

Veikko Kero

# Preliminary Development Research of a Smart Machine Vision System for Lumber Industry

Helsinki Metropolia University of Applied Sciences

Bachelor in Engineering

Electrical Engineering

Bachelor's Thesis

18 November 2015

Author(s) Title	Veikko Kero Preliminary Development Research of a Smart Machine Vision System for Lumber Industry
Number of Pages Date	33 pages 18 November 2015
Degree	Bachelor in Engineering
Degree Programme	Electrical Engineering
Specialisation option	Electronics and Medical Engineering
Instructor(s)	Markus Hyvönen, Product Development Manager Janne Mäntykoski, Senior Lecturer
<p>The purpose of this thesis is to describe the first phase of product development of a new smart camera system for machine vision applications in lumber industry. This project was carried out for Inx-Service, a Finnish engineering company specialized in measurement systems and machine vision systems in lumber industry.</p> <p>A machine vision camera is a device that is used in various industrial imaging applications. The thesis focuses on some of the utilizable techniques and comparison of those. The goal was to present an overview of vital information to understand the operation of the final product.</p> <p>Also a prototype of a simple development system is presented. It enables system design with low-cost equipment before the final components are purchased. Finally, some approaches for the future development are discussed.</p> <p>The study consists of research on available materials and planning based on available literature. A prototype of the development system and some aspects of its operation are presented here as well.</p> <p>The product is planned to be ready for manufacturing by the end of 2016 and is targeted to previous customers as an upgrade. Modular design allows usage also in new application areas different than of lumber industry.</p>	
Keywords	machine vision, lumber industry, camera, automation, FPGA, ARM, Linux

Tekijä(t) Otsikko	Veikko Kero Älykkään konenäkökameran esiselvitystyö metsäteollisuuden tarpeisiin
Sivumäärä Aika	33 sivua 18.11.2015
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Sähkötekniikka
Suuntautumisvaihtoehto	Elektroniikka ja terveydenhuollon tekniikka
Ohjaaja(t)	Markus Hyvönen, Tuotekehityspäällikkö Janne Mäntykoski, Lehtori
<p>Tämän insinööriyön tarkoitus on esitellä mekaanisen metsäteollisuuden tarpeisiin suunnatun älykkään konenäkökameran esiselvitystyö. Esiselvitystyö on tuotekehityksen ensimmäinen vaihe.</p> <p>Konenäkökamera käytetään sahoilla prosessien tarkkailemiseen ja ohjaamiseen erilaisissa konenäkösovelluksissa. Tavoitteena on esitellä tarvittavia työkaluja ja teknologioita kameran toiminnan ymmärtämiseksi. Työ esittelee myös yksinkertaisen testijärjestelmän, jolla voidaan simuloida erään kamerasensorin toimintaa.</p> <p>Lopuksi käydään läpi muutamia sovelluskohteita ja erilaisia lähestymistapoja kameran suunnitteluun.</p> <p>Työ koostuu tarjolla olevan materiaalin tutkimisesta ja aloitetun tuotekehityksen suunnittelusta. Testijärjestelmän osalta esitellään sen osia, toimintaa ja karkea prototyyppi.</p> <p>Insinööriyö tehtiin Inx-Servicelle, joka on kotimainen mekaaniseen metsäteollisuuteen erikoistunut laitevalmistaja. Tuotteen on määrä olla valmis tuotantoon vuoden 2016 loppupuolella ja sen kohderyhmänä on lähtökohtaisesti vanhat asiakkaat. Uudelle kameralle on sovelluskohteita myös metsäteollisuuden ulkopuolella.</p>	
Avainsanat	konenäkö, metsäteollisuus, automaatio, kamera, FPGA, ARM, Linux

## Contents

1	Introduction	1
2	Machine Vision	2
2.1	Definition	2
2.2	Machine Vision in Lumber Industry	3
2.2.1	Breakdown	3
2.2.2	Grading	4
2.3	Image Acquisition	5
2.3.1	Image Sensor	5
2.3.2	Optics	7
2.3.3	Exposure	8
2.3.4	Illumination	9
2.4	Programmable Logic	10
2.4.1	Field Programmable Gate Array	10
2.4.2	System on Chip	11
2.5	Machine Learning	11
2.6	Digital Image Processing	13
2.6.1	Segmentation	14
2.6.2	Thresholding	15
2.6.3	Masking	15
2.6.4	Histogram	15
2.6.5	Data management	16
2.7	Data Transfer and Interfaces	17
2.7.1	LVDS	17
2.7.2	Machine Vision Interface	17
3	Making It Smart	19
4	The Development System	21
4.1	EBV SoCrates II	21
4.2	Altera Quartus II 15.0	23
4.3	Embedded Linux	24
4.4	Booting the System	25
4.5	Generating a Test Image	27
4.6	Reading the Test Image	28

4.7	Image Processing	29
5	Discussion and applications	31
5.1	Proposed Structure	31
5.2	OpenCL	32
6	Conclusions	34
	References	35

## Abbreviations

ARM	Advanced RISC Machines (orig. Acorn RISC Machine)
ASCII	American Standard Code for Information Interchange
ASIC	Application Specific Integrated Circuit
CCD	Charge Coupled Device
CMOS	Complementary Metal-Oxide-Semiconductor
CPLD	Complex Programmable Logic Device
FPGA	Field Programmable Gate Array
GIMP	GNU Image Manipulation Program
GNU	"GNU's Not Unix!"
GPL	General Public License
GSRD	Golden System Reference Design
HDL	Hardware Description Language
AXI	Advanced eXtensible Interface
LVDS	Low Voltage Differential Signal
AD	Analog-to-Digital
RGB	Red Green Blue
QSPI	Quad Serial Peripheral Interface
MMC	Multimedia Card
HPS	Hard Processor System
IC	Integrated Circuit
JTAG	Joint Test Action Group
LE	Logic Element
RTL	Register Transfer Level
SD	Secure Digital
SoC	System on Chip
SOM	Self Organizing Map
VHDL	VHSIC Hardware Description Language
TFT	Thin-Film Transistor

## 1 Introduction

A saw mill can lose millions of euros if material to be handled is not optimized properly. In Finland, the lumber is usually sawn by volumes to get the most out of raw material. Problem is that volume does not automatically translate to profit in lumber industry where different grade categories of the final product might have significantly different value (especially in pine wood). Therefore optimization is essential.

Inx-Service is specialized in these kind of measurement and optimization systems and has been producing them for nearly two decades. The applications vary, but usually the goal is the same: visual optimization.

Camera systems used in company's earlier products, like award-winning OptiGrader, are over ten years old. Although still functioning excellently, an update to today's technology is advantageous.

This thesis is a first step in the product development process. It introduces some parts of preliminary research, like studying of available technologies, high level system design and simple development system to test image sensor operation that can be used for considering equipment and target applications.

The approach is to implement a product for several processes in the lumber industry and not for one specific application. The processes considered here are *first breakdown* and *grading*.

At the end of this study some design flow approaches and target applications are discussed. Also suggestions for the final product are given.

## 2 Machine Vision

This chapter aims to present somewhat coherent overview on the topics essential to operation of the camera and aspects of machine vision. Reader should be advised that the surface on each topic is merely scratched.

### 2.1 Definition

As the name suggests machine vision is an artificial sense that a machine can use to interface with real world through some kind of an imaging device.

Machine vision is used widely in industrial applications to control production processes from visual product inspection to high speed texture analysis. An example of a machine vision application can be seen in the Figure 1.

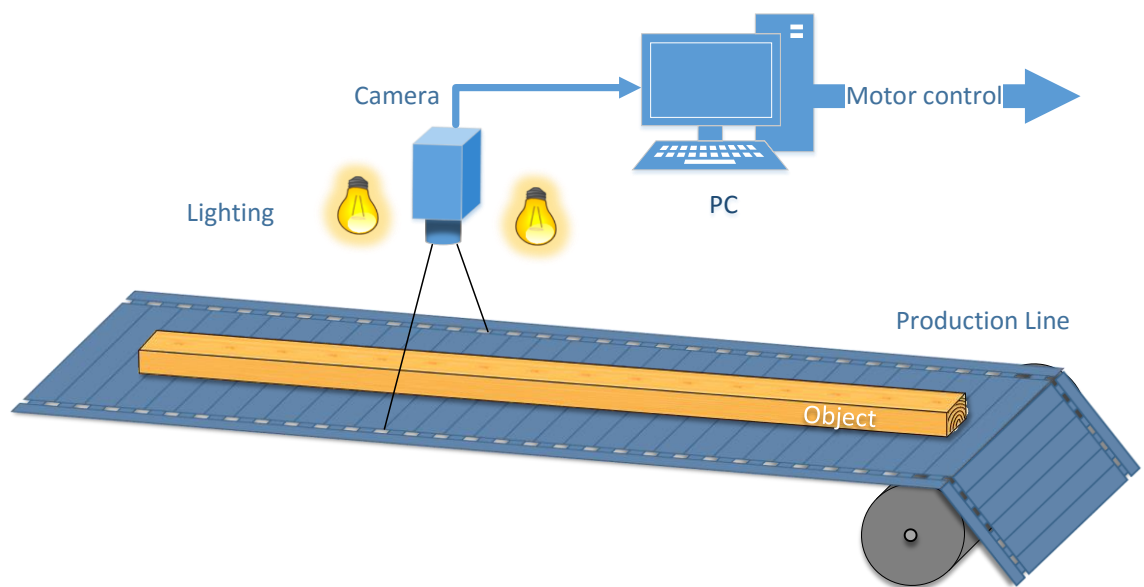


Figure 1. An example of a machine vision application.

