RIGA TECHNICAL UNIVERSITY
Faculty of Electronics and Telecommunications
Institute of Radio Electronics

Rihards Novickis
Doctoral Student of Study Programme ”Electronics”

IMPLEMENTATION OF STEREO-VISION ALGORITHMS IN HETEROGENEOUS EMBEDDED SYSTEMS

Summary of the Doctoral Thesis

Scientific supervisors
Lead researcher Dr. sc. ing
ARTŪRS ĀBOLTIŅŠ

Researcher Dr. sc. ing
ROLANDS ŠĀVELIS

RTU Press
Riga 2022

Novickis R. Implementation of stereo-vision algorithms in heterogeneous embedded systems. Summary of the Doctoral Thesis. – Riga: RTU Press, 2022 – 44 p.

Published in accordance with the decision of the Promotion Council "Nr.206.11" in 22nd of December, 2022

https://doi.org/10.7250/9789934227455
ISBN 978-9934-22-745-5 (pdf)

DOCTORAL THESIS PROPOSED TO RIGA TECHNICAL UNIVERSITY FOR THE PROMOTION TO THE SCIENTIFIC DEGREE OF DOCTOR OF SCIENCE

To be granted the scientific degree of Doctor of Science (Ph. D.), the present Doctoral Thesis has been submitted for the defence at the open meeting of RTU Promotion Council on March 13:00, 2022 at the Faculty of Electronics and Telecommunications of Riga Technical University, 12 Azenes Street, Room 201.

OFFICIAL REVIEWERS

Associate Professor Dr. sc. ing. Dmitrijs Pikuļins
Riga Technical University, Latvia

Assistant Professor Dr. sc. ing. Paolo Meloni
University of Cagliari, Italy

Associate Professor Dr. sc. ing. Wassim Hamidouche
University of Rennes, France

DECLARATION OF ACADEMIC INTEGRITY

I hereby declare that the Doctoral Thesis submitted for the review to Riga Technical University for the promotion to the scientific degree of Doctor of Science (Pf. D) is my own. I confirm that this Doctoral Thesis had not been submitted to any other university for the promotion to a scientific degree.

Rihards Novickis . . . . . (signature)

Date: . . . . . .

The Doctoral Thesis is written in English. It consists of an Introduction; 4 Chapters; Conclusions; 76 figures; 7 tables; 10 appendices; the total number of pages is 122, including appendices. The Bibliography contains 132 titles.

CONTENTS

ABBREVIATIONS

ADC	Analog-to-Digital Converter
AI	Artificial Intelligence
AMP	Asynchronous Multi-Processing
ANN	Artificial Neural Network
ASIC	Application Specific Integrated Circuit
AXI	Advanced eXtensible Interface
CFA	Color Filter Array
CLA-FPU	Control Law Accelerator Floating Point Unit
CMA	Contiguous Memory Allocator
CMOS	Complementary Metal-Oxide Semiconductor
CNN	Convolutional Neuron Networks
CPU	Central Processing Unit
DMA	Direct Memory Access
DNN	Deep Neural Network
DSP	Digital Signal Processing
ELF	Executable Linked Format
ERDF	European Regional Development Fund
FFNN	Feed Forward Neural Network
FIFO	First-In-First-Out
FPGA	Field Programmable Gate Array
HLS	High-Level Synthesis
HPC	High Performance Computing
HSoC	Heterogeneous System on Chip
I/O	Input/Output
IC	Integrated Circuit
IoT	Internet of Things
IP	Intellectual Property
ISA	Instruction Set Architecture
MM	Memory-Mapped
MPU	Micro Processing Unit
NN	Neural Network
OCRAM	On-Chip Random Access Memory
OS	Operating System
PC	Program Counter
PCIe	Peripheral Component Interconnect Express
PWM	Pulse Width Modulation
RT	Real-Time
RTL	Register Transfer Level
SCU	Snoop Control Unit
SIP	Silicon Intellectual Property
SIPO	Serial-In-Parallel-Out

SoA	State of Art
SoC	System on Chip
SRAM	Static Random Access Memory
ST	Streaming
VHDL	Very High Speed Integrated Circuit Hardware Description Language
WTA	Winner-Take-All

GENERAL DESCRIPTION OF THE WORK

The Urgency of Subject Matter

The Thesis addresses the relationship between computerized perception and increasingly complex *Heterogeneous System on Chip* (HSoC) technologies, in particular: the on-chip hardware and software co-architectures, implementation of stereo-vision and *Artificial Intelligence* (AI) algorithms and associated real-time considerations.

The developed methods and algorithms closely relate to Moore's law [1] that has become something more than just a prediction as it guides the intimate relationship between innovation and modern semiconductor companies. The public expects improved performance and innovative features while investors anticipate the new technologies.

The contemporary *System on Chip* (SoC) technologies bare potential of solving the perception challenge in a commercially feasible way, but this still requires solving challenges of system design involving multiple computational paradigms and abstractions ranging from *Register Transfer Level* (RTL) design to the level of *Operating System* (OS) and even extending to an organization of multiple systems. Such aspects as algorithm partitioning across different computational paradigms, reliable on-chip communications' architecture and compliance with real-time control system performance are still major challenges.

The Objective of the Thesis

The primary aim of this Thesis is to **develop and improve computer vision development techniques and methods for HSoC technology.** The developed methods aspire to conceive a new type of engineers and researchers who would excel at multiple aspects of HSoC design and therefore would be capable of solving challenges utilizing multiple sets of complementary technologies. Several tasks have been defined in order to reach the aim of this paper:

- identify methods for complementing RTL and software-based computing paradigms;
- design software architectures and tools for utilizing heterogeneous SoC technology;
- design heterogeneous approach to the implementation of image processing pipelines;
- implement and conduct experimental research on the developed tools and algorithms;
- draw conclusions about the results of this Thesis.

Thy Methodology of Research

The first tasks of the Thesis are accomplished by an analytical research. Analytic methods are used to review the existing literature in the field of the correspondence matching and to develop HSoC-based image processing pipeline. Further research and design utilizes engineer-

ing methods and implemented using *Very High Speed Integrated Circuit Hardware Description Language* (VHDL), C and Python languages. The developed tools and RTL are experimentally tested and validated. The tests measure efficiency, accuracy and real-time characteristics of the developed tools and methods.

Scientific Novelty and Main Results

The Thesis results in novel methods and tools for HSoC-based architecture implementation and deployement for real-time control applications and image processing pipelines. The developed tools and system architecture deployement software is available online.

Compage* and *icom. Modular software component management frameworks suitable for embedded systems. These frameworks collaboratively simplify prototyping activities for systems where *Real-Time* (RT) performance is essential. *Compage* is a software component management framework that has a small footprint and provides means of configuring and replicating software components, while *icom* ensures seamless switching between zero and deep copy communication approaches and supports the standard PUSH-PULL, PUBLISH-SUBSCRIBE and REQUEST-REPLY communication paradigms.

***Asynchronous Multi-Processing* (AMP) subsystem**. A novel approach, where at least one of the processor cores is dedicated to a RT application, while the rest of the system runs high-level OS. The developed solution aspires to provide the best of the two worlds: real-time processing capabilities provided by the AMP core and *Field Programmable Gate Array* (FPGA) and functionality of the Linux software stack. The interface to the AMP subsystem is implemented as a Linux driver, and it ensures (1) control of the AMP core; (2) loading baremetal *Executable Linked Format* (ELF) applications for execution on the AMP core, including the configuration of the virtual memory; (3) access to the diagnostics of the application's execution; (4) on-the-fly configuration of the application; and (5) communication with the AMP core in a form of *stdin*, *stdout*, *stderr* streams.

Method for maximizing the throughput of *Feed Forward Neural Network* (FFNN) processing pipeline. The developed novel approach enables the design of a throughput-optimized processing pipeline for FFNNs. It reexamines the *Neural Network* (NN) implementation challenge and restates the description of FFNNs to adapt it for pipelining. The network is split into elementary layers, where each layer is associated with an abstract resource. These resources can be allocated to each of the layers and determine the delay characteristics of the whole pipeline. The approach accommodates a tool, which converts topology into high-level code for *High-Level Synthesis* (HLS) pipeline. The tool's inputs are the topology of the network and target

latency for a single stage of the pipeline (elementary layer). The developed method is suitable for virtual sensor implementation, especially when a high sample rate is required.

HSoC-based architecture for computer vision processing. A HSoC-based image processing pipeline that combines Linux-based *Micro Processing Unit* (MPU) for system control and FPGA for processing. The developed correspondence matching pipeline is fully implemented in the programmable logic. It consists of deinterleaving, Bayers pattern interpolation, lens distortion correction, rectification, feature extraction and correspondence matching. The processing system is used mainly for accelerator control, image acquisition via *Peripheral Component Interconnect Express* (PCIe) bus and transfer of the resulting image to the demonstrator system. One of the most formidable achievements is the design of a fully pipelined image transformation circuit, which corrects lens distortions and performs perspective transformation of images. The design has been accommodated with a novel approach for parallel access of N-dimensional data in digital logic while conserving memory utilization.

Thesis Statements to Be Defended

1. The developed Asynchronous Multi-Processing subsystem solution enables exploiting functionality of a modern operating system while supporting real-time processing with control-loop latencies' jitter not exceeding a standard deviation of 0.1 ms.
2. The developed throughput-optimized FFNN design method offers better performance when compared to neiron-centric approaches (2–30 times) and methods following RTL design flow (>1000 times).
3. The HSoC technology is suitable for implementing image processing pipelines in a fully pipelined manner achieving performance appropriate for modern real-time systems, whith a control loop's length less than 50 ms for <1.5 MP images.

Practical Value and Approbation

The developed Linux drivers and libraries, which are made available online under MIT license, provide means for implementing on-chip communication between FPGA and software in HSoC devices. The designed frameworks for system architecture implementation – *compage* and *icom* – have been used to deploy AI perception pipeline onto an autonomous vehicle and also utilized for implementing software control architecture of autonomous drones. Furthermore, the developed image processing know-how and developed accelerators are currently being commercialized under *European Regional Development Fund* (ERDF) project "Silicon IP Design House" No. KC-PINOCHET-2020/12.

The doctoral Thesis has been developed in connection with several projects implemented at

the Institute of Electronics and Computer Science, Latvia:

- National Research Programme "Cyber-physical systems, ontologies and biophotonics for safe & smart city and society" (SOPHIS). Project No.1. "Development of technologies for cyber-physical systems with applications in medicine and smart transport".
- Horizon 2020 ECSEL project "Integrated Components for Complexity Control in Affordable Electrified Cars" (3Ccar), G.A. 662192;
- Horizon 2020 ECSEL project "Intelligent Motion Control Platform for Smart Mechatronic Systems" (I-MECH), G.A. 737453;
- Horizon 2020 ECSEL project "Programmable Systems for Intelligence in Automobiles" (PRYSTINE), G.A. 783190;
- Horizon 2020 ECSEL project "Advanced packaging for photonics, optics and electronics for low cost manufacturing in Europe" (APPLAUSE), G.A. 826588;
- Horizon 2020 ECSEL project "Framework of Key Enabling Technologies for Safe and Autonomous Drones" (COMP4DRONES), G.A. 826610.

The results of the Thesis have been described in several papers [2]–[8]. These results have been promoted in the following international conferences:

- 2018 5th International Conference on Control, Decision and Information Technologies (CoDIT), Thessaloniki, Greece;
- 2019 24th IEEE International Conference on Emerging Technologies and Factory, Zaragoza, Spain;
- 2020 17th Biennial Baltic Electronics Conference (BEC), Tallinn, Estonia;
- 2020 23rd Euromicro Conference on Digital System Design (DSD), Kranj, Slovenia;
- 24th Euromicro Conference on Digital System Design (DSD), Palermo, Italy.

Structure of the Thesis

The Thesis comprises 121 pages, 76 figures, 7 tables, refers to 132 sources of literature and has 10 appendixes. The Thesis consists of five main sections. Section 1 summarises digital computing paradigms and principles, and Section 2 depicts image processing principles and different stereo-vision methods and techniques. Section 3 describes the design of HSoC-based system architectures and their real-time characteristics. Section 4 deals with the challenge of adapting and implementing computer vision algorithms in HSoC-based technologies. Section 5 presents the conclusions of the Thesis.

1. TECHNOLOGICAL CONTEXT

1.1 Processing Paradigms

The computational characteristics and requirements of different applications have led to the specialization of hardware and involves the trade-off between computational platform's universality and performance, thus leading to a range of technologies: general-purpose processors, domain-specific processors, application-specific processors, programmable logic and specialized integrated circuits.

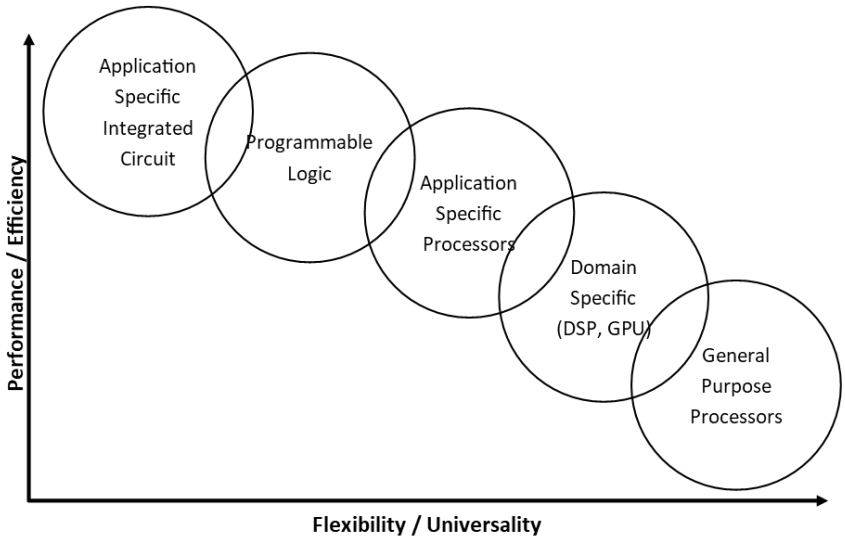


Fig. 1.1. Comparison between implementation platforms.

