Journal 1, Analysis of possible uses

1 Objectives

The overall objective is to investigate the possible uses of an FPGA-board in a PC. This is done to lay out the further course of the project, and to constrain it with regard to functionality, operating systems and similar. The investigation is done mainly on project-ideas and projects that are already running at SDU.
The following four possible uses have been found:

- Multipurpose hardware interface
- Network interface card
- Vision
- Collision detection

2 Problem analysis

All of the possible uses have been investigated with regards to the following aspects:

- *Purpose?* What is the system supposed to do?
- *FPGA?* Are there any particular reasons to use an FPGA? Parallelism / flexibility / etc?
- *In/out?* What are the inputs and outputs? Amount of data / datatypes / external interfaces?
- *Performance?* Are there any requirements regarding performance, latency, etc?
- *Existing?* Is there already some hard-/software in existence that can do the job?
- *Environment?* Are special computers used? Windows / Linux?

The aspects that are deemed central to the project will be summarized for each of the possible uses.

2.1 Multipurpose hardware interface

Project: SDU, Luna, Anders S. Sørensen

Purpose: At SDU it could be interesting to have a flexible and generic hardware interface for use with a long list of standard components. Additionally, an FPGA could also be used for implementing certain components in the FPGA itself.
An example could be to implement a PWM signal generator, or to interface A/D and D/A converters and components with SPI or RS232 interfaces.
The idea is to create a substantial library of components and interfaces that are available together with the physical interface-boards, to make it quick and easy to assemble and configure a specific functionality.

FPGA: The obvious advantage of an FPGA is its flexibility. Reconfiguring the FPGA can quickly adapt it to a given component.

In/out: In- and outputs will vary quite a bit, depending on what it is being used for. For signal generators, a few control-messages will be all that needs to be sent between the PC and the FPGA. However, if using a high-speed A/D converter, large amounts of data will need to be transferred. The software interface therefore needs to be flexible enough to support both kinds.
Additionally, it is desirable to be able to attach daugtherboards for the physical interfaces.

Performance: Like with the in/outs, performance requirements will vary depending on the application.

Existing: There are a quite a few VHDL-projects around SDU that have concerned interfacing various components to an FPGA. However, these are almost exclusively adapted for a specific purpose, and are not directly usable in other projects.

Environment: It would be a great advantage if the system would work on both Linux and Windows machines.

The central aspect here is flexibility. Both the software interface and the VHDL-code must be designed with as few "hard" limitations as possible, to make it easy to extend or modify it with new functionality later on. As it will probably be used during classes, it is also of importance that it is easy to use, and well-documented.

2.2 Network interface card

Project: SDU, Luna, Anders S. Sørensen

Purpose: The objective here is to create a network for data-mirroring between a number of computers and/or microcontrollers/similar systems. The idea is that each machine is connected to a network node with a number of registers, which are then automatically mirrored in all the other nodes across the network.
An existing networking protocol might be used for this, but it is not a necessity.

FPGA: The advantage of using an FPGA for this project is that it is possible to implement pretty much any type of network protocol, which can then run parallel with other functionality. Additional advantages are the availability of dual-port blockram and the possibility of creating a specialized pipeline for the network stream.

In/out: It needs to be possible to read and write the registers from the PC. Additionally, a number of status-commands could be nice to have available. Externally there needs to be some sort of network-interface, which could be based on for example twisted-pair, or optical cables.

Performance: Here it is probably the performance towards the network itself that is of interest, and not the performance of the software interface. Requirements as to the speed of the system have not been specified yet.

Existing: There are several kinds of networking protocols in existence, but probably none that do exactly as desired. It might however be possible to build upon and extend an existing protocol.

Environment: Again it would be advantageous if the system could work on both Windows and Linux machines. Additionally, an FPGA-part should be designed so that it can also be used with other systems than PCs, like microcontrollers for instance.

The central aspect here is the FPGA design, which needs to be generally usable together with other interfaces than the PCI-Express used in this project.

2.3 Vision

Project: SDU, Vision lab, Anders Kjær-Nielsen

Purpose: The objective is to use the FPGA-board as a coprocessor for offline (and eventually online) image processing. Currently, the vision lab employs FPGA-based systems for online image processing, where a camera is connected directly to the FPGA through a FireWire interface. The FPGA processes the image stream, and sends only the results to the connected PC.
However, there is a wish for an offline solution, where images from the PC are sent to the FPGA to be processed, and the results are sent back.

