FPGA Hardware Acceleration of Monte Carlo Simulations for the Ising Model

Francisco Ortega-Zamorano, Marcelo A. Montemurro, Sergio Alejandro Cannas, José M. Jerez, and Leonardo Franco, *Senior Member, IEEE*

Abstract—A two-dimensional Ising model with nearest-neighbors ferromagnetic interactions is implemented in a Field Programmable Gate Array (FPGA) board. Extensive Monte Carlo simulations were carried out using an efficient hardware representation of individual spins and a combined global-local LFSR random number generator. Consistent results regarding the descriptive properties of magnetic systems, like energy, magnetization and susceptibility are obtained while a speed-up factor of approximately six times is achieved in comparison to previous FPGA-based published works and almost 10^4 times in comparison to a standard CPU simulation. A detailed description of the logic design used is given together with a careful analysis of the quality of the random number generator used. The obtained results confirm the potential of FPGAs for analyzing the statistical mechanics of magnetic systems.

Index Terms—Hardware implementation, LFSR random number generator, Monte Carlo simulations, Ising model

1 Introduction

In recent years several advances in microelectronics have permitted the utilization of new devices for carrying out parallel numerical simulations in order to speed up the computational times involved. Among the most used hardware devices we can mention multi-core processors [1], GPU cards [2] and FPGA boards [3]. As it is usually the case, there is no one system better than other for all situations, as the answer is very much dependent on the problem under analysis together with the circumstances of the developers in relationship to budget, programming expertise, development time, etc. [4] In this work a FPGA based implementation of the two-dimensional ferromagnetic Ising model is carried out using Monte Carlo simulations. Field programmable gate arrays (FPGA) are reconfigurable hardware devices that can be reprogrammed to implement different combinational and sequential logic created with the aim of prototyping digital circuits as they offer flexibility and speed. In recent years advances in technology have permitted to construct FPGAs with considerable large amounts of processing power and memory storage, and as a consequence they have been applied in an ever increasing range of domain, like telecommunications, robotics, pattern recognition tasks, and infrastructure monitoring, among others [5], [6], [7].

The Ising model is a paradigm of the statistical physics approach to the study of finite temperature equilibrium properties of many body systems, by reducing complex interactions to their minimal expression (for a short review see [8] and references therein). The model assigns a set of binary variables, called spins, to every site of a regular grid or lattice. Every spin represents the component of the local magnetic moment in a crystalline solid, respect to the direction of an external magnetic field. The model is completed by defining an energy function that depends on the values of all spins in the lattice. Statistical Mechanics provides then a recipe to derive from the energy function the equilibrium probability distribution of the microscopic configurations of the system (i.e., any combination of spin values), from which macroscopic properties—like, the total magnetization or mean energy—are obtained by averaging the appropriate variables.

Despite its relative simplicity, Ising-type models and their generalizations (e.g., Potts model, see [9] and references therein) are extensively used to analyze properties of a large variety of systems exhibiting cooperative phenomena, ranging from simple ferromagnetism to complex disordered materials (e.g., spin glasses) [10]. Moreover, being originally restricted to the realm of solid state physics, they have been shown to be extremely useful in other disciplines, such as soft condensed matter (e.g., soap bubbles and foam [11], [12]), biology (e.g., biological cells [13]) and neural networks [14].

The cooperative phenomena those models intend to describe are a synergistic result of the interaction among a very large (macroscopic) number of relatively simple units. Statistical Mechanics theory seeks then to describe the asymptotic behavior of averaged macroscopic quantities—like energy or magnetization—in the limit of an infinite number of units (the so called thermodynamic limit). However, in most cases it is extremely difficult to obtain such limiting behavior analytically. Thus, the usual approach is to perform numerical simulations for an increasing number of units and then extrapolate the results to the limit of

- *F. Ortega-Zamorano, J. M. Jerez, and L. Franco are with the Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, Campus de Teatinos S/N, Málaga 29071, Spain. E-mail: {fortega, jja, lfranco}@lcc.uma.es.*
- *S. A. Cannas is with the Facultad de Matemática, Astronomía y Física (IFEG-CONICET), Universidad Nacional de Córdoba, Córdoba, Argentina. E-mail: cannas@famaf.unc.edu.ar.*
- *M. A. Montemurro is with the Department of Life Sciences, University of Manchester, Manchester, U.K. E-mail: m.montemurro@manchester.ac.uk.*

Manuscript received 29 Apr. 2015; revised 4 Nov. 2015; accepted 28 Nov. 2015. Date of publication 4 Dec. 2015; date of current version 10 Aug. 2016. Recommended for acceptance by A. Gordon-Ross.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2015.2505725

infinite number of units. This raises the question about the minimum number of units needed to obtain an accurate extrapolation to the thermodynamic limit. While for the estimation of some quantities up to $\sim 10^4$ units may be enough (and attainable in a CPU code in a few hours), others may require at least $\sim 10^6$ or more units to get reliable results (see [9] and references therein). Such calculations are only achievable in reasonable time scales with the use of parallel computing. Therefore, it is of critical importance to develop and assess massively parallel implementations of statistical physics models. In that respect, the Ising model has become one of the most common benchmarks for testing novel statistical mechanics simulation algorithms and parallel computing implementations. The most simple version (namely, that in which only nearest neighbors spins interact ferromagnetically) constitutes one of the few examples of an interacting many body system for which non trivial equilibrium properties are known exactly in two spatial dimensions [15]. This is not true in general for systems with more complicated interactions. Hence, theoretical analysis is usually limited to approximated methods whose ultimate validity strongly relies on Monte Carlo numerical simulations.

One of the most common ways to simulate the behavior of Ising-type models is the Metropolis-Hastings algorithm [16], which works by generating a sequence of sample configurations of the system that converges to the equilibrium finite temperature distribution. The Metropolis-Hastings algorithm requires the use of random numbers and as such is considered a Monte Carlo type simulation [17]. Monte Carlo simulations depend strongly on the generation of pseudo random numbers and for this reason an efficient simulation normally adapts the algorithms used to the hardware utilized. In particular, in FPGA boards the standard random number generators are based on linear feedback shift registers (LFSR) schemes [18]

FPGA boards can be used as hardware accelerators systems in several domains of applications. For example in life sciences, several recent works have benefitted from the use of FPGA boards for speeding up Monte Carlo simulations [19], [20]. The use of FPGA boards to study Ising type systems is relatively recent and thus just very few works have been published so far. Among these, the approach taken by the Janus consortium is worth mentioning as they are using a cluster of FPGA boards [21], [22]. Lin et al. have studied the two-dimensional Ising model [23], and more recently Gilman have analyzed the 3-D Ising model [24]. In this work, further optimization of the parallel updating of spin blocks is achieved by combining a global 32 bit LFSR with a smaller 12-bit local LFSR to get random numbers for the individual spin updates. Based on that strategy, we developed a highly efficient FPGA implementation of the Metropolis-Hastings algorithm for Ising type models, which was checked against the exact results for the two dimensional ferromagnetic nearest-neighbor models. The present implementation shows a considerable performance improvement with respect to previous ones.

