FPGA 上への遺伝的アルゴリズムの柔軟な実装手法の提案

橘　達弘[†a)]　村田　佳洋[†]　柴田　直樹[††]　安本　慶一[†]
伊藤　実[†]

Proposal of Flexible Implementation of Genetic Algorithms on FPGAs

Tatsuhiro TACHIBANA[†a)], Yoshihiro MURATA[†], Naoki SHIBATA[††],
Keiichi YASUMOTO[†], and Minoru ITO[†]

あらまし　遺伝的アルゴリズム（GA）は様々なアプリケーションに用いることができる．GA を用いたアプリケーションは，ハードウェア上に実装することで，安価で資源の少ない情報機器上で利用することができる．本論文では，適用する問題や利用可能な回路規模に従って，実行効率の良い GA 回路を合成するためのアーキテクチャを提案し，またこのアーキテクチャに従って実装される回路の規模予測モデルを提案する．本アーキテクチャ及び回路規模予測モデルを用いて，指定する FPGA デバイスに実装が可能なパラメータ値を探索するツールと，そのパラメータ値をもとに VHDL で記述された RT レベル回路記述を自動で生成するツールを作成した．提案手法の有効性を示すために，ナップサック問題と巡回セールスマン問題を対象とする GA 回路を提案アーキテクチャに従って VHDL で記述し，コンパイルを行いゲートレベルで合成した．合成された回路が優れた解探索性能をもつことをシミュレーションにより確認し，また低消費電力で動作することも確認した．また，回路規模の予測結果が実際に論理合成を行って得たサイズに十分近いことを確認した．

キーワード　遺伝的アルゴリズム，FPGA，ハードウェア自動合成，ナップサック問題，巡回セールスマン問題

1. まえがき

遺伝的アルゴリズム（Genetic Algorithm，以下 GA）は組合せ最適化問題の近似解を得る手法であり，様々なアプリケーションに利用できる．GA は，複数の解候補を個体として扱い，これら複数の個体の評価を繰り返し行うため，比較的計算量が大きくなる．そのため，携帯端末などのリソースの限られた機器での使用は制限があった．GA の携帯端末上でのアプリケーションとして，巡回セールスマン問題（以下 TSP）に時間制約などを加え，観光向けのパーソナルナビゲーションに応用したもの [1] や，GA を用いた，高速かつ高品質な画像の圧縮 [2] 等がある．また，ネットワークにおいてビデオなどのマルチメディアデータを実時間配信するための最適コストのマルチキャスト配送木を高速に算出する手法 [3] も提案されており，ルータ上でこのようなアルゴリズムを実行することができれば便利である．これらのアプリケーションを，携帯端末，ルータ，あるいは FAX/AV 機器などに実装するためには，GA を高速，低消費電力，かつ安価に実現するための実装技術が必要である．また，GA を実行する大規模な並列計算機システムを実現する試み [4] が行われているが，このようなシステムの拡張性を高めるためには，各処理ユニットの高速化，小型化，低消費電力化が必須である．

本論文では，与えられた問題とその問題サイズ，実装先 FPGA の書込み可能な回路規模から適切な回路を構築するための手法を提案する．そのために，まず様々な GA のアプリケーションをハードウェアで実装するためのアーキテクチャを提案する．本論文で提案するアーキテクチャは，GA オペレータだけではなく，評価系も含めてハードウェアで実装することで，計算量の大きい評価系の処理時間も短縮し，効率の良い実装を可能にしている．このアーキテクチャは，基本となるモジュールの組合せで構成され，モジュールを組

† 奈良先端科学技術大学院大学情報科学研究科，生駒市
Graduate School of Information Science, Nara Institute of Science and Technology, 8916–5 Takayama, Ikoma-shi, 630–0192 Japan
†† 滋賀大学経済学部情報管理学科，彦根市
Department of Information Processing and Management, Shiga University, 1–1–1 Bamba, Hikone-shi, 522–8522 Japan
a) E-mail: tatsu-ta@is.naist.jp

み換えることで，様々な問題に適用が可能である．また，並列化の拡張性に優れた並列型 GA の一種である島モデル型 GA（Island GA，IGA）を構成することが可能であり，使用可能な回路規模が大きい場合に，これを十分に生かして高い探索性能を得ることが可能である．また，並列度が高くなってもクリティカルパスが長くなりにくい特徴がある．次に，指定した FPGA 上への柔軟な実装を可能にするために，GA で用いる個体数，並列度等から，FPGA 上に実現される回路規模を予測する手法を提案する．これにより，回路を合成することなく，性能及び最終的な回路規模を予測することが可能になる．性能とコストのトレードオフを考慮し，指定した FPGA 上にその回路規模の範囲内で，並列 GA を実装することができ，ラピッドプロトタイピングに役立つ．提案手法による GA 回路の実装を容易にするため，回路規模の予測結果から，実装可能なデバイス及び並列度を探索する実装判定ツール，と実装判定ツールを用いて得られたパラメータ値から VHDL で記述された GA 回路を導出するツールを実装した．

本論文では，提案したアーキテクチャを用いて，問題サイズの異なるナップサック問題（Knapsack Problem）と TSP に対するハードウェア GA を実装することで本アーキテクチャの問題サイズに対する汎用性を示す．また，それぞれの回路規模予測モデルを導出し，実際に合成された回路と予測を比較することで，提案手法の有効性を示す．また，提案アーキテクチャを用いて作成した回路が，ナップサック問題，TSP ともに，優れた性能を示し，低消費電力で実行可能であることをゲートレベルのシミュレーションにより確認した．

