論　文

FPGAの動的部分再構成を利用した進化型高速パターン認識ハードウェア

川合　浩之[†]　　山口　佳樹[†]　　安永　守利[†]

Development of High-Speed Evolvable Pattern Recognition Hardware Using Dynamic and Partial Reconfiguration

Hiroyuki KAWAI[†], Yoshiki YAMAGUCHI[†], and Moritoshi YASUNAGA[†]

あらまし　再構成可能集積回路と進化計算を用いて構成される進化型ハードウェアに動的部分再構成の機能を適用する．これにより，現実のパターン認識問題で環境の変化等により認識対象のデータに変化が生じた場合でも，即座に動的部分再構成を行い，環境の変化に適応することで常に高い認識精度を保つことができるシステムを提案する．また，パターン認識進化型ハードウェアの高速・高認識精度という特徴を損なうことなく，動的部分再構成を適用することにより，再構成中もパターン認識動作を止めることがない，ノンストップシステムを構築する．FPGA (Field Programmable Gate Array) を用いて提案手法に基づくシステムの実装をソナースペクトル認識を題材に行い，本研究の有効性を示す．提案手法は，従来のシステムよりも30倍以上高速なリコンフィギュレーションサイクルを実現し，現実のパターン認識問題にも十分対応できることを示す．

キーワード　FPGA，動的部分再構成，パターン認識，進化型ハードウェア，遺伝的アルゴリズム

1. まえがき

近年，FPGA (Field Programmable Gate Array) を代表とする再構成可能集積回路の特徴を利用した高速，高性能なパターン認識システムの提案・開発が活発に行われている．とりわけ，進化アルゴリズム (GA：Genetic Algorithms，GP：Genetic Programming) [1] をFPGAに適用することによって実現される，進化型ハードウェア (Evolvable Hardware) [2], [3] に基づいた認識システムが数多く報告されている [4]～[9]．これらのシステムはFPGAの特徴を生かし，アプリケーションに合わせてパターン認識回路の構成を変えることを特徴としている．特にGAの適用により回路構成の最適化を図り，高い認識精度・認識速度を発揮している．

最近の報告では，代表的なものとして文献 [6], [8] が挙げられる．[6] ではVRC (Virtual Reconfiguration Circuit) という構造を提案している．VRCとは，FPGAを論理的にいくつかのブロックに分割し，各ブロック内にあらかじめ数種類の処理回路を実装し，GAによって得た解をもとにマルチプレクサを用いて処理回路の切換を行う手法である．VRCの特徴は基本的にどのFPGAにも実装が可能であるという汎用性にあるが，複数の処理回路の実装により多量の回路リソースが必要となる．また，[8] では，前述VRCを拡張し，FU (Function Unit) という認識処理のためのブロックを構築することで高い認識精度を得ている．なお，進化ハードウェアを用いたパターン認識システムの比較は [10] にて行われている．

我々は進化型ハードウェアに基づくパターン認識システムを [6] や [8] とは異なったアプローチで開発している．これをDDI (Direct Data Implementation) [9], [11], [12] と呼ぶ．DDIの特徴は以下の三つである．

(1) パターン認識に用いるサンプルパターン群を直接，組合せ論理回路としてLSI (Large Scale Integration) 回路内に埋め込む．この際，パラメータの最適化はGAにて行われる．これにより，複雑な演算処理の繰返しが不要で，かつ，並列性を最大限にできるた

[†] 筑波大学大学院システム情報工学研究科，つくば市
Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba-shi, 305-8573 Japan

め，超高速な認識処理が可能となる．

(2) カーネルベース法 [14] というパターン認識の理論に基づき構築されている．これにより，DDI はパターン認識装置としての汎用性が保証されており，ある特定のパターンにしか適用できないということはない．

(3) サンプルパターン群を直接，組合せ論理回路として LSI 回路内に埋め込むことから，ASIC などのフォトマスクベースの LSI によるハードウェア化は非現実的であり（サンプルパターンの変更ごとにフォトマスクを作り直し，新たな LSI を製造する必要があるため），FPGA などのリコンフィギュラブルデバイスを用いることで，初めて現実的なハードウェア化が可能である．

一方で，現実のパターン認識問題では，サンプルパターン群が時々刻々と変化する事例が頻出する．例えば，屋外でリアルタイム顔認識を行う場合，太陽が徐々に雲に隠れていくことにより日差しが弱くなり，認識対象として入力される顔データの輝度が変わり，認識精度が低下していく可能性が高い．日差しの強さに合わせて新たなサンプルパターンを入手し，日差しが徐々に弱くなるに従い新たなサンプルパターンを認識装置に組み込むことが必要となる．また，本論文の **4.** で詳細を説明するが，船舶等で利用されるソナー信号は周囲の環境により変化すると考えられるため，航行に従い新たなサンプルパターンを認識回路に組み込む必要がある．

すなわち，現実のパターン認識問題に対応し，常に安定した認識精度を保つためには，環境の変化に合わせて新たに得られたサンプルパターンを即座に認識回路へと組み込むことが必要不可欠である．しかし，既に述べたように現在の DDI ではサンプルパターンを直接回路化し認識回路を構築するため，即座に新たなサンプルパターンを回路化し，FPGA に再実装することは不可能である．また，FPGA を再構成する際は認識動作が一時ストップするため，頻繁にサンプルパターンの更新が必要なパターン認識問題に DDI は適用できない．

この問題を解決するために，本論文では既に提案されている DDI を拡張し，

（1） 新たに得られたサンプルパターンを即座に学習・回路化し認識回路へ組み込む（オンライン学習・回路化）．

（2） 動的部分再構成の適用により，サンプルパターン変更による回路の再構成中も認識回路としての動作を続ける（ノンストップ認識処理）．

という特徴をもった DP-DDI (Dynamic and Partial DDI) を提案する．更に，DP-DDI を実際に FPGA に実装し評価を行うことで，その有効性を示す．

前述のように，[6], [8] に代表されるような多くのパターン認識用進化ハードウェアが現在まで開発されてきたが，従来のシステムは，環境の変化に合わせて動的に回路の一部を再構成することに対応してはいなかった．本論文の目的は，超高速，高認識精度という DDI の特徴を保ちつつ，動的部分再構成を取り入れることで環境の変化に適応・追従するシステムを提案・開発することである．

本論文は五つの章によって構成される．次章では，DDI の理論と実装方法についての説明を行い，**3.** では DDI を発展させた DP-DDI を提案し，その具体的な実装方法について述べる．更に **4.** では，DP-DDI を用いたソナースペクトル認識システムの試作とその評価結果を述べ，**5.** でまとめを述べる．

2. Direct Data Implementation

我々は，FPGA の書換え可能な特徴を生かした新たなパターン認識ハードウェアである DDI を既に提案し，その有効性を実証してきた [9], [11], [12]．DP-DDI 述べる前に，本章ではこの従来技術である DDI について説明する．

2.1 DDI の理論

DDI は，統計的なパターン認識手法の一つであるカーネルベース法 [14] に基づき，これに GA [1] を適用することで認識精度の向上を図り，更にハードウェア化に適した関数置換を行ったパターン認識手法である．以下，その理論について述べる．

一般的なパターン認識では，未知パターン（n 次元ベクトル）$\boldsymbol{X}$ に対して各カテゴリー C_i ごとに識別関数 $D_i(\boldsymbol{X})$ を作成し，すべての $j(\neq i)$ に対し

$$if\ D_i(\boldsymbol{X}) > D_j(\boldsymbol{X})\ \ then\ \ \boldsymbol{X} \in C_i \qquad (1)$$

のように識別関数の大小関係から $\boldsymbol{X}$ の属するカテゴリーを推定する．ここで，DDI ではカーネルベース法に基づく識別関数 $D_i(\boldsymbol{X})$ として，

$$D_i(\boldsymbol{X}) = \sum_{j=1}^{N_i} K^*(\boldsymbol{X} - \boldsymbol{S}_j^i) \qquad (2)$$

$$K^*(\boldsymbol{x}) = \begin{cases} 1 \ : \ |x_k| \le d \\ \qquad \forall k \in \{1, 2, ..., n\} \\ 0 \ : \ \text{otherwise} \end{cases} \tag{3}$$

を用いる．ここで，$K^*(\boldsymbol{x})$（これを核関数と呼ぶ）は，$\boldsymbol{x}$ が1辺 d（これを核サイズと呼ぶ）の超立方体の内部に入れば1を返す関数である．$\boldsymbol{S}_j^i$ はカテゴリー i に属する j 番目のサンプルパターンを示し，N_i はカテゴリー i のサンプルパターン数である．カーネルベース法では核関数として正規分布型の関数を用いるが，DDIでは核関数としてこれを超立方体型の関数に置換することで，ハードウェア化を容易にしている．なお，この関数置換を行っても十分有効なパターン認識を実行できることを既に示している [9], [11], [12].

