論理シミュレーションマシンのアーキテクチャ†

小池誠彦†† 大森健児†† 佐々木徹†††

本論文はCPU，マイクロプログラムメモリ，キャッシュメモリ，メインメモリを含めた大型計算機の装置全体の論理シミュレーションを高速に実行する専用マシンのアーキテクチャについてその特徴および設計思想に重点をおいて論じている．専用マシンは，数十ゲートで構成される論理的にまとまりのある“ブロック”を単位として用いる．専用マシンのアーキテクチャは高速化および大容量化を実現するために，①32台のマルチプロセッサ構成により負荷分散を図る，②各プロセッサをプログラム制御でなくすべてハードウェアにより実現しパイプライン処理を行う，③論理演算を書換え可能なメモリを用いてハードウェアで実現する，等の特徴を備えている．その結果，32台の専用プロセッサを用いることにより150万ゲートからなる装置を毎秒2.6億ゲートシミュレーションの速度で処理することが可能となる．これは従来のソフトウェアによるシミュレーションの数千から数万倍の処理スピードをもち，大型計算機等の大規模な装置の論理シミュレーションにも十分な性能が得られる．

1. まえがき

論理シミュレーションはLSIや大規模システムの開発に欠かせないツールである．システム規模がますます大きくなっている今日，設計時に混入した誤りを発見することがいっそう困難になってきている．また，LSI完成後の誤りの存在はチップの再製作による費用の増大と開発の長期化をもたらしている．したがって，装置開発のできるだけ早い時期に設計の誤りをなくすことが必要である．従来，論理シミュレーションは大型計算機により逐次的に行われていた．しかし，LSIの目ざましい発達によるシミュレーション規模の増大のため，いくらその時点での最新の超大型計算機をもってきたとしても，その処理能力では満足のいく論理シミュレーションができなくなってきている．

このような技術的隘路を解決するために論理シミュレーション専用マシンによりシステム全体の論理シミュレーションを可能にする方式を考えることとした．この専用マシンの出現によりシミュレーション時間が極端に短縮されることにより次のような効果がある．システム全体の論理シミュレーションが可能になる．いままで一括処理でしか行われなかったシミュレーション処理が高速化されるため，設計者は論理の検証を行いながら会話的に設計を進めることが可能となる．いろいろな場合についてテストすることができるので設計時に見おとしていた仕様の誤りやボトルネック箇所を容易に発見することができるようになる．シミュレーション結果を設計に反映させることによりシステムの最適化を図ることが可能となる．

高速な専用マシンを実現するために論理回路自体がもつ並列性と処理アルゴリズムがもつ並列性を利用することが考えられる．回路を構成するゲート群は自律的に同時動作している．また，処理アルゴリズムにおいてもゲートの論理演算やゲートの入出力状態値の更新といった処理はオーバラップして同時処理可能である．筆者らはこの二つの並列性を生かした専用のシミュレーションマシンを開発した[1),3),4)]．マシンは従来のゲートレベルシミュレーション方式[5)]にかわり，数十ゲートで構成される論理的にまとまりのある“ブロック”を単位として処理する方式[2)]を考えた．パイプライン方式により，このブロックを処理する専用のプロセッサを並列に32台並べた専用マシンを考えた．この専用マシンでは，従来のソフトウェアシミュレータ[7)]に比べ数千から数万倍の高速化を達成することが可能である．設計にあたってはとくに次の点を考慮した．

（1）シミュレーション処理をすべてハードウェア化するとともに並列同時処理を行うことにより高速化を図る．

（2）大容量化を図り大型計算機の装置シミュレーションを可能とする．

（3）シミュレーション対象になっている装置の上で動くソフトウェアを専用マシンの上でも動かすことができるようにするため，メインメモリ，キャッシュ，ファームウェアを含めたメモリ系も専用マシンの構成要素に含めることとする．

† Architecture of the High Speed Logic Simulation Machine by NOBUHIKO KOIKE, KENJI OHMORI (C&C Systems Research Laboratories, Nippon Electric Co., Ltd.) and TOHRU SASAKI (Computer Engineering Division, Nippon Electric Co., Ltd.).

†† 日本電気(株)C＆C システム研究所
††† 日本電気(株)コンピュータ技術本部

本論文では専用マシンアーキテクチャの基本となるブロックレベルシミュレーション方式と並列／パイプライン処理方式および予想性能について述べる.

2. 論理シミュレーションの高速化

ここでは高速なシミュレーション処理を行うための処理アルゴリズムを検討し，従来のシミュレータの問題点を解決する新しい処理方式を考える.

