6L−8 汎用マイクロプログラミングツールによる 1チップCPUプロセッサのマイクロプログラム開発

岩田俊一　清水徹　平岡精一　西川浩司　堀本正文
三菱電機（株）

1．はじめに

現在、マイクロプロセッサの多くでマイクロプログラム制御方式が採用されている。マイクロプロセッサの高機能化とともにマイクロプログラムの規模や複雑度が増えた結果、マイクロプログラム開発にはツールが不可欠となっている。しかし、マイクロプログラムがプロセッサのハードウエアに大きく依存する一方、設計者が極めて限定されていることから、簡易な専用ツールか、汎用的なツールが使用される場合が多い。

今回、我々はビジネスコンピュータ用の1チップCPUプロセッサ（170万トランジスタ）のマイクロプログラムを開発するにあたって、まず汎用マイクロアセンブラを開発し、これをアーキテクチャに合わせてチューニングすることにより、専用のマイクロアセンブラを作成した。また、ハードウエア記述言語を用いてマイクロシミュレータを作成した。さらに、同じ手法で他の2つのプロセッサも開発した。

本稿では、今回開発した汎用マイクロアセンブラの概要を説明し、合わせてこれらのツールを3つのプロセッサ開発に適用した結果について比較検討する。

2．マイクロプログラム開発と汎用ツール

マイクロプログラムは、一般に図1に示すようなフローで開発が行われる。

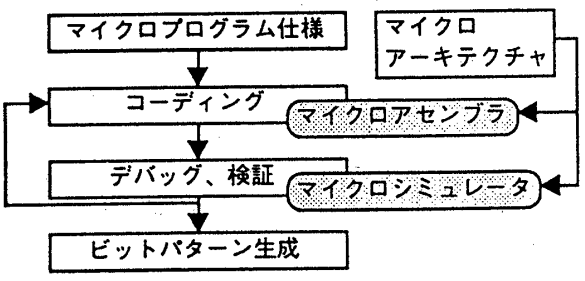

図1　マイクロプログラム開発フロー

Microprogram development methodology of one-chip CPU processor
Shunichi IWATA, Toru SHIMIZU, Seiichi HIRAOKA, Koji NISHIKAWA, Masafumi HORIMOTO
MITSUBISHI ELECTRIC CORPORATION

マイクロプログラムの規模が大きくなると、コーディングやデバッグ、検証の占める割合が多くなる。そこで有効なツールが、マイクロアセンブラとマイクロシミュレータである。

今回、マイクロアセンブラについては、まず汎用のマイクロアセンブラ「uas」を開発した。これにマイクロアーキテクチャに関する情報を記述したテーブルを付加することによって、ターゲットプロセッサ用の専用マイクロアセンブラとなるようにした。このように、汎用アセンブラを最初に開発したのは以下の理由からである。

(1) 複数のプロセッサ開発への適用が予想されていたので、汎用化してツールを共通化することにより、開発・メンテナンスコストの削減が期待できた。
(2) マイクロプログラムの文法やマイクロアーキテクチャに依存する部分を分離することにより、マイクロプログラム設計者が直接開発・メンテナンスすることが可能となる。
(3) ユーザーが限定されているので、自作の簡易ツールでもかなり実用に耐えるものができると予想された。

今回、uasを以下の3つのプロセッサ開発に適用した。

(A) ビジネスコンピュータ用1チップCPU
(B) 32ビット汎用マイクロプロセッサ
(C) 16ビットマイクロコントローラ

マイクロシミュレータについては、既存のハードウエア記述言語を用いて開発した。(A)(B)のプロセッサは社内で開発されたAlhard[1]を使用し、(C)のプロセッサは、Verilog-HDL を使用した。Alhardはレジスタトランスファレベルの言語であり、Verilog-HDL はミックスレベルの言語である。(C)のプロセッサでは、ハードウエアの機能記述と兼用したためこのような選択となった。

3．汎用マイクロアセンブラ「uas」

今回作成した汎用マイクロアセンブラ「uas」は、マイクロアーキテクチャに依存しない共通の処理を行う。マイクロアーキテクチャに依存する情報はテーブルに記述し、uasはそのテーブル

を読み込み、そこに記述されている指示にしたがってマイクロプログラムのソースコードをアセンブルする。テーブルには、①マイクロプログラムの言語仕様、②マイクロ命令フォーマット、③各マイクロオペレーションのビットパターンが記述されている。図2にｕａｓの処理フローを示す。

