システムＬＳＩ設計技術　108－17
(2003. 1. 29)

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

田川　博規[†]　小原　俊逸[†]　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: tagawa@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ハードウェア/ソフトウェア分割手法を提案する．我々は対象とするアプリケーションに応じて利用するハードウェアIPを始めに決定した上で，機能・性能に過不足のないプロセッサコアを合成するシステムLSI設計アプローチを提案している．そこで適切な構成をもったプロセッサコアを合成するためには，ハードウェアIPの応答時間を考慮したハードウェア/ソフトウェア分割が有効である．提案手法はハードウェアIPの応答時間を命令レベルで考慮することで既存手法を拡張しており，これによりプロセッサコアとハードウェアIPが独立したタスクを効率良く並列実行することが可能となる．計算機実験により提案手法を評価し，本設計アプローチの有効性を示す．

キーワード　プロセッサコア，ハードウェアIP，ハードウェア/ソフトウェア分割，応答時間

A Hardware/Software Partitioning Algorithm for Micro Processors Based on Response Time of Hardware IPs

Hiroki TAGAWA[†], Shunitsu KOHARA[†], 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: tagawa@yanagi.comm.waseda.ac.jp

Abstract　This paper proposes a hardware/software partitioning algorithm based on response time of hardware IPs. We have been developing a new design approach which first determines the hardware IPs, then co-synthesizes a processor core. Our approach realizes an application-specific system LSI including the processor core that contains only the necessary functionalities. We can reduce an unnecessary functionalities by hardware/software partitioning for micro processors based on response time of hardware IPs. Our algorithm obtains hardware response time of hardware IPs at instruction level. That realizes the efficient parallel execution of hardware and software. The experimental results show effectiveness of the proposed algorithm and our new design approach.

Key words　processor core, hardware IP, hardware/software partitioning, response time

1. まえがき

近年，1チップに集積可能なトランジスタ数の増加により，システムLSIの設計規模は増加の一途を辿っている．これに伴い設計期間は増大する一方で，市場から要求されるLSI製品化の期間は短くなる傾向にある．この問題を解決するために，対象とするアプリケーションに応じてシステムLSIのハードウェア部分とソフトウェア部分を同時に設計するハードウェア/ソフトウェア協調設計[1],[5]や，過去に設計されたLSIであるIP（Intellectual Property）を再利用するIPベース設計などの設計技術が提唱・実現化されている．

さらに，これら2つの設計技術を組み合わせたものとして，アプリケーションを実現するソフトウェア部分とハードウェア部分を利用可能なIPを用いて最適に分割するLSI設計手法があり，実用化の進んでいるシステムレベルのハードウェア/ソフトウェア協調合成システムでは，複数のプロセッサコアIPとハードウェアIPのライブラリを利用して最適なシステム構成を決定する．その際，ハードウェアの一部は自動合成することも可能である．

ハードウェア/ソフトウェア協調設計，IP再利用の自動化まで含めたシステムLSI設計手法を用いれば，既存プロセッサを用いても短い設計期間でシステム全体の性能・コストに無駄のない設計が期待できる．しかし現実的には利用できるIPライブラリが乏しく，柔軟な設計解をとれないため，プロセッサコアIPの機能・性能の過不足をハードウェアIP側で吸収できず，システム全体で無駄が生じる場合があると考えられる．また，システム全体の機能・性能の過不足から，新規にハードウェア設計が発生した場合，ソフトウェアの再設計や，システム全体の再検証などの工程が複雑な手順で発生するという問題がある．

我々は，前述の設計問題を，文献[2],[3],[8],[10]などで提案されている，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技術を用いることで解決するアプローチを提案する．すなわち，適当なターゲットアーキテクチャおよび対象とするアプリケーションに対して，まず始めに再利用するハードウェアIPを決定した上で，残りのプロセッサコア部分および，その上で動作するソフトウェアを新規に自動合成することを試みる．これにより，利用可能なハードウェアIPの離散的な性能・コストによる無駄を柔軟にプロセッサコア側で吸収すると同時に，システム設計過程において発生する設計工程をソフトウェア設計に集約することが期待できる．

