5E - 8　遺伝的アルゴリズムの高速実行に適した命令セットを持つRISCプロセッサの開発†

小泉 慎哉*　　若林 真一*　　小出 哲士*,**

*広島大学工学部　　**東京大学大規模集積システム設計教育研究センター

1　まえがき

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遺伝的アルゴリズム (Genetic Algorithm, GA) が知られている [1]. GA は優れた解探索能力を持つ一方で，複雑な問題に対しては多大の計算時間を要するという問題点を持つ．この問題点の改善策として，著者らは，著者らが提案した個体の優劣の度合いを表す指標であるエリート度に基づく複数の交差手法と突然変異確率の適応的選択手法を組み込んだ適応的遺伝的アルゴリズム (EAGA) のハードウェア化を研究し，実際に Genetic Algorithm Accelerator - I (GAA-I)，および，GAA-II として LSI チップ化している [4, 3]．しかし，これらのハードウェアは選択・交差・突然変異などの遺伝オペレータをハードウェアで実現しているため適用可能な問題が限られていた．

そこで本稿では，任意の GA を高速に実行可能な GA 専用の RISC プロセッサを提案し，実際に LSI チップとして実現する．本研究で開発する専用プロセッサの命令セットは DLX アーキテクチャ[2] に準拠するが，GA の実行においては通常は必要ない浮動小数点命令を除き，代わりに GA 特有のビット演算命令等を追加する．問題に依存する個体評価回路は外部に付加した FPGA で実現し，個体の評価と交差・突然変異の並列処理を可能にする．このようなハードウェア構成により汎用性を失うことなく GA の高速実行が可能になる．

2　提案プロセッサ

2.1　提案プロセッサのシステム構成

提案プロセッサのシステム構成を図 1に示す．システムは大きく分けて以下の三つから構成されている．

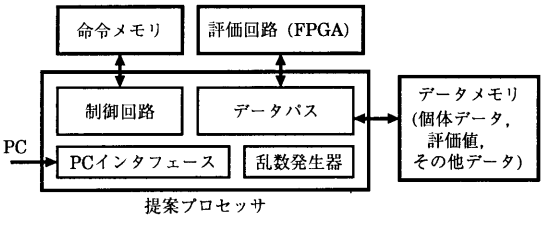

図 1　システム構成

(1) プロセッサ本体

システムのメインモジュールであり，ホストコンピュータからの実行開始信号を受諾すると動作を開始し，命令メモリから読み出した命令に従って交差・突然変異・選択など，個体の適応度の評価値計算を除いた GA の各種の操作を行う．詳細は次節で述べる．

(2) 命令メモリとデータメモリ

提案プロセッサはハーバードアーキテクチャを採用しており，命令バスとデータバスのそれぞれに 1 ワードが 32 ビットの SRAM が接続されている．命令メモリには所望の GA を実現した機械語プログラムを格納し，データメモリには染色体情報，適応度評価値，その他のデータを格納する．

(3) 適応度評価回路 (オプション)

外部評価回路では，プロセッサで遺伝的操作を施した個体の評価値計算を行う．評価値計算を高速に実行するため FPGA により評価値計算を実現し，プロセッサ本体で実行される交差・突然変異との間で並列処理を実現する．なお，提案プロセッサ自身が適応度の評価値計算を行う構成にすることも可能である．

2.2　プロセッサ・アーキテクチャ

提案プロセッサはロード/ストア・アーキテクチャである．32 ビットの汎用レジスタを 32 個持ち，レジスタ R0 の値は常に 0 とする．アドレッシングモードはただ 1 つで，実効アドレスはベースレジスタに符号付きの 16 ビットディスプレースメントを加算して得られる．データメモリはバイトアドレス可能で，すべてのメモリ参照はメモリと汎用レジスタとの間でのロードとストアでなされる．すべての命令は 32 ビット長である．

すべての命令の実行は，命令フェッチ IF，命令デコード ID，実行 EX，メモリアクセス MEM，レジスタ書き込み WB の 5 つの基本ステージから成り，これらを 5 段のパイプラインで処理する．

2.3　命令セット・アーキテクチャ

命令セットは DLX[2] の命令セットに準拠するが，GA の実行においては必要ない浮動小数点演算命令を除き，GA の実行において多用される GA 特有のビット演算を新たに命令として加える．さらに，2 つのレジスタに格納されたデータに対して同じに演算を施す SIMD 型の命令や，所望の乱数を取得する命令なども追加する．それらの命令の一部を表 1に示す．

