LUCASを使用した回路シミュレータの性能評価

八木浩行† 檀良†

我々は，PC環境での拡張ノイマン型ハードウエア・ソフトウエア協調コンピューティングシステムによる回路シミュレーション環境を提案し，行列計算専用ハードウエア・アクセラレータを搭載した回路シミュレータを開発している．本論文では，Berkeley SPICE2G.6を逆アセンブルすることにより得られた演算クロックカウントと，タイミングシミュレーションから得られるハードウエアでの演算クロックカウントの比較を行う．本アクセラレータの使用により，稠密行列の求解処理については1回あたり，従来のソフトウエア処理比で最大およそ20倍の高速化が達成できる．また，提案環境を実現する，従来PC上ではプロセッサの制限により通常使用できなかった，コード生成法の新アルゴリズムについても報告する．提案手法により，ベンチマーク回路の疎行列求解処理を評価すると，SPICE2G.6のコード生成法と比べ，1回の求解あたりおおよそ1.3倍の高速化が可能である．

Estimating the Performance of a Circuit Simulator Using LUCAS

Hiroyuki Yagi† and Ryo Dan (or Dang)†

We propose a new circuit simulation environment using an enhanced Neumann type software hardware co-computing system, and develop a circuit simulator using a dedicated hardware accelerator in this environment. In the present paper, we estimate the performance of this circuit simulator by comparing the operating clock counts for Berkeley SPICE2G.6 and our hardware. In this environment, our accelerator can solve 20 times faster than software calculation for full-matrix. In addition, we report a new "code generation method" algorithm for realizing this environment. Estimation of some benchmark circuits shows it can solve about 1.3 times faster than the original "code generation method" of SPICE2G.6 for each solution of the matrix.

1. はじめに

半導体素子製造技術の飛躍的な発展によりトランジスタの微細化が進み，その結果，集積回路の高集積化とともに設計技術も進歩している．超大規模回路設計においてはディジタル回路においてもアナログ的回路の振舞いが重要視されつつあり，昨今のトレンドであるSOC（System On a Chip）設計の核をなすシステムLSIにおけるデジアナ混在回路では，もはや全アナログ回路としての解析が必要とされ，高速でSPICEレベルの精度を保つアナログ回路シミュレータに対するニーズは増すばかりである．

最近，ハイ・パフォーマンス・コンピューティングの分野において，FPGA（Field Programmable Gate Array）デバイスの進歩（100 MHzを超える実用的な動作周波数の達成および1 Mゲートを超える1チップ内のゲート数の増加）による，FPGAs for Custom Computing Machinesと称される研究分野が注目されている．

回路解析分野においてもハードウエア・アクセラレータという観点からは，従来より様々な提案が行われているが，PCをターゲットとしたハードウエア・アクセラレータは我々の知る限り皆無であり，より安価な環境でのハイ・スピード・コンピューティングへの要求は無視できない段階に入っている．

本論文の主提案は，拡張ノイマン型ハードウエア・ソフトウエア協調コンピューティングシステムアプリケーションによる回路シミュレーション環境の提案である．我々の提案するコンピューティングシステムは，従来のメモリ上にコードおよびデータをストアするノイマン型に，再定義可能なゲートアレイによる演算器を付加したものである．アプリケーションは，従来型コードおよびデータとともに最適化された演算を実行するためのFPGA定義データを持ち，実行時に専用演算器を構成する．専用演算器LUCAS（LU decom-

† 法政大学工学研究科
Graduate School of Engineering, Hosei University

position CAlculation System）は回路シミュレーションにおいて最も重要である行列演算に特化され，ゆえに従来のハードウエア・アクセラレータに比べ単純で，インストラクションレベルでの最適化が達成される．

この提案の妥当性を検討するため，Berkeley SPICE 2G.6（以後単にSPICEと称する）の逆アセンブルを実行し，その演算クロックカウントと最適化されたハードウエア・アクセラレータのタイミングシミュレーションによる演算クロックカウントの比較を行う．さらに従来の一般的なプログラミング技法では達成が不可能であったPC上のコード生成法について議論し，本ハードウエア・アクセラレータの使用により可能となる，より短い生成コード長で高速に求解が可能なアルゴリズムを提案する．

本論文の残りの章では，ハードウエア・ソフトウエア協調コンピューティングシステムおよびアプリケーションプログラム（回路シミュレータDanSpice）とそのFPGA定義データにより構成される専用演算器LUCASについての説明を行う．2章では本論文に関係する回路シミュレーション，ハードウエア・アクセラレータにおけるこれまでの研究成果について略述する．3章で我々の提案する拡張ノイマン型ハードウエア・ソフトウエア協調コンピューティングシステムについての説明を行い，アプリケーションの一部であるFPGA定義ファイルにより実行時に構築される行列計算専用ハードウエアLUCASの構造を示す．4章はLUCASの演算クロックカウントとSPICEの演算クロックカウントの比較によりその性能評価を行う．5章において，従来型ソフトウエアコード生成法のPCへの導入について議論し，ハードウエアを使用したコード生成法アルゴリズムの提案を行う．6章で結論をまとめる．

2. これまでの研究成果

最もよく使用されている回路シミュレータSPICE[1]は，直接法を用い様々な回路の様々な解析について，正確に解を与えてくれる．直接法を用いた回路シミュレーションは，最近の大規模回路解析においてかなりのストレスを感じさせるほど遅いが，これは次の理由による．

(1) デバイスモデル式の複雑化にともない，その評価のため典型的に数百回もの倍精度浮動小数点演算が必要であること（バイパス法を導入するとかなり削減できるが，バイパス法の不完全さは周知のとおりである）．

(2) デバイスモデル式の非線形性がより強くなり，結果としてNewton-Raphson（NR）ループでの収束性が悪化し，NR反復回数の増加を招くこと．

これらは根本的にデバイスモデリングの問題であり，その改善のためにテーブルルックアップ法[2)~4)]やより高速な解析モデル[5)]あるいは，より簡単なモデル[6)]を使用することで対応されている．

また，(2)により，線形行列の求解の回数が結果としてNR反復回数とともに増加し，この非線形方程式を解くための線形方程式求解に必要な時間は $O(n^{\beta})$ で評価できる．ここで，n は行列の次元（以後，本論文中で n は行列の次元として参照される）であり，典型回路で，$1.1 \leq \beta \leq 1.5$ である[7)]．大規模回路においては，この線形行列求解に必要な時間が，全解析時間を支配する[7),8)]．

線形行列求解時間短縮のため，緩和法や回路の潜伏性，分割あるいはその演算並列化等が試まれている．緩和法に基づく解析[7)]は，回路の部分部分を独立に解き，回路の大きさにほぼ線形な時間で解くことを可能にしたが，対象回路がMOS回路等に限られ，帰還パスを持つ回路には適応が難しい．回路の潜伏性を利用するもの[9),10)]はデバイス評価回数を劇的に削減でき，対象回路によっては，かなりの高速化が達成できる．分割解析は行列求解の β を直接効果的に改善することができる．それらの演算並列化は確かに利用演算素子の増加とともに全解析時間の短縮に寄与するが，アムダールの法則からもどこかで必ず飽和し，コスト・パフォーマンスの観点から必ずしも魅力的ではない．

