nature portfolio

https://doi.org/10.1038/s41928-021-00546-4

Supplementary information

Scaling out Ising machines using a multi-chip architecture for simulated bifurcation

In the format provided by the authors and unedited

1 Nomenclature

N	number of oscillators (problem size/machine size)
P_{chip}	chip parallelism (number of chips)
P_{comm}	communication parallelism per chip
P_{comp}	computation parallelism per chip
R_{cc}	architectural index related to the ratio of computation time to communication time
$R_{\mathrm{cc}}^{\mathrm{min}}$	minimum R_{cc} that results in $T_{\mathrm{comp}}/T_{\mathrm{comm}}$ larger than 1
N_{step}	number of SB steps (or number of simulated annealing steps)
T_{step}	time per SB step in μs
T_{comm}	communication time per SB step (inc. T_{stall}) in μs
T_{stall}	stall time per SB step in μs
T_{comp}	computation time per SB step in μs
F_{kernel}	clock frequency of SB kernel in MHz
T_{cyc}	time per clock cycle of SB kernel in ns
N_{packet}	number of clock cycles for one send (/receive) operation
λ_{comm}	circuit latency for communication (send/receive) modules in cycles
λ_{stall}	averaged stall cycles per communication subphase
λ_{PHY}	physical delay for transmission path in cycles (or in ns)
λ_{comp}	circuit latency for computation (MMTE) module in cycles

2 Simulated bifurcation

The SB-original Hamiltonian is defined by

$$H(\mathbf{x},\mathbf{p},t)=\sum_{i=1}^{N}\left(\frac{p_i^2}{2}+\frac{\alpha_0-\alpha(t)}{2}x_i^2+\frac{\beta_0}{4}x_i^4\right)-\frac{\gamma_0}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}J_{ij}x_ix_j+\eta(t)\sum_{i=1}^{N}h_ix_i, \tag{1}$$

where x_i and p_i are the position and momentum, respectively, of the ith oscillator corresponding to the ith spin, and $\alpha(t)$, $\eta(t)$, α_0, β_0, and γ_0 are control and constant parameters (the Hamiltonian in Eq. (1) is extended from that in Ref. [1] to include local fields).

The SB Hamiltonian is initially set to have a single local minimum at the origin of $2N$-dimensional phase space $(x_1,\cdots,x_N,p_1,\cdots,p_N)$ by properly setting the parameters, and the initial state of the N-oscillator system is set to be around there. As $\alpha(t)$ gradually increases with time, the SB Hamiltonian gradually changes to have multiple local minima, leading to bifurcations. Then, the system follows one of the local minima after the bifurcations (adiabatic search) [1]. Assuming that local minima with lower final energies appear earlier, the system will arrive at a local minimum with low final energy. Moreover, the system can find, with a higher probability, a lower local minimum in the energetically allowable region including multiple local minima (ergodic search) [1]. After the predetermined time steps, the continuous variable x_i is digitized to be the ith spin (±1) by taking the sign of final x_i, providing a good approximate solution of the Ising problem.

3 MAX-CUT problem

The MAX-CUT problem [2] is to partition the nodes of a weighted graph (for this, $w_{ij}=w_{ji}$ is the weight between the ith and the jth nodes) into two groups such that the result maximizes the total weight (or cut value) of the edges cut by the division. This problem is mapped to an Ising problem by setting $J_{ij}=-w_{ij}$, $h_i=0$ and assigning an Ising spin to each node of the graph (the spin value, ±1, represents one of the two groups). The cut value $C(\mathbf{s})$ is expressed by

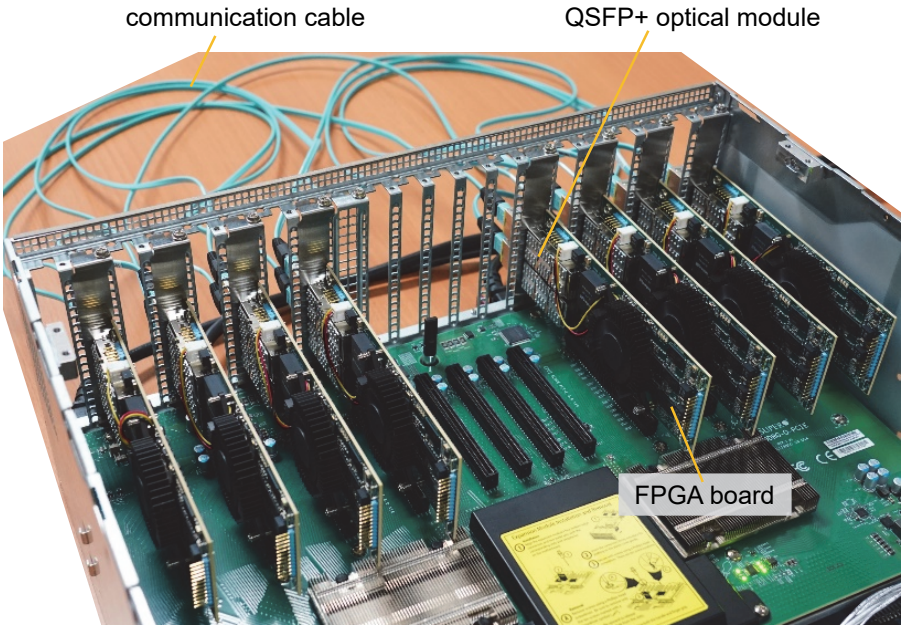
$$C(\mathbf{s})=\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}w_{ij}\frac{1-s_is_j}{2}=\frac{1}{2}\left(C_0-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}w_{ij}s_is_j\right), \tag{2}$$

where $C_0 = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}$. Since C_0 is a constant, maximizing the cut value is equivalent to minimizing the Ising energy.

