MOSダイナミック状態の論理シミュレーション

末次逸夫　伊藤美和子　倉地正　三浦純一　渡辺錠
（東京芝浦電気株式会社）

1. 概要

MOS回路を取扱うシミュレータは今までにいくつか発表されているがMOS特有のダイナミックな動作をシミュレートするには不十分であった。ここで紹介する論理シミュレータはALS－4と呼ばれる7値シミュレーションによりバイポーラの回路はもとよりMOS回路の正確な動作解析を行うことができるものである。シミュレーション・レベルはゲート・レベルの，通常の論理素子に加えMOS回路固有の素子が20種程度用意され，さらに並列な信号の流れを一まとめに扱う並列論理回路や，レジスタ，バス，メモリなどの機能素子も用意されており，これらを組合わせて任意の論理回路を効率よく記述することができる。

さらにシミュレーションの対象となる論理回路をコンパクトに記述するためのいろいろな便法が設けられており，個々に記述する場合の1/2～1/3の量で記述することができる。

今までにALS－2，ALS－3と呼ばれるイベント方式の非同期論理シミュレータを開発し[5]，通常回路の動作，タイミング・チェックに適用してきたが，ALS－4はこれらをもとにMOS回路にまで拡張したものである。

2. はじめに

MOS　LSI設計の期間短縮，品質向上のため論理シミュレータが必要とされており数千から一万ゲートを越える程度の大きさの回路を正確にかつ実用的な範囲内でシミュレートすることが要求されている。

通常，設計は何人かで分担することが多く，最初は個々の担当分を検証し次に全体をまとめて検証するという手順をとる。又設計の段階に応じてシミュレータに対する要求も異る。最初は論理的な誤りを検出するのに主眼が置かれ，次第に詳細なタイミングを考慮したシミュレーションへと移行する。これはゲートの伝搬遅れはLSI内での配置が決ることにより初めて確定するためである。さらに一般に，論理シミュレーションを行うのは論理設計者自身であり，シミュレーションの対象となる論理回路の記述は便い易く，シミュレーション結果は見易くなければならない。シミュレーション結果を得るまでのターン・アラウンド時間も重要である。

ALS－4論理シミュレータはこれらの要求に答えるために次のような機能を持つ。

（1）　マクロ，リピート機能による論理記述のコンパクト化，見易さ。
（2）　リンカーによりいくつかの論理回路をつなぎ合わせる機能。
（3）　MOS特有の動作を正確にシミュレートするための7値シミュレーション。
（4）　並列論理回路や機能素子とゲート・レベルの記述の混在が可能。
（5）　TSS（Time Sharing System）を用いた会話型のシミュレーション。
（6）　伝搬遅れの定義が正確にできる。

