MODELS—ミニコンによるMOS/LSI設計のための遅れを考慮したロジック・シミュレータ

花田 正幸、山崎 勇、盛 常樹
（東京芝浦電気株式会社、総合研究所）

1. まえがき

設計は、本来複雑な総合行為であるが設計する対象が大規模で高度なものとなると、いっそう困難な仕事となる。MOS/LSIの開発設計においても回路の論理的誤りがないかを調べることや、チップとしての動作速度が目標値通りになっているかを調べたりすることを、設計者自身の思考と判断に頼ることはほとんど不可能である。従来はBread Boardによる方法や大型計算機を用いて連立微分方程式を解くことにより信号の伝播遅延を求めるシミュレータによる方法で上記項目を調べてきた。

しかし、前者は動作速度と回路素子の大きさとの関係を調べることについては、必ずしも適切な情報を与えないし、後者は対象とする素子の数が多くなると急激に計算時間が増大し、しかも一般的に大型計算機を専有できないところからturn aroundが非常に長くなるという欠点があった。

そこで我々は、MOSトランジスタの大きさと配線の浮遊容量を回路定数として与えることで各素子の伝播遅延を求め、その遅れを含めて汎用シミュレーション言語の様な方法により、シミュレーション速度を上げ、しかもミニコンピュータでシミュレーションが行なえるように簡略化したところのロジックシミュレータ——MODELS（MOS Delay Logic Simulator）を開発した。

このシミュレータへの回路図や回路定数の入力およびシミュレーションの実行手順の入力を対話型としたのでシミュレーションによって得られた結果を検討して行なう修正のfeed backが早くなり、設計期間の短縮が期待できる。

このシミュレータは、通常の論理機能のANDやNORなど以外にPush-Pull-BufferやMOS特有のTranafer Gateを対象とすることができる。又同じ回路を何度も用いる場合、その部分の回路をブロックとしてサブルーチン機能により記述でき、入力の簡単化と共に回路の内部表現に必要な記憶容量の減少をはかっている。

このシミュレータで処理できる回路の規模は利用できる計算機の記憶容量によってきまるが、48Kバイトで約1000ゲート程度のシミュレーションができる。

2. シミュレーションの原理

このシミュレータは、連立微分方程式を次々と数値積分することで各Nodeの伝播遅延を求めてゆく方法ではなく、あらかじめ各Nodeの遅れ時間を求めて内部のリストに記憶しておく方法でシミュレーションを行なっている。すなわちこのシミュレータは、各Nodeの論理値のみを考え、時間の要素は各論理素子の出力側に遅れ要素が入っているものとしてシミュレーションを行なう。各遅れ時間Td（Time delay）は、シミュレーションを行なう前にシミュレータのコンパイラと呼ばれるプログラムによって各NodeのトランジスタのW（巾）の値と各Nodeの配線部分の浮遊容量の値から、非常に簡略化した数式で計算して求められる。

2.1 遅れ時間の計算方法

図1はInverterをStatic E/Dタイプの MOS トランジスタ回路で書いたものである。
この回路で(1)のNodeが"0"から"1"に変化した場合を考える。この時(2)のNodeは、"1"から"0"に変化するのであるが、その電圧変化はトランジスタ Q_L, Q_D のコンダクタンス g_L, g_D とNode(2)の配線の浮遊容量 C_S とNode(2)に接続された次段のゲートのトランジスタ Q'_D のゲート容量 C'_G で決まる。

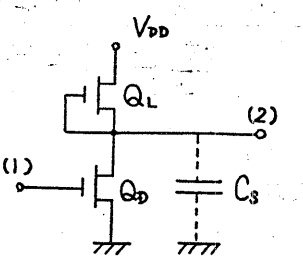


図 1 Inverterの回路

この Node(2) の電圧変化を他のアナログモデルのプログラムで求めると図2のようになる。この図において $T_{d(OFF)}$ は、当然 g_D (及び/又は g_L) に反比例し、C_S と C'_G に比例する。

