Computing unite of a mobile computer vision system

I V Zoev, N G Markov and T A Yagunov
Department of Information Technologies, Tomsk Polytechnic University, Tomsk, Russia
E-mail: ivz3@tpu.ru

Abstract. The work is devoted to the computing unite of a mobile computer vision system and developing his algorithmic software. We developed hardware-implemented the convolutional neural networks on a field programmable gate array. A study of the performance and power consumption of variants of the computing unite.

1. Introduction
Nowadays, the scientific direction moving to using artificial neural networks for development of mobile intelligent computer vision systems (CVS). These CVSs should automatically analyze the images with the help of neural networks trained by modern deep-learning methods. The most promising mobile CVSs are developed using convolutional neural networks (CNN) [1].

The paper proposes various ways for organization of computations in a computing unite (CU) of a mobile CVS, which consists a hardware-implemented CNN on a modern system on a chip (SoC). The results of a study are presented for performance and power consumption of three variants of this CU for various ways of organizing computations.

2. Organization of computations in CU based on SoC

2.1. General statement on the organization of computations
A modern intellectual CVSs are systems which allow you to automate the process of analyzing visual information using artificial neural networks trained using deep learning methods. An important class of these CVSs are mobile CVSs have been installing on autonomous vehicles, for example, on unmanned aerial vehicles, on unmanned vehicles, etc. These CVSs should have low power consumption, for example, in the case of unmanned aerial vehicles not more than 10 watts. Today, mobile CVSs are developed using hardware-implemented CNN on modern SoCs, which have field programmable gate array (FPGA) for high-performance CNN computations. The following arguments shown advantages of high performance of the hardware implementation of the CNN on the FPGA:

• comparison with other implementations of the CNN, for example, on graphics processors, gives that the CU on SoC on modern FPGAs consumes significantly less power;

• flexibility, it is mean possibility to implement on the FPGA, different and even simultaneously operating several CNN architectures.

Therefore, having these advantages of the hardware implementation of the CNN on the FPGA, it is possible to development mobile and high-performance CVS based of modern SoC, which is intellectual in nature because of using CNN.

In [2], it was proposed to use the computational resources of not only the FPGA, but also other components of modern SoCs for the hardware-based CNN. The architecture of most SoCs allows you to organize direct access of a hardware-implemented CNN to external memory. The implementation of this method of interaction of the FPGA with external memory will allows you to perform some operations that differ from the convolution and subsample procedures of the CNN on the SoC processor.

Also in [2] an original method for performing computations in a hardware CNN on a FPGA was proposed. It differs from the known methods by using unified computational convolution and subsampling units. The unification of the convolution / subsample blocks is achieved by extracting a parameters of this blocks, usually specified at the stage of their synthesis, and placing them in a separate variable FPGA memory area, called the configuration memory region. Thereby it allows to use the blocks of the suitable type in the layers of the CNN with different architectural parameters. The implementation of the method assumes that the number of computing blocks involved in the hardware CNN can be variable. And the number of computing blocks is determined only by the FPGA resources. Scaling by the number of using blocks carried out both as a whole for the CNN and for individual layers. It will significantly reduce the required computational resources of the FPGA and will help implement various CNN architectures without FPGA reconfiguring.

When implementing the unification method, in addition to allocate the configuration memory area, it is necessary to organize access of the SoC processor to this memory area and to develop the appropriate software, primarily in the form of a multifunctional driver of hardware CNN.

2.2. Ways of organizing computations in the CU and their implementation

The unification of computing blocks of the CNN hardware and the way proposed in [2] (below – Way 1) of organizing computations in SoC, taking into account the interaction of the FPGA with external memory, made it possible to implement a mobile CU of CVS based on the Terasic SoCKit developing board with the SoC Cyclone V SX [3]. The enlarged functional diagram of this CU is shown in Figure 1.