A microprocessor's functionality is fully characterized by the instruction set that it is capable of executing, which is also called the *Instruction Set Architecture* (ISA). ISA has been defined as a contract between software and hardware, which allows independent development of both. Most often, it defines a set of instructions called *assembly instructions*, which are then provided by the microarchitectures.

The processor instruction cycle encapsulates the intrinsic advantage of the sequential processing paradigm. Most notably, the system is universal and can be used to implement almost any kind of algorithm. This feature has led to a wide application of processors ranging from simple keyboard controllers to massive *High Performance Computing* (HPC) servers. Nevertheless, the intrinsic weaknesses of this processing paradigm are the memory access irregularities and inter-instruction dependence, both of which can lead to pipeline stalls.

Another essential technology is the FPGAs, which have become one of the key mediums for

digital circuit implementation. FPGA is a prefabricated silicon device that can be electrically programmed to become almost any kind of digital circuit or system [9]. Simplified FPGA structure is illustrated in Fig. 1.2, it consists of general logic, memory, *Digital Signal Processing* (DSP) blocks, routing fabric and programmable *Input/Output* (I/O) blocks.

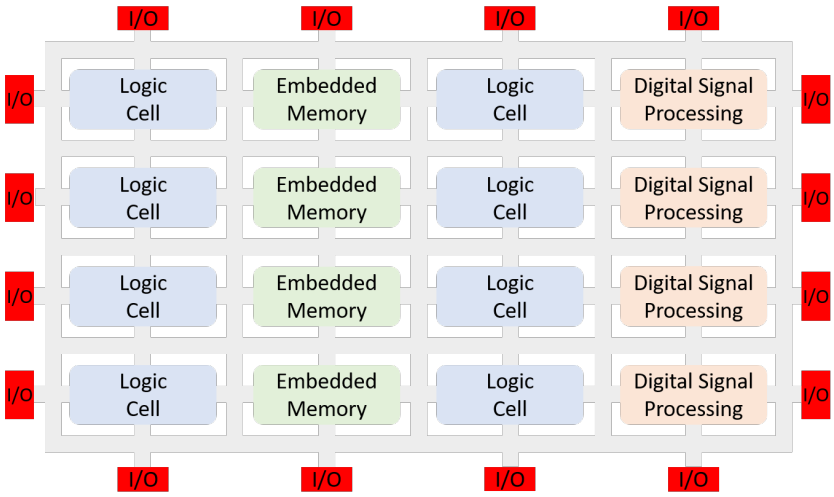


Fig. 1.2. Simplified FPGA structure.

FPGA development often involves balancing between performance and resource utilization. Unlike processor systems where software directs the execution of an algorithm, in FPGAs, different physical parts of the chip can be dedicated to a specific task. One of the main performance-increasing techniques for any digital computing system is *pipelining*, which is a special kind of concurrency increasing the system's performance by overlapping the processing of several tasks [10].

The programmability and monetary considerations have led to numerous applications of the FPGA technology: compensation for non-existing *Integrated Circuit* (IC)s, prototyping/emulation of *Application Specific Integrated Circuits* (ASICs), low-latency customized communication and data routing, data preprocessing and analysis, HPC, *Deep Neural Networks* (DNNs) and many others.

1.2 Heterogeneous System on Chip

Core enablers for reducing energy consumption and enabling personalized mobile computing are SoC and their extension HSoC technologies. An SoC integrates a range of *Silicon Intellectual Property* (SIP) cores designed by different teams around the world.

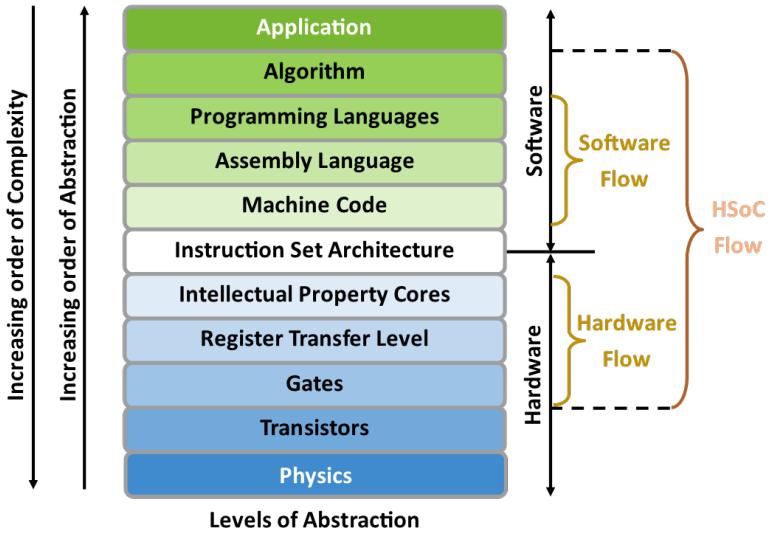


Fig. 1.3. Levels of abstraction for an electronic computing system.

The simultaneous utilization of diverse computing paradigms presents the challenge of harmonizing different design flows. Figure 1.3 represents a variety of abstraction levels that are a part of any semiconductor-based application. The classical FPGA development flow is not concerned with software. The degree of HSoC integration permits the development of more customized hardware-software solutions. These solutions require multi-disciplinary expertise concerning not only implementation but also the algorithm that impacts all stages of the development and calls for the consideration of such factors as arithmetic precision of the accelerator, locality and pipelining potential of the algorithm, choice for on-chip data movement, the robustness of the kernel driver implementation, efficiency of kernel/user API, real-time characteristics of the system, etc.

1.3 Linux Operating System

Linux is a very influential OS and is used by systems of varying scope, ranging from large-scale servers to modern embedded systems [11]. Linux has considerably low requirements, supports a wide range of customization, provides multi-threading and has a variety of available open-source software which can speed up development. OS is responsible for the elementary use and management of resources available to the system, i.e. processor execution time, memory, storage elements, connected devices, network interfaces, etc.

One of the most critical considerations for HSoC-based designs is the concept of virtual memory, which isolates applications, provides means for efficient inter-process communica-

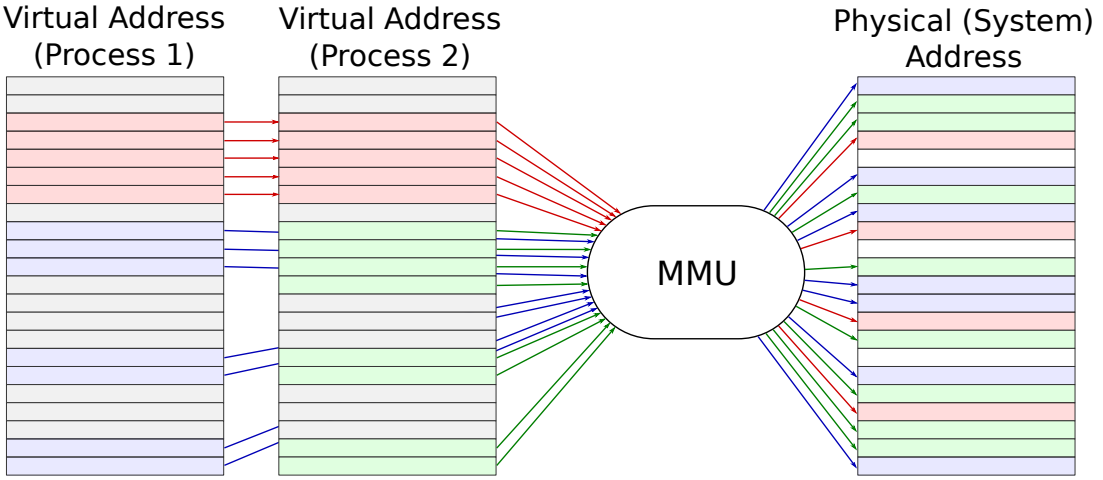


Fig. 1.4. Simplified example of page-based virtual-physical address space mapping for two processes. Pages illustrated in red denote kernelspace, while blue and green pages denote two distinct applications.

tion, enables requesting more dynamic memory than there is available (via page-swapping) and improves security in general [12]. Page-based memory virtualization ensures that physically non-contiguous memory appears contiguous for software as shown in Fig. 1.4. Nevertheless, other bus masters (apart from CPU) operate using physical address space. While some masters (such as PCIe) can reserve memory at startup, it still presents a challenge for HSoC-based system development when both *Central Processing Unit* (CPU) and FPGA share access to the same system memory.

A complete HSoC-based system design also implies kernel-level development because kernelspace accounts for communication between software and designed hardware (accelerators) and implements the interface for the userspace. To properly interface the custom hardware accelerator with the actual application in the context of HSoC technology, one has to touch upon the whole level of software abstractions: ranging from the development of the device driver to the application.

2. IMAGE PROCESSING ALGORITHMS

2.1 Image Formation and Correspondence Algorithms

2.1.1 General Projective Camera and Lens Distortions

In principle [13], a camera is a mapping between the 3D world and a 2D image. Modelling of all cameras is based on a notion of a *general projective camera*, which can be conveyed as a matrix **P** which maps world points Q to the image points q.

Let's consider plane at $\mathrm{Z} = \mathrm{f}$, which is called *image plane* or *focal plane*. The pinhole camera model determines that a point in space $\mathrm{Q} = (\mathrm{X}, \mathrm{Y}, \mathrm{Z})^T$ is mapped to the point on the image plane where a line from Q to the centre of the projection meets the image plane, as shown in Fig. 2.1.

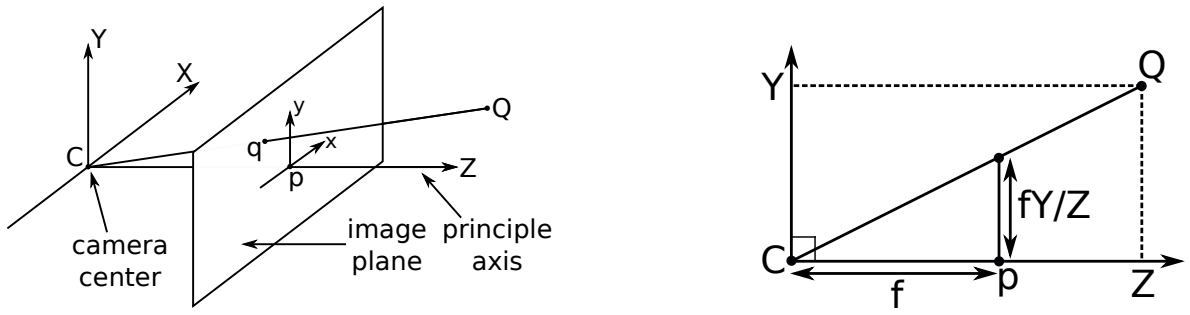


Fig. 2.1. Pinhole camera geometry (image plane is mirrored towards the scene).

If the world and image points are represented in homogeneous coordinates and also considering transformation of the origin point, the point projection can be expressed in terms of matrix multiplication as:

$$\begin{pmatrix} x \\ y \\ \omega \end{pmatrix} = \begin{pmatrix} \mathrm{f} & 0 & p_x & 0 \\ 0 & \mathrm{f} & p_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{2.1}$$

Often the matrix is rewritten by denoting matrix *K*, which is called the *camera calibration matrix*, thus the description of the projection reduces to its concise form:

$$q = K[\mathrm{I}|\mathrm{O}]\mathrm{Q}. \tag{2.2}$$

Further, image processing models assume that cameras obey a *linear* projection model where a straight line in the world results in a straight line in the image [14]. Unfortunately, lenses, which focus light onto the camera pixel matrix, have noticeable lens distortions that manifest as a visible curvature in the projection of straight lines, illustrated in Fig. 2.2.

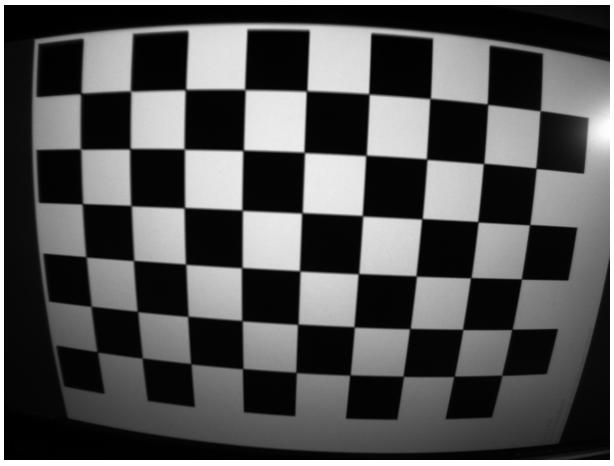


Fig. 2.2. An example of lens distortion effect on the calibration image for bumblebee stereo camera.

