Simulating Building Blocks for Spikes Signals Processing

Departamento de Arquitectura y Tecnología de Computadores
ETS Ingeniería Informática - Universidad de Sevilla
Av. Reina Mercedes s/n, 41012-Sevilla, Spain
angel@us.es

Abstract. In this paper we will explain in depth how we have used Simulink with the addition of Xilinx System Generation to design a simulation framework for testing and analyzing neuro-inspired elements for spikes rate coded signals processing. Those elements have been designed as building blocks, which represent spikes processing primitives, combining them we have designed more complex blocks, which behaves like analog frequency filter using digital circuits. This kind of computation performs a massively parallel processing without complex hardware units. Spikes processing building blocks have been written in VHDL to be implemented for FPGA. Xilinx System Generator allows co-simulating VHDL entities together with Simulink components, providing an easy interface for presented building block simulations and analysis.

1 Introduction

Living beings brains allow them to interact dynamically with their environment, unlike robotics systems that need in best cases a controlled environment, being a great current challenge to improve robots adaptability and cognitive skills. Neuromorphic systems provide a high level of parallelism, interconnectivity, and scalability; doing complex processing in real time, with a good relation between quality, speed and resource consumption. Neuromorphic engineers work in the study, design and development of neuro-inspired systems developed, like aVLSI chips for sensors [1][2], neuro-inspired processing, filtering or learning [3][4][5][6], neuro-inspired control pattern generators (CPG), neuro-inspired robotics [8][17] and so on. Neuromorphic engineering community grows every year as it demonstrate the success of the Telluride and Capocaccia Cognitive Neuromorphic workshops [9][10]. Spiking systems are neural models that mimic the neurons layers of the brain for processing purposes. Signals in spikes-domain are composed by short pulses in time, called spikes. Information is carried by spikes frequency or rate [11], following a Pulse Frequency Modulation (PFM) scheme, and also from other point of view, in the inter-spike-time (ISI) [5]. Spikes rate coded information can be processed in a continuous way, without codifying information into discrete samples. Because of the simplicity of these models, spikes processing do not need a complex hardware to perform

* This work has been supported by the Spanish project VULCANO (TEC2009-10639-C04-02).

information processing, in consequence, this hardware can be easily replicated performing a massively parallel information processing [13][14]. Previous work presented a set of components, designed as building blocks, to process spikes coded signals in real time, mimicking elementary analog components to design spikes filters equivalent to classical analog filters [18], written in VHDL to be stored in a digital devices like a FPGA. This way of processing spikes was previously called Spikes Signals Processing (SSP). This work is focused on presenting in a detailed way a simulation framework to test, measure and analyze SSP building blocks behavior. MATLAB with Simulink provides a good set of tools for simulating almost everything, so they are ideal for simulating our SSP blocks. However we need to simulate together VHDL components with Simulink blocks, for this purpose Xilinx provides a MATLAB toolkit (Xilinx System Generator) that allows the desired kind of simulation and many other things.

2 Simulation FrameWork

Simulation framework has two levels:

a) Lowest level is composed by a Simulink simulation model, which will contain SSP VHDL components and also Simulink ideal elements.
b) A MATLAB script is located in the highest level, this script will set up the component that will be simulated with its parameters. This script will also launch the simulation within, and after simulations it will reconstruct and analyze the spikes from the SSP VHDL components.

Figure 1 a) shows Simulink model, this model contains source stimulus, VHDL components. Stimulus signal is applied to a Synthetic Spikes Generator (this component will be the first studied in next section), which is used to convert Simulink signals to spikes. Those spikes will be processed by a SSP component (figure center). Each SSP component has its own set of parameters, which will be set up using constant blocks. Processed spikes (simulation outputs) will be monitored and analyzed by two Simulink components (Scope and a link to MATLAB workspace respectively). At figure bottom, it has been placed a set of Simulink elements that will perform the same processing to stimulus signal, but with continuous elements (without spikes), as S domain transfer functions. Ideal components will allow us to compare each SSP responses with their ideal equivalents. So for simulating any designed SSP component we only have to change the component under test, its parameters and its ideal transfer function, and we will be ready to simulate and to analyze exhaustively its behavior. Simulations of the Simulink model are managed by a high level MATLAB script. Figure 1 b) shows the flow chart of this script. First, it will set up simulation elements and parameters, then, it will launch the simulation, next, it will reconstruct output simulation spikes, and finally output analysis will be performed. Simulation output should be spikes fired by the simulated SSP blocks in time, which are a stream of short pulses. To analyze them it is necessary to transform output spikes to discrete numbers. For this task we have written in MATLAB a function that takes spikes output from simulation and returns a vector with the spikes average frequency for a fixed period of time (sample time). Sample time represents the number of clock cycles to measure average spikes frequency. The strategy

