フラッシュメモリを内蔵したマイクロプロセッサを基本とした進化型ハードウェアの提案

佐 藤 裕 二†

本稿では，マイクロプロセッサを基本とした，新しい進化型ハードウェアのアイデアを提案する．近年，PLD や FPGA を用いた進化型ハードウェアの研究がさかんに行われている．一方，学習に時間がかかる，いつ有効な機能が出現するか予想が難しい，チップサイズが大きい，などの問題から工学分野への普及は今一歩捗っていない．ここでは，これらの問題点を対策する 1 つの手段として，(1) マイクロプロセッサにフラッシュメモリを内蔵することで，オンボードでのプログラムの書き換えを可能にする，(2) 進化的アルゴリズムを用いた，レジスタ・トランスファ・レベルの学習機能を持たせる，(3) 定型的な処理を行うプログラムと学習により自己組織化するプログラムの共存が簡単に行える構成をとる，ことを提案する．

Proposal for a Field-evolvable Hardware Based on a Microprocessor Incorporated with Flash Memory

Yuji Sato†

A new idea for evolvable hardware based on a microprocessor is proposed. In recent years, there has been much research using Programmable Logic Devices (PLD) and Field Programmable Gate Arrays (FPGA). In particular, the application of digital circuit evolution to engineering fields has already begun. On the other hand, long learning time, difficulty to predict when an effective capability will appear, large chip size and other such problems have hindered progress in diffusion into engineering fields. Here, we propose register transfer level evolution performed on a microprocessor as a means of addressing these problems.

1. はじめに

近年，PLD や FPGA を用いた進化型ハードウェアの研究がさかんになりつつある．進化型ハードウェアとは，進化的計算と可変論理構造 LSI の概念を融合させた新しいハードウェア研究の流れである．従来のハードウェアでは一度設計，製作すると製品出荷後の論理変更は困難である．一方，進化型ハードウェアでは，進化型計算を用いた学習機能により，環境に適応した機能を自律的に再構築可能である．進化型ハードウェアは 1992 年に提案[1] されて以来，Intl. Conf. on Evolvable Systems, NASA/DoD Workshop on Evolvable Hardware など進化型ハードウェアを主体とした国際会議が設立され，研究人口は確実に増加傾向にある．Congress on Evolutionary Computation, Genetic and Evolutionary Computation Conference などの進化的計算に関する代表的な国際会議においても，重要なトピックスの 1 つにあげられるようになった．進化型ハードウェアを扱ったジャーナル “Genetic Programming and Evolvable Machines” も発刊された．主な研究機関としては，日本の産総研（旧電総研），ATR，英国サセックス大学，スイス連邦工科大学，米国スタンフォード大学，NASA などがあげられる．進化型ハードウェアは，大きく分けて，デジタル回路の進化とアナログ回路の進化に分類できる．ここで，デジタル回路の進化に限定すると，研究例は，筋電制御型義手への適用[2]，自律移動型ロボットの制御回路への適用[3]，電子写真印刷機用データ圧縮 LSI への適用[4]，FPGA を用いた周波数分別器の進化[5]，デジタルフィルタのゲートレベルでの設計[6]，マルチプレクサ回路の合成[7]，ゲートレベル回路の自動設計[8] など，すでに，工学分野への応用が始まっている．ただし，学習に時間がかかる，いつ有効な機能が出現するか予想が難しい，チップサイズが大きい，などの問題点から工学分野への普及は今一歩捗っていない．一方，我々はすでに，フィールドプログラマブルマイコン[9] を製品として開発している．また，高速学習機能

† 法政大学情報科学部
Faculty of Computer and Information Sciences, Hosei University

を持つニューロコンピュータ[10]を開発した実績を持つ．これらの経験を活かして，本稿では，マイクロプロセッサを基本とした新しい進化型ハードウェア[11]を提案する．以下，デジタル回路の進化を対象として，2章で，従来法と提案する手法とのアプローチ方法の違いを示し，3章で，提案する進化型ハードウェアの実現案を示す．4章で，従来の進化型ハードウェアとの机上での比較検討を行い，最後に，まとめを示す．

2. 従来技術とのアプローチ方法の違い

従来の進化型ハードウェアの基本的なアイデアは，プログラマブルデバイスのコンフィギュレーションビットを遺伝的アルゴリズム（Genetic Algorithms: GA）の遺伝子と見なすことで，目的とする機能に対応したハードウェア構成を自律的に発見することである[12]．

