設計自動化　40－7
(1987. 12. 17)

回路モジュールのタイミングチャートを利用した論理シミュレータ

LOGIC SIMULATOR USING TIMING CHARTS OF MODULE CIRCUITS

井下　順功†　　橋爪　正樹††　　為貞　建臣†
Toshinori INOSHITA　　Masaki HASHIZUME　　Takeomi TAMESADA

†徳島大学工学部　　††徳島大学工業短期大学部
Tokushima University　　Technical College of Tokushima University

あらまし　マイクロプロセッサ応用回路のような回路モジュールを用いて設計される回路に適した論理シミュレータについて述べる．本シミュレータでは，各回路モジュールに対してその動作記述と入力制約条件の他に，各動作モードごとにタイミングチャートで与えらる入出力動作を使用する．これらの入出力動作によって，自動的に入力パターンを生成し，入出力動作と等価なシミュレーション結果が得られるかを検証する．もし同じでなければその誤りを指摘する．本シミュレータはシミュレーション時の使用者の労力，特に入力パターンの生成，結果の評価の労力を軽減することができる．

Abstract　This paper presents a mix-level simulator which is suitable for circuits made of VLSI, like the microprocessor based circuit. In this simulator, it is used the behaviour descriptions, input constraints, and the input-output operations of the module circuit, which are normally given in the timing chart for each operation mode. Only by means of those operations, this simulator generates input patterns automatically, and checks whether the simulated results are the same as the required input-output operations. This simulator can deduce human work for simulations, especially in the input pattern generation and evaluation of simulated results.

1．はじめに

回路のVLSI化に伴い設計回路が複雑化し設計誤りの検出が非常に困難となっている．特にLSI設計では論理設計時の設計誤りをなくすために論理シミュレータが不可欠となっている．その一方で，設計回路の大規模化に伴い，シミュレーションすべき回路規模が増大している．そのため，高速なシミュレータが必要とされており，シミュレーション高速化のためのさまざまなアプローチがとられている[1,2,3,4]．

シミュレーションしたい回路が大規模になると，高速なシミュレータが必要となるだけでなく，シミュレータへの入力パターンの作成，その入力パターンを入力した時に得られるべきシミュレーション結果の導出およびシミュレーション結果から設計誤りの検出が困難となる．それを解決する1つの方法として，回路の行うべき動作をシミュレータに与え，シミュレーション結果がそれと同じになるかどうかを調べるシミュレータが提案されている[5]．

その一方で，1チップマイクロコンピュータのように前もって設計され，その動作が予め規定されている大規模な汎用回路モジュールを用いた設計が行われることが多くなった．その場合，回路モジュール間の回路が新たに設計される．その回

路の設計誤りを検出するためには，(1)回路全体をシミュレーションする方法，(2)回路モジュール間の回路だけをシミュレーションする方法が考えられるが，回路規模の小さいことから方法(2)が好ましい．

しかし，方法(2)でも回路モジュールが大規模回路となると，入力パターンの決定が困難となるだけでなく，得られたシミュレーション結果が正しいものかを調べることも非常に困難となる．そのため回路モジュールを基に設計された回路を効率よくシミュレーションできるシミュレータを開発する必要がある．そこで我々は各回路モジュールに，各動作モードごとに提示されているタイミングチャートを持たせ，その動作モードを指定するだけで，シミュレータへの入力パターンの作成，シミュレーション結果の評価が行えるシミュレータを開発した．

２．では回路モジュールに持たせた回路データおよびそれを利用したシミュレーション法を，３．では開発したシミュレータについて述べる．

２．回路モジュールのタイミングチャートを利用した論理シミュレーション法

2.1 シミュレーション法

本シミュレーション法は，１チップマイクロコンピュータのような，大規模で汎用の回路モジュールを用いて設計された回路の設計誤り検出を行うためのものである．一般にそのような回路モジュールは，動作モードごとにタイミングチャートでその入出力動作が提示されている．

回路モジュールを用いた設計では図１に示すように回路モジュール間の回路(CIR1,CIR2)だけが新たに設計される．そのため設計回路の正当性を調べる場合，CIR1,CIR2を調べればよい．

