交差手法の適応的選択機能を組み込んだ遺伝的アルゴリズムの LSI チップによる実現

若林真一[†] 小出哲士[††] 八田浩一[†]
中山喜勝[†] 後藤睦明[†] 利根直佳[†]

複雑な制約を持つ大規模最適化問題を解く手法の 1 つとして遺伝的アルゴリズム（GA）が知られている．GA は数理計画法などの通常の最適化手法が適用困難な問題にも適用できるという利点を持つ反面，多くのパラメータを持つため，それらの値を調整して GA の探索能力を最大限に引き出すことは難しいという問題点がある．そのためパラメータ値を GA 実行中に動的に調整し，効率的に GA を実行する方法が数多く提案されている．一方，GA は一般に多大な計算時間を必要とするため，GA をハードウェアで実現することが研究されている．本論文では著者らが提案したエリート度に基づく交差手法の適応的選択機能を組み込んだ GA の LSI 化について報告する．また，LSI の性能評価のためのシミュレーション実験の結果についても述べる．

An LSI Implementation of a Genetic Algorithm with Adaptive Selection of Crossover Operators

Shin'ichi Wakabayashi,[†] Tetsushi Koide,[††] Koichi Hatta,[†]
Yoshikatsu Nakayama,[†] Mutsuaki Goto[†]
and Naoyoshi Toshine[†]

Genetic Algorithms (GAs) have been widely used to solve large-scaled optimization problems with complex constraints. Since GAs have many parameters, it is difficult to set these parameters to appropriate values to obtain good solutions. Therefore, many parameter setting methods have been proposed, in which parameter values were adaptively changed during the execution of a GA. On the other hand, GAs generally require a large amount of computation time, and to solve this problem, many research results for hardware implementation of a GA have been reported. This paper presents an LSI implementation of a GA, which selects crossover operators adaptively during the algorithm execution based on a new measure called "elite degree," that we have proposed to estimate potential superiority of an individual. Simulation results to evaluate the LSI chip are also given.

1. はじめに

工学における様々な分野において複雑な制約を持つ数多くの大規模最適化問題が知られているが，それらの問題の多くは数理的手法や組合せ最適化アルゴリズムを用いて実用的な計算時間で最適解を求めることは困難なため，ヒューリスティック手法を用いて解を求めることが一般的である．このようなヒューリスティック手法として遺伝的アルゴリズム（Genetic Algorithm, GA）が優れた手法の 1 つとして知られている[3),8)]．GA は生物の進化の過程にヒントを得た探索手法であり，アルゴリズムの実行中は複数の解（個体）を保持し，これに交差，突然変異などの遺伝オペレータを適用して解の改善を行う．

しかしながら，GA にはいくつかの問題点があることが知られている．主要な問題点として，(1) GA は多くの遺伝オペレータと遺伝パラメータを持つが，GA を問題に適用する前にあらかじめそれらを適切に調整することは困難である，(2) 大規模問題に対して多大な計算時間が必要である，という 2 点があり，これらの問題点に対処するために多くの研究が行われてきている．まず (1) については，パラメータ値を最適に設定するための手法の研究や，アルゴリズムの実行中にパ

† 広島大学工学部
Faculty of Engineering, Hiroshima University
†† 東京大学大規模集積システム設計教育研究センター
VLSI Design and Education Center, The University of Tokyo

ラメータ値や遺伝オペレータの種類を適応的に調整する手法の研究が行われている[1),2),4),12)]．一方，(2) については，パラメータ値の適応的調整などにより解の精度を保ったまま，優良解への収束を早める手法が研究されているほか，GAを並列化あるいはハードウェア化することにより通常の逐次処理コンピュータ上でのGAの実行と比較して，大幅な計算時間の短縮を実現しようとする研究も行われている[5),9),10),14)]．

本研究では，GAが持つ上記の2つの問題点に対する1つの解決手法として，適応的GAのハードウェア化について研究し，適応的GAを実現する専用ハードウェアをLSIチップとして実現する．GAは汎用的な探索アルゴリズムであるため，GAのハードウェア化にあたっては，得られるパフォーマンスと汎用性の確保を考慮する必要がある．たとえば，特定の問題に特化して専用LSIとしてGAをハードウェア化すれば高いパフォーマンスが得られるが，汎用性は失われる．一方，汎用性を確保しながらハードウェア化を実現しようとすれば，FPGA（Field Programmable Gate Array）などのプログラム可能論理デバイスで実現することが考えられるが，一般にFPGAの動作周波数は専用LSIと比較すると低い．そこで，本研究では専用LSIとFPGAによる実現のトレードオフを考慮し，個別の問題に依存する個体の適応度評価についてはFPGAもしくはソフトウェアで実現することとし，それ以外の部分を専用ハードウェアとしてLSI化することとした．また，LSI化するGAの機能についてもできるだけユーザが外部より設定可能にした．このような考えに基づいて本研究で試作したLSIチップを以下ではGAA（Genetic Algorithm Accelerator）と呼ぶ[13)]．GAAは著者らが文献6)において提案した適応的遺伝的アルゴリズムに基づいて設計されており，ハードウェア化にあたってはハードウェアの特性を考慮して元のアルゴリズムを一部変更している．試作チップを評価した結果，ソフトウェアによるGAと同等の解を高速に求めることが可能であることが示され，適応的GAをハードウェア化する有効性が確認された．

本論文の構成は以下のようになっている．2章では遺伝的アルゴリズムについて簡単に述べた後，GAAが実現している適応的GAについて説明する．3章ではGAAのハードウェア概要を述べる．4章ではGAAの評価について述べる．最後に5章で本論文をまとめるとともに，今後の課題について述べる．

2. 準　　備

2.1 遺伝的アルゴリズム

遺伝的アルゴリズム（GA）の概要を簡単に示す．GAの詳細は文献2)，3)，8)を参照されたい．GAにおいては，解こうとする最適化問題に対し，その許容解を適当なコーディングを採用することにより記号列で表現する．この記号列を染色体もしくは個体と呼ぶ．GAではあらかじめ決められた数の個体の集合（個体群）をつねに保持している．各個体に対しては，最適化問題の目的関数に基づいて，その個体がどの程度良い個体であるかを数値で表す適応度が定義される．