（７）　種々の見易いシミュレーション結果が出力される。
（８）　豊富なシミュレータ制御コマンドがある。

ＡＬＳ－４論理シミュレータの構成は図１で示される．以下各プログラム毎にその機能と特徴を述べる。

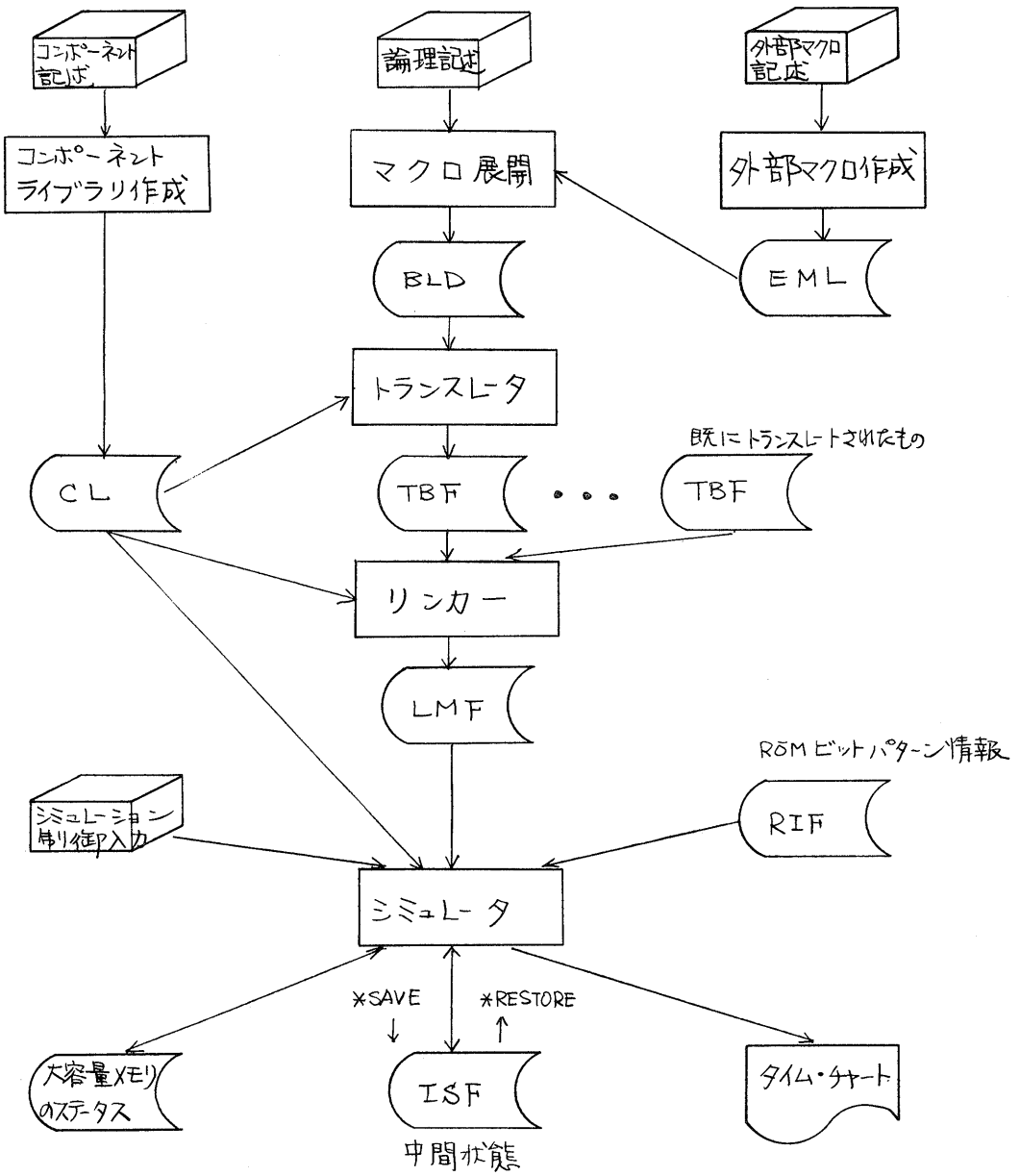
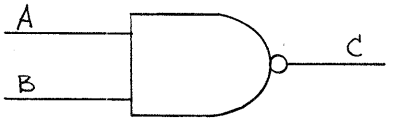


図１　ＡＬＳ－４　プロセス・フロー

3. マクロ展開，トランスレータ

シミュレーションをするために対象となる論理回路を構成する素子（ＡＮＤ，ＯＲなど）の情報とそれらの間の接続情報が必要である。ＡＬＳ－４は素子毎にその入力の情報，出力の情報を次のように記述する。