2.1 シミュレーションアルゴリズム

現在の大規模システムでは同期式回路の設計手法が大勢を占めている．そこで，処理を簡略化し高速処理を実現するために，本論理シミュレータでは，対象とする回路を同期形の組合せ回路で構成され非同期的なフィードバックを含まない回路に限定することとした．このように限定すれば，それぞれのゲートの遅れの扱いが不要となりレベルソートならびにイベント駆動の手法に基づく効率のよいシミュレーション処理が可能となる．さらにシミュレーション単位をゲートの集合体であるブロックにおくことにより数十個のゲートを同時に処理することができる．レベルソートはブロックを信号伝播のレベル順に番号づけし，この順番にブロックの評価を行う手法である．ブロックのすべての入力の変化が終わってから1度だけ論理演算を行うのでシミュレーション回数を減少させることができる．さらにイベント駆動の手法によればブロックの入力に変化があったブロックだけについて論理演算処理を行うのでさらに処理量を減少させることができる．大型計算機等のシミュレーション対象回路では，1クロックのレベル数はたかだか 10 レベルである．しかし，タイムホイール等を用いた従来のシミュレーション方式[6]では，1クロックを最低でも 100 以上のタイムスロットに分けて処理するのが普通である．各レベルあるいはスロットごとに同期してシミュレーション処理をする必要があるのでレベルソートの方式は，従来のタイムホイールによる方式より制御が単純であり，同期処理の回数が少ない利点がある．しかも，一つのレベルに同時処理可能なブロックが多く含まれるので高い並列性を有している.

2.2 高速処理の実現

ソフトウェアによるシミュレーションでは次の2点で処理性能に限界があり高速化が望めないと考える.

（1）ゲート評価の逐次処理

論理回路を構成する各ゲートは自律的に並列動作をしている．しかし，ソフトウェアではゲートを一つずつ逐次的にしか処理できない．入出力状態値の更新，イベントの取出し，ゲート論理演算およびファンアウト処理はオーバラップして並列処理可能であるが，ソフトウェアではこれを生かすことができない.

（2）テーブル／リスト処理のオーバヘッド

各ゲートの入出力値，論理関数，ファンアウト先が表形式で管理されるため，イベントごとにテーブルサーチ，リスト処理，ビット処理等シミュレーションに固有でない処理のために多くの CPU 時間が費やされてしまう.

ソフトウェアシミュレーションをするために生じた無駄な処理を取り除き，また，ソフトウェアシミュレーションでは不可能であった並列処理を生かすために以下の方策を考えた.

（1）専用ハードウェアによるパイプライン処理

シミュレーション処理過程に含まれる途中の各処理を別個の独立したハードウェアで実現する．同時に複数のブロックをハードウェア群がオーバラップしてパイプライン式に並列に処理にあたる．テーブル処理やビット処理も専用に開発した回路を用いて高速化を図る.

（2）ブロック論理演算のハードウェア処理

ブロックの入力値から出力値を求める論理演算処理をダイナミックゲートアレイと呼ぶ専用ハードウェアを用いて高速に実行する．ダイナミックゲートアレイはブロックを構成するゲート群を信号伝播順に従い複数のゲートを同時に処理しながら最終的にブロックの出力を得る.

（3）並列プロセッサ

複数のプロセッサでブロックを並列に処理する．ブロックの処理に必要な情報はすべてそれぞれのプロセッサに分散することによりプロセッサ間での競合が起こらないようにした．これによりプロセッサの数にほぼ比例して性能を向上させることができる.

本シミュレータではブロック論理演算のハードウェア化により約2桁，さらに並列処理とパイプライン処理により約2桁のスピードゲインが得られる．したがって合計で3桁から4桁の高速化が実現される.

3. 論理シミュレーションマシンの構成

高速化の方策に従い，マルチプロセッサ技術を用いて専用マシンとして開発したのが超高速シミュレータである．ここでは，超高速シミュレータのシステム構成について述べる.

