School of Engineering and the Built Environment

Hardware Architectures for Infrared Pedestrian Detection Systems

by

Robert Walczyk

A thesis submitted in partial fulfilment
of the requirements of Edinburgh Napier University,
for the award of Doctor of Philosophy

May 2013

Contact: r.walczyk@napier.ac.uk

This document was set with the help of KOMA-Script and LaTeX

Abstract

Infrared pedestrian detection systems struggle with real-time processing performance. Known solutions are limited to either low resolution systems with basic functionality running at full frame rate, or software based techniques featuring higher detection rates with full set of features, however running only in off-line mode for statistical analysis. Here, a comprehensive solution for real-time pedestrian detection is described.

This research project includes investigation of possible solutions, design, development and implementation of a pedestrian detection system, processing data from infrared video source in real-time. Design requirements include processing at full frame rate as well as low memory and system resource consumption. The memory utilization is one of the major concerns since high demand for memory resources is a critical aspect in most image processing applications.

For the purpose of this task, a number of general purpose image processing techniques were revised, taking into consideration the suitability for infrared pedestrian detection. These tasks include background separation, acquisition noise removal and object detection through connected component labelling. They are discussed and addressed in individual chapters.

Various techniques for background segmentation are discussed. A chronological review of popular techniques is provided. The proposed architecture for background subtraction is based on selective running average for adaptive background model, supported by adaptive thresholding based on histogram calculation. In order to remove acquisition noise, a dual decomposed architecture was introduced, based on mathematical morphology and basic set theory definitions. It includes both erosion and dilation performed in a pipeline. For the purpose of object detection and feature extraction, a connected component labelling technique was employed, based on a single pass approach to fulfil real-time processing requirement.

The system was implemented, verified and tested on XUP FPGA Development Board with Virtex-II Pro XC2VP30 chip from Xilinx. Details and limitation of the specific implementation are discussed. An overview of experimental pedestrian detection results is provided. The thesis concludes with system analysis and suggestions for future work.

Acknowledgements

I would like to express my gratitude to all the people who have contributed towards this research or completion of this thesis. In particular, I would like to give thanks to the following:

To my university supervisors Dr T. David Binnie and Dr Alistair Armitage. For your advice and continued encouragement throughout this research. I appreciate your feedback and value the time you have spent on reading and correcting this thesis.

To my line manager from Cambridge Silicon Radio, Dr Tim Clapp, for generous study-leave allowance and flexible working hours during last months of my study.

To Stefan Maagh for valuable advice and effort when creating this thesis template.

To Piotr Wojtczuk for your contribution to this project in various aspects. Also, for our long night FPGA-whisky sessions, which I consider to be one of best experiences during my academic years.

To Alexander Balazs, for your contribution while working on the implementation of tracking algorithms on embedded processor core. It was a great experience for me to act as a teacher and supervisor. I am glad this cooperation was concluded with joint research paper presented at the 2011 IET Irish Signals and Systems Conference (ISSC 2011) in Dublin.

To my family, including my parents and sister in law - Krystyna, Janusz and Karolina Kołodziejczak, for your continuous support.

To my mother and father Lucyna and Waldemar, for your effort into my life-long education. I hope, after all these years, you can be proud of me.

Special thanks to my wife Ania and our little baby girl Zosia. This research project has affected your life as much as mine. I believe only you know how much effort I have made towards the completion of this thesis. There is no words to express my gratitude for your support, understanding and patience.

Robert Walczyk

I acknowledge that some of the work presented in this thesis has been presented or published at or in the following places:

[1] R. Walczyk, A. Balazs, A. Armitage, T.D. Binnie, ”System architectures for infrared pedestrian tracking,” in: *IET Irish Signals and Systems Conference (ISSC 2011)*, June 2011, Dublin, Ireland

[2] R. Walczyk, A. Armitage, T.D. Binnie, ”Comparative Study on Connected Component Labeling Algorithms for Embedded Video Processing Systems,” in: *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV'10)*, July 2010, Las Vegas, NV, USA

[3] R. Walczyk, A. Armitage, T.D. Binnie, ”FPGA Implementation of Hot Spot Detection in Infrared Video,” in: *IET Irish Signals and Systems Conference (ISSC 2010)*, June 2010, Cork, Ireland

[4] R. Walczyk, A. Armitage, T.D. Binnie, ”FPGA Implementation of Pedestrian Detection System using Infrared Image Sequences,” in: *Faculty of Engineering & Computing Postgraduate Research Conference (FECCI 2010)*, May 2010, Edinburgh, UK

[5] R. Walczyk, A. Armitage, T.D. Binnie, ”An Embedded Real-Time Pedestrian Detection System Using an Infrared Camera,” in: *IET Irish Signals and Systems Conference (ISSC 2009)*, June 2009, Dublin, Ireland

Contents

List of Figures

List of Tables

List of Abbreviations

ADC	Analogue Digital Conversion
ASIC	Application-Specific Integrated Circuit
BRAM	Block RAM
CCD	Charge-Coupled Device
CCL	Connected Component Labelling
CMOS	Complementary Metal-Oxide Semiconductor
CPU	Central Processing Unit
CoG	Centre of Gravity
DCM	Digital Clock Manager
DSP	Digital Signal Processing
EDK	Embedded Development Kit
FIFO	First-In First-Out
FIR	Finite Impulse Response
FOV	Field of View
FPGA	Field-Programmable Gate Array
FPS	Frames per Second
FSM	Finite State Machine
FWFT	First-Word Fall-Through
GCLK	Global Clock
GPP	General Purpose Preprocessor
GPU	Graphics Processing Unit
HREF	Horizontal Reference
HW	Hardware
IC	Integrated Circuit
IOB	Input/Output Block
IP	Intellectual Property
IPIC	IP Interconnect
IR	Infrared
ISE	Integrated Software Environment

LIFO	Last-In First-Out
LUT	Look-Up Table
OCM	On-Chip Memory
OPB	On-chip Peripheral Bus
PAR	Place and Route
PCLK	Pixel Clock
PLB	Processor Local Bus
RAM	Random Access Memory
ROI	Region of Interest
SE	Structuring Element
SW	Software
TRS	Timing Reference Signal
UART	Universal Asynchronous Receiver/Transmitter
UML	Unified Modelling Language
VGA	Video Graphics Array
VHDL	Very-High-Speed Integrated Circuits Hardware Description Language
VSYNC	Vertical Synchronization

CHAPTER 1

Introduction

Pedestrian detection is an active research area. There is a number of publications related to algorithmic development of computer vision application. They report increasing segmentation rates as well as improvements in classification and tracking. However, since they are mostly oriented on algorithmic development, they are targeted for implementation in software or simulation environments running on General Purpose Processing (GPP) platforms. This approach reduces the development time as well as provides extensive simulation capabilities, however places heavy demand on computer power and memory resources. Although the amount of computing power provided by GPP allows for basic real-time video processing (early days of 2008), due to high bandwidth imposed by video source and limited interfacing capabilities of ADC boards, video analysis is limited to post-processing in off-line mode. This is caused by the lack of designated video bus between ADC and PC.

Due to prohibitive costs, the use of infrared (IR) cameras, featuring wavelengths: 8-14μm, was limited mainly to military purposes. Recently, they have become viable for commercial use allowing the detection and tracking of pedestrians to offer an alternative technical approach to the one based on traditional CCD or CMOS sensors. Thanks to the thermal detection capability, the video acquisition is based on emitted thermal radiation rather than reflected as in visible images. Since the amount of emitted radiation is highly related to the temperature, detection of pedestrians, often featuring higher temperature than the surrounding background, will benefit from this technology. Moreover, although the thermal background is not constant, it varies gradually compared to sudden changes caused by ambient lighting in visible detection. On the other hand, due to different clothing styles, detected objects tend to form various shapes, they also may have separated body parts caused by occlusions. Moreover, thermal camera provides limited information about the spatial content of the scene, hence reduced number of image processing techniques can be applied.

1.1 Motivation

Pedestrian detection has a number of applications in security, safety as well as in retail and transportation sectors. Applications vary from post processing analysis of pedestrian movement to accurately studying shop footfall, to real-time surveillance applications, aimed at determination of suspicious behaviour or accidents. When considering retail applications, the high performance is not as relevant. They are mostly used for studying customers' moving patterns to optimize the flow in order to optimize the layout of shop isles or to appropriately position products with highest profit margin. However, real-time performance and good accuracy are critical for security systems to guarantee prompt reaction to accidents or crimes. Such systems operate on-line to detect the presence, unexpected behaviour or measure activity of pedestrians. In critical cases such applications can safeguard human beings. This is a rapidly growing market with a high potential for improvements.

With increasing popularity of Closed Circuit Television (CCTV) cameras, there is a strong demand for supervised detection or, ideally, fully automated systems. With current technology, a single human operator of CCTV control panel is capable of monitoring multiple cameras. However, an increasing number of cameras makes the task more difficult and prone to detection errors, hence the need for process automation. The automation of pedestrian detection is relevant to several areas in security or safety. It would allow reduce the cost of running system by removing the need for human operator, however, the main benefit is elimination of the human-factor causing detection errors due to long shifts or late working hours.

Since pedestrian detection systems become more important in critical applications such as surveillance or people counters employed for evacuation purposes in emergency situations, there is a need for high accuracy of the detection. Vision systems based on traditional cameras employ higher resolution devices. This involves problems related to high bandwidth imposed by video source. For the purpose of pedestrian detection and tracking, a different technical approach such as infrared cameras with wavelengths 8-14μm can be considered.

Segmentation and classification techniques employed within the area of pedestrian detection feature good detection rates, however, due to high complexity they put strong demand on computing power and memory resources. In order to speed up development process, tests and verification, they are often implemented in software, therefore real-time processing from camera streaming live is difficult to achieve on commercially available GPP platforms. Hence, an implementation on application specific device such as Field-Programmable Gate Array (FPGA) shall be considered. This requires optimization of known techniques for pedestrian detection in order to allow implementation in hardware. Moreover, due to omnipresent miniaturization requirement, such implementation shall use limited amount of system resources and memory to keep the package in small size, hence further customization is required.

1.2 Research Objective

The objective of the research presented in this thesis is to develop a self-contained system capable of real-time ($\geq 25fps$) pedestrian detection from infrared video stream. This involves an investigation into segmentation and classification techniques suitable for such implementation. Due to limitations imposed by GPP platforms, the system shall be implemented in hardware using a FPGA device. The implementation shall be small enough to fit a single device with limited memory resources. The architecture developed for this task shall be flexible and prone to further expansion. A modular design approach shall be considered to allow individual processing modules to be used as hardware accelerator within other processing systems.

In particular, this research explores the following areas:

- review of popular algorithms for pedestrian detection,
- Infrared camera utilization to gain detection rates and support light independent environmental scenarios,
- investigation into techniques suitable for hardware implementation,
- development of custom solutions to support real-time detection rates,
- development, test and verification of the digital design architecture for the purpose of self-contained IR pedestrian detection system for real-time video processing.

1.3 Contribution to Knowledge

The main contribution is the digital architecture designed, developed and implemented for IR pedestrian detection system, capable of processing video streams in real-time. As well as the architecture, there is the analysis and evaluation of the algorithms and design decisions that support and justify the digital design. Although the system can be used for surveillance applications as an off-the-shelf solution, an emphasis was placed on further expansion, for instance to support tracking applications as a hardware accelerator. The system was designed in a modular approach where individual modules act as hardware accelerators for particular processing tasks. They were developed based on available solutions reported in the literature. For implementation purposes, they are enhanced by custom solutions allowing the real-time performance of the entire system.

1.4 Outline of the Thesis

In addition to this introduction, the remainder of this thesis is split amongst six chapters, organized as follows:

Chapter 2: This chapter gives an introduction into the field of **Infrared Radiation and Image Processing** techniques used for the purpose of this research. An overview of technology employed for the task of system implementation is also provided.

Such introduction will be given in the following order:

1. Infrared Detectors
2. Digital Image Processing
3. Hardware Development and Implementation
4. Pedestrian Recognition

The theoretical background into the area of IR radiation is given including the definition of IR radiation followed by brief discussion on thermal sensors. An overview of thermal cameras is provided together with popular applications for such devices.

The second section gives an introduction into common digital image processing concepts, further referenced in this thesis.

This is followed by an overview of technology employed for the purpose of the final implementation. A description of the design flow is also included.

The last section of this chapter gives a general overview of the pedestrian detection processing flow. A typical data flow is included to introduce the reader to terminology and processing steps, further described in individual chapters.

This chapter does not provide a coherent literature-review for the entire research project. Instead, the literature-review was split and included within individual chapters.

Chapter 3: This chapter provides an introduction and detailed analysis of popular **Background Segmentation** techniques. It is supported by an extensive literature-review into the area, followed by a summary and discussion on techniques employed for this project. The second part of the chapter gives an overview of the custom algorithm. This is followed by a description of the architecture developed for the purpose of background segmentation.

Chapter 4: An introduction into **Morphological Noise Removal** is provided in this chapter. The literature-review includes set theory concepts and mathematical morphology definitions, further used in the chapter when formulating the algorithm. The implementation details are provided following an overview of popular techniques within this area. The chapter is concluded with a discussion on execution time and memory requirement imposed by the architecture.

Chapter 5: This chapter gives an extensive analysis on **Connected Component Labelling** algorithms, commonly used in modern vision systems. A detailed discussion on three chosen algorithms is provided to investigate the one suitable for this particular application. The second part of the chapter gives details on the architecture developed for this task together with discussion on resource utilization and the overall performance.

Chapter 6: In this chapter the **System Integration** is discussed. The top-level architecture for the entire system is provided. An overview of system components used for tests and verification is given. A brief description of the video acquisition module implementation is also provided together with data handling tailored for the purpose of this project. The data flow is described in detail pointing modules on the critical path. An introduction into tracking as a natural extension to this project is also provided. The chapter concludes with a summary on synthesis and resource utilization. Experimental detection results are also discussed. This is followed by performance comparison and suggestions for future work.

Chapter 7: This chapter provides a critical **Review of the Research**. It provides final conclusions and suggestions for future work.

CHAPTER 2

Image Processing for Infrared Pedestrian Detection

Infrared pedestrian detection is an active research area with a long research history at Edinburgh Napier University. Researchers from both Department of Engineering and the School of Computing were investigating efficient techniques for pedestrian detection and tracking using IR sensors. Due to a number of limitations imposed by available computing power, such processing was mostly limited to basic pyroelectric detectors or low resolution IR cameras. This thesis focuses on pedestrian detection using higher resolution cameras.

In this chapter an introduction into the field of infrared radiation and image processing techniques as well as technology used for this task will be provided. Such introduction will be given in the following order:

1. Infrared Detectors
2. Digital Image Processing
3. Hardware Development and Implementation
4. Pedestrian Recognition

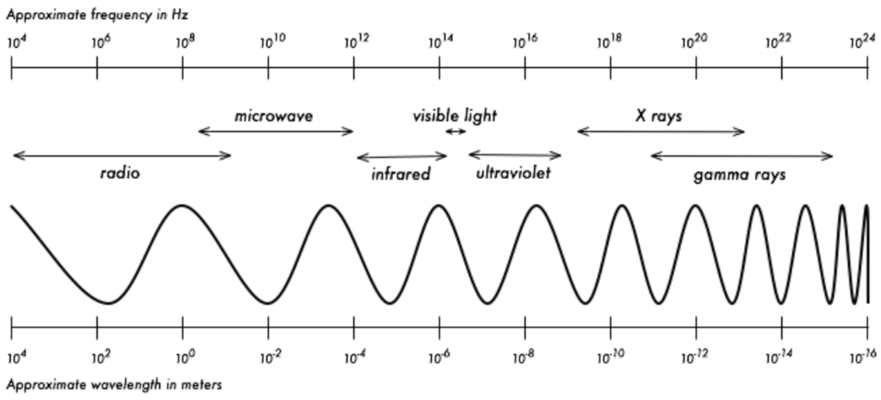


Figure 2.1: The electromagnetic spectrum characterised by wavelength λ and frequency ν [3].

2.1 Infrared Detectors

In this subsection a brief overview of infrared radiation and its applications will be provided. The main purpose is to introduce the reader to IR radiation and its capabilities for thermal detection.

2.1.1 Infrared Radiation

> ”... There are rays coming from the sun... invested with a high power of heating bodies, but with none of illuminating objects... . The maximum of the heating power is vested among the invisible rays... . It may be pardonable if I digress for a moment and remark that the foregoing researches ought to lead us on to others...” [1].
>
> *Sir William Herschel.*

These are the words of Sir William Herschel, Royal Astronomer to King George II of England, who was the first to reveal the existence of invisible partial radiation which nowadays is referred to as infrared portion of the spectrum. He was the first researcher who in 1800 found that there is a part of radiation which brings heat but is invisible. In 1801 he referred to it in two papers but did not work on his discovery ever again. Nowadays, all the characteristics of IR are broadly known and will be presented below.

IR radiation is an electromagnetic radiation with wavelength λ longer than visible light ($\lambda = 0.75\mu m$), but shorter than microwaves (1000μm). The visible range of spectrum is relatively short - in between 0.4μm and 0.75μm [2]. The electromagnetic spectrum can be seen in Figure 2.1.

The source of electromagnetic radiation can be any object at a temperature above absolute zero (-273^0C). The amount of emitted radiation and the wavelength distribution depend on the temperature and emissivity of the body [4]. Each object even at a low temperature of a few Kelvins emits electromagnetic radiation in the range of the far infrared, see Figure 2.1. Human bodies emit the radiation at a wavelength close to 10μm, whereas warmer objects emit more radiation with the shorter wavelength.

A unique feature of IR is the material transparency, being one of the most distinctive differences compared to visible light. For instance, semiconductor materials such as germanium and silicon, opaque in the visible, are transparent in the infrared region beyond 1.8μm and 1μm. On the other hand clear glass, transparent for visible light in its whole spectrum, it is opaque for infrared radiation for wavelengths over 2.5μm.

2.1.2 Thermal Sensors

Thermal sensors can be categorized into one of the following groups: those with cooled infrared image detectors or sensors with uncooled detectors.

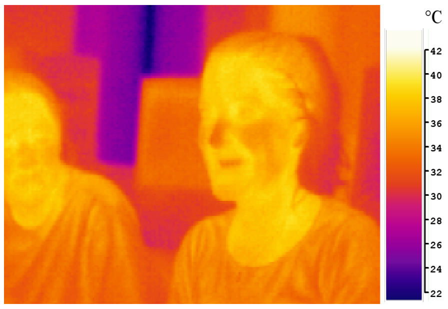
Cooled Infrared Detectors

The main feature of cooled infrared detectors is they are usually contained in a vacuum-sealed case cryogenically cooled. It is necessary to cool them down to temperatures in the range of 4K - 110K. This allows to achieve desired sensitivity through increased temperature contrast between the sensor and the object meant to be detected. In general, cooled infrared detectors are very expensive to manufacture and run because of their exploitation requirements. On the other hand, they are able to achieve higher resolution with better image quality. Due to their capabilities and high cost, they are mainly used for astronomy and military purposes.

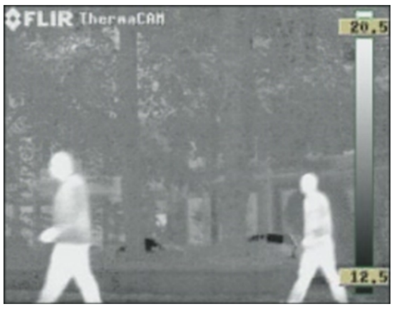
Uncooled Infrared Detectors

Uncooled infrared detectors are more popular than cooled equivalents mostly due to lower costs and smaller sizes. There are a number of advantages over cooled detectors: they are portable, easy to set-up and ready to work in just a few seconds. There is no need to cool them down since they operate at ambient temperature with the use of additional temperature control elements.

The technology employed is a subject to a target market. Most popular are pyroelectric sensors due to lower cost, although the number of applications is very limited - the electric charge on the output reflects to the change in temperature hence the continuous movement is desired. Moreover they are often limited to operate in low resolutions. These problems do not apply to detectors based on microbolometer technology. Bolometers change their resistance according to the heating source giving spatial image of the scene. Although they are more expensive, for certain applications such as detecting loss



(a)



(b)

Figure 2.2: Example images taken by IR camera. (a) Colour coded temperature scale. (b) Grey-scale temperature representation.

in insulated systems, observing the blood flow under the skin or overheating of electrical apparatus, they cannot be replaced by pyroelectric equivalents.

2.1.3 Thermographic Cameras

The difference in infrared radiation is transferred to the change of resistance, voltage or current, which is then compared with the value of operating temperature of the sensor. These values form a new infrared image which is then transferred from the camera using one of the available transmission protocols. A thermographic camera, also referred to as an infrared camera, is a device that forms spatial images based on IR radiation. Thanks to the ability to convert infrared energy onto an image, IR cameras can operate even in total darkness since the radiation of ambient light does not apply. Example pictures taken by IR camera can be seen in Figure 2.2.

Thermographic cameras are more expensive compared to traditional visible light cameras with the operating resolution considerably lower (typically at 160×120, 320×240). Thermal detectors from FLIR operating in 320×240 as of July 2011 are often priced above £5000 [5]. Modern detectors support resolutions up to 640×480 with the high end models operating in 1024×768 [6], aimed only for military and research purposes.

To make the visualisation of infrared imaging systems more clear and accessible for further processing, it is common practice to use density slicing and interpolation techniques [5]. Since the output of the infrared camera reflects the temperature changes in the FOV, colour coding applied corresponds to different temperature ranges and is a result of a post-processing operation, see Figure 2.2.

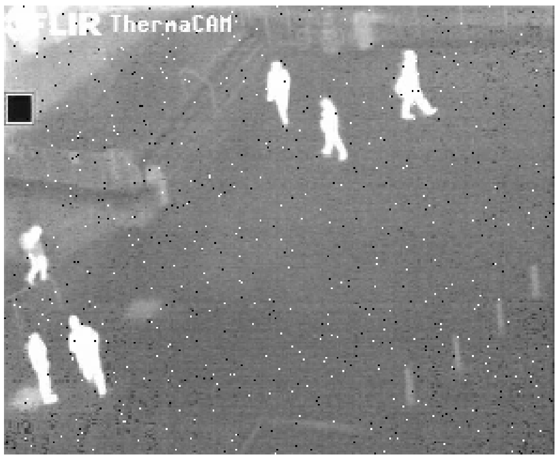


Figure 2.3: An example image frame from IR video surveillance system.

2.1.4 Applications

Until recently, thermal imaging using infrared cameras has been limited to military purposes. The first applications were developed for the Korean War in the second half of previous century [7]. Thanks to advanced optics, dynamically developed interfaces, rapidly falling prices and their increased portability, IR cameras are becoming much more popular for typical surveillance and control applications. There are a number of other applications where IR cameras can be efficiently used, for example detecting loss in insulated systems, tracing the source of heat in electrical apparatus or improving night vision.

An infrared camera is an excellent choice for they purpose of people detection. Many surveillance and control applications are based on infrared people detection since they are capable of working in low-light conditions. Moreover, they provide more accurate information for computer vision (easier to distinguish pedestrians, see Figure 2.3), comparing with cameras operating on the visible portion of the spectrum. They are widely used in security applications where permanent surveillance in both day and night time is needed [8]. Other useful applications are emergency systems for people counting in low or zero light conditions [9].

2.2 Digital Image Processing

This section gives a brief introduction to digital image representation and image processing techniques. Definitions and concepts used in this section are referred from [10, 11]. They shall be used as a theoretical background for the remainder of this document.

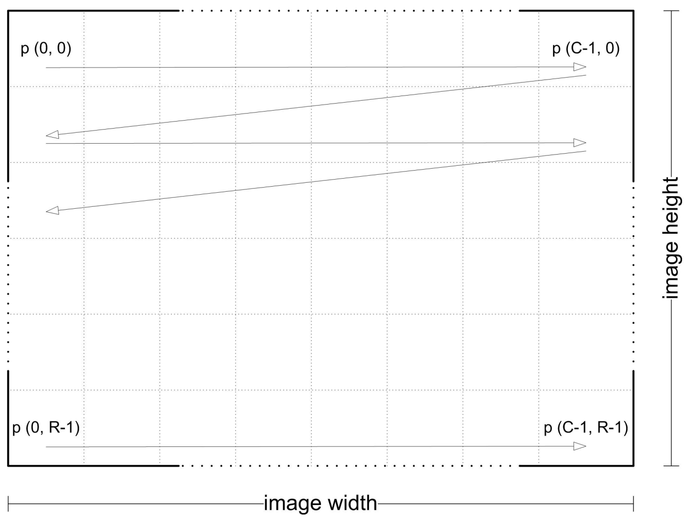


Figure 2.4: Digital image raster scan grid

2.2.1 Digital Image

Digital images are a pictures that have been converted into a binary format consisting of logical 0s and 1s. These images are represented by pixels aligned to a grid with rows and columns. The number of pixels in a row (R) and the number of pixels in a column (C) defines the image resolution. The value of pixel p located at (x, y) corresponds to the colour intensity of the picture at this particular location. Colour image analysis is beyond the scope of this thesis, only grey-scale images will be taken into consideration. In grey-scale images each pixel is assigned with a set of discrete values $I \in \{i_{min}, i_{max}\}$ representing different grey-scale levels from black to white. For an image with 8-bit grey-scale depth, each pixel is assigned with a value within a range from $2^0 - 1 = 0$ to $2^8 - 1 = 255$. Binary (morphological) images are particular case of grey-scale images where each pixel is represented by one bit: binary-0 or binary-1 corresponding to black and white respectively.

Based on to the information listed above, it is possible to calculate the weight of this particular digital image to check how much memory is required to keep the image within a digital system. The image size is defined by the number of R rows multiplied by the number of C columns (image resolution), multiplied by its grey-scale depth. An example QVGA image with 8-bit grey-scale would require $320 \times 240 \times 8 = 614400$ bits of memory. The same size binary image would require only $320 \times 240 \times 1 = 76800$ bits of memory. Memory requirements are often an issue for image processing embedded systems due to limited resources. This aspect will be discussed in detail further in this thesis.

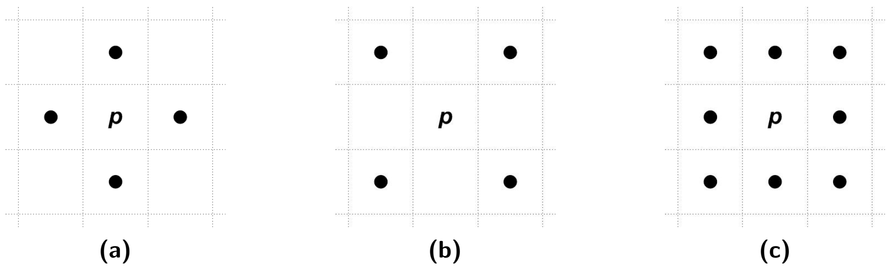


Figure 2.5: Pixel neighbourhood. (a) Horizontal and vertical neighbours of pixel p denoted by $N_4(p)$. (b) Diagonal neighbours of pixel p denoted by $N_D(p)$. (c) All the 8-neighbours of pixel p denoted by $N_8(p)$.

2.2.2 Raster Scan

Due to the nature of video processing systems, digital images are scanned pixel by pixel in raster scan mode. A raster scan is a popular scanning technique for raster graphics images. The scanning starts from the first pixel aligned in the top-left corner of the image with coordinates $(0, 0)$. It runs from the left to the right, line by line, until the last (bottom-right) pixel (R, C) is reached. An example of raster grid can be seen in the Figure 2.4. The raster scan is of key importance for all video processing embedded systems. Since image data provided to the system is aligned with raster scan, it would be a natural to process the data on active sample simultaneous with image acquisition. Further details on image acquisition and data handling will be provided in subsequent chapters.

2.2.3 Pixel Neighbourhood

According to the raster grid, pixel p at location (x, y) has two horizontal and two vertical neighbouring pixels at locations:

$$(x-1, y), (x+1, y), (x, y-1), (x, y+1),$$

these are called 4-connected of p and are denoted as $N_4(p)$. This can be seen in Figure 2.5(a). The pixel p at location (x, y) has also 4 diagonal neighbours at locations:

$$(x-1, y-1), (x-1, y+1), (x+1, y-1), (x+1, y+1),$$

which can be denoted as $N_D(p)$. This was illustrated in the Figure 2.5(b). Both horizontal and diagonal neighbouring groups of pixels are called 8-connected of p and denoted as $N_8(p)$, where $N_8(p) \in N_4(p) \cup N_D(p)$. When one or both coordinates (x, y) of pixel p are equal to 0, R or C, some of its neighbours fall outside scope of N_4 or N_8. This issue will be addressed further in this thesis.

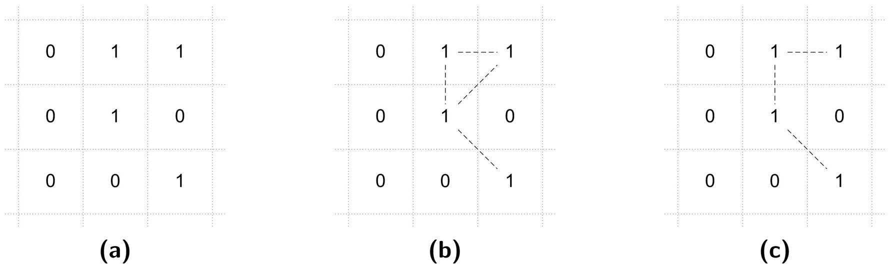


Figure 2.6: Pixels connectivity. (a) An arrangements of pixels. (b) Pixels 8-connectivity, connections shown by dashed line. (c) Connection ambiguities solved by m-connectivity

2.2.4 Pixel Connectivity

There are three different kinds of pixel connectivity (adjacency) for binary images between two pixels p_1 and p_2:

- 4-connectivity - when $p_1 = p_2 = 1$ and $p_1 \in N_4(p_2)$
- 8-connectivity - when $p_1 = p_2 = 1$ and $p_1 \in N_8(p_2)$
- m-connectivity (mixed connectivity) - when $p_1 = p_2 = 1$ and $p_1 \in N_4(p_2)$ or when $p_1 \in N_D(p_2)$ and $N_4(p_1) \cap N_4(p_2) \in \phi$

Mixed connectivity, a modification of 8-connectivity, prevents ambiguities which arise when 8-connectivity is used. An example of this type situation can be seen in Figure 2.6(b). Mixed connectivity eliminates multiple path connections and the result of its use can be seen in Figure 2.6(c).

In most applications for computer vision systems, only 4 and 8-connectivities are taken into consideration. Two examples of component labelling with 4-connectivity and 8-connectivity applied to the binary data from Figure 2.7(a) can be seen in Figure 2.7(b) and Figure 2.7(c) respectively.

2.2.5 Connected Components

Two pixels p_1 and p_2 are part of an L connected component in a B binary image when there is a path of connected pixels between pixels p_1 and p_2. The blob created by all the pixels within the set of pixels connected to p_1 and p_2 is called a connected component. Examples of connected components can be seen in Figure 2.7(b) and Figure 2.7(c).

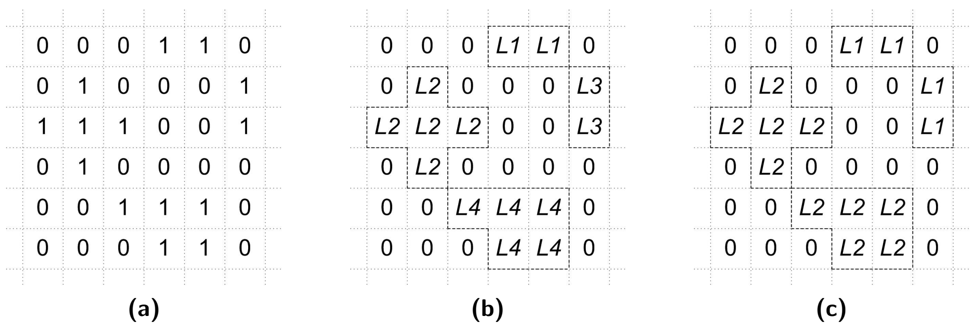


Figure 2.7: Connected Components (CC) labelled with 4 and 8-connectivity. (a) Binary image. (b) CC labelled with 4-connectivity. (c) CC labelled with 8-connectivity.

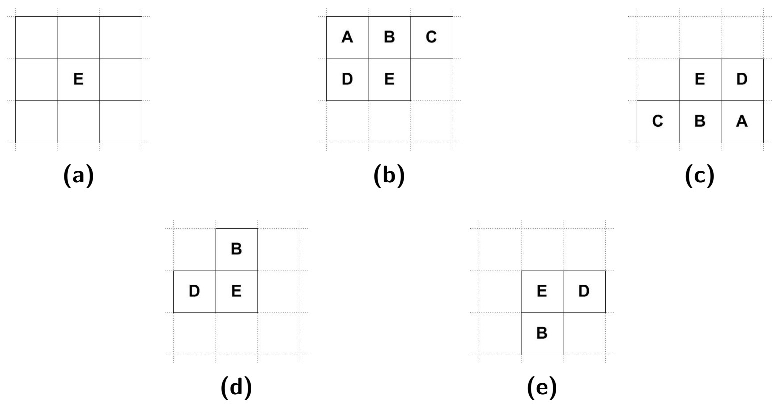
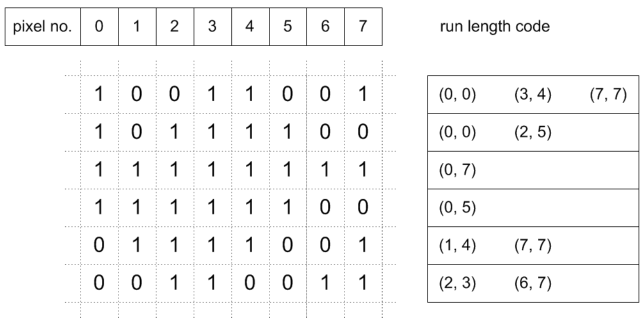


Figure 2.8: Scan mask. (a) 8-connected neighbourhood mask. (b) Forward 8-connected scan mask. (c) Backward 8-connected scan mask. (d) Forward 4 - connected scan mask. (e) Backward 4-connected scan mask.

2.2.6 Scan Mask

In order to create connected components in computer vision, an image has to be scanned through and all the adjacent pixels have to be detected. For this task, a scan mask is employed. The structure of 8-connected neighbourhood mask can be seen in Figure 2.8(a). The centrally placed point E refers to the pixel p located at (x, y) within the binary image B. All the other points are its 8-connected neighbours and are checked for adjacency. The scan mask is being shifted pixel by pixel, line by line - according to the raster scan. Recent algorithms for connected component analysis have reduced structure of the scan mask. The aim is to minimize the number of pixel operations. Scan masks for 8-connected and 4-connected analysis can be seen in Figures 2.8(b), 2.8(c) and Figures 2.8(d), 2.8(e) respectively.



pixel no.	0	1	2	3	4	5	6	7	run length code		
	1	0	0	1	1	0	0	1	(0, 0)	(3, 4)	(7, 7)
	1	0	1	1	1	1	0	0	(0, 0)	(2, 5)	
	1	1	1	1	1	1	1	1	(0, 7)		
	1	1	1	1	1	1	0	0	(0, 5)		
	0	1	1	1	1	0	0	1	(1, 4)	(7, 7)	
	0	0	1	1	0	0	1	1	(2, 3)	(6, 7)	

Figure 2.9: Run Length Encoding

2.2.7 Labelling

It is a fundamental feature of the computer vision system to be able to assign unique identifiers (labels) to different, disjoint connected components. In this process, all the pixels within the binary image B are analyzed in order to distinguish disjoint points between separate connected components. The classic connected component labelling technique employs the following steps:

– assign a label 0 if the pixel forms part of the background (pixel binary-0);

– if only pixel E was found as a foreground element (binary-1), assign a new label;

– if only one of the neighbouring pixels to the $E = 1$ was already labelled, assign its label to the current pixel;

– if two or more of the neighbouring pixels were already assigned with different labels, these labels need to be merged.

The labelling technique together with detailed description of popular algorithms for connected component analysis will be presented further in this thesis.

2.2.8 Run Length Encoding (RLE)

Run length encoding is an approach for encoding connected pixels (*runs*) within a single row of a binary image. Each run is represented as a starting and ending point within particular row. RLE was introduced as an approach to reduce the amount of data required for the processing. The main assumption is that binary-0 and binary-1 pixels continue within a row without a change over longer sections, therefore it is suitable for higher resolution images. An example of encoded data using RLE technique can be seen in Figure 2.9. The RLE-based approach can be also used to represent connected components.

2.3 Hardware Development and Implementation

There are a number of choices facing system designer while developing a video processing system. Hardware development requires multiple electronics devices such as Analogue to Digital and Digital to Analogue Converters (ADC and DAC respectively), processing unit, storage memory as well as Input/Output (IO) peripherals e.g. push buttons, slide switches, keyboard, monitor display.

The key element of the entire design is a processing unit. There is no ideal technology, it is always the matter of a trade-off between flexibility and performance. In this section alternative solutions will be presented and an overview of development process will be given.

2.3.1 Processing Unit

There is a wide range of processing units utilizing different technologies available on the market. Most common are listed below:

1. General Purpose Processor (GPP)
2. Digital Signal Processor (DSP)
3. Field-Programmable Gate Array (FPGA)
4. Application Specific Integrated Circuit (ASIC)

Each of the technologies has a specific set of features since they were designed for different purposes and certain type of applications. They differ in price, size, power consumption as well as development effort and processing power.

GPP

General Purpose Processors are commonly used in personal computers and mobile devices [12]. They support floating point operations, allow to run different software packages, including operating systems. Hence, they constitute great means as development platforms for system prototyping. Programming in one of the high level programming/scripting languages (C, C++, Matlab) is fast and supported with efficient debugging tools. If there is a need for further code optimization, critical code can be implemented at a lower abstraction level, e.g. C, C++ for Matlab or assembly for C, C++. However, this comes at a cost of flexibility and development effort.

DSP

Digital Signal Processors feature similar principle compared to GPPs with advanced architecture and additional instruction set for signal processing operations such as Multiply and Accumulate (MAC) or memory read with simultaneous addition [13]. An extended instruction set is a great advantage when comparing with GPPs. They are also smaller in size and feature lower power consumption therefore oriented on specific application within embedded system.

FPGA

Field-Programmable Gate Arrays are digital Integrated Circuits (ICs) containing configurable (programmable) blocks of logic along with configurable interconnects in between [14]. Hardware Description Language (HDL) and FPGA devices allow designers to quickly develop and simulate a sophisticated digital circuit, run it on a prototyping device, and verify the operation of the physical implementation [15]. Thanks to embedded macro blocks such as multipliers, dual port memory blocks and even embedded processor cores, signal processing algorithms may be implemented more efficiently compared to DSPs. Moreover, Hardware/Software (HW/SW) workload can be partitioned to run house-keeping tasks in software, whereas signal processing in parallel and pipelined fashion on hardware.

The main drawback of FPGAs is the development time, which is much longer compared to GPPs and DSPs due to low-level programming issues (Verilog or VHDL) and variety of system components such as interfaces, pin constraints or timing constraints. Moreover, testing and verification are more involved, hence require greater amount of time and effort.

ASIC

Development of the Application Specific Integrated Circuit can be equated with FPGA development in terms of programming, testing and verification [16]. The major difference is the full scalability provided by ASIC, up to a single logic gate. Hence, final design can be further optimized to meet specific packaging requirement, power dissipation can be reduced and even the overall throughput further increased thanks to the ability to run on higher operational frequency.

However, the back-end process is not a trivial task. It is often conducted by a group of engineers. There are great costs involved in setting up the mask for mass production therefore ASIC design shall be considered only for volume production.

Table 2.1: Comparison of different technologies for signal processing development.

	performance	power	reconfiguration	time-to-market
GPP	fair	poor	excellent	excellent
DSP	good	good	good	excellent
FPGA	excellent	fair	fair	good
ASIC	excellent	excellent	not applicable	poor

Summary

The comparison between different technologies is provided in the Table 2.1. It is based on various surveys [17–19] and comparison tools [20–22]. Such architectures are confronted under the following categories: performance, power, reconfiguration and time to market.

Performance determines the capability of a product. It can be measured using millions of operations per second (MIPS), millions of multiply accumulates per second (MMACS) or millions of cycles per second (MHz). The power criterion determines power consumption of the device which becomes increasingly important in portable devices. It also relates to the heat dissipation. Reconfiguration however gives the measure of the capability to modify or add features to meet changing requirements. Time to market is an important category in terms of business point of view. With the shrinking product life cycle it becomes a critical issue. Further details can be found in [17].

As can be seen, it is the matter of choice between performance and flexibility. Dedicated architectures (FPGA, ASIC) feature great processing power however they require more development effort compared to other technologies. Algorithm implementation comes as a trade-off between development time and algorithm complexity. On the other hand, software based solutions (GPP, DSP) are more flexible, therefore suitable for algorithm development and initial testing. In terms of time-to-market they are better than FPGA and ASIC, however at a cost of limited performance. Due to the lack of dedicated hardware and fixed architecture, they cannot compete with FPGAs and ASICs when it comes to computationally intensive tasks.

Specific applications, such as real-time video processing, require great amount of computing power. Due to high computational load, parallel and pipelined architecture is desired, whereas this can be achieved on DSP only in a limited scale. If the performance is a major concern, FPGA or ASIC shall be considered as a target platform.

FPGAs and ASICs differ in many aspects: physical size of the package, cost of a single chip, maximal frequency of the system clock, power consumption, development flexibility as well as back-end design effort. Considering the data from Table 2.1, ASIC turns out to perform better in most aspects - it is faster, smaller and requires less power. However, for prototyping purposes there is no other choice than FPGA due to high set-up cost of the ASIC production line. Once the design is verified and tested in test environment using software tools, it can be synthesized and mapped onto the FPGA for the lab testing on a real-time system. A successful implementation can be then resynthesized and optimized for ASIC design if mass production is considered.

2.3.2 Design Flow

It is required for a system designer to be familiar with design flow when implementing on FPGA/ASIC. Poor GPP or DSP implementation may lead to memory leakage or longer execution time, whereas even minor RTL-level design issues may significantly impact the overall system performance, e.g. drop in operational frequency from 400MHz to 100MHz [23]. Hence, the coding style in RTL is essential, it must suit target platform and comply with every step of the FPGA design flow: synthesis, mapping, placing and routing. A conceptual block diagram of the FPGA design flow can be seen in Figure 2.10.

HDL

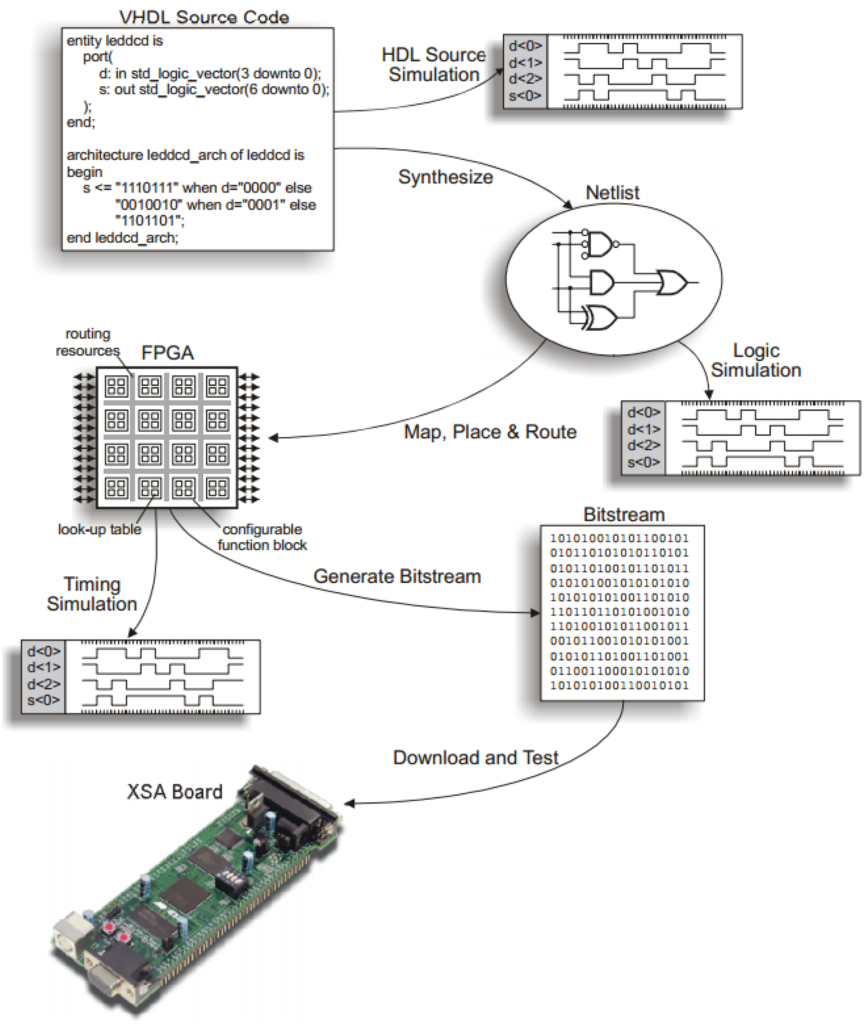
FPGA design flow in modern systems is fully automated and relies on software tools provided by different FPGA vendors. In order to achieve the highest performance it is strongly recommended to learn about the target device and follow certain rules specific to this technology [23]. Synthesis tools as well as map, place and root tools are based on algorithms to match the best possible coverage, however it is still up to the designer to make sure all the constraints are achievable and the asynchronous components comply with the overall system architecture. There is also a certain set of configuration rules for FPGA design related to the use of SRLs, multipliers, RAMs and logic elements [25, 26]. There is also a set of specific design techniques to make tools maximize performance of the device [23, 27].

Synthesis

Digital synthesis is a process when the HDL code is being transformed into the corresponding netlist, which is a standard form of representing the design at a low-level of abstraction. It constitutes a set of gate-level components and interconnections between them. One of the synthesis objectives is to optimize the structure of the design by reducing the number of redundant components [27].

Map, Place and Route

The process of mapping transfers the netlist into FPGA specific primitives. This is done according to specific technology standards and libraries. During the place and



```
entity leddcd is
   port(
      d: in std_logic_vector(3 downto 0);
      s: out std_logic_vector(6 downto 0);
   );
end;

architecture leddcd_arch of leddcd is
begin
   s <= "1110111" when d="0000" else
        "0010010" when d="0001" else
        "1101101";
end leddcd_arch;
```

```
101010010101100101
010110101010110101
010110100101101011
010101001010101010
101010101001101010
110110110101001010
110100101011001011
001011001010101001
010101101001101001
011001100010101010
101010100110010101
```

Figure 2.10: FPGA design flow [24].

route these gates are interconnected according to the netlist. Although the process is fully automated and entirely based on the output from the synthesis, the tool can be guided by specific parameters listed in the user constraints file.

Programming File

Once the design is already mapped and routed, a configuration file is being produced for the purpose of FPGA chip programming. It is a stream of binary data to indicate whether switches on the FPGA should remain open or be closed. The successfully generated file can be downloaded onto the target device for test and verification.

2.3.3 Development Platform

In order to verify and test the design, an FPGA development platform has to be selected. There is a wide variety of prototyping boards available on the market, for the purpose of this project the XUP V2P Development Board from Digilent Inc. [28] was chosen.

System Requirements

For the purpose of video processing application development, a large FPGA chip was required to avoid the need of system partitioning. Such applications often require a substantial amount of fast memory, preferably dual port blocks for buffering video streams. In order to run such application on the development board, multiple peripherals are also required. The check list of system components required for such implementation can be found below:

- large FPGA,
- substantial amount of embedded memory,
- VGA output,
- ADC converter,
- UART port,
- PIOs for debug and configuration,
- external PIOs for 3rd party components.

Platform Selection

At a time of project specification (late 2007), the best possible development board available for academic purposes was XUP V2P Development Board from Digilent Inc. This development platform fulfils all the requirements providing also additional system components for further project expansion. The board is equipped with Virtex-II Pro FPGA [26] from Xilinx providing sufficient logic for project development together with satisfying amount of embedded memory for high speed data handling. Further details regarding the development platform will be provided in the following chapters.

2.4 Pedestrian Detection

This section gives a general introduction into pedestrian detection processing flow. The aim is to emphasize the need for hardware accelerators in order to speed up the computationally intensive repetitive tasks. Detailed description and hardware implementation details will be discussed in the following chapters.

Pedestrian detection as a task can be split up into several related stages. The most common method is a two-step approach: segmentation followed by object detection. The segmentation is a pre-processing stage with the aim to reduce redundant information. Ideally, an image frame after the segmentation should not contain elements of background. Moreover, acquisition noise should be also already removed. Image pixels within the processing buffer consist of Regions of Interest (ROIs), likely correspond to pedestrians. During the detection, ROIs are analysed to check for the correlation with pedestrians. For detected objects some of the features can be extracted for the purpose of further processing such as classification or tracking. In this thesis both segmentation and object detection will be taken into consideration.

The two-step approach for pedestrian detection, compared to the blind approach, reduces computational time by limiting the number of scanned regions. It can be further split into smaller stages as follows, also depicted in the Figure 2.11:

1. Segmentation
    a) Background Subtraction
    b) Intensity Thresholding
    c) Noise Removal
2. Object detection
    a) Connected Component Labelling
    b) Feature Extraction
3. Classification and Tracking

The number of sub-stages is not fixed, it is determined by the application and algorithms applied. The efficacy of pedestrian detection system depends on both efficiency of individual processing units as well as suitable selection and coexistence of the processing chain as a whole.

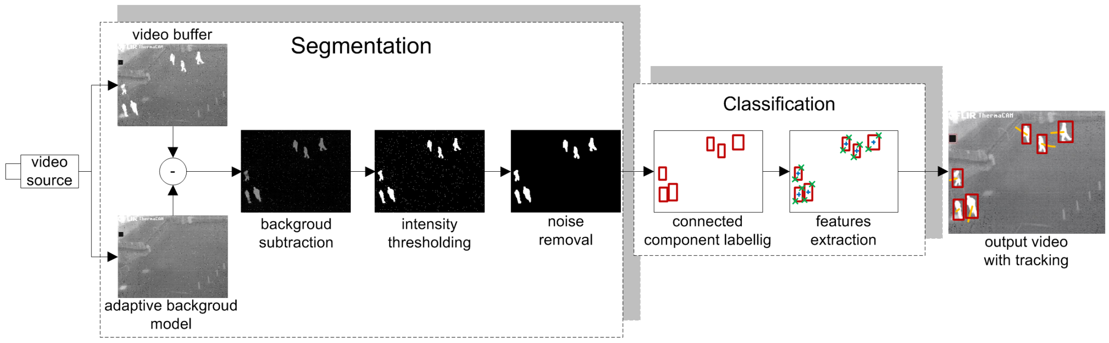


Figure 2.11: A block diagram of the data flow for a conceptual surveillance system.

2.4.1 Segmentation

During the first phase of pedestrian detection, an input image is processed to reduce the amount of information. This is a computationally intensive task with a great impact on the quality of object detection, classification or tracking. Efficient segmentation makes the classification work easier, in some cases even simple classification can lead to good detection rates, an example can be seen in [29]. Most common segmentation techniques are based on background subtraction, intensity thresholding and noise removal. Other intermediate processing stages can be also classified into segmentation, for instance image acquisition can reduce over a half of the incoming data during the colour space conversion [30]. Moreover, the colour space transformation can be also utilized for shadow reduction [31], in order to increase the chance of successful classification.

Background Subtraction

Background subtraction is a popular technique in many image processing applications, often it is a research area as its own [32–35]. The aim is to remove all the elements of the background leaving only ROIs further required by the classifier. There are a number of challenges facing this task: background image is not uniform, the pace of changes may also vary, etc. The choice of background subtraction technique often is the case of compromise between accuracy/quality versus computational load/implementation complexity.

Intensity Thresholding

The aim of the intensity thresholding is to reduce the number of data bits representing a single pixel. Most common approach is a binarization where the number of bits is effectively reduced to one per pixel. Thresholding as a segmentation step is efficient and cost effective for both GPP and DSP, and even more efficient when applied on application specific platforms. There are a number of techniques for calculating intensity criterion (local, global, average, temporal, adaptive, etc.). They will be further discussed in the following chapter.

Noise Removal

Noise removal, similar to background subtraction and intensity thresholding, is a common task in variety of image processing applications. The aim is to remove noise acquired during image acquisition. There are different types of noise. It can be either correlated or uncorrelated, with different distribution over the image frame. Noise removal is not a trivial task as it can lead to significant distortions if wrongly applied.

There are multiple noise removal techniques, such as linear smoothing filters, signal combination, anisotropic diffusion or non-linear filters. The choice is often application specific, it also depends on the target processing platform. For the purpose of this project linear smoothing based on morphological filters was implemented. This choice will be further discussed together with implementation details provided further in this thesis.

2.4.2 Object detection

Object detection is a processing step when the actual pedestrian detection takes place [36]. The purpose is to process the data provided by segmentation unit, typically a stream of binary pixels, in order to distinguish objects within the image. Furthermore, specific features for individual objects can be extracted, such as position, height, weight or Centre of Gravity (CoG).

Connected Component Labelling

The purpose of Connected Component Labelling (CCL) algorithms is to scan binary images in order to distinguish disjoint groups of pixels (objects) and assign them with individual labels. Once they are assigned with labels, each of the components can be further processed as individual object.

Feature Extraction

Labelled objects can be further processed to calculate their features for the purpose of classification, tracking or statistical analysis. During this processing step, a variety of different features can be extracted such as position, width, height or centre of gravity, etc. Such data can be used to classify individual objects whether they likely to be pedestrians or not.

2.4.3 Classification and Tracking

The aim of classification and tracking is to locate an object (or multiple objects), identify as a pedestrian and keep track of its location in consecutive video frames. Tracking is based on extracted features calculated on labelled objects. It is a post-processing step, based on data gathered during classification. However, according to the recent research, tracking can also improve detection rate in the process of object recognition, therefore it can be considered as additional classifier [37].

Both classification and tracking are natural extensions for pedestrian detection systems. However, they are not in the scope of this thesis.

2.5 Conclusions

In this chapter an introduction into image processing, infrared radiation, FPGA development and pedestrian detection was given. The aim of this chapter was to introduce the reader with technology and processing techniques used for the purpose of this research. Subjects introduced in this chapter will be further discussed in individual chapters.

CHAPTER 3

Background Segmentation

3.1 Introduction

In the modern world the number of applications designed for autonomous human detection grows rapidly. Such applications vary from basic human detectors based on elementary image sensors to complex processing platforms employing Infrared (IR) cameras. Although systems based on IR detection were mostly limited to military applications for surveillance purposes, due to falling prices of IR imagers they are becoming popular also in other markets. IR detectors, well known for their capabilities to detect pedestrians in low light conditions, gradually replace CMOS and CCD sensors for specific applications.

In general, human bodies keep higher temperature than the surrounding environment. The contrast level may vary in different weather conditions, for example a sudden change in sun exposure may reduce the contrast between pedestrians and their surroundings, making them less visible to IR camera. Tasks such as people detection or people counting based on low-resolution detectors, such as the Irisys people counters, make an extensive use of the features extracted by the pyroelectic sensor, therefore the decision making (detection) process is relatively straight-forward and does not require great amount of processing power [38]. More involved applications in the area of pedestrian detection and tracking also use this technology as a sensing element. Its main drawback is limited pixel resolution (typically up to 16×16). This amount of data can be processed within embedded system in real-time applications by MCUs [39].

Higher resolution pedestrian detection systems, often associated with surveillance systems, feature both types of image sensing devices, IR and CMOS (or CCD). The choice of technology used is highly related to the application and environmental conditions, e.g. for the purpose of traffic analysis devices operating mostly in day light, a colour sensitive CMOS image sensor would be the right choice [40]. However, the choice is not as obvious in case of pedestrian detectors, used for security applications. A typical colour image sensor can operate in higher resolution than IR, it can also benefit from the RGB or YUV colour component information commonly used for shadow removal or background modelling [31, 41]. It is also necessary to mention that although IR detectors are becoming cheaper, they are still relatively expensive. However, if the application requires higher sensitivity and is requested to operate in low light or even no light conditions, an IR camera has significant advantages [36].

Due to high demand for advanced autonomous security applications, the majority of outdoor surveillance systems are based on IR or use it for visible-infrared fusion [36]. They are also employed in a number of other applications, such as transportation, robotics or home automation. For the purpose of this thesis, one of the high resolution IR cameras was used. From the Digital Signal Processing (DSP) point of view, there is no difference between visual and IR sensors; the data provided can be processed using similar techniques. However, due to the fact different information can be extracted, some of the algorithms being popular for visual image processing cannot be applied to the IR video stream. This issue will be also addressed in the following subsections. An implementation of the autonomous surveillance system operating in real-time on a video stream is a mixture of a number of DSP techniques. A block diagram of an example application data flow is depicted in Figure 3.1.

This chapter concentrates on background segmentation. As can be seen in Figure 3.1, segmentation (adaptive background model + background subtraction) constitutes one of the initial processing stages, hence the quality of results provided is critical for further processing. Background segmentation is a popular research area, giving a wide range of techniques to be used for this task. The trade-off between the quality of data produced and the required processing power is a major concern. Often, algorithms featuring good performance require high computing power, frequently not efficient for hardware imple-

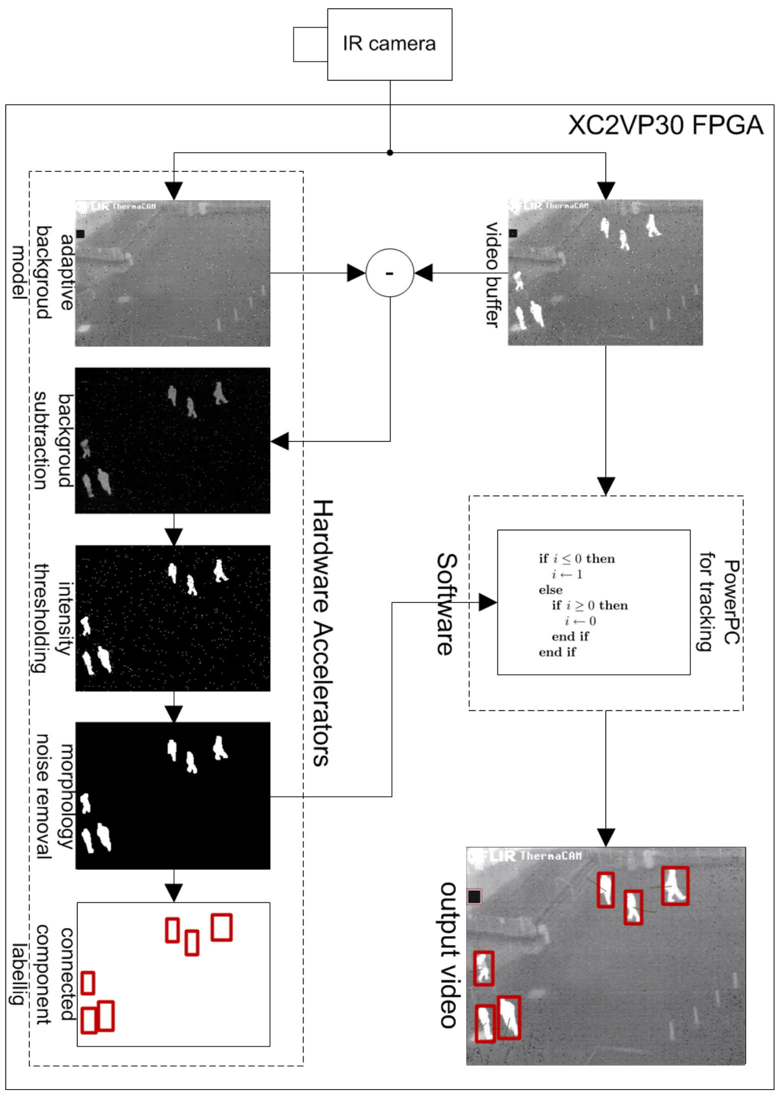


Figure 3.1: A block diagram of the data flow for a conceptual surveillance system.

mentation, e.g. long floating point operations. In this chapter major techniques will be introduced together with analysis of their usability for FPGA implementation.

3.2 Review of Background Subtraction Techniques

A separation of the foreground Regions of Interest (ROIs) from the background is a key processing step in most pedestrian detection systems. The aim of background subtraction is to significantly reduce the amount of processing data. A common approach is to binarize separated ROIs afterwards, hence the information is reduced to 1 bit per pixel. This process can be seen in Figure 3.1.

There is a wide variety of different techniques for background segmentation, three major groups can be distinguished:

- intensity thresholding,
- temporal difference,
- background modelling.