それらの回路を調べる方法として，(1)個別に調べる方法と，(2)入力側回路モジュールと出力側回路モジュールの入出力動作を用いて調べる方法が考えられる．方法(1)は，MOD1,MOD2の動作を調べ，入力パターンの作成および得られるべきシミュレーション結果の予想を行い，設計誤りを検出する方法である．一方，方法(2)はその回路の入力側回路モジュールのある動作モード時に得られるデータをその回路に与えた時の出力が，出力側回路モジュールのある動作モード時の動作と矛盾しないかどうか調べる方法である．ここで対象とする回路モジュールは汎用であるため，いちいち入力パターンの作成などの労力を使わずに，方法(2)で検証したほうが好ましい．そこで，本手法では各回路モジュールに各動作モードごとに示されているタイミングチャートを持たせ，シミュレーションを行う．

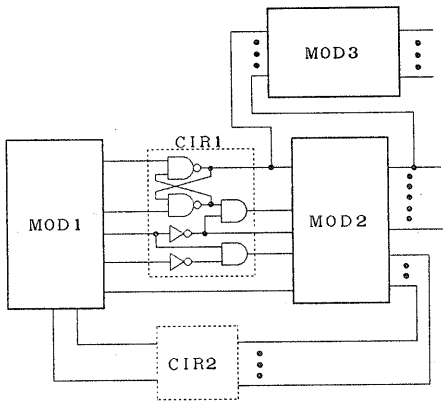


図１　回路モジュールを用いて設計された回路

タイミングチャートには外部回路とのデータのやり取りの順序だけでなく，その動作を行うために最低限満たされなくてはならない条件が示されている．たとえばZ80のメモリリード動作時のタイミングチャートには図２に示すように，「出力信号MREQはＴ1ステートのクロックの立ち上がりから100nsec遅れて立ち上がり，Ｔ3ステートのクロックの立ち下がりから100nsec遅れて立ち上がる」という変化パターンが定義されている．また入力信号D0-7に対しては，「Ｔ3ステートのクロックの立ち上がりより60nsec前に安定し，RDの立ち上がり

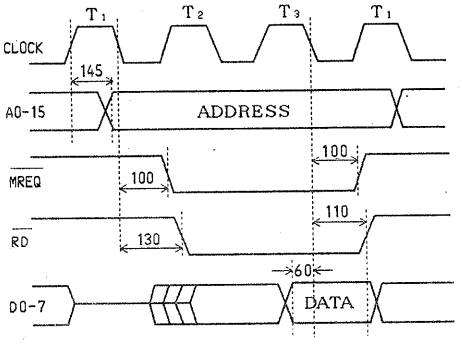


図２　Ｚ８０メモリリード時のタイミングチャート

までホールドされていなければならない」という条件が示されている．

したがって，これらのタイミングチャートで規定されている回路動作は各動作モード時の入力パターンとしても，また得られるべきシミュレーション結果としても利用することができる．そこで各回路モジュールの入出力動作としてタイミングチャートで規定されている動作（ここでは「動作シーケンス」と呼ぶ）をシミュレーションに使用する．

さらに回路モジュールには各動作モードを実行するための条件（ここでは「起動条件」と呼ぶ）を持たせておく．そうすれば，その動作モードに入れるかどうか調べることができ，設計誤り検出を行うことができる．

シミュレーション時には調べたい回路の入力側回路モジュールの動作モードと出力側回路モジュールの動作を指定する．シミュレータは入力側回路モジュールの動作シーケンスを基に入力パターンを発生させ，シミュレーションを行う．その時に得られる回路の出力値は出力側回路モジュールに伝え，それが指定した動作モードの(1)起動条件を満足するかどうか，また(2)動作シーケンスを満たすシミュレーション結果が得られたかを，シミュレータがチェックする．もしそれらのいずれかが満足されなければ，回路内に設計誤りが存在することになるため，シミュレータに設計誤りの存在を提示させる．

出力側回路モジュールが動作シーケンスを持たない場合は動作記述を用いてシミュレーションを行う．入力側回路モジュールが動作シーケンスをもたない場合，使用者が入力パターンを与えてシミュレーションを行う．入力側，出力側の両方の回路モジュールとも動作シーケンスを持たない場合，従来のシミュレーション法と同様な方法でシミュレーションを行う．

