NTNU
Norwegian University of
Science and Technology

A PCI Express communication interface for DMP camera arrays

Tor Arne Lye

Master of Science in Computer Science
Submission date: June 2010
Supervisor: Gunnar Tufte, IDI
Co-supervisor: Leif Arne Rønningen, ITEM

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

To be able to transfer the required amount of data generated by a DMP camera array a high throughput communication link is required. The large collection of data sources in such a camera array would benefit from serial communication. PCI Express offers high throughput serial communication. In this work the possibility of using PCI Express should be investigated and tested by the implementation of an FPGA prototype.

Assignment given: 15. January 2010
Supervisor: Gunnar Tufte, IDI

Abstract

Distributed Multimedia Plays (DMP) is a virtual collaboration system intended to provide real time audiovisual communication between multiple users. The system will produce near-natural picture and sound quality.

This report explores the requirements of a camera interface unit for DMP. This device interfaces with several image sensors and allows them to communicate on a common serial communications channel based on the PCI Express standard. The noteworthy features of PCI Express are outlined, and the standard is compared to the alternative Aurora communications protocol.

A functional prototype of a camera interface system has been implemented in VHDL and synthesised for an FPGA. The theoretical performance of this system is analysed and its suitability for use with DMP is evaluated. The results show that real time performance is possible with this architecture using a single PCI Express lane.

As PCI Express is originally an internal computer bus, a simple test has been performed in order to determine whether it can be employed as an external communications interface. The results show that reliable communication is possible across distances of 1.5 meters or more.

Contents

1 Introduction

The Distributed Multimedia Plays (DMP) [Røn07] is a virtual collaboration system intended to provide real time audiovisual communication between multiple users. This system will provide *near-natural* picture and sound quality, meaning that the quality should approach the limits of human perception. Parts of the DMP system will require technologies that are not yet available, but which have been projected to be developed within the coming years.

The DMP system is superficially similar to traditional videoconference and video telephony systems. Each user has one or more cameras, one or more display devices and a two way communication link for video and audio. A more recent evolutionary step in videoconference technology is the *telepresence* concept, basically a buzzword describing videoconferencing systems with higher audiovisual quality than the traditional solutions. One example is Tandberg's T3 product which supports three simultaneous video streams in 1080p HDTV resolution and CD-quality stereo audio using a 12 Mb/s (megabits per second) network connection. [Tan09]

DMP is the logical extension of this trend. DMP aims to become a system that enables virtual collaboration in new fields such as music production, song lessons, theatre, (figure 1b) opera, games and medicine (e.g. surgery). DMP will also enhance the experience in existing usage areas such as videoconferences (figure 1a) and video telephony.

(a) A work lunch (b) Collaborative theatre production

Figure 1: DMP usage scenarios.[Røn07]

To achieve all this, DMP will introduce new features such as wall-sized display devices covering one or more walls in a room, stereoscopic vision, multi-view and multichannel audio. Image resolution, colour depth and frame rate will have to be improved substantially over existing solutions in order to achieve near-natural

quality.

All these improvements will require significantly improved network resources compared to traditional videoconferencing solutions, in terms of both bandwidth and latency. Traditional videoconferencing systems often have time delays of several hundred milliseconds, which may be acceptable for speech. Collaborative music production on the other hand will require latencies in the order of 10–20 ms for the participants to be able to keep their tempo. The improved picture specifications may require network bandwidths of several tens of gigabits per second, depending on the amount of movement in the scene. [Røn07]

New display and camera technologies will most likely be required, and video encoding and decoding systems will probably have to be purpose-built hardware in order to support the bandwidths and latencies required. These parts can be implemented using field-programmable gate array (FPGA) devices, which have the advantage of being reprogrammable.

This report explores part of the design space for a DMP camera interface system. More specifically, the PCI Express interface standard is outlined and examined with focus on how it may be employed for video transport on a point-to-point serial link. A prototype of a camera interface unit, which reads parallel data from cameras and pushes that out on a PCI Express link, is implemented in synthesisable VHDL (VHSIC hardware description language; VHSIC: very-high-speed integrated circuit). Finally, options for a physical transport medium are discussed.

This report is organised as follows. Section 2 explains DMP in more detail. Section 3 details what requirements a camera interface unit must fulfil in order to be successfully used as part of a DMP system. Section 4 explains relevant parts of the PCI Express interface standard and signalling protocol. Section 5 describes the hardware devices most relevant to an implementation of a camera interface unit, and section 6 explains the relevant third party logic cores.

Section 7 details a VHDL implementation of a prototype camera interface unit. Section 8 explores the efficiency and protocol overhead of the PCI Express interface described in the previous section, discusses some alternative design choices that might have been used, and suggests some possible improvements.

Section 9 explores ways of extending the PCI Express electrical signals over a useful distance using different types of cables. Finally, section 10 summarises the report and makes some concluding remarks.

2 DMP — background information

The DMP system [Røn07] introduces substantially stricter quality requirements compared traditional audiovisual communication systems. A fully realised system may use wall-covering displays, high pixel densities and high bit depths, and special features such as adaptive multi-view and stereoscopic vision may be adopted. Traffic generation experiments have shown data rates of compressed video peaking at 60 Gb/s for part of a scene.

These new characteristics will require new systems for transport, encoding and decoding of image data. The realisation of the complete DMP system calls for work in several different fields of research. While the system as a whole is unlikely to be fully realised for several years, work is ongoing on central parts of the system, many of which can be designed and tested independently.

2.1 Video capture

Images will be captured using a camera array system, 3×3 being a likely configuration. A complete scene in DMP may be captured by several such camera arrays.

A DMP video capture system analyses the image and separates foreground objects from the background. Each object is handled separately, allowing important objects such as human actors to be encoded using parameters different from e.g. the ones used to encode the background object.

2.2 Quality shaping

A DMP system must be able to guarantee a maximum end-to-end latency. If the network is congested, network nodes may employ selective package dropping in order to avoid exceeding the latency requirements. Objects in a scene are therefore split into sub-objects, allowing the network to drop part of the information used to encode an object, without losing the entire object. As long as at least one sub-object is preserved, the object will be visible, but rendered at lower quality.

2.3 Network

DMP introduces the new networking protocol AppTraNet. AppTraNet combines functionality from IPv6 [DH98] and IPsec, [TDG98] and defines new DMP-spe-

cific functionality. In order to increase efficiency, network packets use only one packet header; the AppTraNet header, which combines the necessary parameters from the IPv6 and IPsec headers together with new DMP parameters.

2.4 Display

A wall-covering display may be constructed by combining smaller display devices, each with its own independent video decoder and network interface. The DMP package header includes the pixel address of the image data contained within the package. This allows packages to be routed directly to the correct display unit.

3 Project goals

The purpose of this report is to examine how to transport image data from multiple parallel camera interfaces through a serial interface, using the PCI Express protocol. This protocol was chosen because it is a well established industry standard for high speed serial communication.

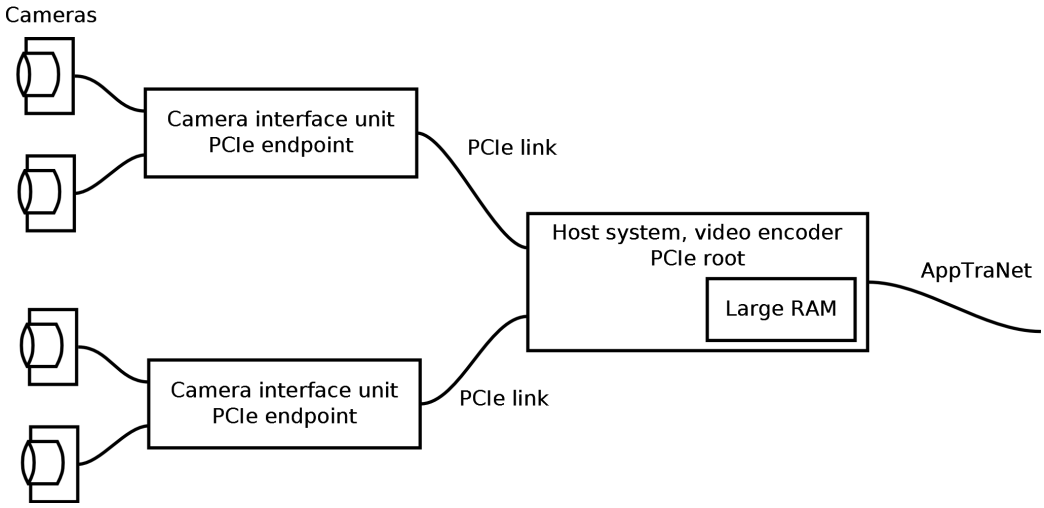

Figure 2: Complete video capture and encoding system

Video compression devices are likely do be located at least a few meters away from capture devices. Using for example a 3×3 camera array with a 10 bit parallel interface for each camera may require over 100 data lines in total, when auxiliary and control signals are factored in. Transporting multiple parallel signals for long distances quickly becomes impractical, so serialising this data would be of great advantage.

Figure 2 shows an example video capture and encoding system. Cameras are connected to devices that serialise the data and sends it to a host system through a PCI Express link. Each camera interface unit may manage several cameras, even a complete camera array if possible. The host system encodes the video data and transmits packets across the AppTraNet network. It may connect to several camera interface units, and is likely to require a relatively large system memory in order to compress several video streams at once. For camera modules that are configurable or controllable, the host system may issue commands through the PCI Express link.