2. 関 連 研 究

GA のハードウェアの実装に関して，いくつかの研究がなされている．[5] では H^3 エンジンと呼ばれる遺伝的アルゴリズム専用ハードウェアを実装している．[6] では，交叉法に適応選択を組み込んだ GA をハードウェア化している．[7] では，不連続閉曲線抽出手法を対象としたハードウェア GA を適用している．[8] では TSP を対象にハードウェア GA を実装している．[9] では，比較的容易な問題である Set Coverage Problem に適用している．その他，[10] を含む既存の手法の多くは，それぞれの問題に特化したアーキテクチャを使用している．そのため，対象外の問題を適用する際の符号化した染色体をどのように扱い，どのように送信するかの仕様が定義されておらず，また，交叉，突然変異などの遺伝的操作をハードウェア化した際の入力と出力の仕様も定義されていない．結果として特定のアーキテクチャを他の問題にそのまま適用することは困難である．

様々な用途に応じた GA 回路を情報機器や小型の電子装置上に実装するためには，汎用的な GA 回路のアーキテクチャが必要である．また，実装する FPGA の種類や GA の対象となる問題ごとに発生する様々な制約条件（性能，コスト，消費電力）を満足する回路を作成するための手法が必要となる．このような，設計の補助を行う手法として，[11] では，リアルタイムの組込み型システムの効率的な開発のための手法が提案されている．しかし，この手法は，GA 回路の開発に特化するものではない．

3. GA のハードウェア化の基本アイデア

この章では，遺伝的アルゴリズム（GA）について簡潔に説明を行い，その後，本論文で用いた GA の回路化手法の概要について述べる．また，提案手法で採用する，ハードウェア化に適していると思われる既存のいくつかの GA アーキテクチャについて述べる．

3.1 遺伝的アルゴリズム

GA では，与えられた問題の探索空間内の一点（解候補）を表現する染色体（chromosome）と適応度（どれだけ最適解に近いかを表す数値，fitness）を含む個体を複数用いて近似解を求める．

GA の動作の概要は以下のようになる．(1) 多数のランダムな個体を作成する．(2) 各個体の適応度を計算する．(3) 良い適応度をもつ個体群を選択し，残りを破棄する．(4) 選択された個体のペアを両親（parent）とし，両親の性質を混ぜ合わせて新しい個体（offspring）を作成する（交叉）．(5) 新しい個体の染色体に対しある確率で突然変異を起こす．(6) ステップ (2) に戻る．これらの操作を特定の時間，または最適解に近い準最適解を得るまで繰り返し適用する．

染色体を構成している要素を遺伝子，その遺伝子の書かれている染色体内の位置を遺伝子座と呼ぶ．

3.2 GA の回路化手法の概要

提案手法の目的は，与えられた問題とその問題サイズ，実装先 FPGA の書込み可能な回路規模から適切な回路を合成することである．FPGA は，ロジックエレメント数と利用できるメモリブロックが限られてい

るため，これらを考慮して回路を合成する必要がある．目的を達成するために，汎用的なアーキテクチャを提案する．提案するアーキテクチャは以下の特徴をもつ．(1) ハードウェア化に適しており様々な問題に適用可能である．(2) 利用可能な回路規模内で，できるだけGA 回路の並列度を増やすことができる．(3) 与えられた問題とそのサイズから回路規模の予測を行う．

(1) を実現するために，Minimal Generation Gap（MGG）モデル [12] を基礎とした世代交代モデルを採用し，モジュール単位での設計を行う．この MGG モデルはハードウェア化に適しており，必要となるメモリとレジスタを削減することができる．世代交代モデルについては，**3.3** で説明する．また，モジュール単位での設計を行うことで様々な問題に適用が可能となる．

(2) を実現するために，複数の並行に実行されるパイプライン間での情報の同期頻度を少なくする必要がある．並列実行を行うアーキテクチャとして島モデル型 GA（Island GA，IGA）を採用する．並列実行については，**3. 4** で説明する．

(3) を実現するために，問題の種類，問題のサイズ，解候補の数，そして，並列度数より回路規模を予測する手法を開発した．詳細は，**5.** に示す．

3. 3　提案手法における世代交代モデル

一般的な GA の一つである simple GA（SGA）では，現世代の個体群とは別に，次世代の個体群を格納する領域を用意する必要がある．したがって，SGA をそのままハードウェア化すると，必要となるメモリ量が増加する．文献 [9], [10] では，survival-based, steady-state GA と Compact Genetic Algorithm を用いることで回路規模とメモリ量の増加を抑制している．しかし，survival-based，steady-state GA は，常に個体群の中で適応度が最も悪い個体を把握する必要がある．特にメモリを用いて個体群を保持する場合，最も適応度が悪い個体を探索するのに時間がかかり，パイプライン処理が困難となる．Compact Genetic Algorithm は，個体群を保持せず，代わりに染色体の集合を確率分布として保持する手法である．この手法は，各遺伝子を 0 若しくは 1 で表現する bit string 形式の符号化手法のみに使用でき，それ以外の符号化手法を用いる問題では適用が困難である．