Unless these distortions are corrected, it is impossible to create highly accurate photo-realistic reconstructions [14]. The radial (lens) distortions can be modelled as:

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = L(\widetilde{r}) \begin{pmatrix} \widetilde{x} \\ \widetilde{y} \end{pmatrix}, \tag{2.3}$$

where

- $(\widetilde{x}, \widetilde{y})$ – the ideal image position (which obeys linear projection);
- (x_d, y_d) – the actual image position, after radial distortion;
- $\widetilde{r}$ – the radial distance $\sqrt{\widetilde{x}^2 + \widetilde{y}^2}$ from the centre for radial distortion;
- $L(\widetilde{r})$ – a distortion factor, which is a function of the radius $\widetilde{r}$ only.

In pixel coordinates the correction can be rewritten as follows:

$$\begin{array}{l} \widetilde{x} = x_c + (x - x_c)L(\widetilde{r}) \\ \widetilde{y} = y_c + (y - y_c)L(\widetilde{r}) \end{array}, \tag{2.4}$$

where x, y are the "required" coordinates of the corrected image, and $\widetilde{x}, \widetilde{y}$ are the coordinates in the distorted (input) image.

The distortion factor is an arbitrary function defined for positive values and at the center of the distortion $L(0)$ is 1. Distortion factor can be given by a Taylor expansion

$$L(r) = 1 + k_1 r + k_2 r^2 + k_3 r^3 + ..., \tag{2.5}$$

where the coefficients for radial correction $k_1, k_2, k_3, ...$ are considered a part of the interior calibration of the camera [13].

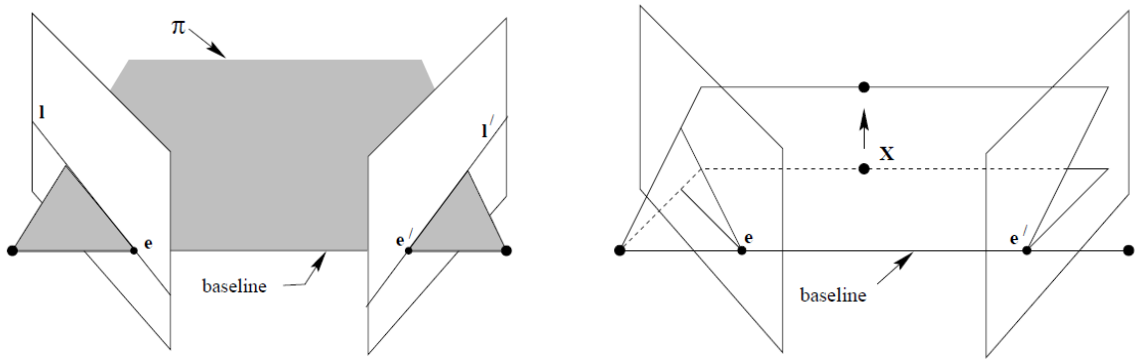
2.1.2 Epipolar Geometry

An important part of any use case involving 3D reconstruction from images involves epipolar geometry that relates two views based only on the cameras' internal parameters and relative pose. This relationship is encapsulated by a *fundamental matrix* F. F is a 3×3 matrix, and it has such property that if the point in 3D space X projects as x in the first view and as x' in the second view, then the following relationship is satisfied:

$$x'Fx = 0. \tag{2.6}$$

The epipolar geometry between two views essentially describes the geometry of intersection of the image planes with the pencil of planes having the baseline as an axis [13]. The epipolar geometry is motivated by the search for correspondences across the views that lay on a plane in 3D space and corresponds to the lines in the images, called *epipolar lines*.

Let's consider Fig. 2.3. The image points x, x', space point X, and camera centres are coplanar. Denoting the this plane as π, the rays back-projected from the x and x' intersect at X are coplanar, lying in π.



(a) Camera baseline intersecting the image planes at the epipoles.

(b) Family of planes obtained by rotating the epipolar plane around the baseline.

Fig. 2.3. Epipolar geometry [13].

When the x is known, the epipolar geometry constrains x' to the plane π, thus localizing the possible correspondences. Fig. 2.3(b) shows the rotation of the π plane, thus resulting in the rotation of the cross-sections and epipolar lines as well.

These epipolar lines can be "straightened" by applying image rectification. Rectification is a process of resampling pairs of stereo images to produce a pair of *matched epipolar projections*, where epipolar lines run parallel with the x-axis; thus, the matching procedure can be performed

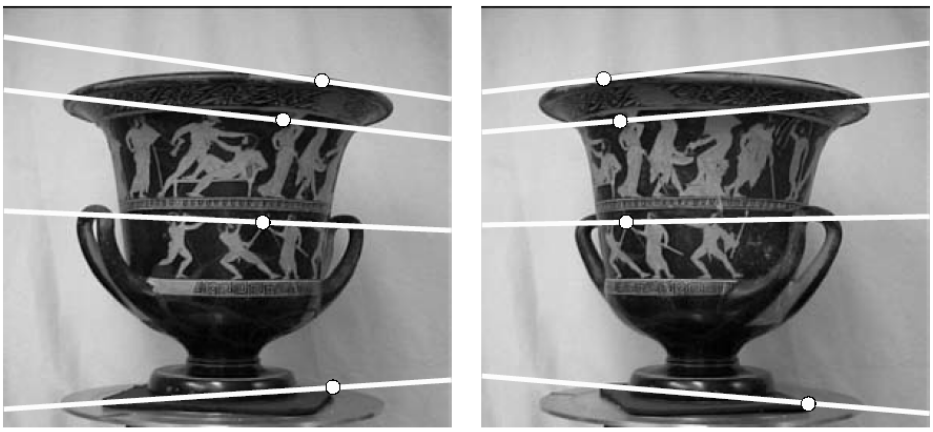


Fig. 2.4. An example of stereo image correspondences and their respective epipolar lines [13].

for consequent disparities between the images in the x-direction only, essentially omitting y disparities.

2.1.3 Stereo Correspondence

Stereo correspondence has been and still continues to be one of the most heavily investigated topics in computer science. The term *disparity* was first introduced in the human vision literature [15] to describe the difference in location of corresponding features seen by the left and right eyes.

There are two broad classes of stereo algorithms *local* and *global*. In the *local* (window-based) algorithms the disparity computation at a given point depends only on intensity values within a finite window, while the *global* ones make an explicit smoothness assumptions and then solve an optimization problem. In between of these classes there are certain iterative algorithms that do not explicitly state a global minimization function.

Szeliski [13] points out the subset of the following four algorithmic ”building blocks” from which a large set of algorithms can be constructed [16]: 1) matching cost computation, 2) cost aggregation, 3) disparity computation, 4) disparity refinment.

The solution of the correspondence problem necessitates establishing a metric to compare individual pixels of the stereo image pair. The metrics are based on either grayscale values of the pixels or post-processed features while varying in their ability to expose pixel uniqueness, receptiveness to camera gain or bias, and their computational complexity. Such costs as census transform, are not only insensitive to the camera gain and bias differences but are good candidates for implementation in digital logic. The drawback of such algorithms is their heightened

sensitivity to noise.

Aggregation is done by summing all matching costs over square windows with a constant disparity. In local methods, the emphasis is on the matching cost computation and cost aggregation steps. Computing the final disparities is trivial: simply the disparity is associates pixel with the minimum aggregation cost value, i.e. essentially performing a local *Winner-Take-All* (WTA). The weakness of such methods is that the uniqueness of matches is only enforced for one image, i.e. the *reference image*, while points in the other image might match multiple points [14].

2.2 AI-Based Algorithms

An NNs is a system that is designed to model how the brain performs a particular task or function of interest [17]. A network can be split into fundamental information-processing units – neurons, which form the basis for designing artificial neural networks. The block diagram in Fig. 2.5 shows neuron's mathematical model. Neuron's inputs x_i are multiplied by coefficients w_{ki}, referred to as weights and summed up together with a bias b_k. This sum is passed to an activation function φ, which is used to normalize neuron's output, k and i designate a specific neuron and its input.

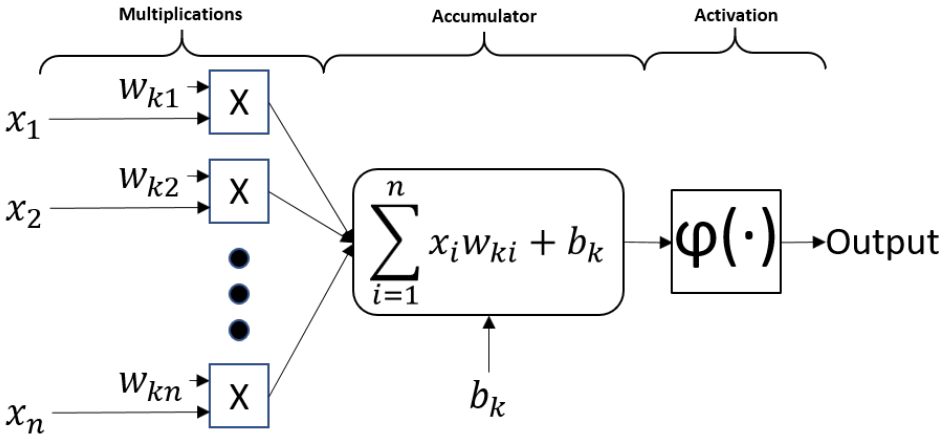


Fig. 2.5. Structure of a single neuron.

A type of frequently used neural network is constructed by arranging neurons in layers, where all neurons in every layer are connected to each neuron in the adjacent forward layer, i.e. fully connected feed-forward network. This type of network is illustrated in Fig. 2.6.

Ever since the early formalization of NNs [18], a significant effort has been made and various paradigms used to adopt different NN structures for digital circuit implementation. For example,

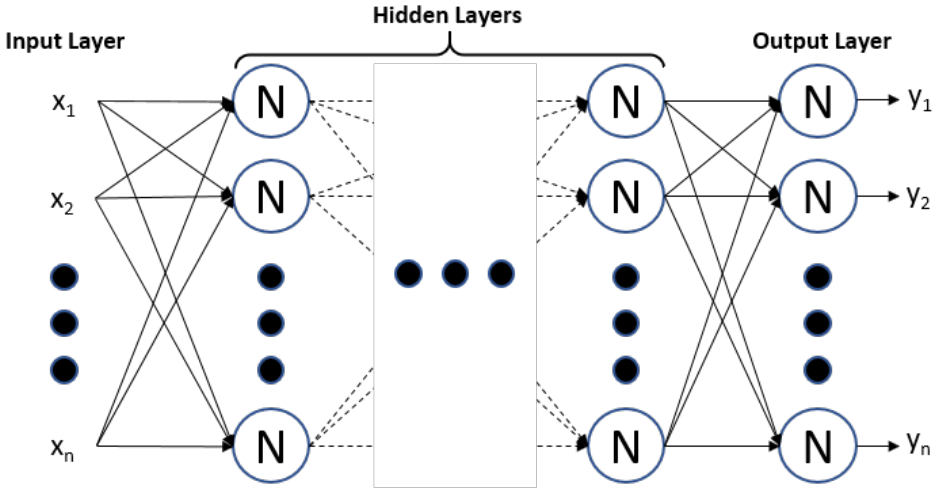


Fig. 2.6. General structure of a feed forward neural network.

Convolutional Neuron Networkss (CNNs) are widely used for image recognition and classification, although classification itself is carried out by a fully connected FFNN. Different kinds of paradigms ranging from co-processor systems [19] to OpenCL-based solutions [20]–[24] have been used to find the optimal trade-off between resource use, latency, and throughput.

The intrinsic programmable logic's parallel nature suggests its suitability for the implementation of FFNNs. Although different architectural approaches and design choices have been investigated, FFNN implementations face a major challenge of limited hardware resources. Furthermore, expansion of the NN topologies [25] and saturation of the manufacturing process improvement [26] suggest the persistence of the resource challenge.

3. HETEROGENEOUS COMPUTING ARCHITECTURES

3.1 Heterogeneous Computing Based on Direct Memory Access

HSoC technology entices implementation of algorithms where every subtask executes on the most appropriate processing technology, i.e. processor is better suited for out-of-context tasks, such as decision making and control, and FPGAs excel at high-throughput number crunching and relatively local algorithms, for example, per-pixel operations, convolution and feature extraction.

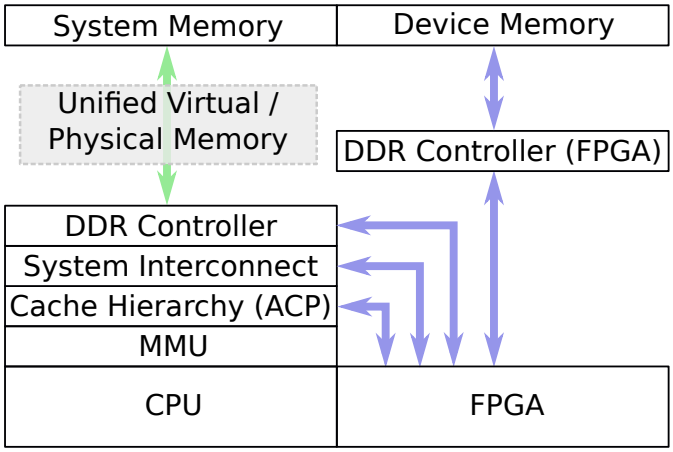


Fig. 3.1. Memory model and possibilities of modern HSoCs, where FPGA has access to separate memory, the shared system memory, internal interconnection logic and cached processor memory.

Many modern HSoC devices provide a unique choice for memory coordination, as shown in Fig. 3.1. FPGA may 1) implement its own DDR controller and utilize large amounts of isolated memory; 2) have direct access to the processing system's DDR controller enabling performant yet non-cached data transfer; 3) have access to the system's interconnect, also enabling the utilization of processing system's *On-Chip Random Access Memory* (OCRAM); and 4) even access the cache-coherency ports that ensure memory access coherency on a hardware level. The aforementioned aspects provide a unique opportunity for the system designer while simultaneously creating a challenge for abstracting and controlling the accelerators from the userspace application.