現在の世代の個体群に対し，選択，交差，突然変異などの操作（これらを遺伝オペレータと呼ぶ）を適用して次の世代の個体群の候補となる個体集合を生成する．生成された個体集合に対し，それらの個体の適応度を評価し，どの個体を次世代の個体として残すかを決める．以上の操作をあらかじめ与えられた終了条件を満たすまで繰り返す．

2.2 適応的遺伝的アルゴリズム

与えられた最適化問題のインスタンスに対してGAの探索能力を最大限に引き出すためには，交差や突然変異などの遺伝オペレータの適切な選択とオペレータ適用確率等の遺伝パラメータ値の適切な設定が必要である．従来のパラメータ調整では，前もって遺伝オペレータを選択し，遺伝パラメータについては経験と勘に基づいて値を絞り込んだ後で試行実験を繰り返すことにより微調整を行っていた．しかし，これには多大な時間を必要とし非常に労力がかかるうえに，適切な遺伝オペレータの選択やパラメータ値の設定が難しいという問題点があった．そこで，個体に応じた遺伝オペレータの選択[12)]，GA実行中での遺伝オペレータの適用確率の動的な変更[1)]など，遺伝オペレータの種類や遺伝パラメータの値を初期値のままで固定しておくのではなくGAの実行中にそれらを動的に変更し，適応させていく適応的GAが近年数多く提案されている．

そのような適応的GAの1つとして著者らはエリート度という概念を導入し，GA実行中に，交差の対象となる個体対ごとに適応的に交差手法を選択する適応的GAを提案している[6)]．ここでエリート度とは個体の潜在的な優劣の度合いを示す指標であり，最大化問題に対し，世代 T における個体 x_i^T のエリート度 $ED(x_i^T)$ は以下の式で定義される[6)]．

$$ED(x_i^T) = \frac{\sum_{j=1}^{l\_max}\left\{|Elite_i^T(j)| \times \beta^j\right\}}{\sum_{j=1}^{l\_max}\left\{|Anc_i^T(j)| \times \beta^j\right\}} \quad (1)$$

ここで，$Anc_i^T(j)$ は個体 x_i^T の世代 $T-j$ における先祖の集合，$Elite_i^T(j)$ は，$Anc_i^T(j)$ の中でエリートである個体の集合，β $(0 \leq \beta \leq 1)$ はエリート影響度係数である．β の値を小さくすることにより遠い祖先の影響を減少させることができる．最小化問題に対するエリート度も同様に定義できる．

ある個体がある世代においてエリートであるかどうかの判定については，最大化問題においてはその世代の中の個体の適応度の分布を正規分布と仮定して，世代 T の個体の適応度の平均を μ_T，適応度の標準偏差を σ_T とし，個体 x_i^T の適応度を $f(x_i^T)$ とするとき，$f(x_i^T)$ が $\mu_T + \alpha \times \sigma_T$ 以上の個体をエリートと定義する．ただし，α は非負の実数である．すなわち，エリート集合 $Elite_i^T(j)$ は以下のように定義される．

$$\begin{aligned} &Elite_i^T(j) \\ &= \{x_k^{T-j} \mid x_k^{T-j} \in Anc_i^T(j), \\ &\quad \mu_{T-j} + \alpha \times \sigma_{T-j} \leq f(x_k^{T-j})\} \end{aligned} \quad (2)$$

このエリート度を用いて，文献 6) では以下の操作により交差手法を交差の対象となった個体対ごとに適応的に選択することを提案している．

[交差手法の適応的選択]

ステップ 1： 交差を行う 2 つの親についてエリート度の和を求める．

ステップ 2： 2 つの親がエリートかどうかを判定し，もしエリートならばエリート度の和に 1（一方の親のみがエリートの場合），もしくは 2（両方の親がエリートの場合）を加える．

ステップ 3： ステップ 2 で求めたエリート度の和が一定のしきい値以上ならば 2 点交差を実行し，そうでなければ一様交差を実行する．

エリート度を導入した GA は，交差手法をエリート度に基づいて適応的に選択すること以外は，通常の世代を持つ GA（generational GA）と同じ動作となっている．エリート度を用いた適応的 GA は通常の GA と比較して，より早い世代で良い解に収束することが示されている[6)]．

2.3 GA のハードウェア化における留意点

著者らは 1 章で述べた GA の 2 つの問題点に対処するために，エリート度を導入した GA のハードウェア化を行うこととした．ハードウェア化にあたって留意した点を以下にまとめる．

まず，ハードウェア設計上の制約について述べる．ハードウェア設計における最も大きな制約はハードウェア資源の制約である．今回の設計では適応的 GA を約 1 万ゲート規模のスタンダードセルチップとして実現することを前提としたため，多量のハードウェア資源を必要とする回路は採用することができない．このため，後述するように，元の適応的 GA のアルゴリズムの一部を変更し，ハードウェア量の削減を図った．また，ハードウェア化により，GA としての機能はハードウェア設計時の仕様に固定されるので，GA 実行時にユーザがなるべく多くのパラメータ値を設定可能とするとともに，問題に依存する個体の適応度の評価については LSI 化は行わず，外部回路もしくはソフトウェアで行うこととした．

次に GA をハードウェア化するうえで最大限利用すべきハードウェアの利点をまとめる．まず，ソフトウェアと異なり，ハードウェアの動作は本質的に並列動作であり，並列処理やパイプライン処理が自然に導入できる．たとえば，ソフトウェアでは実現の難しいビットレベルの並列処理も簡単に実現できる．また，一般に同じ機能を実現する複数の回路構成が存在するので，面積（ゲート数）と処理時間のトレードオフを考慮して，最適な回路を採用する必要がある．メモリを用いたテーブルルックアップによる計算も可能であり，有効に用いれば回路規模の削減に貢献する．

3. 適応的遺伝的アルゴリズムのハードウェア化

3.1 提案ハードウェアの概要

提案ハードウェア GAA は筆者らが文献 6) で提案したエリート度に基づく適応的 GA を元に，いくつかの変更を加えたうえで LSI チップとしてハードウェア化したものである．GAA の主な仕様を**表 1** に示す．

GAA の全体構成を**図 1** に示す．GAA システムは大きく分けて以下の 3 つから構成されている．

(1) GAA チップ本体

システムのメインモジュールであり，個体の適応度

表 1 GAA チップの仕様
Table 1 Specifications of the GAA chip.

