社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

低消費電力プロセッサのための限定的動的再構成アーキテクチャ

平尾　岳志† 　金　　多厚† 　肥田　　格† 　浅井　哲也† 　本村　真人†

† 北海道大学 大学院情報科学研究科 〒060-0814 札幌市北区北 14 条西 9 丁目
E-mail: hirao@lalsie.ist.hokudai.ac.jp

あらまし　高性能・高エネルギー効率のプロセッサを実現するアプローチとして、対象プログラムのホットパスを可変構造のデータパスにマッピングしアクセラレートするリコンフィギュラブルプロセッサが注目されている。本稿では、処理の内容が多岐にわたり、かつ低消費電力性が強く求められる組込み用途をターゲットとし、Control-Flow Driven Data-Flow Switching (CDDS) 可変データパスアーキテクチャを提案する。このアーキテクチャは、(1) 動的再構成を必要最小限な範囲に限定することで、柔軟性と低消費電力性の両立を目指す、(2) 既存命令列をそのままデータパスにマッピングすることで、既存アーキテクチャからスムーズに移行可能なリコンフィギュラブルプロセッサを目指す、の 2 点をその特徴とする。小規模なプログラムによる予備評価の結果、CDDS アクセラレータは、ベースプロセッサに比べ、約 3〜6 倍の性能電力比の向上を達成することができた。

キーワード　リコンフィギュラブルシステム, プロセッサアーキテクチャ, 組込みシステム

A Restricted Dynamically Reconfigurable Architecture for Low Power Processors

Takeshi HIRAO†, Kim DAHOO†, Hida ITARU†, Tetsuya ASAI†, and Masato MOTOMURA†

† Graduate School of Information Science and Technology (IST), Hokkaido University
Kita 14, Nishi 9, Kita-ku, Sapporo, 060-0814 Japan
E-mail: hirao@lalsie.ist.hokudai.ac.jp

Abstract　Reconfigurable processors have widely attracted attention as an approach to realize high-performance and highly energy-efficient processors that map a target program's hot path to a reconfigurable datapath. In this paper, we propose a Control-Flow Driven Data-Flow Switching (CDDS) variable datapath architecture for embedded applications that demand extremely low power consumption in a wide range of uses. This architecture is characterized by following two features: (1) achieving both flexibility and low energy consumption by limiting the scope of the dynamic reconfiguration, (2) realizing smooth migration from the existing architecture by mapping the existing instruction sequence to the datapath. Preliminary evaluation on small programs have revealed that the CDDS accelerator achieves approximately 3 to 6 times the performance/power improvements, compared to a base processor.

Key words　Reconfigurable system, Processor architecture, Embedded system

1. はじめに

近年、センサーネットワークやモバイル端末等のバッテリ駆動型機器の市場拡大に伴い、低消費電力な組み込みプロセッサに対するニーズが高まっている。これまで、プロセッサのコア数を増やして並列処理することで性能電力比を上げる、マルチプロセッサが提案されて来た。しかし、従来の単体プロセッサ構造を基本的にはそのまま踏襲しているために、性能電力比の大幅な向上は見込めない。汎用プロセッサの消費電力内訳の一例 [1] では、命令読み込み、制御、レジスタアクセス、パイプラインレジスタ、Arithmetic Logic Unit(ALU) で全体電力の 2/3 以上を消費している。ALU の電力（高々10%）を演算に必要な電力と考えると、その他の電力は所望の演算を ALU で実行させるために消費される電力であり、工夫次第で削減可能な冗長な電力であると言える。そこで、プロセッサ内にアクセラレータ (=拡張データパス) を附置し、プログラム内のホットパス部をハードウェア的に実行することで、このような冗長な電力を削減して性能電力比向上を達成する各種のプロセッサアー

