設計自動化16－2
（1983. 2. 15）

テーブルドリブン方式論理機能シミュレータ

樋浦尚登　星野民夫　上田和宏　渡辺俊男　菊池満孝
（　日本電信電話公社　武蔵野電気通信研究所　）

1、はじめに

ＶＬＳＩ設計支援システムの一環として、機能記述を用いた論理シミュレーション，テスト発生システムＦＯＲＥＳＴを開発してきた。〔1〕,〔2〕,〔3〕,〔4〕ＦＯＲＥＳＴシステムは、図１に示す構成となっており論理シミュレーションの観点から①機能記述のマクロなモデルを使用するためシミュレーション処理時間が減少する，②機能設計の段階から設計検証を行える，③設計進捗に応じたシミュレーションを行える（機能記述と構造記述〔5〕の混合シミュレーション），④機能記述によりファンクションテスト用のパタン作成が容易にできるなどの特長を持っている。本報告では、〔4〕で述べた機能・構成を持つ論理機能シミュレータの処理アルゴリズムおよび機能レベルシミュレーションを実際の設計に適用した結果を述べる。

2、ゲートレベル

図１に示すようにＨＳＬ〔5〕を用いてゲート，ピン間の接続を記述した場合、ＦＯＲＥＳＴでは構造オブジェクト（ＳＯＢＪ）をシミュレーションモデル（ＳＤＭ）に変換しシミュレーションを行う。ここでは、ＨＳＬで記述されたゲートレベル回路のシミュレーション機能および方法について述べる。

2.1　シミュレーション機能

ゲート遅延の扱い方により、零遅延，標準遅延，最大･最小遅延の３種のシミュレーションモードを有し、零遅延・標準遅延で０，１，Ｘ（不定値），Ｚ（ハイインピーダンス値）の４値、最大･最小遅延では４値に加えＰ（立上り論理値），Ｎ（立下り論理値），Ｗ（エラー論理値）からなる７値を扱う。サポートするゲート種類は、基本論理ゲート，フリップフロップゲート，トランスファゲート，トライステートゲート，プッシュプルゲート，ワンショットゲート，ＲＯＭ･ＲＡＭ等のメモリゲート

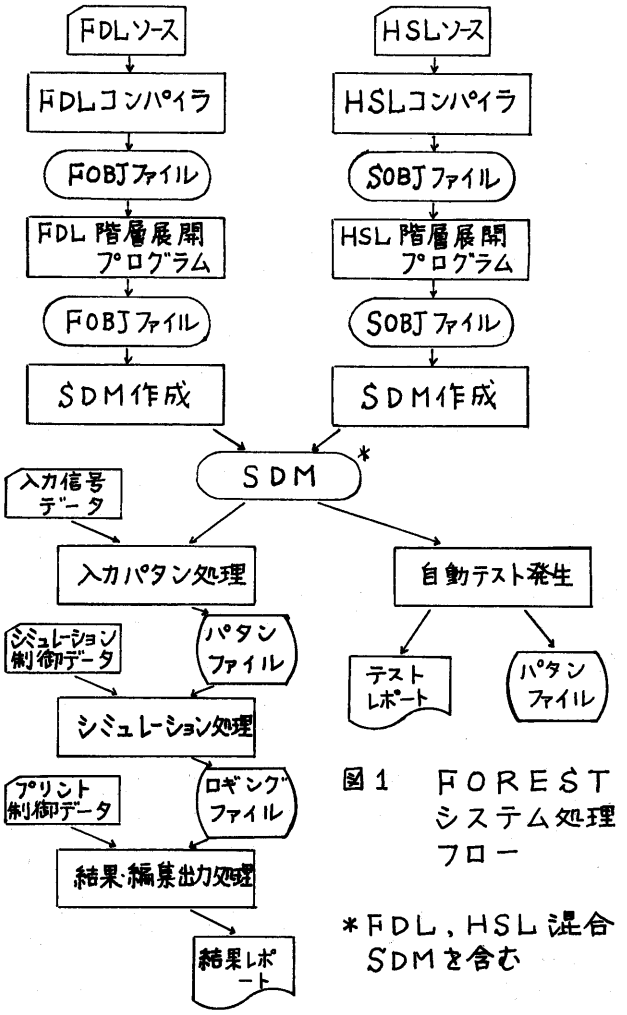


図１　ＦＯＲＥＳＴシステム処理フロー

＊ＦＤＬ，ＨＳＬ混合ＳＤＭを含む

の43種である。遅延を扱うシミュレーションモードでは、ハザードの検出などタイミングエラーの検出が可能である。

2.2 シミュレーション手法

シミュレーションアルゴリズムとしては、通常のタイムマッピングイベント法を用いており図2に示すようなイベントスケジュールの重なりによるスパイク（静的ハザード）の検出を行う。[6]シミュレーションは整数値でコード化された論理値を用いゲートの真理値表（テーブル）を引用しながら進める。

