博士論文要旨

論文題名：組込みシステム設計における
マルチコアタスクスケジューリングと高位合成

立命館大学大学院理工学研究科
電子システム専攻博士課程後期課程
ニシカワ　ヒロキ
西川　広記

マルチコアアーキテクチャは、組込みシステムの性能を引き出す技術として注目されている。この技術により、アプリケーションにおける処理単位であるタスクはマルチコアを用いて並列に実行可能となり、シングルコア以上の高性能化が実現される。しかしながら、組込みシステムにおけるコア数の増加に伴い、システム設計は一層複雑化しており、並列化技術や設計自動化技術のさらなる発展が期待される。本論文では、マルチコアを効率的に利用する種々のタスクスケジューリング手法を提案する。さらに、組込みシステムの回路設計おいてモジュールを効率的に利用する設計自動化手法を提案する。

タスクスケジューリングとは、アプリケーションにおけるタスクの実行順序を決定し、タスクをコアに割り当てる作業を指す。古典的なタスクスケジューリングは、複数のタスクを異なるコアに割り当てることによりスケジュール長の短縮を目指すが、各々のタスクは1個のコア上で実行されることを想定している。一方、本研究では、各タスクが複数のコア上で並列に実行されることを許し、各タスクに割り当てるコア数をタスクスケジューリングと同時に決定する。本論文では、このタスクスケジューリング問題に対し、良い解を高速に求める制約プログラミング手法を提案する。実験により、提案手法は従来手法よりも良いスケジュールを発見可能であり、スケジュール長を最大で81.9%削減することを示す。さらに、提案手法を二つの方向に拡張する。一つ目の拡張は、データ通信のオーバーヘッドの考慮である。マルチコア上では複数のタスクが並列に実行されるが、多くの場合は並列処理のために発生するデータの通信などのオーバーヘッドによってレイテンシが低下する。そこで、データ通信の遅延を考慮したタスクスケジューリング手法を提案する。実験により、提案手法は従来手法に比べスケジュール長を平均22.5%短縮することを示す。二つ目の拡張は、消費エネルギーの考慮である。高性能だが消費電力が高いコアと、低性能だが消費電力が低いコアの2種類のコアをもつ非均質なマルチコアアーキテクチャを対象として、スケジューリングと同時にコアの種類を最適に決定することにより、総消費エネルギーを削減する。実験により、提案手法は従来手法に比べ最大で12.4%消費エネ

ルギーを削減することを示す。

ハードウェア回路設計の自動化の観点から、ソフトウェアプログラムからハードウェア回路を自動的に合成する高位合成（HLS）技術が注目され、普及が進んでいる。HLS 技術は制約条件を満たす回路を自動的に合成できるが、手動で設計した場合に比べ回路面積が増大する。その理由の一つに、プログラム中の複数の箇所から同じ関数が呼び出される場合、その関数を実現する回路モジュールが複数生成されることが挙げられる。本論文では、複数のモジュールを生成することなく、一つのモジュールを共有する高位合成手法を提案する。FPGA を用いた実験により、提案手法は最大でルックアップテーブルを 14.9%削減し、フリップフロップを 19.1%削減することを示す。

Abstract of Doctoral Dissertation

Title: Multicore Task Scheduling and High-Level Synthesis for Embedded Systems

Doctoral Program in Advanced Electrical, Electronic and Computer Systems
Graduate School of Science and Engineering
Ritsumeikan University

ニシカワ　ヒロキ
NISHIKAWA Hiroki

Multicore architectures have been appealing to fully exploit the potential of embedded systems. The multicore architectures enable tasks in an application to run on multiple cores in parallel and achieves higher performance than single-core architectures. However, as the increasing number of the multicore, system-design processes become very complex, and further development of parallel computing and design automation technologies is required.

Task scheduling is one of system-design processes that determines the execution order of the tasks in an application and the assignment of the tasks onto cores. Classical task scheduling assumes that a task is executed on one of the cores, and different tasks can run on different cores in parallel in such a way that the overall schedule length is minimized. On the other hand, this thesis allows each task to run on multiple cores in parallel, and the number of cores assigned to a task is determined during task scheduling simultaneously. This thesis proposes a technique to rapidly find a good schedule based on constraint programming. The experiments demonstrate that our proposed technique reduces the schedule length by up to 81.9%. The proposed technique is further extended towards two directions. One direction is to consider the performance overheads for communications among cores. In this thesis, we propose a communication-aware scheduling technique. The experiments demonstrate that the proposed technique reduces the schedule length by 22.5% on average compared to the state-of-the-art techniques. The other direction is to consider energy efficiency. This work targets heterogeneous multicores that consist of two types of cores, where one is high-performance but high-power cores and the other is low-power but low-performance cores. In this work, the proposed technique determines an optimal type for each core at the same time as scheduling of the tasks. In the experimental results, the proposed technique reduces energy consumption by up to 12.4% compared with the state-of-the-art techniques.

Regarding efficient hardware design, high-level synthesis (HLS) that automatically translates

software programs into hardware descriptions has now become an indispensable technology. Although HLS techniques can quickly generate circuits that satisfy design constraints, the circuit area is often larger than that of manually designed circuits. One of the reasons is that common HLS tools create multiple instances of a same hardware module when a function is called multiple times in the original software program. Our proposed HLS techniques generate only a single instance of the module which is shared in a time-division manner. The experiments using FPGA shows that the proposed techniques reduce the number of look-up tables by up to 14.9% and flip-flops by up to 19.1%.