

ハードウェア IP の応答時間を考慮したプロセッサコア合成システム

小原 俊逸[†] 田川 博規[†] 戸川 望^{†,††} 柳澤 政生[†] 大附 辰夫[†]

[†] 早稲田大学理工学部電子・情報通信学科

^{††} 北九州市立大学国際環境工学部情報メディア工学科

^{†††} 早稲田大学理工学総合研究センター

〒 169-8555 東京都新宿区大久保 3-4-1

Tel: 03-3209-3211(5716), Fax: 03-3204-4875

E-mail: kohara@yanagi.comm.waseda.ac.jp

あらまし 本稿では、ハードウェア IP の応答時間を考慮したプロセッサコアの合成システムと、これを利用したシステム LSI 設計のフレームワークを提案する。ハードウェア IP を利用したシステム LSI 設計では、システム LSI に要求される性能に対して必要にして十分な性能の IP が必ずしも用意されているとは限らない。そこでシステムのハードウェア/ソフトウェア分割後、ハードウェアで実現する機能には IP を用いるが、ソフトウェアを動作させるプロセッサコアには IP を用いず、アプリケーションに応じて性能に過不足のないプロセッサコアを自動合成する。提案するフレームワークに沿って JPEG エンコーダを設計し、計算機実験により提案する合成システムとフレームワークの有効性を示す。

キーワード プロセッサコア、ハードウェア IP、システムレベル設計、ハードウェア/ソフトウェア協調設計

A Processor Core Synthesis System Based on Response Time of Hardware IPs

Shunitsu KOHARA[†], Hiroki TAGAWA[†], Nozomu TOGAWA^{†,††}, Masao YANAGISAWA[†], and
Tatsuo OHTSUKI[†]

[†] Dept. of Electronics, Information and Communication Engineering, Waseda University

^{††} Dept. of Information and Media Sciences, The University of Kitakyushu

^{†††} Advanced Research Institute for Science and Engineering, Waseda University

3-4-1 Okubo, Shinjuku, Tokyo 169-8555, Japan

Tel: +81-3-3209-3211(5716), Fax: +81-3-3204-4875

E-mail: kohara@yanagi.comm.waseda.ac.jp

Abstract This paper proposes a processor core synthesis system based on response time of hardware IPs, and a framework for system LSI design over the synthesis system. In case of designing a system LSI using hardware IPs, IPs which are necessary and sufficient performance for the system LSI are not always provided. Our approach is as follows: After system-level hardware / software partitioning, we use IPs for hardware, but not processor core IPs for software. We use a processor core which is auto-synthesized by the proposed synthesis system and has just enough performance. We design a JPEG encoder within the framework and the results demonstrate its effectiveness and efficiency.

Key words processor core, hardware IP, system level design, hardware / software codesign

1. ま え が き

LSI 集積技術の進歩により、大規模な回路を 1 チップに搭載可能になった。これに伴い、システム LSI は大規模化、複雑化し、設計コストが費用・期間の両面において増加している。設計・開発効率や品質を向上させるには、過去の設計資産である IP (Intellectual Property) の利用や、ハードウェア/ソフトウェアの協調設計が必要である。

IP を利用したシステム LSI 設計における一般的な手法の 1 つに、アプリケーションの機能をハードウェアで実現する機能とソフトウェアで実現する機能に分割した上で、ハードウェア部分は特定の機能を実現するハードウェア IP、ソフトウェア部分はプロセッサコア IP 上で動作するソフトウェアで実現するという手法がある。

このとき、システムに要求される性能に対して必要にして十分な性能の IP が必ずしも用意されているとは限らない。どの IP を選択しても、ある IP は性能不足になり、ある IP は性能過多となる。

そこでハードウェアで実現する機能には IP を用いるものの、プロセッサコアには IP を用いず、アプリケーションに応じて性能に過不足のないプロセッサコアを自動合成するシステムを提案する。動作するアプリケーション・ソフトウェアに応じて最適なプロセッサコアを自動合成する研究は [1], [3], [9] 等があるが、本研究はこういったプロセッサコアレベルのハードウェア/ソフトウェア協調設計のアプローチに加え、プロセッサコアの外にあるハードウェア IP を考慮する。