個体数	64, 128
交差手法	2 点交差，一様交差，適応的交差
実行世代数	512, 1024, 2048, 4096
交差確率	0/256〜255/256
突然変異確率	0/4096〜255/4096
選択	ルーレット選択 + エリート戦略
エリート度のしきい値	0.0〜4.0
個体情報のビット数	64 ビット
エリート度	16 ビット
適応度	16 ビット

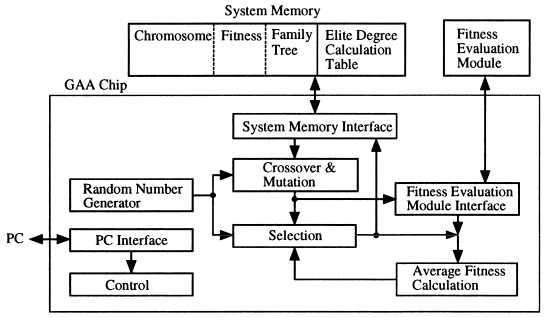

図 1 GAA の構成
Fig. 1 Overview of the GAA.

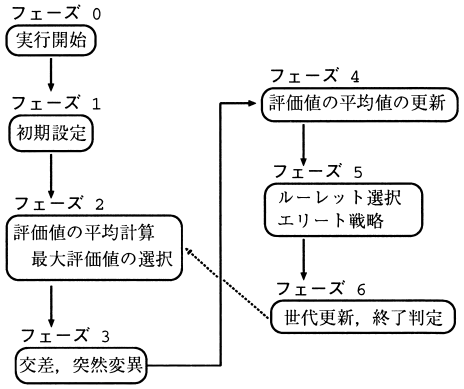

図 2 GAA の実行フロー
Fig. 2 Execution flow of GAA.

評価を除いて，適応的 GA の動作のすべてをこのチップが行う．GAA チップ内部はいくつかのサブモジュールから構成されている．サブモジュールの詳細は次節で述べる．また，ハードウェアのデバッグを容易にするために，GAA の状態レジスタや一部のレジスタについては，コマンド設定により実行中に外部から観測できるようになっている．フェーズごとに実行の中断と再開も可能である．

(2) システムメモリ

GAA には個体に関するデータを保存するためのシステムメモリを持つ．システムメモリは 32 K ワード（1 ワード = 16 ビット）の SRAM で構成されている．各個体に対する個体データは染色体情報（4 ワード），適応度（1 ワード），家系図（1 ワード）の 6 ワードから構成されている．個体データの初期値は外部 CPU から設定する．染色体情報は問題の許容解を 64 ビットの 2 進表現でコード化したものである．また，外部適応度評価回路で計算された個体の適応度も格納される．家系図情報には，各個体に対し，4 世代前までの祖先中のエリート個体の数がそれぞれの世代ごとに格納される．個体を交差するごとに家系図情報も，それぞれの親の家系図情報から加算器を用いて対応する先祖の個体数どうしを加算することにより更新される．

また，このシステムメモリには，エリート度の計算をハードウェアで効率良く行うために，各世代の個体がエリートかどうかのすべての組合せに対してエリート度をあらかじめ計算したテーブルも格納している．本来はエリート度を求めるためには式 (1) を計算する必要がある．しかし，式 (1) は乗算と徐算を含み，これをそのままハードウェアで実現するとハードウェアコストが大きい．ここで，エリート度の計算において 4 世代前までのエリートの数しか考慮しないことにすれば（すなわち，式 (1) の l_max を 4 とする），エリート個体の数の組合せはたかだか $17 \times 9 \times 5 \times 3$ 通りしかないことになり，アドレスが 14 ビットのテーブル検索でエリート度計算を実現できる．

(3) 適応度評価回路

個体の適応度は GA が解く問題に依存するため，GAA では個体の適応度の評価回路は外部回路として構成している．この適応度評価回路は FPGA（Field Programmable Gate Array）や外部 CPU を用いて実現することを想定しており，GAA チップとの並列動作が可能となっている．詳細は後述する．

GAA チップとその周辺回路は A4 サイズのプリント基板上に組み立てられており，パーソナルコンピュータ（PC/AT 互換機）の ISA バスに接続されるようになっている．GAA の実行の制御はパーソナルコンピュータの CPU が行う．

3.2 ハードウェアの詳細

GAA の動作の流れを**図 2** に示す．図 1 に示したように，GAA チップは以下のサブモジュールで構成され，図 2 の一連の動作を実現している．以下では各サブモジュールについて説明する．

3.2.1 コントロールモジュール

ここでは GAA チップ全体の動作を制御している．

3.2.2 乱数発生モジュール

GAA では交差確率と突然変異確率はユーザが任意に設定できる．また，交差においては交差の対象となる個体をランダムに選択する．これらの機能を実現するためには乱数発生回路が必要である．少ないハードウェア量で質の良い乱数を発生させるために，GAA ではセルラオートマトンに基づくアルゴリズム[11]を用いて 24 ビットの疑似乱数を発生させている．乱数の初期値はユーザが設定可能である．

