Design Wave

Linuxで動く
ビデオ再生システムを作る

FPGAキットで始める ハード&ソフト 丸ごと設計

RTLとCソースを全公開!

見本

CPUと周辺回路を作り込んでCプログラミング

栗元 憲一 著

CD-ROMコンテンツ

CD-ROM付属

1. ビデオ再生システム ハードVHDLソースコード
2. ビデオ再生システム ソフトCソースコード
3. Aeroflex Gaisler社製 IPコア群 GRLIB
4. SnapGear Linux

本書の内容を試せるFPGAボード(別売)

GR-XC3S-1500
(Pender Electronic Design社)

Spartan-3 XC3S1500搭載

BLANCA(CQ出版社)

Spartan-3 XC3S1500搭載

NEEK(Altera社)

Cyclone Ⅲ EP3C25F324搭載

CQ出版社

第1部

第1章 オープンソース ソフト&ハードによるFPGAへのSoCシステムの実装

SPARC V8 アーキテクチャ LEON システムの概要とサポート・ボードへのマッピング

ソフトウェアの世界では，Linuxなどのオープンソース・モデルが大きな成果を収め，私達の身の回りの商品にも多数用いられるようになりました．ハードウェアにおいてもオープンソース・モデルの活用が行われており，成果を挙げつつあります．第1部では，オープンソース・ハードウェア（RTLソース・コード）を使用して，読者の手元でFPGA上にLinuxシステムを構築する方法を解説します．

1.1 オープンソース・ハードウェアとLEONシステム

●オープンソース・ハードウェア

ソフトウェアは，いったん開発されたものは，インターネットを用いて非常に安い金額で個人が配布することができます．それに対してハードウェアは，実際に製造するために大きなお金がかかるため，オープンソース・モデルが簡単には適用されにくいとされていました．

しかしFPGAの発展により，RTLソース・コードで設計データを配布する形式のオープンソース・ハードウェアはソフトウェアと同等の配布コストとなり，個人が利用し配布することが可能になりました．

最近のFPGAボードは，個人が購入できる金額でシステム全体を構成できるような高性能なものになっており，今後オープンソース・ハードウェア・モデルは大きな成果を挙げる可能性があります．

現在，様々なRTLソースコードのオープンソース・ハードウェアを展開する活動が行われています．例えば，OpenCores（http：//www.opencores.org/）には，多数のオープンソースIPコアのRTLソース・コードが公開されています．

また，OpenSPARC（http：//www.opensparc.net/）では，最先端の64ビット・マルチコア・プロセッサのRTLソース・コードが公開されています．

そこで本書では，Aeroflex Gaisler社（http：//www.gaisler.com/）からGPLライセンスでRTLソース・コードが提供されているLEON3プロセッサと周辺IPコア（GRLIB）を用いて，FPGA上にLinuxシステムを構築します．

●LEONプロセッサと周辺IPコア（GRLIB）について

最初のLEONプロセッサは，ESA（European Space Agency：欧州宇宙機関）で開発され，1999年に発表されました．宇宙で使用される半導体は，アルファ粒子放射線や宇宙放射線の影響でメモリやレジスタの値が変更されてしまうSEU（Single Event Upset）現象が起きる可能性があります．LEONプロセッサは，宇宙でも使用できるようにSEU対策を行ったプロセッサとして開発され，GNU general public licenseで公開されました．アーキテクチャとして最もオープン化に積極的なSPARCを採用しています．

その後，チーフ・デザイナであったJiri Gaisler氏はESAを離れAeroflex Gaisler社を設立し，LEONの開発を続けています．第2世代のLEON2は，SEU部を取り除いたものがLGPLライセンスでソース・コードが公開されました．その際には，プロセッサだけでなく，周辺IPも接続された形で公開されています．

現在，第3世代のLEON3プロセッサがGPLライセンスで公開されています．

●LEON3プロセッサの特徴

LEON3プロセッサの特徴として，次のようなことが挙げられます．

(1) SPARC V8アーキテクチャ準拠のASICとしての動作実績

LEON3プロセッサは，FPGAのみならず，ASICとしても動作実績があるプロセッサです．SPARC Internationalより，公式にSPARC V8アーキテクチャ準拠の承認を取っており，SPARC向けのバイナリが用意された環境なら問題なく動作します．例えば，Linux，uClinux，eCos，RTEMS，OSなしなど，多数のソフトウェア開発環境が揃っています．ソフトウェア，ハードウェア共にオープン・ソースでありながら，高い信頼性があります．

