汎用エンジンを対象とするアプリケーション設計の自動化

1L－3

橋本匡史，日高哲也†，奥田 知史，沼 昌宏，平野 浩太郎
神戸大学　†中国電力（株）

1. はじめに

大きな計算量を必要とするLSI CAD処理や画像処理の高速化のために，内部の論理を電気的に書き替え可能なLSIであるFPGA（Field Programmable Gate Array）とメモリから構成された汎用エンジン[1]の概念が提案されている。この汎用エンジンのアプリケーション設計を自動化するために，高位合成システム RMAC（Reconfigurable Machine Application Compiler）を開発した。従来の高位合成手法をそのまま適用すると，変数へのレジスタ割当てなどを行うアロケーションが終了してから回路分割を行うことになる。そのため，図1(a)に示すようにReg 3によってレジスタを無理に共有したとき，新たなFPGA 間配線が必要になることがある。本稿では，以下の特徴をもつアロケーションと回路分割によって，図1(b)のような結果を得る手法を提案する。

i) LUT数などの見積もりによるFPGAの有効利用
ii) クラスタリングによるデータパス分割の回避
iii) 汎用エンジンのアーキテクチャを考慮したアロケーションと回路分割の並行処理による，人手に近い設計結果の取得

2. 高位合成システム：RMAC

2.1 RMACの仕様

RMACは，クロックを明示したC言語の動作記述を入力して，最終的に汎用エンジンにアプリケーションを構築するためのゲートレベル回路を記述したファイルを出力する。そのほか，回路マクロ名，入出力端子名，使用するLUT数，FF数などを定義したライブラリも入力する。現段階では，スケジューリングは動作記述の時点で人手により行う。クロックの同期タイミングを指定するには，その部分に“$”を記述する。また，汎用エンジンのアプリケーションを開発するための最小限の機能にとどめ，利用できる演算子や予約語を表1のように決定した。

表1　RMACの入力記述

項目	内容
変数の型	int，char
演算子	& | ~ ^ + - ++ -- << >> * += -= <<= >>= *= |= &= && || != == <= >= < >
制御文	if ～ else ～ 文 for文，while文，do ～ while文
クロック明示	（式）；$

2.2 RMACの処理概要

RMACの処理概要について述べる。

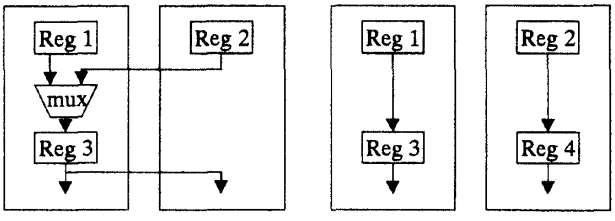
(1) CFGとDFGの作成

動作記述をもとにCDFG（Control Data Flow Graph）を作成する。CDFGは，分岐やループといった制御を表すCFG（Control Flow Graph）と，データの依存関係を表すDFG（Data Flow Graph）からなる。

(2) クリティカルDFGからのデータパス合成とアロケーション

CDFGに含まれる複数のDFGから，まずクリティカルDFG，すなわち最も実行回数の多いDFGを取り出し，そのDFGに含まれるメモリをシードとするクラスタリングによってデータパスを合成する。メモリをシードに選ぶのは，メモリに近いFPGAにデータパスを実現するためである。また，このクラスタリングと並行して，次節で述べるアロケーション，LUT数とFF数の見積もりの各処理を行う。

このようにクリティカルDFGに対するデータパス合成を優先して行うことで，実行回数の多い部分の遅延増加を防ぐ。すなわち，使用するメモリバン



(a) 順次処理結果　(b) 並行処理結果

図1　アロケーションと回路分割結果の例

Design Automation for Reconfigurable Machine Applications
Koji Hashimoto, Tetsuya Hidaka†, Tomofumi Okuda, Masahiro Numa, and Kotaro Hirano
Kobe University, †The Chugoku Electric Power Co., Inc.

