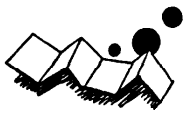
解　説

4. 各種のハードウェアアルゴリズム



4.5 論理シミュレーションマシンのハードウェアアルゴリズム†

大森健児††　小池誠彦†††

1. はじめに

LSIの高集積化，コンピュータの大規模化に伴い，これらを開発する上で，現在の論理シミュレーションの方法―大型計算機を用いてソフトウェアでシミュレーションする方法―では限界にきている．このため，論理シミュレーションの速度を10^2倍から10^4倍向上させることを目指して各所で論理シミュレーションのための専用マシンが開発されている[1)~4)]．本解説では，これらの論理シミュレーションマシンの中でどのようなハードウェアアルゴリズムが使われているかについて述べる．

少し，歴史的なことを述べると，最初の論理シミュレーションマシン[5)]は非常に古く，1960年代にさかのぼる．しかし，それ以後1980年代に入るまで論理シミュレーションマシンは現われず，次の論理シミュレーションマシン[6)]が現われたのは1980年である．この頃，VLSIの開発，大型コンピュータの開発をかかえる機関の中で論理シミュレーションの高速化が切実なものとなった．また，VLSI技術の発展により，このような専用マシンを作ることも現実的なものとなってきた．このようなことを反映して，1982年にベル研が論理シミュレーションマシンの構想を，また，IBM，日電が社内使用の開発マシンを発表した．また，これを契機に，商用マシンも現われてきた．

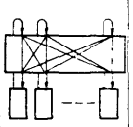
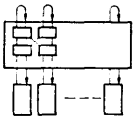
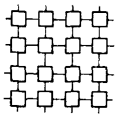
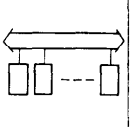
2. 高速化のアルゴリズム

論理シミュレーションマシンの主要な使命は高速性にある．高速なマシンを実現するための有力な方法は，並列性を利用することである．このための方法には，複数のプロセッサを用意したマルチプロセッサシステムと，作業フェーズを少しずつずらして複数の仕事を同時に行えるようにしたパイプライン方式とがある．

マルチプロセッサシステムの構成法には，いくつかの方法がある．それぞれの構成に対する詳細は他の論文[7),8)]にゆずることとして，その性質を表-1に示す．どのような構成のマルチプロセッサシステムにするか，あるいはどのようにパイプラインを実現するかは対象とする論理シミュレーションの方法に大きく左右される．

論理シミュレーションにはいくつかの方法[9)~11)]が

表-1　マルチプロセッサシステムの構成法

	クロスバ方式	ネットワーク方式	アレイ方式	共通バス方式
構成法				
任意のプロセッサ間での通信時間	小／一定（数十n～数μ）	中／変動（数百n～数μ）	大／変動（ただし隣り合う場合小／一定）	中／変動（数百n～数μ）
可能なシステム規模	大（～数百台）	大（～千台）	大（～数百台）	小（～数十台）
システムの伸縮	×	×	×	○
ハードウェア量（N：プロセッサ数）	大($N\times N$)	中($N\times\log N$)	小(N)	小(N)
耐故障性	×	○（冗長パス）	×	△（複数バスとした場合）
制約	複数のプロセッサが同時に一つのプロセッサにアクセスすることは不可	メモリリードのような通信は不可	2次元配列をとりにくい処理は不得手	
例	YSE, LSM	HAL	AAP	LE2001

† Hardware Algorithm for Logic Simulation Machines by Kenji OHMORI (College of Engineering, Hosei University) and Nobuhiko KOIKE (C & C Systems Labs, NEC Corporation).

†† 法政大学工学部経営工学科
††† 日本電気(株) C&C システム研究所

表-2 論理シミュレーションの方式

	ゼロ遅延モデル	ユニット遅延モデル	標準遅延モデル
実行順序	レベルソート	制限なし	タイムマッピング
シミュレーション手順	コンパイル/テーブルドリブン	コンパイル/テーブルドリブン	テーブルドリブン
並列性	中（レベル内の素子）	大（全素子）	小（タイムスロット内の素子）
処理量	1ラウンド/1クロック	1～10^3 ラウンド/1クロック	10～10^3 ラウンド/1クロック
ハード量	制御ハード小，メモリ量小	制御ハード大，メモリ量中	制御ハード大，メモリ量大
適応性	同期回路	同期／非同期回路	同期／非同期回路
その他	レベルソートの前処理必要	ループの終了検出必要	実回路に近い

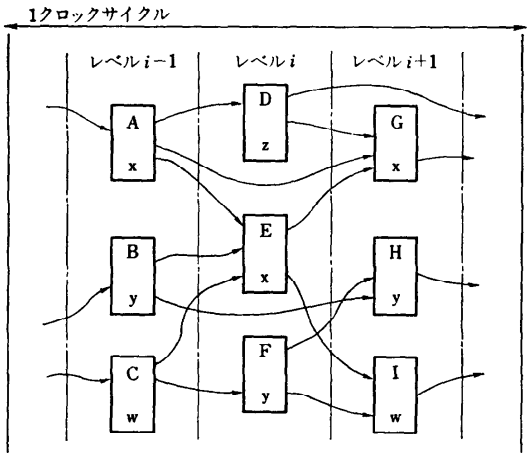


図-1 レベル分割

存在するが，これは，遅延モデルによって分類することができる．論理シミュレーションマシンで利用している遅延モデルには，①論理素子は遅延を持たないとしたゼロ遅延モデル，②すべての論理素子は同一の遅延時間を持つとしたユニット遅延モデル，③各論理素子は種類ごとに定められた遅延時間を持つとした標準遅延モデルの3つがある．これらの特徴を示したのが表-2である．