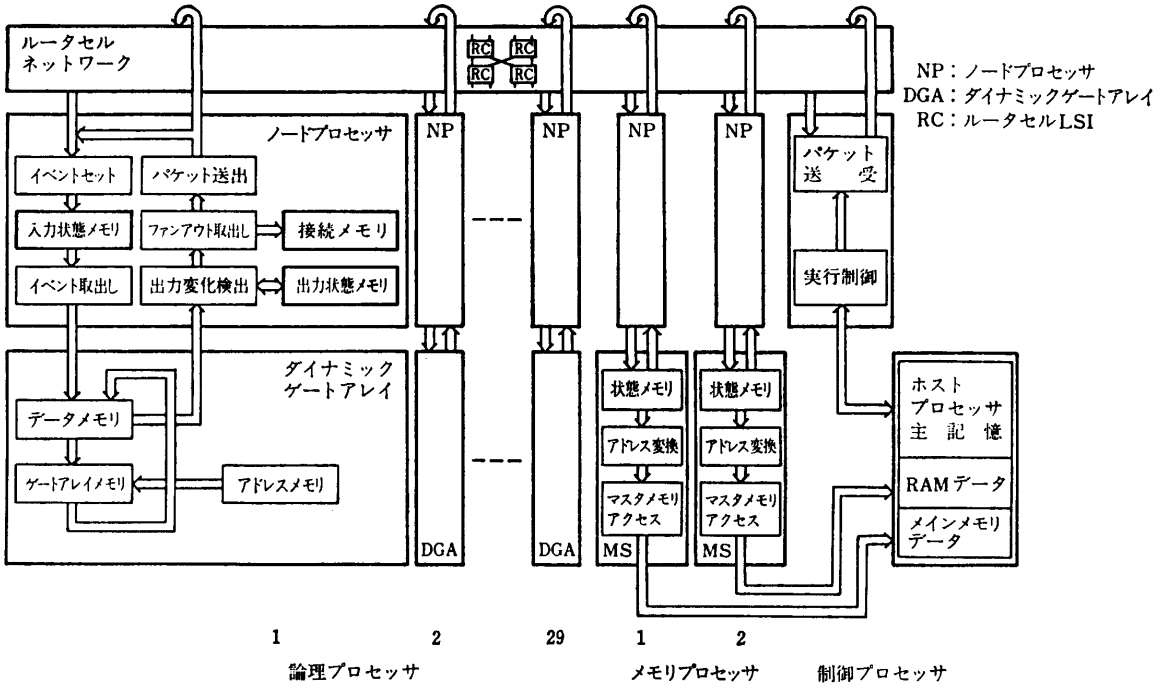


図 1 超高速シミュレータの構成
Fig. 1 High speed logic simulation machine blockdiagram.

3.1 システム構成

超高速シミュレータは**図1**に示すように 29 台の論理プロセッサ，2台のメモリプロセッサおよびコントロールプロセッサからなる集合体であり，プロセッサ群を専用のルータセルネットワークで結合する．論理プロセッサは組合せ回路系のブロック群を分担しメモリプロセッサはメモリ系のブロックを分担する．コントロールプロセッサはホストとなるミニコンピュータとシミュレータ本体とを結び，シミュレーション実行制御およびデータ転送を行う．

論理プロセッサはブロックの入出力状態管理およびファンアウト処理を行うノードプロセッサとブロックの論理演算を行うダイナミックゲートアレイからなる．メモリプロセッサはノードプロセッサとメモリブロックの処理を行うメモリシミュレータからなる．それぞれのプロセッサにシミュレーションに必要なブロックの入力状態表，出力状態表，接続メモリおよびブロックの論理演算部をもたせている．これはプロセッサ間でのメモリアクセスによる競合の回避と独立した並列シミュレーション処理実行を可能にする．出力状態変化に伴うイベントはパケットの形でルータセルネットワークを介して相手先のプロセッサに転送する．

3.2 ノードプロセッサ

ノードプロセッサは図1に示すようにブロックのシミュレーションの一連の処理を別個のハードウェアで実現し，並列にパイプライン式に処理を行う．一つのノードプロセッサは 32 入力，32 出力をもつブロックを最大1,024個まで収容可能とした．ノードプロセッサは各ブロックに対応して次の3種類のメモリをもつ．

（1）入力状態メモリ：1 kW×44 bit の容量をもち，各ブロックの入力ピン状態 (32 bit)，ブロック種 (6 bit)，およびイベントフラグ類 (6 bit) を記憶する．

（2）出力状態メモリ：1 kW×32 bit の容量をもち，各ブロックの出力状態値を記憶する．

（3）接続メモリ：64 kW×26 bit の容量をもち，各ブロックの出力ピンに対するファンアウト先のプロセッサ番号，ブロック番号，ピン番号をリスト形式にもつ．

3.3 ダイナミックゲートアレイ

ダイナミックゲートアレイは組合せ回路で構成されるブロックの論理演算処理を行うハードウェアである．ノードプロセッサからブロックの入力ピン状態とブロックの種類を入力し，出力ピン状態を出力する．ブロックを構成するゲート群を信号の伝播順に従い，

一度に 16 ゲートずつ論理演算を行いながら最終の出力ピン状態を求める．一つのダイナミックゲートアレイで 64 種のブロックあるいは 8,000 ゲートまで収容可能とした．

3.4 メモリシミュレータ

メモリシミュレータはメモリ系のブロックを扱う専用のハードウェアであり，1,024 個までのメモリブロックあるいは 2 MB までの容量を扱うこととした．メモリシミュレータが扱うメモリデータはホスト計算機の主記憶のある領域に記憶される．この領域は論理シミュレーションマシンとホスト計算機との間で共有する．メモリシミュレータはメモリブロックのアクセス状態を検出すると，主記憶へアクセスし読出しあるいは書込みを行う．

