25周年記念論文

HAL：論理シミュレーションマシンの評価†

小池誠彦††　大森健児†††
佐々木徹††††　野水宣良††††

本論文は汎用計算機のCPU，ファームウェア，キャッシュ，主記憶を含めた装置レベルの論理シミュレーションを高速に実行する専用マシンHALについて行った評価について論じている．

実際に装置開発現場において使用される実際のモデルをHALで動作させ自作したハードウェアモニタを用いてデータを収集した．2つの異なるモデルを用いてシミュレーション速度を計測した結果プロセッサ台数を増した時それにつれてシミュレーション速度も向上することを確認した．HALはイベント駆動・ゼロ遅延シミュレーションアルゴリズムをすべてハードウェアで実現し並列処理及びパイプライン処理を行うことにより高速化を実現した．HALのシミュレーション性能を規定するイベント率，プロセッサ稼動率，イベントローカリティなどのパラメータをハードウェアモニタで収集しモデルの動特性を解明しシミュレーション性能との関係について分析を行った．

1. はじめに

著者らは汎用大型計算機など大規模な装置の論理シミュレーションを高速に実行する専用マシン(HAL)を開発した[1)~3)]．本論文では実際の装置に用いられるソフトウェア（検査プログラム）を直接HAL上で動作させ，自作した専用のハードウェアモニタを用いて測定したデータをもとに行った評価について論じる．

論理シミュレーション専用マシンはいくつか実験機の試作[8)~10)]や提案[11)~13)]が行われており一部には製品化されている．LSIの発達によりシステムの設計が大規模化していく中でソフトウェアによる論理シミュレーションの高速化及び大容量化の制約を克服するために近年このような専用論理シミュレータの研究が活発化している．しかし，実際の装置開発現場において実際のモデルを用いた評価結果の報告については非常に少ない．

本論文では論理シミュレーションアルゴリズムの専用ハードウェア化，パイプライン処理化及び並列処理化による効果について，いくつかの実際のモデルをHAL上で動作させデータを収集し定量的にHALの性能評価を行う．このためにまずHALを構成する各ハードウェアモジュールのハード量，シミュレーションアルゴリズムの検討，プロセッサ間ネットワークの評価，シミュレーション性能の評価を行い，HALのシミュレーション性能を規定するパラメータについて分析する．

2. HALの概要

論理シミュレーションマシン(HAL)のアーキテクチャおよびハードウェアについて概要を述べる．

2.1 ブロックレベルシミュレーション方式

HALは“ブロック”と呼ぶ論理的にまとまりのあるゲートの集合体をシミュレーションの単位としている[1)]．少ないハードウェア量で高速化及び大容量化を実現するために次の4つの手法をHALで用いている[2)]．

① ゼロ遅延シミュレーションモデルに基づくレベルオーダリングアルゴリズムを用いる．ブロックを配線の深さの順にレベルわけを行い，レベル順にブロック評価を行う．各ブロックはすべてのイベントが到着した後でたかだか1回だけ評価を行う．

② イベント駆動の手法により変化のあったブロックについてのみシミュレーション処理を行うことで大幅にシミュレーション処理を減少させる．

③ シミュレーション対象のブロックは種類ごとに分け，それぞれプロセッサが分担して並列にブロックの評価を行う．

† Machine Evaluation of HAL; A High Specd Logic Simulation Machine by Nobuhiko KOIKE (NEC Corporation), Kenji OHMORI (Hosei University), Tohru SASAKI and Nobuyoshi NOMIZU (NEC Corporation).

†† 日本電気(株) C & C システム研究所
††† 法政大学経営工学科
†††† 日本電気(株)コンピュータ技術本部

④ 各ブロックのシミュレーション処理を専用のハードウェアモジュール群で実現することによりハードウェアモジュール群が並列にパイプライン処理を行う.

2.2 並列論理シミュレーションアーキテクチャ

ブロックレベルシミュレーションアルゴリズムが持つ並列性を損なわないアーキテクチャとするためにHAL は負荷分散型のマルチプロセッサ方式を採用した[5]. さらにそれぞれのプロセッサは機能分散型のパイプライン処理を行う. HAL のアーキテクチャは以下の特徴を備えている.

① 非同期データフロー型のイベント処理プロセッサ

ブロック間のイベントの伝播, ブロックの入出力状態の管理を行うためにイベントをデータフロー式に処理する専用のプロセッサを実現した. さらにプロセッサ間で状態表やファンアウト表のアクセス競合を無くすためにすべてローカルに分担するブロックのテーブルを分散させている.

② ブロック内部論理演算の並列処理

ダイナミックゲートアレイと呼ぶ専用のハードウェアを用いてブロックの論理機能を実現する. ダイナミックゲートアレイはブロックに含まれるゲート回路を一度に 16 個ずつ並列にゲート評価を行う.

③ 専用 LSI を用いた蓄積交換型の接続ネットワーク

イベントをパケットの形でプロセッサ間を転送する. 2入力2出力を持つ専用のルータセルを用いて多段接続型のネットワークを構築しイベントを並列かつパイプライン式に転送する.