ハードウェアとソフトウェアの通信手段は主に、ボーリングによるものと割込みによるものがある。プロセッサコアのハードウェア/ソフトウェア協調設計では、ソフトウェアの命令列をベースにしていることを考えると、ソフトウェアがハードウェアを命令で制御するボーリングによるものの方が適していると考えられる。このとき、ハードウェア IP とプロセッサのインタフェースが固定であるとする、ソフトウェアから見てハードウェア IP の最も重要な要素は、データ処理の実行時間やデータの転送時間などの応答時間である。そのため、ハードウェア IP を制御する命令を含むソフトウェアを動作させるプロセッサコアの合成は、従来のハードウェア/ソフトウェア協調設計で考慮されていなかった、ハードウェア IP の応答時間を考慮する必要がある。

こうした背景からハードウェア IP の応答時間を考慮したプロセッサコアの合成システムを提案する。ハードウェア IP は主にマルチメディア処理に用いられるような、データ処理に対して応答時間が一定であるものを想定する。また、提案システムはプロセッサコアを合成するものではあるが、システムに対して最適なプロセッサコアを合成するためにはプロセッサコア上で動作するソフトウェアだけでなく、システムのアーキテクチャやハードウェア IP の情報も必要である。したがって、本稿ではプロセッサコア合成システムの他、同システムを利用したシステム LSI 設計のフレームワークを提案する。システムレベル記述からのソフトウェア記述抽出、ハードウェア IP の応答時間の見積り、これらを考慮したコンパイラとプロセッサコア・ハードウェア/ソフトウェア分割系によって、対象とするシステム LSI が要求する性能に対し、過不足のない性能を有するプロセッサコアを自動合成できる。

本稿では、まず対象とするシステム LSI のアーキテクチャを定義する。次に同システムを利用した設計フレームワークを説明し、提案するプロセッサコア合成システムを説明する。最後にアプリケーションとして JPEG エンコーダを設計、本システムを適用して提案手法を評価する。

2. アーキテクチャ・モデル

本システムが対象とするシステムとプロセッサのアーキテクチャを定義する [10]。

2.1 システムアーキテクチャ・モデル

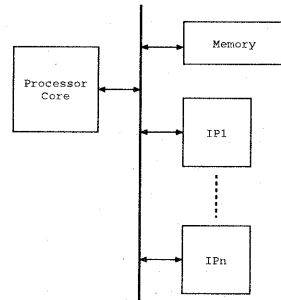


図1 システムアーキテクチャ・モデル

本システムが対象とするシステムアーキテクチャ・モデルを図1に示す。1つのプロセッサとメモリ、及び1つ以上のハードウェア IP がバスで接続される、一般的なアーキテクチャである。ハードウェア IP のインタフェースは固定とし、全てプロセッサからの命令で制御される。よってこれらのハードウェア IP はコプロセッサと見なすことができる。そのインタフェースは ARM7TDMI [2] のコプロセッサインタフェースに準じたものとする。

2.2 プロセッサコアアーキテクチャ・モデル

プロセッサコアのアーキテクチャモデルは、文献 [8] で提案されたモデルに基づいている。プロセッサコアは VLIW 型で、カーネル型として DSP 型と RISC 型を選択可能であり、カーネル型に専用演算器 (ALU, 乗算器等) やレジスタファイルなどのハードウェアユニットを付加することができるものである。命令セットは基本命令と複合命令に加えてハードウェア IP 命令がある。基本命令は、市販のデジタル信号処理プロセッサを基本としており [6]、カーネルを構成するハードウェアおよび付加されるハードウェアユニットに対応した命令である。複合命令は、基本命令を複数個並列に実行する命令である。基本命令のあらゆる組合せを複合命令として持つのではなく、実行ソフトウェアに応じて複合命令となる命令の組合せを決定する。ハードウェア IP 命令は ARM7TDMI のコプロセッサ命令に準じたもので、表1のような命令がある。

表1 ハードウェア IP 命令

データ操作命令	CDP
データ転送命令	LDC, STC
レジスタ間転送命令	MCR, MRC

3. 設計フレームワーク

本システムの利用を前提とした設計フレームワークを提案する。

まず設計者はシステムで実現するアプリケーションを仕様に沿ってシステムレベル記述言語で記述する。この段階ではまだハードウェア/ソフトウェアは分割されていない。次にシステムに要求される性能と評価・検証した上で、アーキテクチャの探索と機能のハードウェア/ソフトウェア分割を行う。その結果として、前述したアーキテクチャでシステムの実現が可能である場合に本システムが利用可能となる。このとき、設計はシステムレベル設計からプラットフォームベース設計へシフトする。対象アーキテクチャに準拠したプラットフォームへのマッピングとも言える。

