A Multidimensional Configurable Processor Array — Vocalise

Jiang LI[a], Student Member, Yusuke ATSUMARI[b], Hiromasa KUBO[c], Yuichi OGISHIMA[d], Satoru YOKOTA[e], Nonmembers, Hakaru TAMUKOH[f], and Masatoshi SEKINE[g], Members

SUMMARY A processing system with multiple field programmable gate array (FPGA) cards is described. Each FPGA card can interconnect using six I/O (up, down, left, right, front, and back) terminals. The communication network among FPGAs is scalable according to user design. When the system operates multi-dimensional applications, transmission efficiency among FPGAs improved through user-adjusted dimensionality and network topologies for different applications. We provide a fast and flexible circuit configuration method for FPGAs of a multi-dimensional FPGA array. To demonstrate the effectiveness of the proposed method, we assess performance and power consumption of a circuit that calculated 3D Poisson equations using the finite difference method.

key words: FPGA array, scalable, hw/sw complex, poisson equation, distributed computing, HPC

1. Introduction

Field-programmable gate arrays (FPGA) are effective for high-performance computing (HPC) applications, and this solution can potentially deliver enormous performance. Special purpose processors built with FPGAs are becoming popular in supercomputing [1]–[6]. When an FPGA is configured as a special purpose circuit, it can be optimized for parallel and distributed operation for parallel applications. Although the number of usable gates in FPGA has increased up to several million, parallel applications will require even larger numbers. A scalable FPGA array structure is also interesting.

Many parallel computing systems with multi-FPGAs have been developed, such as Maxwell [7], the Berkeley Emulation Engine (BEE) [8], Cube [9], programmable active memory (PAM) [10], and the systolic computational-memory array (SCMA) [11]. 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 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 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. 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. The 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. Research shows that a parallel computing system with multi-FPGAs typically has higher cost performance and better power efficiency.

On the other hand, many HPC applications such as partial differential equation (PDE) solvers [12], climate/ocean-modeling systems [13], and molecular dynamics simulations [14] 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 [15].

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 computation. Such sharing results in significant communication contention and performance loss.

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 physical layout of application processes might not match the communication characteristics of the application. Thus, link bandwidth often needs to be doubled and redoubled because data communication among FPGAs might cause system bottleneck. Communication efficiency can be improved with a multi-dimensional processor array. For solving this problem, we propose an FPGA array designed with configurable circuits, i.e., a hardware net. We have designed a reconfigurable parallel computing platform with a multi-dimensional array of FPGAs. This custom computing system has been named the Virtual Object by Configurable Array of Little Scalable Engine (Vocalise).

The purpose of Vocalise is to study the feasibility of an application-specific multi-dimensional configuration of the FPGA array. The personal HPC can be configured to customize it for specific problems, i.e., Vocalise can be configured in a cubic form or a plane form for each specific problem, like LEGO block. To clarify the capability, we have implemented Vocalise and we examined the following features: 1) Performance of the processing element, 2) Overhead of data communications between adjacent FPGA board, 3) Power consumption and power efficiency, 4) Capability of multi-dimensional FPGA array. From our experience, though much improvement is necessary, Vocalise will be suitable for numerical calculation problems, exploration problems, and complex and power-intensive problems that require real-time execution, such as brain processes [16].

Each FPGA card has six-way 3D I/Os that enable implementation of 3-D interconnection. This enables enhancement of off-chip bandwidth. In addition, by using application specific and scalable multi-dimensional interconnection, it is easy to create network topologies of different dimension (1D, 2D, or 3D) and information. 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 for different applications.

For a 3D computing problem, a 3D FPGA array easily achieves higher transmission efficiency than a 1D or 2D arrays of FPGAs. Obviously, a 2D FPGA array is more efficient for 2D computing problem. We implemented the computation examples of the Poisson equation on 1 FPGA, 2D (2 × 2 FPGAs) and 3D (2 × 2 × 2 FPGAs) FPGA array to evaluate the feasibility of the approach.