2.2 本シミュレーション法の能力

本手法は調べたい回路の入力側回路モジュール

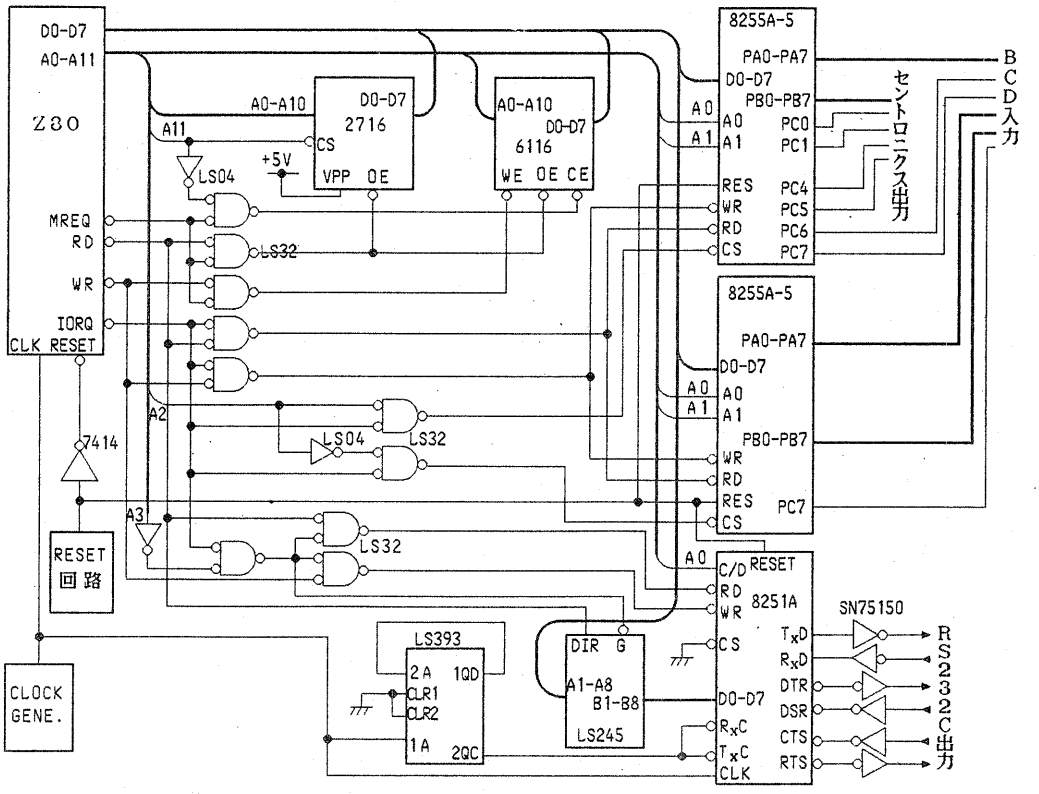


図3　シミュレーション回路例（回路1）

の動作シーケンス，出力側回路モジュールの動作シーケンスとその起動条件で回路の正当性を調べる方法である．本手法が有用となる回路中の回路モジュールは回路規模が大きく，動作記述が複雑となる順序回路である．したがって，タイミングチャートを表した動作シーケンスは回路モジュールの複雑な状態遷移図の１つのパスに対応している．また，動作シーケンスにはそのパス中の各状態で図２に示すような満たされなければならないタイミング条件も格納されている．そのため，動作シーケンスを用いたシミュレーションでは，その回路モジュールの出力側回路に対し，信号を出力もしくは入力し，指定した動作モード時にたどるべき状態遷移図上のパスをタイミング条件を満たしながらたどれるかどうかを調べている．さらに回路の出力側回路モジュールの動作シーケンスを指定することにより，出力側回路モジュールも同様にそのパスをタイミング条件を満たしながら忠実にたどれるような信号値が得られるかどうかを調べている．そのため，本手法で非常にきびしいチェックが行えると同時に，場合によっては図１におけるMOD1の動作シーケンスでCIR1だけでなく，CIR2まで調べることができる．

