システムＬＳＩ設計技術　107－20
(2002. 11. 28)

遺伝的プログラミングのための専用ハードウェアと専用コンピュータシステムの開発

藤原　一成†　　若林　真一†

† 広島大学大学院工学研究科
〒 739-8527 東広島市鏡山一丁目 4 番 1 号
E-mail: †{nari,wakaba}@bsys.hiroshima-u.ac.jp

あらまし　進化型計算手法の 1 つに遺伝的プログラミング (GP) があり，プログラム生成や機械学習，推論など様々な分野の問題解決に用いられている．しかし，GP は汎用逐次処理コンピュータ上にソフトウェアとして実現した場合，複雑な問題に対して多大な計算時間を要するという問題点を持つ．そこで，本研究では GP を高速に実行可能な新しいコンピュータシステムを提案する．提案システムは選択や交差などの問題に依存しない処理を行う GP 専用ハードウェアと，問題に依存する評価値計算を並列に処理する複数のプロセッサからなる．このシステム構成により任意の問題に対して GP の高速実行が可能となる．

キーワード　遺伝的プログラミング，GP 専用ハードウェア，評価の並列化．

A Computer System with Custom Hardware for High-Speed Execution of Genetic Programming

Kazunari FUJIWARA† and Shin'ichi WAKABAYASHI†

† Graduate School of Engineering, Hiroshima University
4-1 Kagamiyama 1 chome, Higashi-Hiroshima, 739-8527 JAPAN
E-mail: †{nari,wakaba}@bsys.hiroshima-u.ac.jp

Abstract　Genetic programming is one of the evolutionaly computation techniques. It is used for problem solving of various fields, such as program generation, machine learning and reasoning. However, GP has a problem of requiring great computation time to a complicated problem, when it is realized as software on a sequential computer in general. Then, in this research, we propose a new computer system which can efficiently execute GP. The proposed system consists of custom hardware, which performs processes independent of problems such as selection and crossover, and parallel processors, which performs evaluation of individuals. This system makes it possible to execute GP very efficiently for solving any problems.

Key words　genetic programming, custom GP hardware, paralellization of fitness caluculation.

1. はじめに

遺伝的アルゴリズム (Genetic Algorithm, GA) は，自然界の遺伝メカニズムに基づく探索アルゴリズムとして，1970 年代に John Holland によって開発された [1]．GA は VLSI 設計自動化や画像処理，スケジューリング問題等の工学の様々な分野における多くの大規模最適化問題を効率良く解くヒューリスティック手法として知られている．

遺伝的プログラミング (GP) [2] は，進化型計算手法を用いてより一般的な問題解決を行うため，GA を拡張して個体として構造的な表現を扱えるようにしたものである．GP では，解を表現するデータ構造としてグラフ構造（特に木構造）を扱えるように GA の手法を拡張している．一般に LISP の S 式は木構造で記述できるので，GP が扱う個体はプログラムとみなすことができる．すなわち，GP は GA のように個々の問題に対する解を直接探索するのではなく，個々の問題を解決するプログラムを探索する．GP は，さまざまなロボットの制御プログラムの生成，回路の合成，ゲームのプログラム，人工知能におけるさまざまな問題解決，機械学習などに応用され，探索の有効性が示されている．

しかし GP は問題によっては非常に大きな人口 (一例として 1000 万) を扱うことを必要とし，また逐次改良法などとは異

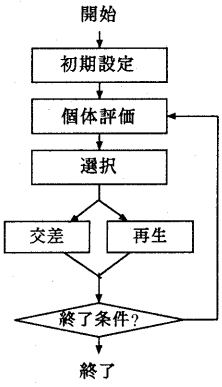


図1　GPの実行フロー

なり，問題空間を大域的に探索するため，非常に多くの計算時間を要する．その解決策としてこれまでに処理の並列化やハードウェア化などのさまざまなアプローチがとられている．本研究では並列処理とハードウェア化を組み合わせた新しいコンピュータシステムを提案する．提案するシステムはGPにおける選択や交差などの問題に依存しない遺伝的操作を高速に実行する専用ハードウェアと，問題に依存する評価値計算の部分を並列に実行する並列プロセッサからなる．このシステム構成により任意のGPの高速実行が可能となる．

