18-819F: Introduction to Quantum Computing 47-779/47-785: Quantum Integer Programming & Quantum Machine Learning

Novel Approaches to Solving Ising Models

Lecture 17

2021.11.02

Electrical & Computer ENGINEERING

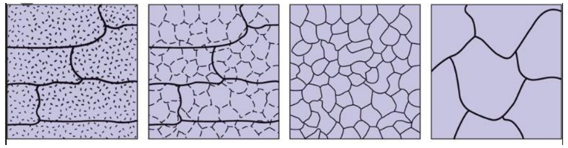
TEPPER

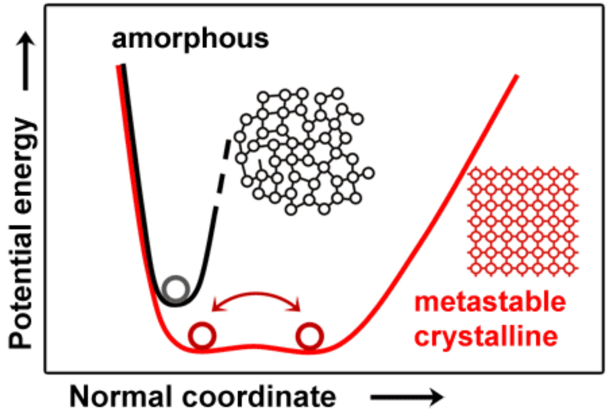
USRA Universities Space Research Association

Agenda

- Refresher of Simulated Annealing
- Benchmarking exercise
- Conventional vs. Natural Computing
- Solving the 2D regular Ising Problem
  - Graphic Processing Units
  - Tensor Processing Units
  - Field-programmable gate arrays
- Solving general Ising models
  - Graphic Processing Units
  - Simulated Bifurcation Machine
  - CMOS
  - Digital Annealers

Simulated Annealing

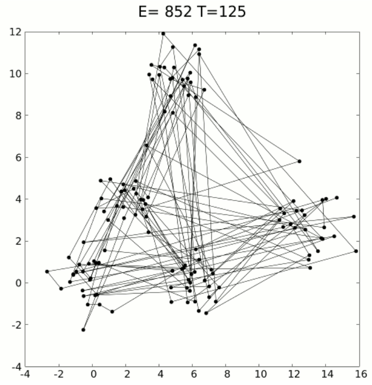




Concept coming from annealing in metallurgy

Slow cooling allows for perfect crystals (minimizing energy)

Start at effective high temperature and gradually decrease the temperature by increments until is slightly above zero



Interesting behavior:

- "Divide-and-conquer": Big features are solved early in the search and small features later while refining
- Ability to escape local-minima
- Guaranteed to reach lowest energy if temperature is lowered slowly enough

[1] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing.Science,220(4598):671–680, 1983.
[2] https://www.esrf.eu/news/general/phase-change-materials/index_html
[3] Alan Lang Chapter 8 Strain hardening and annealing.

Ising Model – Monte Carlo, Physics Inspired Methods

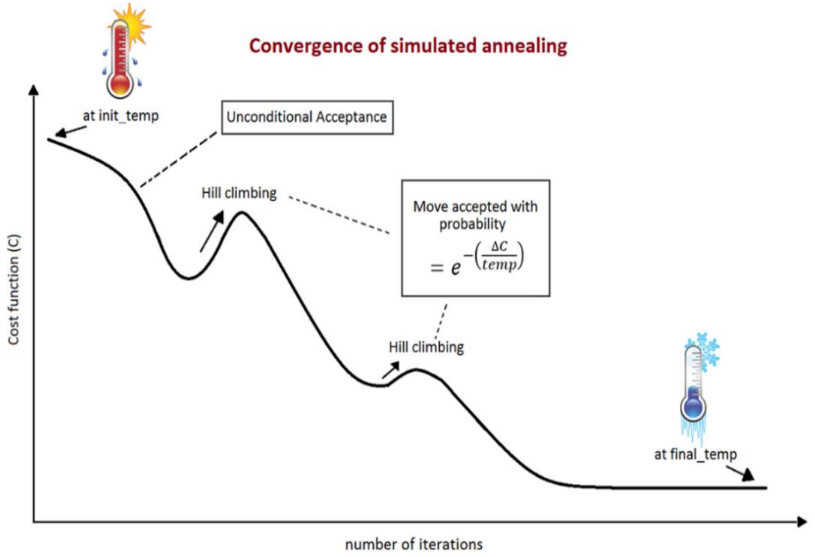
Ising model as Markov-Chain

The immediate probability $P(\boldsymbol{\sigma}^c, \beta) = e^{-\beta H(\boldsymbol{\sigma}^c)}/Z(\beta)$ of transitioning to a future state $\boldsymbol{\sigma}^f$ depends only in the current state $\boldsymbol{\sigma}^c = [\sigma_1^c, \cdots, \sigma_N^c]$

Given single flip dynamics, we can jump from any state to another.

Metropolis-Hastings Monte Carlo Algorithm for Ising Models

1) Start with a known configuration, $\boldsymbol{\sigma}^i = [\sigma_1^i, \cdots, \sigma_N^i]$ corresponding energy $H(\boldsymbol{\sigma}^i)$ and temperature value $T = (k_B\beta)^{-1}$
2) Randomly change the configuration
   - Flip some spins $\sigma^i \rightarrow \sigma^j$
1) Calculate new energy value $H(\boldsymbol{\sigma}^j)$
2) Compare to energy at previous position
   - If , $H(\boldsymbol{\sigma}^j) < H(\boldsymbol{\sigma}^i)$ keep new position
   - If , $H(\boldsymbol{\sigma}^j) > H(\boldsymbol{\sigma}^i)$ keep new position if Boltzmann factor for transition satisfies $\exp\left[-\frac{H(\boldsymbol{\sigma}^j)-H(\boldsymbol{\sigma}^i)}{k_B T}\right] \geq \mathrm{Rand}\,[0,1]$
1) Repeat 2) - 4) K times



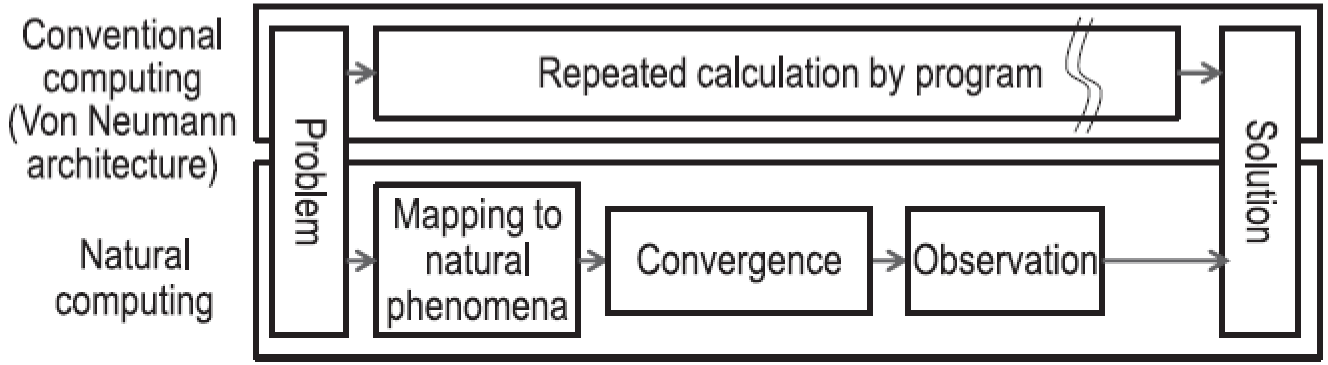
[1] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing.Science,220(4598):671–680, 1983.

Benchmarking Exercise

Let’s go to Colab

https://colab.research.google.com/github/bernalde/QuIPML/blob/master/notebooks/Notebook%204%20-%20Benchmarking.ipynb

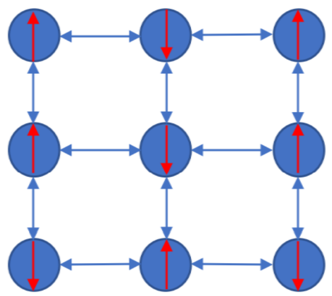
Conventional (Von Neumann) vs. Natural Computing

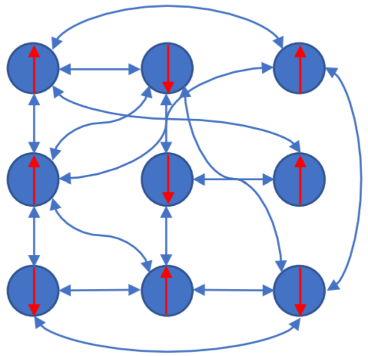


2D Ising model - Simple yet interesting

Arbitrary Ising - Applicable but hard!

Ising Models





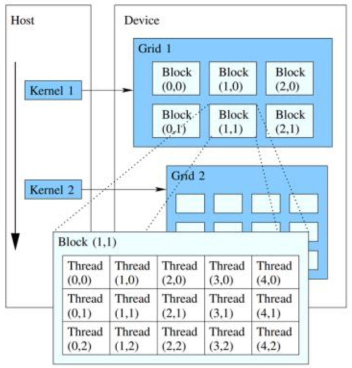
Main concern: How to parallelize Monte Carlo Simulations

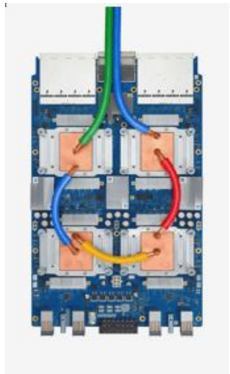
Main concern: How to actually solve NP-Hard Problem

[1] A 20k-Spin Ising Chip to Solve Combinatorial Optimization Problems With CMOS Annealing. Yamaoka, Yoshimura, Hayashi, Okuyama, Aoki, and Mizuno
[2] https://arxiv.org/pdf/1807.10750.pdf

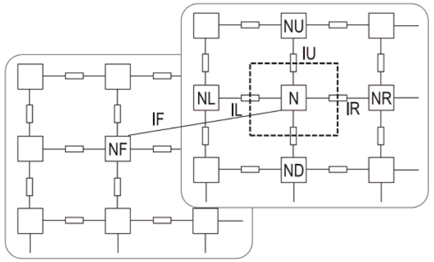
Specialized hardware for solving Ising/QUBO

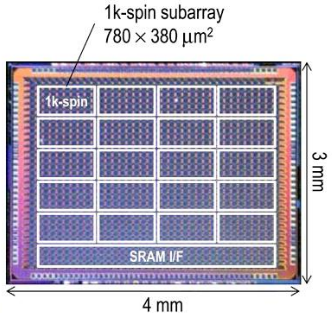
GPUs and TPUs



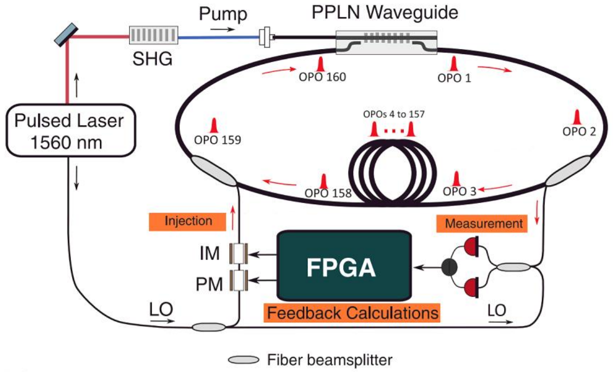


Complementary metal-oxide semiconductors (CMOS)

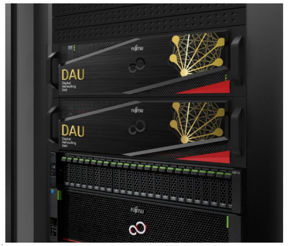




Coherent Ising Machines (CIM)



