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
VLD2019-29,DC2019-53(2019-11)

CGRA のためのアプリケーションマッピングフレームワーク GenMap の実装と実機評価

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　粗粒度再構成可能アーキテクチャ CGRA は小さな電力で、高い性能を得ることができるエネルギー効率に優れたデバイスである。しかしながら、汎用 CPU と比べコンパイラが行うべき処理が多く、最適化などが複雑化する。そのため、性能や電力効率はコンパイラ技術に大きく依存する。本研究ではいくつか異なるアーキテクチャに適応することができるアプリケーションマッピングフレームワーク GenMap を提案する。GenMap は遺伝的アルゴリズムをベースとしており、ユーザーが任意の目的関数を追加することで多目的最適化を行うことができる。GenMap の評価を行なった結果、既存手法と比較して約 20%の配線長削減や約 10%のエネルギー削減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, 遺伝的アルゴリズム, 多目的最適化

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

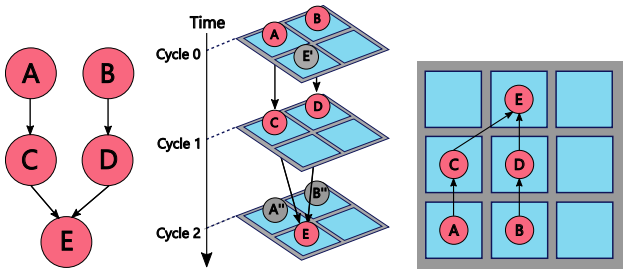
1. はじめに

近年、IoT (Internet of Things) デバイスやエッジコンピューティングの実現のために高いエネルギー効率でプログラム可能なデバイスが求められている。粗粒度再構成可能アーキテクチャ CGRA(Coarse-grained reconfigurable architecture) はこのような要求に応える一つのデバイスである。

CGRA は 2 次元アレイ状に配置された PE(Processing Element) を計算資源として持ち、主にループ処理などの compute-intensive なパートを CPU の代わりに実行するアクセラレータとして利用される。CGRA のコンパイラは PE アレイで実行される処理をデータフローグラフとして受けとり、PE アレイへ配置配線を行う。このデータフローグラフはノードが演算を示し、エッジが演算間のデータ依存を示す。CGRA で達成される性能はこのデータフローグラフマッピングの質に大きく左右されるが、最適なマッピングの探索は NP 完全な問題であると知られている。したがって、効率的な探索手法が多く提案されている。既存手法の多くは性能の最適化のみに焦点を当てており、消費電力の削減は考慮されていない。しかし、CGRA の利用が期待される用途では必ずしも最高性能を達成する必要はなく、低消費電力性が優先される場合がある。

そこで、本研究ではユーザーの要求に応じて様々な項目を同時に最適化する多目的最適化フレームワーク GenMap を提案する。GenMap は PE アレイのアーキテクチャに非依存な最適化を提供する。ユーザーは自分が利用したい PE アレイのアーキテクチャを指定することで、任意のアーキテクチャで最適化を行うことができる。

本研究では低電力志向の CGRA として提案された CMA(Cool Mega Array) [1] を対象に GenMap の評価を行う。また、本研究では 3 種類の実チップ化された CMA: 1)CC-SOTB [2], 2)CC-SOTB2 [3], 3) NVCMA [4] を用いて実機評価を行う。



