Vol.2010-EMB-18 No.10
2010/8/10

実行トレースマイニングを用いたタスク内DVFSに有効なチェックポイント抽出手法

立松知紘[†1]　高瀬英希[†1,†2]　曾剛[†3]
冨山宏之[†4]　高田広章[†1]

タスク内動的電圧・周波数制御（DVFS: Dynamic Voltage and Frequency Scaling）による消費エネルギー削減効果を向上させるには，プログラム内の残り最悪実行サイクル数を正確に見積もることが重要である．さらに，周波数の切替えを行うチェックポイントを，適切に定める必要がある．本稿では，実行トレースマイニングによって見積もった残り最悪実行サイクル数から，DVFS に効果的な適切なチェックポイントを選定する手法を提案する．チェックポイントの選定には，グリーディ法に基づいた手法を適用する．評価実験では，MediaBench の jpeg エンコーダとデコーダのベンチマークに対して提案手法による DVFS を適用し，その有効性を確認した．

Checkpoints Extraction for Efficient Intra-task DVFS with Execution Trace Mining

Tomohiro Tatematsu,[†1] Hideki Takase,[†1,†2]
Gang Zeng,[†3] Hiroyuki Tomiyama[†4]
and Hiroaki Takada[†1]

It is important to estimate the remaining worst case execution cycles (RWCEC) in the program accurately to improve the effect on energy reduction of the intra-task dynamic voltage and frequency scaling (DVFS). Moreover, it is crucial to decide where and how many checkpoints in which frequency can be changed should be inserted into the program. This is because the inserted checkpoints involve time and energy overhead for calculation and DVFS. For this reason, we propose a greedy algorithm in this paper for selecting effective checkpoints. This algorithm is based on the results of execution trace mining, and experimental results on two benchmark programs of jpeg encoder and decoder from MediaBench have validated its effectiveness.

1. はじめに

今日の組込みシステムでは，性能向上や大規模・複雑化が進んでおり，システムの消費エネルギーが増加の一途を辿っている．消費エネルギーの低減は，システムの信頼性の向上や運用コストの抑制につながる．特に，バッテリを電源とする組込みシステムでは，性能のみならず連続駆動時間の向上にも貢献する．以上のことから，消費エネルギー削減の要求は，今日では大きいものとなっている．そこで，これらの要求に対応するべく，組込みシステムの消費エネルギーを最適化を目指す研究がこれまでに数多く行われている．

組込みシステムの消費エネルギー削減技術の 1 つに，動的電圧・周波数制御（DVFS: Dynamic Voltage and Frequency Scaling）がある．DVFS とは，プログラム実行時にプロセッサへの周波数および供給電圧を切替える技術のことである．電圧は周波数と比例関係にあり，消費エネルギーは周波数の 2 乗と実行サイクル数の積に比例すると近似できる[1)]．ゆえに，DVFS によって周波数を降下させることによって，組込みシステムの消費エネルギーを削減することができる．しかし，周波数の降下は実行時間の増大に直結する．組込みシステムでは，デッドラインと呼ばれる特定の時刻までに処理を終えなければならないという時間制約がある．このため，この時間制約を保証しつつ周波数を適切に設定する必要がある．

時間制約の下で DVFS によって効率良く消費エネルギーを削減するためには，周波数を切り替える箇所の決定が重要である．以下，本稿では，DVFS にて周波数を切り替えるプログラムの箇所のことを，チェックポイントと呼ぶ．プログラムがチェックポイントに到達すると，デッドラインまでの時間およびチェックポイント以降の残り最悪実行サイクル数の参照，最適な周波数の計算，そして，必要であれば周波数の切替えの処理が行われる．これらの処理は，実行時間と消費エネルギーのオーバヘッドがかかる．

デッドラインまでの余裕時間が生まれると，周波数を降下させることができる．つまり，チェックポイントは，残り最悪実行サイクル数の見積もりが減少し，デッドラインまでの余裕

†1 名古屋大学 大学院情報科学研究科
Graduate School of Information Science, Nagoya University
†2 日本学術振興会
The Japan Society for the Promotion of Science
†3 名古屋大学 大学院工学研究科
Graduate School of Engineering, Nagoya University
†4 立命館大学 理工学部
College of Science and Engineering, Ritsumeikan University

時間が増える場所に挿入されるのが望ましい．プログラムの残り最悪実行サイクル数は，分岐命令においてその見積もりが変動しうる．これは，分岐命令の振舞い（not taken/taken）によって，後続の処理内容が変わりやすいためである．そこで，残り最悪実行サイクル数が変動する箇所を探索するには，分岐命令の振舞いに応じた残り最悪実行サイクル数の正確な見積もりが必要になる．

我々はこれまでに，DVFS の効果を向上させるための技術として実行トレースマイニングを提案している[2)]．実行トレースマイニングとは，大量の実行トレース群から有益な情報を抽出する処理を指す．実行トレースは，実行された一連の命令の情報を保持している．そのため，実行されるパスや分岐の共起関係を把握し，より正確な残り最悪実行サイクル数の見積もりができる．しかし，文献 2) では，チェックポイントにおけるオーバヘッドの考慮が不充分であり，さらに，同一の分岐命令が何度か実行される場合に対応できていなかった．