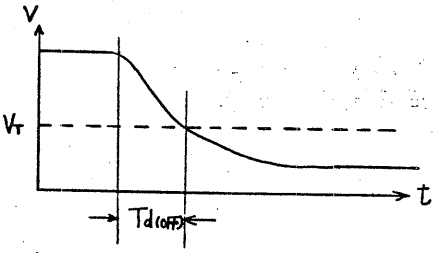


図 2. Node(2)の電圧波形

ここで各ロジックゲートの Driver (Q_D) の W と Load (Q_L) の W の比 (β_R) を2に固定し、各MOSトランジスタのゲートのチャネル長Lをパターン規準に従って固定したとすると、すべてのロジックゲートの寸法は最小ゲートに対するWの倍数Kで一意的に示される。従って各ロジックゲートのKの値を指定すると全トランジスタの寸法が決まる。

g_D (g_L) は、このInverterゲートのKの値 K_A に比例するから比例定数を $D_{(OFF)}$ とすると $T_{d(OFF)}$ は次のように表わせる。

$$T_{d(OFF)} = D_{(OFF)} \frac{C_S + C'_G}{K_A} \quad \text{----- (1)}$$

C'_G は、MOSトランジスタ Q'_D の面積に比例するが、チャネル長Lは固定であるので結局Wに比例することになる。Q'_D のWは次段のロジックゲートのKが与えられれば計算することができる。従って、電源電圧、プロセス、パターン規準、などが一組決まれば、その様な条件の下で一度解析した波形から $D_{(OFF)}$ を求めると、その後は与えられるKの値が違っても同じ $D_{(OFF)}$ を用いて(1)式から各ロジックゲートの $T_{d(OFF)}$ を求めることができる。

入力 Node(1) が"1"から"0"に変わり、出力 Node(2) が"0"から"1"に変化する場合の遅れ時間 $T_{d(ON)}$ についても、比例定数を $D_{(ON)}$ とすれば、全く同様にして次のように表わされる。

$$T_{d(ON)} = D_{(ON)} \frac{C_S + C'_G}{K_A} \quad \text{---- (2)}$$

$T_{d(ON)}$ と $T_{d(OFF)}$ は、一般には等しくないが、このシミュレータでは大きな方の値を T_d として採用する。

C_S には Q_D のドレイン拡散領域と Q_L のソース・ゲート領域の容量が含まれ、これは K_A により計算することができる。

結局、T_d は次の様な簡略式で計算される。

$$T_d = D \frac{\alpha K_A + \beta K_A \cdot N + C'_S + \gamma K_D}{K_A} \quad \text{--- (3)}$$

N: ゲートの入力の数

ここで比例定数 α, β, γ, D は別途、解析により求めてシミュレータに入力し、K と C_s（配線の浮遊容量）は各ロジックゲートごとにデータを入力する。

2.2 シミュレーションの方式

シミュレータには、次に述べる大きな三種類の内部記憶がある。シミュレータの Executer と呼ばれるプログラムが、これらテーブルとリストをもとにシミュレーションを行なう。

(i) ロジック・リスト

このシミュレータでは、ソースデータとして入力された図3の回路を、図4のように各ゲートを遅れのない論理素子と、あらかじめ求めた遅れ時間に対応する遅延線とに分離してリストとして記憶する。言いかえれば、このロジックリストは、各ロジックゲートの論理機能入力 Node名, 出力 Node名, 遅れ時間などからなるデータ（Logic Block）の集合である。

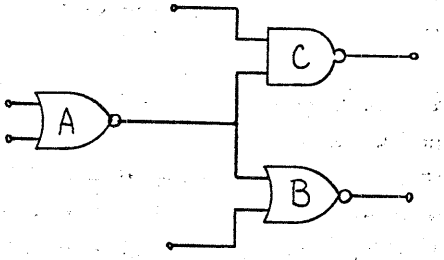


図3. 入力時の回路

(ii) Value テーブル

全 Node の論理値を記憶しているテーブルで、2種類ある。
1つは CVT（Current Node Value Table）で現在の Node の値を記憶する 図4で言えば (1), (2), (4), (5), (7), (8), (10) の Node の値を記憶している。
他の1つは LVT（Last Node Value Table）で未来のある時刻に Node が取るであろうところの値を記憶する。

