データ駆動型仮想ハードウェアにおける自動ページ分割手法

柴田 裕一郎[†] 高山 篤史[†,☆]
岩井 啓輔[†] 天野 英晴[†]

データ駆動型仮想ハードウェアWASMIIは，データ駆動に基づく制御によってマルチコンテキストFPGA上の回路を入れ替えることで，ハードウェアを仮想化するシステムである．WASMIIでアプリケーションを実行する場合には，まず与えられたデータフローグラフをFPGAのサイズのページに分割する作業が必要になる．そこで，本稿はこのページ分割手法について述べる．まず，実行時に各ページがデッドロックを起こさないように分割する方法の枠組みを示し，これに並列化の抽出と通信の削減を目的とした2種類のノード割当て法を組み合わせ，PBP（Parallelism Based Partitioning）とTBP（Transfer-alleviation Based Partitioning）を提案する．そして，これらをWASMIIコンパイラに実装し，実際のアプリケーションに適用した結果をシミュレーションにより評価する．得られた結果を既存の分割法を用いた場合と比較したところ，たとえば逐次的にトークンを転送するハードウェアモデルにおいては，最大14.7%の性能向上が見られた．また，最悪ケースの性能についても，既存の手法に比べて最大18.7%の改善が認められた．

Automatic Page Partitioning Algorithms for Data Driven Virtual Hardware

Yuichiro Shibata,[†] Atsushi Takayama,[†,☆] Keisuke Iwai[†]
and Hideharu Amano[†]

WASMII is a virtual hardware system based on a multi-context FPGA with a data driven reconfiguration mechanism. In WASMII, a large scale dataflow graph which exceeds the limit of the hardware resources is divided into some subgraphs and executed on a multi-context FPGA. In this paper, automatic page partitioning algorithms for data driven virtual hardware are discussed. First, we show a basic algorithm that can partition a graph so as not to cause a deadlock. Then two node choosing algorithms, one aims to extract high degree of parallelism and the other focuses on the communication overhead, are shown to carry out PBP (Parallelism Based Partitioning) and TBP (Transfer-alleviation Based Partitioning). The proposed algorithms are implemented into a WASMII compiler and applied to practical applications for evaluation. The results of simulation show that proposed algorithms can achieve up to 14.7% performance improvement compared to existing partitioning algorithms on a sequential token transfer model. It is also shown that the worst case performance is improved by up to 18.7%.

1. はじめに

FPGA（Field Programmable Gate Array）やCPLD（Complex Programmable Logic Device）などのプログラム可能なデバイスの著しい技術的発展は，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続けている．なか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高速に実行できることから，アプリケーション専用ハードウェアの持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注目されている．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データ駆動型仮想ハードウェアWASMII[5),6)]を提案している．WASMIIはチップ内に複数セットの結線情報を保持可能なマルチコンテキスト型のFPGAに外部バックアップメモリを付加し，これをデータ駆動的に制御することによってハードウェアを仮想化す

† 慶應義塾大学理工学部
Faculty of Science and Technology, Keio University
☆ 現在，株式会社帝人システムテクノロジー
Presently with Teijin Systems Technology Ltd.

るシステムである．すでにこの仮想ハードウェアの制御メカニズムは，可変構造システムテストベッド FLEMING 上や，マルチコンテキスト構成を持つ書き替え可能デバイス DRL[7] 上に実装されており，その有効性が示されている[8),9)]．

一方，WASMII 上でアプリケーションを実行するためには，まず対象のアルゴリズムをデータフローグラフに変換した後，各々が FPGA のサイズに収まるようにページ分割を行う必要がある．このうち，アルゴリズムの記述をデータフローグラフに変換する処理については，既存のデータフロー型言語の研究成果を利用することができる．しかしながら，ページ分割については，ノードの先行性制約を考慮せずに分割するとデッドロックを生じることや，分割の仕方自体が実行性能に大きく影響することなどから，これまで自動化の手法は確立されていなかった．そこで，本稿では仮想ハードウェアにおける新たな自動ページ分割手法を提案し，これを WASMII コンパイラに実装して実際のアプリケーションに適用することにより，その性能を評価する．また，既存のグラフ分割手法を用いた場合の結果とも比較を行う．

2. 仮想ハードウェア WASMII

2.1 マルチコンテキスト FPGA

書き替え可能な FPGA を利用することにより，ハードウェアの構造を動的に変更することが可能となる．しかし，FPGA 上の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，FPGA 内部の結線情報用メモリを複数セットに拡張しマルチコンテキスト化する方法が提案されている[10),11)]．

マルチコンテキスト FPGA では，各メモリのセット（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替えることが可能となる．たとえば，NEC によって開発された DRL[7] では，チップ内の SRAM に 8 ページの結線情報を保持し，これらを最短 4.6 ns で切り替えることが可能である．また，同じく NEC により設計された DRAM 混載型のマルチコンテキスト FPGA[12] では，SRAM に比べて高集積可能な DRAM で結線情報メモリを構成し，256 ページをチップ内部に収めることが可能である．

2.2 仮想ハードウェアの構成

マルチコンテキストの機構を用いて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，任意のハードウェアに対してページ切替えのタイミングを制御することは困難である．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 1 ページ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，その入力アークにすべての必要なトークンが到着し，実行可能になった時点で，FPGA 上に実際のハードウェアとして実現される（以下，アクティベイトと呼ぶ）．また，さらに大規模なアプリケーションにも対応するために，仮想記憶の方式にならって結線情報用のバックアップメモリを外部に付加し，利用していないページに対してのチップ内外での結線情報の入替えも可能にしている．

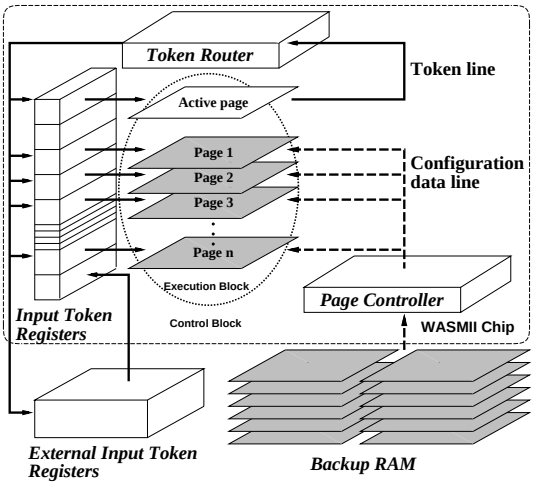


図 1 仮想ハードウェア WASMII
Fig. 1 WASMII: virtual hardware.