Digital annealers

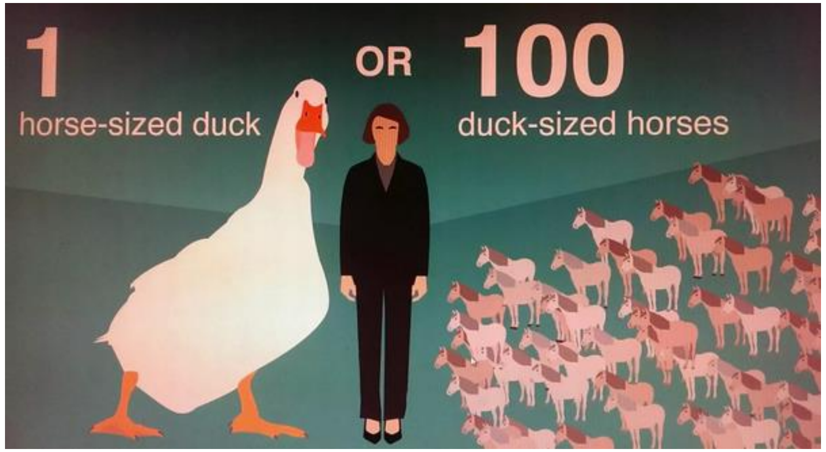


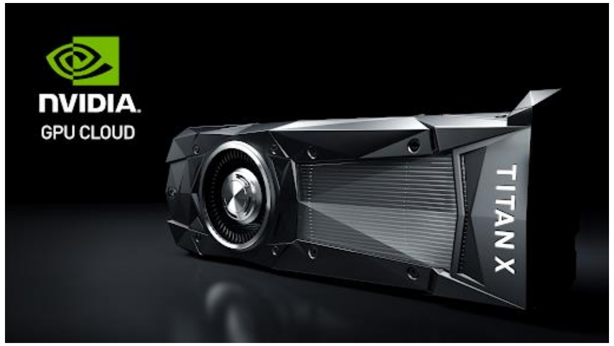
[1]https://arxiv.org/pdf/1807.10750.pdf
[2]https://arxiv.org/pdf/1903.11714.pdf
[3]https://arxiv.org/pdf/1806.08815.pdf
[4]https://spectrum.ieee.org/tech-talk/computing/hardware/fujitsus-cmos-digital-annealer-produces-quantum-computer-speeds
[5]https://science.sciencemag.org/content/sci/354/6312/614.full.pdf

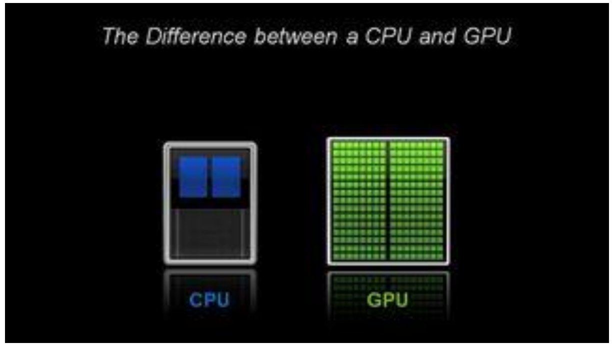
Graphical Processing Units (GPU)

CPU vs GPU

CPU	GPU
Central Processing Unit	Graphics Processing Unit
Several cores	Many cores
Low latency	High throughput
Good for serial processing	Good for parallel processing
Can do a handful of operations at once	Can do thousands of operations at once







Specialized, electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images.
Their highly parallel structure makes them more efficient than general purpose central processing units (CPUs) for algorithms that process large blocks of data in parallel.

[1] https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/
[2] https://www.quora.com/Would-you-rather-fight-100-duck-sized-horses-or-one-horse-sized-duck
[3] https://en.wikipedia.org/wiki/Graphics_processing_unit

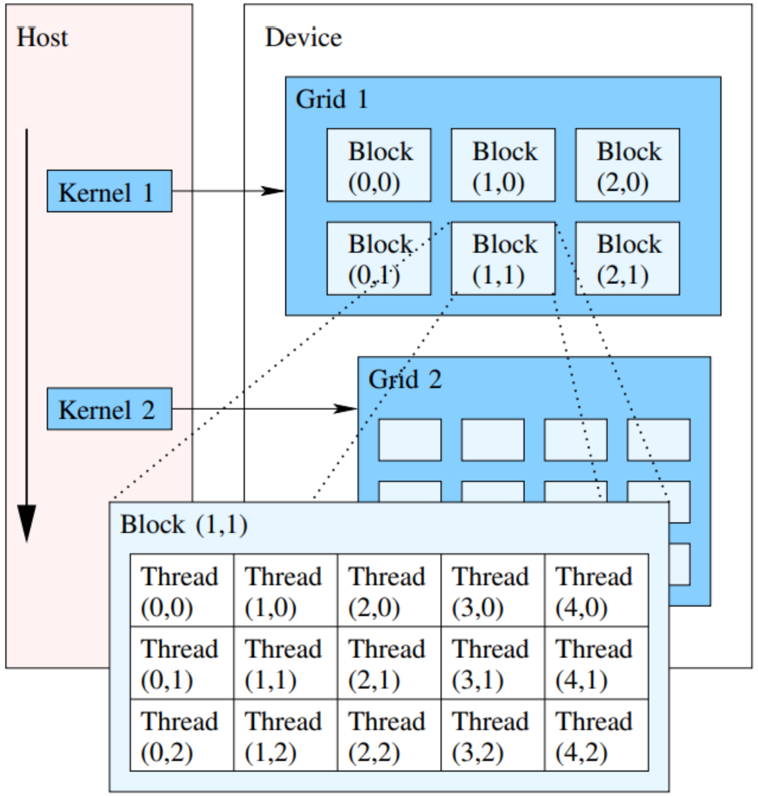
Graphical Processing Units

Algorithm 1 Modified Ising annealing algorithm

1: input: $(M, N, \mathbf{S})$
2: initialize all σ_i in $\mathbf{S}$
3: **for** sweep-id **in** $\{1, 2, \ldots, M\}$ **do**
4:     **for** σ_i **in S do**
5:         $\sigma_i \leftarrow \text{argmin}(H(\sigma_i))$ based on (2)
6:     **end for**
7:     randomly choose and flip N spin glasses in $\mathbf{S}$
8:     decrease N
9: **end for**

Algorithm 2 GPU Simulated Annealing method for Ising model

input: $(F_p, \mathbf{S})$
initialize ALL σ_i in $\mathbf{S}$
while $F_p > 0$ **do**
    for all $\sigma_i \in \mathbf{S}$ **in parallel do**
        $\sigma_i \leftarrow \text{argmin}(H(\sigma_i))$
        flip σ_i with probability F_p
    end for
    reduce F_p
end while



[1] https://arxiv.org/pdf/1807.10750.pdf

GPU for 2D Ising Models

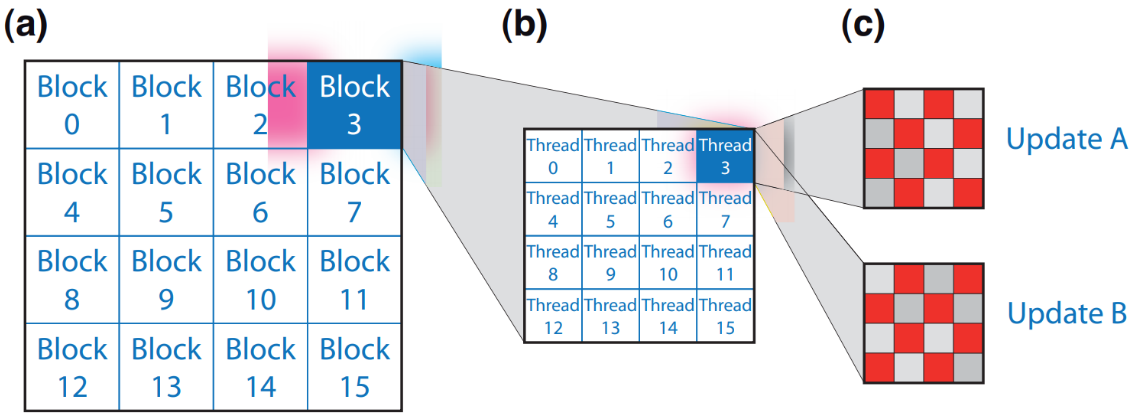


Figure 1: (Color online) The spin lattice is processed by a variable number of blocks (a), where each block runs a variable number of threads (b). The threads update the spin lattice in two steps, A and B, using two kernel invocations (c).

Encode spins into threads and classify +1 and -1 in separate buckets

Perform the Ising Update via easily parallelizable operations.

	Spinflips per μs	Relative speed
CPU simple	26.6	0.11
CPU multi-spin coding	226.7	1.00
shared memory	4415.8	19.50
shared memory (Fermi)	8038.2	35.46
multi-spin unmodified	3307.2	14.60
multi-spin coding on the fly	5175.8	22.80
multi-spin coding linear	7977.4	35.20

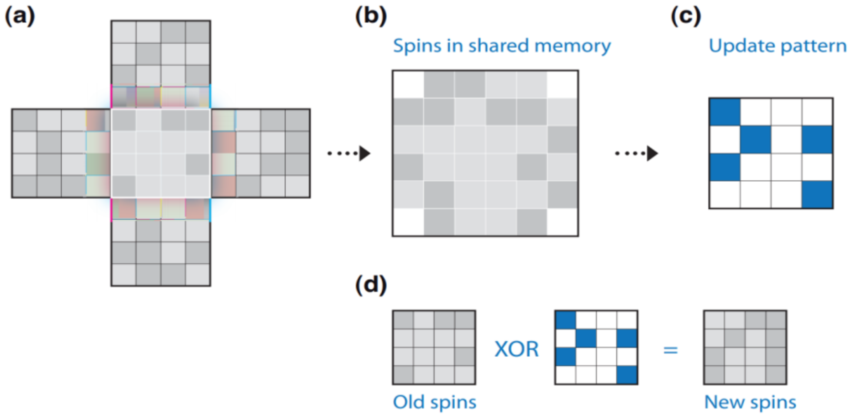


Figure 2: (Color online) (a) The way a kernel processes a 4×4 meta-spin. (b) Spins are extracted into shared memory and an update pattern is created (c). (d) Afterwards, the new spins are obtained using the update pattern (Spins on blue sites will be flipped, spins on white sites will not be flipped), and written back to global memory.

