社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－SLDM－133（15）
2008 / 1 / 17

レジスタ分散型アーキテクチャを対象とした高位合成のためのマルチプレクサ削減手法

遠藤 哲弥† 大智 輝† 戸川 望† 柳澤 政生† 大附 辰夫†

† 早稲田大学大学院基幹理工学研究科 情報理工学専攻
〒 169-8555 東京都新宿区大久保 3-4-1
Tel: 03-3209-3211(5775), Fax: 03-3208-7439
E-mail: †endou@togawa.cs.waseda.ac.jp

あらまし 近年のLSI設計プロセスの微細化に伴い，配線遅延がゲート遅延に対し相対的に増加してきている．また単位面積あたりの総ゲート数，総配線数が増加し，配線制御に必要なマルチプレクサ数が増大してきている．レジスタ分散型アーキテクチャを用いると，レジスタ間データ転送を利用することにより配線遅延が回路の性能に与える影響を低減できるが，レジスタ間接続に要する総配線数の増加に伴い，必要となるマルチプレクサ数の増大を招いてしまう．本稿では，レジスタ分散型アーキテクチャを対象とした高位合成システムにおけるマルチプレクサ削減手法を提案する．提案手法は各演算器，ローカルレジスタ間の配線接続に対し，ポート割当を最適化することで必要なマルチプレクサ数を削減する．計算機実験によって，対象とする高位合成手法に提案手法を組み込んだ場合，平均で10.9%のマルチプレクサ数，4.9%の面積が削減でき有効性を確認した．
キーワード マルチプレクサ，高位合成，レジスタ分散型アーキテクチャ，ポート割当，配線遅延，配線数

A Multiplexer Reduction Algorithm in High-level Synthesis for Distributed-Register Architectures

Tetsuya ENDO†, Akira OHCHI†, Nozomu TOGAWA†, Masao YANAGISAWA†, and Tatsuo OHTSUKI†

† Dept. of Computer Science and Engineering, Waseda University
3-4-1 Okubo, Shinjuku, Tokyo 169-8555, Japan
Tel: +81-3-3209-3211(5775), Fax: +81-3-3208-7439
E-mail: †endou@togawa.cs.waseda.ac.jp

Abstract As device feature size decreases, interconnection delay becomes the dominating factor of total delay. In addition, as the number of total gates and the number of wirings in each unit area increase, the number of multiplexers that is necessary for the wiring control increases. By using a distributed-register architecture, we can synthesize circuits with register-to-register data transfer, and can reduce influence of interconnection delay. However, as the number of wirings required for the connection between registers increases, the needed number of multiplexers is also increased. In this paper, we propose a multiplexer reduction algorithm in high-level synthesis for distributed-register architectures. This algorithm can reduce the number of multiplexers for each functional unit, wiring connection between local registers by optimizing a port re-assignment. We show effectiveness of the proposed algorithm thorough experimental results.
Key words multiplexer, high-level synthesis, distributed-register architecture, port assignment, interconnect delay, the number of wirings

1. まえがき

大規模複雑化するLSI設計の生産性を向上させるため，抽象度の高い動作レベル記述による回路設計を可能とする高位合成を利用することは有効な手段である．従来の高位合成手法ではフロアプラニングを高位合成の後処理として扱っていたため，モジュール（演算器，レジスタ，コントローラ，MUX等）間の配置関係や配線遅延情報を，高位合成の段階で考慮することはできなかった．近年のLSI設計プロセスの微細化に伴い，ゲート遅延に対して配線遅延が相対的に増加しており，今後もこの傾向は継続すると予想される．そこで，高位合成の段階においても，モジュール配置，配線遅延を考慮することが必要となる．一方で，チップ内部の微細化が進行し，単位面積あたりのゲート数，総配線数が増加することに伴い，配線制御に必要なマルチプレクサ数が増加してきている．そこで，マルチプレクサ数を考慮した設計が必要となる．