以降では，2.で遺伝的プログラミングとその高速化に対する従来研究，3.で提案コンピュータシステムの構成，4.で提案コンピュータシステムの性能評価，5.でまとめと今後の課題について述べる．

2. 準　　備

2.1 遺伝的プログラミング

遺伝的プログラミングの実行フローを図1に示す．

遺伝的プログラミングを用いて目的とする問題のプログラムを生成する場合，まず，初期設定として解きたい問題のプログラムの構成要素となるファンクションノードとターミナルノードをユーザが決定する必要がある．ファンクションノードの集合は一例として $F=\{+,-,*,/\}$ のように四則演算などの引数を持つノードを設定し，ターミナルノードは $T=\{A,B,C\}$ のように引数を持たないノードを設定する．これらを文法的に誤りのないように組み合わせてプログラムを構成し，これをランダムに複数生成することで初期個体集団 (初期人口) とする．次に，これらの個体 (プログラム) がどれほど目的にあったものかを評価する．これは各個体に対していくつかの入力パターン (fitness case) を与えてやり，その出力からプログラムの評価値を決定する．

続いて，この評価値に基づいて選択，交差の遺伝的操作を実行し新しい個体プログラム (子孫) を生成する．選択では，基本的に評価値の高い個体がより多くの子孫を残す仕組みとなっている．

ここで，交差手法に関して説明する．本研究ではGPの染色体表現としてリニア表現 [3] [8] を用いており，木構造の遺伝子は一次元配列中に格納されている (図2)．つまり，関数テーブル上に定義されているノードに対応する識別番号をpre-orderで木を探索した順に配列に格納する．ここでStackCount(sc)とはノードの関数が持つ引数の値に1を減じたもので，この和を計算することで，配列データから木の構造を知ることができる．すなわち，一次元配列を任意の場所から右へスキャンした場合，scの和が-1になるところが部分木となる．当然，木の始点から終点までの和を計算すると-1となる．

この表現形式に対し，交差としては選ばれた2つの個体の部分木を交換する部分木交換交差と，GAのような1点交差が可能である．部分木交換交差はscの和を計算することによって見つけた部分木どうしを交換することでなされる．1点交差はそれぞれの親について左からscの和を計算し，scの和が等しい交差ポイントから右を交換することで実現される．なお，GAで用いられるような突然変異は，GPにおいてはターミナル節点での部分木交換交差の操作に等しい．1点交差は部分木交換交差に比べて交換する部分が大きく，大域的な探索に向いている．いずれも交差の後で文法的に誤った子孫が生成されないことが保証されている．

このような交差がユーザの指定した一定の確率で個体に対して適用される．交差が行われない場合は再生と呼ばれる親の個体を子の世代にコピーする操作が行われる．以上のようにして新しく生成された個体は評価値が計算され，再び個体集団の中にとりこまれる．そして再びこの選択，交差，評価の一連のプロセスを規定回数繰り返す．この一連のプロセスにより少しずつプログラムの構造が変化し，より適した（賢い）プログラムが残っていき，最終的に目的の（最適な）プログラムが探索できる．

