A-01

動的再構成プロセッサ用コンフィギュレーション自動生成の一手法

An Automatic Configuration for Dynamic Reconfigurable Processor

吉田 敦郎†　　田中 照人‡　　神戸 尚志††
Atsurou Yoshida　Teruhito Tanaka　Takashi Kambe

1.　はじめに

動的再構成プロセッサ DAPDNA-2 は従来のプロセッサとは異なり、PE(Processing Element)を用いた粗粒度構成をとる場合が多いため、PE を効率よく用いることができる専用の設計環境が重要である。ベンダーからは、GUI を用いて対話的に設計するツール、専用言語による記述、C 言語から回路合成などのツールが提供されているが、制約が多く、PE についての詳細な知識が必要とするなどの問題もある。本文では高級プログラミング言語を入力とし、for、while、if などの制御文を効率的に PE に割りつけることにより、PE 単位のパイプライン回路を自動合成するアルゴリズムについて提案する。

2．　DAPDNA-2

DAPDNA-2 は DAPDNA アーキテクチャを採用した 2 世代目の LSI である。この LSI は主にシーケンシャルな処理を担当する 32 ビット RISC プロセッサコア DAP(Digital Application Processor)と、2 次元アレイ構造で再構成可能な部分に当たる DNA(Distributed Network Architecture)からなる。

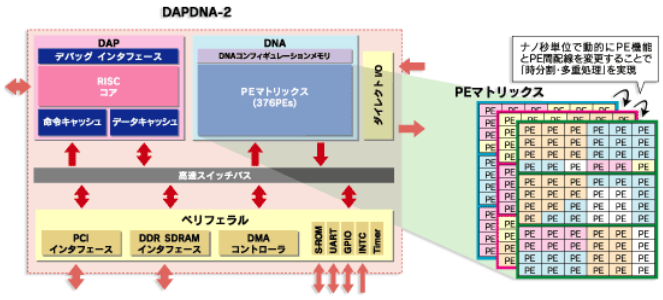

図 1　DAPDNA-2 のブロック図

32 ビット RISC プロセッサである DAP の動作周波数は、この LSI のほかの回路ブロックと同じ 166MHz であり、DNA に対してマスターとなる。用途としては、一般的な汎用プロセッサとしての利用や、PE マトリックスの動的再構成の制御などに使用する。

DNAはPEによるマトリックス構造をとる。PEマトリックスは、各PEの動作設定とPE間の接続設定によって定義され、これを1面のDNAコンフィギュレーションとしてプログラムされる。

DNA コンフィギュレーションは実行バンクと 3 つのバックグランドバンクに格納されており、DNA コンフィギュレーションを動的に切り替えることにより、複雑な処理や大規模な回路を実現する。

DAPDNA-2 には、異なる処理に対応できるようにいくつかの種類の PE が用意されている。PE の一覧を表 1 に示す。

表 1　PE 一覧

PE	機能
EXE	各種演算
Delay	遅延
RAM	内部メモリ
カウンタ	カウンタとして動作
LDB/LDX	DNA への入力部
STB/STX	DNA 外部への出力部

データ処理用の PE には EXE、Delay、RAM などがあるが、ここでは演算やパイプライン化を行う上で重要な役割を持つ EXE エレメントについて述べる。EXE エレメントの内部構造を図 2 に示す。

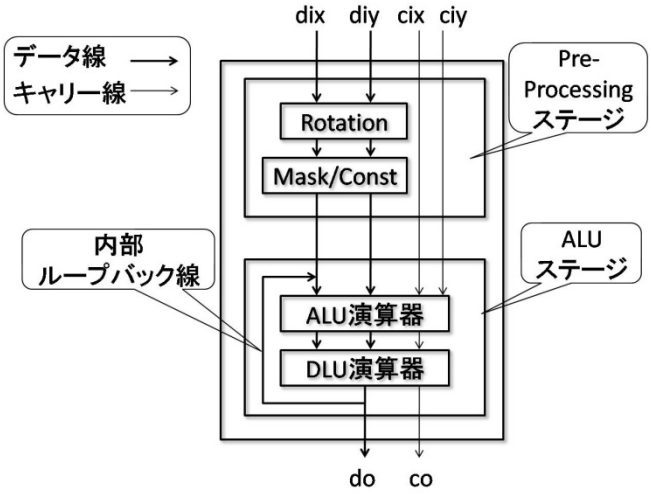

図 2　EXE エレメントの内部構造

DAPDNA-2 では処理値を表す 32bit の「データ」と、主に条件判断の結果に利用され PE の動作を変える 2bit の「キャリー」を扱う。EXE エレメントは Pre-Processing ステージと ALU ステージの 2 ステージから構成されており、各演算器はパイプライン構造を持つ。PE 単位のパイプラ