[1] https://arxiv.org/pdf/1007.3726.pdf

Correctness of GPU's results

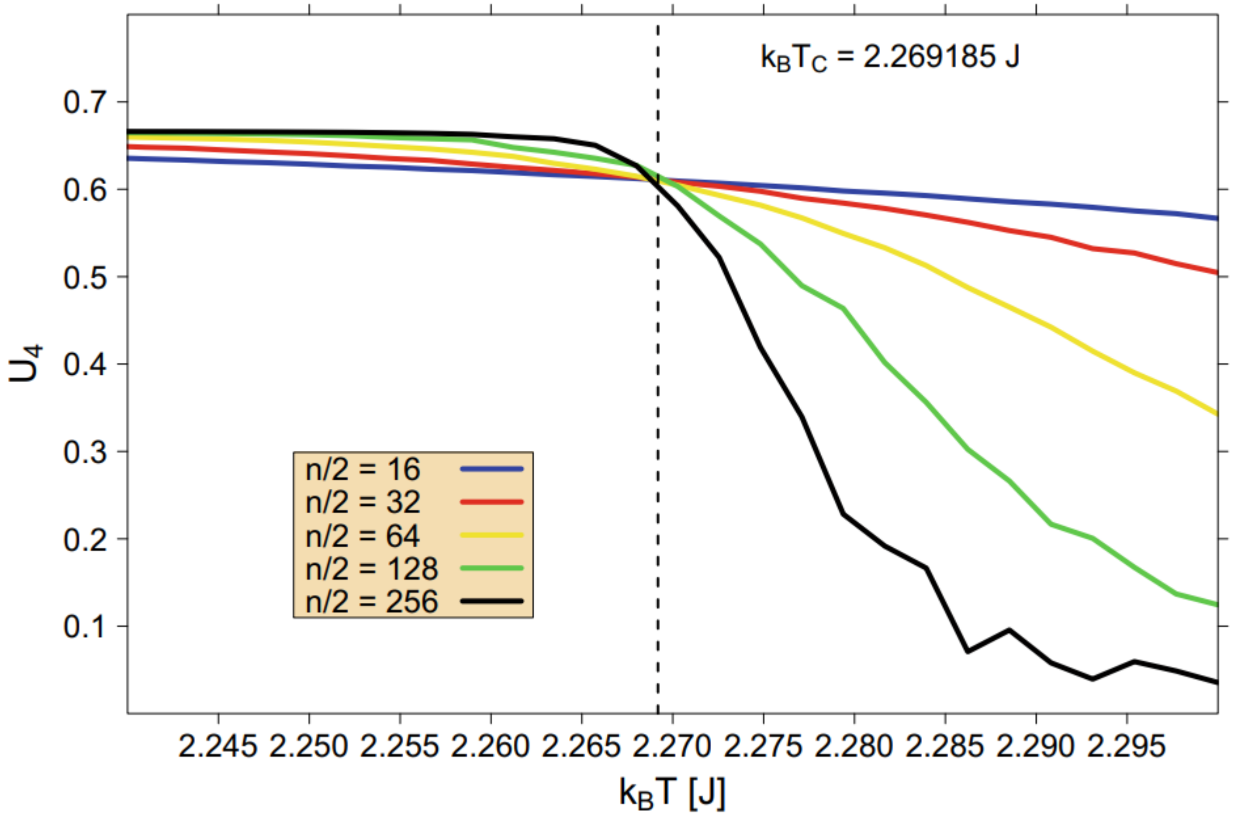


Fig. 5. Binder cumulant U_4 in dependence of k_BT for various numbers n of spins per row and column of the two dimensional square lattice Ising model. $n/2$ corresponds to the involved number of threads per block on the GPU implementation. The curves of the Binder cumulants for various system sizes $N = n^2$ cross almost perfectly at the critical temperature derived by Onsager [3], which is shown additionally as a dashed line. In each temperature step, the average was taken over 10^7 measurements.

It (sort of) matches Onsager analytical prediction!

[1] https://arxiv.org/pdf/1007.3726.pdf

Parallelizing GPUs

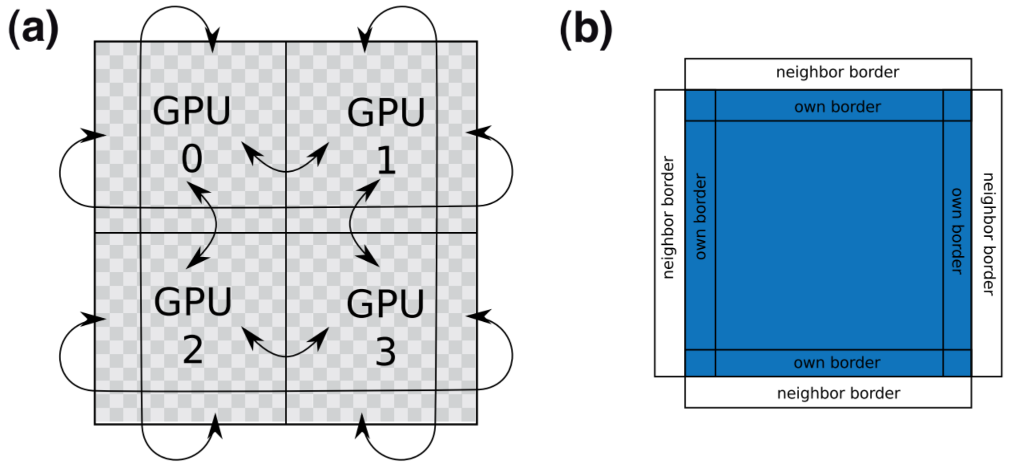


Figure 4: (Color online) (a) Each GPU processes a "meta-spin" lattice of size $N = n^2$. The lattices are aligned on a super-lattice, and the outer borders are connected via periodic boundary conditions. In this example, 4 GPUs process a system of $2^2 \cdot N$ spins. (b) A meta-spin update needs the 4 nearest neighbor meta-spins. On the borders of a lattice, each GPU needs the spin information of the neighboring lattices. The border information has to be passed between the GPUs. In our implementation this is done by using 8 neighbor arrays.

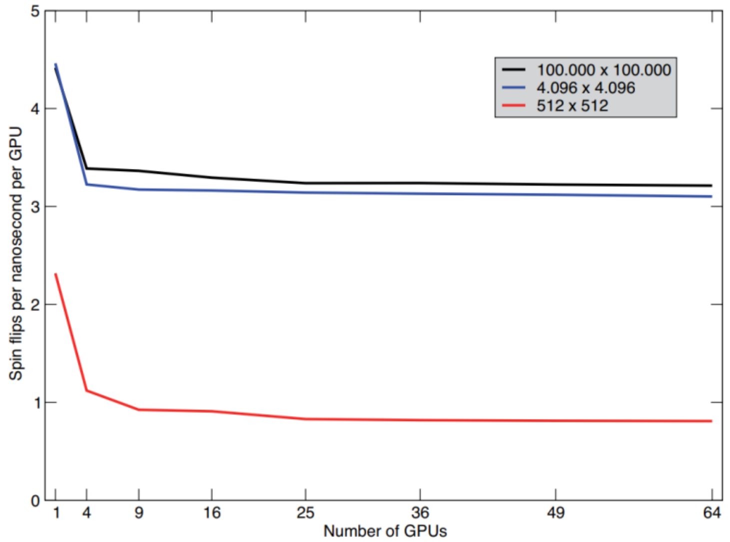
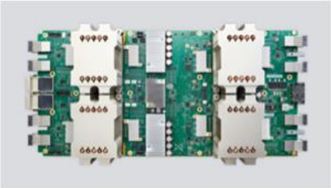


Figure 5: (Color online) Cluster performance for various system sizes (per GPU). For more than one GPU, spin flip performance scales nearly linearly with the amount of GPUs. Again, optimal performance is reached at a lattice size of about 4096 × 4096 per GPU. Using 64 GPUs, a performance of 206 spinflips per nanosecond can be achieved on a 800.000 × 800.000 lattice.

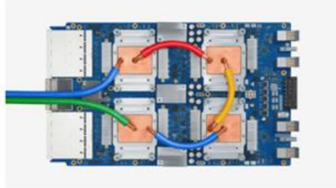
Using 64 GPUs performance of 206 spinflips per **nanosecond** can be achieved on a 800,000x800,000 lattice

[1] https://arxiv.org/pdf/1007.3726.pdf

Tensor Processing Units (TPU)



Cloud TPU v2
180 teraflops
64 GB High Bandwidth Memory (HBM)

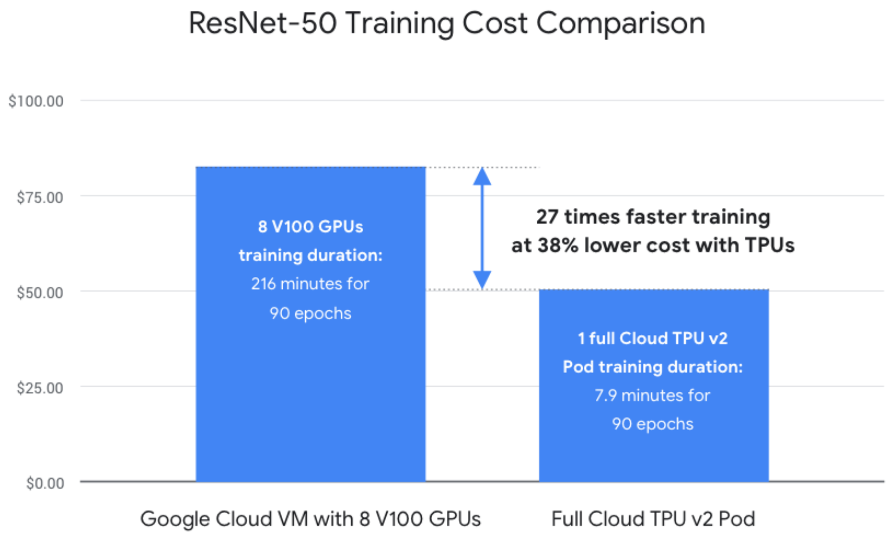


Cloud TPU v3
420 teraflops
128 GB HBM

Cloud TPU v2 Pod
11.5 petaflops
4 TB HBM
2-D toroidal mesh network

Cloud TPU v3 Pod
100+ petaflops
32 TB HBM
2-D toroidal mesh network

Machine learning performance and benchmarks



Tensor Processing Unit (**TPU**) is an AI accelerator application-specific integrated circuit (ASIC) developed by Google specifically for neural network machine learning, particularly using Google's own TensorFlow software.

[1] https://en.wikipedia.org/wiki/Tensor_Processing_Unit#/media/File:Tensor_Processing_Unit_3.0.jpg
[2] https://cloud.google.com/tpu

TPU for 2D Ising

Checkerboard Algorithm: break lattice in sublattices and group equal spins to easily operate on them