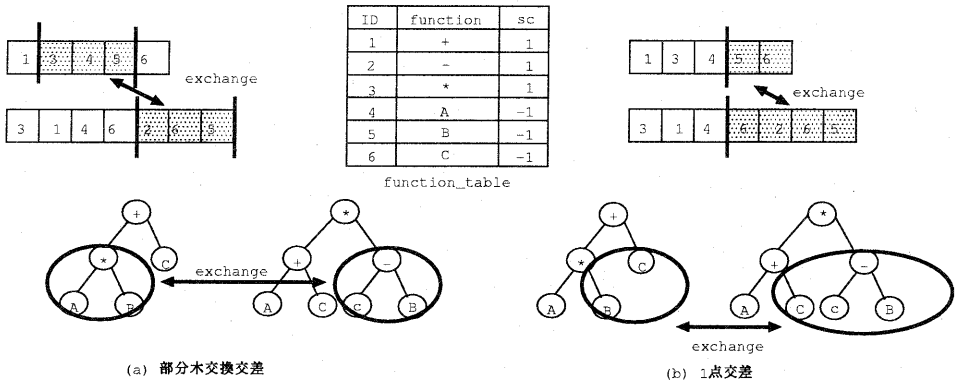
2.2 GPの高速化に関する従来研究

GPの高速化に関する研究には大きく分けて2通りの方法がこれまでに報告されている．これらの研究の内容と問題点を挙げ，本研究の目的を述べる．

1つめは処理の並列化によるもので，これはさらに人口の分割によるものと評価の並列化によるものの2つに分けられる．

人口の分割による高速化にはAndreらの研究 [6] があり，これはGAの並列アイランドモデルに基づいている．すなわち，全体の人口をdemeと呼ばれる部分人口に分割し，各々のプロセッサがその部分人口に対してGPを実行する．また，人口中の多様性を維持するためにプロセッサ間で移住と呼ばれる個体の移動を行う．この手法によりプロセッサ数にほぼ比例した速度向上率が得られる．しかし，この手法は並列度を上げた場合複雑なプロセッサのネットワーク構造が必要である．また，並列度を上げた場合プロセッサ数と同数のdeme数に人口を分割する必要があるが，人口集中型のモデルと比較したこのアイランドモデルの有効性は十分には解明されていない．

評価の並列化による高速化にはOussaidèneらの研究 [7] があり，これはGP全体の実行時間の大部分を占める評価値計算の部分を並列に行うことで全体のパフォーマンスを向上させようというものである．この手法においてもプロセッサの並列度に



ID	function	sc
1	+	1
2	-	1
3	*	1
4	A	-1
5	B	-1
6	C	-1

図 2　部分木交換交差, 一点交差

ほぼ比例した速度向上率が得られる．しかし，並列度はせいぜい 8 にとどまっており，並列度を大きくした場合についての結果は報告されていない．

2 つめは専用ハードウェア化によるもので従来研究には Handel-C というハードウェア記述言語を用いて GP 全体を専用ハードウェアとして FPGA 上に実現した Martin [4] らの研究がある．このハードウェア GP は簡単なベンチマーク問題に対して，ソフトウェアの約 100 倍以上もの高速実行が達成されている．しかし，GP の評価値計算の部分をハードウェアで実現する場合いくつかの問題点がある．まず，問題が変わる度に設計し直さなければならないため，任意の問題を解く場合には，その度に多大な設計時間を要するという欠点がある．また，一般にハードウェア記述言語の記述レベルは C 言語のそれよりも低いため現実的な問題の評価関数をハードウェア化するのは非常に難しい．さらに，問題によっては評価値計算に回路シミュレータ Spice などのアプリケーションを用いることがあり，これらの計算は汎用プロセッサで行うのが現実的である．

本稿で提案するシステムでは GP における評価の部分は並列プロセッサで処理し，評価以外の部分の処理に対しては専用ハードウェアを設けることで，任意の問題に対する処理の高速化を図った．ネットワーク構成は基本的にはアイランドモデルのように人口を分割することはせず，人口集中型である従来の GP のアプローチを採用し，さらに並列度を上げたい場合は 1 ボードあたりのプロセッサ数を増やす，もしくは本システムを複数相互接続することで高並列システムへと拡張が可能な構成にしている．

3. 提案システムの構成

3.1 システム概要

提案コンピュータシステムの構成を図 3に示す．提案コンピュータシステムは，GP ハードウェアと個体メモリ，評価値メモリを搭載したメインボードと個体評価用のプロセッサを搭載した複数のボードをバス結合した構成となっている．通常バス結合においては接続できるプロセッサ数の上限は 20 程度までであるが，さらに並列度を上げたい場合は 1 ボード上に搭載するプロセッサ数を増やすか，もしくは本システムを複数用意し，GP ハードウェア同士を接続することにより容易に拡張が可能である．システムのそれぞれの構成要素について以下で説明する．

(1)GP ハードウェア

提案システムにおける中心的モジュールであり，GP における選択，交差などの遺伝的操作を人口中の全個体に対して実行する．このハードウェアは幾つかのサブモジュールから構成されており，詳細は次節以降で詳しく述べる．

(2) 個体メモリと評価値メモリ

それぞれ人口中の個体とその評価値を格納する．個体とその評価値は GP 専用ハードウェアと評価用のプロセッサの双方からのアクセスがあり，パフォーマンスのボトルネックとなる可能性があるので別々のメモリを設けている．個体メモリに関しては 1 ワード 32 ビットとし，512Mbyte のメモリを接続する．これにより 128 ワードの個体プログラムを最大 2^{20} 個格納することが可能である．また，プロセッサとの通信のためのメモリ空間を設けており，ここにはプロセッサが読み出して評価すべき個体の番号の一覧，次にプロセッサがとりだす個体番号が格納されている位置，次にハードウェアが個体番号を書き込む位置などの情報が記憶されている．評価値メモリに関しては 4Mbyte のメモリを接続し 32 ビットの評価値を個体数と同数格納する．

