A-06

高位合成における可変スケジューリングの近似アルゴリズム
Approximation Algorithm of Scheduling for High-Level Synthesis

曽根 康介 † 石浦 菜岐佐 ‡ 入谷 賢孝 † 戸田 勇希 †
Kousuke Sone Nagisa Ishiura Yoshitaka Iritani Yuki Toda

1 はじめに

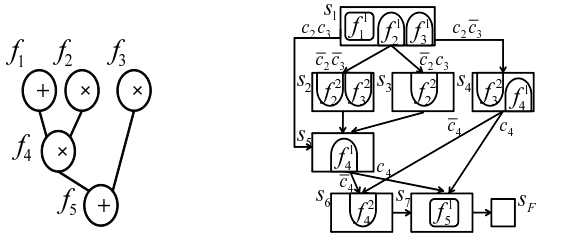
従来の高位合成におけるスケジューリング [1] では, 演算に要するサイクル数を固定として各演算の制御ステップを決定していた. そのため, メモリアクセス演算やシリアル乗算など実行サイクル数がデータに依存して変化する演算では, 無駄な待ちが生じることがある. 可変スケジューリング [2] は, この問題を解決しようとするものであり, 演算器の完了信号を基に各演算の実行タイミングを動的に変更することにより, 効率的なスケジューリングを可能とする. しかし, 可変スケジューリングでは平均サイクル数を短縮できる一方で, 制御のための状態数が増加するという問題点があった.

これに対し, 本稿では, 可変スケジューリングにおいて状態数の増加を抑制する近似アルゴリズムを提案する. この手法は, 各状態から終了状態までの平均サイクル数の短縮効果が小さい場合分けをなくすことによって状態数の増加を抑える. 実験の結果, 従来の可変スケジューリングと比較して, 平均サイクル数は約 3% 増加するが状態数は最大で約 81% 削減することができた.

2 可変スケジューリング

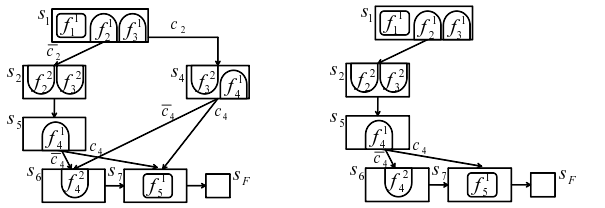
本稿では, 資源制約スケジューリングを対象とし, 演算の実行サイクル数はリストで与えられるものとする. 例えば, [2, 3, 4] は演算に 2, 3, または 4 サイクル要することを表す. 演算 f_j を実行する演算器からその演算の完了信号 c_j が得られるものとする.

図 1 (a) の データフローグラフ (DFG) において, f_2, f_3, f_4 は実行サイクル数が [1, 2] の不定サイクル演算であり, f_1, f_5 はサイクル数 1 の固定サイクル演算である. この DFG に対する可変スケジューリングの結果は図 1 (b) のような状態遷移グラフの形で表現する. 各状態は従来法のスケジューリングの 1 サイクルに相当する. f_j^t は演算 f_j の t サイクル目の実行を表す. 枝のラベルは遷移の条件を表す. 状態 s_1 において, f_2 が 1 サイクルで完了し f_3 が 1 サイクルで完了しない場合, $c_2\bar{c}_3$ の枝をたどって状態 s_4 に遷移する. この可変スケジューリングにより DFG の実行に必要な平均サイクル数は減少するが, DFG によっては状態数が大幅に増大してしまうことがある. これによって合成されるハードウェアのマルチプレクサや制御回路の規模が増加する恐れがあるため, 可能な限り総サイクル数の短縮効果を維持しつつ状態数を削減することが必要となる.



(a) データフローグラフ (b) 可変スケジューリング結果

図 1 可変スケジューリングの結果を表現する状態遷移グラフ



(a) 分岐削除後 ($\beta = 1.00$) (b) 分岐削除後 ($\beta = 0.70$)

図 2 平均サイクル数に注目した状態数削減

3 スケジューリングにおける状態数削減手法

本稿では, 平均サイクル数の削減への貢献が小さい分岐を削除することにより, できるだけ性能を低下させずに状態数を削減する手法を提案する. 状態数の増加を抑制する着眼点を以下に説明する.