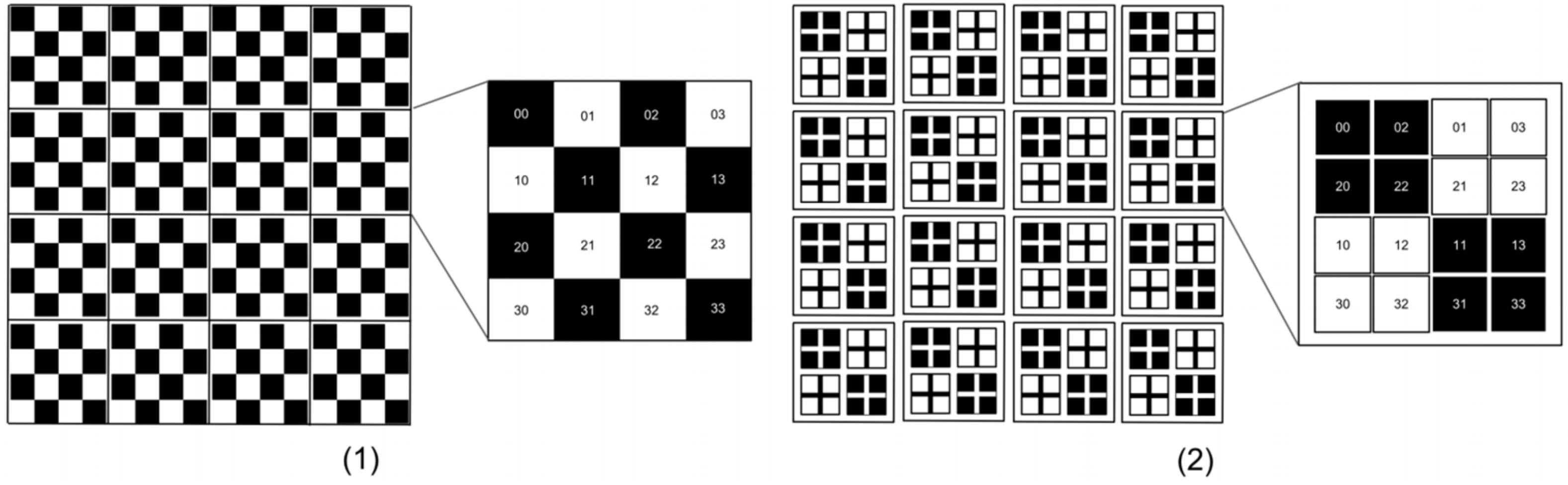
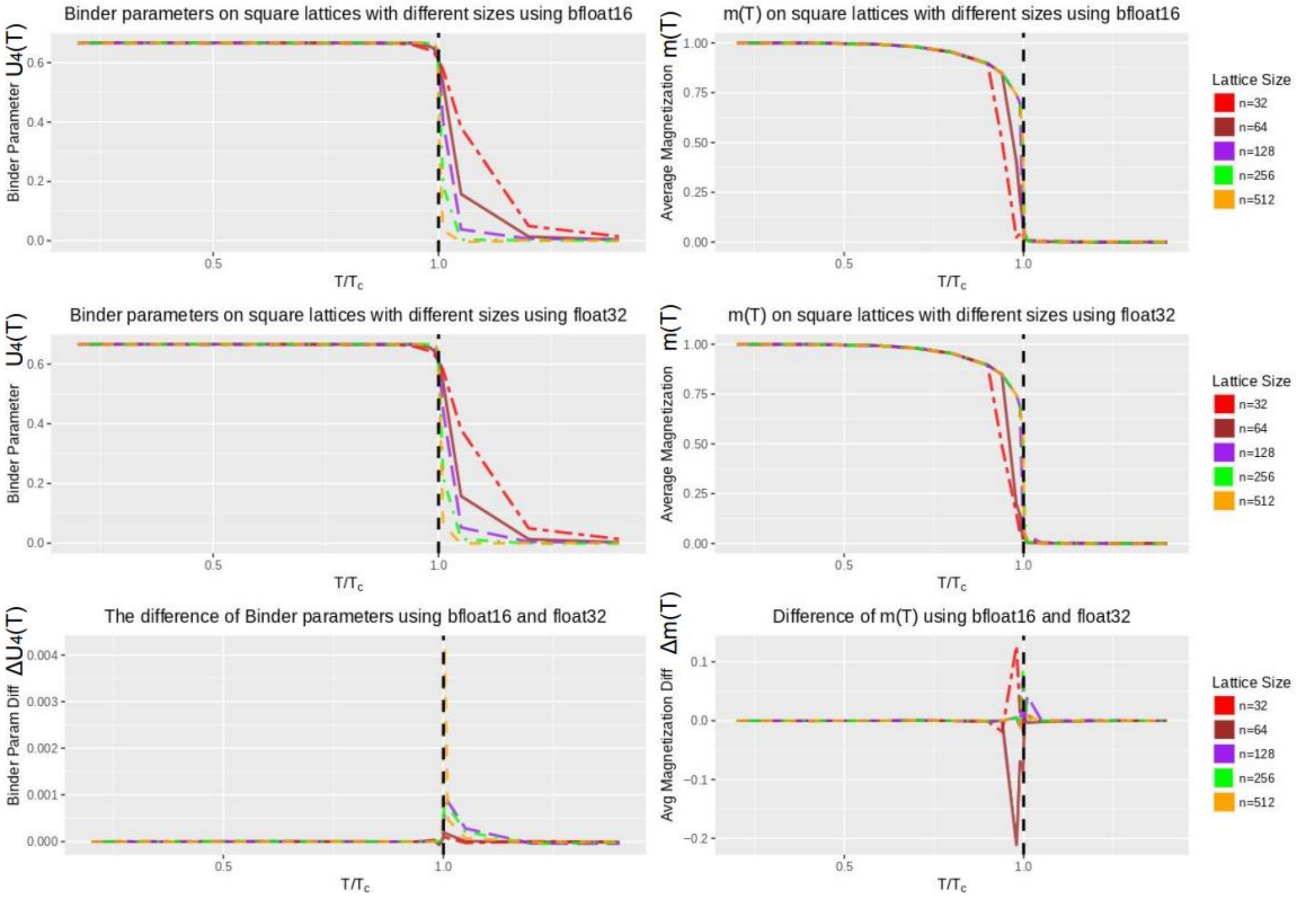


Figure 3: A 2-d checkerboard: (1) Original checkerboard: on the left, the 16×16 board is split into a 4×4 grid of 4×4 sub-lattices, i.e., it is represented by a $[4, 4, 4, 4]$ tensor, where $[l, k, :, :]$ is the sub-lattice at $[l, k]$ of the grid; on the right, the sub-lattice is zoomed in and the indices of its spin sites are shown; (2) Reorganized checkerboard: one the left, each 4×4 sub-lattice is reorganized by 4 "compact" 2×2 sub-lattices; on the right, 4 "compact" 2×2 sub-lattices are zoomed in and their original indices from the 4×4 sub-lattice are shown. In general, such alternate coloring of black and white can be extended to lattices with any dimensions.

[1] https://arxiv.org/pdf/1903.11714.pdf

Correctness of TPU’s results



It (sort of) matches Onsager analytical prediction!

[1] https://arxiv.org/pdf/1903.11714.pdf

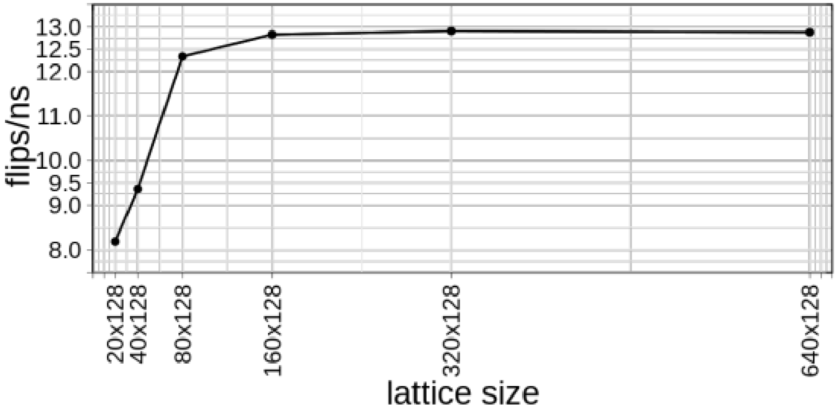
Electrical & Computer ENGINEERING

TEPPER

USRA Universities Space Research Association

Efficiency of TPU cluster

lattice size n^2	(flips/ns)	(nJ/flip)
$(20 \times 128)^2$	8.1920	12.2070
$(40 \times 128)^2$	9.3623	10.6811
$(80 \times 128)^2$	12.3362	8.1062
$(160 \times 128)^2$	12.8266	7.7963
$(320 \times 128)^2$	12.9056	7.7486
$(640 \times 128)^2$	12.8783	7.7650
GPU in [23, 3]	**7.9774**	–
Nvidia Tesla V100	**11.3704**	21.9869
FPGA in [20]	**614.4**	–



Better performance and less energy consumption than Nvidia GPUs, until… (next slide)

Really far from Field-programmable gate array (FPGA)! (a couple slides more)

Code available in Github and replicable results (with a Google Cloud account)
https://github.com/google-research/google-research/blob/master/simulation_research/ising_model/ising_mcmc_tpu.ipynb

[1] https://arxiv.org/pdf/1903.11714.pdf

Nvidia’s Rebuttal!

lattice size	flip/ns
$(1 \times 2048)^2$	231.09
$(2 \times 2048)^2$	318.95
$(4 \times 2048)^2$	379.27
$(8 \times 2048)^2$	411.65
$(16 \times 2048)^2$	420.44
$(32 \times 2048)^2$	420.77
$(64 \times 2048)^2$	418.23
$(123 \times 2048)^2$	417.53
1 TPUv3 core in [7]	12.91
32 TPUv3 cores in [7]	336.01
FPGA (1024^2) [8]	614.40[1]

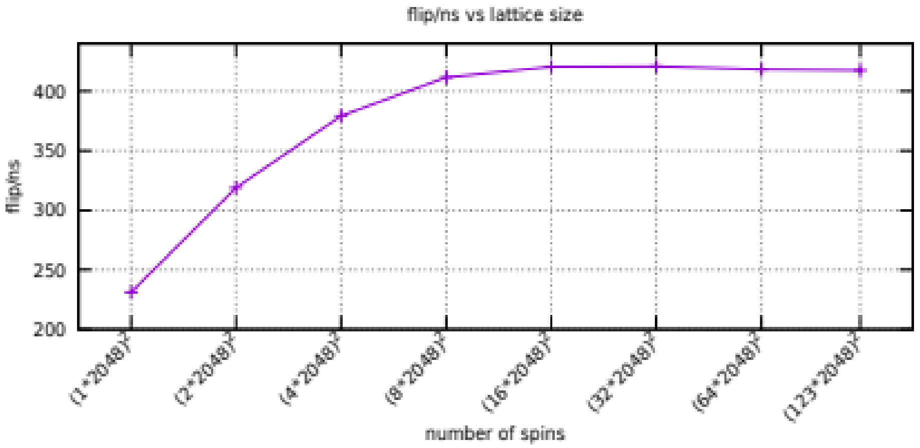


Table 2: Flips per nanosecond obtained by the optimized multi-spin code on a single Tesla V100-SXM card with different lattice sizes, requiring an amount of memory ranging from 2MB to 30GB. For comparison purposes, the table also reports the best timings with 1 and 32 TPUv3 cores from [7], and with 1 FPGA from [8].

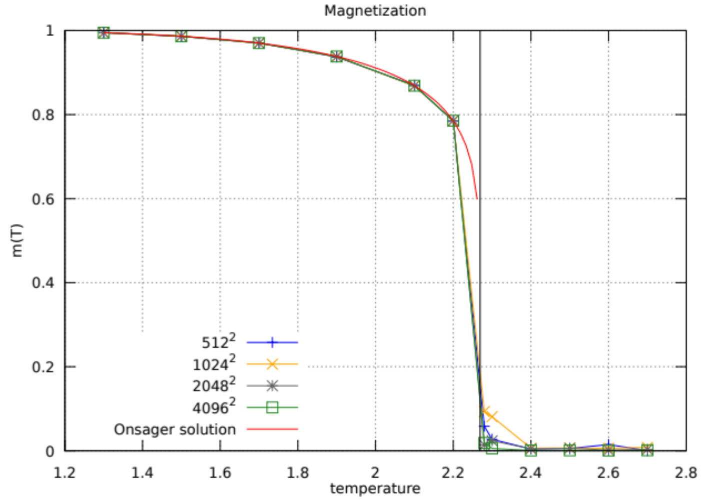


Figure 5: Steady state magnetization measures obtained with the multi-spin code for lattice sizes 512^2, 1024^2, 2048^2, and 4096^2. The solid vertical line marks the critical temperature value $T_c = 2.269185$.

