設 計 自 動 化 52－1
（1990．5．22）

HALⅢ：機能レベル・ハードウェア・シミュレーション・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***

*　日本電気㈱コンピュータ技術本部ＣＡＤ技術部
**　甲府日本電気㈱
***　日本電気㈱Ｃ＆Ｃ研究所

あらまし：

本稿では機能レベルのハードウェアシミュレータＨＡＬⅢについて述べている。ＨＡＬⅢはレジスタ・トランスファ・レベルの設計言語ＦＤＬで書かれたモデルをゲートレベルに展開することなくシミュレーションすることができる。ＨＡＬⅢではパラレル，パイプライン，及び柔軟性のあるＦＤＬ演算機構のアーキテクチャが取り入れられ，さらに，レジスタ・トランスファ・レベルのイベント・ドリブン，レベル・ソートアルゴリズムが使われている。ＨＡＬⅢは３１台構成で，従来のゲートレベルのソフトウェアシミュレータの１万倍以上の性能をもち，最大１２７台構成まで拡張できる。さらにＨＡＬⅢは命令レベル・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，高速なコンカレント法を採用した故障シミュレーション機能，ハードウェアやファームウェアの網羅率を定量的に見積る機能等を備えていて，広範囲に適用可能なハードウェアシミュレータである。

HALⅢ：Function Level Hardware Simulation System

Shigeru Takasaki*, Nabuyoshi Nomizu*, Yoshihiro Hirabayashi*, Hiroshi Ishikura**,
Masahiro Kurashita*, Nabuhiko Koike*** and Toshiyuki Nakata***

*　NEC Corp. Computer Engineering Division, CAD Dept.
**　Kofu NECCorp.
***　NEC Corp.C&C Laboratories.
1-10 Nisshin-cho,Fuchu-city,Tokyo 183 Japan.

Abstract

This paper describes a function level hardware simulator:HALⅢ. HALⅢ can simulate a circuit model written by a register transfer level language FDL without translating it into gate level. HALⅢ adopts parallel, pipeline and flexible FDL evaluation architectures, and employs level sort and event driven algorithms at register transfer level. HALⅢ is more than ten thousand times faster than conventional gate level software simulators in the case of 31 processors used. HALⅢ can be expanded to 127 processors. In addition, when integrated with an instruction-level simutator, HALⅢ can execute test programs quite effectively without sacrificing verification accuracy. Other applications are a fault simulation and hardware/firmware logic change coverage measurements. HALⅢ is effectively used in VLSI designs.

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程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来ている。すなわち，従来の様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，充分論理検証した後，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，充分な検証をしておく必要がある。これを達成するには，従来の様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）

この様な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（ＨＡＬⅢ）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ＨＡＬⅢ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，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もれる網羅率測定機能を備えている。本稿では，2.でＶＬＳＩ時代の設計法とＨＡＬⅢの関係，3.でＨＡＬⅢのシミュレーション方式，4.でＨＡＬⅢのシステム構成，5.でＨＡＬⅢのシュミュレーション実行過程，6.でＨＡＬⅢのシステム性能，7.でＨＡＬⅢの応用について述べている。

2. ＶＬＳＩ時代の設計法とＨＡＬⅢ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及びＨＡＬⅢの開発背景について述べる。

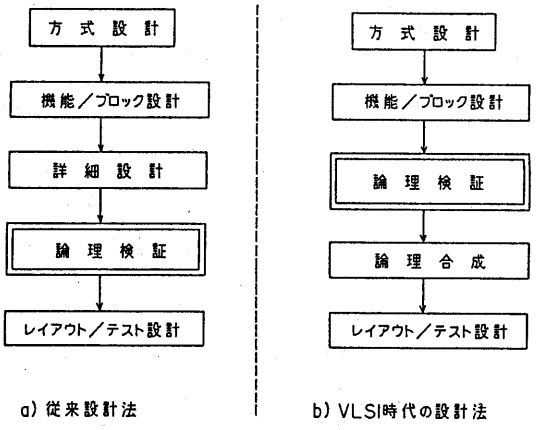

図１　設計法の変遷

従来の設計法は図１（ａ）に示す様に方式設計，機能／ブロック設計，詳細設計と全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。従って，この様な設計法においては，設計の下流工程での論理検証，即ち，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来ている。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設計を従来通りの方法で行うと，論理検証で多数の論理バグが検出されるため，詳細回路の修正及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くる。従って，これら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）