本論文では，MGG モデル [12] を単純化した世代交代モデルを用いる．このモデルでは，まず個体を二つ，個体群より取り出し，これらの個体に交叉，突然変異演算子を適用し，新しい染色体を一つ生成する．次に，生成した子の染色体を評価する．新しく生成した染色体一つと親の染色体二つの中から，二つの最良個体を選択する．本モデルは，文献 [13] で採用されているモデルと同様であるが，二つの個体に交叉，突然変異を適用し，複数の染色体を生成し，親の染色体と生成された染色体間で選択を行う点で異なる．この選択方式は，生成された次世代の個体群の候補を確保する必要がないことに加え，個体群の更新を個体群全体で一括して行わない点でも，並列化とパイプライン化に向いている．

3. 4　島モデル型 GA の概要

GA の並列化には様々な手法が存在する．本論文では，並列 GA の一つである島モデル型 GA（Island GA，IGA）[4] を採用する．IGA は，解候補である個体群を複数の集合に分割する．これらの分割された個体群の集合を，IGA では島とみなし，これらの島で解候補を独自に進化させる．個体群全体での情報の共有を行うために，移民（migration）と呼ばれる操作を行う．移民は，一定間隔ごとに島間で個体を移動させる操作である．IGA は，SGA より多様性が確保されやすいため，局所解に陥りにくく，性能が改善される．回路の並列化において，クリティカルパスが長くなることで，最大動作周波数が減少することがある．IGA は，隣り合う島間で個体の移民を行えばよいため，クリティカルパスがそれほど長くならず，並列化による最大動作周波数の減少が抑えられる．

4.　共通アーキテクチャ

本章では様々な問題に適用可能であり，回路規模が予測可能で，並列化が容易である GA 回路向けアーキテクチャについて述べる．提案アーキテクチャによる GA 回路は，遺伝的操作に対応した複数のモジュールから構成される．提案アーキテクチャでは，モジュール間でのパイプライン化に適した情報の伝達手順や各モジュールの入出力仕様を定義する．これを行うことで，モジュール単位で様々な種類の交叉，突然変異及び，評価関数をハードウェアで実装することができ，それら組み換えることで様々な問題に適用可能となっている．また，同時に提案アーキテクチャは問題に依存せず，かつ，拡張性に優れた並列化手法も定義している．

ここでは，最小単位の GA 回路の構築手法である基本アーキテクチャと並列化を行う手法である並列アー

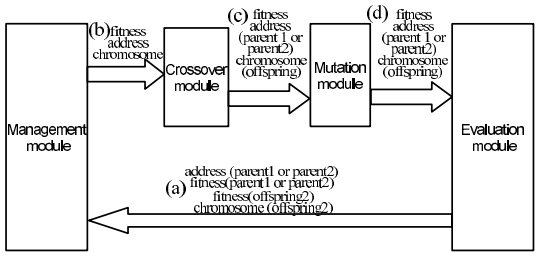

図 1　基本アーキテクチャ
Fig. 1　Basic architecture.

キテクチャについて述べ，最後に各モジュールの処理時間が一定でない場合の対処について述べる．

4.1　基本アーキテクチャ

基本アーキテクチャは，図 1 に示す四つのサブモジュール：管理モジュール（management module），交叉モジュール（crossover module），突然変異モジュール（mutation module），評価モジュール（evaluation module）から構成される．管理モジュールは，個体の染色体と適応度を保持する．交叉，突然変異，評価の各モジュールでは，それぞれ対応する遺伝的操作を行う．選択は，管理モジュールと交叉モジュールで行われる．

各個体の染色体は，n ビットの 2 進数で表される．モジュール間のバス幅は一定であり，m ビットとする．n, m の値はパラメータとして与えることができる．各モジュールは，原則として各クロックごとに m ビットのデータを受け取り，c クロック後に m ビットの演算結果を出力できるように設計される（ここで c は定数）．また，データ処理中も，各クロックごとに新たなデータを受け取れるよう設計されている．すなわち，$n \geq m$ の場合，$\lceil \frac{n}{m} \rceil$ クロックが 1 染色体あたりのスループットとなる．各モジュールは，順番にデータを受信しつつパイプライン的にデータを逐次処理する．

以下，各モジュールの詳細について述べる．

［管理モジュール］

管理モジュールは，メモリに個体群を格納し，管理を行う．評価モジュールより受信した親個体と子個体の適応度を比較する部分と，比較した結果をもとにメモリの読書きと交叉モジュールへの出力を行う二つの部分から構成される（図 1 (a)）．

評価モジュールより受信した子の適応度が，親の適応度より優れている場合，子の染色体及びその適応度をメモリ上の親個体のアドレスに上書きし，子の染色体とその適応度及びアドレスを交叉モジュールへ送信する．メモリへの読書きが同時に行えないため，メモリへの書込みが発生すると交叉モジュールへ情報の送信を行うことができなくなる．そのため，子個体の情報を送信することでストールの発生を防いでいる．

逆に親個体の適応度が優れている場合は，ランダムに選択されたアドレスとそのアドレスの個体の情報を交叉モジュールへ送信する（図 1 (b)）．

［交叉モジュール］

交叉モジュールは，受信した親個体と，一つ前に受信した親個体を交叉し，一つの子個体を生成する．一つ前に受信した親個体（アドレス，適応度，染色体）を格納するための領域をもつ．交叉モジュールでは，選択操作の一部も行う．二つの親の適応度を比べ，悪いほうの個体のアドレスと適応度を突然変異モジュールへ出力する（図 1 (c)）．

［突然変異モジュール］

突然変異モジュールは，交叉モジュールより受信した子の染色体に，与えられた確率で突然変異を適用し，評価モジュールへ送信する．また，交叉モジュールより受信した適応度が悪い方の親個体のアドレス及び適応度を評価モジュールへ出力する（図 1 (d)）．

