CRANFIELD UNIVERSITY

OSCAR ZAYAS QUINTANA

HIGH QUALITY FRAME GRABBER FOR AN IR IMAGING CAMERA

SCHOOL OF AEROSPACE, TRANSPORT AND MANUFACTURING

Computational and Software Techniques in Engineering MSc: Digital Signal and Image Processing

MASTER OF SCIENCE

Academic Year: 2015 - 2016

Supervisor: Dr Jane Hodgkinson

August 2016

CRANFIELD UNIVERSITY

SCHOOL OF AEROSPACE, TRANSPORT AND MANUFACTURING
Computational and Software Techniques in Engineering MSc: Digital Signal and Image Processing

MASTER OF SCIENCE

Academic Year 2015 - 2016

OSCAR ZAYAS QUINTANA

High quality frame grabber for a IR imaging camera

Supervisor: Jane Hodgkinson
August 2016

This thesis is submitted in partial fulfilment of the requirements for the degree of Master of Science

ABSTRACT

Image processing techniques represent an ever-growing field of engineering, which is nowadays taking a great importance in a number of sectors as diverse as gas sensing or three-dimensional imaging. In order to satisfy the high demand of imaging systems, not only new cameras are needed, but also a great deal of frame grabbers in charge of collecting and processing the data captured by those cameras. These frame grabbers must be customised for each camera, dealing with its specific features and operation protocols.

In this MSc thesis, a new frame grabber was designed and developed, customised for an innovative infrared imaging camera. The system is based in a FPGA device, making use of an external ADC converter to digitalise the video data coming from the camera and a 2 MB SRAM memory to save this data. The whole development is done by means of VHDL language, a low level hardware description language (HDL) used for FPGA design.

The result of the project is a fully operating frame grabber, with a number of parameters configurable by the final user and with two operation modes for the final user plus one for calibrating the system. The ones for the final user are: one for the creation of a real time video stream (up to 12 fps) and the other one for buffering up to 254 images at high speed (capture rate up to 1025 fps). Besides, only 2% of the logic resources of the FPGA were used.

Thus, a high quality and inexpensive infrared imaging system was developed. Furthermore, the universal nature of VHDL design and the reduced number of resources employed make the migration to a simpler and cheaper FPGA an easy task. On the other hand, the RS-232 standard used for the transmission of data to the final computer is the only bottleneck of the system, and it could be substituted with a faster standard without modifying any other part of the design.

Keywords:

FPGA, infrared camera, InGaAs sensor, image processing, VHDL, Hamamatsu, Altera, Gas Sensing, image processing.

ACKNOWLEDGEMENTS

I would like to express my greatest gratitude to Dr Jane Hodgkinson, my supervisor, for her wise advice every time I asked for it, always patient and willing to assist me. Besides my supervisor, I could not be more grateful to Dr Thomas Kissinger and Dr Tom Charret, who were deeply involved in the project since the very beginning, providing me with technical support every time I needed it. The success of this thesis would have not been possible without any of them.

I am also immensely thankful to my family for their moral encouragement and infinite patience during the last times. I will always be in debt to you.

Last but not least, I would like to show my appreciation to all those who, directly or indirectly, lent a hand in the development of this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

ADC	Analogue to Digital Converter
CLK	Clock
FPGA	Field Programmable Gate Array
FPS	Frames Per Second
GPIO	General-Purpose Input/Output
GUI	Graphical User Interface
HDL	Hardware Description Language
HDMI	High Definition Multimedia Interface
IEEE	Institute of Electrical and Electronics Engineers
IR	Infrared
LAN	Local Area Network
LSB	Least Significant Bit
MCLK	Master Clock
MSB	Most Significant Bit
MSP	Master Start Pulse
ROM	Read-Only Memory
SRAM	Static Random-Access Memory
TTL	Transistor-Transistor Logic
UART	Universal Asynchronous Receiver/Transmitter
UART RX	UART Reception
UART TX	UART Transmission
UTP	Unshielded Twisted Pair
USB	Universal Serial Bus
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

1 INTRODUCTION

1.1 Context

After a long time under development, image processing is no longer a field of conceptual experimentation, but a reality in terms of practical implementation. Nowadays, these techniques mean both the implementation of innovative methods and the substitution of traditional methods with more efficient and autonomous alternatives, being used in a number of different sectors: gas sensing, medical industry, stereo vision…

In the case of gas sensing, capturing images within the infrared wavelengths range, different gas leakages can be detected. Due to the typically low resolution of some infrared imaging cameras, certain image processing techniques must be applied for each specific system, such as the interpolation of low resolution images in order to obtain the shape of a real gas cloud. Another example of image processing is the technique used for creating stereo vision, where the images coming from two cameras placed one next to the other and focusing to the same point are compared, obtaining a disparity image that provides an output with a three-dimensional effect.

There is a wide variety of cameras available in the market, each of them with different features (infrared imaging, high resolution…) depending on the target application, but all have something in common: the need of a frame grabber for collecting the captured images and transmitting them to a computer. Apart from this, the frame grabbers can also perform the specific image processing techniques requested for each application. However, the process of collecting and transmitting the data is not a trivial task and, being this the basis for any frame grabber, it is paramount to design and develop this part of the system as accurately as possible.

Regarding the technology used for building the frame grabbers, there are some compelling reasons why traditional systems are all based in FPGAs, which are devices composed of a large matrix of logic gates configurable by means of HDL languages. This devices allow the designer to split the processing

algorithms in different parts and to implement them in parallel, working with high frequency clocks. These features make possible to dramatically decrease the latency between video frames. Besides, being HDL low level languages, they are intended to work at bit level and to control the state of every signal in every clock period, which is essential to synchronise the electronic system with any camera.

The motivation for this thesis came with the release of a cheap 64x64 InGaAs image sensor for infrared imaging by Hamamatsu. Traditionally, a sensor with these features could cost tens of thousands of pounds, whilst the one by Hamamatsu is available for £1500. In respect to the type of sensor, the ones based in InGaAs photodiodes cover the wavelength range from ~800 nm to 1700nm, within which many gases absorb the light, whereas standard silicon cameras stop at 1000 nm. Therefore, this kind of sensors are extremely useful for gas sensing applications. On the other hand, the Group of Engineering Photonics of Cranfield University has a research line in gas sensing, where the use of IR cameras is essential, and they had acquired the IR imaging sensor created by Hamamatsu. In this context, the need of a frame grabber able to collect the images captured by the InGaAs image sensor was clear, since it would allow the use of this innovative sensor in future researches.

1.2 Structure of this report

This report is arranged in 5 main sections. In the INTRODUCTION, the project is put into context and the motivation for its development is stated.

In Section 2 (LITERATURE REVIEW) a brief synopsis of the preliminary research is provided. In this section, a deeper view into the frame grabbers is given, explaining some important applications for this systems and their general architecture in the work previously done in the field. Besides, the most suitable communication standards for the transmission of the images are presented.

In Section 3 (DESIGNED SYSTEM) section, an overview of the system is given, presenting the equipment used for the development and giving a detailed view of the system’s architecture.

In section 4 (DEVELOPMENT OF THE SYSTEM) all the details of the development process are stated, outlining the challenges and problems faced during the process and explaining the approach taken to overcome them. This section is separated in four subsections, each of them dedicated to a stage of the development.

In Section 5 (RESULTS) the results obtained with the developed imaging system are presented in different forms. First, the raw data received in the computer is shown and explained. Secondly, the useful data signal decoded from the raw data is presented, along with the data in image format. Besides, the performance of the frame grabber in terms of transmission and capture time is stated, and the usage of the FPGA device's resources is also shown.

In Section 6 (CONCLUSIONS AND FUTURE WORK), the conclusions for the results obtained in the previous section are explained, and future work that could be done to improve the system is stated.

2 LITERATURE REVIEW

2.1 Introduction

A frame grabber is an electronic device aimed to capture, store and transmit frames from an analogue video signal or digital video stream, being also useful for performing image processing in certain applications. Initially, frame grabbers were able to store only one frame of video, but modern ones are already able to store multiple frames, allowing even the capture of multiple video streams concurrently.

Regarding the image processing capabilities, there are many techniques usually performed in frame grabbers, which could well be classified in the following main categories[1]:

- *Transformation* of an image into a new image of the same class. E.g. linear and nonlinear filtering, image resizing.
- *Combination* of two (or more) different images for the creation of a single output image of the same type, by means of combining every pair of elements from the input images.
- *Measurement* of an image in terms of descriptive statistics, reducing the input image into a scalar or vector containing information such as mean or standard deviation of pixel values.
- *Conversion* of an image into an image of a different class. E.g. Discrete Fourier Transform.

Among all the available image processing operations, the image resizing and real time compression (using algorithms like MPEG) [2] stand out due to their potential towards decreasing the delay between frames.

However, any image processing technique performed in a frame grabber relies heavily on the plain process of collecting the data coming from the source of video, converting it into useful digital formats and transmitting it to the opposite endpoint. The development of a system able to perform this process in a reliable way is not a trivial task, and it is clearly the most important part in the

development of any frame grabber with further capabilities. Therefore, the development of this basis is the main target for this project.

2.2 Applications

There are several applications in which the fast rates and resolution of frame grabbers are paramount, having led the technology to great developments in those aspects. Some of the most remarkable applications are found within the next categories:

- Gas Sensing
- Manufacturing
- Medical Imaging
- Stereo Vision
- Consumer Devices
- Surveillance Cameras

The most relevant of these categories in terms of scientific and technological innovation are now explained in more detail.

2.2.1 Gas Sensing

Aimed to detect and measure gas concentrations in different environments, the field of Gas Sensing brings one important application for the targeted frame grabber, since the final imaging system will be mainly used for this purpose. The application of gas sensing is highly valuable for a wide range of applications, from safety in process and petrochemical industries to the monitoring of different gas species (e.g. greenhouse gases) in atmospheric science[3][4].

Gas sensing is a difficult task, with traditional systems limited to detect leaks a short distance away due to the nature of the point sampling detectors in which they are based. Besides, it is hard to set the “zero trace” for which there is no leakage in every single point [5].

On the other hand, traditional systems are not only inefficient in terms of technical implementation, but also in the economic aspect, since sensors are

supposed to be installed every 5-6m. Therefore, the deployment of a traditional system also involves a great cost.

In this context, imaging gas leaks brings the great advantage of covering wide areas from a remote position with an only IR camera. Furthermore, the leakage source could be directly found, along with the wind direction, and multisource-leakages could be more easily identified [5].

Based in the optical absorption of the gas species [3], any imaging system developed for this application requires the selected camera to capture images in the IR range, since gases absorb light within this range and can only be visualised with a sensor able to detect this frequencies, which is not achieved with conventional cameras. This is the reason why this project will be based on an IR camera.

Figure 2-1 shows an image of a gas leakage detected with a gas imaging system:

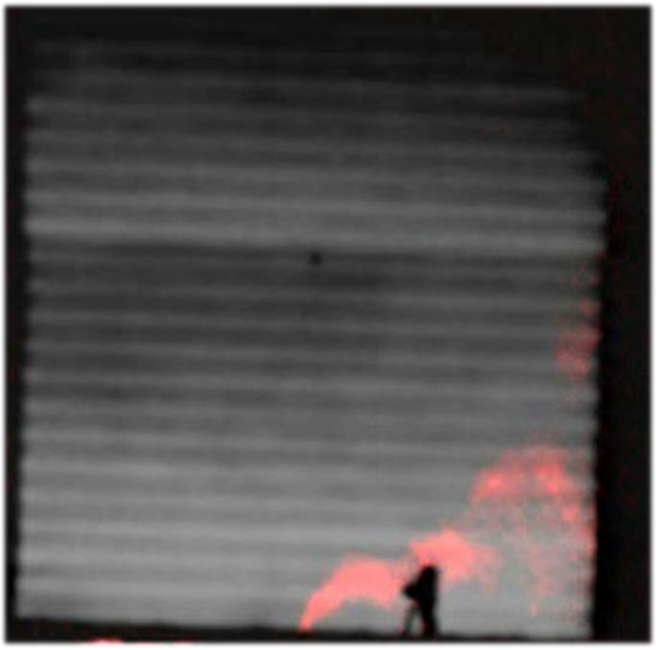


Figure 2-1. Outdoor methane leak [3] **detected with a gas imaging system [image taken from** [6]**]**

On the other hand, regarding the dimensions of the IR images needed for this application, Graham Gibson et. al. [7] proved that image dimensions as reduced as 10x10 pixels are enough for the effective detection of gas leakages. Making use of an incident laser, they combined the resulting beams scanned over an area (single points) to create low-resolution images (10x10 pixels). Later, these low resolution images are interpolated in order to create the image of a gas cloud detected in a highly reliable way, which is combined with a high resolution image of the scene [7]. Thus, 10x10 images created from individually scanned points can be enough to locate gas emissions in a certain scene. The process and results are shown in Figure 2-2:

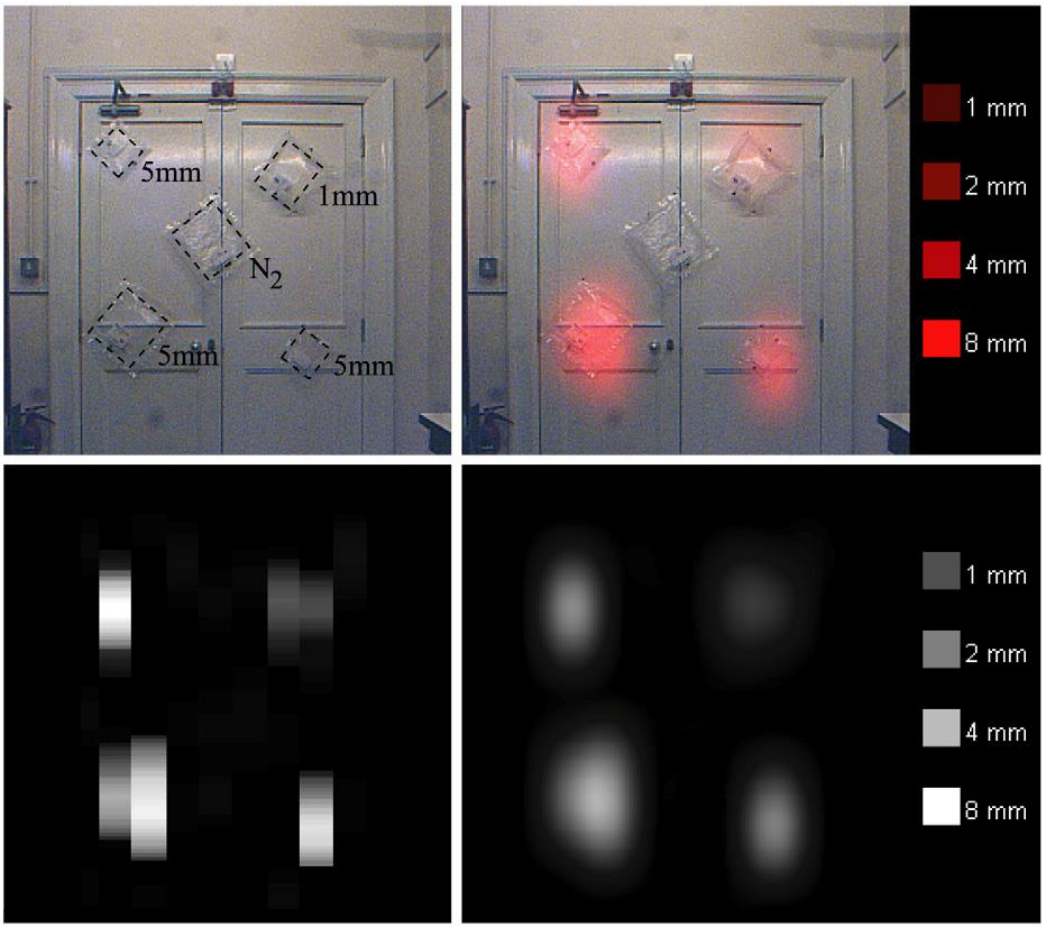


Figure 2-2. Image of sample gas bags (top left), low-resolution gas data (bottom left), gas data interpolated (bottom right) and combined images (top right) [7]**. (image taken from** [7]**)**

2.2.2 Manufacturing

The most representative example of industrial applications is the use of frame grabbers in pick and place machines, implementing vision guidance systems. With the use of this robots for production's automation, the productivity and accuracy are hugely improved, whilst decreasing the costs [8]. Those machines are monitored with cameras whose images are collected and processed with frame grabbers that convert the data to a format understandable for the machines.

During the last ages, major improvements have been made in this field, such as 3D image calibrations, which means estimating the parameters for the interaction among the machine, the camera and the objects' coordinates, allowing the calculation of robot's trajectory using the collected 3D data [8]. In this process, frame grabbers play a crucial role.

2.2.3 Medical Imaging:

In modern operating room environments, highly accurate and low-latency video images are needed for surgery procedures (i.e. X-ray, ultrasound and endoscopy). Since the data can be received in different formats from diverse sources, a quick and reliable normalization is essential [9]. This task is usually performed by high resolution frame grabbers, providing resolutions of up to 1080p in space, 60 fps in time and 10 bit in terms of pixel definition [9].

Besides, IR cameras can be particularly interesting in the field of medical imaging due to their high sensitivity to temperatures, and because some materials present a high absorption at certain IR wavelengths. The applications or IR medical imaging are as diverse as the early detection of dental decay thanks to the low light scattering of dental enamel at certain IR wavelengths [10] (see Figure 2-3), or the diagnosis of body diseased parts by means of temperature anomalies [11], being able to detect even breast tumours [12].

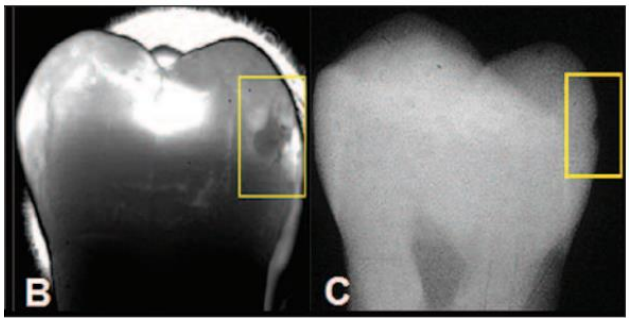


Figure 2-3. IR image (B) vs traditional X-ray image of a lesion in a tooth (image taken from [10]**)**

2.2.4 Stereo Vision

Most commonly known as three-dimensional imaging, stereo vision has been a burgeoning field during the last years. Based in two video input streams generated with two cameras that recreate the position of the human eyes, pointing to the same objective and placed with a slight separation between them, the use of frame grabbers for developing the image processing modules necessary to create the disparity image is particularly useful. The disparity image shows the difference in terms of the image location of an object pointed with both cameras of the stereo vision system.

Thanks to the performance of the FPGAs, the stereo vision processing HW implementation (in FPGA) can be hundreds of times faster than the processing performed in conventional computers [13].

An example of three-dimensional estimation of a surface is shown in Figure 2-4:

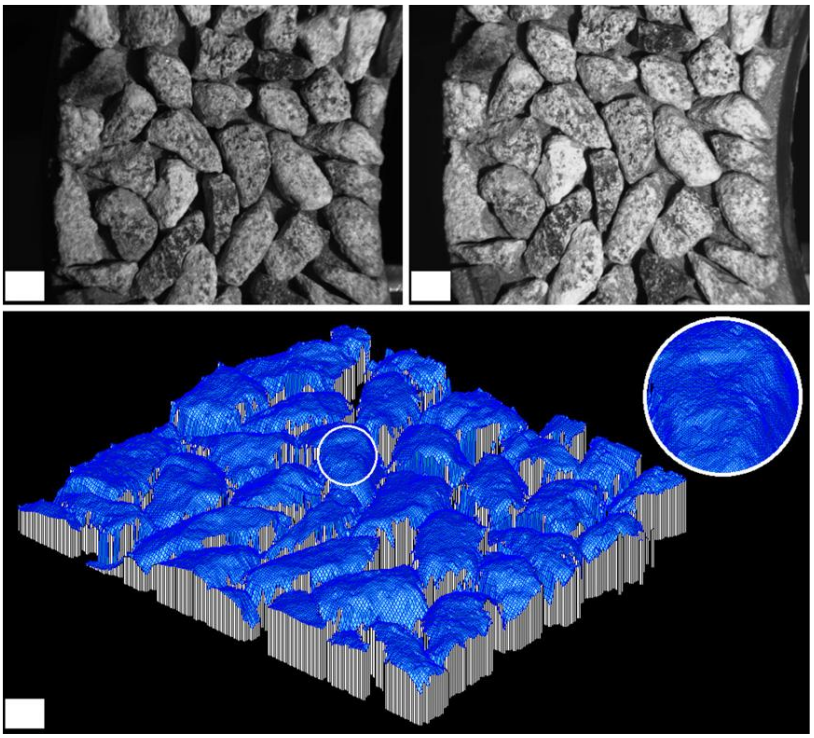


Figure 2-4. Three-dimensional estimation of a surface using stereo vision techniques (image taken from [14]**)**

2.3 FPGA based frame grabbers

Field Programmable Gate Array (FPGA) chips are integrated circuits aimed to be configured after the production process. The internal logic of the chip can be fitted to the particular needs of each application, providing the final user with a great flexibility. The internal logic is composed of an array of gates that can be interconnected however the designer wants to perform complex logical operations. Those relationships can be easily defined by means of Hardware Description Languages (HDL), such as VHDL or Verilog.

Thus, the user can easily design complex electronic systems that, afterwards, can be either transferred to hard-wired designs for manufacturing targeted to end users, or be used simply integrating the FPGA in a more complex system, facilitating the alteration of the electronics when updates or corrections are needed.

When designing a frame grabber system, the use of an FPGA as a base for it is a really valuable choice. There are multiple reasons for this, but the main ones are related to the performance of the device, making FPGA-based frame grabbers particularly useful for applications that require low latency between processed frames, being the latency the time that the data of a certain frame needs to traverse all the system.

FPGAs allow the parallel implementation of different processes. Thus, a vision algorithm can be split into multiple iterative processes that can be performed in parallel on the FPGA. In fact, many image processing algorithms operate in parallel, what makes them suitable for running on FPGA based frame grabbers [15].

