DRAM混載FPGAを用いたデータ駆動型仮想ハードウェア

柴田 裕一郎[†] 宮崎 英倫[†] 高山 篤史[†]
凌 暁萍[††] 天野 英晴[†]

仮想ハードウェアWASMIIは，書換え可能なSRAM型FPGAの結線情報用メモリをマルチコンテキスト化し，データ駆動的な制御方法に基づいて回路を動的に交換することでハードウェアを仮想化するシステムである．しかしながら，SRAM型マルチコンテキストFPGAは，チップ内に多くの結線情報を持たせることができないため，外部メモリとの結線情報の通信にかかる時間がシステムのボトルネックとなる問題があった．一方，急速に進歩を続けるDRAMとロジックの混載技術によって，チップ内に混載したDRAMのカラムバッファをFPGAの結線情報用メモリに対応させることにより，大量の結線情報をチップ内に収めてこれらを高速に入れ替えることが可能となった．そこで本論文では，このようなDRAM型マルチコンテキストFPGAを用いた新たな仮想ハードウェアHOSMIIを提案し，その構成について詳しく述べ，シミュレーションを通して性能評価を行う．この結果，たとえばニューラルネットワークを用いた組合せ最適化問題では，WASMIIに比べて最大17倍の性能向上が見られた．また，高速な汎用ワークステーションと比べても，2倍から65倍の性能が達成されることが複数のアプリケーションにより示された．

A Data Driven Virtual Hardware Using an FPGA Integrated with DRAM

YUICHIRO SHIBATA,[†] HIDENORI MIYAZAKI,[†] ATSUSHI TAKAYAMA,[†]
XIAOPING LING[††] and HIDEHARU AMANO[†]

WASMII is a virtual hardware system using an SRAM based reconfigurable multi-context FPGA that virtualizes hardware with dynamic reconfiguration and a data driven control mechanism. However, the time required for loading configuration data from the off-chip memory tends to be a sever bottleneck of the performance, since the number of configuration contexts that the WASMII chip can provide is small. With recent advanced technologies of semiconductors, an FPGA and DRAM can be implemented into a single chip. Using column buffers of DRAM array as configuration memory for an FPGA, the large number of configuration contexts can be stored in the chip and switched quickly. In this paper, we present a novel virtual hardware system called HOSMII using such a DRAM type multi-context FPGA. The architecture of HOSMII is described in detail and the performance is evaluated through simulation. According to the simulation results, a combination optimization problem using neural networks can be solved on HOSMII 17 times as fast as on WASMII at maximum. It is also shown that HOSMII can achieve from 2 to 65 times better performance than that of a high-speed general purpose workstation in some practical applications.

1. はじめに

FPGA（Field Programmable Gate Array）やCPLD（Complex Programmable Logic Device）などの，ユーザが手元でプログラムすることのできるデバイスの目覚ましい技術的発展は，計算機アーキテクチャの分野にも大きなインパクトを与えている．中でもアルゴリズムを直接FPGA上のハードウェアとして実現する可変構造システムは，従来の計算機とは異なった原理に基づく計算システムとして，最近ますます広く研究されている[1)~4)]．

これらのシステムでは，対象とするアルゴリズムをそのままハードウェア化して実行することにより，従来のシステムに比べ高速に演算を行うことができる可能性があり，アプリケーション専用ハードウェアの持

† 慶應義塾大学理工学部
Faculty of Science and Technology, Keio University
†† 神奈川工科大学工学部
Faculty of Engineering, Kanagawa Institute of Technology

つ高い性能と汎用計算機の持つ柔軟性とを兼ね備えたシステムとして期待が寄せられている．しかしながら，対象の問題の規模が大きくなり，アルゴリズムの実行に必要なハードウェア量がシステムのサイズを超えてしまうと，まったく計算不可能になってしまうという問題点がある．

そこで，我々は，仮想記憶の概念を可変構造システムに応用した仮想ハードウェア WASMII[5),6)]を提案している．WASMII は，チップの内部 SRAM に複数セットの結線情報を保持できるよう拡張したマルチコンテキスト FPGA の外部に，さらにバックアップ RAM を付加し，これをデータ駆動的に制御することによりハードウェアを仮想化するシステムである．ところが，SRAM 型のマルチコンテキスト FPGA はチップ内にあまり多くの結線情報のセットを持たせることができず，チップ内外での結線情報の通信が性能を低下させてしまう恐れがあった．

一方で，最近の半導体プロセス技術の発達により，DRAM とロジックを同一ダイ上に混載することが可能となっている．そこで，本論文では WASMII の機構を DRAM 型のマルチコンテキスト FPGA に応用した新たな仮想ハードウェアシステム HOSMII[7)]を提案し，その構成と動作原理について述べ，その効果についてシミュレーションにより評価する．

2. 仮想ハードウェア WASMII

2.1 SRAM 型マルチコンテキスト FPGA

書換え可能な FPGA の利用は，ハードウェアの構造の動的な変更を可能にする．しかし，通常の FPGA ではハードウェア構成を変えるためには外部から結線情報を送り直す必要があり，このために大きな時間を要する．そこで，富士通では MPLD（Multifunction Programmable Logic Device）[8)]と呼ばれる拡張された FPGA を提案している．

MPLD は図 1 に示すように，FPGA 内部の結線情報用の SRAM を複数セットに拡張した構成を持つ．各 SRAM のセット（以降ページと呼ぶ）は単一の FPGA に必要な結線情報を保持しており，これをマルチプレクサで切り替えることによって，FPGA 上の回路を高速に入れ換えることが可能となる．

最近，Xilinx 社でも，同じ概念に基づいた時分割 FPGA を設計している[9)]．この時分割 FPGA は同社の XC4000E を拡張したものであり，約 10,000 ゲート（400 CLB）に相当する回路の結線情報をチップ内に 8 セット保持することができる．また，チップ内部での回路の交換は 30 ns で可能である．本論文では，このようなチップを SRAM 型マルチコンテキスト FPGA と呼ぶことにする．

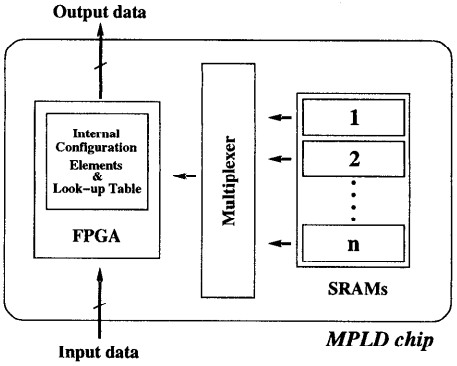


図 1 MPLD の構造
Fig. 1 Structure of MPLD.

