Vol. 21 No. 4　　情報処理学会論文誌　　July 1980

LSI を含む階層的論理システムのための論理シミュレータ SIM/D†

稲垣耕作†† 槌田義之††
酒井丈嗣†† 矢島脩三††

論理シミュレータは論理システムの開発支援のための重要なツールである．本稿では，設計検証支援のため開発した論理シミュレータ SIM/D (Simulator for Design Verification) について述べる．

SIM/D は，ゲート・機能・レジスタ転送の3つのレベルの併用シミュレーションが可能である．SIM/D では，回路をいくつかのモジュールの階層構造として表現する．そのため，設計の各階層との対応がとれ，3つのレベルのシミュレーションを併用することにより，各段階における検証が容易になった．LSI の記述のためには，DDL (Digital System Design Language) に遅延時間の記述およびモジュールの階層性の概念等をとり入れたハードウェア記述言語 XDDL (Extended DDL) を用いる．LSI もひとつのモジュールとして記述することにより，LSI を含んだ回路も容易にしかも精密にシミュレートできる．

さらに，処理の高速化を達成するために，従来のイベント法の改良であるセレクティブイベント法，および回路の持つ意味をモジュール単位に与えて回路を階層的にシミュレートする意味記述法を新たに導入した．これらの方式を利用すればシミュレーションの正確さを犠牲にせずに，従来の方式と比較してきわめて大幅な高速化が実現される．なお，意味記述法は，高速化達成だけでなく，将来の自動検証にも利用できる重要な概念であり，また回路の階層的記述は，設計データベースおよびそれを利用した設計自動化システムへの発展においても本質的であると考える．

1. まえがき

論理システム開発の計算機による支援および自動化は，機器のディジタル化の急速な進展とともに，その必要性を増し，特に超LSIの出現は，今後開発支援・自動化技術に質的な変革をもたらすと予想される．

論理システムの設計検証支援のための論理シミュレータの研究の歴史は長く，CC-TEGAS3[1]，LAMP[2]，LOGOS2[3] など実用されているものも多い．論理シミュレータの順序制御方式は，テーブルドリブンによるイベント法[4]を基本としたものが一般的であり，高速性のためにも適している．素子のモデルはゲートレベル，機能レベル程度が従来の論理シミュレータには多いが，LSI を含む論理システムを対象とするには記述能力が不足すると考えられ，レジスタ転送 (RT) レベルの導入[5] などいくつかの提案がある[6],[7]．しかし，遅延時間の記述などを精密に行えなかったり，LSI を記述するには非現実的な方式など，問題が多い．

本論文で述べる論理シミュレータ SIM/D[8],[9](Simulator for Design Verification) は，LSI を多用した論理システムの設計検証の支援を目的として開発を進めてきたものであり，次のような特徴がある．

（1） LSI を含む論理システムの記述を容易にするため，ゲートレベル，機能レベル，RT レベルを併用し，ほぼゲートレベルの精密さで，かつ RT レベル併用による高速性を有する．

（2） モジュールの概念を導入し，回路記述およびそのシミュレーションの階層的な取り扱いを可能にした．

（3） シミュレーションの大幅な高速化を達成するために，新たにセレクティブイベント法および意味記述を導入した．

（4） RT レベルにおいても遅延時間の記述ができるハードウェア記述言語 XDDL を用い，より精密なシミュレーションを可能にした．また XDDL では信号線の並列処理が可能である．

（5） 設計の検証を容易にするため，各種の警告機能を盛り込んだ．

本論文では，第2章において SIM/D の構成を概説し，第3章においてその中心部であるゲート／機能レベルシミュレータ部 SIM/G の方式を述べる．第4章において SIM/D に導入した高速化方式であるセレクティブイベント法および階層性のもとでの意味記述に

† SIM/D: A Logic Simulator for Hierarchical Digital Systems Containing LSI's by KOSAKU INAGAKI, YOSHIYUKI TSUCHIDA. TAKESHI SAKAI and SHUZO YAJIMA (Department of Information Science, Faculty of Engineering, Kyoto University).

