愛知工業大学研究報告
第 43 号　平成 20 年

FPGA による ITU-656 ビデオカメラのインターフェイスおよび画像演算器の設計

A Design of an Interface and Image Processing Circuit with FPGA for ITU-656 Video Camera

加藤　正義[†]，堀田　厚生[††]
Masayoshi KATO[†], Atsuo HOTTA[††]

Abstract　High-speed processing is necessary to perform image processing in real time. Hardware processing with FPGA is one of the ways to process images fast.　A system processing images from video camera in real time has been designed. The system has the following blocks.　1) An interface circuit getting images from camera, de-inter-race and transfer them to SDRAM.　2) An image Processor. Two images read from the SDRAM are processed by subtraction and segmentation to 2 values. Then the resulted image is stored into the SDRAM.

The processing speed with the FPGA was enough to perform real-time processing.

１．はじめに

１・１ 研究の背景及び目的

画像情報からは視覚的な映像だけでなく、画像を解析することによって速度、温度、圧力などの情報を得ることが出来る。これらの情報を高精度で利便性も高く得るために求められる性能には、高画質、高フレームレート、リアルタイム性などが挙げられる。しかしながら、高画質、高フレームレートな画像及び動画データは単位時間あたりのデータ量が膨大となるため、処理を高速で行わなければならない。

画像処理を高速で行う方法としてハードウェア演算が挙げられる。本研究室ではこれまで FPGA による設計を行ってきた。同研究室 2004 年卒の杉野は MIPS アーキテクチャを用いた CPU を FPGA で設計[1)]し、同研究室 2005 年卒の森川はプログラムを SDRAM に格納する CPU の設計[2)]を行った。またソフトウェアによる画像処理例として同研究室 2005 年卒の佐久間のソフトウェアによる画像処理[3)]がある。近年、画像処理分野でも FPGA の利用が注目されている。同研究室 2007 年卒の山部は FPGA を用いた画像処理演算器の設計[4)]を行った。

本研究ではカメラの出力画像から目的の画像を抽出することに注目し、専用ハードウェアを設計することを目標としている。応用例の一つとしては、道路などに設置されている速度検知システム(オービス)の代用となるシステムが挙げられる。市販のカメラを使用して抽出から速度測定を行うことで安価なオービス代用システムが可能になると考えている。本研究ではその足がかりとして、FPGA を用いてカメラから得た画像の処理を行えるインターフェイス、コントローラなどのシステム構成の設計を行った。そして画像処理の一例として差分器、2 値化処理器を組み込み、動作検証を行った。

１・２　FPGA

FPGA は、「Field Programmable Gate Array」の略であり、1985 年にザイリンクス社により生み出された書き換え可能な SRAM（Static Random Access Memory）ベースの LSI である。新しいコンピュータアーキテクチャのアイデアを実現する際に、試作機として ASIC を開発するか膨大な数の個別 IC をブレッドボードに実装するしかない。しかし膨大なコストと労力を必要とするこれらの作業と違い、一度に複数の FPGA を実装した試作用ボードを作っておけば、設計した新しいアーキテクチャを即座に実行できるようになる。さらに修正・仕様変更も容易にできるようになった。これにより、多くの新しいアーキテクチャが登場するとともにリコンフィギュラブル（再構成可能）プロセッサの研究や新しい FPGA アーキテクチャの研究が盛んになった。その後、通信・画像処

† 愛知工業大学大学院 工学研究科（豊田市）
†† 愛知工業大学 工学部 電気工学科（豊田市）

理分野でもその特徴が大きく評価され、ルータなど通信ネットワーク網を構成する各装置内に多く採用されて行った。また液晶テレビやステレオなどにも搭載されてきており、今後さらに我々の身近で注目を集めていく LSI と言える。

1･3 設計手法及び使用デバイス

設計、動作及びデータの確認は Windows XP Service Pack 2 を OS としたパソコン上で行った。FPGA 設計ソフトは ALTERA 社の「Quartus II　Version 7.2」を使用し、開発言語として Verilog-HDL を用い、カメラ画像入力コントローラ、I/P (Interlace/Progressive)変換コントローラ、画像演算器を設計した。Windows 用のデバイスドライバの設計には Jungo 社の「WinDriver　Version8.1」を、Windows からのアクセスのプログラム作成には Microsoft 社の「Visual　C++ Version 6.0」を使用した。

