計算機アーキテクチャ 115－1
設 計 自 動 化 78－1
（1995. 12. 14）

接続確率に基づくアロケーション手法

吉田 美紀　　平川 裕介　　原嶋 勝美　　福永 邦雄

大阪府立大学 工学部 情報工学科
〒593 堺市学園町 1-1
TEL:0722-52-1161
E-mail: miki@com.cs.osakafu-u.ac.jp

ハイレベルシンセシスにおけるアロケーション問題について，割り当て確率に基づく解法を提案する．アロケーション問題では，演算器，レジスタ，接続素子の 3 つを同時に考慮しながら割り当てる必要があるが，クリーク分割や ILP を用いた手法では，計算時間が大きくなるため規模の大きい問題には適さない．本手法では，問題を 3 つに分割し，それぞれの部分問題において，接続コストを最小化し，効率良く最適に近い解を求めている．ただし，割り当てが一意に決定しない組合せに対しては，接続確率を用いて接続コストを見積もっている．コストの見積もりを定式化することにより，DFG の形に左右されず，より少ない計算時間で最適に近い解が得られる．

A Probability-based Approach for Data Path Allocation

Miki Yoshida　　Yusuke Hirakawa　　Katsumi Harashima　　Kunio Fukunaga

Faculty of Engneering, University of Osaka Prefecture
1-1 Gakuen-tyo, Sakai, Osaka, 593, Japan

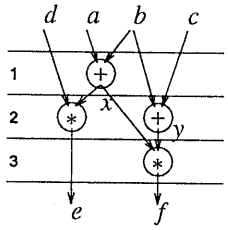
We propose a probability-based approach for data path allocation. Previous approaches can not obtain optimal solutions, because they take large calculation time. While, the proposed approach can obtain solutions close to the optimal by using the probability of assigning operations and variables to hardware units. Moreover, our approach devides an allocation problem into three subtasks to reduce calculation time, and solves subtasks, respectively. At each phase, we estimate costs of interconnections between hardware by using probability, and minimize the amount of costs. Consequently, our approach can solve allocation problem independent from the input DFG form.

1. まえがき

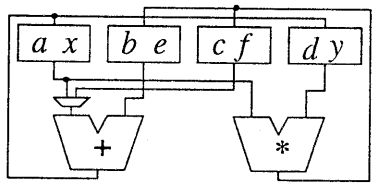
近年, VLSI の大規模化, 多様化, 高集積化に伴い, 従来の人手による設計が困難となり, 設計の自動化が進んできている. ハイレベルシンセシスでは, 主にスケジューリングとアロケーションが行なわれる. アロケーションとはスケジューリングの結果を受けて, 各演算, データ転送に必要な演算器, バス, レジスタ等のハードウェア資源の割り当てを行ない, レジスタ転送レベルの回路を生成する技術である. アロケーションの結果は, 最終的なチップ面積に大きく影響するため, できるだけハードウェア資源が少なくなるように, 演算の演算器割り当て, 変数のレジスタ割り当て, データの転送路等の割り当てを行う必要がある. アロケーションでは, これらの要素のトレードオフを考慮し, できるだけ少ないハードウェアで与えられた動作仕様を実現しなければならない.

従来, アロケーション問題に関しては, グラフ理論における 2 部マッチングを用いた解法 [1] や, クリーク分割を用いた解法 [2], 整数線形計画法を用いた解法 [3] 等, さまざまな解法が提案されている [4]-[8]. しかし, 局所的な最適解しか得られなかったり, 計算量が大きくなるため, 規模の大きい問題には向いていないものが多い.

本報告では, 少ない計算時間で, より最適に近い解を得ることを目的とするアロケーション手法を提案する. 提案手法では, 計算時間の縮小のために, アロケーション問題を 3 つの部分問題に分割し, 個別に解を求めている. ただし, それぞれの部分問題においては, 割り当て済みの情報を利用するとともに, 未割り当ての要素に対する割り当て確率を用いて接続コストを評価することで, 最終的にレジスタ, 演算器, 接続線ができるだけ共有される割り当てが決定できる. 接続コストの評価には割り当て確率を用いるが, 割り当てが決定していない要素について, 割り当て可能な組合せに対しては均等な確率で割り当てられるとし, 既に決定している割り当てに対しては確率を 1 とする. 本手法では, 可能性のある接続に対するコストを全て算出し, 最小コストの組合せを選出することによって, 最適に近い解を得ることを可能にしている.



(a) データフローグラフ (DFG)



(b) データパスモデル

図 1 アロケーション問題

2. 確率による割り当てアルゴリズム

入力として, 図 1(a) のようなスケジューリングされたデータフローグラフ (DFG) を用いる. アロケーション結果は, 図 1(b) のようなデータパスモデルとし, レジスタ及び演算器の複数入力に対してはマルチプレクサを用いる.