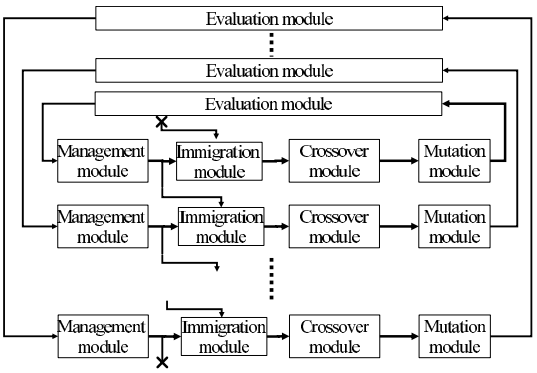

図 2　並列アーキテクチャ
Fig. 2　Parallel architecture.

4.2　並列アーキテクチャ

利用可能な回路規模に応じて性能を改善できるようにするために，基本アーキテクチャを並列化する．基本アーキテクチャにおけるモジュールの組合せを一つの島とみなし，複数の島からなる IGA を構成する．この際，複数の島の間で個体の交換が行えるようにするため，図 2 に示すように，各パイプラインの管理モジュールと交叉モジュールの間に移民モジュール (immigration module) を導入する．移民モジュール

は，図に示すように二つの島の管理モジュールと接続される．通常，交叉モジュールは自身が属する島の管理モジュールから個体を読み込むが，決められた周期で他方の島の管理モジュールから個体を読み込む．移民モジュールに接続する管理モジュールの数は，島の数によらず，常に 2 であるため，並列数の増加はクリティカルパスの長さに影響しない．

4.3 各モジュールの処理が一定クロックでない場合の対処

適用する問題（例えば TSP，Job Shop Scheduling Problem 等）によっては，各遺伝演算子モジュールで行われる処理が複雑になり，処理に必要なクロック数が一定とはならない．このようなモジュールが存在すると，パイプラインのストールによって他のモジュールの性能を生かしきることができない．処理クロック数が一定ではないモジュールが存在する場合，以下の二つの手法により性能の低下を防ぐ．

処理クロック数が一定でないモジュールを複数配置し，その前後のモジュールと，これらのモジュールのいずれの間でも送受信が行えるように回路を構成する．処理クロック数が一定でない複数のモジュールを並行に動作させ，処理が終わったモジュールから順に次のモジュールにデータを送るようにすることで，処理時間を平均化する．本論文の評価実験では，この手法は回路規模の増大を招くため使用していない．

もう一つの方法は，処理時間が一定でないモジュールと，その下流のモジュールの間にバッファを配置し，下流のモジュールはバッファから読み込むようにする．処理時間が一定でないモジュールが早く処理を終えると，バッファに書き込み，次の染色体の処理を開始できる．このようにして，処理時間を平均化する．

5. 回路作成支援ツール

設計した回路を FPGA 上へ実装する際に，その回路の規模と使用メモリ量をもとに，実装可能な FPGA デバイスを決定する必要がある．本章では，回路規模を予測するモデルについて述べた上で回路作成支援ツールについて述べる．

5.1 回路規模予測モデル

提案アーキテクチャに従い設計した GA 回路は，複数のモジュールから構成される．各モジュールの回路規模は個体数の対数と問題サイズの一次関数で予測できることが予備実験の結果から分かっている．また提案アーキテクチャではモジュール間の通信のための制御回路は単純であるため，予測するべき回路全体の回路規模は，使用されるモジュール群の回路規模の総和として近似可能だと考えられる．これらより，回路規模予測モデルを構築する．各モジュールの回路規模を予測するための一次関数の係数は，各モジュールごとに個体数の対数と問題サイズを変化させ，実際に論理合成を行って得た回路規模から，重回帰分析により得る．本論文では，各パラメータを 3 通りに変化させ，モジュールごとに $3 \times 3 = 9$ 通りの論理合成を行った結果から係数を求めることとする．全体の回路規模は，回路全体に含まれる各モジュールの数に，各モジュールの予測回路規模を掛けたものの総和により予測する．

また，各モジュールで用いられる memory block を合計することで，メモリの面から実装可能デバイスが判明する．この使用メモリ量の予測と回路規模予測モデルを組み合わせることで実装可能デバイスが判明する．

実験結果は **6.** で示す．

5.2 回路作成支援ツール

指定した FPGA デバイスに対して，最適な回路のパラメータ値を探索し，回路の RT レベル VHDL 記述を自動的に生成するツールを作成した．ツールは，問題サイズ，個体数，実装先 FPGA などのパラメータから，実装可能である並列度を返す実装判定部と，実装判定部により得られたパラメータ値から，自動的に VHDL で記述した島モデル型 GA 回路を出力する回路作成部からなる．

実装判定部は，前に述べた回路規模予測モデルを用いて判定を行う．

回路作成部は，各モジュールを VHDL で記述したファイルの集合であるライブラリと，回路を自動生成するために必要なテンプレートファイル（図 3）とコンポーネントファイル（図 4）から構成される．テンプレートファイルには library 宣言（図 3 (a)）と entity 宣言（図 3 (b)），architecture（図 3 (c)）の一部が記述されている．コンポーネントファイルは，回路とインタフェースモジュールのコンポーネントインスタンス文が記述されている．回路作成ツールは，テンプレートファイルとコンポーネントファイルを用いて GA 回路を記述する．その際，パラメータに相当するテンプレートファイル内部の%ラベル名%を指定した値に置き換える．このパラメータには，個体数，問題サイズ，並列度がある．コンポーネントファイルでも同様の操作を行う．問題に対応したライブラリとテ

