HAL Ⅲ：機能レベル・ハードウェア・シミュレータ・システム†

高崎 茂†† 野水宣良†† 平林良啓††
石倉 浩††† 蔵下正広††
小池誠彦†††† 中田登志之††††

本論文は機能レベルの高級記述言語 (FDL: Functional Description Language) をそのまま超高速にシミュレーションできるハードウェア・シミュレータ HAL Ⅲのシミュレーション方式，システム構成，シミュレーション実行過程，システム性能および HAL Ⅲの応用領域について述べている．HAL Ⅲは31台構成の場合，ゲートレベル・ソフトウェア・シミュレータの約 10000 倍以上の実行速度をもち，超大型計算機の I/O 部分も含めたシステムモデルやマルチ・システムモデルでもシミュレーションできる．最大は127台まで拡張可能である．さらに HAL Ⅲは効果的に装置診断プログラムを走行させるために命令レベル・シミュレータとリンクしてより一層高速に実行できる機能，コンカレント法をマルチプロセッサで効果的に実行できるようにして，ソフトウェア・シミュレータの約100倍の性能が予測できる故障シミュレーション機能，高信頼化装置の検査のために必要な診断プログラムの品質を定量的に見積ることができる網羅率測定機能を備えていて，広範囲に適用可能なハードウェア・シミュレータである．HAL Ⅲは高級記述言語をそのままシミュレーションできるので設計の上流工程での検証が十分でき，下流工程で論理合成を駆使する VLSI 時代の設計法に良く適合した有益なシミュレータであり，実際の開発に広く使われている．

1. はじめに

VLSI に代表される高集積化，高機能化は従来の設計法を変革しつつある．従来の設計法においては，製造時の開発品質がそれほど高くなくても検査時修復が可能であった．ところが LSI/VLSI 時代においては，致命的な論理バグはチップの再作成となるので，開発費の増大や開発納期の遅れとなり大きな問題となる．このため，設計の上流工程より設計品質に大きな注意がはらわれてきている．すなわち，従来のようにブロック図を逐次詳細化し，人手により詳細回路を作っていたものを，設計言語を使って機能記述し，十分論理検証した後，詳細回路を論理合成を使って自動的に作っていく形に移りつつある．このためには，設計言語を使って作られた計算機モデルを全体としてシミュレーションでき，実機検査で行われるのと同様な機能テストをシミュレータ上で実施し，十分な検証をしておく必要がある．これを達成するには，従来のようなゲートレベル主体のシミュレータに変わって，機能レベルの記述を超高速にシミュレーションできるシミュレータが必要である[1)~7),9)]．

このような VLSI 設計での必要性を満たすために，機能記述をシミュレーションできるハードウェア・シミュレーション・マシン (HAL Ⅲ) が開発された．これは 31 台構成の場合，ゲートレベル・ソフトウェア・シミュレータの約 10000 倍の実行速度を持ち，超大型計算機モデルの I/O 部も含めたシステムモデルやマルチ・システムモデルでもシミュレーションできる．さらに HAL Ⅲは，効果的に装置診断プログラムを走行させるために命令レベル・シミュレータとリンクして実行する機能（この時は単独実行に比べて，さらに数十倍高速化できる．），コンカレント法をベースにし，ソフトウェアの約100倍の性能を目指す故障シミュレーション機能，高信頼化装置のために必要な診断プログラムの品質を定量的に見積れる網羅率測定機能を備えている[11),12)]．本論文では，2章で VLSI 時代の設計法と HAL Ⅲの関係，3章で HAL Ⅲのシミュレーション方式，4章で HAL Ⅲのシステム構成，5章で HAL Ⅲのシミュレーション実行過程，6章で HAL Ⅲのシステム性能，7章で HAL Ⅲの応用について述べている．

† HAL Ⅲ: Functional Level Hardware Simulator System by SHIGERU TAKASAKI, NOBUYOSHI NOMIZU, YOSHIHIRO HIRABAYASHI (Computer Engineering Division, NEC Corporation), HIROSHI ISHIKURA (NEC Corporation, Kofu), MASAHIRO KURASHITA (Computer Engineering Division, NEC Corporation), NOBUHIKO KOIKE and TOSHIYUKI NAKATA (C&C Systems Research Laboratories, NEC Corporation).

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

2. VLSI 時代の設計法と HAL Ⅲ の関係

ここでは VLSI 時代における設計法と，このような設計法におけるシミュレータへの要求および HAL Ⅲの開発背景について述べる．

2.1 VLSI 時代の設計法と HAL Ⅲの関係

従来の設計法は図1（a）に示すように方式設計，機能/ブロック設計，詳細設計とすべて人手により行われ，その後論理検証を行って，レイアウト/テスト設計へと進むのが一般的であった．したがって，このような設計法においては，設計の下流工程での論理検証，すなわち，詳細回路をシミュレーションするゲートレベル・シミュレーションが主力であり，ソフトウェア・シミュレータをはじめとして，最近ではハードウェア・シミュレータも使われてきている[4),5),9)]．

一方，近年の VLSI 設計では多機能化（大規模化）と同時に品質の高い設計が強く求められる．VLSI 設計を従来どおりの方法で行うと，論理検証で多数の論理バグが検出されるため，詳細回路の修正および論理検証というサイクルが多くなり，開発期間やコストの面で大きな問題となっている．したがって，これらの問題を解決する方法が必要であるが，その鍵となるのが高級記述言語と論理合成である[8),10)]．

これらの最近の進歩はめざましく，従来の設計法を変えてきている．すなわち，機能設計を高級記述言語で行い，十分論理検証して，詳細回路は設計言語を入力して論理合成で行う方法である．ここでは設計の上流過程で論理検証が行われる．この模様は図1の（b）に示されている．この方法をとれば多機能で品質の良い VLSI が短期間で設計できる．したがって，この設計法では設計言語を気軽に修正して，すぐ検証結果が得られる超高速の機能シミュレータが強く求められる．HAL Ⅲはこのような VLSI 時代の設計方法に合うように開発された超高速シミュレータである．

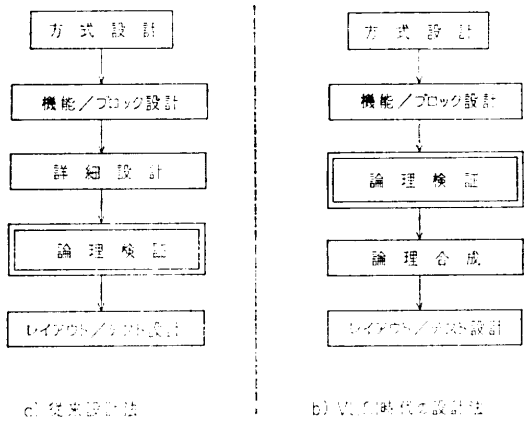


図1 設計法の変遷
Fig. 1 Design methodology change.

