情報処理学会論文誌 Vol.54 No.7 1883–1891 (July 2013)

VLIWプロセッサのための電力制約を考慮した命令スケジューリング手法

藤井 裕也[1,a)] 武内 良典[1,b)] 今井 正治[1,c)]

受付日 2012年11月5日, 採録日 2013年4月5日

概要：本論文では，VLIW プロセッサのための電力制約を考慮した命令スケジューリング手法を提案する．近年，組込みプロセッサは家電製品や携帯電話など様々な製品で用いられる．これらの組込みプロセッサの中には，消費電力に制約がある中で高い性能を発揮することが求められるものもある．低消費電力で高い性能を期待できるプロセッサとして，VLIW（Very Long Instruction Word）プロセッサがある．VLIW プロセッサは複数の演算を同時に実行できるため演算処理能力が高いが，同時に処理する演算の組合せによって消費電力が大きく異なるため，ピーク電力が高くなってしまう可能性がある．そこで，VLIW プロセッサのための電力制約を考慮した命令スケジューリング手法を提案する．本スケジューリング問題を電力制約下で実行サイクル数を最小化する最適化問題と定式化し，この問題の最適解を求めるアルゴリズムを示す．その後，この問題の準最適解を求めるアルゴリズムを提案し，最適解との比較を行う．評価実験では，パイプライン構造とマルチサイクル命令を持つプロセッサに対し様々な電力制約のもとでベンチマークプログラムへのスケジューリングを行い，スケジューリング時間と得られたスケジュールの性能を比較した．その結果，提案するアルゴリズムでは，短い命令列に対してはほぼ最適解と同等の解が得られること，最適解が 1 時間以内に求まらない問題に対しても準最適解を 1 ミリ秒以内に求められることを確認した．

キーワード：VLIW，コンパイラ，命令スケジューリング，消費電力，パイプライン

Instruction Scheduling for VLIW Processors Considering the Power Constraints

Yuya Fujii[1,a)] Yoshinori Takeuchi[1,b)] Masaharu Imai[1,c)]

Received: November 5, 2012, Accepted: April 5, 2013

Abstract: In this paper, an instruction scheduling for VLIW processors considering power constraints is proposed. In recent years, most products such as home electronics and mobile phones contain embedded processors, which are required high performance under the limited power. While Very Long Instruction Word (VLIW) processors can achieve high performance because they can execute several instructions simultaneously, they consume high power at the peak load. Therefore, an instruction scheduling for VLIW processors considering power constraints is proposed. This scheduling problem is defined as an optimization problem for minimizing execution cycles under the power constraint. The algorithms for searching optimal solutions and suboptimal solutions are proposed. Experimental results show that the proposed method can generate almost optimal scheduling results within enough short time for small problems. Scheduling results can be obtained within a millisecond for the input that the optimal solution cannot be obtained within an hour.

Keywords: VLIW, compiler, instruction scheduling, power consumption, pipeline

1. はじめに

近年，組込みプロセッサは家電製品や携帯電話など，様々な製品で用いられる．これらの組込みプロセッサの中に

[1] 大阪大学
Osaka University, Suita, Osaka 565–0871, Japan
[a)] y-fujii@ist.osaka-u.ac.jp
[b)] takeuchi@ist.osaka-u.ac.jp
[c)] imai@ist.osaka-u.ac.jp

© 2013 Information Processing Society of Japan

は，消費電力に制約がある中で高い性能を発揮することが求められるものもある．たとえば，携帯電話には高速なマルチメディア処理が要求されるが，消費電力が高いと熱がより発生し，システムの信頼性を低下させる．

そこで，高い性能を期待でき，低消費電力である VLIW（Very Long Instruction Word）プロセッサが注目されてきた [1]．VLIW プロセッサは複数の演算を同時に実行できるため演算処理能力が高いが，同時に処理する演算の組合せによって消費電力が大きく異なるため，ピーク電力が高くなってしまう可能性がある．今後よりいっそうの高性能が求められ，プロセッサのマルチコア化，メニーコア化が進み大規模な並列処理が実行される場合，電力制約下での最適性能を実現するスケジューリング手法はますます重要となる．VLIW プロセッサは，同時実行による演算の組合せにより消費電力が大きく異なるため，複数の VLIW プロセッサで同じプログラムを並列に実行するアーキテクチャを考慮した場合などには，その単一の VLIW プロセッサにおけるピーク電力を抑えることが重要となる．複数のプロセッサのピーク電力はいっそう高くなってしまうためである．

VLIW プロセッサは前述のとおり複数の機能ユニットを同時に動かすことができるため，電力制約下での使用を考えたシステム設計を行う場合には慎重に設計される必要がある．消費電力の見積りを慎重に行わなければ，供給電力が低い場合に正しく動作しない可能性や，ピーク時の電流が高くなってしまうために経年劣化が進む可能性もある．また，システムの用途によっては，医療装置のように消費電力による発熱が人体に影響を与えるという問題などがあるため，プロセッサを低消費電力で動作させるためには，同時に実行する命令の選択問題は重要である．