第2章 オープンソース ソフト&ハードによるFPGAへのLinuxシステムの構築

LEONシステムで動作するLinuxのビルドと非サポート・ボードへの移植の基本

2.1 Linuxイメージのビルド

●Linuxコンフィグレーション

ハードウェアの準備が整ったので，次はソフトウェアを準備します．第1章で設計したハードウェア上で実行するLinuxイメージを生成します．ここでは，主要な設定項目について説明します．

まず，カレント・ディレクトリを$LEON_HOME/snapgear-2.6-p42へ移動します．

```
make xconfig
```

とコマンドを打ち込むと，Linuxコンフィグレーション GUIが立ち上がります(図2.1)．Vender/Product Selectionボタンをクリックすると，Vendor/Product選択GUIが立ち上がり，Vendorはgaisler，Productはleon3mmuが選択されています．ここはそのままにして，Nextボタンをクリックします．

次に，Gaisler/Leon2/3 MMU options GUIが立ち上がります(図2.2)．最初の項目はシステムのMUL/DIV命令用ハードウェア実装の有無を入力します．今回は，これらの機能を実装したのでyを選択しています．

次の項目は，FPUのハードウェア実装の有無を入力します．これらの項目によって，コンパイラに与えるオプションが変わります．ハードウェアを実装していない場合は，もっと小さな命令の複合に置き換えられます．

また，今回はSDRAM上にファイル・システムを持つので，Initial root filesystemにはinitramfsを選択します．ネットワーク上の他のコンピュータにファイル・システムを持つNFS (Network File System)を使用することもでき，その際にはNoneを選択します．NFSを使用する設定の詳細は，SnapGear Linux for LEON Manualを参考にしてください．

カーネル起動時に与えるコマンド・ラインについては，今回はSVGAコントローラを通してフレーム・バッファにコンソールを出力するために，次のようなコマンドを指定しました．

```
console=tty0,38400 video=grvga:640x480@60,16,614400
```

画面は，VGAサイズで16ビット・カラーです．カーネル・コマンド・ラインの詳細をコントロールしたい場合は，SnapGear Linux for LEON manulを参照してください．

Nextボタンをクリックすると，Kernel/Library/Defaults選択GUIが立ち上がります(図2.3)．Kernel Versionは，Linux-2.6.21.1を選択します．また，Libc Versionは，glibc-from-compilerを選択します．そして，Customize Kernel SettingsとCustomize Vendor/User Settingsをyにします．

これらを選択することにより，それぞれKernelコンフィグレーションGUIとアプリケーション・ソフト

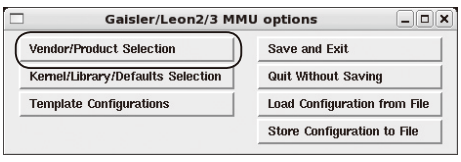

図2.1 Linuxコンフィグレーション GUI

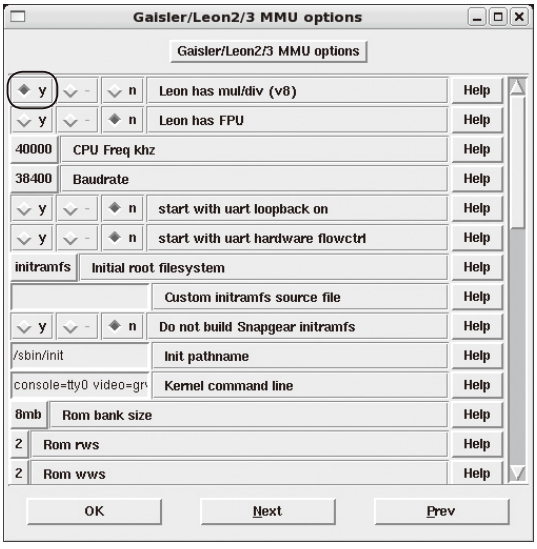

図2.2 MMU options コンフィグレーション GUI

第3章 motionJPEG再生システムを例にFPGAによるSoCを開発する目的

オープンソース ソフト&ハードにより SoC を開発するメリットと第 2 部の構成について

第2部では，いよいよ本格的にFPGAを使ってSoCの開発を行います．本書で開発するシステムは，ネットワーク経由でmotionJPEGファイルを読み込み，それをデコードして画面に表示するという，motionJPEG動画ストリーム再生用SoCです．まず，ソフトウェアだけでJPEGファイルをデコードし，その速度を体感します（第4章）．次にJPEGデコード処理の一部をハードウェア化してみます（第5章）．実際に，JPEGデコード処理全体をハードウェア化することにより，どれだけシステムを高速化できるかを体験してください（第6章）．最後に，システムをネットワーク対応にすることで，motionJPEG動画ストリーム再生システムが完成します（第7章）．