2 The Two Dimensional Ising Model

The Ising model was originally devised to represent a ferromagnetic solid. It assigns a binary variable $S_i = \pm 1$, called spin, to each site of a regular lattice in d dimensions, where i labels the site. S_i represents the component of the magnetic moment at site i respect to the direction of an external magnetic field of intensity B. In this work we have analyzed the two-dimensional Ising model with interactions between nearest neighboring spins, in a square lattice with $N = L \times L$ sites, For this case, the energy of the system (Hamiltonian) is defined as:

$$H = -J \sum_{<i,j>} S_i S_j - B \sum_{i=1}^{N} S_i, \tag{1}$$

where $<i,j>$ denotes a sum over all pairs of nearest neighbor sites of the lattice and $J > 0$ is the ferromagnetic interaction constant or also called exchange constant. In the absence of a magnetic field $B = 0$, the energy is minimized in the ferromagnetic state, i.e., when all spins take the same value. In this work we considered $J = 1$ and $B = 0$, and used periodic boundary conditions.

The spin dynamics is governed by the Metropolis-Hastings algorithm, which essentially consists in the update of the present state of a given spin according to the change in the energy ΔE produced by the flipping of the spin. The spin value is changed if its flip reduces the energy, i.e. if $\Delta E < 0$. If $\Delta E > 0$ the spin flip can be accepted, provided that a generated random number r is smaller than the Boltzmann factor

$$r < e^{-\beta \Delta E} \tag{2}$$

where $\beta = 1/k_B T$, T being the temperature and k_B is the Boltzmann constant (in this work we use $k_B = 1$). In a sequential implementation the algorithm proceeds by picking spins one by one according to some protocol (usually randomly) until all spins (on the average in the case of random picking) have been updated (an update means a single trial; the spin can remain in its previous value after the update). Once all spins have been updated, this is called a Monte Carlo Step (MCS) and constitutes the basic iteration time unit.

Parallel implementations of the Metropolis algorithm in the square lattice make use of the nearest-neighbor interactions as follows. Let S_0 the spin to be updated. Then it is easy to see that the change in energy (1) (with $B = 0$) produced by a single spin flip $S_0 \rightarrow -S_0$ is given by $\Delta E = 2\epsilon$, where

$$\epsilon = (S_1 + S_2 + S_3 + S_4) \cdot S_0, \tag{3}$$

and $S_1, \ldots, S_4$ denote the four nearest neighbors spins of S_0. The square lattice can be divided into two square sublattices with a checkerboard structure, in such a way that the nearest neighbors of any spin in a given sublattice belongs to the other (see section for details). Hence, all spins in a sublattice can be updated simultaneously without risk of concurrency issues. A MCS is then completed by one updating of the two sublattices.

In a typical simulation a sequence of random spins configurations is generated from some initial one, by successive application of the above algorithm. For a long enough sequence (measured in MCS) it can be shown that the probability distribution for spins configurations becomes

TABLE 1
Main Specifications of the Virtex-5 XC5VLX110T FPGA Related to Its Available Slice Logic

Device	Slice Registers	Slice LUTs	Bonded IOBs	Block RAM
Virtex-5 XC5VLX110T	69,120	69,120	34	148

stationary [17]. Further application of the algorithm provides a sample set of configurations over which averages of quantities of interest can be calculated. A typical quantity is the average magnetization per spin:

$$m = \langle M \rangle / N, \tag{4}$$

where $M = \sum_i S_i$ and $\langle \cdots \rangle$ stands for an average over a single equilibrated MC sequence of spins configurations. Another quantity of interest is the zero field magnetic susceptibility, which characterizes the linear response of the system to an externally applied magnetic field. The magnetic susceptibility can be computed form the fluctuations in the magnetization as follows:

$$\chi = \frac{1}{k_B T N} \left(\langle M^2 \rangle - \langle M \rangle^2 \right). \tag{5}$$

In the thermodynamic limit (infinite lattice size) when $B = 0$ and $d \leq 2$, this model undergoes a second order phase transition at a very well defined critical temperature T_c, namely, the magnetization becomes zero for $t \leq T_c$ and different from zero for $T < T_c$. Also the susceptibility diverges at $T = T_c$ as $\chi \sim |T - T_c|^{\gamma}$, where $\gamma > 0$ is a critical exponent that depends only on the dimensionality d [25].

3 FPGA

FPGAs [26] are reprogrammable silicon chips, using prebuilt logic blocks and programmable routing resources. They can be configured to implement custom hardware functionality, and in this sense, FPGAs are completely reconfigurable and can almost instantly change its behavior by recompiling a new circuitry configuration.

The board used for the current implementation is the Virtex-5 OpenSPARC Evaluation Platform (ML509). This device includes a Xilinx Virtex-5 XC5VLX110T FPGA that provides different connector devices: 2 USBports, 2 PS/2 ports, RJ-45 and RS-232 connectors, 2 Audio Inputs, 2 Audio Outputs, Video Input, Video Output, Single-Ended and Differential I/O Expansion. Table 1 shows some characteristics of the Virtex-5 XC5VLX110T FPGA, indicating its main logic resources.

All computations have been performed using fixed point arithmetic, which is the standard way to work with FPGA boards. Even if floating point operations can be codified efficiently in FPGA boards without significant additional resources [27], [28], they tend to be less efficient than fixed point arithmetic, as it is also the case for most digital circuits.

4 Implementation

The FPGA implementation parallelizes the spin updates in order to compute faster the system evolution. The actual value of the spins ± 1 are stored as Boolean values using memory blocks while the spin dynamics is implemented using groups of LUTs. For larger lattice sizes, the number of LUTs available is not enough to simulate the dynamics of the whole lattice so the process is carried out in a number of steps in which a group of spin rows is updated.