式 (2) を識別関数とするということは，以下の識別規則にほかならない．すなわち，「$\boldsymbol{x} = (\boldsymbol{X} - \boldsymbol{S}_i^j)$ を包含する核関数 $K^*(\boldsymbol{x})$ の数を各カテゴリーごとにカウントし，その数が最も多かったカテゴリーをもって $\boldsymbol{X}$ のカテゴリーと推定する」ということである．ここで，核サイズによって認識精度は大きく左右される．このため，パターン認識問題ごとに与えられたサンプルパターンを用いて，高い認識精度が得られるようにその最適な値が決定される．

比較的単純なパターン認識問題では各サンプルパターンに対してすべて一定の核サイズを用いることが可能である．一方，複雑な実応用パターン認識問題においては，核サイズを各サンプルパターンの各次元ごとに独立に与え，最適化することでより高い認識精度を実現することができる．すなわち，式 (2), (3) に代わり，

$$D_i(\boldsymbol{X}) = \sum_{j=1}^{N_i} K_j^i(\boldsymbol{X} - \boldsymbol{S}_j^i) \tag{4}$$

$$K_j^i(\boldsymbol{x}) = \begin{cases} 1 \ : \ |x_k| \le d_{jk}^i \\ \qquad \forall k \in \{1, 2, ..., n\} \\ 0 \ : \ \text{otherwise} \end{cases} \tag{5}$$

を用いる．ここで，d_{jk}^i はカテゴリー i に属する j 番目のサンプルパターンの k 番目の成分に対する核サイズを意味する．厄介なことは，d_{jk}^i の決定が組合せ爆発問題になることである．核サイズのとり得る値を，例えば8通りとし，パターンの次元が60次元とすると，$8^{60} = 2^{180}$ となりすべての網羅的探索（全件探索）は困難となる．このため，我々はこの組合せ爆発問題をGAを用いて解決する．

DDIは，比較的単純なパターン認識においては，式 (2), (3) に基づき実現することができるが，より大規模で複雑なパターン認識においては式 (4), (5) に基づくことが望ましい．よって，我々は式 (4), (5) に基づきシステムを構築した．すなわち，GAによる最適化計算を含めたシステム開発を行った．これは，もちろん，GAを使わないシステム（すべてのサンプルパターンに対して同じ核サイズを用いるシステム）としても利用することができる．

2.2 DDIの回路化

式 (4), (5) で示す核関数 K_j^i はサンプルパターン $\boldsymbol{S}_j^i$ と核サイズ d_{jk}^i が定数であるので，入力パターン $\boldsymbol{X}$ に対して，各成分ごとに1または0を出力する真理値表の論理積 (AND) として表現することができる．

n 次元パターン空間の一つの成分（次元）に対する真理値表の例とそれに対するゲート回路（基本ゲート回路と呼ぶ）をそれぞれ表1，図1に示す．この例（表1）はサンプルパターン (Sample Pattern) $\boldsymbol{S}_j^i$

表1 サンプルパターンから生成される真理値表
Table 1 Truth-table generated from sample pattern.

$S_{jk0}^i = 0$ $S_{jk1}^i = 1$ $S_{jk2}^i = 1$
$d_{jk0}^i = 0$ $d_{jk1}^i = 1$ $d_{jk2}^i = 0$

Input X_k			Output
X_{k0}	X_{k1}	X_{k2}	Z
0	0	0	0 (3_{10})
0	0	1	1 (2_{10})
0	1	0	1 (1_{10})
0	1	1	1 (0_{10})
1	0	0	1 (1_{10})
1	0	1	1 (2_{10})
1	1	0	0 (3_{10})
1	1	1	0 (4_{10})

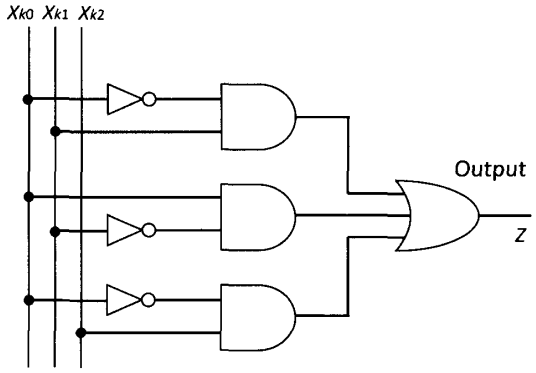


図1 基本ゲート回路の例
Fig. 1 Example of Basic-gate circuit.

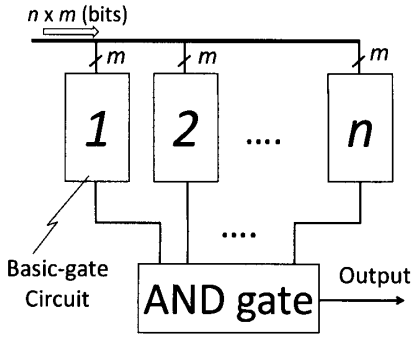


図 2　核関数回路
Fig. 2　Example of Kernel-function circuit.

の k 番目の成分 S^i_{jk} に対する真理値表である．ここでは分かりやすくするために各成分 3 ビット精度としている（添字 0，1，2 は 3 ビットの各ビットを表す）．なお，本論文で後述するソナースペクトル認識システムでは 4 ビット精度を用いている．この例では 3 ビット精度としたため，入力される未知パターン (Unknown Pattern) $\boldsymbol{X}$ の各成分 X_k は $8(=2^3)$ 通りとなる．この例では $S^i_{jk}=011_2=3_{10}$ で，かつ，核サイズ $d^i_{jk}=010_2=2_{10}$ とした場合を示している．この固定値 $(S^i_{jk}=011_2, d^i_{jk}=010_2)$ に対して X_k のとり得る値は 8 通りであり，表 1 はこのすべての入力 X_k に対しての出力 (Output) を網羅している．Output の欄の値は，核サイズの内部に未知パターンの k 番目の成分が入れば 1，入らなければ 0 を出力することを意味しており，( ) 内は 10 進数で表した $|S^i_{jk}-X_k|$ の値である．$|S^i_{jk}-X_k|$ の値が $d^i_{jk}=2_{10}$ 以下であれば，Output=1 であり，それ以外は，Output=0 である．そしてこの真理値表が示す回路（基本ゲート回路）は単純な 3 入力 1 出力の組合せ回路で構成できる（図 1）．

このように，サンプルパターンとそれに対する核サイズ d^i_{jk} が決まれば各成分ごとに真理値表を自動生成することができる．更に，生成された真理値表をハードウェア記述言語 (HDL：Hardware Description Language) による回路記述の中に埋め込むことにより，基本ゲート回路を自動生成することができる．

次に，図 2 に示すように n 次元ベクトル $\boldsymbol{X}$（各成分は m ビット表現）の各成分に対する基本ゲート回路 (Basic-gate circuit) を全成分について AND gate でまとめることにより，式 (5) を回路化することができる．この回路を核関数回路と呼ぶ．

核関数回路（図 2）をもとに，これに式 (1)，(4) に対応する回路を付加したパターン認識回路全体を図 3 に示す．回路全体は，核関数回路 (Kernel-function circuit)，1-出力カウンタ (Ones counter)，最大値検出回路 (Max. Detector) によって構成される．同一カテゴリー (C_i) ごとに，出力が 1 である核関数回路（すなわち，入力パターン $\boldsymbol{X}$ が核の中に入って活性化されている核関数回路）の数を 1-出力カウンタでカウントする．この 1-出力カウンタの出力値が式 (4) の識別関数 D_i の値となる．すなわち，各カテゴリーごとの核関数回路群と 1-出力カウンタが各カテゴリーごとの識別関数回路を形成する．Max. Detector は，すべてのカテゴリーの 1-出力カウンタの最大値を検出し，そのカテゴリーを出力する（式 (1) を実現する）．

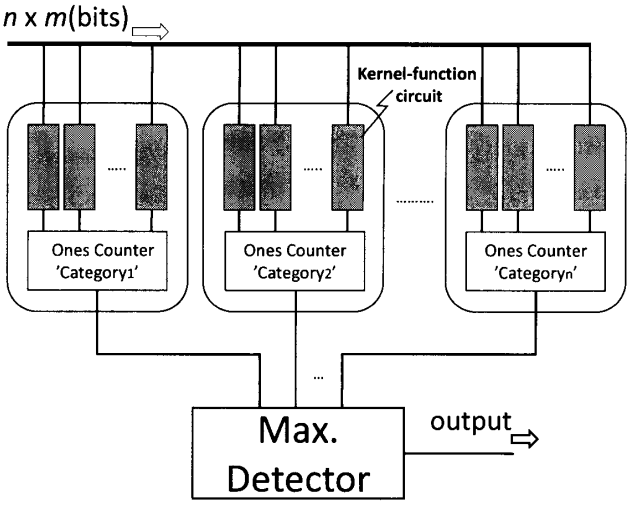


図 3　DDI によるパターン認識回路
Fig. 3　Whole pattern recognition circuit based on DDI.

