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

工藤隆男[†] 張山昌論[††] 亀山充隆[††]

ディープサブミクロン VLSI においては，配線に起因する性能劣化が深刻な問題となっている．本稿では，配線の影響の少ないハードウェアモデルとして，ローカルメモリと演算器を一体化したモジュールをリング状に結合するなど，簡単な相互結合回路網で結合したロジックインメモリ構造を設計の対象とし，演算器数制約下で処理時間最小化を行うためのアロケーション・スケジューリングの高速探索法を提案する．本アルゴリズムでは，GA に ASAP（As Soon As possible）スケジューリングを導入することにより，致死遺伝子を抑制し，効率良く探索を行うことができる．種々の大規模問題に対する評価を行い，解を高速に求められることを示す．

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

Takao Kudoh,[†] Masanori Hariyama[††] and Michitaka Kameyama[††]

One of the most serious problems in recent VLSI systems is data transfer bottleneck between memories and processing elements due to large interconnection delay. A logic-in-memory structure with a simple interconnection such as ring network is employed to solve the problem. An efficient search method for processing-time minimization under a area constraint is presented for the logic-in-memory structure. The proposed method is based on combination of genetic-algorithm-based allocation and as-soon-as-possible (ASAP) scheduling. The use of the ASAP scheduling prevents the generation of non-valid chromosome, and also minimizes the processing time under a given allocation. Evaluation of results for large-size problems demonstrate that it is possible to get high quality solutions in reasonable time.

1. はじめに

VLSI 製造技術の進展にともなう高集積化のため，ハイレベルシンセシスの重要性が高まっている[1)]．

また，ディープサブミクロン VLSI プロセッサにおいては，シグナルインテグリティや配線遅延などの配線に起因する性能劣化が深刻になっている[2)]．特に，メモリ部と演算部との間に複雑な相互結合回路網を備えるアーキテクチャにおいては，相互結合回路網が大規模になればなるほど配線問題が深刻になりがちである．このことから，シンプルな相互結合網を有する並列構造 VLSI プロセッサアーキテクチャが重要となる．

本稿では，そのようなアーキテクチャモデルとして，演算器に専用のローカルメモリを備えるモジュールを，シンプルな相互結合回路網で結合するロジックインメモリアーキテクチャを対象とし[3)~6)]，演算器数制約下で処理時間最小化を達成するアロケーション・スケジューリングを効率良く探索する手法を提案する．

従来この種の問題に対しては，整数計画法に基づく方法が用いられている[7),8)]．しかしながら，整数計画法では，最適解を求めるためには，指数関数的な計算時間を必要とする．そのため，大規模な問題への適用は困難である．大規模な問題に対して効率の良い探索を行うために，提案手法は，遺伝的アルゴリズム（GA）を用いている．GA は，生物の遺伝と進化を模擬しており，交叉・突然変異といった遺伝的操作により解を更新する探索アルゴリズムである[9)~11)]．GA では，致死遺伝子が多数発生すると探索の効率が悪くなるため，致死遺伝子を抑制することが重要となる．また，GA では大局的な探索を得意とする反面，系統的な探索を困難としている．これらの問題を解決するために，

† 八戸工業高等専門学校
Hachinohe National College of Technology
†† 東北大学大学院情報科学研究科
Graduate School of Information Sciences, Tohoku University

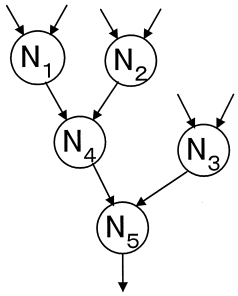


図 1 データ依存グラフ
Fig. 1 Example of a data-flow graph.

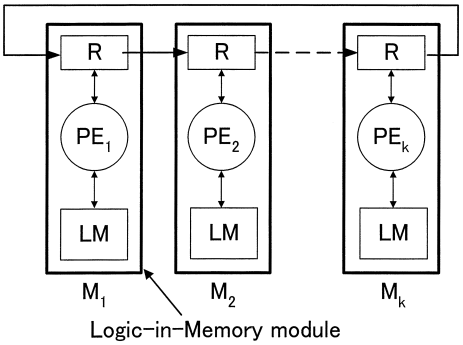


図 2 ハードウェアモデルの例
Fig. 2 Example of a hardware model.

提案手法では，ASAP（As Soon As Possible）スケジューリングを利用する GA を用いる．提案の方法は，アロケーションとスケジューリングの依存性に着目し，アロケーションのみを個体で与え，スケジューリングについてはアロケーション制約下においてデータ依存関係を満足する最も早いクロックステップを割り当てる．交叉候補の選択はスケジューリング結果に基づき，処理時間が小さい個体を選択することから，処理時間の大きい個体は淘汰される．ASAP スケジューリングを用いることにより，演算の依存関係を満たす解（致死遺伝子ではない解）であり，かつ処理時間が小さい解を系統的に得ることができる．また，アロケーションを GA を用いて行うことにより，大局的な探索を行えるという GA の性質は保持されている．

最後に，提案の方法は，他の方法に比較し，より良い解を高速に探索することを，大規模問題の具体例を通して評価をしている．

2. ロジックインメモリ構造 VLSI プロセッサの最適設計問題

2.1 処理アルゴリズム

処理アルゴリズムは**図 1** のようなデータ依存グラフ（DFG）で与えられるものとする．ノードは演算を，エッジはデータ依存関係を表すものとする．ループ構造のない DFG のみを議論の対象とする．

