電子装置設計技術９－３
（１９８１．６．９）

機能記述によるテスト発生手法

星野民夫　和田康　須藤常太
（日本電信電話公社　武蔵野電気通信研究所）

１．はじめに

集積回路技術の進歩により（チップ内に、集積されるゲート規模は、年々増加している。それに伴いＬＳＩの論理シミュレーションとテストパターン発生に必要とされる計算時間は、加速度的に増加している。このような状況を打破する試みとして、機能記述によるテスト発生システム（ＦＯＲＥＳＴ：Functional block ORiEnted Simulation and Test generation system）を開発した[1][2]。本論では主にそのテスト発生手法について述べている。ＦＯＲＥＳＴでは、新だなテストパターン発生手法として、機能ブロック活性化法を導入している。

２．特長

図１に処理フローを示す。以下にＦＯＲＥＳＴの特徴を述べる。

（１）テストパターン発生に、機能ブロック活性化法を用いている。機能ブロック活性化法は、大きなシステムを幾つかのブロックに分割し、ブロックごとにテストパターンを発生する方法である。ブロックの活性化シーケンスの発生は、機能レベル記述をもとに行なうのでブロックの入出力へテ

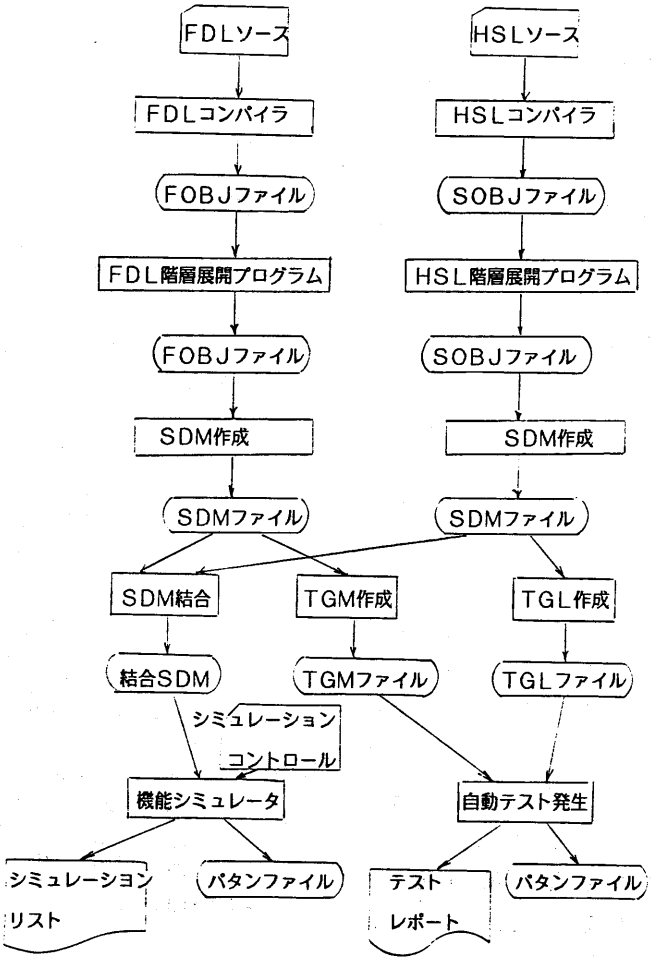


図1　FORESTシステム処理フロー

ストパターンを入出力する径路を効率良く発生できる。
（２）機能ブロックのテストパターン発生は、対応するブロックのゲートレベルの記述を用いて行なう。そのパターンは、機能レベルで発生したブロック活性化シーケンスと結合編集を行なってテストパターンとしているため、効率の良いテストパターン発生が可能である。
（３）発生したテストパターンのタイミングチエック等を行なうために、機能レベルとゲートレベルの混合シミュレーションを行なう事が可能である。
（４）機能レベルの設計時から、テスト設計が可能なため、テスト容易な回路設計が行なえる。
（５）全システムを階層的に記述できる。そのため大規模なシステムの設計をトップダウン、ボトムアップ的に効率よく行なえる。
（６）システムの記述はモジュール単位で行ないモジュールは機能レベルの記述及びゲートレベルの記述のどちらでも記述可能であり、両者を混在させた形でシステムの記述とすることが可能である。
（７）機能記述言語としては、ＦＤＬ[3]（Forest Description Language）を開発し用いている。

