社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2009-SLDM-138 (26)
2009 / 1 / 30

命令メモリアクセス数削減に基づく低エネルギーASIP合成手法

小林 優太† 戸川 望† 柳澤 政生† 大附 辰夫†

† 早稲田大学大学院基幹理工学研究科情報理工学専攻
〒169-8555 東京都新宿区大久保3-4-1
E-mail: †yuuta@ohtsuki.comm.waseda.ac.jp

あらまし 本稿ではVLIW型ASIPを対象としたハードウェア/ソフトウェア（HW/SW）ASIP協調合成システムSPADESにおける消費エネルギー削減手法を提案する．ASIPにおいて命令メモリが占める消費エネルギーの割合は大きく，命令メモリの消費エネルギー削減が課題となっている．そこで我々は，1命令に逐次実行される複数の命令を格納する垂直結合命令を考え，垂直結合命令探索手法を提案する．提案する垂直結合命令探索手法はスケジューリング済みCDFGから消費エネルギー削減量が最大となるように垂直結合命令を決定する．これにより命令メモリのアクセス数を削減し，消費エネルギーを削減することができる．計算機実験により，メモリを含むプロセッサ全体で平均41.9%の消費エネルギー削減を確認した．

キーワード 消費エネルギー，ASIP，VLIW，命令メモリ，命令フェッチ，命令セット

A Low Energy ASIP Synthesis Method Based on Reducing Instruction Memory Access

Yuta KOBAYASHI†, Nozomu TOGAWA†, Masao YANAGISAWA†, and Tatsuo OHTSUKI†

† Dept. of Computer Science and Engineering, Waseda University
3-4-1 Okubo, Shinjuku-ku, Tokyo, 169-8555 Japan
E-mail: †yuuta@ohtsuki.comm.waseda.ac.jp

Abstract In this paper, we propose an energy-efficient ASIP synthesis method based on reducing instruction memory access. Since an instruction memory is one of the main energy consumers in ASIP, reducing consumed energy in instruction memory is an important problem. We propose a vertical combined instruction that stores two or more instructions issued sequentially into a single instruction. Then we propose a method to synthesize the vertical combined instructions from a scheduled CDFG. Since the number of instruction memory accesses is reduced, the energy consumption can also be reduced. In experimental results, we confirm reducing approximately 41.9% energy consumption at a whole processor system including memories.

Key words energy consumption, ASIP, VLIW, instruction memory, instruction fetch, instruction-set

1. まえがき

携帯電話やデジタルカメラなどの組み込みシステムに搭載されるSoC（System-on-a-Chip）は，小面積・高速処理が要求される．これらの要求を満たすために，ターゲットとなるアプリケーションに特化した命令を持つプロセッサであるASIP（Application Specific Instruction-set Processor）が多く使用されるが，ASIPは各アプリケーションに対して個別設計が必要となるので設計時間，設計費用などの設計コストが増大してしまう．この問題の解決策としてASIP自動合成手法が考えられる．ASIP自動合成手法にはASIP Meister [5], Xtensa [15], MeP [20]などの研究がされており，我々はSIMD型演算を対象としたプロセッサコア向けHW/SW協調合成システムSPADES [17], [18], [19]を提案している．SPADESは，アプリケーションプログラムとアプリケーション実行時間制約を入力とし，実行時間制約を満たす中で面積最小のプロセッサコアを出力する．このようにASIP自動合成手法によって小面積・高速処理を実現するプロセッサを設計コストを抑えて設計することができる．

一方で，携帯機器はバッテリ駆動や発熱の問題から低消費エネルギーであることも要求されるため，ASIPの消費エネルギー削減は重大な課題となっている．よって，HW/SW協調合成システムにおいても消費エネルギーを削減する手法が必要である．ASIPの低消費エネルギーを実現するアーキテクチャの1つにVLIWプロセッサがある．VLIWプロセッサはコンパイラが並列実行可能な命令を同時発行する．これにより1回のフェッチで複数演算を処理し，少ない実行サイクル数でアプリケーションを処理できるので，動作周波数を抑えられ消費エネルギーを削減できる．しかし，同時発行する命令数に比例して，命令メモリのビット幅が長くなってしまい，命令メモリ自

体の消費エネルギーが増大してしまう．ASIP における命令メモリの消費電力は約 30%を占めるという報告 [2] もされており，VLIW 型 ASIP における命令メモリの消費エネルギー削減が課題となっている．そこで本稿では，命令メモリの消費エネルギー削減手法に注目する．

メモリの消費エネルギー削減には，メモリ容量削減とメモリアクセス数削減の 2 つのアプローチが考えられる．命令メモリの容量削減に着目した手法には，文献 [1], [4], [8], [10], [11], [21] などがある．文献 [21] は MIPS プロセッサに対してはハフマン符号を用いることで命令コードの圧縮をしている．同様の手法として PowerPC の CodePack [4] があり，デコーダをメモリとキャッシュ間に置くことで，外部メモリのコードを圧縮している．別のアプローチとして，ARM プロセッサの Thumb 命令 [1]，MIPS32/16 [10] がある．これらは通常の 32bit 命令長を使用する命令やレジスタを制限することで 16bit 命令に圧縮している．しかし，命令やレジスタの使用制限は実行時間の増大を招き，32bit 命令では 1 サイクルで処理可能な命令を 16bit 命令では複数サイクルに跨って処理する必要があるため，実行サイクル数の増大に繋がる [9]．文献 [1], [4], [10], [21] は VLIW プロセッサには対応していない問題がある．VLIW プロセッサを対象とした命令ビット幅削減手法には文献 [11] がある．文献 [11] では VLIW 命令の各スロットに対してコード圧縮をしたスロットと通常の命令長を使用するスロットを分けることで VLIW 命令長を削減している．しかし，文献 [11] は ASIP 自動合成手法ではない．ASIP 自動合成における手法としては文献 [8] がある．文献 [8] は命令長のオペコードは命令セット数に依存することに着目して，VLIW プロセッサに同時発行される複数命令の並びを 1 命令と見なす結合命令（結合オペコード）を用いることで命令長を削減している．さらに ASIP 自動合成のレジスタ数を変更できるという特徴とオペランド幅はレジスタ数に依存するという特徴から，大域レジスタと退避レジスタのメモリへのスピルを利用しレジスタ数を削減している．