(a) データフローグラフの例　(b) 時間的マッピングの場合　(c) 空間的マッピングの場合

図 1　CGRA のアプリケーションマッピングと 2 つの再構成方式

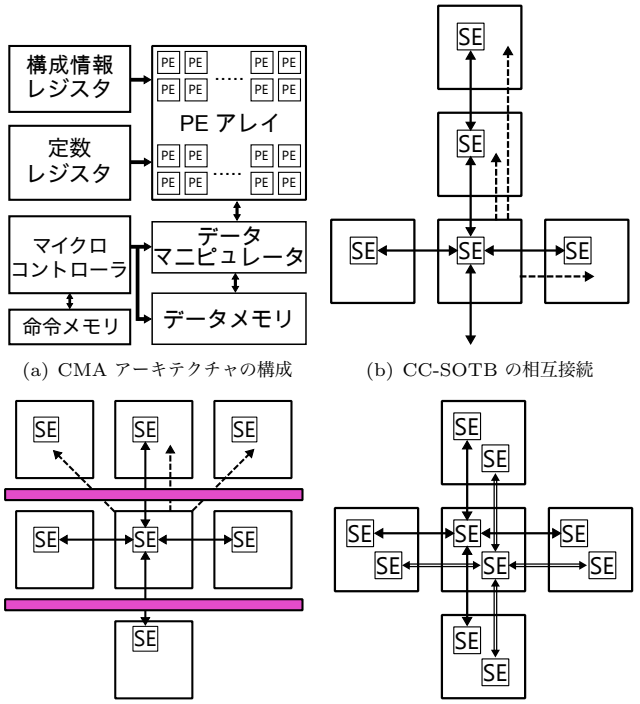
2. 本研究の背景: CGRA

2.1 CGRA におけるマッピング

CGRA のマッピングには時間的マッピング (Temporary-mapping) と空間的マッピング (Spatial-mapping) の 2 種類が存在する。図 1(a) のデータフローグラフをそれぞれの方式でマッピングした例を図 1(b) と図 1(c) に示す。時間的マッピングでは PE アレイがサイクル毎の再構成をサポートし、データフローグラフを時間方向に拡張した PE アレイへマッピングする方法である。図の例では 2×2 の PE アレイに対し 3 サイクルかけてマッピングを行なっている。一般にこの方式ではソフトウェアパイプラインの 1 つであるモジュロスケジューリングを行う。この場合、図に示すノード E' は直前のイテレーション、ノード A'', B'' は次のイテレーションの演算を示す。一方、

表 1　各アーキテクチャの相違点

	CC-SOTB	CC-SOTB2	NVCMA
アレイサイズ	12×8	12×8	8×8
ダイレクトリンク	あり	あり	なし
SE チャネル数	1	1	2
パイプラインレジスタ	なし	あり	なし
ボディバイアス制御	考慮する	考慮する	考慮しない
チップ実装	Renesas SOTB 65nm	Renesas SOTB 65nm	65nm 以降 詳細は割愛



(a) CMA アーキテクチャの構成　(b) CC-SOTB の相互接続

(c) CC-SOTB2 の相互接続　(d) NVCMA の相互接続

図 2　CMA アーキテクチャの概要

空間的マッピングではデータフローグラフを PE アレイに静的に割り付け、あるタスクを実行している間に PE アレイは同じマッピングを利用し続ける。図の例では 3×3 の PE アレイにマッピングし、このマッピングはタスク終了まで変化しない。時間的マッピングでは柔軟性が増す一方で、再構成のための電力オーバーヘッドが大きくなる。このオーバーヘッドを削減するために、本研究で用いる CMA アーキテクチャ [1] は空間的マッピングを採用している。さらに、CMA の PE からはローカルのレジスタファイルが排除され、クロック信号が不要な組合わせ回路となっている。これによって、劇的なダイナミック電力の削減を達成する。

2.2　CMA アーキテクチャ

図 2(a) に CMA アーキテクチャのブロック図を示す。前述の通り PE は組合わせ回路であるため、データメモリの読み書きを自ら行うことはできない。その代わりに CMA は小規模な RISC プロセッサであるマイクロコントローラを備えており、これが PE アレイとデータメモリとの間の入出力を制御する。データメモリは一般的な CGRA と同様にいくつかのバンクに分割され、インタリーブアクセスされる。データメモリと PE アレイの間にはデータマニピュレータと呼ばれるクロスバスイッチに似た接続網があり、柔軟なデータ転送が可能である。

本研究で用いる 3 種類の CMA アーキテクチャ (CC-SOTB, CC-SOTB2, NVCMA) の PE アレイはそれぞれいくつか異なる構造を持つ。表 1 に s の違いをまとめる。特に、PE 間の相互接続網についての違いを図 2(b)(c)(d) に示す。どのアーキテクチャも PE 内に SE(Switcing-Element) を持ち、PE アレイ内にアイランドスタイルの相互接続網を形成する。CC-SOTB と CC-SOTB2 にはこれに加えて、近傍の PE へ SE を使わずにデータを転送するためのダイレクトリンクがある。ただし、これら 2 つのダイレクトリンクの接続は図に示している通り異なっている。一方で、NVCMA には各 PE に 2 つの SE があり、チャネル数が 2 である。CC-SOTB2 では PE 行の間にパイプラインレジスタがあり、必要に応じて行を跨ぐデータをラッチする。CC-SOTB2 は計 8 行の PE 行があるため、7 本のパイプラレジスタが用意されている。これらのパイプラレジスタは個別に有効化 (ラッチモード) または無効化 (バイパスモード) できるため、アプリケーションや要求性能に応じて可変的なパイプライン構造をとることができる。さらに、CC-SOTB、CC-SOTB2 は Renesas SOTB プロセス技術でチップが実装されており、リーク電力と性能のバランスを調整するボディバイアス制御が適用されることを想定している。一方、NVCMA のチップ実装に関する詳細は共同研究先との都合上割愛する。

2.3　関 連 研 究

CGRA のマッピング最適化は NP 完全な問題であると知られており、様々なヒューリスティックスが提案されている。時間的マッピング方式の手法としては RAMP [5], MEMMap [6], HyCube [7] などが提案されている。いずれもソフトウェアパイプラインの Initial Interval(II) を最小化することを目的としている。一方、空間的マッピング方式の手法としては SPKM [8], RLMap [9] などがある。SPKM では利用する PE の行を最小化することを目的とした手法で、RLMap は利用する PE を最小化し、コンパイル時間を短縮するために強化学習を用いている。