3.5 コントロールプロセッサ／ホスト計算機

ホスト計算機はコントロールプロセッサを介し，論理シミュレーションマシンへシミュレーションモデルのロードあるいは論理シミュレーションマシンからのシミュレーション結果のセーブを行う．シミュレーション中はコントロールプロセッサがすべて処理にあたり，例外事象が起きたときのみホスト計算機が介入する．

4. ブロックレベルシミュレーション方式

本シミュレーションマシンの処理単位に採用したブロックレベルシミュレーション方式の特徴とハードウェアによる実現法について検討する．

4.1 ゲートレベルとブロックレベルの比較

シミュレーション単位の設定によって論理シミュレーションマシンの構成および性能は大きな影響を受ける．論理シミュレーションマシンではシミュレーションの単位としてブロックを考えたが，このブロックは数十から数百ゲートを含む一つのまとまった論理回路である．一つのブロックは通常，大型計算機に用いられる IC あるいは LSI のビルディングブロックに対応する．ゲートレベルとブロックレベルをいくつかの項目について比較したものが**表 1** である．ブロックレベルの利点をまとめると次のとおりである．

（1）シミュレーション回数が少ない．

（2）実回路との整合性がよい：実 IC あるいは LSI のビルディングブロックとほぼ 1 対 1 に対応する．

（3）ファンアウト処理が少ない：ブロック内部の結線のファンアウト処理は不要である．

（4）状態表へのアクセスが少ない：複数ゲート分の状態を一括してアクセスする．

表 1 ゲートレベルとブロックレベルの比較
Table 1 Comparison of gate level simulation and block level simulation.

	ゲートレベル	ブロックレベル
論理ユニット数（システム当り）	10^6～10^7	2×10^4～4×10^4
シミュレーション数（クロック当り）	10^6	10^4
種類	≦10	10^2～10^3
状態表アクセス数（クロック当り）	10^6	10^4
論理ユニットの処理時間	数十 ns～数百 ns	数百 ns～数十 μs

また，ブロックレベルシミュレーションの問題点をまとめると次のとおりである．

（A）ブロックの種類が多い：ゲートレベルでは数種類であるのに対し，ブロックレベルでは数百種類にもなる．

（B）ブロックの種類がシミュレーション対象ごとに変化する．

（C）ブロックの評価に時間がかかる．

（D）内部状態を含むブロックの扱いが必要：ブロック内部にフリップフロップ等の状態をもつ素子があると出力値が入力値から一意的に求めることができない．

本シミュレータでは上記の問題を解決するために二つのハードウェアモジュール：ダイナミックゲートアレイとメモリノードシミュレータを開発しシステムに採用した．

4.2 ダイナミックゲートアレイ

ダイナミックゲートアレイは組合せ回路で構成されたブロックの論理機能を実現するものである．一種のデコーダとして働く RAM を用いてそれぞれのゲートの論理演算を行う．ゲートの論理機能を RAM のメモリ領域にビットパターンに展開しておく．**図 2** に

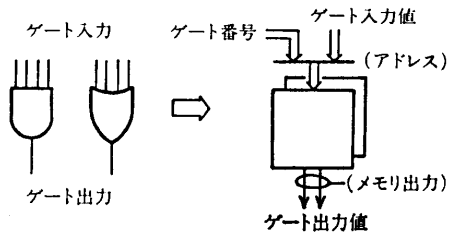


図 2 ゲート論理機能の RAM による実現
Fig. 2 Gate logic function realization by RAMs.

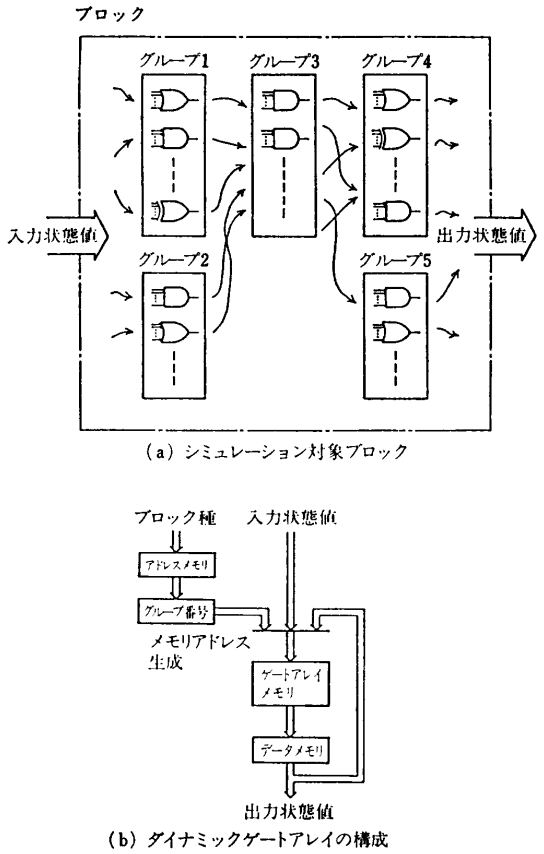


図 3 ダイナミックゲートアレイによるブロック論理演算
Fig. 3 Block logic operation by the dynamic gate array.