To address the fundamental issue of HSoC on-chip communications, a *Direct Memory Access* (DMA)-based high-bandwidth communications architecture was developed for establishing the means of communicating between FPGA and software in an embedded Linux environment. The developed solution targets the Linux memory fragmentation issue and adopts the Linux ker-

nel's *Contiguous Memory Allocator* (CMA) feature. Furthermore, the data paths were benchmarked for Cyclone V SoC device, as this data was still lacking in the scientific literature [27]–[29]. The developed modules and libraries have been made available to the public under MIT license[1].

An important aspect of the particular system's structure is the Level-3 (L3) interconnect, which is a central switch that routes data between memory, FPGA fabric, processor and peripherals [30]. The considerations of DMA control, memory virtualization and memory fragmentation constituted the development of the modular FPGA Master-based architecture shown in Fig. 3.2.

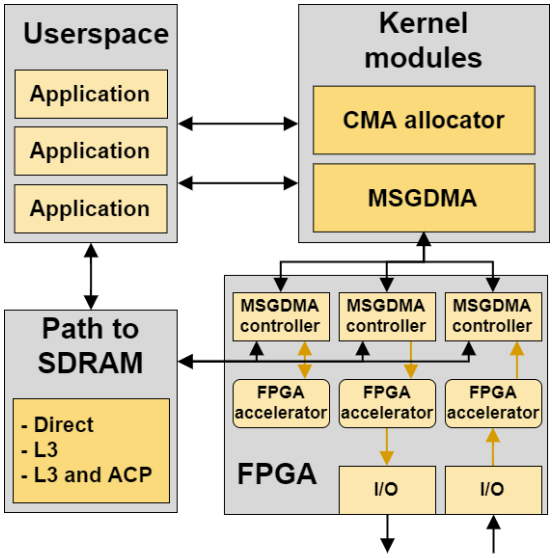


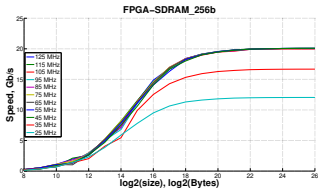
Fig. 3.2. Conceptual design of FPGA Master-based architecture.

Finally, the throughput measurement procedure utilized the VEEK-MT-C5SoC development board for all FPGA master communication bridges by adjusting the interface bus width, transaction size and FPGA clocking frequency. Table 3.1 shows the maximum (saturated) throughput for each of the data path configurations and Fig. 3.3 shows the nature of throughput measurements at the widest configurable bus widths and varying FPGA clock frequencies. The power consumption was estimated in different setup where tests were run continuously for 300 seconds, and the power was estimated using the onboard power monitor with a total unadjusted error of $\pm 1.0\,\%$, but no distinguishable results were found. Power consumption in the idle state was 8.11 W, but for any of the data path scenarios, it was always 8.18 W within the measurement error range.

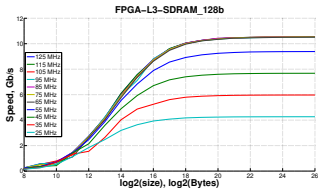
[1] http://git.edi.lv/rihards.novickis/FPSoC_Linux_drivers

Table 3.1. Throughput of simultaneous read/write transactions for all communication interface configurations

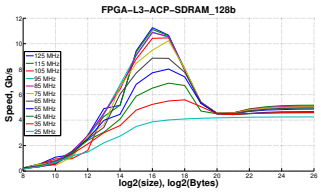
Data path	Bus width	Maximum throughput	Saturation frequency
FPGA-L3-SDRAM	32 bits	5.05 Gbps	120 MHz
	64 bits	10.10 Gbps	120 MHz
	128 bits	10.52 Gbps	65 MHz
FPGA-L3-ACP-SDRAM	32 bits	6.90 Gbps	-
	64 bits	8.64 Gbps	120 MHz
	128 bits	11.26 Gbps	90 MHz
FPGA-SDRAM	32 bits	7.52 Gbps	-
	64 bits	14.64 Gbps	-
	128 bits	17.68 Gbps	80 MHz
	256 bits	20.08 Gbps	45 MHz



(a) FPGA-SDRAM interface at 256 bit bus configuration.



(b) FPGA-L3-SDRAM interface at 128 bit bus configuration.



(c) FPGA-L3-ACP-SDRAM interface at 128 bit bus configuration.

Fig. 3.3. Throughput measurements for different FPGA master communication paths.

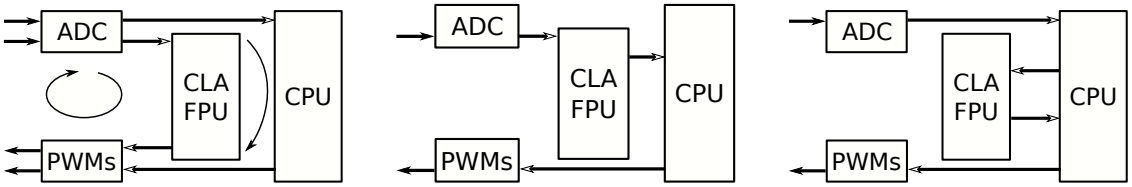
In summary, the examined technology (Intel Cyclone V SoC) can sustain on-chip communications bandwidth of 20.08 Gbps, which is suitable for real-time image co-processing. Figure 3.3(c) shows a curious result where communication bandwidth hits a peak and then drops to saturation. This characteristic is explained by the data requests ”asking” for data that still resides in the cache, which is 512 KB for the Intel Cyclone V SoC.

3.2 Asynchronous Multi-Processing Subsystem

A considerable contribution of this Thesis is the novel employment of heterogeneous architectures for real-time control loop systems by using an approach based on the AMP subsystem. While FPGA technology is a well-suited medium for real-time control due to its determinism, acceleration of such computational tasks where the processing involves leaping across memory can be problematic or even unsuitable.

Simplified examples of different embedded control loop configurations is shown in Fig. 3.4. By acknowledging the requirements of latency, reliability, supporting local feedback loops and

application-level customization, a concept of modular signal processing unit for motion control applications based on HSoC technology was conceived [31]. The developed AMP subsystem is a part of this modular processing unit.



(a) Running parallel control loops. **(b)** Preprocessing input data. **(c)** Co-processing with the main CPU.

Fig. 3.4. Texas instrument embedded control loop's configuration examples using their *Control Law Accelerator Floating Point Unit* (CLA-FPU) [32]. The *Analog-to-Digital Converter* (ADC) measures feedback signals and triggers control-loops execution, the *Pulse Width Modulation* (PWM) represents the outputs of the system.

The developed overall AMP solution aspires to provide the best of the two worlds: real-time processing capabilities provided by the AMP core and FPGA and functionality of the Linux software stack. Figure 3.5 illustrates the overall concept of the developed AMP subsystem; in this dual-core processing system, CPU0 executes the operating system (Linux) while the CPU1 (AMP subsystem) cooperates with FPGA and collaboratively executes RT tasks.

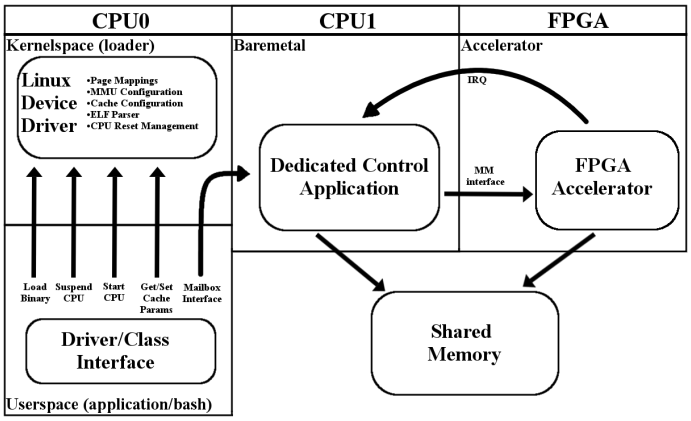


Fig. 3.5. Real-time application subsystem.

3.3 Approach to Management of Software Components

A novel component-based software architecture concept has been conceived, which already has been successfully applied for autonomous vehicles and drone systems. While developing a complex and evolving system, it may be necessary to have an architecture that facili-

tates effortless software component management and provides convenient means for their inter-communication. This approach resembles a “blackboard” pattern from the software development theory [33].

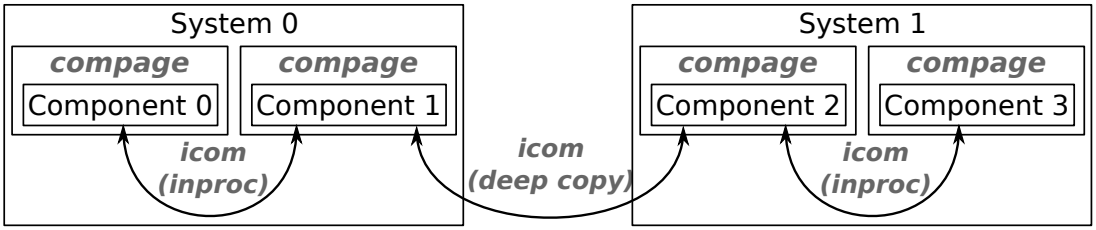


Fig. 3.6. An approach to the management of software components.

Figure 3.6 represents the principle of the developed system’s component management and inter-communication solution. The developed approach utilizes two modular frameworks:

- ***compage*** (component management framework) ensures component management within a single system, including the initialization and execution of the components in separate threads or processes. The framework also provides the means of generating and applying component configuration using *.ini* files. The developed framework is made available to the public[2].
- ***icom*** (component communication framework) ensures coherent communication between different software components by providing various underlying communication mechanisms, thus enabling zero-data copy whenever possible. Essentially, if software components execute on a single system, they can share data references and avoid data duplication, e.g. utilizing the double buffer technique [34]. The developed framework is made available to the public[3].

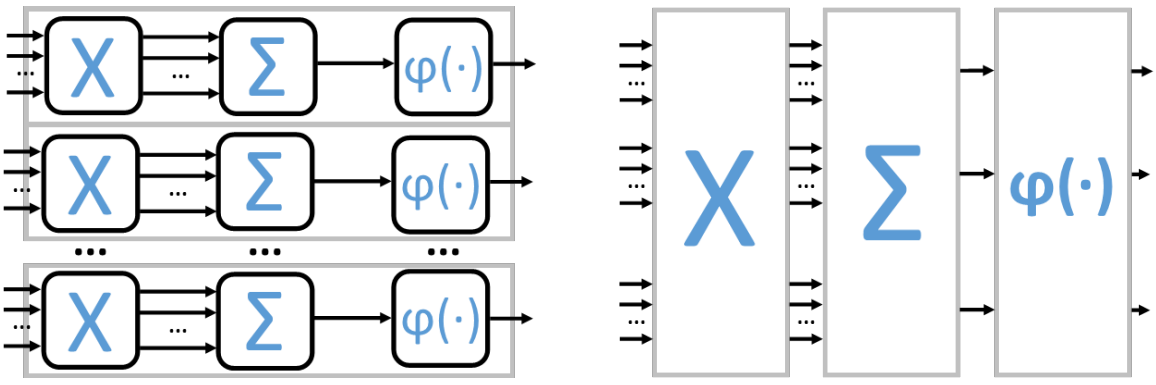
[2] https://gitlab.com/rihards.novickis/compage
[3] https://gitlab.com/rihards.novickis/icom

4. ADAPTATION AND IMPLEMENTATION OF IMAGE PROCESSING ALGORITHMS

4.1 An Approach of Feed-Forward Neural Network Throughput-Optimized Implementation in FPGA

A novel approach for FFNN implementation has been proposed that revises the implementation challenge viewing it in terms of elementary structures and utilizing pipelining paradigm. The FFNN is split into elementary layers, where each layer can be characterized by its resource, e.g. adder, multiplier, activation function. These layers reserve different numbers of resources to achieve even distribution of latencies and attain an optimally pipelined implementation, where every layer's latency is less or equal to the time necessary for the network to accept new input. An accompanying tool, developed for the solution, converts the given network's topology into C++ code that further feeds into an HLS tool. The generated code already incorporates necessary directives to ensure the envisioned solution with the requested pipelining iteration interval.

Although neuron is an intuitive abstraction, it subdivides architecture as shown in Fig. 4.1(a). This approach omits resource sharing between parallel neurons. Therefore a different abstraction is proposed, which is shown in Fig. 4.1(b). In this approach, neural network structure is separated into elementary layers, where each layer is characterized by a specific resource – adder, multiplier or activation function.



(a) Non-optimized delay model for a single layer.

(b) Optimized delay model with resource sharing for a single layer.

Fig. 4.1. Different delay models: (a) delays are analyzed in terms of a neurons in layers; (b) joint delay analysis for "primitive" multiplication, addition and activation layers.

By considering the main bottlenecks, e.g. limited bandwidth or resource availability, it is possible to optimize the network by allocating an appropriate amount of resources for every

layer. Another important consideration for the *Artificial Neural Network* (ANN) implementation is the choice of data type, as floating-point data types contribute high precision and range but are costly to implement and pipeline. Furthermore, recent studies [35], [36] show that fixed-point data types can be used with a relatively small precision loss, especially if the ANN training procedure is aware of the fixed-point data type.

A tool has been developed to automate FFNN implementation that takes the FFNN topology as an input and generates C++ code for the Xilinx HLS. One of the main parameters of the tool, steering the process of the code generation, is the maximum delay acceptable for pipeline's stages, which is given in clock cycles. The parameter ensures allocating "just enough" resources to comply with the given constraint.