CGRA-ME [10], [11] は細かい粒度でアーキテクチャをモデル化可能な CGRA の評価フレームワークとなっており、時間的マッピングと空間的マッピングの両方に対応した最適化ツールを備えている。配線資源の最小化を目的とした整数計画問題 (ILP) [10] を用いた手法とシミュレーテッドアニーリング [11] を用いた手法などが含まれている。

CGRA は組み込み機器などの用途で利用が期待されているにも関わらず、多くの手法が達成可能な最大性能を向上させる最適化が多い。電力削減を意識しつつも多くの手法は積極的な電力削減手法を取り入れたものは少なく、[12] の 2 レベル VDD を用いいる方法や本研究の先行研究 [3] におけるボディバイアス制御を用いた方法などしか提案されていない。しかしながら、先行研究 [3] では CC-SOTB2 にのみ焦点を当てており、有効性もこのアーキテクチャに関してしか確認されていない。そこで本研究では複数にアーキテクチャに対応可能で、多様な最適化をサポートするフレームワーク GenMap を提案する。

3.　提案手法: GenMap

GenMap による最適化フローの概要を図 3 に示す。ユーザーはマッピングするデータフローグラフと要求性能を入力する。その際に、対象となるアーキテクチャを記述した定義ファイルとチップ実装に基づくシミュレーション用のパラメータを指定する。定義ファイルでは PE アレイのサイズや、相互接続網のトポロジー、レジスタの有無、定数レジスタなどが記述可能である。シミュレーションパラメータはマッピングした結果を評価する際に用いるもので、遅延時間や消費電力などの情報を

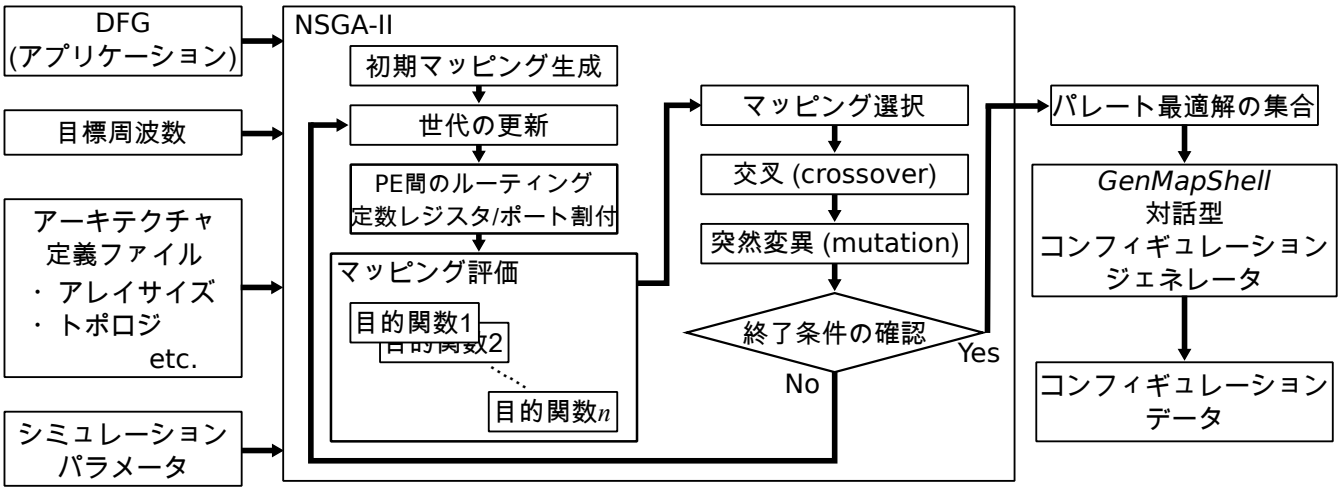
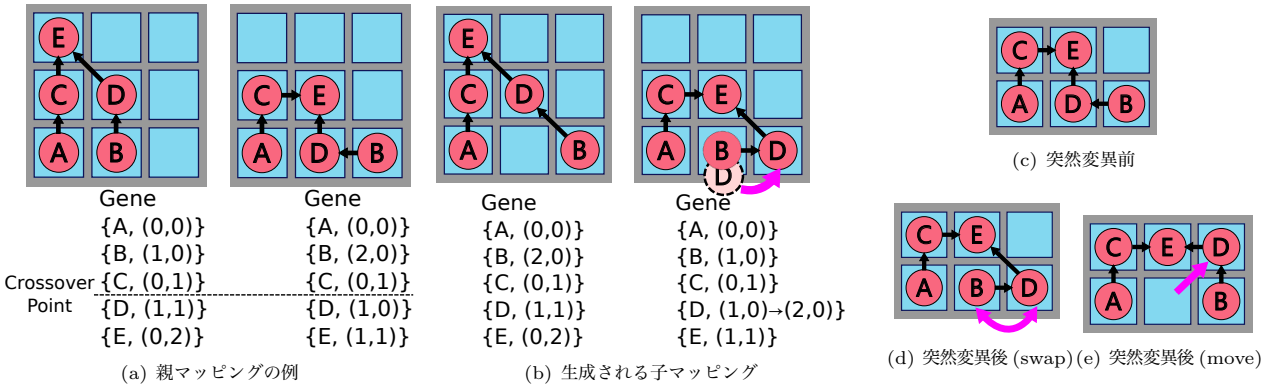