使用デバイスとして CQ 出版社 Stratix 評価キットを用い研究を行った。この評価キットボード上には ALTERA 社 FPGA Stratix EP1S10F780C7ES、および Micron 社の SDRAM MT48LC2M32B2 が実装されている。Stratix EP1S10F780C7ES は LE 10570 個、RAM 920,448bit、DSP ６個、PLL６個などをサポートしており、MT48LC2M32B2 は 64Mbit の記憶容量を持つ。

カメラには１/４インチ２６万画素 CCD 超小型カラーカメラ MTV-54K0DN を使用した。本カメラのディジタル出力には標準規格である ITU-656　8bit ディジタル出力が適用されている。

２．　開発方法と構成

2･1 設計構成

カメラから得た画像データに対して演算処理を行うためには、画像データを論理回路やコンピュータが扱いやすい形式に変換することが必要となる。したがって、カメラ画像入力コントローラおよび I/P 変換コントローラを設計した。

また、画像の変換後データの差分および２値化を行う画像演算器を設計した。

本来ならばカメラから得た画像を直接演算し、結果を転送するシステムが求められる。しかし処理ブロックごとのデバッグが困難となるため、処理別にブロック分けをし、設計、動作確認を行った。

2･2 データの流れ

カメラから入力された画像データは以下の手順で SDRAM に格納される。流れを図１に示す。

カメラからは常に FPGA の入力ピンにデータが入力されている。カメラ画像入力コントローラが画像データの先頭を探し、データのサンプリングを開始する。取り出し範囲の輝度成分のみを抽出し、FIFO にデータを転送する。I/P 変換コントローラが FIFO からの EMPTY 信号をチェックし、データが存在すれば読み込む。インターレースを解除できるように書込み先アドレスを決定し、SDRAM コントローラへと書込み要求を出す。

これを繰り返し、必要分のデータを格納する。

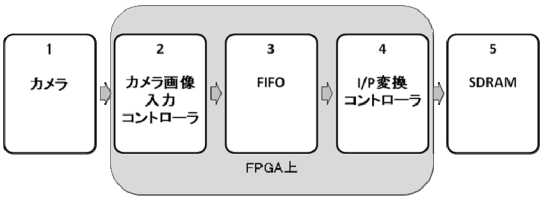

図１　カメラから SDRAM までのデータの流れ

次に、SDRAM に格納されているデータを演算するための手順である。画像 2 枚分のデータを SDRAM に格納した状態で、外部信号により画像演算器が始動する。データの流れを図２に示す。SDRAM から読み込まれたデータは演算器にて差分演算、２値化演算が行われ、再び SDRAM へと演算結果が書き込まれる。

演算器の動作確認のため、演算を行う画像データの SDRAM への書込み、演算結果の SDRAM からの読み込み、データの確認にはパソコンを利用した。FPGA ボードとパソコンとの接続には PCI バスを使用し、PCI バスコントローラにアクセスすることにより行う。

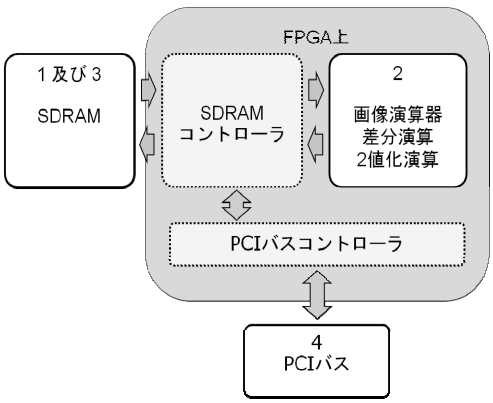

図２　SDRAM 内の画像を演算する流れ

３．　設計回路

3･1　カメラ画像入力コントローラ

カメラからのデータを FIFO に渡すまでのデータの流れを制御するコントローラである。図３の概要図のように、このコントローラは大きく分けて３つのブロックに分けられている。

① get_image タイミングコードとデータを取得し、指定画素範囲を取り出す。
② abstract_y 輝度成分のみを抽出する。
③ convert_width バス幅を 8bit から 32bit に変換する。

