

SDカードへ高速で書き込み可能な画像取得システムのFPGA実装

FUKUDA, Tatsuoki / 福田, 龍宙

(出版者 / Publisher)

法政大学大学院情報科学研究科

(雑誌名 / Journal or Publication Title)

法政大学大学院紀要. 情報科学研究科編 / 法政大学大学院紀要. 情報科学研究科編

(巻 / Volume)

10

(開始ページ / Start Page)

1

(終了ページ / End Page)

6

(発行年 / Year)

2015-03-24

(URL)

<https://doi.org/10.15002/00011702>

SD カードへ高速で書き込み可能な画像取得システムのFPGA実装

FPGA Implementation of Writable Image Acquisition System at High Speed to SD Card

福田 龍宙*
Tatsuoki Fukuda

法政大学大学院情報科学研究科情報科学専攻
E-mail: tatsuoki.fukuda.4w@stu.hosei.ac.jp

Abstract

In recent years, hardware/software co-design has become important. In particular, the development of a composite device of high performance I/O interface by using FPGA become popular. We designed and implemented a hardware system by focusing on two devices of Camera and SD Card. Specifically, we designed a system that can collectively be processed from the acquisition of the image to be taken to store in the data. Moreover, we tried to speed up the data transfer by the control of the SDRAM because it can be controlled without a burden on the transfer rate when there is a change in capacity of the image data. First step is the design of the Camera Interface, LCD Controller, SDRAM Controller, SD Card Controller, VGA Controller, and CPU. Second step is the implementation of the whole system which using the SOPC Builder and the Quartus II development tool. More specifically, the real-time image data is received through the Camera Interface from a camera of tPad. It was controlled by the SD Card Controller and was stored in the SD Card after passing through the Avalon Bus. In addition, the stored in image files are outputted into VGA Display. This paper expresses the integration from the acquisition of the image to the data storage and the speed of the data transfer.

1 まえがき

近年、情報処理の分野でソフトウェアとハードウェアの両方を学ぶ事が必要とされている。ソフトウェアのみで動作するシステムは沢山存在するが、ハードウェアを動作させる為のアプリケーション部分としての役割も果たすので、高速かつ効率的に動作するハードウェアを設計する上で重要な場合がある。ハードウェア設計では論理回路を組み込むのが一般的であるが、その回路をプログラム言語として表現する事が出来るハードウェア記述言語の Verilog HDL があり、設計の負担を解消させることが出来る。ソフトウェア開発で 사용되는のは Java 言語、C 言語、C++言語等があり、画像処理、Web アプリケーション等を目的として表現されている。一方、ハードウェア記述言語の Verilog HDL により設計されるシステムは、ロボット、時計、家電等の制御にあり、高速化や効率化を目的として使用されている。ハードウェアの分野では、この Verilog HDL で組み込みシステムを設計する為に自作の論理回路をシミュレーションさせて高速な処理を可能とさせる FPGA (Field Programmable Gate Array) を使用する事が必要とされている。近年、この FPGA を使用し、各 I/O インターフェースを用いて

複合デバイスによる組み込みシステムの研究開発が盛んになっている。本研究では、カメラと SD メモリーカードの 2 つのデバイスに焦点を置いた。SD メモリーカードはデジタルカメラ等の機器に対応する新世代の小型外部記憶装置としてユーザーから親しまれており、メモリ容量の大きさ、データ転送速度の高速化、動作の柔軟性、そして優れたセキュリティを有する半導体フラッシュメモリがベースとなっている。カメラは現在、主に CCD (Charge Coupled Device) と CMOS (Complementary Metal Oxide Semiconductor) の 2 種類の固定イメージセンサが存在する。近年では、DSP (Digital Signal Processing) をベースとしたリアルタイム画像処理によるシステムが広く使用されており、DSP による処理容量の削減と画像取得システムの速度の向上が要求されている [1]。

本研究では、撮影した画像の取得からデータ転送と保存を一括で処理出来るよう、CMOS イメージセンサによるカメラの撮影から取得した画像を SD カードに自動保存させるシステムの設計と FPGA への実装を行った。さらに、撮影から保存までの画像データ転送の高速化を計る為の手法を検討した上でも実装を行い、SD メモリーカードに保存が完了した画像データを後から読み出して画面で確認することが出来るよう利便性を図る事も目的とした。

