設 計 自 動 化 31－1
（1986. 3. 18）

MOS論理回路のスイッチレベルシミュレータの開発

竹之上 典昭 ・ 古賀 義亮
（防衛大学校）

1．はしがき

MOS 論理回路の論理をシミュレーションする方法としてMOS トランジスタをスイッチとして扱うスイッチレベルシミュレーション法がある。スイッチレベルシミュレーション法としてはBryantやHayes の方法が知られているが、これらはいずれもトランジスタ・ノード等にレベル付けを行い回路の方程式を組み立てて解を求めたり、論理値（0，1，U，Z）のそれぞれにレベル付けを行い回路全体でどれが強いかによって値を決定するというものである。そのためシミュレーションが複雑になりシミュレートできる回路のレベルも決定されてしまうことになる。

本論においてはスイッチレベルシミュレーションにおいて回路や論理値（以後値という）に対してレベル付けを行わず、イベントにより値の流れを制御するシミュレーション方法を提案する。またシミュレータへの回路入力のツールとしてMOS論理回路を階層的に記述でき、さらにモジュール化が可能な回路記述言語ＰＡＬＭも同時に提案する。これら回路記述言語およびシミュレーション法の有効性・実用性を確認するためＭＳ－ＤＯＳ上にＰＡＬＭコンパイラとシミュレータを作成したのでこれらの概要について述べる。

2． MOS 論理回路記述

MOS 論理回路記述言語ＰＡＬＭ(PArallel logic simulation description Language for Mos transistor circuits)はネットワーク型コンピュータによる並列MOS 論理シミュレータ用に作成されたものであり、ここで述べるものは一台のコンピュータでシミュレートする場合の縮小版である。しかし、MOS 論理回路の回路記述は特に変更はない。

ＰＡＬＭはMOS 論理回路の構造を記述する言語であり、論理シミュレーションのための回路記述において必要な内容及びＰＡＬＭの記述文法について以下に述べる。

2．1 構造記述

スイッチレベルの論理シミュレーションにおけるMOS 論理回路は、MOS トランジスタを中心として構成されたスイッチネットワークとして捉えられる。そのためトランジスタ間の接続構造を記述する必要がある。

2．2 階層的な回路記述

MOS 論理回路の構造を記述する場合、一度に大きな回路を記述する代りに小さな回路（一つの機能ブロックを構成するような回路）に分割し階層的に記述することができるようにする。階層的な記述は、回路の中の繁雑なトランジスタ端子の接続を小さな回路内の接続に置き換える。また、小さな回路に分割することにより、大きな回路を機能ブロックごとの接続として記述でき、回路内の部分的な変更を容易に行うことができる。

2．3 回路のモジュール化

すでにシミュレーションを終えた回路は、それらをモジュールとして用いて新たな回路を合成することができる。モジュールは内部の機能を保存するため入出力端子を明確に定義し、定義された端子を通してのみ他の回路と接続する。そのため回路記述は互いに独立して記述する。これにより各回路記述はライブラリモジュールとして蓄積することができる。

2．4 制御記述

コンパイラに対してシミュレーションの母体となる回路を指示したり、回路の入出力を定義し、シミュレーションのためのデータファイルやシミュレーション結果を納めるファイルを指定する必要がある。

そのためこれらの情報を制御記述として回路記

表1　PALMの予約語

御記述語	回路記述語		
	定義	部品	値
#entry #inport #outport #data #result #include	circuit line structure end	nmos pmos resistor diode clockgen	false true Vss Vdd

述とあわせて与える。これにより、コンパイルからシミュレーションまでの一連の処理を容易にすることができると共に各種の論理シミュレーション情報を一つのファイルに履歴として残すことができる。

2．5　PALMの文法

以上のような方針のもとに、まずコンパイラが扱うことのできるMOS 論理回路並列論理シミュレーション記述言語ＰＡＬＭの文法を定めた。ＰＡＬＭは制御記述と回路記述から構成される。表１は、ＰＡＬＭの予約語である。