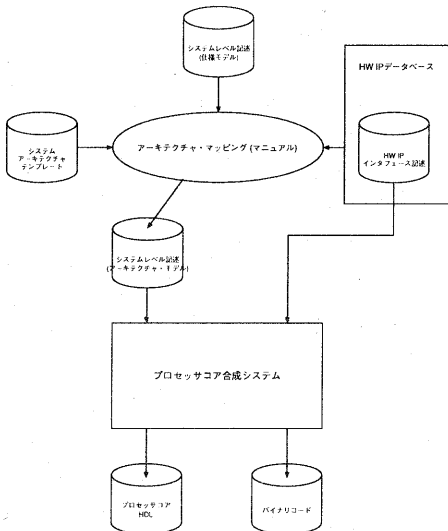


図2 設計フレームワーク

図2はシステム LSI を対象とするアーキテクチャで実現することを前提としたときの、システム設計フレームワークになる。それぞれの要素について説明する。

システムレベル記述 (仕様モデル) システムレベル記述 (仕様モデル) とは、システムレベル記述言語で記述された、対象システムの仕様をモデリングしたものである。一般的なシステムレベル設計の設計フローにならない、まずシステム設計者は要求仕様を分析した後、シミュレーション可能な仕様モデルとしてシステムとして具備すべき機能をシステム記述言語で設計する。システム記述言語は C++ のクラスライブラリとして実装されている SystemC [7] を使用する。

SystemC を使用する理由は主に 2 つある。1 つは C++ の特徴であるクラスの継承が、後述するシステムアーキテクチャ・テンプレートを提供する際に特に有効であるということである。もう 1 つは C/C++ によるプログラミングが制限なしに可能であるということである。これはソフトウェア記述をモデリングの域に留めず、実際にプロセッサコア上で動かすプログラムを組むことを可能にしている。

アーキテクチャ・マッピング アーキテクチャ・マッピングとは設計タスクの 1 つで、仕様モデルのシステムレベル

記述を評価・検証し、探索したアーキテクチャに機能を割り当てるタスクである。ここではハードウェア IP のデータベースから利用するハードウェア IP を選択し、システムアーキテクチャ・テンプレートを利用して、仕様モデルのシステムレベル記述を、対象とするアーキテクチャをモデリングしたアーキテクチャ・モデルのシステムレベル記述にマッピングする。

システムアーキテクチャ・テンプレート システムアーキテクチャ・テンプレートとは、アーキテクチャモデルの SystemC によるシステムレベル記述ファイル群と、それに付随する仕様である。システムアーキテクチャ・テンプレートを用意することで、設計者のアーキテクチャ・マッピングの便宜を図り、プロセッサコア合成システムに適した入力とする。システムアーキテクチャ・テンプレートの概略を図3に示す。

テンプレートファイルはアーキテクチャをモデリングしたもので、SystemC で記述されている。

予約語のうち、クラスと変数はテンプレートファイルで使用されたものについて衝突を避けるために設けられている。

関数は表1のハードウェア IP を制御するアセンブリ命令に直接対応する。通常、C/C++ ソース上でアセンブリ命令を直接扱うにはインラインアセンブラ等を使用するが、このように予約された関数として扱うことで、C/C++ 言語レベルでシミュレーションが可能になる。

定数 `__SW__` は後述するプロセッサコア合成システムでのソフトウェア抽出時に重要な定数で、ソフトウェア抽出は定数 `__SW__` を `#define` した状態でプリプロセッサに入力することで実現する。そのため `#ifdef`、`#ifndef` 等を使用してシステム記述のソースを適宜調整する必要が生じる場合もある。

HW IP データベース HW IP データベースとは、利用可能なハードウェア IP の情報が格納されたデータベースで、ハードウェア IP に対する情報として機能の概要と後述するインタフェース記述を持つ。アーキテクチャ・マッピングでは、アプリケーションをハードウェア/ソフトウェア分割し、ハードウェアで実現する機能には IP を利用する。設計者はハードウェア IP の機能の詳細を知る必要はなく、機能の概要とインタフェースが分かればよい。

HW IP インタフェース記述 HW IP インタフェース記述とは、形式的に定義された、ハードウェア IP の I/O ピン情報や論理的な信号変化の仕様である。ハードウェア IP のインタフェース記述言語として CWL (Component Wrapper Language) [5] が提案されている。CWL は、ハードウェア IP が内部に持つ機能と外部の端子の信号値および信号の変化順序を関連づけるための言語で、正規表現をベースにしている。