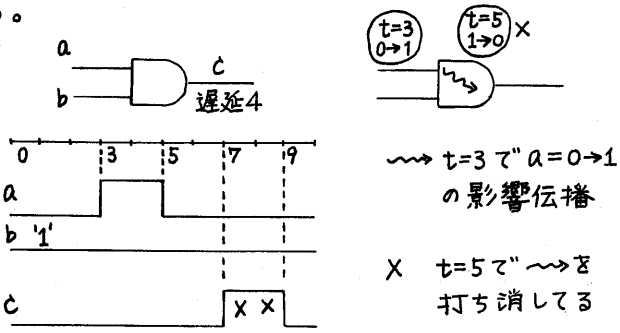


図2 スパイク検出

2.2.1 コード化テーブルルックアップ法

多値を扱うシミュレータでは、ゲートの多値論理演算を必要とする。この論理演算を行う手法には、①論理値をビット列で表しアセンブラもしくはPL/Iでビット列の論理演算を行う，②テーブルルックアップ法[7]と①の組合せ，③デコーダなどの論理機能ゲートにてデシジョンテーブルを用い他を①で行うなどがある。これらの手法は、a)アセンブラなどのサブルーチンコールによる処理時間のオーバヘッドがある，b)5値あるいはそれ以上の多値では多値用のゲート論理演算式を求めることが難しい，c)記憶域を多く必要とする，d)ゲート演算ルーチン内にビット列↔数値コードの変換にオーバヘッドがある[8]などの欠点がある。

以上を解決するため、論理値をコード化しこれをシミュレーション全体（イベント制御部，ゲート論理演算部）に渡って適用する。すなわち、論理値を1バイトの整数値でコード化し、イベント処理，論理演算に用いる。例えば、イベントの伝播判定には図3，ANDゲートの演算には図4のテーブルに対し、コード化論理値をアドレスとして引用する。ビット列論理演算，テーブルルックアップ法，コード化テーブルルックアップ法の3手法の比較を表1に示す。表からコード化テーブルルックアッ

STUPN

New / Old	'0'	'X'	'1'	'Z'	'Y'
	0	1	2	3	4
'0' 0	0	1	2	3	0
'X' 1	0	1	2	3	1
'1' 2	0	1	2	3	2
'Z' 3	0	1	2	3	3
'Y' 4	0	1	2	3	4

```
VAL = STUPN(ゲート論理値,
            イベント論理値);
IF VAL ≠ ゲート論理値
 THEN Update ;
Y：無効イベント論理値
   （イベントキャンセル用）
```

DELSEL

New / Old	'0'	'X'	'1'	'Z'
	0	1	2	3
'0' 0	2	3	1	1
'X' 1	2	3	1	4
'1' 2	2	3	1	2
'Z' 3	2	3	1	4

イベントの伝播判定

DELAY	1	2	3	4 (アドレス)
	Drise	Dfall	Min	Max

```
ADR = DELSEL(ゲート論理値,
             イベント論理値);
ディレイ = DELAY(ADR);
```

イベントディレイの計算

図3. イベント処理部

AND	'0'	'X'	'1'	'Z'
	0	1	2	3
'0' 0	0	0	0	0
'X' 1	0	1	1	1
'1' 2	0	1	2	1
'Z' 3	0	1	1	1

```
DCL FAND(0:3,0:3)
    BIN FIXED(7);

AND(0,X)=0
   ⇓
RESULT = FAND(0,1);
```

図4. ANDテーブル

プ法は、記憶域を多く必要とする，ファンイン数に制限がある（テーブルルックアップ法）、速度が遅い，多値に対する作成デバッグが難しい（ビット列論理演算）などの既存手法の問題点を解決しかつビット列論理演算より高速となっている。

2.2.2 フリップフロップ素子

FORESTでは、ポジティブ・ネガティブエッジトリガ，マスタスレーブなどのクロックに対する区別とSR，JK，D，Tなどのデータ入力に対する区別によって14種のフリップフロップを用意している。これらのフリップフロップは内部的に全てマスタスレーブ動作をすることによってクロックのHigh，Lowレベルでのデータ入力値を間接的に保持している。例えば、ネガティブエッジJKフリップフロップでは図5に示すようにTa時刻でのJ入力変化をマスタ記憶域MQに保存し、Tb時刻にてQ,$\bar{Q}$に（0,1）を出力する。論理シミュレーションでは、通常シミュレーションの済んだ時刻のゲート論理値は外部記憶に出力されることからTb時刻ではTaでのJ=0論理値を観測できない。これに対処するためクロックがHigh，Lowレベルでのデータ入力履歴をマスタフリップフロップに保存し、処理している。（フリップフロップの全入力端子の過去値を保持し行う方法もある）

図6にコード化テーブルルックアップ法によるネガティブエッジJKフリップフロップの演算ルーチンフローを示す。処理フローからわかる様にプリセット，プリクリア端子によるセット，クリア動作はクロック，データ入力に関す