2.2 HAL Ⅲ開発背景と旧型機との相違

上記のような従来にない新しい VLSI 設計要求を満たすために，次のような HAL Ⅲ開発方針がとられた．

i） レジスタトランスファモデルの使用

HAL Ⅲはレジスタトランスファ言語 FDL をシミュレーションモデルとして採用した．これは従来のゲートやブロックモデルのマシンと大きな差異である．この上位モデルを採用した理由は次のとおりである．

a） 上位レベルの論理検証

前に述べたように論理合成が普及し言語が設計されるようになると，上位レベルの論理検証がなくてはならないものになる．

b） 全シミュレーション時間の削減

シミュレーションモデルとしてゲートまたはブロックモデルを使っていた場合，モデルを最終のゲートレベルまで落しておかねばならないことやモデル作成時間が構成単位（ゲート）が膨大になるため非常にかかるという問題がある．したがって，これらの問題が短期間でモデルを作成し，くり返してシミュレーションする場合のネックとなる．設計記述言語を使うとこれらの問題は大幅に削減できる．

c） モデル修正の容易性

シミュレーションモデルがレジスタトランスファレベルの場合，モデルの修正は言語を書き直すだけですむので，擬似ハードウェアの追加/削除等が容易であり，様々なシミュレーションを可能にする．

ii） 回路分割に基づく並列実行方式の採用

大規模なレジスタトランスファモデルを高速にシミュレーションするため，回路分割に基づく並列実行方式を採用している．これは旧型機でも採用された方法であり，シミュレータの容量と速度の両方を向上させることを可能にしている．

iii） 高速アルゴリズムの採用

並列化を十分生かし，高速のシミュレーションを実行するため，レジスタトランスファレベルでのゼロディレイ，レベルソート，イベントドリブン法が採用されている．これらのアルゴリズムはシミュレーションの同時性，すなわち並列実行の比率を増加させ高速

化を実現している．HAL Ⅲは4値（0, 1, X（不定値），Z（ハイ・インピーダンス））を採用し，故障シミュレーション等の広範囲な利用を可能にしている．

[HAL Ⅲと旧型機（HAL Ⅱ）との相違]

HAL Ⅲと旧型機 HAL Ⅱとの相違は次のようである．

（注） HAL と HAL Ⅱの相違は文献7）で記述済．）

- 旧型機はシミュレータの基本処理である評価の単位（粒度）がブロックまたはゲートだったのに反し，HAL Ⅲは FDL である．したがって，シミュレーション処理を実行するプロセッサの構成および演算方式は大幅に異なる（詳細は後述4章）．
- 旧型機はメモリー処理に対し専用のプロセッサを持ち，その容量も全部で 4 MB であったが，HAL Ⅲでは，プロセッサにメモリーを持っているのでプロセッサとしては論理/メモリーの区別なく一種類であり，さらにシステム最大容量は 254 MB である．したがって，メモリー処理に対するアクセスネックがなくなる．
- 旧型機は論理メモリープロセッサ合わせて最大 63 台であったが，HAL Ⅲでは 127 台である．
- 旧型機では故障シミュレーションや網羅率測定はできなかったが，HAL Ⅲではそれらを可能にしている．
- 旧型機では論理の最大容量がブロックモデルで約10ミリオンゲートであったが，HAL Ⅲでは20ミリオンゲート以上となる（これは FDL 1 文を 10 ゲートと想定した場合で，もっとまとまった機能記述なら容量はさらに増大する．）．
- 旧型機ではプロセッサ当りのタスク総数や種類数がそれぞれ 1 K，128 であったが，HAL Ⅲでは 16 K．
- 旧型機ではタスク（ブロック）の入出力信号数に制約（128）があったが，HAL Ⅲでは 2 K まで可能である．

3. HAL Ⅲのシミュレーション方式

3.1 機能記述言語：FDL (Functional Description Language)

HAL Ⅲで実行される FDL とはハードウェアモデルをノードという単位で機能記述できる言語であり，日本電気(株)で開発されたものである[8]．ノードとは**図2**に示されるような機能単位である．

FDL は7種類の文より成り立つ．

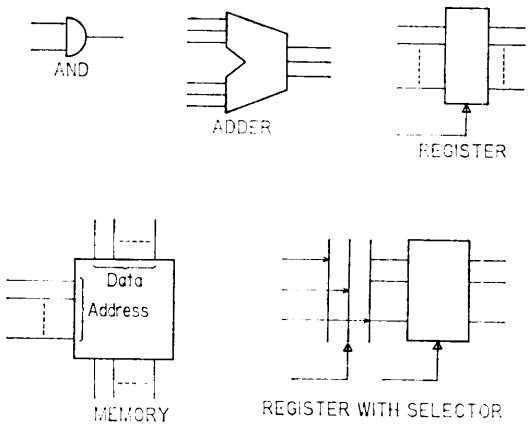


図2 ノード例
Fig. 2 Node examples.

i） INPUT…入力端子記述
ii） OUTPUT…出力端子記述
iii） INOUT…入出力端子記述
iv） REGISTER…レジスタ記述
v） TERMINAL…組合せ回路記述
vi） MEMORY…メモリー記述
vii） MODULE…ライブラリー参照記述

FDL を演算子（オペレータ）からみると次のようなものがある．

A） 論理演算子…例えば，AND, OR, EXOR, …
B） 機能演算子…例えば，SHIFT LEFT, SHIFT RIGHT, …
C） 算術演算子…例えば，ADD, SUBTRACT, …
D） 比較演算子…例えば，EQUAL，GREATER THAN, …
E） タイミング演算子…例えば，GO UP，GO DOWN, …

図3で示される FDL ノードの入出力関係は次のようになる．

$$O_{1\sim m} = I_1.\,OP_1.\,I_2.\,OP_2.\cdots.\,OP_q.\,I_n \qquad (1)$$

ここで $O_{1\sim m}$ ：出力信号
$I_{1\sim n}$ ：入力信号

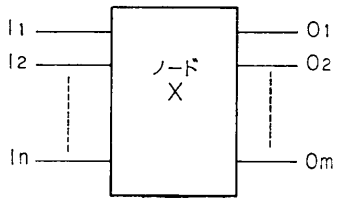


図3 ノード
Fig. 3 A node.

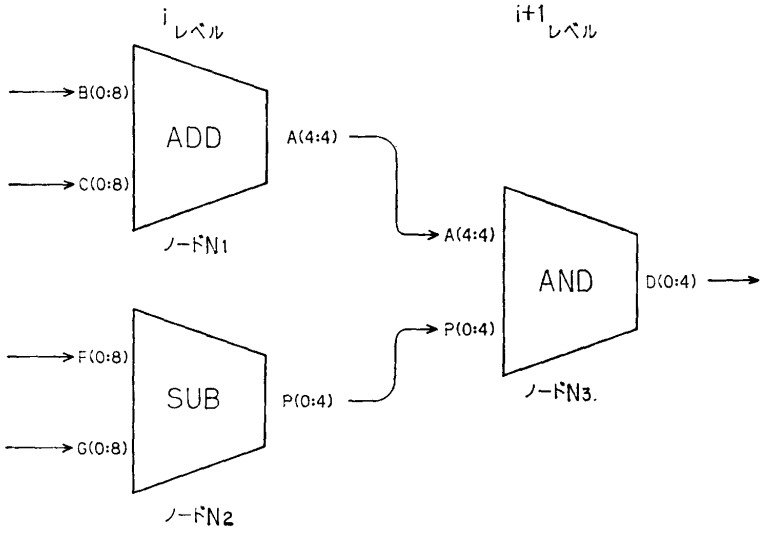


図 4 レベル付されたノード
Fig. 4 Levelized nodes.