2 システムアーキテクチャ

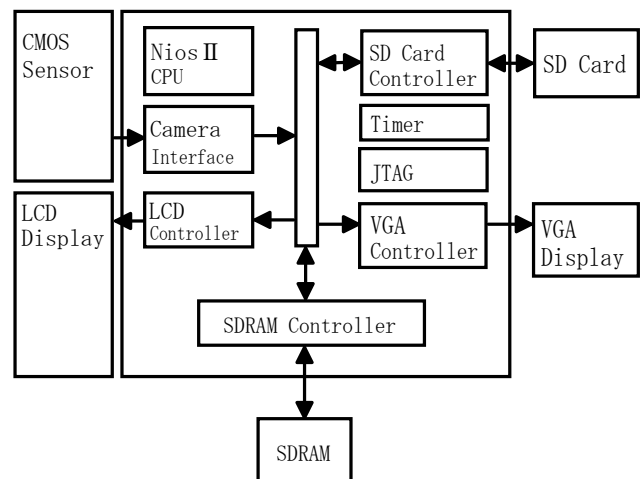


図 1: システム概要図

図 1 は本研究で設計したシステム全体の回路図を表している。システム内にあるモジュールは自動生成も含めてハードウェア記述言語 Verilog HDL を使用して回路設計ツールの Quartus II と SOPC Builder 上で設計と実装を行った。上記のシステムは FPGA に内蔵されているカメラの制御回路の役割を表す Camera Interface から撮影された画像データを SDRAM Controller によって制御され、SDRAM へ一時補間する。そして

*Supervisor: Prof. Yamin Li

て、SD Card Controller で制御する事によって SD メモリーカードにデータを保存させ、LCD Display には撮影するリアルタイムの動画像を表示させる。そして、保存した画像データは VGA Display に表示させて確認が出来る。ここまでの一連の流れを FPGA に実装して転送速度を計測する事を目的とする。

使用した FPGA は Altera 社の DE2-115 であり、SD メモリーカードやカメラのインターフェースも設計できるように一般的な I/O デバイスに対応した環境が整っている。さらに、映像を出力させる LCD は DE2-115 に搭載されている HSMC (High Speed Mezzanine Connector) を経由して裏面にある実験用ディスプレイ tPad から出力される [2]。tPad はカメラ映像に加えてタッチパネルや加速度センサーにも対応している。

その他、Timer はタイマー処理を表し、一定の間隔で信号 (割り込み等) を発生させる事が出来、システムの正常動作を確認する為に使われる。JTAG はシリアル通信 (伝送路を一度に 1bit ずつ逐次的に送る通信) で IC チップの内部回路と通信する仕組みがある。組み込みシステムのソフトウェアのデバッグなどの目的で、CPU や FPGA にアクセスさせる手段に必要な総合的なデバッグのインターフェースである。SDRAM はコンピュータのメインメモリとして使用されるメモリ製品の規格の一つである。従来の DRAM よりも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。そして大容量かつ高速に動作出来るので、SRAM と DRAM にあった課題も解消させることが出来る。最後に Avalon Bus は各モジュール同士を接続する為の内部専用バスである。Avalon インターフェースには、周辺回路とのデータ転送やパケット通信等に使用される Avalon Streaming Interface とメモリの読み書きの為にアクセス等に使用される Avalon Memory Mapped Interface の 2 種類がある。

2.1 Camera インターフェース

現在使用されている固体イメージセンサは CCD と CMOS がある。CCD は高画質で CMOS は低画質であるが、元々性能やノイズ等の面において不利だった CMOS の性能は CCD に近づきつつある。さらに、CMOS センサは高集積化を有する複雑な回路を必要とせず、小型、低消費電力、高い抗放射線能力を持った利点があるので、ロボットビジョン等の様々な画像取得システムに応用されている [3]。ロボットビジョンとはロボットに搭載する事を目的とした画像認識や物体認識等に限定させたコンピュータビジョンの一部を表す。カメラモジュールは DE2-115 の FPGA ボードに搭載されている画像処理装置にリアルタイムで画像データを送信し、24bits で表される RGB 形式にデータを変換して画面に表示させる役割を果たす。

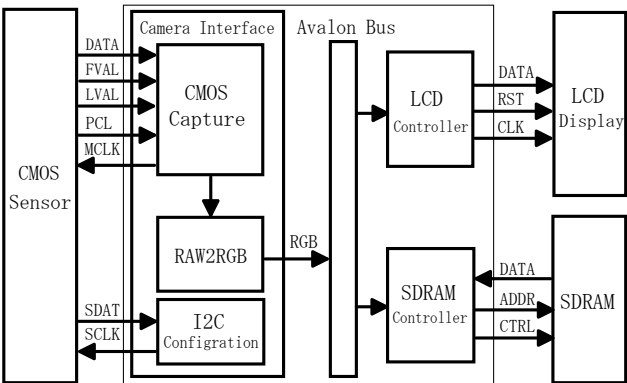


図 2: Camera インターフェースと周辺モジュール

図 2 は Camera インターフェースと周辺モジュールを表す。CMOS Sensor はカメラで撮影する映像を認識してアナログデータ信号に変換する役割があり、Camera インターフェース内で制御する前段階となる。FPGA 上で設計した Camera Interface

内にある回路は CMOS Capture, I2C Configuration, RAW2RGB であり、その他の周辺モジュールは LCD Controller と SDRAM Controller がある。まず、I2C Configuration は CMOS センサを形成させる為にシリアル通信を行うモジュールとして用いられる。本研究でも周辺デバイスの動作を設定させる役割があり、I2C インタフェースを介して CMOS センサを初期化する。

