一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
CPSY2019-22,DC2019-22(2019-07)

粗粒度再構成可能アーキテクチャCMAにおける メモリバンクアクセスの改良

小島　拓也†　　天野　英晴†

† 慶應義塾大学大学院 理工学研究科　223–8522 神奈川県横浜市港北区日吉 3–14–1
E-mail: †{tkojima,hunga}@am.ics.keio.ac.jp

あらまし　エネルギー効率に優れる粗粒度再構成可能アーキテクチャ CGRA(Coarse-grained reconfigurable architecture) は IoT(Internet of Things) デバイスやエッジコンピューティングへの応用が期待されている。CMA (Cool Mega Array) はクロックサイクル単位ではなくタスク単位での再構成を行う CGRA であり、省エネルギー性に特化している。しかし、CMA アーキテクチャには電力削減のためにデータ制御においていくつかの制約がかけられている。本稿では、これらの制約を解消しエネルギー効率を改善した新たなアーキテクチャ *VPCMA2* を提案する。ハードウェアのオーバーヘッドを評価するために、このアーキテクチャの設計を 65-nm プロセスで実装した。評価の結果、追加した機能は 17%の電力オーバーヘッドと 10%の面積オーバーヘッドをもたらしたが、およそ 46%の性能改善を得ることができた。

キーワード　CGRA, 粗粒度再構成可能アーキテクチャ, バンクアクセス, メモリアーキテクチャ

Takuya KOJIMA† and Hideharu AMANO†

† Graduate School of Science and Technology, Keio University　Hiyoshi 3–14–1, Kohoku-ku, Yokohama, Kanagawa, 223–8522 Japan
E-mail: †{tkojima,hunga}@am.ics.keio.ac.jp

1. はじめに

近年、IoT (Internet of Things) やエッジコンピューティングは大きな注目を集め、高いエネルギー効率で柔軟性の高いデバイスへの関心が高まっている。粗粒度再構成可能アーキテクチャ CGRA(Coarse-grained reconfigurable architecture) はこれらの要求を満たすデバイスの一種である。FPGA(Field-Programmable Gate Array) がビット単位の再構成粒度であるのに対し、CGRA はワード単位の再構成粒度を提供する。これによって、再構成のためのハードウェアオーバーヘッドを削減しつつ、プログラマビリティを維持したハードウェアプラットフォームを実現する。

CMA(Cool Mega Array) は低電力志向の CGRA として提案され [1]、他の CGRA と同様に、図 1 のような PE(Processing Element) アレイを計算資源として持つ。この PE アレイはタスク単位に再構成され、あるアプリケーションカーネルを実行している間、構成 (コンフィギュレーション) は変化しない。これによって、再構成に必要なダイナミック電力を劇的に削減できる一方で、サイクル単位の再構成をサポートする他の CGRA と比べ、柔軟性が低下する。この欠点を補うために、CMA にはマイクロコントローラと呼ばれる専用の小規模な RISC プロセッサを備える。マイクロコントローラは命令コードに従い、主に PE アレイとデータメモリの間のデータ転送を制御する。

一般に、PE アレイはクロスバスイッチなどを用いてマルチバンク化されたメモリと接続し、メモリアクセス競合によるストールを避ける。これまで提案されている CMA [2] [3] はデータマニピュレータと呼ばれるメモリアクセス機構を備える。データマニピュレータは N 個のマルチプレクサで構成された N 入力-N 出力のネットワークである。入出力数 N はメモリバンクの数と一致し、ある入力データは転送テーブルに従い、任意の出力へ転送することができる。これによって、PE アレイとデータメモリ間で柔軟なデータ転送が可能になる。

しかし、これまでの CMA では各バンクでアクセスされるデータはアドレス空間上で連続している必要があった。たとえ独立したメモリバンクにあるデータであっても、それらが連続したアドレス空間にない場合同時にアクセスすることはできない。[2] [3] で評価用いられているようなシンプルなアプリケーションではこのような制約は問題にならなかったが、より複雑なデータアクセスを含むアプリケーションを想定すると、こうした制約は非効率的なデータアクセスを招く。

本稿では、前述のメモリアクセス制約を取り除いたバンクアクセス機構を提案する。加えて、PE アレイにおける利用可能な定数レジスタの制約を緩和し、複雑なアプリケーションへのマッピング能力を高める。一方で、こうした機能追加はチップ面積、電力の増加を引き起こす。そこで、高機能なデータ制御と電力、面積オーバーヘッドの間のトレードオフをいくつかのアプリケーションを用いて分析する。