3.2.1 Intensity Thresholding

The intensity thresholding for pedestrian detection is a popular segmentation technique used in many systems based on IR cameras [8, 9, 29, 42, 43]. Due to the fact pedestrians in most cases are considerably warmer than the surrounding background, it is possible to reduce the amount of data by filtering input images using a threshold level factor. The separation is made on the pixel basis, therefore only the current image frame is required. This process can be described with the following equation:

$$B(x, y) = \begin{cases} 1 & \text{if } I(x, y) > th, \\ 0 & \text{if } I(x, y) \leq th. \end{cases} \tag{3.1}$$

where B is a binary image after thresholding, I is an input greyscale image, th denotes level of thresholding and (x, y) gives coordinates of the pixel location. In the simplest approach, this technique may considerably reduce the amount of data already during the image acquisition with no additional latency, in some cases leading to good segmentation results. However, as it becomes obvious, badly calibrated system (th too low or too high) will not be able to provide data suitable for further processing. An example of such situation together with image histogram pointing on the balanced value of th can be seen in Figure 3.2.

Global Average

The global average is a variation of intensity thresholding aiming towards an adaptive approach [29, 44]. It is commonly used in systems operating in constantly changing environmental conditions. It is based on the average intensity for each frame, calculated as follows:

$$th = \alpha \times \sum \frac{I(x, y)}{B_{row} \times B_{col}}. \tag{3.2}$$

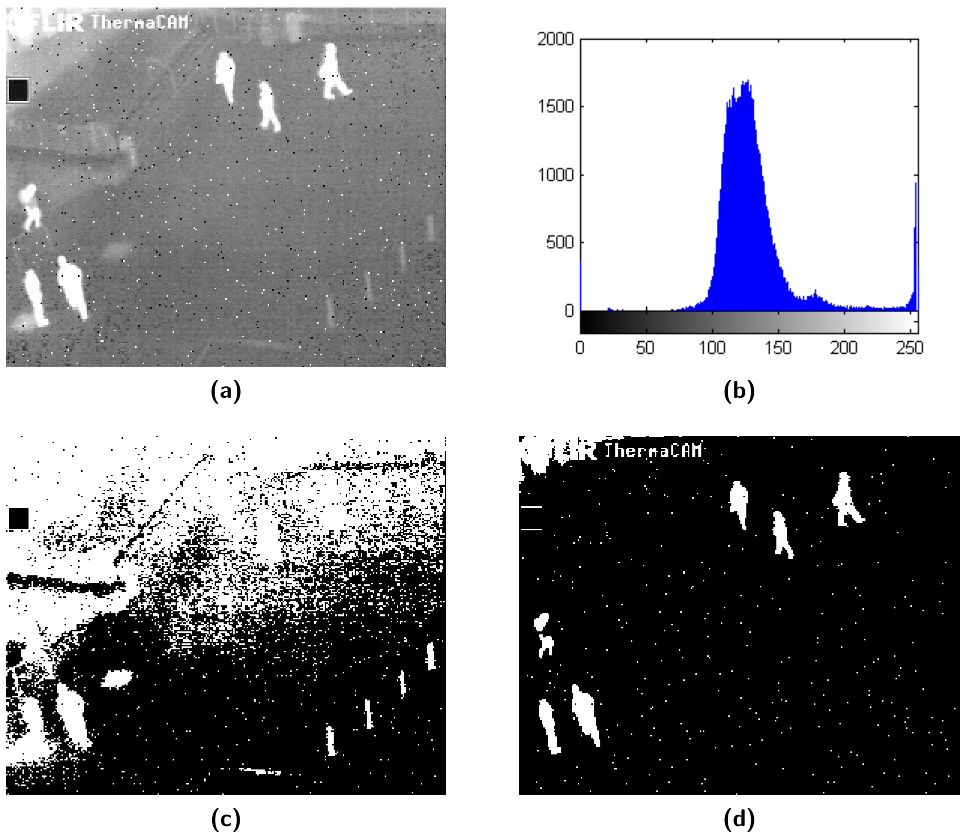


Figure 3.2: Background segmentation based on intensity thresholding. (a) Greyscale input image. (b) Histogram calculated for the input image. (c) Result of segmentation; input image thresholded at a peak level of the histogram $th = 128$; threshold level too low, results include both pedestrians and elements of the background. (d) Result of thresholding with $th = 240$; although th is at its optimal level, the output image reveals all the drawbacks of this technique such as bright elements of the static background (camera logo).

where $(B_{row} \times B_{col})$ determines the number of image pixels, whereas $I(x, y)$ gives the grey scale value of each image pixel. Value α is a fixed coefficient specified by the system operator during the initial calibration. The th is calculated for each image frame and applied to the subsequent one.

Adaptive Local Thresholding

The adaptive local thresholding is more advanced form of segmentation comparing with previously described intensity thresholding techniques [9]. It is based on the thresholding value th being calculated from two diagonally positioned pixels according to the following equation:

$$th = \delta + \alpha \times \sum |I(x, y) - I(x - 1, y - 1)|, \tag{3.3}$$

where both δ and α are calibration values determined during the prior experimentation. Once the th is calculated, it is applied to image data as follows:

$$B(x, y) = \begin{cases} 1 & \text{if } |I(x, y) - I(x - 1, y - 1)| > th, \\ 0 & \text{if } |I(x, y) - I(x - 1, y - 1)| \leq th. \end{cases} \tag{3.4}$$

Such comparison does not require a great amount of hardware resources. However it implies the use of intermediate storage for diagonal pixel values. It is a major drawback of this approach. Moreover, when processing higher resolution images, this approach does not provide significant improvement in quality of output data due to low contrast between neighbouring samples.

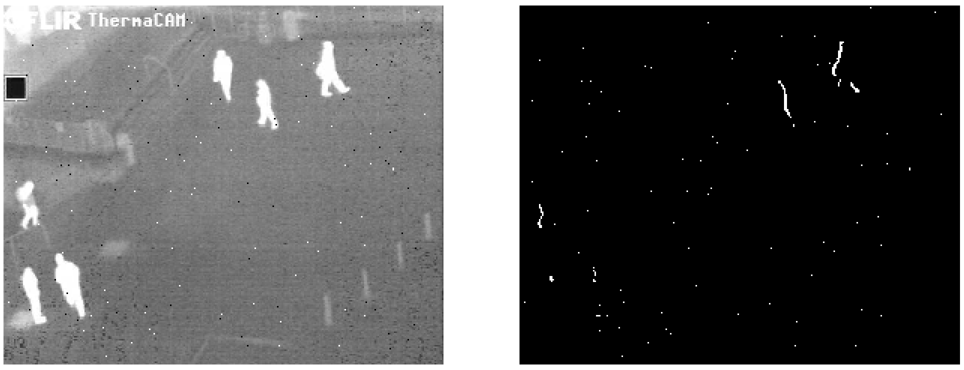
Intensity Thresholding Summary

The major benefit of background segmentation based on intensity thresholding is the processing speed and low memory requirement - no need for intermediate storage for reference image. However, the number of applications is very limited - it can be successfully applied only to IR based processing systems, for certain applications. Although it mostly features high pedestrian detection rates, it is limited to a particular number of case scenarios (e.g. pedestrian must be warmer than the background). Moreover, it does not differentiate other warmer objects from the background, such as street lamps or cars therefore it can not be applied as a stand-alone processing unit.

The intensity thresholding segmentation based on running global average gives reasonably good intermediate results. It is also suitable for robust and efficient hardware implementation for low cost devices. Thanks to low complexity and high processing speed (real-time performance), it should be considered during the system implementation, possibly as part of higher complexity algorithm.

3.2.2 Temporal Difference

The principle of the temporal difference approach for background segmentation is based on two successive images and comparison between them. Such comparison is performed on the pixel basis, line by line, whereas results are provided in the coherent form of another image being the difference between them. General description of the algorithm in the mathematical form can be described with the following equation:



(a) (b)

Figure 3.3: An example of segmentation output using temporal difference. (a) Input image frame. (b) Results of the subtraction between two consecutive image frames.

$$B(x,y) = \begin{cases} 1 & \text{if } |I_t(x,y) - I_{t-1}(x,y)| > th, \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

where $I_{t-1}(x,y)$ is the pixel at a location (x,y) from the preceding frame and th is the threshold value, determined during initial calibration. An example output frame of temporal difference in its basic form can be seen in Figure 3.3. Other variations of temporal difference approach for background segmentation introduce different pre-processing techniques applied on both image frames to be subtracted.

Noise Estimation

An interesting approach for ROIs segmentation described by Alexandropoulos et al. [45] takes advantage of noise estimation within the image followed by brightness normalization. The variance and the average value of pixels per block is calculated, followed by global comparison. Block size (here 8×8) can be determined by the image frame resolution and available system resources. Thanks to block processing, this technique allows for real-time performance.

Homomorphic Filtering

The homomorphic filtering is based on the algorithm proposed by Lou et el. [46]. The filtering is performed on illumination invariant local components in order to distinguish foreground objects from the background. It is based on the assumption that illuminance distribution is gradual over the image frame therefore it is located in the low part of the frequency domain, whereas reflection components will occupy higher part. Based on high-pass filtering followed by Bayesian estimation, foreground objects can be extracted under lighting changes without shadows. This approach, however, is less relevant in the IR detection (no shadows).

Temporal Difference Summary

The major problem temporal difference techniques struggle with is the varying speed of moving objects. Assuming the segmentation is performed on the frame rate in a real-time, slow changes may not be detected, leading to poor detection rate.

The temporal difference for background segmentation is efficient for hardware implementation. Since it operates on two consecutive image frames, the memory requirement is relatively low, making it suitable for memory-limited embedded system. Moreover, in order to increase detection rate, this technique can benefit from integration with one of the intensity thresholding algorithms at a very low cost.

There are also other techniques based on the temporal difference developed over the time, not included in this subsection. They introduce more involved mathematical models leading to complex implementations. However, temporal difference algorithms suit only limited number of applications and such increase in hardware complexity can not be justified with marginal improvement in segmentation quality [47–50].

3.2.3 Background Modelling

Amongst other methods for background segmentation, subtraction of the background model from the current image frame gives substantially more information about new objects entering the Field of View (FOV) [11, 32, 33, 51]. The principle of this technique is based on frame differentiation where one of them is a model of the background, containing information about the changes within the scene in time. An example of the background model together with extracted foreground elements can be seen in Figure 3.4. Results obtained from the subtraction are classified as ROIs if they exceed a pre-defined threshold level. This can be described with the following equation:

$$B(x,y) = \begin{cases} 1 & \text{if } |I(x,y) - M(x,y)| > th, \\ 0 & \text{otherwise.} \end{cases} \tag{3.6}$$

where $M(x,y)$ is a pixel of the background model at a location (x,y), $I(x,y)$ is a grey-scale source and $B(x,y)$ is a binary output pixel.

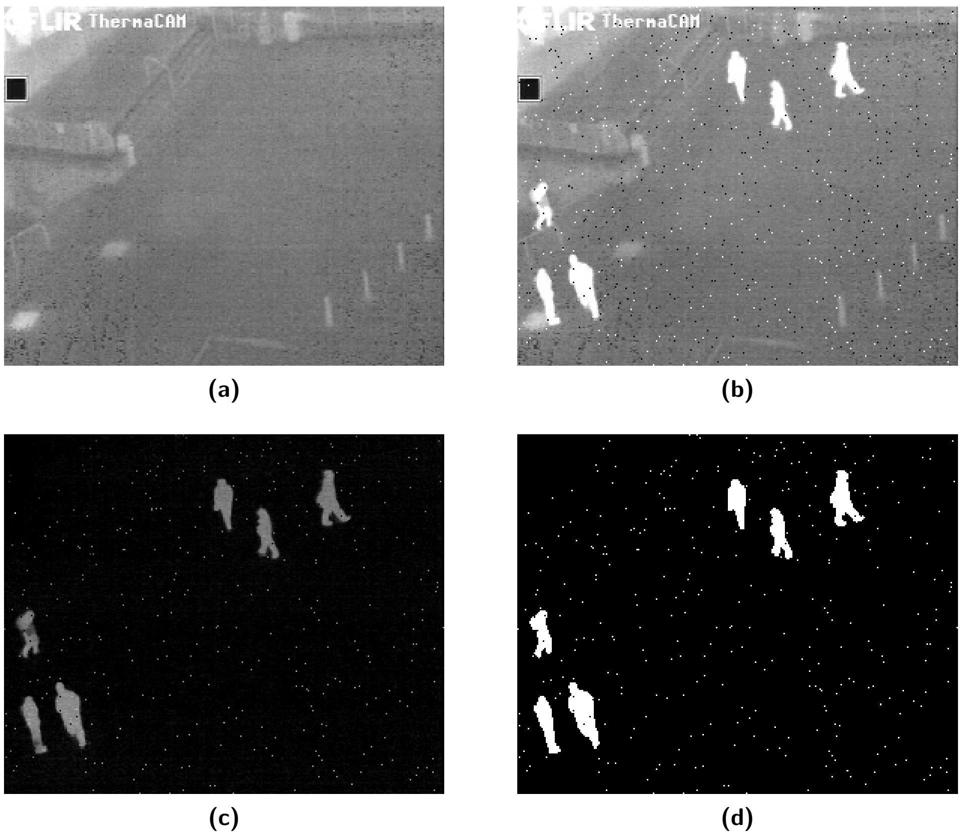


Figure 3.4: Segmentation based on the background modelling. (a) Background model. (b) Input image frame. (c) Results of the background subtraction. (d) Background subtraction after thresholding.

Although it seems to be a trivial task, the background subtraction may lead to false detection in a very short time if the background image (model) is not updated accordingly to the environmental changes. Such a processing must handle gradual illumination changes dependent on the time of the day as well as moving background objects which should not be considered as ROIs forever after [52]. Hence, a number of techniques for updating the background model have been proposed in the literature, where the most popular approaches will be presented below.

Temporal Median Filter

This approach for background modelling is based on the median filter commonly used in image processing for noise filtering in spatial domain [11]. A background reference image is created using a number of pixel values from consecutive image frames [53]. Once the background model is ready for the processing, the median is applied, hence the need for captured data to be sorted in a numerical order.

In order to distinguish foreground elements from the background model, an irreducible test set needs to be acquired. This requires a large buffer to store n values per pixel location, where n must be greater than 50 (typically 50 to 200) [54]. Since this requires a great amount of memory, an alternative approach was suggested where background model is based on the sub-sampled data source with n_s elements, where $n_s \subset n$ [55].

Background separation based on median filtering does not provide good segmentations results for all the case scenarios. It can not be configured (at a reasonable cost) to cope with both slowly as well as rapidly changing environmental conditions. Moreover, it requires a large amount of operational memory to store the test set for median processing. Median filtering also implies data set must be sorted in a numerical order. This adds additional complexity due to the fact sorting is not a trivial task in hardware, alternatively it requires a large number of clock cycles to complete it in the brute force search fashion.

W4

W4 is an independent surveillance system for people detection and tracking with an interesting approach for background segmentation [56]. This technique uses a median filter during the learning phase to distinguish moving pixels from stationary pixels whereas the background frame is modelled with the use of three parameters for each pixel: minimum $m(x, y)$, maximum $M(x, y)$ and the maximum illumination difference $d(x, y)$ between two consecutive frames. This can be written as follows:

$$\begin{bmatrix} m(x,y) \\ M(x,y) \\ d(x,y) \end{bmatrix} = \begin{bmatrix} min\{V_i(x,y)\} \\ max\{V_i(x,y)\} \\ max\{|V_i(x,y) - V_{i-1}(x,y)|\} \end{bmatrix}$$

where V is an array containing N consecutive images, $V_i(x, y)$ gives the intensity of the pixel located at (x, y) for the ith frame of V.

The classification process determines whether the current pixel is a foreground or background pixel based on the following:

$$B(x,y) = \begin{cases} 0 & \begin{cases} I_t(x,y) - m(x,y) < th \cdot d_\mu, \\ I_t(x,y) - M(x,y) < th \cdot d_\mu, \end{cases} \\ 1 & \text{otherwise.} \end{cases} \tag{3.7}$$

where th determines the threshold level and the d_μ gives the median of the largest interframe absolute difference over the entire background model.

This technique gives good segmentation results at a very low cost of two subtractions and comparisons per image pixel. The memory requirement is much lower than previously described median filtering-based approach, however it is still relatively high for implementation within memory-limited embedded systems.

Selective Running Average

The principle of the adaptive background modelling using a running average is based on the mean illumination value calculated for every image pixel in time [44]. Unlike the temporal median filter technique, there is no formal requirement to store n consecutive samples in order to calculate the mean value. Running average is computed on the incoming data rate and can be calculated as follows:

$$\mu_t = \alpha B_t + (1 - \alpha)\mu_{t-1}, \tag{3.8}$$

where μ_{t-1} is the average computed in the previous image scan and the α factor is an empirical weight (learning rate) often chosen as a trade-off between stability and quick response. The learning rate α is a fixed number and can be obtained during the initial calibration. It determines the pace foreground objects are incorporated to the background.

In order to keep the background model cleaner/sharper while reducing the number of memory accesses, a selective adaptation approach can be applied. According to the literature [57, 58], foreground segmentation gives better results when background model is updated only for pixels with negative segmentation, otherwise remains unchanged.

The selective running averaging gives acceptable detection rate compared to other techniques. Such an implementation does not require great amount of hardware resources whereas memory requirement can be kept low. Since the sufficient model can be represented with a single value per image pixel, the amount of required memory does not exceed the size of the input image buffer.

Predictive Filters

An implementation of predictive filters for background segmentation is an attempt for foreground detection based on the prediction criterion [59, 60]. Such a criterion is calculated as a probabilistic measure based on the recent samples using one of the predictive filters, e.g. Wiener [52] or Kalman [61, 62]. Unlike above mentioned techniques for background modelling, predictive filters provide additional measure (variance), describing how the incoming value may differ from the background. Such an approach gives closer estimation therefore the false positive detection rate is further reduced.

There is a variety of techniques for background segmentation using predictive filters. An introduction into linear prediction can be found in [59]. The general approach is to estimate the probabilistic prediction of the current background image using prediction coefficient and incoming data. If the current pixel significantly deviates from its background reference, it will be reported as part of the foreground. For the purpose of calculating prediction filters, a number of previous covariance samples is needed. This implies a large data storage which constitutes main drawback of implementation based on predictive filters. According to the literature, hardware implementation based on the Kalman filter [61, 62] is more efficient in terms of memory usage comparing with its Wiener equivalent.

Mixture of Gaussians

More recent algorithms for background modelling support multi-modal case scenarios thanks to a larger number of statistical distributions used to model the background [63]. An example of such technique is Mixture of Gaussians (MoG) introduced by Stauffer and Grimson in [41]. It attempts to handle quasi-static background scenes as well as gradually changing scenes and regularly moving objects such as branches of a tree or escalators.

This algorithm is based on multiple Gaussian distributions (typically 3 to 5) used to describe each pixel of the model. Every component of the RGB colour space is modelled with a separate Gaussian, with an assumption they are independent, however they have the same variances (to ease computational complexity). A different weight ω_t is assigned to the distribution (giving the total sum 1) and adjusted for the matching model according to:

$$\omega_t = (1-\alpha)\omega_{t-1} + \alpha, \tag{3.9}$$

where α is a learning rate determined during the system calibration. If the model does not match, the learning rate α is further subtracted from the Equation (3.9). Two other parameters such as mean μ and variance σ can be calculated as follows:

$$\mu_t = (1-\rho)\mu_{t-1} + \rho S_t, \tag{3.10}$$

$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(S_t - \mu_t)^T(S_t - \mu_t), \tag{3.11}$$

where ρ is a second learning rate and S_t provides the colour components of the input signal. Then, the Gaussian probability density can be computed as follows:

$$\eta(S_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{1}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(S_t-\mu_t)^T \Sigma^{-1}(S_t-\mu_t)}, \tag{3.12}$$

where Σ is the covariance matrix of the ith Gaussian in the mixture.

The probability of the current pixel estimated using K Gaussian distributions for each pixel can be calculated using Equations (3.9)(3.10)(3.12) according to the following equation:

$$P(S_t) = \sum_{i=1}^{K} \omega_{i,t} \cdot \eta(S_t, \mu_{i,t}, \Sigma_{i,t}), \tag{3.13}$$

The MoG for background subtraction gives good results, particularly in scenarios with repetitive background motion. What is important, it fits best systems where a large amount of information can be extracted from the coded source, for instance RGB or YUV input signals are desired. In case of a single Gaussian distribution this algorithm does not differ much from the running average based on a probability distribution. MoG is suitable for hardware implementation with further modifications [64], however a large memory bandwidth constitutes a major bottleneck.

Kernel Density Estimation

A different approach for background modelling using Gaussian distribution was presented by Elgammal *et al.* in [65]. Here, a probabilistic distribution is used to model the frequency of background samples not being classified as foreground objects. The algorithm is based on the principle that such values can be presented in the form of histogram for n recent samples.

The Kernel Density Estimation (KDE) is a non-parametric model giving the detection probability as a sum of the most recent n Gaussians according to the following equation:

$$P(S_t) = \frac{1}{n} \sum_{i=1}^{N} \eta(S_t - S_i, \Sigma_{i,t}), \tag{3.14}$$

In contrast to MoG, multiple distributions are used to model data samples over the number of recent frames instead of modelling a single pixel using a mixture of Gaussians. For each frame a data sample from the frequency histogram is assigned with a single Gaussian. Therefore, a large storage is required to keep all the distributions used

Table 3.1: Comparison of background segmentation algorithms based on background modelling with 1-5 scale referring to poor/low-strong/high respectively.

	accuracy/ quality	computational/ implementation complexity	memory requirement
Temporal Median Filter	2	4	5
W4	3	2	2/3
Selective Running Average	3	1	1
Predictive Filters	3/4	4	3
Mixture of Gaussians	4	4/5	4
Kernel Density Estimation	5	5	5

by the probability function. The memory requirement of this algorithm is far beyond limitations of a standard embedded system. It is also demanding in terms of computing power due to large amount of processing data. However, this technique reports very good segmentation results, even operating on a single Y component from the YUV colour space.

Background Modelling Summary

A variety of background segmentation techniques was described in this subsection. All of them feature good quality of output data comparing with techniques discussed in previous subsections, however these are more computationally involved and operate on higher bandwidths. It is difficult to determine which one features the highest segmentation rate since this factor is highly influenced by specification of the application, can be also specific to the particular test case. Hence, different reviews may lead to contradictory opinions and conclusions can be subjective [32, 33, 51].

Based on experimental simulations supported by the literature, background segmentation algorithms can be classified in regard to speed, complexity and memory requirement as poor/low - strong/high using 1-5 scale respectively. Such a comparison is shown in Table 3.1.

The accuracy / quality is good and acceptable for all of the algorithms. Those marked with lower grade do not handle multi-modal scenarios, therefore they are not as responsive to both fast and slow changes to the background reference image. Although parametric algorithms (predictive filters, MoG) feature better quality of output data for scenes with repetitive movements, the error rate of a single scalar based algorithms in such cases may be filtered out with noise removal applied within the data flow chain. The major drawback of a single scalar based model is its relatively poor flexibility - it can be very responsive with the background model updated immediately, alternatively it can be set to change the model slowly, to avoid sudden changes in the background model (not susceptible to noise). This applies when the learning rate is set too high or too low, respectively.

Both computational / implementation complexity and memory requirement are closely related. Although computational complexity may be compensated by parallel or pipelined architecture, the memory bandwidth in most cases constitutes the main bottleneck. Both techniques with the highest mark, median filter and KDE, will be inefficient in terms of hardware implementation since a vast majority of processing power will be consumed by memory controllers on data handling. Other techniques, such as predictive filters or MoG are also memory demanding, however the memory bandwidth may be significantly reduced with a couple of assumptions keeping the detection rate still at a high level [64]. Unquestionably, algorithms based on a single (running average) or triple scalar (W4), are the most efficient in terms of memory utilization. The data handling process is also relatively straightforward therefore they are preferable for hardware implementation.

3.3 Algorithm Overview

Although background modelling techniques outperform other algorithms in terms of segmentation quality, they are memory demanding, in some cases also not suitable for hardware implementation. Moreover, the choice of algorithm is application specific and must be tailored to particular environment. Therefore, due to limited information that can be extracted from the source data (single component luminance IR signal), multi-modal techniques are not much of benefit for pedestrian detection systems using infrared cameras. In such cases, the main feature of mixture-based models can not be successfully applied. However, in such environment a thresholding technique can be utilized, often underestimated by system designers. This add-on to the background model can significantly improve the detection rate at a very low cost.

Based on the experimental work in the simulator and the review provided in previous subsections, a unique background segmentation technique is introduced. It is based on a selective running average background modelling in conjunction with a histogram-based feedback loop controller for adaptive thresholding, developed for the purpose of this thesis. Hence, the system keeps memory requirement relatively low, together with computational and implementation complexity, however it is able to quickly correspond to both fast and slow environmental changes, covering all the scenarios for pedestrian detection application.

3.3.1 Background Modelling

According to the review provided in the previous section, the adaptive background modelling based on the running average features good segmentation rates as well as low implementation complexity. The main drawback of this algorithm is the lack of flexibility - it does not support both fast and slow environmental changes at the same time. Thanks to the learning rate α, the updating process can be customized and adjusted to the particular application however it should be rather used for slow or very slow model adjustments. Higher learning rates can cause significant loss of information stored in the background model (foreground objects incorporated to the background too quickly).

The efficacy of background updating can be further increased by pre-classification of input data in the process of selective running average. If the pixel is considered to be a part of the background, the learning rate is applied and the corresponding pixel of the background model is updated by a small percentage of incoming data. However, if an input pixel was classified as a foreground element (ROI), this particular background pixel will remain unchanged. Therefore, the background model can be updated according to the following:

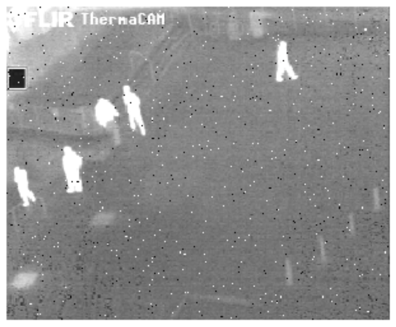
$$M(x,y) = \begin{cases} (1-\alpha)M_{t-1}(x,y) + \alpha I_t(x,y) & \text{if } I_t(x,y) \subset \text{background}, \\ M_{t-1}(x,y) & \text{otherwise.} \end{cases} \tag{3.15}$$

where $M(x,y)$ is a pixel of the background model and $I(x,y)$ refers to input data. By applying selective running average, the background image is sharper, this also increases further detection rates.

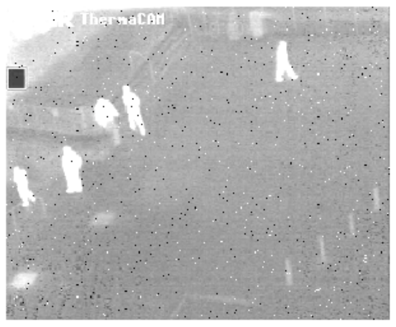
The motivation behind the selection of this algorithm is its relatively low memory consumption and a potential for further improvements with additional techniques applied. The total memory utilization is three times lower comparing with the second memory efficient algorithm W4. It provides satisfactory segmentation rates, competing also with other, even more involved algorithms.

3.3.2 Histogram Calculation

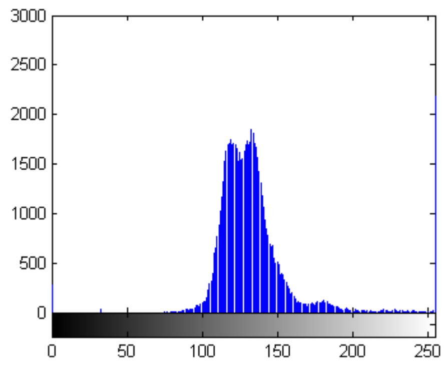
Histograms are commonly used in a number of spatial domain processing applications [11, 66]. They provide great insight into the colour or grey-scale information therefore are often used for image enhancement or statistical analysis. Basic histogram analysis can also be a valuable source of information for segmentation purposes. As an example, sudden illumination changes depicted in Figure 3.5 can be quantized and used for further analysis.



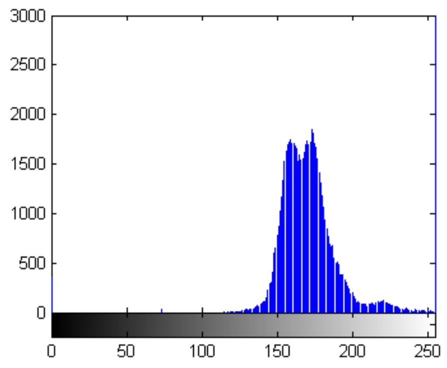
(a)



(b)



(c)



(d)

Figure 3.5: Two frames of IR video sequence with corresponding histograms. (a) Original video frame with artificial noise added and brightness level increased by 10. (b) The same video frame with the brightness level increased by 50. (c) Histogram corresponding to (a). (d) Histogram corresponding to (b).

The histogram, depicted in Figures 3.5(c) and 3.5(d), is a discrete function $h(\gamma) = n_n$ for digital image with grey levels $\gamma \in \langle 0, L-1 \rangle$, with $L = 2^8$ being the maximum intensity level for 8-bit greyscale image. The left side of the plot corresponds to the number of dark pixels within the image, whereas the right side corresponds to the brightest spots. Since both images feature similar medium grey intensity, histograms for both images will be similar. The distribution of γ is uniform and shifted to the right for the brighter image. Although changes between both images are not significant and would not cause problems to human observer in manual pedestrian detection, most of the computer vision systems based on the thresholding criterion would return vast amount of noise or in some cases even fail.

Assuming the threshold criterion th being set at a fixed level of 160 (see Figure 3.5(c)), all the grey-scale values below th would be filtered out returning detected pedestrians together with minimal amount of background data. However, if th would be applied on the image depicted in Figure 3.5(b), according to the corresponding histogram, the

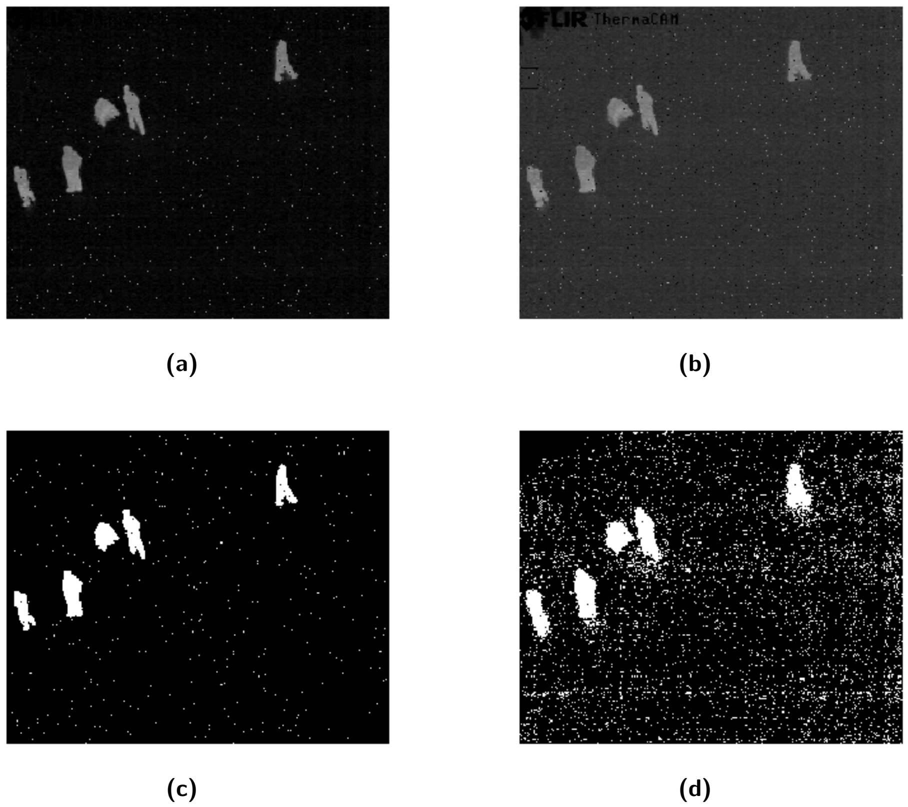


Figure 3.6: Background subtraction and intensity thresholding of images with different brightness levels. (a) Brightness level increased by 10. (b) Brightness level increased by 50. (c) Thresholding $th = 55$ applied on image a. (d) Thresholding $th = 55$ applied on image b.

output data would be a mixture of background and pedestrians, causing a failure of the detection step. Though this is an artificial example, it illustrates the potential problems of relying on a fixed threshold. Hence, support for the adaptive mechanism in order to comply also with fast environmental changes is desired.

3.3.3 Feedback Loop Control

The feedback loop control mechanism for adaptive intensity thresholding is based on the histogram of the source image being subtracted from the background model. Results of the background subtraction depicted in Figures 3.6(a) and 3.6(b) correspond to images from Figures 3.5(a) and 3.5(b) respectively. After subtraction, an intensity thresholding filter is applied, results can be seen in Figures 3.6(c) and 3.6(d). The amount of noise in Figure 3.6(d) is caused by the threshold level th being set too low. This situation could be avoided by applying higher th, according to the difference in brightness between two compared images.

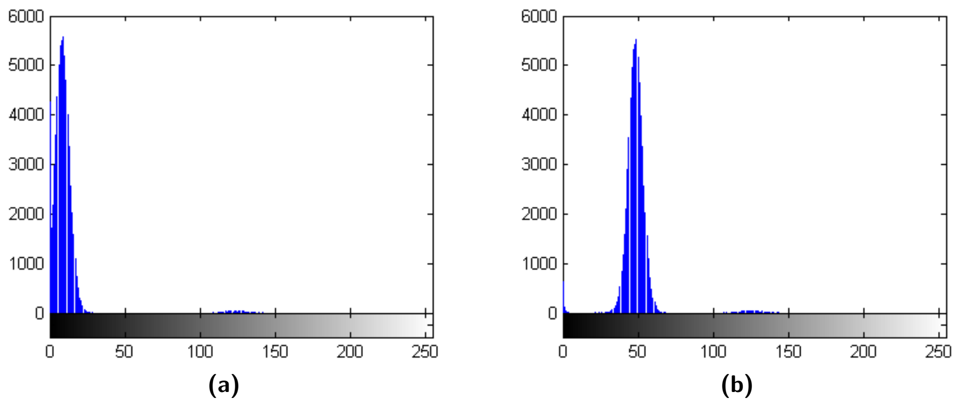


Figure 3.7: Histograms of the images after background subtraction. Left peak corresponds to background pixels, whereas data in between 100 and 150 refers to the detected pedestrians. (a) Histogram corresponding to the image from the Figure 3.6(a). (b) Histogram corresponding to the image from the Figure 3.6(b).

The principle of the adaptive thresholding developed for the purpose of this thesis is based on analysis of histograms calculated for images after subtraction. Since sudden environmental changes do not affect temperature of pedestrians (not immediately), the temperature distribution for ROIs is fixed, i.e. the position of corresponding data on histogram remains unchanged. Histograms corresponding to frames depicted in Figures 3.6(a) and 3.6(b) can be found in Figure 3.7(a) and 3.7(b) respectively. The highest peak correlates to pixels of the background, note the peak shifted to the right for the image with brightness level increased. However, for both frames the distribution of ROIs remain unchanged and can be located in between 100 and 150.

In order to achieve best results, the level of threshold must be kept as low as possible. During the initial system start-up calibration the *th* shall be set by the system operator. This value will be further adjusted by the data from the feedback loop controller in order to keep the *th* low however within the safe distance from the main peak of the histogram. It was found during test verification (model of the system running in Matlab) that Gasussian-shaped background distribution vary in position and shape, therefore the algorithm is based on both mean and variance of the distribution. The new value is calculated on the frame-basis and is added to the reference *th* value set during the calibration.

Considering the case scenario depicted in Figure 3.5, the properly calibrated system should have the threshold level kept low, at a level of 30 and 70 for Figures 3.5(a) and 3.5(b) respectively. In this basic example, the threshold level was increased by 40 units, as much as the peak was shifted as a result of sudden temperature change. Results can be seen in Figure 3.8.

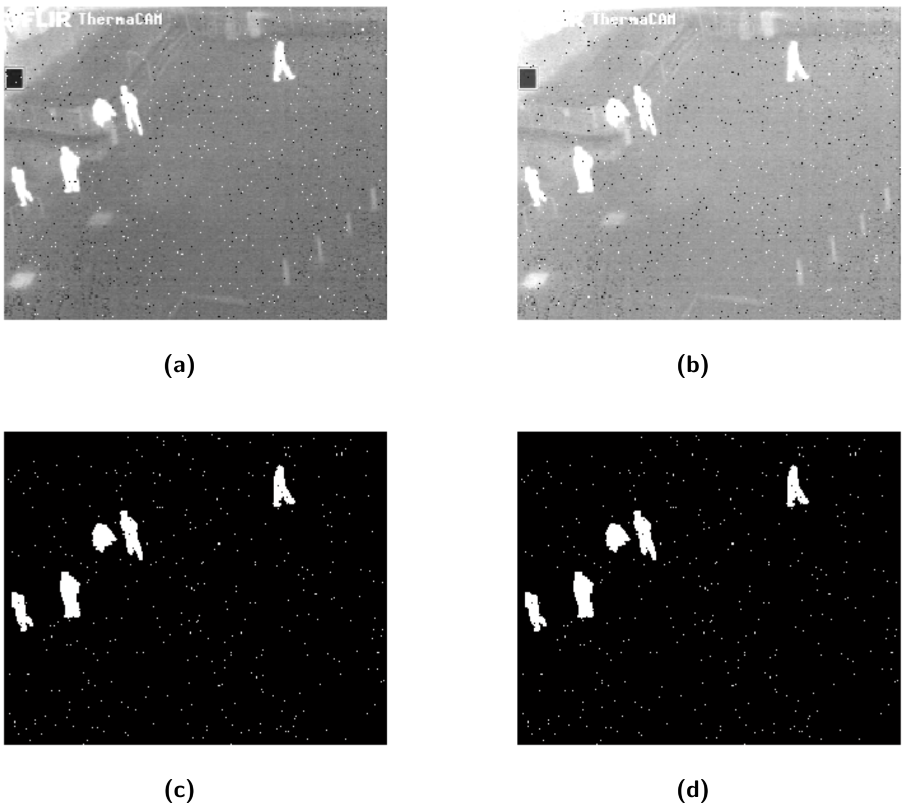


Figure 3.8: Background subtraction at a properly calibrated system for different temperature scenarios. (a) Original video frame with artificial noise added and brightness level increased by 10. (b) The same video frame with the brightness level increased by 50. (c) Background subtraction at a minimal level 30. (d) Background subtraction with the threshold level increased by the adaptive thresholding unit up to a level of 70.

3.3.4 Algorithm Pseudo Code

The algorithm introduced in previous subsections can be described with the following pseudo-code:

```
// initial calibration
$\alpha \leftarrow$ initial_value
$th \leftarrow$ initial_value

// start frame
for y in y_max
    for x in x_max
        // update histogram
        hist ← hist++

        // check input pixel
        if $|I(x,y) - M_{t-1}(x,y)| > th$
```

```
            // object detected, don't update background model
            $M(x,y) \leftarrow M_{t-1}(x,y)$
        else
            // update background model
            $M(x,y) \leftarrow (1-\alpha)\ M_{t-1}(x,y) + \alpha I(x,y)$
        end
    end loop
end loop

// update threshold level for the consecutive image scan
$th \leftarrow th + hist$
```

Listing 3.1: Labelling routine pseudo-code

3.4 Hardware Implementation

For the purpose of this thesis, a novel background segmentation algorithm was developed and implemented. Unlike other algorithms previously described in this chapter, this technique not only provides strong segmentation capabilities but is also suitable for efficient hardware implementation.

3.4.1 Architecture

As was described previously, this algorithm is composed of two major stages. During the first phase foreground elements are separated from the background, followed by update of the background model. During the second stage, extracted data is further filtered out by the adaptive thresholding module operating in the control feedback loop at a frame-rate.

Background Modelling

In order to enable background modelling, a reference background frame must be stored in the memory. As a storage for the background model Dual-Port Block RAM was used to support two independent clock domains (27MHz from video acquisition and 25MHz for video graphics adapter controller). Background subtraction is performed on the pixel basis synchronously with the 27MHz pixel clock provided by the VDEC1 video decoder board [67]. Based on the timing signals provided by ADC, input data is immediately grid aligned. On the incoming 8-bit data sample, corresponding background pixel is fetched from the memory and used for the subtraction. Subtracted data is then thresholded to determine whether the pixel shall be classified as foreground or background. If identified as an element of background, it is then processed according to

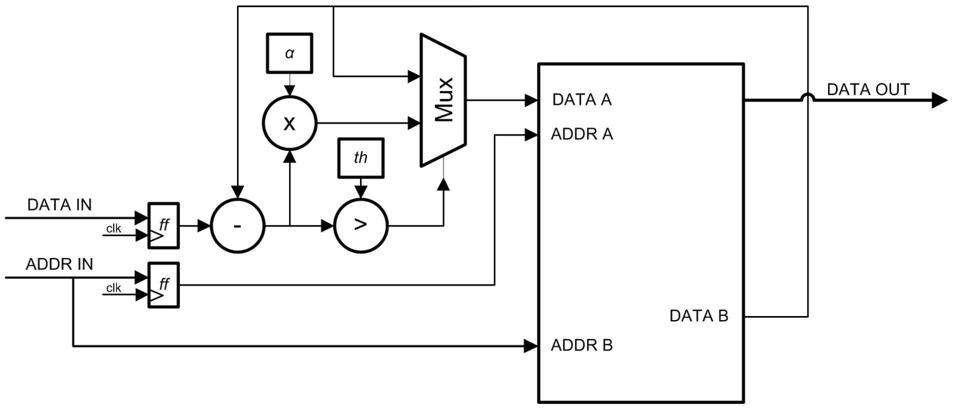


Figure 3.9: A conceptual block diagram of the background segmentation and model updating process.

the Equation (3.15). If current pixel contains elements of foreground, the background model remains unchanged (selective approach for background updating). However, if the pixel was classified as a background, the corresponding element of the background model needs to be updated according to the following:

$$M_t(x, y) = (1 - \alpha)M_{t-1}(x, y) + \alpha I_t(x, y) \tag{3.16}$$

The gain factor α is a fixed value determined during the initial calibration, typically equal or less than 0.125. The multiplication was implemented by shift registering multiplicant with the multiplier, being a multiple of 2. Such implementation avoids the need of complex floating point operations and saves essential clock cycles as well as system resources involved in hardware multiplications. The process of background segmentation and model updating can be seen on the conceptual block diagram depicted in Figure 3.9.

Adaptive Thresholding

As was mentioned previously, the adaptive thresholding technique is based on the data from histogram, calculated on the results of subtraction of the background model from the input frame. During the initial start-up calibration, a fixed value of intensity thresholding must be set. This value will be used in further processing by the adaptive approach as a reference point. Depending on environmental conditions the threshold level will be automatically increased or decreased.

An overview of the feedback loop adaptive thresholding module can be seen in simplified block diagram depicted in Figure 3.10. For the purpose of histogram implementation, a Dual-Port BRAM was used as a storage element. Port A is used by the system to address current grey-scale level, the data becomes available in the next clock cycle.

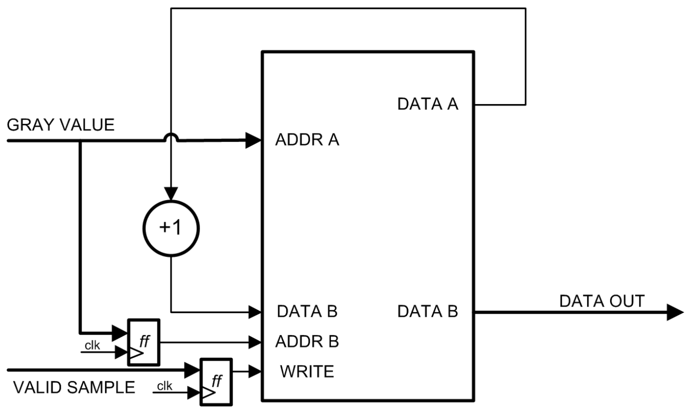


Figure 3.10: A block diagram of the feedback loop adaptive thresholding element.

It is then incremented and written back to the memory using port B, while port A is used to fetch another data sample. Thanks to the pipelined architecture, histogram is being calculated using a single block of memory, in one clock domain, with only a single clock latency. This implementation is very small and robust, performing better than other similar architectures [68, 69].

An analysis of the histogram is performed once the image scan is completed. It takes the time of a single line scan to read all the memory entries and it can be further simplified as was discussed previously in this chapter. The adaptive threshold level is calculated using not only the mean but also the variance of the Gaussian-shaped histogram distribution. It was found during empirical experimentation that the last value above the level of 512 can be effectively used for this purpose. Since the histogram was calculated on results of background subtraction, the distribution features two peaks where the first of them is always higher since it corresponds to the background pixels, whereas the other refers to ROIs. The cut-off region for histogram analysis was marked with the red thin line and can be seen in Figure 3.11. A fixed value `th_offset` (in this application 10) added to the latest memory location with the data sample saturated at 512 gives the current threshold level. The offset value `th_offset` is determined during initial system start-up calibration. It is set at a particular level based on the visual verification by the system operator. Such fixed value is later used by adaptive control mechanism. No further action is required from the system operator.

Since there is no requirement for full histogram analysis, in order to further simplify the design and reduce memory consumption, histogram samples are saturated at a level of 512. Moreover, no further values above grey-scale level of 128 are taken into consideration since they would not benefit in threshold level calculation. Hence, memory requirement can be significantly reduced, in this example from memory array of $2^{13} \times 8$ to $2^9 \times 7$, therefore by over 18 times.

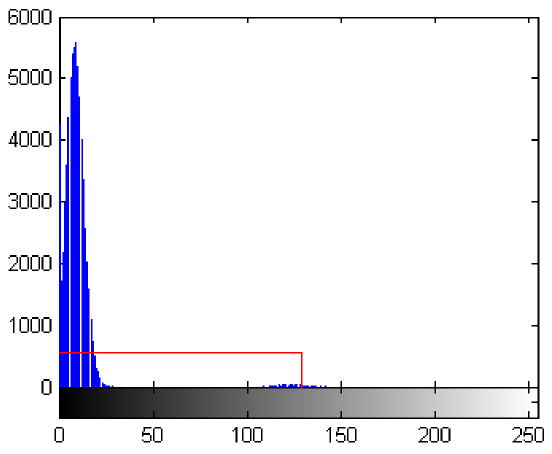


Figure 3.11: A cut-off region for simplified histogram analysis.

3.5 Conclusions

Background segmentation is of key importance in many pedestrian detection systems. The aim is to reduce the amount of processing data, providing results containing only ROIs. The efficacy of the segmentation unit has a strong impact on the quality of data further in the processing chain. Hence, a number of techniques for background segmentation was introduced and discussed in this chapter. They differ in complexity, as well as in memory requirement. Often, algorithms performing good segmentation require vast amount of memory and may not be suitable for hardware implementation. The trade-off between the quality of output data and the required processing power is a major concern.

This chapter gives a general overview of background segmentation as well as deep analysis of a number of known techniques. They can be classified into the following groups:

- intensity thresholding,
- temporal difference,
- background modelling.

There is no formal comparison test case for general evaluation. There were some attempts in the literature to unify the test procedure [32, 33, 51, 52], however, since algorithms were developed for different purposes, evaluations based on different test cases provide contradictory conclusions.

Segmentation techniques based on **intensity thresholding** provide reasonably good quality results, however this technique may be applied only to a limited number of applications - when ROIs are considerably brighter than all the background elements. They are suitable for robust and efficient hardware implementation. Thanks to low complexity and high processing speed (real-time performance), it should be taken into consideration by the system designer, possibly as part of a higher complexity algorithm.

Algorithms based on **temporal difference** provide much different segmentation output. Results are based on the subtraction of two consecutive frames, therefore contain objects currently in motion. These techniques are also efficient for hardware implementation, however they can be successfully applied only in fast and continuously changing scenarios. Since they are very sensitive to noise and unreliable with slow moving objects, they were not taken into consideration while developing the segmentation technique for IR pedestrian detection.

However, the last group of algorithms, based on **background modelling**, perform better than the ones previously mentioned, at a cost of higher complexity and increased memory requirements. These algorithms are based on the model of the background frame, gradually updated over the time. The principle of this technique is based on frame differentiation - for the incoming image pixel, a corresponding element of the background model is fetched from the memory and subtracted. Ideally, results of the subtraction contain ROIs and may be provided for further processing. However, it is not a trivial task to set-up a background model. Fixed background will cause multiple false detections due to the difference with gradual changes in the source image, on the other hand there is a risk that adaptive model will incorporate ROIs to the background too quickly. This chapter gives an overview of background modelling with analysis of most common techniques.

In the second part of the chapter, a description of the background segmentation algorithm developed for the purpose of IR pedestrian detection was provided. This algorithm is a mixture of multiple techniques commonly used in image processing applications. It is based on the selective running average background model, supported by adaptive thresholding. Such thresholding is computed based on the histogram calculated for results of background subtraction. The histogram is calculated during the run time with only a single clock latency. Hence, the threshold level is updated at the end of the image scan and can be applied for the consecutive frame. Thanks to adaptive thresholding, the system is capable of segmentation in both slow and fast changing environments. Intermediate results of the processing were depicted using a single image frame, randomly selected from IR video sequence. The same image was used for different processing stages to visualize the difference between them.

A novel background segmentation algorithm was developed and tailored for this particular application. Unlike other algorithms described in this chapter, it is suitable for efficient hardware implementation while providing strong segmentation capabilities. However, it was tested in limited scope on video sequences not exceeding 10 minutes of the continuous stream. Example video sequences used for testing as well as results of the processing can be found in [70–75].

The chapter provides implementation details on further optimization. The pipelined architecture based on Dual-Port BRAMs ensures real-time performance together with efficient management of hardware resources.

CHAPTER 4

Morphology Filtering

Mathematical Morphology (MM) is a set of theoretical methodologies for image analysis. MM describes operations that can be performed at an image pixel level in order to quantitatively describe the geometrical structure of image objects [76, 77].

This chapter gives an introduction into mathematical morphology and basic set theory definitions for the purpose of morphological filtering. Such filters are commonly used in image processing in order to remove various types of noise. Based on the duality and decomposition of erosion and dilation, an algorithm for a low-level image filtering will be presented. For the purpose of hardware implementation, a padding problem will be discussed and suitable solution will be described. The architecture of such a system will be discussed together with implementation details. The chapter will be concluded with discussion on execution time and both memory and resource utilisation.

4.1 Introduction

Mathematical morphology is extensively used for computer vision applications. Morphological techniques for pre- or post-processing, such as morphological filtering are mostly used in early stages of processing images, as low level neighbourhood operations. They are often employed within complex systems in a series of repetitive tasks such as noise removal [78] or edge enhancement [79].

There is a practical need for robotic vision applications to develop schemes which effectively suppress the noise content of an image, while simultaneously extracting features of interest required for further processing. Both erosion and dilation, which are primary operations for most of the morphological filters, they are well suited for hardware implementation. However, there is a number of ways they can be implemented; these are often not efficient in terms of resource utilization[80–82]. Thanks to the MM set theory features, the architecture for noise removal can be highly optimized for a particular application.

4.2 Mathematical Morphology

Mathematical morphology was initiated in late 1960s as a theoretical methodology for binary image analysis [76, 83]. The theory set was developed for a number of applications in the area of geological or biomedical monochrome data analysis, whereas in the late 1970s it was extended by Serra to grey-scale images [77]. The MM was applied by Rosenfeld and Kak [84] for binary pattern recognition techniques based on Boolean logic, widely used nowadays in image processing applications. Further publications describe how it was generalized to arbitrary lattices [85], also algorithms and techniques for non-linear filtering [86].

The above-mentioned set of theoretical methodology, concepts, techniques and algorithms is nowadays called morphology image processing, published in a coherent form [11, 66], widely used as key reference texts for image processing engineers. Researchers in different fields commonly use these algorithms for a wide range of image processing applications, they are investigated in terms of efficient implementation for image/video processing systems. Although most of these techniques are suitable for implementation within modern image processing platforms, some of the complex algorithms maintain the trade-off between algorithm complexity and implementation efficiency.

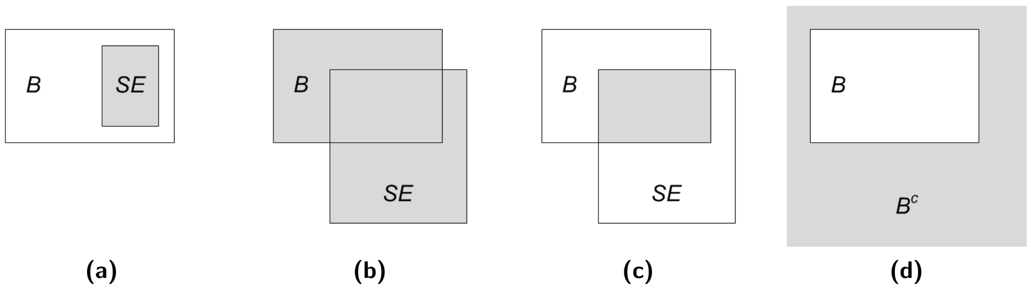


Figure 4.1: Set Theory Concepts. (a) Subset $SE \subseteq B$. (b) Union $B \cup SE$. (c) Intersection $B \cap SE$. (d) B^c complement of B.

4.2.1 Set Theory

The set theory definitions presented in this subsection are based on [11] and [66]. They will be used as a base theory in further analysis.

There are two sets taken into consideration: B and SE of 2-D integer space $\mathbb{Z}^2$, where each element is a tuple (2-D vector) described with a set of features and location coordinates (x, y). For grey-scale images represented with 3-D vectors ($\mathbb{Z}^3$ integer space), all the elements within the image pixel set can be described with additional feature that refers to the discrete intensity value. Elements of binary image sets B and SE are referred to as components $b = (b_x, b_y)$ and $se = (se_x, se_y)$ respectively, where the SE is a sliding window, referred to as Structuring Element. An expression $B = \{b \mid b = -se,$ for $se \in SE\}$ means that set B is the set of elements b, where every b is formed by multiplying both coordinates of all the elements of set SE by -1.

The set SE is a **subset** of B when all the elements of set SE are also elements of the other set, B. This is depicted in Figure 4.1(a) and can be denoted by:

$$SE \subseteq B \tag{4.1}$$

If the set C contains all the elements of both sets B and SE as depicted in Figure 4.1(b), the relationship between both sets is referred to as **union** and can be denoted by:

$$C = B \cup SE \tag{4.2}$$

However, if set C contains elements that belong to both sets B and SE (see Figure 4.1(c)), such set relationship is called **intersection**:

$$C = B \cap SE \tag{4.3}$$

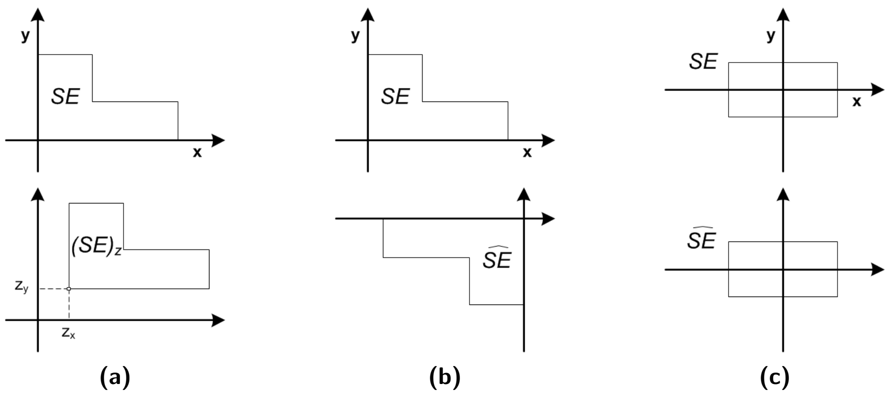


Figure 4.2: Set Operations. (a) Translation of B by z, $(B)_z$. (b) Reflection of B, $\hat{B}$. (b) Reflection invariant $B = \hat{B}$.

The **complementation** of set B, denoted $(B)^c$ and depicted in Figure 4.1(d), is a set of elements that do not belong to set B:

$$(B)^c = \{b \mid b \notin B\} \tag{4.4}$$

Translation

The **translation** of SE by point $z = (z_x, z_y)$, denoted $(SE)_z$, is an operation, where the output set $(SE)_z$ contains the same components as the set SE, which are placed in a different position on the lattice. It is depicted in Figure 4.2(a) and can be defined as:

$$(SE)_z = \{se_z \mid se_z = se + z, \forall se \in SE\} \tag{4.5}$$

Reflection

The **reflection** of set SE, denoted $\widehat{SE}$, is depicted in Figure 4.2(b) and can be defined as:

$$\widehat{SE} = \{\hat{se} \mid \hat{se} = -se, \forall se \in SE\} \tag{4.6}$$

The output set $\widehat{SE}$ is a reflection of set SE when all the components $\hat{se}$ were formed by multiplying both coordinates (se_x, se_y) of all the SE elements by -1. A special case of reflection invariant $\widehat{SE} = SE$ can be seen in Figure 4.2(c).

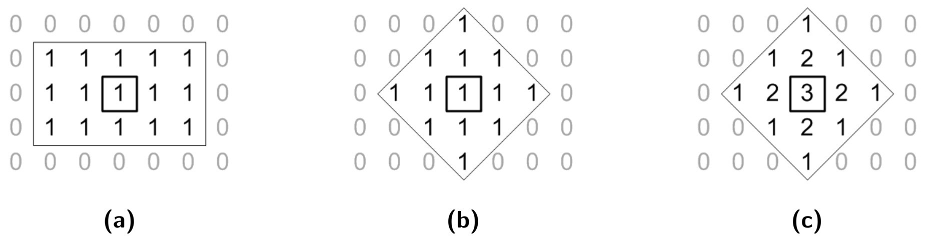


Figure 4.3: Structuring Element. (a) Flat with rectangular shape. (b) Flat with diamond shape. (c) Non - flat with diamond shape.

Minkowski Algebra

For the purpose of set operations, an equivalent to algebraic addition and subtraction was developed by Minkowski [87] to set theory and is referred to as Minkowski algebra. These operations will be extensively used in the further analysis.

The Minkowski set subtraction $\ominus$, called **erosion**, is an equivalent of the Boolean AND transformation of set B by set SE. It can be defined as follows:

$$B \ominus SE = \{b \mid b - s \in B, s \in SE\} = \bigcap_{s \in SE} B_{-s} \tag{4.7}$$

and corresponds to the intersection of sets B and SE assuming that B is translated for $\forall s \in SE$. Likewise, the Boolean OR transformation of set B by set SE is equivalent to the Minkowski's set addition $\oplus$ and denoted by **dilation**. It can be defined as follows:

$$B \oplus SE = \{b + s \mid b \in B, s \in SE\} = \bigcup_{s \in SE} B_{+s}, \tag{4.8}$$

and corresponds to the union of sets B and SE assuming that B is translated for $\forall s \in SE$.

4.2.2 Structuring Element

Image processing applications based on morphological operations operate on sets, where one of them is a set containing image pixels (components) denoted by B, the other set SE is usually smaller, called Structuring Element. Results can be obtained by probing a sliding window (kernel) SE on B. The structure of the SE depends on the application and the data representation. It varies in size, shape, structure or values.

For a typical morphological filtering on a binary image, a *flat SE* is used. Since the $B \in \mathbb{Z}^2$, such SE contains binary data $s = 0, 1, \forall s \in SE$. An example can be seen in Figure 4.3(a) and Figure 4.3(b). For the purpose of processing grey-scale or colour images, a *non-flat SE* is used, depicted in Figure 4.3(c).

Moreover, considering the choice of SE for particular task, there are also other factors that have to be taken into consideration, such as size, shape and structure. These factors have significant impact on computational complexity and memory demand and are strongly application dependent [88]. Typically, SE is small compared to the other set. It was found that for tasks such as random noise removal, regular shapes for SEs are the best choice [89]. These are lines, diamonds, circles or rectangles. Moreover, wider, rectangular-type shapes, depicted in Figure 4.3(b) can significantly improve processing efficiency since a wide range of MM techniques can be applied. Often it is a trade-off between computational complexity and size determined by the application requirement.

4.2.3 Erosion

Both **erosion** (ε) and dilation (δ) are fundamental morphological operations. In MM, many other operations are derived from them, such as opening, closing or hit-or-miss transform. Therefore, it is of crucial importance to develop efficient system architecture for them in order to perform other, more complex tasks.

For both sets B and SE of 2-D integer space $\mathbb{Z}^2$, an erosion (ε) of B by SE is denoted by Minkowski subtraction $B \ominus SE$ and can be defined as:

$$B \ominus SE = \{b | (SE)_z \subseteq B\} \tag{4.9}$$

Thus it is a process based on shifting SE by z in B. An erosion of B by SE is the set of $\forall z$ displacements of SE contained in B. In other words, for an arbitrary SE with centrally based output pixel, the result can be obtained by performing logic AND operation on pixels being located within the intersection of both sets. Therefore, referring to Equation (4.7) it can be written as:

$$\varepsilon_{SE}(B) = \bigcap_{s \in SE} B_{-s} \tag{4.10}$$

An example erosion of B 5×7 by flat rectangular SE 3×3 can be seen in Figure 4.4(a) and 4.4(b).