2.1 PLDを基本とした進化型ハードウェアの考え方

図1にPLDの例を示す．ここでは，ANDアレイ，ORアレイの両方が自由にプログラムできる例を示す．また，入力線 x_1，x_2，x_3 と積項線の交点および出力線 f_1，f_2 と積項線の交点が電気的に書き換え可能なEAPLA（Electrically Alterable PLA）の場合を仮定する．図1では，第1の積項線と入力線 $\overline{x_1}$，$\overline{x_3}$，第2の積項線と入力線 x_2，$\overline{x_3}$，第3の積項線と入力線 x_1，x_2 がANDアレイ内で接続している．また，第1の積項線と第2の積項線が出力線 f_1 と，第2の積項線と第3の積項線が出力線 f_2 とORアレイ内で接続している．したがって，出力線 f_1, f_2 の論理はそれぞれ以下のように表せる．

$$f_1 = \overline{x_1}\ \overline{x_3} + x_2\ \overline{x_3} \quad (1)$$

$$f_2 = x_2\ \overline{x_3} + x_1\ x_2 \quad (2)$$

図1において，入力線または出力線が，積項線と接続している部分に“1”，接続していない部分に“0”を割り当てると，ANDアレイ，ORアレイは，2次元配列に対応させることができる．したがって，この2次元配列をGAの遺伝子と見なして進化させることで，進化的計算を用いてAND-ORの2段論理が合成できる．

2.2 FPGAを基本とした進化型ハードウェアの考え方

FPGAの構造は一般に機能ブロックとデータ接続線から構成される．機能ブロックは，機能ブロック内の特定のビット列を書き換えることにより，さまざまな論理機能を実現することができる．データ接続線は，1ビットのスイッチ群を用いることにより，機能ブロックとの接続/非接続を変更できる．すなわち，機能ブロックおよびデータ接続線の状態を決める2進ビット列を変更することにより，FPGAのハードウェア機能を変更できる．このビット列をコンフィギュレーションビットと呼ぶ．FPGAを用いた進化型ハードウェアでは，このコンフィギュレーションビットをGAの遺伝子と見なすことで，目的とする機能に対応したハードウェア構成を自律的に獲得する．この概念図を**図2**[12]に示す．コンフィギュレーションビットに対して遺伝的操作を繰り返すことにより，論理回路が進化する様子を示している．PLDがAND-ORの2段論理構成だったのに対して，FPGAを用いた場合はNAND，NORなどを基本とした多段論理構成となる．

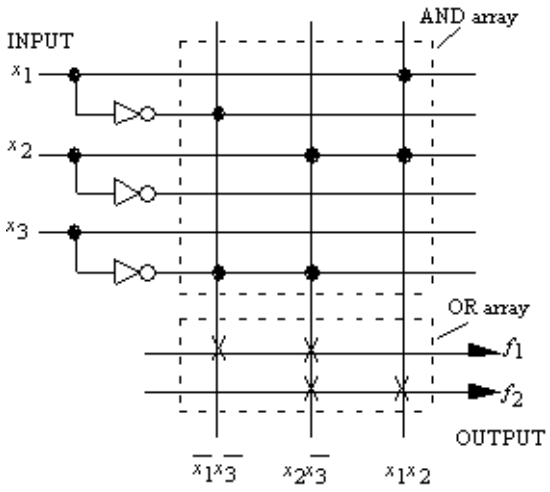

図1 EAPLAの例
Fig. 1 An example of a programmable logic device where an AND array and an OR array are both freely programmable.

2.3 提案する進化型ハードウェアの基本的な考え方

PLD方式，FPGA方式ともに従来の進化型ハードウェアでは，目的とした機能を実現するための論理回路に対応するコンフィギュレーションビットをGAの遺伝子として進化させている．ここで提案する進化型ハードウェアでは，機能を実現するための演算器群の構成は固定である．演算器群の使い方を指示する制御信号の進化を考える．すなわち，レジスタ・トランスファ・レベルの進化をLSI上で行うことを考える．**図3**に演算器群と制御信号の関係を表す概念図を示す．演算器群は，レジスタ・ファイル群，算術論理演算装置（ALU），比較器（Comparator），フラグ制御などの機能モジュールとそれらを結合するデータバスで構成されている．これらの構成要素は，マイクロコードをデコードして得られた制御信号 s_1 から s_n により，制