FPGA: The operations normally used in image processing are typically parallelizable, which is exploitable when using FPGAs. Also, it is possible to pipeline several operations.

In/out: It needs to be possible to transmit an image between the PC and the FPGA. A typical image can be (worst-case) 4 mega-pixels with a 24 bit color-depth – about 12 MB per

image. In practice though, a smaller cut-out of the complete image is usually used – typically 512x512 pixels, which is about 1 MB per image. The output from the processing varies, and can be either several new images, or just a few coordinates.
It could also be desirable to be able to attach a FireWire camera to the FPGA-board so that it can be used for online processing as well.

Performance: Typically, a framerate of 15 frames per second is used. This amounts to 180 MB/s using complete images (4 mega-pixels), which is considered worst-case. Using the smaller cut-outs, the data-stream settles at around 10-20 MB/s though. As some algorithms can return several images, the resulting return stream can be several times greater than the input stream.

Existing: FPGA-implementations of several image processing algorithms are in existence.

Environment: Again, it would be an advantage to be able to use both Windows and Linux machines.

The central aspect is here that the interface needs to be able to transfer large amounts of data very fast.

2.4 Collision detection

Project: SDU, Robot, Jimmy Jørgensen

Purpose: The objective is to accelerate collision detection for use with offline path-planning for robot movements. An initial rough estimate is performed, which is then refined until the necessary precision is reached.

FPGA: The collision detection algorithm is parallelizable, and makes it possible to exploit the parallel structure of an FPGA. Also, it is desirable to unload work off of the PCs CPU, so it can be used for other tasks in the meantime.

In/out: Initially, information regarding the objects to perform the tests on needs to be downloaded to the FPGA. This will typically amount to 2-3 MB, but might reach 50-100 MB worst-case. After this, only a few kB of commands, coordinates and status will need to be transferred.
No external interfaces are needed, although it could be interesting to have an opportunity to stack two or more FPGAs together, for increased performance.

Performance: There are no specific requirements as to the performance, except that it just needs to perform as fast as possible. A feature allowing queuing of commands on the FPGA could also be useful.

Existing: An implementation of collision detection in an FPGA is currently being worked on and partly finished. This uses the RS232 port for communicating with a PC, which is not optimal when needing to transfer large amounts of data. A faster and more stable interface would be preferable.

Environment: The existing framework for working with robot movement planning and inverse kinematics, RobWork, is multi-platform, so it is desirable that an FPGA-system for use with this would work on both Windows and Linux too.

The central aspect of this is the transfer and storage of relatively large amounts of data. The speed at which this happens is not as critical as with the image processing though, as this only happens during startup.

2.5 Other uses

Looking around on the internet, two additional cases have been discovered. These have not been investigated as thoroughly as the previous cases, but are included out of interest.

- *Terma[1], Radar Systems:* A short telephone interview with Allan Skouboe Jensen from Terma Radar Systems has been conducted. Terma are using several powerful Virtex FPGAs for processing images from radars – radar data comes in, and a video stream comes out. Up until

[1] More info: http://www.terma.dk

now, these FPGAs have been connected together through a specialized cabling system. Terma are now in the process of moving this system to a PCI Express platform instead. This will consist of a PCI Express x8 system with four or five FPGA boards, communicating trough the PCI Express interface.

- *Cray Inc. Supercomputers*[1]*:* Cray Inc. offers a series of so-called "hybrid" supercomputers called XT5h. These are configurable with both standard AMD Opteron CPUs and Virtex4 FPGAs. The FPGAs are connected to the system directly through the HyperTransport[2] bus used by the Opteron CPUs. This ensures low latency and high bandwidth, as the system actually sees and treats the FPGA just like a CPU, instead of as a peripheral device.

3 Conclusion

Four cases have been investigated. These concern using the FPGA board as a multipurpose hardware interface, as a network interface, and using it for processing vision algorithms and collision detection. From these it can be seen that there are some rather varying requirements for the system, depending on the application. The following can be summed up:

- The software interface (between FPGA and applications on the host PC) and the hardware interface (inside the FPGA) have to be flexible, and need to be designed to allow for easy add-ons and modifications at a later time.
- The interface needs to be well-documented and easy to use.
- The system must be able to handle large amounts of data being transferred each way at high speeds. The target is about 180 MB/s each way.
- The system needs to be able to store 100 MB of data.
- The system needs to work on both Windows and Linux machines.

[1] More info: http://www.cray.com/products/xt5/xt5h.html
[2] More info: http://www.hypertransport.org/ and http://en.wikipedia.org/wiki/Hypertransport