An Hardware Implementation of the Softmax Function on FPGA

Francesco Di Franco[a], Corrado Rametta[b], Michele Russo[a] and Mario Vaccaro[a]

[a]VICOSYSTEMS S.r.l V.le Odorico da Pordenone, 33, Catania, CT, Italy
[b]Department of Computer, Control, and Management Engineering, Sapienza University of Rome, Via Ariosto 25, Roma, Italy

Abstract

The softmax operation is used in the last layer of deep learning classifiers. This function is characterized by complex arithmetic operations as exponentials and divisions. In hardware implementations, such complexity negatively impacts on hardware resources. In this paper, we present an efficient hardware implementation of the softmax function. The proposed architecture has been designed and simulated in Simulink, coded in VHDL, and finally synthesized using the Xilinx Vivado toolchain. Implementation results are presented in terms of area and power.

Keywords

Softmax, CNN, Deep Learning, FPGA.

1. Introduction

In the last few years, we assisted to a significant increase in Machine Learning based applications and research works [1, 2, 3]. There are three main reasons for this incredible Machine Learning growing up:

1. The availability of a great amount of data thanks to internet diffusion.
2. The computation capability of modern digital systems that makes possible the efficient parallel computations as for example GPUs and FPGAs.
3. The introduction of new technologies that allow the implementation of artificial neural networks more and more similar to natural ones [4].

New technologies can range from sensors [5], new cellular systems [6], satellite [7] and critical services [8, 9, 10]. In all those machine learning systems characterized by high-speed computations requirements, hardware solutions are preferred to software solutions, such as in the case of FIR filters [11], it is very important to design hardware architectures optimized in terms of complexity in order to reduce costs and power consumption. In this scenario, the softmax function represents an important issue.

In Machine learning systems, the softmax [12, 13] function finds its application in the output layer of classifiers, making the output values, typically not normalized, interpretable as a percentage of success in recognizing a certain class.

The name of softmax is a function which, given in input a vector z, of real values, k-dimensional, produces in output a vector, always k-dimensional, containing the exponential values of the inputs normalized with respect to the summation of the latter.

The outputs have values included in the range [0.1] and their summation has a unitary value: it can be interpreted as a probabilistic distribution. The softmax equation is shown in eq. 1

$$\sigma : \mathbb{R}^K \longrightarrow \left\{ z \in \mathbb{R}^K | z_i > 0, \sum_{i=1}^{K} z_i = 1 \right\} \qquad (1)$$
$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \qquad j = 1, \ldots, K.$$

As shown in eq. 2 the Softmax function replaces the activation function of the output layer of the neural network.

$$p(y = j|\mathbf{x}) = \frac{e^{(\mathbf{w}_j^T \mathbf{x} + b_j)}}{\sum_{k \in K} e^{(\mathbf{w}_k^T \mathbf{x} + b_k)}} \qquad (2)$$

The block diagram of the softmax function is shown in Fig. 1 and works as follows:

- The system accepts a k-dimensional vector "y" as input;
- The vector is input to a logic block that produces k results, the result of the exponential function of the input values. This result is input to two logic blocks;

ICYRIME 2020: International Conference for Young Researchers in Informatics, Mathematics, and Engineering, Online, July 09 2020
f.difranco@vicosystems.it (F.D. Franco); m.russo@vicosystems.it (M. Russo); m.vaccaro@vicosystems.it (M. Vaccaro)

