Reducing Preprocessing Overhead Times in a Reconfigurable Accelerator of Finite Difference Applications

Kataoka, Hiroshi
Department of Informatics, Kyushu University

Honda, Hiroaki
Institute of Systems, Information Technologies and Nanotechnologies, Kyushu University

Mehdpour, Farhad
Department of Informatics, Kyushu University

Inoue, Koji
Department of Informatics, Kyushu University

他

Reducing Preprocessing Overhead Times in a Reconfigurable Accelerator of Finite Difference Applications

Hiroshi KATAOKA*, Hiroaki HONDA†, Farhad MEHDIPOUR*, Koji INOUE*, and Kazuaki MURAKAMI*†
*Department of Informatics, Kyushu University
†Institute of Systems, Information Technologies and Nanotechnologies
Email: kataoka@soc.ait.kyushu-u.ac.jp

***Abstract*— Hardware accelerators integrating to general purpose processors (GPPs) are increasingly employed to achieve lower power consumption and higher processing speed. However due to impact of memory-wall problem, this kind of acceleration does not always achieve a demanded performance. To resolve this issue, a Large-Scale Reconfigurable Data-Path (LSRDP) has been proposed which is able to reduce the required memory bandwidth. Since the LSRDP consists of a large number of Processing Elements (PEs), it can potentially achieve a very high performance. To take advantages of the LSRDP architecture, a proper implementation for the target application is essential requirement. In this paper, three ways for implementing applications are introduced which include primitive implementation, software optimized version and software optimized with additional memory access controller hardware to decrease the amount of redundant data transfer. Our experimental results reveal about 100 smaller execution time on the LSRDP compared with GPP when the proposed optimization ideas are developed.**

I. Introduction

In various scientific areas such as fluid dynamics, computational chemistry, materials science, environmental issues and etc., complex numerical computations are indispensable which necessitate employing quite powerful computers. Providing high computational power to individual researchers is crucial for progress of the research and development. Generally, large scale parallel cluster computers with GPPs are often utilized as supercomputer systems. On the other hand, the hybrid architecture comprising an accelerator augmented to a GPP might be chosen for special purpose computations. The accelerator should be designed to feature small size, high performance, and low power consumption.

Recent examples of such accelerator are CSX600 PCI-X board [1], GRAPE-DR processor [2], Cell processor [3] which is heterogeneous multi-core processor and General Purpose computing on Graphic Processing Unit (GPGPU) calculations are often used by graphic accelerator chips [4]. Those accelerators commonly have Single Instruction Multiple Data stream (SIMD) mechanism for total architecture, or part of units. Generally, a large memory bandwidth is demanded in conventional accelerators to perform calculations efficiently. Therefore, an on-chip memory can be utilized for reduction of the required memory bandwidth. Recently, the LSRDP processor has been proposed [5] as shown in Fig. 1. The LSRDP architecture is a pipelined architecture comprising a two dimensional array of PEs for floating point operations and interconnection networks among PEs referred as Operand Routing Networks (ORNs). I/O data are directly transferred from/to main memory through Streaming Memory Access Controller (SMAC). The main intuition behind the LSRDP is that the cascaded PEs can generate a final result without temporally memorizing intermediate data, therefore, the number of memory load/store operations corresponding to spill codes can be reduced. Moreover, for high-throughput computation, it is quite important to utilize efficient Direct Memory Access (DMA) data transfer mechanism between LSRDP and main memory.

On the other hand, high-density packing of a large number of PEs and computing with high-throughput, will result in high electric power consumption and high heat radiation. To avoid these issues, the LSRDP processor is implemented by single-flux quantum (SFQ) circuits which brings about very smaller energy consumption [5]. A SFQ circuit is based on the superconductor technology which includes low power consumption and high speed switching compared to the CMOS circuits. It uses a $\sim$1 mV extremely low-width pulse as an information carrier that is propagated up to light speed in the circuit. High-speed switching, signal transmission, low power consumption, compact implementation (small area), and suitability for pipeline processing of data stream are the main features of the SFQ technology [6].

An architecture including appropriate number of PEs and ORNs has been designed through a quantitative design approach targeting a number of scientific applications [7]. The ORN architecture including appropriate number of cross-bar switches has been introduced in [8] as well.

In this paper, we implement two finite difference applications based on heat and vibration partial differential equations as the target applications for LSRDP. Optimizing software algorithm and SMAC architecture for utilizing DMA transfer mechanism, we have prepared three types of implementations for both above applications: (1) primitive implementation, (2) software optimized version, and (3) software+hardware optimized version. Next, those programs are executed on a simulator based on simple analytical analysis and evaluated for their performance.

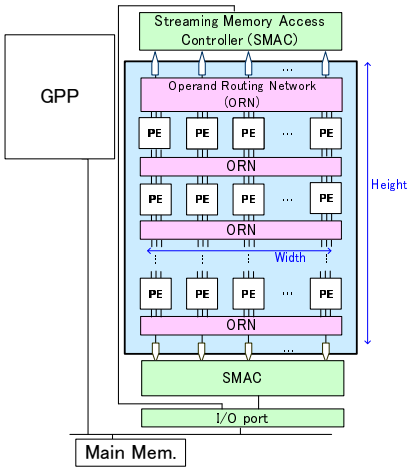

Fig. 1. Overall architecture of the LSRDP computing system

II. Implementation of A Target Applications

As mentioned in previous section, for utilizing efficient data transfer via DMA, software and hardware implementations are optimized to avoid random accesses and redundant data transfers from/to the main memory.