2.3 HAL ハードウェアの概要

図-1 は 32 台の専用プロセッサを用いた HAL のシステム構成を示す. 組合せ回路系のブロックを担当する 29 台の論理プロセッサ, メモリ系のブロックを担当する 2 台のメモリプロセッサと全体の制御を行う 1 台のコントロールプロセッサでシステムが構成される[1]. 論理プロセッサとメモリプロセッサの台数は両プロセッサを加えて 31 台となる範囲でモデルに応じて台数の変更を行うことが可能である. 現在 4 台のメモリプロセッサまで対応できるようにシステムを構成している. これら 32 台の専用プロセッサを多段接続型のルータセルネットワークで結合する. プロセッサ間のデータあるいはイベントの転送はすべて 24 ビットの幅を持つパケットで行う. 論理プロセッサはイベントの送受信処理及びブロックの入出力状態管理を行うノードプロセッサとブロック内部の論理演算を行うダイナミックゲートアレイで構成される. メモリプロセッサはノードプロセッサ及びメモリシミュレータで

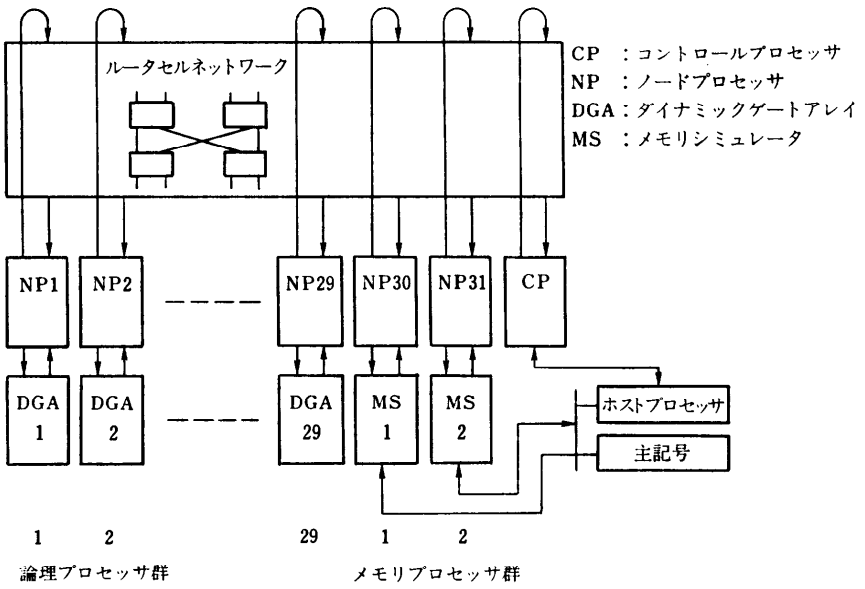

図-1 論理シミュレーションマシン (HAL) の構成
Fig. 1 HAL blockdiagram.

構成される．ノードプロセッサは論理プロセッサと同一のものが使用され，メモリシミュレータは対象マシンの主記憶，キャッシュ，ファームウェアをシミュレートする．メモリシミュレータはホストとなるミニコンピュータの主記憶域の一部を共有し，メモリブロックのメモリデータを収容する．コントロールプロセッサはレベルオーダリング手法に基づき全プロセッサが各レベルごとに同期して並列にブロック評価を行うためのレベル同期機能[1]を実現する．

2.4 動作の概略

図-2 はそれぞれの論理プロセッサにおけるシミュレーション処理の概略を示す．シミュレーション処理をすべてハードワイヤードロジックで実現しハードウェア化している[2]．ノードプロセッサは入力/出力状態及び配線先の情報を3つのメモリに保存する．論理プロセッサは次の手順でシミュレーションを並列に行う．

① イベントを入力すると入力状態表を更新しイベントをセットする．

② ノードカウンタを用いてイベントを順番に取り出しイベントをリセットする．

③ ダイナミックゲートアレイは入力状態値を用いてブロックの論理演算を行い新しい出力状態値を得る．

④ 新しい出力状態値で出力状態表を更新する．

⑤ 出力変化を調べ変化したビット位置を取り出しエンコードする．

⑥ 接続メモリより変化した出力に対する配線先を調べ，イベントをパケットの形で転送する．

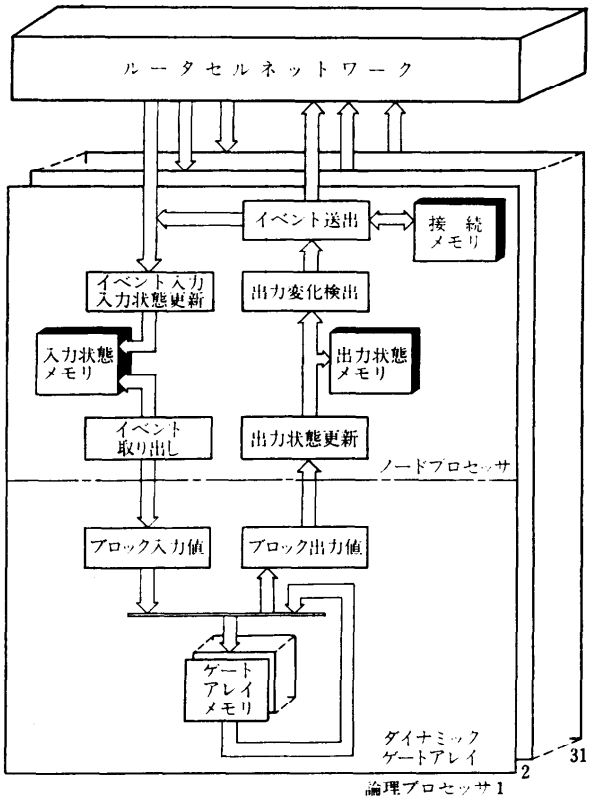

図-2 シミュレーション処理のハードウェア化
Fig. 2 Hardware Algarithm in A Logic Processor.