しかし，本手法を用いて２つの回路モジュール(MOD1,MOD2)間だけの動作モードを指定して調べるだけでは不十分で，回路全体をシミュレーションしないと調べられない場合がある．たとえば図１の回路をシミュレーションする場合，MOD1とMOD2，MOD1とMOD3の関係を調べて正しくても，MOD1,MOD2,MOD3を同時に調べないと設計誤りが検出できない場合が存在する．その場合はシミュレーション時に，同時に調べるべき回路モジュールとその動作モードを使用者が適切に選択しなければならない．

また本シミュレーション法はタイミングは自動的に検証することができるが，論理は使用者が判断しなければならない．たとえば図２のタイミングチャートではD0-7の値に対し何も制約条件がないため，本手法では論理の正当性は自動的に検証できない．しかし，図４に示すように8255A-5のWRの立ち上がりに対して8255A-5のCSのホールド時間が10nsecしかない図３の回路を，本シミュレーション法でシミュレーションすれば，8255の書き込み動作シーケンス中に記述されている「8255A-5はWRの立ち上がりに対してCSのホールド時間は最低20nsec必要」という条件により回路の設計誤りを自動検出することができる．

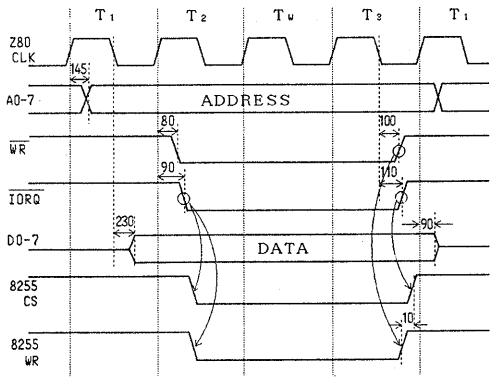


図4　回路１の Z80 IO-WRITE モード時のタイミングチャート

本手法は上記の欠点をもつが，使用者の労力を大幅に減少させることができる．特に，入力パターン作成時に誤りが入ることが多く，また使用者の調べたい動作かどうかもわからない場合がある．しかし，本手法は動作モードの指定だけで入力パターンが生成できるため，入力パターン生成に要する労力を減らすことができる．さらに正しいシミュレーション結果が得られているかは，指定した動作シーケンスと同じ動作をするかを調べることで行えるし，また起動条件を満たすかどうかでも回路の正当性を調べることができるため，シミュレーション結果の評価に要する労力も減らすことができる．

また我々は図１の回路をシミュレーションする場合，回路モジュールの動作モードごとに回路動作を調べる．そのため，動作モードを指定するだけでシミュレーションが行える本手法を用いれば，人間にとって回路の設計誤り検出が容易に行えるシミュレータの開発が期待できる．

さらに回路全体の動作は各回路モジュールの各基本動作から構成されている．したがって，回路全体が正常に動作するためには各回路モジュールの基本動作は最低限調べなければならない事項である．また，検証の初期段階で回路全体をシミュレーションするのは，設計誤りの発見が困難となるため，回路の各部分の正しさを検証した後で，回路全体をシミュレーションする方が設計誤り発見の効率がよい．そのため，本シミュレーション法は検証の初期段階で非常に有効な手法となる．

3．開発した論理シミュレーションシステム

システムの構成を図５に示す．本システムは，回路データを登録しておく「回路データベース」と，そのデータを登録するための「回路エディタ」，シミュレーションを行う「論理シミュレータ」からなる．なお，回路エディタについては，現在専用の回路エディタを開発中であり，現状はテキストエディタを使用している．

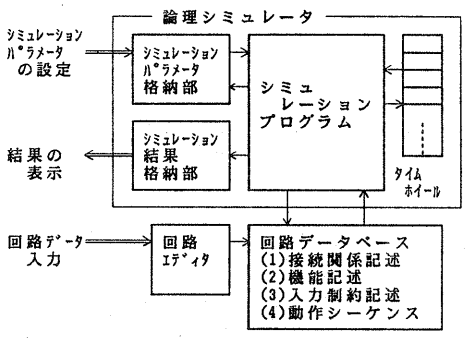


図５　システム構成

3.1 回路データベース

回路データベースには，表１に示す回路モジュールに関する回路データを格納しておく．各回路データはスロット，ファセットとその値の組で知識を表現するフレーム[7]を用いて表現する．