Furthermore, FPGAs are able to receive images data and process individual bits at rates as high as hundreds of MHz (depending on the particular clock), performing the data transfer and processing in every clock cycle and achieving a high-speed bit-level processing [15]. For this two reasons, the image processing time is reduced and therefore the latency between frames is decreased significantly.

Another important reason for choosing FPGA devices is that, being configured by means of low level languages (HDL languages), they are prepared for working at bit level, offering a full control of every bit involved in the operation in each clock cycle. This feature is essential for the development of a frame grabber, because the communication between the system and the camera must be adapted to the specific protocol of the camera in order to collect the images correctly.

Apart from this, there are other great assets that could be really valuable in the FPGA-based frame grabbers, such as the flexibility and accuracy provided by a system whose functionality could be altered by means of a simple modification in the HDL code, or the reduced sized of the whole system thanks to the small package of any FPGA. Besides, for applications in which power consumption is a serious constraint, low power consumption FPGAs are available for the optimisation of the system.

2.3.1 Related work

The field of FPGA-based frame grabbers for image processing has been under constant development during the last times. There are references of fully operating systems from more than ten years ago, developed with the technology constraints of those times, such as memories of only 1MB to store the images and bit resolution of only 8 bits [16]. However, the design principles underlying these obsolete systems are perfectly transferrable to current technologies, what can lead to greater designs.

This is the case of the work developed by Pierre Greisen et. al.[17], who developed a frame grabber aimed to create stereo vision video (3D imaging) taking the video streams of two cameras, and getting to a performance of a 30 fps frame rate with an image resolution as high as 1080p (1920x1080). This system was based in an Altera Stratix III FPGA in conjuction with a DRAM memory, transferring the data to the final computer by means of PCI Express communication standard.

Georgoulas et. al. [18] presented a frame grabber based in an Altera Stratix IV FPGA, also receiving two video streams from two different cameras in order to create an stereo vision video. The modules design within the FPGA are able to create the signals for the three dimensional recreation with a resolution of 1280x1024 at a frame rate of 251 fps. However, the system does not present a transmission method, what would more than likely suppose a bottleneck in the output frame rate.

Mateusz Michalak et. al. [19]. developed a frame grabber for a digital progressive scan image sensor, based in a Xilinx Spartan 6 FPGA. Using only two internal block RAMs, this system is able to transfer frames to the final endpoint with a resolution of 720p (1280x720) at a rate of 60 fps, by means of a HDMI output.

Apart from the stated ones, there are several technological developments on this field but, regrettably, they are all clearly too complex for a MSc project. Therefore, they are not a great target to follow for this project, but they do mean

a great reference for demonstrating that the development of a FPGA-based frame grabber is perfectly feasible, and they give a view of the different equipment and communication systems usually selected for performing the operation.

In all these frame grabber systems, the communication standard is usually a bottleneck in the system, which, being based in FPGAs that work with high frequency clocks, can perform their operation at rates out of the reach of many communication standards. For this reason, the selection of an appropriate communication standard is crucial for maximising the performance of the whole system.

2.3.2 Alternatives

It is also worth to consider the option of implementing the frame grabber in alternative and more traditional platforms, such as microprocessors or microcontrollers. However, although it might be possible to implement a frame grabber system in this kind of platforms, there are some issues making this option unsuitable.

Firstly, this kind of platforms are not prepared to deal with strict timings, which is absolutely mandatory for a frame grabber that needs to be adapted to the clearly defined and high speed communication protocol of any camera. In fact, due to the constraints in the protocols of the cameras, digital communication signals must be typically controlled with maximum accuracy, managing them in every clock cycle, which can be only achieved with a FPGA.

Secondly, although it might be possible to work at bit level in microprocessors, they are not aimed for this purpose and it would require the use of low-level languages that would make such a complex design quite difficult to be implemented. On the contrary, FPGA devices are directly designed for this purpose, and HDL languages, which are also low-level languages, are able to deal with bit level operation in a much more convenient way.

In fact, there is a wide catalogue of commercial frame grabbers available in the market, and all of them are based in FPGA. Regarding the suitability of the

commercial frame grabbers, they include all the necessary hardware for building a high performance frame grabber, offering different alternatives, especially in terms of communication interfaces. Thus, choosing the appropriate frame grabber board, the designer can optimise the operation of the imaging system. However, although manufacturers might provide some predesigned functions to deal with certain aspects of the frame grabber's operation, the final development of the system must be done by the designer.

On the other hand, once the decision of developing the frame grabber in base to a FPGA is made, many evaluation boards available in the market offer interesting features with a price typically lower than the commercial frame grabbers. Evaluation boards are electronic cards that, based in a certain FPGA, include a great deal of generic peripherals, such as memories (SRAM, DRAM, ROM…), SD card slots, displays, communication interfaces (RS-232, HDMI…) etc. Thus, the design of a frame grabber system can also be implemented with this kind of equipment, being able to give a good performance with a more reduced budget.

2.4 Communication standards for image transmission

When selecting the most suitable frame grabber, the communication standard that it uses for transmitting the data to the computer after collecting it from the camera is usually the main factor to be considered. There is a wide catalogue of commercial frame grabbers, based mainly in the following standards:

2.4.1 SDI (Serial Digital Interface):

Standards used for the transmission of uncompressed and unencrypted digital video signals, by means of coaxial wires with BNC connectors (see Figure 2-5). For the data transmission, NRZI (Non Return to Zero Inverted) codification is used: either positive or negative voltages for every bit (not neutral state available) with level transitions only when a certain logical level (either 0 or 1, depending on convention) is detected in the next bit. An example of level transition with low level bits in shown in Figure 2-5.

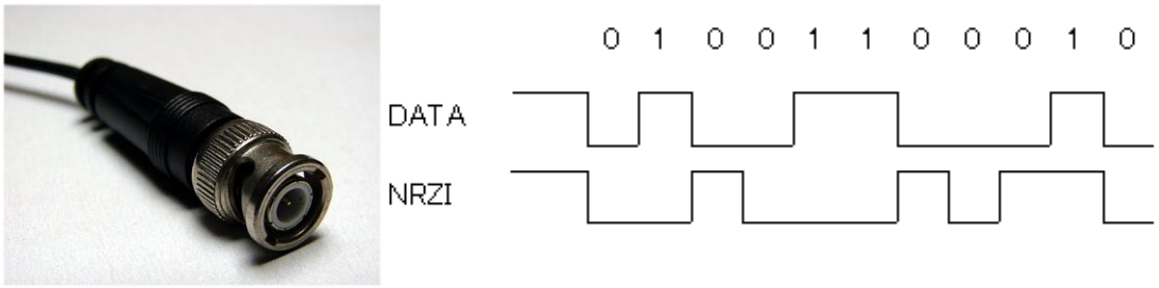


Figure 2-5. BNC connector and NRZI codification example (image taken from [20]**)**

Since the first SDI standard was introduced, multiple versions have been developed, with new and higher bitrates. They are shown in Table 2-1, along with their bitrates and most representative video formats [20]:

Table 2-1. SDI standards

Standard	Bitrates (Gbit/s)	Video format
SMPTE 259M (SD-SDI)	0.27, 0.36	480i, 576i
SMPTE 344M (ED-SDI)	0.54, 0.36	480p, 576p
SMPTE 292M (HD-SDI)	1.485	720p, 1080i
SMPTE 372M (Dual Link HD-SDI)	2.970	1080p60
SMPTE 424M (3G-SDI)	2.970	1080p60
SMPTE ST-2081 (6G-SDI)	6	2160p30
SMPTE ST-2082 (12G-SDI)	12	2160p60

The resolution of the video format allowed for each standard is directly proportional to the maximum bit rate allowed, with displays resolutions from 640x480 pixels in the case of 480i (with typical display resolution of 640x480) to 3840×2160 pixels for 2160p standards (commonly known as 4K UHD).

2.4.2 HDMI (High Definition Multimedia Interface):

Proprietary standard owned by HDMI Licensing LLC used for the transmission of uncompressed video (and compressed or uncompressed audio), implementing the interoperability standards created in CEA (EIA/CEA-861x) [21]**.** There are five variants of the HDMI connector, offering either increased resolution or reduced size (for portable devices) comparing to the original one. Some of them are shown in Figure 2-6:

Figure 2-6. HDMI connectors (image taken from [22]**)**

Regarding the versions, up to 6 have been developed since HDMI was introduced, with different capabilities in terms of throughput and resolution, among others. Not all the features of a certain version are always implemented in the devices, but some variations are usually made. Focussing only in the maximum throughput of each version, the classification of the version could be made as presented in Table 2-2 (only useful throughput, the actual one is a 25% higher since every 8-bit colour signal rides in a ten-bit word, with 2 bits of overhead):

Table 2-2. HDMI versions

Version	Bitrate
HDMI 1.0	3.96 Gbit/s
HDMI 1.1	3.96 Gbit/s
HDMI 1.2	3.96 Gbit/s
HDMI 1.3	8.16 Gbit/s
HDMI 1.4	8.16 Gbit/s
HDMI 2.0	14.4 Gbit/s

2.4.3 USB (Universal Serial Bus):

Standard initially developed for the communication and power supply between computers and peripherals, which nowadays is used for every kind of devices. The typical USB connector is shown in Figure 2-7:

Figure 2-7. USB connector (image taken from [23]**)**

Once again, focusing the classification in the data rate offered by each USB version, the different version of USB can be sorted as in Table 2-3 [24]:

Table 2-3. USB versions

Version	Bitrate
USB 1.0	12 Mbit/s
USB 2.0	480 Mbit/s
USB 3.0	5 Gbit/s
USB 3.1	10 Gbit/s

2.4.4 GigE (Gigabit Ethernet):

Being part of the Ethernet family of communication standards, Gigabit Ethernet can provide data rates of up to 1 Gbit/s while maintaining compatibility with systems of the same family deployed over the last three decades [25].

There are two variants within the GigE standard, each including different versions. The first variant to be released by IEEE, called 1000Base-X, was already capable of providing data rates of 1 Gbit/s, carried on optical fibre. Afterwards, aiming to take this technology to the desktop connection market, IEEE resealed the other variant of the standard, called 1000Base-T. This new version provides a new transceiver type to allow operation of GigE over the traditional UTP (copper lines), using digital signal processing techniques to compensate attenuation, distortion and the effect of crosstalk coming from other pairs within the same cable [25]. Thus, the challenge of getting the same data rate over copper lines as transmitting over optical fibre (1 Gbit/s) is overcome.

The connectors and sockets both variants are shown in Figure 2-8:

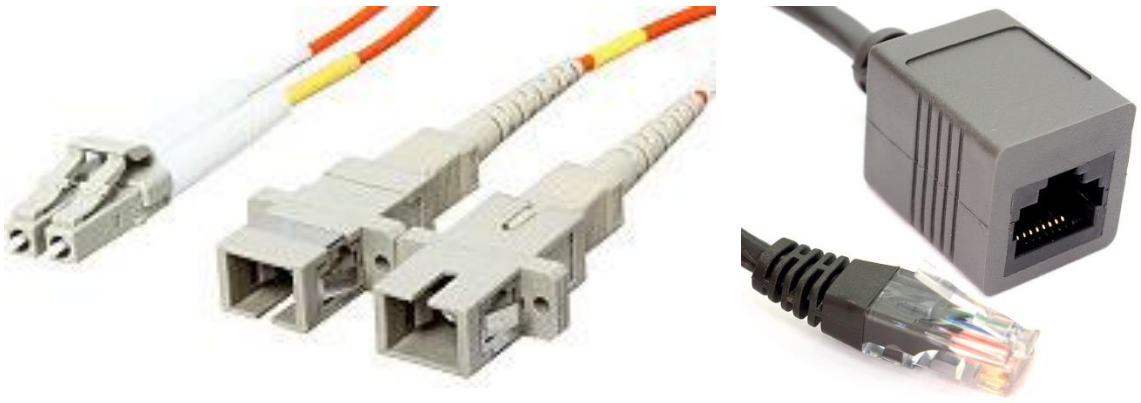


Figure 2-8. Connectors and sockets of 1000Base-X (left, optical fibre) and 1000Base-T (right, UTP) (image taken from [26]**)**

2.4.5 PCI Express (Peripheral Component Interconnect Express)

Presented as an evolution of PCI standard, PCI Express offers a number of improvements comparing to the original PCI.

The operation protocol of PCI express is based in a complex system of layers, where the physical layer is composed of a variable number of transmit and receive pairs (called lanes). The physical layer can provide x1, x2, x4, x8, x12, x16 or x32 lane widths depending on the number of lanes used in each system, splitting the data to be sent among these lanes [27].

Thus, in the case of the first version of PCI Express, being each lane able to give a data rate of 250 MB/s and taking into account that each byte is encoded in a 8b/10b format, the useful data rate achievable with an only lane is up to 200 MB/s per lane. That is, a single lane in the first version of the system was already able to transfer useful data at 1.6G bit/s. Then, this data rate can be multiplied increasing the number of lanes up to 32 [28].

In the case of following versions of PCI Express, the encoding scheme was improved, reducing the number of overhead bits in each transfer, and increasing the data rate of each lane.

The capabilities of each version is shown are shown in Table 2-4, stating the maximum data rate in terms of useful data (after removing overhead bits) for x1, x16 and x32 lane widths as a reference [28]:

Table 2-4. Versions and data rates of PCI Express

Version	Encoding System	Data Rate (Gbit/s)		
		1 lane	16 lanes	32 lanes
1.0	8b/10b	2	32	64
2.0	8b/10b	4	64	128
3.0	128b/130b	7.8	126	252

As it can be noticed in the table above, this communication system is hugely powerful in terms of throughput, offering also a great flexibility due to the

variable number of lanes. However, the implementation of this system is also very complex, what makes it very unreachable for this MSc project.

Two PCI connectors are shown in Figure 2-9. PCI Express x16 (top, 164 pins) and PCI Express x1 (bottom, 36 pins) connectors Figure 2-9:

Figure 2-9. PCI Express x16 (top, 164 pins) and PCI Express x1 (bottom, 36 pins) connectors (image taken from [29]**)**

2.4.6 RS-232:

Being the most common standard for serial transmission of data, its transmission rate is not defined in the standard (although it states that it is intended for data rates lower than 20 kbit/s) [30].

In serial communications, TTL voltage levels (internal levels of the digital electronics systems) are represented by ground level for logical 0 and Vcc level (power supply voltage of the system, typically 3.3V or 5V) for logical 1. On the other hand, RS-232 standard uses two voltage levels: -3V to -15V for logical 1 and 3V to 15V for logical 0. For this reason, when implementing a serial communication it is necessary to convert the signal levels from TTL to the chosen serial standard level. The chip in charge of this conversion is precisely the most important component involved in a digital system performing a RS-232 communication, and it is the key factor for determining the data rate of the system using the serial standard.

Although not being stated in the standard itself, it is widely known that, with appropriate hardware, RS-232 can perform data rates far beyond its initially

targeted value of 20 kbit/s. Thus, with a high performance converter, data rates of up to 2 Mbit/s can be achieved [31].

The greatest advantage of serial RS-232 standard is the ease for implementing it in a digital system. It includes some control signals for the communication, but only the TxD (transmission) and Rxd (Reception) lines are essential. Thanks to this simplicity, a RS-232 communication can be implemented in a FPGA by means of a serial transmission module, which sets the value of a certain output pin of the system alternatively to either logic 0 or 1 every period of the maximum frequency allowed, and a serial reception module, which “listens” to a certain input port to sample the received bits.

Figure 2-10 shows the typical DB9 connector used for RS-232 communications:

Figure 2-10. DB9 female (left) and male (right) for RS-232 communications (image taken from [32]**)**

2.4.7 RS-422

Standard compatible with the RS-232 version, being the use of differential signalling in RS-422 the main difference between them, this standard is especially robust for long distance transmission in noisy environments [33].

In differential signalling, a signal is sent over a pair of wires attached one to the other with opposite polarities. The resonant effect caused by the crosstalk noise might affect the quality of the signal [34], but thanks to the closeness of both differential wires they will presumably be affected in the same way, what means

that the voltage level between them should not be affected. Thus, decoding the values of a bit stream by means of the difference between the positive and the negative wires, the differential signalling gives a greater reliability compared to serial communication of bits with a single line. Thanks to this fact, RS-422 gives a greater reliability against errors in the transmission line compared to RS-232, allowing the serial communications to be performed over longer distances and with higher bit rates [35].

In the same way as RS-232, the RS-422 standard does not define the bit rates achievable with its use. However, with appropriate hardware, the standard has been proved to perform with a data rate of 10 Mbit/s over distances of 1200m [33].

RS-422 standard can be implemented using the same connectors as RS-232. For this purpose, two of the pins used for control signals must be used for the extra lines needed for the differential transmission of data.

2.5 Summary

The field of frame grabbers for image processing has been under constant development for the last decades. The operation of these devices can be separated in two separated stages:

- The process of collecting the video frames from the camera, saving them in the memory of the frame grabber and finally forwarding them to the computer (final endpoint).
- Once the frames are saved in the memory of the frame grabber, apply different image processing techniques to them.

The first stage is mandatory for any frame grabber, whilst the second one is dependent on the final application of the system. Due to the tight time constraints, the target for this MSc thesis is to develope a frame grabber that effectively collects and forwards video frames without processing them.

Being gas sensing the application field initially aimed, there is a wide range of applications demanding a frame grabber for collecting and processing video frames. Some of these are image monitoring of pick and place machines in manufacturing, medical imaging and stereo video (three-dimensional imaging).

The use of FPGA devices for the development of frame grabbers is the most suitable option, since it reduces the latency between frames thanks to its parallel implementation capability and high frequency operation. Besides, it is essential to operate at bit level and to manage the timings of the system in order to synchronise it with the protocol of each camera, which can only be done with an FPGA. A good proof of the suitability of FPGA devices for building frame grabbers is the large amount of previous work in the field and the commercial frame grabbers currently available, which are all based in FPGAs.

The throughput of the system is usually limited by the communication standard used for the transmission of the data from the frame grabber to the computer. Therefore, the selection of this standard is the most delicate issue to manage in the preliminary work. The maximum data rates achievable with each standard are shown in the Table 2-5:

Table 2-5. Summary of data rates of the main communication standards used in frame grabbers

Standard	Maximum data rate (Gbit/s)
SPI (SMPTE ST-2082)	12
HDMI 2.0	14.4
USB 3.1	10
Gigabit Ethernet	1
PCI Express 3.0 (32 lanes)	252
RS-232	0.002
RS-422	0.01

Regarding the difficulty of implementation, serial protocols are clearly the simplest ones. Among these, RS-232 is the simplest one because it only needs a transmission and a reception line to be implemented.

3 DESIGNED SYSTEM

3.1 Preliminary decisions

After the exhaustive research on the field done in the literature review, and taking into account other factors external to the project itself, such as the time available for the project and the background of the designer, the main decisions for the design are made.

First, considering the design based in FPGA as the only suitable option, the frame grabber system is decided to be implemented in an evaluation board instead of a commercial frame grabber. There are a few heavy reasons for this. The equipment of evaluation boards is far enough to achieve the targeted features, being cheaper than a commercial frame grabber with specialised hardware. Commercial frame grabbers are clearly oriented to the development of high performance systems, and the frame grabber needed for this project does not need such powerful features. Furthermore, the Group of Engineering Photonics of Cranfield University, where the project is developed, has experience in the use of a certain evaluation platform. This could undoubtedly mean a great asset, since the support of experienced designers could make a difference in case of getting to unforeseen difficulties.

Secondly, the camera outputs the images data in analogue format. Therefore, the FPGA will need to make use of an ADC peripheral for digitalising the analogue signal but, regrettably, this is not included in the selected evaluation board. For this reason, an external board compatible with the evaluation board is needed to perform this operation.

Finally, the most problematic point is addressed: the communication system used for the transmission of images from the frame grabber to the computer. As stated earlier in Section 2.3.1 (Related work), this is always a potential bottleneck for the system, and can convert a fast operating system into a slow system. However, the communication is very critical for the frame grabber, not only in terms of performance for the final system but also for the constant debugging that will be needed during the development. As a consequence, it is

paramount to deal with this issue as quickly as possible, in order to have time to develop the core of the frame grabber.

In this context, serial communication standards are clearly the easiest options. Then, RS-232 is selected because it does not require differential signalling, what would be an extra difficulty. Besides, USB capable RS-232 chips are widely available, making the connection to the PC very simple without any signal conditioning or amplification, as it would needed in RS-422. In order to facilitate the substitution of this communication standard in the future (once the frame grabber operation is validated) the transmission process performed by the serial line is thought to be set apart from the rest of the design.

3.2 General Description of the system and Equipment

The whole IR imaging system could be divided in three main parts: the Hamamatsu sensor (the IR camera), the computer where the IR images captured by the camera will be saved and displayed, and the electronic system allowing this operation between both endpoints, the so-called frame grabber. This basic view of the system is shown in Figure 3-1:

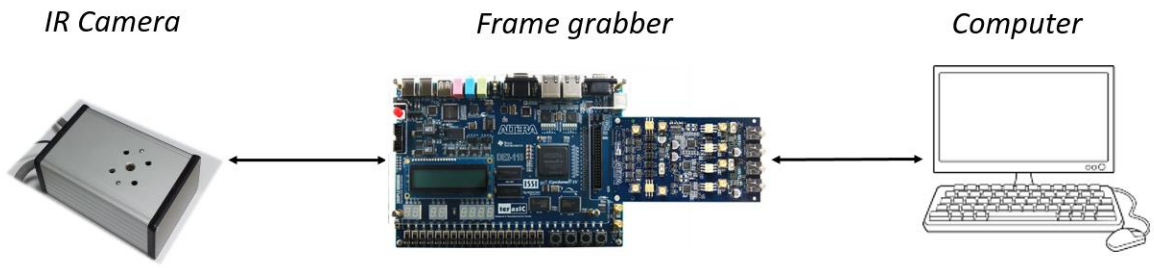


Figure 3-1. Overview of the IR imaging system

The work of this thesis is principally focused in the design and development of the frame grabber, although some time was also invested in the development of the Matlab scripts needed for taking the raw data from the frame grabber in the computer and then building the images from this data.

