Field-Programmable Gate-Array による進化的計算の高速化

丸山 勉[†] 船津輝宣[†] 関 峰伸[†]
山口佳樹[†] 星野 力[†]

遺伝的アルゴリズム，複雑適応系等の進化的計算は，幾種類かの進化的オペレータが多数の個体に繰り返し適用されるという特徴を持つ．専用ハードウェアにより，これらの処理をパイプライン化，並列化することにより著しい高速化が期待されるが，それらのオペレータは各問題ごとに異なるため画一的なハードウェア化は困難である．Field-Programmable Gate Array は，問題ごとにその構成を書き換え可能な LSI であり，各問題に最適な回路構成を提供することができるため，進化的計算システムを構築するのに適している．本論文では，2 個の FPGA からなる小規模なシステム構成により，いくつかの進化的計算の問題において，ワークステーション（SUN Ultra-Sparc 200 MHz）の 80～130 倍の速度向上率が得られることを示す．

A Field-Programmable Gate-Array System for Evolutionary Computation

Tsutomu Maruyama,[†] Terunobu Funatsu,[†] Minenobu Seki,[†]
Yoshiki Yamaguchi[†] and Tsutomu Hoshino[†]

In evolutionary computation, evolutionary operations are applied to a large number of individuals (genes) repeatedly. The computation can be pipelined (evolutionary operators) and parallelized (a large number of individuals) by dedicated hardwares, and we can expect high performance. However, details of the operations depend on given problems and vary considerably. Systems with Field Programmable Gate Arrays can be reconfigured and realize the most suitable circuits for given problems. In this paper, we show that a hardware system with 2 FPGAs and SRAMs can achieve 80~130 times of speedup compared with a workstation (200 MHz) in some evolutionary computation problems.

1. はじめに

遺伝的アルゴリズム，複雑適応系等の進化的計算は，一般に非常に多くの計算時間を必要とするが，これらの問題の多くは，比較的単純な演算から構成される幾種類かの進化的オペレータを，多数の個体に対して繰り返し適用するという特徴を持つ．したがって，専用ハードウェアにより，これらの進化的オペレータの計算をパイプライン化し，また多数の個体に対して同時に適用（並列化）することにより，著しい高速化が期待されが，これらのオペレータの特徴は問題ごとに異なるため，画一的なハードウェア化は困難である．

Field-Programmable Gate Array（FPGA）は，その構成を，実行時に問題に合わせて再構成可能な LSI であり，各問題ごとに個別の演算機能を提供することができるため，進化的計算のためのハードウェアプラットフォームを構成するのに適している．近年，FPGA の回路規模，処理速度の向上は著しく，非常に高速な計算が可能となってきている．特に，回路規模の向上による効果は大きく，進化的計算を行うための最小限の遺伝子（個体）数を 1 チップ上で扱うことが可能となったため，複数チップへの分割にともなう付加的な回路およびチップ間での配線遅延による演算速度の低下，またチップ間配線の制約等による汎用性の低下等の影響を受けない高速なシステムを構築することがが可能となってきている．

本論文では，FPGA 2 個（計 200 K ゲート相当）と SRAM からなる小規模なシステム構成により，いくつかの遺伝的アルゴリズムの問題（ALL1 問題，ナップサック問題，グラフ分割問題）と複雑適応系の問題（繰返し囚人のディレンマゲームを用いた集団の進化

† 筑波大学構造工学系
Institute of Engineering Mechanics, University of Tsukuba

の計算）において，ワークステーション（SUN Ultra-Sparc 200 MHz）の 80～130 倍の速度向上率を得られることを示す．

2. 進化的計算の特徴

2.1 並 列 性

進化的計算は，生物の進化の過程にヒントを得たものであり，遺伝子（Gene）を多数用いて，交叉（Crossover），突然変異（Mutation），生存競争〔評価（Evaluation），選択（Selection）〕等の進化的オペレータを繰り返し適用し（この 1 サイクルを世代と呼ぶ），より優れた遺伝子，また，より多様/複雑な振舞いを示す集団を得ようとするものである．

したがって，進化的計算の処理の基本的な流れは，すべての遺伝子に対して，交叉，突然変異，評価，選択等の進化的オペレータを繰り返し適用するという形となる．このため，ハードウェアにより，

(1) これらの進化的オペレータの計算のパイプライン化

(2) 複数の遺伝子にこれらのオペレータを同時に適用（データ並列化）

することにより，大きな速度向上率が期待される．

また，各オペレータにおいて疑似乱数の生成等が行われるが，ハードウェアでは，これらの処理も他の演算と並列に実行可能であり，より高い演算速度が実現できる．

2.2 ビット演算の多用

進化的計算では，通常，各遺伝子はビット列で表現可能であり，各オペレータの処理においてもビット単位の演算が多用されるが，マイクロプロセッサにおける最小演算単位はバイトであり，ビット列を扱うためには，シフト/マスク演算等が必要となるため，バイト列として扱うのが通常である（バイトデータには 0/1 のみを格納）．

これに対し，ハードウェアでは，ビット列の操作が容易であるため，ビット列の一部を必要に応じてレジスタ（1 ビット幅）上に実現することにより，レジスタの効率的な利用および高速な演算を実現することができる．また，ビット列のデータをそのままメモリに格納することにより，メモリ使用効率およびメモリ入出力の効率化を図ることができる．