2.2 WASMII の構成

SRAM 型マルチコンテキスト FPGA を用い，動的に回路を交換することで，FPGA の見かけの回路規模を大きくすることができる．しかし，各結線情報ページは純粋に FPGA 上の回路構成のみを決定するため，実現された回路上のレジスタの内容や順序回路の状態が，ページの切替えによって消失してしまう．このため，一般的な論理回路では結線情報ページの切替えのタイミングの制御は困難である．

この問題を解決するために，WASMII ではページ切替えの制御を含む処理全体にデータ駆動の考え方を導入した．WASMII では，まず対象とする問題をデータフローグラフに変換し，このグラフを結線情報ページ 1 つで実現可能なサイズのサブグラフに分割する．各サブグラフに対応する一連のページは，そのすべての入力アークにトークンが到着し，実行可能になった時点で，FPGA 上に実現される（以下アクティベイトと呼ぶ）．サブグラフ内の各演算ノードも完全にデータ駆動的に動作する．すなわち，それぞれのノードではすべての入力アークにデータが揃うとその演算処理を行い，結果を次のノードの入力に転送する．このとき，各演算ノード内の記憶素子は，演算中の一時的なデータを記憶するだけでよく，演算終了後はその内容を保持する必要がない．これは，静的なデータフローグラフが副作用を持たないことに対応しており，これによりページ切替え時の状態喪失の問題を解決することができる．

さらに，システムサイズに関する制限を取り除くため，仮想記憶の方式にならって結線情報用のバックアップ RAM を外部に付加し，利用していないページに対するチップ内外での結線情報の入換えを可能にし

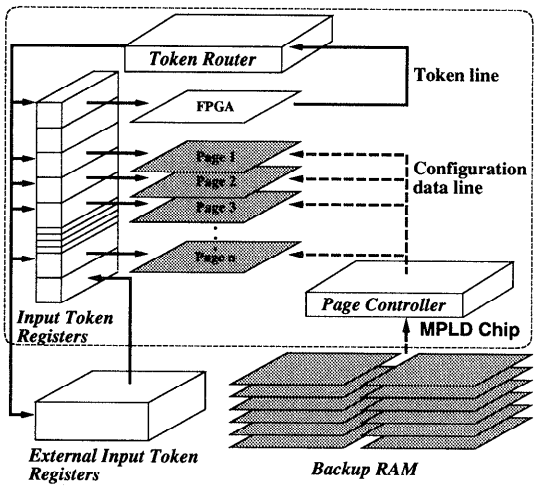


図 2 仮想ハードウェア WASMII
Fig. 2 WASMII: Virtual hardware.

ている．

仮想ハードウェア WASMII は，上記の操作を実現するために，**図 2** に示す構造を持つ．実行中のページから出力されたトークンは，トークンルータにより対応する入力トークンレジスタに送られる．ページコントローラは，この入力トークンレジスタのフラグをチェックすることにより，ページが実行可能かどうかを知り，現在実行中のページからすべてのトークンが出力された後に，新しいページをアクティベイトする．同時に複数のページが実行可能になった場合は，あらかじめ決められた優先順位に従って，その中から 1 ページを選びアクティベイトする．

3. 仮想ハードウェア HOSMII

3.1 DRAM 型マルチコンテキスト FPGA

SRAM 型マルチコンテキスト FPGA ではロジック部とそれに対応する SRAM の面積比は 3 : 1 から 4 : 1 程度である[10),11)]．このため，あまり多くのコンテキストをチップ内に保持できず，結果として WASMII ではチップ内外での結線情報の交換がシステムのボトルネックとなってしまう．

しかし，近年の半導体技術の急速な発展により，DRAM と FPGA を同一のチップ上に構成することが可能となっている．結線情報メモリを高集積可能な DRAM で実現すれば，SRAM の場合に比べて大量のコンテキストをチップ内に収めることが可能となる．また，DRAM モジュールと FPGA の論理セルモジュールを結ぶ内部バスの高いバンド幅を利用することによって，高速に回路の交換を行うことが可能となる．ここでは，このようなチップを DRAM 型マルチコンテキスト FPGA と呼ぶ．

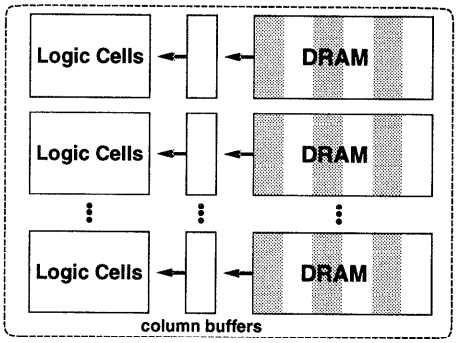


図 3 DRAM 型マルチコンテキスト FPGA
Fig. 3 A DRAM based multi-context FPGA.

図 3 に HOSMII で用いる DRAM 型マルチコンテキスト FPGA の概念図を示す．まず FPGA を一定の単位に分割した後，それぞれの部分の結線情報用メモリに DRAM モジュールのカラムバッファを対応させる．したがって，カラムアドレスを切り替えることにより，1 回のデータの読み出しで対応する FPGA の回路を書き換えることができ，高速かつ容易な回路の入換えが可能となる．また，各カラムバッファごとにデータを読み出すことで，FPGA を部分的に書き換えることもできる．

DRAM 型マルチコンテキスト FPGA の例としては，MIT の DPGA[12)]，NEC の FPGA/DRAM チップ[13)]などがあげられる．DPGA では 4 組の結線情報用 DRAM をマルチプレクサで切り換える仕組みをとっておりチップの実装も行われている．また，NEC のチップでは，0.35 μm の 3 層アルミ CMOS プロセスを使用することにより，19,000 ゲート相当の結線情報をチップ内に 256 セット分収め，これらを約 70 ns で切り換えることができるようになっている．さらに，FPGA 上に構成された回路が DRAM 領域をデータメモリとしてアクセスする機能も備えている．

3.2 チップ内並列化

従来の WASMII では，単一チップに 1 つのユニットを実装し，複数チップにより並列化を行った．しかし，チップのピン数の制限により，アクティブなページから発行されるトークンを，それぞれ行き先のノードの入力トークンレジスタに送る際に，トークンを 1 つずつ排他的に処理する必要があった．このため，データフローの考え方によって抽出したアプリケーションの並列性も，この部分の逐次的な処理によってうまく活かされない恐れがあった．

そこで，HOSMII ではチップ全体で 1 つの HOSMII ユニットを実現するのではなく，チップ内に複数の小

さなユニットを設けそれらを並列動作させる構成をとることにした．このためには，各演算ページ間のトークンのルーティングを行うトークンルータを，他のユニットにも転送が行えるように拡張する必要がある．複数のユニット間の接続には様々な方式が考えられるが，ここでは，最も単純に，隣接ユニットどうしを単方向リングで接続した構成を検討した．