まず, ある状態から分岐した 2 つの状態について, そこからの遷移先が全く同じであれば, 分岐を廃止することにより, サイクル数を維持しつつ状態数を削減することができる. 図 1 (b) において, 状態 s_1 から完了信号 c_3 によ

† 関西学院大学大学院 理工学研究科 情報科学専攻
‡ 関西学院大学 理工学部 情報科学科

表 1　スケジューリング・バインディング結果

プログラム	#op	#unit	max		min		可変 [2]		提案手法		
		(+,*,M)	#cy	#st	#cy	#st	#cy	#st	β	#cy	#st
conv_g.c	100	(1, 1, 1)	94.0	94	78.9	60	71.5	686	0.98	72.5	400
det3.c	64	(3, 3, 1)	68.0	68	47.5	31	39.7	621	0.90	40.7	442
matrix3_c.c	164	(3, 3, 1)	25.0	25	21.5	14	18.0	854	0.90	18.6	161

サイクル数: +[1], *[3, 4], M[1, 4]
確率: +[1], *[$\frac{1}{2}$, $\frac{1}{2}$], M[$\frac{4}{5}$, $\frac{1}{5}$]

り分岐する s_2, s_3 に着目する．両者の遷移先は s_5 で同じなので, この分岐による平均サイクル数の短縮効果は全くない．そこで, s_1 から s_3 への分岐を廃止する．その結果, 性能を低下させることなく図 2 (a) のように状態数を削減できる．

次にこれを一般化し, 遷移の分岐先の 2 状態を比較して, 両状態から終了状態までの平均サイクル数が等しければ, 分岐をせずに遷移先を統一する．これによって, 平均サイクル数を全く増加させることなく状態数を削減することができる．

さらにこれを拡張し, 遷移の分岐先の 2 状態を比較して, 平均サイクル数の短縮効果が小さければ分岐を行なわないようにする．これは, 分岐先の状態から終了までのサイクル数を計算し, その比があらかじめ決めた値 β ($0 \leq \beta \leq 1$) 以上なら分岐を廃止することにより実現する．例えば, 図 2 (a) の s_1 から s_2, s_4 への分岐に注目する．s_2, s_4 から終了までの平均サイクル数はそれぞれ 3.5, 2.5 サイクルである．$\beta = 0.7$ とすれば, 図 2 (b) のように s_4 への遷移を削除することができ, 全体の平均サイクル数は 4.0 サイクルから 4.5 サイクルに増加するが, 状態数はさらに削減することができる．

可変スケジューリングでは初期状態 s_1 から最終状態 s_F まで深さ優先で状態遷移グラフを構築する．分岐を持つ状態の遷移先の両状態の平均サイクル数を比較し, その比が β 以上であれば分岐を廃止する．これにより, 短縮効果が大きいときには分岐を残し, 速いサイクル数で全ての演算を完了できるパスを保持することが実現できる．

4　実験

提案手法に基づく高位合成のスケジューラを Cygwin 上に perl (5.8.7) で実装した．実験結果を表 1 に示す．ベンチマークは, 畳込み演算 (conv_g.c), 3 次正方行列の行列式の計算 (det3.c), 一方の行列を定数とする 3 次正方行列同士の乗算 (matrix3_c.c) である．#op は DFG 中の演算数, #unit は資源制約として与えた演算器数 (+ は加算器, *は乗算器, M はメモリアクセスユニット) である．加算, 乗算とメモリアクセスの実行サイクルはそれぞれ 1, [3, 4], [1, 4] とした．不定サイクル演算である乗算は等確率で実行サイクルが決まるものとし, メモリアクセスでは 1, 4 サイクル要する場合をそれぞれ $\frac{4}{5}$, $\frac{1}{5}$ の確率でとるものとした．残りの列は 4 つの手法について, 平均サイクル数 (#cy) とバインディング後の状態数 (#st) を比較したものである．「max」は各不定サイクル演算を最大サイクル数を要する固定サイクル演算としてリストスケジューリング法によりスケジューリングを行ったものである．逆に,「min」は各不定サイクル演算のサイクル数を最小に固定してスケジューリングを行い, そのサイクル数で演算が完了しない場合にはハードウェア全体をストールさせる方式のものである．「可変」は従来の可変スケジューリング [2] を行ったものである．「提案手法」は可変スケジューリングに本稿の近似アルゴリズムを適用したものである．

従来の可変スケジューリングと比較して, 近似アルゴリズムを適用した提案手法では, 平均サイクル数は約 3% 増加するが, 状態数は最大で約 81% 削減することができた．また, min のリストスケジューリング法と比べ, 状態数は約 7 倍に増加するが, 平均サイクル数は 8% 減少させることができた．

5　むすび

本稿では, 可変スケジューリングにおける状態数増加を抑制する近似アルゴリズムを提案した．今後の課題として, さらにスケジューリング段階における状態数削減手法を考案するとともに, 合成後の回路規模や遅延などの評価が挙げられる．

謝辞

本研究を進めるにあたり, 御助言・御討論を頂きました京都高度技術研究所の神原弘之氏, 名古屋大学の冨山宏之准教授, HLS プロジェクトの諸氏に感謝します．

参考文献

[1] Daniel D. Gajski, Nikil D. Dutt, Allen C-H Wu, and Steve Y-L Lin: *High-Level Synthesis: Introduction to Chip and System Design*, Kluwer Academic Publishers (1992).

[2] Yuki Toda, Nagisa Ishiura, and Kousuke Sone: “Static Scheduling of Dynamic Execution for High-Level Synthesis,” in *Proc. SASIMI 2009,* pp. 107–112 (Mar. 2009).