3.2.3 交差・突然変異モジュール

GAA ではランダムに選ばれた 2 つの個体に対し，交差手法を選択し交差を実行している．まず，乱数によって選択された個体がシステムメモリから読み込まれ，個体のエリート度に応じて一様交差と 2 点交差の 2 つの交差手法のうちどちらかが選ばれて交差が行われる．ある個体がエリートかどうかの判定は前述したように，文献 6) の手法では標準偏差を用いている．しかし，標準偏差を計算するとハードウェアコストが大きい．このため，GAA ではエリートの判定を以下のように簡略化し，ハードウェアコストの削減を図っている．すなわち GAA では対象とする問題を最大化問題とし，エリートの定義を以下のようにする．

$$Elite_i^T(j) = \{x_k^{T-j} \mid x_k^{T-j} \in Anc_i^T(j), \; f_{ave}^{T-j} + \alpha \times (f_{best}^{T-j} - f_{ave}^{T-j}) \leq f(x_k^{T-j})\} \quad (3)$$

ただし，$f(x)$ は個体 x の適応度，f_{ave}^T，f_{best}^T は世代 T における個体の適応度の平均と最良値を表す．また α の値を 0.25 と 0.5 の 2 種類のみとする．このようにすることにより，エリート判定は加算とシフト演算および比較演算のみで実現でき，実現に必要なハードウェア量が削減される．

また，このモジュールでは 2 点交差と一様交差を回路規模の小さい同一回路で効率良く実行できるように工夫している．すなわち，回路は**図 3** のように 2 つの 64 ビットシフトレジスタ，マルチプレクサ（MUX），2 点交差の 2 つの交差ポイントを格納するレジスタ，比較器，カウンタで構成されている．交差手法として 2 点交差が選ばれた場合，乱数発生回路で 2 つの交差ポイントをランダムに生成し，2 つの交差ポイントレジスタに設定する．カウンタの値が小さい方の交差ポイントの値より小さい間はシフトレジスタは自分の最上位ビットを自分の最下位ビットへ，大きくなるともう 1 つのシフトレジスタの最下位ビットへとシフトを行っていく．そして大きい方の交差ポイントよりカウンタの値が大きくなると，また自分の最下位ビットへと最上位ビットをシフトする．64 ビットをすべてシフトすると動作を終了し，次の個体対の処理に移る．

交差手法として一様交差が選ばれた場合は同じ回路を用いて，乱数を 1 ビットずつ直接マルチプレクサに送り，乱数が “1” の場合はシフトレジスタの最上位ビットを他方のシフトレジスタの最下位ビットに，“0” の場合は自分自身の最下位ビットにシフトするという動作により一様交差を実現している．

交差と同時にビットをシフトするごとに突然変異もユーザが指定した確率で実行され，突然変異を起こす場合はシフトされるビットの 0，1 を反転する．64 ビットのシフトが終了すると，交差後の個体データはシステムメモリに格納されるとともに GAA チップ外部の適応度評価回路に外部レジスタを介して受け渡され，個体の適応度評価が行われる．適応度評価回路が計算した個体の適応度は外部レジスタを介して GAA に取り込まれ，システムメモリに格納される．また，交差した個体の家系図データも更新される．

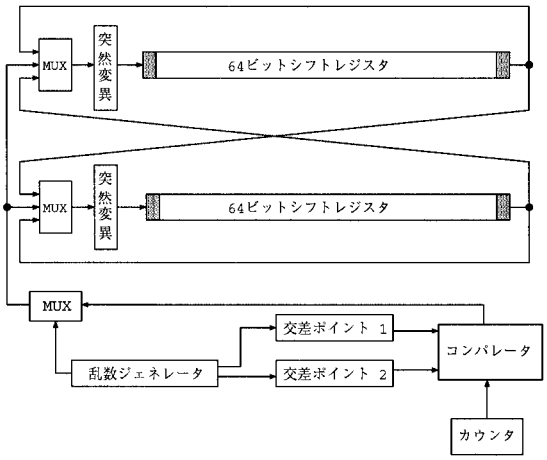

図 3 シフトレジスタを用いた交差
Fig. 3 Crossover using shift registers.

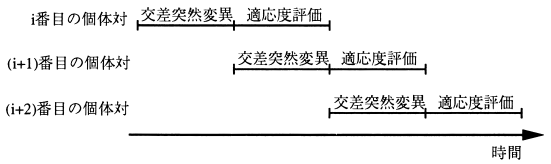

図 4 交差突然変異と適応度評価のパイプライン処理
Fig. 4 Pipeline processing of crossover, mutation and fitness evaluation.

交差突然変異モジュールは外部の適応度評価回路との間でパイプライン処理を実現している．すなわち，i 番目の個体対に対して交差が終了して交差結果を適応度評価回路に渡すと，評価の終了を待たずに交差突然変異モジュールは次の $(i+1)$ 番目の個体対に対する交差を開始する（**図 4**）．このようにすることで，交差突然変異と適応度評価が並列に実行され，計算時間が短縮される．

3.2.4 平均適応度・最大適応度計算モジュール

このモジュールでは個体のエリート判定，ルーレット選択，エリート戦略に用いるための平均適応度と最大適応度の計算を行う．

3.2.5 次世代個体選択モジュール

このモジュールでは各個体の適応度と各世代における適応度の平均に基づいてルーレット選択，エリート戦略を実行する．

ルーレット選択は GA の次世代個体選択手法の最も一般的なものであり，通常は次の手順で行う．

ステップ 1 現世代の個体の適応度の総和を求める．

ステップ 2 0 から適応度の総和までの範囲で乱数 n を生成する．

ステップ 3 個体の適応度を順番に加えていき，適応度の和が n を超えた時点の個体を次世代に残す．

ルーレット選択では高い適応度を持つ個体はそれだけ次世代に残りやすくなるが，適応度の低い個体でも次世代に生き残る可能性も残されている．

ルーレット選択をそのままハードウェア化すると以下の問題点がある．まず，上記の手順は本質的に線形探索を必要とし，m 個の個体を選択するのに $O(m^2)$ のクロック数を必要とする．次に，任意の数の範囲で一様乱数を生成するためには乱数発生回路のハードウェアコストが大きくなる．また，ハードウェア化とは無関係だが，ルーレット選択自体の問題点として，個体数が少ない場合，乱数のゆらぎにより，適応度を正確に反映しない選択がなされる可能性がある[8]．

