Low-Power Heterogeneous Platform for High Performance Computing and Its Application to 2D-FDTD Computation

ERSA'12 Short Paper

Hasitha Muthumala Waidyasooriya, Yasuhiro Takei, Masanori Hariyama and Michitaka Kameyama
Graduate School of Information Sciences, Tohoku University
Aoba 6-6-05, Aramaki, Aoba, Sendai, Miyagi, 980-8579, Japan
Email: {hasitha, takei, hariyama, kameyama}@ecei.tohoku.ac.jp

Abstract—Heterogeneous processing with CPUs and low-power accelerators attract many attentions since they can achieve power-efficient computing. However, the potential of using low-power accelerators such as FPGAs in high-performance computing to reduce the power consumption is rarely exploited. In this paper, we propose a CPU/FPGA heterogeneous platform to implement the finite-difference time-domain (FDTD) computation. According to the experimental results, we found that 95% of the computation can be done in FPGA using 32bit fixed point arithmetic without suffering a major precision loss. According to our estimation, we can achieve the same performance of CPU/GPU computing with 10 times less power consumption by using the proposed low-power heterogeneous platform.

Keywords: Heterogeneous processing, high-performance computing, supercomputing, FDTD

1. Introduction

Applications used in low-power embedded processing to high performance computing have different tasks such as data-intensive tasks and control-intensive tasks. Heterogeneous processing is proposed to execute such different applications power-efficiently. In heterogeneous processing, different processors such as CPUs and accelerators are used as shown in Fig.1. If the tasks of an application are correctly allocated to the most suitable processors, all the processors work together to increase the overall performance. An example of a heterogeneous high-performance computing platform (HPC) is “Tianhe-1A” [1] which has Intel X5670 CPUs and NVIDIA GPUs (graphical processing unit).

The main problem in commercially available HPC platforms such as [1] is the large power consumption. High-end CPUs and GPUs have a power consumption of more than 100W and 300W respectively. A very promising solution to reduce the power consumption of HPC platforms is to use FPGAs. Recent FPGAs have over 300,000 logic cells, over 5MBits of memory and high-speed data transfer interfaces. Therefore, a single FPGA is large enough to hold hundreds of processing elements (PEs). Although FPGAs have a lot of resources and consume a very small power, they are rarely used in the HPC platforms. One main reason for not using FPGAs is their high programming complexity. The other main reason is the weak floating-point performance in FPGAs compared to that in GPUs.

In this paper, we propose a low-power heterogeneous platform with CPUs and FPGA accelerators. We propose a CUDA-like (Compute Unified Device Architecture) SIMD-2D (Single Instruction, Multiple Data-2 dimensional PE array) architecture to reduce the programming complexity. The basic idea of this architecture and its programming environment are already proposed in [2]. In this paper, we develop the basic SIMD-2D architecture proposed in [2] to be applicable for HPC applications. We use the finite-difference time-domain (FDTD) algorithm for electromagnetic field computation as an example. According to the experimental results with FDTD computation, we found that 95% of the computation can be done by fixed-point arithmetic without a major precision loss. However, for the rest of the calculations, we must use double-precision floating-point arithmetic. Therefore, we can perform floating-point arithmetic in CPUs and fixed-point arithmetic in FPGAs. The proposed heterogeneous environment produces almost the same performance of GPU-based processing at a 10 times smaller power consumption.
2. Heterogeneous Platform for 2D-FDTD computation

2.1 FDTD-2D computation

FDTD [3] algorithm is one of the most popular methods of computational electromagnetic simulation due to its simplicity and very high computational efficiency. It has been already implemented successfully in multicore CPUs. There are many recent works such as [4] and [5] that use GPUs to accelerate the FDTD algorithm.

Figure 2 shows the main tasks of the FDTD algorithm. It starts with the initial values of the electric and magnetic fields. Then the initial data are processed to obtain the electric field information for the first time-step. After that, the periodic boundary conditions are applied. Then the magnetic field information are obtained and the periodic boundary conditions are applied. The process continues for a given number of time-steps. Equation (1) shows the electric field computation and Eqs. (2) and (3) show the magnetic field computation. Electric and magnetic fields in \( x, y, z \) directions are denoted by \( E \) and \( H \) respectively. The time-step is denoted by \( n \) and the coordinates of the 2D fields are denoted by \( i \) and \( j \). Note that, the boundaries of the electric and magnetic fields are calculated differently. A detailed description of the FDTD algorithm is given in [3]

\[
E_z^{n+1}(i, j) = E_z^n(i, j)
- P_y(i, j) \left\{ H_x^{n+\frac{1}{2}}(i, j + 1/2) - H_x^{n-\frac{1}{2}}(i, j - 1/2) \right\}
+ P_x(i, j) \left\{ H_y^{n+\frac{1}{2}}(i + 1/2, j) - H_y^{n-\frac{1}{2}}(i - 1/2, j) \right\}
\]  

\[
H_x^{n+\frac{1}{2}}(i, j + 1/2) = H_x^{n-\frac{1}{2}}(i, j + 1/2)
-Q_y(i, j) \{ E_z^n(i, j + 1) - E_z^n(i, j) \}
\]  

\[
H_y^{n+\frac{1}{2}}(i + 1/2, j) = H_y^{n-\frac{1}{2}}(i + 1/2, j)
-Q_x(i, j) \{ E_z^n(i + 1, j) - E_z^n(i, j) \}
\]  