キテクチャが提案されて来た。

Green Droid [2] などの Configurable Processor はホットパス部をハードウェア化してアクセラレータとして実装し、実行するため、汎用プロセッサに比べ、大幅な性能電力比の向上を見込む。しかし、ターゲットアプリケーション専用であるため、汎用性がない。そこで、再構成可能なハードウェアアクセラレータを持つ、Reconfigurable Processor が提案された。これは、動作前にアクセラレータを再構成することが可能で、汎用性を持ちつつ、性能電力比の向上を狙う。しかし、アクセラレータを多数の Processing Element(PE) で実装しても、大規模なプログラムを一度にマッピングすることはできない。この問題を解決するのが Dynamically Reconfigurable Processor(DRP) [3] である。DRP は、アプリケーションを時分割し、アクセラレータを実行時に再構成することで大規模なプログラムを実行することができる。しかし、再構成するときに回路の充放電電流が流れ、再構成を頻繁に行う場合には多くの電力を消費する。このように、汎用性と性能電力比にはトレードオフの関係がある。

本研究では、例えば組み込み制御応用を想定し、制御フローの頻繁な切り替えを含むような、より広範囲なアプリケーションに対して性能電力比の向上を目指した Control-Flow Driven Data-Flow Switching(CDDS) 可変データパスアーキテクチャを提案する。CDDS アーキテクチャでは、アクセラレータの実行に柔軟性を持たせつつも、可能な限り動的再構成の範囲を減らすことを目指す。はじめに、データパスを動的再構成する可変部と動的再構成しない固定部に分け、分岐命令の結果により、可変部のみ限定的に再構成する。これにより、アクセラレータ全体の再構成が必要でない分、従来の DRP に比べて再構成時の動作電流を削減し、低消費電力化を狙う。アクセラレータ上で分岐を多く含むホットパスも後述のコンテキスト数が許す限り実行可能である。

本論文の構成は以下の通りである。2 章で提案する CDDS アーキテクチャの概要を説明する。3 章で CDDS アーキテクチャの詳細な設計を説明する。4 章で複数のプログラムを用いた評価結果について述べる。5 章で総括する。

2. CDDS アーキテクチャ概要

複雑なコントロール/データフローグラフ (CDFG) を再構成可能なアクセラレータにマッピングすることは、困難を伴う。そのため、はじめに CDFG について考察する。プログラムを CDFG で表したときに、コントロールフローの分岐点で、次に実行するデータフローが選択される。このとき、実際に変更されるデータフローは実行中のデータフローから、選択したデータフローへの接続部分のみである。よって、プログラムのすべてのデータフローをアクセラレータにマッピングしたとき、コントロールフローの分岐点では接続部分を構成している部分のみ再構成すればよい。

2.1 Control-Flow Driven Data-Flow Switching

我々はこの観点より、データパスを可変部と固定部に分割し、実行時に可変部のみ再構成することを考えた。これにより、全体を再構成するのに比べ、可変部のみで済むために、動的再構成する部分を減らすことができる。また、固定部には、既存の命令列を配置し、命令間のデータ依存関係に応じて演算器を組み合わせて実行する。このように、CDDS アーキテクチャでは、動的再構成を必要最小限な範囲に限定し、可能な限り組み合わせ的に実行することで低消費電力化を狙う。一例として図 1(a) にプログラムの CDFG を示す。Basic Block(BB)1 から実行し、BB2 を実行し、分岐の結果により、再度 BB2 を実行するか BB3 を実行する。このとき、BB 内のデータフローには分岐がなく、動作中にデータの流れが変わることはない。一方、BB 間では分岐の結果によって、データの流れが変わる。この例の場合では分岐成立時には、図 1(b) に示すように BB2 から BB2 へのデータフローが必要となり、また、分岐不成立時には、図 1(c) に示すように BB2 から BB3 へのデータフローが必要になる。

