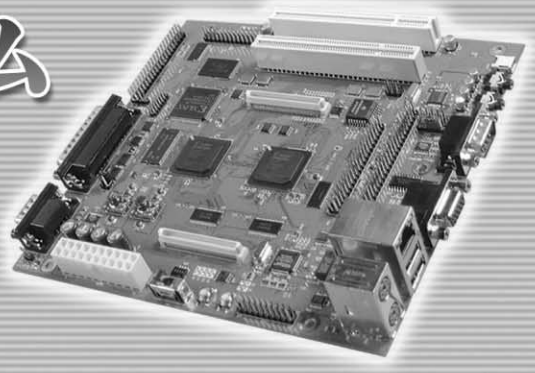
コンピュータ・システム技術学習キット活用通信



山武 一朗

第9回（最終回） M32Rソフト・コア用プログラム作成事例

前回はM32Rソフト・コアをA/Vプロセッサに実装するところまで解説しました．今回はそのハードウェアの上で動作するソフトウェアの作成方法について解説します．

1 M32Rソフト・コアのプログラム・デバッグ環境

● JTAGデバッグ環境を用意

前回の連載で掲載したブロック図でもわかるように，本キットに添付予定のM32Rソフト・コアからはCPUデバッグ用のJTAG信号が出力されています．この信号はJTAGとは呼びますが，FPGAのJTAG端子とは別物の端子である点に注意が必要です．本評価ボードではA/VプロセッサやI/Oプロセッサなどの FPGAのJTAG端子がカスケード接続されていますが，ここにソフトCPUコアのJTAG信号を接続することはできません．

このJTAG信号をFPGAのI/Oピンに接続してコネクタとして引き出せば，市販されているM32R用のJTAGデバッガを接続してデバッグすることができます〔**図1**(**a**)〕．

さらに本キットでは，より安価にデバッグ環境を構築できるよう，M32Rソフト・コアのJTAGを，PC/AT互換機のLPTポート経由で制御できるしくみを用意しています〔**図1**(**b**)〕．

● Linux/Cygwin/Windowsネイティブの各環境に対応

本キットには，M32Rソフト・コア用のソフトウェア開発環境として，Linux用と，Windows＋Cygwin用のGCCとGDB，およびSDIServer（後述）が用意されています．また現在，Cygwinなどのインストールを必要としないWindowsネイティブ環境用の開発環境も準備中です．

今回はWindows＋Cygwin用の環境を使いました．なお，Cygwinインストール時には，ipermもインストールするようにしてください（SDIServerがLPTポート・アクセス時に必要）．パッケージの選択でdevelをフルインストールすれば，いっしょにインストールされます．

● M32R用GDBをJTAG接続

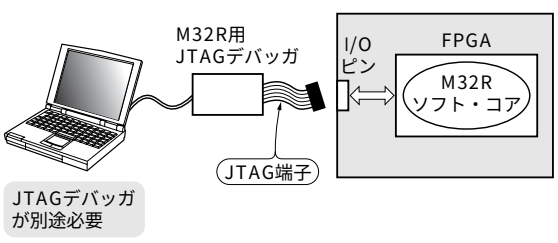
本学習キットに添付されている開発環境の特徴として，M32Rソフト・コアをJTAGを使ってデバッグできるという点が挙げられます．

SDIServerとは，PC/AT互換機のLPTを経由してM32Rソフト・コアのJTAGを制御するために必要なソフトウェアです．SDIServerはGDBプロトコルを理解するので，M32R用のGDBを起動しSDIServerと接続することで，M32Rソフト・コアのJTAGデバッグが可能になります．**図2**にM32R用GDBの起動時のメッセージ表示のようすを示します．

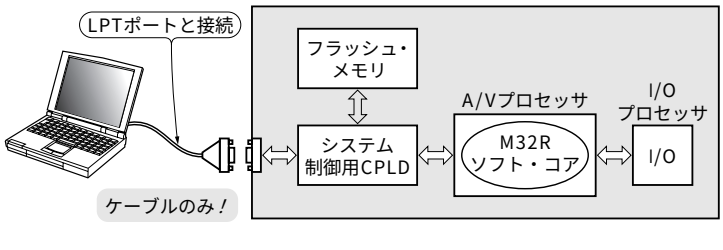
● M32Rソフト・コア評価版の制限

本学習キット添付のM32Rソフト・コア評価版の制限として，起動後連続十数時間が経過するか，SDIServerと接続状態でないとCPUコアが停止するという点があります．SDIServerはPC/AT互換機上で動作するソフトウェアなので，ターゲット・ボードを必ず開発用PCと接続した状態でないと，CPUコア上でプログラムを走らせることができないわけです．

