HDLシミュレータのデバッグ機能を使いこなす

―波形表示，カバレッジ測定，アサーション記述―

伊達啓次郎，富田孝造

デバイスの記事　関連データ

ここではHDLシミュレータや機能検証ツールが備えている波形表示やカバレッジ(網羅率)測定，アサーション記述の機能について解説する．こうしたツールはHDL記述のデバッグに役立つ多くの機能を備えている．ところが，そうした機能の正しい使いかたを理解していなかったり，そうした機能の存在そのものを知らないユーザが少なくないようだ．デバッグの効率を上げるためには，こうしたツールの機能をうまく使いこなす必要がある．（編集部）

最近ではCベース設計などの新たな設計手法が提唱されています．しかし，依然としてHDL(hardware description language)をベースとする方法がもっとも信頼性の高い手法であることはまちがいありません．また，HDLシミュレータはその複雑化する検証環境のベースとなる必須のアイテムであり，LSI設計者にとってもっとも身近なEDAツールの一つです．

HDLによる検証作業には，個々のブロック・レベルの検証，LSI全体の検証，タイミング検証など，さまざまな段階があり，その検証スタイルはユーザによってまちまちです．検証対象が大規模になると，検証フローは大がかりになります．この場合，サード・パーティのデバッグ・ツールを導入することによって検証スタイルが標準化されるケースもありますが，一般にはHDLシミュレータを使って，各人各様のやりかたでデバッグしている場合が多いと思います．すなわち，ひたすら波形とにらめっこしながら，力技でバグを見つけていくというスタイルが中心です．

HDLシミュレータは，デバッグ作業を効率良く進めるための多くの機能をサポートしていますが，この中にはあまりよく知られていないものもあります．本稿では，一般的なHDLシミュレータが備えているデバッグ機能の活用方法について解説していきます[注]．

1 波形を見ながら，効率良く信号をトレース

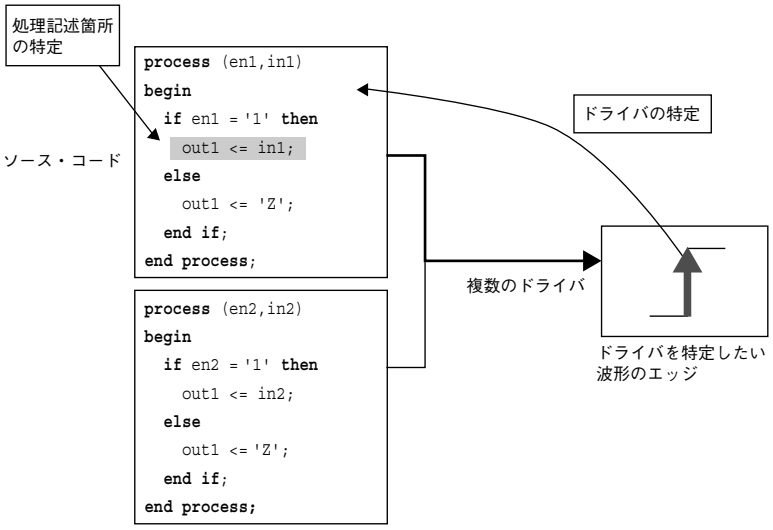
HDLシミュレータによるデバッグと言えば，基本となるのはやはり波形です．HDLシミュレータの波形ビューワは，観測信号のリストと，時間軸に対する論理値の変化を表示します．一般には，この波形ビューワとソース・ファイルを相互に参照しながら，矛盾がないかどうかをチェックしていきます．

波形ビューワを使用したデバッグ作業は，以下のような手順になります．

1) 入力値に対する正しい出力値が正しいタイミングで出力されているかどうかを確認する．
2) 出力波形に問題があれば，出力信号が定義されている階層から，不ぐあいがあると思われるモジュールを特定する．
3) モジュールのソース・コードを表示し，問題の出力信号とその処理内容を確認する．