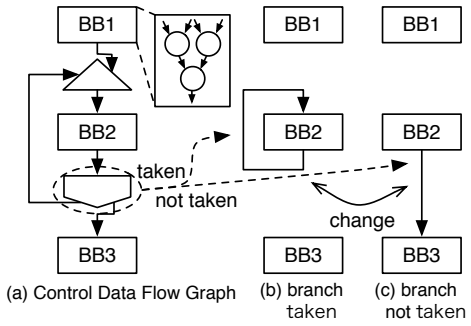

図 1　BB 間のデータフローのみ切り替え

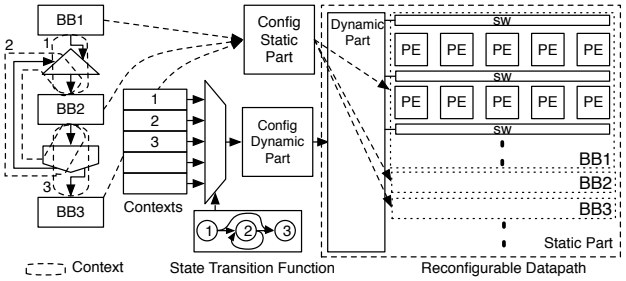

図 2　CDDS アーキテクチャ：ホットパスを固定部と可変部に分割

CDDS アーキテクチャは、図 2 に示すようにプログラムの BB 内のデータフローを固定部 (Static Part) 上に構成し、BB 間のデータフローを可変部 (Dynamic Part) 上に構成する。可変部はコンテキストに従い再構成され、固定部にマッピングされている BB にデータを供給する。例えばコンテキスト 1 の場合、可変部は BB1 から BB2 へデータを供給する。コンテキスト切り替えは、CDFG から生成された状態遷移関数に従って制御される。

命令列内部への分岐命令一つにつき、分岐命令の次の命令と分岐先の命令の二つの入り口があるため、コンテキストは 2 つ増える。プログラム外部からの入り口一つにつき、コンテキストは 1 つ増え、内部から外部への分岐命令もコンテキストは 1 つ増える。これらの総和が全コンテキストとなる。最大コンテキスト数を超えない限り、分岐命令が複数あっても実行するこ

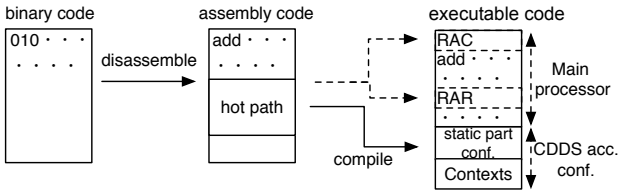

図 3　構成情報生成フロー

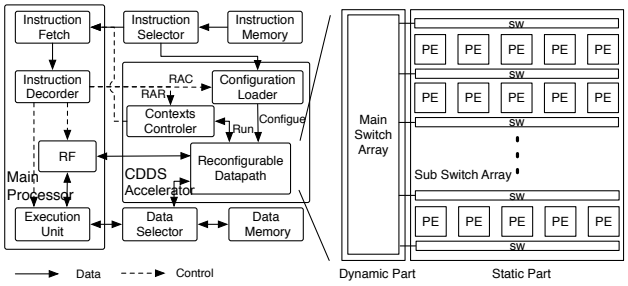

図 4　CDDS プロセッサのブロック図

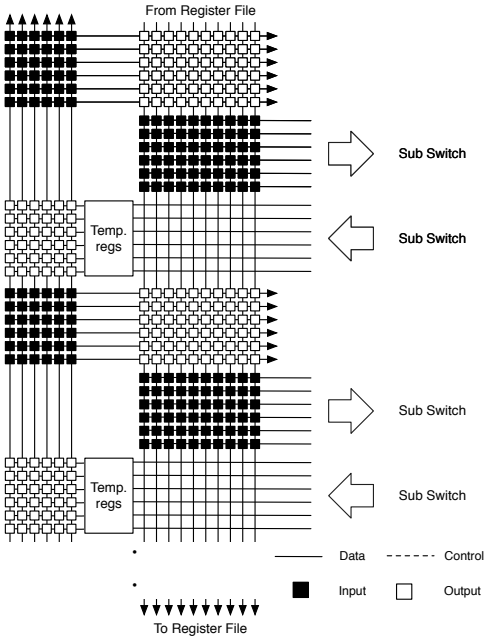

図 5　メインスイッチアレイの構造

とが可能である。ここで、従来からの手法である前方分岐命令を条件付き実行に変えることで、コンテキスト数を削減することができる。可変部には演算器はないため、コンテキストに演算情報は含まない。そのため、従来の DRP に比べて、コンテキストサイズを小さくすることができる。

2.2　構成情報の生成

CDDS アーキテクチャは、既存の組み込みプロセッサ内に組み込むため、既存コードとの移植性が必要である。今までの研究では構成情報を生成するためにプログラムを変更、または新たに記述する必要があった。CDDS アーキテクチャでは構成情報をバイナリファイルから生成することにより、過去のプログラム資産を引き継ぐことができる。また、バイナリファイルを使うことで、小規模な変換ツールで制御情報を生成可能という利点がある。

