論文の要旨

題目　Efficient GPU Implementations for Bulk Computations
(バルク計算のための効率的な GPU 実装手法)

氏　名　　藤田　徹

A GPU (Graphics Processing Unit) is a specialized circuit designed for manipulating images and video encoding. Latest GPUs are designed for general purpose computing and can perform computation in applications traditionally handled by the CPU. Hence, GPUs have recently attracted the attention of many application developers. The bulk computation is to perform the same algorithm for a lot of instances. The bulk computation has many applications and can be implemented in the GPU very efficiently. This dissertation shows efficient GPU implementations of bulk computations.

In Chapter 1, we show the introduction of the dissertation which includes the research background and contributions. We introduce GPUs and CUDA in Chapter 2. CUDA is a general purpose parallel computing platform and programming model. We develop programs running on GPUs using CUDA.

In Chapter 3, we describe the bulk execution of sequential algorithms and performance analysis. We show the obliviousness and semi-obliviousness of sequential algorithms. We also show that the bulk execution of these algorithms can be implemented very efficiently in CUDA-enabled GPUs.

In Chapter 4, we implement the bulk computation of a Euclidean algorithm computing the Greatest Common Divisor (GCD) of two large numbers in a GPU. We present a new efficient Euclidean algorithm that we call the approximate Euclidean algorithm. The idea of the approximate Euclidean algorithm is to compute a good approximation of quotient by just one 64-bit division and to use it for reducing the number of iterations of the Euclidean algorithm. We also present an implementation of the approximate Euclidean algorithm optimized for CUDA-enabled GPUs. The experimental results show that our parallel implementation of the approximate Euclidean algorithm for 1024-bit integer running on GeForce GTX TITAN X GPU is 90 times faster than the Intel Xeon CPU implementation.

We present Bitwise Parallel Bulk Computation (BPBC) technique for accelerating the bulk computation in Chapter 5. The idea of the BPBC technique is to simulate a combinational logic circuit using bitwise logic operations. Bitwise logic operations compute logical OR, AND, NOT, and XOR operations for the individual bits of 32-bit word. In the BPBC technique, we

store 32 inputs for the combinational logic circuit into a particular bit of the words and we apply the bitwise logic operations corresponding to the combinational logic circuit to the words. Thus, we can compute 32 circuits for 32 inputs at the same time. As an example of application of the BPBC technique, we show the pairwise sums of integers can be accelerated if the value of integers are small. We apply the BPBC technique to two computations, the simulation of Conway's Game of Life and the CKY parsing.

In Chapter 6, we implement the simulation of Conway's Game of Life on the GPU using the BPBC technique. Conway's Game of Life is the most well-known cellular automaton. The universe of the Game of Life is a 2-dimensional array of cells, which takes two possible states, alive or dead. The state of every cell is repeatedly updated according to those of eight neighbors. A cell will be alive if exactly three neighbors are alive, or it is alive and two or three neighbors are alive. This dissertation shows several acceleration techniques for simulating the Game of Life using a GPU as follows: (1) the states of 32/64 cells are stored in 32/64-bit words (integers) and the next states are computed by the BPBC technique, (2) the states of cells stored in 2 words are updated at the same time by a thread, (3) warp shuffle instruction is used to directly transfer the current states stored in registers, and (4) multiple-step simulation is performed to reduce the overhead of data transfer and invoking CUDA kernel. Using these technique, we have obtained extremely fast GPU implementation for simulating the Game of Life using GPUs. The experimental results show that our GPU implementation using GeForce GTX TITAN X performs 1350×10^9 updates per second for 16K-step simulation of 512K×512K cells stored in the SSD. Since Intel Core i7 CPU using the same technique performs 13.4×10^9 updates per second, our GPU implementation for the Game of Life achieves a speedup factor of 100.

In Chapter 7, we apply the BPBC technique to the CKY parsing and implement it on the GPU. The CKY parsing determines whether the context-free grammar derives an input string. The idea of the CKY parsing is to compute a 2-dimentional table called CKY table by the dynamic programming technique. Each element of the CKY table stores a subset of non-terminal symbols. Our idea to apply the BPBC technique to the CKY parsing is to compute 32 CKY tables for 32 input strings at the same time. Since the CKY parsing can be done by iterative simulation of the combinational logic circuit, the BPBC technique can be applied to it. We show that the CKY parsing can

be implemented in the GPU efficiently using the BPBC technique. The experimental results using Intel Core i7 CPU and GeForce GTX TITAN X GPU show that the GPU implementation for the CKY parsing runs more than 400 times faster than the CPU implementation.

We conclude the dissertation in Chapter 8.