図 4 にチップ内にユニットを 4 つ設けたマルチ HOSMII チップの構成例を示す．1 個の HOSMII ユニットは演算部と制御部の 2 つのブロックに分けられる．演算部では分割された対象のサブグラフがデータ駆動的に実行され，動的に再構成される．一方，制御部には，各サブグラフへの入力トークンレジスタ，他のユニットへもトークンを転送できるトークンルータ，トークンの到着状況を監視し演算部の再構成を行うページ制御回路などが構成される．

図 5 は制御部の構成を示している．制御部では，演算部のサブグラフから出力されたトークンが到着すると，まずルーティングテーブルによってそのトークンをどこに転送すべきかを調べる．一方，他のユニットから来たトークンは一度トークン FIFO に格納される．そして，クロスバスイッチ（SW）によって，ユニット内のトークンレジスタ，または他のユニットへと振り分けられる．スイッチ上で 2 つのトークン間の出線競合が起きた場合は，演算部から到着したトークンが優先され，一方のトークンはトークン FIFO で待たされる．ページ制御回路ではつねにトークンの到着状況を監視し，演算部に対してアクティベイトすべきページやタイミングについての指示を行う．

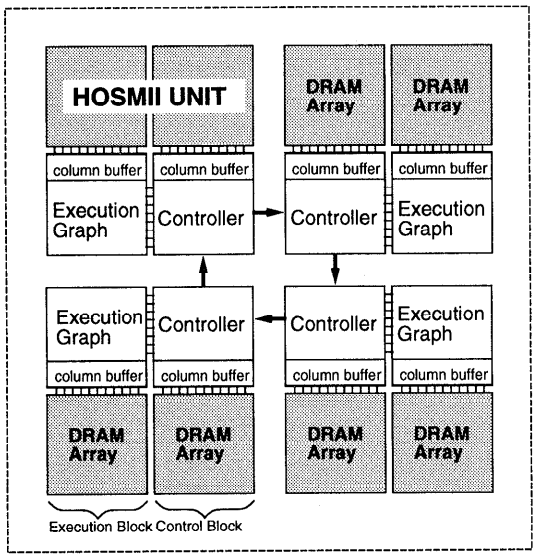


図 4 マルチ HOSMII チップの構成
Fig. 4 Structure of a multi HOSMII chip.

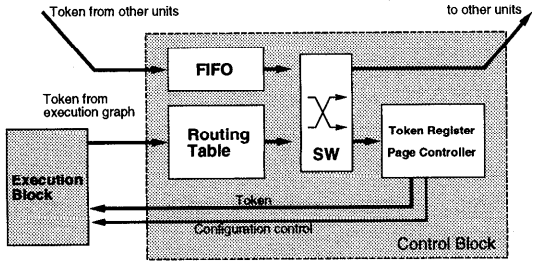


図 5 制御部の構成
Fig. 5 Architecture of a control unit.

3.3 グローバルメモリの導入

HOSMII の実行メカニズムには，WASMII の機構をそのまま用いることができる．すなわち，対象のアルゴリズムをいくつかのデータフローグラフのページに分割し，ページの起動とページ内のすべての演算を静的なデータフローメカニズムによって制御する方法である．しかしこの機構は，前述した富士通の MPLD により結線情報ページをマルチプレクサで切り換える構成の場合には，ページ切換えによってページ内の記憶素子の内容が消失してしまうことから考案されたものである．このため，WASMII ではグラフ中で記憶を行いたい場合は，これを一度ページの外に出して入力トークンレジスタに保持するという方法で実現する必要があった．

しかし，ノード内に状態を持てないという制約は，特殊な演算器構成をアプリケーションに応じて使用できるという可変構造システムの利点を少なからず制限することにつながる．また，大規模な構造体データのうち，ある一部分のみを参照したり変更を施すノードに対して，構造体の全要素をトークンとして転送するのは明らかに非効率である．そこで，NEC の FPGA/DRAM チップなどのように，FPGA の論理セルから混載された DRAM 領域へ直接アクセスさせる機構を HOSMII チップにも設け，グラフ内のノードにグローバルなメモリ空間へのアクセスを許すこととした．この機構を利用することにより，状態変数を持つようなノードもグローバルメモリにその内容をストアし，次にアクティベイトされたときにそれを再び参照することが可能となる．一方，ページの切換えに関しては WASMII と同様にデータ駆動的に行うこととした．

4. 評　　価

4.1 評価条件

以上に述べた HOSMII システムの構成法が性能に

表1 仮定したチップの諸元
Table 1 Estimated parameters.

パラメータ	WASMII	HOSMII
演算用論理セルゲート数	10,000	10,000
結線情報用SRAM/DRAM	0.5 Mb	16 Mb
ページ切り換え時間（チップ内）	30 ns	70 ns
ページ切り換え時間（チップ外）	100 μs	100 μs
コンテキスト数	8	256
トークンレジスタ（ユニットあたり）	80 KB	80 KB
トークンFIFO長	—	128

表2 評価対象システム
Table 2 Evaluated systems.

略号	アーキテクチャ	ユニット数	グローバルメモリ
WA	WASMII	1	なし
HO1	HOSMII	1	なし
HO1m	HOSMII	1	あり
HO2	HOSMII	2	なし
HO2m	HOSMII	2	あり
HO4	HOSMII	4	なし
HO4m	HOSMII	4	あり

与える影響を考察するために，シミュレーションによる評価を行った．評価に際して仮定したWASMIIおよびHOSMIIチップの諸元を**表1**に示す．これらの値は，WASMIIチップに関してはXilinx社の時分割FPGA[9]を，HOSMIIチップに関してはNECのDRAM/FPGAチップ[13]を，それぞれ参考に見積もった．また，評価対象として**表2**に示す7種類のシステムを選んだ．

なお，チップ内を複数のユニットに分割する際の結合方法は，52 bit幅の単方向リングとした．アプリケーションにより演算のデータ幅が異なるため，トークンの構成もアプリケーションごとに異なるが，トークンはデータフィールドのほかに2 bitのユニット識別子と18 bitのレジスタ識別子を持つルーティング用のタグから構成される．したがって，扱うデータサイズが32 bit以下のアプリケーションでは，隣のユニットへは1クロックでトークンを転送することが可能である．

また，表2のシステムのうち，グローバルメモリを用いるHO1m，HO2m，HO4mでは，DRAM領域の半分をグローバルメモリとして用いると考え，チップ内の結線情報のコンテキスト数を128と定めた．さらに，グローバルメモリへのアクセスタイムは70 nsと設定し，同一ユニット内のノードは各々排他的に32 bitのポートを通じてメモリにアクセスするものとした．