メモリアクセス数に着目した研究には，文献 [13], [14], [16] がある．文献 [13], [14] の手法は 16bit 命令を 2 命令同時にフェッチする，rISA プロセッサを提案している．文献 [13], [14] はプログラム中のファンクション毎に，32bit 命令の通常モードと 16bit 命令を 2 命令格納した rISA モードの 2 モードを切り換えることで，32bit 命令の高速性を保ち，メモリアクセス数を削減している．しかし，これらの手法は ARM プロセッサの Thumb 命令 [1] や MIPS32/16 [10] 同様，16bit 命令に割り当てることのできる命令やレジスタ数が限定される問題や，VLIW プロセッサに対応していない問題がある．VLIW プロセッサにおける命令メモリアクセス数を削減する手法としては TMS320C6x [16] がある．TMS320C6x は各スロット間に 1bit の制御ビットを使用することで 1 回のフェッチで複数サイクルの命令を実行することができる．しかし，ASIP 自動合成については対応しておらず，アプリケーションによっては命令のビット幅が冗長になってしまう可能性がある．

以上のように ASIP の命令メモリにおける低消費エネルギーに関する研究は多くされているが，VLIW 型 ASIP 自動合成の研究として，命令メモリの容量とアクセス回数の両方に着目した研究はされていない．こうした背景から，本稿では VLIW プロセッサの命令メモリアクセス数削減に基づくエネルギー削減手法を提案する．本手法は文献 [8] をベースとし，オペコードに複数命令を処理する専用の命令を割り当て，NOP 命令のオペランドに次サイクル以降のオペランドを格納することで，1 回の命令メモリアクセスで複数サイクルの命令をフェッチする．これにより，文献 [8] の命令長削減を保ちつつ，命令メモリのアクセス数を削減できる．計算機実験において，SPADES が対象とする VLIW プロセッサに比べて 31.4%〜47.7%の消費エネルギー削減を確認した．

2. ターゲットモデル

本章では対象とするアプリケーションプログラムモデル，アーキテクチャモデル，およびエネルギーモデルについて説明する．

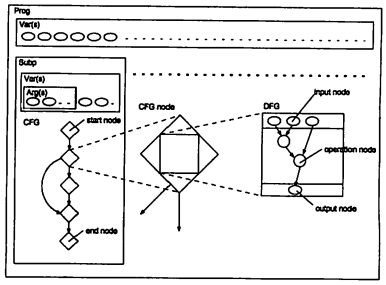


図 1　アプリケーションプログラムモデル

2.1 アプリケーションプログラムモデル

アプリケーションプログラムモデルは図 1 のような CoDaMa フレームワーク [7] に基づく．アプリケーションプログラム（Prog）は複数のサブプログラム（Subp）から構成され，それぞれに変数（Var）が割り当てられる．Prog に属する Var はグローバル変数，Subp に属する Var はローカル変数となる．各 Subp は 1 つの制御を表すグラフ（CFG）を持ち，CFG には開始ノードと終了ノードを 1 つずつ持つ．CFG のノードは基本ブロックを表し，基本ブロックはデータの流れを表すグラフ（DFG）で表され，入力ノード，演算ノード，出力ノードを持つ．オペランドは，汎用レジスタ，即値，データポインタを持つ．

2.2 プロセッサアーキテクチャモデル

アーキテクチャモデルは SPADES に基づく．SPADES におけるターゲットプロセッサはハーバードアーキテクチャに基づき，命令メモリとデータメモリを持つ．プロセッサコアは，プロセッサとして動作するための基本的な機能を持つプロセッサカーネルと，それに付加させるハードウェアユニットによって構成される．プロセッサコアの例を図 2 に示す．命令形式は複数命令を並列実行可能な VLIW 方式を用いる．

2.2.1 プロセッサカーネル

プロセッサカーネルのパイプライン段数は 3〜6 段で可変である．各パイプラインステージの役割は固定であり，5 段パイプラインの場合は [3] に基づき，命令フェッチ（IF），命令デコード（ID），実行（EXE），メモリアクセス（MEM），ライトバック（WB）で構成される．

また，プロセッサカーネルは 1 つ以上の命令を同時に発行可能である．並列に実行可能な命令数は入力として与えられ，固定である．プロセッサカーネルにはデータハザード検出用回路は含まれず，コンパイラ側でデータハザードを回避する必要がある．

2.2.2 ハードウェアユニット

ターゲットプロセッサにおけるハードウェアユニットとして，メモリバス，演算器，フォワーディングユニット，アドレッシングユニット，ハードウェアループユニット，およびレジスタファイルがある．プロセッサコアはプロセッサカーネルにハードウェアユニットを付加することで構成される．