†近畿大学大学院　総合理工学研究科　エレクトロニクス系工学専攻
‡NEC エレクトロニクス株式会社
††近畿大学　理工学部　電気電子工学科

イン化は PE 数を殆ど増加させずに処理の高速化を実現でき、低消費電力化にも有効である。一方、演算器によって実行可能な演算が異なり、一つのコンフィグレーション中の PE 数に制限があるため、演算の実装方法によって処理性能と使用リソースが大きく変化する。このため、演算の PE への割りつけに工夫が必要である。

3. DAPDNA-2 用アルゴリズム変換

ソフトウェアアルゴリズムを DAPDNA-2 で実現する上で、制御構造の取り扱いが最も重要である。本文ではループ処理に焦点を当て、ループパイプライン化を中心とした DNA コンフィギュレーションの自動生成法を提案する。

図 3 に今回提案する変換手法の概略を示す。CDFG に含まれる各種制御構造をパイプライン化していくために、2 段階で探索を行なう。その後、各演算の PE へのアロケーション、各 PE の実行タイミングを調整するスケジューリングを行い、最後に全体のコンフィグレーションを生成する。

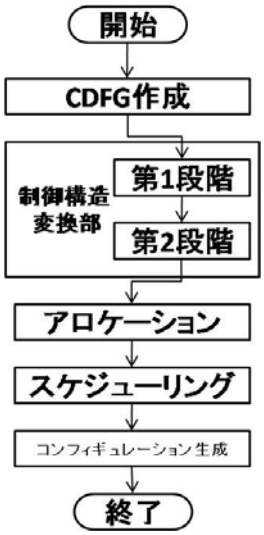

図 3 全体の流れ

(第 1 段階)繰り返しブロック探索

まず、内部ループバックループパイプライン化を適用する。次に、PE へデータが連続入力される場合、PE 標準パイプラインを実現する。

(第 2 段階)比較演算ノード探索

条件文に当たる比較演算を変換する。

3.1 コントロールデータフローグラフ

本アルゴリズムでは 2 種類のコントロールデータフローグラフ「CDFG」及び「D2-CDFG」を用いる。

CDFG は入力となるソフトウェアアルゴリズムを内部表現すため、演算を表す「ノード」、それらの依存関係を表す「エッジ」、及びノードとエッジからなる部分グラフを表す「ブロック」で構成される。ブロックには「基本ブロック」と繰り返し制御を表す「繰り返しブロック」の 2 種類がある。特に、対象アルゴリズムのデータ入力が行われるノードを「START」ノード、結果を出力するノードを「END」ノードと呼ぶ。繰り返しブロックの例を図 4 に示す。

- 基本ブロック

選択・算術・論理・比較演算を表すノードと、ノード間の依存関係を表すエッジで構成される部分グラフ。

- 繰り返しブロック

ループ処理は「繰り返しフラグ」と、内部に 1 つ以上の「基本ブロック」という要素で構成される。代表的な構成は以下の通り。

- 不定回ループ
  「繰り返しフラグ」+「1 つ以上の基本ブロック」
- 定数回ループ
  「繰り返しフラグ」+「1 つ以上の基本ブロック」+「ループ変数更新部」