クと直結するFPGAに可能な限り実行回数の多いデータパス部分を収めることで，複数のFPGAに分割されることによる遅延の増加を防ぐ。また，アロケーション，回路分割の際にLUT数やFF数の見積もりを行うことで，早い段階で結果を見通した処理が可能となり，FPGA内部の資源を可能な限り有効に利用できる。

(3) **制御回路の作成**

CFG，DFGの順序通りに演算器などのリソースを動作させるために，制御回路を合成する。可能な限り，制御対象となるデータパスが実現されるFPGAに制御回路を収めることで，無駄な分割による遅延の増加を避ける。

3. アロケーションと回路分割の並行処理

図1 (b) のような結果を得るために，クラスタリングによるデータパス合成と並行してアロケーションを行い，LUT数とFF数に関する見積もり結果をもとに回路を分割する。最終的にFPGAにマッピングした結果を見通しながら，アロケーションとクラスタリングを並行して行う。とくに，以下の2種類のアロケーションを併用することで，汎用エンジンRM-IV [2] で利用している配置変更用LSIによる遅延を考慮した結果を得る。

(1) ノーマル・アロケーション

最初は，DFGの各ノードに対してそのまま単純にリソースを割り当てるノーマル・アロケーションを用いてクラスタリングを行う。ただし，無駄なデータパスを作らないように，同じ変数ノードは同一のレジスタに割り当てる。

(2) 共有アロケーション

(1) の結果，利用するメモリバンク数を超えるFPGAを必要とする場合，共有アロケーションを用いたクラスタリングを行う。共有したときに最も総LUT数が少なくなるリソースを共有対象とする。また，交換可能な変数について入れ替え（ペア交換）後のLUT数を評価し，減少する場合には入れ替える。

4. 実験と考察

人手設計とRMACとの性能の比較を行った。バブルソート，ウェーブレット変換（WTE）の2種類について，実際にRMACによってゲートレベルの回路を合成した。その結果に対して，遅延を考慮した論理シミュレーションを行った。結果を表2に示す。

表2 設計結果の比較

項目	バブルソート		WTE	
	人手	RMAC	人手	RMAC
設計時間	約1日	3.29 s	約4日	9.77 s
使用メモリ数	1	1	16	4
使用FPGA数	1	1	16	4
平均LUT利用率 (%)	35.0	68.0	31.9	19.0
平均FF利用率 (%)	13.0	19.0	17.3	7.75
状態数	8	12	5	10
動作周波数 (MHz)	8	8	6	8
処理時間 (s)	235	359	0.00068	0.0177

表2から，設計時間については当然ながら従来の人手設計に比べて大幅に短縮されている。処理時間については，バブルソートで人手設計の1.5倍，WTEで25倍を要している。この理由として，クロックの挿入といったスケジューリングを未だ自動化しておらず，冗長な状態数が多くなることが考えられる。さらにWTEについては，人手設計では四つの回路を並列動作させ，パイプライン処理を行っているのに対して，RMACの合成結果では，一つの回路で処理していることが，速度低下の要因となっている。以上のことから，スケジューリングの自動化，パイプライン処理への対応などが今後の課題になる。

その一方で，RMACによる合成結果では高い動作周波数が得られている。よって，アロケーションと回路分割については，汎用エンジンのアーキテクチャを考慮したリソース共有を行う人手設計と同等の設計結果が得られたと考える。

5. まとめ

汎用エンジンのアプリケーションを対象とする高位合成システムRMACを開発した。汎用エンジンのアーキテクチャを考慮したアロケーションと回路分割の並行処理によって，人手設計に近い結果が得られることを確認した。今後の課題として，スケジューリングの自動化，パイプラインへの対応などが挙げられる。

参考文献

[1] 富田 昌宏, 村田 之広, 菅沼 直昭, 平野 浩太郎, "汎用エンジンRM-Iの開発", 情報処理学会第45回全国大会講演論文集, vol. 6, pp. 155-156, 1992年10月.

[2] 井上 真一, 奥田 知史, 高瀬 幹, 沼 昌宏, 平野 浩太郎, "汎用エンジンRM-IVとその応用", DAシンポジウム'96, pp. 99-104, 1996年8月.