論　文　の　要　旨

題　目　　A Study of Multiple-length Multiplication on the GPU
（GPU における多倍長整数乗算に関する研究）

氏　名　本田　巧

Applications require arithmetic operations on integer numbers which exceed the range of processing by a CPU directly is called multiple-length numbers or multiple-length precision numbers and hence, computation of these numbers is called multiple-length arithmetic. More specifically, application involving integer arithmetic operations for multiple-length numbers with size longer than 64 bits cannot be performed directly by conventional 64-bit CPUs, because their instruction supports integers with fixed 64 bits. To execute such application, CPUs need to repeat arithmetic operations for those numbers with fixed 64 bits which increase the execution overhead. Suppose that a multiple-length number is represented by w words, that is, a multiple-length number is $64w$ bits on conventional 64-bit CPUs. The addition of such two number can be computed in $O(w)$ time. However, the multiplication generally takes $O(w^2)$ time. Multiple-length multiplication is widely used in various applications such as cryptographic computation, and computational science. Since multiple-length numbers of size thousands to several tens of thousands bits are used in such applications, the acceleration of the computation of their multiplications is in great demand. Also, considering practical cases, a large number of multiplications are usually computed. Therefore, the acceleration of a lot of multiple-length multiplications is necessary.

A GPU (Graphics Processing Unit) is a hardware specialized for image processing and has a lot of cores and very high memory bandwidth. Modern GPUs are designed for general purpose computing and can perform various computations traditionally handled by the CPU. NVIDIA provides a parallel computing architecture called CUDA (Compute Unified Device Architecture) for NVIDIA GPUs. Using CUDA, we can develop parallel processing programs to be implemented in GPUs. Since GPUs have a lot of cores and very high memory bandwidth, a GPU can handle multiple tasks simultaneously, the GPU has a high efficiency which denotes performance per watt. Thus, a GPU attracts a lot of attention as a computational accelerator for high performance computing. GPUs are also used to accelerate computations of applications in cryptographic computation and computational science. Therefore, the acceleration of multiple-length multiplications on GPUs is in great demand. In addition, a large number of multiple-length multiplications with different inputs are performed in cryptographic computation and computational science, e.g. vector multiplication and matrix multiplication. Therefore, in this work, we consider an efficient algorithm for bulk execution of multiple-length multiplication on a single GPU. This dissertation shows a GPU implementation for bulk execution of multiple-length multiplication. In addition, this dissertation shows GPU implementations for exhaustive verification of the Collatz

conjecture which needs to perform multiple-length multiplications.

We introduce our research background in Chapter 1. Also, we briefly introduce multiple-length multiplication algorithms. In Chapter 2, we show the detail of the GPU and CUDA architectures.

We present a GPU implementation for bulk execution of multiple-length multiplication in Chapter 3. The idea of our GPU implementation is to adopt a warp-synchronous programming technique. We assign each multiple-length multiplication to one warp that consists of 32 threads. In parallel processing using multiple threads, usually, it is costly to synchronize execution of threads and communicate within threads. In warp-synchronous programming technique, however, execution of threads in a warp can be synchronized instruction by instruction without any barrier operations. Also, inter-thread communication can be performed by warp shuffle functions without accessing shared memory. We propose 1024-bit multiple-length multiplication method using warp-synchronous programming technique. The experimental results show that our GPU implementation attains a speed-up factor of 52 for 1024-bit multiple-length multiplications over the sequential CPU implementation. Moreover, we use this 1024-bit multiplication method for larger size of bits as a sub-routine. In addition, we use Toom-Cook method to reduce the number of multiplications. The GPU implementation attains a speed-up factor of 21 for 65536-bit multiple-length multiplications.

We propose GPU implementations of exhaustive verification of the Collatz conjecture in Chapter 4. Consider the following operations on an arbitrary positive number: if the number is even, divide it by two, and if the number is odd, triple it and add one. The Collatz conjecture asserts that, starting from any positive number, repeated iteration of the operations eventually produces the value 1. We use a CPU-GPU cooperative approach, efficient memory access for the GPU memory, and optimization of multiplication to accelerate the verification. The experimental results show that, our GPU implementation can verify 1.31×10^{12} 64-bit numbers per second. While the sequential CPU implementation can verify 5.25×10^{9} 64-bit numbers per second. Thus, our implementation on the GPU attains a speed-up factor of 249 over the sequential CPU implementation. Additionally, we accelerated the computation of counting the number of the above operations until a number reaches 1, called delay, that is one of the mathematical interests for the Collatz conjecture by the GPU. Using similar ideas, we achieved a speed-up factor of 73.