2.2.3 命令セット

各 VLIW スロットで発行される命令を個別命令と呼び，その命令セットは [19] に基づいている．個別命令の集合を I，VLIW スロット数を S，VLIW コアのスロット k に発行される個別命令を $w^k \in I$ とすると VLIW 命令 $\mathbf{w}$ は式 (1) となる．

$$\mathbf{w} = \begin{bmatrix} w^1 \\ w^2 \\ \cdots \\ w^S \end{bmatrix} \tag{1}$$

アプリケーションプログラムの命令ストリーム W が N 個の VLIW 命令で構成されているとする（式 (2)）．

$$W = < \mathbf{w_1}, \cdots, \mathbf{w_{n-1}}, \mathbf{w_n}, \cdots, \mathbf{w_N} > \tag{2}$$

VLIW 命令を実現する方式として，図 3 (a) のように各スロットに対してオペコードを 1 つずつ持つ基本的な方式 [3] と，図 3 (b) のように同時に発行される個別命令を 1 つの命令と解

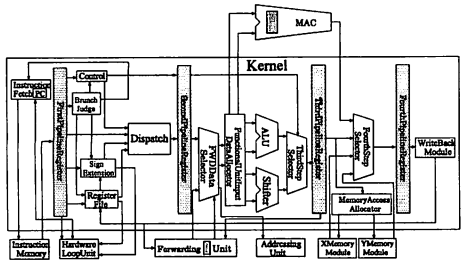


図 2 プロセッサコアの例

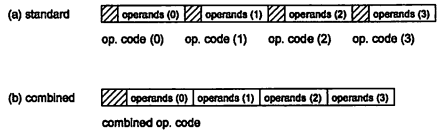


図 3 命令エンコーディング

表 1 ターゲット VLIW プロセッサにおけるオペレーション分類

クラス	内容
NOP	No Operation
ALU	算術演算命令、論理演算命令
MUL	乗算命令
LS	ロード・ストア命令
CNT	制御命令

釈した結合命令 [8] を用いる方式の 2 つがある．本手法では結合命令を対象とする．結合命令は ID ステージにおいてデコードされ，各スロットに個別命令が発行される．結合命令において命令エンコーディグに必要なビット数，すなわち命令メモリのビット幅 M^i_{width} は，結合命令の集合を J，汎用レジスタ数を R とすると式 (3) となる．

$$M^i_{width} = \lceil \log_2 |J| \rceil + 3 \times S \times \lceil \log_2 R \rceil \quad (3)$$

2.3 エネルギーモデル

あるアプリケーションプログラムを実行したときのプロセッサの消費エネルギー E は，式 (4) のようにプロセッサコアの消費エネルギー E_{core} とメモリの消費エネルギー E_{mem} の総和となる．

$$E = E_{core} + E_{mem} \quad (4)$$

2.3.1 プロセッサコアエネルギーモデル

プロセッサコアのエネルギーモデルは [12] に基づく．この手法は，VLIW スロットを独立して見積もり可能であると仮定し，命令を表 1 のように分類してモデル化している．キャッシュミスなどで消費するエネルギーコスト等を無視すると式 (5) のようになる．

$$E_{core} = \sum_{n=1}^{N} \left(U(\mathbf{0}|\mathbf{0}) + \sum_{k=1}^{S} \nu(w^k_n | w^k_{n-1}) \right) \quad (5)$$

$\mathbf{0}$ は [NOP, NOP, $\cdots$,NOP]，$U(\mathbf{0}|\mathbf{0})$ はベースエネルギーコスト，$\nu(w^k_n|w^k_{n-1})$ はスロット k で w^k_{n-1} から w^k_n へ変化するときのエネルギー消費量である．なお，$\nu(w^k_n|w^k_{n-1})$ はプロセッサパラメータに依存するが，命令セットの変更によりデコーダが変化しても見積り式におけるエネルギー消費量の変化は無視できるものとする．

2.3.2 メモリエネルギーモデル

電源電圧を 1.0V とし，RAM および ROM のエネルギーを以下のようにモデル化する．

$$E_{mode} = K_0 + K_1 M_{width} + K_2 M_{length} \quad (6)$$

ここで，E_{mode} はメモリの各状態における消費エネルギーであり，各状態は書き込み（write），読み込み（read），アクセスなし（noaccess）である．K_0，K_1，K_2 は係数，M_{width} はメモリのビット幅，M_{length} はワード長である．また，リーク電力も同様に以下のようにモデル化する．

$$P_{leak} = K_0 + K_1 M_{width} + K_2 M_{length} \quad (7)$$

これに基づき，メモリマクロ生成ツールを使用し，生成されたメモリマクロから回帰分析により，メモリの書き込みサイクルのエネルギー E_{write} [pJ]，読み取りサイクルのエネルギー E_{read} [pJ]，アクセスのないサイクルのエネルギー $E_{noaccess}$ [pJ]，リーク電力 P_{leak} [mW] の見積式の各係数を導出した．メモリの消費エネルギー E_{mem} [pJ] は，命令メモリのビット幅を M^i_{width}，ワード長を M^i_{length}，X,Y データメモリのビット幅を M^d_{width}，ワード長を M^d_{length}，アプリケーション実行時の実行サイクル数を T，命令メモリ読み取り回数を T^i_{read}，命令メモリにアクセスしてないサイクル数を $T^i_{noaccess}$，X データメモリ読み取り回数を T^x_{read}，X データメモリ書き込み回数を T^x_{write}，X データメモリにアクセスしてないサイクル数を $T^x_{noaccess}$，Y データメモリ読み取り回数を T^y_{read}，Y データメモリ書き込み回数を T^y_{write}，Y データメモリにアクセスしてないサイクル数を $T^y_{noaccess}$，動作周波数を f [GHz] とすると式 (8) となる．