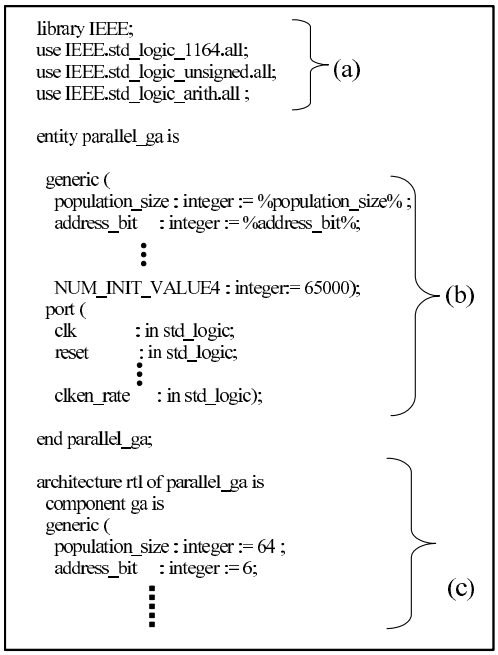

図 3　テンプレートファイル
Fig. 3　Example for template file.

```
interface%num% : interface
 generic map(
    fitness_bit => fitness_bit ,
    num_evaluate_bit => num_evaluate_bit)
 port map( clk => clk ,
    din_num_evaluate1 => line_interface%num_1e%_evaluate ,
    din_num_evaluate2 => line_ga%num_2%_num_evaluate ,
    din_best_fitness1 => line_interface%num_1f%_best_fitness ,
    din_best_fitness2 => line_ga%num_2%_best_fitness,
    dout_num_evaluate => line_interface%num_e%_evaluate,
    dout_best_fitness => line_interface%num_f%_best_fitness);
```

図 4　コンポーネントファイル
Fig. 4　Example for component file.

ンプレートファイル，コンポーネントファイルを用意することで，その問題に対応した島モデル型 GA 回路を出力することができる．

また，これらのツールの実行時間は，**6.** のアプリケーションに適用した場合，いずれも 1 秒以内となり，実用上十分高速であるといえる．

6.　実験結果・評価

本章では，提案アーキテクチャに従った，ナップサック問題と TSP に対する GA 回路の実装について述べる．

次に **5.2** で述べたツールと Altera 社の Quartus II を用いた実験結果について述べる．ターゲットデバイスとして，Altera 社の Cyclone を選び，メモリは，FPGA の内部メモリを使用する．

6.1　ナップサック問題への適用

ナップサック問題は，以下のように定義される．荷物の集合 S とナップサックの大きさが，入力として与えられる．ただし，各荷物には容積と価値が与えられている．ナップサック問題は，荷物の容積の総和がナップサックの大きさを超えず，価値の総和を最大化するような，S の部分集合を求める問題である．

本論文でのナップサック問題に対する実装では，品物の数を s とするとき，染色体の長さを s ビットの長さとする．各ビットが各品物に対応する遺伝子座になる．ある遺伝子座の遺伝子が “1” のときは，対応する品物をナップサックに入れ，“0” である場合は入れないことを示す．s ビットの染色体すべてが，1 クロックごとに各モジュールに入出力されるとする．

［交叉モジュール］

交叉法として，一様交叉を用いる．すなわち，子個体の染色体は，各遺伝子ごとに親個体のどちらか（ランダムに決定）の対応する遺伝子をコピーすることで生成する．

［突然変異モジュール］

突然変異モジュールは，各遺伝子座ごとに，突然変異率で与えられた確率で，ビットを反転する．

［評価モジュール］

適応度は以下のように定義される．

- ナップサックに入っている品物の容積の総和がナップサックの容量を超えている場合，適応度は 0.
- 上記以外の場合，適応度はナップサックに入っている品物の価値の総和．

評価モジュールは，各品物の価値と容積をそれぞれレジスタに保持する．これらのレジスタの内容は，計算開始時に外部より与えられる．容積と利益の総和の計算は，クリティカルパスの延長による最大動作周波数の低下を避けるために，以下のようにして計算する．1 クロック目に，品物を二つずつペアにしてそれぞれのペアの和を並列に計算する．2 クロック目に，1 クロック目で得られた和をペアにして，それぞれの和を並列に計算する．以上を繰り返し，すべての品物の総和を求める．

6.2　巡回セールスマン問題への適用

TSP は，完全グラフ $G(V, E)$（ここで，V は頂点の集合，E は辺の集合）が入力として与えられる．グ

ラフの頂点を都市と呼ぶ．各辺には正の距離が与えられる．TSP は，すべての都市を 1 度ずつ訪れる巡回経路の中で，経路に含まれる辺の距離の合計が最小のものを求める問題である．

TSP では，染色体は，巡回する各都市の番号を各遺伝子として，すべての都市番号を連結したものとする．例えば，51 都市の TSP を扱う場合，各都市を表現するために 6 ビット必要になる．したがって，染色体を表現するために，$51 \times 6 = 306$ ビット必要になる．本実装では，適応度を 16 ビットで表現する．

306 ビットの染色体を 1 クロックで送受信すると，回路規模が大きくなってしまうので，染色体一つの送受信は，複数クロックかけて行う．本実装では，各遺伝子（都市）のビット数を m，訪れる都市数を n とすると，1 クロック当り m ビット，それぞれの染色体を，n クロックで送受信する．

