel vector quantization circuit employing rough-winner-take-all self-organizing neural network implemented on hardware/software complex system," Proc. of 5th International Symposium on Management Engineering (ISME2008), pp.150-155, Mar. 2008. Kitakyushu.

[23] H. Tamukoh, K. Hanai, R. Kurogi, S. Matsushita, M. Watanabe, Y. Kobayashi, and M. Sekine, "Internet Booster: A Networked Hw/Sw Complex System and Its Application to Hi-Performance WEB Application,"Proc. of World Automation Congress (WAC2010), 7th International Forum on Multimedia and Image Processing, 6 pages in CD-ROM, Sep. 2010. Kobe.

[24] Argonne National Laboratory. PETSc. http://www.mcs.anl.gov/petsc

[25] S. Kumar, C. Huang,G. Almasi, L.V. Kale, "Achieving strong scaling with NAMD on Blue Gene/L," Proc. of IEEE international parallel and distributed processing symposium, pp.25-29, Apr 2006.

[26] Naval Research Laboratory. Naval research laboratory layered ocean model (NLOM). http://www.navo.hpc.mil/Navigator/Fall99_Feature.html

[27] P. Balaji, R. Gupta, A. Vishnu, P. Beckman, "Mapping communication layouts to

network hardware characteristics on massive-scale blue gene systems," Computer Science - Research and Development, vol 26, Issue3-4, pp. 247-256, June 2011.

[28] M. Sekine, H. Tamukoh, J. Li, et al,"Brain Process: Hardware/Software Complex System Using Logic Circuits in FPGA Array Named Vocalise," Procedia Engineering, vol.50, pp.253-264, Oct 2012

[29] J.Li, K. Takahashi, H. Tamukoh, M. Sekine,"2D/3D FPGA array for brain process and numerical computation", In Proceeding(s) of IEEE 2012 8th International Conference on Natural Computation, pp.16-19, 2012.

[30] H.Takewaki,A.Nishiguchi and T.Yabe, "Cubic Interpolated Pseudoparticle Method (CIP) for Solving Hyperbolic Type Equations. J.Comput.Phys," 61 , 261-268(1985).

[31] H.Takewaki and T.Yabe : "Cubic-Interpolated Pseudo Particle (CIP) Method - Application to Nonlinear or Multi-Dimensional Problems,"J.Comput.Phys., 70 , 355-372(1987).

[32] Xilinx: Spartan-3 FPGA Family Data Sheet
DS099 June 27, 2013

[33] Xilinx: Spartan-3 Generation configuration User Guide: Extended Spartan-3A, Spartan-3E, and Spartan-3 FPGA Families
UG332(v1.6) October 26,2009

[34] Xilinx: Digital Clock Manager (DCM) Module
DS485 April 24, 2009

[35] T. Yabe, T. Aoki, "Formula are derived in the way such as to recover the CIP method,"Computer Physics Communications,(Phys.Commun). 60 (1991) 219

[36] M. Ida, T. Yabe, "Implicit CIP (Cubic-Interpolated Propagation) method in one dimension,"

[37] T. Yabe, F. Xiao, T. Utsumi, "The Constrained Interpolation Profile Method for Multiphase Analysis"

[38] T.Utsumi, T.Yabe, J.Koga, T.Aoki and M.Sekine, "Accurate Basis Set by the CIP Method for the Solutions of the Shroedinger Equation Comput," Phys.Commun. 157 pp.121-138, 2004.

[39] T. Utsumi, T.Yabe , J. Koga, T.Aoki, M. Sekine, Y.Ogata, E.Matsunaga, "A

Note on the Basis Set Approach in the Constrained Interpolation Profile Method," J.Comput.Phys. Vol 196, pp. 1-7, 2004.

[40] T.Yabe, H.Mizoe, K. Takizawa , H. Moriki, H.N. Im and Y.Ogata, "Higher-Order Schemes with CIP Method and Adaptive Soroban Grid towards Mesh-Free Scheme J," Comput. Phys. Vol.194, pp.57-77, 2004.

[41] Y. Ogata, T. Yabe and K. Odagaki, "An accurate numerical scheme for Maxwell equation with CIP-method of characteristics," Commun. Comput. Phys, Vol.1, No.2, pp. 311-335, Feb 2006.

[42] K. Shiraishi, T. Matsuoka"Wave Propagation Simulation Using the CIP Method of Characteristic Equations," Comput.Phys. Vol.3, pp.121-135, Jan 2008.