GAA では上記の問題点に対処するため，ルーレット選択の変形である期待値選択[8]を採用し，次の手順で実現している．

ステップ 1 現世代の適応度の平均 $\bar{f}$，およびその 3/4，1/2，1/4，1/8 の値 $\bar{f}_{3/4}$，$\bar{f}_{1/2}$，$\bar{f}_{1/4}$，$\bar{f}_{1/8}$ を計算する．個体 x の適応度を f_x とする．

ステップ 2 $f_x \geq \bar{f}$ ならば個体 x を次世代に残し，$f_x \leftarrow f_x - 1$ としてステップ 2 を繰り返す．そうでなければステップ 3 にいく．

ステップ 3 f_x を $\bar{f}_{3/4}$，$\bar{f}_{1/2}$，$\bar{f}_{1/4}$，$\bar{f}_{1/8}$ と同時に比較し，$f_x \geq \bar{f}_p$ が成り立つ最大の p に対し，確率 p で個体 x を次世代に残す．$f_x < \bar{f}_{1/8}$ の場合はその個体は次世代には残らない．

個体の適応度を $\bar{f}$，$\bar{f}_{3/4}$，$\bar{f}_{1/2}$，$\bar{f}_{1/4}$，$\bar{f}_{1/8}$ の 5 つの値としか比較しないのは，ハードウェアコストを抑えるとともに，1 クロックで個体の選択を可能にするためである．上記のように選択を行えば，個体数に比例するクロック数で次世代に残す個体を選択できる．

エリート戦略は現世代で最も適応度の高い個体を次世代の個体として無条件に残す手法であり，GAA では上記のルーレット選択とエリート戦略を併用している．

3.2.6 システムメモリインタフェース

GAA チップとシステムメモリ間のインタフェースであり，アドレス 15 ビット，1 ワード 16 ビットでシステムメモリとデータの転送を行う．1 回のメモリアクセスは 2 クロックで実行される．

3.2.7 適応度評価回路インタフェース

GAA チップと外部適応度評価回路間のインタフェースであり，ハンドシェイク制御でデータ転送を制御している．また，個体データの授受のためのレジスタを GAA と適応度評価回路の間に置くことを想定している．

3.2.8 PC インタフェース

GAA チップとパーソナルコンピュータ間のインタフェースである．GAA は 4 個の 16 ビットコマンドレジスタを持ち，コマンドレジスタにより，遺伝オペレータと遺伝パラメータ値の指定，および GAA の実行制御を行う．

3.3 設計手順と設計結果

GAA を NTT エレクトロニクス（NEL）株式会社製の 4.8 mm 角，ピン数 108 ピン（信号ピン 76，電源ピン 32），ポリシリコン 1 層，メタル配線 2 層，CMOS 0.5 μm テクノロジチップに実装した．スタンダードセルライブラリには京都大学で開発された p2lib[7]の NEL 用ライブラリ Ver.1.6 を用い，チップ試作は VDEC（東京大学大規模集積システム設計教育研究センター）に依頼した．

回路設計はハードウェア記述言語 Verilog-HDL を用いてレジスタトランスファレベル（RTL）の回路記述を行い，論理合成システムによりゲート回路に変換した．RTL 回路記述においては図 2 で示した GAA の実行フロー中，フェーズ 2，3，5 をそれぞれ単独のモジュールとして別個に記述し，残りをトップモジュールとして記述した．論理合成においては全体を階層を取り除いたフラットな形で合成した．Verilog-HDL による回路記述はコメントを含めて約 2000 行である．

シミュレーションは Cadence 社の Verilog-XL，論理合成は Synopsys 社の Design Compiler，配置配線は Cadence 社の Cell Ensemble を用いて行った．設計には仕様の策定から始めてマスクパターンデータを出力するまで約 2 カ月を要した．最終的な論理合成結果は，セル数が 5890，ネット数 5915，ゲート数は 2 入力 NAND ゲート換算で 9369 であった．また，ポストレイアウトシミュレーションの結果，システムメモリに使用する SRAM のアクセス時間を 0 ナノ秒と仮定すれば，GAA はクロック周波数 50 MHz での動作が可能であることを確認した．しかしながら，実際は

SRAM のアクセス時間の制約により，GAA の実際の動作クロック周波数は 20 MHz 以下となる．試作チップを実装した評価ボードではクロック周波数 15 MHz での動作を確認している．

4. 提案ハードウェアの評価

4.1 シミュレーション実験による評価

本研究で試作した GAA チップに対し，GA ハードウェアとしての性能評価を行うために，GAA のアルゴリズムの基となっている文献 6) の手法，および Grefenstette の作成した汎用 GA プログラム GENESIS 5.0[4]を比較手法とし，De Jong が GA の評価のためにベンチマーク問題として提案した 5 つの関数最小化問題[8]のうちの 4 つ（詳細は付録を参照）を対象として比較実験を行った．GENESIS については交差手法として一様交差を実行するものと 2 点交差を実行するものの 2 つを比較手法とした．以下では文献 6) の適応的 GA を Adaptive，一様交差を適用する GENESIS を Uniform，2 点交差を適用する GENESIS を 2-point と表記し，これらを総称してソフトウェア GA と呼ぶことにする．

GAA の実行条件は以下のとおりである．

- 世代数：4096（または最適解を得た時点で終了）
- 個体数：64
- 交差確率：143/256 (= 0.5586)
- 突然変異確率：4/4096 (= 9.766×10^{-4})
- エリート影響度係数（β）：0.5
- 交差手法：適応的交差（2 点交差 + 一様交差）
- 交差手法の適応的選択で用いるしきい値：3.0
- エリート決定係数（α）：0.5

一方，ソフトウェア GA については，対応するパラメータがある場合は GAA と同じ実行条件に設定した．

実験は ULTRA COMP station model 170（167 MHz）上で行った．GAA については，Verilog-HDL で記述されたソースファイルを論理合成システムによりゲートレベルに変換し，Verilog-XL シミュレータでシミュレートした．外部適応度評価回路については，計算部分は C 言語で実現し，Verilog-XL が提供している PLI（Program Language Interface）を用いて，GAA のゲートレベル記述とインタフェースをとってシミュレーションを行った．2 個体に対する適応度評価は GAA の 2 個体に対する交差・突然変異に必要な時間内に終了するものと仮定している．一方，ソフトウェア GA については，C 言語で実現し，上記ワークステーション上で実行した．

