# HDLによる音声処理用DSPの開発(2) – テスト環境 – *

3L－4

中嶋 敏勝　鈴木 俊彦　村山 孝司　星 十郎 †
ヤマハ(株)　電子デバイス事業部 ‡

則安 学　山本 剛士　佐野 直樹　久保 典夫 §
横河電機(株)　EDA開発センター ¶

## 1. はじめに

HDL設計手法を用いて、マイクロプログラム制御方式による音声処理用DSP(YSP)を開発した[1]。本チップの開発に際し、マイクロプログラム制御方式内蔵チップの機能検証、デバッグを効率良く行うことができるテスト環境を汎用シミュレータ上に構築した。本テスト環境の特徴はマイクロプログラムベースのテストプログラムとテスト容易化設計である。本稿ではこれら含めたテスト環境の詳細について述べる。

## 2. テストプログラムの位置付け

まず、テストプログラムがどのようなものであるかを説明する。図1は我々が行っているHDLによるASIC設計フローの一部を示したものである。仕様決定後、チップのRTL(Register Transfer Level)での機能記述を行い、その機能検証のためのテストプログラムをHDLで記述する。そして、これは論理合成後のネットリストに対するゲートレベルシミュレーションにも用いられる。この際、適当なストローブタイミングを与えることで、それ以降のフォールトシミュレーションおよびサインオフシミュレーションのためのテストパターンも同時に生成することもできる。すなわち、テストプログラムは機能検証、タイミング検証、故障検出率確認、サインオフといった行程のすべてに関わって重要な役割を果たすものである。

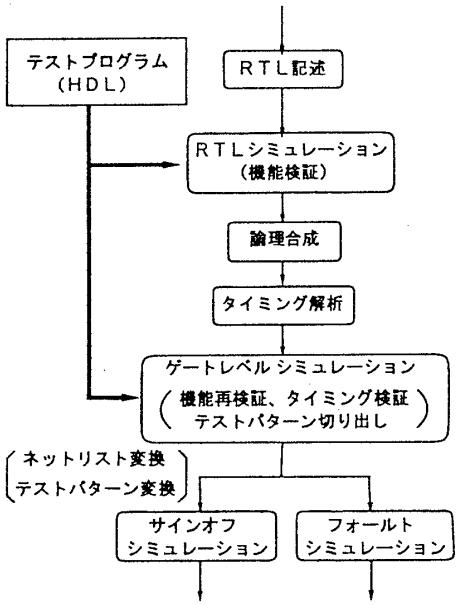


図1: テストプログラムの位置付け

## 3. テスト容易化設計

テスト環境を構築する上で次に重要なのが、テストすべき対象チップのテスト容易化設計である。今回設計した音声処理チップYSPもテストの容易性を考慮している。YSPの概略ブロック図を図2に示す。

実動作時には図の右側にあるマイクロプログラムROMからマイクロコードを読みだして内部制御を行

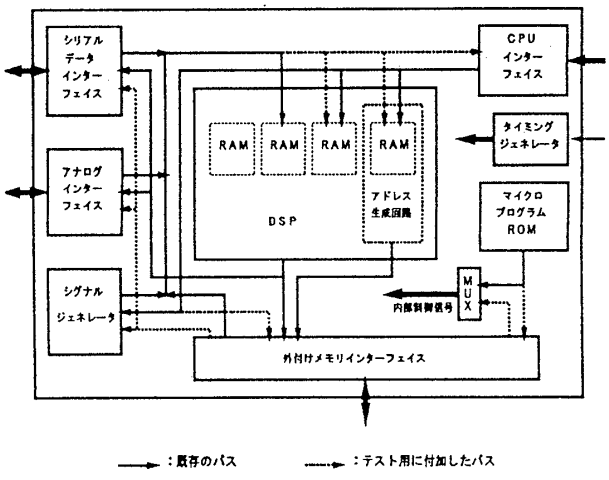


図2: YSPブロック図

*The Development of Sound Processing DSP by using HDL (2) – Test Environment –

†Toshimasa Nakajima, Toshihiko Suzuki, Takashi Murayama, Juro Hoshi

‡Electronic Devices Division, YAMAHA Corporation

§Manabu Noriyasu, Takeshi Yamamoto, Naoki Sano, Norio Kubo

¶EDA Development Center, Yokogawa Electric Corporation

うが、テスト時には、外付けメモリインターフェイスを通して外部から供給されるマイクロコードで制御が行われる。しかもこのコードはテスト専用の制御のために拡張命令を幾つか持っている。また、図中の実線の矢印は実動作時のパスを表し、破線の矢印はテストのために設けたパスを表している。図からも分かるように、テスト時には外部から内部の各々のブロックが可能な限り直接テストできるよう、実動作用のパスに加えてテスト専用のパスを設けている。さらに、図に示しているもの以外にもテスト用の回路を組み込んで、チップ内部の可制御性、可観測性を上げており、これらによって、次節で示すテスト環境の実現を可能にしている。