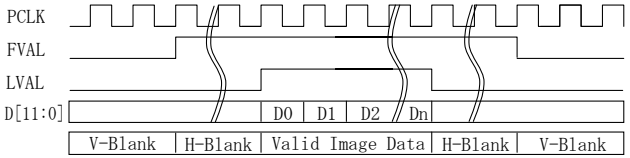


図 3: 画素データ出力タイミング

出力させる画像はフレームとラインに分割される。図 3 は分割された時の画素データの出力タイミングを示している。出力信号の FVAL (Frame Valid) と LVAL (Line Valid) はフレームとラインの境目を示すために使用され、共に有効な場合に 11bit のアナログ画素データを CMOS Capture へ送信する事が出来る。CMOS Capture は画像を 1pixel ずつ 24bit の RGB データに変換させる役割があり、変換による画像データストリームと CMOS センサからの同期信号をベースとした有効なピクセルデータストリームの回路を表している。データストリームはベイヤーパターンのカラーフォーマットから生成される。ベイヤーパターンは 1 つの CMOS カメラまたは CCD カメラとベイヤーパターンフィルターと呼ばれるフィルターを組み合わせでカラー画像の撮影を実現し RGB 形式に変換する事無くデータを出力するカメラである。本研究の場合、RAW2RGB によって RGB データストリームへ変換される [4]。LCD Controller と SDRAM Controller に関しては 2.2, 2.3 で説明する。

2.2 LCD コントローラ

スマートフォン等にあるカメラと同じようにキャプチャをする前にリアルタイムで写している映像を tPad の LCD ディスプレイに表示させる為の制御を行い、図 2 では LCD Controller に該当する。本研究で使用した tPad のディスプレイには画素数が 800 × 600 の映像信号範囲があり、その中で 640 × 480 の範囲が画面に表示する領域となる。残りの非表示領域のタイミングには水平と垂直の同期信号があり映像の開始点を伝える為に使用する [4]。RAW2RGB で処理されるデータ変換によって取得した 24bit の RGB データを 1pixel ずつ連続で 640 × 480 の領域に表示させる。

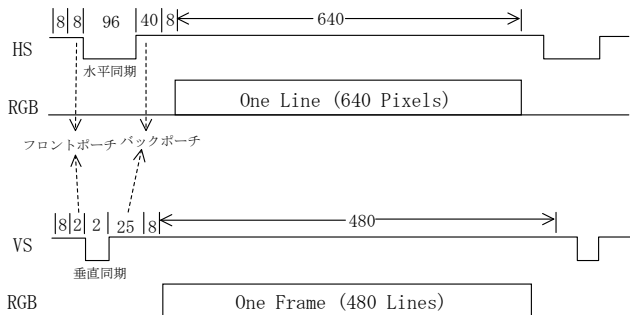


図 4: LCD 信号タイミング

図 4 は 24bit の RGB データを LCD ディスプレイへ書き込む同期信号の詳細を表している。これは VGA (Video Graphics Array) ディスプレイの場合と同じ手法となる。水平クロックを表す HS と垂直クロックを表す VS があり、同期信号の前後にはバックポーチとフロントポーチの部分があり、HS ではバックポーチが 40dot、フロントポーチが 8dot、VS ではバックポーチ

チが 25line, フロントポーチが 2line で表されている. さらに表示の前後境目にはボーダも 8dot と 8line で表されているが, 全て非表示領域となっている [5]. 本研究では, リアルタイムの映像を LCD ディスプレイに表示させ, SD メモリーカードに保存させる為に撮影した画像を VGA ディスプレイに表示させる. どちらも出力アルゴリズムは同じである為, VGA インターフェースの説明は省略させて頂く.

2.3 SDRAM コントローラ

画像ファイルは画素数が同じ場合であったとしてもデータ容量が異なる事が出てくる. 連続的にデータを転送させる際に, 転送速度の変化が生じる為, 負担に繋がる場合がある. 本研究では, データの転送速度に負担をかけず, 画像の転送速度を向上させる事を目的として高容量にも対応できる SDRAM コントローラを導入した. コンピュータのメモリ装置である RAM (Random Access Memory) は CPU の高速化と共に進化し, 現状では SDRAM が最も高速で主流な RAM として使用されている. メインメモリを SDRAM にする事で大容量でも高速にアクセス出来, 全てのメモリが同一のクロックに同期して動作する CPU とメインメモリ間の待ち時間は短縮され動作速度は向上する. しかし, SDRAM 以降の DRAM はコマンド制御が必要となり, CPU のバスに直結する事は出来ず専用のコントローラを経由してアクセスする必要がある. その代わり, 連続的に高速で読み書き出来る機能があり DRAM の持つ課題である低速性を補う役割を果たしている. DE2-115 ボード上の SDRAM チップは 2 つあり, 512Mbits (64Mbytes) の容量を持っており, 各チップは $8M \times 24bits \times 4banks$ で編成される [5].