2.3 スケーラビリティ

進化的計算においては，選択以外の処理は非常に局所性が高く，一般にたかだか 2 個の遺伝子間での処理である．選択処理に関しても，多数（N 個）の遺伝子の間で比較/選択を直接行うのではなく，N 個の遺伝子をある程度の数（P 個）からなるいくつかの島（I 個，ただし $N = P * I$）に分割し，その島の間で数世代に 1 遺伝子を交換することにより，N 個の遺伝子を直接比較し選択する場合と同様の結果を得られることが知られており，アイランドモデルと呼ばれている[7]．

このモデルにおいては，各島の遺伝子数があまりに少ないと期待されるような結果を得ることができないが，各島の遺伝子数が数十あれば十分であり，近年の FPGA の回路規模であれば，評価部分を 1 チップ上に実装可能である．このため，

(1) すべての個体を島ごとに外部メモリに保持

(2) 各島の遺伝子 P 個を同時に計算（島ごとに順次計算）

(3) 数世代に 1 回，島の間で遺伝子（1 個）を交換

という手順により，簡単なハードウェアシステムにより，大規模な問題を扱うことができる．

また，この方式を用いて，より多くのハードウェアを投入することにより，より高い処理速度を得ることができるが，本論文では，FPGA 2 個という小規模な構成により，どの程度の速度向上率が可能かを明らかにする．

3. FPGA システムの構成

図 1 に FPGA システムの構成を示す．図 1 において，FPGA-A は選択，突然変異等の進化的計算固有の処理を，FPGA-B は各遺伝子の評価をパイプライ

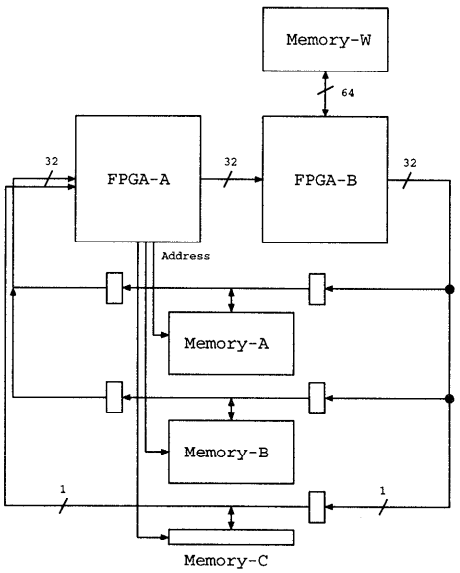


図 1 システム構成
Fig. 1 Block diagram of the system.

ン的に処理する．まず各遺伝子は Memory-A（または B）から読み出され，FPGA-A，FPGA-B で処理された後 Memory-B（または A）に書き込まれる．このとき FPGA-B で計算された各遺伝子の評価値は，直接 FPGA-A へフォワードされる．また，Memory-W は FPGA-B がワークエリアとして用いる．Memory-C はアイランドモデルに用いられる．図 1 において，I 個の島からなる $P*I$ 個の遺伝子に関してアイランドモデルを実行する場合には，Memory-A，B には，それぞれ異なる I 個のベースアドレスから各島の P 個の遺伝子が格納され，Memory-C には数世代に 1 回 P 個の遺伝子のうちの 1 つが格納される．次の世代における Memory-A または B からの k 番目の島の読み出しにおいて Memory-C からは $((k+1)\%I)$ 番目の遺伝子が読み出され，こちらが用いられる．なお，メモリはすべて SRAM とする．

現在 Power Medusa MEB200-A100S（三菱電機マイコン機器ソフトウェア社製）という FPGA ボードを 2 枚用いて評価を進めている．これは，正方形のボードの中心に ALTERA 社 EPF10K100（100 K ゲート相当）が 1 つ載ったものである．この正方形のボードを水平に置き，その 4 辺に他の FPGA ボード，またはユニバーサルボードを接続し，システムを拡張することができる．

4. FPGA による遺伝的アルゴリズムの実行

遺伝的アルゴリズムを用いて，より少ない数の遺伝子により，より短時間に，より優れた解を得るには，遺伝的アルゴリズムとヒューリスティックアルゴリズムを組み合わせたハイブリッドアプローチの方が優れていることが，これまでの数々の研究により示されている．

ここで示すようなヒューリスティックアルゴリズムを用いない単純な遺伝的アルゴリズムは，効率的なヒューリスティックアルゴリズムが知られていない場合，また，手軽にある程度の品質の解を見つけたい場合に有効な手法であるが，一般に多くの遺伝子を用いた緩やかな収束が必要であるため，非常に多くの計算時間を必要とする．しかし，本論文において示すようにハードウェア化により著しい速度向上が実現できること，また，アイランドモデルにより多数の FPGA を用いた場合，ほぼ線形な速度向上率が期待できることから，大規模なシステムの構築により，実用的な時間内に，より多くの困難な問題に対して，有益な解を得ることが期待される．

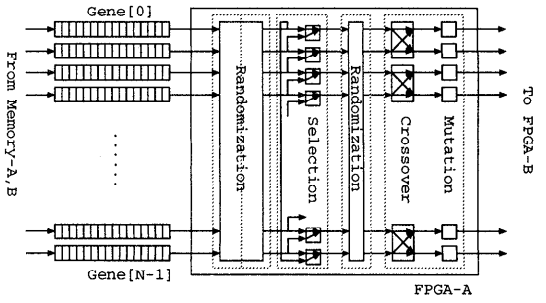


図 2 遺伝的アルゴリズムのパイプライン処理
Fig. 2 Pipeline processing of genetic algorithm.