従来のフロアプランを考慮した高位合成の研究[2],[10]では，モジュール配置を工夫することで，クリティカルパスに含まれる配線遅延の割合を削減してクロック周期を短縮している．これらの手法は，演算器がレジスタを共有するアーキテクチャを採用している．このモデルでは，演算器とレジスタとの間に長い配線を引く必要が生じる場合がある．これにより配線遅延が増大し，回路の性能低下を招いてしまう．

配線遅延がボトルネックとなる状況下を想定して，[3],[4]ではレジスタ分散型アーキテクチャが提案された．このアーキテクチャでは，演算器毎に専用のローカルレジスタを配置する．各演算器は隣接した位置に配置されたローカルレジスタとデータをやり取りするため，配線遅延の影響は小さくなり，クロック周期をほぼ演算器の遅延で占められるようになる．また，離れて配置された演算器間のデータ転送にはレジスタ間データ転送を利用できるという特徴を持つ．この特徴により，1クロックあたりの演算器の利用効率が上がる．しかし，このモデルでは全ての演算器に専用のローカルレジスタを付加するため，従来のレジスタ共有型アーキテクチャよりもレジスタ数が増加してしまう．レジスタ数の増加に伴い，レジスタ間接続に要する総配線数が増加することで，配線制御に必要なマルチプレクサ数が増大するという問題点も生じてしまう．マルチプレクサ数が増加することで回路面積が増加するため，マルチプレクサ数を考慮した設計も高位合成には必要である．

マルチプレクサ削減に関連した既存研究として，[1],[6],[7],[11]がある．[6]は遺伝的アルゴリズムを用いてスケジューリングとアロケーションを同時に行うことで，各ノードに対するモジュール選択，モジュール間接続における最適解を導出する手法を提案している．[11]では整数線形計画法を用いたデータパスアロケーション手法を提案している．しかしながら，これらの手法は冗長な配線を削減しているにも関わらず，マルチプレクサ数削減に関する言及が行われていないため，本稿で提案しているマルチプレクサ削減手法とは目的が異なる．また，[1],[7]では，スケジューリング済みDFGを入力とし演算器，レジスタ間の配線接続に対しポート割当の変更を行うことで配線数を削減する手法を提案し，レジスタ数とマルチプレクサ数に関する評価を行っている．しかしながら，これらの手法はレジスタ共有型アーキテクチャを対象としているため，レジスタ分散型アーキテクチャには対応していない．

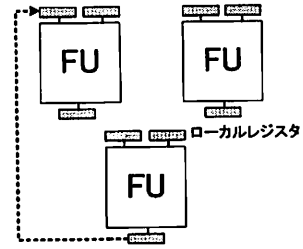

図1 レジスタ分散型アーキテクチャ．

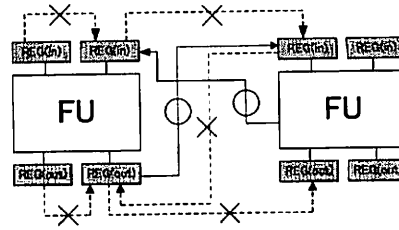

図2 許可するデータ転送（出力側ローカルレジスタは2個とした）

本稿では，レジスタ分散型アーキテクチャを対象とした高位合成手法[8]をベースとし，新たにマルチプレクサ削減処理を組み込んだ手法を提案する．提案手法はモジュール（各演算器，ローカルレジスタ）間における配線接続に対しポート割当最適化を行う．ポート割当変更情報をフィードバックし，再度レジスタバインディングを行うフローを加えることで，冗長な配線を削減し，必要なマルチプレクサ数を削減することができる．

本稿は以下のように構成される．2章では対象とするレジスタ分散型アーキテクチャの概要について説明する．3章では対象とする高位合成手法[8]の概要，合成フローについて説明する．4章ではマルチプレクサ削減手法を提案し，提案手法を加えた合成フローについて説明する．5章では提案手法を加えた高位合成手法に対し計算機実験を行った結果を報告する．