Machine vision is a complex discipline involving areas of optics, lighting, electronics, industrial automation and information technology, which makes it ultimately exciting field of study.

In this thesis the focus is on systems in Nordic lumber industry.



## 2.2 Machine Vision in Lumber Industry

Since the late 70's machine vision has brought notable development to the lumber industry [1]. It presents a flexible and precise tool for various processes in all manufacturing stages. First breakdown and grading are discussed in this thesis.

Here, the term *sawn timber* refers to timber sawn from a log. *Board* is used to define sawn goods.

### 2.2.1 Breakdown

When a log enters sawmill the first phase of the sawing process is called a *breakdown*, where the log is sawn into pieces. Figure 2 shows a typical softwood breakdown used in Nordic countries.

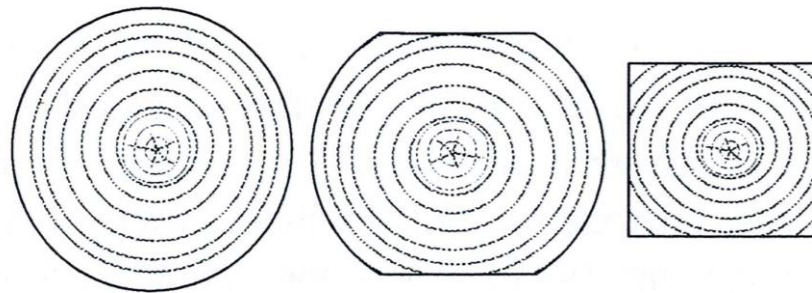


Figure 2. Raw material sawn in breakdown process are logs, cants and square cants of various sizes and lengths. Reprinted from Vuorilehto (2001) [2].

Cants can be used as is or sawn further to *flitches*<sup>1</sup> of various sizes (second or third breakdown).

To determine where to cut the log a machine vision system can be used to measure the profile of the log. The goal is to optimize saw lines, maximize yield and collect data from breakdown process.

---

<sup>1</sup> A rough piece of sawn wood

## 2.2.2 Grading

After the breakdown resulting pieces are sorted and graded.

In grading the surface of a board is inspected to determine grade of the product. Traditionally grading is done by human inspector. Typical defects sought from sawn timber are knots, cracks and colour defects. Timber is then divided to different grade categories (Figure 3) based on the amount and type of defects found.



Figure 3. Three grade categories of 75mm x 200mm pine boards. Modified from Nordic grading instructions book [3].

A is the highest grade and it is divided further into four subclasses. Definitions of these grades are explained in detail in the “Nordic Timber Grading Rules” published by The Associations of Sawmillmen in Nordic Countries [3]. There is also lowest grade D that is not mentioned in the instructions, because it is not defined numerically like the others.

A machine can follow these kind of complex grading rules tirelessly, but might have some difficulties adjusting to variations in material. Human inspector seems to compensate easily for changes in grading conditions, such as effects of lighting or variations in material to be graded, but rarely achieves performance better than 70% when grading the same material [4]. After all, visual inspection is always a highly subjective task.

Although possibly more efficient, developing these kind machine vision systems introduces its own set of challenges, like image acquisition, image processing and computing. To improve flexibility of a machine inspector, sophisticated algorithms needs to be applied, which makes development process more complicated.

## 2.3 Image Acquisition

Image acquisition means acquiring the image. Here the term is used broadly to define the process of obtaining the digital image. It involves both hardware and software components. This chapter deals with the hardware.

### 2.3.1 Image Sensor

The most important component in digital imaging is the image sensor (Figure 4). Its operation is based on a phenomenon called photoelectric effect, more specifically the absorption of a photon.

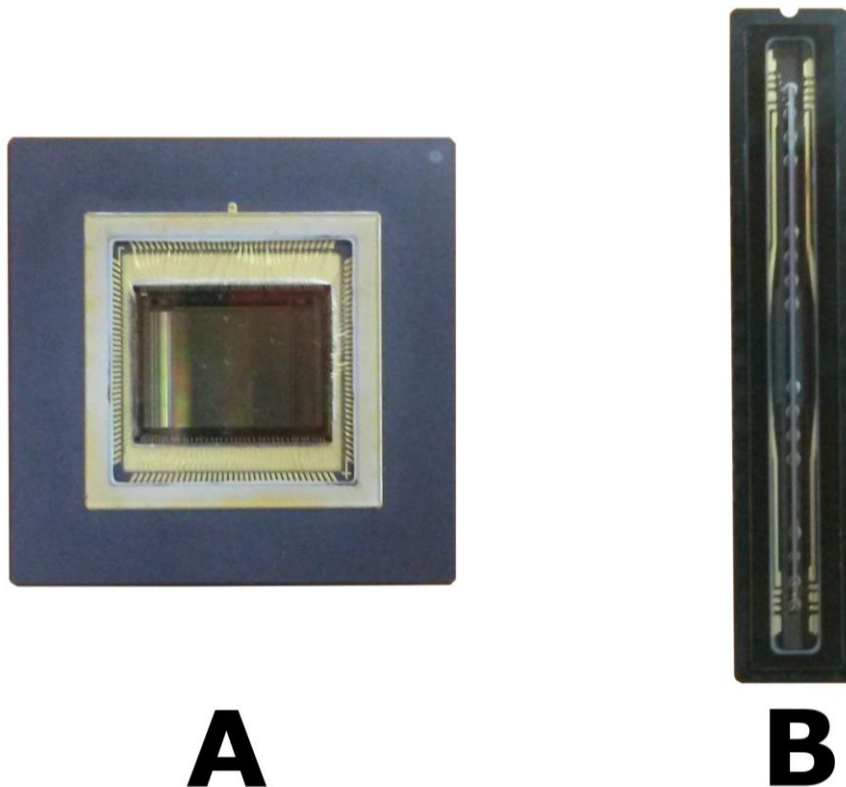


Figure 4. A) 1024x1024 CMOS area-scan (matrix) sensor. B) 5150 pixel CCD line-scan sensor.

In industrial applications both matrix cameras (with sensor A) and line-scan cameras (with sensor B) are used. Matrix camera captures larger area where as a line-scan camera captures only one line at a time. Matrix camera can also be operated in line-scan mode

Image sensors are constructed of photodiodes or so called photon wells that collect the incoming photons. Usually micro lenses or color filters are laid on top of the diodes to control the direction of photons. The negative charge (electron) from a photon is converted to voltage.

Currently CCD and CMOS technologies are prevalent in variety of computer based imaging applications.

In CCD-sensors charge-to-voltage conversion is done pixel by pixel. The charge from a photon is transported along the column down to the serial shift register that delivers them to an integration capacitor, then to an amplifier and eventually AD-converter outside the sensor.

CCD-sensors are commonly described through a familiar hydraulic analogy (Figure 5). Every photon well in a sensor acts like a water bucket that collects water trickling down. The buckets are then conveyed to a bigger bucket at the end of a bigger line.

