7U-7

ラップトップPC上のテストデータ作成支援ツールTDA

岩脇 邦夫　　鷹木 正幸　　上田 正二
(株)東芝

1 はじめに

ＬＳＩの製造技術は年々向上しており、それに伴って開発するＬＳＩも益々大規模化してきている。この大規模ＬＳＩを開発するには、機能／論理設計の段階で十分にシミュレーションを行うことが必要不可欠なものとなっているが、入力となるテストデータも大量に作成しなければならず、この作成が開発工期の中で大きな比重を占めるようになってきた。そこで、この大量のテストデータを容易に作成するために、本システムＴＤＡ(Test Data Assembler) を開発したので報告する。

本システムの特長は

1) ＳＰ(Structured Programming)マクロを使用することにより、少量の入力で大量のテストデータが作成できる。
2) 入力は従来の波形によるものではなくコードで行う。
3) 作成したテストデータをタイミングチャートで確認し、必要ならば修正することもできる。
4) ラップトップＰＣ上で使用する事ができる。

2 システム構成

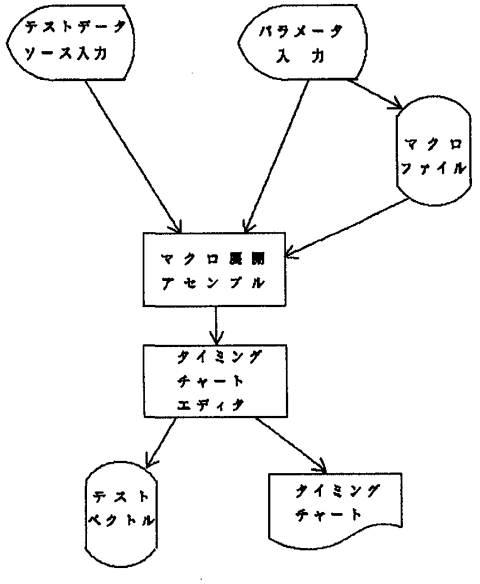

図１　ＴＤＡシステム構成

3 パラメータの入力

3．1 テストベクトルに対する信号名の定義

テストベクトルのフィールドに対し信号名（ノード名）を定義する。信号名は１ビットずつ定義する必要がなく、設計者に分かりやすいようにバス信号やコントロール信号単位で定義できる。また、定義した信号名に対して属性（入力、出力、双方向）を付けることができる。この場合、出力は期待値として処理する。

```
00110100111011110011001001101000
      ↑          ↑       ↑  ↑
    ADDR        DBUS    CNT RST
```

図２　信号名の定義

3．2 テストパタンに対するコードの定義

テストベクトルのフィールドに割り付けるテストパタンにコードを定義し、設計者にとって意味がわかるようにする。例えばＡレジスタのアドレスがＦ３０Ａ$_{(16)}$であるとするとＦ３０Ａ$_{(16)}$に対してＡＲＥＧというコードを付けることになる。テストパタンは４値（０，１，Ｘ，Ｚ）が使用できる。

```
ＡＲＥＧ,   Ｆ３０Ａ,     ０,   １６
ＢＲＥＧ,   Ｆ３ＦＡ,     ０,   １６
ＩＮＩＴ,       ０１,   １６,     ８
ＡＬＬ１,       ＦＦ,   １６,     ８
ＷＴ   ,       ３４,   ２４,     ６
ＲＤ   ,       ３Ａ,   ２４,     ６
ＥＮ   ,         １,   ３０,     １
ＣＬＲ ,         １,   ３１,     １
  ↑          ↑        ↑        ↑
コード   テストパタン   位置     長さ
```

図３　コード定義

テストデータは、このコードを使用して記述する。例えば、『Ａレジスタをクリアした後０１$_{(16)}$（コード：ＩＮＩＴ）を書き込む』といったテストデータを作成したい場合は、図４のように記述する。

```
ＡＲＥＧ, ＣＬＲ
ＡＲＥＧ, ＩＮＩＴ, ＷＴ
ＡＲＥＧ, ＩＮＩＴ, ＷＴ, ＥＮ
ＡＲＥＧ, ＩＮＩＴ, ＷＴ, ＥＮ
```