4.1 従来の研究

FPGA を用いた遺伝的アルゴリズムの高速化の研究としては文献 1)，2) 等がある．1) は遺伝的アルゴリズムの教科書[3] に見られる関数の最適化を行ったものであり，文献 2) は Splash2[4] 上で，Travelling Salesman Problem を実行したものである．これらは，規模の小さな FPGA を用いたことによる分割損および動作周波数の低下，逐次的なソフトウェアのアルゴリズムをそのままハードウェア化することに主眼をおいていること等により，高速化率はワークステーションの 3〜5 倍程度にとどまっている．

また，文献 9) では，SIMD 方式により多数の FPGA を接続し，各 FPGA 内に GA 向き機能を持つプロセッサを実現することにより高速な処理の実現を図っているが，パイプライン処理が有効に活用されていない．

4.2 パイプラインの構成

遺伝的アルゴリズムのパイプライン処理の様子を**図 2** に示す．図 2 では，32 個の遺伝子（Gene）が同時に FPGA-A に 1bit ずつ入力され処理される．図 2 中の点線はパイプラインのクロックごとの切れ目を示す．各遺伝子の評価は，FPGA-B で行われる．

4.2.1 選択処理

選択（Selection）に関してはトーナメント方式[5] を用いる．トーナメント方式とは，N 個の遺伝子からランダムに 2 個を選択し，その評価値の高かった方を選択するという方式である．この方式は，最良の遺伝子は 2 回選択され，最も悪い評価値を持つ遺伝子は選ばれないという特徴を持ち，ランク戦略とほぼ等価であることが知られている[6]．図 2 においては遺伝子をランダムに選択する代わりに Randomization のステージを用いて各入力に対する出力先を乱数により決定している．このステージは冗長性を持たせた多段網と乱数発生器により簡単に実現できる．この方式を用いることにより，局所的なデータ処理のみで選択処理が可能となり，ハードウェアによる高速な選択処理が可能

表 1 All 1 問題の処理速度比
Table 1 Speedup: All 1 problem.

遺伝的アルゴリズム全体の処理（32 遺伝子）		
Work Station*	実行時間（sec）	速度比
lrand48()	52.0	0.059
M 系列	19.8	0.156
random()	12.8	0.241
合同乗積法	3.08	1.00
FPGA System**	0.0250	123
評価部分のみ		
Work Station*	1.12	1.00
FPGA System**	0.0212	52.8

* SUN Ultra-Sparc 200 MHz
** 33 MHz

となる．FPGA-B から送られてきた各評価値は，選択のステージで 1 ビットずつ比較され，どちらの遺伝子を選択するかが決定される．評価値の各ビットはこのステージで捨てられ，それ以降のデータは次ステージへ送られる．選択の後，再び Randomization のステージが用いられる．このステージは同じ遺伝子が交叉ステージの同一ユニットに入力されることを防ぐことのみが目的なので，簡単な構成をとることができる．

4.2.2 交叉・突然変異

交叉（Crossover）は，2 つの遺伝子間で情報の一部を交換する操作（たとえば，遺伝子 1 と遺伝子 2 の間で，K 番目から L 番目のビットを交換する）であり，突然変異（Mutation）は遺伝子の各ビットの値を乱数に応じて他の値（たとえば 0 から 1 へ）に変える操作である．

4.3 All 1 問題

All 1 問題とは，遺伝子のすべてのビットが 1 になったときに最高点を得るという簡単な問題である．この問題について，FPGA 1 チップを用いて実装し評価を行った（外部 SRAM は未使用）．1FPGA 内に実現するために遺伝子数 32，遺伝子長 16 ビットとした．また，同一 FPGA 内での処理であるので（遺伝子もすべてレジスタに格納），パイプラインの簡略化が可能であり，3 段ですべての処理が構成されている（遺伝子の格納にさらに 16 段）．

この問題における FPGA の最高動作周波数は 40 MHz であるが，以下の性能評価には，各問題ごとの速度向上率の違いを明確化するために，すべて共通の 33 MHz を動作周波数として用いる．

表 1 に性能を示す．これは 40000 世代まで実行した場合のものである．表 1 において，lrand48()，random() は C 言語の標準ライブラリ，M 系列は FPGA システムで用いた疑似乱数生成手法，合同乗積法は乗算/除算を行わないようにシフト/マスク演算のみで疑似乱数の生成を行った場合の性能である．表 1 から分かるように，ワークステーション上での実行性能は乱数の生成時間に大きく影響される．進化的計算における疑似乱数の精度はそれほど高いものは必要ではないので，以下合同乗積法の性能と比較することにする（より処理時間を要する乱数生成アルゴリズムほど，より優れた乱数列を生成するが，ここで示す遺伝的アルゴリズムの評価においては，上記のどのアルゴリズムを用いても探索能力は同様である）．シフト/マスク演算のみによる合同乗積法の計算は，Ultra Sparc 上で，整数演算 5 命令程度であるので，全体の処理時間に対する比率はかなり小さい．

32 遺伝子の処理において，評価部分で約 53 倍（1 遺伝子あたり 1.66 倍），全体としてパイプライン処理の効果により 123 倍の速度向上率が得られている．遺伝子長が十分に長ければ，パイプラインがほぼ 100% 動作し，このときの速度向上率は約 146 倍となる．また，回路規模をさらに大きくし同時に処理可能な遺伝子数を増やすことにより，より高い速度向上率を期待することができる．