図６にZ80の記述の一部を示す．図６に示すように，入力ポートはINPUTスロットに，出力ポートはOUTPUTスロットに，双方向性ポートはIN/OUTスロットに記述する．また，各入力ポート名はそのポートの種類が名前からわかるように，入力ポート名には「IN-」を，出力ポート名には「OUT-」を，双方向性ポートには「IO-」をポート名の前につけて区別する．

シミュレーション時に各素子の遅延時間を変更できるようにするため，遅延時間はすべて変数を用いて記述する．その変数にデフォルト値を割り当てるために，TIMEスロットには遅延時間をその変数と遅延時間の組で格納する．

入力信号が満足しなければならない条件はCONSTRAINTスロット，もしくは動作シーケンスを記述するSEQUENCEスロットに格納する．その条件の種類として，パルス制約，値制約と安定制約[8]を用意した．CONSTRAINTスロットには動作モードに関

表１　回路記述

スロット名	記述内容
INPUT	入力ポート
OUTPUT	出力ポート
IN/OUT	双方向性ポート
TIME	信号の伝搬遅延時間，セットアップ時間 ホールド時間
CONSTRAINT	パルス制約，値制約
CPARA	動作記述，動作ｼｰｹﾝｽ記述で使用する変数
BEHAVIOUR	動作記述　（安定制約）
SEQUENCE	動作モードごとにタイミングチャートを表現したもの　（安定制約）
COMPONENT	構成素子名とその素子の種類
CONNECTION	構成素子間の接続関係

係なく満たされなければならない条件を格納し，動作モード特有の条件はSEQUENCEスロット内に記述する．安定制約は基準となるタイミングを必要とするため，動作記述，動作シーケンス記述内でのみ記述する．たとえば，クロックに関する条件は動作モードに無関係に一定であるため，CONSTRANTスロットに記述する．一方，メモリ読みだしモード時の条件は，SEQUENCEスロット中に記述する．またその回路モジュールを用いた回路で経験的によく設計誤りが生じるものを検出するための条件も，このSEQUENCEスロットに格納することができる．

CPARAスロットでは動作記述，動作シーケンス記

```
ﾌﾚｰﾑ  ｽﾛｯﾄ        ﾌｧｾｯﾄ      値
Z80   INPUT       VALUE   IN-CLOCK IN-RESET IN-INT IN-NMI ･･･
      OUTPUT      VALUE   OUT-A*16 OUT-RD OUT-M1 OUT-IORQ ･･･
      IN/OUT      VALUE   IO-D*8
      TIME        VALUE   (TDLM1 130) (TDHM1 130) ･････
      CONSTRAINT  VALUE   (<LWIDTH> IN-CLOCK 180 -- 2000 )
                          (<HWIDTH> IN-CLOCK 180 -- 200000)
      CPARA       VALUE   (T-STATE 1 2 3 4 W)
      SEQUENCE    VALUE

        (MEMORY-READ
          <START>   / IN-CLOCK = UP & T-STATE = 1 /
          <CONSTANT> OUT-IORQ = 1 OUT-WR = 1
                     OUT-BUSAK = 1 OUT-HALT = 1 <ENDC>
           / IN-CLOCK = UP & T-STATE = 1 /
             OUT-A*16 <- ADDRESS?  <DELAY> TDAD  :
           / IN-CLOCK = DOWN & T-STATE = 1 /
             OUT-MREQ  <- 0 <DELAY> TDLCOMR :
             OUT-RD    <- 0 <DELAY> TDLCORD :
             T-STATE   <- 2 :
           / IN-CLOCK = DOWN & T-STATE = 2 & IN-WAIT = 1 /
             T-STATE <- 3 :
           / IN-CLOCK = DOWN & T-STATE = 3 /
             OUT-MREQ  <- 1 <DELAY> TDHCOMR :
             OUT-RD    <- 1 <DELAY> TDHCORD :
            <STABLE> IO-D*8 - TSCOD -- TDHC1RD <ENDCON>
          <END> )
              ････････････････
```