制御記述には、コンパイラがコンパイルを進めていく上で必要とする情報を記述する。#entryはシミュレーションを行う母体となる親回路を示し#inport は親回路のポートの中で入力端子となるものを#outportは出力端子となるものを示している。この３つの制御記述によりシミュレーションを行う対象とその入出力関係を与えることができる。#data はシミュレーションのデータファイル名を示し#result は出力先のデータファイル名を示している。さらに#includeによりコンパイル中に他の回路記述をライブラリとして読み込みコンパイルする機能を持たせている。

回路記述は、図１において定義された構文に基づき個々の回路の構造を定義する。

回路記述はcircuit 文によって行う。circuit 文では回路名及び他の回路と接続するためのポートと回路の本体を定義する。また回路の中で、子回路（部品）を接続する線路を必要に応じてline 文により定義する。回路の本体はstructure 文からend 文の間に定義する。

```
<回路記述> ::= circuit <回路名> ( <ポート名の並び> );
                    <宣言部>
                  structure
                        <回路の本体>
                  end;

<回路名> ::= <名前>

<ポート名の並び> ::= <ポート名>| <ポート名の並び>, <ポート名>
<ポート名> ::= <名前>

<宣言部> ::= <空>| <ライン宣言>
<ライン宣言> ::= line <ライン名の並び>; .
<ライン名の並び> ::= <ライン名>| <ライン名の並び>, <ライン名>
<ライン名> ::= <名前>

<回路の本体> ::= <部品の並び>
<部品の並び> ::= <部品>| <部品の並び>, <部品>
<部品> ::= <部品名> ( <実パラメタの並び> );|
          <部品名> ( <実パラメタの並び> ): <注釈>;
<部品名> ::= nmos|pmos|resistor|diode|clockgen| <回路名>
<実パラメタの並び> ::= <実パラメタ>|
                       <実パラメタの並び>, <実パラメタ>
<実パラメタ> ::= <値>| <ポート名>| <ライン名>

<値> ::= true|false|vdd|vss
```

図1　PALM回路記述についての文法規則

スイッチレベルシミュレーションの基本素子（部品）には、

(1) nmos(NタイプのMOS トランジスタ)

(2) pmos(PタイプのMOS トランジスタ)

(3) resistor（抵抗）

(4) diode(ダイオード)

があり、さらに特殊な組込み素子として

(5) clockgen(クロック発生器)

がある。nmosとpmosの３つのポートの属性は左からgate, drain, source となっている。

ＰＡＬＭで記述された回路は、互いに独立したものとして扱われる。そのため一度に大きな回路を記述しコンパイルすることなく、小さな回路に分けて記述し分割コンパイルすることができる。各回路はポート定義されたポートのみで結合する。

図１の構文からもわかるようにＰＡＬＭでは、回路記述の中にさらに回路記述を行うことはできない。レベルの異なる回路記述を許す利点は、共通の線路を子のレベルの回路中で使用できることであるが、ＰＡＬＭでは各回路接続のためのインターフェースを明確にするためにポート以外の共通線路が生じるような記述はできないようにしてある。

3．シミュレーションの方法

3．1　3ステートイベント法

MOS 論理回路のスイッチレベルシミュレーションはゲートレベルシミュレーションとは異なる特有の難しさがある。それは、シミュレーションの基本素子がAND,OR等のゲートではなくスイッチであることによる。

ゲートには、それ自体に値を演算しデータの流れを一方向に規制する機能がある。しかしスイッチには値を演算しデータの流れを規制する機能はない。MOS 論理回路においてはスイッチネットワークとして動作することにより、データの流れを制御することができる。

この困難さのためMOS 論理回路のスイッチレベルシミュレーションにおいては、トランジスタやノードの状態に強さを設け方程式としてそれを解くことにより状態の推移を求めたり2)、各値にレベルを付けて強弱により、値を決定する3)方法が用いられている。

これらの方法は、回路に一部の修正（追加・削除等）を行うごとに状態の強さや値の強さを変更する必要があり、シミュレータにこのアルゴリズムを搭載する場合には、シュミレーションが複雑になり、高速に回路の論理解析を行うには不向であろう。

本論においては、回路及び値にレベル付けは行わず、トランジスタ（スイッチ）・抵抗・結線・ダイオードを基本素子（タスク）として各タスクの特性に応じた端子（手）の状態（active, waiting, sleeping の３種）を管理することにより、タスクのイベントの発生を判断し、値（真・偽・不定の３値）の流れを制御しシミュレートする方法を用いている。本方式を以後〝３ステートイベント法〟と呼ぶ。