我々の提案する，プロセッサコア合成システムを図1に示す．第4章で提案するハードウェア/ソフトウェア分割手法は図中の**プロセッサコアHW/SW自動分割系**における処理アルゴリズムである．本システムは，アプリケーションプログラムのSystemC記述[13]，総実行時間制約およびハードウェアIPインタフェース記述を入力とする．合成結果としてプロセッサコアのHDL記述と，そのプロセッサコア上で動作するオブジェクトコードを出力する．

本稿ではプロセッサコアから見たハードウェアIPのデータ処理時間を**応答時間**と定義し，ハードウェアIPの応答時間を考慮したハードウェア/ソフトウェア分割手法を提案する．

提案手法は，まずハードウェアIPをプロセッサコアから見て固有の応答時間を持った命令により操作できるようなインタフェースモデルを採用し，文献[9],[11]の分割手法を基に，ハードウェアIPの応答時間を考慮すると同時に，性能低下の要因となるハードウェアIPのビジー待ち状態を回避するようなスケジューリングを導入することで実現される．これにより従来手法をハードウェアIPに対応化すると共に，

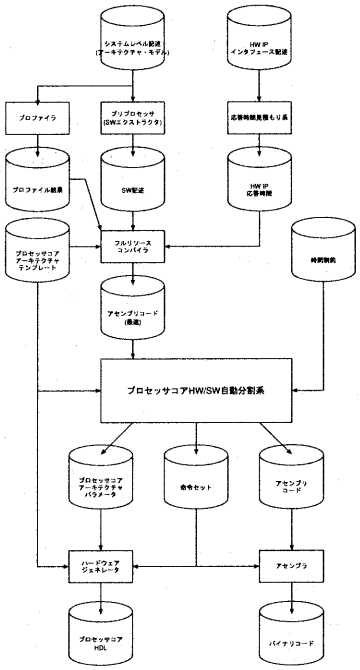

図1　プロセッサコア合成システム

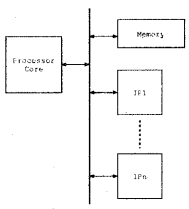

図2　システムアーキテクチャモデル．

プロセッサコアとハードウェアIPが独立したタスクを効率良く並列実行することが可能となる．

本稿は以下のように構成される．2章で図1の合成システムが設計対象とするシステムLSIのアーキテクチャモデルを示す．3章では提案するハードウェア/ソフトウェア分割手法の対象となるプロセッサコアのターゲット・アーキテクチャを示す．4章では応答時間を考慮したハードウェア/ソフトウェア分割手法を提案し，5章では計算機実験結果から，本分割手法と提案システムの有効性を論じる．6章で本稿をまとめる．

2. システムのアーキテクチャモデル

図2に図1の設計対象となるシステムLSIのアーキテクチャモデルを示す．本アーキテクチャは，(1) プロセッサコア，(2) プロセッサコアとハードウェアIP間のインタフェースおよび(3) ハードウェアIPの3点から構成されると定義する．

2.1 プロセッサコア

図1の合成対象となるプロセッサコアであり，以下に示すハードウェアIPとのインタフェースを持つ．3章でそのアーキテクチャモデルと命令セットを詳細に示す．

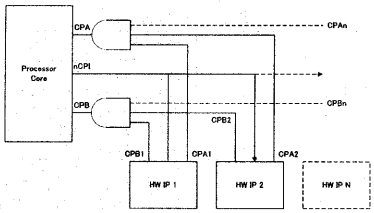

図 3　ハードウェア IP との接続.