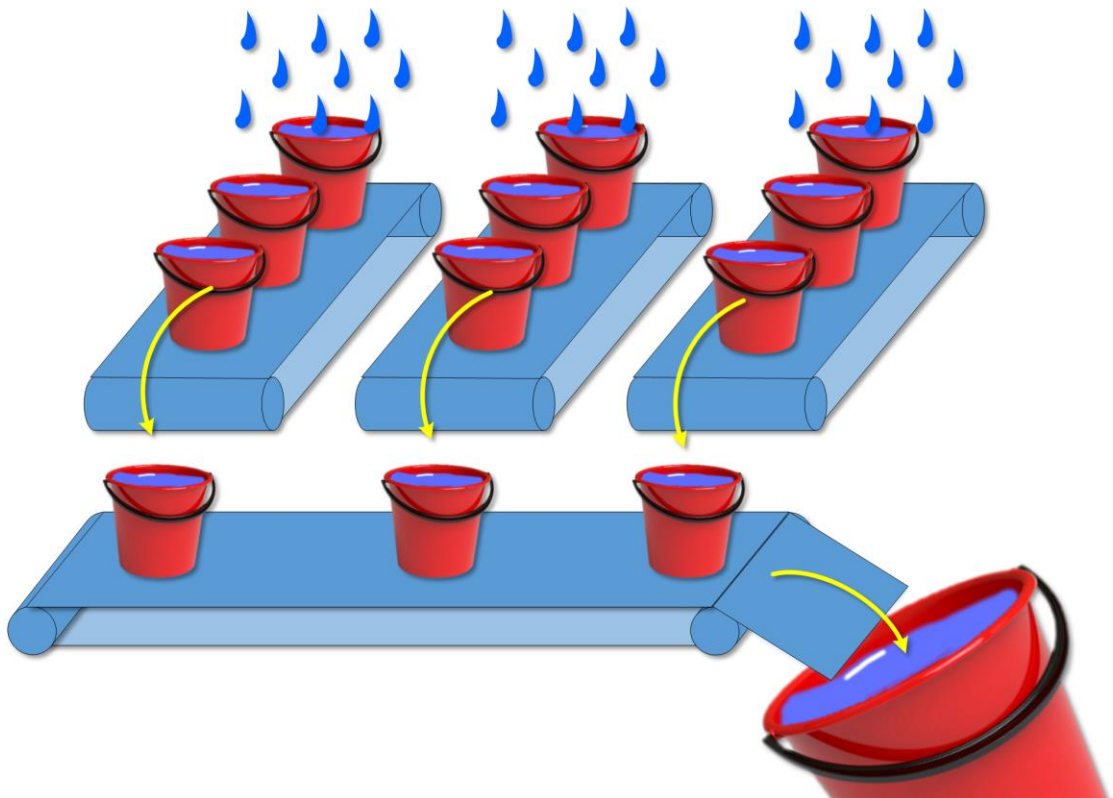


Figure 5. CCD Hydraulic Analogy

CCD-sensor image quality is generally really good and the somewhat perfect global shutter feature (see chapter 2.3.3) is usually favored. However, their operation is rather slow because of their serial nature. They also tend to consume more power.

This good old analogy is partially true also with CMOS-sensors, which are constructed a bit differently from CCDs. The charge-to-voltage conversion is done inside the pixel and sometimes even the AD-converter is there. The operation is more parallel than CCDs. The amount of amplifiers generates more noise, but the speed, low power consumption and AD-conversion inside the sensor are serious advantages.

Most of the mobile devices nowadays use CMOS sensors because of their versatility. CCD-sensor have been dominating the market earlier, but the mobile device industry and more sophisticated manufacturing methods have been boosting CMOS market share over the years. [5]

### 2.3.2 Optics

Optics are found in machine vision systems in various forms, like directional mirrors, lasers, micro lenses and prisms. The goal is always the same: to direct the photons, i.e. to have optimal picture quality for the task at hand.

An object lens is a device that controls the amount of photons passed to the lens by changing aperture and focus. Aperture is the hole through which the light travels. Different apertures can be seen in the Figure 6.



Figure 6. Standard 50mm lens with apertures f1.8, f4 and f22, respectively.

In machine vision systems a small aperture (large f-number) is usually desired for the maximum depth of field and the width of the lens (i.e. angle of view) depends on the size of the object and its distance from camera.

Color filters in image sensor are essentially optical devices. Their operation is based on dispersion of light, where the light spreads into spectrum. The goal is to aim different colors to different parts of the sensor.

Besides the equipment, the imaging conditions affect the picture quality tremendously. For example, when a log from a floating tank is brought to a measuring point, the camera might have difficulties observing the shiny wet surface of the log because the surface responds to light differently than dry surface.

### 2.3.3 Exposure

*Exposure* is a process, where photons are passed to the sensor or film over some period of time and it is controlled with a device called *shutter*. Longer exposure time equals more photons on the sensor, i.e. brighter picture.

The image starts losing information as more light gets (or doesn't get) to the sensor, so again, the shutter speed is also a matter of optimizing.

Traditional shutters block the incoming photons by placing a diaphragm or a leaf over the aperture. Digital image sensors use electronic shutter that is notably faster than its mechanical counterpart. In electronic shutter, exposure is done by moving charges from pixel cells, which is equivalent for blocking the aperture.

The two common types of electronic shutter operation are rolling shutter and global shutter. A rolling shutter exposes image line-by-line. This introduces a distortion problem to moving objects, because the lines are exposed at different times. The distortion seen in Figure 7 is called wobbling, skew or snaking.



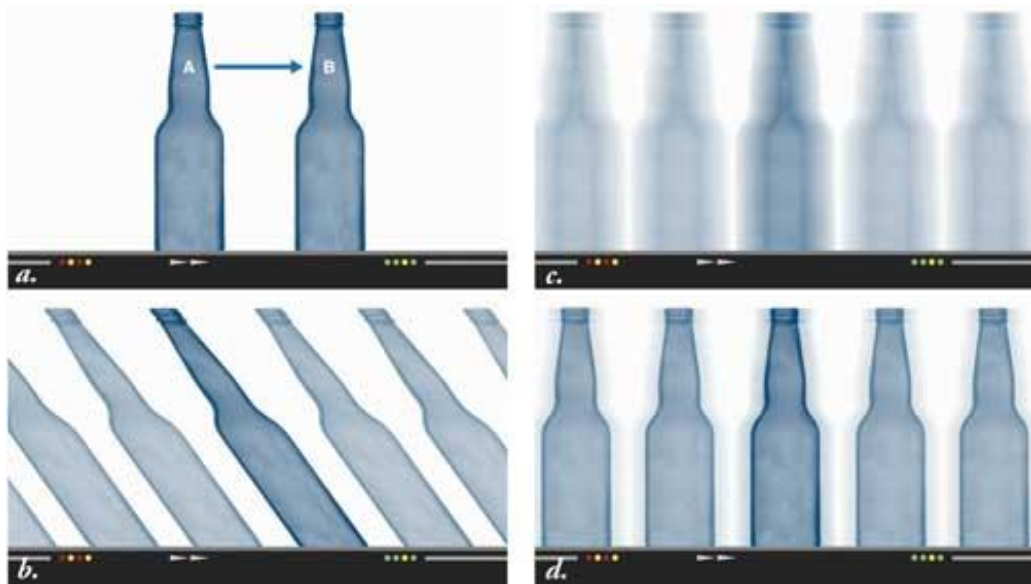


Figure 7. Image sensor capturing a high speed process. a) motion blur (no shutter or exposure period too long), b) Rolling shutter, c) Inefficient global shutter and d) High-performance true global shutter. Reprinted from Automaatioseura.fi (2007) [6]

Global shutter exposes the full frame at one instant which is ideal for high speed imaging. Machine vision system will be used to inspect objects moving on fast conveyor belts so the global shutter is a must to minimize distortion.

Motion blur occurs if shutter is too slow so that the object can move to other location during exposure time.

#### 2.3.4 Illumination

Along with the shutter speed a correct illumination setting is important for the optimal picture. Again, maximum information from the object for the image processing is desirable.

In general, the illumination should be as uniform as possible to minimize any lighting effects. It might be challenging for matrix camera applications inspecting objects moving at high speeds because larger area needs to be illuminated than with line-scan cameras.

Different techniques can be used and suitable technique depends on the imaging conditions and object material as different materials responds differently to light.

## 2.4 Programmable Logic

Engineers mind is wired to think that every single real-life problem is possible to break down to simplest logic statements and that they can also be realized with simple IC's, like NAND or NOR gates.

These statements can be implemented with programmable logic, which essentially refers to a large collection of logic gates that an engineer can “program” or connect together however. It can be found practically everywhere.

### 2.4.1 Field Programmable Gate Array

Field Programmable Gate Array (FPGA) is basically a huge array of programmable logic blocks or element. They can either work as a simple logic gate or more complicated combinatorial functions. An FPGA logic element can be seen in the Figure 8.

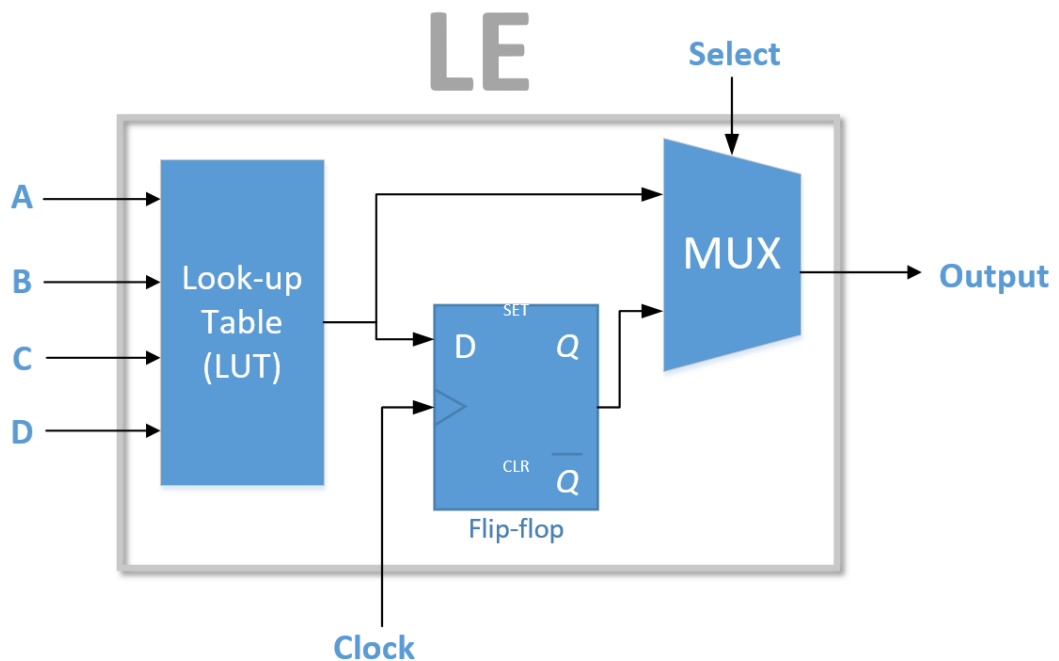


Figure 8. A common, overly simplified example of a logic element.