これらの最近の進捗はめざましく，従来の設計法を変えて来ている。即ち，機能設計を高級記述言語で行い，充分論理検証して，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。従って，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。ＨＡＬⅢ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。

3. ＨＡＬⅢのシミュレーション方式

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

ＨＡＬⅢ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㈱で開発されたものである。7)ノードとは図２に示される様な機能単位である。

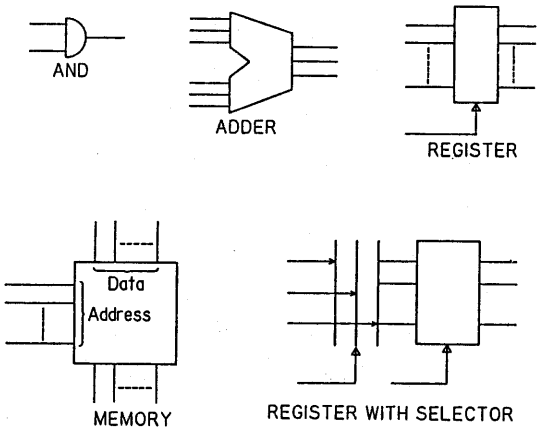

図２　ノード例

ＦＤＬは７種類の文より成立つ。

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）からみると次の様なものがある。

A)　論理演算子 … 例えば，AND,OR,EXOR,･･･
B)　機能演算子 … 例えば,SHIFT LEFT,SHIF, TRIGHT,･･･
C)　算術演算子 … 例えば,ADD,SUBTRACT,･･･
D)　比較演算子 … 例えば,EQUAL,GREAT-ER THAN,･･･
E)　ﾀｲﾐﾝｸﾞ演算子 … 例えば,GO UP,GO DOWN,･･･

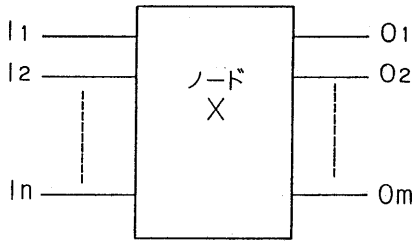

図３　ノード表現

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。

O1～ｍ＝Ｉ1.OP1.Ｉ2.OP2.･･･. OPq.Ｉn …(1)

ここでO1～ｍ　：出力信号
Ｉ1～n　：入力信号
ＯＰ1～q：ＦＤＬオペレータ

ノードとはＦＤＬ文全体をさす。

以上から判る様に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ﾋﾞｯﾄとCの8ﾋﾞｯﾄを加算する
P(0:4)=F(0:8).SUB.G(0:8);Fの8ﾋﾞｯﾄからGの8ﾋﾞｯﾄを減算する
D(0:4)=A(4:4).AND.P(0:4);Aの4ﾋﾞｯﾄから4ﾋﾞｯﾄとPの0ﾋﾞｯﾄから4ﾋﾞｯﾄを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（又はレジスタ）から出力端子（又

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に示す様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，ＨＡＬⅢへ送られる。レベル付されたノードはＨＡＬⅢ上で入力端子からレジスタ（又は出力端子）に向ってレベル毎に並列に実行される。図４において，ｉレベルのノードＮ１とＮ２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．で述べられている。）