4.2 評価用アプリケーション

評価に用いたアプリケーションは，以下に示す4種である．

- NQ（N-Queen）
  Hopfieldスタイルのニューラルネットワークを用いてN-Queen問題を解くアプリケーション[14]である．今回はQueenの数を16とし，256個のニューロンからなるネットワークのエミュレートを行った．データフローグラフは循環グラフとなり，トークンがグラフを巡回するたびに各ニューロンのエネルギーが更新され，解の状態へと収束していく．各ノードではニューロンのエネルギーを表す5 bitのデータに対して演算を行う．
- QUE（QUEuing network simulation）
  待ち行列モデルを用いて並列計算機のネットワークの性能を解析するシミュレータ[15]である．今回は解析の対象として，階層構造を持つ多段結合網であるR-Clos網[16]に64のプロセシングエレメントとメモリモジュールが結合された並列システムを取りあげた．このデータフローグラフも循環グラフとなり，トークンがグラフを1周する動作がシミュレーション上の1ステップに相当する．待ち行列の中で扱うデータ幅は18 bitである．
- EE（Edge Extraction）
  与えられた画像データにフィルタ処理を施し，画像内のオブジェクトの輪郭抽出を行うアプリケーションである．処理内容は3段階に分かれており，まず与えられたカラー画像を輝度情報に変換し，ノイズ成分を除去した後に，画像内のオブジェクトの輪郭を抽出する．入力画像データのサイズは64×64ピクセルとし，1ピクセルあたりの色情報は24 bit（各色8 bit）とした．
- FFT（Fast Fourier Transform）
  与えられた複素数列に対して，高速フーリエ変換を行うアプリケーションである．与える複素数列は要素数を2^9個とし，データタイプは実際の制御用センサ処理で一般的に用いられるA/D変換器の規格に合わせ，実数部，虚数部ともに12 bitの固定小数とした．

シミュレーションに際しては，制御機構のオーバヘッドなども詳細に検討が行えるように，対象システム全体のRTLモデルをVHDLによって記述した．また，アプリケーションに使用される演算ノードについては，RTLモデルを実際に論理合成し，Altera社のFPGAであるFLEX10K-3デバイスで配置配線を行って，ハードウェア量と動作速度を求めた．データフローグラフの分割やシステムクロックの設定は，これらの結果をもとに手動で行った．なお，論理合成にはSynopsys社のFPGA Expressを，FPGAへの配

表3 グラフ分割の結果（NQ）
Table 3 Results of graph decomposition (NQ).

システム	論理ページ数	物理ページ数	動作速度
WA/HO1	578	11	66.7 MHz
HO1m	322	10	66.7 MHz
HO2	666	19	66.7 MHz
HO2m	500	18	66.7 MHz
HO4	1,280	20	66.7 MHz
HO4m	1,024	19	66.7 MHz

表4 グラフ分割の結果（QUE）
Table 4 Results of graph decomposition (QUE).

システム	論理ページ数	物理ページ数	動作速度
WA/HO1	1,400	11	20.8 MHz
HO1m	510	5	22.7 MHz
HO2	1,336	12	20.8 MHz
HO2m	381	7	22.7 MHz
HO4	1,117	12	20.8 MHz
HO4m	383	11	22.7 MHz

表5 グラフ分割の結果（EE）
Table 5 Results of graph decomposition (EE).

システム	論理ページ数	物理ページ数	動作速度
WA/HO1	8,127	9	50.0 MHz
HO2	8,127	11	50.0 MHz
HO4	8,622	15	50.0 MHz

表6 グラフ分割の結果（FFT）
Table 6 Results of graph decomposition (FFT).

システム	論理ページ数	物理ページ数	動作速度
WA/HO1	4,608	2	19.2 MHz
HO2	5,760	5	19.2 MHz
HO4	10,368	6	21.3 MHz

置配線には Altera 社の MAX+PLUS II を，最終的な HDL シミュレーションには Mentor Graphics 社の QuickHDL をそれぞれ用いた．

各アプリケーションのデータフローグラフを分割した結果を**表3**から**表6**に示す．分割後に同じ構造を持ったサブグラフが複数あった場合には，これらの結線情報を共用して再利用することで，結線情報用メモリの容量やページ入替え時間を節約することができる．ここでは，分割されたサブグラフの数を論理ページ数，実際に必要な結線情報の数を物理ページ数として記した．また，動作速度はアプリケーションで使用するすべてのノードのうち，最もクリティカルパスの長いノードを基準に設定した．なお，EE と FFT では，内部に状態変数を持つようなノードは使用していないため，グローバルメモリを設けたシステムでも，グラフ分割の結果は変わらない．また，**表7**には，各アプリケーションによって使用される入力トークンレジスタの量を示した．今回のアプリケーションに関しては，

表7 入力トークンレジスタ使用量（KB）
Table 7 Utilization of input token registers (KB).

システム	NQ	QUE	EE	FFT
WA/HO1	2.8	15.3	67.0	67.5
HO1m	2.0	7.8	67.0	67.5
HO2	1.4	16.1	35.5	47.3
HO2m	1.1	5.8	35.5	47.3
HO4	0.9	11.9	25.6	70.9
HO4m	0.8	3.9	25.6	70.9

仮定した入力トークンの容量に収まる値となった．

これらのグラフは概して規則性が高く，論理ページが1,000を超えるような場合にも，物理ページはたかだか20程度であった．特に FFT は，入力されたデータに対してバタフライ演算と呼ばれる特定の処理を繰り返し行うきわめて規則性の高いアルゴリズムであるため，必要な物理ページの数は WASMII でもチップ内に収まる値となった．したがって，このアプリケーションでは，DRAM 型マルチコンテキスト FPGA の使用によって導入されるチップ内部でのページ入替え時間の増加の影響を評価することができる．

4.3 実行時間とページ交換時間

以上の条件でシミュレーションを行い，各アプリケーションの実行時間を評価した．**図6**から**図9**にその結果を示す．NQ および QUE については，1 イタレーションあたりの実行時間を，その他についてはアプリケーション全体の実行時間をそれぞれ示している．また，この実行時間のうち，演算に要する時間（exec）とチップ内でのページの入替えの時間（hit），およびチップ外とのページ入替えにかかる時間（miss）の内訳も示した．さらに，**表8**に，各アプリケーションのページヒット率を示した．ここでページヒット率とは，ページを再利用せずに交換した回数に占めるチップ内でのページ交換の回数の割合をさす．

これらの結果からも明らかなように，WASMII では，チップの結線情報のコンテキスト数を上回る物理ページ数を扱った NQ と QUE では，全演算時間のそれぞれ 82%と 57%がチップ外とのページ交換に費やされており，これがシステムの性能を大きく制限している．一方で，HOSMII では結線情報用メモリに容量の大きな DRAM を用いることによって，大幅にページ入替えにかかる時間を短縮することに成功している．また，実行中に外部へのページミスを1度しか起こさなかった EE においても，これが取り除かれたことにより 4.6%性能が改善された．また，WASMII でもすべてのページがチップ内に収まった FFT で比較すると，HOSMII において結線情報用メモリにアクセスタイムの遅い DRAM を用いたことによるオーバヘッド

表8 ページヒット率
Table 8 Page hit rate.