2. 研究の背景

CGRA は一般に小規模な演算器 (PE) を 2 次元のアレイ状に

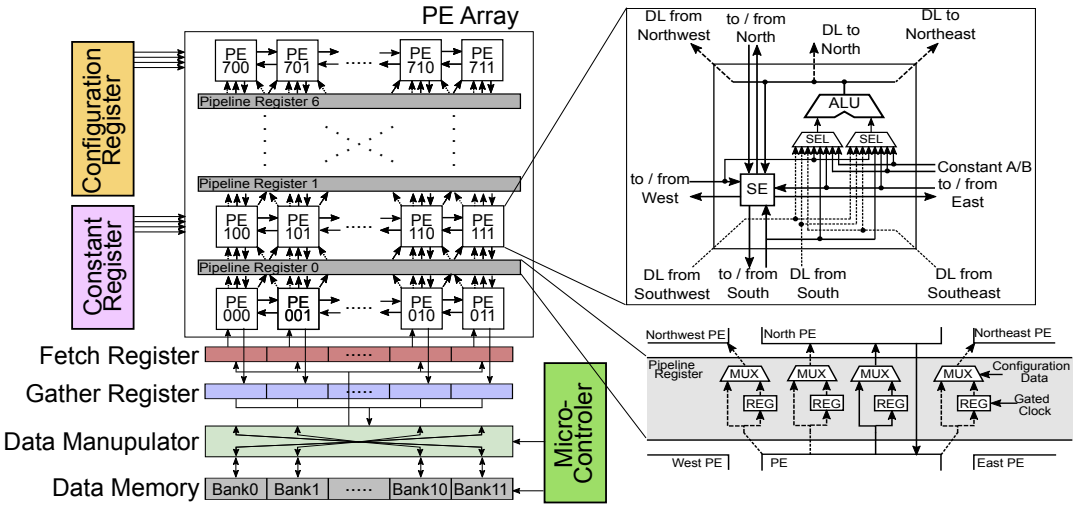
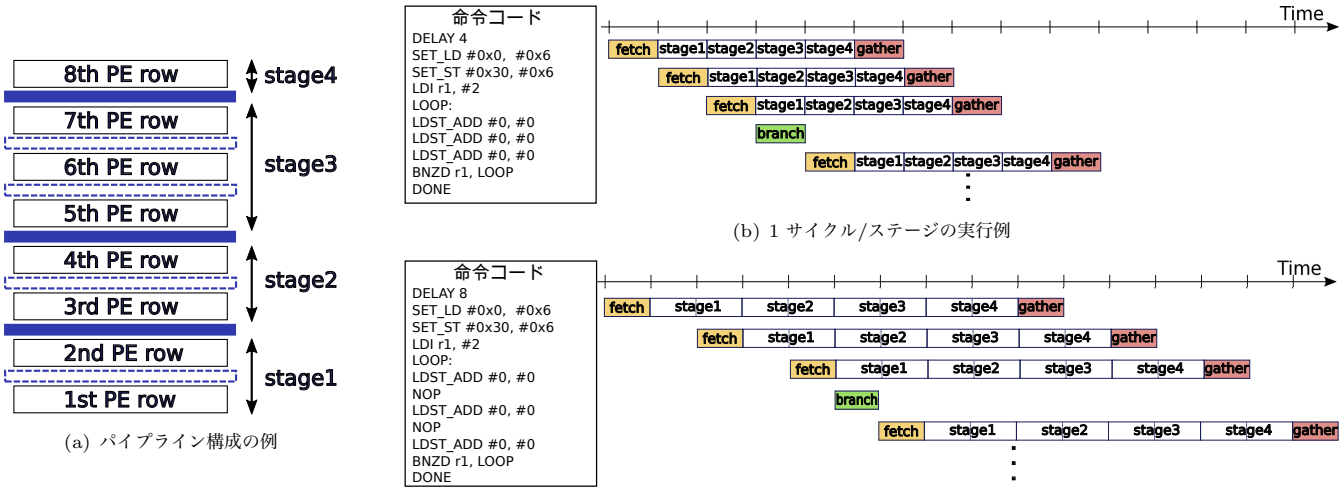


図 1　VPCMA のアーキテクチャ



(a) パイプライン構成の例

(b) 1 サイクル/ステージの実行例

(c) 2 サイクル/ステージの実行例

図 2　PE アレイにおけるパイプライン処理の例

並べた構造を持ち、この豊富な演算資源を用いてアプリケーションのループ処理などを効率的に処理するアクセラレータとして利用される。CGRA の 1 種である CMA アーキテクチャ [1] は再構成に必要な電力を削減するために、クロックサイクル単位での再構成を行わずにひとつのタスクが完了するまで同じコンフィギュレーションを利用し続ける。さらに、PE で計算された結果をそのまま隣接する PE へ転送することで、中間データを保持するためのローカルレジスタを不要にし、劇的にダイナミック電力を削減している。ローカルレジスタを除去したことにより巨大な組合わせ回路となった PE アレイは長いクリティカルパスを形成してしまうが、PE アレイをマルチサイクル実行の演算器にすることでシステムのクロック周波数低下を防いでいる。この実行サイクル数はアプリケーションに応じてプログラムすることが可能である。ところが、マルチサイクル実行にしてもスループットの低下を避けることはできず性能のボトルネックとなっていた。