本稿では，チェックポイントでのオーバヘッドを考慮に入れ，消費エネルギー削減効果の高いチェックポイントを選定する手法を提案する．チェックポイントでは，後続の周波数の計算や周波数の切替えを行うため，時間遅延と消費エネルギーのオーバヘッドが存在する．そのため，チェックポイントを多く挿入すると，このオーバヘッドの影響が無視できないものになり得る．ゆえに，本研究ではチェックポイント候補の中から特に消費エネルギー削減に効果のあるものを，グリーディ法に基づいた手法により選定していく．さらに，本稿では，実行トレースマイニングにおいて，同一の命令の実行順を考慮する．このことにより，ループ構造や関数呼び出しを含むプログラムに対応できるようになる．本研究では，1 タスクで構成されているプログラムを対象とする．

本論文の構成は以下の通りである．まず，2 章にて既存研究について触れる．3 章では DVFS について解説する．4 章で，提案手法の全体を説明する．5 章で，過去に提案された実行トレースマイニングについて解説する．6 章では，抽出したチェックポイント候補から，チェックポイントを選出する処理について述べる．7 章で，提案手法の有効性を検証するための評価実験について述べる．最後に，8 章にてまとめと今後の方針について触れる．

2. 関 連 研 究

これまで，実行トレースからプログラムの性質を捉える研究と，DVFS によって消費エネルギー最適化を目指す手法が数多く提案されている．本章では，今日までに提案されてきたこれらの研究について述べる．

文献 3) では，プログラムの実行トレースを取得する際の処理時間を低減する方法を提案している．プログラム内にラベルを出力する命令を挿入することにより，実行トレースから特定のパスが何度実行されるかのみを調べる．文献 4) では，実行トレースからプログラム内で実行頻度の高いホットサブパスと呼ばれる区間の抽出法を提案している．この手法では，プログラムの CFG 上で通過するパスにラベルを付け，実行トレースで取得したラベル列を解析する．この研究の特徴は，ラベル列の解析の際に文脈自由文法に置換した上に無閉路有向グラフを生成することで解析を容易化していることである．

DVFS の既存研究について触れる．文献 1)，5)，6) では，コンパイル時に作成された対象プログラムの CFG から，各ノードにおける残り実行サイクル数の見積もりとデッドラインまでの時間から周波数を定める手法を提案している．残り実行サイクル数として，文献 5) では残り最悪実行サイクル数，文献 6) では最も通る可能性の高いパスの実行サイクル数，文献 1) では条件分岐での分岐確率や分岐先の残り実行サイクル数から，平均消費エネルギーが最小となる実行サイクル数を見積もっている．そして，残り実行サイクル数が変化するノード間に，プログラム実行前に周波数を切替え命令を挿入し，実行時に，残り実行サイクル数の変化量によって周波数を切替えている．

これらの DVFS の既存手法は，CFG 上の残り実行サイクル数と分岐確率に着目しているだけであり，実際には通り得ないパスの実行サイクル数を見積もっている．我々の提案している実行トレースマイニング[2)] は，分岐の共起関係を把握することができ，実行パスの残り実行サイクル数をより正確に見積もることが可能である．

3. 動的電圧・周波数制御

3.1 消費エネルギーと周波数

本節では，消費エネルギーと周波数の関係について述べる．はじめに，ゲートの出力の変化による動的な消費電力 $P_{dynamic}$ について説明する．これは，プロセッサなどの CMOS 回路における消費電力の中で多くの割合を占めるものであり，

$$P_{dynamic} = \alpha \cdot C_L \cdot V_{dd}^2 \cdot f \tag{1}$$

と近似することができる．ここで，α はスイッチング確率，C_L は回路の静電容量，V_{dd} は回路の供給電圧，f は周波数を指す．次に，動的な消費エネルギー $E_{dynamic}$ は，動的な消費電力の時間積分であらわされ，$P_{dynamic}$ は時間に依存しない定数であると仮定すると，

$$E_{dynamic} = \int_0^T P_{dynamic}(t)dt = \alpha \cdot C_L \cdot V_{dd}^2 \cdot f \cdot T \tag{2}$$

が得られる．T の間，f で命令を実行し続けた場合の実行サイクル数を N_C とすると，$E_{dynamic}$ は次の通りとなる．

$$E_{dynamic} = \alpha \cdot C_L \cdot V_{dd}^2 \cdot f \cdot T = \alpha \cdot C_L \cdot V_{dd}^2 \cdot N_c \quad (3)$$

続いて，V_{dd} と f の関係について述べる．式 (3) より，$E_{dynamic}$ は V_{dd} の 2 乗に比例しているため，V_{dd} を下げることは消費エネルギーの削減につながる．しかし，供給電圧を下げると回路遅延時間 τ が増大し，これに伴い周波数が低くなる．この関係を示すと，

$$\frac{1}{f} \propto \tau \propto \frac{V_{dd}}{(V_{dd} - V_{th})^2} \quad (4)$$