2. レジスタ分散型アーキテクチャ

図1にターゲットアーキテクチャであるレジスタ分散型アーキテクチャを示す[3]．レジスタ分散型アーキテクチャでは，配線遅延がボトルネックとなる状況下を想定して，各演算器が入出力に専用のローカルレジスタを有する構成となっている．いずれの演算器に対しても隣接した位置に専用のレジスタが配置されるため，演算器とレジスタ間の配線遅延が小さくなる．離れたローカルレジスタとデータをやり取りするため，レジスタ共有型のモデルと比較して配線遅延の影響は小さくなり，クロック周期をほぼ演算器の遅延のみで占めることが可能となるため回路性能が向上する．また，離れて配置された演算器間のデータ転送にはレジスタ間データ転送を利用できるという特徴を持ち，レジスタ間データ転送による複数のサイクルをまたぐ配線遅延を扱うこともできる．

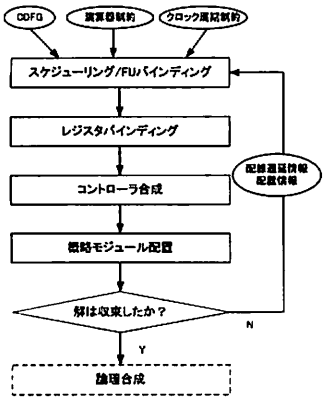

図 3 合成フロー

レジスタ分散型アーキテクチャでは，各演算器に配置する入力側ローカルレジスタ数を 2 個 (固定)，出力側ローカルレジスタ数を可変とし，

- 出力側ローカルレジスタ⇒入力側ローカルレジスタ
- 演算器の出力 (ダイレクト) ⇒入力側ローカルレジスタ

のデータ転送のみ考えることにする (図 2).

3. レジスタ分散型アーキテクチャを対象とする高位合成 [8]

提案手法は高位合成手法 [8] をベースとしている. [8] では分岐処理，クロック周期制約に対応しており，レジスタ分散型をターゲットアーキテクチャ対象とする高位合成手法を確立している．本章では [8] の概要，合成フローを説明する. [8] における高位合成問題は，入力を CDFG とし，制約条件に演算器制約，クロック周期制約を与え，RT レベルのデータパスと制御回路およびモジュール配置情報を出力する問題である．目的関数は入力アプリケーションの実行時間の最小化である．また実行時間が同一の場合は面積を最小化する．CDFG は，有向グラフ $G=(V,E)$ で表現され，ノード集合 V は，演算ノード集合 $V_N=\{n_i \mid i=1,2,\ldots,l\}$ と，分岐制御を表すフォークノードの集合 V_C を含むものとする．また，利用可能な演算器を表す演算器制約のリストとして，$F=\{f_i \mid i=1,2,\ldots,m\}$ が存在するものとする．各演算器は，面積及び演算に要する遅延に関する情報と，何クロックで演算を実行するかというパラメータを持つ．クロック周期制約とは，任意の演算器 f の遅延が T_f で n クロックで演算を処理するときに，f がレジスタから入力を読み込み，出力をレジスタに格納するまでの時間 T_c が，クロック周期制約 T_{clk} に対して $T_c/n \leq T_{clk}$ の条件を満たさなければならないという制約である．

[8] ではレジスタ間データ転送を扱うため，スケジューリング段階で各演算器間の配線遅延情報が必要となる．そのため，図 3 に示すような配置情報，配線遅延情報をフィードバックする合成フローを用いている．