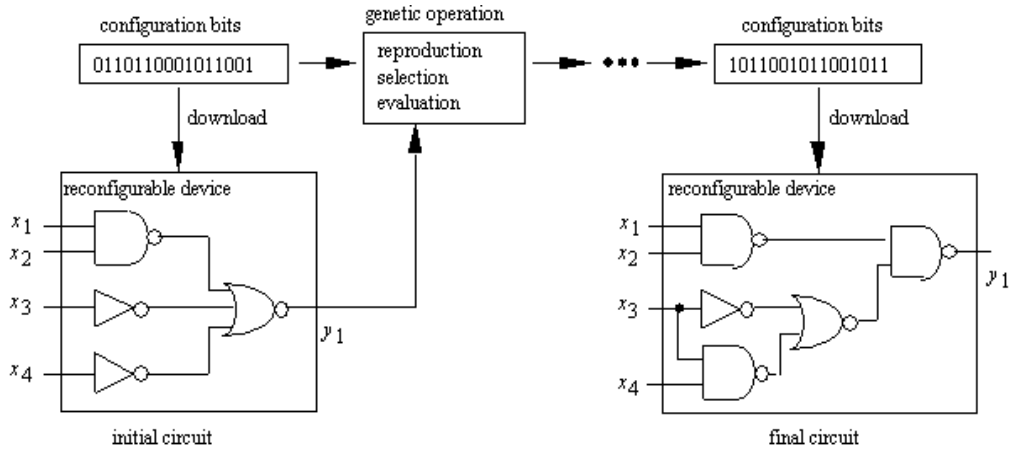

図 2 進化ハードの仕組みを説明するための図
Fig. 2 Basic concept of evolvable hardware based on FPGA (Ref. [Higuchi 1999]).

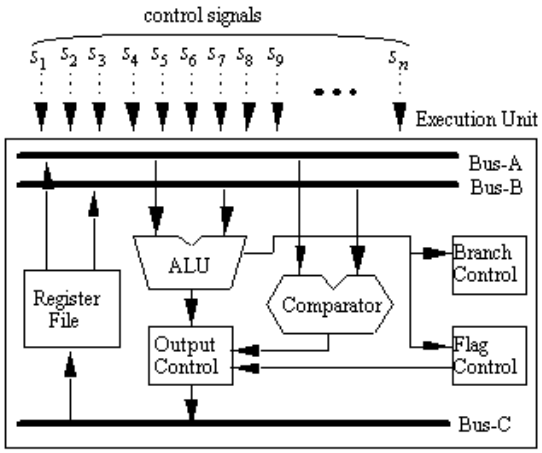

図 3 演算器群と制御信号の関係を説明するための図
Fig. 3 A conceptual diagram that illustrates the relationship of the execution unit and control signals.

御される．たとえば，図において，制御信号 s_1 の値が “1” のとき，アドレス指定されているレジスタ内のデータがバス A に読み出される．同じように，制御信号 s_2 の値が “1” のとき，アドレス指定されているレジスタ内のデータがバス B に読み出される．制御信号 s_4 および s_5 の値が “1” のとき，データバス A およびデータバス B の内容が ALU に入力される．制御信号 $s_6 = 0$，$s_7 = 1$，$s_8 = 1$ は，たとえば ADD 命令を表し，制御信号 s9 の値が “1” のとき，ALU の演算結果がデータバス C に出力される．したがって，制御信号のビット列（s_1, s_2, ..., s_n）を GA の遺伝子と見なせば，目的とした機能を実現するための制御信号を遺伝的操作により求めることが考えられる．

HDL（Hardware Description Language）レベルで遺伝的操作を行い，レジスタ・トランスファ・レベルの回路を合成する研究に関してはすでにいくつかの報告[13)~14)]がされている．しかし，いずれも HDL レベルでの進化であり，仕様変更に関して，再度 LSI を製造し直す必要がある．ここで提案する手法は，LSI 上で直接レジスタ・トランスファ・レベルの進化を扱うものであり，仕様変更に関して再度 LSI を作り変える必要はない．以下，3.1 節で，具体的な実現手段に関して述べる．また，3.2 節で，レジスタ・トランスファ・レベルの進化を補助するための一案を示す．

3. 進化型マイクロプロセッサの実現案

3.1 マイクロコードの進化

進化するマイクロプロセッサを実現する第 1 案は，マイクロコードの進化を考える方式である．マイクロプロセッサでは，一般的に，マイクロプログラム（μP）制御方式を採用している．μP 制御方式では，複雑な制御装置の組織的・系統的設計が可能であり，FPGA を用いたランダムロジックや PLD による設計方式と比較してより複雑な制御装置を容易に実現できる．また，制御記憶装置に冗長性を持たせることにより，制御記憶装置内の内容を書き換えることで，設計変更が容易に実現可能である．μP 制御方式の概念図を**図 4** に示す．制御記憶装置には，AND 命令，ADD 命令などのマイクロ命令に対応した，0/1 のビットコードで符号化されたマイクロコードが格納されている．マイクロコードはデコーダ A〜C によってデコードされ，演算器群を制御するための制御信号に変換される．したがって，マイクロコードを GA の遺伝子と見なすことで，目的とする機能に対応した演算器群の制御を自律的に獲得可能である．従来，制御記憶装置を構成す

