C-009

# FPGA を用いたプロセッサ検証システムの設計と実装

## Design and Implementation of a Processor Verification System with FPGA

中谷嵩之 †　山崎勝弘 †　小柳滋 †

Takayuki Nakatani　Katsuhiro Yamazaki　Shigeru Oyanagi

## 1. はじめに

LSIの回路規模の増大に伴い、デジタル回路の設計における検証作業は非常に困難となっている。特に複数のプロセッサを用い、プロセッサと専用ハードウェアが連携するSoCなどでは、構成するモジュールの動作が互いに深く関係し、精度の高いシミュレーションが求められている。最近のハードウェア設計ではVHDLやVerilog-HDL、より高位のハードウェア記述言語を用いて行われており、ソフトウェアによるRTLシミュレータで容易にシミュレーションが可能である。しかし、RTLシミュレータの動作速度は回路規模が大きくなるほど低下し、数十MGate規模の回路シミュレーションでは数Hzから数百Hz程度の検証速度が限界とされている[1][3]。そのため、最近では設計対象と等価な回路をハードウェアエミュレータやFPGA上で実際に動作させ、検証を行う手法が多く用いられている。

ハードウェアエミュレータとしてはCadence Design System社のIncisive Palladium[1]等がある。動作速度は1MHz程度と高速であるが、非常に高価であり、手軽に利用できる環境ではない。それに対しハードウェアの検証にFPGAを用いる場合、比較的安価に利用が可能であり、検証対象のモジュールの動作速度とほぼ同じ速度での検証が可能である。そのためFPGAを利用する様々な検証システムが開発されており、一般的には検証対象のモジュールに検証用の回路を付加して行う。製品としてはXilinx社のChip Scope[2]等が販売されている。

FPGA 上で検証を行う場合、情報の記録方法が検証速度に大きな影響を与える。1 つの信号を記録する場合でも、100MHz で動作する回路の場合 1 秒間の状態を記録するためには約 12 M Byte のメモリが必要になる。大規模な SoC の検証などでは膨大な数の信号を記録する必要があり、FPGA 内のメモリやロジックブロックにそれだけの情報を記録することは不可能である。そのため一旦検証対象のモジュールを停止させ、外部のメモリや PC 等に記録データを転送する必要が生じ、検証速度に大きな影響を与える。

† 立命館大学大学院理工学研究科 ,Graduate School of Science and Engineering, Ritsumeikan University

記録した信号の中で必要とされる情報は、エラー発生箇所や特定のステップなどのごく一部であり、検証用の回路には信号を記録する際に条件を付け、記録データを削減することが可能になっている。しかし、その条件は 2 値の比較や簡単な状態遷移などで、複雑な動作をするハードウェアの検証に適していると言えない。

我々は、検証速度の高速化と検証にかかるコストを減少させることを目的として、ハードウェア検証システムの研究を行っている。本システムでは、信号線の観測を独立したハードウェアで並列に実行することによって動作速度を向上させ、複合的な観測条件を用いることで不必要なデータを削減する。また、信号線の観測条件の記述から検証用の機能を組み込んだシステムを自動的に生成する。

本論文ではFPGA 上に実装したプロセッサ検証システムと FPGA 上で行った実機検証、研究中のハードウェア検証システムの自動生成について述べる。

## 2. プロセッサ検証システム

### 2.1 設計方針

プロセッサ検証システムの主な目的は、(1)ハードウェア検証の容易化及び高速化、(2)様々なプロセッサの動作検証の容易化、である。この検証システムを使用することで、検証時の記録データを削減し、ハードウェアの機能検証をより高速に実行することが可能である。

### 2.2 システムの概要

図 1 にプロセッサ検証システムの構成を示す。システム全体は制御部、検証対象、観測部、通信部で構成されており、単一または複数の FPGA に実装される。

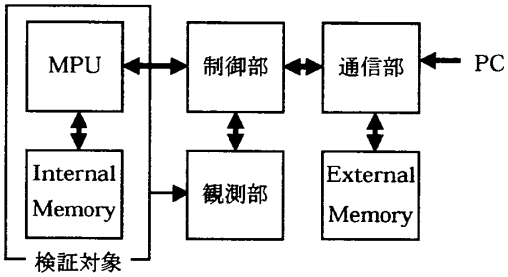


図 1　プロセッサ検証システムの概要

## 2.3 制御部

制御部はSequencerとSystem Status Decoderで構成される。Sequencer はプロセッサ検証システム全体を制御するマルチサイクルプロセッサであり、観測部、通信部の制御と簡単なテストパターンの作成などを行う。Sequencer の命令セットは 8bit/16bit 固定であり、制御信号を操作するために bit 操作命令を持つ。Sequencer は 8 個の 8 bit レジスタを持ち、そのうち 4 個は制御信号の入出力に用いることが可能である。プログラミングには専用のアセンブラを用いる。

