

解説

ゲート・レベル論理シミュレーション†



村 井 真 一†

1. はじめに

論理回路の設計検証の手段として、論理シミュレーション、すなわち既製の計算機（ホスト計算機）上に対象回路のモデルを構築し、その論理動作を模擬することは古くから行われてきた^{1), 2)}。シミュレーションが用いられるのは

1) 実物の製作以前の段階で、早期に設計検証を行えるため、開発期間の短縮が期待できる。

2) 実物では実現困難な回路状態（たとえば故障状態）を容易に実現できる。
等の理由による。

しかしながら論理回路を構成する回路素子が SSI（小規模集積回路：数ゲート）や MSI（中規模集積回路：数十ゲート）であった間は、論理シミュレーションは補助的な役割を果たすにとどまっていた。その理由としては、後述のごときシミュレータ自体の限界もさきながら、装置が SSI/MSI で構成されている場合は再製作が比較的容易であり、前もって設計検証が行われていなくても、重大な損失を被るには至らなかったことが挙げられる。

しかるに LSI（大規模集積回路：数百～数千ゲート）、さらには VLSI（特大規模集積回路：一万ゲート以上）が実用化されるにおよび、

1) 装置の再製作すなわち LSI/VLSI の再製作は、現状では高価かつ長期間を要する* 一方、設計誤りは、プログラム開発に際して我々が経験するごとく、一つの誤りを修正しなければ他の誤りが顕在化しないことがしばしばあり、たび重なる再製作の原因となる。すなわち装置を製作してからその装置を使って設計検証を行うということは、現実には不可能である。

2) Breadboarding, すなわち SSI/MSI を利用して装置のプロトタイプを試作し、その上で設計検証を行うことは、これまで広く行われてきたが、VLSI で実現できる回路規模が大きくなり、プロトタイプの試作が高価かつ長期間を要するようになってきたこと、また最終製品である LSI/VLSI と同一仕様（特にタイミング仕様）のプロトタイプの試作が不可能なこと等により、この方法も適用不可能である。

等の事態を招来し、実際の装置の製作に先だって設計検証を十分に行っておくことが不可欠となった。

以上のごとき環境条件に加え、後述の、シミュレータ自体における模擬技術の向上により、論理シミュレータはいまや論理設計のための最も重要な設計支援ツールの一つとなっている。

本稿ではまず論理シミュレータに課された三つの主要課題について概説し、つづいてシミュレータの基本概念・手法について、機能（モデリング）、実現方式の両面から順次解説する。

2. 論理シミュレータの課題

上述したごとく、設計検証ツールとしての論理シミュレータは、長い間補助的役割に甘んじてきた。これには前述のごとき環境条件によるところ大であるが、それとともに

1) 論理シミュレータの精度に問題があり、論理シミュレータ上では正常に動作しても、現実の装置は正常に動作しない、あるいはその逆の事態が発生した。

2) 論理シミュレータで取り扱うことのできる回路規模に制限があり、特に入出力装置を多数接続した計算機のごとき、大規模装置全体の動作を模擬することは不可能であった。

3) たとえ大規模回路を模擬することが機能的に可能であっても、模擬速度が遅く、禁止的に高価な計算機時間を必要とした。

等が主要な原因であった。

論理シミュレータ発達の歴史はこれら三つの主要課

† Gate Level Logic Simulation by Shinichi MURAI (Computer & Software Department, Information Systems & Electronics Laboratories, Mitsubishi Electric Corporation).