３ステートイベント法における値はゲートレベルシミュレーションで扱う値と同等の概念でありゲートレベルシミュレーションとの親和性はよい。基本素子ダイオードはAND,OR等と性質が同じものであり、ゲートレベルシミュレーションとの親和性を示す一つの例である。

3．2　タスクの構造

MOS 論理回路においてトランジスタ・抵抗等の素子は本来、独立した存在であり、これが互いに接続され回路として構成されている。同様に本方式におけるタスクも互いに独立して処理される。シミュレーションにおいてタスクに必要な情報はタスクの機能及び３本の手（抵抗・ダイオードは２本）に関する情報である。

タスクは図２のような構造をしている。opr はタスクの機能を表し、３本の手には手の状態を示すstat, 値を格納するdata, 属性を示すattrがある。さらにこの属性(joint, gate, source, drain の場合）によって接続情報を格納するjoint_task_number, joint_hand_numberがある。属性には、

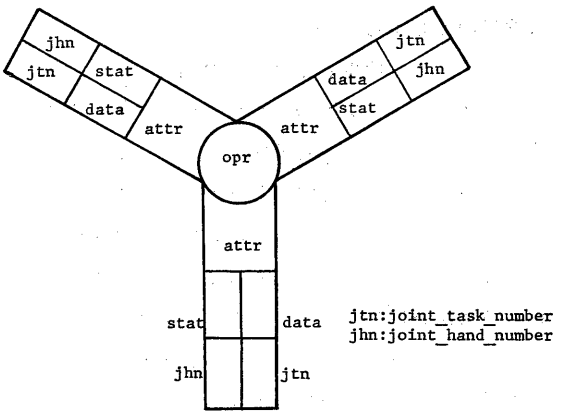


図2　タスクの構造

表2　タスクの機能と手の属性

タスク / 手の属性	NMOSGATE PMOSGATE	CONNECTION	RESISTOR DIODE	CLOCKGEN
JOINT		○	○	○
INPORT	○	○	○	
OUTPORT	○	○	○	
SOURCE	○			
GATE	○			
DRAIN	○			
INGATE	○			
DATAGATE	○			
VALUE	○	○	○	○
NONE			○	

○：手の属性として存在するもの

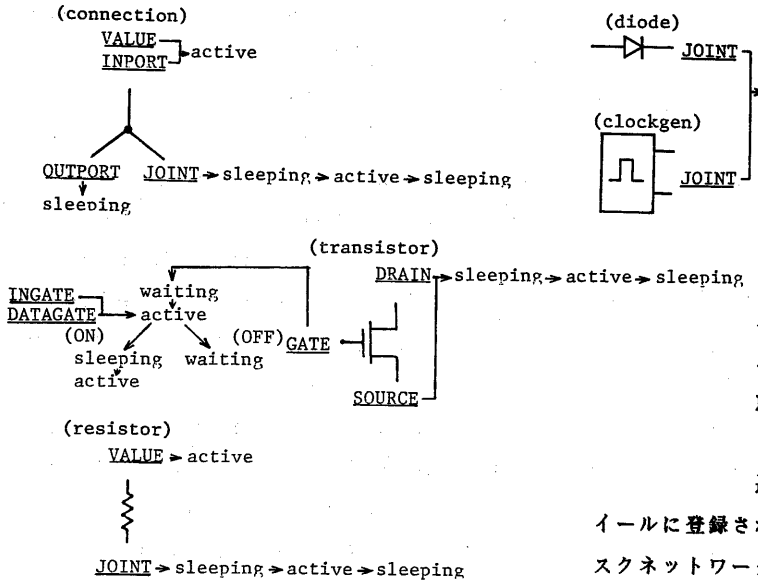


図3　タスクのイベントを決定する手の状態遷移

タスクの機能及び手の役割により表２に示すような種類のものがある。

３．３　手の状態とイベントの発生

３ステートイベント法においては、イベントの発生したタスクのみを処理するイベントドリブン方式を用いている。通常のゲートレベルシミュレーションにおいて、イベントは入力端子に値が変化した時に発生するというものである。しかし、MOS 論理回路におけるイベントは、タスクに対するイベントであり、タスクの手の値が変化したことが直ちにイベントの発生とはならない。