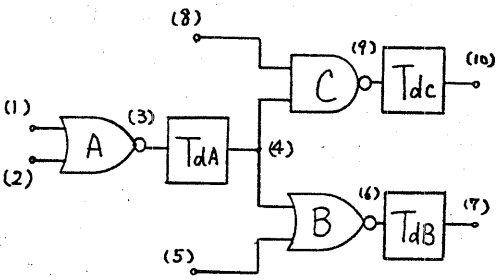


図4. 内部表現された回路

図4でいえば、(3), (6), (9) の Node の値を記憶する。しかし、これら Node は実際上は存在しないもので、(3) は (4) の, (6) は (7) の, (9) は (10) の Node に対応して仮想的に名付けたものである。二つの Value Table 共、シミュレーションの進行とともに変化する。

(iii) Event Table

事象（Event Block）[Node名とその変化後の論理値と、その変化時刻をその内容とする] の集合である。シミュレーションの進行とともにこのテーブルは長くなる。

シミュレーションは次のようにして行なわれる。まずシミュレートしようとする事象を Event テーブルに登録する。Executer は、その登録された事象について他への信号伝播があるかないかを調べあれば、その生じたすべての事象をテーブルに登録する。次に未処理の事象の中で一番現在に近いものを選び、その事象の時刻に時計を進めその事象について上記動作を繰り返す。この繰り返し動作によりシミュレーションを実行してゆく。未処理の事象がなくなった時、シミュレー

ションは終了する。

図3の回路において Node(1) にステップ波形を加え論理値が"0"から"1"に時刻0に変化するという例でシミュレーションの説明を行なう。

最初に、Node(1)が"0"から"1"に時刻0に変化するという事象をテーブルに登録する。そしてシミュレーションを開始する。Executeは Eventテーブルを調べる。現在の時計の時刻0に一番近い将来に起る事象を選ぶ。ここでいう時計とは、シミュレータがもっている内部記憶の一つである。この例では、選ばれるのは、いま登録した事象である。その内容に従ってシミュレータは時計の内容をこの事象の時刻にする。そしてCVTのNode(1)の値を"1"とする。次にロジック・リストを調べることで、このNode(1)がNode(4)のLogic Blockの入力となっていることがわかる。そこでNode(4)の全入力Nodeの現在の値をCVTから調べて、出力の値を計算する。Node(2)の値が"1"であったとするとNode(4)の値は"0"ということになる。これはTdA時間後にNode(4)が"0"となることを意味する。そこでNode(4)のLVT(Node(3)が対応)を調べる。今の事象がなければNode(1)は"0"であったからNode(3)は"1"であったはずである。この結果、Node(4)の値が変化するという事象が生じる。事象をEventテーブルに登録すると同時にNode(4)のLVTの値を"0"とする。他のLogic Blockへの信号伝播はないので、この事象に対する処理を終了する。

次にはTdA後の事象が現在に一番近いので、その事象に対する処理を行なう。まずロジック・リストからNode(4)はNode(7)とNode(10)の入力となっていることがわかる。Node(8)が"1"であればNode(10)は、時刻TdA+Tdcに"0"から"1"になるという事象が生じる。しかしNode(5)が"1"であるとすると計算されてむめられたNode(7)の将来の値は"0"である。Node(6)を調べると"0"であるので、新たに事象として登録する必要がないことがわかる。これでこの事象についての処理を終了する。次はTdA+Tdcの時刻の事象についての処理に移る。しかしNode(10)は、どこにも信号伝播をしないことが、ロジック・リストからわかるので、この事象に対する処理を終了する。これでEventテーブルには未処理の事象がなくなるのでシミュレーションは終了する。しかし、上記の例はごく概念的な記述であり、実際にはさらにいろいろな場合の判断が加わるし、プログラムもできる限り高速のシミュレートが可能なように作られていることはいうまでもない。

3. 回路図の入力形式

シミュレータ『MODELS』には、遅れ要素を含めた回路図を入力する必要がある。それらは、次の三種類に分けられる。

第一は、各Nodeの結合関係を示す論理式であり、他の二つは、遅れを計算するために必要な各ゲートの大きさと容量および定数である。

3.1 論理機能の種類

回路を論理機能の列で表現するため、次のものが用意されている。なおNode名のXXXとして許されるものは0～Fまでの16進数の三ケタである。
故に、このシミュレータで許されるNodeは最大4096個である。これは、このシミュレータが一度に対象とし得る回路規模の最大値と一致する。