4.2.4 Dilation

A **dilation** (δ) of two sets B by SE in 2-D integer space $\mathbb{Z}^2$, with $\emptyset$ being an empty set, can be denoted using Minkowski's addition $B \oplus SE$ and defined as dilation of B by SE as follows:

$$B \oplus SE = \{b | (\widehat{SE})_z \cap B \neq \emptyset\}, \tag{4.11}$$

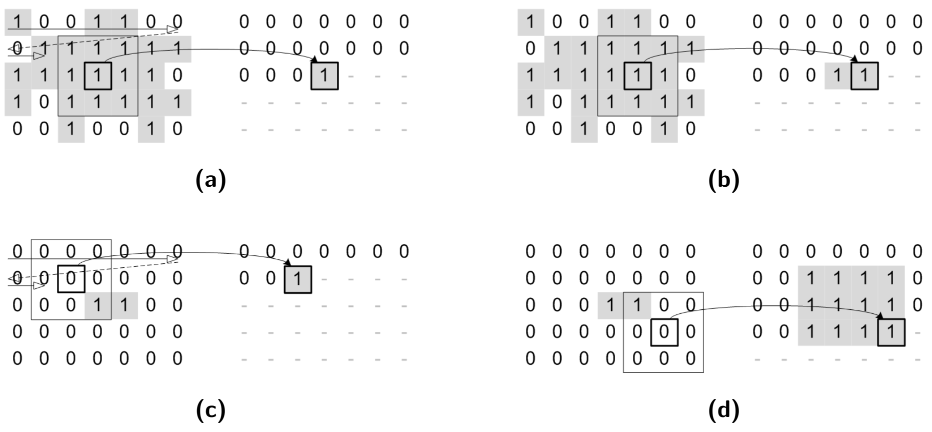


Figure 4.4: Erosion and Dilation using flat rectangular structuring element SE 3×3. (a)-(b) An example of binary erosion $\varepsilon_{SE}(B)$; the mask scans an image in raster order, an output as a logic AND of all the pixels overlapped by SE. (c)-(d) An example of binary dilation $\delta_{SE}(B)$; an output is given as logic OR of pixels under the SE.

which can be interpreted as shifting the reflection of SE by z in B. Therefore, the output is ONE for $\forall z$ displacements such that $\widehat{SE}$ and B overlap with at least one non ZERO element (union of sets). It can be interpreted as logic OR operation on compontents, therefore can be written as:

$$\delta_{SE}(B) = \bigcup_{s \in SE} B_{+s} \tag{4.12}$$

A dilation is illustrated in Figure 4.4(c) and 4.4(d).

4.2.5 Opening and Closing

As it was mentioned earlier, by combining erosion and dilation and processing them in particular order, other morphological operations can be created, e.g. opening and closing. Given the ability of erosion to shrink input images (see Figure 4.5(b)) and dilation to expand it (see Figure 4.5(c)), erosion followed by dilation is often used for smoothering contours or removing random noise. Since an erosion is performed in the first place, it also removes thin connections between connected components. This operation is called **opening**, denoted by Minkowski's opening $B \circ SE$ and can be defined as follows:

$$B \circ SE = (B \ominus SE) \oplus SE \tag{4.13}$$

Thus, the opening is a morphological operation that consists erosion followed by dilation. Results of the opening are highly related to the size and shape of SE, therefore it has

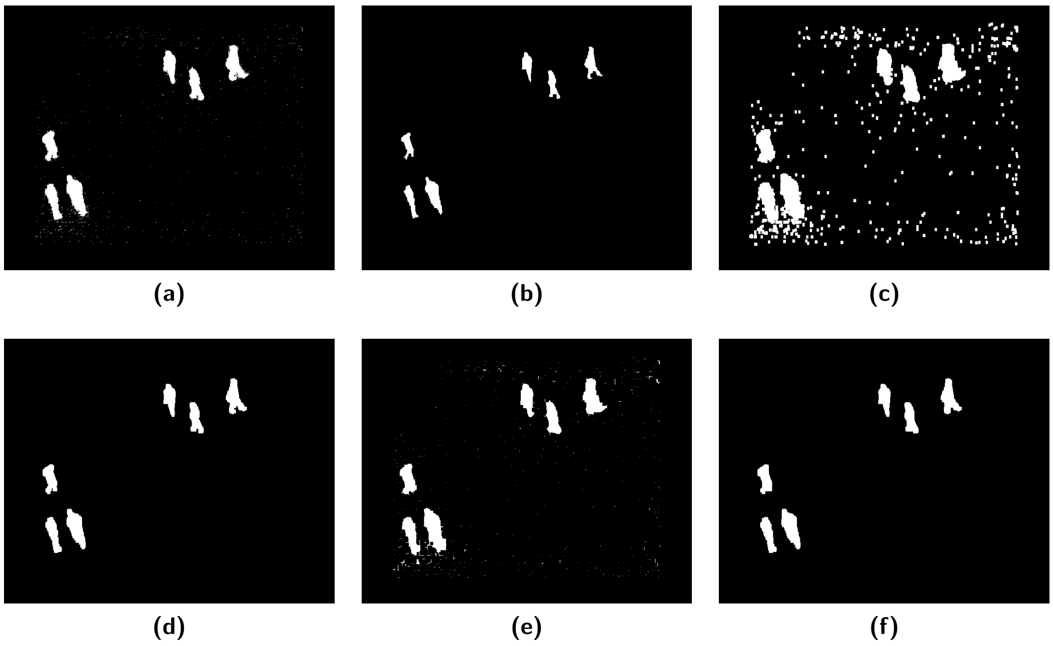


Figure 4.5: Opening and Closing. (a) An input image frame. (b) An image frame after erosion. (c) An image frame after dilation. (d) An image frame after opening. (e) An image frame after closing. (f) An image frame after opening followed by closing.

to be arbitrary defined for particular application. An example image frame before and after opening can be seen in Figures 4.5(a) and 4.5(d) respectively.

A **closing** is a morphological operation containing the same sub-operations, however here dilation is followed by erosion. Using Minkowski closing nomenclature $B \bullet SE$, it can be defined as follows:

$$B \bullet SE = (B \oplus SE) \ominus SE \tag{4.14}$$

Likewise opening, closing also tends to smooth contours, however in contrast to opening, it fuses thin connections between connected components, eliminates small gaps of holes within the contour.

Both opening and closing are illustrated in Figures 4.5(d) and 4.5(e) respectively. The difference between them becomes evident. Moreover, in order to improve filtering capabilities, systems often employ both opening and closing that are applied one after another, an example of such output frame can be seen in Figure 4.5(f). Since the opening removes random noise and thin interconnections, all the image objects become smaller, therefore morphological closing is applied to fill all the gaps and holes. In order to customize morphological filters, different size of SE can be applied. An impact of the SE size for morphological filtering will be discussed in further subsections.

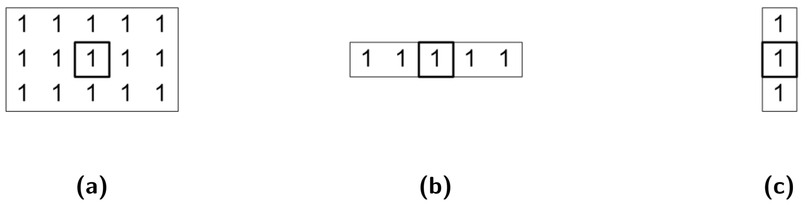


Figure 4.6: Structuring Element Decomposition. (a) Structuring Element SE 3×5. (b) Decomposed SE_1 1×5. (c). Decomposed SE_2 3×1.

4.2.6 Duality

An important MM feature of erosion and dilation is that they are dual of each other with respect to set complementation and reflection [11, 66]. This can be written as follows:

$$(B \ominus SE)^c = B^c \oplus \widehat{SE} \tag{4.15}$$

Therefore, an erosion can be derived from complemented dilation according to:

$$(B \ominus SE)^{cc} = (B^c \oplus \widehat{SE})^c \quad \Rightarrow \quad B \ominus SE = (B^c \oplus \widehat{SE})^c \tag{4.16}$$

Likewise erosion, the following equation can be derived for dilation:

$$B \oplus SE = (B^c \ominus \widehat{SE})^c \tag{4.17}$$

The duality of erosion and dilation is an important feature in MM and can be applied in designs performing morphological operations based on erosion and dilation. With such knowledge, the system architecture can be reduced to one operation since both erosion and dilation can be derived from dilation or erosion respectively.

4.2.7 Decomposition

In particular scenarios, to be discussed in further subsections, the structuring element SE can be decomposed into two smaller sets according to:

$$SE = SE_1 \oplus SE_2, \tag{4.18}$$

which means that the morphological processing based on the SE can be performed either on the whole SE or using smaller sets SE_1 and SE_2 with partial results dilated. Based on the decomposition, illustrated in Figure 4.6, a dilation defined in Equation (4.11)

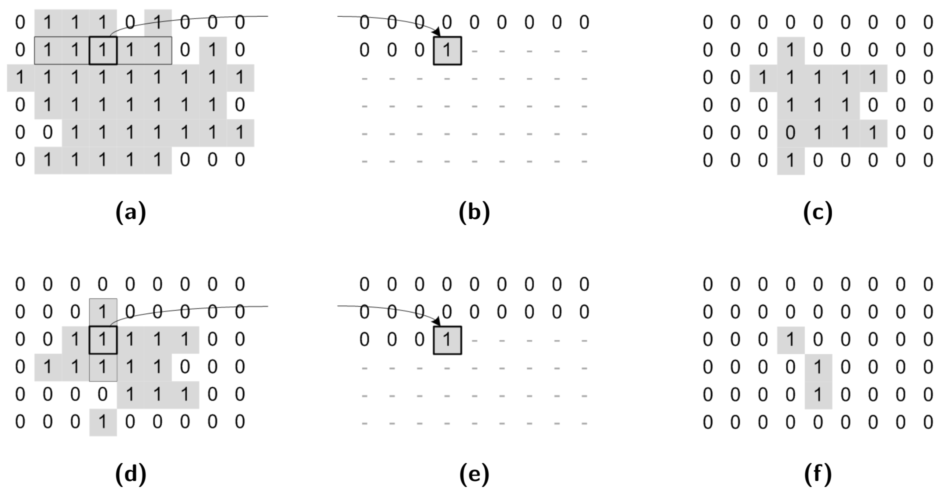


Figure 4.7: Erosion using decomposed Structuring Element SE 3×5. (a) An input binary image frame scanned with decomposed SE 1×5. (b) An output pixel of the current erosion. (c) An output image frame. (d) An output image from from the previous step is scanned with decomposed SE 3×1. (e) An output pixel of the current erosion. (f). An output image frame.

can be written as follows:

$$B \oplus SE = B \oplus (SE_1 \oplus SE_2) = (B \oplus SE_1) \oplus SE_2 \tag{4.19}$$

Thus, dilation using two smaller SEs costs less in terms of pixel comparisons compared to dilation using a single large SE. The computational saving comes from the fact that both dilations using SE_1 and SE_2 can be performed in parallel and generate results as a union of both partial outputs. Moreover, the number of comparisons is significantly reduced since the floating window is smaller. For a SE 3×5 the number is decreased from 15 to 8, however for larger SEs, e.g. 5×9, the number can be reduced to one third of initial value. An example of erosion using decomposed SE 3×5 is depicted in Figure 4.7.

Although decomposition of the structuring element clearly gives an advantage in terms of number of comparisons per pixel, there is a limited number of SEs that can be easily decomposed [90, 91]. However, for the purpose of noise removal in pedestrian detection application, a rectangular-shaped SE can be successfully used.

4.2.8 Summary

As was described in the previous subsections, mathematical morphology methodology allows the system designer to manipulate morphological operations in order to design an architecture that suits further implementation. Assuming the SE is both reflection invariant and decomposable, $SE = \widehat{SE}$ and $SE = SE_1 \oplus SE_2$ respectively, the following equations can be derived:

$$\begin{aligned} B \ominus SE = B \ominus (SE_1 \oplus SE_2) = (B^c \oplus (SE_1 \oplus SE_2))^c = \\ = ((B^c \oplus SE_1) \oplus SE_2)^c = (B^{cc} \ominus SE_1)^c \oplus SE_2)^c = \\ = ((B \ominus SE_1)^{cc} \ominus SE_2)^{cc} = (B \ominus SE_1) \ominus SE_2 \end{aligned} \tag{4.20}$$

$$\begin{aligned} B \oplus SE = (B \oplus SE_1) \oplus SE_2 = (B^c \ominus SE_1)^c \oplus SE_2 = \\ = ((B^c \ominus SE_1)^{cc} \ominus SE_2)^c = ((B^c \ominus SE_1) \ominus SE_2)^c \end{aligned} \tag{4.21}$$

Thus, by combining Equations (4.15), (4.17) and (4.18), morphological operations can be optimized in order to develop a single system architecture for both erosion and dilation, using morphological erosion only. With such architecture, the system performance can be further optimized by scaling SE according to the particular application. It shall be small and allows for processing streaming data with minimal latency and low number of comparisons.

4.3 Implementation

The objective was to develop and implement an architecture that would be flexible for both erosion and dilation. It should be small and robust, capable of processing input signal at a pixel basis simultaneously with an image acquisition in order to maintain real-time processing performance.

4.3.1 Introduction

A known issue of morphological filtering based on the sliding window is the boundary padding problem illustrated in Figure 4.8(a). This is caused by the fact the filter kernel SE with the centre point positioned in the middle of the SE need to overlap such area of an input image. There are two common approaches to overcome such a problem.

The obvious solution is to use data provided, however this will lead to trimmed output image, depicted in Figure 4.8(b). In order to fit output image within provided frame boundaries, additional padding pixels ZEROS for erosion and ONES for dilation. Although it is easy and computationally inexpensive, such a solution leads to false results and output data can not be reliably used for subsequent processing.

An alternative solution is to insert additional padding pixels outside the frame boundaries, already in the processing stage, depicted in Figure 4.8(c) as grey elements. The

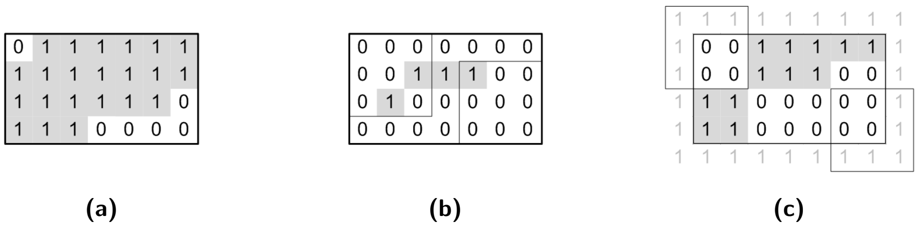


Figure 4.8: Boundary padding problem. (a) A binary image frame. (b) An erosion of a binary image frame without padding using SE 3×3. (c). An erosion of a binary image frame with padding using SE 3×3.

padded area should be filled with logic ZEROS or ONES for dilation and erosion respectively. This technique leads to a temporary resolution increase, which can be significant in case of large SE sizes. However, an implementation of such a processing unit is required in order to generate reliable output which does not affect input data. The solution for processing image frame without additional solution will be described in the following subsections.

4.3.2 Previous Work

There is a number of system implementations for image processing applications employing both erosion and dilation. Over the years their architecture improved making the parallel processing viable and efficient.

System architectures described in [80] and [81] are based on dedicated units in order to perform either erosion or dilation. An architecture described by Diamantaras and Kungin in [80] features cascaded processing element where results from one unit, once completed, are forwarded to the another through the control unit. An overview block diagram is depicted in Figure 4.9(a). In order to compute erosion or dilation, either the dilation or erosion unit remains idle, which leads to unnecessary resource utilisation and extended computing time. Moreover, the SE needs to be stored in both units, which results in additional hardware overhead.

An architecture suggested by Lucke and Chakrabarti [82] features the duality property of erosion and dilation therefore a single processing element is suggested in order to perform both tasks. Although, a single processing element clearly is an advantage for simple applications (erosion or dilation), its major drawback is the large latency when either opening or closing (erosion followed by dilation or dilation followed by erosion respectively) is requested. This comes from the fact that only one morphological operation can be performed at a time. Moreover, it leads to significant memory overhead since the intermediate results have to be stored before they can be processed by the other unit. A straightforward solution to this problem would be to double the archi-

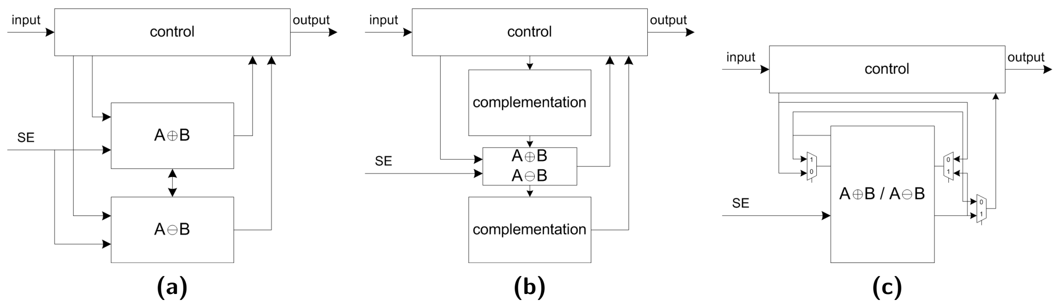


Figure 4.9: System architecture for morphological filtering applications. (a) Erosion vs dilation. (b) Erosion or dilation. (c) Erosion and dilation.

tecture and perform both operations in a pipelined fashion however this requires twice the system resources, therefore it is highly inefficient from the implementation point of view. A block diagram of the system architecture is depicted in Figure 4.9(b).

A trade-off between both systems in terms of hardware complexity and execution time was suggested by Malamas *et al.* in [92]. The system requires a single processing element for both erosion and dilation equipped with multiplexers to control the operation mode of the overall processing unit. It can be seen in Figure 4.9(c). This is achieved thanks to the duality property between erosion and dilation discussed previously. Moreover, this approach supports decomposed SE therefore is capable of processing multiple lines in parallel.

Modern implementations such as [89] share most of the features described above reducing the memory bandwidth by improving memory administration. A customized SE size for performing multiple operation in series is also supported making the design flexible for a wide variety of processing applications.

4.3.3 System Requirements

The system architecture for morphological filtering should be small and robust, capable of performing both erosion and dilation. It will be used as a processing unit for the pedestrian detection system, and since pedestrians have non-uniform shapes, it has to support flexible SE sizes.

The major requirement is that the morphological filtering unit must be capable of processing input data simultaneously with the data acquisition module. This implies processing pixel by pixel, line by line at a pixel clock rate, according to raster scan. Moreover, it is important that the size of SE could be customized since pedestrians tend to form different shapes depending on the perspective of the view, e.g. they are thin and tall looking at the 45^0 angle, whereas they are wider than taller looking from the top perspective. Hence, the width and the height of the SE have to be set accordingly.

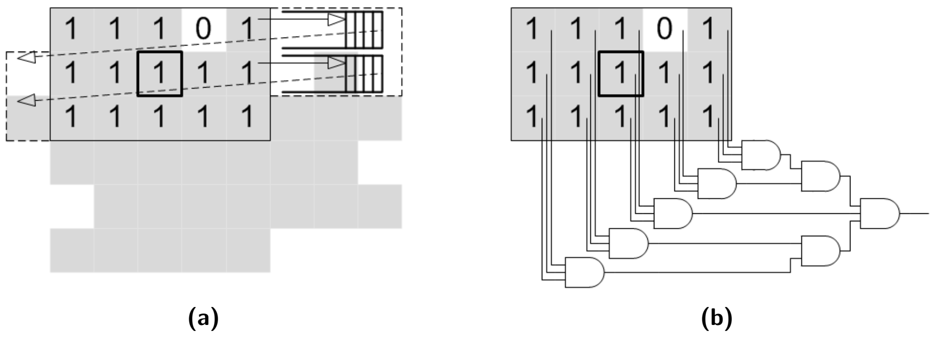


Figure 4.10: Architecture of the delayed element based erosion unit. (a) The SE is shifted in a window filter manner, where every element is fed by the FIFO used to stores an image line. (b) A low-level architecture of an erosion operation.

Also, it is required for the system operator to be able to set the size of SE for both erosion and dilation independently.

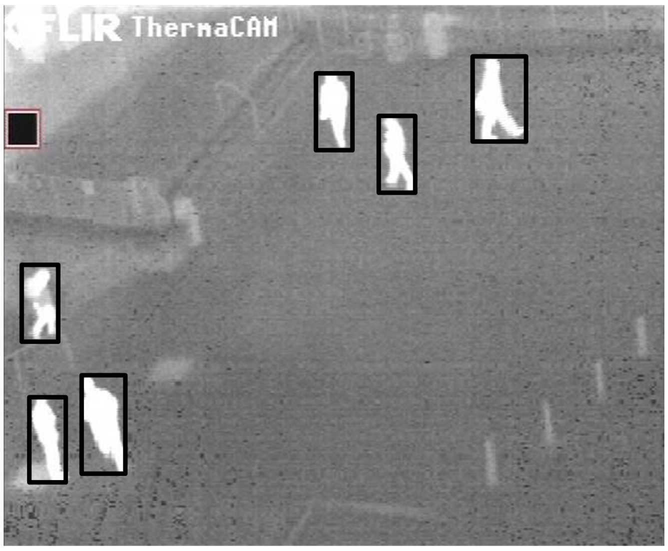
4.3.4 Architecture

In this subsection two architectures for morphological erosion and dilation will be presented. The first approach, FIR-like with delay line elements, will be described and used as a reference design for comparison purposes. The second architecture presented within the following subsections is based on a decomposed SE, features dual mode for both erosion and dilation. The memory bandwidth reduction scheme was applied by splitting each process into consecutive sub-processes. These are based on serial additions instead of parallel comparisons.

Delayed element architecture

An interesting approach for systems with variable SE sizes was suggested by Velten and Kummert in [93]. This paper gives an overview of different approaches for morphology filtering, also with arbitrary SEs. A description of a 2D FIR (Finite Impulse Response) based filtering element suitable for processing with small filter mask was provided and discussed. Such an implementation with both pixel and line delay elements is suitable for systems processing data at an acquisition system frequency, however the arithmetic operators employed within the design lead to significant increase in hardware resources. Therefore, such an approach is not recommended for implementation with larger SEs. As was suggested, the resources occupied by the processing element can be significantly reduced by replacing memory registers with boolean operators.

An implementation of erosion based on the delayed element architecture is depicted in Figures 4.10(a) and 4.10(b). The SE is sliding over the image frame in a window filter



(a)

$$\begin{array}{|ccc|}\hline 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & \boxed{1} & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ \hline\end{array} = \begin{array}{|c|}\hline 1 \\ 1 \\ \boxed{1} \\ 1 \\ 1 \\ \hline\end{array} \oplus \begin{array}{|ccc|}\hline 1 & \boxed{1} & 1 \\ \hline\end{array}$$

(b)

Figure 4.11: Rectangular filter mask (SE) for pedestrian detection applications. (a) Pedestrians with rectangular bounding boxes around them. (b) Decomposed rectangular flat reflection invariant SE.

manner according to the raster scan. An input image data is stored within the SE using flip-flops, whereas delayed line data is stored in FIFOs. The size of FIFO is determined by the binary image frame width defined by B_{col} parameter. The number of storage elements is determined by SE_{row}, which specifies the height of the SE. Once all the elements of the SE are filled with the image pixel data, all the registers are concurrently accessed and the AND boolean operation is performed on line vectors, followed by the global AND giving the output of an erosion for the current pixel location.

Such an implementation is capable of high speed processing, at a pixel clock frequency, therefore it can be used within streaming data systems. However, it is limited to the fixed size SEs, where a single change in architecture implies an increase in hardware complexity (complex management of the data storage) and requires relatively large amount of system resources to instantiate multiple FIFOs.

An analysis of the execution time together with the memory requirement will be provided in the following subsections for comparison purposes.

Decomposed dual architecture

Assuming a decomposable, flat and reflection invariant SE, a dual architecture for both erosion and dilation can be suggested. Such an approach for morphology filtering is superior in terms of memory savings and system complexity compared to the traditional delayed element architecture [94].

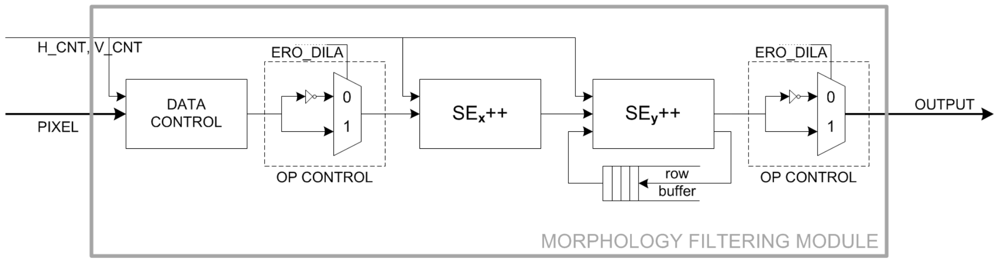


Figure 4.12: Block diagram of the decomposed dual system architecture

For the purpose of the pedestrian detection, a rectangular SE is used. It is a common filter mask for noise removal, perfectly suited for such applications (pedestrian silhouettes form rectangular shapes, see Figure 4.11). Moreover, flat rectangular shapes are reflection invariant and easily decomposable. Hence, both erosion and dilation can take advantage of the structuring element decomposition, derived from Equation (4.18) and Equation (4.19). In comparison to the classical approach based on delayed element architecture, the number of operations per pixel can be significantly reduced (bigger savings for larger SEs). Other advantages of SE decomposition are further discussed in subsection 4.2.7.

The class of flat rectangular objects is also reflection invariant, therefore dilation can be derived from erosion (see Equation (4.15) and (4.16)) and vice versa, according to Equation (4.17). This leads to the main conclusion: since the flat rectangular SE is used, a system architecture for binary erosion can be also used to perform binary dilation. It can be achieved by combining Equations (4.15), (4.17) and (4.18), which leads to the following:

$$\begin{aligned} B \oplus SE = (B \oplus SE_1) \oplus SE_2 = (B^c \ominus SE_1)^c \oplus SE_2 = \\ = ((B^c \ominus SE_1)^{cc} \ominus SE_2)^c = ((B^c \ominus SE_1) \ominus SE_2)^c \end{aligned} \tag{4.22}$$

Therefore, from the implementation point of view, a dilation can be performed as an erosion when both input and output data are inverted. Such implementation is small and allows for processing streaming data with minimal latency and low number of operations per pixel.

A conceptual block diagram of the decomposed dual system architecture is depicted in Figure 4.12. It contains the following elements:

- `DATA CONTROL`
  This unit determines input data based on the `H_CNT` and `V_CNT` synchronization signals in order to manage horizontal and vertical padding. For `(H_CNT` $<$ `H_MAX)`

and (V_CNT < V_MAX), the output value is specified by the input PIXEL signal, however once the H_CNT or V_CNT reach H_MAX or V_MAX respectively, the output data is generated by the control unit. It is a constant value ONE for $\lfloor$SE_X/2$\rfloor$ consecutive clock cycles or $\lfloor$SE_Y/2$\rfloor$ rows respectively. This gives sufficient amount of time to process padding area.

- OP CONTROL
  The OP CONTROL unit specifies the operation mode for the morphology filtering module. If the ERO_DILA signal is set high, the module performs an erosion, otherwise dilation.

- SE_X++
  This unit refers to the arithmetic element for the horizontally decomposed *SE*. Here, the number of consecutive ONES is incremented until it reaches value SE_X - 1, then the module proceeds with the SE_Y++ unit. The data is stored in temporary register using flip-flops. Once the input pixel ZERO is received, the register is cleared immediately and the counter starts again. In order to manage processing pixels close to the image left boundary, the padded area contains pre-calculated value SE_X - 1 which is assigned to the register. This can be described with the pseudo-code listed below:

```
if pixel = 1
    if pix_cnt < SE_X − 1
        pix_cnt ++
    else
        GOTO: SE_Y++
    end
else
    pix_cnt ← 0
end
```

Listing 4.1: SE_X++ control unit data flow pseudo-code

- SE_Y++
  This unit refers to the vertical element of the decomposed *SE*. Once the SE_X++ unit reaches the threshold value, the SE_Y++ unit increments the number of consecutive lines where the SE_X value was reached. Then, it is stored in the ROW BUFFER storage element, which is constantly accessed for the read operation. The unit produces ONE as OUTPUT only if the current pixel is ONE, the number of consecutive pixels ONE is equal to SE_X - 1 and the value stored in the ROW BUFFER is equal to SE_Y - 1. In such case the value stored in the ROW BUFFER remains

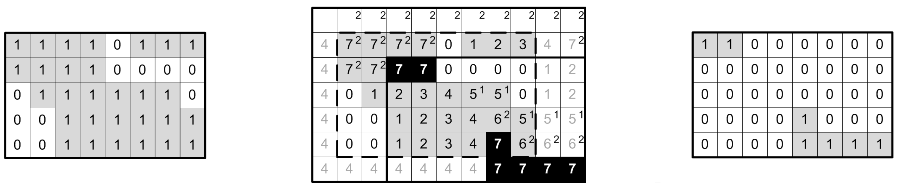
unchanged. However, if the number of pixels within the current row up to this point is lower than SE_X - 1, the vertical counter is cleared and assigned with 0. The operation data flow can be described with the following pseudo-code:

```
if pix_cnt = SE_X − 1
    if row_cnt < SE_Y − 1
        row_cnt++
    else
        OUTPUT ← 1
    end
else
    row_cnt ← 0
end
```

Listing 4.2: SE_Y++ control unit data flow pseudo-code

With such an architecture, the number of pixel accesses is significantly reduced from SE_Y $\times$ SE_X in case of delayed element architecture, to only SE_Y - 1. The boundary problem is handled with padded area controlled by the horizontal and vertical counters. Whenever the current pixel is aligned in the first row or in the first column, it is assigned with one of the two pre-calculated values SE_Y - 1 or SE_X - 1 for row_cnt or pix_cnt respectively. On the other hand, once the border pixel for either B_{col} or B_{row} is reached, further data for the horizontal or vertical padded area is provided by the DATA CONTROL unit. In such situation, a constant value ONE is generated for the consecutive ⌊SE_X/2⌋ clock cycles or ⌊SE_Y/2⌋ lines respectively. This signal is constant for both erosion and dilation. Therefore, the system designer when instantiating this module is not obliged to interface an auxiliary serial data generator to cover padded areas. Moreover, in video processing applications, the processing task to cover padded areas can be handled during the horizontal or vertical blanking periods, therefore there is no need for auxiliary temporary storage to handle input data, the system is stall free.

An erosion data flow using 3×5 SE is depicted in Figure 4.13, where the intermediate step with current register and memory values is illustrated in Figure 4.13(b). Since SE_Y = 3 and SE_X = 5, the top padded and the left padded areas are assigned with values 2 and 4 respectively. Once the B_{col} is reached, the horizontal padding takes another 2 consecutive clock cycles (⌊SE_X/2⌋), whereas vertical processing time takes additional one line scan (⌊SE_Y/2⌋). As can be seen, the output data is generated with the latency proportional to the SE size. The output data alignment is supported by the H_CNT_OUT and the V_CNT_OUT signals, where H_CNT_OUT = H_CNT - ⌊SE_X/2⌋ and V_CNT_OUT = V_CNT - ⌊SE_Y/2⌋. Values in grey refer to the padded data, pix_cnt and



(a) **(b)** **(c)**

Figure 4.13: An erosion data flow for decomposed dual system architecture using 3 × 5 SE. (a) An example binary input data. (b) An intermediate step, where values in grey refer to padded data, `pix_cnt` and `row_cnt` values are displayed in grey cells, whereas black cells refer to the ones that the output signal is generated for. The content of `ROW BUFFER` can be found in the top corner of each cell, otherwise it is 0. (c) An output image frame.

`row_cnt` values are displayed in grey cells, whereas black cells are the ones that the `OUTPUT` = 1 signal is generated for.

The processing of input image starts simultaneously with the first input pixel provided. If the first image pixel (`H_CNT` = 1 and `V_CNT` = 1) is ONE, the `pix_cnt` register is assigned with `SE_X - 1` and the system proceeds with the `SE_Y++` unit. Here, the `row_cnt` register is assigned with `SE_Y` - 1, this value is also copied to the `ROW BUFFER`. In such situation, according to the Listing 4.2, the `OUTPUT` signal should be assigned with 1, however this will be postponed until both horizontal and vertical output counters `H_CNT_OUT` and `V_CNT_OUT` reach threshold values ⌈`SE_X`/2⌉ and ⌈`SE_Y`/2⌉, respectively.

4.3.5 Module Instantiation

The morphology filtering module based on the decomposed dual architecture was developed and implemented as a fully customizable generic HDL module. Thanks to the modular architecture it can be easily adopted by other object detection or pattern recognition systems. The module is semi-device independent thanks to the behavioural HDL inference templates used for memory controllers. Therefore, it can be used as a hardware accelerator for a wide variety of FPGA-based processing systems requiring acquisition noise removal.

The block diagram of the morphology filtering module is depicted in Figure 4.14. The top part of the block is occupied by generic parameters specifying width and height of the binary image and the structuring element *SE*. Signals depicted on the left side refer to input signals as follows: binary pixel data, operation mode control signal for erosion (1) and dilation (0), horizontal and vertical counters giving the coordinates of the current location, reset and clock. To simplify the overall system architecture, the morphology filtering module can operate synchronously with the input data controller therefore the clock signal operates on the video source pixel clock frequency.

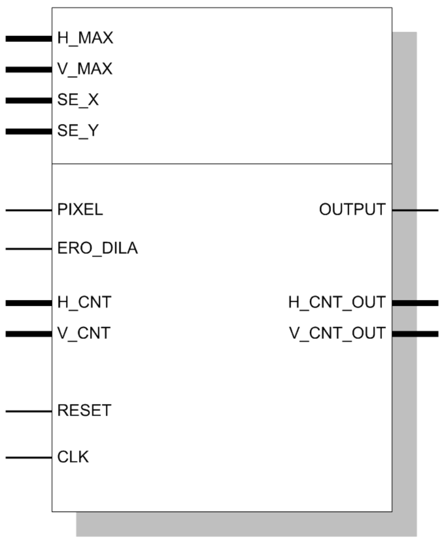


Figure 4.14: Morphology filtering module block

There are three output signals supported by the morphology filtering module. The `OUTPUT` signal is set high if all the pixels within the SE are ONEs or at least one of the pixels is ONE, for erosion and dilation respectively. Two other signals refer to the SE kernel coordinates that the `OUTPUT` signal is generated for.

4.3.6 Results and Performance

The morphology filtering module based on the decomposed dual architecture was developed and tested using Matlab computing environment [95]. This was followed by VHDL implementation for hardware design using Xilinx ISE 9.2 (Integrated Software Environment) WebPACK Design Software [96]. The system was verified and tested using Mentor Graphics ModelSim XE III 6.3c simulator [97]. It was synthesized and routed for both Spartan-3A [25] and Virtex-II Pro [26] Xilinx FPGA devices. The system testing involved also visual verification using both static images stored in the embedded ROM memory block, also real-time video signals provided by two infrared cameras: Thermo-Vision Micron A10 Infrared Camera [98] and FLIR Systems Thermacam PM595 [5]. For comparison purposes both approaches described in this chapter were implemented and synthesized.

While testing on video signals at a resolution of 320×240 pixels, it was found that most of the acquisition noise can be filtered out from the data source with the SEs as small as 5×3 or 7×5 pixels, whereas for VGA resolution (640×480 pixels) 9×5 or 13×7 pixels accordingly. In order to achieve satisfying results (acquisition noise removed whereas objects remain undistorted), the noise removal unit was implemented to support morphological opening (erosion followed by dilation). According to the literature [11],

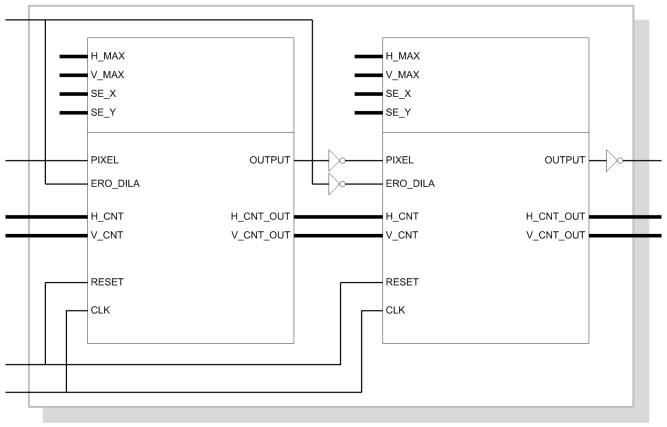


Figure 4.15: Block diagram of the implemented system architecture for morphology opening (erosion followed by dilation).

an opening followed by closing gives even better filtering results, however they are not worth increase in system complexity. Synthesis reports for implementation for all mentioned SE sizes will be provided and discussed in the following subsection.

The block diagram of the architecture for implemented morphology opening is depicted in Figure 4.15. The system is pipelined therefore a single instantiation of the morphological unit is sufficient. Such an architecture features `SE_X - 1` pixel clock cycles with `SE_Y - 1` rows latency, therefore, for most of the sensor based video processing systems it supports real-time processing performance. This can be guaranteed with the horizontal and vertical blanking pulses, whose provide sufficient amount of processing time to accomplish noise removal in a single image scan, even for large SEs.

Memory Requirement

The motivation for this particular module development was to implement a processing unit that would be small and robust, processing data with low latency. Since the noise removal module is integrated within the system as a pre- as well as post-processing step for other blocks, the timing and associated latency are critical for the overall system performance. Moreover, although the prototype is implemented in HDL for FPGA development platform synthesis, the area of the synthesized block needs to be small if further ASIC design is taken into consideration. In order to ensure such design requirements, memory resources must be reduced to the minimum since they are known for being the largest element of the silicon chip in terms of area occupancy.

Here, for development purposes, the module was implemented using dual-port RAM blocks. There are two reasons for such implementation: simultaneous memory read and write is required in order to process data according to the algorithm described in the previous subsections, also, dual-port BRAM memory is already available within the chip therefore it can be used with no additional cost. Since the addressing scheme is straightforward, for the purpose of further implementation on ASIC, simultaneous reads and writes can be achieved by doubling the width of the data bus [89] or by increasing the clock frequency.

Delayed element architecture

The memory requirement for the module implementation based on the delayed element architecture described in Subsection 4.3.4 is highly proportional to the image size and the size of SE. For efficient implementation, image pixels within the SE should be stored in flip-flops, whereas delayed elements within previous lines should be stored in FIFOs implemented using BRAM memory. The number of data bits required can be calculated according to:

$$mem_{DE\ total} = (SE_{row} - 1) \cdot (B_{col} - SE_{col} + 1) + SE_{row} \cdot (SE_{col} - 1)\ bits, \quad (4.23)$$

where the first part of the equation refers to size of the FIFO and is proportional to the image width and the height of the SE, the second part specifies the size of data to be stored locally in registers. The equation listed above does not include memory required by the system to handle padded areas. Therefore, an additional $\frac{SE_{col}}{2} \cdot SE_{col}$ data bits for every side of the image frame must be included into the final calculus.

Decomposed dual architecture

Decomposed dual architecture features advanced memory management scheme thanks to the use of arithmetic operators. Such implementation requires less memory than corresponding implementation based on the delayed element architecture. It is proportional to the size of the SE and the image width. It is determined by the sum of word-length required to store $SE_{col} - 1$ and $SE_{row} - 1$ multiplied by the image width increased by half of SE_{col}. It can be written as follows:

$$mem_{DD\ total} = \lceil log_2(SE_{col} - 1)\rceil + \lceil log_2(SE_{row} - 1)\rceil \cdot (B_{col} + \lfloor \frac{SE_{col} - 1}{2} \rfloor)\ bits \quad (4.24)$$

In order to handle padding area, an additional storage is required hence $\lfloor \frac{SE_{col}-1}{2} \rfloor$ in the equation above.

The memory requirement for both architectures are compared and depicted in Figure 4.16. Figure 4.16a illustrates memory requirement as a function of the SE size increasing in the image at 640 × 480 pixels, whereas Figure 4.16b gives an overview of the difference in memory requirements for the processing unit operating on different image sizes with the SE size proportional to the resolution of the image frame.

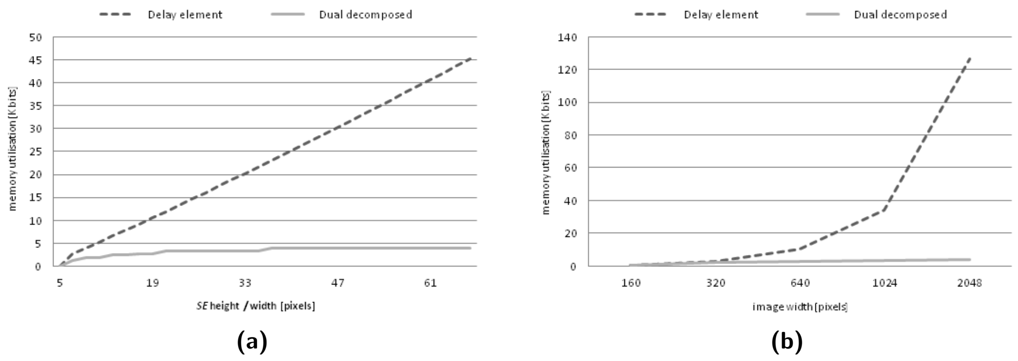


Figure 4.16: Memory requirement for both delayed element and dual decomposed system architectures. (a) This plot shows function of the SE at 640 pixels width image frame. (b) This plot shows difference between both architectures for different size image frames.

As can be seen, the difference in memory consumption between these two architectures is significant. It increases with growing size of SE, whereas for larger image the difference is even bigger. Assuming video stream at a resolution of 640×480 with the SE size 13×7, number of bits to be stored is 7.8 Kbits and 2.6 Kbits for delayed element and dual decomposed architectures respectively, which gives over 300% saving in terms of memory requirement already for such as small SE. When considering implementation for higher resolution image frame, dual decomposed architecture is preferred.

Execution Time

In this subsection the computational complexity for both architectures will be compared. It will be measured as a number of operations on input pixels per each output data sample generated. The execution time will be also provided. This will be calculated in clock cycles required to generate $B_{row} \times B_{col}$ pixels from the time when the first pixel is provided until the last one is generated.

The computational complexity may not affect the time required to process input data since some of the operations can be processed in parallel. This factor is taken into consideration for ASIC since it has impact on power dissipation, which can further affect increased heat generated by the circuit.

Computational complexity

The delayed element architecture is straight-forward and easy to analyse. Depending on the SE size, every input pixel is accessed B_{col} times within each line scan and this is repeated for the B_{row} lines. Therefore it is directly proportional to the SE size and can be expressed as a product of B_{row} and B_{col}. The number of memory accesses per pixel is defined by SE_{row} for each operation due to implemented data shifting when processed horizontally.

The architecture based on the dual decomposed processing is more complex however it features lower computational complexity than predecessor. It requires only a single sum and one comparison for each arithmetic block therefore just 4 operations in total are required. In contrast to the delayed element architecture, this number stays constant for all the SE sizes.

Execution time
In contrast to computational complexity, the execution time has direct impact on the overall system performance. It specifies the time required to generate output data for the whole image frame including padded area therefore can be specified as follows:

$$time_{total} = time_{frame} + time_{padding} \tag{4.25}$$

Assuming processing unit operates on the pixel data clock, the time required to process image frame is equal for both architectures. This can be calculated according to:

$$time_{frame} = B_{row} \cdot B_{col} \cdot \frac{1}{f_{pixel}} \; [seconds] \tag{4.26}$$

whereas the time required to process padding area differs for both architectures. Due to the parallel processing nature of the SE elements, the delayed element architecture is required to scan all the pixels according to raster scan in order to fill the SE mask. This puts a requirement on the scanning step to feed the system with data to process top and left padding areas, whereas for dual decomposed architecture this is already done by the control unit. The time required to process a single side horizontal and vertical padded area can be calculated according to:

$$time_{padding} = \lfloor \frac{B_{row}}{2} - 1 \rfloor \cdot \lfloor \frac{B_{col}}{2} - 1 \rfloor \cdot \frac{1}{f_{pixel}} \; [seconds] \tag{4.27}$$

The time specified by Equation (4.27) must be doubled for delayed element architecture therefore it constitutes the difference between those two processing techniques.

Comparing both architectures, the dual decomposed architecture gives better results in both aspects. Although it features both lower computational complexity and shorter execution time, these differences are marginal in terms of processing the whole image frame. Since the processing element is not involved in the multiple pipeline chain, this latency can be omitted.

Table 4.1: Resource utilisation for implementation of the morphology erosion on the XC2VP30 FPGA with 5×3 SE at 320×240 image size

Resource	in use	total
Slice Flip Flops	14	27392
4 input LUTs	126	27392
Occupied Slices	64	13696
Block RAMs	1	136

Resource Utilisation

Considering lower memory consumption, lower complexity and shorter processing time, the dual decomposed architecture fits better system requirements therefore it was chosen for implementation. It was implemented in VHDL and synthesised for Virtex-II Pro [26] Xilinx FPGA. For the purpose of noise removal for infrared pedestrian detection, a 5×3 rectangular filter mask was chosen for the image at a resolution of 320×240 pixels. Results of the synthesis can be found in the Table 4.1.

As can be seen, implementation of morphology erosion is small and does not require significant amount of system resources. Thanks to generic instantiation and semi-device independent HDL inference templates used for memory controllers, it is scalable and can be can applied within other computer vision system.

4.4 Conclusions

In this chapter, a new implementation approach for morphology filtering was described. It is based on mathematical morphology and basic set theory definitions, also introduced. The architecture presented in this thesis introduces a novel approach for handling padded areas, allowing data flow simultaneously with video acquisition.

This particular implementation was intended for removing various types of noise when detecting pedestrians from infrared video streams. Hence, a specific flat and decomposable scan mask was used to improve its efficacy and make the use of MM set theory features. Therefore, it takes an advantage of the duality and decomposition of erosion and dilation. Moreover, a discussion and solution were also provided for padding problem to cover image pixels within the border area of the image frame.

The objective was to develop and implement an architecture that would be flexible for both erosion and dilation. It should be small and robust, capable of processing input signal at a pixel basis simultaneously with an image acquisition in order to maintain real-time processing performance. Moreover, the system had to support flexible SE sizes since the pedestrians tend to form non-uniform shapes.

The implementation is based on MM set theory concepts such as subset, union, intersection, translation and reflection together with an equivalent of algebraic subtraction and addition referred to as Minkowski algebra. Based on these morphological operations, an optimized version of noise removal algorithm based on morphology erosion and dilation was developed for hardware implementation. Such implementation takes an advantage of the fact that erosion and dilation are dual of each other with respect to set complementation and reflection [11, 66].

As a reference point, an implementation based on delay-element (classical) architecture was described, also used for comparison purposes. Such implementation, based on sliding filter, is capable of high speed processing, at a pixel clock frequency, hence it suits streaming data systems. However, due to non-generic architecture, it is limited to fixed size SEs.

In comparison, assuming flat and reflection invariant SE, a dual architecture for both erosion and dilation was suggested. This approach for morphology filtering is superior in terms of memory utilization and system complexity. It gives a significant savings in terms of number of operations per image pixel, which could be beneficial when processing higher resolution images with larger SEs. The module was implemented to support both erosion and dilation in a single architecture, by using logic inverters on input and output of the module. Moreover, the support for padding reduces system complexity since there is no need for auxiliary serial data generators to be instantiated within the data flow.

The module was developed and tested in software using Matlab, this was followed by successful implementation in VHDL for Virtex-II Pro Xilinx FPGA, as a generic HDL module. Thanks to the modular architecture it can be adopted by other object detection or pattern recognition systems. It is semi-device independent thanks to the behavioural HDL inference templates used for memory controllers. Therefore, it can be used as a hardware accelerator for a wide variety of FPGA based processing systems requiring noise removal.

In terms of memory utilization, the dual-decomposed architecture features better compared to the classical approach for noise removal. It gives even greater margin when processing on higher resolution images, however it is already 300% more efficient when operating on VGA image frame with SE size 13×7. Such data was calculated based on analysis of memory requirements derived for both architectures, using Equations (4.24) and (4.23).

The dual decomposed architecture features also significant savings when considering number of pixel operations per single output sample. This is a result of processing based on decomposed scan mask. In terms of execution time they perform similar, with marginal advantage when processing padded areas. This implementation features novel model of memory architecture developed for the purpose of real-time processing within streaming data system.

In summary, an architecture for morphological noise removal is small in terms of resources utilization, capable of real-time processing, features low memory requirement and is suitable for further ASIC implementation. It is suitable for a variety of video processing applications with flexible size flat rectangular SE.

CHAPTER 5

Connected Component Labelling

Connected Component Labelling (CCL) is a fundamental feature of many computer vision systems. It allows the assignment of unique identifiers (labels) to different, disjoint groups of pixels (connected components). Once it is completed, a variety of features, such as position, size, width, height, etc., can be extracted for every connected component. These features can be further used for other processing, such as classification or tracking. Hence, the process of labelling constitutes a significant stage in automated surveillance or pattern recognition systems.

A typical real-time video processing embedded system involves the following stages: firstly, the video source is processed by one of the background separating algorithms. In this step the foreground objects, often referred to as Regions of Interest (ROIs), are subtracted from the background model. Results of this operation are forwarded to the filtering and thresholding units where the acquisition noise is removed. By applying a thresholding filter to the processed data, the image is binarized, therefore the amount of information is significantly reduced. From now on, groups of connected pixels refer to the detected objects, however to allow classification or tracking, the data has to be further processed by one of the mid-level vision grouping techniques.

CCL algorithms analyse binary images in order to distinguish disjoint groups of pixels (here objects) and assign them with individual labels. The use of grouping techniques changes the type of units being processed. Before the transformation the image units were pixels, whereas after the transformation they are called regions or segments and are composed of groupings of pixels. Once all the labels are distinguished, a variety of property measurements can be made on them, this step is called feature extraction. The data features such as position, size, width, height or Centre of Gravity (CoG) are calculated for each object and can be used for statistical pattern recognition or tracking analysis. The operation sequence beginning after the acquisition noise removal until the region property measurement and statistical pattern recognition is called connected component analysis [10].

This chapter gives an introduction into CCL techniques. An insightful analysis of the memory requirement and the processing time for chosen algorithms will be provided. In the remainder of this chapter, a detailed description of the single pass CCL algorithm will be provided together with a deep analysis of the architecture for hardware implementation. This will be concluded with experimental results and discussion on further improvements.

5.1 Introduction

Connected component labelling is an operation where groups of connected pixels (connected components) are classified as disjoint objects with unique identifiers (labels). This operation can be described as assigning a unique label l taken from a set of integral values $L \subset \mathbb{N}$, to each connected component. Thereby, an input binary image frame $B \in \mathbb{Z}^2$, where all the pixels $p \in B$ correspond to the background or to the foreground objects ($F_b = 0$ or $F_f = 1$ respectively), is transformed into a frame, where each pixel is represented by a decimal value (label) being an identifier of the connected component CC_k it belongs to. Here, $1 \leq k \leq K$ and K defines the total number of connected components within the frame. Labelling of B can be written as $g : B \mapsto \mathbb{N}$, where $g(x, y)$ is described as:

$$g(x,y) = \begin{cases} F_b & \text{if } B(x,y) = F_b, \\ l_k & \text{if } p(x,y) \in CC_k. \end{cases} \tag{5.1}$$

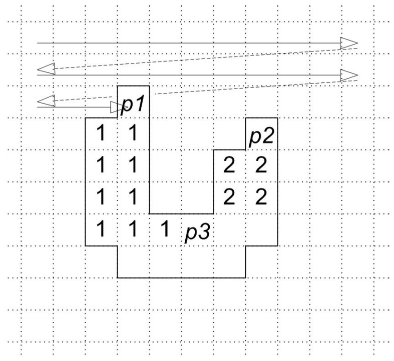


Figure 5.1: A typical label collision

5.1.1 Labelling

The process of labelling is illustrated in Figure 5.1. According to Rosenfeld and Pfaltz [99], who described the classical approach for CCL, the following stages can be distinguished:

- assign a label 0 if the pixel forms part of the background;
- if only the current pixel was found as a foreground element, assign a new label;
- if only one of the neighbouring (adjacent) pixels was already labelled, assign its label to the current pixel;
- if two or more of the neighbouring pixels were labelled with different identifiers, assign the lower label to the current pixel and store both of them in the equivalence table for further merging;

5.1.2 Label Collision

A common problem many of the CCL algorithms struggle with is label collision, caused by the "U" shaped objects. This is when during the progressive scanning two of the neighbouring pixels to the current pixel location are already labelled with different identifiers. A typical label collision is depicted in Figure 5.1.

Since the image is scanned in raster scan order, the first foreground pixel encountered will be the pixel denoted as $p1$. According to the classical approach for CCL [99], this pixel will be assigned with a new label. During the successive line scan, assuming the 8-neighbourhood connectivity, pixels directly below the pixel $p1$ will be assigned with the same label, however the pixel $p2$, despite the fact it belongs to the same object, will be assigned with a different label.

The reason for pixel $p2$ being labelled with different identifier is caused by the fact there is no momentary information about its affiliation with the same object as pixel $p1$

belongs to. The label collision will occur when the pixel $p3$ will be encountered as two of its neighbouring pixels are already assigned with different labels. A typical solution for this problem is to record label equivalences in auxiliary arrays and keep scanning the image while during the second or one of the consecutive image scans all the label ambiguities will be resolved. How to address label collision problem is further discussed in the following subsections.

5.1.3 Feature Extraction

In computer vision, feature extraction is a process when the input data is transformed into a reduced representation set of features. The extracted set of features represents the relevant information from the input data in order to perform the desired task, e.g. classification or statistical pattern recognition.

There is a wide range of features that can be extracted from binary image B. Most common used by classification or tracking algorithms are position, size, width, height, CoG, colour or texture. The size is determined by the number of pixels within the connected component and can be calculated as moment-zero M_{00} according to:

$$M_{ij} = \sum_x \sum_y x^i y^i B(x, y). \tag{5.2}$$

Position, width and height of the object can be calculated in the simplest form using two coordinates of the bounding box (top-left and bottom-right). The CoG coordinates however can be calculated according to the following equation:

$$\overline{x} = \frac{M_{10}}{M_{00}}, \quad \overline{y} = \frac{M_{01}}{M_{00}}. \tag{5.3}$$

Both the colour and texture features can be extracted by superimposing labelled object mask on the input colour image. An example image with extracted features based on the binary motion mask is depicted in Figure 5.2. It is important to define the set of features suitable for particular application, for instance extracted texture does not bring as much information as position or size when tracking.

In video processing surveillance systems, feature extraction constitutes a significant processing stage. Since most applications do not require uniquely labelled object mask, the classical approach for labelling [99] can be significantly improved in terms of speed, memory usage and computation power reduction. This will be addressed in the further subsections.

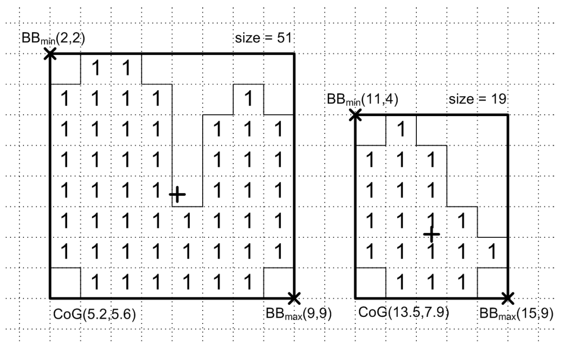


Figure 5.2: An example connected component with extracted binary features

5.2 Literature Review

The way that CCL algorithms distinguish disjoint groups of connected pixels and assign them with unique identifiers has changed over the years. This section gives an overview of how the CCL algorithms evolved since the initial release.

5.2.1 Labelling Algorithms

- **Rosenfeld and Pflatz (1966)** - the classical approach for CCL was described by Rosenfeld and Pflatz in [99]. This algorithm requires two passes through the binary image, also a large matrix to store label equivalences. The first scan is performed as described in Subsection 5.1.1. Whenever the label collision is encountered, both labels are recorded in an equivalence matrix. During the second scan, connected components are re-labelled based on information stored in the equivalence matrix. As a result, after two scans through the image, each connected component is assigned with a unique label.

  The main drawback of this algorithm is a necessity of using equivalence matrix, which for more complex images (higher number of label collisions) can be very large. This implies a long processing time before final labels could be assigned.

- **Nassimi and Sahni (1980)** - the CCL algorithm described by Nassimi and Sahni in [100] was designed for highly specialized mesh-connected parallel computers (MCC), called SIMD (Single Instruction Stream, Multiple Data Stream) platforms. That was one of the first attempts for implementation of CCL using parallel processing systems.

  The implementation of this algorithm requires great amount of logic resources. Also, due to its complexity, the size of an input image frame has to be limited to small arrays.

- **Haralick (1981)** - an iterative algorithm which does not require any auxiliary storage for label equivalences was introduced by Haralick in [101]. This technique involves multiple forward and backward raster scan passes through the image until no label change occurs. All the label collisions are solved on the local neighbourhood basis.

  This algorithm was designed for systems with limited memory resources processing on low resolution images. The performance of this algorithm is highly related to the size and the complexity of the input data thus it is not recommended for higher resolution images.

- **Lumia, Shapiro and Zuniga (1983)** - the CCL algorithm by Lumia *et al.* [102] was introduced as improvement to both previously described serially processing algorithms. It employs the local equivalence table where only labels within the current line are stored. Hence, the equivalence table size will not exceed the line width. The image is scanned multiple times until no label change occurs, however thanks to the use of the local equivalence table, the number of scans was significantly reduced compared to the algorithm proposed by Haralick [101].

  This approach proved to be more efficient in terms of processing time. Thanks to reduced size of the equivalence table, memory requirements were also reduced.

- **Samet and Tamminen (1986)** - a description of the CCL algorithm based on the *Union-Find* approach together with empirical results can be found in [103]. Label equivalences are decoded using rooted trees as depicted in Figure 5.3(a). This research proved that there is a quasi linear solution for typical labelling problems. This algorithm requires two progressive scans through the image. Although it is not the most efficient implementation for the CCL, the *union-find* approach became very popular and was commonly used in a number of future CCL implementations.

  Recent research proved that implementations of the labelling algorithms based on the *union-find* approach are more efficient and less memory consuming when rooted trees are exchanged with the array-based structure, see Figure 5.3(b).

- **Chang and Chen (2003)** - a new variation of the CCL algorithm based on the contour tracing approach was introduced by Chang and Chen in [104]. It is based on improved techniques previously described in [105] and [106]. They claim a single pass through the binary image is sufficient in order to label all the objects within the image frame.

  Although there are implementation limitations caused by the architecture require-

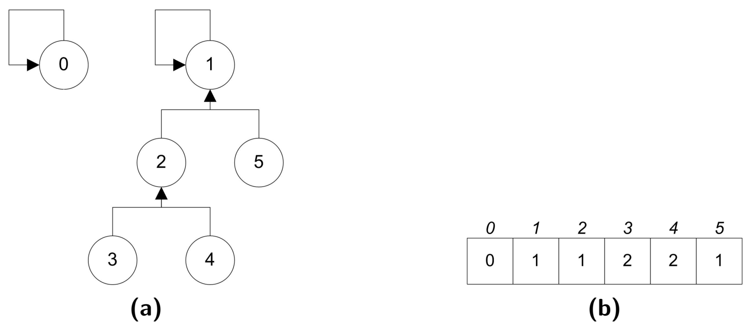


Figure 5.3: Equivalence labels encoding. (a) Equivalence labels encoded using rooted tree. (b) An array-based structure.

ments (e.g. random access to all the image pixels), it was proved that this approach gives superior results comparing with other labelling algorithms.

- **Suzuki, Horiba and Sugie (2003)** - a new approach for CCL based on multiple image scans [101] was introduced by Suzuki *et al.* in [107]. The image is being scanned multiple times in forward and backward directions, while one-dimensional table, which memorizes label equivalences is used for uniting equivalent labels. The proposed algorithm has a desirable characteristic: the execution time is directly proportional to the number of pixels in connected components in an image.

  Since the algorithm is based on the sequential local operations, there is no need to employ large equivalence tables, which often require large amount of memory resources and processing time. The execution time is directly proportional to the number of pixels in connected components within the image. Thanks to the use of local operators, it is suitable for hardware implementation.

- **Bailey and Johnston (2007)** - a new technique for connected component analysis based on the single pass through the binary image was described in [108]. It does not comply with traditional sense of CCL. Instead of using a second image frame to store labelled object masks in order to extract their features of interest, they are calculated simultaneously with the input data being provided to the system. This ensures real-time processing speed. Also, there is no need to buffer an input image frame, hence the memory requirement can be significantly reduced.

  Although the single pass technique does not complete the labelling in the traditional sense, results of the processing are sufficient for most of the image processing applications. The significant memory savings together with high throughput make this algorithm an interesting alternative.

5.2.2 Summary

The problem of labelling dates back to the beginnings of the computer vision. In this section a chronological review of the main CCL algorithms was provided. These algorithms significantly differ in data management and execution time. They can be classified into the following groups:

(a) two scans,

(b) multiple scan,

(c) parallel,

(d) contour tracing,

(e) single pass.