そこで，本論文では，本問題を VLIW プロセッサで電力制約下でのプログラムの実行サイクル数を最小化する最適化問題として定式化し，最適解を求めるアルゴリズムを提案する．

本論文の構成は次のとおりである．まず，2 章で関連研究と本研究の差異についてまとめる．次に，3 章で命令スケジューリング問題を最適化問題として定式化する．4 章で命令スケジューリングのアルゴリズムを説明する．5 章で評価実験の結果を示す．最後に 6 章でまとめと今後の課題を示す．

2. 関連研究

消費電力を考慮した VLIW プロセッサの命令スケジューリングについては，様々な研究が行われている．Xiao らは，電力を考慮しない場合に比べてパフォーマンスが低下しない範囲で，消費電力を平均化する手法を提案している [2]．

Wang らは，命令スケジューリングによりリーク電力を最小化する手法を提案している [3]．

表 1 各命令の消費電力の例
Table 1 An example of the power consumption of each instruction.

	ADD	MUL	LOAD
stage1	0	0	0
stage2	4	4	2
stage3	3	12	3
stage4	2	2	22
sum	9	18	27

表 2 Toburen らの手法により得られたスケジューリング結果の例
Table 2 An example of schedule gained by Toburen's method.

	slot1	slot2	power [mW]
cycle1	LOAD		27
cycle2		MUL	18
cycle3	ADD	ADD	18
cycle4			0
cycle5			0
cycle6			0

Toburen らは，サイクルごとの消費電力量に上限を設け，上限を守るように命令スケジューリングを行うことで消費電力量の最大値を抑える手法を提案している [4]．Toburen らの手法では，各命令は発行されるサイクルにおいてのみ一定の電力を消費するものとし，すべてのサイクルがあらかじめ設定した消費電力の上限を超えないように命令を発行することでスケジューリングを行っている．Toburen らの手法では，パイプライン構造やマルチサイクル命令を考慮していない．そのため，パイプライン構造を持つプロセッサに対して Toburen らの手法を適用すると，サイクルごとの消費電力の制約を守れないことがある．以下に，簡単な電力モデルを用いてパイプラインの考慮の有無による差異の例を示す．

表 1 に各命令の消費電力を，**表 2** に Toburen らの手法を用いて得られたスケジューリング結果の例を示す．消費電力の制約は 30 mW としてスケジューリングを行った．Toburen らの手法では消費電力の制約を守るために同時に発行する命令の消費電力の合計が制約電力を超えないようにスケジューリングを行うが，その際の命令の消費電力は命令を発行するサイクルにのみかかると仮定する．表 2 中の 3 サイクル目の消費電力を例にあげると，ADD 命令の消費電力は表 1 から 9 mW であるため，3 サイクル目の消費電力は合計 18 mW となる．

表 3 に，パイプラインを考慮した場合の表 2 のスケジュールの消費電力を示す．各命令は各パイプラインステージにおいて表 1 中の対応する値だけ電力を消費するものとして消費電力を計算している．表 3 中の 4 サイクル目において実行中であるのは，LOAD 命令の第 4 ステージ，ADD 命令の第 2 ステージ，MUL 命令の第 3 ステー

© 2013 Information Processing Society of Japan

表 3 パイプラインを考慮したときの表 2 のスケジュールの消費電力 [mW]
Table 3 Power consumption of the schedule shown in the Table 2 considering pipeline.

	slot1		slot2		sum
cycle1	0 (LOAD:stage1)	0	0	0	0
cycle2	2 (LOAD:stage2)	0	0 (MUL:stage1)	0	2
cycle3	3 (LOAD:stage3)	0 (ADD:stage1)	4 (MUL:stage2)	0 (ADD:stage1)	7
cycle4	22 (LOAD:stage4)	4 (ADD:stage2)	12 (MUL:stage3)	4 (ADD:stage2)	42
cycle5	0	3 (ADD:stage3)	2 (MUL:stage4)	3 (ADD:stage3)	8
cycle6	0	2 (ADD:stage4)	0	2 (ADD:stage4)	4

ジ，ADD 命令の第 2 ステージの 4 つのステージである．これらの命令の各ステージに対応する消費電力を表 1 から取得して合計すると 42 mW となり，消費電力の上限である 30 mW を超えていることが分かる．

表 2 の例では 5 サイクルの平均をとると消費電力は上限を超えていないが，SIMD プロセッサ [5] において同じ処理が同時に行われる場合，あるサイクルだけ急激に消費電力が高くなり，チップの劣化を招いたり誤った計算結果が得られたりする可能性がある．

本論文で提案するアルゴリズムでは，Toburen らの手法と同様にサイクルごとの消費電力に上限を設け，すべてのサイクルが消費電力の上限を超えないようにスケジューリングを行う．その際，パイプライン構造とマルチサイクル命令に対応するため，各命令が消費する電力をパイプラインステージごとに求めておく．ある命令 n_i をサイクル t において発行できるかどうか調べる際に，サイクル t での消費電力だけでなく，その後のサイクルの消費電力の制約を満たすかどうかを調べることで，より厳密に消費電力の制約を守る．また，マルチサイクル命令を考慮することで，より広い範囲の VLIW プロセッサモデルを命令スケジューリングの対象とすることができる．