2.2 ハードウェアモデル

大規模な相互結合回路網は，ディープサブミクロン VLSI プロセッサの性能劣化の原因になりがちである．メモリ部と演算部の相互結合回路網の影響を少なくするために，演算器（PE）に専用のローカルメモリ（LM）を直結したモジュールを用意する．

多数のモジュールを配置する空間並列構造において，演算器間の相互結合回路網の影響を少なくするために，**図 2** のような，リング構造を設計の対象として取り上げる．R はデータ転送用のレジスタである．相互結合回路網をできるだけシンプルにするために，データ転送を図の矢印で示すように右回りに限定する．$M1$ と $M2$ のように隣接するモジュール間でのデータ転送は，1 個のみに限定し，複数のデータの同時転送はサポートしない．しかしながら，両方向のデータ転送や，隣接モジュール間での複数のデータの同時転送を可能とするモデルへの拡張は可能である．

各 PE は DFG の処理に必要な演算器（たとえば加算器や乗算器）を備えており，アロケーションを行う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．

2.3 アロケーションとスケジューリングの依存性

演算ノードの演算器への割当てをアロケーション，クロックステップへの割当てをスケジューリングと呼び，1 クロック周期を 1 クロックステップと呼ぶことにする．以下の設計の議論においては，隣接するモジュール間のデータ転送に要する時間は 1 クロックステップとし，1 個の演算ノードの演算時間は 2 クロックステップとして考えるが，演算時間については，ノードごとにその種類に応じた設定が可能である．

アロケーションとスケジューリングの依存性について考える．図 2 のリング構造を例として，図 1 のノード N_1 とノード N_4 とを，図 2 の 2 個の演算器にアロケーションを行う場合を取り上げる．2 個のノードを**図 3** (a) のように隣接する演算器にアロケーションを行う場合と，(b) のように離れた演算器に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ステップ数が異なる．このように，演算が可能なクロックステップの範囲すなわちスケジューリング範囲はアロケーションに依存する．

このことから，DFG で与えられるアルゴリズムをできるだけ最小の時間で処理をするアロケーションとスケジューリングを求めるためには，アロケーションとスケジューリングの依存性を考慮した組合せ問題を

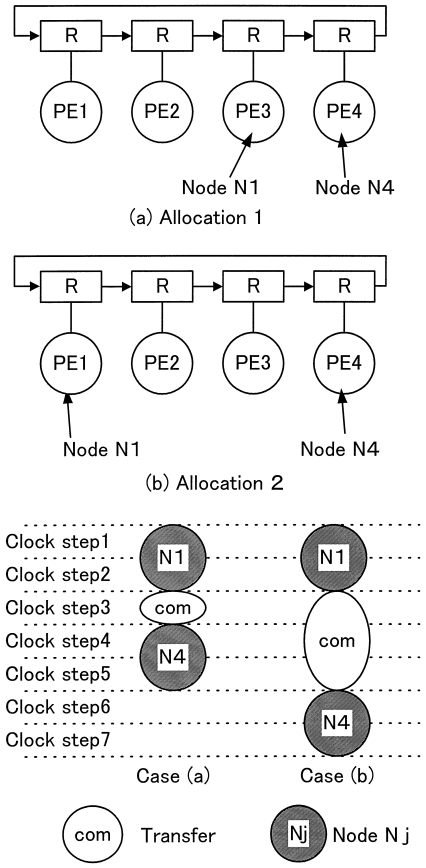


図 3 アロケーションとスケジューリングの依存性
Fig. 3 Dependence in scheduling and allocation.

解くことが必要になる．しかしながら，演算ノードが多くなるとアロケーションとスケジューリングの組合せは爆発的に多くなることから，総当たり的な探索は困難になる．

3. 遺伝的アルゴリズムを用いた高速探索

3.1 遺伝的アルゴリズムの概要

GA は，生物の遺伝と進化を模擬しており，個体の交叉，突然変異といった遺伝的操作により解を更新する探索アルゴリズムである．GA の手順は**図 4** のように示される．その概要を以下に述べる[10]．

(1) （初期個体集合の生成） ランダムに M 個の個体を生成して初期個体集合 $P(0)$ を生成し，世代 $t=0$ とする．繰返し回数 T を設定する．
(2) （評価） 個体集合 $P(t)$ 内の個体について，その適応度を計算する．
(3) （選択） 個体集合 $P(t)$ に選択演算子を適用し，$P'(t)$ を生成する．

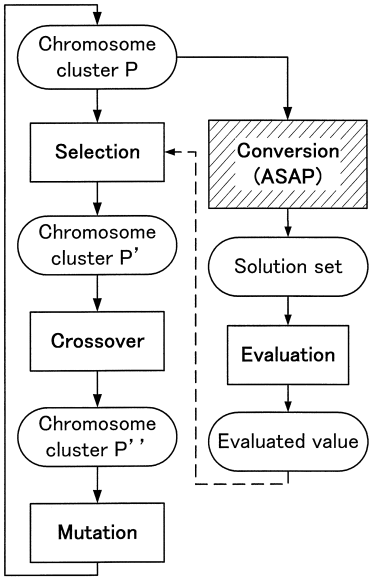


図 4 ASAP スケジューリングを導入する遺伝的アルゴリズム
Fig. 4 GA using the ASAP scheduling.