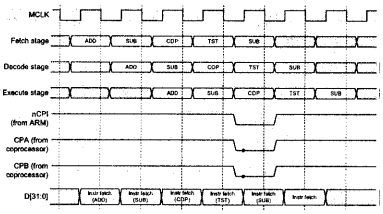

図 4　データ操作命令 CDP

2.2　インタフェース

合成対象のプロセッサコアとハードウェア IP 間のインタフェースは ARM7TDMI [12] のコプロセッサインタフェースに準じたものとして定義する.

AMBA [12] 等の標準バスインタフェースは，ハードウェア IP からの処理終了を通知する割り込みや，共有メモリへのアクセスによるバスの競合の発生を前提とした仕様である．4 章で提案するプロセッサコアのハードウェア/ソフトウェア分割手法において，ハードウェア IP のデータ演算時間・データ転送時間をプロセッサコアの命令レベルで，ハードウェア IP の応答時間として正確に把握することが必須である．そのため，ハードウェア IP をプロセッサコアからの単純な制御のみで扱えるインタフェースを採用している.

2.2.1　シグナルインタフェース

プロセッサコアはハードウェア IP と図 3 に示すように接続され，最大 16 個のハードウェア IP が接続可能である．以下の 3 種類のハンドシェイク信号を用いて通信することで，後述のハードウェア IP 命令の実行制御を可能にしている.
(1)nCPI(プロセッサコアから全ハードウェア IP へ) : プロセッサコアが，あるハードウェア IP 命令を実行したいことを通知する．(2)CPA (ハードウェア IP からプロセッサコアへ) : ハードウェア IP 命令を実行することができるハードウェア IP が存在しないことをプロセッサ側へ通知する.
(3)CPB (ハードウェア IP からプロセッサコアへ) : ハードウェア IP が処理中のため，まだハードウェア IP 命令を実行できないことを通知する.

2.2.2　命令インタフェース

プロセッサコアはハードウェア IP を対象に (a) データ操作命令 CDP，(b) データ転送命令 LDC/STC および (c) レジスタ転送命令 MCR/MRC の 3 種類の命令を発行する．以下本稿ではこれら 3 種類の命令を**ハードウェア IP 命令**と呼ぶ．命令形式を以下に示す.

```
CDP HW#, OP#

LDC|STC HW#, N, Rd, Rn, offset

MRC|MCR HW#, Rd1, Rd2
```

(a) **データ操作命令**は HW#で指定されるハードウェア IP 内部で実行され，ハードウェア IP 内部のレジスタに状態変化を発生させる．処理内容は OP#によって指定される．図 4 に CDP が実行される様子を示す [12]．CDP 命令がデコードされるとプロセッサコアは nCPI を LOW にし，それがハードウェア IP 命令であることを知らせる．同時にこの CDP 命令の HW#に対応したハードウェア IP からこの処理を実行可能かつ処理中ではないことを，それぞれ CPA，CPB を LOW にすることで知らせる．ハードウェア IP はこの処理を開始すると共に CPA，CPB を HIGH にすることで制御をプロセッサコアに返す．これによりプロセッサコアは次サイクルからは通常命令の実行に移ることができる.

(b) **データ転送命令**は HW#で指定されるハードウェア IP に，レジスタの値を共有メモリへロード/ストアさせる．N で指定されたワード数の連続転送が可能である．図 5 に LDC が実行される様子を示す．HW#に対応したハードウェア IP がデータ転送を開始するまでの制御は CDP と同様であるが，連続データ転送の間，プロセッサコア側がアドレス生成をする必要があるため，転送中はプロセッサコアは後続命令の実行に移れない．転送が終わるとハードウェア IP は CPA，CPB を HIGH にすることで制御をプロセッサコアに戻す．この方式によるロード/ストアでは，全てのハードウェア IP のデータ転送がプロセッサコアによって制御されているため，通常のシステム LSI に見られるようなバスの競合は起き得ない.