第1部では，オープンソースCPUであるLEON3をFPGAにマッピングし，その上でオープンソースOSであるLinuxを載せたシステムを動作させました．ここからは，オープンソース・ハードウェアが自由に改変できるという利点を用いて，SoCの開発を行います．

3.1 motionJPEG 再生システムを例に

●FPGAでSoCを開発できる！

最近のLSIは，複数のプロセッサが搭載されてOSが動作する非常に複雑で高度なSoC（System on Chip）となっています．したがって，SoCの開発は，とても個人でできるものではありませんが，半導体技術の進歩はFPGAの低価格化，大容量化ももたらしてくれました．そのため個人で購入できる金額のFPGAボードに，小さなSoCを搭載することが可能になりました．

ここでは，SoC開発のエッセンスを体験するために，図3.1のようなネットワーク越しのmotionJPEG再生システムをFPGA上で開発します．

第2部では，次の2点を目標としています．

(1) ソフトウェアで処理の間に合わない部分をハードウェア化するというSoC開発のエッセンスを学習する

(2) 一つのシステムをオープンソースのSoC，OS，ソフトウェアを使用して実現することにより，それらの使用方法を理解し，今後読者が自由に使いこなして独自のSoCを開発できるようになる

現在，多数のSoCが開発されていますが，その理由にはさまざまなものがあります．例えば，複数のチップで構成されるシステムをより微細化したプロセスを使い1チップにまとめてローコスト化を進めたものや，低消費電力なシステムを作るためにソフトウェアではなくハードウェアで実現したものなどがあります．

また，新機能を実現するシステムを開発する際に，ソフトウェアでは処理が間に合わない部分をハードウェア化したSoCを開発することは，最もよくある理由の一つです．

●ハードウェア化による高速化を体感できる！

最新のLSIは，微細化により性能が向上しているためmotionJPEGをソフトウェアのみで容易に再生できますが，かつてコンピュータ上で動画を扱うことが可能になりはじめた時期には，ハードウェアでサポートすることにより実現していました．

時を経て，motionJPEGがソフトウェアで扱えるようになるとともに，FPGAがその当時のシステムすべてを実現できる規模になりました．当時のLSIのように，FPGA上でソフトウェアによるJPEG処理では厳しい部分をハードウェア化することを体験することにより，他の同様な問題にも通じるエッセンスを学ぶことができます．そして，ハードウェア化を行う部分を追加していくごとに，動画のフレームレートが上昇することで，ハードウェア化による高速化を体感するこ

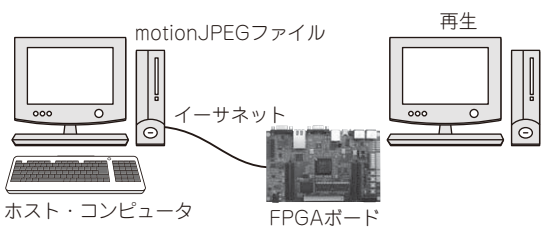

図 3.1 motionJPEG ストリーム再生システム

第4章 JPEGアルゴリズムとソフトウェアによるmotionJPEG再生の実現

JPEG デコード・アルゴリズムを理解し，IJG ライブラリを使ったソフトウェアによるデコードを体感しよう

本章では，JPEGのソフトウェアとして業界標準となっているIndependent Jpeg Group（IJG）のライブラリを使用して，motionJPEGをFPGA上で再生することを目標とします．

4.1 JPEGのアルゴリズム

まず，JPEGのアルゴリズムの概要について説明します．実際に開発するのはデコード・システムですが，JPEGは画像データを効率よく圧縮するために開発されたアルゴリズムなので，ここではエンコードのアルゴリズムについて説明します．