Algorithms from the group (a) are based on the classical approach for CCL developed by Rosenfeld and Pflatz [99]. They share a key feature which is the constant number of two scans through the image frame. They evolved significantly over the years. The amount of additional storage for label equivalences was reduced thanks to the use of the array-based union-find approach and their performance improved. They are currently the most popular algorithms for computer-based image processing applications, e.g. Matlab Image Processing Toolbox [109]. Thanks to the constant processing time they can be applied to real-time video processing systems, however they require relatively large amount of memory to output frame with label equivalences. Therefore they are not the suitable choice for memory limited embedded systems. An example FPGA implementation of such an algorithm can be found in [110].

Multiple scan algorithms can be easily implemented in hardware as they are based on sequential local operations. They also do not occupy significant amount of memory resources since they do not require auxiliary storage for label equivalences. However, since the processing time (number of image scans) is highly dependent on the image complexity, they cannot be applied for applications where the time is of critical importance, e.g. surveillance systems. The architecture aimed for FPGA devices based on a serial and recursive CCL algorithm can be found in [111].

The group (c) contains algorithms highly specialized for parallel processing platforms that are not suitable for ordinary computer architectures. These systems often are based on one logical processing element per pixel. An example of such an implementation was described in [112]. These algorithms, although suitable for FPGA implementation, require great amount of logic resources. Recent implementations proved to be less resource consuming however they are still too large for implementation using a single device.

An FPGA-based processing platform capable of processing an image frame while the image is loaded was proposed by Mozef *et al.* in [113].

The contour tracing algorithms are an interesting alternative to the ones described previously. Their main feature is the contour tracing approach for CCL. In contrast to algorithms classified in groups (a) and (b), the contour tracing approach does not comply with the raster scan. Instead, it requires a random access to the image pixels (here contour pixels). This implies an initial image scan in order to buffer an input frame. Despite this, the contour tracing algorithm proved good efficiency as well as lower memory requirement comparing with algorithms based on the progressive scan. An example of a successful hardware implementation can be found in [114].

The last group (e) of labelling algorithms is relatively new and was developed specifically for streaming data systems. Only one scan per image frame is required in order to extract features of interest for all the connected components. This allows to significantly reduce the processing time as well as the amount of required memory. Although, results of the processing (extracted features) are sufficient for most object detection and counting systems, the algorithm is not applicable to applications where the labelled object mask is required. An example of FPGA-based implementation can be found in [115, 116].

In this section a variety of CCL algorithms were introduced and classified into five general groups. The aim of this review was to compare their capabilities for real-time video processing. Due to obvious reasons, two of the listed groups will not be taken into consideration in further analysis: firstly, the multiple scan algorithms due to variable processing time, and secondly, the parallel algorithms due to large resource requirements even for lower resolution image frames. Other algorithms that comply with real-time CCL processing criteria will be further analysed in the following subsections.

5.3 Algorithm Analysis

In this section the following groups of CCL algorithms will be taken into consideration:

- classical approach,
- contour tracing,
- single pass.

All of them proved real-time video processing capabilities. The performance together with their memory requirements will be discussed in detail. They will be calculated for a binary image frame with R rows and C columns. As a result, one of the algorithms will be chosen for hardware implementation.

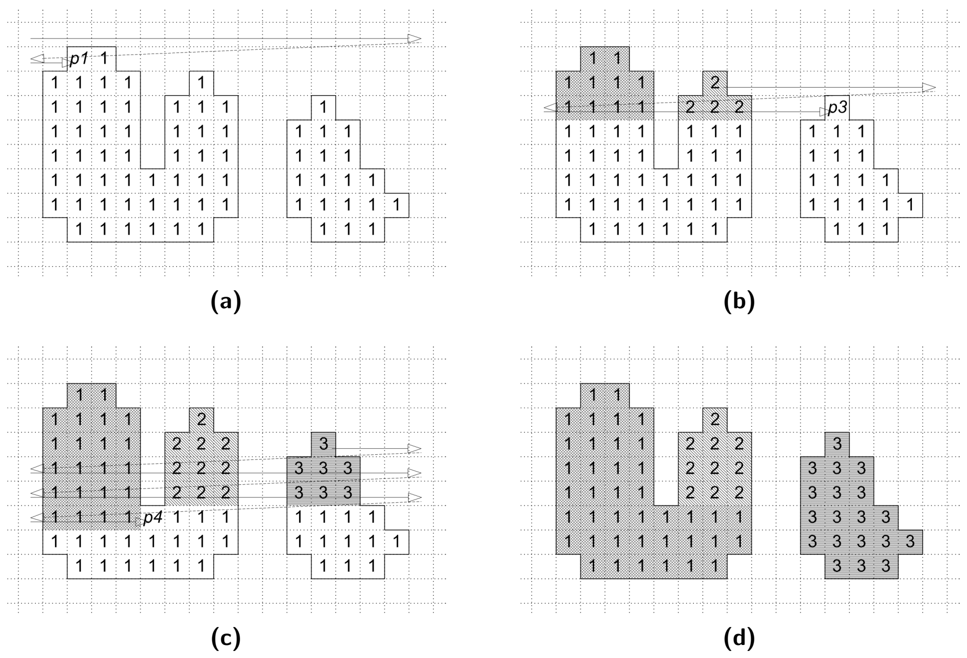


Figure 5.4: Classical approach for CCL. (a) Assign a new label for a foreground pixel. (b) Repeat the previous step. (c) Store both labels in the equivalence table. (d) Scan the image once again and reassign labels with the ones from the equivalence table.

Table 5.1: Equivalence table

provisional label	equivalence label
1	1
2	1
3	3
4	-

5.3.1 Classical Algorithm

The classical approach for CCL requires two consecutive scans through the binary image frame. During the initial scan (see Figure 5.4), foreground pixels are assigned with preliminary labels, whereas label collisions are stored in the equivalence table, as can be seen in Table 5.1. During the consecutive image scan, all the labels are reassigned with the ones stored in the equivalence table. Once the image frame is labelled completely, features of interest can be extracted. This step takes another image scan however the classical algorithm can be optimized in order to complete labelling and feature extraction within two scans.

Table 5.2: Classical algorithm memory requirement for variable image size with up to $CC_{max} = 255$ objects per frame

R [pixels]	C [pixels]	FE [Kbits]	ET [Kbits]	$total$ [Kbits]
320	240	4	585	1,281
640	480	5	585	3,355
1024	768	5	585	7,668

Memory Requirements

It is difficult to estimate the exact amount of memory required by the classical algorithm due to the fact it depends on the image complexity (number of label collisions). It can be calculated according to:

$$\begin{aligned} mem_{total} = &\lceil log_2(CC_{max} + CC_{lcol} + 1)\rceil \cdot (R \times C) + \\ &+ mem_{ET} + mem_{FE}, \end{aligned} \tag{5.4a}$$

where

$$mem_{ET} = \lceil log_2(CC_{max} + CC_{lcol})\rceil \cdot (CC_{max} + CC_{lcol}), \tag{5.4b}$$

and

$$mem_{FE} = (2 \cdot \lceil log_2(R)\rceil + 2 \cdot \lceil log_2(C)\rceil) \cdot (CC_{max}), \tag{5.4c}$$

where CC_{max} defines the maximum number of connected components, CC_{lcol} number of label collisions, $\lceil\cdot\rceil$ is an operator rounding $\cdot$ to the nearest upper integer and the $+1$ comes from the fact that 0 is a preoccupied label. The frame resolution is determined by the number of R rows multiplied by the number of C columns. The Equation (5.4b) specifies the amount of memory required by the equivalence table, whereas the last equation memory required by the table which accumulates extracted features of the connected components. Each memory location stores coordinates of two points $((x_{min}), (y_{min}))$ and $((x_{max}), (y_{max}))$ that correspond to the top-left and bottom-right points of the rectangular bounding box respectively. With this data both position and size of the connected component can be calculated.

Assuming one label collision per connected component, the total amount of required memory for three different image sizes: 320×240, 640×480 and 1024×768 with $CC_{max} = 255$ objects per image frame was calculated and can be found in Table 5.2.

Execution Time

As it was mentioned previously, the classical algorithm requires two consecutive image scans in order to label all the objects within the image frame. In order to extract features such as position, size, bounding box, etc., the classical approach for CCL requires additional image scan. However, with improved data administration, the third image scan can be dismissed [108]. Processing complex images with multiple label collisions can be time consuming (large equivalence table). For hardware-based video processing system this can be completed during horizontal or vertical blanking periods. Hence, during the second image scan all the preliminary labels can be reassigned with the ones from the already preprocessed equivalence table. The total execution time will not exceed two image scans and can be written as $time_{exe} = 2 \cdot (R \times C) + b$, where b is the time required to unify equivalence table during the vertical blanking period between two scans. Therefore, it can be omitted and results in $time_{exe} = 2 \cdot (R \times C)$.

5.3.2 Contour Tracing Algorithm

The contour tracing algorithm for CCL developed by Chang and Chen [104, 117] is an interesting alternative to algorithms based on the progressive scan. Here, the image frame is also scanned in raster order, however thanks to the contour tracing capability, all the connected components can be distinguished in a single image scan as the problem of label collisions does not apply.

In order to label a binary image frame, the input image has to be buffered in the memory. The image is scanned until a foreground pixel $p(x, y) = F_f$ is encountered as depicted in Figure 5.5(a). The complete trace of the contour is performed until the same pixel is reached again, see Figure 5.5(b). The contour pixels are labelled with index l_k for $L \subset \mathbb{N}$, where $3 \leq k \leq K$ and K defines the total number of connected components within a frame. Also, horizontal neighbouring pixels are labelled with the border label ($l_k = 2$). Once the contour is labelled completely, the index k is incremented by 1 and the algorithm resumes scanning step, as depicted in Figure 5.5(c). At this point, one of several pixels can be encountered:

- background pixel: $p(x, y) = F_b$,
- unlabelled foreground pixel: $p(x, y) = F_f$,
- already labelled contour pixel: $p(x, y) > 1$,
- horizontal border pixel: $x = C$.

When background pixel is encountered, the algorithm keeps scanning the image and no further action is required. However, for $p(x, y) = F_f$, the algorithm starts the contour

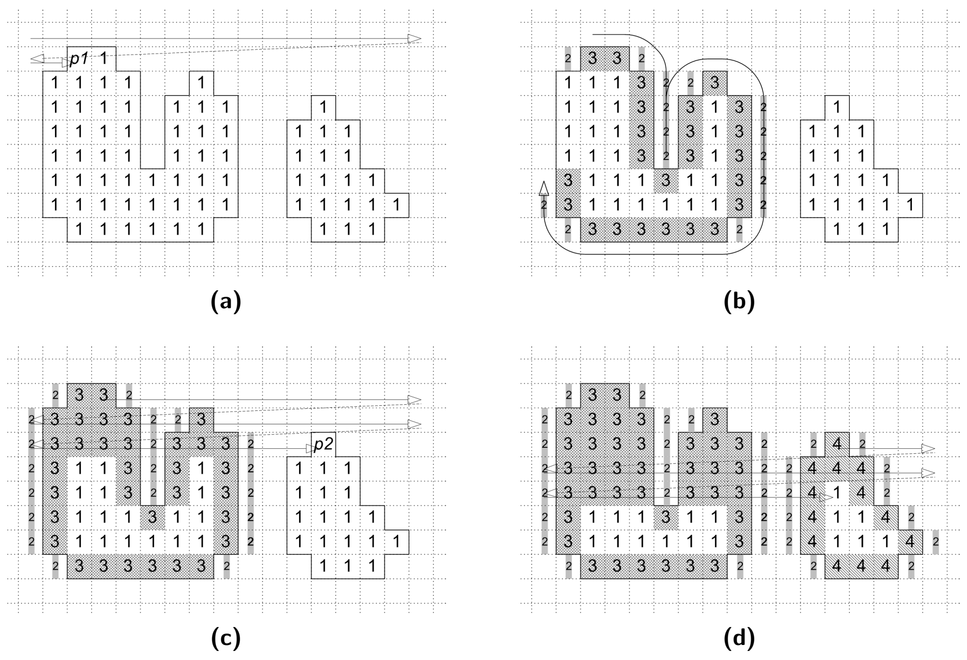


Figure 5.5: Contour tracing approach for CCL. (a) Look for the foreground pixel and start tracing the contour. (b) While tracing, assign horizontal neighbouring pixels with border label. (c) Fill interior areas with labels and keep scanning until a new contour pixel is encountered. (d) Keep scanning until all the components are labelled completely.

tracing procedure as described above. Once the border pixel $p(x, y) = 2$ is encountered, the algorithm reads the label of the subsequent contour pixel $p(x, y) = l_k$ and keeps scanning within the contour pixels while label l_k is assigned to all the pixels $p(x, y) = F_f$ until another pixel $p(x, y) = 2$ is encountered. Once the last pixel in a row is reached ($x = C$), the scan continues from the first pixel in the next row according to raster scan.

However, not all the connected components within a binary image are solid blobs, they may contain holes. The contour tracing algorithm traces internal contours in the same manner as described above. Some pixels may be both internal and external contour pixels at the same time. In order to avoid tracing them multiple times, the surrounding border label $l_k = 2$ is used. A detailed description of the contour tracing algorithm implementation can be found in [117].

Memory Requirements

Unlike CCL algorithms based on the classical approach, the amount of memory required by the contour tracing algorithms is constant for a specified number of connected components. It varies in direct proportion to the image resolution and can be calculated according to:

Table 5.3: Contour tracing algorithm memory requirement for variable image size with up to $CC_{max} = 255$ objects per frame

R [pixels]	C [pixels]	FE [Kbits]	$total$ [Kbits]
320	240	4	158
640	480	5	619
1024	768	5	1578

$$mem_{total} = \lceil log_2(CC_{max} + 3) \rceil \cdot (R \times C) + mem_{FE}, \tag{5.5}$$

where mem_{FE} is derived from Equation (5.4c) and +3 comes from the fact labels 0, 1 and 2 are already preoccupied by background, foreground and reserved labels respectively. In comparison with Equation (5.4a), assuming one label collision per connected component ($CC_{max} = CC_{lcol}$), algorithms based on the classical approach require additional $(R \times C)$ bits.

Moreover, memory requirement for the contour tracing algorithm can be significantly reduced. It comes from the fact that labelling algorithms are often employed in order to extract features of interest for detected objects and the uniquely labelled object masks are not needed. For those applications contour pixels for all the connected components can be assigned with the same label $l_k = 3$ whereas their features can be still extracted. Hence, memory requirements can be calculated according to:

$$mem\_2\text{-}bit_{total} = 2 \cdot (R \times C) + mem_{FE}. \tag{5.6}$$

The Table 5.6 shows the difference between the regular and the reduced 2-*bit* approaches for CCL based on the contour tracing.

Execution Time

The total execution time is determined by the number of memory operations on the image frame. During the initial image scan, all the image pixels are written into the labelling memory simultaneously with data acquisition. The second image scan however involves additional read/write operations for the contour tracing procedure. Since the number of contour pixels to be traced depends on the image complexity, it is difficult to determine the number of pixel operations, but it will not exceed $R \times C$ operations. Hence, the total execution time can be written as $time_{exe} \leq 3 \cdot (R \times C)$.

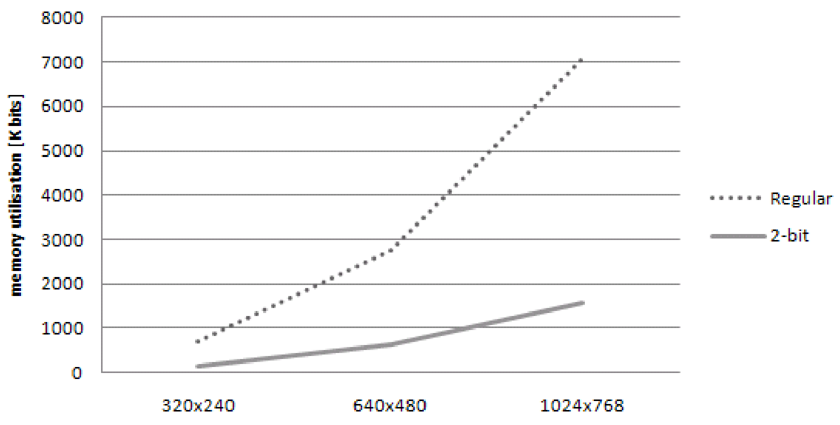


Figure 5.6: Classical algorithm memory requirement for the regular and the reduced 2-*bit* approaches for variable image size with $CC_{max} = 255$ objects per frame

5.3.3 Single Pass Algorithm

The data management for the single pass algorithm significantly differs from other previously discussed algorithms. Firstly, since they operate on the streaming data, there is no need to buffer an input image. Secondly, all the features of interest can be extracted while the data is provided to the system, an output image with labelled object masks does not have to be stored neither. This allows to significantly reduce the amount of memory required by the system.

In order to extract features of interest in a single image scan, the algorithm requires fully pipelined architecture. There are multiple memory units required, such as Line Buffer (`BUF`), Lookup Table (`LOOKUP`) and Data Table (`DATA`), depicted in Figure 5.7(b). The `BUF` stores all the labels assigned within the last line. The `LOOKUP` is a data table used to manage pointers/equivalence labels and the `DATA` is used to store all the extracted features of interest.

Once the new foreground pixel $p(x, y) = F_f$ is appointed, a new label l_k is assigned and stored in the `BUF` at location x. Simultaneously, features for this object are extracted and stored in the `DATA` table at location l_k. If the label collision is encountered (pixel $p4$) as can be seen in Figure 5.7(a), both labels stored in the `BUF` at $(x - 1)$ and $(x + 1)$ need to be merged as well as their features. Also, the `LOOKUP` needs to be updated. A detailed description of the single pass algorithm will be provided in the subsequent section. A block diagram of the single pass algorithm operation overview is depicted in Figure 5.7.

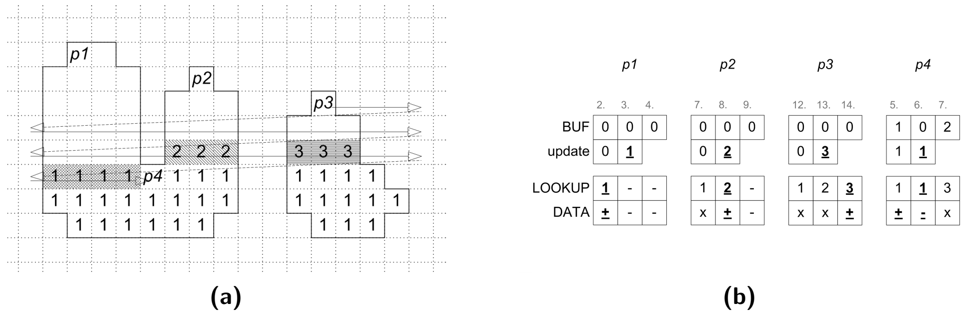


Figure 5.7: Single pass approach for CCL. (a) Scan the image frame in raster order. (b) Memory registers. Values recently changed are underlined, "-" indicates empty cell, "x" previously assigned value, "+" stands for an update.

Memory Requirements

According to the description of the single pass algorithm in [108], the amount of required memory is proportional to the image size (the width of the image to be precise), the maximum number of connected components per frame and the number of label collisions. Hence, the total memory requirement can be calculated according to:

$$mem_{total} = mem_{BUF} + mem_{LOOKUP} + mem_{DATA}, \tag{5.7a}$$

where

$$mem_{BUF} = \lceil log_2(CC_{max} + CC_{lcol} + 1)\rceil \cdot C, \tag{5.7b}$$

$$mem_{LOOKUP} = \lceil log_2(CC_{max} + CC_{lcol} + 1)\rceil \cdot (CC_{max} + CC_{lcol}), \tag{5.7c}$$

and

$$mem_{DATA} = (2 \cdot \lceil log_2(R)\rceil + 2 \cdot \lceil log_2(C)\rceil) \cdot (CC_{max} + CC_{lcol}). \tag{5.7d}$$

In further subsections a detailed description of the algorithm with improved memory management will be provided. With this approach, the problem of label collisions will not affect the demand for memory resources, the sum of $CC_{max} + CC_{lcol}$ in Equations (5.7b) to (5.7d) can be reduced to CC_{max} with a little overhead which can be omitted. The algorithm also uses small amount of operating memory for label merging and data handling. However, it is small enough to be ignored. The memory requirement for improved implementation was calculated and can be found in Table 5.4.

Table 5.4: Single pass algorithm memory requirement for variable image size with up to $CC_{max} = 255$ objects per frame

R [pixels]	C [pixels]	`DATA` [Kbits]	`BUF` [Kbits]	`LOOKUP` [Kbits]	*total* [Kbits]
320	240	4	2	2	9
640	480	5	4	2	11
1024	768	5	7	2	14

Table 5.5: CCL algorithms comparison table for 640×480 image size with $CC_{max} = 255$

CCL algorithm	memory requirement [Kbits]	execution time [clock cycles]
Classical approach	$3,355$	$= 2 \cdot (R \times C) + b$
Contour tracing	619	$\leq 3 \cdot (R \times C)$
Single pass	11	$= (R \times C) + m$

Execution Time

The analysis of the execution time for the single pass algorithm is straightforward. The algorithm requires a single scan through the image frame with a little overhead m for label merging that can be handled during horizontal blanking periods. Hence, the execution time can be written as $time_{exe} = (R \times C) + m$.

5.3.4 Conclusions

In this section three major groups of CLL algorithms were described and analysed in terms of execution time and memory consumption. All of them meet real-time video processing requirement. They are also suitable for hardware implementation. The comparison table for memory utilisation and execution time is presented in Table 5.5. Moreover, the Figure 5.8 gives a graphical comparison of memory requirement for CCL algorithms with variable image size 320×240, 640×480 and 1024×768 pixels, with up to $CC_{max} = 255$ connected components per frame.

In comparison, both the classical and the contour tracing algorithms feature similar capabilities. The contour tracing algorithm requires less memory resources, whereas the classical approach can guarantee constant processing time in two image scans. However, the single pass approach outperforms both the previously mentioned algorithms in all the compared aspects. This comes from the fact the single pass algorithm does not require stalled input image prior to labelling neither the output labelled frame in order to extract objects' features. Moreover, the algorithm can operate simultaneously with the data acquisition system on the pixel clock frequency (the lowest frequency for real-time

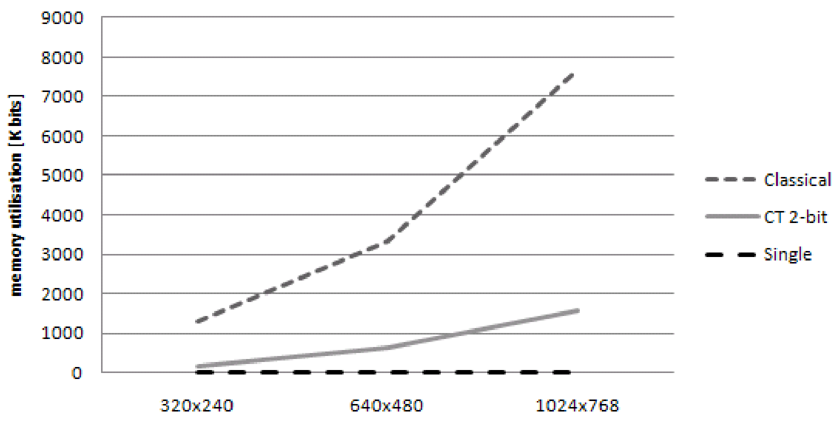


Figure 5.8: Memory requirements for labelling algorithms with $CC_{max} = 255$ objects per image

processing), whereas other algorithms provide results at the same rate while operating on twice (or three) times higher frequency.

In overall, the single pass approach for CCL operates at least twice faster compared to other algorithms. When operating on 640×480 pixels and up to $CC_{max} = 255$ connected components per image frame, it requires up to 299 and 55 times less memory resources comparing with classical approach and the contour tracing algorithms respectively. Hence, the single pass algorithm was chosen for the hardware implementation.

5.4 Implementation

Based on the overview provided in the previous subsections, the single pass CCL algorithm was chosen for hardware implementation. This algorithm outperforms other techniques in terms of memory requirement as well as execution time. It is capable of extracting features of interest such as position, width, height, size and CoG in a single pass through the binary image frame.

In this section a unique approach for CCL feature extraction will be described. Since it is based on the single pass CCL algorithm, the raster scan labelling ambiguities comply and will be discussed. An overview of the system architecture will be provided with detailed description of the system components. This includes description of module instantiation. The section will be concluded with experimental results and performance analysis. This will be followed by conclusions and discussion on further improvements.

5.4.1 Introduction

Thanks to the low memory requirement and short execution time the single pass algorithm for CCL was adopted for a wide range of image processing applications [118–120].

Although, all of them feature single line labelling buffer, they differ in data administration and memory management. The architecture of implemented algorithm is determined by the set of features to be extracted.

The relatively straightforward approach for single pass CCL was developed and described by Pedre *et al.* in [118]. All the labels within the last line are kept in the stack. If the current pixel is a foreground pixel $p(x, y) = F_f$, two of the adjacent pixels are checked whether they are already labelled: the one on the left $p(x - 1, y)$ and the one directly above $p(x, y - 1)$. The current pixel is assigned with a label according to the classical labelling routine as described in subsection 5.1.1. However, if the label collision occurs, the smaller label is chosen and all the occurrences of the other label within the stack are overwritten with the smaller label. All the stack entries are updated in a single clock cycle. Although this algorithm allows to label and extract features for the "worst-case-scenario" input stream in a single image scan with minimal overhead, an implementation of such algorithm requires great amount of logic resources due to the use of comparators and multiplexers for every stack entry. Therefore, it should not be considered for higher resolution video signals.

Other implementations, such as [115] and [119], process data serially, with most of the memory elements implemented in BRAM embedded memory blocks. Such an approach guarantees lower complexity and occupies less hardware resources. Therefore, it is suitable even for smaller FPGA devices. However, it was not specified whether it can handle the "worst-case-scenario" - multiple cross mergers within a single line. This issue will be further discussed in the following subsections.

Algorithms described in [115, 116] and [119, 120] were developed on the pointer based approach. Therefore, every time labels are merged, not only the label line buffer (`BUF`), but also the pointer Lookup Table (`LOOKUP`) is updated. The `LOOKUP` keeps pointers to all the labels stored within the `BUF`. The pipelined system fetches data from the `BUF` (the label of the top-adjacent pixel) and then looks in the `LOOKUP` for its equivalence label. Thanks to the `LOOKUP` the size of the system architecture can be significantly reduced.

The architecture of the algorithm described in the following subsections is based on the 4-neighbourhood connectivity $N_4(p)$ mask. Since it allows to extract the same set of features of interest as the system based on the 8-neighbourhood connectivity $N_8(p)$ mask, it guarantees lower complexity system architecture.

5.4.2 Single Pass Algorithm

An implementation of the single pass algorithm described in these thesis is based on the pipelined pointer based approach. An overview of the system architecture is illustrated in Figure 5.9. As can be seen, there are multiple memory units involved, such as `DATA`,

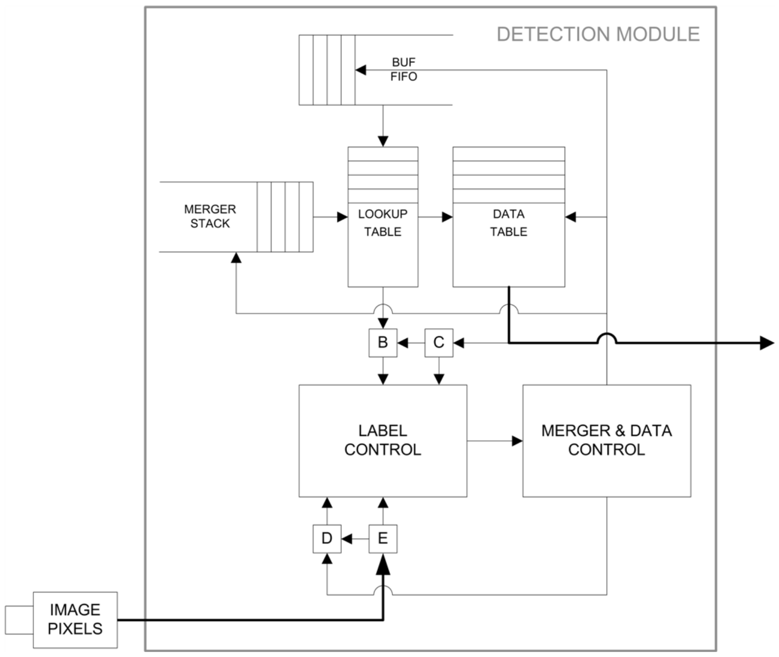


Figure 5.9: System architecture block diagram

BUF and LOOKUP to store processing results, labels assigned within the last line and their pointers, respectively. They will be described in the following subsections. The MERGER STACK together with MERGER & DATA CONTROL unit were implemented in order to manage multiple cross mergers. A unique ”label-reuse” approach was developed and embedded within the LABEL CONTROL unit to keep the memory requirement as low as possible. All these memory and control blocks will be described in detail in the following subsections. This architecture assures correct object detection and feature extraction even for the worst case scenario binary input data.

Scan Mask

An input data is provided to the detection module on the pixel basis, according to the raster scan. The current pixel $p(x, y) = F_f$ or $p(x, y) = F_b$ is stored in the register E, located at the position (x, y) of the scan mask M_s, introduced in Chapter 2. The register D, located at position $(x - 1, y)$ stores the label assigned in the previous clock cycle. This data was shifted in the window filter manner from the register E after it was stored in the BUF. The label of the second adjacent pixel located at $(x, y-1)$ is stored in the register B. The data for this register is provided by the LOOKUP. The register C was employed in order to read previously calculated features from the DATA for one clock cycle in advance.

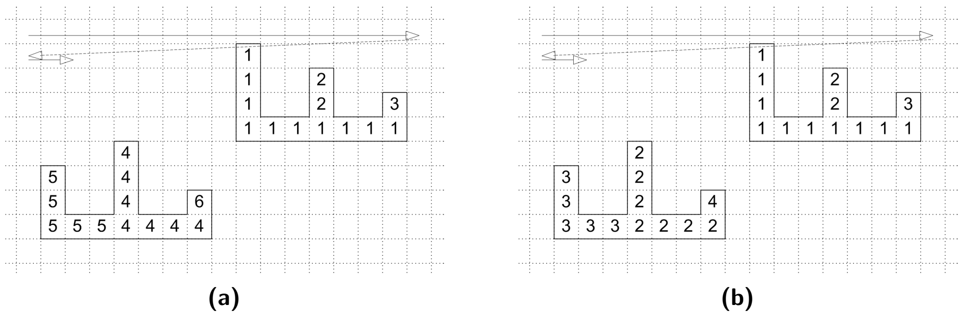


Figure 5.10: Multiple label merge in a single line scan. (a) Traditional approach. (b) Label-reuse approach.

Label Selection

The current location is assigned with a label according to the classical labelling routine described in subsection 5.1.1. If the label collision occurs, a common approach is to use the lower label and rewrite the pointer in the `LOOKUP`. This approach (lower-label first) simplifies the way multiple mergers are resolved - the lower label (the one assigned earlier during the raster scan) is always used as a pointer for other labels [116, 118, 119].

The Figure 5.10(a) illustrates two objects requiring multiple mergers. Once the label collision occurs, the label 1 is used as a pointer for labels 2 and label 3. Similar situation applies to the second object - both labels 5 and 6 within the `LOOKUP` are overwritten with label 4. After the merging, higher labels are disposed, they will not be used any more. This is an inefficient use of memory resources.

For the purpose of this implementation, the label-reuse approach was developed. As can be seen in Figure 5.10(b), labels 2 and 3 that were merged with label 1 within the first object were also used for the second detected object during the image scan. This approach however requires additional circuitry for label selection control, labels are assigned based on their extracted features. Therefore, a random access to the `DATA` memory is required. These issues will be discusses in the following subsection.

Lookups

The `LOOKUP` equivalence table is in the key importance of the pointer based single pass algorithm implementation. It is hard-wired to the `BUF` and gives current pointers to all the labels assigned within the last line. Let us consider a "W" shaped object in the binary image illustrated in Figure 5.11. Once labels 1 and 3 are merged, the `LOOKUP` pointer for label 3 is updated with 1, hence the next occurrence of label 3 in the `BUF` will be pointing to the label 1. In most cases this approach is sufficient. The problem of cross mergers will be further discussed.

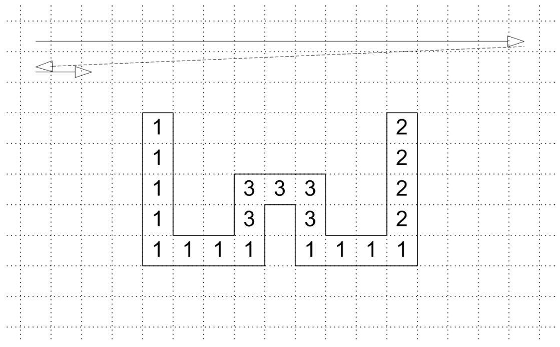


Figure 5.11: *"W"* shaped object

Mergers

All the algorithms based on the raster scan struggle with the label collision problem. In a typical video processing application the data to be merged is not usually as complex as examples presented in this subsection, however they have to be taken into consideration for the system integrity.

There are two chains of mergers illustrated in Figure 5.12. For every merger depicted in Figure 5.12(a) the `LOOKUP` will be updated immediately therefore at the end of the line scan all the `BUF` entries will have their pointers in the `LOOKUP` updated. Once the pixel "?" is appointed, it will be assigned with label 1.

The chain of mergers depicted in Figure 5.12(b) is more complex and requires different approach for label merging. After the first merger the `LOOKUP` will be updated as follows: $(4 \leftarrow 3)$, whereas during the next merger the lower label will be updated according to: $(3 \leftarrow 2)$, similar with the last merger: $(2 \leftarrow 1)$. Therefore, once the pixel "?" is appointed, it will be assigned with label 3. In order to obtain its correct identifier, the `LOOKUP` would have to be accessed a number of times $(4 \rightarrow 3,\ 3 \rightarrow 2,\ 2 \rightarrow 1)$. Due to the variable execution time, it can not be accepted for real time video processing applications. A straightforward solution would be to search through the whole `LOOKUP` every time merger occurs and check whether the current merger does not affect one of the previous mergers. However, this may be time consuming and could affect further data analysis. The other solution would be to access all the `BUF` entries and update them with new data. Such an approach, suggested by Pedre *et al.* in [118], does not involve the `LOOKUP`, however it would require concurrent access to all the memory cells which results in great demand for hardware resources. Therefore, it is not suitable for higher resolution image arrays.

An interesting alternative was suggested by Bailey and Johnston in [108]. For every merger with the lower label assigned to the left-adjacent pixel, the `LOOKUP` is updated immediately. However, for mergers with the top-adjacent pixel assigned with the lower

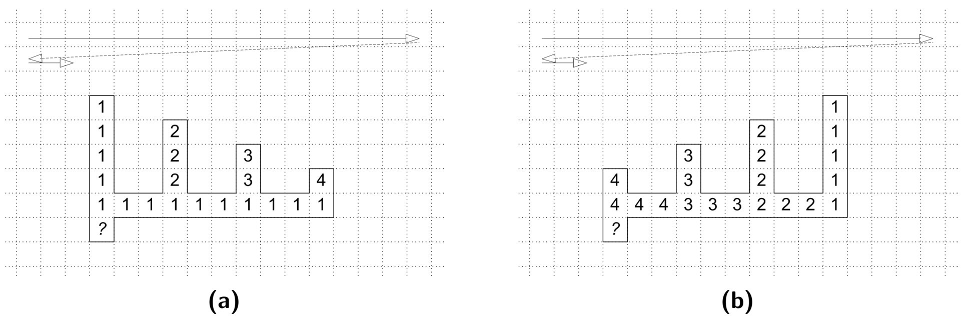


Figure 5.12: Merger chain. (a) Descending labels. (b) Ascending labels.

label, both labels are pushed onto the `MERGER STACK`, all the ambiguities could be resolved at the end of the line scan, in the reverse order. Hence, the image is scanned without delays. According to the binary image depicted in Figure 5.12(b), every couple of labels to be merged is pushed onto the `MERGER STACK` as follows: 4 & 3, 3 & 2, 2 & 1, whereas at the end of the line scan, the `LOOKUP` is updated in the following order: $1 \rightarrow 2, 2(1) \rightarrow 3, 3(1) \rightarrow 4$. As a result, the pixel ”?” will be assigned with the label 1.

This approach involves implementation of a stack unit, it also involves additional circuitry to manage label mergers during the horizontal blanking period, however it results in a relatively small resource utilisation comparing with other solutions.

Data Table

The purpose of Data Table memory module is to store extracted features. It was depicted as `DATA` in Figure 5.9. It is accessed through the label pointer stored in the `LOOKUP`. Since the label-reuse approach applied, the merger routine does not depend on the label numbers as in [108, 116]. This implies an update of y_{min} values for both the labels in `DATA`.

5.4.3 Architecture

The architecture of the detection and feature extraction unit was developed as fully customizable generic HDL module. It was developed in VHDL (Very-high-speed integrated circuit Hardware Description Language) using behavioural HDL inference templates to incorporate embedded memory modules. Other storage elements such as stack unit were also developed in VHDL. This approach is semi-device independent, therefore design files can be used for other implementations as third party IP. The block diagram of the system architecture was depicted in Figure 5.13. All the system blocks will be described in the following subsections.

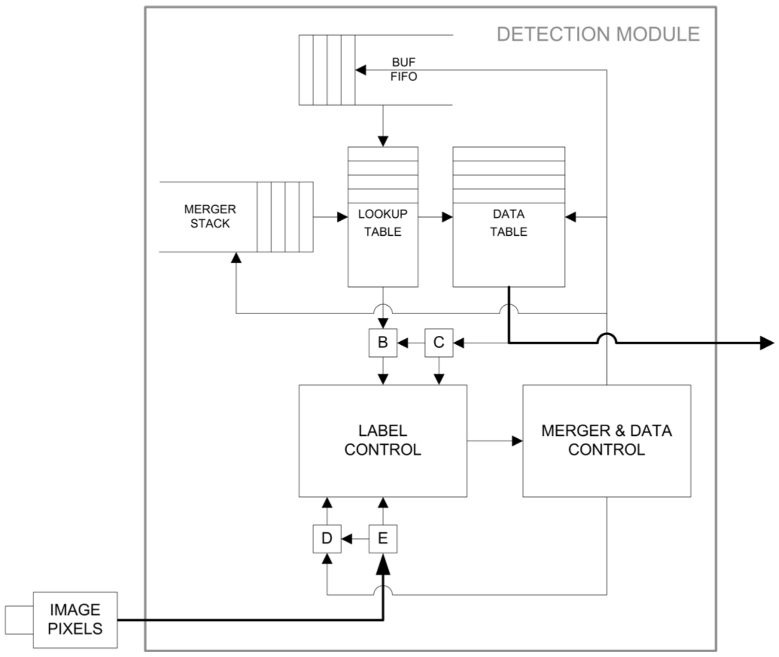


Figure 5.13: System architecture block diagram

BUF FIFO

The `BUF FIFO` is a line buffer used to store all the labels assigned during the line scan. The write operation is performed every clock cycle if the current pixel $p(x, y)$ is located within the visible area of the image frame ($x < C$ and $y < R$). Every memory location is assigned with the number of the current label $0 \leq l_k \leq CC_{max}$, the address width is determined by the image width C.

Since the `BUF` operates on the pixel clock synchronously with the image scan, the data read from the buffer is accessed for three pixels in advance to allow the `LOOKUP` and `DATA` time margin to access data before it will be processed by the `LABEL CONTROL` unit. The constant simultaneous read and write operations require dual-port architecture of the `BUF` memory unit.

LOOKUP TABLE

The `LOOKUP` memory module is hard-wired with the `BUF` and is accessed every time the `BUF` read operation is issued. All the label processing is performed on the data read from the `LOOKUP`. Moreover, every time during the image scan a new label is created, the pointer to this label is written to the memory. There are two cases when the write operation is issued - when two labels are merged therefore an immediate update of the `LOOKUP` is required, also during the stack-based merging during the horizontal blanking period. Therefore, a dual-port memory architecture is required. The size of this memory module is defined by the maximum number of connected components per image frame, which was determined based on the simulation results performed on experimental video sequences.

DATA TABLE

The `DATA` memory is of key importance for the implementation of the single pass CCL algorithm. This is where all the extracted features of interests are stored. This data is constantly used during the image scan in order to distinguish disjoint connected components. Once the image scan is completed, it can be forwarded for further processing, e.g. classification or tracking. It is the largest memory unit within the object detection module, also implemented as a dual-port RAM. The size of `DATA` is determined by the maximum number of connected components (CC_{max}), and the image size ($R \times C$). The minimal implementation requires both horizontal and vertical coordinates for top-left and bottom-right pixels to be stored. This results in $(2{\cdot}\lceil log_2(R)\rceil{+}2{\cdot}\lceil log_2(C)\rceil){\cdot}(CC_{max})$ bits of data.

The extracted features are written into the `DATA` whenever the first background pixel is encountered after a series of foreground pixels ($\mathtt{E} = 0$ and $\mathtt{D} \neq 0$). The read operation however is issued every time the `BUF` changes and the $\mathtt{LOOKUP} \neq 0$. In order to reduce the number of read and write memory operations, the data used to calculate features of the current connected component is stored in temporary registers.

MERGER STACK

The `STACK` unit was developed as a Last-In First-Out (LIFO) memory element in order to store both (or more) labels to be merged. These mergers are resolved at the end of the line scan, in reverse order. The size of the `STACK` unit is relatively small, was determined during experimental simulations. Therefore, it can be implemented using distributed memory without affecting BRAM resources.

This memory module support two independent read and write control signals. They are asserted for a single clock cycle to perform read or write operation respectively. The write operation takes place during the line scan when label collision occurs, whereas the read operation is issued during the stack-based merging in horizontal blanking period only.

LABEL CONTROL

The `LABEL CONTROL` is one of the two major control units embedded within the detection module. It is responsible for label selection according to the classical labelling routine. The label being assigned to the register `E` is selected using data stored in registers `B`, `D` and data features calculated for these labels by the `MERGER & DATA CONTROL` unit. The labelling routine can be described with pseudo-code listed below:

```
if E = 0
    E ← 0
else
    if B = 0
        if D = 0
            E ← new_label
        else
            E ← D
        end
    else
        if D = 0
            E ← B
        else
            //merger
            if y_top(D) > y_top(B)
                //     1
                // 2   1
                // 2 2 x
                E ← B
                STACK ← push[D, B]
            elsif y_top(D) < y_top(B)
                // 1
                // 1   2
                // 1 1 x
                E ← D
                LOOKUP(B) ← D
            elsif (y_top(D) = y_top(B)) and (B ≠ D)
                // 1   2
                // 1   2
                // 1 1 x
                E ← B
                STACK ← push[D, B]
            else
                E ← D;
            end
        end
    end
end
```

Listing 5.1: Labelling routine pseudo-code

The label choice is straightforward until the label collision is appointed (see line 13). Due to the label-reuse approach, the label selection can not be solely based on the label numbers of adjacent pixels as it takes place in other algorithms. The comparison is based on coordinates calculated for top pixels from both labels. The following cases apply:

- (y_top(D) $>$ y_top(B))

  Lines 15 - 20 refer to the situation when the connected component left-adjacent to the current pixel was labelled after the top-adjacent one. In this scenario the label stored in the register `D` will not occur again in the current line scan, hence the register `E` is assigned with the label stored in `B`. Moreover, both labels are pushed onto the `MERGER STACK`. At the end of the current line scan all the labels within the `STACK` will be updated with most recent pointers to avoid situation depicted in Figure 5.14(a), where the pixel "?" is assigned with label 2, already merged with label 1, therefore not valid any more.

- (y_top(D) $<$ y_top(B))

  Here we take into consideration the opposite situation to the one discussed above. Lines 21 - 26 refer to the merger where label stored in `B` will not occur at the left side of the label stored in `D`, however it can be still be appointed in the `BUF` during the current line scan, as depicted in Figure 5.14(b). Therefore the immediate update of the `LOOKUP` is required.

- (y_top(D) = y_top(B) *and* B $\neq$ D)

  The code listed within lines 27 - 32 refers to the situation when both objects were appointed for the first time during the same line scan. This case has to be taken into consideration since the label stored in `D` could be involved in one of the other mergers within the current line, as depicted in Figure 5.14(a). Hence, the stack-based `LOOKUP` update.

- In all other cases the register `E` will be assigned with the last known label stored in the temporary register.

The `LABEL CONTROL` unit was implemented according to the label-reuse technique. The label-reuse approach is based on the First-In First-Out (FIFO) storage element where redundant labels are collected for further use. This FIFO operates synchronously with the labelling routine and the size is determined by the image width C. In the worst case scenario the binary image requires enough storage for $C/4$ labels, however according to simulation results, typical binary image does not exceed $C/10$ mergers per

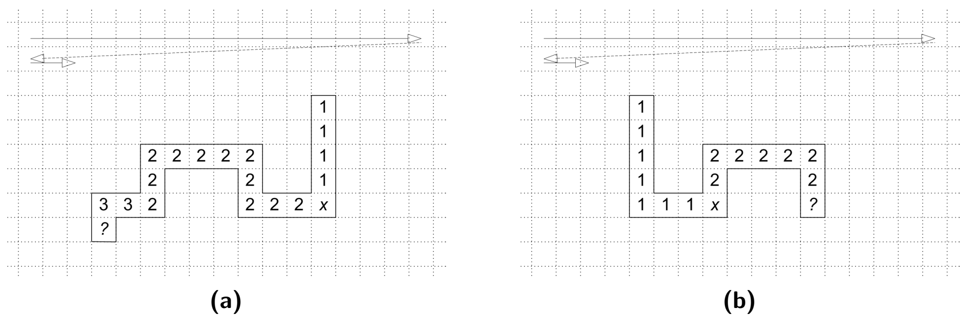


Figure 5.14: `LOOKUP` and `STACK` based merging. (a) The left-adjacent label could be already merged in the line scan therefore the `STACK`-based merging is required. (b) The top-adjacent label can still occur in the `BUF` during the line scan hence the immediate `LOOKUP` update.

line therefore the FIFO is relatively small and can be implemented in distributed logic. The label-reuse technique can be described with the pseudo-code listed below:

```
//collect label
if E = 1
    if B = 1 and D = 1
        //merger
        if y_top(D) > y_top(B)
            LAB_FIFO ← [y, D]
        elsif y_top(D) < y_top(B)
            LAB_FIFO ← [y, B]
        elsif (y_top(D) = y_top(B)) and (B ≠ D)
            LAB_FIFO ← [y, D]
        end
    end

//retrieve label
else
    if B = 0 and D = 0
        if (LAB_FIFO_empty = false) and (LAB_FIFO(1) < y−1)
            E ← LAB_FIFO(2)
        else
            E ← new_label
        end
    end
end
```

Listing 5.2: Label-reuse technique pseudo-code

As can be seen in the Listing 5.2, every time the merger occurs, one of the labels together with the current line number are written into the `LAB_FIFO`. The line number stored in the FIFO is used as a control signal - once a new pixel is appointed, it is compared with the current line number. If the difference is greater than 1, the label stored within the FIFO can be used. Since the line number stored within the `LAB_FIFO` is accessed every time the new pixel is appointed, the First-Word Fall-Through (FWFT) FIFO support is required [121]. The FWFT provides the low-latency access to data thanks to the ability to look ahead to the next word available from the FIFO without having to issue a read operation.

MERGER & DATA CONTROL

The `MERGER & DATA CONTROL` unit is responsible for managing mergers and extracting features of interest, as well as for the data administration. It also manages control signals for all the memory blocks within the module.

The stack-based merging was implemented to manage the chain of multiple cross mergers within the single line scan. The process of merging starts at the end of the active line scan and is performed according to the pseudo-code presented in Listing 5.3.

```
while STACK_empty = false
    if merging_state = "init"
        STACK_re ← 1;
        index ← STACK(1)
        LOOKUP_addr_B ← STACK(2)
        merging_state ← "merger"
    elsif merging_state = "merger"
        STACK_re ← 1
        LOOKUP_addr_A ← index
        LOOKUP_A ← LOOKUP_B
        index ← STACK(1)
        LOOKUP_addr_B ← STACK(2)
        merging_state ← "merger"
    end
else
    STACK_re ← 0
    merging_state ← "init"
end
```

Listing 5.3: Stack-based merging pseudo-code

The stack-based merging was implemented as a Finite State Machine (FSM) and involves the following operations:

(a) read data from the `STACK` (both labels to be merged),

(b) set the second label as the `LOOKUP` address in order to read its pointer,

(c) set the first label as the `LOOKUP` address and rewrite this memory location with the pointer of the second label,

(d) disable the `STACK` read operation, finish merging.

The series of read and write memory operations requires multiple clock cycles however with efficient implementation it can be pipelined with a throughput of one merger per clock cycle. This can be achieved only with the dual-port architecture of the `LOOKUP` where the port B is constantly used for read and the port A for write. Moreover, it ought to be synthesized in the write-first mode (Read After Write). This will further reduce the system latency. It is useful when a chain of labels needs to be merged, e.g. $1 \rightarrow 2$, $2(1) \rightarrow 3$, $3(1) \rightarrow 4$, etc. The `STACK` also supports the FWFT feature therefore the stack-based merging starts immediately at the end of the active line scan.

The `DATA CONTROL` unit can be customized during the synthesis to extract a number of different features for connected components during the image scan. They are listed below:

- Position, object height and width
  The minimal implementation of the single pass algorithm supports extraction of the top-left and bottom-right corners for the bounding box for each object. With this data its position as well as the width and height can be easily calculated.

- Object counter

  Gives the number of detected connected components. It is implemented with a single comparator to check at the end of the image scan whether the data stored in the `DATA` is valid or not for every memory location. The sum of positive comparisons gives a number of objects in the current frame.

- Size

  The size gives a number of foreground pixels $p(x, y) = F_f$ within the connected component. It is calculated for each object and is implemented with an additional entry in the `DATA`, incremented every time the $p(x, y) = F_f$ is encountered. Once two labels are merged, their sums are added together.

- CoG

  The implementation of the CoG extraction is more complex. Both moments M_{01} and M_{10} are calculated in the same manner as size (moment M_{00}, see Equation (5.2)). In order to calculate CoG coordinates, a division is required. For this task, the Pipelined Divider IP Core from Xilinx Core Generator [122] was employed. The implementation of the CoG extraction causes little increase in hardware complexity and requires relatively large amount of memory.

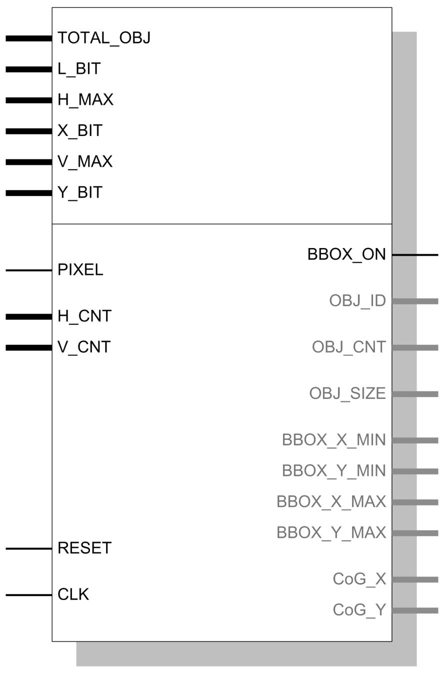


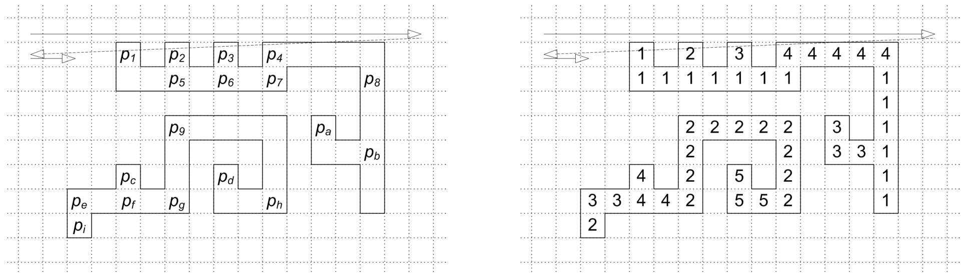
Figure 5.15: Object detection module block

5.4.4 Module Instantiation

The object detection with feature extraction based on the single pass CCL algorithm was developed and implemented as a fully customizable generic HDL module. Thanks to the modular architecture it can be easily adopted by other object detection or pattern recognition systems. It was developed using behavioural HDL inference templates therefore it is semi-device independent and can be used as a hardware accelerator for a wide variety of FPGA-based processing systems.

The object detection module is depicted in Figure 5.15, where vectors on the top of the diagram refer to the generic variables defining the total number of objects, the binary width of this number and horizontal as well as vertical size of the image frame. Signals depicted at the left side refer to input signals as follows: binary pixel data, horizontal and vertical counters (specify position within the image frame), reset and clock. The detection module operates synchronously with the input data controller therefore the clock signal operates on the video source pixel clock frequency.

The object detection module can be instantiated in a minimal form (signals written in black) with the single bit bounding box control signal which asserted whenever the current pixel location belongs to the rectangle box bounding one of the connected components. Such implementation simplifies debug and allows for visual verification, with no external logic analyser required. The `OBJ_ID` output vector shall be used to indicate the current object.



(a) **(b)**

Figure 5.16: Binary image frame with multiple mergers. (a) Image frame with points of interest. (b) Image frame with label assignment.

5.4.5 Results and Performance

The CCL module was implemented in VHDL using Xilinx ISE (Integrated Software Environment) WebPACK Design Software [96] and simulated in Mentor Graphics ModelSim simulator [97]. The final implementation was tested using both Spartan-3A [25] and Virtex-II Pro [26] Xilinx FPGA devices.

The object detection unit was verified and tested with complex images containing multiple cross mergers in order to fulfil the "worst-case-scanario" criteria. An example image is depicted in Figure 5.16(a), whereas the Figure 5.16(b) gives an overview of how labels are assigned during the image scan.

Since the image is scanned in raster order, all the pixels p_1, p_2, p_3 and p_4 will be assigned with unique labels. During the consecutive line scan, multiple label collision occurs when pixels p_5, p_6 and p_7 are appointed. While merging, the `LOOKUP` is immediately updated for labels assigned to pixels p_2, p_3 and p_4. Hence, when the pixel p_8 is appointed, the `LOOKUP` entry for the top-adjacent pixel refers to the label assigned to pixel p_1. Moreover, all the labels that were overwritten in the `LOOKUP` while merged, they will be accumulated in the `LAB_FIFO` so they could be further reused. Once the pixel p_9 is appointed, the first label will be taken from the `LAB_FIFO`, the second one will be assigned to the pixel p_A. Once the pixel p_B is appointed, another label collision takes place, however this time both labels are pushed onto the `MERGER STACK` and the merger will be resolved at the end of the line scan. When the last active pixel in the line scan is reached, the system starts stack-based merging routine. The `LOOKUP` for label assigned to pixel p_A is updated with the one from pixel p_8. The redundant label will be also accumulated by the `LAB_FIFO`. Once the `STACK` is empty, the system returns to the scanning routine. A new pixel is appointed at p_C, it will be assigned with the first label available at the output of the `LAB_FIFO`. The pixel p_D however will be labelled with the incremented global label number, even the `LAB_FIFO` is not empty. This is caused

Table 5.6: Video sequence analysis, resolution 320×240 pixels

	Sequence 1	Sequence 2	Sequence 3
Max no. of objects per frame	9	6	8
Mean no. of objects per frame	4.3	1.8	2.7
No. of image frames in a sequence	1800	1800	1824

by the fact the label collision at p_B took place in the line directly above, therefore it does not comply with the condition stated in the 17th line of the Listing 5.2. However, it can be used in the consecutive line. At pixel p_F both adjacent labels are pushed onto the `MERGER STACK` and the system keeps scanning with the older label. Similar situation happens at pixels p_G and p_H. At the end of the line scan, the `STACK` contains the following pairs: (p_E, p_C), (p_C, p_9), (p_D, p_9). These mergers need to be resolved in the reverse order they were accumulated. The first pair (p_D, p_9) is processed according to the routine previously described for labels (p_A, p_8), it also applies for the second pair of labels (p_C, p_9). The last merger can be also performed in a single clock cycle thanks to the fact the `LOOKUP` was synthesized in the write-first mode. The `LOOKUP` entry for the label assigned to the pixel p_E is overwritten with the one written in the previous clock cycle for label assigned to pixel p_C. This labelling routine allows the assignment of the correct label when the pixel p_I is encountered.

Resource Utilisation

Based on the video analysis for three different scenarios, the maximum as well as the mean number of connected components per video frame was determined (see Table 5.6). The peak number of objects for PAL video signals did not exceed 10, however this refers to the simulation with the system properly calibrated (low volume of noise). The mean value oscillated around 2-4 objects per frame.

All the generic parameters of the object detection unit can be set with suitable values thanks to the data summarized in Table 5.6. Since the number of connected components per image frame does not exceed 255, the `TOTAL_OBJ` number is set to 255 (with `L_BIT` = 8). The maximum resolution of the video source is currently limited by the VDEC1 Video Decoder Board [67] with ADV7183B Video Decoder chip from Analog Devices. It supports NTSC, PAL and SECAM input signals therefore after the video source adjustment (cropped active video area), system parameters are set to 320, 9 , 240, 8 for `H_MAX`, `X_BIT`, `V_MAX` and `Y_BIT` respectively.

For the purpose of resource utilisation analysis, the CCL module was synthesized as a top-level file. It was found the size of the `MERGER STACK` has critical impact on the overall system resource utilisation. The synthesis results summarized in the Table 5.7

Table 5.7: Implementation and resource utilisation for the XC2VP30 FPGA implementation

Resource	in use	total	[%]
Slice Flip Flops	2321	27,392	8
4 input LUTs	2209	27,392	8
Occupied Slices	2081	13,696	15
Block RAMs	3	136	2
Resolution [pixels]		320×240	
Frame rate [fps]		60	
Speed [MHz]		25	
No of CC_{max}		255	

were collected for the system with 8-bit width `STACK` address bus, whereas a change into 6-bit address width significantly reduces demand for system resources (4%, 4% and 8% for Slice Flip Flops, 4 input LUTs and Occupied Slices respectively). The instantiation of the stack-based merging circuit results in a large combinational logic circuit with 16-bit 256-to-1 (or 64-to-1 for 6-bit address width) multiplexer for the data read signal. The `STACK` is synthesized as 4096-bit (1024-bit respectively) register.

The system was tested with a wide variety of test images, also with real-time video signals. Results were verified with on-screen data as well as using external logic analysers. The system is capable of processing video streams without the need for buffering input data. This significantly reduces memory requirement. The synthesis of such a system architecture results in lower resource utilisation comparing with other implementations previously mentioned in this chapter.

Optimization

The labelling module described in this chapter is capable of processing real-time video streams with a high throughput at the video source pixel clock frequency. Therefore, since the video source is provided on the pixel basis, all the data features can be extracted and forwarded for further processing immediately after the image scan is completed. With the increasing resolution of the video signal, the system would require more memory in order to store extracted features of interest, however even for high-resolution images this can be handled in a single FPGA using embedded BRAM.

In order to further reduce memory consumption, the ”memory-reuse” approach should be considered. The principle is analogical to the label-reuse approach described in previous subsections. Once the component is labelled completely, extracted features stored in the `DATA` memory block should be transmitted to the external system for further processing. The table entry could be re-used during the image scan, therefore both the

BUF and the LOOKUP sizes could be reduced (smaller number of labels). This approach could be highly beneficial for systems with a high volume of acquisition noise.

5.5 Conclusions

In this chapter an overview of connected component labelling algorithms was given, a fundamental feature of many computer vision systems. Following the literature review, a discussion on memory requirement and resource utilization for hardware implementation was provided. This was followed with implementation details of the algorithm based on the single pass approach.

The literature review consists of algorithms grouped as follows:

- processing an image in two consecutive passes through the frame,
- multiple scans, with the number of scans depending on the image complexity,
- parallel algorithms, processing a number of pixels at a time,
- contour tracing techniques, following the contour of the object,
- single pass algorithms processing data sequentially in one scan through the image.

An overview for each group was provided together with a discussion on suitability for hardware implementation within a real-time video processing system. Therefore, aspects such as processing time, memory consumption and resource utilization were taken into consideration. The label collision problem caused by the "U" shaped objects was introduced and a discussion on how it affects the overall system performance was provided.

Based on the discussion provided, algorithms from the following groups were considered for the implementation: two passes, contour tracing and single pass. In comparison, both the classical approach (two passes) and the contour tracing algorithms share similar features. The contour tracing algorithm requires less memory resources, whereas the classical approach can guarantee constant processing time in two image scans. However, the single pass approach outperforms both the previously mentioned algorithms in all the compared aspects. Thanks to the advantageous architecture, it does not require an auxiliary storage for neither input image nor output labelled frame, being the main drawback for other algorithms. Moreover, it is capable of processing input data simultaneously with the image acquisition, providing extracted features immediately at the end of the image scan. Therefore, it is at least twice faster compared to other algorithms. Thanks to the shortest processing time and the lowest memory utilisation, the single pass algorithm was chosen for the final implementation.

