Supplementary materials: Scaling advantage of chaotic amplitude control for high-performance combinatorial optimization

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

[1]Institute of Industrial Science, University of Tokyo, 4-6-1 Komaba, Meguro City, Tokyo 153-8505, Japan
[2]International Research Center for Neurointelligence, University of Tokyo, 7-3 Hongō, Bunkyo City, Tokyo 113-0033, Japan
[3]Theoretical Quantum Physics Laboratory, RIKEN Cluster for Pioneering Research, Saitama, 2-1, Hirosawa, Wako, Saitama 351-0198, Japan
[4]Physics Department, The University of Michigan, 500 S State St, Ann Arbor, MI 48109, USA
[5]IMS laboratory, CNRS UMR 5218, University of Bordeaux, Talence, 33405, France
[6]LIMMS/CNRS-IIS, University of Tokyo, 4-6-1 Komaba, Meguro City, Tokyo 153-8505, Japan
[7]Research Laboratory of Electronics, MIT, 50 Vassar Street, Cambridge, Massachusetts 02139, USA
[8]PHI (Physics & Informatics) Laboratories, NTT Research Inc., 940 Stewart Drive, Sunnyvale, CA 94085, USA
[*]corresponding author(s): Timothée Leleu (timothee.leleu@ircn.jp)

November 19, 2021

Supplementary Note 1 Theoretical analysis

Supplementary Note 1.1 Derivation of a potential function

First, we analyze the system described by eq. (1) only, when $\sigma_0 = 0$ and $\beta_i = \beta$, $\forall i$. That is to say, we assume that error signals β_i are constant in this section. 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]:

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

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

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

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.

Supplementary Note 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[1] 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).

[1] 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.

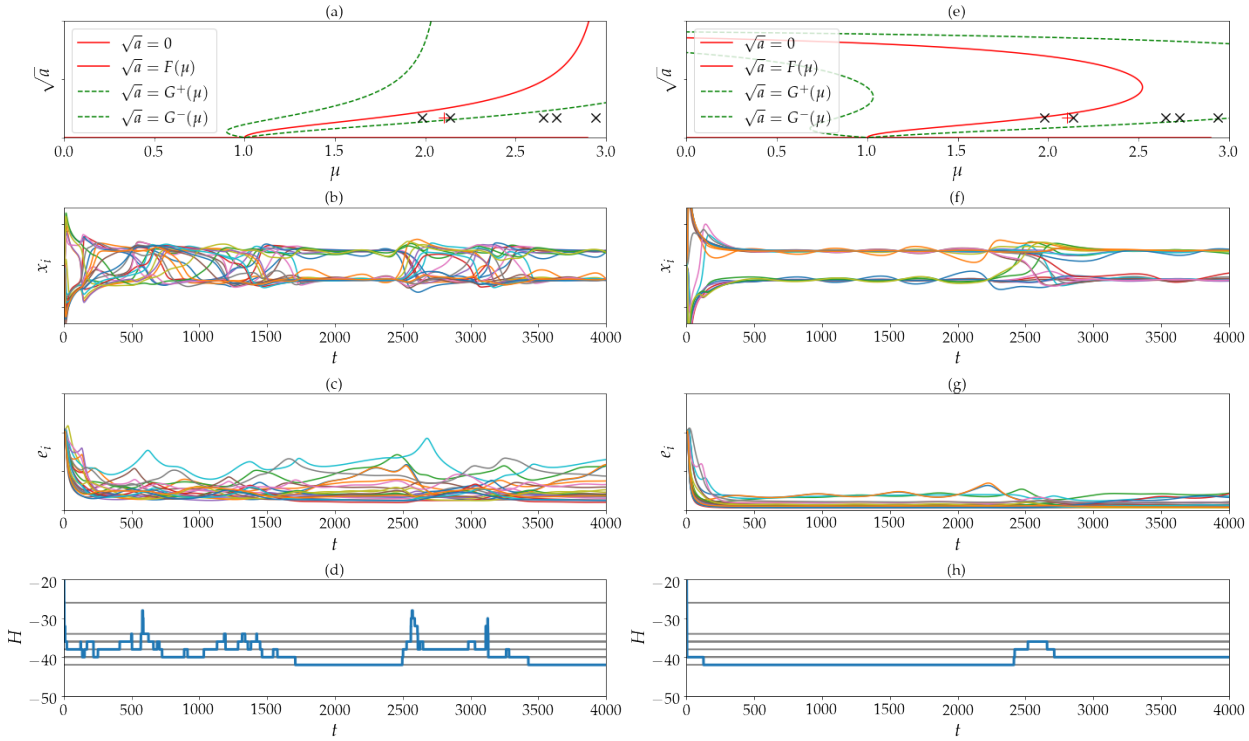

Figure S1: (a,e) Bifurcation diagram in the space $\{\sqrt{a}, \mu = \mu_j(\boldsymbol{\sigma})\}$, $\forall j \in \{1, \ldots, N\}$, at configuration $\boldsymbol{\sigma}$. The full red and green dotted lines correspond to the set for which $Re[\lambda_i^{\pm}] = 0$ (where fixed points corresponding 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 $\beta = 0.1$, $\xi = 0.1$.

Supplementary Note 1.3 Comparison with the system proposed in Leleu et al. 2019[2]

The principles governing the system described in the current work and Leleu et al. 2019[2] are similar, but the details of the implementation are different in order to render the system implementable on a FPGA notably because of the requirement for a fixed point precision encoding.

Although the simplified system proposed in the current manuscript behaves similarly to the one proposed in Leleu et al. 2019[2], there are difference in the behavior of the algorithm for some special instances which can be observed by comparing the decay of the number of instances unsolved after n steps, noted $E(n)$, when using both algorithms (see Fig. S2 (a)). $E(n)$ was previously used for the comparison with BLS algorithm in Fig. 3 of Leleu et al. 2019[2]. It is shown that the $E(n)$ curves of the two schemes are superimposed up to a point where $E(n)$ tends to saturate. This implies that there are rare instances (less than 1%) for which the heuristic scheme of the current manuscript does not visit the solution state in a single run.

