

遺伝的アルゴリズムを用いたロジックインメモリ構造 VLSIプロセッサのハイレベルシンセシス

High-Level Synthesis of a Logic-in-Memory-Based VLSI Processor Using a Genetic Algorithm

○工藤隆男*, 張山昌論**, 亀山充隆**

○Takao Kudoh*, Masanori Hariyama**, Michitaka Kameyama**

*八戸高専, **東北大学情報科学研究科

*Hachinohe National college of Technology,

*Graduate School of Information Sciences, Tohoku University

キーワード: 並列構造VLSIプロセッサ(Parallel VLSI processor), 転送ボトルネックフリーアーキテクチャ(Bottleneck free architecture), アロケーションとスケジューリングの統合(Integration of allocation and scheduling), ロジックインメモリ構造(Logic-in-memory architecture), 遺伝的アルゴリズム(Genetic Algorithm)

連絡先: 〒039-1192 八戸市田面木字上野平16-1 八戸工業高等専門学校 電気工学科 工藤隆男

Tel.: (0178)27-7279, Fax.: (0178)27-7279, E-mail: tkudoh-e@hachinohe-ct.ac.jp

1. はじめに

危険状態を事前に検知し運転者に注意を促す高安全知能自動車のようなリアルワールド応用知能集積システムを実現する場合, 知能情報処理を高度化しようとすればするほど, 種々の膨大な演算を瞬時のうちに行うことが要求される. よって, リアルワールド応用知能集積システムを実現するためには転送ボトルネックのないVLSIアーキテクチャが絶対的に必要になる.

ディーブサブミクロン技術の進展に伴い, 配線遅延が深刻な問題になりつつある¹⁾. 配線遅延を小さくするためには, 短い配線によるデータ転送を可能とする規則的なアーキテクチャが望まれる. 規則的なアーキテクチャを実現する手法として, 筆者等は演算機能と記憶機能を融合させたロジック

インメモリVLSIアーキテクチャに基づくプロセッサを提案しており^{2,3,4)}, この一般的設計法が望まれる.

本稿では1個の演算器と数ワードの記憶機能からなるロジックインメモリ構造のモジュールを処理要素(PRE)とし, 隣接したPREを接続したハードウェアモデルに対して, 演算仕様がデータ依存グラフで与えられるとき, 出きる限り小さい処理時間の演算を可能とするアロケーションとスケジューリングを探索する方法を提案する. ここで用いるハードウェアモデルにおいては, 1度のデータ転送を隣接するモジュールに制限していることから, 演算器のアロケーションとスケジューリングは互いに影響し合う. よって, アロケーションとスケジューリングとを統合した設計が必要になる. デー

タ依存グラフのノード数が多くなると、スケジューリングとアロケーションの組合せは膨大になることから、厳密な最小の処理時間を実現できるアロケーションとスケジューリングを、実用的な探索時間内で求めることは極めて困難である。

そこで、遺伝的アルゴリズムに基づき、可能な限り最小の処理時間に近い処理時間で演算を実行できるアロケーションとスケジューリングを高速に探索する方法を提案している。データ依存グラフの1個のノードへのアロケーションとスケジューリングを1個の遺伝子にマッピングすることにより、データ依存グラフへのアロケーションとスケジューリングを1個の個体で表現できる。これにより、アロケーションとスケジューリングの膨大な組合せの1部を個体群として扱うことを可能とすることから、交叉を繰り返すことにより、データ依存グラフで与えられる演算仕様を最小の処理時間でできる限り近い処理時間で実行できるアロケーションとスケジューリングを高速に探索できる。

最後に、列挙法や分枝限定法を用いるとき、実用的な計算時間で探索できないほどノード数の多いデータ依存グラフに対して、遺伝的アルゴリズムを用いると、最適解に近い近似解を高速に探索できること具体例を示す。

