Scaling advantage of nonrelaxational dynamics for high-performance combinatorial optimization

Timothée Leleu[1,2], Farad Khoyratee[3,4], Timothée Levi[5,6], Ryan Hamerly[7,8], Takashi Kohno[1,2,6], and Kazuyuki Aihara[1,2]

[1]Institute of Industrial Science, University of Tokyo, Japan
[2]International Research Center for Neurointelligence, University of Tokyo, Japan
[3]Theoretical Quantum Physics Laboratory, RIKEN Cluster for Pioneering Research, Saitama, Japan
[4]Physics Department, The University of Michigan, Ann Arbor, MI, USA
[5]IMS, University of Bordeaux, France
[6]LIMMS/CNRS-IIS, the University of Tokyo, Japan
[7]Massachusetts Institute of Technology, Cambridge, MA, USA
[8]NTT Research, 1950 University Ave #600, East Palo Alto, California 94303, USA

March 10, 2021

Abstract

The development of physical simulators, called Ising machines, that sample from low energy states of the Ising Hamiltonian has the potential to drastically transform our ability to understand and control complex systems. However, most of the physical implementations of such machines have been based on a similar concept that is closely related to relaxational dynamics such as in simulated, mean-field, chaotic, and quantum annealing. We show that nonrelaxational dynamics that is associated with broken detailed balance and positive entropy production rate can accelerate the sampling of low energy states compared to that of conventional methods. By implementing such dynamics on field programmable gate array, we show that the nonrelaxational dynamics that we propose, called chaotic amplitude control, exhibits a scaling with problem size of the time to finding optimal solutions and its variance that is significantly smaller than that of relaxational schemes recently implemented on Ising machines.

Introduction

Many complex systems such as spin glasses, interacting proteins, large scale hardware, and financial portfolios, can be described as ensembles of disordered elements that have competing frustrated interactions[1] and rugged energy landscapes. There has been a growing interest in using physical simulators called "Ising machines" in order to reduce time and resources needed to identify configurations that minimize their total interaction energy, notably that of the Ising Hamiltonians $\mathcal{H}$ with $\mathcal{H}(\boldsymbol{\sigma}) = -\frac{1}{2}\sum_{ij}\omega_{ij}\sigma_i\sigma_j$ (with ω_{ij} the symmetric Ising couplings, i.e., $\omega_{ij} = \omega_{ji}$, and $\sigma_i = \pm 1$) that is related to many nondeterministic polynomial-time hard (NP-hard) combinatorial optimization problems and various real-world applications[2] (see [3] for a review). Recently proposed implementations include memresistor networks[4], micro- or nano-electromechanical systems[5], micro-magnets[6, 7], coherent optical systems[8], hybrid opto-electronic hardware[5, 10, 11], integrated photonics[12, 13, 14], flux qubits[15], and Bose-Einstein condensates[16]. In principle, these physical systems often possess unique properties, such as coherent superposition in flux qubits[17] and energy efficiency of memresistors[18, 4], which could lead to a distinctive advantage compared to conventional computers (see Fig. 1 (a)) for the sampling of low energy states. In practice, the difficulty in constructing connections between constituting elements of the hardware is often the main limiting factor to scalability and performance for these systems[19, 15]. Moreover, these devices often implement schemes that are directly related to the concept of annealing (either simulated[20, 21], mean-field[2, 3], chaotic[18, 24], and quantum[25, 17]) in which the escape from the numerous local minima[26] and saddle points[27] of the free energy function can only be achieved under very slow modulation of the control parameter (see Fig. 1 (b)). These methods are dependent on non-equilibrium dynamics called aging that, according to recent numerical studies[28], is strongly non-ergodic and seems to explore only a confined subspace determined by the initial condition rather than wander in the entire configurational space[29] for mean-field spin glass models. In general, such systems find the solutions of minimal energy only after many repetitions of the relaxation process.

Interestingly, alternative dynamics that is not based on the concepts of annealing and relaxation may perform better for solving hard combinatorial optimization problems[30, 31, 32]. Various kinds of dynamics have been proposed[33, 34, 35, 36, 3], notably chaotic dynamics[37, 38, 18, 39, 40], but have either not been implemented onto specialized hardware[41, 37] or use chaotic dynamics merely as a replacement to random fluctuations[38, 18]. We have recently proposed that the control of amplitude in mean-field dynamics can significantly improve the performance of Ising machines by introducing error correction terms (see Fig. 1 (d)), effectively doubling the dimensionality of the system, whose role is to correct the amplitude heterogeneity[30]. Because of the similarity of such dynamics with that of a neural network, it can be implemented especially efficiently in electronic neuromorphic hardware where memory is distributed with the processing[42, 43, 44]. In this paper, we show that this nonrelaxational dynamics is able to escape at a much faster rate than relaxational ones from local minima and saddles of the corresponding energy function. Importantly, the exponential scaling factor with respect to system size of the time needed to reach ground-state configura-

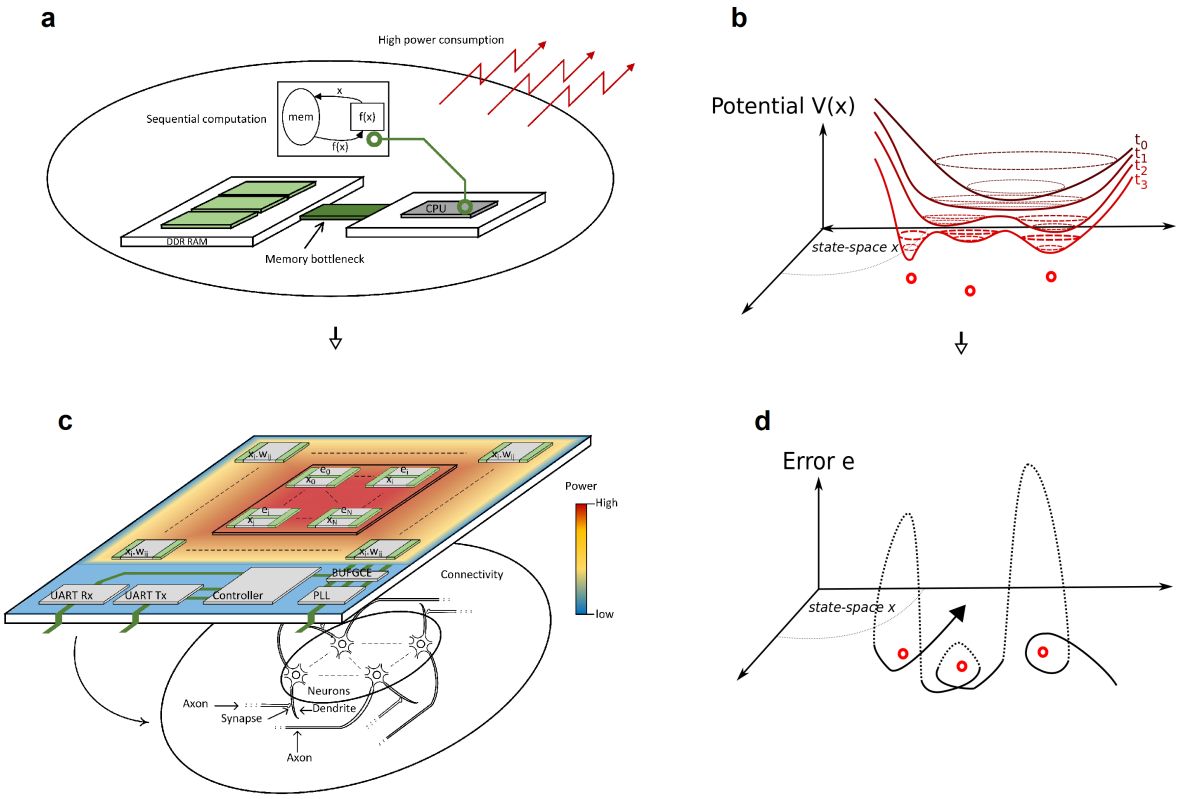

Figure 1: Schematic representation of (a) conventional CPU architecture with the von Neumann bottleneck problem and (c) the proposed neuromorphic chip for combinatorial optimization. Schema of state-space dynamics of algorithms based on (b) annealing on a potential function and (d) the proposed chaotic amplitude control scheme.

tions of spin glasses, called time to solution, is significantly smaller in the former case, which raises the question whether this nonrelaxational dynamics is qualitatively different from the very slow relaxation observed in classic Monte Carlo simulations of spin glasses. In order to extend numerical analysis to large problem sizes and limit finite-size effects, we implement a version of nonrelaxational dynamics that we name chaotic amplitude control (CAC) on a field programmable gate array (FPGA, see Fig. 1 (c)) and show that the developed hardware can be faster for finding ground-states in the limit of large problem sizes than many state-of-the-art algorithms and Ising machines for some reference benchmarks with enhanced energy efficiency.

Results

For the sake of simplicity, we consider the classical limit of Ising machines for which the state space is often naturally described by analog variables (i.e., real numbers) noted x_i in

the following. The variables x_i represent measured physical quantities such as voltage[4] or optical field amplitude[8, 5, 10, 11, 12, 13] and these systems can often be simplified to networks of interacting nonlinear elements whose time evolution can be written as follows:

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = f_i(x_i) + \beta_i(t)\sum_j \omega_{ij} g_j(x_j) + \sigma_0 \eta_i, \tag{1}$$

where f_i and g_i represent the nonlinear gain and interaction, respectively, and are assumed to be monotonic, odd, and invertible "sigmoidal" functions for the sake of simplicity; η_i, experimental white noise of standard deviation σ_0 with $\langle \eta_i(t)\eta_j(t')\rangle = \delta_{ij}\delta(t-t')$[1]; and N, the number of spins. Ordinary differential equations similar to eq. (1) have been used in various computational models that are applied to NP-hard combinatorial optimization problems such as Hopfield-Tank neural networks[45], coherent Ising machines[46], and correspond to the "soft" spin description of frustrated spin systems[47]. Moreover, the steady states of eq. (1) correspond to the solutions of the "naive" Thouless-Anderson-Palmer (nTAP) equations that arise from the mean-field description of Sherrington-Kirkpatrick spin glasses when the Onsager reaction term has been discarded[48]. In the case of neural networks in particular, the variables x_i and constant parameters ω_{ij} correspond to firing rates of neurons and synaptic coupling weights, respectively.

It is well known that, when $\beta_i = \beta$ for all i and the noise is not taken into account ($\sigma_0 = 0$), the time evolution of this system is motion in the state space that seeks out minima of a potential function[1] (or Lyapunov function) V given as $V = \beta\mathcal{H}(\boldsymbol{y}) + \sum_i V_b(y_i)$ where V_b is a bistable potential with $V_b(y_i) = -\int_0^{y_i} f_i(g_i^{-1}(y))\,\mathrm{d}y$ and $\mathcal{H}(\boldsymbol{y}) = -\frac{1}{2}\sum_{ij}\omega_{ij}y_i y_j$ is the extension of the Ising Hamiltonian in the real space with $y_i = g_i(x_i)$ (see Supplementary Materials S1.1). The bifurcation parameter β, which can be interpreted as the inverse temperature of the naive TAP equations[48], the steepness of the neuronal transfer function in Hopfield-Tank neural networks[45], or to the coupling strength in coherent Ising machines[8, 5, 10], is usually decreased gradually in order to improve the quality of solutions found. This procedure has been called mean-field annealing[3], and can be interpreted as a quasi-static deformation of the potential function V (see Fig. 1 (b)). There is, however, no guarantee that a sufficiently slow deformation of the landscape V will ensure convergence to the lowest energy state contrarily to the quantum adiabatic theorem[50] or the convergence theorem of simulated annealing[51][2]. Moreover, the statistical analysis of spin glasses suggests that the potential V is highly non-convex at low temperature and that simple gradient descent very unlikely reaches the global minimum of $\mathcal{H}(\boldsymbol{\sigma})$ because of the presence of exponentially numerous local minima[26] and saddle points[27] as the size of the system increases. The

[1] δ_{ij}; $\delta(t)$, the Kronecker delta symbol and Dirac delta function, respectively.

[2] At fixed β, global convergence to the minimum of the potential V can be assured if σ_0 is gradually decrease with $\sigma_0(t)^2 \sim \frac{c}{log(2+t)}$ and c sufficiently large[52]. The parameter σ_0^2 is analogous to the temperature in simulated annealing in this case. The global minimum of the potential V does not, however, generally correspond to that of the Ising Hamiltonians $\mathcal{H}$ at finite β.

slow relaxation time of Monte-Carlo simulations of spin glasses, such as when using simulated annealing, might also be explained by similar trapping dynamics during the descent of the free energy landscape obtained from the TAP equations[27]. In the following, we consider in particular the soft spin description obtained by taking $f_i(x_i) = (-1+p)x_i - x_i^3$ and $y_i = g(x_i) = x_i$, where p is the gain parameter, which is the canonical model of the system described in eq. (1) at proximity of a pitchfork bifurcation with respect to the parameter p. In this case, the potential function V_b is given as $V_b(x_i) = (-1+p)\frac{x_i^2}{2} - \frac{x_i^4}{4}$ and eq. (1) can be written as $\frac{\mathrm{d}x_i}{\mathrm{d}t} = -\frac{\partial V}{\partial x_i}$, $\forall i$.

In order to define nonrelaxational dynamics that is inclined to visit spin configurations associated with lower Ising Hamiltonian, we introduce error signals, noted $e_i \in \mathbb{R}$, that modulate the strength of coupling β_i to the ith nonlinear element such that $\beta_i(t)$ defined in eq. (1) is expressed as $\beta_i(t) = \beta e_i(t)$ with $\beta > 0$. The time evolution of the error signals e_i are given as follows[30]:

$$\frac{\mathrm{d}e_i}{\mathrm{d}t} = -\xi(g(x_i)^2 - a)e_i, \tag{2}$$

where a and ξ are the target amplitude and the rate of change of error variables, respectively, with $a > 0$ and $\xi > 0$. If the system settles to a steady state, the values $y_i^* = g(x_i^*)$ become exactly binary with $y_i^* = \pm\sqrt{a}$. When $p < 1$, the internal fields h_i at the steady state, defined as $h_i = \sum_j \omega_{ij}\sigma_j$ with $\sigma_j = y_j^*/|y_j^*|$, are such that $h_i\sigma_i > 0$, $\forall i$[30]. Thus, each equilibrium point of the analog system corresponds to that of a zero-temperature local minimum of the binary spin system.