†† 京都大学工学部情報工学教室

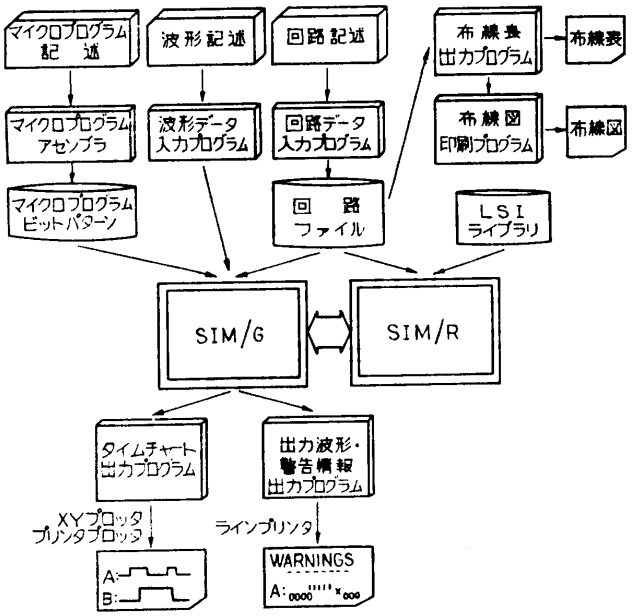


図 1 SIM/D とその周辺ソフトウェアの構成
Fig. 1 Organization of SIM/D and its peripheral software.

ついて詳述する．第5章では RT レベルシミュレータ部 SIM/R を説明し，第6章で SIM/G と SIM/R の結合シミュレーションについて述べる．第7章では SIM/D の使用例を示す．

2. SIM/D の構成

SIM/D の中心的なプログラムはゲートレベルシミュレータ SIM/G であり，ゲートレベルおよび機能レベルのシミュレーションとシミュレータ全体の順序制御を行い，また RT レベルシミュレータ SIM/R の起動を行う(**図1参照**)．入出力のために回路データ入力プログラム，波形データ入力プログラム，出力波形・警告情報出力プログラムおよびタイムチャート印刷プログラムを持ち，タイムチャートは XY プロッタあるいはプリンタプロッタに出力できる．なお SIM/R は単独でも動作可能である．

また周辺ソフトウェアとして，マイクロプログラム制御のシステムのためのマイクロプログラムアセンブラおよび回路データを多目的に利用してワイヤラッピング用の布線表および布線図を印刷するプログラムを持つ[10]．

プログラムは主として PL/I で記述され，一部 FORTRAN からなり，全体で約7,500ステップである．

3. SIM/G の構成と方式

3.1 概 要

SIM/G はゲートレベルおよび機能レベルのシミュレーションを行うことができる．基本素子としては，ゲート素子をはじめとして，機能素子として各種フリップフロップ，データセレクタ，メモリ等が扱える．

図2に SIM/G の構成の概略を示す．LSI の出力値の計算には，インタフェースを通して SIM/R を呼び出し，RT レベルによって LSI 内部のシミュレーションを行う．この方式では，順序制御部から LSI あるいは RT レベルのみで記述されたモジュールは，他の素子と同様にひとつの機能を持ったブラックボックスとして統一的に扱われる．

基本的には，テーブルドリブン，イベント法を用いている．信号の値は，0, 1の他，不確定を表わす X およびハイインピーダンスを表わす Z の4値であり，遅延時間は各素子ごとに与えることができる．フリップフロップ等の順序回路素子に対して，セットアップ／ホールドタイムその他入力に対する制限が守られているかを監視し，誤っている場合には警告を発する機能がある．

3.2 シミュレーション方式

SIM/G は基本的にはイベント法を採用しているため，信号線の値の変化というイベントを追跡してシミュレーションが進められる[11]．各素子に対応して，機能をシミュレートするサブルーチンが各々用意され，必要に応じ呼び出されて出力値が決定される．警告機能は素子のサブルーチン内に用意されている．なお第

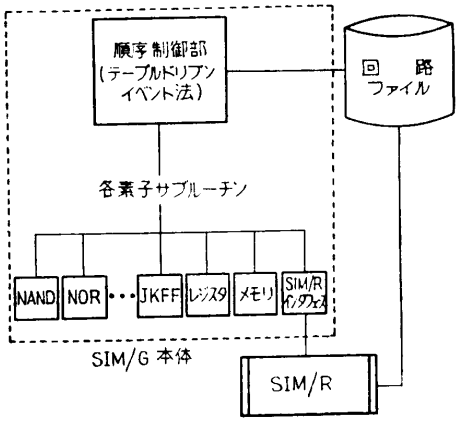


図 2 SIM/G の構成
Fig. 2 SIM/G.