$$\begin{aligned} E_{mem} = \; & T^i_{read} \times E_{read}(M^i_{width}, M^i_{length}) \\ & + T^i_{noaccess} \times E_{noaccess}(M^i_{width}, M^i_{length}) \\ & + (T^x_{read} + T^y_{read}) \times E_{read}(M^d_{width}, M^d_{length}) \\ & + (T^x_{write} + T^y_{write}) \\ & \times E_{write}(M^d_{width}, M^d_{length}) \\ & + (T^x_{noaccess} + T^y_{noaccess}) \\ & \times E_{noaccess}(M^d_{width}, M^d_{length}) \\ & + T/f \times P_{leak} \end{aligned} \quad (8)$$

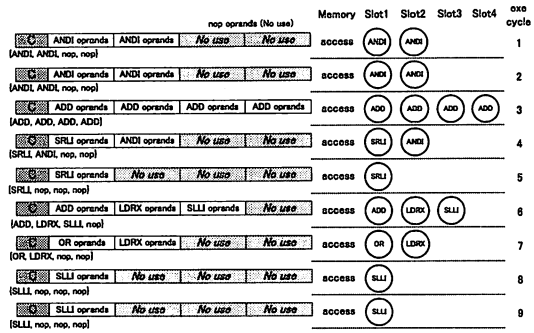


図 4 結合命令による各実行サイクルの演算内容

3. 命令メモリアクセス数削減による消費エネルギー削減手法

本手法では，文献 [8] で導入された結合命令の空き領域に次サイクル以降の命令を格納することで，命令メモリへのアクセス数を削減する．以下，基本アプローチ，アーキテクチャ，および垂直結合命令探索手法について説明する．

3.1 基本アプローチ

結合命令の各実行サイクルにおける演算内容，および命令メモリからのフェッチ内容の例を図 4 に示す．図 4 の左側が命令メモリからフェッチされる命令コードであり，右側が各実行サイクルにおける VLIW スロット毎の演算内容である．図 4 の access と表示されている実行サイクルで命令メモリへアクセスがされる．通常の VLIW 命令と同様，結合命令の実行では全ての実行サイクルにおいて命令メモリへのアクセスが必要となり，図 4 の例では，実行サイクルの 9 サイクル全てで命令メモリへのアクセスが必要となる．

VLIW プロセッサはデータ依存の問題から並列実行できない命令が多く存在する．そのため，並列実行できないスロットに対して VLIW プロセッサは NOP 命令を発行する．図 4 の 1 サイクル目の例だと，スロット 1，2 では演算がされるが，スロット 3，4 では NOP 命令が発行されることになり無駄となる．文献 [6] では 8 スロット VLIW プロセッサに対して，複数アプリケーションにおけるループ中の命令レベル並列性を調査し，1.50〜4.10 であったと報告している．

結合命令の集合が J の時，結合オペコードの必要ビット数は $\lceil \log_2 |J| \rceil$ となる．そのため，$2^{\lceil \log_2 |J| \rceil} - J$ の割り当てがされていない結合オペコードが存在することになる．

本手法はこれらの点に着目し，1 命令に複数サイクルの結

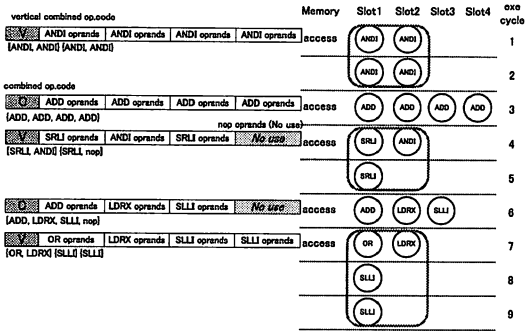


図 5　垂直結合命令導入による各実行サイクルの演算内容

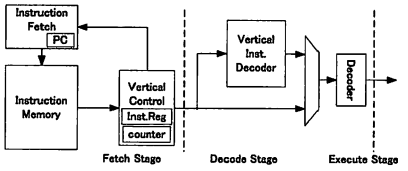


図 6　垂直結合命令アーキテクチャ

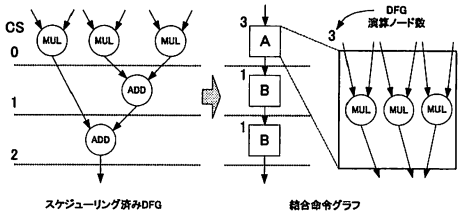
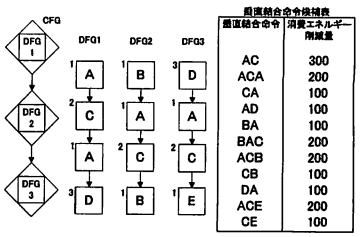