The host system may be a computer which does video encoding in software, a computer that delegates encoding tasks to a dedicated FPGA in order to minimise

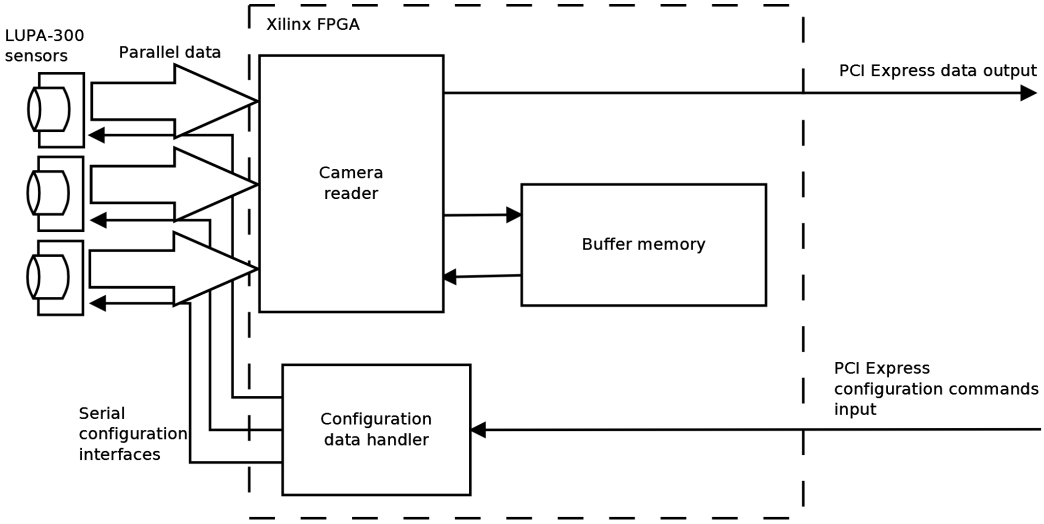

Figure 3: Camera interface unit overview

latency, or even a standalone device based only on FPGAs acting as PCIe root and video encoders.

The design will be targeted at the Cypress LUPA-300 image sensor [Cyp09] and a Xilinx brand FPGA. It should be capable of receiving data from image sensors operating at the highest possible picture resolution at 120 frames per second (fps). Figure 3 shows an overview of the camera interface unit architecture.

4 PCI Express

PCI Express [PCI05, PCI06, PCI10] or PCIe is a general purpose I/O interconnect standard, intended to replace the earlier PCI (now renamed 'Conventional PCI' to avoid confusion) and PCI-X standards. PCIe retains several attributes of the older standards but replaces their parallel bus architectures with a new scalable serial point-to-point interface and packet-based transmission.

4.1 Basic concepts

4.1.1 Differential signalling

Differential signalling is a technique which involves using two transmission lines to send a signal. One line sends the signal with positive voltage, and the other sends the same signal using negative voltage. At the receiver, these signals are combined using a subtractor (see figure 4). The opposite of differential signalling is single-ended signalling, which uses one transmission line per signal.

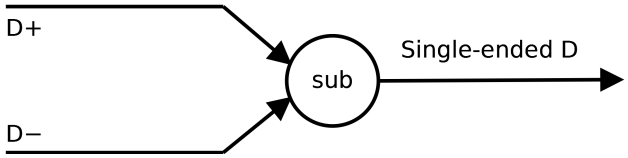

Figure 4: Differential signal pair and subtractor

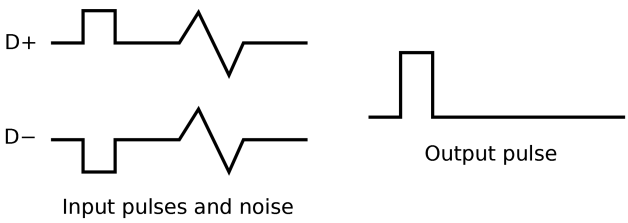

Figure 5: Signal pulses and noise on a differential pair

Figure 5 shows an example digital signal on a differential pair. The signal pulses are essentially amplified by a factor of two when they go through the subtractor, making differential signalling especially suitable for low voltage transmission. External interference tends to affect the two signal lines equally, and thus gets subtracted away at the receiver. This gives a high degree of noise immunity.

4.1.2 PCI Express Link

PCI Express uses two unidirectional communications channels, one for each direction of transmission, to make up a lane. These transmit and receive channels are each implemented as differential signal pairs, which means that a single lane consists of four data wires.

Lanes can be grouped together to form a PCI Express Link. The most basic link consists of one lane, but larger links can be formed by combining groups of 2, 4, 8, 12, 16 or 32 lanes. An extra differential signal pair is added to carry the clock signal. The width of a link is usually denoted using an 'x' followed by the number of lanes, e.g. 'x16'. The total bandwidth of a link scales linearly with the number of lanes.

4.1.3 Transmission Rate

A PCIe lane has a raw transmission rate of 2.5 gigatransfers per second (GT/s) per direction. The aggregate raw bandwidth of a link is 2.5 GT/s multiplied by the number of lanes. Second generation PCI Express devices (version 2.0 or higher) may optionally transmit at 5.0 GT/s per lane but are backwards compatible with the first generation transmission rate.

GT/s, GHz and Gb/s (gigabits per second) are, depending on the publication, used interchangeably as units of measure for the raw transmission rate. Version 1.1 of the PCI Express Base Specification uses Gb/s while version 2.0 uses GT/s.

PCI Express encodes data in a 8b/10b format, which means that every 8-bit byte is encoded and transmitted as a 10-bit word on the physical link. This is done to reduce the longest possible sequence of consecutive ones or zeroes, and to achieve DC balance. [WF83] This introduces a 20% overhead, so the actual data rate of a 2.5 GT/s lane is 250 MB/s.

4.1.4 Compatibility

PCI Express devices are forwards and backwards compatible, with regards to both link speed and lane width. Any endpoint can connect to any port, regardless of which link speed each of the devices support, and an endpoint may connect to a port with more or fewer lanes. The PCIe devices will negotiate for the highest mutually supported link speed and lane width. Some devices support disabling faulty lanes, reversing the order of the lanes if they are mapped incorrectly, or even inverting the polarity of the differential pairs if they are incorrectly connected.

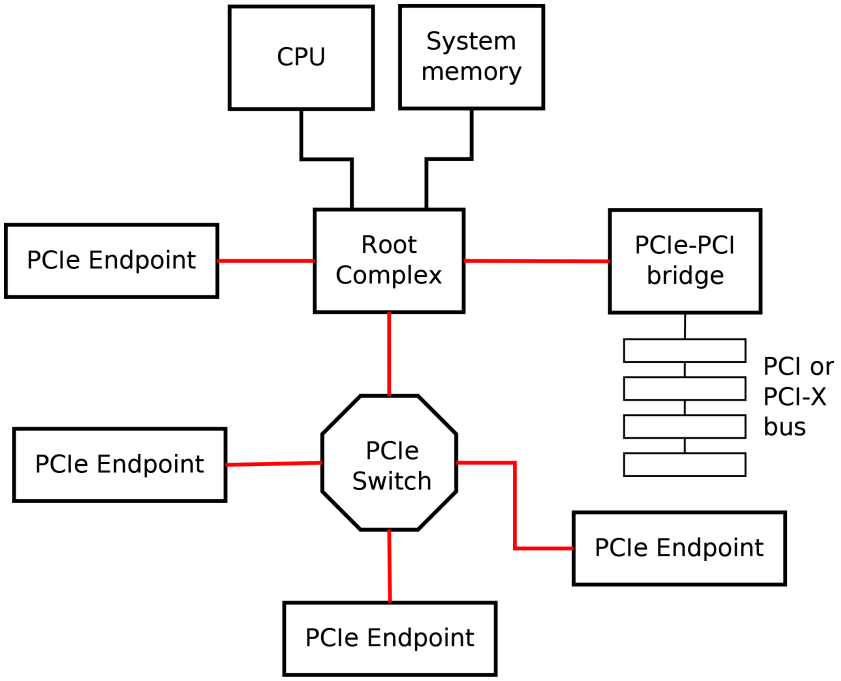

Figure 6: PCI Express topology. Red lines denote PCI Express links.

4.1.5 Topology

Figure 6 shows the topology of an example PCI Express based system. The PCI Express root complex is at the base of the PCIe I/O hierarchy. The root complex connects to the CPU and memory of the host system (e.g. a personal computer). The root complex has one or more root ports, (three in the case of figure 6) each port provides one link. The ports connect to PCI Express endpoints which may be any type of I/O device. If the root complex does not have a sufficient number of ports for the requirements of the system, it may be connected to a switch. A switch allows several endpoints to connect to one port in the root complex. The hierarchy may also incorporate a PCI-X or Conventional PCI bus, connected through a PCI/PCI-X bridge.

4.2 Layered architecture

The PCI Express architecture is specified in terms of three logical layers; the Transaction Layer, the Data Link Layer and the Physical Layer. Their basic relationships are shown in figure 7.

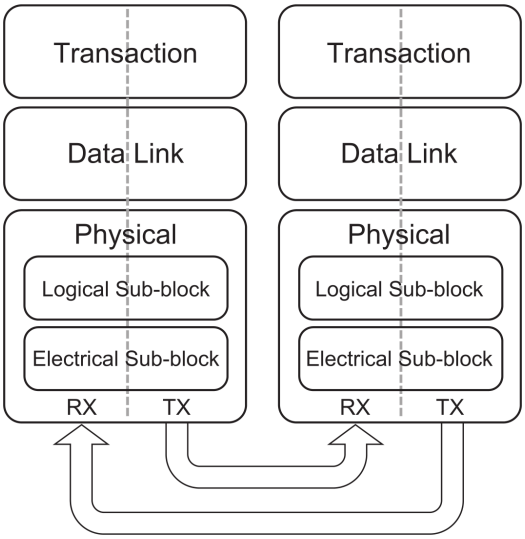

Figure 7: PCI Express layers. [PCI05]

4.2.1 Transaction Layer

The topmost layer is the Transaction Layer. This layer generates and consumes Transaction Layer Packets (TLPs). These packets communicate read and write transactions as well as certain other types.

4.2.2 Data Link Layer

The Data Link Layer is the intermediate layer. It it responsible for forwarding TLPs and for handling data protection codes and sequence numbering for the packets. The Data Link Layer can generate and consume Data Link Layer Packets (DLLPs) for the purposes of link management, for example TLP retry messages in case of transmission error, TLP acknowledgements, and power state requests.

4.2.3 Physical layer

The Physical Layer is responsible for converting information to an appropriate serialised format suitable for the link width currently in use. It handles negotiation for link width and frequency when the link is initialising, and takes care of 8b/10b encoding, plus hot-plugging, lane reversal, and polarity inversion in supported devices.

4.3 Transactions

PCI Express transactions are performed using requests and completions. There are four types of requests; message, configuration, I/O, and memory. The I/O request type is available for interoperability with Conventional PCI, and is deprecated in PCI Express. Configuration requests are used to access PCI configuration registers, and memory space requests are used to access memory-mapped locations. Message request is a new type which may be used to signal events between PCIe devices, and may optionally contain data.