3. 評価環境

HAL のシミュレーション処理性能を評価するために実際のモデルを用いて動特性の測定を行う．32台の非同期的に動作するプロセッサ群から同時に各種のイベントデータを収集し集計する必要があるので専用のハードウェアモニタを自作した．各プロセッサボードにあらかじめプローブポイントを埋設しておいたのでケーブルを接続するだけで各パイプラインステージの動作状況がモニタできるようになっている．ハードウェアモニタは**図-3** に示すとおり32台のプロセッサから同時に5種類のイベントを計測する．このうち2つは各プロセッサのイベントを個別に集計する32ビット×32組のカウンタ群から構成される．他の3つは全プロセッサからのイベント数を集計する32ビットのカウンタで構成される．各カウンタは別個にイベントの種類を指定可能である．さらにプロセッサマスク機能を用いて特定のプロセッサ群についてのみ選択して計測することができる．個別カウンタにはイベントの集計の他に時間カウントモードがあり，イベントでクロックをゲート制御することによりイベントが立っている間の時間を集計することができる．この時間カウントモードとイベントカウンタを併用することによりプロセッサ稼働時間やブロックの平均処理時間などを求めることができる．本ハードウェアモニタはパソコン (PC 9800) を用いてイベントの指定，結果の出力などを行う．全体カウンタは各ボード 2 MHz まで，また個別カウンタは 10 MHz までのイベントを収集することができる．

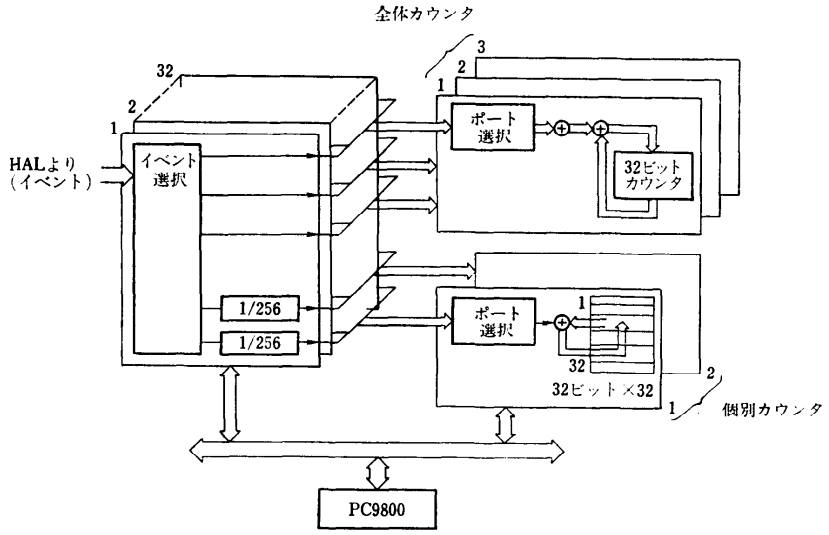

図-3 ハードウェアモニタの構成
Fig. 3 Hardware Monitor blockdiagram.

表-1 シミュレーションモデルの特性
Table 1 Simulation Model Characteristics.

	モデル0	モデル1	モデル2	モデル3
モデルの特性	16ビットカウンタ	計算機A	メモリサブシステム	計算機B
モデルのゲート数		13万ゲート	11万ゲート	8万ゲート
シミュレーション単位	ブロックレベル	ゲートレベル	ブロックレベル	ゲートレベル
シミュレーション時間 (1万クロック) クロック当り		901 秒 9.1 ミリ秒	153.2 秒 1.53 ミリ秒	134.1 秒 13.41 ミリ秒
使用プロセッサ数		31 台	26 台	27 台

評価に4種類のモデルを用いた．**表-1** に各モデルの特性を示すようにモデル0は主にハードウェア及びネットワークの評価デバッグ用に作成したカウンタモデル，モデル2はブロックレベル，モデル1および3はゲートレベルのシミュレーションを行う．

4. ハードウェア量

HAL のプロセッサは論理シミュレーションのアルゴリズムをすべてハードウェアで実現するために従来の汎用計算機と異なり徹底した専用化を行っている．ハードウェア化にあたっては1ボードに収めるためのIC 数の制限がある中で，できるだけ並列化およびパイプライン化による高速化につとめた．**表-2** が HAL に用いた，それぞれのモジュールについてハードウェア量を機能別に示したものである．ノードプロセッサは9個の機能に大別されるが，初めの7つの機能がシミュレーションアルゴリズムを実現するものである．機能5の変化デコード部は16ビット幅のデータから変化したビット位置に対応するコードを順番に求めるものである．変化ビット数に比例した時間で処理できるのでシフトレジスタなどを用いる方法に比べて変化ビット数が少ないとき，特に効果が大きい．機能6のファンアウト処理はダイナミックメモリのリフレシュ制御，リスト処理，ページモードリードなどをハードウェアで実現するために相当数の IC を使用したが，メモリのサイクルの限界近くまで処理を短縮すること

表-2 HAL のハードウェア量
Table 2 HAL IC counts.

		IC 数 (内メモリ)	機能
ノードプロセッサ		401	イベント処理
31台	機能 1	25	パケット入力
	2	75(11)	イベントセット/取出し
	3	44	DGA インタフェース
	4	21(4)	出力メモリ更新
	5	42	変化ピンエンコード
	6	93(30)	ファンアウト処理
	7	33	パケット送出
	8	57	マスタ制御
	9	11	レベル制御
ダイナミックゲートアレイ		208	ブロック内論理演算
29台	機能 1	61(40)	ゲートアレイメモリ
	2	46(11)	論理選択
	3	47	論理演算
	4	26	制 御
	5	28	ノードプロセッサインタフェース
メモリシミュレータ		295	メモリシミュレーション
2台	機能 1	61	SRAM 系のシミュレーション
	2	62	DRAM 系のシミュレーション
	3	29	制 御
	4	70	ホストバスインタフェース
	5	48	状態メモリ
コントロールプロセッサ		191	シミュレーション実行制御
1台	機能 1	65	DMA インタフェース
	2	57	レベル/ブロック制御
	3	43	パケット送受信
	4	26	DMA コマンド解読