followed is to count the number of spikes fired during the sample time, being the average spikes rate the number of counted spikes divided by the sample time and the clock frequency used in simulations for VHDL components. Once spikes are converted to discrete numbers, we are ready to analyze simulation results.

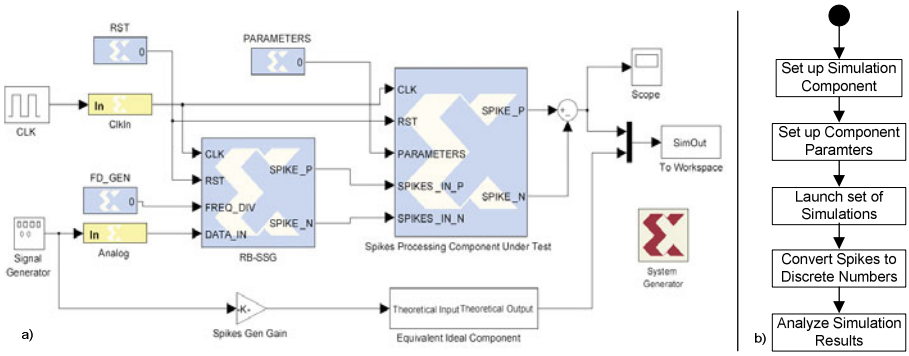

Fig. 1. A) Simulation scenario designed for spikes processing components testing. B) MATLAB script for simulation managing.

3 Building Blocks for SSP

In this section we make a brief introduction to SSP building blocks already presented in [18]. Then we show and comment simulations made for those SSP components in a detailed way. SSP components have been designed using four basic building blocks, and finally all of them can be combined for implementing, for example, a spike-based low pass filters.

3.1 Synthetic Spikes Generator (RB-SSG)

A Synthetic Spikes Generator (SSG) will transform a digital word (SSG input) into a frequency rate of spikes (SSG output). This element is necessary in those scenarios where is needed to transform digital signals to spikes, as commented before about Simulink source signals. There are several ways to design SSGs as presented in [15], selected SSG implements the reverse bitwise method found detailed in [16] for synthetic AER images generation. Figure 2 shows the internal components of the Reverse Bitwise SSG (RB-SSG) implemented. Since a reference or an analog signal can be negative, it is necessary to generate positive and negatives spikes. RB-SSG gain ($k_{BWSpikesGen}$) can be calculated as follows:

$$k_{BWSpikesGen} = \frac{F_{CLK}}{2^{N-1}(genFD + 1)} \quad (1)$$

Where Fclk represents system clock frequency, N the RB-SSG bits length, and genFD clock frequency divider value. RB-SSG has been the first element simulated, to verify (1) and its right behavior. Figure 2 b) bottom shows RB-SSG output spikes in time, and at top we can find spikes reconstructed frequency and theoretical value. Input stimulus signal from Simulink is a 2kHz sine, in consequence, RB-SSG output spikes

rate starts to increase, until input signal reach its maximum value. From that moment, input signal starts to decrease, as output spikes rate, crossing 0 and changing of sign. Output spikes also changes its sing, observing negatives spikes, although spikes if different signs are fired by different signals, from the point of view of MATLAB workspace, positive spikes are represented by '1', negative ones by '-1', and the absence of spikes by a '0'.