3. スケジューリング問題の定式化

3.1 VLIW プロセッサモデル

本論文の命令スケジューリングで対象とするプロセッサについて説明する．**図 1** に VLIW プロセッサの例を示す．

あるサイクルにおいて使用しない機能ユニットは，そのサイクル中電力を消費しないと仮定する．

機能ユニットの種類と数，パイプライン段数，各機能ユニットの消費電力，消費電力の上限，命令セットはパラメータとし，命令スケジューリングへの入力として扱う．

3.2 電力モデル

本論文における電力モデルを説明する．機能ユニットの 1 サイクルあたりの消費電力があらかじめ求められているとし，t サイクル目で使用されている機能ユニットの消費電力の合計を t サイクル目での消費電力とする．また命令フェッチは毎サイクル行われ，一定の電力を消費するものとする．以下に消費電力の計算例を示す．

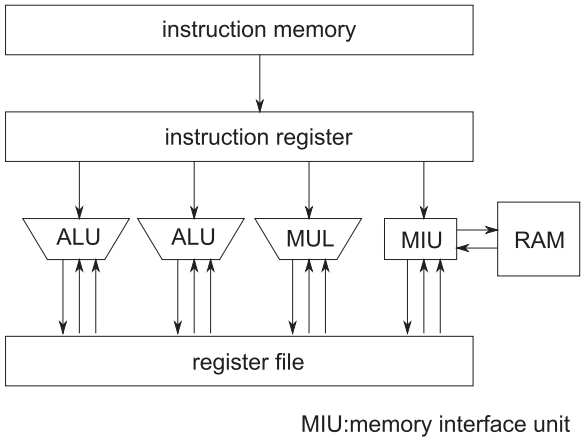

図 1 VLIW プロセッサの例
Fig. 1 An example of VLIW processors.

表 4 消費電力の計算例（命令の割当て）
Table 4 An example of power calculation (schedule of instructions).

cycle	slot1		slot2
1	LOAD(IF)		MUL(IF)
2	LOAD(ID)	ADD(IF)	MUL(ID)
3	LOAD(EX)	ADD(ID)	MUL(EX)
4	LOAD(WB)	ADD(EX)	MUL(EX)
5		ADD(WB)	MUL(WB)

表 4 のようなスケジューリングによる命令の割当てがあったとする．表 4 では LOAD 命令が 1 サイクル目に slot1 から，ADD 命令が 2 サイクル目に slot1 から，MUL 命令が 1 サイクル目に slot2 から発行されることを表している．括弧内はパイプラインステージを表しており，それぞれ IF は命令フェッチ，ID は命令デコード，EX は演算の実行，WB は計算結果の書き戻しを行うステージである．図 4 のスケジュールに対応する機能ユニットの使用状況は**表 6** で表される．各機能ユニットの 1 サイクルあたりの消費電力は**表 5** の値を使用する．2 サイクル目を例に消費電力の計算方法を説明する．命令フェッチの際に命令メモリからのリードと命令レジスタ 2 個へのライトが発生するため，命令フェッチ時の消費電力は $44 + 2 * 2 = 48$ [mW] となる．命令フェッチの他にレジスタ 4 個へのライトが発生するた

© 2013 Information Processing Society of Japan

表 5 消費電力の計算例（機能ユニットの消費電力）
Table 5 An example of power calculation (power of functional units).

functional unit	power [mW]
ALU	1
MUL	5
REG	2
DM_READ	20
IM_READ	44

表 6 消費電力の計算例（機能ユニットの割当て）
Table 6 An example of power calculation (schedule of functional units).

cycle	slot1		slot2	power
1				48
2	REG, REG		REG, REG	56
3	ALU, REG	REG, REG	MUL	60
4	DM_READ, REG	ALU, REG	MUL, REG	80
5		REG		50

め，2 サイクル目の消費電力の合計は $48+2*4=56$ [mW] となる．なお表 5 では，演算器，レジスタ，メモリ以外の論理回路は電力が小さいと仮定し，0 [mW] としている．また命令フェッチの際に使用する機能ユニットは可読性のため表 6 から除いている．

3.3 問題への入力と出力

入力は以下の項目からなる．

- 対象プロセッサのアーキテクチャ情報
- 命令間の依存関係を表したデータ依存グラフ（DDG）
- 消費電力の制約
- アセンブリ命令列

出力は各命令がどのサイクルで実行されるか，というアセンブリ命令列のスケジューリング結果である．

3.4 制約条件と目的関数

本スケジューリング問題の目的は，実行サイクル数を最小にすることである．制約条件は以下の 5 つである．

- 命令間の依存関係を違反しない．
- 1 サイクルで発行する命令の数が，同時発行可能である命令数を超えない．
- 1 サイクル中で使用する演算器の数が，使用可能である演算器の数を超えない．
- 1 サイクル中で使用する演算器の消費電力の合計が，消費電力の上限を超えない．
- 各命令が 1 回ずつ発行される．