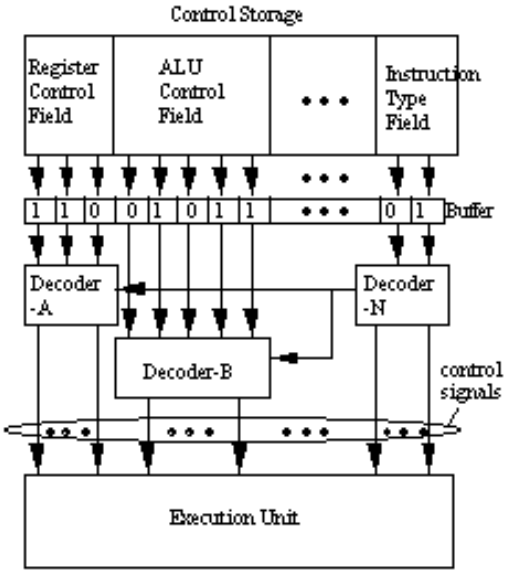

図 4 マイクロプログラム制御方式の概念図
Fig. 4 The concept of the microprogram control method.

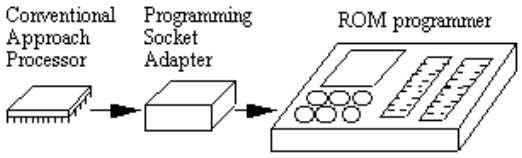

図 5 ROM ライタを用いたプログラムの説明図
Fig. 5 An example of programming with ROM programmer.

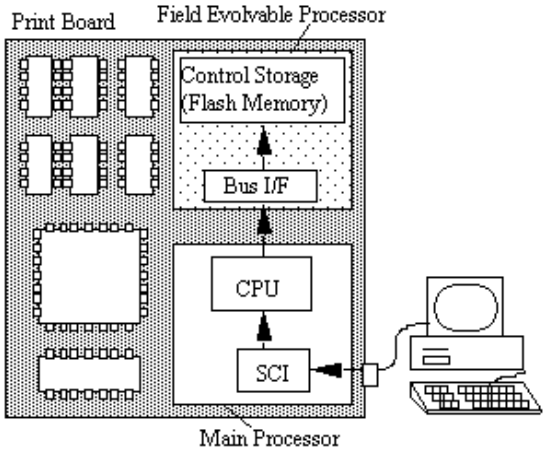

図 6 オンボード・プログラミングシステムの例
Fig. 6 An example of on-board programming systems.

るメモリとして，マスク ROM や EPROM が用いられていた．したがって，動作中に制御記憶装置の内容を書き換えることはできず，プログラムの書き換えに

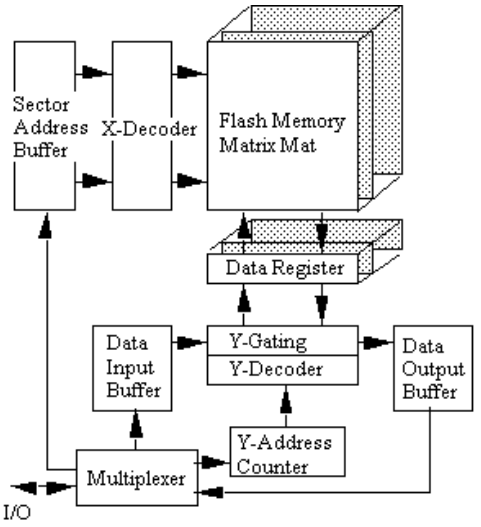

図 7 フラッシュメモリの概略ブロック図の例
Fig. 7 An example of a block diagram of flash memory.

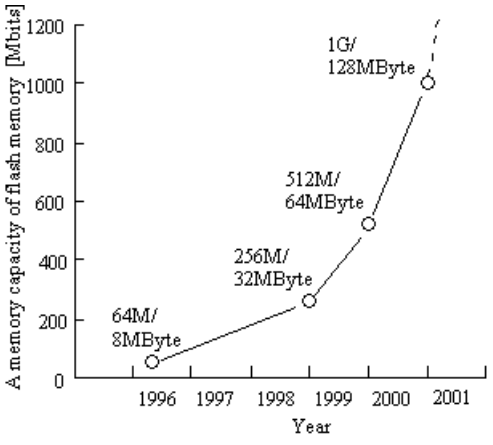

図 8 フラッシュメモリ容量のトレンドマップ
Fig. 8 A trend map of flash memory capacity. The memory capacity has steadily increased in recent years.