Better performance than TPUs

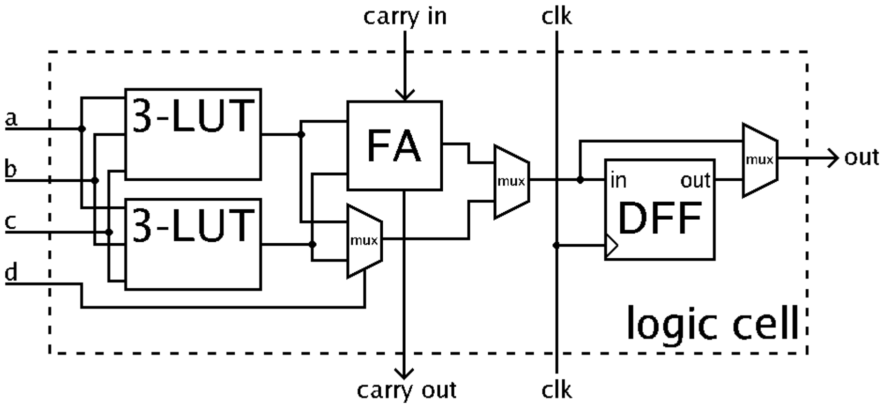
And of course, they compare against Onsager

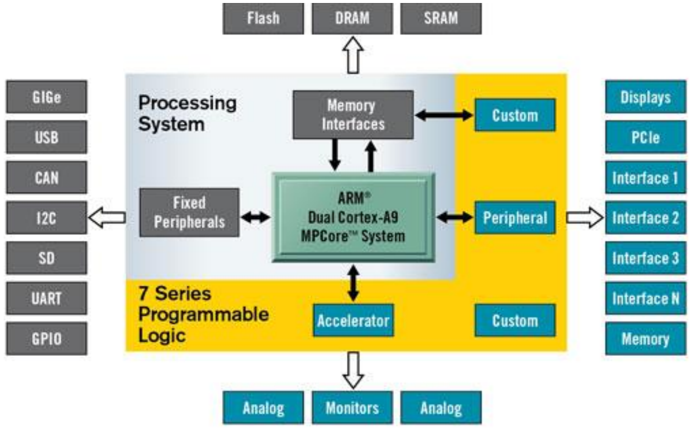
Still far from Field-programmable gate array (FPGA)! (next slide)

Code available in Github and replicable results https://github.com/NVIDIA/ising-gpu

[1] https://arxiv.org/pdf/1906.06297.pdf

Field-programmable gate array (FPGA)

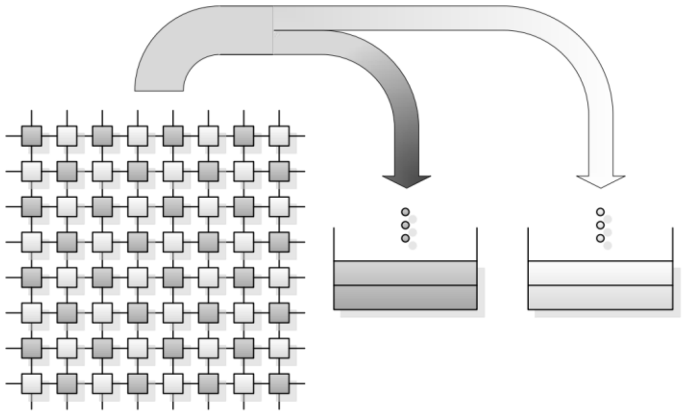




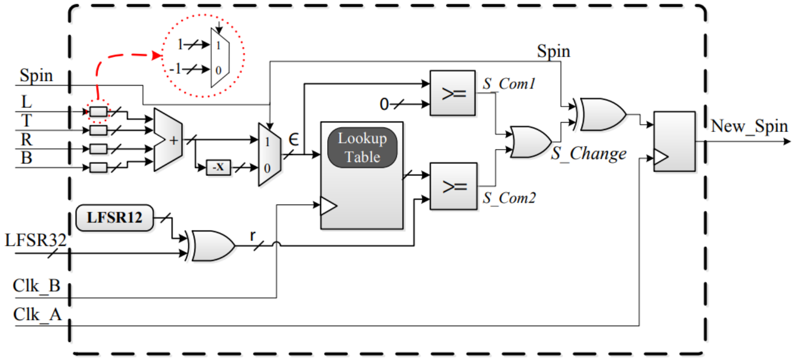
A **field-programmable gate array** (**FPGA**) is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence the term "field-programmable"... Circuit diagrams were previously used to specify the configuration, but this is increasingly rare due to the advent of electronic design automation tools.

[1] https://en.wikipedia.org/wiki/Field-programmable_gate_array
[2] https://arxiv.org/pdf/1602.03016.pdf

FPGA for 2D Ising Models



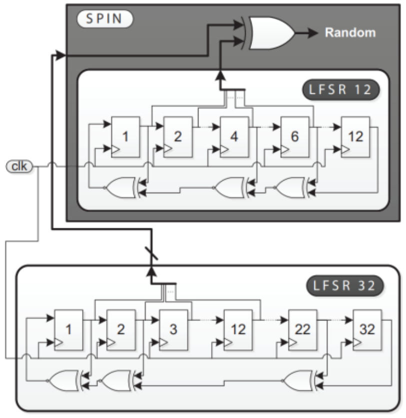
Checkerboard Algorithm diagram



Circuit Diagram Single Spin

Platform	# updated spins	Ratio
CPU	62	1
Single GPU	7977	129
Previous FPGA	94127	1518
64 GPUs	206000	3322
Our FPGA	614400	9909

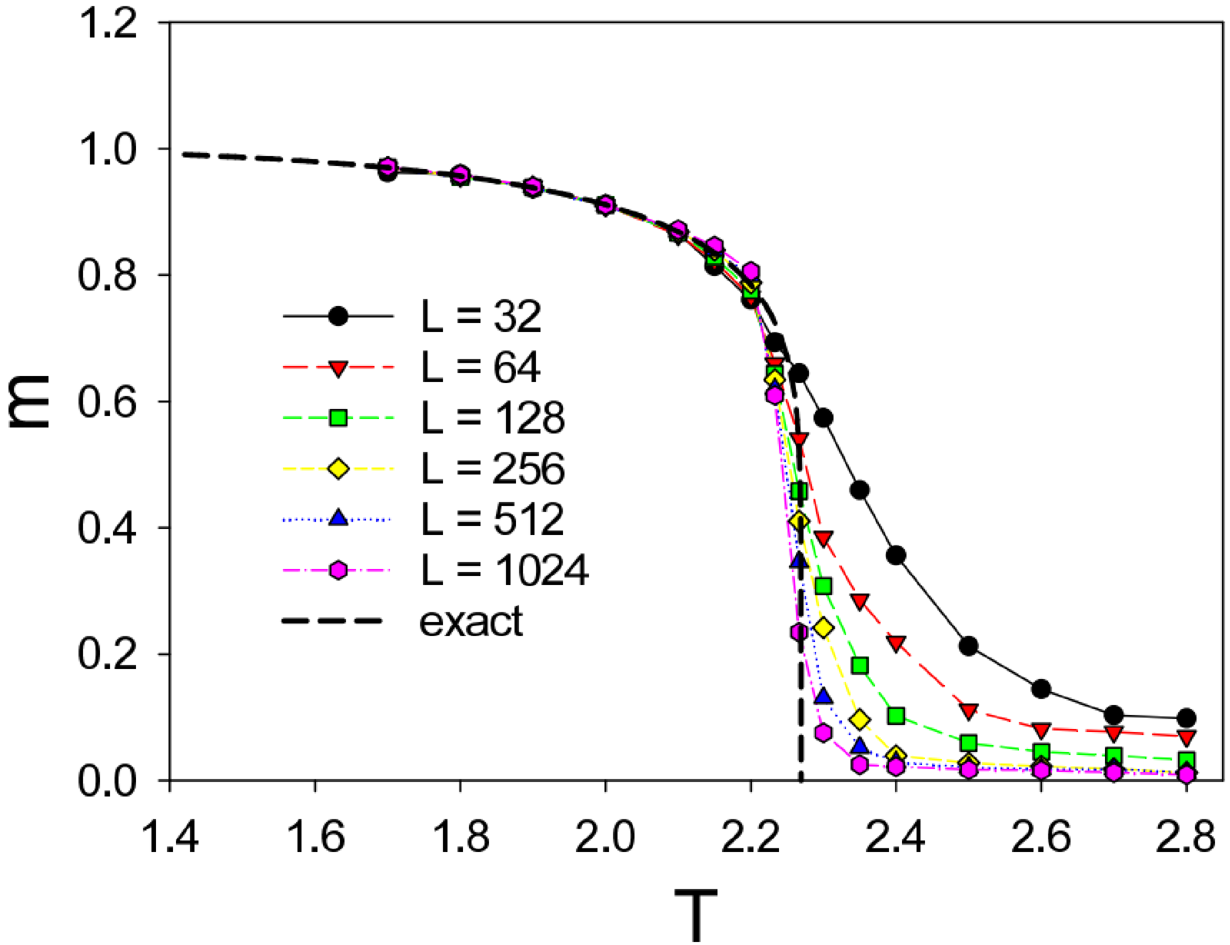
Number of spinflips per microsecond
for the 1024x1024 lattice



Circuit Diagram Random Number Generation

[1] https://arxiv.org/pdf/1602.03016.pdf

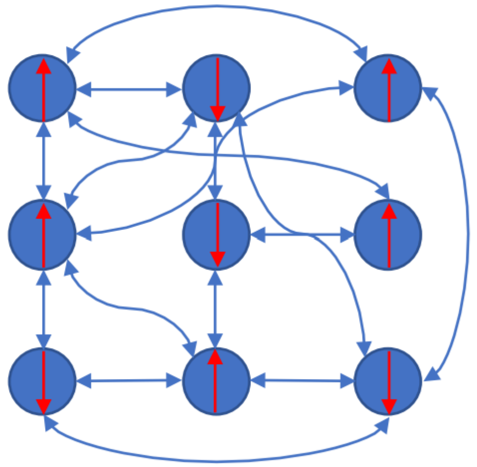
Correctness of FPGA’s results



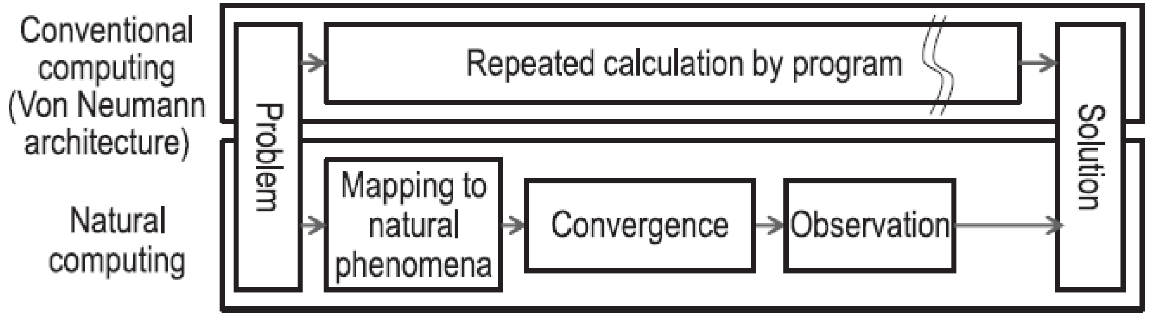
It (sort of) matches Onsager analytical prediction!

[1] https://arxiv.org/pdf/1602.03016.pdf

Working with general Ising Models



Main concern: How to actually solve NP-Hard Problem?



Using a natural computing approach you would ideally use Adiabatic Quantum Computing, and realistically Quantum Annealing

Schrödinger equation

$$i\eta \frac{d}{dt}|\psi\rangle = H|\psi\rangle$$

$$H(\sigma_1, \sigma_2, \cdots, \sigma_n) = -\frac{1}{2}\sum_i \sum_j J_{i,j}\sigma_i\sigma_j + \sum_i h_i\sigma_i$$

One cannot efficiently solve this equation using classical computers (if so, why would we need quantum computers after all!)