2. ハードウェアモデル

データ転送における配線遅延を小さくするためには、短い配線によるデータ転送を可能とする規則的な構造のハードウェアモデルが望まれる。このようなハードウェアモデルとして、図1のように1個の処理要素 (PE) と1ワードのレジスタ (R) とローカルメモリ (LM) を一体化したロジックインメモリモジュールから成る空間並列構造のハードウェアモデルをとりあげる。kは正の整数でロジックインメモリモジュールの番号を表す。

モジュール間のデータ転送をレジスタを介して

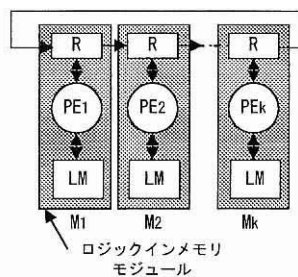


Fig. 1 ハードウェアモデル

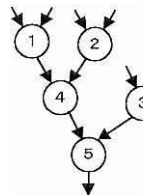


Fig. 2 DFG

隣接するモジュールに限定することにより、データ転送用の配線を短くできる。データは、モジュール M_1 から M_2 へ、 M_k から M_1 へというように右周りで転送される。各PEはデータ依存グラフ (DFG) の処理に必要な演算器、たとえば加算器や乗算器などの演算器を備えており、アロケーションを行う演算ノードの種類に対応して、演算を切り替えて実行できるものとする。

このハードウェアモデルの利点は、負荷分散タイプやデータ再利用タイプなどのように、演算ノード間の転送が少ないか、規則性のある転送のクラスに適する。

以下の例題については、隣接するモジュール間のデータ転送に要する時間は1クロックとし、演算に要する時間は2クロックとするが、任意の設定が可能である。

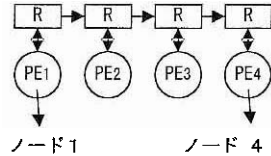
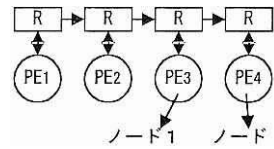
3. アロケーションとスケジューリングの統合の必要性

PE数を限定する本ハードウェアモデルに基づくプロセッサ上で、DFGで与えられる演算仕様を実時間で処理をする場合、データ転送時間と演算時間とを合わせた処理時間全体の最小化が重要になる。

DFGで与えられる動作仕様を満足するようにPEのアロケーションとスケジューリングを行う場合、PE間のデータ転送に必要なクロック数は、転送のために通過するモジュール数に依存するため、スケジューリングはアロケーションが確定しないと決定できない。この例を示すために、Fig.2のノード1とノード4に、Fig.1のPEのアロケーションを行う場合をとりあげる。Fig.2のDFGのノードは演算を、矢印はデータ依存関係を表す。2個のノードにFig.3(a)のように隣接するPEのアロケーションを行う場合と、(b)のように離れたPEのアロケーションを行う場合とでは、(c)のように、データ転送に要するクロック数が異なる。このようにアロケーションとデータ転送に要するクロック数とは、不可分の関係にあることから、アロケーションとスケジューリングを統合する設計法が必要になる。

4. アロケーションとスケジューリングの組合せ

PEの個数を n とし、スケジューリングされるクロックステップはクロック1からクロック s までにあるとする。アロケーションとスケジューリングの組合せを表すために、Fig. 4のように、行方向には、 PE_1 から PE_n までをとり、列方向には、クロック1からクロック s までをとる。このことにより、ノードの演算時間は2クロックステップであるとしているので、あるノードに PE_i のアロケーションを行い、連続する $n, n+1$ の2クロックステップ



(b) アロケーション 2



(a) の場合

(b) の場合



(c) スケジューリングへの影響

Fig. 3 アロケーションとスケジューリングの依存性

ブのスケジューリングを行うことを、2個の要素 (i, h) , $(i, h+1)$ で表すことができる。この表を用いると、例えば、Fig.2へのアロケーションとスケジューリングは、Fig.5のように表すことができる。要素2,3は、ノード1にアロケーションするPE1の出力データをPE2へ転送するための転送クロックステップに対応し、Fig.3(c)の(a)の場合のcomに相当する。