2.1　VPCMA アーキテクチャ

VPCMA (Variable Pipelined CMA) は前述のボトルネックを解消するためにパイプライン化された PE アレイを搭載したアーキテクチャである [3]。Fig. 1 に示すように従来の CMA とは異なり、限られた数のレジスタを用いて PE アレイをパイプライン化している。PE アレイのサイズが 8×12 であり、各 PE 行の間に計 7 本のパイプラインレジスタが挿入されている。PE アレイの他にマイクロコントローラ、データマニピュレータ、12 バンクに分割されたデータメモリを備える。また、図 1 は PE の構成も示している。各 PE は ALU(Arithmetic Logic Unit)、入力用のセレクタ、SE(Switching Element) を持ち、隣接する PE 同士は相互に接続されている。相互接続には図中の実線で表される SE によるリンクと、図中の点線で表されるダイレクトリンクがある。ダイレクトリンクは北方向、北西方向、北東方向の 3 方向に延びている。

定数レジスタは 1 行の PE(12 個) で共有されており、1 行あたり 2 つのレジスタを利用できる。もし 1 行の PE で 2 つ以上の定数値が必要な場合は、他の行で未使用の定数レジスタを相互接続網を通して利用する。

各パイプラインレジスタは図 1 に示すようにコンフィギュレーションデータによって選択的に利用される。アクティブなパイプラインレジスタは通常のレジスタと同様の振る舞いをするが、そうでないパイプラインレジスタはクロックゲーティングが施され、入力されたデータはそのまま上段の PE へ転送される。これによって、可変的なパイプライン構造をとることができ、アプリケーションに応じてオーバーヘッドを最小化したパイプラインを実現できる。ただし、南方向へのデータリンクは最終的な演算結果を返すためだけに利用されるため、この

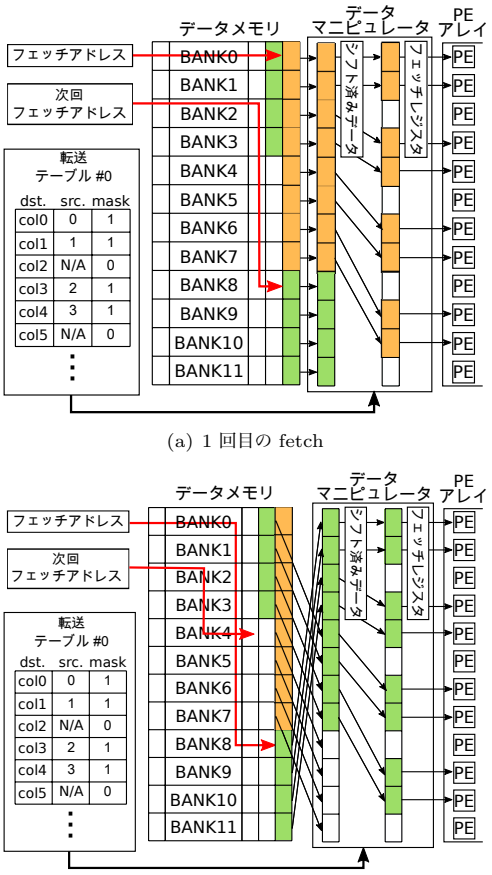
データパスはパイプラインレジスタによってラッチされることはない。

マイクロコントローラは 16bit 長命令の小規模な RISC プロセッサで、命令コードに従い PE アレイとバンクメモリとの間のデータ転送を制御する。フェッチレジスタとギャザーレジスタはそれぞれ PE アレイの入力、出力に接続されていて、マイクロコントローラがフェッチレジスタにデータを書き込むことで、PE アレイでの処理が開始する。数クロックサイクル後、演算結果はギャザーレジスタに書き込まれ、マイクロコントローラはこれをデータメモリに書き戻す。本項ではフェッチ命令で前者の処理を、ギャザー命令で後者の処理を行う。