比較に用いたソフトウェアのアルゴリズムを**図 3** に示す．RAND() は，上述したシフト/マスク演算のみによる合同乗積法による乱数生成マクロである．また，値の交換に用いている SWAP_char(), SWAP_char_p() もマクロであり関数呼び出しは行われていない．プログラム中には，余りを計算するために演算子%が用いられているが，評価に用いた問題の定数がすべて 2 の巾乗であるため，シフト演算が行われており，割り算は行われていない．交叉は，1 点交叉であり，2 つの遺伝子間で，ある位置（乱数により決定）以降のデータの交換を行っている．FPGA では，交叉点数を増やしても演算速度は変わらないが（乱数生成器が並列に動作しているため），ソフトウェアでは乱数の生成が 1 回ですむ 1 点交叉が最も高速である．突然変異は，乱数の値によりビットの反転を行う操作である．選択処理では，トーナメント方式を用いており，まず遺伝子個数分の乱数列を生成し，この乱数列を用いて，2 つの遺伝子の評価値の比較を行い，優れた評価値を持つ遺伝子をコピーしている．また，プログラム中の gene[], ngene[] は，ポインタであり，ポインタの値を入れ替えることにより，世代が変わるごとに遺伝子のコピーが行われることを防いでいる．後述する遺伝的アルゴリズムでも，処理の流れはまったく同じであり，評価の部分のみが異なる．

4.4 ナップサック問題

ナップサック問題は，N 個の物体（それぞれ異なっ

```
ga()
{
  for (g = 0; g < G; g++) {   /* G 世代まで計算を行う */
    /* 交叉 (一点交叉) */
    for (i = 0; i < N; i+=2)  /* 遺伝子数 N 個 */
      for (j = RAND() % K; j < K; j++)  /* 遺伝子長 K */
        SWAP_char(gene[i][j], gene[i+1][j]);

    /* 突然変異 & 評価 & 選択処理準備 (乱数生成) */
    for (i = 0; i < N; i++) {
      for (perf[i] = 0, j = 0; j < K; j++) {
        if (RAND() < MUTATION_RATE)
          gene[i][j] = !gene[i][j]; /* 突然変異 */
        perf[i] += gene[i][j];  /* 評価値の計算 */
      }
      rand_tbl[i] = RAND() % N; /* 選択処理準備 */
    }

    /* 選択 (トーナメント選択) */
    for (i = 0; i < N; i++) {
      a = rand_tbl[i];
      b = (i < N-1)? rand_tbl[i+1]: rand_tbl[0];
      x = (perf[a] > perf[b])? a: b;
      for (j = 0; j < K; j++)
        ngene[i][j] = gene[x][j];
    }

    /* ポインタの交換 */
    for (i = 0; i < N; i++)
      SWAP_char_p(gene[i], ngene[i]);
  }
}
```

図 3 All 1 問題のソフトウェアアルゴリズム
Fig. 3 A software algorithm for All 1 problem.

た重さと価値を持つ）の中から何個かを選択し，これを重さ W まで収納可能なナップサックに入れるときに，ナップサックに入れられた物体の価値を最大にするような組合せを見つけるという問題である．

遺伝子長 N の遺伝子を用いて，その各ビットの値が 1 である場合にその物体が選択されているものとし，選択されている物体の総重量が W を超えたときに適当なペナルティを評価値に課すことによって，遺伝的アルゴリズムを用いて解くことができる（ただし，重すぎるペナルティを与えると評価関数に崖が生じ遺伝的アルゴリズムにとっては難しい問題となり，ペナルティが軽すぎると閾値を超えたところに収束する可能性が生じる．遺伝的アルゴリズムを用いたナップサック問題に関しては，より洗練された手法が提案されているが，ここでは，最も単純な上記の方法を用いることにする）．

評価ステージは，各遺伝子の i 番目のビットの値を受け取ると同時に，Memory-W からそのビットに対応する重さ，価値を読み出し（あらかじめ Memory-W にこれらの情報が書き込まれているとする）加算することにより簡単に実現できる．32 個の遺伝子の演算を同時に行うので，1 遺伝子あたり 2 個，合計 64 個の加算器が必要となる．重さ，価値の合計が 16 ビットを超えない範囲であれば，64 個の加算器が 1FPGA 内に配置可能であり，また，各加算は 1 クロックで終了する（ALTERA EPF10K100 は，4992 個のロジックエレメント（FPGA の基本演算素子）を持ち，16 ビット Accumulator は，16 ロジックエレメント（計 1024 ロジックエレメント）で構成できる）．なお，この回路における FPGA 内部の最高動作周波数は 50 MHz である．

表 2 ナップサック問題の処理速度比
Table 2 Speedup: knapsack problem.

全体の処理	実行時間（sec）	速度比
Work Station*	1.29	1.00
FPGA System**	0.00950	136
評価部分のみ		
Work Station*	0.36	1.00
FPGA System**	0.00865	41.6

* SUN Ultra-Sparc 200 MHz & 合同乗積法
** 33 MHz

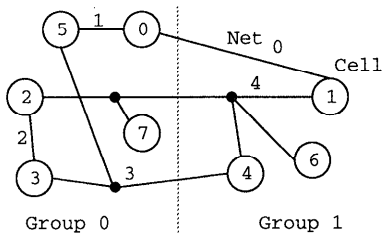


図 4 グラフ分割
Fig. 4 Graph partitioning problem.