Due to its parallel nature FPGA is generally suitable for computationally heavy tasks, like image processing or data crunching. Technically an FPGA can be used for any task that requires any computation, complex or simple. Different applications range from aerospace and defense solutions to commercial applications.



FPGAs have been on the market for a while along with ASIC, that is likewise programmable logic and basically a programmable IC, but the structure is decided before manufacturing and cannot be modified later. FPGA can be reprogrammed, so the designer can even install system updates on the field, hence the name field-programmable.

Traditionally FPGAs have been used to prototype a product before hard-wired ASIC implementation. Possibly because production cost-related reasons ASIC-technology has been dominating the market, but recently low-cost and yet powerful FPGAs have emerged. Also tools for system design have become increasingly more sophisticated making the development process easier.

The biggest manufacturers are Xilinx and Altera. The latter one was recently bought by the world's biggest processor manufacturer Intel, which is a faint wind of a change in the industry. Traditionally processor manufacturers have increased clock frequencies of their processors, but over last two decades multi-core processor have been entering the market and the rise of clock frequencies have been slowing down. It seems that parallel programming is becoming increasingly common.

#### 2.4.2 System on Chip

System on Chips (SoC) are basically any kind of systems on a single IC. At the simplest SoC can be a processor and a memory on the same chip. Trend seems to be nowadays to join FPGAs and mobile processors. SoCs manufactured by Altera have an FPGA and one or more ARM-processors.

SoC is divided into two parts, FPGA and Hard Processor System (HPS). They are programmed independently, but they can communicate with each other and usually HPS is used for clock generation.

#### 2.5 Machine Learning

Human brain possesses astounding processing power and can arguably learn from its previous processes. When human detects an object, it is compared to earlier experiences and things learned previously from which the interpretation or decision is made.

Machine learning is trying to achieve same kind of intelligence for a computer. For example, the input data can be compared to previously defined data sets or machine can sort out data based on some classification.

Famous electrical engineer Arthur Samuel defined machine learning in 1959 as a field of study that gives computers the ability to learn without being explicitly programmed.

Machine learning is commonly divided to supervised and unsupervised learning. In supervised learning the user teaches computer or machine to operate on given data set. In this case an answer to specific problem is usually known by the user, the computer just finds more “correct” answers.

Classical example would be an email spam filter that flags incoming mail as “spam” or “junk” based on few example junk emails provided by the user.

In unsupervised learning the computer learns by itself. One example of unsupervised learning is data clustering where data is organized into specific groups (clusters) based on some regularity or common characteristic.

Other areas of machine learning are reinforced learning (trial and error) and deep learning (hierarchical and contextual).

Operating with two discrete states is familiar for a pipe-brained engineer and a convenient way to classify simple things. In the real world this is not the case, humans tend to analyze the world by probability, a state between zero and one or the likeliness of some event occurring.

One way to make computer “smarter” is to introduce it to some probability. Rather than classifying data with discrete labels (junk mail or non-junk mail), data could be organized into groups where the data would most likely belong.

For example, if there’s a 90% chance that a mail is junk it will go to trash, but if the chance is only 50%, let the user decide.

Self-organizing maps (SOMs) are excellent tool for machine vision and a great example of unsupervised learning. The algorithm was developed in 1980s by Finnish professor Teuvo Kohonen.

SOM is a type of artificial neural network. It consists of graphical representation of different features of an image sorted based on their similarities. It is used to interpret large high-dimensional data sets by projecting them to a low-dimensional space [4].

SOM can be implemented completely with unsupervised learning or it compromise between that and supervised learning.

## 2.6 Digital Image Processing

Key aspect of a machine vision system is image processing. European Machine Vision Association (EMVA) describes it as follows:

It [vision technology] deals with images or sequences of images with the objective of manipulating and analysing them in order to

- a) Improve image quality (contrast, colour, etc.),
- b) Restore images (e.g. noise reduction),
- c) Code pictures (data compression, for example) or
- d) Understand and interpret images (image analysis, pattern recognition). [7]

Generally, digital images are operated as matrices of data. An image is considered as a function of its  $x$  and  $y$  coordinates,  $p(x,y)$ , where  $p$  is the value of a pixel.

Range of  $p$  depends on pixels sensitivity to light or so-called *dynamic range*. In an 8-bit image, one pixel has  $2^8 = 256$  different light levels.

More demanding conditions requires more sensitivity. In lumber industry, 10- and 12-bit dynamic ranges are commonly used. Their decimal ranges are:

8-bit dynamic range:  $0 \leq p_8(x,y) \leq 255$

10-bit dynamic range:  $0 \leq p_{10}(x,y) \leq 1023$

12-bit dynamic range:  $0 \leq p_{12}(x,y) \leq 4095$ .

Figure 9 shows different dynamic ranges graphically. On the y-axis is the light level (pixel value) and pixels are on the x-axis.

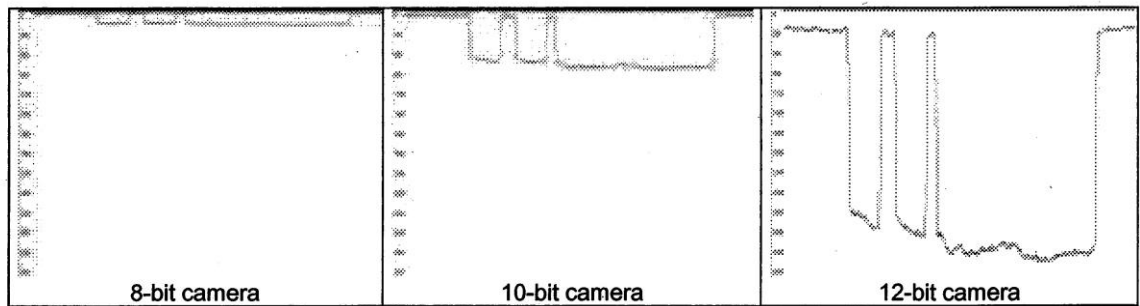


Figure 9. Dynamic range of camera with different sensitivity. The cameras are viewing the same object. The light level is set so that each camera signal is at its saturation threshold. The scale is identical for all three cameras. Reprinted from Vuorilehto (2001) [2].

Saturation threshold means the point where the pixel value saturates, i.e. exceeds the maximum value generating noise into the image.

If object contains relevant colour information, RGB or other colour spaces can be used, which results more complex, although usually more accurate computation than with gray-scale images. With RGB images three sets of processes needs to be accomplished, one for each colour channel.

This thesis focuses mainly on image acquisition. Detailed discussion of image processing methods is not in the scope of this thesis, but some of the most common techniques used in lumber industry that are also suitable for FPGA computing are presented below.

### 2.6.1 Segmentation

Segmentation partitions an input image into its constituent parts or objects [8]. It can be used to identify different features from an image or just to divide image into smaller elements to process.

Goal of segmentation is to organize data so that it can be analysed and processed further. For example, segmentation can be separating foreground (or object) and background from an image.

### 2.6.2 Thresholding

One example of segmentation is *thresholding*, where the image is binarized. That is, the bit values are set as 1 or 0 based on some threshold value  $T$ :

$$p(x, y) = \begin{cases} 0, & p(x, y) \leq T \\ 1, & p(x, y) > T \end{cases} \quad (1)$$

There's plenty of different variations of thresholding algorithms, it would be a completely different thesis to discuss them in detail.

### 2.6.3 Masking

Masking is similar to thresholding where some layers of the image are separated. The goal is to filter out unwanted information. Unlike thresholding the image is not binarized, the original colour space is preserved.

For example, a pixel value bigger than assigned (automatically or manually) value  $M$  is processed further as image  $P(x, y)$ . Values lower than  $M$  are discarded.

### 2.6.4 Histogram

Histogram is a graphical representation of continuous data, where the data is organized into ranges or bins based on their occurrence. It shows distribution of data and allows various statistical methods to be applied.

A bin is represented by rectangle and its area corresponds to percentage of total occurrences. Histograms are widely used in statistics and also in image processing, where pixel information is used to form a histogram.

Histograms can be used with color images as seen in the Figure 10, where a histogram is produced for each of the color channels.