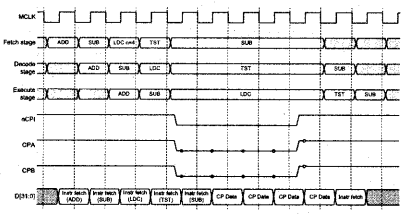

図 5　データ転送命令 LDC

(c) **レジスタ転送命令**はで指定されるハードウェア IP のレジスタ Rd1 とプロセッサコアのレジスタ Rd2 間の転送を行う．通常は 1 サイクルで処理を完了し，制御をプロセッサコアに戻す.

2.3　ハードウェア IP

ハードウェア IP は，固有のデータ処理を行うデータパス，レジスタバンク以外に，(1) 命令パイプライン，(2) 命令デコードロジックといった機能を持つものとする．これは，ハードウェア IP がプロセッサコアのパイプラインを流れる命令列を共有し，前述のハンドシェイク信号を生成，解釈するために必要だからである．一般のハードウェア IP はこれらの機能を全て持たないが，本稿においてはこれらの機能をハードウェア IP が全て持つ，あるいは中間にこれらの機能を持つラッパーがあると仮定して議論を進める.

2.4　ビジー待ち状態によるストールの発生

CDP によって既に処理中のハードウェア IP に対してさらに CDP，LDC/STC，MCR/MRC を発行すると，待ち状態が発生し，パイプラインがストールする．図 6 にその様子を示す．CDP のデコードが終わると，HW#に該当するハードウェア IP 自体は存在するので CPA は LOW になるが，この場合該当ハードウェア IP が未だ処理中のため，CPB は HIGH のままである．この場合 3 サイクル後に，その処理が終了したためハードウェア IP は次の CDP の処理に移り，制御をプロセッサコアに戻している．このビジー待ち状態の

発生は，直接性能の低下に繋がるので注意が必要である．

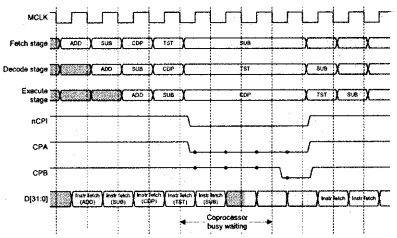

図 6　ビジー待ち状態

3. プロセッサコアのアーキテクチャモデル

4 章で提案するハードウェア/ソフトウェア分割手法が対象とする，プロセッサコア内部の演算器構成のモデルと命令セットを示す．本アーキテクチャモデルは，文献 [9] で提案されたモデルに基づいている．

3.1 アーキテクチャモデル

プロセッサのターゲットアーキテクチャとして，RISC アーキテクチャを持つ汎用のマイクロプロセッサからディジタル信号処理プロセッサにわたるものを考える．図 7 にプロセッサカーネルおよびプロセッサカーネルに付加されるハードウェアユニットの一部を示す．プロセッサカーネルに，いくつかのハードウェアユニットが付加されたものをプロセッサコアと呼ぶ．

3.1.1 プロセッサカーネル

プロセッサカーネルは，RISC 型および DSP 型のうちいずれかを取る．RISC 型は，IF（命令フェッチ），ID（命令デコード），EXE（命令実行），MEM（メモリアクセス），WB（書き込み）の 5 ステージ・パイプライン構成をとる．一方 DSP 型は，IF，ID，EXE の 3 ステージ・パイプライン構成をとる．DSP 型カーネルは RISC 型カーネルと比較して動作が低速になる反面，マルティメディア処理に有効なアドレッシングユニットおよびハードウェアループを付加することが可能である．各カーネルタイプの詳細なアーキテクチャは文献 [9] に基づく．また，図 3 に示したようなハードウェア IP 間とのハンドシェイク信号を生成する機構を持つ．

3.1.2 ハードウェアユニット

プロセッサカーネルに付加できるハードウェアユニットは，(1) 専用演算器，(2)Y データメモリバス，(3) レジスタファイル，(4) アドレッシングユニットおよび (5) ハードウェアループユニットである．これらはいずれもコア内部のハードウェアであり，本研究の対象となっているハードウェア IP とは明確に区別される．