図 7　スケジューリング済み DFG 演算ノード概略図

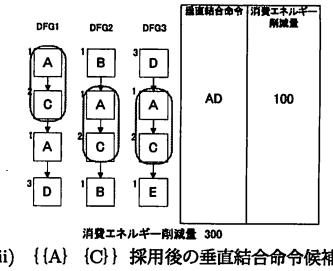
合命令を格納する新たな命令を考え，**垂直結合命令**と定義する．垂直結合命令は逐次実行される結合命令の集合であり，垂直結合命令に含まれる NOP 命令を除く個別命令の数はスロット数を超えないものとする．図 4 の結合命令に垂直結合命令を導入した例を図 5 に示す．図 5 でオペコードに V と示されている 1，4，7 サイクル目の命令が垂直結合命令となり，これらは複数サイクルの命令を格納した専用命令（**垂直結合オペコード**）である．垂直結合命令は NOP 命令のオペランド部を次サイクル以降のオペランドに利用することで複数サイクルの命令を格納することができる．図 5 における 7 サイクル目の例だと {OR,LDRX,nop,nop}，{SLLI,nop,nop,nop}，{SLLI,nop,nop,nop} と 7〜9 サイクル目で実行される結合オペコードに対して {{OR,LDRX} {SLLI} {SLLI}} という新たな垂直結合オペコードを割り当てている．{{OR,LDRX} {SLLI} {SLLI}} 垂直結合命令は 7 サイクル目でのみ命令メモリからフェッチされ，8，9 サイクル目の実行には命令メモリへのアクセスはされない．図 4 では 9 サイクルの実行全てで命令メモリのアクセスが必要であったのに対して，図 5 では垂直結合命令を導入することで，命令メモリへのアクセスは 5 回に削減される．このように垂直結合命令の導入により，命令メモリへのアクセス数を削減することができる．

3.2　アーキテクチャ

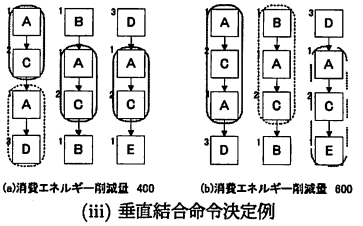
図 6 に垂直結合命令を実現するためのアーキテクチャを示す．命令メモリからフェッチされた命令は，命令レジスタに保存される．保存された命令が結合命令である場合はデコーダにより各スロットに個別命令が発行される．垂直結合命令である場合は，一度結合命令にデコードされた後に個別命令にデコードされる．垂直結合命令から結合命令へのデコードは制御部のカウンタ値を使用し，カウンタ値は垂直結合命令の実行サイクル数，つまり垂直結合命令に格納された結合命令数だけカウントされる．図 5 の 7〜9 サイクル目における {{OR,LDRX} {SLLI} {SLLI}} の例だと，結合命令数 3 がカウンタ値にセットされる．データ依存の問題から，垂直結合命令中の結合命令



(i) DFG における垂直結合命令候補およびエネルギー削減量



(ii) {{A} {C}} 採用後の垂直結合命令候補表



(iii) 垂直結合命令決定例

図 8　CDFG における垂直結合命令候補表

は並列実行はできないので，{{OR,LDRX} {SLLI} {SLLI}} が読み出されると，1 サイクル目で {OR,LDRX,nop,nop} が実行されカウンタ値がデクリメントされて 2 になる．2 サイクル目は {SLLI,nop,nop,nop} が実行されカウンタ値は 1 に，3 サイクル目は {SLLI,nop,nop,nop} が実行され，カウンタ値は 0 にデクリメントされる．カウンタ値が 0 になったら垂直結合命令の実行は終わり，命令メモリへのアクセスが再開される．{{OR,LDRX} {SLLI} {SLLI}} の例だと 1 回の命令メモリへのアクセスで 3 サイクルの結合命令が実行されることになる．

3.3　垂直結合命令探索手法

ある垂直結合命令を適用したときに命令メモリのアクセス数削減によって減らすことのできるエネルギーを消費エネルギー削減量と定義する．図 5 の 7〜9 サイクル目の例だと，8，9 サイクル目の {SLLI,nop,nop,nop}，{SLLI,nop,nop,nop} で必要だった命令メモリアクセスエネルギーが垂直結合命令 {{OR,LDRX} {SLLI} {SLLI}} の消費エネルギー削減量となる．命令の割り当てがされていない結合オペコードにこの消費エネルギー削減量が最大となるように垂直結合命令を割り当てる．対象となるアプリケーションプログラムに対して全探索を考えた場合，垂直結合命令の候補数を m，割り当てがされていない結合オペコードの数を c とすると，最大で $\binom{m}{c}$ の探索空間が存在する．JPEG の処理の 1 つである離散コサイン変換（DCT）で探索数が計算上 10^{14} を超えるなど非常に大きくなるため，ヒューリスティックな探索手法が必要となる．

提案する垂直結合命令探索手法はスケジューリング済み CDFG における DFG の演算ノードに着目する．以下の説明では，スケジューリング済み DFG の演算ノードを図 7 の右側のような結合命令グラフで表す．結合命令グラフは逐次実行される結合命令のシーケンスを表しており，各ノードは結合命令，各ノードの重みは結合命令に含まれる演算ノード数とする．また，CS_i はコントロールステップ i において実行される結合命令とする．

提案する垂直結合命令探索手法の基本方針は，スケジューリング済み CDFG に対して，垂直結合命令の候補とその垂直結

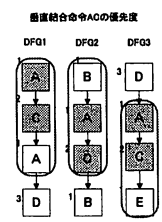
入力: スケジューリング済み CDFG
出力: 垂直結合命令候補表

Step t0. CFG の開始ノードに含まれる DFG から探索を開始する.
Step t1. CS=0, i=1 とする.
Step t2-0. CS から CS + i までの演算ノード数をカウントし, スロット数以下ならば, CS から CS + i までの演算ノードで構成される垂直結合命令を候補 v とし, 消費エネルギー削減量 e を計算する. そうでなければ, Step t3. へ
Step t2-1. v が既に垂直結合命令候補表に含まれていれば, 垂直結合命令候補表における v の消費エネルギー削減量に e を加算する. そうでなければ, 新たな候補として消費エネルギー削減量を e とした v を追加する.
Step t2-2. i++とし, Step t2-0 へ.
Step t3. 探索がされていない CS があるなら, CS++, i=1 とし, Step t2-0. へ
Step t4. 全ての DFG で探索がされたら終了. そうでなければ次の DFG に探索を進め Step t1. へ