は**図 5** に示すような ROM ライタが必要であった．すなわち，マイクロプロセッサを基本とした進化型ハードウェアを考えることは困難であった．一方，近年，フラッシュメモリと呼ばれる，動作中に電気的に書き換え可能な記憶素子が急速に普及し始めている．**図 6** に，フラッシュメモリを用いたオンボード・プログラミングシステムの一例を示す．また，フラッシュメモリのブロック図の一例とメモリ容量のトレンドマップを**図 7** と**図 8** に示す．フラッシュメモリの容量は近年急激に増加しており，現在，制御記憶装置に適用するために十分な容量に達している．したがって，フラッシュメモリを制御記憶装置として用いることにより，

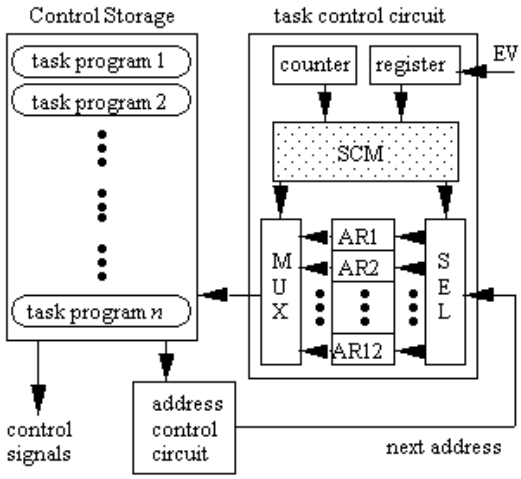

図 9 メモリマットを用いたタスク制御方式の例
Fig. 9 An example of task control method using Schedule Control Memory (SCM). The SCM is a two-dimensional memory map encoding a 0/1 bit code that stores task numbers.

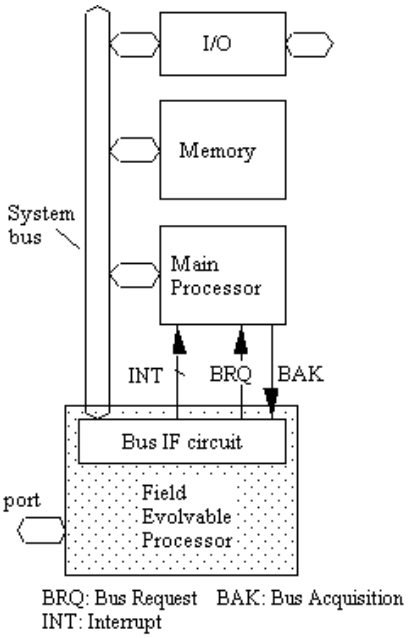

図 10 進化ハードを用いたシステム構成の例
Fig. 10 An example of a system configuration using the field-evolvable processor.

動作中にメモリの内容の書き換えが可能となり，上記マイクロコードを GA の遺伝子と見なした遺伝的操作が可能となる．マイクロコード進化方式の利点の 1 つは，固定的なプログラムと GA などで自己組織化するプログラムの共存が容易に行える点である．図 4 では，制御記憶装置内が，レジスタ制御フィールド，ALU 制御フィールド，命令タイプ制御フィールドなど独立した複数の制御フィールドに分割されている．したがって，制御フィールドごとに，人間が設計するか GA などにより自己組織化するかを，容易に切り分けることができる．

3.2 タスクシーケンサの進化

進化するマイクロプロセッサを実現する補助的な手段として，タスクシーケンサを用いた例を示す．実行する可能性のあるタスクが限られている場合，あるいは，いくつかのタスクの組合せで目的とする機能が実現される場合に有効である．**図 9** にスケジュール・コントロール・メモリ（Schedule Control Memory: SCM）を用いたタスク制御の例を示す．タスクプログラムは，EPROM で構成された 512 ワード × 64 ビットの制御記憶装置内に，最大 12 個格納されている．タスク制御回路は 64 ワード × 6 ビットの SCM と 12 本のアドレスレジスタ（AR）で構成され，カウンタと外部信号（EV）を反映するレジスタで制御される．SCM は 0/1 のビットコードで符号化された 2 次元メモリマットであり，タスク・スケジューリング・プログラムとしてタスク番号が格納される．SCM に格納されたタスク番号は，カウンタとレジスタの値をアドレスとして読み出されて，AR の選択情報となり，実行されるタスクプログラムが決定される．すなわち，選択された AR_i で指定されたタスクプログラム i が実行される．文献 9) では，タスクのスケジューリングを人手で行い，応用基盤組み立て前に，ROM ライタなどを利用して，対応するビットマップを SCM に書き込んでいる．SCM を構成するメモリをフラッシュメモリなどシステム動作中に電気的に書き換え可能な記憶素子に変更することで，SCM を GA の遺伝子と見なした遺伝的操作が可能となる．