System Status Decoder は各部および検証対象への制御信号を出力するモジュールであり、Sequencer によって指定された状態に応じて制御信号を出力する。このモジュールを用いることで、Sequencer は多数の制御信号を 1 クロックで変更が可能であり、制御よるオーバヘッドを減少させることができる。

## 2.4 観測部

本検証システムでは動作中の信号を観測・記録するために Instruction Clock Counter(ICC)モジュールを用いる。ICC は観測する信号線の状態監視、条件比較を行う condition モジュール、観測された状態に従って遷移する state モジュール、信号の状態を記録するための record モジュールという 3 つのサブモジュールを組み合わせて構成されている。

condition モジュールは信号線と状態の比較を行う回路であり、信号の状態が設定条件に一致した場合、match 信号を出力する。match 信号はそのまま record モジュールの動作信号として用いられるほか、state モジュールの状態遷移や観測の終了信号としても用いられる。

state モジュールは有限状態機械であり、観測信号や condition モジュールからの match 信号によって状態を遷移する。記録状態に遷移した場合、record モジュールに match 信号を出力する。このモジュールを用いることで、時間的な条件を用いたエラー検出が可能である。

record モジュールはレジスタやメモリで構成され、条件に従って信号の状態を記録する。状態を記録する場合はそのままメモリへ書き込み、状態に応じた前処理を行う場合はカウンタなどの機能を持つことが可能である。内部記憶領域を使い切ったときは fill 信号を出力し、Sequencer に対して割り込みをかける。

信号が特定の状態であるクロック数をカウントもしくは記録する場合、ICC は図 2 のような構成になる。また、条件と記録する信号を別にしたときの構成は図 3 のような構成となる。

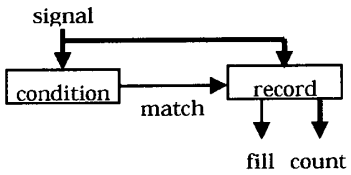


図 2 単純な信号の記録

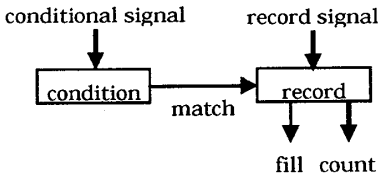


図 3 条件と異なる信号の記録

観測の条件が複数サイクルに及ぶ場合、図 4 のように state モジュールを用いる。state モジュールは内部状態に応じて condition モジュールからの信号を選択する。

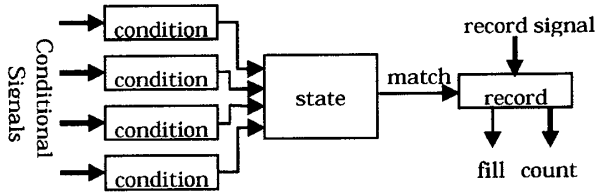


図 4 複数サイクルに渡る条件での観測

また、図 5 のように内部にバッファを設けることで、条件に一致した前後の状態を観測することができる。

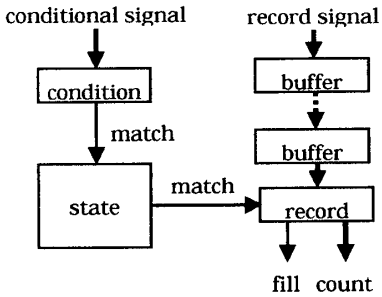


図 5 条件の前後の状態の記録

## 2.5 通信部

ICC 内の記憶領域を使い切った場合や、テストデータを変更するときなど、検証システム外部との通信を行う部位である。要求に従って外部に搭載したメモリ(DRAM, Flash RAM など)への転送を行い、PC と通信する場合は RS232C や PCI バスなどを通じて接続する。通信部の動作は制御部の Sequencer のプログラムによって規定される。