This remainder of the paper is organized as follows. Section 2 introduces system hardware and software compositions. Section 3 demonstrates system implementation, multi-FPGA configuration, the data transfer mechanism and parallel computing implementations. Section 4 discusses and evaluates system. Section 5 presents conclusions and suggestions for future work.

2. Hardware and Software Environment

2.1 Hw/Sw Complex

The Vocalise system is based on a hardware and software (hw/sw) complex that the authors previously proposed in [17]. In Sect. 5 of the previous paper, we described the user programming environment in detail. In the case of image processing, the items discussed in that section were “implementation from software to hardware,” “cooperation of objects,” “properties from the software designer’s view,” and “properties from the hardware designer’s view.” Using the hw/sw complex design style, we can design a system that temporarily includes swObjects only at the beginning of the design process. We can then replace the swObjects with hwObjects when completing design and implementation. The hw/sw complex comprises a host PC and FPGA board called “hwModule” (Figs. 1-1 and 1-2). The host PC is a standard x86 system, and all software components are compiled using Borland C++ running on Windows operating system. The hwModule board is constructed by FPGAs and local SDRAM. The hwModule device driver is designed according to Windows Driver Model. The custom circuits, which are loaded into FPGAs, are named “hwNet”, and are handled as objects in an application. The object is referred to as a “hwObject”.

The design of Vocalise is shown in Fig. 1. The “hw-Module V2” FPGA board is a PCI device that connects extensible multi-FPGAs, i.e., an 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.

2.2 FPGA Array

The FPGA array is a core component for operating applications in the proposed system. It is scalable and uses many small FPGA boards, named “hwModule VS”. Figures 2-1 and 2-2 show that a VS board comprises a single Sub Board and a single Processing Element (PE) Board. The Sub Board is equipped with one Xilinx Spartan-3A XC3S700A FPGA. It mainly supports the implementation of special circuits onto the PE board or the next Sub Board. Moreover, Sub Board also transfers computing data between the host PC and the PE Board via 100 pin connectors. The PE Board is equipped with one Xilinx Spartan-3 XC3S4000 FPGA and 32bit-16MB SDRAM. It is primarily used to implement the arithmetic circuit for different types of applications. The PE board has six-way General Purpose Interface (GPIF) I/O ports; thus, the system is extensible to 1D, 2D, or 3D FPGA arrays. The computational mesh is homogeneously partitioned into each hwModule VS. Figure 2-3 shows a photograph of the 3D FPGA array, comprising 64 FPGAs (4 × 4 × 4). This is a promising approach that
provides bandwidth-aware structures and easy to achieve high-efficiency data communication between multi-FPGAs for multi-dimensional computational problems.

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 [7], 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.

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

The overall Vocalise system structure is shown in Fig. 1. To achieve implementation and data transmission of the multi-dimensional FPGA array, we exploit the hwModule VS for bridge circuits to connect the hwModule V2, other hwModule VSs for bridge circuits, and FPGA array. Thus, the VS, which implements bridge circuits, is named “Bridge VS” (BVS) Board (Fig. 1-3). This board is used to connect many VSs via the multi-dimensional GPIF I/O. The VS that implements the application circuit is named the “Processing VS” (PVS) Board (Fig. 1-3). The host PC can be connected to as many as 32 PVSs (4 rows × 8 VSs) through a single Bridge VS.

2.3 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 bus mode.

2.4 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 during application operations. The route of the VI Bus is variable and can
be changed to accommodate the communication requirements of different applications. Figure 3 shows VC Bus and VI Bus network connections among PVSs.

3. Vocalise System Implementation

3.1 VC Bus

3.1.1 Circuit Configuration

In the Vocalise platform, a configuration solution based on SelectMap configuration schemes is provided to minimize configuration time and maximize flexibility. The system uses a SelectMap configuration control circuit (Fig. 4) on hwModule V2 to write byte-wide configuration data to all FPGAs via VC Bus, which works in the configuration bus mode.