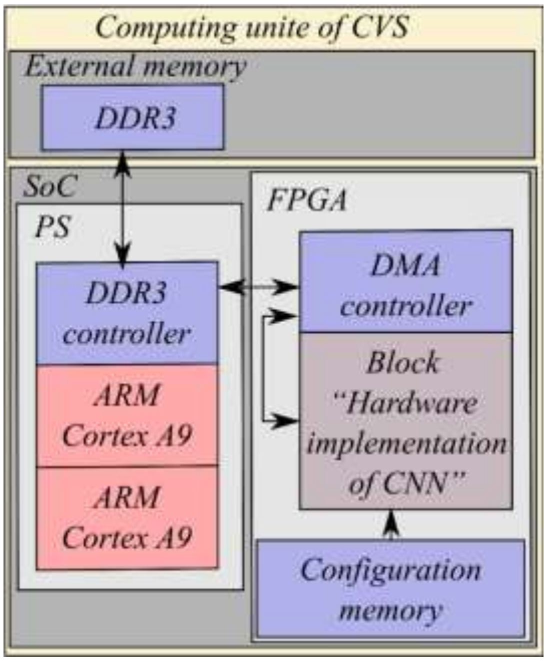

Figure 1. The enlarged functional diagram of the CU for the mobile CVS.

The diagram shows two main functional units. The SoC is the first one. It contains a processor system (PS) and FPGA. The PS in turn includes a DDR3 external memory controller and two ARM Cortex A9 processor cores. On FPGA we developed a controller of direct access to external memory (DMA controller), a block "Hardware implementation of CNN" and configuration memory. The block "Hardware implementation of CNN" consists of a neural computing unit (NCU). The NCU performs the computations of the convolution and subsampling procedures for different CNN architectures of the LeNet5 subclass. The second functional unit is external DDR3 memory.

An executing the convolution procedure for each convolutional layer of the CNN was required two types of data independent of each other. The first is the input and output data (the analyzed image and feature maps) of the CNN convolution layer. The second type is the weights of the convolution layer. This statement allows us to propose a second way (below – Way 2) of organizing computations in a CU, which includes using two channels of external memory in SoC and two different external memory modules. One of these channels will be reserved only for the transfer of the first type of data, and the second – for the transfer of data of another type. The Figure 2 shows the enlarged functional diagram of the CU with that way of implementation.

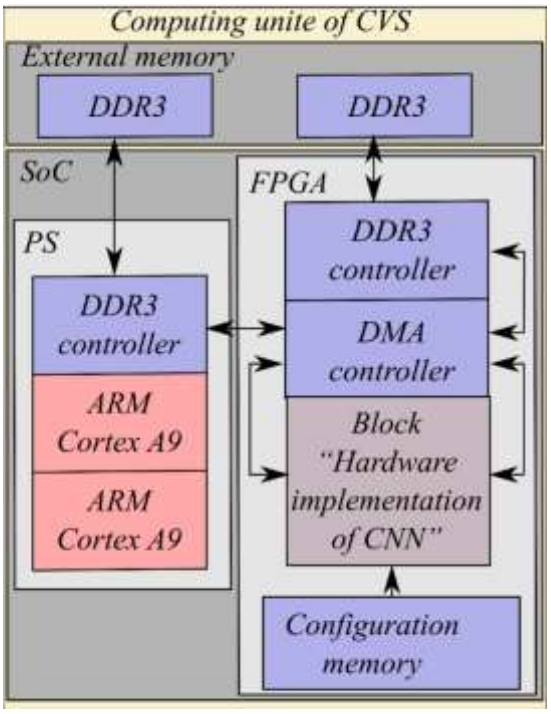

Figure 2. The enlarged functional diagram of the CU for implement Way 2 computations of organization**.**

The difference between the new functional diagram of the CU (new version of the CU) and the diagram of the CU in Figure 1 (the first version of the CU) is that the controller of direct access to external memory (DMA controller) has access in two physically separated memory modules. It provides data transfer between these two external memory modules and the hardware implementation of the CNN. Considering this, it make possible the simultaneous loading input data and weights into the internal buffers of the NCU module.

The best option for this variant of CU is the placement of the CNN weights in the external memory module connected directly to the FPGA through its own DMA controller. Because when perform calculating the CNN, the weights do not change, and the necessity of their unloading from the PS memory appears only during the initialization of the CU. The input and output data of the CNN layers should be stored in the PS external memory module. Because it gives the possibility of their operational pre- and post-processing.

It is possible to propose a third way (below – Way 3) of organizing computations in a CU, which includes using several modules of the NCU in the "Hardware implementation of the CNN" block. In fact, this method is an extension of the capabilities of Way 1, which have only one NCU module for use. Figure 3 shows the enlarged functional diagram of the CU (the third version of the CU) with the implementation of Way 3 of the organization of computations. This diagram shows the usage of n NCU modules.