表1　3手法の比較

項目＼手法	コード化テーブルルックアップ	テーブルルックアップ	ビット列論理演算
速度	中	速	遅
記憶量	中	大	小
作成デバッグの容易さ	易	難*	難
多値への適合性	有	有*	有
ファンイン数制限	無	有	無
マシン互換	有	無**	無**

＊多値に対する部分
＊＊演算部をアセンブラで作成時

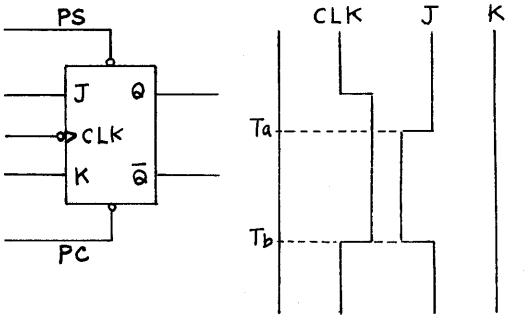


図5　フリップフロップ動作

PS＼PC	'0' 0	'x' 1	'1' 2	'z' 3
'0' 0	10	0	0	0
'x' 1	2	1	1	1
'1' 2	2	1	20	1
'z' 3	2	1	1	1

ARY

CLKO＼CLKC	'0' 0	'x' 1	'1' 2	'z' 3
'0' 0	10	10	20	10
'x' 1	40	40	40	40
'1' 2	30	30	30	30
'z' 3	40	40	40	40

ARY1

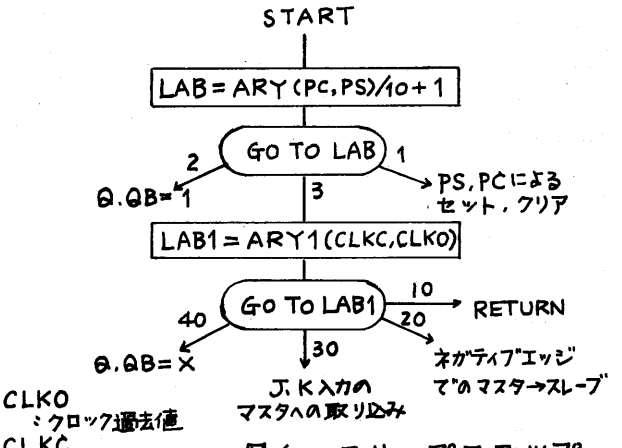


図6　フリップフロップ処理フロー

る処理を経て直接出力端子に論理値を出力している。

3、機能レベルシミュレーション

ＦＤＬ[2]による機能記述は、ＨＳＬと同様コンパイル，展開，モデル作成の処理を経てＳＤＭに変換される。(図１)　シミュレーションは、このＳＤＭ(接続線，素子など要素はゲートレベルＳＤＭと異りビット幅を持つ)を用いて行う。ここでは、ビット幅が可変となるイベントの処理，１リソースに対し同一時刻で複数の信号源からのアクセスが生ずる設計エラーの検出処理，イベントの処理順序などについて述べる。

3.1　シミュレーション機能

ゲートレベルと同様、素子遅延の扱いにより３種のシミュレーションモードを持っているが、零遅延と標準遅延を通常用いる。素子遅延は、複数ビットを持つ素子の各ビットに同一遅延を割り当てるものとして扱う。処理可能な素子はレジスタ，ラッチ，メモリ，ターミナル，オートマトン制御，クロック，定数などがあり0,1,X,Zの４値を基本とする。機能レベルでは、8,10,16進データの入出力が有効でありこれらのデータを扱い得る。また、特に１レジスタに複数のデータ転送が同時に生ずるエラーなどリソースの共用に関する設計誤りをチェックすることができる。

3.2　シミュレーション手法

ゲートレベルと同様、タイムマッピングイベント法を用いており、各素子がビット幅(最大２５６ビット)を持つことからイベントテーブルが可変長となっている。(図7)また立上り，立下り遅延が異る素子のイベントは、各ビットの論理値に従い時間軸上分割してスケジュールされる必要がある。これを表すため、イベントとして無効な論理値Ｙを内部的(利用者からは見えない)に導入した。(図８)