To configure a multi-dimensional FPGA array, the configuration bus requires an 8-bit data-line and a 14-bit control-line.

For example, the circuit configuration technique for a 3D FPGA array shown in Fig. 1 is achieved by the following steps.

1). Configure BVS’s Sub Board: The hwModule V2 configures a bridge circuit (shown in Fig. 5 a) 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 Fig. 5 b) on PE boards of BVS in parallel with its Sub Board.

3). Configure PVS’s Sub 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 Boards in sequence through the BVS. There are four links from each BVS; thus, a BVS can configure 32 PVS’s Sub Boards (4rows × 8FPGAs).

4). Configure PVS’s PE Board: When the above configuration steps are completed; 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.

3.1.2 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.

Data communication modules are configured to hw-Module V2 and each PVS PE board (Fig. 6). The modules are based on a typical master-slave transmission mechanism and can achieve 32-bit burst transmission. The bus line is equipped with a 32-bit A/D line, a 6-bit control line, and a 1-bit clock line.
Fig. 6 VC Bus data communication elements.

Fig. 7 PVS address signal format in VC Bus network.

Fig. 8 The write operation between host PC and FPGA array via VC Bus.

Table 1 Essential VI Bus signals for telecommunication.

<table>
<thead>
<tr>
<th>Telecommunication Mode</th>
<th>Simplex</th>
<th>Half-duplex</th>
<th>Full-duplex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data line (bit)</td>
<td>32</td>
<td>32</td>
<td>Rx(16) + Tx(16)</td>
</tr>
<tr>
<td>control line (bit)</td>
<td>2</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>clock line (bit)</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>total (bit)</td>
<td>35</td>
<td>40</td>
<td>42</td>
</tr>
</tbody>
</table>

Fig. 9 Simplex transmission operation between adjacent FPGAs via VI Bus.

hwNtMgr unit is an hwNet controller; it is used to store commands from the host PC and the status of hwNets. The applications on the host PC send specific commands to any PVS and read the status of each PVS through the implemented hwNtMgr.

Each PVS has a unique ID in the VC Bus network. The PVS address signal format is shown in Fig. 7. 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 achieved with bridge circuits implemented on the Bridge VSs.

Figure 8 shows write operation signals between the hwModule and the FPGA array. When the 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.

3.2 VI Bus

3.2.1 Data Communication among Process VSs

In our system, Process VSs communicate with each other via the VI Bus. The VI Bus network topology can be changed as the requirements of applications come in. To reduce the circuit design development cycle, we have designed a foundational communication module, the VI Bus connector Module, to establish direct pin-to-pin connection to adjacent FPGAs in each six-way direction. The modules are implemented in each processing 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 1 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 (133 MHz × 32 bits). Figure 9 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 way × 4.26 Gbps) among interconnected FPGAs.

3.3 Implementation of Applications

The hwNet accesses local memory (32 bit-16 MB SDRAM)
describe an experimental implementation of a 3D Poisson

A Poisson equation is an elliptic PDE that has broad

Electrostatics

One of the principle cornerstones of electrostatics is the formulation and resolution of problems described by Poisson equation. Equation (1) is Poisson equation applicable to electrostatics.

$$\nabla^2 \phi = -\rho / \epsilon_0$$  \hspace{1cm} (1)

Here, $\nabla^2$ is a Laplace operator, $\rho$ is charge density, $\phi$ is electric potential, and $\epsilon_0$ is the vacuum permittivity. In a case of 3D space, central-difference methods with second order accuracy give the approximations for the 3D collocated grids in Fig. 11.

With approximate operations, a 3D Poisson equation can be expressed in following common form.

$$\phi_{i,j,k}^{new} = -h^2 \rho + (\phi_{i,j,k-1}^{old} + \phi_{i,j,k}^{old} + \phi_{i,j,k+1}^{old} + \phi_{i,j+1,k}^{old} + \phi_{i,j-1,k}^{old} + \phi_{i,j,k+1}^{old})/6.$$  \hspace{1cm} (2)

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. (2) all grid points only require the accumulation computations 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. (2) to exploit these locality and parallelism properties. The PEs are the core components of the implemented application circuits.

