Implementation of FPGA based star tracker pre processing pipeline

Oscar Björkgren

Supervisors, Åbo Akademi: Annamari Soini, Johan Lilius
Supervisors, Aboa Space Research Oy: Tero Säntti, Tuomo Komulainen
Master's thesis in Computer Engineering
Åbo Akademi
2021

Abstract

Keywords

Abbreviations

Contents

1 Introduction **1**

2 System requirements **2**

2.1 Specification 3

2.1.1 System components 3

3 Custom hardware design **5**

3.1 Single and general purpose processors 5

3.2 FPGA Overview 6

3.3 System modeling with HDL 9

3.4 System modeling with IP cores 10

4 Development process **11**

5 Documentation **12**

5.0.1 Box Filter 12

5.0.2 Connected component labeling and analysis 12

6 Review and reflection **13**

7 Discussion **14**

Bibliography **14**

List of Figures

2.1 Flowchart of StreakDet software [1] 2

3.1 Graph showing increasing popularity of specialized hardware in high performance computing [2] 6
3.2 FPGA architecture. An FPGA consists of configurable logic blocks, wires and input/ output pads [3]. 7
3.3 Diagram of a configurable logic block. A CLB consists of logic element containers called slices, inputs and outputs, and a switch matrix which connects the CLB to other CLBs [4]. 8
3.4 Look-up table with 4 memory cells and two multiplexers. The FPGA design initializes memory cells **a-b** and uses inputs **x1** and **x0** to select value **y** [3] 8
3.5 DSP system design on FPGA [3] 9

List of Tables

2.1 Table of functional and non-functional system requirements 3

1. Introduction

A hardware based solution for a software algorithm is sometimes needed within application areas where performance matters the most. One example of this is the aerospace industry where the most optimized solution is sometimes the only choice. In this thesis I work together with Aboa Space Research Oy to create a hardware implementation of a pre-processor module for a star tracker system using a software component as a basis. A star tracker is commonly used as part of a larger system where the role of the tracker is to provide orientational data to other components. This information might be used for navigation and control or in combination with scientific instruments to help with further analysis. Hardware design can be inspired by software algorithms when designed for a specific purpose as in this case. The software defines the functional part of the system, which leaves a big part of the supporting architecture to be designed. In this thesis I will explain the process of implementing functionality to a digital circuit, by example.

To complete a set of tasks in any environment, a scientific project needs a well defined tools to reach its goals. In technologically restricted environments such as space, tools often needs to be multi purpose for optimal use. An image sensor is in this case used for providing reference data for a star tracker. This image sensor can provide data to multiple systems by reuse of data or capturing new images with other settings. The purpose of the system in this project is to refine data from pre defined sensors to provide additional and reinforcing information about the environment for the use of other scientific instruments. The goal of this thesis is to create a roadmap of the system design process and include theory of relevant areas.

2. System requirements

A hardware implementation of an image analysis software component is to be developed to be used on a FPGA. The software in question is a product of a European space agency funded research project regarding detection and analysis of space debris in low earth orbit. Using computer vision among other analysis, space debris is identified from image data. With the help of a star catalog the image is mapped to a position in space to provide positional data of the detected debris. A diagram of the system is shown in figure 2.1. The software component relevant to this project is the Pre-processing & Segmentation part of the diagram, which is in the input phase of the complete system [1] [5].

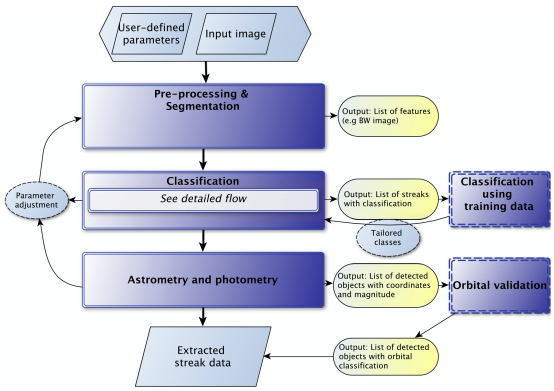

Figure 2.1: Flowchart of StreakDet software [1]

The star extraction process in StreakDet passes the input image trough a series of algorithms to identify stars. The algorithms are well known image filters and analysis methods that in combination with each other produces a way to pick out objects from an image. The main task in this project is to identify these algorithms and fit them into a hardware design using a hardware description language.

