汎用エンジンと論理診断への応用

沼 昌宏[†,*] 菅沼直昭[†]
黒木修隆[†] 平野浩太郎[†]

構成情報を電気的に設定することで内部論理を変更できる複数の FPGA とメモリを相互に接続した構成によって，多様な用途に利用可能な汎用エンジンを提案する．従来 LSI を対象とする CAD の分野では，アルゴリズムの専用ハードウェア化によって高い処理性能を実現する，専用エンジンが開発されてきたが，徹底した専用化のために他の用途に利用できないという問題があった．汎用エンジンでは，FPGA の構成情報を複数用意するとともに，処理対象となるデータを格納するメモリを備えることによって，単一のハードウェアで複数の処理を実行する．比較的小規模の問題を対象とする汎用エンジンのプロトタイプとして，RM-I (Reconfigurable Machine-I) を開発した．RM-I は，5個の FPGA と 384 K バイトのメモリから構成される．FPGA 間の配線資源とメモリの利用効率が，RM-I の速度向上の鍵となる．RM-I 上に構築するアプリケーションとして，論理診断手法の一つである拡張 X-伝搬法と論理シミュレータを実現した．実験の結果，拡張 X-伝搬法の複数の処理工程において，15 MIPS の計算機上のソフトウェア処理に比べて約 35〜50 倍の高速化が達成され，汎用エンジンが性能と柔軟性のトレードオフの問題への一解決策となることが示された．

Reconfigurable Machine and Its Application to Rectification of Logic Design Errors

Masahiro Numa,[†,*] Naoaki Suganuma,[†]
Nobutaka Kuroki[†] and Kotaro Hirano[†]

This paper presents a Reconfigurable Machine (RM), capable of efficiently implementing a wide range of computationally complex algorithms. Its highly flexible architecture combining re-programmable FPGA's with RAM's supports a wide range of applications. Since its "gate-level programmability" allows us to implement various kinds of parallel processing techniques, RM provides a performance comparable to existing "special-purpose" engines for LSI CAD. The dynamic reconfiguration capability of FPGA's is used to reload several kinds of configuration data while power is applied to them. Thus, RM behaves itself like a general-purpose computer applicable to various kinds of applications by loading programs. A Reconfigurable Machine prototype, called RM-I, has been built as the first prototype incorporating five FPGA's and four SRAM memory banks. RM-I has been applied to logic diagnosis, which locates logic design errors in gate-level combinational circuits. Two kinds of locating processes have been implemented on a Logic Diagnosis Engine (LDE) employing RM-I as a hardware platform. LDE has achieved the processing speeds 35 to 50 times as fast as that on a 15 MIPS computer. The concept of RM is one of the best solution to the trade-offs between general-purpose machines and special-purpose ones.

1. はじめに

計算機が扱うデータ量の増大とともに，処理時間の増加が問題となっている．特に LSI を対象とする CAD の分野では，対象とする回路の大規模化・複雑化により，設計期間の延長を余儀なくされる場合が生じている．設計期間を短縮するためには，特定の CAD 処理を専用のハードウェア上で高速に実行する専用エンジン[1]が用いられる．特に，シミュレーション[2]や配置・配線[3]を処理の対象として，数多くの提案や開発が行われてきた．

図1にこれらの高速化手法に関する性能と柔軟性の関係を示す．これらのエンジンは，汎用の計算機によるソフトウェア処理と比較して2桁から3桁の処理速度向上を達成している．その反面，特定の処理向けに徹底した専用化が図られているために，適用可能な処理の柔軟性が欠ける点に問題がある．さらに，一般には大きな筐体を必要としており，設計者単位で専有利

† 神戸大学大学院自然科学研究科
The Graduate School of Science and Technology, Kobe University
* 旧姓 冨田
Formerly, Tomita

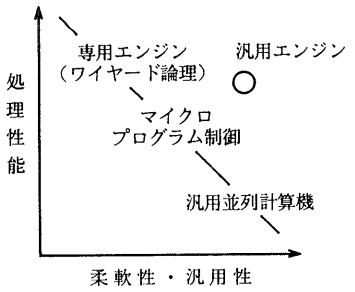


図 1 高速化手法の位置付け
Fig. 1 Performance and flexibility with acceleration techniques.

用することが困難な点も問題である．

一方，汎用の並列計算機上で，並列処理によって高速に処理を行う試み[4]もなされている．しかし，これらは比較的粒度の粗い並列性に頼らざるを得ないため，専用エンジンに比べると一般的に処理速度が低下する．また，汎用並列計算機による処理と専用エンジンの中間に位置付けられるアーキテクチャとして，マイクロプログラム制御を導入したエンジン[5]が挙げられる．これらのエンジンは，制御に関する柔軟性を確保しているが，通常は固定したデータパスをもち，確保できる柔軟性には限界がある．以上のことから，性能と柔軟性・汎用性を両立させる高速化手法は存在しておらず，これらのトレードオフを十分に考慮して，処理方式を決定する必要があった．

この問題を解決するために，我々は汎用エンジンを提案する．汎用エンジンは，複数の FPGA (Field Programmable Gate Array)[6]とメモリから構成される．FPGA は，内部論理機能をプログラム可能なゲートアレイの一種である．特に SRAM 型の FPGA を用いることで，動的な内部機能の変更を可能とする．汎用エンジンは，ワイヤード論理の再構成が可能であるため，さまざまな処理を高速化する可能性をもつ．汎用エンジンのプロトタイプとして，我々は RM-I (Reconfigurable Machine-I) を開発し，複数の処理の高速化を実現した．

2. 汎用エンジン

2.1 汎用エンジンの概念

汎用エンジンは，複数の書換え可能な FPGA とメモリを組み合わせた構成をとる．エンジンによる高速化を達成するためには，対象とする処理がもつあらゆる粒度の並列性を抽出することが必要である．汎用エンジンで用いる FPGA はこの目的に合致するばかりでなく，マイクロプログラムによるエンジンよりも高い柔軟性を実現できる．また，処理に必要なデータを記憶するためにメモリを搭載することによって，汎用エンジン単体での処理実行を可能としている．