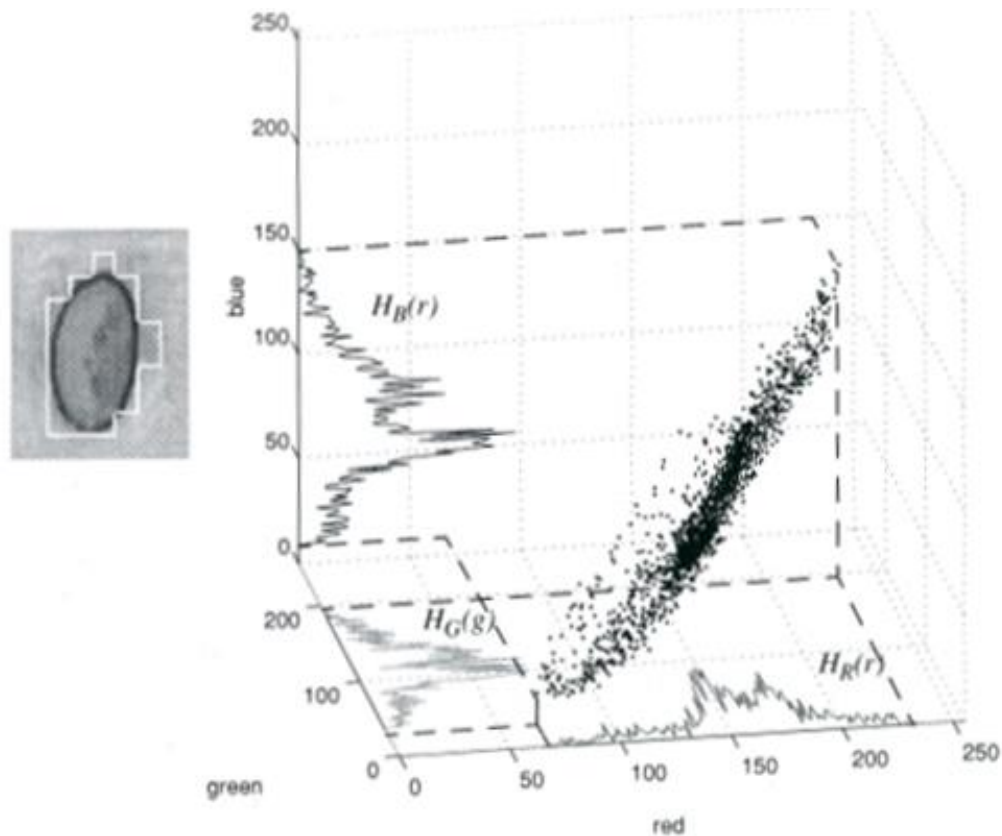


Figure 10. A dry knot<sup>2</sup> and the pixels inside the white boundary plotted in RGB space. Each dot represents one pixel of the knot. The one dimensional marginal distributions, the color histograms, are shown on each axis. [9]

In machine vision, feature extraction is sometimes essential. Histogram can be used to compare the information in segmented area. It plays an important role especially in SOM applications.

### 2.6.5 Data management

If sensor resolution is large the amount of data can cause some problems. Data size depends also on material to be inspected. Data can be reduced with lowering sensor resolution, filtering and compressing.

There are several different digital communications protocols each with their limitations so real-time imaging can prove to be problematic if the data needs to be moved via physical connections.

---

<sup>2</sup> Knot is a base of a branch. Generally it is considered as defect.

## 2.7 Data Transfer and Interfaces

Besides acquiring an image, the data needs to be transferred many times along its way and transfer is always introducing delays to system operation. In industrial systems high speed and accuracy are essential, so optimal data communication techniques must be considered.

### 2.7.1 LVDS

LVDS is a standardized serial communications protocol used for high speed communications, e.g. video processing. LVDS is a low power signal and operates in differential mode that reduces electromagnetic noise due to opposite and equal electric fields.

Maximum distance for LVDS is around 10 meters and maximum baud rate can be as high as 1-3 Gbit/s.

This is useful for moving parallel data on the board.

### 2.7.2 Machine Vision Interface

Traditional machine vision systems need to move data from camera through some kind of communication interface, which usually are combination of different cables and communication standards. In industrial applications speed and high throughput (how much data goes through) is required.

There exists several different machine vision interface standards in the industry each with their advantages and disadvantages (Table 1).

Table 1. Comparison of interface standards. Adapted from "Global Machine Vision Interface Standards Brochure" (2009) [10].

Name of the standard	Camera Link	Camera Link HS	CoaXPress	GigE Vision	USB3 Vision
Latest release	February 2012	May 2012	February 2013	November 2011	January 2013
Transmission	parallel	packet-based	packet-based	packet-based	packet-based
Max. data throughput	850 MB/s	16 800 MB/s	3 600 MB/s	1 100 MB/s	400 MB/s
Frame grabber	yes	yes	yes	no	no
Latency	≥100ns	≥100ns	≥100ns	N/A	N/A

As seen in the Table 1 some interface standards provide plenty of speed. They usually require frame grabber and multi-cable configuration. For example, the CoaXPress can operate in its maximum throughput using eight coaxial cables.



### 3 Making It Smart

Traditionally, a machine vision system is implemented in four parts: lighting, a camera, a *frame grabber* and a computer. Frame grabber is a device that is used to get or “grab” still frames from a camera device and send them further. It can also be responsible of some of the control signals.

Some of the machine vision standards, like USB3 Vision does not need the frame grabber, but their bandwidth is limited making real-time high speed applications impossible.

An increase in image resolution and frame rate (i.e. measuring accuracy) the data gets heavier to process. Data rate can be calculated from

$$Data\ rate = resolution \times \frac{frame}{s} \times bit\ depth. \quad (2)$$

For example, if we run an eight megapixel sensor at 300 frames per second with the bit depth of 8 bits, our data rate would be 19.2 Gbit/s for which only CoaXPress or Camera Link HS would be sufficient. Besides the need of a frame grabber, multiple cables are needed for this kind of system making it awfully robust [10].

In fact, existing machine vision standards cannot cope with very high speed data rate without these kind of configurations. Furthermore, these systems usually rely on PC as the main processing unit. Traditional graphics card does a decent job with its multiple processing elements, but a normal processor with only few cores is a major bottle neck in real-time applications.

Therefore, one solution would be to implement a smart camera, where the main processing unit is combined with the camera thus eliminating the need of frame grabber and a computer. Obviously some sort of user interface is required to control the system.

The main processing unit in smart camera would be an FPGA with sufficient LEs and IOs, such as Altera Cyclone series SoCs. ARM processors could be used for mastering the system and as user interface. Both Altera and Xilinx products are considered for the final product.

These kind of smart cameras exists on the market. Often they are marketed as all-in-one camera systems where all software components, PC, frame grabber and even lighting or lenses are combined. [11]

Inx-Service has decided to pursue own design with the help of existing resources and expertise.

## 4 The Development System

This chapter introduces a prototype of a simple development system that will be used to simulate certain image sensor operation with a test image that is further processed.

The finished system enables development without the actual sensor. Verification of clocking signals and some decisions on the final SoC device can be made before the most recent family of devices is released by the manufacturer, thus accelerating product development.

A high level block diagram of a development system can be seen in the Figure 11.

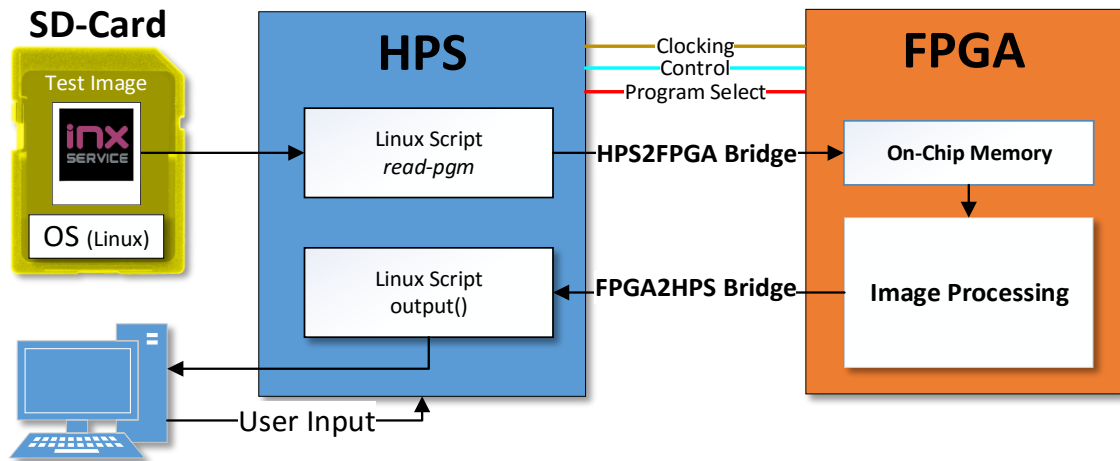


Figure 11. The Development System

The scripts and the test image are saved to SD-card. The Image is loaded to FPGA memory through HPS2FPGA Bridge which is a communication channel defined on the SoC chip. FPGA analyzes the image and is responsible of removing unwanted data from the image. After the raw input is filtered, the data is sent back to HPS for further processing like histogram plotting, SOM generation and other image processing methods.