ができた．ダイナミックゲートアレイは16入力の汎用ゲートを16組同時に評価を行う．ゲート機能はRAMを用いたテーブル駆動方式を用いており 300 ns ごとにゲート評価を行う．評価結果はレジスタファイルに書き込みを行い次のサイクルの入力となる．1個のダイナミックゲートアレイで8K ゲートまで収容可能である．ゲートあたりの評価に換算すると1ゲートを19 ns で処理することに相当し，専用化の効果が大きい．

5. 支援ソフトウェア

支援ソフトウェアはブロックのレベル付け，ダイナミックゲートアレイのテーブル展開，ブロックのプロセッサ割付け，インタラクティブな実行環境の提供など，専用マシンを効率良く動作させるために重要な役割を担っている．HAL の支援ソフトウェアの開発において考慮した点は従来の CAD システムとの整合性である．モデル作成やシミュレータへのモデル変換に手間がかかってはシミュレータ自体のスピードゲインが相殺されるおそれがある．そのため支援ソフトウェアは従来のソフトウェアシミュレータ (MIXS) と同等のインタフェースをユーザに提供し，従来の DA データベースから自動的に HAL のロードファイルを生成する．支援ソフトウェアのプログラム量および処理時間は表-3 に示すとおりである．テーブル作成などの前処理は ACOS 1,000 で行い，シミュレーション実行環境はホストのミニコンピュータ (MS-50) 上で実現した．

6. シミュレーションアルゴリズムの評価

HAL で採用したゼロ遅延シミュレーションモデルに基づくブロックレベルシミュレーションのアルゴリズムの評価を行う．

6.1 ゼロ遅延シミュレーション方式

今までにいくつかの論理シミュレーションアルゴリズムが提案されているが，専用マシンで実現されているのはゼロ遅延[1),8)]，ユニット遅延[8),9)] およびノミナル遅延[9),10)] の3方式である．専用マシン化にあたっては適用分野，ハードウェアの規模と目標性能によってアルゴリズムが選択される．HAL の主要な適用分野である汎用計算機の装置シミュレーションに目標を絞ってアルゴリズムを検討する．汎用計算機では数十万から数百万ゲートのランダムロジックと数メガバイトのメモリがシミュレーション対象となる．装置全体の論理シミュレーションを行うにはシミュレーション速度を格段に早くしなくてはならない．3つのシミュレーションモデルのうち処理時間が一番少ないのがゼロ遅延モデルであるが，同期式回路にしか適用できないと言う面を持っている．しかし，幸いなことに多くの汎用計算機は同期式の回路網で設計されており論理の検証にはゼロ遅延モデルで十分に役立つ．ゼロ遅延モデルではレベルオーダリングの手法によりシミュレーション回数が少なくてすむこと，状態メモリを更

表-3 支援ソフトウェア量および処理時間
Table 3 Software Size and Execution Time for Support System.

プログラム容量		フォートランライン数		
ACOS 1,000	ブロックモデル生成	13.5Kライン		
	リンケージ	6.5Kライン		
	ブロック間接続表生成	16.5Kライン		
	ブロック割当て	4.3Kライン		
MS 50	コマンド処理	15.8Kライン		
処 理 時 間		モデル1	モデル2	モデル3
ACOS 1,000	リンケージ	678 秒	60 秒	533 秒
	レベルオーダリング	108 秒	12.2 秒	85 秒
	ブロック割当て	85 秒	33.6 秒	67 秒
	HAL ロードファイル生成	135 秒	34.2 秒	106.1 秒

表-4 テーブル方式とコンパイル方式の比較
Table 4 Comparison between Table Driven and Compiled Method.

	テーブル方式	コンパイル方式
オペランドの記憶	テーブル	マルチポートメモリ
オペランドのアクセス	直 接	ポインタ
オペランドの更新	ファンアウト数だけ	1 回
イベント駆動	○	×
パイプライン処理	非同期	完全同期
命令長	入力数×オペランド値	入力数×オペランドアドレス
ファンアウト	ファンアウト数	1

新しながら状態を取り出せるので状態メモリを多重に持つ必要がないこと，さらに同一レベルに属する素子はすべて並列にシミュレーションを行うことができるので並列性が高いこと，などの特徴を持っている．このような点から HAL はゼロ遅延モデルを採用した．IBM の YSE[6] なども同様のモデルに基づいているが，HAL との大きな違いは HAL がテーブル方式を用いているのに対し YSE などはコンパイル方式を用いている点にある．表-4 に両者の比較を行った．コンパイル方式はゲートの評価をマシンの1命令にコンパイルし実行する．オペランドとなるゲートの入力はオペランドメモリから命令実行時にそのつど取り出しを行う．複数のオペランドを取り出すためにオペランドメモリを同時複数アクセス可能とするために多重化する必要がある．一方イベント伝播についてみると，コンパイル方式は出力変化の伝播をプロセッサ当り1回ですますことができる．それに対しテーブル方式では入力となるすべてのゲートに出力変化を伝播させる必要がある．しかし，一般的に言ってファンアウト数は少なくそれほど伝播処理は多くないこと，さらにテーブル方式によればイベント駆動の手法が利用できるので不要なシミュレーションを大幅に減少させることが可能となる．

HAL では 4 章の表-2 で示されているように入力/出力状態メモリのためにわずか 15 個のメモリ IC しか使用していない．内訳は 1K×48 ビットの入力状態/フラグに 12 個，1K×32 ビットの出力状態に 3 個を使用している．さらにファンアウトメモリには大容量の DRAM を用いることができるので（30 個で 64K ファンアウト数を収容）少ないハードウェア量で効率の良い処理が可能である．