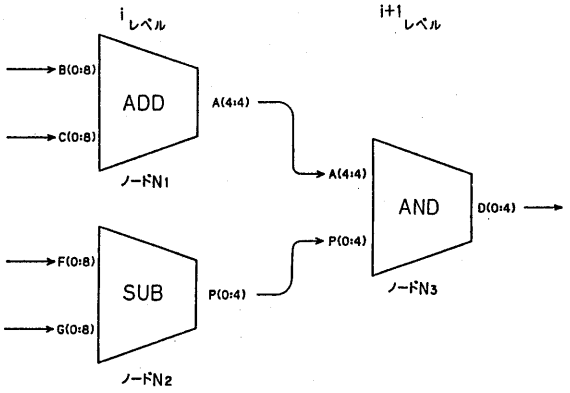

図４　レベル付されたノード

3.3 シミュレーション・アルゴリズム

ＨＡＬⅢの論理シミュレーション・アルゴリズム，シミュレーション値，及び遅延は以下の通りである。

- ｼﾐｭﾚｰｼｮﾝ･ｱﾙｺﾞﾘｽﾞﾑ　：レベルソート法，イベント・ドリブン法
- シミレーション値：４値（０，１，Ｘ，Ｚ）
- 遅　延　　：零

ＨＡＬⅢは論理機能検証を高速に実現するため，並列化効果が最大限に発揮できる上記の様な方法をとっている。

4. ＨＡＬⅢのシステム構成

ＨＡＬⅢ全体のシステム構成は図５の様である。

4.1 ハードウェア構成

1)　プロセッサー部

図５に示される様にプロセッサー部は処理部と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）

他又は自プロセッサーからのイベント・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。

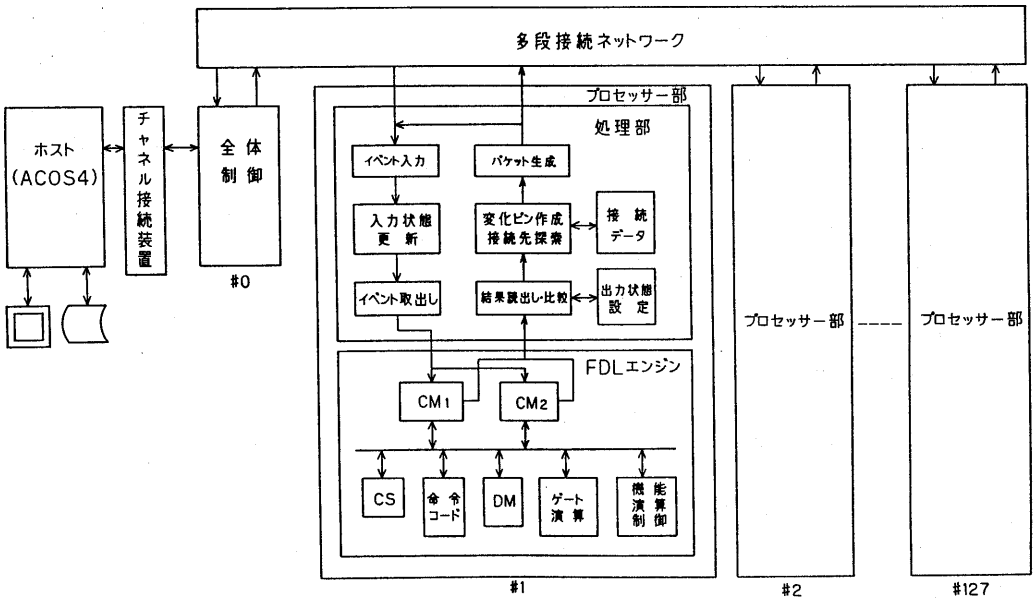

図５　ＨＡＬⅢ　システム構成図

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。従って，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様な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，各プロセッサーにシミュレーションを実行させ，終了後に各プロセッサー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ー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出来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ACOS4汎用計算機が使われ，HALⅢ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，HALⅢ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6に示される。