The dynamics described by the coupled equations (1) and (2) is not derived from a potential function because error signals e_i introduce asymmetric interactions between the x_i and the computational principle is not related to a gradient descent. Rather, the addition of error variables results in additional dimensions in the phase space via which the dynamics can escape local minima. The mechanism of this escape can be summarized as follows. It can be shown (see the Supplementary Materials S1.2) that the dimension of the unstable manifold at equilibrium points corresponding to local minima $\boldsymbol{\sigma}$ of the Ising Hamiltonian depends on the number of eigenvalues $\mu(\boldsymbol{\sigma})$ with $\mu(\boldsymbol{\sigma}) > F(a)$ where $\mu(\boldsymbol{\sigma})$ are the eigenvalues of the matrix $\{\frac{\omega_{ij}}{|h_i|}\}_{ij}$ (with internal field h_i) and F a function given as $F(y) = \frac{\psi'(y)}{\psi(y)}y$ and $\psi(y) = \frac{f(g^{-1}(y))}{(g^{-1})'(y)}$. Thus, there exists a value of a such that all local minima (including the ground state) are unstable and for which the system exhibits chaotic dynamics that explores successively candidate boolean configurations. The energy is evaluated at each step and the best configuration visited is kept as the solution of a run. Interestingly, this chaotic search is particularly efficient for sampling configurations of the Ising Hamiltonian close to that of the ground state using a single run although the distribution of sampled states is not necessarily given by the Boltzmann distribution. Note that the use of chaotic dynamics for solving Ising problems has been discussed previously[53, 18], notably in the context of neural networks, and it has been argued that chaotic fluctuations may possess better properties than Brownian

noise for escaping from local minima traps. In the case of the proposed scheme, the chaotic dynamics is not merely used as a replacement to noise. Rather, the interaction between nonlinear gain and error-correction results in the destabilization of states associated with lower Ising Hamiltonian.

Ensuring that fixed points are locally unstable does not guarantee that the system does not relax to periodic and chaotic attractors. We have previously proposed that non-trivial attractors can also be destabilized by ensuring the positive rate of entropy production using a modulation of the target amplitude[30]. In this paper, we propose an alternative heuristic modulation of the target amplitude that is better suited for a digital implementation than the one proposed in [30]. Because the value of a for which all local minima is unstable is not known *a priori*, we propose instead to destabilize the local minima traps by dynamically modulating a depending on the visited configurations $\boldsymbol{\sigma}$ as follows:

$$a(t) = \alpha - \rho \tanh(\delta \Delta \mathcal{H}(t)), \tag{3}$$

where $\Delta\mathcal{H}(t) = \mathcal{H}_{\text{opt}} - \mathcal{H}(t)$; $\mathcal{H}(t)$, the Ising Hamiltonian of the configuration visited at time t; and $\mathcal{H}_{\text{opt}}$, a given target energy. In practice, we set $\mathcal{H}_{\text{opt}}$ to the lowest energy visited during the current run, i.e., $\mathcal{H}_{\text{opt}}(t) = \min_{t' \leq t} \mathcal{H}(t')$. The function tanh is the tangent hyperbolic. ρ and δ are positive real constants. In this way, configurations that have much larger Ising energy than the lowest energy visited are destabilized more strongly due to smaller target amplitude a. Lastly, the parameter ξ (see eq. (2)) is modulated as follows: $\frac{\mathrm{d}\xi}{\mathrm{d}t} = \gamma$ when $t - t_r < \Delta t$, where t_r is the last time for which either the best known energy $\mathcal{H}_{\text{opt}}$ was updated or ξ was reset. Otherwise, ξ is reset to 0 if $t - t_r \geq \Delta t$ and t_r is set to t. Numerical simulations shown in the following suggest that this modulation results in the destabilization of non-trivial attractors (periodic, chaotic, etc.) for typical problem instances.

In order to verify that the nonrelaxational dynamics of chaotic amplitude control is able to accelerate the search of mean-field dynamics for finding the ground state of typical frustrated systems, we look for the ground-states of Sherrington-Kirkpatrick (SK) spin glass instances using the numerical simulation of eqs. (1) to (3) and compare time to solutions with those of two closely related relaxational schemes: noisy mean-field annealing (NMFA)[2] and the simulation of the coherent Ising machine (simCIM) [3]. Because the arithmetic complexity of calculating one step of these three schemes is dominated by the matrix-vector multiplication (MVM), it is sufficient for the sake of comparison to count the number of MVM, noted ν, to find the ground state energy of a given instance with 99% success probability, with $\nu(K) = K \frac{\ln(1-0.99)}{\ln(1-p_0(K))}$ and $p_0(K)$ the probability of visiting a ground state configuration at least once after a number K of MVMs *in a single run*. In Fig. 2, NMFA (a) and the CAC (b) are compared using the averaged success probability $\langle p_0 \rangle$ of finding the ground state for 100 randomly generated SK spin glass instances per problem size N. Note that the success probability of the mean-field annealing method does not seem to converge to 1 even for

[3] See Supplementary Materials S4.1-2

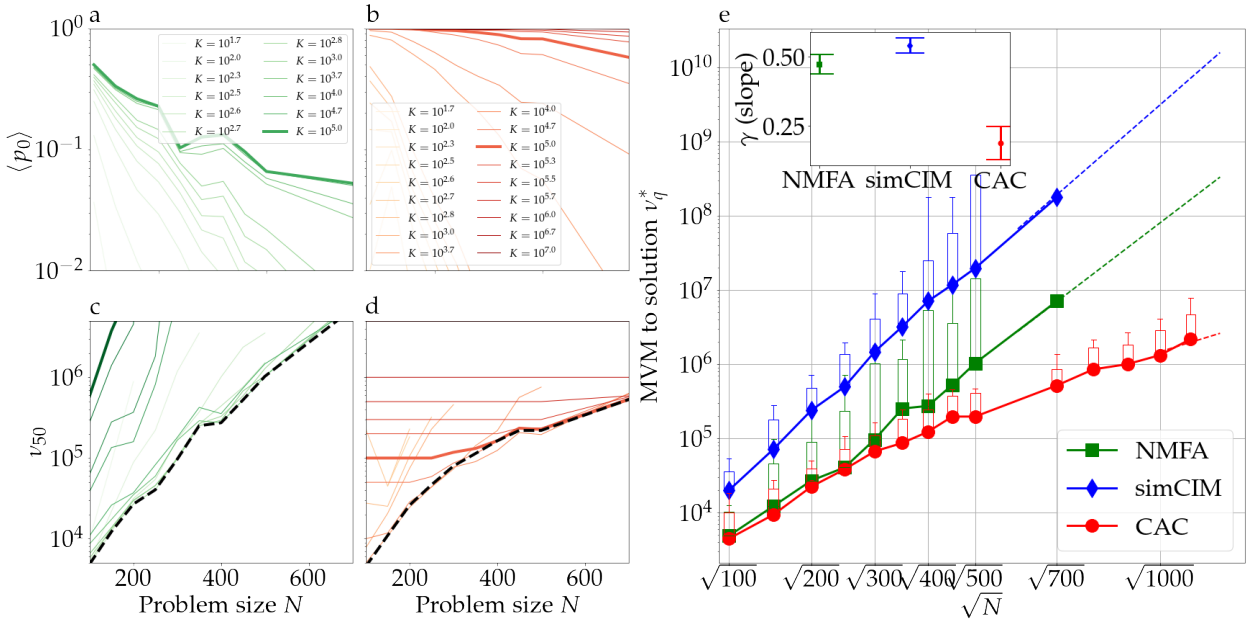

Figure 2: (a,b) Average success probability $\langle p_0 \rangle$ of finding the ground state configuration of 100 Sherrington-Kirkpatrick spin glass instances and (c,d) 50^{th} percentile of the MVM to solution distribution ν_{50} vs. system size N. (a,c) NMFA. (b,d) CAC. (e) Number of matrix-vector multiplication MVM to solution distribution. Lower, higher, and upper whisker of boxes show the $50th$, $80th$, and $90th$ percentiles of the distribution. The upper right inset shows the exponential scaling factor γ of the $50th$ percentile with $\nu_{50} \sim e^{\gamma\sqrt{N}}$ for CAC, NMFA, and simCIM.

large annealing time (see Fig. 2 (a)). Because the success probability of NMFA and simCIM remains low at larger problem size, its correct estimation requires simulating a larger number of runs which we achieved by using GPU implementations of these methods. On the other hand, the average success probability $\langle p_0 \rangle$ of CAC is of order 1 when the maximal number of MVM is large enough, suggesting that the system rarely gets trapped in local minima of the Ising Hamiltonian or non-trivial attractors. In Figure 2 (c) and (d) are shown the q^{th} percentile (with $q = 50$, i.e., the median) of the MVM to solution distribution, noted $\nu_q(K; N)$, for various duration of simulation K, where K is the number of MVMs of a single run. The minimum of these curves, noted $\nu_q^*(N)$ with $\nu_q^*(N) = \min_K \nu_q(K; N)$, represents the optimal scaling of MVM to solution vs. problem size N[15]. Using the hypothesis of an exponential scaling with the square root of problem size N, CAC exhibits significantly smaller scaling exponent ($\gamma = 0.18 \pm 0.06$) than the NMFA ($\gamma = 0.47 \pm 0.04$) and simCIM ($\gamma = 0.54 \pm 0.03$, see inset in Fig. 2 (e)). We have verified that this scaling advantage holds for various parameters of the mean-field annealing (see Supplementary Materials S4.1). Note that a root-exponential scaling of the median time to solution has been previously reported for SK spin glass problems[15, 54] and other NP-Hard problems[55].

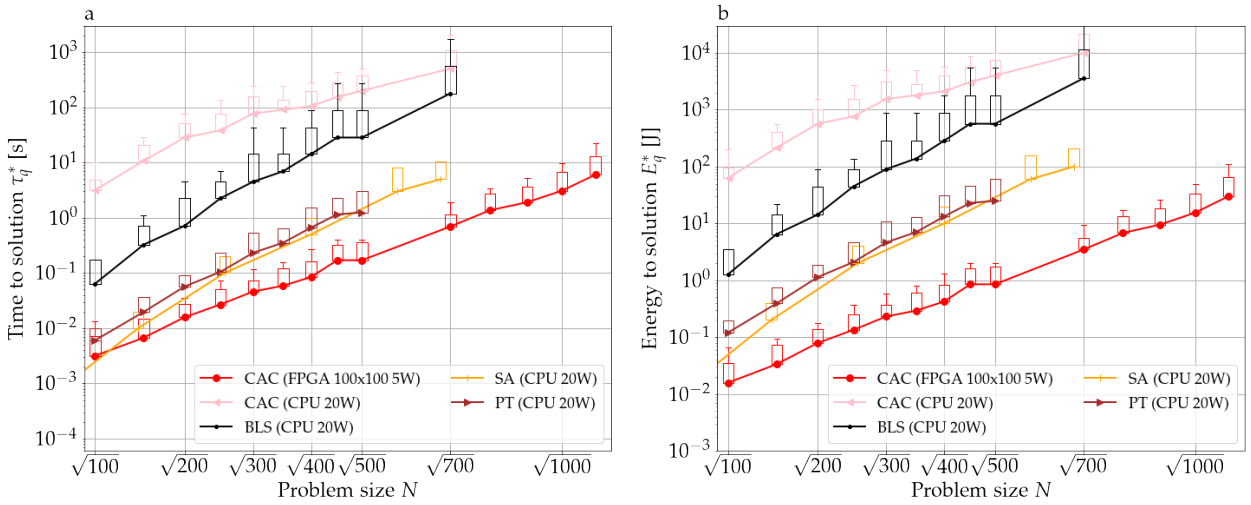

Figure 3: (a) Lower, higher, and upper whisker of boxes show the 50$^{\text{th}}$, 80$^{\text{th}}$, and 90$^{\text{th}}$ percentiles of the time to solution distribution in seconds for the FPGA implementation of CAC with a maximum of 5W power consumption, CAC, SA, and PT algorithm running on a CPU (20W). The dashed and dotted red lines show the predictions of the real time to solution in the case of a 10W and 20W FPGA implementation, respectively. (b) The same as (a) for the energy-to-solution E^*.

Although comparison of algebraic complexity indicates that CAC has a scaling advantage over mean-field annealing, it is in practice necessary to compare its physical implementation against other state-of-the-art methods because the performance of hardware depends on other factors such as memory access and information propagation delays. To this end, CAC is implemented into a FPGA because its similarity with neural networks makes it well-fitted for a design where memory is distributed with processing (see Supplementary Materials S2 for the details of the FPGA implementation). The organization of the electronic circuit can be understood using the following analogy. Pairs of analog values x_i and e_i, which represent averaged activity of two types of neurons, are encoded within neighboring circuits. This micro-structure is repeated N times on the whole surface of the chip which resembles the columnar organization of the brain. The nonlinear processes $f_i(x_i)$, which model the local-population activation functions and are independent for $i \neq j$, are calculated in parallel. The coupling between elements i and $j \in \{1, \ldots, N\}$ that is achieved by the dot product in eq. (1) is implemented by circuits that are at the periphery of the chip and are organized in a summation tree reminiscent of dendritic branching (see Fig. 1 (c)). The power consumption of the developed hardware never exceeds 5W because of limitations of the development board that we have used.

First, we compare the FPGA implementation of CAC against state-of-the-art CPU algorithms: break-out local search[56] (BLS) that has been used to find many of the best known maximum-cuts (equivalently, Ising energies) from the GSET benchmark set[57], a

well-optimized single-core CPU implementation of parallel tempering (or random replica exchange Monte-Carlo Markov chain sampling)[58, 59] (PT, courtesy of S. Mandra), simulated annealing (SA)[60]. Figure 3 (a) shows that the CAC on FPGA has the smallest real time to solution τ_q^* against most other state-of-the-art algorithms despite just 5W power-consumption where $\tau_q^*(N)$ is the optimal q^{th} percentile of time to solution with 99% success probability and is given as $\tau_q^*(N) = \min_T \tau_q(T;N)$ where $\tau(T)$ of a given instance is $\tau(T) = T\frac{\ln(1-0.99)}{\ln(1-p_0(T))}$ and T is the duration in seconds of a run. The probability $p_0(T)$ is evaluated using 100 runs per instance. The results of the CAC algorithm run on a CPU are also included in Fig. 3. The CPU implementation of CAC written in python for this work is not optimized and is consequently slower than other algorithms. However, its scaling of time to solution with problem size is consistent with that of CAC on FPGA. Figure 3 shows that CAC implemented on either CPU or FPGA has a significantly smaller increase of time to solution with problem size than SA run on CPU. Note that the power consumption of transistors in the FPGA and CPU scales proportionally to their clock frequencies. In order to compare different hardware despite the heterogeneity in their power consumption, the q^{th} percentile of energy-to-solution E_q^*, i.e., the energy E_q^* required to solve SK instances with $E_q^* = P\tau_q^*$ and P the power consumption[4], is plotted in Fig. 3 (b). CAC on FPGA is 10^2 to 10^3 times more energy efficient than state-of-the-art algorithms running on classical computers.