†† 三菱電機(株)情報電子研究所情報処理開発部

* 電子ビーム露光装置による LSI/VLSI チップの直接製作が一般化すれば、事態が変わることも予想される。

題、すなわち 1) 模擬可能回路種類および規模の拡大、2) 模擬速度の向上、3) 模擬精度の向上を実現するための手法の考案・実施の歴史であるとも言えよう。

3. モデリングの問題

3.1 模擬単位

論理シミュレータは、対象回路をどのような単位で模擬するかに応じて、表-1 に示すごとく三つのレベルがある。

(1) ゲート・レベル論理シミュレータ

文字通り、AND、OR、NOT 等論理ゲートを模擬単位とするもので、単位素子の出力端子数はすべて1である。

(2) 機能シミュレータ

ラッチ、フリップ・フロップ等からカウンタ・レジスタ等にいたる機能素子を模擬単位とするもので、単位素子の出力端子数は1とは限らない。

(3) RTL シミュレータ

シミュレータのユーザが、DDL¹⁸⁾ 等のごとき、回路の動作を記述する機能を有する記述言語、すなわちレジスタ転送言語 (RTL: Register Transfer Language)¹⁷⁾ を使用して、対象回路の機能を自由に定義できるシミュレータで、これも含めて機能シミュレータと呼ばれることも多い。

本稿で機能シミュレータと呼んでいるものは、シミュレータにライブラリとしてもともと用意されている機能素子しか模擬できないのに対して、RTL シミュレータの場合にはユーザが自由に定義できるところが大きな違いであり、必要に応じてたとえば入出力装置のごとき大きな機能ブロックまで含めて取り扱うことが可能となる。またユーザから見た機能のみでなく、シミュレータの実現方式から見ても、RTL シミュレータと機能シミュレータとの間には大きなギャップがある。一方、ゲート・レベル・シミュレータと機能シミュレータとの間には、実現方式上さほどの相異はない。

(4) 実際のシミュレータ

さて現実に使用されているシミュレータは、上記各レベルが必要に応じて混在

表-1 模擬単位にもとづくシミュレータの分類

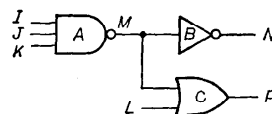
種 類	模擬単位	信号値
回路シミュレータ	トランジスタ	電圧/電流
論理シミュレータ	ゲート・レベル論理シミュレータ	ゲート
	機能シミュレータ	機能素子
	RTL シミュレータ	RTL (機能ブロック)
方式シミュレータ	命 令	ベクトル

しており、特に今日では純粋のゲート・レベル・シミュレータが使われることはまれである。また一般にゲート・レベル論理シミュレータと呼ばれているシミュレータでも、通常少なくともフリップ・フロップ程度の機能素子は直接取り扱えるようになっている。

以下本稿では、ゲート・レベル論理シミュレータで採用されている手法を中心に説明するが、これらはそのまま機能シミュレータでも採用されている手法である。

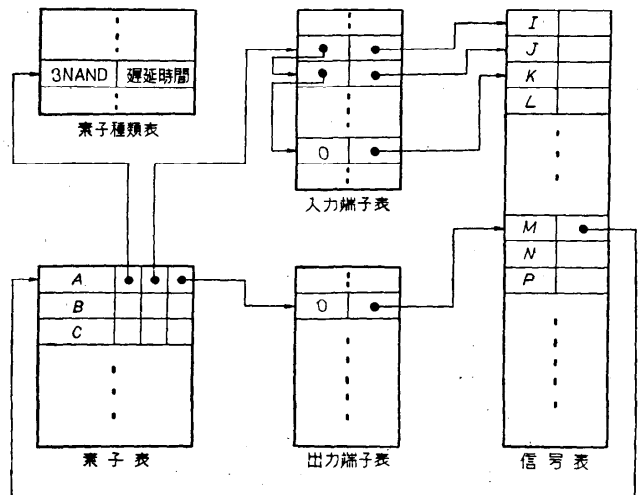
3.2 回路記述

論理シミュレータを用いて論理回路の動作解析を行うためには、まず対象回路の仕様を記述してシミュレータに入力する必要がある。初期のゲート・レベル



(a) 回路例

(b) ネットワーク記述



(c) 表モデル

図-1 回路記述と内部モデル

論理シミュレータではブール式による記述も用いられたが、機能素子まで含めて扱える近年の論理シミュレータにおいては、ネットワーク記述すなわちそのシミュレータで取り扱い可能な素子間の接続状態の記述を用いることが最も一般的である。図-1 (a)の回路のネットワーク記述の一例を 図-1 (b)に示す。

