第1章

FPGAでソフト・マクロのCPUを使う理由

“FPGAマイコン”の特徴と将来への期待

浅井 剛

2000年にFPGAベンダが自社製品向けにCPUコアをリリースしてから7年が経過した．ここでは，FPGA向けに提供されているソフト・マクロのCPUを活用する理由について解説する．特に，一つのLSIに複数のCPUコアを搭載するマルチコアに注目する．少量多品種の開発でCPU処理を行いたい場合，32ビットCPUに周辺回路をあらかじめ搭載した汎用マイコンを使うことが多い．しかし汎用マイコンとリアルタイムOSの組み合わせでは，高まる要求仕様を満たせない場合がある．高い性能を得るために動作周波数を上げると，消費電力が増えてしまうといった問題がある．このような状況を解決してくれるのがFPGAでCPUコアを活用する“FPGAマイコン”である．FPGAの大規模化に伴い，一つのFPGAで複数のCPUコアを活用することも現実的になっている．（編集部）

汎用マイコンなどでは実現困難な機能を達成するために，カスタムLSIは不可欠です．しかし少量多品種の開発では，ASIC（Application Specific Integrated Circuit）の開発を断念せざるを得ません．FPGA（Field Programmable Gate Array）とソフト・マクロのCPUの組み合わせは，独自機能のSOC（System on a Chip）を得るための，唯一の選択肢です．

1．汎用マイコン＋リアルタイムOSの限界とマルチコア

組み込みシステムといえば，16ビットか32ビットの汎用マイコンが機能の中心になるのが一般的です．最近では，システムLSI（SOC）として1チップ化することも可能になっています．ソフトウェアの開発効率を上げるために，リアルタイムOSも広く利用されています．この構成はコスト・パフォーマンスにも優れています．

● 一つのCPUですべて処理しようとするから困難になる

汎用マイコンを用いた組み込みシステムの開発過程では，「タスクが多くなりすぎてシステム全体のデバッグが難しい」とか，「割り込み要因が多くなりワーストケースでアプリケーションの処理時間を満たせない」という課題にぶつかることが珍しくありません．前者についてはタスク分割の見直しや優先順位を再整理することで，後者については割り込みハンドラのチューニングなどで対応することになります．

しかし，どうしてもソフトウェアの組み替えだけでは対処しきれないこともあります．CPUの動作周波数を上げられればよいのですが，できない場合は製品としての機能・性能を下げるか，開発初期段階まで戻るかの決断を迫られます．

そのような経験を何度となくしていると，ある疑問が湧いてきます．

「なぜ一つのCPUですべてを処理しなければならないのだろうか？」

具体的な事例を以下に示します．

● 事例1：周辺機能側でデータ前処理を行ってしまう

32ビットCPUにリアルタイムOSを搭載したシステムの

KeyWord FPGA，ソフト・マクロ，CPUコア，マルチコア，汎用マイコン，SOC，リアルタイムOS，マルチプロセッサ，負荷分散型，機能分散型

例を図1に示します．

このシステムでは，アプリケーション・タスクで32ビットのデータ処理性能が求められているとします．割り込みハンドラでは，リアルタイムOSの周期タイマや外部とのシリアル通信（SCI：Serial Communication Interface），スイッチ入力などを処理します．

ここでシリアル通信による入力を考えてみましょう．SCI割り込みハンドラとして必要な機能は，

①SCI受信バッファから受信文字の読み出し
②受信可能文字かの判定
③1文字のエコーバック(受信不可能文字の場合はエラー・コード)出力
④受信バッファへの格納
⑤通信終了デリミタ検出
⑥受信完了のイベント・フラグを立てる

とします．

本システムにおいて，8ビット（1文字）単位で行われるこれらの処理を32ビットCPUで実行する必要があるのでしょうか．必要性だけを考えれば誰が見ても答えは“NO”です．しかし，システムのCPUが32ビットで，かつ1個しかないため，やむなく処理させているだけです．処理性能に余裕があり，その余力の範囲内で対応可能であれば何の問題もないかもしれません．しかし，例えばシリアル通信が高速で，頻繁に割り込みを発生させる場合，本来処理しなければならないアプリケーション・タスクに対して大きなオーバヘッドとなります．