トランジスタの場合gateの値によりスイッチとして動作するため、スイッチがOFF の状態では、sourceやdrain の値が変化してもタスクとしてはイベントが発生したことにはならない。OFF のトランジスタはgateの値が変化したときにイベントが発生したことになる。そのためタスクの手の状態を判断してイベントの発生とする必要がある。手の状態としては以下の３つがある。

(1) active　:値が変化し活性化した状態
(2) waiting　:値の到着を待っている状態
(3) sleeping :値の処理が終り静的な状態

イベントはこの状態を判断し、<u>waiting な手がなくactiveな手が一つ以上ある場合</u>に発生する。イベントの発生したタスクはタイムホイールに登録される。手の状態は、値の変化及びシミュレーションの実行によって図３のように遷移し、シミュレータはタイムホイールに登録されたタスクのみを処理しながらタスクネットワークをシミュレートする。

３．４　タスクの機能と処理方法

シミュレーションにおいて扱う値は真(T) ・偽(F) ・不定(U) の三値である。以下にタスクの機能ごとの動作と処理方法を示す。

(1) NMOSGATE: スイッチ機能(n-mosタイプ)
gateの値により動作を決定する
<gate_value>
T ：スイッチオンの状態となりsource, drain 間は１つの閉路となる
F ：スイッチオフの状態となりsource, drain の両端子に対して不定値を送出する
U ：動作しない

(2) PMOSGATE: スイッチ機能(p-mosタイプ)
gateの値により動作を決定する
<gate_value>
T ：スイッチオフの状態となりsource, drain の両端子に対して不定値を送出する
F ：スイッチオンの状態となりsource, drain 間は１つの閉路となる
U ：動作しない

(3) CONNECTION: 結線の機能
３つの線路の結合点であり、値を伝える機能

(4) RESISTOR: 抵抗の機能

不定値が存在する場合のみ演算を実施、他は値を保持して動作しない

(5) DIODE:ダイオードの機能

値の流れを一方向に制御する

(6) CLOCKGEN: クロック発生器の機能

clockgen(clk1, clk2)とＰＡＬＭにおいて記述され、

clk1 : 1,0,1,0,...

clk2 : 0,1,0,1,...

とクロックを発生する

以上の機能のタスクはタイムホイールのイベント情報により１ユニットタイム（タイムホイールにおける一単位時間）ごとに次に示すアルゴリズムによって処理を行う。

＜タスク処理アルゴリズム＞

(1) タイムホイールからタスク番号を取りだす。

(2) 指示されたタスク番号の機能を取り出しactiveな手の値を被演算値とする。

(3) 被演算値が２つ以上の場合、表３の真理値表に基ずきタスクの値を決定する。１つの場合は被演算値をタスクの値とする。ただし、手の属性がinport,valueである場合はその値をタスクの値とする。

表３　真理値表

	F	T	U
F	F	F	F
T	F	T	T
U	F	T	U

(4) タスクの値と異なる値を有する手に対してタスクの値を送出する。

(5) タイムホイールが空でなければ(1) へ。

(6) 終り。

このようにシミュレータの中におけるタスクは１つ１つが独立した回路構成要素であり、実行順序の制御はイベントの発生のみに依存させる。

またタスクに必要とする情報はすべてタスク中に含んでいる。そのため１つのＣＰＵですべてのタスクを管理する必要はなく、タスクの手によって接続されたタスク間の通信を行うことができれば複数のＣＰＵによる並列処理も可能である。

４．シミュレーションシステム

４．１　シミュレーションの流れ

本シミュレーションシステムは、ＰＡＬＭコンパイラ及びシミュレータＭＯＳＰＬＵＳにより構成され、パーソナルコンピュータのＭＳ－ＤＯＳ上に作成されている。

シミュレーションの流れは図４のようになり、以下のステップで行うことができる。

(1) シミュレートするMOS 論理回路をＰＡＬＭにより記述する。

(2) ＰＡＬＭコンパイラによりコンパイルしシミュレーション回路のオブジェクト（タスクネットワーク情報）ファイルを生成する。

(3) ＭＯＳＰＬＵＳにオブジェクトファイル名を入力する。その後はその中に記述された情報に従い自動的にシミュレーションデータを読み取りシミュレーションを実行して指定されたファイルに結果を出力する。