線形行列求解のため，コード生成法[11)]が，第1世代回路シミュレータ開発当時から検討され，プラットフォームあるいはその使用CPUによっては導入されてきた．最近では研究段階ではあるが，コード生成法を用いた並列処理[12)]も報告されている．これにより，かなりの実行インストラクションカウント削減が達成されるが，プラットフォームあるいはその使用CPUごとに，アセンブリ言語で記述するコード生成部を書き換えなければならない面倒な作業がある．また，PC環境においては，我々の知る限り，このコード生成法を利用した報告はなく，さらにキャッシュが標準的に備わる現在のコンピューティングシステムでは，巨大な生成コードの実行は，その性能を台無しにする等の困難がともなう．これらの様々な試みがなされても，回路シミュレーションはいまだに，かなり時間のかかる解析であり，より高速な回路シミュレータが望まれている．

ハードウエア・アクセラレータを従来のコンピュー

ティングシステムにおいて付加的に，様々なアプリケーションに特化した形で利用することが，いろいろな分野で行われてきた．回路シミュレーションへの適応についてのみ，いくつか例をあげると，回路シミュレーションマシンとしてのもの[13]，汎用共有分散メモリ型コンピューティングシステムであるがほぼ回路シミュレーション専用と見なせるもの[14]，アレイプロセッサを利用したもの[15]，汎用数値演算プロセッサを用いたもの[16]，列指向の並列性を導き並列化したもの[17]，連想スイッチ（比較器を使用し行列インデックスでon/off制御を行う）を使用したもの[18]，モデル評価部分を含めた回路シミュレータ全体についてCompiled Code方式によりハードウエア化したもの[19]，等がある．回路シミュレーションマシンはハードワイヤリングで構成され，汎用性はまったくない．Compiled Code方式は我々の提案に最も近いものであり，最近の報告[20]では，従来法よりおよそ300倍という高速化を達成しており，注目されるが，回路シミューレータ全体を最適コード化し，専用ハードウエアでのインストラクションレベルでの高速化を目指しており，我々の提案するFPGAを用いたソフトウエア・ハードウエア協調コンピューティング環境とは異なるものである．

3. 拡張ノイマン型ハードウエア・ソフトウエア協調コンピューティングシステム

3.1 システム構成

我々の提案する拡張ノイマン型ハードウエア・ソフトウエア協調コンピューティングシステムの構成を**図1**に示す．提案システムは従来のノイマン型コンピューティングシステムに高速なバスで接続されたFPGAで構成される再定義可能な演算処理部を持つ（あるいは，CPU内に再定義可能な演算処理部を構成できる）ものである．通常のアプリケーションは，CPUおよびFPUのみを使用し，ゆえにコードとデータをメモリ上にストアしてノイマン型コンピューティングシステムとして実行される．

特に高速化が求められ，一定の処理を何度も繰り返すアプリケーションは，起動時にFPGAのコンフィギュレーションを行い，その処理専用に最適化された演算処理部を構成する．このソフトウエアコードとハードウエア定義データをあわせ持つハイブリッドアプリケーションは，自己定義したハードウエアを使用することが許され，このハードウエアを使用するために最適化されたソフトウエアコード（Compiled Code）を実行できる．この例として，5章において本システムにおける回路シミュレーションのコード生成法を報告する．

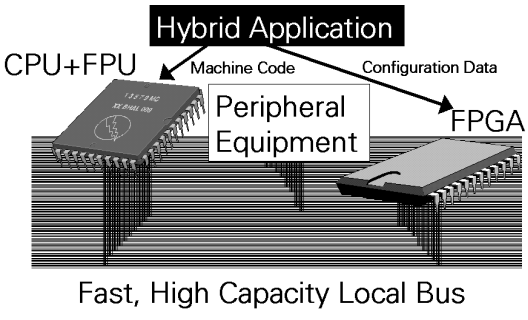


図1 提案システムの構成
Fig. 1 Structure of proposed system.

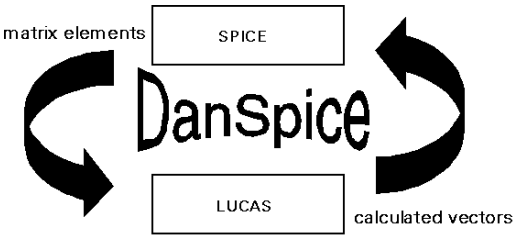


図2 回路シミュレータDanSpice
Fig. 2 Circuit simulator DanSpice.

今回我々が提案するプロトタイプシステムは一般的なPCのPCI32バス上にFPGAテストボード（以後，PCIボードと呼ぶ）を付加する構成となっている．SPICEベースの回路シミュレータDanSpiceを実行する場合の信号の流れを**図2**に示す．ホストPCのメモリ上にロードされたDanSpiceは，同時に，PCIボード上のFPGA内に，行列求解専用ブロックおよびその制御ブロックを定義し，以後この最適演算処理部（LUCAS）を利用することができる．回路シミュレーション中に行列求解処理が必要になると，ホストPC側からは処理の入力データである行列要素が送られ，PCIボード側からは計算されたベクトルが送られ，この専用演算器はハードウエアサブルーチンとして，利用される．

3.2 PCIボードの構成

図3に，今回使用するPCIボードの構成を示す．このうちI/FブロックおよびMemoryブロックについては，ハードワイヤリングによる構成で，アプリケーション側では再定義できない．アプリケーションは，定義データによりFPGAブロック（網掛部分）内に構成される制御部および専用演算処理部（行列求解専用アクセラレータLUCAS）のみを再定義し利用する．

3.2.1 I/FおよびMemoryブロック

現状では，FPGA内に構成されるLUCASがホストPCの主記憶装置に直接アクセスすることは許され

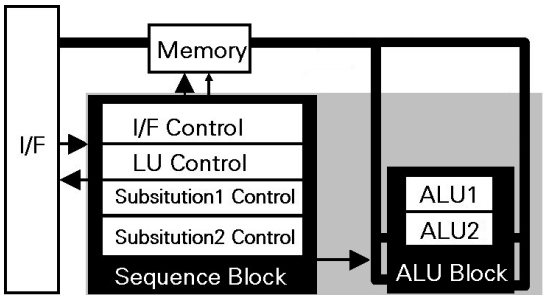


図 3 PCI ボードの構造
Fig. 3 Architecture of PCI board.