汎用エンジンは以下の点を特徴とする．

- ゲートレベルでの柔軟性

  FPGA はゲートレベルでプログラム可能であるので，専用エンジンと同様のワイヤード論理の実現も含め，対象とする処理についてあらゆる粒度の並列性を開拓できる可能性をもっている．

- メモリの搭載

  データ格納用のメモリを用意することによって，汎用エンジン単体でのアルゴリズムの実行が可能になる．また，メモリを複数バンク構成とすることによって，メモリアクセスの並列化による処理速度の向上が期待できる．

- 内部機能の動的変更

  電気的に内部ロジックを変更できるので，電源を投入したまま処理内容を変更することができる．

汎用エンジン上での動作手順の例を**図 2** に示す．まず，ホスト・コンピュータから送られる構成情報 (Configuration Data) によって FPGA の内部機能を設定する．次に，必要に応じて汎用エンジンのメモリに初期データを設定する．汎用エンジンは，ホスト・コンピュータからの指示によって処理を開始する．結果がメモリに格納される場合は，インタフェース・モジュールを介してホスト・コンピュータに転送する．異なる処理を実行する際には，対応する別の構成情報によって FPGA を再設定する．汎用計算機ではソフトウェアがアプリケーションと呼ばれるのに対して，汎用エンジンでは構成情報によって設定されるハードウェアをアプリケーションと呼ぶ．

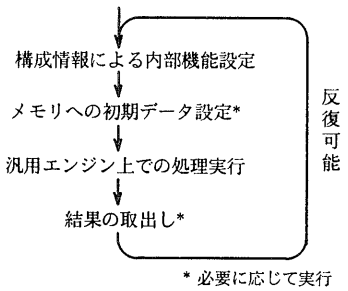


図 2 汎用エンジン上での動作手順例
Fig. 2 Process flow on reconfigurable machine.

近年，MARS III[7] や Enterprise Emulation System[8]のように，論理回路を FPGA 上に実現することで LSI 製作前にその動作を実時間でエミュレーションする，論理エミュレータの利用が広まりつつある．FPGA 上に種々の回路を実現する点では汎用エンジンと類似しているが，異なる目的で利用される．すなわち，汎用エンジンは，ある処理を対象としてその高速化のために回路を実現するのに対して，論理エミュレータでは与えられた回路を対象としてその動作を忠実に再現することを目的としている．種々の回路について自動分割を適用するために，各 FPGA の利用効率を低く抑え，結果的に多数の FPGA を搭載する高価なシステムとなっている．

汎用エンジンと同様の概念に基づいて開発されたマシンとして，Splash[9]と AnyBoard[10]が挙げられる．Splash では 32 個の FPGA を直線的に並べるリニア・アレイ構造を採用して，各 FPGA 間に隣接する FPGA からアクセス可能なメモリを配置しており，一次元のパターン・マッチング処理に適している．しかし，FPGA 間の結合形態がリニア・アレイで固定されている点と，任意のメモリへのアクセスが困難な点等によって，応用分野が限定される．もう一方の AnyBoard では 5 個の FPGA が直線的に配置され，18 ビットの共通バスで接続する．中央部の 3 個の FPGA はそれぞれ RAM のデータ線と接続するが，必要な外部端子数を節約するため，左端の FPGA から出力される共通のアドレスによってのみアクセスされる．そのため，独立したアドレスで各メモリを参照するパイプライン処理には適していない．

汎用エンジンの概念は，搭載する FPGA の数やメモリ容量，FPGA 間接続，メモリ配置，外界とのインタフェース等を規定するものではない．これらは，対象とする複数の処理がもつ特性を考慮して決定すべきである．したがって，Splash や AnyBoard も汎用エンジンに含まれるが，柔軟性の不足によってその適用範囲が限定されると考えられる．

2.2 汎用エンジン・プロトタイプ：RM-I

我々は，汎用エンジンの概念により実現される，処理の高速性と柔軟性の両立を確認するために，その一形態として比較的小規模の問題を対象とするプロトタイプ RM-I を開発した．動的に書換え可能な FPGA として Xilinx 社の XC 3090[6] を用いた．XC 3090 は 320 個の CLB (Configurable Logic Block) と 144 個の IOB (Input/Output Block) をもつ．内部には，エッジトリガ型Dフリップフロップが 928 個含まれ，2 入力 NAND ゲート換算で約 9,000 ゲート相当の回路が実現可能とされている．

表 1 RM-I の仕様
Table 1 RM-I specification.

実現可能な回路規模	約 20K ゲート
総メモリ容量	384K バイト
実行モジュールに搭載する FPGA	XC3090×4
メモリバンク数	4
メモリビット構成	32K 語×24 ビット
結 合 網	完全結合の固定配線，共通バス

RM-I の仕様を**表 1** に示す．次の点を考慮して決定した．

- ゲート規模
  これまでに開発したエンジン[11]のゲート数をもとに，20 K ゲートまでの回路を実現可能とする．
- メモリ容量
  CAD 分野の処理実行に最低限必要と考えられる 384 K バイトのメモリを用意する．
- FPGA・メモリバンク数
  対象とする処理に含まれる並列性をできるだけ抽出して 20 K ゲート規模の回路を実現するため，4 個の FPGA を対象とする処理実行のために使用する．これに対応して，メモリアクセスの並列化を行うために，メモリを各 96 K バイト (32 K 語×24 ビット) の 4 バンクに分割し，それぞれの FPGA に割り当てる．
- FPGA 間接続
  接続方式としては，クロスバスイッチ，バス，アレイ，ハイパキューブ等が考えられる．RM-I は，それ自身を単一プロセッサとしてもマルチプロセッサとしても設定できるように，それぞれの FPGA 間に個別の通信路を設ける．また，FPGA 間で共通して用いるデータのためにバスを設ける．

これらの仕様に基づいて設計された RM-I の構成を**図 3** に示す．実行モジュール，インタフェース・モジュール，構成情報インタフェースから構成される．

(1) 実行モジュール

実行モジュールには 4 個の FPGA を配し，対象となる処理を行う回路を実現する．各 FPGA 間は完全結合により接続される．ゲートレベルで設計された回路と処理対象のデータをどのように 4 個の FPGA・

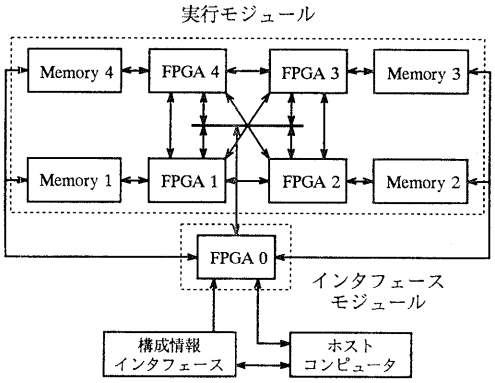


図 3 RM-I の構成
Fig. 3 RM-I architecture.