## 4. テスト環境

図3にテストプログラムとテスト容易化設計をベースにしたテスト用シミュレーション環境の概略を示す。図に示された対象となるチップを含めた全ての環境は、Verilog-HDL によって記述されおり、汎用シミュレータ上で動作するようになっている。

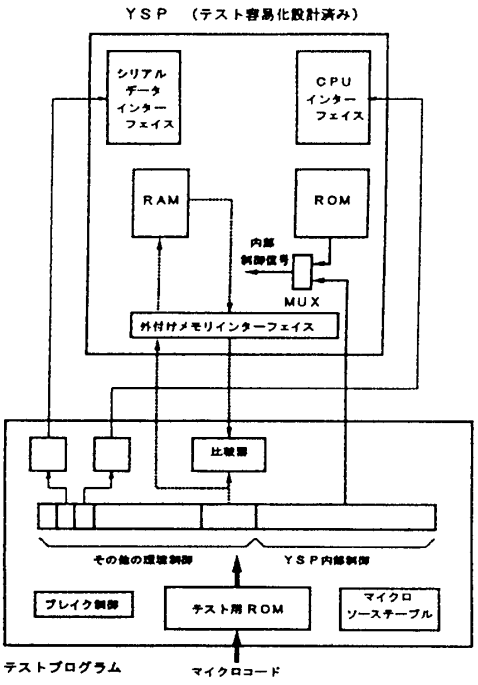


図 3: YSP テスト環境

テストモードが設定されるとチップ外部にあるテスト用 ROM モデルからのマイクロコードが有効になる。このマイクロコードは対象となるチップの内部制御用フィールド以外に、チップの外部インターフェイス駆動制御用フィールド、検証用入力値、期待値を与えるためのフィールド、分岐制御フィールド等のチップ外部あるいは環境自体を制御するための拡張フィールドを持っている[2]。従って本環境を用いれば、マイクロプログラムを記述してアセンブラ (自社開発) でオブジェクトコードにし、テスト用 ROM モデルにダウンロードすることにより、外部からのデータ入力、目的のブロックへの転送あるいはそこでの演算処理、外部への結果の出力、期待値との比較に基づく分岐といった一連の処理 (例：図3の破線で示されたフロー) が自動的に行われる。

さらに、マイクロプログラム自体のデバッグ効率を高めるため、この環境にマイクロコードのアドレスによるブレイク機能、現在実行中のソースコード表示機能等を HDL で実現し、付加した。実際にこれらの機能を使用した例を図4に示す。このような環境を実現することの利点として、プログラムという形でテスト内容が記述されているため、'比較的分かりやすい'、'修正や再利用が容易である'、'系統だったテストを行いやすい'、'テストが自動化しやすい' 等が挙げられる。

```
C2 > disasm;
C3 > bp(2);.                                   ← PC値によるブレイクポイントの設定
1                   cjs(irw);
14  L(irw);         mr(16'h0000); ldct(5);
15  L(11);          rpct(11);
15  L(11);          rpct(11);
15  L(11);          rpct(11);
15  L(11);          rpct(11);
15  L(11);          rpct(11);
15  L(11);          rpct(11);                  ← 実行中のPC値およびマイクロソースコードの表示
16                  ibsel(DR); iw(I0);
17                  nop;
18                  ir(I0); ldy;
                    :
                    :
  mdout = 16'b0000000000000000
1d  L(12);          rpct(12);
1e                  cmpne('h0000); cjp(error);
1f                  crtn;
___ Break at address 'h0002 (BP[1]) ___        ← ブレイク実行
```

図 4: マイクロデバッグ機能の実行例

## 5. 適用結果

本テスト環境を適用することによりテスト全般を約1カ月で行うことができた。また、これに付随して故障検出率：89.5% (テストパターン数：約75K) というフォールトシミュレーション結果も得られている。ここでのパターン数は外部クロック (内部はこの 3 分周) のサイクル数で表現している。

## 6. まとめ

HDL を用いてマイクロコードベースのテスト環境を構築したことで、マイクロコードのテストプログラムを書くだけでチップの機能およびマイクロコード自体の検証、デバッグを効率的かつ容易に行うことができた。このような手法は対象となるチップがマイクロコード制御方式でない場合にも応用は可能であると考えられる。

今後はテスト環境をより現実に近付けて、ボードあるいはシステムレベルでのシミュレーション環境を構築することが考えられる。

## 参考文献

[1] 中島、鈴木、他:"HDL による音声処理用 DSP の開発 (1)"、情報処理学会第 43 回全国大会、3L-04、(1994-9)
[2] 佐野、久保:"ASIC 設計デバッグサポートシステム CEEDS-ASIC"、信学技報 VLD93-29、(1993-7)