K_{2000} is a 2,000-node complete-graph MAX-CUT problem solved as a benchmark problem by a CIM [3]. K_N (N ={4,096, 8,192, 16,384}) is an N-node complete-graph MAX-CUT problem whose weights are defined similarly as for K_{2000}: $w_{ij} = w_{ji} = 2r_{f(i,j)} - 1$ $(i < j)$, where $f(i,j) = N(i-1) - i(i-1)/2 + (j-i)$ and r_n is the lowest bit, {0,1}, of the nth random integer generated by the pseudorandom integer generation function in the algorithm called the Mersenne Twister [4] with seed = 1.

4 Implementation

We implemented SB-based Ising machines with FPGA clusters having up to eight FPGAs (Supplementary Fig. 1). Multiple FPGA boards (Aval APX-AA10L1 [5]), each of which has an Arria 10 GX 1150 FPGA [6] and two QSFP+ communication channels (40 Gbps TX plus 40 Gbps RX per channel), are connected with optical fibre cables (1 meter). The FPGA has 427,200 adaptive logic modules (ALMs) including 854,400 adaptive look-up-tables (ALUTs, 5-input LUT equivalent) and 1,708,800 flip-flop registers, 2,713 20Kbit-size RAM blocks (BRAMs), and 1,518 digital signal processor blocks (DSPs). We coded the SB custom circuits presented in Chip design (Main text) in a high-level synthesis (HLS) language (Intel FPGA SDK for OpenCL, ver. 17.1 [7]). The FPGA and optical cable are interfaced with a QSFP+ optical module (Avago AFBR-79EQDZ [8]). The multiple FPGA boards are mounted on a host server (Supermicro SYS4028GR-TR2 [9]).



Supplementary Figure 1: FPGA-cluster implementation of SB-based Ising machine ($P_{\mathrm{chip}} = 8$)

Supplementary Figure 2 shows the details for the single/multiple-FPGA implementations. The single-FPGA implementation (after Ref. [10]) has the same SB core as in the multi-chip implementation, but it is cross-coupled with a $\mathbf{x}_{\mathrm{mem}}$ module that enables the collection and distribution of the position information between the multiple MMTE modules.

For all the implementations, P_{comp} is 8,192 ($P_b = 8$, $P_r = 32$, and $P_c = 32$). P_{comm} is 32, where W_{comm} is 256 bits, W_{data} is 16 bits per x datum, and N_{ring} is 2. F_{channel} is 150 MHz (37.5 Gbps for each TX/RX path), which is smaller than F_{kernel} (266 MHz on average). λ_{comp} (circuit latency for the computation module) and λ_{comm} (circuit latency for the communication modules) are determined by circuit design and synthesis. λ_{comp} and λ_{comm} are slightly different between the instances as a result of the optimization by the HLS compiler (we selected the compiler options to obtain almost the same F_{kernel} for the instances).

	Instance ID	1	2	3	4	5	6	7	8
Architecture	N	2,048	4,096	4,096	8,192	4,096	8,192	8,192	16,384
	P_{chip}	1		2		4		8	
	P_{comp}	8,192							
	P_{b} / P_{r} / P_{c}	8 / 32 / 32							
	P_{comm}	-		32					
Implementation	λ_{comp} [cyc]	48	48	47	47	47	47	47	47
	λ_{comm} [cyc]	-		30	30	28	28	27	27
Resource Usage	ALM	39%	40%	46%	46%	46%	46%	46%	46%
	ALUT	31%	30%	40%	40%	40%	40%	40%	40%
	Flip Flop	16%	17%	24%	24%	24%	24%	24%	24%
	BRAM	28%	56%	35%	91%	25%	53%	34%	91%
	DSP	7%	7%	7%	7%	7%	7%	7%	7%
Evaluated	N_{step} [cyc]	624	2,224	1,660	5,086	1,386	3,271	2,711	6,462
	N_{comp} [cyc]	624	2,224	1,199	4,399	687	2,351	1,327	4,655
	N_{comm} [cyc]	-		461	687	699	920	1,384	1,807
	$N_{\mathrm{comm\ w/o\ stall}}$	-		315	571	367	623	727	1,239
	N_{packet} [cyc]	-		64	128	32	64	32	64

Supplementary Figure 2: Implementation details (architecture parameters, resource usage, and evaluated parameters).

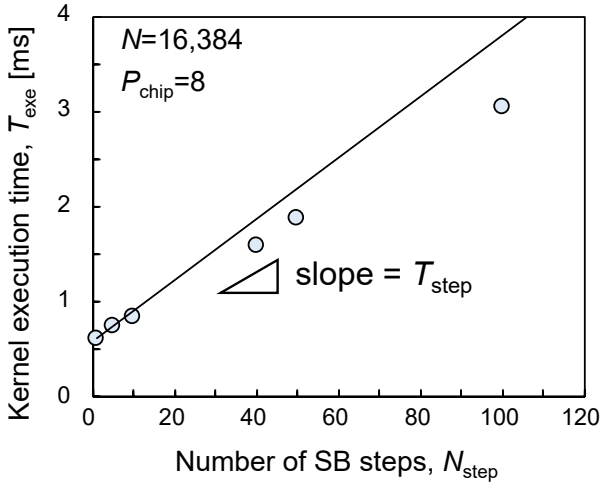
λ_{PHY} (including the cable delay and the latencies in TX/RX PHYs and QSFP+ optical modules) is a calibration parameter for our cluster simulator, estimated to be 283 ns by fitting to experimental data. This value is well aligned with the result of our preliminary experiment; we measured propagation delay from ch1 to ch2 in a simple loop-back configuration by using Intel SignalTap II logic analyser.

