A-5　並列分散型遺伝的アルゴリズムＶＬＳＩ“GAP/D”における動的移住調整

Dynamic Migration Control in Parallel-Distributed Genetic-Algorithm VLSI “GAP/D”

小林憲貴†　吉田紀彦‡　楢崎修二†
Kazutaka Kobayashi　Norihiko Yoshida　Shuji Narazaki

1．　はじめに

遺伝的アルゴリズム（以下，GA と略す）において，複数の個体集団を並行に進化させる分散型 GA は，高速化ならびに高い収束性能をもたらす技術としてよく知られている．分散型 GA では，各集団での局所解への誤収束を避けるために，集団間で個体を移住させる．この移住の頻度は収束性能を大きく左右するが，事前に決定しておくことは難しく，動的に決定する種々の手法が提案されている．

我々は GA を VLSI ハードウェアで直接に実行する GA-VLSI「GAP」の設計開発を進めてきており，これを複数配置して分散型 GA を実現するハードウェアもすでに設計している [1][2]．しかしこの第１版では，まず GA-VLSI の実現性を実証する目的から，移住頻度は固定している．そこで，分散型 GA のネットワーク実装に関する我々自身の研究成果 [3] も踏まえて，本研究では，ハードウェア実装に適した移住頻度の動的調整の方式を導入した「GAP/D」を設計し，論理シミュレーションでその効果を確認した．

この GA-VLSI はこれまでレジスタ・トランスファ・レベル（RTL）のハードウェア記述言語 SFL [4] を用いて設計を進めてきており，GAP/D も最初は SFL で設計を行った．その後，より抽象度の高いシステムレベル設計言語 SpecC [5] を用いて改めて設計を行った．これは，この２～３年で急速に注目を集めつつあるシステムレベル設計に関して，比較実験ともなっているので，そこから得られた知見などにも触れる [6]．

2．　ＧＡＰの概要

GAP（Genetic Algorithm Processor）は問題を特定しない汎用の GA-VLSI であり，問題ごとの評価値計算 VLSI である FEP（Fitness Evaluation Processor）と組み合わせて用いる．GAP は集団メモリ，選択，遺伝的操作（交叉・変異），乱数発生などのモジュールから構成され，GA における選択→遺伝的操作→集団への格納という一連の処理をパイプライン的に実行する．そして，個体の適応度を評価する際に FEP に計算を依頼する．

GAP は，他の GA-VLSI に比較して次の特徴を持つ．

- 定常状態 GA の採用によるパイプライン高速化と収束性能の向上
- 単純化トーメント選択方式による回路量の削減と高速化
- 複数 FEP による並列化，複数 GAP による分散化

特に並列化では，複数 FEP によって複数個体の評価値を並列に計算する．GA 処理において最も計算負荷が高いのは評価値計算であるため，速度が大きく向上する．一方で分散化では，複数 GAP によって複数の個体集団を並行に進化させることで，収束性能が向上してより良い解が得られるようになる．

3．　移住頻度の動的調整

GAP の分散化では複数 GAP 間の個体ビット列の移住を行うが，第１版ではこれを遺伝的操作で子の個体を作る毎に行っていた．しかし，移住頻度は収束性能を大きく左右するため，収束状況を監視しつつ動的に調整することが望ましい．それには様々な方式が提案されているが，ハードウェア実装を考えると，有効なだけでなく高速な方式でなければならない．そこで，以前の研究成果も踏まえて，次のような方式を導入した．

局所解への誤収束が起きると，収束カーブの微分係数，すなわち評価値の向上率が鈍化する．そこで，評価値の微分が小さくなったら移住を 行うようにする．この方式の有効性はすでに確認しており，また差分の計算は容易なことから，ハードウェアで高速に実行できる．具体的には，個体集団の評価値の総和について，遺伝的操作のループ毎に前回と今回との差分をとり，それがある閾値を下回ったら移住を行う．なお，この閾値は問題の収束の様子に応じて設定することになる．

4．　ハードウェア設計とシミュレーション実験

前節で述べた移住頻度の動的調整の方式を実際にハードウェアとして設計した．前回の値を保持するレジスタと今回の値との差分を計算し，それとあらかじめ設定した閾値との比較回路からの出力に応じて，移住を行うと判定したならば移住回路を起動する．

移住回路では，隣の GAP に個体を受け取るように信号を送出し，この信号を受けた GAP は個体を受け取って，自ら生成した個体の代わりに集団メモリに格納する．信号がない場合は自ら生成した個体をそのまま格納する．以上を GAP が２台の場合について示すと，図１のようになる．このように個体移住は一方向の流れになり，全体で循環構造を構成する．