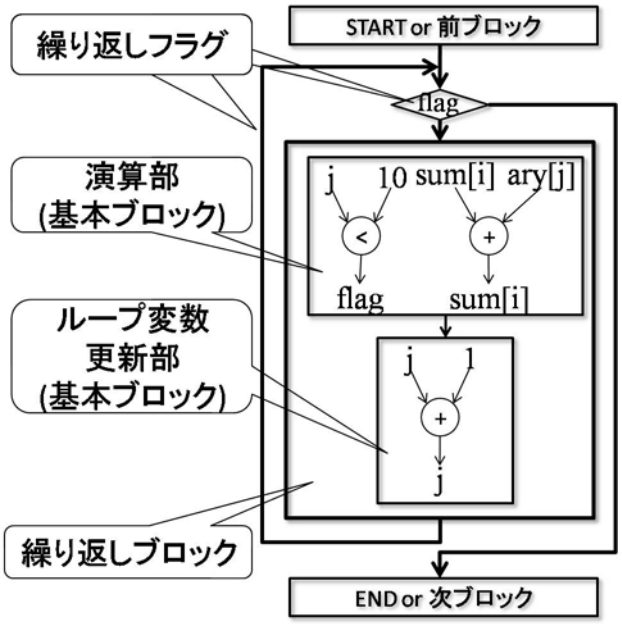

図 4　繰り返しブロック仕様

CDFG を DAPDNA-2 用に変換したグラフを、DAPDNA-2 用 CDFG(D2-CDFG)と呼ぶ。このグラフのノードは DAPDNA-2 の演算器に対応し、制御とデータの流れを表現したグラフである。D2-CDFG では、データの流れを実線エッジで、キャリーの流れを点線エッジで表現する。

3.2 制御構造変換の第 1 段階(ループパイプライン化)

DAPDNA のパイプライン構造は各 PE で入力データの数分だけの演算を行うデータフロー型なので、定数回繰り返しは特に制御を必要としない。これを「PE 標準パイプライン」と呼ぶ。一方、PE 単体の演算で繰り返し演算が出来る場合は、内部ループバック線を使用する。これを「PE 内部ループバックパイプライン」と呼ぶ。不定回の繰り返しの場合は、他ノードへのループバック線を用いて実現する。

第 1 段階では、CDFG を入力として各繰り返しブロックを探索し、繰り返し制御部を D2-CDFG に変換する。

3.2.1 PE内部ループバックパイプライン

PE 内部ループバックパイプラインは PE への実装方法の違いから、出力が配列になる「配列型」と、1 つの変数になる「単変数型」がある。ここでは配列型の PE 内部ループバックパイプラインが適用可能な CDFG の例を図 5 に示す。

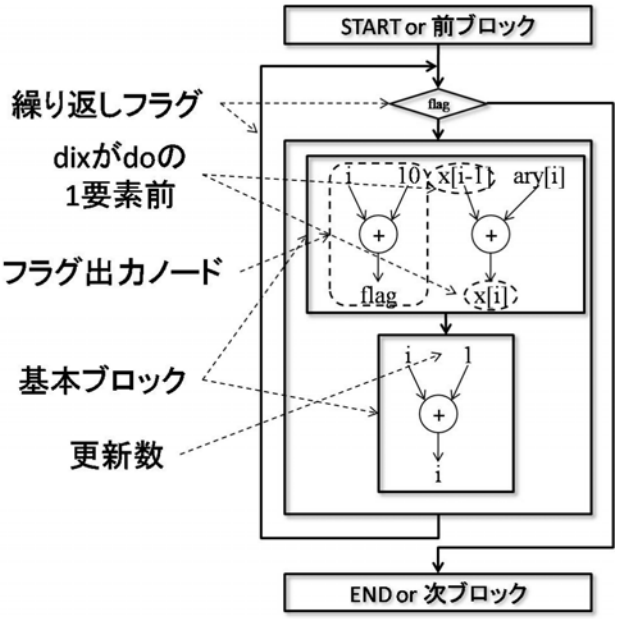

図 5　配列型 PE 内部ループバックパイプライン
適用可能例

START から END までのすべてのパスについて、ブロックの探索を行う。図 5 の CDFG の場合、PE 内部ループバックパイプラインを適用出来、生成した D2-CDFG を図 6 に示す。D2-CDFG の各ノードは、アロケーションのステップで PE 内部の演算器に割りつけられる。

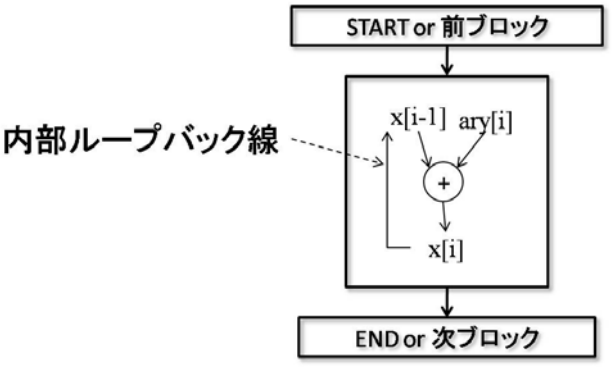

図 6　配列型 PE 内部ループバックパイプラインの適用

3.2.2　PE標準パイプライン

PE 内部ループバックパイプラインが適用できない繰り返しブロックに対し、PE の並列処理または PE 標準パイプラインを適用することによって繰り返しブロックを D2-CDFG に変換する。