The resource utilization for the FPGA implementations is summarized also in Supplementary Fig. 2. Each FPGA in P_{chip}-FPGA cluster stores $N/P_{\mathrm{chip}} \times N$-size J data. On-chip memory resource (BRAM) for storing J data determines the maximum problem size (32 million coupling coefficients per chip).

In Supplementary Fig. 2, N_{step}, N_{comp}, and N_{comm} are shown in cycle count, which correspond to T_{step}, T_{comp}, and T_{comm}, respectively. Note that throughout the paper cycle count is based on the clock cycle of SB kernel (not on that of communication channel).

5 Measurement

We repeatedly measured SB kernel's execution time (T_{exe}) while varying N_{step}. Supplementary Figure 3 shows T_{exe} as a function of N_{step} for the eight-FPGA cluster. The slope of T_{exe}-N_{step} relationship gives T_{step} (24.7 μs per step in this case). The good linearity extending to $N_{\mathrm{step}} = 1$ means that the initial synchronization between FPGAs completes within one SB step and the autonomous synchronization mechanism works steadily after that.

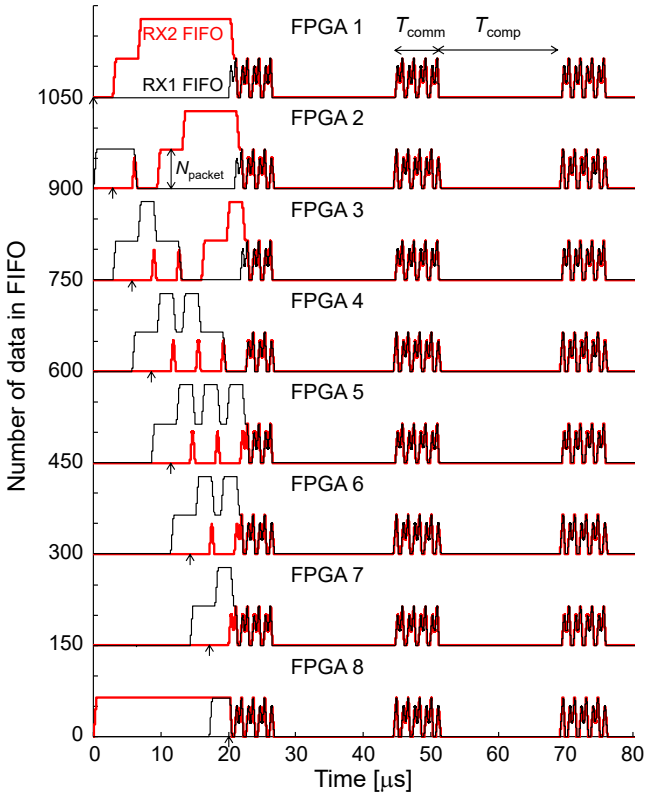


Supplementary Figure 3: Kernel execution time T_{exe} versus number of SB steps N_{step} ($N = 16,384$, $P_{\mathrm{chip}} = 8$)

6 Cluster simulator

For analysing the synchronization process of the FPGA-cluster for SB in cycle-by-cycle detail, we developed an in-house cluster simulator. It consists of independent software objects for SB kernel, TX FIFO, RX FIFO, RX PHY, and TX PHY, each of which works and communicates as described in Chip design (Main text). The simulator simulates the time-evolution process of the objects for all FPGAs in a single time domain. Supplementary Figure 4 shows the simulation results for the numbers of data in RX1/RX2 FIFOs as a function of time for the eight-FPGA cluster assuming $\lambda_{\mathrm{PHY}} = 283$ ns. SB kernels are launched every 2.86 μs (the SB kernel in the 8th FPGA is launched at 20 μs). The initial synchronization finishes within one communication phase, reproducing the experimental observation in Supplementary Fig. 3. λ_{stall} (stall cycles per communication subphase) extracted from the simulation results are summarized in Table 1 (Main text), in good agreement with the experimentally extracted ones. It is also confirmed by the simulations that λ_{stall} does not depend on launch timing.

The simulator has shown that if N_{packet} (number of clock cycles for one send/receive operation, see Supplementary Fig. 2) is comparable to λ_{PHY} (75 cycles when $F_{\mathrm{kernel}} = 266$ MHz, or 283 ns), λ_{stall} becomes a similar value to λ_{PHY}, and if N_{packet} is much larger than λ_{PHY}, λ_{stall} approaches to zero because of the buffer functions of FIFOs. Supplementary Figure 5 shows the effect of λ_{PHY} on λ_{stall}. Supplementary Figure 5a illustrates the timing chart for adjacent chips when $\lambda_{\mathrm{PHY}} > N_{\mathrm{packet}}$ assuming no time difference between the SB kernels. Even without the time difference, if λ_{PHY} is greater than N_{packet}, the SB kernel has to wait (stall) for $\lambda_{\mathrm{PHY}} - N_{\mathrm{packet}}$ ($= \lambda_{\mathrm{stall1}}$) cycles before starting the receive operation. In this case, since there is no data accumulation in the RX FIFO at the beginning of the receive operation, the speed of receive operation is limited by that of the communication channel, resulting in stall cycles ($\lambda_{\mathrm{stall2}}$) during the receive operation. In the case when λ_{PHY} is smaller than N_{packet} (Supplementary Fig. 5b), the SB kernel can move to the receive operation immediately after the send operation. However, if the data accumulation in the RX FIFO at the beginning of the receive operation is not enough, stall cycles can happen during the receive operation ($\lambda_{\mathrm{stall2}}$). As N_{packet} becomes relatively large with respect to λ_{PHY}, the number of stall cycles during the receive operation