### 4.1 EBV SoCrates II

The core of this system is a development board called SoCrates II (Figure 12), manufactured by EBV.

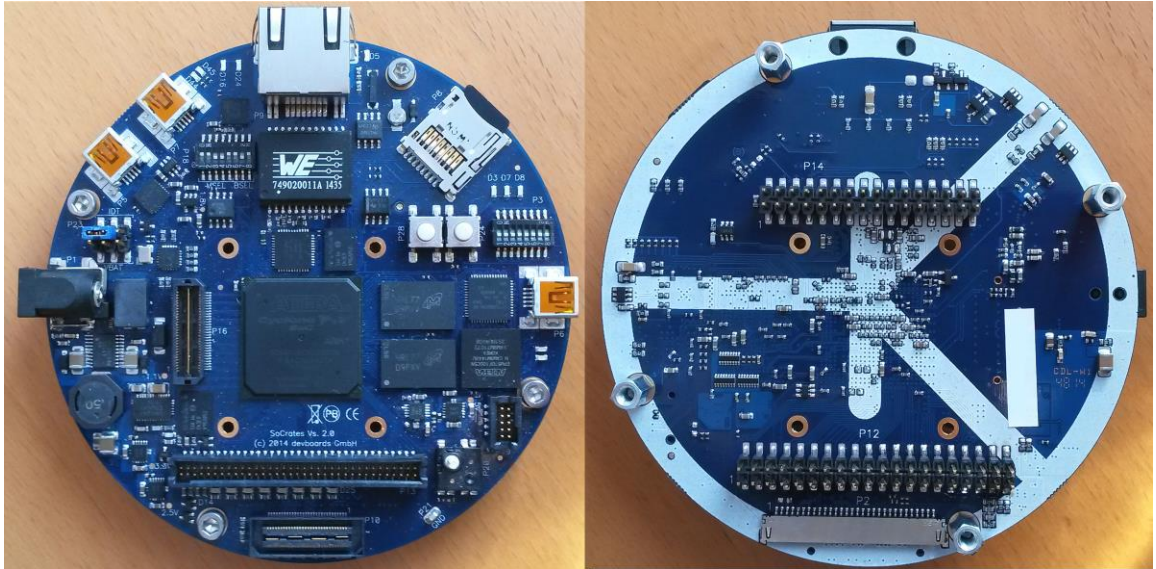


Figure 12. EBV SoCrates II front and back. Altera Cyclone chip can be seen in the middle.

It is based on Altera Cyclone V SoC series and has 110KLE's and 1GB DDR3 memory. SoCrates has also myriad of peripherals on the board like Ethernet, LVDS interface, TFT interface, MMC Reader, USB-Blaster and QSPI Flash (Figure 13).

SoCrates was selected for development purposes based on its SoC device, simplicity and price. It also serves as a learning platform for the designer. EBV uses same board in their own workshops and has provided lab manuals to get started.

## SoCrates II – Cyclone V SoC Evaluation Board

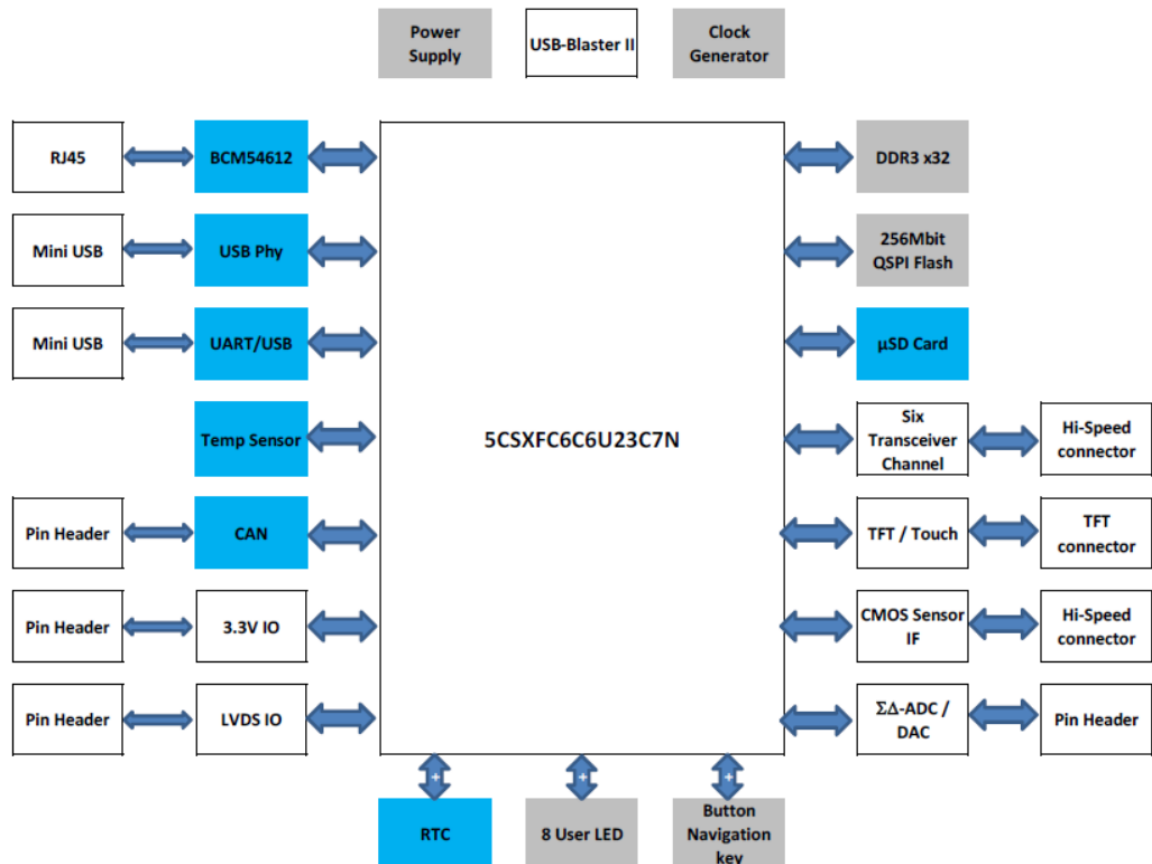


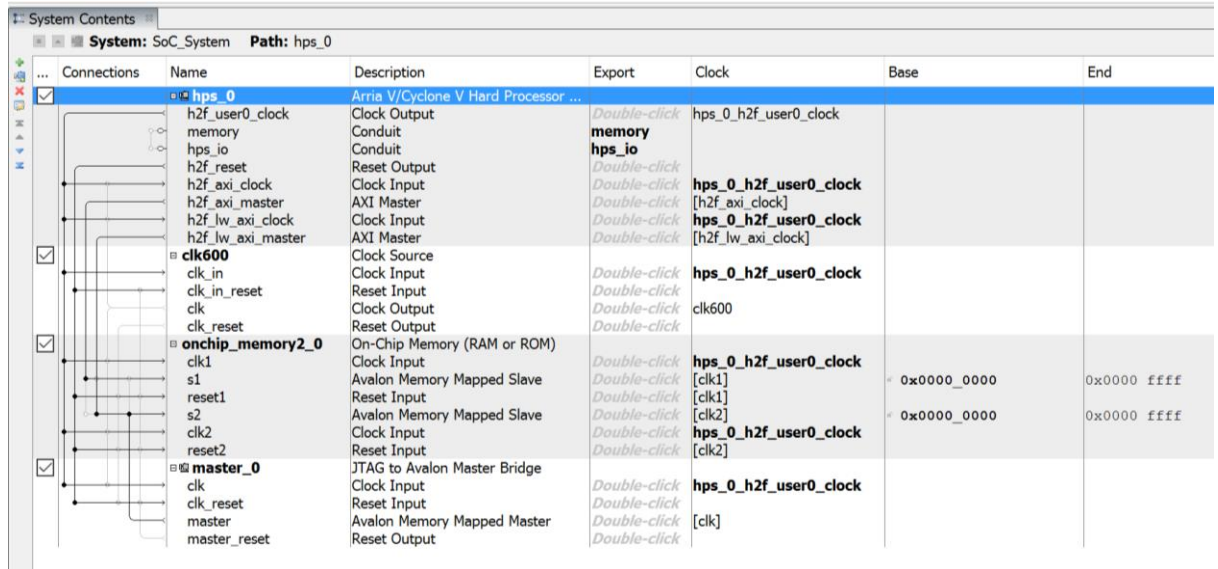
Figure 13. Block diagram from SoCrates II datasheet.

Ideally physical connection from SoCrates output pins to its inputs would be good for simulating the actual sensor. Problem is that the connector pin count is limited and newest sensors can use quite a lot of pins. Other option is to move data completely on chip.