The developed tool is open-source and made available on-line[4]. Generated FFNN *Intellectual Property* (IP) cores are tested using two hardware interfaces: *Memory-Mapped* (MM) interface for latency estimation and *Streaming* (ST) interface for latency and throughput estimation. MM interface implies active control from the side of MPU, but ST interface is set up by utilizing LogiCORE DMA IP cores [37] and *Advanced eXtensible Interface* (AXI) high performance interface [38].

Table 4.1. Small topology implementation resource utilization and benchmark result comparison table with [39], [40], [41], [42] (LUT-Look-up-Table, FF-Flip Flop, DSP-Digital Signal Processor core, BRAM-Block Random Access Memory).

Topology	Interface	LUTs	FFs	DSPs	BRAM	Target Initiation Interval	Theoretical Initiation Interval	Theoretical Latency	Latency		Throughput (Samples/s)	
									Original	Achieved	Original	Achieved
[39] 2-2-1	Memory-mapped	246 (0.46%)	118 (0.11%)	6 (2.73%)	3 (2.14%)	1	2	6	34 ns	555 ns $\sigma = 3.4$ ns	29,412,000	1,803,200
	Streaming	205 (0.39%)	135 (0.13%)	6 (2.73%)	3 (2.14%)	1	2	9		2.68 µs $\sigma = 37.5$ ns		49,272,000
[40] 2-4-1	Memory-mapped	980 (1.84%)	800 (0.75%)	48 (21.82%)	9 (6.43%)	1	2	10	44 ns	586 ns $\sigma = 3.4$ ns	2,272,700	1,705,600
	Streaming	983 (1.85%)	946 (2.92%)	48 (21.82%)	9 (6.43%)	1	2	12		2.69 µs $\sigma = 44$ ns		49,231,000
[41] 4-8-3-3	Memory-mapped	1304 (2.45%)	912 (0.86%)	0 (0.0%)	3.5 (2.5%)	1	4	12	1.16 µs	620 ns $\sigma = 8.7$ ns	862,070	1,612,400
	Streaming	1356 (2.55%)	1028 (0.97%)	0 (0.0%)	3.5 (2.5%)	1	4	19		2.78 µs $\sigma = 91$ ns		24,796,000
[42] 1-5-1	Memory-mapped	257 (0.5%)	78 (0.1%)	0 (0.0%)	1.5 (1.1%)	1	3	5	683 ns	578 ns $\sigma = 9.5$ ns	1,463,100	1,730,100
	Streaming	257 (0.5%)	81 (0.1%)	0 (0.0%)	1.5 (1.1%)	1	3	7		2.68 µs $\sigma = 35$ ns		33,005,000

[4] http://git.edi.lv/rihards.novickis/generation_tool_hls_c_fully_connected_feed_forward_neural_network

Table 4.2. Implementation resource use and benchmark result comparison table with [43] (Theoretical II = 100, Theoretical Latency = 987). (LUT-Look-up-Table, FF-Flip Flop, DSP-Digital Signal Processor core, BRAM-Block Random Access Memory)

Implementation	LUTs	FFs	DSPs	BRAM	Latency	Throughput (Samples/s)
[43]A	2232 (4.2%)	1210 (1.1%)	2 (0.9%)	-	33.1 ms	30.2
[43]B	3306 (6.2%)	1326 (1.3%)	4 (1.8%)	-	24.7 ms	40.5
[43]C	41,297 (77.6%)	33,395 (31.4%)	33 (15.0%)	-	5.7 ms	175.4
[43]D	51,028 (95.9%)	35,655 (33.5%)	65 (29.5%)	-	3.5 ms	285.7
Impl. Memory-Mapped	30,197 (56.8%)	55,231 (51.9%)	122 (55.5%)	6 (4.3%)	10.4 µs $\sigma = 0.011$	96,435
Impl. Streaming	31,246 (58.7%)	56,067 (52.7%)	122 (55.5%)	6 (4.3%)	12.5 µs $\sigma = 0.027$	997,852

All tests were performed on the Xilinx Zynq ZC702 SoC evaluation board using a bare metal stack with FPGA logic clocked at 100 MHz frequency. Timing measurements are made by using the Cortex-A9 *Snoop Control Unit* (SCU) timer. The performed tests and detailed comparison with other solutions are provided in Tables 4.1 and 4.2.

Results in Table 4.1 illustrate that ST-based implementation throughput approaches the theoretical initiation interval and can be characterized with about 2.7 μs latency.

Although the topology presented in [43] implements the hyperbolic tangent function using Xilinx LOGICore IPs, the solution outperforms any of the versions presented in the paper, marking solution's suitability for a floating-point implementation. The reason for such an impressive performance difference is the distinct design goals. In [43], the author prioritizes on-the-fly reconfiguration of the network, while presented solution targets maximum throughput of the network.

The developed solution also has been applied for virtual sensor use-case, and some configurations achieved an impressive performance of two mega-samples per second, which even overcomes the results in the original article [3].

4.2 Heterogeneous System Architecture for Stereo Image Processing

A HSoC-based point-correspondence computational system has been developed – a challenge involving a range of design abstractions, i.e. digital circuit design, on-chip communica-

tions, Linux driver and system development and system architecture. The partitioned functional architecture of the designed solution is shown in Fig. 4.2.

The processor (software) side of the HSoC ensures the overall control of the system and establishes communication with the off-board hardware (stereo camera through PCIe and demonstrator system through Ethernet) by utilizing the available software stack.

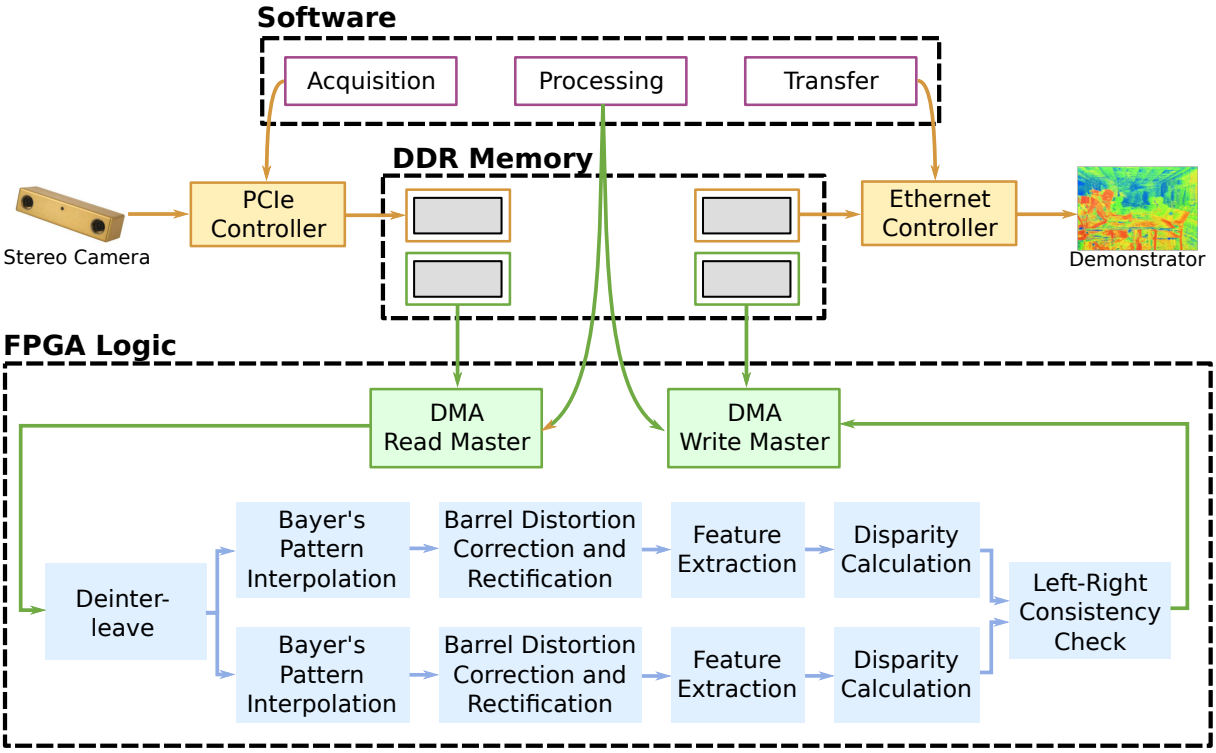


Fig. 4.2. Functional architecture of the developed HSoC stereo-vision solution partitioned across processing paradigms and components.

The designed inter-communication mechanism utilizes shared coherent memory, as it enables other bus masters apart from the processor to perform memory transfer operations; nevertheless, there is a challenge of software component synchronization. The challenge has been solved by utilizing a double buffering technique [34].

The developed system utilizes the double buffering technique two times – in between *Acquisition-Processing* and *Processing-Transfer* components. Two memory regions are shared and handled to ensure parallel component execution; for example, while the *Acquisition* thread writes input image into one of the buffers, the *Processing* thread uses the other buffer for configuring DMA transactions and transferring data through the accelerator. The same mechanism provides communication between the *Processing* and *Transfer* components; therefore, SoC simultaneously performs image acquisition, image processing and output image transfer resulting

in a high-level pipeline.

4.3 Stereo Image Processing Pipeline

4.3.1 Deinterleaving of the input stream

The first procedure implemented in the stereo-vision pipeline is the deinterleaving algorithm that splits the input image stream from the Bumblebee camera into multiple output streams. Figure 4.3 illustrates such input stream. Note that the used variant of the camera system incorporates three cameras.

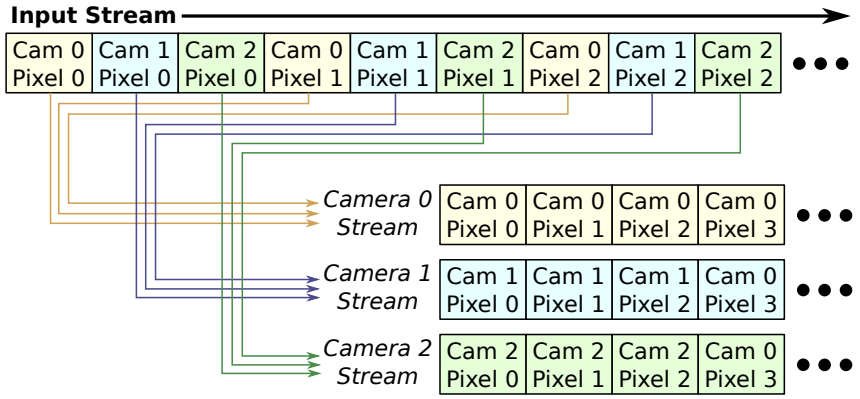


Fig. 4.3. Interleaved input stream of the bumblebee camera.

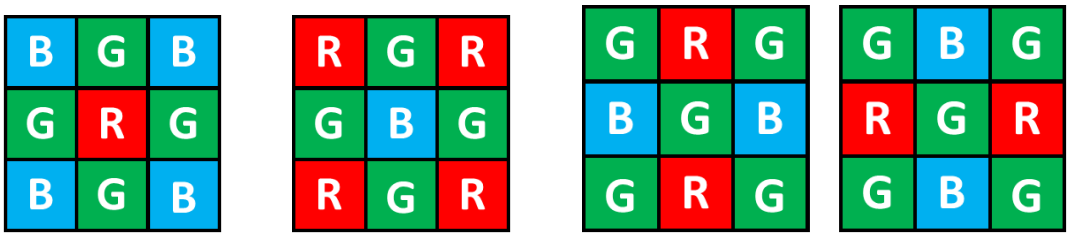
The challenge is solved in the digital logic by using an of-the-shelf stream adapter and converting input stream width to 24 bits, i.e. resulting in 3 parallel pixels each expressed using a single byte. Notably, directly interfacing cameras would require additional *First-In-First-Out* (FIFO) buffers, as image sensors stream data assuming an always-ready data sink.

4.3.2 Bayers pattern interpolation and RGB-to-Grayscale conversion

Modern cameras employ *Color Filter Array* (CFA), where different-colour filters reside on alternating sensor pixels. The most commonly used pattern in modern cameras is the *Bayer pattern* [44]. The process of *interpolating* the missing colour values to acquire valid RGB values for all pixels is known as *demosaicing*.

Although many demosaicing methods have been developed [45], the designed pipeline utilizes simple reconstruction based on linear and bilinear interpolation algorithms. Figure 4.4 illustrates different patterns to be considered by the demosaicing algorithm.

On closer examination, it becomes evident that at any particular clock cycle, two colour



(a) Red value is known, green and blue values are inferred from the adjacent pixels.

(b) Blue value is known, red and green values are inferred from the adjacent pixels.

(c) Green value is known, red and blue values are inferred from the adjacent pixels.

Fig. 4.4. Bayer pattern variants considered for demosaicing algorithm.

values have to be inferred simultaneously. Further, the applied interpolation algorithm examines a region of 3×3 by using a sliding window approach, and at any particular clock cycle, either 4 (two and two) or 8 (four and four) values contribute to the reconstruction process. Fig. 4.5 represents the high-level structure of the designed circuit.