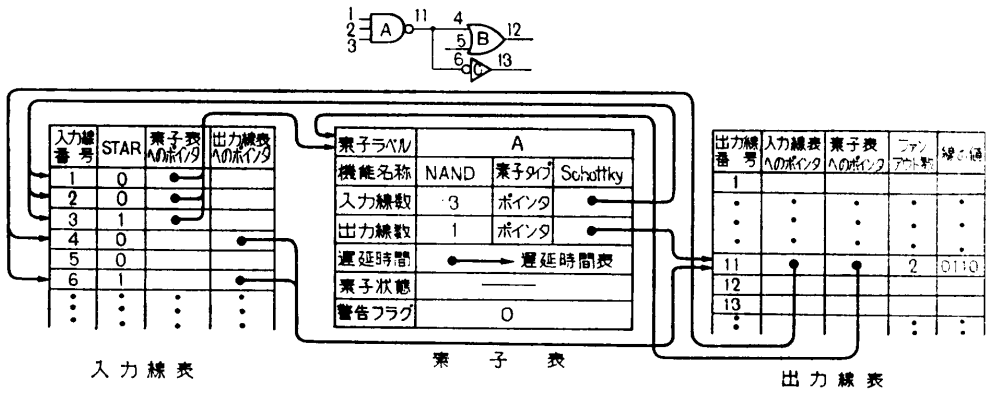


図 3 素子表，入力線表および出力線表
Fig. 3 Element table, input line table and output line table.

4章において本シミュレータの高速化方式を述べる．

3.3 データ形式

SIM/G ではテーブルドリブン法を採っているため，回路情報は表の形で蓄えられる．**図3**に主な表を模式化して示す．

（1） 素子表：素子ラベル，機能名称の他，入力線数，出力線数および接続している入／出力線のデータへのポインタおよび遅延時間を蓄えた表へのポインタがあり，フリップフロップ等の記憶を持つ素子のために内部状態を蓄えている．警告機能によって監視すべき素子は警告フラグで識別する．

（2） 入力線表：入力線が接続している素子および出力線表へのポインタを蓄える．他に，後述するセレクティブイベント法に利用するため STAR フラグがある．

（3） 出力線表：出力線が接続している素子および入力線表へのポインタの他，数時刻前からの線の値が蓄えられ，処理の都度更新される．

これらの表の他に，イベントを登録するためのフューチャイベント表等がある．フューチャイベント表は Δt-ループ[11]と呼ぶ環状の表になっており，現時刻で処理が完了すればイベントは消去され，その欄は後で再利用される．

4. 高速化方式

論理回路のシミュレーションは，対象となる回路が大規模になるに従いその処理時間は飛躍的に増大する．SIM/D ではシミュレーションの正確さを損わずに処理時間の短縮を実現するため，以下の2つの概念を導入した．

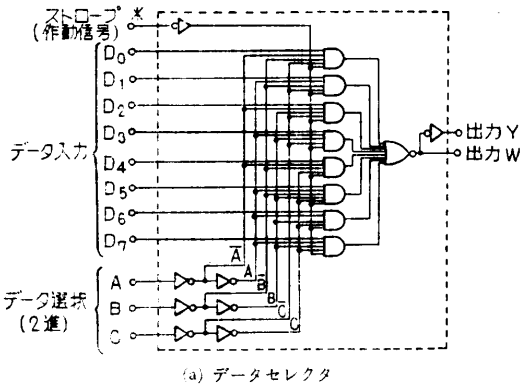
4.1 セレクティブイベント法

実際の論理回路において，信号の値や素子の状態の変化は同時に複数個所で起こるのに対し，計算機でのシミュレーションはその値や状態の変化を一度にひとつずつしか取り扱えない．そのためどのような順序でシミュレーションを行うかが問題となる．イベント法では，入力にイベントが発生した素子だけを出力値の変化の可能性のある素子として出力値を計算し，入力にイベントが発生しない素子の出力値の計算は行わない．

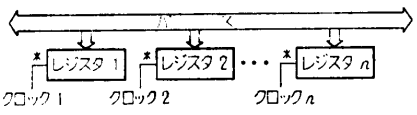
ところが，入力にイベントの発生した素子が必ずしも出力変化をもたらすとは限らない．たとえば，2入力 NAND ゲートにおいて一方の入力が0であれば他方の入力の変化は出力に影響が出ない*．また，レジスタやフリップフロップはクロックが変化しなければデータ入力の変化は出力変化をもたらさない．このように出力変化が起こらないことが明らかな場合は，そのイベントに対して処理を行わなければ処理の高速化が期待できる．この方式をセレクティブイベント法と名付け SIM/D に採り入れた．この方式は，**図4**（a）に示すような回路や，図4（b）のバス結合を持った回路などで特に効果的である．バス結合方式は現在の論理システムにおいて非常によく用いられるが，通常ある時刻におけるバス上の信号源は1つで，また受信素子も1つである．しかしながら従来のイベント法では，バス上の信号変化があるとバスにつながる素子すべての出力変化を計算しなければならなかった．しかし本方式の導入によりこのような場合にイベントの計算は1つの素子のみでよい．

この方式はイベント法の変形として導入が容易であ

* H. Y. Hsieh 他による FET 回路のシミュレーションの最近の文献[12]に素朴な概念がある．



(a) データセレクタ



(b) バス結合におけるセレクティブイベント法

図 4 セレクティブイベント法
Fig. 4 Selective event method.