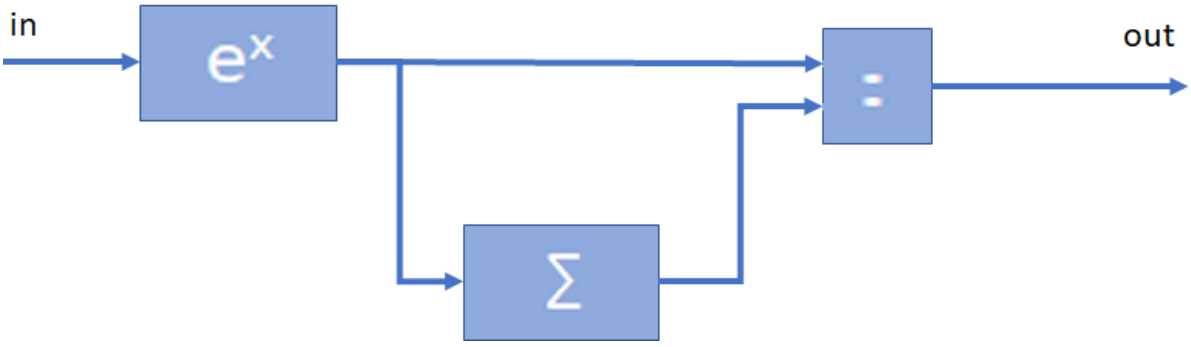

Figure 1: Softmax blockdiagram.

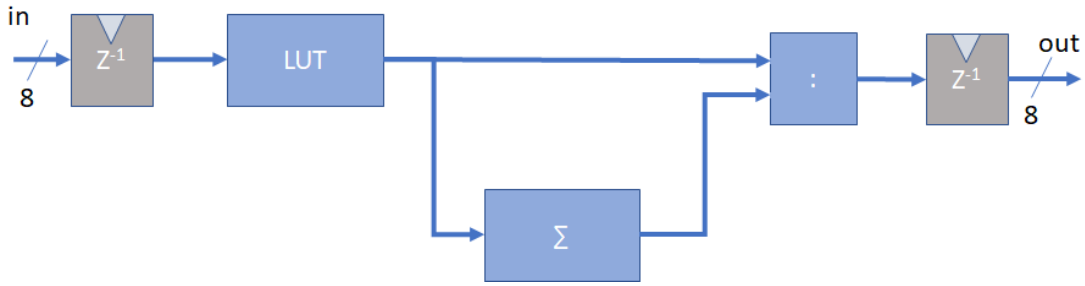

Figure 2: The proposed softmax hardware architecture

- The "summation" block compresses a k-dimensional vector into a numerical value equal to the sum of the k values;
- The last block divides each element produced by the exponential by the sum of the exponentials themselves, producing a k-dimensional vector containing the percentages.

2. Softmax hardware implementation

The proposed architecture for the hardware softmax implementation is shown in Fig. 2. It is composed of three main blocks:

- Look Up tables for the exponential computation of e^x
- An adders tree for the summation computation
- Dividers for the division computation

Because the exponential relationship between the number of the bit of the input and the location number of the Look-Up table used for the exponential function computation, it is necessary to introduce a technique able to reduce the location number of the Look-Up table in order to not negatively impact on the hardware resources used for the implementation. For this purpose we use the linear interpolation.

Let's consider the function $f(x) = e^x$ we define the function g as linear interpolation function in the interval $x_0 < x < x_0 + h$, with h defined as the interpolation step. The function g is defined in eq. 3.

$$\begin{aligned} g(x, x_0) &= f(x_0) + m \cdot (f(x_0 + h) - f(x_0)) \\ m &= \frac{x - x_0}{x_0 + h - x_0} = \frac{x - x_0}{h} \\ g(x, x_0) &= e^{x_0} + \frac{e^{x_0}}{h} \cdot (e^h - 1) \cdot (x - x_0) \end{aligned} \tag{3}$$

It is now possible to define the absolute error as shown in eq. 4 and the relative error shown in eq. 5