## 3. 検証対象のプロセッサアーキテクチャ

検証対象のMPUとして独自命令形式のプロセッサを設計し、プロセッサの機能及び性能の検証を行った[5][6]。プロセッサは設計が容易でパイプラインに適したRISC

プロセッサを設計した。MPUは同一の命令形式をシングルサイクル方式とパイプライン方式の 2 つの方式で検証、実装した。

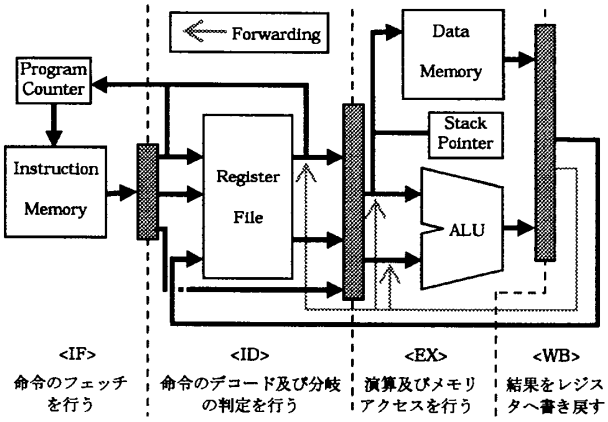


図 6　パイプライン MPU のデータパス

## 4. FPGA 上での動作検証

### 4.1 FPGA への実装

プロセッサ検証システムの実装対象は Celoxica 社製 RC100 ボードである。システム全体を Xilinx 製 SpartanII FPGA の 1 チップ内に実装した。設計においてはメモリのみ IP を使用したが、それ以外のモジュールは Verilog-HDL を用いて設計を行った。

RC100 ボードは通信用の端子を持たず PC との直接のデータ入出力が不可能であったため、ボードに搭載された Intel 製 Strata Flash Memory をデータの入出力に使用した。そのため、通信部は Flash Memory の読み書きを行う Flash RAM Module で構成される。

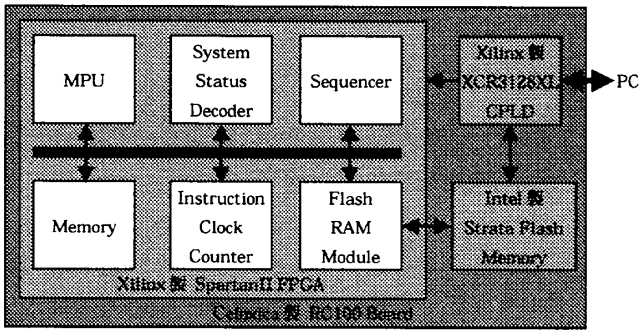


図 7　FPGA 上に構築したプロセッサ検証システム

### 4.2 検証の手順

#### 4.2.1 プログラムとデータのロード

RC100 ボード上での検証では最初に Sequencer が Flash Memory から必要な命令とデータを Memory へ移す。この時、Sequencer のプログラムを変更することで、Memory へ移動させるデータの数や種類を変更できる。

#### 4.2.2 MPU によるプログラムの実行

次に MPU がプログラムを実行する。このとき、Sequencer は MPU からの動作終了信号を待つ。ICC は実行中の MPU の状態を監視し、条件に従って各信号線の状態などを観測する。ICC の record 部が用意された記憶領域を使い切った場合も次の状態へ遷移する。

#### 4.2.3 プログラムとデータの書き戻し

MPU がプログラムを実行後、Sequencer が結果を Flash Memory へ書き戻す。書き戻すデータは Memory の内容と ICC の内容である。そのためにまず Flash Memory の内容を消去する必要がある。Sequencer は Flash RAM module を制御し、書き戻しに必要なメモリの量だけを消去する。次に Memory と ICC から Flash Memory へデータを転送し、PC から Flash Memory 内のデータを読み出すことで結果を確認する。

### 4.3 条件の設定

観測を行ったのは、パイプライン方式での命令実行数、命令の種類毎の頻度、さらに内部配線を用いた観測としてパイプライン方式のみストール回数である。命令実行数は MPU の命令のフェッチを監視し、命令のオペコード毎に回数を測定した。ストールは MPU 内部のストール信号を観測し、MPU 動作中にアクティブになった回数を測定した。

## 5. 実験結果と考察

### 5.1 プログラムの実行結果

FPGA 上でプログラムを実行し、命令の統計結果を表 1 に示す。実行したプログラムは両方式とも要素数 50 の最大値検索・要素数 40 のバブルソート・掛け算・割り算である。

表 1 の数値は MPU の動作時に取得したものである。シングルサイクルとパイプラインの実行した命令に差異はなく、割り算のプログラムのみ、ストールが発生した。

表 1　MPU の命令実行数と頻度

| 命令 | 最大値検索 (要素数 50) | バブルソート (要素数 40) | 掛け算 | 割り算 |
|---|---|---|---|---|
| nop,halt | 2(1%) | 3(0%) | 2(6%) | 2(5%) |
| レジスタ間演算 | 106(26%) | 2882(33%) | 13(38%) | 18(43%) |
| 即値演算 | 102(25%) | 1707(19%) | 6(18%) | 4(10%) |
| 条件分岐 | 101(24%) | 1639(19%) | 7(20%) | 9(22%) |
| メモリアクセス | 51(12%) | 1772(20%) | 3(9%) | 4(10%) |
| 相対アドレスジャンプ | 50(12%) | 819(9%) | 3(9%) | 4(10%) |
| 合計 | 412 | 8822 | 34 | 41 |
| ストール | 0 | 0 | 0 | 1 |