3.3 信号値

シミュレータで取り扱われる信号値には、模擬対象の回路自体が本質的に持っている状態値 (0, 1等)のほかに、シミュレータの模擬方式の限界等から発生する状態値 (不定状態 X 等)が存在する。

(1) 実回路固有の信号値

回路自体が本来持っている状態値はさらに定常状態と非定常状態に分けることができる。定常状態には通常の0, 1のほかに、3値素子が含まれる場合、 Z (高インピーダンス状態)が加わる。非定常状態には P (パルスまたはスパイク状態) および S (発振状態)がある。

(2) 模擬回路のみの信号値

一方実回路には存在せず、シミュレータ内部の模擬回路上でのみ存在する状態としては、安定状態 X (不知状態) および過渡状態 U ($0 \rightarrow 1$ すなわち 0 から 1 への遷移状態), D ($1 \rightarrow 0$), T ($0 \rightleftharpoons Z$, $1 \rightleftharpoons Z$) 等が考えられている。 X の発生原因には、(1)初期状態、(2)禁止/不定義入力、等がある。初期状態は、模擬開始時期に模擬回路の内部状態として発生する。禁止/不定義入力とは、たとえば R/S フリップ・フロップの J, K 端子に同時に1が入力されるような場合で、そのフリップ・フロップの出力端子の状態は X となる。禁止入力、ハザード、レース、慣性遅延 (次章参照)以下の長さの短いパルス等が原因で発生する X を特に区別して E (エラー) とする場合もある。

過渡状態 U, D 等は、次章で述べる最大最小遅延モデルにともなって発生するもので、素子の信号値がたとえば0から1に変化する場合、その素子の0から1への変化に要する遅延時間の最小値から最大値までの期間その信号値を U とする。

(3) 実 現 例

現実のシミュレータはその目的に応じてこれらの信号値のサブセットを取り扱う。最も簡単な初期のシミュレータは0および1だけしか扱えず2値シミュレータと呼ばれる。2値シミュレータでは模擬開始時期に模擬回路の外部端子のみならず、その内部信号まですべて0, 1いずれかの値に設定しなければならない。ま

た前述のごとき不定義入力に対しても、その時の出力値を0, 1いずれかに決めてしまう必要があり、模擬回路の動作が実回路のそれと食い違ってくる恐れがあった。この欠点を補うため X を導入した3値シミュレータが考案され⁴⁾、さらに模擬精度を向上させる目的で、上述の各状態値を要求精度に応じて取り入れた多値シミュレータが開発されている⁵⁾。特に VLSI の実用化によってますます重要な問題の一つとなっているパッケージの端子数制限に対処する手法の一つとして、3値論理採用による端子の多目的使用が広く行われるようになり、高インピーダンス状態 Z を取り扱うシミュレータが主流となりつつある。

3.4 遅延の取扱い

模擬回路動作と実回路動作の相異の主要な要因は、回路動作のタイミングを正確に模擬できない、すなわち回路素子の遅延時間の取扱いが不正確ということであった。以下に各種の遅延モデルとその特徴・限界について概説する。

3.4.1 遅延の種類

(1) 0遅延モデル⁶⁾

各素子は遅延を持たないとして模擬する方式で、実回路との相異がはなはだしいのみならず、非同期回路は扱えないという制限がつく。初期のコード生成方式のシミュレータでは原理上このモデルしか扱えなかったために用いられた。

(2) 単位遅延モデル⁶⁾

回路中のすべての素子は同一単位時間の遅延を持つとして模擬する方式で、タイミングの正確な模擬という観点からは0遅延モデルと大差ないモデルであるが、原理的に非同期回路も取り扱えるところが大きな違いである。

(3) 標準遅延モデル

公称遅延モデルあるいは割当遅延モデルとも呼ばれるモデルで、必要とされるタイミング精度に応じて、AND, OR 等の素子種類ごとにはじまって、個別素子ごとに至るまでの遅延を前もって割り当てておくもので、現在最も広く実用されている方式である。