提案する設計フレームワークの環境として、ハードウェア IP のデータベースを想定し、データベースには各ハードウェア IP の CWL 記述を含むとする。CWL 記述を使用する理由は 2 つある。1 つは CWL 記述によってハードウェア IP の応答時間を機械的に見積もることができるからであり、もう 1 つは対象とするシステムのアーキテクチャにそぐわない IP でも、CWL 記述によってラッパーを容易に合成、或いは設計することが可能であると考えられるからである。

テンプレートファイル

- main.cpp: シミュレーション部本体
- bus.cpp, bus.h: HW/SW バスモデル
- sw_interface.h: HW/SW インタフェースモデル
- sw.cpp, sw.h: SW テンプレート
- hwip.cpp, hwip.h: HW IP モデル
- sim.cpp, sim.h: シミュレーション用 IP モデル

予約語

クラス BUS, SW_INTERFACE, SW, HWIP, SIM

関数 cdp(), ldc(), stc(), mrc(), mcr();

変数 swif

定数 __SW__

図3 システムアーキテクチャ・テンプレート

システムレベル記述（アーキテクチャ・モデル） システムレベル記述（アーキテクチャ・モデル）とはアーキテクチャ・マッピングを経て得られる、仕様モデルをアーキテクチャにマッピングしたシステムレベル記述である。システムアーキテクチャ・テンプレートとそれに付随する仕様に沿った記述にすることで、下流のプロセッサコア合成システムの適切な入力となる。

4. プロセッサコア合成システム

提案するプロセッサコア合成システム（図4）を説明する。このシステムはプロセッサコア合成に関して文献[11]と同様のアプローチをとる。

文献[11]で提案されたアプローチは次のとおりである。C言語記述のアプリケーションプログラム、アプリケーションデータ、アプリケーションに対する実行時間制約を入力とし、まずプロセッサカーネルに想定される全てのハードウェアユニットを付加し、アプリケーションの実行速度は最速だが、プロセッサコアの面積が最大であるプロセッサコアの構成と、そのプロセッサ上で動作するアセンブリコードを得る。続いて、このアセンブリコードに対し、ハードウェアによる実現部分を徐々にソフトウェアに代替する。これによって実行に有するクロックサイクル数は徐々に増えるが、プロセッサコアに必要とされる面積は小さくなる。時間制約を満たす限りこの操作を繰り返すことによって、最小面積のプロセッサを合成する。

本システムもソフトウェアの命令列をベースにしていることからこのアプローチを踏襲する。ただし、入力プロセッサ上で動作するソフトウェアのC言語記述ではなく、前述した設計タスクから作成されるSystemCによるシステムレベル記述（アーキテクチャ・モデル）と、選択したハードウェアIPのCWLによるインタフェース記述となる。アプリケーションデータはシステムレベル言語で記述されたモデルのシミュレーションに必須であるので、ここではシステムレベル記述に含むものとする。これらとアプリケーションの時間制約を入力とし、時間制約を満たす最小面積のプロセッサコアのHDLとそのプロセッサ上で動作するソフトウェアのバイナリコードを出力する。

以下、それぞれの要素について説明する。

応答時間見積り系 選択したハードウェアIPのインタフェース記述を入力とし、見積もった応答時間を出力する。ここで応答時間とは、ハードウェアIPのデータ転送時間やデータ処理時間であり、クロックサイクル数の固定値として与