示すようにゲートの入力信号をメモリのアドレスとして与え読出しデータをゲートの出力とする．ブロックを構成するゲート群は**図 3** に示すように信号伝播順にレベルソートし，レベル順にゲートの出力値を求めていく．中間レベルに属するゲートの出力値はデータメモリに保存され次のレベルの入力値となる．ブロックを構成するゲートをいくつかのグループに分け，ゲートアレイメモリを構成する複数の RAM デコーダがゲート群の論理演算を並行して処理することにより高速化を図る．この RAM デコーダ群を繰り返し用いることにより次々とゲート論理演算を行い最終的なブロックの出力値を得る．

ダイナミックゲートアレイはゲート機能を RAM で実現しているのでブロック種類の変更が容易である．RAM のアクセスサイクルごとにグループの論理演算を行うので高速である．また，一つのダイナミックゲートアレイで最大 64 種類のブロックを扱うので，29 台を用いると千種類以上のブロック種を収容することができる．

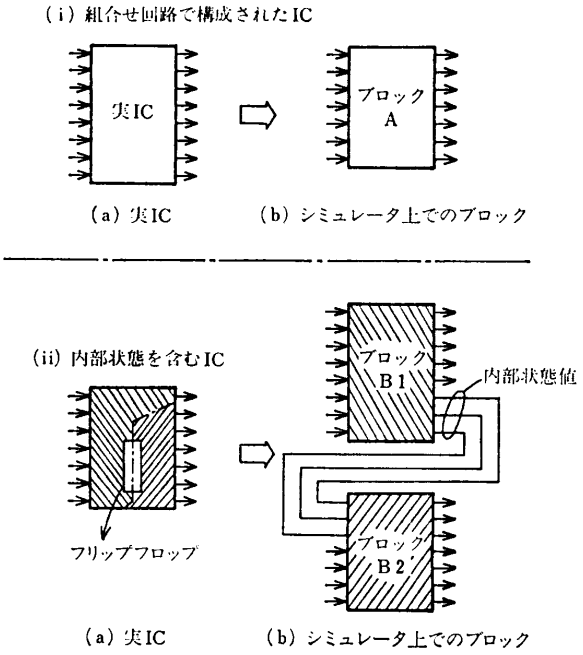


図 4 対象回路のブロックへの変換
Fig. 4 Transformation of an actual IC into blocks.

4.3 メモリシミュレータ

ダイナミックゲートアレイは組合せ回路を対象としているが，一般的にブロックは内部状態をもつものがある．ブロックを分類すると次の三つに分かれる．

(イ) 組合せ回路のみによる実ブロック
(ロ) フリップフロップ等を含む実ブロック
(状態数が数十個まで)
(ハ) メモリ (RAM 等) ブロック
(状態数が数十〜数十万個)

上記(イ)のブロックは直接ダイナミックゲートアレイで実現できる．上記の(ロ)の場合も**図 4** に示すように実ブロックを複数のブロックに分割し，内部状態を新規にブロックの外部ピンと定義することによってダイナミックゲートアレイで対応できる．これは同期形の回路ではフリップフロップ類も 1 クロックにたかだか 1 回しか変化しないので内部状態をブロックの外部ピンとして定義しておけば，入力状態表と出力状態表に状態が記憶されるのでブロック自体は組合せ回路とみなすことができる．しかし，上記(ハ)のメモリブロックは内部状態数が多く，ダイナミックゲートアレイで処理するのは容量が多すぎて不適当である．大型計算機を対象とするとキャッシュメモリ，メインメモリ等を含めて最少限でも数 MB の容量が必要となる．そこでメモリブロックを扱う専用のメモリシミュレータを考え用いることとした．本マシンではシミュレー

