DOI: 10.35598/mcfpga.2021.010

# Neuron Networks Design in Matlab and Vivado

Oleksandr Vorgul
ORCID 0000-0002-7659-8796
*Department of Microprocessor Technologies and Systems*
*Kharkiv National University of Radio Electronics*
Kharkiv, Ukraine
oleksandr.vorgul@nure.ua

Iryna Svyd
ORCID 0000-0002-4635-6542
*Department of Microprocessor Technologies and Systems*
*Kharkiv National University of Radio Electronics*
Kharkiv, Ukraine
iryna.svyd@nure.ua

Oleg Zubkov
ORCID 0000-0002-8528-6540
*Department of Microprocessor Technologies and Systems*
*Kharkiv National University of Radio Electronics*
Kharkiv, Ukraine
oleh.zubkov@nure.ua

***Abstract*—This article is devoted to design of a measurement system based on specialized FPGA. A balance of ACD and DAC channels through output from one side and computation power of FPGA from another side is considered. Possibilities for obtaining one more tool for screening the signal processing is proposed.**

***Keywords—FPGA, analog signal, digital signal processing, measurement system***

## I. Introduction

Computing tools, in particular for solving problems implemented in embedded execution, are currently being rapidly developed. In its development, for a highly effective solution of problems, the complexity increases and the quality is quickly reached its limit [1, 2]. In a sense, by changing the quantity it is no longer possible to obtain an adequate change in quality for reasonable money. Apparently, all such roads turn to problems of complexity theory.

Neural networks in embedded systems can be used as a universal decisive node. In this work, an attempt is made to review the current state of arts and chart a path for solution of a simple typical problem with means we can reach.

## II. Neuron Networks from a Bird's Eye

A neural network is a model of a multilayer system with redundant connections in layers, with the possibility of feedback (recurrent) or without the possibility (convolutional), fully described by a wide set of dynamically changing parameters (weights) [2-4].

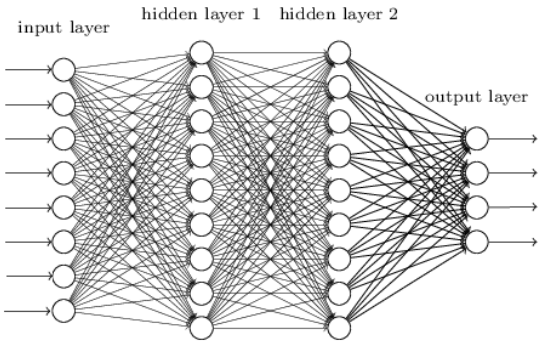



Fig. 1. Convolutional Neuron Network.

Neural network element – neuron can be described as a node with multiple inputs and a decision rule.

One can borrow some experience in implementing a neural network on analog elements [1-3], but at the same time, in addition to the complexity inherent in large systems (controllability, number of parameters, speed, cost at all stages of development), energy consumption also becomes important.

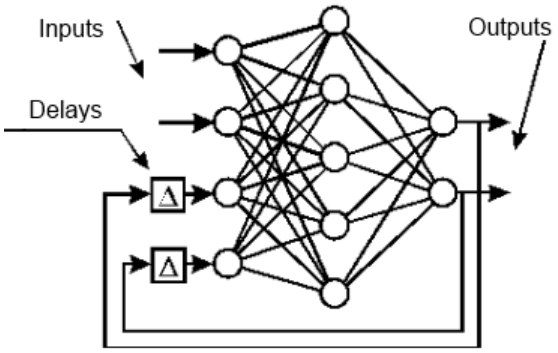



Fig. 2. Recurrent neuron Network.

Applying neural networks seems to have to go through several stages [1, 2]:

- development (on paper) and compilation of a mathematical model
- choice of element base and implementation (classic detailing and construction)
- training a neural network (a feature that is not available for alternative implementations)
- work and correction of the model (retraining)
- utilization of the neural network. Better to foresee it.

In this sense, the use of ready-made solutions can significantly reduce the time at all stages, if the ready-made solutions can be trusted to solve the required task.

## III. Matlab Offers

Matlab offers all possible assistance in the neural network setting, development and training. The first version of neural network toolbox in Matlab seems to have appeared in version 4.0, 1992, under Windows 3.1. The site today has a toolbox history that stretches via time from version 2021a

to version 2018a. And no other versions, unfortunately. Though that would be interesting. Since some version, the toolbox is called the Deep Learning Toolbox [5-9].

### A. *Workflow for Neural Network Design*

The work flow for the neural network design process has seven primary steps:

*1)* Collect data.

*2)* Create the network.

*3)* Configure the network.