The Monte Carlo methods SA and PT have moreover been recently implemented on a special-purpose electronic chip called Digital Annealer (DA)[20]. In Fig. 4, we show the scaling exponents of 50th and 80th percentiles of the time to solution distribution for problem sizes $N = 800$ to $N = 1100$ based on the hypothesis of scaling in $e^{\gamma N}$ obtained by fitting data shown in Figs. 2 and 3 (see "CAC fully parallel" and 'CAC 100×100", respectively, in Fig. 4 for the numerical values of the scaling exponents[5]), and compare them to that reported for SA and PT implemented on CPU and DA. The scaling obtained from fitting the time to solution in Fig. 2 is based on the assumption that the matrix-vector multiplication can be calculated fully in parallel in a time that scales as $\log(N)$ instead of N^2 (see Materials and Methods section) at least up to $N = 1100$. We include this hypothesis because many other Ising machines exploit the parallelization of matrix-vector multiplication for speed up[20, 61], whereas the current implementation of CAC iterates on block matrices of size 100 by 100 and is thus only partially parallel because of resource limitations specific to the downscale FPGA used in this work. Note that the time to compute the matrix-vector multiplication is not the dominant term in the exponential scaling behavior of time to solution at large N. The scaling of time to solution for the nonrelaxational dynamics observed is significantly smaller than the ones of standard Monte Carlo methods SA and PT[20], especially in the case of a fully parallel implementation. The scaling exponents of fully parallel CAC is smaller than that of DA and on par with that of PT on DA (PTDA), although CAC does not require

[4]For the sake of simplicity, we assume a 20 watts power consumptions for the CPU. These numbers represent typical orders to magnitude for contemporary digital systems.

[5]We replicated the benchmark method of [20] for the sake of the comparison by fitting time to solution from $N = 800$ up to $N = 1100$.

simulating replica of the system and is thus faster in absolute time than PTDA[20].

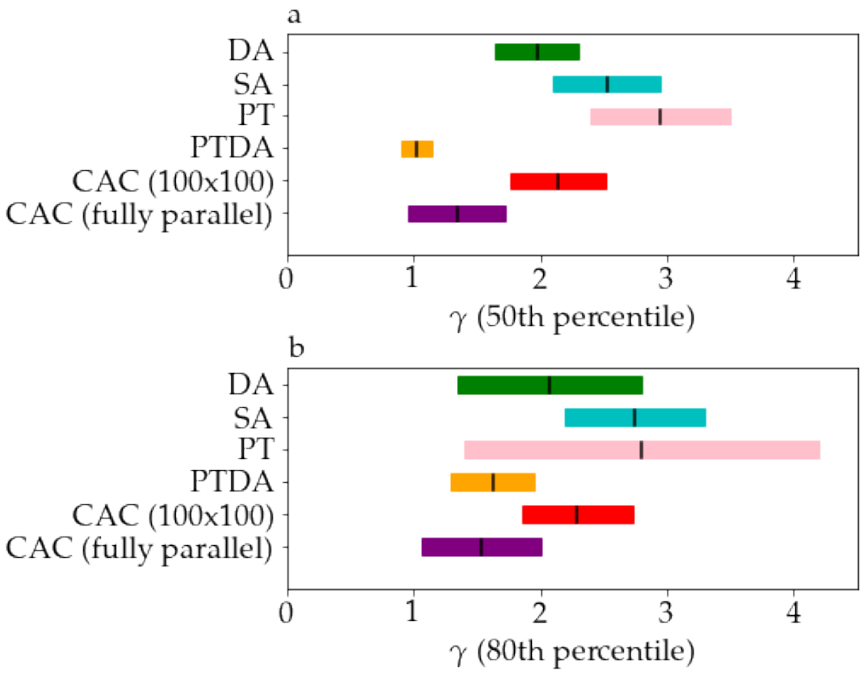

Figure 4: Scaling exponents $\frac{\gamma}{\log(10)}$ of the 50$^{\text{th}}$ (a) and 80$^{\text{th}}$ (b) percentiles of the time to solution distribution based on the hypotheses of scaling in $e^{\gamma N}$ obtained by fitting data shown in Fig. 2 of the proposed nonrelaxational dynamics and the scaling exponents reported in [20]. Colored boxes show the 90% confidence interval in the scaling exponents. SA: simulated annealing; PT: parallel tempering; DA: digital annealer; PTDA: parallel tempering on DA. Exponents of SA, PT, DA, and PTDA are taken from [20].

Next, the proposed implementation of nonrelaxational dynamics is compared to other recently developed Ising machines (see Fig. 5). The relatively slow increase of time to solution with respect to the number of spins N when solving larger SK problems using CAC suggests that our FPGA implementation is faster than the Hopfield neural network implemented using memresistors (mem-HNN)[4], the restricted Boltzmann machine using a FPGA (FPGA-RBM)[54] at large N. Extrapolations are based on the hypotheses of scaling in $e^{\gamma N}$ and $e^{\gamma\sqrt{N}}$ by fitting the available experimental data up to $N = 100$ for mem-HNN and FPGA-RBM, $N = 150$ for NTT CIM, and $N = 1100$ for FPGA-CAC. Figure 5 shows that mem-HNN, FPGA-RBM, and NTT CIM have similar scaling exponents, although FPGA-RBM tends to exhibit a scaling in $e^{\gamma N}$ rather than $e^{\gamma\sqrt{N}}$ for $N \approx 100$[54]. It can be nonetheless expected that the algorithm implemented in mem-HNN, which is similar to mean-field annealing, has the same scaling behavior as simCIM and NMFA (see Fig. 2).

It is noteworthy to mention that a recent implementation of the simulated bifurcation machine[61] (SBM) which is not based on the gradient descent, similarly to CAC, but based on

adiabatic evolutions of energy conservative systems performs well in solving SK problems. Both SBM and CAC exhibit smaller time to solution than other gradient based methods. SBM has been implemented on a FPGA (the Intel Stratix 10 GX) that has approximately 5 to 10 times more adaptive logic modules than the KU040 FPGA used to implement CAC. In order to compare SBM and CAC if implemented on an equivalent FPGA hardware, we plot in Fig. 5 the estimation of the time to solution for a fully parallel implementation of CAC using the hypothesis that one matrix-vector multiplication of size 1100×1100 can be achieved in $0.3\mu s$. This is the same time to compute a MVM that we can infer from time to solution reported in [61] for SBM with binary connectivity given that problems of size $N = 100$ ($N = 700$) are solved in $29\mu s$ ($55ms$) and 94 (81000) MVMs, respectively. Note that SBM can reach the ground-states after approximately 20 times less MVMs than CAC at $N = 100$ but only 5 times less MVM at $N = 1000$, suggesting that the speed of SBM depends largely on hardware rather than an algorithmic scaling advantage. Moreover, the simulated bifurcation machine[61] does not perform significantly better than our current implementation of CAC for solving instances of the reference MAXCUT benchmark set called GSET[57] (see Tab. 1) even compared to the case of the implementation on the smaller KU040 FPGA with the probability of finding maximum cuts of the GSET in a single run that is much smaller with SBM. Comparison of the scaling behavior of time to solution between CAC and SBM is unfortunately not possible based on available data[61].

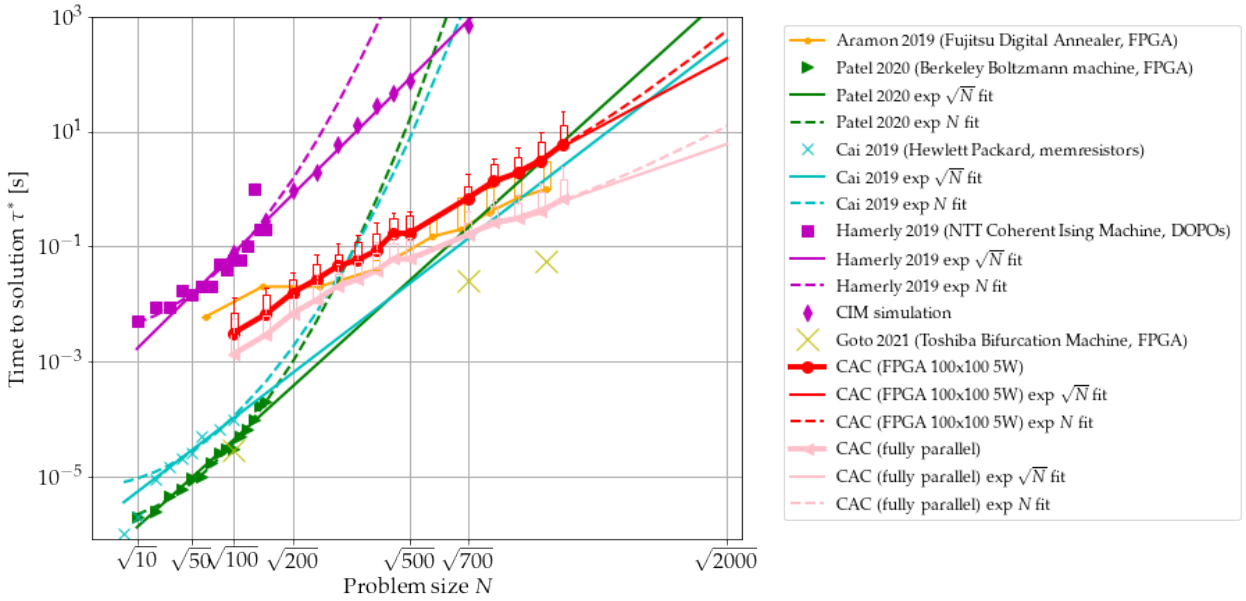

Figure 5: Median time to solution and extrapolations based on the hypotheses of scaling in $e^{\gamma N}$ and $e^{\gamma\sqrt{N}}$ by fitting the available experimental data for each Ising machine. Shaded regions show the 95% confidence interval in the scaling exponents. For FPGA-CAC and DA, the lower, higher, and upper whisker of boxes show the 50^{th}, 80^{th}, and 90^{th} percentiles of the real time to solution distribution.

Lastly, we consider the whole distribution of time to solution in order to compare the ability of

id	N	$C^{\rm opt}$	$C^{\rm CAC}$	$C^{\rm SBM}$	$C^{\rm CAC}-C^{\rm SBM}$	$p_0^{\rm CAC}$	$p_0^{\rm SBM}$	$< t^{\rm BLS} >$ (s)	$< t^{\rm CAC} >$ (s)	$< t^{\rm SBM} >$ (s)
22	2000	13359	13359	13359	0	0.15	0.018	560.00	2.76	**2.7**
23	2000	13344	13342	13342	0	0.25	0.0490	278.00	(4.33)	(**0.94**)
24	2000	13337	13337	13337	0	0.35	0.0077	311.00	**2.48**	3.1
25	2000	13340	13340	13340	0	0.25	0.0022	148.00	**9.49**	15
26	2000	13328	13328	13328	0	0.25	0.0050	429.00	7.14	**3.2**
27	2000	3341	3341	3341	0	1.00	0.0479	449.00	9.96	**0.26**
28	2000	3298	3298	3298	0	0.20	0.0690	432.00	13.86	**0.45**
29	2000	3405	3405	3405	0	0.20	0.0039	17.00	9.90	**1.2**
30	2000	3413	3413	3413	0	0.05	0.0045	283.00	3.12	**2.8**
31	2000	3310	3309	3310	-1	0.25	0.0012	285.00	(14.22)	5.3
32	2000	1410	1410	1410	0	0.05	0.0012	336.00	35.90	**33**
33	2000	1382	1380	1382	-2	0.05	0.0002	402.00	(5.68)	120
34	2000	1384	1384	1384	0	0.05	0.0015	170.00	**1.55**	27
35	2000	7687	7685	7685	0	0.15	0.0002	442.00	(**12.94**)	(479)
36	2000	7680	7679	7677	2	0.20	0.0001	604.00	(19.71)	(1597)
37	2000	7691	7690	7691	-1	0.05	0.0001	444.00	(33.87)	1278
38	2000	7688	7688	7688	0	0.05	0.0003	461.00	**1.94**	213
39	2000	2408	2408	2408	0	0.45	0.0006	251.00	**15.01**	266
40	2000	2400	2398	2400	-2	0.05	0.0001	431.00	(2.45)	48
41	2000	2405	2405	2405	0	0.05	0.0020	73.00	**5.57**	48
42	2000	2481	2481	2479	2	0.50	0.0002	183.00	19.15	(240)

Table 1: Performance of the FPGA implementation of CAC in finding the maximum cuts known, i.e., lowest Ising Hamiltonian known, of graphs in the GSET benchmark. *id*, $C^{\rm opt}$, $C^{\rm CAC}$, $C^{\rm SBM}$ are the name of instances, best maximum cuts known from [62], the proposed method after 20 runs, and Toshiba bifurcation machine on FPGA[61] (FPGA-SBM), respectively. $C^{\rm SBM}$ is evaluated using more than 20 runs. $p_0^{\rm CAC}$ and $p_0^{\rm SBM}$ are the probability that FPGA-CAC and FPGA-SBM find the cut $C^{\rm CAC}$ and $C^{\rm SBM}$ in a single run, respectively. Moreover, $< t^{\rm BLS} >$, $< t^{\rm CAC} >$, and $< t^{\rm SBM} >$ are the averaged time to solution using BLS written C++ and running on a Xeon E5440 2.83 GHz[56], the proposed scheme implemented on the KU040 FPGA, and FPGA-SBM, respectively.

various methods to solve harder instances. As shown in Fig. 6 (a), the cumulative distribution function (CDF) $P(\tau;T)$ of time to solution with 99% success probability τ is not uniquely defined as it depends on the duration T of the runs. We can define an optimal CDF $P^*(\tau)$ that is independent of the runtime T as follows: $P^*(\tau) = \max_T P(\tau;T)$. Numerical simulations show that this optimal CDF is well described by lognormal distribution, that is $P^*(log(\tau)) \sim \mathcal{N}(\mu, \sqrt{v})$ where $\sqrt{v}$ is the standard deviation of $log(\tau)$ (see Figs. 6 (b), (c), and (d) for the cases of CAC, SA, and NMFA, respectively). In Fig. 6 (e), it is shown that the scaling of the standard deviation $\sqrt{v}(N)$ with the problem size N is significantly smaller for CAC, which implies that harder instances can be solved relatively more rapidly than using other methods. This result confirms the advantageous scaling of higher percentiles for CAC that was observed in Figs. 2 and 3.