This behavior is possible because, contrary to the scheme in Leleu et al. 2019[2], the simplified scheme of the current manuscript is heuristic and there is less analytical justification for the dynamics to not get trapped in some attractors. Note that the proportion of these exceptional instances for which the dynamics can become trapped does not seem to increase with the problem size, as shown by the fact that the probability to reach the solution state is of the order of 1 even for problem size $N = 1000$ (see Fig. 2b). Thus, the simplified version retains most of the properties of the original system as a fast solver for most instances, has similar scaling properties, and can be implemented more easily on a FPGA.

As explained in Leleu et al. 2019[2], a small but positive Gibbs entropy production rate can generally ensure that trajectories eventually visit the solution states. In the case of the heuristic scheme proposed in this work, the failure of visiting the solution state that occurs for a few special instances (see Fig. S2 (a)) can be linked to the fact that the Gibbs entropy production rate does not remain positive. In Fig. S2 (c), we have counted the average number of unique configurations visited after n steps of the system proposed in this paper and compared it to the failure of these instances to find the solution state (see Fig. S2 (b)). Instances for which the number of unique configurations visited saturates, i.e., for which the Gibbs entropy production rate becomes therefore either zero or negative, are the ones that have a lower probability of visiting the solution state. For these instances, the dynamics after many iterations is stuck in a limit cycle (less frequently, a chaotic attractor) and there is not a sufficient increase in Gibbs entropy for escaping these non-trivial attractors.

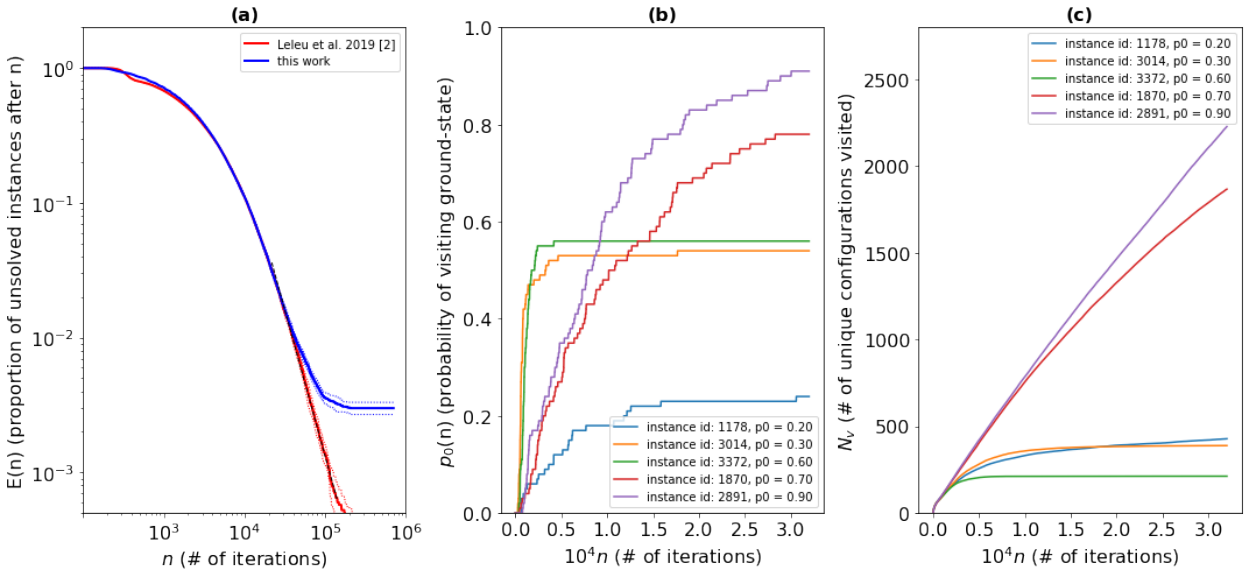

Figure S2: (a) Proportion of unsolved SK instances of size $N = 100$ after n steps, noted $E(n)$, when using the system described in Leleu et al. 2019[2] and this work (blue). $E(n)$ is estimated using 5000 randomly generated instances. (b) Probability $p_0(n)$ of visiting the solution-state in a single run after n steps for 5 instances (instances with id 1178, 3014, 3372, 1870, and 2891) that have the lowest probability $p_0(\infty)$ of being solved after a very large number of steps and (c) average number of unique configurations visited after n steps for these instances.

Supplementary Note 2 FPGA implementation

Supplementary Note 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.

Supplementary Note 2.2 Top level entity

The circuit is organized into four principal building blocks as shown on Fig. S3 (a). Several clock frequencies have been generated to concentrate the electrical power on the circuits that needs high speed clock when these circuits constitute a bottleneck in the current implementation. Finally, the organization of the main circuit is represented in Fig. S3 (b). The control block is composed of Finite State Machines (FSM) and is well tuned to pilot all computation cores, 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 further reduce the global power dissipation.