以下それぞれの制約条件と目的関数を定式化する．

3.4.1 命令間の依存関係の制約

命令 n_l が命令 n_m より何サイクル後に発行する必要があるかという情報を $d_{l,m}$ と表す．命令 n_l が命令 n_m に依存していない場合は $d_{l,m}=-1$ とする．命令 n_i が発行されるサイクルを t_i とすると，以下の式が制約条件となる．

$$\forall\{l,m|d_{l,m}\geq 0\}\ t_l \geq t_m + d_{l,m} \tag{1}$$

3.4.2 スロット数の制約

変数 x を以下のように定義する．

$$x_{i,t}=\begin{cases} 1：\text{命令 } n_i \text{ がサイクル } t \text{ で発行される場合} \\ 0：\text{それ以外の場合} \end{cases}$$

同時発行可能命令数を s_{max} とすると，各サイクルでの発行命令数が s_{max} 以下である必要があるので，以下の式が制約条件となる．

$$\forall t \sum_i x_{i,t} \leq s_{max} \tag{2}$$

3.4.3 演算器の制約

以下のように y を定義する．

$$y_{i,j,t}=\begin{cases} 1：\text{命令 } n_i \text{ が演算器 } f_j \text{ を} \\ \quad \text{サイクル } t \text{ に使用する場合} \\ 0：\text{それ以外の場合} \end{cases}$$

$y_{i,j,t}$ は $x_{i,t}$ と命令 n_i が使用する演算器の情報から求まる．同時に使用可能な演算器 f_j の数を c_j とすると，以下の式が制約条件となる．

$$\forall j,t\ \sum_i y_{i,j,t} \leq c_j \tag{3}$$

3.4.4 消費電力の制約

消費電力の制約を P_{max}，演算器 f_j の 1 サイクルあたりの消費電力を p_j とする．演算器 f_j の t サイクル目での消費電力は以下の式で表される．

$$\sum_i y_{i,j,t}p_j \tag{4}$$

各サイクルにおいてすべての演算器の消費電力の合計が P_{max} 以下である必要があるので，以下の式が制約条件となる．

$$\forall t\ (\forall j\ \sum_i y_{i,j,t}p_j \leq P_{max}) \tag{5}$$

3.4.5 命令の制約

各命令はちょうど 1 回発行される必要があるので，以下の式を満たす必要がある．

$$\forall i\ \sum_t x_{i,t} = 1 \tag{6}$$

3.4.6 目的関数

本問題の目的は与えられた命令列の実行サイクル数を最小にすることである．実行サイクル数を T とすると，目的関数は以下の式で表せる．

$$T \rightarrow min \tag{7}$$

© 2013 Information Processing Society of Japan

4. スケジューリング手法

スケジューリングの範囲は基本ブロックとする．最適解を求めるスケジューリングアルゴリズムと準最適解を求めるスケジューリングアルゴリズムをそれぞれ説明する．

4.1 最適解を求めるアルゴリズム

本論文では，分枝限定法を用いて最適解を求める．分枝限定法では組合せ問題を部分問題に分割し（分枝操作），解の探索を行う．その際，ある部分問題を解く必要がなければその部分問題についての分枝操作は行わないことで探索時間の削減を図る（限定操作）．**図 2**，**図 3** に最適解を求めるアルゴリズムの擬似コードを示す．

図 2 において，集合 R とは現在のスケジュール s のサイクル t で発行しても命令間の依存関係を違反しないような命令の集合である．スケジュールとは，どの命令がどのサイクルで実行されるかという情報を表す．図 3 に示した check_constraints 関数では，命令 n_i を割り当てた後のスケジュールが 3.4 節で示した制約条件をすべて守るとき真を返し，そうでないときに偽を返す関数である．本アルゴリズムでは，スケジュールはサイクルごとに使用可能な機能ユニットと電力をあらかじめ持っているとし，命令を割り当てると対応する複数のサイクルから使用可能な機能ユニットと電力を減ずるという方針で消費電力と機能ユニットを管理する．提案手法では check_constraints 関数に見られるように，命令 n_i が使用する機能ユニット f は，命令 n_i を発行するサイクルではなく実際に機能ユニット f が使用されるサイクルにおいて電力を消費するとしている点で従来手法と異なる．reduce_resources 関数では check_constraints 関数と同様の方法で命令の使用する各機能ユニットの情報を調べ，各サイクルから使用可能な電力と機能ユニットの個数を減ずる．

本アルゴリズムでは，あるスケジュール中のサイクル t において次にどの命令を発行するかという問題で場合分けを行い，部分問題を作成する．サイクル t では新たに命令を発行しないスケジュールも存在するため，ある問題に対する子ノードの数は「発行する候補となる命令数 +1」となる．