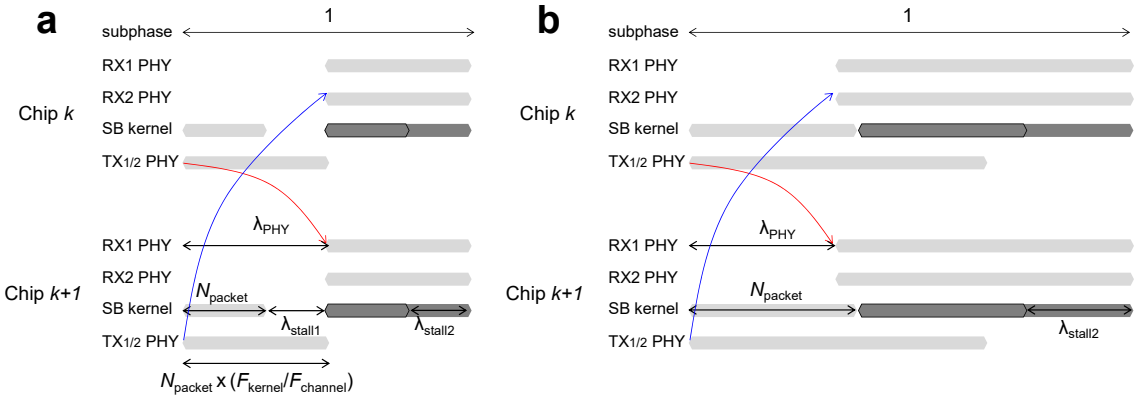


Supplementary Figure 4: Cluster simulator: Number of data in RX1/RX2 FIFOs (plus offsets for visibility) are shown as a function of time (for $N = 16,384$, $P_{\text{chip}} = 8$). Note that during the computation phase, there are no data in RX FIFOs. The launch time of each SB kernel is indicated by an arrow.

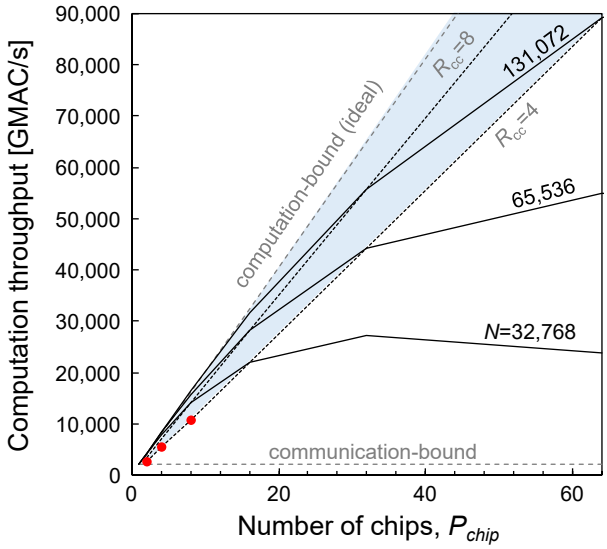
(λ_{stall2}) decreases because the data accumulation in the RX FIFO at the beginning of the receive operation increases; the buffer function of RX FIFO bridges the speed gap between the channel and SB kernel.

If we increase P_{chip} at a fixed N (strong scaling), the stall cycles can increase as the N_{packet} ($= N/P_{\text{chip}}/P_{\text{comm}}$) decreases and approaches to λ_{PHY}. Thus, λ_{PHY} can be a limiting factor in terms of performance for highly-parallelized architectures. Hence, minimizing λ_{PHY} (by adopting low-latency communication technology and designing a simple communication protocol) is important to enhance the maximum computational throughput attainable. On the other hand, if we increase both P_{chip} and N in the same proportion (weak scaling), N_{packet} is kept to be constant, leading to the weak-scaling characteristics (constant-efficiency scaling) of the proposed architecture.

Supplementary Figure 6 shows the computation throughput of the proposed architecture as a function of P_{chip} (ranging from 1 to 64) with N as a parameter. The computation-bound region of $R_{\text{cc}} \geq 4$ has been depicted similarly as in Fig. 4b (Main text). We can see a remarkable similarity for the scalability in both the scales of the smaller P_{chip} (up to 16, in Fig. 4b) and larger P_{chip} (up to 64, in Supplementary Fig. 6) regions, which again demonstrates the scale-out capability (weak scaling) of the cluster architecture.



Supplementary Figure 5: Effect of physical delay for transmission path (λ_{PHY}) on stall cycles per communication subphase (λ_{stall}): (**a**) Timing chart when $\lambda_{\mathrm{PHY}} > N_{\mathrm{packet}}$. (**b**) Timing chart when $\lambda_{\mathrm{PHY}} < N_{\mathrm{packet}}$.



Supplementary Figure 6: Scalability of the cluster architecture for SB: P_{chip} dependency of computation throughput with N as a parameter (for P_{chip} ranging from 1 to 64).

7 Simulated annealing

The simulated annealing algorithm [11] starts from a random initial state at a high temperature and then updates the state via a Monte Carlo (MC) process while the temperature is decreased. Each MC trial chooses a spin s_i and evaluates the energy difference (ΔE_i) for the flip of the spin:

$$\Delta E_i = 2s_i \sum_{j=1}^{N} J_{ij} s_j. \tag{3}$$

The spin s_i is flipped if $\beta \Delta E_i < -\ln \gamma$, where β is the inverse temperature and γ is a random number between 0 and 1. Simulated annealing must update the spin configuration one by one for convergence; once a spin is updated from s_k to s'_k based on the spin configuration $(s_1, \cdots, s_k, \cdots, s_N)$, the next spin must be updated based on the updated spin configuration $(s_1, \cdots, s'_k, \cdots, s_N)$.

