木構造並列プロセッサによる遺伝的プログラミングの高速処理

High Performance Genetic Programming using Parallel Proccessor with Tree-network

余村 昌則* 福士 将 * 堀口 進 *

Masanori Yomura Masaru Fukushi Susumu Horiguchi

概 要

遺伝的プログラミング (Genetic Programming : GP) の専用ハードウェア化による処理の高速化が望まれている．GP は木構造で表現された染色体の評価値の計算に莫大な時間を要する．そこで，演算器を木構造に結合した木構造並列プロセッサを用い，パイプライン処理が可能なアーキテクチャを提案し，設計したシステムを Field Programmable Gate Array (FPGA) 上に実装した．

1 はじめに

様々な動的適応問題を高率良く解く方法として，GA や GP のような進化的手法が注目されている．GP は，遺伝的アルゴリズム (Genetic Algorithms : GA) を木構造へと拡張した手法として，自律ロボットの制御プログラム取得等，数多くの問題に対して有効性が検証されている．このような進化的手法は，染色体の評価値の計算に多くの計算時間を必要とするため，並列計算機による高速処理が研究されてきた．しかしながら，並列計算機を用いて高速処理を行う場合，自律ロボット上に搭載することは困難であり，小型で高速な専用処理ハードウェアが必要とされている [1].

これまでに，ハードウェアにより GA の計算時間を短縮する試みとして，いくつかの GA エンジンが提案されている．例として進化オペレータとして突然変異だけを用いてハードウェア化した研究が行われている [3]．しかしながら，これ等の手法は GA に特化した手法であり，構造を持った情報を扱う GP に適用するのは困難である．これまで GP のハードウェア化に関する研究はほとんどない．そこで本稿では，ハードウェアとして実現可能な GP 手法を提案する．GP では木構造で表現された染色体の評価値の計算に莫大な計算時間を要する．本稿では，演算器を木構造に結合した木構造並列プロセッサを用い，各ノードに位置的に対応する染色体の情報を格納することで，パイプライン処理が可能なアーキテクチャを提案し，計算時間の削減が可能なことを示す．また，このアーキテクチャを用いることにより，従来高速なハードウェア処理が困難であった木構造上での交叉などの処理が実現可能となることを示す．

2 章では，一般的な GP の概略を説明する．3 章ではハードウェア実装を考慮した GP 処理アーキテクチャを提案する．4 章では提案アーキテクチャを用いた設計事例について説明し，5 章で本稿をまとめる．

2 遺伝的プログラミング

GP は，GA を構造を持った問題に適用できるように拡張した手法であり，与えられた条件を満たすプログラムを自動的に取得するための手法である．このため GP では解候補となるプログラムを木構造の式として表現し，これを一般に染色体と呼ぶ．また，木構造の染色体の中で，ノードに相当する部分のデータを，遺伝子と呼ぶ．また，遺伝子を格納する染色体中の位置のことを，遺伝子座と呼ぶ．この染色体を進化させることで，目的とする解を探索する．

ある評価関数に対して，評価値の高い染色体を次世代に残すために，GA や GP は一般に次のような処理を繰り返す．

(1) 交叉 : 二つの染色体のある部分を取り換え，新しい染色体を作成する．(2) 染色体の評価 : 評価関数に基づき，各染色体の適合度を計算する．(3) 選択 : 評価値に基づき，次世代に残す染色体を決定する．

3 木構造並列プロセッサ

GP は各染色体の評価値計算を独立に行えるため，高い並列性を持つ．多数の独立な計算を高速化するには，一般に，複数のプロセッサで並列に処理するアプローチと，演算機能を固定したプロセッサでパイプライン的に処理するアプローチが考えられる．本稿では，制御が簡単であることと，ハードウェア量を少なく抑えられることから，後者のアプローチをとる．本稿で提案するアーキテクチャの全体図を図 1 に示す．