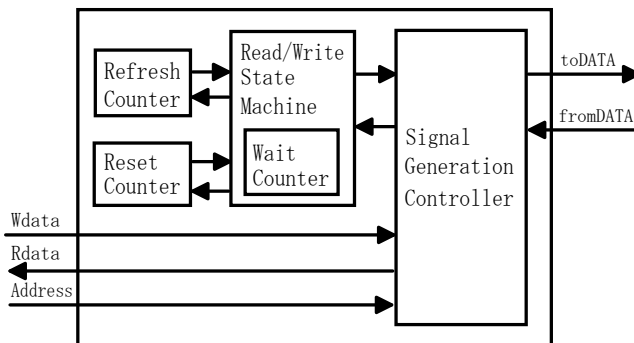


図 5: SDRAM コントローラ

図 5 は SDRAM の制御回路を表している. Wdata が SDRAM へ書き込むデータ, Rdata が読み出すデータ, Address がデータの格納先である SDRAM のアドレスを表す. Reset Counter は最初のデータの読み書き前に SDRAM の状態を初期化させる. Refresh Counter は連続的なデータの読み書きの際に, 1 回 1 回動作前の状態に戻す必要がある為, その為にリフレッシュをする. Read/Write State Machine では 2 つのカウンタによる命令を制御させ, その中の Wait Counter は 1 つのデータを読み書き中に次のデータが読み書き出来ないよう状態を維持させる. そして, Signal Generation Controller で SDRAM の制御信号, アドレス信号, バンクアドレス信号, 格納するデータの信号を生成し, SDRAM 本体へ送信される. 受信の場合はその逆となる.

2.3.1 PLL 処理

PLL (Phase Locked Loop) とは入力信号による基準周波数と出力信号の周波数の誤差を失くす電子回路を表している. 両信号との位相差を検出し, 回路のループを制御する事で同期した周波数の信号を出すことが出来る. 携帯電話や無線機の周波数制御の他, マイクロプロセッサの内部クロック周波数を外部クロックの整数倍にして高速化するものに用いられている.

図 6 は PLL を導入した際の制御の流れを表している. SDRAM

はクロックを必要とするメモリである為, これに接続するシステム側も同じクロックを与えた場合, 基板上での遅延などにより SDRAM 入力の AC 特性を満たさなくなる危険がある. そこで FPGA 内蔵の PLL を用いて, SDRAM に与えるクロックをシステムのクロックに対して少しずらすことで, 特性を満たすように対策が出来る. FPGA 上のクロックを PLL に通し, 周波数が同じ 2 つのクロックを作成する.

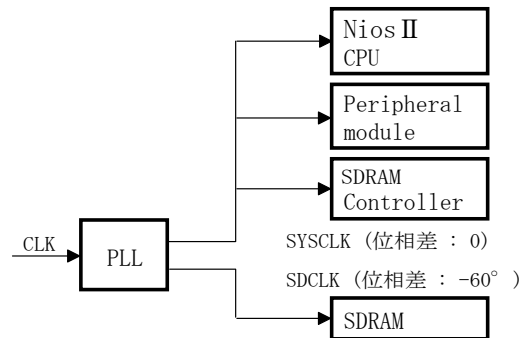


図 6: PLL 処理の構造

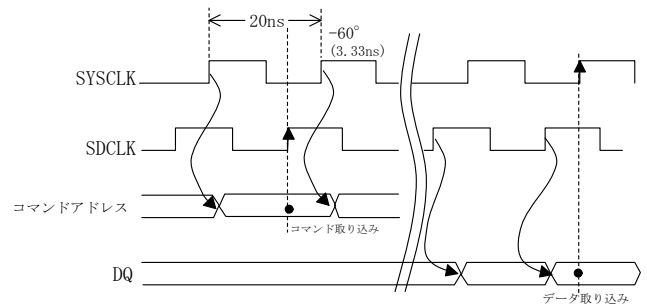


図 7: システムクロックと SDRAM クロック

図 7 は PLL を導入した上でシステム側のクロック SYSCLK と SDRAM のクロック SDCLK を表す. 位相差 0 のクロック SYSCLK を Nios プロセッサや SDRAM コントローラに与え, 位相差 -60° のクロック SDCLK を SDRAM に与える. 位相差 -60° とは基準クロックに対して位相が 60° 進んでいる状態を表す. 1 周期は 360° であるので, $1/6$ 周期分 SDCLK が早く変化する. これにより SYSCLK で作成したコマンドやアドレスの出力を, 余裕をもって SDCLK で取り込むことが出来る. 逆に SDRAM からの出力を SYSCLK で取り込む場合も確実に出来るよう, 位相差が大き過ぎない値とする [5].