One of parallel computation methods for simulated annealing [12] is performing forward computation of ΔE_i for all spins and then keeping ΔE_i up-to-date at each spin update; At the initial time, we compute ΔE_i for all spins in advance. Then, in the MC process, if $\beta \Delta E_i < -\ln \gamma$ for a chosen ith spin, we flip the ith spin and the sign of ΔE_i, and update ΔE_j for all j other than i as

$$\Delta E_j \leftarrow \Delta E_j - 4J_{ij} s_i s_j, \tag{4}$$

where s_i is the previous value before its flip. Otherwise (if $\beta \Delta E_i > -\ln \gamma$ for a chosen ith spin), we do no computation. The updates of ΔE_i can be performed simultaneously and therefore accelerated by parallel processing. This method can accelerate simulated annealing, especially at low temperatures (when the acceptance ratio for each trial flip is low). In the best case, we prepare N number of spin units ($P_{\text{comp}} = N$), each of which has a processing module for updating ΔE_i and a register for storing ΔE_i. Since the simultaneous update of multiple spins is not allowed, once a spin is flipped, the information has to be sent to all spin units. In the best case, we have broadcast wires to all spin units ($P_{\text{comm}} = N$). In the best case ($P_{\text{comp}} = N$, $P_{\text{comm}} = N$), the computation-to-communication ratio for simulated annealing is $\mathcal{O}(1)$.

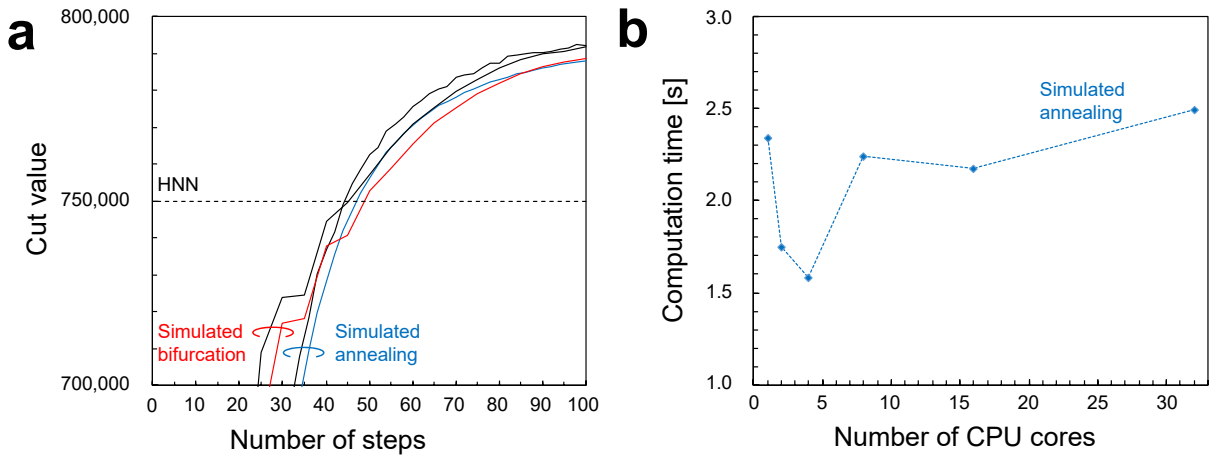
8 Hopfield neural network

The Hopfield neural network (HNN) algorithm [13] can be regarded as the most naïve local search and is equivalent to simulated annealing at zero temperature (without noise effects), that is, the trial flip is accepted if $\Delta E_i < 0$. The cut value (energy value) achieved by HNN is used in this work as a standard when comparing more advanced methods. We repeatedly solved $\text{K}_{16,384}$ by the HNN algorithm with a long iteration. The cut value attainable by HNN was 749,888 on average over 100 trials with different initial states.

9 Implementation of SB and simulated annealing for $\text{K}_{16,384}$

We define a time-evolution *step* for simulated annealing as a procedure repeating the MC trial N times. The computational complexity of the *step* procedure is $\mathcal{O}(N^2)$, thus similar to that of the SB *step* procedure. The control parameters for SB and simulated annealing were optimized so that final cut values at N_{step}=100 were maximized. The SB parameters are $\alpha_0 = \beta_0 = 1$, $\gamma_0 = 0.7\alpha_0/\sqrt{N_{\text{prblm}}}$, $\Delta t = 0.5$, and $M = 2$. The control parameter $\alpha(t)$ increases from 0 to 1. The initial values of x_i and p_i are, respectively, zero and random values between -0.1 and 0.1. The control parameter β for simulated annealing increases linearly from 0.0, reaching 0.04 at the final step iteration.

Supplementary Figure 7a shows the cut values by SB and simulated annealing as a function of time evolution steps (corresponding to the amounts of pairwise interaction computations) for $\text{K}_{16,384}$. There are two lines for each of SB and simulated annealing; the coloured one indicates the averaged curve over 100 trials with different initial states and the black one corresponds to the best envelope. It is found that the amounts of pairwise interaction computations needed for similar-quality solutions



Supplementary Figure 7: Implementation of simulated annealing for $K_{16,384}$: (**a**) Cut values by simulated bifurcation and simulated annealing as a function of time evolution steps. The computational complexities per time-evolution step for simulated bifurcation and simulated annealing are in the same order of magnitude [$\mathcal{O}(N^2)$]. (**b**) Computation time of simulated annealing for $N_{\rm step}$=100 versus the number of CPU cores.

are not different between SB and simulated annealing. The remarkable speed-up for SB shown in Fig. 5 (Main text) comes from the higher computational throughput for SB.