以上，我々が既に提案し，その基本的な有効性を示してきた DDI の理論とその回路化について述べた．DDI には以下の特徴があり，これにより高い認識精度（カーネルベース法に基づく）に加えて，超高速な認識処理を実現している．

（1）　核関数計算や識別関数計算は，真理値表から直接生成された組合せ回路により行い，演算器は用いない．すなわち，サンプルパターンを組合せ回路として直接埋め込み，核関数計算や識別関数計算を直接実行する．

（2）　サンプルパターン数 × 全成分数（次元数）と同数の核関数回路を用いることにより，対象問題の並列度を最大限に生かした並列処理が実現できる．

これらの特徴より，DDI では超高速な認識処理が実現でき，後述する DP-DDI も同じようにこの特徴を有する．本論文の最後に開発したシステムの認識処理速度の測定結果を示すとともに，ソフトウェアを用いて

測定した値との比較と考察を行う．

3. DP-DDI（動的部分再構成 DDI）

3.1 DP-DDI の基本的な実現方法

DDI の問題点を解決し，本論文で提案する DP-DDI を実現するためには，DDI が実装された FPGA の認識動作を止めることなく，一部の核関数回路の基本ゲート回路群を新たに得られたサンプルパターンに応じて書き換えることが必要である．一般に既存の FPGA では一部の回路を変更する必要が生じた場合は，FPGA の回路設計と回路書換えを行う統合ソフトウェア上（例えば，Xilinx 社の ISE や，Altera 社の QuartusII）にて，

(i) 回路の一部を変更した後に全体回路を論理合成し，FPGA のビットストリームを作成する

(ii) FPGA の動作を止めて，生成したビットストリームにより FPGA 全体を書き換える

必要がある．このため，従来の DDI でサンプルパターン更新に対応するには，図 4 (a) が示すように，更新サンプルパターンも含めた全サンプルパターンから生成された真理値表 (Truth tables of sample pattern)，また，図 4 (a) には記述されていないが，1-出力カウンタと最大値検出回路の HDL データを統合ソフトウェア (CAD tools) へ送り，統合ソフトウェアはこれに対して，論理圧縮 (Design & Logic Reduction)，論理合成 (Logic Synthesis)，ターゲットとする FPGA デバイスへの配置配線 (Implementation)，FPGA へ実際に送る回路データであるビットファイルの生成 (Generate Bitstream) を実行する．すなわち，FPGA に実装する前にすべての回路の論理圧縮を行うため，FPGA 内の LUT (Look-Up Table) には論理圧縮された回路が実装される．よって一部の真理値表のみを変更し，FPGA に再実装する場合でも，すべての HDL データを使って再び論理圧縮からビットストリーム生成までを再実行しなくてはならない．

この問題点は，上記処理の再実行に長い時間を要することと，上記 (ii) のとおり，再生成したビットストリームを用いて FPGA を書き換えるときに動作が一時ストップすることである（これらの具体的な実測値は **4.** にて示す）．したがって，このままでは DP-DDI を実現することはできない．我々は DDI の特徴を生かすことで上記の (i)，(ii) にはよらない方法で FPGA を書き換え，DP-DDI を実現する方法を提案する．具体的には以下のとおりである．

既に述べたように，DDI の基本となる基本ゲート回路は，サンプルパターン $\boldsymbol{S}_j^i$ から作成される真理値表と等価である．一方，現在一般的に使用されている FPGA の LUT は真理値表そのものであり，したがって，$\boldsymbol{S}_j^i$ の真理値表（基本ゲート回路）は，FPGA 内の一個または複数個の LUT に直接マッピングすること

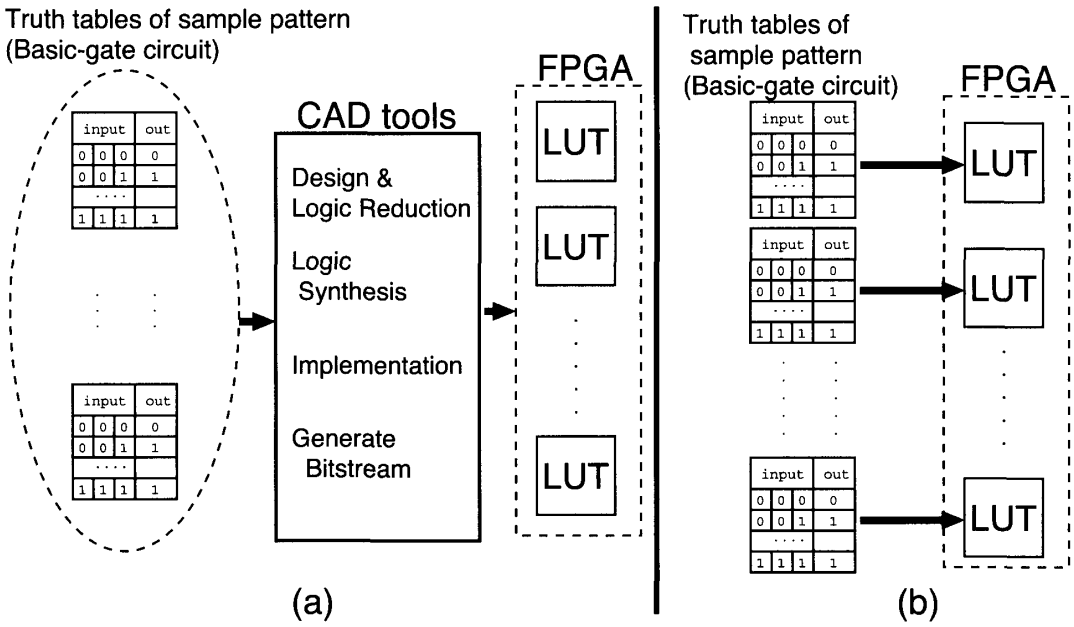


図 4 従来の DDI と DP-DDI の比較
Fig. 4 Comparison between DDI and DP-DDI.

で実装可能と考えられる（図 4 (b)）．我々は DP-DDI 実現のためにこの特徴を利用する．

なお，現在一般的に使用されている FPGA の中で，例えば Xilinx 社の FPGA では LUT の内容（真理値表の値）を統合ソフトウェアを用いずに FPGA の動作中に直接書き換えることができる機能を提供している．したがって，この機能を利用すれば，FPGA の動作を止めることなく，DDI の一部の核関数回路を新たに得られたサンプルパターンに応じて書き換えることができる．また，DP-DDI を用いたシステムを運用する際，回路書換えのために統合ソフトウェアのライセンスは必要なく，ユーザがソフトウェアの使用方法を覚える必要もなくなるという利点が生まれる．

一方，統合ソフトウェアを用いず，サンプルパターン $\boldsymbol{S}_j^i$ に基づく真理値表を直接 LUT に実装する場合の問題点は，論理圧縮が行われない点である．例えば従来の DDI では，同じ出力値の真理値表は一つにまとめられ FPGA にマッピングされることがある．また，真理値表の出力によっては LUT が使用されない場合がある（基本ゲート回路と LUT が 1 対 1 で対応するわけではない）．このため，FPGA 上に実装できるサンプルパターン数が従来の DDI に比べて減少することが考えられる．この点については，後述の実際の評価において定量的に議論する．