A PCIe device responds to memory read, I/O read/write, and configuration read or write requests with a completion. In the case of successful read requests, the completion contains the requested data. For I/O and configuration writes, or any unsuccessful requests, a completion packet without data is used to signal the status of the request. Memory write requests and message requests require no completion.

Bulk data transfer is typically done using memory-mapped read and write transactions. As memory write transactions require no completion message, they are usually more efficient than reads.

PCI Express devices can use Base Address Registers (BARs) to request blocks of memory space in the host system's memory map. When the operating system assigns the appropriate address blocks, the BARs are programmed with the addresses. BARs can be 32 or 64-bit, and are numbered from zero and up, e.g. BAR0, BAR1, and so on.

By default, the root complex acts as bus master and initiates transactions. It is possible for endpoint devices to temporarily take control and act as bus master, this allows the endpoint to initiate data transfers on its own accord. This capability is commonly used to let endpoints perform Direct Memory Access (DMA) transfers to and from the host system memory, without involving the host CPU.

4.3.1 Protocol overhead

Figure 8 shows all the types of protocol overhead added to a TLP. DW here means double word, using the same definition as in the PCIe Base Specification; 4 bytes. The transaction layers adds a 3 or 4 DW packet header, depending on whether 32 or 64-bit addresses are used, and may add an optional 32-bit CRC field. The Data Link Layer adds a sequence number 2 bytes in size and another 32-bit CRC. The Physical Layer adds a one-byte framing symbol at the beginning and the end of the packet.

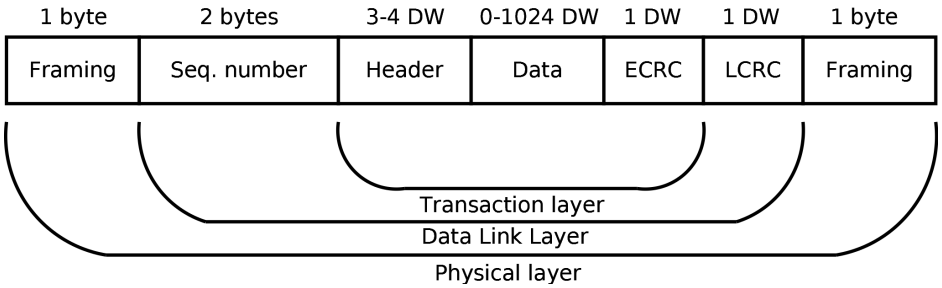

Figure 8: TLP overhead

PCIe devices define a maximum payload size. All devices must support 128 byte payloads, but some may accept payloads of up to 4096 bytes in a single packet. When a PCIe link is initialised, the devices negotiate for the highest mutually supported payload size.

4.4 PCI Express standards

PCI-SIG [PCI10] is the group that publishes most PCI-related standards. The base standard covering PCI Express is the PCI Express Base Specification. Version 1.0 was released in 2002. Version 1.1 with minor additions was published in 2005. The next major revision was published in 2007; version 2.0 added an optional 5.0 GT/s per lane transmission rate.

PCI-SIG has announced the PCI Express 3.0 specification, which will double the bandwidth again. While the raw transmission rate is increased only to 8.0 GT/s, the effective bandwidth will be doubled due to the removal of the requirement for the 8b/10b encoding scheme. [PCI07a]

The Base Specification covers architecture, protocol and interfaces, but does not define implementation details such as the electrical auxiliary signals, power supply, thermal requirements, or physical connectors. Several companion standards are available, providing these details for different types of implementations.

Popular implementations of PCI Express are the PCI Express Card Electromechanical Specification (CEM), [PCI03a] which covers add-on cards for ATX-compatible [Int04] desktop and server computers, PCI Express Mini Card Electromechanical Specification (Mini CEM), [PCI03b] covering miniature add-on cards often used in laptop computers, and the ExpressCard Standard, [PCM09] which describes a metal-encased hot-pluggable card for laptop computers.

5 Hardware

5.1 LUPA-300

The Cypress LUPA-300 [Cyp09] is a high speed image sensor. It has a resolution of 640×480 pixels with 10 bit precision. The sensor is capable of outputting pixels at a rate of 80 million per second, this is equivalent to 250 frames per second at the maximum resolution. If the image resolution is lowered, the frame rate can be increased as long as the pixel rate is 80 MHz or lower.

5.1.1 Configuration

The LUPA-300 is configurable through a serial three-wire interface referred to as Serial-to-Parallel Interface, or SPI. This interface should not be confused with the de facto industry standard Serial Peripheral Interface Bus, also SPI. [KK02] They are superficially similar, but Serial-to-Parallel Interface is unidirectional using three wires, while Serial Peripheral Interface Bus uses four and is bidirectional. The two interfaces also use the enable signal differently, but data transmission works identically.

The Serial-to-Parallel Interface can be operated while the image sensor's reset line is asserted. The SPI input shifts one bit of data into a receive buffer for each cycle of the SPI clock. When the enable signal is asserted after 16 cycles, the data is loaded into the internal configuration registers. The most significant 4 bits are address bits, while the remaining 12 bits are the configuration data. There are 16 configuration registers in total.

All registers are pre-loaded with configuration data that enable the sensor to operate at full resolution without further configuration. If modified, the configuration registers can be used to enable windowing (reading only a rectangular subset of the camera pixels), subsampling (skipping every other line in the Y direction), or to modify gain, read direction, timing and various calibration settings.

5.1.2 Data output

The LUPA-300 is driven by an externally applied clock running at up to 80 MHz. The image sensor outputs pixels through a 10 bit parallel interface, one pixel per clock cycle. By default, the pixels are read in sequence left to right, top to bottom. There is a short delay at the beginning of each line, the Row Overhead Time, (ROT) and a longer delay, Frame Overhead Time, (FOT) before each frame. With

default configuration settings these are 32 and 624 cycles respectively. If the pixel clock is running at a frequency of 40 MHz or lower, the ROT and FOT may be lowered by updating the appropriate configuration register.

The colour version of LUPA-300 uses a colour filter with a Bayer mosaic [Bay75] to filter the light that reaches the camera pixels. The specific mosaic arrangement used in Bayer filters can be seen in figure 9 and is intended to approximate the light sensitivity characteristics of the human eye. 25% of the total number of pixels are red sensitive, 25% are blue sensitive and 50% are green sensitive. The green pixels contributes most of the luminance (brightness) information, these are therefore represented in a higher number because the human eye is most responsive to luminance. Image data captured using a Bayer filter should be processed using a demosaicing algorithm to generate a full colour representation of the image.

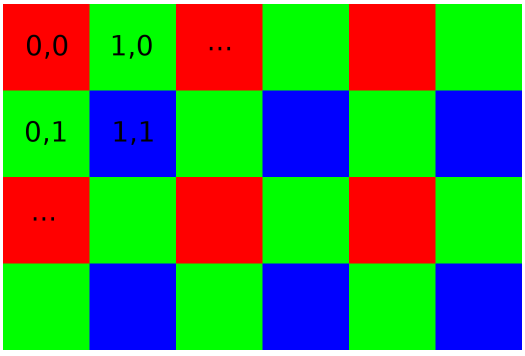

Figure 9: Bayer filter. The text labels indicate X,Y pixel coordinates.

The digital signals required to operate the LUPA-300 are listed in table 1. A point of confusion is that the SPI_DATA signal is listed as bidirectional in the pinlist within the LUPA-300 datasheet. [Cyp09] This seems to imply that configuration data can be read out on the same pin, this capability is not referred to anywhere else in the document however. In this report it has been assumed that 'bidirectional' is a misprint, and the pin is treated as an output pin only.

5.2 Xilinx FPGAs

Three recent generations of Xilinx FPGAs have been evaluated for this project. The Spartan-6, Virtex-5 and Virtex-6 [Xil10d, Xil09e, Xil10g] are the only Xilinx FPGA families that feature PCI Express Endpoint blocks. A Xilinx PCI Express Endpoint block will, when instantiated, implement much of the functionality of a complete PCIe endpoint using dedicated hardware. This means that the endpoint functionality will not consume as many of the FPGA's logic resources as would

Name	Type	Description
CLK	Input	Pixel clock, max. 80 MHz
RESET_N	Input	active low reset signal
DATA	Ouput	Pixel data output, 10-bit bus
LINE_VALID	Output	Indicates valid data at the pixel output
FRAME_VALID	Output	Indicates that the current pixel is part of a valid frame
SPI_ENABLE	Input	Causes a buffered SPI word to be written to registers
SPI_CLK	Input	SPI clock, max. 20 MHz
SPI_DATA	Input	SPI data input

Table 1: LUPA-300 digital signals

otherwise have been required.

All three FPGA families come in many different sub-families, which emphasise different features over others. The LXT variants are of particular interest; these focus on high speed logic circuits plus advanced serial connectivity. They feature dedicated circuits for high speed serial transceivers, which can be used as part of a PCI Express Endpoint block, or for other types of serial connectivity.

5.3 Xilinx evaluation cards

Xilinx provides various evaluation cards many of their FPGAs. The LXT variants of Spartan-6, Virtex-5 and Virtex-6 are all available on evaluation cards with PCI Express capabilities. [Xil09d, Xil09c, Xil10c, Xil10e, Xil10f, Xil10h] Each evaluation card is in the form factor of a PCIe CEM card, and has a PCIe CEM edge connector on the side. The cards can operate as computer add-in cards or as standalone devices. All the cards come with various other connectivity features such as networking and display interfaces.

The cards include connectors for user I/O. The Virtex-5 card uses the ubiquitous 2.54 mm headers, while the other two use VITA 57 FMC connectors [VIT10a, VIT10b] in the Low Pin Count (LPC) or High Pin Count (HPC) variants. All the lines on the FMC connectors can be used in pairs for differential signalling, or individually for single-ended communication.

Table 2 shows a comparison of the evaluation cards. Table 3 compares some key features of the three FPGA models used in the evaluation cards.