*4)* Initialize the weights and biases.

*5)* Train the network.

*6)* Validate the network.

*7)* Use the network.

### B. *Neuron Model*

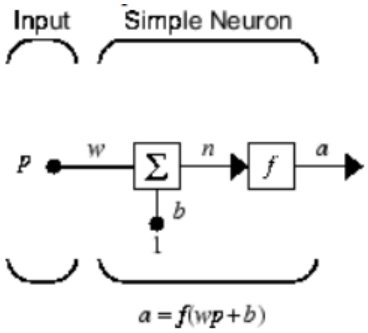



Fig. 3. Simple Neuron.

The simple neuron can be extended to handle inputs that are vectors. A neuron with a single R-element input vector is shown below. Here the individual input elements:

$$p_1, p_2, p_3, \cdots, p_R \tag{1}$$

are multiplied by weights

$$w_{,1,1}, w_{1,2}, w_{1,3}, \cdots, w_{1,R} \tag{2}$$

and the weighted values are fed to the summing junction. Their sum is simply Wp, the dot product of the (single row) matrix W and the vector p.

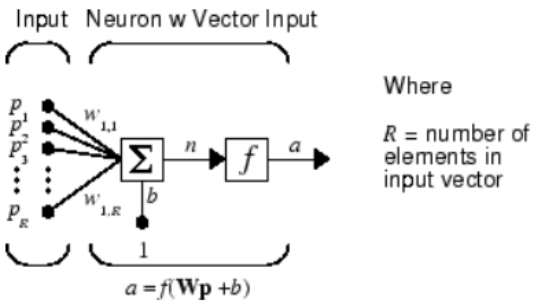



Fig. 4. Neuron with Vector Input.

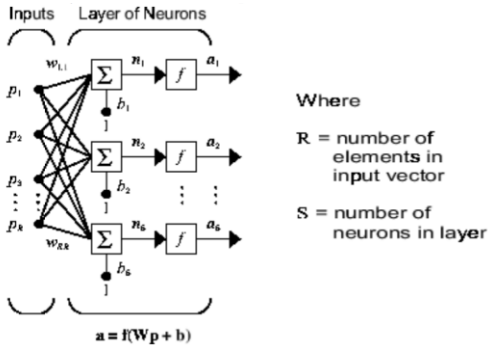
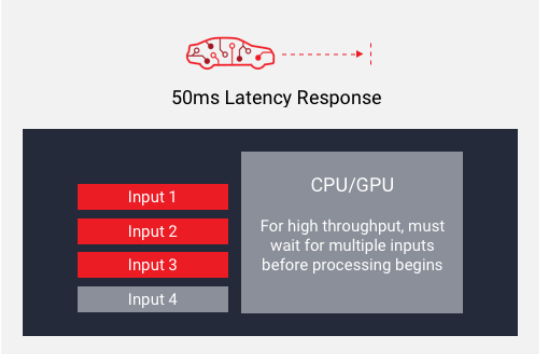



Fig. 5. Neural Network Architectures.

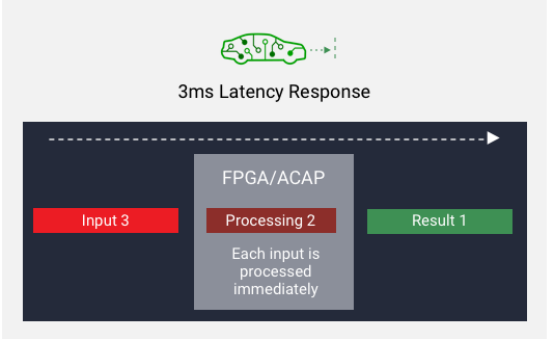
## IV. FPGA Implementation

Apparently, at present, the implementation of a neural network can be performed on anything on a CPU, GPU, FPGA. Probably, somewhere there are specialized neuroprocessors that are hybrid (analog-digital). Let's consider briefly only FPGA and only Xilinx case.

Front panel of Xilinx announces lowest latency AI inference [2, 4].




High Throughput OR Low Latency Achieves throughput using high-batch size. Must wait for all inputs to be ready before processing, resulting in high latency




High Throughput AND Low Latency Achieves throughput using low-batch size. Processes each input as soon as it's ready, resulting in low latency [10].

Then the Xilinx provides DPU for convolutional neural network

Product Description The Xilinx® Deep Learning Processor Unit (DPU) is a programmable engine dedicated for convolutional neural network. The unit contains register configure module, data controller module, and convolution

computing module. There is a specialized instruction set for DPU, which enables DPU to work efficiently for many convolutional neural networks. The deployed convolutional neural network in DPU includes VGG, ResNet, GoogLeNet, YOLO, SSD, MobileNet, FPN, etc.

