愛知工業大学研究報告
第 42 号 B 平成 19 年

FPGA による画像処理演算器の設計

Design of image processing operation machine by FPGA

山部　選†，堀田　厚生††
Suguru YAMABE†, Atsuo HOTTA††

Abstract An image processing system with a FPGA has been developed. the system has the following functions.1) BMP images taken with a digital camera and stored in a PC are transferred to a SDRAM on a board including a FPGA through a PCI bus.2) Two images are read from the SDRAM and are processed by background subtraction method in the FPGA, and the resulted image is stored into the SDRAM.3) The result image are transferred to the PC via a PCI bus, and displayed.

Processing time of background subtraction with the FPGA and with a software in a PC has been compared, and found that the former is several times faster then the latter.

1. はじめに

1・1　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・2　研究の背景及び目的

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。また近年、画像処理分野でも FPGA の利用が注目されている。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 年院生卒の杉野は MIPS アーキテクチャを用いた CPU を FPGA で設計[1]し、また同研究室 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。オービスシステムには専用ハードウェアによる画像の抽出が必要になる。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]があげられる。本研究では画像の抽出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]が挙げられる。これを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の第一歩として FPGA を用いて画像処理の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を組み込み、動作検証を行った。

1・3　本研究の流れ

FPGA での画像処理を行うためには、最初に画像を格納する場所が必要になる。FPGA の外部メモリとして画像を格納する十分なメモリ領域を持ちなおかつ高速に動作する理由から SDRAM を使用する。(SDRAM コントロ

†　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）

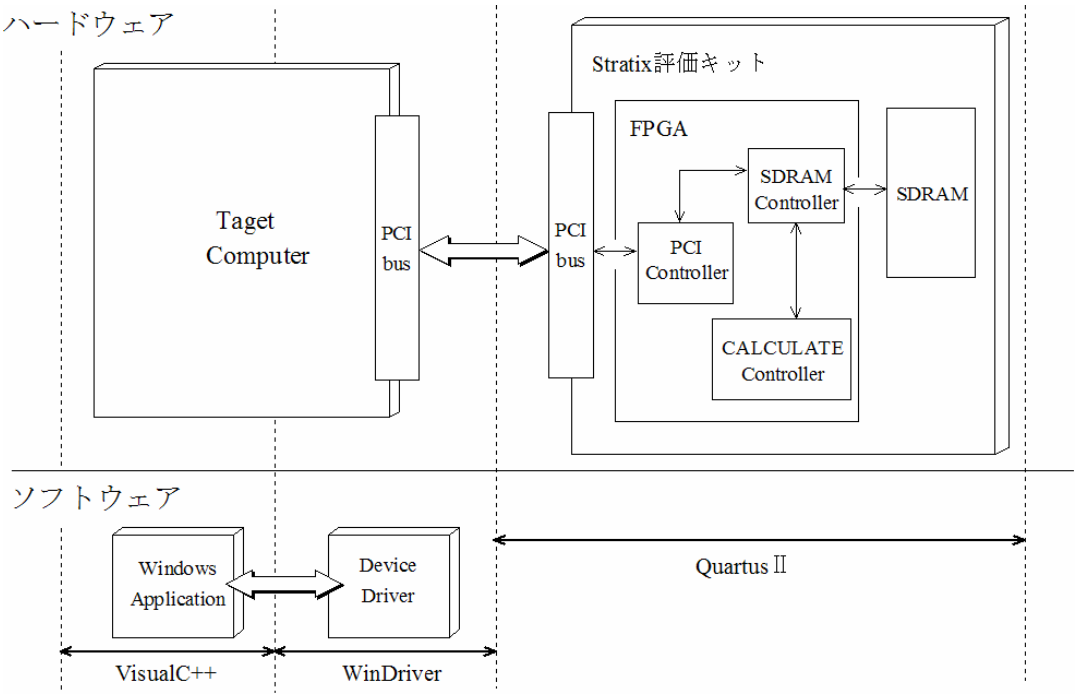

図 1　設計構成図

―ラ設計) 第二段階として画像を入出力するためのインターフェイスが必要になる。今回は高速動作の点から PCI バスを採用する。（PCI コントローラ設計）　第三段階はドライバと入出力のためのアプリケーションでは汎用性の面から Windows を採用する。本研究では動作の確認として BMP 画像の差分器（演算コントローラ設計）を例にとり FPGA による画像処理が動作しているかどうかを検証した。設計構成図を図 1 に示す。ハードウェア設計ソフトに Altera 社の Quartus Ⅱ、ドライバ開発には Xlsoft 社の WinDriver、アプリケーション開発には Microsoft 社の Visual C++を使用する。