(4) （交叉） $P'(t)$ に交叉演算子を適用し，$P''(t)$ を生成する．
(5) （突然変異） $P''(t)$ に突然変異演算子を適用し，次世代の個体集合 $P(t+1)$ を生成する．
(6) （判定） $t \leq T$ ならば，$t=t+1$ として 2. へ，そうでなければ計算終了．これまでに得られた最大適応度の個体を解（準最適解）とする．

高速探索のためには，個体の設計が重要である．本稿では，つねに可能解のみを探索するために，アロケーションを個体で表し，アロケーション制約下においてデータ依存関係を満足する最も早いクロックステップのスケジューリングを行う手法を提案する．すなわち，図 4 の変換（Conversion）において，最も早いクロックステップのスケジューリングを行う．次世代の個体集合を生成するための交叉候補の選択（Selection）においては，各個体のスケジューリング結果に基づき，処理時間の短い個体を選択する．このことにより，処理時間の長い個体は淘汰される．

3.2 可能解の必要条件

可能解は物理的に実行可能な演算器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．可能解が存在するための必要条件を示す．

(1) 異なるノードが，同一のクロックステップにスケジューリングされているとき，同一の演算器にアロケーションを行わない．
(2) 演算器の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．

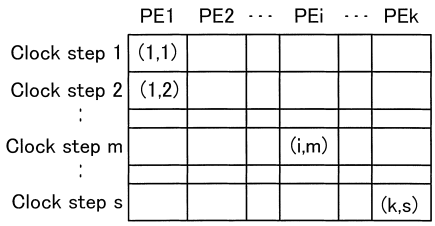


図 5 アロケーションとスケジューリングの探索空間
Fig. 5 Search area for allocation and scheduling.

(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．

(4) 異なるエッジ（データ転送）が同一のクロックステップにスケジューリングされるとき，同一のレジスタ（図 2 の R）にアロケーションを行わない．

3.3 アロケーションとスケジューリングを表す個体

アロケーションされる演算器の個数を 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ステップ 1 からクロックステップ s までにあるとし，探索空間を**図 5** のように，行方向に，PE_1 から PE_k までをとり，列方向に，クロックステップ 1 からクロックステップ s までをとって表すことにする．たとえば，要素 (1,1),(1,2) は，演算器 PE_1 のアロケーションとクロックステップ 1，2 のスケジューリングを行うことを表す．

可能解の条件を満足しない，すなわち物理的に実行不可能なアロケーションとスケジューリングを表す個体を致死遺伝子を持つ個体と呼ぶことにする．交叉により致死遺伝子を持つ個体が多数発生すると，解の質が悪くなることから，致死遺伝子を抑制することが重要となる．

致死遺伝子の発生について考える．アロケーションとスケジューリングは不可分の関係にあることから，**図 6** のように，個体によりアロケーションとスケジューリングを直接表すものと仮定すると，交叉により，致死遺伝子が発生する場合がある．図 6 の (i,j) は遺伝子を表し，i はアロケーションを，j は演算開始クロックステップを表す．致死遺伝子発生の例を示すために，図 6 の個体表現を図 5 の探索空間で表すことにし，**図 7** (a)，(b) の個体を交叉させる．ここで，(a) は図 6 の個体である．交叉により，ノード N_3，ノード N_4，ノード N_5 の遺伝子を交換するとき，(c) のように致死遺伝子を持つ個体が発生する．(c) は，ノード N_1 のクロックステップとノード N_3 のクロックステップが同じで，演算器が同じであることから，可能解の条件 (1) を満足しない．また，ノード N_4 はノード N_1 のデータを必要とするが，データ転送時間が足りないことから，可能解の条件 (2) を満足しない．致死遺伝子を持つ個体に基づくアロケーションとスケジューリングは物理的に実行不可能であることから，破棄するものとする．すると，個体群の多様性が失われることから，解の質の低下，ひいては多くの探索時間を必要とすることが予想される．

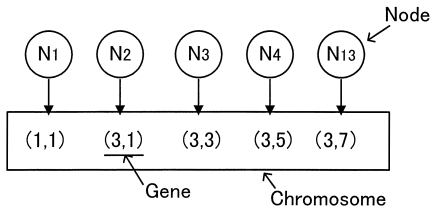


図 6 アロケーションとスケジューリングを表す個体
Fig. 6 Chromosome representing allocation and scheduling.

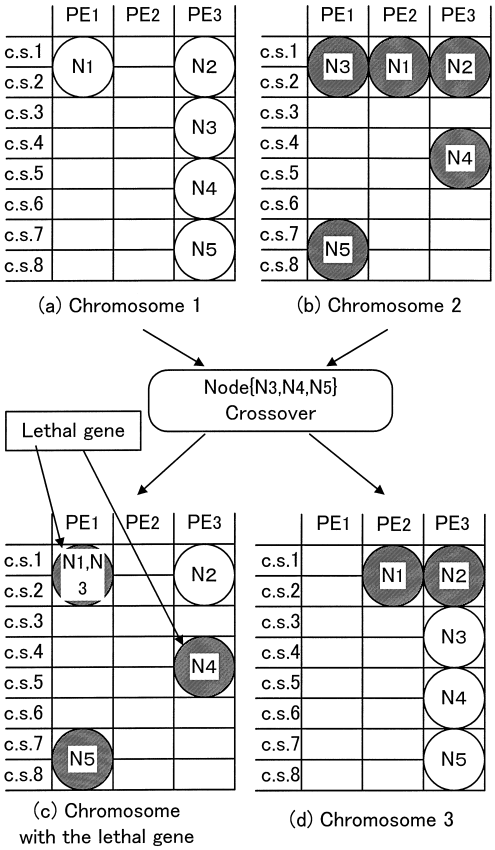


図 7 交叉における問題
Fig. 7 Problem in the crossover.

