超大形電子計算機 HITAC M-200 H の論理シミュレーション†

大野泰廣†† 森田 宏†† 小高俊彦††
宮本俊介††† 三善正之†† 鬼塚宣彦††

現時点で世界最高速の汎用処理装置 HITAC M-200 H の開発は論理シミュレーションを用いて行われ，従来大形機の開発に比較して，開発期間を大幅に短縮することができた．

LSI を使用する論理装置の開発には，従来ハードウェアシミュレーションが必須とされていたが，論理シミュレーション手法がそれに代って十分実用になることを示した．

M-200 H の開発に使用された論理シミュレーションシステムは，TSS で設計データの入力・修正，結果の確認を行うことができ，レーザビームプリンタによる論理回路図作成プログラム，レジスタトランスファレベル言語とゲートレベル言語で記述されたハードウェアモデルを扱うことのできる論理シミュレータを中心としており，とくに使いやすさに種々の工夫がなされている．

しかし，短期間に効率よく大規模な論理装置のシミュレーションを行うには，テスト実施手順および運用がきわめて重要である．この観点から，本論文では，M-200 H の論理シミュレーションの進め方について，テスト実施手順の考え方，テスト項目の設計法，運用方法などについて論ずる．

1. はじめに

HITAC M-200 H は，新たに開発した最大 550 ゲートの大規模集積回路（以下 LSI と略す），アクセスタイム 7 ns の高速半導体メモリなど最新の半導体技術を結集した現時点で世界最高速の超大形電子計算機であり，これまでの日立の最上位機種である M-180 や H-8800 の 2.5～3 倍の処理能力をもつ．論理規模も 3 倍以上となり大規模・複雑化している．

このように LSI を多用する超大形論理装置の開発には，設計の自動化 (DA : Design Automation) 技術，なかでも論理シミュレーション，実装設計の自動化，診断およびテスタ技術が不可欠である．

論理シミュレーションについて種々報告されているが，論理規模が数 10 万ゲートの超大形計算機全体をゲートレベルで実施した例は少ない．このような論理シミュレーションにおいては，論理シミュレーションシステム自体の機能・性能だけでなく，論理シミュレーション利用技術（テスト実施手順，運用方法）がきわめて重要であり，以下で M-200 H の論理シミュレーションについて，システム概要・シミュレーションの進め方およびその成果について述べる．

† Logic Simulation of Very Large Computer HITAC M-200 H by YASUHIRO OHNO, HIROSHI MORITA, TOSHIHIKO ODAKA (Kanagawa Works, Hitachi Ltd.), SHUNSUKE MIYAMOTO (Central Lab., Hitachi Ltd.), MASAYUKI MIYOSHI and NOBUHIKO ONIZUKA (Kanagawa Works, Hitachi Ltd.).

†† (株)日立製作所神奈川工場
††† (株)日立製作所中央研究所

2. 論理設計の手順

図 1 は大形計算機を開発する設計過程の概要である．

大形計算機の開発機関を左右する重要な問題の一つに論理不良のデバッグがある．このような設計は多数の設計者により行われるため論理不良も発生しやすい．

特に最近の大形機では，論理規模も大きく，複雑化し，LSI が多用される．論理不良が LSI 内に多発すれば，それらの LSI を作り直すなどの作業が生じ，開発期間に著しい影響を与える．このため論理不良を管理し，論理不良を早期に摘出する必要性が一段と高まりつつある．

論理不良の摘出は，論理設計が完了した時点から開始される．論理設計者自身およびチェック担当者によって定められた期間内でチェックが行われた後，実機の製作・組立てが行われる．その後，試験プログラムなどを用いて残された不良を摘出し，最終的に各種の検査を行い合格したのち，製品として出荷する．

処理装置の開発過程での論理不良の摘出状況の典型的な例を**図 2** に示す．横軸の時間の経過とともに縦軸に摘出された論理不良の累積値が増加してゆく様子を示す．

曲線①は従来の小規模集積回路 (SSI) や中規模集積回路 (MSI) を使用した処理装置の開発過程を示す．

LSI を多用した処理装置では，実機パワーオン後の論理不良件数が①の場合と同じであれば，LSI の作り

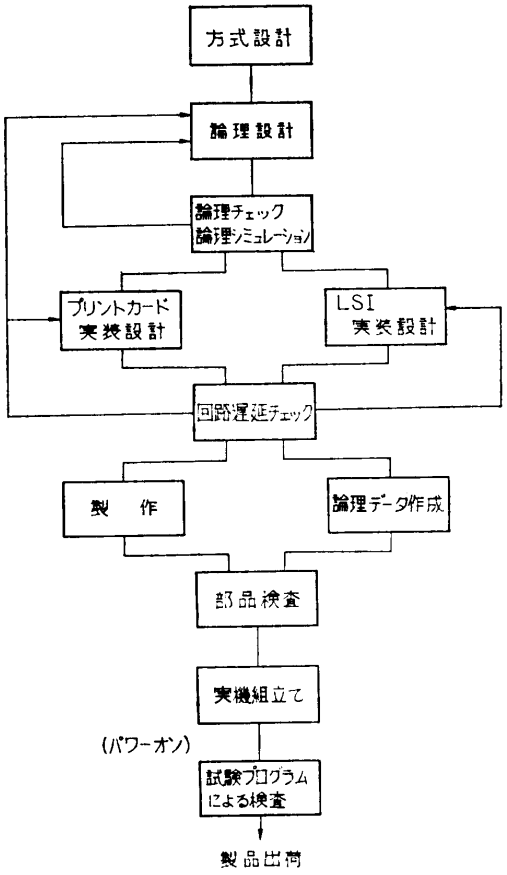


図 1　大形計算機の設計過程
Fig. 1　Design process of large computer.