表 2 はプロセッサを単独で動作させた場合と、プロセッサ検証システムに搭載して動作させた場合の動作周波数及び回路規模である。

単独に動作させた場合と比較して、シングルサイクル、パイプライン共に同じ程度動作速度が低下した。

表 2　実装方式による MPU の違い

| | 動作周波数 | 回路規模 |
|---|---|---|
| シングルサイクル | 10.09 MHz | 516 LUT / 171 FF |
| パイプライン | 30.96 MHz | 677 LUT / 266 FF |
| シングルサイクル(検証) | 7.63 MHz | 980 LUT / 342 FF |
| パイプライン(検証) | 23.53 MHz | 1226 LUT / 449 FF |

## 5.2　考察

### 5.2.1　観測条件の自由度と設定の容易性

FPGA というプログラム可能なデバイスを用いることで、任意の信号を観測対象とすることができる。プロセッサのみの検証であれば、PC 上でのエミュレーションでも可能であるが、他のハードウェアと組み合わせたときや、独自の機能を加えた場合などは、シミュレータを変更する必要がある。しかし、このプロセッサ検証システムであれば、FPGA 上でプロセッサにハードウェアを接続し、観測したい信号を指定するだけでよい。

### 5.2.2　動作速度

このプロセッサ検証システムは実際にプロセッサが動作する速度と非常に近い速度で観測が可能であった。また、多くの条件を設定していてもそれぞれの判定を独立したハードウェアで行うことで、条件判定の並列実行が可能である。その結果、非常に高速な検証が可能であり設計における機能検証に必要な時間を削減できた。

しかし、実装時においてプロセッサ単体で動作させた場合と比較して、プロセッサ検証システムでは動作速度が 25%ほど低下した。これは制御を容易にするため ICC 内に観測機能を集中させたことによって、検証対象の MPU から ICC まで信号を伸ばす必要が生じ、クリティカルパスが伸びてしまったと考えられる。ICC を条件ごとに分割し、対象モジュールに内蔵させることで、速度の低下を防ぐことができると考えられる。

### 5.2.3　機能の追加

今回の設計では ICC からの出力信号を Sequencer の制御にのみ用いていたが、検証対象に対してブレーク信号として用いることで、任意のタイミングで検証対象のモジュールを停止させることが可能である。これにより、検証対象の動作中にレジスタやメモリ内容の書き換えや、実行速度を変化させることが可能であり、デバッガとしても使用可能である。

実際に MPU の設計中は、一定期間の内部の制御信号の値を 1 クロックずつメモリに記録し、正常な動作を行っているか比較を行いデバッグの補助とした。現在、図 8 に示すような ICC の条件の設定や検証対象のモジュールに対する観測用信号線の追加などを自動で行うツールを開発中である。このツールを用いることで、バスのプロトコル違反や意図しない制御信号の変化などの観測条件の設定が容易になり、検証に必要なコストを削減することが可能である。

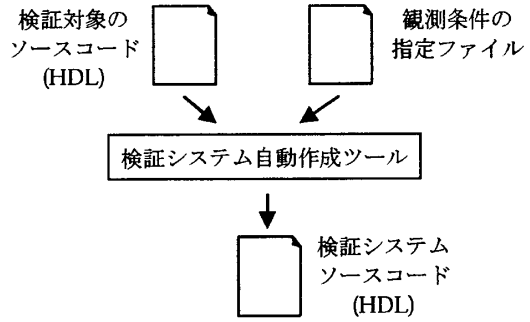


図 8　検証システムの自動作成

## 6.　おわりに

本研究ではプロセッサの機能の検証を容易にするプロセッサ検証システムを作成し、それを FPGA 上で用いて実際にプロセッサの性能を検証し、有用性を確認した。

本システムを用いることで、モジュール内部の信号を様々な条件で観測可能であり、実機上の動作と等しい状態で高速に検証が可能である。

ブレークポイント設定の追加、観測条件の設定作業の自動化など、検証システムとしての機能を充実させることが今後の課題である。

## 参考文献

[1] Cadence Design System 社 Incisive Palladium, http://www.cadence.co.jp/
[2] Xilinx社ChipScope,http://www.xilinx.com/
[3] EDA Oline,http://techon.nikkeibp.co.jp/edaonline/
[4] 中川雄一他, FPGA と PC の連携による System On a Chip の検証手法”, VLSI 設計技術研究会,2003
[5] 池田修久他,“ ハード/ソフト・コラーニングシステムにおける FPGA ボードコンピュータの設計”,情報処理学会第 66 回全国大会論文集 5T-5, 2004.
[6] John L.Hennessy, David A.Patterson,“ コンピュータの構成と設計(上)(下)”,日経 BP 社,1999.
[7] Andrew S. Tanenbaum,“ 構造化コンピュータ構成, ピアソン・エデュケーション”,2000