大容量論理シミュレーションシステム(LASS)

須山正人 (日本電信電話公社 武蔵野電気通信研究所)　木谷有一 (日立製作所戸塚工場)　福田正広　米倉秀光　白木昇 (沖電気工業株式会社)

1. まえがき

論理シミュレータは、論理装置の論理設計が一通り終った段階で、論理設計に誤りがなかったかを調べるのに使われる。

昨今、論理装置の高速化、素子の高集積化が著しい。電子交換機用中央処理系装置の分野においても同様で、この度、次の点をねらいとした大規模装置向け論理シミュレータを開発した。

①.各種の高集積化した素子を一素子として扱えるよう、素子種別に対する汎用性を持たせること。
②.シミュレーション制御の容易さを向上させること。
③.処理効率を向上させること。
④.タイミング上のきめ細かいチェックが行えること。
⑤.論理設計から実装設計に至るまでの設計サポートシステムの一環をなすものであること。

これらをねらいとした、本論理シミュレータの主な諸元を、表1·1に示す。特に、①のねらいからLBL(Logic Block Library)を、②のねらいからTDL(Test Definition Language)を用意したことが、大きな特徴である。

ここでは、これらを中心に、システム構成、制御方法、処理方式、データ例、及び初期評価結果を報告する。

表1.1　主な諸元

項目	内容	ねらい
プログラム方式	インタプリティブ方式	②
演算順序制御方式	タイムマッピング方式	③
時間制御方式	非同期式	④
素子機能	内蔵機能のほか，LBLに登録する	①
シミュレーション制御方式	専用言語(TDL)によるテスト手順の記述	②
状態値の数	2値	③
並列シミュレーション	最大8バッチ	③
処理可能素子数	約10万ゲート	－
出力形式	レジスタ·タイムチャート，全素子状態表	－

2. システム構成

図2·1に、システム構成を示す。各ジョブはそれぞれ次の処理を行っている。

(1) LBL作成ジョブ

論理機能等をライブラリとして登録する。AND，NAND，OR，NOR，NOT，RS-F/Fの論理については、システム側で用意されている。それ以外の論理機能を、ユーザは遅延時間、入出力端子間の関係等を含めてSYSLを用いて定義できる。

(2) SMF作成ジョブ

装置の論理接続情報を、言語処理システムから入力する。言語処理システムでは、実装情報を含んだデータからも、シミュレーション用ファイルを出力する。

(3) TDF作成ジョブ

ユーザは、シミュレーション実行制御、周辺回路(擬似回路)の動作、タイム

チャートの出力対象素子の指定、等のデータ（ＴＤＤ）を専用言語ＴＤＬを用いて定義する。

(4) ＳＤＦ作成ジョブ

初期値等の多量データを、一般のデータ形式で記述したカード、あるいはビットパターンを作成するマイクロアセンブラ（ＭＰＡＳＳ－１）の出力から、セットデータ（ＳＴＤ）として作成する。

(5) シミュレーションジョブ

シミュレーション実行管理、並列シミュレーション時のテストケースとＴＤＤ、ＳＴＤの対応づけ等を記述したシミュレーション・コントロール・データ（ＳＣＤ）と、予め用意した各ファイルを入力し、シミュレーションを実行する。結果は、ＴＣＦ（指定素子のみについて細かいタイミング間隔での結果を持つ）、ＳＲＦ（全素子の値を指定タイミング間隔で持つ）として出力される。

