Hardware-Level Bayesian Inference

Alexantrou Serb, Edoardo Manino, Ioannis Messaris,
Long Tran-Thanh and Themistoklis Prodromakis
University of Southampton
Highfield campus, Southampton, United Kingdom
`A.Serb@soton.ac.uk`

Abstract

Brain-inspired, inherently parallel computation has been proven to excel at tasks where the intrinsically serial Von Neumann architecture struggles. This has led to vast efforts aimed towards developing bio-inspired electronics, most notably in the guise of artificial neural networks (ANNs). However, ANNs are simply one possible substrate upon which computation can be carried out; their configuration determining what sort of computational function is being performed. In this work we show how Bayesian inference, a fundamental computational function, can be carried out using arrays of memristive devices, demonstrating computation directly using probability distributions as inputs and outputs. Our approach bypasses the need to map the Bayesian computation on an ANN (or any other) substrate since computation is carried out by simply providing the input distributions and letting Ohm's law converge the voltages within the system to the correct answer. We show the fundamental circuit blocks used to enable this style of computation, examine how memristor non-idealities affect the quality of computation and exemplify a 'Bayesian learning machine' performing a simple task with no need for any digital arithmetic-logic operations.

1 Introduction

Modern electronics has rested on the pillar of the Von Neumann architecture for over 50 years. However, the ability of the human brain to effortlessly carry out tasks that overtax even powerful supercomputers (e.g. object recognition) has led to intense research in alternative architectures. One such approach is deep learning (LeCun et al., 2015), which draws inspiration from biological neural networks. The promise of massive parallelism (Krizhevsky et al., 2012) and the elusive memory/computation collocation that eliminates the Von Neumann bottleneck (McKee, 2004) have been great driving forces for the field, eventually leading to the development of on-chip neural network hardware (Indiveri et al., 2011; Merolla et al., 2014). Another approach is performing computing directly in the probabilistic domain using Bayesian inference and computing on distributions rather than on numbers. Arguably, Bayesian inference is one of the key functions of biological circuits (Legenstein and Maass, 2014), therefore developing hardware specifically tailored to this purpose -even if not in the guise of artificial neural networks- may offer great advantages vs. complex, power- and area-hungry implementations of the same task (processor-, or FPGA-based solutions (Murray, 2013; Marsono et al., 2008)).

The core concept of Bayesian inference is to utilise our belief of correctness of different hypotheses within the calculations. To do so, we capture this belief as a distribution over the set of possible hypotheses, and we update it as more evidences and information are gathered. In particular, we start with a prior distribution, which captures our existing background knowledge and/or subjective belief over this set of hypotheses. We then use Bayes' rule to calculate the posterior distribution (i.e., update our belief). That is, we calculate the probability of a particular hypothesis to be correct, based on the newly collected information or evidence. A key challenge of Bayesian inference is