3.1 スケジューリング/FU バインディング

スケジューリング/FU バインディングの工程では，モジュール配置工程からフィードバックされた配線遅延を考慮し，レジスタ間データ転送を利用する CVLS [9] ベースのスケジューリングと FU バインディングを同時に実行する．目的関数はコントロールステップの最小化である．CVLS は，CV と FUV という概念を用いたスケジューリング手法であり，分岐処理に対応している．CV とは，図 4 のように各演算処理に与えられるベクトルである．共通のビットに 1 が立っていない演算同士は互いに排他的であるため，同じコントロールステップでも同じ演算器に割り当て可能であるということを表す．CV は条件式の結果が利用可能かどうかにより動的に変化する．$FUV_f(k)$ は，コントロールステップ k において FU f に割り当てられた演算の CV の合計である．各コントロールステップでの割り当て可能な演算を求めるものである．CV と FUV を用いることにより，条件分岐の排他性を利用して演算器を有効活用し，なおかつ投機実行を可能にするという特徴をもつ．またこの手法では，データ転送制約テーブル，クリティカルパス長リスト，レイテンシの見積もりという概念を用い，クリティカルパス長を優先度としてリストスケジューリングを行う．

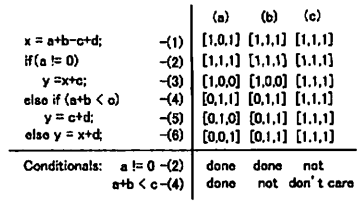

図 4 CV の例

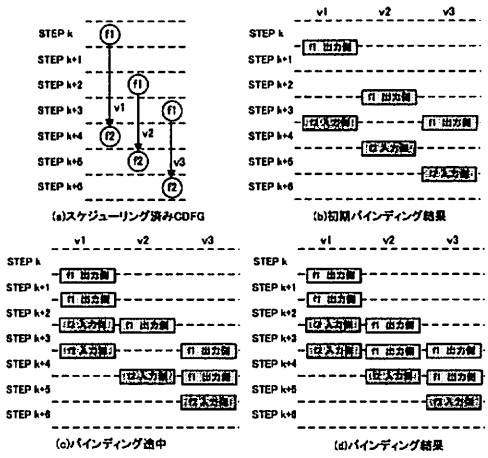

図 5 レジスタバインディング

3.2 レジスタバインディング

スケジューリング済みの CDFG のエッジは，アプリケーションプログラムにおける変数に対応している．レジスタバインディングの工程では，入力をスケジューリング済み CDFG とし，CDFG から抽出される全てのエッジの各コントロールステップにレジスタを対応づける．目的関数は総レジスタ数の最小化である．レジスタバインディング手法では，最初は入力側ローカルレジスタ数を 2 個 (固定)，出力側ローカルレジスタ数を 1 個 (可変) と考え，可能な限り出力側のレジスタで変数を保持するように変数をバインディングする．図 5(a) のスケジュー

リング済み DFG を例にレジスタ分散型レジスタバインディング手法を説明する．図 5 は，3 つの変数 (v_1,v_2,v_3) がいずれも演算器 f_1 から演算器 f_2 の一方の入力側ローカルレジスタへ転送される場合を例とした，バインディング過程である．ただし，f_1 から f_2 へのデータ転送は 1 クロックのレジスタ間データ転送を要するものとする．各変数のライフタイムの 1 クロック目 (例えば，v_1 ならば Step k〜Step k+1) は必ず f_1 の出力側レジスタに格納することにするため，初期バインディング結果は図 5(b) となる．その後，f_1 の出力側レジスタの数が 1 個，入力側ローカルレジスタの一方を用いる (もう一方は使用されている) という制約で，割り当てを行う．途中過程である図 5(c) において，v_2 がレジスタに未割り当てステップが残っているので，f_1 の出力側レジスタの制約が 1 個増やし再度割り当てを行う．最終的に，図 5(d) のバインディング結果が得られる．