ていない．したがって，演算に必要なデータはすべて PCI ボード上のメモリーに一度転送し，演算後ホスト側に書き戻す必要がある．

プロトタイプシステムにおいては，PC において最も一般的である PCI32 バス（PCI インタフェース LSI：PLX Technology 製 PCI9052，バス幅 32 ビット，最大動作クロック 33 MHz）を使用し，SRAM により構成される Memory ブロック（最大 256 Mbytes）インタフェースを標準で持つアリテック社製 PCI ボード（P5E-N）を改造し使用している．PCI バスの性能を最大限に引き出すバースト転送，マスタ機能は現時点では採用していない．また，PCI ボードを使用するための OS 側のドライバ開発には，ツールクラフト社製 WinRT を使用した．

3.2.2 FPGA ブロック

3.2.2.1 制 御 部

制御部は次節で説明する専用演算処理ブロックの制御（LU 分解および代入計算 1：前進消去，2：後退代入の添え字処理等），メモリ I/O（メモリからレジスタへの代入，結果のメモリへの保存）および I/F の制御等 LUCAS におけるすべての演算を制御し，それぞれの演算状態をステイトに割り当てるステイトマシンとして構成される．

3.2.2.2 専用演算処理部

直接法を用いた行列求解は，すべて，式 (1) の 3 項演算を含む 3 重ループによる行列分解部と，2 重ループの消去演算に帰着される．

```
while(条件 1){
  while(条件 2){
    while(条件 3){
```

$$a_{ij} = a_{ij} - (a_{ik} \times a_{kj}/a_{kk}); \qquad (1)$$

```
}}}
```

条件式 1〜3 は使用するアルゴリズムにより異なる．どのアルゴリズムを用いても，以下の 2 つの演算式のみを使用すれば，すべての行列求解演算が可能である．

表 1 演算クロックサイクルの比較
Table 1 Comparizon of operating clock cycles.

	除算	乗算	加算	ロード	ストア
i387	94	32-57	29-37	25	45
i486	73	14	8-20	3	8
Pentium	39	3, 1	3, 1	1	2
LUCAS_1	56	56	5	8	8
LUCAS_2	21	5（積和算）		1	1

$$a' = a/k, \qquad (2)$$

$$b' = b - a \times k. \qquad (3)$$

よって，行列求解専用アクセラレータ LUCAS ではこの 2 つの演算を行う ALU（Arithmetic Logical Unit，以後式 (2) に対して ALU1，式 (3) に対して ALU2 と称する）のみが定義される．

3.3 FPGA への導入手順

FPGA ブロックのハードウエア開発はすべて HDL（Hardware Description Language）により行った．

3.3.1 制 御 部

制御部は RTL（Register Transfer Level）記述によるステイトマシンとして構成される．ステイトマシンにはデコード方式（少ないフリップフロップ出力をデコードして制御する方式）とワンホット方式（各ステートに 1 つのフリップフロップを割り当てる方式）があり，我々は動作周波数に有利なワンホット方式を採用している．

3.3.2 専用演算処理部

専用演算処理部は RTL で論理合成すると膨大な回路を生成する．そのためゲートレベルでの記述により構成される．

通常回路シミュレーションに必要な数値精度は倍精度であり，我々の演算処理部についてもすべて IEEE754 規格フォーマットの倍精度が保証されている．

倍精度（64 ビット）浮動小数点演算の場合，仮数部は 52 ビットであり，誤差を考慮すると，通常の乗除算では仮数部の桁合わせのため最悪値で 56 clock かかってしまう．構成された ALU および，メモリ処理に必要な演算クロックを**表 1** にまとめる．インテル社製 MPU の演算クロックサイクルは，データブック[21]によるパイプライン処理を考慮した最良値であり，各演算は付加的演算を含まない（必要な数値はレジスタ内にあると仮定している）．LUCAS の演算クロックは，最悪値（現時点ではパイプライン化を行っていない）を示している．ここで，LUCAS_1 は，演算処理部においてシフトレジスタ型の構成を採用したものであり，LUCAS_2 はアレイプロセッサおよび高基数除算型の構成を採用したものである．また，制御部につ

いてはどちらも同じものである．i486以降のMPUではLUCAS_1より優秀な性能を示しているが，これは，たとえば演算パイプライン等の並列化手法を用いた高速化がはかられているためである．

FPGAにおいて，高速化手法と使用ゲート数はトレイドオフの関係にあり，LUCAS_2では演算部において，除算については16基数除算器，加算および乗算については配列型積和算回路を構成し，ロード・ストア演算についても，メモリーアドレスは8ビット単位であるがこれを8並列でアクセスするすることにより高速化を行った．

各々の構成図を**図4**，**図5**に示す．ここで，図5においては，8ビット演算の場合を示している．Xilinx社製Vertex（XCV300-5-BG432）デバイスを用いた16基数除算器単体での最大動作周波数は，19.912 MHz，使用ゲート数は，25,178ゲートであり，配列型積和算回路単体での最大動作周波数は，16.286 MHz，使用ゲート数は，36,481ゲートである．

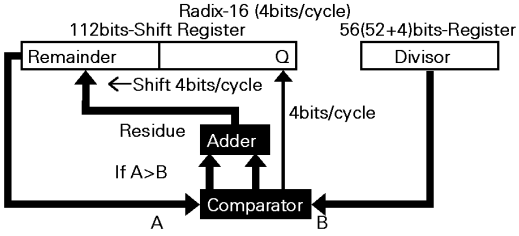


図4 16基数除算器
Fig. 4 Divider hardware (Radix16).

Xilinx社製Vertex（XCV400-5-BG432）デバイスを用いたLUCAS_1システムの制御および専用演算処理部の配置配線図を**図6**に示す．この場合，最大動作周波数はおよそ20 MHz，使用ゲート数はおよそ4万ゲートであった．

4. 性能評価

ここでは，稠密行列による性能評価を行う．これはLUCASの最大演算処理能力を評価するものである．回路シミュレーションにおいて，生成される行列は不規則な疎行列であり，一般的な評価はできない．ベンチマーク回路を使用した数値例による疎行列に対する見積もりは，次節で示される．

4.1 演算クロック評価

性能の評価には種々の方法があるが，ここでは純粋にプログラムの実行演算クロックサイクルの回数により比較を行う．アルゴリズムの見地からの性能評価については文献22)，23)を参照されたい．

4.1.1 理想評価

我々は，将来的にFPGAもCPUと同等の周波数で動作し，FPGAからホストPCの主記憶装置にアクセス可能となると考えており，以後これを理想評価とする．すなわち，理想評価 $= N_{SP}/N_{LU}$，ここで，N_{SP} はSPICEの必要クロックサイクルであり，N_{LU} はLUCASのクロックサイクル（通信を含まない）である．