4.1 最適解の探索問題

物理的に可能なアロケーションとスケジューリングの組合せを可能解と呼ぶことにする。可能解が存在できる必要条件を示す。

- (1) 異なるノードに、同一のクロックステップ

	PE1	PE2	...	PE _i	...	PE _k
(1, 1)						
				(i, m)		
						(k, s)

Fig. 4 アロケーションとスケジューリングの組合せ

	PE1	PE2	PE3
Clock 1			
Clock 2	①	②	③
Clock 3			
Clock 4		④	
Clock 5			
Clock 6		⑤	
Clock 7			
Clock 8			

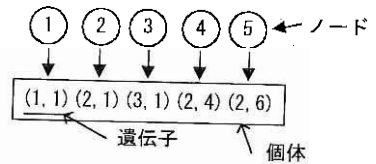
Fig. 5 可能解

プのスケジューリングが行われているとき、同一のPEのアロケーションを行わない。

(2) アロケーションとスケジューリングが、ノード間のデータ依存関係を満たす。

(3) ノードの演算時間は2クロックであるので、連続した2クロックステップのスケジューリングを行う。

データ依存グラフで与えられた演算を最小クロックステップ数で実行できる可能解を最適解と呼ぶことにする。最適解を求めるためには、すべての可能解の中から、処理時間が最小の可能解を探索すればよい。しかし、演算ノード数が多くなると可能解の個数は爆発的に多くなることから、実用的な探索時間内において、すべての可能解をしらみつぶ的に調べることは不可能に近い。また、最適解になり得る可能解だけを探索する分枝限定法を用いても、組合せの爆発問題は避けられない。



(i, m) : iはPE番号,
mはクロックステップ

Fig. 6 可能解の個体表現

5. 遺伝的アルゴリズムに基づく最適解の探索

最適解の処理時間に可能な限り近い処理時間を実現できる可能解を近似解と呼ぶことにし、近似解の探索について考える。

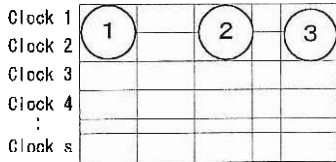
1個の可能解を1個の個体で表すことにする。DFGのノード数が m のとき、1個のノードへのアロケーションとスケジューリングを1個の遺伝子で表すことにより、1個の可能解すなわち個体は、 m 個の遺伝子の並びで表すことが出来る。例えば、Fig. 5の可能解は、Fig.6のように表すことが出来る。個体の左から n 番目の遺伝子が、ノード n へのアロケーションとスケジューリングを表すものとする。

5.1 データ依存関係を満足する個体の生成

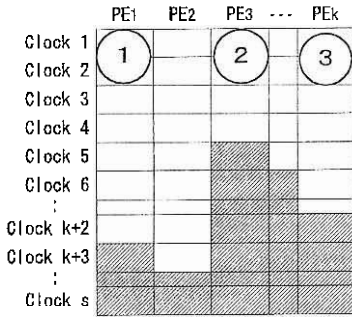
Fig. 4において、アロケーションが同じで連続する2個のクロックステップを2連接要素と呼ぶことにする。

1個の個体を生成するためには、DFGの m 個のノードへの物理的に実行可能なアロケーションとスケジューリングを求めればよい。多様性のある個体群を生成するために、乱数を用いて物理的に実行可能なアロケーションとスケジューリングを行うことにする。

1個の個体を生成する例として、Fig.2のDFGを取り上げる。DFGのノードには半順序関係を満たすようにノード番号が与えられているものとし、



(a) 個体の生成



(b) ノード4への遺伝子候補の集合

Fig. 7 個体の生成

ノード番号の順に、遺伝子を決定することにより、個体を生成する。

ノード1に対する遺伝子を決定するためには、ノード1は、他のノードの出力データを必要としないので、Fig. 7(a)の中の2連接要素を割当てればよい。そこで、乱数に基づき1番の演算器のアロケーションを行い、クロックステップ1のスケジューリングを行う。ノードの演算には2クロックを必要とすることから、2連接要素{(1,1), (2,1)}を割り当てる。このことを表すために、Fig. 7(a)のクロック1、クロック2の行と、 PE_1 の列が交差する2連接要素に、ノード番号1を記入する。

