XtokaxtikoX: A Stochastic Computing-Based Autonomous Cyber-Physical System

Rui Policarpo Duarte and Horácio Neto
INESC-ID, Instituto Superior Técnico
Universidade Técnica de Lisboa, Portugal
Email: rui.duarte@tecnico.ulisboa, hcn@inesc-id.pt

Mário Véstias
INESC-ID, Instituto Superior de Engenharia de Lisboa,
Instituto Politécnico de Lisboa, Portugal
Email: mvestias@inesc-id.pt

***Abstract*—This paper presents XtokaxtikoX, a fully autonomous cyber-physical system employing only stochastic arithmetic to perform computations on its data-path. Traditional implementations of stochastic computing systems benefit from fast and compact implementation of arithmetic operators, and high tolerance to errors, but depend heavily on the conversion between stochastic bitstreams and binary to implement many parts of the system. Furthermore, if a system requires any interaction with analog electronic components it must have additional ADC/DAC conversion circuitry, which further increases the complexity of the system. Conversely, the proposed work is able to directly translate analog signals into stochastic bitstreams, process the stochastic bitstreams and finally control analog actuators relying only on the information on the stochastic bitstreams. Details on the architectures to accomplish such functionality are presented as well as other stochastic arithmetic units. This paper also presents a small stochastic computing-based autonomous cyber-physical system implemented on a Cyclone IV FPGA to carry out a proof-of-concept.**

Keywords: stochastic computing, FPGA, cyber-physical reconfigurable system, fault tolerance

I. Introduction

Low resource consumption and complexity implementation of arithmetic operators, high resilience to errors, and high performance of Stochastic Computing has attracted a lot of interest from the research community to this computational paradigm. Its main characteristic is the encoding of data as pseudo-random bitstreams of 0s and 1s. The value encoded on a bitstream represents a number between 0 and 1 and it is defined by the ratio of the number of 1s over the total number of bits.

This concept was introduced by [1] as an alternative to binary systems. Nevertheless, stochastic computing has been applied to a limited set of applications which include DSP applications: Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters [2], [3], neuromorphic and bio-inspired systems: binary synapses for low-power neuromorphic systems [4], digital neurosynaptic network for neuromorphic chips to develop brain-like computational structures [5], decoding of Low-Density Parity Code (LDPC) codes [6], [7], and Bayesian computing machines [8]. Most of these applications are based on additions and multiplications, and are tolerant to some errors in their computations. Therefore, in the event of occurring a bit-flip, or transmission error, on any bit of a stochastic bitstream its impact won't make the result deviate as much as it would one of the most significant bits in binary representation. Nevertheless, interfacing analog sub-systems such as sensors, which are common in cyber-physical systems, requires Analog to Digital Converter (ADC) and/or Digital to Analog Converter (DAC) circuits and the corresponding controller for their interface on the digital system. In this scenario the main advantages of stochastic computing rapidly vanish because there's a constant need to convert data from binary to stochastic bitstreams, and vice-versa.

This work addresses this problem by proposing novel architectures for stochastic units, which exploit the nature of the pseudo-random bitstreams, to directly interface analog electronic circuits, hence avoiding costly conversions. In more detail, the waveform of a stochastic bitstream can be recognized as a Pulse Width Modulation (PWM) signal with a pseudo-random duty-cycle. Examining particular cases of encoded values close to zero and close to one, it is close to PWM signals with a very small and very high duty cycles, respectively. Hence, an analog output is directly derived from a stochastic bitstream requiring an R-C network for low-pass filtering, and in some cases, an output buffer to supply the correct voltage and current to the analog circuit. Complementarily, the determination of an analog voltage level is done using a programmable voltage comparator circuit (op-amp) and a voltage reference. If a periodic triangular voltage reference is used then the output of the voltage comparator circuit will be HIGH when the input voltage is greater, and LOW otherwise. Thus, generating a sequence of pulses whose width, or duty-cycle, is proportional to the analog voltage level. Moreover, considering the aforementioned analog voltage source, it can be derived from a PWM signal, which can be generated inside the Field-Programmable Gate Array (FPGA). Thus, it is possible to generate a triangular voltage reference inside the same device without adding extra controllers or sub-systems.