これを達成するために、図 2 に示す PE は、基本的に既存プロセッサの ALU を用いるように設計する。図 3 に構成情報生成フローを示す。始めに、コンパイラが生成したバイナリコードを逆アセンブルする。次に、そのアセンブリコードからホットパスを抽出し、その部分から構成情報を生成する。プログラムの先頭には、アクセラレータを構成する命令 (RAC) を置き、もとのホットパスの位置には呼び出し命令 (RAR) を置く。本体プロセッサは RAC をデコードすると、構成情報を読み込み、マッピングする。次に呼び出し命令をデコード次第、アクセラレータが実行をはじめる。プロセッサ起動時にアクセラレータを構成するために、実行時に構成のためのオーバーヘッドはない。

3.　CDDS プロセッサ

図 4 に提案する CDDS プロセッサの全体ブロック図を示す。本体プロセッサには既存のプロセッサを若干の拡張を加えて用いる。CDDS アクセラレータはデータパスを構成するサブスイッチアレイとメインスイッチアレイ、コンテキストを切り替えるコンテキストコントローラ、構成情報を読み込むコンフィギュレーションローダで構成する。メインスイッチアレイは、2.1 節の可変部、サブスイッチアレイは固定部に対応する。

本体プロセッサが RAC 命令をデコードするとコンフィギュレーションローダが命令メモリから構成情報を読込み、データパスを構成する。RAR 命令をデコードすると、コンテキストコントローラがコンテキストを切り替え、CDDS アクセラレータが実行を始める。CDDS アクセラレータではプログラム中のホットパスを実行し、その他は本体プロセッサで実行する。ホットパスまでの計算結果はレジスタファイル (RF)、データメモリに保存されており、RF とデータメモリからデータを読み書きすることが必要である。そのため、本体プロセッサの RF の一部を共有することで、必要なデータの読み書きを達成する。

3.1　メインスイッチアレイ

図 5 に示すメインスイッチアレイはサブスイッチアレイ上に構成された BB、RF、一時保存用のレジスタ間の接続を構成する。RF からのデータをサブスイッチアレイの BB を構成する部分に送り、その BB からの演算結果を次の BB に送る。マッピングしたプログラムの実行が完了し、本体プロセッサに制御を戻すときに演算結果をメインスイッチアレイを介して RF に格納する。実行中のコンテキストの演算結果を、遷移先のコンテキストが使用するときには、コンテキスト切り替えにより演算結果は失われるため、保存する必要がある。このとき、逐次、演算結果を RF に保存し続けるようにすると、データの大規模な移動が頻発し、消費電力、性能の点で問題がある。そのため、演算結果を一時的に保存するためのレジスタをメインスイッチアレイに配置している。