表 2 に実行時間の計算値を示す．これは，256 個の物体に対するナップサック問題を 32 遺伝子を用いて 1024 世代まで実行したときの実行時間である．評価部分の速度向上率は，約 42 倍で前節の性能よりやや悪いが，これは遺伝子中のビットの値が 1 となる比率が前節の問題では次第に高くなるのに対しこの問題ではある程度以下にとどまることによるものである．1 の比率が高まるに従ってソフトウェアによる計算時間は加算のために増加するが，ハードウェアによる処理時間は，1 の比率にかかわらず一定である．

4.5 グラフ分割問題

グラフ分割問題とは，いくつかのセルとセル間を結ぶネットで構成されたグラフ（**図 4**）が与えられたとき，セルを 2 つのグループ（0,1）に 2 分割し，グループ間にまたがるネット数が最小となるような分割を求める問題である．

グラフ分割問題の評価は以下のようにして行うことができる．

(1) 各セルごとに，そのセルに接続されているネットの一覧表を用意する（図4のセル0ではネット0,1）．

(2) 総ネット数と同じ大きさを持つテーブル T（2ビット幅初期値は0）を用意する．

(3) グループ間にまたがるネット数を数えるカウンター CS を0に，グループ0に所属するセル数を数えるカウンター $G0$ を0に初期化する．

(4) 遺伝子長をセル数とし，遺伝子の各ビットにセル Ci が属するグループ番号（0,1）を割り振る．

(5) 各セル（Ci）に対し以下の処理を行う．

(a) Ci がグループ0に所属していたら $G0$ に1加算する．

(b) Ci に接続されているすべてのネット Nk について T から Nk に関する情報 Sk を読み出す．

(c) Sk の値を以下のように更新する．

Sk	New Sk	
	Ci==0	Ci==1
00 →	01，	10
01 →	01，	11
10 →	11，	10
11 →	11，	11

この演算は，

$$\text{New } Sk \leftarrow Sk|(1 << Ci);$$

ただし，Ci の値は0か1

で計算することができる．

(d) Sk の値が，01または10から11に変わったら（そのネット上にグループ0と1に所属するセルがそれぞれ存在するということなので）CS に1加算する．

(e) New Sk を T に書き戻す．

(6) すべての Ci に関する計算を終了したら評価値を計算する．評価値は，グループ間でのセルの片寄りを防ぐために，

$$Pi = CS/(G0 \times (\text{総セル数} - G0))$$

で与えられる．

図5にFPGAを用いた評価ステージの実現方式を示す．Memory-A，Bには各セルに接続されているネットの一覧表（1ワードに2ネット分保持．FPGA-Aはネット情報をそのままFPGA-Bへフォワードする），Memory-Wには各ネットに関する状態（1ワードに32遺伝子分）が保持されている．**表3**に図5におけるFPGA-Bの処理のタイミングを示す．表3において，

(1) まずセル Ci に接続されているネット情報（Na, Nb）がFPGA-Bに入力され，次いで遺伝子の i 番目のビット（Ci のグループ番号）が入力される．

(2) Nb, Nc 等の′はそれぞれ Ci, Cj 等に接続されている最後のネットであること示す．これにより次の遺伝子情報（Cj, Ck）の入力が行われる（この制御はFPGA-Aが行う）．

(3) *は，その遺伝子の評価において最初にネットが参照されたことを示す．これにより，Memory-Wの初期化作業が不要となる

である．このように1ネットあたりの評価は2クロックで終了する．また，FPGA-B-INが32ビットの場合，扱うことができる最大ネット数は16384（14ビット）本となる（これ以上の場合には1ネットずつの入

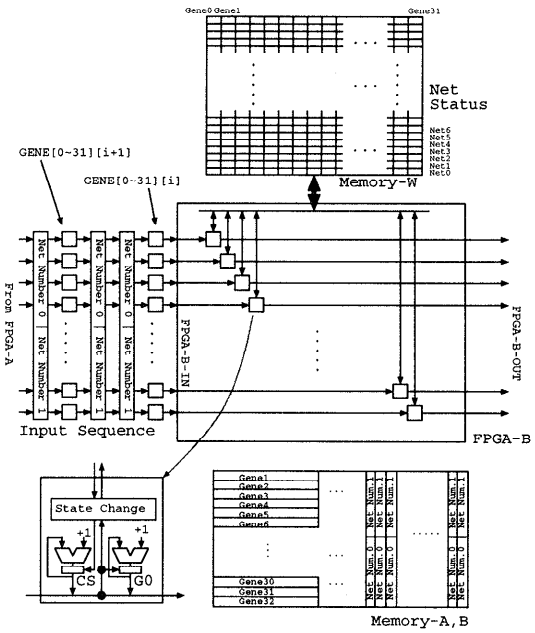


図5 グラフ分割問題の評価

Fig. 5 Evaluation stage of graph partitioning problem.

表3 グラフ分割問題の処理タイミング

Table 3 Process timing: graph partitioning problem.

	FPGA-B-In	Memory-W	FPGA-B
$t0$	Na* & Nb'*		
$t1$	Ci		
$t2$		Read Sa	
$t3$		Read Sb	Update Sa
$t4$	Nc'* & Nd*	Write Sa	Update Sb
$t5$	Cj	Write Sb	
$t6$	Ck	Read Sc	
$t7$		Read Sd	Update Sc
$t8$	Na & Nc'	Write Sc	Update Sd
$t9$		Write Sd	
$t10$		Read Na	
$t11$		Read Nc	Update Sa

表 4 グラフ分割問題の処理速度比
Table 4 Speedup: graph partitioing problem.