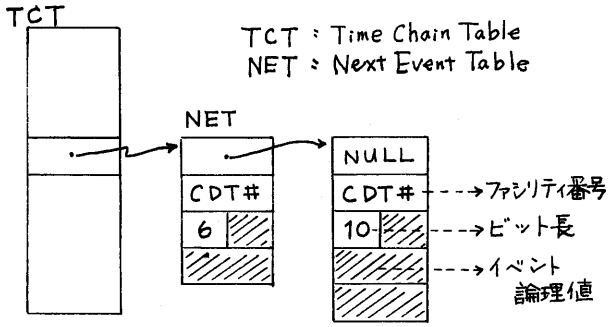


図７　可変長イベントテーブル

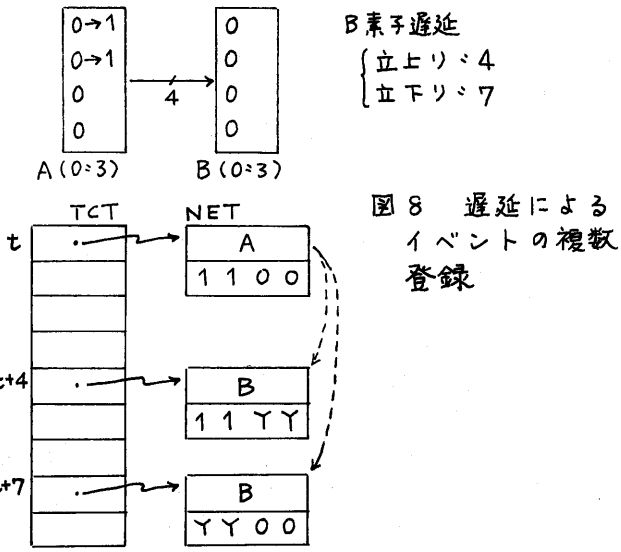


図８　遅延によるイベントの複数登録

3.2.1　シミュレーションモデル

図９，付録１にＦＤＬによる記述例とそれに対応するシミュレーションモデルの構造を示す。記述例ＳＡＭＰＬＥというモジュールは、ＣＯＮＴＲＯＬというオートマトンを持ちIF1,IF2,IF3,IF4,EXLD,EXAR,EXST なるステートにおいてレジスタのクリア(IF2ステート)外部端子からの命令取り込み(IF3ステート)命令レジスタIRの解釈･分岐(IF4ステート)バッファレジスタのロー

ド（EXLDステート）ＡＬＵによる演算（EXARステート）アキュームレータの外部出力（EXSTステート）などの動作をクロックＰに同期して行う。シミュレーションモデルは、これらの記述を信号のＳＯＵＲＣＥ，ＳＩＮＫに従った有向グラフモデルに変換したものである。例えば、レジスタＡＣＣへのデータ転送はPcファシリティ（モデル作成プログラムにより作られたパスファシリティ：ＦＤＬ記述上は動作として現われレジスタなどのようにExplicitに定義されていない）によって、ステートIF2がオンしているときクロックＰに同期して８ビットの０（8D0）が転送される、あるいはステートEXARがオンしているとき、BOX.X（ＡＬＵの出力）のデータがクロックＰに同期して転送されることを示す。

3.2.2　同時転送・接続の検出

タイミングまたは条件などによって、複数の信号源から１つのレジスタあるいはターミナルにデータが同一時刻に転送あるいは代入されることがある。例えば、図10の記述は条件C_1，C_2がオンしているときにクロックが立下りを起こしたとするとＡレジスタにＢの値とＤの値を転送することになり設計エラーである。これを検出するため転送を制御するパスファシリティでは条件オフ時にハイインピーダンス値Ｚを出力しておき、ＳＩＮＫのレジスタ演算処理にてＺ以外のデータを取り込む（通常処理）あるいはＺ以外のデータがファンインに複数存在するときはエラーメッセージを出力する。同様の設計エラーはターミナルへの接続動作でも生ずる。（図11）

```
NAME    : SAMPLE ;
PURPOSE: FDLSIM ;
LEVEL   : CHIP ;
<EXT>     RUN,CLR,READ,DBUS(1:8),IRIN(1:4),
          XBUS(1:8) ;
<INP>     .RUN,.CLR,.READ,.DBUS,.IRIN ;
<OUT>     .XBUS ;
<CLK>     P ;
<TYP>     ALU(A(1:8),B(1:8),C(1:2),X(1:8)) ;
          ALU : BOX ;
<END>   ;
<REG>     ACC(1:8),BUF(1:8),IR(1:4) ;
<BOO>     BOX.A:=ACC, BOX.B:=BUF, BOX.C:=IR(3:4) ;
<AUT>     CONTROL: P ;
          <STA> ;
          IF1:  .RUN ! .CLR : ?.CLR? ->IF2 % ->IF3 ; ;
          IF2:  ACC<-8D0, BUF<-8D0, IR<-4D0, ->IF4 ;
          IF3:  .READ : IR<-.IRIN; ->IF4 ;
          IF4:  .RUN : ?IR(1:2) #1 ->EXLD
                                #2 ->EXAR
                                #3 ->EXST
                                %  ->IF1  ;;
          EXLD: .READ : BUF<-.DBUS,->IF1 ;
          EXAR: ACC<-BOX.X,->IF1 ;
          EXST: .XBUS:=ACC,->IF1 ;
          <END> ;
<END>     CONTROL ;
END     : SAMPLE ;
```

```
NAME    : ALU ;
PURPOSE: FDLSIM ;
LEVEL   : END ;
<EXT>     A(1:8),B(1:8),C(1:2),X(1:8) ;
<INP>     .A,.B,.C ;
<OUT>     .X ;
<BOO>     ?.C #0 .X:=.B
               #1 .X:=¬.B
               #2 .X:=.A&.B
               #3 .X:=.A!.B ;;
END     : ALU ;
```