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

$$\phi_{i,j,k}^{new} = (\phi_{i-1,j,k}^{old} + \phi_{i+1,j,k}^{old} + \phi_{i,j-1,k}^{old} + \phi_{i,j+1,k}^{old} + \phi_{i,j,k+1}^{old} + \phi_{i,j,k-1}^{old} + 2\phi_{i,j,k}^{new} - 6h^2 \rho)/8.$$  \hspace{1cm} (3)

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

$$\phi_{i,j,k}^{new} = (\phi_{i-1,j,k}^{old} + \phi_{i+1,j,k}^{old} + \phi_{i,j-1,k}^{old} + \phi_{i,j+1,k}^{old} + \phi_{i,j,k+1}^{old} + \phi_{i,j,k-1}^{old} + \delta_{i,j,k} - 6h^2 \rho)/8.$$  \hspace{1cm} (4)

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 Eq. (2) to Eq. (4) easily to simplify an arithmetic circuit design.

3.3.2 Architecture of hwNet

We developed a hwNet to solve 3D Poisson problems. Figure 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 Fig. 13. The PE contains seven adders, one divider and one multiplier to operate one grid-point with Eq. (3) for single-precision floating-point numbers that comply with IEEE754, the 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 3D Poisson equation is 41 clk, 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 XC3S4000 FPGA (shown in Table 2), 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 Fig. 13. Each PE operates sub-grids which are distributed on a plane. Therefore, the eight PEs are able to process eight sub-grids 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 66MHz 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 DMA module via the FPGA Inner Bus (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 BRAM and LMController 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.

3.3.3 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 grids. 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 VIBus 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 Fig. 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 PVSes 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/0s. 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.

By considering the synchronous operation of a large scale multi-dimensional 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.

4. Results and Discussion

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. Therefore, 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 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.

Table 3  Circuit configuration execution time of multi-FPGA.

<table>
<thead>
<tr>
<th>FPGA Number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub Board</td>
<td>0.051s</td>
<td>0.101s</td>
<td>0.152s</td>
<td>0.203s</td>
</tr>
<tr>
<td>PE Board</td>
<td>0.201s</td>
<td>0.201s</td>
<td>0.201s</td>
<td>0.202s</td>
</tr>
</tbody>
</table>

Network A

<table>
<thead>
<tr>
<th>FPGA</th>
<th>FPGA</th>
<th>FPGA</th>
<th>FPGA</th>
</tr>
</thead>
</table>

Network B

<table>
<thead>
<tr>
<th>FPGA</th>
<th>FPGA</th>
<th>FPGA</th>
<th>FPGA</th>
</tr>
</thead>
</table>

Network C

<table>
<thead>
<tr>
<th>FPGA</th>
<th>FPGA</th>
<th>FPGA</th>
<th>FPGA</th>
</tr>
</thead>
</table>

Fig. 15  Different types of VI Bus network topology.
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 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 [16]. These brain process circuits such as voice recognition, voice synthesis, and image recognition were designed as hw/sw complex systems using the Vocalise method.

4.3 Data Communication Efficiency in Multi-Dimensional Interconnection

In this section, 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, Fig. 16, shows a 3D computation mesh is decomposed to many sub-grids, each sub-grid 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 \( 4 \times 2 \times 2 \) 3D 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 implement 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.

4.4 Performance Evaluation

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 \( 10 \times 10 \times 8 \) grids at 66 MHz. In comparison, six PEs on 1PVS to compute cubical space of \( 10 \times 10 \times 6 \) grids at 66 MHz is shown. The floating-point performance of a PVS can be calculated to measure the execution time of different iterations. Table 4 shows the floating-point performance for the 3D Poisson equation with six PEs and eight PEs. As the number of iterations increased, the final performance (4.72 GFlops) approached the peak performance of 1 PVS, which was 4.79 GFlops at 66 MHz.

