論 文 の 内 容 の 要 旨

論文題目 Design and Implementation of Hardware Accelerators with Multi-Level Parallelization and Application-Oriented Data Layout
(マルチレベル並列化とアプリケーション指向データレイアウトを用いるハードウェアアクセラレータの設計と実装)

氏　　名　　小泉　賢一

In this thesis, we describe the design and implementation of high-performance hardware accelerators and propose a design methodology. In software programming, architecture of the processor, functions of operating system, and optimization of compiler support acceleration. However, in hardware design, there is no support as there is in software programming; thus, hardware engineers must optimize designs on their own. Engineers have greater flexibility when designing hardware; however, a design methodology for high-performance accelerators has not been established. We have established a design methodology through the design and implementation of accelerators using field-programmable gate arrays (FPGAs). To explain the proposed methodology, we select three accelerators and describe three targeted studies.

The first application is improving the throughput performance of TCP communication in long-distance fat-pipe networks (LFNs). It is known that TCP communication on LFNs is difficult, and obtaining good performance in parallel TCP communication is more difficult than with single TCP. To address this problem, we propose a hardware solution that balances streams. We implement the merging stream harmonizer (MSH) on MaSTER-1, our programmable network testbed. We can utilize 99.6% of 10-Gbps LAN PHY bandwidth in pseudo LFNs and 87.0% of 9.2-Gbps WAN PHY bandwidth in real LFNs. Note that different sections of LFNs have various bandwidths, and packet loss often occurs when the total input bandwidth is greater than that of the output. Using MaSTER-1, we analyzed the buffer effects of such switches and the relationship between round-trip time (RTT) and buffer size.

The second study involved the computer Go game. A Monte Carlo tree search method that involves Monte Carlo simulations has been developed to find the best next move in the Go

game. The method increases the strength of the Computer-Go program. The effectiveness of this method depends on the number of simulations. Unfortunately, FPGA-based acceleration was difficult because, in this context, resource consumption tends to be high. FPGA-based acceleration was feasible for a 9 x 9 grid board; however, it was not feasible for a 19 x 19 grid board. We propose a triple line-based playout for Go (TLPG) hardware algorithm. By reproducing global information redundantly, the TLPG algorithm generates simulations using only local operations, which helps in realizing compact hardware logic implementations. We implemented TLPG in MaSTER-1. The results indicate that the TLPG algorithm can perform 40,649 playouts per second for a 9 x 9 grid board and 4,668 playouts per second for a 19 x 19 grid board.

The third study involved skyline computation, which is a method to extract interesting entries from a large population with multiple attributes. When the population changes dynamically, calculating a sequence of skyline sets is referred to as continuous skyline computation. Previous methods that employ divide and conquer and geometric algorithms are not robust in higher dimensional space. We propose the balanced jointed rooted tree (BJR-tree), which can represent a dominance relation as an arc. In addition, tree traversal at a deep position can be delayed to reduce unnecessary calculations. We also propose the low-latency skyline computation accelerator (LSCA) as a hardware algorithm. The LSCA parallelizes dominance relation calculations and evaluates postponed calculations during idle states. We implemented the LSCA on an FPGA and evaluated our software and hardware implementations. BJR-tree is approximately up to 70 times faster than LookOut on synthetic datasets. In addition, the LSCA is approximately 2.5 to 4.4 and 1.7 to 35 times faster than an Intel CPU running software implementations on synthetic and real-world datasets, respectively.

The proposed methodology, established through our studies including the above three, indicates the design flow for multi-level parallelization and application-oriented data layout. Note that our perspective relative to hardware design is not considered in current behavioral synthesis technology. In another three calculations, we compared the performance of a circuit generated automatically using a behavioral synthesis tool and a circuit designed based on the proposed methodology. The results show that the design based on the proposed methodology is more efficient than the behavior-based design. We expect that the proposed design methodology will provide a guideline for hardware designers and will be incorporated into behavioral synthesis in the future. Thus, the proposed design methodology is expected to contribute to an effective and efficient accelerator design.