図 2 に VPCMA による演算実行の様子を示す。ここでは、2 つの例—1) ステージあたり 1 サイクル (Fig.2(b)) と 2) ステージあたり 2 サイクル (Fig. 2(c))—を示す。どちらの例でも、図 2(a) のように 4 ステージのパイプラインが用いられていることとする。DELAY 命令は PE アレイにデータを入力し、結果が出力されるまでの遅延時間を指定する専用命令である。各ステージを 1 サイクルで実行する場合、4 サイクルの遅延時間を指定する。一方、各ステージを 2 サイクルで実行する場合、8 サイクルの遅延時間を指定する。SET_LD 命令と SET_ST 命令はフェッチ命令およびギャザー命令の設定を行う命令で、2 つの引数はそれぞれ、アクセス開始アドレスとアドレスのインクリメント量を指定する。フェッチ命令とギャザー命令は一つの複合命令 LDST_ADD 命令にまとめられている。LDST_ADD 命令が発行されると即座にフェッチ命令が実行されるが、ギャザー命令の実行は図 2(b) に示すように前述の DELAY 命令で指定したサイクル数だけ遅延する。LDST_ADD 命令はデータマニピュレータにおける転送テーブル番号を引数とする。この転送テーブルはフェッチ、ギャザー用で独立に指定することができる。転送テーブルの詳細は 3.1 節で述べる。フェッチ命令、ギャザー命令は実行が完了すると読み出し/書き込みアドレスを自動的にインクリメントする。各ステージを 2 サイクルで実行する場合 NOP 命令 (もしくは依存しない別の命令) を LDST_ADD の後に挿入する必要がある。マイクロコントローラは BNZD(Branch Not equal to Zero with Decrement) 命令をはじめとするいくつかの分岐命令をサポートする。どちらの例においても BNZD 命令がレジスタ *r0* の値をチェックし、0 でなければ分岐する。また、BNZD 命令は分岐が成立すると比較に利用したレジスタの値を 1 デクリメントする。分岐命令は 1 サイクルを要するため各ステージを 1 サイクルで実行する場合、次のイテレーションの LDST_ADD 命令実行が 1 サイクル遅れる。対して、各ステージを 2 サイクルで実行する場合、この遅延は NOP 命令挿入の代わりとなる。他にもマイクロコントローラはレジスタ間で汎用的な算術演算、論理演算が可能である。最後に DONE 命令が実行されると、タスク完了がホストプロセッサに通知される。

2.2 データマニピュレータ

データマニピュレータはデータメモリとフェッチ/ギャザーレジスタの間で柔軟なデータ転送を可能にする。図 3 はフェッチ命令実行時のデータマニピュレータの動作例である。この例では、SET_LD 命令によってフェッチ開始アドレスは 0x0、自動インクリメント量は 8 と初期化されているものとする。フェッチ命令が実行されると、フェッチ開始アドレスから連続した 12 個のデータを 12 のメモリバンクから読み出す。各バンクから読み出されたデータは、フェッチ開始アドレスの指すデータが先頭になるようにシフトされる。1 回目のフェッチ (図 3(a)) ではすでに先頭のデータがフェッチ開始アドレスにあるデータであるためシフトは発生しない。対して 2 回目のフェッチ (図 3(b)) では先頭のデータがバンク 8 にあるため、シフトする必要がある。データマニピュレータは 12 個のマルチプレクサで構成されているため、12 個の入力データは転送テーブルに応じて任意の出力ポートへ転送することができる。転送テーブルは各出力ポートのマスク情報も含んでおり、図 3 における 3 番目のポートのようにどのデータも転送させないこともできる。VPCMA はフェッチ命令、ギャザー命令用でそれぞれ 16 個のテーブルを持っている。これによって、ある一つのタスクが異なるメモリアクセスパターンを持っていても、マイクロコントローラはテーブル番号を変えるだけでよく、再構成が不要となる。



(a) 1 回目の fetch

(b) 2 回目の fetch

図 3　データマニピュレータによるデータ転送

3. 提案手法

VPCMA アーキテクチャはデータ制御において次のような制約を持っている。本稿では、これらの制約を緩和した新しいアーキテクチャ VPCMA2 を提案する。

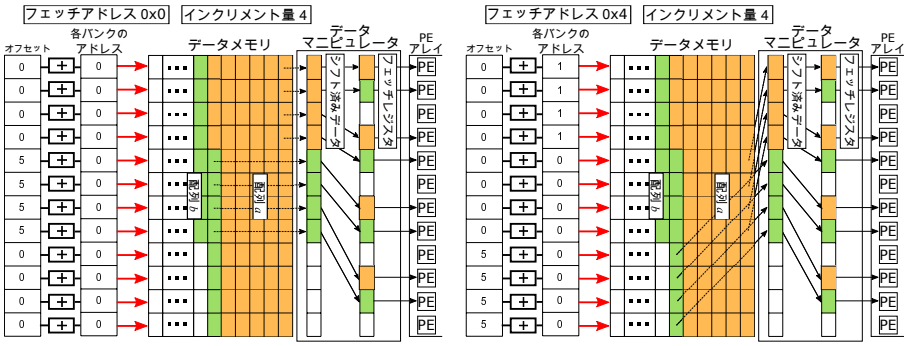
- 連続した 12 個のデータのみをインタリーブアクセス可能
- PE1 行あたり利用可能な定数レジスタは 2 つ
- マイクロコントローラ内の汎用レジスタにデータメモリなどのデータをロードできない

```
for i ← 0 to 11 do
  fetch_addr_i ← fetch_addr div 12
  if i < fetch_addr mod 12 then
    fetch_addr_i ← fetch_addr_i + 1
  end if
end for
```