$OP_{1\sim\varphi}$: FDL オペレータ

ノードとは FDL 文全体をさす.

以上からわかるように FDL はゲート記述からレジスタ・トランスファ記述まで可能な高級記述言語である.

3.2 FDL 記述のシミュレーション方法

ここでは簡単な FDL 記述を示し，それらがどのようにシミュレーションされるかを示す.

[FDL 記述例]

```
≀
A (0:8)=B (0:8). ADD. C (0:8) ; Bの8ビットとCの8ビットを加算する
P (0:8)=F (0:8). SUB. G (0:8) ; Fの8ビットからGの8ビットを減算する
D (0:4)=A (4:4). AND. P (0:4) ; Aの4ビットから4ビットとPの0ビットから4ビットをANDする
≀
```

これら FDL 記述（ノード）は信号の連なり関係を考慮して，入力端子（またはレジスタ）から出力端子（またはレジスタ）に向かってレベルが付されていく.

上記の FDL 記述に対しては，**図 4** に示すようなレベルが付される.

これらはソフトウェア上で実行され，HAL Ⅲへ送られる．レベル付されたノードは HAL Ⅲ上で入力端子からレジスタ（または出力端子）に向かってレベルごとに並列に実行される．図 4 において，i レベルのノード N_1 と N_2 は同時に実行される（シミュレーション実行過程の詳細については 5 章で述べられている.）.

4. HAL Ⅲのシステム構成

HAL Ⅲ 32 台の概観を**図 5** の写真に示す．HAL Ⅲ全体のシステム構成は**図 6** のようである.

4.1 ハードウェア構成

[HAL Ⅲプロセッサ・ハードウェア構成]

シミュレーション処理は以下の 3 つの段階より成る.

i） イベント検出：入力信号が変化したノードが検出され対応するノード入力が更新される.

ii） ノード評価：ノードシミュレーションが実行

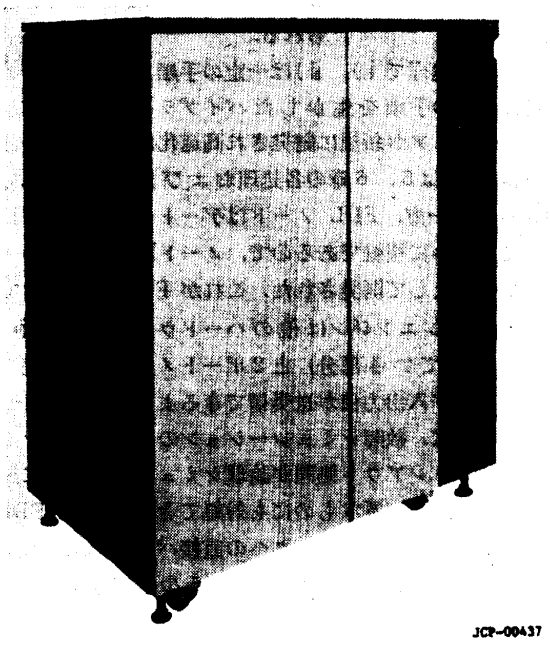


図 5 HAL Ⅲ 概観
Fig. 5 A photograph of HAL Ⅲ.

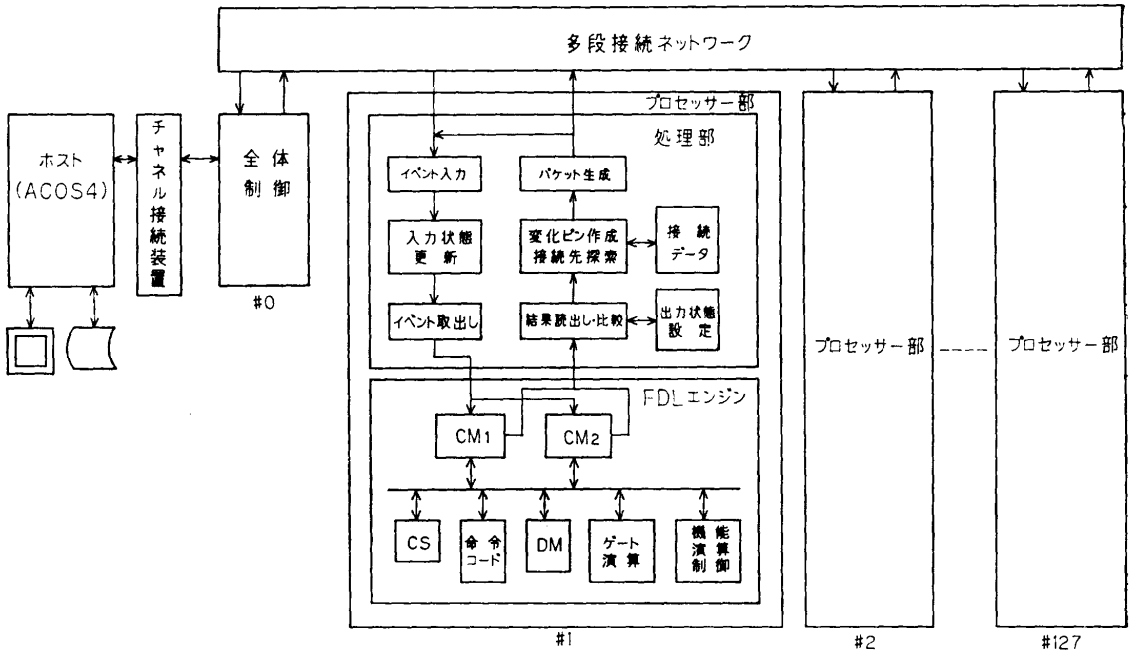


図 6 HAL Ⅲシステム構成
Fig. 6 HAL Ⅲ system configuration.

され，ノードの出力信号が決定される．

ⅲ) ファンアウト処理：変化した信号に接続している全ノードが探索され，それらにノードの評価された値が送られる．

これら処理でⅰ)，ⅲ)は一定の手順で行われるので，各々の手順を生かしたパイプライン構造の専用ハードウェアが新規に開発され高速化が計られている(この模様は5，6章の各処理および図9に示されている.)．一方，FDL ノードはゲートやブロックに比べてはるかに複雑であるので，ノード評価用専用ハードウェアとして開発された．これが FDL エンジンである．FDL エンジンは他のハードウェア（ⅰ)，ⅲ)を実現している部分）と2ポートメモリーで接続され，ノード入出力値が送受信できるようになっている．さらに，故障シミュレーションのようにイベント検出やファンアウト処理が論理シミュレーションに比べてはるかに複雑なものにも対処できるように，FDL エンジンからネットワークへの直接パスも準備されている．FDL エンジンについてもう少し詳述すると次のようである．

[FDL エンジンのハードウェア構成]

FDL のような機能記述言語を処理するには2つの方法がとられる．

- 専用のコンパイラか言語をCのような言語に落して機械語レベルのコードを生成する．
- スタックベースの中間コードを生成する．

HAL Ⅲでは次の理由から後者のアプローチがとられた．

- フルコンパイラの方法は効率的なコード生成ができるが，スタックベースのコード生成に比べて容量が増える．大型計算機をシミュレーションするにはこれが大きな問題である．
- 機械語のコード生成はスタックベースの方法に比べ時間がかかる．前にも述べたように上位レベルの論理検証において，短い TAT (Turn Around Time) は基本的条件である．

