オンチップ・メモリ搭載型PLD/ASICの有効活用方法

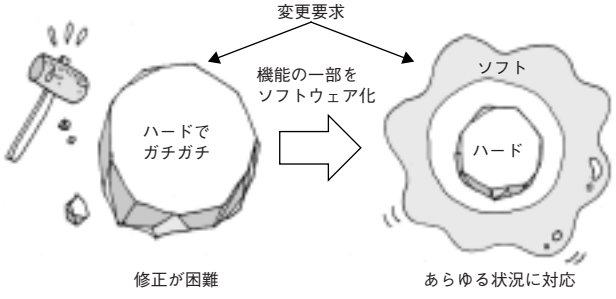
ポータブル・プロセッサによるロジック設計の新しい方式

田中良平/茂木建二

1．オンチップ・メモリは使いよう

最新の大規模PLDやASICはオンチップ・メモリを内蔵して，大規模な論理回路を構築できます．論理設計者なら，同一チップ内にメモリがあることの利便性は容易に想像がつくことと思います．しかし，具体的に内蔵メモリをどのように使用しているのでしょうか？ おそらく，一時データ格納用に使用することが多いのではないでしょうか．一般に，外付けメモリと比較してオンチップ・メモリは容量が小さいので，このような用途になるのは仕方がないのかもしれません．そこで筆者らは，オンチップ・メモリを別の目的に使用する新しい設計方式を提案します．

たとえばプログラマブル・コントローラをPLD/ASICに実装して使用する場合，オンチップ・メモリは，そのソフトウェアを格納するためのエリアとして使用できます．大きなプログラムは組めなくても，小規模なドライバ程度は作成できます．すると，いままでハードウェアで構成していた複雑なコントローラを簡単に作ることができます．これを複数のIP（intellectual property）コアを接続する場合に応用すると，各IPコア間のインターフェースを調整するコントローラが，PLD/ASIC上に有効に実装できる，と考えることができます．IPコアそのものをカスタマイズするのではなく，コントローラ側で各IP間の仕様の相違を吸収するわけです．また，コントローラを専用ハードウェアで構成するよりも，ソフトウェア・ベースのプログラマブル・コントローラを構成したほうが，ソフトウェアで体系的にシステムを管理でき，全体の設計スピードを向上できるものと考えました．オンチップ・メモリにソフトウェアを上手に埋め込んで，ソフトウェア処理が得意な範囲とハードウェア処理が得意な範囲のすみ分けを明確にすることで効率のよい設計が可能となるのです．