Fig. 1 shows the hardware implementation of a spin. On the left side of the figure, six different inputs and two clock signals are shown. From top to bottom, we see the first input ("Spin") that gets the spin value from the register. The following four inputs correspond to the adjacent nearest-neighbor spins of the one under consideration, and they are indicated by 'L' (left), 'T' (top), 'R' (right) and 'B' (bottom). The current spin value is taken from the grey block of registers, while the four adjacent spin values are taken from the white one (See Fig. 3). The input indicated by LFSR32 is a generic (pseudo) random signal that will be combined with a local LFSR through a XOR gate for obtaining a local random value (see details of this process in Section 5). Two clock signals are used, one corresponding to the system clock signal (Clk_A), and another with double frequency

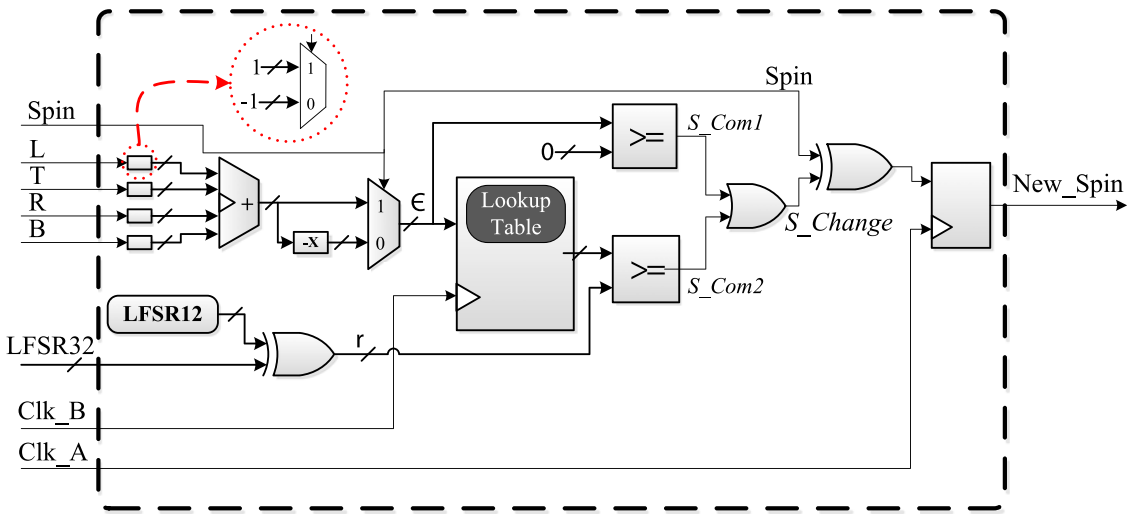

Fig. 1. Logic circuit representation used for the dynamic simulation of an Ising spin (see text for details).

TABLE 2
Logic Utilization for the Hardware Implementation of a Single Spin

Logic Utilization	Used
# Slice Registers	17
# LUTs	30

(Clk_B) obtained using a PLL in order to read the values of the Lookup table. The frequency of Clk_B is twice that of Clk_A to allow the whole updating process to be completed in one system clock cycle.

The spin updating process starts by computing the summation of the values of the four adjacent spins (this process is indicated in the left top corner of the Fig. 1), noting that first the representation of these four spins is changed from the Boolean one used in the register $(0, 1)$ to a binary one (± 1) used in the dynamics of the system (The representation modification is indicated within a dotted circle at the top left of the figure). The process continues with the calculation of the local function ϵ (see Eq. (3)), that is computed multiplexing the sum value of the four adjacent spins the current spin value.

The following step consists in the computation of the exponential of $-\beta\Delta E$, which is done using a lookup table that stores every possible values of the function $e^{-\beta\Delta E}$.

Thus, the number of entries of the look-up table is five as these are the number of possible different values of $\Delta E = (-4, -2, 0, 2, 4)$. The values of $e^{-\beta\Delta E}$ were represented using a word length of 12. The lookup table was implemented using configurable logic blocks (CLBs), in order to save RAM resources for storing spin configurations, as only five LUTs are needed.

The new value of the spin can be the same as the current one or its opposite (according to the dynamics of the model). This last case can be due to a decrease on the energy associated with the spin ($\Delta E \leq 0$) or in case that there is an energy increase, if the value of $e^{-\beta\Delta E}$ is larger than a random number (indicated as r in Fig. 1). These two cases are implemented through an OR gate that receives signals from two comparison gates that evaluates the procedure previously described. The output signal of the OR gate (indicated by S_change) is then XOR with the spin value in order to obtain

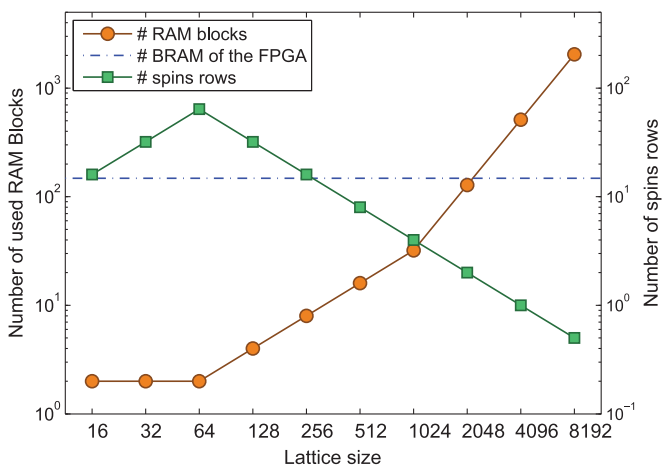

Fig. 2. RAM blocks resource utilization and number of spin rows that can be updated simultaneously as a function of the lattice size.

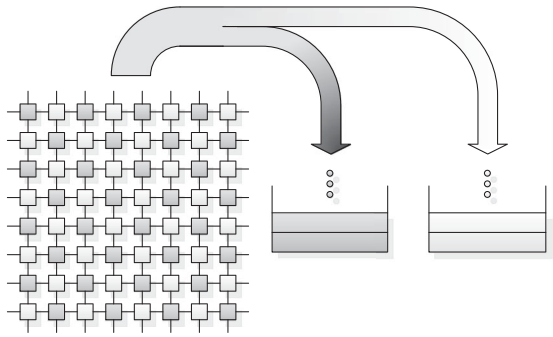

Fig. 3. Schematic representation of an 8×8 lattice showing the checkerboard division into two sub-lattices and the use of two memory modules to store spin values of each of the sublattices.

the updated one. Finally, the new updated value is registered in order to synchronize the procedure.

The updating procedure just described requires the logic resources indicated in Table 2, noting that the whole process can be executed in one clock cycle with a maximum frequency of 316.156 MHz. Nevertheless, the system frequency has been set to 300 MHZ, as this is the maximum frequency that can be obtained using a phase-locked loop (PLL) in order to optimize the implementation.