したがって，スタックベースの中間コードを処理する専用のマイクロコードエンジンが開発された．FDL はエンジンの特徴は次のようである．

ⅰ) 4値の論理オペレータを評価する専用のゲート演算ユニットが使われている．これは4 VLO と呼ばれる．レジスタレベルの記述でも，多くの FDL 記述は AND, OR, および NOT 等の単純オペレータが使われている．これは VLSI の中に多くの制御論理が入っているためである．4値の論理演算は一般のマ

イクロプロセッサでは単純な形で実現できないため，専用の4値演算ハードウェアが開発された．

ii) ハードウェア・スタックを使用している．

iii) 複雑な FDL オペレータの評価は汎用のマイクロプログラム ALU が使われ，全体の制御にはマイクロ・シーケンサが使われている．

iv) 中間コード言語のフェッチ用に専用ハードウェアを使用している．これはスタックベース方式のオーバヘッドの1つが中間言語のフェッチ/デコードであるからである．このオーバヘッドを少なくするためビデオ RAM を用いた中間命令キャシュおよびノード・オペレーションをデコード/ディスパッチする専用のハードウェアにより高速化を実現している．

HAL Ⅲシステムのプロセッサ，コントローラ，ネットワーク，チャネル，ホストコンピュータの各々の機能は次のとおりである．

1) プロセッサ部

図6に示されるようにプロセッサ部は処理部と FDL エンジンより構成されている．

a) 処理部（イベント制御部）

他または自プロセッサからのイベント・データを受けとり，入力状態更新をすると共に以前に立っているイベントを取り出してノードの種別や入力値を FDL エンジンへ送る．FDL エンジンからのノード演算(評価) 結果は旧結果と比較され，出力変化信号があるかどうかチェックされる．変化がある信号はその接続先が調べられ，演算値を入れてイベント・データを作っていく．

b) FDL エンジン

FDL エンジンは処理部よりノードの種別と入力値をもらって，イベント・ノードを評価する．ノードがゲートレベルで記述されていると，ゲート演算専用回路がノード評価に使われる．これはスタックや4値演算実行機構をもち，ゲート評価を高速に実行できる．ノードが機能レベルの時は機能ユニットが使われる．したがって，ノードがゲートレベル演算を多く含んでいても評価時間を増やすことなくシミュレーションできる．

処理部と FDL エンジンとの間は片方の CM に入力信号データが入って処理が行われている間にもう一方の CM に別のノードの入力信号が入るような2ウェイ構成になっていて処理効率を高めている．

2) 共通部

a) 全体制御部 (MSC : Master Control Processor)

MSC はパケット・トランシーバ，シミュレーション全体実行制御，ホストとの DMA (Direct Memory Access) インタフェースを持つ．MSC は最初に各レベル実行開始信号をブロードキャストし，各プロセッサにシミュレーションを実行させ，終了後に各プロセッサからレベル終了信号をうけて，これらの同期をとっている．さらにレベル付の大きいノードから小さいノードへの信号の後もどりがある場合の実行制御やシミュレーション中の論理発振の検出も行えるようになっている．

b) ネットワーク

ネットワークは多段ステージのインタコネクションネットワークである．最小構成は4入力ポート4出力ポートの専用 LSI より作られ，ブロードキャストやマルチパケット処理が可能になっている．ネットワーク全体としてはこれらの LSI が接続されて，パイプライン形で高速転送ができるようになっている．

3) ホスト・コンピュータ

ホストとしては ACOS 4 汎用計算機が使われ，HAL Ⅲ本体の実行制御をはじめ，後述の命令レベルシミュレータもこの上で実行され，HAL Ⅲと命令レベルシミュレータのデータ乗り移り，実行制御等も行われている．

4.2 ソフトウェア構成

ソフトウェア構成は**図7**に示される．

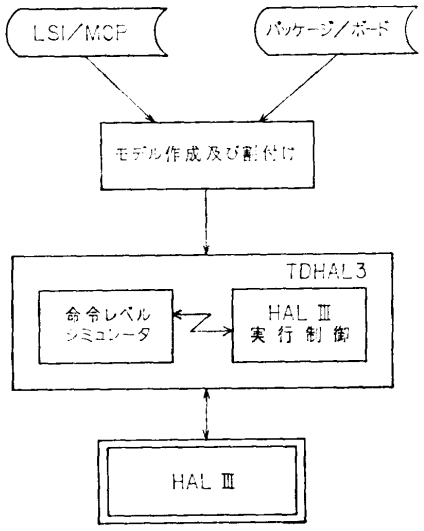


図7 ソフトウェア構成図
Fig. 7 Software configuration.

1) モデル作成部

ここでは装置設計の論理ファイル，すなわち，LSI, Multi-Chip-Package, パッケージ，ボード等のファイルを入力し，シミュレーションモデルを作成する．通常 LSI が FDL で記述されている．ここで，ノードのレベル付，ノードの各プロセッサへの割付け等が行われる．割付けには各種の工夫が施されている．例えば，LSI 単位の FDL ノードを同一プロセッサに割り付ける．こうすることでプロセッサ間通信量が減り，通信オーバヘッドが小さくなる．その他個々のノードの実行時間を見積り，ノードの実行時間が平均になるように割り付けるとか，最大モデル容量が入るように割り付けるとか，他シミュレータとの乗り移りデータがまとまるように割り付ける等の方法をとっている．

2) HAL Ⅲ実行制御部

ここでは HAL Ⅲへのモデルロード，メモリーロード，HAL Ⅲ起動，条件付停止，モデルの実行結果観測等 HAL Ⅲの実行制御を行っている．豊富なコマンド群を用意することによって，ユーザが HAL Ⅲを使いやすいものにしている．モデルに設定するパターンの生成や簡易的な論理修正機能も備えている．さらに後述の命令レベル・シミュレータと結合 (TDHAL 3) し，相互にデータを授受する機能も備えている．

5. HAL Ⅲのシミュレーション実行過程

図6に示されるプロセッサ部を詳細に示したものが図8である．図8の構成要素は以下のようである．

① Event：ノードのイベント生起データが格納される．
② Node：ノードの命令コードの種類や制御データが格納される．
③ Ipin：ノードの入力信号状態が格納される．
④ CM_1(CM_2)：ノードの種類や入力信号状態および評価結果が入る．
⑤ CS：ノードの評価ルーチン（命令）が格納される．
⑥ IM：ノード命令コードが格納される．
⑦ DM：メモリーノードやレジスタ状態，観測データ等を格納する．
⑧ Gate：ゲート記述を高速に演算する回路．

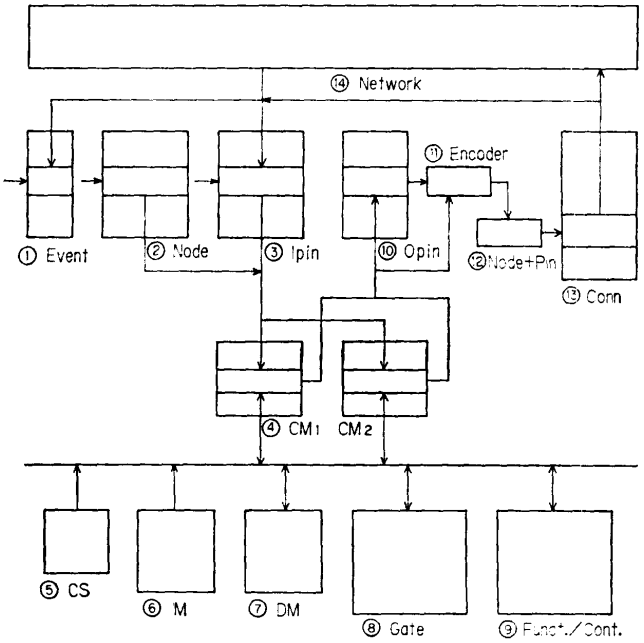


図 8 プロセッサ・ブロック・ダイアグラム
Fig. 8 Processor block diagram.