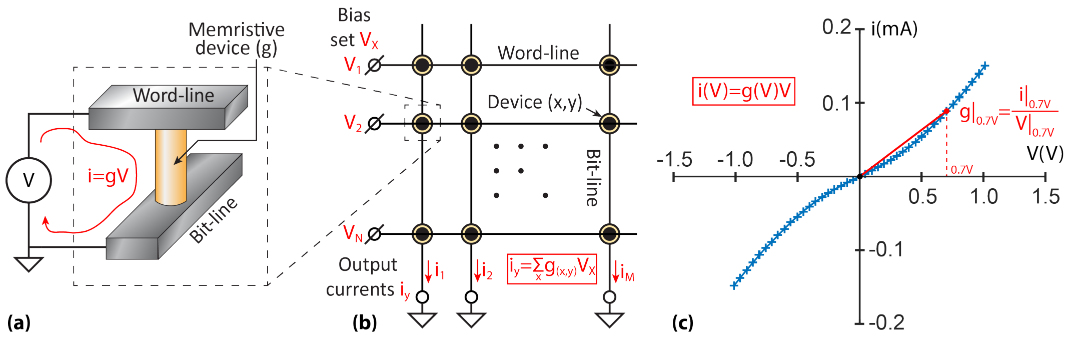


Figure 1: Fundamental memristor operation. (a) Memristor device diagram showing top (word-line) and bottom (bit-line) electrodes sandwiching the device's active material. Multiplier action is achieved through Ohm's law: $i = g \cdot V$. (b) The crossbar topology. Typically voltages are applied at the word-lines whilst the bit-lines are virtually earthed and currents through them measured. As a result a crossbar is capable of carrying out multiply-accumulate operations in parallel (see equation in inset). (c) Typical, measure current-voltage (IV) characteristic of a memristive device. Due to non-linearities the conductance of the device shows a dependence on applied voltage (also see Eq. **??**). Red lines show an example where conductance is calculated at $0.7\,V$.

the computational complexity of the belief update, While many techniques have been proposed to speed up the computation of Bayesian updating (by using, e.g., conjugate priors, variational Bayes, or approximate Bayesian computation), these are software-based, and thus, they are still disadvantageous, compared to other hardware-based computational solutions, such as neural-net-on-the-chip systems. This disadvantage in fact prevents the efficient usage of Bayesian inference in many real-world applications, where computational cost is a key performance indicator (e.g., larges-scale networks or embedded systems).

Against this background, this paper aims to lay down the theoretical concept of building hardware-based Bayesian inference. In particular, we focus on the description of the Bayesian inference at the hardware level. Note that at this level, any practical implementation of computation on distributions will necessarily involve a discretisation of the distributions involved and heavy reliance on matrix multiplication in order to carry out key probabilistic computing operations such as inference and marginalisation. This renders the quickly maturing technology of memristors (Waser and Aono, 2007) particularly suitable for the hardware implementation of the proposed Bayesian machine. Memristors are simple, extremely scalable (Khiat et al., 2016), low-power (Pickett and Williams, 2012), two-terminal devices that can be manufactured using a variety of fabrication techniques (Yang et al., 2013) but ultimately all act as non-linear, electrically programmable resistors (Chua, 1971). Their *tuneable resistive states* (which can be mapped to probability values) can be used to store variables and carry out multiplication (i.e., to be *programmed*) by applying a voltage across them and measuring the resulting current in accordance to Ohm's law $i = gV$, where i is the current flowing through the memristor, g the conductance of the device and V the voltage applied across its terminals, as shown in Fig. 1(a). The memristive multipliers can be arranged in a crossbar configuration (Likharev, 2005) as shown in Fig. 1(b). In-silico Bayesian computation relies on providing a discrete distribution as an input to the wordlines of the crossbar (we will call such a collection of signals an ‘input vector’) and receiving another distribution as an ‘answer’ at the bitlines of the array. Importantly, g can itself be a function of applied voltage as illustrated in Fig. 1(c).

In what follows, we describe our paradigm in detail. Note that while we are also in the progress of building a proof-of-concept real hardware system, this paper only focusses on the theoretical foundations of the concept. As such, this paper demonstrates the paradigm's basic functions in a simulated system that estimates the probability of an asthmatic attack given air quality and heart rate measurements. In particular, we first illustrate the conceptual model of various blocks and the system as a whole using ideal, linear memristors. Then, known characteristics of metal-oxide memristors, specifically the non-linear current-voltage (IV) characteristics modelled in (Messaris et al., 2017), are included in the simulations and their effect on performance is assessed in order to demonstrate the physical implementability of the paradigm. We conclude the paper with a discussion on how to apply our paradigm to wearable device based health monitoring.

In summary, our paper extends the state of the art in the following ways: We propose a new concept to implement Bayesian inference at the hardware level, using the recent advances of (tuneable) memristors. We then demonstrate its applicability to computationally restricted embedded systems through the example of personal health monitoring using wearable devices. This concept allows us to perform efficient Bayesian inference directly at the hardware level, and thus, enables us to build low cost and fast hardware-level Bayesian machine learning modules that can compete with the current in-silico neural networks and other non-Bayesian counterparts.