In terms of implementation, FPGAs were adopted since their bit-level circuit specification is inline with the specification of stochastic computational systems; and their reconfiguration power, which allows to test different stochastic systems on the same device, and with different design parameters. Even though other competitive technologies such as Complex Programmable Logic Device (CPLD) offer the same functionality, when targeting very small systems, they don't scale up support other real-life systems.

The main contributions of this work are:

- novel arithmetic architectures in stochastic computing;

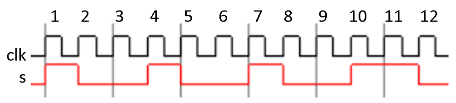

Fig. 1. Example of a stochastic bitstream.

- novel stochastic computing units to provide I/O support to interface external analog signals;
- XtokaxtikoX[1], a fully autonomous stochastic cyber-physical system to demonstrate the proposed architectures.

The remaining of this paper is organized as follows: Section 2 provides the background on stochastic computing to frame the proposed stochastic units, presented in Section 3. Section 4 presents the organization of the proposed architectures which are demonstrated in section 5. The paper ends with the conclusions in Section 6.

II. Background

J. Von Neumann introduced Stochastic computing in [1] as a method to design probabilistic logic circuits and synthesize robust systems from unreliable components. Later, in [9], Gaines has introduced the use of stochastic streams to represent operators with high levels of error tolerance.

Stochastic computing has been applied to design many applications which include: digital filters FIR [10], IIR [2], neural networks [11][12], decoding of error correcting LDPC codes [6], [7], high-throughput Bayesian computing machines [8], and probabilistic neural networks [13], [12].

A stochastic signal is characterized as result of a stochastic process which produces binary values. A stochastic bitstream is defined as a sequence of stochastic signals over time whose value is determined by the number of 1s over the total number of bits. Hence, a stochastic number is represented as a ratio between $[0, 1]$. [14] has presented a survey on the stochastic arithmetic units covering the most common arithmetic units. This contribution also introduces details about hardware realizations of the building-blocks for stochastic systems.

Figure 1 exemplifies a small pseudo-random bitstream. On top (in black) the clock signal is used for synchronization; and below (in red) the stochastic bitstream. In this example, the encoded value, which is given by the number of 1 bits over the total number of bits, that is $\frac{5}{12} \approx 0,41667$.

A. Stochastic Arithmetic Building-Blocks

To conduct arithmetic computations some stochastic arithmetic architectures have been proposed which include: adder, complement, multiplier, and a squarer, as in figures 2 and 3.

These arithmetic operations are performed over time on the pseudo-random bitstream and thus operate on the bit level, which require very little resources, when compared to typical binary implementations. The stochastic multiplication is supported by the logic AND of its stochastic inputs. The complement is the negation of the bitstream. Addition, or more

[1]The Xs in the name are used as homophonous to emphasize constant signal transitions on the stochastic bitstream

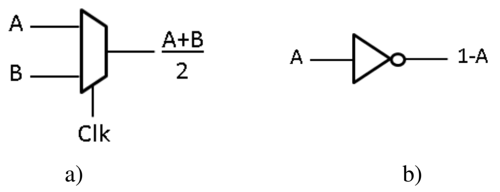

Fig. 2. Logic representation of the a) 2-input stochastic adder and b) complement.

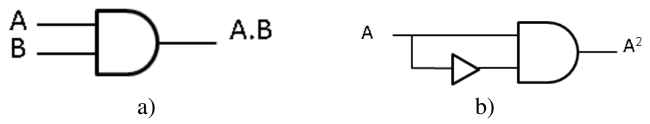

Fig. 3. Logic representation of the stochastic multiplier (left) and squarer (right).

precisely average, is obtained via a round-robin multiplexation of the stochastic inputs, which depends on a N-module counter corresponding to N inputs in the multiplexer. The squarer of a stochastic bitstream is the equivalent to a multiplication of a bitstream by itself, de-phased by a delay element (asynchronous) or a clock cycle with a register (synchronous).

Even though the few resources required are of great interest, the main fragilities of stochastic computing are: exponential increase in the length of the bitstream with the linear increase in the precision of stochastic bitstream; sensitivity to temporal correlations; and the supporting blocks are usually the performance bottleneck. Nevertheless, [15] proposed a method to reduce the sensitivity to temporal correlations via spread-spectrum, individual and uncorrelated clock sources, based on Self-Timed Ring-Oscillators (STROs).