(4) 最大-最小遅延モデル⁹⁾

標準遅延モデルよりもさらに正確なタイミング解析を意図して考案されたモデルで、領域遅延、あいまい遅延とも呼ばれる。その名の示すごとく、素子に割り付ける遅延時間に幅を持たせる方式で、たとえば 図-2のごとく、入力変化に応じて出力に変化が生じるに際して、その素子の起こりうる最小遅延時間から最大遅

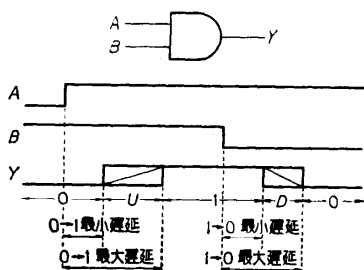


図-2 最大-最小遅延モデル

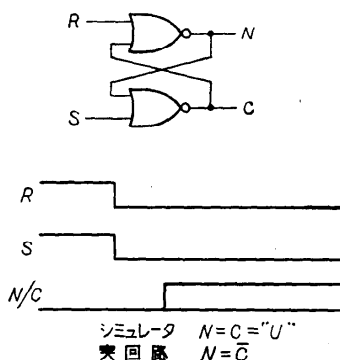


図-3 最大-最小遅延モデルに基づく情報喪失

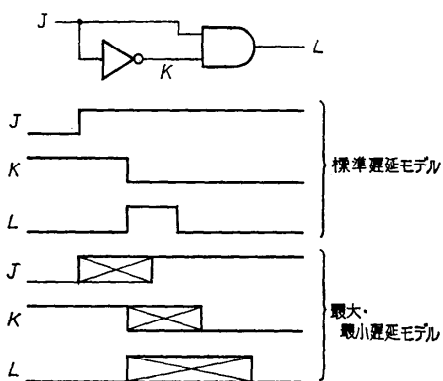


図-4 微分回路と遅延モデル

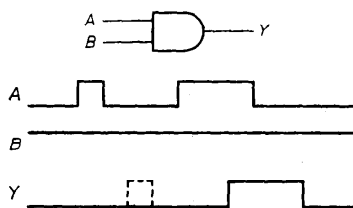


図-5 慣性遅延

延時間までの期間その素子の出力値を不知状態 X 、または遷移状態 U 、 D 等とする。

本モデルにより実回路で問題となりうるタイミング上の問題はほとんど模擬回路上でも発見できることとなったが、(1)単位遅延または標準遅延モデルにくらべ多大の模擬時間を要するため、実質的には大規模回路に適用困難である、(2)実回路中の素子の遅延時間は、ある特定の素子についてみればその時々で変化することはほとんどなく一定であるにもかかわらず、本モデルではあたかも常に幅があるかのごとく仮定しており、模擬回路の方が実回路に比しあまりに悲観的な動作を行う、(3)ラッチ回路や微分回路等日常よく使われる回路の模擬に問題がある (図-3、4)^{10), 11)}、等の重大な問題点があり、小規模回路を対象とした、詳細なタイミング解析を必要とする場合を除き、あまり実用されていない。

(5) 慣性遅延モデル¹²⁾

実回路が反応しないような極めて短いパルスまたはスパイクに対しても、これまでの遅延モデルでは応答してしまうのに対して、慣性遅延モデルでは、各素子が応答するのに必要な最小パルス間隔を指定できるようになっている (図-5)。

3.4.2 割当方法

以上、論理シミュレータで取り扱われる5種類の遅延について説明したが、次に、これらの遅延をどのような方法で割り当てるかについて簡単に触れる。

A. 割当場所

(1) 素子単位

最も単純かつ通常行われている割当方法で、一つの素子に1種類の遅延を割り当てる。特に多入出力端子素子であっても、全端子に同一遅延時間を割り当てる。この時、デバイス・テクノロジーごと (バイポーラ、MOS 等)、素子種類ごと (AND、OR 等)、あるいは個別素子ごとに異なる遅延時間を与えることができる。

(2) 端子単位

多入出力端子素子の場合、端子単位に遅延を割り当てる。この時出力端子のみに割り当てるか、入力端子のみに割り当てるか、あるいは入出力全端子に割り当てる。さらに素子に応じた値のみでなく、各端子に接続されている配線長に応じて異なる遅延時間を割り当てることにより、実回路動作に近い模擬を行うことができる。