ノード{1,2}に対する遺伝子を求める。ノード2はノード1からの出力データを必要としない。よって、ノード1に割当てた2連接要素以外の2連接要素の中から、乱数に基づき1個の2連接要素を割り当てる。Fig. 7(a)はノード{1,2,3}へのアロケーションとスケジューリングを表す。

ノード{1,2,3,4}に対する遺伝子を求める。ノード

4には、ノード{1,2}とのデータ依存関係を満足する2連接要素の割当てが必要である。ノード4がノード1の出力データのみを必要とする場合について考える。ノード1の出力データはクロックステップ3で発生するので、ノード4に PE_1 のアロケーションを行う場合、クロックステップ3,4以降のスケジューリングを行う必要がある。隣接するPE間のデータ転送には、1クロックを要するので、 g を2以上の整数とし、ノード4に PE_g のアロケーションを行う場合、ノード4がノード1の出力データを入力するためには、ノード4にはクロックステップ $(g+2)$ 以降のスケジューリングを行う必要がある。ノード{1,2,3,4}に対する遺伝子を求めるためには、ノード4がノード{1,2}とのデータ依存関係を満足できることが必要である。よって、Fig. 7(b)の斜線の要素の中の2連接要素をノード4に割当てればよい。

n 個 ($n \leq m-1$)のノードに対する遺伝子が与えられているとき、 n 個のノードと、そのノード集合に含まれていないノード $n+1$ に対する遺伝子の求め方を一般化する。PEの個数を k とし、スケジューリングされるクロックステップはクロック1からクロック s までにあるとする。

ノード集合に含まれるどのノードも、ノード $n+1$ とデータ依存関係がない場合、 ks 個の要素から n 個のノードに割当てた $2n$ 個の要素を除く、 $ks-2n$ 個の要素の中の2連接要素を、ノード $n+1$ に割当てればよい。

ノード集合に含まれるノード j と、ノード $n+1$ との間にデータ依存関係がある場合、 $ks-2n$ 個の要素の中の2連接要素において、ノード $n+1$ がノード j の出力データを入力できる2連接要素をノード $n+1$ に割当てればよい。

n 個のノードとノード $n+1$ に対する遺伝子は、 n 個のノードに割当てた2連接要素と、ノード $n+1$ に割当てた2連接要素で表すことができる。

以上のようにして、ノード1に対応する遺伝子

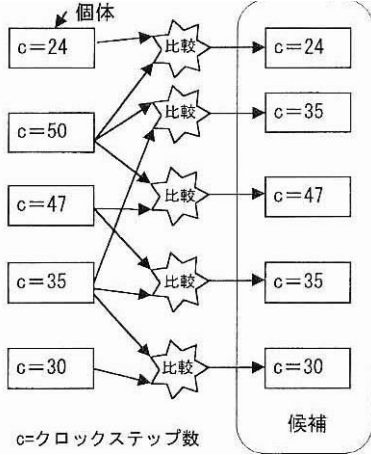


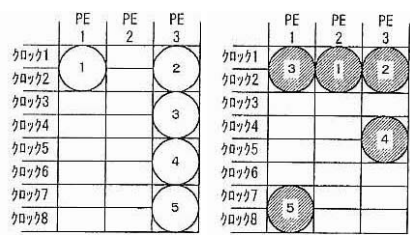
Fig. 8 交叉候補の選択

から順にm個の遺伝子を並べることにより1個の個体を生成できる。初期個体群を生成するためには、同様の手順を繰り返せばよい。

5.2 交叉

最適解に近い近似解を探索するために、小さい処理時間（クロックステップ数）を実現できる個体同士を交叉することにする。トーナメント方式⁵⁾に基づき、個体群から選択した2個の個体のうち処理時間の少ない個体を交叉の候補に登録する。処理時間の多い個体は淘汰される。Fig.8では、クロックステップ数が50の個体が淘汰される。