Since most conventional computing data sources and collectors use binary representation, a stochastic converter from/to binary is required to interface stochastic computing systems, as shown in figure 4 and figure 6. The process of generating the stochastic bitstream is illustrated in figure 5, where a specific binary value (*val*) is compared with the output of a uniform pseudo-random generator. Whenever the pseudo-random number is smaller, it produces a 1 and 0 otherwise. Therefore, the ratio between the number of ones and the total number of bits converges to *val*. In this example, the encoded value is $9/16 = 0,5625$. On the other hand, the conversion from stochastic-to-binary is based on the integration of the 1s on a bitstream, which is accomplished using a binary counter. A second counter is required to count the total number of bits.

Regarding sensitivity to errors on the bitstream, [16] has presented results where the quality of the results degrades gracefully with the increase of errors.

To bring more detail into the implementation and synthesis of the aforementioned components, the Register Transfer Level (RTL) for some units is presented. The binary-to-stochastic unit is presented in figure 7, and it is comprised of a pseudo-random generator connected to a binary comparator. In this case the unit is configured to use an internal STRO as clock source for its synchronous logic, instead of the global clock signal, as proposed in [15]. Inputs of this unit are: reset, load, seed and binary number to be encoded.

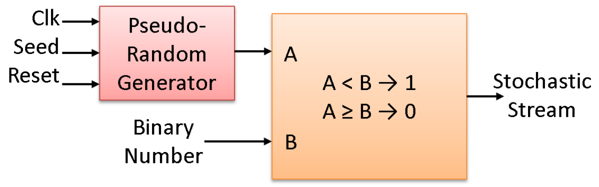

Fig. 4. Block diagram of a binary-to-stochastic unit.

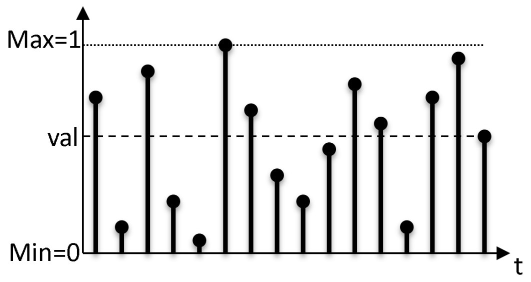

Fig. 5. Detail on process of generating a pseudo-random bitstream for a given binary value.

Figure 8 shows the RTL, for a 3-input stochastic multiplier. It corresponds to the logic AND of its inputs, and hence and doesn't require a clock signal. The stochastic adder, or average, is shown in figure 9. It consists of a multiplexer and a modulo-3 counter attached to its input selector.

III. Proposed Stochastic Units

A. Multiplication by Factor of 2 (Double)

At the moment the multiplications supported in stochastic computing are the based on the Boolean AND, and therefore they consider normalized inputs, which will produce a bit-stream lesser or equal than one, e.g. $0.5 \times 0.4 = 0.2$. As a consequence, the result of a series of multiplications on a

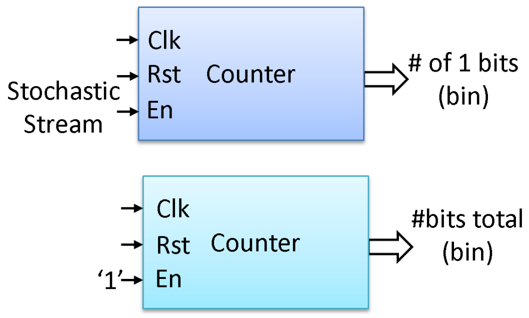

Fig. 6. Block diagram of a stochastic-to-binary unit.

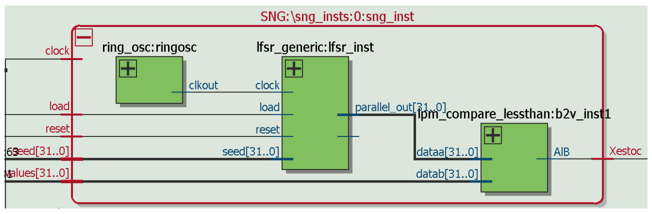

Fig. 7. RTL of the Stochastic Number Generator for a stochastic bitstream.