表 1　GA に特化した命令

型	ニーモニック	内容
R'	extract_bits	レジスタ中の指定ビットを取り出す
R'	insert_bits	レジスタ中の指定ビットにデータを入れる
R'	comp_bits	レジスタ間の指定ビットを比較する
R'	move_bits	レジスタ間で指定ビットを移動する
R'	and_bits	レジスタ間で指定ビットの AND をとる
R'	or_bits	レジスタ間で指定ビットの OR をとる
R'	not_bits	レジスタ間で指定ビットの NOT をとる
R'	add_bits	レジスタ中の指定ビットに加算する
R'	sub_bits	レジスタ中の指定ビットから減算する
I	reset_rand	乱数ジェネレータに初期シードを渡す
I	set_rand_num	0 から指定した数までの乱数を発生する
I	set_rand_bit	指定ビット幅の乱数を発生する
I	set_rand_bits	m ビットに指定確率で 1 をセットする

†"An Implementation of a RISC Processor with Instruction Set Suitable for High-Speed Execution of a Genetic Algorithm", Shinya KOIZUMI*, Shin'ichi WAKABAYASHI*, Tetsushi KOIDE*,**, *Faculty of Engineering, Hiroshima University, **VLSI Design and Education Center, The University of Tokyo. e-mail:koi@ecs.hiroshima-u.ac.jp

```
L2_LF0:
    lw    r1,-32(r30)
    lw    r2,-36(r30)
    slt   r1,r1,r2
    bnez  r1,L5_LF0
    j     L3_LF0
L5_LF0:
    lw    r1,-32(r30)
    snei  r2,r1,#0
    beqz  r2,L6_LF0
    lw    r1,-20(r30)
    slli  r2,r1,#0x4
    sw    -20(r30),r2
L6_LF0:
    lw    r1,-20(r30)
    addi  r2,r1,#15
    sw    -20(r30),r2
```

```
L4_LF0:
    lw    r2,-32(r30)
    addi  r1,r2,#1
    add   r2,r0,r1
    sw    -32(r30),r2
    j     L2_LF0
L3_LF0:
    nop
    lw    r1,-36(r30)
    sw    -32(r30),r1
L7_LF0:
    lw    r1,-32(r30)
    lw    r2,-40(r30)
    slt   r1,r1,r2
    bnez  r1,L10_LF0
    j     L8_LF0
```

```
L10_LF0:
    lw    r1,-20(r30)
    slli  r2,r1,#0x1
    sw    -20(r30),r2
L9_LF0:
    lw    r2,-32(r30)
    addi  r1,r2,#1
    add   r2,r0,r1
    sw    -32(r30),r2
    j     L7_LF0
L8_LF0:
    nop
    lw    r1,-40(r30)
    sw    -32(r30),r1
```

```
L11_LF0:
    lw    r1,-32(r30)
    lw    r2,-44(r30)
    slt   r1,r1,r2
    bnez  r1,L14_LF0
    j     L12_LF0
L14_LF0:
    lw    r1,-20(r30)
    slli  r2,r1,#0x4
    sw    -20(r30),r2
    lw    r1,-20(r30)
    addi  r2,r1,#15
    sw    -20(r30),r2
L13_LF0:
    lw    r2,-32(r30)
    addi  r1,r2,#1
    add   r2,r0,r1
    sw    -32(r30),r2
    j     L11_LF0
```

```
L12_LF0:
    lw    r1,-12(r30)
    lw    r2,-20(r30)
    and   r1,r1,r2
    lw    r3,-20(r30)
    sub   r2,r0,r3
    subi  r2,r2,#1
    lw    r3,-16(r30)
    and   r2,r2,r3
    or    r1,r1,r2
    sw    -24(r30),r1
    lw    r1,-16(r30)
    lw    r2,-20(r30)
    and   r1,r1,r2
    lw    r3,-20(r30)
    sub   r2,r0,r3
    subi  r2,r2,#1
    lw    r3,-12(r30)
    and   r2,r2,r3
    or    r1,r1,r2
    sw    -28(r30),r1
```

(a)DLXアーキテクチャにおける2点交差のアセンブリコード

```
L12_LF0:
    lw    r3,-16(r30)
    lw    r4,-20(r30)
    sub   r4,r4,r3
    lw    r5,-24(r30)
    multi r3,r3,r5
    multi r4,r4,r5
    lw    r1,-8(r30)
    lw    r2,-12(r30)
    move_bits r1,r2,r3,r3,r4
    sw    -8(r30),r1
    sw    -12(r30),r2
```