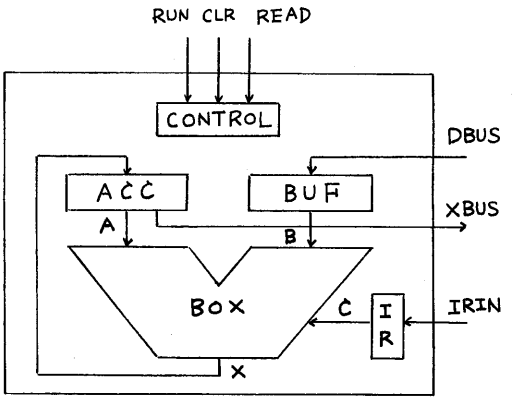


図９　ＦＤＬ記述例

ST1：？C_1？A←B；，？C_2？A←D；；

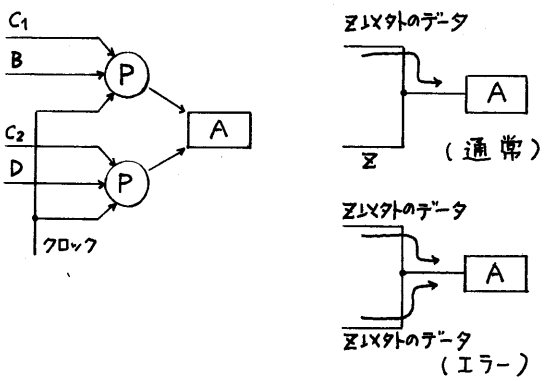


図10　同時転送例とエラー検出

また、オートマトン内のステートが同時にオンすることもあり、オートマトンステート論理値（0，1のみ）更新後、ステート間リンクをたどりチェックする。（図12）

3.2.3 イベントの処理

機能レベルにおいては、レジスタ同志のデータ交換操作が可能である。[9] 例えば、図13のST1におけるEとFに関するレジスタ転送はデータ交換動作を示している。この動作は、E←F処理後にF←E処理を行ったのでは実現できない。このため、データ交換操作のレジスタ転送はマスタスレーブ動作をさせクロックのポジティブエッジもしくはHighレベルでマスタへの取り込みを、ネガティブエッジでスレーブ出力を行う動作としている。

また、FDLでは一相クロックシステムを記述する場合オートマトン・ステート内のレジスタ転送とオートマトン・ステート間のステート遷移は同一のクロックで制御される。そして、同一ステートにあるレジスタ転送とステート遷移には動作の順序があり、レジスタ転送動作後ステート遷移動作が起こる。この動作をシミュレートするため、同一時刻内のイベントに対しファシリティ種類（レジスタ&ステートファシリティ）に応じた処理順序を設けている。

```
ST1 :  ?C1?  A := B ;,
       ?C2?  A := D ;;
```

C1とC2が同時にオンしてるとエラー

図11　同時接続例

```
ST1 :  ?A?  -> ST2 ;,
       ?B?  -> ST3 ;;
```

条件A,Bが同時にオンするとエラー

図12　ステート同時オンエラー例

```
<AUT> EXAMPLE : P ;
<STA> ;
  ST1 :  E←F , F←E , -> ST2 ;
  ST2 :  B←D , -> ST3 ;
    }
<END> ;
<END> EXAMPLE ;
```

図13　レジスタのデータ交換とイベント処理順序

4. 共通機能

ゲートレベル，機能レベルに共通した機能として、発振の検出および後処理，初期シミュレーションについて述べる。

4.1 発振検出および後処理

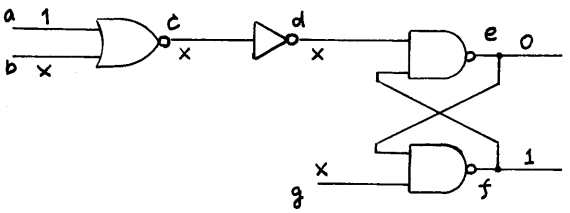
零遅延シミュレーションあるいは遅延を考慮したシミュレーションでは、トポロジ的にループを形成した回路がある場合発振を起こす可能性がある。この場合の発振判定は、「カレントイベント数が回路中の全ファシリティ数を越えてもまだイベントが残っているとき発振とみなす」によって行う。発振検出後、発振を止めシミュレーションを続行するあるいは処理を中止するかどうかは、コマンドにより指定する。発振を止めるには、検出ファシリティの論理値をXに変えることで実施される。