ＦＤＬはＤＤＬと同様にオートマトンを用いた状態遷移の表現、if－then－else, case文等を用いて効率よくシステム動作が記述できる他、システムの階層記述も可能である。図２にＡＭ２９０１の記述例を示す。
（８）ゲートレベルの記述言語としては、ＨＳＬ[4]

```
FOREST    FDL  (80-03-03,V=01,L=01)
 STMT              SOURCE  STATEMENT
   1  USER    :TESTABILITY.GROUP ;
   2  IDENT   :AM2901 ;
   3  VERSION:V02 ;
   4  DATE    :80/06/13 ;
   5  AUTHOR  :K.OCTOPUS ;
   6  COMMENT:******* 4-BIT SLICE RALU (AM2900-SERIES) ******* ;

FOREST    FDL  (80-03-03,V=01,L=01)
 STMT              SOURCE  STATEMENT
   1  NAME    :AM2901 ;
   2  PURPOSE:TPG ;
   3  LEVEL   :CHIP ;
   4  <EXT>    AADD(3:0),BADD(3:0),DDAT(3:0),CN,OEBR,
               IADD(8:4),IN(5:3),ISOD(2:0),
               GBAR,PBAR,CNP4,OVR,F0,F3,Y(3:0),
               RAM0,RAM3,Q0,Q3,CLK ;
   5  <INP>    .IADD,.IN,.ISOD,.AADD,.BADD,.DDAT,.CN,.OEBR ;
   6  <OUT>    .GBAR,.PBAR,.CNP4,.OVR,.F0,.F3,.Y ;
   7  <BUS>    .RAM0,.RAM3,.Q0,.Q3 ;
   8  <CLK>    .CLK ;
   9  <REG>    QREG(3:0) ;
  10  <MEM>    RAM1(1:16,0:3),RAM2(1:16,0:3) ;
  11  <LAT>    ALAT(3:0),BLAT(3:0) ;
  12  <TYP>    ALU((5:3),CN,R(3:0),S(3:0),F(3:0),ST(5:0));
  13  ALU   :  BOX ;
  14  <END> ;
  15  <BOO>    BOX.CN:=.CN,.GBAR:=BOX.ST(1),.PBAR:=BOX.ST(4),
               .CNP4:=BOX.ST(2),.OVR:=BOX.ST(3),
               BOX.I:=.IN ;
  16  <TER>    RMEN:QEN ;
  17  <TER>    QIN(3:0),RMIN(3:0),OUT(3:0) ;
  18  <BOO>    ?.ISOD  #0 BOX.R:=ALAT,BOX.S:=QREG
                       #1 BOX.R:=ALAT,BOX.S:=BLAT
                       #2 BOX.R:=4B0 ,BOX.S:=QREG
                       #3 BOX.R:=4B0 ,BOX.S:=BLAT
                       #4 BOX.R:=4D0 ,BOX.S:=ALAT
                       #5 BOX.R:=.DDAT,BOX.S:=ALAT
                       #6 BOX.R:=.DDAT,BOX.S:=QREG
                       #7 BOX.R:=.DDAT,BOX.S:=4B0 ; ,
               ?.IADD  #0 QIN:=BOX.F,OUT:=BOX.F
                       #1 OUT:=BOX.F
                       #2 RMIN:=BOX.F,OUT:=ALAT
                       #3 RMIN:=BOX.F,OUT:=BOX.F
                       #4 RMIN:=.RAM3||BOX.F(3:1),.RAM0:=BOX.F(0),
                          QIN:=.Q3||QREG(3:1),.Q0:=QREG(0),OUT:=BOX.F
                       #5 RMIN:=.RAM3||BOX.F(3:1),.RAM0:=BOX.F(0),
                                       .Q0:=QREG(0),OUT:=BOX.F
                       #6 RMIN:=BOX.F(2:0)||.RAM0,.RAM3:=BOX.F(3),
                          QIN:=QREG(2:0)||.Q0,.Q3:=QREG(3),OUT:=BOX.F
                       #7 RMIN:=BOX.F(2:0)||.RAM0,.RAM3:=BOX.F(3),.Q3:=QREG(3),
                          OUT:=BOX.F ; ;
  19  <BOO> RMEN:= ?.IADD #0 1D0
                          #1 1D0 X1D1 ; ,
                QEN := ?.IADD #0 1D1
                              #4 1D1
                              #6 1D1 X1D0 ; ;
  20  <BOO> .F0:=BOX.ST(5),
               .F3:=BOX.ST(0) ;
  21  <AUT> DYNM : .CLK ;

FOREST    FDL  (80-03-03,V=01,L=01)
 STMT              SOURCE  STATEMENT
  22        <TER>  ADR(3:0) ;
  23        <COP>    ?.OEBR? .Y:=OUT ; ;
  24        <STA> ;
  25        HIGH   : ALAT<-RAM1(ADR) ,ADR:=.AADD ,
                     BLAT<-RAM2(.BADD) ;
  26         LOW   : ?RMEN? RAM2(.BADD)<-RMIN,RAM1(ADR)<-RMIN,ADR:=.BADD ; ;
  27        RISE   : ?QEN?  QREG<-QIN ; ;
  28        <END> ;
  29  <END> DYNM ;
  30  END   : AM2901 ;
```