ゼロ遅延モデルでは，論理素子は図-1に示すように配線の深さに従って，レベルソートし，レベルに従ってシミュレーションを行う．同一のレベルに属す論理素子については，シミュレーションを同時に行っても構わないので，並列度は同一レベル中に属す論理素子の数ということになる．1クロックに対するシミュレーションは，すべてのレベルを一度なめるだけでよいので，シミュレーション時間は少ないが，同期式回路にしか利用できない．

ユニット遅延モデルでは，入力と出力のテーブルを用意しておき，入力のテーブルを用いて論理素子のシミュレーションを行い，その結果を出力のテーブルに書き込んでおく．すべての論理素子についてのシミュレーションが終了したとき，入力と出力のテーブルを入れ換えて，また，シミュレーションを行う．全論理素子について同時にシミュレーションを行うことができるので，並列度はきわめて高い．しかし，1クロックに対するシミュレーションは，信号変化がなくなるまで続けることになるため，何回も繰り返すことになる．

なお，ゼロ遅延モデル，ユニット遅延モデルともすべての論理素子についてシミュレーションを行うということで説明したが，入力に変化のあった論理素子についてシミュレーションを行うことで，シミュレーション回数を減らすことができる．この方法をイベント駆動と呼ぶ．

標準遅延モデルはタイムホイールを用意しておき，現時刻につながっている論理素子をシミュレーションする．シミュレーションの結果，出力に変化があったならば，その出力につながっている論理素子を遅延時間分だけ後の時刻の所でタイムホイールにつなぐ．同一時刻につながっている論理素子については，同時にシミュレーションしても構わないが，並列度はそれほど高くならない．

現在までのところ，論理シミュレーションマシンとして日電から HAL[12)~19)]，IBM から YSE[20)~22)] と LSM[23)~27)]，ベル研から Abramovici のマシン[28),29)]，Levendel のマシン[30)]，Zycad から Logic Evaluator，Daisy から Megalogician，Valid から Realfast，MIST から ULTIMATE[31)]，通研から AAP[32)] が発

表-3 論理シミュレーションマシンの比較

ハードウェアアルゴリズム ＼ システム名		HAL	YSE	LSM	Bell-A	Bell-L	LE	ML	RF	UL	AAP
遅延	ゼロ	○	○				○	○	○		
	ユニット		○	○			○	○	○		○
	標準			○	○	○	○	○	○	○	
シミュレーション単位	ゲート		○ (4入力)	○ (5入力)	○	○	○ (3入力)	○	○	○	○
	ブロック/機能	○			○	○				○	
シミュレーション手順	コンパイル/テーブル	テーブル	コンパイル	コンパイル	テーブル	テーブル	テーブル	テーブル	テーブル	テーブル	
ゲート/ブロックの評価方法	ハード/ソフト	ハード	ハード	ハード			ハード	ファーム		ハード	ソフト
	リアルチップ							○	○		
メモリシミュレーション		○		○			○				
信号値	2値	○			不明	不明				不明	○
	多値		○ (4値)	○ (4値)			○ (3値)	○ (3値)	○ (4値)		
高速化	イベント駆動	○			○	○	○	○	○	○	
	パイプライン	○	○	○			○	○		○	
並列処理結合方式	クロスバー		○	○		○					
	ネットワーク	○ (オメガ)									
	アレイ										○
	バス					○	○				
分散処理					○			○	○	○	

Bell-A: Abramovici のマシン　LE: Logic Evaluator　RF: Realfast
Bell-L: Levendel のマシン　ML: Megalogician　UL: ULTIMATE

表されている．これらの特徴を表-3 に示す．

論理シミュレーションの高速化を進める上で，シミュレーションの単位も重要である．これには，ゲートを単位としたゲートレベルシミュレーション，フリップフロップ，レジスタ，カウンタを単位とした機能レベルシミュレーション，ゲートとフリップフロップで構成されるブロックを単位としたブロックレベルシミュレーションがある．単位を大きくした方がシミュレーション時間は短くなるが，精度は粗くなる．

シミュレーション手順の表わし方には入力信号，出力信号，接続情報をテーブルで表現しておくテーブル方式と，命令の形で表わすコンパイル方式とがある．コンパイル方式はコンパイル時にシミュレーションの順序を決めるので，コンパイラにより最適化を行えれば，シミュレーション時間は短縮できる．ただし，コンパイル方式では，イベント駆動を用いるのは難しい．

3. 論理シミュレーションマシンの例

3.1 HAL

HAL は，図-2 に示すように，29 台の論理プロセッサと 2 台のメモリプロセッサと 1 台の制御プロセッサとで構成される．論理プロセッサは，ブロックのシミュレーションをするものでノードプロセッサとダイナミックゲートアレイ (DGA) とで構成される．ノードプロセッサは，イベントの立っているブロックの検出，入力信号の取り出し，新旧出力信号の比較，イベントの送出，受け取り等を行う．DGA はノードプロセッサより，入力信号をもらってブロックの評価を行い，新しい出力信号をノードプロセッサに渡す．メモリプロセッサは，シミュレーション対象マシンのメモリ部のシミュレーションを行うもので論理プロセッサと似たような構成となっている．制御プロセッサは，ホストプロセッサとのデータの授受を行うとともにレベルの制御を行う．