(a) 各バンクのアドレス計算

```
for i ← 0 to N = 64 do
  c[i] ← a[i] × b[i mod 16]
end for
```

(b) ループのサンプルコード



(c) 1 回目のフェッチ　　(d) 2 回目のフェッチ

図 4　インタリーブアクセスの改良

3.1　データマニピュレータの改良

先述のバンクメモリへのアクセス制約を解消するために、VPCMA2 ではデータマニピュレータが改良されている。図 4(b) のようなループコードを想定すると、以前のデータマニピュレータではこのアクセスパターンをそのまま取り扱うことができない。同時にアクセスされる配列 a と配列 b の要素は最大で 64 ワードの距離にあるが、以前のデータマニピュレータは連続した 12 個のデータしかアクセスすうことができないためである。そのため、VPCMA でこのループを取り扱うには配列 a と配列 b の要素を $a[0], b[0], a[1], b[1], \ldots$ のように交互に織り交ぜてデータメモリに格納する必要がある。しかし、このようなメモリ配置の変更はホストプロセッサ側で行う必要があり、データ転送時間を悪化させる。加えて、配列 b は $a[16:31]$、$a[32:47]$、$a[48:63]$ との計算のためにコピーして転送する必要があり、データ転送時間を増加させ、メモリ容量を浪費してしまう。

そこで、VPCMA2 のデータマニピュレータでは転送テーブルに図 4(c) のようなバンク毎のアドレスオフセットを追加している。図 4(c) における他の転送テーブルの内容は図 3(a) のものと同一であるため省略している。図 4(c) と図 4(d) は改良されたインタリーブアクセスの様子を示している。この例では SIMD プロセッサのようにループのイテレーション 4 回分が同時に PE アレイで実行される。まず、各バンクでアクセスされるアドレスは図 4(a) に示した方法で計算される。このアドレス計算自体はこれまでのデータマニピュレータと同様であるが、本手法ではこれらの計算されたアドレスにオフセットを加えた値が実際にアクセスされる。1 度目のフェッチでは、すべてのバンクのアドレスは 0 と計算されるが、5~8 番目のバンクに設定されたオフセットは 5 であるため、$a[0:3]$ と $b[0:3]$ が同時にロードされる。2 度目のフェッチではフェッチ開始アドレスが 4 だけインクリメントされるため、各バンクのアドレスが再計算される。前節で説明したシフトと同様に、アドレスオフセットも合わせてシフトされる。したがって、同じ転送テーブル (およびオフセット) を利用して $a[4:7]$ と $b[4:7]$ をロードすることができる。この例では 4 度目のフェッチ ($a[12:15]$ と $b[12:15]$) から 5 度目のフェッチ ($a[16:19]$ と $b[0:3]$) へ移る際に配列 b へのオフセットを変更する必要がある。また、バンク競合が起きないようにメモリ配置を十分に最適化しておく必要がある。

3.2　定数レジスタの制約緩和

前節で述べたように VPCMA は 16 個の定数レジスタを持っているが、各 PE はそれらを自由に利用することができない。1 行の PE で 2 つ以上の定数を必要とする場合、相互接続網を通して別の PE 行で余っている定数レジスタを利用する必要があるが、配線資源を消費してしまうため、アプリケーションのマッピングに失敗する可能性が高まる。

この制約を解消するために、VPCMA2 では定数レジスタの数は 16 のまま、すべての PE から 16 個の定数値を利用可能にした。代わりに、SE による定数値を転送する機能を廃止した。ただし、PE で指定可能なレジスタ数が増えたことによりコンフィギュレーションデータの増加を招く。