⑨ Funct/Cont：機能演算やノード評価の制御を行う回路．
⑩ Opin：出力状態が格納される．
⑪ Encoder：出力変化ピンを検出しコード化して順次出力する回路．
⑫ Node+Pin：イベントノードと変化ピンで接続先を調べるデータを出力する．
⑬ Conn：ノードの接続先が格納される．
⑭ Network：パケット転送経路．

図4で示したレベル付されたノードが HAL Ⅲでシミュレーションされる経過は次のようである．

[ノード・シミュレーション実行過程]

S-1：イベント・メモリー（① Event）をサーチし，イベントが立っているノードを探す（今ノード N_1 のイベントが立っているものと仮定する）．

S-2：イベントが立っているノードが見つかったなら，ノードの種類と入力値をノード種（② Node）と入力状態メモリー（③ Ipin）から通信メモリー（④ CM_1 または CM_2）へ送る（本例においては，ノード N_1 の種類と入力信号値（B(0:8) と C(0:8)）が CM_1 へ送られる．）．

S-3：通信メモリー CM_1 または CM_2 に送付さ

れた種類に基づき IM の命令コードを読み，このコードを解釈して，これを実行する ⑤ CS のマイクロ命令を読む．このマイクロ命令に基づく制御を ⑨ Cont で行いながら，⑧ Gate や ⑨ Funct を使ってノードの演算が行われ，その結果が CM_1 または CM_2 へ格納される（本例においては，ノード N_1 の入力 B(0:8) と C(0:8) を入力し，加算演算が実行され，その結果が CM_1 へ格納される．）．

S-4：通信メモリーへのノード演算結果の格納が終了すると，これらの値が⑪へ読み出され，出力状態メモリー（⑩ Opin）に入っている旧値と比較され，ノード出力信号の変化があったかどうか調べられる．その後，ノードの評価値が Opin へ格納される（本例においては，評価されたノード N_1 の A(0:8) が旧値と比較され出力変化があったかどうか調べられる．その後 A(0:8) の値が Opin へ格納される．）．

S-5：出力変化があると，出力変化信号ごとに接続先が調べられ，接続先のプロセッサ番号，ノード番号，入力信号（ピン）番号およびそこへの格納値（評価値）がつけ加えられ，送付形が作られる（⑪，⑫，⑬）．（本例において，A(4:4) の信号が変化したとすると，これらの信号が接続されているノード N_3 のプロセッサ番号，ノード番号，入力信号番号，および評価値の送付形が作られる．）

S-6：作成された送付形は送付先のプロセッサが他プロセッサか自プロセッサかチェックされ，他プロセッサに向かう場合はパケットにして⑭ネットワークを通して送られる．自プロセッサの場合はノード番号，入力信号番号，および評価値が ③ Ipin 側へ送られる（本例において，ノード N_1 に接続されているノード N_3 が自プロセッサにあるとすると，該当ノード番号，入力信号番号，および評価値が送られる）．

S-7：ネットワークからのパケットや自プロセッサからのデータはいったん順次 ③ Ipin 前にバッファリングされる．

S-8：前記バッファリングされたデータは順次読み出され，該当ノード番号の入力状態メモリー（③ Ipin）に設定され，イベント・メモリー（① Event）の該当ノードにイベントが立てられる（本例においては，ノード N_1 に接続されているノード N_3 の入力信号値が新しい値に置き換わり，イベント・メモリーのノード N_3 に対応する箇所にイベント・フラグが立つ．）．

これら各ステージはパイプライン的に実行される．さらに HAL Ⅲ は並列プロセッサであるから，複数台のプロセッサが同時に動作して，シミュレーションの高速化が実現されている．

6. システム性能

6.1 システム容量

HAL Ⅲ のシステム容量は**表 1** に示されている．

6.2 システム性能予測

HAL Ⅲ の性能は 5 章で述べた各処理をそれぞれ見積ることによりわかる．

D_1＝イベント・スキャン＆フェッチ＋FDL エンジンへノードデータ設定 (2)

D_2＝FDL ノード評価 (3)

D_3＝ノード出力結果の転送＋出力信号変化検出＋変化ピンエンコード (4)

D_4＝変化ピンの接続アクセス (5)

D_5＝パケット作成 (6)

D_6＝ネットワーク転送 (7)

D_7＝パケット入力 (8)

D_8＝イベント，入力状態の書き換え (9)

D_1〜D_8 は**図 9** に示すようにパイプライン的に実行できるので，任意のノード $i(N_i)$ の処理は以下のように見積ることができる．

$$N_i = \mathrm{Max}\{D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8\} \quad (10)$$

総ノードに対する全処理時間 S は次のようになる．

$$S = \frac{1}{N_p} \times \frac{1}{P_r} \times \sum_{i=1}^{T_n} N_i \times E_r \times \delta \quad (11)$$

ここで

N_p：総プロセッサ数

表 1 HAL Ⅲ システム容量
Table 1 HAL Ⅲ system capacity.

台 数	FDL ノード数	メモリー容量
1	16 k	2 MByte
127	2032 k	254 MByte

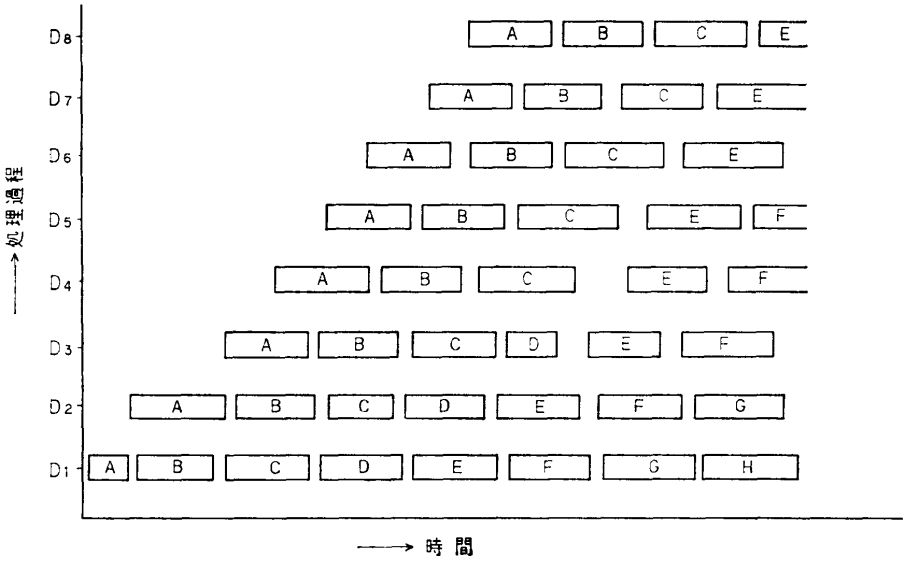


図 9 シミュレーションの処理過程
Fig. 9 Simulation processes.