(6) 出力ジョブ

シミュレーション結果を持つＴＣＦ，ＳＲＦから、タイムチャート，全素子状態表を出力する。シミュレーションジョブと独立させたことで、出力範囲の指定に融通性がある。

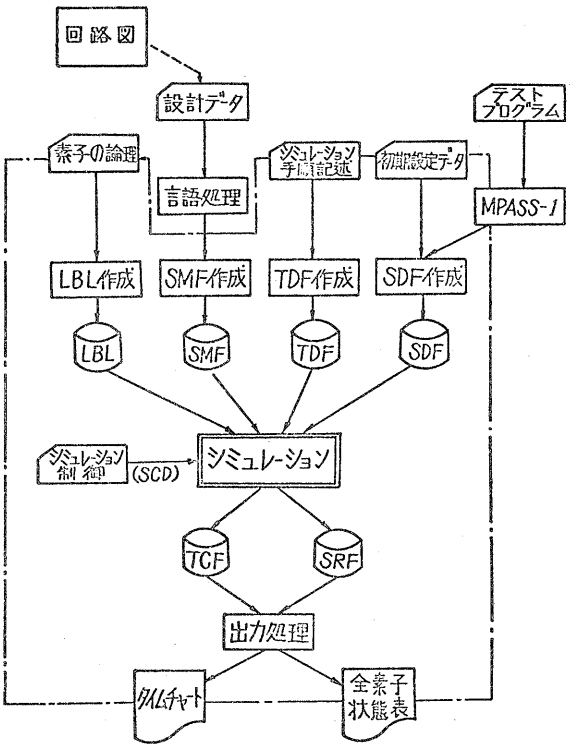


図2.1 論理シミュレータの構成

3. シミュレーション制御方式

3.1 本システムにおける制御の特徴

シミュレーション制御情報は、制御内容から次のように分類できる。(a)実行管理データ (b)実行モード (c)データの入出力指示 (d)実行順序指示 (e)周辺回路とのインタフェース定義 (f)強制セットするデータ

本システムでは、これらを表3.1に示す三種のデータにより分類制御している。なお(f)は本質的にはＴＤＤの一部と考えられるが、単純なデータの羅列が大量にある場合には、ＴＤＤと切り離してＳＴＤとして入力できる。

表3.1 制御用データ

データ種別	入力時期	記述形式	対象制御情報
SCD	シミュレーション実行時	パラメータ形式	(a),(b),(c)
TDD	シミュレーション実行前に解釈・編集	言語形式	(d),(e)
STD	シミュレーション実行前に解釈・編集	パラメータ形式	(f)

3.2 ＳＣＤ、ＴＤＤ、ＳＴＤ間の対応

ＴＤＤは、プロシージャ（ＰＲＯＣ）を一まとまりとして定義する。ＳＴＤは

ＤＡＴＡを一まとまりとして定義する。一つのテストケースは、一般に複数のＰＲＯＣと複数のＤＡＴＡを使用する。このようなテストケースとＰＲＯＣ，ＤＡＴＡとの対応づけは、ＳＣＤにより実行時に指定できるようにし、ＰＲＯＣ，ＤＡＴＡが複数のユーザにより共用できる等、それらの間の組合せの融通性を保っている。

3.3　ＴＤＬ

本言語の処理方法としては、コンパイラ・インタプリタ方式を採用している。コンパイラ部分（ＴＤＦ作成ジョブに当る）では、宣言部のテーブル化、文の種類やオペランドの種類によるコード化、演算式の逆ポーランド記法への変換、名称による参照部分のポインタ化を行っている。また、インタプリタ部分（シミュレーションジョブのＴＤＤの実行部に当る）では、コードに応じた処理ルーチンを呼び出し、オペランドに対応する処理を行っている。

ＴＤＬの言語仕様上の特徴を、以下に示す。

①. シミュレーション実行制御が行える ── シミュレーションの開始、終了、実行待ち、更に他のプロシージャの起動などの制御を、それぞれ一文で記述できる。また、複数の制御手順を並列に処理する制御方法が、簡単に表現できる。

②. データセット機能を有する ── 各種レジスタ、素子の入出力端子などにデータ、値を代入文の形でセットできる。

③. 判定機能を有する ── 各種レジスタの内容、信号の値等を判定し、その結果によりシミュレーション制御手順を変更できる。

④. タイミングの記述機能を有する ── 絶対時刻、相対時刻の記述、ある条件が成立するまで待つ待機の記述、時系列的なデータの代入が可能である。

⑤. 出力指定機能を有する ── タイムチャートの出力素子の指定、及びシミュレーション実行時のメッセージ類のプリントアウト指定が可能である。

4. シミュレーションの処理方式

4.1　処理概要

本論理シミュレータでは、処理の融通性を増す目的からインタプリティブ方式を採り、更に処理時間の短縮が図れるイベント法のうち、遅延機能が付加しやすいタイムマッピング法（ＴＭ法）を採った。