4.1.2 重み付き評価

現実的評価として，CPUの動作周波数 f_{cpu}，通

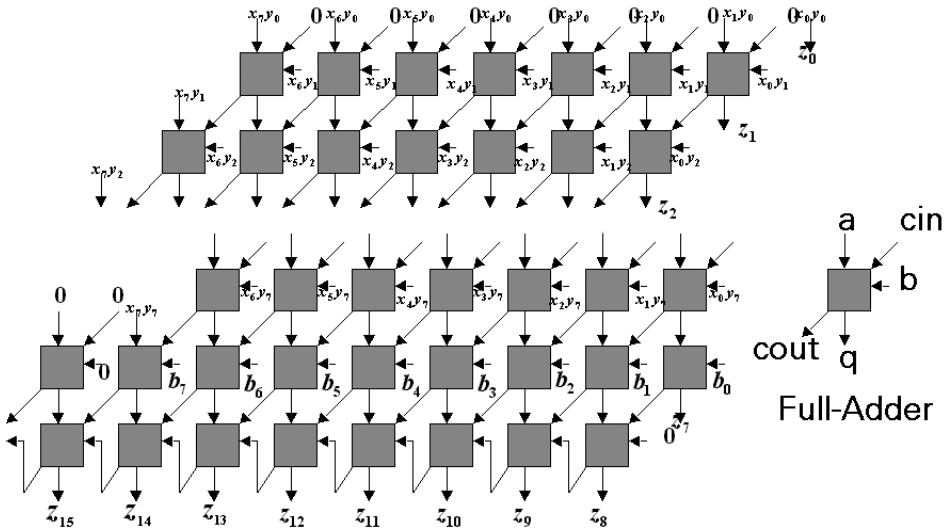


図5 配列型積和算回路
Fig. 5 Array type multiplier-adder hardware.

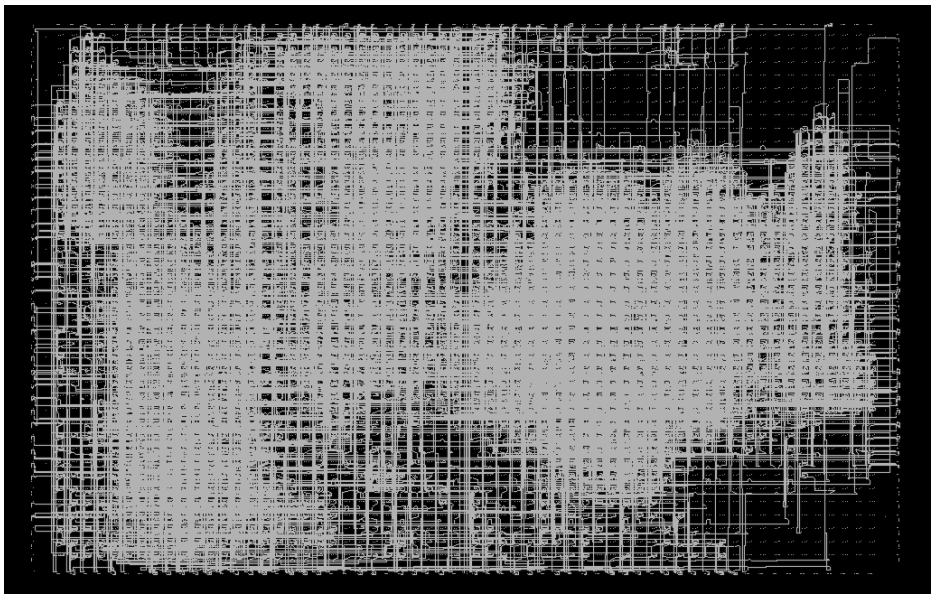


図 6 LUCAS の配置配線図
Fig. 6 Allocation and wiring of LUCAS.

信時の動作周波数 f_{bus}，FPGA を含むアクセラレータボード上の動作周波数 f_{fpga} について，おおよそ $f_{cpu} = 500\,\mathrm{MHz}$，$f_{bus} = 33\,\mathrm{MHz}$，$f_{fpga} = 20\,\mathrm{MHz}$ として，25 : 1.65 : 1 であり，以後，重みとして $w_{cpu} = 1$，$w_{bus} = 15$，$w_{fpga} = 25$ を各クロックサイクルに乗じ評価したものを重み付き評価とする．すなわち，

$$\text{重み付き評価} = \frac{w_{cpu} \cdot N_{SP}}{w_{fpga} \cdot N_{LU} + Nw}.$$

ここで，Nw は通信部に必要な，重み付きのクロックサイクルである．

4.2 評価手順

4.2.1 SPICE

まず SPICE の行列演算部から純粋に LU 分解処理部および代入計算部を抜き出し，Markowitz オーダリング[24]，添え字処理（bi-directional threaded list [25]）等，行列演算アルゴリズムに直接関与しない部分を削除し，より簡単な構造とする．これを新たなモジュールとしてコンパイルを行い，生成オブジェクトを逆アセンブルする．**表 2** に SPICE を逆アセンブルすることにより得られる各オペコードの個数を示す．ここで，clock count はアセンブリ・リファレンス[26]による Intel Pentium Processor のものである．n は行列の次元であり，j*はすべての jump 命令を意味する．

PC による演算ではいわゆる最良値の場合においても，とりうる値が複数ある場合があり，それらを Best case，Worst case と定義し，それぞれ必要な演算クロック回数およびその出現回数を乗じ総和をとると，

$$\text{Best case} : \frac{266}{3}n^3 + \frac{323}{2}n^2 - \frac{31}{6}n - 34, \quad (4)$$

$$\text{Worst case} : \frac{259}{3}n^3 + 158n^2 + \frac{2}{3}n - 34, \quad (5)$$

となる．

表 2 SPICE における演算回数
Table 2 Number of oparations for SPICE.

op	clock count	number
mov	1	$9n^3 + \frac{41}{2}n^2 + \frac{21}{2}n - 16$
sub	1	$2n^3 + \frac{13}{2}n^2 + \frac{5}{2}n - 4$
cmp	1	$\frac{n^3}{3} + 2n^2 + \frac{14}{3}n - 4$
j*	1	$\frac{2}{3}n^3 + \frac{5}{2}n^2 + \frac{29}{6}n - 5$
add	1	$\frac{8}{3}n^3 + \frac{9}{2}n^2 + \frac{5}{6}n - 6$
imul	10	$\frac{20}{3}n^3 + \frac{19}{2}n^2 - \frac{49}{6}n$
lea	1	$\frac{5}{3}n^3 + n^2 - \frac{5}{3}n$
fld	1	$\frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{7}{6}n$
fcomp	4-1	$\frac{n^3}{3} + \frac{n^2}{2} - \frac{5}{6}n$
fnstsw	2	$\frac{n^3}{3} + \frac{n^2}{2} - \frac{5}{6}n$
and	1	$\frac{n^3}{3} + \frac{n^2}{2} - \frac{5}{6}n$
fdiv	39	$\frac{n^2}{2} + \frac{n}{2}$
fstp	2	$\frac{n^3}{3} + n^2 - \frac{n}{3}$
fmul	3-1	$\frac{n^3}{3} + \frac{n^2}{2} - \frac{5}{6}n$
fsubr	3-1	$\frac{n^3}{3} + \frac{n^2}{2} - \frac{5}{6}n$
xor	1	1
cdq	2	n
idiv	46	n