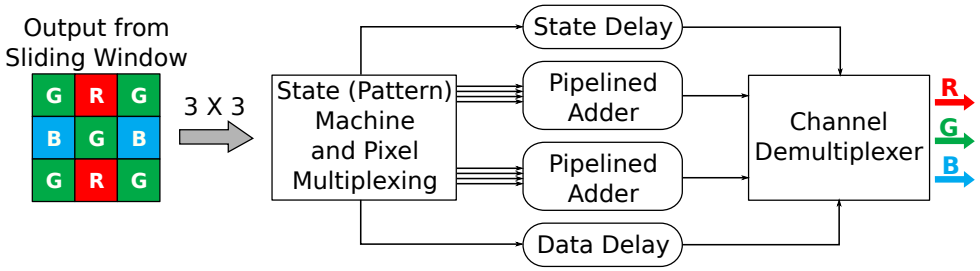


Fig. 4.5. High-level structure of the designed demosaicing circuit.

The input to the demosaic logic is provided by the sliding window block, which can be conveniently implemented in the digital logic [46]. The fully-pipelined demosaic block produces sets of data corresponding to Fig. 4.4. Further, the proposed structure consists of a state generator, where each state corresponds to the patterns in Fig. 4.4. The block arranges these inputs for the pipelined adders, while the centre pixel is simply delayed because its value requires no reconstruction. The generated state is also delayed and fed into the demultiplexing block for output rearrangement.

The demosaic block also includes optional *RGB-to-Grayscale* conversion functionality, as grayscale images reduce the number of operations triple-fold while not having major precision drawbacks, e.g. edge detection rate may be reduced by less than $10\,\%$ [47], [48]. Therefore, a hardware-friendly *lightness* colourspace conversion method has been selected [49].

4.3.3 An approach to spatial image transformation

An essential part of any image pre-processing is the algorithms for pixel transformation or mapping, i.e. lens distortion correction, image rectification, digital zoom. Executing such tasks in digital logic is associated with higher complexity due to the semi-global nature of the memory access patterns. Notably, the storage of the entire image in the OCRAM memory of the programmable logic is either impossible (FPGA chips often have on-chip memory less than 1 MB) or expensive (a single dual-port *Static Random Access Memory* (SRAM) cell in the on-chip memory requires eight *Complementary Metal-Oxide Semiconductor* (CMOS) transistors, which results in a large area).

Figure 4.6 illustrates the functional architecture of the spatial transformation accelerator that consists of input and output coordinate counters, dual-port memory matrix, write and read masters, output buffering logic, control logic and external inverse transformation calculation logic.

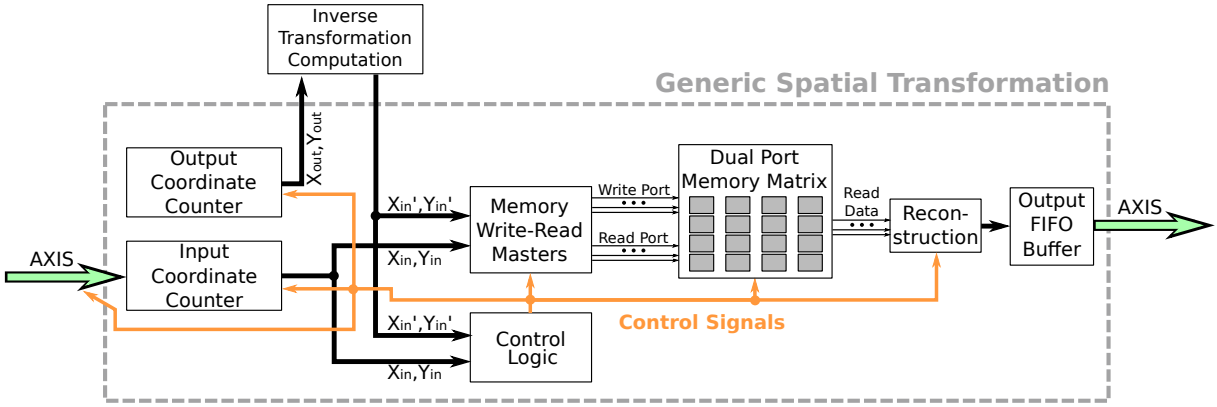


Fig. 4.6. Approach to fully pipelined image transformation in digital logic.

Any spatial image transformation can be expressed as a mapping of input pixels to output:

$$x_{out}, y_{out} = f(x_{in}, y_{in}), \tag{4.1}$$

where x_{in}, y_{in} and x_{out}, y_{out} denote the input/output image coordinates, and f is some arbitrary function, often expressed as a matrix in the case of linear transformations. In such cases, incrementing input coordinates may result in ”hopping” for output coordinates. The proposed solution necessitates the opposite: computing the inverse transformation and essentially retrieving the next input coordinate pair for the consecutive output coordinates, i.e.:

$$x_{in}, y_{in} = f^{-1}(x_{out}, y_{out}). \tag{4.2}$$

The solution adopts a unique technique for signal reconstruction that enables N-point reconstruction while preserving memory resources. This technique is also generalized to any number of dimensions. The tedious task of generating addresses for the individual memories in the *Dual Port Memory Matrix* is performed by the *Memory Write-Read Masters*.

One of the principal requirements for carrying out a fully pipelined spatial transformation with interpolation is a memory buffering scheme that would permit interpolation logic to have simultaneous access to adjacent pixels. The conventional approaches can be optimised considering that pixel data for interpolation is adjacent. Figure 4.7 illustrates a more efficient solution, where memories are reduced in size and only contain data for the respective (odd / pair) columns and rows.

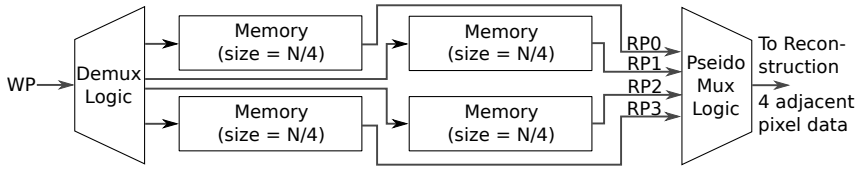


Fig. 4.7. Optimized data access scheme for 4-point reconstruction with $4\times$ reduction in memory size.

Considering that the memory matrix consists of M rows and N columns, the last $log_2(M)$ row signal bits and $log_2(N)$ column signal bits control the demultiplexing logic for the write pointer's write request signal as illustrated in Fig. 4.8. Naturally, all write ports of the memory matrix share the write data and address signals.

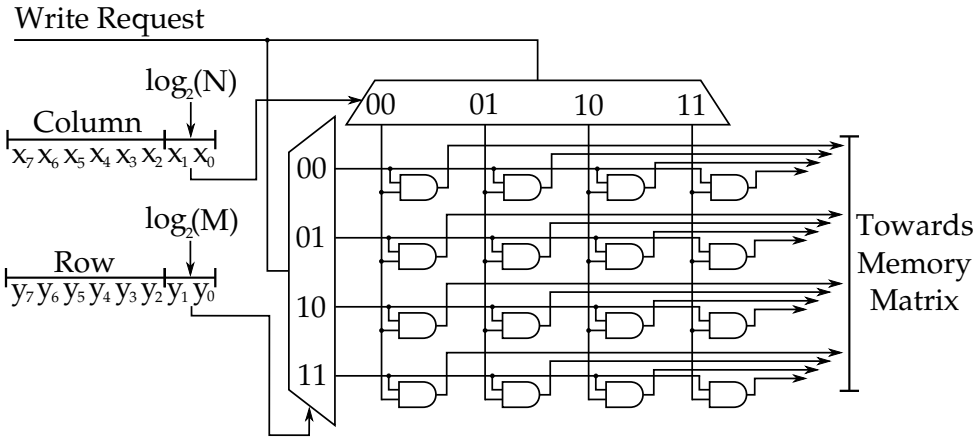


Fig. 4.8. An example of write request demultiplexing logic when using 4×4 memory matrix.

Data retrieval is more challenging because there is a need of generating varied addresses for each of the memories' read ports. The Thesis formalized this process and extended it to any

number of dimensions. Figure 4.9 illustrates how the combination of vertical and horizontal retrieval mechanisms generate required addresses for all memories.

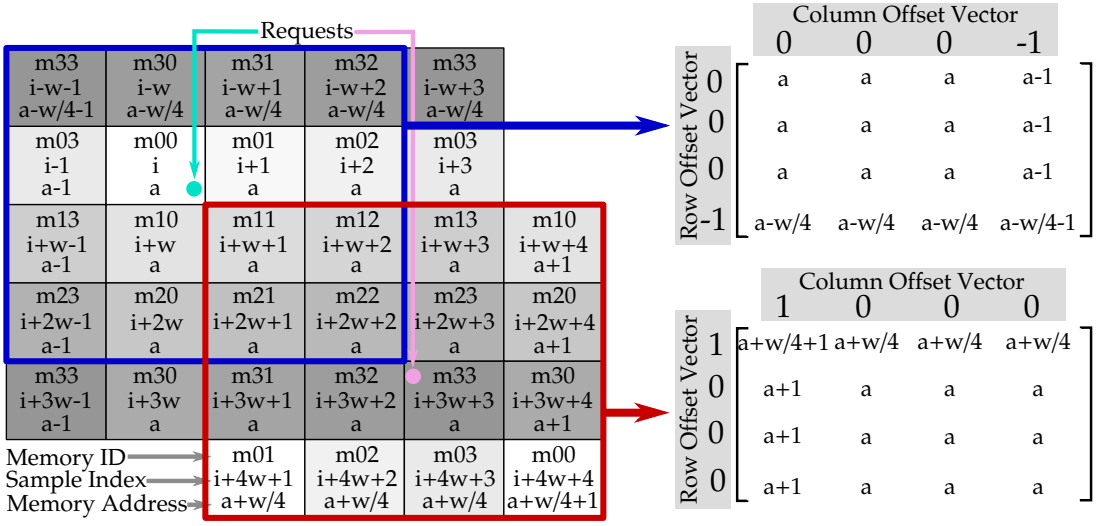


Fig. 4.9. Address generation for 4x4 memory matrix.

Finally, the approach extends to any number of dimensions. For N dimensions memory addresses would be expressed as an N-dimensional matrix $A \in E^N$, where each element is computed by:

$$A_{i_1 i_2 \cdots i_N} = \sum_{n=1}^{N} C_n S_n v_{o_{i_n}}, \tag{4.3}$$

where S_n denotes shift matrix, v_o is offset vector, and C_n is offset's constant determined by the data resolution of the particular dimension.

Further, the developed mathematical model can be mapped into particular digital circuits. Figure 4.10 illustrates the overall data retrieval concept and the organizational structure of the necessary components.

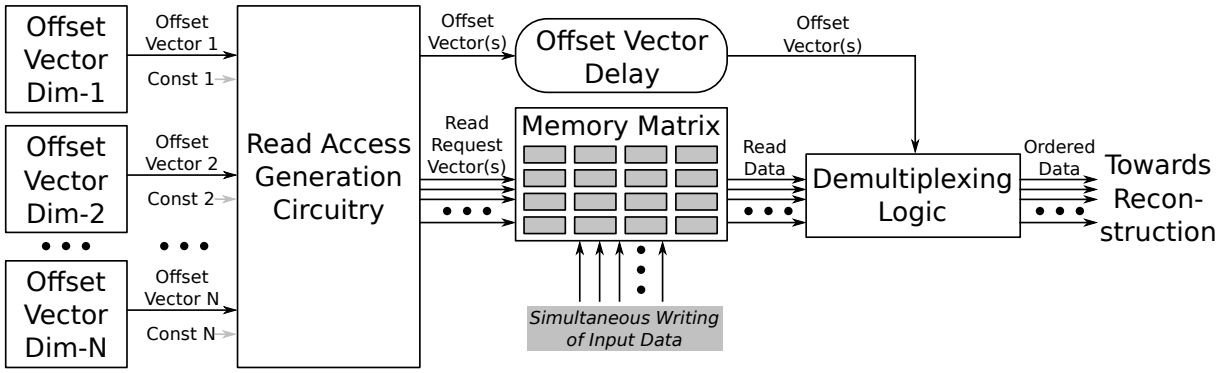


Fig. 4.10. General address vector computing concept for N-dimensions.

One of the most complex parts of the model is the read address generation that has to be implemented separately for each number of dimensions; nonetheless, patterns exist assisting such designs. An example of *Read Access Generation* logic for two dimensions is shown in Fig. 4.11. The concept presents an approach consisting of three parts: computing all possible combinations for offset constants, computing all possible combinations for a memory address, multiplexing possible memory addresses to the actual memories.

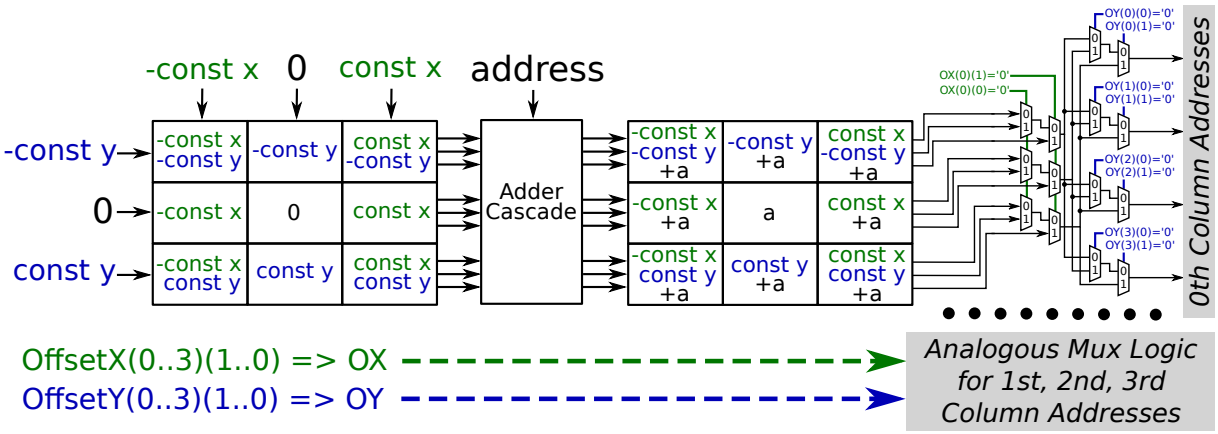


Fig. 4.11. Conceptual design of the read access circuitry.