3.3 モジュール配置

モジュール配置の工程では，データ構造に Sequence-pair [5] を用い，モジュール配置を SA によって最適化する．SA のコスト関数は，A をデッドスペースを含む回路の総面積，W を各モジュールを結ぶ総配線長，V を各演算器間のデータ転送においてクロック周期制約条件を違反した遅延の総合計としたとき，

$$\alpha A + \beta W + \gamma V \qquad (1)$$

と計算する．ただし，α, β, γ は任意のパラメータである．また，概略モジュール配置工程では，i 回目のイタレーションにおける最終的な配置結果を，$i+1$ 回目のイタレーションのモジュール配置工程における SA の初期配置として用いる．ただし，毎回初期温度が高い状態から SA によるモジュール配置を実行すると，フィードバックされた配置情報が反映されなくなるので，イタレーション毎に SA の初期温度を下げ，イタレーションの回数を重ねるごとにモジュール配置が固定し，解が収束されるようにする．合成フローの i 回目のイタレーションのモジュール配置工程における SA の初期温度を T_i とすると，任意のパラメータ $K > 1$ を用いて，

$$T_{i+1} = T_i / K \qquad (2)$$

とする．モジュール配置が終了した後，回路の面積及び実行時間が収束したとみなされなければ，スケジューリング/FU バインディングの工程へ配線遅延情報，配置結果をそれぞれフィードバックする．

4. ポート割当最適化によるマルチプレクサ削減手法

[8] の手法により，レジスタ分散型をターゲットアーキテクチャとした高位合成として，実用的なアプリケーションを対象とした回路の合成が可能となる．しかしレジスタ分散型アーキテクチャでは，全演算器に専用のローカルレジスタが付加することで，レジスタ間接続に要する配線数が増加してしまうため，必要なマルチプレクサ数の増大を招いてしまう．モジュール (演算器，ローカルレジスタ) 間の配線接続において，各データ転送ごとにどの配線を用いるのか，またどの入出力ポートに配線を接続するかで必要なマルチプレクサ数は大きく変動するため，

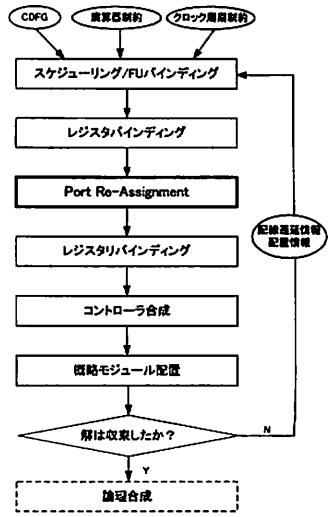

図 6 提案手法を加えた合成フロー

増加する配線に対し，適切なポート割当処理を行うことがマルチプレクサ数削減に必要な処理となる．[8] の手法では，FU バインディングの段階で，モジュール間の配線接続に対し適切な入出力ポート選ぶような割当が行われていないため，レジスタバインディング後に冗長な配線が増加し，マルチプレクサ数の増大を招くという問題点を生じる．

本章では，モジュール間の配線接続におけるポート割当最適化を目的とする Port Re-Assignment 手法 (以下 PRA 手法) を提案する．提案手法を組み込んだ合成フローを図 6 に示す．[8] で提案された手法で用いている合成フローをベースとし，レジスタバインディング後に PRA を行いポート割当最適化を行う．PRA を行った後，変更されたポート割当情報を元に再度レジスタバインディングを行う．その結果，変数が最適なレジスタに割り当てられ，必要とするマルチプレクサ数が削減される．

4.1 問題定義

PRA 問題を次のように定義する．任意の演算器 f_i の入出力に配置されたローカルレジスタをそれぞれ $R_{in}(i) = \left\{r^1_{in}(i), r^2_{in}(i)\right\}$, $R_{out}(i) = \left\{r^k_{out}(i), \mid k = 1,2,\ldots,n\right\}$ と表す．入力をスケジューリング/FU バインディング済み CDFG，レジスタバインディング結果とし，任意の演算器 f_h から演算器 f_i へのデータ転送を考える．このとき PRA 問題とは，各データ転送ごとに，f_h の出力，あるいは f_h の k 番目の出力側ローカルレジスタ $r^k_{out}(h)$ の出力を，f_i の入力側ローカルレジスタ $r^1_{in}(i)$，$r^2_{in}(i)$ の一方，もしくは両方の入力側ポートに割当を行うかを決定する問題である．目的関数は総マルチプレクサ数の最小化である．