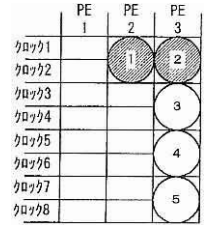
交叉は、候補の中から任意の2個の個体を選択し、1点交叉を行うことにする。例えば、Fig.9の(a)(b)の個体のノード{3, 4, 5}に対する遺伝子を交叉する場合、新しく生成される個体は、(d)のようにそのまま可能解を表す場合もあるが、(c)の場合は、アロケーションとスケジューリングが物理的に実行不可能である。



(a) 個体1

(b) 個体2

ノード{3, 4, 5}を交叉)



(c) 致死遺伝子発生

(d) 個体3

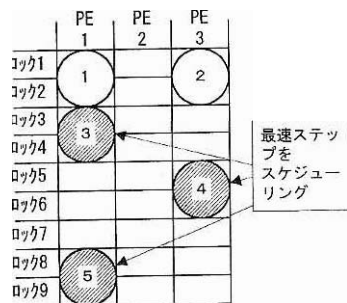
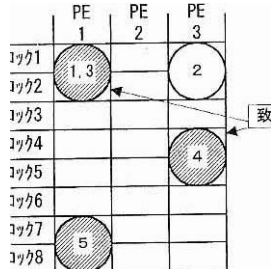
Fig. 9 交叉

5.3 致死遺伝子の復活

アロケーションとスケジューリングを物理的に実行できない遺伝子を致死遺伝子と呼ぶことにする。物理的に実行可能なアロケーションとスケジューリングの条件を満足しつつ、可能な限り早い段階で演算を開始できるクロックステップを最速ステップと呼ぶことにする。

致死遺伝子を含む個体が多くなると、最適解に近い近似解の探索が困難になると予想される。また、最適解に近い近似解を求めるためには、小さい処理時間の個体を得たい。そこで、致死遺伝子を含む個体を処理時間の少ない個体に復活させることについて考える。致死遺伝子の特性を残しつつ、処理時間の短い個体にするために、アロケーションは変更しないで、スケジューリングのみを調整することにする。

致死遺伝子の復活の例として、Fig.10において、



復活後

Fig. 10 致死遺伝子の復活

ノード{3,4}の遺伝子を復活させる。ノード3へのPE1のアロケーションは変更しないで、最速ステップを含む2 連接要素(1,3),(1,4)をスケジューリングする。ノード4についても、アロケーションは変更しないで、最速ステップのスケジューリングを行い、2 連接要素(3,5),(3,6)を割当てる。この場合、ノード4からノード5へのデータ転送ができなくなる。そこで、ノード5にあらためて2 連接要素(1,8),(1,9)を割当てる。

致死遺伝子の復活を一般化する。致死遺伝子が発生したノードの最小番号をjとする。ノードjからノードmまでノード番号順に、ノードへのアロケーションは変更せずに、最速ステップを含む2 連接要素をスケジューリングする。

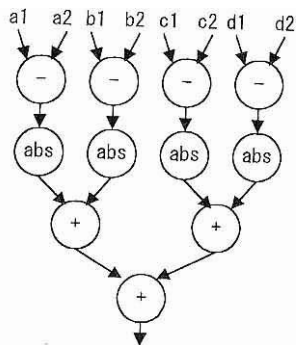


Fig. 11 評価に用いるDFG

6. 探索結果

ハードウェアモデルの演算器数を4とする。Fig.11に示す絶対差分演算用の演算をできる限り小さい処理時間で実行できるアロケーションとスケジューリングを求める。Fig.11のノード数は11であるので、ノード数が11の倍数の場合は、倍数だけのDFGを用いる。それ以外のノード数の場合は、DFGの1部を用いる。

6.1 探索時間

解の探索にパソコン (cerellon400MHz) を用いるとき、列挙法や分枝限定法に基づく探索においてはFig. 12に示すように、ノード数が15個以上で、探索時間が指数関数的に大きくなることから、これ以上のノード数の場合、実用的な探索時間内の探索は行えない。