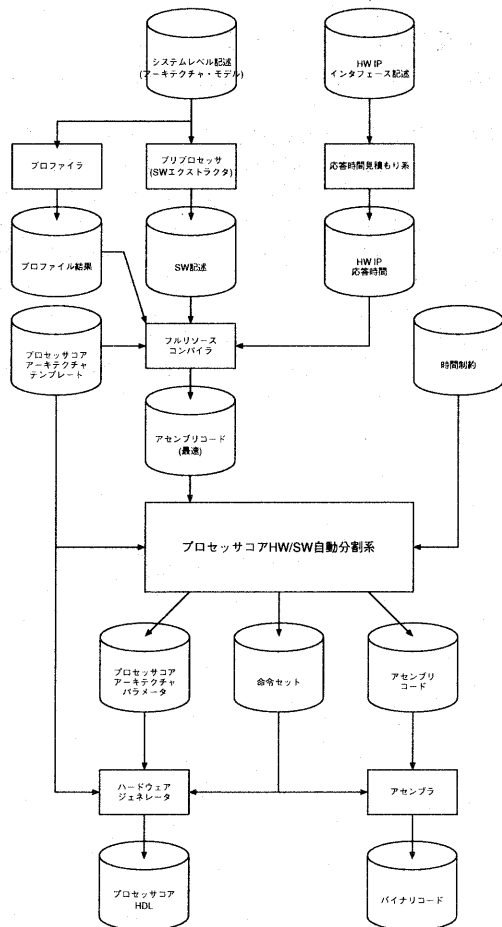


図4 プロセッサコア合成システム

えられる。本システムでは、画像処理用途などの応答時間が固定であるハードウェアIPのみを扱う。

CWLは論理的な信号変化を正規表現ベースの記法で表現しているため、これを適切に処理することで応答時間を見積ることができる。応答時間はプロセッサコア上で動作するハードウェアIP命令と1対1の対応を持つ。

プリプロセッサ（SWエクストラクタ） システムアーキテクチャ・テンプレートを利用して、その仕様に従ったアーキテクチャ・モデルのシステム記述を入力とし、その中からソフトウェア記述（C/C++）を抽出する。図2のシステムアーキテクチャ・テンプレートの仕様の1つである定数__SW__を#defineしてプリプロセッサを通すことでソフトウェア記述を抽出することができる。このソフトウェア記述が通常のC/C++プログラムソースと異なるのは、システムアーキテクチャ・テンプレートで規定したハードウェアIP命令に関する関数を予約関数として扱っている点のみである。

プロファイラ アーキテクチャ・モデルのシステム記述を入力とし、シミュレーション時のソフトウェア部分のプロファイリングを行う。計測用のコードを埋め込み、これを実行することでソフトウェア部分のプロファイリングが可

能になる。

フルリソースコンパイラ 使用するハードウェア IP に対する命令と応答時間、抽出されたソフトウェア記述、プロファイル結果、及びプロセッサコアアーキテクチャ・テンプレートを入力とし、ソフトウェア記述をコンパイルしてアセンブリコードを依存グラフとプロファイル情報を付加して出力する。このコンパイラでは、プロセッサコアに最大数のハードウェアユニットが付加されていると仮定し、命令を可能な限り並列に実行するようにコンパイルする。つまり、出力されるアセンブリコードは実行時間が最短であるが、プロセッサコアの面積は最大となる。また、予約関数として定義されているハードウェア IP に関連した関数は、入力された応答時間を考慮して処理される。

プロセッサコア HW/SW 自動分割系 入力は、フルリソースコンパイラから出力されるアセンブリコード、プロセッサコアアーキテクチャ・テンプレート、及び時間制約である。出力は入力された時間制約を満たす最速のアセンブリコード、このアセンブリコードが動作する面積最小のプロセッサコアを表すプロセッサコアアーキテクチャ・パラメータ、及び合成されたプロセッサコアの命令セットである。ハードウェア/ソフトウェア分割では、最速のアセンブリコードにおいてハードウェアで実現している機能を、ソフトウェアに置き換えていくことによってハードウェア面積を小さくしていく。ハードウェア面積は徐々に小さくなるが、アセンブリコードが長くなり、実行に時間がかかるようになる。これを時間制約が満たす限り繰り返す [10]。

ハードウェアジェネレータ プロセッサコアアーキテクチャ・テンプレート、プロセッサコアアーキテクチャ・パラメータ、命令セットを入力とし、プロセッサコアの HDL 記述を出力する。系内にライブラリとして命令動作、プロセッサ構造、ハードウェアユニットを用意することで、入力されたデータ幅、命令セット、命令フォーマットに対して適切なプロセッサコアの HDL 記述を自動生成できる。

アセンブラ アセンブリコード、命令セットを入力として、入力された命令セットを持つプロセッサコア上で動作するバイナリコードを出力する。

5. 計算機実験結果

5.1 JPEG エンコーダの設計

JPEG エンコーダをアプリケーションとして、提案システムを利用したシステム設計を行った。JPEG [4] はフルカラーまたはグレースケールの静止画像に対する圧縮符号化方式である。画像を 8×8 画素のブロックに分割し、これを基本単位として圧縮する。基本 DCT 方式の場合、各ブロックに対し次のような処理が行われる。