システムの中心部であるシミュレーションの処理の流れは、図4.1に示すように、１タイミングについてＴＤＤの実行、論理演算、演算結果の出力を繰り返す。ＴＤＤの実行だけは各バッチ（並列シミュレーションの個々のテストケース）直列に実行するが、他はバッチ並列に実行し、処理時間の短縮を図っている。

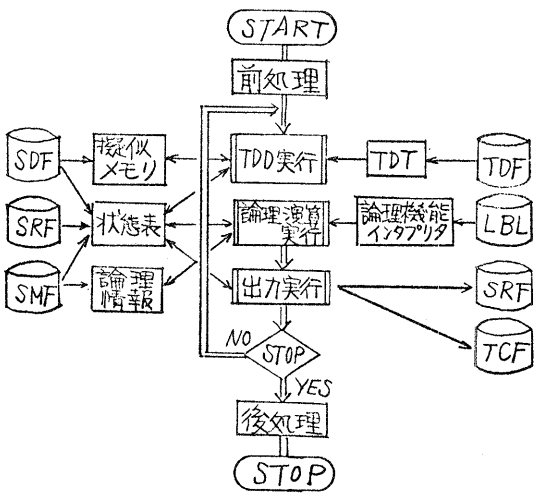


図4.1　シミュレーション処理概要

4.2 論理演算処理方式

本システムの論理演算処理部（ＴＭ法）の概要フローを、図4.2に示す。ＴＭ法は、入力に変化のあった素子のみを演算するイベント法の一種である。イベント法ではすべてのイベントを同一のスタックに格納するが、ＴＭ法はスタックを時間対応に持ち、イベントを時間軸に沿って登録しておくことで、細かいタイミングでの演算を可能にした方法である。