図2　AM2901のFDL記述例

（Ｈierarchical　Ｓpecification　Ｌanguage ）を用いている。

３．機能ブロック活性化法

自動テストパターン発生や故障シミュレーションに必要となる計算機使用時間は、ゲート数の２乗から３乗に比例する。それゆえ、大規模な回路についてテストパターンを発生する場合は、その全回路を１つの回路ブロックとして扱う事は得策でない。

機能ブロック活性化法は、大規模な回路を幾つかの機能ブロックに分割して、機能ブロック毎に活性化してテストパタンを発生する方法である。機能ブロックごとに分割してテストパタンを発生する利点を以下に述べる。

（１）テスト発生に必要となる計算時間を減少できる。

ブロックごとに分割してテストパタンを発生することにより、計算機使用時間の増加をゲート数に比例する程度に押えることができる。今１つの回路ブロックのテスト発生時間を平均Ｔ１とし、全回路はＮのブロックからなるとする。又、機能ブロック活性化のための計算機使用時間（α）とすれば、分割した場合の計算機使用時間Ｔｓ＝Ｔ１　Ｘ　Ｎ　＋　α　である。一方分割しない場合の計算機使用時間はＴｔ＝Ｔ１　ｘ　$N^{2\sim3}$ となる。

（２）機能ブロックに適した方法でテストパターンが発生できる。

組み合せ回路のテストパターンは故障検出率が高くて短いパターン長のテストパターンを発生するアルゴリズムがある。全回路がたとえ順序回路であっても、分割した機能ブロックが組み合せ回路であれば、このようなアルゴリズムを用いてテストパターンを発生できる。また機能ブロックがメモリであればマーチングパターン等のメモリに適したテストパターンを用いることができる。

４．テストパターン発生手法

テストパターンの発生は、（１）機能ブロックの活性化探索、（２）機能ブロック内のテストパターン発生、（３）テストパターンの連結編集、からなっている。

４．１　機能ブロックの活性化探索

探索手法は大別して２つの方法が用いられている。１つは、テストシーケンスの伝播／活性化探索であり、他の１つは、１クロック内の組み合せ回路内での伝播／活性化探索である。