2 Fundamental building blocks

Computing on distributions necessitates the development of fundamental circuit modules that operate directly on input distributions and yield outputs that may or may not be themselves distributions. These can be split into a set of 'computational modules' that enable Bayesian inference in-silico and a set of 'supporting modules' that ensure that the system operates correctly and can be interfaced in a useful manner.

2.1 Computational modules

Bayes' rule is given by:

$$p(A|B) = \frac{p(A) \cdot p(B|A)}{p(B)} = \frac{p(A,B)}{p(B)} \tag{1}$$

where $p(A)$ is the prior probability of hypothesis A, $p(B)$ is the probability of evidence B, $p(A|B)$ is the posterior probability of A given evidence B, $p(B|A)$ is the likelihood function of B given A is correct, and $p(A,B)$ is the joint probability of A and B. Notably, $p(A)$ and $p(B)$ are distributions over a single random variable and can be discretised into a vector, whilst $p(A|B), p(B|A)$, and $p(A,B)$ are distributions over two variables and need to be discretised into matrices.

A very common implementation of Bayes' rule occurs when $p(A)$ and $p(B|A)$ are known and we seek to infer the distribution of $p(B)$. This can be easily achieved in a fully parallel fashion if $p(A)$ is given as a voltage input to a crossbar array that stores $p(B|A)$ in the conductance of its memristors (units of S). The current flowing through each device is then proportional to $p(A) \cdot p(B|A) = p(A,B)$, which in turn is marginalised by the summing action of the crossbar bitlines as shown in Fig. 2(a). Notably, extracting the values of the joint distribution from the memristive matrix is simply a matter of feeding the input distribution one word-line at a time, keeping all other wordlines grounded in order to avoid the marginalisation effect. The same principle of operation can be applied for other operations too: A vector array ($N \times 1$) consisting of independent memristive devices may be used to implement element-wise multiplication. Distribution $p(A)$ is fed into the array as voltage and the multiplier is stored in the condictances of the memristors g_k; thus the multiplier computes $p(A) \cdot g_k$. If all g_k values are equal scalar multiplication is obtained. This architecture is shown in Fig. 2(b). Notably, other implementations are possible where the memristors store variables of interest in their resistance, as opposed to conductance. In these cases the memristive arrays act as dividers rather than multipliers.

Another typical implementation is the update of the belief distribution through Bayes' rule. In particular, Eq. (1) can be reformulated as follows:

$$p(A|B) \propto p(A) \cdot p(B|A) \tag{2}$$

This can easily calculated by using the distribution multiplier hardware module (Fig. 2(b)). To make sure that the sum of $p(A|B)$ over all possible A is always 1, we apply the normaliser module (see Sec. 2.2 for more details). A concrete example of Bayesian belief update will be described in Sec. 3.

All computational blocks transform input voltages into currents, which means that a method for transforming the output currents to voltages is required. This can be achieved using a bank of trans-impedance amplifier and inverting buffer cascades as shown in Fig. 2(c). The transimpedance amplifiers maintain the virtual earthing of the bitlines while converting their currents into voltages. These in turn are inverted by the inverting buffers. Depending on the specifics of the system implementation the inversion stage may not be necessary; memristive crossbars can operate by mapping probabilities to negative voltages as well, in which case the direction of the bit-line output currents is simply reversed.