### 4.2 Altera Quartus II 15.0

Quartus II is a logic design software from Altera which is used to design FPGA and CPLD devices. It enables HDL synthesis, timing analysis and RTL simulation. One can design the whole system with hardware description languages like VHDL or Verilog, or simply drawing graphical logic gates and connections. Some tools for conversion from higher to lower level languages are also possible. Having different tool options and extensive intellectual property library or IP library that accelerates design process makes Quartus a heavyweight logic design software. The library consists of IP cores that are prewritten blocks of code that can be treated as components in the design.

One of the key features of the software used in this thesis is Qsys system integration tool. It was used to implement HPS system for our device with all the interconnections and components used in the SoC system that can be seen in the Figure 14. One of the advantages of Qsys is available intellectual property cores or IP cores that can be added to the design with ease.



Connections	Name	Description	Export	Clock	Base	End
✓	hps_0	Arria V/Cyclone V Hard Processor ...				
	h2f_user0_clock	Clock Output	Double-click	hps_0_h2f_user0_clock		
	memory	Conduit	Double-click	memory		
	hps_io	Conduit	Double-click	hps_io		
	h2f_reset	Reset Output	Double-click			
	h2f_axi_clock	Clock Input	Double-click	hps_0_h2f_user0_clock		
	h2f_axi_master	AXI Master	Double-click	[h2f_axi_clock]		
	h2f_lw_axi_clock	Clock Input	Double-click	hps_0_h2f_user0_clock		
	h2f_lw_axi_master	AXI Master	Double-click	[h2f_lw_axi_clock]		
✓	clk600	Clock Source				
	clk_in	Clock Input	Double-click	hps_0_h2f_user0_clock		
	clk_in_reset	Reset Input	Double-click			
	clk	Clock Output	Double-click	clk600		
	clk_reset	Reset Output	Double-click			
✓	onchip_memory2_0	On-Chip Memory (RAM or ROM)				
	clk1	Clock Input	Double-click	hps_0_h2f_user0_clock		
	s1	Avalon Memory Mapped Slave	Double-click	[clk1]	0x0000_0000	0x0000_ffff
	reset1	Reset Input	Double-click			
	s2	Avalon Memory Mapped Slave	Double-click	[clk1]	0x0000_0000	0x0000_ffff
	clk2	Clock Input	Double-click	hps_0_h2f_user0_clock		
	reset2	Reset Input	Double-click	[clk2]		
✓	master_0	JTAG to Avalon Master Bridge				
	clk	Clock Input	Double-click	hps_0_h2f_user0_clock		
	clk_reset	Reset Input	Double-click			
	master	Avalon Memory Mapped Master	Double-click	[clk]		
	master_reset	Reset Output	Double-click			

Figure 14. Stripped-down version of the SoC system used in development system.

First block in the system is the HPS component that defines ARM processor peripherals. It is connected to FPGA side through AXI bridges, which are defined by Altera.

There are three AXI bridges on Altera Cyclone chip. HPS-to-FPGA and FPGA-to-HPS are used for more heavyweight data and high-speed communication. Their width can be configured up to 128 bits. Lightweight HPS-to-FPGA is fixed to 32-bit and it is used for non-critical signals. Here only lightweight bridge is used.

There is also On-Chip Memory to add data storage to FPGA side and JTAG bridge that enables debugging through serial terminal. Also 600 MHz sensor clock is reserved for future use.

#### 4.3 Embedded Linux

Nowadays Linux is widely used in embedded systems and it has almost become de facto operating system of the industry.

Compared to other regular operating systems the open source based Linux has advantages like extensive developer community, easy modification and relatively stable kernel. Open source community provides good support and free material in various different areas.

Maybe the most famous application of Linux are Google Android and Raspberry Pi's Debian.

Intellectual rights must always be considered when designing highly copyright-sensitive products. Although Linux is open source and the source code can be freely modified by anyone, these modifications must always be open source as well. Some of scripts used in this thesis are under the terms of the GNU General Public License.

Since there's two ARM-processors on SoCrates II, Linux was chosen for this project as well. SD-Card is partitioned for an ARM Linux Kernel as seen in the Figure 15.

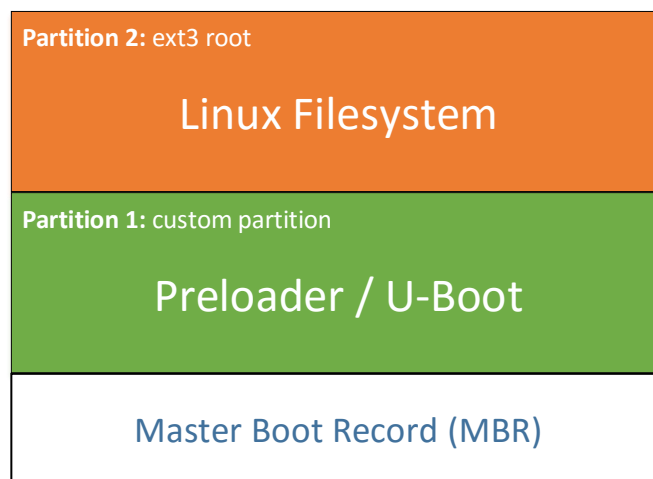
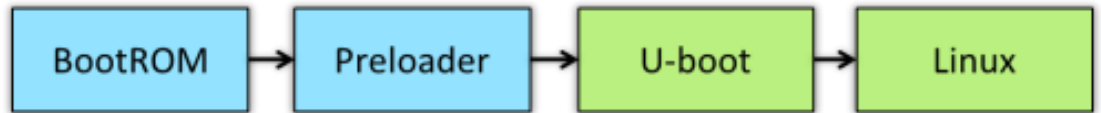


Figure 15. SD-card partitions.

Custom partition holds FPGA configuration files and U-Boot. Linux OS is stored in the partition 2. All the user files are also stored there. This formatting convention is based on the Golden System Reference Design (GSRD) from Rocketboards.org [12].

#### 4.4 Booting the System

SoCrates II boot flow (Figure 16) was also set according to GSRD [13].



Stage	Description
BootROM	Performs minimal configuration and loads Preloader into 64KB OCRAM
Preloader	Configures clocking, IOCSR, pinmuxing, DDRAM and loads U-boot into DDRAM
U-boot	Configures FPGA, loads Linux kernel
Linux	Runs the end application

Figure 16. GSRD Boot Flow. Adapted from Rocketboards.org [13]

Preloader was generated from Quartus II. It contains hardware information of our development system, like clock signals, I/O pins and memory addresses.

U-Boot is an open-source bootloader used in embedded systems. It is GPL-licensed. It was customized together with Preloader with Altera SoC Embedded Design Suite (EDS) software.

Finally Linux is booted and the OS is accessed to interface with the system. Before using FPGA side it needs to be configured. It can be done via JTAG connection on SoCrates and Quartus Programmer, automatically during boot or manually from U-boot or Linux. In development system it is done manually with a simple Linux script (see Listing 1).

```

#!/bin/sh

cat /sys/class/fpga/fpga0/status

echo "Disabling all bridges..."

echo 0 > /sys/class/fpga-bridge/fpga2hps/enable
echo 0 > /sys/class/fpga-bridge/hps2fpga/enable
echo 0 > /sys/class/fpga-bridge/lwhps2fpga/enable
  
```



```

cat /sys/class/fpga-bridge/*/enable

echo "Configure FPGA now (y/n)?"

read ANS

if [ "$ANS" != "y" ] ; then
    exit 1
fi

dd if=/boot/soc_system.rbf of=/dev/fpga0 bs=1M

cat /sys/class/fpga/fpga0/status

echo 1 > /sys/class/fpga-bridge/hps2fpga/enable

echo 1 > /sys/class/fpga-bridge/lwhps2fpga/enable

cat /sys/class/fpga-bridge/*/enable

```

Listing 1. Linux script used for configuring FPGA after HPS boot

Configuration was done manually throughout this study, because the FPGA image is constantly developed. Automatic configuration will be implemented later during product development.

#### 4.5 Generating a Test Image

Test image is generated in image processing software such as Gnu Image Manipulation Program (GIMP). The image is then exported as *pgm-format* that contains bit depth information and pixel data in ASCII- or raw-format.

When simulating a certain image sensor the desired bit depth can be found from datasheet provided by manufacturer. Test image should resemble actual sensor image as precisely as possible.

Here, a simple 8-bit grayscale test image is used for convenience. Figure 17 shows the test image and the .pgm data in a text file.



Figure 17. Example of generated image as .jpeg and the .pgm format.

On the first line on *testimage.pgm* is a version number and after that few comments. It should be ensured that the image is in correct format during readout. The comments can be ignored.

After the comments are first really valid pieces of data, image dimensions and the bit depth. After that bit values are listed individually starting from the first bit. Only 24 lines of values are displayed in the example, since the test image contains total of 256 003 lines, including all pixels and picture information.