(i)	XXX	IN	
(ii)	XXX	AND	XXX, XXX, ···, XXX
(iii)	XXX	NOR	XXX, XXX, ···, XXX
(iv)	XXX	XOR	XXX, XXX
(v)	XXX	NAND	XXX, XXX, ···, XXX
(vi)	XXX	TRG	XXX, XXX, ···, XXX, XXX (α) (β) (α) (β)
(vii)	XXX	PBF	XXX, XXX (α) (β)

(i) は XXX という Node が回路の入力端子であることを表わす。

(ii)～(v) は、論理記号の後に書かれた入力 Node の論理値に通常に用いられている意味で論理演算を行なって、出力 Node にセットする
但し、MOS 回路では実際上 AND, NAND, XOR (Exclusive OR) は2入力で用いられる。

(vi) は、MOS 特有の Transfer Gate である。MOSFET のソース側の Node が (α) であり、ゲートにあたる Node が (β) である。必ず (α) と (β) は対をなす。その機能は (β) の入力 Node が "1" の時、その直前に書かれた (α) の入力 Node の値が出力 Node にセットされる。すべての (β) 入力 Node が "0" の時出力 Node は不変とする。又二つ以上の (β) 入力 Node が "1" の時も、出力 Node は不変として扱う。但し、その旨はメッセージとして出力される。

(vii) は、Push-Pull-Buffer ゲートで図5の値が出力 Node にセットされる。ただし、疑不変は Transfer ゲートの最後の説明と同様にメッセージは出力される。

(β) \ (α)	"0"	"1"
"0"	不変	"0"
"1"	"1"	疑不変

図5. PBF の真理値表

論理機能ではないが、MODELS にはサブルーチン機能があり、入力の簡略化と共に、ロジック・リストに必要な記憶容量の減少を計っている。図6の例は、図7の回路をサブルーチン機能を用いて表わしたものである。

```
        SUB     5
A00     AND     B00, C00
<A10    NOR     A00, @D00
        RET
```

図6. サブルーチンの例.

SUB と RET の間に書かれた回路を SUB の後に書かれた数字分だけ繰り返して解釈する。「<」が頭についた Node は、サブルーチンで繰り返される部分の回路からその他の外部の回路への出力 Node であることを示す。「@」が頭についた Node はサブルーチンの回路へ共通に入力する Node であることを示す。

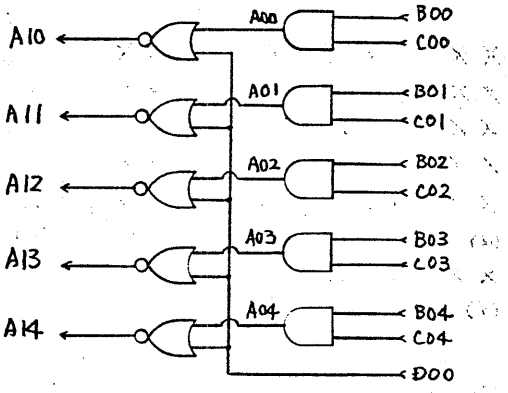


図7. 回路の例

3.2 Nodeのパラメータの入力形式

(i) XXX K, C

上記形式で論理機能がINのものを除いた全Nodeについて入力しなければならない。Kは、最小InverterのMOSトランジスタの大きさを1とした時の各ゲートのMOSトランジスタの大きさを表わす数値である。MOSトランジスタの大きさを決める要素の一つであるチャネル長Lは、通常設計規準で示される最小値に固定されるので、Kは、チャネル中Wに比例する。 Cは、各出力Nodeの配線の浮遊容量を表わす。

3.3 比例定数の入力形式

シミュレーションの原理で述べたように、Nodeの回路定数(パラメータ)が与えられた時に、各Nodeの遅れ時間を求める(3)式には、比例定数としてD, α, β, γがあった。遅れを計算するには、それら定数をあらかじめ、MOSトランジスタのアナログモデルのプログラムを用いて決定し、入力しておく必要がある。

(i) D1 = NN. NN
(ii) D2 = NN. NN
(iii) D3 = NN. NN
(iv) C1 = NN. NN
(v) C2 = NN. NN
(vi) C3 = NN. NN

D1, D2, D3は(3)式のDに対応し、C1, C2, C3はそれぞれ(3)式のα, β, γに対応する。

4．使用手順とプログラムの構成

このシミュレータを使ってシミュレーションを行なう時の手順は図8に示す通りである。手順の中に書かれているNameリストは、シミュレーションの実行に際し、ある特定のNode（たとえば、入力端子，出力端子など）の値を指定したりシミュレーション実行中あるいは、実行後の各Nodeの値を出力することが必要であり、そのような指定を対話形式で簡単に行なえるように考えられたものである。