3.2　サブスイッチアレイ

サブスイッチアレイは図 6 に示すように、BB 内のデータフ

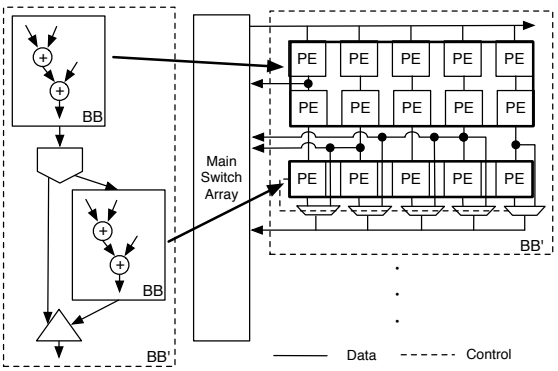

図 6　サブスイッチアレイへの BB のマッピング

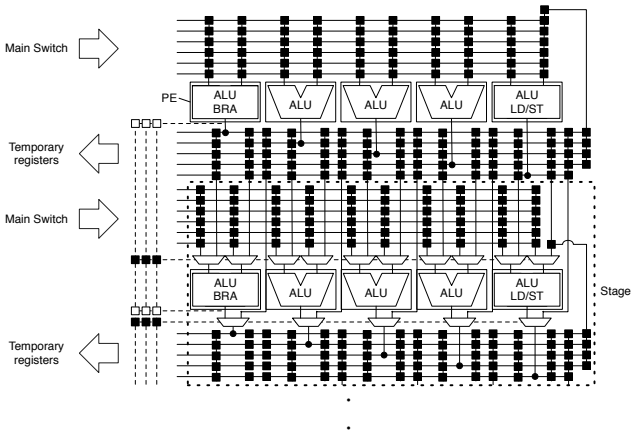

図 7　サブスイッチアレイの構造

ロー (図中左部) を実装するために、依存関係に応じて PE 間の接続を構成する。図 7 にサブスイッチアレイの構造を示す。サブスイッチアレイの 1 列をステージと呼び、ステージ内の PE 数は並列実行可能な命令の最大値となる。一般的な分岐を多く含むアプリケーションの命令レベル並列性は 5 程度 [4] であることが知られているため、本検討ではまずステージを 5PE とした。一方、依存関係のある命令は別ステージに配置するため、少なくとも、対象プログラム部分のクリティカルパス命令数分のステージ数が必要となる。この 1 ステージの PE 数とステージ数を掛け合わせたアレイサイズがマッピング可能な最大の命令数を規定する。すなわち、マッピングするホットパス部の最大規模を想定して CDDS アクセラレータのアレイサイズを設計する必要がある。

1 ステージを構成する PE は、平均的に、5 命令に 1 度程度以上は分岐 (BRA)、LD/ST 命令を実行すると見込まれる [4] ため、1 ステージ毎の 5 つの PE のうち、先頭 PE は分岐ユニットと ALU を含み、末尾 PE は LD/ST ユニットと ALU を併用するようにした (残りの PE は ALU のみ)。PE の出力を次ステージの PE が使うときには、その出力を次ステージの PE の入力にスイッチを使って接続する。これにより、組み合わせ的に命令を実行することができる。各 PE の入力はメインスイッチアレイからのデータか、前ステージの出力かを選択できる。例外として、LD/ST ユニットには同じステージからでも結果を送ることができるようにした。これは、ALU でメモリアドレスを生成して、その次に LD/ST を行うことが多いためである。また、一つのコンテキスト内に複数の LD/ST がある場合、これらをマッピングした LD/ST ユニットが競合しないように、メモリアクセス命令が配置された PE を順番に選択し、その PE のみがメモリにアクセスするようにした。メモリから読み込んだデータはコンテキストが切り替わるまで、外部レジスタに格納する。分岐ユニットでは条件付き実行やコンテキスト切り替えのための条件判定結果を生成する。

サブスイッチアレイには条件付き実行が可能な機構を取り入れる。一般的に知られているように、プログラムは短い分岐命令を複数含む。これらの分岐のためにコンテキストを使い、動的に切り替えるのは無駄がある。そのため、短い前方分岐を条件付き実行に変換し、サブスイッチアレイ上に構成して実行することで、コンテキスト数を削減する。条件付き実行を行うためにサブスイッチアレイの分岐ユニットの条件判定結果を下段に送れるように構成する。また、各段の PE の出力にセレクタを配置し、条件判定結果により、前ステージの出力か、その PE の出力かを選ぶようにした。

ここで、マッピングされた命令間の依存関係により縦列接続された一連の PE をチェーンと呼ぶことにする。コンテキスト毎のチェーンの最大長が、そのコンテキストの実行時間になるため、コンテキスト実行時間は可変となる。

3.3　コンテキスト、状態遷移コントローラ

コンテキストコントローラを図 8 に示す。本体プロセッサが構成命令を実行しだい、構成情報読み込み部 (図中 Configuration Loader) が、命令メモリから構成情報を読み出し、コンテキストメモリに格納する。次に、呼び出し命令をデコードすると、命令で指定する初期状態を STATE 部に格納する。すると、コンテキストが切り替わり、メインスイッチアレイのデータパスを構成する。データパスを構成するとサブスイッチアレイに RF のデータが流れ演算が始まる。コンテキストごとの実行時間はそれぞれのコンテキストにサイクル数として付随しており、そのサイクル数まで実行する。実行時間になると、図 9 に示す状態遷移コントローラ (STC) が、現状態 (STATE によって保持) と条件判定結果 (condition) を元に次の状態を決める。遷移先を決定する状態遷移関数はメモリ内に格納され、条件判定結果は全ステージの分岐ユニットの出力 (図中 condition) からコンテキスト (cond_sel) により指定する。コンテキストコントローラは本体プロセッサの戻り先アドレスを保持しており、終了状態の実行時間になると、このアドレスをプログラムカウンタに書き込む。