仮想ハードウェア WASMII は，上記の操作を実現するために，**図 1** 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って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し演算を進めていく．

WASMII では，新たにアクティベイトされるページの結線情報がチップ内部にある場合には高速な切替えが可能だが，外部のバックアップメモリから読み込む場合には通常の FPGA と同様に大きなオーバヘッ

ドを生じる．しかし，内部に大容量のページメモリを持つDRAM混載FPGAを用いれば，このオーバヘッドを大幅に削減できることが分かっている．このような構成の仮想ハードウェアを特にHOSMIII[13]と呼んでいる．

2.3 WASMIIコンパイラ

WASMII上でアプリケーションを実行するためには，ユーザの記述したアルゴリズムをデータフローグラフに変換する必要がある．この処理については，既存のデータフロー型言語の研究成果を利用することができる．現在のWASMIIコンパイラ[14]ではエントリ言語として，電子技術総合研究所でデータフローマシンSIGMA-1[15]用に開発されたCライクなデータフロー言語DFC[16]を用いている．

WASMIIコンパイラのバックエンドはフロントエンドが中間表現で生成したグラフに対し，FPGAのサイズに合わせたページ分割やページ間をまたがって転送されるトークンの入力レジスタ割当てなどの処理を行った後，各ページに対応するハードウェア記述言語（ここではVHDL）のコードを出力する．生成されたVHDLコードは一般の論理合成ツールと配置配線ツールにより結線情報へと変換される．バックエンドの一連の処理もこれまでに自動化されていたが，ページ分割についてはユーザが個別に指定する必要があった．

3. 自動ページ分割法の提案

3.1 要求事項

まず，自動分割法の詳細を述べる前に，WASMIIにおいてデータフローグラフを分割する際に，どのような事項を考慮する必要があるのかを整理する．

(1) デッドロックの回避
WASMIIでは分割された各ページがデータ駆動的に時分割で実行されるので，ノードの先行性制約を無視した分割を行ってページ間にデータ依存の循環を生じると，デッドロックを起こす．したがって，ページ分割を行う際にはデッドロックを回避する必要がある．たとえば，**図2**の分割例では，太矢印で示したように“page1”と“page2”が互いに相手の出力トークンを待つ格好になり，どちらのページもアクティベイトすることができずにデッドロックする．

(2) ページ面積制約の順守
1ページに割り当てられるノードの総面積がシステムのハードウェア規模を超えることは許されない．

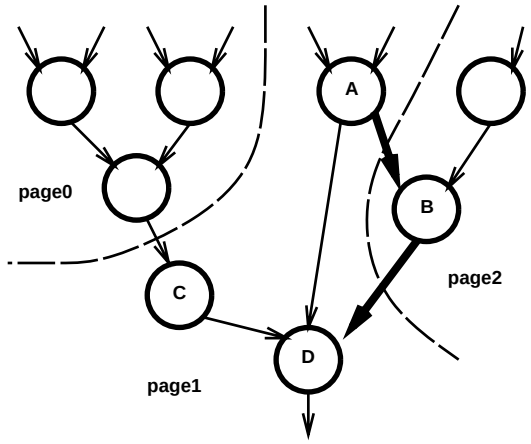


図2 グラフ分割によるデッドロックの例
Fig. 2 An example of deadlocked partitioning.

(3) ハードウェア利用率
ページ切替えによるオーバヘッドを抑制するためには，ページの面積制約の中で，できるだけ多くのノードを割り当て，分割によって生じるページの数を少なくする必要がある．

(4) 並列性の抽出
実行性能の観点からは，ページ内で演算の並列性がうまく抽出されるように，なるべく同時刻に発火可能なノードが同一のページに割り当てられることが望ましい．

(5) 入出力トークン数の削減
ページ間のトークン転送のオーバヘッドを抑えるためには，ページの境界によって切断されるエッジの数は少ないほうが望ましい．

以上の事項のうち，デッドロックの回避とページ面積制約の順守は，違反すればシステム上での実行が不可能になる制約条件である．一方，他の3つは必ずしも守る必要はないが，分割後の実行性能の向上のために考慮すべき事項である．また，一般に並列性の抽出と入出力トークン数はトレードオフの関係にあり，バランスポイントはアプリケーションの性質やトークン転送のハードウェア機構にも依存するため，最適な分割を考えることは難しい．

さまざまな制約条件や最適化条件の下でグラフを分割する手法については，他の分野でもさかんに研究が行われている．たとえばVLSI設計の分野では，規模の大きな回路を複数のFPGAやMCM（Multi Chip Module）で実現するために分割する問題が研究されている[17]．これらの手法では，デバイスのI/Oピンの本数が最も厳しい制約条件となることから，分割によって切断するエッジの数の最小化に焦点が当てられ

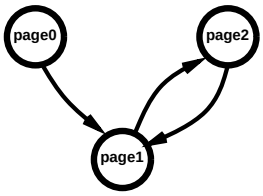


図 3 図 2 のデッドロックに対応する WFG
Fig. 3 WFG corresponding to the deadlock in Fig. 2.