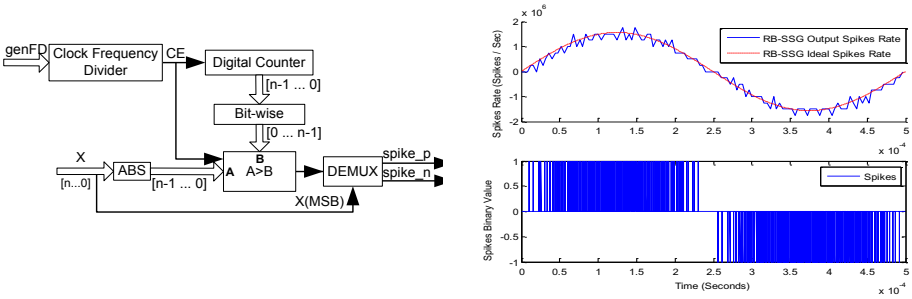

Fig. 2. A) Reverse Bitwise Synthetic Spikes Generator block diagram. B) RB-SSG output and reconstructed spikes rate, for sinusoidal input RB-SSG Simulation Parameters.

3.2 Spikes Integrate and Generate (SI&G)

Spikes Integrate & Generate (SI&G) is composed by a spikes counter, and by a RB-SSG, as showed in Figure 3 a). Spikes counter is a digital counter that is increased by one when a positive spike is received, and its value is decreased by one with a negative spike. Counter output is the RB-SSG input. Therefore, new spikes generated have a frequency proportional to spikes integration. The SI&G gain is set by RB-SSG parameters. With these considerations, SI&G spikes output frequency, $f_{I\&G}$, and SI&G gain, $k_{SI\&G}$, can be expressed as:

$$f_{SI\,\&G} = k_{SI\&G} * \int f_{inputSpikes} dt = \frac{F_{CLK}}{2^{N-1}(genFD+1)} \int f_{inputSpikes} dt \tag{2}$$

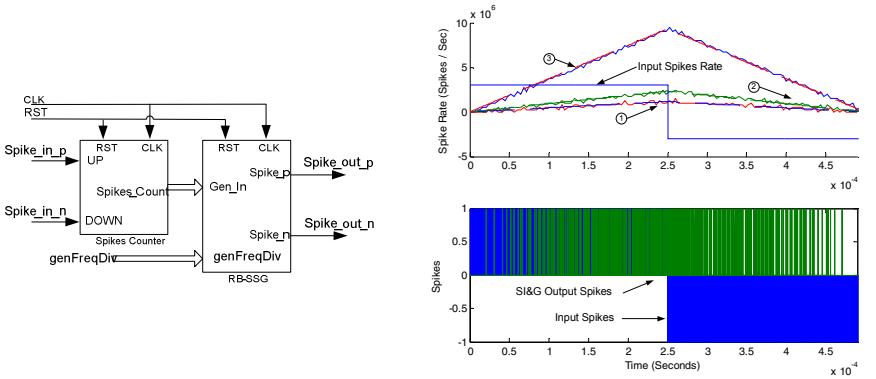

Fig. 3. A) Spikes Integrate & Generate block diagram. B) SI&G simulation output spikes rate reconstructed, top, and input/output spikes, bottom.

Similarly to analog systems, we can calculate equivalent SI&G transfer function in "spikes-domain" using Laplace transform. SI&G transfer function is presented in (2), being equivalent to an ideal integrator with a gain of $k_{I\&G}$.

$$SI\&G(s) = \frac{F_{SI\,\&G}\,(s)}{F_{inputSpikes}(s)} = \frac{k_{SI\&G}}{s} = \frac{F_{CLK}}{2^{N-1}(genFD+1)*s} \tag{3}$$

For SI&G testing purpose we have executed a set of simulations for different pairs of parameters, in Table 1, and a constant rate of input spikes. Figure 4 b) shows SI&G temporal simulation results, at figure top is represented the frequency of input spikes in blue (a step signal), and three SI&G frequency output spikes couples for different parameters. These couples are composed by simulated reconstructed output (solid line) and theoretical response (discontinuous line). At figure bottom we can see SI&G inputs spikes in blue, and also SI&G output spikes in green. Input spikes represent a step signal, having a constant spike rate, SI&G output spikes have a constant linear frequency increasing when input spikes are positives, like a ramp, with a slope equivalent to $k_{SI\&G}$, and also decreasing output spikes frequency with negative spikes, as expected from an ideal analog integrator with same features. Simulations also denoted a good accuracy with theoretical responses.