3.2 DP-DDI の具体的な実現方法

現在我々が DP-DDI の実装対象としている Xilinx 社の FPGA には，上述したように LUT の出力値を直接（統合ソフトウェアを使わず）書き換える機能が備えられている．具体的には，SRLC16E（または SRL16E）という LUT の機能をもったシフトレジスタ (SR) を用いて，シフト操作によって LUT の出力値を変更する方法である [15], [17]．これらの SR を図 5 に示す．これらの SR は通常の LUT としても動作する．すなわち，LUT としての出力値を保存しており，Address 線に 4 入力のアドレスデータを入力すると，それに対応する値が Q 線より出力される（通常の 4 入力 LUT と同じく $2^4 = 16$ ビットを保持する）．また，CE はクロックイネーブルを示しており，high が入力されたとき，シフト処理が実行される（D 線に入力された値がシフトされる）．ここで，二つの SR の違いは Q15 出力である．Q15 線には LUT の出力値の最終ビットが出力される．よって，この Q15 線を次の SR の D 線に入力することで，カスケード接続が容易に実現できる．一つの LUT に基本ゲート回路が実装できる場合，DP-DDI では SRLC16E をカスケード接続することで核関数回路を実現することができる．また，基本ゲート回路を実現するために，複数の LUT が必要な場合（例えば，6 ビット精度のデータを 4 入力の LUT で実現する場合），図 6 のように SRLC16E を組み合わせることで基本ゲート回路を構築できる．図中の実線は，パターン認識処理の流れを示しており，破線はシフト操作による再構成の流れを示している．破線は図 5 の Q15 出力と D 入力を結んでいる．

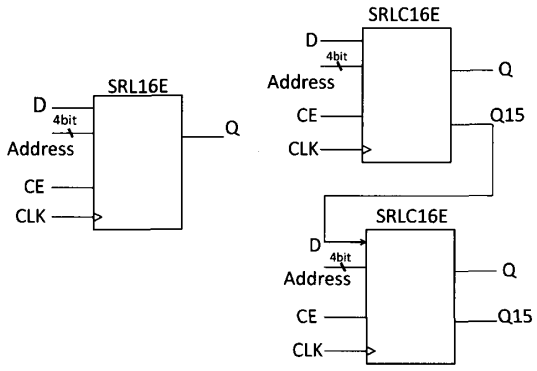


図 5　本研究で用いるシフトレジスタ
Fig. 5　Two kinds of shift registers intstantiated.

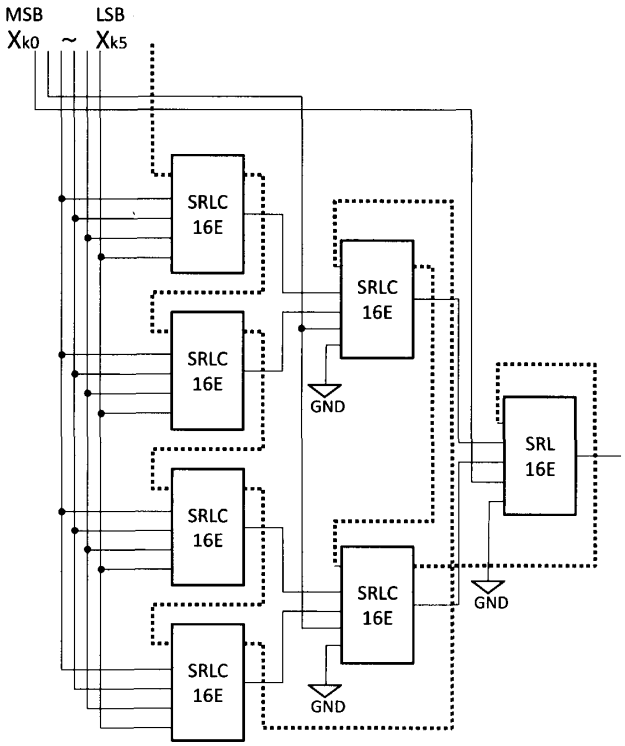


図 6　複数の LUT を用いて基本ゲート回路を構成する例（6 ビット精度）
Fig. 6　Realization of basic-gate circuit using multiple LUTs (6-bit precision).

次に，核関数回路の構造を DP-DDI に特化したものに改良する．核関数回路が図 2 の構造のままでは，再構成中（シフト操作中）に AND ゲートの出力が “1” となる場合があり，認識結果に影響を与える可能性がある．再構成中の回路を他の回路から完全に切り離すために，再構成中の核関数回路の AND ゲートは常に “0” を出力する必要がある．これを実現するために，我々は図 2 の AND ゲートに入力線を一本追加する．再構成中の核関数回路には接続された入力線から “0” が入力され，それ以外の核関数回路には “1” が入力される．これにより，再構成中の核関数回路内の AND ゲートは常に “0” を出力することが保証される．DP-DDI では入力線が追加された核関数回路を常に使用する．

そして，複数の核関数回路を図 7 に示すようにカスケード接続することを提案する．図中の K_j^i はカテゴリー i に属する j 番目の核関数回路を示している．各カテゴリーには m 個の核関数回路が存在し，総カテゴリー数は n とする．ここで，図 7 が示すように異なるカテゴリーの j 番目の核関数回路をカスケード接続し，これを単位として部分再構成を行うことを提案する．これにより，各カテゴリーの j 番目以外の核関数回路は動作を続けつつ，j 番目の回路を再構成することが可能となり，DP-DDI の特徴である動的部分再構成が実行可能となる．カスケード接続された一つの単位を IPRU (Isolated Partial Reconfigurtaion Unit) と呼ぶ．IPRU 構成のために，各カテゴリーの j 番目の回路を接続する理由は以下である．

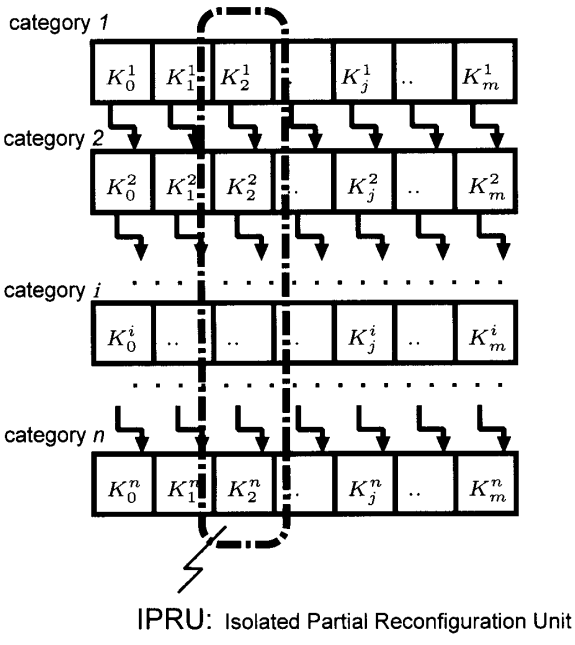


図 7　動的部分再構成を実現するための構造
Fig. 7　Structure to achieve dynamic and partial reconfiguration.

一般的なパターン認識問題では，同じ環境下で得られたサンプルパターンを一つのグループとしてまとめ，それを一括で更新することで，高い認識精度を得る．例えば，実際の顔画像認識では同じ明るさ（環境下）で撮影された画像を一つのグループにまとめる．これを必要に応じて違う環境下で撮影されたグループと置き換えることで，認識精度を向上できる．各カテゴリーの j 番目のデータを一つの書換え単位とすることで，常に高い認識精度を得ることができると考えられる．

このように，基本ゲート回路をカスケード接続して核関数回路を構築し，更にそれをカスケード接続することにより効率良く回路を再構成することができる．なお，本論文では IPRU を再構成するためのデータを Configuration data と呼び，統合ソフトウェアによって生成された FPGA にダウンロードするための回路情報ファイルをビットストリームと呼ぶ．

4. プロトタイプシステムの構築

我々は，ソナースペクトル認識システムを対象として DP-DDI のプロトタイプシステムを構築した．ソナーとは，船舶が音波を発信し，その反射音が戻る時間から周囲の物体までの距離を測定するシステムである．一方，反射音は反射物体の特徴を含んでおり，そのスペクトル（周波数応答）から反射物体を識別するシステムの構築も可能である．これがソナースペクトルの認識システムである．

既に，ソナースペクトル認識システムに DDI を用いることを我々は提案し，その基本的な有効性を示した [11]．しかしその後，DDI を実際の船舶で利用するには大きな課題を解決しなければいけないことが分かった．その課題は次のとおりである．実際に海洋を航行する船舶のソナー信号は，時々刻々と変化する周囲の環境の影響を受けるため，同じ物体であってもそのスペクトルは周囲環境に伴って変化する．このため，周囲環境の変化に伴う新たなサンプルパターンを動作中に常に取得・更新し，これに対応した核関数回路を生成し，システムを書き換える必要がある．