(3) 個体評価用並列プロセッサ

GP における評価値計算を並列に実行する．送られてきた個体 (個体のプログラム) について，問題毎に設定された入力パターンに対しプログラムを実行する．そしてその出力値が所望とする値に近いかどうかによって評価値を決定する．この部分は既存の汎用プロセッサを使用するものと仮定している．

次に，個体や評価値のデータ転送について簡単に説明する．GP 専用ハードウェアでの遺伝的操作で生成した個体は個体メモリに格納される．これと同時に GP 専用ハードウェアは各プロセッサに対して個体評価のリクエスト信号を出す．これを受け取ったプロセッサはアクセス要求を出し，バスの使用権を得た後に，バスを通して個体メモリから個体を読みだす．その後評価を行い，その評価値をバスを通して評価値メモリに書き込む．バスは PCI バスを用いるものと仮定し，これらの転送は

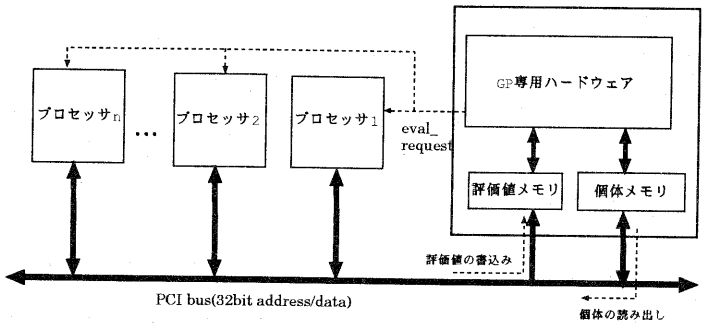


図3　提案システム構成

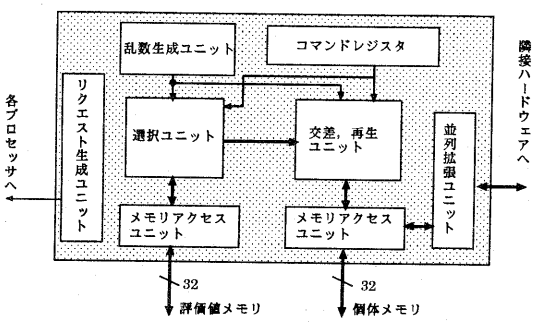


図4　専用ハードウェアの構成

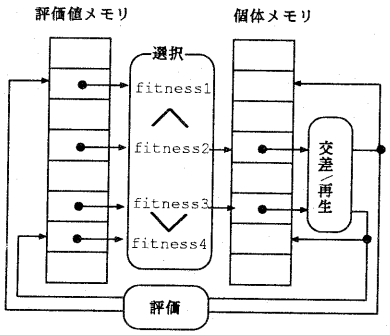


図5　ハードウェアの実行フロー

表1　GP ハードウェアの仕様

個体数	任意 (最大 2^{20})
個体の最大長	任意 (最大 508 ノード)
関数テーブル	ノードの ID 番号とそれに対応する引数の数を設定.
	1 ノードは 8 ビット (256 種類) とする.
終了基準	設定した評価回数に達した時点で終了 (最大 2^{30})
選択手法	トーナメント選択
交差手法	部分木の交換による交差，1 点交差
交差確率	2 つの交差をそれぞれについて
	0/255〜255/255 まで設定可能.