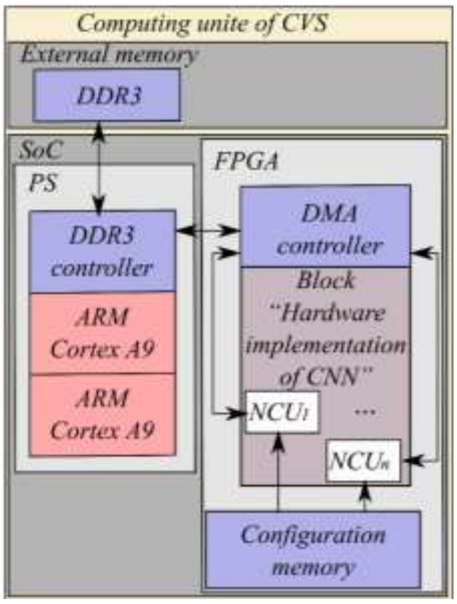

Figure 3. The enlarged functional diagram of the CU for implement Way 3 organization of computations**.**

The number *n* of NCU modules can be variable and their number depends only on the amount of computational resources of SoC. Also, the each NCU is a completely independent and self-sufficient module. The internal architecture of individual NCU in the "Hardware implementation of CNN" block also may be differ.

3. Algorithmic software of CU

We implemented the hardware components (diagrams in Figures 1–3) and also developed the corresponding algorithmic software executing on PS. The algorithmic software have difficult original algorithm of locating and marking external memory (memory mapping) for hardware-implemented CNN on FPGA with various architectures. The point of this algorithm is to reduce the amount of external memory required for CNN computations by providing direct access for NCU to external memory areas that are common to different layers of the CNN. Also we have developed the algorithms for the reconfiguration of the NCU in the FPGA.

A multifunctional software driver for the Linux kernel OS was developed for ensure interaction between the FPGA, PS and external memory. This driver was written by using C language. The main functions of the driver is following:

- functions for interaction with FPGA registers;
- functions that work with data buffers in external memory modules;
- functions that mapping the external memory;
- reconfiguration of the NCU in the FPGA;
- interrupt handling, which come from FPGA.

This driver easily adapts to the peculiarities of different the CU versions and the CNN architectures.

In C++ language, we have developed a library for the interaction of user with the CU settings. Library functions allow him to interact with the CU through the character devices registered by multifunctional driver. Each operation in this driver is initiated by a user's command transferred to the character device through a specific library function.

4. The task of studying the effectiveness of CU

In each of the three variants of CU, was implemented one of the methods of computations organization discussed above. The original CNN architecture of the LeNet5 subclass from [4] was used as a hardware-implemented CNN in CU. This CNN architecture has the following parameters. The first layer is a convolutional, the number of input feature maps 3, output - 6, the kernel size of convolution is 7×7, stride is 1. The second layer is subsample, the number of input feature maps is 6, the output – 6, kernel size of the subsample is 2×2, stride is 2. The third layer is convolutional, the number of input maps is 6, output – 32, kernel size of convolution is 5×5, stride is 1. The fourth layer is subsample, the number of input feature maps is 32, output 32, kernel size of subsample is 2×2, stride is 2. The fifth layer is convolutional, the number of input feature maps is 32, output - 100, kernel size of convolution is 5x5, stride is 1. The sixth (fully connected) layer is the convolution, the number of input feature maps 100, output - 10, kernel size of convolution is 1×1, stride is 1.

Initially, training was carried out on the software-implemented CNN of this architecture on a sample of images of hand-written numbers MNIST [5]. It contains 60,000 training images and 10,000 test pairs (tag-image). The CNN training was carried out using the Caffe library [6], used the method of back propagation of error and the method of stochastic gradient descent. Optimization parameters were taken from the library's training examples. The accuracy of the classification of handwritten numbers for the software-implemented CNN of this architecture, using the 32-bit floating-point format, is 98.7%. The weight coefficients, obtained during training, then were transferred to each of the CU variants.

In all experiments, was solved the problem of classifying each image from 10,000 MNIST test images. The classification of images was carried out using each of the three variants of the CU. For classifying images, we used configurations K1, K2, K3 and K4 of the NCU module presented in table 1. The operating frequency of the NCU in each of the configurations is 50 MHz.

Table 1. Configurations of NCU module.