システム	NQ	QUE	EE	FFT
WA	77.7% (87/112)	73.9% (51/69)	91.7% (12/11)	100.0% (18/18)
HO1	100.0% (100/100)	100.0% (141/141)	100.0% (13/13)	100.0% (18/18)
HO1m	100.0% (137/137)	100.0% (30/30)	100.0% (13/13)	100.0% (18/18)
HO2	100.0% (323/323)	100.0% (163/163)	100.0% (11/11)	100.0% (45/45)
HO2m	100.0% (401/401)	100.0% (163/163)	100.0% (11/11)	100.0% (45/45)
HO4	100.0% (69/69)	100.0% (222/222)	100.0% (15/15)	100.0% (63/63)
HO4m	100.0% (93/93)	100.0% (590/590)	100.0% (15/15)	100.0% (63/63)

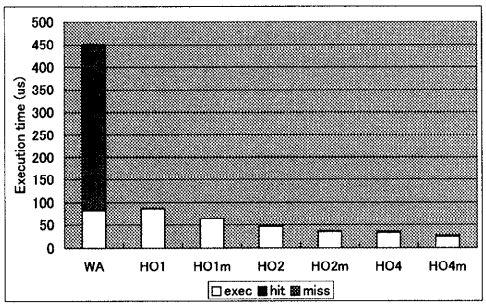


図6 イタレーションあたりの実行時間 (NQ)
Fig. 6 Execution time per iteration (NQ).

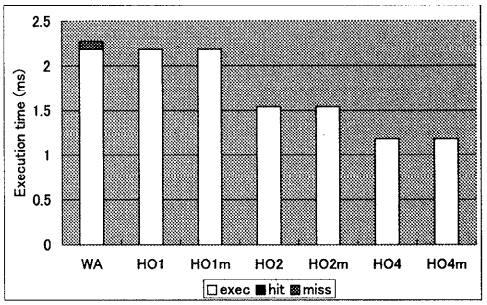


図8 実行時間 (EE)
Fig. 8 Execution time (EE).

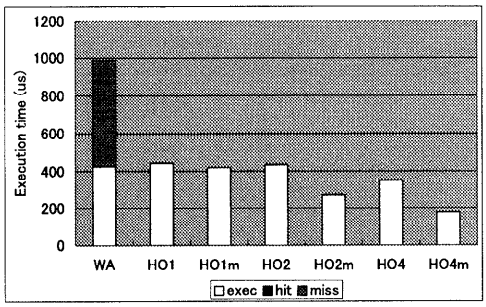


図7 イタレーションあたりの実行時間 (QUE)
Fig. 7 Execution time per iteration (QUE).

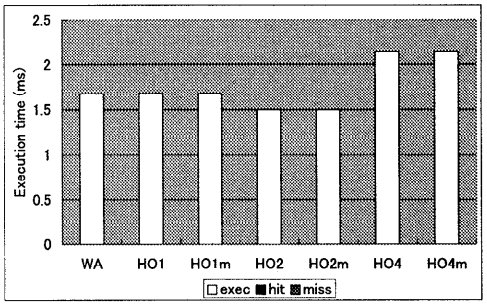


図9 実行時間 (FFT)
Fig. 9 Execution time (FFT).

は，わずかに0.06%であった．

さらに，演算に使用する論理セル数は等しいにもかかわらず，チップ内を複数のユニットに分割して並列に処理することによって性能が向上したことが分かる．これは，演算グラフから出力されたトークンを，目的のページの入力トークンレジスタに書き込む処理が並列化された効果や，グラフ内の演算レベルでの並列性に加えて，さらに粒度の粗いレベルでの並列性についても抽出できたためだと考えられる．ただし，FFTではチップ内ユニット数が4になると，かえって性能が低下している．これは，バタフライ演算に用いる複雑なノードがユニット内に入りきらず，より小さな複数のノードに分割して割り当てたためである．また，NQとQUEでは，グローバルメモリの導入により，さらに性能が向上している．これは，状態を保持するためにグラフ外に転送していたトークンが削減され，制御部内のレジスタへトークンを書き込む時間が短縮されたためだと考えられる．これらの方法を総合的に用いた結果，WASMIIに比べて，たとえばNQでは5倍から17倍，QUEでは2倍から5倍の性能改善を達成している．

4.4 通信のオーバヘッド

本節では，さらにシミュレーションの結果を詳しく分析し，特にチップ内並列化における通信制御機構のもたらすオーバヘッドについて考察する．一般に並列システムでは，ユニット間の通信性能がシステムの性

表9 ユニット間トークン転送の発生割合
Table 9 Incidence ratio of inter-unit token transmission.

システム	NQ (%)	QUE (%)	EE (%)	FFT (%)
HO2	10.4	0.1	2.1	2.2
HO2m	14.1	0.2	2.1	2.2
HO4	15.5	0.3	3.8	1.5
HO4m	19.4	0.5	3.8	1.5

能を支配する傾向が強い．マルチ HOSMII チップにおいても，通信制御機構の性能が十分でない場合，ユニットがストール状態に陥る可能性がある．ここで，ストールとは，もはやアクティベイトすべきページがなく，演算を行うことのできないまま他のユニットからのトークンの到着を待つ状態をいう．

この問題を議論するために，まず，ユニット間を接続するリングネットワークに対してどの程度のトークン転送が行われているのかを明らかにする．**表9** は，演算部から発行されるトークンのうち，ユニット間ネットワークを経由して他のユニットに転送されるトークンの割合を示したものである．最もユニット間通信の多かった NQ では，全体の 1 割から 2 割のトークンがリングネットワークに流れていることが分かる．また，**図10** には，アプリケーションの実行中にトークンがネットワークを利用する時間の割合を示す．比較的利用率の高かった NQ と QUE においても，実行時間全体で見たネットワークへの負荷は 5%から 15%程度であった．

次に，ユニット間を転送されるすべてのトークンの履歴をとり，トークンが演算グラフを出発した後，目的のユニットの入力トークンレジスタに到着するまでの平均レイテンシを評価した．この結果を**図11** に示す．転送のレイテンシは，ユニット数が 2 の場合は，最も平均の高かった EE においても 12.7 クロック程度であった．一方，ユニット数が 4 の場合は，NQ と EE で平均レイテンシが 35 クロックを超え，この図には示していないが，EE ではトークン転送の最高のレイテンシは 600 クロックを超えた．