The second part of the chapter gives implementation details of the algorithm based on the single pass approach with unique memory management and label-reuse technique applied. This constitutes part of the contribution since known implementations struggle with memory resources. An overview of the system architecture was provided together with detailed description of the system components. The architecture of the algorithm described is based on the 4-neighbourhood connectivity $N_4(p)$ mask. This was chosen over the 8-neighbourhood connectivity $N_8(p)$ mask due to lower complexity while providing the same set of features.

For the purpose of this implementation, the novel label-reuse approach was developed. It allows labels previously assigned and merged during the image scan to be used for new objects, once the labelling and merger of the current connected component is finished. This approach is particularly beneficial for video streams, where multiple label collisions occur. Thanks to improved memory management, the majority of memory entries can be reused.

The system implementation was developed in VHDL as a customizable generic HDL module. Thanks to the modular architecture and the use of behavioural HDL inference templates for embedded memory modules, the system is semi-device independent, therefore it can be applied within other object detection or pattern recognition systems.

The final implementation was verified and tested on both Spartan-3A and Virtex-II Pro Xilinx FPGAs. It was tested using still images stored in ROM to emulate the worst-case-scenario as well as on live data provided by the video stream.

CHAPTER 6

System Integration

This chapter gives an overview of the IR pedestrian detection system implemented in VHDL and targeted on XUP V2P FPGA development board. It provides an overview of the system components, gives details on implementation and the overall system integration. The purpose of this chapter is to illustrate how system components described in these thesis, designed as hardware accelerators, can be instantiated in a real-time video processing platform. Moreover, an introduction into implementation of tracking algorithms on embedded processor core will be also provided as possible extension to the project for future work. The chapter will be concluded with experimental results collected from the system running in real-time, processing pre-recorded IR video sequences.

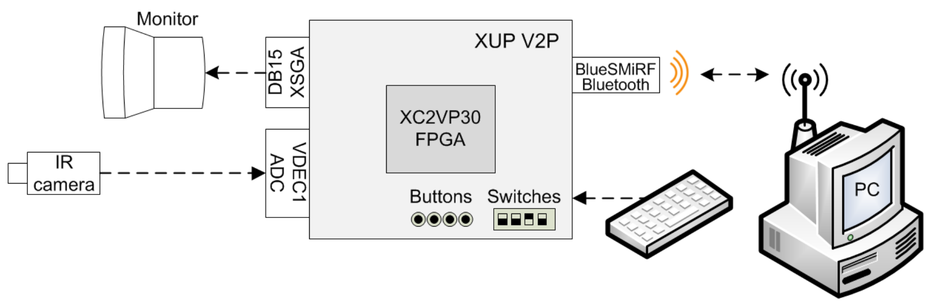


Figure 6.1: An overview of processing platform together with auxiliary system components.

6.1 Development Platform Overview

This section gives an overview of the processing platform and other system components used for the purpose of this project. HDL drivers for system components were implemented in VHDL, synthesised, built and verified on FPGA device from Xilinx. For this task Virtex-II Pro XC2VP30 FPGA chip was used [26], embedded within XUP V2P Development Board provided by Digilent Inc [28]. The board features a number of ports and auxiliary connectors, therefore various external devices can be interfaced. As an input video source FLIR Systems Thermacam PM595 IR camera was used [5]. Video signal from the camera (PAL) was digitized by VDEC1 Video Decoder Board from Digilent Inc [67]. For the purpose of control and configuration push buttons and slide switches were used, as well as standard PC keyboard equipped with PS/2 interface. Results of the processing become available on the monitor display for visual verification. Output data could be also also transmitted to the external device using BlueSMiRF Bluetooth device [123]. An overview of the processing platform together with system peripherals can seen in Figure 6.1.

6.1.1 XUP V2P Development Board

For the development purposes, XUP V2P Development Board from Digilent Inc. was used. The board features comprehensive collection of peripherals, as follows: XSGA video output; user LEDs, switches and push buttons; AC97 audio CODEC and stereo amplifier; 10/100 Ethernet PHY; RS-232, PS/2, Serial ATA and multi-gigabit transceiver ports; up to 2GB DDR SDRAM DIMM module; 5V tolerant expansion headers as well as high speed expansion port. The main advantage of the board is the FPGA chip Virtex-II Pro with fair amount of internal memory and two embedded processor cores.

6.1.2 FPGA Chip with Embedded PowerPC

The on-board FPGA device, Virtex-II Pro XC2VP30, features 30,816 Logic Cells, 18×18-bit multiplier blocks, two PowerPC processor cores and 2,448 Kbits of block RAM (136 blocks). Such amount of embedded memory is sufficient for implementation of video processing algorithms operating on lower resolution images (QVGA in 8-bit grey-scale). Thanks to PowerPC processor cores, embedded system development can be speeded up with the use of internal control registers through designated software tools.

6.1.3 Video Decoder

The VDEC1 Video Decoder Board from Digilent Inc. was used for the purpose of analogue to digital video conversion. It is based on ADV7183B Video Decoder chip from Analog Devices [124]. The chip features three 54MHz 10-bit ADCs and provides 12-bit output data at a rate of 27MHz. Such data is encoded to ITU-R BT.656 video format, which defines the colour space, number of samples and sampling format. Thanks to the custom interface, redundant data can be disposed simultaneously with data acquisition. Hence, output of the decoding unit can be immediately used for the processing.

6.1.4 System Peripherals

For the purpose of debug and visual verification, a monitor display was interfaced through the XSGA video port. A VGA controller was created according to VESA standards, providing live output at a rate of 60*fps*. A preview of the video source and intermediate processing is of key importance during the start-up calibration.

In order to transmit detection results to the PC, BlueSMiRF Gold Bluetooth module from SparkFun was interfaced via expansion connector. Thanks to supported standard UART interface, a bidirectional communication link was established to allow further configuration and control also remotely from the PC. A reliable link can be achieved for mid-range distances (up to 10 m), with the data transmission rate of 115,200 bps. Further details on link controller and a description of the successful system implementation with software-based control panel running on the PC can be found in [30].

Although Bluetooth link is reliable and provides additional configuration flexibility, for the purpose of laboratory experimentation it was replaced with slide switches and push buttons. However, for additional settings and more configuration options (limited number of on-board switches), an interface for standard PS/2 keyboard was provided.

6.1.5 IR Camera

The video processing system was tested with two different IR cameras: a ThermoVision Micron Infrared Camera featuring a 164×128 Indigo VOx uncooled microbolometer sensor array and a FLIR Systems Thermacam PM595 equipped with 320×240 uncooled microbolometer focal plane array. The temperature range of the ThermoVision Micron in the standard package is 0^0C to 40^0C, where the scene temperature may reach up to 150^0C. This camera delivers performance and features typically found in larger and more expensive infrared systems. The second camera provided by FLIR Systems supports temperature range of -40^0C up to 500^0C. Thanks to higher resolution and greater robustness, as well as the ability run on the battery (portable), the second camera was selected as the preferred choice.

6.2 Video Data Acquisition from IR Camera

As it was mentioned in Section 6.1, VDEC1 video decoder board was used for the purpose of IR video acquisition. The ADV7185 ADC chip encodes input video signal from IR camera (PAL) into ITU-R.656 video standard, defined in ITU-R BT.601 and SMPTE125M[125]. These standards define Timing Reference Signals (TRS) used to determine image data and synchronisation signals such as video field and line timing [126]. This subsection gives an overview of video conversion and details custom timing model with control signals extracted from TRS, further used for synchronisation purposes.

6.2.1 Analogue to Digital Video Conversion

Although IR camera FLIR Systems Thermacam PM595 used for the purpose of this project features 320×240 uncooled microbolometer focal plane array, the output signal, interpolated by the internal circuitry, is transmitted in PAL video format [5]. Hence, it provides 625 lines of information in interlaced mode at 50Hz refresh rate [127]. Horizontal scan line description for composite PAL can be seen in Figure 6.2.

Full ADC conversion is performed by ADV7185 chip from Analog Devices providing 10-bits of output active data coded in ITU-R BT.656 standard at a 27MHz pixel clock rate. Based on 8 MSBs, TRS signals can be extracted since neither FFh nor 00h values occur in the regular video stream, as it was depicted in Figure 6.3.

In order to decode timing information, the following sequence of data must be encountered: FF 00 00 XY, where XY gives timing reference definition and corresponds to the following data: XY = 1,F,V,H,P3,P2,P1,P0. Timing signals F, V and H can

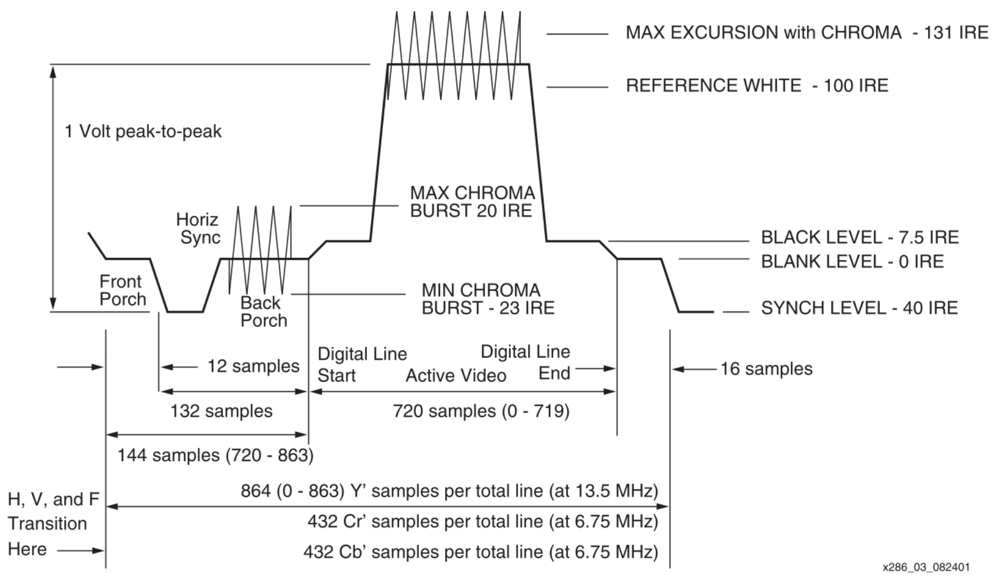


Figure 6.2: Horizontal scan line for composite PAL [126].

Table 6.1: Vertical timing reference in ITU-R BT.656 standard, where EAV/SAV are End/Start of Active Video respectively, XY = 1,F,V,H,P3,P2,P1,P0 and P3-P0 are protection bits.

Line #	F	V	H(EAV)	H(SAV)	Notes
1-22	0	1	1	0	Blanking
23-310	0	0	1	0	Field 1 (Odd) Active Video
311-312	0	1	1	0	Blanking
313-335	1	1	1	0	Blanking
336-623	1	0	1	0	Field 2 (Even) Active Video
624-625	1	1	1	0	Blanking

be decoded according to the Table 6.1, where P3, P2, P1 and P0 are protection bits used for corruption control and EAV with SAV refer to End or Start of Active Video respectively.

Further details on video decoding can be found in [126].

6.2.2 PAL to Progressive Scan Conversion

Modern video processors operate in progressive (raster) scan as opposed to interlaced scanning common in PAL or NTSC formats. Detailed information regarding progressive scan can be found in Section 2.2.1.

In order to extract odd and even fields from interlaced video stream, F timing reference signal from ITU-R BT.656 formatted data stream shall be used. It indicates whether the scan is odd or even according to the Table 6.1. Such signal can be aligned with or incorporated within vertical line counter of the input buffer controller.

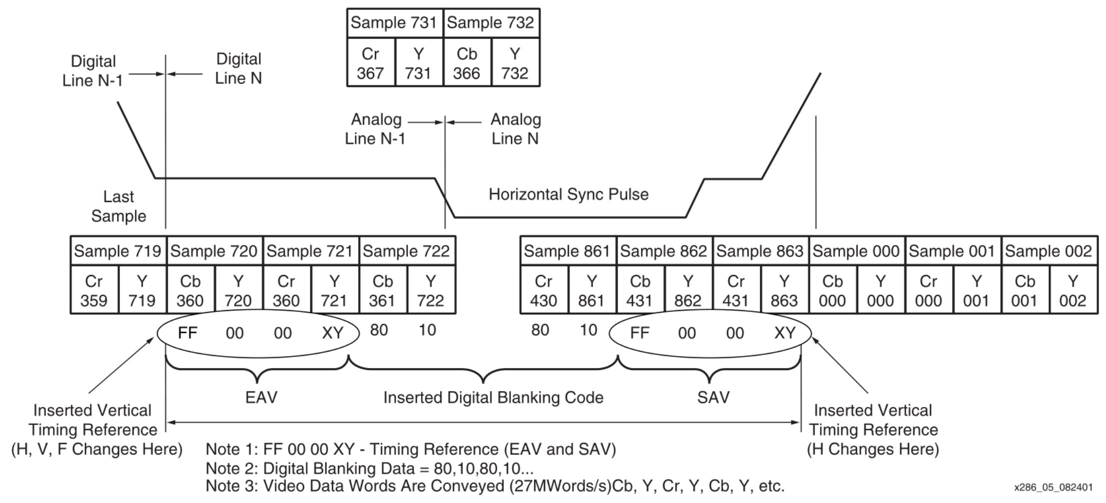


Figure 6.3: Horizontal scan line codes in ITU-R BT.656 standard [126].

Since the IR camera used for the purpose of this project is equipped with 320 × 240 uncooled microbolometer focal plane array, the output signal from the camera is subsampled to match native resolution of the sensor array. Hence, the deinterlacing function is simplified to data acquisition of odd frames at a rate of 25*fps* with 20ms time space between two consecutive image frames.

6.2.3 Active Data Extraction

For the purpose of video analysis, the amount of data extracted from the raw input signal is substantially reduced by removing not relevant components from the YCbCr colour space. As it was described in Chapter 3, the information relevant for further analysis is carried by the luminance Y component. The colour space in ITU-R.656 format is coded as follows: Cb, Y, Cr, Y, according to the Figure 6.3.

In order to match decoded data within the system framework, a custom video decoding wrapper module was created. The purpose was to align input data with the corresponding pixel location as well as to provide synchronisation signals for further processing. For the convenience of data analysis, input data traverse was matched with raster scan - horizontally from left to right, line by line, from top to bottom. Video acquisition module generates Horizontal Reference (`HREF`) and Vertical Synchronization (`VSYNC`) signals for synchronization purposes according to the Figure 6.4.

The data is synchronized with the 27MHz Pixel Clock (`PCLK`) provided by the video acquisition module. Active data is clocked on `HREF` set high, whereas `VSYNC` pulses indicate the end of active image frame, as it can be seen in Figure 6.5.

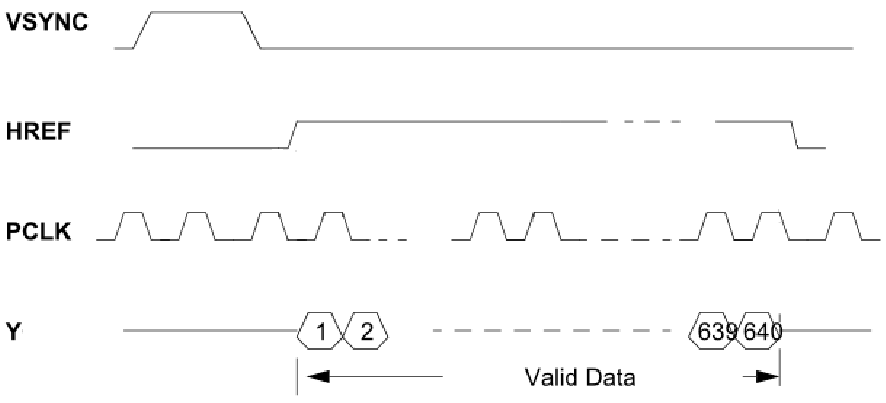


Figure 6.4: Horizontal timing diagram of the video acquisition controller.

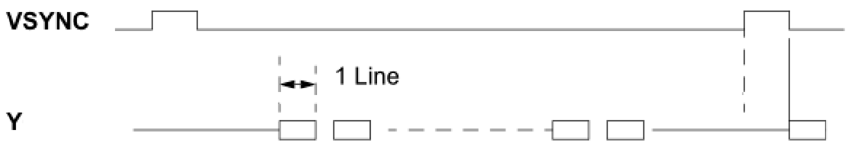


Figure 6.5: Vertical timing diagram of the video acquisition controller.

6.3 System Architecture

This section gives an overview of the system architecture designed and developed for the purpose of this thesis. It illustrates how system components described in these thesis, designed as hardware accelerators, can be instantiated within a self-contained system. The data flow is described at every stage of the processing to reveal all the instantiation subtleties. In further subsections an initial approach to the implementation of tracking on PowerPC embedded processor core will be also described. This will be followed by the summary of synthesis results and discussion on available system resources for further expansion.

6.3.1 Design Goals

The aim was to create a platform for pedestrian detection from IR video stream, which could be used as a self-contained processing system. The system should be capable of further expansion. Moreover, individual subsystem blocks portable, to be used as subsystems within other processing systems. Hence, a modular approach was investigated for hardware accelerators, supported by their own control logic for synchronization purposes.

At an initial stage of development, a conceptual model of the system was created using block diagram to describe data flow. This was followed by a software development to verify and test concepts and algorithms. For this task Matlab was used. The final step was implementation in RTL to speed up the processing time by accelerating repetitive routines and enabling parallel processing. Prior to HW implementation, a distinct separation between HW and SW workload was made to partition implementation for logic primitives and embedded processor core.

Designing a digital architecture is not a trivial task. It requires a significant amount of planning in order to maintain data stream from IR camera in a real-time with minimum latency. A number of design issues was encountered such as clock domain crossing, synchronizing ADC data with raster scan grid and managing memory bottleneck while transferring data between processing units.

Pedestrian detection system for IR video streams, being the subject of these thesis, was designed, developed and implemented in multiple stages. An overview of the top-level system hierarchy can be seen in Figure 6.6.

6.3.2 Data Flow

As it can be seen on the block diagram depicted in Figure 6.6, the data flow of the system starts form the IR camera. Once `Video Acquisition ADC` module becomes enabled, image pixels drive `Active Data Extraction` block through the 10-bit data `YCrCb` vector synchronously with the rising edge of 27MHz pixel clock provided by the VDEC1 board. Within the synchronisation block, ITU-R BT.656 signal is decoded according to Figures 6.2 and 6.3, active data is extracted, de-interlaced and aligned with the raster scan grid.

Extracted data is provided to both `Video Buffer` as well as `Segmentation` unit at a rate of 27MHz. `Video Buffer` is a Dual-Port Block RAM and is used to cross the acquisition pixel clock domain with the system clock running at 25MHz, driven by `Video Output Controller` for monitor display. The memory address bus is 17-bits wide, whereas grey-scale information is stored with 4-bit depth.

While data within `Video Buffer` is available instantly for `Video Output Controller`, matching data provided to the `Segmentation` unit during the first image scan is used to build a background reference image, according to the description provided in Chapter 3. To reduce the latency in background segmentation, input data is matched with the corresponding pixel from the background model, fetched one clock cycle in advance. Moreover, while written to the memory, image data is simultaneously thresholded and

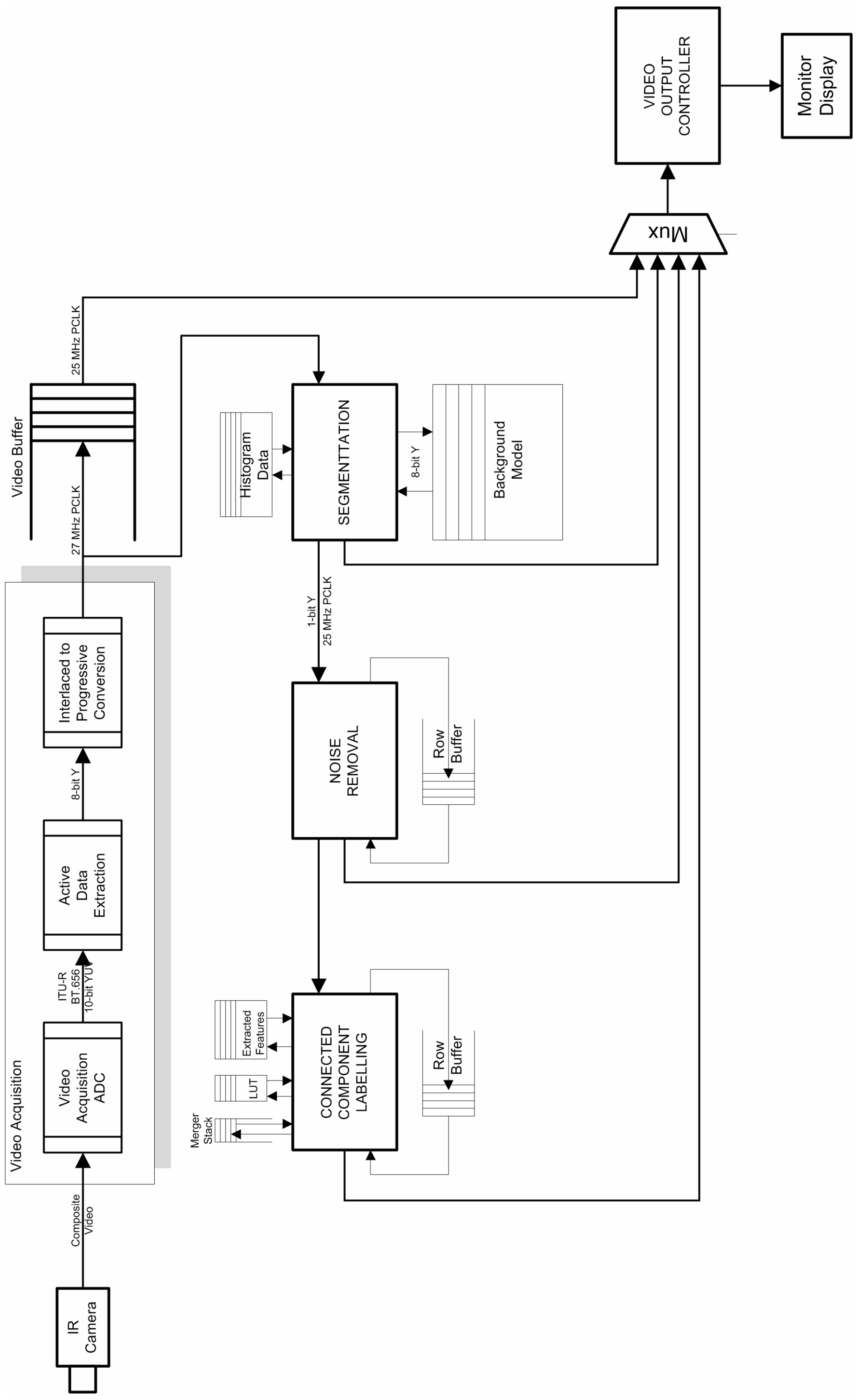


Figure 6.6: IR Pedestrian detection system architecture. Top-level hierarchy block diagram.

provided for further processing. The storage element is a true Dual-Port BRAM with 8-bit data width and 17-bits address bus. Background adaptation is performed synchronously with 27MHz ADC pixel clock, whereas output data provided to `Noise Removal` is clocked at 25MHz.

Since background segmentation is instantaneous, thresholded data becomes available to the `Noise Removal` in the same clock cycle, with only gate propagation delay (can be omitted). It is immediately used for the filtering, according to the specification provided in Chapter 4. Due to the nature of morphological filtering, the output of this processing step is transmitted further with constant latency, constrained by the size of the Structuring Element SE. In this particular implementation, where `SE_SIZE_X` = 5 and `SE_SIZE_Y` = 3, it is delayed by the time of 2 line scans and 4 additional pixel clocks, to accommodate horizontal and vertical padding.

The `Connected Component Labelling` block described in Chapter 5, takes advantage of the VGA image scanning technique by performing various merging tasks during horizontal blanking periods. This however does not affect the overall system performance, therefore results of the labelling and feature extraction are available when the last pixel of the image scan is reached. They can be accessed immediately during the consecutive line scan, one label per clock cycle.

At the end of the image scan, an update of the adaptive thresholding value is performed based on the data stored in the `Histogram Data` unit. The histogram data is read synchronously with the image scan, updated on the last value reached. Simultaneously, extracted features are read from the `Extracted Features` table and written to a set of output registers, also used by the VGA controller to draw bounding boxes around pedestrians. Since VGA mode horizontal line scan takes 800 clock cycles including display time, pulse width as well as front and back porches, it shall be considered as a limit of labels supported by the display module for current system configuration.

6.3.3 Summary

The data flow chain is driven by image samples provided by the ADC. It is aligned by the synchronization control logic during intermediate processing. This is a relatively flexible approach, prone to further expansion. Additional HW accelerator blocks can be embedded within the design at any stage of the processing. Moreover, it benefits from fixed memory structure - no need for intermediate FIFOs, which is advantageous in terms of limited memory resources as well as the overall system complexity.

The overall system latency is fixed to one image frame. Delays caused by data registers on individual processing blocks are marginal. The main bottleneck of the system is the `Noise Removal` unit, holding the data queue. It is caused by the row buffer, required

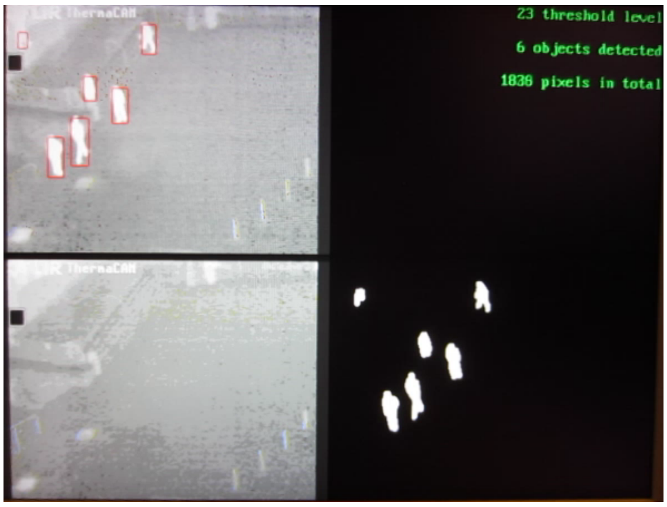


Figure 6.7: Multi-window view on the monitor output.

to support Structuring Element in morphological processing. The overall number of additional line scans is determined by the sum of $\lceil \texttt{SE\_Y}/2 \rceil$ for both erosion and dilation.

Image pixels are provided by ADC to the `Video Buffer` at a rate of 27MHz and are grid aligned. They are immediately available for VGA controller to be plotted on the monitor display. The acquired data is accessed via the second port of the Block RAM. This port is synchronized with local pixel clock running at 25MHz. While displayed on the monitor, the data is processed in the pipelined system, where intermediate results (background model, results of the subtraction and morphology filtering) can be also accessed for visual verification using multi-window view as depicted in Figure 6.7. Extracted features are available for display at the end of image scan. Based on such data, rectangular bounding boxes for pedestrians are generated and overlaid on the monitor display to enhance visual verification.

Although every processing unit was individually verified and tested in the simulator, visual verification using the live system benefits in various aspects. Video systems are difficult to debug due to large amount of data required for testing and high volume of output vectors. With the system implemented and running on FPGA board, a number of issues was encountered, not spotted in the simulator previously. Moreover, using multi-window display mode, the system can be properly calibrated.

Table 6.2: Detailed synthesis report part 1.

	ADC Pal Decoder	Video Buffer	Segmentation	Morphology	Labelling
Slices	66	25	184	128	2081
Flip Flops	83	45	87	28	2321
4 input LUTs	111	57	343	226	2209
IOBs	40	3	76	43	49
BRAMs	21	32	73	2	3
GCLKs	1	2	3	1	2

6.3.4 Tracking on PowerPC

Although implementation of tracking algorithms using embedded processor core is a natural extension for this research, is was not in the scope of these thesis. An initial approach to tracking was developed as an honours project by Alexander Balazs under guidance and supervision of the author. The project was concluded with the conference paper [128].

An introduction into tracking together with brief description of the system implementation based on [129] can be found in Appendix A.1.

6.3.5 Synthesis and Resource Utilization

This subsection gives an overview of FPGA resource utilization. This particular system build was synthesized for Virtex-II Pro XC2VP30 Xilinx FPGA [26]. Since other implementations referenced in these thesis are targeted for different FPGA devices using variable synthesis tools, the comparison one-to-one is not feasible. Synthesis results provided in this subsection shall be used for basic comparison as well as to give an overview and scale of the final implementation. The data was collected with each block synthesized independently, it can be seen in Tables 6.2 and 6.3. These tables provide detailed information about individual blocks, grouped into the following categories: Slices, Flip Flops, 4 input LUTs, IOBs, BRAMs and GCLKs. The term ”Slice” is Xilinx specific and refers to an element containing two 4-input function generators, carry logic, arithmetic logic gates, wide function multiplexers and two storage elements. Each of the function generators can be configured as 4-input Look-Up Table (LUT), 16-bits of distributed RAM or a 16-bit variable-tap shift register [26]. The number of slices used gives an indication about the physical size of the block on the FPGA die. Detailed synthesis results can found in Appendix A.2.

Table 6.3: Detailed synthesis report part 2.

	Font Display	VGA Controller	Sum	Available	Resource Utilization [%]
Slices	301	45	2830	13696	21
Flip Flops	196	55	2815	27392	10
4 input LUTs	575	88	3609	27392	13
IOBs	55	28	-	556	-
BRAMs	3	0	134	136	99
GCLKs	1	2	-	16	-

Synthesis results for all the subsystems can be seen in Tables 6.2 and 6.3. All the modules were synthesized individually with the synthesis tool set to automatically choose most efficient way for resource allocation, at a module level. The Resources Utilization column in Table 6.3 refers to the sum of individual synthesis results. The Table 6.5 however provides synthesis results for the entire system. As it can be seen, at a cost of additional logic, synthesis tool reduced the number of BRAMs used in order to reduce the effort of Place and Root (PAR) tool further in the build flow chain. All the synthesis results presented in this subsection are given for the following configuration:

- `H_MAX` = 320
- `V_MAX` = 240
- `SRC_IM_BRAM_ADDR_WIDTH` = 17
- `SRC_IM_BRAM_DATA_WIDTH` = 4
- `BCKG_IM_BRAM_ADDR_WIDTH` = 17
- `BCKG_IM_BRAM_DATA_WIDTH` = 8
- `SE_SIZE_X` = 3
- `SE_SIZE_Y` = 5
- `TOTAL_LABELS` = 250
- `L_BIT` = 8
- `X_BIT` = 9
- `Y_BIT` = 8

This is a set of generic parameters specified within the top-level module to configure all the subsystems. With such configuration, an image source, background model and output video is set to resolution 320 × 240 pixels. Video source is buffered with 4 - bit grey-scale, whereas background model is handled in 8-bit depth. `SE_SIZE_X` and `SE_SIZE_Y` refer to the size of the structuring element for morphology noise removal. In this implementation, settings for both erosion and dilation are the same. `TOTAL_LABELS` parameter determines the maximum number of detected objects per image frame. The following parameters such as `L_BIT`, `X_BIT` and `Y_BIT` are used to determine the size of both FIFO and STACK units.

Table 6.4: Detailed synthesis report for chosen subsystems.

	Histogram	Stack	Fifo buffer
Slices	40	446	440
Flip Flops	54	530	529
4 input LUTs	51	325	322
IOBs	36	38	38
BRAMs	1	0	0
GCLKs	1	1	1

Table 6.5: Synthesis report for the entire system.

	Top Module	Available	Resource Utilization [%]
Slices	3389	13696	24
Flip Flops	3260	27392	11
4 input LUTs	4157	27392	15
IOBs	47	556	8
BRAMs	118	136	86
GCLKs	5	16	31

The Table 6.4 gives a breakdown of two modules: Segmentation and Labelling. Data presented in these tables gives indication about main contributors to the logic count. The histogram subsystem is small containing one Block RAM to store 128 9-bit words. However, if the synthesis tool is set arbitrary to synthesize using logic elements instead of RAM Blocks (as it takes place in the top-level system synthesis), the number of Slices utilization increases to 109, being over a half of the Segmentation module. The following two subsystems Stack and Fifo buffer are main contributors of the Labelling module. As can be seen, they both combined constitute almost a half of the overall resource utilization of the block. It is due to their generic VHDL instantiation to be semi portable and allow synthesis also using non-Xilinx specific tools.

In overall, the system implementation on Virtex-II Pro XC2VP30 FPGA from Xilinx occupies a quarter of the logic available on the chip. The embedded memory utilization ratio is high, at a level of 86%. This number is determined by the storage requirements from both Segmentation unit and Video Buffer. The synthesis breakdown for individual modules shows that Labelling module utilized most of the system resources, over 10 times more than Segmentation or Morphology. Font Display module, included in this summary, was added to the system to enhance visual verification by displaying on-screen features such as number of objects detected, their size or current threshold level. It was added for debug purposes and can be safely removed from the system.

6.4 Experimental Detection Results

This subsection gives details on testing and experimental results obtained from the system running in real-time. For this task a set of test IR video samples was captured using video camcorder [130] to allow repetitive tests running on the system in different configurations. The test set contains three video samples, each around 60 seconds long. All of them were taken during Scottish autumn, from the top of one of the university buildings. The FOV varies from being the main university entrance, a way out from the university area, or entrance to the computing centre. Videos were taken during academic term within lecture hours hence increased student activity can be noticed. Prior to each test, the system was calibrated with an empty background reference image and set with initial intensity value for thresholding. The output from the monitor display was captured at 30*fps* frame rate to allow frame-by-frame post processing analysis. For the purpose of detection and error rate analysis, the following features were taken into account:

- number of pedestrians,
- number of bounding boxes,
- number of merges,
- number of splits.

The term merge refers to two separate pedestrians with their body parts overlapping, hence detected as a single object. Split, however, refers to pedestrians partially occluded with not all the body parts contained within a single component, therefore detected as multiple separate objects. The table with all the experimental results collected can be found in Appendix A.3. For illustration purposes, results were plotted and can be seen in Figure 6.8. Videos for both test data and experimental results can be found in [70–75].

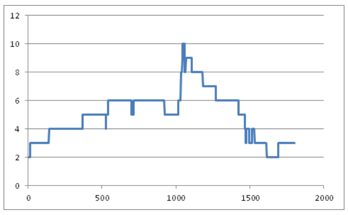
Based on the data collected, the mean value of the detection rate as well as associated error rate were calculated throughout the whole period of each video seqience and can be found in Table 6.6. The Detection Rate was calculated including splits as follows:

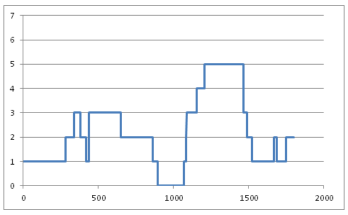
$$Detection\_Rate = \frac{boxes - splits}{pedestrians} \times 100\%. \tag{6.1}$$

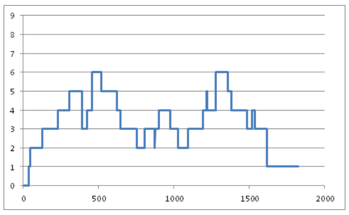
The Error Rate however was calculated as a ratio of bounding boxed around splits and merges to total number pedestrians:

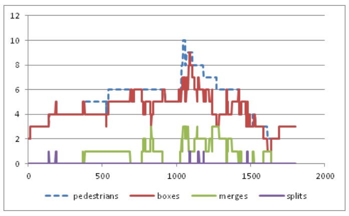
$$Error\_Rate = \frac{splits + merges}{pedestrians} \times 100\%. \tag{6.2}$$

The mean values of the Detection Rate as well as Error Rate for all the video sequences were calculated and can be seen in Table 6.6. Although the system was calibrated

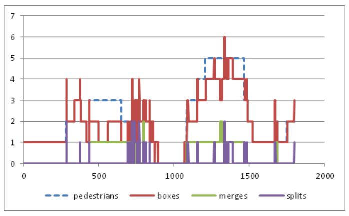

(a)


(b)

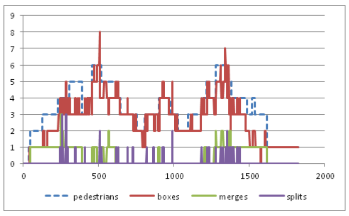

(c)



(d)



(e)



(f)

Figure 6.8: Results of the experimental pedestrian detection. (a, b, c) Video sequences 1, 2, 3 - pedestrians. (d, e, f) Video sequences 1, 2, 3 - bounding boxes, merges and splits.

Table 6.6: Experimental Detection and Error Rates, mean values over the simulation period.

	Sequence 1	Sequence 2	Sequence 3
Detection Rate [%]	87.9	87.1	79.6
Error Rate [%]	12.21	14.59	14.66

properly (no background object detected as pedestrians), relatively high value of Error Rate is caused by frequent merges. This is a consequence of increased activity within the FOV throughout the time of the simulation, see Figure 6.9a.

An increase in Error Rate can be also noted when pedestrians enter or leave FOV. This is caused by the fact they enter/leave FOV with the foot first/last, followed by the rest of the body, no associated within a single object at a time. This can be seen in Figure 6.9b.

Moreover, badly positioned IR camera was also causing an increase in Error Rate. The camera logo shown in the top left corner, as bright as pedestrians, was incorporated to the background model. Hence, pedestrians within this area were partially occluded causing multiple splits, as can be seen in Figure 6.9c.

An interesting case of merge was depicted in Figure 6.9d. This scenario involves two pedestrians walking close together without a single split while in FOV. A different approach for scene analysis is required to handle such situations. Such case was common in video sequence 3 therefore lower detection rate listed in the Table 6.6.

In summary, both detection and error rates obtained during experimental testing are satisfying, at acceptable level for basic surveillance applications. Since there is no unified

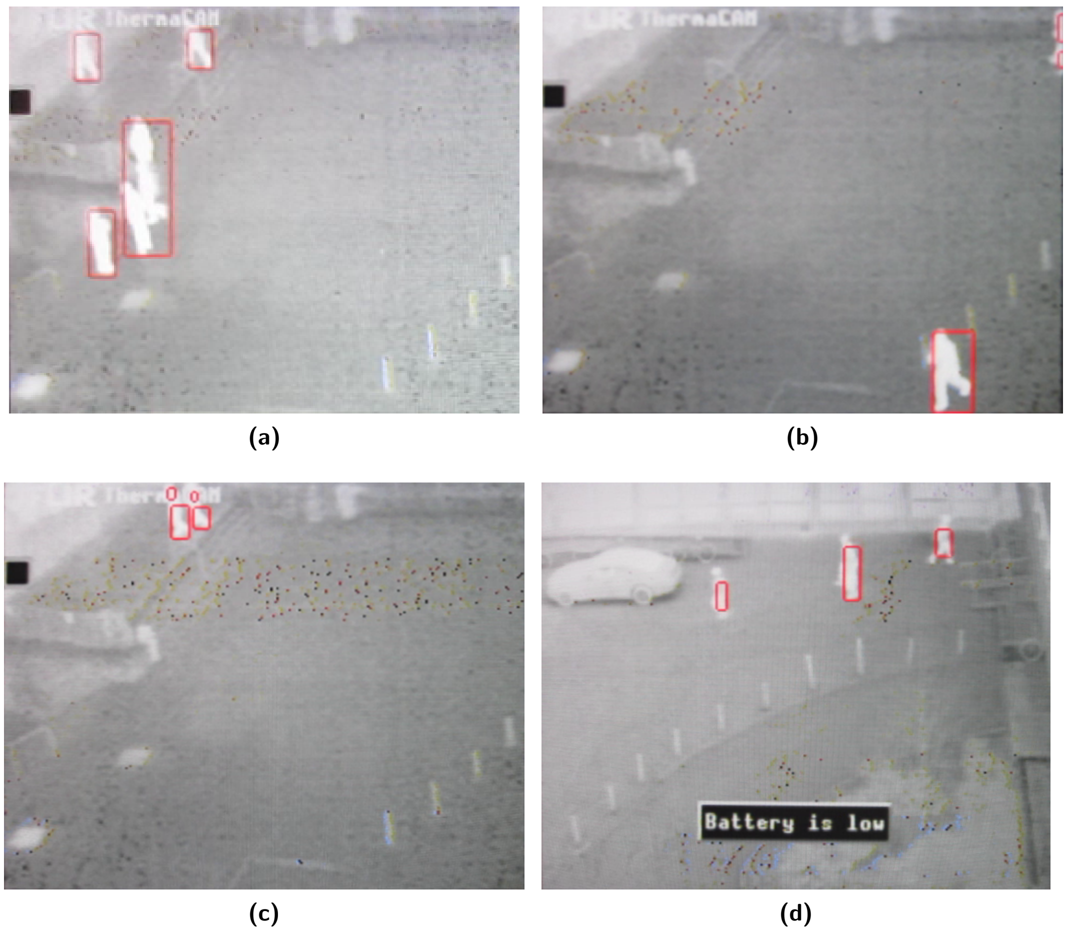


Figure 6.9: Samples from test video sequences. (a) Typical merge, the case when three pedestrians are detected and marked with a single bounding box as one object. (b) Pedestrians entering FOV often segmented and split due to not associated body parts. (c) Badly positioned camera when the manufacturer's logo is treated as background causing multiple short splits. (d) Continuous merge when two pedestrians walk next to each other therefore are detected as a single object.

test framework it is difficult to compare results with other implementation such as [131, 132]. They feature detection and error rates at a level of 88-95% and 1-2%respectively. These systems feature more sophisticated detection algorithms, moreover test video sequences do not contain as many complex and difficult for pedestrian detection scenes as used for the purpose of this research.

Although this system features satisfactory detection rates, it cannot be applied for specific applications such as people counters due to classification issues in handling merges and splits. This is a common issue in pedestrian detection. An implementation of prediction-based tracking algorithm could reduce the Error Rate, however it would not solve the problem entirely.

6.4.1 Performance Comparison

The system running on FPGA-based processing platform benefits from all the aspects of application specific hardware design, such as parallel processing, pipelined design approach and fully optimized architecture. The application of IR pedestrian detection being the subject of these thesis runs in a real-time providing results at a frame rate. This was limited by the ADC subsystem to 30*fps*, however the system is capable of running at pixel rate synchronously with VGA controller (60*fps*, running at 25MHz pixel clock). Based on the synthesis report, the system clock could potentially be running at a frequency of 124MHz, which gives up to 300*fps*. Individual modules such as segmentation, morphology or labelling can be synthesized with the following frequencies: 131, 355 and 147MHz respectively. Due to the pipeline, the system features latency described in detail in previous sections. The overall latency is less than a singe image scan.

The prototype of the system was created in software. The same algorithm was implemented in Matlab, without support from parallel processing toolbox. The code was not optimized to benefit from Matlab embedded functions, which could potentially increase processing speed. Although the system was running on 1.6 GHz Dual Core processor with 2GB RAM, steady data flow was not achieved. The limiting factor was the video source provided in PAL format. The video buffer was not big enough to handle such bandwidth, hence the system was crashing. The second attempt was made with pre-recorded video sequence stored on the hard drive of the computer. In this case a significant amount of computing power was allocated on video decoding. Similar to previous case, the video buffer could not handle such amount of data in continuous flow failing the simulation. Finally, the system was simulated and tested using a number of single PNG files extracted from the video stream.

Experimental results obtained from the real-time running system were compared with simulation results from the prototype. The system proved to be error free and working according to design requirements. Due to bandwidth limitation, the maximum processing rate achieved was up to 5*fps* when reading / writing source / results to PNG files. However, if there would be a need, this particular Matlab implementation could be further optimized or ported to C/C++ for further processing speed increase.

Due to the sequential nature of the image processing as well as video acquisition and operational memory bottlenecks, it is not physically possible to increase the processing speed on general purpose processor (GPP) to match corresponding implementation in HW. The application specific solution is capable of real-time processing with up to 60*fps*, benefiting from simultaneous processing on every stage of the data flow.

6.5 Conclusions

In this chapter, an overview of the IR pedestrian detection system implementation was given. The purpose of this chapter was to illustrate how system components described in these thesis, designed as hardware accelerators, can be instantiated in a real-time video processing platform.

The system was implemented in VHDL and targeted on XUP V2P FPGA development board. An overview of system components was provided, together with implementation details and system integration. A successful implementation was verified and tested with pre-recorded IR video sequences. After the synthesis at a subsystem level, an overview of resource utilization was provided for individual blocks to give a breakdown of the overall implementation. The chapter was concluded with a brief analysis of pedestrian detection rates obtained during experimental testing on a real-time running system. Moreover, an introduction into implementation of tracking algorithms on embedded processor core was provided as a possible extension to the project for future work.

For the purpose of system integration, XUP V2P Development Board with Virtex-II Pro XC2VP30 FPGA chip from Xilinx was used. This particular development platform was chosen thanks to fair amount of internal memory and embedded processor core within the FPGA chip, as well as wide variety of system peripherals and expansion connectors, appreciated during tests and verification. This overview was supported by detailed description of video conversion from PAL to Progressive Scan and custom timing model, created for the purpose of video acquisition. This included discussion on timing reference signals extraction, further used for synchronisation purposes.

Prior to implementation, a distinct separation between HW and SW workload was made to partition implementation for logic primitives and embedded processor core. A number of implementation issues was discussed, such as clock domain crossing, ADC data synchronization with raster scan grid and intricacies of memory management between processing units. An initial approach to implementation of tracking algorithm was undertaken as an honours project under the guidance and supervision of the author. Details regarding the implementation can be found in [129].

The data flow of the described implementation is driven by image samples, provided by the ADC. This approach is relatively flexible, prone to further expansion. Thanks to synchronization logic supported locally by each subsystem, an auxiliary IP can be embedded within the system, at any stage of the processing. Moreover, thanks to the fixed memory structure and data driven approach, there is no need for further FIFOs in order to instantiate the module. The overall system latency, although related to the size of scanning mask involved in morphological processing, will not exceed a single image scan. Detailed analysis of the overall system latency was provided.

The system after synthesis occupies 24% of entire logic available on the chip. The critical resource in system integration is embedded BRAM memory, occupied in 86%. Hence, the platform can be further used for auxiliary component or accelerator development, however, for memory intensive tasks an interface to either DDR SDRAM or external memory module shall be considered.

In terms of detection rates obtained during experimental testing, they were satisfactory for basic surveillance applications. Although the system features good segmentation, an increased Error Rate (at a level of 13% for tested sequences with a large number of pedestrians in FOV) is caused by classification issues in handling merges and splits. An implementation of prediction-based tracking algorithm could significantly reduce the Error Rate and further improve Detection Rate. However, for critical application, the system shall be considered only as a support for the human operator.

CHAPTER 7

Review of Research and Conclusions

The research reported in this thesis develops the state of the art in hardware architectures for infrared pedestrian detection. The implementation of pedestrian detection system reported in this thesis takes an advantage of infrared spectrum utilized by IR camera being the video source of the system. Both image processing as well as hardware implementation issues have been addressed in this thesis.

Pedestrian detection is an active research area, a number of implementation approaches have been reported in the literature. They are based on different algorithms, feature unique techniques for data handling and processing flow management. They meet the trade-off between the system performance and the quality of output data. Often, systems featuring higher detection rates are based on complex, memory intensive algorithms. The challenging aspect is the real-time processing constraint, where most of the algorithms implemented in software fail due to high data bandwidth imposed by the video source. A solution is an expansion to the application specific platforms for custom implementation at a lower abstraction level.

The architecture reported in this thesis is targeted for the FPGA device and utilizes all the aspects of custom implementation in hardware such as parallel or pipelined processing. Moreover, various data reduction techniques based on mathematical morphology were introduced to improve existing algorithms. The system is capable of processing video stream in real-time providing results for analysis or further processing at a frame rate. It was verified and tested as a stand-alone platform as well as at RTL subsystem level. The challenging aspect of this project, handling high data bandwidth with limited memory resources on embedded system, was achieved thanks to the unique combination of image processing and data reduction techniques described in preceding chapters. The final implementation is capable of processing IR video stream in real-time simultaneously with data acquisition using a single FPGA device with no need for auxiliary memory resources.

7.1 Review of Research

In this thesis an architecture and a successful implementation of IR pedestrian detection system is reported. The system is capable of processing video stream provided by infrared camera in a real-time. Results of the processing are available at a frame-rate, at the end each image scan.

For this task, an extensive review of algorithms for pedestrian detection was performed. This includes literature-review in multiple areas, such as infrared detectors, digital image processing and pedestrian recognition. Moreover, an introduction into hardware development and implementation was provided as means for high speed implementation.

The research is spread amongst various chapters referring to particular subsystem and the overall system integration. There are the following subsystems employed: background segmentation, morphological noise removal and connected component labelling. Moreover, a detailed description of system integration on FPGA development platform was provided to emphasize intricacies of real-time video processing system.

An introduction to background segmentation is provided together with deep analysis of system requirements. An architecture for adaptive background subtraction module is introduced. It is based on the selective running average background model, supported by adaptive thresholding. The architecture provides strong segmentation capabilities while being suitable for efficient hardware implementation.

In order to remove video acquisition noise, an implementation approach for morphology filtering was described. It is based on mathematical morphology and basic set theory definitions, commonly used in this field by other researchers. This particular implementation benefits from flat and decomposable scan mask. It takes an advantage of the duality and decomposition of erosion and dilation. Hence, the implementation is small and robust, suitable for embedded systems with limited resources.

For the purpose of pedestrian recognition, a single-pass approach for connected component labelling was employed. It is a relatively new processing technique suitable for streaming data video processing systems. As opposed to other algorithms presented in this thesis, this approach features constant processing time, extracted features are available immediately at the end of the image scan, and most of all, it requires significantly less memory compared to other techniques. The implementation benefits from horizontal blanking periods used by the algorithm to further optimize memory consumtion.

The system architecture was implemented using VHDL in semi-device independent approach thanks to behavioural HDL inference templates used for memory controllers. The initial implementation was targeted on Virtex-II Pro XC2VP30 FPGA chip. The system was verified and tested using pre-recorded IR video sequences. A final implementation occupies 24% of available logic resources and 86% of embedded memory. The system is running synchronously with 25MHz pixel clock. It is capable of processing video stream at a rate of 60 frames per second.

7.2 Conclusions

In this thesis, a novel architecture for self-contained IR pedestrian detection system was presented, being the main contribution. It is based on low-level algorithms customized for hardware implementation. This research has addressed issues of both infrared pedestrian detection as well as system integration capable of real-time video processing.

The main advantage of this particular architecture is the ability to process video stream in real-time utilizing limited memory resources. This is in benefit of possible further implementation on ASIC, where memory resources are in the critical importance. This was achieved thanks to the use of multiple data reduction techniques as well as parallel and pipelined design approaches applied.

The system is fully customizable and is open to further expansion. Thanks to the data driven approach, the timing is fixed and fully predictable. Therefore, further hardware accelerators can be embedded within the data path at a minimal engineering cost. Current implementation occupies 24% of a single Virtex-II Pro XC2VP30 FPGA chip, leaving plenty space for further development. The on-chip memory is utilized in 86%, therefore an interface to an off-chip memory should be considered for memory demanding applications.

The system is capable of processing IR video stream provided through the ADC video decoder board in the PAL format. Video stream is converted into its digital form, de-interlaced and encoded with synchronization signals at a rate of 27MHz clock provided by ADC. Digital data is further processed synchronously with 25MHz pixel clock provided by the controller for video graphics adapter. Results of the background segmentation, intensity thresholding and connected component labelling are available at the end of the image scan, at a rate of $60\textit{fps}$.

There are three hardware accelerators involved in the processing chain: background subtraction, morphology filtering and connected component labelling. For the purpose of background segmentation, a number of processing techniques was reviewed. The trade-off between the quality of output data and the processing power was taken into consideration. Therefore, algorithms based on intensity thresholding as well as on temporal difference, although suitable for robust implementation in HW, were not taken into consideration due to poor quality of data produced. For the purpose of this project, a new background subtraction technique was developed based on background modelling approach. The background model is adaptive, hence capable of processing in changing environmental conditions. The technique is a mixture of multiple algorithms commonly used in image processing applications. Background model is based on selective running average. In order to allow segmentation in both slow and fast changing environmental conditions, it is supported by adaptive thresholding mechanism based on live histogram calculation. This is a novel enhancement for segmentation based on background modelling, not reported in the literature yet. Thanks to efficient memory management, the memory consumption of this architecture is a major advantage compared to other algorithms described in previous subsections. The quality of output data is at satisfactory level. The final integration of the system revealed the increase in error detection rate was caused by multiple splits and merges, an issue out of the scope segmentation module.

The second module further in the processing chain is responsible for acquisition noise removal. This particular architecture is based on mathematical morphology and basic set theory definitions. Since flat rectangular scan mask was used, a number of MM set theory features could be applied. Furthermore, it takes an advantage of the duality and decomposition of erosion and dilation. This technique was compared with the classical approach for noise removal based on the delayed-element and sliding window. It features lower complexity and benefits from lower memory consumption. Although both techniques can be applied within the system to perform real-time noise removal, the new architecture suits better hardware implementation - a single architecture for both erosion and dilation. Moreover, the memory consumption becomes an issue for classical approach when considering higher resolution images. The decomposed dual architecture is already up to 300% more efficient when implemented for VGA image frame with SE size 13×7, with better margin for higher resolution and larger SEs. In order to avoid the need for additional data generators, a static approach to handle padding areas was introduced, utilizing horizontal blanking periods. This approach is suitable for streaming data systems and shall be considered as a contribution to this thesis. Although the padding can be processed simultaneously with the image scan, it implies a fixed latency to the pipeline, proportional to the size of the scan mask.

In order to distinguish disjoint groups of pixels and extract their features, a connected component labelling algorithm was applied, based on a single-pass approach. This technique was chosen based on extensive review of CCL algorithms. It features constant processing time and is capable of feature extraction such as position, width, size, CoG, therefore it is suitable for this particular application. Moreover, the single pass algorithm requires less memory resources due to unique labelling approach. It is the major benefit of this architecture. In order to further improve memory management, a label-reuse technique was developed, being a significant contribution to this technique. This approach is highly beneficial when processing video streams with a significant number label collisions. For implementation purposes, the 4-neighbourhood connectivity $N_4(p)$ mask was used to reduce system complexity. The implementation of CCL algorithm performs labelling and feature extraction in a single image scan. However, since it provides different type results compared to classical labelling approach (no output image with labelled objects), hence, it is not suitable for applications where the labelled mask is required.

In order to verify and test system components, a full system integration was performed. For this task an architecture for real-time video processing was designed and developed. It is based on the assumption the IR video signal is provided to the system in a PAL format. The custom video acquisition module allows to significantly reduce the amount of incoming data already during analogue to digital conversion. The image data is synchronously grid aligned within the video buffer, at 27MHz pixel clock. It is also provided to the segmentation module for further processing. At the final stage of the processing, both input video stream and results of the processing (extracted features used to draw bounding boxes around pedestrians) are muxed to be plotted on the monitor display. The system was tested with pre-recorded video streams. Initial experimental detection rates give results around 80%-85% for busy environments (around 3 pedestrians per FOV in average), with up to 10 pedestrians in a single image frame. These are satisfactory results for basic surveillance system. However, due to repetitive mergers and splits, the error detection rate varies around 13% for the same test set.

The main contribution is the digital architecture designed, developed and implemented for IR pedestrian detection system, capable of processing video streams in real-time. As well as the architecture, there is the analysis and evaluation of the algorithms and design decisions that support and justify the digital design. Although the system can be used for surveillance applications as an off-the-shelf solution, an emphasis was placed on further expansion, for instance to support tracking applications as a hardware accelerator. The system was designed in a modular approach where individual modules act as hardware accelerators for particular processing tasks. They were developed based on available solutions reported in the literature. For implementation purposes, they are enhanced by custom solutions allowing the real-time performance of the entire system.

7.3 Systems of Today

Since early days of this research (October 2007) the technology has changed rapidly. Thanks to improved manufacturing process, prices of silicon devices are lower whereas the maximum operating frequency in in digital circuits is significantly higher.

According to recent surveys throughout different architectures, there are several alternatives for FPGA devices in the field of low-cost high performance computing [133–136]. These are multi-core CPUs, Graphical Processing Units (GPUs) and Parallel Processor Arrays. They feature high peak performance, some of them are energy efficient whereas others are cost efficient. In order to select a device for particular application, a number of features has to be considered, such as programmability, performance, development cost and sources of overhead in the design flow.

For the purpose of real-time pedestrian detection from video stream, two architectures shall be considered: multi-core CPUs and FPGAs. In terms of flexibility and power consumption, FPGA is a clear leader. However, there are major drawbacks such as accessibility (implementation difficulties) and limited resources. On the other hand, if implementation of high complexity algorithms is desired with no consideration for power consumption or portability, multi-core devices are better choice.

In terms of video source, although CCD and CMOS sensors feature much higher spatial resolution compared to IR cameras, for the purpose of pedestrian detection in outdoor environments, IR-based devices cannot be replaced with cheaper alternatives. They provide significantly more information thanks to the use of thermal radiation, moreover they can operate in low or even no-light conditions, when automated surveillance is highly desired.

7.4 Future Work

Although IR cameras feature relatively low resolution compared to traditional image sensors and it does not change as fast, the aspect of increasing bandwidth imposed by the video source shall be considered as a future work. Moreover, an investigation into further improvement in classification shall be considered, possibly through the implementation of tracking algorithms.

When considering higher resolution video signals, an interface to auxiliary memory is required to support increased bandwidth on segmentation unit. Moreover, an expansion of the CCL algorithm to Run Length Encoding-based approach is desired to reduce the system complexity.

According to the discussion provided in Chapter 6, an implementation of tracking algorithms on embedded processor core is a natural extension to this project. Although the quality of segmentation makes an impact on classification, it was found that most detection errors in pedestrian recognition are caused by repetitive mergers and splits. This issue can be resolved by implementing alternative detection approach - detection based on tracking.

The implementation of tracking algorithms shall be software-based and targeted for either HW or SW embedded processor core. This approach would allow rapid algorithm prototyping while not affecting the overall system architecture.

References

Reference list is arranged by the order of citation.

[1] R. Hudson Jr, *Infrared system engineering.* Wiley-Interscience, New York and London, 1969.

[2] R. Smith, F. Jones, and R. Chasmar, *The detection and measurement of infra-red radiation.* Clarendon press, 1968.

[3] S. Frei, *Capture and transmission of signal data from infrared detectors using a microcontroller and graphical display of received data in a Windows environment.* Bachelors thesis, School of Engineering and Built Environment, Edinburgh Napier University, Edinburgh, UK, Aug 2009.

[4] F. Sowan, *Applications of infrared detectors.* Mullard Ltd., 1971.

[5] FLIR Systems Thermacam PM595. [Online]. Available: http://www.flir.com/, 2010.

[6] Selex Galileo SLX Merlin. [Online]. Available: http://www.selexgalileo.com/, 2011.

[7] J. Lloyd, *Thermal imaging systems.* Springer, 1975.

[8] A. El Maadi and X. Maldague, "Outdoor infrared video surveillance: A novel dynamic technique for the subtraction of a changing background of ir images," *Infrared physics & technology*, vol. 49, no. 3, pp. 261–265, 2007.

[9] I. Amin, A. Taylor, F. Junejo, A. Al-Habaibeh, and R. Parkin, "Automated people-counting by using low-resolution infrared and visual cameras," *Measurement*, vol. 41, no. 6, pp. 589–599, 2008.

[10] R. Haralick and L. Shapiro, *Computer and robot vision.* Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1992.

[11] R. Gonzalez and R. Woods, *Digital image processing.* Reading, Mass. ; Wokingham : Addison-Wesley, 1993.

[12] Wikipedia, “General purpose preprocessor,” 2011. [Online]. Available: http://en.wikipedia.org/wiki/Preprocessor#General_purpose_preprocessor

[13] Kalimba DSP User Guide. [Online]. Available: http://www.csr.com/, 2003.

[14] C. Maxfield, *The design warrior’s guide to FPGAs: devices, tools and flows.* Elsevier, 2004, vol. 1.

[15] P. Chu, *FPGA prototyping by VHDL examples: Xilinx Spartan-3 version.* John Wiley & Sons, 2008.

[16] Wikipedia, “Application-specific integrated circuit,” 2011. [Online]. Available: http://en.wikipedia.org/wiki/Application-specific_integrated_circuit

[17] L. Adams and S. Marketing, “Choosing the right architecture for real-time signal processing designs,” *Texas Instruments, Document Number SPRA879*, 2002.