直しに必要な時間のために，パワーオン後の不良累積過程は曲線②のようになる．この場合すべての不良を修正するまでの時間，すなわち開発期間が検査完了②の時点までかかる．

このため，実機組立て前の不良摘出を十分に行い，曲線③のような開発を目標としなければならない．

LSI を使用する論理装置の開発には，従来ハードウェアシミュレーションが必須とされてきた．しかし，この方法では LSI 化する論理の再設計が必要であり，論理不良の危険性が高いだけでなく，開発期間も長い．

M-200 H の開発では，レジスタトランスファレベル言語とゲートレベル言語で記述されたハードウェアモデルが扱える論理シミュレータを中心とした論理シミュレーションシステムを用いて，できるだけ短期間に効率よく論理シミュレーションを行うためのテスト実施手順を工夫することにより，上記の目標を達成することができた．

また，信号線の配線長，配線径路に起因する回路遅延時間不良については回路遅延チェックプログラムを併用し摘出した．

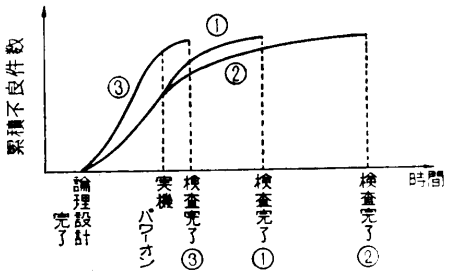


図 2　論理不良の摘出過程
Fig. 2　Relationship between design errors and development time.

3. 論理シミュレーションシステム

3.1　システムの狙い

超大形電子計算機の開発は，論理規模が大きく複雑であり，しかも多数の論理設計者によって分担されて進められる．論理シミュレーションによる論理の確認，論理不良の摘出を可能な限り効率よく行うため，論理シミュレーションシステムの建設にあたり，次の点を留意した．

（1）　論理記述データを短期間に少ない工数で設計データベース上に作成・更新できる．

（2）　論理変更後，直ちに最新の読みやすい論理回路図が設計者に提供される．

（3）　論理シミュレーションのテストデータが，簡単に用意できる．

（4）　装置全体の論理シミュレーションの処理が，短時間に行える．

（5）　論理シミュレーション結果の確認が，容易にできる．

（6）　設計進捗状況に応じて任意の段階で，論理シミュレーションが行える．

（7）　論理シミュレーションシステムは，実装設計 DA システムなどほかの DA サブシステムと同一の設計データベースを共用する．

3.2　システム概要

図 3 に，論理シミュレーションによる論理検証の流れを示す．このシステムは，HITAC M-180 を用いて設計室内に設置された VDT (Video Data Terminal; H-9415) から TSS で運用されている．入力データの準備，ジョブの申込み，結果の確認などすべての作業が短時間にできる．

（1）　論 理 入 力

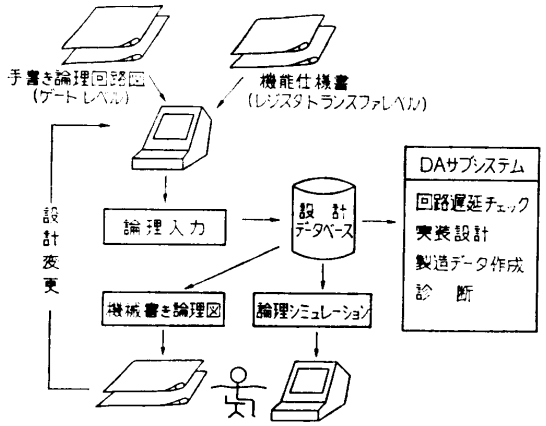


図 3 論理シミュレーションシステムの概要
Fig. 3 Outline of logic simulation system.

論理シミュレーションの対象となる論理は，2通りの形式で記述され，設計データベースに格納される．手書き論理回路図のゲートレベル論理記述はデータシートへ転記することなく VDT 上で直接行うことができる．このとき，文法の誤りだけでなく，種々の論理的・物理的制約条件が守られているか即時にチェックされ，誤りがあれば VDT 画面に表示される．

詳細論理設計が終っていない論理ユニットやすでに設計が終っていて検証する必要のない論理の動作はレジスタトランスファレベル論理記述言語で記述される．設計進捗状況に応じて任意の段階で論理シミュレーションを行ったり，装置全体の論理シミュレーションをより少ないメモリで短時間に行うためレジスタトランスファレベル言語が必須である．

（2） 機械書き論理回路図

設計データベースに格納された論理データから論理回路図が M-180 に接続された LBP (Laser Beam Printer ; H-8195) を用いて自動作成される．論理回路図には，信号のクロスレファレンス（先行情報）がプログラムにより付加される．多数の論理設計者が分担して設計を行う超大形電子計算機の開発には，ほかの設計者の設計した論理を追跡することが必要であり，クロスリファレンスはその手段を提供する．

機械書き論理回路図は，設計者に設計データベースの内容を見やすい形式で示すものであり，論理変更後，直ちに設計者へ提供されなければならない．LBP による論理回路図作成時間は，A3 図面1枚あたり約10秒と高速であり，この目的に適している．

（3） 論理シミュレーション

設計者は，あらかじめ作成したテスト仕様に従って，論理シミュレーションに必要なテストデータを作成し，VDT から設計データベースに登録後シミュレーションジョブを依頼する．シミュレーションの結果は，種々の形式に編集されリストへ出力されるとともに設計データベースにも格納される．

設計者は，機械書き論理回路図を参照しながら，設計データベースに格納されているシミュレーション結果を VDT に表示させ，論理回路の確認・不良原因の追求を短時間に行うことができる．

論理シミュレーションの結果，論理不良が摘出されたら論理データを修正後再びシミュレーションを行い確認する必要がある．このように論理シミュレーションは，インタラクティブであり，その作業サイクルを短縮することが開発期間の短縮になる．そのため装置全体の論理シミュレーションをできるだけ短時間で行えるよう，論理シミュレータの性能向上だけでなくシステムの運用にも工夫がなされている．

（4） 設計データベース

論理シミュレーションシステムは，回路遅延チェック・実装設計・診断などのほかの DA システムと同一の設計データベースを使用する．したがって，論理シミュレーションによって検証された論理データがそのまま後続の DA サブシステムに引渡される．すなわち，検証済の設計情報から直接に製造・検査データが作られ，信頼性が高い．

3.3 論理記述言語

（1） ゲートレベル論理記述言語

図 4 に記述例を示す．記述は論理シンボル単位に行われ，その文法は簡単である．論理シンボルにはゲート・フリップフロップなどのほかシフトレジスタなどのマクロ機能や LSI がある．論理回路は SSI・MSI・LSI・プリントカード・バックボードなどの階層で実現されるが，どの階層でも同じ言語を用いて同様の方法で行うことができる．