図 3　GenMap における最適化の流れ



(a) 親マッピングの例　(b) 生成される子マッピング　(c) 突然変異前　(d) 突然変異後 (swap)　(e) 突然変異後 (move)

図 4　遺伝子操作の説明

持つ。GenMap は与えられた定義情報を元に、PE アレイのリソースグラフを生成し、これをマッピング対象として利用する。

3.1　NSGA-II による多目的最適化

GenMap ではユーザーの要求に応じて、目的関数を追加、削除することが可能である。デフォルトでは総配線長が目的関数に含まれる。多目的最適化を可能にするために、GenMap は多目的遺伝的アルゴリズムの一つである NSGA-II(Non-dominated Sorting Genetic Algorithms-II) [13] を用いている。多目的最適化では他の解に優越されない評価値を持つ解をパレート最適解と呼び、一般に複数個の解が存在する。NSGA-II は一般的な遺伝的アルゴリズムを多目的にかつ効率的に探索することができるアルゴリズムである。アルゴリズムの流れは一般的なものと大きな違いはなく、解の集合 (世代) からいくつかの個体を選択し、交叉、突然変異させることで世代の更新を進めていく。

3.1.1　遺伝子コーディングと遺伝子操作

遺伝的アルゴリズムでは解を遺伝子として表現する。GenMap では遺伝子情報として 1) 演算ノード-PE のマッピング, 2) パイプライン構造 (パイプライン化されている場合) を持つ。ルーティング情報は遺伝子に含めていない。これは、遺伝子操作 (交叉、突然変異) を行う場合にルーティング情報を含めることが困難であるためである。

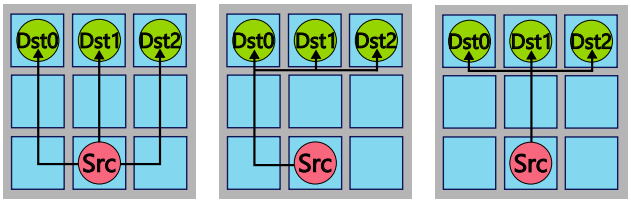
交叉は 2 つの親から 2 つの子を生成する。本研究では 1 点交叉を用いている。図 4 にマッピングの遺伝子コーディングと交叉の例を示す。図 4(a) は 2 つの親として選択されたマッピングを示す。マップされているデータフローグラフは 1(a) と同一のものを想定する。各演算ノードと PE の座標をペアにして遺伝子情報に組み込まれている。パイプライン構造に関してはここでは省略している。図のように交叉点としてノード C とノード D の間が選択された場合、子マッピングは図 4(b) のようになる。左の親の遺伝子前半 (ノード A∼C) と右の親の遺伝子後半 (ノード D,E) が一方の子になり、反対の組み合わせがもう一方の子になる。ただし、右の子に関してはノード B,D が重複してしまう。この場合、等確率でどちらか一方のノードが選ばれ、近傍の空いている PE へ移動される。等距離の近傍の空いている PE が複数存在する場合、ランダムに選択される。

また、一定の確率で遺伝子は突然変異を起こす。本研究では図 4(d) 図 4(e) に示す 2 種類の突然変異を用いる。一方は選択された 2 つのノードの割り当て PE を交換し (swap, 図 4(d))、もう一方は選択された 1 つのノードを別の空いている PE へ移動する (move, 図 4(e))。本研究では広く利用されている 0.7 の交叉確率と 0.3 の突然変異確率を用いる [14]。

3.1.2　PE 間のルーティング

3.1.1 節で述べたように、各個体は遺伝子情報に PE 間のルーティング情報を含んでおらず、子が誕生するたびに PE 間を配線資源でルーティングしなくてはいけない。GenMap では先行研究 [3] の手法をベースとした A∗ アルゴリズムを用いる方法を採用している。マッピングと異なりルーティングは貪欲アルゴリズムである理由は [15] の評価結果を見ると蟻コロニー最適化を利用して広範囲の探索を行なっても、貪欲法と比較して数パーセント程度の配線長削減しか得られないためである。

ただし、ルーティングを行う際にどの PE を優先的に配線するかは結果に大きな影響をもたらす。また、ある source ノードから複数の destination ノードへエッジを持つデータフローグラフではパスの共有を考慮すべきである。図 5 にルーティングの例を示す。図 5(a) ではパスの共有を考慮していないため、同じデータを転送しているだけにも関わらず多くの配線資



(a) パスの共有を考慮　(b) ソートなしのパス共有 (c) ソートありのパス共有

図 5　パス共有を考慮したルーティング

源を浪費する。そこで、本手法では同一のデータに関する配線を行う場合、ルーティングコストを 0 にすることで A∗ アルゴリズムを用いてもなるべくパスが共有されるように促す。この場合、どのエッジから先にルーティングするべきかが重要となる。例えば図 5(b) のようにノード Dst0 から先にルーティングすると、あとの二つもそれに引きずられてしまう。この場合、図 5(c) のように Dst1 から先にルーティングするべきである。