る．以下に処理手順を示す．

（1） イベントが発生すれば出力変化を起こす可能性のある入力線に＊をつける．

（2） ＊のつけられた入力線にイベントが発生したときにのみ，素子の出力値の計算および＊のつけかえを行う．

例は**図 5** を参照されたい．

4.2 意味記述

4.2.1 階層性と意味記述

このような考えを発展させることにより，さらにシミュレーションの高速化を達成できる．

通常論理システムはモジュールの階層構造をとって設計される．しかもシステム全体あるいはモジュールは，クロック信号，ゲート信号，ストローブ信号等のなんらかの特別な信号により制御されているとみなせることがほとんどである．しかし一般にはシステムのそのような制御のされ方を，回路の局所的な構造のみから判定することは困難である．たとえば**図 6** に示す AND 素子は入力について対称な回路であるが，入力Aは頻繁に変化する信号であり，入力Bはそのゲート信号であるとする．このような場合，出力Cの計算は入力Bの値が1の時にのみ行えばよい．このような使い方は現実の回路において非常に多く，対称回路が必ずしも対称的には使われず非対称な使い方をされていることになる．

さらにモジュール単位で回路が制御を受けている場

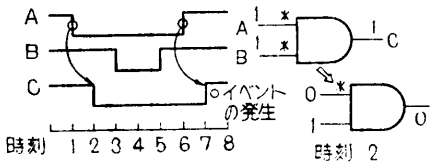


図 5 セレクティブイベント法の例
Fig. 5 An example of selective event method.

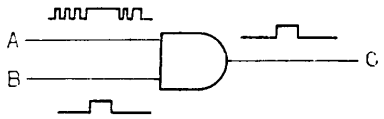


図 6 非対称的に用いられた対称回路
Fig. 6 Symmetrical circuit used asymmetrically.

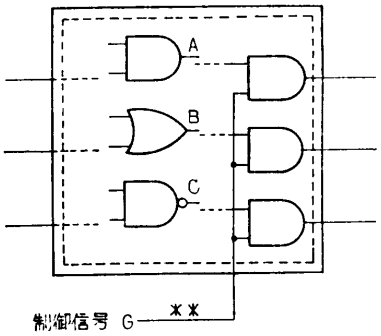


図 7 制御信号により制御されたモジュール
Fig. 7 A module controlled by a control signal.

合を考えると，そのような使われ方を回路の構造記述をもとにして判定することはほとんど不可能である．回路のこのような使われ方はむしろ設計者が与えた意味であるとするのが妥当であり，そのような意味を別に記述してシミュレータに与えてやることにより，シミュレーションの高速化が期待できる．さらに，この意味記述は論理システムの自動検証への研究の発展においても意義があると考える．

4.2.2 意味記述に基づく処理法

図 7 のモジュールにおいて，制御信号が0の間はモジュールの動作は出力に影響を与えない．それゆえ制御信号が1になったときにのみシミュレーションを行えばよい．このときモジュールは活性化されたと呼ぶ．

モジュールの制御線には＊を2つつけ，そのつけかえは行わない．入力線表 STAR 欄は，0のとき＊なし，1のとき＊1つ，2以上のとき＊2つとした．2以上のときその値は，**図 8** に模式化した制御線表へのポインタを兼ねる．制御線表から条件表における条件

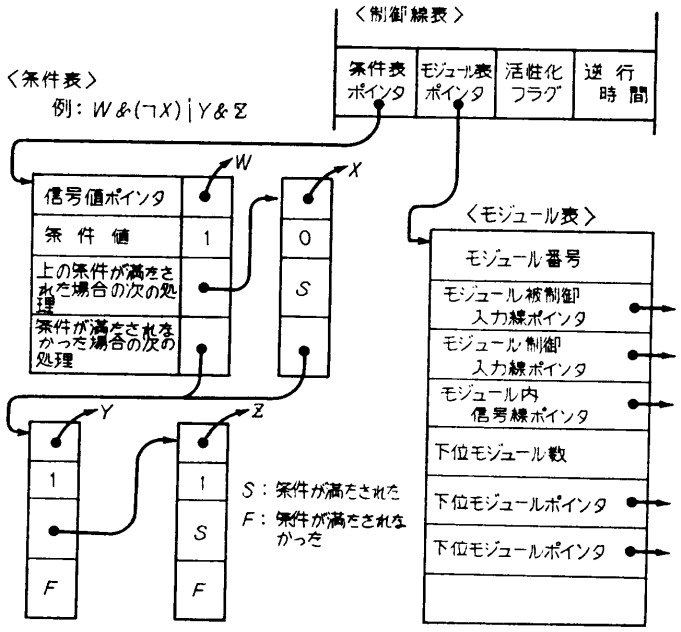


図 8 意味記述法のための表
Fig. 8 Tables for processing semantics description.