[18] T. Instruments, “Picking the right architecture,” 1995-2013. [Online]. Available: http://www.ti.com/lsds/ti/dsp/getstarted/dspright_arch.page

[19] S. Bove, “Digital signal processing solutions,” 2005. [Online]. Available: http://satelliteconferences.noaa.gov/Miami04/docs/weds/Receiver_Technology.pdf

[20] T. Instruments, “Dsp and arm mpu selection tool,” 2013. [Online]. Available: http://focus.ti.com/en/multimedia/flash/selection_tools/dsp/dsp.html

[21] ——, “Embedded processor module and single board computer selection tool,” 2013. [Online]. Available: http://focus.ti.com/en/multimedia/flash/selection_tools/dsp-3p/dsp3p-tool.htm

[22] ——, “Mcu selection tool,” 2013. [Online]. Available: http://www.ti.com/lsds/ti/microcontroller/products.page

[23] Garrault P. and Philofsky B., “Hdl coding practices to accelerate design performance,” Xilinx., Tech. Rep., Jan 2005. [Online]. Available: http://www.xilinx.com/support/documentation/white_papers/wp231.pdf

[24] XESS Corp., “Using Xilinx WebPACK Software to Create FPGA Designs for the XSA Board,” XESS Corp., Tech. Rep., May 2005. [Online]. Available: http://www.xess.com/appnotes/

[25] Official Xilinx Spartan-3A documentation website. [Online]. Available: http://www.xilinx.com/products/spartan3a/, 2010.

[26] Official Xilinx Virtex-II documentation website. [Online]. Available: http://www.xilinx.com/support/documentation/virtex-ii.htm, 2010.

[27] P. Chu, *RTL hardware design using VHDL: coding for efficiency, portability, and scalability.* Wiley-IEEE Press, 2006.

[28] XUP V2P Development Board, Digilent Inc. [Online]. Available: http://digilentinc.com/xupv2p.htm, 2010.

[29] A. Benkhalil, S. Ipson, and W. Booth, "A novel cpld based implementation of a motion detection algorithm for surveillance applications," in *Custom Integrated Circuits Conference, 1998. Proceedings of the IEEE 1998.* IEEE, 1998, pp. 105–108.

[30] R. Walczyk, A. Armitage, and T. Binnie, "An Embedded Real-Time Pedestrian Detection System Using an Infrared Camera," in *IET Irish Signals and Systems Conference, 2009. Dublin, Ireland*, 2009, pp. 1–6.

[31] F. Kristensen, P. Nilsson, and V. Öwall, "Background segmentation beyond rgb," *Computer Vision–ACCV 2006*, pp. 602–612, 2006.

[32] M. Piccardi, "Background subtraction techniques: a review," in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, vol. 4. Ieee, 2004, pp. 3099–3104.

[33] Y. Benezeth, B. Emile, and C. Rosenberger, "Comparative study on foreground detection algorithms for human detection," in *Image and Graphics, 2007. ICIG 2007. Fourth International Conference on.* IEEE, 2007, pp. 661–666.

[34] S. Cheung and C. Kamath, "Robust background subtraction with foreground validation for urban traffic video," *Eurasip Journal on applied signal processing*, vol. 2005, pp. 2330–2340, 2005.

[35] S. Benton, "Background subtraction, part 1: Matlab models," 2008.

[36] E. Goubet, J. Katz, and F. Porikli, "Pedestrian tracking using thermal infrared imaging," *Mitsubishi Electric Research Laboratories, Technical Report, TR2005-126*, 2006.

[37] N. Dhulekar, A. Felch, and R. Granger, "Tracking moving objects improves recognition," in *IPCV*, 2010, pp. 798–803.

[38] J. Kerridge, S. Keller, T. Chamberlain, and N. Sumpter, "Collecting pedestrian trajectory data in real-time," *Pedestrian and Evacuation Dynamics 2005*, pp. 27–39, 2007.

[39] M. Shankar, J. Burchett, Q. Hao, B. Guenther, and D. Brady, "Human-tracking systems using pyroelectric infrared detectors," *Optical Engineering*, vol. 45, no. 10, pp. 106 401–106 401, 2006.

[40] M. Bramberger, R. Pflugfelder, B. Rinner, H. Schwabach, and B. Strobl, "Intelligent traffic video sensor: Architecture and applications," in *Proceedings of the Telecommunications and Mobile Computing Workshop*, 2003.

[41] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2. IEEE, 1999.

[42] A. Treptow, G. Cielniak, and T. Duckett, "Active people recognition using thermal and grey images on a mobile security robot," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on.* IEEE, 2005, pp. 2103–2108.

[43] H. Nanda and L. Davis, "Probabilistic template based pedestrian detection in infrared videos," in *Intelligent Vehicle Symposium, 2002. IEEE*, vol. 1. IEEE, 2002, pp. 15–20.

[44] A. Benkhalil, S. Ipson, and W. Booth, "Real-time video surveillance system using a field programmable gate array," *International Journal of Imaging Systems and Technology*, vol. 11, no. 2, pp. 130–137, 2000. [Online]. Available: http://dx.doi.org/10.1002/1098-1098(2000)11:2<130::AID-IMA4>3.0.CO;2-A

[45] T. Alexandropoulos, S. Boutas, V. Loumos, and E. Kayafas, "Real-time change detection for surveillance in public transportation," in *Advanced Video and Signal Based Surveillance, 2005. AVSS 2005. IEEE Conference on.* IEEE, 2005, pp. 58–63.

[46] J. Lou, H. Yang, W. Hu, and T. Tan, "An illumination invariant change detection algorithm," in *Asian Conf. Computer Vision.* Citeseer, 2002, pp. 13–18.

[47] S. Chien, S. Ma, and L. Chen, "Efficient moving object segmentation algorithm using background registration technique," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 12, no. 7, pp. 577–586, 2002.

[48] J. Moon, D. Kim, and R. Park, "Video matting based on background estimation," *Proc. World Acad. Sci., Eng. Technol*, vol. 2, pp. 149–152, 2005.

[49] Q. Zang and R. Klette, "Robust background subtraction and maintenance," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 2. IEEE, 2004, pp. 90–93.

[50] F. Porikli, "Multiplicative background-foreground estimation under uncontrolled illumination using intrinsic images," in *Motion and Video Computing, 2005. WACV/MOTIONS'05 Volume 2. IEEE Workshop on*, vol. 2. IEEE, 2005, pp. 20–27.

[51] S. Cheung and C. Kamath, "Robust techniques for background subtraction in urban traffic video," in *Proceedings of SPIE*, vol. 5308, 2004, pp. 881–892.

[52] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and practice of background maintenance," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1. Ieee, 1999, pp. 255–261.

[53] B. Lo and S. Velastin, "Automatic congestion detection system for underground platforms," in *Intelligent Multimedia, Video and Speech Processing, 2001. Proceedings of 2001 International Symposium on.* IEEE, 2001, pp. 158–161.

[54] R. Cutler and L. Davis, "View-based detection and analysis of periodic motion," in *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, vol. 1. IEEE, 1998, pp. 495–500.

[55] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 10, pp. 1337–1342, 2003.

[56] I. Haritaoglu, D. Harwood, and L. Davis, "W4: Real-time surveillance of people and their activities," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 8, pp. 809–830, 2000.

[57] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russell, "Towards robust automatic traffic scene analysis in real-time," in *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*, vol. 1. IEEE, 1994, pp. 126–131.

[58] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, “Pfinder: Real-time tracking of the human body,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 7, pp. 780–785, 1997.

[59] J. Makhoul, “Linear prediction: A tutorial review,” *Proceedings of the IEEE*, vol. 63, no. 4, pp. 561–580, 1975.

[60] B. Anderson and J. Moore, *Optimal filtering.* Prentice-Hall Englewood Cliffs, NJ, 1979, vol. 11.

[61] J. Zhong and S. Sclaroff, “Segmenting foreground objects from a dynamic textured background via a robust kalman filter,” in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on.* IEEE, 2003, pp. 44–50.

[62] A. Monnet, A. Mittal, N. Paragios, and V. Ramesh, “Background modeling and subtraction of dynamic scenes,” in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on.* Ieee, 2003, pp. 1305–1312.

[63] P. Power and J. Schoonees, “Understanding background mixture models for foreground segmentation,” in *Proceedings Image and Vision Computing New Zealand*, vol. 2002, 2002.

[64] H. Jiang, V. Owall, and H. Ardo, “Real-time video segmentation with vga resolution and memory bandwidth reduction,” in *Video and Signal Based Surveillance, 2006. AVSS’06. IEEE International Conference on.* IEEE, 2006, pp. 104–104.

[65] A. Elgammal, D. Harwood, and L. Davis, “Non-parametric model for background subtraction,” *Computer Vision ECCV 2000*, pp. 751–767, 2000.

[66] A. Bovik, *The essential guide to Image Processing.* Academic Pr, 2009.

[67] VDEC1 Video Decoder Board. [Online]. Available: http://www.digilentinc.com/Data/Products/VDEC1/VDEC1-rm.pdf, 2010.

[68] Mohit Kumar, “Compute a histogram in an fpga with one clock,” Texas Instruments, Bangalore, India, Tech. Rep., Feb 2011. [Online]. Available: http://www.edn.com/design/integrated-circuit-design/4363979/Compute-a-histogram-in-an-FPGA-with-one-clock

[69] A. Shahbahrami, J. Hur, B. Juurlink, and S. Wong, “Fpga implementation of parallel histogram computation,” in *2nd HiPEAC Workshop on Reconfigurable Computing, Göteborg, Sweden*, 2008, pp. 63–72.

[70] R. Walczyk, “Ir video sequence 1,” 2013. [Online]. Available: http://youtu.be/_HOp6zvmajw

[71] ——, “Ir video sequence 2,” 2013. [Online]. Available: http://youtu.be/b4iV2PHkHcw

[72] ——, “Results of the processing, demo video sequence 1,” 2013. [Online]. Available: http://youtu.be/iTF9UDEjpV0

[73] ——, “Results of the processing, demo video sequence 2,” 2013. [Online]. Available: http://youtu.be/n_wDjdPK69s

[74] ——, “Results of the processing, demo video sequence 3,” 2013. [Online]. Available: http://youtu.be/yUiNS3MkjWs

[75] ——, “Results of the processing, demo video sequence 4,” 2013. [Online]. Available: http://youtu.be/1uQMua0b72E

[76] G. Matheron, G. Matheron, G. Matheron, and G. Matheron, *Random sets and integral geometry.* Wiley New York, 1975, vol. 9.

[77] J. Serra, *Image analysis and mathematical morphology.* Academic Press, Inc. Orlando, FL, USA, 1983.

[78] R. Hsu, Y. Lee, B. Kao, and D. Chan, “Hardware Design of Shape-Preserving Contour Tracing for Object of Segmented Images,” *Advances in Image and Video Technology*, pp. 976–987, 2009.

[79] C. Chen, R. Hwang, and Y. Chen, “A passive auto-focus camera control system,” *Applied Soft Computing*, vol. 10, no. 1, pp. 296–303, 2010.

[80] K. Diamantaras and S. Kung, “A linear systolic array for real-time morphological image processing,” *The Journal of VLSI Signal Processing*, vol. 17, no. 1, pp. 43–55, 1997.

[81] A. Loui, A. Venetsanopoulos, and K. Smith, “Flexible architectures for morphological image processing and analysis,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 2, no. 1, pp. 72–83, Mar. 1992.

[82] L. Lucke and C. Chakrabarti, “A digit-serial architecture for gray-scale morphological filtering,” *Image Processing, IEEE Transactions on*, vol. 4, no. 3, pp. 387–391, 1995.

[83] R. Haralick, S. Sternberg, and X. Zhuang, "Image analysis using mathematical morphology," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 4, pp. 532–550, 1987.

[84] A. Rosenfeld and A. Kak, *Digital picture processing.* Academic Press, Inc. Orlando, FL, USA, 1982.

[85] J. Serra, "Image Analysis and Mathematical Morphology. 11: Theoretical Advances," 1988.

[86] P. Maragos and R. Schafer, "Morphological systems for multidimensional signal processing," *Proceedings of the IEEE*, vol. 78, no. 4, pp. 690–710, 1990.

[87] H. Minkowski, *Volumen und oberfläche.* Math. Ann., 1903, vol. 57.

[88] E. Dougherty and R. Lotufo, *Hands-on morphological image processing.* Society of Photo Optical, 2003.

[89] H. Hedberg, F. Kristensen, and V. Owall, "Low-complexity binary morphology architectures with flat rectangular structuring elements," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 55, no. 8, pp. 2216–2225, 2008.

[90] G. Anelli, A. Broggi, and G. Destri, "Decomposition of arbitrarily shaped binary morphological structuring elements using genetic algorithms," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 2, pp. 217–224, 1998.

[91] F. Shih and Y. Wu, "Decomposition of binary morphological structuring elements based on genetic algorithms," *Computer Vision and Image Understanding*, vol. 99, no. 2, pp. 291–302, 2005.

[92] E. Malamas, A. Malamos, and T. Varvarigou, "Fast implementation of binary morphological operations on hardware-efficient systolic architectures," *The Journal of VLSI Signal Processing*, vol. 25, no. 1, pp. 79–93, 2000.

[93] J. Velten and A. Kummert, "FPGA-based implementation of variable sized structuring elements for 2D binary morphological operations," in *Electronic Design, Test and Applications, 2002. Proceedings. The First IEEE International Workshop on.* IEEE, 2002, pp. 309–312.

[94] H. Hedberg, F. Kristensen, P. Nilsson, and V. Owall, "A low complexity architecture for binary image erosion and dilation using structuring element decomposition," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on.* IEEE, 2005, pp. 3431–3434.

[95] MathWorks Matlab Suite. [Online]. Available: http://www.mathworks.com/products/matlab/, 2010.

[96] Xilinx ISE WebPACK Design Software. [Online]. Available: http://www.xilinx.com/tools/webpack.htm, 2010.

[97] Mentor Graphics ModelSim official website. [Online]. Available: http://model.com/, 2010.

[98] ThermoVision Micron A10 Infrared Camera. [Online]. Available: http://www.flir.com/, 2010.

[99] A. Rosenfeld and J. Pfaltz, "Sequential operations in digital picture processing," *J. ACM*, vol. 13, no. 4, pp. 471–494, 1966.

[100] D. Nassimi and S. Sahni, "Finding connected components and connected ones on a mesh-connected parallel computer," *SIAM Journal on computing*, vol. 9, p. 744, 1980.

[101] R. Haralick, "Some neighborhood operations," *Real Time/Parallel Computing Image Analysis*, pp. 11–35, 1981.

[102] R. Lumia, L. Shapiro, and O. Zuniga, "A new connected components algorithm for virtual memory computers," *Computer Vision, Graphics, and Image Processing*, vol. 22, no. 2, pp. 287–300, 1983.

[103] H. Samet and M. Tamminen, "An improved approach to connected component labeling of images," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition, Miami, Florida*, 1986, pp. 312–318.

[104] F. Chang and J. Chen, "C., A Component-Labelling Algorithm Using Contour Tracing Technique," in *IEEE Proc. 7th International Conference on Document Analysis and Recognition (ICIDAR 2003), 0-7695-1960-1/03*, 2003.

[105] H. Freeman, "Techniques for the digital computer analysis of chain-encoded arbitrary plane curves," in *Proc. Nat. Electronics Conf*, vol. 17, 1961, pp. 412–432.

[106] T. Pavlidis, "Algorithms for graphics and image processing," *Computer Science Press*, 1982.

[107] K. Suzuki, I. Horiba, and N. Sugie, "Linear-time connected-component labeling based on sequential local operations," *Computer Vision and Image Understanding*, vol. 89, no. 1, pp. 1–23, 2003.

[108] D. G. Bailey and C. T. Johnston, “Single pass connected components analysis,” in *Image and Vision Computing New Zealand.* Citeseer, 2008, pp. 282–287.

[109] R. Haralick and L. Shapiro, *Computer and Robot Vision.* Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1992, vol. 1.

[110] M. Jablonski and M. Gorgon, “Handel-c implementation of classical component labelling algorithm,” *Digital Systems Design, Euromicro Symposium on*, vol. 0, pp. 387–393, 2004.

[111] D. Crookes and K. Benkrid, “FPGA implementation of image component labelling,” *Reconfigurable Technology: FPGAs for Computing and Applications*, pp. 17–23, 1999.

[112] M. Manohar and H. K. Ramapriyan, “Connected component labeling of binary images on a mesh connected massively parallel processor,” *Computer Vision, Graphics, and Image Processing*, pp. 133–149, 1989.

[113] E. Mozef, S. Weber, J. Jaber, and G. Prieur, “Parallel architecture dedicated to image component labeling in o(n log n): Fpga implementation,” in *Proceedings of SPIE*, O. Loffeld, Ed., vol. 2784, no. 1. SPIE, 1996, pp. 120–125.

[114] H. Hedberg, F. Kristensen, and V. Owall, “Implementation of a labeling algorithm based on contour tracing with feature extraction,” in *IEEE International Symposium on Circuits and Systems, 2007. ISCAS 2007*, 2007, pp. 1101–1104.

[115] J. Trein, A. Schwarzbacher, B. Hoppe, K. H. Noffz, and T. Trenschel, “Development of a FPGA Based Real-Time Blob Analysis Circuit,” in *Irish Systems and Signals Conference, 2007. Derry, N. Ireland*, 2007, pp. 121–126.

[116] C. Johnston and D. Bailey, “FPGA implementation of a single pass connected components algorithm,” *Electronic Design, Test and Applications*, pp. 228–231, 2008.

[117] F. Chang, C. Chen, and C. Lu, “A linear-time component-labeling algorithm using contour tracing technique,” *Computer Vision and Image Understanding*, vol. 93, no. 2, pp. 206–220, 2004.

[118] S. Pedre, A. Stoliar, and P. Borensztejn, “Real Time Hot Spot Detection Using FPGA,” *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pp. 595–602, 2009.

[119] M. Huang, C. Wang, and Y. Liu, "High-Speed Video Transfer and Real-Time Infrared Spots Detection Based on FPGA," in *Computer Science and Engineering, 2009. WCSE'09. Second International Workshop on*, vol. 2. IEEE, 2010, pp. 154–159.

[120] R. Walczyk, A. Armitage, and T. Binnie, "FPGA Implementation of Hot Spot Detection in Infrared Video," in *IET Irish Signals and Systems Conference, 2010. Cork, Ireland*, 2010, pp. 1–6.

[121] Official Xilinx website. [Online]. Available: www.xilinx.com, 2010.

[122] Official Xilinx Intellectual Property website. [Online]. Available: http://www.xilinx.com/ipcenter/, 2010.

[123] Official SparkFun Electronics product website. [Online]. Available: http://www.sparkfun.com/products/582/, 2010.

[124] ADV7183B: 10-bit NTSC/PAL/SECAM TV & Video Decoder. [Online]. Available: http://www.analog.com/en/audiovideo-products/video-decoders/adv7183b/products/product.html, 2005.

[125] K. Jack, *Video Demystified: A Handbook for the Digital Engineer*, 2nd ed. L L H Technology Publishing, 1995.

[126] Gregg Hawkes, "Line field decoder," Xilinx., Tech. Rep., Dec 2001.

[127] K. Jack, *Video demystified: a handbook for the digital engineer.* Newnes, 2007.

[128] R. Walczyk, A. Balazs, A. Armitage, and T. Binnie, "System architectures for infrared pedestrian tracking," in *IET Irish Signals and Systems Conference, 2011. Dublin, Ireland*, 2011, pp. 1–6.

[129] A. Balazs, *Design and implementation of a real-time pedestrian tracking system utilizing FPGA and processor co-design.* Bachelors thesis, School of Engineering and Built Environment, Edinburgh Napier University, Edinburgh, UK, Aug 2011.

[130] Sony Digital Video Cassette Recorder. [Online]. Available: http://www.manualowl.com/p/Sony/GV-D300/Manual/32067, 2011.

[131] J. W. Davis and M. A. Keck, "A two-stage template approach to person detection in thermal imagery," in *Application of Computer Vision, 2005. WACV/MOTIONS'05 Volume 1. Seventh IEEE Workshops on*, vol. 1. IEEE, 2005, pp. 364–369.

[132] C. Dai, Y. Zheng, and X. Li, “Pedestrian detection and tracking in infrared imagery using shape and appearance,” *Computer Vision and Image Understanding*, vol. 106, no. 2, pp. 288–299, 2007.

[133] C. Grozea, Z. Bankovic, and P. Laskov, “Fpga vs. multi-core cpus vs. gpus: hands-on experience with a sorting application,” in *Facing the multicore-challenge.* Springer, 2011, pp. 105–117.

[134] S. Che, J. Li, J. W. Sheaffer, K. Skadron, and J. Lach, “Accelerating compute-intensive applications with gpus and fpgas,” in *Application Specific Processors, 2008. SASP 2008. Symposium on.* IEEE, 2008, pp. 101–107.

[135] D. B. Thomas, L. Howes, and W. Luk, “A comparison of cpus, gpus, fpgas, and massively parallel processor arrays for random number generation,” in *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays.* ACM, 2009, pp. 63–72.

[136] A. R. Brodtkorb, C. Dyken, T. R. Hagen, J. M. Hjelmervik, and O. O. Storaasli, “State-of-the-art in heterogeneous computing,” *Scientific Programming*, vol. 18, no. 1, pp. 1–33, 2010.

[137] Official Xilinx PowerPC documentation website. [Online]. Available: http://www.xilinx.com/products/powerpc/, 2008.

[138] Xilinx Embedded Development Kit. [Online]. Available: http://www.xilinx.com/tools/edk.htm, 2010.

[139] J. Noseworthy and M. Leeser, “Efficient use of communications between an fpga’s embedded processor and its reconfigurable logic,” in *Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate arrays*, ser. FPGA ’06. New York, NY, USA: ACM, 2006, pp. 233–233. [Online]. Available: http://doi.acm.org/10.1145/1117201.1117257

[140] Xilinx, “Edk concepts, tools and techniques: A hands on guide to effect embedded system design,” Xilinx., Tech. Rep., Apr 2011.

[141] X. Li, K. Wang, W. Wang, and Y. Li, “A multiple object tracking method using kalman filter,” in *Information and Automation (ICIA), 2010 IEEE International Conference on*, 2010, pp. 1862 –1866.

APPENDIX A

An Appendix

A.1 Implementation of Tracking on PowerPC

Although implementation of tracking algorithms using embedded processor core is a natural extension for this research, is was not in the scope of this thesis. An initial approach to tracking was developed as an honours project by Alexander Balazs under guidance and supervision of the author. The project was concluded with the conference paper [128].

This section gives an introduction into tracking together with brief description of the system implementation. It is based on [129]. A pointer to this section can be found in Chapter 6, Section 6.3.4.

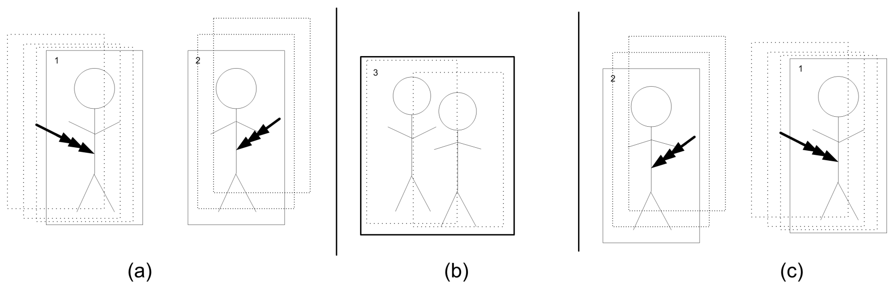


Figure A.1: Merging pedestrians. (a). Before merge. (b). Pedestrians crossing their ways. (c). Split.

Introduction

In order to track an object in a video stream, its location has to be known for certain number of frames. Such data is then stored in the memory and can be used to display the object's path in time or for further analysis. Pedestrian tracking is not a trivial task. In real life case scenarios, moving pedestrians cross their ways or occasionally overlap, as depicted in Figure A.1. The problem of tracking becomes even more complex when objects split, merge, disappear or become partially occluded.

An implementation of tracking algorithms was partitioned for SW using PowerPC embedded processor core [137]. For the purpose of system architecture development, a Base System Builder (BSB) within Embedded Development Kit (EDK) was used [138]. In order to emulate movement of pedestrians, a custom HDL module was created and added to the system. Such module generates corner coordinates for two pedestrians to simulate possible splits and merges.

System Overview

The system implementation was targeted for XUP V2P Development Platform using embedded PowerPC processor code [137]. It was built with the following settings:

- PowerPC as a target processor,
- 100MHz processor clock frequency,
- 100MHz bus frequency,
- 64KB program size,
- UART for debug using Serial Port.

The system generated in EDK includes the following components: Digital Clock Manager (DCM), On-Chip Memory (OCM), Processor Local Bus (PLB), On-chip Peripheral Bus (OPB), UART, as well as an OPB-to-PLB bus bridge for UART and Block RAM. It also includes a custom IP to emulate movement of pedestrians within the FOV.

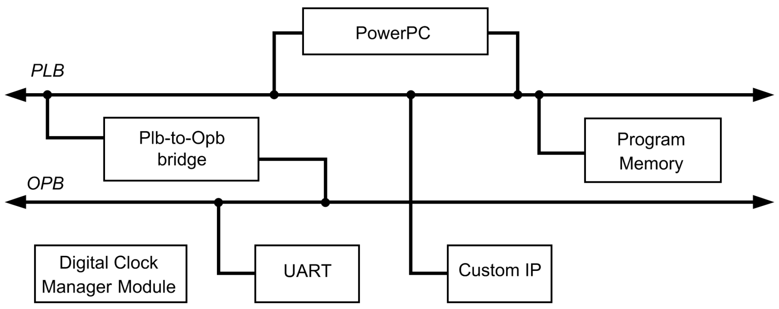


Figure A.2: A block diagram of the top-level system hierarchy including processor buses.

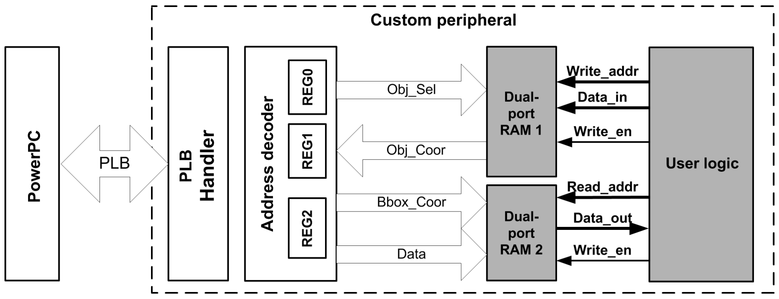


Figure A.3: Register level interface.

System Architecture

The framework of the system architecture is based on the BSB configuration and auto generated by EDK. An overview of the top-level design structure including buses for processor interface can be seen in Figure A.2. OPB was instantiated for the purpose of interfacing slower peripherals such as UART and control switches. The custom IP however was connected to the processor core through PLB, following suggestions from [139]. In order to instantiate custom logic with PowerPC, the RTL code required several modifications to match IP Interconnect (IPIC) interface protocol. Detailed description regarding of custom IP interfaces can be found in [140].

In order to interface the processor core with user logic emulating pedestrian movement, three 32-bit registers were used. The custom peripheral interface supports two sets of (x, y) coordinates referring to the top-left and the bottom-right corners of the rectangular box for each pedestrian. Such data is temporarily stored in registers `REG0`, `REG1` and `REG2` as depicted in Figure A.3. The `PLB Handler` manages communication with PowerPC responsible for processing merges or splits as well as generating bounding boxes around. Further details regarding this implementation can be found in [128, 129].

Tracking

The custom IP was created to generate movement of two pedestrians. The purpose of such implementation was to generate a set of test vectors to emulate most common tracking case scenarios.

Due to the time-scale of this project, a limited number of tests was performed. An initial algorithm for handling merges and splits was suggested, based on [141]. It is a predictive approach based on Kalman filter. When objects merge, the system constantly preserves the height and width for both objects. The movement prediction is calculated based on samples for previous position. After the split, objects are assigned with labels based on both criterions: height and width before the merge as well as movement prediction.

A natural way to improve prediction accuracy is to use additional features extracted during connected component labelling, such as size (number of pixels) or Centre of Gravity (CoG). This, as well as multiple objects tracking was considered as a future work.

Summary

In this subsection, an initial approach to implementation of tracking algorithm using PowerPC was described. It gives details on HW/SW wrapper developed for the purpose of this task. The custom logic used to emulate pedestrian movement was implemented as a subsystem for the embedded processor core. It was optimized in order to be interfaced with the processor using bi-directional 32-bit software addressable registers. The system build involves modules such as peripheral cores, processor buses, program memory blocks and digital clock managers. It was generated using proprietary tools from Xilinx with custom settings for the internal clock of the processor, transmission bus and program memory size.

A.2 Experimental Detection Results

This attachment presents experimental detection results collected during the simulation performed on the running system. The data refers to simulations descried in Chapter 6. It was listed in three different tables, each one corresponding to different video sequence. Columns are assigned with the following symbols: #, P, B, M, S. They correspond to the index of the frame within the video sequence, number of pedestrians, bounding boxes, mergers and splits respectively. The data was plotted and can be seen in Figure 6.8.

#	P	B	M	S
302	2	2	0	0
303	2	2	0	0
304	2	2	0	0
305	2	2	0	0
306	2	2	0	0
307	2	2	0	0
308	2	2	0	0
309	2	2	0	0
310	2	2	0	0
311	2	2	0	0
312	2	2	0	0
313	2	2	0	0
314	3	3	0	0
315	3	3	0	0
316	3	3	0	0
317	3	3	0	0
318	3	3	0	0
319	3	3	0	0
320	3	3	0	0
321	3	3	0	0
322	3	3	0	0
323	3	3	0	0
324	3	3	0	0
325	3	3	0	0
326	3	3	0	0
327	3	3	0	0
328	3	3	0	0
329	3	3	0	0
330	3	3	0	0
331	3	3	0	0
332	3	3	0	0
333	3	3	0	0
334	3	3	0	0
335	3	3	0	0
336	3	3	0	0
337	3	3	0	0
338	3	3	0	0
339	3	3	0	0
340	3	3	0	0
341	3	3	0	0
342	3	3	0	0
343	3	3	0	0
344	3	3	0	0
345	3	3	0	0
346	3	3	0	0
347	3	3	0	0
348	3	3	0	0
349	3	3	0	0
350	3	3	0	0
351	3	3	0	0
352	3	3	0	0
353	3	3	0	0
354	3	3	0	0
355	3	3	0	0
356	3	3	0	0

#	P	B	M	S
371	1	1	0	0
372	1	1	0	0
373	1	1	0	0
374	1	1	0	0
375	1	1	0	0
376	1	1	0	0
377	1	1	0	0
378	1	1	0	0
379	1	1	0	0
380	1	1	0	0
381	1	1	0	0
382	1	1	0	0
383	1	1	0	0
384	1	1	0	0
385	1	1	0	0
386	1	1	0	0
387	1	1	0	0
388	1	1	0	0
389	1	1	0	0
390	1	1	0	0
391	1	1	0	0
392	1	1	0	0
393	1	1	0	0
394	1	1	0	0
395	1	1	0	0
396	1	1	0	0
397	1	1	0	0
398	1	1	0	0
399	1	1	0	0
400	1	1	0	0
401	1	1	0	0
402	1	1	0	0
403	1	1	0	0
404	1	1	0	0
405	1	1	0	0
406	1	1	0	0
407	1	1	0	0
408	1	1	0	0
409	1	1	0	0
410	1	1	0	0
411	1	1	0	0
412	1	1	0	0
413	1	1	0	0
414	1	1	0	0
415	1	1	0	0
416	1	1	0	0
417	1	1	0	0
418	1	1	0	0
419	1	1	0	0
420	1	1	0	0
421	1	1	0	0
422	1	1	0	0
423	1	1	0	0
424	1	1	0	0
425	1	1	0	0

#	P	B	M	S
3748	0	0	0	0
3749	0	0	0	0
3750	0	0	0	0
3751	0	0	0	0
3752	0	0	0	0
3753	0	0	0	0
3754	0	0	0	0
3755	0	0	0	0
3756	0	0	0	0
3757	0	0	0	0
3758	0	0	0	0
3759	0	0	0	0
3760	0	0	0	0
3761	0	0	0	0
3762	0	0	0	0
3763	0	0	0	0
3764	0	0	0	0
3765	0	0	0	0
3766	0	0	0	0
3767	0	0	0	0
3768	0	0	0	0
3769	0	0	0	0
3770	0	0	0	0
3771	0	0	0	0
3772	0	0	0	0
3773	0	0	0	0
3774	0	0	0	0
3775	0	0	0	0
3776	0	0	0	0
3777	0	0	0	0
3778	0	0	0	0
3779	0	0	0	0
3780	0	0	0	0
3781	0	0	0	0
3782	1	0	0	0
3783	1	0	0	0
3784	1	0	0	0
3785	1	0	0	0
3786	1	0	0	0
3787	1	1	0	0
3788	1	1	0	0
3789	1	1	0	0
3790	1	1	0	0
3791	1	1	0	0
3792	1	1	0	0
3793	1	1	0	0
3794	2	1	1	0
3795	2	1	1	0
3796	2	1	1	0
3797	2	1	1	0
3798	2	1	1	0
3799	2	1	1	0
3800	2	1	1	0
3801	2	1	1	0
3802	2	1	1	0

#	P	B	M	S
357	3	3	0	0
358	3	3	0	0
359	3	3	0	0
360	3	3	0	0
361	3	3	0	0
362	3	3	0	0
363	3	3	0	0
364	3	3	0	0
365	3	3	0	0
366	3	3	0	0
367	3	3	0	0
368	3	3	0	0
369	3	3	0	0
370	3	3	0	0
371	3	3	0	0
372	3	3	0	0
373	3	3	0	0
374	3	3	0	0
375	3	3	0	0
376	3	3	0	0
377	3	3	0	0
378	3	3	0	0
379	3	3	0	0
380	3	3	0	0
381	3	3	0	0
382	3	3	0	0
383	3	3	0	0
384	3	3	0	0
385	3	3	0	0
386	3	3	0	0
387	3	3	0	0
388	3	3	0	0
389	3	3	0	0
390	3	3	0	0
391	3	3	0	0
392	3	3	0	0
393	3	3	0	0
394	3	3	0	0
395	3	3	0	0
396	3	3	0	0
397	3	3	0	0
398	3	3	0	0
399	3	3	0	0
400	3	3	0	0
401	3	3	0	0
402	3	3	0	0
403	3	3	0	0
404	3	3	0	0
405	3	3	0	0
406	3	3	0	0
407	3	3	0	0
408	3	3	0	0
409	3	3	0	0
410	3	3	0	0
411	3	3	0	0

#	P	B	M	S
426	1	1	0	0
427	1	1	0	0
428	1	1	0	0
429	1	1	0	0
430	1	1	0	0
431	1	1	0	0
432	1	1	0	0
433	1	1	0	0
434	1	1	0	0
435	1	1	0	0
436	1	1	0	0
437	1	1	0	0
438	1	1	0	0
439	1	1	0	0
440	1	1	0	0
441	1	1	0	0
442	1	1	0	0
443	1	1	0	0
444	1	1	0	0
445	1	1	0	0
446	1	1	0	0
447	1	1	0	0
448	1	1	0	0
449	1	1	0	0
450	1	1	0	0
451	1	1	0	0
452	1	1	0	0
453	1	1	0	0
454	1	1	0	0
455	1	1	0	0
456	1	1	0	0
457	1	1	0	0
458	1	1	0	0
459	1	1	0	0
460	1	1	0	0
461	1	1	0	0
462	1	1	0	0
463	1	1	0	0
464	1	1	0	0
465	1	1	0	0
466	1	1	0	0
467	1	1	0	0
468	1	1	0	0
469	1	1	0	0
470	1	1	0	0
471	1	1	0	0
472	1	1	0	0
473	1	1	0	0
474	1	1	0	0
475	1	1	0	0
476	1	1	0	0
477	1	1	0	0
478	1	1	0	0
479	1	1	0	0
480	1	1	0	0

#	P	B	M	S
3803	2	1	1	0
3804	2	1	1	0
3805	2	1	1	0
3806	2	1	1	0
3807	2	1	1	0
3808	2	1	1	0
3809	2	1	1	0
3810	2	1	1	0
3811	2	1	1	0
3812	2	1	1	0
3813	2	1	1	0
3814	2	1	1	0
3815	2	1	1	0
3816	2	1	1	0
3817	2	1	1	0
3818	2	1	1	0
3819	2	1	1	0
3820	2	1	1	0
3821	2	1	1	0
3822	2	1	1	0
3823	2	1	1	0
3824	2	1	1	0
3825	2	1	1	0
3826	2	1	1	0
3827	2	1	1	0
3828	2	1	1	0
3829	2	1	1	0
3830	2	1	1	0
3831	2	1	1	0
3832	2	1	1	0
3833	2	1	1	0
3834	2	1	1	0
3835	2	1	1	0
3836	2	1	1	0
3837	2	1	1	0
3838	2	1	1	0
3839	2	1	1	0
3840	2	1	1	0
3841	2	1	1	0
3842	2	1	1	0
3843	2	1	1	0
3844	2	1	1	0
3845	2	1	1	0
3846	2	1	1	0
3847	2	1	1	0
3848	2	1	1	0
3849	2	1	1	0
3850	2	1	1	0
3851	2	1	1	0
3852	2	1	1	0
3853	2	1	1	0
3854	2	1	1	0
3855	2	1	1	0
3856	2	1	1	0
3857	2	1	1	0

412	3	3	0	0
413	3	3	0	0
414	3	3	0	0
415	3	3	0	0
416	3	3	0	0
417	3	3	0	0
418	3	3	0	0
419	3	3	0	0
420	3	3	0	0
421	3	3	0	0
422	3	3	0	0
423	3	3	0	0
424	3	3	0	0
425	3	3	0	0
426	3	3	0	0
427	3	3	0	0
428	3	3	0	0
429	3	3	0	0
430	3	3	0	0
431	3	3	0	0
432	3	3	0	0
433	3	3	0	0
434	3	3	0	0
435	3	3	0	0
436	3	3	0	0
437	3	3	0	0
438	3	3	0	0
439	3	3	0	0
440	3	3	0	0
441	3	4	0	1
442	3	4	0	1
443	4	4	0	0
444	4	4	0	0
445	4	4	0	0
446	4	4	0	0
447	4	4	0	0
448	4	4	0	0
449	4	4	0	0
450	4	4	0	0
451	4	4	0	0
452	4	4	0	0
453	4	4	0	0
454	4	4	0	0
455	4	4	0	0
456	4	4	0	0
457	4	4	0	0
458	4	4	0	0
459	4	4	0	0
460	4	4	0	0
461	4	4	0	0
462	4	4	0	0
463	4	4	0	0
464	4	4	0	0
465	4	4	0	0
466	4	4	0	0
467	4	4	0	0

481	1	1	0	0
482	1	1	0	0
483	1	1	0	0
484	1	1	0	0
485	1	1	0	0
486	1	1	0	0
487	1	1	0	0
488	1	1	0	0
489	1	1	0	0
490	1	1	0	0
491	1	1	0	0
492	1	1	0	0
493	1	1	0	0
494	1	1	0	0
495	1	1	0	0
496	1	1	0	0
497	1	1	0	0
498	1	1	0	0
499	1	1	0	0
500	1	1	0	0
501	1	1	0	0
502	1	1	0	0
503	1	1	0	0
504	1	1	0	0
505	1	1	0	0
506	1	1	0	0
507	1	1	0	0
508	1	1	0	0
509	1	1	0	0
510	1	1	0	0
511	1	1	0	0
512	1	1	0	0
513	1	1	0	0
514	1	1	0	0
515	1	1	0	0
516	1	1	0	0
517	1	1	0	0
518	1	1	0	0
519	1	1	0	0
520	1	1	0	0
521	1	1	0	0
522	1	1	0	0
523	1	1	0	0
524	1	1	0	0
525	1	1	0	0
526	1	1	0	0
527	1	1	0	0
528	1	1	0	0
529	1	1	0	0
530	1	1	0	0
531	1	1	0	0
532	1	1	0	0
533	1	1	0	0
534	1	1	0	0
535	1	1	0	0
536	1	1	0	0

3858	2	1	1	0
3859	2	1	1	0
3860	2	1	1	0
3861	2	1	1	0
3862	2	1	1	0
3863	2	1	1	0
3864	2	1	1	0
3865	2	1	1	0
3866	2	1	1	0
3867	2	1	1	0
3868	2	1	1	0
3869	2	1	1	0
3870	2	1	1	0
3871	2	1	1	0
3872	2	1	1	0
3873	3	1	1	0
3874	3	1	1	0
3875	3	1	1	0
3876	3	1	1	0
3877	3	1	1	0
3878	3	1	1	0
3879	3	2	1	0
3880	3	2	1	0
3881	3	2	1	0
3882	3	2	1	0
3883	3	2	1	0
3884	3	2	1	0
3885	3	2	1	0
3886	3	1	1	0
3887	3	1	1	0
3888	3	1	1	0
3889	3	1	1	0
3890	3	1	1	0
3891	3	1	1	0
3892	3	1	1	0
3893	3	1	1	0
3894	3	1	1	0
3895	3	1	1	0
3896	3	1	1	0
3897	3	1	1	0
3898	3	1	1	0
3899	3	1	1	0
3900	3	1	1	0
3901	3	1	1	0
3902	3	1	1	0
3903	3	1	1	0
3904	3	1	1	0
3905	3	1	1	0
3906	3	1	1	0
3907	3	2	1	0
3908	3	2	1	0
3909	3	2	1	0
3910	3	2	1	0
3911	3	2	1	0
3912	3	2	1	0
3913	3	2	1	0

468	4	4	0	0
469	4	4	0	0
470	4	4	0	0
471	4	4	0	0
472	4	4	0	0
473	4	4	0	0
474	4	4	0	0
475	4	4	0	0
476	4	4	0	0
477	4	4	0	0
478	4	4	0	0
479	4	4	0	0
480	4	4	0	0
481	4	4	0	0
482	4	4	0	0
483	4	4	0	0
484	4	4	0	0
485	4	4	0	0
486	4	5	0	1
487	4	5	0	1
488	4	5	0	1
489	4	5	0	1
490	4	5	0	1
491	4	5	0	1
492	4	5	0	1
493	4	5	0	1
494	4	4	0	0
495	4	4	0	0
496	4	4	0	0
497	4	4	0	0
498	4	4	0	0
499	4	4	0	0
500	4	4	0	0
501	4	4	0	0
502	4	4	0	0
503	4	4	0	0
504	4	4	0	0
505	4	4	0	0
506	4	4	0	0
507	4	4	0	0
508	4	4	0	0
509	4	4	0	0
510	4	4	0	0
511	4	4	0	0
512	4	4	0	0
513	4	4	0	0
514	4	4	0	0
515	4	4	0	0
516	4	4	0	0
517	4	4	0	0
518	4	4	0	0
519	4	4	0	0
520	4	4	0	0
521	4	4	0	0
522	4	4	0	0
523	4	4	0	0

537	1	1	0	0
538	1	1	0	0
539	1	1	0	0
540	1	1	0	0
541	1	1	0	0
542	1	1	0	0
543	1	1	0	0
544	1	1	0	0
545	1	1	0	0
546	1	1	0	0
547	1	1	0	0
548	1	1	0	0
549	1	1	0	0
550	1	1	0	0
551	1	1	0	0
552	1	1	0	0
553	1	1	0	0
554	1	1	0	0
555	1	1	0	0
556	1	1	0	0
557	1	1	0	0
558	1	1	0	0
559	1	1	0	0
560	1	1	0	0
561	1	1	0	0
562	1	1	0	0
563	1	1	0	0
564	1	1	0	0
565	1	1	0	0
566	1	1	0	0
567	1	1	0	0
568	1	1	0	0
569	1	1	0	0
570	1	1	0	0
571	1	1	0	0
572	1	1	0	0
573	1	1	0	0
574	1	1	0	0
575	1	1	0	0
576	1	1	0	0
577	1	1	0	0
578	1	1	0	0
579	1	1	0	0
580	1	1	0	0
581	1	1	0	0
582	1	1	0	0
583	1	1	0	0
584	1	1	0	0
585	1	1	0	0
586	1	1	0	0
587	1	1	0	0
588	1	1	0	0
589	1	1	0	0
590	1	1	0	0
591	1	1	0	0
592	1	1	0	0

3914	3	2	1	0
3915	3	2	1	0
3916	3	2	1	0
3917	3	2	1	0
3918	3	2	1	0
3919	3	2	1	0
3920	3	2	1	0
3921	3	2	1	0
3922	3	2	1	0
3923	3	2	1	0
3924	3	2	1	0
3925	3	2	1	0
3926	3	2	1	0
3927	3	2	1	0
3928	3	2	1	0
3929	3	2	1	0
3930	3	2	1	0
3931	3	2	1	0
3932	3	2	1	0
3933	3	2	1	0
3934	3	2	1	0
3935	3	2	1	0
3936	3	2	1	0
3937	3	2	1	0
3938	3	2	1	0
3939	3	2	1	0
3940	3	2	1	0
3941	3	2	1	0
3942	3	2	1	0
3943	3	2	1	0
3944	3	2	1	0
3945	3	2	1	0
3946	3	2	1	0
3947	3	2	1	0
3948	3	2	1	0
3949	3	2	1	0
3950	3	2	1	0
3951	3	2	1	0
3952	3	2	1	0
3953	3	2	1	0
3954	3	2	1	0
3955	3	2	1	0
3956	3	2	1	0
3957	3	2	1	0
3958	3	2	1	0
3959	3	2	1	0
3960	3	2	1	0
3961	3	2	1	0
3962	3	2	1	0
3963	3	2	1	0
3964	3	2	1	0
3965	3	2	1	0
3966	3	2	1	0
3967	3	2	1	0
3968	3	2	1	0
3969	3	2	1	0

524	4	4	0	0	593	1	1	0	0	3970	3	2	1	0	580	4	4	0	0	649	1	1	0	0	4026	4	3	1	0
525	4	4	0	0	594	1	1	0	0	3971	3	2	1	0	581	4	4	0	0	650	1	1	0	0	4027	4	3	1	0
526	4	4	0	0	595	1	1	0	0	3972	3	2	1	0	582	4	4	0	0	651	1	1	0	0	4028	4	4	1	1
527	4	4	0	0	596	1	1	0	0	3973	3	2	1	0	583	4	4	0	0	652	1	1	0	0	4029	4	4	1	1
528	4	4	0	0	597	1	1	0	0	3974	3	2	1	0	584	4	4	0	0	653	2	1	0	0	4030	4	5	1	3
529	4	4	0	0	598	1	1	0	0	3975	3	2	1	0	585	4	4	0	0	654	2	1	0	0	4031	4	5	1	3
530	4	4	0	0	599	1	1	0	0	3976	4	2	1	0	586	4	4	0	0	655	2	4	0	2	4032	4	4	1	1
531	4	4	0	0	600	1	1	0	0	3977	4	2	1	0	587	4	4	0	0	656	2	4	0	2	4033	4	4	1	1
532	4	4	0	0	601	1	1	0	0	3978	4	2	1	0	588	4	4	0	0	657	2	2	0	0	4034	4	4	1	1
533	4	4	0	0	602	1	1	0	0	3979	4	2	1	0	589	4	4	0	0	658	2	2	0	0	4035	4	4	1	1
534	4	4	0	0	603	1	1	0	0	3980	4	3	1	0	590	4	4	0	0	659	2	2	0	0	4036	4	3	1	0
535	4	4	0	0	604	1	1	0	0	3981	4	3	1	0	591	4	4	0	0	660	2	2	0	0	4037	4	3	1	0
536	4	4	0	0	605	1	1	0	0	3982	4	3	1	0	592	4	4	0	0	661	2	2	0	0	4038	4	3	1	0
537	4	4	0	0	606	1	1	0	0	3983	4	3	1	0	593	4	4	0	0	662	2	2	0	0	4039	4	3	1	0
538	4	4	0	0	607	1	1	0	0	3984	4	3	1	0	594	4	4	0	0	663	2	2	0	0	4040	4	4	1	1
539	4	4	0	0	608	1	1	0	0	3985	4	3	1	0	595	4	4	0	0	664	2	2	0	0	4041	4	4	1	1
540	4	4	0	0	609	1	1	0	0	3986	4	3	1	0	596	4	4	0	0	665	2	2	0	0	4042	4	4	1	1
541	4	4	0	0	610	1	1	0	0	3987	4	3	1	0	597	4	4	0	0	666	2	2	0	0	4043	4	4	1	1
542	4	4	0	0	611	1	1	0	0	3988	4	3	1	0	598	4	4	0	0	667	2	2	0	0	4044	4	3	1	0
543	4	4	0	0	612	1	1	0	0	3989	4	3	1	0	599	4	4	0	0	668	2	2	0	0	4045	4	3	1	0
544	4	4	0	0	613	1	1	0	0	3990	4	3	1	0	600	4	4	0	0	669	2	2	0	0	4046	4	3	1	0
545	4	4	0	0	614	1	1	0	0	3991	4	4	1	1	601	4	4	0	0	670	2	2	0	0	4047	4	3	1	0
546	4	4	0	0	615	1	1	0	0	3992	4	4	1	1	602	4	4	0	0	671	2	2	0	0	4048	4	3	1	0
547	4	4	0	0	616	1	1	0	0	3993	4	4	1	1	603	4	4	0	0	672	2	2	0	0	4049	4	3	1	0
548	4	4	0	0	617	1	1	0	0	3994	4	4	1	1	604	4	4	0	0	673	2	2	0	0	4050	4	3	1	0
549	4	4	0	0	618	1	1	0	0	3995	4	4	1	1	605	4	4	0	0	674	2	2	0	0	4051	4	3	1	0
550	4	4	0	0	619	1	1	0	0	3996	4	4	1	1	606	4	4	0	0	675	2	2	0	0	4052	4	3	1	0
551	4	4	0	0	620	1	1	0	0	3997	4	2	3	0	607	4	4	0	0	676	2	2	0	0	4053	4	3	1	0
552	4	4	0	0	621	1	1	0	0	3998	4	2	3	0	608	4	4	0	0	677	2	2	0	0	4054	5	4	1	0
553	4	4	0	0	622	1	1	0	0	3999	4	2	3	0	609	4	4	0	0	678	2	2	0	0	4055	5	4	1	0
554	4	4	0	0	623	1	1	0	0	4000	4	2	3	0	610	4	4	0	0	679	2	2	0	0	4056	5	4	1	0
555	4	4	0	0	624	1	1	0	0	4001	4	2	3	0	611	4	4	0	0	680	2	2	0	0	4057	5	3	1	0
556	4	4	0	0	625	1	1	0	0	4002	4	2	3	0	612	4	4	0	0	681	2	2	0	0	4058	5	3	1	0
557	4	4	0	0	626	1	1	0	0	4003	4	2	3	0	613	4	4	0	0	682	2	2	0	0	4059	5	3	1	0
558	4	4	0	0	627	1	1	0	0	4004	4	2	3	0	614	4	4	0	0	683	2	2	0	0	4060	5	3	1	0
559	4	4	0	0	628	1	1	0	0	4005	4	3	2	1	615	4	4	0	0	684	2	2	0	0	4061	5	3	1	0
560	4	4	0	0	629	1	1	0	0	4006	4	4	2	3	616	4	4	0	0	685	2	2	0	0	4062	5	3	1	0
561	4	4	0	0	630	1	1	0	0	4007	4	3	2	1	617	4	4	0	0	686	2	2	0	0	4063	5	3	1	0
562	4	4	0	0	631	1	1	0	0	4008	4	3	2	1	618	4	4	0	0	687	2	2	0	0	4064	5	3	1	0
563	4	4	0	0	632	1	1	0	0	4009	4	3	2	1	619	4	4	0	0	688	2	2	0	0	4065	5	3	1	0
564	4	4	0	0	633	1	1	0	0	4010	4	3	2	1	620	4	4	0	0	689	2	2	0	0	4066	5	3	1	0
565	4	4	0	0	634	1	1	0	0	4011	4	3	2	1	621	4	4	0	0	690	2	2	0	0	4067	5	3	1	0
566	4	4	0	0	635	1	1	0	0	4012	4	3	2	1	622	4	4	0	0	691	2	2	0	0	4068	5	3	1	0
567	4	4	0	0	636	1	1	0	0	4013	4	4	1	0	623	4	4	0	0	692	2	2	0	0	4069	5	3	1	0
568	4	4	0	0	637	1	1	0	0	4014	4	4	1	0	624	4	4	0	0	693	2	2	0	0	4070	5	3	1	0
569	4	4	0	0	638	1	1	0	0	4015	4	4	1	0	625	4	4	0	0	694	2	2	0	0	4071	5	3	1	0
570	4	4	0	0	639	1	1	0	0	4016	4	3	1	0	626	4	4	0	0	695	2	2	0	0	4072	5	3	1	0
571	4	4	0	0	640	1	1	0	0	4017	4	3	1	0	627	4	4	0	0	696	2	2	0	0	4073	5	3	1	0
572	4	4	0	0	641	1	1	0	0	4018	4	3	1	0	628	4	4	0	0	697	2	2	0	0	4074	5	3	1	0
573	4	4	0	0	642	1	1	0	0	4019	4	3	1	0	629	4	4	0	0	698	2	2	0	0	4075	5	3	1	0
574	4	4	0	0	643	1	1	0	0	4020	4	3	1	0	630	4	4	0	0	699	2	2	0	0	4076	5	3	1	0
575	4	4	0	0	644	1	1	0	0	4021	4	3	1	0	631	4	4	0	0	700	2	2	0	0	4077	5	3	1	0
576	4	4	0	0	645	1	1	0	0	4022	4	3	1	0	632	4	4	0	0	701	2	2	0	0	4078	5	3	1	0
577	4	4	0	0	646	1	1	0	0	4023	4	3	1	0	633	4	4	0	0	702	2	2	0	0	4079	5	3	1	0
578	4	4	0	0	647	1	1	0	0	4024	4	3	1	0	634	4	4	0	0	703	2	2	0	0	4080	5	3	1	0
579	4	4	0	0	648	1	1	0	0	4025	4	3	1	0	635	4	4	0	0	704	2	2	0	0	4081	5	3	1	0

636	4	4	0	0	705	2	2	0	0	4082	5	3	1	0	692	5	4	1	0	761	2	2	0	0	4138	5	3	1	0
637	4	4	0	0	706	2	2	0	0	4083	5	3	1	0	693	5	4	1	0	762	2	2	0	0	4139	3	3	0	0
638	4	4	0	0	707	2	2	0	0	4084	5	3	1	0	694	5	4	1	0	763	2	2	0	0	4140	3	3	0	0
639	4	4	0	0	708	2	2	0	0	4085	5	3	1	0	695	5	4	1	0	764	2	2	0	0	4141	3	3	0	0
640	4	4	0	0	709	3	3	0	0	4086	5	3	1	0	696	5	4	1	0	765	2	2	0	0	4142	3	3	0	0
641	4	4	0	0	710	3	3	0	0	4087	5	3	1	0	697	5	4	1	0	766	2	2	0	0	4143	3	3	0	0
642	4	4	0	0	711	3	3	0	0	4088	5	3	1	0	698	5	4	1	0	767	2	2	0	0	4144	3	3	0	0
643	4	4	0	0	712	3	3	0	0	4089	5	4	1	1	699	5	4	1	0	768	2	2	0	0	4145	3	3	0	0
644	4	4	0	0	713	3	3	0	0	4090	5	4	1	1	700	5	4	1	0	769	2	2	0	0	4146	3	3	0	0
645	4	4	0	0	714	3	3	0	0	4091	5	4	1	1	701	5	4	1	0	770	2	2	0	0	4147	3	3	0	0
646	4	4	0	0	715	3	3	0	0	4092	5	4	1	1	702	5	4	1	0	771	2	2	0	0	4148	3	3	0	0
647	4	4	0	0	716	3	3	0	0	4093	5	4	1	1	703	5	4	1	0	772	2	2	0	0	4149	3	3	0	0
648	4	4	0	0	717	3	3	0	0	4094	5	3	1	0	704	5	4	1	0	773	2	2	0	0	4150	3	3	0	0
649	4	4	0	0	718	3	3	0	0	4095	5	3	1	0	705	5	4	1	0	774	2	2	0	0	4151	3	3	0	0
650	4	4	0	0	719	3	3	0	0	4096	5	4	1	0	706	5	4	1	0	775	2	2	0	0	4152	3	3	0	0
651	4	4	0	0	720	3	3	0	0	4097	5	3	1	0	707	5	4	1	0	776	2	2	0	0	4153	3	3	0	0
652	4	4	0	0	721	3	3	0	0	4098	5	3	1	0	708	5	4	1	0	777	2	2	0	0	4154	3	3	0	0
653	4	4	0	0	722	3	3	0	0	4099	5	3	1	0	709	5	4	1	0	778	2	2	0	0	4155	3	4	0	1
654	4	4	0	0	723	3	3	0	0	4100	5	3	1	0	710	5	4	1	0	779	2	2	0	0	4156	3	4	0	1
655	4	4	0	0	724	3	3	0	0	4101	5	3	1	0	711	5	4	1	0	780	2	2	0	0	4157	3	3	0	0
656	4	4	0	0	725	3	3	0	0	4102	5	3	1	0	712	5	4	1	0	781	2	2	0	0	4158	3	3	0	0
657	4	4	0	0	726	3	3	0	0	4103	5	3	1	0	713	5	4	1	0	782	2	2	0	0	4159	3	3	0	0
658	4	4	0	0	727	3	3	0	0	4104	5	3	1	0	714	5	4	1	0	783	2	2	0	0	4160	3	3	0	0
659	4	4	0	0	728	3	3	0	0	4105	5	3	1	0	715	5	4	1	0	784	2	2	0	0	4161	3	3	0	0
660	4	4	0	0	729	3	3	0	0	4106	5	3	1	0	716	5	4	1	0	785	2	2	0	0	4162	3	3	0	0
661	4	4	0	0	730	3	3	0	0	4107	5	3	1	0	717	5	4	1	0	786	2	2	0	0	4163	3	3	0	0
662	4	4	0	0	731	3	3	0	0	4108	5	4	1	0	718	5	4	1	0	787	2	2	0	0	4164	3	3	0	0
663	4	4	0	0	732	3	3	0	0	4109	5	4	1	0	719	5	4	1	0	788	1	1	0	0	4165	3	3	0	0
664	4	4	0	0	733	3	3	0	0	4110	5	3	1	0	720	5	4	1	0	789	1	1	0	0	4166	3	3	0	0
665	4	4	0	0	734	3	3	0	0	4111	5	3	1	0	721	5	4	1	0	790	1	1	0	0	4167	3	3	0	0
666	4	4	0	0	735	3	3	0	0	4112	5	3	1	0	722	5	4	1	0	791	1	1	0	0	4168	3	3	0	0
667	4	4	0	0	736	3	3	0	0	4113	5	3	1	0	723	5	4	1	0	792	1	1	0	0	4169	4	4	0	0
668	4	4	0	0	737	3	3	0	0	4114	5	3	1	0	724	5	4	1	0	793	1	1	0	0	4170	4	4	0	0
669	4	4	0	0	738	3	3	0	0	4115	5	3	1	0	725	5	4	1	0	794	1	1	0	0	4171	4	4	0	0
670	5	4	1	0	739	3	3	0	0	4116	5	3	1	0	726	5	4	1	0	795	1	1	0	0	4172	4	4	0	0
671	5	4	1	0	740	3	3	0	0	4117	5	3	1	0	727	5	4	1	0	796	1	1	0	0	4173	4	4	0	0
672	5	5	0	0	741	3	3	0	0	4118	5	4	1	0	728	5	4	1	0	797	1	1	0	0	4174	4	4	0	0
673	5	5	0	0	742	3	3	0	0	4119	5	4	1	0	729	5	4	1	0	798	1	1	0	0	4175	4	4	0	0
674	5	5	0	0	743	3	3	0	0	4120	5	3	1	0	730	5	4	1	0	799	1	1	0	0	4176	4	4	0	0
675	5	5	0	0	744	3	3	0	0	4121	5	3	1	0	731	5	4	1	0	800	1	1	0	0	4177	4	4	0	0
676	5	5	0	0	745	3	3	0	0	4122	5	3	1	0	732	5	4	1	0	801	1	1	0	0	4178	4	4	0	0
677	5	5	0	0	746	3	3	0	0	4123	5	3	1	0	733	5	4	1	0	802	1	1	0	0	4179	4	4	0	0
678	5	5	0	0	747	3	4	0	1	4124	5	3	1	0	734	5	4	1	0	803	1	1	0	0	4180	4	4	0	0
679	5	5	0	0	748	3	3	0	0	4125	5	3	1	0	735	5	4	1	0	804	1	1	0	0	4181	4	3	0	0
680	5	5	0	0	749	3	3	0	0	4126	5	3	1	0	736	5	4	1	0	805	1	1	0	0	4182	4	3	0	0
681	5	5	0	0	750	2	2	0	0	4127	5	3	1	0	737	5	4	1	0	806	1	1	0	0	4183	4	3	0	0
682	5	4	1	0	751	2	2	0	0	4128	5	3	1	0	738	5	4	1	0	807	2	1	0	0	4184	4	3	0	0
683	5	4	1	0	752	2	2	0	0	4129	5	3	1	0	739	5	4	1	0	808	3	3	1	1	4185	4	3	0	0
684	5	4	1	0	753	2	2	0	0	4130	5	3	1	0	740	5	4	1	0	809	3	2	1	0	4186	4	3	0	0
685	5	4	1	0	754	2	2	0	0	4131	5	3	1	0	741	5	4	1	0	810	3	2	1	0	4187	4	4	0	0
686	5	4	1	0	755	2	2	0	0	4132	5	3	1	0	742	5	4	1	0	811	3	2	1	0	4188	4	4	0	0
687	5	4	1	0	756	2	2	0	0	4133	5	3	1	0	743	5	4	1	0	812	3	2	1	0	4189	4	4	0	0
688	5	4	1	0	757	2	2	0	0	4134	5	3	1	0	744	5	4	1	0	813	3	2	1	0	4190	4	4	0	0
689	5	4	1	0	758	2	2	0	0	4135	5	3	1	0	745	5	4	1	0	814	3	2	1	0	4191	4	4	0	0
690	5	4	1	0	759	2	2	0	0	4136	5	3	1	0	746	5	4	1	0	815	3	2	1	0	4192	4	4	0	0
691	5	4	1	0	760	2	2	0	0	4137	5	3	1	0	747	5	4	1	0	816	3	2	1	0	4193	4	4	0	0