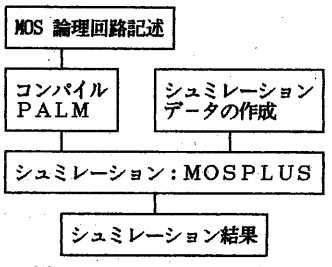


図４　シュミレーションの流れ

４．２　ＰＡＬＭコンパイラ

ＰＡＬＭコンパイラはＰＡＳＣＡＬによって記述されている。そしてMOS 論理回路記述をシミュレーション用のタスクネットワーク情報に変換する役割を果たしており、以下の特徴を持っている。

(1) 階層的に回路記述できる。

(2) 回路記述，中間ファイルの各レベルにおいてモジュール化ができる。

(3) NMOS回路・CMOS回路だけでなくNMOSトランジスタとPMOSトランジスタの混在する回路も記述できる。

(4) 制御記述においてシミュレーションデータファイル及びシミュレーション結果ファイルを指定でき、ファイル管理の一元化を図ることがで

きる。

4．3　シミュレータ

シミュレータMOSPLUSはPALMコンパイラが出力したタスクネットワーク情報をもとにMOS 論理回路のスイッチレベルシミュレーションを行うことができ、PASCALによって記述されている。

MOSPLUSは、元来並列MOS 論理シミュレーション用のシミュレータ1)であるが、ここで述べるMS－DOS上のものは一台のコンピュータ用に変更し、さらに3ステートイベント法を採用しシミュレーションの高速化を図っている。

MOSPLUSは以下の特徴を持つ。

(1) MOS 論理回路をスイッチネットワークとしてシミュレーションできる。

(2) 組合せ回路・順序回路のシミュレーションを行える。

(3) NMOS回路・CMOS回路だけでなくNMOSトランジスタとPMOSトランジスタの混在する回路もシミュレーションできる。（ただし、トランジスタは単なるスイッチとして扱うためワイヤードOR,AND等は扱えない。）

(4) トレースモードにより、シミュレート中の値の流れを知ることができる。

以上の特徴を有するコンパイラ及びシミュレータをパソコン上に構築してあり、軽易に利用できるシステムとなっている。

5．シミュレーション例

本シミュレーションシステムの使用例を以下に示し、PALMによる回路記述の例及びシミュレーション時間について述べる。

5．1　3入力排他的論理和の回路

3入力排他的論理和の回路は、AND,OR,NOTゲートを組合せて設計することができるが、MOS トランジスタレベルで設計すれば図5のように8つのトランジスタで構成することができる。これはゲートを組合せた回路の1/2-1/3 のトランジスタ数であり、MOS トランジスタレベルで設計することの利点といえる。図5のような回路はスイッチレベルでなければ論理シミュレーションを行うことはできない。

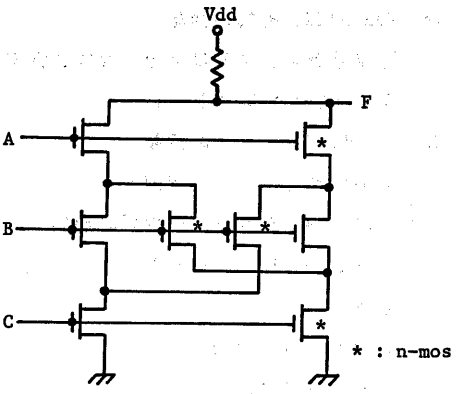


図5　3入力排他的論理和の回路(XOR3)