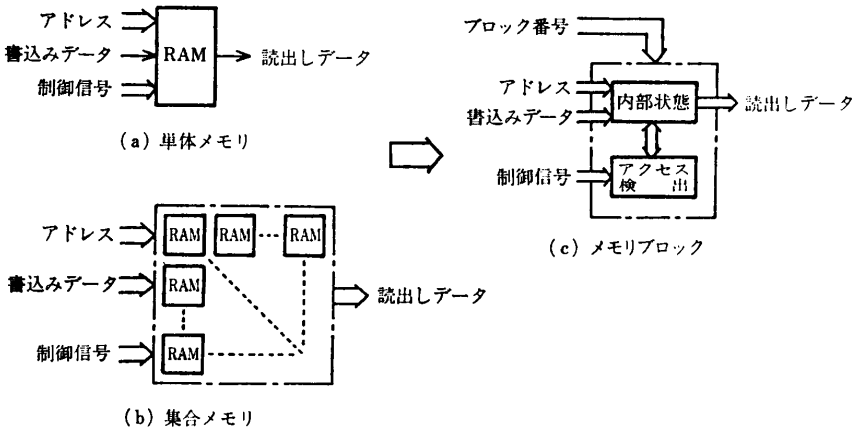


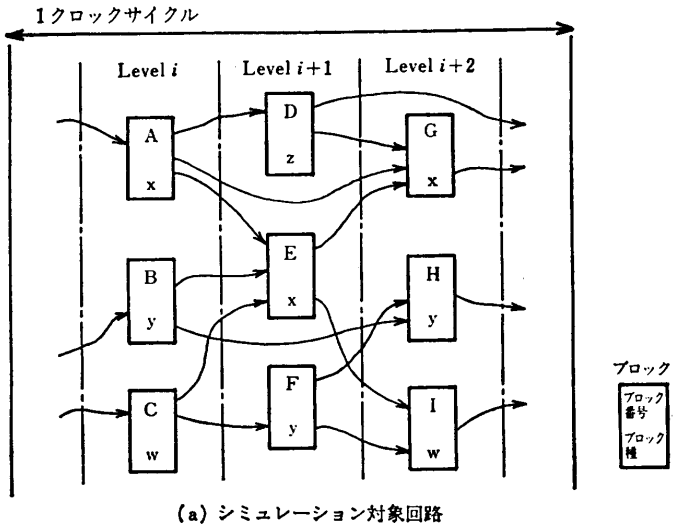
図 5　メモリブロックの扱い
Fig. 5　Representation of memory chips.

ション対象のメモリあるいはメモリの集合体を図5に示すように一つのメモリブロックとして扱うこととした．メモリブロックごとにメモリアクセスを検出する部分とアドレスで指定された数の内部状態を記憶する部分が必要である．これだけの容量をメモリシミュレータ上にもたせることは得策でないのでホスト計算機の主記憶域を内部状態の記憶場所として使用することとした．ホスト計算機とメモリシミュレータはメモリ共有型とし，双方から直接メモリをアクセス可能とする．こうすることによりシミュレーションデータの取扱いがホスト側からメモリロード／ストアの形式で行えるのでデータ操作が容易となる利点がある．

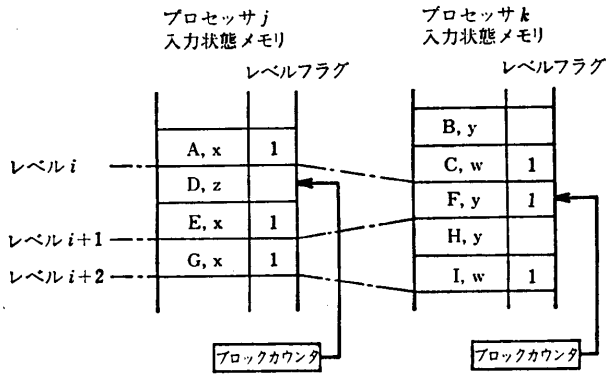
5. レベル同期方式

レベルソートの手法を並列プロセッサで実現するためには各プロセッサをレベルごとに同期させる手段が必要である．

レベルソートはシミュレーション対象の回路をあらかじめ入力側から信号伝播順序に従って番号をふり，この番号順にシミュレーションする方式である．ここで注目すべきことは同じ伝播レベルに属す複数のブロックは同時に処理できることである．本シミュレータではここに着目して全プロセッサはレベルごとに同期して処理を行うが，同一レベル内では各プロセッサは独自のタイミングで並列にブロックのシミュレーションを行う機構を考えた．レベル同期の動作を図6に示す．この例ではレベル分けされた対象の回路を2台のプロセッサが分担して処理にあたる．レベル同期のために各プロセッサの入力状



(a) シミュレーション対象回路



(b) レベル同期処理

図 6　レベル同期方式
Fig. 6　Level synchronization mechanism.

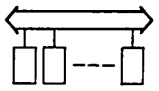
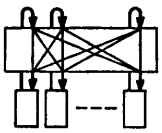
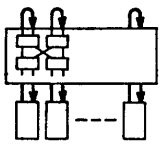
	(1) 共通バス方式	(2) クロスバー方式	(3) 多段接続ネットワーク方式
構成法			
ハードウェア量	N	$N\times N$	$N\times\log N$
配線数	N	$N\times N$	$N\times\log N$
ファンアウト数	N	N	1
転送量	バス容量まで	$\approx N$	$\approx N$

図 7 プロセッサ間結合方式の比較
Fig. 7 Comparison of inter-processor communication methods.