しかし，実際の作業ではさまざまな理由から，デバッグはすんなりとは進みません．例えば，ファイル内の記述量(行数)が多すぎて，ソース・コード内の問題となる記述箇所が見つからない場合があります．また，必ずしも出力信号が定義されている階層内にバグの原因があるとは限りません．さらに，観測信号数が多すぎて，波形ウィンドウ内に所望の信号をうまく表示できない場合もあります．

注：本記事の内容に関して，米国Aldec社のHDLシミュレータを使用した場合の具体的なデバッグ機能の操作例などは，ソリトンシステムズのホームページ(http://lsi.soliton.co.jp/CQ/)で紹介されている．また，同社のHDLシミュレータ「Active-HDL」と「Riviera」は，本誌付属のDVD-ROMに収録されている．



〔図1〕Show Event Sourcesコマンド

Show Event Sourcesコマンドを実行すると，信号のイベントを発生させているドライバを特定することができる．信号の波形のエッジを選択することで，そのイベントを発生させているソース・コードの行にカーソルが移動する．複数のドライバがある場合にも，選択したイベントを発生した行が表示される．この機能を使うことで，波形ウィンドウで問題のあるイベントを見つけたとき，ソース・コードのどの部分をチェックすればよいのかがわかる．

〔リスト1〕条件分岐によって出力信号に対する代入が分かれている場合

```
show_event : process (A, B, C, D, E, SEL)
begin
  if (SEL = "00") then
    OUT_SIG <= A and B;
  elsif (SEL = "01") then
    OUT_SIG <= B and C;
  elsif (SEL = "10") then
    OUT_SIG <= C and D;
  else
    OUT_SIG <= D and E;
  end if;
end process;
```

このような問題は，シミュレーションとRTL記述の修正を繰り返す過程で頻繁に発生します．そこで，HDLシミュレータが備えているクロス・リファレンス機能やトレース機能を利用してデバッグを進めていくことになります．

●出力波形からソース・コードの対応箇所を参照

波形ビューワ内の特定の信号の波形に問題が発見されたとき，該当する信号からソース・ファイルを参照します．多くのHDLシミュレータは，波形ウィンドウ内の信号からソース・コードの中の信号定義箇所などを参照する機能を備えています（例えば，筆者が利用している米国Aldec社のHDLシミュレータ「Active-HDL」の場合は，「View Declaration」などのコマンドを選択する）．ただし，この操作では出力信号に対応する処理記述へはジャンプできません．

この問題を解決するために，出力のイベントに直接かかわった記述を特定する必要があります．一般に，HDLシミュレータやデバッグ・ツールは，出力波形のエッジ部分からそのイベントを発生させている元の記述箇所を参照する機能を備えています．こうした機能は，「Show Event Sources」，「Cause」，「Cause & Effect」などと呼ばれています．これはシンプルな機能ですが，ソース・コード・デバッグにたいへん効果があります．

Show Event Sources機能を実行すると，指定したノードに対するドライバが特定され，イベントが発生した時間におけるドライバのイベント処理情報をもとに，カーソル時間の出力に関与したソース行が表示されます（**図1**）．例えば**リスト1**のように，ドライバとなるプロセス内において，条件分岐によって出力信号 OUT_SIG に対する代入が分かれている場合でも，イベント発生に関与した代入行を特定できます．また，複数の階層にまたがってマッピングされたネットの最終端のポートに対してShow Event Sources機能を適用し，所望の信号をダイレクトに検索することもできます．

●ドライバ情報を表示するコマンドが用意されている

単一のネットが複数のドライバから別々の値の信号で駆動されると，信号が衝突してそのネットの値が不定（X）になります．しかし，波形ビューワを見ても，結果として出力された（resolveされた）値しかわかりません．それぞれのドライバがどのような値の信号を出力したのか，また，そもそもそのネットに対するドライバがどこにあり，いくつ存在するのかすらわかりません．ドライバを特定できなければ，信号が衝突した原因を特定することもできません．では，特定のネットについて，そのネットに対するドライバ情報を簡単に抽出することはできないのでしょうか？