図 9 垂直結合命令候補表作成手法

合命令を採用した時の消費エネルギー削減量を持つ垂直結合命令候補表を作成し, 消費エネルギー削減量の大きいものから垂直結合命令として採用していく. 図 8 (i) にスロット数を 4, 各コントロールステップの命令メモリへのアクセスエネルギーを 100 としたときのアプリケーションプログラム (CDFG) と垂直結合命令候補表の例を示す. 図 8 の例は CFG ノードを 3 つ持ち, 各 CFG ノードはそれぞれ DFG1, DFG2, DFG3 を持つ. まず, 垂直結合命令候補表の作成であるが, 垂直結合命令の候補は複数の連続する結合命令で演算ノード数がスロット数以下の演算ノードで構成される命令とし, 図 9 に示す候補表作成手法により候補を決定する. 例えば, 図 8 (i) における DFG1 の例だと, まず CS0 から探索がされ, CS0〜1 の演算ノード数は A が 1 (CS0), C が 2 (CS1) の 3 となり, スロット数以下であるので AC は垂直結合命令の候補となる. 同様にして, CS0〜2 も演算ノード数が 4 となり, スロット数以下であるので ACA は候補となるのだが, CS0〜3 の ACAD は演算ノード数が 7 となり, スロット数を超えてしまうので候補にはならない. 演算ノード数がスロット数を超えたら, 次の CS に探索を進める. 以上のような探索を図 9 のように全ての DFG ですることで, 垂直結合命令候補表は図 8 (i) の右側のようになる. 垂直命令候補表を作成したら, 消費エネルギー削減量の最も大きい垂直結合命令を命令セットとして採用する. 図 8 (i) では, 消費エネルギー削減量 300 である AC を採用する. 垂直結合命令を採用した後, 図 8 (ii) のように採用した垂直結合命令の結合命令を除いて垂直結合命令候補表を更新し, 再び消費エネルギー削減量の最も大きい垂直結合命令を採用する. 提案する手法の基本方針は, 以上のようなステップを結合オペコード全てに割り当てがされるか垂直結合命令の候補がなくなるまで繰り返すことで垂直結合命令を決定する.

しかし, 消費エネルギー削減量の大きい候補から垂直結合命令として採用していくと図 8 (i) の例のような場合に (iii) a の AC, AD の垂直結合命令を採用してしまう. 実際には (iii) b の ACA, BAC, ACE の垂直結合命令の採用が消費エネルギー削減量が最大の命令セットであるのだが, 最も消費エネルギー削減量の大きな垂直結合命令 AC を採用したことにより, ACA などの次に消費エネルギー量が大きな垂直結合命令を採用することができなくなったためである. そこで各垂直結合命令間の影響関係を考慮するために, 図 10 のように各垂直結合命令に**優先度 (priority)** を付加する. CS〜(CS+i) の演算ノードで構成される垂直結合命令の優先度は CS〜(CS+i+1) または (CS-1)〜(CS+i) の演算ノードで構成される垂直結合命令のいずれか大きいほうの消費エネルギー削減量とする. 図 10 の垂直結合命令 AC の例だと, DFG1 の CS0〜1 における AC の優先度は CS0〜2 の **ACA** の消費エネルギー削減量 200, DFG2 の CS1〜2 における AC の優先度は CS0〜2 の **BAC** の消費エネルギー削減量 200, DFG3 の CS1〜2 における AC の優先度は CS1〜3 の **ACE** の消費エネルギー削減量 200 となる. CDFG 全体ではこれらの総和である 600 が AC の優先度となる. つまり, 垂直結合命令 AC を採用することで 300 の消費エネルギーを削減できるが, AC が使用される演算ノードで別の **ACA**,



垂直結合命令	消費エネルギー削減量	優先度
AC	300	600
ACA	200	0
CA	100	200
AD	100	0
BA	100	200
BAC	200	0
ACB	200	0
CB	100	200
DA	100	0
ACE	200	0
CE	100	200

図 10 垂直結合命令の優先度

入力: 結合命令適用済みアプリケーションプログラム (CDFG), 結合命令セット
出力: 垂直結合命令適用済みアプリケーションプログラム (CDFG), 垂直結合命令セット
目的関数: 消費エネルギー削減量最大
制約: 割り当てがされていない結合オペコードの数 > 垂直結合命令セット数

Step 0. 入力アプリケーションプログラムから垂直結合命令候補表を作成する. 表は垂直結合命令の候補 v および各垂直結合命令適用時の消費エネルギー削減量 $E(v)$, 優先度 $P(v)$ を持つ
Step 1. $E(v) \geq P(v)$ かつ $E(v)$ 最大の垂直結合命令 v を垂直結合命令セットに加え, 垂直結合命令 v を適用した CDFG に変形する
Step 2. 各垂直結合命令候補の消費エネルギー削減量および優先度の再計算をし, 垂直結合命令候補表を更新する.
Step 3. 垂直結合命令の候補が存在し, かつ制約を満たしていたら Step1. へ戻る. そうでなければ, アプリケーションプログラムと垂直結合命令セットを出力して終了.