Table 1. SI&G Simulation Parameters

Sim. Case	N (Bits) - genFD	$k_{SI\&G}$
1	13 - 0	$12.207*10^3$
2	14 – 1	$3.052*10^3$
3	16 – 0	$1.526*10^3$

3.3 Spikes Hold&Fire (SH&F)

This block performs the subtraction between two spikes stream. SH&F will allow us to implement feedback in the SI&G block, obtaining new transfer functions. Subtracts a spikes input signals (f_U) to another (f_Y), means to get a new spikes signal which spike rate ($f_{SH\&F}$) will be the difference between both inputs spikes rate:

$$f_{SH\&F} = f_U - f_Y \tag{4}$$

The procedure of the SH&F is to hold the incoming spikes a fixed period of time while monitoring the input evolution to decide output spikes: holding, canceling, or firing spikes according to input spikes ports and sign. This block has been successfully used previously to design spike-based closed-loop control systems in mobile robots by authors [17][18]. And will be used later to feedback the SI&G to design spikes filters [18]. Figure 4 contains simulation results, SH&F has been excited with two saw tooth signals with different amplitude and sign. At simulation begin input spikes has opposed sign, in consequence output spikes frequency is the addition of inputs spikes frequency. When input spikes frequency is the same, there is no output spikes, with a frequency of 0. Finally, in the case that both inputs spikes have the same sign, SH&F output spikes are the difference between both signals, performing perfectly the subtraction operation.

3.4 Spikes Frequency Divider (SFD)

This block will divide the spike rate of an input signal by a constant. To ensure spikes distribution we have implemented this block inspired in the way that RB-SSG works. Figure 5 shows SFD internal components. In SFD *spikesDiv* behaves like the constant to divide. Output buffers only enable output spikes according *spikesDiv* homogenously in time. SFD transfer function can be calculated using (5), where N represents the SFD number of bits, and *spikesDiv* the signal presented before. SFD transfer function is equivalent to a gain block with a value in the range of [0, 1], with 2^N possible steps. SFD accuracy can also be adjusted with N (N=16 bits in Fig.5) getting an accuracy of 2^{-N}.

$$k_{SFD} = \frac{F_{outSpikes}}{F_{inputSpikes}} = \frac{spikesDiv}{2^N} \tag{5}$$

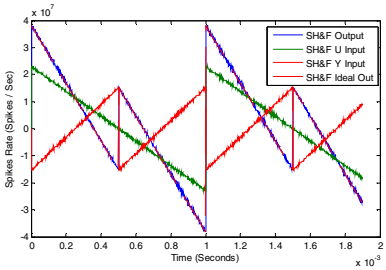

Fig. 4. Spike Hold&Fire temporal simulations

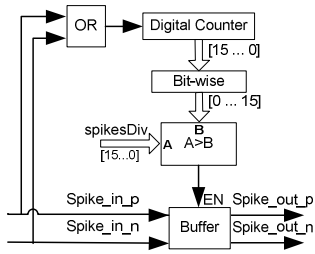

Fig. 5. Spikes Frequency Divider internal components

3.5 Spikes Low-Pass Filter (SLPF)

As analog frequency filters modifies signal frequency components from amplitude changes, spikes filters will work on spikes rate changes frequency components. For example, a spikes low pass filter will attenuate spikes rate high frequency components, if constant spikes rate signal is applied to spikes filter input, as an input step, spikes output frequency will start to increase exponentially, as expected from an analog signal amplitude, until spikes rate reach some frequency. To build a SLPF a SI&G have been feedback using a SH&F and two SFD, as it is shown in Figure 6.
Next equation shows SLPF ideal transfer function, where SI&G(s) can be obtained from (3), k_{SDout} represents output SFD gain and k_{SDFB} the gain of the SFD placed in the feedback loop, both detailed in (5).

$$F_{SLPF}(s) = \frac{k_{SFDOut} * SI\&G(s)}{1 + k_{SFDFB} * SI\&G(s)} = \frac{k_{SFDOut} * k_{I\&G}}{s + k_{SFDFB} * k_{I\&G}} \tag{6}$$