メモリに分割するかが性能向上の鍵となる．なお，FPGA 1～FPGA 4 のすべてに共通して接続する信号線 24 本をバスと呼んでいる．単純に接続するだけで特にプロトコルを定めておらず，ビット単位でデータ転送や制御など，異なる用途に利用可能である．

（2） インタフェース・モジュール

アルゴリズムの実行に先立つメモリへの初期データの設定や，処理終了後のデータの読出し等のホスト・コンピュータから RM-I へのアクセスは，インタフェース・モジュールを通して行われる．メモリ・アクセスに関して，実行モードと非実行モードの二つのモードで異なる動作を行う．実行モードでは，メモリ 1～4 に対して 4 個の対応する FPGA がそれぞれローカルにアクセスを行う．非実行モードでは，インタフェース・モジュールから与えるグローバル・アドレスによって，すべてのメモリに対してアクセスできる．

（3） 構成情報インタフェース

構成情報インタフェースは，FPGA に対して動的に構成情報を設定する機能を果たす．この部分は，FPGA の内部機能が未設定でも動作する必要性から，TTL-IC を用いて構成されている．

3. 論理診断エンジン：LDE

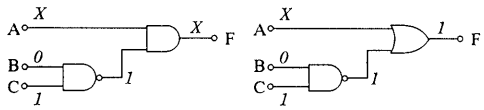
論理診断とは，ゲートレベルの回路に含まれる論理設計誤り（以下，誤り）を特定し，修正方法を提示することである．一般には人手で論理診断を行う必要があったために，設計時間を長引かせる原因となっていた．我々は，すでに論理診断を自動化する手法の一つとして，単一出力の組合せ回路に含まれる多重の誤りを対象とした，拡張X-伝搬法[12]を提案している．

我々は，拡張 X-伝搬法における複数の工程を，RM-I 上のアプリケーションとして実現した．これら，RM-I 上に構築される論理診断のためのハードウェアをまとめて，**論理診断エンジン LDE** (Logic Diagnosis Engine) と呼ぶ．LDE の処理対象とするデータの作成とその制御，さらに後処理を行う支援ソフトウェアを含めて，**論理診断システム**または単にシステムと呼ぶ．論理診断システムが論理診断処理の対象として入力する回路を，**診断対象回路**と呼んで区別する．

以下，拡張 X-伝搬法の概要について述べた後，論理診断エンジンの仕様と構成を示す．

3.1 拡張 X-伝搬法

拡張X-伝搬法は，誤り追跡入力と呼ばれる入力パターンを用いて，誤りの特定と修正を行う．**図 4** にその概念を示す．誤り追跡入力 $(X, 0, 1)$ を与えた場合を考える．0 および 1 は論理定数を表す．ブール変数 X は，いずれかの論理定数を値とする．その否定を $\overline{X}$ で表す．誤り追跡入力に対して，機能仕様を満足する正しい回路では外部出力が X となるが，誤りを含む回路では定数が出力される．そこで，誤りを含む回路に X を出力させるような修正方法を提示する．一つの



（a） 正しい回路 （b） 誤りを含む回路
（a） Correct circuit （b） Errorneous circuit

図 4 拡張 X-伝搬法の概念
Fig. 4 Concept of EX-algorithm.

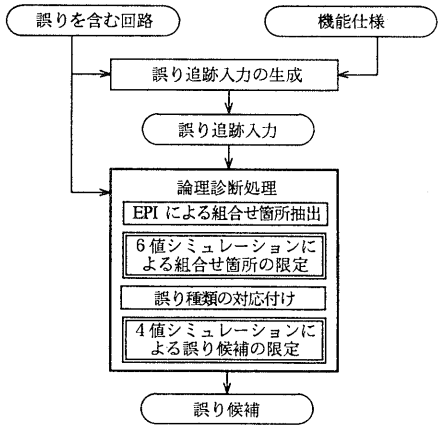


図 5 拡張 X-伝搬法による論理診断
Fig. 5 Logic diagnosis based on EX-algorithm.

ゲートまたは外部入力端子に対応する**誤り箇所**に，ゲート機能選択誤り等の**誤り種類**を対応させて，その可能性を判断する．異なる m 個の誤り箇所からなる空でない集合を，多重度 m の**組合せ箇所**と呼ぶ．

図5に，拡張X-伝搬法による論理診断処理を示す．誤り種類を特定しない組合せ箇所の段階で，候補となりえない修正方法を早期に削除することで，計算量の増加を抑える点を特徴とする．

（1） 誤り可能性の指標 EPI (Error Possibility Index) による組合せ箇所の抽出

誤り可能性の指標 EPI は，回路中のある箇所の外部出力に対する可制御性を表す指標である．EPI を用いて，誤りを含む回路の外部出力値が修正によって変化する可能性がある組合せ箇所のみを抽出する．

（2） 6値シミュレーションによる組合せ箇所の限定

4信号値 *0*, *1*, *X*, $\overline{X}$ に，X または $\overline{X}$ となる可能性がない不定信号値D，可能性がある不定信号値Eを加えた6信号値を用いたシミュレーション，すなわち6値シミュレーションの結果，外部出力値がXまたはEとならない組合せ箇所を削除する．

（3） 誤り種類の対応付け

最初の誤り追跡入力に対する各部の信号値を決定し，(1)，(2)の処理で限定された組合せ箇所のそれぞれについて，外部出力に X が出力される各箇所の出力値の組合せをシミュレーションによって求める．さらに，組合せ箇所に含まれる各誤り箇所に，求めた出力値を実現する誤り種類を対応づけて，誤り候補とする．