図６　回路記述例１（Z80CPU）

述で使用する変数と取り得る値を定義する．

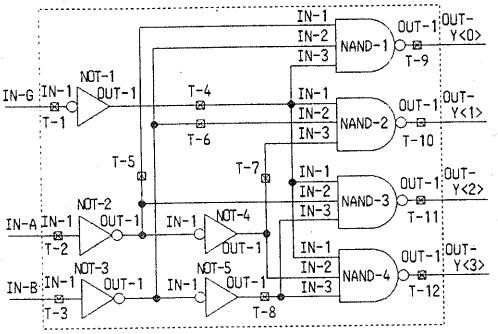
回路モジュールの動作記述はBEHAVIORスロットに，動作シーケンスはSEQUENCEスロットに格納する．条件式は図６に示すように"/"で囲んで表現する．起動条件は<START>の後に記述する．このシーケンス中に一定値をとる信号値に関しては出力するポート名とその値を<CONSTANT>の後に書き込む．具体的な動作シーケンスはそれ以降に記述する．

図７に2TO4DEMPXの回路記述例を示す．いくつかの内部素子から構成される回路モジュールとして記述する場合，図７に示すようにCOMPONENTスロットに構成素子名とその素子の種類を記述する．回路の接続関係は「ポート」，「モジュール」，「ターミナル」[8]で記述し，CONNECTIONスロットに格納する．

3.2 論理シミュレータ

開発した論理シミュレータはミックスレベルシミュレータ[4,6]でセレクティブトレース法に基づきシミュレーションを行う．使用した遅延モデルは標準遅延モデルで，各素子ごとに遅延時間を割り当てることができる．シミュレーションで使用する論理値は，(0,1,U,Z,UP,DOWN)の６値で，その他に記号値も扱える．記号値を扱えるようにした理由はタイミングチャートでは図２のアドレス信号のように論理値が具体的に定められていないこと，記号値はバス幅に無関係に１つの記号で表せるため，回路モジュールに関する記述に汎用性をもたせることができること，さらに記号でシミュレーション結果が得られれば使用者にとって論理の誤り検出が容易に行える[9]ためである．ただし，記号値は制御信号での使用は禁止している．制御信号に記号を許すとすると，動作記述およびシミュレーション結果の評価が困難となるので，制御信号線に記号値が到達した時点で，シミュレーションを一時中断させ，値の設定を行った後でシミュレーションを再開する．信号値として記号を持つことができるかどうかは，陽に宣言せず，動作記述，動作シーケンス記述を記号で定義している場合に使用できる．

また，WHY機能も用意しており，指定したターミナルがその値を出力するに至った過程をターミナル値の履歴をさかのぼりながら表示したり，指定した回路モジュールの動作シーケンス中の各ステップの変化も表示することができる．



(a) 回路図

```
スロット      ファセット     値
INPUT         VALUE     IN-G IN-A IN-B
OUTPUT        VALUE     OUT-Y*4
TIME          VALUE     (TDE 21 32) (TDS 25 38)
COMPONENT     VALUE     (NOT-1 LS04) (NOT-2 LS04) ･･･
                        (NAND-1 LS10) (NAND-2 LS10) ･･･
BEHAVIOUR     VALUE
                <CASE>
                  / IN-G = 1 /
                    OUT-Y*4 <- B1111 <DELAY> TDE :
                  ･･･････････････････････
                  / IN-G = 0 & IN-A = 1 & IN-B = 1 /
                    OUT-Y*4 <- B0111 <DELAY> TDS :
                <ENDCASE>
CONNECTION    VALUE
        (INPORT (T-1 IN-G) (T-2 IN-A) (T-3 IN-B))
        (OUTPORT (T-9 OUT-Y<0>) (T-10 OUT-Y<1>)
                 (T-11 OUT-Y<2>) (T-12 OUT-Y<3>) )
        (NOT-1 (T-1 IN-1) (T-4 OUT-1))
        ･･･････････････････････
        (NAND-4 (T-4 IN-1) (T-7 IN-2)
                (T-8 IN-3) (T-12 OUT-1))
        (T-1 (NOT-1 IN-1))
        ･･･････････････････････
        (T-7 (NOT-4 OUT-1) (NAND-2 IN-3) (NAND-4 IN-2))
        ･･･････････････････････
        (T-12 (OUTPORT OUT-Y<3>) (NAND-4 OUT-1))
```

(b) 回路表現

図７　回路記述例２ (2 TO 4 DEMPX)