上記３つの処理を行い、32bitFIFO にデータを格納する。データ幅はメモリバスとバス幅を合わせるために 8bit 幅のカメラデータを 32bit 幅に調整した。この３つのブロックはすべて、カメラからの 27MHz の基準クロックに同期して動く。

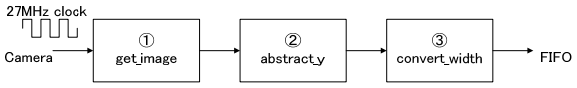

図３ カメラ入力コントローラ ブロック図

① get_image ブロック

以下の手順でタイミングコードを判別し、指定画素範囲を取り出す。

1. カメラから出力されたデータ 4 クロック分、4byte のデータをバッファリングする。
2. バッファリングしたデータをタイミングコードと比較する。（図４）
3. 順次送られてくるデータを比較し続け、EAV を探して発見したらフラグを立て、続けて SAV を探す。
4. 取り出す画素範囲(幅 640pixel)のみを取得するためにオフセットとして指定画素範囲の先頭の手前までを読み飛ばす。（図５）
5. 指定画素範囲の先頭からデータを取得していく。指定画素範囲の先頭データと同時に head 信号を出し、1 ラインの先頭であることを伝える。
6. 出力している信号が輝度信号の場合には同じタイミングで輝度信号であることを表す信号(y_element)を High に上げる。
7. このときカウンタを動作させ、何画素目を取得しているのかをカウントする。
8. 640 ピクセルのデータを取得し終えたら、取得を止め、次の EAV まで待機する。

以上の動作を繰り返す。

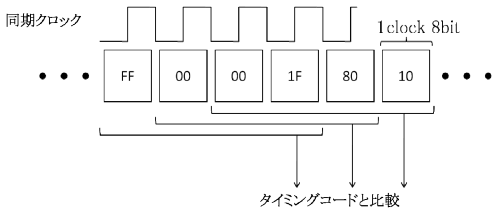

図４ タイミングコードとの比較

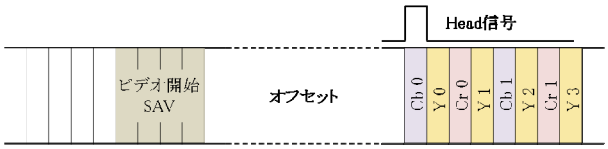

図５ オフセット

② abstract_y ブロック

get_image で出力されるタイミング信号を元に輝度成分だけ抜き出す。get_image により輝度情報を出力している事を示すための信号 y_element が high のときのみ、convert_width ブロックにデータ書き込み要求を出す。

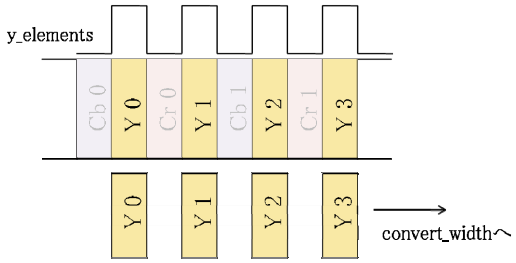

図６ 輝度成分の抜き出し

③ convert_width ブロック

abstract_y から出力される 8bit 幅のデータを 32bit 幅に変更する。入ってきたデータの数を数えるデータカウンタを設けている。

1. 取得する画像の先頭を確認したらデータカウンタを 0 にする。
2. データが入力される毎にデータカウンタの数を１つ加算する。
3. 8bit のデータが４つ溜まった時点で 32bit のデータとして FIFO に書き込み要求を出す。

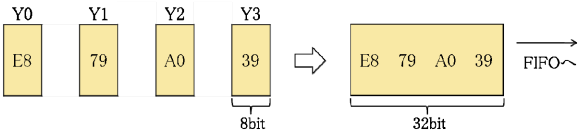

図７ 8bit/32bit 変換

3･2　I/P 変換コントローラ

カメラ画像入力コントローラから FIFO に取り込まれたデータが貯まると、プログレッシブ化コントローラに取り込まれる。取り込まれたデータはカメラから指定範囲の画像を記録しただけであるため、インターレース形式の画像である。よって、I/P 変換を行い、インターレース画像であるカメラ画像データをプログレッシブ化（インターレースでない画像に変換）する。図８の通りデータを格納するメモリアドレスの生成を画像 1 列ごとに適したアドレスにすれば、プログレッシブ化される。