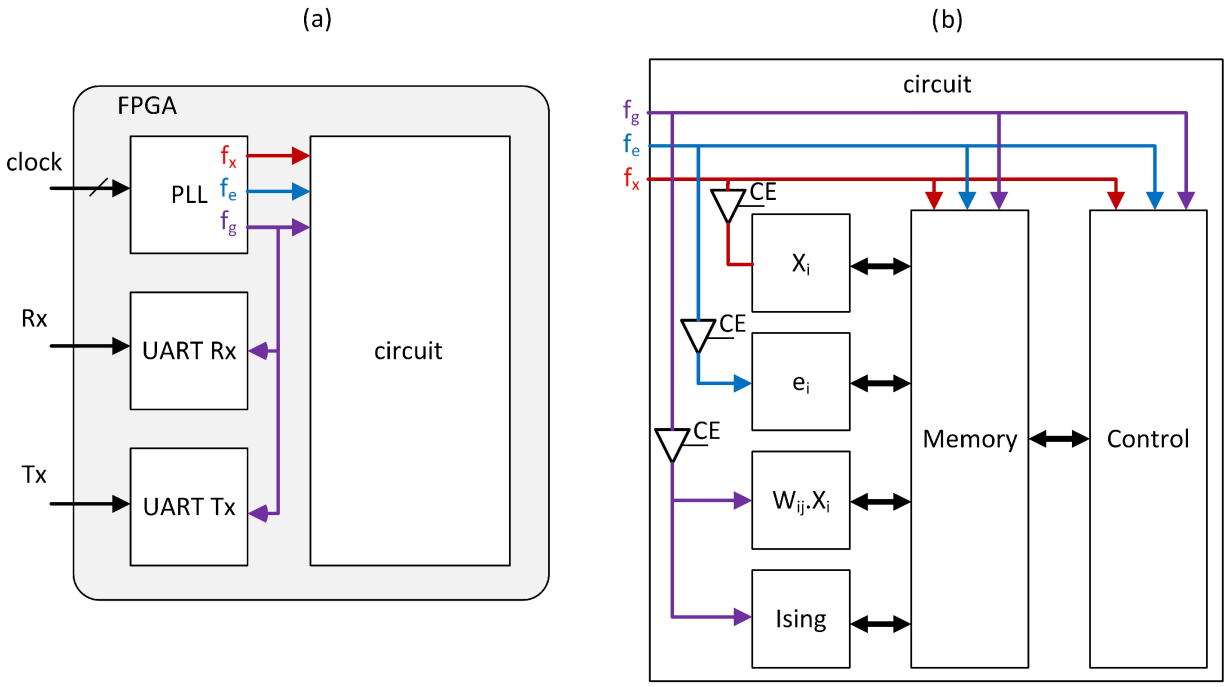

Figure S3: Organization of the FPGA circuit. (a) The top layer entity is divided into four modules: The Phased-Locked Loop (PLL) that convert a differential clock of 250MHz into three clocks f_g=50MHz, f_x=300Mhz and f_e=100MHz respectively representing the global clock, the clock for x_i and the clock for e_i. Two UART modules are used to receive 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.

Supplementary Note 2.3 Temporal organization of the computation

The control circuit pilots 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
  $\boldsymbol{I} \leftarrow \epsilon\boldsymbol{\beta}\dot{(}\Omega\boldsymbol{x}')$ ▷ calculation of injection term
  $\boldsymbol{\sigma} \leftarrow \frac{x_i'}{|x_i'|}$
  $H \leftarrow -\frac{1}{2}\boldsymbol{\sigma}\dot{(}\Omega\boldsymbol{\sigma})$ ▷ 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
    $\boldsymbol{\Delta}_e \leftarrow -\xi((\boldsymbol{x}')^2 - a)\boldsymbol{e}$
    $\boldsymbol{e} \leftarrow \boldsymbol{e} + \boldsymbol{\Delta}_e dt_e$
  $\xi \leftarrow \xi + \Gamma dt$ ▷ Update ξ
  $\Delta H \leftarrow H - H_{\text{opt}}$ ▷ Update a
  $a \leftarrow \alpha + \rho\tanh(\delta\Delta H)$
  if $\nu - \nu_c > \tau/dt$ **then** ▷ Reset of ξ
    $\nu_c \leftarrow \nu$
    $\xi \leftarrow 0$
  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. S4 shows how is controlled the reading and writing state on the RAM in the case of a problem size $N = 500$ spins. Fig. S4 (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. S4 (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. S5 (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.