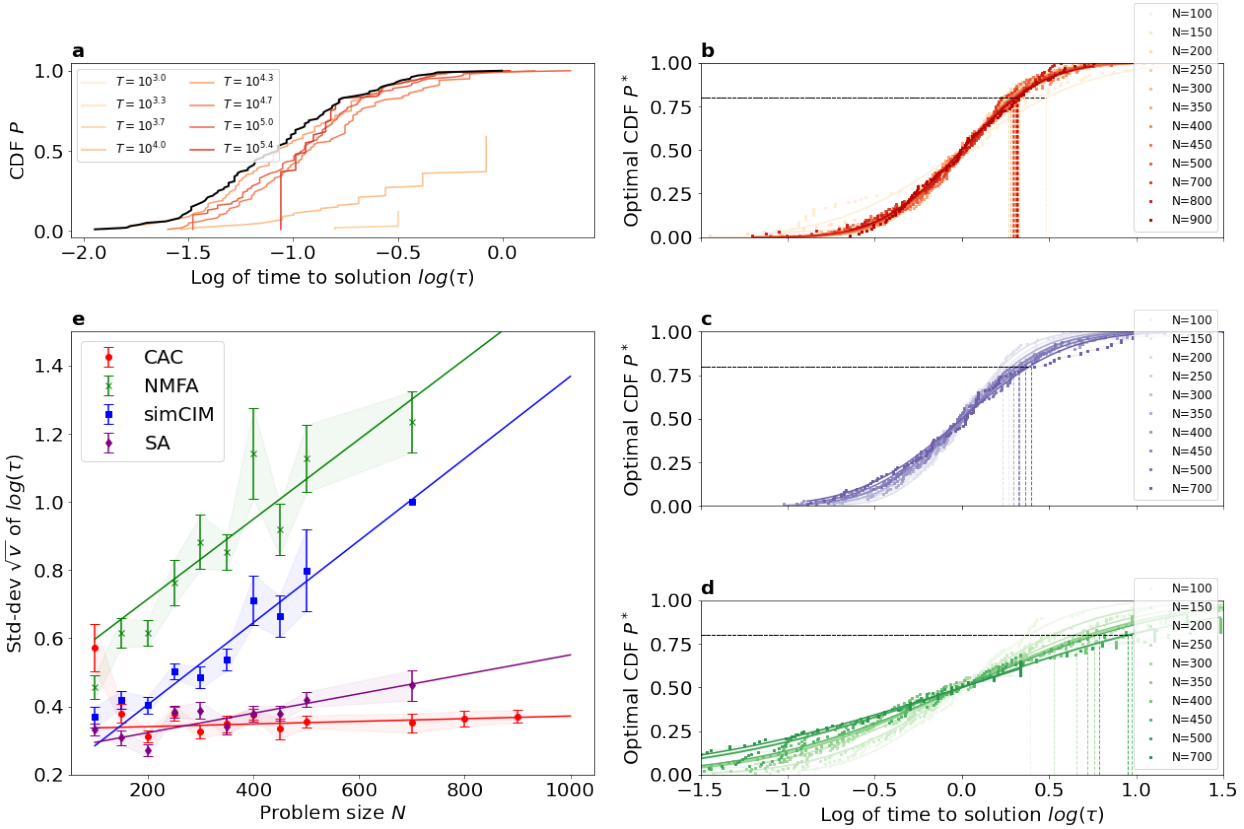

Figure 6: (a) Cumulative distribution of the time to solution $P(\tau)$ for $N = 400$ SK problems. (b,c,d) Optimal cumulative distribution $P^*(\tau)$ with $P^*(log(\tau)) \sim \mathcal{N}(\mu(N), \sqrt{v}(N))$ for CAC (b), SA (c), and NMFA (d), respectively. (e) Standard deviation $\sqrt{v}$ of the logarithm of time to solution distribution vs. problem size N. Shaded regions show the 99% confidence interval in the standard deviation.

Conclusions

The framework described in this paper can be extended to solve other types of constrained combinatorial optimization problems such as the traveling salesman[45], vehicle routing, and lead optimization problems. Moreover, it can be easily adapted to a variety of recently proposed Ising machines[4, 5, 6, 7, 8, 5, 10, 11, 12, 13, 15] which would benefit from implementing a scheme that does not rely solely on the descent of a potential function. In particular, the performance of CIM[5, 10], mem-HNN[4], and chip-scale photonic Ising machine[14], which have small time to solution for small problem sizes[6] ($N \approx 100$) but with a relatively large scaling exponent that limit their scalability, could be significantly improved by adapting the implementation we propose if these hardware can be shown to be able to simulate larger numbers of spins experimentally. Rapid progress in the growing field of Ising machines may

[6]Relaxation dynamics may be faster for solving small problem sizes for which it may be sufficient to do rapid sampling based on convex optimization[63].

allow to verify scaling behaviors of the various methods at larger problem sizes and, thus, limit further finite-size effects.

The scaling exponents we have reported in this paper are based on the integration of the chaotic amplitude control dynamics using a Euler approximation of its ODEs. We have noted that the scaling of MVMs to solution of SK spin glass problems is reduced when the Euler time step is decreased (see Supplementary Materials S2.8). The scaling exponents of CAC might thus be smaller than reported in this paper in the limit of a more accurate numerical integration over continuous time. It is therefore important future work to evaluate the scaling for $N \gg 1000$ using a faster numerical simulation method. Such numerical calculations require a careful analysis of the integration method of ODEs, numerical precision, and tuning of parameters. It is also of considerable interest to implement CAC on an analog physical system for further reduction of power consumption.

Nonrelaxational dynamics described herein is not limited to artificial simulators and likely also emerge in natural complex systems. In particular, it has been hypothesized that the brain operates out of equilibrium at large scales and produces more entropy when performing physically and cognitively demanding tasks[64, 65] in particular for the ones involving creativity for maximizing reward rather than memory retrieval. Such neural processes cannot be explained simply by the relaxation to fixed point attractors[1], periodic limit cycles[66], or even low-dimensional chaotic attractors whose self-similarity may not be equivalent to complexity but set the conditions for its emergence[67, 65]. Similarly, evolutionary dynamics that is characterized as non-ergodic when the time required for representative genome exploration is longer than available evolutionary time[68] may benefit from nonrelaxational dynamics rather than slow glassy relaxation for faster identification of high-fitness solutions. A detailed analytic comparison between the slow relaxation dynamics observed in Monte Carlo simulations of spin glasses with the one proposed in this paper is needed in order to explain the apparent difference in their scaling of time to solution exhibited by our numerical results.

Materials and Methods

We target the implementation of a low-power system with maximum power supply of 5W using a XCKU040 Kintex Ultrascale Xilinx FPGA integrated on an Avnet board. The implemented circuit can process Ising problems of up to at least 1100 spins fully connected and of more than 2000 spins sparsely connected within the 5W power supply limit. Data are encoded into 18 bits fixed point vectors with 1 sign, 5 integer and 12 decimal bits to optimize computation time and power consumption. An important feature of our FPGA implementation of CAC is the use of several clock frequencies to concentrate the electrical power on the circuits that are the bottleneck of computation and require high speed clock. For the realization of the matrix-vector multiplication, each element of the matrix is encoded with 2 bits precision (w_{ij} is -1, 0 or 1). An approximation based on the combination of logic equations describing the behavior of a multiplexer allows to achieve 10^4 multiplications

within one clock cycle. The results of these multiplications are summed using cascading DSP and CARRY8 connected in a tree structure. Using pipelining, a matrix-vector multiplication for a squared matrix of size N is computed in $2 + 5\frac{\log(N-4)}{\log(5)} + (\frac{N}{u})^2$ clock cycles (see Supplementary Materials S2.4) at a clock frequency of 50MHz with $u = 100$ which is determined by the limitation of the number of available electronic component of the XCKU040 FPGA. The block size u can be made at least 3 times larger using commercially available FPGAs, which implies that the number of clock cycles needed to compute a dot product can scale almost logarithmically for problems of size $N \approx 1000$ (see Supplementary Materials S2.4 for discussions of scalability) and that the calculation time can be further significantly decreased using a higher-end FPGA. The calculation of the nonlinearity f_i and error terms is achieved at higher frequency (300MHz and 100Mhz) using DSP in $8 + (N/u)$ and $9 + (N/u)$ clock cycles, respectively. In order to minimize energy resources and maximize speed, the nonlinear and error terms are calculated multiple times during the calculation of a single matrix-vector multiplication (see Supplementary Materials S2).

Prior to computing the benchmark on the Sherrington-Kirkpatrick instances, the parameters of the system (see Supplementary Materials S2.8) are optimized automatically using Bayesian optimization and bandit-based methods[69]. The automatic tuning of parameters for some previously unseen instances is out of the scope of this work but can be achieved to some extent using machine learning techniques[70]. Sherrington-Kirkpatrick instances used in this paper are available upon request. The GSET instances are available at https://web.stanford.edu/ yyye/yyye/Gset/.

Acknowledgments

The authors thank Salvotare Mandra for providing results of the PT algorithm. This research is partially supported by the Brain-Morphic AI to Resolve Social Issues (BMAI) project (NEC corporation), AMED under Grant Number JP20dm0307009, UTokyo Center for Integrative Science of Human Behavior(CiSHuB), and Japan Science and Technology Agency Moonshot R&D Grant Number JPMJMS2021

References

[1] Parisi, G., Mézard, M. & Virasoro, M. Spin glass theory and beyond. *World Scientific, Singapore* **187**, 202 (1987).

[2] Kochenberger, G. *et al.* The unconstrained binary quadratic programming problem: a survey. *Journal of Combinatorial Optimization* **28**, 58–81 (2014).

[3] Vadlamani, S. K., Xiao, T. P. & Yablonovitch, E. Physics successfully implements lagrange multiplier optimization. *Proceedings of the National Academy of Sciences* **117**, 26639–26650 (2020).

[4] Cai, F. *et al.* Power-efficient combinatorial optimization using intrinsic noise in memristor hopfield neural networks. *Nature Electronics* 1–10 (2020).

[5] Mahboob, I., Okamoto, H. & Yamaguchi, H. An electromechanical ising hamiltonian. *Science advances* **2**, e1600236 (2016).

[6] Camsari, K. Y., Faria, R., Sutton, B. M. & Datta, S. Stochastic p-bits for invertible logic. *Physical Review X* **7**, 031014 (2017).

[7] Camsari, K. Y., Sutton, B. M. & Datta, S. p-bits for probabilistic spin logic. *Applied Physics Reviews* **6**, 011305 (2019).

[8] Marandi, A., Wang, Z., Takata, K., Byer, R. L. & Yamamoto, Y. Network of time-multiplexed optical parametric oscillators as a coherent Ising machine. *Nature Photonics* **8**, 937–942 (2014). URL http://www.nature.com/doifinder/10.1038/nphoton.2014.249.

[9] McMahon, P. L. *et al.* A fully programmable 100-spin coherent Ising machine with all-to-all connections. *Science* **354**, 614–617 (2016). URL http://www.sciencemag.org/lookup/doi/10.1126/science.aah5178.

[10] Inagaki, T. *et al.* Large-scale Ising spin network based on degenerate optical parametric oscillators. *Nature Photonics* **10**, 415–419 (2016). URL http://www.nature.com/doifinder/10.1038/nphoton.2016.68.

[11] Pierangeli, D., Marcucci, G. & Conti, C. Large-scale photonic ising machine by spatial light modulation. *Physical Review Letters* **122**, 213902 (2019).

[12] Roques-Carmes, C. *et al.* Heuristic recurrent algorithms for photonic ising machines. *Nature communications* **11**, 1–8 (2020).

[13] Prabhu, M. *et al.* Accelerating recurrent ising machines in photonic integrated circuits. *Optica* **7**, 551–558 (2020).

[14] Okawachi, Y. *et al.* Demonstration of chip-based coupled degenerate optical parametric oscillators for realizing a nanophotonic spin-glass. *Nature Communications* **11**, 1–7 (2020).

[15] Hamerly, R. *et al.* Experimental investigation of performance differences between coherent ising machines and a quantum annealer. *Science advances* **5**, eaau0823 (2019).

[16] Kalinin, K. P. & Berloff, N. G. Global optimization of spin hamiltonians with gain-dissipative systems. *Scientific reports* **8**, 17791 (2018).

[17] Johnson, M. W. *et al.* Quantum annealing with manufactured spins. *Nature* **473**, 194 (2011).

[18] Kumar, S., Strachan, J. P. & Williams, R. S. Chaotic dynamics in nanoscale nbo 2 mott memristors for analogue computing. *Nature* **548**, 318 (2017).

[19] Heim, B., Rønnow, T. F., Isakov, S. V. & Troyer, M. Quantum versus classical annealing of ising spin glasses. *Science* **348**, 215–217 (2015).

[20] Aramon, M. *et al.* Physics-inspired optimization for quadratic unconstrained problems using a digital annealer. *Frontiers in Physics* **7**, 48 (2019).

[21] Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. Optimization by simulated annealing. *Science* **220**, 671–680 (1983).

[22] King, A. D., Bernoudy, W., King, J., Berkley, A. J. & Lanting, T. Emulating the coherent ising machine with a mean-field algorithm. *arXiv preprint arXiv:1806.08422* (2018).

[23] Bilbro, G. *et al.* Optimization by mean field annealing. In *Advances in neural information processing systems*, 91–98 (1989).

[24] Chen, L. & Aihara, K. Chaotic simulated annealing by a neural network model with transient chaos. *Neural Netw.* **8**, 915–930 (1995).

[25] Kadowaki, T. & Nishimori, H. Quantum annealing in the transverse ising model. *Phys. Rev. E* **58**, 5355 (1998).

[26] Tanaka, F. & Edwards, S. Analytic theory of the ground state properties of a spin glass. i. ising spin glass. *Journal of Physics F: Metal Physics* **10**, 2769 (1980).

[27] Biroli, G. Dynamical tap approach to mean field glassy systems. *Journal of Physics A: Mathematical and General* **32**, 8365 (1999).

[28] Bernaschi, M., Billoire, A., Maiorano, A., Parisi, G. & Ricci-Tersenghi, F. Strong ergodicity breaking in aging of mean-field spin glasses. *Proceedings of the National Academy of Sciences* **117**, 17522–17527 (2020).

[29] Cugliandolo, L. F. & Kurchan, J. On the out-of-equilibrium relaxation of the sherrington-kirkpatrick model. *Journal of Physics A: Mathematical and General* **27**, 5749 (1994).

[30] Leleu, T., Yamamoto, Y., McMahon, P. L. & Aihara, K. Destabilization of local minima in analog spin systems by correction of amplitude heterogeneity. *Physical review letters* **122**, 040607 (2019).

[31] Ercsey-Ravasz, M. & Toroczkai, Z. Optimization hardness as transient chaos in an analog approach to constraint satisfaction. *Nat. Phys.* **7**, 966–970 (2011). URL http://www.nature.com/doifinder/10.1038/nphys2105.

[32] Molnár, B., Molnár, F., Varga, M., Toroczkai, Z. & Ercsey-Ravasz, M. A continuous-time maxsat solver with high analog performance. *Nature communications* **9**, 4864 (2018).

[33] Aspelmeier, T. & Moore, M. Realizable solutions of the thouless-anderson-palmer equations. *Physical Review E* **100**, 032127 (2019).

[34] Boettcher, S. & Percus, A. G. Optimization with extremal dynamics. *Phys. Rev. Lett.* **86**, 5211–5214 (2001). URL https://link.aps.org/doi/10.1103/PhysRevLett.86.5211.

[35] Zarand, G., Pazmandi, F., Pal, K. & Zimanyi, G. Using hysteresis for optimization. *Physical review letters* **89**, 150201 (2002).

[36] Leleu, T., Yamamoto, Y., Utsunomiya, S. & Aihara, K. Combinatorial optimization using dynamical phase transitions in driven-dissipative systems. *Phys. Rev. E* **95**, 022118 (2017). URL https://link.aps.org/doi/10.1103/PhysRevE.95.022118.

[37] Aspelmeier, T., Blythe, R., Bray, A. J. & Moore, M. A. Free-energy landscapes, dynamics, and the edge of chaos in mean-field models of spin glasses. *Physical Review B* **74**, 184411 (2006).

[38] Hasegawa, M., Ikeguchi, T. & Aihara, K. Combination of chaotic neurodynamics with the 2-opt algorithm to solve traveling salesman problems. *Phys. Rev. Lett.* **79**, 2344 (1997).