入出力の指定の時、個々のNodeをその都度指定することは面倒であるので、入出力の指定に使うNodeをいくつかのグループとして定義しておき、入出力などの指定には、そのグループ番号を用いることにした。このグループがNameリストと呼ばれ、各グループには16個までのNodeが登録できる。又グループは最大16個まで定義できる。同様に論理値についても、Valueリストと呼ばれるものが考えられており、各グループには4ビット単位で32ビットの論理値のパターンを登録することができ、最大16グループまで定義できる。Nameリストと Valueリストのグループ番号を組合せて指定することにより、Nodeにある値をセットしたり、事象を登録したりする動作を対話型で行なうことを容易にしている。

手順の主要部分は、3章で述べた形式で入力した回路を、内部記憶の一つであるロジック・リストに変換する過程と、外部からよえられるコマンドに従って、シミュレーションを実行する過程からなる。変換する（Compile）プログラムをCompilerと呼び、実行する（Execute）プログラムをExecuterと呼ぶ。

この二つのプログラムは、独立に使用できるようにもなっている。又、この二つのプログラムを使用する時、不可欠のプログラムとしてTosbac-40の標準プログラムの一つであるText Editor「TIDE」がある。

この三つのプログラムはすべてTTYを用いた対話形式で使用することができるようになっている。

4.1 Compiler

Compilerは、3章で述べた回路の各Nodeについての論理機能及びパラメータと比例定数を入力して、内部記憶のロジック・リストに変換する能力をもつ。又、Nameリストを変換する能力もそつ。ロジック・リストに変換することで、Executerによるシミュレーションの実行速度を早めたり、記憶容量の節約を計っている。

4.2 Executer

Executerは、コマンド〔3章で述べたものは回路が決まれば、大体決定されるものであるが、実際にシミュレーションを行なわせるためには、その都度指定しなければならない条件などがある。このような指定をコマンドと呼ぶコを入力データとして、その指示により、各種の内部テーブルやリストの設定を行なったり、シミュレーションを実行したり、あるいはNodeの値及びその変化の経歴を出力したりする。Executerの主要部分は、実行命令でこれはシミュレーションの原理で述べたような動作をする。