（4） 4値シミュレーションによる誤り候補の限定

2番目以降の誤り追跡入力について，4信号値 *0*，*1*，*X*，$\overline{X}$ を用いた4値シミュレーションを行い，外部出力が機能仕様と一致しない誤り候補を削除する．残存する誤り候補を修正方法として提示する．

我々は，図5の論理診断処理において二重枠で囲まれた，6値シミュレーションと4値シミュレーションの処理を RM-I 上に実現した．対応する構成情報を用意して，逐次的に RM-I の内部機能を設定変更することで，単一のハードウェア上で複数の処理を実行した．

3.2 論理診断エンジンの仕様

構築した論理診断エンジン LDE の仕様を示す．

- 対象とする誤り種類

  ゲート機能選択誤り，インバータ過剰/欠落誤り，信号線過剰誤りに対応する．

- 対象回路規模

  最大 4,000 ゲートとする．

- 処理速度

  2重までの誤りを対象として1,000ゲート規模の回路に100個の誤り追跡入力を加えたときに，1分程度で論理診断処理を完了するために，50万イベント/秒を達成する．

- 信号値

  0, *1*, *X*, $\overline{X}$, *D*, *E*, *I* の7値を用いる．6値シミュレーションで用いる6値に加えて，各ゲートの入出力端子および外部入出力端子の信号値の初期値として *I* (Initial) を用いる．

- 処理方式

  遅延評価の必要がないため，ランク順素子評価に基づくイベント駆動方式とする．すなわち，外部入力端子からのランク順で各素子の入力に対するイベント（信号値変化）の有無を調べ，イベントがある場合は素子の出力値を評価する．評価前に対して素子出力値が変化した場合，それをイベントとして伝搬する．

3.3 処理方式

論理診断システムの処理手順を以下に示す．

Step 1 誤りを含む回路情報と誤り追跡入力をもとに，指標 EPI を用いて多重度 m の組合せ箇所を抽出する．

Step 2 RM-I に6値シミュレーション・エンジンの構成情報を設定し，抽出された組合せ箇所について6値シミュレーションを行う．

Step 3 Step 2で限定された組合せ箇所について，誤り種類を対応させることで誤り候補を生成する．

Step 4 RM-I に4値シミュレーション・エンジンの構成情報を設定し，生成された誤り候補について4値シミュレーションを行う．限定された誤り候補を修正方法として出力する．

Step 1 および3については支援ソフトウェアで実行する．

論理診断エンジンで実行する6値シミュレーションと4値シミュレーションの処理では，適用するすべての誤り追跡入力について，それぞれ残存する組合せ箇所および誤り候補に対応する修正の結果，外部出力が X（または E）となる可能性を調べる．これら二つの処理は，互いに異なる素子評価法を必要とするが，処理方式については同様に議論できるので，以下では両

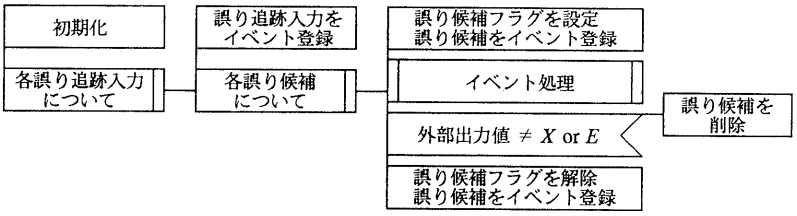


図 6 論理診断エンジンの処理
Fig. 6 Processs flow on Logic Diagnosis Engine.

者を代表して4値シミュレーションについて扱うこととする．そこで，誤り追跡入力と誤り候補に関する二重ループの順序が問題となる．従来のソフトウェア[12]では，誤り候補に関するループを外側に位置づけるとともに，誤りを仮定しないシミュレーションの結果を初期値として毎回設定していた．しかし，シミュレーションによってイベント発生率を評価した結果，**図6**に示す処理方式，すなわち誤り追跡入力を外側に位置づけ，初期値設定を行わずに前回のシミュレーションによる評価値をそのまま利用する方式が，最低のイベント発生率を示した．そこで，この方式を採用した．各誤り候補に対応して，誤り候補フラグの設定とその素子に関するイベントの登録を行うことで，誤りを仮定した素子評価演算を行う．次の誤り候補に対する処理結果に影響を及ぼさないように，イベント処理の後で誤り候補フラグを解除して，イベントの再登録を行う．イベント登録された素子は必ず評価の対象となるので，矛盾のない素子評価が可能となると同時に，毎回必要であった初期値設定を省くことができる．

3.4 構成と高速化のための工夫

論理診断エンジンの構成を**図7**に示す．論理診断エンジンは，プロパゲータ，スケジューラ，演算・比較器の三つのブロックから構成される．それぞれの役割を以下に示す．

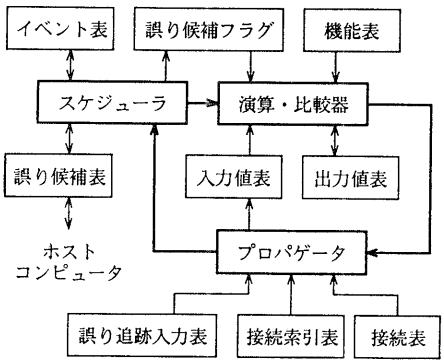


図 7 論理診断エンジンの構成
Fig. 7 Logic Diagnosis Engine.