これに対し、遺伝的アルゴリズムを適用すると、ノード数が多くなっても探索時間が指数関数的に大きくなる現象は見られない。すなわち高速な探索を可能とする。なお、遺伝的アルゴリズムの適用条件は、最大世代に達する前に近似解のステップ数を飽和させられるようにするために、実験値から最大世代数を200とし、個体数を100とした。探索時間は、第1世代の生成開始から処理時間が

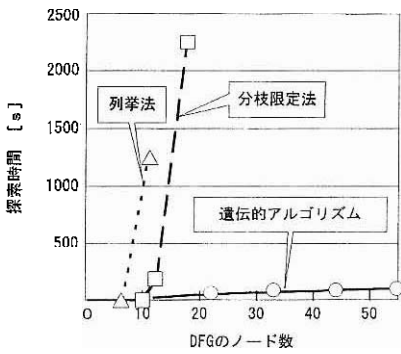


Fig. 12 探索時間の比較

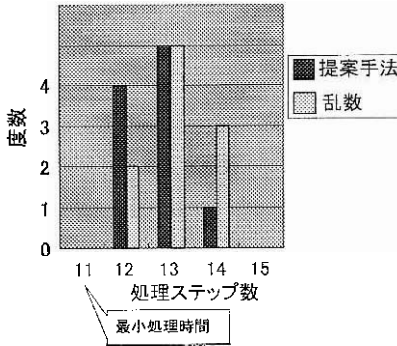


Fig. 13 最適解と近似解との距離

飽和するまでに要する計算時間をとした。

6.2 最適解と近似解との距離

最適解の処理時間（ステップ数）と近似解の処理時間の差を、最適解と近似解との距離と呼ぶことにする。

Fig. 13に示すように、致死遺伝子の復活に物理的に実行可能な最も早いスケジューリングを行う場合のほうが、乱数を用いたスケジューリングを行うより、最適解に近い近似解を得ることができる。近似解の探索に必要な平均時間は乱数を用いると約30秒、提案手法を用いると約15秒となり、ほぼ半減する。

物理的に実行可能な最も早いステップのスケジ

ューリングを行う致死遺伝子の復活のほうが、より最適解に近い近似解のより高速な探索を可能にする。

7. むすび

ハードウェアモデルとデータ依存グラフが与えられるとき、通信時間も含めた処理時間全体を、できる限り小さくするアロケーションとスケジューリングの決定問題に遺伝的アルゴリズムを適用した結果、分枝限定法を用いる場合に比較し、極めて高速に実行解を探索できることを明らかにした。

このアロケーションとスケジューリングとを統合したハイレベルシンセシスの考え方は、ロジックインメモリVLSIプロセッサに限らず、演算子のアロケーションとスケジューリングが相互に影響する一般のプロセッサのハイレベルシンセシスにも適用可能である。

参考文献

- 1) 櫻井貴康, "総論—システムLSIのアプリケーションとシステムLSIの課題", 信学会誌, vol.81, no.11, pp.1082-1086, Nov., 1998.
- 2) 工藤隆男, 羽生貴弘, 亀山充隆, "ロジックインメモリアーキテクチャに基づく道路抽出VLSIプロセッサの構成", 計測自動制御学会論文集, vol.36, No.11, 1009/1018(2000).
- 3) 堀井崇史, 羽生貴弘, 亀山充隆, "共通バス本数最小化に着目したロジックインメモリVLSIシステム的设计", 電気関係学会東北支部連合大会講演予稿集, 2H17, 1998.
- 4) 張山昌論, 工藤隆男, 亀山充隆, "最適アロケーションに基づく道路抽出VLSIプロセッサとその高安全知能自動車への応用", 電子情報通信学会論文集, vol.J84-D-1, No.6, pp.531-539, 2001.
- 5) 伊庭奇志, "遺伝的アルゴリズムの基礎", オーム社, 2000.