２・１　SDRAM コントローラ

２・２・１　RAM

RAM(Random Access Memory)は SRAM(Static RAM)と DRAM(Dynamic RAM)に分けられデータの記憶方法に違いがある。SRAM はトランジスタによる順序回路(FF)で構成され論理値レベルでデータが記憶される。一方、DRAM はトランジスター個キャパシター個で構成され、キャパシタに電荷を蓄えるか否かで“1”“0”を記憶する。DRAM は SRAM と比べ構造が簡単であるため大容量化、コストが低い利点がある。しかし、情報記憶用コンデンサに蓄えられた電荷は放っておくとリーク電流として漏れでてしまい、一定時間後には電荷がなくなってしまう。それを防ぐために電荷を再補充してデータの消失を防ぐリフレッシュが必要となる。SDRAM とは外部バスインターフェースが一定周期のクロック信号に同期して動作するように改良された DRAM を表す。

２・２・２　SDRAM コントローラ概要

表 1 に設計仕様を示す。CAS　Latency はリードコマンドが挿入されて、データが排出されるまでの遅れ時間を表す。なおリフレッシュは FPGA 内部にカウンターを設けて 64ms 以内でリフレッシュコマンドを実行する様に設計した。

表 1　SDRAM コントローラ設計仕様

動作周波数	100MHz
(CAS Latency)	3
データ長	32bit
アクセス(read,write)	Single
Refresh	Self
Refresh Time	64ms
Prechage	Auto

図 2 に SDRAM コントローラ状態遷移図を示す。SDRAM は電源投入直後は論理状態が不明であり、正常な動作を保障するためには初期化を行う必要がある。今回の SDRAM では 100μs のポーズ期間をあけ、プリチャージを行い、リフレッシュを二回繰り返し行う。その後

モードレジスタの設定を行う。主に CAS Latency、アクセス長などを設定する。その後 IDLE ステートに移り、読み書きのコマンド受け付けが可能となる。コマンドとしては PCI、SDRAM 間の要求、演算コントローラ、SDRAM 間の要求、リフレッシュ要求の５つのコマンドがある。リード、ライト要求を受けると、ACTIVE コマンドに移行しバンク、行アドレスを出力する。なお NOP とは必要時間をかせぐための何もしない状態を表す。WRITE、READ ステートでは列アドレスを出力し書き込み読み込みが完了するまでまち GET、SUBGET でデータを受け取る。リフレッシュはプリチャージを行った後リフレッシュコマンドを二回繰り返す動作になる。