PCI のプロトコルで行うように設計した．アービタは集中型アービタのシミュレーションモデルを使用した．

3.2　GP 専用ハードウェアの概要

GP ハードウェアのブロック図を図 4に示す．モジュールには GP における遺伝的操作を実行する選択ユニット，交差/再生ユニットがある．また，評価値メモリや個体メモリへのアクセスはプロセッサからのアクセスと重複する可能性があるため，これを制御するメモリアクセスユニットを設けている．初期個体生成はソフトウェアで行い，予め個体メモリに個体が格納されているものとする．リクエスト生成ユニットはプロセッサに対して，個体評価のリクエスト信号を送信する．コマンドレジスタには各種実行パラメータを設定する．ハードウェア仕様を表 1に示す．

3.3　ハードウェア内のモジュールとその機能

GP ハードウェア内の各モジュールについて説明する．併せて GP ハードウェアの実行フローの概略を図 5に示す．

3.3.1　選択ユニット

GP の遺伝的操作である選択を実行する．提案するハードウェア GP は Steady-State-GP(世代の概念を持たない GP) として実現されるため，選択手法はトーナメント選択を行う．トーナメント選択とは複数の個体 (通常 2 個) の中から最も評価値の高い個体のみを次世代に残すという選択手法である．ハードウェアの処理を以下で説明する．まず乱数生成ユニットより乱数を取得し，ランダムに 2 つの個体を決定する．次にその個体に対応する評価値を評価値メモリからリードする．ただし，その個体に対応する評価値がロックされていた場合は，個体を選び直す．そして選んだ 2 つの評価値を比較し，勝敗を決定する (勝者 1，敗者 1 の決定)．勝者 2，敗者 2 に関しても同様の操作を行う．勝者 1，2 に関しては交差/再生を行うため，その個体番号が交差/再生ユニットに送られる．敗者に関しては，敗者の個体は勝者が交差/再生を行った時に生成される新しい個体に置き換えられるため，その新しい個体がプロセッサで評価されその評価値が評価値メモリに書き込まれるまでの間，対応する評価値メモリの番地にロック変数が書き込まれる．

3.3.2　交差/再生ユニット

GP の遺伝的操作である交差および再生を行う．交差手法としてハードウェアにおいては 2 節で述べた部分木交換交差と 1 点交差の両方をサポートしている．

ハードウェアの処理について説明する．まず，乱数生成ユニットから得られた乱数とユーザが設定した各交差確率を比較し，

実行する交差を決定する．そして，選択ユニットから送られてきた勝者の個体番号をアドレスとして個体メモリにアクセスし，交差対象となる個体の長さをロードする．次にその個体の長さを基に2つの親の交差ポイントをランダムに選択する．そして，部分木交換交差の場合はその交差ポイントからスキャンを開始し交換する部分木を決定する．一点交差の場合はそれぞれ交差ポイントまでのStackCountを計算して和が等しければ交差ポイントとして採用し，そうでなければ再び交差ポイントを選び直す．GAとは異なりGPにおいては交差前と交差後では一般に個体の長さが変化するが，効率的な探索のためには個体の長さがあまりに大きくなってしまうのは一般に望ましくない．よって個体長の制約を設け，選んだ交差ポイントで交差した結果の個体長が制約を越える場合は交差ポイントを選び直す．以上の条件を満たすと交差が実行され，交差の結果生成された個体は選択における敗者に上書きされる．交差が行われない場合は再生を行う．すなわち個体メモリから勝者の個体を読み出し，そのまま敗者のアドレスに格納する．

最後に生成された個体の番号をプロセッサに渡すために敗者の番号を個体メモリ中の指定の空間に書き込む．

3.3.3 乱数生成ユニット

GPは乱数により遺伝的操作を実行する．GPハードウェアではcellular automaton based algorithmを利用して32ビット乱数を発生させている．確率的探索を行う場合，しばしば乱数の質と探索効率との関係が論じられるが，Martin [5] によると基本的にGPにおいては一般的な乱数を使用している限りは，乱数の質が探索効率に影響する度合はかなり小さいことが報告されている．よって以下に述べる乱数発生の方法で十分である．乱数発生は次に示す2つの規則に従って状態遷移を行う32個のセルからなる．

Rule 90: $s_i^+ = s_{i-1} \oplus s_{i+1}$

Rule 150: $s_i^+ = s_{i-1} \oplus s_i \oplus s_{i+1}$

今 s_i がセル i の現在の状態であるとすると，s_i^+ は s_i の次の状態を示す．$\oplus$ は排他的論理和である．規則90においては両隣の値によってのみ現在の値が更新されるのに対して，規則150においては自身の状態も考慮に入れた上で，更新を行う．初期状態はユーザが指定できる．このような規則列を用いて，乱数生成ユニットは周期の長い乱数を発生させている．