We implemented simulated annealing on a PC cluster, which is comprised of multiple nodes linked via InfiniBand. Each node has two 14-core processors (Intel Xeon E5-2697 v3) sharing eight 16 GB memories. We adopted the acceleration techniques called "forward computation of ΔE_i" and "deterministic traversal" in Ref. [12]. Supplementary Figure 7b shows the computation time of simulated annealing for $N_{\rm step}$=100 versus the number of CPU cores. For $K_{16,384}$, the optimal number of cores is four; larger numbers of cores result in no speed-up because of large communication overheads.

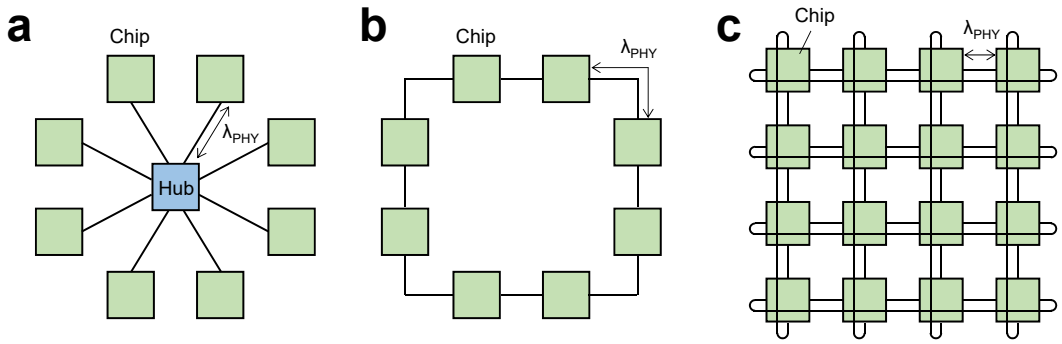
10 Comparisons

To highlight the proposed scale-out architecture of SB-based Ising machines with all-to-all spin-spin couplings, we discuss about all-to-all communication networks, conventional N-body simulators, simulated annealing-based Ising machines, and optical Ising machines.

10.1 All-to-all communication networks

The scale-out capability of the proposed multi-chip architecture depends on the communication part, which corresponds to all-to-all communication between chips. From the viewpoint of application to the proposed architecture, we compare the network topologies for all-to-all communication (Supplementary Fig. 8). In the portioned SB implemented with a $P_{\rm chip}$-size cluster, each chip is responsible for updating $N/P_{\rm chip}$ oscillators and shares all the position information of N oscillators with the other chips through multiple communication subphases ($N_{\rm subphase}$).

First, we consider a star network with a central hub (Supplementary Fig. 8a). To realize a similar autonomous synchronization mechanism instead of complex mechanisms like synchronization-acknowledgement protocols, the hub is required to have not only broadcast ability but also data buffering and arbiter functions adequate for collision-free operation. In that case, each chip can complete the send operation without stall cycles and then move to the receive operation (for $P_{\rm chip}$ subphases). Assuming each chip has communication buffers large enough for storing N-size data, the synchronization between chips would be achieved by a stall mechanism for the emptiness of communication buffers. Obviously, the size of the cluster is limited by the capability of the central hub. Additional latencies caused by intermediation of the hub and address management needed for each chip (each chip has to know the source of received data) can limit the scalability in throughput.



Supplementary Figure 8: Network topologies for all-to-all communication: (**a**) Star network for $P_{\text{chip}} = 8$. (**b**) Ring network for $P_{\text{chip}} = 8$ (this work). (**c**) Two-dimensional torus network for $P_{\text{chip}} = 16$.

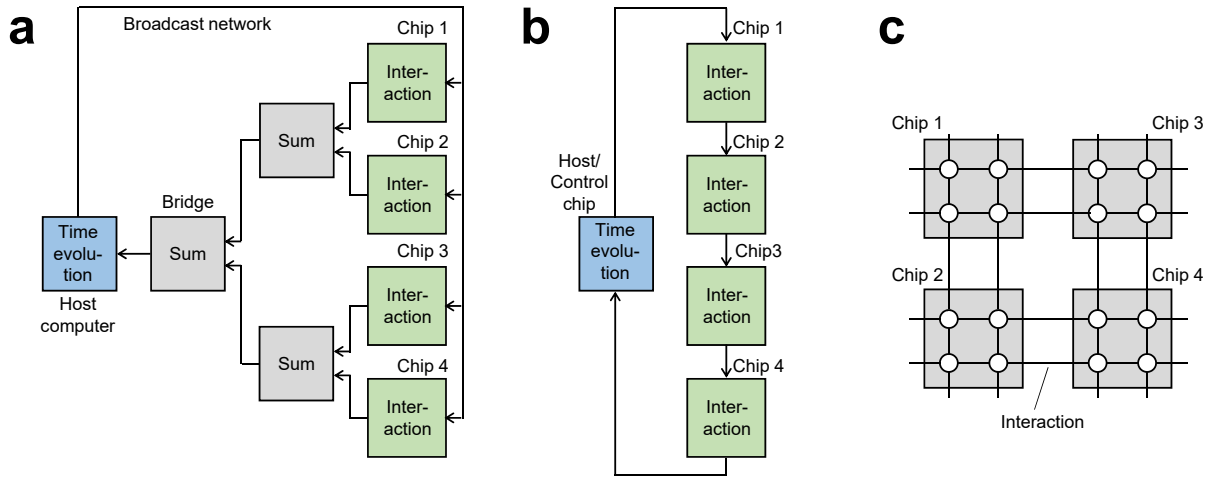
Second, consider a ring network without a centralized control node (Supplementary Fig. 8b). This is our choice in this work. In the ring topology where nodes are homogeneous and symmetric, each chip can obtain all the oscillator information through iterating an exchange process of oscillator information between neighbouring chips for P_{chip} subphases. All chips can carry out the exchange processes simultaneously. The synchronization of adjacent nodes is attained based on the stall mechanism for the emptiness of communication buffers for the oscillator exchange, and then the fronts of local synchronization propagate to adjacent nodes every communication subphase, achieving global synchronization. The physical delay between chips has been minimized in terms that there is no repeater between chips and no need for address management. To realize the global synchronization without a centralized control node, each chip must be autonomous. That is, all the communication and computation modules needed for an oscillator subsystem in SB must be integrated in each chip.