(b)提案プロセッサにおける2点交差のアセンブリコード

図 3 DLX 命令セットによる 2 点交差アセンブリコードと提案プロセッサにおける 2 点交差アセンブリコード

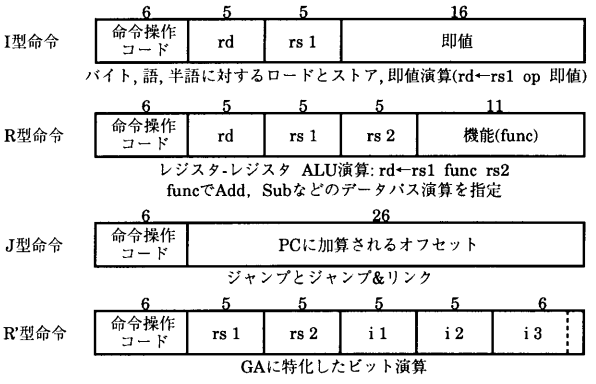

図 2 命令フォーマット

新たに加えるビット演算命令は二つのレジスタをソースレジスタとして指定し，さらにレジスタ中のどのビットに対して演算を施すかを指定する．そこで，DLX アーキテクチャの命令形式に新しい形式として R' 型の命令形式を加える．R' 型の命令では rs1 と rs2 で演算対象の二つのレジスタを選択し，i1 と i2 でレジスタ中のビット位置を指し，i3 でビット幅を選択する．図 2に提案プロセッサの命令フォーマットを示す．すべての命令は 32 ビット長で，6 ビットの命令コードを有している．図 2 において，i3 の右端の 1 ビットは comp_bits 命令において結果を格納するデスティネーションレジスタを指定するために使用される．

3 提案プロセッサにおける GA パフォーマンスの見積もり

元の DLX の命令のみを使用した場合と，今回，DLX プロセッサに加えた命令を使用した場合とで，GA の実行速度にどれほどの差が出るかを，GA の代表的な交差手法である 2 点交差で見積もった．以下では，2 ワード (64 ビット) 中に 0 から 15 の数値 (4 ビット) の順列をコーディングしたデータに対する 2 点交差と，この順序表現のデータに対するスワップ突然変異を考える．

図 3(a) に DLX コンパイラにより生成された 2 点交差のアセンブリコードを示す．DLX の命令セットにはビット演算命令が無いため，交差を行うにはマスクを作成する必要がある．図 3(a) において，左 4 列はマスク作成を行っているアセンブリコードである．そして，右端の 1 列でそのマスクを用いて 2 点交差を実現している．1 命令を 1 クロックで実行すると仮定すると，このコードを実行するためには 138 クロックが必要である．

次に，提案プロセッサの命令セットを用いて実現した 2 点交差のアセンブリコードを図 3(b) に示す．提案プロセッサに追加した命令を有効に利用することで，かなり少ない命令数で 2 点交差が実現できる．1 命令を 1 クロックで実行できるとすれば，このコードの実行に必要なクロック数は 11 クロックである．

表 2に 2 点交差とスワップ突然変異に対する比較結果を示す．この結果から提案プロセッサは GA 実行に対し，パフォーマンスの多大の向上が期待できる．

表 2 クロック数の比較

	2 点交差	スワップ突然変異
DLX プロセッサ	138	75
提案プロセッサ	11	7

4 あとがき

本稿では，任意の GA を高速に実行可能な GA 専用の RISC プロセッサを提案した．また，GA の実行時間の大部分を占める交差を実際にアセンブリコードで記述し，どのくらいの速度増加が見込まれるかを見積もることにより，命令セットの有効性も示した．今後，実際にプロセッサを設計・製作し，評価・検証を行なっていく予定である．本研究の一部は平成 12 年度科学研究費補助金 (C) (2) (課題番号 12838008) による．

文献

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

[2] David A. Patterson and John L. Hennessy : "Computer Architecture: A Quantitative Approach," Morgan Kaufmann Publishers, Inc. (1990).

[3] S. Wakabayashi, et al : "Genetic Algorithm Accelerator GAA-II," Proc. Asia and South Pacific Design Automation Conference, pp. 9-10 (2000).

[4] 若林, 小出, 八田, 中山, 後藤, 利根 : "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," 情報処理学会論文誌 Vol.41, No.6, pp. 1766-1776 (2000).