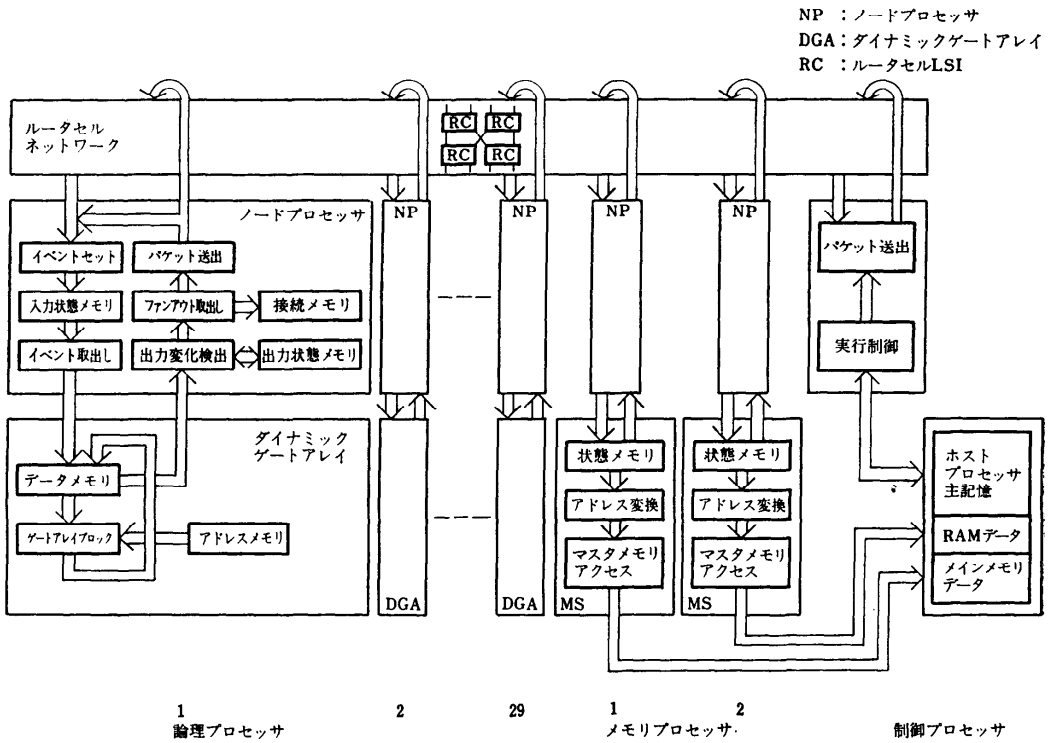


図-2　HAL の構成

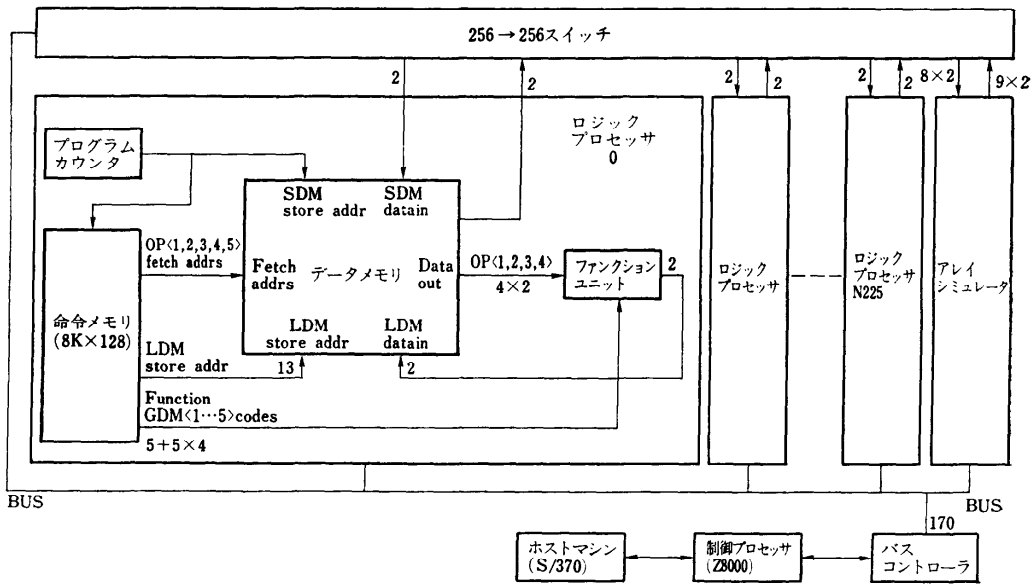


図-3　YSE の構成

3.2 YSE

YSE は，図-3 に示すように，最大256台までのプロセッサで構成される．各プロセッサは，命令メモリ，データメモリ，ファンクションユニットよりなる．命令メモリは命令を記憶するためのもので，命令はデータメモリからのデータの取り出し場所，書き込み場所，ファンクションの種類を示す．データメモリは，ゲートの入力信号，出力信号を記憶しておくためのものである．ファンクションユニットは，4入力1出力のゲートのシミュレーションを行うためのものである．

4. 全体回路/装置シミュレーションのためのハードウェアアルゴリズム

4.1 ゼロ遅延モデル

HAL では，ゼロ遅延モデルのシミュレーションをランクオーダに基づいて行う．各論理プロセッサには，シミュレーションすべきブロックが分配される．図-4 の例は図-2 の回路を分配したものである．この分配に基づいて，論理プロセッサ対応に入力状態テーブル，出力状態テーブル，接続テーブルが作られ，それぞれの入力状態メモリ，出力状態メモリ，接続メモリにロードされる．シミュレーションは次のように行われる．

今，レベル i に対してシミュレーションを行うとする．このとき，HAL のノードプロセッサは図-5 に示すようなハードウェアアルゴリズムに基づいてシミュレーションを行う．

