5J-08 MMU 内蔵 M32R の開発(2)
～機能設計検証～

岩崎保男，村田裕，佐藤雅人，毛利篤史
三菱電機(株)情報技術総合研究所

1. はじめに

M32R ファミリは高性能機器組み込み用に開発された 32 ビット RISC マイクロコンピュータである．RISC アーキテクチャを採用したコンパクトな CPU コアと命令コードを持ちながら，パイプライン処理や積和演算機能の内蔵により，高い性能を発揮する．また，DRAM や SRAM 等のメモリを内蔵しており，外部にメモリチップを配することなく少ないチップ数でシステムを構築でき，一般的な組み込み機器やデジタル情報家電等幅広い分野に適応可能である．

MMU 内蔵 M32R の開発は M32R ファミリの既存する CPU コアを流用し，そこへ MMU(Memory Management Unit)機能を加えることによって短期間での開発を実現している．本稿では，MMU 内蔵 M32R 開発において，既存データの流用を活かした検証方法について述べる．

2. MMU 内蔵 M32R

2.1. 概要

今回開発した MMU 内蔵 M32R は MMU を実装し，仮想記憶管理を実現するもので，これによって様々なオペレーティングシステムに対応することが可能となり，より柔軟にシステムへ組み込むことができるようになる．また，実装する MMU には TLB(Transfer Lookaside Buffer)を内蔵しており，仮想アドレスから物理アドレスへの変換を高速化している．

MMU 内蔵 M32R は既存の M32R ファミリの 1 モデルをベースに MMU 機能を追加しており，短期間での開発及び検証を実現している．

2.2. ブロック構成

MMU 内蔵 M32R のブロック構成を図 1 に示す．MMU 内蔵 M32R は既存 M32R ファミリの CPU コアインタフェース部を MMU ブロックで置き換えたものであり，既存アーキテクチャを継承しつつ，他のブロックについて極力影響を及ぼさないように設計を行なった．

Development of M32R with MMU (2)
–Verification–
Yasuo Iwazaki, Hiroshi Murata, Masato Sato, Atsushi Mohri
Mitsubishi Electric Corporation, Information Technology R&D Center
5-1-1 Ofuna Kamakura Kanagawa,247-8501,Japan

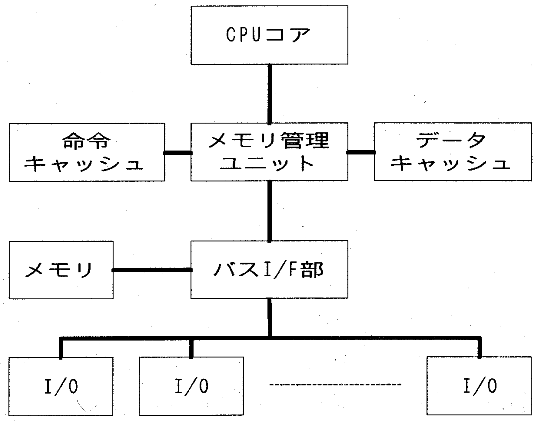

図 1 MMU 内蔵 M32R ブロック構成

3. 設計検証

3.1. 検証方法

近年の LSI の開発においては，回路の大規模化等によって設計検証に多大の時間がかかってしまう．そこで，今回は以下の点を考慮して検証を進めることとした．

(1) 過去のデータを極力再利用する
(2) 設計と緊密なインタフェースをとる
(3) 効率的に全ての検証項目を網羅する

MMU 内蔵 M32R の設計検証は，既存モデルの検証データを極力再利用し，新規に追加した機能部分に対してのみ新しくテストパターンを作成するという方針で行なった．即ち，MMU 機能を追加したことによって仮想記憶管理が行われるが，システムリセット後の設定では仮想記憶モードを無効にするためアドレス変換は行われず，従って既存のテストパターンをそのまま利用できる．そして，仮想記憶管理周りの機能検証については新規にテストパ