3.2.1 The Hamamatsu Image Sensor (G11097-0606S)

It provides 64x64 pixels infrared images, captured by means of a series of InGaAs photodiodes (one per pixel) with a spectral response range (wavelengths captured) from 0.95 µm to 1.7 µm. Each pixel's dimension are 50x50 µm, with a pixel pitch (distance between two adjacent pixels' centres) of 50 µm, what means an overall image size of 3.2x3.2 mm. The Hamamatsu Imaging Sensor is shown in Figure 3-2:

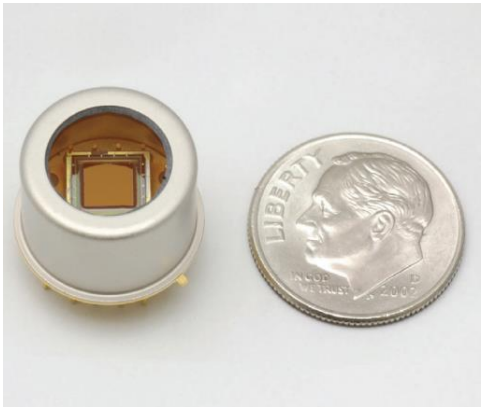


Figure 3-2. Hamamatsu sensor (image from [36]**)**

The captured pixels are transmitted by means of an analogue video output, which is triggered with a digital signal and easily obtained by just supplying a master clock and master start pulse from external digital inputs [36].

It is extremely important to understand this operation, since the whole system is based on these control signals. The signals involved in this communication are listed below:

- MSP (Master Start Pulse, digital input): also called integration signal, it is used for indicating the camera when it should start the process of capturing and sending an image. This order is given to the camera by means of a low level pulse in the signal. As soon as the camera detects a low level in the MSP input, it starts capturing a new image, and keeps the lens capturing until the MSP input is released (until it comes back to high level). Therefore, the duration in time of the MSP low pulse, the so-

called integration time, states the exposure time for capturing the new frame.

- MCLK (Master Clock, digital input): the reference clock for the camera, up to 40 MHz.
- VIDEO (video signal, analogue output): Right after the end of the image capture, the camera starts to send the pixels' analogue values, one by one, throughout the analogue VIDEO output at a frequency rate of up to 5 MHz (the frequency of the clock divided by 8). It is important to highlight that this is an analogue signal and, therefore, it needs to be digitalised in order to be received in the FPGA.
- AD_TRIG (trigger, digital output): this video trigger digital output is given to the frame grabber along with the VIDEO analogue output, and provides a falling edge every time a new pixel value is stable to be read.

The transmission process is arranged in an accurate way, alternating pixels' transmission with a set of blank spaces, previous to the first pixel of the frame, after the last one and between rows of the image (blank spaces after each set of 64 pixels). This is illustrated in Figure 3-3, which is a timing diagram taken from the datasheet of the IR camera:

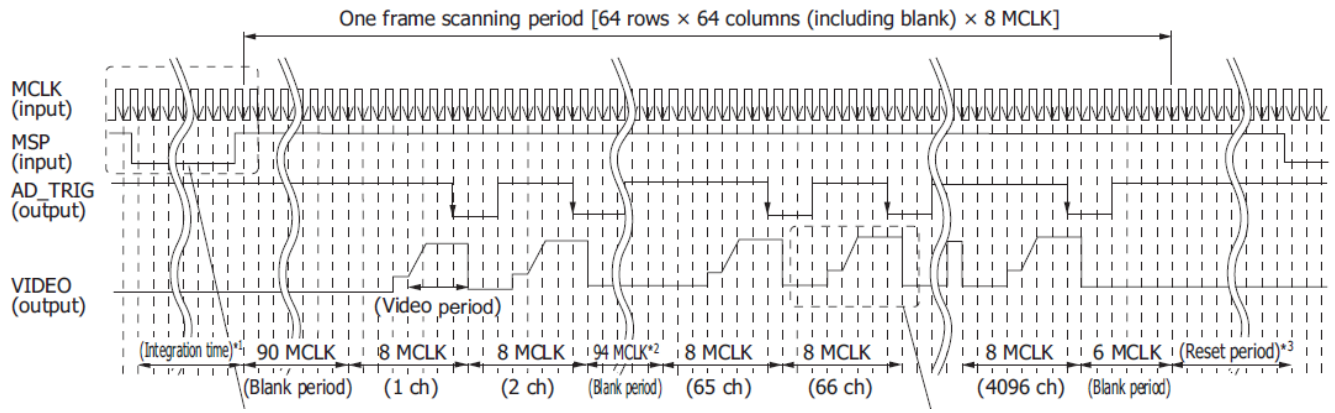


Figure 3-3. Timing diagram of the camera's signals (image taken from [36]**)**

3.2.2 The frame grabber

It is the electronic system in charge of controlling the whole imaging system. It allows the configuration of the operation mode and other parameters from the computer, and afterwards it takes the images from the camera for sending them to the computer. It is designed with two different operation modes, plus a third mode for calibration purposes:

1. Live mode: it takes a frame from the camera, saves it in a SRAM and sends it to the computer. It repeats this process iteratively, creating a real time video stream to be displayed in the computer.
2. Buffer mode: it captures at very high speed a certain number of images, storing them all in a SRAM. Once finished taking the images from the camera, it transmits them all to the computer.
3. Calibration mode: it takes up to 25 consecutive values of the digitalised VIDEO signal around the falling edge of the trigger signal, in order to calculate the delays between both signals. It is used for the calibration of the system, making possible to find the perfect moment to capture each pixel.

For the development of the frame grabber, two electronic boards are used:

1. Altera DE2-115 Evaluation Board: based in an Altera Cyclone IV FPGA, where the main part of the frame grabber will be implemented by means of VHDL language. It also includes different peripherals needed for the operation of the system, such as a SRAM, GPIO ports or a HSMC connector for plugging a data acquisition card that will allow the digitalisation of the analogue data coming from the camera. The DE2-115 board is shown in Figure 3-4:

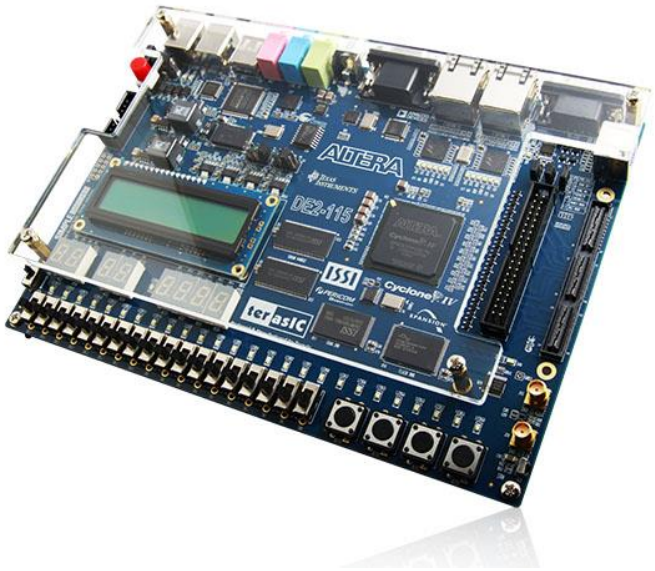


Figure 3-4. Altera DE2-115 Evaluation Board (image taken from [37]**)**

2. AD/DA Data Conversion Card: this is the Data Acquisition Card, equipped with the ADC used remotely by the FPGA to digitise the analogue data coming from the camera in order to process it. The board is shown in Figure 3-5:

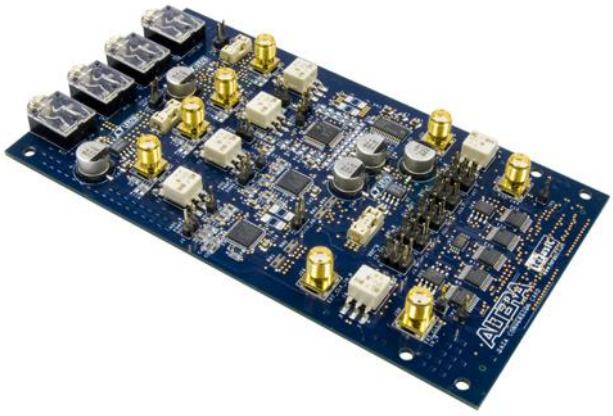


Figure 3-5. AD/DA Data Conversion Card (image taken from [38]**)**

3.2.3 Computer

No major developments were needed for the computer. The data is transmitted from the frame grabber to the computer using the RS-232 standard, and the configuration of the frame grabber is set sending the parameters to the frame grabber using the same standard. For this purpose, a Serial Protocols emulator (Docklight) was used. Apart from this, a Matlab script for decoding raw the data coming from the frame grabber and building the final images from this data was developed.

3.2.4 Other hardware

The IR camera could not be connected directly to the Evaluation Board and the Data Acquisition Card. Some adaptation work must be done first. On the one hand, the voltage levels of the signals for the communication between the camera and the frame grabber need to be adapted to the inputs of the frame grabber. This is managed in an adaptation board attached to the image sensor.

On the other hand, some aspects which are secondary for the communication but essential to allow camera's operation, such as power supply and temperature control, must be also managed with a great deal of care. For this purpose, the camera is connected to another board. The main blocks of the resulting system are shown in the diagram of Figure 3-6:

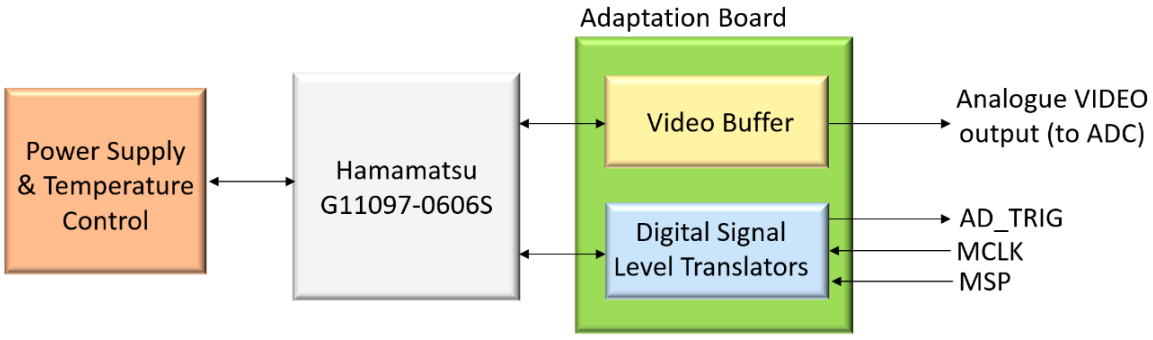


Figure 3-6. Blocks diagram of the adaptation of Hamamatsu imaging sensor to the overall system

First, the board for power supply and temperature control is packed in a box separated to the test of the system. This box is connected to the electrical power supply of the laboratory, using this to create the voltage levels needed to power up the camera. Besides, it also performs the needed temperature control mechanisms using the signals provided by the camera for this purpose[1].

Regarding the adaptation board, in the Digital Signal Level Translators it converts the voltage levels of the digital signals of the camera to the levels needed in the frame grabber. The Video Buffer, which is an amplifier configured for an amplification of 1, decouples the VIDEO output from the Hamamatsu module from the cable and receiver.

Then, both the camera and the adaptation board are packed into a protective case, getting to the box shown in Figure 3-7. The box includes 4 holes around the image sensor that will be used for fixing a system of lenses for getting images once the frame grabber is working.

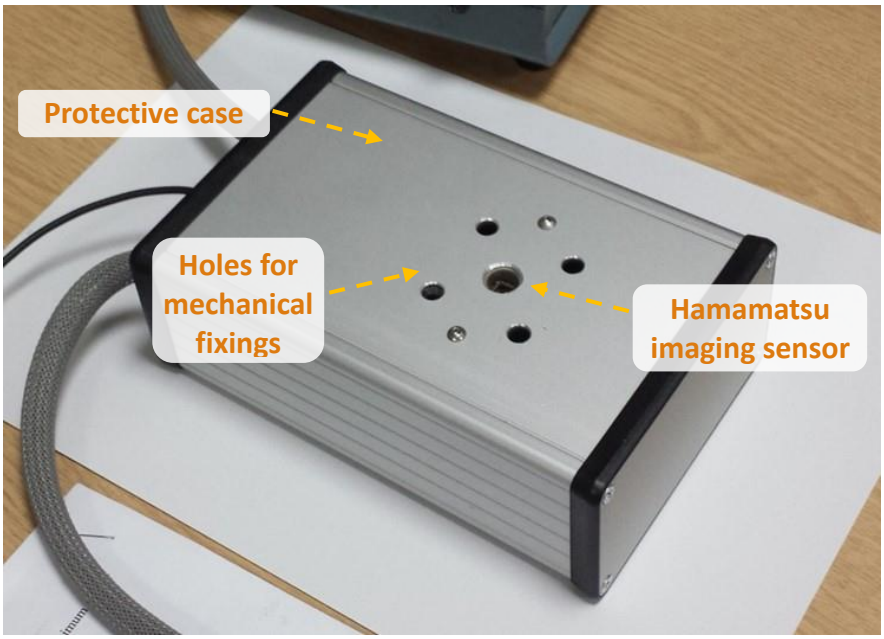


Figure 3-7. Protective case containing the IR camera chip and the adaptation board

[1] All the adaptation work was done by Dr Steve Staines and Dr Thomas Kissinger

Regarding the communication between the frame grabber and the computer, although the DE2-115 includes the DB9 connector typically used for RS-232 communications, another external small platform is used. The serial communication lines (TX and RX) of the system are mapped to the GPIO ports of the Evaluation Board, and these are connected to the external platform. Thus, thanks to the high speed driver included in this platform for the conversion of the signals coming from the FPGA (TTL) to RS-232 levels, the serial data transmission is increased to a rate of 1 Mbps. If needed, this driver could operate at a rate of up to 2 Mbps. The external platform for TTL to RS-232 conversion is shown in Figure 3-8:

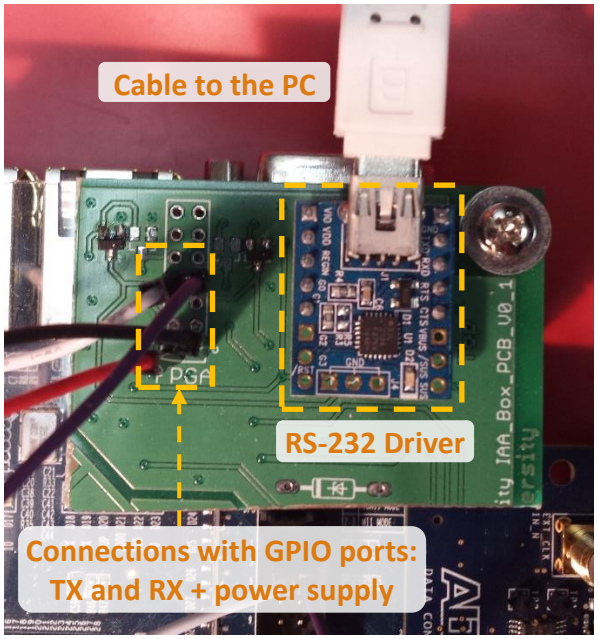


Figure 3-8. External platform affixed in a corner of the Evaluation Board

As seen in the picture, the driver is directly connected to the PC by means of a micro-USB cable.

3.3 Architecture of the IR imaging system

The general architecture of the overall system is shown in Figure 3-9:

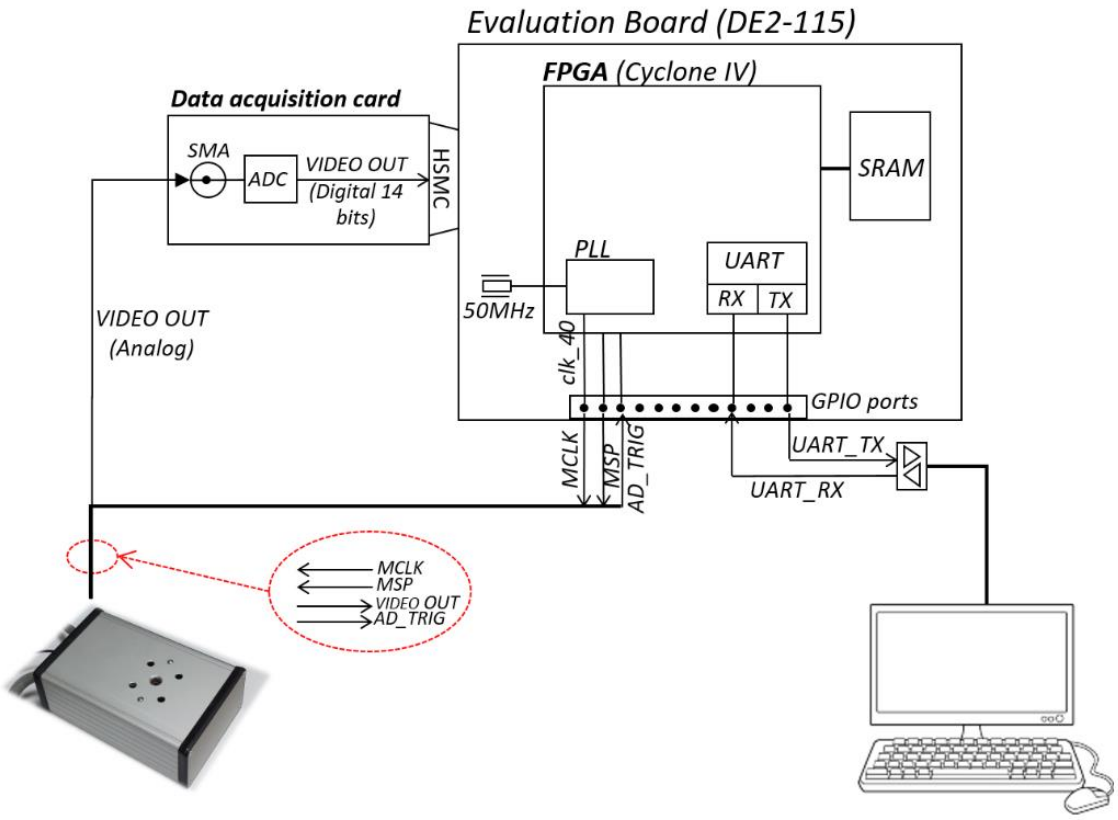


Figure 3-9. Architecture of the IR imaging system

As stated above, there are two endpoints in the overall imaging system: the camera (shown at the left of image) and the computer (shown at the right).

The control and data signals from the camera are listed in the red ellipse, connected to the frame grabber after the adaptation explained in Section 3.2.4 (Other hardware). In the particular case of the VIDEO output analogue signal, the adaptation board outputs it over a coaxial cable. Thus, the signal is received in the Data Acquisition Card through its SMA connector, passing it to the ADC. In the ADC, the analogue signal is digitalised into a 14 bits digital signal, which

is transmitted to the Evaluation Board by means of the HSMC connector. These connections are shown in Figure 3-10:

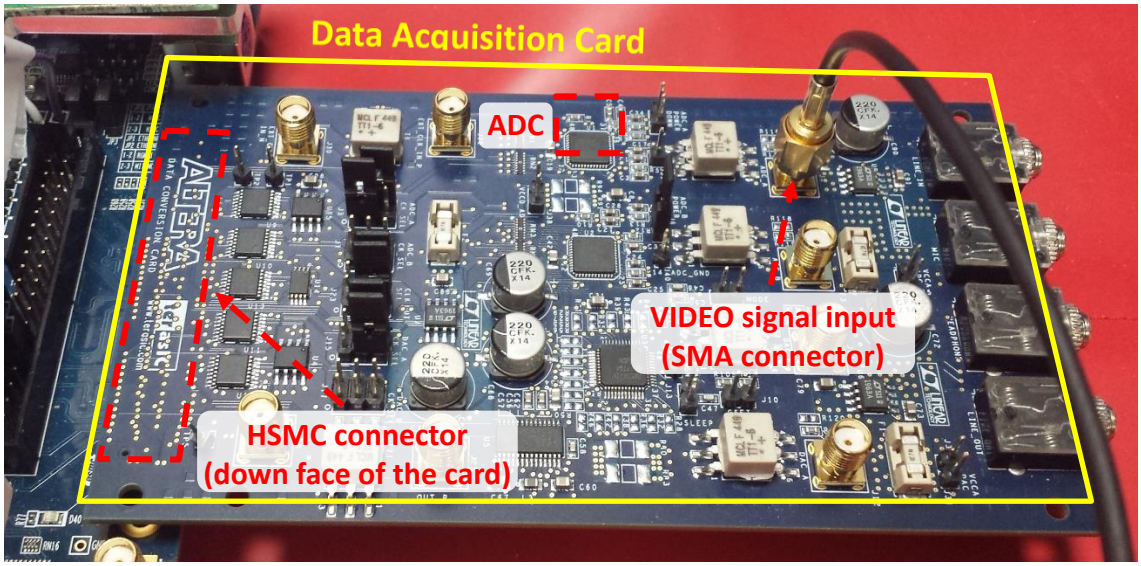


Figure 3-10. Connections in the Data Acquisition Card

Once the digital VIDEO output is introduced in the Evaluation Board, it is easily received in the FPGA to process it. The value of the analogue VIDEO output is digitalised and transmitted to the FPGA every clock period. This is the reason why the AD_TRIG signal is very important, because it states when the VIDEO data is giving the value of a pixel, once every 8 clock cycles.

Regarding the control signals of the camera, which are already digital, they are directly connected to the GPIO ports of the Evaluation Board (see Figure 3-11). One of those signals is MCLK, the reference clock provided to the camera, which should be not faster than 40 MHz. However, the oscillator of the Evaluation Board provides the system with a 50 MHz clock. For this reason, it is necessary to implement a PLL within the FPGA in order to create the 40 MHz clock needed for synchronising the whole system.