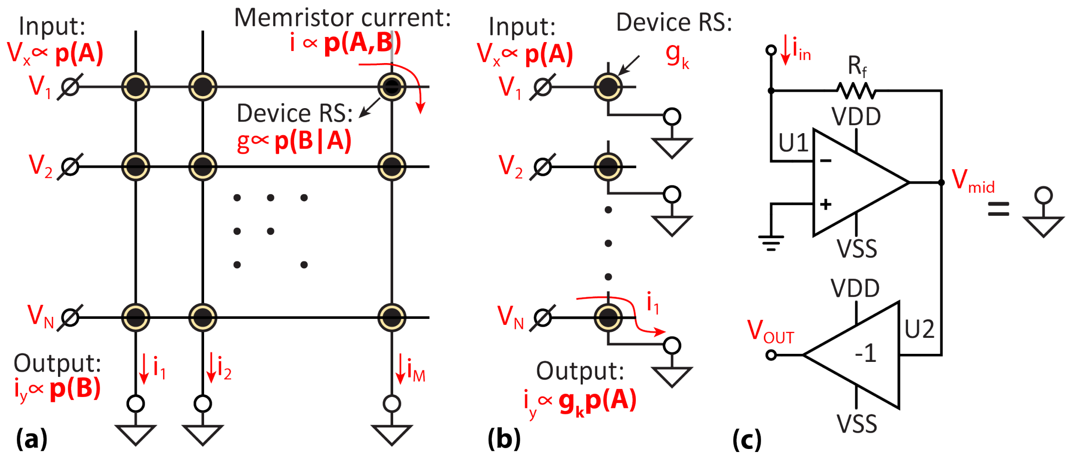


Figure 2: Core modules. (a) Memristive crossbar for performing inference. A discrete distribution is presented at the word-lines as a voltage, memristor resistive state (RS) is set such that device conductance is proportional to the conditional probability $p(B|A)$ and all bit-lines are virtually earthed. As a result, the current through each memristor is proportional to $p(B|A) \cdot p(A) = p(A, B)$, which is marginalised to $p(B)$ as a result of the current summing action of the crossbar bit-lines. (b) A distribution multiplier. Input voltages are multiplied element-wise by memristor conductances g_k (which may be common to all devices in each multiplier for scalar multiplication) and exit as currents. (c) Transimpledance amplifier (TIA) and inverting buffer cascade. The transimpedance configuration converts i_{in} to V_{mid} via: $V_{mid} = i_{in} \cdot R_f$ whilst the inverting buffer enforces $V_{OUT} = -V_{mid}$. R_f may be a memristor.

2.2 Supporting modules

Probability distributions of random variables must sum up to 1. In a physical analogue system, however, noise, mismatch and other sources of imprecision may mean that input vectors don't necessarily satisfy that condition. For that reason one of the most important supporting modules is the 'normaliser' circuit. The normaliser receives an input vector and applies it on a set of M, binary-encoded, N-input scalar multipliers as shown in Fig. 2(b) forming a binary encoder of depth M as shown in Fig. 3(a). If multiplication factors $g_0, g_1, ..., g_{M-1}$ are set as $g_x = 2^x$, then 'multiple vectors' $p(A), 2{\cdot}p(A), 4{\cdot}p(A), ...$ (or similarly $\frac{p(A)}{2}, \frac{p(A)}{4}, ...$) are generated. Summing the individual elements of any of those vectors (or indeed across any combination of vectors) is then simply a matter of summing the currents through all relevant memristors. This can be achieved using a single operational amplifier per normaliser (U1 in Fig. 3(a)), which also converts the sum of currents into a voltage. The resulting voltage can then be easily compared against a reference level (V_{ref}) in order to determine whether the chosen combination of multiple vectors sums up to more or less than the equivalent of 'probability 1'. A simple state machine searching through partial vector combinations can then determine the optimal combination, i.e. the one that transforms input vector $p(A)$ into a valid distribution to the best approximation. This is essentially a successive approximation register (SAR) technique as used in standard SAR analogue-to-digital converters (Sauerbrey et al., 2003). Finally, the correct combination of input $p(A)$ multiples is summed up into an output current vector i_{amp} by a bank of N amplifiers, which can thereafter be provided as an input to any of the computational blocks described above.

Another important supporting block is the 'maximum finder', which receives an input vector and outputs a digital flag indicating the highest probability density sample in the input vector. This is achieved by a simple, multiple input comparator as illustrated in Fig. 3(d). The transistor with the highest gate voltage will provide most of the tail current because a higher gate voltage translates directly into a higher gate-source voltage. The resistive elements (themselves potentially memristors) resistive state settings can then combine with a digital threshold in such way as to only trigger a digital flag indicating the location of the maximum only when $> \frac{1}{2}$ of the tail current is provided by a single transistor, indicating that a clear maximum exists in the distribution. Alternatively if the tail current threshold is set to just below $\frac{1}{2}$ the system will output a single digital flag if there is a clear maximum and two digital flags if two points compete for the maximum very closely, but will