We designed a test vehicle with six PEs and six-way data communication circuits in each PVS for multi-dimensional connection FPGAs. The six PEs execute the operations on the \( 10 \times 10 \times 6 \) sub-grids for which the data communications of \( 10 \times 10 \) and \( 10 \times 6 \) planes are required. The execution and data communication are also processed at 66 MHz. The synchronous data transmission among FP-
Table 5 Execution time and data communication among 3D connection for 3D Poisson equation (Ex Time = Execution Time).

<table>
<thead>
<tr>
<th>Transmission direction</th>
<th>X-axis way</th>
<th>Y-axis way</th>
<th>Z-axis way</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(Front→Back)</td>
<td>(Left→Right)</td>
<td>(Up→Down)</td>
</tr>
<tr>
<td>Iterations</td>
<td>$10^6$</td>
<td>$10^9$</td>
<td>$10^6$</td>
</tr>
<tr>
<td>Exchange Data Plane</td>
<td>$10 \times 6$</td>
<td>$10 \times 6$</td>
<td>$10 \times 10$</td>
</tr>
<tr>
<td>Ex Time with Communication [s]</td>
<td>26.73</td>
<td>26.52</td>
<td>35.34</td>
</tr>
<tr>
<td>Ex Time without Communication [s]</td>
<td>16.43</td>
<td>16.35</td>
<td>16.9</td>
</tr>
<tr>
<td>Communication Overhead [s]</td>
<td>10.03</td>
<td>10.17</td>
<td>18.44</td>
</tr>
</tbody>
</table>

Table 6 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).

<table>
<thead>
<tr>
<th>Communication Available</th>
<th>2 × 2 PVSs</th>
<th>2 × 2 × 2 PVSs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>$10^6$</td>
<td>$10^8$</td>
</tr>
<tr>
<td>Execution time [s]</td>
<td>165.34</td>
<td>264.06</td>
</tr>
<tr>
<td>Performance [GFlops]</td>
<td>13.05</td>
<td>8.18</td>
</tr>
</tbody>
</table>

GAs is described in Sect. 3.3.3. 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 = 10.3 sec, 10.17 sec for the Y-axis, and 18.44 sec 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 5 shows the overhead required for data communication between adjacent PVSs, where the 3D Poisson equation with 2 × 2 PVSs was evaluated. We implemented 2 × 2 PVSs, i.e., 24 (2 × 2 × 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 6 shows the communication 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 × (10 × 10)/(10 × 6) = 164.53 sec. 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 × 6 = 48 PEs) and evaluated the communication overhead. The results are shown in Table 5. The 2 × 2 × 2 PVSs performed 25.63 GFlops without communication and 12.46 GFlops with communication. The communication overhead time among 2 × 2 × 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 results, the performance of 4 × 4 × 8 FPGAs can be estimated; 128 FPGAs implemented at 4 × 4 × 8 × 6 = 768 PEs with communications can achieve 199.36 GFlops.

The FPGA array is powered by a 12V DC power supply. We acquired the power consumption by measuring the current value. The power consumption of a single PVS was only 3.36 W, and the idle power of the PVS was 2.11 W, which exceeds 50% of its load power. The power efficiency was 1.40 GFlops/W when eight PEs are implemented on one PVS. The power consumption of FPGA is one-tenth than a general-processor so that without additional cooling devices design for the system.

However, the results do not show superior performance compared to near-term high-end processors. 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. 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[18], [19].

5. Conclusions

Vocalise is a scalable multi-dimensional interconnection FPGA array computing platform. We designed specific circuits for a 3D Poisson equation. The results of the application specific circuits on Vocalise showed operational efficiency, scalability, communication overhead and power consumption. The PVS for which eight PEs were implemented achieved 4.72 GFlops for the 3D Poisson equation. The 2D FPGA array (2 × 2 PVSs), which used 2 × 2 × 6 PEs with communication, achieved 8.18 GFlops. The 3D FPGA array (2 × 2 × 2 PVSs) implemented with 2 × 2 × 6 PEs performed 12.46 GFlops. We estimate that if a 4 × 4 × 8 FPGA array is implemented with 4 × 4 × 8 × 6 = 768 PEs, then the performance will achieve 199.36 GFlops while consuming 435W of power.