3.4 ASAP スケジューリングの導入

ASAP スケジューリングとは，データ依存関係を満たす範囲のステップの中から，最も早いステップのス

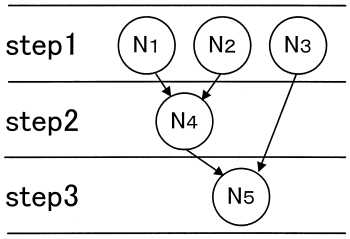


図 8 ASAP スケジューリング
Fig. 8 ASAP scheduling.

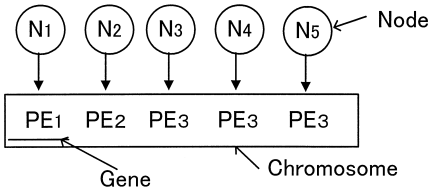


図 9 ASAP スケジューリングを導入する個体
Fig. 9 Chromosome for the ASAP scheduling.

ケジューリングを行う方法である[1)]．図 1 の DFG に対して，演算器数を制約しない ASAP スケジューリングを**図 8** に示す．すべてのノードが，データ依存関係を満足する範囲において，最も早いステップにスケジューリングされている．

本稿では，演算器数制約下における処理時間最小化を目的としていることから，スケジューリングとしては，できるだけ処理時間を短くするという観点から，アロケーション候補を与えたうえで，その制約下での ASAP スケジューリングを用いることにする．この理由は，スケジューリングを行うノードの順番により処理時間が異なるが，可能解が保証され，かつ，可能な限り処理時間が小さいスケジューリングを行う方法であるからである．この方法は，物理的に実行可能なスケジューリングをつねに行うことから，致死遺伝子の発生を抑制できる．

3.4.1 初期個体の生成と可能解

図 5 において，演算器が同じで連続する 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．

初期個体により，最終的に得られる解の質が異なると考えられる．局所解に陥らないようにするためには探索範囲を広く探索することが重要である．その一方，最適解探索をより効率化するためには，最適解が存在する可能性のある範囲を探索することが重要である．そこで，乱数に基づき生成する個体と，可能な限り処理時間が小さい 1 個の個体とで，初期個体群を構成することにする．

まず，乱数に基づく初期個体の生成と，その可能解

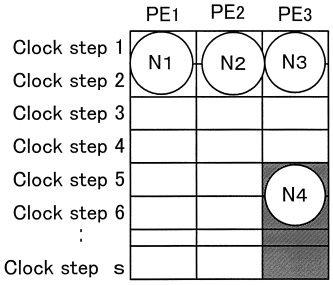


図 10 ASAP スケジューリングを用いた局所可能解
Fig. 10 Local feasible solution using the ASAP scheduling.

の求め方について述べる．初期個体を生成するために，DFG のすべてのノードのアロケーションを乱数で与える．図 1 の DFG に対する初期個体の例を**図 9** に示す．ノード N_1 を PE_1 に，ノード N_2 を PE_2 に，というようにアロケーションが行われる．

可能解を求めるために，個体制約下において ASAP スケジューリングを行う．ノード N_j の添え字 j をノード番号 j と呼ぶことにする．DFG のノードには半順序関係を満たすようにあらかじめノード番号が与えられているものとし，ノード番号の順に，ASAP スケジューリングを行うことにより，可能解を求めることにする．

DFG の一部のノードのアロケーションとスケジューリングの対を局所可能解と呼ぶことにする．図 9 を例として，可能解を求める．ノード N_1 までの局所可能解を求める．ノード N_1 は，他のノードの出力データを必要としないので，**図 10** において，PE_1 の列の最早ステップであるクロックステップ {1, 2} のスケジューリングを行う．ノード $\{N_1, N_2\}$ の局所可能解を求める．ノード N_2 はノード N_1 からの出力データを必要としない．よって，アロケーションの制約下において最早ステップを持つ 1 個の 2 連接要素を割り当てる．図 10 の記号 N_1，N_2，N_3 はノード $\{N_1, N_2, N_3\}$ の局所可能解を表す．

ノード $\{N_1, N_2, N_3, N_4\}$ の局所可能解を求めるためには，ノード $\{N_1, N_2\}$ とのデータ依存関係を満足できる最早ステップのスケジューリングが必要である．ノード N_4 は PE_3 にアロケーションされており，ノード $\{N_1, N_2\}$ からのデータ転送時間が必要であることから，データ依存関係を満たすスケジューリング可能な範囲は図 10 の斜線の部分になる．ASAP スケジューリングから 2 連接要素（3,5),（3,6）を割り当てる．ノード $\{N_1, N_2, N_3, N_4, N_5\}$ の局所可能解すなわち可能解を**図 11** に示す．(N_i) はノード N_i からの

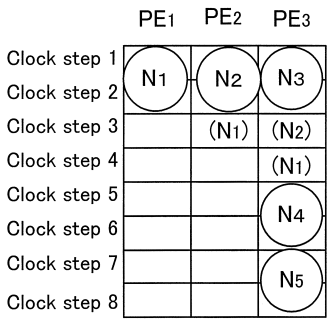
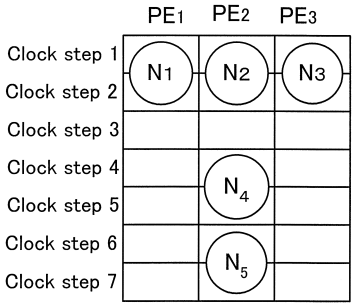
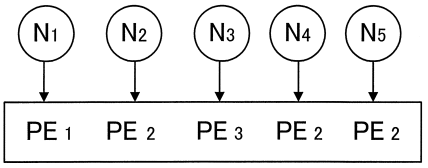


図 11 可能解
Fig. 11 Feasible solution.