イベントフェッチ

① ブロックカウンタで示されるブロックのイベントを調べる．イベントが立っていれば②へ，立っていなければ③へ

② 入力状態メモリより，このブロックへの入力信号とブロックの種類を得て④へ

③ ブロックカウンタを1進める．同一レベルであるならば①へ，次のレベルに移ったなら，このレベルに対するシミュレーションの開始信号を待つ．

DGA への出力

④ DGA に入力信号とブロックの種類を送り，ブロックの評価を開始し，⑤へ

DGA からの入力

⑤ DGA でブロックの評価が終了したとき，そのブロックの新しい出力信号を得て⑥へ

出力ピンアクセス

⑥ 出力状態メモリから前の出力信号を取り出し，これと，今得た新しい出力信号と比較する．変化があった場合には⑦へ，変化のない場合は③へ

⑦ 信号に変化のあった出力ピンを取り出すとともに，出力状態メモリを新しい出力信号で更新し，⑧へ

接続メモリアクセス

⑧ 変化のあった出力ピンに対して，ファンアウト先（出力ピンにつながっている入力ピン）を調べ，ファンアウト先の論理プロセッサに信号値が変わったことを伝える．すべての出力ピンについて終了したとき，③へ移る．

ノードプロセッサは，信号値が変わったという通信を受けて，入力状態メモリの更新を行う．また，制御プロセッサは，すべてのノードプロセッサが次のレベルの開始待ちになり，しかもネットワークの中から転送中のデータがなくなったとき，次のレベルの開始を伝える．

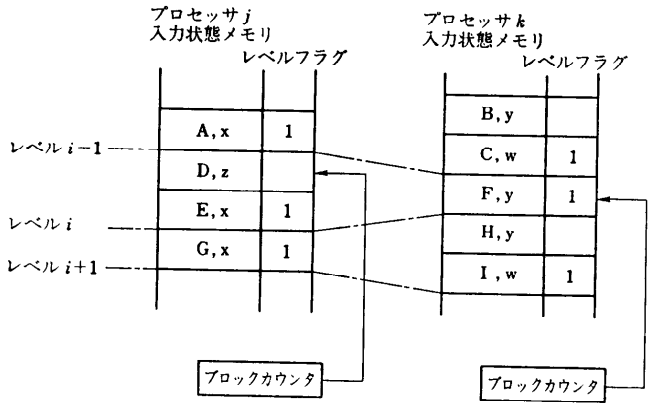


図-4 プロセッサのゲート/ブロックの分配

4.2 ユニット遅延モデル

YSE はゼロ遅延モデルのシミュレーションを行うことができるが，ここでは，ユニット遅延モデルについて説明する．

YSE ではホストプロセッサが用意するのはテーブルではなく，命令である．すなわち，各プロセッサの命令メモリには，実行すべき命令が入る．YSE では4入力のゲートを扱うので命令は次のような構成となっている．①ゲートの種類（ファンクションユニットへの入力となる），②入力データの読み出し場所（データメモリのアドレスを示す），③出力データの書き込み場所（データメモリのアドレスを示す），

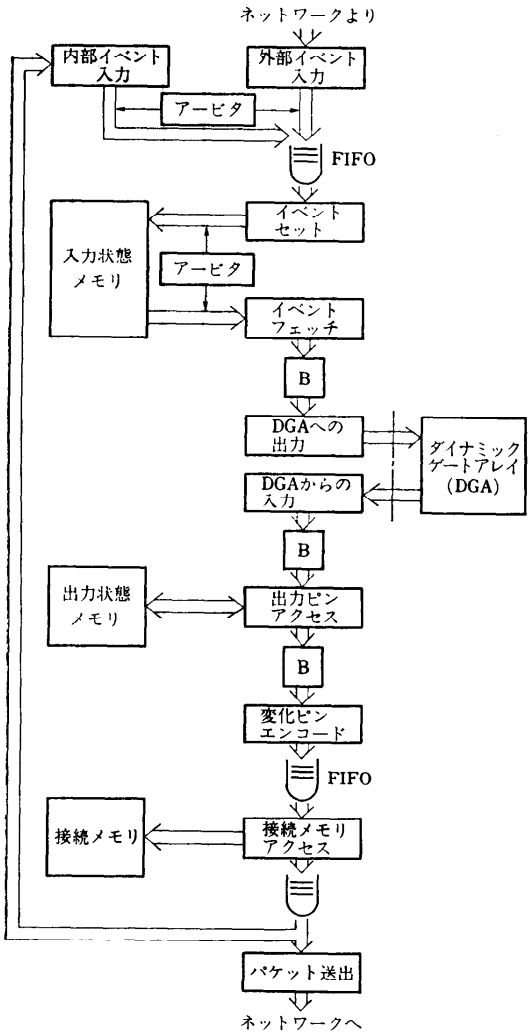


図-5 ゼロ遅延モデルのハードウェアアルゴリズム

④出力データの値を他のプロセッサに与えるために，それを読み出す場所（データメモリのアドレス）．（ただし，③と④の出力データはこの命令の実行によって得るものではなく，前の命令の実行によって得たものである．）

データメモリは2分割され，入力データを記憶しておくための入力側と，出力データを記憶するための出力側とに分けられる．すべての命令の実行が終了したとき，入力側と出力側を入れかえ，先に得られた出力データを新たな入力データとし，最初の命令から再び実行する．