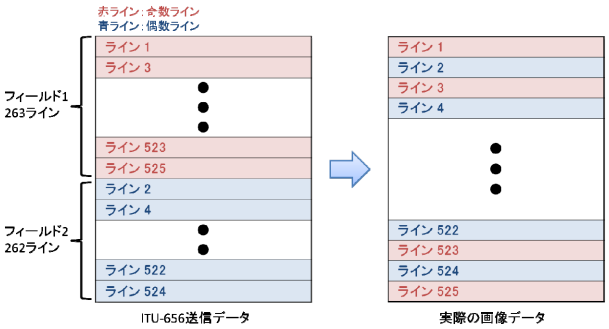

図８　インターレース・トップファーストオーダ方式

トップファーストのインターレースでは奇数列のデータが先に、偶数列のデータがあとに送られてくる。変換するためには以下の手順でアドレスを決定していく。

1. データの書込みは先頭から順に行われる。
2. 現在が何ピクセル目なのかをカウントして 1 列の終わりを判断する。
3. 1 列書き終わったら 1 列分のアドレスを飛ばす。このとき、何列目なのかをカウントする。
4. 奇数列数がすべて書き終わったら、メモリアドレスを 2 列目の開始、つまり 1 列目最後のピクセルの次まで戻す。
5. 以後、同様に偶数列を 1 列飛びにすべて書きこむ。

メモリへのデータ転送は 32bit、FIFO からのデータも 32bit であるため、輝度データ(Y)は 4 ピクセル分(8bit × 4 = 32bit = 4Byte)を 1 単位として書き込まれる。

カウンタはメインとなるステート内の記述によりカウントアップが行われる。続けて、カウンタの数値に従ってメモリへの書込みアドレスが生成される。アドレス生成部もカウンタと同じく基準クロックと同期して動作し、ステート内の記述によって開始と停止が行われる。

3･3 画像演算器

SDRAM に二枚の画像分のデータを書き込んだ後に外部からの信号により画像演算器が始動する。図９にメモリへデータの格納の概要を示す。

① 画像演算器の動作は画像一枚目のはじめのアドレスにアクセスし 32bit データを格納する。
② 同様に二枚目の画像のアドレスにアクセスする。それぞれレジスタに格納されたデータの絶対値の差分を取って別のレジスタに格納する。その後、２値化を行う。
③ 計算を終えると再び SDRAM コントローラに演算結果を書き込む。なお画像演算器にカウンタを設けて自動的にアドレスを生成するように設計した。アドレス計算の方法を以下に示す。

本来ならば２値化をすることによりデータサイズが小さくなるが、1 バイトが画像 1 ピクセルの関係を保つ事によりパソコン上での処理がしやくすなるため、0x00 と 0xFF の 2 つの値へと変換した。本研究では対象物体と背景、もしくは移動中の動的対象物を抽出目的の対象としているため、多くの画素である抽出対象物以外の部分では差分は 0 に近い。よって、2 値化の閾値は比較的小さな固定値(0x3F)とした。

格納先メモリアドレスは以下の式で求められる。

640×480 pixel サイズの画像の場合

640×480×8(1 pixel のデータサイズ)÷32(アドレスデータ幅)＝$13EC0_H$

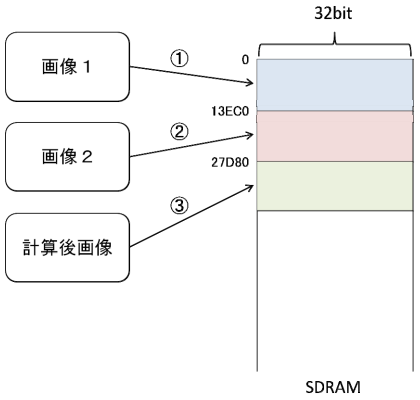

図９　画像演算器メモリマップ

４．結果及び動作検証

４・１ 論理合成結果

Quartus II で設計したカメラ画像入力コントローラ、I/P 変換コントローラ、画像演算器、及び使用した FIFO、入力部全体と演算部全体(SDRAM コントローラと PCI コントローラを追加)の論理合成結果を表２に示す。対象デバイスは ALTERA FPGA Stratix EP1S10F780C7ES である。