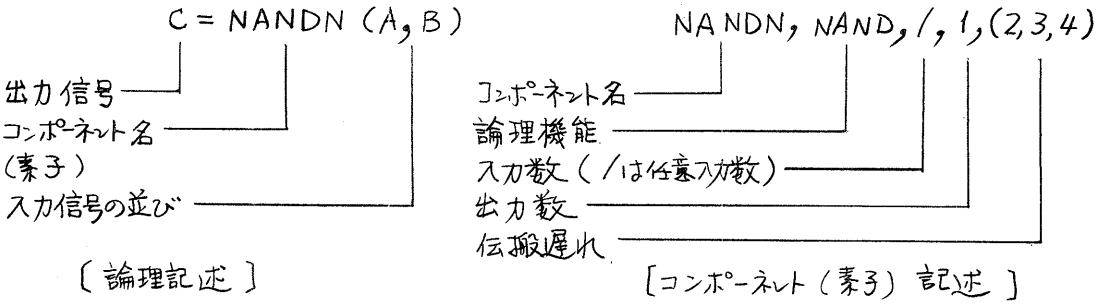


図２　２入力ＮＡＮＤゲートの記述

まず各つながりにユニークな名前（信号名）をつけ，素子にもその論理動作，入出力本数，伝搬遅れなどを含めて素子名をつける。そして図２のように出力信号名，素子名，入力信号名を記述する。これを回路内のすべての素子について記述する。

この方式では素子数分の記述が必要であるが回路内のくり返しに着目し入力量を減らすために次のような機能が用意されている。

（１）マクロ機能

くり返し現われる回路を一度マクロ定義で記述し，以降はそのマクロをあたかも１つの素子とみてその入出力に実際の信号名を割りあててマクロ・コールを行なう。

図３に簡単な論理回路図を示す。図４はこれを記述したリストである。ここで行番号００２０～００７０はＡＮＤＮＲ３というマクロの定義部分であり，行番号０１３０，０１４０，０１６０，０１８０はそのマクロをコールしている部分である。行番号００２０はマクロを１つの素子と見た時の入出力の並びを示す。行番号００３０～００６０は通常の論理回路記述と同様であるがマクロの内部の回路を記述している。内部の信号名は＃をつける。この＃のついた信号名はマクロ・コール毎に重複のないユニークな信号名に展開される。

行番号００２０の入力信号の並びとマクロ・コールは：を区切りとして対応している。従って行番号００２０のＩＮ１は行番号０１３０のマクロ・コールではＢ，Ｄに対応し，行番号０１６０ではＡ，ＢＢＡＲ，ＣＢＡＲと３つの入力信号に対応する。このようにして入力数は異なるが論理動作の同じものは１つのマクロで記述することができる。

さらにゲートの種類もマクロ・コールの引数にできるため，接続構造が同じでゲートのタイプが異なるものも同一のマクロにまとめることが可能である。

（２） リピート機能

同一種類の素子が規則的に並んでいる時，信号名に規則性を持たせることによりリピート機能を使ったコンパクトな記述が可能となる。

図４の行番号０２００～０２４０の５行がリピートの記述である。この例では＊がＡから順にＤまで変化する。この５行の記述は３×４＝１２行の記述に相当する。１次元だけでなく２次元，３次元，・・・のリピート記述が可能である。