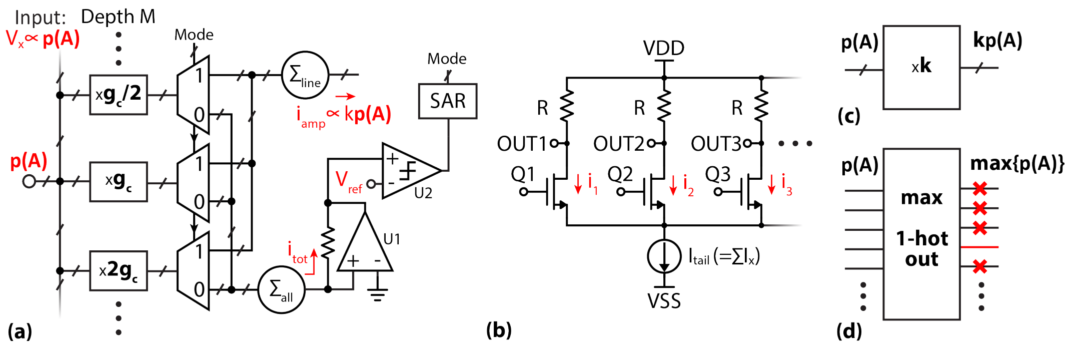


Figure 3: Supporting modules. (a) Normaliser circuit. The input distribution is sent to multiplier blocks in parallel and then the combination of multiplers that transforms the input to a valid probability distribution is selected using a successive approximation register techinque. Circuit lines cut by oblique bars denote buses (many wires in parallel). (b) Maximum finder circuit. The input distribution is applied at the gates of transistors Q1-N, which then compete for supplying the current I_{tail}. The current supplied by each transistor then causes the voltage at the corresponding OUT node to drop to $VDD - i_x \cdot R$. A simple voltage comparator can then determine whether $i_x > \frac{I_{tail}}{2}$ at every current branch, i.e. if $OUTx < VDD - \frac{R \cdot I_{tail}}{2}$. Note: tail current sinks into negative power supply VSS and not ground. (c,d) Circuit diagrams for multipler and max. finder respectively. Max. finder diagram illustrates 1-hot operation.

not output anything if 3 or more points are joint maxima. Notably, because only the location of the maximum is sought by the maximum finder, the input vector need not be a valid distribution.

3 Bayesian inference example

In this section we show how the techniques introduced before can be used to tackle real-world problems. Inspired by the work on wearable devices for continuous health monitoring (Dieffenderfer et al., 2016), we design a small graphical model for the prediction of potential health issues (see Figure 4(a)). This model uses the data coming from sensors to monitor the air quality as $A \in \{bad, medium, good\}$, and the heart beat to assess the current activity of the patient $B \in \{resting, exercising\}$. Once combined, the two are used to predict whether a crisis is probable or not, and thus it is necessary to warn the patient. More formally, we can predict $C \in \{safe, crisis\}$ with a classic Naïve Bayes classifier:

$$C^* = \max_C\{p(C|A,B)\} \qquad \text{where} \qquad p(C|A,B) \propto p(A|C)p(B|C)p(C) \tag{3}$$

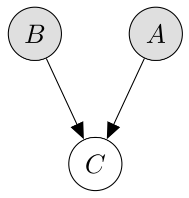
where C^* is also known as the maximum-a-posteriori estimate.

This system can be implemented efficiently using the modules introduced before (see Figure 4(b)). First, we have a crossbar that stores $p(A,C) = p(A|C)p(C)$, receiving air quality level A as input and outputting the crisis level prediction C before factoring heart rate B in. Then, we feed the output in parallel to two array of memristors that store $p(B = resting|C)$ and $p(B = exercising|C)$ respectively. Depending on the actual value of the heart beat B, we choose only one of the two outputs and we feed it into the normaliser to compute $p(C|A,B)$. Finally, we employ the max-finder module to produce the estimate C^*. Interposed between all of the four modules we have a current-to-voltage converter (not shown in the figure).