4.　CDDS アクセラレータの予備評価結果

本章では、テスト設計した小規模な CDDS アクセラレータとベースプロセッサの消費電力、エネルギーを見積もり、比較を行う。ベースプロセッサとして、命令拡張が可能な Lattice 社の Lattice Mico32(LM32) [5] を利用した。この評価では、CDDS アクセラレータはステージ数は 10 にし、コンテキスト数は 9 に

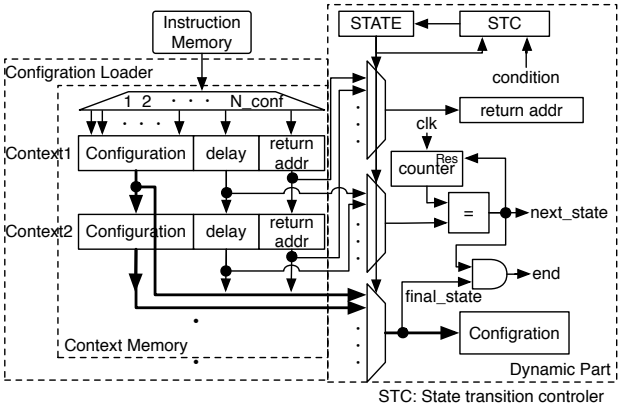

図 8　コンテキストコントローラ

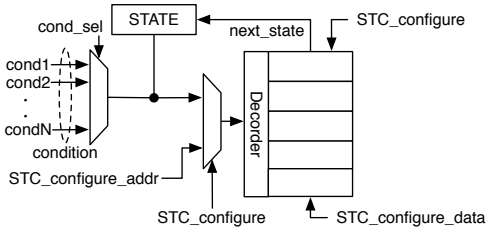

図 9　状態遷移コントローラ

設定している。3. 2 節で説明したように、これは、CDDS アクセラレータは、最大命令数は 50、最大クリティカルパスは 10、最大分岐数が 4 まで実行可能ということを示す。各ステージ毎に、レジスタは 5 つ用意し、メインスイッチからサブスイッチへ 4 つデータを流すように設定した。また、この評価では、条件付き実行に関しては実装はしていない。

LM32, CDDS アクセラレータともに RTL 設計し、TSMC 0.18um CMOS セルライブラリを用いて物理設計した。表 1 に論理合成、配置配線、配置配線後のシミュレーション、消費電力の見積もりに使用したツールを示す。論理合成時には、クロックゲーティングを挿入するように設定している。命令、データメモリはともに、一般的な組み込みプロセッサを想定して、32bit×4k ワード (16KB) の容量を持つものを使用した。このメモリサイズの単位周波数当たりの電力は、[6] より、0.475mW/MHz となり、シミュレーションから求めたメモリアクセスの平均周波数に掛けることでメモリの消費電力を求めた。

表 1　ツール、実装結果

論理合成		Synopsys Design Compiler
配置配線		Cadence Encounter
配置配線後シミュレーション		Mentor Graphics ModelSim
消費電力の見積もり		Synopsys Design Compiler
アーキテクチャ	ゲート数	クロック周波数 [MHz]
CDDS Accelerator	561,279	100
LM32	30,459	100

4.1　ベンチマークプログラム

評価アプリケーションとして sepia filter, crc32, AES で用いる sbox を用いた。LM32 の開発環境を使い、プログラムをコンパイルし、バイナリコードを生成する。このバイナリコードを逆アセンブリし、アセンブリコードから構成情報を手動で生成した。これらのアプリケーションは問題なく、全命令をマッピング、実行することが可能であった。