Finally we are going to simulate SLPF with different parameters. So different features spikes filters have been simulated, fixing different parameters sets, getting equivalent

filters with various cut-off frequencies and gain. Simulations results are presented in Figure 7, this figure is composed by two different kinds of simulations, in a) there are spikes filters temporal response, and in b) the frequency response. As first simulations we present a set of temporal simulations, Figure 7 a), simulating spikes filters for a fixed time, being X axis simulation time, and Y axis instantaneously spikes rate. For temporal simulation these spikes filters have been excited by a constant spike rate input, as an analog voltage step. Filter output spikes have been analyzed and their instantaneous frequencies have been reconstructed. Theoretical responses of analog ideal equivalent filters have been also added to the figure, to compare simulation response respect to ideal response. In the case of SLPF (1-5), output spikes frequency start to increase exponentially, like expected from an ideal low pass-filter, with a rise time near to 4 times $ù_{cut\text{-}off}$. SLPF output spikes rate reach a steady state value proportional to SLPF gain, reaching input spikes frequency when is 1 (1-3), higher output spikes rates in the case that SLPF gains is higher than 1 (5), and opposed effect happens when SLPF gains is lower than 1 (4).

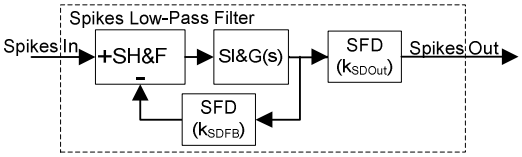

Fig. 6. Spikes Low Pass Filter Architecture

Table 2. SLPF Simulation Parameters

Sim. Case	N (Bits) - genFD	Spikes Div. Out	Spikes Div. Feedback	$ù_{cut\text{-}off}$	Filter Gain (abs)	Rise Time (mSec)
1	13 – 0	0.5147	0.5147	1kHz	1	0.6366mSec
2	11 – 0	0.634	0.634	5kHz	1	0.1273mSec
3	9 – 0	0.634	0.634	20kHz	1	0.0318mSec
4	13 – 0	0.6691	0.5147	1kHz	0.5	0.6366mSec
5	13 – 0	0.6691	0.5147	1kHz	1.3	0.6366mSec

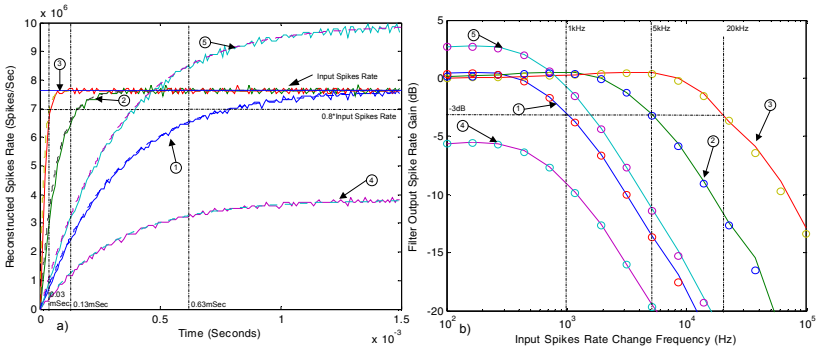

Fig. 7. Simulation Results: a) Spikes Filters time response b) Spikes Filters Bode diagram

After simulating the temporal behavior of designed spikes filters, we have studied their frequency responses. If our aim were to characterize an analog filter in frequency domain, one simple way to perform this task could be to excite the analog filter with pure frequency tones, and annotate analog filter output power for each tone. Translating this experiment to spikes domain, we are going to stimulate RB-SSG with an input sinusoidal signal, getting a spikes output which frequency changes according the sinusoidal signal, codifying a pure tone as spikes rate changing. Generated spikes will stimulate the spikes filters, whose output will be a spikes stream with same input frequency tone, but with its amplitude and phase modified by the spikes filters. So we have exited every spikes filter with a set of spikes coded tones, and recording all the spikes filter output power at that tone, obtaining finally the spikes filter Bode diagram. Results of these simulations are presented in Figure 7 b), X axis represents the frequency of input tones in Hz, and Y axis contains the spikes filter gain in dB. Figure contains SLPF simulated and theoretical responses marked with a cross, where theoretical responses are marked in these cases with circle. SLPF with a gain of 1 (1-3) have predicted gain of 0dB in the pass band, and then gain starts to decrease when frequency is near to cut-off frequencies, cutting them with a gain of 3dB, providing an attenuation of 20dB by decade as expected of analog filters. SLPF with different gains (4-5), having a gain in the band pass of -6dB and +2.6dB respectively.