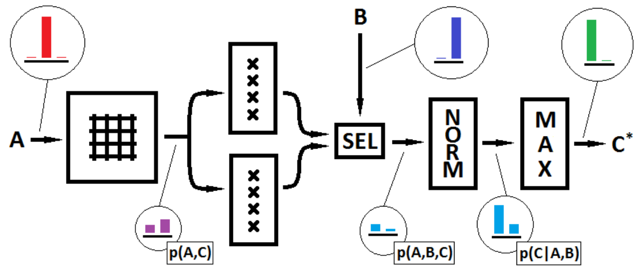
Notice how this approach can scale well to more complicated graphical models, as it simply consists of a cascade of small modules.

4 Discussion

The proposed system architecture was designed to allow in-silico implementation of Bayesian processing in a massively parallel and energy efficient way. We have argued that during normal operation inference can be carried out by making direct use of the multiply-accumulate capabilities of the crossbar configuration. Notably, this requires no resistive switching from the devices; rather



(a)



(b)

Figure 4: Bayesian machine example, (a) graphical model, (b) implementation. See text for interpretation of variables.

the computation is carried out in the analogue domain in the form of settling the crossbar/TIA system under the corresponding RC constant. The extent to which spending the energy required to settle that RC constant is competitive against standard digital number-crunching approach will ultimately be determined at the circuit layout level and will largely depend on the minimum achievable size for the memristive crossbar array and the specifications of the TIAs involved in maintaining virtual earthing. This situation changes, however, as soon as some sort of learning is implemented. In this case, the conditional probability matrix might need to be updated as new information arrives, thus requiring the devices in the corresponding crossbar to be periodically programmed. The overall energy, speed and circuit complexity cost associated with these updates will depend on the intrinsic energy efficiency of memristor switching (see (Pickett and Williams, 2012)) and the stability/predictability of the devices (i.e., how many attempts do we generally need to make before the memristors reach the desired state (Serb et al., 2015)). Note that a similar problem can be found in Flash memories (Suh et al., 1995). The cost of programming devices will affect the computation of reciprocal probabilities $p(A)^{-1}$ too in a similar fashion. The normaliser, on the other hand, can be expected to consume similar power to a crossbar used for passive inference (for binary encoder depth similar to the number of discrete distribution samples - $M \approx N$) as similar numbers of devices and TIAs are involved in both cases.

Practical implementation of the proposed analogue building blocks will imply the presence of noise, mismatch, and other imperfection factors. Whilst the normaliser circuit is designed to cope with the gradual degradation of probability distributions as they propagate through the machine, a linear mapping between probability and voltage levels or memristor resistive states may lead to limited probability dynamic range, i.e. relatively very small probability values may be mapped to voltages below the noise levels in the system. Notably, if all probabilities in a distribution are very low, but similar, normalisers/multipliers can be used to scale as appropriately; only the coexistence of very large probability values with very low ones in the same distribution may become problematic. To overcome this issue, we are investigating the option of mapping resistive state/voltage to log probability instead or using non-uniform multipliers to pre-distort probability distributions.

Another point worth to mention is that the presence of TIA blocks naturally leads to a dual supply rail implementation. In the simplest approach probabilities are represented by positive voltages covering much of (but not the entire) positive voltage range $[GND, VDD]$ whilst the TIAs use the negative part of the range $[VSS, GND]$ in order to transform currents into voltages. In the positive range, the input voltage range of the maximum finder circuit from Fig. 3(d) means that probabilities can be safely mapped only in the region falling within $[GND, VDD - \frac{I_{tail} \cdot R}{2} - V_{gs,max}]$, where $V_{gs,max}$ is the maximum gate-source voltage of transistors Qx for drain current equal to I_{tail}. This may be improved by replacing the R elements with current mirrors in more optimised implementations. However, the detailed investigation of this solution remains as future work.

Finally, another key module that would prove of great use to Bayesian computing would be a hardware random number generator capable of generating numbers from a discrete distribution $p(A)$ provided as voltages at its inputs. This is currently under development.

References

Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

S. A. McKee, “Reflections on the memory wall,” in *Proceedings of the 1st conference on Computing frontiers*. ACM, 2004, p. 162.