YSE ではシミュレーションは図-6 のように行われる．その中の基本的な部分は次のように実行される．

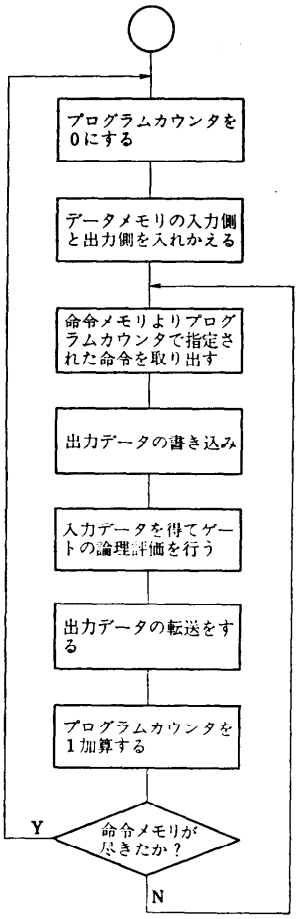


図-6 ユニット遅延モデルのハードウェアアルゴリズム

出力データの書き込み

ファンクションユニットからの出力データを，命令の③で指定された場所に書き込む．この出力データは，一つ前の命令を実行した時に，ファンクションユニットがゲートの論理評価によって得たものである．また，スイッチより送られてきたゲートの出力をプログラムカウンタで指定された場所に書き込む．これらは後で入力データとして利用される．

ゲートの論理評価

命令の②で示された4カ所より入力データを読み出し，これをファンクションユニットに送る．ファンクションユニットでは，命令の①に基づいて新しい出力データを得る．

出力データの転送

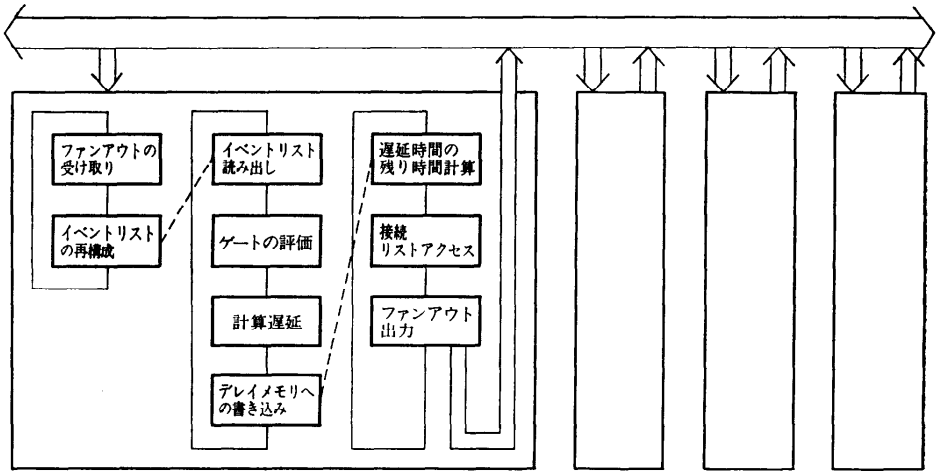


図-7 標準遅延モデルのハードウェアアルゴリズム

ゲートの論理評価と同時に命令の④で示された場所より出力データを読み出し，スイッチに出力する．このとき，バスコントローラは各プロセッサに対してどのプロセッサからデータを受け取るべきかを示す．これも，命令の一部である．なお，メモリはデータの書き込み，読み出しのために時間を費やさなくてもよいように，2入力ポート，5出力ポートの特別な構成になっている．

4.3 標準遅延モデル

標準遅延モデルのシミュレーションは，商用のマシンで行われているが，これらについては発表論文がなく不明である．しかし，おおむね，図-7 のハードウェアアルゴリズムにより行われているものと推察される．

イベント処理

① 入力変化を受け取る．

② 入力信号に変化のあったゲートをイベントリストに接続する．

ゲートの評価

① イベントリストより，ゲートの種類と入力の値を得る．

② ゲートの評価を行い，新しい出力信号を得る．

③ 出力信号の値が変わったなら，このゲートに対する遅延時間を得てデレイメモリに書き込む．

ファンアウト出力

① 時間を1つ進めるという信号がきたとき，デレイメモリに書き込まれている遅延時間の残り時間を1つ減らす．

② 残り時間がゼロになった出力ゲートに対し，そのファンアウトを接続リストより求める．

アドレス		メモリの内容
ゲートの種類	入力信号	出力信号
0 (リセット)	00	0
	01	0
	10	0
	11	0
1 (AND)	00	0
	01	0
	10	0
	11	1
	00	0
	01	0
14 (NAND)	00	1
	01	1
	10	1
	11	0
15 (セット)	00	1
	01	1
	10	1
	11	1