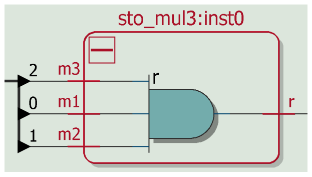

Fig. 8. RTL of a stochastic multiplier with 3 inputs.

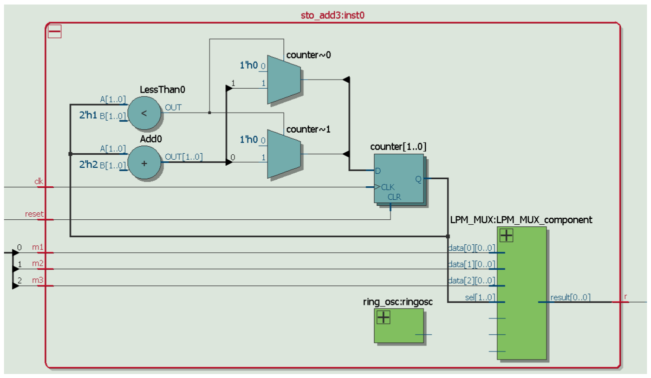

Fig. 9. RTL of a stochastic adder with 3 inputs.

bitstream will tend to zero. Keeping in mind that this can phenomenon can occur, a novel multiplication unit is here proposed to scale up.

Considering the bitstream as a pseudo-random sequence of zeros and ones, it is not straightforward to envision a method double the number ones in it. Moreover, it is assumed that converting to binary, performing a multiplication and converting the result back to stochastic is too computational expensive to be considered as a possible solution, and this can be only correctly applied to bitstreams encoding values lesser or equal than 0.5.

Taking advantage of the approximate results inherent to Stochastic Computing, and the low-complexity of the arithmetic operands, the solution here proposed is based on the simple idea of repeating a 1 whenever it appears on the bitstream. If the first bit of a bitstream and its predecessor are connected to a logic OR gate, its output will present a 1 whenever any of the two is 1, hence fulfilling the purpose of the unit. Thus far, this solves the problem, notwithstanding it's not an ideal solution as the *replicated* bit will always follow the original bit, hence generating correlations on the new bitstream. Moreover, even though this unit could be adapted to add any number of bits, there would be greater impact on sensitivity to statistic correlations, as bitstreams of consecutive 1s are formed on the new bitstream.

In the original bitstream, if two consecutive bits are 1 then the unit will only add one 1 instead of two, which will be placed after the second 1. In other words, even though the unit is replacing the consecutive bit with a new one, it will overwrite the bit being replaced, hence not producing an extra one bit. Nevertheless, the unit doesn't ignore any of the bits whenever they are consecutive.

The block diagram for the proposed unit is presented in figure 10 and its operation is illustrated in table I. The top row shows a small sample of a given bitstream and the bottom row

Bit Num	8	7	6	5	4	3	2	1
Original	0	0	1	0	0	1	1	0
Double	0	1	1	0	1	1	1	0

TABLE I. OPERATION OF THE DOUBLE UNIT OVER A PSEUDO-BITSTREAM, FROM RIGHT TO LEFT.

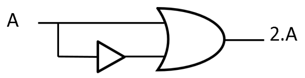

Fig. 10. RTL of the multiplier by 2 arithmetic unit

the output of the proposed double unit.

B. Stochastic I/O Interface

The proposed I/O interface offers interconnection of analog systems with stochastic bitstreams, without using ADCs and DACs, and it is made of two fundamental blocks: stochastic analog input and stochastic analog output.

The output block is accomplished by passing a stochastic bitstream to an output port, as it is similar to a PWM signal. A bitstream holding the value 0 is similar to a PWM signal with a 0% duty cycle, whereas a 1 is similar to a 100% duty-cycle. In-between values have their ON time proportional to the analog signal voltage. To smooth the pulses of the PWM signal, a R-C network is connected to the output pins of the FPGA. The values of R and C are adjusted depending on the problem, stability of the generated analog signal and its time/frequency response. Since the maximum current supplied by the FPGA's output ports is very limited, it may be worth connecting a buffer between both systems. Moreover, this buffer also isolates the impedance of the R-C network and the load being driven.