従来の DDI を用いた場合，前章で述べたように，一部の核関数回路の更新のために統合ソフトウェアを用いた全体回路書換えが必要である．このため，一部の

回路を更新するために多大な時間がかかり，周囲環境の変化に適応することが困難となる．また，全体回路の書換え中は認識処理ができないため，認識処理が一時中断する（ノンストップ処理ができない）ことも問題となる．

そこで我々は，この問題を解決するために DP-DDI によるソナースペクトル認識システムの開発に着手した．本章では，まずソナーデータに関する説明を行い，そして，開発したプロトタイプシステムとその評価結果について述べる．

4.1 評価データと評価の準備

本評価で用いるソナースペクトルのパターンデータ（以下，ソナーデータ）はカーネギーメロン大学より提供されている潜水艦ソナースペクトルの実パターンデータ [18] であり，岩 (Rock) と金属 (Mine) の 2 カテゴリーのソナーデータである．図 8 は金属を示すソナーデータの例である．データの総数は 208 個で金属のデータは 111 個，岩のデータは 97 個である．一つのソナーデータは 60 個の周波数成分 (frequency) における信号強度 (Signal intensity) を示しており，60 次元のベクトルで表される．また，信号強度は [0, 1] の範囲で正規化されている．元データのままではデータ量が多いため，前処理として信号強度 [0, 1] を 16 分割する．つまり信号強度を 4 ビット精度で表す．先行研究 [9] より，8 ビット精度での実験がパターン認識問題において一般的であると考えられるが，本研究では 4 入力 1 出力の LUT をもった FPGA (Xilinx 社 Virtex-4) をターゲットデバイスとするため，その構造に合わせて 4 ビット精度のデータを用いた．

予備実験として，4 ビット精度と 8 ビット精度のデータでの認識精度の差を確認するため，ソフトウェアシミュレーションによって認識精度を文献 [16] と同じ leave-one-out 法によって評価を行い，比較した結果が表 2 である．DDI と DP-DDI は全く同じ認識アルゴリズムであるため，両者の認識精度に差はない．また，比較のため，ニューラルネットを用いて認識精度を測定した文献 [16] の結果を記載した．8 ビット精度のデータを用いたときは，ニューラルネットとほぼ同じ認識精度を示している．4 ビット精度とすることにより，認識精度の低下は見られるが，それでも 77.9% という高い水準を満たしている．なお，先に述べたとおり，この認識精度を得るために GA を用いて核サイズを決定した．

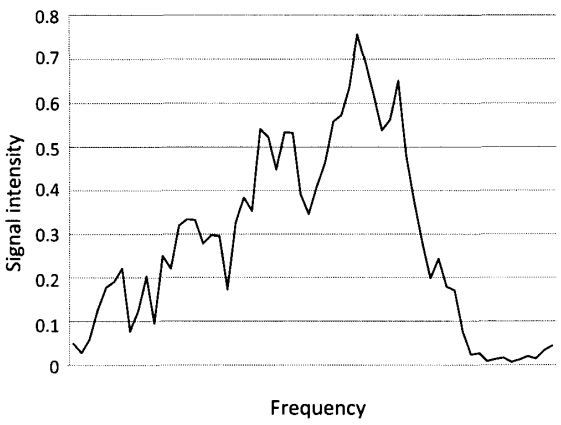


図 8　ソナーデータ（金属）の例
Fig. 8　Example of sonar spectrum data: Mine.

4.2 GA の遺伝操作

GA 処理ではエリート選択と一点交差を用いた．また，突然変異確率は 1% とした．GA 実行時のパラメータを表 3 に示す．本評価では遺伝子の種類，すなわち核サイズ d^i_{jk} の値は 1/8, 2/8, ..., 8/8 の 8 通りとした．適応度関数などの詳細なパラメータに関しては，付録に記載する．

4.3 2 種類の DP-DDI システム構築

本論文では，ソナーデータを用いて 2 種類の DP-DDI システムを構築し，これらを従来の DDI システムと比較・評価することで DP-DDI の有効性を示す．最初に，DP-DDI を実装した FPGA を PC (Personal Computer: Core2 Duo 2.66 GHz) と接続した DP-DDI プロトタイプシステムを構築した．更に，ソフトコア CPU を FPGA 内に実装し，PC を必要とせず DP-DDI を実現するオンチップ DDI システムの試作も行った．これらのシステム構築のために，Xilinx 社から提供されている FPGA ボード ML401 Evaluation board を用いた．このボードには Virtex-4 XC4VLX25 FPGA (LUT 数：21,504，BlockRAM 容

表 2　ソフトウェアシミュレーションによる認識精度の比較
Table 2　Comparison of each system's recognition accuracy under the software simulation.

4 ビット精度 DDI	77.9%
8 ビット精度 DDI	82.9%
8 ビット精度ニューラルネット	83.0%

表 3　評価に使用した GA のパラメータ
Table 3　GA parameters used in evaluation.

個体数	14	交差確率	40%
学習世代数	1200	突然変異確率	1%

量：1,296 kbit) が搭載されている．以下，これらのシステムの詳細を記述する．

4.3.1 PC を併用した DP-DDI システム

図 9 は我々が構築した DP-DDI プロトタイプシステムを示す．FPGA ボードと PC は RS-232C で接続されている．図中の表記で，例えば K_{1_mine} は一番目の金属データを用いて生成された核関数回路であり，本システム内の核関数回路の数は 100 個（前述のとおり岩と金属の二つのカテゴリーに対して 50 個ずつ）である．また，この認識問題は岩と金属の 2 カテゴリーであり，したがって，二つの核関数回路がカスケード接続されている．FPGA 内には UART (Universal Asynchronous Receiver Transmitter) controller が実装され，RS-232C 経由で送られたデータの受信が行われる．PC 上で GA が実行され，GA 処理が完了した時点で，Configuration data とどの IPRU を再構成するかの Address データが送信される．Decorder を通して，指定した IPRU を再構成可能状態にし，そこへ I/O buffer から Configuration data を送信することによって動的部分再構成を実行する．FPGA 内のメモリ部 (BlockRAM) には認識精度評価のためのテスト入力データ (Unknown data) が格納されている（詳細は **4.4** で説明する）．この PC が接続された DP-DDI プロトタイプシステムを以降では単に “PC 併用 DP-DDI” と記述する．

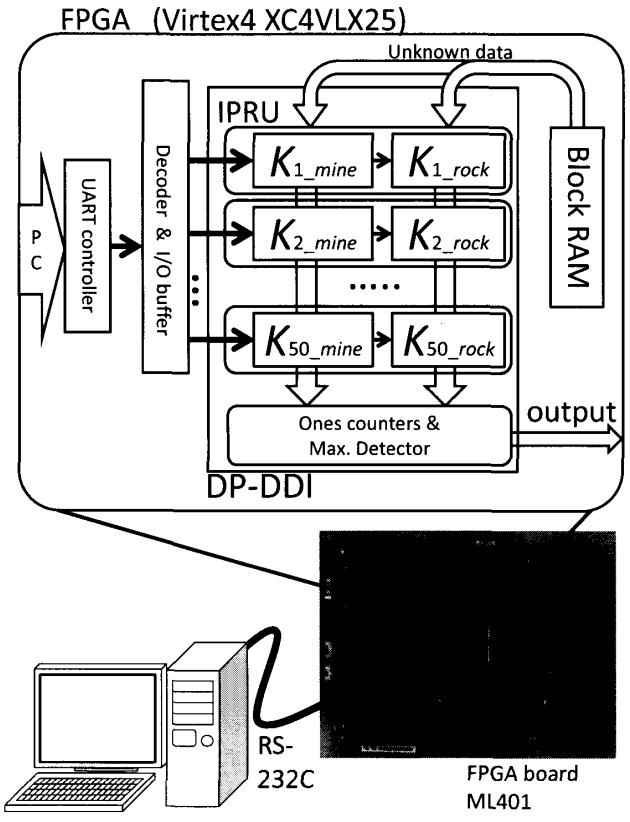


図 9 PC を併用した DP-DDI システム
Fig. 9 Overall structure of the DP-DDI using PC.