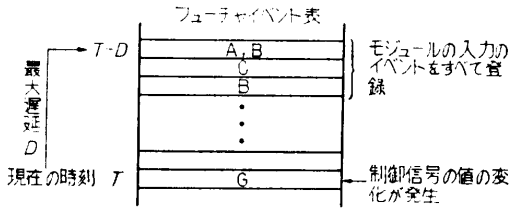


図 9 シミュレーション時刻の逆行
Fig. 9 Retracing of simulation step.

を判定し，そのモジュールが新たに活性化されたとき時刻をさかのぼってシミュレーションが行われる．制御信号変化時以後の出力の変化には，このモジュールの最大遅延で規定される，以前の入力変化も考慮する必要があるためである．そしてモジュール表に基づき，モジュールの被制御入力線のポインタをたどってそれらの入力線のイベントを登録し(**図 9** 参照)，またそれらの入力線およびモジュール内部の必要な線すべてに＊をつける．一方活性化されていたモジュールが不活性化されたときは，モジュールの被制御入力線の＊をすべて取り去る，モジュールには記憶素子を含むこともあるため，モジュールのシミュレーションは出力変化時のみでなく，記憶素子の状態変化時にも必要である．

4.2.3 意味記述の方法

SIM/D のひとつの特色は論理システムの記述を階層的なモジュール構成で行うことである．各モジュールに対しその構造記述と，必要に応じ意味記述を与える．**図 10** に回路記述の方法の例を示す．

〈MODULE〉は階層構造で記述され，1つの〈MODULE〉は外部入力〈EXIN〉，外部出力〈EXOUT〉，意味記述〈SEMANTICS〉，構造記述〈STRUCTURE〉に分かれる．〈EXIN〉，〈EXOUT〉は〈STRUCTURE〉中の〈INPUT〉，〈OUTPUT〉と対応がつけられる．

〈SEMANTICS〉では，〈CONTROL〉においてこのモジュールの制御入力線の集合を宣言し，〈CONDITION〉においてモジュールが活性化される入力条件を積和形で表現する．かっこの中にさかのぼるべき時間を記入する．

〈STRUCTURE〉における構造記述は，ゲート／機能レベルかRTレベルかによって，それぞれ SIM/G, SIM/R にひき渡される．

5. SIM/R の構成と方式

5.1 SIM/R の構成

ゲート／機能レベルの SIM/G のみでは，複雑な LSI を含む論理システムをシミュレーションの対象とするのは実際上困難であり，RT レベルで LSI を記述し，シミュレートすることにした．LSI の記述は RT レベルでライブラリ化することができる．またユーザはその他のモジュールも RT レベルで記述することができ詳細な論理設計の完了していないモジュールとの結合シミュレーションも可能である．あるいは SIM/G によりすでに検証されたモジュールを RT レベルに書き直して SIM/R でシミュレートし，高速化を図るこ

```
〈MODULE〉 CPU:
    .....
  〈MODULE〉 ADDER:
    〈EXIN〉 A [1:4], B [1:4], D, E
    〈EXOUT〉 C [1:4]
    〈SEMANTICS〉
      〈CONTROL〉 D, E
      〈CONDITION〉 ¬D&E | D&¬E (5)
    〈STRUCTURE〉
      〈INPUT〉 PIN [1:8], PIN [13], PIN [14]
      〈OUTPUT〉 PIN [9:12]
          .....
  〈END〉 ADDER
    .....
〈END〉 CPU
```

図 10 回路記述の方法
Fig. 10 Method of circuit description.

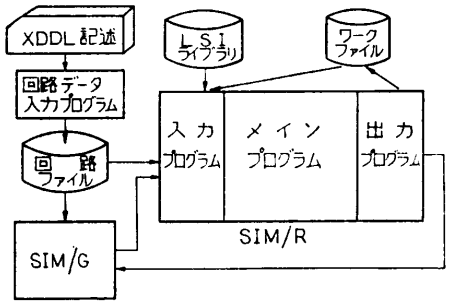


図 11 SIM/R の構成
Fig. 11 SIM/R.

ともできる.