図 11 垂直結合命令探索手法

表 2 分類された命令の差分エネルギー (pJ)

	NOP	ALU	MUL	LS	CNT
NOP	0	1.76	1.94	1.79	2.21
ALU	-	2.19	2.01	2.01	2.27
MUL	-	-	2.50	2.23	2.33
LS	-	-	-	2.25	2.42
CNT	-	-	-	-	1.98

表 3 ROM 消費エネルギー等の見積式の係数

	K_0	K_1	K_2
E_{read}	6.60	2.78	1.14×10^{-2}
P_{leak}	3.03×10^{-4}	4.23×10^{-5}	4.78×10^{-6}

表 4 SRAM 消費エネルギー等の見積式の係数

	K_0	K_1	K_2
E_{write}	8.43×10^{-2}	7.04×10^{-1}	3.16×10^{-3}
E_{read}	1.35	5.01×10^{-1}	1.94×10^{-3}
P_{leak}	5.77×10^{-4}	3.60×10^{-4}	9.68×10^{-6}

BAC, **ACE** を採用することで 600 の消費エネルギーを削減できることを意味している. よって垂直結合命令 v の消費エネルギー削減量を $E(v)$, 優先度を $P(v)$ とするとき, 垂直結合命令候補表から $E(v) \geq P(v)$ かつ $E(v)$ が最大の垂直結合命令を選択することで, 図 10 のような例でも消費エネルギー削減量が最大となる垂直結合命令を探索することができる. 図 11 に優先度を用いた垂直結合命令探索手法を示す. 手法の主な流れは基本方針と変わらないが, Step 1. において $E(v) \geq P(v)$ かつ $E(v)$ が最大の垂直結合命令候補を採用している.

提案するアルゴリズムは垂直結合命令の全ての組合せを探索せずに, パラメータとして優先度を用いて消費エネルギー削減量の大きいものから採用していく. そのため図 11 のアルゴリズムの計算量は, スロット数を S, DFG ノード数を n_{dfg}, 垂直結合命令の候補数を m, 割り当てがされていない結合オペコードの数を c とすると $O(Sn_{dfg}mc)$ になる.

4. 計算機実験

垂直結合命令探索手法を CoDaMa フレームワーク [7] 上で複数アプリケーションに対して実験した. 入力とするアプリケーションはアルファブレンド, JPEG エンコーダである. JPEG エンコーダの各処理は, 色空間変換 (YUV), 最小符号化ユニット生成 (MCU), 離散コサイン変換 (DCT), 量子化 (Q), 可変長符号化 (VLC) である. 各アプリケーションの入力画像は, アルファブレンドを 320x240[pixel], 8bitRGBA, JPEG エンコーダを 256x256[pixel], 8bitRGB とした.

プロセッサの構成は, 4 スロット, 5 段パイプライン, ALU × 4, SIMD 型乗算器 × 2, ハードウェアループユニット有,

表 5 エネルギー削減割合と探索時間

application	energy reduction [%]	exploration time [sec]
ALPHA_BLEND	40.9	17.7
JPEG_ENC	42.1	184
DCT	47.3	28.9
MCU	44.0	20.7
YUV	31.4	22.7
VLC	47.7	46.8
Q	40.4	112

表 6 実験結果

app	ASIP	word length	bit width	v.op num	Inst.Mem access	energy [pJ]
ALPAH BLEND	VLIW	29	128	-	1305610	877601361
	文献 [8]	31	65	-	1305612	589724402
	優先度なし	19	53	9	921605	530791214
	提案手法	19	53	8	844806	519042828
JPEG ENC	VLIW	559	128	-	255149398	140146868097
	文献 [8]	570	92	-	255205976	103548085053
	優先度なし	324	80	91	162170685	81245176894
	提案手法	315	80	97	161926023	81169838582
DCT	VLIW	86	128	-	228698624	119848384350
	文献 [8]	92	67	-	228705794	77977958205
	優先度なし	68	66	14	151190018	63211993649
	提案手法	68	66	14	151190018	63211993649
MCU	VLIW	52	128	-	497920	314151118
	文献 [8]	56	66	-	498944	212977319
	優先度なし	44	65	7	302336	176214219
	提案手法	44	65	7	302336	176214219
YUV	VLIW	60	128	-	2036224	1465337921
	文献 [8]	64	78	-	2037248	1092709147
	優先度なし	43	66	15	1574912	1004647819
	提案手法	42	66	14	1574656	1004579834
VLC	VLIW	188	128	-	15523926	10808799220
	文献 [8]	188	67	-	15524438	7573776297
	優先度なし	114	66	22	5695339	5688872304
	提案手法	111	66	22	5523090	5655816939
Q	VLIW	173	128	-	8392704	8257545907
	文献 [8]	186	79	-	8700416	4719196674
	優先度なし	110	67	33	3887222	3781728677
	提案手法	107	67	38	3646529	3735117058

アドレッシングユニット有，フォワーディングユニット有，Yデータメモリ有とした．テクノロジライブラリはSTARC 90nm(注1)，電源電圧は1.0V，動作周波数は150MHzとした．また命令メモリはROMとし，各アプリケーションに必要最小限の容量とした．X,Yデータメモリは SRAMとし，容量を512KB，ビット幅32bitで固定とした．消費エネルギーの評価方法としては，RTL設計にVHDL，セルの生成および論理合成にSynopsys社 Design Compiler Z-2007.03-SP4，スイッチング確立の導出にSynopsys社 VCS-MX，その結果を基にした電力解析にSynopsys社 Prime Timeを用いた．この構成での，メモリを除くプロセッサコア消費エネルギー見積手法におけるベースエネルギーコストは31.5pJ，各スロットの命令遷移におけるエネルギーの各パラメータは表2に示す．SRAMおよびROMのパラメータはSTARC (CMOS90nm)ARMメモリマクロを用いて評価実験し，$E_{noaccess}$ は1pJで固定とし，E_{write}，E_{read}，P_{leak} の各パラメータは表3，4となった．