Finally, all possible memory addresses are available, and they are routed accordingly for the memories.

4.3.4 Feature extraction

Further, it is necessary to extract features used for the correspondence matching. Figure 4.12 illustrates the feature extractor as implemented by the demonstrator system. The feature extractor provides four types of features: 1) pixel's intensity; 2) values of horizontal Sobel filter for adjacent left right pixels; 3) vertical Sobel filter at the pixel; and 4) a census transform using 5x5 pixel region. For experimentation, the RTL has been developed using generics allowing for different feature configurations. Each feature also incorporates variable delay blocks, whose generation depends on the latency of the respective extraction block. The extraction itself is done in a fully pipelined manner using the sliding window approach. Finally, a vector from all the features is formed that further feeds into the correspondence matching components.

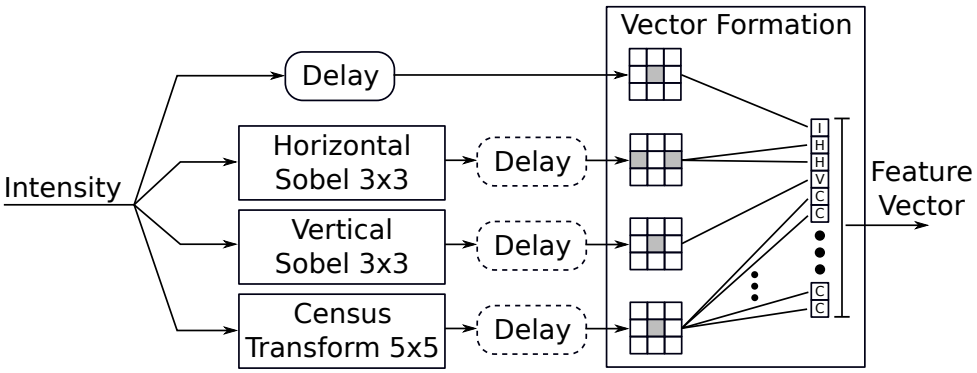


Fig. 4.12. High-level composition of the implemented feature extractor.

4.3.5 Correspondence calculations

Correspondence matching is the most resource consuming part of the whole stereo-image processing pipeline, because a fully pipelined implementation requires that all correspondence descriptor matching is accomplished in parallel. Figure 4.13 illustrates the overall concept for simultaneous left-to-right and right-to-left correspondence matching using 128 points. The extracted feature descriptors are buffered using *Serial-In-Parallel-Out* (SIPO) buffers and linked through *Feature Comparator* blocks. The comparison results are further fed into *Correspondence Search Circuits* that identify a correspondence with the least amount of ”energy” (opposite of confidence).

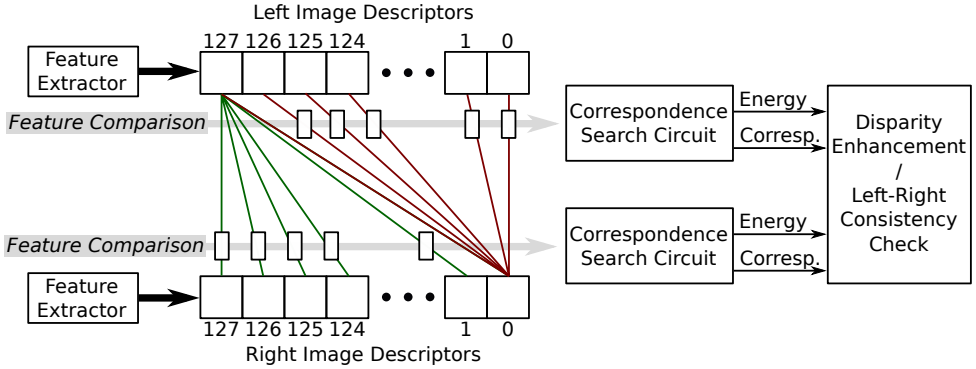


Fig. 4.13. Composition of correspondence calculation and matching logic.

4.4 Demonstrator system

The demonstrator system has been developed according to the previously derived system architecture. Figure 4.14 illustrates deployed demonstrator that is composed of Bumblebee BBX3 stereo-vision camera, Terasic’s VEEK-MT Cyclone V heterogeneous SoC development kit and

OpenGL-based host demonstrator. The developed technology targets high computational performance while having low-power consumption (< 10W) and overall costs. The SoC software application ensures image acquisition via PCIe, control of the FPGA accelerator pipeline and dispatching processed images to *Program Counter* (PC) -based demonstrator via Ethernet. All disparity-related computations are carried out on FPGA logic (schematic described in VHDL), including an interpolation of Bayer filter mosaic, correction for barrel distortions, rectification, feature extraction and disparity calculation.

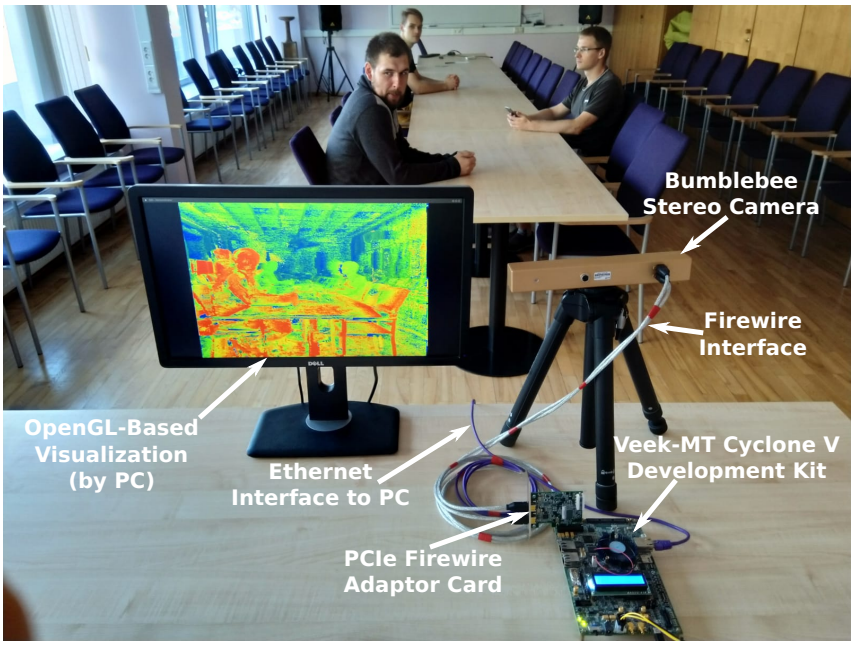


Fig. 4.14. A demonstration of the stereo-vision demonstrator in action.

During the development of this Thesis, Intel has disclosed an ASIC-based solutions, which has rapidly concured applications in field of robotics. Nonetheless, considering the sparsity of the computed disparity map, the developed technology already can be used for depth sensing applications in power-critical systems such as mobile drones (obstacle avoidance) and some *Internet of Things* (IoT) use-cases.

5. CONCLUSIONS

The Thesis addresses the relationship between computerized perception and increasingly complex HSoC technologies, in particular, the on-chip hardware and software co-architectures, implementation of stereo-vision and AI algorithms and associated real-time considerations. The primary aim of the Thesis is to develop and improve computer vision development techniques and methods for HSoC technology. In order to achieve the set aim, five tasks were defined.

1. Identify methods for complementing RTL and software-based computing paradigms. This task was accomplished in Section 1 and Subsection 3.1. The literature review acknowledged the *State of Art* (SoA) in different computing paradigms that were further used to analyze their tradeoffs. Further, this knowledge facilitated the development of architecture based on control by the MPU and offloading computing tasks to the FPGA. Both computing paradigms utilize memory-centric communication mechanisms for maximizing the overall throughput of the system.

2. Design heterogeneous architectures and tools for utilizing heterogeneous SoC technology. This task was accomplished in Section 3 by developing an FPGA-master based architecture for SoCs running Linux with the accompanying drivers and libraries for coherent contiguous memory management and DMA engine control. Further, an Asymmetric Multiprocessing (AMP) subsystem has been developed that enables real-time processing in Linux-based systems by offloading the critical applications to a fully dedicated bare-metal processing core. The developed solution unites the broad availability of open-source software with real-time control using a Linux driver interface. The driver ensures the setup of the bare-metal application and its configuration, AMP core's control, and it provides means for aggregating real-time performance characterization using *sysfs* interface. Finally, software-based system architecture implementation tools – *compage* and *icom* – have been developed. The tools enable the implementation of *blackboard* programming patterns for constrained Linux systems by giving the user means of configuring, replicating and interconnecting different software components, i.e. threads.

3. Design heterogeneous approach to the implementation of image processing pipelines. This task was accomplished in Subsections 3.1 and 4.2 by establishing a heterogeneous architecture for image processing pipelines using stereo-vision use-case as an example. The developed architecture utilizes software for image acquisition into coherent memory from a Bumblebee camera through PCIe, oversees the processing pipeline implemented in the FPGA and handoffs the produced results to the demonstrator system via Ethernet interface. In the developed system, essentially, all components – software and hardware – execute simultaneously, thus achieving software + hardware parallelism.

4. Implement and conduct experimental research on the developed tools and algorithms. This task was accomplished in Section 4. First, an ANN-based solution was implemented in the

FPGA hardware probing the possibility of a fully pipelined design. The developed approach is accompanied by a software tool for converting FFNN topologies into SIP cores with a streaming or memory-mapped interface. The achieved results not only outperform other solutions described in the literature but also show the applicability of the developed method for virtual sensor implementation, i.e. using the electric vehicle torque vectoring use-case as an example.

Further, a range of fully-pipelined accelerators have been designed for the implementation in digital logic, including deinterleaving of the combined input image streams, demosaicing of Bayer's RGB pattern, spatially transforming images to perform lens distortion correction and rectification, extracting features and computing the correspondence matching challenge. One of the main contributions is the generalization of the circuit for enabling fully-pipelined access to adjacent data samples for reconstruction in a memory-conserving way. While the developed solution is utilized for image processing, i.e. two dimensions, the generalized model enables the construction of a circuit for any number of dimensions, therefore enabling reconstruction, for example, of volumetric data. Finally, the developed accelerators were utilized for the development of the stereo-vision demonstrator.

5. Draw conclusions about the results of the Thesis. The main conclusion regarding the stereo-vision algorithm implementation using heterogeneous SoCs is that the unique blend of software and hardware ensures the feasibility of implementing such image processing pipelines. Furthermore, the technology achieves that in an energy-efficient way and provides extendability. The developed approach to image processing in heterogeneous SoC technology can (and is) applicable to other applications, e.g. processing of large (>50MP) images.

The value of the achieved results is further highlighted by several ongoing and finalized international research projects. For example, the software architecture implementation frameworks are being used for the development of AI-based perception system for vehicles (PRYSTINE, G.A. 783190, AI4CSM, G.A. 101007326) and control software for the control of autonomous drones (COMP4DRONES, G.A. 826610), while the developed image processing pipeline is reused in the design of infrared image preprocessing algorithms (APPLAUSE, G.A. 826588).

Further, the outcomes of the Thesis serve as a basis for an ongoing commercialization activity (SilHouse, No. KC-PI-2020/12), where a framework is being developed that joins a range of accelerators and provides a convenient software-based framework for their application to the industry.

BIBLIOGRAPHY

[1] Moore, G. E. Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff. *IEEE Solid-State Circuits Society Newsletter*. - 2006. - Vol.11 - Nr.3. - 33–35 p. 2006.

[2] Novickis, R. and Greitāns, M. FPGA Master based on chip communications architecture for Cyclone V SoC running Linux// *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*. ISSN: 2576-3555 - Apr. 2018. - 403–408 p. DOI: 10.1109/CoDIT.2018.8394842.

[3] Dendaluce Jahnke, M., Cosco, F., Novickis, R., Pérez Rastelli, J., and Gomez-Garay, V. Efficient Neural Network Implementations on Parallel Embedded Platforms Applied to Real-Time Torque-Vectoring Optimization Using Predictions for Multi-Motor Electric Vehicles en *Electronics*. - Feb. 2019. - Vol.8 - Nr.2. - 250 p. Feb. 2019. ISSN: 2079-9292. DOI: 10.3390/electronics8020250. / URL - http://www.mdpi.com/2079-9292/8/2/250 (visited on 07/25/2020).

[4] Setka, V., Jezek, O., and Novickis, R. Modular Signal Processing Unit for Motion Control Applications Based on System-on-Chip with FPGA en// *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. - Zaragoza, Spain: IEEE, Sep. 2019. - 857–863 p. ISBN: 978-1-72810-303-7. DOI: 10.1109/ETFA.2019.8869121. / URL - https://ieeexplore.ieee.org/document/8869121/ (visited on 07/25/2020).

[5] Novickis, R., Justs, D. J., Ozols, K., and Greitāns, M. An approach of feed-forward neural network throughput-optimized implementation in fpga *Electronics*. - 2020. - Vol.9 - Nr.12. - 2193 p. 2020. DOI: 10.3390/electronics9122193. / URL - https://www.mdpi.com/2079-9292/9/12/2193.