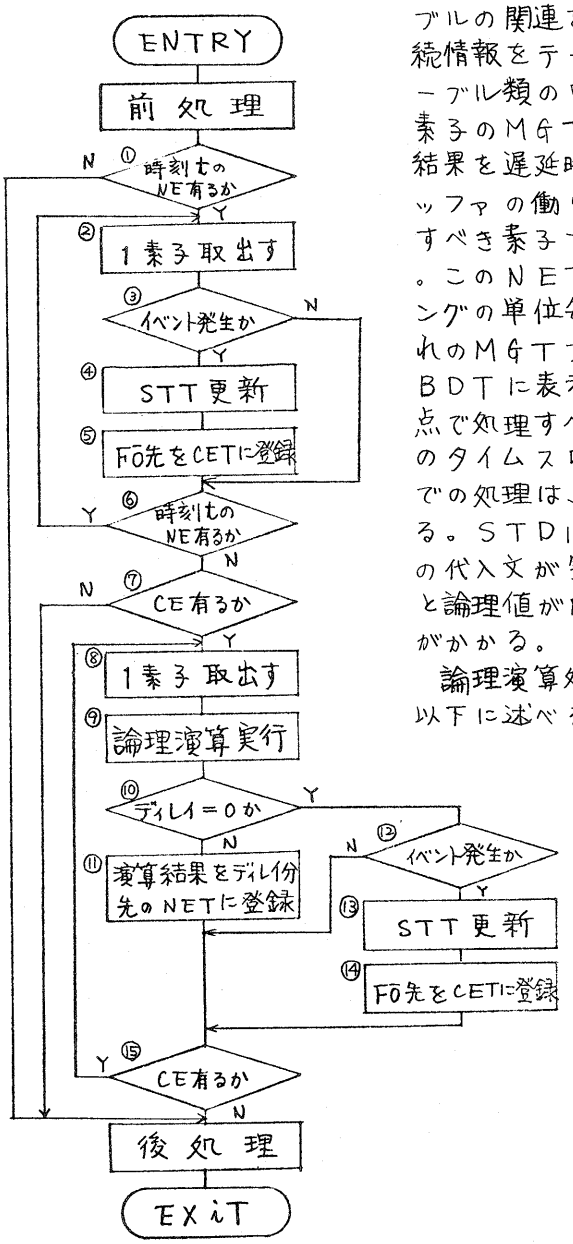


図4.2　論理演算処理フロー

論理演算を行なう場合に使用される各種テーブルの関連を、図4.3に示す。ＭＧＴは論理接続情報をテーブル化したもので、処理されるテーブル類の中心である。ＮＥＴは、処理すべき素子のＭＧＴアドレス、論理値等を持ち、演算結果を遅延時間の間だけ一時的に貯えておくバッファの働らきをし、ＣＥＴは、次に論理演算すべき素子をスタックしておくのに用いられる。このＮＥＴ，ＣＥＴは、ＭＧＴのバッファリングの単位毎にブロック化されており、それぞれのＭＧＴブロックの先頭アドレスは、ＭＧＦＢＤＴに表示されている。ＴＣＴＰＴは、現時点で処理すべきＮＥＴがつながっているＴＣＴのタイムスロットを指している。１タイミングでの処理は、このＮＥＴ中素子の処理から始まる。ＳＴＤによるデータの初期セット、ＴＤＤの代入文が実行されると、その指定された素子と論理値がＮＥＴに登録され、演算処理に起動がかかる。

論理演算処理における本システムの特徴を、以下に述べる。

(1) イニシャライズシミュレーション

イベント法を採る２値シミュレータでは、実際の論理演算の前に、イベントが正しく伝播するように論理の連続性を保証しておく処理が必要であり、これをイニシャライズシミュレーションと呼ぶ。ＳＭＦに論理接続情報の外にＳＴＴを持たせ、これにシミュレーション結果を保存させることで次回からのイニシャライズ処理を省けるようにしている。

(2) 処理時間、使用エリアの節約

論理装置の中でのクロック信号は、発振回路で作られファンアウト回路により装置各部に分配され

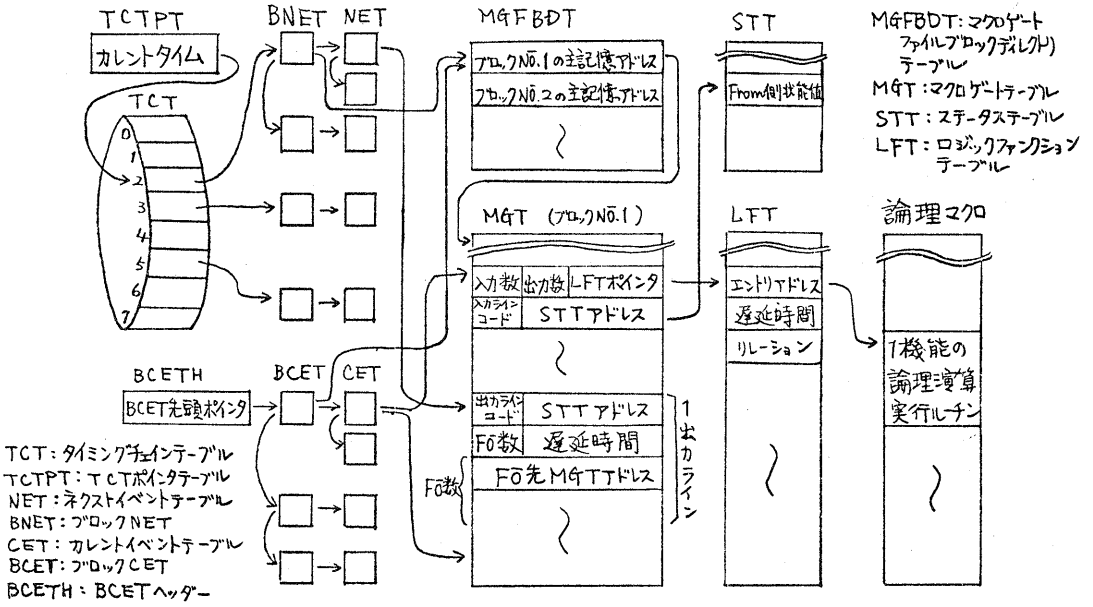


図4.3. 論理演算実行時のテーブル関連図

る。従って発振器にイベントを発生させなくても、クロック信号のファンアウト先に直接イベントを発生させれば十分であり、その結果としてシミュレーション時間の短縮が図れる。

入力が複数本ある場合には、それぞれの入力からイベントが伝播してくることがある。それが同時刻ならば処理は一度だけでよいので、NET登録, CET登録の際に同一素子二重登録を避けている。その際、前述のNET, CETのブロック化が二重登録判定時間の短縮に寄与している。また、並列シミュレーションでは、複数バッチにイベントが生じても、イベントの処理は1回で済む。