[43] T. Matsuoka,M. Matsunaga, T. Matsunaga, "Electromagnetic wave propagation analysis by using the CIP method and quadratic interpolation," Antennas and Propagation Society International Symposium, 2009. APSURSI '09. IEEE , vol., no., pp.1,4, 1-5 June 2009

[44] T. Matsuoka, "2D Wave Propagation Characteristics of the CIP Method with Amplitude Error Compensation," 2010 International Conference on Broadband, Wireless Computing, Communication and Applications (BWCCA), pp.596 - 599, Nov 2010.

[45] R. Courant, E. Isaacson and M. Rees "On the solution of nonlinear hyperbolic differential equations by finite differences," Communications on Pure and Applied Mathematics, Vol. 5, Issue. 3, pp. 243-255, Aug 1952.

[46] W. chen, P. Kosmas, M. Leeser, and C. Rappaport, "An fpga implementation of the two-dimensional finite-difference time-dimensional (FDTD) algorithm," Proc. of FPGA 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays, pp. 213-222, Feb 2004.

[47] J. P. Durbano, F. E. Ortiz, "Fpga-based acceleration of the 3D finite-difference time-domain method," Proc. of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, Vol. 1, pp. 156-163, Apr 2004.

[48] T. Kohonen, W. Chen, P. Kosmas, M. Leeser and C.rappaport, "Self-organizing Maps 3rd Edition," Springer, 2001.

[49] Adobe Corporation, [Online]. http://www.adobe.com/devnet/, 2009.

[50] J. Li, H. Tamukoh and M. Sekine, “Hardware Accelerated WEB Platform based on FPGA Array Server and Mobile FPGA Card,” Proc. of 3rd International Conference on Internet Technology and Applications (iTAP2012), Aug. 2012, Wuhan, China.

[51] T. Kohonen, W. Chen, P. Kosmas, M. Leeser and C.rappaport, “Self-organizing Maps 3rd Edition,” Springer, 2001.

[52] R. P. Martin, A. M. Vahdat, , D. E. Culler and T. E. Anderson, “Effects of communication latency, overhead, and bandwidth in a cluster architecture,” ACM, Vol. 25, No. 2, pp. 85-97.

[53] L. T. Bui, D. Essam and H. A.Abbass, “The Role of Explicit Niching and Communication Messages in Distributed Evolutionary Multi-objective Optimization,” In Parallel and Distributed Computational Intelligence, Vol. 269, 2010, pp. 181-206.

[54] B. L. Buzbee, F. W. Dorr, J. A. George, and G. H. Golub, “The Direct Solution of the Discrete Poisson Equation on Irregular Regions,” SIAM Journal on Numerical Analysis, Vol. 8, Issue. 4, pp.722-736, 1971.

[55] A. Nicholls, B. Honig“A rapid finite difference algorithm, utilizing successive over-relaxation to solve the PoissonCBoltzmann equation,” Journal of Computational Chemistry, Vol 12, Issue 4, pp. 435-445, May 1991.

[56] Xilinx Power Estimator User Guide
UG440 (v2014.1) April 23, 2014

List of Research Achievements

論文誌

1. Masatoshi Sekine, Hakaru Tamukoh, and Jiang Li, “Hardware/Software Complex System Model for Brain Process by Configurable Circuits,” International Journal on Computing, Vol. 3, No. 1, pp. 1-7, 2013. (Invited paper)
2. Jiang Li, Yusuke Atsumari, Hiromasa Kubo, Yuichi Ogishima, Satoru Yokota, Hakaru Tamukoh, Masatoshi Sekine “A Multidimensional Configurable Processor Array - Vocalise,” IEICE Trans. on Information and System, Vol.E98-D,No.2,pp, Feb. 2015.

査読付国際会議

1. Jiang Li, Kenichi Takahashi, Hakaru Tamukoh and Masatoshi Sekine, “Distributed Computing Circuits in Scalable 2D/3D FPGA Array for 2D/3D Poisson Equation Problem,” Proc. of IEEE Symposium on Low-Power and High-Speed Chips COOL Chips XV, 2 pages in CD-ROM, Apr. 2012, Yokohama. (Best Feature Award)
2. Jiang Li, Kenichi Takahashi, Hakaru Tamukoh and Masatoshi Sekine, “2D/3D FPGA Array for Brain Process and Numerical Computation,” Proc. of 8th International Conference on Natural Computation (ICNC’12), pp.16-19, May 2012, Chongqing, China.
3. Jiang Li, Hakaru Tamukoh and Masatoshi Sekine, “Hardware Accelerated WEB Platform based on FPGA Array Server and Mobile FPGA Card,” Proc. of 3rd International Conference on Internet Technology and Applications (iTAP2012),

Aug. 2012, Wuhan, China.