The values shown in Table 2 determine the maximum number of spins that can be updated in one cycle according to the FPGA board specifications, number that can be obtained from the following equation:

$$\text{Max\_Spin} \leq \frac{\text{Available LUTS}}{\text{LUTs per spin}}. \tag{6}$$

In the present case, the used board contains 69,120 LUTs, permitting a simultaneous maximum implementation of $\lfloor \frac{69120}{30} \rfloor_{2^m} = 2{,}048$ spin sites, value that is obtained from Eq. (6), where the obtained number should be a power of two to optimize resource utilization. The previous analysis does not take into account the whole process as in addition the storage of the lattice spin values are needed, and this introduces a memory limiting factor. To analyze both memory and LUTS requirements we need to analyze the whole updating process considering the following. The spins are located in a two-dimensional $L \times L$ square lattice that will be divided in two checkerboard sub-lattices, as the whole system cannot be updated simultaneously at the risk of a feedback catastrophe [29]. Fig. 3a) shows an 8×8 spin lattice in which two sub-lattices are shown (white and grey colors are used for differentiating them), together with two memory modules that will be used for storing these sub-lattice spin values. In our case the FPGA BRAM is organized in 148 blocks of 1,024 $\times$ 36-bits long. As each of the two memory modules store only half of the total spin site, the number of used RAM blocks for different lattice sizes is indicated in Fig. 2. It is also shown in this figure by a horizontal dashed dotted line the maximum number of BRAM of the board that sets a maximum lattice size of 2,048. Nevertheless, in order to maximize the parallelism of the procedure and in order to compare with previous works, we focused out study in the $1{,}024 \times 1{,}024$ case.

We describe below the whole updating process that will be carried simultaneously for a group of rows, with a limiting factor given by the maximum number of spins that can

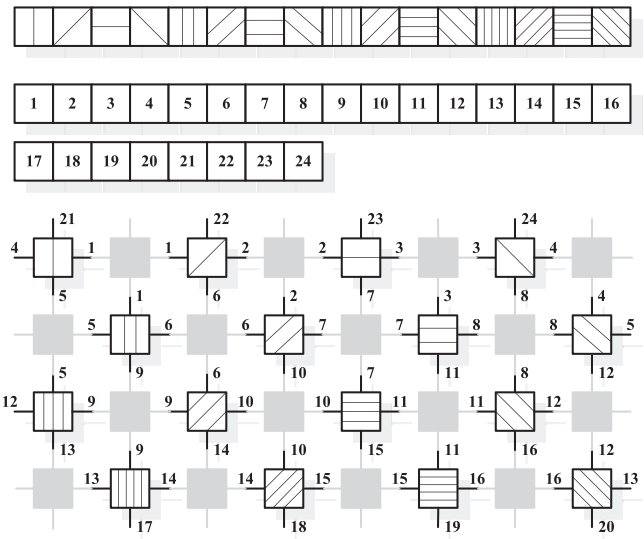

Fig. 4. Connectivity scheme for the update of four rows of an 8×8 Ising spin lattice. On top of the figure, three registers are used to store spin values corresponding to sites to be updated (first row) and to the neighboring sites (second and third rows).

be implemented. The maximum number of rows that can be updated simultaneously can be computed simply by considering the total number of spins that can be represented according to the number of LUTs of the board ($2{,}048$ in our case, see the analysis described above in relationship to Eq. (6)) divided by half of the lattice size:

$$\#\ \text{rows} = \frac{\text{Max\_Spin}}{L/2}. \tag{7}$$

The number of spin rows that can be updated simultaneously as just mentioned is shown in Fig. 2.

Fig. 4 shows the process implemented for updating the values of one of the two sub-lattices. On top of the figure, registers for storing the values of the spin to be updated and of their neighboring ones are shown. The first row of 16 bit register corresponds to the 16 spins that can be updated that are displayed at the bottom of the figure. The second and third rows of register are used to store the neighboring spins. In the bottom graph the white squares are the spins to be updated and on grey color the neighboring ones, where the indicated numbers and filling pattern correspond to the codification used in the registers shown.

The whole process for updating a 8×8 grid of spins is shown in Fig. 5, where the four steps involved are shown. In the first step the first four rows of spins belonging to the grey sub-lattice (16 spins for the 8×8 size case stored in the bottom address row of the RAM) are updated (a); to then update the remaining four rows (stored in the second address row) in the second step (b). Third and fourth steps correspond to the same updating process but for the spins belonging to the white sub-lattice.

The updating process starts with an initial configuration given by random spin values and a given value of the temperature (T), generating blocks of memory with these initial values. An update iteration starts by transferring the value of a row of spins to the slice registers and then to the block of LUTS that simulate the spin dynamics.

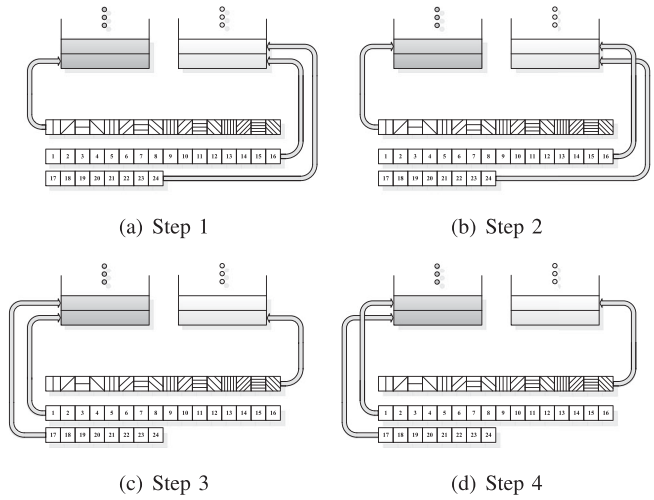

Fig. 5. Schematic representation of the hardware implementation for the update procedure of a 2D 8×8 spin grid.

5 LFSR Random Number Generation

Random numbers are an important part of Monte Carlo simulations, as they should be properly generated in order to avoid errors. As such, several authors have analyzed this topic in great depth [30], [31], [32]. Among the different algorithms to create random numbers, LFSR random number generators provide one of the most efficient alternatives to use in hardware logic implementations (FPGAs, ASICS, etc.). A LFSR works by taking a string of bits and producing the next sequence by shifting all except the rightmost bit one position to the right, setting the value of the new leftmost bit through a linear combination of the rest of the sequence.

In a recent work Lin et al. [23] have used a combination of a LFSR random number generator plus celullar automata (CA) in order to study a FPGA based simulation of the Ising model. Instead, in our implementation we used a novel approach based on the XOR combination of two LFSRs: a global 32-bit length and a shorter 12-bit local one. This method was chosen in order to use more efficiently the FPGA resources to allow larger numbers of spins to be updated simultaneously, and thus per unit of time, leading to a significant overall increase in computation speed (see the detailed analysis in Table 3 and related text).

The hardware LFSR implementation consists in M number of registers in series (synchronized by a unique clock signal), with M being the length of the bit sequence. The output of one register is the input of the next, except for the first one, which is a linear function (normally XOR or XNOR) of some bits of the overall shift register value. The two modules are shown in Fig. 6, where the top one corresponds to a part included in the implementation of every spin block. The bottom graph corresponds to a generic module that uses a 32 bits LFSR. Each spin block contains a 12 bit LFSR and the generation of pseudo random numbers for the Ising model updating is done by combining through an XOR function the 12 bits of the spin block LFSR with the first 12 bits of the sequence generated by the global LFSR32 module.