[交叉モジュール]

交叉法として，部分写像交叉（PMX）[14] を採用した．PMX は，致死個体を発生させない交叉法であり，ハードウェア化が比較的容易である．まず二つの異なる遺伝子座を乱数により決定する．二つの間の遺伝子は，親 1 からコピーし，それ以外の部分は親 2 からコピーすることで染色体を生成する．染色体中に 2 回以上現れる都市については，一度も出現しない都市に置換する．

[突然変異モジュール]

突然変異には 2 都市交換を用いる．すなわち，染色体内部の乱数で選択した二つの遺伝子座の遺伝子を入れ換える．

[評価モジュール]

染色体の適応度は，染色体に記された順序に従いすべての都市を巡回したときの経路の距離の総和によって計算される．評価モジュールに，各都市間の距離のテーブルを保持させる．このテーブルは，回路が動作する前にあらかじめ初期値として与えられる．各都市間の距離は，8 ビットの整数で保持する．

評価モジュールが一つの遺伝子を受信すると，受信した遺伝子と一つ前に受信した遺伝子間の距離を距離テーブルより取得し，それを適応度に加算する．

6.3 アーキテクチャ性能評価

提案手法により生成される回路の性能の妥当性を調べるため，生成した回路の解探索能力と処理速度を調べた．性能比較には，問題サイズが 64 のナップサック問題と eil51 と呼ばれる TSP を採用した．eil51 は，

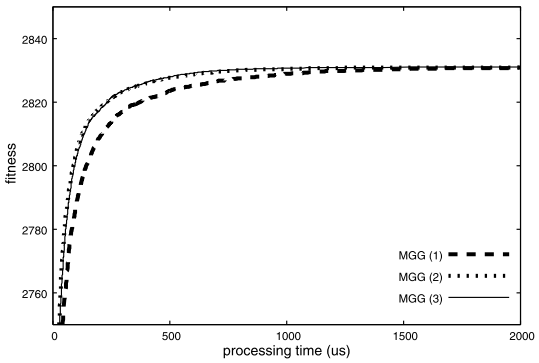

図 5 ナップサック問題の解探索能力
Fig. 5 Search efficiency for Knapsack problem.

問題サイズが 51 の TSP である．

比較対象として SGA を用いる．TSP では，交叉法に PMX と枝交換交叉（EXX）[15] を採用した SGA を用いた．これはソフトウェア GA では，複雑であるが解探索能力の高い EXX の実装が妥当と思われるためである．

ソフトウェア GA は，Pentium 4 2.4 GHz と 256 MByte のメモリをもつ PC 上で実行した．またプログラムのコンパイルには gcc のバージョン 2.95.4 を用いコンパイルオプションとして `-O3` を用いた．

最初に，単位時間当りにどれだけ良い解が得られるかを計測した．提案アーキテクチャの処理時間は，Quartus II でのシミュレーションにより算出した．ターゲット FPGA デバイスとして，Altera 社の Cyclone シリーズを用いた．

それぞれの結果を図 5，図 6 に示す．ただし，図 5 は，適応度が高いほど良い解を，図 6 では低いほど良い解を表している．各図において MGG は提案アーキテクチャの性能を示し，括弧内の数字はその並列度を示している．図 6 の SGA（PMX）と SGA（EXX）は，それぞれの交叉法を用いた場合でのソフトウェア GA の性能を示している．また，MGG（8），MGG（16）は回路規模の点で一つの FPGA 上に実装することは不可能である．そのため計算速度の算出には，MGG（4）を実装した複数の FPGA を用いるものとして計算した．また，提案アーキテクチャに従い設計した回路の最大動作周波数（Maximum Operation Frequency，MOF）を表 1 に示す．また図 5 では，ソフトウェア GA を用いたところ収束まで約 1 秒かかり，結果が図に入らなかったため省いている．

図 5，図 6 より，提案手法による回路は解が収束す

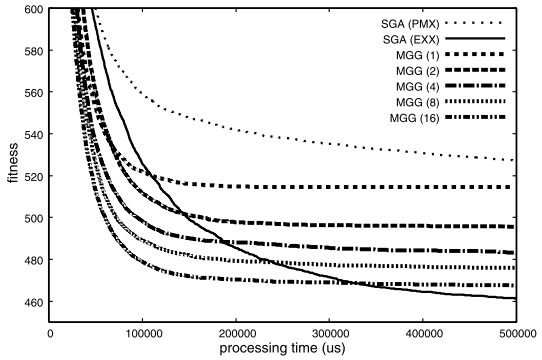

図 6　TSP（eil51）の解探索能力
Fig. 6　Search efficiency for TSP (eil51).