４．１．１　　テストシーケンスの探索

テストシーケンスの探索法は状態探索法[5]を用いている。順序回路の状態は、レジスタ、ラッチ、入力端子の状態によって、一意に決定される。この回路状態は、制御信号とクロックの入力により次ぎの状態へ遷移する。これは1つの状態から他の状態に変換する作様素（Operator）とみなすことができる。探索の初期状態として、機能ブロックの入力に入力パターンが入力され、出力に出力パターンが観察されている状態である。探索の最終状態としては、これらのパターンが、システムの外部端子まですべて伝播、又は活性化された状態が考えられる。テストシーケンスを求めることは、初期状態を最終状態に変換する一連の作様素列を求めることに帰着する。初期状態から到達可能な状態空間を、状態に相当する接点（node）を含むグラフとして考える。この接点の状態は、その時点での回路のレジスタや入力端子の信号値の集合で表現される。ノードは初期問題ノードから探索によって拡張していく。この場合ノードの拡張順序が、探索の効率に大きな影響を与える。図3に状態探索の流れ図を示す。図4にシーケンスノードの拡張過

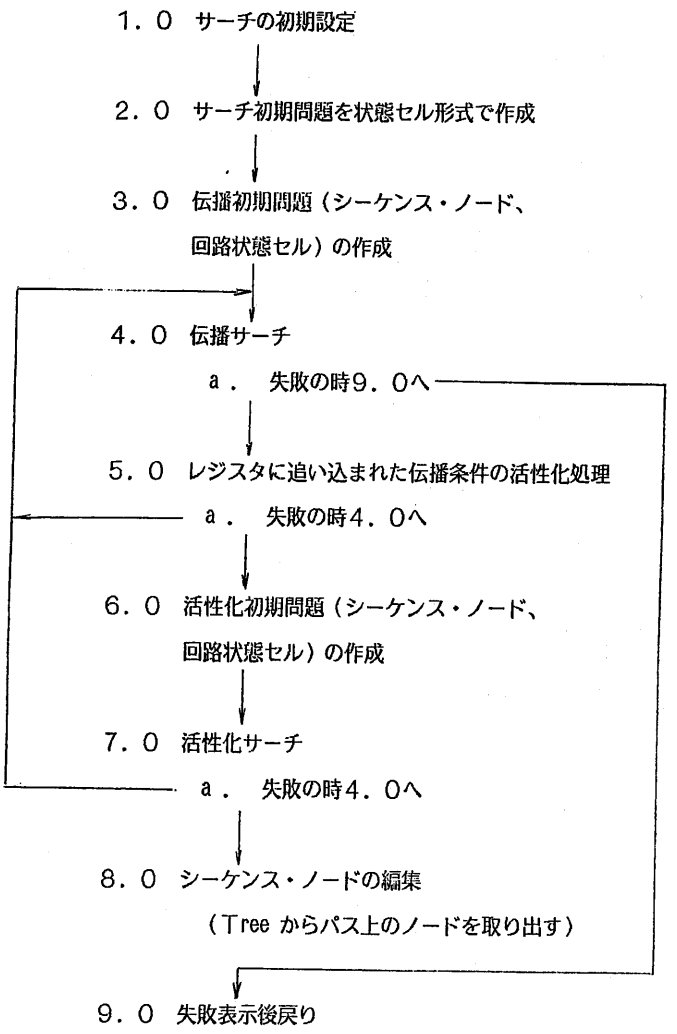


図3 状態探索の流れ図

程を示す。図中、アルファベットの順序は、ノード拡張順序を示している。（）内の数字はノードのweightを示す。拡張するノードの決定は、以下の順序で行なう。

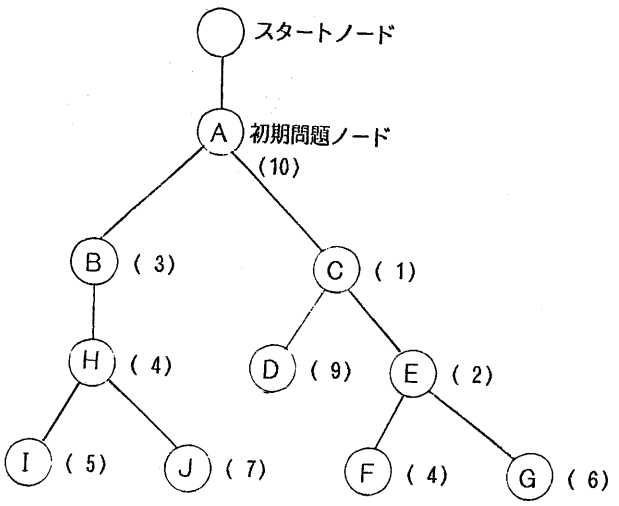


図4　シーケンスノード拡張過程

（１）伝播／活性化初期問題ノードを拡張対象Ｔreeの親ノードとする。（初期問題ノードが未定の時はスタートノードを親ノードとする。）

（２）伝播対象Ｔree 内の端末ノードをすべて候補として選びだす。

（３）端末ノード内でもっともweightの値の小さいものを選びだし拡張用ノードとする。

ノードのweightの計算は以下の式で計算している。

weight＝（ノードの深さ）Ｘ（深さの重み）＋
　全内部ファシリティ
　　Σ　｛（外部入出力からの距離）　Ｘ（ビット種類の重み）Ｘ
　　（ビットの重み）Ｘ（１－seq 達成率・達成率重み／１００００）｝／

　　Ｍax（１，指定プライオリティ）