4.3.2 オンチップ DP-DDI システム

前項で述べた DP-DDI プロトタイプは，PC による GA 処理の結果を元に IPRU を書き換えている．したがって，この PC の機能を FPGA 内で実現することで，オンチップ DP-DDI の構築が可能である．オンチップで DP-DDI が実現することで PC が不要となり，軽量，コンパクトな超高速パターン認識システム実現への道が開かれる．我々は更に，オンチップ DP-DDI を実現するためのプロトタイプ開発を，Xilinx 社より提供されている FPGA のための組込みシステム開発用ソフトウェア EDK (Embedded Development Kit) を用いて試みた．

図 10 は我々が提案するオンチップシステムの全体図である．図中の各ブロックは EDK が提供する PLB (Processor Local Bus) によって接続されており，ブロック間のデータの送受信は PLB 経由で行うことができる．認識精度評価用のテスト入力データ (Unknown data) は Data I/O Buffer を経由して，DP-DDI に送られる．そして DP-DDI は Data I/O buffer に結果を返す．Decoder は DP-DDI 内のどの IPRU を再構成するかを制御する．MicroBlaze は Xilinx 社が提供するプロセッサマクロである．MicroBlaze によって生成された Configuration data が指定した IPRU に送られ，動的部分再構成が実行される．UART は RS-232C によって PC と接続するための回路であり，オンチップシステムの動作をモニタするために使われ

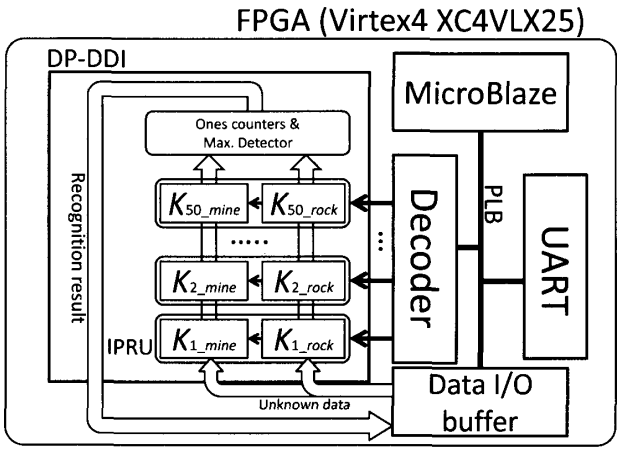


図 10 オンチップ DP-DDI システムのブロック図
Fig. 10 Block diagram of on-chip DP-DDI.

る．また，更新用のサンプルデータは Microblaze のメインメモリ（実際には BlockRAM が使われる）に格納されている．

4.4 システム性能比較

DP-DDI では，GA 処理を含めてサンプルパターンの更新（回路再構成）をどれだけ高速に行うことができるか，そして，再構成に伴い認識精度がどのように変化するかを評価する．

評価実験のため，最初にソナーデータを図 11 に示すように三つのグループに分割した．つまり，岩と金属のソナーデータからそれぞれ無作為に 50 個のデータを選び，それらを核関数回路生成のためのサンプルデータ (Sample data) として，FPGA に実装する（図 11 (a)）．次に，残りのデータからそれぞれ無作為に 10 個のソナーデータを選択し，それを認識精度評価用のテスト入力データ（Test data，図 11 (b)）とする．(b) のデータは FPGA 内の BlockRAM に格納されている．最後に，各カテゴリーの残ったソナーデータ（Extra Sample data，図 11 (c)）は更新用サンプルデータとして一つ一つ順番に回路化され，回路として既に FPGA に実装されているサンプルデータの更新（回路再構成）用として使われる．(c) のデータは PC 併用 DP-DDI では PC 上に，オンチップ DP-DDI では BRAM 内に格納されている．50 個のサンプルデータの中からどのデータを削除するか，また，どの更新用サンプルデータを用いて再構成するかについては無作為に選択される．このデータ更新による回路再構成を行ったときのリコンフィギュレーションサイクル（GA 処理から実際に回路再構成を終えるまでの時間）を表 4 にまとめる．

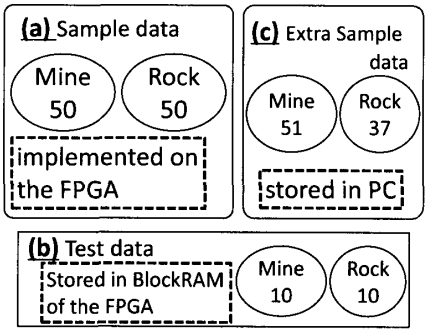


図 11　評価実験のためのソナーデータ分割
Fig. 11　Set of sonar data used for experiments.

表中の PC 併用 DP-DDI では，PC を使って GA 処理を行っている．このために必要な時間は，11.74 秒である．また，Configuration data は，GA によって得られた最優秀個体の値そのものであるため，生成に必要な時間は 0.01 秒以下である．Configuration data を RS-232C を介して FPGA ボードに送り，一つの IPRU を再構成するためにかかる時間は 3.98 ミリ秒である．

一方，オンチップ DP-DDI では，Microblaze を用いて GA 処理を行うため，PC を用いた場合に比べてかなり多くの時間を要する（870 秒）．Configuration data を生成するための時間は，PC 併用 DP-DDI と同様に 0.01 秒以下である．そして，Configuration data を Microblaze から PLB を経由して DP-DDI 部に送り，一つの IPRU を再構成するために必要な時間は 2.39 ミリ秒である．

また，従来の DDI の場合は PC 併用 DP-DDI と同じく GA 処理は PC 上で行われるため，11.74 秒を要する．次に，統合ソフトウェアを用いて論理合成を行い，ビットストリームを生成するまでの時間は 363 秒である．最後に，FPGA の動作を一度ストップし，

表 4　リコンフィギュレーションサイクル（時間）比較
Table 4　Measurement of reconfiguration cycle in each system.

処理内容＼システム		PC 併用 DP-DDI	オンチップ DP-DDI	従来の DDI
GA 処理		11.74 秒	870 秒	11.74 秒
回路情報生成	Configuration data 生成	0.01 秒以下	0.01 秒以下	
	論理合成，ビットストリーム生成			363 秒
再構成	動的部分再構成	3.98 ミリ秒	2.39 ミリ秒	
	全体再構成			2.94 秒
合計		12 秒	870 秒	378 秒

表 5 認識精度比較
Table 5 Measurement of recognition accuracy in each system.

システム / 状態	PC 併用 DP-DDI	オンチップ DP-DDI	従来の DDI
通常動作中	80.12%		
再構成中（再構成時間）	79.1%（3.98 ミリ秒）	79.1%（2.39 ミリ秒）	0%（2.94 秒）

JTAG によるコンフィギュレーションパスを用いて FPGA 全体を再構成するためにかかる時間は 2.94 秒である．

各システムの認識精度に関しては，サンプルパターンの更新を 30 回行ったときの平均値を表 5 に記述する．すべてのシステムは同じ認識アルゴリズム（ハードウェア実装のために最適化されたカーネルベース法）用いているため，表 5 が示すとおりシステム通常動作中の認識精度はすべて 80.12%である．しかしながら大きな違いは再構成中の認識精度である．DP-DDI を用いたシステムの場合は，動的部分再構成が行われるため，再構成中の認識精度は 79.1%とその低下は非常にわずかである．この認識精度低下の理由は IPRU の再構成により，核関数回路の数が減ることが原因と考えられる．ただし，括弧内に示されているとおり動的部分再構成が行われる時間はわずか数ミリ秒であり，この値は従来の DDI（2.39 秒）に比べ 600 倍以上高速であるためこの認識精度低下がシステム運用の際に問題になることはないと考えられる．一方，従来の DDI では全体再構成が必要となるため，認識精度は 2.39 秒の間 0%（システム停止）となり，頻繁に再構成が必要となるパターン認識問題を扱う場合，大きな問題となる可能性がある．この結果より，DP-DDI を用いることでノンストップシステムが構築でき，更に再構成中の性能の低下はほとんど発生しないことが分かる．

本論文で試作した PC 併用 DP-DDI では，ノンストップでパターン認識を実行しながらリコンフィギュレーションのサイクルを約 12 秒とすることができた．これは，従来の DDI に比べ 30 倍以上高速であり，通常の船舶が巡航速度で移動する際のソナースペクトルデータの変化に対して十分追従するリコンフィギュレーションサイクルであると考えられる．また，ソナースペクトル認識に限らず，サンプルパターンが時々刻々と変化する多くの実応用パターン認識システムに適用できると考えられる．オンチップ DP-DDI で

表 6 PC 併用 DP-DDI とオンチップ DP-DDI の Configuration data 転送速度の比較
Table 6 Comparison of the transfer speed for Configuration data in each system.