Card name	SP605	ML505	ML605
FPGA	Spartan-6 LXT XC6SLX45T	Virtex-5 LXT XC5VLX50T	Virtex-6 LXT XC6VLX240T
PCIe edge connector	x1	x1	x8
RAM	128 MB DDR3 (fixed)	256 MB DDR2 (upgradable)	512 MB DDR3 (upgradable)
Easily accessible user I/O	68 lines, FMC LPC connector	16 differential pairs plus 32 single-ended lines, all on 2.54 mm headers	156 lines on FMC HPC connector, 68 lines on FMC LPC connector
Price	495 USD	1195 USD	1995 USD

Table 2: Xilinx evaluation card comparison

	Spartan-6 LXT XC6SLX45T	Virtex-5 LXT XC5VLX50T	Virtex-6 LXT XC6VLX240T
Slices	6822	7200	37680
Max distributed RAM	401 kb	480 kb	3650 kb
Max block RAM	2088 kb	2160 kb	14976 kb
Max user I/O	296	480	720
PCI Express capabilities			
PCIe blocks	1	1	2
Lane configurations	x1	x1, x4, x8	x1, x2, x4, x8
Spec. supported	1.1	1.1	1.1 (up to 8 lanes) 2.0 (up to 4 lanes)
Maximum TLP payload size	512 B	512 B	1024 B
Roles supported	Endpoint	Endpoint	Endpoint or root port

Table 3: Spartan-6, Virtex-5, Virtex-6 LXT feature comparison.

6 Logic cores

6.1 Xilinx PCIe Endpoint block

The PCIe endpoint block in certain Xilinx FPGAs allows a PCIe endpoint to be implemented using dedicated logic rather than the general logic resources on the chip. For ease of implementation, Xilinx provides the CORE Generator tool to generate code that instantiates the core.

Using this tool, the PCIe block can be customised in numerous ways, such as setting the maximum number of lanes, the maximum link speed, Base Address Registers, device identification registers, and interrupt capabilities.

The frequency of the interface between the PCIe block and the user logic is configurable. All combinations of link width and link frequency have a default recommended frequency, but the designer can select a higher frequency if it is required for the user logic.

The Xilinx PCIe endpoint blocks and the reference design from CORE Generator feature no bus master capabilities, only a minimal programmed I/O (PIO) implementation of an endpoint. In order to support the requirements of this project, the endpoint must be extended with bus master capabilities.

6.2 Xilinx XAPP1052

Xilinx has published an application note [WA09] demonstrating a bus master DMA design combined with a Xilinx PCIe endpoint solution. It is more of a technology demonstration and benchmark application than something intended to be used in a production system. The feature set is minimal but it might be used as a starting point for developing a fully featured system.

6.3 PLDA EZDMA2

PLDA's EZDMA2 core [PLD09a, PLD09b] provides a comprehensive DMA solution for Xilinx PCIe integrated blocks. It will receive incoming transactions when acting as a slave, and can issue its own transactions when acting as a bus master. It also features interrupt capabilities and an interface for reading the PCI configuration registers.

Figure 10 shows how the EZDMA2 core interfaces with the Xilinx PCIe block and the application logic.

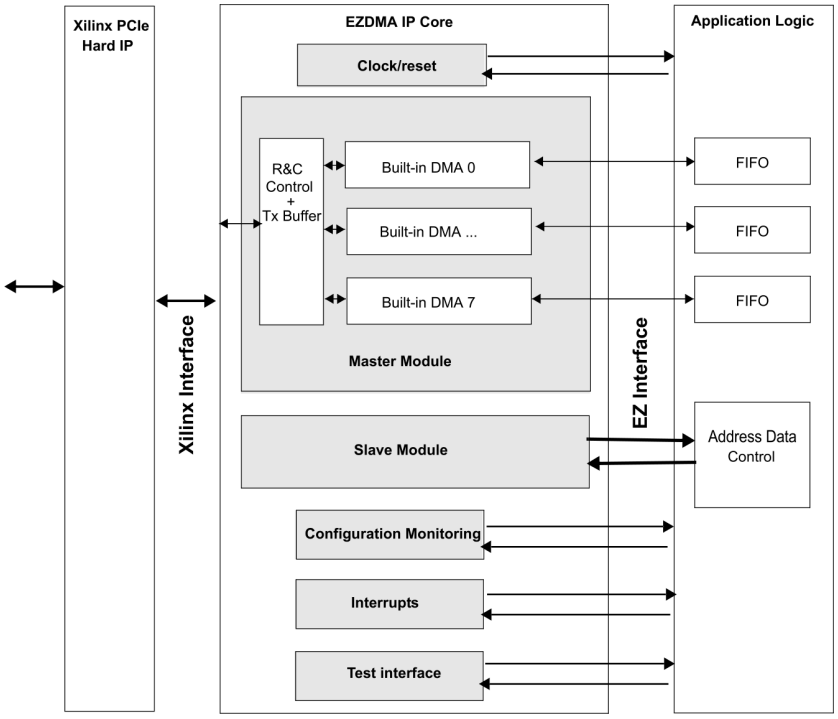

Figure 10: EZDMA2 overview [PLD09b]

6.3.1 Master module

The master module can implement up to eight DMA channels. When activated, the master module can be used to send read and write requests to the host system. DMA channels can execute seven types of commands: I/O reads and writes, memory read and write with a single data word, memory read and write bursts, and completion with data. I/O and data word transfers are four bytes in size, while burst transfers can be of any size. Completion with data is a special burst transfer type which is issued in response to a read request from the PCIe root.

The master module is connected to a memory defined by the application logic, either a FIFO device or a RAM-like device. Figure 10 shows a FIFO-based implementation. The module will autonomously read or write to the memory to fulfil read and write requests.

Each DMA channel is controlled by writing to two sets of registers, the DMA Channel Registers and the DMA Parameters. Both sets of registers must be programmed by the application logic before a DMA channel begins operating.

The DMA Channel Registers stores local and host memory addresses for transfers, plus the transfer size. During a transfer, the address registers are incremented by

the master module as each data word is read and written. The Transfer size register is decremented at the same rate. When the transfer size register reaches zero, the transfer stops.

The DMA Parameters store information about the memory device used for local memory, and defines which PCI Express command should be used for a transfer.

6.3.2 Slave module

The slave module receives incoming messages from the host system. Read and write requests are presented to the application logic, which should respond by asserting one of three signals; 'slave unsupported' in the case of an unsupported request type, 'slave abort' in the case of an error, or 'slave accept' to accept the transaction. If the request is a write transaction, the data to be written is presented at the slave module interface. The application logic must write the information to the local memory. If the transaction is a read request, the application logic can issue a DMA completion with the requested data.

6.3.3 Other interfaces

In addition to interfaces for the master and slave modules, the EZ Interface has signals for clock and reset. The clock signal is provided by the Xilinx PCIe core. There are also signals that allow the application logic to issue interrupts, and to allow it to read the PCIe configuration registers. Finally, there are a set of test mode signals which can be used during simulation and debugging.

7 Implementation

As part of this project, a prototype implementation of a camera interface unit has been created in synthesisable VHDL. The implementation is targeting the ML605 Virtex-6 evaluation board, as this model has the most versatile PCIe connectivity features.

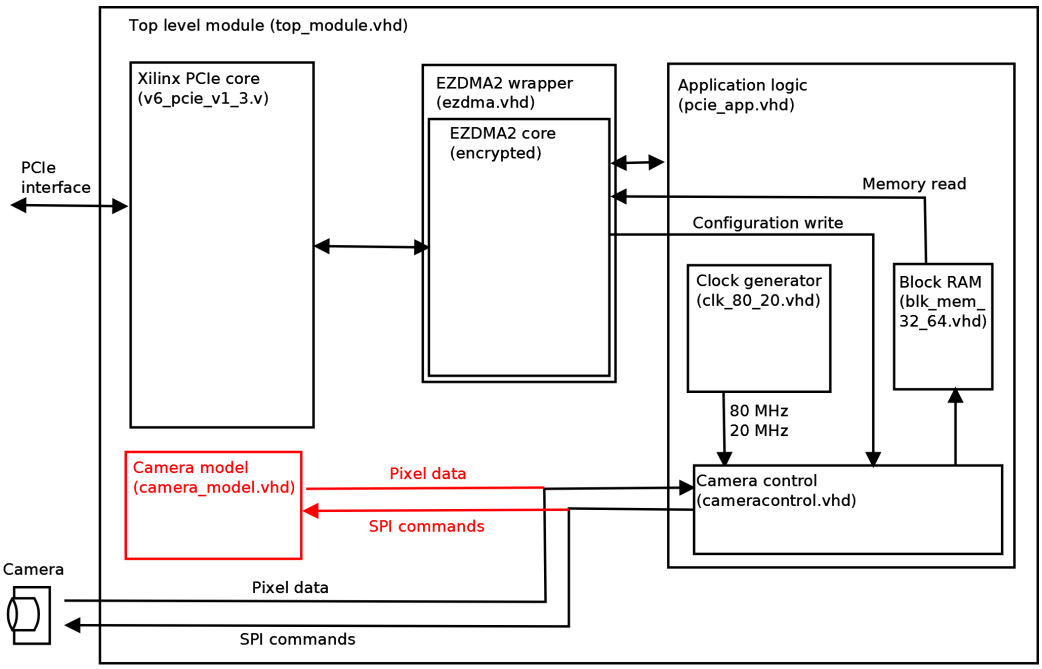

Figure 11: Camera interface unit implementation

The implementation is based on a Xilinx PCIe Endpoint and the PLDA EZDMA2 core. The basic outline can be seen in figure 11. In lieu of an actual LUPA-300 image sensor, the design instantiates a camera model (shown in red) which simulates its behaviour.

The implementation works as follows. The camera control unit reads data from the camera and writes it to a RAM. The data is written to the PCIe bus using bus master DMA transactions. The RAM is mapped to the host system address space using BAR1. In the current implementation this serves no specific purpose as the RAM is only used as an image buffer, however future implementations may use it for other purposes as well—some of which may require the memory to be accessible from the host system.

The LUPA-300 configuration registers are also memory-mapped, in BAR0. If the host system writes data to this location, the data will be written to the camera

using the SPI interface.

7.1 EZDMA2 core

The EZDMA2 core is instantiated using a VHDL wrapper file, which in turn is generated by a wizard application provided with the core. EZDMA2 version 1.4.3, build 185 has been used in this implementation.

The core was configured to use a TLP payload size of 256 bytes and a clock frequency of 250 MHz, it was also configured to support four outstanding requests, one DMA channel, a local address width of 16 bits and local memory read latency of one cycle.

7.2 CORE Generator components

The following components were created using Xilinx CORE Generator. They all instantiate specific hardware resources on the FPGA chip, and CORE Generator is able to generate the appropriate wrapper code based on configuration data.