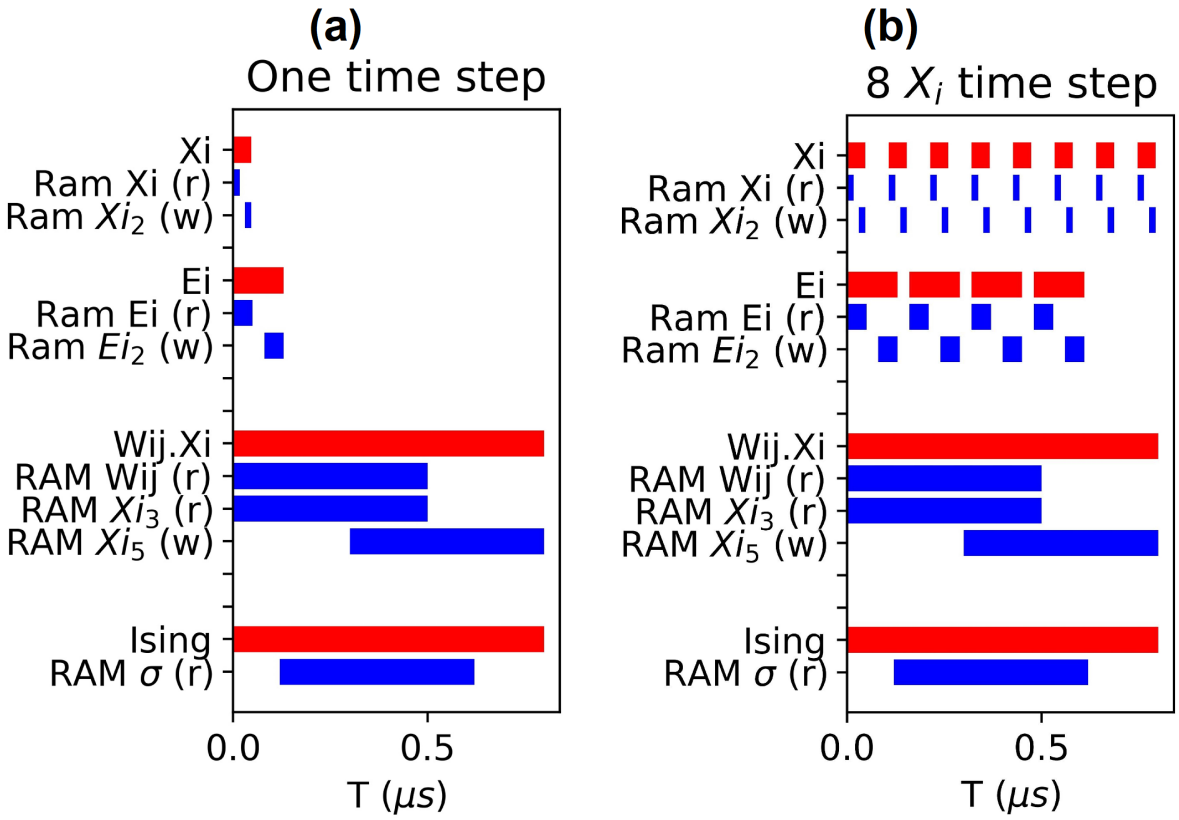

Figure S4: Temporal representation of the circuits where the red bars represent the computation cores and the blue bars the use of RAM. Here, (r) stands for read and (w) for write. (a) One time step representing the classical way of solving differential equations. In this configuration, the dot product creates a bottleneck to the computation speed. (b) 8 time steps for x_i and 4 time steps for x_i accelerating the computation time to find the solution state energy and create a new bottleneck to the number of possible time step possible.

The use of several x_i and e_i calculations per matrix vector multiplication allows to reduce the bottleneck created the long calculation time of one MVM which is in principle the most time consuming operation. Fig. S5 (a) to (c) show the reduction in time to solution vs. the number of x_i calculations per matrix vector multiplication.

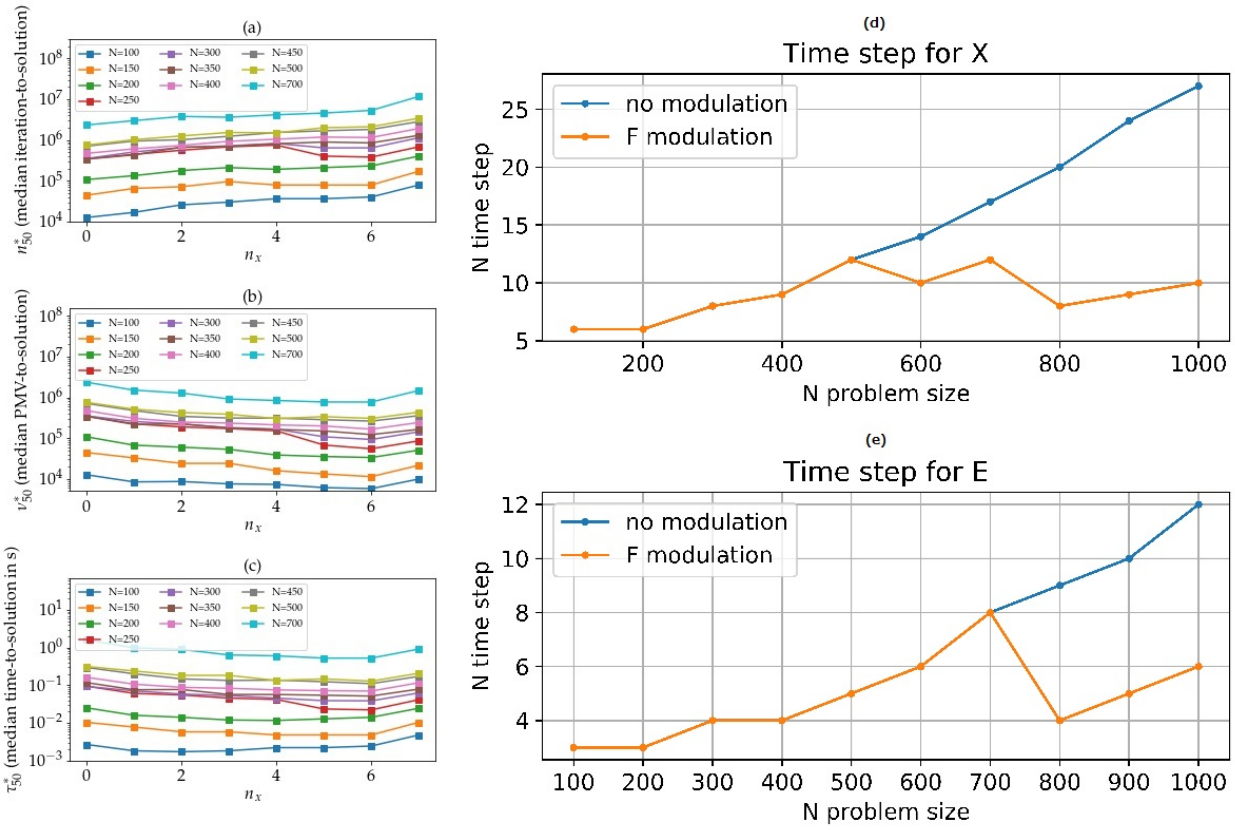

Figure S5: (a) The number of iterations of the $\boldsymbol{x}$ variable to solution vs. the number of update, noted n_x, of the $\boldsymbol{x}$ variable per matrix-vector product. (b) The number of matrix-vector multiplications (MVM) to solution. (c) Real time to solution. (d) Maximum possible time step with and without clock modulation on x_i. (e) Maximum possible time step with and without clock modulation on e_i.

Supplementary Note 2.4 Coupling

Supplementary Note 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.

Supplementary Note 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 an 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.

Supplementary Note 2.4.3 First adder stage

The circuit shown in Fig. S6 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. S6 (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. S6 (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).

Supplementary Note 2.4.4 DSP tree

For block matrices and vectors of size u with $u = 100$, the matrix vector multiplication requires 10,000 multiplications and 8,100 additions. The circuit of the Fig. S6 (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. S6 (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.