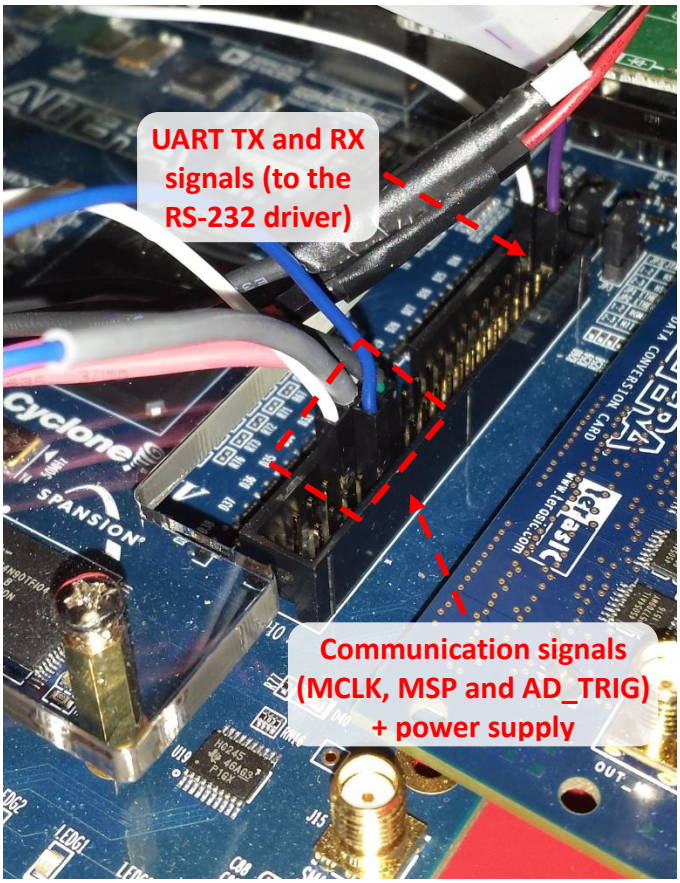


Figure 3-11. GPIO ports connections

Once inside the FPGA, the pixels data coming from the VIDEO output of the camera are saved in the SRAM included in the Evaluation Board. Afterwards, depending on the operation mode, this data is processed in diverse ways, being always read from the SRAM and sent to the computer through the UART module, which is implemented within the FPGA.

Finally, the data managed by the UART module is converted to RS-232 levels. The UART module is used for both transmitting the video data to the computer and for receiving the configuration parameters from the computer before starting the data transmission.

4 DEVELOPMENT OF THE SYSTEM

The development of the system has been divided in four different stages, which are formed by several subtasks:

- Building Blocks.
- Developing a basic version of the system able to capture an only frame and to transmit it to the computer.
- Creating the Live and Buffer operating modes, based in the previous basic version.
- Adapting the design to the actual camera, including the creation of the third operation mode, the Calibration Mode.

In this section, the most important aspects of the development are stated, along with the technical details involved in each of the stages.

4.1 Building Blocks

Provided the great deal of hardware involved in the design, before starting the development of the system, it was important to have a clear view of the operation of every single component. For this reason, the next modules were developed:

4.1.1 SRAM read/write module.

A basic module for testing the operation of the memory that would be used for saving the pixels as they are received from the camera, and to read this data in order to transmit it to the computer. This might look an easy task, but there are a few signals to deal with, and the operation protocol is not stated anywhere. For this reason, it was necessary to invest some time in this apparently simple module.

4.1.2 UART modules.

Including both data reception and transmission modules, the development of this modules was paramount for the communication of the frame grabber with

the computer, for both receiving the configuration parameters from the computer and sending the images' data to the computer[2].

4.1.3 Mirror test for the Data Acquisition Card (ADC and HSMC connection test).

For testing the operation of the Data Acquisition Card, although only the ADC of the card was supposed to be used in the final design, a mirror test involving both the ADC and DAC was performed. This test was, basically, inputting to the ADC an analogue signal directly created with a functions generator, receiving the digitalised data in the FPGA by means of the HSMC connection between the Data Acquisition Card and the Evaluation Board, and finally outputting the signal in the opposite way. That means, sending the digitalised signal received in the FPGA back to the Data Acquisition Card through the HSMC connection, this time to the DAC, which would convert the signal from digital to analogue again and output it from the system. This mirrored signal is analysed with an oscilloscope, which plots the signal with the same shape as the signal generated by the functions generator.

4.1.4 Hamamatsu Debug Module.

Due to a delay in the delivery of the Hamamatsu image sensor, the operation of the frame grabber could not be physically tested until the last stage of the thesis. During the first stages of the design, all the process was tested and debugged by means of the simulation tools provided for the FPGA. Regrettably, only the operations performed within the FPGA could be analysed with this method, and provided that the system manages some external peripherals, the final operation of the whole system could not be tested by means of the simulation. Therefore, it was necessary to find a mechanism to perform a physical debug of the system. Thus, a module that recreates the operation of the camera was developed in VHDL code and integrated within the FPGA. The pinout of the module, which includes the signals presented in Section 3.2.1, is shown in Figure 4-1:

[2] UART RX module based in a previous work by Dr. Thomas Kissinger

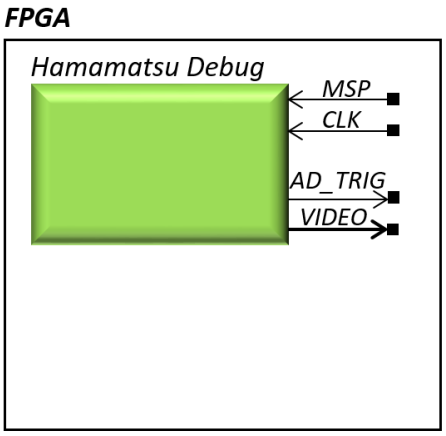


Figure 4-1. Pinout of the Hamamatsu Debug Module

This module performs a communication with the frame grabber system (implemented both endpoints within the FPGA), based in the operation protocol used by the actual IR camera. The module recreates only the signals of the camera that are involved in the communication process (MCLK, MSP, VIDEO and AD_TRIG), ignoring the signals dealing with secondary aspects (power supply and temperature control).

The main constraint of this physical debugging method is that, due to the impossibility of recreating the analogue VIDEO signal within the FPGA, the digitalisation of the signal by means of the Data Acquisition Card cannot be tested this way. Therefore, the VIDEO signal is provided as it would be after the digitalisation, with a digital bus of 14 bits. As real values are not needed for the debug, a sequence of numbers from 0 to 4095 (for the 4096 pixels of the image) is given as each image's pixels values.

Thanks to the mirror test performed for testing the correct operation of the Data Acquisition Card (see section 4.1.3), this part of the system is considered to be a non-problematic factor to be implemented afterwards in the final version of the system, once the camera is available.

The operation of this VHDL module is shown in Figure 4-2, which gives a view of the simulation of the behaviour of the module when it receives the MSP signal:

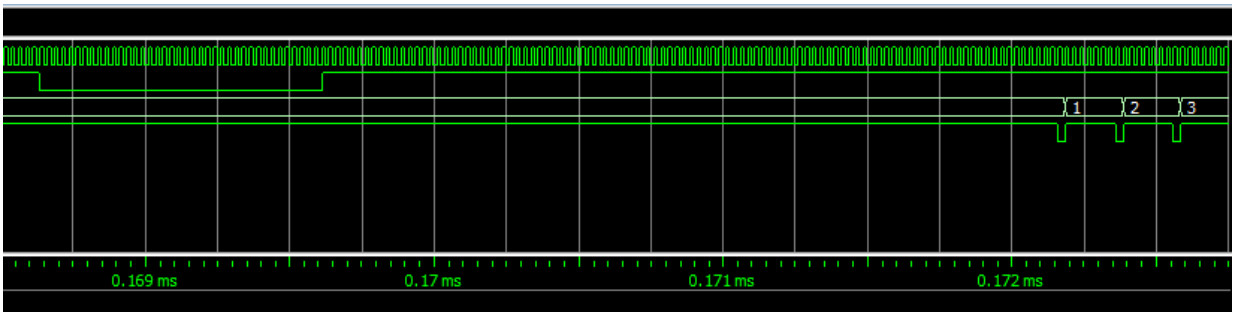


Figure 4-2. Simulation of the Hamamatsu Debug Module

In this simulation it is shown how the Hamamatsu Debug Module receives the MSP low pulse (second line) and waits for the first blank period to expire before starting to send the pixel values (the increasing sequence in the third line, already in digital format). It gives a falling edge in the AD_TRIG (fourth line) in certain points where the value of the pixel is supposed to be stable.

4.2 Basic version

Once all the peripherals of the system have been tested, the first basic version of the system is developed. This is the first stage in the actual development of the system and, provided that all the posterior design is based upon this first basic version, it is also the most important stage.

The first point in this development is to define the flow diagram of the system, which is transferred to the finite-state machine within the FPGA. The finite-states machine will rule the operation of the system according to the timings set by the camera's operation protocol. A basic representation of the flow diagram is presented in Figure 4-3 (for further details, see the full flow diagram in Appendix A (Flow Diagrams):

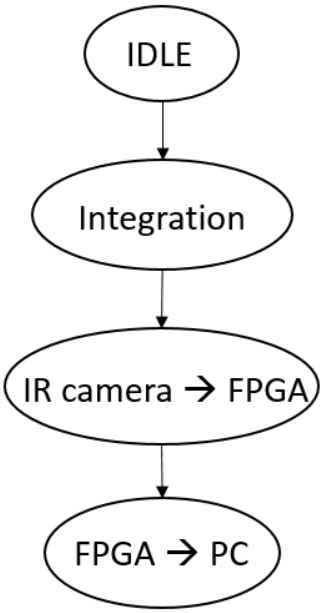


Figure 4-3. Flow diagram of the basic version of the system

The operation of the basic system is based in three main stages: integration time, collecting the data from the camera in the frame grabber and sending the collected data to the computer. In the first state (IDLE), the system is simply waiting for the order from the user to start the process, order that it receives when a button of the Evaluation Board is pressed.

4.2.1 Integration Time

The first step in the operation of the imaging system. The only action performed in this state is setting the value of the MSP signal to 0. Thus, the time this state is run sets the exposure time for the capture of the image by the camera, which in the basic version of the system is a fixed value. The capture of the image by the camera ends right in the same moment in which the MSP comes back to high level.

4.2.2 Collecting the image‘s data in the frame grabber

After capturing the image, the camera waits for a certain blank period of time to expire and then starts to send the pixels one by one through the VIDEO output analogue signal to the frame grabber, along with the AD_TRIG trigger signal. A

schematic view of this process is given in Figure 4-4, with only the first two pixels out of 4096 in each image represented in the diagram:

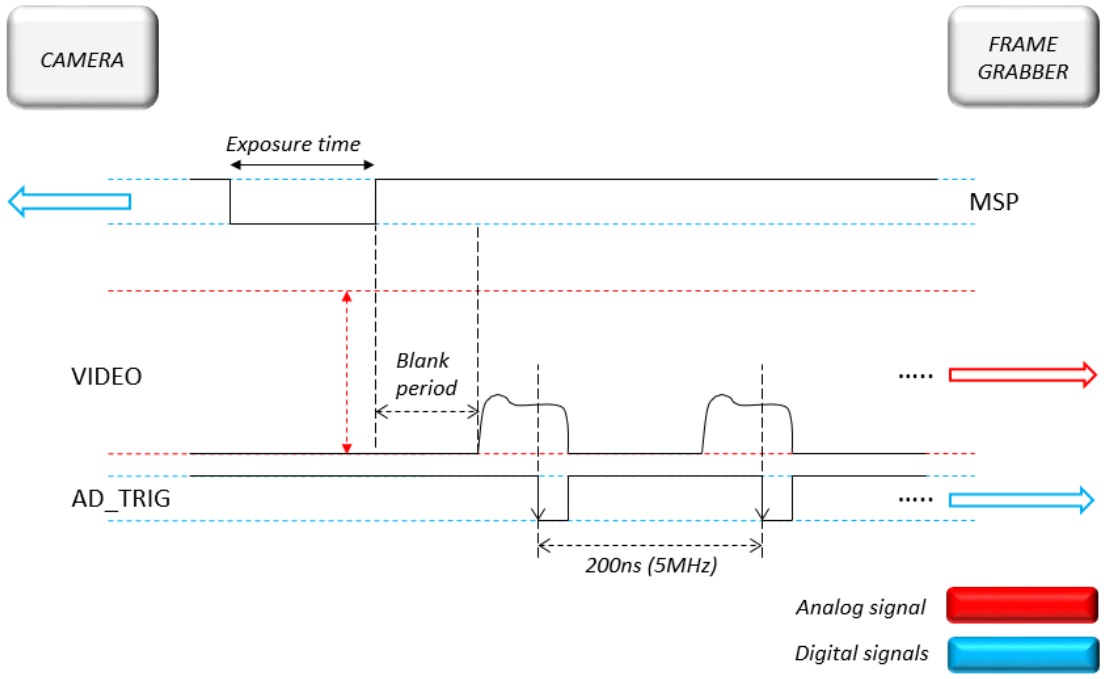


Figure 4-4. Beginning of the communication between the camera and the frame grabber

Every falling edge of the AD_TRIG, a new pixel value is taken in the FPGA. This value is aimed to be a digitalised version (14 bits) of the analogue VIDEO output of the camera, but in the first stages of the development this 14 bits value will be directly taken from the Hamamatsu Debug Module's VIDEO digital output.

Once the pixel value gets to the FPGA, it is initially saved in an internal register. Then, it will be saved in the first position available in the SRAM memory of the Evaluation Board. Being the SRAM cells 16 bits wide, the pixel value must be first filled with two extra bits in the most significant bits of the word. Following an iterative process, the SRAM is filled with the pixel values in consecutive positions, from the first position of the memory to the 4095th position, the one corresponding to the last pixel of the image. This process is shown in Figure 4-5, where the values saved in the memory are the ones received from the Hamamatsu Debug Module: an increasing binary sequence of numbers from 0

to 4095. When reading the pixels from the real camera these values are obviously random.

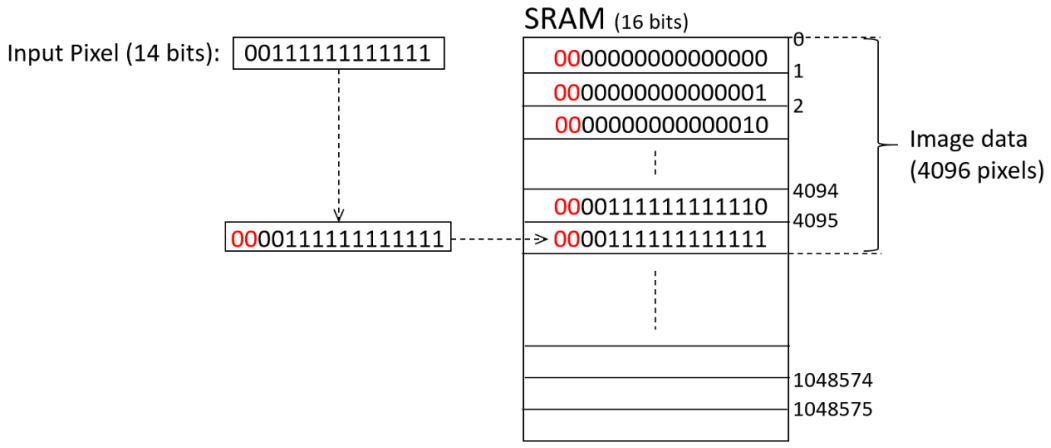


Figure 4-5. Saving the last pixel of the image in its corresponding position (4095th) in the SRAM memory

4.2.3 Transmitting the image's data

The last step of this basic version of the frame grabber system is taking the image's data saved in the SRAM and sending it to the computer. For this purpose, the words of the SRAM (16 bits) are passed one by one to the UART TX module, which is in charge of transmitting the data at a rate of 1Mbps to the computer by means of the external platform mentioned in Section 3.2.4.

Once the pixel, made by 14 useful bits plus two extra 0 bits, is in the UART TX module, the next procedure is followed.

In order to understand the procedure of the UART TX module, it is essential to understand previously that RS-232 is an asynchronous communication standard. This means that it does not transmit any clock for the synchronisation of both endpoints within the cable. Therefore, it is necessary to arrange the data in sets of 8 bits and to add a bit stop and a bit start at the beginning and the end of each byte (8 bits), respectively, in order to let the receptor know when a data byte is beginning and finishing. For this reason, each data byte must be transmitted within a set of 10 bits.

Regarding the pixel values to be transmitted, they are made by 14 bits plus two extra bits in the SRAM, which are discarded within the UART TX module. Then, 14 bits must be arranged in two separated bytes for the transmission of the pixel value. For this purpose, the set of 14 bits is split in two sets of 7 bits, leaving a spare bit in each byte to be transmitted.

Taking advantage of this spare bit, an integrity check and synchronisation method is implemented in the frame grabber. This bit, called Frame Begin bit, will be 1 when the transmitted byte is the first byte of the first pixel of each frame. This mechanism will be used in the receiver to check when a new frame is received, which will be particularly useful in the future operation modes of the system, where frames will be sent sequentially one after another.

This process is illustrated in Figure 4-6, with the SRAM memory filled in the previous stage, represented this time with random values (as it would look like in the real system) instead of a sequence of numbers from the Hamamatsu Debug Module:

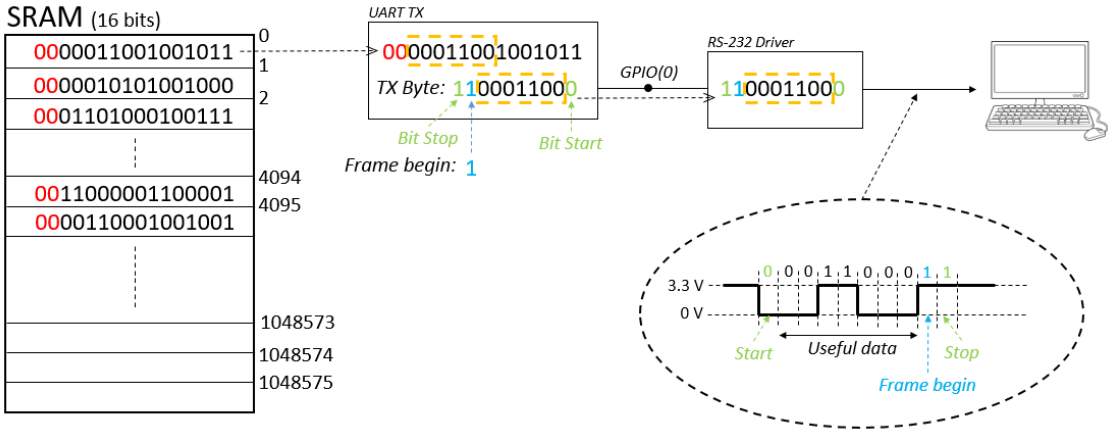


Figure 4-6. Transmission of the first pixel of the image (first byte)

The byte is transmitted from the LSB to the MSB. For this reason, the bit start is placed at the end of the byte (right after the LSB) and the stop bit at the beginning of the byte (before the MSB). Once the set of 10 bits is created, it is sent to the RS-232 Driver in the external platform mentioned through the GPIO pin #0. There, the driver converts the FPGA's internal voltage levels (TTL) to the levels required for the RS-232 serial line, creating the physical sequence of

bits shown inside the ellipse and sending it to the computer throughout the serial line.

After the transmission of the first byte of the pixel, the second byte of the same pixel is sent following the same process, with the only difference of the Frame Begin bit, whose value is now set to 0. This difference is shown in Figure 4-7:

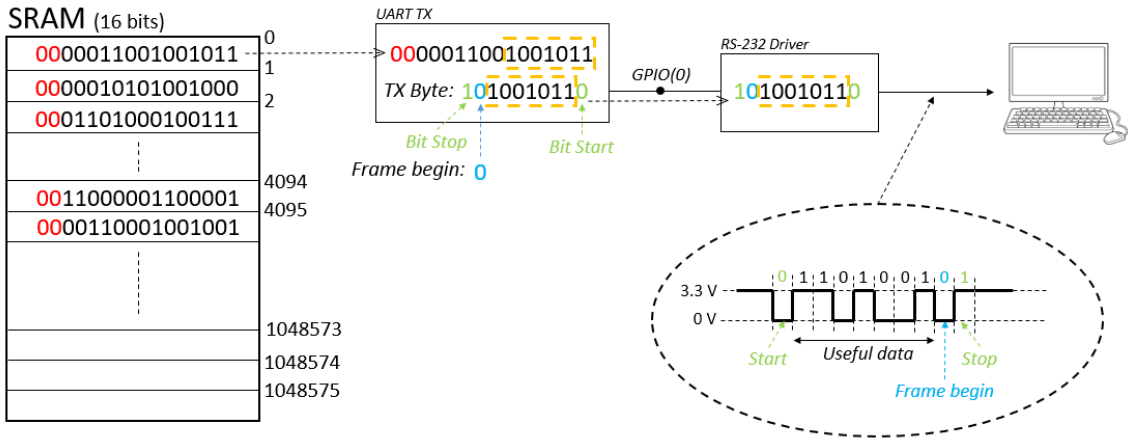


Figure 4-7. Transmission of the second pixel of the image (second byte)

After this, the process of transmission of a pixel will be repeated iteratively another 4095 times (one iteration for each remaining pixel), with the Frame Begin bit set always to 0.

4.3 Live and Buffer modes

In this stage, the basic design presented in Section 4.2 is used as a base for the first version of the fully operating system. Like in the development of the basic version, during the development of this stage the system is debugged by means of the simulation tool and physically tested using the Hamamatsu Debug Module.

In this version of the system, the frame grabber already incorporates most of the functionalities of the final system, with two operation modes already implemented: Live Mode and Buffer Mode.

Another important functionality added to the system in this stage is the capability to get the parameters configuration from the final user, in order to set the operation of the frame grabber as he wishes in any moment.

Both the new operation modes and the process of configuring the parameters are presented in this section.