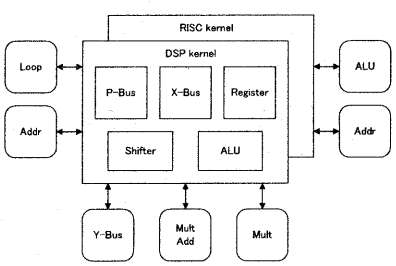

図 7　プロセッサカーネルおよび付加されるハードウェアユニット

表 1　基本命令．

算術論理命令	ADD, SUB, SRA, SRL, SLL, AND, OR, XOR, MUL, DIV, SLT, SEQ, SNE, COM2, MAC, INC, DEC, ADDI, SUBI, SRAI, SRLI, SLLI, ANDI, ORI, XORI, MULI, DIVI
ロード/ストア命令	LDX, LDY, STX, STY, LDRX, LDRY, STRX, STRY, LDXI, LDYI, STXI, STYI, LDIX, LDIY, STIX, STIY, MV, IMM
ジャンプ命令・他	BEQ, BNE, BZ, BNZ, JP, LOOP, RPT, CALL, RET, NOP, HLT
並列ロード/ストア命令	LDPX, STPX

3.2 命令セット

合成されるプロセッサコアは，命令セットとして (1) 基本命令，(2) 複合命令および (3) ハードウェア IP 命令を持つ．

(1) は，市販のディジタル信号処理プロセッサを基本としており [7]，プロセッサ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．プロセッサが持つ基本命令を表 1 に示す．(2) 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，アプリケーションプログラムに応じて複合命令となる命令の組合せを決定する．(3) は 2 章で定義したように，ARM7TDMI コプロセッサ命令に準じた，ハードウェア IP を対象に発行される命令である．

(1)，(2) は 1 サイクルで実行される．(3) は 2 章で示した通り，CDP/MCR/MRC に関してはプロセッサコアから見て 1 サイクルで処理が完了する．LDC/STC は転送ワード数に応じたサイクル数だけ後続命令の実行が不可能になる．

4. 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

2，3 章で示したアーキテクチャモデルを対象としたハードウェア/ソフトウェア分割手法を提案する．提案手法は，文献 [9], [11] の分割手法を基に，(1) ハードウェア IP 命令を導入したことによる発行可能タイミング制約および (2) ビジー待ち状態の回避，の 2 点を考慮した命令スケジューリング，を導入することで実現される．これにより，従来手法をハードウェア IP 命令に対応化させることが可能になると共に，プロセッサコアとハードウェア IP が独立したタスクを効率良く並列実行することができる．

本章では，まず文献 [9], [11] で提案されているハードウェア/ソフトウェア分割問題およびアルゴリズムを紹介し，ハードウェア IP 命令を導入するために新規に拡張した部分について詳細に述べる．

4.1 ハードウェア/ソフトウェア分割問題

アセンブリコードは文献 [11] と同様のグラフ形式によって表現される．**コールグラフ**とは，アプリケーションプログラムに含まれる関数間の呼び出し関係を表したグラフである．コールグラフの各節点には，コントロールフローグラフが対応する．**コントロールフローグラフ**とは，各関数内部における制御の流れを表すグラフである．コントロールフローグラフの各節点には，データフローグラフが対応する．**データフローグラフ**とは，各基本ブロックのデータの流れを表す

入力: 初期スケジューリング済みアセンブリコード，時間制約，実行プロファイル
出力: アセンブリコード，プロセッサコアアーキテクチャ，命令セット

Step1. 初期アロケーションによって決定したプロセッサカーネルに付加されている 1 つのハードウェアユニット hu に対して，削減を試みる．

Step2. Step1 で試みた削減に対応する，プロセッサコアの面積/遅延および総実行サイクル数を算出し，遅延と総実行サイクル数の積からアプリケーションプログラムの総実行時間を導出する．