7.2.1 Xilinx PCIe core

This component instantiates the PCIe endpoint block. This implementation uses version 1.3 of the Virtex-6 Integrated Block for PCI Express. [Xil10b] Newer versions are currently available, but 1.3 is required for compatibility with the ML605 card. [Xil10a]

The core is configured to x4, 5.0 GT/s mode, and has an interface frequency of 250 MHz and a maximum TLP payload of 256 bytes. The base address registers were configured to use 32-bit addresses, and a memory size of 128 bytes on BAR0 and 512 kB on BAR1. To ensure compatibility with the EZDMA2 core, the 'Trim TLP Digest ECRC' feature was enabled. This removes the CRC field from incoming TLPs.

7.2.2 Block RAM

The block RAM component instantiates 512 kB of FPGA block memory. It is used to store pixel data between read and transmit.

This component was generated using Block Memory Generator 3.3, [Xil09a] and was configured to act as a simple dual port RAM; i.e. one input port and one

output port. The write port has a width of 32 bits and the read port 64 bits.

With a width of 32 bits, the logic circuits can write three pixels at a time every third cycle (relative to the camera pixel clock), or if nine cameras are used, one pixel from each camera can be written using three memory accesses and a memory clock rate three times that of the pixel clock. 2 bits out of each 32 bits of memory are wasted however.

The read port must be 64 bits because it interfaces with the EZDMA2 core. The write port should optimally have a width divisible by ten, because it is used to store 10-bit pixels. The width of the two ports can, however, only differ by a factor of 1, 2, 4, 8, 16, or 32.

7.2.3 Clock generator

The clock generator was generated using Clocking Wizard 1.4. [Xil09b] It takes one input clock; the 250 MHz PCIe core interface clock, and is configured to provide two output clocks of 80 and 20 MHz. The former is used as the pixel clock for the LUPA-300 image sensor. The latter is the SPI clock. The component was configured to provide the 'Locked' output signal, which indicates when the output clocks are stable.

7.3 Application logic

This component constitutes the top module for the application logic. It instantiates a clock generator, a block memory, and the camera control component, and communicates directly with the EZDMA2 core. It uses three VHDL processes, one to handle bus master transactions, one for slave transactions, and the third to issue interrupts. Each process is based around a finite-state machine (FSM).

7.3.1 Master process

The master process FSM, shown in figure 12, starts in an *initialize* state, which programs the DMA Parameters and one of the DMA Channel Registers. It then transitions to the *idle* state, in which it does nothing except counting the number of pixel lines stored by the camera control module. When the number of buffered lines reaches a pre-set threshold, the FSM transitions to the *sendline* state, in which the DMA Registers are programmed with a local address and transfer size. The FSM now transitions back to the *idle* state, while the EZDMA2 core activates and starts the transaction.

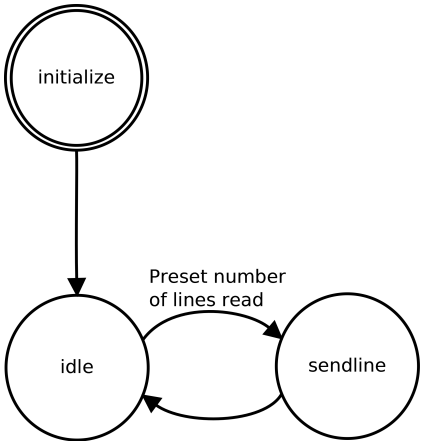

Figure 12: Master process FSM

Pixel data is sent using memory write transactions.

The master process sends a whole number of image lines per transaction. It waits until a certain number of lines are ready in the buffer before it sends the data. In the current implementation the number of lines per transmission is hard-coded to four, but it could be any factor of the picture height (480).

The destination address for the memory writes is hard-coded in this implementation, each successive frame sent to the host system will overwrite the previous one.

In a production system, the destination address and number of lines per transmission are likely to be configurable through registers. These registers might be programmed by a device driver on a PC host system, or the user logic on an FPGA-based host system.

7.3.2 Slave process

The FSM in the slave process (figure 13) starts in an *idle* state, and transitions to a new state when the slave write request or read request signal is asserted by the EZDMA2 core. If the incoming operation is a read request, or if it is a write request for a BAR different than BAR0, the FSM transitions to the *unsupported* state. Otherwise it transitions to the *accept* state.

The *unsupported* state aborts a request by asserting the slv_ur signal to the EZDMA2 core. Likewise, the *accept* state asserts slv_accept to accept the request. If the request is accepted, the *writebar2* state follows. Here, data from the EZDMA2 core is passed to the camera control component.

Data payloads in incoming write transactions are outputted from the EZDMA2 core in the form of one or more 64-bit words. In order to simplify the logic that handles configuration writes, any memory accesses to BAR0 will ignore bits other than the lower twelve. The rest of the input data is discarded. The lower twelve bits are written to the twelve bit configuration register indicated by the address of the transaction.

BAR0 is configured to address a 128 Byte memory, (the smallest possible memory size for a BAR) so the address is seven bits wide. The upper four bits are used to address the LUPA-300 configuration registers.

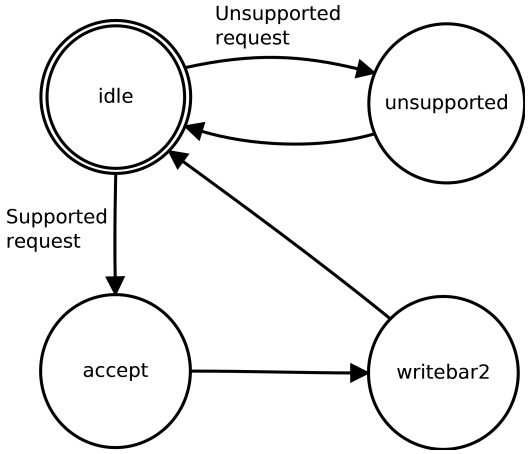

Figure 13: Slave process FSM

7.3.3 Interrupt process

The interrupt process sends an interrupt signal to the host system whenever a complete frame has been read from the camera. It uses the FSM shown in figure 14, which has two states; *idle* and *interrupt*. It transitions to the *interrupt* state when the frame_complete signal from the camera control component goes high. In this state an interrupt request is issued. The FSM switches back to the *idle* state when the EZDMA2 core acknowledges the interrupt request.

7.4 Camera control

This component handles the direct communication with an image sensor. It uses two VHDL processes, one to write SPI configuration data and the other for reading pixels. Each process is based on a finite-state machine.

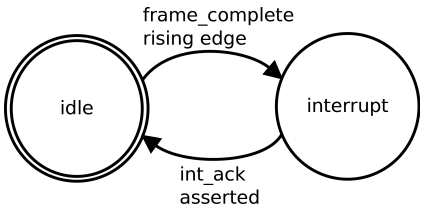

Figure 14: Interrupt process FSM

The camera operates on two clocks; the pixel clock at up to 80 MHz and the SPI clock at up to 20 MHz. Both of these clocks are generated by the clock generator instantiated in the application logic top module. As the clocks can take a while to stabilise, the camera control module will keep the camera's reset signal asserted until the clock generator reports a stable clock. The camera reset signal is also asserted when the SPI interface is active.

7.4.1 Camera read process

Under normal operation, the camera read process (figure 15) alternates between states *readpx* and *writepx*. In the former state, a pixel is read to an internal variable. Every third clock cycle the latter state is active, here the three pixels are written to the block memory as a 32-bit word. The FSM is in the *idle* state during the inactive periods after each line.

Because three pixels are written to memory as a 32-bit word, two bits are wasted per 32 bits. One line consists of $\lceil 640/3 \rceil = 214$ 32-bit words; 856 bytes.

The process signals the parent component every time it has written one complete line of the picture. At the same time it outputs the start address of the last line written to memory.

In the current implementation, only the first 401.25 kilobytes of memory are in use. The last 110.75 kilobytes are free to be used for other purposes.

7.4.2 SPI write process

This process uses the FSM shown in figure 16, and has three states; *idle*, *spiwrite* and *spicomplete*. When activated by the enable signal from the application logic, the FSM transitions to *spiwrite*. The 4-bit SPI address is concatenated with the 12 bits of data, and the complete 16-bit data word is shifted out on the SPI interface in time with the 20 MHz clock. When complete, the FSM transitions to *spicomplete* and asserts the SPI_ENABLE signal to the camera. This signals the

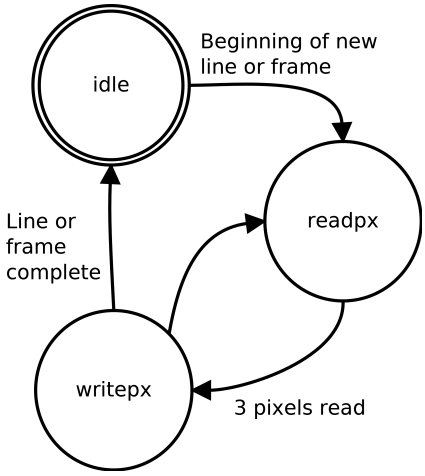

Figure 15: Camera read process FSM

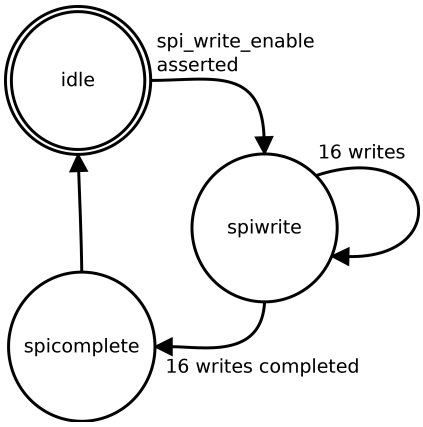

Figure 16: SPI write process FSM

camera to write the buffered data word to memory.

7.5 Camera model

The camera model can be used to emulate the data output of a LUPA-300 image sensor during simulation. It is synthesisable, so it may also be part of a design implemented on an FPGA. This will allow for testing of the implemented design without an actual image sensor attached. For testing of a design with multiple cameras, the camera module may be instantiated several times.

The camera model operates at the rate of its input clock. The image dimensions and overhead timings are specified using generics. If the module is instantiated

using the default parameters, it will operate using the same default parameters as the actual LUPA-300 sensor (640×480 pixels, timings suitable for a 80 MHz clock).