一つの素子の二出力が補数の関係にあるものは、そのことをLBLで定義しておくことでSTTを共有化し、メモリを節約している。

(3) 論理マクロとのインタフェース

演算論理を記述した論理マクロを使って論理演算を行うには、MGT中の入出力ライン数、STTアドレス等を知る必要がある。システム組込み機能については、処理の高速化を図る目的で、論理マクロ中でこれらの情報を参照するところまで記述している。しかしユーザが論理をLBLで定

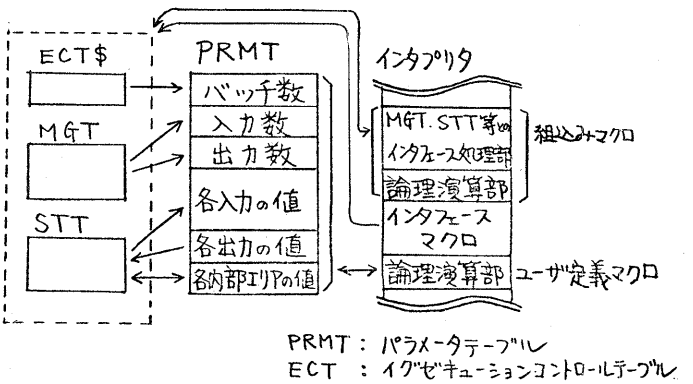


図4.4 論理マクロの位置づけ

義する際、これらの情報のはいっているＭＧＴアドレス等を意識して記述させることはできない。そこで論理演算部とＭＧＴ等とのインタフェース処理を司どるルーチン（インタフェース・マクロ）を独立させ、図4.4に示すように、これらの必要データを全てＰＲＭＴに準備しておく。その結果、ユーザがこれらのデータだけを使って論理を記述することを可能とした。

(4) ＭＧＴのバッファリング

対象回路が大規模になると、接続情報のテーブル（ＭＧＴ）が大きくなり、全テーブルをコアに常駐できなくなり、バッファリングが必要になる。ＭＧＴ上の素子の配列は図面位置順とすることで、イベントの伝播していく先をＭＧＴ中の互いに近くに配置している。また、クロック信号を受ける回路等の、イベントが頻繁に発生する回路のＭＧＴセグメントは、コア上に常に留まるようにすることが望ましい。

これらのことを考慮して、以下に示す不要ページ追い出しアルゴリズムＬＲＵＲＳ（Least Recently Used Excluding Resident Set）を採用した。

これは表4.1に示す基本的なアルゴリズムＬＲＵ，ＬＦＵを基本とし、ＲＳの条件を加味したものである。すなわち表4.2の例に示すように、最終参照時刻を第１キー，参照回数を第２キーとして、ソートされた順に追い出すが、その際ＲＳならば追い出しを禁止する、というものである。

表4.1　バッファリングの基本アルゴリズム

アルゴリズム名称	追い出し基準	対象データの特性
ＬＲＵ (Least Recently Used)	最近参照されてないページを追い出す	最近参照されているものは、近い将来に又参照される可能性が高い。
ＬＦＵ (Least Frequently Used)	参照頻度の少ないページを追い出す。	参照回数の多いものは、将来また参照される可能性が高い。
ＲＳ (Resident Set)	参照頻度の特に高いページは、追い出しを禁止する。入れ替えエリアページ数のα％のページが、レジデントセットとなる	周期的に参照されるものは、通算の参照頻度が高くなる。