（1） プロパゲータ

イベント伝搬処理を担当する．また，誤り追跡入力表から，外部入力に与えられた素子番号と新しい信号値を読み出す．入力値表の対応する部分にその値を書き込み，イベント登録のためスケジューラに素子番号を送る．また，演算・比較器から送られる素子出力値変化を，接続索引表・接続表から求めた接続先の素子に伝搬する．

（2） スケジューラ

イベント登録/取出し，誤り候補の管理に加えて，エンジン全体の実行制御を行う．誤り候補表から誤りを仮定する素子の番号を読み出して，対応する誤り候補フラグの設定/解除を行う．各素子に対応する1ビットのイベント・フラグで，その入力へのイベントの有無を表す．イベント処理においては，このフラグが設定された素子の番号を順次取り出し，終了時に外部出力信号値が X（または E）とならない場合は，その誤り候補を削除する．

（3） 演算・比較器

演算器は，スケジューラから送られる各素子について，その出力値を評価する．誤り候補フラグが設定された素子については，他と異なる演算を行う．比較器によって，出力値の変化が検出されると，プロパゲータに新たなイベントとして送られる．

以上の構成に基づいて，論理診断エンジンを試作した[13]．ここではこのエンジンを，LDE0と呼ぶことにする．LDE0は，200ゲート程度までの診断対象回路例について，約30万イベント/秒の処理速度を達成した．しかし，1,000ゲート程度の回路に対して実験を行った結果，イベント発生率の低下によって10万イベント/秒以下に処理速度が低下したため，目標とする1分程度の処理時間の達成は困難となった．RM-Iでは FPGA 上に回路が実現されるため，設計変更は容易である．そこで，高速化のために以下の工夫を行った論理診断エンジン LDE を新たに開発した．

●イベント・フラグの並列読出し
LDE0 ではビット単位でイベント・フラグを読み出していたため，イベント発生率が低い例ではその読出し処理に多くの時間を費やしていた．LDE では16ビット単位で並列して読み出す．さらに，読み出した値をレジスタに格納しておき，イベント登録/取出しが該当する16素子の範囲内であれば，そのレジスタに対して読み書きを行うことで，メモリ・アクセスの回数を削減する．さらに，誤り候補としてイベントを登録する場合の最小素子番号を管理することで，シミュレーションの範囲を限定する．

●誤り候補表のデータ構造
外部出力値の判断に基づいて誤り候補が削除される場合，削除されたことを示すフラグを誤り候補表に設定する．LDE0 では，毎回誤り候補表を読み出してこのフラグを調べていたため，時間を要していた．LDE では，削除された誤り候補について，素子番号の代わりに次に有効な誤り候補のアドレスを格納することで，誤り候補表の走査に必要な時間を短縮する．

●演算処理，伝搬処理の高速化
LDE では素子の出力値評価を LDE0 の2クロックから1クロックに短縮した．また伝搬処理に際して，LDE0 ではファンアウト数に関する制限をなくすために，まず素子番号をアドレスとして接続索引表を参照して，得られた接続表の先頭アドレスから順次ファンアウト先を求めていた．LDE では，最初のファンアウト先については素子番号をアドレスとして接続表から直接読み出すことで，伝搬処理を高速化する．2番目以降のファンアウトについても，接続表の読出しとイベント登録のパイプライン処理によって時間を短縮する．

以上の工夫の結果，初期状態を含む状態数について15から8に削減された．

3.5 RM-I への割付け

RM-I のアプリケーション開発にあたっては，最初になんらかの記述言語を用いて動作レベルのシミュレーションを行う．我々は，もとのソフトウェアとの親和性を考慮して，C言語を用いている．この段階で，アルゴリズムの確認のみならず，格納すべきデータのビット幅や容量など，RM-I のメモリの割付けを考慮する．前節で述べたようなデータ構造の工夫による高速化は，しばしば大きな効果を発揮することがあり，RM-I のアーキテクチャが許す範囲で検討する価値がある．動作レベルの確認が終了した段階で，メモリへの割付けを仮に決定する．クロックを意識したレジスタ転送レベルの記述によって，性能を評価する．この段階で回路の FPGA への割付け，ならびにバスや個別配線など FPGA 間の転送路の割付けを決定して，転送路の衝突がないことを確認する．メモリの割付けについても，性能向上のために変更する可能性がある．以上の割付け工程について，現状では次の点を考慮してすべて人手で行っている．

●ハードウェア資源の分散と遅延
FPGA 内部の配置・配線効率を考慮して，各 FPGA にできるだけ均等にハードウェア資源を割り付ける．また，出力バッファの遅延が比較的大きい（通常 23 ns, fast 設定で 6 ns）ので，クリティカルな経路を FPGA 間で分断しないように注意する．

●並列化
同一クロックで並列実行する操作を増やして並列度を高めるため，同時に参照される可能性のあるデータを異なるメモリに割り付ける．

●FPGA 間の転送路
RM-I では FPGA 間接続は固定されており，有効な利用法を考える必要がある．実行モジュールの FPGA を共通に接続するバスと，各 FPGA 間の個別接続について，転送路の性格を考慮していずれかに割り付ける．

LDE に関する回路とデータの割付け結果を，**表2**に示す．回路については，ほぼブロック単位で分割している．複数のブロックから参照される素子番号や状態番号について，バスを転送路とした．データの割付けについては，大きな容量を必要とする誤り候補表を独立して Memory 4 に配置した．もともと接続索引表については Memory 3，出力値表については Memory 2 に配置していたが，それぞれ接続表，入力値表との並列アクセスのために，Memory 1，Memory 3 に移した．

メモリバンク数の制限により，同時には書き込みの対象とならないデータをワード方向に連結する必要があった．そこで，1クロックで部分書き込みを完了するread-modify-write 方式と，2クロックに分ける方式を比較検討した．クロック周期を τ とし，クロック立上りから状態デコードとメモリ・アドレス確定までの時間を t_a，メモリ読出し時間を t_r，メモリ読出しデータの確定から次のクロックまでに必要な時間を

表 2 回路およびデータの割付け結果
Table 2 Mapping of circuits and data tables.