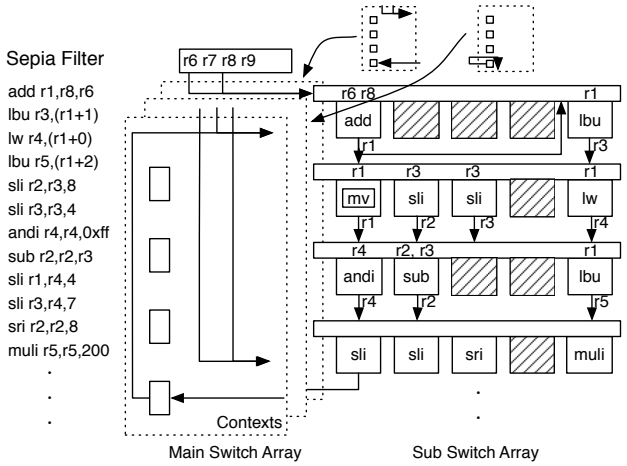

図 10　sepia filter 評価

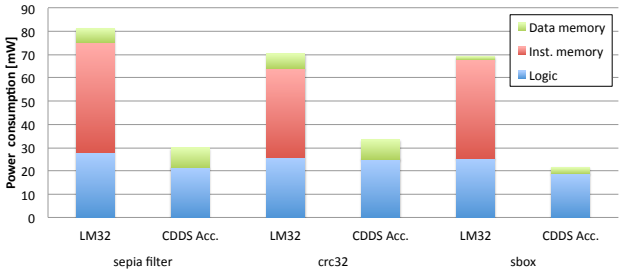

図 11　アプリケーションごとの電力評価

一例として、図 10 に sepia filter をマッピングした図を示す。命令列 (図中左) を、サブスイッチアレイ上の PE に割り当てることがわかる。sepia filter は一つの分岐命令を含むため、コンテキスト数は三つになる。図中の斜線部分は命令をマッピングすることができなかった PE である。表 2 にベンチマークの基本的な特性を示す。PE 使用率は、命令列をマッピングした結果より、プログラムの命令が割り当てられている PE 総数を全 PE 数で割ることで求め、36〜50%程度という結果になった。

表 2　アプリケーション毎のマッピング結果

アプリケーション	命令数	分岐命令数	PE 使用率	コンテキスト数
sepia filter	22	1	44	3
crc32	18	2	36	5
sbox	25	2	50	5

4.2　消費電力・消費エネルギー

図 11 にベンチマークを実行中の LM32 と CDDS アクセラレータの平均消費電力の見積もり結果を示す。CDDS アクセラレータは、LM32 に比べて、3〜5 倍ほど低消費電力化を達成している。これは、LM32 が命令メモリに毎クロックアクセスしているのに比べ、CDDS アクセラレータは実行時にアクセスし

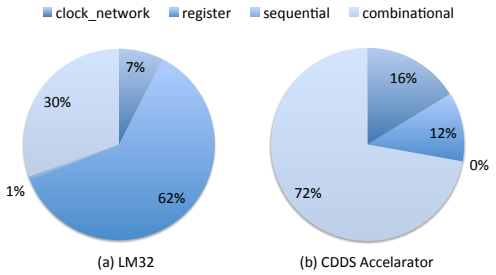

図 12　sepia filter 実行時の電力内訳

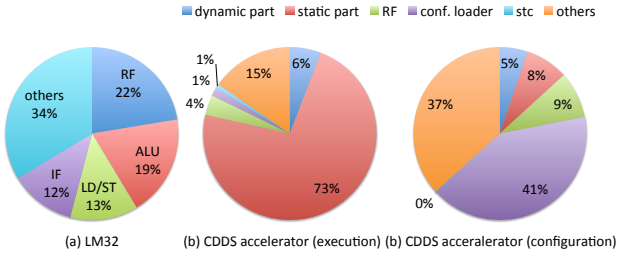

図 13　sepia filter 実行時の各モジュールの電力

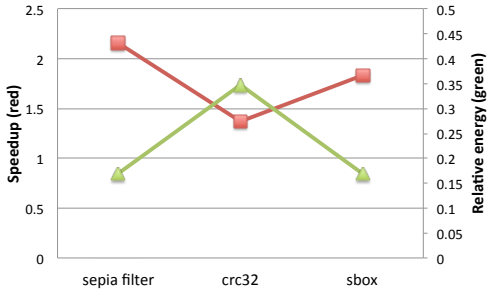

図 14　性能向上、相対エネルギー

ないことが最大の理由である。ロジックの消費電力の違いは相対的に小さい。

次に、図 12 に、sepia filter 実行時のメモリによる電力を除いた消費電力内訳を求めた結果を示す。LM32 は、組み合わせ実行よりもレジスタによる電力の方が約 2 倍の割合を占めていることがわかる。一方、CDDS アクセラレータは、組み合わせ実行の割合が全体の 3/4 を占めている。crc32 と sbox の消費電力内訳は、これとほぼ同じであった。