ている．一方，分割された各回路は並列に動作するため，ノードの先行性制約を考慮して分割する必要はない．したがって，これらの手法をそのまま WASMII のページ分割に用いることはできない．そこで，本稿ではまずデッドロックフリーなページ分割法の枠組みを考え，そこから実行性能を向上させるための要求に対処していくことにする．

3.2 デッドロックフリーなページ分割法

デッドロックはページ間のトークンの依存関係に循環が生じることにより引き起こされる．これは，**図 3** に示すように，分割後の各ページをノード，それらの間のトークンの依存関係を有向エッジとして表した WFG（Waits-for Graph）[18]が閉路を持つことに対応する．したがって，もとのデータフローグラフにおけるノードの先行性制約を満たす順序でページに割り当ててゆけば，デッドロックを防ぐことができる．つまり，すべての直接先行ノードがいずれかのページに割り当てられているノードのみを，順次同一のページに割り当てる．そして，割り当てられたノードの総面積がページ面積制約に達した時点でその境界を確定し，以降のノードは次のページに割り当てていけばよい．この手順を以下に示す．

(1) ページ番号用カウンタ p を 0 とし，すべての入力が初期入力であるノードを割当て可能とする．

(2) 割当て可能なノード群から任意のノード v を選び，ページ p に割り当てたときのページ面積を計算する．これがページ面積制約を超える場合には p を 1 増加させる．

(3) ページ p にノード v を割り当てる．v の直接後続ノードのうち，すべての直接先行ノードがいずれかのページに割り当てられているノードを新たに割当て可能とする．

(4) 割当て可能なノードがなくなるまで (2) と (3) の手順を繰り返す．

この方法に従ってページ分割を行う限り，ページ p に属するノードは，$q \leq p$ を満足するようなページ q に属するノードへしかデータ依存を持つことはない．このため，ページ間のデータ依存は半順序の関係となり，対応する WFG に循環構造を生じることはない．

3.3 ノード選択手法

前節で示した分割手順のうち，(2) において割当て可能なノード群から実際にどのノードを選ぶかによって，分割の結果は変化し，最終的な実行性能にも影響を与えると考えられる．そこで，ノード間の並列性を抽出することによる実行性能の向上をねらった **PBP（Parallelism Based Partitioning）**と，トークン転送のオーバヘッドの削減に焦点を当てた **TBP（Transfer-alleviation Based Partitioning）**の 2 種類のノード選択手法を提案する．

3.3.1 PBP

並列性の抽出を目的とする PBP では，なるべく同時に発火可能なノードを同一ページに割り当て，グラフを水平方向に分割する必要がある．そこで，以下のような方法でノードを割り当てる．

(1) 割当て可能な各ノードについて，グラフ出口からノードまでの最長パス長を求め，これが最も長いノードを選択する．ここで，パス長とは，パス上に存在するノードの演算時間の和をいう．この値はそのノードを実行した後，残りのすべてのノードの実行を終えるまでの時間の下限値を示す．このステップの処理は，終了までの演算時間が長いノードほど先に実行されるようにすることで，総実行時間をなるべく下限値に近づけ並列性を抽出することを狙ったものである．

(2) 上記のパス長が等しいならば，直接後続ノードの数が最も多いノードを選択する．この処理は同時に実行可能なノードの数を早い段階で増やすことによって，さらに良い割当て結果を得ようとするもので，マルチプロセッサにおける優れた実行終了時間最小スケジューリング手法として知られる CP/MISF（Critical Path/Most Immediate Successors First）法[19]で用いられている手法である．

(3) 直接後続ノードの数も等しいならば，現在割り当てているページの入口からノードまでの最長パス長が最も短いものを選択する．これは，ページのパス長が延びてページの形状が縦長になることで，並列性が損なわれるのを防ぐための処理である．

3.3.2 TBP

一方，TBP では転送のオーバヘッドの削減のために，グラフをなるべく垂直方向に分割して切断エッジ数の減少を図る．

(1) 割当て可能な各ノードについてグラフ出口からの最長パス長を求め，最も短いパス長のものを選択する．グラフ出口に近いノードを優先的に割り当てることにより，ページの形状を縦長にして切断エッジ数を

表 1 評価に用いたアプリケーション
Table 1 Evaluated applications.

アプリケーション	ノード数	ページ数	ページあたりノード数	処理の内容
elliptic	34	4	8.5	Elliptic フィルタの演算[20]
iir	249	12	20.8	TI C6000 ベンチマーク集の IIR フィルタ[21]
neural	658	23	28.4	8 Queen 問題を解く Hopfield ニューラルネット[22]
epower	56	8	7.0	母線数 3 の電力潮流計算[8]
arm	88	11	8.0	6 関節ロボットアームの動的制御演算[8]

減らす効果を意図している．

(2) 上記のパス長が等しいならば，各ノードの直接後続ノードが持つ直接先行ノードのうち，現在割り当てているページに属するものの数を求め，これが最も多いノードを選択する．(1) の手順では，最初のノードの選び方によっては関連の薄いノードが同一のページに割り当てられ，切断されるエッジが増えてしまう恐れがある．そこで，ここでは同じ直接後続ノードを共有するノードを優先的に同じページに割り当てることを狙っている．

(3) それも等しいならば，現在割り当てているページの入口からの最長パス長が最も長いノードを選択する．これも (1) 同様，ページのパス長を伸ばして形状を縦長にすることによって，切断エッジ数の削減を図った処理である．

なお，PBP と TBP のいずれの場合も，3 番目の手順でも評価値が等しく選択すべきノードが一意に定まらない場合には，その中から任意のノードを選ぶものとする．

4. 性能評価

4.1 評価の手順と条件

以上に述べた分割方法が実行性能に与える影響を評価するため，WASMII コンパイラのバックエンドに PBP と TBP を実装し，実際のアプリケーションに適用して自動分割を行った．現在コンパイラのバックエンドは Perl で実装されている．この評価には**表 1** に示した 5 種類のアプリケーションを用いた．なお，表 1 には，自動分割によって生成されたページの数（今回の例では，いずれの分割法についても変わらなかった）と，ページあたりの平均ノード数（ノード数をページ数で割ったもの）も示した．また，実行性能を評価するために，コンパイラにより生成された VHDL コードを HOSMII システムの VHDL モデル[13]と合わせ，Mentor Graphics 社の HDL シミュレータ ModelSim でシミュレーションを行った．ハードウェアのパラメータは，NEC による DRAM 混載 FPGA[12]を参考に，ページサイズ 19,000 ゲート，内部ページ数 256，ページ切替えのコストを 2 クロックと仮定した．