手書き論理回路図を VDT から直接入力するとき，記述量をできるだけ少なくするため，次のような機能を用いている．設計者はまず論理回路に示された論理シンボル名をタイプインする．VDT 画面には，図4の形式でコマンド，論理シンボル名，ピン番号が表示される．設計者はゲート名，信号名をタイプインすればよい．

（2） レジスタトランスファレベル論理記述言語

表 1 に示すコマンドを用いて，論理ユニットの動作が記述できる．**図 5** はその記述例である．文⑥に記述

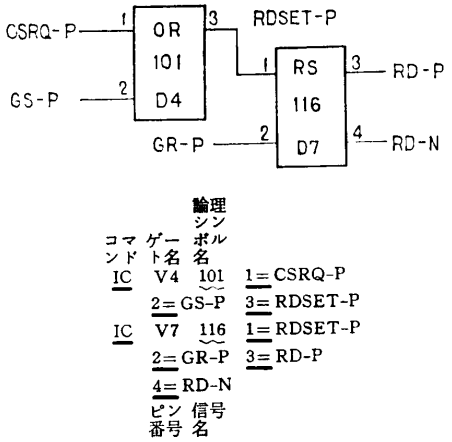
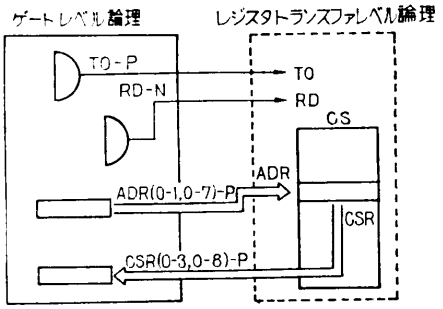


図 4 ゲートレベル論理記述例（＿部をキーインすると＿一部が表示される）
Fig. 4 An example of gate level logic description.



① GS TO TO-P
② GS RD RD-N
③ GS ADR ADR (0-1, 0-7)-P
④ GS CSR CSR (1-3, 0-8)-P
⑤ DS CS (0000-FFFF) 36
⑥ ON TO=1 & RD=1;
⑦ EX 50: CSR=CS (ADR)

図 5 レジスタトランスファレベル論理記述例
Fig. 5 An example of register transfer level logic description.

された条件 (TO=1 かつ RD=1) が成立すると，50 ns 後に ADR が示すアドレスに従い擬似メモリ CS の内容が CSR へ転送される．

この言語により ROM, RAM, PLA，マイクロプロセッサなども容易に記述できる．たとえば，マイクロプロセッサ Intel 8080 は，1500 ステートメントで記述できた．

論理シミュレータは，ゲートレベル言語で記述された論理とレジスタトランスファレベル言語で記述された論理を同時に扱うことができる．

3.4 論理シミュレータ

（1） 論理シミュレータの構成

図 6 に示すように，論理シミュレータは設計データベース上の 4 種のファイルを扱う．

表 1 レジスタトランスファレベル論理記述言語コマンド
Table 1 Commands of register transfer level logic description language.

コマンド	機能
DS (Define Storage)	擬似レジスタ，アドレスづけされた擬似メモリと定義する．
GS (Group Signal)	レジスタトランスファレベル論理で用いられるゲートレベル論理の信号群を定義する．
ON/EX (ON/EXecute)	コマンド ON で記述した条件が成立したら，コマンド EX で記述した処理が，指定された時間経過後実行される．

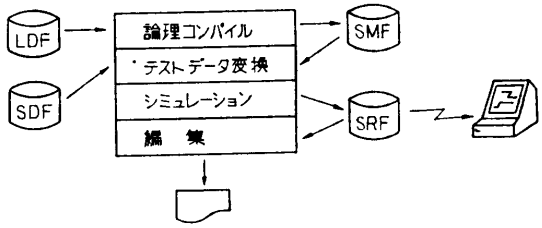


図 6 論理シミュレータの構成
Fig. 6 Configuration of logic simulator.

LDF (Logic Description File) には論理データ（ゲートレベル／レジスタトランスファレベル）が格納されており，これらはシミュレーションに適したテーブルの形に変換されて SMF (Simulation Master File) に格納される．この処理を論理コンパイルと呼んでいる．論理データに変更のない限り，SMF を用いて何回でもシミュレーションできる．

SDF (Simulation Data File) にはシミュレーション対象論理回路を動作させるのに必要なマイクロプログラム，試験プログラム，外部からの入力信号のほか，シミュレーション結果の編集方法を指定するコマンドなどが格納される．

シミュレーションは，タイムマッピング法による信号変化追跡方式を用いて，3 値 (0, 1, X) で行われる[2]．極力少ないメモリにシミュレーションテーブルを格納するため，テーブルを階層化するなどの手法を用いている．また前述のようにゲートレベル／レジスタレベル論理記述を同時に扱うことにより，シミュレーションの性能向上を図っている[4]．

シミュレーションの結果は，SRF (Simulation Result File) に格納されたのち，編集されてリストへ出力されるとともに，設計者の指示により VDT へ表示できる．

（2） シミュレーションテストデータコマンド

テストデータは**表 2** に示すコマンドで記述される．

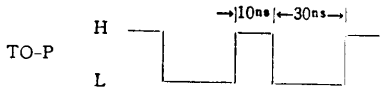
表 2 シミュレーションテストデータコマンド
Table 2 Commands of simulation test data.