以上から本手法では次のような優先度でエッジを選択し、ルーティングを行う。

（1） 出力次数の小さいノードが source ノードが持つエッジ

（2） source ノードからマンハッタン距離が小さいノードへのエッジ

出力次数が高いものが優先されるのは [16] でも取られている方針であり、前述のパス共有によっていくつか経路を選択可能であるからである。もし、単一の出力しかない source ノードのエッジを後回しにすると配線不能になってしまう可能性が高まる。

3.2　初期マッピングの生成

遺伝的アルゴリズムでは初期解の生成がアルゴリズムの収束時間と得られる解の多様性、探索空間の広がりに大きな影響をもたらす。つまり、初期解の集団は短時間で収束できるようにある程度の良好な解で、局所最適解に陥らないように多様性を持っている必要がある。そこで、本研究ではグラフ描画アルゴリズムとして用いられる graphviz [17] を用いた手法を提案する。一般にグラフ描画アルゴリズムはノード間の距離をできるだけ均一にし、エッジの交差を少なくすることを目的とする。他にも力学モデルに基づくアルゴリズムなどが多数提案されているが、graphviz はデータフローグラフのような有向グラフにおいて階層構造を意識したレイアウトを行うアルゴリズムであり、CGRA の空間的マッピングに適する。

以下に graphviz を用いて初期マッピングを生成する流れと図 6 にその例を示す。

（1） 対象のデータフローグラフ (図 6(a)) を graphviz でレイアウト

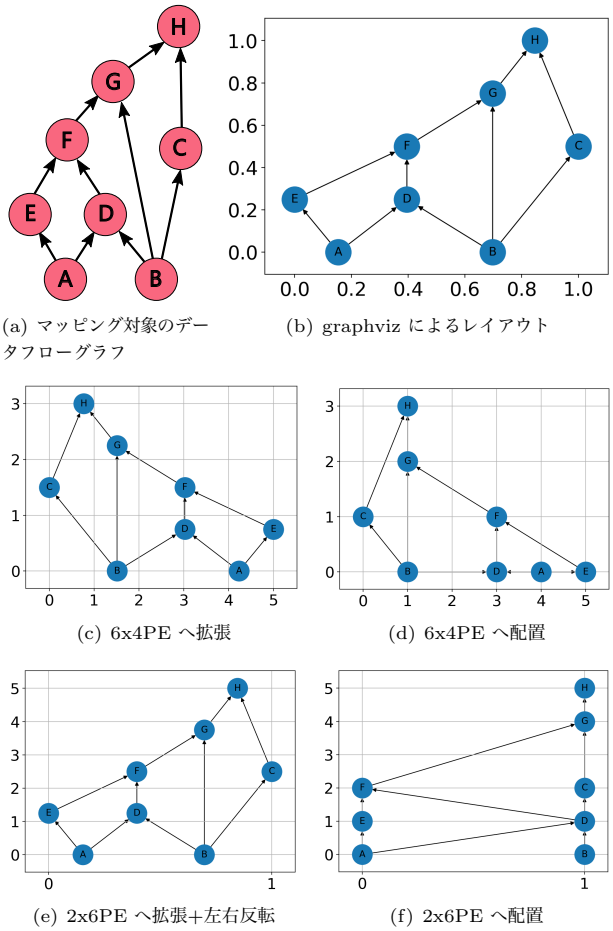
（2） レイアウト結果の座標を正規化 ((図 6(b)))

（3） 50%の確率で正規化結果を左右反転

（4） マップする PE のサイズをランダムに決定し座標を拡張 (図 6(c)、図 6(e))

（5） 各ノードを近傍の格子点 (PE) へランダムに移動 (図 6(d)、図 6(f))

(3)~(5) の操作を繰り返すことで複数の異なるマッピングを生成することができる。(5) の操作において複数のノードが同じ PE にマップされた場合、交叉処理と同様に別の近傍 PE へ移動させられる。