デコードは，このアルゴリズムを逆方向に行えば実現できます．

●RGB形式からYCbCr形式へ

図4.1に，JPEGエンコードの概要を示します．最初に，入力されたRGB形式の画像をYCbCr形式のフォーマットに変換します．これは基本的に，

$$Y = 0.29900R + 0.58700G + 0.11400B$$

$$Cb = -0.16874R - 0.33126G + 0.50000B + 128$$

$$Cr = 0.50000R - 0.41869G - 0.08131B + 128$$

という式で表せるもので，単に数値の表現方法を変更するものです（JPEGではレベル・シフトが同時に行われる）．

ただし，人間の目にはY成分にだけ敏感であるという特徴があるため，データのサンプリング時にCbとCr成分に対して間引くことが可能になります．この間引いたY：Cb：Crの割合によって，4：4：4や4：2：2，4：1：1と呼ばれるJPEG画像が存在します．Y成分に対して，同じ割合でサンプリングする画像を4：4：4としています．図4.2に，4：2：2の例を示します．

この図の例では，画面がMCUと呼ばれる16×16の画素に分けられています．そして，Y成分は8×8が4個の細かさで変換されていますが，Cb，Cr成分は横二つ分を一つの値で代表させることによって8×8が2個で変換されています．これにより，情報が間引かれて圧縮されることになります．

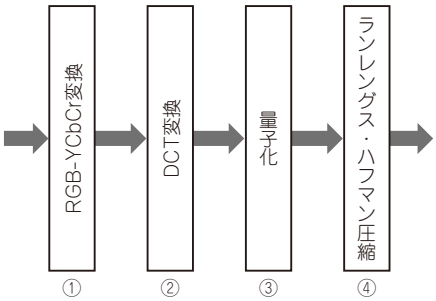

図4.1 JPEGエンコードの概要

この圧縮では，人間の目の特性によりCb，Cr成分の8×8が4個の場合（4：4：4）の画像と差はあまり感じません．

- 1次元離散コサイン変換（1D Discrete Cosine Transform，DCT）は，図4.3のような行列演算で表される
- 2次元離散コサイン変換（2DDCT）は，この変換を繰り返して実現できる

これは，図4.4の右上のような8×8の2次元波形の重ね合わせで，元の絵を表現する正規直交変換です．Y，Cb，Crの成分それぞれを，8×8の単位で変換します．

変換後に，図中の基底波形それぞれを，どれぐらいの割合で重ね合わせると元図形になるかを表す8×8の係数行列が求まります．

基底波形のうち，左上の部分は単一の値となり，DC係数と呼ばれます．そのほかの63要素は，AC係数と呼ばれます．

DCTの式は，実際に計算するときは図4.5のような小数の行列演算として求めます．

●DCT変換と量子化

DCT変換そのものは正規直交変換なので，情報の圧縮は全く行われません．それでは，何故このような計算量の大きな変換を行っているのでしょうか？ その理由は，最後に行われるランレングス・ハフマン圧縮において，人間の目ではあまり変化を認識できないが，

第5章 YCbCr-RGB変換モジュールをハードウェア化するシステムの開発方法

ソフトウェア処理の一部分を実際にハードウェア化することにより，SoC開発のエッセンスを体験しよう

本章では，SoC開発のための要素技術の習得を目的として，YCbCr-RGB変換部分をハードウェアで処理するシステムを開発します．

YCbCr-RGB変換のみをハードウェア化しても，システムの性能はそれほど上がりませんが，ソフトウェア処理の一部をハードウェア化してシステムを動作させることが体験できます．本章だけでも自分で開発を行えば，自分専用のSoCを開発するための要素技術を習得することができます．

習得する要素技術は，RTLによるハードウェア設計，AMBAバスの仕様の理解とインターフェース設計，ユーザランド・アプリケーションからAMBA接続されたハードウェアにデータを送るためのデバイス・ドライバ設計などです．

●本章で設計するハードウェア

本章で設計するハードウェアは，図5.1のように，YCbCr-RGB変換モジュールをAMBA AHBバスに追加したものになります．図5.1では，ここで開発するシステム内でのデータ処理を示しています．ハフマン(Huffman)デコードや2DIDCTは，これまでと同じようにソフトウェアで処理されます．