（３） 同一素子省略記法

１つ前の素子の記述と素子名又はマクロ名が同じならば／で代用できる。例の行番号０１４０がこれに相当する。

（４） 代替名

信号名が長い場合に短い代替名で入力することが出来る。プログラムで自動的に元の名前に戻される。

これらの機能を使うと素子毎に書くのに比べ１/２～１/３以下の入力量ですますことができる。

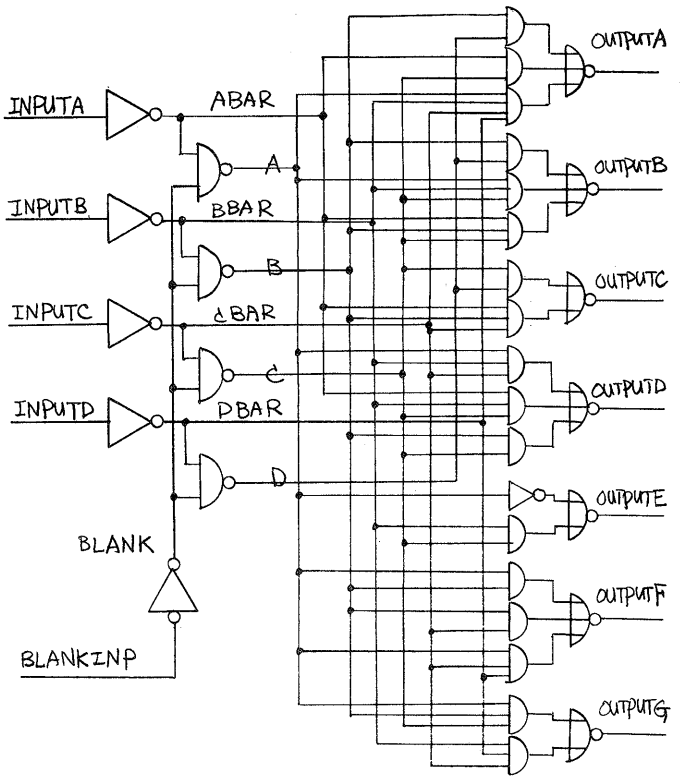


図３． 論理回路例

```
0010 **BLOCK,S7449
0020          /*  MACRO DEFINITION */
0030 *MACRO     OUT=ANDNR3(IN1:IN2:IN3)
0040            OUT=NOR3(#1,#2,#3)
0050            #1 =ANDN(IN1)
0060            #2 =ANDN(IN2)
0070            #3 =ANDN(IN3)
0080 ENDM
0090
0100 *MACRO     OUT=ANDNR2(IN1:IN2)
0110            OUT=NOR2(#1,#2)
0120            #1 =ANDN(IN1)
0130            #2 =ANDN(IN2)
0140 ENDM
0150          /*  LOGIC DESCRIPTION */
0160 OUTPUTA=ANDNR3(B,D:ABAR,C:A,BBAR,CBAR,DBAR)
0170 OUTPUTB=  /    (B,D:A,BBAR,C:ABAR,B,C)
0180 OUTPUTC=ANDNR2(C,D:ABAR,B,CBAR)
0190 OUTPUTD=ANDNR3(A,BBAR,CBAR:ABAR,BBAR,C:A,B,C)
0200 OUTPUTE=ANDNR2(A:BBAR,C)
0210 OUTPUTF=ANDNR3(A,B:B,CBAR:A,CBAR,DBAR)
0220 OUTPUTG=ANDNR2(A,B,C:BBAR,CBAR,DBAR)
0230          /*  REPEAT DESCRIPTION  */
0240 REPEAT *=(A-D)
0250       *=NAND(*BAR,BLANK)
0260       INPUT*=EXTINP
0270 ENDR
0280
0290 BLANK=ANDN(BLANKINP)
0300 BLANKINP=EXTINP
0310 ENDB
```

図４ 論理記述

4. リンカー

通常ＬＳＩの設計は複数の担当者で分担する場合が多い。例えばシステム・バスで分割して機能ブロックに分けて設計することが行なわれている。論理シミュレーションも個々のブロックで単体のシミュレーションを行なった後全体をつないで動作確認を行うことが多い。

この場合おのおのの機能ブロック単位で論理記述を行う。マクロ機能は各ブロックに固有のものは論理記述の中に含めるが，全体に共通するマクロについては外部マクロとして登録し，各ブロックから共通に参照するよう記述することができる。

リンカーは信号名の共通なものを探してつなぎ合わせる。従って各機能ブロック相互に共通な信号名は同じ名前に，それ以外は同じ名前が出てこないように信号名をつける必要がある。各ブロック内でユニークな名前をつけるために信号名の頭に自動的にブロック名をつける機能がある。