748	5	4	1	0
749	5	4	1	0
750	5	4	1	0
751	5	4	1	0
752	5	4	1	0
753	5	4	1	0
754	5	4	1	0
755	5	4	1	0
756	5	4	1	0
757	5	4	1	0
758	5	4	1	0
759	5	4	1	0
760	5	4	1	0
761	5	4	1	0
762	5	4	1	0
763	5	4	1	0
764	5	4	1	0
765	5	4	1	0
766	5	4	1	0
767	5	4	1	0
768	5	4	1	0
769	5	4	1	0
770	5	4	1	0
771	5	4	1	0
772	5	4	1	0
773	5	4	1	0
774	5	4	1	0
775	5	4	1	0
776	5	4	1	0
777	5	4	1	0
778	5	4	1	0
779	5	4	1	0
780	5	4	1	0
781	5	4	1	0
782	5	4	1	0
783	5	4	1	0
784	5	4	1	0
785	5	4	1	0
786	5	4	1	0
787	5	4	1	0
788	5	4	1	0
789	5	4	1	0
790	5	4	1	0
791	5	4	1	0
792	5	4	1	0
793	5	4	1	0
794	5	4	1	0
795	5	4	1	0
796	5	4	1	0
797	5	4	1	0
798	5	4	1	0
799	5	4	1	0
800	5	4	1	0
801	5	4	1	0
802	5	4	1	0
803	5	4	1	0

817	3	2	1	0
818	3	2	1	0
819	3	2	1	0
820	3	2	1	0
821	3	2	1	0
822	3	2	1	0
823	3	2	1	0
824	3	2	1	0
825	3	2	1	0
826	3	2	1	0
827	3	2	1	0
828	3	2	1	0
829	3	2	1	0
830	3	2	1	0
831	3	2	1	0
832	3	2	1	0
833	3	2	1	0
834	3	2	1	0
835	3	2	1	0
836	3	2	1	0
837	3	2	1	0
838	3	2	1	0
839	3	2	1	0
840	3	2	1	0
841	3	2	1	0
842	3	2	1	0
843	3	2	1	0
844	3	2	1	0
845	3	2	1	0
846	3	2	1	0
847	3	2	1	0
848	3	2	1	0
849	3	2	1	0
850	3	2	1	0
851	3	2	1	0
852	3	2	1	0
853	3	2	1	0
854	3	2	1	0
855	3	2	1	0
856	3	2	1	0
857	3	2	1	0
858	3	2	1	0
859	3	2	1	0
860	3	2	1	0
861	3	2	1	0
862	3	2	1	0
863	3	2	1	0
864	3	2	1	0
865	3	2	1	0
866	3	2	1	0
867	3	2	1	0
868	3	2	1	0
869	3	2	1	0
870	3	1	1	0
871	3	1	1	0
872	3	1	1	0

4194	4	4	0	0
4195	4	4	0	0
4196	4	4	0	0
4197	4	4	0	0
4198	4	4	0	0
4199	4	4	0	0
4200	4	4	0	0
4201	4	4	0	0
4202	4	4	0	0
4203	6	3	1	0
4204	6	3	1	0
4205	6	5	1	0
4206	6	5	1	0
4207	6	5	1	0
4208	6	5	1	0
4209	6	5	1	0
4210	6	5	1	0
4211	6	5	1	0
4212	6	5	1	0
4213	6	5	1	0
4214	6	5	1	0
4215	6	5	1	0
4216	6	6	0	0
4217	6	6	0	0
4218	6	6	0	0
4219	6	5	1	0
4220	6	5	1	0
4221	6	5	1	0
4222	6	5	1	0
4223	6	5	1	0
4224	6	5	1	0
4225	6	5	1	0
4226	6	5	1	0
4227	6	5	1	0
4228	6	5	1	0
4229	6	5	1	0
4230	6	5	1	0
4231	6	5	1	0
4232	6	5	1	0
4233	6	5	1	0
4234	6	5	1	0
4235	6	5	1	0
4236	6	5	1	0
4237	6	6	0	0
4238	6	6	0	0
4239	6	6	0	0
4240	6	6	0	0
4241	6	6	0	0
4242	6	6	0	0
4243	6	6	0	0
4244	6	6	0	0
4245	6	6	0	0
4246	6	6	0	0
4247	6	6	0	0
4248	6	6	0	0
4249	6	6	0	0

804	5	4	1	0
805	5	4	1	0
806	5	4	1	0
807	5	4	1	0
808	5	4	1	0
809	5	4	1	0
810	5	4	1	0
811	5	4	1	0
812	5	4	1	0
813	5	4	1	0
814	5	4	1	0
815	5	4	1	0
816	5	4	1	0
817	5	4	1	0
818	5	4	1	0
819	5	4	1	0
820	5	4	1	0
821	5	4	1	0
822	5	4	1	0
823	5	4	1	0
824	5	4	1	0
825	5	4	1	0
826	5	4	1	0
827	4	3	1	0
828	4	3	1	0
829	4	3	1	0
830	5	4	1	0
831	5	4	1	0
832	5	4	1	0
833	5	4	1	0
834	5	4	1	0
835	5	4	1	0
836	5	4	1	0
837	5	4	1	0
838	5	4	1	0
839	5	4	1	0
840	5	4	1	0
841	5	4	1	0
842	6	5	1	0
843	6	5	1	0
844	6	5	1	0
845	6	5	1	0
846	6	5	1	0
847	6	5	1	0
848	6	5	1	0
849	6	5	1	0
850	6	5	1	0
851	6	5	1	0
852	6	5	1	0
853	6	5	1	0
854	6	5	1	0
855	6	5	1	0
856	6	5	1	0
857	6	5	1	0
858	6	5	1	0
859	6	5	1	0

873	3	1	1	0
874	3	1	1	0
875	3	1	1	0
876	3	1	1	0
877	3	1	1	0
878	3	1	1	0
879	3	1	1	0
880	3	1	1	0
881	3	1	1	0
882	3	1	1	0
883	3	1	1	0
884	3	1	1	0
885	3	1	1	0
886	3	1	1	0
887	3	1	1	0
888	3	1	1	0
889	3	1	1	0
890	3	1	1	0
891	3	1	1	0
892	3	1	1	0
893	3	1	1	0
894	3	1	1	0
895	3	1	1	0
896	3	1	1	0
897	3	1	1	0
898	3	1	1	0
899	3	1	1	0
900	3	1	1	0
901	3	1	1	0
902	3	1	1	0
903	3	1	1	0
904	3	1	1	0
905	3	1	1	0
906	3	1	1	0
907	3	1	1	0
908	3	1	1	0
909	3	1	1	0
910	3	1	1	0
911	3	1	1	0
912	3	1	1	0
913	3	1	1	0
914	3	1	1	0
915	3	1	1	0
916	3	1	1	0
917	3	1	1	0
918	3	1	1	0
919	3	1	1	0
920	3	1	1	0
921	3	1	1	0
922	3	1	1	0
923	3	1	1	0
924	3	1	1	0
925	3	1	1	0
926	3	1	1	0
927	3	1	1	0
928	3	1	1	0

4250	6	6	0	0
4251	6	6	0	0
4252	6	6	0	0
4253	6	8	0	2
4254	6	8	0	2
4255	6	4	1	0
4256	6	4	1	0
4257	6	4	1	0
4258	6	4	1	0
4259	6	4	1	0
4260	6	4	1	0
4261	6	4	1	0
4262	6	4	1	0
4263	6	4	1	0
4264	6	4	1	0
4265	6	4	1	0
4266	5	4	1	1
4267	5	4	1	0
4268	5	4	1	0
4269	5	4	1	0
4270	5	4	1	0
4271	5	4	1	0
4272	5	4	1	0
4273	5	4	1	0
4274	5	4	1	0
4275	5	4	1	0
4276	5	4	1	0
4277	5	4	1	0
4278	5	4	1	0
4279	5	4	1	0
4280	5	4	1	0
4281	5	5	0	0
4282	5	5	0	0
4283	5	5	0	0
4284	5	5	0	0
4285	5	5	0	0
4286	5	5	0	0
4287	5	5	0	0
4288	5	5	0	0
4289	5	5	0	0
4290	5	5	0	0
4291	5	5	0	0
4292	5	5	0	0
4293	5	5	0	0
4294	5	5	0	0
4295	5	5	0	0
4296	5	5	0	0
4297	5	5	0	0
4298	5	5	0	0
4299	5	5	0	0
4300	5	4	1	0
4301	5	4	1	0
4302	5	4	1	0
4303	5	4	1	0
4304	5	4	1	0
4305	5	4	1	0

860	6	5	1	0	929	3	1	1	0	4306	5	4	1	0
861	6	5	1	0	930	3	2	1	0	4307	5	4	1	0
862	6	5	1	0	931	3	2	1	0	4308	5	4	1	0
863	6	5	1	0	932	3	2	1	0	4309	5	4	1	0
864	6	5	1	0	933	3	2	1	0	4310	5	4	1	0
865	6	5	1	0	934	3	2	1	0	4311	5	4	1	0
866	6	5	1	0	935	3	2	1	0	4312	5	4	1	0
867	6	5	1	0	936	3	2	1	0	4313	5	3	2	0
868	6	5	1	0	937	3	2	1	0	4314	5	3	2	0
869	6	5	1	0	938	3	2	1	0	4315	5	3	2	0
870	6	5	1	0	939	3	2	1	0	4316	5	3	2	0
871	6	5	1	0	940	3	2	1	0	4317	5	3	2	0
872	6	5	1	0	941	3	2	1	0	4318	5	4	1	0
873	6	5	1	0	942	3	2	1	0	4319	5	4	1	0
874	6	5	1	0	943	3	2	1	0	4320	5	4	1	0
875	6	5	1	0	944	3	2	1	0	4321	5	4	1	0
876	6	5	1	0	945	3	2	1	0	4322	5	4	1	0
877	6	5	1	0	946	3	2	1	0	4323	5	4	1	0
878	6	5	1	0	947	3	2	1	0	4324	5	4	1	0
879	6	5	1	0	948	3	2	1	0	4325	5	4	1	0
880	6	5	1	0	949	3	2	1	0	4326	5	4	1	0
881	6	5	1	0	950	3	2	1	0	4327	5	4	1	0
882	6	5	1	0	951	3	2	1	0	4328	5	4	1	0
883	6	5	1	0	952	3	2	1	0	4329	5	4	1	0
884	6	5	1	0	953	3	2	1	0	4330	5	4	1	0
885	6	5	1	0	954	3	2	1	0	4331	5	4	1	0
886	6	5	1	0	955	3	2	1	0	4332	5	4	1	0
887	6	5	1	0	956	3	2	1	0	4333	5	4	1	0
888	6	5	1	0	957	3	2	1	0	4334	5	4	1	0
889	6	5	1	0	958	3	2	1	0	4335	5	4	1	0
890	6	5	1	0	959	3	2	1	0	4336	5	4	1	0
891	6	5	1	0	960	3	2	1	0	4337	5	4	1	0
892	6	5	1	0	961	3	2	1	0	4338	5	4	1	0
893	6	5	1	0	962	3	2	1	0	4339	5	4	1	0
894	6	5	1	0	963	3	2	1	0	4340	5	4	1	0
895	6	5	1	0	964	3	2	1	0	4341	5	4	1	0
896	6	5	1	0	965	3	2	1	0	4342	5	4	1	0
897	6	5	1	0	966	3	2	1	0	4343	5	4	1	0
898	6	5	1	0	967	3	2	1	0	4344	5	4	1	0
899	6	5	1	0	968	3	2	1	0	4345	5	4	1	0
900	6	5	1	0	969	3	2	1	0	4346	5	4	1	0
901	6	5	1	0	970	3	2	1	0	4347	5	4	1	0
902	6	5	1	0	971	3	2	1	0	4348	5	4	1	0
903	6	5	1	0	972	3	2	1	0	4349	5	4	1	0
904	6	5	1	0	973	3	2	1	0	4350	5	4	1	0
905	6	5	1	0	974	3	2	1	0	4351	5	4	1	0
906	6	5	1	0	975	3	2	1	0	4352	5	4	1	0
907	6	5	1	0	976	3	2	1	0	4353	5	4	1	0
908	6	5	1	0	977	3	2	1	0	4354	5	4	1	0
909	6	5	1	0	978	3	2	1	0	4355	5	4	1	0
910	6	5	1	0	979	3	2	1	0	4356	5	4	1	0
911	6	5	1	0	980	3	2	1	0	4357	5	4	1	0
912	6	5	1	0	981	3	2	1	0	4358	5	5	0	0
913	6	5	1	0	982	3	2	1	0	4359	5	5	0	0
914	6	5	1	0	983	3	2	1	0	4360	5	5	0	0
915	6	5	1	0	984	3	2	1	0	4361	5	5	0	0

916	6	5	1	0	985	3	2	1	0	4362	5	4	0	0
917	6	5	1	0	986	3	2	1	0	4363	5	4	0	0
918	6	5	1	0	987	3	2	1	0	4364	5	5	0	1
919	6	5	1	0	988	3	2	1	0	4365	5	4	0	0
920	6	5	1	0	989	3	2	1	0	4366	5	5	0	1
921	6	5	1	0	990	3	2	1	0	4367	5	5	0	1
922	6	5	1	0	991	3	2	1	0	4368	5	4	0	0
923	6	5	1	0	992	3	2	1	0	4369	4	4	0	0
924	6	5	1	0	993	3	2	1	0	4370	4	4	0	0
925	6	5	1	0	994	3	2	1	0	4371	4	4	0	0
926	6	5	1	0	995	3	2	1	0	4372	4	4	0	0
927	6	5	1	0	996	3	2	1	0	4373	4	4	0	0
928	6	5	1	0	997	3	2	1	0	4374	4	4	0	0
929	6	5	1	0	998	3	2	1	0	4375	4	4	0	0
930	6	5	1	0	999	3	2	1	0	4376	4	4	0	0
931	6	5	1	0	1000	3	2	1	0	4377	4	4	0	0
932	6	5	1	0	1001	3	2	1	0	4378	4	5	0	1
933	6	5	1	0	1002	3	2	1	0	4379	4	5	0	1
934	6	5	1	0	1003	3	2	1	0	4380	4	4	0	0
935	6	5	1	0	1004	3	2	1	0	4381	4	4	0	0
936	6	5	1	0	1005	3	2	1	0	4382	4	4	0	0
937	6	5	1	0	1006	3	2	1	0	4383	4	4	0	0
938	6	5	1	0	1007	3	2	1	0	4384	4	4	0	0
939	6	5	1	0	1008	3	2	1	0	4385	4	4	0	0
940	6	5	1	0	1009	3	2	1	0	4386	4	4	0	0
941	6	5	1	0	1010	3	2	1	0	4387	4	3	0	0
942	6	5	1	0	1011	3	2	1	0	4388	4	3	0	0
943	6	5	1	0	1012	3	2	1	0	4389	4	3	0	0
944	6	5	1	0	1013	3	2	1	0	4390	4	3	0	0
945	6	5	1	0	1014	3	2	1	0	4391	4	3	0	0
946	6	5	1	0	1015	3	2	1	0	4392	3	3	0	0
947	6	5	1	0	1016	3	2	1	0	4393	3	3	0	0
948	6	5	1	0	1017	3	2	1	0	4394	3	3	0	0
949	6	5	1	0	1018	3	2	1	0	4395	3	3	0	0
950	6	5	1	0	1019	3	2	1	0	4396	3	3	0	0
951	6	5	1	0	1020	3	2	1	0	4397	3	3	0	0
952	6	5	1	0	1021	2	1	1	0	4398	3	3	0	0
953	6	5	1	0	1022	2	1	1	0	4399	3	3	0	0
954	6	5	1	0	1023	2	1	1	0	4400	3	3	0	0
955	6	5	1	0	1024	2	1	1	0	4401	3	3	0	0
956	6	5	1	0	1025	2	1	1	0	4402	3	3	0	0
957	6	5	1	0	1026	2	1	1	0	4403	3	3	0	0
958	6	5	1	0	1027	2	1	1	0	4404	3	3	0	0
959	6	5	1	0	1028	2	1	1	0	4405	3	3	0	0
960	6	5	1	0	1029	2	1	1	0	4406	3	3	0	0
961	6	5	1	0	1030	2	1	1	0	4407	3	3	0	0
962	6	5	1	0	1031	2	1	1	0	4408	3	3	0	0
963	6	5	1	0	1032	2	1	1	0	4409	3	3	0	0
964	6	5	1	0	1033	2	1	1	0	4410	3	3	0	0
965	6	5	1	0	1034	2	1	1	0	4411	3	3	0	0
966	6	5	1	0	1035	2	1	1	0	4412	3	3	0	0
967	6	5	1	0	1036	2	1	1	0	4413	3	3	0	0
968	6	5	1	0	1037	2	1	1	0	4414	3	3	0	0
969	6	5	1	0	1038	2	1	1	0	4415	3	3	0	0
970	6	5	1	0	1039	2	1	1	0	4416	3	3	0	0
971	6	5	1	0	1040	2	1	1	0	4417	3	3	0	0

972	6	5	1	0
973	6	5	1	0
974	6	5	1	0
975	6	5	1	0
976	6	5	1	0
977	6	5	1	0
978	6	5	1	0
979	6	5	1	0
980	6	5	1	0
981	6	5	1	0
982	6	5	1	0
983	6	5	1	0
984	6	5	1	0
985	6	5	1	0
986	6	5	1	0
987	6	5	1	0
988	6	5	1	0
989	6	6	0	0
990	6	6	0	0
991	6	6	0	0
992	6	6	0	0
993	6	6	0	0
994	6	6	0	0
995	6	6	0	0
996	6	6	0	0
997	6	6	0	0
998	6	6	0	0
999	6	6	0	0
1000	6	6	0	0
1001	5	5	0	0
1002	5	5	0	0
1003	5	5	0	0
1004	5	5	0	0
1005	5	5	0	0
1006	5	5	0	0
1007	5	5	0	0
1008	5	5	0	0
1009	5	5	0	0
1010	5	5	0	0
1011	5	5	0	0
1012	5	5	0	0
1013	5	5	0	0
1014	5	5	0	0
1015	5	5	0	0
1016	6	6	0	0
1017	6	6	0	0
1018	6	6	0	0
1019	6	6	0	0
1020	6	6	0	0
1021	6	6	0	0
1022	6	6	0	0
1023	6	6	0	0
1024	6	6	0	0
1025	6	6	0	0
1026	6	6	0	0
1027	6	6	0	0

1041	2	1	1	0
1042	2	1	1	0
1043	2	1	1	0
1044	2	1	1	0
1045	2	1	1	0
1046	2	1	1	0
1047	2	1	1	0
1048	2	1	1	0
1049	2	1	1	0
1050	2	1	1	0
1051	2	1	1	0
1052	2	1	1	0
1053	2	1	1	0
1054	2	1	1	0
1055	2	1	1	0
1056	2	1	1	0
1057	2	1	1	0
1058	2	1	1	0
1059	2	1	1	0
1060	2	1	1	0
1061	2	1	1	0
1062	2	2	1	1
1063	2	2	1	1
1064	2	1	1	0
1065	2	1	1	0
1066	2	1	1	0
1067	2	1	1	0
1068	2	1	1	0
1069	2	1	1	0
1070	2	1	1	0
1071	2	1	1	0
1072	2	1	1	0
1073	2	1	1	0
1074	2	1	1	0
1075	2	2	1	1
1076	2	2	1	1
1077	2	2	1	1
1078	2	2	1	1
1079	2	1	1	0
1080	2	2	0	0
1081	2	2	0	0
1082	2	2	0	0
1083	2	2	0	0
1084	2	2	0	0
1085	2	2	0	0
1086	2	2	0	0
1087	2	1	1	0
1088	2	2	1	1
1089	2	4	0	2
1090	2	4	0	2
1091	2	4	0	2
1092	2	4	0	2
1093	2	4	0	2
1094	2	4	0	2
1095	2	4	0	2
1096	2	4	0	2

4418	3	3	0	0
4419	3	3	0	0
4420	3	3	0	0
4421	3	3	0	0
4422	3	3	0	0
4423	3	3	0	0
4424	3	3	0	0
4425	3	3	0	0
4426	3	3	0	0
4427	3	3	0	0
4428	3	3	0	0
4429	3	3	0	0
4430	3	3	0	0
4431	3	3	0	0
4432	3	3	0	0
4433	3	3	0	0
4434	3	3	0	0
4435	3	3	0	0
4436	3	3	0	0
4437	3	3	0	0
4438	3	4	0	1
4439	3	3	0	0
4440	3	3	0	0
4441	3	3	0	0
4442	3	3	0	0
4443	3	3	0	0
4444	3	3	0	0
4445	3	3	0	0
4446	3	3	0	0
4447	3	3	0	0
4448	3	3	0	0
4449	3	3	0	0
4450	3	3	0	0
4451	3	3	0	0
4452	3	3	0	0
4453	3	3	0	0
4454	3	3	0	0
4455	3	3	0	0
4456	3	3	0	0
4457	3	3	0	0
4458	3	3	0	0
4459	3	3	0	0
4460	3	3	0	0
4461	3	3	0	0
4462	3	3	0	0
4463	3	3	0	0
4464	3	3	0	0
4465	3	3	0	0
4466	3	3	0	0
4467	3	3	0	0
4468	3	3	0	0
4469	3	3	0	0
4470	3	3	0	0
4471	3	3	0	0
4472	3	3	0	0
4473	3	3	0	0

1028	6	6	0	0
1029	6	6	0	0
1030	6	6	0	0
1031	6	6	0	0
1032	6	6	0	0
1033	6	6	0	0
1034	6	6	0	0
1035	6	6	0	0
1036	6	6	0	0
1037	6	6	0	0
1038	6	6	0	0
1039	6	6	0	0
1040	6	6	0	0
1041	6	6	0	0
1042	6	6	0	0
1043	6	6	0	0
1044	6	6	0	0
1045	6	6	0	0
1046	6	6	0	0
1047	6	6	0	0
1048	6	6	0	0
1049	6	6	0	0
1050	6	6	0	0
1051	6	6	0	0
1052	6	6	0	0
1053	6	6	0	0
1054	6	6	0	0
1055	6	6	0	0
1056	6	6	0	0
1057	6	6	0	0
1058	6	6	0	0
1059	6	6	0	0
1060	6	6	0	0
1061	6	6	0	0
1062	6	6	0	0
1063	6	6	0	0
1064	6	6	0	0
1065	6	6	0	0
1066	6	6	0	0
1067	6	6	0	0
1068	6	6	0	0
1069	6	5	1	0
1070	6	5	1	0
1071	6	5	1	0
1072	6	5	1	0
1073	6	5	1	0
1074	6	5	1	0
1075	6	5	1	0
1076	6	5	1	0
1077	6	5	1	0
1078	6	5	1	0
1079	6	4	2	0
1080	6	4	2	0
1081	6	4	2	0
1082	6	4	2	0
1083	6	4	2	0

1097	2	3	0	1
1098	2	2	0	0
1099	2	2	0	0
1100	2	2	0	0
1101	2	2	0	0
1102	2	2	0	0
1103	2	2	0	0
1104	2	2	0	0
1105	2	2	1	1
1106	2	2	1	1
1107	2	2	1	1
1108	2	2	1	1
1109	2	2	1	1
1110	2	3	1	2
1111	2	3	1	2
1112	2	3	0	1
1113	2	3	0	1
1114	2	3	0	1
1115	2	3	0	1
1116	2	3	0	1
1117	2	3	0	1
1118	2	3	0	1
1119	2	3	0	1
1120	2	3	0	1
1121	2	3	0	1
1122	2	3	0	1
1123	2	3	0	1
1124	2	1	1	0
1125	2	1	1	0
1126	2	2	0	0
1127	2	2	0	0
1128	2	2	0	0
1129	2	3	0	1
1130	2	3	0	1
1131	2	3	0	1
1132	2	3	0	1
1133	2	3	0	1
1134	2	3	0	1
1135	2	3	0	1
1136	2	2	0	0
1137	2	2	0	0
1138	2	2	0	0
1139	2	4	0	2
1140	2	3	0	1
1141	2	2	1	1
1142	2	2	0	0
1143	2	2	0	0
1144	2	2	0	0
1145	2	2	0	0
1146	2	2	0	0
1147	2	2	0	0
1148	2	2	0	0
1149	2	2	0	0
1150	2	2	0	0
1151	2	2	0	0
1152	2	2	0	0

4474	3	3	0	1
4475	3	3	0	1
4476	3	2	0	0
4477	3	2	0	0
4478	3	2	0	0
4479	3	2	0	0
4480	3	2	0	0
4481	3	2	0	0
4482	3	2	0	0
4483	3	2	0	0
4484	3	2	0	0
4485	3	2	0	0
4486	3	2	0	0
4487	3	2	0	0
4488	3	2	0	0
4489	3	2	0	0
4490	3	3	0	0
4491	3	3	0	0
4492	3	3	0	0
4493	3	3	0	0
4494	3	3	0	0
4495	3	3	0	0
4496	3	2	0	0
4497	3	2	0	0
4498	3	2	0	0
4499	3	2	0	0
4500	3	2	0	0
4501	2	2	0	0
4502	2	2	0	0
4503	2	2	0	0
4504	2	2	0	0
4505	2	2	0	0
4506	2	2	0	0
4507	2	2	0	0
4508	2	2	0	0
4509	2	2	0	0
4510	2	2	0	0
4511	2	2	0	0
4512	2	2	0	0
4513	2	2	0	0
4514	2	2	0	0
4515	2	2	0	0
4516	2	2	0	0
4517	2	2	0	0
4518	2	2	0	0
4519	2	2	0	0
4520	2	2	0	0
4521	2	2	0	0
4522	2	2	0	0
4523	2	2	0	0
4524	2	2	0	0
4525	2	2	0	0
4526	2	2	0	0
4527	2	2	0	0
4528	2	2	0	0
4529	2	2	0	0

1084	6	4	2	0	1153	2	2	0	0	4530	2	2	0	0	1140	6	4	2	0	1209	2	3	0	1	4586	3	3	0	0
1085	6	4	2	0	1154	2	2	0	0	4531	2	2	0	0	1141	6	4	2	0	1210	2	1	0	0	4587	3	3	0	0
1086	6	4	2	0	1155	2	2	0	0	4532	2	2	0	0	1142	6	4	2	0	1211	2	1	0	0	4588	3	3	0	0
1087	6	4	2	0	1156	2	2	0	0	4533	2	2	0	0	1143	6	4	2	0	1212	2	1	0	0	4589	3	3	0	0
1088	6	4	2	0	1157	2	2	0	0	4534	2	2	0	0	1144	6	4	2	0	1213	2	2	0	0	4590	3	3	0	0
1089	6	4	2	0	1158	2	2	0	0	4535	2	2	0	0	1145	6	4	2	0	1214	2	2	0	0	4591	3	3	0	0
1090	6	4	2	0	1159	2	2	0	0	4536	2	1	0	0	1146	6	5	1	0	1215	2	2	0	0	4592	3	3	0	0
1091	6	5	1	0	1160	2	2	0	0	4537	2	2	0	0	1147	6	5	1	0	1216	2	2	0	0	4593	3	3	0	0
1092	6	5	1	0	1161	2	2	0	0	4538	2	2	0	0	1148	6	5	1	0	1217	2	2	0	0	4594	3	3	0	0
1093	6	5	1	0	1162	2	2	0	0	4539	2	2	0	0	1149	6	5	1	0	1218	2	2	0	0	4595	3	3	0	0
1094	6	5	1	0	1163	2	2	0	0	4540	2	2	0	0	1150	6	5	1	0	1219	2	2	0	0	4596	3	3	0	0
1095	6	5	1	0	1164	2	2	0	0	4541	2	1	0	0	1151	6	5	1	0	1220	2	2	0	0	4597	3	3	0	0
1096	6	5	1	0	1165	2	2	0	0	4542	2	1	0	0	1152	6	5	1	0	1221	2	2	0	0	4598	3	3	0	0
1097	6	5	1	0	1166	2	2	0	0	4543	2	2	0	0	1153	6	5	1	0	1222	2	2	0	0	4599	3	3	0	0
1098	6	5	1	0	1167	2	2	0	0	4544	2	2	0	0	1154	6	5	1	0	1223	2	1	0	0	4600	3	3	0	0
1099	6	5	1	0	1168	2	3	2	1	4545	2	2	0	0	1155	6	5	1	0	1224	2	1	0	0	4601	3	3	0	0
1100	6	5	1	0	1169	2	3	2	1	4546	2	2	0	0	1156	6	5	1	0	1225	2	1	0	0	4602	3	3	0	0
1101	6	5	1	0	1170	2	3	2	1	4547	2	2	0	0	1157	6	5	1	0	1226	2	1	0	0	4603	3	3	0	0
1102	6	5	1	0	1171	2	2	0	0	4548	2	2	0	0	1158	6	5	1	0	1227	2	2	0	0	4604	3	3	0	0
1103	6	5	1	0	1172	2	2	0	0	4549	2	2	0	0	1159	6	5	1	0	1228	2	2	0	0	4605	3	3	0	0
1104	6	5	1	0	1173	2	2	0	0	4550	2	2	0	0	1160	6	5	1	0	1229	2	2	0	0	4606	3	3	0	0
1105	6	5	1	0	1174	2	2	0	0	4551	2	2	0	0	1161	6	5	1	0	1230	2	2	0	0	4607	3	3	0	0
1106	6	5	1	0	1175	2	2	0	0	4552	3	2	0	0	1162	6	5	1	0	1231	1	1	0	0	4608	3	3	0	0
1107	6	5	1	0	1176	2	3	0	1	4553	3	2	0	0	1163	6	5	1	0	1232	1	1	0	0	4609	3	3	0	0
1108	6	5	1	0	1177	2	2	0	0	4554	3	1	0	0	1164	6	5	1	0	1233	1	1	0	0	4610	3	3	0	0
1109	6	5	1	0	1178	2	2	0	0	4555	3	2	0	0	1165	6	5	1	0	1234	1	1	0	0	4611	3	3	0	1
1110	6	5	1	0	1179	2	2	0	0	4556	3	1	0	0	1166	6	5	1	0	1235	1	1	0	0	4612	3	3	0	1
1111	6	5	1	0	1180	2	2	0	0	4557	3	1	0	0	1167	6	5	1	0	1236	1	1	0	0	4613	3	3	0	1
1112	6	5	1	0	1181	2	2	0	0	4558	3	3	0	0	1168	6	5	1	0	1237	1	1	0	0	4614	3	2	0	0
1113	6	5	1	0	1182	2	2	0	0	4559	3	3	0	0	1169	6	5	1	0	1238	1	1	0	0	4615	3	2	0	0
1114	6	5	1	0	1183	2	2	0	0	4560	3	3	0	0	1170	6	5	1	0	1239	1	1	0	0	4616	3	2	0	0
1115	6	5	1	0	1184	2	2	0	0	4561	3	2	0	0	1171	6	5	1	0	1240	1	1	0	0	4617	3	2	0	0
1116	6	5	1	0	1185	2	3	0	1	4562	3	3	0	0	1172	6	5	1	0	1241	1	1	0	0	4618	2	2	0	0
1117	6	5	1	0	1186	2	2	0	0	4563	3	3	0	0	1173	6	5	1	0	1242	1	1	0	0	4619	2	2	0	0
1118	6	5	1	0	1187	2	2	0	0	4564	3	3	0	1	1174	6	5	1	0	1243	1	0	0	0	4620	2	2	0	0
1119	6	5	1	0	1188	2	2	0	0	4565	3	3	0	1	1175	6	5	1	0	1244	1	1	0	0	4621	3	2	0	0
1120	6	5	1	0	1189	2	2	0	0	4566	3	2	0	0	1176	6	5	1	0	1245	1	1	0	0	4622	3	2	0	0
1121	6	5	1	0	1190	2	2	0	0	4567	3	2	0	0	1177	6	5	1	0	1246	1	1	0	0	4623	3	3	0	0
1122	6	5	1	0	1191	2	2	0	0	4568	3	2	0	0	1178	6	5	1	0	1247	1	1	0	0	4624	3	3	0	0
1123	6	5	1	0	1192	2	2	0	0	4569	3	2	0	0	1179	6	5	1	0	1248	1	1	0	0	4625	3	3	0	0
1124	6	5	1	0	1193	2	2	0	0	4570	3	2	0	0	1180	6	5	1	0	1249	1	1	0	0	4626	3	3	0	0
1125	6	5	1	0	1194	2	2	0	0	4571	3	2	0	0	1181	6	5	1	0	1250	1	1	0	0	4627	3	3	0	0
1126	6	4	2	0	1195	2	2	0	0	4572	3	2	0	0	1182	6	5	1	0	1251	1	1	0	0	4628	3	3	0	0
1127	6	4	2	0	1196	2	2	0	0	4573	3	3	0	0	1183	6	5	1	0	1252	1	1	0	0	4629	3	3	0	0
1128	6	4	2	0	1197	2	2	0	0	4574	3	3	0	0	1184	6	5	1	0	1253	1	1	0	0	4630	3	3	0	0
1129	6	4	2	0	1198	2	2	0	0	4575	3	3	0	0	1185	6	5	1	0	1254	1	1	0	0	4631	3	3	0	0
1130	6	3	3	0	1199	2	2	0	0	4576	3	3	0	0	1186	6	5	1	0	1255	1	1	0	0	4632	3	3	0	0
1131	6	3	3	0	1200	2	2	0	0	4577	3	3	0	0	1187	6	5	1	0	1256	1	1	0	0	4633	3	3	0	0
1132	6	3	3	0	1201	2	2	0	0	4578	3	3	0	0	1188	6	5	1	0	1257	1	1	0	0	4634	3	3	0	0
1133	6	4	2	0	1202	2	2	0	0	4579	3	3	0	0	1189	6	5	1	0	1258	1	1	0	0	4635	3	3	0	0
1134	6	4	2	0	1203	2	2	0	0	4580	3	3	0	0	1190	6	5	1	0	1259	1	1	0	0	4636	3	3	0	0
1135	6	4	2	0	1204	2	2	0	0	4581	3	3	0	0	1191	6	5	1	0	1260	1	1	0	0	4637	3	3	0	0
1136	6	4	2	0	1205	2	2	0	0	4582	3	3	0	0	1192	6	5	1	0	1261	1	1	0	0	4638	3	3	0	0
1137	6	4	2	0	1206	2	2	0	0	4583	3	3	0	0	1193	6	5	1	0	1262	1	1	0	0	4639	3	3	0	0
1138	6	4	2	0	1207	2	2	0	0	4584	3	3	0	0	1194	6	5	1	0	1263	1	1	0	0	4640	3	3	0	0
1139	6	4	2	0	1208	2	2	0	0	4585	3	3	0	0	1195	6	5	1	0	1264	0	0	0	0	4641	3	3	0	0

1196	6	5	1	0	1265	0	0	0	0	4642	3	3	0	0
1197	6	5	1	0	1266	0	0	0	0	4643	3	3	0	0
1198	6	5	1	0	1267	0	0	0	0	4644	3	2	1	0
1199	6	5	1	0	1268	0	0	0	0	4645	3	2	1	0
1200	6	5	1	0	1269	0	0	0	0	4646	3	2	1	0
1201	6	5	1	0	1270	0	0	0	0	4647	4	2	1	0
1202	6	5	1	0	1271	0	0	0	0	4648	4	2	1	0
1203	6	5	1	0	1272	0	0	0	0	4649	4	2	1	0
1204	6	5	1	0	1273	0	0	0	0	4650	4	2	1	0
1205	6	5	1	0	1274	0	0	0	0	4651	4	3	1	0
1206	6	5	1	0	1275	0	0	0	0	4652	4	4	0	0
1207	6	5	1	0	1276	0	0	0	0	4653	4	4	0	0
1208	6	5	1	0	1277	0	0	0	0	4654	4	4	0	0
1209	6	6	0	0	1278	0	0	0	0	4655	4	4	0	0
1210	6	6	0	0	1279	0	0	0	0	4656	4	4	0	0
1211	6	6	0	0	1280	0	0	0	0	4657	4	4	0	0
1212	6	6	0	0	1281	0	0	0	0	4658	4	4	0	0
1213	6	6	0	0	1282	0	0	0	0	4659	4	4	0	0
1214	6	6	0	0	1283	0	0	0	0	4660	4	4	0	0
1215	6	6	0	0	1284	0	0	0	0	4661	4	4	0	0
1216	6	6	0	0	1285	0	0	0	0	4662	4	4	0	0
1217	6	6	0	0	1286	0	0	0	0	4663	4	4	0	0
1218	6	6	0	0	1287	0	0	0	0	4664	4	4	0	0
1219	6	6	0	0	1288	0	0	0	0	4665	4	4	0	0
1220	6	6	0	0	1289	0	0	0	0	4666	4	4	0	0
1221	6	6	0	0	1290	0	0	0	0	4667	4	4	0	0
1222	6	6	0	0	1291	0	0	0	0	4668	4	4	0	0
1223	6	6	0	0	1292	0	0	0	0	4669	4	4	0	0
1224	5	5	0	0	1293	0	0	0	0	4670	4	4	0	0
1225	5	5	0	0	1294	0	0	0	0	4671	4	4	0	0
1226	5	5	0	0	1295	0	0	0	0	4672	4	5	0	1
1227	5	5	0	0	1296	0	0	0	0	4673	4	4	0	0
1228	5	5	0	0	1297	0	0	0	0	4674	4	4	0	0
1229	5	5	0	0	1298	0	0	0	0	4675	4	4	0	0
1230	5	5	0	0	1299	0	0	0	0	4676	4	4	0	0
1231	5	5	0	0	1300	0	0	0	0	4677	4	5	0	1
1232	5	5	0	0	1301	0	0	0	0	4678	4	5	0	1
1233	5	5	0	0	1302	0	0	0	0	4679	4	5	0	1
1234	5	5	0	0	1303	0	0	0	0	4680	4	4	0	0
1235	5	5	0	0	1304	0	0	0	0	4681	4	4	0	0
1236	5	5	0	0	1305	0	0	0	0	4682	4	4	0	0
1237	5	5	0	0	1306	0	0	0	0	4683	4	4	0	0
1238	5	5	0	0	1307	0	0	0	0	4684	4	4	0	0
1239	5	5	0	0	1308	0	0	0	0	4685	4	4	0	0
1240	5	5	0	0	1309	0	0	0	0	4686	4	4	0	0
1241	5	5	0	0	1310	0	0	0	0	4687	4	4	0	0
1242	5	5	0	0	1311	0	0	0	0	4688	4	4	0	0
1243	5	5	0	0	1312	0	0	0	0	4689	4	4	0	0
1244	5	5	0	0	1313	0	0	0	0	4690	4	4	0	0
1245	5	5	0	0	1314	0	0	0	0	4691	4	4	0	0
1246	5	5	0	0	1315	0	0	0	0	4692	4	4	0	0
1247	5	5	0	0	1316	0	0	0	0	4693	4	4	0	0
1248	5	5	0	0	1317	0	0	0	0	4694	4	4	0	0
1249	5	5	0	0	1318	0	0	0	0	4695	4	4	0	0
1250	5	5	0	0	1319	0	0	0	0	4696	4	4	0	0
1251	5	5	0	0	1320	0	0	0	0	4697	4	4	0	0

1252	5	5	0	0	1321	0	0	0	0	4698	4	4	0	0
1253	5	5	0	0	1322	0	0	0	0	4699	4	4	0	0
1254	5	5	0	0	1323	0	0	0	0	4700	4	4	0	0
1255	5	5	0	0	1324	0	0	0	0	4701	4	4	0	0
1256	5	5	0	0	1325	0	0	0	0	4702	4	4	0	0
1257	5	5	0	0	1326	0	0	0	0	4703	4	4	0	0
1258	5	5	0	0	1327	0	0	0	0	4704	4	4	0	0
1259	5	5	0	0	1328	0	0	0	0	4705	4	4	0	0
1260	5	5	0	0	1329	0	0	0	0	4706	4	4	0	0
1261	5	5	0	0	1330	0	0	0	0	4707	4	4	0	0
1262	5	5	0	0	1331	0	0	0	0	4708	4	4	0	0
1263	5	5	0	0	1332	0	0	0	0	4709	4	4	0	0
1264	5	5	0	0	1333	0	0	0	0	4710	4	4	0	0
1265	5	5	0	0	1334	0	0	0	0	4711	4	4	0	0
1266	5	5	0	0	1335	0	0	0	0	4712	4	3	0	0
1267	5	5	0	0	1336	0	0	0	0	4713	4	3	0	0
1268	5	5	0	0	1337	0	0	0	0	4714	4	3	0	0
1269	5	5	0	0	1338	0	0	0	0	4715	4	3	0	0
1270	5	5	0	0	1339	0	0	0	0	4716	4	3	0	0
1271	5	5	0	0	1340	0	0	0	0	4717	4	3	0	0
1272	5	5	0	0	1341	0	0	0	0	4718	4	3	0	0
1273	5	5	0	0	1342	0	0	0	0	4719	4	3	0	0
1274	5	5	0	0	1343	0	0	0	0	4720	4	3	0	0
1275	5	5	0	0	1344	0	0	0	0	4721	4	3	0	0
1276	5	5	0	0	1345	0	0	0	0	4722	4	3	0	0
1277	5	5	0	0	1346	0	0	0	0	4723	4	3	0	0
1278	5	5	0	0	1347	0	0	0	0	4724	3	3	0	0
1279	5	5	0	0	1348	0	0	0	0	4725	3	3	0	0
1280	5	5	0	0	1349	0	0	0	0	4726	3	3	0	0
1281	5	5	0	0	1350	0	0	0	0	4727	3	3	0	0
1282	5	5	0	0	1351	0	0	0	0	4728	3	3	0	0
1283	5	5	0	0	1352	0	0	0	0	4729	3	3	0	0
1284	5	5	0	0	1353	0	0	0	0	4730	3	3	0	0
1285	5	5	0	0	1354	0	0	0	0	4731	3	3	0	0
1286	5	5	0	0	1355	0	0	0	0	4732	3	3	0	0
1287	5	5	0	0	1356	0	0	0	0	4733	3	3	0	0
1288	5	5	0	0	1357	0	0	0	0	4734	3	3	0	0
1289	5	5	0	0	1358	0	0	0	0	4735	3	3	0	0
1290	5	5	0	0	1359	0	0	0	0	4736	3	5	0	2
1291	5	5	0	0	1360	0	0	0	0	4737	3	5	0	2
1292	5	5	0	0	1361	0	0	0	0	4738	3	3	0	0
1293	5	5	0	0	1362	0	0	0	0	4739	3	3	0	0
1294	5	5	0	0	1363	0	0	0	0	4740	3	3	0	0
1295	5	5	0	0	1364	0	0	0	0	4741	3	3	0	0
1296	5	5	0	0	1365	0	0	0	0	4742	3	3	0	0
1297	5	5	0	0	1366	0	0	0	0	4743	3	3	0	0
1298	5	5	0	0	1367	0	0	0	0	4744	3	3	0	0
1299	5	5	0	0	1368	0	0	0	0	4745	3	3	0	0
1300	5	5	0	0	1369	0	0	0	0	4746	3	3	0	0
1301	5	5	0	0	1370	0	0	0	0	4747	3	3	0	0
1302	5	5	0	0	1371	0	0	0	0	4748	3	3	0	0
1303	5	5	0	0	1372	0	0	0	0	4749	3	3	0	0
1304	5	5	0	0	1373	0	0	0	0	4750	3	3	0	0
1305	5	5	0	0	1374	0	0	0	0	4751	3	3	0	0
1306	5	5	0	0	1375	0	0	0	0	4752	3	3	0	0
1307	5	5	0	0	1376	0	0	0	0	4753	3	3	0	0

1308	5	5	0	0	1377	0	0	0	0	4754	3	3	0	0	1364	8	5	3	0	1433	0	0	0	0	4810	2	2	0	0
1309	5	5	0	0	1378	0	0	0	0	4755	3	3	0	0	1365	8	5	3	0	1434	0	0	0	0	4811	2	2	0	0
1310	5	5	0	0	1379	0	0	0	0	4756	3	2	0	0	1366	8	5	3	0	1435	0	0	0	0	4812	2	2	0	0
1311	5	5	0	0	1380	0	0	0	0	4757	3	2	0	0	1367	8	5	3	0	1436	0	0	0	0	4813	2	2	0	0
1312	5	5	0	0	1381	0	0	0	0	4758	3	2	0	0	1368	9	6	3	0	1437	0	0	0	0	4814	2	2	0	0
1313	5	5	0	0	1382	0	0	0	0	4759	3	2	0	0	1369	9	6	3	0	1438	1	1	0	0	4815	2	2	0	0
1314	5	5	0	0	1383	0	0	0	0	4760	3	2	0	0	1370	9	7	2	0	1439	1	1	0	0	4816	2	2	0	0
1315	5	5	0	0	1384	0	0	0	0	4761	3	2	0	0	1371	9	7	2	0	1440	1	1	0	0	4817	2	2	0	0
1316	5	5	0	0	1385	0	0	0	0	4762	3	2	0	0	1372	9	7	2	0	1441	1	1	0	0	4818	2	2	0	0
1317	5	5	0	0	1386	0	0	0	0	4763	3	2	0	0	1373	9	7	2	0	1442	1	1	0	0	4819	2	2	0	0
1318	5	5	0	0	1387	0	0	0	0	4764	3	2	0	0	1374	9	7	2	0	1443	1	1	0	0	4820	2	2	0	0
1319	6	6	0	0	1388	0	0	0	0	4765	3	2	0	0	1375	9	7	2	0	1444	1	1	0	0	4821	2	2	0	0
1320	6	6	0	0	1389	0	0	0	0	4766	3	2	0	0	1376	9	7	2	0	1445	1	1	0	0	4822	2	2	0	0
1321	6	6	0	0	1390	0	0	0	0	4767	3	2	0	0	1377	9	6	3	0	1446	1	1	0	0	4823	2	2	0	0
1322	6	6	0	0	1391	0	0	0	0	4768	3	3	0	0	1378	9	6	3	0	1447	1	1	0	0	4824	2	2	0	0
1323	6	6	0	0	1392	0	0	0	0	4769	3	3	0	0	1379	9	6	3	0	1448	1	1	0	0	4825	2	2	0	0
1324	6	6	0	0	1393	0	0	0	0	4770	3	2	0	0	1380	9	6	3	0	1449	1	1	0	0	4826	2	2	0	0
1325	6	6	0	0	1394	0	0	0	0	4771	3	2	0	0	1381	9	6	3	0	1450	1	1	0	0	4827	2	2	0	0
1326	6	5	1	0	1395	0	0	0	0	4772	3	2	0	0	1382	9	8	1	0	1451	1	1	0	0	4828	2	2	0	0
1327	6	5	1	0	1396	0	0	0	0	4773	2	2	0	0	1383	9	8	1	0	1452	1	1	0	0	4829	2	2	0	0
1328	6	5	1	0	1397	0	0	0	0	4774	2	2	0	0	1384	9	8	1	0	1453	1	1	0	0	4830	2	2	0	0
1329	6	5	1	0	1398	0	0	0	0	4775	2	2	0	0	1385	9	8	1	0	1454	2	1	1	0	4831	2	2	0	0
1330	6	5	1	0	1399	0	0	0	0	4776	2	2	0	0	1386	9	8	1	0	1455	2	1	1	0	4832	2	2	0	0
1331	6	5	1	0	1400	0	0	0	0	4777	2	2	0	0	1387	9	8	1	0	1456	3	2	1	0	4833	2	2	0	0
1332	6	5	1	0	1401	0	0	0	0	4778	2	2	0	0	1388	9	8	1	0	1457	3	2	1	0	4834	2	2	0	0
1333	6	5	1	0	1402	0	0	0	0	4779	2	2	0	0	1389	9	8	1	0	1458	3	3	1	1	4835	2	2	0	0
1334	8	6	2	0	1403	0	0	0	0	4780	2	2	0	0	1390	9	9	1	1	1459	3	3	1	1	4836	2	2	0	0
1335	8	6	2	0	1404	0	0	0	0	4781	2	2	0	0	1391	9	9	1	1	1460	3	3	1	1	4837	2	2	0	0
1336	8	6	2	0	1405	0	0	0	0	4782	2	2	0	0	1392	9	9	1	1	1461	3	3	1	1	4838	2	2	0	0
1337	8	6	2	0	1406	0	0	0	0	4783	2	2	0	0	1393	9	9	1	1	1462	3	3	1	1	4839	2	2	0	0
1338	8	6	2	0	1407	0	0	0	0	4784	2	2	0	0	1394	9	8	1	0	1463	3	3	1	1	4840	2	2	0	0
1339	8	6	2	0	1408	0	0	0	0	4785	2	2	0	0	1395	9	8	1	0	1464	3	3	1	1	4841	2	2	0	0
1340	8	6	2	0	1409	0	0	0	0	4786	2	2	0	0	1396	9	8	1	0	1465	3	3	1	1	4842	3	2	0	0
1341	8	7	1	0	1410	0	0	0	0	4787	2	2	0	0	1397	9	8	1	0	1466	3	2	1	0	4843	3	2	0	0
1342	8	6	2	0	1411	0	0	0	0	4788	2	2	0	0	1398	9	8	1	0	1467	3	2	1	0	4844	3	2	0	0
1343	8	6	2	0	1412	0	0	0	0	4789	2	2	0	0	1399	9	8	1	0	1468	3	2	1	0	4845	3	2	0	0
1344	9	6	3	0	1413	0	0	0	0	4790	2	2	0	0	1400	9	8	1	0	1469	3	2	1	0	4846	3	2	0	0
1345	9	6	3	0	1414	0	0	0	0	4791	2	2	0	0	1401	9	8	1	0	1470	3	2	1	0	4847	3	2	0	0
1346	10	7	3	0	1415	0	0	0	0	4792	2	2	0	0	1402	9	8	1	0	1471	3	2	1	0	4848	3	2	0	0
1347	10	7	3	0	1416	0	0	0	0	4793	2	2	0	0	1403	9	8	1	0	1472	3	2	1	0	4849	3	2	0	0
1348	10	7	3	0	1417	0	0	0	0	4794	2	2	0	0	1404	9	8	1	0	1473	3	2	1	0	4850	3	2	0	0
1349	10	7	3	0	1418	0	0	0	0	4795	2	2	0	0	1405	9	8	1	0	1474	3	2	1	0	4851	3	2	0	0
1350	10	7	3	0	1419	0	0	0	0	4796	2	2	0	0	1406	9	8	1	0	1475	3	2	1	0	4852	3	2	0	0
1351	10	7	3	0	1420	0	0	0	0	4797	2	2	0	0	1407	9	8	1	0	1476	3	2	1	0	4853	3	2	0	0
1352	10	7	3	0	1421	0	0	0	0	4798	2	2	0	0	1408	9	8	1	0	1477	3	2	1	0	4854	3	2	0	0
1353	10	7	3	0	1422	0	0	0	0	4799	2	2	0	0	1409	8	7	1	0	1478	3	2	1	0	4855	3	2	0	0
1354	10	7	3	0	1423	0	0	0	0	4800	2	2	0	0	1410	8	7	1	0	1479	3	2	1	0	4856	3	2	0	0
1355	10	7	3	0	1424	0	0	0	0	4801	2	2	0	0	1411	8	7	1	0	1480	3	2	1	0	4857	3	3	0	0
1356	10	7	3	0	1425	0	0	0	0	4802	2	2	0	0	1412	8	7	1	0	1481	3	2	1	0	4858	3	3	0	0
1357	10	7	3	0	1426	0	0	0	0	4803	2	2	0	0	1413	8	7	1	0	1482	3	2	1	0	4859	3	2	0	0
1358	10	7	3	0	1427	0	0	0	0	4804	2	2	0	0	1414	8	7	1	0	1483	3	2	1	0	4860	3	2	0	0
1359	9	6	3	0	1428	0	0	0	0	4805	2	2	0	0	1415	8	7	1	0	1484	3	2	1	0	4861	3	2	0	0
1360	8	6	2	0	1429	0	0	0	0	4806	2	2	0	0	1416	8	7	1	0	1485	3	2	1	0	4862	3	2	0	0
1361	8	6	2	0	1430	0	0	0	0	4807	2	2	0	0	1417	8	7	1	0	1486	3	2	1	0	4863	3	2	0	0
1362	8	5	3	0	1431	0	0	0	0	4808	2	2	0	0	1418	8	7	1	0	1487	3	2	1	0	4864	3	2	0	0
1363	8	5	3	0	1432	0	0	0	0	4809	2	2	0	0	1419	8	7	1	0	1488	3	2	1	0	4865	3	2	0	0

1420	8	7	1	0	1489	3	2	1	0	4866	3	2	0	0	1476	8	6	2	0	1545	4	3	1	0	4922	3	2	0	0
1421	8	7	1	0	1490	3	2	1	0	4867	3	3	0	0	1477	8	6	2	0	1546	4	3	1	0	4923	3	2	0	0
1422	8	6	2	0	1491	3	2	1	0	4868	3	3	0	0	1478	8	6	2	0	1547	4	3	1	0	4924	3	2	0	0
1423	8	6	2	0	1492	3	2	1	0	4869	3	2	0	0	1479	8	6	2	0	1548	4	3	1	0	4925	3	2	0	0
1424	8	6	2	0	1493	3	2	1	0	4870	3	2	0	0	1480	8	6	2	0	1549	4	3	1	0	4926	3	3	0	0
1425	8	6	2	0	1494	3	2	1	0	4871	3	2	0	0	1481	8	6	2	0	1550	4	3	1	0	4927	3	3	0	0
1426	8	6	2	0	1495	3	2	1	0	4872	3	2	0	0	1482	8	6	2	0	1551	4	3	1	0	4928	3	4	0	1
1427	8	6	2	0	1496	3	2	1	0	4873	3	2	0	0	1483	7	5	2	0	1552	4	3	1	0	4929	3	4	0	1
1428	8	6	2	0	1497	3	2	1	0	4874	3	2	0	0	1484	7	6	2	1	1553	4	3	1	0	4930	3	4	0	1
1429	8	6	2	0	1498	3	2	1	0	4875	3	2	0	0	1485	7	5	2	0	1554	4	3	1	0	4931	3	4	0	1
1430	8	6	2	0	1499	3	2	1	0	4876	3	2	0	0	1486	7	5	2	0	1555	4	3	1	0	4932	3	3	0	0
1431	8	6	2	0	1500	3	2	1	0	4877	3	2	0	0	1487	7	5	2	0	1556	4	3	1	0	4933	3	3	0	0
1432	8	6	2	0	1501	3	2	1	0	4878	3	2	0	0	1488	7	5	2	0	1557	4	3	1	0	4934	3	3	0	0
1433	8	6	2	0	1502	3	2	1	0	4879	3	2	0	0	1489	7	5	2	0	1558	4	3	1	0	4935	3	3	0	0
1434	8	6	2	0	1503	3	2	1	0	4880	3	2	0	0	1490	7	5	2	0	1559	4	3	1	0	4936	3	3	0	0
1435	8	6	2	0	1504	3	2	1	0	4881	3	2	0	0	1491	7	5	2	0	1560	4	3	1	0	4937	3	3	0	0
1436	8	6	2	0	1505	3	2	1	0	4882	3	2	0	0	1492	7	5	2	0	1561	4	3	1	0	4938	4	3	0	0
1437	8	6	2	0	1506	3	2	1	0	4883	3	2	0	0	1493	7	5	2	0	1562	4	3	1	0	4939	4	3	0	0
1438	8	6	2	0	1507	3	2	1	0	4884	3	2	0	0	1494	7	5	2	0	1563	4	3	1	0	4940	4	3	0	0
1439	8	6	2	0	1508	3	2	1	0	4885	3	2	0	0	1495	7	5	2	0	1564	4	3	1	0	4941	4	3	0	0
1440	8	5	3	0	1509	3	2	1	0	4886	3	2	0	0	1496	7	5	2	0	1565	4	3	1	0	4942	4	3	0	0
1441	8	5	3	0	1510	3	2	1	0	4887	3	2	0	0	1497	7	5	2	0	1566	4	3	1	0	4943	4	3	0	0
1442	8	5	3	0	1511	3	2	1	0	4888	3	2	0	0	1498	7	5	2	0	1567	4	3	1	0	4944	4	4	0	0
1443	8	6	2	0	1512	3	2	1	0	4889	3	2	0	0	1499	7	5	2	0	1568	4	3	1	0	4945	4	4	0	0
1444	8	6	2	0	1513	3	2	1	0	4890	3	2	0	0	1500	7	5	2	0	1569	4	3	1	0	4946	4	4	0	0
1445	8	6	2	0	1514	3	2	1	0	4891	3	2	0	0	1501	7	6	1	0	1570	4	3	1	0	4947	4	4	0	0
1446	8	6	2	0	1515	3	2	1	0	4892	3	2	0	0	1502	7	6	1	0	1571	4	3	1	0	4948	4	4	0	0
1447	8	6	2	0	1516	3	2	1	0	4893	3	2	0	0	1503	7	6	1	0	1572	4	3	1	0	4949	4	4	0	0
1448	8	7	2	1	1517	3	2	1	0	4894	3	2	0	0	1504	7	6	1	0	1573	4	3	1	0	4950	4	3	1	0
1449	8	7	2	1	1518	3	2	1	0	4895	3	2	0	0	1505	7	6	1	0	1574	4	3	1	0	4951	4	3	1	0
1450	8	7	2	1	1519	3	2	1	0	4896	3	2	0	0	1506	7	6	1	0	1575	4	3	1	0	4952	4	3	1	0
1451	8	7	2	1	1520	3	2	1	0	4897	3	2	0	0	1507	7	6	1	0	1576	4	3	1	0	4953	4	3	1	0
1452	8	7	2	1	1521	3	2	1	0	4898	3	2	0	0	1508	7	6	1	0	1577	4	3	1	0	4954	4	3	1	0
1453	8	7	2	1	1522	3	2	1	0	4899	3	2	0	0	1509	7	6	1	0	1578	5	3	1	0	4955	4	3	1	0
1454	8	6	2	0	1523	3	2	1	0	4900	3	2	0	0	1510	7	6	1	0	1579	5	3	1	0	4956	4	3	1	0
1455	8	6	2	0	1524	4	3	1	0	4901	3	2	0	0	1511	7	6	1	0	1580	5	3	1	0	4957	4	3	1	0
1456	8	6	2	0	1525	4	4	1	1	4902	3	2	0	0	1512	7	6	1	0	1581	5	3	1	0	4958	4	3	1	0
1457	8	6	2	0	1526	4	4	1	1	4903	3	2	0	0	1513	7	6	1	0	1582	5	3	1	0	4959	4	3	1	0
1458	8	6	2	0	1527	4	4	1	1	4904	3	2	0	0	1514	7	6	1	0	1583	5	4	1	0	4960	4	3	1	0
1459	8	6	2	0	1528	4	4	1	1	4905	3	2	0	0	1515	7	6	1	0	1584	5	4	1	0	4961	4	3	1	0
1460	8	6	2	0	1529	4	3	1	0	4906	3	2	0	0	1516	7	6	1	0	1585	5	4	1	0	4962	4	3	1	0
1461	8	6	2	0	1530	4	3	1	0	4907	3	2	0	0	1517	7	6	1	0	1586	5	4	1	0	4963	5	3	0	0
1462	8	6	2	0	1531	4	3	1	0	4908	3	2	0	0	1518	7	6	1	0	1587	5	4	1	0	4964	5	3	0	0
1463	8	6	2	0	1532	4	3	1	0	4909	3	2	0	0	1519	7	6	1	0	1588	5	4	1	0	4965	5	3	0	0
1464	8	6	2	0	1533	4	3	1	0	4910	3	2	0	0	1520	7	6	1	0	1589	5	4	1	0	4966	5	4	0	0
1465	8	6	2	0	1534	4	3	1	0	4911	3	2	0	0	1521	7	5	2	0	1590	5	4	1	0	4967	5	4	0	0
1466	8	6	2	0	1535	4	3	1	0	4912	3	2	0	0	1522	7	5	2	0	1591	5	4	1	0	4968	4	5	0	1
1467	8	6	2	0	1536	4	3	1	0	4913	3	2	0	0	1523	7	5	2	0	1592	5	4	1	0	4969	4	5	0	1
1468	8	6	2	0	1537	4	3	1	0	4914	3	2	0	0	1524	7	5	2	0	1593	5	4	1	0	4970	4	4	0	0
1469	8	6	2	0	1538	4	3	1	0	4915	3	2	0	0	1525	7	5	2	0	1594	5	4	1	0	4971	4	4	0	0
1470	8	6	2	0	1539	4	3	1	0	4916	3	2	0	0	1526	7	5	2	0	1595	5	4	1	0	4972	4	4	0	0
1471	8	6	2	0	1540	4	3	1	0	4917	3	2	0	0	1527	7	5	2	0	1596	5	4	1	0	4973	4	4	0	0
1472	8	6	2	0	1541	4	3	1	0	4918	3	2	0	0	1528	7	5	2	0	1597	5	4	1	0	4974	4	4	0	0
1473	8	6	2	0	1542	4	3	1	0	4919	3	2	0	0	1529	7	5	2	0	1598	5	4	1	0	4975	4	4	0	0
1474	8	6	2	0	1543	4	3	1	0	4920	3	2	0	0	1530	7	5	2	0	1599	5	4	1	0	4976	4	4	0	0
1475	8	6	2	0	1544	4	3	1	0	4921	3	2	0	0	1531	7	5	2	0	1600	5	4	1	0	4977	4	5	0	1