全体の処理	実行時間（sec）	速度比
Work Station*	2.13	1.00
FPGA System**	0.0261	81.6
評価部分のみ		
Work Station*	1.24	1.00
FPGA System**	0.0261	47.5

* SUN Ultra-Sparc 200 MHz & 合同乗積法
** 33 MHz

力とする必要があり，やや性能が低下する）．

表 4 に実行時間の計算値を示す．これは乱数を用いて生成したグラフ（2048 セル，3096 ネット，1 セルあたりの平均ネット数 3）に関して，32 個体を用いて 64 世代遺伝的アルゴリズムを実行したときのものである．各遺伝子の評価値の計算には乗算/除算器が必要であるが，ネット数およびグループ 0 に属するセル数の加算に用いられる加算器を用いて 1 ビットずつ加算/減算を繰り返すことによって行うものとする．この乗算/除算は，この問題の場合 16 ビット演算であり，評価時間は少なくとも遺伝子長 2048 よりは大きいため，全評価時間に占める割合は非常に小さい．評価部分の速度向上率は約 48 倍（1 遺伝子あたり 1.5 倍）とナップサック問題の場合よりやや大きいが，遺伝子 1bit あたりの評価が 1 クロックで終了しないため，パイプライン処理の効果が低減し，全体としては約 80 倍にとどまっている．

4.6 スケーラビリティ

上記の評価では，メモリアクセス速度の影響（FPGA システムはすべて SRAM であるのに対し，ワークステーションでは問題規模が大きいとキャッシュミスヒットの多発等が起きる）を抑えるために小規模な問題を用いた．

遺伝的アルゴリズムにおいて，大規模な問題を扱うためには，

(1) 問題規模に合わせて遺伝子長を長くする．
(2) 遺伝子数を多くする．
(3) より大きな世代数まで計算を行う．

必要がある．単純な遺伝的アルゴリズムでは，遺伝子長は問題規模に比例し，また，全計算時間は，遺伝子長×遺伝子数×世代数に比例する．遺伝子数は，問題規模から一意に決まるが，必要とされる遺伝子数と世代数は，必要とされる解の質に依存し，より良い解が必要な場合には，より多くの遺伝子数と世代数が必要となる．

たとえば，グラフ分割問題で 20 K セルの問題を扱う場合には，遺伝子長は 20 K ビット（ソフトウェアでは一般に 20 K バイト使用）となる．512 個の遺伝子，4 K 世代の計算が必要であるとすると（単純な遺伝的アルゴリズムにおいては少なめの設定である），その計算時間は，ワークステーションにおいて約 6 時間となる．この場合においても，FPGA システムの速度向上率は変わらない．ただし，512 個体を一度に扱うことはできないので，2.3 節で述べたようにアイランドモデルを用いる必要がある（ソフトウェアでは，アイランドモデルを用いるとループ制御がやや複雑になるが，その処理時間はほとんど変わらない）．

表 5 囚人のディレンマゲームの利得表
Table 5 Payoff matrix: iterated prisoner's dilemma.

		player-2 C	player-2 D
player-1	C	3/3	0/5
	D	5/0	1/1

FPGA システムにおいては，大規模な問題の実行においても，カウンター等の演算部分は完全にパイプライン化されているため，動作周波数には影響はないが，カウンターのビット幅の増加等による回路規模の増加により，評価部分が 1 チップに実現できなくなることが起こりうる．この場合には，より大規模な FPGA を用いる，または，複数の FPGA を用いる等の手段が必要となるが，複数の FPGA チップを用いた場合には，チップあたりの速度向上率は低下する．また，問題規模が大きくなると SRAM ではなく，SDRAM 等を用いる必要が生じる．この場合には，メモリの多バンク化等によるランダムアクセス時のスループットの確保〔本システムでは 2 バンク構成（SDRAM 内部の 2 バンクと合わせて計 4 バンク）で十分である〕，および遅延時間を隠すために，評価部分の計算のより深いパイプライン処理の実現が必要となる．

5. FPGA による複雑適応系の実行

5.1 繰返し囚人のディレンマゲーム

囚人のディレンマゲームとは，同じ犯罪により別々の監獄に入れられた 2 人の囚人が，パートナーを裏切る（Defect，以下 D と略す）/協調（黙秘）する（Cooperate，以下 C と略す）ことにより，**表 5** のような得点を得られるとき，どのように振る舞うべきかという問題である．行動の選択が 1 回限りであれば D による期待値の方が高い．繰返し囚人のディレンマゲーム（Iterated Prisoner's Dilemma Game，以下 IPD と略す）は，これを繰り返し行うものである．

5.2 IPD による進化的計算

ある集団を用いた進化的計算において，その集団内

の各個体が，このIPDのルールに従い他のすべての個体と対戦を行い，その総得点を評価値とするものとする．このとき，毎回Cを選択する者同士が対戦するとつねに3点を獲得することができ，集団の全個体が毎回Cを選択すれば，集団としての最大値を得る．しかし，ここに毎回Dを繰り返すものが侵入すると毎回5点という大きな得点を得ることができるため，この個体はしばらくの間，順調に増殖することができる．しかし，やがて毎回Dを選択する者同士の対戦が増え，得られる点数が低下し，他の行動パターンを持つ個体の増殖を許すことになる．