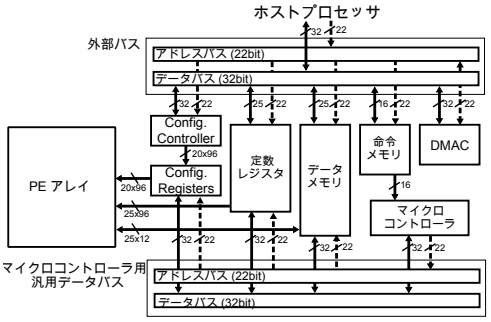


図 5　VPCMA2 の拡張されたデータバス

3.3　マイクロコントローラの命令セット拡張

VPCMA はホストプロセッサとデータを交換するための外部バスのみを持っており、これを利用してホストプロセッサは PE の再構成、命令コード、処理用データ、定数データ、計算結果のすべてのデータ転送を行う。つまり、どれほど少量のデータ変更であっても、ホストプロセッサによる制御を必要としていた。また、マイクロコントローラは汎用レジスタにデータメモリの値をロードすることはできず、ループ回数が入力データに依存するようなアプリケーションなどを効率的に処理することができなかった。

この問題に対処するために、VPCMA2 ではマイクロコントローラに図 5 に示す汎用のデータバスを追加し、これを通して、データメモリや命令メモリ含む各モジュールへアクセスできるよう命令セットが拡張されている。しかし、マイクロコントローラは 16bit 幅のレジスタファイルであるのに対し、ほとんどのモジュールは 25bit や 32bit などデータ幅である。したがって、16bit 以上のデータをロードする際は 2 つのレジスタを結合して利用される。また、32bit 幅の演算を行う ADD.D 命令なども合わせて導入されている。追加したデータバスによって、再構成するパートが小さい場合、ホストプロセッサなしに

表 1　実装環境と消費電力の比較

	VPCMA [4]	VPCMA2 (本手法)
設計	Verilog HDL	
プロセス ライブラリ	Renesas SOTB 65 nm LP (Low Power)	LSTP (Low STandby Power)
標準電圧	0.55 V	0.75 V
論理合成	Synopsys Design Compiler 2016.03-SP4	2017.09-SP1
配置配線	Synopsys IC Compiler 2016.03-SP4	N/A
	消費電力 (30MHz)	
リーク電力	126.0 μW	25.18 μW
ダイナミック 電力	3.337 mW	4.029 mW
合計	3.463 mW	4.053 mW

表 2　オーバーヘッドの比較

	VPCMA	VPCMA2 (1 cycle f/g)	VPCMA2 (2 cycle f/g)
周波数 (MHz)	87.71	95.23	125.0
75%スケール	N/A	71.42	93.75
面積 (mm^2) (PE アレイを除く)	10.04	14.55	14.22

マイクロコントローラがこれを行うこともできる。マイクロコントローラが動作中はバス獲得権はホストプロセッサよりもマイクロコントローラが優先的に獲得できる。

4. 評　　価

4.1 VPCMA2 の実装

データ制御の制約を緩和するのに生じたハードウェアオーバーヘッドを評価するために、提案した VPCMA2 を Verilog HDL で設計し、Renesas SOTB 65-nm プロセスで合成した。VPCMA に関しては、同様に SOTB プロセスで実チップ化およびその評価が行われている [4]。実装環境を表 1 に示す。利用した SOTB プロセスには LP(Low Power) と LSTP(Low STanby Power) の 2 種類があり、LSTP は LP と比べ待機電力 (省リーク電力性) に優れる。一方で、LSTP は LP よりも標準電圧が高いため、動作時のダイナミック電力も大きい。予備評価によれば、LSTP では LP よりも 25%高いクロック周波数でもタイミング制約を満たすことができることがわかっている。VPCMA は LP プロセスで実装されているが、現在利用可能できないため、VPCMA2 の実装には LSTP プロセスが用いられている。

4.2 ハードウェアオーバーヘッド

4.2.1 最大動作周波数

VPCMA2 では VPCMA と比べデータマニピュレータにおける機能が追加されており、これは動作周波数低下を招き得る。そこで、VPCMA2 の 2 種類の設計—1) 1 サイクル実行のフェッチ/ギャザー命令、2) 2 サイクル実行のフェッチ/ギャザー命令— を用意した。表 2 はそれぞれの合成結果による最大動作周波数を示している。また、LP プロセスで実装された VPCMA と比較するために、75%にスケールした値も示している。1 サイクル版のものと比べ、2 サイクル版の方がより高い周波数で動作可能であることがわかる。さらに、2 サイクル版の設計は VPCMA と比較して、約 6%高い周波数で動作可能である。これは、VPCMA のデータマニピュレータがクリティカルパスを含んでいたが、VPCMA2 ではデータマニピュレータが 2 サイクルかけてフェッチ/ギャザーを行い、クリティカルパスが別の箇所になったためである。したがって、1 サイクル版の VPCMA2 では VPCMA と 18%比べ動作周波数が悪化している。当然 VPCMA の設計においても 2 サイクル実行のフェッチ/ギャザーに変更すれば同程度の動作周波数向上を得ることはできる。