The issue then relies on (classically) Simulating Quantum Annealing

[1] A 20k-Spin Ising Chip to Solve Combinatorial Optimization Problems With CMOS Annealing. Yamaoka, Yoshimura, Hayashi, Okuyama, Aoki, and Mizuno
[2] https://arxiv.org/pdf/1807.10750.pdf

Graphical Processing Units

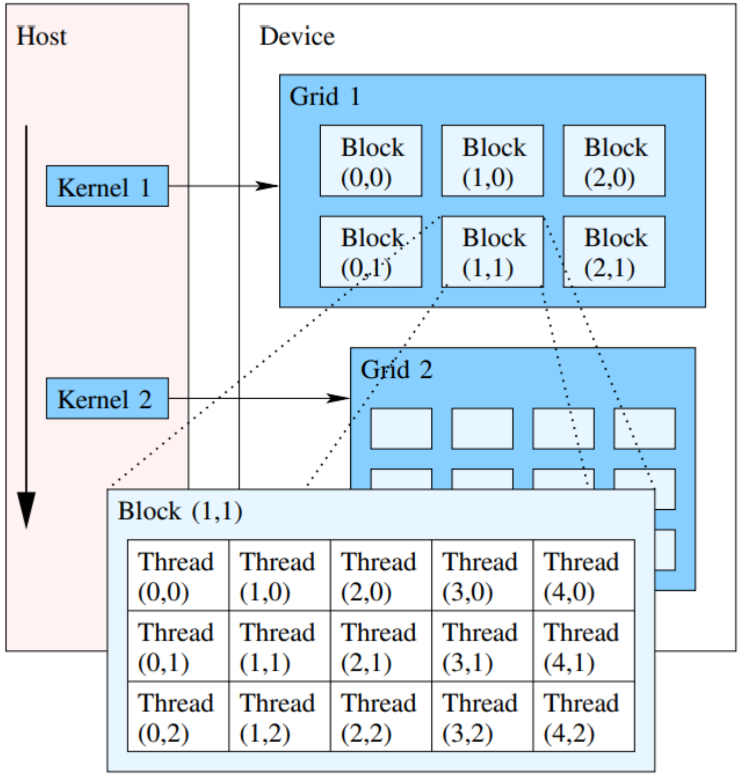
Algorithm 1 Modified Ising annealing algorithm

input: $(M, N, \mathbf{S})$
initialize all σ_i in $\mathbf{S}$
for sweep-id **in** $\{1, 2, \ldots, M\}$ **do**
  for σ_i **in** $\mathbf{S}$ **do**
    $\sigma_i \leftarrow \mathrm{argmin}(H(\sigma_i))$ based on $\mathcal{H}_i(\sigma_i) = \left(-\sum_j J_{i,j}\sigma_j - h_i\right)\sigma_i$
  end for
  randomly choose and flip N spin glasses in $\mathbf{S}$
  decrease N
end for

Algorithm 2 GPU Simulated Annealing method for Ising model

input: $(F_p, \mathbf{S})$
initialize ALL σ_i in $\mathbf{S}$
while $F_p > 0$ **do**
  for all $\sigma_i \in \mathbf{S}$ **in parallel do**
    $\sigma_i \leftarrow \mathrm{argmin}(H(\sigma_i))$
    flip σ_i with probability F_p
  end for
  reduce F_p
end while

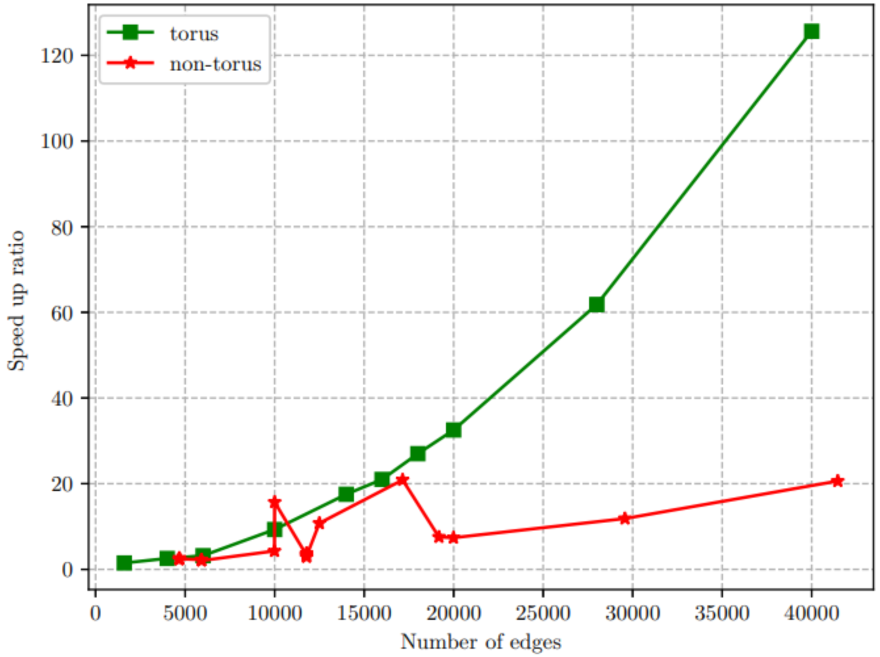
"One may notice that since each spin glass may have a different number of neighbors, then the threads will not be perfectly load balanced."

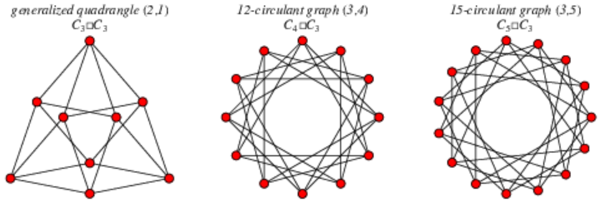


[1] https://arxiv.org/pdf/1807.10750.pdf Cook, Zhao, Sato, Hiromoto

GPU Performance

	Days	Seconds
# edges	CPLEX cut	GPU cut (%accuracy)
9999	9473	8884 (93.78%)
14999	13357	12776(95.65%)
24998	20206	19981(98.88%)
49995	35248	36228(100.29%)
39998	33605	32914(97.94%)
59997	46371	46510(100.29%)
99995	70566	72009(102.04%)
199990	128448	131930(102.71%)
249995	176556	179391(101.60%)
374993	248505	255078(102.64%)
626988	392912	400540(101.94%)
1249975	741709	751050(101.25%)





[1] https://arxiv.org/pdf/1807.10750.pdf
[2] https://mathworld.wolfram.com/TorusGridGraph.html

Simulated Bifurcation Machine

"The method in [previous slide] ignores the data dependencies to implement parallel computation on fully connected spin models. Since the modified algorithm in [previous slide] does not follow the mathematical model that the Quantum Monte Carlo is based on, the output of the simulation could deviate from the optimum."

How can we efficiently simulate quantum annealing?

We can take a classical approximation

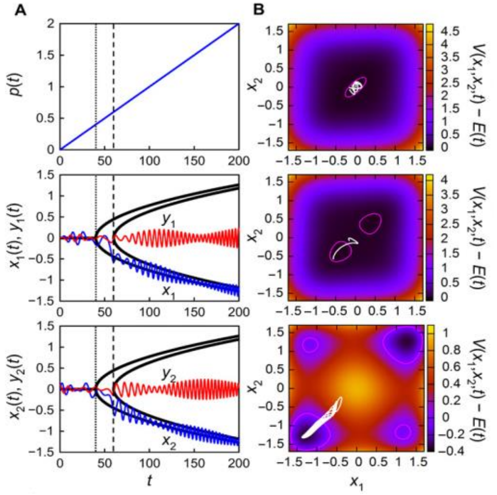
$$H_{SB}(\vec{x},\vec{y},t) = \sum_{i=1}^{N} \frac{\Delta}{2} {y_i}^2 + V(\vec{x},t)$$

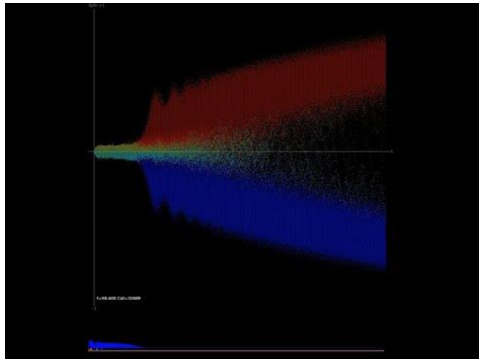
$$V(\vec{x},t) = \sum_{i=1}^{N} \left[\frac{K}{4} {x_i}^4 + \frac{\Delta - p(t)}{2} {x_i}^2 \right] - \frac{\xi_0}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} J_{i,j} x_i x_j$$

$$\frac{\partial x_i}{\partial t} = \frac{\partial H_{SB}}{\partial y_i} = \Delta y_i$$

$$\frac{\partial y_i}{\partial t} = -\frac{\partial H_{SB}}{\partial x_i} = -[K{x_i}^2 - p(t) + \Delta] x_i + \xi_0 \sum_{j=1}^{N} J_{i,j} x_j$$

Equations model the bifurcation (Anil's lecture)

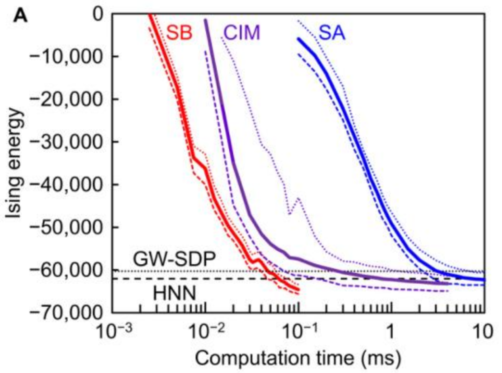




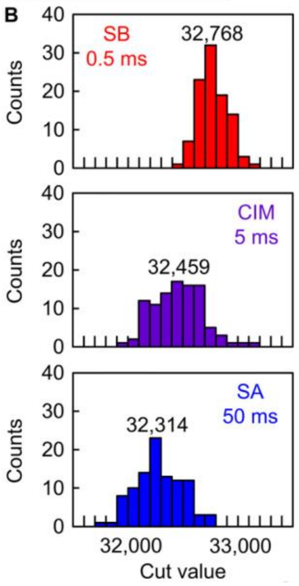
[1] Waidyasooriya, Hasitha, and Masanori Hariyama. "Highly-parallel FPGA accelerator for simulated quantum annealing." IEEE Transactions on Emerging Topics in Computing (2019).
[2] Goto, Hayato, Kosuke Tatsumura, and Alexander R. Dixon. "Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems." Science advances 5.4 (2019): eaav2372.

Simulated Bifurcation Machine

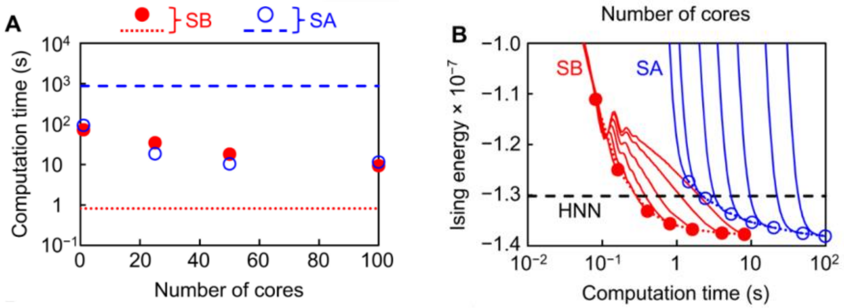
- Authors implemented algorithm in FPGA to solve up to 20,000 nodes fully connected graphs



	Ave. of HNN			GW-SDP		
	Best (ms)	Ave. (ms)	Worst (ms)	Best (ms)	Ave. (ms)	Worst (ms)
SB	0.047	0.061	0.074	0.040	0.047	0.058
CIM	0.155	0.769	N/A	0.071	0.264	1.16
SA	2.64	6.80	N/A	2.10	3.20	7.15



- Authors implemented algorithm in CPU and GPU to solve up to 1’000,000 nodes fully connected graphs



Dots: CPU
Lines: GPU

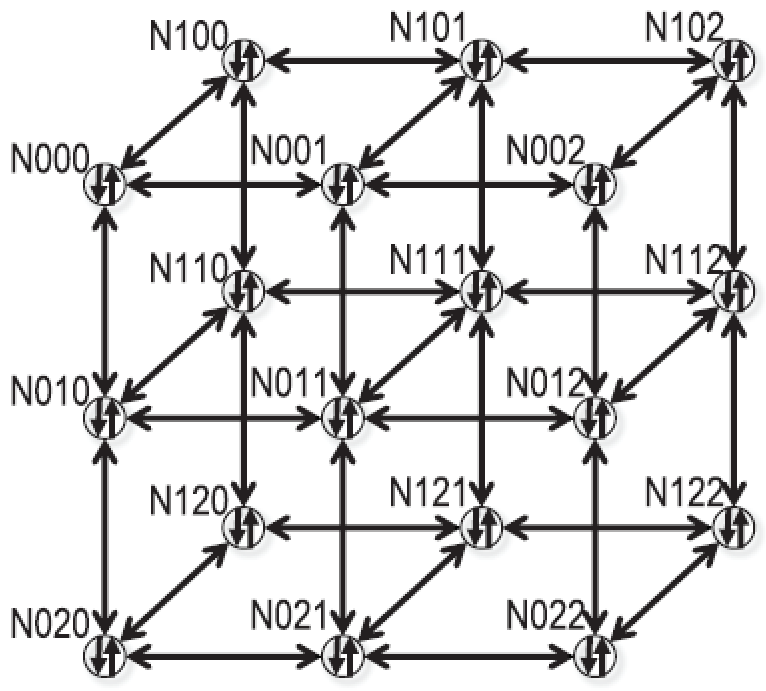
Dots: Cutoff value
Line: 10-run avg.