Third, consider a two-dimensional (2D) torus network (Supplementary Fig. 8c). A 2D torus network is an enhanced version of a one-dimensional torus (a ring): the nodes in a column are connected by a ring network and the nodes in a row are also connected by another ring network. It would be possible to implement a similar communication/synchronization mechanism with a 2D torus. To complete information sharing with all the chips, all the chips first carry out exchange processes of oscillator information between neighbouring chips in the column direction for $\sqrt{P_{\text{chip}}}$ subphases and then perform exchange processes in the row direction also for $\sqrt{P_{\text{chip}}}$ subphases. Thus, the number of communication subphases is reduced from P_{chip} (for a ring network) to be $2\sqrt{P_{\text{chip}}}$ (for a 2D torus). The reduction in the number of subphases leads to a further decrease in the communication overheads. Note that λ_{comm} and λ_{stall} are overheads per subphase. The advantage of 2D torus network versus ring network can appear for large clusters (plausibly, $P_{\text{chip}} \geq 16$) and increase with increasing P_{chip}. Implementation with a 2D torus is a possible future work.

10.2 N-body simulators

N-body simulations simulate the motions of N particles interacting with each other over a number of discrete time steps. When interaction considered is based on a long-range pairwise force such as gravitational or Coulomb forces, the N-body simulation has an algorithm structure similar to full-connection SB; the computation procedure for one time step consists of interaction part with complexity $\mathcal{O}(N^2)$ and time-evolution part with complexity $\mathcal{O}(N)$. The interaction part computes, for each particle, the joint force acting on it by summation of pairwise forces with all the other particles. The time-evolution part updates the status of each particle based on the joint force. A figure of merit for N-body simulators is the number of pairwise interactions evaluated per second (pairs/s).

Our multi-chip architecture for SB is different from conventional multi-chip architectures for N-body simulations [14–17] and unique in terms of the high degree of total computation parallelism $P_{\text{comp}}^{\text{tot}}$ (up to 65,536) and synchronization capability enabling one time step T_{step} of simulation in microseconds (24.7 μs for P_{chip}=8, N=16,384 with 256 million connections), achieving an outstanding throughput (up to 10,870 Gpairs/s). Note that a pairwise gravitational-force (or Coulomb-force) computation needs 38 to 40 floating-point operations [18], while the evaluation of a pairwise interaction



Supplementary Figure 9: Multi-chip architectures for N-body simulations: (**a**) Reduction-tree network. (**b**) Uni-directional ring network. (**c**) Two-dimensional array network.

for Ising formulation is one MAC operation. The simple computation of Ising interactions enables increasing the degree of computational parallelism remarkably, which necessitates more sophisticated synchronization mechanisms to increase the computational throughput while avoiding saturation due to communication overheads. Many N-body simulators have been designed for a very large N with compared to $P_{\rm comp}^{\rm tot}$ attained, so less sensitive to synchronization/communication capabilities.

The most famous project for N-body simulators would be the GRAPE project [14,15]. Among various attempts, GRAPE-6 [14] is the representative of parallel-computing and reduction architectures of the GRAPE project, where multiple chips computing partial forces are connected to a reduction tree network (Supplementary Fig. 9a) composed of hierarchical sum units, and the final joint forces output from the reduction tree are sent to a host computer updating the status of particles. To redistribute the updated data of particles to the multiple chips, the GRAPE-6 system has a broadcast network. The theoretical peak speed of single-host GRAPE-6 system ($P_{\rm chip}$=128, $P_{\rm comp}^{\rm tot}$=768) is 3.94 TFlops (69 Gpairs/s) for a very large N ($> 10^6$).

Some multi-FPGA architectures for N-body simulation [16,17] are characterized by uni-directional ring networks (Supplementary Fig. 9b), where cascaded interaction processing chips are connected with a time-evolution chip (or a host). Guo *et al.* [16] have reported a heterogeneous multi-FPGA accelerator ($P_{\rm chip}$=5, $P_{\rm comp}^{\rm tot}$=106) with a throughput of 504 GFlops (13 Gpairs/s) for N=131,072. Huthmann *et al.* [17] have demonstrated a throughput of 47 Gpairs/s for N=524,288 with a stream computing system ($P_{\rm chip}$=8, $P_{\rm comp}^{\rm tot}$=256).

10.3 Simulated annealing-based Ising machines

In simulated annealing, if spin-spin couplings are limited to adjacent spins, it is possible to update multiple spins simultaneously for only isolated spins. Yoshimura *et al.* [19] has reported a single-chip simulated annealing-based accelerator consisting of a two-dimensional array of spin unit circuits with a local connectively (King's graph), where the spin variables are separated into four independent groups and each group is updated simultaneously in one of four operation phases. Such a two-dimensional array of spin unit circuits may be partitioned with multiple chips (Supplementary Fig. 9c). Takemoto *et al.* [20] has presented an extended version of Yoshimura's work, which is composed of two chips synchronized by sharing a clock signal. The system corresponds to $P_{\rm chip}$=2, $P_{\rm comp}^{\rm tot}$=15,488, N=61,952 with 0.5 million connections, 2,253 Gpairs/s, and $T_{\rm step}$=220 ns.