Star extraction is a key stage of a star tracker system which uses a star catalog to match the extracted stars against. Using the star matches and catalog information the position and orientation of the captured image can be determined.

The goal of this project is to create a pre-processing pipeline to be used in a star tracker.

2.1 Specification

The specification consists of development and system requirements as well as description of tools and components. System requirements are created based on meetings and discussions with Aboa Space Research Oy. Some requirements are derived from higher level requirements to suit the development process. Table 2.1 presents an overview of the system in a functional and non-functional requirements fashion.

Req. Id	Functional requirement	Non-functional requirement
1	The FPGA is used as processing node for extracting stars from an optical image	Convert StreakDet star extraction software algorithm to a hardware design
2	A communication channel is available for shared memory usage	FPGA implements memory controller interface and USB port controller
3	Allow configuration of processing pipeline parameters	Frontend software is used for pipeline configuration
4	Processed images and intermediate results can be verified	Pipeline stage results are written to storage
5	Performance metrics of process and stages can be viewed	Pipeline captures and stores metrics along the stages

Table 2.1: Table of functional and non-functional system requirements

2.1.1 System components

Processing nodes

The system consists of two nodes, the host machine and target FPGA. Because the application of this system different in a production environment, the FPGA part should not be too depending on the host machine. When integrated into a larger system with shared memories and communication, the FPGA will have to be able to read input image directly from memory. The same applies for the

output, the register where the output is written needs to be configurable and not sent to a host machine by default.

FPGA module

The FPGA module Opal Kelly XEM7305 chip with a Xilinx Spartan-7 FPGA will function as a testbed for development. The module contains a part from the FPGA external SDRAM memory and communication interfaces such as USB 3. These features allows for host machine communication and data storage of features such as images, star catalog and output. These data could be stored on the FPGA itself but using the SDRAM is a sustainable solution because it allows for shared memory usage in production and more flexible image and star catalog sizes.

3. Custom hardware design

3.1 Single and general purpose processors

A general purpose processor refers to a hardware computing platform which is designed for universal use with broad benefits across different problem solving domains. Computing platforms such as microprocessors and CPUs are examples of this type of processor, their hardware logic is implemented in such a way that it enables a large variety of computations to be performed. The characteristics of a processor for general problem solving are well suited to common tasks where there might be many hard- and software abstractions between the application interface and hardware logic. This is a necessity in for example PCs.

When hardware is required for a single purpose with a finite set of tasks to be performed there will be drawbacks with using hardware designed with flexibility in mind. This could be compared to the use of a multitool for driving a screw into a piece of wood when in fact only a screwdriver is needed. It is not the wrong way to do it but there is a more optimized way of achieving the result. Any CPU could be used for computer graphics calculations but since GPUs are designed for the single purpose of this type of processing it is a better tool for the job.

Single purpose processors have a rich history with roots in the early ages of computers. Vector processors were for example used in supercomputers. This was before computers were seen as general purpose technology and high performance computing was about as common as ordinary PCs. Since recent years the utilization of specialized hardware has been rising, with new application areas evolving. Single purpose processors such as GPUs and FPGAs have found themselves into areas such as machine learning, cryptocurrency mining, and other high performance computing applications. As the hardware used in aerospace applications generally is specialized due to strict requirements, single purpose computing platforms are heavily used. For example, the NASA perseverance Mars rover