2.4 SD メモリーカードインターフェース

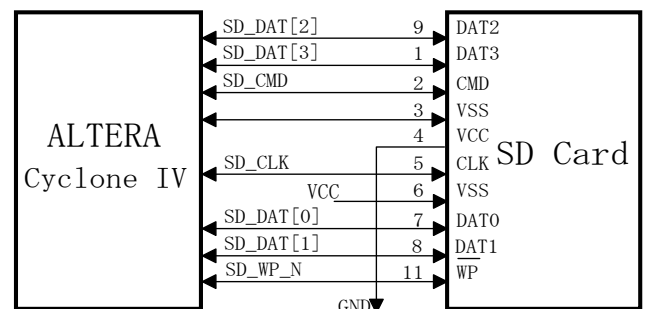


図 8: FPGA と SD カード間の接続端子

SD メモリーカードは高いメモリ容量, 高速なデータ転送速度, 大きな動きの柔軟性, 優れたセキュリティを有する半導体

フラッシュメモリがベースとなった新世代のストレージ装置であり、様々な小型記憶媒体（メモリーカード）が存在する中でもより小型化させている。主にデジタルカメラ等の携帯機器を主として幅広く適用されている。本研究では FPGA の DE2-115 に付いている terasic 2GB を使用した。SD メモリーカードは、マルチメディアカードを拡張させて、著作権の保護機能が装備されている製品であり、データの転送速度の規格が独立して決められているので、万一容量が同じであったとしても転送速度が異なってくる [6]。本研究で使用した FPGA, DE2-115 の場合では標準サイズの SD メモリーカードソケットが組み込まれている。SD メモリーカードには 9 つの端末があり、電源はその中で 3 つ占めているので有効な信号線の本数は 6 つである。データの送受信はブロック (512byte) 単位となり、データ伝送に関してはシリアル通信のクロックで行われる。図 8 は SD メモリーカードソケットと FPGA 上で設計した SD メモリーカードのコントローラ間の対応信号図を表しており、示された入出力信号によるピン・アサイン一覧を表 1 に記している。

表 1: ピンアサイン

Pin Name	Pin Number	Signal Name
SD_DAT[3]	PIN_AC14 (1)	CS
SD_CMD	PIN_AD14 (2)	DIN
GND	- (3)	Vss
-	- (4)	Vcc
SD_CLK	PIN_AE13 (5)	CLK
GND	- (6)	Vss
SD_DAT[0]	PIN_AE14 (7)	DOUT
SD_DAT[1]	PIN_AF13 (8)	-
SD_DAT[2]	PIN_AB14 (9)	-
SD_WP_N	PIN_AB14 (11)	WP

SD.CLK はコントローラから SD メモリーカード本体へ信号を送る為のクロックを表す。SD.DAT[2], SD.DAT[1], SD.DAT[0] は Nios II のシステムと SD メモリーカードとの間のデータ伝送の制御をする役割がある。SD.DAT[3] は SD メモリーカードの選択信号を表している。SD メモリーモードではその幅は 4bit で設定されているように、パラレルデータ幅の最大ビット数は 4bit である事が可能である。SD_CMD はコントローラからコマンドを送信し、SD メモリーカードから受け取る。SD_WP_N はカードのライトプロテクトを検出する。ライトプロテクトとは SD メモリーカードなどに備えられた書き込みや削除を禁止した状態を表しており、「書き込み保護」「書き込み禁止」とも呼ばれている [7]。

2.4.1 SD コマンドフォーマット

6bit		32bit	7bit
0	1	Command code Command Parameters Checksum Code	Check code 1

図 9: SD コマンドフォーマット

図 9 は SPI モードにおける SD メモリーカードのコマンドフレームは 48bit の固定長で表されている。そのうち上位から 2bit がスタートコードを表し、次の 6bit が Command code (コマンドコード) で文字通りコマンドの番号を表す。そして 32bit で表される Command Paramaters (引数でコマンドパラメータ) がデータを読み書きする SD メモリーカード内のメモリアドレスの役割を果たし、万一コマンドパラメータ無しでコマンドを使用するとしても、SD メモリーカードまたはホスト側にコマンドを送信した際に 32bit の空のパラメータを送信する必要がある。この場合はパラメータを任意の値とすることが出来る

ので、SD メモリーカード側が自動的に無視出来る。そして、次の 7bit が CRC (チェックコード) を表し、最後の 1bit がストップビットを表す。

2.4.2 SPI モード

SPI (Serial Peripheral Interface) とは、オンボード IC 間にある通信の方式の一つで以前存在したアメリカ合衆国の電子・通信機器メーカーのモトローラが提唱したものである。SPI モードは、SD メモリーカードが普及される前からある MMC (Multi Media Card) / SDC (Secure Digital Memory) を簡単に使う為に設定されたモードである。SPI モードの伝送プロトコルは本来のモードよりも単純なので、マイコン内蔵の SPI ポートで簡単に制御することが出来る。ちなみにデータの入出力方向は固定され、バイト方向のシリアル通信となる。表 2 は通常のデータの読み書きで使用される SPI コマンドの一覧を表している。それぞれのコマンドは CMD (n) のような形で表現される。n はそのコマンドのインデックス番号を示し範囲は 0~63 となっている。その中で今回使用するコマンドは初期化の際に必要な CMD0, CMD1, CMD8, CMD9, ACMD41, CMD55 とデータブロックを読み出す CMD17 とデータブロックを書き込む CMD24 がある。

表 2: SPI コマンド

index	Command description
CMD0	ソフトウェアリセット
CMD1	初期化開始
ACMD41	SDC 専用, 初期化開始
CMD8	動作電圧確認 (SDC V2 専用)
CMD9	CSD の読み出し
CMD10	CID の読み出し
CMD12	リード動作の停止
CMD16	R/W ブロック長の変更
CMD17	シングルブロックの読み出し
CMD18	マルチブロックの読み出し
CMD24	シングルブロックの書き込み
CMD25	マルチブロックの書き込み
CMD55	アプリケーションの特化コマンド
CMD58	OCR の読み出し