4.2 初期シミュレーション

原始入力から信号パタンを印加せずとも、電源ON時に論理値の定まる回路があり、それを模擬するため任意のファシリティに初期値を設定できる。この初期値は、原始入力から信号パタンを印加する前に回路中に伝播させておく必要がある。初期シミュレーションを通常シミュレーションと同様に行った場合、図14のように指定した初期値が消失してしまう。このため図15の2フェーズ処理を行う。

本手法はLAMP[10]で採用された方法であるが、LAMPでは初期状態でのXとその後のXとで異る論理値を割り当てている。一方、FORESTは論理値を増やさず初期値伝播フェーズと不定値伝播フェーズとで論理値の更新を図16に従って行う。残留X値の伝播シミュレーションは、図17の初期値伝播シミュレーション後の予盾を除くためのものである。

このように論理値が増えることに対する処理の複雑さを各フェーズで用いるテーブルの入れ換えのみで対処することができる。



a, e, f 端子の初期値からシミュレートすると、eの初期値0がXとなる。

図14 初期値からの通常シミュレーション

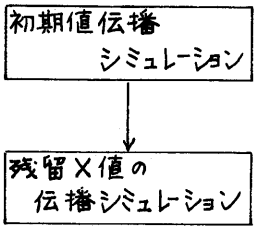
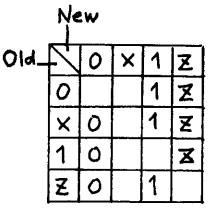
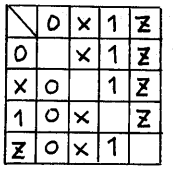


図15 初期シミュレーション



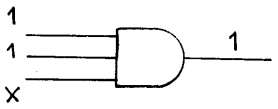
初期値伝播シミュレーションでのイベント伝播判定



残留X値の伝播シミュレーション(通常シミュレーション)でのイベント伝播判定

<注> ブランクはイベント発生なし

図16 初期シミュレーションでのイベント伝播判定



出力の1はXが正しい

図17 初期値伝播シミュレーション後の予盾例

5. 実設計への適用結果

2万ゲート規模・VLSIの機能設計にFORESTの機能シミュレーションを適用した結果、ゲートレベルシミュレーションに比べ最高20倍，平均10倍の高速性を得た。比較結果を表2に示す。高速結果が得られた要因としては、モデル作成・シミュレーション処理において機能レベルとゲートレベルに対するモデルの複雑度の違いがあげられる。機能レベルとゲートレベルでの記述量は、

機能/ゲートで1/7となっている。また、設計工数は機能設計に33人日，論理設計に140人日である。

表2　機能シミュレーションとゲートシミュレーションの比較

回路規模〔ゲート〕	機能レベル〔SEC〕	ゲートレベル〔SEC〕	ゲートレベル/機能レベル(倍)
80	74.64	145.02	1.94
166	40.32	163.87	4.06
610	50.10	990.49	19.77
1064	80.05	1380.47	17.24
1454	358.54	2012.45	5.61
1854	242.60	3460.96	14.26
14617	1827.64	10078.74	5.51

今回の適用により、現状のLSI設計に機能シミュレータを導入すれば以下の利点が得られることが判った。

①機能シミュレーションを行うことで設計の早い時期にバグを発見できる。

②機能シミュレータを用いることにより、機能設計，論理設計の作業分担が容易に行える。

③機能記述では、論理図作成が不要であるため論理設計工程に至るまで論理の詳細化を行う必要がなく設計工数が短縮される。

④機能記述により、ファンクションテスト用のパタン作成が容易にできる。

6. おわりに

テーブルドリブン方式論理機能シミュレータの処理アルゴリズムを述べるとともに実設計データによる機能シミュレータとしての評価を行った。その結果、複数ビットのレジスタ，ラッチなどのマクロなシミュレーションモデルによる高速な機能シミュレーションが可能となった。