ここで、ビットの重みは、ビット数の多いファシリティに高い優先順位を与える為のものである。seq 達成率は探索の完了度についての重みである。このようなweightを探索に導入することにより効率の良い探索を可能としている。

４．１．２　クロック内の探索

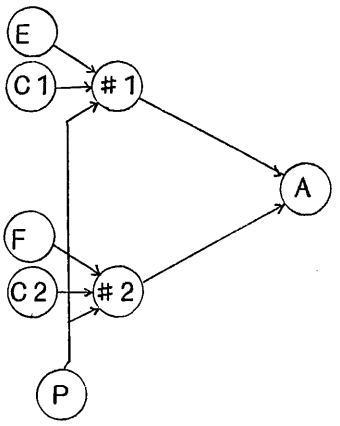
１クロック内の探索は、状態探索の作様素（Ｏperator ）の探索であり、この探索の結果、シーケンスノードが１つ拡張される。一般には、レジスタ間

の信号伝播を探索するものでありる。図5にＦＤＬによる記述と内部回路モデル、及びそのときの探索グラフを示す。ＦＤＬ記述において最初の？はif次ぎの？はthenである。この内部モデルとしてはパスが生成される（図中の＃１、＃２）。

FDL記述

```
?C1?  A<-E  ;
?C2?  A<-F  ;
```

内部表現



探索グラフ

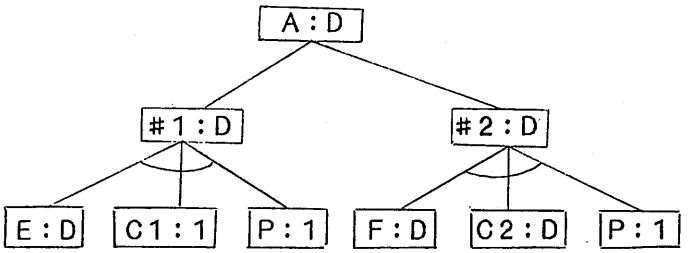


図5　AND−ORグラフを用いた探索

１クロック内の探索法は問題置換探索法[5),(6)]を用いている。図５でレジスタａに信号Ｄをセットすることを考えてみる。これは、パス＃１又はパス＃２に信号Ｄをセットするという問題に置き換えられる。パス＃１に信号Ｄをセットするには、ターミナルＥに信号Ｄをセットし、コンデションＣ１の信号値を１にし、かつ、クロックＰの信号値を１にするという問題にを解く必要がある。このように、１つの問題を次々と小さな問題に分解してゆき、解を求める方法が問題置換探索法である。一つ一つの問題は図に示すように一つのノードとして現わすことが出来る。ノードには、そのしたのすべての問を解く必要のあるノード（ＡＮＤノード）と、

どれか一つの解を求めれば良いノード（ＯＲノード）がある。このような手法を用いることにより、機能レベルの探索を容易にしている。

４．２　機能ブロック内のテストパターン発生

機能ブロックとしては、大別して（１）ランダム・ロジックの組み合せ回路、（２）メモリ、（３）レジスタ、ターミナル等がある。（１）組み合せ回路については、ランダム・パターンを故障シミュレーションする方法、及び４．１．２の問題置換探索法を用いてテストパターンを発生している。（２）のメモリについては、すべてのワードについて１と０をそれぞれ書いて読むパターンを自動発生する。（３）のレジスタについてはall １、all ０のパターンを用いている。