個体の適応度評価に必要な時間が交差・突然変異に必要な時間内に終了するという仮定の妥当性については実験の対象とした関数の評価回路を FPGA 上で論理合成ツールにより実現することにより確認している．たとえば，4 つの関数の中で最も複雑な f_5 に対し，Altera 社製の FPGA である FLEX10K100 をターゲットデバイスとして論理合成ツールを用いて回路設計を行ったところ，使用ロジックエレメント（LE）数が 4693，動作周波数 16 MHz の回路を実現できた．また，1 個の個体対（すなわち，2 個の個体）の適応度評価に必要なクロック数は 109 クロックだった．一方，GAA は 2 個体の交差・突然変異とその前後処理の実行に 150 クロックが必要なので，パイプライン処理により 2 個体の適応度評価と交差突然変異の並列実行が可能となる．なお，毎世代，平均 m 個の個体対の交差突然変異と個体評価を行うと仮定した場合，実際の回路では最後の 1 個の個体対の適応度評価については交差突然変異との重複は起こらない（図 4 参照）．一般に，m = 個体数 × 突然変異確率 ÷ 2 の関係があるが，ここでの実験の数値を代入すると，$m \simeq 17.9$ であり，最後の 1 個の個体対の適応度評価の処理時間は交差突然変異の処理時間と重複しないことを考慮した場合と m 個の個体対のすべてについて処理時間が重複すると仮定した場合を比較すると後者はたかだか 5.6%の誤差しか生じないことが分かる．GA 実行全体の実行時間を考える場合はこの誤差はさらに小さくなる．そこでシミュレーション実験を簡単にするためにここでは上記のように仮定した．

実験は各関数に対してそれぞれ 10 個の初期個体群をランダムに生成し，それらに対して各手法を適用して最適解を求めたときの世代数と実行時間の平均値と最良値を比較した．実験結果を**表 2** に示す．実行時間（単位はミリ秒）は GAA においてはクロック周波数 15 MHz を仮定してシミュレーションより求めたクロック数より算出した．ソフトウェア GA においては CPU 時間を計測した．適応度評価時間は，GAA については前述したとおり，パイプライン処理により GAA の実行時間と完全に重複できると仮定して，実行時間には含めていない．一方，ソフトウェア GA の実行時間は個体の適応度の評価時間も含んでいる．また，10 回の試行において，最適解を得ることができなかった回数を失敗回数として示す．世代数と実行時間の平均値と最良値は，最適解を求めることができた試行のみを対象として求めている．10 回の試行のいずれにおいても最適解と求めることができなかった場合は「-」で示す．GAA においては，個体の染色体長が 64 ビット，適応度が 16 ビットなので変数および適応度の

表 2 実験結果
Table 2 Experimental results.

			GAA	Adaptive	Uniform	2-point
f_1	世代数	平均	87	54	310	58
		最良	55	32	178	54
	実行時間	平均	31.92	43980	5180	550
	（ミリ秒）	最良	18.70	26060	2970	500
	失敗回数		0	0	0	0
	評価時間（%）		0.0	0.4	0.1	1.6
f_2	世代数	平均	17	75	-	83
		最良	1	33	-	39
	実行時間	平均	5.63	25060	-	730
	（ミリ秒）	最良	0.33	11050	-	340
	失敗回数		0	4	10	2
	評価時間（%）		0.0	0.2	0.1	3.4
f_3	世代数	平均	62	44	420	50
		最良	43	23	276	34
	実行時間	平均	20.83	17650	6260	680
	（ミリ秒）	最良	14.21	9250	4110	460
	失敗回数		0	0	2	0
	評価時間（%）		0.0	1.8	1.0	5.4
f_5	世代数	平均	371	259	-	317
		最良	53	24	-	42
	実行時間	平均	125.32	101600	-	5130
	（ミリ秒）	最良	17.70	9410	-	670
	失敗回数		1	6	10	7
	評価時間（%）		0.0	17.6	12.0	56.7

数値表現に丸め誤差が生じる．このため，ソフトウェア GA では最適解と求めた解の差が GAA の丸め誤差以内になった時点で最適解を求めたと判定している．さらに，ソフトウェア GA に対しては，全実行時間の中で関数評価が占める割合を，C コンパイラの profile 機能を利用して，上記の実験とは別に計測した．その結果も表 2 に示す（全体の実行時間を 100%としたときの関数評価にかかった時間を%で示す）．

いずれの実験結果を見ても，GAA は非適応的 GAA のソフトウェア実行（表中の Uniform，2-point）と比較して 15 倍以上，適応的 GA のソフトウェア実行（表中の Adaptive）と比較して 500 倍以上高速に，同等の解を求めている．Adaptive との比較で実行時間の差が特に大きい要因としては，現在のプログラムでは各個体の家系図データの処理のオーバヘッドが大きいことがある．このため，この部分を改良すれば差は小さくなると考えられるが，それでも GAA と比較して数 10 倍の計算時間の差はあると予想される．また，GAA ではエリート判定とルーレット選択については，元のアルゴリズムを簡略したものをハードウェア化しているが，最適解が求まる世代数が少し大きくなっていること以外は，アルゴリズムの簡略化は GAA の適応的 GA としての性能に大きな影響を与えていないものと考えられる．

さらに，ソフトウェア GA における全体の実行時間に対する関数評価時間の割合から，適応度の評価計算が簡単な場合は外部評価回路をソフトウエアで実現する場合でも GAA の導入により十分なパフォーマンスの向上が望めるが，適応度の評価計算が複雑な場合は，外部評価回路を FPGA で実現するなどしてハードウェア化しないと，GAA のみの導入ではパフォーマンスの向上に限界があることも分かる．