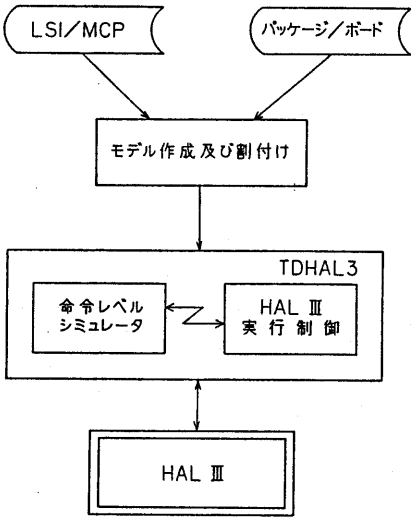

図6 ソフトウェア構成図

1) モデル作成部

ここでは装置設計の論理ファイル，即ち，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ー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ーに割り付ける。こうすることでプロセッサー間通信量が減り，通信オーバー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もり，ノードの実行時間が平均になるように割付けるとか，最大モデル容量が入る様に割付けるとか，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) HALⅢ実行制御部

ここではHALⅢへのモデルロード，メモリーロード，HALⅢ起動，条件付停止，モデルの実行結果観測等HALⅢ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，ユーザーがHALⅢ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（ＴＤＨＡＬ３）し，相互にデータを授受する機能も備えている。

5. ＨＡＬⅢのシミュレーション実行過程

図５に示されるプロセッサー部を詳細に示したものが図７である。図７構成要素は以下の様である。

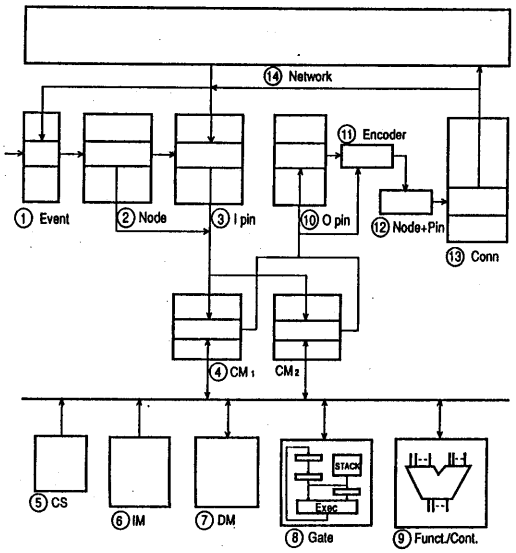

図７　プロセッサ・ブロック・ダイアグラム

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 CM1(CM2)：ノードの種類や入力信号状態及び評価結果が入る。

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。

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ＨＡＬⅢでシミュレーションされる経過は次の様である。

［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Ｎ1 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（②Node）と入力状態メモリー（③Ipin）から通信メモリー（④CM1　又はCM2 ）へ送る。
（本例においては，ノードＮ1 の種類と入力信号値(B(0:8) とC(0:8))がＣＭ1 へ送られる。）

S-3:通信メモリーＣＭ1 又はＣＭ2 に送付された種類に基き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⑤ＣＳ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⑨Contで行いながら，⑧Gateや⑨Funct を使ってノードの演算が行われ，その結果がＣＭ1 又はＣＭ2 へ格納される。
（本例においては，ノードＮ1 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ＣＭ1 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（⑩Opin）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Ｎ1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，出力変化信号毎に接続先が調べられ，接続先のプロセッサー番号，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N3 のプロセッサー番号，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ーが他プロセッサーか自プロセッサーかチェックされ，他プロセッサー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。自プロセッサーの場合はノード番号，入力信号番号，および評価値が③Ｉｐｉｎ側へ送られる。

（本例において，ノードN1 に接続されているノードN3 が自プロセッサーにあるとすると，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ーからのデータは一旦順次③Ｉｐｉｎ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（③Ipin）に設定され，イベント・メモリー（①Event ）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N1 に接続されているノードN3 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N3 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ＨＡＬⅢ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