4.2 PRA によるポート割当最適化

PRA では，f_i に配置された 2 つの入力側ローカルレジスタ $r^1_{in}(i)$，$r^2_{in}(i)$ に着眼し，$r^1_{in}(i)$，$r^2_{in}(i)$ の入力側ポートに対しどのように配線が接続されているかを調べる．そこで，f_h から f_i へのデータ転送がある際，転送元演算器である f_h の出力側ポート，あるいは f_h の k 番目の出力側ローカルレジスタの出

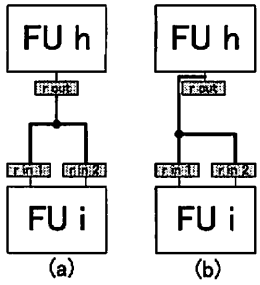

図 7 ポート再割当候補

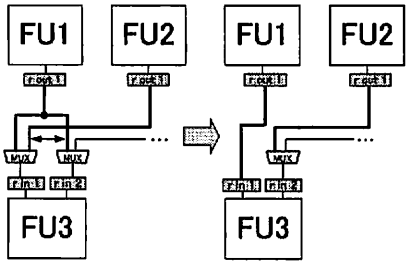

図 8 PRA の例

力側ポート $r^k_{out}(h)$ が，転送先ローカルレジスタ $r^1_{in}(i)$，$r^2_{in}(i)$ の両方の入力側ポートを利用するように配線が接続されている場合，冗長な配線が増えることによってマルチプレクサ数が増加する可能性がある．このため，PRA の工程ではこのような接続に対する割当を最適化することで，冗長な配線が減らしていく．PRA の候補となる接続を

- 転送元ローカルレジスタの出力側ポート⇒転送先ローカルレジスタの入力側ポート (図 7(a))
- 転送元演算器の出力側ポート⇒転送先ローカルレジスタの入力側ポート (図 7(b))

とする．各データ転送ごとにどのポートを用いているかを探索し，PRA の候補となる接続に対し，一方の入力側ローカルレジスタの入力側ポートに接続を割当直せるかどうかを調べ，ポート交換が可能な演算器 (加算器，乗算器，AND 演算) である場合，一方に割当を変更し固定していく．PRA の例を図 8 に示す．図 8(a) では，FU1 の出力側ローカルレジスタ $r^1_{out}(FU1)$ の出力側ポートが，転送先演算器 FU3 の入力側ローカルレジスタ $r^1_{in}(FU3)$，$r^2_{in}(FU3)$ の両方の入力側ポートと接続されているため，図 8(b) で表されるように一方に割当を変更し冗長な配線を削減する．

4.3 優先度リストを利用したポート割当手法

PRA の候補となる接続に対し，どの候補から順番に再割当を行うのか，また f_h から f_i への接続が候補であったとき，$r^1_{in}(i)$，$r^2_{in}(i)$ のどちらに割当を固定するのかを決定するため，優先度リストを利用したポート割当手法を行う．

まず，変数 (各データ転送) を全探索し，PRA の候補となる接続を検出する．そして，そのデータ転送回数 (接続頻度と呼ぶ) を元に優先度を決定する．$r^1_{in}(i)$ の入力側ポートへの接続頻度を p[回]，$r^2_{in}(i)$ の入力側ポートの接続頻度を q[回] とした時，$p+q$[回] を再割当候補に対する優先度とし，優先度リスト

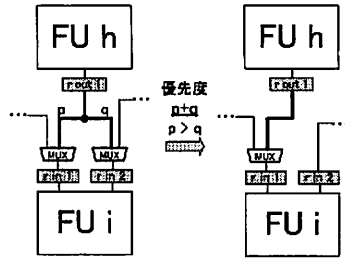

図 9 優先度決定，再割当

Step 1 レジスタバインディング結果より，各演算器，又は各出力側ローカルレジスタの出力側ポートから各入力側ローカルレジスタの入力側ポートへのデータ転送回数を算出する．