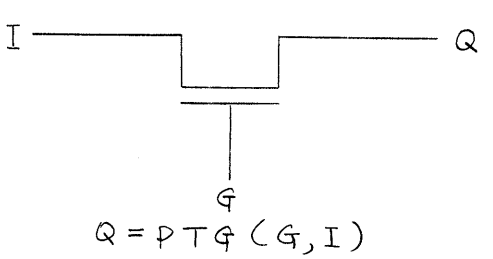
5. シミュレータ

(1) 7値シミュレーション

ＡＬＳ－４論理シミュレータはＭＯＳ回路のシミュレーションを精密に行うために次の7値を導入する。

0 —— 通常の論理 0
1 —— 通常の論理 1
X —— unknown (0か1か不明)
$\tilde{0}$ —— ダイナミックな0状態
$\tilde{1}$ —— ダイナミックな1状態
$\tilde{X}$ —— ダイナミックな unknown
E —— エラー

ここで $\tilde{0}$, $\tilde{1}$, その意味を説明するためにＭＯＳ回路に頻繁に現われるトランスファゲートでどのようなシミュレーションが行なわれるかを見ることにする。ここでいうトランスファゲートは一方向の流れを持つものである。

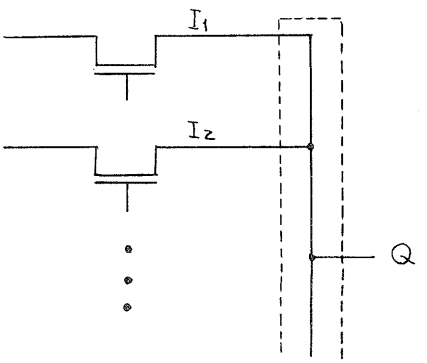


G	Q(n)
0	$\tilde{Q}_{(n-1)}$
1	I
その他	E

図５. Positive トランスファゲート.

入力Ｉが1でＧが1ならば入力と出力は導通し出力Ｑのキャパシタンスは正にチャージされる。ここでＧが0になると出力と入力の接続が切れてしまうが出力に蓄えられた電荷によって今までの1の論理レベルが保たれる。しか

し一定時間以上クロックを供給しないと出力に蓄えられた電荷は次第に放電し論理的にレベルを決められない状態になってしまう。
このようにクロックが切れていても前の論理レベルを保っている状態をダイナミックな値と呼び，一定時間経過後の状態をダイナミックなunknown状態と呼ぶことにする。
従来の論理シミュレータではクロックが切れた場合，出力は状態保存にしていたため例えばクロック回路の設計ミスによるクロックの入れ忘れのようなエラー検出は不可能である。ALS-4では論理0又は1からダイナミックな0又は1になった時刻に，一定時間経過後ダイナミックなunknown状態になるイベントを発生させる。クロックが入ればこのイベントはキャンセルされるが，クロックが入らずに一定時間経過してしまうとダイナミックなunknown状態Xになるシミュレーションを行う。
次にMOS回路特有のバス回路のシミュレーションを見ることにする。



$Q = MBUS(I_1, I_2, \cdots I_n)$

I_j \ I_i	0	1	X	$\tilde{0}$	$\hat{1}$	Z
0	0	E	E	0	0	0
1	E	1	E	1	1	1
X	E	E	E	X	X	X
$\tilde{0}$	0	1	X	$\tilde{0}$	*	$\tilde{0}$
$\hat{1}$	0	1	X	*	$\hat{1}$	$\hat{1}$
Z	0	1	X	$\tilde{0}$	$\hat{1}$	Z

*—入力のうち最後にダイナミックな状態になった値になる。

図6. MOS バス

バスで7値の強弱関係を示すと次のようになる。

$$\left.\begin{matrix}0\\1\\X\\E\end{matrix}\right\} > \left.\begin{matrix}\tilde{0}\\\hat{1}\end{matrix}\right\} > Z$$