FIFO は Quartus II に IP 機能として備わっている MegaWizard Plug-In Manager を用いて生成したものを使用した。Bit 数は 32bit、サイズは 1024Words である。

表２　論路合成結果

	Logic Cells	LC Registers	Max frequency
カメラ画像入力コントローラ	207	180	N/A
FIFO (32bit 1024Words)	281	269	N/A
I/P 変換コントローラ	222	131	N/A
カメラ入力変換部全体	702	571	150.74MHz
画像演算器	616	401	153.82MHz

４・２ 動作検証

４・２・１ カメラ入力側デバイス

波形シミュレーション、及びロジックアナライザでカメラ入力側デバイスの動作検証を行った。まず、シミュレーション波形で設計した通りの動作が行われているか確認した。EAV、SAV を確認後、オフセットを飛ばして書込要求、書込先アドレス、書込データが正常に出力されているのが図１０から確認できる。また、ラインが移ったときにアドレスがジャンプしているため、インターレースも解除されることを確認した。よってカメラ画像入力コントローラ、I/P 変換コントローラは正しく設計されていると言える。

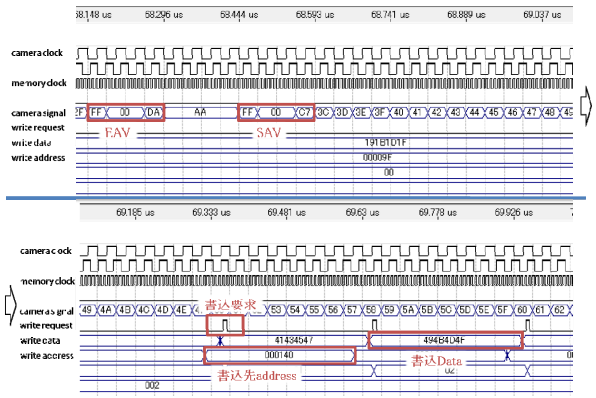

図１０　　カメラ入力-メモリ書き込み信号図

その後、カメラからの信号を FPGA に入力し、メモリへの書き込み信号をロジックアナライザで確認した。ロジックアナライザは FPGA 上に構築され、SRAM に実際の波形の状態が記録される。よって、シミュレーション結果と同様の結果が得られれば、回路は正しく動作していると考えられる。FPGA にはカメラ MTV-54K0DN からのデータを実際に入力した。

結果、シミュレーションと同じく、SAV が確認された後、書込要求、書込先アドレス、書込データが出力されたのが図１１から確認できる。これにより、回路が正常に動作していると判断できる。

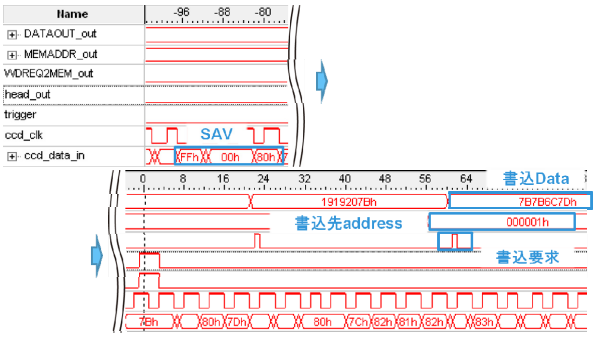

図１１　ロジックアナライザ信号図

４・２・２ 画像演算器

二枚の画像データ(図１２、図１３)を SDRAM へ送り込み、FPGA 内の演算器で差分、2 値化を行った。出力画像を(図１４)に示す。

図１２　対象物画像

図１３　背景画像

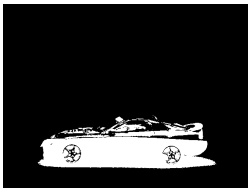

図１４　差分 2 値化画像

出力結果の確認のため、ソフトウェアで同様の演算をしたファイルとハードウェア演算のデータの比較確認をした。同一のデータが出力されたため、演算は正しく行われていることが確認できる。