Step 2 Step1 を元に，PRA の候補となる接続を抽出する．各候補に対し優先度を算出し，優先度リストを作成する．

Step 3 優先度リストを元に，優先度が高い順に PRA を行う．リストが空の場合 PRA を終了する．ここで，転送先演算器が加算器，乗算器，AND 演算の場合，Step4 へ．それ以外の演算器の場合，該当候補に対する処理を打ち切りリストから削除する．

Step 4 再割当候補に対し，接続頻度の多いポートに再割当を行う．全ての接続に再割当を行った後リストから削除すし Step 3 へ．

図 10 提案 PRA 手順

表 1 演算器の面積と遅延

	面積 (um^2)	遅延 (ns)
加算器	287	1.36
減算器	318	1.37
乗算器	4507	2.93
比較器	148	0.88
AND 演算	68	0.03
シフタ	270	0.48
レジスタ	208	0.09
マルチプレクサ	112	0.04

を作成する．優先度リストを元に，優先度の高い候補から順番に PRA を行う．ここで，p と q で接続頻度の多いポートに割当を固定していく．図 9 に例を示す．

4.4 手　　順

提案 PRA 手法の手順を図 10 に示す．この手順はスケジューリング済み CDFG とレジスタバインディング結果を入力とし，総マルチプレクサ数を最小とするようにポート割当最適化を行う．この手順によりレジスタバインディング結果に変更を加え，ポートを再割当する．その後，再度レジスタバインディングを行うことにより変数が最適なレジスタに割り当てられ，必要とするマルチプレクサ数が削減される．

5. 計算機実験結果

提案手法を C 言語を用いて計算機上に実装した．実験環境は OS が Debian/Sarge，CPU が Intel Core2 Duo 1.80GHz，メモリ容量が 2GB，C コンパイラが gcc(egcs-3.3.5) である．

演算器は 16bit 幅と仮定し，面積/遅延は Synopsys 社 Design Compiler を用い，セルライブラリには STARC(注1)(CMOS

(注1)：STARC[90nm] ライブラリは東京大学大規模集積システム設計教育研究センターを通し，株式会社半導体理工学研究センター（STARC）と株式会社先端 SoC 基盤技術開発（ASPLA）の協力で開発されたものである．

表 2 実験結果

App.	演算器制約	手法	面積 [um^2]	コントロールステップ数	ローカルレジスタ数	MUX 数	CPU time (sec)
DCT	+3*3	[8]	31330	14	25	65	201
		提案手法	30030	14	25	59	134
	+4*4	[8]	36995	9	28	65	176
		提案手法	36019	9	28	58	132
FIR	+2*2	[8]	22528	41	23	45	256
		提案手法	21971	41	23	43	185
	+3*3	[8]	28710	33	26	48	206
		提案手法	28362	33	26	47	227
EWF	+1*1	[8]	12768	28	13	29	181
		提案手法	12540	28	13	26	216
	+2*1	[8]	13740	22	12	29	189
		提案手法	13222	22	12	25	212
EWF3	+2*1	[8]	17466	60	16	45	435
		提案手法	15368	60	16	34	440
	+3*2	[8]	26720	54	22	62	379
		提案手法	25024	54	21	48	390
copy	+3-1<1*5	[8]	127104	148	144	378	3481
	AND1 shift2	提案手法	121864	144	146	358	5619
	+4-2<1*6	[8]	140685	144	161	403	3581
	AND1 shift2	提案手法	136822	143	158	385	7382