Two different kinds of tests have been used to check for the correct implementation of the random number

TABLE 3
Number of Spins Updated Per Microsecond for the 1,024 × 1,024 Ising Model

Platform	# updated spins	Ratio
CPU	62	1
Single GPU	7,977	129
Previous FPGA	94,127	1,518
64 GPUs	206,000	3,322
Our FPGA	614,400	9,909

generation process, the first described below based on a standard suite of statistical tests (National Institute of Standards and Technology (NIST)) and the a second one based on visual analysis.

5.1 Tests for Random Numbers

The NIST is a measurement standards laboratory belonging to the United States Department of Commerce [33]. One of its groups (the Random Number Generation Technical Working Group) has provided software containing a battery of statistical tests suitable for the evaluation of random number generators [34].

We have applied the battery of 10 tests provided by NIST to analyze the quality of the generated random numbers. The random number generator used a combination of a 32bit global LFSR and a 12 bit local LFSR passed all the ten tests and the results are shown in Fig. 7, obtaining an average for the p-values of 0.6311. As a comparison, we have also executed the tests on the 32 bit LFSR resulting in lower p-values (mean 0.5008), and a failure on the Rank test.

6 Results

To verify the correct FPGA implementation of the Metropolis-Hastings algorithm for the 2D-Ising model, we analyzed

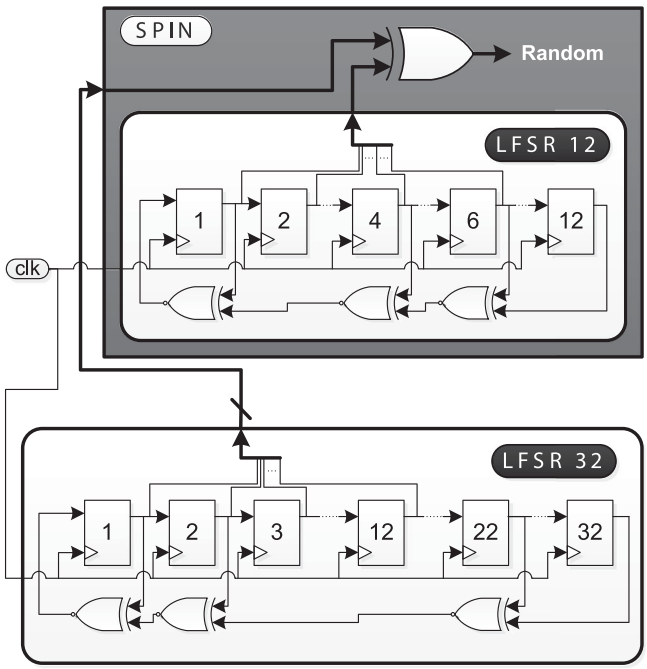

Fig. 6. Hardware representation of the creation of the random value in a generic spin, using a local LFSR12 and the general LFSR32.

```
------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND
THE PROPORTION OF PASSING SEQUENCES
------------------------------------------
   generator is <data/Data.24M>
------------------------------------------
 P-VALUE    PROPORTION   STATISTICAL TEST
------------------------------------------
 0.534146     10/10      Frequency
 0.350485     10/10      BlockFrequency
 0.739918     10/10      CumulativeSums
 0.991468     10/10      Runs
 0.350485     10/10      LongestRun
 0.739918     10/10      Rank
 0.739918     10/10      FFT
 0.213309     10/10      OverlappingTemplate
 0.739918     10/10      Universal
 0.911413     10/10      ApproximateEntropy

- - - - - - - - - - - - - - - - - - - - - - -
The minimum pass rate for each statistical
test with the exception of the random
excursion (variant) test is approximately = 8
for a sample size = 10 binary sequences.
- - - - - - - - - - - - - - - - - - - - - - -
```

Fig. 7. Output screenshot of the NIST random number evaluation program that includes ten statistical tests (see text for more details).

the model's critical behavior, which is characterized by the presence of long-range spatial and temporal correlations, as well a diverging response functions [35] (for instance, magnetic susceptibility). In this regime the system becomes extremely sensitive to external perturbations like, in particular, implementation errors in the simulation algorithm. The typical analysis of critical behavior involves the calculation of the magnetization and magnetic susceptibility as a function of the temperature for increasing system sizes. For an infinite system size the magnetization is zero above the critical temperature T_c and different from zero below it, while the susceptibility diverges at T_c. The respective curves for finite size systems for both quantities have to develop those singularities as the size increases, according to specific rules determined by finite size scaling [35].

We started by obtaining the magnetization m in the absence of an external magnetic field $B = 0$ as a function of the temperature for different lattice sizes according to Eq. (4). In order to thermalize the system 1,000 MCS were executed prior to any measurement, using 1,000 values for obtaining the average value for each temperature with a 100 MCS separation between consecutive measurements.

The results are compared with the exact result [15] in Fig. 8. Although the qualitative agreement is evident, the calculation of a quantity more sensitive to the fluctuations (such as a critical exponent) is needed in order to ensure that the random numbers implementation does not introduce a statistical bias. Hence, we also computed the magnetic susceptibility (cf. Eq. (5)), which is displayed in Fig. 9. The maximum of the susceptibility it is known to scale as $\chi_{max} \sim L^{\gamma/\nu}$ with the system size, where γ and ν are the critical exponents of the susceptibility and the correlation length respectively [36].

In order to estimate $\chi_{max}(L)$ we performed a Lorentzian fitting of the susceptibility curves for each value of L (see Fig. 9). The log-log plot of $\chi_{max}(L)$ versus L shown in the inset of Fig. 9 displays the expected power law behavior; a

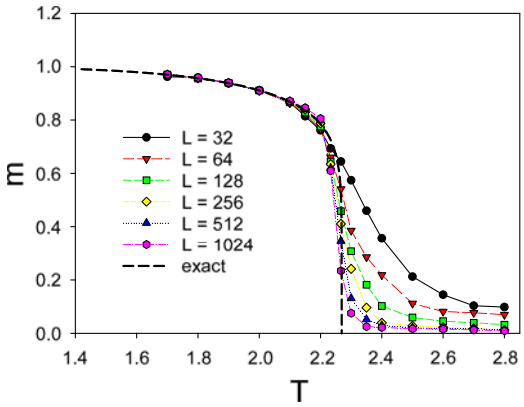

Fig. 8. Magnetization of a 2D Ising model as a function of the temperature for different lattice sizes, compared with the exact result (dashed line). The continuous lines are a guide to the eye.

linear fitting of this plot provides an estimate $\gamma/\nu = 1.72 \pm 0.05$, in a very good agreement with the exact result $\gamma/\nu = 1.75$ [25].