コマンド	機能
LD (LoaD)	コマンドDSで定義した擬似レジスタ，擬似メモリに初期値を格納する．
ST (STimulus)	指定した信号に信号波形を与える．
SD (Strage Dump)	擬似レジスタ，擬似メモリの内容をメモリダンプ形式で表示する．
SS (Snap Shot)	コマンドGSで定義した擬似信号の値を，コマンドTPで指定した信号が変化する時点でサンプリングして表示する．
TC (Timing Chart)	シミュレーション結果を信号単位に信号波形で表示する．
TP (Test Point)	コマンドSSとともに用いて，サンプリング同期信号を定義する．
CF (Cale File)	あらかじめ作成されたテストデータファイルからテストデータを呼び出し展開する．

図7にテストデータの記述例を示す．文①は，$(1000)_{16}$番地から8ビットごとにパリティを自動生成して擬似メモリCSにマイクロプログラムを格納する．文②は図7(b)の信号波形を信号TO-Pに与える．文③はコマンドGSで定義した信号ADRに，はじめの100 nsは強制的に信号値 F0F0を与えるが，次の 2000 ns の間，信号 ADR は外部からの強制信号値に従うことなく自由に動作する．この機能により論理回路の初期値設定を容易に行える．文④〜⑦はシミュレーション結果の編集形式を指定している．文④は擬似メモリCSのダンプリストを指示し，文⑤は ADR, CSR の内容を文⑥で指定した信号 TO-P, RD-N がそれぞれ変化した時点でサンプリングし，表示することを指示している．これをスナップショットと呼ぶ．**図8**(a)

```
① LD   CS/P  1000: 80000000  84400000
                    32476FB0 ……
② ST   TO-P  (H 10  L 30)*10000
③ ST   ADR  F0F0/100 T/2000  0F0F/200
④ SD   CS/P
⑤ SS   ADR.0  CSR.1/P
⑥ TP   TO-P  TI-P
⑦ TC   TO-P  RD-N
```

(a) シミュレーションテストデータ

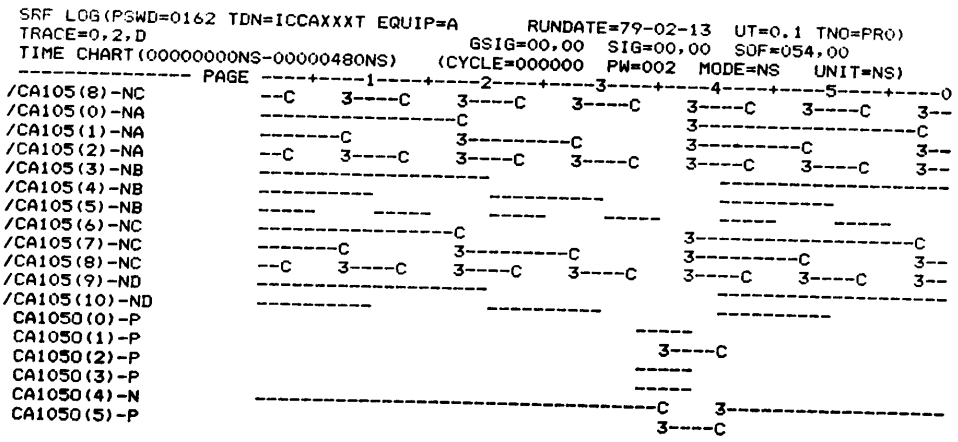


(b) TO-P の信号波形

図 7 シミュレーションテストデータ記述例
Fig. 7 An example of simulation test data.

```
### CYCLE ###          <000053>              <000054>
CSARDD.0                  145                   145
   BKD.0                    0                     0
DFSTOP.3                    0                     0
   PAL.2  #00@00@26#62#00@00@00@00@  00@00@26#62#00@00@00@00@
   WAR.3   00@00@26#5E#00@00@00@00@  00@00@26#5E#00@00@00@00@
   WBR.3   00@00@00@00@00@00@00@00@  00@00@00@00@00@00@00@00@
   WCR.3  #80#00@26#62#00@00@00@00@  80#00@26#62#00@00@00@00@
   WDR.3   E0#00@00@04#00@00@00@00@  E0#00@00@04#00@00@00@00@
  SI07.1  #00@00@00@00@00@00@00@00@ #E0#00@00@04#00@00@00@00@
 GSH07.1  #00@00@00@00@00@00@00@00@ #E0@00@00@04#00@00@00@00@
  SHOL.2  #00@00@00@00@00@00@00@00@  00@00@00@00@00@00@00@00@
  GP07.1  #00@00@26#78@00@00@00@00@  00@00@26#78@00@00@00@00@
  GNCT.2                    #FFFF                 #00FF
 GSHFT.2                   00@00@                #E0#00@
 SHNCT.1                    0000                  #0101
    WP.3                   #00@                   00@
```

(a) スナップショット



(b) タイムチャート

図 8 論理シミュレータ出力結果例
Fig. 8 Examples of logic simulator output.

にその出力例を示す．レジスタの値がマシンサイクルごとに表示され，論理動作の確認が容易に行える．文字コード@，#によってパリティが識別できる．図8(b)は，VDT に表示されたタイムチャートの例である．論理が期待通り動作していない場合に，信号名とマシンサイクルを指定すれば信号波形が確認できる．これは，実機検査時に使用されるシンクロスコープなどと同様の機能を果し，それより使いやすい．信号波形に16進数（ただし0はスペース，Fは－）を用いておりコンパクトな表示が可能になっている．

4. M-200 H の論理構成

M-200 H での論理シミュレーションについて述べる前に，論理構成の概要を説明する．

図9に，M-200 H 中央処理装置(CPU)の構成を示す．CPU は主記憶装置 (MS)，演算処理装置 (BPU)，入出力処理装置 (IOP)，コンソールサービスプロセッサ (SVP) とからなる．

ここで取上げている論理シミュレーションの対象は BPU である．

BPU は，主記憶制御ユニット (SCU)，命令制御ユニットと (IU)，演算ユニット (EU)，およびサービスユニット (SVU) とからなる．これらのユニットは独立に並列動作できる．

（1） 命令制御ユニット

IU は命令を読出し，解読し，オペランドを読出して，演算を実行できる状態にして EU に送る．1マシンサイクルピッチに命令を EU にセットアップできるようなパイプライン制御方式にしている．

（2） 主記憶制御ユニット

SCU は 64 kB の容量をもつ高速メモリによるバッファ記憶 (BS) をもち，IU からの命令読出し，IU，EU からのオペランド読出し，書込み要求を受付け，優先順位に従い処理する．