表4.2　追い出し順序のソート例

最終参照時刻	2	2	3	5	5	5	6	----
参照回数	8	2	4	15	10	3	7	----

→

5. 各種データ例

本シミュレータの各種データ例と、シミュレーション結果を表5.1に示す。

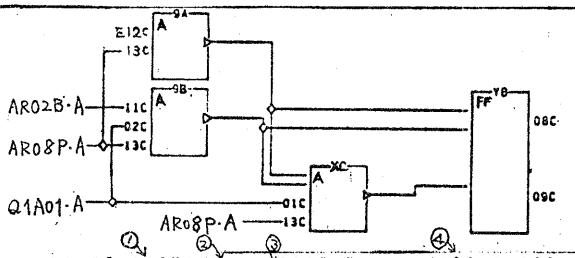
表5.1　データ例　（1/3）

種類	内容	データ例
論理機能記述データ	①.論理ブロック名定義 ②.論理マクロ名定義 ③.入出力ライン名定義 ④.内部エリア定義 ⑤.入力ラインの空き処理 ⑥.出力に対するディレイ値 ⑦.入力ライン間、出力ライン間の関係 ⑧.論理マクロの動作定義	① *FUNC　JK/F ;　⑤ *IN　③→S/1,J/1,C/1,K/1,R/1 ; *OUT　③→A/15.5N/20.5N/30N,N/15.5N/20.5N/30N ; *INT　④→JK(2) ;　⑥ *REL　⑦→C(A,N),NEQ(S,R) ; *PROC　②→ECL132/SYSL/S ; DCL B BIN ; DO B=1 TO BNO ; ⑧ IF SUBSTR(S,B,1)='0'B THEN DO ; SUBSTR(A,B,1)='1'B ; SUBSTR(N,B,1)='0'B ; } JKEND:　END ; *FEND ;

表5.1　データ例（2/3）

論理接続情報

①.素子名定義
②.論理ブロック名
③.図面位置
④.接続関係




```
*LOGIC AR09A<N  > @AR0.9A     /I01=QF000    ,I02=AR08P.A ;
*LOGIC AR09B<N  > @AR0.9B     /I01=AR02B.A ,I02=Q1A01.A ,
                                                I03=AR08P.A ;
*LOGIC AR0XC<N  > @AR0.XC     /I01=AR09A.A ,I02=AR09B.A ,
                                 I03=Q1A01.A ,I04=AR08P.A ;
*LOGIC AR100<FF > @AR0.YB     /S01=AR09A.A ,S02=AR09B.A ,
                                                R01=AR0XC.A ;
```

SMF入力データ

①.クロック入力ラインと相指定
②.固定値入力指定
③.ディレイ値指定

```
①*CLOCK FQ101.I01/1;FQ101.I02/1;FQ102.I01/1;FQ102.I02/1;
        FQ103.I01/1;FQ103.I02/1;FQ104.I01/1;FQ104.I02/1;
②*TERM  AR09B.I01/0    ;
③*DELAY AR0XC.A  /ION   ;
```

TDD

①.タイムチャート出力レジスタ定義
②.シミュレーション手順定義
③.周辺回路動作定義

```
①*FAM R0(16) : AR0F0,AR0E0,AR0D0,AR0C0,AR0B0,AR0A0,AR090,
               AR080,AR070,AR060,AR050,AR040,AR030,AR020,
               AR010,AR000;
  *FAM R1(16) : AR1F0,AR1E0,AR1D0,AR1C0,AR1B0,AR1A0,AR190,
               AR180,AR170,AR160,AR150,AR140,AR130,AR120,
               AR110,AR100;
②*TEST EINIT ;
          ECL0A = '1'B , '0'B ;
       +5 START 0 ;
          LR = '0010'X ;
          ES13A = '1'B ;
          EST0A = '1'B ;
  *END ;
③*PSEUDO FMEM ;
  /* MAIN MEMORY */
  *DEF     FMADR(13) ;
  *DEF     RC(8) ;
  *TAB     $FMEMORY(17,1024,0):PRIVATE ;
  FMACES:  IF CEN06=0 THEN FMADR= MAB ELSE WAIT ;
           IF CLOE1 BRANCH FWRITE,FREAD ;
  FWRITE:  +1 CALL RWRITE($FMEMORY,MDB,FMADR,RC) ;
           GO TO FMEND ;
  FREAD:   +1 CALL RREAD($FMEMORY,MWB,FMADR,RC) ;
  FMEND:   CASE1=0 ;
           CPTE1=0 ;
           +2 GO TO FMACES ;
  *END;
```