7.6 Design synthesis

The design has been synthesised for the Virtex-6 XC6VLX240T FPGA. The maximum possible frequency is reported as 329.381 MHz; only 250 MHz is required for this design. Table 4 shows a summary of FPGA resource utilisation. It is evident that a smaller FPGA might have be used instead.

	Used	Available	Utilisation
Number of Slice Registers	1022	301440	0%
Number of Slice LUTs	2223	150720	1%
Number of fully used LUT-FF pairs	816	2429	33%
Number of bonded IOBs	27	600	4%
Number of Block RAM/FIFO	119	416	28%
Number of BUFG/BUFGCTRLs	5	32	15%

Table 4: Device utilisation summary

8 Evaluation

8.1 Transmission overhead

Because of protocol overhead, [PCI05, PLD09b] the maximum effective bandwidth of 250 MB/s per PCIe lane is impossible to reach. The protocol overhead depends on the transaction size and type of transaction, but can be calculated if the type of traffic is predictable.

Transmission latency might of course limit the usable bandwidth because the link doesn't get saturated. If it takes a long time to get a completion for a request, the link will stay idle much of the time. This issue can be alleviated by allowing the sender to have several outstanding requests at the same time. This will let the sender keep transmitting, at the cost having to use larger buffer memories. The number of outstanding requests for the EZDMA2 core is configurable, the optimal setting will depend the latencies and the memory resources of the system.

The implementation described in this report uses memory write transactions to transfer image data from the endpoint to the host system. This has the advantage of eliminating much of the risk associated with high latency, as the memory write transactions require no completion. As a result, these transactions will be able to utilise the available bandwidth more easily than other types.

The amount of overhead is dependant on the maximum TLP payload size, as a smaller payload means more packets for a given amount of data and thus a greater amount of the bandwidth is used for packet headers and footers.

This implementation has a maximum payload size of 256 bytes. If using a 3-byte TLP header and no transaction layer CRC, the protocol overhead is 5 DW per packet. (See section 4.3.1) This adds up to $1 - 256/(256 + 20) \approx 0.07$ or about 7% overhead when sending packets of the maximum size.

The current version of the design stores three 10-bit pixels in 32 bits of memory, i.e. 1920 useful bits per 256-byte TLP payload. This adds further overhead. Together this all gives an overhead of 13%, or about 217 MB/s of usable bandwidth per lane.

Each line of the image is stored in memory as 856 bytes. This is obviously not divisible by the maximum TLP payload size, so the last TLP in an outgoing memory write request will not be of the maximum size. This adds a bit of further overhead, depending on the number of lines per transaction.

In order to issue a transaction containing four lines of the image, the core must transmit 14 TLPs, 13 with the maximum payload and one with 96 bytes. The final

estimation of the overhead (equation 1) means the usable bandwidth per lane is about 216 MB/s.

$$1 - \frac{4 \times 640 \times 10/8}{13 \times 256 + 96 + 14 \times 20} \approx 14\% \tag{1}$$

The LUPA-300 outputs 76.8 million pixels per second when running at full resolution and frame rate, this is equivalent to 91.55 MB/s. At 120 fps, the data rate is about 43.95 MB/s.

This means that one first generation PCIe link can transfer the data from two cameras at 250 fps, or four at 120 fps. To run the suggested configuration of nine cameras at 120 fps, an x2 2.5 GT/s link or an x1 5.0 GT/s link is sufficient. To run the cameras at 250 fps, an x4 2.5 GT/s link or an x2 5.0 GT/s link is required.

If the PCIe link has to fall back to a payload size of 128 bytes, the overhead is significantly impacted (equation 2). The total overhead rises to 19%, and wider PCIe links may have to be used to compensate.

$$1 - \frac{4 \times 640 \times 10/8}{26 \times 128 + 96 + 27 \times 20} \approx 19\% \tag{2}$$

These calculations show that adequate bandwidth is achievable with realistic link configurations; x1 or x2. A host system supporting TLP payloads of at least 256 bytes should be used for maximum efficiency, however.

8.2 Clocking

The pixel clock signal for the image sensor can not be generated by simple clock division, as the required frequency is not a factor of the application logic clock frequency. It must be generated by an external oscillator, or a clocking circuit implemented in the user application.

If the cameras operate at one specific frame rate at all times, it's a simple matter of using a single oscillator or clocking circuit to generate that one frequency. If multiple frame rates are required, a clocking circuit with multiple outputs may be used, or simple frequency divisors if the needed frequencies are all factors of the same reference frequency.

The frequency required for exactly 120 fps operation is $(336 + 480 \times (24 + 640)) \times 120 \approx 38.3$ MHz. Here, 640 and 480 are the frame dimensions, and 336 and 24 are, respectively, the frame overhead time and row overhead time for 20–40 MHz operation. A 40 MHz clock may be used for simplicity, this produces a frame rate

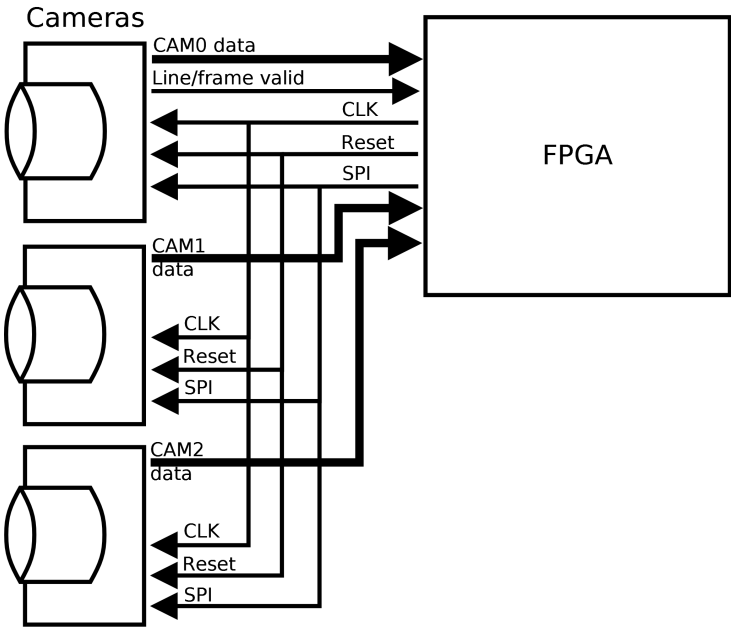

Figure 17: Multiple cameras connected to one FPGA

of 125 fps. To enable the correct timings for 20–40 MHz operation, the LUPA-300 'clock granularity' register must be updated.

8.3 Multiple cameras

The LUPA-300 image sensor operates according to an externally applied clock, and has well-defined timings for readouts and frame/line delays. Multiple cameras will operate in exact synchronisation if they are connected to the same clock and reset signals, and that makes it easy to connect several cameras to the same device.

Figure 17 shows a way to connect multiple cameras to one FPGA using this principle. There is only one set of outgoing control signals for clock, reset and SPI, and these are connected to all cameras. This ensures that they all operate synchronously and with the same configuration. Only one set of LINE_VALID and FRAME_VALID signals is needed, because the cameras are all synchronous.

This setup will require 90 data lines for nine cameras, 97 FPGA I/O pins in total when all auxiliary signals are factored in. The Spartan-6, Virtex-5 and Virtex-6 can all provide the sufficient number of I/O pins, but only the ML605 Virtex-6 evaluation card has enough pins accessible on the card.

With synchronous operation, it is easy to read out data from all cameras in a single

process, much like the one described in section 7.4.1. The process would read the same pixel coordinate from each camera at the same time—and could store these in memory using a fixed, interleaved pattern, like the one in figure 18.

With a 32-bit wide memory write port, this would require three writes to store the 90 bits of data for each pixel clock cycle. The memory write port would have to operate at a clock rate of at least three times that of the pixel clock; e.g. 240 MHz for a 80 MHz pixel clock or 120 MHz for a 40 MHz pixel clock. Most likely it would operate at the common user logic clock rate, which is 250 MHz in the suggested implementation.

CAM0 0,0	CAM1 0,0	CAM2 0,0	…	CAM8 0,0	CAM0 1,0	…	CAM8 639,0
CAM0 0,1	CAM1 0,1	CAM2 0,1	…	CAM8 0,1	CAM0 1,1	…	CAM8 639,1
…	…	…		…	…		…
CAM0 0,479	CAM1 0,479	CAM2 0,479	…	CAM8 0,479	CAM0 1,479	…	CAM8 639,479

Figure 18: Pixel interleaving pattern which enables write combining. Each labelled square represents one pixel. The rows contain 640×9 pixels, and each column is 480 pixels high.

The advantage of writing data from different cameras in such an interleaved fashion is that writes from three cameras are consolidated into one memory access, and that it makes data readout for PCIe transmission easier and more efficient when the data is stored continuously in memory.

8.4 Alternative communications protocols

The PCI Express protocol has the advantage of being a standard bus interface. In the particular application studied in this report however, the PCI Express link may be used to connect two custom-designed pieces of equipment. In that case, interoperability with third party products isn't an issue and using a widely accepted standard becomes less important. It is worth considering other protocols which may prove more useful.

If using a standard compliant PCI Express system, the endpoint must connect to a root port, which in turn is part of a root complex. While Virtex-6 PCIe blocks can act as root ports, Xilinx provides no easy way of implementing a full root complex. Such a feature must be implemented from scratch or be purchased from

a third party. Using an alternative communications protocol may eliminate the need for a root complex.

8.4.1 Xilinx XAPP869

Xilinx have provided a reference design that allows two Virtex-5 FPGAs to communicate directly using their integrated PCIe endpoint blocks. [JP07] This design does not adhere to the PCI Express standard, but it allows FPGA-to-FPGA communication using minimal logic resources. The design is compatible with the ML505 evaluation card, but is currently only available for Virtex-5 devices.

8.4.2 Aurora protocol

Xilinx have created their own communications protocol, specifically for inter-FPGA serial communication. The Aurora protocol borrows several features from PCI Express at the physical layer, but has key differences as well. Like PCIe, Aurora transmits data over scalable serial point-to-point channels. Each channel consists of one or more lanes. The devices at each end of the channel are called channel partners.

Unlike PCIe, Aurora channels can be unidirectional, which us useful in applications where high speed communication is only needed in one direction. A unidirectional circuit is simpler and uses fewer logic resources than a full-duplex one.