このようにIPDは非常に単純なモデルでありながら非常に複雑な振舞いを示すことが知られており，複雑適応系の研究にしばしば用いられている．さらに2個体間の対戦において，お互いの行動が相手に伝わるまでにノイズが入り，異なった行動が伝えられるモデルが提案されており，これによりさらに複雑な挙動を観測することができる．

5.3 IPDの計算モデル

IPDの計算のモデルとしてLindgren[8]のモデルを用いる．Lindgrenのモデルは計算が容易であるとともに，突然変異による多様な個体（遺伝子）の出現が可能であるため，IPDの計算モデルとして興味深いものの1つである．

Lindgrenのモデルは，

(1) 相手と自分の過去の行動を記憶（初期値は1，最大5個）し，これを用いて表（最大32ビット）を引くことにより次の行動を決定する．
(2) 過去いくつまでの行動を記憶するかは突然変異によって変わる（±1個）．これに合わせて表の大きさも変わる〔2倍（内容を複製）または1/2（前半または後半をランダムに選択）となる〕．
(3) 表の値は突然変異によって変わる．
(4) 自分の行動はあるエラー率を持って相手に伝えられる（自分の持つ過去の行動履歴と相手の持つ行動履歴が異なる）．
(5) すべての個体が互いに対戦を行い
計 $(N*(N-1)/2)$ 回
その得点の合計を各個体の評価値とする．

というものである．

5.4 パイプラインの構成方式

Lindgrenのモデルでは，個体長が35ビットと短い〔3ビット（過去の行動を記憶する長さを保持）＋32ビット（表）〕，すべての個体間での対戦を行いそれが評価値となるという特徴を持つため，遺伝的アルゴリズムの問題のように各個体を1ビットずつ同時に処理するのではなく，**図6**に示す方式をとる．

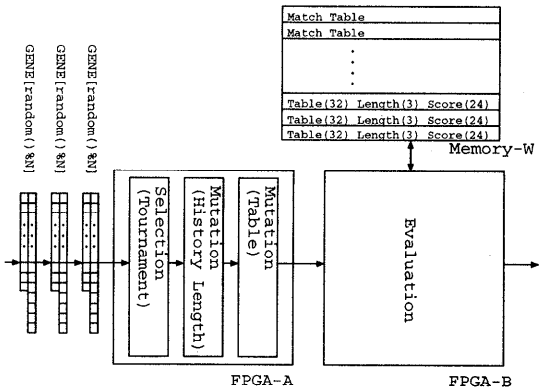


図6 IPDにおけるパイプライン処理
Fig. 6 Pipeline processing of IPD problem.

表6 IPDの評価部分の実行時間
Table 6 Speedup: iterated prisoner's dilemma.

	実行時間（msec）	速度比
Work Station*	7.23	1.00
FPGA System**	0.0675	107

* SUN Ultra-Sparc 200 MHz & 合同乗積法
** 33 MHz

図6において，

(1) FPGA-A はランダムに1個体を選択しMemory-A（またはB）から，その個体とその個体の評価値を読み込む（2クロックで1個体）．
(2) 連続する2個体の評価値を比較し評価値の大きな方を選択する（トーナメント選択）．
(3) 突然変異のステージには2つあり，表の大きさ，表の値の2種類の突然変異処理を行う．
(4) 全個体間の対戦を実現するために，FPGA-Bは，受け取った個体をMemory-Wにいったん格納する．
(5) FPGA-Bは，すべての個体を受け取ると定められた順番で（Memory-Wに順番情報が格納されているものとする），個体をFPGA内へ読み込み，個体間での対戦（1対戦あたりの繰返し回数2048回）を行う．
(6) すべての対戦が終了すると，FPGA-Bは個体とその得点をMemory-A（またはB）へ書き込む．
(7) 上記の処理を1世代としこれを繰り返す．

FPGA（ALTERA EPF 10K100）では，同時に16対戦（32個体間での対戦）が可能である．

表6に評価部分（32個体を用いた16対戦）の処理時間の計算値を示す．この表に示した計算時間は，

```
hist_a = 0; hist_b = 0;              /* 各個体の過去の手の履
歴 */
for (i = 0; i < N; i++) {
  an = a = tbl_a[hist_a];            /* 次手の決定 */
  bn = b = tbl_b[hist_b];            /* 次手の決定 */
  if (RAND() < ERROR_RATE) an = !a;  /* 通信エラーの発
生 */
  if (RAND() < ERROR_RATE) bn = !b;  /* 通信エラーの発
生 */
  sum_a += score_table[a][bn];       /* 評価値の計算 */
  sum_b += score_table[b][an];       /* 評価値の計算 */
  /* 履歴の更新 */
  hist_a = ((hist_a<<2) | (a<<1) | bn) & 0x1f;
  hist_b = ((hist_b<<2) | (b<<1) | an) & 0x1f;
}
```

図7 IPDの評価部のプログラム
Fig. 7 A program for IPD evaluation.

