動体検出処理を例に

第9章 ハードウェア化のためには物理アドレスを意識することが重要

大量データ転送処理の最適化

小山 忠昭 Tadaaki Koyama

画像処理のように大量のデータを転送する必要がある場合，メモリの物理アドレスの意識が重要になります．Linuxで動作するソフトウェアでは通常意識することがなく，そのままハードウェア化しても，期待通りの性能を得られない場合があります．ここでは，大量データ転送を高速化する際のコツを，Zynqボードで実際に動作させながら確認します．評価には動体検出処理を用います．

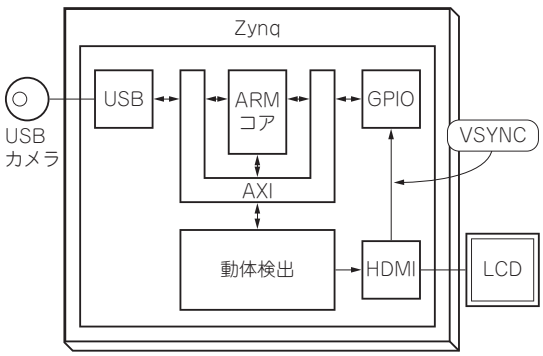

図1 実験環境…動体検出システム

実験の構成

実験で用いたシステムを**図1**に示します．

● **仕組み**

動体検出はカメラで撮った2枚の画像を，輝度を比較し，差のあるところをしきい値をとって，そこだけ色変換して画面に出すという，簡単なものです．これを，Cプログラムから SDSoCだけでハードウェアに実装します．

● **画像入力にはUSBカメラを活用**

画像入力にはUSBカメラを使い，Linux上から制御します．つなぐだけで使えます．

● **HDMI出力コアを利用**

HDMI出力用のIPコアをSDSoC向けにプラットフォーム化しました．このコアは，小型Zynqボード Kiss4とともに提供されます．

表示する画像は，AXIストリームを使って入力します．VSYNCに合わせてデータをソフトウェア側から送ってもえるように，1本だけVSYNCの信号をGPIOに接続しています．

AXIストリームはダイレクトI/O仕様としてライブラリ化しています．SDSoCで指定の関数（今回は`pf_wrete_stream`）をアクセスすれば，データが送り込まれて，HDMIから画像信号が出る仕組みです．

VSYNCは，ポーリングでも，割り込みでも使えるように，GPIOポートに接続します．

USBカメラは640×480画素のYUV出力のため，HDMI出力も640×480画素にしました．YUV→RGB変換も必要です．

リスト1 データ転送関数

```
void trans(unsigned int *data){
     #pragma SDS data copy(data[0:BUF_SIZE])
     pf_write_stream(data);

}
```

実験1：メモリ上の画像をHDMI出力

最初に，メモリからHDMI出力コアに画像データを転送して，画像表示を行います．

● **仕様…データをそろえて関数を呼び出すだけ**

VSYNCに合わせて，画面サイズに必要なデータをそろえて出力すれば画面が出るようにしています．

プラットフォームとして用意した関数`pf_write_stream`を**リスト1**のように呼び出します．あらかじめ，`data`に画像データを入れておいて，VSYNCに合わせて，この関数を実行すれば，画像が出力されます．

● **純粋なソフトウェア記述では正しく表示できない**

リスト1の記述で出力された画像を**写真1**に示します．残念ながら，きれいに表示されませんでした．

VSYNCに合わせて適切にデータが転送されていないことが原因でした．HDMI出力コアは，VSYNCを出してから，4万CPUクロック以内にデータを受け取