#### 4.6 Reading the Test Image

The image is read from the SD-card using a script (Listing 2) into FPGA on-chip memory.

```
#!/bin/sh

linecount=0

address=0xFF21000

while read line;
do

#Start counting lines from 1
linecount=$((linecount+1))
```

```

#Check image details and exit if filetype is invalid
if [ ``$linecount`` -eq 1 ] && [ ``$line`` != ``P2`` ]; then
    echo ``Invalid filetype!``
    exit
elif [ ``$linecount`` -gt 4]; then
    ./devmem2 $address w $line
    address=$((address+1))
fi

done <testimage.pgm

```

Listing 2. Linux script used for reading the test image into FPGA on-chip memory.

The script is based on GNU GPL-licensed C-program devmem2 that is used for reading and writing into memory. [14]

As seen, the script is limited so that only pixel values are read to the memory. Later the reading could be done automatically with Altera IPs. For now, image resolution and bit depth needs to be known by the designer.

Also, devmem2 prints out its actions which can be flood the console screen. Further development is required to include devmem2 functions into the script used here.

#### 4.7 Image Processing

Since there are few suitable IP blocks available for image processing, some of the blocks needs to be written in VHDL. Some algorithms from previous Inx-Service applications exists in C which are used as reference. Because of the publicity of this thesis these copyrighted algorithms cannot be published here in detail.

The suitable image processing methods for this product are still under discussion and are part of next product development step. For development and sensor test purposes a simple thresholding algorithm could be used.

## 5 Discussion and applications

The smart camera can be used in variety of applications and with the re-programmability of FPGA it is possible to start with one application and develop further from there. Here the focus is maintained in the lumber industry.

One application could be a log profiler that analyzes the log before first breakdown. This could be implemented with several cameras in different angles and laser based triangular methods.

Other application could be a board optimizer that optimizes flitch before *edging*<sup>3</sup>. The goal is to find defects and determine where to saw to get the best grade for the board and thus the best value.

### 5.1 Proposed Structure

Besides internal operation camera hardware and mechanics must be taken into account. A camera can be built a number of ways depending on the final circuit design. One way would be a modular three-layer structure (Figure 18) where different parts of the camera are stacked together to form an application specific camera. Different layers can be updated or repaired individually making the design more flexible to user's needs.

---

<sup>3</sup> Trimming of edges from a flitch

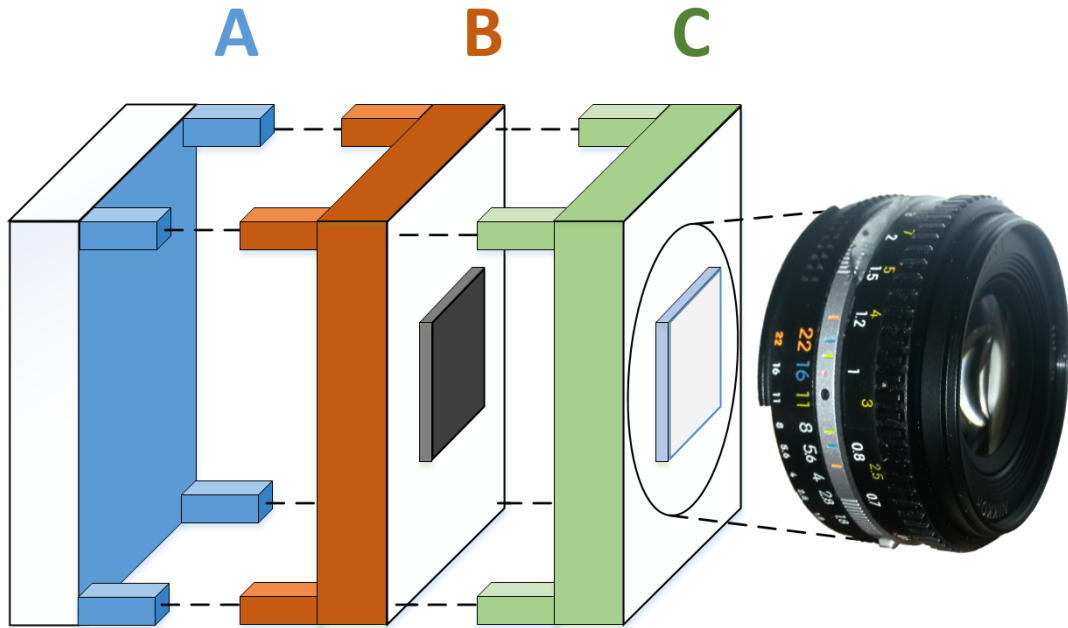


Figure 18. Proposed camera structure: (A) I/O layer (B) FPGA layer (C) Sensor layer and a lens.

Although maybe more flexible, this kind of design can have problems like connecting the high-speed data paths between the layers. One solution to this might be to combine some layers, for example A and B.

## 5.2 OpenCL

Next steps of the product development would be rigorous software development. IP blocks and system operation needs to be written.

Rather than operating with traditional high level programming languages, like C, FPGAs are usually configured with a more hardware driven language like VHDL or Verilog.

There are tools available for high to low level synthesis, which usually speeds up the development process, but inaccuracies in synthesis might lead to variety optimization problems, so HDLs must be kept in mind to maximize the efficiency if needed, like designing some of the IP blocks.

One synthesis tool is Alteras SDK for a programming standard called OpenCL, which is an open standard developed and maintained by Khronos Group. Its strengths are parallelism and its basis on C and C++, which makes it relatively easy to learn if programmer has any background in C.

In OpenCL, a host program controls the whole system that can be combination of multiple work elements or *kernels*. These work in parallel each processing some piece of information and returning the answer when complete.

Operation of the FPGA is essentially parallel which makes OpenCL a great candidate for development purposes. It considered for some uses in the system discussed in this thesis.

## 6 Conclusions

The goal of this thesis was to present first steps of the design process of a new smart camera system. It involved comparing suitable techniques and technologies, some preliminary planning and prototyping a simple development system.

Major contribution of this thesis is the research on available materials. Some design approaches were considered and the development system might prove useful in the near future. Some improvements in the system, such as more advanced FPGA integration needs to be made before further development.

The next step would be to start software development with the finished test equipment. The focus will be on the first processes in saw line. Research on OpenCL could accelerate the software development and should be researched more.

Further discussion on whether to continue pursuing own design or to utilize some product from other manufacturer is also needed.



## References

- [1] R. Leino, "Konenäkö kasvaa uusille markkinoille - T&T," Tekniikka&Talous, 9 September 2003. [Online]. Available: <http://www.tekniikkatalous.fi/tekniikka/ict/2003-09-09/Konen%C3%A4k%C3%B6-kasvaa-uusille-markkinoille-3281637.html>. [Accessed 5 November 2015].
- [2] J. Vuorilehto, Size control of sawn timber by optical means in breakdown saw machines, Helsinki: Helsinki University of Technology, 2001.
- [3] STMY, FSS, TTF, "Pohjoismainen sahatavara: Lajitteluohjeet," Gummerus Kirjapaino Oy, 1994.
- [4] O. Silvén, M. Niskanen and H. Kauppinen, Wood inspection with non-supervised clustering, Oulu: Springer-Verlag, 2003.
- [5] Teledyne DALSA, "CCD vs. CMOS," [Online]. Available: <https://www.teledynedalsa.com/imaging/knowledge-center/appnotes/ccd-vs-cmos/>. [Accessed October 2015].
- [6] Machine Vision News, "Electronic Shuttering for High Speed CMOS Machine Vision Applications," 2007. [Online]. Available: <http://www.automaatioseura.fi/jaostot/mvn/mvn2007/parameter.html>. [Accessed November 2015].
- [7] European Machine Vision Association (EMVA), "An introduction to machine vision," [Online]. Available: <http://www.emva.org/cms/index.php?idcat=38>. [Accessed October 2015].
- [8] R. C. Gonzalez and R. E. Woods, Digital Image Processing, Addison-Wesley Publishing Company, Inc, 1992.
- [9] H. Kauppinen, Development of a Color Machine Vision Method for Wood Surface Inspection, Oulu: University of Oulu, 1999.
- [10] AIA, EMVA and JIIA., "AIA - Vision Online - Vision Standards," 2014. [Online]. Available: <http://www.visiononline.org/vision-standards.cfm>. [Accessed August 2015].
- [11] A. Wilson, "Smart Cameras Challenge PC-based Vision Systems for Dominance," Vision Systems Design, 1 May 2013. [Online]. Available: <http://www.vision-systems.com/articles/print/volume-18/issue-5/features/smart-cameras-challenge-pc-based-vision-systems-for-dominance.html>. [Accessed November 2015].

- [12] "Creating and Updating SD Card," Rocketboard.org, [Online]. Available: <http://rocketboards.org/foswiki/view/Documentation/GSRD151SdCard>. [Accessed September 2015].
- [13] R. Bacrau, "GSRD - Boot Flow," Rocketboards.org, 26 June 2014. [Online]. Available: <http://rocketboards.org/foswiki/view/Documentation/GSRDBootFlow>. [Accessed September 2015].
- [14] J.-D. Bakker, *devmem2.c*, 2000. Available: [http://www.makelinux.net/books/embedded\\_linux\\_kernel\\_and\\_drivers/text152](http://www.makelinux.net/books/embedded_linux_kernel_and_drivers/text152)