The input block is responsible for the determination of the analog voltage present at its input. The determination of an analog voltage level is done using a voltage comparator circuit (op-amp) and a voltage reference. The output of the voltage comparator circuit will be HIGH when the input voltage is greater, and LOW otherwise. If this process is repeated for a range of analog voltages, then it is possible to determine the analog voltage present at the comparator's input. With this is mind, using a periodic triangular signal as voltage reference it is possible to generate a sequence of pulses whose width, or duty cycle, related to the analog voltage level. Moreover, considering the analog voltage source described above, at the expense of a PWM signal, which can be generated inside the FPGA, it is possible to generate the triangular voltage reference on the same device without adding complex voltage regulators, controllers or sub-systems, as depicted in figure 11.

Considering a triangular wave signal with 50% duty-cycle, and an analog voltage half the power supply of the comparator, it produces, at its output, a binary signal which is HIGH half the time. The stochastic signal is achieved by maintaining the 50% duty-cycle but changing the period of the signal pseudo-randomly, since it generates longer and shorter pulses. This is of great importance as it generates extra entropy and reduce correlation between bitstreams. This concept is illustrated in figure 12.

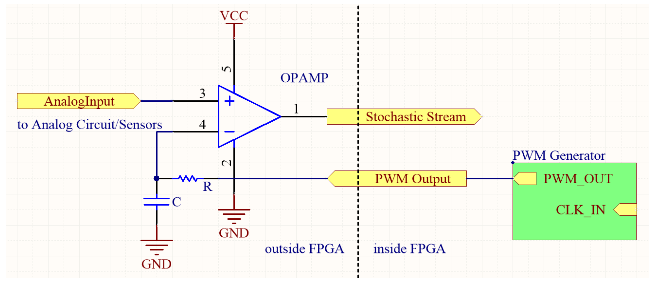

Fig. 11. Stochastic input circuit schematic.

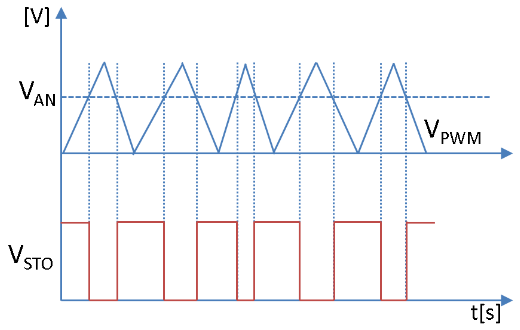

Fig. 12. Principle of working of the proposed stochastic stream generator.

IV. STOCHASTIC SYSTEM ARCHITECTURE

In a system which incorporates the proposed stochastic elements, it is now possible to interface analog signals without the necessity to use extra components such as an ADC followed by binary-to-stochastic converter and an stochastic-to-binary followed by a DAC. The proposed stochastic system follows the architecture presented in figure 13. It interfaces the analog system via the proposed input block to read analog signals (An2Sto), usually from sensors, and then a Stochastic Machine which is responsible for all the digital processing in the data-path. The outputs of the system are then placed at the output of the system (Sto2An) to *attack* the analog actuators. The simplicity, modularity and regularity of the system allows it to scale to problems of any size, being limited by the resources available on the reconfigurable device.

V. EVALUATION & DEMONSTRATION

To demonstrate the proposed advancements a cyber-physical system was envisioned to act as a line-follower. It is based on a small off-the-shelf line-follower platform, without any control logic, making only use of its sensors and motors. The new controller, implemented on an FPGA, receives information from the analog sensors (photo-resistor) and sends information to the analog actuators (DC motors).

As its principle of operation, the line-follower senses a

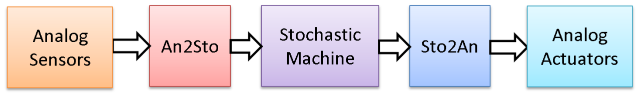

Fig. 13. Top-level architecture of a Stochastic System, including the supporting units connected to the analog sub-systems.