開発したシミュレータによるシミュレーション過程を図８に示す．はじめにシミュレーションに必要な各パラメータの設定を行う．まずシミュレーション回路名を入力し，シミュレーション終了時に値の変化を表示したいターミナル名を入力する．ここでシミュレータはシミュレーションレベルを自動的に設定し，シミュレーションに必要な回路データを読み込む．次に，各素子の遅延時間を決定する．ここでは標準値以外の値でシミュレーションする場合に入力する．標準遅延時間が規定されていない場合は与えられている遅延時間を表示し，それを基に遅延時間を使用者に指定させる．その後，タイムホイールの１スロットの時間

幅，タイムホイールの長さを入力する．

次に入力パターンの作成を行う．その方法として，動作シーケンスを用いる方法と，波形記述言語で記述する方法がある．はじめにシミュレータが動作シーケンスをもつ回路モジュールを検索し，なければ波形記述言語で入力パターンを作成する．もし存在すれば，その回路モジュール名をメニュー形式で提示し，回路モジュールを指定する．ここでは複数個指定することができる．シミュレータはその回路モジュールがもつ動作シーケンスをメニュー形式で提示し，ここで動作シーケンスを選択する．

動作シーケンスで出力されない信号が存在する．通常動作オペレーションで定義されない信号はそのシミュレーションでは必要としないため，入力する必要はない．しかし，複数個の回路モジュールを同時に調べる必要がある場合，指定する必要が生じる．その場合はここで入力することができる．

次にシミュレータ出力監視条件を設定する．その方法として動作シーケンスを用いる方法とブレイクポイントを使用する方法がある．動作シーケンスを使用する場合は，上記の動作シーケンス選択と同様な方法で選択する．

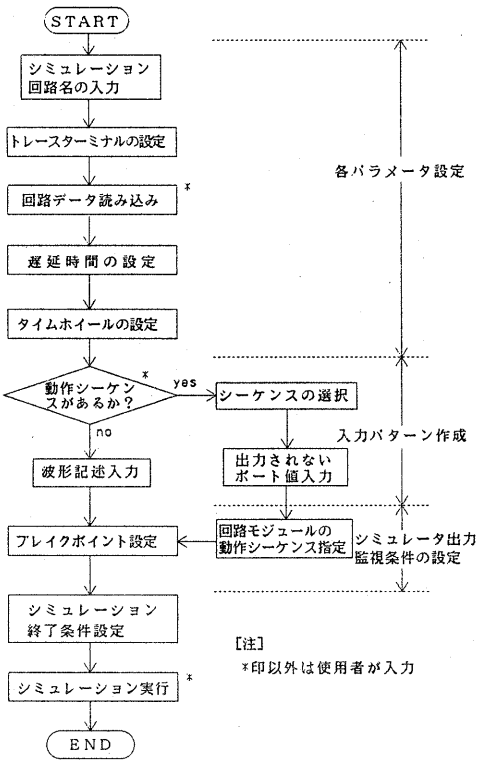


図8　シミュレーション過程

最後に，シミュレーション終了条件を設定し，シミュレーションを開始する．シミュレーション中，シミュレータは指定された動作モードの起動条件を調べ，その動作モードに入るかどうかを監視する．もし異なった動作モードに入ると警告を出し，使用者に知らせる．

シミュレーション終了時には各回路モジュールの現在の状態を表示する．それにより正しい動作を行ったか調べることができる．また，シミュレーション過程の任意の時点で指定したターミナルの値を画面上に表示することができる．またWHY機能によりその値変化，回路モジュールの状態変化を知ることができる．さらに，次に調べたい動作モードを指定すれば，続けてその動作モードをシミュレーションすることができる．

ある動作モード実行中に他の回路モジュールの動作モードが終了する場合がある．その場合は終了した時点で使用者に終了したことを示し，現時点の回路状態で次に実行可能なモードを提示し，次にどの動作モードを調べるか指定することができる．また使用者が回路モジュールの動作モードの選択を誤って行った場合，本シミュレータでは自動的に検証できないが，起動条件およびシミュレーションの途中過程の表示により使用者が容易に検出できる．