態メモリにブロックをレベル順に並べ，各レベルの終了箇所に1ビットのレベルフラグを設けることとした．各プロセッサはブロックカウンタを用いて順番に処理を行う．もし，レベルフラグが立っているときは全体を制御するプロセッサにレベル終了を伝える．全体制御プロセッサはすべてのプロセッサからレベル終了が届いた後，次のレベル開始を全プロセッサに通報する．

6. プロセッサ間結合方式

大型計算機をシミュレーションするためには少なくとも数十台のプロセッサを用いる必要がある．そのため，多数のプロセッサを結合しイベントを多量に転送できる方式を考える．結合方式として図7に示す三つを検討した．

方式(1)の共通バスではイベント転送が時分割となるのでプロセッサが十数台になるとバス上での競合が増大してバスネックとなってしまう．方式(2)のクロスバースイッチは大きな転送容量をもつが，ネットワークのハードウェア量が2乗のオーダで増大するのでコスト高となる．また，実装上の面でもファンアウト数が多いのでモジュール化が困難である．方式(3)の多段接続ネットワークはハードウェア量が $N\times\log N$ のオーダであるのでプロセッサ台数が増えてもそれほどのハードウェア増にならない．また，各モジュールのファンアウトが少ないので台数によらずモジュール構成がとれる利点がある．しかし，通過する段数が多いので各段で起こる衝突による性能低下の問題がある．

本シミュレータでは転送容量が大きく，ハードウェア量が比較的少ない多段接続ネットワークを用いることとした．しかも，パケットによる蓄積交換を行うことにより衝突の問題を解決することとした．ネットワークの構成要素としてルータセルと呼ぶ2入力2出力をもつクロスバースイッチを考え，さらにこのルータセルの各入出力ポートにバッファをもたせたモジュールを用いることとした．このルータセルを多段に接続し，蓄積交換形のネットワークを構成する．イベントはパケットの形で蓄積交換される．このため，一つのパケットの通過時間は回線交換より余計にかかるが，パケットの衝突が各段にあるバッファで緩和されるとともに，並列かつパイプライン式にネットワークをパケットが流れるので，高いパケット転送率を期待できる．しかも本ルータセルはLSI化に向いているのでネットワークのモジュール化，小型化が容易である．

7. シミュレーション手順と性能

7.1 シミュレーション手順

シミュレーションが開始されると各プロセッサは次の手順で並列にシミュレーション処理を実行する．

(1) 入力状態メモリを順番に調べ，イベントを検出する．もし，イベントがあれば，入力状態値とブロック種を入力状態メモリから取り出しダイナミックゲートアレイに論理演算を依頼し，イベントフラグをリセットする．

(2) ダイナミックゲートアレイもしくはメモリシミュレータはブロック種と入力状態値からブロックの出力状態値を求め出力する．

(3) 新しく得た出力状態値を出力状態メモリにセットする．このとき，以前の出力状態値と比較し，出力状態値の変化の有無を調べる．

(4) もし，出力状態値に変化があれば，ブロック番号と変化したピン番号から接続メモリをアクセスし，接続先の入力ピンに対する変化ピンパケットを得る．

(5) 変化ピンパケットをルータセルネットワークを介して相手側のプロセッサに送る．もし，相手先が自分自身のときはネットワークをバイパスし，直接自身のプロセッサ内のパケット入力部へ送る．

(6) ブロックの変化ピンパケットをブロック番号，ピン番号の形で受ける．

(7) ブロック番号とピン番号を用いて入力状態メ

モリの状態値を更新し，同時にイベントフラグをセットする．

このシミュレーション手順(1)において入力状態メモリをアクセスし，かつ，レベルフラグを検出した場合，メモリのイベント取出し作業を中断し，コントロールプロセッサにレベル終了を連絡する．コントロールプロセッサは全プロセッサがレベル終了した後，レベル再開信号をブロードキャストする．これにより次のレベルのシミュレーションが始まる．

7.2 シミュレーション性能

1台のプロセッサは最大1,024ブロックまで収容可能であるからシステム全体では 29 台の論理プロセッサを用いると 29,696 ブロックまで収容できる．しかし，ブロック内に状態を含む場合2ブロックに分割することを考慮し，約半分のブロックが内部状態をもつと仮定すると実際に収容可能な容量はおよそ次のとおりである．

- シミュレーション最大ブロック数
  15,000 ブロック
- シミュレーションメモリブロック数
  2,000 ブロック (2 MB まで)

32 台の専用マシンを用いて得られる処理性能は対象とする装置の特性に依存するところが大きく正確には実際に動作させないとわからないが，概算を行うため次の仮定をおく．