We start with example of GPP program shown in Fig. 2. Here, C, D, and E are constants. This program is simply converted to one fitting the LSRDP as Fig. 3. Inner loop i in Fig. 2 can be executed on the LSRDP in a pipeline fashion. A[n,i-1], A[n,i] and A[n,i+1] elements on the main memory are read simultaneously for inner-loop calculation. Therefore, elements of array A are rearranged as A[n,0],A[n,1],A[n,2],A[n,1],A[n,2],A[n,3],$\cdots$ in the main memory via the "Rearrange A[ ]" procedure. This is referred as the **BASE** implementation.

```
Loop n:
  Loop i:
    A[n+1, i+1] = C*A[n, i-1]+D*A[n, i]
                              +E*A[n, i+1];
  End
End
```

Fig. 2. Example of GPP original program

```
Loop n:
  Rearrange A[ ];
  Run LSRDP (pipeline execution);
    (computing above inner loop)
End
```

Fig. 3. LSRDP program converted from GPP original program

A. Software Optimization

Basically, the data rearrangement can be performed dynamically within the program execution as above example. However, if original order of elements of array A are modified as the access order of calculation and the program is implemented based on the modified array, explicit data rearrangement procedure can be eliminated. This is a software modified version which is referred as **REARR**.

B. Hardware optimization

During two consecutive cycles of the LSRDP's pipeline execution, some overlapping data is observed. For instance, in the above code (Fig. 2), A[n,1] and A[n,2] are used in the first and second iterations, whereas (A[n,0], A[n,1], A[n,2]) and (A[n,1], A[n,2], A[n,3]) are the input data within the first and second iterations, respectively. By utilizing additional hardware as the SMAC (shown in Fig. 4), such redundant data transfers can be eliminated. Input data which are transferred from the main memory, slide over the buffers in each clock cycle as FIFO operation. At given interval cycles, controller sends a signal to registers which are shifting data to the neighboring registers during the execution. After receiving the signal, each register fetches datum from the corresponding buffer pair and sends it to the LSRDP's input port. By this procedure, replicated data in consecutive inputs can be generated without any extra data transfer from the main memory. This hardware optimization can be utilized along with previous software optimization **REARR**. We call this software + hardware optimized version as **REARR+MEMHW**.

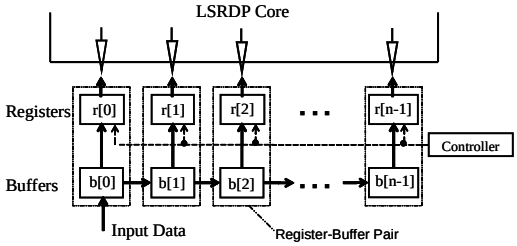

Fig. 4. Additional hardware as memory access controller for eliminating redundant data transfer in finite difference applications

III. Heat and Vibration Applications

Generally, 2nd order partial differential equations with constant coefficients are categorized to three types: heat or diffusion equation (Heat), vibration equation (Vib) and Poisson equation. Here, we consider 1-dimensional Heat and Vib applications. First, these two applications are converted from original partial differential equation form to basic finite difference formula.

In the next stage, DFGs are manually extracted from these basic finite different equations, then mapped onto the LSRDP by utilizing a mapping tool introduced in [7]. However, it is inefficient to map only small DFGs which are extracted directly from basic equations. Therefore larger DFGs are generated through merging the smaller ones.

For the implementation of total application, we prepared three types of implementations for each of two above applications: **BASE**, **REARR**, and **REARR+MEMHW** explained in previous section.

IV. Performance Evaluations for Heat and Vibration Applications

To evaluate performance of the proposed implementations of Heat and Vib applications on the LSRDP computer, an evaluation was accomplished based on simple analytical analysis and simulation. The base and target reconfigurable processors with characteristics displayed in Table I were employed.

Two applications including Heat and Vib equations were examined. Fig. 5 shows performance on the GPP+LSRDP for those applications, respectively. Vertical axis denotes the normalized execution time (ratio of execution time on GPP+LSRDP to the execution time on GPP). For the Heat and Vib, **BASE** calculations are about 2.8 and 1.2 times faster than original GPP calculations, respectively.

In Fig. 5, a breakdown of the execution time can be seen. For both applications in the **BASE** bar, the largest portion of execution time (referred as 'Trearr') is the time required for data rearrangement. Input/output data should be arranged in a proper order for the next execution step on the LSRDP. The second major fraction ('Tst') is the time elapsed for transferring data from main memory to the LSRDP and the vice versa. Remaining time fractions including the LSRDP execution time ('Tcal'), the LSRDP reconfiguration time ('Trec'), signal transfer time between GPP and LSRDP ('Ttra'), and the GPP execution time ('Tgpp') are almost negligible.