Step3. Step1, Step2 の処理をプロセッサカーネルに付加されるハードウェアユニットの各々に対して適用したとき，時間制約を満たす中で hu の削減による実行時間増加率 $T_{rate}(hu)$ が最小となるような hu を実際に削減する．ただし，時間制約を満足するハードウェアユニットが存在しなければ終了する．削減後のプロセッサコアアーキテクチャとアセンブリコードをそれぞれ入力されたプロセッサコアアーキテクチャ，アセンブリコードとして Step1 へ戻る．

図 8 プロセッサコアアーキテクチャの決定アルゴリズム．

グラフである．データフローグラフの各節点は，1 つの基本命令あるいはハードウェア IP 命令に対応する．

ハードウェア/ソフトウェア分割問題は，入力として (1) 初期スケジューリングされたアセンブリコード，(2) 基本ブロックの実行回数，(3) アプリケーションの総実行時間制約，(4) プロセッサコア内部のハードウェアユニットを持つ．出力として (1) アセンブリコード，(2) プロセッサアーキテクチャおよび (3) 命令セットを出力する．時間制約の下でプロセッサコア面積の最小化を目的とする．

4.2 ハードウェア/ソフトウェア分割アルゴリズム

文献 [9], [11] で提案されている分割アルゴリズムは (1) **初期リソースアロケーション**および (2) **プロセッサコアアーキテクチャ決定**の 2 つの手続きからなる．(1) は並列化コンパイラより入力として与えられたアセンブリコードを実行するためにプロセッサコアが必要とするハードウェアユニットの個数と種類を決定する処理である．(2) 初期リソースアロケーションによって得られたプロセッサアーキテクチャに対し，図 8 の手順に従ってハードウェアユニットの削減およびアセンブリコードの再構成をする．

4.3 ハードウェア IP 命令の分割手法への導入

ハードウェア IP 命令を前述の分割手法に導入することを考える．ハードウェア IP 命令の実行のために，プロセッサコアはどのハードウェアユニットも必要としない．そのため初期リソースアロケーションの処理内容に拡張すべき部分はない．一方，プロセッサコアアーキテクチャ決定に関しても，図 8 の Step1,Step3 については文献 [9], [11] と同様の手法を用いることで対応する．これもハードウェア IP の存在が，プロセッサコアのハードウェアユニット削減処理と直接関係がないからである．拡張する部分は，Step2 における処理に関してであり，**面積/遅延見積もり式の拡張**と**ハードウェア IP の応答時間を考慮した命令スケジューリング**の 2 点がある．

4.3.1 面積/遅延見積もり式の拡張

図 8 の Step2 において hu の削減に対応した面積/遅延見積もりは，文献 [4] の見積もり手法を適用することで実現される．ただし，ハードウェア IP 命令を導入するにあたって，ハードウェア IP とのインタフェース部分の増分を面積/遅延見積もり式に新たに考える必要がある．ハードウェア IP 命令の導入による (1) 命令デコード部 (2) ハンドシェイク信号生成部および (3) ビジー状態に対応したパイプラインストール機構について，実際に文献 [4] で提案される基準プロセッサに VHDL 記述を追加，論理合成することで，面積増分を見積もった．

4.3.2 ハードウェア IP の応答時間を考慮した命令スケジューリング

図 8 の Step2 において，hu の削減に対応した総実行サイクル数を算出するため，アセンブリコードを再スケジューリングする．これは各基本ブロックのリストスケジューリングによって実現する．ここでリストスケジューリングとは以下のような処理である．

(1) 基本ブロックの先頭ステップを対象に，まず基本ブロック中の全命令ノードのうち，親ノードのないノードを全て，このステップにスケジュールされる候補集合に入れる．(2) 候補集合の中からノード番号順に**スケジュール可能判定**をし，可能と判定された場合は実際にこのステップにスケジュールし，不可能と判定された場合は次ステップの候補集合に入れる．(3) 全てのノードが判定されたら親ノードを持たなくなったノードを次ステップの候補集合に加える．(4) 次ステップに関して同様の処理を繰り返す．