システム	PC 併用 DP-DDI	オンチップ DP-DDI
送信方法	RS-232C	PLB
転送速度	9,600 bit/s	800 Mbit/s
再構成時間	3.98 ミリ秒	2.39 ミリ秒

は GA 処理のために多くの時間が必要となるため，リコンフィギュレーションサイクルの高速化を現時点では実現できていない．しかし，動的部分再構成によるノンストップ処理を実現しており，また，再構成時間は PC 併用 DP-DDI よりも短い（この理由は次節で述べる）．将来的には，FPGA やソフトコア CPU の性能向上により GA 処理時間が短縮され，実用に耐え得る性能をオンチップ DP-DDI で実現することが可能であると考えられる．

4.5 考　　察

4.5.1 動的部分再構成に必要な時間の比較

一つの IPRU を再構成するために要する時間は，PC 併用 DP-DDI は 3.98 ミリ秒であり，オンチップ DP-DDI は 2.39 ミリ秒であった．この速度差は，Configuration data の送信方法の違いにより生じるものである．表 6 が示すように，PC 併用 DP-DDI，オンチップ DP-DDI はそれぞれ RS-232C，PLB を用いている．RS-232C の転送速度は 9,600 bit/s であり，PLB は 800 Mbit/s であるため，オンチップ DP-DDI では Configuration data を非常に高速に送ることができる．しかし，実際に再構成時間の差は 1.7 倍程度であり，転送速度との比は大きく異なっている．この原因は，IPRU の再構成方法にある．つまり，IPRU を構成するためにカスケード接続された SR を用い，シフト操作によって 1 ビットずつ再構成が行われていくため，PLB による転送速度に比べて IPRU 再構成の速度が遅いことが理由である．しかし，どちらの場合も既に数ミリ秒という高速な再構成時間を実現して

表 7　GA 実行用の CPU とメインメモリの性能
Table 7　CPU and main memory for GA execution in each system.

CPU	MicroBlaze	Core2 Duo
周波数	100 MHz	2.66 GHz
メインメモリ	64 kByte	3.25 GByte

いるため，この速度差が問題になることはないと考えられる．

4.5.2　GA 処理に必要な時間の比較

オンチップ DP-DDI では，GA を実行して核サイズを求め LUT の書換え値を決定するために 870 秒を要した．PC 併用 DDI では GA 実行時間は 11.74 秒であり，オンチップ DP-DDI は PC 併用 DP-DDI に比べ，約 74 倍もの時間を要している．表 7 に示すように両システムで用いる CPU の動作周波数は約 26 倍の差であるにもかかわらず，GA を実際に実行するとそれ以上の差が生じている．これは，図 9 の CPU はデュアルコアであること，またメインメモリの容量が大幅に異なることが要因として挙げられる．今後オンチップ DP-DDI を実用化するにはこの GA 実行時間を高速化する必要がある．そのためには GA 処理の並列化を行うことが必要であると考えられる．特に，GA の評価関数による各個体の適応度評価の際，並列化が有効である．よって，

（1）　Xilinx 社より提供されている FPGA を用いて，MicroBlaze のマルチコアシステムを構築することが可能であるため，複数の CPU で GA を実行する．

（2）　GA 処理の一部をハードウェア化して実行し，CPU の負荷を分散する．

といった方法が考えられる．

4.5.3　回路規模の比較

DP-DDI はサンプルデータから生成される真理値表を直接 LUT にマッピングし，これを書き換えることで高速な動的部分再構成を実現している．したがって，回路が圧縮されていないため，従来の DDI に比べて回路規模が増加すると考えられる．本論文で試作したシステムの回路規模比較を表 8 に示す．PC 併用 DP-DDI では使用した LUT の 73%がシフトレジスタとして使われ，また，オンチップ DP-DDI では使用した LUT の 51%がシフトレジスタとして使われている．

PC 併用 DP-DDI では，従来の DDI に比べ 2.5 倍の LUT が必要となる．しかし，それでも本研究で用いた FPGA の回路使用率は 38.2%であり十分余裕のある実装が可能であった．40,000 個以上の LUT をもったミドルクラスの FPGA を用いれば更に多くの核関数回路を余裕をもって実装することができる．一方，オンチップ DP-DDI では従来の DDI に比べ約 3.6 倍の LUT が必要である．また，フリップフロップ使用数も他のシステムに比べて多い．この理由は，図 10 で示したようにソフトコア CPU である MicroBlaze（LUT 数：1,847 個）やその他の回路の実装が必要なためである．しかし，オンチップ DP-DDI についても，ミドルクラスの FPGA に十分実装が可能であることが分かる．また，これは PC を必要としないシステムであるため，結果的には他のシステムに比べ明らかに低消費電力，コンパクトなシステムになる．

表 8　各システムの回路規模
Table 8　Circuit size of each system.

システム	**PC 併用 DP-DDI**	**オンチップ DP-DDI**	**従来の DDI**
LUT 数（シフトレジスタ数）	8,230 (6,000)	11,961 (6,067)	3333 (0)
フリップフロップ数	2,170	4,039	278
LUT 使用率	38.2%	55.6%	15.5%

注：シフトレジスタは内数である．また，本研究で用いた FPGA の LUT 数は 21,504 個である．

4.5.4　より複雑な実応用問題への対応

本実験ではソナースペクトル認識という現実のパターン認識問題を 2 カテゴリー，4 ビット精度で扱った．ただし，以下の理由によりカテゴリー数やビット精度の増加にも対応できると考えられる．

高いビット精度で DP-DDI を実現する場合，例えば 6 ビット精度であれば，図 6 より約 7 倍の LUT (SRLC16E) が必要となるが，本実験で使用した FPGA の 7 倍以上の LUT をもった FPGA (XC4VLX200 など) は既に市場に存在し，今後は更に大規模な FPGA が普及すると考えられる．そのため，高精度化に十分対応できると考えられる．また，我々は 40 個のカテゴリーをもつ顔画像認識アプリケーションを既に構築しており [13]，DP-DDI を用いてもこのアプリケーションを実装することは可能である．よって，DP-DDI は 2 カテゴリー以上のパターン認識問題にも対応することが可能である．よって，DP-DDI はビット精度，カテゴリー数の増加に対して十分対応可能であると考えられる．

4.5.5　DP-DDI の認識処理速度

2. で述べたとおり，DDI 及び，DP-DDI は並列性を最大限に利用するため，非常に高速な認識処理が可

表 9 DDI (DP-DDI) と PC の認識処理速度の比較
Table 9 Each system's recognition speed.

	顔画像認識	ソナースペクトル認識
サンプルデータ数	360	100
PC (CPU：2.66 GHz)	221 マイクロ秒	51 マイクロ秒
DDI (DP-DDI)	40 ナノ秒	40 ナノ秒
速度比	5,000 倍	1250 倍

能である．本論文の焦点は認識処理速度の向上ではないが，既に DDI の認識処理速度を報告 [12], [13] してから数年が経つため，改めて DDI (DP-DDI) の処理速度と近年の PC の処理速度を二つのアプリケーション（顔画像 [13]，ソナースペクトル）を用いて比較した結果を表 9 に示す．PC (Core2 Duo 2.66 GHz) を用いた場合，顔画像認識では 221 マイクロ秒，ソナースペクトル認識では 51 マイクロ秒の処理時間が必要であった．一方，DDI を用いれば，両アプリケーションとも 40 ナノ秒で認識処理を行うことができた．PC を用いたときに処理時間に差が出る理由は，テストデータと各サンプルデータを一つずつ順次比較していくため，サンプルデータの数に比例して処理時間が増大するからである．一方，DDI では並列性が非常に高いため，サンプルデータの数が増えても同じ処理速度を保つことが可能である．

この結果，DDI を用いた場合，顔画像認識では約 5,000 倍，ソナースペクトル認識では約 1,250 倍の認識速度となることが分かった．PC の性能の向上にもかかわらず，この速度比は以前の結果 [12], [13] よりも 2 倍以上速い．この理由は，以前の研究では，FPGA の集積度の低さから DDI を複数の FPGA に分割実装していたが，本研究では，一つの FPGA に全回路を実装しているためである．

5. むすび

既に提案している DDI に動的部分再構成の機能を加えた DP-DDI を提案し，従来の DDI の問題点を解決した．ソナースペクトル問題を対象とした DP-DDI のプロトタイプシステムを構築，評価した．その結果，新しいデータ（サンプルパターン）が入力された際，約 12 秒で回路再構成を完了できることが分かった．このリコンフィギュレーションサイクルは従来の DDI よりも 30 倍以上高速である．また，動的部分再構成の適用により，再構成中も動作を中断しないノンストップシステムを実現することができた．更に，再構成中の認識精度の低下はわずかであり，かつ，再構成時間も従来の DDI より 600 倍以上高速である．DP-DDI を用いてオンチップシステムを構築することが可能であり，これにより PC を必要とする従来手法よりもはるかに低消費電力でコンパクトなシステムを構築できることが分かった．