図-8 デコードメモリの例

③ ファンアウト先に入力信号に変化のあったことを伝える．

なお，Logic Evaluator では，入力側にも遅延を指定できるようになっている．

5. ゲート/ブロックの論理評価アルゴリズム

5.1 デコードメモリ

デコードメモリは，ゲートあるいはブロックの論理

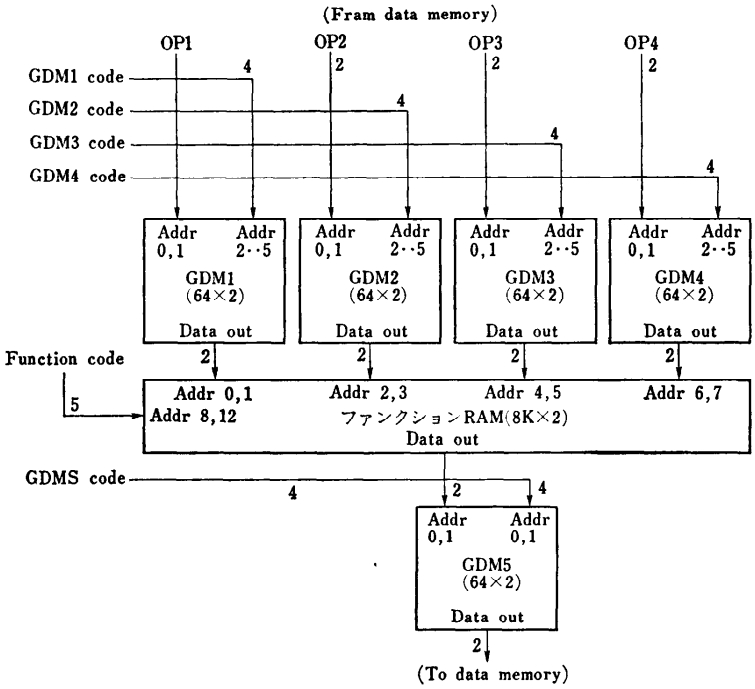


図-9　ゲートの論理評価

評価のためのハードウェア素子として利用されるので，その原理について簡単に述べておこう．デコードメモリはメモリを用いて作られたもので，これには，ゲートあるいはブロックの種類とこれに対する入力信号をアドレスとし，各アドレスで指定されたメモリ位置にはこれに対応した出力信号が書き込まれている．今，入力が2個のゲートについて考えるとすると，一般に知られているように16種類のゲートが存在する．そこで，図-8 に示すように，このパターンを RAM あるいは ROM に書き込んだとする．上位のアドレス4ビットでゲートの種類に対するパターンを選択し，下位のアドレス2ビットで入力を与えるようにすれば，ゲートに対する論理評価を，メモリからの出力として，得ることができる．

5.2　ゲートの論理評価

YSE の論理評価の対象となるのは4入力ゲートである．また，各入力は0，1，ハイインピーダンス，不定の4値をとる．このようなゲートに対して，可能な組み合わせをすべて表わそうとすると，膨大な量のデコードメモリを必要とする．また，可能な組み合わせのうちの多くのものは不要なものである．そこで，YSE では，ゲートの種類選択のための信号線を 25 本とし，デコードメモリの組み合わせにより，必要なゲートを表わせるように図-9 に示すように工夫している．ファンクションユニットは，3段に別れていて，一段目には4個の GDM（汎用ドモルガンメモリ），二段目にはファンクション RAM，三段目には1個の GDM がある．GDM は4値を持つ入力に対して，アイデンティティ（そのまま），論理の反転等16種類の演算を指定するものである．GDM の出力はやはり4値である．ファンクション RAM は，GDM からの出力を得てそれらに対して，AND, OR, XOR 等32種類の演算を施す．

5.3　ブロックの論理評価

HAL でもブロックの評価は，デコードメモリの原理を用いたダイナミックゲートアレイ（DGA）で行う．DGA の構成を図-10 に示す．HAL では，DGA にロードできるようにするため，ブロックは，入力に否定をもつことのできる AND, OR, XOR の3種類のゲートで構成された回路に変換される．その例を図-11 に示す．変換後の回路では，ブロックを信号の伝播順序に従ってランクオーダに分けたのと同じ原理より，ゲートにランクオーダがつけられる．DGA では，同一のランクオーダに属すゲートは同時に評価す

る．このため，各ゲートの評価は，そのランクオーダで入力となっている信号すべてを使って行われる．

入力信号は4つずつに束ねられ，それらは，ゲートの評価を行うために利用されるゲートアレイメモリ(GAM) への入力となる．GAM はデコードメモリであり，その出力は，ゲートへの入力となっていなかった入力信号は無視し，ゲートへの入力となっていた入力信号が，そのゲートの論理と同じ論理をとった時に出す値と同じ値をとるようにする．たとえば，ゲートがAND で4つに束ねた信号 A_1,

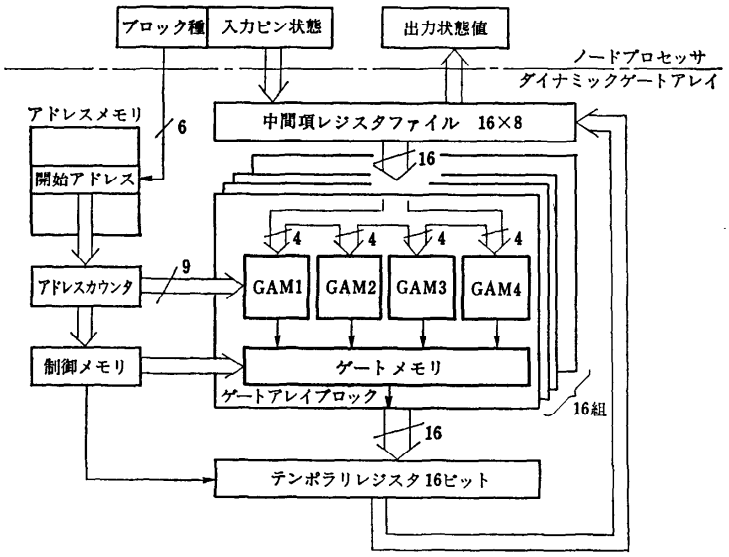


図-10 ダイナミックゲートアレイの構成

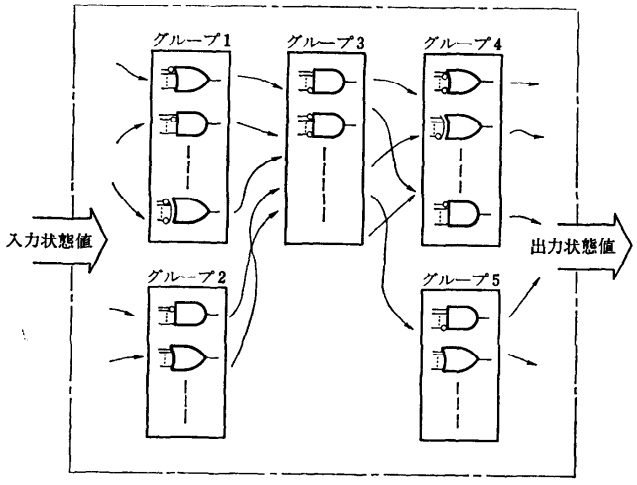


図-11 ブロック内のゲートの順序付け

A_2, A_3, A_4 のうちゲートへの入力は，$\bar{A}_2$ と A_3 であったとき，デコードメモリは $\bar{A}_2 \cdot A_3$ を表わすようにする．

