ASIP MeisterによるARMプロセッサの設計*

須崎 英之† 塩見 彰睦‡
静岡大学大学院情報学研究科 静岡大学情報学部情報科学科

1 背景・目的

プロセッサのアーキテクチャの複雑化、回路規模の増大に伴い、開発工数の増加及び、良いアーキテクチャの探索が困難になってきた。より良いアーキテクチャの探索には、多くの設計工数が必要であり、アーキテクチャの向上と設計工数の削減の間のトレードオフが難しくなっている。また組込み用途向けプロセッサは、製品としての寿命が短いので、迅速なプロセッサの開発が求められている。そこで我々は、アーキテクチャ設計の効率化、開発工数の削減を目的とした、組込み用途向けプロセッサ設計支援システム ASIP Meister[1] を提案した。

本研究では、ASIP Meister の設計能力の確認のためにアーキテクチャや命令セットに特徴のある ARM プロセッサ [2][3] を設計する。

2 ASIP Meister

本研究では、ASIP Meister を使用してプロセッサの設計を行う。ASIP Meister はプロセッサ設計支援システムであり、設計記述レベルの高位化・設計資産の再利用・精度の良い設計品質の見積りという点から設計品質の向上と設計工数の削減を図っている。ASIP Meister は GUI を利用しており、ユーザは GUI を使って、設計パラメタとマイクロ動作記述を入力する。

ASIP Meister では、プロセッサを設計するのにプロセッサアーキテクチャの設定、プロセッサに使用するリソースの宣言、プロセッサのインタフェースの宣言、命令タイプと命令コードの定義、プロセッサ性能の見積り、命令のマイクロ動作の記述、HDL 記述の生成という 7 つの項目に分割して設計する。ASIP Meister は、入力されたデータから設計品質見積りと RTL(Register Transfer Level) 設計記述ファイルを出力する。RTL 設計記述は VHDL で記述され、論理合成モデルとシミュレーションモデルが自動生成される。

3 ARM プロセッサ

ARM プロセッサは、今日、世界中の多くの半導体メーカーにライセンスされており、現在入手可能な ARM プロセッサは ARM7、ARM9、ARM9E、ARM10E、ARM11、SecurCore の 6 種類がある [3]。ARM アーキテクチャは、バークレー RISC の設計からロード/ストア・アーキテクチャ、固定長 32 ビット命令、3 アドレス命令形式等の特徴を採用している。

ARM プロセッサはこれらの RISC としての特徴を持っている一方で、命令セットの構造に他の RISC プロセッサには見られないユニークな特徴がある。以下に ARM プロセッサの特徴を挙げる。

- 32 ビット長の命令セットである ARM 命令は、条件実行を行っており、命令コードの上位 4 ビットにあるコンディションフィールドの値と CPSR(カレント・プログラム・ステータス・レジスタ) の条件コードフラグの値で命令の実行/非実行を決定する。
- 複数のレジスタに対して任意 (またはすべての) サブセットを一括でメモリからのロードまたは、メモリへストアすることができる。
- ARM 命令セットのサブセットの 16 ビット圧縮形式と考えることができる Thumb 命令セットがある。

以上の特徴は全ての ARM プロセッサで共通の特徴である。

本研究では、図 1 に示すパイプライン構造をもつ ARM7 プロセッサ [4] をターゲットとする。ARM7 プロセッサは、現在入手可能な ARM プロセッサの中で最も基本的なものである。今後、設計した ARM7 プロセッサを基にして、設計記述を変更すれば派生品の設計が行えるため、ARM7 プロセッサをターゲットとした。

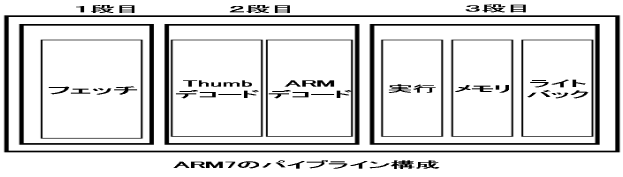

図 1: パイプライン構成図

ARM7 プロセッサでは、1 段目でフェッチ、2 段目でデコード、3 段目で実行とメモリアクセスとライトバックを行うようになっている。図 1 でわかるように、デコード部には Thumb デコードと ARM デコードがある。ARM7 プロセッサは、16 ビット表現の Thumb 命令セットを処理するときにデコード部で伸張し 32 ビット表現の ARM 命令セットへと変換して、処理するためである。

4 設計方針

ASIP Meister におけるプロセッサの設計では、設計目標、パイプラインの概要、命令及びデータのビット幅等のプロセッサの概要の決定と ASIP Meister に登録されているリソースから必要

*The design of ARM processor by ASIP Meister
†Hideyuki Suzaki : Graduate School of Information, Shizuoka University
‡Akichika Shiomi : Faculty of Information, Shizuoka University

とするリソースの宣言、入出力ポートの宣言、命令・命令タイプの定義、マイクロ動作の記述がある。

条件実行については、3 章で述べている。実行条件が多いため、条件実行をマイクロ動作で記述するのは困難である。そのため、条件実行のためのリソースを設計した。設計したリソースは、命令コードの上位 4 ビットと CPSR の条件コードフラグの関係から、実行/非実行を決定する条件判定回路である。