4.3.1 Parameters configuration

Before starting to take any image from the camera, it is necessary to configure the frame grabber with a certain set of parameters that define its operation. These parameters are:

- Operation mode of the frame grabber: two operation modes available for this version of the system: Live Mode and Buffer Mode.
- Integration Time: the time length of the MSP low pulse that the frame grabber must send to the camera to indicate the beginning of the image capture and transmission process. As stated in Section 3.2.1, this time sets also the exposure time for the image capture in the camera.
- Buffer Size: number of images to be stored in the SRAM in the Buffer Mode.
- FPS. Frames per second rate for the transmission of a stream of images from the frame grabber to the computer in both Live Mode and Buffer Mode.

This configuration must be set by the user from the computer and, therefore, it is implemented by means of the serial cable used for the transmission of the images from the frame grabber to the computer. This cable is bidirectional, including a serial line for each direction. For the reception of data coming from the computer, the UART RX module was developed.

In order to allow an easy access and modification of the parameters, their values are saved in internal registers of the FPGA, which are identified each with a different address. Therefore, the configuration of the parameters is based in an address-value duality, firstly waiting in the FPGA for the data coming from the computer to identify the parameter to be configured (the address of the

register), and secondly listening for the value to be saved in the selected register.

The process of configuration is illustrated in Figure 4-8:

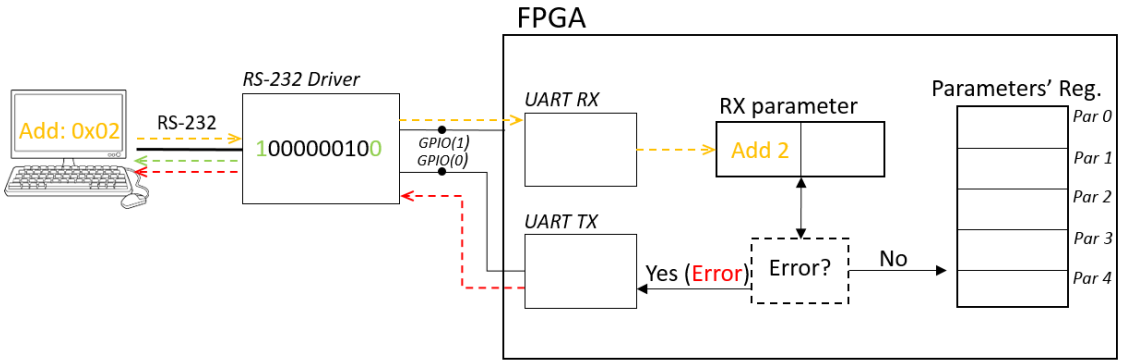


Figure 4-8. Configuring the parameter with address 0x02 (1)

As stated above, first the address of the targeted parameter is sent from the computer to the frame grabber by means of the serial lines using the standard RS-232. Then, the data enters the UART RX module within the FPGA throughout the RS-232 driver and the GPIO pin #1, following the inverse process as for the transmission of an image from the frame grabber to the computer (see Section 4.2.3), this time in the opposite direction. Then, the UART RX takes the received byte from the serial line in digital format and saves it in the FPGA, saving it as the address within the dual structure of the parameter.

Then, the system checks the received address and, in case of being wrong (non-existent parameter), it sends an error code back to the computer by means of the UART TX module following the transmission process explained earlier (see Section 4.2.3). In case of being a wrong value, the system comes back to the previous point, listening again for an address. Otherwise, it waits for the selected parameter's value.

Once the address of the incoming parameter is saved in the FPGA, it starts to listen to the corresponding value, taking as many bytes as needed for filling the

value of the selected parameter, since the parameters can be from one to four bytes long. Therefore, the frame grabber will iterate the byte reception as many times as needed depending on which parameter has been selected by means of the address. The process is shown in Figure 4-9:

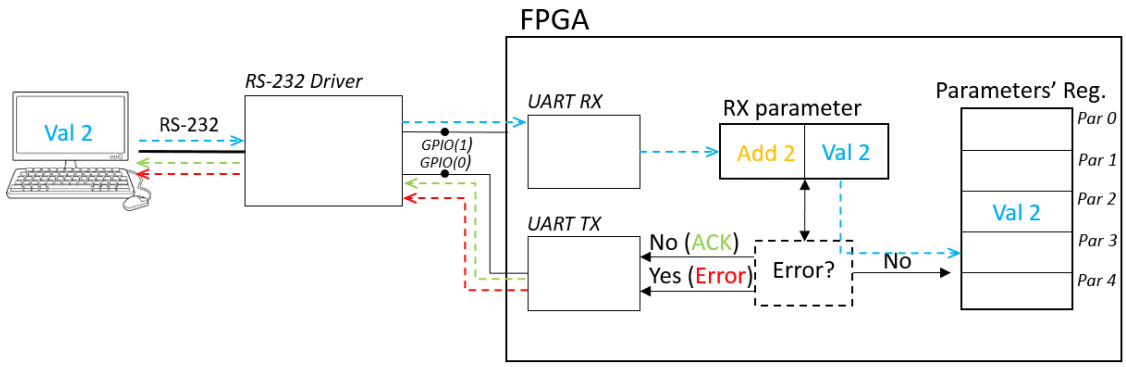


Figure 4-9. Configuring the parameter with address 0x02 (2)

After having iterated as many times as needed for taking all the bytes of the parameter value, the frame grabber checks the validity of the value for this specific parameter. If it is wrong, it sends the corresponding error code to the computer and starts to listen again to the same parameter's value (it keeps the address previously received). If the value is correct, the frame grabber sends an acknowledgment code (ACK) and saves the value in its corresponding register.

This process of configuration is followed until every parameter has been configured, and it can be done in any order. The only special parameter is the START parameter (the number 0). This parameter must be set to 0x01 in order to start the image capturing operation, and it must be done when the rest of parameters have already been configured. In that case, the process gets started. Otherwise, the frame grabber will respond to the computer with a specific error code.

The addresses and allowed values for the parameters are stated in Table 4-1, both of them expressed in hexadecimal code to facilitate the visualisation.

Table 4-1. Parameters' addresses and allowed values

Parameter name	Address	Value
Start	0x00	0x01
Operation Mode	0x01	0x01: Live Mode
		0x02: Buffer Mode
Integration Time	0x02	0x000028 – 0x061A80
Buffer Size	0x03	0x01 – 0xFE
FPS	0x04	0x00320009 – 0x02625A00

The range of values of the integration time correspond to the maximum and minimum exposure time allowed by the camera (0.001ms to 10ms) in terms of number of clock cycles of the system (clock of 40MHz).

In the case of the buffer size, which states the number of frames to be stored in the SRAM memory before transmitting them in the Buffer Mode, the minimum is one frame and the maximum is 254 frames (limited by the size of the SRAM).

Finally, the FPS parameter sets the number of clock cycles that the transmission of any frame should take in order to get to a certain FPS rate. This value is calculated with the next equation:

$$FPS\ atribute\ value = \frac{clock\ frequency}{desired\ FPS\ rate} = \frac{40 \cdot 10^6}{desired\ FPS\ rate} \quad \textbf{(4-1)}$$

Taking into account the influence of the variation in the integration time in the overall time for capturing and transmitting every image, the maximum FPS rates corresponding to the minimum and maximum integration times stated above are, respectively, 12.21 fps and 10.76 fps. However, it was also interesting to allow the user to set rates as low as 1 fps. Therefore, the allowed range for this

parameter is from the value corresponding to 12.21 fps (0x00320009) to the value corresponding to 1 fps (0x02625A00).

Regarding the responses of the frame grabber to the computer, the codes' meanings are shown in Table 4-2:

Table 4-2. Error and acknowledgement codes

Code	Meaning
0xAA7A	Acknowledgement (sent after the correct configuration of any parameter)
0xDD7D	Error 1: incomplete configuration (sent when START parameter is activated before configuring the rest of parameters)
0xEE7E	Error 2: wrong value (sent when the stated value is out of range)
0xFF7F	Error 3: wrong address (sent when the stated parameter number does not exist)

There is also a global RESET code (0xFF) that, apart from stopping the operation wherever it is to come back to IDLE state, it also deletes the configured value of any parameter. This is the only code that can be received in any operation point of the frame grabber system.

4.3.2 Live mode

Taken the first basic version of the system (see Section 4.2), it is only necessary to introduce minor changes in the flow diagram to create the Live Mode. In this stage, the parameters' configuration explained in the previous section is introduced as a new state in the finite-state machine of the system, right after the IDLE state. Thus, the way of starting the system is changed from simply pressing a button for running it with default configuration parameters to a configuration process fully controlled by the user of the frame grabber from the computer. Apart from this, the whole process of the system is constantly repeated in a loop, starting again in the integration time state every time the system finishes the transmission of an image. This new flow diagram is represented in Figure 4-10:

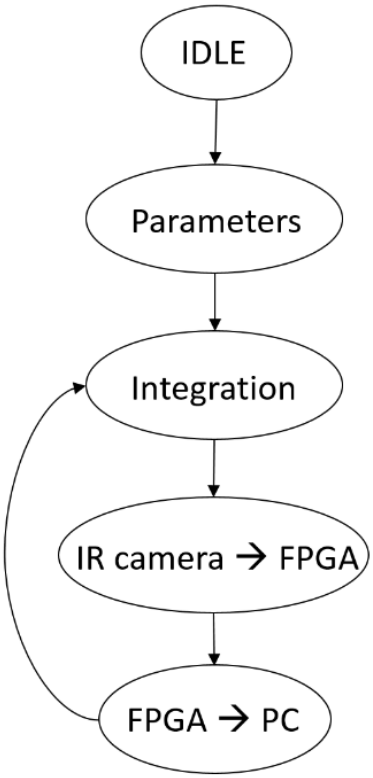


Figure 4-10. Flow diagram of the Live Mode

With this new operation mode, the system runs continuously with the initial parameters configuration: it receives an image from the camera, saving it pixel by pixel in the SRAM, and right after saving the last pixel in the SRAM it starts to transmit this image. Once the last pixel of the image has been sent, the system comes back to the integration state, asking the camera for a new image by means of the MSP signal and, afterwards, overwriting the previous image in the same positions of the SRAM (from memory cell number 0 to 4095). This process is run at the frame rate specified in the FPS parameter, which can be tightly fitted to the estimated minimum times for the whole process or set to a more conservative frame rate that could allow the user to take the frames of this video stream in an easier way.

In case of needing to change any of the parameters once the system is already running, the user must use the global reset (send the code 0xFF from the computer to the frame grabber) and reconfigure all the parameters again.

In this operation mode, the maximum frame rates achievable depend only on the configured exposure time for the capture of the images, since the rest of factors with any influence in this matter are fixed by the Hamamatsu camera. It is important to remember that the exposure time is stated by the length of the integration signal (the MSP low pulse). Thus, the time it takes a video frame (an image) to be captured and to traverse the whole system can be split in different stages, as shown in Table 4-3:

Table 4-3. Timings in Live Mode

Exposure Time	0.001ms to 10ms
Image reception from the camera @ 5 MHz (1 pixel of 14 bits every 200 ns)	974.65 µs
Image transmission from the frame grabber to the PC @ 1 Mbit/s	81.92 ms
TOTAL TIME PER FRAME	E.T. + 82.89 ms

Looking at these timings, one of the conclusions obtained in the literature review is confirmed: the transmission from the frame grabber to the computer brings the main bottleneck of the system. In fact, the time it takes to transmit a single frame represents from 88.19% (10 ms of exposure time) to 98.82% (0.001 ms of exposure time) of the total time the frame need to be captured by the camera, received in the frame grabber and sent to the computer.

Taking into account these timings, it is easy to calculate the maximum throughput achievable in Live Mode (in terms of frame rates), depending on the exposure time. These rates are shown in Table 4-4:

Table 4-4. Maximum throughputs in Live Mode

Exposure time (ms)	Maximum throughput (fps)
0.001 ms	12.06
10 ms	10.8

It is important to note that these are the maximum achievable frame rates, but the throughput can be configured to lower values with the corresponding parameter.

4.3.3 Buffer Mode

In this new operation mode, instead of transmitting each image as soon as it has been received, it captures and saves a number of images in a row, transmitting them afterwards. For this purpose, images are saved in consecutive positions of the SRAM, saving all of them in a row without overwriting the previous ones. Once all the requested images are saved in the SRAM, they are transmitted in a row to the computer, at the frame rate specified in the FPS parameter. This is represented in the flow chart of the Buffer Mode, shown in Figure 4-11:

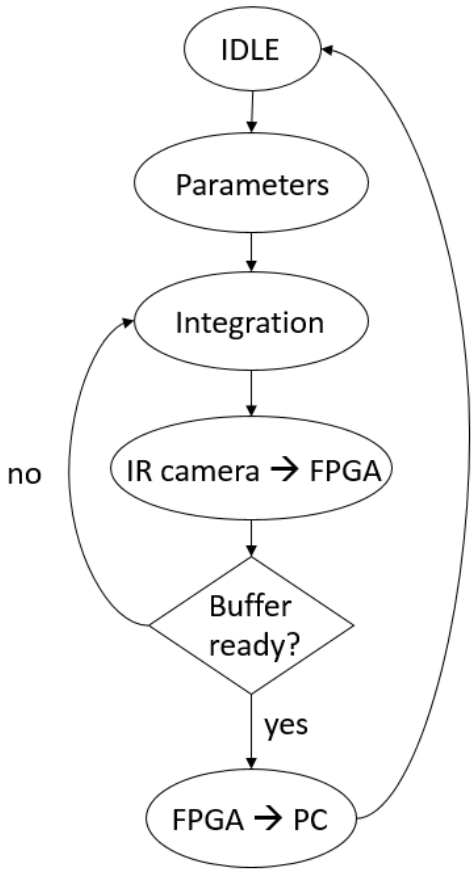


Figure 4-11. Flow diagram of the Buffer Mode

This operation is achieved by simply checking the number of images saved in the SRAM memory after saving each image: if there are as many frames saved as stated in the Buffer Size parameter, the buffer is considered to be full and the system jumps to the second stage of the overall operation, starting to transmit all the frames (saved in the memory one after another) in a row. If the number of frames saved is still below the Buffer Size stated in the parameters, the system sends again the integration signal in order to take another image from the camera. The way in which the images are saved in the SRAM memory is presented in Figure 4-12, with pixel values represented by random numbers, as it would happen in the system with the data coming from the camera instead of the Hamamatsu Debug Module:

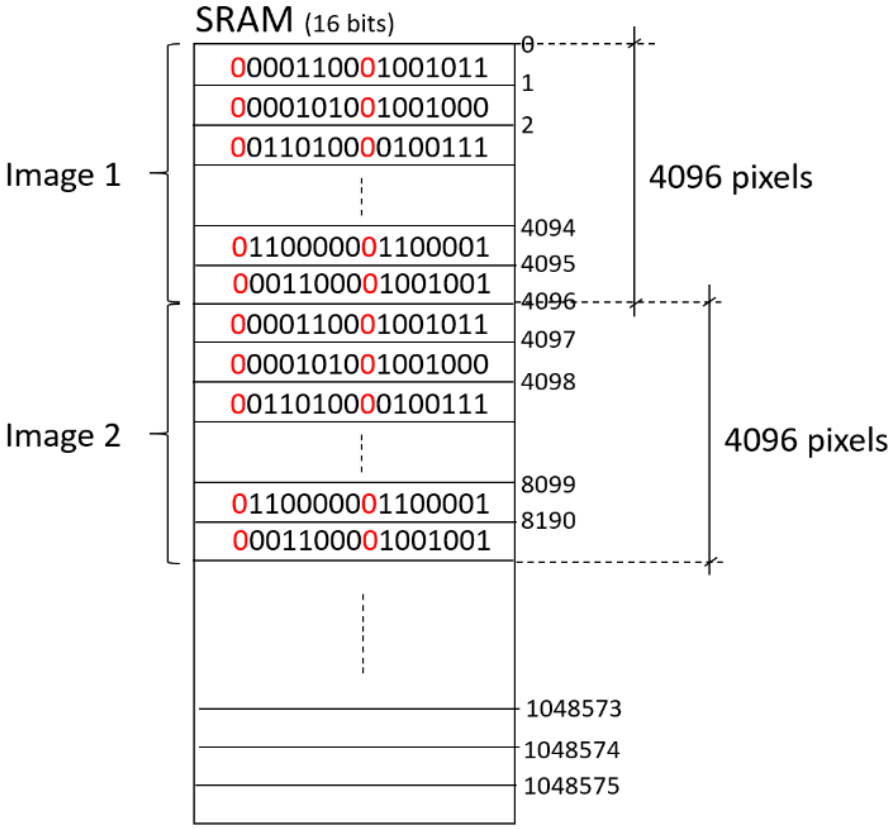


Figure 4-12. SRAM filled with images in Buffer Mode

Transferring this new flow chart to the finite-state machine of the system, the communication process between both endpoints of the imaging system (camera and computer) is separated in two parts: the image reception process in the frame grabber and image transmission process from the frame grabber to the

computer. Thanks to this, in the Buffer Mode the bottleneck of the transmission from the frame grabber to the computer is skipped, getting to a very high rate of images captured per second. Obviously, in exchange for this high speed capture, the frames are not transmitted in real time and the number of images captured is limited by the size of the SRAM memory.

Thus, the capture rate depends uniquely on the duration of the exposure time. Looking at the timings shown in Table 4-3, ignoring the transmission time from the frame grabber to the PC, the capture rate is calculated using the next equation:

$$Capture\ rate = \frac{1}{Exposure\ Time + 974.65\mu s} \quad \textbf{(4-2)}$$

Some reference values of this “sampling” speed in relation with the exposure time for the capture are presented in Table 4-5:

Table 4-5. Reference image capture rates in Buffer Mode

Exposure time (integration time)	Image capture rate (images/s)
0.001 ms	1024.95
0.01 ms	1015.58
0.1 ms	930.53
1 ms	506.41
10 ms	91.11

On the other hand, the second stage of the Buffer Mode operation, the transmission of data from the frame grabber to the computer, can be also slightly increased. However, the time saved from the reception thanks to this separation of stages is not significant compared to the transmission time, since

the reception of each is proved to be much faster. Looking at the timings shown in Table 4-3 and ignoring the exposure and reception times, the transmission time is taken (81.92 ms) and the maximum frames per second rate from the frame grabber to the computer is easily calculated: 12.21 fps.

On the other hand, this mode is not aimed for real time operation, and therefore the FPS rate is not as important as in the Live Mode.

4.4 Adapting the design to the real camera

When the Buffer Mode was almost finished but still being debugged, the IR camera finally arrived to the laboratory. After creating the adaptation board for the camera and packing it along with the camera in the protective case (see Section 3.2.4), in order to check the correct operation of the Hamamatsu imaging system it was necessary to analyse how the signals coming out from the camera get to the FPGA. These signals measured are shown in Figure 4-13, Figure 4-14 and Figure 4-15, with the system running in Live Mode:

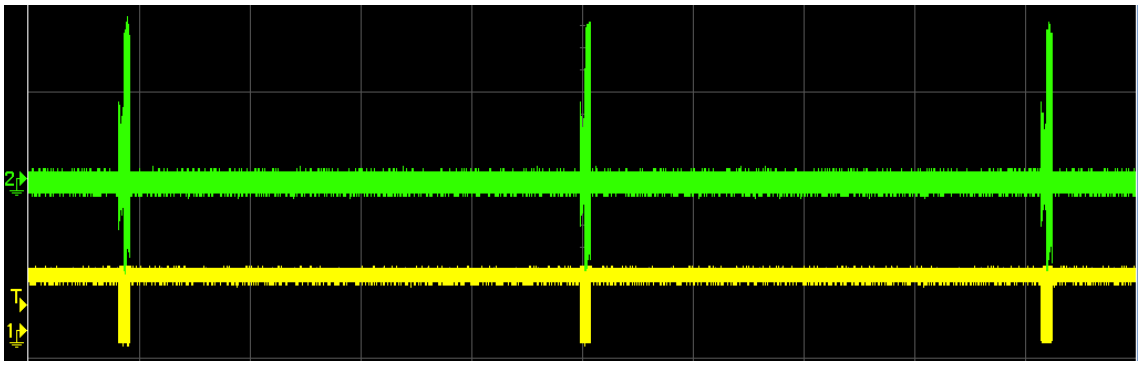


Figure 4-13 VIDEO (green) and AD_TRIG (yellow) signals with the frame grabber running in Live Mode (1)

In this oscilloscope screenshot (Figure 4-13) the periodical reception of images coming from the camera into the frame grabber can be checked. Each of this pulses is a whole image, with the data coming in the VIDEO signal (green signal) and with the AD_TRIG's falling edges represented in the yellow signal. Zooming in this graph, the structured transmission of each row can be checked:

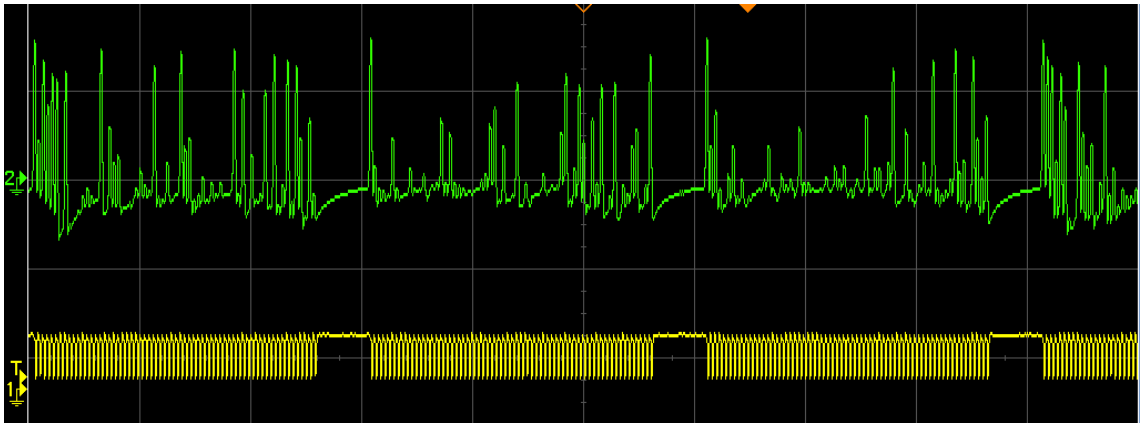


Figure 4-14. VIDEO (green) and AD_TRIG (yellow) signals with the frame grabber running in Live Mode (2)

In this zoomed graph (Figure 4-14), the transmission of rows with a blank space between each of them starts to be clearly plotted. Each of those groups of AD_TRIG's falling edges represent a row, with the 64 pixels' values of each row received by means of the analogue VIDEO signal.