ターンを作成して対応する．

また，テストパターンとは独立して，システムのモードを設定する初期化ファイルを用意し，キャッシュの有効/無効を設定したり，各種I/Oコントローラの設定を行なう．システムの動作モードをテストパターンと独立させることで，テストパターンを各種動作モードに対応して変更する必要がなく，システム動作モードを設定する初期化ファイルを複数用意することで，全てのシステム動作モードを網羅することを可能としている．

(2)においては，設計モデルや検証環境の開発途中でのバージョン管理は必須である．これらを管理することによって，設計と検証とのインタフェースを緊密にし，極力杜撰なミスを無くすことで，設計及び検証の効率化を図った．具体的には，設計モデルや検証環境及びテストケース等を全てバージョン管理して検証の対象を明確化することで，不要な混乱を排除した．

(3)については，既存のテストパターンによってモデル全体のどの部分が検証されるかをまず検討し，それを元に今回の機能追加及び変更によって新たにテストパターンを追加もしくは既存テストパターンの変更に関する検討を行なった．例えば，MMU機能追加に伴い，仮想アドレスから実アドレスへの変換に関する機能やTLBについては全くの新規部分であり，新たにテストパターンを追加する必要がある．それに対して，命令セット等既存モデルから全く変更を受けない部分については既存のテストパターンを流用する．また，若干の変更を受ける場合は既存のテストパターンを流用しつつ，変更内容に合わせて検証項目もくしはテストパターンを変更する．このようにして，新たに作成するテストパターンを極力減らし，検証時間の短縮を図った．

3.2. 検証環境

図2にテストパターン作成からシミュレーション実行までの流れを示す．

図において大きく設計モデル，テストベンチ，テストパターンに分けられる．設計モデル及びテストベンチはVerilog-HDLで記述され，それぞれコンパイルされてシミュレータ実行の対象となる．また，テストパターンは初期化プログラムと対象テスト毎のテストケースに分けられ，それぞれアセンブリ言語で記述される．初期化プログラムはシステムに関するモード設定などを行ない，テストケースはテスト対象となる設計モデルの動作確認を行なう．

シミュレーション実行はテストベンチによって制御され，1つのテストパターン毎にテストが正しく終了したか否かの判断がなされログに記録される．

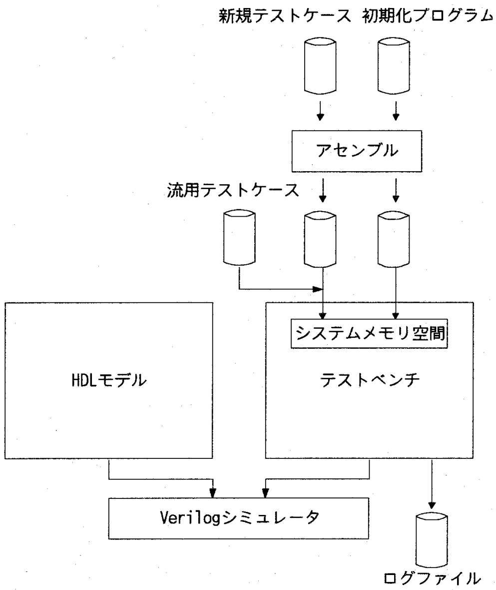

図 2　検証フロー

今回は既存のテストパターンを流用しつつ，新たに220個の検証項目を洗い出した．そして，シミュレーション実行を効率よく行なうために，少ないテストパターンで複数の検証項目を網羅するよう考慮してテストパターンを作成した．

4. おわりに

今回MMU内蔵M32Rの開発における機能設計検証を短期間に行なうために，既存の検証データを活用した効率的な検証方法について述べた．今後，益々LSIの開発期間は短くなり，かつ回路規模は大きくなる．この相反する要求に対応するためには，設計回路のIP化のみならず設計検証に関するデータについてもIP化を考える必要がある．今回は既存機種に対するテストパターンを流用することで検証の効率を上げることができた．また，新規のテストパターンについても将来の再利用を考慮して作成した．そして，さらなる検証の効率化を目指し，今後の製品開発に貢献していく．