G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. Van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud *et al.*, “Neuromorphic silicon neuron circuits,” *Frontiers in neuroscience*, vol. 5, 2011.

P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

R. Legenstein and W. Maass, “Ensembles of spiking neurons with noise support optimal probabilistic inference in a dynamically changing environment,” *PLoS computational biology*, vol. 10, no. 10, p. e1003859, 2014.

L. M. Murray, “Bayesian state-space modelling on high-performance hardware using libbi,” *arXiv preprint arXiv:1306.3277*, 2013.

M. N. Marsono, M. W. El-Kharashi, and F. Gebali, “Binary lns-based naïve bayes inference engine for spam control: noise analysis and fpga implementation,” *IET Computers & Digital Techniques*, vol. 2, no. 1, pp. 56–62, 2008.

R. Waser and M. Aono, “Nanoionics-based resistive switching memories,” *Nature materials*, vol. 6, no. 11, pp. 833–840, 2007.

A. Khiat, P. Ayliffe, and T. Prodromakis, “High density crossbar arrays with sub-15 nm single cells via liftoff process only,” *Scientific reports*, vol. 6, 2016.

M. D. Pickett and R. S. Williams, “Sub-100 fj and sub-nanosecond thermally driven threshold switching in niobium oxide crosspoint nanodevices,” *Nanotechnology*, vol. 23, no. 21, p. 215202, 2012.

J. J. Yang, D. B. Strukov, and D. R. Stewart, “Memristive devices for computing,” *Nature nanotechnology*, vol. 8, no. 1, pp. 13–24, 2013.

L. Chua, “Memristor-the missing circuit element,” *Circuit Theory, IEEE Transactions on*, vol. 18, no. 5, pp. 507–519, 1971.

K. K. Likharev, “Cmol: A silicon-based bottom-up approach to nanoelectronics,” *Interface*, vol. 14, pp. 43–45, 2005.

I. Messaris, A. Serb, A. Khiat, S. Nikolaidis, and T. Prodromakis, “A compact verilog-a reram switching model,” *arXiv preprint arXiv:1703.01167*, 2017.

J. Sauerbrey, D. Schmitt-Landsiedel, and R. Thewes, “A 0.5-v 1-/spl mu/w successive approximation adc,” *IEEE Journal of Solid-State Circuits*, vol. 38, no. 7, pp. 1261–1265, 2003.

J. Dieffenderfer, H. Goodell, S. Mills, M. McKnight, S. Yao, F. Lin, E. Beppler, B. Bent, B. Lee, V. Misra *et al.*, “Low-power wearable systems for continuous monitoring of environment and health for chronic respiratory disease,” *IEEE journal of biomedical and health informatics*, vol. 20, no. 5, pp. 1251–1264, 2016.

A. Serb, A. Khiat, and T. Prodromakis, “An rram biasing parameter optimizer,” *IEEE Transactions on Electron Devices*, vol. 62, no. 11, pp. 3685–3691, 2015.

K.-D. Suh, B.-H. Suh, Y.-H. Lim, J.-K. Kim, Y.-J. Choi, Y.-N. Koh, S.-S. Lee, S.-C. Kwon, B.-S. Choi, J.-S. Yum *et al.*, “A 3.3 v 32 mb nand flash memory with incremental step pulse programming scheme,” *IEEE Journal of Solid-State Circuits*, vol. 30, no. 11, pp. 1149–1156, 1995.

S. Larentis, F. Nardi, S. Balatti, D. C. Gilmer, and D. Ielmini, “Resistive switching by voltage-driven ion migration in bipolar RRAMPart II: Modeling,” *IEEE Transactions on Electron Devices*, vol. 59, no. 9, pp. 2468–2475, 2012.

P. R. Mickel, A. J. Lohn, B. Joon Choi, J. Joshua Yang, M. X. Zhang, M. J. Marinella, C. D. James, and R. Stanley Williams, “A physical model of switching dynamics in tantalum oxide memristive devices,” *Applied Physics Letters*, vol. 102, no. 22, 2013.