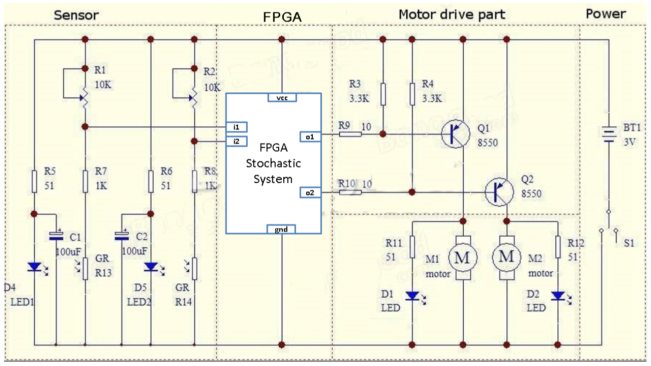

Fig. 14. Circuit diagram for the supporting elements connected to the stochastic system on the FPGA.

black line on its floor and moves forward following it. It has two independent wheels, each attached to a small electric motor, which are used to drive and change direction. The direction is estimated from the information provided by the two photo-resistors, one at each side on its front. Whenever the photo-resistors detect darkness it means that the line-follower is crossing the black line, and assuming it was previously in the correct position, it must avoid moving in that direction, and therefore, it should reduce the speed or even stop the corresponding electric motor. On the other hand, whenever light is detected in means that the photo-resistor is far from the black line, and hence it can move at full speed. Figure 14 shows the circuit diagram for the implemented system. On the left, there are the LEDs, which provide a constant light source to the floor, and the photo-resistors, which sense the reflected light from the floor. In the middle, there's the FPGA which holds the Stochastic System. It is considered that the proposed I/O ports belong in the system. On the right, there are the motors and its drivers. The system is powered from the FPGA board which can be supplied from a USB port or a battery.

A. Inputs

The sensors, or inputs, of the cyber-physical system considered are the two photo-resistors. They are connected to a resistor network to create a voltage divider controlled by the light they receive. One of these resistors is potentiometer to allow calibration of the sensor. In normal operating conditions, the resulting voltage is approximately 1.5 V when in the presence of light, and 3 V when in darkness. The two input voltages are sensed by the Stochastic System via the new stochastic input blocks that convert an analog voltage into a pseudo-random bitstream with the corresponding value. The circuit uses LM339 op-amps to act as comparators. The PWM signal generated inside the FPGA to act as reference voltage was modified to include small variations in its amplitude. Such variations were thought to introduce glitches in the digital signal so that it acquires more transitions when close to the input voltage, rather than simply setting a unit step signal.

Figures 15 and 16 present all the main signals involved in the conversion of two analog voltages, of 3V and 1.5V, into two stochastic bitstreams. In each figure there are 3 signals: analog voltage at the input (blue,ch1), voltage reference from a PWM generator inside the FPGA (green,ch3) and the digital output

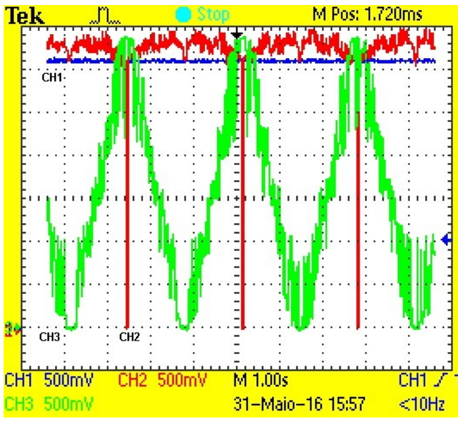

Fig. 15. Analog, comparison and stochastic bitstream for a voltage of approximately 3V.

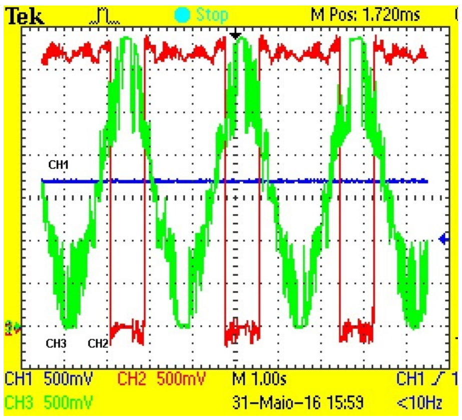

Fig. 16. Analog, comparison and stochastic bitstream signal for a voltage of approximately 1.5V.

from the comparison (red,ch2). In both figures it is observable that the duty-cycle of the digital waveform is proportional to the analog voltage.