[39] Horio, Y. & Aihara, K. Analog computation through high-dimensional physical chaotic neuro-dynamics. *Physica D: Nonlinear Phenomena* **237**, 1215–1225 (2008).

[40] Aihara, K. Chaos engineering and its application to parallel distributed processing with chaotic neural networks. *Proceedings of the IEEE* **90**, 919–930 (2002).

[41] Montanari, A. Optimization of the sherrington-kirkpatrick hamiltonian. *arXiv preprint arXiv:1812.10897* (2018).

[42] Furber, S. B., Galluppi, F., Temple, S. & Plana, L. A. The spinnaker project. *Proceedings of the IEEE* **102**, 652–665 (2014).

[43] Davies, M. *et al.* Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* **38**, 82–99 (2018).

[44] Benjamin, B. V. *et al.* Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE* **102**, 699–716 (2014).

[45] Hopfield, J. J. & Tank, D. W. "neural" computation of decisions in optimization problems. *Biol. Cybern.* **52**, 141–152 (1985).

[46] Wang, Z., Marandi, A., Wen, K., Byer, R. L. & Yamamoto, Y. Coherent ising machine based on degenerate optical parametric oscillators. *Physical Review A* **88**, 063853 (2013).

[47] Sompolinsky, H. & Zippelius, A. Relaxational dynamics of the edwards-anderson model and the mean-field theory of spin-glasses. *Physical Review B* **25**, 6860 (1982).

[48] Thouless, D. J., Anderson, P. W. & Palmer, R. G. Solution of'solvable model of a spin glass'. *Philosophical Magazine* **35**, 593–601 (1977).

[49] Hopfield, J. J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences* **81**, 3088–3092 (1984).

[50] Farhi, E., Goldstone, J., Gutmann, S. & Sipser, M. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106* (2000).

[51] Granville, V., Krivánek, M. & Rasson, J.-P. Simulated annealing: A proof of convergence. *IEEE transactions on pattern analysis and machine intelligence* **16**, 652–656 (1994).

[52] Geman, S. & Hwang, C.-R. Diffusions for global optimization. *SIAM Journal on Control and Optimization* **24**, 1031–1043 (1986).

[53] Goto, H. Bifurcation-based adiabatic quantum computation with a nonlinear oscillator network. *Scientific reports* **6**, 21686 (2016).

[54] Patel, S., Chen, L., Canoza, P. & Salahuddin, S. Ising model optimization problems on a fpga accelerated restricted boltzmann machine. *arXiv preprint arXiv:2008.04436* (2020).

[55] Hoos, H. H. & Stützle, T. On the empirical scaling of run-time for finding optimal solutions to the travelling salesman problem. *European Journal of Operational Research* **238**, 87–94 (2014).

[56] Benlic, U. & Hao, J.-K. Breakout local search for the max-cutproblem. *Eng. Appl. Artif. Intell.* **26**, 1162–1173 (2013).

[57] https://web.stanford.edu/~yyye/yyye/Gset/.

[58] Hukushima, K. & Nemoto, K. Exchange monte carlo method and application to spin glass simulations. *Journal of the Physical Society of Japan* **65**, 1604–1608 (1996).

[59] Mandra, S., Villalonga, B., Boixo, S., Katzgraber, H. & Rieffel, E. State-of-the-art classical tools to benchmark nisq devices. In *APS Meeting Abstracts* (2019).

[60] Isakov, S. V., Zintchenko, I. N., Rønnow, T. F. & Troyer, M. Optimised simulated annealing for ising spin glasses. *Computer Physics Communications* **192**, 265–271 (2015).

[61] Goto, H. *et al.* High-performance combinatorial optimization based on classical mechanics. *Science Advances* **7**, eabe7953 (2021).

[62] Ma, F. & Hao, J.-K. A multiple search operator heuristic for the max-k-cut problem. *Annals of Operations Research* **248**, 365–403 (2017).

[63] Ma, Y.-A., Chen, Y., Jin, C., Flammarion, N. & Jordan, M. I. Sampling can be faster than optimization. *Proceedings of the National Academy of Sciences* **116**, 20881–20885 (2019).

[64] Lynn, C. W., Papadopoulos, L., Kahn, A. E. & Bassett, D. S. Human information processing in complex networks. *Nature Physics* **16**, 965–973 (2020).

[65] Wolf, Y. I., Katsnelson, M. I. & Koonin, E. V. Physical foundations of biological complexity. *Proceedings of the National Academy of Sciences* **115**, E8678–E8687 (2018).

[66] Yan, H. *et al.* Nonequilibrium landscape theory of neural networks. *Proceedings of the National Academy of Sciences* **110**, E4185–E4194 (2013).

[67] Ageev, D., Aref'eva, I., Bagrov, A. & Katsnelson, M. I. Holographic local quench and effective complexity. *Journal of High Energy Physics* **2018**, 71 (2018).

[68] McLeish, T. C. Are there ergodic limits to evolution? ergodic exploration of genome space and convergence. *Interface Focus* **5**, 20150041 (2015).

[69] Falkner, S., Klein, A. & Hutter, F. Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, 1437–1446 (PMLR, 2018).

[70] Dunning, I., Gupta, S. & Silberholz, J. What works best when? a systematic evaluation of heuristics for max-cut and qubo. *INFORMS Journal on Computing* **30**, 608–624 (2018).

Supplementary materials: Scaling advantage of nonrelaxational dynamics for high-performance combinatorial optimization

1 Theoretical analysis

1.1 Derivation of a potential function

First, we analyze the system described by eq. (1) only, when $\sigma_0 = 0$ and $\beta_i = \beta$, $\forall i$. In this case, the potential function $V(\boldsymbol{y}) = -\frac{1}{2}\beta \sum_{ij} \omega_{ij} y_i y_j - \sum_i \int_0^{y_i} f_i(g_i^{-1}(y))\,\mathrm{d}y$ has the following property[1]:

$$\frac{\mathrm{d}V}{\mathrm{d}t} = -\sum_i \frac{\mathrm{d}y_i}{\mathrm{d}t}(\beta \sum_j \omega_{ij} y_j + f_i(g_i^{-1}(y_i))), \tag{S1}$$

$$= -\sum_i \frac{\mathrm{d}y_i}{\mathrm{d}t}(\frac{\mathrm{d}x_i}{\mathrm{d}t}), \tag{S2}$$

$$= -\sum_i (\frac{\mathrm{d}y_i}{\mathrm{d}t})^2 (g^{-1})'(y_i). \tag{S3}$$

Consequently, V is such that $\frac{\mathrm{d}V}{\mathrm{d}t} < 0$ because g is strictly monotonic and $\frac{\mathrm{d}V}{\mathrm{d}t} = 0 \implies \frac{\mathrm{d}y_i}{\mathrm{d}t} = 0$, $\forall i$. In other words, the dynamics of the system can be understood in this case as the gradient descent on the potential function V and its stable steady states correspond to local minima of V.

1.2 Analysis of CAC

Next, we analyze the system described in eqs. (1) and (2) (a is constant) by considering the change of variable $y_i = g(x_i)$ with g such that $g^{-1}(y_i) = x_i$. In this case, eqs. (1) and (2) can be rewritten as follows (note that f and g are odd functions):

$$\phi'(y_i)\frac{\mathrm{d}y_i}{\mathrm{d}t} = f(g^{-1}(y_i)) + \beta e_i \sum_j \omega_{ij} y_j, \tag{S4}$$

$$\frac{\mathrm{d}e_i}{\mathrm{d}t} = \xi(y_i^2 - a)e_i, \tag{S5}$$

where $\phi(y) = g^{-1}(y)$.

We analyze the dimension of the unstable manifold of CAC by linearizing the dynamics near the steady states and calculating the real part of eigenvalues of the corresponding Jacobian matrices. The steady state of the system in eqs. (S4) and (S5) can be written as follows:

$$y_i^* = \sigma_i \sqrt{a}, \tag{S6}$$

$$e_i^* = -\frac{f(g^{-1}(\sigma_i\sqrt{a}))}{\beta\sqrt{a}h_i} = -\frac{f(g^{-1}(\sqrt{a}))}{\beta\sqrt{a}\sigma_i h_i}. \tag{S7}$$

with $h_i = \sum_j \omega_{ij}\sigma_j$. The last equality is a consequence of the fact that g, and thus also g^{-1}, are odd functions.

The Jacobian matrix J corresponding to the system of eqs. (1) and (2) at the steady state can be written in the block representation $J = [J^{yy} J^{ye}; J^{ey} J^{ee}]$ with its components given as follows:

$$J_{ij}^{yy} = \psi'(\sqrt{a})\delta_{ij} - \frac{\psi(\sqrt{a})}{\sqrt{a}}\frac{\omega_{ij}}{h_i\sigma_i}, \tag{S8}$$

$$J_{ij}^{ye} = \frac{\beta\sqrt{a}}{\phi'(\sqrt{a})}h_i\delta_{ij}, \tag{S9}$$

$$J_{ij}^{ey} = \frac{-2\xi f(g^{-1}(\sqrt{a}))}{\beta h_i}\delta_{ij}, \tag{S10}$$

$$J_{ij}^{ee} = 0. \tag{S11}$$

with $\psi(y) = \frac{f(g^{-1}(y))}{\phi'(y)}$ and $\phi(y) = g^{-1}(y)$. Note that we define $J_{ii}^{ey} J_{jj}^{ye} = b$ with $b = -2\xi\sqrt{a}\psi(\sqrt{a})$.

The eigenvalues of the Jacobian matrix J are solutions of the polynomial equation $P(\lambda) = \det[J - \lambda I] = \det[(J^{yy} - \lambda I)\lambda I - bI]$. The eigenvalues of J are thus solutions of the quadratic equation $z(z - \lambda_i) - b = 0$ where λ_i is the i^{th} eigenvalue of the matrix J^{yy}. Therefore, the eigenvalues of J can be described by pairs λ_i^+ and λ_i^- given as follows:

$$\lambda_i^{\pm} = \frac{1}{2}(\lambda_i \pm \sqrt{\Delta_i}), \text{ with} \tag{S12}$$

$$\lambda_i = \psi'(\sqrt{a}) - \frac{\psi(\sqrt{a})}{\sqrt{a}}\mu_i, \text{ and} \tag{S13}$$

$$\Delta_i = \lambda_i^2 + 4b, \tag{S14}$$

where μ_i is the i^{th} eigenvalue of the matrix $D[(\boldsymbol{\sigma h})^{-1}]\Omega$.

The eigenvalues λ_i^+ and λ_i^- become complex conjugate when the $\Delta_i = 0$, i.e., $[\psi'(\sqrt{a}) - \frac{\psi(\sqrt{a})}{\sqrt{a}}\mu_i]^2 - 8\xi\sqrt{a}\psi(\sqrt{a})) = 0$, which can be rewritten as follows:

$$\mu_i = G_\xi^{\pm}(\sqrt{a}), \tag{S15}$$

with $G_\xi^{\pm}(y) = \frac{y}{\psi(y)}[\psi'(y) \pm 2\sqrt{2\xi|y|\psi(|y|)}]$.

Lastly, the real part of eigenvalues λ_i^+ become equal to zero at the condition given as follows (note that $\mathrm{Re}[\lambda_i^+] \geq \mathrm{Re}[\lambda_i^-]$ and μ_i are real at local minima[7] for which, $\forall j$, $\sigma_j h_j > 0$):

$$\sqrt{a} = 0 \text{ or } \psi(\sqrt{a}) = 0 \text{ if } \Delta_i \leq 0, \tag{S16}$$

$$\mu_i = F(\sqrt{a}) \text{ otherwise.} \tag{S17}$$

with $F(y) = \frac{\psi'(y)}{\psi(y)}y$. The dimension of the unstable manifold at a given fixed point is then given by the number of indices i for which $\mathrm{Re}[\lambda_i^{\pm}]$ is positive. To illustrate the destabilization of the ground state configuration of an Ising problem, we show in Fig. S1 the dynamics of CAC when solving a problem of size $N = 15$ spins when the state encoding for the ground state configuration unstable for two different examples of functions f and g defining eqs. (1) and (2). The first set of functions f and g with $f(x) = (-1 + p)x - x^3$ and $g(x) = x$ shown in Fig. S1 (a,b,c,d) corresponds to the soft spin model (or simulation of CIM) with chaotic amplitude control whereas the second one (e,f,g,h) with $f(x) = -x + \tanh[0.99x]$ and $g(x) = \tanh[x]$ corresponds to an Hopfield neural network with amplitude heterogeneity error correction. These two figures show that the stability of a given local minima of the Ising Hamiltonian depends on the value of the target amplitude a as predicted in eq. (S17).

[7] Because the vector $\boldsymbol{\sigma} \cdot \boldsymbol{h}$ has positive components at local minima, the eigenvalues of $D[(\boldsymbol{\sigma} \cdot \boldsymbol{h})^{-1}]\Omega$ are the same as the ones of $D[(\boldsymbol{\sigma} \cdot \boldsymbol{h})^{-1}]^{\frac{1}{2}}\Omega D[(\boldsymbol{\sigma} \cdot \boldsymbol{h})^{-1}]^{\frac{1}{2}}$ (Sylvester's law of inertia), which is a symmetric real matrix. Thus, the eigenvalues μ_j are always real.