の通りになる．ここで，V_{th} は閾値電圧である．簡単のため，$V_{th} \ll V_{dd}$ と仮定すると，V_{dd} と f の関係は，

$$f \propto V_{dd} \quad (5)$$

と近似できる．式 (3)(5) より，$E_{dynamic}$ と f および N_C の関係は，次の通りになる[7]．

$$E_{dynamic} \propto f^2 \cdot N_c \quad (6)$$

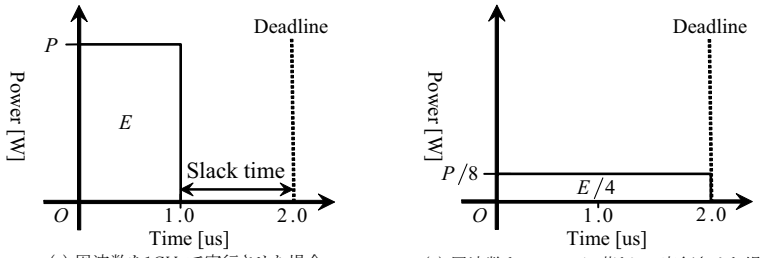
ゆえに，動的な消費エネルギー $E_{dynamic}$ を削減するためには周波数 f を下げることが有効であることがわかる．

3.2 消費エネルギーと周波数

本節では，組込みシステムでの DVFS における電圧・周波数の決定方法について述べる．本研究におけるタスク内 DVFS とは，タスク実行内での DVFS を指す．

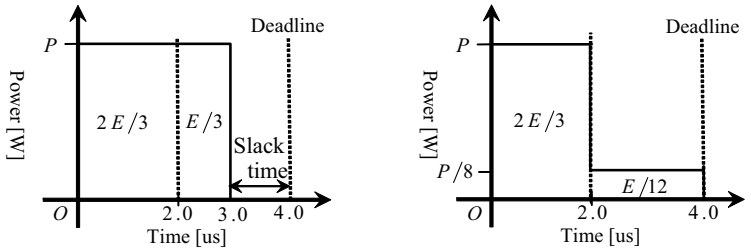
3.1 節で述べたように，消費エネルギー削減には，DVFS によって周波数を下げることが有効である．しかし，周波数を下げることは，実行時間の増大につながる．リアルタイム性を持つ組込みシステムでは，デッドラインまでにタスクの実行を終えなければならない．そのため，デッドラインの時間制約を守るような周波数を設定する必要がある．

デッドラインの時間制約の下では，式 (6) より，デッドラインに実行を終えるように周波数を設定した場合が最も消費エネルギーが削減できる．例えば，デッドラインが 2.0 us で全体の実行サイクル数が 1000 サイクルのタスクを考える．仮に周波数を 1.0 GHz として最後まで実行すると，デッドラインまで 1.0 us の時間的な余裕がある（**図 1(a)**）．タスクの実行が終了する時刻からデッドラインまでの残り時間をスラック時間と呼ぶ．デッドラインの時間制約の下で消費エネルギーを最適にするためには，スラック時間を有効に使い，デッドラインで実行を終える周波数で実行する必要がある．そのため，全体の実行サイクル数をデッドラインまでの残り時間で割った商を周波数として設定する．この例の場合，周波数を 500 MHz に設定すれば，デッドラインの時間制約の下で消費エネルギーが最適になる



(a) 周波数を1GHzで実行させた場合　(b) 周波数を500MHzに落として実行させた場合

図 1 DVFS による消費電力および消費エネルギーの変化
Fig. 1 The alternation of power and energy consumption with DVFS



(a) 実行終了まで周波数を1GHzで実行させた場合　(b) 2.0usで周波数を500MHzに落として実行させた場合

図 2 残り最悪実行サイクル数が変動する場合の DVFS による消費電力および消費エネルギーの変化
Fig. 2 The alternation of power and energy consumption with DVFS in case of variation of RWCEC

（**図 1(b)**）．このとき，式 (6) より，1.0 GHz で実行する場合に比べて，消費エネルギーは 4 分の 1 にまで削減できる．

実際のタスクでは，条件分岐やループ脱出によって残り実行サイクル数の見積もりが実行中に変動することがある．例えば，デッドラインまでの残り時間が 4.0 us で最初に残り実行サイクル数を 4000 サイクルと見積もった場合を考える．スラック時間を有効に活用するために，周波数を 1.0 GHz と設定して実行を開始する．続いて，2.0 us だけ時間が経過したところで，分岐命令での振舞いやループの脱出などにより，残り実行サイクル数の見積もりが 2000 サイクルから 1000 サイクルと変化したと仮定する．このとき，1.0 GHz のまま実行を続けると，1.0 us のスラック時間が生じることになる（**図 2(a)**）．よって，消費エネルギーを効果的に削減するには，残り実行サイクル数の見積もりが 1000 サイクルに減少した際に，デッドラインまでの残り時間である 2.0 us から，周波数を 500 MHz と設定すれ