2.1. 概略

アロケーション問題では, DFG の構成要素である演算, 変数, データ転送の, 演算器, レジスタ, 接続線への最適な割り当てを同時に決定することが望ましいが, 計算時間が大きくなるため困難である. そこで本手法では, まずアロケーション問題を 3 つの部分問題に分割し, 演算の演算器割り当て, 変数のレジスタ割り当て, データ転送路の割り当ての順で, 解を求める. 各部分問題において, 確率を用いて接続コストを見積もり, ハードウェア資源が最小になるように割り当てを逐次決定していく.

2.1.1. 割り当て確率

アロケーション問題は, DFG 上の要素 (演算, 変数, 枝) と, ハードウェアユニット (演算器, レジスタ, 接続線) の最適な組合せを決定する組合せ問題と考えられる. 本手法では, 組合せの候補が複数存在する場合に, 最適な組合せを選択するため, 割り当て確率を用いる. 各要素は組合せ可能なハードウェアユニット中に, 割り当て確率だけ存在するものとし, その確率によって接続コストを見積もることによって, 最適な要素とハードウェアの組合せを選択していく. ここでは, 複数の候補ユニットに対する割り当て確率は全て等しいものとし, 確率の値を次のように定義する.

- 割り当てがおこらない要素–ユニットの組合せ対して, 割り当て確率の値を 0 とする.
- 割り当てが決定している要素–ユニットの組合せに対して, 割り当て確率の値を 1 とする.
- 割り当てが決まっていない要素–ユニットの組合せに対して, 割り当て確率の値を, (1/その要素の割り当て可能なユニット数) とする.

2.1.2. 確率による接続コストの見積もり

割り当て確率を用いて, 接続コストを評価するために, 前節の割り当て確率に基づいてレジスタ-演算器間に仮接続を行なう. 仮接続の手順を以下に述べる.

データ転送が必要な変数と演算に対し, それぞれが割り当て可能なレジスタ, 演算器間に接続線を付加する. このとき, 接続線は (変数のレジスタへの割り当て確率) と, (演算の演算器への割り当て確率) の積を重みとして持つものとする. 同一レジスタ–演算器間に複数の接続線ができる場合には, 重みが最大になるものをそのレジスタ–演算器間の接続線とし, 同一レジスタ–演算器の接続線は 1 本とする. また, レジスタおよび演算器の入力側に複数の接続線が接続される場合, 接続線を選択するために, マルチプレクサを付加する. 各マルチプレクサは, 接続している接続線の重みの和を, そのマルチプレクサの重みとして持つものとする. 以上より, 接続コスト (C) は,

$$C = \sum(\text{接続の重み}) + \sum(\text{マルチプレクサの重み}) \quad (1)$$

と定義できる. ただし,

(マルチプレクサの重み)
$$= \alpha \times \sum(\text{入力の接続の重み})$$

とし, αは係数パラメータとする.

2.1.3. 割り当てアルゴリズム

前節の割り当て確率と接続コストを用いて以下のアルゴリズムによって割り当てを決定していく.

(1) 初期条件と割り当て確率により, 要素–ユニット間の割り当て確率表を作成する.

(2) 割り当ての決定していない要素について, 割り当て可能なユニットのうちの 1 つの確率を 1, 他を 0 とし, 他の要素の確率を修正する. 修正した確率表に従い, 式 (1) を用いて接続コストを計算する. これを割り当て可能な全ユニットについて行なう.

(3) ユニットの決定していない要素全てについて (2) の操作を行い, 接続コストが最小になる要素–ユニットの組を 1 つ選び, その割り当て確率を 1 とする.

(4) (3) の結果に基づいて, 割り当て確率表を更新し, 全ての要素–ユニット間の割り当てが決定するまで (2),(3),(4) の操作を繰り返す.

2.2. アルゴリズムの適用

本節では, アロケーション問題の各部分問題に対する本手法の適用手順について述べる.

2.2.1. 演算の演算器割り当て

接続コストの小さいアロケーション結果を得るためには, 割り当て済みの要素の情報を利用することが重要となってくるが, 演算器割り当ては最初に行なわれるために, この段階では, まだ変数–レジスタ間の割り当て情報を用いることができない. そこで, 本手法では, 少ない計算時間で, レジスタ数最小の割り当てが得られる left-edge アルゴリズム [2] により, 変数–レジスタ間の割り当てを

仮に決定する. left-edge アルゴリズムによる変数–レジスタ割り当ての結果をもとに, 2.1.3 のアルゴリズムを適用し, 演算–演算器間の割り当てを行なう. ここで left-edge アルゴリズムにより仮に決定した変数–レジスタ割り当ては, 演算器の割り当てが決定した後, 確率を用いて再度割り当てを行なう.