4.2.2 LUCAS

一方，LUCAS の性能は，アクセラレータ上での演算とデータの送受信に必要な通信部分に分けられ，ど

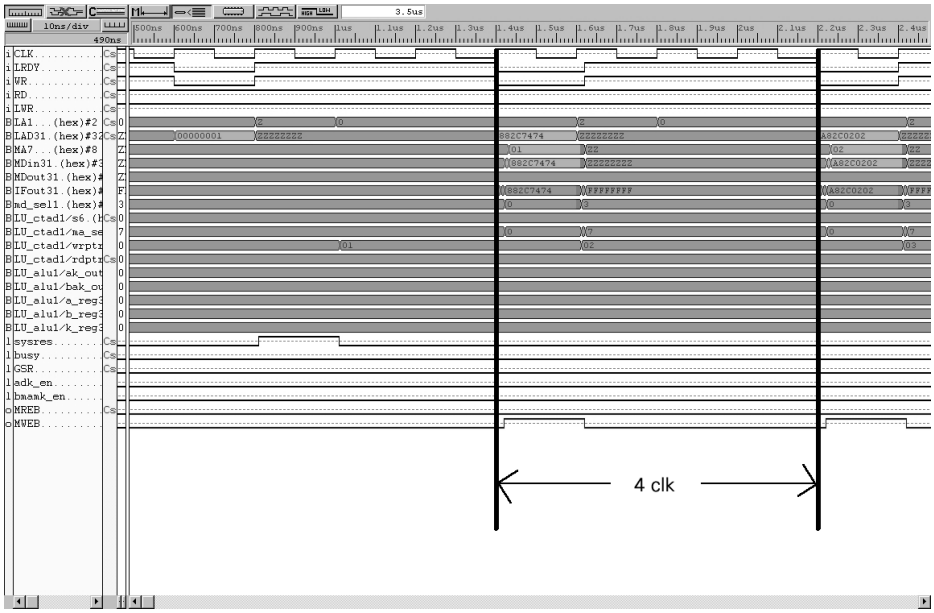


図 7 PCからアクセラレータへの書き込み
Fig. 7 Writing process from PC to board.

ちらもタイミングシミュレーションによって見積もることができる．普通，計算時間の見積もりは（システム時間を含む）実時間ではなくアプリケーションが使用した CPU 時間（ユーザ時間）で行うのと同様に，通信の際のたとえばホスト PC における他のソフト・ハードモジュールでのインタラプトリクエスト（IRQ）等は OS により制御されており，これらは考慮しないこととする．

4.2.2.1 演 算 部

LUCAS による演算量は各ステイトに必要な演算クロックと各ステイト回数を乗じ，総和をとることで得られる．LU 分解，前進消去，後退代入の行列求解に必要な演算数は，LUCAS_1 における理想評価で，

$$\text{LU decomp.} : \frac{97}{3}n^3 - \frac{11}{2}n^2 - \frac{143}{6}n - 3, \quad (6)$$

$$\text{Forward subst.} : 97n^2 - 191n + 94, \quad (7)$$

$$\text{Backward subst.} : 90n^2 - n - 1, \quad (8)$$

となり，全体で，

$$\text{Matrix} : \frac{97}{3}n^3 + \frac{363}{2}n^2 - \frac{1295}{6}n + 90, \quad (9)$$

となる．

4.2.2.2 通 信 部

通信時間は使用するバスの性能に依存する．今回は，PCI32 バスを使用し，通信部は使用する PCI インタフェース LSI の動作タイミング[27]に完全に合わせて設計を行った．PCI ローカルバス（アクセラレータボード内のバス）の通信時タイミングシミュレーション結果を**図 7**，**図 8** に示す．PC–アクセラレータ間の通信には，ほかにホスト側チップセットによるデータ処理[27]，ホストアプリケーション（DanSpice）によるデータ抽出処理（in，out 命令）が必要で，それらを**表 3** にまとめる．ここで，( ) 内は重み付きのクロックサイクルである．転送に必要な要素数は，

- 64 ビット入力：n（計算後のベクトル）
- 64 ビット出力：$n \times (n+1)$（行列要素と励起ベクトル）

これより，重み付きの全通信時間は，

$$\begin{aligned}&(100 + 30 + 6) \times 2 \times n \\ &\quad + (100 + 60 + 10) \times 2 \times n \times (n+1) \\ &\quad = 340n^2 + 612n \end{aligned} \quad (10)$$

となる．

4.3 SPICE vs. LUCAS

式 (4)，(5)，(9) および LUCAS_2 の演算クロック数で式 (9) を計算し直したもの，また双方に重みを掛け，全通信時間の式 (10) を加えたものを**図 9** にプロットする．

LUCAS の最大性能は理想評価で，1 回の行列求解あたり SPICE 比で LUCAS_1 でおよそ 2.7 倍，LUCAS_2 でおよそ 24 倍である．重み付き評価では，LUCAS_1 でおよそ 0.11 倍，LUCAS_2 で，およそ 0.92 倍である．重み付き評価の性能低下は，主として現段階の FPGA 上の専用演算器の動作周波数が MPU に

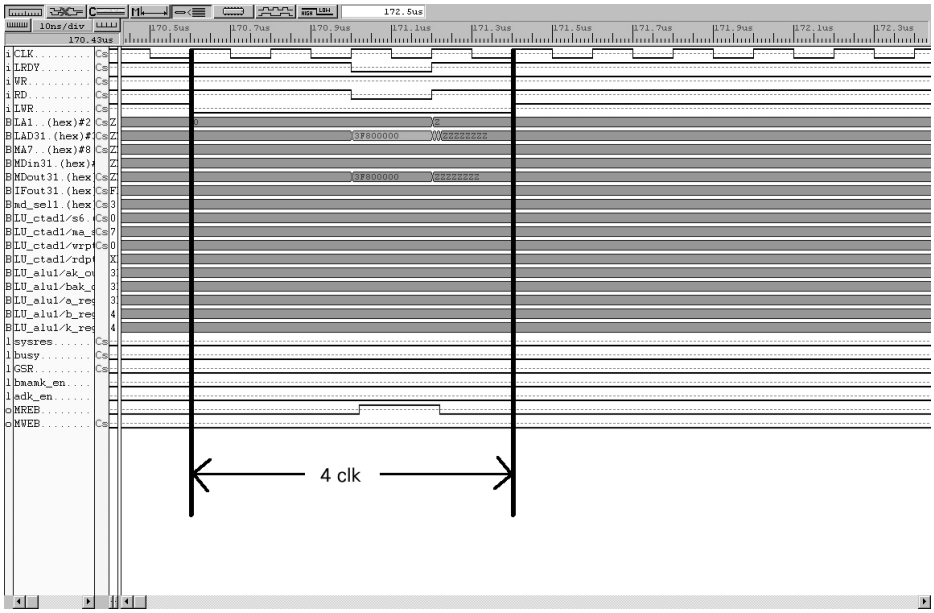


図 8 アクセラレータから PC への読み込み
Fig. 8 Reading process from board to PC.