Table 1: Precision vs. computation error

<table>
<thead>
<tr>
<th>Method</th>
<th>Maximum absolute error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Electric field</td>
</tr>
<tr>
<td>1</td>
<td>Double-precision (DP) floating point</td>
</tr>
<tr>
<td>2</td>
<td>Single-precision (DP) floating point</td>
</tr>
<tr>
<td>3</td>
<td>5% DP floating point 95% 32bit fixed point</td>
</tr>
<tr>
<td>4</td>
<td>25% DP floating point 75% 32bit fixed point</td>
</tr>
</tbody>
</table>

Table 2: Specifications of the heterogeneous platform

<table>
<thead>
<tr>
<th>Processor</th>
<th>Number of cores</th>
<th>Maximum power</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU: Intel core i7 3960X</td>
<td>6</td>
<td>130W</td>
</tr>
<tr>
<td>GPU: GeForce GTX5900</td>
<td>1024</td>
<td>360W</td>
</tr>
</tbody>
</table>

Figure 2: Flowchart of the FDTD computation
FPGA. To reduce the programming complexity, we propose a CUDA-like architecture for the FPGA. Figure 4 shows the proposed FPGA architecture. It consists of a Nios II CPU core, on-chip memory and SIMD accelerator cores. An external DDR2 SDRAM is connected to the CPU core through the FPGA board. The Nios II CPU core is mainly used as the control unit of the accelerators. The proposed SIMD accelerator is designed similar to the GPU accelerator so that we can use the same CUDA code. The basic idea of the SIMD accelerator and its programming environment are discussed in the previous work [2]. The major difference of the architecture in [2] and this paper is the structure of the PE.

Figure 5 shows the architecture of a 32bit fixed-point PE. It consists of 5 adders and 3 multipliers. The data path is fully pipelined so that an output is produced in every clock cycle after the pipeline is filled. We can have 191 such PEs in a single “Stratix IV GX EP4SGX530” FPGA. It is possible to increase the number of PEs by using a much powerful FPGA.

In the proposed platform, the data close to the boundaries of the electric and magnetic fields are processed in the CPU (Intel core i7-3960X) and the rest of the data are processed in the FPGA. After the boundary data are processed, they are transferred to the FPGA. We can hide this CPU processing overhead by overlapping the boundary data processing in the CPU with the electric and magnetic field computation in the FPGA.

### 3. Evaluation

We estimated the resource usage, power and frequency of the proposed architecture for Stratix IV GX EP4SGX530 FPGA in the Altera DE4 [6] board. According to the estimation shown in Table 3, we can implement up to 191 PEs in the FPGA at 50MHz frequency. The power consumption is less than 20W.

Table 4 shows the comparison with the CPU/GPU heterogeneous platform. The processing time of the CPU/GPU implementation is measured under the visual studio 2008 environment using CUDA timer. The time for the CPU/FPGA environment is estimated by calculating the number of clock cycles at 50MHz frequency. We consider the method 3 in Table 1 for the CPU/FPGA implementation. According to the results, CPU/FPGA implementation is slightly slower than the CPU/GPU implementation. However, the power consumption is reduced by more than 10 times. Note that,
Table 4: Processing time

<table>
<thead>
<tr>
<th>Method</th>
<th>Processing time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU + GPU</td>
<td>10.08 (measured)</td>
</tr>
<tr>
<td>CPU + FPGA</td>
<td>11.26 (estimated)</td>
</tr>
</tbody>
</table>

over 90% of the processing time in CPU/FPGA implementation is due to the boundary data transfers from the FPGA global memory to the local memories. Therefore, reducing the data amount by applying techniques such as data compression is necessary.

4. Conclusion

We have proposed a low-power heterogeneous HPC platform with a multicore CPU and an FPGA. We have shown that 95% of the FDTD computation can be done using fixed-point computation without a major precision loss. The low power consumption is achieved by using FPGA accelerators instead of GPU accelerators for most of the computation. Using the proposed low-power heterogeneous platform, we achieved almost the same performance of the CPU/GPU computing at a 10 times lower power consumption.

Acknowledgment

We would like to thank Altera university program for donating us the DE4 FPGA boards for this research.

References