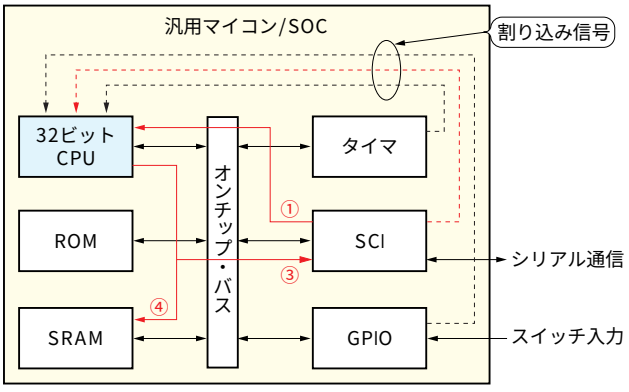

図1　32ビットCPUを搭載し，リアルタイムOSを搭載したシステムの例

SCI受信割り込みハンドラのデータの流れを示す．CPUでは，SCI受信バッファから受信文字を読み出し，受信可能文字か判定し，1文字のエコーバック（受信不可能文字の場合はエラー・コード），受信バッファへの格納，通信終了デリミタ検出，受信完了のイベント・フラグを立てるといった処理を行う必要がある．水色はバス・マスタ機能を有しているモジュールを表す．

そこで，もしSCI側で通信処理を行い，かつバス・マスタとしてSRAMに直接受信データを書き込める機能があったらどうでしょうか（図2）．このSCIに求められるデータのビット幅は8ビットですから，SCIにCPUを内蔵するとしても8ビットで済むことになります（SRAMへのフル・アドレス生成機能は必要）．

この場合32ビットCPUがSCIからの割り込みによって行う処理は，通信受信完了のイベント・フラグを立てるだけになります．OSの内部状態を変えるこの処理だけは必要ですが，言い換えればSCIにシーケンス処理＋バス・マスタ機能［こちらはDMA（Direct Memory Access）コントローラで実現してもよい］を追加するだけで，CPUに対して最少のオーバヘッドを実現できるわけです．

● 事例2：二つのCPUに機能を分散して処理する

一定周期（数百μs）で割り込みによってA-Dコンバータからアナログ情報を読み出し，所定の処理後PWM（Pulse Width Modulation）制御をかけるシステムの例を図3に示します．システム全体のシーケンス制御は数十ms単位でよく，高精度な処理も必要ありません．

CPUが一つの場合，数百μsごとの32ビット処理の余力（10％～20％）を使ってリアルタイムOS＋シーケンス制御のアプリケーション・タスクを実行させることになります．このため，PWM制御が重くなるにつれてシーケンスの応答性確保に影響が出ることになります．

その解決策が図4です．高精度でかつ高速な応答が求められる部分を32ビットCPUの処理系で構成します．決められた処理だけで，余計なシーケンス処理を担当しないの

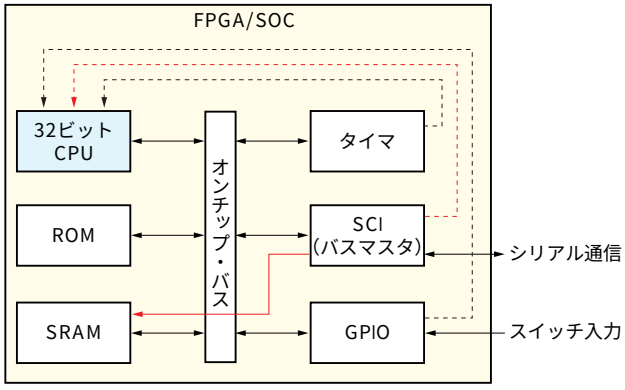

図2　SCI部でデータ処理を行い，バス・マスタ機能を備える

SCIからの割り込みによって行う処理は，通信受信完了のイベント・フラグを立てるだけになる．水色はバス・マスタ機能を有しているモジュールを表す．