(a) 回路ブロックの割付け
(a) Circuits

FPGA	回路ブロック
FPGA 1	スケジューラ (1) プロパゲータ (1)
FPGA 2	演算・比較器
FPGA 3	プロパゲータ (2)
FPGA 4	スケジューラ (2)

(b) データの割付け
(b) Data tables

Memory	データ
Memory 1	イベント表，誤り追跡入力表，接続索引表
Memory 2	入力値表，機能表，誤り候補フラグ
Memory 3	接続表，出力値表
Memory 4	誤り候補表

t_l，マージンを t_m とすると，

$$t_a+t_r+t_l+t_m \leqq \tau \qquad (1)$$

を満足する必要がある．LDE では，$t_a \fallingdotseq 50$ ns，$t_r \fallingdotseq 30$ ns，$t_l \fallingdotseq 120$ ns となり，マージン $t_m=50$ ns を確保するには，$\tau=250$ ns，すなわち 4 MHz のクロックを採用する必要があった．t_l を支配する組合せ論理の部分を2クロックに分割することも検討したが，かえって処理時間の増大を招くために回避した．部分書込みのための modify-write は約 50 ns$<t_l$ で完了するため，read-modify-write を採用した．

4. 実　　験

RM-I を，230 mm×330 mm のボード上に実現した．パーソナル・コンピュータ (NEC PC-H 98 S model 8+DX 2 OP : 40 MHz, 15 MIPS) をホストコンピュータとした．支援ソフトウェア，および比較対象となるソフトウェアについては，C 言語 (GCC 2.4.1) で記述した．構成情報の設定については，各 FPGA にビット並列で与えることにより，0.06 秒で完了することを確認した．

表 3 に，実験に利用した診断対象回路例を示す．Y2，Y3 は，我々が設計したシミュレーション・エンジン TASSE II[11]から，その他は ISCAS ベンチマーク回路[14]から抽出された単一出力回路である．前者については設計時の誤りの履歴が存在したので，これらを仮定した．ISCAS ベンチマーク回路では，ランダムに誤りを仮定した．Y1-1, Y1-2, Y1-3 は，同一の回路にそれぞれ多重度 1，2，3 の誤りを加えた回路例である．その他は，すべて多重度 2 の誤りを含む．実験では，100 個の誤り追跡入力を利用して，仮定した誤りの数に等しい多重度までの誤り候補を列挙した．

6 値シミュレーションおよび 4 値シミュレーションを LDE で実行した場合の処理速度を，それぞれ**表 4**，**表 5** に示す．入力信号値の変化に応じて実行される素子評価の回数を，イベント数としている．図 6 に示した処理方式を採用することにより，特に大きな回路例

表 3 診断対象回路例
Table 3 Sample circuits for logic diagnosis.

回路名	ゲート数	外部入力数	誤り箇所数*	誤りの多重度	原回路（出力名）
Y1-1	40	8	48	1	C 7552 (397)
Y1-2	40	8	48	2	C 7552 (397)
Y1-3	40	8	48	3	C 7552 (397)
Y2	60	21	81	2	TASSE-II
Y3	75	19	94	2	TASSE-II
Y4	116	25	141	2	C 5315 (623)
Y5	102	41	143	2	C 499 (OD31)
Y6	146	36	182	2	C 432 (421)
Y7	130	60	190	2	C 880 (878)
Y8	322	41	363	2	C 1355 (1355)
Y9	553	33	586	2	C 1908 (75)
Y10	828	122	950	2	C 2670 (308)
Y11	1,096	194	1,290	2	C 7552 (418)
Y12	1,458	50	1,508	2	C 3540 (405)

* 誤り箇所数＝ゲート数＋外部入力数

表 4 6値シミュレーションに関する LDE の処理速度
Table 4 Processing speed of LDE for six-valued simulation.

回路名	イベント数	処理時間(秒)	イベント発生率(%)	処理速度(万イベント/秒)	ソフトウェア処理に対する速度比
Y1-1	3.72×10^{3}	0.0026	17.47	145	54.4
Y1-2	2.47×10^{5}	0.18	11.73	139	22.3
Y1-3	1.32×10^{6}	1.04	11.77	127	20.1
Y2	2.11×10^{5}	0.15	7.47	136	25.0
Y3	1.62×10^{5}	0.11	7.82	144	28.8
Y4	8.79×10^{5}	0.66	4.88	134	27.1
Y5	9.48×10^{5}	0.75	4.31	127	26.8
Y6	1.81×10^{6}	1.30	4.63	139	30.6
Y7	9.97×10^{4}	0.074	4.41	135	31.4
Y8	9.90×10^{6}	8.23	2.58	120	40.4
Y9	1.55×10^{7}	10.37	4.35	149	37.5
Y10	2.28×10^{6}	2.16	1.35	106	44.6
Y11	1.29×10^{6}	1.51	0.76	85	59.0
Y12	1.55×10^{7}	11.81	2.13	131	46.7
平 均	3.58×10^{6}	2.74	6.12	130	35.3

表 5 4値シミュレーションに関する LDE の処理速度
Table 5 Processing speed of LDE for four-valued simulation.

回路名	イベント数	処理時間(秒)	イベント発生率(%)	処理速度(万イベント/秒)	ソフトウェア処理に対する速度比
Y1-1	3.30×10^{3}	0.0023	16.85	142	81.8
Y1-2	4.55×10^{4}	0.043	5.55	105	100.9
Y1-3	5.04×10^{4}	0.063	4.15	80	15.2
Y2	1.81×10^{4}	0.017	4.11	109	41.5
Y3	8.87×10^{3}	0.0066	9.40	135	57.9
Y4	6.84×10^{4}	0.065	2.46	106	40.0
Y5	1.16×10^{5}	0.13	1.72	89	44.4
Y6	6.28×10^{5}	0.73	1.34	86	53.8
Y7	1.60×10^{4}	0.012	7.28	132	38.6
Y8	1.48×10^{5}	0.17	1.10	88	44.0
Y9	1.87×10^{5}	0.20	0.88	92	36.0
Y10	2.27×10^{5}	0.39	0.48	58	44.7
Y11	1.67×10^{5}	0.25	0.49	66	47.9
Y12	4.15×10^{5}	0.44	0.68	95	59.6
平 均	1.50×10^{5}	0.18	4.04	99	50.5

でイベント発生率が低く抑えられている．処理速度の平均値について，6値シミュレーションでは130万イベント/秒，4値シミュレーションでは99万イベント/秒となり，仕様を満足している．6値シミュレーションのほうが，削除されずに残存する誤り候補（組合せ箇所）の割合が高いため，イベント発生率，処理速度ともに高い値を示している．また，15 MIPS の計算機上で，同一の処理方式を採用したソフトウェアによる6値および4値シミュレーションのみの処理速度に対して，それぞれ平均で 35.3 倍，50.5 倍の処理速度を達成した．

3.3節で示した Step 1, Step 3 は，支援ソフトウェアで実行されるが，このために必要な時間も含んだ全体の処理時間を**表 6** に示す．支援ソフトウェアの処理時間（A 1），RM-I への構成情報設定とデータの授受に要する通信時間（A 2），LDE の実行時間（A 3），さらにそれらの和である総処理時間(A)を示す．Y 8 から Y 12 までの回路例では，支援ソフトウェアの実行時間の割合が大きいが，ほぼ1分程度で処理全体を終了している．ソフトウェアですべての処理を行った場合の総処理時間（B）と比べて，4.66 倍の高速化を達成している．Step 3 におけるシミュレーショ

表 6 論理診断処理全体に要する時間
Table 6 Total processing time.