YCbCr-RGB変換部分は，2DIDCTが終わったデータをデバイス・ドライバによって追加したハードウェアに送ります．データを受け取ったYCbCr-RGB変換ハードウェアは，自律的にRGBデータに変換してメイン・メモリ中のフレーム・バッファにデータを送ります．

また，VGAコントローラは，プロセッサ処理とは関係なく自律してメイン・メモリ内のフレーム・バッファ領域のデータを読み取り，VGAポートからディスプレイ表示に合わせたタイミングで画面データを出力し続けています．これら三つの動作は，同時に起きていることに注意してください．

2DIDCTデータをYCbCr-RGB変換ハードウェアに転送した後，ユーザランドのアプリケーションは次のデータのハフマン・デコードを始めており，それと同時にYCbCr-RGB変換がハードウェアで行われています．ハードウェアによる処理はソフトウェアによる処理とは異なり，並列に処理を行うことができます．

5.1 AMBAバスの基礎知識

YcbCr-RGB変換ハードウェアを設計する前に，接続先のバスであるAMBAについて基本的なところを解説します．

●YCbCr-RGB変換回路の概要

ここでは，YCbCr-RGB変換ハードウェアの設計を行います． YCbCr-RGB変換は，次のような式で表される演算です．

$R = Y + 1.40200\,(Cr - 128)$

$G = Y - 0.34414\,(Cb - 128) - 0.71414\,(Cr - 128)$

$B = Y + 1.77200\,(Cb - 128)$

ハードウェアは，上記の演算を行うデータ・パス部分と，プロセッサからデータが書き込まれるAHBスレーブ・インターフェース部，このハードウェアが自律的にDRAMに計算結果を書き込むAHBマスタ・インターフェース部からなります．

LEONプロセッサ・システムは，32ビットのAMBAバスを採用しているので，ソフトウェアからこのハードウェアIPコアに送るデータも32ビット単位になります．しかし，JPEGのYCbCrデータは3×8ビット＝24ビットなので，今回は32ビット幅の24ビットのみを使用します．設計者が，バスのデータのどの位置にどのデータが来るかを決めます．

このハードウェアIPから出力する計算結果も，32ビット幅のバスから出力されます．今回のLinuxシステムは，16ビット/ピクセル(bps)のフレーム・バッファを使用しています（R成分：5ビット，G成分：6ビット，B成分：5ビット）．そこで，2ピクセル分をまとめて32ビットとし，バスに送出するハードウェアIPを設計することにします．データ・パス演算を行った後に，2ピクセル分をハードウェアでまとめます．以上のことからざっくりと，図5.2のような概要図を書くことができます．

第6章 JPEG処理をハードウェア化したシステムの開発方法

DCT処理やハフマン・デコードをハードウェア化することにより，さらに高性能なSoCを実現しよう

前章までの解説で，ソフトウェア処理の一部をハードウェア化し，システムの性能を上げるSoC開発のエッセンスと，必要となる要素技術をマスタできました．そこで，この章では，さらにハフマン・デコーダや2DDCTなどの処理をハードウェア化し，motion JPEGムービを再生できるシステムを開発します．

6.1 データの流れ

最初に，IJGのソフトウェアdjpegの中で，どのようにデータが流れているかを確認します．Structure.docに全体構造が説明してあります．

●djpegの構造

図6.1は，djpegの大まかな全体構造を示しています．main関数中でjpeg_read_scanline()を呼ぶことでJPEG処理を行っていますが，その実態は，process_data_simple_main()というAPI関数です．

Structure.docに説明がありますが，IJGのライブラリは一つの関数が前半処理と後半処理の二つの関数を呼び出すという構造をとっています．

process_data_simple_main()は，decompress_onepassというハフマン・デコードとDCT処理を行う関数と，post_process_dataというupsampleとYCbCr-RGB変換を行う関数をコールしています．これらの処理を順次ハードウェア化し，システムを高速化していきます．

前章までのハードウェアIPコアでは，前後のモジュールとのデータのやりとりには，FIFOの要素数を判定して信号を送っていました．この方法は，いわゆるハンドシェークと呼ばれるやり方です．

受信する側のモジュールが受信可能なときにready信号をアサートし，送信側に知らせます．送信側は，送るべきデータが揃っている場合に受信側からのready信号を確認し，受信側の準備ができていればデータを送ります．

データを送るときに，有効なデータを送信しているというvalid信号も同時に送ります．受信側は，受信可能なときにvalid信号がアサートされていれば，有効なデータだと判断して受信します．