3.3.4 メモリアクセスユニット

メモリアクセスユニットは評価値メモリ，個体メモリへのアクセスを制御する．ハードウェアが選択を実行している時はプロセッサからの評価値メモリへのアクセスが制限され，交差を実行している時はプロセッサからの個体メモリへのアクセスが制限されるためこのユニットが必要となる．メモリアクセスユニットは選択ユニットおよび交差/再生ユニットからメモリアクセス要求信号を受信し，プロセッサからのメモリアクセスが実行中でなければ受諾信号を返す．その間，選択ユニットもしくは交差/再生ユニットは待機する．

3.3.5 コマンドレジスタ

GPの実行における各種パラメータを設定する．個体数，ノードの種類，評価回数については，現実的な問題を解くことを想定し十分大きな値が設定可能である．個体長は4ノードが通常のC言語のプログラム1行に相当すると換算し，120行程度までのプログラムを扱えるようにした．

3.3.6 リクエスト生成ユニット

個体評価のリクエストをプロセッサに対して送信する．モジュール内にカウンタを設け，交差および再生によって新しい個体が生成される毎にインクリメントする．カウンタの値が1以上であればリクエスト信号を出力し，それを受信したプロセッサはバスアービタに要求信号を出力する．バスの所有権を得て，個体をロードしたプロセッサはカウンタをデクリメントする．これによりプロセッサは必要な時のみハードウェアにアクセスして個体を受け取ることが可能となる．

3.3.7 並列拡張ユニット

提案システムにおいては，GPハードウェア同士を接続してより高いプロセッサの並列度でのGPが実行可能である．GPハードウェアを接続してアイランドモデルを構成する場合は人口中の多様性を維持するために，ある一定の割合で移住と呼ばれる個体の移動が行われる必要がある．このユニットは隣接ハードウェアとの通信を行い，個体交換を行うことで並列GPを可能としている．

3.4 ハードウェア設計

回路設計はVerilog-HDLを用いて行った．記述量はGP専用ハードウェア本体が約3000行，システム全体も含めると約4000行である．シミュレーションはCadence社のVerilog-XLシミュレータを用い，論理合成はSynopsys社のDesign Compilerを用いて行った．使用したライブラリはローム0.35μmテクノロジ東大版EXDライブラリを使用した．現在のところ論理合成結果は，セル数が約13000，ネット数が約14000，ゲート数は2入力NANDゲート換算で約35000となっている．

4. 性能評価

4.1 ハードウェアのパフォーマンス

まず，ハードウェアGP単体でのパフォーマンスを評価した．シミュレーションはGPのベンチマーク問題である3bit-parity関数生成問題と人工蟻の探索問題 [2] に対して実行し，ハードウェアGPとC言語で記述したソフトウェアGPの比較を行った．ハードウェアGPはVerilog-XLによるRTLレベルシミュレーションを行い，ソフトウェアGPはSun UltraSparcII 450MHzのマシン上で実行した．

ここで各問題の定義を示す．3bit-parity関数生成問題とはブール関数合成の一種で3入力のうち1の数が奇数個の時に1，偶数個の時に0を出力するような関数を，以下のノード集合と評価値を基に生成させるような問題である．ファンクションノードは $F = \{AND, OR, NAND, NOR\}$ で全て2入力ブール関数である．ターミナルノードは $T = \{A, B, C\}$ でブール関数への入力値である．評価値はすべての入力パターン (2^3) のうち正しい出力を出した数としている．

人工蟻の探索問題 (ANT 問題) とは人工生命の簡単な練習問題で 32 × 32 マスのフィールドにおかれた 89 個の餌を制限エネルギー内でできるだけ多く発見できるような蟻の行動プログラムを見つけるという問題である．評価値は蟻が見つけた餌の数となる．ノードの種類を表 2に示す． 各実験はそれぞれ初期

表 2 ANT 問題のノードの種類

関数名	引数の数	意味
If_Food_Ahead	2	目前に餌があるとき第 1 引数，ないとき第 2 引数を実行する．
PROG2	2	第 1，2 引数を順に実行．
PROG3	3	第 1，2，3 引数を順に実行．
RIGHT	0	右に 90 度向きを変える．
LIGHT	0	左に 90 度向きを変える．
FORWARD	0	一歩前に進む．