表 3 32 ビット・データの転送に必要なクロックサイクル
Table 3 Clock cycles needed for communication of a 32-bit data.

	PCI9052 ($w_{fpga} = 25$)	chip-set ($w_{bus} = 15$)	Host App. ($w_{cpu} = 1$)
in	4 (100)	2 (30)	6 (6)
out	4 (100)	4 (60)	10 (10)

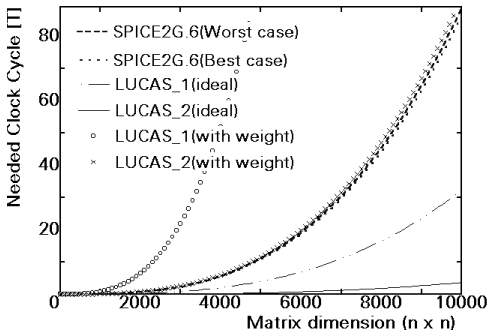
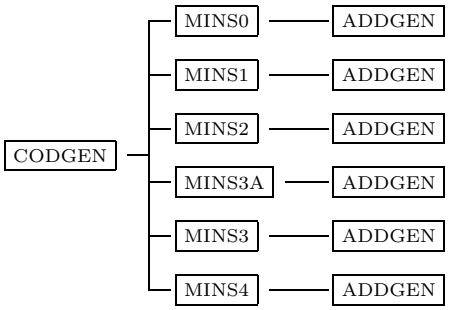


図 9 クロックサイクルの見積もり
Fig. 9 Estimation of clock cycles needed.

比べ遅いことに起因し，構成される専用演算器の設計に課題を残している．



cf. Code Generator, Machine Instraction, Address Generator

図 10 コード生成部の Tree 構造
Fig. 10 Tree of codgen.

5. LUCAS を使用したコード生成法アルゴリズム

5.1 SPICE におけるコード生成法

SPICE におけるコード生成法の基本構造は**図 10** のとおりで，CODGEN は FORTRAN，ほかはアセンブリ言語で記述され関数型である．実行モジュールは，CODEXC という実行時に生成されるモジュールである．ここで，ADDGEN は元々IBM S360/370 用に開発されたパッケージにおいて必要であったオフセットアドレス生成用であり，PC への適応には必要では

ない．MINS0，MINS4はモジュールとしてのセットアップ部分であり，MINS1からMINS3が生成コードの本体となる．MINS1は許容最小ピボットPIVTOL処理部であり，MINS2，MINS3は式(2)，(3)に対応する処理部である．現在のPC環境での一般的なOS（Windows）では，プロテクトモードと呼ばれるCPU（一般にPCではMPUと呼ぶ）の動作モード下で実行されており，通常自己改変コード（Self-Modifying Code: SMC）の一種である実行時コード生成は，普通は推奨されないプログラミング技法であり，単純にアセンブリを書き直すだけでは一般保護違反を引き起こすことになる．最も簡単な解決法は，一度解析前にソースコードを生成し，コンパイルして使用するコンパイル方式あるいはDLLを実行時に生成し，行列求解時にダイナミック・リンクする方法であるが，これだとコード生成法の高速性をスポイルする結果となる．テクニカルな方法，たとえばページモードを変更する関数を使ったり，アセンブラのコンパイルオプションにより書き込み可能な実行ファイルを生成する等により動的なコード生成が可能ではある[28]．

5.2 LUCASを使用したコード生成法

3.2.2.2項で説明したようにLUCASは式(2)，(3)に対応した2つのALUを持ち，それゆえこの2つのALUを切り替えることにより一連の行列求解処理を行うことができる．LUCAS（以後，コード生成法版のLUCASをLUCAS_cと呼ぶ．LUCAS_cは前述のLUCAS_2と同じ演算部を持ち，制御部のみが異なる）を使用するためのコードデータを**表4**に示す．すべてのコードデータは32ビット×3（#1から#3までのデータセット）を1組に区切られ，この1組1組が実行コードセットとして認識される．アドレスは正数であり，実行コードの#1データの正負（すなわち，最上位の1ビットのみ）で判別され，高速なデータマシンとして構成することができる．実行のアルゴリズムを**図11**に示す．第1ビットの比較は回路的には1ビットコンパレータで構成され，LUCAS_cの制御ブロックの大幅な簡略化に寄与することとなる．

動作速度は結局ALUの性能に強く依存し，基本的な四則演算回路を自由に構成できるLUCAS_cの優位となる．さらに，生成コード量に関してもLUCAS_cを使用する場合，各演算あたり12バイトで構成できるのに対し，ソフトウエア実行時コード生成では，Intel Pentium Processorの場合，1演算あたりMINS0およびMINS4それぞれ11バイト，MINS2に対し18バイト，MINS3に対し24バイト[28]と，LUCASを使用したコード生成法が有利となる．また，MINS1に対してもSPICEの場合62バイト必要となるが，LUCAS_cの場合は，除算の前での比較を行うことにより特にコードとして生成する必要はない．

表4 LUCAS_c上でのコードデータ構造
Table 4 Code data structure for LUCAS_c.

	#1data	#2data	#3data
ALU1	−1	Addr. of a	Addr. of k
ALU2	Addr. of b	Addr. of a	Addr. of k

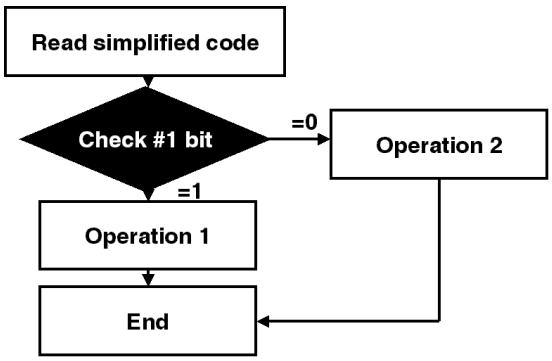


図11 LUCAS_cの生成コード実行フロー
Fig. 11 Flow for code generation on LUCAS_c.