2.2.2. 変数のレジスタ割り当て

演算器割り当ての結果を用いて, 変数–レジスタの再割り当てを行なう. ここでは, 要素, ユニットをそれぞれ変数, レジスタとし, 2.1 の割り当て確率とアルゴリズムを適用する.

アルゴリズムの (1) では, 初期条件として, 変数の並列度が最大のステップを 1 つ与え, そのステップの変数を各レジスタに 1 つずつ割り当てる. 一意に割り当てが決定する変数をレジスタに固定することにより, 計算時間を縮小することができる. レジスタの決定した変数のライフタイムと重なりのある変数において, その変数のレジスタへの割り当て確率を 0 とし, 変数とレジスタの割り当て確率表を作成する.

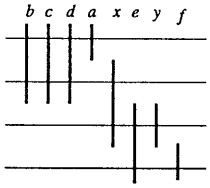


図 2 ライフタイムテーブル

入力 DFG を図 1(a) とすると, 変数に対して, 図 2 のライフタイムテーブルが得られる. 同一クロックを横切る変数はライフタイムが重なるので, 同一レジスタを共有することはできない. そのため, 図 2 の場合は変数 b, c, d, a および b, c, d, x は同一レジスタに割り当てることができないことがわかる. 初期条件により, 第 1, 2 クロックの変数がそれぞれレジスタに割り当てられると, 変数 x とのライフタイムの重なりから, 変数 e, yは, 変数 x と同じレジスタには割り当てられないので, 変数 x が割り当てられたレジスタへの割り当て確率は 0 となる. この結果, 初期の割り当て確率表は表 1 のようになる.

表 1 変数–レジスタ割り当て確率表

	レジスタ			
	1	2	3	4
b	1	0	0	0
c	0	1	0	0
d	0	0	1	0
a	0	0	0	1
x	0	0	0	1
e	$\frac{1}{3}(1)$	$\frac{1}{3}(0)$	$\frac{1}{3}(0)$	0
y	$\frac{1}{3}(0)$	$\frac{1}{3}(\frac{1}{2})$	$\frac{1}{3}(\frac{1}{2})$	0
f	$\frac{1}{4}(0)$	$\frac{1}{4}(\frac{1}{3})$	$\frac{1}{4}(\frac{1}{3})$	$\frac{1}{4}(\frac{1}{3})$

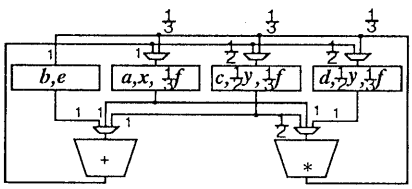


図 3 接続コストの見積もり

(2) における確率表の修正を, 変数 e の レジスタ 1 への割り当て確率を 1 とした場合を用いて説明する.

変数 e のレジスタ 1 への割り当て確率を 1 とすると, ライフタイムの重なりから, 他の変数–レジスタ間の割り当て確率が変化し, 表1の括弧内の確率に修正される. この表に従って仮接続を行なうと図 3 のように接続コストを見積もることができる.

接続線のコスト ($C_{connect}$) は, レジスタの出力–演算器の入力間では, $1+1+1+\frac{1}{2}+1+1=5\frac{1}{2}$, 演算器の出力–レジスタの入力間では, $\frac{1}{2}+\frac{1}{2}+1+\frac{1}{3}+\frac{1}{3}+\frac{1}{3}+1=4$ であるので, $C_{connect}=5\frac{1}{2}+$

$4 = 9\frac{1}{2}$ となる．また，マルチプレクサのコスト (C_{mux}) に関しても同様に，レジスタの入力側について，$1 + \frac{1}{3} + \frac{1}{2} + \frac{1}{3} + \frac{1}{2} + \frac{1}{3} = 3$，演算器の入力側について，$1 + 1 + 1 + \frac{1}{2} + 1 + 1 = 5\frac{1}{2}$ となり，$C_{mux} = 3 + 5\frac{1}{2} = 8\frac{1}{2}$ と計算される．従って，$C = C_{connect} + C_{mux} = 9\frac{1}{2} + 8\frac{1}{2}\alpha$ と，接続コストを見積もることができる．

(3) では，上記のようにして得られる C の値が最小になる変数-レジスタの組合せを選択する．

2.2.3. 接続端子の決定

演算器の出力からレジスタの入力への接続は一意に決定するが，演算器の入力端子は複数ある場合がある．どちらの端子に接続するかにより，演算器の入力側にマルチプレクサが必要となってくることがあるため，できるだけマルチプレクサを必要としないように接続を決定しなければならない．ここでは，要素，ユニットはそれぞれ DFG 上の枝，レジスタ-演算器間の接続線とし，2.1 のアルゴリズムを適用することで，マルチプレクサを最小化する接続端子の決定をしている．