〔図1〕ハードウェアの一部をソフトウェア化する
オール・ハードではなく，ソフトを取り入れて柔軟性をもたせる．ソフトでハードをつつむ．

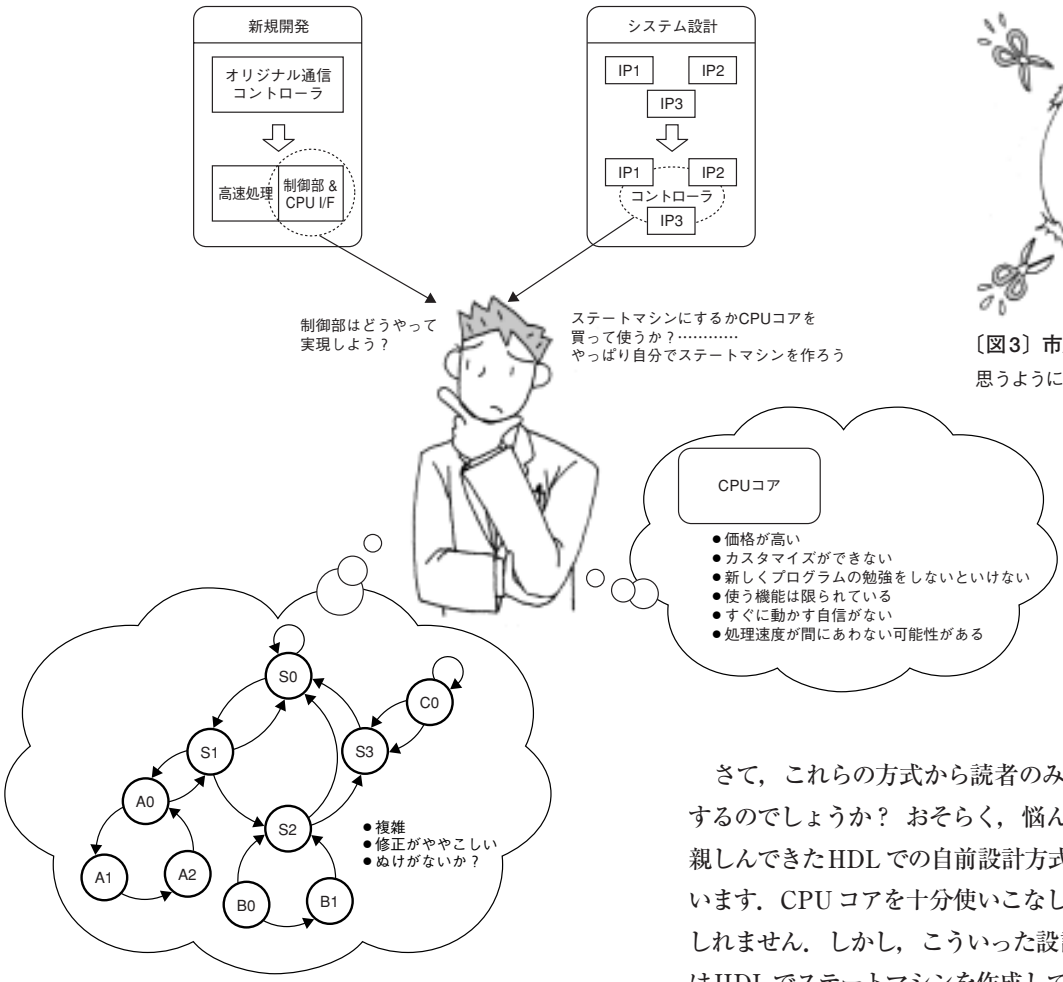
2．ソフトウェア化のメリット

コントローラのソフトウェア化により，仕様変更や不具合の修正に柔軟に対応できます．論理設計において仕様変更や不具合修正との付き合いは腐れ縁のようなもので，つねに設計者につきまとう話です．少しでも，仕様変更/不具合修正に対する設計負荷を軽減することが，論理設計者の永遠テーマの一つではないでしょうか．せっかく苦労して作成したコントローラが次の日に仕様変更されると，ハードウェアによるコントローラでは，回路の変更に伴って，配線遅延が変化します．そのため，タイミング検証からやり直さなければならなくなるので，うんざりすることでしょう．もし，このコントローラの動作をソフトウェアで記述していれば，処理内容を記録しているメモリの中身を書き換えるだけで対応できます．配置配線に変更がないため，検証に要する時間を短縮できます．

3．どうしてソフトウェア化できないのか？

ソフトウェアを組み込めばよいことはわかったとして，なお論理設計者はハードウェアのみでシステムを構築しようとするのでしょうか？簡単な例で，その理由を述べてみます．

通信コントローラのような高機能回路は，通常ハードウェアで実装して高速性をもたせる部分と，ホスト・インターフ



〔図2〕コントローラを実現するには…

複雑なコントローラを実現するのにステートマシンを設計するのも，CPU コアを購入して使用するのも，どちらも気が重い….

〔図3〕市販のIP コアの現状

思うようにカスタマイズできない．

ェースなど，データ・フローのコントロールを行う部分とで構成されます．一方，複数のIP コアを組み合わせてIP コア間のデータ・ストリームをコントロールするには，専用のコントローラが必要になります．読者のみなさんは，これらのコントロール機能を実現するテーマを受けたとき，どのような設計手法をとっているのでしょうか？（**図2**）

いろいろ検討はするものの，結局，ステートマシンを組み合わせてシーケンサを構築することを考えてしまうのではないでしょうか．もちろん状態数が少なければ問題はありません．しかし，複雑な制御を行う場合や，メモリ・コントロールも含めたコントローラを設計する場合，あらゆる状態を想定する必要があるので，制御の複雑さにうんざりする設計者が多いことと思います．コントローラが高機能化するほど，ハードウェア設計者にかかる負荷とゲート規模が増大します．PLD/ASIC では，複雑なコントロール関係は市販のCPU コアを使用して，ソフトウェアで対応するといった選択肢もあり得ます．

さて，これらの方式から読者のみなさんはどの方式を採用するのでしょうか？ おそらく，悩んだあげくにいままで慣れ親しんできたHDL での自前設計方式を採用していることと思います．CPU コアを十分使いこなしている設計者もいるかもしれません．しかし，こういった設計者はごく少数で，大半はHDL でステートマシンを作成していることと思います．

4．市販CPU コアでは過大？

コントローラに，いろいろ市販されている組み込み用CPU コアを使用せず，ステートマシンによるハードウェアを採用している設計者がいまでも多いのはなぜでしょうか？ 筆者らは設計期間が重要な要因であると考えています．設計者は要求仕様を短納期で実現するという課題をつねにもらいます．運よくいままで使い慣れているCPU が市販されていれば，比較的楽に使用できます．しかし，新規にCPU を導入する場合は，ソフトの組み方やCPU の初期化設定など新たに学習することが多く，かなりの抵抗があります．また，多くの市販のIP コアはユーザによるカスタマイズが認められておらず，不要な回路をいっしょに実装しなければなりません（**図3**）．CPU コアはシステムのメイン・コントローラとして使用するには有効ですが，単純なコントローラや，汎用ロジックを実現するためだけに使用するには十分すぎるのです．また，こういったIP コアは高価なものが多く，導入するにはリスクが高すぎると感じている設計者も多いことでしょう．

5．お手軽プロセッサを作りました！

以上のコントローラを作成することに対するジレンマを解