また，3.3.1 項と 3.3.2 項で述べたそれぞれ 3 段階からなる PBP と TBP のノード選択手順の効果を評価するため，次の 4 つの分割手法についても同様にシミュレーションを行った．

PBP2 3.3.1 項の PBP の手順のうち，(2) で評価値が同一であればその中から任意のノードを選択する方法

PBP3 3.3.1 項の PBP の手順のうち，(1) で評価値が同一であればその中から任意のノードを選択する方法

TBP2 3.3.2 項の TBP の手順のうち，(2) で評価値が同一であればその中から任意のノードを選択する方法

TBP3 3.3.2 項の TBP の手順のうち，(1) で評価値が同一であればその中から任意のノードを選択する方法

分割手順の中で任意のノードを選択する場合には，分割法間の結果の比較を公平に行うため，あらかじめ乱数によって各ノード間の順序を定め，この順序に基づいて行うこととした☆．

4.2 平均切断エッジ数

ページあたりの平均切断エッジ数が，分割法によってどう変化したかを**図 4** に示す．TBP と PBP の差は neural で最も大きく，TBP は PBP の切断したエッジ数の 51%を削減した．また，両者の差の最も少なかった epower でも，PBP の 4.3 本に対して TBP は 3.9 本であり，わずかではあるが TBP の方が切断したエッジ数は少なかった．これは，PBP が演算の並列性を抽出するために，ページの形状が水平方向に長くなるように分割したのに対し，TBP がトークン転送のオーバヘッドを抑制するために，ページの幅を短くし深さを増すような形状で分割したことを示している．

たとえば，neural が他のアプリケーションと比較して，PBP と TBP による切断エッジ数の差が大幅に開

☆ なお，PBP3，TBP3 は当初 CF，DF と呼んでいた方法に対応する[23]．

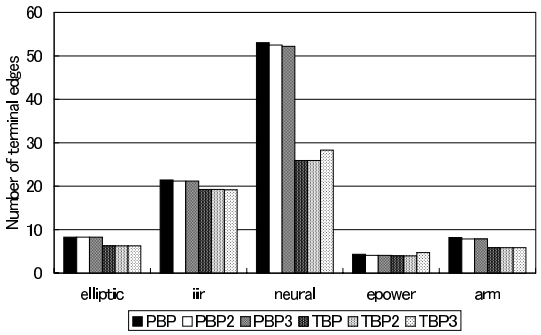


図 4 平均切断エッジ数
Fig. 4 Average number of terminal edges.

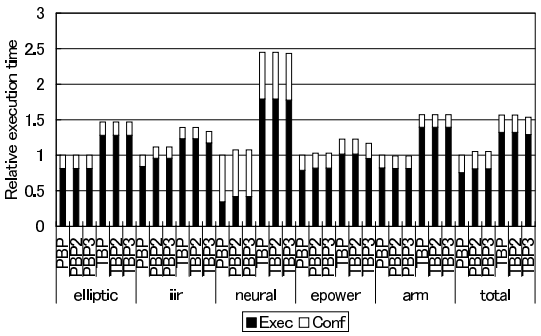


図 5 並列トークン転送モデルにおける実行時間
Fig. 5 Execution time on the parallel transfer model.

表 2 最大並列効果
Table 2 Maximum impact of parallelism.

アプリケーション	最大並列効果
elliptic	2.5
iir	5.3
neural	94.0
epower	2.1
arm	4.4

いているように，両者の差がアプリケーションによって大きく異なるのは，PBP によって切断されるエッジ数がアプリケーションの持つ並列性の上限によって抑えられるからだと考えられる．**表 2** に各アプリケーションの並列性の上限の目安として，最大並列効果を示す．ここでの最大並列効果とは，すべてのノードを逐次実行した場合の演算時間を，無限のハードウェア資源により最大の並列性のもとで実行した場合の演算時間で割ったものである．

この表より，neural は他のアプリケーションに比べて際立った並列性を持っており，その並列効果は表 1 に示したページあたりの平均ノード数よりも大きいことが分かる．このため，PBP による分割ではページ内のほとんどのノードが並列に実行され，それぞれのノードがページの境界をまたぐエッジを持つことになる．反対に TBP では縦方向にページを分割し，ページ内でノードからその後続ノードを結ぶエッジの割合が増えるため，境界切断エッジ数は減少する．一方，epower のように最大並列効果が小さいアプリケーションでは，ページ内で横方向に並べることのできるノード数は限られる．このため，PBP においても多くの後続ノードが同じページに割り当てられ，境界を切断するエッジの数も抑制される．したがって，このようなアプリケーションでは，PBP と TBP との切断エッジ数の差が生じにくい．PBP2 や PBP3 について見ると，それほど顕著な差は認められないものの，たとえば内在する並列性の高い neural では PBP に比べてそれぞれ 1%程度切断数が減っている．また，TBP と TBP2 の差は現れなかったが，neural において TBP3 は TBP に比べてページあたり 4 本程度の切断エッジ数の増加が認められた．

4.3 並列トークン転送モデルにおける実行時間

次に HDL シミュレーションの結果について考察する．まず，トークンの転送にはいっさいのコストがかからず，必要なトークンはすべて並列に供給されると仮定したモデル（並列トークン転送モデル）上での実行時間を**図 5** に示す．図中で “Exec” と示されているのは演算処理に要した時間，“Conf” と示されているのはページの切替えに要した時間であり，縦軸はそれぞれ PBP での実行時間を 1 として正規化されている．また，分割法ごとの総合的な比較の指標として，すべてのアプリケーションの実行時間の和を “total” として示している．