●JPEGデコード・コアの全体構造

これから具体的に，DCTハードウェア，ハフマン・デコード・ハードウェアなどを開発して接続して行きます．それぞれのモジュールもハンドシェークを行い，データのやりとりを行います．図6.2に，これから開発するJPEGデコード・コア全体の構造を示します．これは，参考文献(4)を参考にしています．

モジュール間には，データを一時的に記憶するメモリが接続されます．FIFOまたはダブルバッファと呼ばれる方法で構成しています．

FIFOで前後のモジュールをつなぐ方法は，第5章ですでに説明しました．ダブルバッファは，図6.3に示すように，二つのメモリを置いて，それぞれを切り替

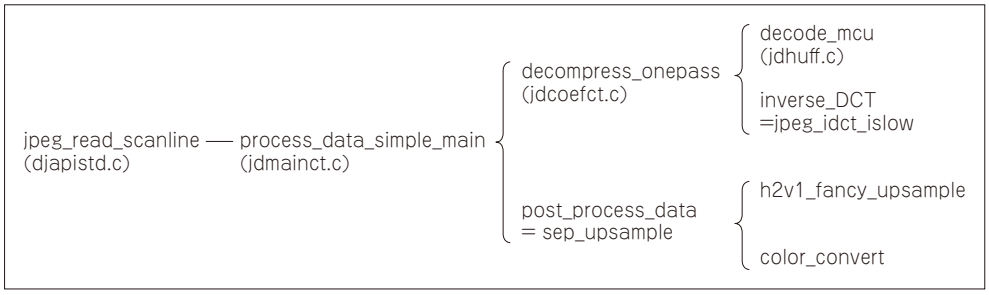

図6.1 djpegの全体構造

第2部

第7章 motionJPEG再生システムのネットワークへの対応

開発したFPGAによるSoCを実際に起動させて，ネットワーク経由でmotionJPEG動画ストリームを再生しよう

7.1 ネットワークへの対応

●動画サーバの準備

JPEG処理部分をハードウェア化するSoCの開発は，第6章までの説明で完了しました．本章では，Linuxが動作するSoCであることを利用して，ネットワークに対応させる方法について解説します（ただし，使用しているFPGA評価ボードがEthernetに対応している必要がある）．

具体的には，PC-Linux上でffserverという動画サーバ・プログラムを実行し，FPGA側で再生します．ffserverは，ffmpegに付属するプログラムです．

```
ffserver -f コンフィグレーション・ファイル名
```

でネットワーク動画サーバが立ち上がります．

コンフィグレーション・ファイルのサンプルは，ffmpegのdocディレクトリにあります（ffserver.conf）．そこで，その一部をリスト7.1のように書き換えます．そして，PC-Linux上の/home/kurimoto/LEON/ffserver/に，gfdl-qvga.mjpegを配置します．さらに，

```
ffserver -f ffserver.conf
```

と実行するとメッセージが出力され，動画サーバの実行がはじまり，アクセス待ちの状態になります．

まずは，ffserverが立ち上がっていることを確認するために，他のマシン（PC-LinuxがVMware上で実行されているなら，ホストのWindowsマシンでもよい）からWebブラウザでアクセスしてみます．

仮に，PC-LinuxのIPアドレスが192.168.24.51の場合，Webブラウザに以下のアドレスを打ち込みます．

```
http://192.168.24.51:8090/stat.html
```

すると，図7.1のような画面が表示されます．

また，gfdl-qvga.mjpegというリンクがあるので，これをクリックするとダウンロードされます．何らかのmotionJPEGビューワ・ソフトをヘルパ・アプリとしていれば，ストリームで再生されます．これで，動画サーバ側を立ち上げることができました．

リスト7.1　コンフィグレーション・ファイルの例

```
<Stream file.rm>
File "/usr/local/httpd/
     htdocs/tlive.rm"
NoAudio
</Stream>
```

(a) 変更前

```
<Stream video.mjpeg>
File "/home/kurimoto/LEON/
   ffserver/gfdl-qvga.mjpeg
NoAudio
</Stream>
```