同様の原理により，タスクプログラム自体を合成することも可能である．制御記憶装置内にはあらかじめ複数種類のマイクロコードを格納しておき，SCM を用いて，マイクロコードの組合せ，すなわち，タスクプログラムを合成することが考えられる．設計制約が厳しく，マイクロコード自体の進化が困難な場合に有効と考えられる．

3.3 工学的応用の検討

自動車，通信機器，OA 機器などのマイコンを応用した制御システムでは，それぞれの応用に従って，異なる種々のマイコン周辺機能が必要である．周辺機能の例としては，タイマやシリアル通信インタフェースなどがある．従来は，周辺機能を実現するために，ゲー

Register Control Field		ALU Control Field		
Rd	*Rs*	Instruction	Code	Operation
000000	000000	NOP	000*	No operation
		MOV	0010	$Rs(Imm) \rightarrow Rd$
		ADD	0011	$Rd + Rs(Imm) + Cin \rightarrow Rd$
		SUB	0100	$Rd - Rs(Imm) + Cin \rightarrow Rd$
▪	▪	CMP	0101	$Rd - Rs(Imm)$
▪	▪	AND	0110	$Rd \wedge Rs(Imm) \rightarrow Rd$
▪	▪	OR	0111	$Rd \vee Rs(Imm) \rightarrow Rd$
▪	▪	XOR	1000	$Rd \oplus Rs(Imm) \rightarrow Rd$
▪	▪	TST	1001	$Rd \wedge Rs(Imm)$
		SHLL	1010	Shift Left with Carry
		SHLR	1011	Shift Right with Carry
		INCCMP	1100	Increment & Compare
		DECCMP	1101	Decrement & Compare
		INCLR	1110	Increment & Compare & Clear
111111	111111	DECLD	1111	Decrement & Compare & Load

図 11 命令セットとマイクロコードの例
Fig. 11 An example of instruction set and microinstruction code.

トアレイやカスタム LSI を用いていた．しかし，これらの応用分野は技術革新が激しく，応用システムのライフサイクルが短くなる傾向にある．したがって，周辺機能を進化型ハードウェアなどの可変論理構造 LSI で構成することにより，設計変更や仕様変更の問題にフィールドで対応できる．

図 10 に進化型ハードウェアを用いた制御システムの構成例を示す．進化型ハードウェアはシステムバスを介して，他のプロセッサ，メモリ，I/O デバイスと結合可能である．プロセッサに対しては割込みとシステムバスのマスタ使用権を要求することができ，メモリや I/O デバイスを共有して使用することができる．進化型ハードウェア内のバス IF 回路はシステム内の他のモジュールとのデータ通信を制御する回路であり，マスタモードとスレーブモードがある．マスタモードは進化型ハードウェア自身の命令の実行により起動され，システムバス権を獲得して他のモジュールとの間でデータ通信を行う．スレーブモードでは，メインプロセッサから進化型ハードウェアのリセットなどの状態制御や，レジスタのリード/ライトを行う．すなわち，定型的な処理を行うメインプロセッサと学習により自己組織化する進化型ハードウェアが共存しながら，目的とするマイコン周辺機能を探索することができる．

4. 従来方式との机上での比較検討

4.1 学習効率と設計負荷

図 11 に，マイクロ命令の種類，マイクロ命令に対応したマイクロコード，オペレーションの一例を示す．レジスタは，ソース（R_s），デスティネーション（R_d）ともに最大 64 個指定できると仮定している．提案した手法では，たとえば，クロックサイクルごとに，どのレジスタのデータを用いて，どの命令コードが実行されるかを進化ハードで獲得する．あるいは，命令コードの実行順序はあらかじめ指定しておき，各命令コードで用いるデータの読み書きに用いるレジスタアドレスを指定するマイクロコードのみを進化ハードで獲得することも可能である．すなわち，提案した手法では，設計困難な，あるいは，仕様変更の激しい一部の機能のみを対象として進化を扱うことが容易である．したがって，GA を用いてすべての機能を獲得する場合に比べて，探索空間を縮小でき，機能獲得に必要な遺伝的操作の回数を大幅に削減できる可能性を持つ．一方，FPGA を用いた遺伝的操作と比べて，設計者の負担は大きくなる．たとえば，進化の過程で，レジスタの内容を書き換える命令の実行など，異常動作につながる可能性がある命令の実行は監視する必要がある．すなわち，一般に，設計制約を考慮した進化の枠組みを考える必要がある．