この並列トークン転送モデルでの実行時間は，ページ分割法によってどれだけ並列性が抽出されたのかを評価する指標となる．図 5 からも明らかなように，PBP がすべてのアプリケーションにおいて TBP に勝り，最大で 2.45 倍，平均でも 1.67 倍の性能向上を達成した．また，PBP2 と PBP3 の差はほとんどなかったが，PBP と PBP2 では最大で約 15%，平均で約 5%程度 PBP の方が良い性能を示した．なお，並列性があまり高くない arm では，逆に PBP よりも PBP2 の方が性能が向上したが，その差はわずかに 0.9%だった．

これらの結果から，3.3.1 項で述べたノード選択手順のうち，(2) の直接後続ノードの数の多いものを優先する手順の性能向上に対する寄与はあまりなく，マルチプロセッサ用のスケジューリングとはやや問題の性質が異なっていることが分かる．逆に，手順 (3) の

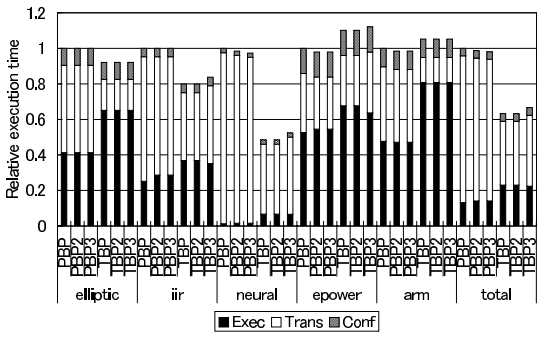


図 6 逐次トークン転送モデルにおける実行時間
Fig. 6 Execution time on the sequential transfer model.

ページ内のパス長の増加を抑える処理の効果がアプリケーションによっては顕著であることが，PBP と PBP2 の性能差として確認できた．

4.4 逐次トークン転送モデルにおける実行時間

次に，トークンの転送コストを考慮したモデルを考える．ここでは，トークンの転送は 1 クロックに 1 トークンのみが許可される逐次転送モデルを仮定する．このモデルは演算ページと制御部が別チップで構成された WASMII エミュレータ FLEMING[8]で実際に採用されている．ページの出口では，同時に 2 つ以上のトークンの出力要求がなければそのまま出力されるが，アービトレーションのためのレイテンシはつねに 1 クロック生じる．このモデル上におけるシミュレーションの結果を**図 6** に示す．

図 6 も図 5 と同様に縦軸を PBP による結果を 1 として正規化している．さらに，“Trans” として実行時間に占めるトークン転送オーバヘッドの割合も示している．本来 WASMII においては，トークンの転送と演算はオーバラップしながら処理が進んでいく．たとえば，すべてのトークンがページに入力される前であっても，すでに入力されたトークンだけで発火可能なノードは演算を開始する．したがって，トークンの転送にかかる時間と演算にかかる時間を厳密に分けて考えることは難しい．そこで，ここでは，逐次転送モデルにおける実行時間から並列転送モデルにおける実行時間を減じた時間を転送によるオーバヘッドと見なして表示している．

この図より，1 ページあたりのノード数が多く（表 1 参照），境界切断エッジ数が多いアプリケーションほど，TBP の優位性が顕著であることが分かる．たとえば iir と neural では，TBP は PBP に対してそれぞれ 1.25 倍と 2.06 倍の性能を達成している．これは，転送するトークンの数を減らした効果が，並列性を損ねることによって増加する演算時間よりも大きいあるからである．一方，ページあたりのノード数が少なく，PBP による結果でも実行時間に占める演算時間の割合が 47%を超えた epower や arm では，反対に PBP の方が TBP よりも 3%から 10%程度性能が良くなっている．たとえば，epower では TBP によって，トークン転送のオーバヘッドは自体 PBP より約 15%改善されているが，各ページのグラフの段数が増加したことにより演算時間が約 1.28 倍に増加し，これが実行性能の悪化につながっている．このことは，逐次転送モデルのようにトークン転送のコストが比較的高いハードウェアであっても，アプリケーションで使用するノードの粒度や内在する並列性によっては，命令並列性の抽出が実行性能における重要な要素となりうることを示している．

また，TBP2 と TBP3 を比べると最大で約 7%，TBP2 のほうが良い性能を示した．3.3.2 項で述べたように，TBP ではノードを深さ優先的に割り当てていくため，初段のノードの選び方によっては関連の薄いノードが同一のページに割り当てられ，切断されるエッジが増えてしまう．TBP のノード選択手順 (2) はこの問題の対処のため，同じ直接後続ノードを共有するようなノードをなるべく同一のページに割り当てることを狙ったものである．TBP3 に対する TBP2 の優位性は，この効果を示していると考えられる．一方で，TBP と TBP2 の間には実行時間に大きな差は生じなかったことから，手順 (3) におけるページ入口からのパス長によるノード選択は，性能向上に対する寄与がごくわずかであることが明らかになった．

5. 他の方法との性能比較

5.1 LBP と CBP

本節では，本稿で提案した PBP と TBP による分割結果の性能を，Gajjala Purna らが提案している Level Based Partitioning および Clustering Based Partitioning（以下，LBP および CBP と略）[24]と比較する．LBP は演算の並列性の抽出を目的とした分割法で，CBP はページ間の通信コストの抑制を目的とした分割法であり，着眼点はそれぞれ PBP および TBP のそれと近い．アルゴリズムの詳細は原典に譲るが，以下にその大まかな手順を述べる．

LBP は，まずグラフ中のすべてノードの ASAP（As Soon As Possible）レベルを計算したのち，ASAP レベルの低いものから順にページに割り当てていくアルゴリズムである．ASAP レベルはグラフの入口からそのノードに至るパスの最大の段数に相当し，すべての直接先行ノードの ASAP レベルが計算されている

とき，その中で最も高いレベルに 1 を加えたレベルを割り当てるという手順を繰り返して計算される．したがって，ASAP レベルの昇順にページを割り当てる LBP は 3.2 節で述べたデッドロックフリーなページ分割法の要件を満たしていることが分かる．