4. Masatoshi Sekine, Hakaru Tamukoh, Jiang Li, et al, “Brain Process: hardware/software Complex System using Logic Circuits in FPGA Array named Vocalise,” Proc. of International Conference on Advances Science and Contemporary Engineering 2012(ICASCE 2012), pp. 253-264, Oct 2012, Jakarta, Indonesia.

国内査読なし研究会論文

1. 黎 江, 佐藤一輝, 高橋 健一, 田向 権, 小林祐一, 関根優年 “ポアソン方程式における FPGA アレイ実装回路と CUDA の比較,” 第 23 回数値流体力学シンポジウム, E2-5, Dec 2009.
2. 佐藤 一輝, 黎 江, 高橋 健一, 田向 権, 小林祐一, “FPGA アレイに実装するポアソン方程式と CIP 法演算回路の性能評価,” 関根優年, 電子情報通信学会技術研究報告 回路とシステム研究会 (CAS), Vol.109 No.396 CAS2009-67 pp.19-24, Jan 2010.
3. 高橋 健一, 黎 江, 磯貝弘毅, 番場 大貴, 田向 権, 関根 優年, “FPGA アレイを用いた再構成可能な HPC システムの評価及び高位言語による回路生成,” 電子情報通信学会技術研究報告 再構成研究会 (RECONF), Vol.110 No.319 RECONF2010-39 pp.1-6, NOV 2010.
4. 高橋 健一, 黎 江, 磯貝弘毅, 田向 権, 関根優年 “高位言語回路生成ツールを用いた数値演算回路の生成及び回路実装,” 第 24 回数値流体力学シンポジウム, B2-3. Dec 2010.
5. 黎 江, 高橋 健一, 田向 権, 関根 優年, “一次元 FPGA アレイから二次元アレイに拡張した CIP 回路,” 電子情報通信学会技術研究報告 ディペンダブルコンピューティング（DC）, 信学技報, vol.110, no.413, DC2010-67, pp51-56, Feb 2011.
6. 高橋 健一, 黎 江, 集 祐介, 嶋崎 俊輔, 田向 権, 関根 優年, “3 次元 FPGA アレイ HPC システムへの数値演算回路の実装評価, ” 電子情報通信学会技術研究報告 VLSI 設計技術研究会（VLD), VLD2011-115, vol.111, no.397, pp.141-146, Jan

2012.

7. 集 祐介, 黎 江, 久保 泰正, 田向 権, 関根 優年, “3 次元 FPGA アレイ HPC システム Vocalise の性能評価,” 電子情報通信学会技術研究報告 VLSI 設計技術研究会（VLD）, VLD2012-94, vol.112, no.320, pp.201-206, Nov 2012.
8. 久保 泰正, 黎 江, 集 祐介, 小笠原 麦, 関根優年, “3 次元 FPGA アレイ Vocalise の回路構成方法,” 電子情報通信学会技術研究報告 リコンフィギャラブルシステム研究会 (RECONF), RECONF2013-32, vol.113, No.221, pp.73-78, Sep 2013.
9. 集 祐介, 黎 江, 久保 泰正, 反町 彰宏, 小笠原 麦, 関根優年,“3 次元 FPGA アレイ”Vocalise”の通信方式,”電子情報通信学会技術研究報告 リコンフィギャラブルシステム研究会 (RECONF), Vol. 113, No.418, pp.79-84, Jan 2014.
10. 黎 江, 久保 泰正, 横田 怜, 荻島 裕一, 関根優年,“数値演算における多次元 FPGA アレイ HPC system-Vocalise の実装と性能評価,” 電子情報通信学会技術研究報告 リコンフィギャラブルシステム研究会 (RECONF) , Vol. 114, No.331, pp.7-12, Nov 2014.
11. 久保 泰正, 黎 江, 横田 怜, 荻島 裕一, 関根優年,“3 次元 FPGA アレイ”Vocalise” を用いた hw/sw 複合体による移動体システム,” 電子情報通信学会技術研究報告 リコンフィギャラブルシステム研究会 (RECONF), Vol. 114, No.331, pp.19-24, Nov 2014.
12. 横田 怜, 黎 江, 久保 泰正, 荻島 裕一, 関根優年,“Vocalise による特徴領域を学習する認識システム,”電子情報通信学会技術研究報告 リコンフィギャラブルシステム研究会 (RECONF), Vol. 114, No.331, pp.25-30, Nov 2014.
13. 荻島 裕一, 黎 江, 久保 泰正, 横田 怜, 関根優年 “3 次元 FPGA アレイ”hw/sw 複合体を用いた音声認識システム,”電子情報通信学会技術研究報告 リコンフィギャラブルシステム研究会 (RECONF), Vol. 114, No.331, pp.51-56, Nov 2014.