P_r：並列性の損失
T_n：総ノード数
E_v：全体のイベント率
δ ：くり返し実行数（レベルもどり数）

例えば，同期型の計算機モデルを 31 台構成で実行する場合，従来の同種モデルで得られている値より求めると $N_p=31$, $P_r=1/2$, $T_n=30\,k$, $N_i=5\,\mu$s, $E_v=1/2$, $\delta=1$ となり，$S=5$ ms となる．これは ACOS 1000 (15 MIPS) 上で走るゲートレベル・ソフトウェア・シミュレータの約 10000 倍の性能となる．機能レベルのソフトウェア・シミュレータは FDL の記述の仕方により変動するので一概には言えないが，ゲートレベルに比べて数倍以上は良いことが別種モデルでわかっているので，HAL Ⅲは機能レベル・ソフトウェア・シミュレータより数千倍高速の性能と見積れる．

6.3 性能解析

1) プロセッサ台数増加に伴う性能変遷

異なる実モデルを使用して，プロセッサ台数増加に伴う性能の変化を実際に調べ，それを相対的に示したものが**表 2** である．

これをグラフとして示したものが**図 10** である．

表 2 や図 10 からわかるように台数増加に伴う性能向上がきれいに出ている．例えば，モデルAの場合，4 台構成の実行時間に対して，8 台構成では 64%，16 台構成では 41%，31 台構成では 34% で処理を終

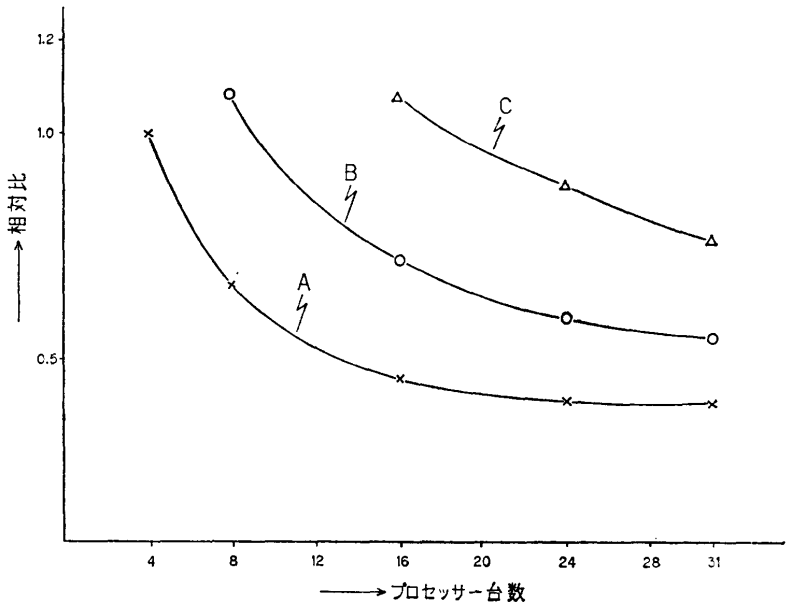


図 10 プロセッサ台数増加に伴う性能変遷
Fig. 10 System performance vs processor increments.

表 2 プロセッサ台数増加に伴う性能変遷
Table 2 System performance vs processor increments.

モデル	LSI 数	ノード数	プロセッサ台数				
			4	8	16	24	31
A	40個	15843	1*	0.64	0.41	0.35	0.34
B	114個	33286	—	1.08	0.69	0.55	0.5
C	178個	64836	—	—	1.08	0.86	0.72

*：モデルAの4台構成 HAL Ⅲ全体の実行処理時間を1として，プロセッサ台数増加や他モデルを相対的に示したもの．

表 3 割付け方法の変更による性能変遷
Table 3 System performance with various circuit model allocations.

割付け方法	プロセッサ台数			
	8	16	24	31
a) LSI まとめる メモリーまとめる	1.08	0.69	0.55	0.5
b) LSI まとめる メモリー分割	1.08	0.68	0.53	0.45
c) LSI 分割 メモリー分割	1.28	0.93	0.62	0.5
d) LSI, メモリー分割 容量優先	1.36	0.96	0.62	0.5

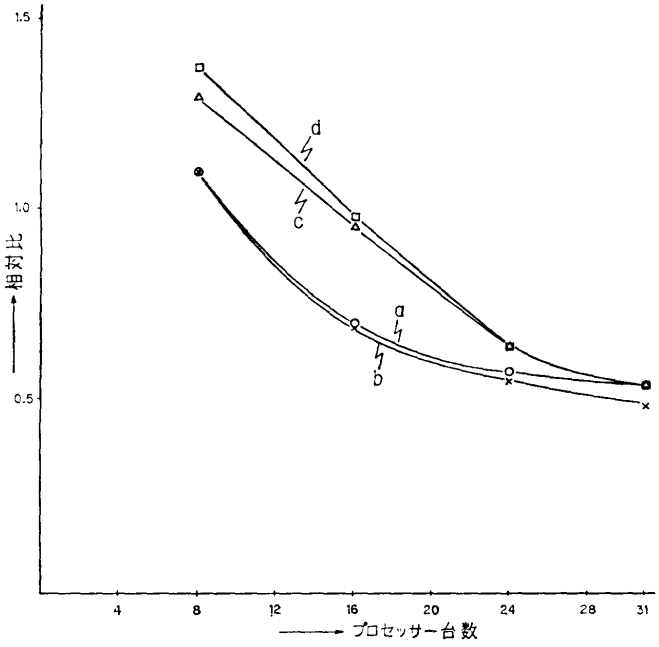


図 11 割付け方法の変更による性能変遷
Fig. 11 System performance with various circuit model allocations.

了することが示されている．モデルBやCの値はモデルAの4台構成時の実行時間を1として相対的に示しているが同様な傾向を示している．ただプロセッサ台数が多くなれば，性能向上率が小さくなってきている．これはプロセッサ台数増加によるオーバヘッドのためと考えられる．

2) 割付け方法変更による性能変遷

HAL Ⅲ内のプロセッサに割り付けるデータをいろいろ変更して，性能に及ぼす影響を調べるものが**表3**であり，グラフとして示したものが**図 11** である．

表3の数値は表2のモデルBをもとに割付け法を変えた時の値であり，相対比の基準も表2と同じである．

表3や図11からわかるように，16台以下の割付けにおいては，LSI のノードをまとめる効果は非常に大きく 20～40% 向上している．これはイベント伝播がプロセッサ内で閉じているため，プロセッサ間通信量が減り，このオーバヘッドが少なくなったためと考えられる．ノードができるだけ多くつめこむような割付け法は一番時間がかかっている．表3での“メモリーまとめる/分割”はモデル上のメモリーを一箇所にまとめておくか，分散しておくかということを示す．

7. HAL Ⅲの応用

7.1 HAL Ⅲと命令レベル・シミュレータとのリンク（TDHAL 3）

装置診断プログラムをより有効に実行するため，HAL Ⅲと命令レベル・シミュレータ（TDSIM：T & D Simulator）とリンクしたシステムが開発された．装置診断プログラムは大きく分けて，前処理部（試験環境の設定），機能試験部，後処理部（結果の編集処理）と分かれているが，このうち大部分が前後処理部で，機能試験部はほんの数%にすぎないため，モデルの詳細機能まで立入ったシミュレーションをしなくてもよい前後処理部を高速な命令レベル・シミュレータで，本当に調べたい機能試験部を HAL Ⅲで実行すれば，より効果的に実行できる．この模様は**図 12** に示されている．

命令レベル・シミュレータは実マシンの1命令を約300命令でシミュレーションす

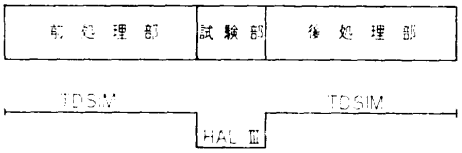


図 12 TDHAL 3 実行過程
Fig. 12 TDHAL 3 execution processes.