個体をランダムに生成して 20 回実験を行い，選択，交差などの遺伝的操作に要する平均のクロック数を計測した．双方の実験で使用した共通パラメータを図 3に示す．

表 3 使用パラメータ

個体数	512
終了評価回数	2^{14}
初期個体長	7
最大個体長	64
遺伝操作の確率	部分木交換交差 0.8
	1 点交差 0.1
	再生 0.1

シミュレーション実験の結果，ソフトウェア GP が 1 つの個体に対し選択と交差を施すのに要する時間は平均 5181 クロックであった．これに対し，ハードウェア GP は 1 つの個体あたり平均 375 クロックである．よってソフトウェア実行の CPU と GP 専用ハードウェアのクロック周波数を同一と仮定すれば，GP ハードウェアは選択や交差をソフトウェアで実行する場合と比べて約 14 倍高速に実行可能である．

4.2 システム全体のパフォーマンス

次にプロセッサを並列に接続し並列システムを構築した場合のパフォーマンスをシミュレーションにより求めた．対象とする問題，パラメータは 4.1 節と同様である．プロセッサ数は 1 および 4 とし，各プロセッサで実行する評価値計算は C 言語で記述し，Verilog-HDL ソースとのインターフェースを PLI1.0(Program Language Interface) を用いて記述し，PCI バスについても Verilog-HDL で記述し，システム全体のシミュレーションを Cadence 社の Verilog-XL シミュレーション上で行った．結果を表 4，5に示す．

表 4，5ではソフトウェアで実現した GP と，評価の並列度を 1 (GPHW+1Pro) および 4 (GPHW+4Pro) とした場合の提案システムで GP を実行した場合で得られた CPU 時間，速度向上比について比較した．提案システムの GP 専用ハードウェアはクロック周波数 100MHz を仮定し，並列プロセッサはクロック周波数 1GHz を仮定している．実験の都合上，ソフトウェア

表 4 実験結果 (3bit-parity)

	実行時間 (sec)	速度向上比
ソフトウェア GP	1.0625	1.00
GPHW + 1Pro	1.0169	1.04
GPHW + 4Pro	0.2564	4.14

表 5 実験結果 (ANT 問題)

	実行時間 (sec)	速度向上比
ソフトウェア GP	4.2003	1.00
GPHW + 1Pro	4.2261	0.99
GPHW + 4Pro	1.0595	3.96

GA は UltraSPARCII450MHz プロセッサ上で実行し，実行時間をクロック周波数 1GHz の場合に換算した．表からわかるようにシステムはプロセッサが 1 個の場合においてソフトウェアとほぼ同等の速度パフォーマンスを示しており，提案システムにおいて，バスはほとんどパフォーマンスのボトルネックとはなっていないことがわかる．また，プロセッサが 4 個の場合においては計算時間が約 1/4 に短縮されており，プロセッサ数に比例した速度向上比が得られていることがわかる．

5. あとがき

本研究では任意の GP を高速に実行可能な新しいコンピュータシステムを提案し，シミュレーションによりその有効性を示した．今後の課題としては，より現実的な問題への適用，さらに並列度を上げた時の本システムの有効性の検証などが挙げられる．

謝辞：ハードウェア設計とシミュレーション実験に御協力頂いた本学学部生の竹村和浩君と濱本泰治君に感謝します．

文 献

[1] D. E. Goldberg,"Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wesley Publishing Company (1989).

[2] John R.Koza, "Genetic Programming," MIT Press(1992).

[3] Mike J.Keith and Martin C.Martin, "Genetic Programming in C++:Implementation Issues," Advances in Genetic Programming, pp 285-300 (1994).

[4] Peter Martin, "A pipelined hardware implementation of genetic programming using FPGAs and Handel-C,"Technical Report, Department of Computer Science, University of Essex (2002).

[5] Peter Martin, "An analysis of random number generators for a hardware implementation of genetic programming,"Technical Report, Department of Computer Science, University of Essex (2002).

[6] David Andre and John R. Koza,"A parallel implementation of genetic programming that achieves super-linear performance, "Proc. of the International Conference on Parallel and Distributed Processing Techniques and Applications, Vol III, pp 1163-1174 (1996).

[7] Mouloud Oussaidene, et al., "Parallel genetic programming and its application to trading model induction " Parallel Computing,vol 23, no.8, pp 1183-1198 (1997).

[8] Nao Tokui and Hitoshi Iba, "Empirical and statistical analysis of genetic programming with linear genome "Proc.IEEE International Conference on Systems,Man and Cybernetics. (1999).