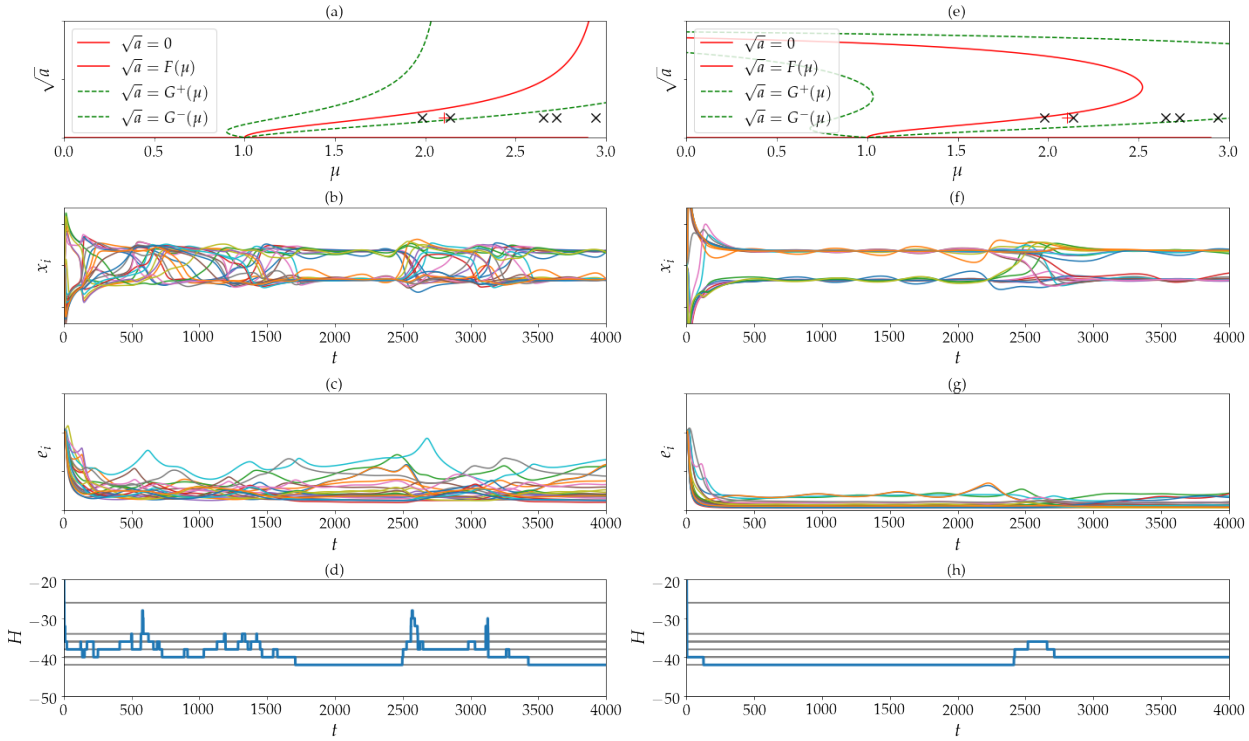

Figure S1: (a,e) Bifurcation diagram in the space $\{\sqrt{a}, \mu = \mu_j(\boldsymbol{\sigma})\}$, $\forall j \in \{1, \ldots, N\}$, at configuration $\boldsymbol{\sigma}$. The full red and green dotted lines correspond to the set for which $Re[\lambda_i^{\pm}] = 0$ (where fixed points corresponding to local minima become stable) and $\Delta_i = 0$ (where oscillations start to occur around fixed points). Red + and black $\times$ symbols correspond to the largest eigenvalues $\mu_0(\boldsymbol{\sigma})$ of the Jacobian at the ground state and excited states, respectively, of an Ising problem of size $N = 15$ and $a = 0.03$. In (b,f) and (c,g) are shown the dynamics of the variables $\boldsymbol{x}$ and $\boldsymbol{e}$, respectively. In (d,h) are shown the Ising Hamiltonian $\mathcal{H}(t)$ with horizontal lines showing the values of the Ising Hamiltonian at local minima. (a,b,c,d) $f(x) = (-1 + p)x - x^3$ and $g(x) = x$ with $p = 0.95$, $\beta = 0.1$, $\xi = 0.1$. (e,f,g,h) $f(x) = -x + \tanh[0.99x]$ and $g(x) = \tanh[x]$ with $\beta = 0.1$, $\xi = 0.1$.

2 FPGA implementation

2.1 FPGA implementation of CAC

CAC has been implemented on a XCKU040 Xilinx FPGA integrated into the KU040 development board designed by Avnet. The circuit can process Ising problems of more than 2000 spins. The reconfigurable chip is a regular Ultrascale FPGA including 484,800 flip-flops, 242,400 Look Up Table (LUT), 12,000 Digital Signal Processing (DSP) and 600 Block RAM (BRAM). Initial value for x_i, e_i and ω_{ij} are sent through Universal Asynchronous Receiver Transmitter (UART) transmission. UART protocol has been chosen because it does not require a lot of resources to be implemented and its simplicity.

Data are encoded into 18 bits fixed point with 1 sign bit, 5 integer bits which represents a good compromise between accuracy and power consumption. The power consumption of the FPGA is equal or lower to 5W depending on the problem size. The system is defined by its top-level entity that represents the highest level of organization of the reconfigurable chip.

2.2 Top level entity

The circuit is organized into four principal building blocks as shown on Fig. S2 (a). Several clock frequencies have been generated to concentrate the electrical power on the circuits that need high speed clock when these circuits constitute a bottleneck in the current implementation. Finally, the organization of the main circuit is represented in Fig. S2 (b). The control block is composed of Finite State Machines (FSM) and is well tuned to pilot all computation core, the Random-Access Memory (RAM), and data flowing between computation cores and RAM. All circuits are synchronized although the system utilizes multiple clock frequencies. Clock Enable (CE) ports on the BUFGCE component are used to stop the power when a circuit is not used in order to reduce further the global power dissipation.

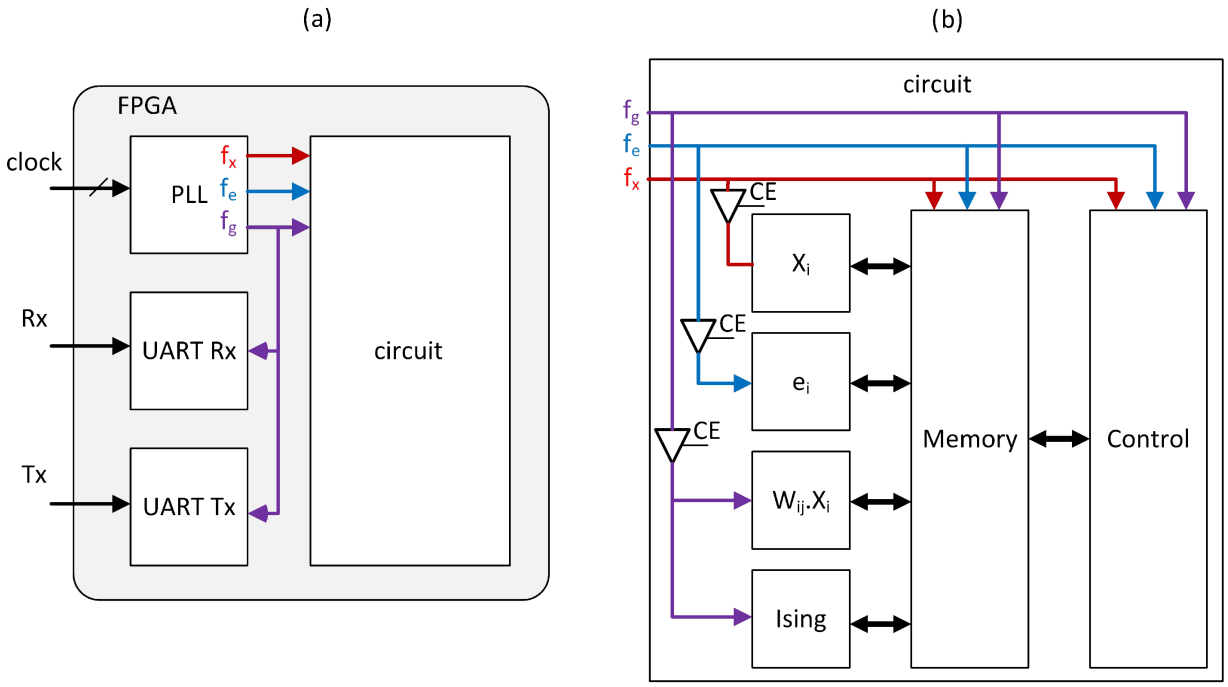

Figure S2: Organization of the FPGA circuit. (a) The top layer entity is divided into four modules: The Phased-Locked Loop (PLL) that convert a differential clock of 250MHz into three clocks f_g=50MHz, f_x=300Mhz and f_e=100MHz respectively representing the global clock, the clock for x_i and the clock for e_i. Two UART modules are used to receives parameters and to return the results of the computation. (b) The Max-Cut circuit is composed of several processes aiming to control the exchange of data between the memory and the computation cores (x_i, e_i, $x_i\omega_{ij}$ and Ising). The three generated clocks are controlled by the Clock Enable (CE) port of the BUFGCE component of the FPGA.

2.3 Temporal organization of the computation

The control circuit pilot the computation in order to calculate several steps of x_i and e_i per calculation of the Ising coupling. The dynamics of CAC (see eqs. (1), (2) and (3)) is implemented based on the following pseudocode:

Algorithm Pseudocode from which the FPGA implementation is adapted

for $\nu \in \{0, \cdots, K\}$ **do** ▷ Iterate on the number of MVMs
  $\boldsymbol{x}' \leftarrow \boldsymbol{x}$ ▷ Save state
  $\boldsymbol{I} \leftarrow \epsilon \boldsymbol{e}(\Omega \boldsymbol{x}')$ ▷ calculation of injection term
  $\boldsymbol{\sigma} \leftarrow \frac{x_i'}{|x_i'|}$
  $H \leftarrow -\frac{1}{2}\boldsymbol{\sigma}(\Omega \boldsymbol{\sigma})$ ▷ Ising energy calculation
  for $i \in \{0, \cdots, n_x\}$ **do** ▷ Update nonlinear terms
    $\boldsymbol{\Delta}_x \leftarrow (-1+p)\boldsymbol{x} - (\boldsymbol{x})^3 + \boldsymbol{I}$
    $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{\Delta}_x dt_x$
  for $i \in \{0, \cdots, n_y\}$ **do** ▷ Update error terms
    $\boldsymbol{\Delta}_e \leftarrow -\beta((\boldsymbol{x}')^2 - a)\boldsymbol{e}$
    $\boldsymbol{e} \leftarrow \boldsymbol{e} + \boldsymbol{\Delta}_e dt_e$
  $\beta \leftarrow \beta + \lambda dt$ ▷ Update β
  $\Delta H \leftarrow H - H_{\text{opt}}$ ▷ Update a
  $a \leftarrow \alpha + \rho \tanh(\delta \Delta H)$
  if $\nu - \nu_c > \tau/dt$ **then** ▷ Reset of β
    $\nu_c \leftarrow \nu$
    $\beta \leftarrow 0$
  if $H < H_{\text{opt}}$ **then** ▷ Update optimal H
    $H_{\text{opt}} \leftarrow H$
    $\nu_{\text{opt}} \leftarrow \nu$
    $\nu_c \leftarrow \nu$

Note that the order of operations described in the pseudocode are a simplification of the ones occurring on the FPGA. The CPU implementation of CAC used in this work is written in python. For the python simulation, variables e_i saturate at the value $e_i^{\text{MAX}} = 32$ in order to approximate the digital encoding of the FPGA implementation.

The temporal organization of the circuit represented in Fig. S3 shows how is controlled the reading and writing state on the RAM in the case of a problem size $N = 500$ spins. Fig. S3 (a) shows the case of a single x_i and e_i calculation per dot product which is the classic way to integrate the differential equations (1) and (2) using the Euler method. Fig. S3 (b) represents the case of eight and four calculations of x_i and e_i, respectively, per dot product. In this case, the error correction term is computed with a normalized time step of 2^{-4} (half of the time step used in the computation of x_i). The total power consumption can be reduced by increasing the frequency of circuits involved in the calculation of x_i because these circuits are smaller than those involved in the dot product calculation (see Fig. S4 (d) and (e)).

Increasing the problem size will increase the power consumption by filling up the calculation pipelines but this can be balanced out by reducing the frequency used for the calculation of x_i for larger problem size. The end of the dot product is followed by an update of all RAM and saving of σ_i which correspond to the Most Significant Bit (MSB) of x_i values. This step is not represented here.

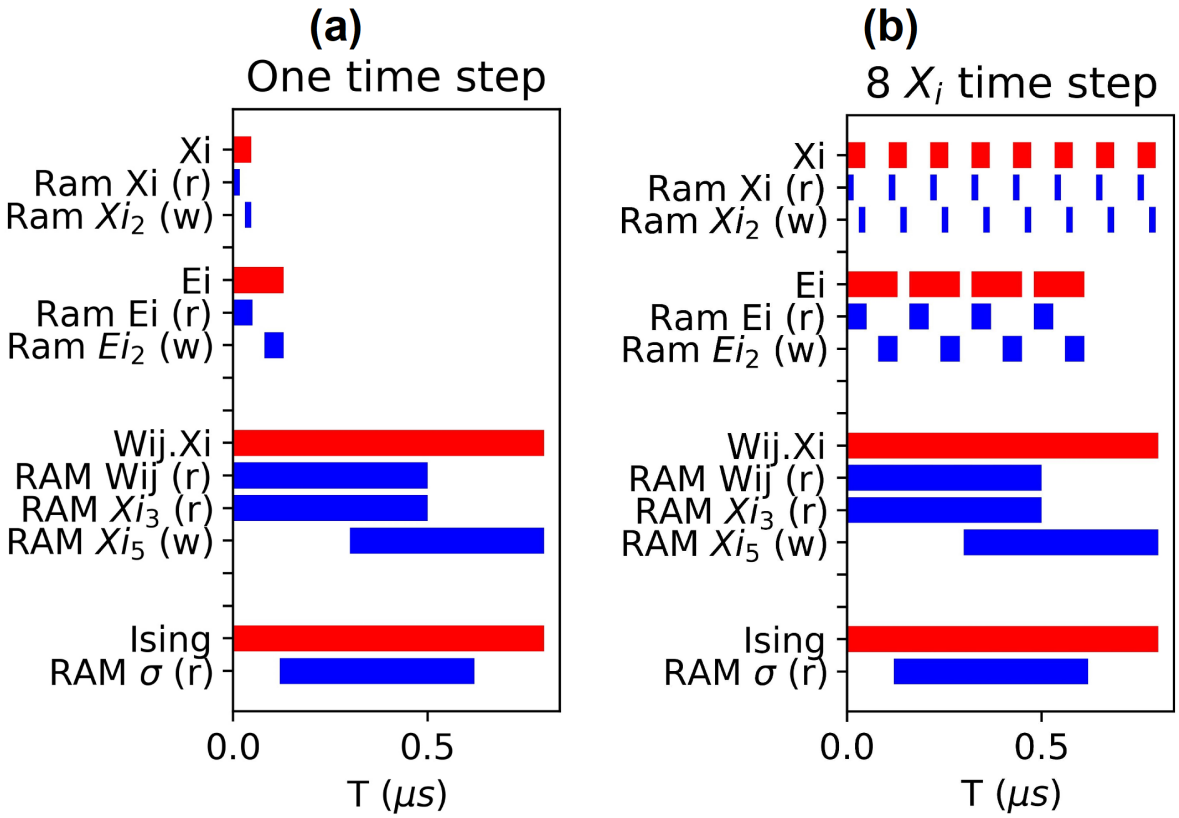

Figure S3: Temporal representation of the circuits where the red bars represent the computation cores and the blue bars the use of RAM. Here, (r) stand for read and (w) for write. (a) One time step representing the classical way of solving differential equation. In this configuration, the dot product create a bottleneck to the computation speed. (b) 8 time steps for x_i and 4 time step for x_i accelerating the computation time to find the ground state energy and create a new bottleneck to the number of possible time step possible. Note that the time between every red bars are necessary update the x_i RAM which is not represented in this figure for better clarity.

The use of several x_i and e_i calculations per dot product allows to reduce the bottleneck created by the later whose calculation is in principle the most time consuming. Fig. S4 (a) to (c) show the reduction in time to solution vs. the number of x_i calculations per dot product.