（3） 演算ユニット

EU は，IU によりセットアップされた命令を実行する．Load, Add, Compare などの主要な命令は1マシンサイクルで処理することができる．

演算器には，直列演算器，並列演算器，シフタ，高速乗除算器などがある．高速演算を行うため，並列演算器，シフタ，乗除算器などのデータ幅は64ビットを基本としている．

（4） サービスユニット

SVU は，SVP との接続機能および保守機能をもつ．

論理シミュレーションは，上記の各ユニット単体で

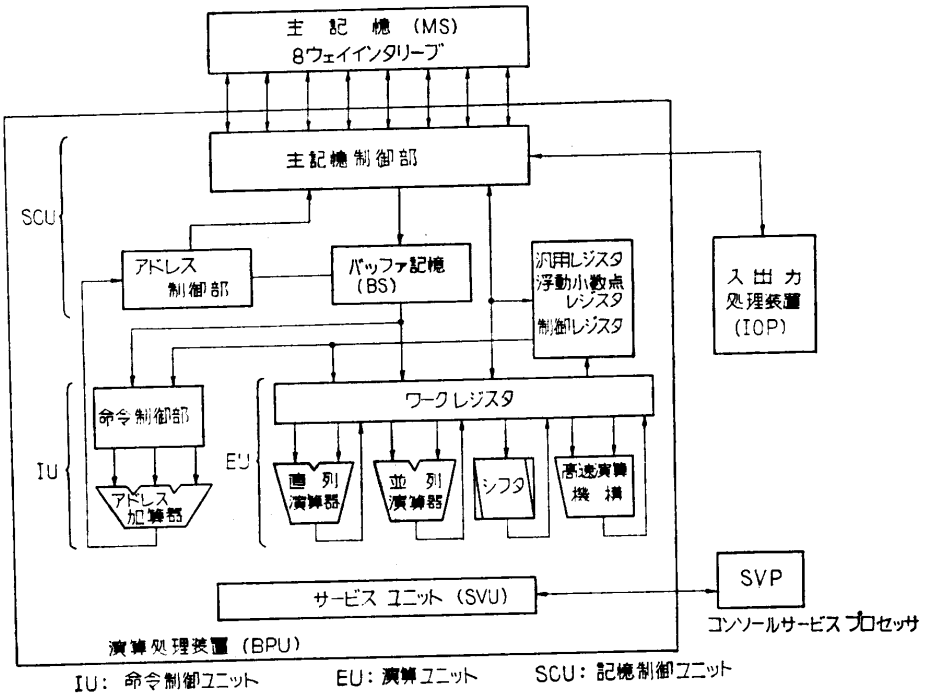


図9 M-200 H 中央処理装置の構成
Fig. 9 Configuration of M-200 H central processing unit.

のシミュレーションと，各ユニットを接続した状態でのシミュレーションとを行った．前者はユニットシミュレーション，後者は装置シミュレーションと呼ばれている．

5. M-200 H 論理シミュレーションの進め方

論理シミュレーションは，多量の計算機資源を必要とする．限られた計算機資源で短期間に超大形機の論理シミュレーションを行うため，以下の点に留意した．

（1） 計算機時間とジョブターンアランド時間を短縮するため，シミュレーション対象論理を階層化し，論理規模を必要最少限とする．

（2） シミュレーション期間を短縮するため，各階層のシミュレーション対象論理ごとに有効なテスト項目を選択し，テスト項目を必要最少限におさえる．

5.1 シミュレーション対象論理の階層

ソフトウェアのテストと同様に，ハードウェアのテストも基本構成要素を確認したのち，これらが組合わされた複雑な論理の確認へと進む，いわゆる small start の考え方が適用できる．同時に必要な計算時間の短縮，テストデータの作りやすさ，確認の容易さを考慮して，シミュレーション対象論理をLSI単体，論理ユニット，装置に階層化した．

LSI 論理に不良が発生すると，その再製に時間がかかるため，LSI 単体のテストでは必要な入力変化を与え，期待された動作がなされているか徹底的に検証する必要がある．

論理ユニットは，そのインタフェースが内部仕様として明確に定義されており，テストデータの作成，結果の確認が比較的容易である．この段階で，論理ユニットの基本指令に対する動作を確認後，装置全体へのシミュレーションへと段階的に拡大していった．

5.2 シミュレーションテスト項目の作成方針

シミュレーション対象論理の各階層において，できるだけ少ないテスト項目で論理を確認するため，次の方針によりテスト項目を選択しテストを行った．

（1） small start

実機検査の場合と同様に，基本構成要素の機能を確認後，次第にテスト対象範囲を拡大する．実機検査で必須の基本機能を早期に確認すると同時に，論理不良発生時，不良原因の追求を限定できる．

（2） 独 立 性

テストされる機能が重なり合わないようにテスト項目を用意する．同一の不良を多数のテストで指摘してしまう無駄を避ける．

（3） 複 合 性

基本機能確認後は，これらの機能が組合わされたできるだけ複雑な機能のテストを行う．テスト項目あたりの不良摘出効率を上げるためである．

（4） 並 列 性

テストの実施方法として，短期間にテストを終えるため互いに独立なテスト項目は，可能な限り多面的に並列して行うことにした．

5.3 装置 (CPU) シミュレーションの狙い

装置全体の論理シミュレーションは，一回あたりの計算機時間も長く，用意しなければならないテストデータも多い．しかし，次の理由により装置全体のシミュレーションは重要である．

（1） LSI，論理ユニットのシミュレーションが論理方式に着目した内部仕様のテストであるのにくらべ，装置全体のシミュレーションは最終的に確認されるべき機能に着目した外部仕様のテストである．

（2） 装置全体を動作させることにより，LSI，論理ユニットの階層では作成困難なテストパターンの組合せに対する機能が確認できる．また，論理シミュレーションで確認された機能について信頼がもてるため，実機検査時，不良原因の追求を限定することもできる．

（3） 設計者に理解しやすいプログラム，マイクロプログラムでテストデータが作成できる．既存の試験プログラム，MD (Micro Diagnosis) などを利用する

表 3 シミュレーション対象論理の階層とテスト項目の階層との関係
Table 3 Relationship between simulated logic and test items.