2.4.3 ファイルシステム

DOS のファイルシステムにおけるディスク内のファイルの位置情報などを記録する為の領域である。これが転じて現在は MS-DOS に採用されていた FAT (File Allocation Table) を用いるファイルシステムの名前としても FAT と呼ぶようになった。その中でもクラスタ番号を 16bit で管理しているのが FAT16 であり、これが FAT32 になると SD メモリーカードは大容量となる。SD メモリーカード内の FAT16 ファイルシステムの構造は、四つの部分が含まれており、PBR, FAT, FDT, データ選択がある。FAT テーブルはこれらのパラメーターを介して計算することが出来る計算領域があり、クラスタ間で保存されたファイルのリンク情報を記録する [6]。

2.4.4 SD カード初期化

SD メモリーカードインターフェースで用いられる初期化はデータを全て消すのではなく、SD カード用の書式フォーマットに設定させる事を意味している。手順としては CS を立ち上げ 100kHz~400kHz の周波数で 74 クロック程余分なクロックを送信する。次に CMD0 を送信しソフトウェアリセット、ACMD41 を送信し初期化を開始する。さらに CMD17,

CMD24 よりデータの読み書きを行う。最後に CMD9 で CSD を、CMD58 によりカードの電源電圧範囲を表す OCR を読みだすことで SD カードの情報を確認できる。

2.4.5 SD カードデータブロック書き込み

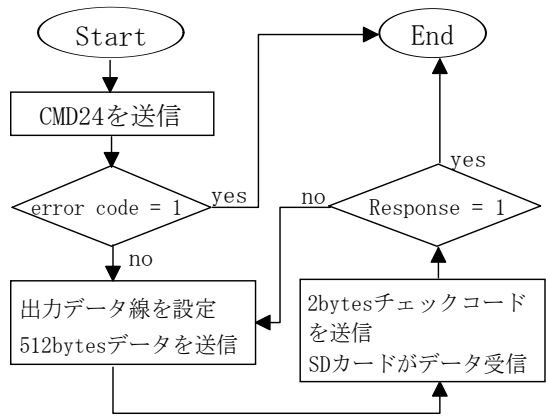


図 10: 書き込みフローチャート

図 10 はデータブロックを送信して SD メモリーカードに書き込むアルゴリズムをフローチャートで表している。ここでいうデータブロックとは取得した画像データの 1pixel を表している。SD メモリーカードへデータブロックを書き込む回路は複雑である。SD メモリーカード側が単一ブロックのデータを書き込むコマンド CMD24 を受信すると、システム側に応答した事を送信しデータが書き込まれるのを待つ。エラーコードが 1 でない場合、スタートコード 0xFE を送信し、システムから 512bytes のデータブロックを SD メモリーカードへ書き込む事が出来る。さらに、送信時に生じるエラーを検出する為に 16bit のチェックコード CRC を送信させる事でデータブロックの安全性を確認する。チェックコードが 00101 である場合、データブロックが SD メモリーカードに正しく書き込まれていることになる。もし、チェックコードの検証に失敗した場合送信データブロックは消去される。SD メモリーカード側がデータブロックとチェックコードを受信した後、それに対する応答をシステム側に返す。書き込みにはデータブロックとチェックコードを連続的に送信する時間を必要とする。全て送信し終わり、Response が 1 となれば完了となる。

2.4.6 SD カードデータブロック読み出し

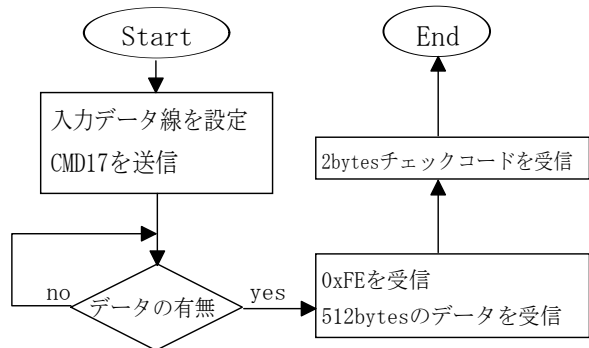


図 11: 読み出しフローチャート

本研究では、SD メモリーカードに書き込んだ画像データを後から読み出して確認出来るように試みた。図 11 は SD メモリーカードに保存されているデータブロックを読み出すアルゴリズムをフローチャートで表している。SD メモリーカードのデータ読み込みのアルゴリズムは単純である。まず単一ブロッ

クのデータを検索して読み出すコマンド CMD17 を SD メモリーカードに送信すると、システム側はスタートコード 0xFE を受信して CMD17 で送信されたデータのアドレスに応じてどのデータを読み出すか検索される。データを検索して発見した後、書き込み時と同じように 512bytes 連続してデータを読み出す。そして、受信時に時に生じるエラーを検出する為に 16bit のチェックコード CRC を受信させる事でデータの安全性を確認する。書き込み時と違って最後にチェックコードが出るのでそのまま全て受信し終わり完了となる。