4.2 実機による評価

GAA チップと周辺回路を A4 サイズのボード上に組み立てた GAA 評価ボードを作成し，実機による GAA の評価も行った．この評価ボードでは外部適応度評価回路としてホストコンピュータおよび FPGA ボードを利用できる．しかしながら，実験の結果，ホストコンピュータを利用した場合は，GAA チップとホストコンピュータとの間の個体情報および適応度データの授受のオーバヘッドが大きい，という問題点があることが判明した．オーバヘッドが大きい原因は，現在のプログラムではデータ授受のためのハンドシェーク制御などもすべて C 言語で記述しており，フラグを 1 ビット変化させるのにも関数呼び出しで実行しているなど，この部分の実行に多くの CPU 時間が必要になるためである．クロック周波数が 15 MHz の GAA の場合，交差フェーズでは約 9 マイクロ秒おき

にハンドシェーク制御によりGAAと適応度評価回路の間でデータの授受が実行されるが，適応度評価回路がFPGAの場合はデータの授受は約2マイクロ秒で終了する．これに対してホストコンピュータ（CPU: Pentium 133 MHz）を適応度評価回路として使用した場合はデータの授受に数十マイクロ秒を要している．したがって，GAAとのインタフェースをアセンブラ言語で記述したり，DMA（Direct Memory Access）を利用したりすれば，オーバヘッドは小さくなると予想される．

一方，外部適応度評価回路にFPGAを用いた場合はGAAとFPGAとの間での通信のオーバヘッドは無視でき，ほぼ，シミュレーションと同じ計算時間で実行できることが確かめられた．

4.3 考　　察

本研究で開発したGAAチップは，試作チップということで多くの制約がある．このため，チップの実用化のためには改良すべき点が多い．以下に改良すべき点を列挙する．

(1) 汎用性の向上

今回のGAAチップ試作においては，デザインルールが0.5 μmのメタル2層CMOSスタンダードセルテクノロジで，チップの大きさが4.8 mm^{2}に限られていたために，ハードウェア量の制約からチップの仕様が限定されている．このため，より汎用性の高いチップとするためには仕様を拡張する必要がある．特に，今回の個体ビット数（染色体長）は64ビットと短いので，これを1000ビット以上に拡張するとともに，染色体長を外部よりパラメータで指定できるようにする必要がある．また，交差手法についても一様交差，2点交差以外の手法についても選択可能にすることが望まれる．その他，各種パラメータ値の指定可能範囲を拡大することが望まれる．

(2) 処理速度の向上

より処理速度を向上させるためには，クロック周波数の向上が必要である．また，並列処理やパイプライン処理をさらに導入することが望まれる．特に交差は現在はシフトレジスタを用いてビットごとに逐次処理する回路となっているが，処理速度の向上のためには複数ビットを並列に処理することが必要である．また，GAAではメモリアクセスが全体のパフォーマンス向上のボトルネックの一因になっているので，メモリアクセスの高速化と高メモリデータバンド幅の実現も処理速度の向上には有効である．

5. おわりに

本論文ではGAにおいて，各個体における潜在的な優劣の度合いを示す指標であるエリート度を用いて，アルゴリズムの実行中に交差手法を適応的に選択する手法を導入し，これを専用ハードウェア化し，より効率的かつ高速な解の探索を実現することを提案した．実際に提案ハードウェアをLSIチップとして試作した結果，適応的GAのハードウェア化はパフォーマンスの向上に大変有効であることが分かった．

今後の課題としては，GAAのパフォーマンスのよりいっそうの向上を目的として，4.3節で述べたハードウェアの改良があげられる．また，GAの大幅なパフォーマンスの向上を実現するためには並列GAが有効であることが知られており，適応的GAに基づく新しい並列GAアーキテクチャの開発とLSIによる実現も今後の課題である．また，GAAは世代GAを基としているが，世代GAのハードウェア化においては本質的にパイプラインストールが発生することが知られており，これに対処するために世代を持たない定常GA（steady state GA）に基づくGAのハードウェア化も試みられている[5]．このため，遺伝オペレータの適応的選択機能を組み込んだ定常GAのハードウェア化も今後の興味深い課題である．

謝辞 GAAの評価実験にご協力いただいた本学学部生の上野初君，平木辰志君，山根正孝君に感謝する．本研究の一部は平成10年度科学研究費補助金基盤研究（C)(2)(課題番号10680356）による．本研究におけるLSIチップの試作は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NTTエレクトロニクス株式会社および大日本印刷株式会社の協力で行われたものである．

参 考 文 献

1) Davis, L.: Adapting operator probabilities in genetic algorithms, *Proc. 3rd International Conference on Genetic Algorithms*, pp.61–69 (1989).
2) Davis, L. (Ed.): *Handbook of Genetic Algorithms*, Van Nostrand Reinhold (1991).
3) Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company (1989).
4) Grefenstette, J.J.: Optimization of control parameters for genetic algorithms, *IEEE Trans. Systems, Man, and Cybernetics*, Vol.SMC-16, No.1, pp.122–128 (1986).
5) 北浦　理, 杉浦弘幸, 川合隆光, 安藤秀樹, 島田俊

