

7R-2

マイクロプロセッサ“C”言語における
カバレッジ・テスト環境 (BBA)

今野芳弘、岡田慎一

横河・ヒューレット・パッカー(株)

1. はじめに

近年、マイクロプロセッサの応用範囲が広がり、プログラムの欠陥による、損害が多大なものとなる事が少なくなっている。

そこでテストが重要となるが、“C”言語が普及した現在、“C”言語レベルでのテストツールが必要とされている。

テスト手法の一つとしてカバレッジテストが上げられる。カバレッジテストは、C0カバレッジ(命令網羅度)とC1カバレッジ(分岐条件網羅度)に大別出来るが、“C”言語レベルのテストにはC1カバレッジまでのテストが必要と思われる。現在このテストを行うものがBBA(Basis Branch Analyzer)として発売している。本論文では、BBAの機能及び特長について述べる。

2. BBAの概要

BBAの作業手順を図1に示す。

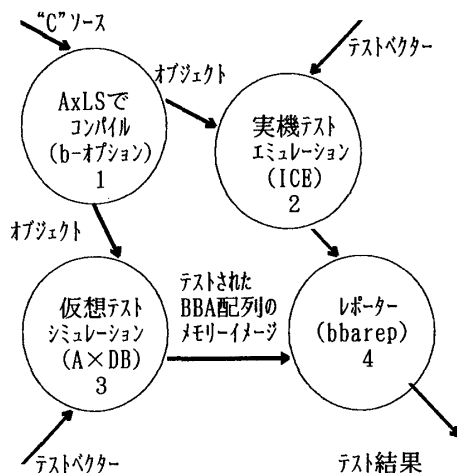


図1 BBAの作業手順

BBAは以下のツールにより構成されている。

- ① A x L S (Advanced cross Language System): A N S I 準拠のマイコン用クロス“C”コンパイラ
- ② H P エミュレータ: I C E
- ③ A x D B (Advanced cross Debug System): マイコン用クロス・シミュレータ。
- ④ B B A レポータ
- ⑤ U N I X コマンド

3. 各ツールの詳細説明

(1) A x L S

A x L S は、-b オプションを付けてコンパイルすると、BBA用プリプロセッサにより、BBA用の配列を発生させる。例を以下に示す。

```

fanc ( )
{ int a, b, c;
  if (a==0)
    b++;
  else
    c++;
}
  
```

以上のソースをBBAプリプロセッサに通すと以下の結果になる。

```

static short _ba_array [ 3 ] ;
fanc ( )
{ int a, b, c;
  _ba_array [ 0 ] =1; if (a==0)
  { _ba_array [ 1 ] =1; b++;}
  else
  { _ba_array [ 2 ] =1; c++;}
}
  
```

The Coverage Test System (BBA) for a microprocessor “C” language.

Yoshihiro Konno , Shinichi Okada

Yokogawa - Hewlett - Packard Corporation

例の様に各分岐命令毎にユニークな `_ba_array [ ]` を持ち、分岐命令が実行されると、配列に1が代入されるような命令が付け加えられる。これらのテスト用配列は、テスト前にはゼロに初期セットされるので、テスト後、この配列の値を見る事で、分岐命令が実行されたか判断できる。

大きなプログラムでは全ての分岐に対して、テスト用の配列を発生させては、テスト時に要するメモリが莫大なものとなる為、テストするファンクションを限定する事が出来る。またオプションにより、`else` 文の無い `if` に `else` を自動追加する事で、`false` の状態があった事も検査する事が可能である。その他多くのオプションを持つ。

(2) HPエミュレータ

インサーキットエミュレータで、実機に接続してテストを行う事が可能である。`A x L S` コンパイルされたオブジェクトは、エミュレータのRAMにロードされる。エミュレータでは基本的にリアルタイムで実行テストを行う事ができる。(厳密に言うとBBAの配列の代入文の実行分だけリアルタイム性が欠ける。)

多種のテストベクターを実機に与えテストが終了した後、エミュレータの**bbaunload** コマンドによりBBAの配列内容をダンプする。(HPのエミュレータはUNIX上で動作しておりダンプファイルは、UNIXのファイルとなる)

(3) A x D B

シミュレータによりテストを行う事も出来る。実行速度はエミュレータと比較して劣るが、オブジェクトをロードするメモリが必要なだけ取れるので大きなプログラムを一度にテストする時に有利である。また、実機の代わりに、EWSのリソース(CRT、キーボード、ディスク)を容易に利用出来るので、実機を使った場合と、同じ様なテスト環境の構築も可能である。

テスト終了後、BBA配列の内容をダンプする。

(4) BBAレポート

テスト後のBBA配列内容のダンプデータから見やすいリポート形式に変換を行うツールである。BBAレポートのファンクション毎のヒット率のレポート例を表1に示す。

<u>_hit_</u>	<u>total</u>	<u>%_function_</u>	<u>file_</u>
8 /	12 (66.67)	keyconvert	convert.c
6 /	9 (66.67)	bitpos	convert.c
0 /	1 (0.00)	error	driver.c
5 /	5 (100.00)	getkey	getkey.c
19 out of 27 retained branches executed (70.37%)			

表1 各ファンクション毎のカバレッジ結果例

また、ヒットしなかったソース及びその前後のソースを表示するレポート形式もあり、次のテスト時に入力すべきテストベクターの参考となる表示も可能である。

Keyconvert	Convert.c
37 → 37	'then' Part of 'if' was never executed
47 → 47	conditional of 'do while' was never TRUE
bitpos	Convert.c
77 → 87	'case' code was never executed

表2 ヒットしなかったカ所のレポート例

3. おわりに

本システムはUNIXのEWS上で動作するものでUNIXとの親和性が良く設計されている。カバレッジテストを行う時は**-b**オプションを付けてコンパイルするだけで複雑な構文解析も必要としない。同EWSには構造化設計ツール(SD)もあり、今後、クロス開発における設計システムとテストシステムの有機的リンクも推進していきたい。