# 高位合成のバインディングの整数線形計画法による分割解法における実数制約の導入

Introducing Real Constraints in Partitioned ILP-Based Biding in High-Level Synthesis

大迫 裕樹 石浦 菜岐佐
Yuuki Oosako Nagisa Ishiura

関西学院大学 理工学部 School of Science and Technology, Kwansei Gakuin University

## 1 はじめに

バイディングは高位合成において合成される回路規模や遅延に最も大きな影響を及ぼす処理である. 整数線形計画法 (以下 ILP) を利用すれば質の高い解を得られるが, 回路が大きくなると現実的な時間内に解を求めることは困難になる. 部分問題毎に ILP を適用するのがこれを解決する一手法であるが, 大局的な視点が失われて解の品質が低下してしまう. そこで本稿では, 部分問題を解く際に, 他の部分問題の制約条件を実数制約として追加することによって解の質の向上を図る手法を提案する.

## 2 整数線形計画法 (ILP) によるバインディング

本稿では, スケジューリング済みの CDFG (control dataflow graph) の演算と値に対し, 演算器とマルチプレクサの総コストが最小になるように, 演算器とレジスタを割り当てる問題を扱う. この時, [1] と同様に, チェイニングに起因するフォルスパスを含めた全てのパスが 1 クロックの遅延を越えないようにする. そのようなバインディングが存在しない場合, スケジューリングを変更するのではなく, [2] のように演算器数を変更する.

## 3 提案手法

本稿では, 制御ステップに基づいてバインディング問題を分割して ILP で解く. この際, あるステップ集合のバインディングを求める際に, 別のステップ集合に関する制約条件を実数制約で導入する手法を提案する.

CDFG 中のノード (演算および値) の集合を $N$, データバス中のユニット (演算器およびレジスタ) の集合を $U$ とする. $n \in N$ が割り当てられたステップを $\sigma(n)$ とし, ステップ集合 $S$ に対し $N_S = \{n \in N \mid \sigma(n) \in S\}$ とする. 変数 $x_{n,u}$ は $n \in N$ を $u \in U$ に割り当てるときそのときに限り 1 となる 0-1 変数とし, 演算器とマルチプレクサのコストの和を最小化する ILP を定式化する.

本手法では, バインディングを行う全ステップの集合 $S$ を適切な大きさの部分集合に分割し, その部分集合ごとにバインディングを決定していく. まだバインディングを求めていないステップの集合を $S_U$ とする. 次に $S_I \subseteq S_U$ のバインディングを求める際には, $S_R \subseteq (S_U - S_I)$ を選び, $x_{n,u}$ 以外のすべての補助変数, および $n \in N_{S_R}$ である $x_{n,u}$ を実数変数として, 混合整数計画問題を解く. これを $S_U = \phi$ となるまで繰り返すことにより, 全体のバインディングを求める.

## 4 実験結果

提案手法に基づくバインディング処理系を Perl5 で実装した. ILP のソルバーには CPLEX 12.6.1.0 を使用し, モデルと解の受け渡しはファイルを介して行った. $S_I$ と $S_R$ は CDFG ファイル中の出現順で選択した.

実験結果を表 1 に示す. “cost” は ILP の目的関数で, ALU, 乗算器, マルチプレクサ, レジスタのコストをそれぞれ, 32, 128, 32, 32 としたものである. $|S_I| = 1, |S_R| = 0$ とした実行例は二部グラフマッチングに基づく手法 [3] と等価である. $|S_I| = k, |S_R| = 0$ は $k$ ステップ毎にバインディングを行った場合である. “RSA” では, ノード数に基づいてステップの分割を行っており, $s(n)$ は次の $n$ 個のノードを含む最小のステップ数を表す. 複数ステップ毎に解いた場合, 1 ステップ毎に解くよりもコストが抑えられている. しかし, $|S_R| > 0$ とした本手法では, より短時間で更に良い解が得られた. $S_R = |S_U|$ は未バインディングの全ステップを $S_R$ とすることを表す. ただし, この結果はモデルに強く依存しており, 条件次第では, $S_R = \phi$ とした従来手法の方が, 本手法よりも良い解が得られる場合もある.

表 1 実験結果

| | $\|N\|$ | $\|S\|$ | $\|S_I\|$ | $\|S_R\|$ | cost | CPU[s] | |
|---|---|---|---|---|---|---|---|
| ellip | 75 | 17 | 1 | 0 | 1,456 | 1.07 | * |
| | | | 5 | 0 | 1,296 | 357.99 | * |
| | | | **1** | $\|S_U\|$ | **1,264** | **20.68** | * |
| s2m | 81 | 14 | 1 | 0 | 1,264 | 0.59 | * |
| | | | 6 | 0 | 1,168 | 1.89 | * |
| | | | 7 | 0 | 1,136 | 5.98 | * |
| | | | **1** | $\|S_U\|$ | **1,136** | **3.27** | * |
| RSA | 5,417 | 854 | 1 | 0 | 12,032 | 2950.61 | ** |
| | | | s(30) | 0 | 9,728 | 206.05 | ** |
| | | | **s(30)** | **s(870)** | **8,992** | **8271.24** | ** |
| | | | s(350) | 0 | 9,408 | 8913.13 | ** |
| | | | s(400) | 0 | 9,248 | 7555.20 | ** |

$N$: ノード全体の集合, $S$: ステップ全体の集合
$S_I$: ILP で解くステップの集合, $S_R$: LP で解くステップの集合
cost: ALU=32, 乗算器=128, MUX=32(入力数 − 1), レジスタ=32
$|S_U|$: 未バインディングのステップ全体
$s(n)$: 次の n 個のノードを含む最小のステップ
* 1.7GHz Core i7 (RAM 8GB), ** 3.4GHz Core i7 (RAM 16GB)

## 5 むすび

本稿では, 高位合成における整数線形計画法を利用した複数ステップ単位のバインディングに実数制約を導入する手法を提案した. $S_I$ および $S_R$ の大きさや, ステップの処理順序に関する改良を行うことが今後の課題である.

### 参考文献

[1] A. Kondratyev, L. Lavagno, M. Meyer, Y. Watanabe: “Share with care: A quantitative evaluation of sharing approaches in high-level synthesis,” in *Proc. DATE 2013*, pp. 1547–1552 (Mar. 2013).

[2] Y. Hara-Azumi and H. Tomiyama: “Clock-constrained simultaneous allocation and binding for multiplexer optimization in high-level synthesis,” in *Proc. ASP-DAC 2012*, pp. 251–256 (Jan.–Feb. 2012).

[3] C.-Y. Huang, Y.-S. Chen, Y.-L. Lin, and Y.-C. Hsu: “Data path allocation based on bipartite weighted matching,” in *Proc. 27th DAC*, pp. 499-504 (June 1990).