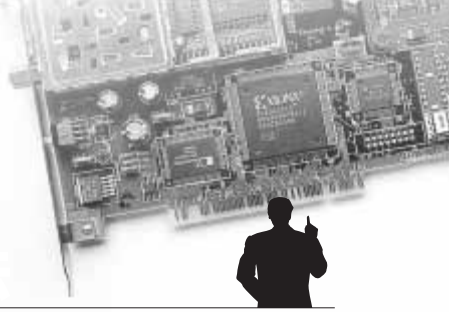
IP 活用事例

FPGAによるPCIバス・インターフェースの実現



横川 宏

PC組み込み用のカード（基板）の開発に際し，バスに何を使うかは悩むところです．データ転送速度がISAバスで十分であっても，PnP（プラグ・アンド・プレイ）によるリソースの管理が容易であることや，今後のPC仕様（PC99やPC2001）ではISAバスが完全に排除されることから，実質上PCの標準バス・アーキテクチャとなったPCIバスを選択することが非常に増えています．PCIバスは，ISAバスより速度的にも機能的にも優れており魅力あるバスですが，反面，仕様が複雑でもあります．そのため，PCIバス・インターフェースの設計は，相当敷居が高いことも事実です．さらにこれをFPGAによって実現するとなると，速度の点で大きな壁にぶつかりそうです．ここでは，FPGAベンダ提供のPCI用IPコアを利用することで，比較的容易にPCIデバイスが実現できることを紹介します．

1.FPGAとIPコアの導入

☑PCIコントローラ・チップを使えば

量産品ではなく，評価用のPCIカードを開発しようと考えた場合，安価で確実に，かつ短期間でとなると，市販のPCIコントローラ・チップを使うことが多いでしょう．システム設計者としては，“PCIバス・インターフェース回路”はあくまでも一つのデータ転送手段でしかなく，できるだけユーザ回路の設計に力を注ぐためにも，PCIバス周辺設計には時間を割きたくありません．PCIコントローラ・チップを利用すれば，バスの基本的な動作は保証されるし，動作検証の時間も最少に抑えられます．

☑PCIコントローラ・チップが使えないとき

ところが，「デバイスを1チップで」という要求が入っていると，判断を変えなくてはなりません．量産品でもない評価用にいきなりASICはリスクが大きいですし，システム的な実機検証も事前に行いたいところです．となると，おのずと方向は決まってきて「FPGAやCPLDを使って1チップに納める」というところに行き着きます．

しかし筆者らの場合，この選択には大きな問題がありました．第一にPCIの仕様を熟知した手すきの技術者がいませんでした．調べれば調べるほどPCI規格は複雑であることがわかり，詳細な設計ができるレベルに到達するには相当時間が必要で，規格を理解する作業だけで予定している試作期間が終わってしまいそうでした．

第二に，FPGAでPCIの要求する動作速度が本当に実現できるのか疑問でした．雑誌に掲載された設計記事などを見ると，PCI規格のプロトコルに準じてそのタイミング仕様を満たすのは簡単ではない，との指摘が散見されます．PCIで要求される33MHzが，FPGAで実際に実現できるのか心配だったのです．

☑IPコアの利用

そこで検討したのがIPコアの利用です．IPを利用することによって，PCIバスの制御回路部分の100%を完成させられるわけではありませんが，かなりの設計期間の短縮が図れると期待できそうでした．さらに特定FPGA用にチューニングした状態で提供されるため，動作速度の問題も解決しやすいように思えました．

PCI用のIPコア（以下PCIコア）は数社から提供されていますが，今回はXilinx社のものを選択しました．選択の理由は社内事情などいろいろあるのですが，「ユーザ回路とともに1チップに納めることが可能」ということが最大のポイントでした．

ユーザ回路のためにどれくらいのサイズをFPGA内に確保できるのかはたいへん重要です．つまりその大半をPCIコアで占有されていたのでは，ユーザ回路を入れ込む余地がありませんし，複数チップになってしまったのでは既製のPCIコントローラ・チップを利用するのと変わりありません．

またXilinx社の場合，最終的にできあがったFPGAの回路情報をハード・ワイヤード化し，比較的安価なLSIを提供するというサービスも用意されています．試作機の評価後，将来ある程度のまとまった数を量産したいときなどは有効だと考えました．