夫：パイプラインストールを除去した遺伝的アルゴリズム専用ハードウェア，電子情報通信学会コンピュータシステム研究会技術研究報告，CPSY98-81 (1998).
6) 八田浩一，松田憲治，若林真一，小出哲士：遺伝的アルゴリズムにおける交差手法の適応的選択の一手法，電子情報通信学会論文誌（D-I），Vol.J81-D-I, No.7, pp.900–909 (1998).
7) 小野寺秀俊，平田昭夫，北村晃男，田丸啓吉：P2Lib：スタンダードセルライブラリ自動生成システム，情報処理学会設計自動化研究会研究報告，84-4 (1997).
8) 坂和正敏，田中雅博：遺伝的アルゴリズム，朝倉書店 (1995).
9) 佐野雅彦，井上富夫，高橋義造：FPGA による SIMD 型 GA マシンの設計，情報処理学会計算機アーキテクチャ研究会研究報告，125-6 (1997).
10) Scott, S.D., Samal, A. and Seth, S.: HGA: A hardware-based genetic algorithm, *Proc. ACM/SIGDA 3rd Int'l Symposium on FPGA*, pp.53–59 (1995).
11) Serra, M., Slater, T., Muzio, J.C. and Miller, D.M.: The analysis of one-dimensional linear cellular automata and their aliasing properties, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.9, No.7, pp.767–778 (1990).
12) Spears, W.M.: Adapting crossover in evolutionary algorithms, *Proc. 4th Evolutionary Programming Conference*, pp.367–384 (1995).
13) Wakabayashi, S., Koide, T., Toshine, N., Goto, M., Nakayama, Y. and Hatta, K.: An LSI implementation of an adaptive genetic algorithm with on-the-fly crossover operator selection, *Proc. 1999 Asia-South Pacific Design Automation Conference*, pp.37–40 (1999).
14) 吉田紀彦，森木俊臣，安岡智宏：SFL による遺伝的アルゴリズム VLSI の設計，第 10 回パルテノン研究会資料集，pp.63–70 (1997).

付　　録

A.1 テスト関数

De Jong の関数最小化問題のうち実験に用いた 4 つの各関数の定義，最適解，GAA でのコード化を以下に示す．評価値はいずれも固定小数点表現としている．ここで，$\min(f_i)$ は関数 f_i の最小値を示す．

(1) **球面モデル**

$$f_1(\vec{x}) = \sum_{i=1}^{3} x_i^2$$
$$-5.12 \le x_i \le 5.12,\ i = 1,2,3$$
$$\min(f_1) = f_1(0,0,0) = 0$$

1 変数のコード：20 ビット
評価値のコード：整数部 7 ビット，小数部 9 ビット
最適解：0.0（丸め誤差：0.00195）

(2) **一般化 Rosenbrock 関数**

$$f_2(\vec{x}) = \sum_{i=1}^{2}(100 \cdot (x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$$
$$-5.12 \le x_i \le 5.12,\ i = 1,2,3$$
$$\min(f_2) = f_2(1,1,1) = 0$$

1 変数のコード：32 ビット
評価値のコード：整数部 12 ビット，小数部 4 ビット
最適解：0.0（丸め誤差：0.0625）

(3) **ステップ関数**

$$f_3(\vec{x}) = 30 + \sum_{i=1}^{5} \lfloor x_i \rfloor$$
$$-5.12 \le x_i \le 5.12,\ i = 1,\ldots,5$$
$$\min(f_3) = f_3([-5.12,-5),\ldots,[-5.12,-5)) = 0$$

1 変数のコード：12 ビット
評価値のコード：整数部 6 ビット，小数部 10 ビット
最適解：0.0（丸め誤差：0.000977）

(4) **Shekel の塹壕**

$$\frac{1}{f_5(\vec{x})} = \frac{1}{K} + \sum_{j=1}^{25} \frac{1}{c_j + \sum_{i=1}^{2}(x_i - a_{ij})^6}$$
$$(a_{ij}) = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \cdots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \cdots & 32 & 32 & 32 \end{pmatrix}$$
$$K = 500\ ;\ f_5(a_{1j}, a_{2j}) \approx c_j = j$$
$$-65.536 \le x_i \le 65.536, i = 1,2$$
$$\min(f_5) = f_5(-32,-32) \approx 1$$

1 変数のコード：32 ビット
評価値のコード：整数部 9 ビット，小数部 7 ビット
最適解：0.9746（丸め誤差：0.00781）

(平成 11 年 3 月 26 日受付)
(平成 12 年 3 月 2 日採録)

若林　真一（正会員）

昭和 31 年生．昭和 54 年広島大学工学部電気工学科卒業．昭和 56 年同大学大学院工学研究科システム工学専攻博士課程前期修了．昭和 59 年同博士課程後期修了．同年日本アイ・ビー・エム（株）入社．東京基礎研究所副主任研究員．昭和 63 年 7 月より広島大学工学部助教授．VLSI CAD，組合せ最適化，遺伝的アルゴリズムに関する研究に従事．工学博士．IEEE，ACM，電子情報通信学会各会員．

小出　哲士（正会員）

昭和 42 年生．平成 2 年広島大学工学部第二類（電気系）卒業．平成 4 年同大学大学院工学研究科博士課程前期修了．同年広島大学工学部第二類（電気系）助手．平成 11 年 3 月広島大学工学部助教授．平成 11 年 4 月より東京大学大規模集積システム設計教育研究センター（VDEC）助教授．VLSI CAD，VLSI 設計・教育，組合せ最適化，遺伝的アルゴリズムに関する研究に従事．博士（工学）．IEEE，ACM，電子情報通信学会各会員．

八田　浩一（正会員）

昭和 43 年生．平成 2 年広島大学工学部第二類（電気系）卒業．平成 4 年同大学大学院工学研究科博士課程前期修了．同年中国電力（株）入社．平成 8 年 4 月より平成 11 年 3 月まで広島大学大学院工学研究科博士課後期に在学．博士（工学）．広島大学大学院工学研究科博士課程後期在学中は遺伝的アルゴリズムに関する研究に従事．電子情報通信学会，IEEE 各会員．

中山　喜勝

昭和 47 年生．平成 8 年静岡大学工学部電気工学科卒業．平成 10 年広島大学大学院工学研究科博士課程前期修了．同年キヤノン（株）に入社．広島大学大学院在学中は遺伝的アルゴリズムのハードウェア化に関する研究に従事．

後藤　睦明

昭和 50 年生．平成 8 年沼津工業高等専門学校卒業．同年広島大学工学部第二類（電気系）に編入学．平成 10 年同学部卒業．同年（株）デンソーに入社．広島大学在学中は遺伝的アルゴリズムを実現する LSI 設計に従事．

利根　直佳

昭和 51 年生．平成 10 年広島大学工学部第二類（電気系）卒業．平成 12 年同大学大学院工学研究科博士課程前期修了．同年富士通（株）入社．広島大学在学中は遺伝的アルゴリズムのハードウェア化，および並列アルゴリズム化に関する研究に従事．