Supplementary Note 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 becomes larger with more available resources.

Supplementary Note 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.

Supplementary Note 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. S7 shows

the circuit used 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. S3.

Supplementary Note 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 accessed either from the FPGA or from external bus with the PMBUS protocol which is based on i2c. We used 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.

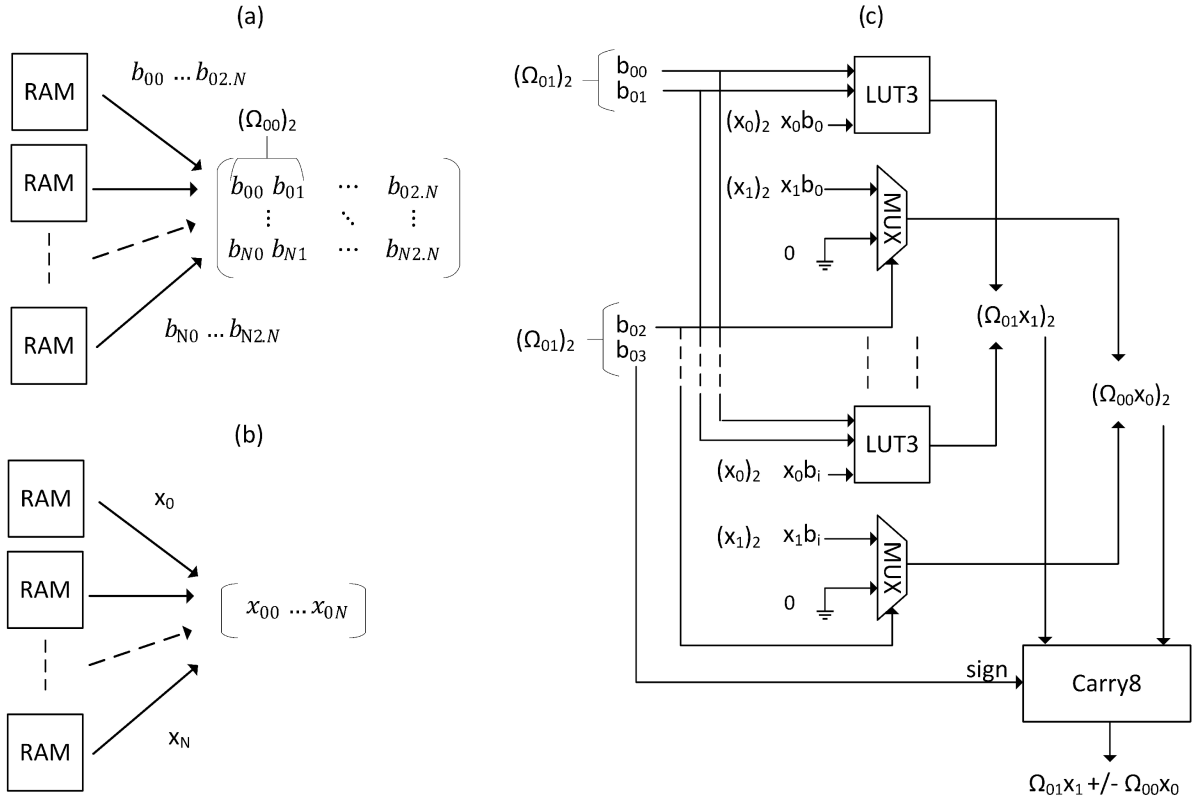

Figure S6: Representation of the high speed and multiple memory access apply to a circuit used for the dot product. (a) 100 RAM are instantiated and can be accessed at the same time. Each RAM corresponds to a row of the matrix Ω. Each element ω_{ij} of Ω is encoded on a two bits vector. (b) As in (a), 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.