ARM プロセッサは、3 章で述べた Thumb 命令セットを持っている。16 ビット表現の Thumb 命令は、Thumb デコード部で伸張した後、ARM 命令デコード部で処理される。しかし、ASIP Meister ではデコード部が自動生成されるため、デコード部で Thumb 命令を伸張することができない。また、ASIP Meister は異なる命令ビット幅の命令を定義できないため、Thumb 命令セットに 16 ビットのオペランドを追加し、32 ビット表現とした。

さらに、ASIP Meister では、ARM 命令セットと Thumb 命令セットの命令モードを考慮せずに設計すると命令の競合が起こる。そこで、命令コードは命令モード用フラグを最上位ビットとして追加した 33 ビット表現として命令を定義することとした。

次に、命令タイプごとの設計方針を述べる。() の数字は設計した命令数である。

- データ処理命令 (48) /ワードおよび符号無しバイト転送命令 (4) / ハーフワードおよび符号付きバイト転送命令 (6)
  これらの命令は、ソース・オペランドの種類が複数ある。これらを 1 つの命令タイプとして扱うと、ASIP Meister では命令の競合が起こるため、ソース・オペランドの種類で命令のタイプを分けた。そのため、本来の ARM プロセッサの命令数より増加した。
- SWI 命令 (1) /ステータスレジスタ転送命令 (3) /分岐命令 (3) /乗算を含む命令 (6)
  これらの命令は、ASIP Meister に登録されているリソースを使用しマイクロ動作を記述した。
- 複数レジスタ転送命令 (2) /スワップ命令 (1)
  これらの命令は、 1 命令で複数回のメモリアクセスが必要であり、ASIP Meister のマイクロ動作記述では対応できない。また、 3 章で述べた複数レジスタ転送命令はメモリアクセス数が固定ではないので、可変メモリアクセスに対応できるリソースを設計した。
- コプロセッサ命令 (5)
  本来の ARM プロセッサでは、コプロセッサとのデータの受け渡しを行っているので、コプロセッサへデータを受け渡すためのマイクロ動作を記述した。
- Thumb 命令 (53)
  Thumb 命令は ARM 命令と同等の動作を行うものなので、ARM 命令の入力データを参照して設計した。

以上の方針で、ASIP Meister へ設計データを入力した。

5 結果と考察

ASIP Meister で自動生成される VHDL で記述されたシミュレーションモデルを用いてシミュレーションを行い、動作の確認を行った。設計した命令数は ARM7 プロセッサの全 132 命令である。今回、コプロセッサの設計は行っていないため、コプロセッサ命令を除く 127 命令については正しく動作することが確認できた。切替命令 (BX) 命令による ARM 命令/Thumb 命令の命令セットの切替、条件実行についても正しく動作することを確認した。

さらに、アルテラ社の PLD 開発ツール Quartus II Version 3.0 Service Pack 2 を用いて、ASIP Meister で生成された論理合成モデルを論理合成・配置配線した。デバイスは、APEX20K,EP20K200EFC 484-2X (平均 20 万ゲート相当) とした。その結果、デバイスのロジックエレメント 8320 個のうち 4430 個を使用し、その使用率は 53%であった。タイミング解析の結果、最大で 14.28MHz で動作することがわかった。

また、設計工数は、リソース設計に約 30 時間、プロセッサ設計は約 11 時間で、プロセッサアーキテクチャの設定、プロセッサに使用するリソースの宣言、プロセッサのインタフェースの宣言に 1 時間、命令タイプと命令コードの定義に 2 時間、命令のマイクロ動作の記述に 8 時間を要した。

設計工数から、ASIP Meister を使用したプロセッサ設計の容易性が確認できる。今回設計したほとんどの命令の動作がマイクロ動作の記述のみで設計することが可能であった。本研究では、ASIP Meister の多様な命令動作への適応可能性を確認することができた。

6 まとめ

本研究では、ASIP Meister を使用したプロセッサの設計事例として ARM プロセッサの設計を行った。条件実行、複数レジスタ転送命令、スワップ命令はマイクロ動作記述だけでは設計できなかったが、リソースを追加することで設計できることが確認できた。

今回の設計は ASIP Meister に不足しているリソースを追加して、ARM プロセッサの設計を行った。しかし、現在リリースされている ASIP Meister はユーザによるリソースの追加を支援する環境が整備されていない。ASIP Meister の課題として、ユーザが ASIP Meister に不足しているリソースを自由に追加できるようにすることがある。

本研究で設計した ARM プロセッサの課題としては、FPGA による動作確認が挙げられる。

参考文献

[1] 今井正治, 武内良典, 塩見彰睦, 佐藤淳, 北嶋暁, ”特定用途向きプロセッサ開発システム ASIP Meister”, 電子情報通信学会 信学技報, ICD2002-113, vol.102, No.401, pp.39-44, 2002 年 10 月.

[2] Steve Furber 著/アーム (株) 監訳 : 改訂 ARM プロセッサ : CQ 出版社 : 2001

[3] ARM プロセッサ入門 : CQ 出版社 : 2003

[4] ”ARM7TDMI Data Sheet” : ARM 社 : 1995