ＨＡＬⅢのシステム容量は表１に示されている。

表１　ＨＡＬⅢシステム容量

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

6.2 システム性能予測

ＨＡＬⅢの性能は５章で述べた各処理をそれぞれ見積もることにより判る。

D1 ＝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)

D2 ＝ＦＤＬノード評価 ……(3)

D3 ＝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)

D4 ＝変化ピンの接続アクセス ……(5)

D5 ＝パケット作成 ……(6)

D6 ＝ネットワーク転送 ……(7)

D7 ＝パケット入力 ……(8)

D8 ＝イベント，入力状態の書き換え ……(9)

D1 ～D8 はパイプライン的に実行できるので，任意のノードｉ（Ｎｉ）の処理　は以下の様に見積もることができる。

$$Ni = Max\{D1, D2, D3, D4, D5, D6, D7, D8\} \quad \cdots\cdots(10)$$

総ノードに対する全処理時間Ｓは次の様になる。

$$S = \frac{1}{Np} \times \frac{1}{Pr} \times \sum_{i=1}^{Tn} N_i \times E_{\upsilon} \times \delta \quad \cdots\cdots(11)$$

ここで

Ｎｐ ：総プロセッサー数

Ｐｒ ：並列性の損失

Ｔｎ ：総ノード数

E_{υ}：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，ＮＰ ＝３１，Ｐｒ ＝1/2 ，Ｔn＝３０ｋ，Ｎｉ＝５μｓ，E_{υ}＝1/2 ，δ＝１とするとＳ＝５ｍ

sとなり，これはＡＣＯＳ1000(15MIPS)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。

7. ＨＡＬⅢの応用

7.1 ＨＡＬⅢと命令レベルシミュレータとのリンク（ＴＤＨＡＬ３）

装置診断プログラムをより有効に実行するため，ＨＡＬⅢ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，機能試験部，後処理部と分かれているが，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，前後処理部を命令レベルシミュレータで，機能試験部をＨＡＬⅢで実行すれば，より効果的に実行できる。この模様は図８に示されている。

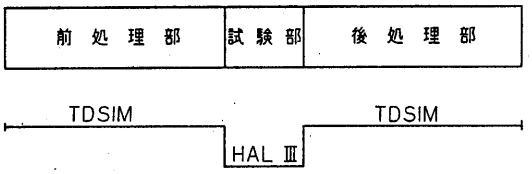

図８　ＴＤＨＡＬ３　実行過程

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。従って，ＨＡＬⅢと命令レベルシミュレータとをリンクしたＴＤＨＡＬ３はシミュレータ間のデータ乗り移りはあるが，ＨＡＬⅢ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，ＨＡＬⅢ単独で行うと１８００秒かかるがＴＤＨＡＬ３で行うとシミュレータ間の乗り移りも含めて３８秒程で終了し約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

ＨＡＬⅢ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又はレジスタから出力端子又はレジスタへ向かってレベル付された後各プロセッサーへ分割されたモデルが作られる。ＨＡＬⅢで実行される過程は以下の様である。

［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ＨＡＬⅢの各プロセッサーへロードする。

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ーに設定して故障シミュレーションの実行をＨＡＬⅢ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通りである。

① ＭＳＣ（全体制御）が各プロセッサーへ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ー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毎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) 全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ーにより，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

故障シミュレーションの性能を予測することは極めて難しいが，各プロセッサーに割り付けられているレベ

ル毎の処理時間をもとに次のように見積もることが出来る。

プロセッサーのレベルｉの処理時間Ｒｉは故障シミュレーションの各々の処理を考慮して次の様に求めることができる。

Ｒｉ＝イベント・ノード・データ設定時間
＋正マシン及びリスト故障処理時間
＋伝播故障処理時間＋定義故障処理時間
＋故障リスト編集処理時間
＋入力パケット処理時間
＋出力パケット処理時間
＝（ＮＳ＋ＬＳ×ＬＬ）×Ｎ
＋（ＧＳ＋ＦＬＳ×ＬＬ）×Ｇ
＋ＦＳ×ＦＥ＋ＦＤＳ×ＦＤＥ
＋ＬＤ×ＬＨ
＋Ｉｐ×ＰＮ
＋Ｏｐ×ＰＮ ……(12)

