Rapid Prototyping of Approximate Signal Processing Using Stochastic Processors on FPGAs

Rui Policarpo Duarte*, Mário Véstias† and Horácio Neto*
*INESC-ID, Instituto Superior Tecnico, Portugal
Email: rpd, hcn@inesc-id.pt
†INESC-ID, ISEL, Portugal
Email: mvestias@deetc.isel.pt

***Abstract*—In the context of IoT, silicon devices are employed to perform intense signal processing computations. They tend to operate under heavy constraints for power and area, traditional computing paradigms struggle to exhibit high levels of availability and fault-tolerance. Stochastic Computing is a computing paradigm that provides fast and compact implementations of arithmetic operations, with high levels of parallelism, and graceful degradation. Stochastic Computing is based on the computation of pseudo-random sequences of bits, hence requiring only a single bit per signal. Because of that, FPGAs are often adopted to implement such systems. This work presents a rapid prototyping framework that takes a high-level specification from the user and creates a complete Stochastic Computing systems capable of interfacing analog sensors directly on the FPGA, and perform signal processing computations over the stochastic bitstreams.**

***Index Terms*—Approximate Computing, Fault-Tolerance, Stochastic Computing, Stochastic Bitstreams, FPGA, Prototyping Framework, IoT.**

I. Introduction

Stochastic arithmetic has emerged as a computing paradigm that provides approximate computations requiring less hardware, towards a solution with simpler but massively parallel components, trading off precision for computation time [1].

Applications such as neuromorphic systems [2], Bayesian inference [3], digital filters [4], [5], and cyber-physical systems [6] are characterized for their regularity in their datapath. Their computations are mainly based on multiple multiplications followed by accumulations. Moreover, many of these applications do not require exact results and can tolerate some deviations in their computations.

Insofar, the majority of research conducted on Stochastic Computing (SC) is confined to a set of applications, which are highly customized, and difficult to extend its adoption. Furthermore, the benefits of SC are not always clear due to the resources of the supporting elements and the clock latency to process long bitstreams. Often, the benefit of SC is shadowed by the latency and resources required to interface a traditional computing systems.

The main challenge addressed in this paper is to ease the definition and evaluation of a Stochastic Processor (SP) to compute mathematical expressions as alternative to other more time consuming and prone-to-error design approaches, and without having to delve into the technicalities of High-Level Synthesis (HLS).

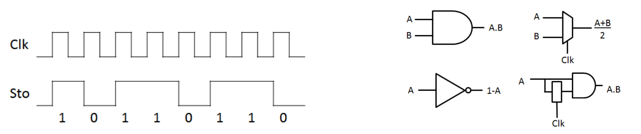

Fig. 1. Example of a stochastic bitstream encoding 0.625 and 0.25 in unipolar and bipolar encodings, respectively (left); and architecture of basic stochastic arithmetic units: multiply, sum, negative and square (right).

The main contribution of this work is a highly customizable and scalable framework that given a datapath's specification, it generates the corresponding SP, and its supporting blocks, targeting Field-Programmable Gate Arrays (FPGAs). This work is intended to facilitate automated architectural changes via unified and regular interfaces, and design-space exploration often sought in research due to the long execution times. Moreover, this work provides an early assessment of resources, power and performance metrics. This enables the usage of the SC in stand-alone stochastic systems or accelerators for heterogeneous and System-on-a-Chip (SoC) platforms.

By definition, a stochastic signal is the result of a continuous-time stochastic process which produces two values: 0 and 1. According to [1], a stochastic bitstream is a sequence of stochastic signals over time whose value is within $[0;1]$ for unipolar, or within $[-1;1]$ for bipolar. It is defined as the number of ones (1) over the total number of bits. On stochastic bitstreams there are no weights in the representation, as in typical binary-radix representation, thus all bits have the same contribution for the encoded value. For example, the same sequence of 8 bits 10110110 represents $5/8 = 0.625$ in unipolar and $2 * (5/8 - 0.5) = 0.25$ in bipolar. Stochastic bitstreams intrinsically provide graceful degradation as the impact of bit-flips on the bitstream, regardless their position on the bitstream, is the same as the least significant bit, in binary-radix.

Fig. 1 illustrates the aforementioned stochastic bitstream. On top there's the clock signal, to ensure synchronism; and on bottom the encoded value.