6.2 ブロックレベルシミュレーション

処理単位の粒度をどこに設定するかは処理単位内の処理量および処理単位間の通信量のバランスを考えて決める必要がある．処理単位が小さいとそれぞれの処理時間は短いが，通信オーバヘッドが大きくなりすぎる問題がある．また，反対に大きすぎるとそれぞれの処理に時間がかかりすぎたり並列性が少なくなったりする．HAL の処理単位としては中程度の粒度を持つブロックレベルを採用した．ダイナミックゲートアレイは 16 入力を持つ汎用ゲートを 16 個同時に評価を行う．1つのブロックの論理機能はこのゲート群の評価をブロックの入力側からレベル順に続けながら実現する．中間結果はレジスタファイルに保存し次の入力となる．ダイナミックゲートアレイは各レベルのすべての信号線を入力として扱う．もし，16 入力を越える時

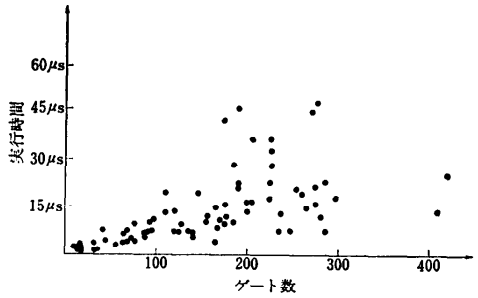

図-4 DLA のブロック評価時間
Fig. 4 Block Evaluation Time.

は16入力ずつに分けて評価を行う．したがって，各レベルの信号線数が多くなるとブロックの評価時間は長くなる．1回のゲート群評価時間をτとし，レベルiのゲート群の数をL_iとするとダイナミックゲートアレイでのブロック評価時間T_sは下式で与えられる．

$$T_s=\tau\sum_{i=1}^{n}Li-1\cdot Li \qquad (1)$$

$$\begin{cases}L_0\text{；入力レベル}\\ L_n\text{；最終レベル}\end{cases}$$

図-4 は実際に一つの計算機に用いられるブロックについてブロックのゲート数とシミュレーション時間をプロットしたものである．ブロックに含まれるゲート数が増大するとシミュレーション時間は増大する傾向を示している．しかし，シミュレーション時間はゲート数だけでなくブロックの回路構成によっても大きく影響されることがわかる．このモデルの扱うブロックの平均ゲート数は HAL が初め仮定していたゲート数 (75 ゲート) よりやや多くなっている．したがって，後の評価結果でわかるように平均シミュレーション時間は 6.9 μs となっている．今回の評価に用いたモデル1および3はゲートレベルを用いている．HAL でゲートレベルを扱うと，その能力をかなり犠牲にして使うことになる．しかし，ゲートレベルでもダイナミックゲートアレイに複数個のゲートを入れることにより並列性を生かすことが可能である．たとえば，4入力ゲートであれば8ゲートを入れることが可能である．

7. プロセッサ間接続ネットワークの評価

ルータセルネットワークの性能を評価するためにモデル0を用いて実験した．実験には図-5 で示す8入力の小型のネットワークを使用した．基板構成の都合上4段構成となっている．一つのパケットは 400 ns ごとに投入することが可能であり4段を通過するのに 3.15 μs を要している．これは一段のルータセルを通過するのに 790 ns かかっていることになる．蓄積交換型のネットワークは通常のプロセッサメモリ間のように早い応答が必要なところは不向きであるが，HAL のように応答性より転送率を重視する用途に適していると言える．

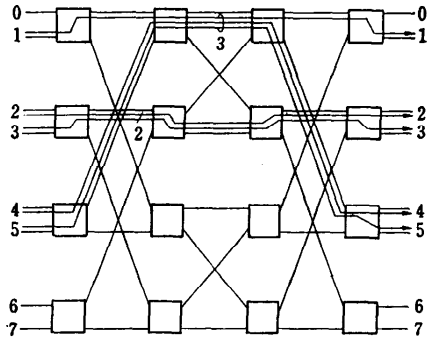

図-5 ネットワークにおけるアクセスパスの競合
Fig. 5 Access Path conflicts in the Route Cell Network.

評価に用いたモデル0はイベントを大量に発生させるために作成したモデルでありイベントのローカリティが56%のものと 0％のものとがある．外部に出たイベントはネットワークを介して再び自プロセッサに戻って来る．ローカリティ 56% の場合は 1.5 μs ごとに外部にパケットが出る．また，ローカリティ 0％のものは 800 ns ごとにパケットが出てくる．図-6 がプロセッサ数と1秒当りのシミュレーションクロック数の関係を示している．ローカリティ 56% のものはプロセッサ数7台までほとんど性能低下が見られないの

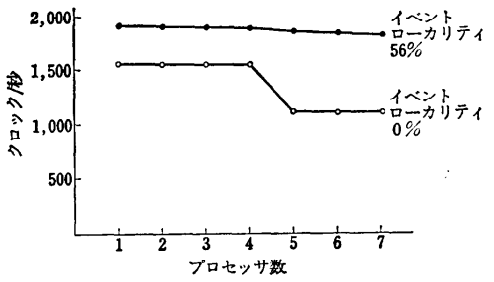

図-6 パケット競合に起因する性能低下
Fig. 6 Performance Degradation due to Packet Conflict.