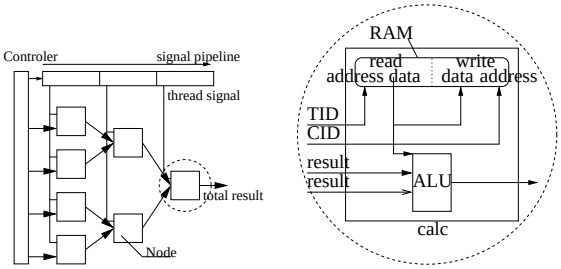

図 1: システムのモデル図

システムは，演算器を木構造に結合した構成であり，染色体の遺伝子各ノード上に分散して配置する．木構造中で同じ深さのノードの集合をレイヤーと呼ぶ．図 1 の signal pipeline により，染色体 ID を順次送出する．ある時刻では同一レイヤーに同一のパイプライン ID が入力されるため，各レイヤーはパイプライン的に処理されることになる．すなわち，図 1 の各ノードは染色体中の遺伝子座に相当し，各ノードには，全染色体の該当する遺伝子座の遺伝子が格納される．この他に，計算を行う ALU と乱数発生器を持つ．同一ノード内で，子となる染色体の遺伝子にアクセスするために，染色体 ID を用いる．各ノードはシグナルの与え方により，次の二つの処理を選択的に実行する．

- **計算:** 各ノードでは，パイプラインから入力されたスレッド ID(TID) を処理すべき染色体 ID とみなし，これをアドレスとして遺伝子メモリにアクセスし，当該ノードでの遺伝子 (命令コード) を取り出す．この命令コードと下位ノードで計算された結果を用いて，当該ノードでの計算を行う．計算結果は上位ノードに送信される．この計算フローにより，レイヤーレベルでパイプライン処理される．

*北陸先端科学技術大学院大学, JAIST

- **交叉:** ここでは交叉手法として，GPUX[2] を用いる．GPUX では通常の部分木を単位とした交叉と異なり，木構造中の同じ遺伝子座間でのみ遺伝子を交換する．この手法は通常のものと比較して精度が高く，また木の深さが変化しないという特徴があるためにハードウェア化に適している．これを実現するため，ノードに TID と CID の二つの ID を入力する．アドレス TID から遺伝子を読出し，アドレス CID へと書き込むことで，情報のコピーを実現する．また，この読出しと書込みをパイプラインにより分割する．最初にアドレス A から A の遺伝子を読出す．次に A の遺伝子をアドレス B に書込む．また，これと同時に B の遺伝子を読出す．最後に，B の遺伝子を A に書き込む．この情報の交換を各ノードにおいてある確率で行い，交叉を実現する．

これらのノードを制御する制御部について説明する．制御部は以下のようなフェーズを遷移しながら繰り返し処理されるように連続して制御信号を発行する．

- **初期化フェーズ:** システムが動作を開始するとまず，各ノードの遺伝子に対して乱数による初期化を行う．これは，交叉と同じ動作を行うことで実現する．具体的には，読み出しアドレスとしてある特別な ID が与えられたときに，メモリの代りに乱数発生器から値を読出す．これを対象とする ID に書き込むことで実現する．全染色体の初期化が終了すると，評価フェーズに移る．

- **評価フェーズ:** 評価を行う染色体の ID をパイプラインに投入する．最上位ノードから出力された値を染色体の評価値として，評価値メモリに格納する．全染色体の評価が終了すると，選択フェーズに移る．

- **選択フェーズ:** 一般の GP では，交叉は二つの親からその部分木を取り換えた子染色体 2 つを生成するため，一時的にある世代の染色体数は増加し，その全染色体の集合の中から，評価値の高い染色体を選択する．しかしながら，ハードウェア実装上の観点から，ここでは次のような方法をとる．

  1. 染色体を，ランダムに二つ選ぶ．
  2. 評価値の高い方を A, 低いほうを B とする．
  3. 読出しアドレスを A，書込みアドレスを B とする．これにより，評価値の高い染色体から低い染色体へと遺伝子のコピーが行われる．
  4. 以上を交叉比率で決められた回数分だけ繰り返す．終了すると評価フェーズへと移る．

