第6章　FPGAのブロックRAMを使ったバッファリング機構を搭載 FPGAによるMMCカード・コントローラの設計事例

山武 一朗

第4章は，ほとんどの処理をソフトウェアで制御することで，可能な限り簡単な外付けのハードウェアでMMC（MultiMedia Card）カードをマイコンにつなぐという事例について述べられています．性能を要求しないのであればこれでも十分ですが，よりデータ転送レートを上げたい場合や，CPUに対する負荷をもっと減らしたい(CPUパワーをほかの処理にまわしたい)場合には，MMCカードの制御に特化したハードウェアが必要になります．

MMCカードのアクセスには，SPIモードとMMCモードがありますが，ここでは第4章で作成したMMCカード・ドライバと比較対比できるように，SPIモードに対応したコントローラを設計します．第4章で作成したソフトウェアを主とした処理事例を踏まえ，どの処理をどうやってハードウェアに置き換えるかを考察しながら，FPGAを使ったMMCカード・コントローラの設計事例について解説します．

1　SPIモードのMMCカード・アクセスの実際

実際のコントローラの設計に入る前に，第4章で作成したドライバで，どの処理に時間がかかっているかを調べてみましょう．

● 数百μs～1msのビジー待ち

図1にセクタ・リード/ライト時のMMCの各信号のようすを示します．1セクタ分のアクセスを時間軸を縮めて表示しているので，クロックやデータの詳細は判別できません．しかし，**図1**(**a**)のセクタ・リードでは，コマンドを発行してから実際にセクタ・データが出てくるまでの間Doutが“H”レベルになっていることがわかります．また，**図1**(**b**)のセクタ・ライトでは，セクタ・データの書き込みの最後から次のセクタのコマンド発行までの間Doutが“L”レベルになっています．

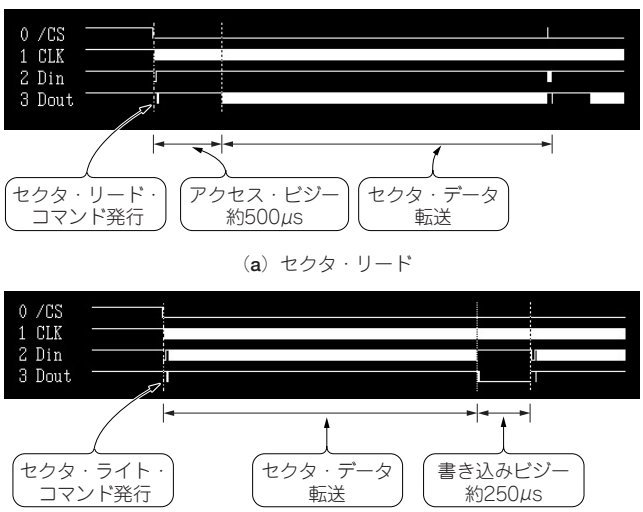

図1
セクタ・リード/ライト時の各信号のようす

Column 1
さらなるCPU負荷の低減

今回はMMCカード・コントローラの学習用ということもあり，システム・バス側のデータ転送をあくまでもCPU転送で行う事例を紹介しています．

しかし，コントローラを接続するバスとしてPCIバスを使う場合は，バス・マスタ転送対応に設計することで，データ転送にCPUパワーを使うことがなく，よりCPU負荷の軽いコントローラを実現できます．また組み込みマイコンのローカル・バスでも，マイコン内蔵のDMAコントローラと連携させることで，CPUの力を借りずにデータ転送を済ませることもできます．

もっとも，もともと絶対的な転送レートが遅いSPIモードで，そこまでパフォーマンスを追求しても，あまり報われないという話も聞こえてきますが….

これらは，セクタ・リードのアクセス待ち時間，およびセクタ・ライトの書き込み完了時間を示しており，この波形の計測で使用したカードの場合はそれぞれ数百μsの時間となっています．経験的には，高速性をうたう最新のカードほど時間が短く（アクセスが速い），容量が大きいかまたは古いカードほど長い（アクセスが遅い）傾向が見られます．

その結果，非常に長い間ビジー待ちが発生します．これをそのままソフトウェア・ループでCPUが待っていたのでは，CPUパワーのむだづかいとなります．ビジー状態が解除されたことを，割り込みなどでCPUに通知できるようなハードウェアが望ましいと考えられます．

● データ転送処理――クロック同期式シリアル通信

もっともスタンダードなMMCカードの仕様では，クロック周波数は最高20MHzと規定されています．クロック同期式シリアル通信なのでスタート・ビットやストップ・ビットはありません．したがって，データ転送時は単純計算で最大2.5Mバイト/秒の転送レートになります．

このコントローラを接続するバスとして，かりにクロックが20MHzのバスを考えると，そのまま8クロックで1バイトの転送ができます．さらにデータ・バス幅が32ビットと広い場合は，1バイト単位の転送はバスの使用効率が悪くなるので，4バイト分をまとめて転送したいところです．すると32クロックで4バイト（32ビット/1ワード）の転送が行えます．

しかし1ワードのデータ転送をするために32クロックもの間，バスを占有してしまうのはもったいない話です．かといって32ビット/1ワードごとに割り込みを使ってCPUに通知するとなると，

$$\frac{1}{20\text{MHz}/(8\times 4)} = 1.6\mu\text{s}$$

となり，1セクタ512バイトの転送を行いたい場合，非常に短い時間に割り込みが128回も連続して発生することになります．

● 転送レートの差を埋めるバッファを活用

このような場合は，低速バスと高速バスの間にバッファを設けて，高速バス側はある程度のデータが溜まったら転送処理を開始することで，空いた時間はほかのリソースにバスを使わせることができます．低速バス側はバッファとの間で一定レートで転送を続けることができます．

そこで今回設計するMMCカード・コントローラでは，この速度差を埋めるバッファを内蔵させて，システム・バス側を効率よく使用できるように考慮してみます．幸いなことに，最近のFPGAはメモリを内蔵しているものが増えているため，数Kバイト程度のバッファ・メモリであれば外付けメモリを必要とせず，内蔵メモリだけでバッファを実現できます．

● 32ビット幅のシステム・バスを想定

さらに今回は，PCIバスや32ビット・ローカル・バスに対応した評価ボードなど，データ・バス幅として32ビットのシステムにMMCカード・コントローラを接続することを考えます．

第4章で作成したMMCカード・ドライバは，I/Oドライバ部とのやり取りはすべてバイト単位の送受信関数となっています．これをこのまま使うと，32ビット幅のデータ・バスを使ってバイト単位でコマンドやレスポンスを受け取ることになり，効率がよくありません．

そこで，I/Oドライバ部の構造を大きく変更し，広いバス幅を生かしてMMCコマンドや引き数を一度にレジスタに設定する構造を採用することにします．制御レジスタの仕様の詳細については後述します．

● SPIモード専用/CRC非対応

ここで設計するMMCカード・コントローラは，基本的には第4章で設計したMMCカード・ドライバの処理のうち，CPU負荷の高い部分をハードウェア化するという方針で設計するので，対応モードはSPIの