PE 標準パイプラインが適用できる CDFG の例を図 7 に示す。PE の並列処理と PE 標準パイプラインが適用できる CDFG は基本的に同じ構造である。DAPDNA-2 ではパイプライン化が有効なため、PE 標準パイプラインを優先して適応する。

しかし、繰り返しブロックがネストされていて、ネストブロック内に単変数型 PE 内部ループバックパイプラインが存在する場合は PE 標準パイプラインを適応することはできないので、PE の並列処理を適応する。

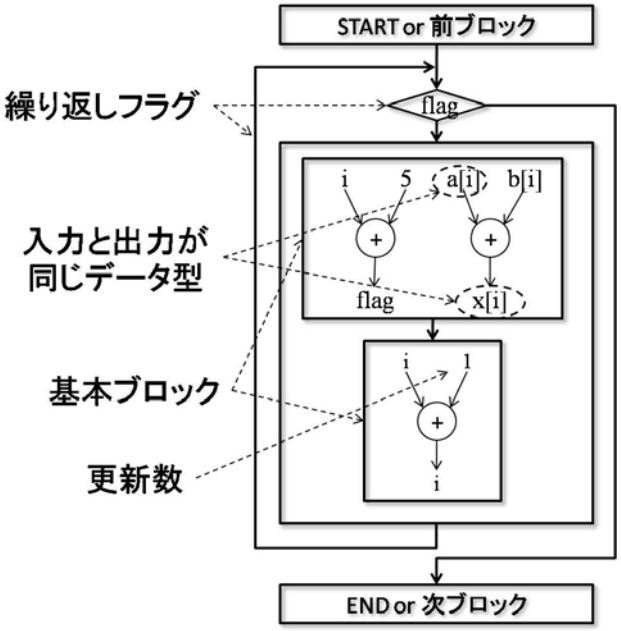

図 7　PE 標準パイプライン適用可能例

この例では、同じ長さの配列 2 つによる同要素数の値の繰り返し演算である。並列化もしくは PE 標準パイプライン化が適用可能である。図 7 のグラフを D2-CDFG に変換した結果を図 8 に示す。

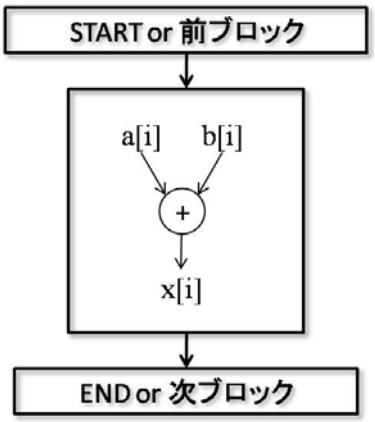

図 8　PE 標準パイプライン適用後

3.3　制御構造変換の第 2 段階(条件判断)

第 2 段階では繰り返しブロックを変換した D2-CDFG 部を含む CDFG を入力として、グラフ中の条件判断のアルゴリズム変換を行い、すべてのブロックを D2-CDFG 表現とする。

ソフトウェアアルゴリズムにおける条件文処理は、まず条件判定を行い、条件に合致する構文のみを実行する。DAPDNA-2 の条件文処理は、入力されたデータに対して条件判定を比較演算で行い、その結果をキャリー信号で出力する。これを選択演算が可能な演算器に入力し、各条件下

の処理文の実行結果を選択出力する。このように条件選択が決定されるよりも先に真偽両方の処理を行う投機実行型である。

条件判断の CDFG の例を図 9 に示す。条件判定にあたる「比較演算部」と、判定結果を反映する「選択演算部」で構成され、実装の際にはキャリー信号を別の PE に出力して選択処理を行う「キャリー信号型」と、1 つの PE で条件判断を実現する「PE 単独型」2 種類がある。