Finite size scaling provides also a method to estimate the critical temperature T_c. The maximum of the susceptibility is located at a pseudo critical temperature T^* that depends on the system size and converges to the critical temperature T_c as $T^*(L) \sim T_c + b/L$, for large enough values of L, with $b > 0$ [36]. From the Lorentzian fittings we also estimated $T^*(L)$. In Fig. 10 we plot $T*(L)$ versus $1/L$. A linear extrapolation to $1/L \to 0$ allowed us the estimate $T_c = 2.27 \pm 0.1$, in excellent agreement with the exact result [15] $T_c = 2/\ln(1+\sqrt{2}) = 2.269\ldots$. This set of results confirms the correct implementation of the algorithm.

Further, we have also compared the number of spins updates in a microsecond obtained from the current implementation to previously obtained values from [23] regarding CPU, GPU and previous FPGA implementations. The results are shown in Table 3, where all values except those in the last row were extracted from the work of Lin et al. [23]. To put the results shown in Table 3 in context, we give next some details of the hardware and methods used for their obtention: the CPU platform results were obtained

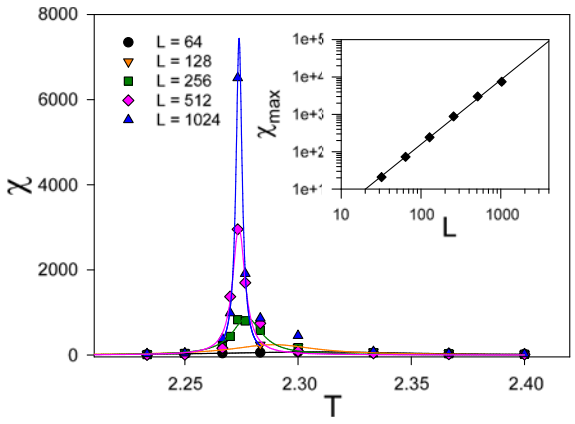

Fig. 9. Magnetic susceptibility χ for the zero magnetic field 2D Ising model as a function of the temperature for different lattice sizes. The continuous lines correspond to a Lorentzian fitting close to the maximal. The inset shows a log-log plot of the susceptibility maximum χ_{max} as a function of the system size (error bars are smaller than the symbol size); the linear fitting ($r^2 = 0.9978$) provides an estimate of the critical exponent $\gamma/\nu = 1.72 \pm 0.05$.

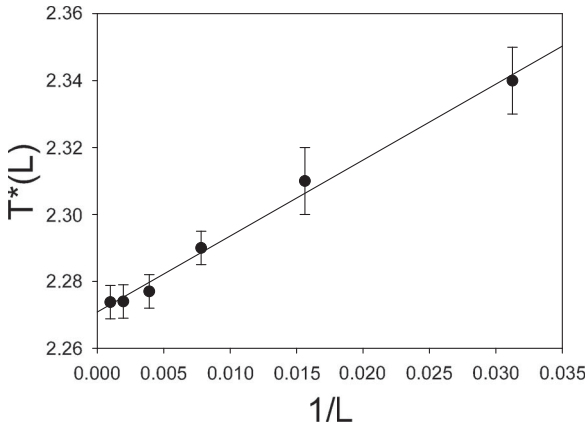

Fig. 10. Pseudo critical temperature $T^*(L)$ as a function of $1/L$. The linear fitting ($r^2 = 0.99$) provides the estimate $T_c = 2.27 \pm 0.1$ (see text for details).

using an Intel core i5 at 2.67 GHz (CPU) applying some optimization techniques like a linear congruential generator (LCG), sequential updating and cashed Boltzmann. "Previous FPGA" values were obtained using a DK-DEV-3CI20N board from Altera, while results in relationship to the "64 GPUs" system were achieved using 64 interconnected GPU processors in a supercomputer. For a discussion about the advantages/disadvantages of the current available hardware computing devices, in relationship to magnetic systems and in a more general context, we refer to the works of [9], [23], [37], [38].

In relationship to the computation time required to execute the numerical simulations presented in this work, we have estimated the times employed to compute one MCS and a whole simulation for obtaining one data value at a given temperature. The results are shown in Table 4, as a function of the system lattice size.

7 Possible Extensions

The implementation described in the sections above can be extended with relatively modest modifications to systems of current research interest. Here we will describe three basic extensions that will illustrate the required changes in the implementation.

Perhaps the simplest extension involves the inclusion of next-nearest-neighbor interactions. The next-nearest neighbors of the spin at position (i, j) are defined as the four spins at positions $(i \pm 1, j \pm 1)$, that is along the diagonals of the spin at (i, j). Of particular interest is the case in which the interaction with first neighbors is ferromagnetic and anti-ferromagnetic with next-nearest neighbors. This can be modelled by introducing a negative exchange constant that mediates the interaction of each spin with

TABLE 4
Time Employed to Compute One MCS and a Whole Simulation Carried to Obtain Data at a Given Temperature, as a Function of System Lattice Size

Size Lattice	16	32	64	128	256	512	1,024
1 MCS (ns)	6.6	6.6	6.6	26.6	106.6	426.6	1,706.6
TOTAL (ms)	0.6	0.6	0.6	2.7	10.8	43.1	172.4

the second neighbors, as represented in the following Hamiltonian:

$$H = -J_1 \sum_{\langle i,j \rangle} S_i S_j - J_2 \sum_{\langle\langle i,j \rangle\rangle} S_i S_j \,, \tag{8}$$

where $J_1 > 0$, $J_2 < 0$, and $\langle\langle \ldots \rangle\rangle$ denotes that the sum runs over all next-nearest neighbor pairs. This model represents a binary spin version of the J_1/J_2 model [39], [40].

The implementation of the binary J_1/J_2 model requires a small number of changes in the logic of the FPGA circuit shown in Fig. 1. The main differences are the following:

a) The values of the four next-nearest neighbor of the spin to be updated (L′, T′, R′, and B′) need to be considered in the circuit.
b) Instead of the checkboard division into two sublattices the system can be divided into four sublattices in such a way that spins belonging to the same sublattice are neither nearest neighbors nor next-nearest neighbors—hence, requiring four memory modules to store the spin values of each sublattice.
c) The change in energy associated with a single spin flip is equal to $\Delta E = 2\epsilon'$, where ϵ' is given by the following expression,

$$\begin{aligned} \epsilon' = (&J_1(S_1 + S_2 + S_3 + S_4) \\ &+ J_2(S'_1 + S'_2 + S'_3 + S'_4)) \cdot S_0 \,. \end{aligned} \tag{9}$$

d) The look-up table for the energy values needs to be expanded since now the change in energy can take a wider range of values. In practical implementation it is usual to take J_1 or J_2 equal to 1 and then allow the other exchange constant to adjust the relative strength of the ferromagnetic/anti-ferromagnetic interactions.