一方，CBP の手順も 3.2 節で述べた方法と同一だが，割当て可能なノード群をスタック構造で管理する点に特徴がある．すなわち，あるノードが割当て可能になったら，割当て可能ノード群を管理するスタックの先頭に push してゆく．ページに割り当てるノードを選択する際には，単純にスタックの先頭のノードを pop する．この仕組みにより，最も最近に割当て可能になったノードが優先されることとなり，分割後の各ページは段数が深く幅が短い形状になる．LBP も CBP も各ノードの演算時間は分割の際に考慮していない．

以上の LBP と CBP を WASMII コンパイラへ実装し，4.1 節と同じ条件で評価を行った．以下にその結果を示し，PBP，TBP との結果と比べて議論する．

5.2 平均切断エッジ数

図 7 は，各分割方法によって切断されるエッジ数のページあたりの平均をまとめたものである．並列性の抽出を指向した PBP と LBP は，いずれも転送オーバヘッドの抑制を指向した TBP や CBP に比べて平均切断数が多いことが分かる．しかし，PBP と LBP，および TBP と CBP の間ではアプリケーションにより結果が異なり，定まった得失の傾向は認められなかった．

5.3 並列トークン転送モデルにおける実行時間

次に並列トークン転送モデル上でのシミュレーションの結果を**図 8** に示す．縦軸は PBP による分割結果を 1 として正規化した実行時間を示す．この図からも明らかなように，並列転送モデルで PBP と LBP を比べると，すべてのアプリケーションにおいて PBP の性能が LBP のそれを上回る結果となった．また，その性能差は最大で約 19%，平均で約 13%となった．

図 7 に示した結果もあわせて比較すると，たとえば arm ではページあたりの平均エッジ数は LBP の方が約 0.6 本多いにもかかわらず，実行性能は PBP の方が約 19%向上していることが分かる．この逆転現象は iir や epower でも見られ，PBP は LBP に比べて，アプリケーションによっては，より多くの並列性を抽出しながらも転送オーバヘッドを削減できることが分かる．

この PBP の LBP に対する優位性には 2 つの要因が考えられる．第 1 は，LBP はすべてのノードの処理時間が同一と仮定した ASAP レベルを用いるため，並列性抽出の精度が PBP に比べて低いことである．第 2 は，ASAP レベルは，たかだかグラフの段数分の順序関係しかノードに与えることができないことである．このため，同一のレベルを持つ多くのノードを任意に選ぶことになり，並列性の抽出という方針が結果として曖昧になると考えられる．

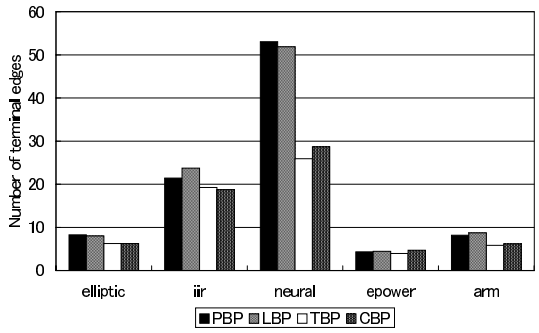


図 7 平均切断エッジ数の比較
Fig. 7 Comparison of average number of terminal edges.

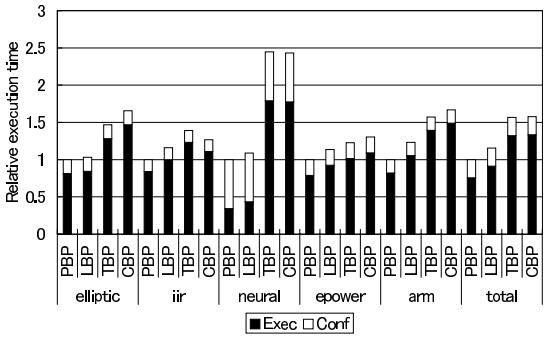


図 8 並列トークン転送モデルでの比較
Fig. 8 Comparison on the parallel transfer model.

5.4 逐次トークン転送モデルにおける実行時間

逐次トークン転送モデルにおける演算時間は**図 9** に示す結果となった．トークン転送量の削減を狙った TBP と CBP を比べると，すべてのアプリケーションで TBP が CBP に対して 3.4%から 14.7%程度の優位性を示した．図 7 の結果と比べると，たとえば iir では平均エッジ数は TBP の方がページあたり約 0.5 本多いが，転送のオーバヘッドは逆に約 15%軽減されている．純粋な演算に要する時間は TBP の方が約 11%増加しているが，全体の実行時間を評価すると，約 3.4%の性能改善となる．TBP では演算とトークン転送の重なりも効果的に引き出していることが分かる．全アプリケーションの和で見ると，TBP は CBP に対して実行時間で約 6.3%，転送オーバヘッドで約 10%の向上を示したうえ，さらに演算時間についても

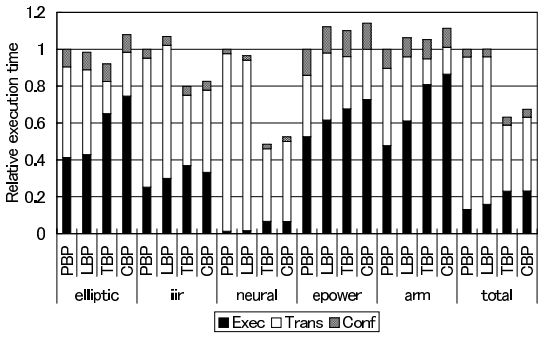


図 9 逐次トークン転送モデルでの比較
Fig. 9 Comparison on the sequential transfer model.

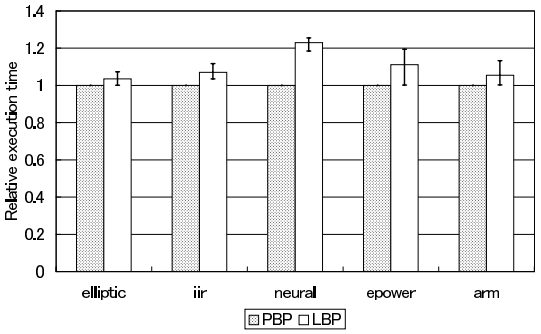


図 10 PBP と LBP の安定性（並列トークン転送モデル）
Fig. 10 Stability of PBP and LBP (parallel transfer model).