90[nm]) を用い，あらかじめ合成して得られた値を利用した．実験で用いた演算器の面積，遅延を表 1 に示す．配線遅延は配線長の 2 乗に比例すると仮定し，250[μm] 当たり 1[ns] と設定した．各演算器の入出力ポートはモジュールの中心にあると仮定し，クロック周期制約は 1.8[ns] とした．また，実験結果における総面積は，演算器，レジスタ，マルチプレクサ，制御回路の面積を含む値であり，配置のデッドスペースも含んだ評価となっている．CPU Time はイタレーションを含む合成フロー全体にかかる時間である．

対象としたアプリケーションは以下の 5 つである．

- DCT：ノード数 48
- FIR(4 並列 7 次 FIR フィルタ)：ノード数 75
- EWF：ノード数 34
- EWF3(EWF を 3 連結したもの)：ノード数 102
- copy(複写機アプリケーション)：ノード数 378，分岐有

実験結果を表 2 に示す．実験結果より，提案手法は [8] と比較して，コントロールステップ数，ローカルレジスタ数をほぼ維持しつつ，平均で 10.9%マルチプレクサ数を削減し，平均で 4.9%総面積を削減している．これより提案手法を組み込んだ場合，[8] と同等の性能を維持しながらマルチプクレサ数，総面積を削減できることを確認した．

6. むすび

本稿では，レジスタ分散型アーキテクチャを対象とする高位合成手法のためのマルチプレクサ削減手法を提案した．計算機実験により提案手法はレジスタ分散型アーキテクチャを対象とする従来の高位合成手法 [8] と比較し，コントロールステップ数，ローカルレジスタ数をほぼ維持しつつ，平均で 10.9%マルチプレクサ数を削減でき，平均で 4.9%総面積を削減することを確認できた．今後の課題は，全体フローを通してマルチプレクサ数を削減する手順の構築である．

文　献

[1] D. Chen and J. Cong, "Register binding and port assignment for multiplexer optimization," in *Proc. ASP-DAC 2004*, pp. 68–73, 2004.

[2] Y. M. Fang and D. F. Wong, "Simultaneous functional-unit binding and floorplanning," in *Proc. ICCAD 1994*, pp. 317–321, 1994.

[3] J. Jeon, D. Kim, D. Shin, and K. Choi, "High-level synthesis under multi-cycle interconnect delay," in *Proc. ASP-DAC 2001*, pp. 662-667, 2001.

[4] D. Kim, J. Jung, S. Lee, J. Jeon, and K. Choi, "Behavior-to-placed RTL synthesis with performance-driven placement," in *Proc. ICCAD 2001*, pp. 320–325, 2001.

[5] H. Murata, K. Fujiyosho, and S. Nakatake, "Rectangle-packing-based module placement," in *Proc. ICCAD 1995*, pp. 472–479, 1995.

[6] 大森健児, "遺伝的アルゴリズムによる高レベル合成," 電子情報通信学会論文誌 A, vol. J81-A, no. 5, pp. 854–862, 1998.

[7] S. Raje and R. Bergamaschi, "Generalized resource sharing," in *Proc. ICCAD 1997*, pp. 326–332, 1997.

[8] 田中真, 内田純平, 宮岡祐一郎, 戸川望, 柳澤政生, 大附辰夫, "レジスタ分散型アーキテクチャを対象とするフロアプランとタイミング制約を考慮した高位合成手法," 情報処理学会論文誌, vol. 46, no. 6, pp.1383–1394, 2005.

[9] K. Wakabayashi and T. Yoshimura, "A resource sharing and control synthesis method for conditional branches," in *Proc. ICCAD 1989*, pp. 62–65, 1989.

[10] J. P. Weng and A. C. Parker, "3D scheduling: High-level synthesis with floorplanning," in *Proc. 28th ACM/IEEE DAC 1992*, pp. 668–673, 1991.

[11] S. Yamada, "An optimal block terminal assignment algorithm for VLSI data path allocation," *IEICE* Transactions on Fundamentals of Electronics Communications & Computer Sciences, vol. E80-A, no. 3, pp.564–566, Mar. 1997.