るため実マシンに比べて100倍から1000倍程度遅いだけの速度をもっている．したがって，HAL Ⅲと命令レベル・シミュレータとをリンクしたTDHAL 3はシミュレータ間のデータ乗り移りはあるが，HAL Ⅲ単体で行うよりも数十倍以上高速に実行される．例えば，CPU 命令試験の構成は前後処理部が全体の99%以上を占め，実試験部はわずか1%程度である．この試験は前後処理部を含めた全体のステップ数が非常に長く，HAL Ⅲ単独で行うと1800秒かかるがTDHAL 3で行うとシミュレータ間の乗り移りも含めて38秒ほどで終了し約50倍性能が改善されている．

7.2 故障シミュレーション

1) 故障シミュレーション方式，アルゴリズム

HAL Ⅲは故障シミュレータとして使用することもできる．この時はノードがゲートレベルとなりFDLエンジンで故障シミュレーションが実行される．回路モデルはゲート展開されたもので，モデル作成時に入力端子またはレジスタから出力端子またはレジスタへ向かってレベル付された後各プロセッサへ分割されたモデルが作られる．HAL Ⅲで実行される過程は以下のようである．

[故障シミュレーション実行過程]

ⅰ) 故障シミュレーションモデルをHAL Ⅲの各プロセッサへロードする．

ⅱ) 次に，ホストより数百〜数千の入力パターン集合をプロセッサに設定して故障シミュレーションの実行をHAL Ⅲに指示する．

ⅲ) これらの数百〜数千のパターンの実行は次のとおりである．

① MSC（全体制御）が各プロセッサへ1パターンの実行を指示する．

② 入力パターンがパターンを設定されたプロセッサから発生され伝播先のノード（ゲート）に転送される．

③ モデルの各ノードがレベルごとに故障シミュレーションされ，最終レベルまで到達する．

④ 最終レベルの実行終了後，出力の期待値と検出故障を登録し，検出故障を削除する．

⑤ 設定された全パターンが終了すると，終了通知をホストに出す．

ⅳ) ホストは終了通知を受けとると，前記入力パターン集合に対する実行結果をとり出し，次の入力パターン集合を設定して同様の処理を行う．

ⅴ) すべての入力パターンを終了したところで処理を終了する．

アルゴリズムの特徴としては，各レベルで複数プロセッサにより，コンカレントシミュレーションが同時に実行され高速化されている．シミュレーション値は4値であり，遅延はゼロである．

2) 性能予測

モデルやテスト・パターンが与えられた時，それらを処理する故障シミュレーション時間を求めることは難しい問題であるが，本論文で提案している方式での故障シミュレーション時間（オーダ）は，各プロセッサに割り付けられているレベルごとの処理時間をもとに次のように見積ることができる．プロセッサのレベル i の処理時間 R_i は故障シミュレーションの各々の処理を考慮して次のように求めることができる．

$$
\begin{aligned}
R_i = &\text{イベント・ノード・データ設定時間} \\
&+\text{正マシンおよびリスト故障処理時間} \\
&+\text{伝播故障処理時間}+\text{定義故障処理時間} \\
&+\text{故障リスト編集処理時間} \\
&+\text{入力パケット処理時間} \\
&+\text{出力パケット処理時間} \\
=&(N_S+L_S\times L_L)\times N_C \\
&+(G_S+FL_S\times L_L)\times G_E \\
&+F_S\times F_E+F_{DS}\times F_{DE} \\
&+L_D\times L_H \\
&+I_P\times P_N \\
&+O_P\times P_N \qquad (12)
\end{aligned}
$$

ここで，

N_S ：ノードテーブルアクセス時間，
L_S ：1故障リストの平均処理時間，
L_L ：1ノード当りの故障リスト長，
N_C ：各レベル当りのノード数，
G_S ：正マシンの平均処理時間，
FL_S：リスト故障の平均処理時間，
G_E ：レベル当りの正イベント数，
F_S ：伝播故障の平均処理時間，
F_E ：レベル当りの故障イベント数，

F_{DS}：定義故障の平均処理時間,
F_{DE}：レベル当りの定義故障数,
L_D：リスト編集処理時間,
L_H：レベル当りの故障リスト変化数,
I_P：入力パケット処理時間,
O_P：出力パケット処理時間,
P_N：レベル当りのパケット数

パターン当りの処理時間 P_T は次のようになる.

$$P_T=\frac{R_i\times R_N+F_{DL}\times F_{DLN}}{AV_1} \tag{13}$$

ここで,
R_N：モデルの最大レベル数,
F_{DL}：故障削除処理時間,
F_{DLN}：パターン当りの削除故障数,
AV_1：プロセッサのパターン当りの稼働率.

したがって，パターン N では，全体の稼働率を AV_2 として次のようになる.

$$T_N=\frac{P_T\times N}{AV_2} \tag{14}$$

例えば，十万ゲート，十万パターン，一万故障のモデルをプロセッサ 30 台に割り付けて，正，故障イベント率を 15%，25%，プロセッサ稼働率を 0.5，全体稼働率を 0.75 等より見積ると数時間のオーダとなり，ソフトウェア・シミュレータの約 100 倍の性能となる.

7.3 網羅率測定

HAL Ⅲはさらに次に示すようなハードウェアやファームウェアの網羅率を測定でき，装置診断プログラム等の検査品質を測ることもできる.

1) ハードウェア網羅率

$$\mathrm{HWC}_1=\frac{\sum_i L_{01i}}{L} \tag{15}$$

$$\mathrm{HWC}_2=\frac{\sum_i L_{10i}}{L} \tag{16}$$

$$\mathrm{HWC}_3=\frac{\sum_i L_{Bi}}{L} \tag{17}$$

$$\mathrm{HWC}_4=\frac{\sum_i L_{vi}}{L_v} \tag{18}$$

ここで
L_{01i}：論理 0→1 変化した信号線 i は 1,
L：全信号線,
L_{10i}：論理 1→0 変化した信号線 i は 1,
L_{Bi}：論理 0→1, 1→0 両方変化した信号線では 1,
L_{vi}：信号線 i の論理 (0, 1) が観測可能地点（出力端子またはレジスタ）まで伝播する時 1 となり，論理 0，1 の両方が伝播する時は 2,
L_v：全信号線の論理の総和.