Aurora comes in two variants; Aurora 8B/10B and Aurora 64B/66B, named for the bit encoding schemes used. Aurora 8B/10B uses the same 8b/10b encoding as PCI Express, Aurora 64B/66B encodes 64 bits as 66 bits which significantly lowers the transmission overhead.

Aurora transmits user data as frames, which unlike PCI Express packets can have any length. This has the potential for further reducing the protocol overhead. Data can also be transmitted as a stream, which acts as a single, never-ending frame.

On an FPGA, the Aurora interface is implemented using the same serial transceivers as the PCIe endpoints. The number of lanes per channel which can be implemented on any given chip depends on the number of transceivers available, up to a maximum of 16. The transmission speed of the Aurora channel depends on the speed of the transceivers. Table 5 shows a summary of Aurora features for the same three FPGAs that were compared in section 5.3 on page 19. Spartan-6 and Virtex-5 are capable of faster than first generation PCIe signalling rates, while Virtex-6 is capable of faster than second generation PCIe rates.

	Spartan-6 LXT XC6SLX45T	Virtex-5 LXT XC5VLX50T	Virtex-6 LXT XC6VLX240T
Max. lanes per channel	4	8	16
Lane bandwidth	614 Mb/s to 3.125 Gb/s	500 Mb/s to 3.75 Gb/s	750 Mb/s to 6.5 Gb/s
Protocols	8B/10B	8B/10B	8B/10B 64B/66B

Table 5: Aurora feature sets of three FPGAs

Xilinx claims that Aurora is an open standard, free for anyone to implement without restriction.

9 PCI Express external cabling

9.1 PCI Express External Cabling spec

PCI-SIG released the PCI Express External Cabling Specification in 2007. This specification allows for 2.5 GT/s signalling, and x1, x4, x8, and x16 lane configurations. [PCI07b, Sol07] Several sideband signals are provided for auxiliary lines, in order to be compatible with existing implementations such as the PCIe CEM. The specification does not make assumptions about cable lengths, but PCI-SIG workgroups have reportedly used cable lengths of 0.5–7 meters.

The cable assemblies are based on copper wiring and specifically designed connectors. One cable carries at most four lanes, so x8 and x16 cable assemblies use two and four cables respectively.

Cables and connectors are commercially available, and at least one manufacturer claims to have 5.0 GT/s capable products, [Mol06] even though this is currently outside the specification.

9.2 HDMI Cables

High-Definition Multimedia interface [Hit06] (HDMI) cables have several properties that make them suitable for use as makeshift PCI Express external cables. Like PCI Express, HDMI uses differential signalling to transmit data. Standard cables contain four such differential pairs; three for data and one for the clock signal. Each pair is individually shielded to avoid crosstalk and other noise.

HDMI data is encoded using 10 bit characters, each character may represent 2, 4 or 8 bits of data. Each differential pair transmits one character per clock cycle. Since version 1.3 of the standard, the highest clock frequency is 340 MHz, which means that each differential pair has a maximum raw data rate of 3.40 Gb/s, or 10.2 Gb/s for all three pairs.

HDMI 1.3 defines three sets of connectors and receptacles; types A, B and C. Cables with type B connectors have dual-link capability (six differential data pairs) to double the available bandwidth. Cables with A and C connectors use the default single link with three data pairs. C is a miniaturised version of A.

Cables are certified based on the highest tested frequency; category 1 (marketed as “Standard”) cables are tested at 74.25 MHz, and category 2 (marketed as “High Speed”) cables at 340 MHz.

HDMI is designed to carry high definition video and multichannel audio, and

is commonly used in consumer electronics, especially high-definition television (HDTV) equipment. As such, HDMI cables are readily available at low cost due to extensive mass production. Connectors and receptacles which may be used in custom-designed electronics are available from several manufacturers, such as Molex or FoxConn. [HDM10c]

	PCIe Gen. 1 x1	HDMI Cat. 2 cable
Number of differential pairs, including clock pair	3	4
Reference clock	Implementation specific, 100 MHz for CEM	Up to 340 MHz
Max raw data rate per pair	2.5 Gb/s	3.4 Gb/s
Differential impedance	100 Ω ± 20%	100 Ω ± 10%
Diff. pair signalling voltage	1.2 V	1.2 V
Power supply wires	Implementation specific	5 V, 50 mA
Number of wires	Implementation specific	19

Table 6: PCIe and HDMI interfaces, comparison of key specifications.

Table 6 shows some key specifications of HDMI and a one-lane first generation PCIe link. Most of the entries indicate that a category 2 HDMI cable is well suited to be repurposed as a PCIe external cable. There are two caveats, however.

Power supply: PCI Express devices are often bus powered, but the details vary in different implementations. An x1 socket as defined in the PCIe CEM [PCI03a] can provide up to 16 W of power, using multiple 3.3 V and 12 V power pins. HDMI is evidently not designed to carry high currents—though it might work if multiple wires are used in combination. If the PCI Express endpoint is self-powered however, the HDMI cable won't be required to carry power at all.

Total number of wires: The number of wires used in a PCIe connection also varies depending on implementation. The PCIe CEM x1 socket has 36 pins, but many of these are mapped to 3.3 V, 12 V, ground and auxiliary lines. If all power lines and optional auxiliary wires are omitted, it is possible to implement the interface using only 12 wires: Nine wires for the three differential pairs, assuming

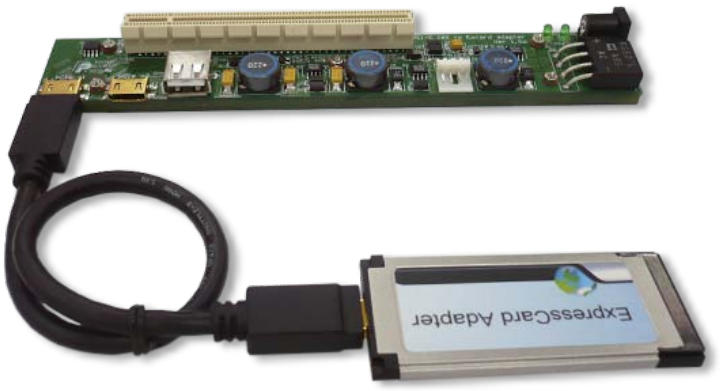

Figure 19: PE4H and E2C2 PCI Express/ExpressCard adaptors with HDMI cable

a separate ground wire for each pair's individual shield, two presence detection wires and one wire for the PERST# signal. The latter signal is used to indicate stable power supply and clock after power-up. If hot-plugging support is not required, the presence detect signals may be hardwired at the host side, saving an additional two wires.

This leaves several free wires in the HDMI cable, which can be used for specialised purposes. For example, a 'power-on' signal may be used to enable externally powered endpoints to power on at the same time as the host system.

9.3 PE4H: A commercially available adapter

A commercial product that uses HDMI cables for PCI Express is already available. [Bpl09] The PE4H is a passive PCI Express adapter designed by Taiwanese manufacturer Bplus Technology. The adapter has a x16 PCIe socket for desktop add-in cards, and provides two type C HDMI sockets which can connect the device to a host system. One HDMI cable may be used for a single lane link, or two cables for an x2 link. The adapter also provides connectors for an external power supply.

Three different secondary adaptors are available to connect the PCIe link to the host system. EC2C, PM3N and HP1A provide interfaces to ExpressCard, Mini PCIe and PCIe CEM, respectively. Figure 19 shows a PE4H connected to the EC2C ExpressCard adapter using a single HDMI cable.

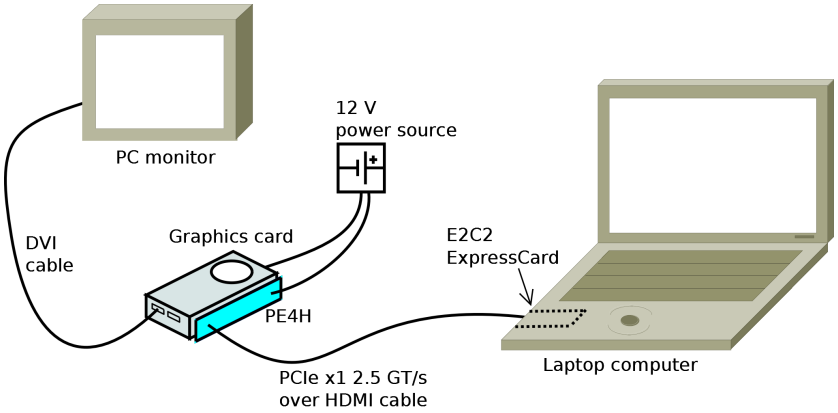

Figure 20: Graphics card test setup

9.4 Test results

For the purposes of this project, a PE4H and an EC2C adapter was tested using a laptop computer and a PCI Express graphics card; the ATI Radeon HD 5850. [Adv10] The graphics card and PE4H were powered using a 12 V power supply. The EC2C adapter was plugged into the laptop's ExpressCard slot, and connected to the PE4H using a HDMI cable. The graphics card was connected to an external PC monitor. The complete test setup is shown in figure 20.

After verifying that the test system powered on and that the graphics card operated normally, the card was tested using the 3DMark06 benchmark software from Futuremark Corporation. [Fut10] Using graphics benchmark is very likely to saturate the bandwidth of the PCI Express interface, and successful operation is as such an indicator of system stability.

The whole system was tested twice, first using a 30 cm HDMI cable supplied with the PE4H, and second using a 150 cm cable from an electronics retailer. Both were inexpensive unbranded cables, and it is not known whether they were category 1 or 2 certified. The system showed stable operation for the entire test period; over ten hours with each of the cables.

An unstable PCI Express link might work correctly but would cause a high number of retransmits, and this would reduce performance. The benchmark software showed identical results regardless of which cable was used, which would seem to indicate that both setups were of equivalent quality.

These rather informal tests suggest that HDMI cables are well suited for transporting PCIe signals, at least for shorter cable runs. More tests are needed however, to determine whether practical cable lengths are possible. As HDMI connectors

and cables are much more easily available than the purpose-built PCIe External Cabling Spec. products, HDMI may be a useful option in the short or long term.

As HDMI cables are rated for a transmission rate lower than that of generation 2 PCI Express, a generation 1 interface may have to be used. Even so, using a dual link HDMI cable or two single link cables, transmission of signals from nine cameras should be possible in the suggested resolution and frame rate, using a PCIe 1.1 x2 link.