(a) ASAP without the allocation constraint



(b) Chromosome for shortening the processing time

図 12 処理時間が小さい個体
Fig. 12 Chromosome for shortening the processing time.

転送を表す.

n 個（$n \leqq m-1$）のノードに対する局所可能解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_{n+1} とに対する局所可能解の求め方を一般化する．DFG のノード数を m とし，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_{n+1} とデータ依存関係がない場合，ks 個の要素から n 個のノードに割り当てた $2n$ 個の要素を除く，$ks-2n$ 個の要素の中の 2 連接要素の中から，アロケーション制約下における ASAP スケジューリングを行う 2 連接要素をノード N_{n+1} に割り当てる.

ノード集合に含まれるノード N_j と，ノード N_{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_{n+1} がノード N_j の出力データを入力できる 2 連接要素の中から，アロケーション制約下における ASAP スケジューリングを行う 2 連接要素をノード N_{n+1} に割り当てる.

n 個のノードとノード N_{n+1} とに対する局所可能解は，n 個のノードに割り当てる 2 連接要素と，ノード N_{n+1} に割り当てる 2 連接要素で表すことができる．m 個のノードの局所可能解が可能解になる.

次に，ASAP スケジューリングは可能な限り処理時間が小さいスケジューリングを行うことに着目して，可能な限り処理時間が小さい初期個体の生成について述べる．アロケーションとスケジューリングを自由度とし，ノード番号順に ASAP スケジューリングを行った結果から，初期個体を求めることにする．スケジューリング可能なクロックステップが複数存在する場合は，演算器番号が若い 2 連接要素を割り当てるものとする．図 1 の DFG を例として，可能な限り処理時間が小さい初期個体を求める．演算器数は 3 個とする．アロケーションとスケジューリングの探索空間を**図 12** (a) に示す．ノード N_1 に最早ステップのスケジューリングを行うとき，アロケーション可能な演算器が複数存在することから，2 連接要素（1,1）(1,2) を割り当てる．同様に，ノード N_2 には，2 連接要素 (2,1),(2,2)，ノード N_3 には，2 連接要素 (3,1),(3,2) を割り当てる．ノード N_4 には，ノード N_1 とノード N_2 のデータを入力できる最も早い 2 連接要素である (2,4),(2,5) を割り当てる．ノード N_5 には，ノード N_3 とノード N_4 のデータを入力できる最も早い 2 連接要素である (3,7),(3,8) を割り当てる.

処理時間が小さい初期個体は，(a) から，各ノードのアロケーションのみを取り出すことにより (b) のように求められる.

3.4.2 交叉候補の選択と交叉

適応度の評価基準を満足する可能解が個体群から求められない場合は，交叉をする．適応度には目的関数である処理時間を用いる．個体ごとに可能解から算出した処理時間を記憶しておく．交叉候補の選択には，個体群の中から任意に取り出した 2 個の個体のうち，処理時間の短い個体を候補とするトーナメント方式を用いる．交叉は，候補の中から，任意の 2 個の個体を取り出し，一様交叉を行う．一様交叉は任意個の交叉点をとれる交叉法であり，探索の効率が良い方法であるとされている[11]．交叉後の個体から可能解を得る手

順は，初期個体から可能解を得る手順に同じである．

4. 実　　験

4.1 評価用の DFG

評価用の DFG として，**図 13** に示す EW，HAL を用いる．EW はデータ依存関係の制約が多く，HAL は，アロケーションとスケジューリングの自由度が大きい．

上記のベンチマーク（EW，HAL）を大規模問題へと拡張するためには，複数の EW，あるいは HAL を用いる．アロケーションとスケジューリングの自由度が大きいほど，大規模問題にすることができる．複数の DFG を独立させる方が，アロケーションとスケジューリングの自由度が大きい．そこで，**図 14** のように互いの subDFG を接続しないで用いる．subDFG は 1 個の HAL あるいは EW を表す．たとえば，EW5 の DFG は 5 個の EW により構成され，ノード数は 170 個である．同様に，HAL19 は，19 個の HAL の DFG により構成され，ノード数は 209 個である．

4.2 比較の対象

多くの高位合成の方法が提案されているが，この問題に直接適用できる方法は報告されていない[7)~9),12),13)]．そこで，整数計画法に基づく解法を考案しこの問題に適用してみたが，EW5，HAL19 について，72 時間の探索を行ったが，可能解を得ていない．計算時間を 24 時間に制限し，EW5，HAL19 以下の規模の各問題を対象とし探索を行った結果，HAL のみについて最適解を探索できた．その処理時間は 9 クロックを得た．HAL2 や HAL3，あるいは EW や EW2 などのように，ノードが多い DFG については，最適解を探索できていないだけでなく，実行可能解をも探索できていない．