In the **BASE** calculation for Vib, a large portion of time comes from data rearrangement time. Therefore, by utilizing other two implementations the rearrangement time is substantially alleviated and total execution time is drastically decreased. Especially, for **REARR+MEMHW** implementation, the execution time on the LSRDP is about 1% of that time on the GPP for both Heat and Vib applications, and very high-performance values, namely 210.0 GFLOPS and 104.9 GFLOPS are achievable, respectively.

TABLE I
Configuration of the base and reconfigurable processors

GPP	Processor type	Out-of-order
	GPP operating frequency	3.2GHz
	Inst. issue width	4 instruction/cc
	Inst. decode width	4 instruction/cc
	L1 data cache	64KB(128B Entry, 2way, 2cc)
	L1 instruction cache	64KB(64B Entry, 1way, 1cc)
	L2 unified cache	4MB(128B Entry, 4way, 16cc)
	Latency of main mem.	300 CC
	L2 - main memory bus width	64 Bytes
	L2 - main memory freq.	800 MHz
LSRDP	LSRDP operating frequency	80 GHz
	Reconfiguration Latency	1cc
	Mem. Bandwidth	102.4GB/sec
	No. of PEs in a row	48
	No. of PE rows	35

V. Conclusion

A high-performance computer comprising an accelerator named Large-Scale Reconfigurable Data-Path with 2-dimensional floating point array architecture implemented by superconducting circuits was introduced. Three different types of implementations of heat and vibration finite difference applications were proposed and performance evaluation results

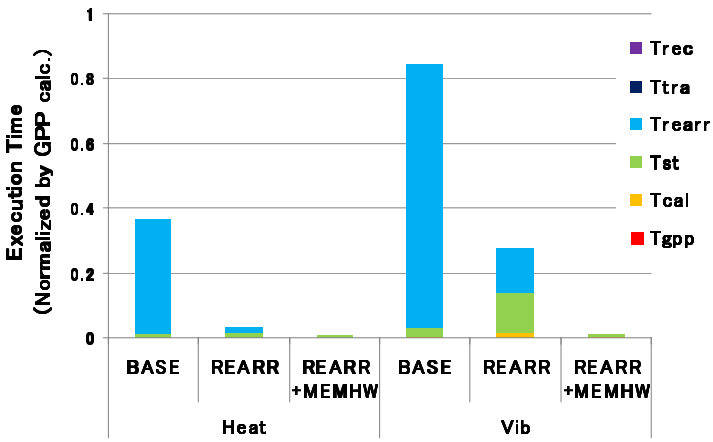

Fig. 5. Execution times normalized by GPP executions

were reported. These implementations of applications include primitive implementation, software optimized version and software optimized with additional memory access controller to decrease the amount of data transfer. The results show that procedure of dynamical I/O data rearrangement and the stall time caused by redundant data transfers to the LSRDP consume the largest fraction of execution time. Finally, by eliminating dynamical data rearrangement and redundant data transfers using a memory access controller, we archived about 100 smaller execution time than original general purpose processor for both heat and vibration applications.

Acknowledgment

This research was supported in part by Core Research for Evolutional Science and Technology (CREST) of Japan Science and Technology Corporation (JST).

References

[1] ClearSpeed processor. [Online]. Available: http://www.clearspeed.com/
[2] J. Makino, K. Hiraki, and M. Inaba, "GRAPE-DR: 2-Pflops massively-parallel computer with 512-core, 512-Gflops processor chips for scientific computing," in *SC07*. American Chemical Society, November 2007.
[3] Cell Broadband engine. [Online]. Available: http://cell.scei.co.jp/index.html
[4] J. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krger, A. Lefohn, and T.Purcell, "A survey of general-purpose computation on graphics hardware," in *Computer Graphics Forum*, March 2007, pp. 80–113.
[5] N. Takagi, K. Murakami, A. Fujimaki, N. Yoshikawa, K. Inoue, and H. Honda, "Proposal of a desk-side supercomputer with reconfigurable data-paths using rapid single flux quantum circuits," *IEICE Trans. on Elec.*, vol. E91-C(3), pp. 350–355, 2008.
[6] K. Likharev and V. Semenov, "RSFQ logic/memory family: a new Josephson junction technology for sub-teraherz clock frequency digital systems," *IEEE Trans. Appl. Supercond.*, vol. 1, no. 1, pp. 3–28, 1991.
[7] F. Mehdipour, H. Honda, K. Kataoka, K. Inoue, I. Kataeva, K. Murakami, H. Akaike, and A. Fujimaki, "Mapping scientific applications on a Large-Scale Data-Path accelerator implemented by Single-Flux Quantum (SFQ) circuits," in *Proceedings of the 13th Design, Automation and Test in Europe (DATE10)*, March 2010, pp. 993–996.
[8] I. Kataeva, H. Akaike, A. Fujimaki, N. Yoshikawa, N. Takagi, and K. Murakami, "An operand routing network for an SFQ reconfigurable data-path processor," *IEEE Trans. Appl. Supercond.*, vol. 19, no. 3, pp. 665–669, 2009.