４．３　テストパターンの結合編集

４．１及び４．２で求めたテストパターンを結合編集して一つのテストパターンとしている。この処理により、はじめてテスタに入力出来る形式となる。

５　テストパターン発生例

図６に示す４ビット・スライス。マイクロプロセッサについてテストパターンを求めた。そのＦＤＬ記述を図３にしめした。図７に機能ブロックＡＬＵの活性化シーケンス発生例を、図８に機能ブロック内のテストパターン発生例を示す。

６．　おわりに

以上、機能記述を用いたテスト発生手法について述べた。機能記述を用いることにより複数ビットのパラレル活性化、機能ブロックの活性化を可能とした。また、ブロック内はゲートレベルのテストパターンを発生し、故障検出の詳細化をはかるとともに、効率の良いテストパターン発生を行なうことを可能としている。ＦＯＲＥＳＴはＦＯＲＴＲＡＮ，ＰＬ／Ｉ，及びアセンブラを用いて作成した。

謝辞　本研究を遂行するにあたり終始ご指導をいただいた渡辺誠集積回路研究部長に深謝します。またプログラムの作成に協力いただいた富士通（株）の方々並びに関係各位に感謝します。

```
TEST SEQ. # =     1
                  EXTERNAL PIN = CLK     (  1) : SIG = <1>

TEST SEQ. # =     2
                  EXTERNAL PIN = CLK     (  1) : SIG = <1>

TEST SEQ. # =     3
                  EXTERNAL PIN = CLK     (  1) : SIG = <1>
                  EXTERNAL PIN = IADD    (  6) : SIG = <0>
                  EXTERNAL PIN = IADD    (  7) : SIG = <0>
                  EXTERNAL PIN = IADD    (  8) : SIG = <0>
                  EXTERNAL PIN = IN      (  3) : SIG = <0>
                  EXTERNAL PIN = IN      (  4) : SIG = <0>
                  EXTERNAL PIN = IN      (  5) : SIG = <0>
                  EXTERNAL PIN = DDAT    (  3) : SIG = <D>     D-ORIGIN = BOX   S    (  3)
                  EXTERNAL PIN = DDAT    (  2) : SIG = <D>     D-ORIGIN = BOX   S    (  2)
                  EXTERNAL PIN = DDAT    (  1) : SIG = <D>     D-ORIGIN = BOX   S    (  1)
                  EXTERNAL PIN = DDAT    (  0) : SIG = <D>     D-ORIGIN = BOX   S    (  0)
                  EXTERNAL PIN = ISOD    (  0) : SIG = <1>
                  EXTERNAL PIN = ISOD    (  1) : SIG = <1>
                  EXTERNAL PIN = ISOD    (  2) : SIG = <1>

TEST SEQ. # =     4
                  EXTERNAL PIN = IN      (  5) : SIG = <D>     D-ORIGIN = BOX   I    (  5)
                  EXTERNAL PIN = IN      (  4) : SIG = <D>     D-ORIGIN = BOX   I    (  4)
                  EXTERNAL PIN = IN      (  3) : SIG = <D>     D-ORIGIN = BOX   I    (  3)
                  EXTERNAL PIN = CN      (  1) : SIG = <D>     D-ORIGIN = BOX   CN   (  1)
                  EXTERNAL PIN = DDAT    (  3) : SIG = <D>     D-ORIGIN = BOX   R    (  3)
                  EXTERNAL PIN = DDAT    (  2) : SIG = <D>     D-ORIGIN = BOX   R    (  2)
                  EXTERNAL PIN = DDAT    (  1) : SIG = <D>     D-ORIGIN = BOX   R    (  1)
                  EXTERNAL PIN = DDAT    (  0) : SIG = <D>     D-ORIGIN = BOX   R    (  0)
                  EXTERNAL PIN = ISOD    (  0) : SIG = <0>
                  EXTERNAL PIN = ISOD    (  1) : SIG = <1>
                  EXTERNAL PIN = ISOD    (  2) : SIG = <1>
                  EXTERNAL PIN = OEBR    (  1) : SIG = <1>
                  EXTERNAL PIN = IADD    (  8) : SIG = <0>
                  EXTERNAL PIN = IADD    (  7) : SIG = <0>
                  EXTERNAL PIN = IADD    (  6) : SIG = <0>
                  EXTERNAL PIN = F3      (  1) : SIG = <D>     D-ORIGIN = BOX   ST   (  0)
                  EXTERNAL PIN = GBAR    (  1) : SIG = <D>     D-ORIGIN = BOX   ST   (  1)
                  EXTERNAL PIN = PBAR    (  1) : SIG = <D>     D-ORIGIN = BOX   ST   (  4)
                  EXTERNAL PIN = Y       (  3) : SIG = <D>     D-ORIGIN = BOX   F    (  3)
                  EXTERNAL PIN = Y       (  2) : SIG = <D>     D-ORIGIN = BOX   F    (  2)
                  EXTERNAL PIN = Y       (  1) : SIG = <D>     D-ORIGIN = BOX   F    (  1)
                  EXTERNAL PIN = Y       (  0) : SIG = <D>     D-ORIGIN = BOX   F    (  0)
```