5.2.1 数値例

ここでは，ベンチマーク回路を使用したSPICEのコード生成法による演算クロックサイクルおよびLUCAS_cによるプロトタイプシステムでの性能評価を行う．ベンチマーク回路は，**表5**に示す，

(1) circuit_A：CMOSリングオシレータ，
(2) circuit_B：CMOSシフトレジスタ，
(3) circuit_C：4つの積分器と差動増幅器からなるアナログ補正回路，

を使用した．ここで，nは生成行列の次元，nttarはフィルインを含む行列の非零要素数，perspaはスパーシティ（%）を示し，ALU1は式(2)の実行回数，ALU2は式(3)の実行回数，ifは許容最小ピボットPIVTOL処理回数である．CL_は生成コード長（バイト）である．生成コード長は，明らかにLUCAS_cの方が少なく，およそ半分である．

ベンチマーク回路の必要クロックサイクル数の見積りを**表6**に，LUCAS_cの通信に必要なクロックサイクルを**表7**に示す．SPICEの必要クロックサイクルは，我々の開発したPC用コード生成ルーチン[28]から算出し，LUCAS_cのものは前述の評価方法と同様に，各ステイトに必要な演算クロックと各ステイト回数を乗じ，総和をとることで得た．

ここで，SPICEでは，上から各演算名，演算あたりの必要クロックサイクル，1回あたりの行列求解に必

表 5 ベンチマーク回路のアカウント情報
Table 5 Account information of benchmark circuits.

	n	nttar	perspa	ALU1	ALU2	if	CL_SPICE	CL_LUCAS_c
circuit_A	38	214	85.180	126	310	36	11962	5232
circuit_B	231	1681	96.850	997	2161	229	83472	37896
circuit_C	1180	6779	99.513	3989	11970	1178	432140	191508

表 6 ベンチマーク回路の必要クロックサイクル
Table 6 Clock cycles needed of benchmark circuits.

	SPICE ($w_{cpu}=1$) (lvlcod=2)				LUCAS_c matrix ($w_{fpga}=25$)			
operation	ALU1	ALU2	if	TOTAL	ALU1	ALU2	CNTL	TOTAL
clock cycle/op	42	9, 5	12, 7	-	21+4	5+4	5	-
circuit_A	5292	2790-1550	432-252	8514-7094	3150	2790	5	5945
circuit_B	41874	19449-10805	2748-1603	64071-54282	24925	19449	5	44379
circuit_C	167538	107730-59850	14136-8246	289404-235634	99725	107730	5	207460

表 7 LUCAS_c の通信に必要なクロックサイクル
Table 7 Clock cycles needed for communication of LUCAS_c.

	LUCAS_c								
	PCI9052 ($w_{fpga}=25$)			chip-set ($w_{bus}=15$)			Host App. ($w_{cpu}=1$)		
operation	out-64bit-in		32bit-out	out-64bit-in		32bit-out	out-64bit-in		32bit-out
clock cycle/op	4×2	4×2	4	4×2	2×2	4	10×2	6×2	10
circuit_A	2016	304	5232	2016	152	5232	5040	456	13080
circuit_B	15296	1848	37896	15296	924	37896	38240	2772	94740
circuit_C	63670	9440	191508	63670	4720	191508	159180	14160	478770

要なクロックサイクルを示す．前述のように，PC による演算ではとりうる値が複数あるため，性能評価結果の値には範囲がある．LUCAS_c では行列求解部と通信部を分けて見積もっている．また，LUCAS_c で ALU1 および ALU2 には，メモリアドレスのデコードに必要な 4 clock が足されており，CNTL は 1 回の行列求解時に 1 度だけ実行される第 1 コードデータセット（32 ビット × 3）の読み込み部である．第 2 コードデータセット以降は，ALU 動作時にプリフェッチされる．なお表 7 における，LUCAS_c での 64 ビット出力は，演算に必要な励起ベクトルと行列の非零要素の（n + nttar）個であり，64 ビット入力は計算されたベクトルの n 個である．64 ビット入出力は 32 ビットに分割されて転送されるので，演算あたりの必要クロックサイクルは 2 倍されている．

32 ビットのデータ（ALU1+ALU2）個はすべてコードデータ部であり，従来必要とされた行列の非零要素のアドレスデータはホスト PC 側でアクセラレータボードのローカルメモリアドレスデータに変換され，コード中に埋め込まれるため，転送は行わない．また，このホスト PC によるアドレス変換は SPICE におけるコード生成法のアドレス変換とほぼ同じであるため（SPICE でもメモリ上に変数のアドレスをバイト単位でストアするためアドレス変換を行う[28]），今回の比較では考慮しなかった．

5.2.2 ベンチマーク回路による性能評価

LUCAS_c の SPICE を基準とした性能向上率は，理想評価について，circuit_A：1.43-1.19，circuit_B：1.44-1.22，circuit_C：1.39-1.14 となる．重み（$w_{fpga}=25$，$w_{bus}=15$，$w_{cpu}=1$）を乗じた重み付き評価は，circuit_A：0.0182-0.0152，circuit_B：0.0258-0.0218，circuit_C：0.0250-0.0204 である．SPICE による評価では，MPU のキャッシュヒット率を 100%として行っているが，コード生成法を使用した場合，キャッシュミスが増大することが予想され，また，より高性能な PCI-X バス（データ幅：64 ビット，動作周波数：133 MHz）の実用化が迫っており，これは主メモリへのアクセスに迫る速度のバスであり，将来は理想評価に近い性能を達成することが期待できる．

6. ま と め

本論文では，将来のコンピューティングシステムの一提案として，FPGA を用いたハードウエア・ソフトウエア協調コンピューティングシステムを構築し，回路シミュレータ DanSpice とその FPGA 定義データにより実行時に定義される LUCAS についての性能評価を行った．この提案により，従来のコンピューティングシステムでは不可能であったハードウエア・アー

キテクチャの変更，すなわちアプリケーションごとに最適化された演算処理装置を構成することが既存の環境下で使用可能となる．

回路シミュレーションの高速化に関する最も重要な要素は，デバイスモデル評価ではなく，NR アルゴリズムの各反復で繰り返される線形方程式の求解であり，LUCAS を使用することで，精度はまったく落とさずに SPICE のソフトウエア処理と比較し，行列求解 1 回あたり，おおよそ 20 倍の最大演算能力を達成できる．これは，稠密行列に限られるのではなく，理論上は疎行列でも同様の高速化が図られるが，より回路シミュレータに適した求解法であるコード生成法の提案を行った．LUCAS を使用したコード生成法はそのアルゴリズムの簡素さゆえに，従来法より，生成コード長が短く，より安全に PC 上で実行が可能である．コード生成法を使用したベンチマーク回路による比較では，おおよそ 1.3 倍程度の高速化が達成可能なことが示された．

しかしながら，重み付き評価により明らかなように，現状でのホスト CPU に比べかなり低速なバス，FPGA の動作周波数下では，我々のプロトタイプシステムは意味をなさない．すなわちより高速なバス，FPGA デバイスが必要である．逆に提案システムを生かす環境が整えば，アプリケーションごとにそれ自身に必要な演算器を構成することが可能となる．