このような高いレイテンシが発生する原因としては，主として制御部内のクロスバスイッチでトークンが出線競合を起し，トークン FIFO 内に待ち行列ができることがあげられる．トークン待ち行列のアプリケーション実行中における最大長を**図12** に示す．トークン転送のレイテンシの高かったユニット数を 4 とした場合の NQ と EE では，60 から 90 と大きな値となっていることが分かる．一方で，図示はしていないが，アプリケーション実行中の待ち行列の平均の長さは，NQ で 0.2 から 2.9，EE でも 0.1 から 0.7 程度であった．これらの結果より，全体的にはユニット間接続ネットワークの転送能力にはまだ余裕があるものの，一時的にはかなり高い負荷がネットワークにかかることが分かった．

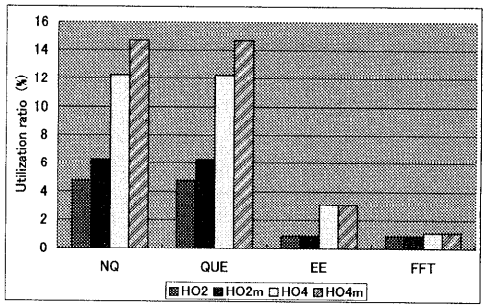


図10 ユニット間接続ネットワークの利用率
Fig. 10 Utilization ratio of inter-unit connection networks.

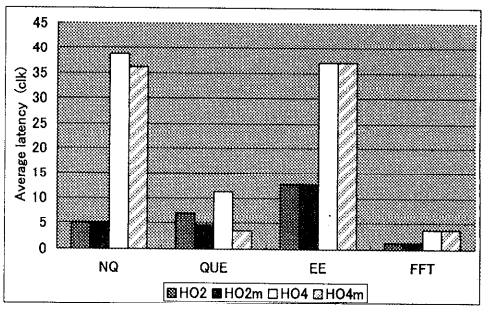


図11 トークン転送の平均レイテンシ
Fig. 11 Average latency of token transmission.

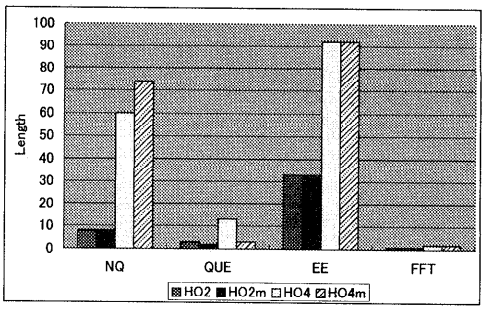


図12 トークン待ち行列の最大長
Fig. 12 Maximum length of token queues.

ここで，これらのネットワークへの負荷がどの程度性能へ影響するかを考察するために，**表10** から**表13** に各システムでの処理時間の内訳を示す．表中，“Exec” はページ内で演算を実行する時間の割合，“Miss” はチップ外部からのページ読み込みの時間の割合，“Hit”

はチップ内部でのページ交換の時間の割合，“Token”はアクティベイトしたページへのトークン供給にかかる時間の割合，“Stall”はストールの状態にある時間の割合をそれぞれ示している．

これらの表から，ほとんどの場合で実際にはストール状態は発生せず，また，発生した場合でもそれが総演算時間に占める割合は1%前後にすぎないことが分かる．これは，ページのアクティベイトがデータ駆動的に行われるため，他のユニットから転送されるいくつかのトークンの到着が遅延された場合でも，データの依存関係のないページが先にアクティベイトされるためである．この結果，比較的転送能力の低い単方向リングによる結合法を用いた場合でも，トークンのユニット間の転送レイテンシは隠蔽され，いずれのアプリケーションでも77%を超える演算ページの稼働率が達成されていることが分かる．

ただし，チップ内のユニット数をさらに増加させた場合は，今回の評価で仮定した単方向リング型のネットワークでは，トークンの転送要求量にネットワークのバンド幅が飽和し性能を低下させる恐れもある．現在の水準では，ユニットの粒度をこれ以上細かくしてユニット数を増加させることはあまり現実的ではないが，将来さらに半導体の集積度が向上した場合には，メッシュやトーラスなどの転送バンド幅のより大きな接続形式を考慮する必要がある．

表10 処理時間の内訳（NQ）
Table 10 Breakdown of the operation time (NQ).

システム	Exec (%)	Miss (%)	Hit (%)	Token (%)	Stall (%)
WA	16.5	81.5	0.1	1.8	0.0
HO1	89.1	0.0	0.9	10.0	0.0
HO1m	91.2	0.0	1.3	7.5	0.0
HO2	86.6	0.0	3.0	10.4	0.0
HO2m	87.7	0.0	3.8	8.5	0.0
HO4	78.2	0.0	7.3	13.0	1.5
HO4m	79.2	0.0	7.5	13.0	0.3

表11 処理時間の内訳（QUE）
Table 11 Breakdown of the operation time (QUE).

システム	Exec (%)	Miss (%)	Hit (%)	Token (%)	Stall (%)
WA	36.1	56.9	0.1	6.5	0.0
HO1	85.5	0.0	0.5	14.0	0.0
HO1m	95.5	0.0	0.1	4.4	0.0
HO2	85.7	0.0	0.5	13.8	0.0
HO2m	93.6	0.0	0.5	5.9	0.0
HO4	84.7	0.0	0.7	14.6	0.0
HO4m	88.6	0.0	1.6	9.3	0.5

表12 処理時間の内訳（EE）
Table 12 Breakdown of the operation time (EE).

システム	Exec (%)	Miss (%)	Hit (%)	Token (%)	Stall (%)
WA	88.5	4.4	0.0	7.1	0.0
HO1	92.5	0.0	0.0	7.5	0.0
HO2	89.3	0.0	0.1	10.6	0.0
HO4	85.3	0.0	0.1	14.6	0.0

表13 処理時間の内訳（FFT）
Table 13 Breakdown of the operation time (FFT).

システム	Exec (%)	Miss (%)	Hit (%)	Token (%)	Stall (%)
WA	85.7	0.0	0.0	14.3	0.0
HO1	85.6	0.0	0.1	14.3	0.0
HO2	79.7	0.0	0.3	20.0	0.0
HO4	77.0	0.0	0.3	22.7	0.0

4.5 ワークステーションとの性能比較

次に，本論文で提案した仮想ハードウェアシステムと高性能ワークステーション（プロセッサ：UltraSparc II，動作周波数：300 MHz，メモリ：2 GB，キャッシュ：2 MB，OS：SunOS 5.6，以後WSと略す）との間で行った性能比較について述べる．シミュレーションによる評価に用いた4つのアプリケーションをC言語で記述しWS上で動作させ，その実行時間を計測した．なお，今回のアプリケーションの扱う総データ量は，4.2節にも示すようにWSのキャッシュメモリサイズに対して十分小さいため，ほとんどキャッシュミスは起こしていないと考えられる．性能比較の結果を**図13**に示す．

この結果，HOSMIIチップはNQでWSの2倍以上，FFTやEEでは6倍から9倍以上となり，QUEでは最大65.1倍の性能改善がみられた．また，NQのように主として5 bit加算器などの単純で汎用的なノードから構成されるグラフの場合は，WSに対する性能改善率は低く，逆にQUEにおける待ち行列ノードな

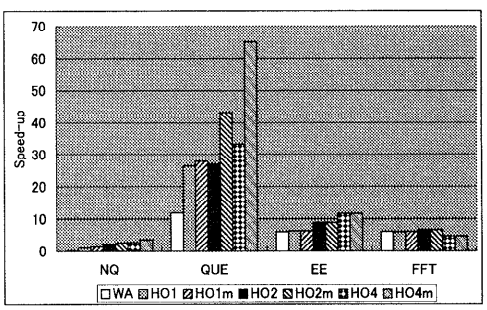


図13 WSとの性能比較
Fig. 13 Performance comparison with a WS.