10.4 Optical Ising machines

Inagaki *et al.* [3] has presented a 2000-node coherent Ising machine (CIM), which is an optical parametric oscillator network equipped with a measurement and feedback system that uses two FPGAs

to compute spin-spin interactions. All the system components (optical systems and electrical systems) are synchronized by clock distribution. For a 2,000-node complete-graph MAX-CUT problem named K_{2000}, the CIM needs 5 μs per step (N spin update) and the computation throughput using two Xilinx Virtex-7 (XC7VXV690T) FPGAs is 800 Gpairs/s (or GMAC/s). For the same problem, a single-FPGA SB [10] has a 2.3 times higher computation throughput and requires 6.2 times fewer computations to get below the HNN energy, leading to 14 times faster time-to-target-solution.

Pierangeli *et al.* has reported several variations of spatial-photonic Ising machines (SPIMs) [21–23], where binary phases on the wavefront of an amplitude-modulated laser beam controlled by a spatial light modulator (SLM) correspond to artificial spins and the spin interactions are evaluated in the optical system. Updating of spins based on the measured interference intensity according to some annealing algorithms are realized by a measurement and feedback system consisting of a CCD camera, a processor, and an SLM controller. The time per step (N spin update) is limited by the camera rate (greater than a few milliseconds per image), SLM response, and data processing [21]. An SPIM [22] improves the convergence to the Ising ground state by changing the spin couplings adiabatically.

References

[1] Goto, H., Tatsumura, K. & Dixon, A. R. Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems. *Science Advances* **5**, eaav2372 (2019).

[2] Goemans, M. X. & Williamson, D. P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)* **42**, 1115–1145 (1995).

[3] Inagaki, T., Haribara, Y., Igarashi, K., Sonobe, T., Tamate, S., Honjo, T., Marandi, A., McMahon, P. L., Umeki, T., Enbutsu, K., Tadanaga, O., Takenouchi, H., Aihara, K., Kawarabayashi, K., Inoue, K., Utsunomiya, S. & Takesue, H. A coherent Ising machine for 2000-node optimization problems. *Science* **354**, 603–606 (2016).

[4] Matsumoto, M. & Nishimura, T. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. on Modeling and Computer Simulation* **8**, 3–30 (1998).

[5] Aval Data Corporation. APX-AA10L1: PCI Express Compatible FPGA accelerator Board. *http://aval-global.com/?page_id=1817*, (2018).

[6] Intel Corporation. Intel Arria 10 Core Fabric and General Purpose I/Os Handbook. *https://www.intel.com/*, (2018).

[7] Intel Corporation. Intel FPGA SDK for OpenCL Pro Edition: Programmnig Guide (ver 17.1). *https://www.intel.com/*, (2018).

[8] Avago Technologies. AFBR-79EQDZ: 40 Gigabit Ethernet QSFP+ Pluggable, Parallel Fiber-Optics Module. *https://www.avagotech.com/*, (2014).

[9] Super Micro Computer, Inc. SuperServer 4028GR-TR2. *https://www.supermicro.com/*, (2018).

[10] Tatsumura, K., Dixon, A. R. & Goto, H. FPGA-based Simulated Bifurcation Machine. In *IEEE Int'l Conf. on Field-Programmable Logic and Applications (FPL)*, 59–66 (2019).

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

[12] 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).

[13] Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences* **79**, 2554–2558 (1982).

[14] Makino, J., Fukushige, T., Koga, M. & Namura, K. GRAPE-6: Massively-parallel special-purpose computer for astrophysical particle simulations. *Astronomical Society of Japan* **55**, 1163–1187 (2003).

[15] Makino, J. & Daisaka, H. GRAPE-8–An accelerator for gravitational N-body simulation with 20.5 Gflops/W performance. In *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 1–10 (2012).

[16] Guo, S., Wang, T., Tao, L., Tian, T., Xiang Z. & Jin, X. RP-ring: A Heterogeneous Multi-FPGA Accelerator. *International Journal of Reconfigurable Computing* **2018**, 6784319 (2018).

[17] Huthmann, J., Shin, A., Podobas, A., Sano, K. & Takizawa, H. Scaling Performance for N-Body Stream Computation with a Ring of FPGAs. In *International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies (HEART)*, 1–6 (2019).

[18] Del Sozzo, E., Rabozzi, M., Di Tucci, L., Sciuto, D. & Santambrogio, M. D. A scalable FPGA design for cloud n-body simulation. In *IEEE Int'l Conf. on Application-specific Systems, Architectures and Processors (ASAP)*, 1–8 (2018).

[19] Yoshimura, C., Hayashi, M., Okuyama T. & Yamaoka, M. Implementation and evaluation of FPGA-based annealing processor for Ising model by use of resource sharing. *International Journal of Networking and Computing* **7**, 154–172 (2017).

[20] Takemoto, T., Hayashi, M., Yoshimura C. & Yamaoka M. A 2×30k-Spin Multi-Chip Scalable Annealing Processor Based on a Processing-In-Memory Approach for Solving Large-Scale Combinatorial Optimization Problems. *IEEE Journal of Solid-State Circuits*, 1–12 (2019).

[21] Pierangeli, D., Marcucci, G. & Conti, C. Large-Scale Photonic Ising Machine by Spatial Light Modulation. *Phys. Rev. Lett.* **122**, 213902 (2019).

[22] Pierangeli, D., Rafayelyan, M., Conti, C., & Gigan, S. Scalable spin-glass optical simulator. Preprint at https://arxiv.org/abs/2006.00828 (2020).

[23] Pierangeli, D. and Marcucci, G. & Conti, C. Adiabatic evolution on a spatial-photonic Ising machine. *Optica* **7**, 1535-1543 (2020).