評価版IPコアの制限として一般的なのは，インストールから数か月でライセンスが切れるといったものや，内部でクロッ



（**a**）CPUコアのJTAG信号をFPGAのI/Oピンに出力する



（**b**）PC/AT互換機のLPTポート経由でCPUコアのJTAGを制御する

図1 M32Rソフト・コアのJTAGデバッグ環境

KEYWORD ―― M32Rソフト・コア，JTAGデバッガ，GDB，SDIServer，割り込みプログラム

```
$ m32r-linux-gdb[リターン]
GNU gdb 6.4.50.20060110-cvs
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "--host=i686-pc-cygwin --target=m32r-linux".
(gdb) target m32rsdi[リターン]
Remote m32rsdi connected to M32R/VDEC2_CQ
(gdb)
```

図2
M32R用GDBの起動のようす

クをカウントして，起動後30分や1時間といった時間が経過したら停止するというものです．しかしこの制限では，学生や技術者の教育カリキュラムでの採用が難しいといった問題や，規定時間以上の連続動作ができないため，非常に使いにくいものがあります．

しかしCPUコアの場合，しかも開発/評価用を前提としたものであれば，必ずコンパイラ/デバッガなどのソフトウェア開発環境をターゲット・ボードと接続した状態で使うのが一般的です．本M32Rソフト・コア評価版では，このスタイルで使用している限りは，起動後の連続動作時間も十分長いので機能的な制限はほぼないと言ってもよいでしょう．

2 M32Rソフト・コア用サンプル・プログラムの作成

今回は，連載第4回と第5回(2006年5月号，6月号掲載)で紹介した，LEDの点灯制御とディップ・スイッチの状態の読み

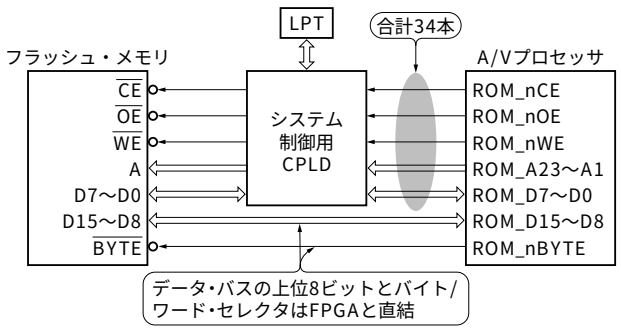
Column システム制御用CPLDの設計

図1(b)を見ると，LPTの各信号は直接FPGAには接続されておらず，システム制御用のCPLDを経由していることがわかります．この接続形態は評価ボードの回路構成上，変更することはできません．

M32RソフトコアのJTAG信号は合計6本の信号が必要になります．しかし，システム制御用CPLDとA/Vプロセッサの間は，CPLDの先に接続されているフラッシュ・メモリにアクセスするために必要とされる信号しか接続していません(**図A**)．

そこで，A/VプロセッサにM32Rソフト・コアを使い，JTAG信号をLPT経由で制御する場合は，システム制御用CPLDとA/Vプロセッサの間の信号を**図B**のように変更します．

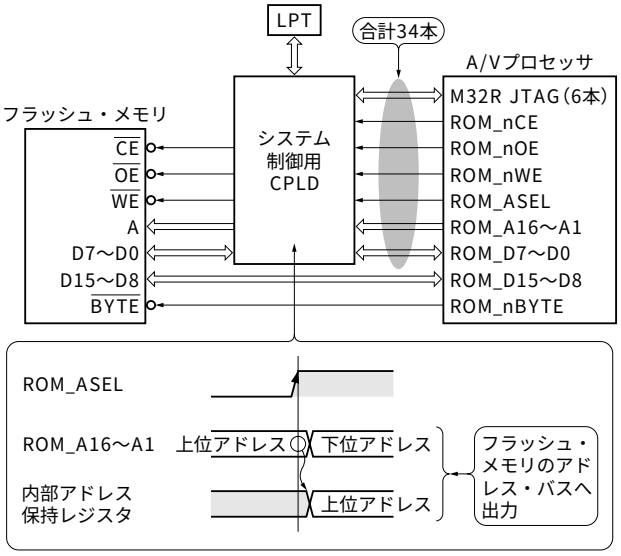
標準的な使い方では，アドレス・バスはA23～A1(フラッシュ・メモリは16ビット幅で使うため)の23本ありますが，これをアドレスの上位と下位の2回に分け，時分割で出力するようにします．



図A　システム制御用CPLDとA/Vプロセッサ間の標準的な接続信号

今回はアドレス・バスとしてROM_A16～A1と，アドレス上位/下位選択信号としてROM_ASELという信号を割り当てました．そして，残りの信号6本をM32RソフトコアのJTAG信号として割り当てました．

M32Rソフト・コアを実装するA/Vプロセッサのフラッシュ・メモリ・アクセス回路やI/Oピン定義を変更するのはもちろんですが，システム制御用CPLDの中身も書き換える必要があります．しかし本学習キットは，システム制御用CPLDのHDLソースも公開しているので，CPLDとA/Vプロセッサの間の接続信号の変更も自由に行うことができます．



図B　M32Rソフト・コアをLPT経由でJTAG制御する場合の接続信号