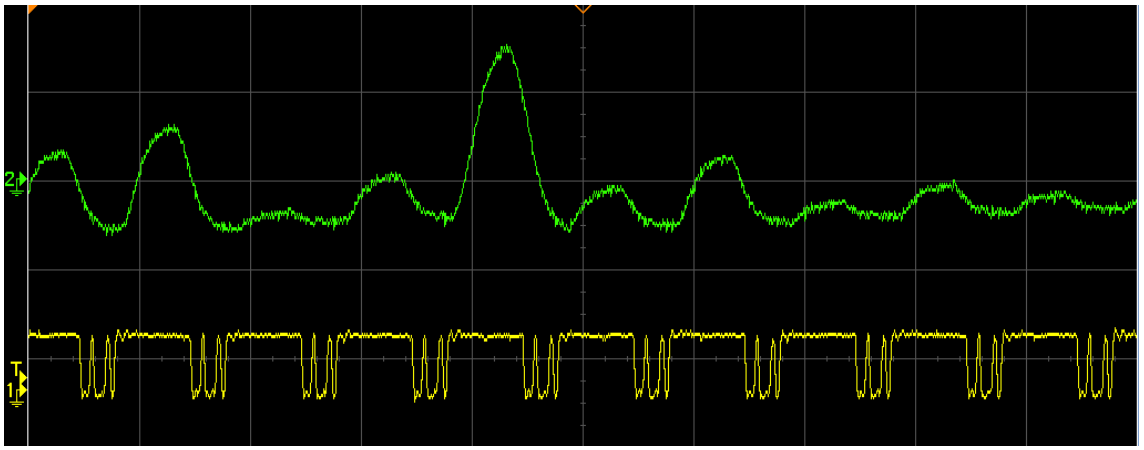


Figure 4-15. VIDEO (green) and AD_TRIG (yellow) signals with the frame grabber running in Live Mode (3)

Finally, when zooming in the graph to pixel level (Figure 4-15), the reception of each pixel along with its corresponding AD_TRIG falling edge can be checked.

It is important to note that the VIDEO signal is measured directly from the camera, whilst the AD_TRIG signal plotted above is the signal after getting into the FPGA, showed as an output of the FPGA for debugging purposes. In order to know how to signals managed by the FPGA are, the most valuable stage to plot in the life of the signal is when it is already in the FPGA (the case of AD_TRIG).

Regrettably, although the AD_TRIG signal is correct right when coming out from the camera (not shown in the graphs above), for some reason, in the FPGA it is received with a strong ringing. This means that, after the falling edge of the signal, the signal should be constant in a low level for a certain period of time and then come back to high level, but the plot clearly shows some parasite peaks within the low level pulses. This peaks are strong enough to be a potential problem for the system, which could read the peaks as rising and falling edges. Therefore, it is necessary to create a clean AD_TRIG signal to use it as a reference for the system to know when to take each pixel value.

Regarding the VIDEO signal, after digitalising and receiving it in the FPGA, the graphical representation of this digitalised signal by means of the oscilloscope would involve converting the signal back to analogue format. For this purpose, it would be needed to use the DAC of the Data Acquisition Card, what would have an extra effect in the VIDEO signal, making impossible to plot the signal in the oscilloscope just as it is within the FPGA. Therefore, the VIDEO signal managed by the FPGA cannot be plotted in the oscilloscope, being necessary to find a way of displaying the actual received video, relating it to the position of the clean AD_TRIG's falling edges. This is essential to know the delays between the received triggers and pixels values (the peaks in VIDEO signal) and, therefore, to know when to take each pixel value from the digitalised VIDEO signal.

Finally, there is no fixed ground value in the VIDEO signal. This can be easily understood looking at Figure 4-14, where the value of VIDEO signal during the blank space between rows (when it is not giving any pixel's value) is not constant. Besides, analysing the VIDEO signal in the oscilloscope zoomed to

pixel level (like in Figure 4-15) it can be checked that not only the peaks of the VIDEO signal, which are the values corresponding to each pixel, can increase or decrease, but also the lowest values between pixels can increase or decrease. Therefore, it would not be useful to save the value of each pixel (each peak in the VIDEO signal) directly in the SRAM: it is necessary to calculate the difference between the peak and the next lowest value of the VIDEO signal in order to get the real value of the pixel. For this purpose, it is paramount to get the view of the VIDEO signal inside the FPGA in order to relate its delay with the phase of the clean AD_TRIG signal.

Once stated the problems in the adaptation of the frame grabber using the Hamamatsu Debug Module to the final frame grabber using the real IR camera, the whole adaptation process can be separated in three main stages:

- Creating the clean AD_TRIG signal
- Creating an extra operation mode (Calibration Mode) to display the VIDEO signal as it is within the FPGA in relation to the clean AD_TRIG signal.
- Calculating the real value of the pixel (difference between VIDEO peaks and relative grounds).

4.4.1 Creating the clean AD_TRIG signal

For the creation of the clean AD_TRIG signal, which will be internal to the FPGA, a falling edge of the original AD_TRIG signal must be understood as valid only if it has been at high level for a minimum time right before the falling edge.

This method is implemented by means of a shift register of 4 bits. Using the 40 MHz clock, the FPGA takes a bit value from the AD_TRIG signal every 25 ns, putting it in the LSB of the shift register right after shifting the bits of the register one position to the left. Thus, when the first falling edge of the original AD_TRIG signal is received for every pixel, the FPGA will take a 0 value for the AD_TRIG, having in this precise moment the shift register full of bits with value 1. Then, the clean AD_TRIG signal will pass from high value to low value, holding the

low value for two clock cycles and then coming back to high value. Summarising, the clean AD_TRIG will have a falling edge when the next conditions are met: shift register = “1111” and original AD_TRIG = ‘0’. This process is graphically explained in Figure 4-16:

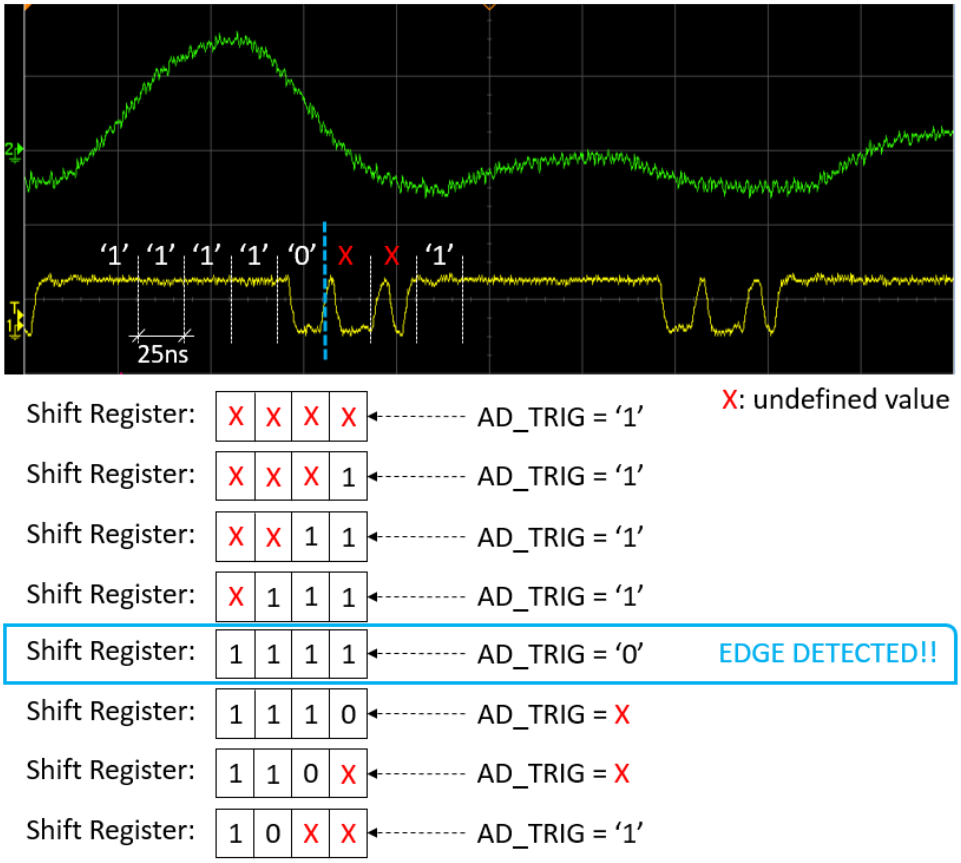


Figure 4-16. Process of detecting falling edges in the original AD_TRIG

In each clock period, the LSB of the shift register is the AD_TRIG value received in the previous clock period. With this method, the system will detect only a falling edge per pixel (in the first falling edge of the original AD_TRIG signal). The next edges within the same low stage of the pixel will be discarded, regardless of the values that the next two values may take. For instance, if the first undefined value is 1 and the second one is 0, when detecting the value 0 in the AD_TRIG the system will not create a falling edge in the clean AD_TRIG since the values in the shift register will be “1101”.

This method will create a clean AD_TRIG with a certain delay with respect to the actual falling edge. Nevertheless, this delay does not mean a new problem for the development, since the VIDEO and AD_TRIG signals are not synchronised anymore once inside the FPGA, and the delay between them is still unknown. This will be calibrated in the next stages of the process, when displaying the VIDEO signal managed inside the FPGA in relationship with the clean AD_TRIG signal. The created AD_TRIG signal can be checked in Figure 4-17, were the pink signal is the clean AD_TRIG coming from the FPGA and the red signal is also the clean AD_TRIG, but this time read as a digital signal in the oscilloscope (as it would be inside the FPGA):

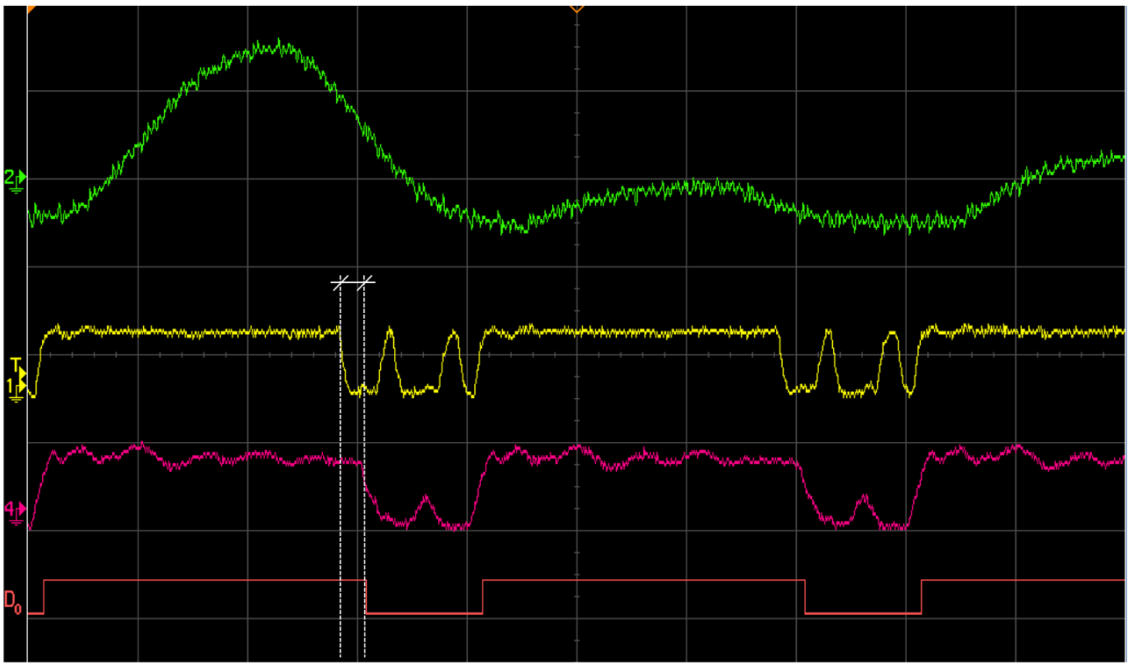


Figure 4-17. VIDEO (green), original AD_TRIG (yellow), and clean AD_TRIG read as both analogue (pink) and digital (red)

4.4.2 Creating the Calibration Mode

Firstly, although this operation mode is developed only for calibration purposes, it will be kept in the final version of the system. For using it, the only parameter affected with respect to the original parameters configuration is the Operation Mode parameter, shown in Table 4-6:

Table 4-6. Operation Mode parameter's addresses and allowed values

Parameter name	Address	Value
Operation mode	0x01	0x01: Live Mode
		0x02: Buffer Mode
		0x03: Calibration Mode

Once the clean AD_TRIG has been created, the next step is to calculate the delay between the clean AD_TRIG and the VIDEO signal. For this purpose, as the VIDEO signal cannot be plotted in the oscilloscope, it is necessary to take the digital values sampled around the clean AD_TRIG and output them to the computer. This is the function of the third operation mode: the Calibration Mode.

For this purpose, a shift register as wide as 25 pixel values (25 times 14 bits) is implemented, which will be running constantly, shifting the register to the left and inputting the last sampled value of the VIDEO signal in the 14 LSB. Thus, the system saves in every moment the last 25 values that the VIDEO signal has taken.

When the first falling edge of the clean AD_TRIG is received, the system shifts the register 12 more times to the left, in order to leave the value sampled when receiving the falling edge of the clean AD_TRIG in the middle of the buffer. The same process is followed shifting only once, to leave the value sampled with the AD_TRIG at the right, and 23 times, to leave it at the left. Thus, the VIDEO signal available in the FPGA can be displayed in the surroundings of the first falling edge of the clean AD_TRIG. This data is taken in the computer and represented with Matlab in three separated graphs, since the three 25 pixels data sets are captured for three different images and they cannot be merged.

The first two graphs are available in the Appendices (Calibration mode's images), whilst the third one is shown in Figure 4-18:

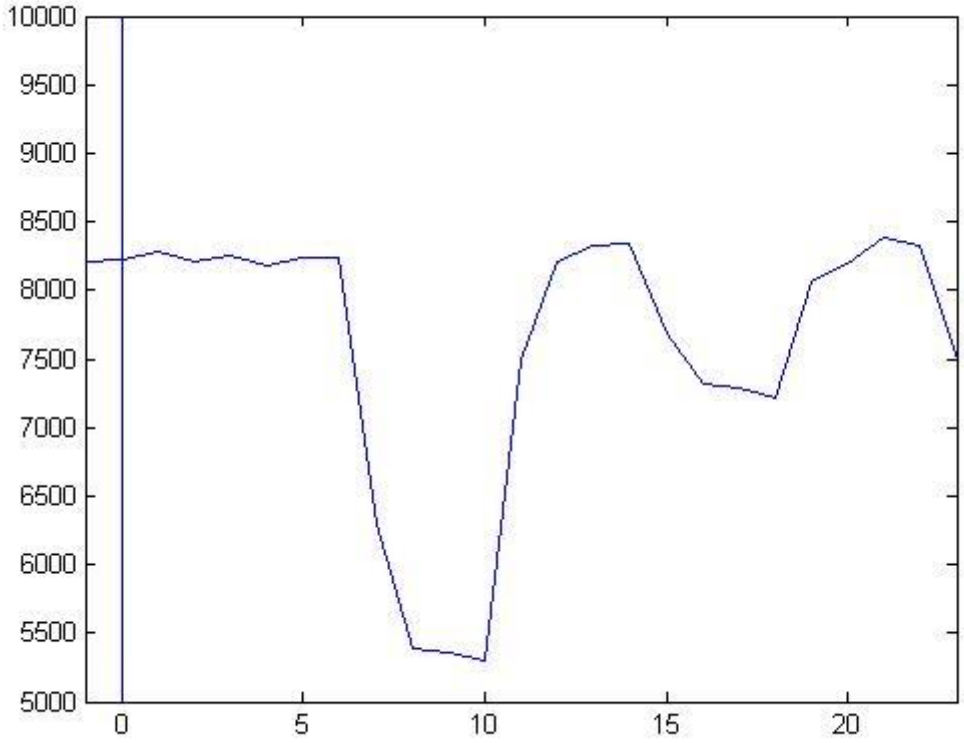


Figure 4-18. Digitalised values of the VIDEO signal around the first falling edge of the clean AD_TRIG signal

First off all, the form of the wave is inverted, most probably because of the polarity of the transformer involved in the digitalisation of the analogue signal in the Data Acquisition Card. However, this means no problem, since the target is not the value of the pixel itself, but the difference between maximums and minimums of the VIDEO signal.

In Figure 4-18 it is easily noticed how the VIDEO signal gives the value of a new pixel every 8 clock cycles (from sample 10 to sample 18). This is the expected behaviour, since the pixels transmission rate from the camera to the computer is supposed to be 8 times slower than the frequency of the clock that the camera receives as a reference (which is the same one used in the system: 40MHz).

Regarding the delay between the falling edge of the clean AD_TRIG and the correct value to be sampled in the VIDEO signal, it is clearly 10 clock cycles, since the peak value (minimum) of the plotted wave (the maximum peak in the real VIDEO signal) is sampled ten clocks cycles after the falling edge.

This can be double checked with the second pixel value, which is in the 18th position, just 8 samples after the first one, what makes sense considering the issue of the frequency of image transmission.

4.4.3 Calculating the real value of the pixel

Knowing from the previous section that the pixel is sampled 10 clock cycles after the falling edge of the trigger, it is only needed to find the position of the sample that should be considered as the relative ground for each pixel to calculate the difference between both values. Looking at the graph in Figure 4-18, the sixth sample value after the trigger is chosen for this purpose, since it is the last sample before the signal starts to change to the pixel value.

In order to perform the subtraction between the ground and the pixel values, a slight modification in the system is needed. In the versions developed in base to the Hamamatsu Debug Module, the pixel values were directly taken from the 14 bits bus coming from the Data Acquisition Card. However, for the final version, the values for calculating the difference between ground and pixel values are taken from the shift register used for buffering the last 25 VIDEO samples.

Besides, as pixels arrive to the frame grabber with a separation of only 8 clock cycles and the delay of the targeted samples is beyond this number (10 clock cycles for the peak), the values cannot be taken sequentially anymore. In case of doing it sequentially using an only counter to indicate the system when to take the samples from the shift register, this counter would count up to 10 clock cycles after the last falling edge of the trigger. However, the next falling edge of the trigger would appear in the 8th clock cycle (before the counting has been finished) and the counter would start again from zero. Thus, the frame grabber would never take a pixel and the whole system would crash.

For this reason, it is necessary to implement to parallel counters, initialising them alternately every time a new falling edge in the clean AD_TRIG is detected, and running both of them at the same time. Thanks to the nature of FPGA devices, this can done easily.

Regarding the subtraction operation, it is implemented using an IP core developed by Altera, which forces the FPGA to perform the operation with a delay of an only clock cycle.

Once the difference between ground and pixel value to be sent for each pixel is calculated, the frame grabber follows the same process as in the versions developed for the Hamamatsu Debug Module, getting to the final system developed for the real IR camera.

5 RESULTS

The final version of the frame grabber is a fully configurable system with two operating modes correctly working and an integrity check and synchronisation method implemented by means of the frame begin bit.