に対しローカリティ0％のものは初め約18％性能低下があり，しばらく一定し5台目になって急に約42％の性能低下となり再び一定する．これには次の2つの要因によるものと考えられる．最初の低下はパケットが外部に出されたための遅れによるもの，第2の性能低下はパケットの通過ルートの競合によるものである．先に示した図-5にそれぞれのプロセッサの通過ルートを示している．各ルータセルのパケット転送率の制約からルートが2本までなら性能低下なしでパケットが通過するが，3本になるとルータセルの転送率のリミットで性能が抑えられてしまうためと考えられる．

本実験で用いたモデルは全プロセッサが同時に同一のルートを使用する最悪のケースである．実際のモデルではこのようなことはほとんど起こらないが，実験に用いたネットワークはルートが集中しやすい面があるので代替ルートを用いてパケットを分散させる必要がある．

8. 実行性能の評価

モデル2及び3についてプロセッサ台数を変えて実行性能の評価を行った．モデル2はブロックレベルに基づき全体で4,462ブロックを35のレベルに分けてシミュレーションを行う．レベル当り平均255ブロック存在するので27台のプロセッサを用いても並列度は高いと言える．それに対しモデル3はゲートレベルを用いているのでレベル数が140と多くブロックが細分化される．しかも毎クロック6回のレベル戻りが行われるのでシミュレーション条件は良いとは言えない．モデル3ではゲートが11,731ブロックにまとめられ，レベル当り84ブロックになる．

1）シミュレーション回数

モデル2(ブロックレベル)についてプロセッサ数と1秒当りのシミュレーションクロック数を示したのが図-7である．プロセッサ数を増すとそれにつれてシミュレーションクロック数も増大してくる．26台構成の時毎秒655クロックのシミュレーションを行っている．プロセッサ数が多い時の性能向上の度合はしだいに緩くなる傾向を示している．これはモデルの規模がまだ小さいのでプロセッサ数を増大してもブロックの負荷分散の不均衡によるレベル待ち損が多いためと考えられる．

図-8はモデル3について同様の測定を行ったものである．プロセッサ数を増すとシミュレーションクロック数も増大し並列化の効果があることを示している．ただし，レベル数が多いこととレベル戻りがあることでシミュレーションクロック数は27台構成時75クロック/秒とブロックレベルに比べ1桁程度性能が低下している．

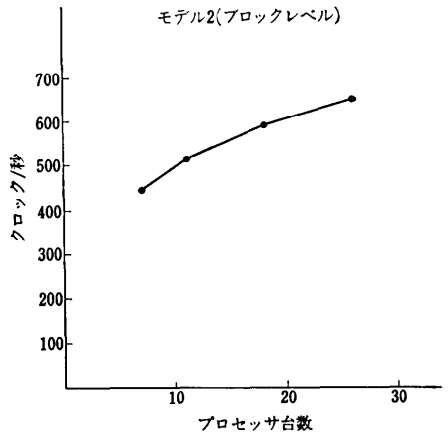

図-7 プロセッサ数とシミュレーション回数
Fig. 7 Simulated Clocks versus Processor Counts in case of Block Level Simulation.

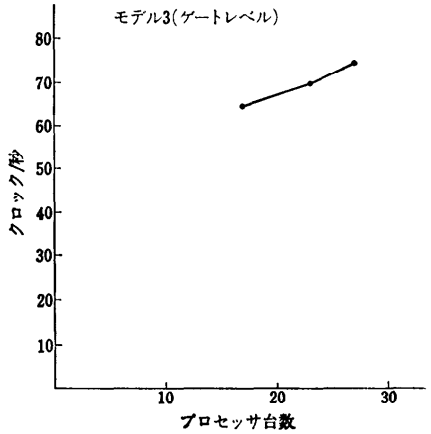

図-8 プロセッサ数とシミュレーション回数
Fig. 8 Simulated Clocks versus Processor Counts in case of Gate Level Simulation.

2）プロセッサ稼働率

パイプラインプロセッサの稼働率を定義するのは難しいが，ここではすべてのパイプラインステージが空の状態をアイドルとし1つでもパイプラインステージが動作状態になるとプロセッサ稼働中とした．プロセッサ稼働率を計測するためには各プロセッサから出る

レベル終了信号を使用することができる．レベル終了信号はプロセッサ内のすべてのパイプラインステージ及び FIFO，バッファがイベントなしの時にアクティブとなる．コントロールプロセッサは全プロセッサがレベル終了状態になると，ただちに次のレベル開始を指令しレベル終了状態は解除される．したがって，レベル終了状態にある時間はレベル同期待ち損による無駄時間となり，全シミュレーション時間から無駄時間を引いたものが稼働時間となる．**図-9** はモデル2に対するそれぞれのプロセッサの稼働率を示している．プロセッサ数が増大すると各レベルに対するブロック数が少なくなるのでレベル待ち損が増大し稼働率が低下するものと考えられる．7台構成の時，平均の稼働率が52%であるのに対し26台構成では27%に低下する．プロセッサ番号 30(メモリプロセッサ) の稼働率が他のプロセッサと異なるのはメモリプロセッサは1台しかないのでプロセッサ構成台数によらずメモリブロック数およびイベント数が一定なためプロセッサ数が増大すると相対的には稼働率が高くなるためである．**図-10** はモデル3に対するプロセッサ稼働率を示している．モデル2の場合と大きく異なるのはプロセッサ構成台数によらず稼働率がほぼ一定している点である．これはイベントがプロセッサ全体に広がって，プロセッサ当りのシミュレーション数は低くとも，さみだれ式にイベントが到着しいつもプロセッサが稼働状態におかれるためと考えられる．

3) パケットのローカリティ

HAL ではイベントパケットの宛先が自プロセッサの場合はルータセルネットワークを経由せず直接自プロセッサのパケット入力部へパケットを転送することにより処理効率を高めている．モデル 2, 3 についてそれぞれ外部出力パケット数と内部パケット数の配分を調べたものを**図-11, 12** にそれぞれ示す．図-11 の

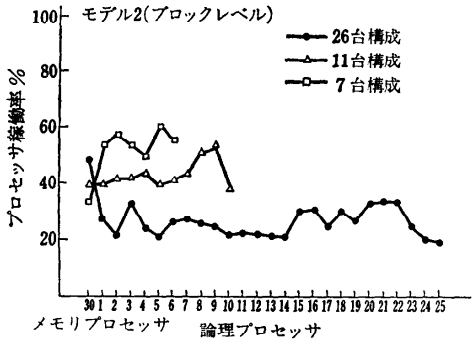

図-9 プロセッサ稼動率
Fig. 9 Processor Utilization Factors in Case of Block Level Simulation.