ばよい．この場合，スラック時間を有効に活用することができ，消費エネルギーを削減させることが出来る（**図 2(b)**）．

4. 提案手法の全体像

本章では，提案手法全体の概要について述べる．提案手法のワークフローを**図 3** に示す．提案手法は，実行トレースマイニングによるチェックポイント候補の抽出，チェックポイント候補から DVFS に特に効果のあるチェックポイントの選定．および DVFS によるチェックポイントが挿入されたプログラムの実行の順に行われる．

実行トレースマイニングは，実行トレース群からマイニングテーブルを作成するフェーズと，取得したマイニングテーブルと実行トレース群からチェックポイント候補を抽出するフェーズから構成される．詳細は，5 章で述べる．

実行トレースマイニングによって生成されたチェックポイント候補リストから，DVFS において特に効果のあるチェックポイント候補を選定する．本研究では，このチェックポイント選定方法にグリーディ法を用いた手法を提案する．チェックポイント候補は，DVFS に効果のある順にチェックポイントリストとしてまとめられる．詳細は，6 章で述べる．

挿入するチェックポイントの個数は，設計者が決定することができる．チェックポイントリストから指定した数までの順位のチェックポイントが，対象となるプログラムに挿入される．プログラムの実行がチェックポイントに到達すると，チェックポイント以降の残り最悪実行サイクル数と，デッドラインまでの時間を参照する．そして，これらの値から周波数を切替えるかどうかを判断し，周波数が降下できる場合に周波数を切替え，システムの消費エネルギーを削減していく．

5. 実行トレースマイニング

5.1 マイニングテーブル作成のフェーズ

本節では，実行トレースの取得からマイニングテーブルの作成までの過程について述べる．

実行トレースは，対象となるプログラムに入力データを与え，サイクルレベルシミュレーションを行うことで取得する．実行トレースからは，分岐命令のアドレス，同一の分岐命令の実行順，振舞い，および残り実行サイクル数の情報を抽出する．同一の分岐命令の実行順とは，同一の分岐命令が何度も実行される場合に，それが何度目の実行かを識別するものである．我々が過去に提案した実行トレースマイニングでは，この情報は付加していなかった．振舞いは，実行トレースにおいて，分岐での命令アドレスとその次の命令のアドレス

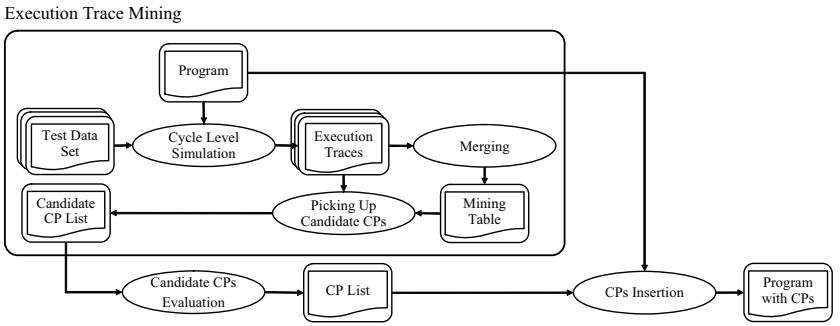


図 3 提案手法のワークフロー
Fig. 3 The workflow of our proposed approach

が連続していれば not taken，連続していなければ taken である．残り実行サイクル数は，全体の実行サイクル数と注目している命令の実行サイクルの差として求まる．全体の実行サイクル数は，実行トレースの最終命令の実行サイクルから取得する．

マイニングテーブルは，実行トレース群のマージによって作成される．マイニングテーブルとは，各分岐命令の振舞いにおける残り実行サイクル数と，タスク全体のサイクル数をまとめたテーブルのことである．例えば，**図 4(a)** の実行トレース群の場合，これらをマージすることで，**図 4(b)** のマイニングテーブルが生成される．マイニングテーブルの作成の詳細については，文献 2) を参照されたい．

5.2 チェックポイント候補抽出のフェーズ

本節では，作成したマイニングテーブルからチェックポイント候補となる箇所を抽出する処理について述べる．

DVFS によって消費エネルギーを削減するためには，タスクの残り最悪実行サイクル数の見積もりが減少する箇所にチェックポイントを置くと効果的である．なぜなら，残り最悪実行サイクル数の見積もりが減少することでスラック時間が発生し，より低い周波数に切り替えることができるためである．残り最悪実行サイクル数は，分岐命令の振舞いによって変動する．そのため，実行トレース群とマイニングテーブルから，残り最悪実行サイクル数の見積もりが減少する分岐命令のジャンプ先を探索し，チェックポイント候補とする．

チェックポイント候補の抽出過程を**図 5** を例にとって説明する．図 5(a) の CFG を持ったプログラムに対して，図 5(b) の実行トレース群が取得できると仮定する．図 5(c) のマイニングテーブルはこの実行トレース群から作られる．チェックポイント候補は，このマイニ

Execution Trace No.1		
Execution Cycles: 1000 cycles		
Address	Condition	REC
0x0248(1)	taken	850 cycles
0x0248(2)	taken	650 cycles
0x0248(3)	n_taken	450 cycles
0x026c(1)	taken	350 cycles
0x02a0(1)	n_taken	50 cycles