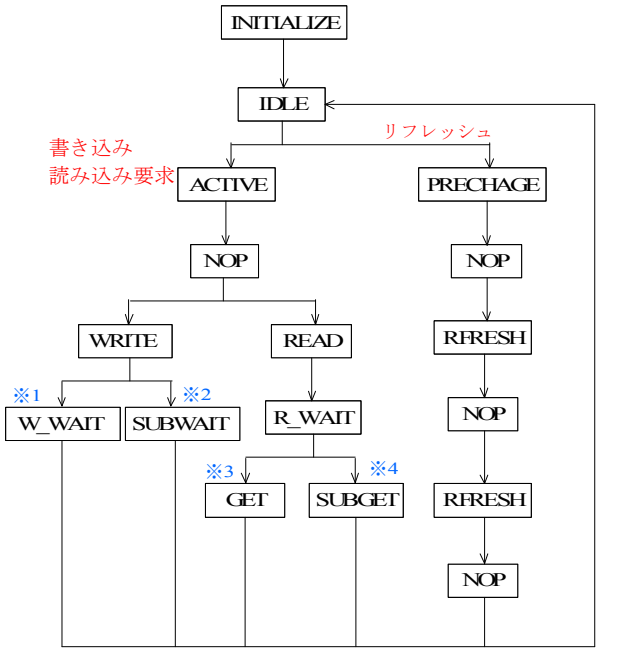

図２　SDRAM コントローラ状態遷移図

２・２　PCI コントローラ

２・２・１　PCI 設計規格

PCI の特徴としてはパラレル転送方式で 32bit のデータ幅を持つ、動作周波数が 33MH z で 132MB/S のデータ転送が可能、CPU に直結したバス PCI の間にはバスブリッジ回路が存在し、PCI バスが特定のシステムアーキテクチャを前提とした拡張バスの仕様となっていないため CPU の依存性が低いなどの特徴がある。PCI コントローラは図３に示すようターゲットシーケンサとローカルバスシーケンサの二つのシーケンサを持っている。おおまかな動作としてはデバイスの後で説明するコンフィグレーション空間に格納されているデバイス情報と比較し自分が選択されているかを確認することを行う。実際のデータ受け渡しなどはローカルバスシーケンサで行われることになる。

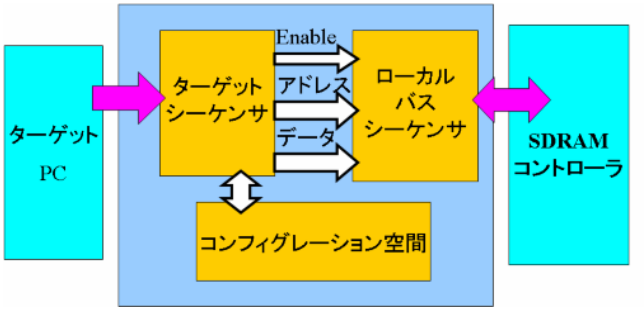

図３　PCI コントローラ全体図

２・２・２　ターゲットシーケンサ

状態遷移図を図４に示す。BUS_IDLE で待機し要求を受けると ADRS_COMPARE でコンフィグレーション空間（２・２・４）に格納されているデバイス情報と比較し自分が選択されているかを確認する。WAIT_IRDY でローカルバスシーケンサ（２・２・３）が動作し SDRAM

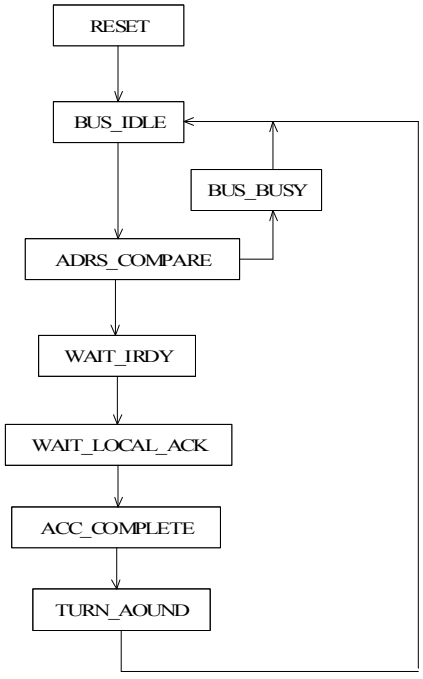

図４　ターゲットシーケンサ状態遷移図

と実際のデータの受け渡しをする。受け渡しの完了シグナルを受け取ると ACC_COMPLETE に移行し TURN_AROUND でドライブの切り離しを行う。

２・２・３　ローカルバスシーケンサ

状態遷移図を図５に示す。ターゲットシーケンサから動作要求が来るまでは LOCAL_IDLE で待機し、要求を受けるとメモリアクセス要求、IO アクセス要求、コンフィグレーションアクセス要求にそれぞれ移行する。相手方のデバイスに応じたタイミング調整を行い受け渡しが完了すると STATE_COMP 移りターゲットシーケンサに完了シグナルを送る。

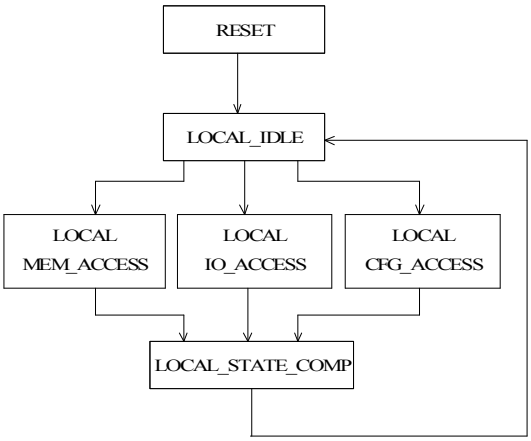

図５　ローカルバスシーケンサ状態遷移図

２・２・４　コンフィグレーション空間

PCI バスは、デバイス 1 つあたり最大 256 バイトのコンフィグレーション空間を持ち、PCI のデバイス ID、ベンダーID、メモリ空間の占有容量、割り込み情報などを持つレジスタなどを実装する。この空間に実装されているレジスタのほとんどが、BIOS や OS の起動時にメモリ・アドレスや割り込みが衝突しないように設定を行うプラグ＆プレイ・システムのために利用される。