なお本システムはパソコン(NEC製PC-9800)上のLISPでインプリメントしている．

3.3 シミュレーション例

図９に本システムによる図３の回路のシミュレーション過程を示す．この回路には，動作シーケンスを持つ回路モジュールがZ80,8255A-5,8251Aの３種類４個ある．図９には，Z80と8255A-5間の回路の検証を示す．使用者は，Z80のIO-WRITEと8255A-5のINITIALIZEを選択し，8255A-5の初期化を検証する．この回路には2.2で述べたようにタイミング上の誤りがあり，これを図９に示すように自動的に検出することができる．

上記のシミュレーションは約３分で行える．またこの規模の回路であれば大量のメモリを持たないパソコンでもシミュレーションが行える．

```
回路に与える入力パターンを入力して下さい.
入力パターンの与え方に次の方法があります.
   1 ··· 波形記述で与える
   2 ··· 動作シーケンスを利用する
   どちらにしますか?  >>> 2
BCDTOCEN には動作シーケンスを持つ
          次の回路モジュールがあります.
   1 ··· BCDTOCEN の Z80CPU
   2 ··· BCDTOCEN の SIO
   3 ··· BCDTOCEN の PPI-1
   4 ··· BCDTOCEN の PPI-2
  どの回路モジュールの動作シーケンスを
          入力パターンとして使用しますか?
        >>> 1
回路モジュール BCDTOCEN の Z80CPU
    のシーケンスとして次のものがあります.
   A ··· MEMORY-READ
   B ··· MEMORY-WRITE
   C ··· IO-READ
   D ··· IO-WRITE
        :
  どのシーケンスを使用しますか?
        >>> D
        ⋮ 中略
スロット数:2530
[CAUTION]
BCDTOCEN の PPI-1 の IN-CS がホールド条件を満たしません.
2510 スロットから今までホールドしていなければいけません.
しかし, 2520 スロットで変化しています.
  ^C ···[終了]  ^B ···[中断]  リターン···[続行]
```

図9 シミュレーション過程(一部)

4. おわりに

本研究では大規模で動作記述が非常に複雑な汎用の回路モジュールを用いて設計された回路に対し, 動作記述の他にタイミングチャートを表した動作シーケンスを格納し, シミュレーションに利用するシミュレーション法およびシミュレータを開発した. そして本シミュレータをマイクロプロセッサ応用回路に適用した. その結果, 本シミュレータは使用者の多大な労力を必要とせずに設計誤りを容易に検出できることを確認した. 今後は他の回路に対して本シミュレータを適用し, 本シミュレータの能力をより明確にする予定である.

[謝辞] 本研究に関し, 京都大学工学部矢島脩三教授, 平石裕実助教授をはじめ矢島研究室の皆様に御討論していただきました. ここに深謝いたします.

[参考文献]

(1) 小池誠彦:"シミュレーションエンジン",電子情報通信学会誌,Vol.70,No.7,pp.728-737,(1987).

(2) L.Gindrcux and G.Gaalin: "CAE Station' Simulators Tackle 1 Million Gates", Electronic Design, Vol.31, No.23, pp.127-136, Nov. (1983).

(3) N.Ishiura, H.Yasuura, T.Kawata and S.Yajima:"High-Speed Logic Simulation on a Vector Processor",Proc. of ICCAD-85, 4C.2, pp.119-121,(1985).

(4) T.Sasaki, A.Yamada, S.Kato, T.Nakazawa, K.Tomita and N.Nomizu :"MIXS : A Mixed Level Simulator for Large Digital System Logic Verification",Proc. of 17th Design Conference, pp.626-633, June (1980).

(5) N.S.Woo:"A Prolog Based Verifier for the Functional Correctness of Logic Circuits", Proc. of ICCD-85, pp.203-207, Oct.(1985).

(6) 安浦寛人, 蚊野浩, 大井康, 木村晋二, 石浦菜岐佐, 矢島脩三:"入力制約監視機能を持つ会話型シミュレーションシステム ISS", 情報処理学会論文誌, Vol.25, No.2, pp.285-292,(1984).

(7) 上野晴樹: "知識工学入門", オーム社, pp.88-108,(1985).

(8) R.Davis and H.Shrobe: "Representing Structure and Behavior of Digital Hardware",COMPUTER,pp.75-82,(1983).

(9) 斉藤隆夫,上原貴夫: "論理回路の記号シミュレーション",情報処理,pp.7-16,Jan.(1985).

(10) Zilog 1983 Data Book, Zilog Inc.,(1983).