現在では，PCIコアの選択肢はかなり広がっています．選択に当たっては，各ベンダごとに特徴があるので，要求条件と照らし合わせて十分比較検討するべきだと思います．

2.前提となる基礎知識

PCI仕様

PCI仕様をおおまかに分類すると，

①バス信号の電気的仕様

②バス信号のタイミング仕様

③基板レイアウトなどの機械的仕様

④プロトコル仕様

になります．PCI仕様の詳細については本稿では解説しませんが，今回のようにFPGAとPCIコアを使用する場合，①の電気的仕様と②のタイミング仕様については，PCI仕様に準拠したFPGAを選択すれば，あとはPCIコアが制御してくれるので，設計者はほとんど気にする必要はありません．③の機械的仕様は，PCIバスに接続する信号のピン配置は前もって適切に配置されているので，パターン長を不用意に長くしなければ問題は生じません．一番たいへんなのが④のプロトコルです．

問題はPCIとユーザ回路の間のインターフェース

PCIコアを利用したとしても，PCIコアとユーザ回路の間のインターフェースは，既製のPCIデバイスのように単純には利用できません．既製のPCIデバイスは，内部にプロトコル用のシーケンサを固定パターンとしてもっているのに対し，PCIコアを利用する場合はこのシーケンサをユーザ側で目的に応じて設計する必要があるからです．逆にいえば，PCIコアを利用するほうが，それだけ自由度の高い設計ができるのです．基本的なプロトコルについては，PCIコアが自動的に応答してくれる部分もありますが，PCIデバイスとしての総合的なプロトコルの実現については，ユーザ回路側の機能設計に大きく左右されます．仕様を満足するプロトコルの実現には，PCIコアを利用するといえどもPCI規格をよく理解しておく必要があることになります．

残念なことにPCI規格は，相当高度でありながら自由度もあわせもった規格になっているため，デバイス設計のための情報を具体的にすべて解説してくれるような参考書はなかなかないようです．われわれが設計時に参考とした書籍のリストを稿末に記載しておきますのでご参照ください．

実際は書物だけでは知り得ない情報も多数あります．PCIデバイスの相手側，つまりメイン・ボード上のPCIチップセットの動作です．チップセットの違いによってバスの動作が異なる場合があるし，BIOSの設定内容によっても動作は違ってきます．事前に調査できる場合はいろいろ情報を集めておいたほうが手戻りが少なくてすみます．

FPGAアーキテクチャ

FPGAの場合，クロック線のスキュー，配置できるレジスタの数や場所，およびこれらから派生する配線遅延といった重要な課題があります．PCIのように高速かつ多ビット幅バス回路の実現では，それなりの努力が必要です．

まずは，FPGAのアーキテクチャについて十分知ることが重要です．アーキテクチャを理解していれば，FPGAの構成要素であるCLB(Configurable Logic Block)を有効に利用し効率よく配置配線することが可能となり，上記の課題を解決しやすくなります．再利用性の点ではあまり好ましいことではありませんが，FPGAの構成要素であるIOB(Input/Output Block)内のフリップフロップを有効に活用するといったような，Xilinx社のFPGAアーキテクチャに特化する論理記述を行うことが必要になってくる場合もあります．

さらに，特定のフリップ・フロップからフリップ・フロップへのタイミング条件を規定する「タイミング制約」付けも，ユーザ回路の動作速度を満足するための重要なポイントになります．

ツール＆設計環境

IPコアを利用して設計を進めるにあたっては，論理シミュレータ，論理合成ツール，そしてFPGAへの配置配線ツールなどが必要ですが，IPベンダは，そのIPを利用するにあたって前提となる環境を「推奨」という形で指定しています．

指定されたツール以外では動作を保証しない，というところまで厳密に制約を受けるわけではありませんが，サポートの面やマニュアルに記載されている手順は推奨される環境に基づいているので，可能であれは推奨ツールを用いたほうがよいでしょう．推奨外の環境では，動作の確認がされていないようなので，細かな動作が保証されにくいということになります．じつは，筆者らも開発開始当初は推奨環境とは異なるツールで設計を進めていた時期があったのですが，とくに論理合成において，

- ソース上に記述したレジスタ名がネットリストに反映されない場合がある
- モジュールの階層構造を保持できない(フラットに展開されてしまう)

などの問題があって，ツールを変更した経緯があります．この問題は「タイミング制約ファイル」の作成時に非常に大きな障害となりました．ツール類を熟知された方なら問題ないかもしれませんが，新規に環境を作られる場合は推奨の環境を構築されることをお勧めします．