4 Conclusions

This paper presents a framework for the simulation of building blocks to process signals using a spike rate coded representation. Simulations have been possible thanks to the power, flexibility and scalability of Simulink and Xilinx System Generator, providing an excellent scenario to co-simulate VHDL files with other kind of components. We have also presented the SSP building blocks that had been simulated, analyzing, and comparing with ideal components, in an exhaustive way for each presented SSP block. Simulations have denoted an accurate behavior of designed SSP blocks compared with ideal results, validating theoretical equations. Presented framework is nowadays being used in our lab for designing and testing new SSP building blocks that will improve current blocks features, and they will also allow the development of more complex SSP components.

References

1. Lichtsteiner, P., et al.: A 128×128 120dB 15 us Asynchronous Temporal Contrast Vision Sensor. IEEE Journal on Solid-State Circuits 43(2) (February 2008)
2. Chan, V., et al.: AER EAR: A Matched Silicon Cochlea Pair With Address Event Representation Interface. IEEE TCAS I 54(1) (January 2007)
3. Serrano-Gotarredona, R., et al.: On Real-Time AER 2-D Convolutions Hardware for Neuromorphic Spike-Based Cortical Processing. IEEE TNN 19(7) (July 2008)
4. Oster, M., et al.: Quantifying Input and Output Spike Statistics of a Winner-Take-All Network in a Vision System. In: IEEE International Symposium on Circuits and Systems, ISCAS 2007 (2007)

5. Hafliger, P.: Adaptive WTA with an Analog VLSI Neuromorphic Learning Chip. IEEE Transactions on Neural Networks 18(2) (March 2007)
6. Indiveri, G., et al.: A VLSI Array of Low-Power Spiking Neurons and Bistables Synapses with Spike-Timig Dependant Plasticity. IEEE Transactions on Neural Networks 17(1) (January 2006)
7. Gomez-Rodríguez, F., et al.: AER Auditory Filtering and CPG for Robot Control. In: IEEE International Symposium on Circuits and Systems, ISCAS 2007 (2007)
8. Linares-Barranco, A., et al.: Using FPGA for visuo-motor control with a silicon retina and a humanoid robot. In: IEEE International Symposium on Circuits and Systems, ISCAS 2007 (2007)
9. Telluride Cognitive Neuromorphic workshop, `https://neuromorphs.net/`
10. Capo Caccia Cognitive Neuromorphic workshop, `http://capocaccia.ethz.ch`
11. Shepherd, G.: The Synaptic Organization of the Brain. Oxford University Press, Oxford (1990)
12. Chicca, E., et al.: An event based VLSI network of integrate-and-fire neurons. In: IEEE International Symposium on Circuits and Systems, ISCAS 2004 Misha Mahowald. VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function. PhD. Thesis, California Institute of Technology Pasadena, California (1992)
13. Serrano-Gotarredona, R., et al.: CAVIAR: A 45k-neuron, 5M-synapse AER Hardware Sensory-Processing-Learning-Actuating System for High-Speed Visual Object Recognition and Tracking. IEEE Trans. on Neural Networks 20(9), 1417–1438 (2009)
14. Gomez-Rodriguez, F., et al.: Two Hardware Implementation of the Exhaustive Synthetic Aer Generation Method. In: Cabestany, J., Prieto, A.G., Sandoval, F. (eds.) IWANN 2005. LNCS, vol. 3512, pp. 534–540. Springer, Heidelberg (2005)
15. Paz-Vicente, R., et al.: Synthetic retina for AER systems development. In: International Conference on Computer Systems and Applications, AICCSA 2009 (2009)
16. Jiménez-Fernández, A., et al.: AER-based robotic closed-loop control system. In: IEEE International Symposium on Circuits and Systems, ISCAS 2008 (2008)
17. Jiménez-Fernández, A., et al.: AER and dynamic systems co-simulation over Simulink with Xilinx System Generator. In: IEEE International Conference on Electronics, Circuits and Systems, ICECS 2008 (2008)
18. Jimenez-Fernandez, A., et al.: Building Blocks for Spike-based Signal Processing. In: IEEE International Joint Conference on Neural Networks, IJCNN 2010 (2010)