ハードウェア IP 命令を導入するためには，(1) ハードウェア IP 命令の応答時間特性によるスケジュール可能タイミング制約 (2) ビジー待ちによるストール発生の回避という 2 つの条件を満たすようなリストスケジューリングをする必要がある．(1) は 2 章で説明したハードウェア IP 命令の応答時間特性を考慮してスケジューリングすることで従来通り図 8 の Step2 で正しく総実行時間を見積もるために，(2) は性能の低下を避けるためにそれぞれ必要である．

ハードウェア IP の応答時間は，応答時間見積り系により正確に見積もられており，ハードウェア IP 命令ノード v の出力エッジ長としてハードウェア/ソフトウェア分割系に与えられていると仮定している．そのため，前述のリストスケジューリングにおけるスケジュール可能性判定の処理に図 9 のアルゴリズムを適用することで (1)(2) の条件を満たすスケジューリングが可能である．ここで図 9 の HUT(Hardware IPs Usage Table) はハードウェア IP の数に対応して最大 16 個の要素を持つ，各ステップにおけるハードウェア IP の使用状況を表す管理テーブルである．各要素は (a) ハードウェア IP 番号 `HW#`，(b) 現在処理中のハードウェア IP 命令タイプ（使用されていな場合は OFF）(c)(b) の命令の発行ステップ番号 s(d) 応答時間 t といった情報を持つ．

図 9 の Step1，Step2 の判定条件が (1) の条件に対応し，Step3 の判定条件が (2) の条件に対応している．

ここで表 2 および表 3 は，2 章で定義したインタフェースによる命令発行上の制約を表している．

表 2 はスケジュール対象となっているステップに既に列先頭要素の命令がスケジュールされている場合，各行先頭要素の命令がこのステップにスケジュール可能 (○) であるか不可能である（×）かを示している．表 3 はスケジュール対象となっているステップにおいて列先頭要素が処理中の命令である場合，各行先頭要素の命令がこのステップにスケジュール可能であるかどうかを示している．

5. 計算機実験結果

提案手法は C 言語を用いて実装されている．提案手法を 2 次元 DCT(8×8) の機能を持つハードウェア IP を利用することを前提に JPEG エンコードアプリケーション（680×480

入力: 候補集合に含まれる1つの命令ノード v, スケジュール対象のステップ番号 i,
出力: v のスケジューリング判定結果

Step1. 表2の v の命令タイプに対応した行を参照し，×である命令タイプが既に i 番目のステップでスケジュールされているか判定する．つまり×である命令タイプが HUT の $s=i$ である要素中の命令タイプに存在するかを判定する．

Step2. 表3の v の命令タイプに対応した行を参照し，×となる命令タイプが既に実行中であるか判定する．つまり HUT の全要素中に1つでも該当する命令タイプを持った要素がないかを判定する．

Step3. v がハードウェア IP 命令の場合，HUT を参照し，v の対象となる **HW#** のハードウェア IP が処理中であるか判定する．つまり HUT 中の **HW#** で指定される要素に注目し，$s+t>i$ であるか判定する．

Step4. Step1〜Step3 の判定結果が全て偽だった場合実際に i 番目のステップに v をスケジュール可能と判定し，HUT を更新する．それ以外なら v をスケジュール不可能と判定し，HUT を更新後次のノードの判定に移る．

図9 スケジュール可能判定アルゴリズム

表2 同一ステップに関する制約

	基本命令	CDP	LDC/STC	MCR/MRC
基本命令	○	×	×	×
CDP	×	×	×	×
LDC/STC	×	×	×	×
MCR/MRC	×	×	×	×

表3 処理中の命令があるステップに関する制約

	CDP	LDC/STC	MCR/MRC
基本命令	○	×	×
CDP	○	×	×
LDC/STC	○	×	×
MCR/MRC	○	×	×

pixel× 8bit）に適用した．

JPEG エンコードアプリケーションに実行時間制約を与えた時，提案手法を適用した場合のプロセッサコアの面積，総実行時間および内部ハードウェア構成を表4に示す．表4の各分割解に対応するアセンブリコードにおいて，ハードウェア IP の処理中もプロセッサコアはソフトウェア実行を並列で行い，かつバスの競合やパイプラインストールの発生を避ける命令スケジューリング結果であることが確認できた．