謝辞 貴重な指摘，コメントを頂いた査読員の方々に心より感謝致します．

文献

[1] D. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.
[2] 樋口哲也，“進化型ハードウェア，” 情報処理，vol.40, no.8, pp.795–800, 1999.
[3] 邊見 均，五味隆志，“進化するハードウェア，” 信学論 (C)，vol.J84-C, no.7, pp.543–551, July 2001.
[4] 岩谷昌也，梶谷 勇，村川正宏，平尾友二，伊庭斉志，樋口哲也，“進化するハードウェアを用いたパターン認識システム，” 信学論 (D-II)，vol.J81-D-II, no.10, pp.2411–2420, Oct. 1998.
[5] J. Torresen, “Two-step incremental evolution of a digital logic gate based prosthetic hand controller,” Proc. 4th International Conference on Evolvable Systems, vol.2210 of LNCS, pp.1–13, Springer, 2001.
[6] L. Sekanina and R. Ruzuicka, “Design of the special fast reconfigurable chip using common FPGAs,” Proc. Design and Diagnostics of Electronic Circuits and Systems, pp.161–168, 2000.
[7] J. Wang, C. Piao, and C. Lee, “Implementing multi-VRC cores to evolve combinational logic circuits in parallel,” Proc. 7th International Conference on Evolvable Systems : From Biology to Hardware (ICES), LNCS 4684, pp.23–34, 2007.
[8] K. Glette, J. Torresen, and M. Yasunaga, “Online evolution for a high-speed image recognition system implemented on a Virtex-II Pro FPGA,” Proc. NASA / ESA Conference on Adaptive Hardware and Systems, pp.463–470, 2007.
[9] M. Yasunaga, T. Nakamura, and I. Yoshihara, “Evolvable sonar spectrum discrimination chip designed by genetic algorithm,” Proc. Systems, Man, and Cybernetics, vol.5, pp.585–590, 1999.
[10] K. Glette, J. Torresen, P. Kaufmann, and M. Platzner, “A comparison of evolvable hardware architectures for classification tasks,” International Conference on Evolvable Systems (ICES), pp.22–44, Springer, 2008.
[11] M. Yasunaga, T. Nakamura, I. Yoshihara, and J. Kim, “The kernel-based pattern recognition system designed by genetic algorithms,” IEICE Trans. Inf. & Syst., vol.E84-D, no.11, pp.1528–1539, Nov. 2001.
[12] 安永守利，吉原郁夫，“パターン認識用進化型ハードウェアシステムの開発—ソナースペクトル信号認識を対象として，” 信学論 (D-I)，vol.J86-D-I, no.1, pp.1–13, Jan.

2003.
[13] 安永守利，高見知規，吉原郁夫 “FPGA を用いたナノ秒オーダ画像認識ハードウェア,” 信学論（D-II），vol.J84-D-II, no.10, pp.2280–2292, Oct. 2001.
[14] K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press, 1990.
[15] G. Tufte and P. Haddow, “Biologically inspired: A rule-based self-reconfiguration of a virtex chip,” 4th International Conference on Computational Science, Lect. Notes Comput. Sci., pp.1249–1256, Springer, 2004.
[16] P. Gorman and T. Sejnowkski, “Analysis of hidden units in a layered network trained to classify sonar targets,” Neural Netw., vol.1, pp.75–89, 1988.
[17] http://www.xilinx.com/
[18] http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/neural/bench/cmu/

付　　録

GA のパラメータ

適応度関数として

$$F(C_j^i)=\frac{\sum_{J=1}^{N_i} k_j^i(\mathbf{S}_J^i-\mathbf{S}_j^i)}{1+\sum_{I\neq i}^{C_{total}}\sum_{j+1}^{N_I} K_j^i(\mathbf{S}_J^I-\mathbf{S}_j^i)} \quad (A\cdot1)$$

を用いた．ここで，式 (A·1) に対して，$f=\sum_{j=1}^{N_i} k_j^i(S_J^i-S_j^i)$，$g=\sum_{I\neq i}^{C_{total}}\sum_{J=1}^{N_I} K_j^i(S_J^I-S_j^i)$ であり，それぞれ以下の意味をもつ．f はサンプルパターン j に対する核関数の中に j と同じカテゴリー i のサンプルパターンが何個含まれるかをカウントする．同じカテゴリーのパターンを含むほど適応度関数 $F(C_j^i)$ の値は増加する．ここで，進化の度合をより精度良く評価するために核関数としてこれまでの $K_j^i(\cdot)$ をわずかに変形した $k_j^i(\cdot)$ を用いた．$k_j^i(\cdot)$ の詳細は後述する．g はサンプルパターン j に対する核関数の中に j とは異なるカテゴリー（C_{total} 個のカテゴリーのうち，j が含まれるカテゴリー i 以外のすべてのカテゴリー）のサンプルパターンが何個含まれるかをカウントする．異なるカテゴリーのパターンを多く含んでしまうほど適応度関数値は減少する．

式 (A·1) において，$k_j^i(\cdot)$ は核関数 $K_j^i(\cdot)$ をもとに定義した関数であり，次のように与えられる．

$$k_j^i(\mathbf{x})=\begin{cases}1 & : K_j^i(\mathbf{x})=1\\ f_j^i(\mathbf{x}) & : \text{otherwise}\end{cases} \quad (A\cdot2)$$

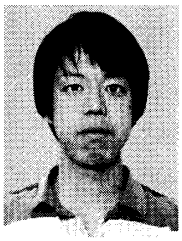
$$f_j^i(\mathbf{x})=e^{-(\alpha+cnt_j^i(\mathbf{x}))}, \alpha>0 \quad (A\cdot3)$$

$$cnt_j^i(\mathbf{x})=\sum_{k=1}^{n} unit(|x_k|-d_{jk}^i) \quad (A\cdot4)$$

$$unit(y)=\begin{cases}1 & : y>0\\ 0 & : \text{otherwise}\end{cases} \quad (A\cdot5)$$

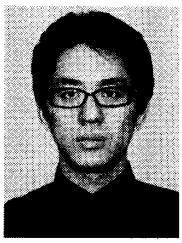
$cnt_j^i(\cdot)$ は，パターン $\mathbf{x}$ に対して各要素ごとに $|x_k|d_{jk}^i$ となった場合を数え上げる関数である．$\alpha>0$ であるため，$0<f_j^i(x)<1$ が成り立つ．$k_j^i(\cdot)$ を用いた理由は以下である．$K_j^i(\cdot)$ を用いた場合，核の中に入らなかったパターンの適応度に関する寄与は 0 になる．これに対し，$k_j^i(\cdot)$ を用いることで各要素ごとに核の中に入るか否かを評価し，$k_j^i(\cdot)$ を用いた場合に比べて進化の度合をより精度良く評価できると考える．なお，本論文では $\alpha=3$ とした．

（平成 22 年 1 月 13 日受付，5 月 10 日再受付）



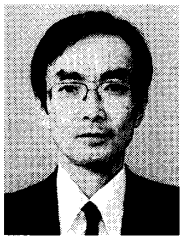
川合　浩之

平 17 筑波大・情報学類卒．平 19 同大大学院システム情報工学研究科博士前期課程了．現在，同大学院博士後期課程在学中（平 19〜20 オスロ大学客員研究員兼務）．



山口　佳樹　（正員）

平 12 筑波大学大学院理工学研究科理工学専攻修士課程了．同年日本学術振興会特別研究員．平 15 同大学院工学研究科知能機能システム専攻了．工学（博士）．同年理化学研究所横浜研究所 GSC 研究員．平 17 筑波大学大学院システム情報工学研究科講師．PLD を用いた並列計算及びその応用に関する研究に従事．IEEE，情報処理学会，計測自動制御学会各会員．



安永　守利　（正員）

昭 56 筑波大・基礎工学類・物理工学卒．昭 58 同大大学院工学研究科修士課程了．同年（株）日立製作所入社，以来同社中央研究所において大型計算機ハードウェアの研究，ウェーハスケール集積回路研究に従事．平 3〜4 米国カーネギーメロン大学客員研究員兼務．平 6 筑波大学電子・情報工学系助教授．平 16 同大学院システム情報工学研究科教授．超高速 VLSI，再構成可能計算機，進化ハードウェア，フォールトトレラントシステム，ニューラルネットワーク，遺伝的アルゴリズムの研究に従事．著書「Massively Parallel Artificial Intelligence (MIT Press)」（共著）ほか．博士（工学）．IEEE，情報処理学会会員．