(3) 端子の組み合わせ単位

割当法をさらに詳細化して、各入力端子の組み合わせ単位に異なる遅延時間を割り当てることも考えられる。たとえば3入力2出力の素子の場合 $3 \times 2 = 6$ 種類の遅延時間を割り当てる。実際に実現された例はなさそうである。

B. 信 号 値

初期のシミュレータでは信号値に無関係に遅延時間を割り当てたが、近年のシミュレータでは、たとえば $0 \rightarrow 1$ 遷移の場合と $1 \rightarrow 0$ 遷移の場合とで異なる遅延時間を与えられるようになっている。

4. 実 現 方 式

4.1 模 擬 方 式

シミュレータ内で模擬対象回路をどのような形で保持し、どのように模擬をすすめるか、すなわち回路のモデリング法には大別して次の2種類がある。

(1) コード生成方式

対象回路の構成する論理素子群を、それぞれに対応した一連の直接実行可能なサブプログラム群にコンパイルする方式で、コード生成方式あるいはコンパイル方式と呼ばれる。模擬にあたっては、生成されたプログラムを直接実行できるため高速であるが、各素子の遅延時間をモデル化することが困難なため、タイミング解析ができず、また組み合わせ回路あるいは同期回路しか扱えないという致命的弱点があるため、近年のシミュレータにはほとんど採用されなくなっている。

(2) 表生成方式

対象回路を、その接続関係を忠実に表現する表形式のデータ構造に変換しておき、実行時にそれをたどりながら解釈する方式で、表生成法、表駆動法または解釈実行法等と呼ばれる方式である。各論理素子の動作を模擬すること、これらの表を解釈するので、そのオーバーヘッドは大きい、(1)遅延時間の挿入が容易で、非同期回路でも支障なく模擬できる、(2)回路の部分修正が容易、(3)動作中の回路部分のみ模擬すること(選択追跡)が容易で高速化が可能、等得がたい利点があり、近年のシミュレータはほとんどがこの手法を採用している。図-1(a)の回路の表記の一例を図-1(c)に示す。

4.2 順序制御方式

実際の論理回路においては状態の変化は並列に進行するが、シミュレータの中では各信号値の変化を一つずつ順番に処理しなければならない。このための制御方式、すなわち順序制御方式としては次の種類が代表

的である。

(1) レベル・ソート方式⁹⁾

対象回路中の素子をあらかじめ信号の伝搬する順序にソートしておき、入力変化のあるごとに(たとえばクロックごと)その順序に従って信号値を計算してゆく方式で、組み合わせ回路や同期回路には適用できるが、ループの存在や、素子ごとの遅延時間の影響により一義的レベル付けの困難な非同期回路の取扱いは不可能である。初期のコード生成方式シミュレータで使用されたが、近年の表駆動方式シミュレータでは使用されていない。

(2) 事象リスト方式(選択追跡方式)⁶⁾

リスト処理技術と動的メモリ割当技術を利用して、模擬中の回路のある素子の状態が変化した時これを事象としてとらえ、その出力先の素子を事象リスト中の適当な箇所に挿入し、一方、各素子の論理動作の計算は、この事象リストから順次取り出して行う方式で、各事象に対して任意の時刻を与えることができる。すなわち素子の遅延時間を自由に設定できるために、非同期回路の模擬を自由に行うことができる、また状態変化のおよぶ素子のみを選択的に模擬できる(選択追跡)ため高速化できる等の利点がある。

事象リストは図-6に示すごとく、事象の生起順にポインタでつながれた線型リスト構造を持つ。大規模回路ではこのリストは長大なものとなる場合が多く、事象リストへ新規の事象をその生起時刻に応じて適当な位置に挿入する処理のためのオーバーヘッドは大きくなる。この弱点を補うために考案された手法が次の時刻写像方式である。

(3) 時刻写像方式⁷⁾

時刻写像方式(Time Mapping Technique)は上述の事象リスト方式と同様に事象駆動方式の一種であるが、事象リスト方式の弱点、すなわち大規模並列事象の処理が効率よく行えるよう工夫された方式で、時刻写像すなわち時刻を番地に写像することを基本概念としている。