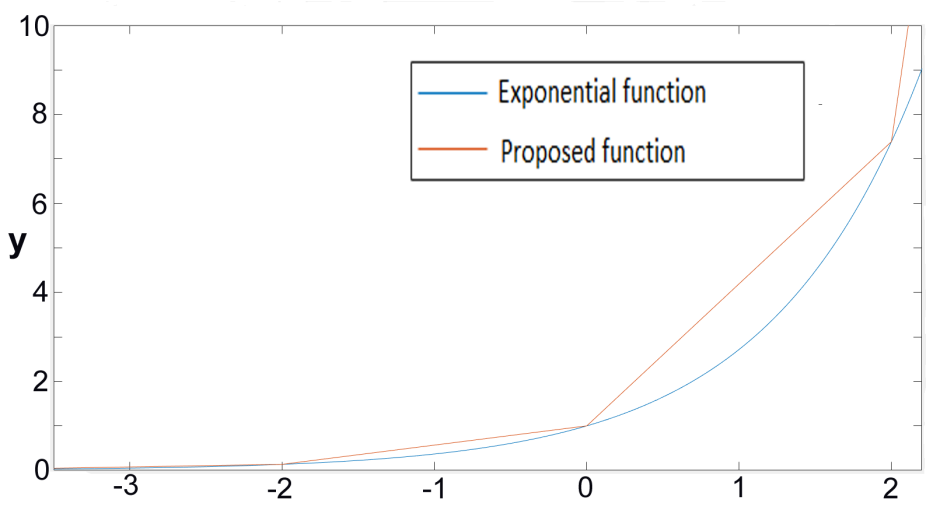

Figure 3: Exponential function vs proposed function

$$err(x, x_0) = g(x, x_0) - f(x) = \\ = e^{x_0} + \frac{e^{x_0}}{h} \cdot (e^h - 1) \cdot (x - x_0) - e^x \quad (4)$$

$$Err_{rel}(x, x_0) = \frac{err}{e^{x_0}} = 1 + \frac{e^h - 1}{h} \cdot (x - x_0) - \frac{e^x}{e^{x_0}} = \\ = 1 - e^{x-x_0} + \frac{e^h - 1}{h} \cdot (x - x_0) \quad (5)$$

Choosing h = 0.25 and = 0.125 it is possible to obtain maximum relative errors less than 0.8% with rms = 0.566% and less of 0.2% with rms = 0.14%.

In fig. 2 is shown a comparison between the exponential function and an interpolated exponential function.

In order to characterize the performance of the proposed system in terms of maximum frequency, hardware resources, and power consumption, we performed experiments on several implementations varying the range of the exponent and the size of the LUTs.

As regards the range, simulations were carried out with x in the range [-16; 15] and [-2; 1].

The device used for our experiments is the Zynq Ultrascale + ZCU 104, which allows you to manage the numerous inputs and outputs of 16-bit architectures. In our experiment we fix the value of the number of input to 8.

For x in the range [-15;15] we choose 5 bit for the integer part and 11 bit for the fractional part. The linear interpolation has been implemented with h=0.5. Using this approach the total amount of locations is equal to 64. The minimum relative error in terms of rms is 2.22%. The values stored in the Look-up table are in the range $[e^{-16} = 1.1254 * 10^{-7}; e^{15.5} = 5.3897 * 10{-}6]$.

For x in the range [-2;1] we choose 2 bit for the integer part and 7 for the fractional part. The linear interpolation has been implemented with h=0.125. Using this approach the total amount of locations is equal to 8. The minimum relative error in terms of rms is 0.25%. The values stored in the Look-up table are in the range $[e^{-2} = 0.1353; e^2 = 7.3891]$.

3. Experimental results

After a floating point and a fixed point Simulink simulation, the proposed softmax architectures were coded in VHDL and implemented using the XILINX Vivado toolchain. Finally, they have been characterized in terms of hardware resources speed and power consumption. For our experiments we use a Xilinx Zynq Ultrascale+ device.