5.1 Parameters configuration protocol

Before analysing the received images, it is important to show the result of the parameters configuration stage, which is the first step in the operation of the frame grabber. This process is done by the final user and from the computer, using any serial protocols emulator. In Figure 5-1 the process of configuration (using the serial protocols emulator called Docklight) is presented:

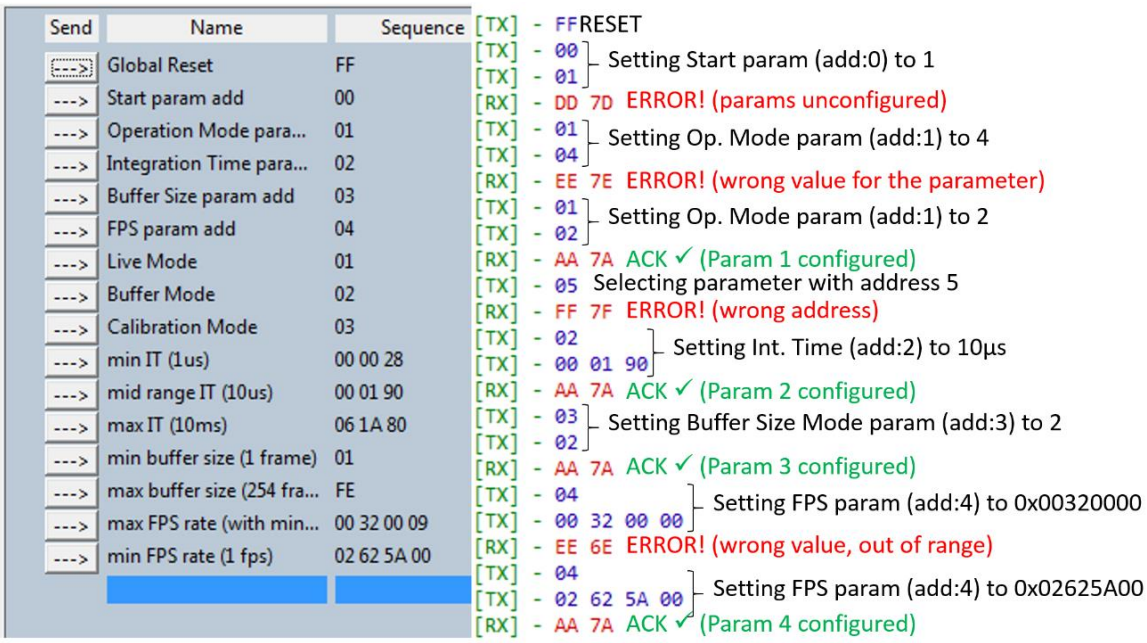


Figure 5-1. Example of configuration sequence

5.2 Received raw data

Once all the parameters are correctly configured, the start parameter can be set to 0x01 to start the capturing process. Figure 5-2 shows how images start to be received right after setting the start parameter. This communication is performed using the intermediate version of the system based in the

Hamamatsu Debug Module because, being an increasing sequence of numbers, it helps in the understanding of the process:

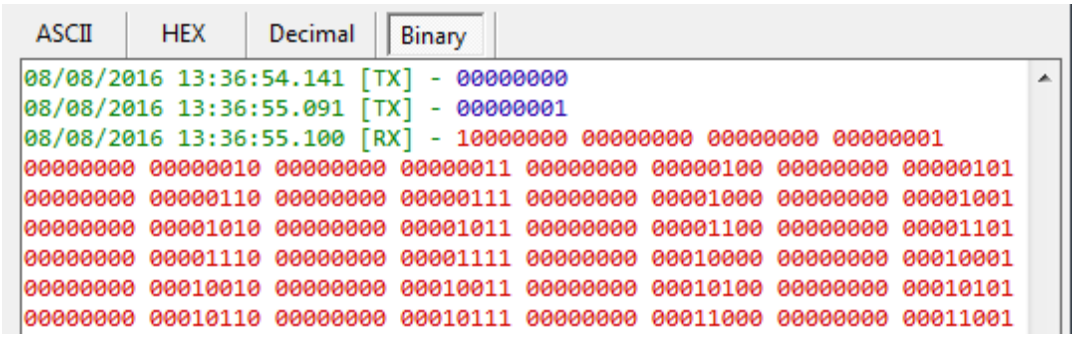


Figure 5-2. Example of image reception from the Hamamatsu Debug Module (first image of the stream)

This time the values are displayed in binary format to facilitate the identification of the Frame Begin bit. It can be seen how the values received for the pixels (2 bytes each) are a sequence of binary numbers increasing one by one. The integrity and synchronization method implemented by means of the Frame Begin bit can be also checked in this figure: in every byte the first bit is a 0, but in the first byte of the frame this bit is 1. This can be also checked when the image is not the first one received from the frame grabber, as shown in Figure 5-3:

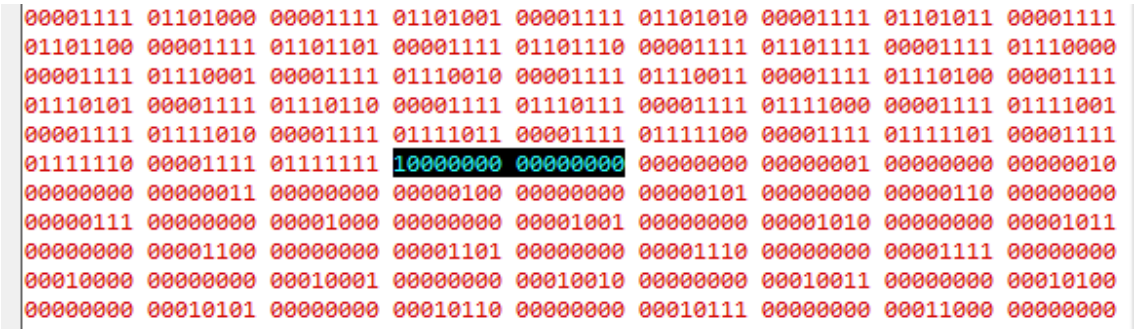


Figure 5-3. Example of image reception from the Hamamatsu Debug Module (any image in the stream after the first one)

The shaded two bytes correspond to the first pixel of the second frame, identified thanks to the first bit of the pixel, which is a 1. Once again, it can be checked that the first bit of every pixel is 0, except in the first pixel.

Regarding the values of the sequence, the last pixel received from the previous frame (the one before the shaded pixel) takes the binary value “0000111101111111”. The 0 in the MSB of the second byte of the pixel is there because of the Frame Begin bit, which is 0, since the UART TX module is designed this way and, being the mechanism of the increasing sequence just a debugging method, it was not worth to modify the UART TX module only for this issue. Therefore, the real value for the last pixel of the first frame is “0000111111111111”, which in decimal format can be read as 4095, the number of the last pixel of the sequence. On the other hand, after the first pixel the values start to represent the same sequence again, from 0 to 4095.

Obviously, although it is not as descriptive as an image which is just a sequence of numbers, it is mandatory to show the reception of images with the final version of the system, taking images generated by the real camera. As expected, the values received are random, as shown in Figure 5-4:

```
08/08/2016 13:24:33.137 [TX] - 00000000
08/08/2016 13:24:35.077 [TX] - 00000001
08/08/2016 13:24:35.086 [RX] - 10001000 00011110 00000111 00101011
00000111 00110001 00000110 00100001 00000110 01011111 00000101 01111101
00000101 01010001 00000101 00011011 00000101 00101101 00000100 01010110
00000100 01000011 00000100 01111101 00000100 01101111 00000100 00010101
00000100 00001110 00000011 01101010 00000011 00101101 00000011 00101100
00000011 01110111 00000011 01110100 00000011 01000011 00000011 00011111
```

Figure 5-4. Example of image reception from the real IR camera (first image of the stream)

The first bit of every byte is a 0, except in the first byte of the image, where the frame begin bit sets it to 1. This can be checked for any image received after the first one, as shown in Figure 5-5, right the same way as when receiving the mock images from the Hamamatsu Debug Module:

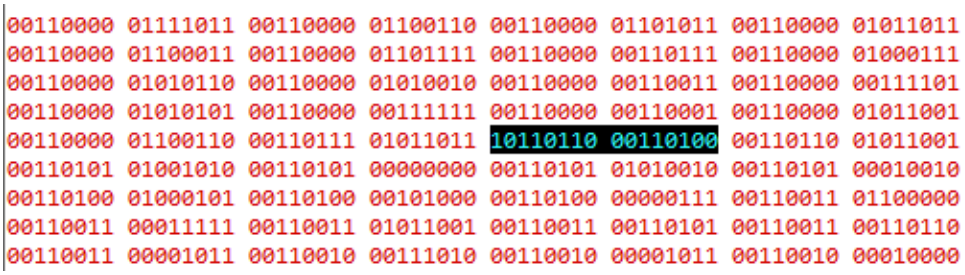


Figure 5-5. Example of image reception from the real camera (any image in the stream after the first one)

5.3 Received images (decoded raw data)

Once the images are received in the computer, they can be decoded by means of a simple script in Matlab. After normalising the values received for each pixel into the range from 0 to 255 (the full range for greyscale images), the resulting image signal can be plotted along with the final image. For an easy first testing, a coin is placed in the edge of the sensor, taking only the first frame of the stream and getting to the resulting signal plot and image shown in Figure 5-6:

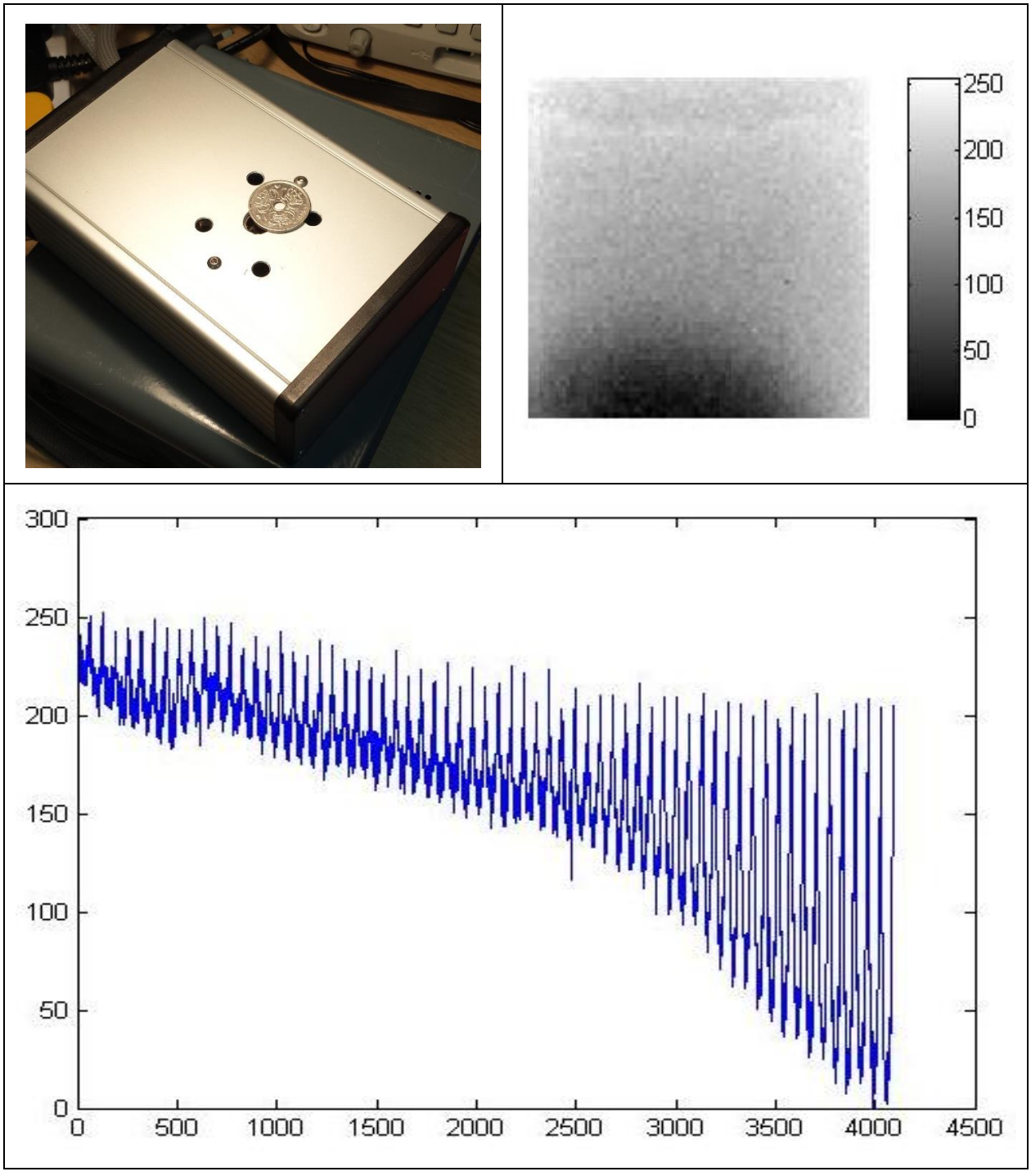


Figure 5-6. a) IR camera with the coin in front of the lens; b) resulting IR image; c) resulting signal plotted

Placing the coin in the centre of the lens (note than the coin has a hole in its centre), the buffer mode is tested. For this purpose, the frame grabber is

configured in buffer mode with a buffer size of 10 images, and with an integration time of 10µs, what gives a capturing rate of 1015 fps. The 10 buffered images are displayed one after another without separation, just as they are received in the computer. This is shown in Figure 5-7:

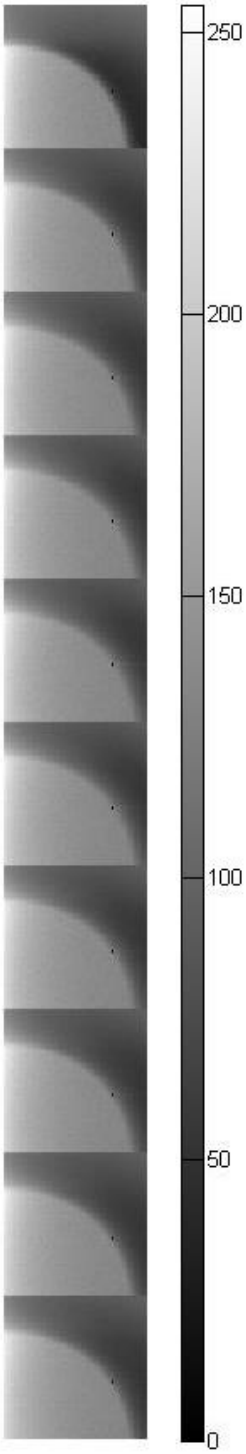


Figure 5-7. 10 buffered images received in a row

Regarding the Live Mode, it cannot be checked displaying the video stream in real time because the software needed for this purpose has not been developed yet (out of the scope of this thesis). Nevertheless, having checked this operation mode in the serial protocols emulator (checking the reception of raw video frames in real time) and taking into account that the Buffer Mode, which is based in the same principles, is working efficiently, it can be concluded that the Live Mode is also correctly working.

With the correct operation of the camera already proved, it is also interesting to check its accuracy. Using a complex system of lenses[3] attached to the protective case of the imaging system, the IR camera focuses now in a small bulb. The system of lenses is shown in Figure 5-8, and the resulting images of the bulb in Figure 5-9:

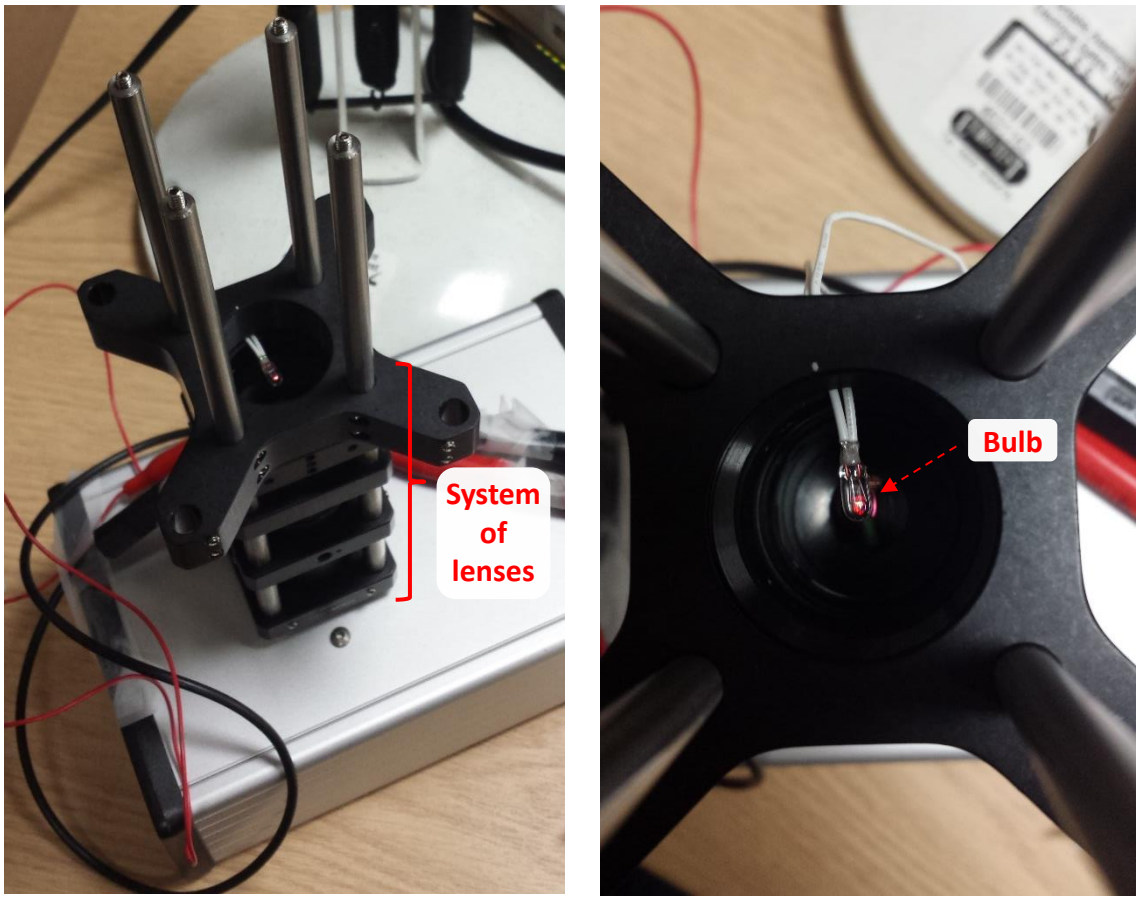


Figure 5-8. System of lenses for focusing the bulb

[3] Mounted by Dr. Tom Charret and Dr. Thomas Kissinger

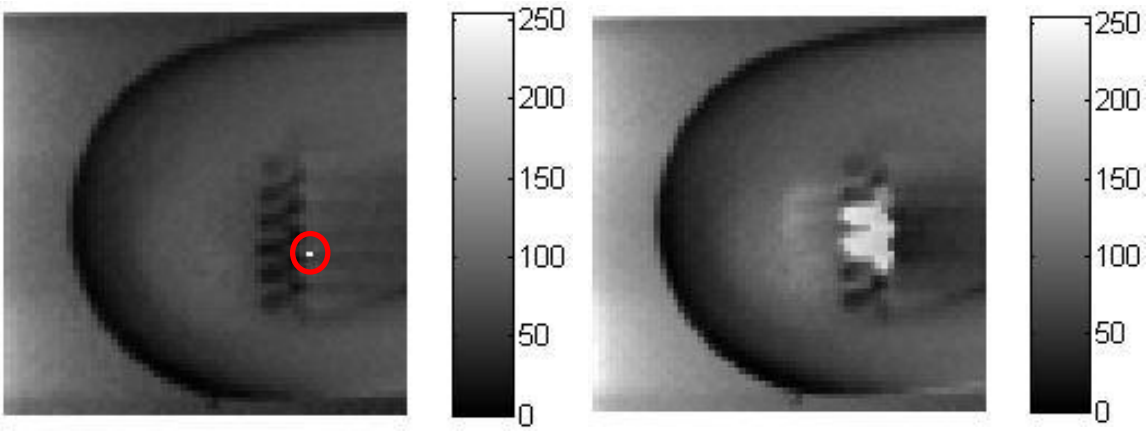


Figure 5-9. Images of the bulb off (left) and on (right)

As seen in Figure 5-9, the accuracy of the camera is very good, being able to catch the filament inside the bulb well defined. It is important to note that the white pixel in the red circle is a dead pixel of the image sensor, which means no problem since it does not affect the rest of the pixels.

Now, the bulb is switched on and off intermittently, and the camera is set once again in buffer mode. The aim of this test is to detect the variation in the bulb's illumination. With a buffer of 5 frames, the resulting transition is as shown in Figure 5-10.

With an integration time of 10µs, the camera is capturing images at a rate of 1015 frames per second. On the other hand, the bulb is not able to change its state (on or off) at such a high frequency. For this reason, the change in only 5 frames is not strong, as it only covers less than 5ms of the transition. Nevertheless, it can be noticed how the brightness of the images increases from the first frame to the last one.

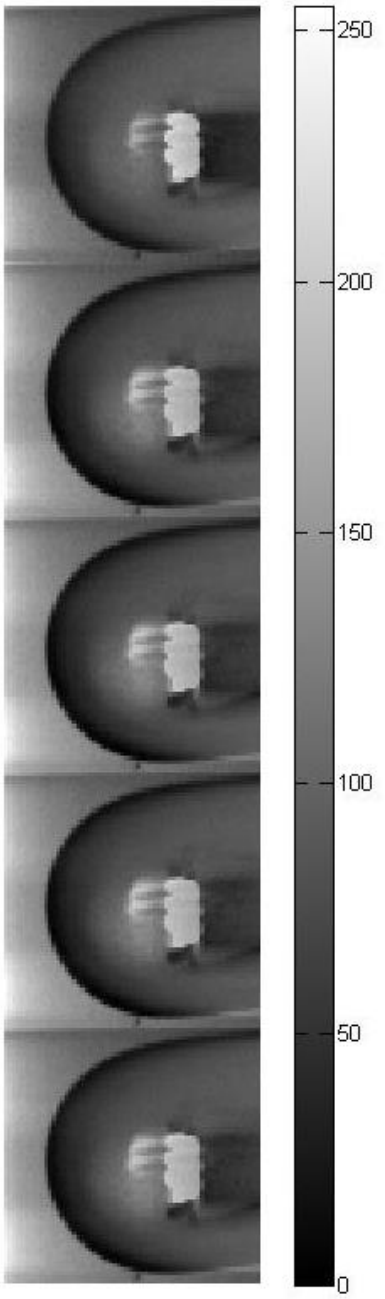


Figure 5-10. 5 frames transition of the bulb, captured in buffer mode at 1015 fps

This slight increase in the intensity of the image pixels can be also checked looking at the signal directly plotted without formatting it as an image, shown in Figure 5-11:

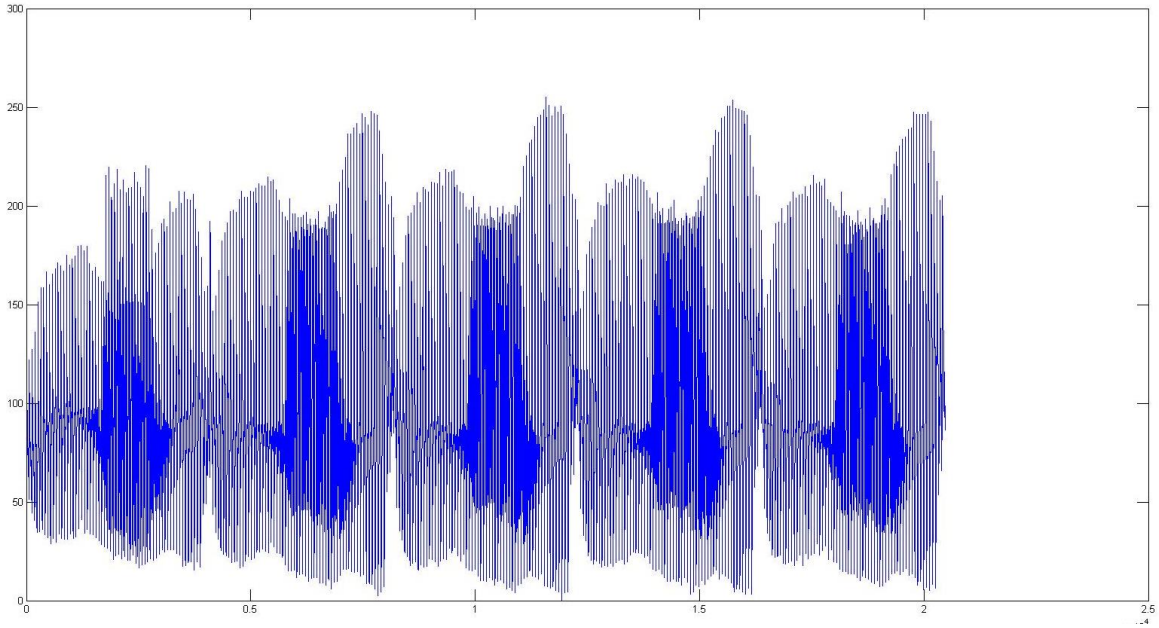
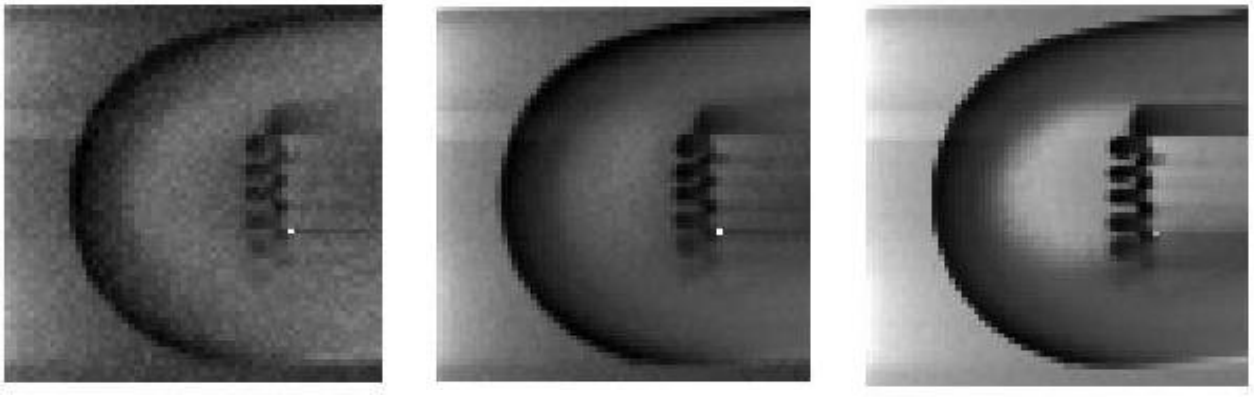


Figure 5-11. 5 frames transition of the bulb, captured in buffer mode at 1015 fps (signal plotted)

In this plot the separated frames can be easily distinguished as the wave form is repeated periodically up to 5 times. As expected, the amplitude of the signal is clearly greater in the last frame than in the first one, what means that the last image is brighter.