事象リスト方式においては長大な事象リストが一つ管理されているが、時刻写像方式ではこれを図-7に示すように Δt 単位の時刻に分割して保持する。したがって事象リスト方式では事象を事象リスト中の適当な位置に登録するために長大なリストを探索する必要があったが、時刻写像方式においては当該事象の生起時刻につらなる事象リストに挿入するだけでよく、事象リスト方式にくらべ格段に高速である。

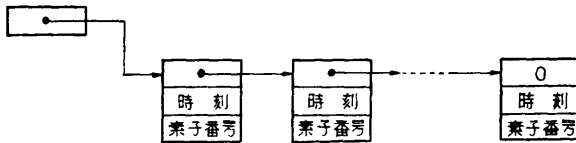


図-6 事象リスト

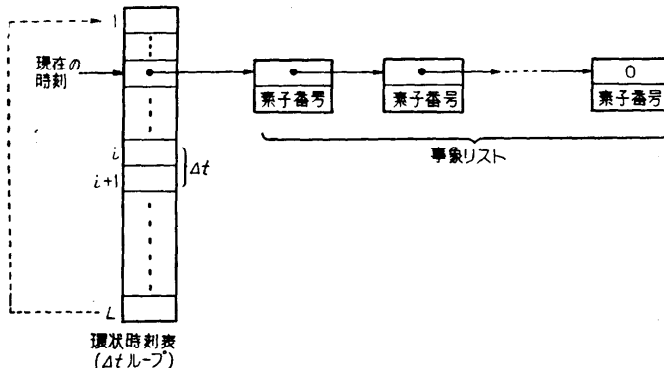
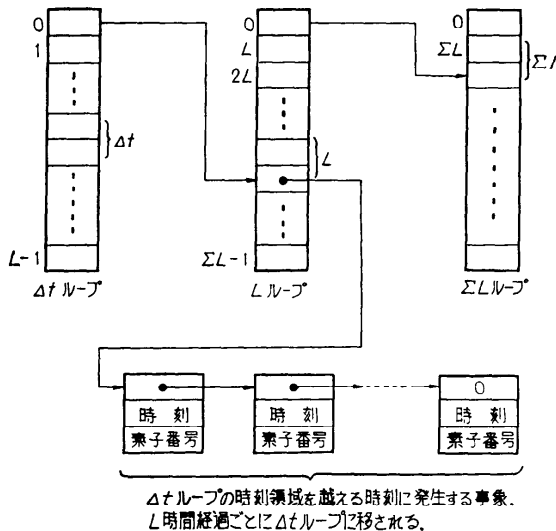


図-7 時刻写像法



Δtループの時刻領域を越える時刻に発生する事象、
L時間経過ごとにΔtループに移される。

図-8 多重時刻写像方式

模擬動作は Δt ループとも呼ばれる環状時刻表を Δt ずつ進みながら、各時刻につらなる事象リスト中の事象を順次実行してゆく。 Δt ループの周期は対象回路中に発生する最大遅延時間に等しくしておけばよい。時刻写像方式においてタイミング精度を上げるためには Δt を短くする必要があるが、周期一定のまま Δt を短くすると Δt ループのための記憶容量が膨

大となる。この欠点を解消するために図-8 のとき多重時刻写像方式⁷⁾ や事象リスト方式との混合方式⁸⁾ も試みられている。

4.3 高速化手法

模擬速度の向上は論理シミュレータはじまって以来の永遠の課題と言えるが、VLSI 時代の到来により一層切実な問題となっている。一つの解決法は言うまでもなく模擬単位の高レベル化(機能シミュレーション)であるが、ここでは本稿の主旨に沿ってゲート・レベル・シミュレータで採用されている代表的な高速化手法について概説する。

(1) 選択追跡法 (Selective Trace Technique)

上述の事象リスト法および時刻写像法で用いられている方法で、入力信号

に変化の発生した素子のみについて模擬を行う。通常の汎用計算機の場合、一時に変化の起こっている素子は全体の 2.5% 程度とも言われているが¹³⁾、このような場合には本手法の採用により 40 程度の高速化を達成できることになる。