るまでの時間が十分に短く，また，並列度数が高い場合，より高度な交叉アルゴリズムを用いたソフトウェア実装に匹敵する解探索能力を達成できることが分かる．また，提案手法は，回路規模に応じて並列度数を上げることで，GA を高速化し，性能を向上させることができることが分かる．図 6 における SGA（EXX）と比較すると，MGG（16）とほぼ同等の性能を示していることから提案手法において複雑な交叉法をモジュール化することにより，更なる性能向上が可能になると考えられる．

次に，基本アーキテクチャの 1 個体当りの処理時間を表 2 に示す．表 2 より，提案手法により作成した回路は問題サイズが増加するにつれ最大動作周波数が低下するものの，ソフトウェアで実装した SGA より計算速度の点で十分優れた性能を示している．

提案する並列アーキテクチャの利点を明確にするため，提案アーキテクチャにより作成した GA 回路と単純な並列化を行った GA 回路（単純並列化 GA 回路，simple parallel circuit）の性能の比較を行った．単純並列化 GA 回路は，提案アーキテクチャによる並列回路とは異なりすべてのパイプライン間でデータの共有を行う回路である．対象となる問題には，問題サイズ 32 のナップサック問題を使用した．表 3 はそれぞれの回路の並列度ごとの最大動作周波数の比較を示したものである．結果より，提案手法による島モデル型 GA 回路の最大動作周波数の変化は単純並列化 GA 回路に比べ十分低く抑えられていることが分かる．

また，消費電力の結果を表 4 に示す．消費電力は問題サイズの増加に伴い増加しているが，Pentium 4 (2.4 GHz) の TDP （Thermal Design Power）の約 1/80 であり，低消費電力を実現できていることが分かる．

表 1　提案アーキテクチャの最大動作周波数
Table 1　Maximum operating frequency.

Problem	number of pipeline	MOF
Knapsack Problem	1	108 (MHz)
	2	105 (MHz)
	3	102 (MHz)
TSP	1	135 (MHz)
	2	109 (MHz)
	4	110 (MHz)

表 2　1 個体にかかる処理時間
Table 2　Processing time per one individual.

Problem	Problem Size	Basic Architecture		SGA
		Processing time	MOF	
Knapsack Problem	16	7.09 (ns)	141 (MHz)	0.508 (μs)
	32	7.25 (ns)	138 (MHz)	0.765 (μs)
	64	9.26 (ns)	108 (MHz)	1.36 (μs)
	128	8.47 (ns)	118 (MHz)	2.24 (μs)
TSP	51	1.28 (μs)	135 (MHz)	2.07 (μs)
	76	2.06 (μs)	120 (MHz)	2.25 (μs)
	101	2.99 (μs)	113 (MHz)	3.32 (μs)

表 3　比較対象の手法の最大動作周波数
Table 3　Maximum operation frequency of the subject circuits of comparison.

concurrent pipeline	our parallel circuit (MHz)	simple parallel circuit (MHz)
2	106	139
3	119	107
4	109	93
5	114	96

表 4　消 費 電 力
Table 4　Power consumption of basic architecture.

Problem	Device	Total Power
-	Pentium4 (2.4 GHz)	57.8 (W)
Knapsack Problem	FPGA($s = 16$)	293 (mW)
	FPGA($s = 32$)	362 (mW)
	FPGA($s = 64$)	427 (mW)
	FPGA($s = 128$)	700 (mW)
TSP	FPGA($s = 51$)	476 (mW)
	FPGA($s = 76$)	511 (mW)
	FPGA($s = 101$)	611 (mW)

かる．

6.4　回路規模予測モデル

5. で提案した回路規模予測モデルの正確さを評価するために，並列度の異なる複数の回路を構築し，その実測値と予測値との比較を行った．

それぞれの回路における予測値と実測値の比較を図 7，図 8 に示す．ここで，括弧内に記された数字は並列度を示している．回路規模の予測値と実測値の予測誤差は最大 3%であり，提案モデルが実用上十分正確であることが確認された．

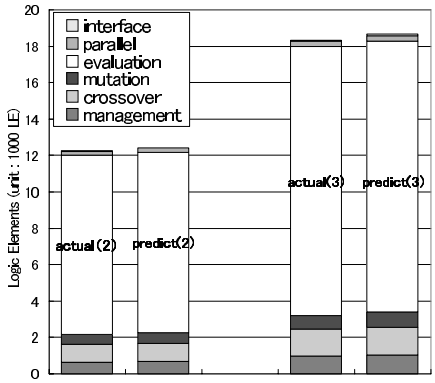

図 7　回路規模予測モデル（ナップサック問題，64）
Fig. 7　Prediction model (Knapsack problem, 64 items).

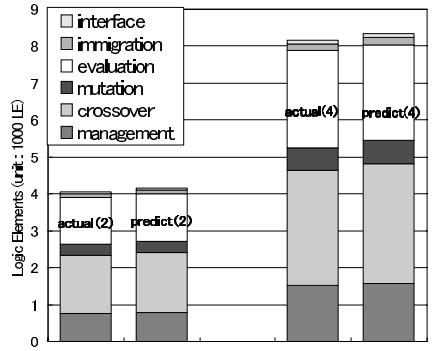

図 8　回路規模予測モデル（TSP，eil51）
Fig. 8　Prediction model (TSP, eil51).

7. むすび

本論文では，FPGA 上への GA の柔軟な実装方法を提案した．提案手法により，デバイスに適した回路を作成することができる．実験結果より，設計した回路は，消費電力が Pentium 4 2.4 GHz の約 1/80 で動作することを確認した．また，回路規模の予測値と実測値の誤差が 3%程度であることを確認した．

今後の課題として，(1) 提案した GA 回路を FPGA デバイス上へ実装し性能を評価すること，(2) 提案手法を様々な問題に対応させるため，それぞれの問題で用いられる遺伝演算子に相当するモジュールを作成すること，(3) 解のコーディング，交叉，突然変異，評価，選択のアルゴリズムを記述するための高水準言語を設計し，RT レベル回路記述生成ツールを作成すること，(4) 論理合成を行うことなく消費電力を予測するモデルを考案すること，などが挙げられる．

文　献

[1] 丸山敦史，柴田直樹，村田佳洋，安本慶一，伊藤　実，“P-Tour：観光スケジュール作成支援とスケジュールに沿った経路案内を行うパーソナルナビゲーションシステム，” 情処学論，vol.12, no.45, pp.2678–2687, Dec. 2004.