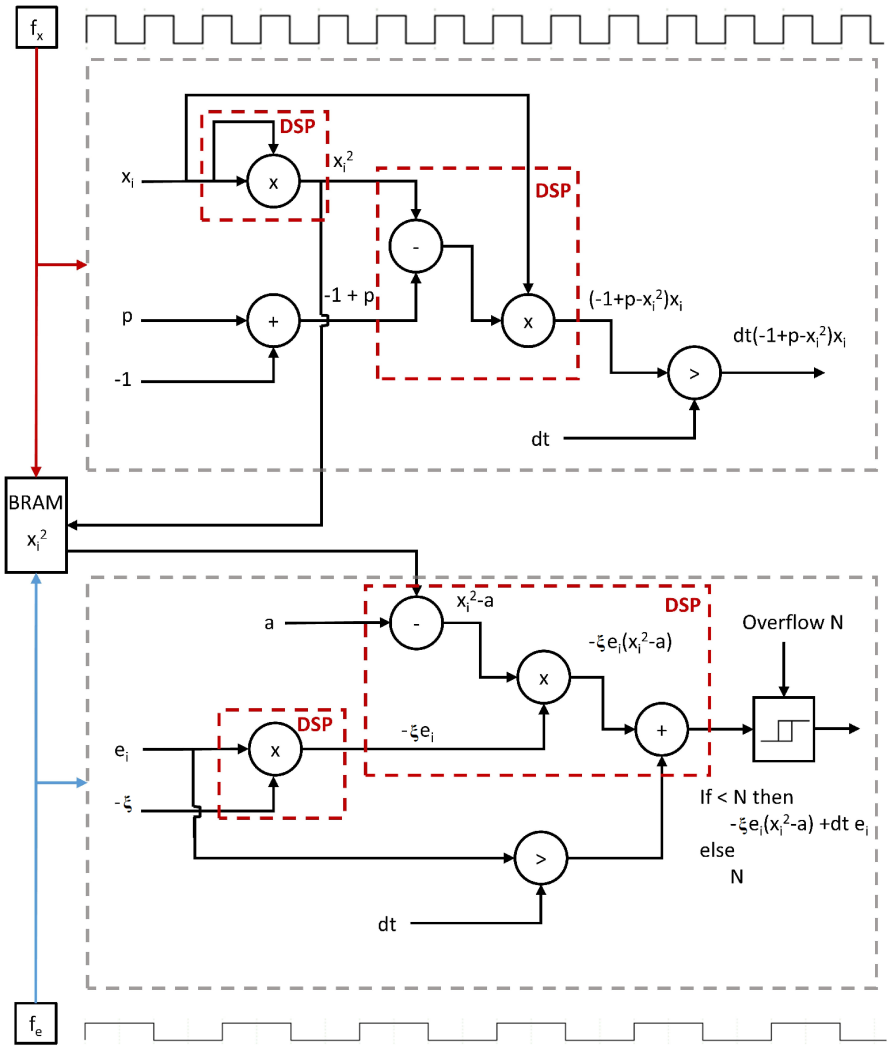

Figure S7: Circuits of the non-linear terms x_i and e_i. The two terms use different clocks and share their results through a dual port block RAM allowing to read and write at different speeds. Both circuit use DSP for high speed computation and are generated one hundred times to perform parallel computation.

Supplementary Note 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. S1.

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 S1: 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. S8). This is likely because the Euler approximation for larger problem sizes are prone to numerical instability for larger integration time steps.

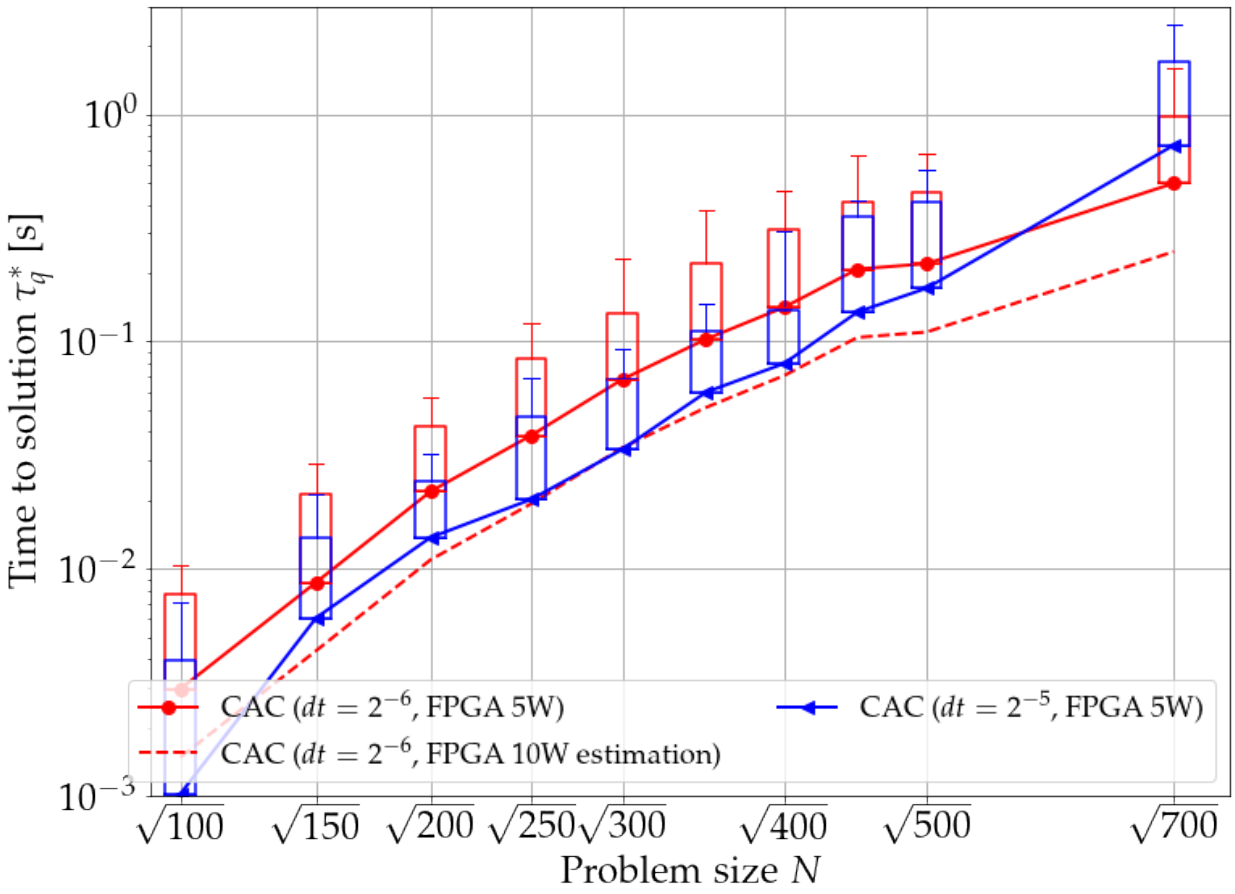

Figure S8: Lower, higher, and upper whisker of boxes show the 50^{th}, 80^{th}, and 90^{th} percentiles of the real time to solution distribution in seconds for the FPGA implementation of CAC with a maximum of 5W power consumption with $dt_x = 2^{-6}$ and $dt_x = 2^{-5}$, and estimation of FPGA implementation of CAC with a maximum of 10W power consumption with $dt_x = 2^{-6}$.

Supplementary Note 3 Benchmarks

Supplementary Note 3.1 Benchmark on SK

In order to construct the benchmark set used for the estimation of time to solution of Sherrington-Kirkpatrick instances, we have confirmed that the same optimal energies are found by using 6 different heuristics (BLS, NMFA, SA, PT, simCIM, and CAC) with very

large computational resources. In order to check the energies found are not different from the ground-state energies by a very large difference, we compare the energies found to the ground-state energy predicted using finite size scaling theory of mean field spin glasses. Fig S9 shows that the distribution of solution-states energies of the constructed benchmark set is well-fitted with that of the analytical estimates[3]. Note that Fig. S9 does not prove that the solution states found are the real ground states.