GPU Version Made available through

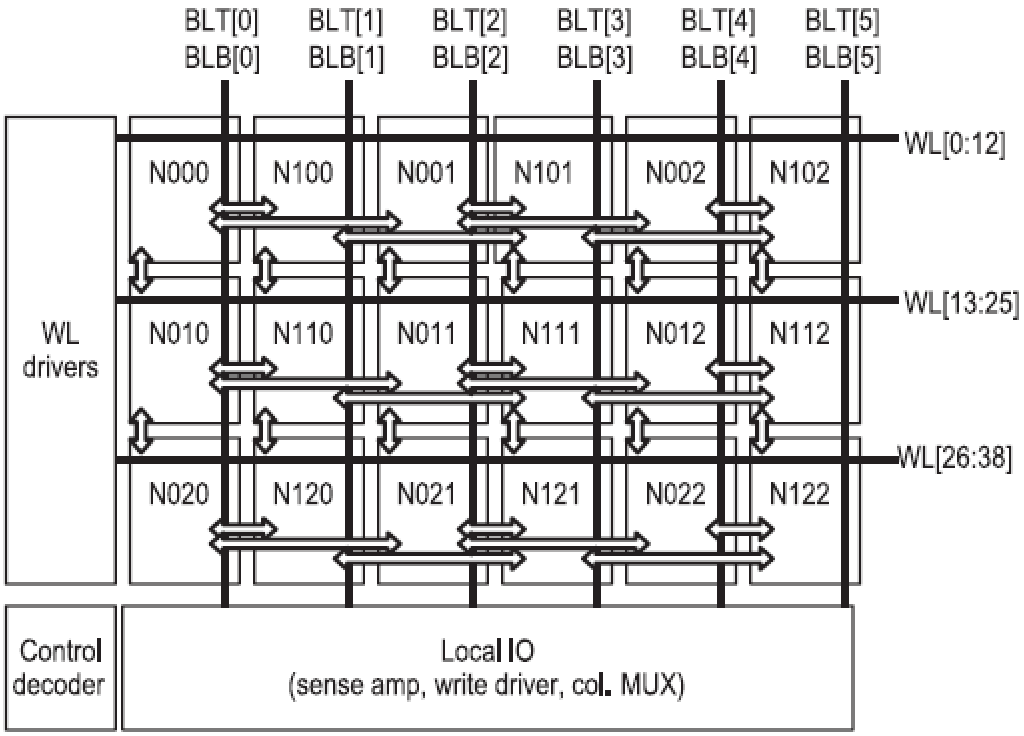
TOSHIBA aws

[1] Goto, Hayato, Kosuke Tatsumura, and Alexander R. Dixon. "Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems." Science advances 5.4 (2019): eaav2372.
[2] http://www.toshiba-sol.co.jp/en/pro/sbm/index.htm

Complementary metal-oxide semiconductors (CMOS)



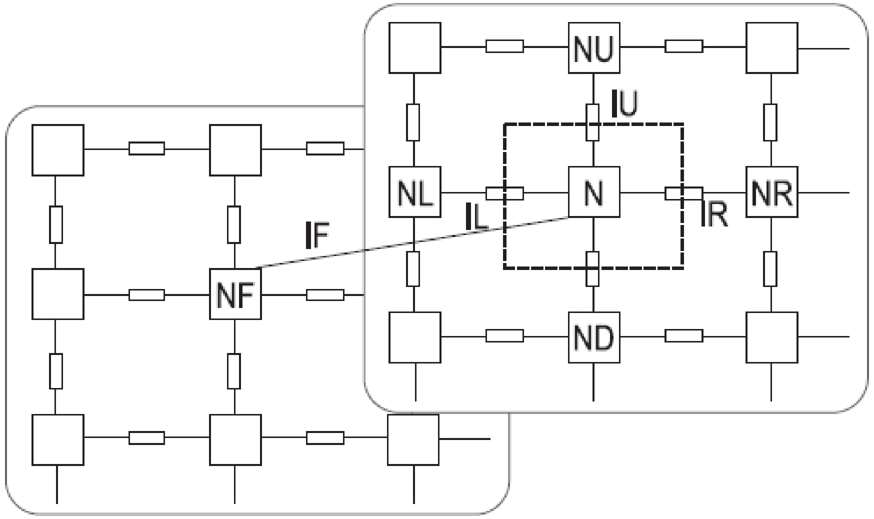
3D Ising Model



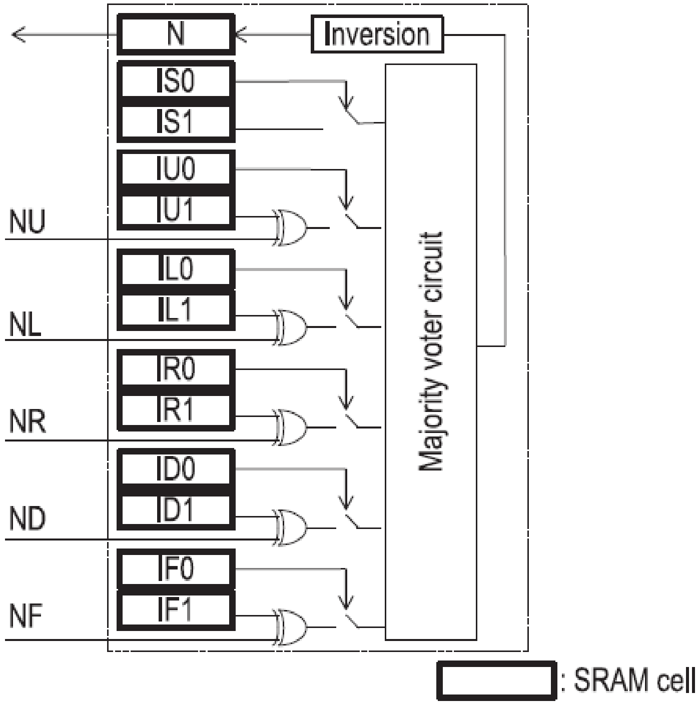
CMOS Static RAM (SRAM) Circuits

[1] Yamaoka, Masanao, et al. "A 20k-spin Ising chip to solve combinatorial optimization problems with CMOS annealing." IEEE Journal of Solid-State Circuits 51.1 (2015): 303-309.

Complementary metal-oxide semiconductors (CMOS)



Each Spin has 5 neighbors (Up, Down, Right, Left, Front)



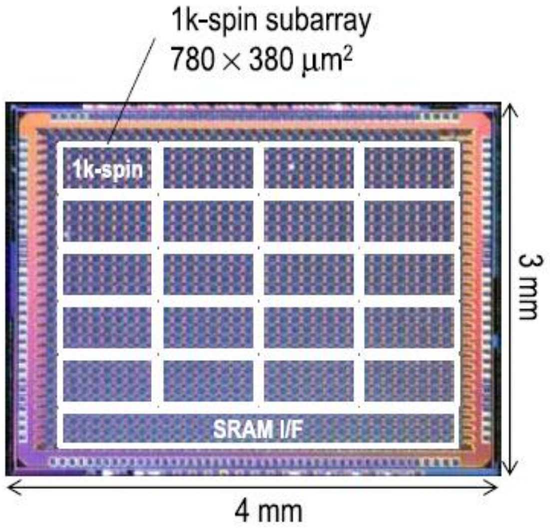
Spin implementation as logic gates

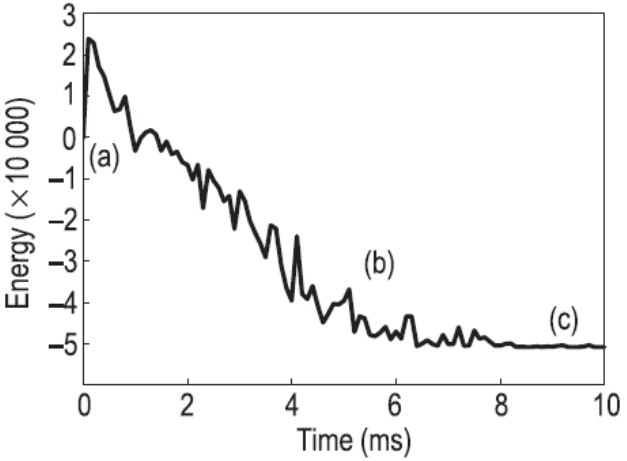
+

Use low voltage to induce random errors in SRAM and jump local minima

[1] Yamaoka, Masanao, et al. "A 20k-spin Ising chip to solve combinatorial optimization problems with CMOS annealing." IEEE Journal of Solid-State Circuits 51.1 (2015): 303-309.

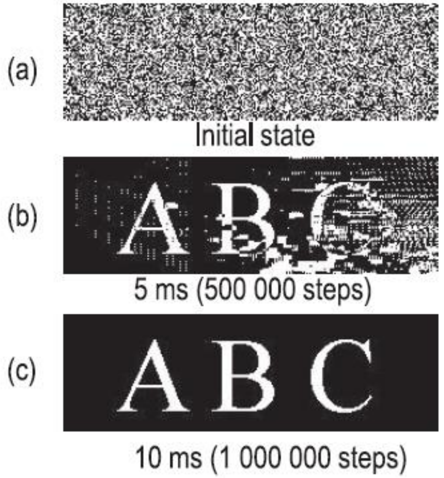
Complementary metal-oxide semiconductors (CMOS)





Actual Chip and Specs

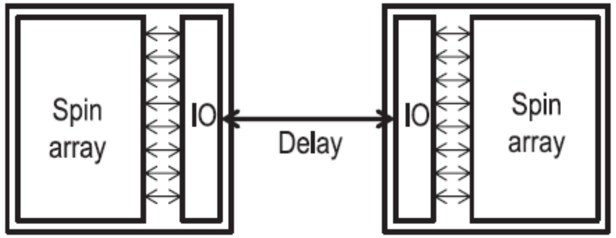
Items	Value
Number of spins	20k (80 × 256)
Process	65 nm
Chip area	$4 \times 3 = 12$ mm^2
Area of spin	$11.27 \times 23.94 = 270$ μm^2
Number of SRAM cells	260k bits Spin value: 1 bit Interaction factor: 2 bit × 5 = 10 bits External magnetic coefficient: 2 bits
Memory IF	100 MHz
Interaction speed	100 MHz
Operating current of core circuits (1.1 V)	Write: 2.0 mA Read: 6.0 mA Interaction: 44.6 mA



[1] Yamaoka, Masanao, et al. "A 20k-spin Ising chip to solve combinatorial optimization problems with CMOS annealing." IEEE Journal of Solid-State Circuits 51.1 (2015): 303-309.

