Synthesis of Bayesian Machines On FPGAs Using Stochastic Arithmetic

Rui Policarpo Duarte, Jorge Lobo, João Filipe Ferreira and Jorge Dias
Institute of Systems and Robotics
DEEC - University of Coimbra - Polo II
3030-290 Coimbra, Portugal
Email: {rduarte, jlobo, jfilipe, jorge}@isr.uc.pt

***Abstract*—Probabilistic inference allows artificial systems to cope with uncertainty, but it can be computationally demanding. Inspired by biological neural systems, stochastic arithmetic modules on reconfigurable hardware can provide massively parallel systems with limited resources. This work presents a framework to automatically implement Bayesian Machines to perform computations using stochastic bitstreams.**

I. Introduction

Biological neural systems excel in robustness and power-efficient operation, despite relying on low-precision, unreliable and massively parallel neural elements. However, they have highly reconfigurable and plastic connections, capable of self-organising driven by a template based architecture [1].

Probabilistic modelling approaches allow artificial systems to cope with the uncertainty and incompleteness inherent to the knowledge regarding a particular phenomenon, much as the human brain does. Moreover, these models can be designed and even implemented in a hierarchical fashion [2], [3].

The Bayesian programming paradigm [3] allows the specification of Bayesian models in broad sense. Using the ProBT API [4], questions can then be "asked" to the model about the phenomenon, generating specific Bayesian Machines (BMs) implementing the computation specification corresponding to the desired probabilistic inference process. However, for many practical applications for which inference is needed, Von Neumann machines present performance, power and area bottlenecks, making them costly and inefficient. To overcome that, stochastic arithmetic has emerged as an alternative providing approximate computations requiring less hardware and energy, towards a neuromorphic solution with simpler but massively parallel components [5], trading off precision for computation time. Previous research has addressed the use of stochastic arithmetic units in neuromorphic systems [6], [7], image processing [8] and inference [9].

Combining the trade-offs of stochastic computing, between precision and computation time, with the regularity of probabilistic computations and the fine-grain parallelism from reconfigurable logic hardware (FPGAs), we have developed a framework to explore the design space and implement such Bayesian Machines (Fig. 1). In this paper, we will show how it can be used to translate a probabilistic formulation into a specification of a circuit to be implemented on an FPGA. In the following sections, we describe the fundamentals of the developed framework and its evaluation.

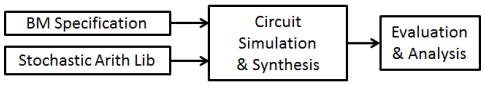

Fig. 1. Flow of the proposed framework to synthesise and evaluate probabilistic computations on FPGAs using stochastic computing.

II. From a Question to a Bayesian Model to a Circuit

A BM uses probability distributions to perform inference: its inputs consist of distributions that represent *soft evidence* concerning random variables corresponding to observations, which are then processed in a chain of computations that include distributions encoding information about the modelled phenomenon, and which outputs *soft evidence* concerning the unknown random variables of interest.

Consider a case-study example of a BM generated from a joint probability distribution on a set of discrete and finite variables: $P(M \wedge D \wedge L)$, where M, D and L are themselves conjunctions of variables, e.g. $D = D_1 \wedge \ldots \wedge D_k$. Additionally, consider that soft evidence is defined for variables D_k as probability distributions $\tilde{P}(D_k)$, to be used as inputs for the BM, and that we wish to infer $P'(M)$. Let us now imagine that the model is specified using the Bayesian Programming paradigm. ProBT can then be used to produce an internal simplification of a question to a model which minimises the computational load by reducing the number of sums according to the structure of the joint distribution. The algorithm used by ProBT to produce this simplification is the SRA [4], an algorithm similar to the sum-product algorithm. For example, assuming the joint distribution is described as $P(M \wedge D_1 \wedge D_2) = P(M)P(D_1 \mid M)P(D_2 \mid M)$, the SRA will transform the question to the Bayesian model into:

$$P'(M) = \frac{1}{Z} P(M)(\sum_{D_1} \tilde{P}(D_1)P(D_1 \mid M))(\sum_{D_2} \tilde{P}(D_2)P(D_2 \mid M)) \quad (1)$$

Since discrete variables are being used, and given that the aforementioned computation relies on a regular set of sums and multiplications, this can be efficiently implemented by exploiting the parallelism offered by the FPGA. Figure 2 shows the RTL for the synthesised VHDL given the specification for the aforementioned problem.

III. Stochastic Arithmetic Units on FPGAs

Previous work on stochastic computing involving bit streams has been presented in [10] and [11]. Based on these

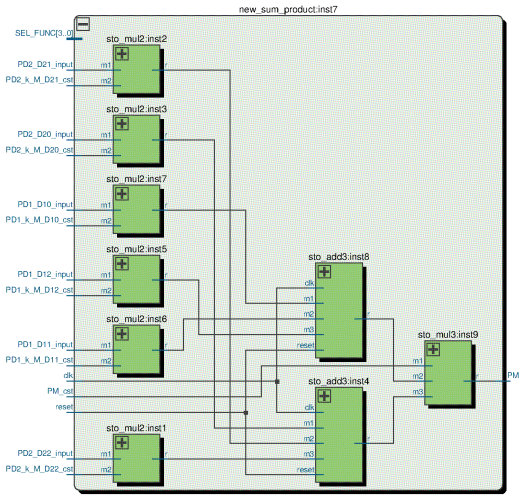

Fig. 2. Block Diagram of the Bayesian Machine, instantiating stochastic adders and multipliers.