We developed the parallel circuit configuration solution for a large-scale multi-dimensional 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.

The system can implement different network topologies using the multi-dimensional direct interconnection, and this scalability is critical to improve communication.
performance. The network can be configured to provide application-specific data communication for each application. As the more FPGA chips have been required, a lower price has become important. Compared with a multi-core general processor system approach, the FPGA array is a cost-effective and power-effective method. In the future, we will use the multi-dimensional FPGA array to a broad spectrum of applications such as other PDE problems or massively-parallel brain processes, to enhance or accelerate various scientific simulations and real-time brain processes. It is our expectation that many implementations of an application-specific personal computer including scalable multi-dimensional FPGA arrays, will be realized.

References


Jiang Li received his B.S degree in Electrical and Information Engineering from North-east Forestry University, China, in 2006. Then, he received the M.S degree in Electrical and Electronic Engineering from Tokyo University of Agriculture and Technology, Japan, in 2011. Currently, he is a Ph.D. student at the Department of Electronic and Information Engineering of Tokyo University of Agriculture and Technology. His research interests are reconfigurable system, parallel processing.

Yusuke Atsumari received a B.S degree in Electrical and Electronic Engineering from Tokyo University of Agriculture and Technology, Japan, in 2012. Currently, he is a M.S student at the Department of Electronic and Information Engineering of Tokyo University of Agriculture and Technology. His research interests are reconfigurable system, communication system, and mobile intelligent robot.

Hiromasa Kubo received a B.S degree in Electrical and Electronic Engineering from Tokyo University of Agriculture and Technology, Japan, in 2013. Currently, he is a M.S student at the Department of Electronic and Information Engineering of Tokyo University of Agriculture and Technology. His research interests are FPGA-Array configuration method and reconfigurable robot.
Yuichi Ogishima received a B.S degree in Electrical and Electronic Engineering from Tokyo University of Agriculture and Technology, Japan, in 2013. Currently, he is a M.S student at the Department of Electronic and Information Engineering of Tokyo University of Agriculture and Technology. His research interests are voice recognition systems and reconfigurable robots.

Satoru Yokota received a B.S degree in Electrical and Electronic Engineering from Tokyo University of Agriculture and Technology, Japan, in 2013. Currently, he is a M.S student at the Department of Electronic and Information Engineering of Tokyo University of Agriculture and Technology. His research interests are image recognition systems and reconfigurable robots.

Hakaru Tamukoh received a B.S. degree in Engineering from University of Miyazaki in 2001, the M.S. degree from Institute of Technology in 2003, and the Ph.D. degree from Kyushu Institute of Technology in 2006. Since 2006 till 2007, he was a Postdoctoral Research Fellow in Kyushu Institute of Technology. He has been engaged in Tokyo University of Agriculture and Technology as a Professor from 2007 to 2013. In 2013 he joined Kyushu Institute of Technology as an Associate Professor in Brain Science and Engineering. His research interests include LSI design, soft computing, image and video processing, and autonomous robotics.

Masatoshi Sekine received a B.S. degree in physics from Tokyo Institute Technology in 1974, the M.S. degree from University of Tokyo in 1976, and the Ph.D. degree from the University of Tokyo in 1998. Since 1976 till 1999, he has been engaged in Toshiba Research and Development Center. He was a visiting researcher of UC Berkley from 1991 to 1993. In 1999 he joined Tokyo University of Agriculture and Technology as a Professor in Electrical and Electronics Engineering. His research interests include reconfigurable systems, circuits for robotics, brain computers, pattern recognition with wavelet transforms, and artificial intelligence.