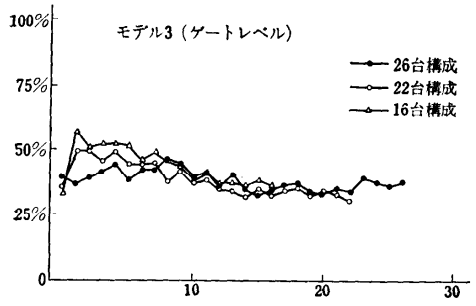

図-10 プロセッサ稼動率
Fig. 10 Processor Utilization Factors in Case of Gate Level Simulation.

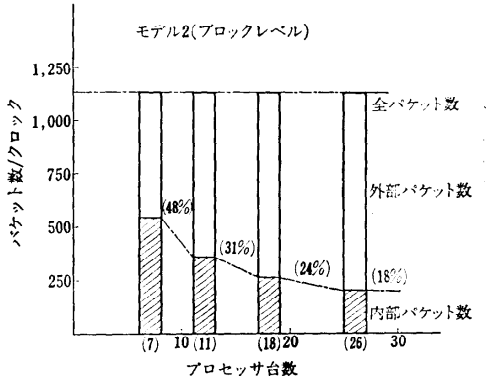

図-11 プロセッサ構成台数とイベントパケット数
Fig. 11 Event Packet Locality in Case of Block Level Simulation.

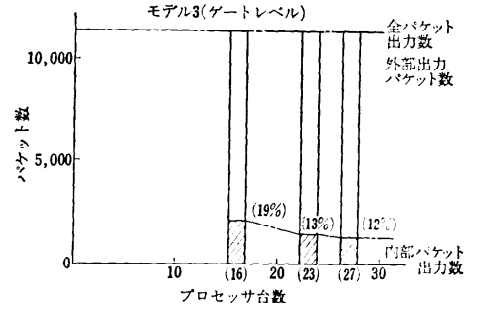

図-12 プロセッサ構成台数とイベントパケット数
Fig. 12 Event Packet Locality in Case of Gate Level Simulation.

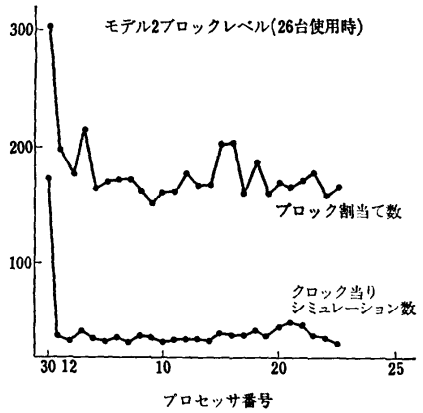

図-13 ブロック割当て数とシミュレーション数
Fig. 13 Block Allocations and Simulation Counts.

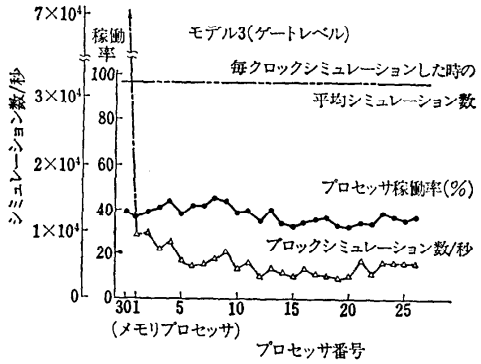

図-14 プロセッサ稼動率とシミュレーション数
Fig. 14 Processor Utilization and Simulation Counts in case of Gate Level Simulation.

モデル2の場合7台構成時ローカリティが48%なのに対し26台構成では18%に減少している．もし，まったくパケットの行先がランダムの場合は1/26=3.8%となるはずであるが，実測値は18%となっておりローカリティが高く部分回路にイベントがかなり集中する傾向にあることを示している．図-12はゲートレベルの場合の分布を示している．ブロックレベルに比べローカリティが低目に出ているが，それでもイベントのローカリティはかなり高いことがわかる．ローカリティがかなり高いのでパケットのバイパス径路を実現するためのハードウェアの効果（表-2のノードプロセッサの機能1および7のうちIC数約20個分に相当）が大きいことを示している．

4）イベント率

HALのイベント駆動方式の評価を行う．**図-13**がモデル2における各プロセッサごとに割合てられたブロック数と実際に1クロック内にシミュレーションされたブロック数の平均を示している．両者の割合がイベント率となる．この例ではイベント率の平均が13.7%となる．**図-14**はモデル3におけるシミュレーション数の分布を示している．イベント率は論理系が16.9%であるのに対しメモリ系は197.5%と突出している．これは1クロックのシミュレーションに6回のレベル戻りが行われるのでメモリ系は毎クロック2回ずつシミュレーションされていることになる．この理由は論理系がゲートレベルであるのに対しメモリ系はブロックレベルとなっているのでレジスタファイル系などのメモリブロックは1クロック内に複数回シミュレーションを行う必要があるためである．イベント駆動を実現するために表-2のノードプロセッサの機能2のうちメモリ1個，制御回路約40個を費やしているがその効果は大きいと言える．