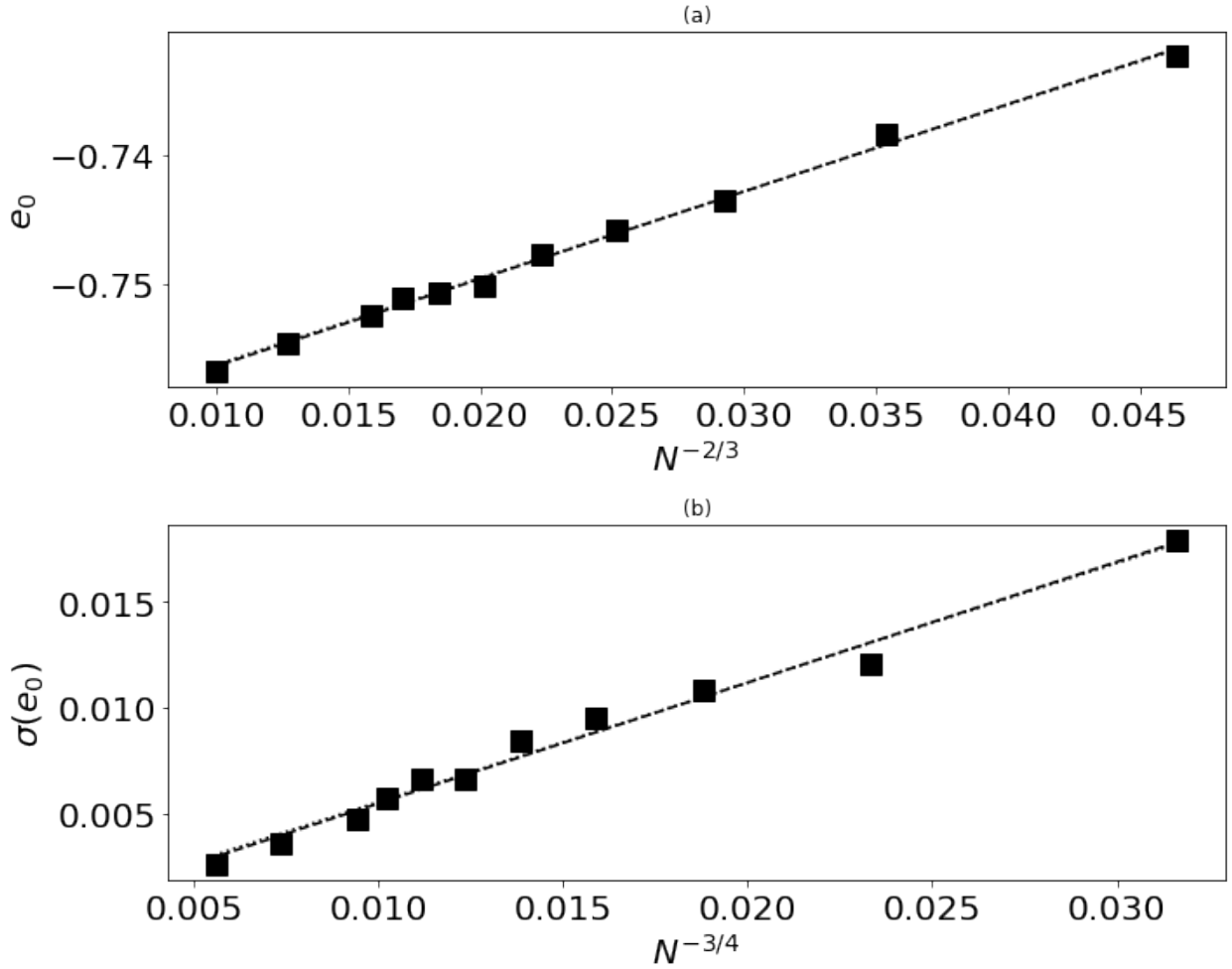

Figure S9: Mean (a) and standard deviation (b) of the energy per site $e_0 = \frac{\mathcal{H}}{N}$ of solution states in the constructed benchmark based on 100 randomly generated instances shown by black squares. The analytical estimates based on finite size scaling theory are shown by dotted lines[3].

Supplementary Note 3.2 Benchmark on GSET

Parameter values used for solving instances of the GSET are shown in Tab. S2.

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 ξ	$\frac{2}{N}$
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 S2: Parameters used for solving GSET problem instances.