HDMI Licensing, the organisation that publishes the HDMI standards, suggests that category 2 cables can be made in lengths of no more than five to eight meters. [HDM10b] Some manufacturers and retailers however offer cables of up to 15 meters which are claimed to be category 2 compliant. [Atl10, Kno10, Par10]

HDMI Licensing suggests using cables with active amplification, category 5 or 6 networking cable, coaxial cable, or fibre-optic cable for cable runs longer than 5–8 meters. [HDM10a] Similar solutions may work well for a PCI Express link, if required. With such special measures, a 5.0 GT/s link may be possible, which will enable a single link PCIe connection to be used.

10 Conclusion and future work

10.1 Conclusion

Implementing a camera interface system for DMP poses several challenges, some of which are addressed in this report.

The PCI Express interface standard's flexibility and scalability make it a useful candidate for a high speed serial communications link between cameras and the video encoder. Alternative protocols such as Aurora should also be considered an option however, as they may offer even grater flexibility and performance.

The prototype device implemented in this report demonstrates how a camera interface unit might operate. Analysis shows that the design is capable of forwarding data from nine cameras at full resolution and 120 fps if connected to a two lane first generation PCIe link or a one lane second generation link.

While the design proposed in this report was targeting a relatively powerful Virtex-6 FPGA, synthesis results indicate that a smaller device will suffice.

A simple practical test determined that HDMI cables may be used as a makeshift transport medium for first generation PCI Express signals. As HDMI cables and connectors are produced in high volume and low cost, they should be considered as a serious alternative to PCIe External Cabling Specification compliant cables.

10.2 Future work

Some of the topics discussed in this report will benefit from further study.

10.2.1 Host system; device driver

The camera interface unit cannot work without being connected to a host system. During testing it might be convenient to implement the design on an evaluation card with PCI Express, and to connect the card to a computer. For this to work, a software driver will have to be written. For the endpoint to operate without being connected to a computer, a complete host system with a PCIe root complex will have to be implemented e.g. on a different evaluation card.

10.2.2 Performance measurements

The predictions about transmission overhead and interface bandwidth should be backed up by real world experiments. While the predictions in this report are a close match with benchmark test results from Xilinx [WA09], they represent idealised best-case situations. Only physical tests can determine conclusively whether the suggested PCI Express interface configuration is sufficient.

10.2.3 PCI Express external cables

The PCI Express cable tests presented in this report were limited in scope by the availability of C-connector HDMI cables. Future tests may be performed with a PCIe electrical interface based on A-connectors, for which a much greater range of cable products are available.

As a host system with PCI Express 2.0 root ports was not easily available when the tests were performed, the HDMI cables were only tested at first generation PCI Express speeds. Future tests should be made to determine whether 5.0 GT/s speeds are achievable as well.

References

[Adv10] Advanced Micro Devices, Inc. *ATI Radeon HD 5850 Graphics*, 2010. http://www.amd.com/us/products/desktop/graphics/ati-radeon-hd-5000/hd-5850/.

[Atl10] Atlona. *15 m (50 ft) Altona HDMI digital video and audio cable*, 2010. http://www.atlona.com/15M-50FT-ATLONA-HDMI%2DDIGITAL-VIDEO-and-AUDIO-CABLE.html.

[Bay75] Bryce E. Bayer. *Color imaging array*. Eastman Kodak Company, July 1975. United States Patent 3,971,065.

[Bpl09] Bplus Technology. *PCIe passive adapter Ver1.0*, 2009. http://www.bplus.com.tw/PDF/PE4H_brief.pdf.

[Cyp09] Cypress Semiconductor Corporation. *CYIL1SM0300AA LUPA-300 CMOS Image Sensor*, October 2009. Rev. *F.

[DH98] S. Deering and R. Hinden. *Internet Protocol, Version 6 (IPv6) Specification*, December 1998. Draft Standard.

[Fut10] Futuremark Corporation. *Futuremark 3DMark06 Introduction*, 2010. http://www.futuremark.com/benchmarks/3dmark06/introduction/.

[HDM10a] HDMI Licensing, LLC. *HDMI :: Installers :: Running Long Cable Lengths*, 2010. http://www.hdmi.org/installers/longcablelengths.aspx.

[HDM10b] HDMI Licensing, LLC. *HDMI :: Resources :: Knowledge Base*, 2010. http://www.hdmi.org/learningcenter/kb.aspx.

[HDM10c] HDMI Licensing, LLC. *HDMI Approved Connectors*, 2010. http://www.hdmi.org/manufacturer/approved_connectors.aspx.

[Hit06] Hitachi, Ltd., Matsushita Electric Industrial Co., Ltd., Philips Consumer Electronics International, B.V., Silicon Image, Inc., Sony Corporation, Thomson Inc., and Toshiba Corporation. *High-Definition Multimedia Interface, Specification 1.3a*, November 2006.

[Int04] Intel Corporation. *ATX Specification, Version 2.2*, 2004.

[JP07] Sunita Jain and Guru Prasanna. *Point-to-Point Connectivity Using Integrated Endpoint Block for PCI Express Designs*. Xilinx, October 2007. XAPP869 (v1.0).

[KK02] David Kalinsky and Roee Kalinsky. Introduction to Serial Peripheral Interface. *Embedded Systems Programming*, 15(2):55–56, February 2002.

[Kno10] Knoxed Limited. *Ivuna 15m HDMI Cable*, 2010. http://ukhdmi.com/Ivuna-15m-HDMI-Cable_QQ101582.

[Mol06] Molex. *External PCI Express (PCIe) x1, x4, x8, x16 I/O Products*, 2006.

[Par10] Parts Express. *Dayton HR13HG15 High-Speed HDMI Cable 15m (49 ft.) V1.3 CL2*, 2010. http://www.parts-express.com/pe/showdetl.cfm?Partnumber=181-810.

[PCI03a] PCI-SIG. *PCI Express Card Electromechanical Specification, Revision 1.0a*, April 2003.

[PCI03b] PCI-SIG. *PCI Express Mini Card Electromechanical Specification, Revision 1.0*, June 2003.

[PCI05] PCI-SIG. *PCI Express Base Specification, Revision 1.1*, March 2005.

[PCI06] PCI-SIG. *PCI Express 2.0 Base Specification, Revision 0.9*, September 2006.

[PCI07a] PCI-SIG. *PCI-SIG Announces PCI Express 3.0 Bit Rate For Products In 2010 And Beyond*, August 2007. http://www.pcisig.com/news_room/08_08_07/.

[PCI07b] PCI-SIG. *Press release: PCI Express External Cabling Specification Completed by PCI-SIG*, 2007. http://www.pcisig.com/news_room/news/press_release/02_07_07.

[PCI10] PCI-SIG. *PCI-SIG Website*, 2010. http://www.pcisig.com/.

[PCM09] PCMCIA. *ExpressCard Standard, Release 2.0*, February 2009.

[PLD09a] PLDA. *EZDMA2 IP for Xilinx Hard IP, Getting Started*, December 2009. Version 1.4.2.

[PLD09b] PLDA. *EZDMA2 IP for Xilinx Hard IP, Reference Manual*, December 2009. Version 1.4.2.

[Røn07] Leif Arne Rønningen. *The DMP System and Physical Architecture*, 2007. http://www.item.ntnu.no/~leifarne/The%20DMP%2014Sep07/The%20DMP%20System%20and%20Physical%20Architecture%2016Sep09.htm.

[Sol07] Richard Solomon. *PCI-SIG Developer's Conference Europe 2007: PCI Express Cabling*. PCI-SIG, 2007. http://www.pcisig.com/developers/main/training_materials/get_document?doc_id=60bc4b488481d00bb6862ad756bfe2f9a29f2c90.

[Tan09] Tandberg. *Tandberg Telepresence T3 and T1 Product Sheet*, October 2009.

[TDG98] R. Thayer, N. Doraswamy, and R. Glenn. *IP Security Document Roadmap*, November 1998. RFC 2411.

[VIT10a] VITA. *FMC - FPGA Mezzanine Cards*, 2010. http://www.vita.com/fmc.html.

[VIT10b] VITA. *VME Technology Specifications*, 2010. http://www.vita.com/specifications.html.

[WA09] Jake Wiltgen and John Ayer. *Bus Master DMA Performance Demonstration Reference Design for the Xilinx Endpoint PCI Express Solutions*. Xilinx, December 2009. XAPP1052 (v2.5).

[WF83] A. X. Widmer and P. A. Franaszek. A DC-balanced, partitioned-block, 8B/10B transmission code. *IBM Journal of Research and Development*, 27(5):440–451, September 1983.

[Xil09a] Xilinx. *Block Memory Generator v3.3*, September 2009. DS512.

[Xil09b] Xilinx. *LogiCORE IP Clocking Wizard v1.4*, December 2009. DS709.

[Xil09c] Xilinx. *ML505/ML506/ML507 Evaluation Platform User Guide*, October 2009. UG347 (v3.1.1).

[Xil09d] Xilinx. *SP605 Hardware User Guide*, November 2009. UG526 (v1.1).

[Xil09e] Xilinx. *Virtex-5 Family Overview*, February 2009. DS100 (v5.0).

[Xil10a] Xilinx. *AR #34009 - Virtex-6 FPGA ML605 Board - PCI Express link will not train; implementations for PCI Express must use the v1.3 Integrated Block Wrapper for PCI Express*, June 2010. http://www.xilinx.com/support/answers/34009.htm.

[Xil10b] Xilinx. *LogiCORE IP Virtex-6 FPGA Integrated Block v1.5 for PCI Express – Product Specification*, April 2010. DS715.

[Xil10c] Xilinx. *ML605 Hardware User Guide*, May 2010. UG534 (v1.3).

[Xil10d] Xilinx. *Spartan-6 Family Overview*, March 2010. DS160 (v1.4).

[Xil10e] Xilinx. *Spartan-6 FPGA SP605 Evaluation Kit*, May 2010. http://www.xilinx.com/products/devkits/EK-S6-SP605-G.htm.

[Xil10f] Xilinx. *Virtex-5 LXT FPGA ML505 Evaluation Platform*, May 2010. http://www.xilinx.com/products/devkits/HW-V5-ML505-UNI-G.htm.

[Xil10g] Xilinx. *Virtex-6 Family Overview*, January 2010. DS150 (v2.2).

[Xil10h] Xilinx. *Virtex-6 FPGA ML605 Evaluation Kit*, May 2010. http://www.xilinx.com/products/devkits/EK-V6-ML605-G.htm.