スケジュール中ですべての命令が発行されると，そのスケジュールを解とする．各部分問題の解を比べ，最も良いものを最適解として採用する．探索時間を削減するため，集合 R をあらかじめ構築することや命令 n_i が発行可能かどうかを調べてから部分問題を作成することで，限定操作を行っている．また，探索中にそれまでに得られた解のうち最も実行サイクル数が小さい解を暫定解としておき，作成中の解がすでに暫定解より実行サイクル数が大きい場合には作成中の解をそれ以上探索しないように限定操作を

```
function scheduling_recur(Schedule s)
begin
 if (すべての命令を発行完了)
 begin
  return s
 end
 if (s の実行サイクル数が暫定解以上である)
 begin
  return s
 end
 best_schedule を初期化
 集合 R を構築
 foreach (n_i ∈ R)
 begin
  if (check_constraints(s, n_i) == true)
  begin
   reduce_resources(s, n_i)
   n_i を s の t サイクル目に割り当て
   schedule := scheduling_recur(s)
   if (schedule が best_schedule より実行サイクル数が小さい)
   begin
    best_schedule := schedule
   end
  end
 end
 t := t + 1
 schedule := scheduling_recur(s)
 if (schedule が best_schedule より実行サイクル数が小さい)
 begin
  best_schedule := schedule
 end
 return best_schedule
end
```

図 2 最適解を求めるアルゴリズム
Fig. 2 Algorithm for optimal solutions.

```
function check_constraints(Schedule s, Instruction n)
begin
 for i := 0 to t // t は n の実行にかかるサイクル数
 begin
 FUs := s.cycle[i].FUs
 power := s.cycle[i].power
  foreach (f ∈ n が i ステージ目で使用する機能ユニット)
  begin
   FUs[f] := FUs[f] − 1
   if FUs[f] < 0
   begin
    return false
   end
   power := power − f.power
  end
  if power < 0
  begin
   return false
  end
 end
 return true
end
```

図 3 check_constraints 関数
Fig. 3 check_constraints function.

© 2013 Information Processing Society of Japan

```
begin
 集合 R を構築し，優先度順にソート
 t := 1 // サイクル数の初期化
 while (未割り当ての命令がある)
 begin
  foreach (n_i ∈ R)
  begin
   if (check_constraints(s, n_i) == true)
   begin
    reduce_resources(s, n_i)
    n_i を t サイクル目に割り当て
    R を再構築し，優先度順にソート
   end
  end
  R を再構築し，優先度順にソート
  t := t + 1
 end
end
```

図 4 準最適解を求めるアルゴリズム
Fig. 4 Algorithm for approximate solutions.

行っている．

4.2 準最適解を求めるアルゴリズム

本スケジューリング問題は与えられた資源の中でできるだけ実行時間を短くする資源制約スケジューリングに分類できる．資源制約スケジューリング手法として広く用いられる手法に，リストスケジューリング [6] がある．リストスケジューリングに対しては様々な拡張が考えられている [7], [8]．本論文では，リストスケジューリングを拡張して本スケジューリング問題の準最適解を求める．

図 4 に準最適解を求めるアルゴリズムの擬似コードを示す．

図 4 において，集合 R とは現在のスケジュールのサイクル t で発行しても命令間の依存関係を違反しないような命令の集合である．check_constraints 関数および reduce_resources 関数は図 2 に示した関数である．集合 R を後述する優先度順に並べ替え，最も優先度の高いものを現在サイクルに割り当てようと試みる．命令 n_i が発行可能であれば命令 n_i を現在サイクルに割り当て，集合 R を再構築して優先度順にソートする．ここで R を再構築するのは，命令 n_i の発行によって新たに優先度の高い命令が発行可能になる可能性があるためである．現在サイクルで発行できる命令がなくなると，次のサイクルに進む．この操作をすべての命令が発行されるまで繰り返す．

優先度の決定方法を説明する．まずスケジューリング対象となる命令列について命令間の依存グラフを作成し，依存グラフに最終ノードを追加する．すべての命令ノードから最終ノードに依存関係があるとしたときに，最終ノードまでの最長距離を優先度として採用する．最終ノードまでの最長距離の計算方法を**図 5** に示す．

図 5 において，$d_{i,j}$ は命令 n_i を発行してから命令 n_j を

```
begin
 l_i := 0
 foreach {j| n_j は n_i に依存している }
 begin
  l_i := max(l_i, l_j + d_{i,j})
 end
end
end
```

図 5 最終ノードまでの距離を求めるアルゴリズム
Fig. 5 Algorithm for length to the end node.

発行するまでに待つ必要があるサイクル数である．ここでは，$i < j$ のときに命令 n_i は命令 n_j に依存しないと仮定し，与えられるアセンブリ命令列の出現順に命令 $n_1, n_2, \ldots$ としている．命令 n_j を経由した場合の命令 n_i から最終ノードまでの最長距離は，「命令 n_j から最終ノードまでの最長距離 $+d_{i,j}$」で求まる．命令 n_i に依存するすべての命令についてその命令を経由する場合の最終ノードまでの最長距離を求め，その中で最も大きいものが命令 n_i から最終ノードまでの最長距離となる．命令 n_i の最終ノードまでの最長距離を求めるには命令 n_i に依存するノードを再帰的にたどる必要があるが，$i < j$ のときに命令 n_i は命令 n_j に依存しないと仮定しているため，アセンブリ命令列を逆順にたどることで 1 度の探索で最長距離を求める．