4.2.2 面　　積

次に 50MHz で動作するようにタイミング制約を与え、合成したネットリストのセル面積を分析する。この制約条件は VPCMA の実チップと同様のものである。表 2 に PE アレイを除く部分のセル面積をまとめる。VPCMA と比較して VPCMA2 では 40%面積が増加している。しかし、実際には VPCMA と VPCMA2 の両方の設計において PE アレイが全体の面積の大きな割合を占めており、2 サイクル版の VPCMA2 の場合、全体の面積は $50.28mm^2$ である。したがって、全体の面積で比較すると面積オーバーヘッドは 10%以下である。2 サイクル版の VPCMA2 は 1 サイクル版と比較してやや小さな面積になっている。動作周波数の結果も考慮すると、VPCMA2 の設計では 2 サイクル版を採用するべきである。フェッチ命令、ギャザー命令が実行されてからメモリバンクへアクセスするのが 1 サイクル遅れることになるが、PE アレイ自体がマルチサイクル実行であるため大きな問題にはならない。以降は 2 サイクル版の設計のみを用いる。

4.2.3 電力オーバーヘッド

表 1 は VPCMA と VPCMA2 の電力比較も示している。VPCMA の消費電力は実チップ測定の結果 [4] であり、VPCMA2 は Synopsys PrimeTime によってシミュレーションされた結果である。電力シミュレーションに用いたスイッチング率は Cadence NC-Verilog によるゲートレベルシミュレーションより得た。実チップ測定と同じ条件でシミュレーションを行うため、クロック周波数は 30MHz に設定し、同一のアプリケーション (カラー画像のグレースケール) を動作させた。VPCMA2 のリーク電力は先述のプロセス特性の違いにより VPCMA の約 5 分の 1 となっている。しかしながら、どちらの場合もダイナミック電力が支配的であるため、VPCMA2 は VPCMA と比べ 17%電力が増加している。ただし、これは標準電圧が VPCMA が 0.55V であるのに対し、VPCMA2 では 0.75V であることが大きな原因であり、機能追加に起因する電力増加はこれよりも小さい。

4.3 アプリケーション実行の改善

最後に、VPCMA2 で改善されたデータ制御がアプリケーション実行時間に与える効果の評価を行う。本評価では JPEG エンコードに含まれる 3 つのタスク: 1) RGB-YCC 変換, 2) 離散コサイン変換 (DCT), 3) 量子化 (Quantize)、加えて、2 次元の Jacobi 法を VPCMA,VPCMA2 で実行した場合の実行時間を比較する。

図 6 に VPCMA の実行時間で正規化した結果を示す。これらの実行時間は計算時間 (Comp) 以外にホストプロセッサから各種データを転送する時間 (Data) や再構成にかかる時間 (Conf)、

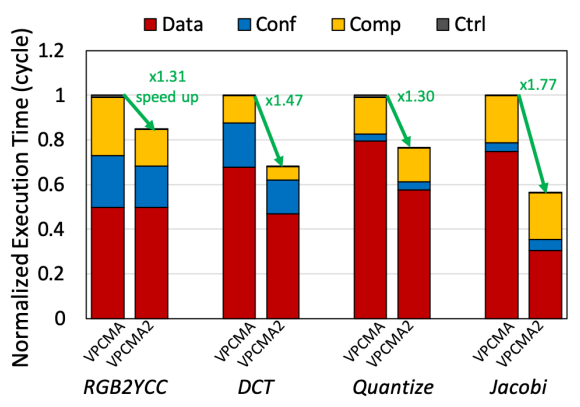


図 6　各アプリケーションの実行時間比較

その他の制御 (Ctrl) の時間を含んでいる。ホストプロセッサからのデータ転送にはシングル転送と 16 ワードのブロック転送の 2 種類を想定し、シングル転送ではアドレスとデータに 2 サイクル、ブロック転送ではアドレスと 16 ワードのデータに 17 サイクルを要する。

4.3.1　マッピング可能性の向上による効果

定数レジスタの接続数が増加したことにより、より大きなアプリケーションカーネルを PE アレイにマップすることができるようになった。これによって、*RGB2YCC, DCT* で計算時間が削減されている。VPCMA の場合、これらのカーネルは大きすぎて一度に PE アレイにマップすることができない。したがって、いくつかのサブグラフに分割して実行する必要がある。