そこで，探索途中の可能解の処理時間の下限に着目し，可能解が存在する可能性のある範囲のみを探索するという分枝限定法に基づく解法を考案し，適用した．1 個のノードのアロケーションとスケジューリングを 1 個の木のノードとする探索木を作る．木のレベルとノード番号を対応させ，ノード N_n を割当て可能な 2 連接要素をレベル n の木のノードとして列挙することにより，深さ優先探索に基づく最適解の探索を可能にする．ここで，最適解が存在する可能性のある探索空間のみを探索するために，可能解の処理時間の下限と，暫定的に見つかっている最良解の処理時間（クロックステップ数）とを比較することにより，探索空間を限定する．レベル n の木のノードを選択するとき，それから分岐する可能解が持つ処理時間の下限を

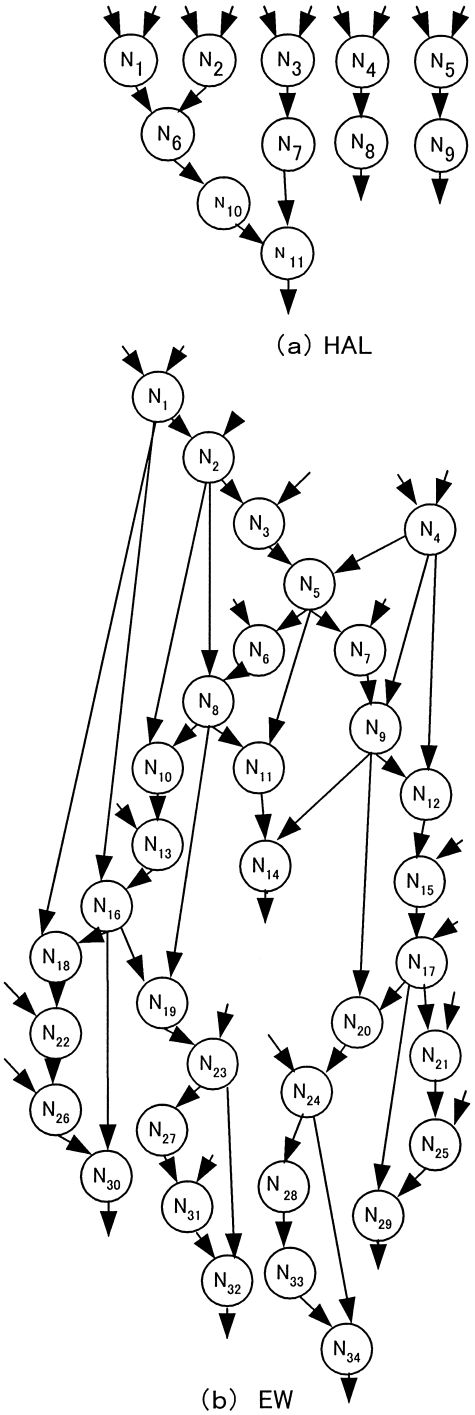


図 13　評価に用いた DFG
Fig. 13　DFG for the evaluation.

D として，式 (1) を考える．

$$D = s_n + t_n + d_n - 1 \qquad (1)$$

s_n はノード N_n のスケジューリングを行ったクロックステップ，t_n はノード N_n の演算に必要なクロッ

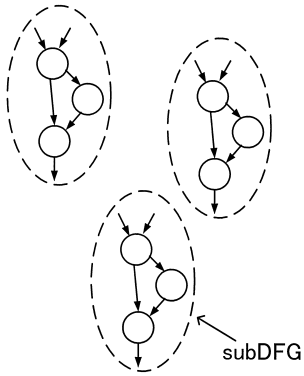


図 14 大規模な DFG
Fig. 14 DFG with many nodes.

クステップ数を表す．これらは既知である．DFG の最終出力ノードが 1 個以上ある場合，d_n は，ノード N_n が出力を発生してから，DFG のすべての最終出力ノードが出力を発生するまでの処理に必要な時間の下限である．d_n について考える．DFG のノード N_n の出力が，DFG の 1 個以上の最終出力ノードに伝搬される場合，それらのパスのうち，最も伝搬遅延の大きいパスをノード N_n からのクリティカルパスと呼ぶことにする．ノード N_n からのクリティカルパスに含まれるすべてのノードに，同じ PE のアロケーションをするとき，転送を必要としないので，転送時間は 0 である．よって，d_n を求めるためには，ノード N_n からのクリティカルパスに含まれるノードのデータ依存関係を満足する演算時間のみの下限を求めればよい．これは，ノード N_n からのクリティカルパスに含まれるノードに対する ASAP スケジューリングから求めることができる．

さらに，探索空間の限定のために，処理時間が小さい解を最初に探索することが重要である．そこで，ASAP スケジューリングを用いた処理時間が小さい解の探索を行う．この方法は，GA において処理時間が小さい初期個体を求めた手法に同じである．図 12 の (a) がその例である．すなわち，提案手法においても，比較対象である分枝限定法においても，処理時間が小さい解の探索を最初に行う．