以上の事から、今迄論理設計工程からしかDAツールを利用できなかった設計者は、機能設計という設計の初期段階から効率の良い設計検証，設計詳細化を行えることが判った。

謝辞　本研究を遂行するにあたり、御指導頂いた須藤集積応用研究室長ならびに室員各位に深謝致します。

参考文献

〔1〕和田，星野，須藤，一宮，渡多野"機能記述によるテスト発生システム（FOREST）"昭和56年度信学全大，456

〔2〕樋浦，渡辺，菊池，遠藤，和田，杉浦"FOREST言語（FDL）トランスレータ"同上，457

〔3〕星野，和田，一宮，提"FOREST自動テスト発生プログラム·フェーズI"同上，458

〔4〕菊池，樋浦，渡辺，上田，星野"機能レベルシミュレータFDLSIM"昭和56年度部門別信学全大，S3-3

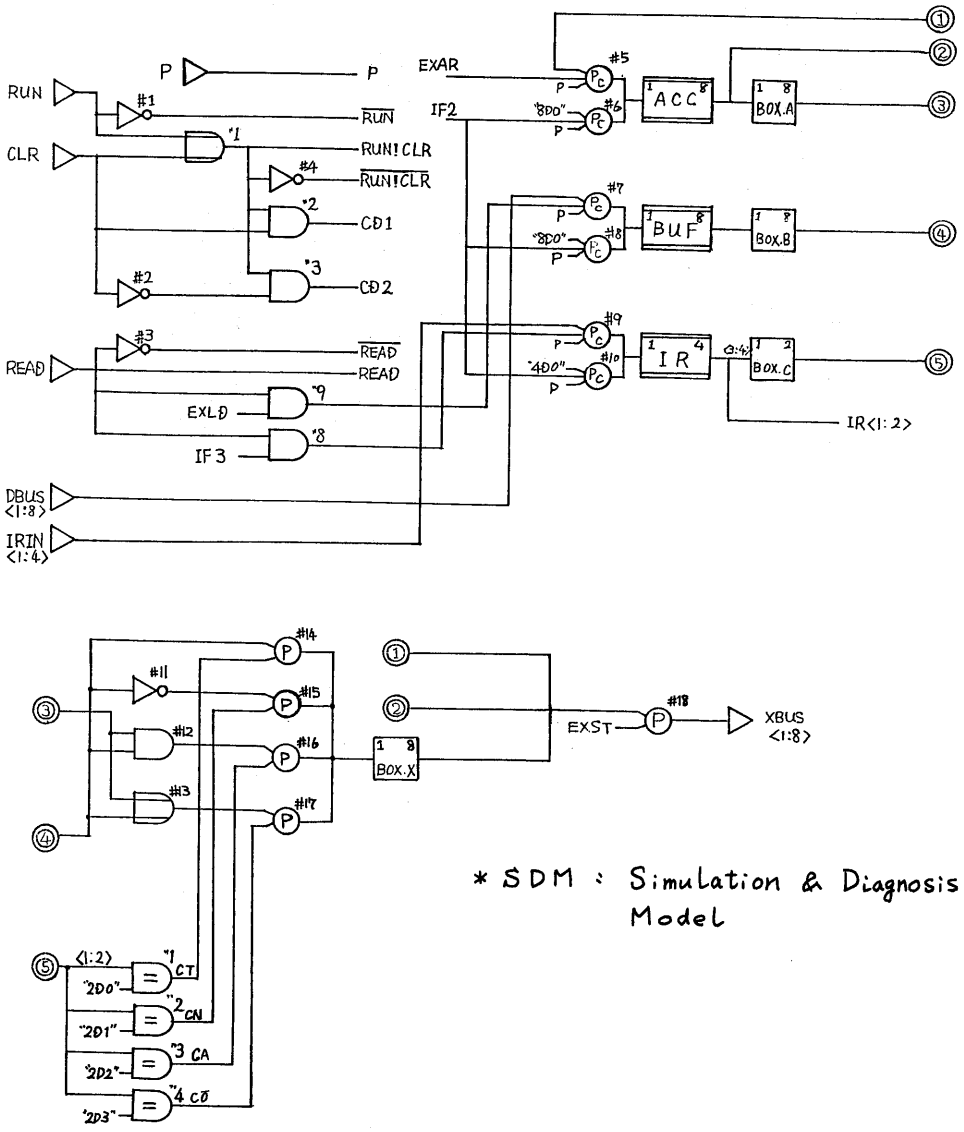
〔5〕唐津，星野，須藤"LSI設計記述処理システム"昭和56年度信学全大S9-2

〔6〕E.W. Thompson, et al "Timing Analysis For Digital Fault Simulation Using Assignable Delays" 11th DA Conf. P266

〔7〕Dan Holt, et al "A MOS/LSI Oriented Logic Simulator" 18th DA Conf. P280

[8] E.W.Thompson, et al " The Incorporation Of Functional Level Element Routines Into An Exsting Digital Simulation System" 17th DA Conf. P394
[9] Melvin A. Breuer " Digital System Design Automation: Languages, Simulation & Data Base " Computer Science Press Inc. '75
[10] S.G.Chappell, et al " LAMP : Logic Circuit Simulators" BSTJ, vol53, NO.8, P1451 1974

付録1 SDM* 構成図(SAMPLE回路の一部)