図 11 に SIM/R の構成の概略を示す. SIM/R の特徴としては次の 2 点がある.

(1) 従来の RT レベルシミュレータでは余り考慮されていなかった時間要素の取り扱いを可能にしたこと.

(2) 意味記述を導入し, シミュレーションの高速化を図ったこと. 特に RT レベルではひとまとまりの意味を持つ複数の信号線をまとめて並列にシミュレートできる.

SIM/R ではこれらを実現するために, ハードウェア記述言語 DDL (Digital System Design Language)[13] を拡張し, XDDL (Extended DDL) と名づけ, これで回路記述を行う.

5.2 ハードウェア記述言語 XDDL

XDDL では前章図 10 のようにゲート／機能レベルとできるだけ記述を共通にし, 〈STRUCTURE〉 部に XDDL 特有の記法を用いる. DDL 同様に, 対象回路が複数のオートマトンと組合せ回路によりなりたつとして, 回路の動作記述をオートマトンの状態遷移により行っている. DDL に基づくハードウェア記述言語にした理由は, DDL は回路の構造の記述能力が他の言語に比較して高く, LSI の構造をかなり忠実に表現できると考えたからである.

遅延時間については, 端子の宣言部および 〈AUTOMATON〉 部の各状態におけるレジスタ転送, 端子の接続, 状態遷移等のオペレーションに対し指定することができる. 第 7 章に XDDL による回路記述例を示す.

後述するゲートレベルとの結合シミュレーションにおいて, SIM/R では複数の信号線をまとめて並列にシミュレートする. まとめうる信号線は, 〈INPUT〉 および 〈OUTPUT〉 の宣言部で配列で表現する.

5.3 SIM/R の方式

SIM/R はテーブルドリブン法により 4 値シミュレーションを行う. 図 11 において回路ファイルおよび LSI ライブラリには各種モジュールの XDDL 記述が, 入力プログラムで変換された表の形で入っている. シミュレーションはこれらの表を読込んで翻訳しながら行われる. SIM/R の順序制御は XDDL の状態遷移表現に基づいており, シミュレーションは時刻を単位時間ずつ進めながら, 各時刻で端子の値の計算とオートマトンの状態に応じたオペレーションの実行により行われる.

SIM/R ではレジスタ, データ線等ひとまとまりで扱われる信号値は, 複数ビットを 1 語にまとめて並列的に取り扱う, SIM/G における信号値の表現との相互変換は信号値の受け渡し時に SIM/R 側で行う.

6. 結合シミュレーション

XDDL で記述された LSI あるいはモジュールは, SIM/G からは 1 つのブラックボックスと見なされる. 一般に LSI では信号の経路により遅延時間がかなり異なる. 正しいシミュレーションを行うには, その時点の入力値だけでは必ずしも十分とは言えず, 最大遅延時間から最小遅延時間までの入力値を参照する必要がある. SIM/D では下記のように SIM/R へ $T1$ から $T2$ の間の入力値を伝えることによりシミュレーションを行っている.

CALL SIM-R (MOD#, $T1$, $T2$, $T3$,
INPUT, OUTPUT);

MOD#: モジュール番号
$T1$: 入力値バッファの最初の時刻
$T2$: 入力値バッファの最後の時刻
$T3$: 出力値変化時刻 (現在時刻)
INPUT: 入力値バッファ
OUTPUT: 出力値計算結果

ひとつの入力の変化により複数の出力が異なる時刻に変化することもありうる. これらはフューチャイベント表に複数のイベントとして登録される.

SIM/R は SIM/G から呼び出されると以下のようにしてシミュレーションを行う.

(1) 回路データを回路ファイルあるいは LSI ライブラリから読み込む.

(2) そのモジュールの前回のシミュレーション終了時の値 (初回のときは設定された初期値) をワークファイルからロードする.

(3) 時刻 $T1$ から $T2$ までのシミュレーションを

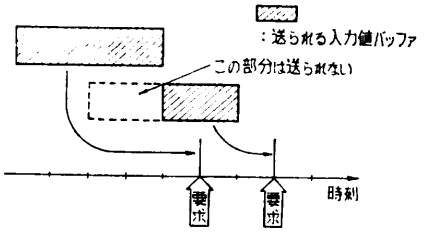


図 12 結合シミュレーションにおける入力バッファの受け渡し
Fig. 12 Method of delivering the input buffer in combined simulation.