(a) マッピング対象のデータフローグラフ　(b) graphviz によるレイアウト

(c) 6x4PE へ拡張　(d) 6x4PE へ配置

(e) 2x6PE へ拡張+左右反転　(f) 2x6PE へ配置

図 6　graphviz を用いた初期マッピング生成

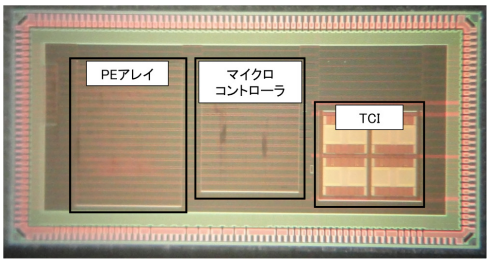


図 7　CC-SOTB のチップ写真

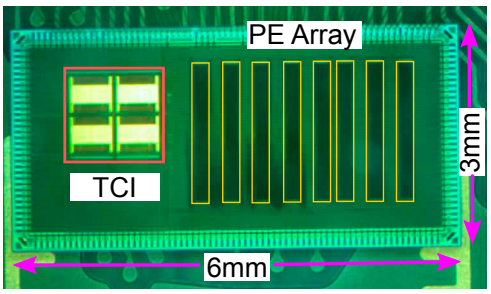


図 8　CC-SOTB2 のチップ写真

4.　評　　価

GenMap の評価を行う上で、消費電力は実機測定を行い取得する。図 7 と図 8 はそれぞれ CC-SOTB と CC-SOTB2 のチップ写真である。

4.1　予 備 評 価

本研究では GenMap で最適化する目的関数を以下の 4 つに設定して評価を行う。

表 2　モデル式の精度

	CC-SOTB	NVCMA
平均誤差	13.24%	11.84%

- 総配線長 (最小化)
- マッピング幅 (最小化)
- 消費電力 (最小化)
- タイムスラック (最大化)

消費電力はダイナミック電力とスタティック電力の和を最小化する。ただし、ボディーバイアス制御を考慮しない NVCMA では全ての解個体で同一のスタティック電力である。CC-SOTB と CC-SOTB2 に関しては、ルーティングが完了した個体は [18] で提案されている整数計画法を用いてタイミング違反を起こさない範囲でリーク電力を最小化し、最適なボディーバイアス電圧を決定する。一方、ダイナミック電力はマッピングおよびルーティング結果から [19] で提案されている電力モデル式を用いて推定される。モデル式の詳細は文献 [19] を参照されたい。

しかしながら、前述の先行研究 [19] ではこのモデル式のパラメータを CC-SOTB2 についてしか導出していない。したがって、本研究では CC-SOTB および NVCMA に関してモデル式のパラメータを決定する必要がある。パラメータ決定のために、まず両チップに異なる約 100 個のマッピングをコンフィギュレーションし、それぞれ消費電力を測定し、その結果を元にモデル式のパラメータを決定する。パラメータの決定は非線形最小二乗問題として扱い、Levenberg-Marquardt 法を用いる。決定後のパラメータを用いて算出した電力値と実測値の平均誤差を表 2 に示す。CC-SOTB と NVCMA はそれぞれ異なるアーキテクチャ、プロセス技術で実装されているものの、共通のモデル式でパラメータを適切に決定することで平均誤差 10%程度の精度で電力を推定することができる。本モデルはポストレイアウトシミュレーションのような長い時間要して正確に電力を見積もるためではなく、高速に各マッピングの評価を行うことが目的である。したがって、10%程度の誤差はこの利用目的には十分許容できるもである。

4.2　最適化結果の比較

4.2.1　遺伝的アルゴリズムの収束

NSGA-II による最適化の収束を確認する。3.2 節で提案した初期マッピングの生成手法の効果を確認するために、3 つのアーキテクチャに対して 3 種類の初期マッピング生成手法をそれぞれ評価し、比較を行う。1 つは graphviz を用いた提案手法で、残りはランダムにマッピングを生成した場合である。ただし、配置する PE をランダムに選択しつつ、データフローグラフのノード割り当てにトポロジカルソートを用いる場合 (tsort)、とノードの割り当ても完全にランダムに行う (random) 場合の 2 つを評価している。

各世代におけるパレート最適解の Hypervolume が変化する様子を図 9 にまとめる。Hypervolume は解の多様性と収束性の両方を示す尺度であり、アーキテクチャ間で値の違いは大きな意味を持たない。この結果には探索空間の大きさが関係している。3 つの中で最も探索空間が広いのは CC-SOTB2 であり、マッピングに加えてパイプライン構造、ボディバイアス電圧の決定も行う必要がある。一方、NVCMA は他の 2 つ比較して PE アレイサイズも小さく、ボディバイアス電圧の決定も不要であるため探索空間は小さい。提案手法の graphviz を用いた手法では探索空間の広い CC-SOTB2 で特に効果的であり、トポロジカルソートを利用する場合、100 世代を過ぎたあたりから解の更新が進まなくなり、graphviz を用いる場合と比べて多様性や最適化結果は劣る。また、どのアーキテクチャにおいても完全にランダムな初期マッピングを用いると他の方法に比べ収束に時間がかかる。しかし、探索空間の小さい NVCMA では 3 つの方法にそれほど大きな違いはない。

表 3　最適化結果の比較 (af)

手法	CC-SOTB		CC-SOTB2		NVCMA	
	配線長	map 幅	配線長	map 幅	配線長	map 幅
CGRA-ME [10]	45	3	69	3	103	3
SPKM [8]	62	3	57	4	79	3
GenMap$_{wire}$	41	5	38	6	71	4
GenMap$_{width}$	53	3	56	3	81	3