わずかながら（約 0.7%）改善を示した．

この CBP に対する TBP の優位性の要因としては，前述のように，TBP では同じ直接後続ノードを共有するようなノードをなるべく同一のページに割り当てることを狙っている点があげられる．CBP では割当て可能なノード群をスタック構造で管理するため，ひとたび割当てが始まれば任意のノードを選ぶ機会はあまりなく，ノードのページ割当てによって新たに割当て可能になったノードが複数存在した場合に，それらをスタックに push する順番に自由度が残されている程度である．それだけに，CBP では始めに初段のノードをどのような順に push するかが分割結果を左右するといえる．しかし初段のノードをスタックに積む順序を決定する評価値はなく完全に任意である．このため，関連の薄いノードを初めのページに割り当てる可能性が TBP よりも高く，良好な性能を得られにくいものと考えられる．

なお，PBP や LBP も含めて比較すると，4.4 節でも述べたようにノードの粒度が大きい epower や arm などでは，転送のオーバヘッドがある程度抑えられるため，CBP よりも LBP のほうが良い性能を示している．しかし，これらのアプリケーションでも，TBP の性能は LBP に対し epower で約 1.8%，arm で約 1.0%ほど良い．このことからも，CBP が並列性の抽出効果をうまく性能に反映できていないことがうかがえる．

5.5 分割結果の安定性

以上の考察から，割当て可能なノードを選択する際に，評価値が同じノードを任意に選ぶことにより，分割手法の方針が曖昧になり，実行性能に悪影響を及ぼす恐れのあることが指摘できる．4.1 節でも述べたように，以上に示した評価では各分割法の公平な比較のために，あらかじめ乱数によって決めておいた共通の順序に基づいて任意のノードの選択を行っていた．そこで，同じ分割法による結果が任意なノード選択の影響をどの程度受けるのかを考察するため，任意のノード選択の際に用いる乱数順序を 100 通り用意し評価を行った．

まず，PBP と LBP によるそれぞれ 100 回の分割結果を並列トークンモデルでシミュレーションした結果を**図 10** に示す．図中の棒グラフは 100 回のシミュレーションによって得られた実行時間の平均を示し，さらにマーカによって実行時間の最小値（ベストケース）と最大値（ワーストケース）もあわせて示している．また，縦軸は PBP の平均実行時間を 1 として正規化していてる．

この結果，今回用いたアプリケーションでは PBP はいずれの系列の場合にも，同じ実行時間が得られることが分かった．一方，LBP は PBP に対して演算時間の平均が 3.6%（elliptic）から 23%（neural）増加するうえ，ワーストケースではさらに平均値から最小で 2.5%（neural），最大で 8.3%（epower）の演算時間の増加が認められた．また，LBP のベストケースと PBP による分割結果を比較すると，PBP が同等（elliptic，epower，arm）もしくはそれ以上の性能を示すことが分かった．

次に TBP と CBP による 100 回の試行を逐次トークン転送モデルで評価した結果を**図 11** に示す．図の縦軸は TBP による平均演算時間を 1 として正規化されている．この図からも分かるように，逐次転送モデルでは TBP も，たとえば neural ではベストケースとワーストケースの差が平均値の 14%に達するなど，乱数系列によって実行性能に大きく影響が出ている．これは逐次転送モデルでは，同じ形状のページであっても，たとえば入力トークンの投入順序が変化するだけで出力トークンの衝突の様子が変わるなど，演算時間を変化させる要素が多いことが影響している．また，

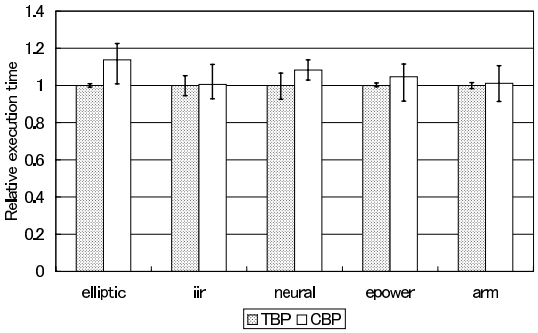


図 11 TBP と CBP の安定性（逐次トークン転送モデル）
Fig. 11 Stability of TBP and CBP (sequential transfer model).

表 3 100 回の試行のうち CBP の実行性能が上回った回数
Table 3 The number of cases in which CBP shows better result during 100 trials.

アプリケーション	CBP が上回った回数
elliptic	0
iir	43
neural	0
epower	11
arm	35

初段ノードの割当てが変化することによる影響も大きいと考えられる．一方で，elliptic や epower，arm などでは CBP に比べて実行性能の安定性の差は歴然としている．

ところが，TBP と CBP ベストケースで比較すると，iir，epower，arm の 3 つのアプリケーションでは CBP のほうが最大約 8.4%の良い性能を示した．そこで，同じ乱数系列を用いた結果どうしで TBP と CBP を比較し，100 回の試行のうち何回 CBP の結果が TBP を上回るかを評価して**表 3** にまとめた．CBP のほうが良い性能を示した回数が最も多かったのは iir で，その確率は 4 割を超えた．

しかしながら，実行時間の平均で比較すると，TBP による結果のほうがすべてのアプリケーションで CBP よりも良好であり，iir でも 0.6%ほど高い性能を示している．また，ワーストケースで比較しても，TBP はすべてのアプリケーションで CBP よりも最大 18.5%，平均 7.2%の改善を示し，iir でも 5.5%TBP の結果が優れている．これらのことを全体的に考えると，TBP は CBP よりも安定的に良好な結果を示しており，より安全な分割法であるといえる．一方で，CBP も初段ノードの割当て順序によっては，発見的に良好な性能を達成できることも示された．このことは，もし初段ノードの効果的な割当て法を，CBP に組み合わせることができれば，単純かつ強力なページ分割法を構成できる可能性があることを示唆している．