* SDM : Simulation & Diagnosis Model

付録2　シミュレーション結果（SAMPLE回路）

FOREST　FDLPRI(82-08-30,V=01,L=04)　SIGNAL　DATA　LIST

TIME	1	2345678	9	10	11	12	13	14	15	16	17	18
	P	IIIIEEE FFFFXXX 1234LAS DRT	RULN	CLAD	RERIN	IR	IR	DBUS	ACC	BUF	X	XBUS
0	0	1000000	0	0	0	XXXX	XXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX
1	0	1000000	0	0	0	XXXX	XXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX
2	1	1000000	0	1	0	XXXX	XXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX
3	0	0100000	0	0	0	XXXX	XXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX
4	0	0100000	0	0	0	XXXX	XXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX
5	1	0100000	0	0	0	XXXX	XXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX	XXXXXXXX
6	0	0001000	0	0	0	XXXX	0000	XXXXXXXX	00000000	00000000	00000000	XXXXXXXX
7	0	0001000	0	0	0	XXXX	0000	XXXXXXXX	00000000	00000000	00000000	XXXXXXXX
8	1	0001000	0	0	0	XXXX	0000	XXXXXXXX	00000000	00000000	00000000	XXXXXXXX
9	0	0001000	0	0	0	XXXX	0000	XXXXXXXX	00000000	00000000	00000000	XXXXXXXX
10	0	0001000	0	0	0	XXXX	0000	XXXXXXXX	00000000	00000000	00000000	XXXXXXXX
11	1	0001000	1	0	0	XXXX	0000	XXXXXXXX	00000000	00000000	00000000	XXXXXXXX
12	0	1000000	1	0	0	XXXX	0000	XXXXXXXX	00000000	00000000	00000000	XXXXXXXX
13	0	1000000	1	0	0	XXXX	0000	XXXXXXXX	00000000	00000000	00000000	XXXXXXXX
14	1	1000000	1	0	0	XXXX	0000	XXXXXXXX	00000000	00000000	00000000	XXXXXXXX
15	0	0010000	1	0	0	0100	0000	XXXXXXXX	00000000	00000000	00000000	XXXXXXXX
16	0	0010000	1	0	0	0100	0000	XXXXXXXX	00000000	00000000	00000000	XXXXXXXX
17	1	0010000	1	0	1	0100	0000	XXXXXXXX	00000000	00000000	00000000	XXXXXXXX
18	0	0001000	1	0	1	0100	0100	XXXXXXXX	00000000	00000000	00000000	XXXXXXXX
19	0	0001000	1	0	1	0100	0100	XXXXXXXX	00000000	00000000	00000000	XXXXXXXX
20	1	0001000	1	0	1	0100	0100	XXXXXXXX	00000000	00000000	00000000	XXXXXXXX
21	0	0000100	1	0	1	0100	0100	XXXXXXXX	00000000	00000000	00000000	XXXXXXXX
22	0	0000100	1	0	1	0100	0100	XXXXXXXX	00000000	00000000	00000000	XXXXXXXX
23	1	0000100	1	0	1	0100	0100	00000010	00000000	00000000	00000000	XXXXXXXX
24	0	1000000	1	0	0	0100	0100	00000010	00000000	00000010	00000010	XXXXXXXX
25	0	1000000	1	0	0	0100	0100	00000010	00000000	00000010	00000010	XXXXXXXX
26	1	1000000	1	0	0	0100	0100	00000010	00000000	00000010	00000010	XXXXXXXX
27	0	0010000	1	0	0	1000	0100	00000010	00000000	00000010	00000010	XXXXXXXX
28	0	0010000	1	0	0	1000	0100	00000010	00000000	00000010	00000010	XXXXXXXX
29	1	0010000	1	0	1	1000	0100	00000010	00000000	00000010	00000010	XXXXXXXX
30	0	0001000	1	0	1	1000	1000	00000010	00000000	00000010	00000010	XXXXXXXX
31	0	0001000	1	0	1	1000	1000	00000010	00000000	00000010	00000010	XXXXXXXX
32	1	0001000	1	0	1	1000	1000	00000010	00000000	00000010	00000010	XXXXXXXX
33	0	0000010	1	0	1	1000	1000	00000010	00000000	00000010	00000010	XXXXXXXX
34	0	0000010	1	0	1	1000	1000	00000010	00000000	00000010	00000010	XXXXXXXX
35	1	0000010	1	0	1	1000	1000	00000010	00000000	00000010	00000010	XXXXXXXX
36	0	1000000	1	0	0	1000	1000	00000010	00000010	00000010	00000010	XXXXXXXX
37	0	1000000	1	0	0	1000	1000	00000010	00000010	00000010	00000010	XXXXXXXX
38	1	1000000	1	0	0	1000	1000	00000010	00000010	00000010	00000010	XXXXXXXX
39	0	0010000	1	0	0	1100	1000	00000010	00000010	00000010	00000010	XXXXXXXX
40	0	0010000	1	0	0	1100	1000	00000010	00000010	00000010	00000010	XXXXXXXX
41	1	0010000	1	0	1	1100	1000	00000010	00000010	00000010	00000010	XXXXXXXX
42	0	0001000	1	0	1	1100	1100	00000010	00000010	00000010	00000010	XXXXXXXX
43	0	0001000	1	0	1	1100	1100	00000010	00000010	00000010	00000010	XXXXXXXX
44	1	0001000	1	0	1	1100	1100	00000010	00000010	00000010	00000010	XXXXXXXX
45	0	0000001	1	0	1	1100	1100	00000010	00000010	00000010	00000010	00000010
46	0	0000001	1	0	1	1100	1100	00000010	00000010	00000010	00000010	00000010