4.3 評　　価

4.3.1 解の精度

解の精度について評価を行うために，最適解の探索を行う．分枝限定法を用いても，DFG のノードが多くなると，組合せの爆発は避けられない．複数の HAL を用いるとき，あるいは複数の EW を用いるとき，24 時間の探索を行っても最適解を探索できない．このことから，実用的な計算時間内で最適解を探索するために，評価用の DFG として，EW と HAL とを，各々1 個用い，モジュール数が 4 個の場合について評価を行う．最適解の処理時間は，EW の場合，29 クロック，HAL の場合，9 クロックである．最適解と，提案手法で求めた解の処理時間をまとめて**表 1** に示す．各々について 5 回の試行を行った．EW の場合，5 回とも最適解を探索し，HAL では最適解（9 クロック）を 3 回探索できた．最適解を探索できた理由は，初期個体群の中に 1 個の処理時間が小さい初期個体を用意しておいたことによるものと考えられる．HAL の場合，この個体の処理時間は 10 クロックである．この個体を含む個体群を交叉対象としていることから，提案手法で得られる解の処理時間は 10 クロックより少なくなることはあっても，多くなることはない．すなわち，提案手法は，処理時間が小さい初期個体で得られる解の処理時間より質の良い解を得ることを保証する．また，EW の場合，すべての試行において第 1 世代において最適解を探索できていることから，処理時間が小さい初期個体の解が最適解であったものと考えられる．

表 1 最適解との比較
Table 1 Comparison with the optimum solution.

DFG	EW	HAL
Optimum solution	29	9
Solution by the proposed method	29	9, 10

分枝限定法の探索を途中で打ち切っても，それまでの暫定的な最良解を得ることができる．大規模問題（EW5，HAL19）に対して，限定された計算時間内で求めた解の質の評価を行う．ハードウェアモデルのモジュール数を 10 個とする．遺伝的アルゴリズムの場合，計算時間と解の質は，個体数と最大世代数によって異なりがちである．そこで，最適解を探索する可能性の高い個体数と，解が飽和する最大世代を予備実験で求めておき，1 世代の個体数は 20 とし，最大世代数は 100 にする．最大世代に達するために必要な計算時間は EW5 の場合も HAL19 の場合も約 5 分である．そこで，分枝限定法を用いた探索も 5 分で打ち切ることにする．

計算時間を 5 分に制限したときの，HAL19 と EW5 の処理時間を**表 2** に示す．提案手法の解は，5 回の試行結果に対する解の範囲を示したものである．分枝限定法の場合，HAL19 も EW5 も，最初に探索された可能解が更新されないまま 5 分間経過し，HAL19 の場合，51 クロック，EW5 の場合，69 クロックである．提案手法で求めた解の処理時間は，HAL19 の場

表 2 探索時間一定条件下における比較
Table 2 Comparison under the same search time.

DFG	EW5	HAL19
Branch and bound method	69	51
Proposed method	48〜50	50, 51

合，50，51 クロックであり，分枝限定法とほぼ同等の質の解を探索している．EW5 の場合は，提案手法の解の処理時間が，分枝限定法の解の処理時間に比較し，70%程度に短縮できている．これらのことから，提案手法は質の良い解を探索しているといえる．

4.3.2 計算時間の比較

HAL の最適解の探索時間は，分枝限定法を用いた場合，約 2 分を要したことに対して，提案手法の場合，5 回の試行のうち，最短で 6 秒程度で探索できた．EW について，分枝限定法を用いた場合，約 1 分で最適解を探索できた．提案手法では初期個体が最適解になることから，5 回とも約 1 秒であった．提案方法が高速に最適解の探索を行っているといえる．

次に，大規模問題の計算時間の評価を行う．表 2 の EW5 の場合，分枝限定法を用いる場合，処理時間が 69 ステップの可能解の探索のために，5 分を要している．提案手法を用いると，制限時間内（約 4 分）に処理時間が少ない解を探索している．このことから，処理時間を制約条件とする探索を行うとき，提案手法がより少ない計算時間での探索を可能にするものと推測できる．

処理時間が少ない解を高速に探索することに対して，GA に ASAP スケジューリングを導入することの効果を評価する．GA に ASAP スケジューリングを導入する場合を▲で表し，導入しない場合すなわちアロケーションとスケジューリングを個体で直接表す場合を□として，世代ごとの処理時間の推移の典型的な例を**図 15** に示す．(a) は EW5，(b) は HAL19 である．EW5 において，ASAP スケジューリング導入の効果が大きい．この理由は，EW が HAL よりも，データ依存関係の条件が多いことに対して，ASAP スケジューリングにより転送時間を少なくできたためと考えられる．

実験には，Cerelon400 MHz のパソコンを用いた．

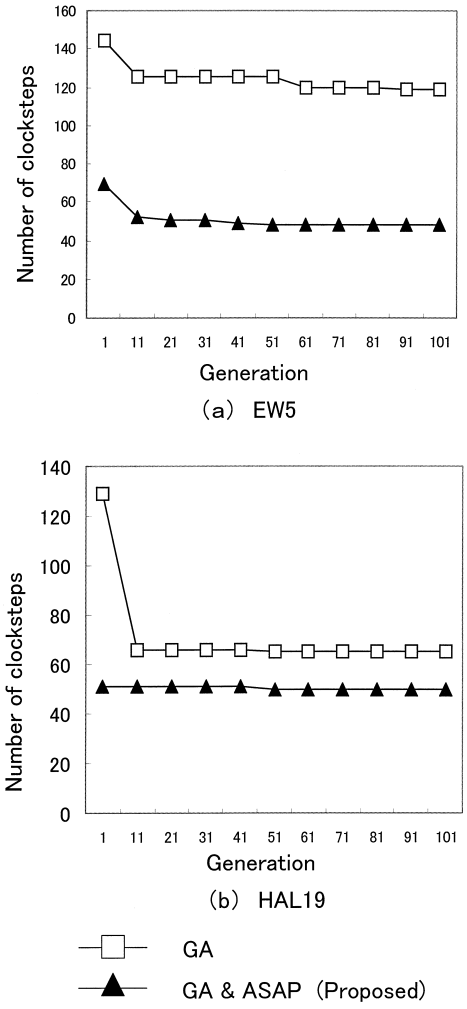


図 15 ASAP 導入の効果
Fig. 15 Effect of the ASAP scheduling.