行う．

SIM/G と SIM/R の結合によるオーバヘッドを小さくするため，SIM/R ではシミュレーション終了後も回路データやシミュレーション結果は主記憶内に残され，連続して同じモジュールのシミュレーションが要求された場合にはファイルへのアクセスはしない．また，**図 12** のように要求が短時間に連続して発生した場合には，重複する部分の入力値は SIM/R には送られず，処理を重複して行うことはない．

セレクティブイベント法および意味記述の利用によるイベント数の減少は，RT レベルのモジュールに対し非常に効果的になりうる．

7. SIM/D の使用例

SIM/Dは現在京都大学大型計算機センターFACOM M-200 上で稼動している．2入力ゲート素子のみからなる回路を対象とする場合には，1素子あたり約57バイトを要し，フューチャイベント表，プログラム領域を合わせても，利用可能なメモリ容量6Mバイトを最大限に利用すれば約10万ゲートまでシミュレーション可能である．意味記述を利用すると過去の信号値の蓄積が必要であり，1つの信号に1バイトを用いるので，5万ゲートのシミュレーションで約50単位時間の逆行が可能である．

```
〈MODULE〉 AM 2910:
  〈SEMANTICS〉
    〈CONTROL〉 CP, NOE
    〈CONDITION〉 CP | ～NOE (64)
  〈STRUCTURE〉
    〈INPUT〉 D [1:12], I [1:4], CCL, CCENL (6),
             CI, RLDL, OEL (15), CP
    〈OUTPUT〉 Y [1:12] (14), FULLL, PLL, MAPL,
              VECTL
    〈TERMINAL〉 JZ, PASS, PUSHS, POPS, CLEARS,
        .....
    〈BOOLEAN〉
      JZ=(I=0),
      PASS=(CCENL=1) | (CCL=0),
      PUSHS=PUSH | JSRP | (PASS&CJS),
          .....
    〈AUTOMATON〉 MICROPC: CP:
      〈REGISTER〉 MPC [1:12]
      〈STATE〉
        S1: | CI | MPC←MULTI+1;
            MPC←(48) MULTI., →S1..
```

図 13 Am 2910 の記述
Fig. 13 Description of AM 2910.

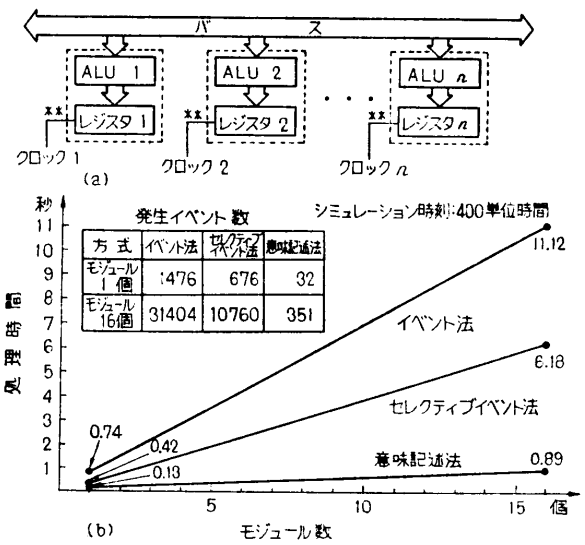


図 14 高速化方式の効果
Fig. 14 Effect of speed-up methods.

SIM/D は研究室におけるハードウェアの開発，論理設計，診断理論の研究などに利用されている．XDDL を用いることにより，LSI を簡潔に記述できる．たとえば，AMD 社の Am 2910 を記述すると，約 80 行で記述できる．その一部を XDDL の記述例として**図 13** に示す．図中（ ）の中は遅延時間を表わす．

以下にイベント法，セレクティブイベント法および意味記述法の3つの方法の処理速度の比較の一例を示す．対象とした回路構成を**図 14**（a）に示す．この回路は制御信号のあるモジュールがバス結合された例であり，意味記述利用による高速化の効果がよくわかる．図 14（b）に示すように，モジュール数が 16 個の場合にはイベント法と比べ1桁以上の処理時間の高速化が実現される．なお，ここで意味記述法はセレクティブイベント法を併用している．また，ALU がなくレジスタが直接バスに接続されている回路においては，レジスタをひとつの機能素子とみなせるため，セレクティブイベント法利用のみで大幅な処理の高速化が実現される．レジスタ数が 16 個の場合には，従来のイベント法 (1,760 ms) に対し，セレクティブイベント法 (400 ms) では 4 倍以上の高速化となっている．セ

レクティブイベント法は回路の構造記述のみで高速化が達成できるので，使用が容易である．なお，この他の処理時間のデータや出力例のいくつかを文献 9) に示した．

8. あとがき

本論文ではLSIを含む論理システムのシミュレーションに適した論理シミュレータ SIM/D について述べた．SIM/D はゲートレベル，機能レベル，RT レベルを併用したシミュレーションが可能である．RT レベルにおいても遅延時間の記述が可能な XDDL により回路の記述を行うので，全体としてのシミュレーションはほとんどゲートレベルの精密さで，RT レベル併用の高速性も有する．