5. 評価実験

本章では，最適解を求めるスケジューリングと準最適解を求めるスケジューリングを行い，それぞれのスケジューリングに必要な時間と得られたスケジュールの実行サイクル数を比較する．また，パイプラインを考慮せずにスケジューリングを行った場合に比べ，ピーク電力をどれだけ削減できるかを示す．

5.1 実験対象の VLIW プロセッサ

実験対象の VLIW プロセッサのアーキテクチャを説明する．消費電力，パイプライン構造，命令セットは Brownie STD 32 [9] をもとに設定した．対象プロセッサは，4 段パイプライン構造で同時発行可能命令数が 4 である．パイプラインステージは IF，ID，EX，WB からなり，IF は命令フェッチ，ID は命令デコード，EX は演算の実行，WB は結果の書き戻しを行う．**表 7** に機能ユニットの構成と各機能ユニットの消費電力を示す．

表 7 において，MUL は乗算器，DIV は除算器，SFT はシフタ，REG はレジスタへのライト，DM_READ はデータメモリからのリードを，DM_WRITE はデータメモリへのライトを，IM_READ は命令メモリからのリードをそれぞれ表す．レジスタには汎用レジスタ，プログラムカウンタ，命令レジスタ，パイプラインレジスタがあり，すべてのレジスタの消費電力は同じであるとする．

MUL 命令（乗算命令）は EX ステージに 5 サイクルか

© 2013 Information Processing Society of Japan

表 7 機能ユニットの構成と消費電力
Table 7 The number and power consumption of each functional unit.

機能ユニット名	同時使用可能数	消費電力 [μW]
ALU	3	837
MUL	1	2,242
DIV	1	1,515
SFT	1	50
REG		2,300
DM_READ	2	19,340
DM_WRITE	2	27,531
IM_READ		44,634

かり，DIV 命令（除算命令）は EX ステージに 32 サイクルかかるとする．表 7 中の消費電力は 1 サイクルあたりの消費電力を表す．表 7 にあげた機能ユニット以外の論理回路やバスの消費電力は，十分小さいとして本実験では考慮しないこととする．

5.2 テストプログラム

命令スケジューリング手法の性能評価に用いたテストプログラムは以下の 4 種である．

- convolution：畳み込みフィルタリングを行うプログラム
- dot_product：2 つのベクトルの積を求めるプログラム
- fir：FIR フィルタリングを行うプログラム
- n_real_update：データの配列に対してアップデートを行うプログラム

これらのプログラムは，組込み分野で使用されるテストベンチである DSPstone [10] のコードを使用した．これらのプログラムのループ部分の基本ブロックを抽出し，命令レベルでの並列度を上げるためにソフトウェアパイプライニングを適用した [11]．

5.3 実験環境

実験に使用した環境は以下のとおりである．スケジューリングアルゴリズムは C 言語で実装した．

- CPU：intel CORE i7
- メモリ：8 GB
- OS：Ubuntu 12.04.1 LTS

5.4 実験結果

表 8 に最適解と準最適解の実行サイクルと実行時間の比較結果を示す．プログラムの隣の括弧内の数字はループ展開を行った回数を表す．

opt は最適解，subopt は準最適解であることを，cycle は実行サイクル数を，time はスケジューリングの実行時間をそれぞれ表す．最適解が 1 時間以内に求まったプログラムについてのみ比較を行っている．表 8 から，最適解が現実的な時間で求まる範囲では準最適解がほぼ最適解に近い値が得られていることが分かる．

表 8 最適解と準最適解の実行サイクル数の比較
Table 8 Comparison of optimal solutions and suboptimal solutions.

power constraints	program	cycle		time	
		opt	subopt	opt	subopt
90 mW	convolution (0)	5	5	1 s	<1 ms
	convolution (1)	10	11	634 s	<1 ms
	dot_product (0)	5	5	1 s	<1 ms
	dot_product (1)	10	11	1,209 s	<1 ms
	fir (0)	8	8	29 s	<1 ms
	n_real_update (0)	8	8	405 s	<1 ms
100 mW	convolution (0)	5	6	1 s	<1 ms
	convolution (1)	10	10	2,353 s	<1 ms
	dot_product (0)	5	6	1 s	<1 ms
	fir (0)	6	8	6 s	<1 ms
	n_real_update (0)	7	8	627 s	<1 ms
110 mW	convolution (0)	5	5	1 s	<1 ms
	convolution (1)	10	10	1,869 s	<1 ms
	dot_product (0)	5	5	1 s	<1 ms
	fir (0)	6	7	10 s	<1 ms
	n_real_update (0)	6	8	417 s	<1 ms
120 mW	convolution (0)	5	5	1 s	<1 ms
	convolution (1)	10	10	1,953 s	<1 ms
	dot_product (0)	5	5	1 s	<1 ms
	fir (0)	5	5	4 s	<1 ms
	n_real_update (0)	6	6	885 s	<1 ms