回路名	LDE 利用時の総処理時間（秒）				従来の総処理時間B（秒）	総処理時間の比（B/A）
	A1	A2	A3	A		
Y1-1	0.55	0.58	0.005	1.135	0.88	0.78
Y1-2	1.14	1.26	0.22	2.62	9.47	3.61
Y1-3	10.13	1.34	1.11	12.58	32.08	2.55
Y2	1.34	1.21	0.17	2.72	5.90	2.17
Y3	1.59	0.83	0.12	2.54	5.20	2.05
Y4	4.26	1.31	0.72	6.29	24.68	3.92
Y5	4.56	0.96	0.88	6.40	30.35	4.74
Y6	8.08	1.47	2.03	11.58	87.04	7.52
Y7	1.94	0.89	0.09	2.92	4.73	1.62
Y8	56.99	1.21	8.40	66.60	397.12	5.96
Y9	32.26	1.59	10.58	44.43	428.85	9.65
Y10	20.87	1.38	2.55	24.80	134.58	5.43
Y11	20.50	1.40	1.76	23.66	121.45	5.13
Y12	47.79	1.44	12.25	61.48	625.18	10.17
平 均	15.14	1.21	2.92	19.27	136.25	4.66

A1：支援ソフトウェアの処理時間，A2：構成情報設定と通信時間，
A3：LDE の実行時間，A：総処理時間（A1～A3 の和）

ン処理に，回路例 Y8 で 48 秒，Y12 で 27 秒程度を要しており，この部分も RM-I 上で実現すれば，さらなる時間短縮が可能となる．

5. 考　　察

5.1 高速化の工夫による効果

3.4 節で述べたように，最初に開発した LDE0 に対して，ⅰ)イベント・フラグの並列読出し，ⅱ)誤り候補表のデータ構造，ⅲ)演算・伝搬処理の高速化等の工夫を加えた結果が LDE である．そこで，両者の各処理項目に関する処理時間を比較した．その結果を，**表 7** に示す．6 値シミュレーション，4 値シミュレーションのそれぞれについて，LDE による処理時間を 1 とする相対値で示している．特にⅰ)による効果が大きく，イベント取出し処理については，それぞれ LDE0 に対して，24.1 倍，34.7 倍の高速化を達成している．シミュレーション処理全体では，それぞれ 6.3 倍，11.78 倍の効果があった．

このように汎用エンジンでは，汎用計算機上のソフトウェアと同様に，アプリケーション開発後にそのバージョン・アップが可能であることが確認された．

5.2 論理シミュレータへの応用

我々は，論理診断エンジンに加えて，割当て遅延モデルによるゲートレベル論理シミュレータ（LSIM）を RM-I 上に実現した．文献 11)に示されるイベント駆動アルゴリズムを採用した．LSIM は，伝搬フェー

表 7 処理時間に関する LDE と LDE0 の比較
Table 7 Comparison between LDE and LDE0 in processing time.

処 理 項 目	6 値シミュレーション		4 値シミュレーション	
	LDE の処理時間	LDE0 の処理時間（LDEとの比）	LDE の処理時間	LDE0 の処理時間（LDEとの比）
誤り追跡入力のイベント登録	0.02	0.02(1.0)	0.05	0.05(1.0)
誤り候補フラグの設定と解除	0.16	0.44(2.8)	0.25	1.85(7.4)
イベント取出し	0.19	4.58(24.1)	0.26	9.02(34.7)
出力値演算・比較	0.32	0.65(2.0)	0.25	0.49(2.0)
イベント伝搬	0.31	0.61(2.0)	0.19	0.37(2.0)
合　　計	1.00	6.30	1.00	11.78

* 処理時間はそれぞれ LDE の合計を 1 とする相対値

表 8 バスおよびメモリのアクセス比率
Table 8 Bus and memory access ratio.

アクセス対象	6値シミュレーションでのアクセス比率 (%)			4値シミュレーションでのアクセス比率 (%)		
	最小値	平均値	最大値	最小値	平均値	最大値
バス	62.4	69.6	80.0	62.7	71.8	84.5
Memory 1	63.5	79.9	94.9	65.3	80.2	94.8
Memory 2	47.1	71.9	80.5	33.2	58.1	80.5
Memory 3	42.5	63.1	76.6	25.8	43.4	67.5
Memory 4	3.9	15.9	29.8	8.7	25.5	51.3

ズと評価フェーズを繰り返し実行する．それぞれのフェーズは，5段のパイプラインで処理が進む．しかし，RM-I では FPGA 数とメモリバンク数が4であるため待ち状態が発生した．このため，各フェーズの実行に要するクロック数は2クロックずつ，合計4クロックとなった．4 MHz のクロックで平均 100 万イベント/秒を超える処理速度を達成した．RM-I が5個以上のメモリバンクをもつと仮定すると，ハザードなしにパイプラインを動作させることが可能となるため，1イベントあたり各フェーズ1クロック，合計2クロックの処理時間が必要となり，性能は2倍になることが見込まれる．