FPGA1チップにおける繰返し計算の1回分であり，実際にIPDにおける複雑な挙動を観測するためには，256個体程度を用いて，30000世代ほどの計算を行う必要がある[8]．このための計算時間は，ワークステーションにおいて約120時間ほどである．

この問題においては評価部分の計算時間が支配的であるため表6の結果が全体の性能となる．表6から，約107倍の速度向上率が期待できることが分かる．1対戦あたりの速度向上率は，約6.7倍であり，これまでの例題よりかなり高いが，これは，まず，ハードウェアでは32ビットのテーブルをレジスタにより実現することにより，1回の選択（CまたはD）に関する処理が1クロックで可能であること，また，ノイズに用いられる乱数生成も同時に演算可能であること，さらに，1対戦には2個体の計算が必要なことによるものである．

図7にIPDの評価部分のプログラムを示す．tbl_a[]，tbl_b[]は，各個体の次手を決定するための表（32ビット）であり，score_tbl[]は，表5に示した利得表である．FPGAでは，このループ1回の処理を2サイクルで実行可能である（利得表はレジスタで実現している）．

6. おわりに

現在，我々はField-Programmable Gate Array（FPGA）を用いた大規模な進化的計算のための専用システムの研究を進めており，これまでに，2個のFPGAを用いた小規模なシステム構成により，いくつかの問題において，ワークステーションに比べて80～130倍の速度向上率が得られること確認した．進化的計算においては，パイプライン化，並列化が容易であり，また，ビット演算が多用されるため，このような高速な処理が可能となると考えられる．

現状のFPGAを用いたシステムでは，問題が複雑になると，評価時のメモリアクセスの増加等により，評価部分の処理時間が他の部分の処理時間より大きくなるため，パイプラインが効率的に働かなくなり全体の処理速度が低下するという問題点がある．現在，高速に書き換え可能（書き換えのための内部SRAMを複数面保有し，瞬時に回路構成を切替え可能）なFPGAの提案等が行われている．これらのFPGAを用い，パイプラインの乱れに応じて演算回路の切替えを行い（評価部がボトルネックとなる場合には，他の計算を行っているFPGAを適宜書き換え，評価計算を行わせる），ハードウェア使用効率をより高めることにより，より複雑な問題においても，より高速な処理が実現できると考えられる．

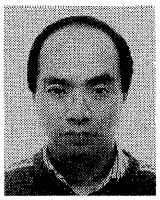
今後，より複雑なアルゴリズムを必要とする問題への展開，より大規模なシステムの開発を行う予定である．

参考文献

1) Scott, S.D., Samal, A. and Seth, S.: HGA: A Hardware-Based Genetic Algorithm, *Int. Symposium on Field-Programmable Gate Array*, pp.53–59 (1995)
2) Graham, P. and Nelson, B.: Genetic Algorithm In Software and In Hardware – A Performance Analysis of Workstation and Custom Computing Machine Implementations, *FPGAs for Custom Computing Machines*, pp.216–225 (1996).
3) Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley (1989).
4) Buell, D.A., Arnold, J.M. and Klenfelder, W.J.: *Splash2: FPGAs in a Custom Computing Machine*, IEEE Computer Society Press (1996).
5) Goldberg, D.E. and Deb, K.: Messy Genetic Algorithms: Motivation, Analysis and First Results, *Complex System*, Vol.3, pp.493–530 (1989).
6) Goldberg, D.E. and Deb, K.: *A Comparative Analysis of Selection Schemes Used in Genetic Algorithms*, pp.69–93 (1991).
7) Gordon, V.S. and Whitley, D.: Serial and Parallel Genetic Algorithms as Function Optimizer, *5th International Conference on Genetic Algorithms*, pp.177–190 (1993).
8) Lindgren, K.: Evolutionary Phenomena in Simple Dynamics, *Artificial Life II*, pp.295–312

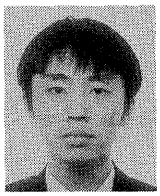
(1991).
9) 佐野ほか：FPGA による SIMD 型 GA マシンの設計，情報処理学会報告，97-ARC-125-6 (1997).

(平成 10 年 8 月 21 日受付)
(平成 11 年 3 月 5 日採録)



丸山　勉（正会員）

昭和 33 年生．昭和 62 年東京大学大学院工学系研究科情報工学専門課程博士課程修了．同年日本電気（株）入社．並列オブジェクト指向言語，並列遺伝的アルゴリズム，並列マシン Cenju の開発/研究に従事．平成 9 年より筑波大学構造工学系助教授．書き換え可能なハードウェアを用いた計算の高速化に関する研究に従事．

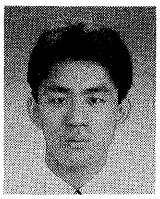


船津　輝宣（学生会員）

1975 年生．1997 年筑波大学第 3 学群工学システム学類卒業．1999 年同大学大学院工学研究科修士課程修了．複雑適応系の計算の研究に従事．

関　峰伸（学生会員）

1976 年生．1998 年筑波大学第 3 学群工学システム学類卒業．同年筑波大学大学院工学研究科入学．FPGA による遺伝的アルゴリズムの高速計算，将棋の高速計算の研究に従事．



山口　佳樹（学生会員）

1975 年生．1998 年筑波大学第 3 学群工学システム学類卒業．同年筑波大学大学院理工学研究科入学．FPGA を用いた複雑適応系の高速化とその解析の研究に従事．

星野　力（正会員）

1965 年京都大学工学研究科博士課程電気工学専攻修了．工学博士．京都大学原子エネルギー研究所を経て，80 年より筑波大学構造工学系教授．原子力工学の研究を経て，77 年より並列計算機 PAX シリーズの研究と開発，89 年より人工生命，進化的計算に関する研究に従事．