(2) 選択事象法 (Selective Event Technique)¹⁴⁾

上述の選択追跡法の考え方を一層発展させたもので、入力信号に変化のある素子が常に出力変化をもたらすとは限らないことに注目し、(たとえば、1 個以上の入力端子が 0 となっている AND ゲート)、このような素子に対しては入力信号に変化があっても事象リストに登録しないようにする方法である。

(3) 意味記述法¹⁴⁾

選択事象法の考え方をさらに発展させ、より複雑な回路に対しても選択的に模擬を実行できるように、その回路の構造記述のみでなく、意味記述、すなわちたとえば図-9 のような回路においては、ゲート信号が 1 の場合だけ模擬を実行すればよいという記述をつけ加えることにより、図-10 のとき高速化が達成されることが報告されている¹⁴⁾。図-9、図-10 において**の付与されている信号には意味記述が付随していることを示す。

(4) 表参照法 (Table Look-up Technique)¹⁵⁾

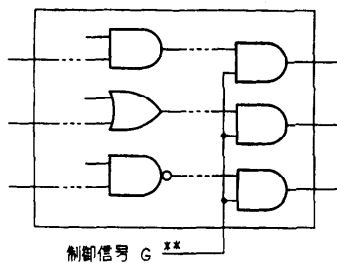
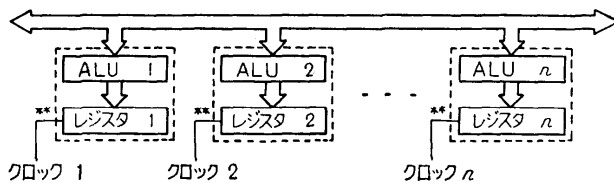


図-9 意味記述が効果的な回路の一例



(a) 対象回路

方 式	選 択 追 跡	選 択 事 象	意 味 記 述
1モジュール	1,476	676	32
16モジュール	31,404	10,760	351

(b) 発生事象数

図-10 高速化方式の効果

論理素子または回路の演算を行う際、ホスト計算機の演算命令を組み合わせて所定の機能を模擬するのではなく、対象素子／回路への入力信号の可能なあらゆる組み合わせに対する出力結果をあらかじめ表の形で記憶しておき、模擬時の入力信号値に応じてこれを参照する方式で、LSI/VLSI 化の進展に応じて安価となった主記憶をフルに活用しようという手法である。

(5) 並 列 法

計算機の1語に対する論理演算は1ビット単位で独立(並列)に行われることに着目して高速化を図る方式で、ホスト計算機の1語中の各ビットに、それぞれ異なる入力系列に対応する模擬信号値を割り当てる。たとえば1語32ビットの計算機をホストにする場合、32種類の異なる入出力系列を並列に模擬できる。

5. む す び

以上、ゲート・レベル論理シミュレータの基本的な手法について概説した、これらの手法は、論理シミュレータの古くからの課題、すなわち模擬可能回路種類・規模の拡大、模擬速度の向上および模擬精度の向上を

達成する上で重要な貢献をして来た。しかしながらVLSI 化の進展に伴ってこれらの課題自体もそのレベルが一段と押し上げられ、なお一層の対処・工夫が必要とされることに変わりはない状況にある。

かかる事態の対処法として、ゲート・レベル論理シミュレータの枠内でのみ解決しようと試みることは得策ではなく、また現実に行われてもいない。むしろ最近の傾向は、大規模化・高速化には機能シミュレータで、また高精度化に対してはタイミング・シミュレータで対応しようとするようになっており、特に一つのシミュレータが必要に応じて機能シミュレーション、ゲート・シミュレーションからタイミング・シミュレーション、さらには回路シミュレーションまで自由に実行できるようにしようとの試みがなされはじめている¹⁶⁾。すなわちより総合的な見地からの上記課題へのアプローチが重要となっている。