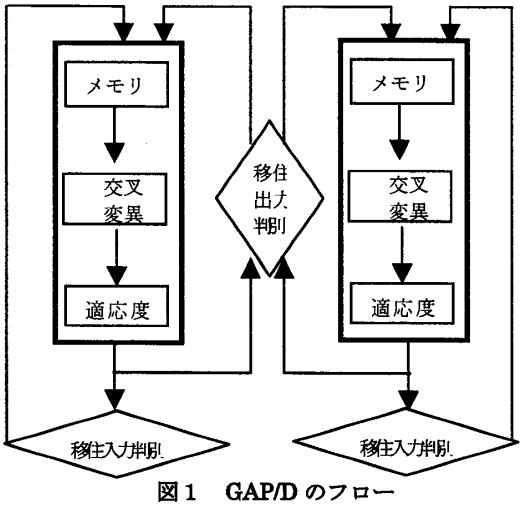


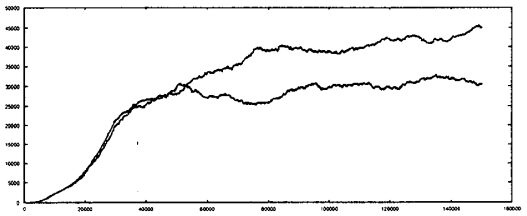
図１　GAP/D のフロー

† 長崎大学　Nagasaki University
‡ 埼玉大学　Saitama University

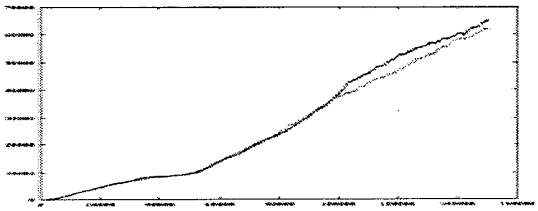
以上のように設計したハードウェアについて，論理シミュレーションによる動作実験を行い，収束状況を測定して本方式の有効性を検証した．まず，回路追加によって，1回の GA ループに要するマシンサイクルは，17 から 21 に増加した．

次に，GA システムの評価によく使われる RoyalRoad 関数 [7] を例題とし，2 台の GAP で実験を行った．以前の GAP，すなわち毎回移住の場合（閾値∞に相当する）の結果を図 2 に，動的移住方式で閾値を 16，40 とした場合の結果をそれぞれ図 3 と図 4 に示す．グラフの各線は 2 台それぞれの GAP の収束状況を表している．また，図 5 は 4 台の GAP，閾値 16 の場合である．（X 軸は時間軸、Y 軸は適応度の合計）

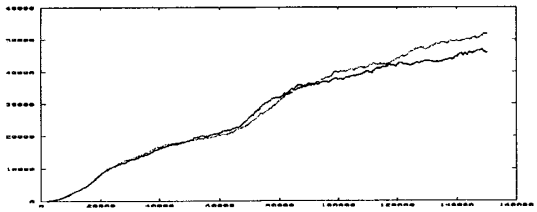
以上を比較すると，閾値 16 ではいったん鈍化した収束カーブが（移住によって）改めて向上していくこと，すなわち初期収束からの回復が見られ，最終的な適応度が毎回移住の場合よりも向上している．閾値 40 では，移住がより頻繁に起きるようになり，最終的な適応度が小さい．なお，さらに閾値を大きくしていくと，収束状況が元の毎回移住方式に近づくこと，GAP の台数を増やすと，最終的な適応度がさらに向上することも確認した．



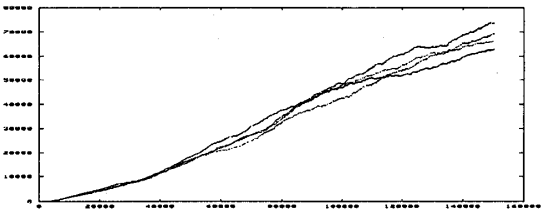
X 軸の最大値　160000
Y 軸の最大値　50000
図 2　以前の GAP



X 軸の最大値　160000
Y 軸の最大値　70000
図 3　GAP/D（閾値 16）



X 軸の最大値　160000
Y 軸の最大値　60000
図 4　GAP/D（閾値 40）



X 軸の最大値　160000
Y 軸の最大値　80000
図 5　GAP/D（GAP4 台：閾値 16）

5．　記述の比較

本研究で設計した GAP/D は，ハードウェア記述言語 SFL，次いで C をベースとするシステムレベル設計言語 SpecC [5] を用いて記述している．両者を比較すると，ハードウェア記述言語では状態遷移など詳細に定義する必要があったのが，システムレベル設計言語では言語に備わっているパイプライン構文や並列処理構文を用いて抽象化することができ，また型が充実していることから，設計効率が向上した．ビジュアル設計環境の存在も貢献している．しかし，システムレベル設計言語でも遺伝的操作におけるビット列処理などはそのまま記述する必要があり，言語にその機能が乏しいことから新たに定義しなければならないのが問題であった．

6．　おわりに

本研究では，GA-VLSI の分散化における収束性能の向上を目指し，ハードウェア実装に適した形で移住頻度の動的調整を行うべく，評価値の差分に着目する方式を導入し，論理シミュレーションでその効果を確認した．現在は，さらに詳細なシステムレベル設計と論理シミュレーションを進めている．

参考文献

[1] N. Yoshida and T. Yasuoka, "Multi-GAP: Parallel and Distributed Genetic Algorithms in VLSI", Proc. 1999 IEEE Int'l Conf. on Systems, Man and Cybernetics, Vol.5, pp.571-576 (1999)

[2] N. Yoshida, T. Yasuoka, T. Moriki and T. Shimokawa, "VLSI Hardware Design for Genetic Algorithms and Its Parallel and Distributed Extensions", Int'l J. of Knowledge-Based Intelligent Engineering Systems, Vol.5, No.1, pp.14-21 (2001)

[3] N. Yoshida and R. Araki, "Efficient Implementation of Distributed Genetic Algorithms on Network of Workstations", Proc. Second Int'l ICSC Symp. on Soft Computing, pp.336-340 (1997)

[4] http://www.kecl.ntt.co.jp/parthenon/index_j.htm

[5] http://www.ics.uci.edu/~specc/

[6] 小林，並列分散型遺伝的アルゴリズム VLSI のシステムレベル設計とハードウェア設計，長崎大学卒業論文 (2002)

[7] M. Mitchell and S. Forrest, "Fitness Land-scapes: Royal Road Functions", in Handbook of Evolutionary Computation (T. Back, D. Fogel and Z. Michalewicz eds.), Oxford (1997).