Execution Trace No.2		
Execution Cycles: 550 cycles		
Address	Condition	REC
0x0248(1)	n_taken	400 cycles
0x026c(1)	n_taken	300 cycles
0x0294(1)	n_taken	200 cycles
0x02a0(1)	taken	100 cycles

Execution Trace No.3		
Execution Cycles: 1000 cycles		
Address	Condition	REC
0x0248(1)	taken	550 cycles
0x0248(2)	n_taken	350 cycles
0x026c(1)	n_taken	250 cycles
0x0294(1)	n_taken	150 cycles
0x02a0(1)	n_taken	50 cycles

(a) 実行トレース群 (REC:Remaining Execution Cycles)

Mining Table		
Worst Case Execution Cycles: 1000cycles		
Address	RWCEC	
	n_taken	taken
0x0248(1)	400 cycles	850 cycles
0x0248(2)	350 cycles	650 cycles
0x026c(1)	300 cycles	350 cycles
0x02a0(1)	50 cycles	100 cycles

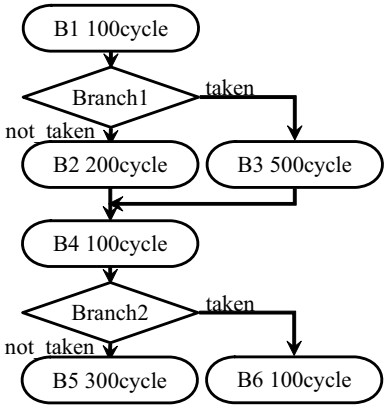
(b) (a)の実行トレース群から生成されたマイニングテーブル

図 4 実行トレースのマージによるマイニングテーブルの作成
Fig. 4 Merging execution traces into a mining table

ングテーブルを用いて，各実行トレースに対して残り実行サイクル数の変動をシミュレートすることで抽出する．

例えば，実行トレース 1 に相当する実行パスを通過する場合を考える．まず，実行開始時は，マイニングテーブルから残り最悪実行サイクル数は 800 サイクルであると見積もる．次に，分岐 1 が not taken だった場合，実行を開始してから 100 サイクルだけ実行されているため，この時点では残り最悪実行サイクル数を 700 サイクルと見積もっている．しかし，マイニングテーブルを参照すると，分岐 1 が not taken だった場合，残り最悪実行サイクル数は 600 サイクルである．すなわち，分岐 1 で not taken のジャンプ先では残り最悪実行サイクル数が減少する．ゆえに，この箇所をチェックポイント候補に含める．分岐 2 に到達したときは，分岐 1 から 300 サイクル実行されているため，残り最悪実行サイクル数の見積もりは 300 サイクルとなっている．しかし，ここで not taken でジャンプした場合，マイニングテーブルを参照しても，残り最悪実行サイクル数の見積もりは減少しない．そのため，この場所はチェックポイント候補に含めない．

チェックポイント候補の抽出では，以上の操作を，入力に用いた全ての実行トレースに対して行う．その際，一度でも残り最悪実行サイクル数が減少した箇所をチェックポイント候補として取り扱う．そして，チェックポイント候補に挙がったものは，チェックポイント候補リストに記録される．



(a) プログラムのCFG

Execution Trace No.1		
Execution Cycles: 700		
Address	Condition	REC
Branch1(1)	n_taken	600
Branch2(1)	n_taken	300

Execution Trace No.2		
Execution Cycles: 800		
Address	Condition	REC
Branch1(1)	taken	700
Branch2(1)	taken	100

(b) 実行トレース群

Mining Table		
Worst Case Execution Cycles : 800		
Address	Remaining Worst Case Execution Cycles	
	n_taken	taken
Branch1(1)	600	700
Branch2(1)	300	100

(c) 実行トレース群から生成されたマイニングテーブル

図 5 例題プログラムから取得できる実行トレースとマイニングテーブル
Fig. 5 The execution traces and the mining table from a program

6. DVFS に有効なチェックポイントの選定

6.1 チェックポイントでの処理

本節では，タスク実行時にチェックポイント内で行われる処理について触れる．チェックポイントでは以下の処理が行われる．

- チェックポイント以降のタスクの残り最悪実行サイクル数の参照
- チェックポイントからデッドラインまでのタスクの残り時間の参照
- チェックポイント後の周波数を計算し，周波数が下げられるならば，周波数を切替え

周波数を切替える際には時間的なオーバヘッドが発生する．文献 7) によれば，商用のプロセッサでは，おおよそ 200 us から 500 us ほどのオーバヘッドがあるといわれている．さらに，チェックポイントでは，周波数を切替えなくとも，前者 2 つの情報を参照するための命令の実行による遅延時間や消費エネルギーのオーバヘッドが発生する．そのため，チェックポイント後の周波数はこれらのオーバヘッドを考慮した上で計算される．商用のプロセッサ