2) ファームウェア網羅率

$$\mathrm{FWC}_1=\frac{\sum_i S_i}{S} \tag{19}$$

$$\mathrm{FWC}_2=\frac{\sum_i P_i}{P} \tag{20}$$

ここで
S_i：実行された FW ステップ i は 1，その他 0,
S：全 FW ステップ数,
P_i：パス i が実行されると 1,
P：取り得るすべてのパス数.

8. むすび

HAL Ⅲのシミュレーション方式，ハードウェア，ソフトウェア構成，性能予測・評価および応用領域について述べた．HAL Ⅲは高級言語 FDL をそのまま扱えて，設計の上流工程で論理検証ができる．さらに HAL Ⅲは，故障シミュレーション，網羅率測定等多様な使い方ができる．これは HAL Ⅲが高速性を維持しながら，柔軟な使い方ができるアーキテクチャとなっているためである.

HAL Ⅲは非常に有力な論理検証のツールとして広く使われているが，マルチ・プロセッサ・システムであるためまだまだ全体のオーバヘッドが大きい．これらの問題点を探し，全体の最適化/効率化を良くすること，さらに他シミュレータとのリンク時は，相互データ乗り移り性能を向上させることが必要である．検査関係ではテストデータ生成も本システム内で実現することが今後の課題である.

謝辞 最後に日頃御指導頂く第三 OA 装置事業部古勝事業部長，コンピュータ技術本部桑田本部長，高橋部長，日本電気フィールド・サービス(株)第一技術本部，杉本本部長代理，C&C システム技術開発推進本部舩津部長に深謝します.

またコンピュータ技術本部の成友嬢，橋本氏，岡田氏，甲府日本電気(株)岡本課長，清水氏，若月氏，北陸日本電気ソフトウェア(株)大窪氏，中田氏に感謝します.

参考文献

1) Koike, N., Ohmori, K., Kondo, H. and Sasaki, T.: A High Speed Logic Simulation Machine, *Compcon 83 Spring*, pp. 446-451 (1983).
2) Sasaki, T., Koike, N., Ohmori, K. and Tomita, K.: HAL; Block Level Hardware Logic Simulation, *Proc. 20th DA Conf.*, pp. 150-156 (June 1983).
3) Pfister, G. F.: The Yorktown Simulation Engine: Introduction, *Proc. 19th DA Conf.*, pp. 51-54 (June 1982).
4) Howard, J. K., Malm, R. R. and Warren, L. M.: Introduction to the IBM Los Gatos Logic Simulation Machine, *IEEE Conf. on Computer Design*, pp. 580-585 (1983).
5) Blank, T.: A Survey of Hardware Accelerators Used in Computer Aided Design, *IEEE Design & Test*, Vol. 1, No. 4, pp. 21-39 (Aug. 1984).
6) Takasaki, S., Sasaki, T., Nomizu, N., Koike, N. and Ohmori, K.: Block-Level Hardware Logic Simulation Machine, *IEEE Trans. CAD*, Vol. CAD-6, No. 1, pp. 46-54 (Jan. 1987).
7) Takasaki, S., Sasaki, T., Nomizu, N., Ishikura, H. and Koike, N.: HAL II: A Mixed Level Hardware Logic Simulation System, *Proc. 23-rd DA Conf.*, pp. 581-587 (June 1986).
8) Kato, S. and Sasaki, T.: FDL: A Structural Behavior Description Language, *6th Int. Symp. CHDL*, pp. 137-152 (May 1983).
9) Hirose, F., Ishii, M., Niitsuma, J., Shindo, T., Kawato, N., Hamamura, H., Uchida, K. and Yamada, H.: Simulation Processor SP, *ICCAD-87*, pp. 484-487 (Nov. 1987).
10) Yoshimura, T. and Goto, S.: A Rule-Base and Algorithmic Approach for Logic Synthesis, *ICCAD-86*, pp. 162-165 (Nov. 1986).
11) 髙﨑, 野水, 平林, 石倉, 蔵下, 小池, 中田: HAL III: 機能レベル・ハードウェア・シミュレーション・システム, 設計自動化研究会報告, Vol. 90, No. 42, pp. 1-10 (May 1990).
12) Takasaki, S., Nomizu, N., Hirabayashi, Y., Ishikura, H., Kurashita, M., Koike, N. and Nakata, T.: HAL III: Function Level Hardware Logic Simulation System, *ICCD '90*, pp. 167-170 (Sept. 1990).

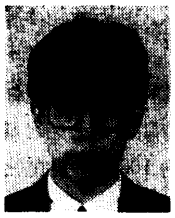
(平成元年 8 月 31 日受付)
(平成 2 年 10 月 9 日採録)

髙﨑　茂（正会員）

1975 年工学院大学工学部電子工学科卒業. 77 年同大学院修士課程修了. 同年日本電気(株)入社. 以来論理回路の試験容易化設計, テスト生成, シミュレーション, CAD 専用エンジン, CAD システムの研究開発に従事. 工学博士. 現在, コンピュータ技術本部 CAD 技術部技術課長. 電子情報通信学会, 米国 IEEE 各会員.

野水　宣良（正会員）

1950 年生. 1972 年新潟大学工学部電子工学科卒業. 同年日本電気(株)入社. 現在, 同社コンピュータ技術本部においてコンピュータ設計用 CAD システムに関する研究開発に従事.

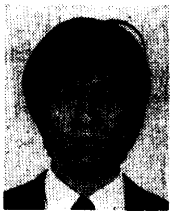


平林　良啓（正会員）

昭和 31 年生. 昭和 55 年電気通信大学電子通信学部電子計算機学科卒業. 同年日本電気(株)入社. 現在, コンピュータ技術本部 CAD 技術部主任. 論理検証ツールの開発に従事.

石倉　浩

昭和 33 年生. 昭和 57 年鹿児島大学工学部電子工学科卒業. 同年日本電気(株)に入社. 主に論理検証用のハードウェアエンジンの開発に従事. 昭和 63 年甲府日本電気(株)に勤務, 現在にいたる.



蔵下　正広（正会員）

昭和 35 年生. 昭和 58 年関西大学工学部電子工学科卒業. 同年日本電気(株)入社. 以来, コンピュータ技術本部にて, 論理検証手段開発に従事.

小池 誠彦（正会員）

昭和22年生．昭和45年東京大学工学部電気工学科卒業．昭和47年同大学院修士課程修了．同年日本電気(株)に入社．以来並列計算機システム，論理シミュレーションエンジン(HAL)，並列回路シミュレーションマシン(Cenju)などの研究開発に従事．最近では，並列計算機のアーキテクチャ，CADマシン，AIシステム，ニューラルネットワークなどの研究に興味を持つ．現在，同社C&Cシステム研究所コンピュータシステム研究部長．著書「CADマシン」(電子情報通信学会編，オーム社)．電子情報通信学会会員．昭和59年度情報処理学会論文賞受賞．平成2年本学会創立30周年記念論文入選．

中田登志之（正会員）

昭和32年生．昭和57年京都大学大学院工学研究科修士課程修了．昭和60年同大学院博士後期課程単位取得退学．同年日本電気(株)入社．京都大学工学博士．現在同社C&Cシステム研究所コンピュータシステム研究部に勤務．並列計算機システムの研究に従事．昭和61年度本学会論文賞受賞．平成2年本学会創立30周年記念論文入選．電子情報通信学会会員．