また，実験結果の合成プロセッサを既存のプロセッサコアで置き換えた場合と比較するために，仮想的な RISC プロセッサとして，5段パイプライン，32bit 汎用レジスタ 32 本，命令長 32bit，乗算命令を持つを持つプロセッサコアを，本分割手法が対象とするものと同様なプロセッサ・アーキテクチャモデル上で実現し，同様な JPEG アプリケーションおよび DCT のハードウェア IP に関して面積と総実行時間を見積もった．面積/遅延見積もりは，本分割手法での方法と全く同様の方法を用いた．その結果，プロセッサコア面積は 2,107,831[μm^2]，総実行時間は 225.621[ms] であった．これと表4を比較することにより，本分割手法の結果，汎用のプロセッサコアと比較して，同面積規模で高性能なプロセッサコア構成が得られている．従って，我々の提案する設計アプローチの有効性を示せたと考える．

表4 実験結果（JPEG エンコード）

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
120.0	113.740	4,106,896	RISC	4	ALU*2, 乗算器*2	20
130.0	129.799	2,298,954	RISC	4	ALU*1, 乗算器*1	5
150.0	146.212	1,758,458	RISC	2	ALU*1, 乗算器*1	6
170.0	169.273	1,623,141	DSP	2	ALU*1, 乗算器*1	6
175.0	174.058	1,582,719	DSP	2	ALU*1, 乗算器*1	5

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におけるハードウェア/ソフトウェア分割手法を提案した．今後の課題として，割り込みまでを含んだシステムのアーキテクチャモデルに対して，応答時間見積り系との連携による本手法の適用可能性を検討することが挙げられる．

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．本論文作成に御協力いただいた本学 太刀掛 宏一氏に感謝いたします．

文 献

[1] H. Akaboshi and H. Yasuura, "COACH: 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] N. N. Bình, M. Imai, A. Shiomi and N. Hikichi, "A hardware/software partitioning algorithm for designing pipelined ASIPs with least gate count," in *Proc. 33rd DAC*, pp. 527–532, 1996.

[3] I. J. Huang and A. M. Despain, "Synthesis of instruction sets for pipelined microprocessors," in *Proc. 31st DAC*, pp. 5–11, 1994.

[4] 片岡 義治, 吉澤 大, 戸川 望, 柳澤 政生, 大附 辰夫, "ディジタル信号処理向けプロセッサコアの面積/遅延見積もり手法," 信学技報, VLD99-2, ICD99-204, FTS99-53, 1999.

[5] H. Liu and D. F. Won, "Integrated partitioning and scheduling for hardware/software codesign," in *Proceedings of the InternationalConference on Conputer Design*, 1998.

[6] Y. Miyaoka, Y. Kataoka, N. Togawa, M. Yanagisawa, and T. Ohtsuki, "Area/delay estimation for digital signal processor cores," in *Proc. ASP-DAC2001*, pp. 156–161, 2001.

[7] NEC, 信号処理 LSI(DSP/音声) データブック, 1996.

[8] 大槻 典正, 竹内 良典, 今井 正治, 浜口 清治, 柏原 敏伸, 引地 信之 "VLIW プロセッサ自動生成における演算器構成最適化の一手法," 信学技報, VLD101–108, 1997.

[9] 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ィ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.

[10] 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.

[11] N. Togawa, T. Sakurai, M. Yanagisawa and T. Ohtsuki, "A hardware/software partitioning algorhithm for processor cores of digital signal processing," in *Proc. Asia and South Pacific Design Automation Conference* , pp. 335–338, 1999.

[12] http://www.arm.com/

[13] http://www.systemc.org/