次に Toburen らの手法と提案手法の比較結果を**表 9** に示す．Toburen らの手法の電力の評価は，本研究での電力モデルを用いて行った．

表 9 において，subopt は準最適解であることを，Toburen は Toburen らの手法を用いて得られた結果であることを，cycle は実行サイクル数を，peak power rate はピーク電力の制約電力に対する割合を，total violation は電力制約を超えたサイクルの制約を超えた分の消費電力の合計をそれぞれ表す．表 9 から，Toburen らの手法では最大 42%の制約電力違反があったことが分かる．また Toburen らの手法と提案手法の実行サイクル数が離れるほど Toburen らの手法では電力制約を大きく違反しているといえる．total violation の値からは Toburen らの手法の制約電力違反が極端に短いサイクルの間だけ起こるものではないことが読み取れる．制約電力が 90 mW のときの n_real_update (2) の実行サイクル数は 18 であり total violation は 205.63 mW であるので，1 サイクルあたり平均 10 mW 以上の違反が起こったことが分かる．

6. おわりに

本論文では，VLIW プロセッサのための電力制約を考慮した命令スケジューリング問題を電力制約下で実行サイク

© 2013 Information Processing Society of Japan

表 9 Tobure らの手法と提案手法の比較結果
Table 9 Comparison of Toburen's method and proposed method.

power constraints	program	cycle		peak power rate		total violation of Toburen
		subopt	Toburen	subopt	Toburen	[mW· cycle]
90 mW	convolution (0)	5	5	0.97	1.12	11.02
	convolution (1)	11	10	0.96	1.18	28.57
	convolution (2)	15	15	0.98	1.23	57.95
	dot_product (0)	5	6	0.97	1.16	14.16
	dot_product (1)	11	10	0.99	1.25	43.94
	dot_product (2)	17	15	0.97	1.26	70.79
	fir (0)	8	5	0.98	1.26	59.95
	fir (1)	17	11	1.00	1.31	143.64
	fir (2)	24	15	0.98	1.33	226.63
	n_real_update (0)	8	7	1.00	1.34	70.88
	n_real_update (1)	16	12	1.00	1.31	155.92
	n_real_update (2)	24	18	1.00	1.31	205.63
100 mW	convolution (0)	6	5	0.99	1.01	1.02
	convolution (1)	10	10	0.99	1.06	6.85
	convolution (2)	15	15	0.99	1.11	27.95
	dot_product (0)	6	6	0.99	1.09	14.59
	dot_product (1)	13	10	1.00	1.13	14.38
	dot_product (2)	18	15	1.00	1.13	22.07
	fir (0)	8	5	0.99	1.13	29.69
	fir (1)	14	11	0.99	1.18	74.92
	fir (2)	24	15	0.99	1.34	150.31
	n_real_update (0)	8	7	0.99	1.36	50.66
	n_real_update (1)	18	12	0.99	1.42	108.90
	n_real_update (2)	24	17	0.99	1.36	139.43
110 mW	convolution (0)	5	5	0.96	0.96	0.00
	convolution (1)	10	10	1.00	1.00	0.43
	convolution (2)	15	15	1.00	1.03	6.09
	dot_product (0)	5	5	0.96	0.96	0.00
	dot_product (1)	11	10	0.98	1.04	4.82
	dot_product (2)	16	15	0.98	1.06	6.50
	fir (0)	7	5	1.00	1.05	5.03
	fir (1)	11	11	1.00	1.27	90.22
	fir (2)	17	16	1.00	1.26	103.55
	n_real_update (0)	8	6	1.00	1.28	41.19
	n_real_update (1)	14	11	1.00	1.32	97.96
	n_real_update (2)	20	16	0.99	1.32	160.35
120 mW	convolution (0)	5	5	0.88	0.88	0.00
	convolution (1)	10	10	0.92	0.92	0.00
	convolution (2)	15	15	0.94	0.94	0.00
	dot_product (0)	5	5	0.88	0.88	0.00
	dot_product (1)	10	10	0.96	0.96	0.00
	dot_product (2)	15	15	0.97	0.97	0.00
	fir (0)	5	5	0.96	0.96	0.00
	fir (1)	11	10	0.97	1.15	44.74
	fir (2)	15	15	0.98	1.19	79.36
	n_real_update (0)	6	6	0.99	1.17	21.19
	n_real_update (1)	11	11	0.97	1.21	59.95
	n_real_update (2)	16	16	0.99	1.21	104.87

© 2013 Information Processing Society of Japan

ル数を最小化する最適化問題と定式化し，この問題の最適解と準最適解を求めるアルゴリズムを示した．スケジューリング対象の命令が多く最適解が求まらない場合にも準最適解が現実的な時間で求まることが確認できた．またパイプラインを考慮しない先行手法と比較し，より厳密に消費電力の制約を守ることを確認した．今後の課題としては，電力モデルをより詳細化することが考えられる．電力モデルの例としては，バスの消費電力に着目すること [12]，命令の実行順に着目すること [13]，命令フェッチの際の消費電力のばらつきを考慮すること [14] などがあげられる．

参考文献

[1] Fisher, J.A., Faraboschi, P. and Young, C.: *Embedded Computing*, ELSEVIER (2005).