RGB2YCC の場合、VPCMA では RGB 画素値からそれぞれ Y,Cr,Cb 値を計算する 3 つのサブグラフに分割して実行している。そのため、VPCMA では 3 回の再構成を行なっているが、再構成回数が 1 回の VPCMA2 と比較してそれほど再構成時間が大幅に増加しているわけではない。これはグラフサイズが小さくなったため、1 回の再構成にかかる時間が短くなっているからである。RGB-to-Y を計算するカーネルは PE を 2 列消費するが、VPCMA2 で用いられる RGB-to-YCC を同時に計算するカーネルは PE を 4 列消費している。PE アレイは 12 列あるため、同一のマッピングは横方向に複製され、並列に実行される。RGB-to-Y の場合、6 並列で計算することが可能である。対して、VPCMA2 は RGB-to-YCC を 3 並列で計算する。マッピングの複製はコンフィギュレーションデータのマルチキャスティングによって効率的に行われる [5]。これらの理由から VPCMA2 が VPCMA と比べ 3 倍高速に計算が行なえるようになったわけではないが、全体として 1.31 倍の高速化を達成している。

4.3.2　バンクアクセス改良による効果

データマニピュレータの改良によって、*DCT, Quantize, Jacobi* でデータ転送時間を削減することができている。前述のように、VPCMA では *DCT* を 2 つに分割して実行しているが、前半の計算で得られた結果を中間データとして保持し、後半の計算で利用する必要がある。しかし、VPCMA では 12 ワード以上離れたデータを同時にアクセスすることはできない。したがって、入力データ 8 ワードごとに 8 ワードの中間データ用のスペースを残しながらデータ転送を行う必要があり、ブロック転送の効率が低下している。よって、*DCT* では計算時間の短縮の加え、データ転送時間の削減によって全体で 1.47 倍の高速化を達成した。

Quantize では入力データを量子化テーブルで量子化していくため、ループで 2 種類の配列にアクセスする必要がある。3. 節の図 4 に示したように VPCMA2 ではこれを効率的に行うことができるが、VPCMA の場合 2 つの配列の要素を交互に並べて転送する必要があるためデータ転送時間が長くなっている。2 次元配列にアクセスする *Jacobi* でも同様の理由で大幅にデータ転送時間を削減できている。しかし、これらのアプリケーションでは PE アレイで実行されるカーネルは同一であるため計算時間の短縮は得られていない。結果的にそれぞれ、1.30 倍、1.77 倍の高速化を達成している。また、データマニピュレータにおけるオフセットの更新などは変更箇所が十分に小さいため、拡張したデータバスを通してマイクロコントローラ自らが変更を行なっている。

5.　結　　論

本稿では、データ制御においていくつかの制約を持っていた VPCMA アーキテクチャを改良した VPCMA2 を提案した。複雑なメモリアクセスパターンを含むアプリケーションでは VPCMA の非効率的なデータ転送が原因で、達成可能な性能に限界が生じていたが、VPCMA2 ではこれらを解消し、平均 1.46%の性能向上を達成した。機能追加によって面積が約 10%、電力が約 17%増加したが、本稿の評価に利用したアプリケーションではエネルギー効率を向上させることができた。

謝　　辞

本研究は、JSPS 科研費 (B) ビルディングブロック型計算システムにおけるチップブリッジを用いた積層方式 (18H03125) および科研費 3 次元積層技術を応用した粗粒度再構成可能デバイスの研究 (19J21493) の助成を受けたものである。また、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、シノプシス株式会社の協力で行われたものです。関係者の皆様に感謝致します。

文　　献

[1] N. Ozaki, Y. Yasuda, M. Izawa, Y. Saito, D. Ikebuchi, H. Amano, H. Nakamura, K. Usami, M. Namiki and M. Kondo: “Cool Mega-Arrays: Ultralow-Power Reconfigurable Accelerator Chips”, IEEE Micro, **31**, 6, pp. 6–18 (2011).

[2] K. Masuyama, Y. Fujita, H. Okuhara and H. Amano: “A 297mops/0.4mw ultra low power coarse-grained reconfigurable accelerator CMA-SOTB-2”, 2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig), pp. 1–6 (2015).

[3] N. Ando, K. Masuyama, H. Okuhara and H. Amano: “Variable Pipeline Structure for Coarse Grained Reconfigurable Array CMA”, 2016 International Conference on Field-Programmable Technology, pp. 231–238 (2016).

[4] T. Kojima, N. Ando, Y. Matshushita, H. Okuhara, N. A. V. Doan and H. Amano: “Real chip evaluation of a low power CGRA with optimized application mapping”, Proceedings of the 9th International Symposium on Highly-Efficient Accelerators and Reconfigurable TechnologiesACM, p. 13 (2018).

[5] T. Kojima and H. Amano: “A configuration data multicasting method for coarse-grained reconfigurable architectures”, 2018 28th International Conference on Field Programmable Logic and Applications (FPL)IEEE, pp. 239–243 (2018).