3. 実験結果

数種の DFG に対し，本手法を適用した結果を表 2に示す．表中の+,*は，各 DFG のスケジューリング結果における加算，乗算器の個数で，♯Reg, ♯Mux, ♯Connection はそれぞれ必要なレジスタ数，マルチプレクサ数，及び接続線の本数である．

表 2　実験結果

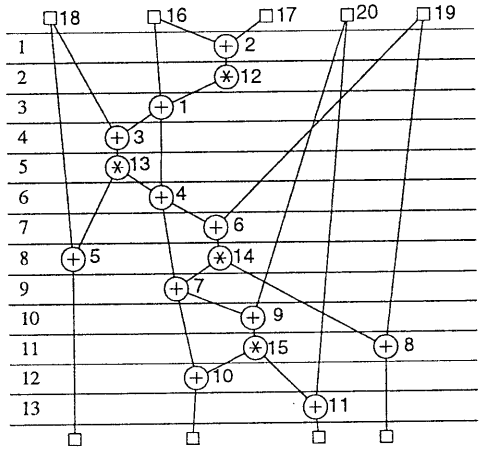
DFG	+	*	♯Reg	♯Mux	♯Connection
apl(1)	1	1	6	2	12
apl(2)	2	1	6	2	12
jmn	1	1	7	3	18
fewf	2	1	10	9	37

apl: +12,*4

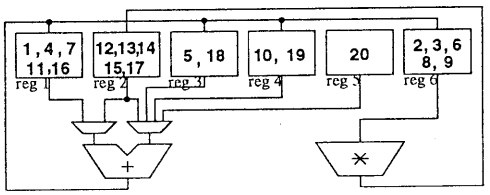
jmn: +13, *4

fewf: +26, *8

図 4(a) は，apl(1) のスケジューリング済みの DFG を示し，図4(b) は，そのアロケーション結果を示し，レジスタ中の数字は変数を表し，変数の番号はその変数が生成される演算の番号と一致する．図 4(b) の変数-レジスタ割り当てでは，レジスタ 2 に，乗算から出力される変数が集められている．このことからもわかるように，DFG 上で，同じパターンの変数が同じレジスタに集められる傾向がある．



(a) データフローグラフ



(b) アロケーション結果

図 4　実験結果 (apl)

apl, jmn に対しては，最適解が得られている．fewf に関しては，問題の規模が大きいため，最適解を得るのは困難であるが，最適に近い解が得られている．比較のため，総当たりに近い形で解を求め，apl, jmn に関しては最適であることがわかった．fewf は，組合わせの数が多く，計算時間が大きくなるため，最適解は得られなかったが，得られた解のうち最適に近いと考えられるものとほぼ同等

の結果となっている．以上の結果から，本手法では，接続線が共有され，接続コストが増加しないように割り当てが決定されていることがわかる．また，計算時間に関しても，数秒以内のCPU時間で解を得ることができた．

4. まとめ

確率を用いて接続コストを見積り，割り当てを決定していく手法を提案した．実験により，本手法によって，最適解，または準最適解の得られることを確認した．今後の課題として，計算時間の縮小と，接続コストのより厳密な計算法を検討中である．

参考文献

[1] Chu-Yi Huang, Yen-Shen Chen, Youn-Long Lin, Yu-Chin Hsu: "Data Path Allocation Based on Bipartite Weighted Matching," 27th ACM/IEEE DAC, pp499–504,(1990).

[2] D.Gajski, A.Wu, N.Dutt,and S.Lin: "HIGH-LEVEL SYNTHESIS Introduction to Chip and System Design," Kluwer Academic Publishers,(1992).

[3] Minjoong Rim, Ashutosh Mujumdar, Rajiv Jain, and Renato De Leone: "Optimal and Heuristic Algorithms for Solving the Binding Problem," IEEE Tans. VLSI SYSTEMS, pp211–225,(1994).

[4] Nam-Sung Woo: "A Global, Dynamic Register Allocation and Binding for a Data Path Synthesis System," Proc. of 27th ACM/IEEE DAC pp505–510,(1990).

[5] P.G.Paulin and J.P.Knight:"Force-Directed Scheduling for the Behavioral Synthesis of ASIC's," IEEE Tans. Computer-Aided Design, **CAD-8**, pp.661–679,(1989).

[6] 高橋 隆一, 吉村 猛: "ハイレベルシンセシスの動向", 信学論 (A),**J74-A**, 2, pp.143–151,(1991).

[7] 若林一敏: "VLSI 設計における最適化算法", システム/制御/情報 Vol.37, No.4,pp.214–222,(1993).

[8] 今井正治 編著: "ASIC 技術の基礎と応用", 電子情報通信学会.

[9] S.Y.Kung, H.J.Whitehouse and T.Kailath: "VLSI and Modern Signal Processing," Englewood Cliffs,NJ:Pretice Hall,(1985).