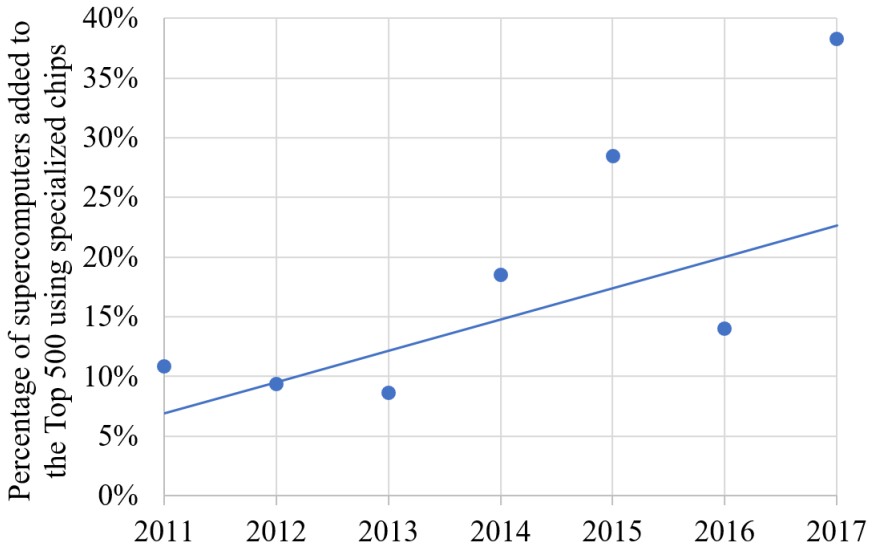

Figure 3.1: Graph showing increasing popularity of specialized hardware in high performance computing [2]

relies heavily on Xilinx manufactured FPGAs for different tasks, such as image processing pipelines [6].

3.2 FPGA Overview

A field programmable gate array, or FPGA, is a digital integrated circuit that consists of millions of logic blocks that can be configured to perform different operations. The process of configuring the logic blocks is called FPGA design and can be compared to the design of integrated circuits. Reconfigurability and ease of design are main advantages when using an FPGA as computing platform. When comparing an FPGA design to a solution that is developed using software, the FPGA stands out with process pipelining and high level of parallel processing. The difference between executing a software program on a processor and executing the equivalent operation on an FPGA is that the compiled software instructs the processor to do operations in a generic way, as in loading a register and shifting a bit. An FPGA does not need to explicitly load a register because the registers are by default connected to the input of the operation, which is defined by the FPGA design. A bit shift operation is typically an inexpensive operation regardless of computing platform, but the FPGA can pipeline the result directly into another register or operation which saves clock cycles [3].

Xilinx is one of the key companies involved in FPGAs both currently and historically. They introduced the world to FPGAs in 1985 and has since then led the programmable logic technology industry with a 51% market share of programmable logic device suppliers in 2017. Major end market categories for Xilinx are aerospace, defense and communications. The main competitor of Xilinx is Intel with a market share of 37% [7][8].

The basic internal elements of an FPGA consists of circuits such as look-up tables, flip-flops, wires and input/ output pads. Using these elements the FPGA design is implemented in the hardware. The architecture is explained by figure 3.2 which shows a matrix arrangement of configurable logic blocks which are connected to each other with wires running between them. The wires also run to the input/ output pads which enables interfacing with off-chip components such as SDRAM or sensors [3]. Configurable logic blocks contains the look-up tables and flip flops. An overview of a CLB is shown in figure 3.3.

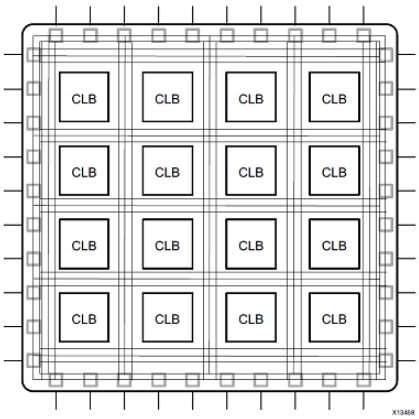

Figure 3.2: FPGA architecture. An FPGA consists of configurable logic blocks, wires and input/ output pads [3].