sepia filter 実行時の機能毎のロジックによる消費電力内訳を図 13(a, b) に示す。LM32 の消費電力内訳は、典型的な汎用プロセッサと同じような性質を示している。CDDS アクセラレータは、固定部の割合が 73%、可変部が 6%を占めている。図 12, 13 が示す LM32 と CDDS アクセラレータの違いにより、CDDS アーキテクチャのコンセプトであるレジスタを介さず可能な限り組み合わせ実行すること、が実現されていることがわかる。

一方で、CDDS アクセラレータの sepia filter 構成時の電力は、59.5mW(メモリによる電力が 47.5mW で、プロセッサの電力が約 12mW) であった。このように構成時の電力は大きいが、プロセッサ起動時の一度しか実行されないので問題ないと考えている。sepia filter 構成時のメモリ電力を除いた機能毎の電力内訳を図 13(c) に示す。構成に関わる部分 (conf. loader) の割合が 41%と大きいことがわかる。

4.3　性能・消費エネルギー

図 14 に LM32 を基準にとった時の CDDS アクセラレータの性能向上比 (赤線) を示す。アプリケーションによって、1.4〜2.2 倍ほどの向上を示している。crc32 で相対的に性能が低い理由は、プログラムにデータ依存が多く含まれるためである。

次に、消費エネルギーを消費電力に実行時間を掛けることで求め、LM32 との比較評価を行った。図 14 に示す LM32 を基準にした時の相対エネルギー (緑線) は、約 1/3〜1/6 倍となっており、CDDS アクセラレータは、ベースプロセッサと比較して性能電力比の向上を達成していることがわかる。

5.　ま　と　め

本稿では、低消費電力な組込みプロセッサのための、再構成可能なハードウェアアクセラレータのアイディアについて述べた。これは、動的再構成可能な範囲を限定することで、汎用性を保ったまま、性能電力比の向上を狙う。このアイディアを具体化するアーキテクチャの一つである Control-Flow Driven Data-Flow Switching (CDDS) 可変データパスアーキテクチャは、データパスを固定部と可変部とに分割し、実行時に可変部に限定して動的再構成することで、柔軟性と低消費電力性の両立を目指す。テスト設計した CDDS アクセラレータの予備評価では、1〜2 つの分岐を含む小さなプログラムではあるが、3 つのベンチマークで約 3〜6 倍の性能電力比の向上を示した。

本稿で提案したアーキテクチャは、様々な面で予備段階である。例えば、メインスイッチアレイ、サブスイッチアレイの構造、サイズを定量的な評価を元に決める必要があること、現実的なアプリケーションを使い、消費電力の詳細を求め、非効率な部分を調査すること、構成情報生成ツールがないこと、などの点である。今後は、これらを進める予定である。

謝辞　本研究の一部は科学研究費補助金 (基盤 B：24300012) の助成を受けたものです。ここに感謝の意を表します

文　　献

[1] Rehan Hameed, Wajahat Qadeer, Megan Wachs, Omid Azizi, Alex Solomatnikov, Benjamin C. Lee, Stephen Richardson, Christos Kozyrakis, and Mark Horowitz.: *Understanding sources of inefficiency in general-purpose chips*, *SIGARCH Comput. Archit. News*, 38(3):37–47, June 2010.

[2] S. Swanson and M.B. Taylor.: *Greendroid: Exploring the next evolution in smartphone application processors*, *Communications Magazine, IEEE*, 49(4):112 –119, april 2011.

[3] Masato Motomura.: *A dynamically reconfigurable processor architecture*, *Microprocessor Forum, Oct. 2002*, 2002.

[4] David W. Wall.: *Limits of instruction-level parallelism*, *SIGOPS Oper. Syst. Rev.*, 25(Special Issue):176–188, April. 1991.

[5] LatticeMico32 development tools, `http://www.latticesemi.co.jp/products/designsoftware/micodevelopmenttools/index.cfm`.

[6] 3.1 On-Chip SRAM `http://www.csd.uoc.gr/~hy534/03a/s31_ram_bl.htm`.