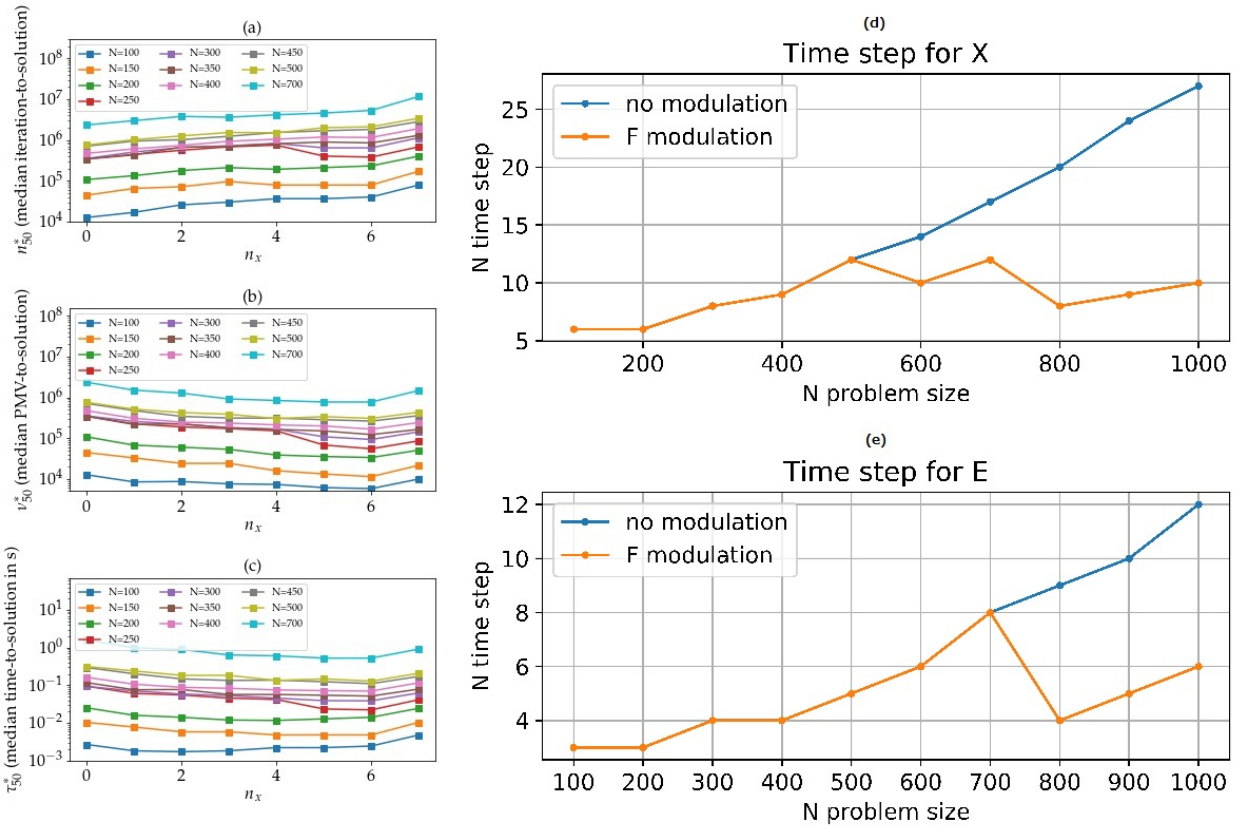

Figure S4: (a) The number of iterations of the $\boldsymbol{x}$ variable to solution vs. the number of update, noted n_x, of the $\boldsymbol{x}$ variable per matrix-vector product. (b) The number of matrix-vector multiplications MVM to solution. (c) Real time to solution. (d) Maximum possible time step with and without clock modulation on x_i. (e) Maximum possible time step with and without clock modulation on e_i.

2.4 Coupling

2.4.1 Overview of the coupling circuit

The dot product operation is preceded by a multiplication by β so that the domain of x_i is reduced and, in turn, the number of digits required to encode the integer part of x_i. The result of the dot product is multiplied by the error correction term. Since the dot product is performed at 50MHz clock speed, the optional registers of the DSP are no longer required and have been removed for the two multiplications resulting in 1 clock cycle operation for each operation.

2.4.2 High speed and massively parallel multiplication

The multiplication of the elements of a vector $\boldsymbol{x}$ by the element of a matrix Ω can be performed using an approximation based on a specific circuit combining logic equations describing the behavior of a multiplexer and the optimized use of FPGA components. This circuit allows the design to achieve 10,000 multiplications in 1 clock cycle. In our case an

element of $\boldsymbol{x}$ is fixed point binary vector on k bits that are multiplied by an element of a matrix composed of two bits vectors where ω is an element of Ω and $\boldsymbol{\omega} \in \{-1, 0, 1\}$. The behavior of a multiplexer that implements the multiplication of x by ω by selecting $R \in \{-x, 0, x\}$ is given as follows ($\bar{x}$ represent $-x$):

$$R = x\bar{\omega}_1\bar{\omega}_0 + \bar{x}\omega_1\omega_0 \tag{S18}$$

where x is an element of a vector, ω_i a binary vector and an element of the matrix Ω with i the index of the binary vector that is either 1, the most significant bit (MSB) or 0, the less significant bit (LSB). If we expand eq. (S18) to each bits x_i of the vector x and use Boolean operation, we obtain the equation eq. (S19) given as follows:

$$R = x\bar{\omega}_1\omega_0 + (\bar{x} \oplus C)\omega_1\omega_0 \tag{S19}$$

where $-x$ is represented by the two-complement operation $\bar{x} \oplus C$ that consist of inverting the bits of x and adding C a constant corresponding to 2^{-d} with d the decimal part size of the vector. Here, $\bar{x}$ now represents the bit-wise not operation. Applying De Morgan's law on eq. (S19) will lead to the following:

$$R = x\bar{\omega}_1\omega_0 + \bar{x}\omega_1\omega_0 \oplus C\omega_1\omega_0, \tag{S20}$$
$$R = \omega_0(x \oplus \omega_1) \oplus C\omega_1\omega_0, \tag{S21}$$
$$R = \omega_0(x \oplus \omega_1), \tag{S22}$$

In eq. (S21), we consider $C = 0$ which introduces an absolute error of -2^{-d} when the MSB and the LSB of ω are equal to 0. The aim of such approximation is that eq. (*S22*) can be implemented by a single LUT3 component. Consequently, achieving 10,000 operations require $k \times 10^4$ LUT3 where k represents the precision of x and is chosen to lower the required resources and error.

2.4.3 First adder stage

The circuit shown in Fig. S5 represents the operations of multiplication used in the dot product based on eq. (S22). To accelerate the computation time, multiple access memory is utilized: Fig. S5 (a) and (b) show the implementation of multiple block RAM (BRAM) that output 100 rows of 100 values at the same time. Eq. (S22) is implemented in LUT3 as described in the circuit of the Fig. S5 (c) which compute the addition of two elements of the vector $\{\omega_{ij}x_i\}_i$. Using LUT3 for the elements at index $2i$, with i the index of the generated vectors beginning at 0, and multiplexers for the elements at index $2i + 1$ ensures the use of lower resources and the optimal use of the configurable logic block (CLB).

2.4.4 DSP tree

For block matrices and vectors of size u with $u = 100$, the dot product requires to perform 10,000 multiplications and 8,100 additions. The circuit of the Fig. S5 (c) computes two multiplications and one addition. Thus, by reproducing this circuit 5,000 times, the FPGA computes 15,000 operations in a clock cycle and generates 100 vectors of 50 values that need to be added. This operation is done using an adder tree. A first stage of adder, that is not represented between the circuit of the Fig. S5 (c), is realized using the CARRY8 component that reduces the 100 vectors of 100 elements to 100 vectors of 25 elements each. The remaining elements are then added using a DSP tree in adder mode. The advantage of using an adder tree of DSP is the low LUT number that is required, the optimal use of power consumption and the possibility to increase the frequency of the circuit. The Ultrascale architecture provided by the FPGA KU040 possesses a high number of DSP that can be cascaded allowing to reduce the routing circuit and the computation time. The DSP tree is repeated 100 times to compute at the same time all elements of the dot product resulting in the use of 1,200 DSP.

2.4.5 Scalability of such architecture for bigger FPGA

The number of clock cycles required to perform the dot product is given as follows:

$$C_t(u, N) = K + (N/u)^2 \tag{S23}$$

where N is the problem size, u the divider that partitions the matrix (in the current implementation u=100) and K the number of clock cycles required for the multiplications and additions. The adder tree composed of CARRY8 and DSP previously proposed is constrained by the following condition:

$$\frac{N}{2^{h_C} + 5^{h_D}} = 1, \tag{S24}$$

where h_C represents the height of the CARRY8 tree and h_D the height of the DSP tree. The CARRY8 requires only one clock cycle when two cascaded DSP need 5 clock cycles. The height K of the adder tree (in clock cycle) is then:

$$K = h_C + 5h_D. \tag{S25}$$

We can fix h_C as a constant according to the proposed FPGA circuit and find h_D as follow:

$$N = 2^{h_C} + 5^{h_D}, \tag{S26}$$

$$\begin{aligned} N - 2^{h_C} &= 5^{h_D}, \\ \log(N - 2^{h_C}) &= h_D \log(5), \\ h_D &= \frac{\log(N - 2^{h_C})}{\log(5)}. \end{aligned} \tag{S27}$$

Then the height K is equal to:

$$K = h_C + 5\frac{\log(N - 2^{h_C})}{\log(5)}. \tag{S28}$$

The adder tree increases logarithmically if we assume that an infinite amount of resources is available. Also, we show here that the design can be significantly improved if u become larger with more available resources.

2.5 Ising energy circuit

The Ising energy is computed at the same time as the main dot product of x_i by ω_{ij} because it shares the same output from the RAM that store the ω_{ij}. As for the dot product, the Ising energy has been computed using logic equations to fit in a minimal number of LUT. The logic equation for the multiplication of σ by the matrix Ω can be described as follows:

$$S_1 = (\omega_1 \oplus \sigma_j)\omega_0, \tag{S29}$$

$$S_0 = w_0, \tag{S30}$$

where σ_i is the sign of x_i and S is a 2 bits vector representing the multiplication of σ_i at index i by ω (representing ω_{ij}) which is also represented by 2 bits whose index is either 0 (LSB) or 1 (MSB).

A circuit is also used to compute the Ising energy. Note that the Ising energy of a matrix M divided into matrices m_{ij}, where i and j are the indexes of the partitioned M, is equal to the sum of the energies of m_{ij}. Thus, the output of the pipelined Ising energy circuit is added with itself.

2.6 Non-linear term

To optimize the use of the electrical power, x_i has been designed to use the highest frequency f_x and the error correction use and intermediate frequency f_e. Both are computed several times during the operation of the dot product to reduce the computation time. Fig. S6 shows

the circuit use to compute one element of the vectors x_i and e_i that are reproduced 100 times. The circuits use pipelined DSP to compute additions, subtractions and multiplications. A shift register is used to multiply by the dt of Euler approximation. To reduce the number of DSP into the design, x_i^2 is shared between x_i and e_i through a true dual port RAM that can be used with two different clocks to synchronize the two circuits. The RAM is controlled by an external FSM in the control module of Fig. S2.

2.7 Power consumption

Energy consumption of the circuit is determined by the number of logic transitions (from 0 to 1 or 1 to 0) and the frequency with P = $\langle s \rangle CV^2F$ where P is the power dissipated by a transistor based circuit, C the switching capacitance, V the voltage and F the frequency, and $\langle s \rangle$ the average number of switch per clock cycle. Voltage and capacitance are FPGA dependent since it is already manufactured. The digital circuit are at the highest power consumption when the components are enabled and when the clock signal drives the Configurable Logic Block (CLB) or the DSP. Thus, Enable and disable the block RAM is efficient to reduce the consumption however, clock gating shows better performances in this implementation. The methods have been extended on the available FF of the Xilinx FPGA and DSP which possess clock enable gate. However, when the design becomes large, high number of signal propagating enable towards CLB or DSP increase fanout and routing complexity. This can be solved by using BUFGCE component that are available in the FPGA and able to enable or disable the clock. Thus, when a given circuit is not needed, the clock can be disabled which will reduce significantly the power consumption.

The KU040 boards use Infineon regulators IR38060 incorporated voltage and current sensors. These sensors can be access either from the FPGA or from external bus with the PMBUS protocol which is based on i2c. We used here an Arduino to communicate with the regulators and record power data.

The power has been measured for different problem sizes and does not exceed 5W. Experiments show that the most critical operation for energy efficiency and computation time is the dot product which dissipates most of the FPGA power and needs more clock cycles to operate.

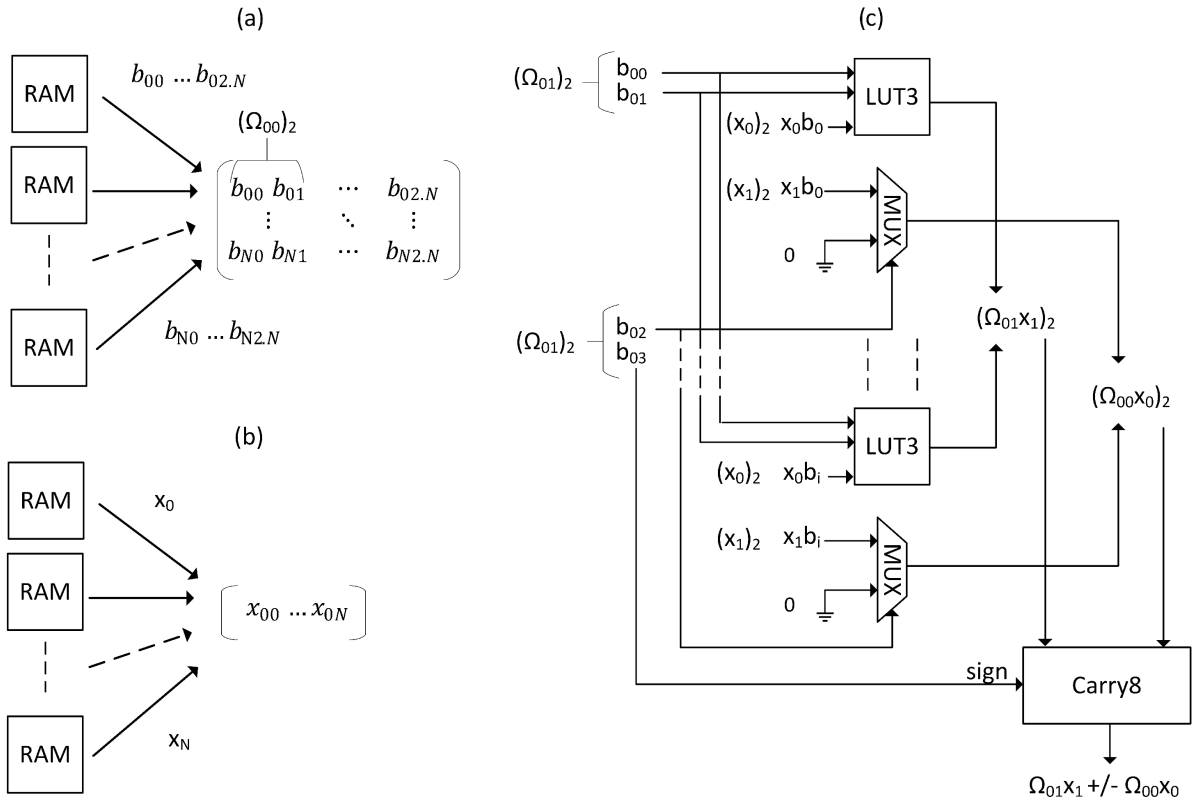

Figure S5: Representation of the high speed and multiple memory access apply to a circuit used for the dot product. (a) 100 RAM are instantiated and can be accessed at the same time. Each RAM corresponds to a row of the matrix Ω. Each element ω_{ij} of Ω is encoded on a two bits vector. (b) As in (a), 100 RAM can be accessed at the same time to compose the vector x. (c) The $\Omega_{i,2j}$ and x_{2i} values are injected into a LUT3 to compute the eq. (S22). The x_{2i+1} values are injected into a multiplexer (MUX) whose selector is controlled by the LSB of $\Omega_{i,2j+1}$. The output of the LUT3 and the MUX are propagated through the CARRY8 component that add or subtract according to the value of $\Omega_{i,2j+1}$ MSB.