The DPU IP can be integrated as a block in the programmable logic (PL) of the selected Zynq®-7000 SoC and Zynq UltraScale™+ MPSoC devices with direct connections to the processing system (PS). To use DPU, you should prepare the instructions and input image data in the specific memory address that DPU can access. The DPU operation also requires the application processing unit (APU) to service interrupts to coordinate data transfer.

The DPU presume to utilize Zynq 7000 and Zynq UltraScale+ products.

TABLE I. PROCESSING SYSTEM

| Features | ZU*CG | Z7000s | Z7000 |
|---|---|---|---|
| Application Processing Unit | Dual-core Arm Cortex-A53 MPCore up to 1.3GHz | Single-core ARM Cortex-A9 MPCore | Dual-core ARM Cortex-A9 MPCore |
| Real-Time Processing Unit | Dual-core Arm Cortex-R5F MPCore up to 533MHz | - | - |
| Maximum Frequency | n/a | Up to 766MHz | Up to 866MHz or 1 GHz |
| Dynamic memory Interface | DDR4, LPDDR4, DDR3, DDR3L, LPDDR3 | DDR3, DDR3L, DDR2, LPDDR2 | DDR3, DDR3L, DDR2, LPDDR2 |
| High-Speed periferials | PCIe® Gen2, USB3.0, SATA 3.1, DisplayPort, Gigabit Ethernet | USB 2.0, Gigabit Ethernet, SD/SDIO | |
| Dedicated Periferial Pins | n/a | Up to 128 | Up to 128 |

Programmed Logic:

*a)* One slave AXI interface for accessing configuration and status registers.

*b)* One master interface for accessing instructions.

*c)* Supports configurable AXI master interface with 64 or 128 bits for accessing data.

*d)* Supports individual configuration of each channel.

*e)* Supports optional interrupt request generation.

*f)* Some highlights of DPU functionality include:

- Configurable hardware architecture includes: B512, B800, B1024, B1152, B1600, B2304, B3136, and B4096
- Configurable core number up to three
- Convolution and deconvolution
- Max pooling
- ReLu and Leaky ReLu
- Concat
- Elementwise
- Dilation
- Reorg
- Fully connected layer
- Batch Normalization
- Split

## V. CONCLUSION

To see how good this idea - using a neural network to solve a typical complex problem - is, one needs to test it in practice. A fairly common task for a neural network is pattern recognition. But this will be the subject of further research.

## REFERENCES


[1] A. Galushkin Neural Networks: Foundations of Theory. - M .: Hot line-Telecom, 2012. - 496 p.

[2] V. Bezruk, I. Svyd, I. Korsun. Neural technologies in telecommunications and control systems: navch. posibnik with the stamp of the Ministry of Education and Science. - Kharkiv, SMIT, 2008 . – 230 p.

[3] Neural Networks and Deep Learning [Online]. Available: http://neuralnetworksanddeeplearning.com/chap5.html [Accessed: 24-May- 2021]

[4] O. Lozovich, A. Maksimov Application of FPGA-based neural networks for solving problems of reliability of communication information systems "Artificial Intelligence" 3'2011.

[5] Matlab Neural Network Toolbox [Online]. Available: https://se.mathworks.com/products/deep-learning.html [Accessed: 24-May- 2021]

[6] O. Vorgul, O. Zubkov, I. Svyd and V. Semenets, "Teaching microcontrollers and FPGAs in Quarantine from Coronavirus: Challenges and Prospects", *MC&FPGA-2020*, 2020. doi: 10.35598/mcfpga.2020.005.

[7] I. Svyd, O. Maltsev, O. Zubkov and L. Saikivska, "Matlab Use in Design of Digital Systems on the FPGA in CAD Xilinx VIVADO", *I International Scientific and Practical Conference*, 2019. doi: 10.35598/mcfpga.2019.010.

[8] I. Svyd, O. Vorgul, V. Semenets, O. Zubkov, V. Chumak and N. Boiko, "Special features of the educational component design of devices on microcontrollers and FPGA", *MC&FPGA-2020*, 2020. Available: 10.35598/mcfpga.2020.017.

[9] I. Svyd, O. Maltsev, L. Saikivska and O. Zubkov, "Review of Seventh Series FPGA Xilinx", *I International Scientific and Practical Conference*, 2019. Available: 10.35598/mcfpga.2019.008.

[10] Xilinx AI Inference Acceleration [Online]. Available: https://www.xilinx.com/applications/megatrends/machine-learning.html [Accessed: 24- May- 2021]