The slices in figure 3.3 represents the core element of any FPGA. Slices consist of look-up tables, storage elements, multiplexers and carry logic. Depending on the FPGA there are different number of slices per CLB. The slices are collectively responsible for implementing a significant portion of FPGA design [4].

Look-up tables or LUTs, are mainly used as logic elements in FPGAs. They can implement any boolean logic operation by combining memory cells with multiplexers. The look-up table is essentially a truth table that can be used for reading the result of boolean operations. Memory cells and multiplexer routings are initialized by the FPGA design, which is what makes the look-up table el-

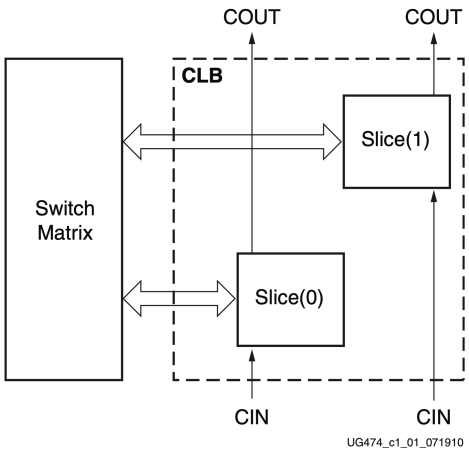

Figure 3.3: Diagram of a configurable logic block. A CLB consists of logic element containers called slices, inputs and outputs, and a switch matrix which connects the CLB to other CLBs [4].

ement configurable [3]. An example of a 4 memory cell wide LUT is shown in figure 3.4.

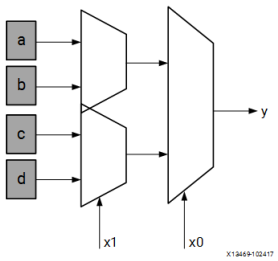

Figure 3.4: Look-up table with 4 memory cells and two multiplexers. The FPGA design initializes memory cells **a-b** and uses inputs **x1** and **x0** to select value **y** [3]

The circuit elements discussed in this chapter forms the basic blocks of FPGA technology. A larger set of elements are often included in an FPGA to enable some optimizations and further functionality in the design. For example there are different types of slices a CLB can contain which makes it more suitable for storage purposes rather than logic. Signal processing pipelines are a popular use case for FPGAs and they often contain slices specialized for operations in this problem domain. Digital signal processing can be accelerated by parallelizing operations in a SIMD fashion, thus to maximize the advantage there can often be found a large number of these slices. Figure 3.5 shows an example of a FPGA configuration with some of these specialized blocks mapped out. The green CLBs

are designated to memory usage and are closely coupled to the red DSP blocks to enable fast processing of the stored data. The purple external memory controllers uses the input/ output pads to interface with external memory such as SDRAM [3] [4].

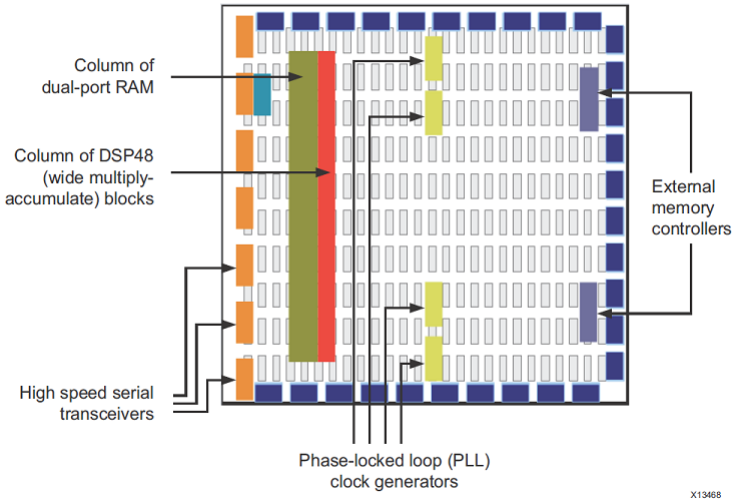

Figure 3.5: DSP system design on FPGA [3]

When comparing FPGAs among each other the main performance metrics are power consumption and number of logic cells. The number of logic cells describes size and complexity of the design that the FPGA can fit. Additionally the FPGAs can often be compared by specifications such as specialized CLB slices, IO interfaces and communication rates. Clock speed is seldom compared because although there is an upper limit, it is often design specific [4] [3].