To perform arithmetic computations several stochastic arithmetic units have been proposed, including an adder, and a multiplier, as illustrated in Fig. 1.More details on stochastic arithmetic units can be found in the survey presented in [7] which covers the most common arithmetic units.

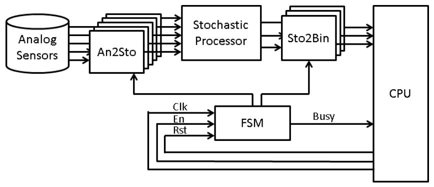

Fig. 2. Top-level architecture of the circuit design to test the Stochastic Processor, including the interface and support units.

II. Rapid Prototyping Framework

Like any datapath for signal processing, SP implements a chain of computations, but over a bitstream. This work proposes a novel method to specify it as a mathematical expression, defined as a list of operands and operators in a data structure. These mathematical expressions can be variable in size and type of operations. The data structure is organized as a regular tree of computations which maintains the data dependencies in the datapath. From this data structure it is possible to identify the requirements for a system. Essentially, the framework recognizes the different operands and operators for a datapath, and then generates the corresponding circuit description in Very High Speed Integrated Circuits (VHSIC) Hardware Description Language (VHDL).

Considering the following motivational example of a function to be implemented: $f = \frac{1}{N}\left(in_0 \times in_1 + in_2 \times in_3 \times in_4\right)$ has the corresponding internal representation:

```
[['in0','in1','*'],['in2','in3','in4','*'],'+']
```

In this example, variables `in0` to `in4` are the inputs, or operands, of the datapath, and `f` the output. The operators are + and $\star$, for addition and multiplication, respectively. The inputs and outputs of the SP correspond to the number of variables and are automatically determined.

The generated SP was planned to be autonomous or part of a larger system, as illustrated in Fig. 2. The SP is in the middle and the rest of the circuit is formed by the supporting units to do the computations. The system is interfaced via the input and output bitstreams, and also the Finite State Machine (FSM)'s control signals, namely *Clk*, *Enable* and *Reset*. In particular, the FSM is responsible for the generation of the control signals for all units in the design. It also controls the burn-in period to compensate the clock cycles required any FSM-based stochastic arithmetic units present.

The conversion between binary-radix and stochastic bitstreams is the major limitation in interfacing typical digital systems or analog I/Os. Connecting the SP from the rest of the supporting elements allows to integrate it in other systems, capable of interfacing with stochastic bitstreams, such as [6]. In this work cyber-physical system interfaces analog sensors and actuators without the need to have either analog to digital and binary-to-stochastic converters to acquire input data; and neither stochastic-to-binary and digital to analog converters to drive the actuators.

In essence, performing signal processing computations on a bitstream from a binary-radix value requires more resources than an analog interface, but the analog interface requires a dedicated input pin.

To improve the statistical quality of the stochastic bitstreams on the datapath, this work adopts the Self-Timed Ring-Oscillator (STRO) proposed in [8].

A. Test Platform

The proposed framework provides a test platform to incorporate any SP generated on an FPGA. This test platform manages the input and output signals required by the SP, along with the required conversions to be accessed by the host computer, or analog sensors and actuators.

The test platform circuit is constituted by the circuit under test (i.e. a simple arithmetic unit or a SP), the bitstream generators, and the output calculators.

The framework relies on Quartus, from Altera, to synthesize the test circuit and to produce a configuration bitstream, along with resource consumption, and timing and power estimates.

III. Conclusions and Future Work

This work introduces an open-source framework easy implementation of SC systems for signal processing. It receives a high-level specification for a datapath and produces a synthesizable description of a signal processing system for SP on an FPGA. The framework is available at https://fenix.tecnico.ulisboa.pt/homepage/ist14551/stochastic, under an open-source license.

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] B. R. Gaines, "Techniques of identification with the stochastic computer," in *in "Proc. International Federation of Automatic Control Symposium on Identification, Progue*, 1967.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] R. P. Duarte, M. Vestias, and H. Neto, "Xtokaxtikox: A stochastic computing-based autonomous cyber-physical system," in *Proceedings of the 1st IEEE International Conference on Rebooting Computing (ICRC)*, 2016.

[7] 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

[8] 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.