- (1) 色空間変換 (RGB to YCbCr)
- (2) DCT (離散コサイン変換)
- (3) 量子化
- (4) ハフマン符号化

ただし画像がグレースケールの場合、色空間変換は行われぬ。今回のモデルでは、横 640 画素 $\times$ 縦 480 画素 $\times$ 8 ビットグレースケールの画像のエンコードを行うものとした。

5.1.1 SystemC による仕様のモデリング

SystemC で JPEG エンコーダの仕様をモデリングした。シミュレーション時には BMP ファイルを入力とし、JPEG ファイルを出力する。まずは、1 ブロックに対して DCT、量

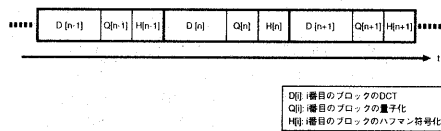


図 5 JPEG エンコード処理 (仕様モデル)

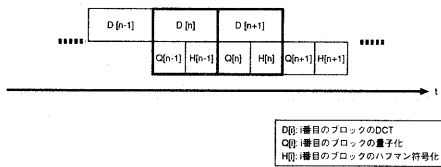


図 6 JPEG エンコード処理 (HW/SW バイブライン動作)

子化、ハフマン符号化を順番どおり行った (図 5)。

これを PC/AT 互換機 (CPU: Pentium III 500MHz, RAM: 192MB, OS: Linux 2.4) 上でプロファイルした結果、DCT の実行時間割合が 35% で最も大きかった。そこで、DCT にハードウェア IP を用い、他をプロセッサ上のソフトウェアで実現することとした。

5.1.2 アーキテクチャ・マッピング

システムアーキテクチャ・テンプレートを利用して、仕様モデルをアーキテクチャモデルにマッピングした。DCT をハードウェアで実現するため、プロセッサ上で動作するソフトウェアと DCT がパイプライン動作するようにした (図 6) また、ハードウェア IP として使用する DCT は、Xilinx の提供する 2 次元 DCT の IP [12] を参考として用いた。

5.2 実験 I

表 2 は、JPEG エンコーダのアーキテクチャモデルと DCT の IP のインタフェース記述をプロセッサコア合成システムへの入力とし、実行制約時間を与えた時のプロセッサコア HW/SW 分割系の出力結果である。ただし応答時間見積りに関しては、応答時間見積り系が未実装のため人手で行なった。与えた実行時間制約に対する動作ソフトウェアの総実行時間、プロセッサコアの面積、及びプロセッサコア構成である。プロセッサコア構成にはカーネル型、命令並列度、専用演算器の種類と数、レジスタ数、Y メモリの有無がある。

また、比較のための汎用 RISC プロセッサコアとして、5 段パイプライン、32bit 汎用レジスタ 32 本、命令長 32bit、乗算命令を持つ仮想的なプロセッサコアを、対象アーキテクチャで実現した。動作ソフトウェアの総実行時間とプロセッサコアの面積を見積もった結果、総実行時間 225.621[ms]、面積 2,107,831[μm^2] になった。これと表 2 の結果をグラフにしたものを図 7 に示す。

5.3 実験 II

ハードウェア IP である DCT のデータ処理時間を T とすると、実験 I で T は 93 サイクルであったが、同機能で性能の異なる IP を使用したと仮定して、T を変化させて実験 I と同様の実験を行った。

T を短くしていった場合には実験 I と得られる解に変化はなかった。T を長くしていった場合、

- 解 A は T が 393 サイクル以上
- 解 B は T が 420 サイクル以上

表 2 設計 JPEG エンコーダに対するプロセッサコア HW/SW 分割系の出力結果

解	時間制約 [ms]	総実行時間 [ms]	面積 [μm^2]	プロセッサコア構成				
				カーネル型	並列度	専用演算器	レジスタ数	Y メモリ
A	120.0	113.740	4,106,896	RISC	4	ALU*2, 乗算器*2	20	Yes
B	130.0	129.799	2,298,954	RISC	4	ALU*1, 乗算器*1	5	Yes
C	150.0	146.212	1,758,458	RISC	2	ALU*1, 乗算器*1	6	Yes
D	170.0	169.273	1,623,141	DSP	2	ALU*1, 乗算器*1	6	Yes
E	175.0	174.058	1,582,719	DSP	2	ALU*1, 乗算器*1	5	Yes