(b) 変更後

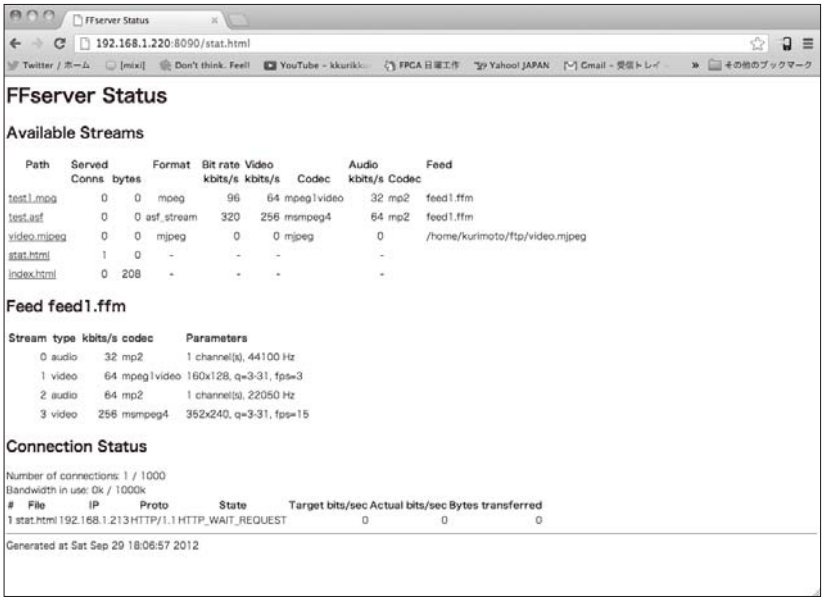

図7.1　Webブラウザからffserverへのアクセス

第3部

第8章 AMBA AHBバスの仕組みとModelSimによるシミュレーション

バスの基本概念とAMBAバスの基本仕様，AMBAマスタ/スレーブとシミュレータを使ったAMBAバスの動作を理解しよう

第3部では，第2部で開発したシステムの理解を深めるために，AMBA AHBバスやIPコアの仕様について解説します．第8章では，AMBAバスの概要について解説したあと，AMBAマスタとAMBAスレーブを組み合わせたシステムを想定し，シミュレータを使ってAMBAバスの動作を解説しています．第9章では，LEONシステムを構成する主なIPコアについて，重要な部分にポイントをしぼって解説します．取り上げたIPコアはLEONプロセッサ，Ethernetコントローラ，AMBAプラグ＆プレイ・コントローラ，SDRAMコントローラです．

LEONプロセッサの特徴として，業界標準のAMBAバスを採用しているという点があります．さらに，AMBAバスに接続できるIPコアを開発するための，AMBAマスタ・エミュレータなども開発環境と一緒に公開されています．

AMBAバスは，産業界では非常によく採用されているため，AMBAバスに接続できるIPコアの開発ニーズは非常に大きいと考えられます．

本章では，主にLEONシステムで用いられているAMBAバスの一つとして定義されているAHBバスについて説明します．

最近は，AMBAの最新仕様であるAXIを使用する例が増えていますが，AHBを使用したシステムもまだまだ使われています．AHBは，使用しているSoCの実際のソース・コードを見ながら理解していくことができます．

AMBAバスの仕様はオープンになっており，ARM社のWebサイトやAeroflex Gaisler社のWebサイトからダウンロードできます．しかし，この仕様書を一般的なバス・システムを知らない人が読んだだけでは，理解することは少し難しいと思います．そこで，本章では，Grlib（Aeroflex Gaislerが公開しているIPコアと開発環境）のAMBAバスに関連するソース・コードを使用してAMBA AHBの仕組みを理解します．

AMBA準拠の設計を一度理解してしまえば，それほど難しいものではないことが分かります．そこで本章の説明では，AHB部分のみに注目して深く掘り下げています．他のバス・システムも本質的には似たようなものです． LSIの内部で使用されているバス・システムというものがどういうものかを知らない方が，なるべく具体的なイメージを簡単につかむことができることを目標に記述しています．

学習する項目は，次のようなものとなります．

(1) AMBAバス仕様（AHB，APB）
(2) Grlibの中でのAMBA関連部分の実装（RTL）

LEONプロセッサとは関係なく，一般的な知識として使えるので，若手エンジニアや学生にとっては様々な場面で役に立つと思います．

本章は第1部や第2部とは異なり，きれいにまとまった論理を記述するのではなく，部分的なイメージの説明を繰り返し，実際にソース・コードを見て動かすことを繰り返した結果，いつの間にか全体像を把握しているという形を目指しています．

```
git checkout -b my_amba origin/work_amba
```