実施順序	対象論理の階層	テスト項目の階層		テスト内容
1	LSI	LSI 動作テスト		LSI 機能を，多数の入力データを用いて検証する．
2	論理ユニット	論理ユニット動作テスト		論理ユニット間インタフェース信号の指令に対し，対象論理ユニットが正しく応答するかを検証する．
3	論理ユニットまたは複数論理ユニット	マイクロファンクション動作テスト		指定されたマイクロプログラム機能が正しく動作するかを検証する．
4	処理装置	命令動作テスト	単独命令動作テスト	命令が一命令，正しく動作するかを検証する．ここで検証されたものが命令のハードコアとして使用できる．
5			複合命令動作テスト	先行制御の乱れを複数の命令を使用し，各種状態を変えながら多くの状況を検証する．
6	複数論理ユニットまたは処理装置	特殊状況動作テスト		実機検査機能，マシンエラー処理，高負荷時の動作など，特殊な状況での動作の正しさを検証する．

ことにより，テストデータ作成時間も短縮できる．

表3に論理シミュレーションの実施順序と，シミュレーション対象論理の階層およびテスト項目の階層との関係を示す．

6. 装置 (CPU) シミュレーションの方法と具体例

装置シミュレーションをより効果的に行うため，シミュレーションのテスト環境の設定に次の点を留意した．

（1） 可能な限り実機検査と同一の環境を設定する．たとえば，試験プログラムは実機と同じものを用いる．

（2） 生産に直結しているゲート論理のレベルで極力検証を行う．

6.1 擬似プロシジャ

ゲートレベル言語だけで記述されたシミュレーション対象論理を扱うことが，実機に最も忠実である．しかし，テストの目的に直接関連しない論理のシミュレーション時間の短縮，メモリ容量の削減，初期値設定のやりやすさなどから，論理の一部をレジスタトランスファレベル言語で記述するのが効率的である．これを擬似プロシジャと呼んでいる．

擬似プロシジャを多用すると，ゲートレベルでの検証範囲をせばめることになるため，メモリ回路とその周辺にのみ擬似プロシジャを用いることとした．

表4に M-200 H の論理シミュレーションで使用した擬似プロシジャの例を示す．

メモリ制御の擬似プロシジャは，ゲートレベル論理

表 4 擬似プロシジャの構成
Table 4 Configuration of pseudo procedure.

メモリ制御	主記憶装置 read/write，パリティ発生制御 TLB (Translation Lookaside Buffer) 制御 バッファ記憶制御 制御記憶読出し制御 記憶保護制御
タイミング制御	タイミング発振制御
ディレイ制御	プラッタ間ディレイ値定義 特定信号ディレイ値定義
モード制御	構成制御情報定義 機器処理モード定義
シミュレーション制御	汎用レジスタ初期値設定制御 内部レジスタ初期値設定制御 シミュレーションスタート制御 強制シミュレーション打切り制御 仮対策制御

```
*** INSTRUCTION COMPARE STOP CONTROL
*** PARAMETER = STOP INSTRUCTION ADDRESS(IALCA)
*** DEFINITION
③ GS  IALCA   IALCA(1-3,0-7)-1P
④ GS  TSTOP   IXGIFSTOP-P
⑤ DS  TSTOPD 1
*** STOP CONTROL
⑥ ON  TSTOP=1  ;
⑦ EX  80 : TSTOPD=1  ;
⑧ ON  TSTOPD=1  ;
⑨ EX  STOP CSTP  ;
```

図 10 強制シミュレーション打切り制御ルーチン
Fig. 10 Compulsory suspending procedure of logic simulator.

と擬似メモリの間のデータ転送を行うものである．アセンブラやマイクロプログラムコンパイラにより作成されたオブジェクトは，この擬似プロシジャ内の擬似メモリに格納される．

タイミング制御とディレイ制御は，時間系の制御を行う擬似プロシジャである．マスタタイミングパルスがタイミング制御の擬似プロシジャで作成され，ゲートレベル論理で組まれたタイミング制御回路に与えられる．

モード制御とは，実機と同様に主記憶装置の構成制御情報，割込み許可動作などを指示する機器処理モード情報を，テストの目的に応じて容易に変更できるようにした擬似プロシジャである．

レジスタに初期値設定をしたり，目的のテストプログラムを起動したりするなど，シミュレーションを容易にするための擬似プロシジャもいくつか用意した．

図10に擬似プロシジャの例を示す．この擬似プロシジャは，実行中の命令アドレスがあらかじめ指定したアドレスに一致したとき，シミュレーションを強制的に打ち切るものである．この擬似プロシジャは CSTP と名付けられており，任意のテストデータ中から次の方法で呼び出し使用される．

① ST IALCA 422/9000

② CF CSTP

文①で停止すべきアドレス，すなわち IALCA の値を 422 番地に 9,000 ns の間保持する．実際のアドレス比較動作はシミュレーション対象論理内で行われ，実行命令アドレスが 422 番地になると，一致検出信号 TSTOP が 1 になる．文⑥で TSTOP が 1 にセットされたことを検出すると，80 ns 後に文⑦が実行され，擬似レジスタ TSTOPD に 1 がセットされる．文⑧でこれを検出し，文⑨でシミュレーションを停止する．

アドレスが一致しても 80 ns 間シミュレーションを継続するのは，その時点で実行されている命令処理の完了を待つためである．

```
 LOC  OBJECT CODE          SOURCE STATEMENT
001D48 41F0 BD88      R351    LA    15,R351X          SET NEXT ADR
001D4C D108 BD6C BD75         MVN   R351F(9),R351S    *** TEST ***
001D52 D508 BD6C BD7E R351B   CLC   R351F(9),R351E    CHECK DATA
001D58 4780 BD68              BE    R351Z
001D5C 8200 BD60              LPSW  R351CK            ERROR WAIT
001D60                        DS    0D
001D60 000E1F00       R351CK  DC    XL4'000E1F00'
001D64 00001D68               DC    AL4(R351Z)
001D68 45E0 C200      R351Z   BAL   LINK,INI          GO TO NEXT
001D6C                R351F   DS    9C          INITIAL DATA OP1
001D75                R351S   DS    9C          INITIAL DATA OP2
001D7E                R351E   DS    9C        EXPECTED DATA OP1
001D88                R351X   DS    0H
001D88                        ORG   R351F
001D6C 0123456789ABCDEF97.    DC    XL9'0123456789ABCDEF97'
001D75 FEDCBA987654321036     DC    XL9'FEDCBA987654321036'
001D7E 0E2C4A6886A4C2E096     DC    XL9'0E2C4A6886A4C2E096'
001D88 41F0 BDC8      R352    LA    15,R352X
```

図 11 簡単な試験プログラム例
Fig. 11 An example of simple test program.