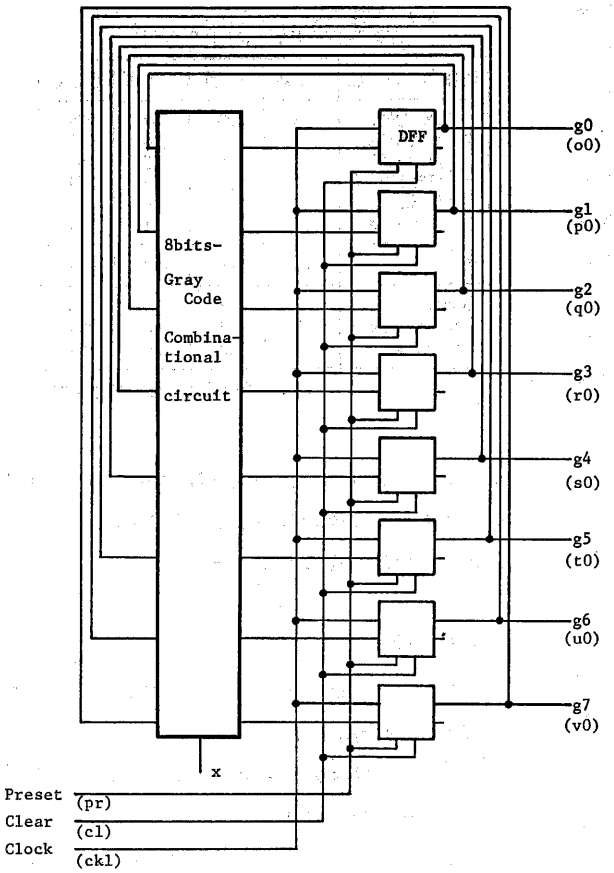
```
#entry xor3
#inport a,b,c
#outport f
#data <value3>
#result <result3>

circuit xor3(a,b,c,f);
        line 11,12,13,14;
        structure
                resistor(Vdd, f);
                pmos(a,  f, 11);
                pmos(b, 11, 14);
                pmos(c, 14, Vss);
                nmos(b, 11, 13);
                nmos(b, 12, 14);
                nmos(a,  f, 12);
                pmos(b, 12, 13);
                nmos(c, 13, Vss);
end;
```

図6　XOR3回路のPALMによる記述例

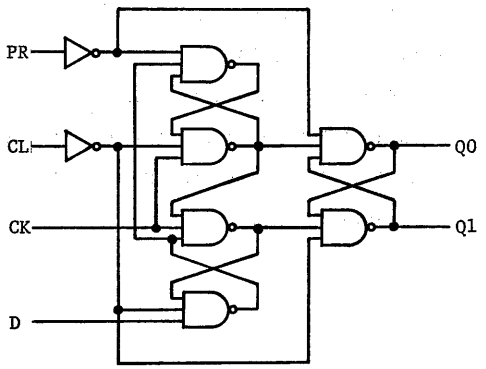
```
## Simulation circuit file name : EXOR3.S ##
 >>> input | output  <<<
 0 0 0 |  0
 0 0 1 |  1
 0 1 0 |  1
 0 1 1 |  0
 1 0 0 |  1
 1 0 1 |  0
 1 1 0 |  0
 1 1 1 |  1
```

図7　XOR3回路のシュミレーション結果

これをPALMで記述したものが図6である。この記述においては、回路はxor3という名で定義され、入出力の端子としてa,b,c,f を、内部の線路として11,12,13,14 を定義している。回路の構



A. 8ビットグレイコードカウンタ



B. Dフリップフロップ

図8　順序回路例

造はstructure からend の間に記述してある。nmosとpmosの端子は左からgate,drain,source となっている。

```
#entry gcc8
#inport ckl,pr,cl
#outport o0,p0,q0,r0,s0,t0,u0,v0,x
         o1,p1,q1,r1,s1,t1,u1,v1
#data <value14>
#result <result14>

circuit gcc8(ckl,pr,cl,
             o0,o1,p0,p1,q0,q1,r0,r1,s0,s1,
             t0,t1,u0,u1,v0,v1,x);
  line f41,f42,f43,f45,f46,f44,f47,f48;

  structure  gc8(o0,p0,q0,r0,s0,t0,u0,v0,
                 f41,f42,f43,f44,f45,f46,
                 f47,f48,x);
             dff2(ckl,f41,pr,cl,o0,o1);
             dff2(ckl,f42,pr,cl,p0,p1);
             dff2(ckl,f43,pr,cl,q0,q1);
             dff2(ckl,f44,pr,cl,r0,r1);
             dff2(ckl,f45,pr,cl,s0,s1);
             dff2(ckl,f46,pr,cl,t0,t1);
             dff2(ckl,f47,pr,cl,u0,u1);
             dff2(ckl,f48,pr,cl,v0,v1);

end;


circuit dff2(ck, d, pr, cl, q0, q1);
        line g1,g2,l1,l2,l3,l4;
        structure
                 not(pr, g1);
                 not(cl, g2);
                 nand3(g1, l4, l2, l1);
                 nand3(g2, l1, ck, l2);
                 nand3(ck, l2, l4, l3);
                 nand3(d, l3, g2, l4);
                 nand3(g1, l2, q1, q0);
                 nand3(q0, g2, l3, q1);
end;
```