で，本章の設計データのあるブランチに切り替わります．

バスとは何か

8.1 プロセッサと各種コントローラの接続例

●プロセッサとメモリ

図8.1に，LEONシステムのバス構造を示します．LEONシステムは，AMBAバスのうち，AHBとAPBバスを使用しています．

最初に，この構造のバスがどのように動作しているかを理解することによって，バスがどういうものかを理解します．

第3部

第9章 LEON システムのGRLIBの主なIPコアの詳細

LEON プロセッサ，Ethernet コントローラ，AMBA プラグ&プレイ，SDRAM コントローラの仕様を理解しよう

LEONシステムには，多数のAMBA AHBおよびAPB準拠のオープンソースIPコアがあります．また，詳細な英語の仕様書も添付されており，GPLライセンスに従う限り再利用が可能です．

本章では，本書で使用した主要なIPコアを，容易に使うことができるように，仕様書の中から必要と思われる最低限の部分を説明します．

9.1 LEON3 プロセッサ

●プロセッサの概要

LEON3は，32ビット，SPARC V8アーキテクチャに準拠したプロセッサで，次のような特徴を持っています．

• Integer Unit

LEON3のInteger Unitは，SPARC V8完全準拠で，ハードウェア乗算，除算命令を含みます．レジスタ・ウィンドウの数は2～32の間で，コンフィグレーション可能です（デフォルトは8）．7段のパイプラインで構成され，命令キャッシュとデータ・キャッシュが分離されたインターフェースを持ちます（ハーバード・アーキテクチャ）．

•キャッシュ

LEON3は，非常に細かくコンフィグレーションできる命令キャッシュとデータ・キャッシュを持ちます．それぞれのキャッシュは，1～4セット，1～256Kバイト/セット，16または32バイト/ラインでコンフィグレーションできます．

データ・キャッシュはライト・スルーで，ダブル・ワードのライト・バッファを持ちます．データ・キャッシュは，AHBバスのバス・スヌーピングを行うことができます．

•浮動小数点ユニットとコプロセッサ

LEON3のInteger Unitは，浮動小数点ユニット（FPU）と，カスタムのコプロセッサのインターフェースを持ちます．また，二つのFPUコントローラがあります．一つはGRFPU，もう一つはMeiko FPU用です．

浮動小数点ユニットとコプロセッサは，Integer Unitと並列動作可能で，データ・リソースの依存がない限りオペレーションをブロックしません．

• Memory Management Unit（MMU）

SPARC V8 Reference Memory Management Unit（SRMMU）をオプションとして有効にすることができます．SRMMUは，SPARC V8 MMU仕様を完全に満たしています．32ビットの仮想アドレス・スペースと，36ビットの物理メモリ間のマッピングを行います．3レベルのテーブル・ウォーク・ハードウェアが実装されており，最大64のフル・アソシエイティブTLBエントリをコンフィグレーションできます．

•オンチップ・デバッグ機能をサポート

LEON3パイプラインは，デバッグをサポートする機能を持っています．ソフトウェア・デバッグのために，最大四つのウォッチポイント・レジスタを有効にできます．

それぞれのウォッチポイント・レジスタは，ブレーク・ポイント割り込みを発生させます．オプションのデバッグ・サポート・ユニットをつけた場合，ウォッチポイントはデバッグ・モードへ入るために使用できます．デバッグ・サポート・ユニットのインターフェースを通して，すべてのプロセッサ・レジスタやキャッシュにアクセスできます．デバッグ・インターフェースは，シングル・ステップ実行やハードウェア・ブレーク・ポイントやウォッチポイントのコントロールを可能にします．

また，内部のトレース・バッファにより，実行された命令を後で読み出すことが可能です．

•割り込みインターフェース

LEON3は，15個の割り込みを持つ，SPARC V8 Interrupt Modelをサポートします．

●Integer Unit

Integer Unitは，次のような特徴を持っています．7段パイプライン，分離された命令キャッシュとデータ・キャッシュのインターフェース，2～32のレジスタ・ウィンドウ，ハードウェア乗算器（オプションで

CQ出版社

見本

Design Wave

FPGAキットで始める ハード&ソフト丸ごと設計

CPUと周辺回路を作り込んでCプログラミング