STD

①.データ名定義
②.データのタイプ指定
③.設定値
④.対象素子名
⑤.ビット数
⑥.開始番地
⑦.データ

```
$DATA, IN SYSTEM   ;
0      APF50, APF70, APF80,
       GR0CF, GR1CF, GR2CF, GR3CF, GEMAF, GRPTF,
       ASF30, ASF40 ;
1      APF40, APF60,
       ASF00, ASF10, ASF20, ASF50, ASF60, ASF70,
       AIM00, AIM10, AIM20, AIM30, AIM40, AIM50, AIM60,
       AIM80, AIM90, AIMA0, AIMB0, AIMC0, AIMD0, AIME0;
$END ;

$DATA, RN ADD,17 ;
0  15FFF ;
10  0C100,0C001,06100 ;
$END ;
```

表5-1　データ例　(3/3)

SCD

①対象装置名
②クロック定義
③テーブルエリア量
④MGTバッファエリア量
⑤実行モード
⑥ユニットタイム
⑦チェック機能選択
⑧ディレイ値選択
⑨TDD選択
⑩STD選択
⑪実行時間
⑫SRFへの出力タイミング

```
$EQUIP  ①'D20CC' / / 'D20CCTG' , 4(50N/96N)  ;
$AREA   ③EVT = 1500  ;                        ②
$BUF    ④57  ;
$MODE     I  ;
$SMLGO  ⑤    ;   ⑥
$MODE     L,  1ON,  L/NR/NF,  ⑧MIN  ;
                    ⑦
$TEST     ESUY01 / 'LOAD-ADD TEST', EINIT/FMEM  ;
$IPL    ⑩$FMEMORY(0) , ADD(0)  ;                ⑨
$INIT     SYSTEM  ;
$SMLGO  ⑪30C, SRF001 (0/30/1P(1))  ;
                      ⑫
```

出力指定データ

①テストケース名
②出力タイミング
③入力ファイル名
④出力フォーマット

```
                    ①          ②
$PEQUIP  D20CC , ESUY01    , 0/45/1P(1), T/ 1 ;
$PCHART ;
    REG  R0/R1/R2/R3,RBR,BR,LR,PF,SF,ISF/IMF,IR,PBA/PBB,
         MCTL/MAB/MDB/MWB ;
                    ①
$PEQUIP  D20CC ,ESUY01, 8/10/1U(0), S/SRF004 ;
$PALLST ;
```

シミュレーション結果

①タイムチャート
②オールステート

①

CYCL.PHNO	R0	RBR	BR	LR
	R1			
	R2			
	R3			
0. 1	FFFF	0000	0000	0010
	FFFF			
	FFFF			
	FFFF			

②

ALL STATE LIST ### AT 8CYCLE 16UNIT

NO 000				064				128
0	6000	0000	0000	0000	0300	0E00	7003	800E	0300
320	01E1	F006	3FFF	E10F	FE89	AF27	4883	14C1	8007
640	2027	3E80	03DE	A47F	F21F	5E5B	8010	FD7C	3F80
960	8C00	003F	5C41	401F	FBFF			FF08	CAFF
	60FF	FEDF	FFFD	FFFF					FFFF

6. システム評価

5章で示したデータ等を使って、約7000ゲートの論理装置(中小局用電子交換中央処理装置)をモデルに、その基本命令を組合せたTPを擬似プロシージャで記述したメインメモリに設定し、実行させた結果を述べる。

素子数が一定の場合、シミュレーション処理時間(CPUタイム＝T)に影響を及ぼすと考えられる要因として、次のものが考えられる。

①処理イベント数　(E)
②バッチ数　(B)
③ユニットタイム間隔　(U)

表6-1に実行例を、図6-1～図6-3にこれら要因と処理時間の関係、MGTバ

ッファエリアとバッファリング回数の関係を示す。

表6.1　実行例

項　目	値	備　考
処理イベント数(E)	136891個	基本命令のTPを4相30サイクル間実行
バッチ数 (B)	1バッチ	
ユニットタイム間隔(U)	10 NS	各相間10等分
MGTバッファリング回数(T回)	56回	全MGT(56ページ)がはいるエリア量で実行
処理時間(CPU)(T)	123 S	1.4 S/クロック

<考察>

①.図6.1から、処理時間は処理イベント数に大きく依存する。

②.バッチ数を増やしても、イベント数が多少増すこと以外に直接の影響はなく、個々に複数回実行させるよりもずっと効率がよい。例えば異種の6本のTPを一度に実行しても、イベント数は、そのうちの1TPを単独で実行した場合の約5割増にしかならない例がある。(この数値は、どのようなテストケースを組合せて並列シミュレーションを行なうかによって、大きく変ってくると思われる。)