[2] 坂無英徳，岩田昌也，樋口哲夫，“遺伝的アルゴリズムを用いた高解像度 2 値画像データの可逆符号化，” 情処学論，vol.45, no.5, pp.1460–1470, May 2004.

[3] L. Layuan and L. Chunlin, “Genetic algorithm-based Qos multicast routing for uncertainty in network parameters,” Proc. 5th Asian-Pacific Web Conf. (APWeb 2003), pp.430–441, Xian, China, April 2003.

[4] E. Cantú-Paz, “A survey of parallel genetic algorithms,” Technical Report 97003, Illinois Genetic Algorithms Laboratory, 1997.

[5] 北浦　理，浅田英昭，松崎元昭，河合隆光，安藤秀樹，島田俊夫，“パイプライン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，” 計測自動制御学会論文集，vol.35, no.11, pp.1496–1504, Nov. 1999.

[6] 若林真一，小出哲士，八田浩一，中山喜勝，後藤睦明，利根直佳，“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，” 情処学論，vol.41, no.6, pp.1766–1776, June 2000.

[7] 小林亮一，阿部正英，川又政征，“遺伝的アルゴリズムを用いた不連続閉曲線抽出手法の FPGA 上での実現，” 信学技報，CAS2000-131, DSP2000-189, CS2000-151, March 2001.

[8] P. Graham and B. Nelson, “A hardware genetic algorithm for the traveling salesman problem on SPLASH 2,” Proc. 5th Int. Workshop on Field Programmable Logic and Applications (FPL’95), LNCS 975, pp.352–361, Aug. 1995.

[9] B. Shackleford, E. Okushi, M. Yasuda, H. Koizumi, K. Seo, T. Iwamoto, and H. Yasuura, “High-performance hardware design and implementation of genetic algorithms,” in Hardware Implementation of Intelligent Systems, pp.53–87, Physica-Verlag GmbH, Heidelberg, 2001.

[10] C. Aporntewan and P. Chongstitvatana, “A hardware implementation of the compact genetic algorithm,” Proc. 2001 Congress on Evolutionary Computation (CEC2001), pp.624–629, Seoul, Korea, May 2001.

[11] T. Kitani, Y. Takamoto, K. Yasumoto, A. Nakata, and T. Higashino, “A flexible and high-reliable HW/SW co-design method for real-time embedded systems,” Proc. 25th Int. Real-Time System Symposium (RTSS2004), pp.437–446, Lisbon, Portugal, Dec. 2004.

[12] H. Satoh, I. Ono, and S. Kobayashi, “Minimal generation gap model for GAs considering both exploration and exploitation,” Proc. 4th Int. Conf. on Soft Computing (IIZUKA’96), pp.494–497, Fukuoka, Japan, Sept. 1996.

[13] 小野　巧，佐藤　浩，小林重信，“サブシーケンス交換交叉と GT 法に基づくジョブショップスケジューリングの

進化的解法,” 電学論（C），vol.117, no.7, pp.888–895, July 1997.
[14] S.M. Sait and H. Youssef, “Genetic algorithms (GAs),” in Iterative Computer Algorithms with Applications in Engineering, pp.109–181, IEEE Computer Society, Los Alamitos, 1999.
[15] 前川景示, 玉置　久, 喜多　一, 西川示韋一, “遺伝的アルゴリズムによる巡回セールスマン問題の一解法,” 計測自動制御学会論文集，vol.31, no.5, pp.598–605, May 1995.

（平成 17 年 8 月 10 日受付，12 月 22 日再受付）

橘　　達弘

平 16 奈良先端科学技術大学院大学情報科学研究科情報処理学専攻博士前期課程了．現在同大情報科学研究科情報処理学専攻博士後期課程在学中．遺伝的アルゴリズムの研究に従事．

村田　佳洋　（正員）

平 12 奈良先端科学技術大学院大学情報科学研究科修士課程了，平 15 同大学院大学情報科学研究科博士後期課程了，平 15 年 4 月より奈良先端科学技術大学院大学情報科学研究科助手．遺伝的アルゴリズムとエージェント技術の研究に従事．

柴田　直樹

平 13 大阪大学大学院基礎工学研究科情報数理系専攻博士後期課程了．平 13 年 4 月より奈良先端科学技術大学院大学情報科学研究科助手．平 16 年 4 月より滋賀大学経済学部情報管理学科助教授．分散システム，遺伝的アルゴリズムと ITS などの研究に従事．

安本　慶一

平 3 阪大・基礎工・情報工学卒．平 7 同大大学院博士後期課程退学後，滋賀大学経済学部助手．平 14 より奈良先端科学技術大学院大学情報科学研究科助教授．博士（工学）．分散システム，マルチメディア通信システムに関する研究に従事．IEEE/CS, ACM 各会員．

伊藤　　実　（正員）

昭 52 阪大・基礎工・情報工学卒．昭 54 同大大学院基礎工学研究科博士後期課程退学後，同大助手．昭 58 工博（阪大）．平 3 カナダウォータールー大学数学部客員準教授．平 5 奈良先端科学技術大学院大学情報科学研究科教授．データベース理論，分散システム等に関する研究に従事．ACM，IEEE/CS 各会員．