(1) 1回のシミュレーションサイクルで 2/3 のブロックがイベントを発生する．

(2) 1回のブロック評価で 10 個のイベントを発生する．

(3) 1個のブロックは 75 ゲートからなる．

以上の仮定のもとにマシン各部の処理にどの程度時間がかかるかを見積もる．

(1) ルータセルネットワーク

ルータセルはパケットの衝突がなければ1個のイベントを 400 ns ごとに転送できる．簡単な計算機シミュレーションの結果，衝突による低下は 1.4 以下である．また，ブロックの行先が自分宛のときはネットワークがバイパスされるので局所性を考えて外へ出る割合を 0.8 とすると

$$10\times400\times1.4\times0.8=4.5\ \mu s$$

(2) ダイナミックゲートアレイ

1回の RAM アクセスサイクル (300 ns) で 16 入力のゲートを 16 個同時に求める．75 ゲートを評価するのに 32 ゲート×4 段と仮定すると

$$0.3\times32/16\times2\times4=4.8\ \mu s$$

(3) 接続メモリ

10 個のパケットを出すために 10 回のページモード読出し (450 ns) を行う．

$$0.45\times10=4.5\ \mu s$$

(4) 入力状態メモリ

3ブロックに2個のイベントがあるのでイベント当り 10 回のアクセスと 1.5 回のイベントスキャンが行われる．1回のアクセスを 300 ns とすると

$$(10+1.5)\times0.3=3.45\ \mu s$$

以上の処理がパイプライン式に行われると理想的には 5 μs ごとにブロックのイベント処理が可能となる．シミュレーション対象マシンの規模を2万ブロックと仮定すると1クロックのシミュレーションは下式から理想的には 2.3 ms で行われる．

$$20,000\times1/29\times2/3\times5\ \mu s=2.3\ ms$$

しかし，実際にはプロセッサ間の負荷不均衡やパイプラインステージ間の停滞等による同期損が存在する．仮に 2.5 倍の損失を見込んでも1クロックのシミュレーションが約 5.7 ms で実行できるといえる．これはゲートレベルに換算すると下式から 150 万ゲートの装置を毎秒 2.6 億ゲート評価の速度で処理することに匹敵する．

容量：75 ゲート×20,000 ブロック$=1.5\times10^6$ ゲート

速度：$1.5\times10^6\times1/5.7\ ms=2.6\times10^8$ ゲート/s

これは従来のソフトウェアシミュレータの数千から数万倍の処理スピードをもち，大型計算機のシミュレーションにも十分な性能をもっている．

8. むすび

本論文は従来のゲートレベルシミュレーションとは異なるブロックレベルに基づく論理シミュレーションを高速に実行する専用のマルチプロセッサのアーキテクチャを提案し，その設計思想に基づき超高速論理シミュレータを設計・試作した．超高速論理シミュレータは論理シミュレーション処理の各処理ステップをすべてハードウェア化し，次の新しい概念に基づく試みがなされている．

- ブロックレベルシミュレーション方式
- 専用プロセッサによる並列パイプライン処理
- ダイナミックゲートアレイによる論理演算
- メモリプロセッサによる大容量メモリの扱い
- ルータセル LSI による蓄積交換形多段接続ネッ

トワーク

本シミュレータを評価した結果，メモリを含めた大型計算機のような大規模なモデルを取り扱うに十分な容量と処理性能をもつことがわかった．

今後，本マシンを実際に現場で使用し，実際の運用状態での性能評価を行い，システムの有効性の検証および問題点の解明を進めていく予定である．

謝辞 最後に研究の機会を与えてくださった，日本電気(株) C & C システム研究所三上所長代理，箱崎部長，山本課長，コンピュータ技術本部斉藤本部長桑田部長，情報処理製造・装置システム事業部山田技師長，病院情報システム事業部北野事業部長代理に深謝します．数々の提案をいただいた第二 OA 装置事業部赤井主任，伝送通信事業部富田主任に感謝します．また，ハードウェアの設計・製作を担当してくださった三野輪氏，近藤氏，幅田氏，ソフトウェアサポートシステムの製作を担当してくださった堀田主任，野水主任，伊藤氏，田中氏に感謝します．

参考文献

1) 小池，大森，佐々木：論理シミュレーションマシン，信学研資，EC 82-42 (1982).
2) 大森，小池，佐々木：論理の動的可変なダイナミックロジックアレイ，信学研資，EC 82-12(1982).
3) Koike, N., Ohmori, K., Kondo, H. and Sasaki, T.: A High Speed Logic Simulation Machine, Compcon 83 Spring, pp. 446-451 (1983).
4) Sasaki, T., Koike, N., Ohmori, K. and Tomita, K.: HAL: A Block Level Hardware Logic Simulation, 20th DA Conf., pp. 150-156 (1983).
5) Pfister, G. F.: The Yorktown Simulation Engine; Introduction, 19th DA Conf., pp. 51-54 (1982).
6) Abramovici, M. et al: A Logic Simulation Machine, 19th DA Conf., pp. 65-73 (1982).
7) Sasaki, T. et al.: MIXS: A Mixed Level Simulator for Large Digital System Logic Verification, 17th DA Conf., pp. 626-633 (1980)

(昭和 59 年 2 月 13 日受付)
(昭和 59 年 4 月 17 日採録)