Configuration Name of NCU	**K1**	**K2**	**K3**	**K4**
Number of computational convolution blocks	6	10	32	100
Number of computational subsample blocks	6	10	32	32

The performance of each CU version was estimated as the difference between the time getting the result from the SoC as a handwritten digit class and the launch time of this CU for classification. The moments of time were determined using the standard chrono library of the C++ programming language. The average time for classifying one image was calculated as the average of 10,000 times the classification of test images.

There are we also researched the energy consumption of each variants of CU and their efficiency.

5. The results of the research and their discussion

Table 2 shows the results of studying a performance for two variants of the CU that use Way 1 and Way 2 for organization of computations.

Table 2. CU performance with Way 1 and Way 2 computing organization.

Configuration Name of NCU	**Runtime Way 1, ms**	**Runtime Way 2, ms**
K1	16.464	15.487
K2	13.985	11.27
K3	6.933	6.579
K4	7.202	5.986

The results from table 2 show that in the case of the variant CU implementing Way 2, for all configurations of NCU, there is a reduction in the average time for classifying one image with comparing to the variant CU implementing Way 1. At the same time, optimal performance according to the CU performance criteria, related to the amount of computing resources, has a CU with NCU K3 configuration (32 convolution blocks and 32 subsample blocks).

The following series of experiments was carried out for a variant of CU implemented Way 3 computing organization, which containing two or three NCU modules with a K3 configuration. The maximum possible number of NCU modules is three. It is the limit amount of FPGA resources in the Cyclone V SX SoC. Table 3 shows the results of these experiments. For comparison, table 3 includes the performance of CU variant that implements Way 1. This way, like Way 3, takes into account the interaction of the FPGA with only one external memory module. In experiments when computations of computations in CU Way 3 for each analyzed image was performed synchronous launch of NCU modules.

Table 3. CU performance with Way 1 and Way 3 computing organization.

Performance variants CU	**Way 1**	**Way 3 (2 modules NCU)**	**Way 3 (3 modules NCU)**
Number of simultaneously classified images, pcs	1	2	3
Average image classification time on one module of the NCU, ms	6.933	7.811	12.763
Average classification time per image, ms	6.933	3.905	4.161

The best performance has the version of the CU, which implements Way 3 of the organization of computational processes with two NCU modules. So the preference should be given this variant of CU.

The measuring of power consumption in this experimental shows that the CU variants in organization of computations for Way 1 and for Way 3 is 5.1 watts. The power consumption of the variant of the CU implementing the Way 2 of the organization of computations increases to 7.4 watts. The reason of such increase covered in connection of another DDR3 external memory module.

All results obtained in the experiments indicate that taking into account modern requirements for CVSs in terms of performance and power consumption, in principle any of the three CU variants can be used as part of a mobile and high-performance CVS with low power consumption. However, the developer must choose the variant of the CU in accordance with the scale of the real-time work for the CVS.

6. Conclusion

Today, the current research area is focused on the development of mobile and high-performance CVSs using modern SoCs. This paper is devoted to the development of a CU these CVSs. In this CU the hardware is implemented on the FPGA SoC Cyclone V SX CNN allows to solve the tasks of classifying objects in images.

Two new ways for organization of computations were proposed. And different CU variants were implemented based on proposed functional diagrams. Also algorithmic software for these CUs was developed.

The task of the researching of the performance was solved for all proposed variants of the CU with the hardware implementation of the original architecture of the CNN of the LeNet5 subclass. The results of carried out research show that the best performance has a CU version that implements Way 3 of the organization of computations, and preference should be given to CU with two hardware-implemented neural computing units. Also we found that the lowest power consumption have devices that implement Way 1 and Way 3 of the organization of computations.

Acknowledgment
The reported study was funded by Russian foundation for basic research, project no. 18-47-700010 p_a.

References

[1] Russakovsky O *et al* 2015 *International J of Computer Vision* **115** (3) 211–252
[2] Beresnev A P, Zoev I V and Markov N G 2018 *Proc. of 28th Int. Conf. on Computer Graphics and Vision* 184–187
[3] SoCKit – the Development Kit for New SoC Device Terasic. Available at: http://www.terasic.com.tw/cgi-bin/page/archive.pl?CategoryNo=167&No=816
[4] Zoev I V *et al* 2017 *Computer Optics* **41** (6) 938–949
[5] LeCun Y, Cortes C and Burges C 2018 The MNIST database – MNIST handwritten digit database. Available at: http://yann.lecun.com/exdb/mnist/
[6] Caffe - Deep Learning Framework 2019. Available at: https://caffe.berkeleyvision.org/