4.2.2　配線長とマッピング幅の比較

GenMap による最適化結果と比較を行うために、CGRA-ME の ILP mapper [10] と SPKM [8] をベースにした手法でもマッピングを行なった。SPKM では CMA のような配線資源として SE を持つ CGRA を直接扱うことができない。そのため、マッピングのみ SPKM を用い、ルーティングには GenMap に組み込まれているものを利用する。一方で、CGRA-ME は CMA の各アーキテクチャを記述することができ、ルーティングまで行うことができる。本論文ではスペースの都合上 24bit アルファブレンド (24 演算ノード) のアプリケーションのみに焦点を当てる。

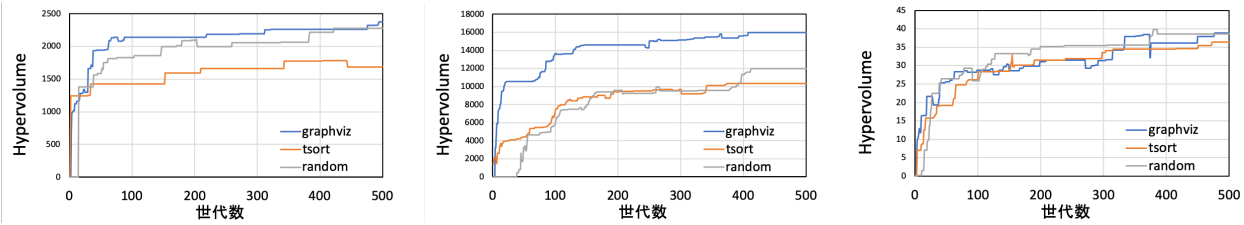
マッピング結果の比較を表 3 に示す。CGRA-ME はマッピング問題を整数計画問題へモデル化することにより、非常に強力なマッピング能力を持つ。本評価で用いたアプリケーションは 24 ノードであるため、3 列 ×8 行の PE が最低限必要なマッピングサイズであるが、CGRA-ME はいずれのアーキテクチャにおいてもこのサイズでのマッピングに成功している。SPKM をベースとした手法では CC-SOTB2 を除き、CGRA-ME と同じ幅でマッピングに成功している。

GenMap による結果はパレート最適解の中から、配線長が最小のもの (GenMap$_{wire}$) とマッピング幅が最小のもの (GenMap$_{width}$) のみを抽出し、表 3 に示している。CGRA-ME で確認されている最小マッピング幅をいずれのアーキテクチャでも得ることができている。さらに、配線長とマッピング幅を同時に最適化していくため、同じマッピング幅でありながら CC-SOTB2、NVCMA においてそれぞれ約 19%, 21%の配線長削減を達成した。しかし、CC-SOTB に関しては CGRA-ME と比べ 17%程度配線長が増加している。一方、配線長を優先すると GenMap はいずれのアーキテクチャにおいて他の 2 つの手法よりも配線長の短いマッピングを得ることができている。

多くのマッピング手法では配線資源として SE を持つことを想定としておらず、CMA におけるダイレクトリンクのように隣接 PE との相互接続のみで配線を行うことが多い。CGRA-ME は SE を持つ PE をモデル化することは可能であるが、それを考慮したルーティング能力はやや低いと推測される。CC-SOTB には 2.2 節の図 2(b) に示した通り、1 つ先の PE にもダイレクトリンクが伸びている。したがって、ダイレクトリンクを活用すれば SE を利用しなくても配線可能なことが多く、結果的 CGRA-ME の方がよい配線結果を示している。

4.2.3　消費電力の比較

最後に、消費電力の比較を行う。本稿では CC-SOTB を用いた場合の結果のみを報告する。様々な要求性能を想定して GenMap、CGRA-ME、SPKM が生成したマッピングを動作



(a) CC-SOTB の結果　(b) CC-SOTB2 の結果　(c) NVCMA の結果

図 9　各アーキテクチャにおける最適化収束の様子 (24bit アルファブレンド)

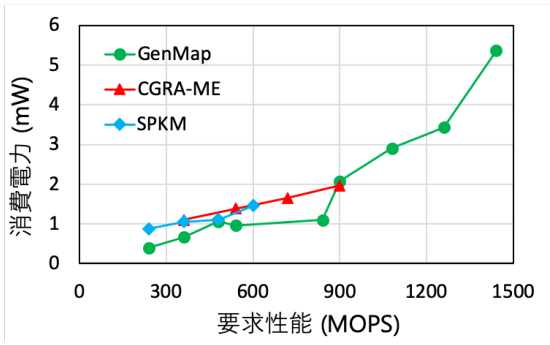


図 10　消費電力の比較 (CC-SOTB, af)

させ電力を測定した結果を図 10 に示す。ただし、これらの電力値は PE アレイ部のみの消費電力である。CGRA-ME、SPKM は要求性能に関わらず、常に一定のマッピングを生成するため、動作周波数の増加によって線形にダイナミック電力が増加している。一方で、GenMap はボディバイアス制御を考慮しマッピングを決定できるため、要求性能が低い場合では、他の 2 つのマッピングよりも小さい電力を達成し、CGRA-ME と比べ平均 11.3%、SPKM と比べ平均 14.6%の消費エネルギーを削減している。さらに、ボディバイアス制御によってリーク電力増加の対価として CGRA-ME と比べ 1.6 倍、SPKM と比べ 2.4 倍の成功性能を達成している。