1532	7	5	2	0	1601	5	4	1	0	4978	4	5	0	1	1588	6	4	2	0	1657	5	4	1	0	5034	6	5	1	0
1533	7	5	2	0	1602	5	4	1	0	4979	4	5	0	1	1589	6	4	2	0	1658	5	4	1	0	5035	6	5	1	0
1534	7	5	2	0	1603	5	4	1	0	4980	4	5	0	1	1590	6	4	2	0	1659	5	4	1	0	5036	6	5	1	0
1535	7	5	2	0	1604	5	4	1	0	4981	4	5	0	1	1591	6	4	2	0	1660	5	4	1	0	5037	6	5	1	0
1536	7	5	2	0	1605	5	4	1	0	4982	4	4	0	0	1592	6	4	2	0	1661	5	4	1	0	5038	6	5	1	0
1537	7	5	2	0	1606	5	4	1	0	4983	4	4	0	0	1593	6	4	2	0	1662	5	4	1	0	5039	6	5	1	0
1538	7	5	2	0	1607	5	4	1	0	4984	4	4	0	0	1594	6	4	2	0	1663	5	4	1	0	5040	6	5	1	0
1539	7	5	2	0	1608	5	4	1	0	4985	4	4	0	0	1595	6	4	2	0	1664	5	4	1	0	5041	6	5	1	0
1540	7	6	1	0	1609	5	4	1	0	4986	4	4	0	0	1596	6	4	2	0	1665	5	4	1	0	5042	6	5	1	0
1541	7	6	1	0	1610	5	4	1	0	4987	4	4	0	0	1597	6	4	2	0	1666	5	4	1	0	5043	6	5	1	0
1542	7	6	1	0	1611	5	4	1	0	4988	4	4	0	0	1598	6	4	2	0	1667	5	4	1	0	5044	6	5	1	0
1543	7	6	1	0	1612	5	4	1	0	4989	4	4	0	0	1599	6	4	2	0	1668	5	4	1	0	5045	6	5	1	0
1544	7	4	3	0	1613	5	4	1	0	4990	4	4	0	0	1600	6	4	2	0	1669	5	4	1	0	5046	6	5	1	0
1545	7	4	3	0	1614	5	4	1	0	4991	4	4	0	0	1601	6	4	2	0	1670	5	4	1	0	5047	6	5	1	0
1546	7	4	3	0	1615	5	4	1	0	4992	4	4	0	0	1602	6	4	2	0	1671	5	4	1	0	5048	6	5	1	0
1547	7	4	3	0	1616	5	4	1	0	4993	4	4	0	0	1603	6	4	2	0	1672	5	4	1	0	5049	6	5	1	0
1548	7	4	3	0	1617	5	4	1	0	4994	4	4	0	0	1604	6	4	2	0	1673	5	4	1	0	5050	6	5	1	0
1549	7	4	3	0	1618	5	4	1	0	4995	4	4	0	0	1605	6	4	2	0	1674	5	4	1	0	5051	6	5	1	0
1550	7	4	3	0	1619	5	4	1	0	4996	4	4	0	0	1606	6	4	2	0	1675	5	3	2	0	5052	6	5	1	0
1551	7	4	3	0	1620	5	4	1	0	4997	4	4	0	0	1607	6	4	2	0	1676	5	3	2	0	5053	6	5	1	0
1552	7	4	3	0	1621	5	4	1	0	4998	4	4	0	0	1608	6	4	2	0	1677	5	3	2	0	5054	6	5	1	0
1553	7	4	3	0	1622	5	4	1	0	4999	4	4	0	0	1609	6	4	2	0	1678	5	3	2	0	5055	6	6	1	1
1554	7	4	3	0	1623	5	4	1	0	5000	4	4	0	0	1610	6	4	2	0	1679	5	3	2	0	5056	6	5	1	0
1555	7	4	3	0	1624	5	4	1	0	5001	4	4	0	0	1611	6	4	2	0	1680	5	3	2	0	5057	6	5	1	0
1556	7	4	3	0	1625	5	4	1	0	5002	4	4	0	0	1612	6	4	2	0	1681	5	3	2	0	5058	6	5	1	0
1557	7	4	3	0	1626	5	4	1	0	5003	4	4	0	0	1613	6	4	2	0	1682	5	3	2	0	5059	6	5	1	0
1558	7	4	3	0	1627	5	4	1	0	5004	4	4	0	0	1614	6	4	2	0	1683	5	3	2	0	5060	6	4	2	0
1559	7	4	3	0	1628	5	4	1	0	5005	4	4	0	0	1615	6	4	2	0	1684	5	3	2	0	5061	6	4	2	0
1560	7	4	3	0	1629	5	4	1	0	5006	4	4	0	0	1616	6	4	2	0	1685	5	3	2	0	5062	6	4	2	0
1561	7	4	3	0	1630	5	4	1	0	5007	4	4	0	0	1617	6	4	2	0	1686	5	3	2	0	5063	6	4	2	0
1562	7	4	3	0	1631	5	4	1	0	5008	4	4	0	0	1618	6	4	2	0	1687	5	3	2	0	5064	6	4	2	0
1563	7	4	3	0	1632	5	4	1	0	5009	4	4	0	0	1619	6	4	2	0	1688	5	3	2	0	5065	6	4	2	0
1564	7	4	3	0	1633	5	4	1	0	5010	4	4	0	0	1620	6	4	2	0	1689	5	4	1	0	5066	6	4	2	0
1565	7	4	3	0	1634	5	4	1	0	5011	4	4	0	0	1621	6	4	2	0	1690	5	4	1	0	5067	6	4	2	0
1566	7	4	3	0	1635	5	4	1	0	5012	4	4	0	0	1622	6	4	2	0	1691	5	4	1	0	5068	6	4	2	0
1567	7	4	3	0	1636	5	5	1	1	5013	4	3	1	0	1623	6	4	2	0	1692	5	4	1	0	5069	6	4	2	0
1568	7	4	3	0	1637	5	5	1	1	5014	4	3	1	0	1624	6	4	2	0	1693	5	4	1	0	5070	6	5	2	1
1569	7	4	3	0	1638	5	5	1	1	5015	4	3	1	0	1625	6	4	2	0	1694	5	4	1	0	5071	6	5	2	1
1570	7	4	3	0	1639	5	5	1	1	5016	4	3	1	0	1626	6	4	2	0	1695	5	4	1	0	5072	6	4	2	0
1571	7	4	3	0	1640	5	5	1	1	5017	4	3	1	0	1627	6	4	2	0	1696	5	4	1	0	5073	6	4	2	0
1572	7	4	3	0	1641	5	5	1	1	5018	4	3	1	0	1628	6	4	2	0	1697	5	4	1	0	5074	6	5	1	0
1573	6	3	3	0	1642	5	5	1	1	5019	4	3	1	0	1629	6	4	2	0	1698	5	4	1	0	5075	6	5	1	0
1574	6	3	3	0	1643	5	5	1	1	5020	4	3	1	0	1630	6	4	2	0	1699	5	4	1	0	5076	6	5	1	0
1575	6	3	3	0	1644	5	4	1	0	5021	4	3	1	0	1631	6	4	2	0	1700	5	4	1	0	5077	6	5	1	0
1576	6	3	3	0	1645	5	4	1	0	5022	4	3	1	0	1632	6	4	2	0	1701	5	4	1	0	5078	6	5	1	0
1577	6	3	3	0	1646	5	4	1	0	5023	6	4	2	1	1633	6	4	2	0	1702	5	4	1	0	5079	6	5	1	0
1578	6	3	3	0	1647	5	4	1	0	5024	6	4	2	1	1634	6	4	2	0	1703	5	4	1	0	5080	6	5	1	0
1579	6	3	3	0	1648	5	4	1	0	5025	6	4	2	0	1635	6	4	2	0	1704	5	4	1	0	5081	6	5	1	0
1580	6	3	3	0	1649	5	4	1	0	5026	6	4	2	0	1636	6	4	2	0	1705	5	6	1	2	5082	6	5	1	0
1581	6	3	3	0	1650	5	4	1	0	5027	6	5	1	0	1637	6	5	1	0	1706	5	6	1	2	5083	6	5	1	0
1582	6	3	3	0	1651	5	4	1	0	5028	6	5	1	0	1638	6	6	0	0	1707	5	5	1	1	5084	6	5	1	0
1583	6	3	3	0	1652	5	4	1	0	5029	6	5	1	0	1639	6	6	0	0	1708	5	5	1	1	5085	6	6	1	0
1584	6	3	3	0	1653	5	4	1	0	5030	6	5	1	0	1640	6	6	0	0	1709	5	5	1	1	5086	6	7	1	2
1585	6	3	3	0	1654	5	4	1	0	5031	6	5	1	0	1641	6	6	0	0	1710	5	5	1	1	5087	6	7	1	2
1586	6	3	3	0	1655	5	4	1	0	5032	6	5	1	0	1642	6	6	0	0	1711	5	5	1	1	5088	6	6	1	1
1587	6	3	3	0	1656	5	4	1	0	5033	6	5	1	0	1643	6	6	0	0	1712	5	5	1	1	5089	6	6	1	1

1644	6	5	1	0	1713	5	5	1	1	5090	6	5	1	0
1645	6	5	1	0	1714	5	5	1	1	5091	6	5	1	0
1646	6	5	1	0	1715	5	5	1	1	5092	6	5	1	0
1647	6	4	2	0	1716	5	5	1	1	5093	6	6	1	1
1648	6	4	2	0	1717	5	5	1	1	5094	6	6	1	1
1649	6	4	2	0	1718	5	5	1	1	5095	6	6	1	1
1650	6	4	2	0	1719	5	4	1	0	5096	6	6	1	1
1651	6	4	2	0	1720	5	4	1	0	5097	6	6	1	1
1652	6	4	2	0	1721	5	4	1	0	5098	6	6	1	2
1653	6	4	2	0	1722	5	4	1	0	5099	6	6	1	2
1654	6	4	2	0	1723	5	4	1	0	5100	6	6	1	2
1655	6	4	2	0	1724	5	4	1	0	5101	6	4	1	0
1656	6	4	2	0	1725	5	5	1	1	5102	6	4	1	0
1657	6	4	2	0	1726	5	5	1	1	5103	5	4	1	0
1658	6	4	2	0	1727	5	4	1	0	5104	5	4	1	0
1659	6	4	2	0	1728	5	4	1	0	5105	5	4	1	0
1660	6	4	2	0	1729	5	4	1	0	5106	5	4	1	0
1661	6	4	2	0	1730	5	4	1	0	5107	5	4	1	0
1662	6	4	2	0	1731	5	4	1	0	5108	5	5	1	1
1663	6	4	2	0	1732	5	4	1	0	5109	5	5	1	1
1664	6	4	2	0	1733	5	4	1	0	5110	5	6	1	1
1665	6	4	2	0	1734	5	5	1	1	5111	5	6	1	1
1666	6	4	2	0	1735	5	5	1	1	5112	5	4	1	0
1667	6	4	2	0	1736	5	5	1	1	5113	5	3	2	0
1668	6	4	2	0	1737	5	5	1	1	5114	5	3	2	0
1669	6	4	2	0	1738	5	5	1	1	5115	5	3	2	0
1670	6	4	2	0	1739	5	5	1	1	5116	5	3	2	0
1671	6	4	2	0	1740	5	5	1	1	5117	5	4	2	1
1672	6	4	2	0	1741	5	5	1	1	5118	5	4	2	1
1673	6	4	2	0	1742	5	5	1	1	5119	5	4	2	1
1674	6	5	1	0	1743	5	5	1	1	5120	5	4	2	1
1675	6	5	1	0	1744	5	5	1	1	5121	5	4	2	1
1676	6	5	1	0	1745	5	5	1	1	5122	5	2	2	0
1677	6	5	1	0	1746	5	5	1	1	5123	5	2	2	0
1678	6	5	1	0	1747	5	5	1	1	5124	5	3	2	1
1679	6	5	1	0	1748	5	5	1	1	5125	5	3	1	0
1680	6	5	1	0	1749	5	5	1	1	5126	5	3	1	0
1681	6	5	1	0	1750	5	5	1	1	5127	4	3	1	0
1682	6	5	1	0	1751	5	5	1	1	5128	4	3	1	0
1683	6	5	1	0	1752	5	5	1	1	5129	4	3	1	0
1684	6	5	1	0	1753	5	5	1	1	5130	4	3	1	0
1685	6	5	1	0	1754	5	5	1	1	5131	4	4	1	1
1686	6	5	1	0	1755	5	5	1	1	5132	4	3	1	0
1687	6	5	1	0	1756	5	5	1	1	5133	4	4	1	1
1688	6	5	1	0	1757	5	5	1	1	5134	4	4	1	1
1689	6	5	1	0	1758	5	5	1	1	5135	4	4	1	1
1690	6	5	1	0	1759	5	4	1	0	5136	4	4	1	1
1691	6	5	1	0	1760	5	4	1	0	5137	4	4	1	1
1692	6	5	1	0	1761	5	4	1	0	5138	4	4	1	1
1693	6	5	1	0	1762	5	4	1	0	5139	4	3	1	0
1694	6	5	1	0	1763	5	4	1	0	5140	4	3	1	0
1695	6	5	1	0	1764	5	4	1	0	5141	4	3	1	0
1696	6	5	1	0	1765	5	4	1	0	5142	4	3	1	0
1697	6	5	1	0	1766	5	4	1	0	5143	4	3	1	0
1698	6	5	1	0	1767	5	4	1	0	5144	4	4	1	1
1699	6	5	1	0	1768	5	4	1	0	5145	4	3	1	0

1700	6	5	1	0	1769	5	4	1	0	5146	4	3	1	0
1701	6	5	1	0	1770	5	4	1	0	5147	4	3	1	0
1702	6	5	1	0	1771	5	4	1	0	5148	4	3	1	0
1703	6	5	1	0	1772	5	4	1	0	5149	4	3	1	0
1704	6	5	1	0	1773	5	4	1	0	5150	4	3	1	0
1705	6	5	1	0	1774	5	4	1	0	5151	4	3	1	0
1706	6	5	1	0	1775	5	4	1	0	5152	4	3	1	0
1707	6	5	1	0	1776	5	4	1	0	5153	4	3	1	0
1708	6	5	1	0	1777	5	4	1	0	5154	4	3	1	0
1709	6	5	1	0	1778	5	4	1	0	5155	4	3	1	0
1710	6	5	1	0	1779	5	4	1	0	5156	4	3	1	0
1711	6	5	1	0	1780	5	4	1	0	5157	4	3	1	0
1712	6	5	1	0	1781	5	4	1	0	5158	4	3	1	0
1713	6	5	1	0	1782	5	4	1	0	5159	4	3	1	0
1714	6	5	1	0	1783	5	4	1	0	5160	4	3	1	0
1715	6	5	1	0	1784	5	4	1	0	5161	4	3	1	0
1716	6	5	1	0	1785	5	4	1	0	5162	4	3	1	0
1717	6	5	1	0	1786	5	4	1	0	5163	4	3	1	0
1718	6	6	0	0	1787	5	4	1	0	5164	4	3	1	0
1719	6	6	0	0	1788	5	4	1	0	5165	4	3	1	0
1720	6	6	0	0	1789	5	4	1	0	5166	4	3	1	0
1721	6	6	0	0	1790	5	4	1	0	5167	4	4	1	1
1722	6	6	0	0	1791	5	4	1	0	5168	4	4	1	1
1723	6	6	0	0	1792	5	4	1	0	5169	4	4	1	1
1724	6	6	0	0	1793	5	4	1	0	5170	4	3	1	0
1725	5	4	1	0	1794	5	4	1	0	5171	4	3	1	0
1726	5	4	1	0	1795	5	4	1	0	5172	4	3	1	0
1727	5	4	1	0	1796	5	4	1	0	5173	4	3	1	0
1728	5	4	1	0	1797	5	4	1	0	5174	4	3	1	0
1729	5	4	1	0	1798	5	4	1	0	5175	4	3	1	0
1730	5	4	1	0	1799	5	4	1	0	5176	4	3	1	0
1731	5	4	1	0	1800	5	4	1	0	5177	4	3	1	0
1732	5	4	1	0	1801	5	4	1	0	5178	4	3	1	0
1733	5	4	1	0	1802	5	4	1	0	5179	4	3	1	0
1734	5	4	1	0	1803	5	4	1	0	5180	4	3	1	0
1735	5	4	1	0	1804	5	4	1	0	5181	4	3	1	0
1736	5	4	1	0	1805	5	4	1	0	5182	4	3	1	0
1737	5	4	1	0	1806	5	4	1	0	5183	4	3	1	0
1738	5	4	1	0	1807	5	4	1	0	5184	4	4	1	1
1739	5	4	1	0	1808	5	4	1	0	5185	4	4	1	1
1740	5	4	1	0	1809	5	4	1	0	5186	4	3	1	0
1741	5	5	0	0	1810	5	4	1	0	5187	4	3	1	0
1742	5	5	0	0	1811	5	4	1	0	5188	4	3	1	0
1743	5	5	0	0	1812	5	4	1	0	5189	4	3	1	0
1744	5	5	0	0	1813	5	4	1	0	5190	4	3	1	0
1745	5	5	0	0	1814	5	4	1	0	5191	4	3	1	0
1746	5	5	0	0	1815	5	4	1	0	5192	4	3	1	0
1747	5	5	0	0	1816	5	4	1	0	5193	4	3	1	0
1748	5	5	0	0	1817	5	4	1	0	5194	4	3	1	0
1749	5	5	0	0	1818	5	4	1	0	5195	4	3	1	0
1750	5	5	0	0	1819	5	4	1	0	5196	4	3	1	0
1751	5	5	0	0	1820	5	4	1	0	5197	4	3	1	0
1752	5	5	0	0	1821	5	4	1	0	5198	4	3	1	0
1753	5	5	0	0	1822	5	4	1	0	5199	4	3	1	0
1754	5	5	0	0	1823	5	4	1	0	5200	4	3	1	0
1755	5	5	0	0	1824	5	4	1	0	5201	4	3	1	0

1756	5	5	0	0	1825	5	4	1	0	5202	4	3	1	0
1757	5	5	0	0	1826	5	4	1	0	5203	4	3	1	0
1758	5	5	0	0	1827	5	4	1	0	5204	4	3	1	0
1759	5	5	0	0	1828	5	4	1	0	5205	4	3	1	0
1760	5	5	0	0	1829	5	4	1	0	5206	4	3	1	0
1761	5	5	0	0	1830	5	4	1	0	5207	4	3	1	0
1762	5	5	0	0	1831	5	4	1	0	5208	4	3	1	0
1763	5	5	0	0	1832	5	4	1	0	5209	4	3	1	0
1764	5	5	0	0	1833	5	4	1	0	5210	4	3	1	0
1765	5	5	0	0	1834	5	4	1	0	5211	4	3	1	0
1766	5	5	0	0	1835	3	4	0	1	5212	4	3	1	0
1767	5	5	0	0	1836	3	4	0	1	5213	4	3	1	0
1768	4	4	0	0	1837	3	4	0	1	5214	4	3	1	0
1769	4	4	0	0	1838	3	4	0	1	5215	4	3	1	0
1770	4	4	0	0	1839	3	3	0	0	5216	4	3	1	0
1771	4	4	0	0	1840	3	3	0	0	5217	4	3	1	0
1772	4	4	0	0	1841	3	3	0	0	5218	4	3	1	0
1773	3	3	0	0	1842	3	3	0	0	5219	4	3	1	0
1774	3	3	0	0	1843	3	3	0	0	5220	4	3	1	0
1775	3	3	0	0	1844	3	3	0	0	5221	4	2	1	0
1776	3	3	0	0	1845	3	3	0	0	5222	4	2	1	0
1777	3	3	0	0	1846	3	3	0	0	5223	4	3	1	0
1778	3	3	0	0	1847	3	3	0	0	5224	4	2	1	0
1779	4	5	0	1	1848	3	3	0	0	5225	4	2	1	0
1780	4	5	0	1	1849	3	3	0	0	5226	4	2	1	0
1781	4	4	0	0	1850	3	3	0	0	5227	4	2	1	0
1782	4	4	0	0	1851	3	3	0	0	5228	4	2	1	0
1783	4	4	0	0	1852	3	3	0	0	5229	4	2	1	0
1784	4	4	0	0	1853	3	3	0	0	5230	4	2	1	0
1785	4	3	0	0	1854	3	4	0	1	5231	3	2	1	0
1786	4	3	0	0	1855	3	4	0	1	5232	3	2	1	0
1787	4	3	0	0	1856	3	3	0	0	5233	3	2	1	0
1788	4	3	0	0	1857	3	3	0	0	5234	3	2	1	0
1789	4	3	0	0	1858	3	3	0	0	5235	3	2	1	0
1790	4	3	0	0	1859	3	3	0	0	5236	3	2	1	0
1791	4	3	0	0	1860	3	3	0	0	5237	3	2	1	0
1792	4	4	0	0	1861	2	2	0	0	5238	3	2	1	0
1793	4	4	0	0	1862	2	2	0	0	5239	3	2	1	0
1794	4	4	0	0	1863	2	2	0	0	5240	3	2	1	0
1795	4	4	0	0	1864	2	2	0	0	5241	3	2	1	0
1796	4	4	0	0	1865	2	2	0	0	5242	3	2	1	0
1797	4	4	0	0	1866	2	2	0	0	5243	3	2	1	0
1798	3	3	0	0	1867	2	2	0	0	5244	3	2	1	0
1799	3	3	0	0	1868	2	2	0	0	5245	3	2	1	0
1800	3	3	0	0	1869	2	2	0	0	5246	3	2	1	0
1801	3	3	0	0	1870	2	2	0	0	5247	3	2	1	0
1802	3	3	0	0	1871	2	2	0	0	5248	3	1	1	0
1803	3	3	0	0	1872	2	2	0	0	5249	3	1	1	0
1804	3	3	0	0	1873	2	2	0	0	5250	3	1	1	0
1805	3	3	0	0	1874	2	2	0	0	5251	3	1	1	0
1806	3	3	0	0	1875	2	2	0	0	5252	3	1	1	0
1807	3	3	0	0	1876	2	2	0	0	5253	3	1	1	0
1808	3	3	0	0	1877	2	2	0	0	5254	3	1	1	0
1809	3	3	0	0	1878	2	2	0	0	5255	3	1	1	0
1810	3	3	0	0	1879	2	2	0	0	5256	3	1	1	0
1811	3	3	0	0	1880	2	2	0	0	5257	3	1	1	0

1812	3	3	0	0	1881	2	2	0	0	5258	3	1	1	0
1813	3	3	0	0	1882	2	2	0	0	5259	3	1	1	0
1814	4	3	1	0	1883	2	2	0	0	5260	3	1	1	0
1815	4	3	1	0	1884	2	2	0	0	5261	3	1	1	0
1816	4	3	1	0	1885	2	2	0	0	5262	3	1	1	0
1817	4	3	1	0	1886	2	2	0	0	5263	3	1	1	0
1818	4	3	1	0	1887	2	2	0	0	5264	3	1	1	0
1819	4	3	1	0	1888	2	2	0	0	5265	3	1	1	0
1820	4	3	1	0	1889	2	2	0	0	5266	4	1	1	0
1821	4	3	1	0	1890	2	2	0	0	5267	4	1	1	0
1822	4	3	1	0	1891	1	1	0	0	5268	4	1	1	0
1823	4	4	0	0	1892	1	1	0	0	5269	4	1	1	0
1824	4	4	0	0	1893	1	1	0	0	5270	4	1	1	0
1825	4	4	0	0	1894	1	1	0	0	5271	4	2	1	0
1826	4	4	0	0	1895	1	1	0	0	5272	4	2	1	0
1827	4	4	0	0	1896	1	1	0	0	5273	4	2	1	0
1828	4	4	0	0	1897	1	1	0	0	5274	4	2	1	0
1829	4	4	0	0	1898	1	1	0	0	5275	4	2	1	0
1830	4	4	0	0	1899	1	1	0	0	5276	4	2	1	0
1831	4	4	0	0	1900	1	1	0	0	5277	4	2	1	0
1832	4	4	0	0	1901	1	1	0	0	5278	4	2	1	0
1833	3	3	0	0	1902	1	1	0	0	5279	4	2	1	0
1834	3	3	0	0	1903	1	1	0	0	5280	4	2	1	0
1835	3	3	0	0	1904	1	1	0	0	5281	4	2	1	0
1836	3	3	0	0	1905	1	1	0	0	5282	4	2	1	0
1837	3	3	0	0	1906	1	1	0	0	5283	4	2	1	0
1838	3	3	0	0	1907	1	1	0	0	5284	4	2	1	0
1839	3	3	0	0	1908	1	1	0	0	5285	3	2	1	0
1840	3	3	0	0	1909	1	1	0	0	5286	3	1	1	0
1841	3	3	0	0	1910	1	1	0	0	5287	3	1	1	0
1842	3	3	0	0	1911	1	1	0	0	5288	3	1	1	0
1843	3	3	0	0	1912	1	1	0	0	5289	3	1	1	0
1844	3	3	0	0	1913	1	1	0	0	5290	3	1	1	0
1845	3	3	0	0	1914	1	1	0	0	5291	3	1	1	0
1846	3	3	0	0	1915	1	1	0	0	5292	3	1	1	0
1847	3	3	0	0	1916	1	1	0	0	5293	3	1	1	0
1848	3	3	0	0	1917	1	1	0	0	5294	3	1	1	0
1849	3	3	0	0	1918	1	1	0	0	5295	3	1	1	0
1850	3	3	0	0	1919	1	1	0	0	5296	3	1	1	0
1851	3	3	0	0	1920	1	1	0	0	5297	3	1	1	0
1852	3	3	0	0	1921	1	1	0	0	5298	3	1	1	0
1853	3	3	0	0	1922	1	1	0	0	5299	3	1	1	0
1854	3	3	0	0	1923	1	1	0	0	5300	3	1	1	0
1855	3	3	0	0	1924	1	1	0	0	5301	3	1	1	0
1856	3	3	0	0	1925	1	1	0	0	5302	3	1	1	0
1857	3	3	0	0	1926	1	1	0	0	5303	3	1	1	0
1858	3	3	0	0	1927	1	1	0	0	5304	3	1	1	0
1859	3	3	0	0	1928	1	1	0	0	5305	3	1	1	0
1860	3	3	0	0	1929	1	1	0	0	5306	3	1	1	0
1861	3	3	0	0	1930	1	1	0	0	5307	3	1	1	0
1862	3	3	0	0	1931	1	1	0	0	5308	3	1	1	0
1863	3	3	0	0	1932	1	1	0	0	5309	3	1	1	0
1864	3	3	0	0	1933	1	1	0	0	5310	3	1	1	0
1865	3	3	0	0	1934	1	1	0	0	5311	3	1	1	0
1866	3	3	0	0	1935	1	1	0	0	5312	3	1	1	0
1867	3	3	0	0	1936	1	1	0	0	5313	3	1	1	0

1868	3	3	0	0	1937	1	1	0	0	5314	3	1	1	0
1869	3	3	0	0	1938	1	1	0	0	5315	3	1	1	0
1870	3	3	0	0	1939	1	1	0	0	5316	3	1	1	0
1871	3	3	0	0	1940	1	1	0	0	5317	3	1	1	0
1872	3	3	0	0	1941	1	1	0	0	5318	3	1	1	0
1873	3	3	0	0	1942	1	1	0	0	5319	3	1	1	0
1874	3	3	0	0	1943	1	1	0	0	5320	3	1	1	0
1875	3	3	0	0	1944	1	1	0	0	5321	3	0	1	0
1876	3	3	0	0	1945	1	1	0	0	5322	3	0	1	0
1877	3	3	0	0	1946	1	1	0	0	5323	3	0	1	0
1878	3	3	0	0	1947	1	1	0	0	5324	3	0	1	0
1879	3	3	0	0	1948	1	1	0	0	5325	3	0	1	0
1880	3	3	0	0	1949	1	1	0	0	5326	3	1	1	0
1881	3	3	0	0	1950	1	1	0	0	5327	3	0	1	0
1882	3	3	0	0	1951	1	1	0	0	5328	3	1	1	0
1883	3	3	0	0	1952	1	1	0	0	5329	3	1	1	0
1884	3	3	0	0	1953	1	1	0	0	5330	3	1	1	0
1885	3	3	0	0	1954	1	1	0	0	5331	3	1	1	0
1886	3	3	0	0	1955	1	1	0	0	5332	3	1	1	0
1887	3	3	0	0	1956	1	1	0	0	5333	3	1	1	0
1888	3	3	0	0	1957	1	1	0	0	5334	3	1	1	0
1889	3	2	1	0	1958	1	1	0	0	5335	3	1	1	0
1890	3	2	1	0	1959	1	1	0	0	5336	3	1	1	0
1891	3	2	1	0	1960	1	1	0	0	5337	3	1	1	0
1892	3	2	1	0	1961	1	1	0	0	5338	3	1	1	0
1893	3	2	1	0	1962	1	1	0	0	5339	3	1	1	0
1894	3	2	1	0	1963	1	1	0	0	5340	3	1	1	0
1895	3	2	1	0	1964	1	1	0	0	5341	3	1	1	0
1896	3	2	1	0	1965	1	1	0	0	5342	3	1	1	0
1897	3	2	1	0	1966	1	1	0	0	5343	3	1	1	0
1898	3	2	1	0	1967	1	1	0	0	5344	3	1	1	0
1899	3	2	1	0	1968	1	1	0	0	5345	3	1	1	0
1900	3	2	1	0	1969	1	1	0	0	5346	3	1	1	0
1901	3	2	1	0	1970	1	1	0	0	5347	3	1	1	0
1902	3	2	1	0	1971	1	1	0	0	5348	3	1	1	0
1903	3	2	1	0	1972	1	1	0	0	5349	3	2	1	0
1904	3	2	1	0	1973	1	1	0	0	5350	3	2	1	0
1905	3	2	1	0	1974	1	1	0	0	5351	3	2	1	0
1906	3	2	1	0	1975	1	1	0	0	5352	3	2	1	0
1907	3	2	1	0	1976	1	1	0	0	5353	3	2	1	0
1908	3	2	1	0	1977	1	1	0	0	5354	3	2	1	0
1909	3	2	1	0	1978	1	1	0	0	5355	3	1	1	0
1910	3	2	1	0	1979	1	1	0	0	5356	3	1	1	0
1911	3	2	1	0	1980	1	1	0	0	5357	3	1	1	0
1912	3	2	1	0	1981	1	1	0	0	5358	3	1	1	0
1913	3	2	1	0	1982	1	1	0	0	5359	3	1	1	0
1914	2	1	1	0	1983	1	1	0	0	5360	3	1	1	0
1915	2	1	1	0	1984	1	1	0	0	5361	3	1	1	0
1916	2	1	1	0	1985	1	1	0	0	5362	3	1	1	0
1917	2	1	1	0	1986	1	1	0	0	5363	1	1	0	0
1918	2	1	1	0	1987	1	1	0	0	5364	1	1	0	0
1919	2	1	1	0	1988	1	1	0	0	5365	1	1	0	0
1920	2	1	1	0	1989	1	1	0	0	5366	1	1	0	0
1921	2	1	1	0	1990	1	1	0	0	5367	1	1	0	0
1922	2	1	1	0	1991	1	1	0	0	5368	1	1	0	0
1923	2	1	1	0	1992	1	1	0	0	5369	1	1	0	0
1924	2	1	1	0	1993	1	1	0	0	5370	1	1	0	0
1925	2	1	1	0	1994	1	1	0	0	5371	1	1	0	0
1926	2	1	1	0	1995	1	1	0	0	5372	1	1	0	0
1927	2	1	1	0	1996	1	1	0	0	5373	1	1	0	0
1928	2	1	1	0	1997	1	1	0	0	5374	1	1	0	0
1929	2	1	1	0	1998	1	1	0	0	5375	1	1	0	0
1930	2	1	1	0	1999	1	1	0	0	5376	1	1	0	0
1931	2	1	1	0	2000	1	1	0	0	5377	1	1	0	0
1932	2	1	1	0	2001	1	1	0	0	5378	1	1	0	0
1933	2	1	1	0	2002	1	1	0	0	5379	1	1	0	0
1934	2	1	1	0	2003	1	1	0	0	5380	1	1	0	0
1935	2	1	1	0	2004	1	1	0	0	5381	1	1	0	0
1936	2	1	1	0	2005	1	1	0	0	5382	1	1	0	0
1937	2	1	1	0	2006	1	1	0	0	5383	1	1	0	0
1938	2	1	1	0	2007	1	1	0	0	5384	1	1	0	0
1939	2	2	0	0	2008	1	1	0	0	5385	1	1	0	0
1940	2	2	0	0	2009	1	1	0	0	5386	1	1	0	0
1941	2	2	0	0	2010	1	1	0	0	5387	1	1	0	0
1942	2	2	0	0	2011	1	1	0	0	5388	1	1	0	0
1943	2	2	0	0	2012	1	1	0	0	5389	1	1	0	0
1944	2	2	0	0	2013	1	1	0	0	5390	1	1	0	0
1945	2	2	0	0	2014	1	1	0	0	5391	1	1	0	0
1946	2	2	0	0	2015	1	1	0	0	5392	1	1	0	0
1947	2	2	0	0	2016	1	1	0	0	5393	1	1	0	0
1948	2	2	0	0	2017	1	1	0	0	5394	1	1	0	0
1949	2	2	0	0	2018	1	1	0	0	5395	1	1	0	0
1950	2	2	0	0	2019	1	1	0	0	5396	1	1	0	0
1951	2	2	0	0	2020	1	1	0	0	5397	1	1	0	0
1952	2	2	0	0	2021	1	1	0	0	5398	1	1	0	0
1953	2	2	0	0	2022	1	1	0	0	5399	1	1	0	0
1954	2	2	0	0	2023	1	1	0	0	5400	1	1	0	0
1955	2	2	0	0	2024	1	1	0	0	5401	1	1	0	0
1956	2	2	0	0	2025	1	1	0	0	5402	1	1	0	0
1957	2	2	0	0	2026	1	1	0	0	5403	1	1	0	0
1958	2	2	0	0	2027	1	1	0	0	5404	1	1	0	0
1959	2	2	0	0	2028	1	1	0	0	5405	1	1	0	0
1960	2	2	0	0	2029	1	1	0	0	5406	1	1	0	0
1961	2	2	0	0	2030	1	1	0	0	5407	1	1	0	0
1962	2	2	0	0	2031	1	1	0	0	5408	1	1	0	0
1963	2	2	0	0	2032	1	1	0	0	5409	1	1	0	0
1964	2	2	0	0	2033	1	1	0	0	5410	1	1	0	0
1965	2	2	0	0	2034	1	1	0	0	5411	1	1	0	0
1966	2	2	0	0	2035	1	1	0	0	5412	1	1	0	0
1967	2	2	0	0	2036	1	1	0	0	5413	1	1	0	0
1968	2	2	0	0	2037	1	1	0	0	5414	1	1	0	0
1969	2	2	0	0	2038	1	2	0	1	5415	1	1	0	0
1970	2	2	0	0	2039	2	3	0	1	5416	1	1	0	0
1971	2	2	0	0	2040	2	3	0	1	5417	1	1	0	0
1972	2	2	0	0	2041	2	3	0	1	5418	1	1	0	0
1973	2	2	0	0	2042	2	3	0	1	5419	1	1	0	0
1974	2	2	0	0	2043	2	2	1	1	5420	1	1	0	0
1975	2	2	0	0	2044	2	2	1	1	5421	1	1	0	0
1976	2	2	0	0	2045	2	2	1	1	5422	1	1	0	0
1977	2	2	0	0	2046	2	1	1	0	5423	1	1	0	0
1978	2	2	0	0	2047	2	1	1	0	5424	1	1	0	0
1979	2	2	0	0	2048	2	1	1	0	5425	1	1	0	0

1980	2	2	0	0	2049	2	1	1	0	5426	1	1	0	0	2036	3	3	0	0	2105	1	1	0	0	5482	1	1	0	0
1981	2	2	0	0	2050	2	1	1	0	5427	1	1	0	0	2037	3	3	0	0	2106	1	1	0	0	5483	1	1	0	0
1982	2	2	0	0	2051	2	1	1	0	5428	1	1	0	0	2038	3	3	0	0	2107	1	1	0	0	5484	1	1	0	0
1983	2	2	0	0	2052	2	1	1	0	5429	1	1	0	0	2039	3	3	0	0	2108	1	1	0	0	5485	1	1	0	0
1984	2	2	0	0	2053	2	1	1	0	5430	1	1	0	0	2040	3	3	0	0	2109	1	1	0	0	5486	1	1	0	0
1985	2	2	0	0	2054	2	1	1	0	5431	1	1	0	0	2041	3	3	0	0	2110	1	1	0	0	5487	1	1	0	0
1986	2	2	0	0	2055	2	1	1	0	5432	1	1	0	0	2042	3	3	0	0	2111	1	1	0	0	5488	1	1	0	0
1987	2	2	0	0	2056	2	2	0	0	5433	1	1	0	0	2043	3	3	0	0	2112	1	1	0	0	5489	1	1	0	0
1988	2	2	0	0	2057	2	2	0	0	5434	1	1	0	0	2044	3	3	0	0	2113	1	1	0	0	5490	1	1	0	0
1989	2	2	0	0	2058	2	2	0	0	5435	1	1	0	0	2045	3	3	0	0	2114	1	1	0	0	5491	1	1	0	0
1990	2	2	0	0	2059	1	1	0	0	5436	1	1	0	0	2046	3	3	0	0	2115	1	1	0	0	5492	1	1	0	0
1991	2	2	0	0	2060	1	1	0	0	5437	1	1	0	0	2047	3	3	0	0	2116	1	1	0	0	5493	1	1	0	0
1992	3	3	0	0	2061	1	1	0	0	5438	1	1	0	0	2048	3	3	0	0	2117	1	1	0	0	5494	1	1	0	0
1993	3	3	0	0	2062	1	1	0	0	5439	1	1	0	0	2049	3	3	0	0	2118	1	1	0	0	5495	1	1	0	0
1994	3	3	0	0	2063	1	1	0	0	5440	1	1	0	0	2050	3	3	0	0	2119	2	1	0	0	5496	1	1	0	0
1995	3	3	0	0	2064	1	1	0	0	5441	1	1	0	0	2051	3	3	0	0	2120	2	1	0	0	5497	1	1	0	0
1996	3	3	0	0	2065	1	1	0	0	5442	1	1	0	0	2052	3	3	0	0	2121	2	1	0	0	5498	1	1	0	0
1997	3	3	0	0	2066	1	1	0	0	5443	1	1	0	0	2053	3	3	0	0	2122	2	2	0	0	5499	1	1	0	0
1998	3	3	0	0	2067	1	1	0	0	5444	1	1	0	0	2054	3	3	0	0	2123	2	2	0	0	5500	1	1	0	0
1999	3	3	0	0	2068	1	1	0	0	5445	1	1	0	0	2055	3	3	0	0	2124	2	2	0	0	5501	1	1	0	0
2000	3	3	0	0	2069	1	1	0	0	5446	1	1	0	0	2056	3	3	0	0	2125	2	2	0	0	5502	1	1	0	0
2001	3	3	0	0	2070	1	1	0	0	5447	1	1	0	0	2057	3	3	0	0	2126	2	2	0	0	5503	1	1	0	0
2002	3	3	0	0	2071	1	1	0	0	5448	1	1	0	0	2058	3	3	0	0	2127	2	2	0	0	5504	1	1	0	0
2003	3	3	0	0	2072	1	1	0	0	5449	1	1	0	0	2059	3	3	0	0	2128	2	2	0	0	5505	1	1	0	0
2004	3	3	0	0	2073	1	1	0	0	5450	1	1	0	0	2060	3	3	0	0	2129	2	2	0	0	5506	1	1	0	0
2005	3	3	0	0	2074	1	1	0	0	5451	1	1	0	0	2061	3	3	0	0	2130	2	2	0	0	5507	1	1	0	0
2006	3	3	0	0	2075	1	1	0	0	5452	1	1	0	0	2062	3	3	0	0	2131	2	2	0	0	5508	1	1	0	0
2007	3	3	0	0	2076	1	1	0	0	5453	1	1	0	0	2063	3	3	0	0	2132	2	2	0	0	5509	1	1	0	0
2008	3	3	0	0	2077	1	1	0	0	5454	1	1	0	0	2064	3	3	0	0	2133	2	2	0	0	5510	1	1	0	0
2009	3	3	0	0	2078	1	1	0	0	5455	1	1	0	0	2065	3	3	0	0	2134	2	2	0	0	5511	1	1	0	0
2010	3	3	0	0	2079	1	1	0	0	5456	1	1	0	0	2066	3	3	0	0	2135	2	2	0	0	5512	1	1	0	0
2011	3	3	0	0	2080	1	1	0	0	5457	1	1	0	0	2067	3	3	0	0	2136	2	2	0	0	5513	1	1	0	0
2012	3	3	0	0	2081	1	1	0	0	5458	1	1	0	0	2068	3	3	0	0	2137	2	2	0	0	5514	1	1	0	0
2013	3	3	0	0	2082	1	1	0	0	5459	1	1	0	0	2069	3	3	0	0	2138	2	2	0	0	5515	1	1	0	0
2014	3	3	0	0	2083	1	1	0	0	5460	1	1	0	0	2070	3	3	0	0	2139	2	2	0	0	5516	1	1	0	0
2015	3	3	0	0	2084	1	1	0	0	5461	1	1	0	0	2071	3	3	0	0	2140	2	2	0	0	5517	1	1	0	0
2016	3	3	0	0	2085	1	1	0	0	5462	1	1	0	0	2072	3	3	0	0	2141	2	2	0	0	5518	1	1	0	0
2017	3	3	0	0	2086	1	1	0	0	5463	1	1	0	0	2073	3	3	0	0	2142	2	2	0	0	5519	1	1	0	0
2018	3	3	0	0	2087	1	1	0	0	5464	1	1	0	0	2074	3	3	0	0	2143	2	2	0	0	5520	1	1	0	0
2019	3	3	0	0	2088	1	1	0	0	5465	1	1	0	0	2075	3	3	0	0	2144	2	2	0	0	5521	1	1	0	0
2020	3	3	0	0	2089	1	1	0	0	5466	1	1	0	0	2076	3	3	0	0	2145	2	2	0	0	5522	1	1	0	0
2021	3	3	0	0	2090	1	1	0	0	5467	1	1	0	0	2077	3	3	0	0	2146	2	2	0	0	5523	1	1	0	0
2022	3	3	0	0	2091	1	1	0	0	5468	1	1	0	0	2078	3	3	0	0	2147	2	2	0	0	5524	1	1	0	0
2023	3	3	0	0	2092	1	1	0	0	5469	1	1	0	0	2079	3	3	0	0	2148	2	2	0	0	5525	1	1	0	0
2024	3	3	0	0	2093	1	1	0	0	5470	1	1	0	0	2080	3	3	0	0	2149	2	2	0	0	5526	1	1	0	0
2025	3	3	0	0	2094	1	1	0	0	5471	1	1	0	0	2081	3	3	0	0	2150	2	2	0	0	5527	1	1	0	0
2026	3	3	0	0	2095	1	1	0	0	5472	1	1	0	0	2082	3	3	0	0	2151	2	2	0	0	5528	1	1	0	0
2027	3	3	0	0	2096	1	1	0	0	5473	1	1	0	0	2083	3	3	0	0	2152	2	2	0	0	5529	1	1	0	0
2028	3	3	0	0	2097	1	1	0	0	5474	1	1	0	0	2084	3	3	0	0	2153	2	2	0	0	5530	1	1	0	0
2029	3	3	0	0	2098	1	1	0	0	5475	1	1	0	0	2085	3	3	0	0	2154	2	2	0	0	5531	1	1	0	0
2030	3	3	0	0	2099	1	1	0	0	5476	1	1	0	0	2086	3	3	0	0	2155	2	2	0	0	5532	1	1	0	0
2031	3	3	0	0	2100	1	1	0	0	5477	1	1	0	0	2087	3	3	0	0	2156	2	2	0	0	5533	1	1	0	0
2032	3	3	0	0	2101	1	1	0	0	5478	1	1	0	0	2088	3	3	0	0	2157	2	2	0	0	5534	1	1	0	0
2033	3	3	0	0	2102	1	1	0	0	5479	1	1	0	0	2089	3	3	0	0	2158	2	2	0	0	5535	1	1	0	0
2034	3	3	0	0	2103	1	1	0	0	5480	1	1	0	0	2090	3	3	0	0	2159	2	2	0	0	5536	1	1	0	0
2035	3	3	0	0	2104	1	1	0	0	5481	1	1	0	0	2091	3	3	0	0	2160	2	2	0	0	5537	1	1	0	0

2092	3	3	0	0
2093	3	3	0	0
2094	3	3	0	0
2095	3	3	0	0
2096	3	3	0	0
2097	3	3	0	0
2098	3	3	0	0
2099	3	3	0	0
2100	3	3	0	0

2161	2	2	0	0
2162	2	2	0	0
2163	2	2	0	0
2164	2	2	0	0
2165	2	2	0	0
2166	2	2	0	0
2167	2	2	0	0
2168	2	2	0	0
2169	2	3	0	1

5538	1	1	0	0
5539	1	1	0	0
5540	1	1	0	0
5541	1	1	0	0
5542	1	1	0	0
5543	1	1	0	0
5544	1	1	0	0
5545	1	1	0	0
5546	1	1	0	0
5547	1	1	0	0
5548	1	1	0	0
5549	1	1	0	0
5550	1	1	0	0
5551	1	1	0	0
5552	1	1	0	0
5553	1	1	0	0
5554	1	1	0	0
5555	1	1	0	0
5556	1	1	0	0
5557	1	1	0	0
5558	1	1	0	0
5559	1	1	0	0
5560	1	1	0	0
5561	1	1	0	0
5562	1	1	0	0
5563	1	1	0	0
5564	1	1	0	0
5565	1	1	0	0
5566	1	1	0	0
5567	1	1	0	0
5568	1	1	0	0
5569	1	1	0	0
5570	1	1	0	0

A.3 Synthesis Reports

```
Top Level Output File Name : pal_decoder
Design Statistics
IOs : 40
Cell Usage :
BELS : 220
  GND : 1
  INV : 19
  LUT1 : 24
  LUT2 : 17
  LUT2_L : 1
  LUT3 : 19
  LUT4 : 29
  LUT4_L : 2
  MUXCY : 61
  MUXF5 : 1
  VCC : 1
  XORCY : 45
FlipFlops/Latches : 104
  FDCE : 12
  FDR : 45
  FDRE : 26
  LD : 21
Clock Buffers : 1
  BUFGP : 1
IO Buffers : 39
  IBUF : 9
  OBUF : 30
------------------
Device utilization summary:
------------------
Selected Device : 2vp30ff896-7
Number of Slices: 66 out of 13696 0%
Number of Slice Flip Flops: 83 out of 27392 0%
Number of 4 input LUTs: 111 out of 27392 0%
Number of IOs: 40
Number of bonded IOBs: 40 out of 556 7%
IOB Flip Flops: 21
Number of GCLKs: 1 out of 16 6%
```

```
Top Level Output File Name : video_buffer
Design Statistics
IOs : 57
Cell Usage :
BELS : 60
  GND : 1
  LUT2 : 1
  LUT3 : 33
  LUT4 : 11
  MUXF5 : 9
  MUXF6 : 4
  VCC : 1
FlipFlops/Latches : 3
  FD : 3
RAMS : 32
  RAMB16_S1_S1 : 32
Clock Buffers : 2
  BUFGP : 2
IO Buffers : 47
  IBUF : 43
  OBUF : 4
------------------
Device utilization summary:
------------------
Selected Device : 2vp30ff896-7
Number of Slices: 25 out of 13696 0%
Number of 4 input LUTs: 45 out of 27392 0%
Number of IOs: 57
Number of bonded IOBs: 49 out of 556 8%
IOB Flip Flops: 3
Number of BRAMs: 32 out of 136 23%
Number of GCLKs: 2 out of 16 12%
```

```
Top Level Output File Name : segmentation
Design Statistics
IOs : 76
Cell Usage :
BELS : 477
  GND : 1
  INV : 2
  LUT1 : 6
  LUT2 : 71
  LUT2_D : 1
  LUT2_L : 1
  LUT3 : 130
  LUT3_D : 4
  LUT3_L : 2
  LUT4 : 113
  LUT4_D : 3
  LUT4_L : 10
  MUXCY : 45
  MUXF5 : 48
  MUXF6 : 17
  VCC : 1
  XORCY : 22
FlipFlops/Latches : 87
  FD : 8
  FDR : 24
  FDRE : 36
  FDSE : 2
  LDCPE : 17
RAMS : 73
  RAMB16_S1_S1 : 72
  RAMB16_S9_S9 : 1
Clock Buffers : 3
  BUFGP : 3
IO Buffers : 73
  IBUF : 56
  OBUF : 17
------------------
Device utilization summary:
------------------
Selected Device : 2vp30ff896-7
Number of Slices: 184 out of 13696 1%
Number of Slice Flip Flops: 87 out of 27392 0%
Number of 4 input LUTs: 343 out of 27392 1%
Number of IOs: 76
Number of bonded IOBs: 76 out of 556 13%
Number of BRAMs: 73 out of 136 53%
Number of GCLKs: 3 out of 16 18%
```

```
Top Level Output File Name : noise_removal
Design Statistics
IOs : 44
Cell Usage :
BELS : 266
  GND : 1
  INV : 2
  LUT1 : 16
  LUT2 : 19
  LUT3 : 37
  LUT3_L : 2
  LUT4 : 143
  LUT4_D : 1
  LUT4_L : 6
  MUXCY : 16
  MUXF5 : 6
  VCC : 1
  XORCY : 16
FlipFlops/Latches : 28
  FDE : 10
  FDRE : 18
RAMS : 2
  RAMB16_S4_S4 : 2
Clock Buffers : 1
  BUFGP : 1
IO Buffers : 42
  IBUF : 21
  OBUF : 21
------------------
```

```
Device utilization summary:
------------------
Selected Device : 2vp30ff896-7
Number of Slices: 128 out of 13696 0%
Number of Slice Flip Flops: 28 out of 27392 0%
Number of 4 input LUTs: 226 out of 27392 0%
Number of IOs: 44
Number of bonded IOBs: 43 out of 556 7%
Number of BRAMs: 2 out of 136 1%
Number of GCLKs: 1 out of 16 6%
```

```
Top Level Output File Name : morphology
Design Statistics
IOs : 44
Cell Usage :
BELS : 129
  GND : 1
  INV : 1
  LUT1 : 8
  LUT2 : 12
  LUT3 : 21
  LUT4 : 60
  LUT4_D : 1
  LUT4_L : 3
  MUXCY : 8
  MUXF5 : 5
  VCC : 1
  XORCY : 8
FlipFlops/Latches : 14
  FDE : 5
  FDRE : 9
RAMS : 1
  RAMB16_S4_S4 : 1
Clock Buffers : 1
  BUFGP : 1
IO Buffers : 42
  IBUF : 21
  OBUF : 21
------------------
Device utilization summary:
------------------
Selected Device : 2vp30ff896-7
Number of Slices: 61 out of 13696 0%
Number of Slice Flip Flops: 14 out of 27392 0%
Number of 4 input LUTs: 106 out of 27392 0%
Number of IOs: 44
Number of bonded IOBs: 43 out of 556 7%
Number of BRAMs: 1 out of 136 0%
Number of GCLKs: 1 out of 16 6%
```

```
Top Level Output File Name : lab_module
Design Statistics
IOs : 51
Cell Usage :
BELS : 3001
  GND : 1
  INV : 4
  LUT1 : 7
  LUT2 : 158
  LUT2_D : 1
  LUT2_L : 1
  LUT3 : 1224
  LUT3_D : 15
  LUT3_L : 19
  LUT4 : 364
  LUT4_D : 38
  LUT4_L : 40
  MUXCY : 126
  MUXF5 : 530
  MUXF6 : 256
  MUXF7 : 128
  MUXF8 : 64
  VCC : 1
  XORCY : 24
FlipFlops/Latches : 2331
  FDC : 48
  FDCE : 2081
  FDE : 180
  FDP : 1
  FDPE : 21
RAMS : 4
  RAMB16_S2_S2 : 1
  RAMB16_S36_S36 : 1
  RAMB16_S9_S9 : 2
Shift Registers : 8
  SRL16E : 8
Clock Buffers : 2
  BUFG : 1
  BUFGP : 1
IO Buffers : 50
  IBUF : 22
  OBUF : 28
------------------
Device utilization summary:
------------------
Selected Device : 2vp30ff896-7
Number of Slices: 2020 out of 13696 14%
Number of Slice Flip Flops: 2331 out of 27392 8%
Number of 4 input LUTs: 1879 out of 27392 6%
Number used as logic: 1871
Number used as Shift registers: 8
Number of IOs: 51
Number of bonded IOBs: 51 out of 556 9%
Number of BRAMs: 4 out of 136 2%
Number of GCLKs: 2 out of 16 12%
```

```
Top Level Output File Name : histogram
Design Statistics
IOs : 36
Cell Usage :
BELS : 74
  GND : 1
  INV : 1
  LUT1 : 6
  LUT2 : 13
  LUT3 : 4
  LUT3_L : 1
  LUT4 : 24
  LUT4_D : 1
  LUT4_L : 1
  MUXCY : 15
  VCC : 1
  XORCY : 6
FlipFlops/Latches : 54
  FDR : 24
  FDRE : 30
RAMS : 1
  RAMB16_S9_S9 : 1
Clock Buffers : 1
  BUFGP : 1
IO Buffers : 35
  IBUF : 28
  OBUF : 7
------------------
Device utilization summary:
------------------
Selected Device : 2vp30ff896-7
Number of Slices: 40 out of 13696 0%
Number of Slice Flip Flops: 54 out of 27392 0%
Number of 4 input LUTs: 51 out of 27392 0%
Number of IOs: 36
Number of bonded IOBs: 36 out of 556 6%
Number of BRAMs: 1 out of 136 0%
Number of GCLKs: 1 out of 16 6%
```

```
Top Level Output File Name : font_display
Design Statistics
```

```
IOs : 56
Cell Usage :
BELS : 677
  GND : 1
  LUT2 : 40
  LUT2_D : 9
  LUT2_L : 5
  LUT3 : 100
  LUT3_D : 9
  LUT3_L : 11
  LUT4 : 352
  LUT4_D : 16
  LUT4_L : 33
  MUXCY : 5
  MUXF5 : 94
  MUXF6 : 1
  VCC : 1
FlipFlops/Latches : 196
  FDC : 112
  FDE : 7
  FDR : 65
  FDRE : 7
  FDS : 4
  FDSE : 1
RAMS : 3
  RAMB16_S4_S4 : 2
  RAMB16_S9 : 1
Clock Buffers : 1
  BUFGP : 1
IO Buffers : 54
  IBUF : 53
  OBUF : 1
------------------
Device utilization summary:
------------------
Selected Device : 2vp30ff896-7
Number of Slices: 301 out of 13696 2%
Number of Slice Flip Flops: 196 out of 27392 0%
Number of 4 input LUTs: 575 out of 27392 2%
Number of IOs: 56
Number of bonded IOBs: 55 out of 556 9%
Number of BRAMs: 3 out of 136 2%
Number of GCLKs: 1 out of 16 6%
```

```
Top Level Output File Name : fifo_module
Design Statistics
IOs : 38
Cell Usage :
BELS : 563
  BUF : 1
  LUT2 : 1
  LUT2_D : 1
  LUT3 : 258
  LUT3_D : 4
  LUT4 : 51
  LUT4_D : 4
  LUT4_L : 3
  MUXF5 : 128
  MUXF6 : 64
  MUXF7 : 32
  MUXF8 : 16
FlipFlops/Latches : 529
  FDC : 16
  FDCE : 512
  FDP : 1
Clock Buffers : 1
  BUFGP : 1
IO Buffers : 37
  IBUF : 19
  OBUF : 18
------------------
Device utilization summary:
------------------
Selected Device : 2vp30ff896-7
Number of Slices: 440 out of 13696 3%
Number of Slice Flip Flops: 529 out of 27392 1%
Number of 4 input LUTs: 322 out of 27392 1%
Number of IOs: 38
Number of bonded IOBs: 38 out of 556 6%
Number of GCLKs: 1 out of 16 6%
```

```
Top Level Output File Name : vga
Design Statistics
IOs : 28
Cell Usage :
BELS : 198
  GND : 1
  INV : 3
  LUT1 : 31
  LUT2 : 18
  LUT3 : 3
  LUT4 : 28
  LUT4_D : 4
  LUT4_L : 1
  MUXCY : 57
  VCC : 1
  XORCY : 51
FlipFlops/Latches : 55
  FDC : 11
  FDCE : 10
  FDE : 1
  FDR : 32
  FDS : 1
Clock Buffers : 2
  BUFG : 1
  BUFGP : 1
IO Buffers : 27
  IBUF : 1
  OBUF : 26
------------------
Device utilization summary:
------------------
Selected Device : 2vp30ff896-7
Number of Slices: 45 out of 13696 0%
Number of Slice Flip Flops: 55 out of 27392 0%
Number of 4 input LUTs: 88 out of 27392 0%
Number of IOs: 28
Number of bonded IOBs: 28 out of 556 5%
Number of GCLKs: 2 out of 16 12%
```

```
Top Level Output File Name : stack
Design Statistics
IOs : 38
Cell Usage :
BELS : 566
  BUF : 1
  INV : 1
  LUT2 : 2
  LUT2_D : 1
  LUT2_L : 1
  LUT3 : 290
  LUT3_L : 2
  LUT4 : 16
  LUT4_D : 8
  LUT4_L : 4
  MUXF5 : 128
  MUXF6 : 64
  MUXF7 : 32
  MUXF8 : 16
FlipFlops/Latches : 530
  FDC : 7
  FDCE : 512
  FDE : 10
  FDP : 1
Clock Buffers : 1
  BUFGP : 1
IO Buffers : 37
  IBUF : 19
  OBUF : 18
------------------
Device utilization summary:
```

```
------------------
Selected Device : 2vp30ff896-7
Number of Slices: 446 out of 13696 3%
Number of Slice Flip Flops: 530 out of 27392 1%
Number of 4 input LUTs: 325 out of 27392 1%
Number of IOs: 38
Number of bonded IOBs: 38 out of 556 6%
Number of GCLKs: 1 out of 16 6%
```

```
Top Level Output File Name : top_module
Design Statistics
IOs : 48
Cell Usage :
BELS : 6985
  GND : 1
  INV : 112
  LUT1 : 84
  LUT2 : 1022
  LUT2_D : 2
  LUT2_L : 1
  LUT3 : 1507
  LUT3_D : 17
  LUT3_L : 4
  LUT4 : 1367
  LUT4_D : 11
  LUT4_L : 22
  MUXCY : 1599
  MUXF5 : 624
  MUXF6 : 278
  MUXF7 : 128
  MUXF8 : 64
  VCC : 1
  XORCY : 141
FlipFlops/Latches : 3260
  FD : 14
  FDC : 44
  FDCE : 2441
  FDE : 160
  FDP : 1
  FDPE : 359
  FDR : 103
  FDRE : 80
  FDS : 1
  FDSE : 2
  LD : 21
  LDCPE : 34
RAMS : 118
  RAMB16_S1_S1 : 112
  RAMB16_S36_S36 : 1
  RAMB16_S4_S4 : 2
  RAMB16_S9_S9 : 3
Shift Registers : 8
  SRL16E : 8
Clock Buffers : 5
  BUFG : 3
  BUFGP : 2
IO Buffers : 45
  IBUF : 16
  OBUF : 29
------------------
Device utilization summary:
------------------
Selected Device : 2vp30ff896-7
Number of Slices: 3389 out of 13696 24%
Number of Slice Flip Flops: 3260 out of 27392 11%
Number of 4 input LUTs: 4157 out of 27392 15%
Number used as logic: 4149
Number used as Shift registers: 8
Number of IOs: 48
Number of bonded IOBs: 47 out of 556 8%
Number of BRAMs: 118 out of 136 86%
Number of GCLKs: 5 out of 16 31%
```