6.2 マイクロプログラムを入力としたシミュレーション

small start の原則より，装置シミュレーションはマイクロプログラム命令を構成するマイクロファンクションの検証から開始した．

マイクロファンクションの大部分は演算機能に関係するものであり，演算ユニット (EU) 単独のユニットシミュレーションで検証されている．汎用レジスタへのデータ転送機能など命令制制御ユニット (IU)，主記憶制御ユニット (SCU) に関係するマイクロファンクションの検証が，マイクロプログラムを入力とした装置シミュレーションの目的である．

テストに用いられるマイクロプログラムは初期値設定ルーチン，検証対象のマイクロ命令および結果判定ルーチンという基本構造をもち 10〜20 マイクロ命令で作成される．このうち初期値設定ルーチンおよび結果判定ルーチンで使用するマイクロ命令は，すでにユニットシミュレーションで確認されたマイクロファンクションを用いて記述される．これは，不良原因の追求を容易にするためである．

6.3 命令を入力としたシミュレーション

装置シミュレーションの最も重要な目的は，試験プログラムを用いて各論理ユニットの相互関連動作を実機と同一の条件の下で検証することにある．これは，擬似的に実機をパワーオンしたのと同じであり，論理不良の定量的な管理と早期摘出を可能にする．

テストに用いられる試験プログラムは，2通りの方法で作成される．

(1) 数種のハードコア命令を用いて，初期値設定ルーチンおよび結果判定ルーチンを作成し，一般命令をテストする．これは small start の考え方に従った方法である．

(2) パイプライン制御機能の検証を目的として，任意の一般命令を使用し，命令間の動作をテストする．これは，前述の複合性の考え方に従ったテスト方法といえる．

簡単な試験プログラムの例を，**図 11** に示す．試験プログラムはアセンブラによってオブジェクト形式に変換され，擬似メモリとして定義された主記憶あるいはバッファ記憶に格納される．必要なマイクロプログラムも，同様の方法で制御記憶に格納される．

シミュレーションは，実機検査と同様に，リセット信号発生，レジスタ初期設定，スタート制御の後，命令読込み動作が開始される．試験プログラムの開始・終了アドレスも任意に指定できる．命令実行に際しては，設定されたモードやプログラム格納場所のちがいにより，アドレス変換動作やブロック転送動作を任意に起動できる．

これまでの経験から，パイプライン制御計算機は，分岐命令や割込みの発生などによりパイプラインに乱れが生じたとき，すでにデコードを開始していた命令の影響により論理不良が生じやすいということが分かっている．今回行ったテストの多くは，このパイプラインの乱れを意識的に発生させ，その動作が正常に行われるか確認するものである．たとえば，分岐命令を複雑に組合せたテストやストア動作の順序性が保証されるか検定するテストなど複雑な条件を引起すプログラムを多数作成した．

6.4 特殊機能のシミュレーション

以上のほかにも，下記のような特殊機能のシミュレーションを実施した．

(1) 実機検査を行う上で必要不可欠な機能（例え

ば，各種のアドレス比較による停止機能）のシミュレーション

（2） マシンエラー発生時の命令再実行機能のシミュレーション

（3） 主記憶制御装置を中心とした密結合マルチプロセッサ環境のシミュレーション

（4） チャネル制御装置を含んだ入出力処理のシミュレーション

特に，レジスタトランスファレベル言語をうまく利用すれば，実機では準備に時間のかかる特定の事象に同期させたマシンエラーの発生を比較的簡単に実現できる．マシンエラーに付随する動作は実機検査でも，状況設定，解析が難しいものであり今後この分野での論理シミュレーションが期待される．

7. 運　用

多数の設計者ができるだけ多くのテスト項目を短期間に消化するには，限られた計算機資源を有効に運用する必要がある．

（1） 設計作業の流れとジョブスケジューリング

図12（a）に論理シミュレーション作業の流れを示す．論理シミュレーションジョブは，計算機時間，メモリとも多量に使用する大形バッチジョブである．しかし，図12（a）（b）に示すとおり，シミュレーションの結果により次に行う作業が決まるという意味で，論理シミュレーション作業は本質的にマン・マシン・インタラクティブなものである．図12（b）に示す作業サイクルを各設計者が1日何回繰り返すことができるかにより，開発期間が左右されるといってもよい．すなわち，論理シミュレーションジョブを夜間あるいは週末に実行するのでは不十分である．われわれは，論理シミュレーションジョブを優先的に実行する専用ジョブクラスを設けることで，作業サイクルを短縮した．

（2） 論理変更とジョブ管理

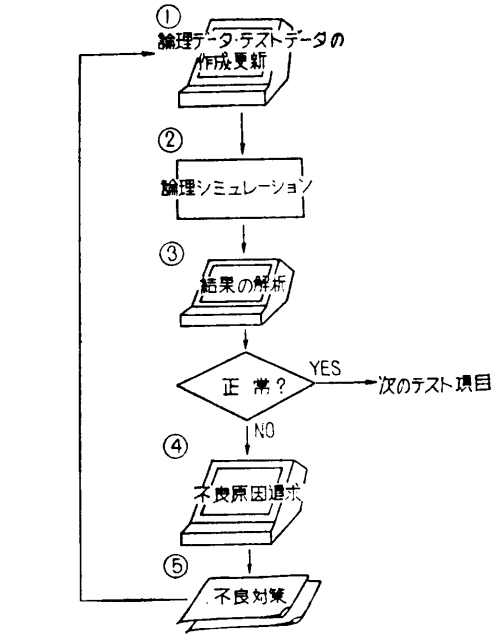
多数の設計者が同時に論理シミュレーションジョブを依頼しているので，計算機時間を節約するため論理変更に同期したジョブ管理を行った．

（a） 論理変更の同期化

論理変更のたびに論理コンパイルを行うのではなく，複数の論理変更を同期化し，論理コンパイルする．

（b） 論理不良と独立なテスト項目の選択

論理不良に対する論理変更は，必ずしも短時間に対策されるとは限らない．未対策の論理変更と独立なテスト項目を選択し，無意味なシミュレーションを防止する．



（a） 設計作業の流れ

（b） 論理シミュレーションのマン・マシンインタラクション

図 12 論理シミュレーション作業手順
Fig. 12 Work flow of logic simulation.