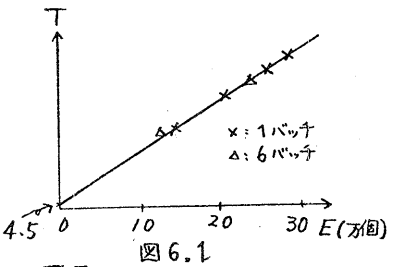


図6.1

③.イベント率をE÷(全素子数×全演算相数×2)×100(%)と仮定すると、このモデルの基本命令実行時のイベント率は、表6.1から約8%である。

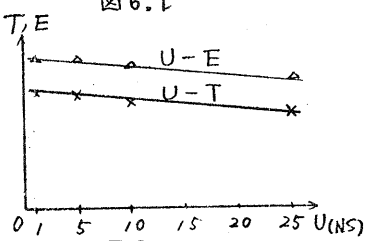


図6.2

④.図6.2から、ユニットタイムを小さくとり細かいタイミングで演算しても、処理イベント数はあまり変化なく、処理時間に大きな影響を与えない。

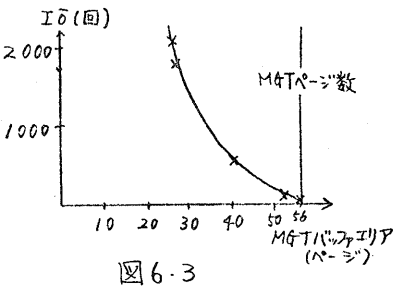


図6.3

⑤.イニシャライズシミュレーションでは、CPUタイムで約900秒とかなりの時間を要する。これは、論理の切り口(T素子と呼ぶ)からイベント発生と否とにかかわらず全素子無条件に演算する必要があり、その際、イベントテーブルがオーバフローしない様、少しづつ処理しているためである。

7. おわりに

以上、新たに開発した論理シミュレータの構成、処理の特徴を中心に述べた。現在、武蔵野通研にて運用中であり、今後は、運用評価にもとずく検討、改良を通じ、LSI等他の分野への適用範囲拡大化を図りたい。なお、運用結果並びにそれにもとずく効率、ユーザへの運用性等の評価については、今後引継き報告する予定である。

<謝辞>

本システムは、通研、交換機製造会社4社が協力して開発したものであり、多くの方々に検討に参加していただいた。日本電気(株)、富士通(株)の関係諸氏、並びに武蔵野通研交換室鵜野室長に、深く感謝いたします。

＜参考文献＞

(1) 高島他 ：論理シミュレータLSS4
電気通信学会交換研究会 1965年2月

(2) 高島他 ：並列処理を用いた大容量高速論理シミュレータ
情報処理，Vol.7，NO.5，P263，1966年

(3) 西村他 ：NELDA－Ⅱ論理 Simulatorに於けるblock化について
電気四学会連合大会 1970年

(4) 永峰他 ：論理シミュレータ（LSP－3）について
電子通信学会全国大会 1971年

(5) 村上他 ：大規模回路用汎用論理シミュレータ
情報処理学会DA研究会 1975年

(6) 星、中林他 ：マイクロプログラムアセンブラの一構成法
情報処理学会DA研究会，資75-24，1975年

(7) 森、中林他 ：設計データ記述言語の一構成について
電子通信学会全国大会，1304，1976年

(8) 星他 ：論理シミュレーション制御の一方法について
電子通信学会全国大会，1302，1976年

(9) 米倉他 ：専用言語によるシミュレーション実行制御の一方法について
電子通信学会全国大会，1301，1976年

(10) 木谷他 ：論理シミュレーションの処理方式に対する一考察
電子通信学会全国大会，1300，1976年

(11) Y.T. Yen，"A Mathematical Model Characterizing Four-Phase MOS Circuits for Logic Simulation"
IE³ Trans. Computers Vol. C-17 NO.9 1968

(12) S.G. Chappell 他 "Simulation of large asynchronous logic circuits using an ambignous gate model"
，FJCC 1971

(13) E.G. Ulrich "Serial/Parallel event schduling for the simulation of large systems"
Proceedings－1968 ACM National Conference