B. Outputs

The actuators, or outputs, of the system are the small DC motors attached to the wheels of the line-follower to make it move, and are to be controlled by the stochastic system. In this case, the outputs of the system have the following specifications:

- a load can be controlled via a PWM signal, which is similar to a stochastic signal;
- electric motors don't cope with high-frequency control signals, thus requiring a low-pass filter;
- the output pins from the FPGA board have enough current to actuate the motor drivers.

In this case, the output of the stochastic system is connected directly to the motor drivers via an R-C network that acts as a passive low-pass filter, e.g. $\tau = [1; 1000]\, ms$.

C. Controllers

To test the cyber-physical system two controller designs were created from mathematical expressions. Given the sim-

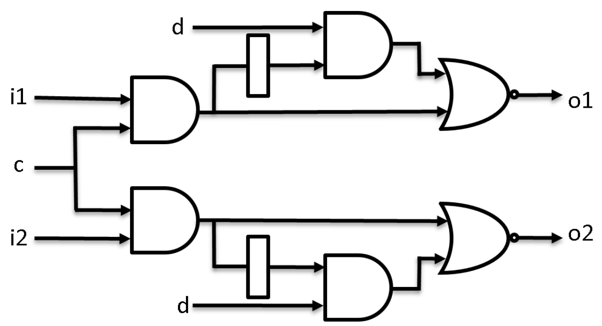

Fig. 17. RTL of the controller I.

plicity of the system, these designs weren't thought towards high performance and efficiency but rather to employ the proposed units.

1) Controller I: This is an elementary linear proportional controller for which each output is given by the inverse of the reading from the sensor, and weighted with a value c that acts as a speed limiter, as described in equations 1 and 2. This controller doesn't perform any check on the validity of the values at the inputs, i.e. within normal operating range. Figure 17 shows the RTL for this controller. To slow down the motors, each input bitstream is multiplied by c using the logic ANDs. To speed up the DC motors, each input stream can be multiplied by a constant factor of two, whenever signal an activation signal d is present. Because of the control logic the end result if negated to produce the correct actuation level on the DC motors.

$$o_1 = i_1 \times c \tag{1}$$
$$o_2 = i_2 \times c \tag{2}$$

2) Controller II: The second controller is a on-off controller as it introduces a threshold that needs to be met before activating each electric motor. In this scenario, when each of the inputs is lower than the programmed threshold, their MSbit at the output is 1. In more detail, given two pseudo-random bitstream connected to an up(down counter, means that a 1 in one of the bitstreams counts up and the other down. When both are 1, there's no counting. If the two bitstreams encode the same value, over a period of time they will cancel each other out. Contrarily, if they encode different values, then the output of the counter is positive or negative, in 2s complement. Looking at the MSbit it is possible to tell which one of the two has the greatest value.

The threshold signal is also a pseudo-random bitstream and hold a value between 0 and 1. In practice, when the input signal is far from the threshold there will be no stochastic bitstream generated, but instead a continuous 1 or 0, depending on the signal. This is desirable as it related with the operation of the line-follower. Figure 18 shows the RTL for half of this controller, to process only one of the bitstreams, as the other half is identical.

Figure 19 shows the photo of the prototype. The FPGA board is on top of the line-follower platform, and provides power to the analog converter circuit placed on top of the

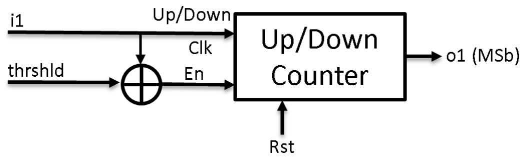

Fig. 18. RTL of the controller II.

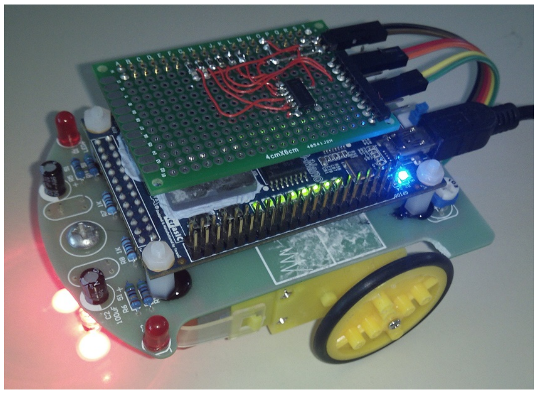

Fig. 19. Photo of the prototype for the demonstration system.

robotic platform. The analog converter circuit establishes the connections between the FPGA board and the platform. The complete stochastic system occupies 38 Logic Cells.