3.3 System modeling with HDL

The main tool for creating a FPGA design is a hardware description language. VHDL and Verilog are examples of commonly used HDLs. Compared to conventional computer programming languages, HDLs are used in a data flow fashion to describe the timing and logic of an integrated circuit. The data flow description of a circuit in a HDL is such that all code blocks are executed simultaneously by default. This is one of the main advantages of FPGAs as the system is inherently parallelized. It creates also one of the biggest challenges in HDL based design as it complicates the timing of events and resource concurrency.

The output of a HDL design is produced by a build process that computes the high level data flow description into low level circuit elements and routing. This can be compared to the compilation phase of code written in a statically typed software programming language.

The synthesis process is where the HDL design known as register transfer level code is transformed to a gate level description called netlist. The netlist is a mapping of circuit components that are found inside the FPGA. The convenience of HDL based design is being able to describe the circuit at a higher level. In practice this means that registers and buses are created and connected to be inferred into low level circuitry at the synthesis phase. When developing a design against a specific FPGA, a bitstream file is created to configure the device with. The end product of a HDL based design is the bitstream file, or alternatively the synthesis result as it can be used for simulation purposes by a test bench [9].

3.4 System modeling with IP cores

Designing a hardware system often involves components of reusable nature, such as dividers, communication protocols and memory controllers. Intellectual property cores are distributions of components such as these. An IP core can be of varying complexity as it can consist of a solution for a specific task or describe a complete SoC. They are licensed trough a patented design which is what the name refers to. IP cores can often be configured to suit different properties of the implementation such as data width and clock rate. The core is then connected to the system using a HDL. The core it self may consist of software or hardware macros. The software macro is a HDL implementation of the IP core logic that is built to the specific FPGA with rest of the system design. A hardware macro IP core is pre built and FPGA specific [10] [9].

4. Development process

5. Documentation

5.0.1 Box Filter

5.0.2 Connected component labeling and analysis

6. Review and reflection

7. Discussion

Bibliography

[1] Jenni Virtanen, Jonne Poikonen, Tero Säntti, Tuomo Komulainen, Johanna Torppa, Mikael Granvik, Karri Muinonen, Hanna Pentikäinen, Julia Martikainen, Jyri Näränen, Jussi Lehti, and Tim Flohrer. Streak detection and analysis pipeline for space-debris optical images. *Advances in Space Research*, 57:1607–1623, 2016.

[2] Neil C. Thompson and Svenja Spanuth. The Decline of Computers as a General Purpose Technology. Visited 20.1.2021, 2018.

[3] Xilinx Inc. *SDAccel Environment Profiling and Optimization Guide*, 2018.

[4] Xilinx Inc. *7 Series FPGAs Configurable Logic Block*, 2016.

[5] Aboa Space Research Oy. Starmatch, 2020.

[6] Xilinx Employee. Touchdown! NASA's Perseverance Rover Lands on Mars with Xilinx FPGAs On Board. Visited 19.4.2021, 2021.

[7] Wim Roelandts. 15 Years of Innovation. *XCell*, 32:2, 1999.

[8] Investor Overview. https://investor.xilinx.com/static-files/2958145c-e3a2-456f-a461-bc9cba375af3. Visited 19.4.2021.

[9] Xilinx Inc. *Vivado Design Suite User Guide*, 2021.

[10] DJ Holding. *Electrical Engineer's Reference Book (Sixteenth Edition)*. Aston University, 2003.