4.2 機能獲得後の実行速度

一般に，μP 制御方式はゲートレベルの結線論理方式と比較して実行速度が遅い．ただし，この比較は，ゲートレベルの結線論理が十分に論理圧縮されている場合のことである．**表 1** に示した真理値表から論理合成した，2 種類のゲートレベル論理回路の例を**図 12** と**図 13** に示す．機能的にはまったく同一であるが，面積で約 3 倍，クリティカルパスのゲート段数で約

表 1 真理値表の例
Table 1 Truth table correspond to the logic circuit shown in Fig. 12 and Fig. 13.

Input				Output
x_1	x_2	x_3	x_4	y_1
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

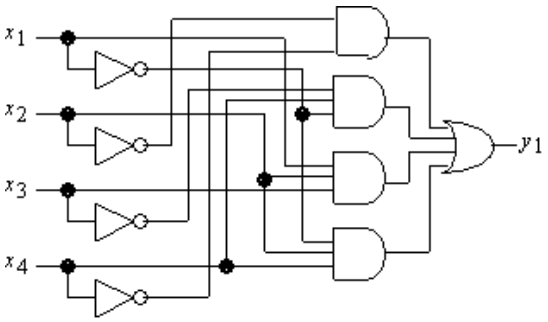

図 12 表 1 に対応した論理図（論理圧縮後）
Fig. 12 Logic circuit corresponds to the truth table shown in Table 1 (After logic minimization).

1.7 倍の差がある．すなわち，動作時の消費電力で約 3 倍，最大遅延速度で約 1.7 倍の差がある．FPGA を用いたゲートレベルの進化の場合，一般的には多くの冗長論理を含んだ回路構成となり，十分論理圧縮された回路が獲得される確率は低い．すなわち，μP 制御方式と比較して，どちらの実行速度が速いかの比較は困難である．むしろ，提案した手法の方が，動作時の消費電力，最大遅延速度ともに安定した見積りが可能であり，工学的な応用を考えやすい．

4.3 信 頼 性

提案した手法では，定型的な処理を行うプログラム部分と学習により自己組織化するプログラム部分の共存が容易に行える．したがって，初期状態から，定型的な処理の部分で，最低限の機能を保証できる．また，定型的な処理の一部を使って，進化的に獲得する処理の異常動作を監視するように設計することができる．さらに，定型的な機能と自己組織化する機能を 1 チップ化しやすく，ボードサイズの縮小に有効である．一般的に，FPGA よりもマイクロプロセッサの方が，チップサイズが小さくなるために故障発生確率も低くなる．

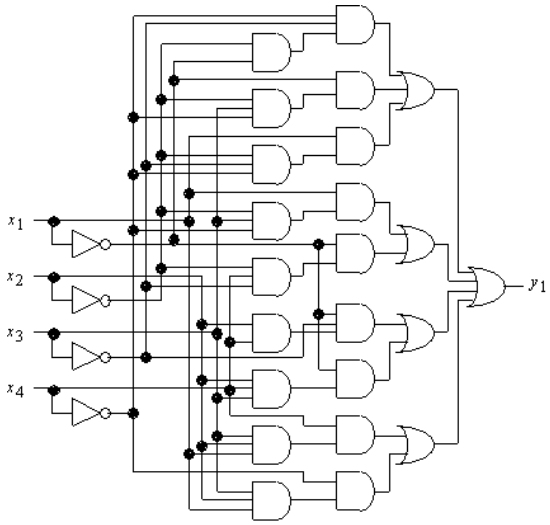

図 13 表 1 に対応した論理図（論理圧縮前）
Fig. 13 Logic circuit corresponds to the truth table shown in Table 1 (Before logic minimization).

5. ま と め

本稿では，レジスタ・トランスファ・レベルでの進化をマイクロプロセッサ上で直接行う，新しい進化型ハードウェアのアイデアを提案した．すなわち，(1) マイクロプロセッサにフラッシュメモリを内蔵することにより，オンボードでのプログラムの書き込みや書き換えを可能にする，(2) 進化的アルゴリズムを用いた，レジスタ・トランスファ・レベルの学習機能を持たせる，(3) 定型的な処理を行うプログラムと学習により自己組織化するプログラムの共存が簡単に行える構成をとる，ことを提案した．また，マイクロコードあるいはタスクシーケンサを用いた具体的な実現案を示した．簡単な机上評価から，FPGA や PLD を用いた従来手法と比較して学習効率および信頼性の観点から有効であり，進化型ハードウェアの工学的な応用分野拡大に貢献できる可能性があることを示した．ただし，上記はあくまでも机上評価であり，今後実際に試作を行い，従来法との比較を改めて行う必要があると考える．