２・３　演算コントローラ

PCI から 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演算コントローラが始動する。SIDLE で要求を待ち、READ1FIRST で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。READ2FIRST でも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 SUMWRITE で SDRAM コントローラに演算結果を書き込む。画像すべてのデータを処理するまでは CONTINUE に移行し上記の動作を繰り返す。なお CALCULATE コントローラにカウンターを設けて自動的にアドレスを生成するように設計した。

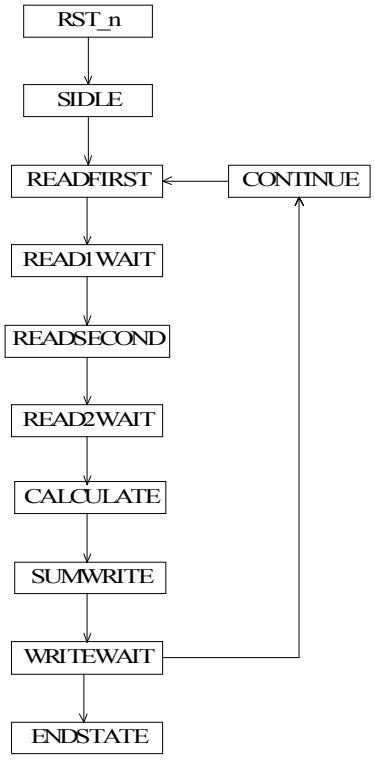

図６　演算コントローラ状態遷移図

４　WinDriver

４・１　WinDriver 概要

保護されたオペレーティングシステムでは通常開発が行われるアプリケーションから直接ハードウェアにアクセスできない。ハードウェアへのアクセスはオペレーティングが「デバイスドライバ」と呼ばれるソフトウェアモジュールを使ってアクセスする必要がある。
デバイスドライバの開発が可能だが開発に数ヶ月という膨大な時間を費やすことになってしまう。そこで WinDriver を使用することによりは短期間でデバイスドライバを作成することができる。

WinDriver のアーキテクチャを図 7 に示す。ハードウェアにアクセスする場合、アプリケーションは WinDriver ユーザーモードライブラリから WinDriver 関数を呼び出す。そしてユーザーモードライブラリがハードウェアにネイティブコールでアクセスする WinDriver カーネルを呼び出しハードウェアとデータのやり取りをする。

WinDriver を使用することにより開発者は『ドライバコード』、必要に応じてパフォーマンス向上のためのカーネルプラグイン『パフォーマンス上重要な関数』のみを記述すれば Windows のドライバとしての認識が完了する。

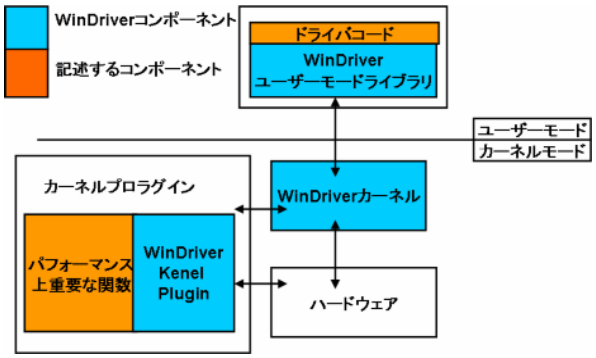

図７　WinDriver アーキテクチャー

４・２　画像処理ソフト

Windows からデバイスドライバにアクセスするソフト開発に『Microsoft　Visual　C++Version6.0』を使用する。取り扱う画像は BMP 画像ファイルである。BMP はヘッダ部分と画像データ部分に分かれる。本設計ではヘッダ部分の画像サイズを計算しループ回数を決定する。その後図――のようにヘッダ部分を切り取り画像データのみを PCI バスに送る。一枚目の画像データを送った後、二枚目の画像データを送る。SDRAM には一枚目の画像データ、二枚目の画像データ、差分結果画像が入る。結果を描画させる時には切り取ったヘッダを付け PC 上に BMP ファイルとして描画する。