6. おわりに

本稿ではデータ駆動型仮想ハードウェアにおけるページ分割の方法について述べた．まず，デッドロックを回避可能な分割法の枠組みを示し，次に並列性の抽出とトークン転送の削減を方針とした 2 つのノード選択手法を組み合わせることによって PBP と TBP を提案した．また，これらをコンパイラに実装し，アプリケーションに適用して評価を行い，既存の方法との比較して議論した．今後の課題としては，ノードの粒度やハードウェアモデルに応じて最適な分割を行う手法や，複数の WASMII ユニットを並列接続したシステムへの対応などがあげられる．また，ページ分割の自動化の実現によって，転送のオーバヘッドなどの詳細な評価が行えるようなったが，今後はこれらの成果をアーキテクチャ側にフィードバックし，性能向上を図ってゆく必要がある．

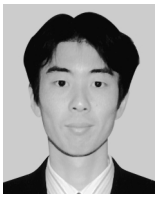
謝辞 本研究の一部は文部省科学研究費補助金（特別研究員奨励費），Mentor Graphics 社 Higher Education Program による．

参考文献

1) Vemuri, R. and Harr, R.: Configurable Computing: Technology and Applications, *IEEE Comput.*, Vol.33, No.4, pp.39–40 (2000).
2) 末吉敏則，飯田全広：リコンフィギャラブル・コンピューティング，情報処理，Vol.40, No.8, pp.777–782 (1999).
3) Miyazaki, T.: Reconfigurable Systems: A Survey, *Proc. Asia and South Pacific Design Automation Conference*, pp.447–452 (1998).
4) Amano, H., Shibata, Y. and Uno, M.: Reconfigurable Systems: New Activities in Asia, *Proc. Intl. Workshop on Field-Programmable Logic and Applications*, pp.585–594 (2000).
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) Fujii, T., Furuta, K., Motomura, M., Nomura, M., Mizuno, M., Anjo, K., Wakabayashi, K., Hirota, Y., Nakazawa, Y., Ito, H. and Yamashina, M.: A Dynamically Reconfigurable Logic Engine with a Multi-Context/Multi-Mode Unified-Cell Architecture, *Proc. Intl.*

Solid-State Circuits Conf., pp.360–361 (1999).
8) 宮崎英倫，高山篤史，柴田裕一郎，天野英晴：データ駆動型仮想ハードウェア制御機構のエミュレーションによる評価，FPGA/PLD Design Conf. & Exhibit 予稿集，pp.15–22 (1999).
9) Uno, M., Shibata, Y., Amano, H., Furuta, K., Fujii, T. and Motomura, M.: Implementation of Virtual Hardware on Dynamically Reconfigurable Logic, *Proc. World Multiconf. on Systemics, Cybernetics and Informatics*, pp.124–129 (2000).
10) 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).
11) 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).
12) Motomura, M., Aimoto, Y., Shibayama, A., Yabe, Y. and Yamashina, M.: An Embedded DRAM-FPGA Chip with Instantaneous Logic Reconfiguration, *Proc. Symposium on VLSI Circuits* (1997).
13) 柴田裕一郎，宮崎英倫，高山篤史，凌 暁萍，天野英晴：DRAM 混載 FPGA を用いたデータ駆動型仮想ハードウェア，情報処理学会論文誌，Vol.40, No.5, pp.1935–1946 (1999).
14) 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).
15) Hiraki, K., Shimada, T. and Nishida, K.: A Hardware Design of the SIGMA-1 – A Data Flow Computer for Scientific Computations, *Proc. Intl. Conf. on Parallel Processing*, pp.851–855 (1983).
16) 島田俊夫，関口 智，平木 敬：データフロー言語 DFC の設計と実現，電子情報通信学会論文誌，Vol.J71-D, No.3, pp.501–508 (1988).
17) Alpert, C. and Kahng, A.: Recent Directions in Netlist Partitioning: A Survey, *Integration: VLSI J.*, Vol.19, No.1–2, pp.1–81 (1995).
18) Bernstein, P., Hadzilacos, V. and Goodman, N.: *Concurrency Control and Recovery in Database Systems*, Addison Wesley (1987).
19) Kasahara, H. and Narita, S.: Practical Multiprocessor Scheduling Algorithms for Efficient Parallel Processing, *IEEE Trans. Comput.*, Vol.C-33, No.11, pp.1023–1029 (1984).
20) Rakhmatov, D., Vrudhula, S., Brown, T. and Nagarandal, A.: Adaptive Multiuser Online Reconfigurable Engine, *IEEE Design & Test of Computers*, Vol.17, No.1, pp.53–67 (2000).
21) Texas Instruments Semiconductors: *C6000 Benchmarks* (1997).
22) Takefuji, Y.: *Neural Network Parallel Computing*, Kluwer Academic Publishers (1992).
23) 高山篤史，柴田裕一郎，岩井啓輔，天野英晴：仮想ハードウェア WASMII システム用コンパイラバックエンドの実装と評価，並列処理シンポジウム JSPP2000 論文集，pp.285–292 (2000).
24) Gajjala Purna, K. and Bhatia, D.: Temporal Partitioning and Scheduling Data Flow Graphs for Reconfigurable Computers, *IEEE Trans. Comput.*, Vol.48, No.6, pp.579–590 (1999).

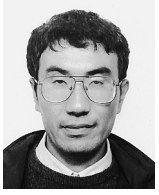
(平成 12 年 8 月 28 日受付)
(平成 13 年 3 月 9 日採録)



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 年同大学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．可変構造システムの研究に従事．

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．平成 12 年同大学大学院理工学研究科計算機科学専攻修士課程修了．現在帝人システムテクノロジー勤務．在学中は可変構造システムのソフトウェア環境の研究に従事．

岩井 啓輔
平成 8 年早稲田大学理工学部電気工学科卒業．平成 10 年慶應義塾大学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．自動並列化コンパイラの研究に従事．



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 年同大学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．