  この手法では，パイプラインストールを避けるために，集合全体を用いた選択を行うのではなく，交叉に用いる二つの親染色体のみを用いた選択を行っている．この手法はハードウェア化が容易であるため，文献 [1] 等のシステムで多く用いられる手法であり，その有効性が検証されている．

- **突然変異:** 初期化と同様に，指定する染色体と乱数発生器からの出力とを，ある確率で混合することにより，容易に突然変異を実現することが可能である．ただし，交叉比率と精度の関係についての考察が未完のため，今回は実装していない．

以上で説明したように，各遺伝子を木構造並列プロセッサの該当する位置に分散配置することにより，signal pipeline に入力する染色体 ID の組み合わせを制御するという単純な制御で，GP の評価値の計算，交叉処理をパイプライン処理を実現している．

4 試作システムと評価

4.1 システムの試作

表 1 の仕様に基づき，ハードウェア記述言語 Verilog-HDL で回路設計を行い，得られた回路を ALTERA 社の EP20K200EFC484-2X 上に実装した．このときの回路の使用量が表 2 となる．表 2 において，Total logic elements は FPGA で使用されたロジックエレメント数を表し，Total ESB bits はメモリに使用したブロック数を表している．問題の規模にもよるが，単純な問題であれば 1 つの FPGA チップ内に充分実装することが可能である．

表 1: 試作した GP の仕様

染色体数	32
命令の種類	4
交叉比率	16/1 世代
突然変異	なし
木の深さ	4 (15 ノード)

表 2: 回路使用量 (EP20K200EFC484-2X)

	使用量	使用率
Total logic elements	1964	20%
Total ESB bits	1088	1%

4.2 システム評価

提案したアーキテクチャにより遺伝的計算が実現できることを確かめるために，関数最大化問題を実装し検証した．関数最大化問題は，非終端記号={NOPL, NOPR, ADD, SUB}, 終端記号={X0,X1} からなる関数木 F(X) の出力を最大にする組み合わせを自動取得する．NOPL と NOPR は左と右の下位ノードの値をそのまま出力する関数である．ここでは 1000 世代分の計算を行って時間を評価した．このときの実行時間をソフトウェア上で同様の処理を行った場合と比較した結果を表 3 に示す．結果が示すとおり，約 150 倍高速に処理されていることが確認された．ただし，ここでハードウェアのクロック周波数は，FPGA を実装しているボードの周波数にしているが，設計回路はより高速なクロック周波数でも動作可能であることを補足する．

表 3: 実行時間の比較

	動作クロック周波数	実行時間
ハードウェア	33 MHz	1.5ms
ソフトウェア	450 MHz(CPU)	230ms

5 まとめ

本稿では，GP を小型でかつ高速に処理できるハードウェアアーキテクチャの提案を行った．演算器が木構造に結合されている木構造並列プロセッサに，染色体の位置的に対応する遺伝子を分散配置することで，GP のパイプライン処理を可能とし，交叉処理を実現することを示した．応用例として，関数最大化問題を FPGA に試作実装することにより，提案アーキテクチャのハードウェア実装が可能なことを示し，高速に動作可能なことを確かめた．今後の課題として，提案した手法の定量的な評価等を含めた考察を行う．

参考文献

[1] Kajitani, I., Hoshino, T. and et al: A Gate-Level EHW-Chip : Implementing GA Operations and Reconfigurable Hardware on a Single LSI, *Int'l. Conf. on Evolvable System (ICES98)* (1998).

[2] Poli, R. and Langdon, W.: On the Search Properties of Different Crossover Operators in Genetic Programming, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pp. 298–301 (1998).

[3] 樋口哲也, 村川正宏: 関数レベルにおけるハードウェア進化, 産業図書, pp. 271–296 (1997).