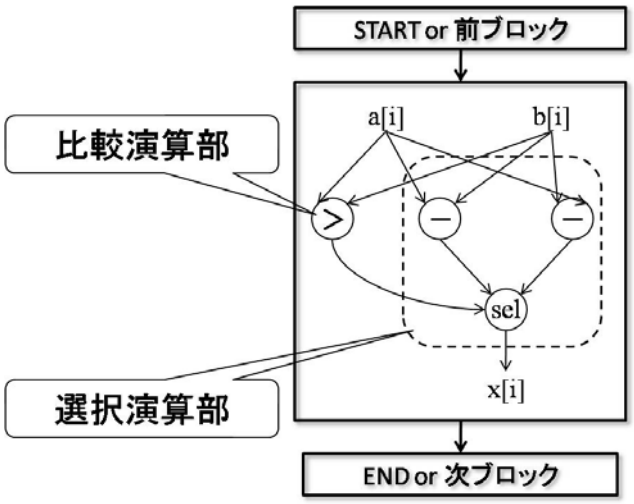

図 9　CDFG 上での条件判断

3.3.1　キャリー信号型条件判断

キャリー信号型では別の PE にキャリー信号を出力して、条件判断を実現する。D2-CDFG への変換は比較演算部の出力を点線エッジに変更するだけで、グラフの構造は変化しない。演算器は 2 入力なので、分岐数に応じて使用する演算器の数も増加する。

3.3.2　PE単独型条件判断

条件判定部と処理選択部を 1 つの PE で処理する実装できる「PE 単独型条件判断」には、以下の 2 種類がある。

① ALU 演算器に搭載されている比較演算 CMP 系の動作を利用することで、条件判定と選択処理を ALU 演算器 1 つで実現する。

② ALU 演算器で条件判定を行い、発生したキャリー信号を DLU 演算器に入力して DLU 演算器で処理選択を実行する。

4. PE のアロケーションとスケジューリング

D2-CDFG に変換されたグラフは、ノードの PE への割り付け（アロケーション）と、ディレイノードの挿入によるタイミング調整（スケジューリング）を行い、DNA コンフィギュレーションを生成する。

4.1　演算器のアロケーション

最初に、各ノードの演算子に対し、割り当て可能な演算器候補の情報を生成する。各ノードの割り当て可能な演算器の数を「割り当て自由度」と呼ぶ。この「割り当て自由度」の低いノードを優先しつつ、各ノードをデータフローに沿って PE に割り当てる

4.2　スケジューリング

DAPDNA-2 では入力から出力まで使用する PE の数によって最大遅延が決定される。アロケーション後の D2-CDFG では演算器単位でノードを構成していたが、PE 単位にノードの表現単位を拡大する。ノードの表現単位が PE であり、各ノードが PE の遅延情報を持つグラフを「タイミンググラフ」と呼ぶ。ディレイノードを追加・移動する手順を以下に示す。

① タイミンググラフについて、対象ノードの入力タイミング（根～親ノードまでの遅延数の合計）を算出

② 対象ノードの双方の親ノードの入力タイミングを比較して、同じ遅延数になるように、ディレイノードを挿入。

5. まとめ

動的再構成プロセッサのコンフィギュレーション自動生成アルゴリズムの提案を行った。ループ処理のパイプライン化に焦点を当て、デバイスの特性を生かしたコンフィグレーションの自動設計を行う手法である。

今後、本文のアルゴリズムに基づいたプログラムの設計と、大規模なアルゴリズムに対する適用とその評価を行う。

参考文献

[1] DAPDNA-2 DNA リファレンス,IPFlex 株式会社

[2] 藤田昌宏：システム LSI 設計工学,オーム社

[3] 築山修治：アルゴリズムとデータ構造の設計法,コロナ社

[4] D.A. Patterson, J.L. Hennessy：コンピュータの構成と設計,日経 BP 社,1999

[5] 吉田敦郎、他：“リコンフィギュラブルプロセッサを用いたリードソロモン符号復号化回路の設計”、電子情報通信学会 VLSI 設計技術研究会、2008 年 3 月

[6] 天野英晴：“（特別講演）最近のリコンフィギャラブルシステム,動的リコンフィギャラブルシステムの動向”、電子情報通信学会 IEICE Technical Report SR2005-5(2005-5)

[7] Takayuki S., et al. “Dynamically Reconfigurable Processor Implemented with IPFlex’s DAPDNA Technology”, IEICE TRANS. INF. & SYST., VOL.E87-D, NO.8 AUGUST 2004

[8] Takao T., et al. “High-Level Synthesis Challenges and Solutions for a Dynamically Reconfigurable Processor”, Proceedings of 2006 IEEE/ACM ICCAD, 2006.