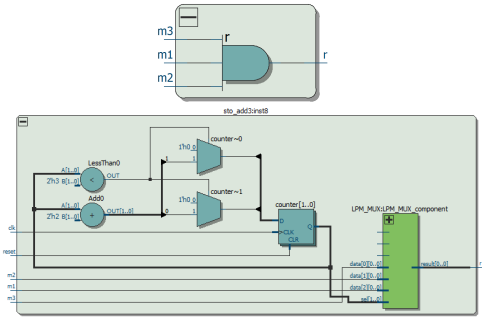

Fig. 3. Block Diagram of the stochastic multiplier (top) and adder (bottom) with 3 inputs.

architectures, and in the fact that the aforementioned probabilistic problems are based on addition and multiplication, the proposed framework creates the set of stochastic arithmetic units required by the problem under consideration. They are then instantiated by the Bayesian Machine design.

The implementation of the stochastic arithmetic units targeting FPGAs, offers implementation of computations on bit streams, requiring less resources. Furthermore, they also allow to parallelize computations, which contributes to decrease the computing time, and increase the efficiency of total number of computations per device per second. Figure 3 shows the block diagram, or RTL, for a 3-input stochastic multiplier and adder. The stochastic multiplication corresponds to the AND of all stochastic inputs. Addition is obtained via a MUX of the stochastic inputs, selected using a cyclic counter.

The proposed framework includes a test platform to test any BM generated by it, using the stochastic arithmetic units. This platform generates all the stimulus signals required by the circuit under test, and produces conversion of the results to be read. This test platform is described in VHDL, and is synthesized to configure a Cyclone IV FPGA, from Altera, present on the DE2-115 board, from Terasic. It can be easily adapted to any other reconfigurable platforms.

Figure 4 depicts the block diagram of the test circuit, used

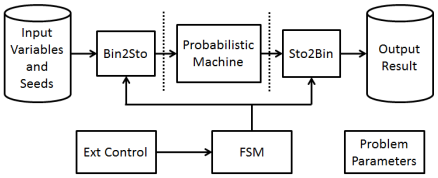

Fig. 4. Top level architecture of the circuit design to test the Bayesian Machines, including the supporting units.

to test the BMs. The units for the generation of the stochastic inputs (bin2sto) from binary values stored in memory. The result converter (sto2bin) and its storage. The Finite State Machine (FSM) controls the test process.

IV. Conclusions and Future Work

This contribution presents a generic framework to implement and test Bayesian Machines to compute questions to Bayesian models. Future work involves analysis and modelling of the results through mathematical expressions as well as performing a complementary study on the tradeoffs between the size of the bitstream, errors, processing time and resources.

Acknowledgment

This publication has been partially supported by the European Commission collaborative FET project BAMBI - Bottom-up Approaches to Machines dedicated to Bayesian Inference - FP7-ICT-2013-C, project number 618024 (www.bambi-fet.eu).

References

[1] R. Vogelstein, U. Mallik, J. Vogelstein, and G. Cauwenberghs, "Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses," *Neural Networks, IEEE Transactions on*, vol. 18, no. 1, pp. 253–265, Jan 2007.

[2] T. S. Lee and D. Mumford, "Hierarchical bayesian inference in the visual cortex," 2002.

[3] P. Bessière, E. Mazer, K. Mekhnacha, and J. M. Ahuactzin, *Bayesian Programming*. Chapman & Hall/CRC Press, Dec. 2013. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00905797

[4] K. Mekhnacha, J.-M. Ahuactzin, P. Bessière, E. Mazer, and L. Smail, "Exact and approximate inference in ProBT," *Revue d'Intelligence Artificielle*, vol. 21/3, pp. 295–332, 2007. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00338763

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

[6] 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.

[7] N. L. Zhang and D. Poole, "Exploiting causal independence in bayesian network inference," *Journal of Artificial Intelligence Research*, vol. 5, pp. 301–328, 1996.

[8] 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.

[9] J. B. Tenenbaum, E. M. Jonas, and V. K. Mansinghka, "Stochastic digital circuits for probabilistic inference," Massachusetts Institute of Technology, Tech. Rep., November 2008. [Online]. Available: http://hdl.handle.net/1721.1/43712

[10] B. R. Gaines, "Stochastic computing," in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, ser. AFIPS '67 (Spring). New York, NY, USA: ACM, 1967, pp. 149–156. [Online]. Available: http://doi.acm.org/10.1145/1465482.1465505

[11] 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