即ちバスに0又は1の値が出ており他にダイナミックな値が出ている場合，出力は0又は1に引きずられる。バスに$\tilde{0}$又は$\hat{1}$が出ており残りがダイナミックなunknown状態の場合は$\tilde{0}$又は$\hat{1}$に引きずられる。このようなシミュレーションは7値を考慮しなければ正しく行なえない。
上の例からもダイナミックな値の導入が必要であることが分ったが回路すべてにわたって7値による精密なシミュレーションは必要でないことも多い。ALS-4論理シミュレータは通常のAND，NANDなどの素子を通過す

ると7値が次のように0，1，Xの3値に縮退する。

0 ——— 0
1 ——— 1
X ——— X
↑0 ——— 0
↑1 ——— 1
↕ ——— X
E ——— X

このことにより必要な部分は精密な7値シミュレーション，他の部分は3値による高速シミュレーションと有効に組合わせて使用することが可能となる。

(2) 双方向バス

双方向バスはプログラムで自動的に検出しシミュレーションしやすい形に再構成する。

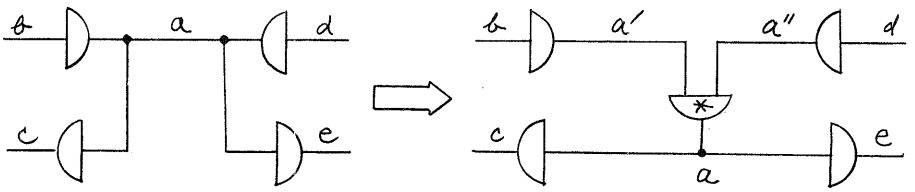


図7. 双方向バスの再構成

*の素子はプログラムで生成した仮想的な素子である。この素子の機能をwired AND，wired OR，又は前述のバスのうちから選ぶことができる。

次の図のような双方向トランスファ・ゲートを用いたバスもプログラムで自動的に検出し，シミュレーションしやすい形に再構成する。

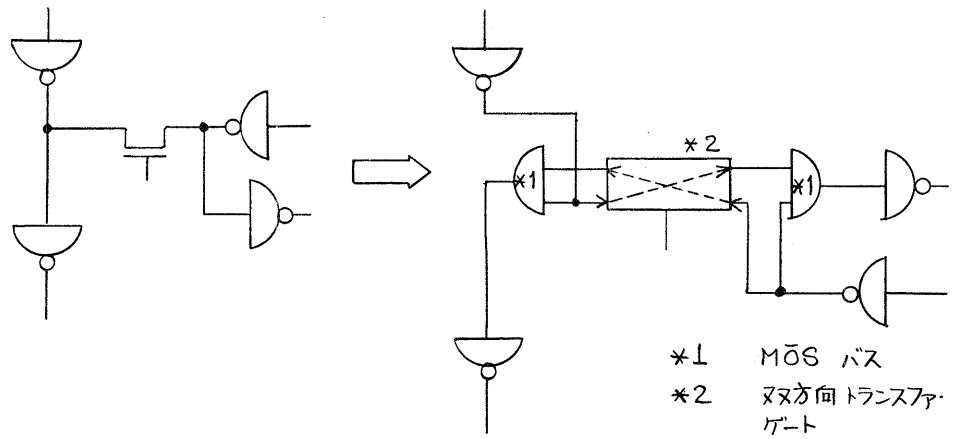


図8. 双方向トランスファ・ゲートを含むバス

(3) 伝搬遅れ

伝搬遅れは各素子に対しminimum, typical, maximum, の3種の遅れを素子の出力に対して定義することができる。

シミュレーションでは回路の全素子に対し一律minimumかtypicalかmaximumかを選んでシミュレーションを行う。設計の初期の段階では論理的な動きのみをチェックする場合がある。このようなケースに対応するため一律伝搬遅れを0にしてシミュレーションを行う方法も用意されている。