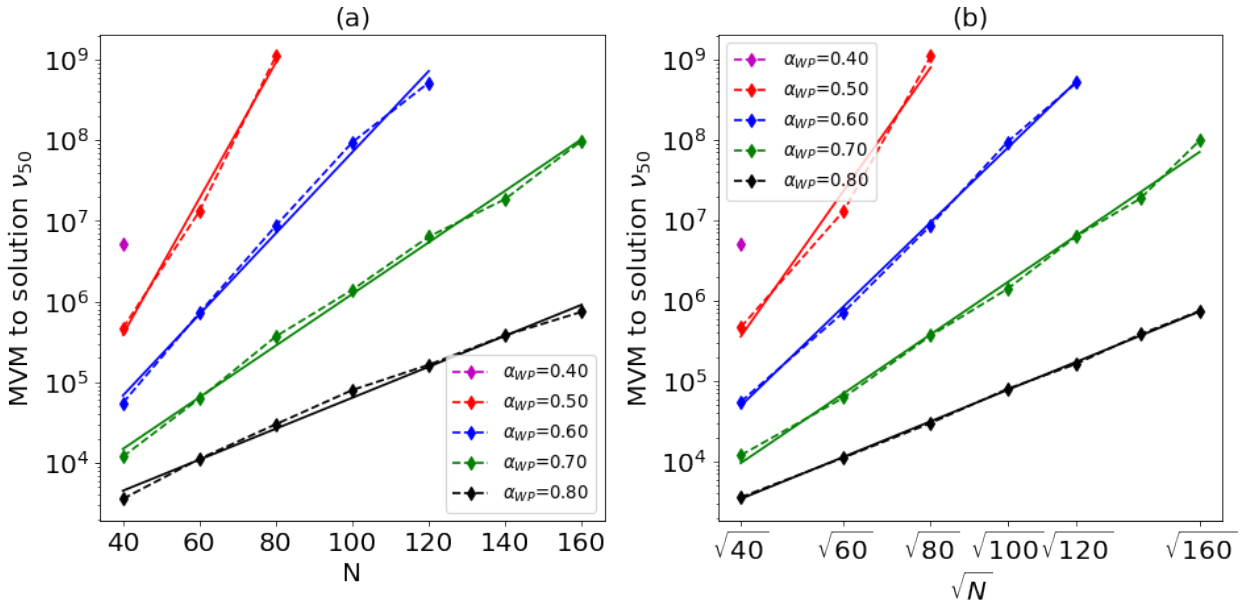

Figure S10: 50^{th} percentile (median) of the MVM to solution distribution ν_{50} vs. system size N (a) and its square root $\sqrt{N}$ (b). 10 instances per problem size were generated using the Chook library[5] and the parameters were fine-tuned to each individual instances: ξ and p are fixed and a is linearly increased from a_0 to a_1, with a_0 and a_1 two real-valued parameters. x_i is constrained to the interval $[-1, 1]$ for increasing numerical stability. Dotted lines show exponential and root-exponential fit in (a) and (b), respectively.

Supplementary Note 3.3 Benchmark on Wishart planted instances

We apply CAC to solving instances from the Wishart planted (WP) problem set[4]. These recently proposed problems have the interesting feature that the ground states are known in all cases and their energy landscapes can be tuned with a parameter α_{WP} that is arguably related to the algorithmic hardness for local search heuristics. Figure S10 shows the number of MVMs to find the known ground states of WP instances generated with integer precision by the publicly available Chook library[5]. When the parameter α_{WP} is close to 1, which corresponds to more funnel-like energy landscapes and the algorithmic "easy" regime[4], the MVM to solution scaling with problem size appears to be root-exponential at finite problem sizes (up to $N = 160$ for $\alpha_{\text{WP}} = 0.8$) which is reminiscent of the scaling behavior shown for SK instances (see Fig. 2 (e)). On the other hand when the parameter α_{WP} is close to 0.4, which is closer to the hard-easy transition, the scaling of MVM to solution seems to become an exponential increase even at small N[4] but it is difficult to tell if this is because we are using sub-optimal parameters for such harder problems. It is an open question to verify whether the sub-exponential scaling observed is finite size effect that holds for larger and larger N as α_{WP} increased or whether there is a critical transition at which the scaling behavior of the MVM to solution suddenly changes.

Supplementary Note 4 Other algorithms

Supplementary Note 4.1 Details of NMFA simulation

Noisy mean-field annealing can be simplified to the following discrete system[6]:

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

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 [7]). 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[8].

The default parameters used in the numerical simulations are given as follows[6]: $\alpha = 0.15$ 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 [6]. 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 solution states of SK spin glass problems (see Figs. S11).

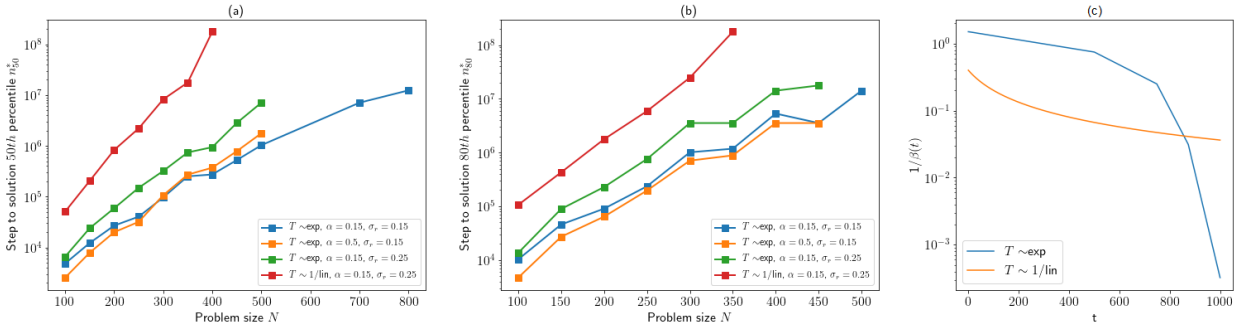

Figure S11: 50^{th} (a) and 80^{th} (b) percentiles of the step to solution distribution vs. the problems size N of bimodal Sherrington-Kirkpatrick spin glass problems. (c) Exponential and inverse linear scaling of $T(t) = \frac{1}{\beta(t)}$ for $T = 1000$.

Supplementary Note 4.2 Details of CIM simulation (simCIM)

The physical model of the measurement feedback coherent Ising machine developed in [9] 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 accurately approximate the success probability when it is small.

Supplementary 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] 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).

[3] Oppermann, R., Schmidt, M. J. & Sherrington, D. Double criticality of the Sherrington-Kirkpatrick model at t= 0. *Physical review letters* **98**, 127201 (2007).

[4] Hamze, F., Raymond, J., Pattison, C. A., Biswas, K. & Katzgraber, H. G. Wishart planted ensemble: A tunably rugged pairwise Ising model with a first-order phase transition. *Phys. Rev. E* **101**, 052102 (2020). URL https://link.aps.org/doi/10.1103/PhysRevE.101.052102.

[5] Perera, D. *et al.* Chook – a comprehensive suite for generating binary optimization problems with planted solutions (2021). 2005.14344.

[6] 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).

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

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

[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.