ゲートメモリは，GAM の出力を入力とするデコードメモリであり，その論理としてはゲートを表わすのに使われた論理を表わす．先の例では，ANDを表わす．

ランクオーダに属すゲートが M 個あり，ランクオーダへの入力信号の数が N 個であるとする．このとき，一つのゲートを表わすためには，$N/4$ 個のデコードメモリを必要とする．また，M 個のゲートを表わすためには，デコードメモリの出力の1ビットずつをそれぞれのゲートに対応させることとすると，M ビット出力のデコードメモリが必要となる．しかし，ハードウェア的には，M と N が大きくなったとき，これをそのまま実現することは不可能なので，DGA では，基本的には M を 16，N を 16 とし，それを越えるものについては，たたみこみによって処理を行うようにした．

6. プロセッサ間通信アルゴリズム

ブロックあるいはゲートの評価の結果，出力に変化があったとき，それにつながれたブロックあるいはゲートに伝えなくてはならない．並列処理で論理シミュレーションを行っているシステムでは，このためにプロセッサ間通信を必要とする．

HAL では，プロセッサ間を接続するネットワークは図-12 に示すように，蓄積交換型のオメガネットワークになっている．ネットワークを通るパケットは，行く先のプロセッサを示すアドレスと，そこに伝える情報から成り立っている．ルータセルは，パケットを取り込んだとき，行く先のアドレス中の指定されたビット(たとえば，N 段目のルータセルでは N ビット目) に従って，パケットをいずれかの出口から出力

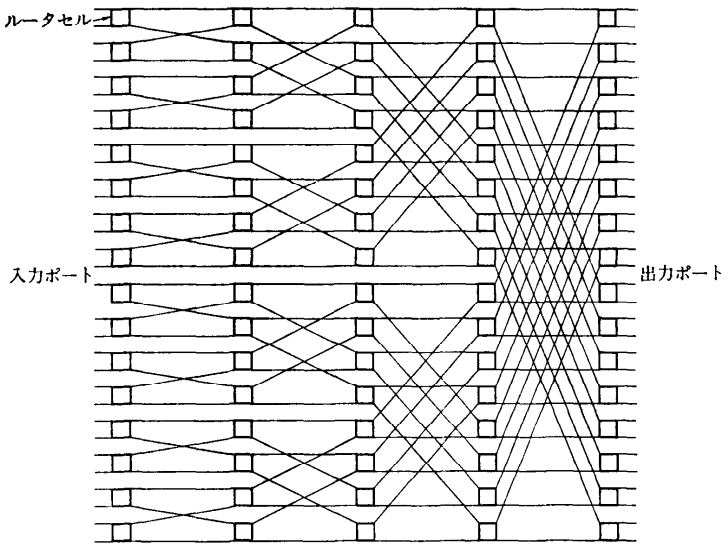


図-12　HAL の通信メカニズム

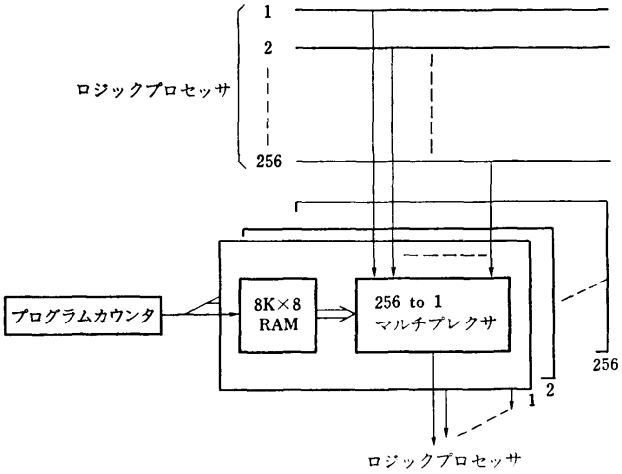


図-13　YSE の通信メカニズム

表-4　性能比較

諸元 / システム名	プロセッサ数	容量(Mゲート)（2入力1出力のゲート換算）	性能(Mゲート/秒)
Megalogician	1	1	0.1
Realfast	1	2.5	0.5
Logic Evaluator	16	3.8	60
HAL	31	3	360
YSE	256	4	960

する．YSE は，図-13 に示すようにクロスバースイッチになっている．データを送る出力側のプロセッサと，データを受け取る入力側のプロセッサとは，マルチプレクサで接続される．そのときの選択は，入力側のプロセッサ各々に対して，どの出力側のプロセッサを選ぶかということによってなされる．どのように選択したらよいかは，RAM で指定されるが，RAM の中のデータは，命令メモリの命令とともにコンパイラによって生成される．

最後に文献1）から各マシンの性能を引用し，これを表-4 に示す．

7. おわりに

論理シミュレーションマシンでは，従来のソフトウェアで利用していたシミュレーションのアルゴリズムをハードウェア化して，ハードウェアアルゴリズムとして利用している．しかし，ソフトウェアの中で広く利用されているアルゴリズムが，ハードウェアアルゴリズムとして必ずしも利用されているわけではなく，専用マシン出現への強い要求となった高速性を基本にしてアルゴリズムが利用されていることに特徴がある．