が設定できる周波数は，離散的な値をとる[⋆1]．そこで，設定できる周波数を値が小さいものから $\{f_0, f_1, \cdots, f_n\}$ とすると，チェックポイントで計算される周波数は次の通りになる．

$$f_{tmp} = \frac{N_{remaining} + N_{overhead}}{D - t - \Delta t} \tag{7}$$

$$f_{next} = \begin{cases} f_0 & (f_{tmp} \leq f_0) \\ min\{f_{current}, f_i\} & (f_{i-1} \leq f_{tmp} < f_i, i \in \{1, 2, \cdots, n\}) \\ f_n & (otherwise) \end{cases} \tag{8}$$

ここで，$N_{remaining}$ は対象となるチェックポイント以降の残り最悪実行サイクル数，$N_{overhead}$ は 1 つのチェックポイントにおける実行サイクル数のオーバヘッド，t は現在時刻，Δt は周波数を切替えた場合の遅延時間，D はデッドラインの時刻，そして，$f_{current}$ は現在の周波数である．つまり，設定しうる周波数が離散値であるときに式 (7) を用いて周波数を計算する場合，チェックポイント到達時の周波数と後続の周波数が変わらない場合がある．そのため，式 (7) の結果から，周波数が下げられる場合のみ周波数を切替える．

6.2 チェックポイントの順位付け

本節では，抽出したチェックポイント候補から，DVFS の効果を向上させるチェックポイントを選定する過程について説明する．

6.1 節で述べたように，チェックポイントでは，遅延時間および消費エネルギーのオーバヘッドが存在する．そのため，プログラム中にチェックポイントを過剰に挿入すると，消費エネルギーの削減効果が減少する可能性がある．そのために，チェックポイント候補から消費エネルギー削減効果の高いチェックポイントを選定する必要がある．本研究では，実行トレースマイニングで抽出されたチェックポイント候補を順位付けすることによって，消費エネルギー削減効果の高いチェックポイントを選定する．

チェックポイントの順位付けは，グリーディ法に基づいて行われる．まず，チェックポイント候補の中から 1 つ候補を選択する．そして，そのチェックポイント候補に対してチェックポイント関数を挿入すると仮定する．チェックポイント候補がリストに N 個記録されている場合，チェックポイントが 1 つ挿入された N 通りのプログラムを考える．例えば，対象となるプログラムが**図 6(a)** のような CFG とチェックポイント候補を持っていると仮定する．この場合，**図 6(b)** のように，3 通りのチェックポイントの挿入が考えられる．

次に，チェックポイント関数が挿入されたプログラムに対して，実行トレースマイニングに用いた入力データを与えて，DVFS を試行する．このシミュレートでは，特定のデッドラインや周波数切替えに伴うオーバヘッドに限定しないように，理想的な DVFS の環境を想定して行われる．すなわち，チェックポイント関数実行に伴うオーバヘッドは考慮せず，設定できる周波数を連続的と仮定する．さらに，デッドラインは 1 単位時間とする．

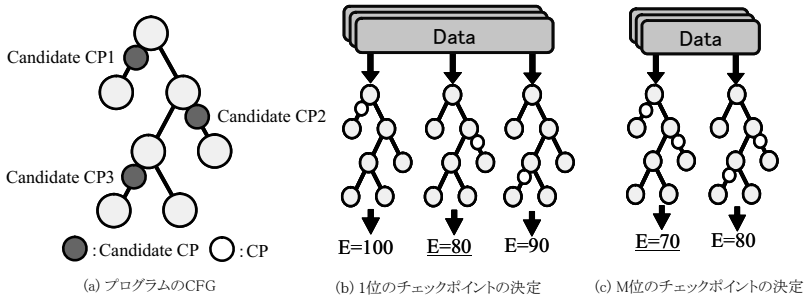


図 6 チェックポイントの順位付け
Fig. 6 An example for ranking checkpoints

最後に，最も消費エネルギーが少なかったプログラムに挿入されたチェックポイント候補を 1 位のチェックポイントとする．例えば，図 6(b) では，チェックポイント候補 2 を挿入した場合が最も消費エネルギーが少ないという結果が得られている．そのため，この場合ではチェックポイント候補 2 が 1 位のチェックポイントと認定される．

2 位以下のチェックポイントの選定についても，同様の方法が用いられる．2 位のチェックポイントの決定を行う場合は，まず，すでに順位付けされた 1 個のチェックポイントと，新たに選んだチェックポイント候補 1 つに対して，チェックポイント関数を挿入すると仮定する．**図 6(c)** の例では，すでに 1 位と認定されたチェックポイント候補 2 と，残り 2 つのチェックポイント候補のどちらかが挿入された 2 通りのプログラムを示している．続いて，1 位のチェックポイント選定する際と同様に，実行トレースマイニングで用いた入力データ群を与えて DVFS をシミュレートする．この結果，チェックポイント候補 1 を挿入した場合が，最も消費エネルギーが少なくなる．そのため，チェックポイント候補 1 が 2 位のチェックポイントとなる．以下同様に，M 位のチェックポイントを M-1 個のチェックポイント候補から選定していく操作を，N 回繰り返していく．

順位付けされたチェックポイント候補は，選定の際に決定された順位の順にチェックポイントリストに記録される．チェックポイントリストには，チェックポイントを挿入する命令