MOS個有の素子ではクロックが入らずに一定時間経過するとダイナミックなunknown状態になってしまうが, この一定時間を素子毎に定めることができる。

(4) 並列論理回路

データ・ラインやアドレス・ラインのように何ビットかのデータが平行して走るケースがある。リピート機能を使った記述も可能であるがこれらのビットをまとめて1つのビット中を持った信号の流れとして記述すると, 記述が容易で見易くかつシミュレーション速度も速い。並列論理回路と通常の記述とのインターフェイスにはIMPLOD, EXPLODと呼ぶ仮想的な素子を用いる。

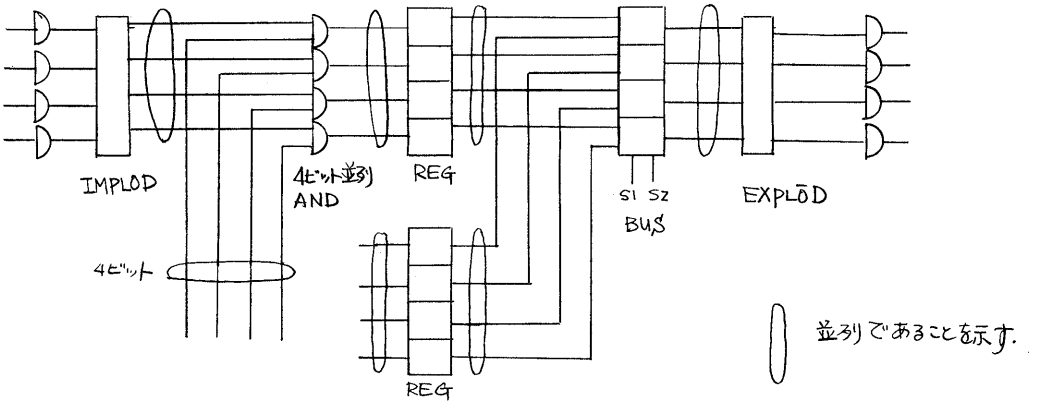


図9. 並列論理回路

6. 使用例

今までの使用実績の一部を下の表に示す。

	データ1	データ2	データ3	データ4	
素子数	800	1,200	2,500	8,000	
実行クロック	370	28,080	16,800	80*	576*
イベント数	188,866	367,360	—	—	—
メモリ・サイズ (KB)	284	286	328	560	560
CPU時間 (分)	1.9	4.2	18	8	27
1イベント実行時間(ms)	0.6	0.7	—	—	—

使用計算機はACOS/700

*はステップを表わす。 1ステップ=200ms

7. おわりに

ALS-4はSFORと呼ばれる構造化プログラミングの可能なFORTRAN[4]と一部アセンブリ言語により記述されている。総ステップ数は約25,000である。

現在は約8,000～10,000素子のMOS LSIのシミュレーション等に使用されているが，論理記述のコンパクトさ，リンカーの有効性，MOS用素子が十分であったこと，7値の精密なシミュレーションなどが実証された。

今後の考察としては設計段階に応じ，初めはスピードを重視した論理チェックを行い，徐々に精密なシミュレーションを行い，さらに故障シミュレーションも可能なきめ細いシミュレーション・レベルを備えたものが必要と思われる。

8. 参考文献

(1) P. Kozak et al., "Operational Features of MOS Timing Simulator", Proc. of the 12th DA Conference, June 1975, PP. 95-101

(2) B.A. Prasad, "Modelling Techniques for Dynamic Logic and Test Pattern Generation of a Micro-Processor", Proc. of the Sym. on DA and M Proc. PP. 100-107

(3) 里部，根本，"LSI論理シミュレータ；LOGOS2"，情報処理学会 夏季シンポジウム，1976

(4) 三好，伊藤，倉地，"Structured FORTRAN"，情報処理学会17回全国大会，1976，PP. 13-14

(5) 倉地，"非同期論理シミュレータ"，東芝レビュー，Vol. 29-7，Jul. 1974，PP. 636-640