[2] Xiao, S. and Lai, E.M.-K.: Instruction scheduling of VLIW architectures for balanced power consumption, *Proc. 2005 Asia and South Pacific Design Automation Conference*, *ASP-DAC '05*, New York, NY, USA, ACM, pp.824–829 (online), DOI: http://doi.acm.org/10.1145/1120725.1121027 (2005).

[3] Wang, M., Shao, Z., Liu, H. and Xue, C.J.: Minimizing Leakage Energy with Modulo Scheduling for VLIW DSP Processors, *DISTRIBUTED EMBEDDED SYSTEMS: DESIGN, MIDDLEWARE AND RESOURCES*, IFIP The International Federation for Information Processing, Vol.271, pp.111–120, Springer US (online), DOI: 10.1007/978-0-387-09661-2_11 (2008).

[4] Toburen, M., Conte, T.M. and Reilly, M.: Instruction Scheduling for Low Power Dissipation in High Performance Microprocessors, *Proc. Power Driven Microarchitecture Workshop in Conjunction with the ISCA '98* (1998).

[5] He, Y., Pu, Y., Kleihorst, R., Ye, Z., Abbo, A.A., Londono, S.M. and Corporaal, H.: Xetal-Pro: An ultra-low energy and high throughput SIMD processor, *Proc. 47th Design Automation Conference*, *DAC '10*, New York, NY, USA, ACM, pp.543–548 (online), DOI: 10.1145/1837274.1837409 (2010).

[6] Aho, A.V., Lam, M.S., Sethi, R. and Ullman, J.D.: *Compilers*, Pearson Education (2006).

[7] 竹内陽一郎，境　隆二：命令スケジューリングアルゴリズム，全国大会講演論文集，Vol.45, No.5, pp.59–60（オンライン）(1992)，入手先 ⟨http://ci.nii.ac.jp/naid/110002889831/⟩.

[8] Su, C.-L., Tsui, C.-Y. and Despain, A.: Low power architecture design and compilation techniques for high-performance processors, *Compcon Spring '94, Digest of Papers*, pp.489–498 (1994).

[9] 岩戸宏文，稗田拓路，田中浩明，佐藤　淳，坂主圭史，武内良典，今井正治：ASIP 短期開発のための高い拡張性を有するベースプロセッサの提案，技術報告 114，アーキテクチャ合成，デザインガイア 2007—VLSI 設計の新しい大地を考える研究会 (2007).

[10] živojnović, V., Velarde, J.M., Schläger, C. and Meyr, H.: DSPstone: A DSP-Oriented Benchmarking Methodology, *Proc. ICSPAT'94 - Dallas* (1994).

[11] Lam, M.: Software pipelining: an effective scheduling technique for VLIW machines, *Proc. ACM SIGPLAN 1988 Conference on Programming Language Design and Implementation*, *PLDI '88*, New York, NY, USA, ACM, pp.318–328 (online), DOI: 10.1145/53990.54022 (1988).

[12] Lee, C., Lee, J.K., Hwang, T. and Tsai, S.-C.: Compiler optimization on VLIW instruction scheduling for low power, *ACM Trans. Design Automation of Electronic Systems*, Vol.8, No.2, pp.252–268 (online), DOI: 10.1145/762488.762494 (2003).

[13] Lee, M.T.-C., Tiwari, V., Malik, S. and Fujita, M.: Power analysis and low-power scheduling techniques for embedded DSP software, *Proc. 8th International Symposium on System Synthesis*, *ISSS '95*, New York, NY, USA, ACM, pp.110–115 (online), DOI: http://doi.acm.org/10.1145/224486.224525 (1995).

[14] Shin, D., Kim, J. and Chang, N.: An operation rearrangement technique for power optimization in VLIW instruction fetch, *Proc. Design, Automation and Test in Europe, Conference and Exhibition 2001, DATE '01*, Vol.97, p.809, IEEE (2001).

藤井 裕也

2012 年大阪大学基礎工学部情報科学科卒業．現在，同大学院博士前期課程在学中．VLIW プロセッサのコンパイラに関する研究に興味を持つ．

武内 良典　（正会員）

1987 年東京工業大学工学部電気電子工学科卒業．1992 年同大学院博士後期課程修了．博士（工学）．1996 年大阪大学大学院基礎工学研究科情報数理系専攻講師．現在，同大学院情報科学研究科准教授．VLSI 設計および VLSI CAD の研究に従事．IEEE，電子情報通信学会各会員．

今井 正治　（正会員）

1974 年名古屋大学工学部電気工学科卒業．1979 年同大学院博士後期課程修了（工学博士）．同年豊橋技術科学大学奉職．1994 年同教授．1996 年大阪大学大学院基礎工学研究科情報数理系専攻教授．その間，1984 年から 1985 年にかけて米国サウスカロライナ大学工学部電気計算機工学科客員助教授（文部省在外研究員）．これまで組合せ最適化アルゴリズム，ハードウェア/ソフトウェア協調設計等の研究に従事．1991 年より日本電子機械工業会および IEEE/DASC において EDA 標準化作業に従事．現在，情報処理学会設計自動化研究会主査．IEEE，ACM，電子情報通信学会，人工知能学会各会員．

© 2013 Information Processing Society of Japan