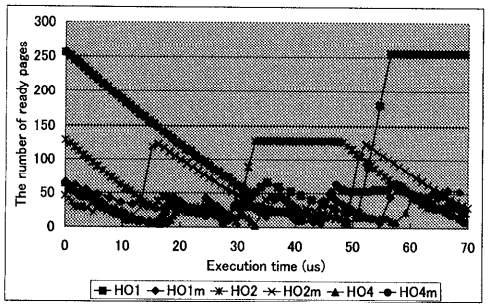


図14 実行可能ページ数（NQ）
Fig. 14 The number of ready pages (NQ).

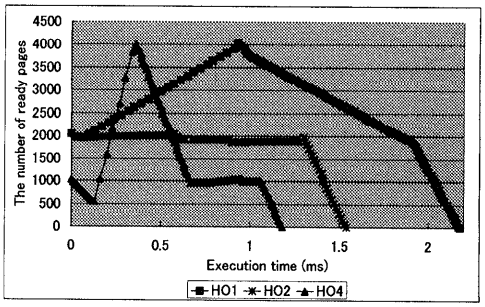


図16 実行可能ページ数（EE）
Fig. 16 The number of ready pages (EE).

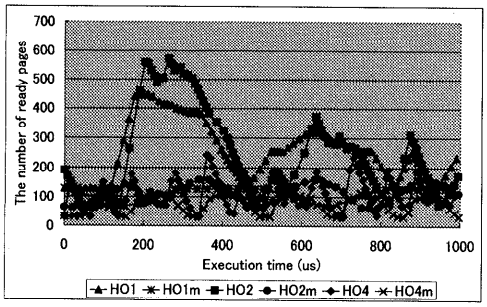


図15 実行可能ページ数（QUE）
Fig. 15 The number of ready pages (QUE).

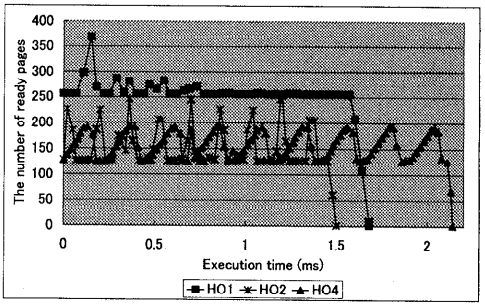


図17 実行可能ページ数（FFT）
Fig. 17 The number of ready pages (FFT).

ど専用性の高い高機能なノードを用いたグラフの場合には高い性能改善率を示す傾向があることが分かった．

4.6 実行可能ページ数

最後に，各アプリケーションの並列実行可能なページ数について考察する．アクティベイトされたページが演算を終えてすべてのトークンを出力すると，必要な入力トークンがすべて揃ったページが次にアクティベイトされるが，このときアクティベイト可能なページが複数存在することもありうる．実際にアクティベイトできるページは1ユニットあたり1ページであるから，ページコントローラが候補のページ中から1つを選択するが，ページ入替え時に選択可能であったページ数はアプリケーションの持つ並列性を示す1つの指標となる．

そこで，アプリケーションの実行中に1ユニットあたりの選択可能なページ数がどのように変化したかを調べ，**図14**から**図17**に示した．この結果，アプリケーションにも左右されるが，最も並列性の少なかったNQでチップ内ユニット数を4とした場合でも，平均20ページ以上が実行可能状態であったことが分かる．また，他のアプリケーションにおいても，平均で100から200以上のページが同時実行可能であった．

以上の結果は，データ駆動型の制御がトークン転送のレイテンシを隠蔽するという議論を裏付けるだけでなく，将来予想されるチップ面積の増加によるチップ内ユニット増設や，HOSMIIチップを要素とした並列システムの構築によって，さらに性能が改善される余地があることを示唆している．

5. 関連研究

本文中ですでに言及したもの以外にも，FPGAを動的に再構成するシステムの研究は幅広く行われている．Brigham Young大学のRRANNプロジェクト[17]では，バックプロパゲーションアルゴリズムに基づくニューラルネットワークシステムを，動的に再構成可能なFPGAプラットフォーム上に実装している．ここではマルチコンテキストFPGAの概念は取り入れられていないが，回路書換えの際に回路の差分だけを再構成するpartial configurationの手法を用いて回路書換え時間の短縮化を図っている．

UCLAでは移動体間画像通信に必要となる画像符号化処理を行うcodecを開発している[18]．このcodecでは通常のFPGAを用いているため，この再構成には約1 msかかるが，フルレートの動画でも毎秒30フ

レームの処理であるため，そのオーバヘッドは全体の約10%に収まっている．

以上のプロジェクトは，アプリケーションに特化したシステムであり，回路の切換え順序も一意に定まっている．一方，FPGAの動的な再構成技術をより汎用的なシステムに応用した例としては，Brigham Young大学のDISC（Dynamic Instruction Set Computer）[19]があげられる．DISCでは，ユーザは所望のカスタム命令のハードウェアモジュールを設計し，それらをC言語上から関数呼び出しの形で実行することができる．各命令モジュールはFPGA上に実現されるが，これを動的かつ部分的に再構成することにより，あたかも命令モジュールのキャッシュのように使用する．

また，九州工業大学で提案しているS-Machine[2]はHOSMIIと同じようにデータフローグラフをページに分割してそれらを切り換える方法を採用しているが，ページの切換え制御には，データ駆動ではなくスレッドスケジューリングの方法を適用している点で異なっている．

6. おわりに

本論文では，DRAM型マルチコンテキストFPGAを用いた仮想ハードウェアHOSMIIを提案し，その構成法についてシミュレーションによる性能評価を通して検討した．これらのシステムのソフトウェア環境としては，データフロー言語からハードウェア記述言語を生成するWASMIIコンパイラ[20]がすでに実装されているが，言語レベルでは特殊な演算ノードやグローバルメモリなどの取扱いには対応していない．今後，これらを取り入れた新たなプログラミングモデルを提供し，グラフの分割やユニット割当てなどの処理も自動化した統一的ソフトウェア環境の構築が課題となる．特に問題の分割法や割当て法は，トークン転送のレイテンシに大きな影響を与える可能性があり，将来の重要な課題である．