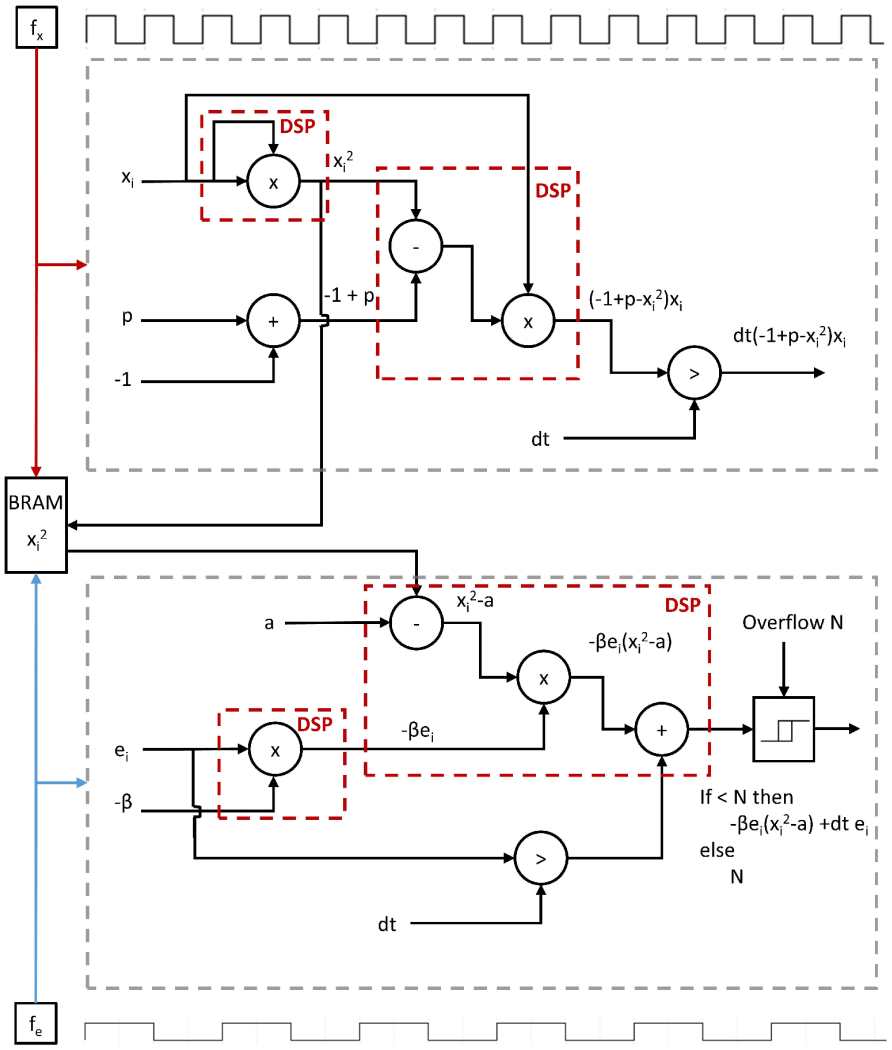

Figure S6: Circuits of the non-linear terms x_i and e_i. The two terms use different clocks and share their results through a dual port block RAM allowing to read and write at different speed. Both circuit use DSP for high speed computation and are generated one hundred times to perform parallel computation.

2.8 Parameter values used for solving SK spin glass instances

The parameter values used for solving SK spin glass instances are shown in Tab. 1.

Symbol	meaning	value
β	coupling strength	0.25
α	target amplitude baseline	3.0
p	linear gain	$0.8 - (\frac{N}{220})^{-2}$
ρ	amplitude and gain variation	3
δ	sensitivity to energy variations	10
γ	rate of increase of ξ	0.00011
τ	max. time w/o energy change	1000
n_x	number of iterations for nonlinear terms	6
n_e	number of iterations for error terms	3
dt_x	normalized time-step of nonlinear terms	2^{-6}
dt_e	normalized time-step of error terms	2^{-4}

Table 1: Parameters used for solving SK problem instances.

It is important to note that increasing the Euler step dt_x does not always decrease the time to solution for all problem sizes; in particular, the time to solution of large problem sizes ($N = 700$) is not significantly reduced when using $dt_x = 2^{-5}$ instead of $dt_x = 2^{-6}$ (see Fig. S7). This is likely because the Euler approximation for larger problem sizes are prone to numerical instability for larger integration time steps.

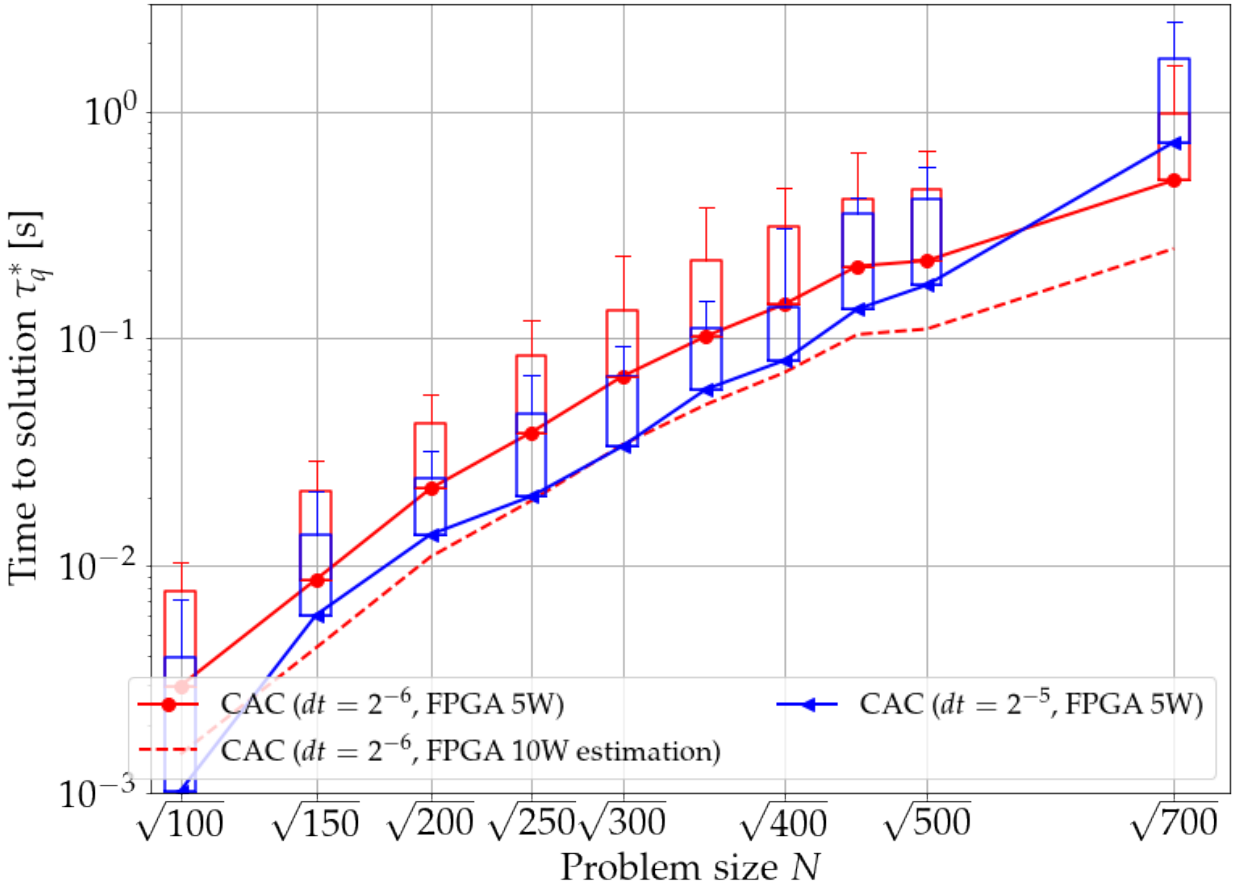

Figure S7: Lower, higher, and upper whisker of boxes show the 50^{th}, 80^{th}, and 90^{th} percentiles of the real time to solution distribution in seconds for the FPGA implementation of CAC with a maximum of 5W power consumption with $dt_x = 2^{-6}$ and $dt_x = 2^{-5}$, and estimation of FPGA implementation of CAC with a maximum of 10W power consumption with $dt_x = 2^{-6}$.

3 Benchmark on GSET

Parameter values used for solving instances of the GSET are shown in Tab. 2.

where d_1 is a function of the maximum degree given as $d_1 = \max\{d_0, 10\}$ and $d_0 = \text{mean}_i\{\sum_j |\omega_{ij}|\}$.

Symbol	meaning	value
β	coupling strength	$\frac{3}{d_0}$
α	target amplitude baseline	3.0
p	linear gain	$1 - 400d_1^{-2.5}$
ρ	amplitude and gain variation	1.0
δ	sensitivity to energy variations	$\frac{2.6}{N}$
γ	rate of increase of β	$\frac{2}{N}$
τ	max. time w/o energy change	$9N$
n_x	number of iterations for nonlinear terms	6
n_e	number of iterations for error terms	4
dt_x	normalized time-step of nonlinear terms	2^{-6}
dt_e	normalized time-step of error terms	2^{-4}

Table 2: Parameters used for solving GSET problem instances.

4 Other algorithms

4.1 Details of NMFA simulation

Noisy mean-field annealing can be simplified to the following discrete system[2]:

$$y_i(n+1) = (1-\alpha)y_i(n) + \alpha \tanh[\frac{1}{\sigma_\omega T(t)}(\sum_j \omega_{ij} y_j(n)) + \sigma_r r_i], \tag{S31}$$

with $\sigma_\omega = \sqrt{\sum_j J_{ij}^2}$. When the noise is not taken into account (i.e., $r_i = 0$), the steady state of eq. (S31) is given as follows:

$$y_i^* = \tanh[\frac{1}{\sigma_\omega T(t)}(\sum_j \omega_{ij} y_j(n))], \tag{S32}$$

Note that the solutions of the eq. (S32) are the same as those of the TAP naive mean-field equations (see [3]). Moreover, they are the same as the steady state of eq. (1) when considering the change of variable $y_i = g(x_i)$ with $g(x) = \tanh(x)$ and $\beta_i(t) = \frac{1}{T(t)}$. In fact, it can be shown that the two systems have the same set of attractors[4].

The default parameters used in the numerical simulations are given as follows[2]: $\alpha = 0.15$ and $\sigma_r = 0.15$.

Moreover, the temperature $T(t)$ is decreased with time according to an annealing schedule.

The eq. (S31) is simulated on a GPU using CUDA code provided in [2]. Various parameters of the temperature scheduling $T(t)$ and parameters α and σ_r have been tried in order to maximize the performance of this algorithm in finding the ground state of SK spin glass problems (see Figs. S8).

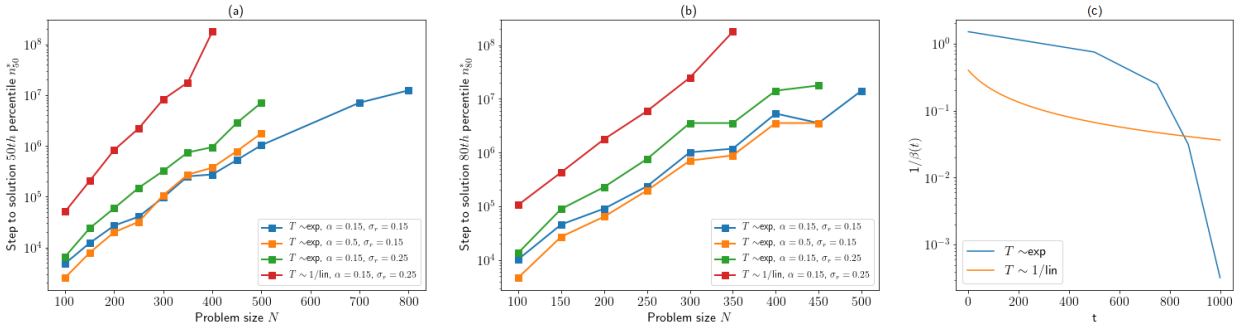

Figure S8: 50^{th} (a) and 80^{th} (b) percentiles of the step to solution distribution vs. the problems size N of bimodal Sherrington-Kirkpatrick spin glass problems. (c) Exponential and inverse linear scaling of $T(t) = \frac{1}{\beta(t)}$ for $T = 1000$.

4.2 Details of CIM simulation (simCIM)

The physical model of the measurement feedback coherent Ising machine developed in [5] can be simplified as follows:

$$x_i(n+1) = AG(x_i(n)) + r_1 + \sqrt{\xi_0}\Theta(B\sum_j \omega_{ij} G(x_i(n)) + r_3)) \tag{S33}$$

where $\Theta(x) = R(-Fx; x_{\max})$ and $R(x; y)$ is the saturation function defined as $R(x; y) = x$ if $|x| < y$ and $R(x; y) = x_{\max}$ otherwise. If we assume, for simplicity, that the saturation function $\Theta(x)$ is simply linear with $\Theta(x) = F_t x$, then eq. (S33) can be written under the form of eq. (1) by using the following: $f(x_i) = AG(x_i)$, $g(\beta x_i) = G(x_i)$, $\beta_i(t) = F(t)B\sqrt{\xi_0}$, and $r_1 + \sqrt{\xi_0} + r_3 = \sigma\eta_i$.

Eq. (S33) is simulated using a GPU implementation in order to approximate accurately the success probability when it is small.

References

[1] Hopfield, J. J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences* **81**, 3088–3092 (1984).

[2] King, A. D., Bernoudy, W., King, J., Berkley, A. J. & Lanting, T. Emulating the coherent ising machine with a mean-field algorithm. *arXiv preprint arXiv:1806.08422* (2018).

[3] Bilbro, G. *et al.* Optimization by mean field annealing. In *Advances in neural information processing systems*, 91–98 (1989).

[4] Pineda, F. J. Dynamics and architecture for neural computation. *Journal of Complexity* **4**, 216–245 (1988).

[5] McMahon, P. L. *et al.* A fully programmable 100-spin coherent Ising machine with all-to-all connections. *Science* **354**, 614–617 (2016). URL http://www.sciencemag.org/lookup/doi/10.1126/science.aah5178.