ここで，

ＮＳ：ノードテーブルアクセス時間，

ＬＳ：１故障リストの平均処理時間，

ＬＬ：１ノード当りの故障リスト長，

ＮＣ：各レベル当りのノード数，

ＧＳ：正マシンの平均処理時間，

ＦＬＳ：リスト故障の平均処理時間，

ＧＥ：レベル当りの正イベント数，

ＦＳ：伝播故障の平均処理時間，

ＦＥ：レベル当りの故障イベント数，

ＦＤＳ：定義故障の平均処理時間，

ＦＤＥ：レベル当りの定義故障数，

ＬＤ：リスト編集処理時間，

ＬＨ：レベル当りの故障リスト変化数，

ＩＰ：入力パケット処理時間，

ＯＰ：出力パケット処理時間，

ＰＮ：レベル当りのパケット数

パターン当りの処理時間ＰＴは次の様になる。

$$PT = \frac{Ri \times RN + FDL \times FDLN}{AV1} \quad \cdots\cdots(13)$$

ここで，

ＲＮ：モデルの最大レベル数，

ＦＤＬ：故障削除処理時間，

ＦＤＬＮ：パターン当りの削除故障数，

ＡＶ１：プロセッサーのパターン当りの稼働率。

従って，パターンＮでは，全体の稼働率をＡＶ２として次のようになる。

$$TN = \frac{PT \times N}{AV2} \quad \cdots\cdots(14)$$

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ー３０台に割付けて，正，故障イベント率を１５％，２５％，プロセッサー稼働率を０．５，全体稼働率を０．７５等より見積もると数時間のオーダーとなり，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

ＨＡＬⅢ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出来る。

1) ハードウェア網羅率

$$HWC1 = \frac{\sum_i L01i}{L} \quad \cdots\cdots(15)$$

$$HWC2 = \frac{\sum_i L10i}{L} \quad \cdots\cdots(16)$$

$$HWC3 = \frac{\sum_i LBi}{L} \quad \cdots\cdots(17)$$

$$HWC4 = \frac{\sum_i L\upsilon i}{Lv} \quad \cdots\cdots(18)$$

ここで

Ｌ０１ｉ：論理０→１変化した信号線ｉは１，

L　：全信号線，

L10i：論理1→0変化した信号線i は1，

LBi　：論理0→1，1→0両方変化した信号線では1，

Lυi　：信号線iの論理（0，1）が観測可能地点（出力端子又はレジスタ）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，

Lv　：全信号線の論理の総和。

2) ファームウェア網羅率

$$FWC1 = \frac{\sum_i Si}{S} \quad \cdots\cdots(19)$$

$$FWC2 = \frac{\sum_i Pi}{P} \quad \cdots\cdots(20)$$

ここで

Si：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，

Pi：パスi が実行されると1，

P　：取り得る全てのパス数。

8. むすび

HALⅢ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・評価及び応用領域について述べた。HALⅢ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HALⅢは，故障シミュレーション，網羅率測定等多様な使い方が出来る。これはHALⅢが高速性を維持しながら，柔軟な使い方が出来るアーキテクチャーとなっているためである。

HALⅢ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・プロセッサー・システムであるためまだまだ全体のオーバー・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，更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。

参考文献

1) Koike,N.,Ohmori,K.,Kondo,H.and Sasaki,T.:A High Speed Logic Simulation Machine,PP.446-451,Compcon 83 Spring.

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,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,CAD-6,1,PP46-54 (Jan.1987).

7) Takasaki,S.,Sasaki,T.,Nomizu,N.,Ishikura,H.and Koike,N.:HALⅡ:A Mixed Level Hardware Logic Simulation System,Proc.23rd 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,PP.162-165 (1986).