４・３　画像演算器の演算速度

ハードウェアでの演算速度を計測した。外部からの信号が入力され、画像演算器が起動してからカウンタを起動する。画像演算器による演算がすべて終了した時点で

カウンタを停止させ、結果を 7 セグメント LED に出力した。

表３　演算速度

横 [pixel]	縦 [pixel]	画素数 [pixel]	演算時間 [msec]	1 秒間での処理枚数[枚]
160	120	19200	2.1	476
240	180	43200	4.6	217
320	240	76800	8.2	122
480	360	172800	18	55.5
640	480	307200	33	30.3

※8bit、グレースケールの画像データを使用

結果、640x480、8bit グレースケールで演算時間は 33ms となった。ロス無く連続処理ができるならば約 30 回/秒の差分 2 値化演算が可能となるので、カメラ画像に対するリアルタイム処理も可能である。比較対象として組み込み用途ではない CPU ではあるが、パソコンでのソフトウェア処置と比較した。CPU: Core2Extreme 3GHz、メモリ: Dual CH DDR2-SDRAM 800MHz 2GB のパソコンで 640x480 の同様の画像を同じような計算方法を用いたプログラムで処理をすると、演算速度は 11msec であった。

組み込み用途の CPU の動作速度は 100MHz～400MHz 程度である。仮に組み込み用途の CPU で同様のプログラムを動作させた場合、周波数のみで判断ではあるが、400MHz でも 80msec 以上はかかると推測される。（実際にはバス幅、メモリ速度なども低下するため、それ以上だと考えられる。）

本研究ではまず、動作させることに重点を置いたため、高速化処理はされていない。ハードウェア演算のさらなる高速化をするために、次のような方法が考えられる。

- 本設計ではメモリ 1 アドレス単位(32bit)ずつ個別に演算しているのを、複数アドレス単位をまとめて計算するようにする。
- 演算工程をパイプライン処理にする。
- 無駄な待機ステートを減らす。
- 高速な FPGA デバイスの使用 （ex. Stratix III、Stratix II GX）
- 高速なメモリの使用 （ex. DDR3-SDRAM）

５．　結言

本研究では、FPGA による ITU-656 ディジタルビデオカメラ用のインターフェイスと画像演算器の設計を行った。そのため、以下のことを行い、検証した。

1. カメラ画像入力コントローラと I/P 変換コントローラの設計を行った。
2. 波形シミュレーション、ロジックアナライザでカメラ画像入力コントローラ、I/P 変換コントローラの動作検証を行った。
3. 演算器として差分 2 値化演算器を設計し、2 枚の画像から差分 2 値画像を生成した。
4. 演算時間を測定し、演算速度はカメラからの画像を連続で処理可能な速度であることを確認した。演算時間は 640x480 のグレースケール画像 2 枚を演算した場合、33msec であった。
5. 論理合成結果は、カメラ画像入力コントローラが 207 ロジックセル、180 レジスタ、I/P 変換コントローラが 281 ロジックセル、269 レジスタ、画像演算器が 616 ロジックセル、401 レジスタとなった。

６．　今後の課題

- カメラ入力側のコントローラ、変換器と PCI/SD-RAM を使った演算器を 1 つのシステムとして組み上げる。
- 連続してカメラ画像の演算処理を行い、データを蓄積する。
- 演算速度を向上させる。
- 物体の認識、移動距離の算出などの演算処理を行う。

参考文献

1) 杉野晃洋，堀田厚生：MIPSCPU の FPGA 化，2004.
2) 森川良，杉野晃洋，堀田厚生：SDRAM をメインメモリとする MIPSCPU の FPGA 化，2005.
3) 佐久間湖，堀田厚生：動画像からの移動物体抽出と速度の推定，2005.
4) 山部選，堀田厚生：FPGA による画像処理演算器の設計，2006.
5) トランジスタ技術 2005 年 2 月号, CQ 出版社, 2006.
6) 木村　真也：トランジスタ技術 SPECIAL 2006 SUMMER “わかる VerilogHDL 入門”，CQ 出版社, 2006.

（受理　平成20年３月１９日）