3 システム実装

3.1 ハードウェア実装

本研究では、SOPC Builder という論理回路生成ツールを使用してハードウェアの設計を行う。この SOPC Builder は回路生成ツールの Quartus II 上で Verilog HDL を使用して回路を設計する際に Altera 社が提供する設計支援ツールであり、各種コンポーネントや IP コア、Nios 等を GUI 上で接続しながら設計することが出来る。Altera 社の Nios コア・プロセッサは世界で最も汎用性に優れているプロセッサであり、FPGA の業界でも最も広く利用されているソフトウェア・プロセッサである。コストに対する考慮やリアルタイム制御、アプリケーション処理等の様々な柔軟性がある [6]。

本研究では、この Nios II コア・プロセッサで設計した回路を FPGA の一種である DE2-115 へ実装させる事で組み込みシステムの設計を成り立たせる。Quartus II が FPGA のコンパイルや配置配線、シミュレーションによるタイミング検証、ダウンロード機能など全体をつかさどっている事に対して、SOPC はコンポーネント・エディタ機能を使用する事で Altera 社が提供する各種ライブラリ群 (IP コア) を一元管理し FPGA に実装する事が可能な回路を構築する事が出来る。ただし、SOPC Builder にサポートされていないコンポーネントもあるが、その場合は Quartus II にて Verilog HDL で設計した回路のファイルをコンポーネント化させる事でツール上にて設計に使用する事が可能となる。

Device Family: Cyclone IV E		Name		
		clk_0		
		clk_1		
		altpll_c0		
		altpll_c0		
Use	Connections	Name	Description	Clock
☒		camera_ctrl	cameraVGA	
☒		avalon_slave_0	Avalon Memory Mapped Slave	
☒		cpu	Nios II Processor	clk]
☒		instruction_master	Avalon Memory Mapped Master	altpll_c0
☒		data_master	Avalon Memory Mapped Master	clk]
☒		jtag_debug_module	Avalon Memory Mapped Slave	clk]
☒		jtag_uart	JTAG UART	clk]
☒		avalon_jtag_slave	Avalon Memory Mapped Slave	altpll_c0
☒		sdram_ctrl	sdram_ctrl2	
☒		avalon_slave_0	Avalon Memory Mapped Slave	altpll_c0
☒		altpll	Avalon ALTPLL	clk_inter
☒		pll_slave	Avalon Memory Mapped Slave	clk_0
☒		onchip_memory2	On-Chip Memory (RAM or ROM)	clk1]
☒		s1	Avalon Memory Mapped Slave	altpll_c0
☒		sd_cmd	PIO (Parallel I/O)	clk]
☒		s1	Avalon Memory Mapped Slave	altpll_c0
☒		sd_clk	PIO (Parallel I/O)	clk]
☒		s1	Avalon Memory Mapped Slave	altpll_c0
☒		sd_dat3	PIO (Parallel I/O)	clk]
☒		s1	Avalon Memory Mapped Slave	altpll_c0
☒		sd_dat0	PIO (Parallel I/O)	clk]
☒		s1	Avalon Memory Mapped Slave	altpll_c0
☒		pix_stb	PIO (Parallel I/O)	clk]
☒		s1	Avalon Memory Mapped Slave	altpll_c0
☒		pix_rd	PIO (Parallel I/O)	clk]
☒		s1	Avalon Memory Mapped Slave	altpll_c0
☒		pix_wr	PIO (Parallel I/O)	clk]
☒		s1	Avalon Memory Mapped Slave	altpll_c0

図 12: SOPC Builder で実装 (一部)

図 12 は本研究のシステムを SOPC Builder でハードウェア構成した図である。クロック周波数 50MHz を発生させる事で、FPGA 上でデータ転送をする為の環境が整う。カメラのインターフェース回路の部分を表す camera_ctrl とカメラでキャプチャした画像を VGA ディスプレイに出力させる回路を表す vga_ctrl, そして SDRAM 制御回路部分の sdram_ctrl は Verilog HDL で設計し, SD メモリーカード制御の部分を表す sd_dat, sd_cmd, sd_clk 等はツール上で用意されている通信コネクタのコンポーネントである PIO (Parallel I/O) を組み込む方法で設計した。全体を設計し, 実装ボタンの generate を押すことで回路上の問題の有無を自動で調べ, 異常がなければ完了となり, 3.2 で説明する SD メモリーカードの読み書きを行う為のソフトウェアを回路に実装させることが出来る。

3.2 ソフトウェア実装

次に, 本研究で設計したシステムのソフトウェア部分の開発を C 言語で行った。プログラムの記述とシミュレーションに関しては, オープンソース総合開発環境である Eclipse が基盤となる組み込みシステムのソフトウェア制御部分を設計できる Nios II Embedded Processor というソフトウェア生成ツールを使用した。本研究では SD メモリーカードへのデータの読み書き制御を C 言語で設計した。主な関数を下記に示す。