謝辞 本研究を進めるうえで机上評価に関して議論いただいた日立製作所佐藤未来子氏，日立超 LSI システムズ落合辰男氏に深く感謝いたします．

参 考 文 献

1) Higuchi, T., Niwa, T., Tanaka, T., Iba, H., de Garis, H. and Furuya, T.: Evolving Hardware with Genetic Learning, *Proc. Simulation of Adaptive Behaviour*, pp.417–424, MIT Press (1992).
2) Kajitani, I., Iwata, M., Yokoi, H., Nishikawa, D. and Higuchi, T.: An evolvable hardware chip and its application as a multi-function prosthetic hand controller, *AAAI*-99, pp.182–187 (1999).
3) Keymeulen, D., Durantez, M., Konaka, K., Kuniyoshi, Y. and Higuchi, T.: An Evolutionary Robot Navigation System using a Gate-Level Evolvable Hardware, Evolvable Systems: From Biology to Hardware, *Proc. ICES*-1996, *LNCS* 1259, pp.195–209, Springer-Verlag (1997).
4) Tanaka, M., Sakanashi, H., Salami, M., Iwata, M., Kurita, T. and Higuchi, T.: Data Compression for Digital Color Electrophotographic Printer with Evolvable Hardware, Evolvable Systems: From Biology to Hardware, *LNCS* 1478, *Proc. ICES*-1998, pp.106–114, Springer-Verlag (1998).
5) Thompson, A.: An evolved circuit, intrinsic in silicon, entwined with physics, Evolvable Systems: From Biology to Hardware, *Proc. ICES*-1996, *LNCS* 1259, pp.390–405, Springer-Verlag (1997).
6) Miller, J.F.: Digital Filter Design at Gate-level using Evolutionary Algorithms, *Proc. Genetic and Evolutionary Computation Conference*, Vol.2, pp.1127–1134 (1999).
7) Drechsler, R. and Gunther, W.: Evolutionary Synthesis of Multiplexor Circuits under Hardware Constraints, *Proc. Genetic and Evolutionary Computation Conference*, pp.513–518 (2000).
8) Hounsell, B. and Arslan, T.: A Novel Evolvable Hardware Framework for the Evolution of High Performance Digital Circuits, *Proc. Genetic and Evolutionary Computation Conference*, pp.525–532 (2000).
9) Nakamura, H., Sawase, T., Akao, Y., Masumura, S., Hayashi, M., Ohsuga, H., Sato, Y. and Aizawa, T.: An Intelligent Subprocessor for Hardware Emulation with 20-MOPS Performance, *IEEE Journal of Solid-State Circuits*, Vol.26, No.11, pp.1662–1668 (1991).
10) Sato, Y., Shibata, K., Asai, M. and Sugie, M.: Development of a High-Performance, General Purpose Neuro-Computer Composed of 512 Digital Neurons, *Proc. IEEE and INNS Intl. Conf. on Neural Networks*, pp.1967–1970 (1993).
11) Sato, Y.: Proposal for a Field-Evolvable Hardware based on a Microprocessor Incorporated Flash Memory, *Proc. Congress on Evolutionary Computation*, pp.608–615 (2001).
12) Higuchi, T., Iwata, M., Keymeulen, D., Sakanashi, H., Murakawa, M., Kajitani, I. and Takahashi, E.: Real-World Applications of Analog and Digital Evolvable Hardware, *IEEE Trans. Evolutionary Computation*, Vol.3, No.3, pp.293–308 (1999).
13) Ohmori, K.: High-level Synthesis Using Genetic Algorithm, *Proc. IEEE Intl. Conf. On Evolutionary Computing* (1995).
14) Ohmori, K.: VLSI Design Using Genetic Algorithms, *Proc. Australia Pacific Forum on Intelligent Processing and Manufactureing of Materials* (1997).

(平成 14 年 1 月 25 日受付)
(平成 14 年 4 月 9 日再受付)
(平成 14 年 5 月 15 日採録)

佐藤 裕二（正会員）

昭和 56 年東京大学工学部物理工学科卒業．同年（株）日立製作所入所．同中央研究所を経て，平成 12 年 4 月法政大学情報科学部助教授．平成 13 年 4 月同教授．工学博士．可変論理構造 LSI の設計，ニューラルネットワークのハードウェア化，進化的計算を用いた機械学習等の研究に従事．IEEE Computer Society，International Society for Genetic and Evolutionary Computation 各会員．