It is also interesting to test the correct operation of the parameter for setting the exposure time. For this purpose, the same picture is taken after setting the exposure time to different values: 2.925 µs, 6.4 µs and 10 µs. As expected, the image gets brighter as the exposure time for the capture is increased. These images are shown in Figure 5-12:



(a) (b) (c)

Figure 5-12. Images captured varying only the exposure time: a) 2.925µs, b) 6.4µs, c) 10µs.

5.4 Performance

Finally, it is interesting to give a view of the results in terms of performance of the system. First, the performance in terms of processing time is stated, as a summary to the values already presented in Sections 4.3.2 and 4.3.3.

In the Live Mode, the interesting feature is the throughput of the system. That means, the number of frames per second that the frame grabber is able to receive from the camera and send to the computer. This values must be configured in the FPS parameter within a range between 1 fps and 12.06 or 10.8, depending on the exposure time set for the image capture. In case of minimum expose time (0.001 ms), the throughput can get ot 12.06 fps; if the maximum exposure time is set, the throughput can get to a lower rate: 10.8 fps.

In the Buffer Mode, the most interesting feature is the capture rate. Once again, it depends of the exposure time set for the camera, ranging from 91.11 frames captured per second (with 10 ms of exposure time) to 1024.95 frames captured per second (with 0.001 ms of exposure time).

Regarding the FPGA usage, the percentage values are very low. These are shown in Table 5-1:

Table 5-1. FPGA usage percentage values

Device	Logic Elements (%)	Pins (%)	PLLs (%)
Altera Cyclone IV E (EP4CE115F29C7)	1,858/114,480 (2%)	149/529 (28%)	1/4 (25%)

6 CONCLUSIONS AND FUTURE WORK

First, after the analysis of the results, it can be concluded that the project has been successfully finished, delivering a fully configurable frame grabber, perfectly operating in not specialised hardware, getting to a system cheaper than the average commercial frame grabber.

The designed frame grabber offers two different operation modes for the final user, plus a third method for calibration purposes for the designer. Regarding the Live Mode, it provides a video stream in real time at a rate of up to 12 fps, which is enough but not very high to create a video that flows in a natural way for human eye.

This limitation in the throughput of the Live Mode is only set by the transmission standard used for sending the images from the frame grabber to the computer (RS-232), being the core part of the frame grabber (the process implemented in the FPGA) able to operate much faster. Fortunately, the transmission part was implemented in a separate state of the VHDL finite-states machine. That means, this part of the code is separated from the rest and it can be easily substituted with a more complex standard if faster operation is required.

Regarding the Buffer Mode, it is able to capture images at a rate of up to 1025 fps, being the throughput not relevant in this case. This is a very high capturing rate, but sometimes it is also interesting to decrease this rate. Regrettably, in the final system this cannot be configured by the user, depending only on the exposure time set in the corresponding parameter. For instance, in the case of Figure 5-10, a transition of a bulb from off to on states was aimed to be captured with a buffer of 5 images. Although a slight increase in the bright of the images can be seen, the capture rate is too high for the blinking frequency of the bulb.

In respect of the FPGA usage, most of the resources are not used (only 2% of the logic blocks are in use). Taking this into account, and considering the universal nature of the VHDL language, the system could be transferred to simpler (and cheaper) equipment using the source files developed in this project

and doing a bit of adaptation work. Therefore, a simple custom board could be created, including only the FPGA and the peripherals used in this system (also the ADC of the external Data Acquisition Card), and be easily configured with the files created in this MSc thesis. Thus, a very low cost frame grabber could be achieved for this specific IR camera.

As future work, the first point would be adding another parameter for configuring the capture rate. Using the already designed structure this could be done very easily and it would add a valuable feature to the system, solving the problem of the blinking bulb stated above.

Secondly, now that the system is proved to be working, the bottleneck created by the RS-232 standard should be eliminated. For this purpose, a faster and more complex communication standard should be implemented in the FPGA, making use of the multiple interfaces of the Evaluation Board to communicate with the computer using this new standard. Thus, throughput rates close to

Thirdly, it would be interesting to develop the Python software required for decoding the images in real time in the computer. Besides, it could be complemented with a GUI for the final user, making the control of the whole imaging system user-friendly.

Finally, there are some image processing techniques that are very relevant to gas sensing applications, which could be implemented in the frame grabber. There are several techniques that make use of the frames saved in a buffer, combining them to create an only output image. For instance, the average image of a number of frames stored in the buffer could be calculated by means of the combination of equivalent pixels (combining first pixel of each image and calculating the average of all of them, and iterating this process for every position in the pixels array). This image processing techniques could be implemented in a fourth operating mode.

REFERENCES

1. Athanas PM., Abbott AL. Real-time image processing on a custom computing platform. Computer. 1995; 28(2). Available at: DOI:10.1109/2.347995

2. Gatos L. United States Patent: INTERACTIVE CABLE TELEVISION SYSTEM WITH FRAME GRABBER. 2001. Available at: https://docs.google.com/viewer?url=patentimages.storage.googleapis.com/pdfs/US6253238.pdf

3. Hodgkinson J., Tatam RP. Optical gas sensing: a review. Measurement Science and Technology. 2013; 24(1): 012004. Available at: DOI:10.1088/0957-0233/24/1/012004

4. Laj P., Klausen J., Bilde M., Pla??-Duelmer C., Pappalardo G., Clerbaux C., et al. Measuring atmospheric composition change. Atmospheric Environment. 2009; 43(33): 5351–5414. Available at: DOI:10.1016/j.atmosenv.2009.08.020

5. Hodgkinson J., Pride RD. Gas Leak Imaging. 2005; (2): 47–50.

6. Padget M., Well B Van. Implementation of optical technologies for portable gas leak detection. Proc. Int. Gas Research Conf. 2004; : 1–20. Available at: papers2://publication/uuid/C1389F14-737A-4E2B-9F37-048862A70260

7. Gibson G., Well B Van., Hodgkinson J., Pride R., Strzoda R., Murray S., et al. New Journal of Physics. 2006; 2630(06): 1–8. Available at: DOI:10.1088/1367-2630/8/

8. ADLINK. Vision Guidance Robotics. Application: Pick and Place. 2016. Available at: http://www.adlinktech.com/industrial_automation/3D_Robot_Guidance.php?utm_source= (Accessed: 31 July 2016)

9. ADLINK. Full HD Medical Imaging. 2016. Available at:

http://www.adlinktech.com/medical/Full_HD_Medical_Imaging.php?utm_source (Accessed: 31 July 2016)

10. Fried D., Staninec M., Darling CL. Near-Infrared Imaging of Dental Decay at 1310 nm AND NEW OPTICAL DIAGNOSTIC. Journal of Laser Dentistry. 2010; 18(1): 8–16.

11. Gong J., Chen Z., Fan J., Yan L. Infrared medical image visualization and anomalies analysis method. 2015; 9814: 98140B. Available at: DOI:10.1117/12.2203648

12. Case JR., Young MA., Dréau D., Trammell SR. Non-invasive thermal IR detection of breast tumor development in vivo. 2015; 9412: 94124Q – 94124Q – 7. Available at: DOI:10.1117/12.2081398

13. Jin S., Cho J., Pham XD., Lee KM., Kim M., Jeon JW. FPGA Design and Implementation of a Real-Time Stereo Vision System. IEEE Transactions on Circuits and Systems for Video Technology. 2010; 20(1): 15–26. Available at: DOI:10.1109/TCSVT.2009.2026831

14. Kieu H., Pan T., Wang Z., Le M., Nguyen H., Vo M. Accurate 3D shape measurement of multiple separate objects with stereo vision. Measurement Science and Technology. 2014; 25(3): 035401. Available at: DOI:10.1088/0957-0233/25/3/035401

15. Instruments N. Introduction to FPGA Vision Using the NI Vision Development Module. 2016. Available at: http://www.ni.com/white-paper/11417/en/ (Accessed: 1 July 2016)

16. Xu D., Boussakta S., Bentley JP. An FPGA-based low-cost frame grabber for image processing applications. Proceedings of the IEEE International Conference on Electronics, Circuits, and Systems. 2000; 1: 333–336. Available at: DOI:10.1109/ICECS.2000.911549

17. Greisen P., Heinzle S., Gross M., Burg AP. An FPGA-based processing pipeline for high-definition stereo video. EURASIP Journal on Image and Video Processing. 2011; 2011(1): 18. Available at: DOI:10.1186/1687-

5281-2011-18

18. Georgoulas C., Andreadis I. A real-time fuzzy hardware structure for disparity map computation. Journal of Real-Time Image Processing. 2011; 6(4): 257–273. Available at: DOI:10.1007/s11554-010-0157-6

19. Michalak M., Sekalski P., Grabowski K., Izydorczyk S. Fast FPGA-based frame grabber for digital progressive scan image sensors. Proceedings of the 21st International Conference on Mixed Design of Integrated Circuits and Systems, MIXDES 2014. 2014; : 533–536. Available at: DOI:10.1109/MIXDES.2014.6872258

20. Wikipedia. Serial Digital Interface. Available at: https://en.wikipedia.org/wiki/Serial_digital_interface (Accessed: 13 August 2016)

21. Wikipedia. HDMI. Available at: https://en.wikipedia.org/wiki/HDMI

22. Technologies C. HDMI connectors. Available at: http://www.cnctec.net/images/hd2.jpg (Accessed: 17 August 2016)

23. Cult of Mac. The Future of Apple’s Dock Connection. 2012. Available at: http://cdn.cultofmac.com/wp-content/uploads/2012/07/1-1211027648Yk8B.jpeg (Accessed: 16 August 2016)

24. Wikipedia. Universal Serial Bus. Available at: https://en.wikipedia.org/wiki/USB (Accessed: 17 August 2016)

25. Frazier H., Johnson H. Gigabit Ethernet: From 100 to 1,000 Mbps. IEEE Internet Computing. 1999; 3(1): 24–31. Available at: DOI:10.1109/4236.747317

26. Kenable. RJ45 Plug to Socket. Available at: http://www.kenable.co.uk/images/RJEXT-ends.jpg (Accessed: 7 August 2016)

27. Instruments N. PCI Express – An Overview of the PCI Express Standard. 2014. Available at: http://www.ni.com/white-paper/3767/en/#toc5

(Accessed: 13 August 2016)

28. PCI Express. PCI Express Base Specification - Version 3.0. ReVision. 2010;

29. Hardware T. PCI Express 2.0 Graphics Cards Tested. 2008. Available at: http://www.tomshardware.com/reviews/pci-express-2.0,1915-2.html (Accessed: 6 August 2016)

30. Texas Instruments. Interface Circuits for TIA/EIA-232-F. Advanced Packaging. 2007; 77(August): 63–64. Available at: DOI:10.1016/0141-9331(80)90217-3

31. Sillicon Labs. Single-chip USB-to-UART Bridge. 2013. pp. 1–22. Available at: https://www.silabs.com/Support Documents/TechnicalDocs/cp2104.pdf

32. Amazon. RS-232 serial male to female cable. Available at: https://www.amazon.co.uk/StarTech-RS232-Serial-Modem-Female/dp/B00CEMGMMM (Accessed: 5 August 2016)

33. Soltero M., Zhang J., Cockril C., Industrial HP a., Zhang K., Kinnaird C., et al. RS-422 and RS-485 Standards Overview and System Configurations. Configurations. 2010; (May): 25. Available at: http://www.ti.com/general/docs/lit/getliterature.tsp?baseLiteratureNumber=slla070&track=no

34. Huang S., Member S., Huang T., Liu C. Ringing Noise Suppression for Differential Signaling in Unshielded Flexible Flat Cable. 2015; 5(8): 1152–1159.

35. Saha S., Rahman MA., Thakur A. Design and implementation of a BIST embedded high speed RS-422 utilized UART over FPGA. 2013 4th International Conference on Computing, Communications and Networking Technologies, ICCCNT 2013. 2013; Available at: DOI:10.1109/ICCCNT.2013.6726481

36. Hamamatsu. InGaAs area image sensor. 2016; : 1–8. Available at:

https://www.hamamatsu.com/resources/pdf/ssd/g11097-0606s_kmir1016e.pdf

37. Terasic. Altera DE2-115 Development and Education Board. Available at: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=502 (Accessed: 17 August 2016)

38. Terasic. AD/DA Data Conversion Card. Available at: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=China&No=360 (Accessed: 17 August 2016)

APPENDICES

Appendix A Flow Diagrams

A.1 Basic version of the system

In the full flow diagram of the basic system, it can be checked how the operation of the whole system is based in the timings set by the Hamamatsu camera. All this times are set in terms of clock cycles, and the system waits by means of counters that it compares to the expected value and increases in every clock cycle. This operation can be checked in Figure A.1:

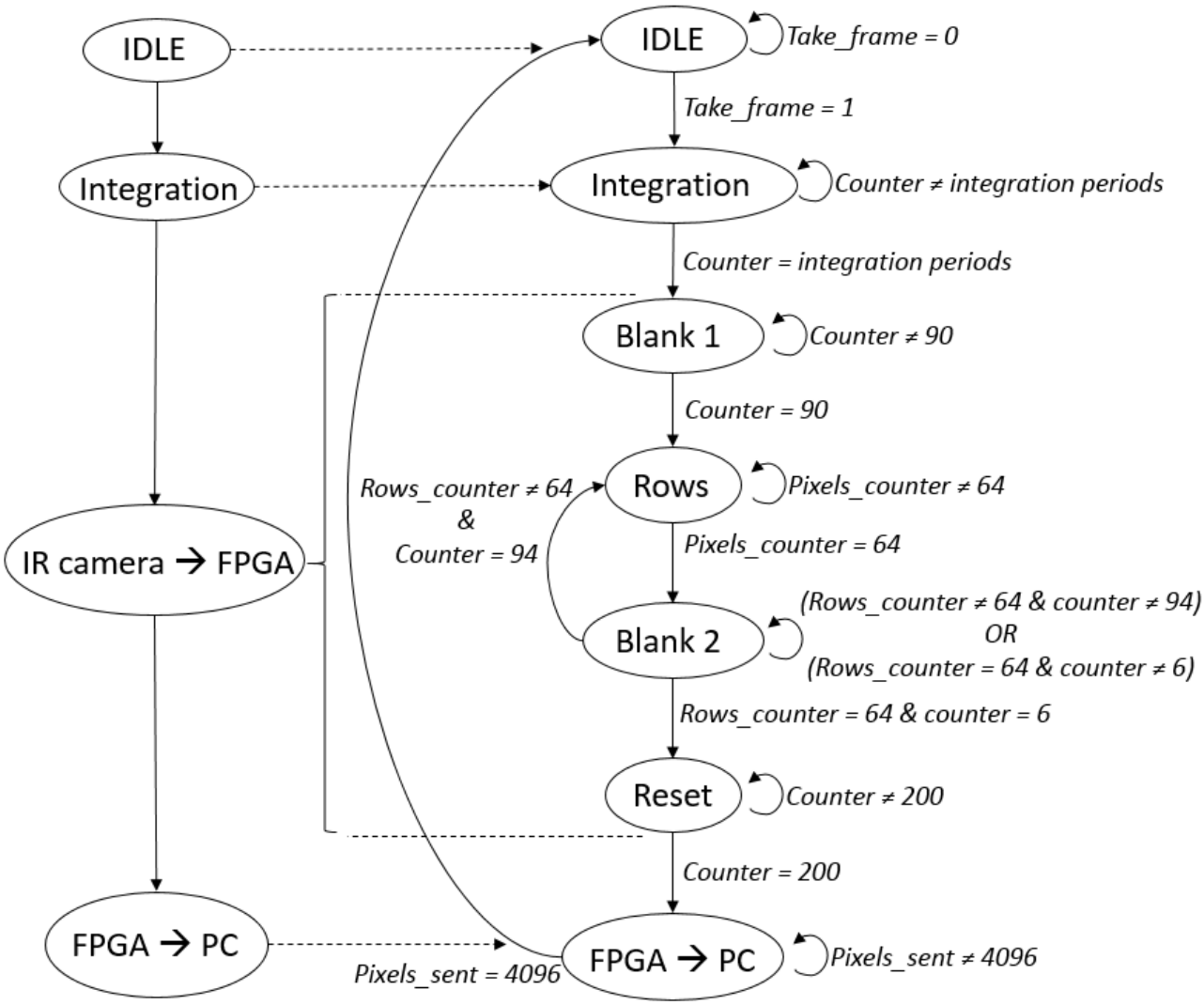


Figure A-1. Simplified (left) and complete (right) flow diagrams of the basic version

Appendix B Calibration mode's images

In this appendix the VIDEO waves plotted with the data obtained in the Calibration Mode are shown. First, the plot with the values of the VIDEO signal taken before the trigger and next with the trigger in the middle of the window. These are shown in Figure B-1:

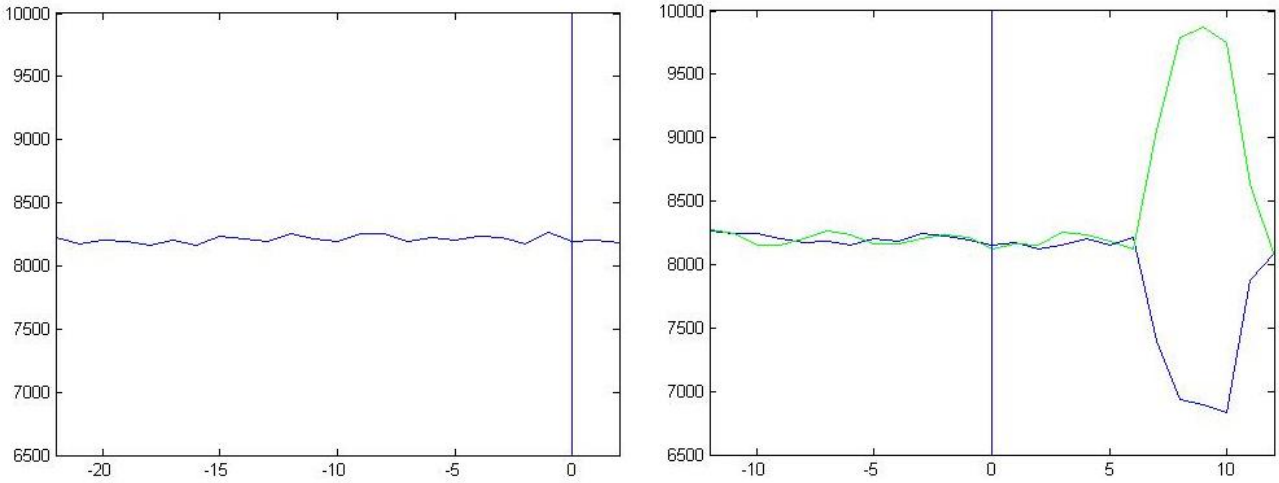


Figure B-1. VIDEO signal plotted with data of Calibration Mode, with the window shifted to the left of the trigger (left) and to the right (right)

As expected, before the first falling edge of the AD_TRIG no pixels are received in the VIDEO signal (flat wave). On the other hand, right after the falling edge of the AD_TRIG, the same waveform presented in Figure 4-18 can be seen again. Thus, the delay of 10 clock periods between the falling edge of the AD_TRIG signal and the VIDEO pixel value (negative peak in the wave) can be double checked. The green wave is the corresponding to an image with the light over the camera off.

The value is high because VIDEO signal is digitalised in two's complement format, where the first bit of the bus is used for stating the negative sign of the number. Therefore, without the proper decoding (removing the sign bit from the useful bits) negative values appear to be higher values.