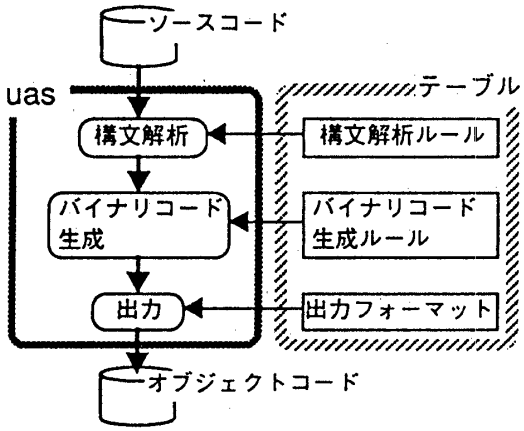

図2　uasの処理フロー

(1) 構文解析

構文解析ルールにしたがってソースコードを読んでいき、各フィールド毎にソースコードに記述されたマイクロオペレーションを設定していく。その他、主な機能を以下に示す。

・マイクロアドレスとしてのラベルの使用
・式の計算機能
・Ｃ言語の#defineと同様のマクロ機能

(2) バイナリコード生成

バイナリコード生成ルールにしたがってフィールド毎に設定されたマイクロオペレーションをバイナリコードに変換する。主な機能を以下に示す。

・単一フィールドだけでなく、複数フィールドの組み合わせを見てバイナリコードを生成する機能
・単一／複数のフィールドの値に基づく条件付き変換機能

(3) 出力

出力フォーマットにしたがってバイナリコードを出力する。出力されたコードがオブジェクトコードとなる。

なお、ｕａｓはＣ言語で1500行のプログラムとなった。開発には１人で約１カ月間を要した。

4．３つのプロセッサ開発への適用結果とその比較検討

表２に３つプロセッサにおけるマイクロアセンブラとマイクロシミュレータのデータを示す。

表1　３つのプロセッサにおけるマイクロアセンブラとマイクロシミュレータ

	(A)	(B)	(C)
μ命令数	2700 ﾜｰﾄﾞ	2700 ﾜｰﾄﾞ	1000 ﾜｰﾄﾞ
μ命令長	171 ﾋﾞｯﾄ	190 ﾋﾞｯﾄ	75 ﾋﾞｯﾄ
μﾌｨｰﾙﾄﾞ数	78	95	43
μｿｰｽｺｰﾄﾞ	34000 行	28000 行	10000 行
μアセンブラ			
汎用部(uas)	1500 行	←	←
テーブル	1433 行	1758 行	593 行
構文解析	130 行	145 行	80 行
ｺｰﾄﾞ生成	1300 行	1610 行	510 行
出力	3 行	3 行	3 行
実行時間*1*2	170 秒	150 秒	25 秒
μシミュレータ			
記述言語	Alhard	Alhard	Verilog-HDL
ステップ数	13500 行	10500 行	5200 行
開発人工	2 人月	1.5 人月	3 人月
用途	μｼﾐｭﾚｰﾀ	μｼﾐｭﾚｰﾀ	機能ｼﾐｭﾚｰﾀ
ｼﾐｭﾚｰｼｮﾝ速度*1	15 ｸﾛｯｸ/秒	30 ｸﾛｯｸ/秒	20 ｸﾛｯｸ/秒

*1 SPARCstation1での測定値　*2 全ｿｰｽｺｰﾄﾞのｱｾﾝﾌﾞﾙ時間

この結果をまとめると、次のとおりになる。

(1) 異なるアーキテクチャに対応できたことにより、ツールの汎用性の高さが実証された。

(2) マイクロアセンブラ全体に対するテーブルの行数の割合が30～50％と決して小さい値ではないが、マイクロアーキテクチャに依存する部分をテーブルとして切り分けることにより、プログラム経験のない設計者が直接作成したり改訂できるメリットは大きかった。

(3) 汎用マイクロアセンブラの場合、テーブルの読み込み等のオーバーヘッドが考えられるが、実行時間を見る限り、実用上問題なかった。

(4) マイクロシミュレータの行数は、マイクロアセンブラのテーブルの行数の10倍近くになっている。マイクロアセンブラに比べてさらに開発効率の向上が望まれる。

5．結　論

異なるアーキテクチャを持つ３つのプロセッサにおいて、マイクロアセンブラとマイクロシミュレータを作成し、これを用いてマイクロプログラムを開発した。その結果を比較検討した結果、マイクロアセンブラでは汎用マイクロアセンブラが有効であること、それに比べてマイクロシミュレータの開発コスト低減が一層望まれていることがわかった。

参　考　文　献

1) 小島ほか：「ﾊｰﾄﾞｳｪｱ動作の対話型ﾋﾞｼﾞｭｱﾙｼﾐｭﾚｰﾀ構築ｼｽﾃﾑ」，信学論，J75-D-II，2