以上の課題におとらず重要であり、かつこれまでのところ解決されていない課題として、模擬回路への入力信号系列の選定の問題がある。プログラム・デバッグの経験からも容易に類推できるように、これは極めて解決困難な問題であり、これまでのところもっぱら設計者の経験に頼っているだけというのが実状であるが、ソフトウェアの世界で試みられている手法、すなわち仕様記述の明確化、構造化設計、階層化設計等と結びついた形での解決法が有力な方向であるように思われる。前章で述べた意味記述の導入も、単に高速化手法にとどまらず、この問題解決のための一つの手法へと発展するかもしれない。これらと結びついた総合的設計支援システムの構築が今後の課題であろう。

参 考 文 献

- 1) Connolly, T. A. : Automatic System and Logical Design Techniques for the RW-33 Computer System, 1960 IRE Conv. Rec., pt. 2, pp. 124-132.
- 2) 高島他: 論理構成のシミュレーション・プログラム, 情報処理, Vol. 4, No. 2, pp. 64-72 (1963).
- 3) Seshu, S. and Freeman, D. N. : The Diagnosis of Asynchronous Sequential Switching Systems, IEEE Trans. Electronic Computers, Vol. EC-11, No. 8, pp. 459-465 (1962).

- 4) Hays, G. G. : Computer-aided Design : Simulation of Digital Design Logic, IEEE Trans. Computers, Vol. C-18, No. 1, pp. 1-10 (1969).
- 5) Thompson, E. W. and Szygenda, S. A. : Three Levels of Accuracy for the Simulation of Different Fault Types in Digital Systems, Proc. 12th Design Automation Conf., pp. 105-113 (1975).
- 6) Evans II, G. W., Wallace, G. F. and Sutherland, G. L. : Simulation Using Digital Computers, Prentice-Hall Inc., Englewood Cliffs, NJ (1967).
- 7) Ulrich, E. G. : Serial/Parallel Event Scheduling for the Simulation of Large Systems, Proc. 1968 ACM National Conf., pp. 279-287 (1968).
- 8) Szygenda, S. A. and Thompson, E. W. : Digital Logic Simulation in a Time-Based, Table-Driven Environment-Part I. Design Verification, IEEE Computer. Vol. 8, No. 3, pp. 24-36 (1975).
- 9) Benning, L. C. : Simulation of High Speed Computer Logic, Proc. 6th Design Automation Workshop, pp. 103-112 (1969).
- 10) Breuer, M. A. : A Note on Three-valued Logic Simulation, IEEE Trans. Computers, Vol. 21, No. 4, pp. 399-402 (1972).
- 11) Breuer, M. A. and Friedman, A. D. : Diagnosis and Reliable Design of Digital Systems, Computer Science Press, Woodland Hills, Calif., pp. 174-254 (1976).
- 12) Koford, J. and Walker, R. : FAIRSIM II User's Manual, Fairchild Semiconductor (1969).
- 13) Scheff, B. H. and Young, S. P. : Ch. 3 Gate-Level Logic Simulation, Design Automation of Digital Systems, Vol. 1-Theory and Techniques (Breuer, M. A. Ed.), Prentice-Hall, Englewood Cliffs, NJ, p. 137 (1972).
- 14) 稲垣耕作, 植田義之, 酒井文嗣, 矢島脩三 : 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, 情報処理学会論文誌, Vol. 21, No. 4, pp. 332-339 (1980).
- 15) Ulrich, E. G., Baker, T. and Williams, L. R. : Fault Test Analysis Techniques Based on Simulation, Proc. 9th Design Automation Workshop, pp. 111-115 (1972).
- 16) Agrawal, V. D., Bose, A. K., Kozak, P., Nham, H. N. and Pacas-Skewes, E. : A Mixed Mode Simulator, Proc. 17th Design Automation Conf., pp. 618-625 (1980).
- 17) van Cleeput, W. M. : Computer Hardware Description Languages and their Applications, Proc. 16th Design Automation Conference, pp. 554-560 (1979).
- 18) Duley, J. R. and Dietmeyer, D. L. : A Digital System Design Language (DDL), IEEE Trans. Computers, Vol. C-17, No. 9, pp. 850-861 (1968).

(昭和56年4月21日受付)