VI. Conclusions and Future Work

This work introduces novel components to support and process stochastic bitstreams on FPGAs, along with a small stochastic computing-based autonomous cyber-physical system to demonstrate the proposed component architectures. Future work involves a study on the statistical properties of the proposed architectures, the performance of the system when in the presence of voltage scaling, and implementation of more complex systems.

VII. Acknowledgment

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013.

References

[1] J. von Neumann, "Probabilistic logics and synthesis of reliable organisms from unreliable components," in *Automata Studies*, C. Shannon and J. McCarthy, Eds. Princeton University Press, 1956, pp. 43–98.

[2] N. Saraf, K. Bazargan, D. Lilja, and M. Riedel, "IIR filters using stochastic arithmetic," in *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, March 2014, pp. 1–6.

[3] M. Alawad and M. Lin, "Fir filter based on stochastic computing with reconfigurable digital fabric," in *Field-Programmable Custom Computing Machines (FCCM), 2015 IEEE 23rd Annual International Symposium on*, May 2015, pp. 92–95.

[4] M. Suri, O. Bichler, D. Querlioz, G. Palma, E. Vianello, D. Vuillaume, C. Gamrat, and B. DeSalvo, "Cbram devices as binary synapses for low-power stochastic neuromorphic systems: Auditory (cochlea) and visual (retina) cognitive processing applications," in *Electron Devices Meeting (IEDM), 2012 IEEE International*, Dec 2012, pp. 10.3.1–10.3.4.

[5] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. Modha, “A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm,” in *Custom Integrated Circuits Conference (CICC), 2011 IEEE*, Sept 2011, pp. 1–4.

[6] S. Tehrani, W. Gross, and S. Mannor, “Stochastic decoding of ldpc codes,” *Communications Letters, IEEE*, vol. 10, no. 10, pp. 716–718, Oct 2006.

[7] C. Winstead and S. Howard, “A probabilistic ldpc-coded fault compensation technique for reliable nanoscale computing,” *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 56, no. 6, pp. 484–488, June 2009.

[8] M. Lin, I. Lebedev, and J. Wawrzynek, “High-throughput bayesian computing machine with reconfigurable hardware,” in *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA ’10. New York, NY, USA: ACM, 2010, pp. 73–82. [Online]. Available: http://doi.acm.org/10.1145/1723112.1723127

[9] B. Gaines, “Stochastic computing systems,” A. in Information Systems Science, Ed., vol. 2, 1965, p. 37.

[10] Y.-N. Chang and K. Parhi, “Architectures for digital filters using stochastic computing,” in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, May 2013, pp. 2697–2701.

[11] H. Li, D. Zhang, and S. Foo, “A stochastic digital implementation of a neural network controller for small wind turbine systems,” *Power Electronics, IEEE Transactions on*, vol. 21, no. 5, pp. 1502–1507, Sept 2006.

[12] F. Zhou, J. Liu, Y. Yu, X. Tian, H. Liu, Y. Hao, S. Zhang, W. Chen, J. Dai, and X. Zheng, “Field-programmable gate array implementation of a probabilistic neural network for motor cortical decoding in rats,” *Journal of Neuroscience Methods*, vol. 185, no. 2, pp. 299 – 306, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165027009005366

[13] B. Brown and H. Card, “Stochastic neural computation. i. computational elements,” *Computers, IEEE Transactions on*, vol. 50, no. 9, pp. 891–905, Sep 2001.

[14] A. Alaghi and J. P. Hayes, “Survey of stochastic computing,” *ACM Trans. Embed. Comput. Syst.*, vol. 12, no. 2s, pp. 92:1–92:19, May 2013. [Online]. Available: http://doi.acm.org/10.1145/2465787.2465794

[15] R. P. Duarte, M. Vestias, and H. Neto, “Enhancing stochastic computations via process variation,” in *Field Programmable Logic and Applications (FPL), 2015 25th International Conference on*, Aug 2015, pp. 519–522.

[16] W. Qian, X. Li, M. Riedel, K. Bazargan, and D. Lilja, “An architecture for fault-tolerant computation with stochastic logic,” *Computers, IEEE Transactions on*, vol. 60, no. 1, pp. 93–105, Jan 2011.