（c） 論理修正の確認

論理変更完了後，標準テストデータによるシミュレーションを行い，論理が正しく修正されたことを確認したのち，新しいテスト項目のシミュレーションに移る．

（d） ジョブの実行抑止

論理シミュレーション専用ジョブクラスへすでに投入されているジョブに影響する論理不良が摘出された場合は，直ちに VDT から関連するジョブの実行を抑止する．

8. 適 用 結 果

（1） 論理シミュレータの性能

表5に，一般命令を約10命令シミュレーションした場合の処理結果を示す．また LSI，論理ユニット，装置各階層でのシミュレーションに要する計算機時間が

表 5 論理シミュレータの性能 (M-180 使用)
Table 5 Performance of logic simulator.

条　件	
論理回路規模	371,000 ゲート
シミュレーション時間	3,650 ナノ秒
処理時間	
論理コンパイル	404 秒
テストデータ変換	132
シミュレーション	887
編　集	245
計	1,668 秒

表 6 論理シミュレーション速度 (実時間比, M-180 使用)
Teble 6 Processing time ratio of logic simulation.

LSI　シミュレーション	10^5 倍
ユニット　シミュレーション	$10^{6\sim7}$
装　置　シミュレーション	10^8

表 7 論理不良摘出効率の比較 (不良1件あたり)
Table 7 Performance of detecting design errors.

項　目	論理シミュレーション	実機検査(注)	備　考
設計者工数	1	3	論理データファイルの修正を含む
計算機時間	1	0.1	
論理変更ターンアラウンド時間	1	5～200	

(注) 論理シミュレーションの場合を1としたときの比

時間に対し，何倍に相当するかを表6に示す．

なお，論理シミュレータのイベント処理速度は，14,000イベント／秒である．

(2) 論理シミュレーションの効率

表7は，論理不良1件あたりの摘出効率を示している．論理変更のターンアラウンド時間に著しい相異がある．論理シミュレーションの場合は，設計データベース上の論理データを VDT から短時間に修正できるのに反し，実機検査の場合には，プリントカードの配線パターンカット・布線などの作業に時間がかかる．とくに LSI に不良が起った場合には，その再製に長時間かかる．

図13に，論理シミュレーション結果の内訳を示す．3回に1件の割合で論理不良が摘出されているが，前述したテスト方法の有効性を示すものと考えられる．

シミュレーションテストデータの誤りは擬似プロシジャの記述の誤り，環境条件の設定誤りなどによるものであり，その作成は容易ではないことが分かる．

(3) 実機検査期間の短縮

図14は，従来大形機と M-200 H との実機検査期間を比較したものである．M-200 H の論理規模は，従来大形機の3倍以上でもあるにかかわらず，実機検査

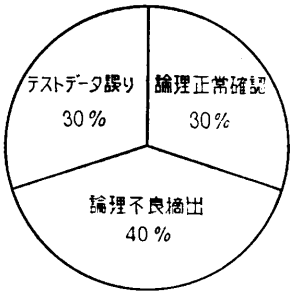


図 13 論理シミュレーションラン結果
Fig. 13 Items of logic simulation jobs.

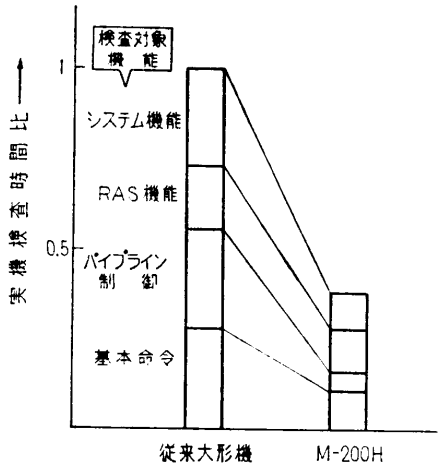


図 14 実機検査期間の比較
Fig. 14 Changes of hardware testing time.

期間は1/2以下に短縮された．とくに最もむずかしいパイプライン制御テスト期間は大幅に短縮できており，論理シミュレーションを本格的に適用した成果と考えられる．

9. おわりに

上述のとおり，論理シミュレーション手法を使って論理不良を早期に摘出し，超大形計算機 M-200 H を短期間に開発することができた．

LSI を使用する論理装置の開発では，従来ハードウェアシミュレーション手法が必須とされてきたが，論理シミュレーション手法がそれに代わって十分実用になることが示された．しかし，論理シミュレーションの効率は必ずしも満足できるものではない．設計工数および計算時間の低減のための論理シミュレータ自身の改善と，テスト項目の選定をより科学的に行うことが今後の研究課題といえよう．

最後にこのプロジェクトの推進にあたり，終始適切なご助言，ご支援をいただいた，波多野神奈川工場

長，中沢同副工場長，萱島旭工場長，堀越中央研究所部長，石原システム開発研究所主任研究員，はじめ，関係各位にお礼申し上げる．

参考文献

1) Ulrich, E. G.: Exclusive simulation of activity in digital networks, Comm. of ACM, pp. 102-110 (Feb. 1969).
2) Szygenda, S. A. et al.: Digital logic simulation in a time-based table-driven environment, Part I, Design Verification, Computer, pp. 24-36 (Mar. 1975).
3) Chappell, S. G. et al.: Functional simulation in the LAMP system, J. of Design Automation & Fault Tolerant Computing, pp. 203-215 (May 1977).
4) Ohno, Miyoshi, Sato: Logic verification system for very large computers using LSI's, the 16th Design Automation Conference, pp. 367-374 (June 1979).

(昭和54年8月31日受付)
(昭和55年5月15日採録)