またシミュレーションの高速化のため，新たにセレクティブイベント法と意味記述を導入し，従来のイベント法に比べて回路のシミュレーション時間を大幅に短縮することができた．意味記述を利用すればモジュールの階層的記述をうまく利用することにより最大限に効果が期待できる．また意味記述は，システムの自動検証において本質的な役割を果たすと考えられ，今後意味記述の方法を深化させこの方向へも研究を発展させてゆく予定である．

意味記述の誤りは，制御線，被制御線の指定あるいはその論理，遅延時間の誤りが代表的である．これらはシステムの構造記述と意味記述の無矛盾性の検証として扱うことができ，モジュール単位の局所的な検証のみでほとんどが処理できるので，比較的容易と考える．現在そのプログラム化を研究しつつある．

最近，論理システムの開発の初期からのシミュレーション，検証，設計自動化を一貫して行う設計・検証ソフトウェアシステムが研究されている[14]．このような総合的な開発支援システムは，今後論理設計データベースを中心として構築されてゆくものと考える．SIM/D は論理システムの設計自動化の研究の一環として開発を始めたものであり，今後論理設計データベースの研究を軸として，総合的なシステムへと発展させてゆきたい．

謝辞 種々貴重なご助言，ご討論いただいた上林彌彦助教授はじめ矢島研究室の諸氏に深謝する．プログラムおよびデータの作成には，後藤浩一氏，石黒俊太郎氏，塚本勝氏，吉川正俊氏のご協力をいただいた．なお本研究は一部文部省科学研究費および作行会研究助成金による．

参考文献

1) Szygenda, S. A. and Lekkos, A. A.: Integrated Techniques for Functional and Gate-Level Digital Logic Simulation, Proc. 10th Design Automation Workshop, pp. 159-172 (1973).
2) Chappell, S. G., Elmendorf, C. H. and Schmidt, L. D.: Lamp: Logic-Circuit Simulators, Bell Sys. Tech. J., Vol. 53, pp. 1451-1476 (1974).
3) 黒部，根本，志方，可児：LSI 論理シミュレーションシステム；LOGOS2，情報処理設計自動化研資，DA 31-2 (1977).
4) Szygenda, S. A. and Thompson, E. W.: Digital Logic Simulation in a Time-Based, Table-Driven Environment —Part 1. Design Verification, IEEE Computer, Vol. 8, pp. 24-36 (1975).
5) Chappell, S. G., Menon, P. R., Pellegrin, J. F. and Schowe, A. M.: Functional Simulation in the Lamp System, J. Design Automation & Fault Tolerant Computing, Vol. 3, pp. 203-215 (1977).
6) Tokoro, M., Sato, M., Ishigami, M., Tamura, E., Ishimitsu, T. and Ohara, H.: A Module Level Simulation Technique for Systems Composed of LSI's and MSI's, Proc. 15th Design Automation Conf., pp. 418-427 (1978).
7) Hill, D. and vanCleemput, W.: SABLE: A Tool for Generating Structured, Multi-Level Simulations, Proc. 16th Design Automation Conf., pp. 272-279 (1979).
8) 稲垣，槌田，矢島：論理シミュレータの高速化について，電気関係学会関西連大，G6-5 (1979).
9) 稲垣，槌田，酒井，矢島：LSI を含んだ論理システムに適した論理シミュレータ SIM/D，信学技報，EC 79-73 (1980).
10) 稲垣，後藤，石黒，矢島，岩間：計算機結合へのマイクロプログラム制御インタフェースの応用とその開発・保守支援手法，電気学会情報処理研資，IP-79-46 (1979).
11) Ulrich, E. G.: Exclusive Simulation of Activity in Digital Networks, C. ACM, Vol. 12, pp. 102-110 (1969).
12) Hsieh, H. Y. and Rabbat, N. B.: Macrosimulation with Quasi-General Symbolic FET Macromodel and Functional Latency, Proc. 16th Design Automation Conf., pp. 229-234 (1979).
13) Duley, J. R. and Dietmeyer, D. L.: A Digital System Design Language (DDL), IEEE Trans. Computers, Vol. C-17, pp. 850-861 (1968).
14) Kawato, N., Saito, T., Maruyama, F. and Uehara, T.: Design and Verification of Large-Scale Computers by Using DDL, Proc. 16th Design Automation Conf., pp. 360-366 (1979).

(昭和55年3月17日受付)
(昭和55年5月15日採録)