Implementation results are shown in Fig. 3 and in Fig. 3 Implementations have been performed with a clock constraint of 20 ns. Experimental results show a very low use of hardware resources available in the device For what concerns power consumption, the results show a reduced dissipation. This is aspect is very important especially in all those applications where power is not provided directly by the power line but it is provided by batteries or energy harvesting systems. This is the case for example of IoT systems. Results show a power consumption of 0.120w and 0.107w for the two analyzed cases.

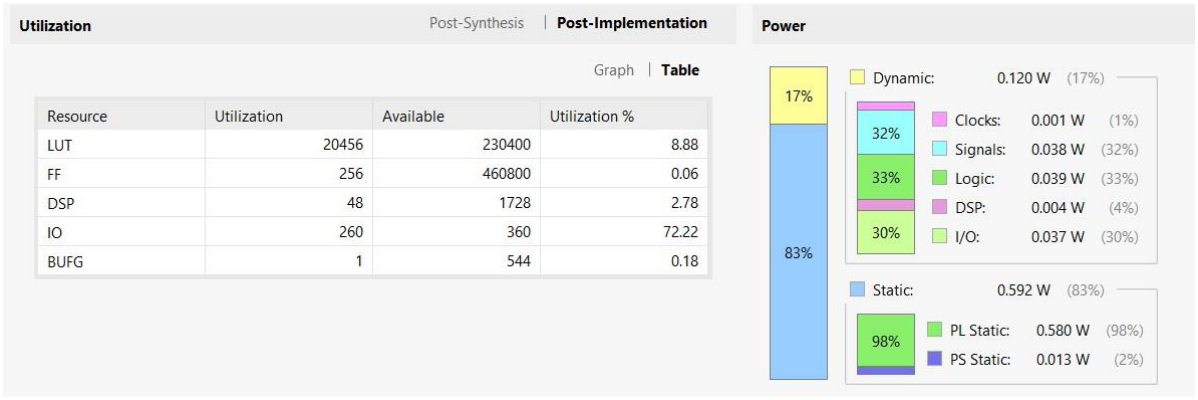

Figure 4: Implementation results range [-16:15]

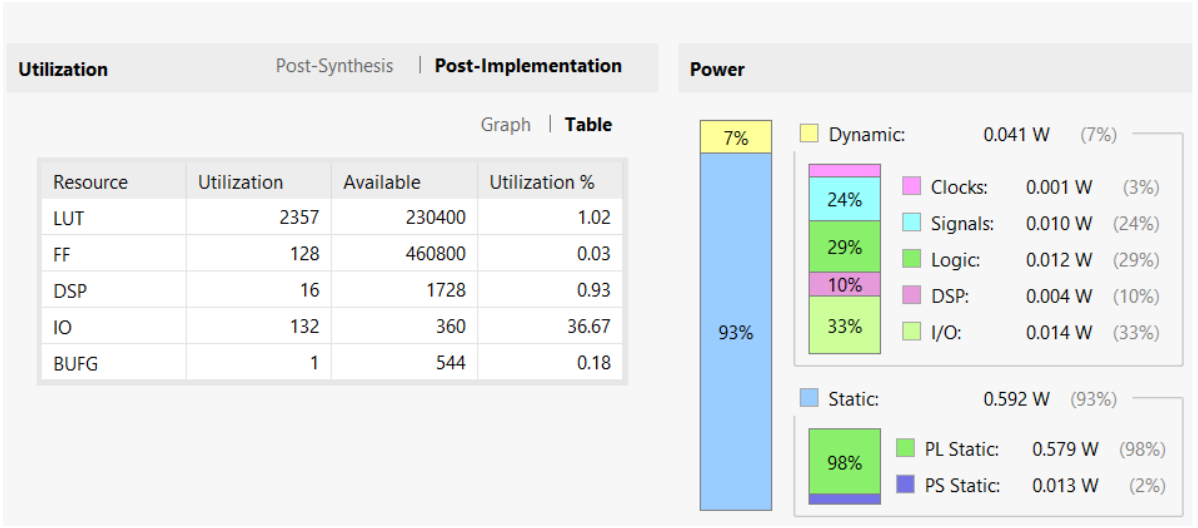

Figure 5: Implementation results range [-2:1]

4. Conclusion

In this paper, we presented the main features and potential of Xilinx RFSoCs. Thanks to RF converters (until 6 Gsps) they are very suitable for direct sampling of RF signals with bandwidth up to 4 GHz. As well as being fully supported from Vivado tool, these devices are also supported by other tools such us Mathworks Simulink.