以上のような条件でCPU: Intel Core Duo 1.86GHz，Memory: 1GB，OS: Debian GNU/Linux sargeの環境で実験した．各アプリケーションの実験結果を表5，6に示す．それぞれ通常の命令エンコード方式VLIWアーキテクチャ（VLIW），結合命令による命令メモリビット幅削減手法手法（文献[8]），優先度を用いずに消費エネルギー削減量が大きいものから採用した垂直結合命令（優先度なし），優先度を用いて決定した垂直結合命令（提案手法）である．提案手法は，通常のVLIWアーキテクチャと比べて31.4%～47.7%で平均41.9%，文献[8]手法と比べて8.1%～25.3%で平均17.7%の消費エネルギーを削減した．優先度について比較すると，提案手法は優先度を用いない場合と比べて全く同じか2%程度の消費エネルギーを削減した．

5. むすび

本稿では命令メモリアクセス数削減に基づく低エネルギーASIP合成手法を提案した．計算機実験において平均41.9%の消費エネルギー削減を確認した．

文献

[1] ARM7TDMI Technical Manual ARM, http://www.arm.com/

[2] T. Glokler and H. Meyr, *Design of energy-efficient application-specific instruction set processors*, Springer, 2004.

[3] J. L. Hennessy and D. A. Patterson, *Computer architecture: A quantitation approach*, Morgan-Kaufman, 1990.

[4] IBM, "CodePack PowerPC code compression utility user's manual," 1998.

[5] 今井正治, 谷口一徹, 武内良典, 坂主圭史, "コンフィギュラブル・プロセッサ開発環境 ASIP Meister," 信学技報, VLD2006-5, Vol. 106, No.31, pp.25-30, 2006.

[6] M. Jayapala, F. Barat, T. Vander Aa, F. Catthoor, H. Corporaal, and G. Deconinck, "Clustered loop buffer organization for low energy VLIW embedded processors," *IEEE Trans. on Computers*, Vol.54, No.6, pp.672-683, 2005.

[7] S. Kohara, Y. Shi, N. Togawa, M. Yanagisawa, and T. Ohtsuki, "CoDaMa: An XML-based framework to manipulate control data flow graphs," in *Proc. of SASIMI 2007*, pp.545-549, 2007.

[8] 小原俊逸, 史又華, 戸川望, 柳澤政生, 大附辰夫, "命令メモリビット幅削減に基づく低エネルギー ASIP 合成手法," 信学技報, Vol.107, No. 506, VLD2007-141, pp. 25-30, 2008.

[9] A. Krishnaswamy and R. Gupta, "Profile guided selection of ARM and Thumb instruction," in *Proc. of ACM SIGPLAN*, 2002.

[10] MIPS Technologies, "MIPS32 architecture for programmers volume IV-a: The MIPS16 application specific extension to the MIPS32 architecture," 2001.

[11] P. Morgan and R. Taylor, "ASIP instruction encoding for energy and area reduction," in *Proc. of DAC 2007*, pp.797-800, 2007.

[12] M. Sami, D. Sciuto, C. Silvano, and V. Zaccaria, "An instruction-level energy model for embedded VLIW architectures," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 21, No.9, pp. 998-1010, 2002.

[13] A. Shrivastava and N. Dutt, "Energy efficient code generation exploiting reduced bit-width instruction set architecture (rISA)," in *Proc. of ASP-DAC 2004*, pp. 475-477, 2004.

[14] A. Shrivastava, P. Biswas, A. Halambi, N. Dutt, and A. Nicolau, "Compilation framework for code size reduction using reduced bit-width ISAs (rISAs)," *ACM Tran. on Design Automation of Electronic Systems*, Vol. 11, No. 1, pp. 123-146, 2006.

[15] Tensilica Inc. Xtensa Microprocessor, http://www.tensilica.com/

[16] Texas Instruments Incorporated, "TMS320C62xx CPU and instruction set: Reference guide," 1997.

[17] N. Togawa, M. Yanagisawa, and T. Ohtsuki, "A hardware/software cosynthesis system for digital signal processor cores," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E82-A, No.11, pp.2325-2337, 1999.

[18] N. Togawa, K. Tachikake, Y. Miyaoka, M. Yanagisawa, and T. Ohtsuki, "A hardware/software partitioning algorithm for processor cores with Packed SIMD-type instructions," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E86-A, No.12, pp.3218-3224, 2003.

[19] N. Togawa, K. Tachikake, Y. Miyaoka, M. Yanagisawa, and T. Ohtsuki, "A SIMD instruction set and functional unit synthesis algorithm with SIMD operation decomposition," *IEICE Trans. on Information and Systems*, Vol. E88-D, No.7, pp.1340-1349, 2005.

[20] Toshiba Semiconductor Company, Microcomputer: MeP, http://www.mepcore.com/

[21] A. Wolfe and A. Chanin, "Executing compressed programs on an embedded risc architecture," in *Proc. of the 25 annual International symposium on Microarchitecture*, Vol. 13, No.7, pp.181-210, 1992.

(注1)：STARC90nm ライブラリは東京大学大規模集積システム設計教育研究センターを通し，株式会社半導体理工学研究センター (STARC) と株式会社先端 SoC 基盤技術開発 (ASPLA) の協力で開発されたものである．