シミュレーションを始める前に「網羅度」を考えよう
―ソフトウェア工学のテクニックがHDL設計を支援

●ソフトウェア・コンサルタント●
(ニルソフトウェア)
伊藤 昌夫
いとう まさお

ディジタル回路設計にHDLを利用する場合，記述されたデータは一種のプログラムと見なすことができます．したがって，ソフトウェア開発の世界のさまざまなテクニックが有効となります．たとえば，「カバレージ(coverage)」の概念も，そのうちの一つです．

●テストの「網羅度」を測定

カバレージとは，ソフトウェアのテスト(回路設計の「検証」に相当)における「網羅度」を意味します．今，対象となるプログラムのふるまい(behavior)がわかっているとします．書かれたコードのうち，どのくらいがテストの対象になったのかを示すのが，このカバレージです．ただし，網羅度という言葉があいまいであるのと同様，カバレージの意味も，テストの対象となるパス(テスト法)によって異なります．

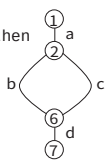
たとえば，カバレージの代表的なものとして，「ブランチ(分岐)・カバレージ」があります．これは，コードのなかの分岐(if文，case文，for文など)に注目したカバレージです．何回かのテストを繰り返した結果，すべての分岐先を1度以上通過したとき，100%のブランチ・カバレージであるといいます．

図1を見てください．左側のVHDLのコードはif文を一つ含んでいます．したがって，分岐が一つ存在し，実行時のパスは二つになります(a→b→dとa→c→d)．片側のパスしかテストされない場合，ブランチ・カバレージは50%になります．

●実運用上の三つの問題

さて，定義は明確になったとしても，実際の適用では，いくつかの問題を考慮する必要があります．

```
1 process begin
2       if RESET='0'or I=15 then
3             I<=0;
4       else
5             I<=1+1;
6       end if;
7 end process;
```



〔図1〕VHDLによるコード断片および制御フロー図

第1の問題は，テスト・データ(テスト・パターン)の選択方法です．コードの制御構造が複雑な場合には，ブランチ・テストにおけるパスの数が膨大になります．したがって，ある特定のパスを通すための入力データ(ないしはその組み合わせ)を見つけることが困難になります．加えて，**図1**の2行目のように，条件式がorやandでつながれている場合，テスト・データを見つけだすことが，さらに困難になります．

第2の問題は，期待値の計算方法です．ソフトウェアの場合，期待値を求めることは非常に困難です．たとえば，収束計算により近似値を求めるような技術計算プログラムにおいて，何が望ましい出力かを知ることは，特別な条件の場合をのぞいて，不可能です(もともと解析解が求められないから計算機を用いているわけですから)．一方，ディジタル回路設計の場合には，入力値が離散的であることなどから，比較的容易に期待値を求めることができます．とはいえ，テスト・パターンやテストベンチの開発に手間がかかることは，ご存じのとおりです．

3番目の問題が，実際のパスの記録です．テスト・データが決まり，期待値が求まったとします．テストを実行した結果，出力値が期待値と等しい場合でも，実行したパスを記録することには意味があります．まず，意図したパスを通っているかを確認するため，次に，使われていないコードを見つけだすためです．もちろん，このためには何らかのテストベッド(テスト用環境)が必要になります．

●ソフトでは，高信頼性の現場で実績

さて，ソフトウェア開発では，このカバレージの概念は，どの程度用いられているのでしょうか．筆者の知る限り，かなりの信頼性を要求される場面以外では，契約上このカバレージを規定していることは少ないようです．ずいぶん前のDOD(米国国防総省)規約では，最低80%という記載があったように記憶しています．しかし，これも軍事関係の特殊なソフトウェアに対する要求ですし，現在のDOD規約にはありません．

この理由として，ソフトウェアの場合，値が必ずしも離散的でないこと，数値処理や記号処理よりも，GUIの部分の処理の比重が大きいことがあげられるかもしれません．筆者はCASEツールを開発していますが，ツールそのもののテストはどうしているのかと聞かれて困ることがあります．HDLシミュレータ・ベンダに，そのツールの信頼性は100%カバレージでテストされているか，聞いてみるとおもしろいかもしれません．

●カバレージの応用範囲は広い

さて，ここまではHDLコードのカバレージに関する話でした．しかし，それ以外のテストにもカバレージの概念は適用できます．たとえば，ゲート・レベルの論理回路図の場合はどうでしょう．いずれもノードとノードの間の関連からなるグラフによって構成されます．これは，**図1**の制御フロー図と同じようなグラフです．したがって，このレベルでのカバレージを考えることができます．LSIの製造テストで問題となるテスト・カバレージ(故障検出率)は，これにあたります．このほか，状態遷移図についても，カバレージの概念を考えることができます．

テストに関してグラフとカバレージの概念は，普遍的に使える道具です．これらの概念は，ソフトウェア開発の世界で用いられ始めてから約30年の歴史があります[1]．それは，テストしやすく，読みやすいコードを作るには，どうしたらよいかを探求する歴史でした．

いきなりシミュレーションを始めるのではなく，まずパスを見つけ出し，カバレージについて考えてみることをお勧めします．きっと，よいコードを書くためのヒントが見つかることでしょう．テストしやすく，十分高いカバレージを得られるコードは，また同時によいコードでもあるのです．

参考文献

1) ボーリス・バイザー，『ソフトウェアテスト技法』，日経BP出版センター，1994(ソフトウェアのテスト技法のほか，論理テストやグラフ行列といった，回路設計者にも有益な説明が多くなされている)．