図４　テストデータソース

TDA: Test data generation system on laptop computer
Kunio Iwawaki, Masayuki Takagi, Shoji Ueda
TOSHIBA corporation

4　マクロを使用したテストデータの作成

機能（論理）シミュレーションを行うためのテストデータは通常似たようなテストベクトルを繰り返すことが多い。そこで、ＴＤＡは、それらの繰り返しパタンをマクロとして定義できるようにした。例えば、図４の例においてレジスタ（ＡＲＥＧ）とセットする値（ＩＮＩＴ）の部分をパラメータ化し、マクロとして定義すると図５のようになる。

```
%DEF  REGWT ($ADR, $DT)
      $ADR, CLR
      $ADR, $DT, WT
      $ADR, $DT, WT, EN
      $ADR, $DT, WT, EN
%ENDDEF
```

図５　マクロ定義

図５のマクロに適当なパラメータを設定しコールすると、展開されて、クリア信号（ＣＬＫ）やライト信号（ＷＴ）、イネーブル信号（ＥＮ）が付加された４ステップのテストベクトルが作成できる。即ち、設計者は書き込むレジスタ（アドレス）とデータだけ入力すれば、指定されたレジスタ（アドレス）にデータを書き込むという一連の作業を行うためのテストベクトルが作成できることになる。従って、レジスタＡＲＥＧ、ＢＲＥＧにそれぞれデータ５５$_{(16)}$、ＡＡ$_{(16)}$を書き込むテストをしたい場合は、図６のように記述するだけで良い。

```
%REGWT (AREG, [55])
%REGWT (BREG, [AA])
              ↑
        即値を意味する
```

図６　マクロコール

さらに、マクロによる一連の作業を定義しやすくするために、プログラミング機能を追加した。レジスタやバスのテストでは、パタンを１ビットずつシフトして入力することが多く、そのためにシフトマクロを用意した。その他、乱数発生用のマクロや展開の流れをコントロールするリピートマクロ、条件分岐マクロ等１０数種のマクロを用意した。

例として、Ａレジスタのビットテストをするために初期パタン＄Ｘを１ビットずつ左にシフトしたデータを順次ＡＲＥＧに書き込むというマクロを図７に示す。

```
%DEF  ATEST ($X)
  %FOR  ?I=0, 7
      AREG, %SHL ($X, ?I)
      AREG, %SHL ($X, ?I), WT
  %ENDFOR              ↑
%ENDDEF            シフト量
```

図７　プログラミング機能を使用したマクロ

5　アセンブル

マクロ展開した後のテストデータソースを、すでに定義してあるコードとテストパタンの対応表に従ってアセンブルする。この部分の動作内容は、通常のアセンブラと同一である。

アセンブルされたテストデータは、タイミングチャートエディタに入力し、波形でビジュアルに確認することができる。また、各機能（論理）シミュレータの入力フォーマットに従ったテストベクトルに変換することもできる。

6　おわりに

マクロ及びコードを使用したテストデータ作成支援システムＴＤＡについて報告した。一度、メモリライト等の一連の決まった動作に対してマクロを定義してしまえば、設計者はマクロのパラメータを変化させてコールするだけで大量のテストデータを作成することができるようになった。これにより、機能（論理）シミュレーションの効率が向上し設計工期の短縮に大きく貢献した。更に、テストデータを容易に作成するだけでなく、テストの内容も分かり易くなり、抜けや重複も減少している。

今後は、入力値だけでなく期待値も簡単に作成できるようにし、機能シミュレーション結果のチェックを自動的化することを考えている。

［参考文献］

１）矢野　他：『ＰＣ－ＦＡＬ　機能シミュレータの移植』
情報処理学会第35回全国大会，5F-9，1987
２）市村　他：『ラップトップＰＣ上の機能シミュレータ：ＰＣ－ＦＡＬ』
情報処理学会設計自動化研究会，40-9，1987
３）木暮　他：『ラップトップＰＣ上の機能図入力システムＦＳＥＴ』
情報処理学会第36回全国大会，1X-1，1988