5）ブロック評価時間およびファンアウト数

モデル2のブロック評価時間は平均で6.9 μsの値が得られた．これは**6.2**節で得たブロック評価時間分布のうち80〜120ゲートのブロックが多く用いられていることによる．一方モデル3ではゲートレベルであるのでほとんどがダイナミックゲートアレイの1サイクル（300 ns）で終了する．

1つのブロックのシミュレーションの結果発生するイベントはモデル2で1.78イベント，モデル3で6.12イベントとなっている．この数値はモデルの特性によるものと考えられる．モデル2はメモリサブシステムであるので構造が比較単純でメモリへのアクセスパターンが単調なためイベントが少ないと思われる．モデル3は中型計算機のCPU，ファームウェア，主記憶，キャシュを含んでおり検査ソフトウェアを動かしているので状態変化が多いと思われる．

9. むすび

HALの性能およびアーキテクチャ，ハードウェアの評価結果について述べた．

異なる2つのモデルを用いて詳細に評価した結果プロセッサ数を増大した時に2つのモデルとも単位時間当りのシミュレーションクロック数が増加することを確認した．HALはブロックレベルに基づく専用マシンであるが，ゲートレベルシミュレーションにもHALが十分使えることが判明した．HALはすでに

いくつかの装置開発に実際につかわれ，かなりの実績をあげている．

今までの使用経験から今後の検討課題をあげると次のとおりである．

- ダイナミックゲートアレイの大容量化

ブロックシミュレーション方式は効率良い方式であることを確認した．しかし現状のシステムでは各ブロック当りのゲート数が増大するとダイナミックゲートアレイのメモリ容量が不足する恐れがある．今後は数倍から数十倍の容量増大をはかるとともに，その場合での効率良い処理手法の検討が必要となろう．

- 多値論理の扱い

HAL は現状では２値の論理状態しか扱うことができない．双方向素子・ワイヤード論理・バスなどをモデル化する場合には２値では不十分な場合がある．そのため多値論理の導入が考えられる．しかし多値化にあたっては状態テーブルのメモリ容量の増大，ダイナミックゲートアレイのテーブル容量の増大の問題があり，今後ブロック当りのビット幅および論理レベル数を含めた総合的な検討が必要である．

- レベル戻りの効率化

レベルオーダリング手法は効率が良いがブロック間で相互に信号の往復があるとレベル戻りが必要となり，シミュレーション時間を悪化させる要因となる．レベル戻りのイベントを検出し自動的にレベル戻りを行う機能をハードウェアで実現する，などレベル戻りの効率化を検討する必要がある．

- ノミナル遅延アルゴリズムの実現

VLSI チップの論理シミュレーションなどではノミナル遅延シミュレーションを必要とする利用者も多い．大規模な装置レベルのシミュレーションにおいてノミナル遅延モデルがどれだけ有効であるか疑問であるが HAL の適用範囲を広げるという意味ではノミナル遅延モデルの実現も一つの課題である．タイムホイールに基づくノミナル遅延モデルとレベル同期方式に基づくゼロ遅延モデルの両者をいかに効率良く同時にシステムがサポートするかの検討が必要である．

謝　辞　最後に研究の機会を与えて下さった日本電気(株)C＆C システム研究所箱崎主管研究員，山本部長，コンピュータ技術本部斉藤本部長，桑田部長，杉本部長代理，情報処理システム技術本部山田技師長，メディカルシステム事業部北野事業部長に深謝します．数々の提案をいただいた第二 OA 事業部赤井課長，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・製作を担当して下さった三野輪氏，近藤氏，幅田氏，ソフトウェアサポートシステムを担当して下さった堀田氏，髙崎主任，伊藤氏，田中主任に感謝します．

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のアーキテクチャ，情報処理学会論文誌 Vol. 25, No. 5, pp. 864-872 (Sep. 1984).
2) 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 (Sep. 1984).
3) 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マシンアーキテクチャ，アーキテクチャワークショップシンポジウム，pp. 3-10 (Nov. 1984).
4) 小池，大森，佐々木：論理シミュレーションマシン，電子通信学会研究会資料，EC 82-42.
5) Koike, N., Kondo, K. and Sasaki, T.: A High Speed Logic Simulation Machine, pp. 446-451, Compcon 83 Spring.
6) Sasaki, T., Koike, N., Ohmori, K. and Tomita, K.: HAL; A Block Level Hardware Logic Simulation, Proc. 20th DA Conf., pp. 150-156.
7) Nomizu, N., Sasaki, T., Itoh, O., Tanaka, H., Koike, N. and Ohmori, K.: Block Level Hardware Logic Simulator-its Application and Results, Proc. ICCAD 84, pp. 254-256 (1984).
8) Pfister, G.F.: The Yorktown Simulation Engine: Introduction, Proc. 19th DA Conf., pp. 51-54 (1982).
9) Howard, J.K., Malm, R.R. and Warren, L.M.: Introduction to the IBM Los Gatos Logic Simulation Machine, Proc. IEEE Conf. on Computer Design, pp. 580-585 (1983).
10) Blank, T.: A Survey of Hardware Accelerators Used in Computer Aided Design, IEEE Design & Test, pp. 21-39, (Aug. 1984).
11) Abramovici, M. Levendel, Y. H. and Mennon, P.R.: A Logic Simulation Machine, 19th DA Conf., pp. 65-73 (1982).
12) Lazier, M.E.G. and Ambler, A.P.: ULTIMATE: A Hardware Logic Simulation Engine, 21th DA Conf., pp. 336-342.
13) Marino, J.T.: Low-cost Hardware Logic Simulator/Fault Grade Machine, Proc. Custom Circuit Conf., pp. 242-244 (1984).

(昭和 60 年 7 月 1 日受付)