5. 結　　論

本稿では、空間的マッピングを行う CGRA のためのマッピングフレームワーク GenMap を提案した。GenMap は遺伝的アルゴリズムを用いて多目的な最適化を行う。すでに実チップ化されている 3 つのアーキテクチャ CC-SOTB、CC-SOTB2、NVCMA を対象に評価を行なった結果、20%程度の配線長削減や 10%程度のエネルギー削減、1.6 倍以上の性能向上を達成した。

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] Y. Matsushita, H. Okuhara, K. Masuyama, Y. Fujita, R. Kawano and H. Amano: “Body bias grain size exploration for a coarse grained reconfigurable accelerator”, 2016 26th International Conference on Field Programmable Logic and Applications (FPL), pp. 1–4 (2016).

[3] 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).

[4] T. Ikezoe, H. Amano, J. Akaike, K. Usami, M. Kudo, K. Hiraga, Y. Shuto and K. Yagami: “A Coarse Grained-Reconfigurable Accelerator with energy efficient MTJ-based Non-volatile Flip-flops”, 2018 International Conference on ReConFigurable Computing and FPGAs (ReConFig)IEEE, pp. 1–6 (2018).

[5] S. Dave, M. Balasubramanian and A. Shrivastava: “RAMP: resource-aware mapping for CGRAs”, Proceedings of the 55th Annual Design Automation ConferenceACM, p. 127 (2018).

[6] S. Yin, X. Yao, D. Liu, L. Liu and S. Wei: “Memory-aware loop mapping on coarse-grained reconfigurable architectures”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, **24**, 5, pp. 1895–1908 (2016).

[7] M. Karunaratne, A. K. Mohite, T. Mitra and L.-S. Peh: “HyCUBE: A CGRA with reconfigurable single-cycle multi-hop interconnect”, Design Automation Conference (DAC), 2017 54th ACM/EDAC/IEEEIEEE, pp. 1–6 (2017).

[8] J. W. Yoon, A. Shrivastava, S. Park, M. Ahn, R. Jeyapaul and Y. Paek: “SPKM: A novel graph drawing based algorithm for application mapping onto coarse-grained reconfigurable architectures”, Proceedings of the 2008 Asia and South Pacific Design Automation ConferenceIEEE Computer Society Press, pp. 776–782 (2008).

[9] D. Liu, S. Yin, G. Luo, J. Shang, L. Liu, S. Wei, Y. Feng and S. Zhou: “Data-Flow Graph Mapping Optimization for CGRA with Deep Reinforcement Learning”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2018).

[10] M. J. Walker and J. H. Anderson: “Generic connectivity-based cgra mapping via integer linear programming”, 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)IEEE, pp. 65–73 (2019).

[11] S. A. Chin, N. Sakamoto, A. Rui, J. Zhao, J. H. Kim, Y. Hara-Azumi and J. Anderson: “Cgra-me: A unified framework for cgra modelling and exploration”, Application-specific Systems, Architectures and Processors (ASAP), 2017 IEEE 28th International Conference onIEEE, pp. 184–189 (2017).

[12] J. Gu, S. Yin, L. Liu and S. Wei: “Energy-aware loops mapping on multi-vdd CGRAs without performance degradation”, Design Automation Conference (ASP-DAC), 2017 22nd Asia and South PacificIEEE, pp. 312–317 (2017).

[13] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan: “A fast and elitist multiobjective genetic algorithm: NSGA-II”, IEEE transactions on evolutionary computation, **6**, 2, pp. 182–197 (2002).

[14] L. Davis: “Adapting operator probabilities in genetic algorithms”, Proceedings of the third international conference on Genetic algor ithms, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., pp. 61–69 (1989).

[15] L. Zhou, J. Zhang and H. Liu: “Ant Colony Algorithm for Steiner Tree Problem in CGRA Mapping”, Information Science and Control Engineering (ICISCE), 2017 4th International Conference onIEEE, pp. 198–202 (2017).

[16] H. Park, K. Fan, S. A. Mahlke, T. Oh, H. Kim and H.-s. Kim: “Edge-centric modulo scheduling for coarse-grained reconfigurable architectures”, Proceedings of the 17th international conference on Parallel architectures and compilation techniquesACM, pp. 166–176 (2008).

[17] E. R. Gansner, E. Koutsofios, S. C. North and K.-P. Vo: “A technique for drawing directed graphs”, IEEE Transactions on Software Engineering, **19**, 3, pp. 214–230 (1993).

[18] T. Kojima, N. Ando, H. Okuhara, N. A. V. Doan and H. Amano: “Body bias optimization for variable pipelined CGRA”, Field Programmable Logic and Applications (FPL), 2017 27th International Conference onIEEE, pp. 1–4 (2017).

[19] T. Kojima, N. Ando, H. Okuhara and H. Amano: “Glitch-aware variable pipeline optimization for CGRAs”, 2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig), pp. 1–6 (2017).