5.3 RM-I のアーキテクチャに関する考察

RM-I のアーキテクチャについて，簡単な検討を加える，文献6)によれば，FPGA 間のデータ転送について，33 M 語/秒の転送が可能とされている．RM-I ではデータ転送の方式および回路について設計者に任せているが，これまでの応用例では 4 M 語/秒と，一般の並列計算機と比較すると低い速度で利用していた．しかし，RM-I では FPGA 数が4と少ない点，クロック単位の細粒度並列処理を行うために1回で転送すべきデータ量も少ない点，さらに LDE では通信が静的にスケジューリングされていた点などの要因により，ボトルネックとはならなかった．**表 8** に，LDE におけるバスおよびメモリについて，全クロック数に対するアクセスがあったクロック数の比，すなわちアクセス比率を示す．平均でバスについては約 70%，メモリについては 16〜80% 程度であった．Memory 4 については，容量を重視して誤り候補表のみを割り付けたため，低い値を示している．RM-I のように FPGA を用いて論理回路を構築する場合，3.5 節で示したように組合せ論理の部分や，入出力部分の遅延が問題となる．この部分を高速化するためには，FPGA の上位モデル[15]の採用が有効である．

また，RM-I は論理シミュレータ LSIM への応用で問題となったように，メモリバンク数，ゲート・メモリ容量等に制約がある．FPGA およびメモリバンク数の拡張にともなって，FPGA 間の接続形態が与える影響が大きくなると予想される．接続形態を可変とするために FPGA，または配線変更用 LSI である FPIC[16]を導入して，より高いレベルでの性能と柔軟性の両者を可能とすることが望まれる．

6. おわりに

本論文では，複数の処理を単一のハードウェアで実行できる汎用エンジンを提案するとともに，そのプロトタイプとして開発した RM-I について述べた．さらに，RM-I 上に論理診断エンジンにおける二つの処理と論理シミュレータを構築し，単一のエンジンが高速性を失わずに，複数の用途に利用できることを確認した．特に論理診断エンジンについては，高速化のための工夫によってソフトウェア処理の約 35〜50 倍の処理速度を達成し，性能向上のための設計変更も比較的容易に行えることが示された．

現状では，汎用エンジンへの割付けを人手で決定しているが，その自動化を含んだ設計支援環境の構築と，性能と柔軟性をさらに高めるプロトタイプの開発が課題として考えられる．

参考文献

1) Blank, T.: A Survey of Hardware Accelerators Used in Computer-Aided Design, *IEEE Design and Test of Computers*, Vol. 1, No. 3, pp. 21-39 (1984).
2) Pfister, G. F.: The Yorktown Simulation Engine; Introduction, *Proc. 19th ACM/IEEE Design Automation Conference*, pp. 55-59 (1982).
3) Sato, M., Kubota, K. and Ohtsuki, T.: A

Hardware Implementation of Gridless Routing Based on Content Addressable Memory, *Proc. 27th ACM/IEEE Design Automation Conference*, pp. 646-649 (1990).
4) Tham, K. Y.: Parallel Processing for CAD Applications, *IEEE Design and Test of Computers*, Vol. 4, No. 5, pp. 13-17 (1987).
5) Agrawal, P. et al.: MARS: A Multiprocessor-Based Programmable Accelerator, *IEEE Design and Test of Computers*, Vol. 4, No. 5, pp. 29-36 (1987).
6) プログラマブル・ゲートアレイ データブック, ザイリンクス社 (1990).
7) *MARS III—IP Series*, PiE Design Systems, Inc. (1993).
8) *Enterprise Emulation System—Product Description*, Quickturn Systems, Inc. (1992).
9) Gokhale, M., Holmes, W., Kopser, A., Lucas, S., Minnich, R. and Lopresti, D.: Building and Using a Highly Parallel Programmable Logic Array, *Computer*, Vol. 24, No. 1, pp. 81-89 (1991).
10) Van den Bout, D. E., Morris, J. N., Thomae, D., Labrozzi, S., Wingo, S. and Hallman, D.: AnyBoard: An FPGA-Based, Reconfigurable System, *IEEE Design and Test of Computers*, Vol. 9, No. 3, pp. 21-30 (1992).
11) Suganuma, N., Tomita, M. and Hirano, K.: A Compact Simulation Engine with Flexible Logic Model Expansion, *IEEE International Conference on Systems Engineering*, pp. 416-419 (1992).
12) 上田伸人, 菅沼直昭, 山本 保, 冨田昌宏: 多重論理設計誤りを対象とする自動追跡手法, 情報処理学会研究会報告, 91-DA-60, pp. 185-192 (1991).
13) 菅沼直昭, 村田之広, 冨田昌宏, 平野浩太郎: 汎用エンジンの開発と論理診断への応用, DA シンポジウム '92, pp. 89-92 (1992).
14) Brglez, F. and Fujiwara, H.: A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translation in FORTRAN, Special Session on ATPG and Fault Simulation, *Proc. ISCAS '85* (1985).
15) *The XC 4000 Data Book*, Xilinx, Inc. (1991).
16) *Programmable Interconnect Data Book*, Aptix Corp. (1993).

(平成5年6月 3 日受付)
(平成6年7月 14 日採録)

沼 昌宏 (正会員)
(旧姓 冨田)

1960年生. 1983年3月東京大学工学部精密機械工学科卒業. 助手を経て1989年東京大学講師. 1990年5月より神戸大学大学院自然科学研究科講師, 現在に至る. 工学博士. 主に論理装置のCAD, 情報処理工学に関する研究に従事. IEEE, 電子情報通信学会, プリント回路学会各会員.

菅沼 直昭 (正会員)

1965年生. 1989年3月神戸大学工学部電子工学科卒業. 1991年3月同大学大学院工学研究科修士課程修了. 1994年3月同大学院自然科学研究科博士課程修了. 工学博士. 論理装置のCAD, 設計自動化に関する研究に従事.

黒木 修隆

1965年生. 1990年3月神戸大学工学部電子工学科卒業. 1992年3月同大学大学院工学研究科修士課程修了. 現在, 同大学院自然科学研究科博士課程に在学中. 画像処理の研究に従事. 電子情報通信学会会員.

平野浩太郎

1935年生. 1960年3月大阪大学工学部通信工学科卒業. 1965年同大学大学院博士課程修了. 同年神戸大学工学部電気工学科助教授. 現在は, 同学部電気電子工学科教授, 同大学大学院自然科学研究科教授を兼任. 工学博士. ディジタル信号処理, ディジタル通信, LSI設計の研究に携わる. IEEE上級会員, Eta Kappa Nu, 電子情報通信学会, システム制御情報学会各会員.