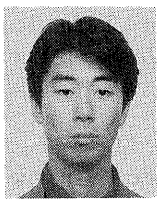
謝辞 本研究の遂行にあたり，日本電気（株）システムシリコン研究所の本村真人博士には有益なご助言をいただいた．また，使用したツールの一部はMentor Graphics社のHigher Education Program，Altera社のUniversity Programの適用を受けた．ここに深く感謝の意を表する．なお，本研究の一部は文部省科学研究費補助金（特別研究員奨励費）による．

参考文献

1) 沼 昌宏：FPGAを利用したアーキテクチャとシステム設計，情報処理，Vol.35, No.6, pp.511–518 (1994).
2) 末吉敏則：Reconfigurable Computing Systemの現状と課題—Computer Evolutionへ向けて，信学技報，Vol.96, No.426, pp.111–118 (1996).
3) Miyazaki, T.: Reconfigurable Systems: A Survey, *Proc. Asia and South Pacific Design Automation Conference*, pp.447–452 (1998).
4) Amano, H. and Shibata, Y.: Reconfigurable Systems: Activities in Asia and South Pacific, *Proc. Asia and South Pacific Design Automation Conference*, pp.453–457 (1998).
5) Ling, X. and Amano, H.: WASMII: A Data Driven Computer on a Virtual Hardware, *Proc. Intl. Conf. on FPGAs for Custom Computing Machines*, pp.33 42 (1993).
6) Ling, X. and Amano, H.: WASMII: An MPLD with Data-Driven Control on a Virtual Hardware, *J. Supercomputing*, Vol.9, No.3, pp.255–276 (1995).
7) 柴田裕一郎，宮崎英倫，高山篤史，凌 暁萍，天野英晴：DRAM混載型仮想ハードウェアHOSMIIの構成と性能，並列処理シンポジウムJSPP'98論文集，pp.303–310 (1998).
8) 吉見昌久：マルチファンクションプログラマブルロジックデバイス，公開特許公報(a)，平2-130023 (1990).
9) Trimberger, S., Carberry, D., Johnson, A. and Wong, J.: A Time-Multiplexed FPGA, *Proc. Intl. Conf. on FPGAs for Custom Computing Machines*, pp.22–28 (1997).
10) Albaharna, O., Cheung, P. and Clarke, T.: Area & Time Limitations of FPGA-based Virtual Hardware, *Proc. Intl. Conf. on Computer Design*, pp.184–189 (1994).
11) Faura, J., Moreno, J., Aguirre, M., Duong, P. and Insenser, J.: Multicontext Dynamic Reconfiguration and Real-Time Probing on a Novel Mixed Signal Programmable Device with On-Chip Microprocessor, *Proc. Intl. Workshop on Field-Programmable Logic and Applications*, pp.1–10 (1997).
12) DeHon, A.: Dynamically Programmable Gate Arrays: A Step Toward Increased Computational Density, *Proc. Canadian Workshop on Field Programmable Devices* (1996).
13) Motomura, M., Aimoto, Y., Shibayama, A., Yabe, Y. and Yamashita, M.: An Embedded DRAM-FPGA Chip with Instantaneous Logic Reconfiguration, *Proc. Symposium on VLSI Circuits* (1997).
14) Takefuji, Y.: *Neural Network Parallel Computing*, Kluwer Academic Publishers (1992).
15) 山本 欣，柴田裕一郎，天野英晴：可変構造を持つ，並列システム解析用確率モデルシミュレー

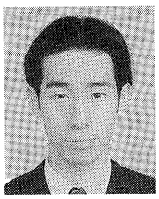
ションシステム，並列処理シンポジウム JSPP '98 論文集，pp.295–302 (1998).
16) Morimura, T., Iwai, K. and Amano, H.: Multi-stage Interconnection Network Recursive-Clos (R-Clos): Emulating the hierarchical multi-bus, *Proc. Intl. Conf. on Parallel and Distributed Computing and Systems*, pp.99–104 (1998).
17) Eldredge, J. and Hutchings, B.: Run-Time Reconfiguration: A Method for Enhancing the Functional Density of SRAM-based FPGAs, *J. VLSI Signal Processing*, Vol.12, pp.76–86 (1996).
18) Villasenor, J., Jones, C. and Schoner, B.: Video Communications using Rapidly Reconfigurable Hardware, *IEEE Trans. Circuits and Systems for Video Technology*, Vol.5, pp.565–567 (1995).
19) Wirthlin, M. and IIutchings, B.: DISC: The dynamic instruction set computer, *Proc. Intl. Conf. on FPGA for Fast Board Development and Reconfigurable Computing* (1995).
20) 日暮浩一，宮崎英倫，柴田裕一郎，天野英晴：仮想ハードウェア WASMII のためのデータフローコンパイラの研究，信学技報，Vol.97, No.27, pp.65–72 (1997).

(平成 10 年 9 月 2 日受付)
(平成 11 年 2 月 8 日採録)



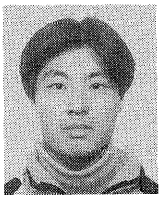
柴田裕一郎（学生会員）

平成 8 年慶應義塾大学理工学部電気工学科卒業．平成 10 年同大大学院理工学研究科計算機科学専攻修士課程修了．現在同大学院後期博士課程．平成 10 年より日本学術振興会特別研究員．可変構造アーキテクチャの研究に従事．



宮崎　英倫

平成 9 年慶應義塾大学理工学部電気工学科卒業．平成 11 年同大大学院理工学研究科計算機科学専攻修士課程修了．同年（株）電通国際情報サービスに入社．可変構造アーキテクチャの研究に従事．

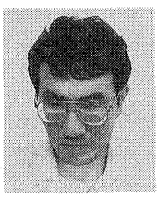


高山　篤史

平成 10 年慶應義塾大学理工学部電気工学科卒業．現在同大大学院理工学研究科計算機科学専攻修士課程．可変構造アーキテクチャのソフトウェア環境に関する研究に従事．

凌　　暁萍（正会員）

1983 年中国中山大学大気科学科卒業．1987 年慶應義塾大学大学院理工学研究科電気工学専攻修士課程修了．1993 年同大学院博士課程単位取得退学．現在神奈川工科大学情報工学科助手．工学博士．並列・分散処理，分散オブジェクト環境，最適化，ネットワーク流通，セキュリティ等の研究に従事．



天野　英晴（正会員）

昭和 56 年慶應義塾大学工学部電気工学科卒業．昭和 61 年同大大学院工学研究科電気工学専攻博士課程修了．現在慶應義塾大学理工学部情報工学科助教授．工学博士．計算機アーキテクチャの研究に従事．