A second example of possible extensions is the implementation of the Potts model [41]. The Potts model is a generalization of the Ising model in which each individual spin can take an integer value $S_i = 1, \ldots, q$. Its Hamiltonian is given by the following:

$$H = -J \sum_{\langle i,j \rangle} \delta(S_i, S_j) \,, \tag{10}$$

where $\delta(S_i, S_j)$ is the Kronecker delta, whose value is 1 if $S_i = S_j$ and zero otherwise. The implementation of the Potts model implies some more significant changes to the logic in Fig. 1. The main ones are the following:

a) Instead of using Boolean variables as in the case of the Ising model, the Potts model requires that each spin can take one of q different states
b) For the update of spin S_i into the new state S'_i the Monte Carlo algorithm proceeds by choosing at random one of the $q-1$ states for which $S_i \neq S'_i$, then computing the corresponding energy difference ΔE that the change implies, and finally the decision to accept it or not follows the same logic as the one for the Ising model.

One final example is the 3D Ising model. Its implementation requires a small modification of the spin block in comparison with the 2D Ising, as six neighboring spins should be considered instead of four. This change will affect the size of the lookup table because the energy range will be larger. Memory management for this model will be also more complex and as in principle memory resources may not be enough to handle the whole system divided in two sub-lattices, but a layered scheme should be used where the whole system is analyzed as composed of M 2D Ising models.

8 Conclusions and Discussion

The results of this work confirm the potential of FPGA boards for the simulation of statistical mechanics models, demonstrating a significant improvement in terms of the numbers of spins updated per second was obtained in comparison to previous work (Table 3). The improvements in performance are a consequence of efficient spin implementation and resource utilization. The spin representation was clearly described and analyzed in terms of the logic resources involved (cf. Figs 1 and 2, and Table 2), and the use of a combined global-local random number LFSR generator proved to be a significant factor in increased performance. The procedure based on the combination of two LFSR (one local and one global with the respect to the implementation of a block of spins) was used, rigorously tested in terms of the quality of the random numbers produced using a standard suite of statistical tests [33]. In relation to the previous work by Lin et al. [23], it is worth noting that their analysis showed that using a true RNG does not improved much the quality of the random numbers obtained. On the other hand, our approach used a local LFSR instead of a CA, and this fact permitted us to utilize efficiently the board resources to get faster spin updates.

These improvements summed to the parallelism provided by the FPGA implementation offer a significant potential for extensions to other models of magnetic systems. The 2D Ising model requires a minimal number of modifications to be extended to more complicated models like the binary spin version of the J_1/J_2 model, the Potts model, or the 3D Ising model. Further work is required to exploit the massive parallelism of FPGAs in systems with more complex (long-range) interactions like the Ising dipolar model [42], which would require a complete redesign of the logic involved.

Acknowledgments

The authors gratefully acknowledge the constructive comments made by the anonymous reviewers. The authors acknowledge support from Junta de Andalucia through grants P10-TIC-5770, from MICIIN(Spain) through grants TIN2010-16556 and TIN2014-58516-c2-1-R (all including FEDER funds), and from CONICET (Argentina) through grant PIP11220110100213.

References

[1] A. Vajda, *Programming Many-Core Chips*, 1st ed. New York, NY, USA: Springer, 2011.
[2] M. A. Suchard, Q. Wang, C. Chan, J. Frelinger, A. Cron, and M. West, "Understanding GPU programming for statistical computation: Studies in massively parallel massive mixtures," *J. Comput. Graph. Statist.*., vol. 19, no. 2, pp. 419–438, 2010.