しかし，論理シミュレーションを必要とする人達は LSI 設計者側の方が多い．論理規模が小さかったために，ソフトウェアでのシミュレータで我慢できていた人々が大規模な LSI を設計するようになり始めると，標準遅延モデルのようなシミュレーション精度の高い，高速な論理シミュレータを求めるようになる．このため，ワークステーションの中に組み込まれている論理シミュレータの一層の高速化と大容量化が求められるようになる．これを達成するためには，革新的なハードウェアアルゴリズムの発見が必要とされるが，このような進歩があることを期待したい．

参 考 文 献

1) Blank, T.: A Survey of Hardware Accelerators Used in Computer-Aided Design, IEEE Design & Test of Computers, Vol. 1, No. 4, pp. 21-39 (1984).
2) 大森, 小池: CAD 専用処理装置, 情報処理, Vol. 25, No. 10, pp. 1144-1152 (1984).
3) Lineback, J.R.: Logic Simulation Speeded with New Special Hardware, Electronics, pp. 45-46 (1982).
4) CAE Stations' Simulators Tackle 1 Million Gates: Electronic Design (1983).
5) Makay, A.R.: Comment on 'Computer-Aided Design: Simulation of Digital Design Logic', IEEE Trans., Vol. C-18, No. 9, p. 862 (1969).
6) Barto, R. and Szygenda, S.A.: A Computer Architecture for Digital Logic Simulation, Electronic Engineering (1980).
7) 高橋: 並列処理のためのプロセッサ結合方式, 情報処理, Vol. 23, No. 3, pp. 201-209 (1982).
8) 大森: マイクロプロセッサによるマルチプロセッサシステムの技術動向, 情報処理, Vol. 23, No. 7, pp. 651-658 (1982).
9) 村井: ゲート・レベル論理シミュレーション, 情報処理, Vol. 22, No. 8, pp. 762-769 (1981).
10) 可児, 川西, 舩津: 超 LSI CAD の基礎, オーム社 (1983).
11) 樹下: 論理装置の CAD, 情報処理学会叢書 (1980).
12) 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).
13) 大森, 小池, 佐々木: 超高速論理シミュレータ, 情報処理学会設計自動化研究会資料, 18-2 (1982).
14) 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).
15) Koike, N., Ohmori, K., Kondo, H. and Sasaki, T.: A High Speed Logic Simulation Machine, Compcon '83 Spring, pp. 150-156 (1983).
16) Sasaki, T., Koike, N., Ohmori, K.and Tomita, K.: HAL; A Block Level Hardware Logic Simulation, 20th DA Conference, pp. 150-156 (1983).
17) 小池, 大森, 佐々木: 論理シミュレーションマシンのアーキテクチャ, 情報処理学会論文誌, Vol. 25, No. 5, pp. 864-872 (1984).
18) 小池, 大森, 佐々木: 論理シミュレーションマシンのハードウェア構成, 情報処理学会論文誌, Vol. 25, No. 5, pp. 873-881 (1984).
19) 小池, 大森, 佐々木, 野水: HAL; 超高速論理シミュレータにおける専用アーキテクチャ, アーキテクチャワークショップインジャパン '84.
20) Pfister, G.F.: The Yorktown Simulation Engine: Introduction, Proc. 19th DA Conf., pp. 51-54 (1982).
21) Denneau, M.M.: The Yorktown Simulation Engine, Proc. 19th DA Conf., pp. 55-59 (1982).
22) Kronstadt, E. and Pfister, G.: Software Support for the Yorktown Simulation Engine, Proc. 19th DA Conf., pp. 60-64 (1982).
23) Howard, J.K., Malm, R.L. and Warren, L.M.: Introduction to the IBM Los Gatos Logic Simulation Machine, Proc. IEEE conf. on Computer design, pp. 580-583 (1983).
24) Burggraff, T. et al.: The IBM Los Gatos Logic Simulation Machine Hardware, Proc. IEEE Int'l Conf. Computer Designs, pp. 584-587 (1983).
25) Kon, J. et al.: The IBM Los Gatos Logic Simulation Machine Software, Proc. IEEE Int'l Conf. Computer Design, pp. 588-591 (1983).
26) Howard J.K. et al.: Using the IBM Los Gatos Logic Smulating Machine, Proc. IEEE Int'l Conf. Computer Design, pp. 592-594 (1983).
27) Dunn, L.N.: IBM's Engineering Design System Support for VLSI Design and Verification, IEEE Design & Test of Computers, Vol. 1, No. 1, pp. 30-40 (Feb. 1984).
28) Abramovici, M., Y. Levendel, H. and Menon, P.R.: A Logic Simulation Machine, IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, Vol. CAD-2, No. 2, pp. 82-94 (Apr. 1983).
29) Abramovici, M., Levendel, Y.H. and Menon, P.R.: A Logic Simulation Machine, 19th DA Conference, pp. 65-73 (1982).
30) Levendel, Y.H., Menon, P.R. and Patel, S.H.: Special-Purpose Computer for Logic Simulation Using Distributed Processing, BSTJ, Vol. 61, No. 10, pp. 2873-2909 (1982).
31) Glazier, M.M. and Ambler, A.P.: ULTIMATE: A Hardware Logic Simulation Engine, Proc. 21st DA Conference, pp. 336-342 (1984).
32) 近藤, 土屋, 永谷, 中島: アダプティブ・アレイ・プロセッサ—AAP—, 情報処理学会計算機アーキテクチャ研究会資料, 54-2 (1984).

(昭和 59 年 12 月 12 日受付)