図7　ＡＬＵ活性化シーケンス発生例

```
111                    0XX 00
11110                000X1XX 11
1110011100 110100X0XX 00
1001010111 011101X1XX 11
0111010111 000100X1XX 10
0001001011 001111X1XX 11
0011110001 011001X0XX 11
0001111001 000011X0XX 00
1000110010 011101X1XX 11
0101110001 010111X0XX 00
0101101010 101110X1XX 01
0001110101 010011X0XX 00
0010010011 001001X0XX 01
0101111001 001000X0XX 01
0010111111 101110X1XX 11
  FOREST      ATG(81-02-25,V=02,L=01)     <<<  TEST STATISTICS  :

FBT      : NO.=     1 NAME =      BOX
                      TYPE =      BLOCK
                                  (TGL) MODULE NAME = ALU
                      TOTAL PIN/BIT NO. =.   22
                                  INPUT   =   12
                                  OUTPUT  =   10
                                  BUS     =    0
          SEQUENCE: NAME     = SEQ00002   GROUP NO. =     2
                    LENGTH   =        4   ( SENS. =       4 )
                    DETECTION RATE =  90 (%)

          PATTERN : LENGTH   =        38
```

図8　ブロック内のテスト発生例

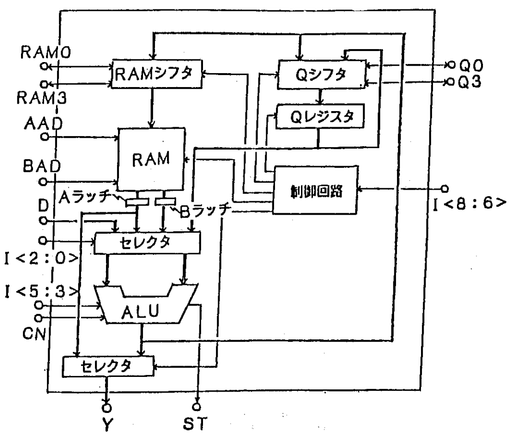


図6　ＡＭ２９０１の回路構成図

文献

（１）和田、星野、須藤、一宮、波多野、「機能記述によるテスト発生システム（ＦＯＲＥＳＴ）」昭和５６年度電子通信学会総合全国大会、４５６

（２）星野、和田、一宮、堤、「ＦＯＲＥＳＴ自動テスト発生プログラム・フェーズⅠ」　同上、４５８

（３）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

（４）唐津、星野、須藤、「ＬＳＩ設計記述処理システム」、同上、Ｓ９－２

（５）Ｎ．，Ｊ．，Nilson，“Problem Solving Methods in Artificial Inteligence”，Mc Grow－Hill，１９７１

（６）Ben M. Heuy，“Guiding Sensitization Search using Problem Reduction Graphs”，15th DA Conf．ｐ３１２