References

[1] P. Ferroni, F. Zanzotto, N. Scarpato, A. Spila, L. Fofi, G. Egeo, A. Rullo, R. Palmirotta, P. Barbanti, F. Guadagni, Machine learning approach to predict medication overuse in migraine patients, Computational and Structural Biotechnology Journal 18 (2020) 1487–1496.

[2] R. Giuliano, G. C. Cardarilli, C. Cesarini, L. Di Nunzio, F. Fallucchi, R. Fazzolari, F. Mazzenga, M. Re, A. Vizzarri, Indoor localization system based on bluetooth low energy for museum applications, Electronics 9 (2020) 1055.

[3] F. Bonanno, G. Capizzi, G. Sciuto, C. Napoli, Wavelet recurrent neural network with semi-parametric input data preprocessing for micro-wind power forecasting in integrated generation systems, in: 5th International Conference on Clean Electrical Power: Renewable Energy Re-

sources Impact, ICCEP 2015, 2015, pp. 602–609.
[4] G. M. Khanal, G. Cardarilli, A. Chakraborty, S. Acciarito, M. Y. Mulla, L. Di Nunzio, R. Fazzolari, M. Re, A zno-rgo composite thin film discrete memristor, in: 2016 IEEE International Conference on Semiconductor Electronics (ICSE), IEEE, 2016, pp. 129–132.
[5] S. L. Ullo, G. R. Sinha, Advances in smart environment monitoring systems using iot and sensors, Sensors (Switzerland) 20 (2020) 3113.
[6] D. Gutiérrez, F. Giménez, C. Zerbini, G. Riva, Measurement of 4g lte cells with sdr technology, IEEE Latin America Transactions 18 (February 2020) 206–213.
[7] R. Giuliano, F. Mazzenga, A. Vizzarri, Satellite-based capillary 5g-mmtc networks for environmental applications, IEEE Aerospace and Electronic Systems Magazine 34 (2019) 40–48.
[8] C. Napoli, G. Pappalardo, E. Tramontana, A hybrid neuro–wavelet predictor for qos control and stability, in: AI*IA 2013: Advances in Artificial Intelligence, volume 8249, Springer International Publishing, 2013, pp. 527–538.
[9] T. Yoshizawa, S. B. M. Baskaran, A. Kunz, Overview of 5g urllc system and security aspects in 3gpp, in: IEEE Conference on Standards for Communications and Networking (CSCN), IEEE, Granada, Spain, 2019, pp. 1–5.
[10] G. Borowik, M. Wozniak, A. Fornaia, R. Giunta, C. Napoli, G. Pappalardo, E. Tramontana, A software architecture assisting workflow executions on cloud resources, International Journal of Electronics and Telecommunications 61 (2015) 17–23. doi:10.1515/eletel-2015-0002.
[11] G. Capizzi, S. Coco, G. Lo Sciuto, C. Napoli, A new iterative fir filter design approach using a gaussian approximation, IEEE Signal Processing Letters 25 (2018) 1615–1619.
[12] G. Du, C. Tian, Z. Li, D. Zhang, Y. Yin, Y. Ouyang, Efficient softmax hardware architecture for deep neural networks, in: Proceedings of the 2019 on Great Lakes Symposium on VLSI, 2019, pp. 75–80.
[13] I. Kouretas, V. Paliouras, Hardware implementation of a softmax-like function for deep learning, Technologies 8 (2020) 46.