⋆1 われわれの既存研究[2] では，プロセッサのとる周波数は連続的な値であるという仮定を置いていた．

Vol.2010-EMB-18 No.10
2010/8/10

表 1 評価実験で用いた各ベンチマークのデッドライン
Table 1 Deadlines of the cjpeg and djpeg benchmarks

Slack Time	Deadlines	
	cjpeg	djpeg
No slack	45594 us	12747 us
10 % of deadline	50660 us	14163 us
20 % of deadline	56992 us	15933 us
30 % of deadline	65134 us	18209 us

のアドレスと，実行回数，および残り最悪実行サイクル数が記録される．

7. 評価実験

提案する実行トレースマイニングを用いたチェックポイント選定手法の有効性を示すために，評価実験を行った．本章では，評価実験の手順と結果，および考察について述べる．

7.1 実験環境

評価実験では，提案手法を用いて DVFS を行った場合の消費エネルギー量で評価した．実験対象プログラムは，MediaBench から jpeg のエンコード（cjpeg）およびデコード（djpeg）のプログラムを採用した．各プログラムに対して 100 種類の画像を用意し，入力データとして用いた．入力データの画像は，縦横の大きさは全て同じものとした．サイクルレベルシミュレータとして，SimpleScalar/ARM[8] を用いた．SimpleScalar/ARM は組込みシステムのプロセッサとして広く使われている ARM7TDMI[9] の命令セットシミュレータである．

実験では以下を仮定した．プロセッサが設定できる周波数は，10 MHz 刻みで {10 MHz, 20 MHz, …, 100 MHz} とした．チェックポイントでは，周波数が式 (7) で切替えられるものとした．また，チェックポイントのオーバヘッドを 1000 サイクル，周波数切替えにかかる遅延時間を 300 us と仮定した．デッドラインまでの時間は，**表 1** に示すように，最大周波数である 100 MHz（Highest Speed）で実行した際にスラックが生じない時間と，スラック時間がデッドラインまでの時間が 10 %，20 %，および，30 %になる時間を設定した．

消費エネルギーは，式 (6) から，周波数 f_i における実行サイクル数 N_i の重み付き和 $\sum_i f_i^2 \times N_i$ として評価した．比較対象は，実行開始から終了まで最大周波数で実行させた Highest Speed と，実行開始時のみ周波数を切替え，プログラム内ではチェックポイントを置かずに周波数を切替えない Inter-Task DVFS の 2 種とした．提案手法による Intra-task DVFS では，チェックポイントの個数をチェックポイントリストに記録されているものの中から上位の 5 個，10 個，30 個，50 個，100 個，150 個の 6 種と，チェックポイント候補リストに記録されている全てのチェックポイント候補の数とした．チェックポイント候補リストには，cjpeg では 389 個，djpeg では 173 個のチェックポイント候補が記録されていたため，それらを用いた．消費エネルギーは，プログラムに対して，実行トレースマイニングで用いた入力データと同等のものを与えて計測した．

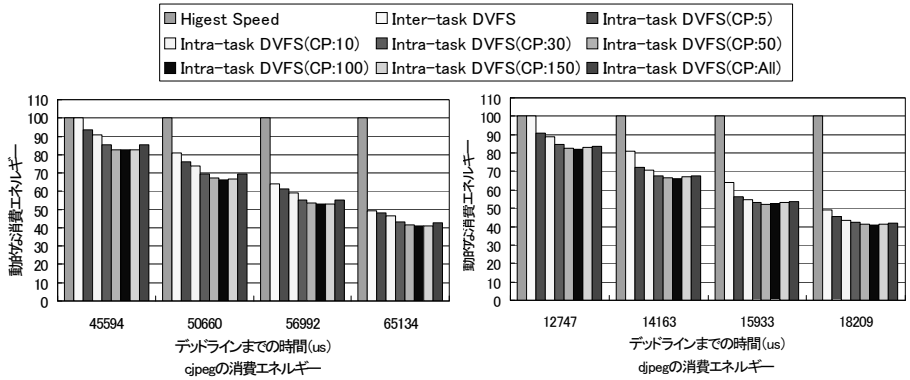


図 7 実験結果（消費エネルギー）
Fig. 7 Experimental results (energy consumption)

7.2 結果と考察

図 7 に実験結果を示す．グラフの縦軸は Highest Speed でプログラムを実行した際の消費エネルギーを基準に正規化を行った消費エネルギーである．

実験結果に対する考察を述べる．まず，図 7 から，全ての状況において Highest Speed と比較して，提案手法による DVFS では消費エネルギーが削減できていることがわかる．さらに，デッドラインまでの時間が長いほど消費エネルギーが多く削減できていることが確認できる．この傾向は，デッドラインまでの時間と同様にスラック時間も長くなり，周波数を大きく下げられるようになったためと考えられる．

Inter-task DVFS との比較について述べる．Inter-task DVFS を行った場合の消費エネルギーと比較すると，提案手法による DVFS では消費エネルギーが多く削減されている．これは，プログラムの途中に挿入されたチェックポイントによって，細かい粒度で周波数の切替えを行っているためであると考えられる．最大で，チェックポイントを 100 個に選定したときに 18.4 %もの消費エネルギーが削減できていることが分かった．