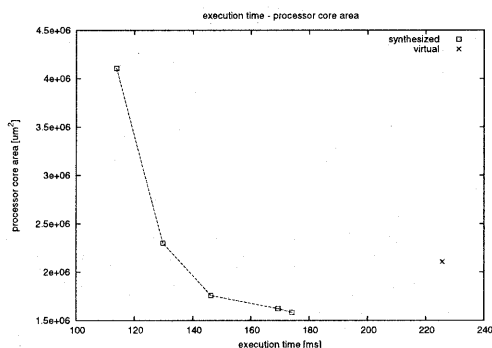


図 7 動作ソフトウェアの総実行時間とプロセッサコアの面積

のときに制約を満たせなくなった。

これはアセンブリコードの命令並列度が上がり、少ないステップ数で実行されるソフトウェアを持つ解にとって、ハードウェア IP の応答時間の占める割合が大きくなり、これがボトルネックになったためと考えられる。

以上より、一般的な汎用 RISC プロセッサよりも面積の小さいプロセッサを合成、かつ動作ソフトウェアを高速に動作させること、プロセッサコアの面積とソフトウェア実行時間をトレードオフとした柔軟な解が得られることが示された。また、ハードウェア IP の応答時間を考慮することで、ハードウェア IP の性能に対して無駄な性能を持つプロセッサを解としないことも示された。システムに対して性能に過不足のないプロセッサを合成するという提案フレームワークは有効性を示せたと考える。

6. む す び

ハードウェア IP の応答時間を考慮したプロセッサコア合成システムと同システムを利用したシステム LSI 設計のフレームワークを提案した。提案フレームワークを利用することでシステムに対して性能に過不足のないプロセッサを合成することが可能になる。今後の課題としては、DMA コントローラやマルチプロセッサ等、システムアーキテクチャを変えた場合の、本手法の拡張を含めた検討が考えられる。

謝辞 本研究を進めるにあたり、有用な議論、討論をいただいた(株)日立製作所 鈴木敬博士、同 荒宏視氏に感謝いたします。本研究に関して、御協力いただいた本学 鈴木伸治氏に感謝いたします。

文 献

- [1] H. Akaboshi and H. Yasuura, "COARCH: A computer aided design tool for computer architects," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol.E76-A, no.10,

pp. 1760-1769, 1993

- [2] ARM, Ltd., ARM7TDMI, <http://www.arm.com/>
- [3] N. N. Binh, M. Imai, A. Shimoi, and N.Hikichi, "A hardware/software partitioning algorithm for designing pipelined ASIPs with least gate count," In *Proc. the 33rd DAC*, pp. 527-532, 1996.
- [4] ISO/IEC, International Standard DIS 10918, Digital Compression and Coding of Continuous-Tone Still Images.
- [5] 岩下 洋哲, 中田 恒夫, 古渡 聡, 鈴木 敬, 荒 宏視, 矢野 和男, "インタフェース仕様記述言語 CWL とその応用," 第 15 回回路とシステム (軽井沢) ワークショップ, pp. 305-310, 2002.
- [6] NEC, 信号処理 LSI (DSP/音声) データブック, 1996
- [7] Open SystemC Initiative, SystemC <http://www.systemc.org/>
- [8] 桜井 崇志, 戸川 望, 柳澤 政生, 大附 辰夫, "2 種類のレジスタファイルを持つデジタル信号処理向けプロセッサのハードウェア/ソフトウェア分割手法," 信学技報, VLD99-76, ICD99-205, FTS99-54, 1999.
- [9] J. Sato, A. Y. Alomary, Y. Honma, T. Nakata, A. Shiomi, N.Hikichi, and M. Imai, "PEAS-I: A hardware/software codesign system for ASIP development," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E77-A, no. 3, pp. 483-491, 1994.
- [10] 田川 博規, 小原 俊逸, 戸川 望, 柳澤 政生, 大附 辰夫, "ハードウェア IP の応答時間を考慮したプロセッサコアのハードウェア/ソフトウェア分割手法," 信学技報, VLD02, 2003.
- [11] N. Togawa, M. Yanagisawa, and T. Ohtsuki, "A hardware/software cosynthesis system for digital signal processor cores," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E82-A, no.11, 1999.
- [12] Xilinx, Inc., 2-D Discrete Cosine Transform V2.0, http://www.xilinx.com/ipcenter/catalog/logiccore/docs/da_2d_dct.pdf