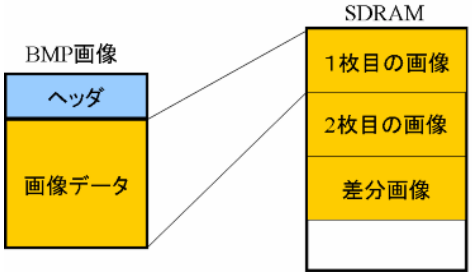

図８　アプリケーション動作

５　結果及び検証

５・１　論理合成結果

Quartus II で設計した SDRAM コントローラ、PCI コントローラ、演算コントローラ及び全体の論理合成結果を以下に示す。対象デバイスは ALTERA FPGA Stratix EP1S10F780C7ES である。表２にはそれぞれ単体のロジックセル数、レジスタ数を示す。表３には全体の論理合成結果を示す。PCI バスは 33MHz と固定のためここで表す最高動作周波数とは SDRAM コントローラ、演算コントローラの最高周波数を指す。

表２　論理合成結果単体

	Logic Cells	LC Register
PCIコントローラ	279	223
SDRAMコントローラ	480	345
演算コントローラ	605	397

表３　論理合成結果全体

最高動作周波数	114.55MHz
Logic Cells	1375/10570(13%)
LC　Register	965
使用PLL	1

５・２　動作検証

図９の背景画像と図１０の物体の入った画像を送り込み FPGA 内の演算器で絶対値の差分を行った。出力結果を図１１に示す。図１２にはソフトウェアで差分をとった結果を示す。図１１と図１２を比較し物体の部分が切り取られ正確に差分演算ができていることが確認できる。

図９　背景画像

図１０　物体画像

図１１　ハード結果

図１２　ソフト結果

次に処理時間の比較を行った。ハードウェアではカウンターを設け速度時間を測定しソフトウェアでも BMP ファイルのヘッダを切り取った時点で GetTimeGet 関数と用いてタイマーを開始させ、画像データのみの差分行った後タイマーを停止させ出力させる。どちらも BGR の差分のみの時間を測定した。なおソフトウェアで時間測定をした実験パソコンスペックは CPU　Pentium4　3.2GHz、メモリ 1GB である。比較結果を表 5-3 に示す。比較するとソフトウェアよりハードウェアの方が処理速度が速い事がわか

る。しかし本来ならばハードウェア処理とソフトウェア処理時間の差はさらに大きく、処理時間差 1.4 倍から 1.9 倍しか違いがないのは非常に遅いことがわかる。原因として考えられることは、ハードウェアの設計で SDRAM から演算コントローラ間をバースト転送にしていないため転送回数が大幅に増大したことが考えられる。

表４　ハードソフト時間比較

	ソフトウェア	ハードウェア
320×240	45ms	24ms
640×480	173ms	98ms
800×600	244ms	165ms
1024×768	352ms	251ms

第六章　結言

本研究では、FPGA による画像演算器の設計を行った。そのために以下の事を行った。

1) SDRAM コントローラ、PCI コントローラ、を設計しデバイスドライバ割り当てを行い Windows からのアクセスを行った。
2) 演算器の例として差分演算器を設計した。
3) 二枚の画像の差分を正確に行い、差分の BMP 画像を生成することができた。
4) 画像演算処理時間の比較を行った。しかし処理速度としてはハードウェアの方が 1.4 倍から 1.9 倍速くなった。バースト転送することでさらなる速度向上が望めると考えられる。

参考文献

［1］　杉野　晃洋、　堀田　厚生“MIPSCPU の FPGA 化”平成 16 年
［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”平成 17 年
［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”平成 17 年
［4］　PCI デバイス設計入門　CQ 出版社
［5］　堀田　厚生“半導体の基礎理論”技術評論社
［6］　浜田　憲一郎“WindowsXP デバイスドライバプログラミング”技術評論社
［7］　小林　優“入門 VerilogHDL 記述”CQ 出版社
［8］　浅田　邦博“ディジタル集積回路の設計と試作”培風館

（受理　平成 19 年 3 月 19 日）