次に，チェックポイントの個数と消費エネルギーの関係について触れる．図 7 から，チェッ

表 2 jpeg エンコーダにおいてデッドラインミスした入力データ数
Table 2 The number of input data which causes deadline miss in the cjpeg

Deadline[us]	Intra-task DVFS						
	CP:5	CP:10	CP:30	CP:50	CP:100	CP:150	CP:389
45594	0	0	0	0	1	0	7
50660	0	0	0	0	0	1	11
56992	0	0	0	0	0	1	3
65134	0	0	0	0	0	1	8

表 3 jpeg デコーダにおいてデッドラインミスした入力データ数
Table 3 The number of input data which causes deadline miss in the djpeg

Deadline[us]	Intra-task DVFS						
	CP:5	CP:10	CP:30	CP:50	CP:100	CP:150	CP:173
12747	0	0	0	2	0	7	7
14163	0	0	1	0	0	4	8
15933	0	0	0	2	4	7	6
18209	0	0	0	0	2	6	7

クポイントの個数を 5 個から 100 個に増やすと同時に，消費エネルギーもより削減できていることが分かる．これは，チェックポイントを多く挿入することで，より細かい間隔で低い周波数に切替えられるためである．しかし，チェックポイントの数が 150 個の場合，および全てのチェックポイントを選んだ場合の消費エネルギーは，100 個の場合と比べて増加していることが確認できる．これは，チェックポイント関数によって実行サイクル数が増加したためと考えられる．そのため，消費エネルギーを最適化するためには，最適なチェックポイントの数を選定する意義があることがわかった．以上より，提案手法の有効性が確認できた．

最後に，チェックポイントの数とデッドラインミスを起こした入力データ数の関係について述べる．**表 2** および**表 3** を見ると，チェックポイントの数が増えると同時に，デッドラインミスを起こすデータ数が増えていることがわかる．これは，チェックポイントで次の周波数を計算する際に，今後通り得る全てのチェックポイント関数の実行サイクル数を考慮していないことが原因であると考えられる．この問題の対策については今後の課題とする．

8. おわりに

本研究では，シングルタスク環境下での DVFS によるプロセッサの消費エネルギー削減を目標とした，実行トレースマイニングによるチェックポイント抽出手法を提案した．消費エネルギー削減に効果の高いチェックポイントは，チェックポイントによるオーバヘッドの影響を考慮したうえで選定される．チェックポイント候補は，グリーディ法に基づいてランク付けされ，上位から一定個数のチェックポイントだけがプログラムに挿入される．同時に，何度も実行される同一の分岐命令に対しての対応も行った．設定できる周波数が離散的である想定のもとで，提案手法の評価実験を行った．評価実験では，チェックポイントの個数を適切な数にすることで，消費エネルギーを下げられることが確認できた．

本研究におけるチェックポイントの選定は，グリーディ法による手法を採用している．今後の方針として，より最適かつ効率の良いチェックポイント選定アルゴリズムを確立することが挙げられる．また，チェックポイントを挿入することによって発生するデッドラインミスへの対策が挙げられる．

謝辞 本研究にて，多くのご指導を頂きました菊地武彦氏，横山哲郎博士に深く感謝致します．本研究の一部は，科学技術振興事業団（JST）戦略的創造研究推進事業（CREST）「情報システムの超低消費電力化を目指した技術革新と統合化技術」の支援による．

参 考 文 献

1) Seo, J., Kim, T. and Chung, K.-S.: Profile-Based Optimal Intra-Task Voltage Scheduling for Hard Real-Time Applications, *Annual Conference on Design Automation*, ACM, pp.87–92 (2004).
2) 立松知紘，横山哲郎，菊地武彦，冨山宏之，高田広章：組込みシステムのタスク内 DVFS のための実行トレースマイニング，電子情報通信学会技術研究報告 [VLSI 設計技術]，Vol.108, No.478, 沖縄県，pp.11–16 (2009).
3) Ball, T. and Larus, J.R.: Efficient Path Profiling, *In Proceedings of the 29th Annual International Symposium on Microarchitecture*, pp.46–57 (1996).
4) Larus, J.R.: Whole Program Paths, *Proceedings of the SIGPLAN '99 Conference on Programming Languages Design and Implementation*, Atlanta, GA (1999).
5) Shin, D., Kim, J. and Lee, S.: Intra-Task Voltage Scheduling for Low-Energy, Hard Real-Time Applications, *IEEE Design & Test of Computers*, Vol.18, No.2, pp.20–30 (2001).
6) Shin, D. and Kim, J.: A Profile-Based Energy-Efficient Intra-Task Voltage Scheduling Algorithm For Real-Time Applications, *International Symposium on Low Power Electronics and Design*, ACM, pp.271–274 (2001).
7) Henkel, J. and Parameswaran, S.(eds.): *Designing Embedded Processors: A Low Power Perspective*, Springer (2007).
8) SimpleScalar LLC, http://www.simplescalar.com/.
9) ARM7TDMI ARM Processor, http://www.arm.com/products/CPUs/ARM920T.html.