5. むすび

微細化技術の進展にともない，配線に起因する諸問題がディープサブミクロン VLSI，特に近年のシステム VLSI の構成において，性能向上のボトルネックになっている．アーキテクチャレベルにおいて配線問題を軽減するために，ロジックインメモリ構造 VLSI プロセッサのハードウェアモデルを設計の対象とするハイレベルシンセシスについて述べた．

演算器数制約下において，DFG で与えられるアルゴリズムの処理時間最小化問題を，ASAP スケジューリングを利用した遺伝的アルゴリズムを用いることで，高速に探索をする方法を提案した．

専用 VLSI プロセッサ設計の効率化のためには，設計の初期段階において，与えられるアルゴリズムを最高性能で処理をするハードウェアモデルの選択を可能とすることが切望される．提案する設計法は，データ転送の最小離散時間を 1 クロックステップとして表すことで，たとえばモジュールをバスで結合するモデルや完全網で結合するモデルなどのように，モジュール

間の転送を必要とするロジックインメモリ構造 VLSI プロセッサのハードウェアモデルに拡張可能である．このことにより，与えられるアルゴリズムを，できる限り最小の時間で処理をするハードウェアモデルを自動選択する問題に発展できることはもちろんのこと，アロケーションとスケジューリングが依存する一般の VLSI プロセッサの最適設計問題にも拡張可能である．

VLSI プロセッサ内部の配線問題をアーキテクチャレベルから軽減するハイレベルシンセシスの考え方は，今後ますます重要になるものと考える．

参 考 文 献

1) Gajski, D., Dutt, N., Wu, A. and Lin, S.: *HIGH-LEVEL SYNTHESIS Introduction to Chip and System Design*, p.359, Kluwer Academic Publishers (1992).
2) 桜井貴康：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 (1998).
3) Kautz, W.H.: Cellular Logic-in-Memory Array, *IEEE Trans. Computers*, Vol.18, No.8, pp.719–727 (1969).
4) Hanyu, T. and Kameyama, M.: Multiple-Valued Logic-in-Memory VLSI Architecture Based on Floating-Gate-MOS-Pass-Transistor Logic, *IEICE Trans. Electron*, E82-C9, pp.1662–1668 (1999).
5) 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, pp.1009–1018 (2000).
6) 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誌，Vol.J84-D-1, No.6, pp.531–539 (2001).
7) Hwang, C.-T., Lee, J.-H. and Hsu, Y.-C.: A Formal Approach to the Scheduling Problem in High Level Synthesis, *IEEE Trans. computer-Aided design*, Vol.10, No.4, pp.465–475 (2002).
8) Gebotys, C.H. and Elmasry, M.I.: Optimal VLSI Architectural Synthesis Area, Performance and Testability, pp.98-140, Kluwer Academic Publishers (1992).
9) 大森健児：遺伝的アルゴリズムによる高レベル合成，電子情報通信学会論文誌，Vol.J81-A, No.5, pp.854–862 (1998).
10) 電気学会（編）：遺伝アルゴリズムとニューラルネット，p.272, コロナ社 (1998).
11) 伊庭斉志：遺伝的アルゴリズムの基礎，p.254, オーム社 (2000).
12) 石川淳士：ASIC 技術の基礎と応用，論理合成技術，第 5 章，今井正治（編），pp.74–87, 電子情報通信学会 (1994).
13) 柄澤匡彦，張山昌論，亀山充隆：分枝限定法に基づく VLSI プロセッサのハイレベルシンセシス，電気関係学会東北支部連合大会，1D–17, p.129 (2002).

(平成 14 年 10 月 16 日受付)
(平成 15 年 3 月 4 日採録)

工藤 隆男（正会員）

1976 年岩手大学工学部電子工学科卒業．同年八戸工業高等専門学校助手，1990 年同助教授，現在に至る．リアルワールド応用 VLSI プロセッサの研究に従事．

張山 昌論（正会員）

1992 年東北大学工学部電子工学科卒業．1997 年同大学大学院博士課程修了．現在，同大学助手．知能集積システムに関する研究に従事．

亀山 充隆（正会員）

1973 年東北大学工学部電子工学科卒業．1978 年同大学大学院博士課程修了．同年同大学工学部助手，1981 年同助教授，1991 年同教授，1993 年同大学大学院情報科学研究科教授，現在に至る．リアルワールド応用知能集積システム，ロボットエレクトロニクスシステム，VLSI プロセッサのハイレベルシンセシス，多値集積システム等の研究に従事．IEEE 多値論理国際シンポジウム優秀論文賞（1984 年〜1989 年，計 5 回），1986 年度計測自動制御学会技術賞，1990 年度電子情報通信学会論文賞，1990 年日本ロボット学会技術賞等を受賞．IEEE Fellow．