1. unsigned char sd_write (long address) :
SD メモリーカードに画像データのブロックを書き込む関数を表す。ここでは主に SPI 関数による処理を行う。関数内に定義された long address は書き込み先のブロックアドレスを表し, CMD24 に与える事で送信が開始される。
2. unsigned char sd_read (long address) :
SD メモリーカードに格納された画像データブロックを読み出す関数を表す。ここでも SPI 関数を主に使用するが, 後半のレスポンスによる if 文とそれは出るまでの for 文が無いので処理が若干楽になる。CMD17 に呼び出し先のアドレスを与えることで受信が開始される。
3. card_cson () :
SD メモリーカードヘデータブロックの書き込みや読み出しを行う前の段階として, スレーブセレクト (SS) を Low (0) の状態にさせておく。スレーブセレクトとはどの素子に対してリードライトを行うかという意味を表している。
4. spi_tx (int cmd number) :
SPI 通信におけるコマンド・コードを表す。関数内の cmd number は指定された送信番号を表しており, CMD にアドレスを与える際に使用される。アドレスは 24bit を 8bit ずつシフト演算しながら与えていく。
5. spi_getresponce (num of response, command number) :
書き込み先と読み出し先のアドレスである num of response と同時に指定した番号である command number の CMD を送信し, それに対するレスポンスを表す関数である。
6. card_csoff () :
SD メモリーカードヘデータブロックの書き込みや読み出しが終わった段階で, スレーブセレクト (SS) を High (1) の状態にさせておく。
7. void Line_writeSD (int frame, int line) :
データブロックの 1 ライン分を SD メモリーカードに書き込む関数を表す。関数内に定義された int frame は書き込む画像データが何枚目か, int line はデータブロックが何ライン目かを表す。ラインバッファを設けてから読み書きを行うが, ブロック・リードライトは 512byte しか読み書き出来ない関係から 1 ラインの左と右半分, 上位と下位バイトを分けて書き込む必要がある。今回の場合は sd_write 関数を 4 回 for 文でループさせて画像 1 枚の書き込み完了となる。
8. void Line_readSD (int frame, int line) :
データブロックの 1 ライン分を SD メモリーカードから読み出す関数を表す。sd_write と同じように 1 ラインの左

と右半分, 上位と下位バイトを分けて読み出す必要がある。今回の場合は sd_read 関数を 4 回 for 文でループさせ, ラインバッファを設けて読み出し完了となる。

4 実装結果と考察

まずはハードウェア設計において, Camera インターフェース回路, LCD コントローラ, SD メモリーカード, VGA ディスプレイの 4 つの I/O インターフェースによる回路と SDRAM コントローラの回路をハードウェア記述言語 Verilog HDL で, Nios II プロセッサによる自動生成で CPU を設計した。そして, 組み込み式, 回路生成ツールの SOPC Builder を使用し, 各モジュール同士を繋ぎ合わせて構成させる事が出来た。そして, C 言語によるソフトウェア設計において, Eclipse ベースの Nios II Embedded Processor 上で SD メモリーカードヘデータを読み書きさせるコントローラの構成が完了した。全体システムの FPGA 実装では FPGA によるカメラ, CMOS Sensor による撮影から Camera インターフェースの内部で RGB のデジタルによる色データへ変換されて取得するリアルタイム映像の RGB データを SDRAM へ一時的に補間し, そのデータの連続的な送信に従って LCD ディスプレイへリアルタイムの映像を表示させる事が出来た。図 13 は FPGA へ実装したその時の LCD ディスプレイの状態を示している。表示されているのは FPGA に搭載されたカメラから撮影したリアルタイムの映像であり, 640 × 480 の画素データによって映像が出力されている。



図 13: 実装後の LCD ディスプレイ

SDRAM 制御によってリアルタイムの画像を取得して SD メモリーカードへ保存させる為の転送速度が高速で動作するか計測する事が目的であったが, システム全体を FPGA へ実装させる際に正しく動作しなかった為, 計測させる所までは辿りつかなかった。原因はハードウェア設計の部分にあり, 各モジュール同士のピン接続か回路の記述にあると考えられる。

参考文献

- [1] Bei Yan, Yuefeng Sun, Fengfeng Ding, and Haiwen Yuan. Design of CMOS image acquisition system based on FPGA[J]. Industrial Electronics and Applications (ICIEA), 2011, pp.1726-1730.
- [2] DE2-115 User Manual : ALTERA terasic
- [3] Dongyun Wang, Xuehua Tao, and Rong Hu. The design of the interface for camera link and DM642[J]. Control and Decision Conference (CCDC), 2010, pp.2914-2916.
- [4] tPad User Manual : ALTERA terasic
- [5] 小林 優. FPGA ボードで学ぶ組み込みシステム開発入門 [Altera 編]
- [6] Altera. アルテラ製品カタログ Version 14.1 pp.64
- [7] Zhenlin Lu, Jingiao Li, and Yao Zhang. The Reading/Writing SD Card System Based on FPGA[J]. Pervasive Computing Signal Processing and Applications (PCSPA), 2010, pp.419-422.