[6] Novickis, R., Levinskis, A., Kadikis, R., Fescenko, V., and Ozols, K. Functional architecture for autonomous driving and its implementation *2020 17th Biennial Baltic Electronics Conference (BEC)*. - 2020. 2020. DOI: 10.1109/bec49624.2020.9276943. / URL - https://ieeexplore.ieee.org/abstract/document/9276943.

[7] Druml, N., Debaillie, B., Anghel, A., Ristea, N.-C., Fuchs, J., Dubey, A., Reisland, T., Hartstem, M., Rack, V., Ryabokon, A., and al., et Programmable systems for intelligence in automobiles (prystine).2222em technical progress after year 2 *2020 23rd Euromicro Conference on Digital System Design (DSD)*. - 2020. 2020. DOI: 10.1109/dsd51259.2020.00065. / URL - https://ieeexplore.ieee.org/document/9217654.

[8] Druml, N., Ryabokon, A., Schorn, R., Koszescha, J., Ozols, K., Levinskis, A., Novickis, R., Nigussie, E., Isoaho, J., Solmaz, S., Stettinger, G., Diaz, S., Marcano, M., Villagra, J., Medina, J., Schwarz, M., Artuñedo, A., Comi, M., Beekelaar, R., Özçelik, O., Taşdelen, E. A., Gürbüz, Y., Saijets, J., Kyynäräinen, J., Morits, D., Debaillie, B., Rykunov, M., Escamilla, J., Vanne, J., Korhonen, T., Holma, K., Matzhold, E.-M., Novara, C., Tango, F., Burgio, P., Calafiore, G., Karimshoushtari, M., Boulay, E., Dhaens, M., Praet, K., Zwijnenberg, H., Palm, H., Ortega, D. A., Kalali, E., Pensala, T., Kyytinen, A., Larsen, M., Veledar, O., Macher, G., Lafer, M., Giraudi, L., Reckenzaun, J., Hammer, D., Mohan, N., Schmid, J., Höß, A., Ophir, S., Dubey, A., Fuchs, J., Lübke, M., Anghel, A., Ristea, N.-C., Törngren, M., Musralina, A., Harter, M., Jose, J. M., and Dimitrakopoulos, G.

Programmable systems for intelligence in automobiles (prystine): Final results after year 3// *2021 24th Euromicro Conference on Digital System Design (DSD)*. - 2021. - 268–277 p. DOI: 10.1109/DSD53832.2021.00049.

[9] Kuon, I., Tessier, R., and Rose, J. FPGA Architecture: Survey and Challenges en *Foundations and Trends® in Electronic Design Automation*. - 2007. - Vol.2 - Nr.2. - 135–253 p. 2007. ISSN: 1551-3939, 1551-3947. DOI: 10.1561/1000000005. / URL - http://www.nowpublishers.com/article/Details/EDA-005 (visited on 08/26/2017).

[10] Chu, P. P. RTL hardware design using VHDL. -. John Wiley & Sons, 2006. ISBN: 9780471720928. DOI: 10.1002/0471786411.

[11] Insight Technologies "State of Linux in the public cloud for enterprises" Red Hat Solution overview 2018. / URL - https://www.redhat.com/cms/managed-files/cl-state-of-linux-in-public-cloud-for-enterprises-f11154kc-201802-en_0.pdf (visited on 08/10/2020).

[12] Corbet, J., Rubini, A., and Kroah-Hartman, G. Linux Device Drivers. Third -. O'Reilly, Dec. 2010.

[13] Szeliski, R. Multiple view geometry in computer vision. en -. 2004. OCLC: 171123855 ISBN: 978-0-511-18711-7 978-0-511-18618-9 978-0-511-18895-4 978-0-511-18535-9 978-0-511-18451-2 978-0-511-81168-5 978-1-280-45812-5. / URL - http://dx.doi.org/10.1017/CBO9780511811685 (visited on 03/12/2019).

[14] ——, Computer Vision: Algorithms and Applications en 2011. - 979 p. 2011. ISSN: 1868-0941.

[15] Marr, D. Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. en - Nr.2. -. W. H. Freeman and Company, 1982. ISBN: 0-7167-1567.

[16] Scharstein, D., Szeliski, R., and Zabih, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms// *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*. - Dec. 2001. - 131–140 p. DOI: 10.1109/SMBV.2001.988771.

[17] Haykin, S. Neural Networks: A Comprehensive Foundation. 2nd -. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998. ISBN: 978-0-13-273350-2.

[18] Fine, T. L. Feedforward Neural Network Methodology. 1st -. Cornell University, Ithaca, NY, USA: Springer-Verlag New York, 1999. ISBN: 978-0-387-98745-3.

[19] Chakradhar, S., Sankaradas, M., Jakkula, V., and Cadambi, S. A dynamically configurable coprocessor for convolutional neural networks// *ACM SIGARCH Computer Architecture News*. - Vol.38 - ACM, 2010. - 247–257 p. / URL - http://dl.acm.org/citation.cfm?id=1815993 (visited on 09/06/2017).

[20] Suda, N., Chandra, V., Dasika, G., Mohanty, A., Ma, Y., Vrudhula, S., Seo, J.-s., and Cao, Y. Throughput-Optimized OpenCL-based FPGA Accelerator for Large-Scale Convolutional Neural Networks en// *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate*. - ACM Press, 2016. - 16–25 p. ISBN: 978-1-4503-3856-1. DOI: 10.1145/2847263.2847276. / URL - http://dl.acm.org/citation.cfm?doid=2847263.2847276 (visited on 09/06/2017).

[21] Zhang, C., Li, P., Sun, G., Guan, Y., Xiao, B., and Cong, J. Optimizing fpga-based accelerator design for deep convolutional neural networks// *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. - ACM, 2015. - 161–170 p. / URL - http://dl.acm.org/citation.cfm?id=2689060 (visited on 09/06/2017).

[22] Foumani, S. N. A. An fpga accelerated method for training feed-forward neural networks using alternating direction method of multipliers and lsmr. Master's thesis Imperial College London, Department of Computing 2020.

[23] Blott, M., Preußer, T. B., Fraser, N. J., Gambardella, G., O'brien, K., Umuroglu, Y., Leeser, M. E., and A., V. K. FINN-R: An End-to-End Deep-Learning Framework for Fast Exploration of Quantized Neural Networks *ACM Transactions on Reconfigurable Technology and Systems*. - 2018. 2018. DOI: 10.1145/3242897.

[24] Guan, Y., Liang, H., Xu, N., Wang, W., Shi, S., Chen, X., Sun, G., Zhang, W., and Cong, J. FP-DNN: An Automated Framework for Mapping Deep Neural Networks onto FPGAs with RTL-HLS Hybrid Templates// *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. ISSN: 978-1-5386-4038-8 - 2017. DOI: 10.1109/FCCM.2017.25.

[25] Khan, A., Sohail, A., Zahoora, U., and Qureshi, A. S. A survey of the recent architectures of deep convolutional neural networks en *Artif Intell Rev*. - 2020. - Vol.53 - Nr.8. - 5455–5516 p. 2020. ISSN: 1573-7462. DOI: 10.1007/s10462-020-09825-6. / URL - https://doi.org/10.1007/s10462-020-09825-6 (visited on 10/31/2020).

[26] Shalf, J. and Leland, R. Computing beyond Moore's Law *Computer*. - 2015. - Vol.48 - 14–23 p. 2015. DOI: 10.1109/MC.2015.374.

[27] Sadri, M., Weis, C., Wehn, N., and Benini, L. Energy and Performance Exploration of Accelerator Coherency Port Using Xilinx ZYNQ// *FPGAworld '13 Proceedings of the 10th FPGAworld Conference*. - 2013.

[28] Molanes, R. F., Salgado, F., Fariña, J., and Rodríguez-Andina, J. J. Characterization of FPGA-master ARM communication delays in Cyclone V devices// *41st Annual Conference of the IEEE Industrial Electronics Society*. - 2015. - 4229–4234 p.

[29] Vogel, P., Marongiu, A., and Benini, L. An Evaluation of Memory Sharing Performance for Heterogeneous Embedded SoCs with Many-Core Accelerators// *COSMIC '15 Proceedings of the 2015 International Workshop on Code Optimisation for Multi and Many Cores*. - 2015.

[30] Altera corp. *Cyclone V Hard Processor System Technical Reference Manual* - Oct. 2016.

[31] Šetka, V., Ježek, O., and Novickis, R. Modular signal processing unit for motion control applications based on system-on-chip with fpga// *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. - 2019. - 857–863 p. DOI: 10.1109/ETFA.2019.8869121.

[32] Sangmin Chon "What it takes to do efficient and cost-effective real-time control with a single microcontroller the c2000™advantage" Texas Instruments White paper 2011. / URL - https://www.ti.com/lit/wp/spry157/spry157.pdf?ts=1637148859226\&ref_url=https\%253A\%252F\%252Fwww.google.com\%252F (visited on 11/17/2021).

[33] F. Buschmann, K. Henney, and D. C. Schmidt Pattern-Oriented Software Architecture Volume 4: A Pattern Language for Distributed Computing. -. Chichester: Wiley, 2007. - Vol.Volume 4.

[34] NVIDIA Corporation NVIDIA GPU Programming Guide en 2005. - 80 p. 2005.

[35] Fu, Y., Wu, E., Sirasao, A., Attia, S., Khan, K., and Wittig, R. Deep Learning with INT8 Optimization on Xilinx Devices en 2017. - 11 p. 2017.

[36] Dettmers, T. 8-Bit Approximations for Parallelism in Deep Learning en *arXiv:1511.04561 [cs]*. - Nov. 2015. Nov. 2015. arXiv: 1511.04561. / URL - http://arxiv.org/abs/1511.04561 (visited on 10/23/2018).

[37] AXI DMA v7.1, LogiCORE IP Product Guide Apr. 2018. - 95 p. Apr. 2018. / URL - https://www.xilinx.com/support/documentation/ip_documentation/axi_dma/v7_1/pg021_axi_dma.pdf.

[38] Zynq-7000 All Programmable SoC Technical Reference Manual (UG585) en 2016. - 1863 p. 2016.

[39] Hariprasath, S. and Prabakar, T. N. FPGA implementation of multilayer feed forward neural network architecture using VHDL// *Computing, Communication and Applications (ICCCA), 2012 International Conference on*. - IEEE, 2012. - 1–6 p. / URL - http://ieeexplore.ieee.org/abstract/document/6179225/ (visited on 09/06/2017).

[40] Youssef, A., Mohammed, K., and Nasar, A. A Reconfigurable, Generic and Programmable Feed Forward Neural Network Implementation in FPGA// *2012 UKSim 14th International Conference on Computer Modelling and Simulation*. - IEEE, 2012. - 9–13 p. ISBN: 978-1-4673-1366-7 978-0-7695-4682-7. DOI: 10.1109/UKSim.2012.12. / URL - http://ieeexplore.ieee.org/document/6205543/ (visited on 09/06/2017).

[41] Yuan Jing, Youssefi, B., Mirhassani, M., and Muscedere, R. An efficient FPGA implementation of Optical Character Recognition for License Plate Recognition en// *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*. - IEEE, 2017. - 1–4 p. ISBN: 978-1-5090-5538-8. DOI: 10.1109/CCECE.2017.7946734. / URL - http://ieeexplore.ieee.org/document/7946734/ (visited on 04/17/2018).

[42] Oliveira, J. G. M., Moreno, R. L., Oliveira Dutra, O. de, and Pimenta, T. C. Implementation of a reconfigurable neural network in FPGA en// *2017 International Caribbean Conference on Devices, Circuits and Systems (ICCDCS)*. - Cozumel, Mexico: IEEE, 2017. - 41–44 p. ISBN: 978-1-5386-1962-9. DOI: 10.1109/ICCDCS.2017.7959699. / URL - http://ieeexplore.ieee.org/document/7959699/ (visited on 09/08/2018).

[43] Hajduk, Z. Reconfigurable FPGA implementation of neural networks en *Neurocomputing*. - 2018. - Vol.308 - 227–234 p. 2018. ISSN: 09252312. DOI: 10.1016/j.neucom.2018.04.077. / URL - https://linkinghub.elsevier.com/retrieve/pii/S0925231218305393 (visited on 09/08/2018).

[44] Bayer, B. E. Color imaging array. U.S. Patent - Nr.US3971065A. 1976.

[45] Li, X., Gunturk, B., and Zhang, L. Image demosaicing: A systematic survey en Pearlman, W. A., Woods, J. W., and Lu, L., Eds. - San Jose, CA, Jan. 2008. - 68221J p. DOI: 10.1117/12.766768. / URL - http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.766768 (visited on 08/09/2020).

[46] Yu, H. and Leeser, M. Automatic sliding window operation optimization for fpga-based// *2006 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*. - 2006. - 76–88 p. DOI: 10.1109/FCCM.2006.29.

[47] Hagara, M., Stojanović, R., Bagala, T., Kubinec, P., and Ondráček, O. Grayscale image formats for edge detection and for its fpga implementation *Microprocessors and Microsystems*. - 2020. - Vol.75 - 103056 p. 2020. ISSN: 0141-9331. DOI: https://doi.org/10.1016/j.micpro.2020.103056. / URL - https://www.sciencedirect.com/science/article/pii/S0141933119305034.

[48] Huntsberger, T. and Descalzi, M. Color edge detection *Pattern Recognition Letters*. - 1985. - Vol.3 - Nr.3. - 205–209 p. 1985. ISSN: 0167-8655. DOI: https://doi.org/10.1016/0167-8655(85)90054-6. / URL - https://www.sciencedirect.com/science/article/pii/0167865585900546.

[49] Cook, J. D. (- 2009.). Three algorithms for converting to grayscale / URL - https://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/.