4.3 TIDE

ミニコンTosbac-40の標準プログラムで、TTYで入力したデータをText Bufferと呼ばれる領域に記憶し、その内容の編集を行なうことができる。

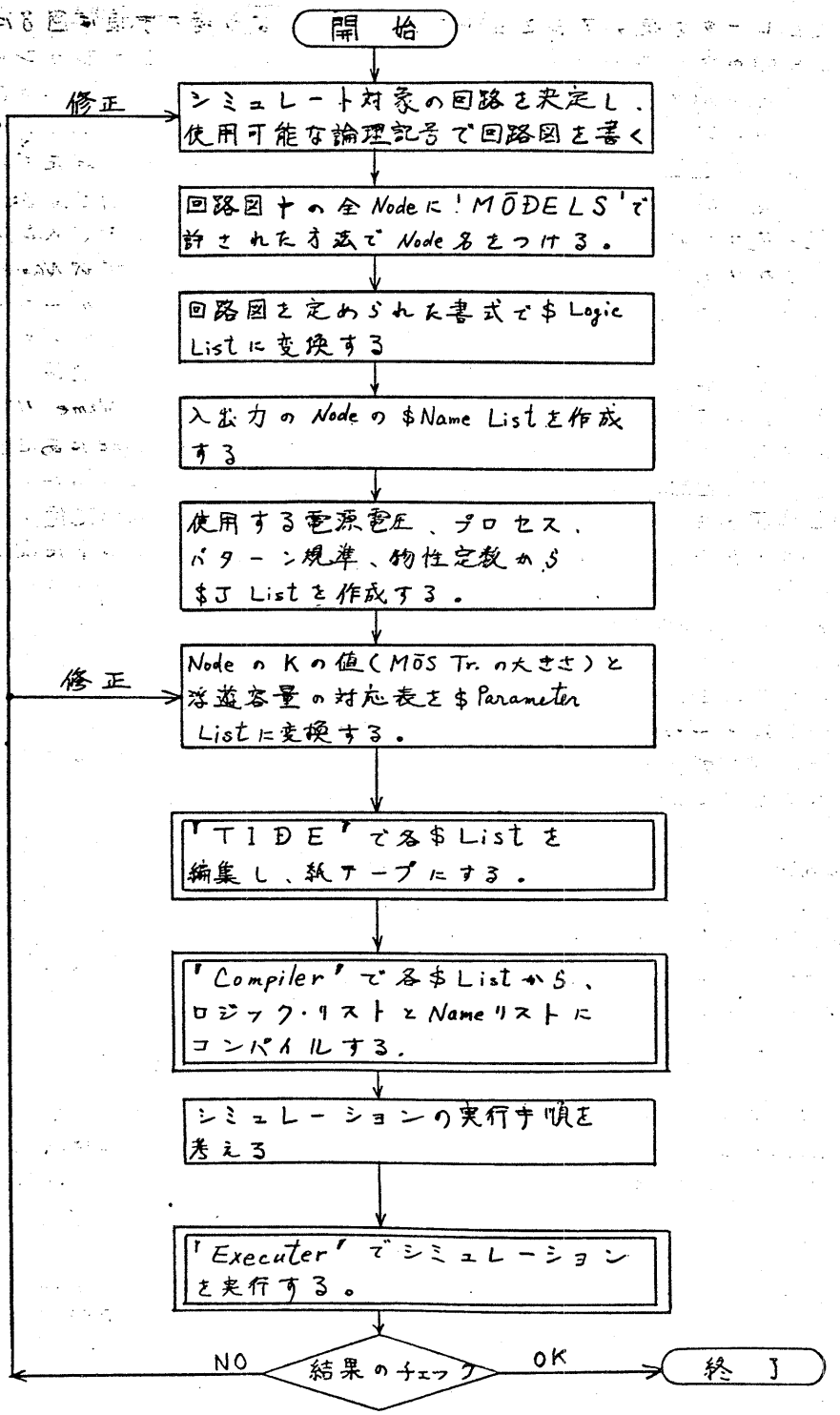
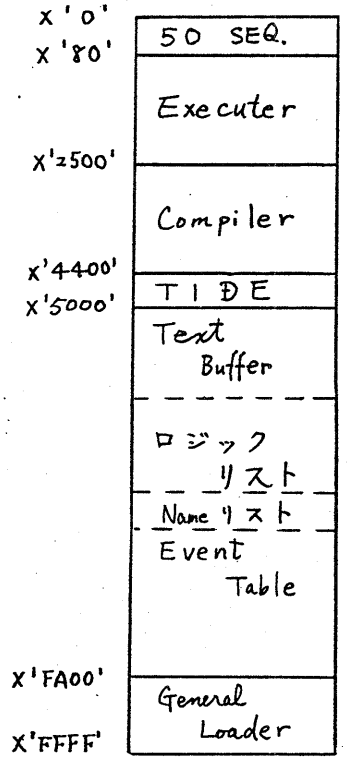


図8. 標準のシミュレーション手順

ローディング・マップを図9に示す。



三つのプログラムを常駐してシミュレーションを行なう時MODELSは48Kバイトの計算機で、約1000ゲート程度の回路のシミュレーションが可能である。

図9. MODELSのローディングマップ

5. あとがき

このシミュレータは、先に発表された'MOLS'[1]というシミュレータプログラムと対をなすように開発されており、できる限り同じデータを入力として利用できるようにしている。'MOLS'が回路の論理的誤りがないことを調べることを主目的としているのに対して、'MODELS'は回路全体の動作速度を調べることを主目的としている。そして、得られた結果をパターン設計に適用することで、性能のよいマスク設計ができることが期待される。なお、Node数約1800のLSIを対象とした時、一マシンサイクルのシミュレーションで、生じる事象の総数は1000～2000程度で、計算時間は1～2秒程度であった。

6. 謝辞

このプログラムの仕様決定に際し、有益な御意見を与えて下さった東芝総研、集積回路研究所の吉田主務をはじめ研究者の皆様に感謝致します。

参考文献

(1) 吉田，山崎，菅沼，渡辺："MOS/LSIの設計のための対話型ロジックシミュレータ" 昭和49年度信学会大会 1974年7月