図9　階層的なPALMの記述例
（8ビットグレイコードカウンタ）

制御記述では、コンパイラ及びシミュレータに対する指示を与える。ここでは、「a,b,c を入力端子 fを出力端子としてxor3の回路をvalue3のファイルのデータによりシミュレートして、result3のファイルに出力せよ」という指示が与えられたことになる。

この指示に従いシミュレータは図7に示す出力結果を得ることができる。

5．2　順序回路のシミュレーション

本システムでは図8-Aに示す順序回路もシミュレートできる。図8-Aは8ビットカウンタの回路であり、組合せ回路部（トランジスタ179個）にDフリップフロップを接続した構成となっている。図8-Bは、図8-Aで使用しているDフリップ

フロップの回路である。これは、図９のように階層的に記述することができる。そのため回路の一部の変更を行う場合、例えばＤフリップフロップの内部構造を変更する場合にはＤフリップフロップの記述部分だけを変更すればよいことになる。またこの回路をnmos回路にすることもcmos回路にすることもライブラリモジュールを変更するだけで簡単に行なうことができる。

ＰＡＬＭコンパイラは、この記述によって回路全体をトランジスタの回路に合成し、その接続情報（タスクネットワーク情報）を作成する。シミュレータＭＯＳＰＬＵＳはこれを読み込みスイッチレベルのシミュレーションを行う。

５．３　処理時間

本システムにおける回路のコンパイル及びシミュレーション時間を表４に示す。これはＣＰＵに80286(8MHz) のパーソナルコンピュータにおいてラムディスクを使用した場合の結果である。この表から小規模のMOS 論理回路は、コンパイルは数秒から数十秒、シミュレーションは数秒から数分のオーダで行うことができ、シミュレーション結果から回路の変更そしてまたシミュレーションという繰り返しを軽易に行うことのできるシステムとなっている。

６．　あとがき

MOS 論理回路のスイッチレベルシミュレーションについて回路記述言語及び３ステートイベント法によるスイッチレベルシミュレーション方式を提案し、そのコンパイラ及びシミュレータを開発した。

スイッチレベルシミュレーションはこれまで回路シミュレーションにたよっていたトランジスタレベルの設計をより軽易なものとすることができる。MOS 論理回路の論理シミュレーションをインタラクティブに行うことのできる本システムは、回路設計者の強力なツールとなるであろう。

本システムはいまだ試作段階にあり、コンパイラのエラーメッセージの充実等改良すべき点が多々存在する。今後の課題としては故障シミュレーションへの応用を考えている。

表４　回路の構成とシミュレーション時間
{CPU:80286(8MHz) RAM-DISK使用}

回路名：組／順		A：組	B：順	C：順	D：順
タスク数	NMOSGATE	4	20	47	333
	PMOSGATE	4	0	24	6
	RESISTOR	1	8	22	155
	CONNECTION	11	18	86	379
	DIODE	0	0	0	0
	CLOCKGEN	0	0	1	0
	合計	20	46	180	873
Compile (sec)		2	3	7	18
Simulate (sec)		1	2	43	204
データ数		8	15	128	256

組：組合せ回路／順：順序回路

参考文献

1)　竹之上，古賀：ネットワーク型コンピュータによるMOS 論理回路シミュレーションの一方法について，情報処理学会論文誌，Vol.26，No.3，pp.420-428　(1985)

2)　Bryant　R. E.: A Switch-Level Model and Simulator for MOS Digital Systems,IEEE TRANSACTIONS ON COMPUTERS, VOL.C-33, NO.2, pp.160-177 (1984)

3)　Kawai, M. and Hayes, J.P. : An Experimental MOS Fault Simulation Program CSASIM, IEEE 21th DAC, Paper 2.1, pp.2-9 (1984)