アプリケーション実行時において，使用下にある FPGA 内の未使用領域に新たな演算器を構成することが可能になれば，自己改変ハードウエアとしてのコンピューティングシステムが可能になる．これはアプリケーションが外部からの入力により，最適なハードウエア（並列マシンを含む）を構成することを意味し，連想あるいは記憶および学習能力をハードウエアが持つことにつながる．我々の考える未来のコンピューティングシステムにおいて，CPU は入出力データ管理，最適ハードウエアの自動合成およびタスクのスケジューリング機能のみを持ち，個々のアプリケーションはすべて自分自身をハードウエア化する定義ファイルを持つ．

今後の課題は，プロトタイプボードの完成を急ぎ，さらに将来のオンチップ並列処理に必要なデバイスモデル評価部のハードウエア化および演算パイプラインを含むアクセラレータ上での細粒度並列アルゴリズムの開発を進めていく予定である．

参 考 文 献

1) Nagel, W.: SPICE2, A computer program to simulate semiconductor circuit, Memo ERL-M250, University of California, Berkeley (1975).
2) Shima, T., Sugawara, T., Moriyami, S. and Yamada, H.: Three-dimensional table lookup MOSFET model for precise circuit simulation, *IEEE Journal of Solid-State Circuits*, Vol.SC-17, pp.449–453 (1982).
3) Subramaniam, P.: Modeling MOS VLSI circuits for transient analysis, *IEEE Journal of Solid-State Circuits*, Vol.SC-21, pp.276–285 (1986).
4) Rofougaran, A. and Abibi, A.A.: A table lookup FET model for accurate analog circuit simulation, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.12, No.2, pp.324–335 (1993).
5) Sheu, B.J., Scharfetter, D.L., Ko, P.-K. and Jeng, M.-C.: BSIM: Berkeley short-channel IGFET model for MOS transistors, *IEEE Journal of Solid-State Circuits*, Vol.SC-22, pp.308–331 (1984).
6) コネリー，J.A.，チェイ，P.（著），青木 均（訳）：SPICE による回路設計，トッパン，東京 (1994).
7) Newton, A.R. and Sangiovanni-Vincentelli, A.L.: Relaxation-based electrical simulations, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.CAD-3, pp.308–331 (1984).
8) Antognetti, P., Pederson, D.O. and de Man, H.: *Computer Design Aids for VLSI Circuits*, pp.107–110, Martinus Nijhoff Publishers (1984).
9) Sakallah, K.A. and Director, S.W.: SAMSON2: An event driven VLSI circuit simulator, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.CAD-4, pp.668–684 (1985).
10) Saleh, R.A. and Newton, A.R.: The exploitation of latency and multirate behavior using nonlinear relaxation, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.8, pp.1286–1298 (1989).
11) Gustavson, F.G., Liniger, W. and Willoughby, R.: Symbolic generation of an optimal crout algorithm for sparse systems of linear equations, *Journal of the Association for Computing Machinery*, Vol.17, No.1, pp.87–109 (1970).
12) 前川仁孝，田村光雄，中山 功，吉成泰彦，笠原博徳：直接法を用いた電子回路シミュレーションの近細粒度並列処理，電気学会論文誌（C），Vol.114, pp.579–587 (1994).
13) Powerspice simulates circuits faster and more accurately, *Electronics*, Vol.58, No.34, pp.50–51

(1985).
14) 中田登志之，田辺記生，梶原信樹，松下　智，小野塚裕美，浅野由裕，小池誠彦：並列回路シミュレーションマシン Cenju，情報処理，Vol.31, No.5, pp.593–601 (1990).
15) 鹿毛哲郎，大石優子，石井光雄：ミニコンとアレイプロセッサによる汎用回路解析プログラム（FINAP-AP）の開発，電子情報通信学会技術研究報告，Vol.CAS84-113, pp.49–55 (1984).
16) 田辺　昇，石坂賢一，村越英樹，泉谷昭二，土肥康孝：並列パイプライン式疎行列用専用計算機 RAMP の構成，電子情報通信学会論文誌（D），Vol.J71-D, No.10, pp.1939–1948 (1988).
17) 浅井英樹，浅井光男，田中　衛：大規模スパース行列の LU 分解専用並列計算機，電子情報通信学会論文誌（D），Vol.J69-D, No.7, pp.1044–1053 (1986).
18) 田辺　昇，土肥康孝：連想スイッチによる疎行列用計算機の構成，電子情報通信学会論文誌（D），Vol.J70-D, No.12, pp.2393–2401 (1987).
19) Lewis, D.M.: A compiled-code hardware accelerator for circuit simulation, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol.11, No.5, pp.555–565 (1992).
20) Lewis, D.M.: html ドキュメント，www 上で公開.（http://www.eecg.toronto.edu/ lewis/）
21) Intel Corporation: インテル・アーキテクチャー・ソフトウエア・ディベロッパーズ・マニュアル.（http://www.intel.co.jp より入手可能）
22) 小国　力，後　保範，西方政春，長堀文子：直接解法による連立 1 次方程式のコンピュータ解法の特性解析，情報処理学会論文誌，Vol.25, No.5, pp.804–812 (1984).
23) 八木浩行，檀　良：ハードウエアを用いた PC 環境での回路シミュレータの性能評価，1999 年電子情報通信学会総合大会講演論文集，基礎・境界，p.13 (1999).
24) Tewarson, R.P.: *Sparse matrices*, pp.87–90, Academic press (1973).
25) McCalla, W.J.: *Fundamentals of computer-aided circuit simulation*, pp.37–45, Kluwer academic publishers (1988).
26) Microsoft Corporation: Microsoft MASM Reference および PENTIUM.TXT.
27) PLX Technology: *PCI 9052 Data Sheet* (1997).
28) 八木浩行，檀　良：PC 環境での回路シミュレーション用実行時コード生成法，電子情報通信学会論文誌（A），Vol.J83-A, No.4, pp.444–446 (2000).

(平成 11 年 9 月 10 日受付)
(平成 12 年 2 月 4 日採録)

八木　浩行（学生会員）

平成 6 年法政大学工学部電気工学科電気電子専攻卒業．平成 8 年同大学院修士課程修了．同年同大学院博士課程入学，主として集積回路 CAD の研究に従事．電子情報通信学会学生会員．

檀　　良（正会員）

昭和 37 年東京大学工学部電子工学科卒業．昭和 39 年同大学院修士課程修了．昭和 43 年同大学院博士課程修了．昭和 44 年から 47 年にかけて東芝半導体事業部研究開発．昭和 47 年ベトナムサイゴン国立大学教授．昭和 48 年から 50 年にかけて同サイゴン工科大学学長．昭和 51 年東芝総合研究所主任研究員．昭和 58 年より法政大学工学部教授．主として半導体デバイスモデリング，集積回路 CAD の研究に従事．工学博士．電気学会，応用物理学会，電子情報通信学会，日本シミュレーション学会，Senior IEEE 各会員．