[3] K. Sano, Y. Hatsuda, and S. Yamamoto, "Multi-FPGA accelerator for scalable stencil computation with constant memory bandwidth," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 695–705, Mar. 2014.
[4] R. Weber, A. Gothandaraman, R. Hinde, and G. Peterson, "Comparing hardware accelerators in scientific applications: A case study," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 1, pp. 58–68, Jan. 2011.
[5] S. Sun and J. Zambreno, "Design and analysis of a reconfigurable platform for frequent pattern mining," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 9, pp. 1497–1505, Sep. 2011.
[6] E. Monmasson, L. Idkhajine, M. N. Cirstea, I. Bahri, A. Tisan, and M. W. Naouar, "FPGAs in industrial control applications," *IEEE Trans. Ind. Informat.*, vol. 7, no. 2, pp. 224–243, May 2011.
[7] T. Ganegedara, W. Jiang, and V. Prasanna, "A scalable and modular architecture for high-performance packet classification," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 5, pp. 1135–1144, May 2014.
[8] K. Binder. (2001). Ising model. in *Encyclopedia of Mathematics*, M. Hazewinkel, Eds. [Online]. Available: http://www.encyclopediaofmath.org/index.php/Ising_model
[9] E. E. Ferrero, J. P. De Francesco, N. Wolovick, and S. A. Cannas, "Q-state potts model metastability study using optimized GPU-based monte carlo algorithms," *Comput. Phys. Commun.*, vol. 183, p. 1578, 2012.
[10] W. P. Wolf, "The Ising model and real magnetic materials," *Brazilian J. Phys.*, vol. 30, pp. 794–810, 2000.
[11] J. A. Glazier and D. Weaire, "The kinetics of cellular patterns," *J. Phys: Condens. Matter*, vol. 4, pp. 1867–1894, 1992.
[12] S. Sanyal and J. A. Glazier, "Viscous instabilities in flowing foams: A cellular potts model approach," *J. Statist. Mech.: Theory Experiment*, vol. 2006, no. 10, p. P10008, 2006.
[13] F. Graner and J. A. Glazier, "Simulation of biological cell sorting using a two-dimensional extended potts model," *Phys. Rev. Lett.*, vol. 69, pp. 2013–2016, 1992.
[14] D. J. Amit, *Modeling Brain Function*. Cambridge, U.K.: Cambridge Univ. Press, 1992.
[15] K. Huang, *Statistical Mechanics*, 2nd ed. New York, NY, USA: Wiley, 1987.
[16] W. Hastings, "Monte carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, pp. 97–109, 1970.
[17] K. Binder, *The Monte Carlo Method in Condensed Matter Physics*, 1st ed. New York, NY, USA: Springer, 1995.
[18] W. Hurd, "Efficient generation of statistically good pseudonoise by linearly interconnected shift registers," *IEEE Trans. Comput.*, vol. C-20, no. 2, pp. 146–152, Feb. 1974.
[19] P. Kinsman and N. Nicolici, "Noc-based FPGA acceleration for monte carlo simulations with applications to spect imaging," *IEEE Trans. Comput.*, vol. 62, no. 3, pp. 524–535, Mar. 2013.
[20] J. Cassidy, L. Lilge, and V. Betz, "Fast, power-efficient biophotonic simulations for cancer treatment using FPGAs," in *Proc. IEEE 22nd Annu. Int. Symp. Field-Programmable Custom Comput. Mach.*, May 2014, pp. 133–140.
[21] M. Baity-Jesi and A. E. A. Banos, *An FPGA-Based Supercomputer for Statistical Physics: The Weird Case of Janus*. New York, NY, USA: Springer, 2013.
[22] F. Belletti, M. Cotallo, and A. Cruz et al., "Janus: An FPGA-based system for high-performance scientific computing," *Comput. Sci. Eng.*, vol. 11, no. 1, pp. 48–58, 2009.
[23] Y. Lin, F. Wang, X. Zheng, H. Gao, and L. Zhang, "Monte carlo simulation of the Ising model on FPGA," *J. Comput. Phys.*, vol. 237, pp. 224–234, 2013.
[24] A. Gilman, A. Leist and K. Hawick, "3d lattice monte carlo simulations on FPGAs," in *Proc. Int. Conf. Comput. Design*, 2013, pp. 72–78.
[25] L. Reichl, *A Modern Course in Statistical Physics*, 2nd ed. New York, NY, USA: Wiley, 2004.
[26] S. Kilts, *Advanced FPGA Design: Architecture, Implementation, and Optimization*. New York, NY, USA: Wiley, 2007.
[27] A. W. Savich, M. Moussa, and S. Areibi, "The impact of arithmetic representation on implementing MLP-BP on FPGAs: A study," *IEEE Trans. Neural Networks*, vol. 18, no. 1, pp. 240–252, Jan. 2007.
[28] Z. Jovanovic and V. Milutinovic, "FPGA accelerator for floating-point matrix multiplication," *IET Comput. Digital Techn.*, vol. 6, no. 4, pp. 249–256, Jul. 2012.
[29] G. Y. Vichniac, "Simulating physics with cellular automata," *Physica D*, vol. 10, pp. 96–116, 1984.
[30] P. L'Ecuyer, "Tables of maximally equidistributed combined LFSR generators," *Math. Comput.*, vol. 68, no. 225, pp. 261–269, 1999.
[31] G. Marsaglia, "Random number generators," *J. Modern Appl. Statistical Methods*, vol. 2, no. 1, pp. 2–13, May 2003.
[32] M. Saito and M. Matsumoto, "SIMD-oriented fast mersenne twister: A 128-bit pseudorandom number generator," in *Monte Carlo and Quasi-Monte Carlo Methods 2006*, A. Keller, S. Heinrich, and H. Niederreiter, Eds. Berlin, Germany: Springer, 2008, ch. 36, pp. 607–622.
[33] U. D. Commerce. National Institute of Standard Technology. [Online]. Available: http://www.nist.gov/
[34] A. Rukhin, J. Soto, J. Nechvatal, E. Barker, S. Leigh, M. Levenson, D. Banks, A. Heckert, J. Dray, S. Vo, A. Rukhin, J. Soto, M. Smid, S. Leigh, M. Vangel, A. Heckert, J. Dray, and L. E. Bassham III, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," Nat. Inst. Standards & Technol., Gaithersburg, MD, USA, Tech. Rep. SP 800-22 Rev. 1a., 2010.
[35] N. Goldenfeld, *Lectures on Phase Transitions and the Renormalization Group*. Boulder, CO, USA: Westview, 1992.
[36] D. Landau and K. Binder, *A Guide to Monte Carlo Simulations in Statistical Physics*, 3rd ed. Cambridge, U.K.: Cambridge Univ. Press, 2009.
[37] D. Jones, A. Powell, C. Bouganis, and P. Cheung, "GPU versus FPGA for high productivity computing," in *Proc. Int. Conf. Field Programmable Logic Appl.*, Aug. 2010, pp. 119–124.
[38] C. Cullinan, C. Wyant, T. Frattesi, and X. Huang, "Computing performance benchmarks among CPU, GPU, and FPGA," *WPI*, 2012.
[39] H. Shang and C. Lamas, "Exotic disordered phases in the quantum j1-j2 model on the honeycomb lattice," *Phys. Rev. B*, vol. 87, p. 024415, 2013.
[40] A. I. Guerrero, D. A. Stariolo, and N. G. Almarza, "Nematic phase in the [J1-J2] square-lattice ising model in an external field," *Phys. Rev. E*, vol. 91, May 2015.
[41] F.-Y. Wu, "The potts model," *Rev. Modern Phys.*, vol. 54, no. 1, p. 235, 1982.
[42] S. Cannas, M. Michelon, D. Stariolo, and F. Tamarit, "Interplay between coarsening and nucleation in an Ising model with dipolar interactions," *Phys. Rev. E*, vol. 78, p. 051602, 2008.

Francisco Ortega-Zamorano was born in 1980. He received the MSc and PhD degree in computer science from the University of Málaga, Málaga, Spain, in 2013 and 2015, respectively. He joined the Department of Computer Languages and Computer Science, University of Málaga in 2011, where he is currently a postdoctoral researcher. His current research interests include FPGA hardware implementation, neurocomputational real-time systems, smart sensor networks, and deep learning.

Marcelo A. Montemurro received the PhD degree in theoretical physics from the National University of Córdoba, Córdoba, Argentina, in 2002. He then moved to Italy with a fellowship from UNESCO to work at the International Centre for Theoretical Physics in Trieste. In 2004, he moved to the University of Manchester where he held a number of fellowships before being appointed Lecturer. His research interests include the statistical physics of complex systems and computational neuroscience.

Sergio Alejandro Cannas was born in Córdoba, Argentina, on November 21, 1961. He received the MSc degree from the National University of Córdoba, Córdoba, Argentina, in 1984, and the PhD degree from the Centro Brasileiro de Pesquísas Físicas, Brazil, in 1992, both with a thesis and dissertation on statistical physics. In 1994, he became an assistant professor, in 2004 an associate professor, and since 2010, he is a full professor of the National University of Córdoba. In 1995, he became a researcher of the National Research Council (CONICET) of Argentina. His research interests include neural networks and statistical physics of complex systems.

José M. Jerez received the PhD degree in computer science from the University of Málaga, Spain, in 2003. He is currently an associate professor with the University of Málaga. His research interests include the areas of computational intelligence, image analysis, and bioinformatics. In particular, he is developing prediction software for biomedical problems using artificial intelligence techniques in collaboration with the Málaga University hospital. He has participated in more than 15 research projects funded by international and national boards and he belongs to several reviewing scientific committees.

Leonardo Franco (M'06-SM'13) received the MSc and PhD degrees in 1995 and 2000, respectively, both from the National University of Córdoba, Córdoba, Argentina. He was then a postdoctoral researcher at SISSA, Italy, and Oxford University, U.K. Since 2005, he has been with the Department of Computer Science, Málaga University, Spain, where he is currently an associate professor. His research interests include neural networks, computational intelligence, biomedical applications, and computational neuroscience. He has authored more than 60 publications in journals and international conference proceedings. He is a senior member of the IEEE.