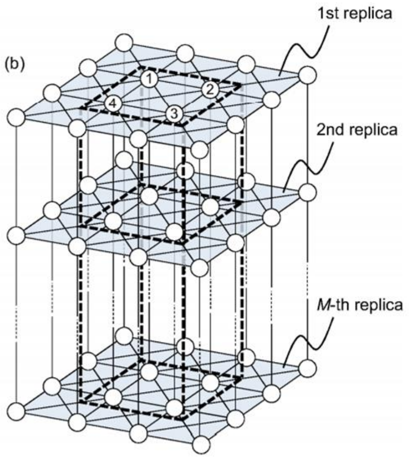
Complementary metal-oxide semiconductors (CMOS)

Easily Parallelizable and manageable

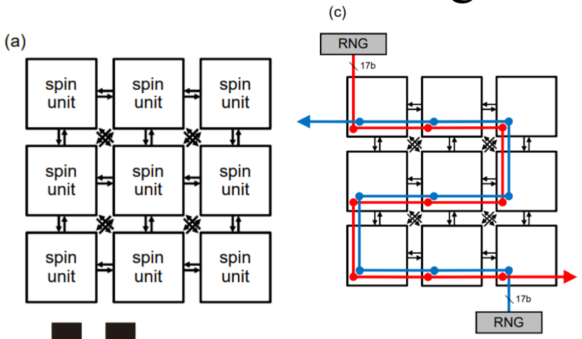


Item	This work
Spin	•SRAM cell (digital bit)
Interaction coefficient & operation	•SRAM cell (digital bit) •Logic circuits (digital bit, asymmetry)
Scalability (one device)	•Easy: CMOS scaling (over 10k spins)
Scalability (multi chip)	•Easy: digital IF can be used
Annealing	•CMOS circuits (digital operation)
Operating condition	•Room temperature (300 K)

Same Idea also implemented in FPGA



King Unit cell useful for both spins and random number generation



Now available through HITACHI

	Hitachi CMOS Annealer [15], [26]	Hitachi CMOS Annealer [16], [26]
Maximum number of spins	61,952	6,400
Type of coupling	King graph	King graph
Number of couplings	0.37 million	0.4 million
Computation	not mentioned	8-bit fixed point
Implementation	ASIC	FPGA

[1] Yamaoka, Masanao, et al. "A 20k-spin Ising chip to solve combinatorial optimization problems with CMOS annealing."
[2] Okuyama, Takuya, Masato Hayashi, and Masanao Yamaoka. "An Ising computer based on simulated quantum annealing by path integral Monte Carlo method."
[3] https://annealing-cloud.com/en/about/cmos-annealing-machine.html

Playing with Hitachi’s CMOS

Let’s go to this interactive interface of the CMOS device from Hitachi
https://annealing-cloud.com/en/play/ising-editor.html

Digital Annealers

CMOS Implementation of Ising solution method
Fully connected 1024 nodes
16-bit precision vs. 4-bit precision D-Wave

"For obtaining exact solutions of small-size problems, the machine called "Digital Annealer" may be the fastest so f

[1] https://arxiv.org/pdf/1806.08815.pdf
[2] https://spectrum.ieee.org/tech-talk/computing/hardware/fujitsus-cmos-digital-annealer-produces-quantum-computer-speeds
[3] Goto, Hayato, Kosuke Tatsumura, and Alexander R. Dixon. "Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems." Science advances 5.4 (2019): eaav2372.

Digital Annealers

Algorithm 1 Simulated Annealing (SA)

for each run **do**
  initialize to random initial state
  for each temperature **do**
    for each MC sweep at this temperature **do**
      for each variable **do**
        propose a flip
        if accepted, update the state and effective fields
      end for
    end for
    update the temperature
  end for
end for

Arbitrary start
Initial fields not required to be calculated

Parallel-trial
Boost acceptance probability

Dynamic off-set escape
Helps surmount short, narrow barriers

Algorithm 2 The Digital Annealer's Algorithm

initial_state $\leftarrow$ an arbitrary state
for each run **do**
  initialize to initial_state
  $E_{\text{offset}} \leftarrow 0$
  for each MC step (iteration) **do**
    if due for temperature update, update the temperature
    for each variable j, in parallel **do**
      propose a flip using $\Delta E_j - E_{\text{offset}}$
      if accepted, record
    end for
    if at least one flip accepted **then**
      choose one flip uniformly at random amongst them
      update the state and effective fields, in parallel
      $E_{\text{offset}} \leftarrow 0$
    else
      $E_{\text{offset}} \leftarrow E_{\text{offset}} +$ offset_increase_rate
    end if
  end for
end for

[1] https://arxiv.org/pdf/1806.08815.pdf
[2] https://spectrum.ieee.org/tech-talk/computing/hardware/fujitsus-cmos-digital-annealer-produces-quantum-computer-speeds

Parallel Tempering

Algorithm 1 Simulated Annealing (SA)

```
for each run do
    initialize to random initial state
    for each temperature do
        for each MC sweep at this temperature do
            for each variable do
                propose a flip
                if accepted, update the state and effective fields
            end for
        end for
        update the temperature
    end for
end for
```

Algorithm 3 Parallel Tempering with Isoenergetic Cluster Moves (PT+ICM)

```
initialize all replicas with random initial states
for each MC sweep do
    for each replica, for each variable do
        propose a flip
        if accepted, update the state and effective fields
    end for
    for each pair of sequential replicas do
        propose a replica exchange
        if accepted, swap the temperatures between the replicas
    end for
    perform ICM update, swapping the states of a cluster of variables that
have opposite states in the two replicas; update the states and the effective
fields for both replicas
end for
```

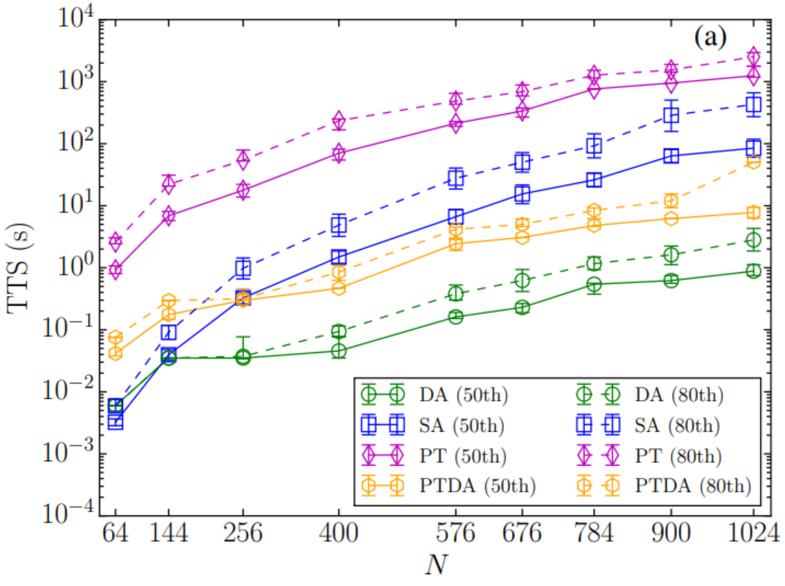
- Instead of having a single state you have several replicas
- Then the flips can be done among replicas
- It can be implemented in the Digital Annealer

- Additionally: There can be cluster updates (flip more than one spin if they are “connected”)
    - Similar to Anil’s intuition on the Swendsen-Wang Algorithm

https://arxiv.org/pdf/1806.08815.pdf
https://spectrum.ieee.org/tech-talk/computing/hardware/fujitsus-cmos-digital-annealer-produces-quantum-computer-speeds

Digital Annealing v Simulated Annealing v Parallel Tempering

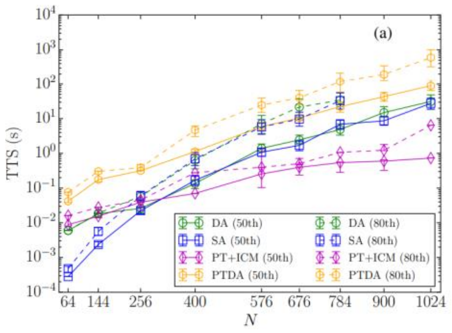
Fully connected instances



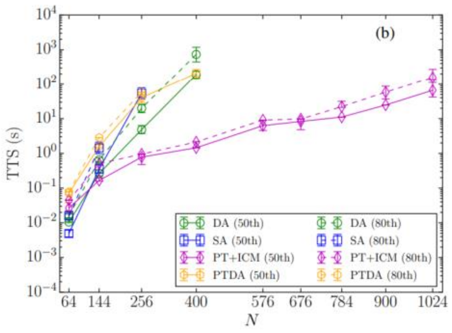
Digital Annealer Wins

Sparse instances

2D-Bimodal



2D-Sparse



- DA Digital Annealer
- SA Simulated Annealing
- PT(+ICM) Parallel Tempering (+Isoenergetic Cluster Moves)
- PTDA Parallel Tempering Digital Annealer

Parallel Tempering Wins

[1] https://arxiv.org/pdf/1806.08815.pdf
[2] S. V. Isakov, I. N. Zintchenko, T. F. Rønnow, and M. Troyer, Optimized simulated annealing for Ising spin glasses, Comput. Phys. Commun. 192, 265 (2015)

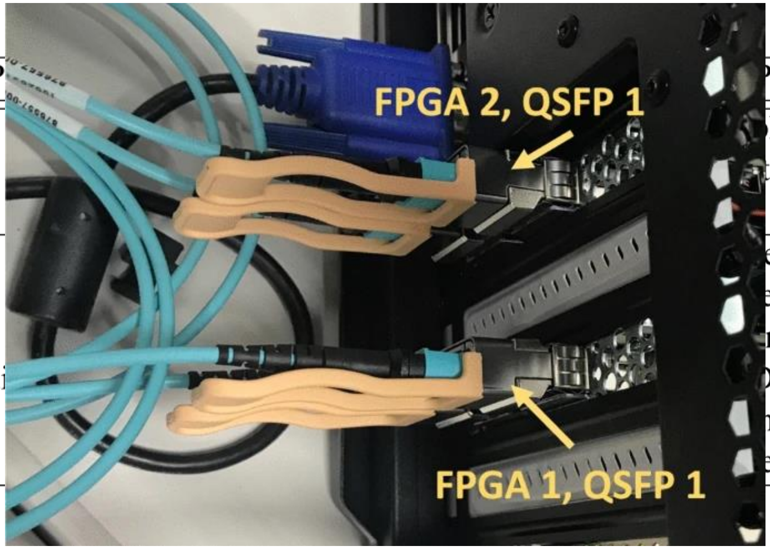
Digital Annealers

CH TCO	Quantum Computing Challenge Series - Max Cut Marathon Match Ended Apr 04 · Marathon Match	$11,500 Purse
CH TCO	Quantum Computing Learning Challenge #3 - Max Cut Ended Aug 04 · Python · Data Science · Other	$250 Purse
CH TCO	Quantum Computing Learning Challenge 2 - Scheduling Ended Feb 28 · Python · Data Science · Other	$250 Purse
CH TCO	Quantum Computing Learning Challenge #1 - Solve Sudoku Instantly Ended Feb 14 · Algorithm · Python · Data Science · +1	$250 Purse

[1] https://arxiv.org/pdf/1806.08815.pdf
[2] https://spectrum.ieee.org/tech-talk/computing/hardware/fujitsus-cmos-digital-annealer-produces-quantum-computer-speeds
[3] https://tc3-japan.github.io/DA_tutorial/index.html

Digital Annealer v Application-specific integrated circuit v FPGA v GPU

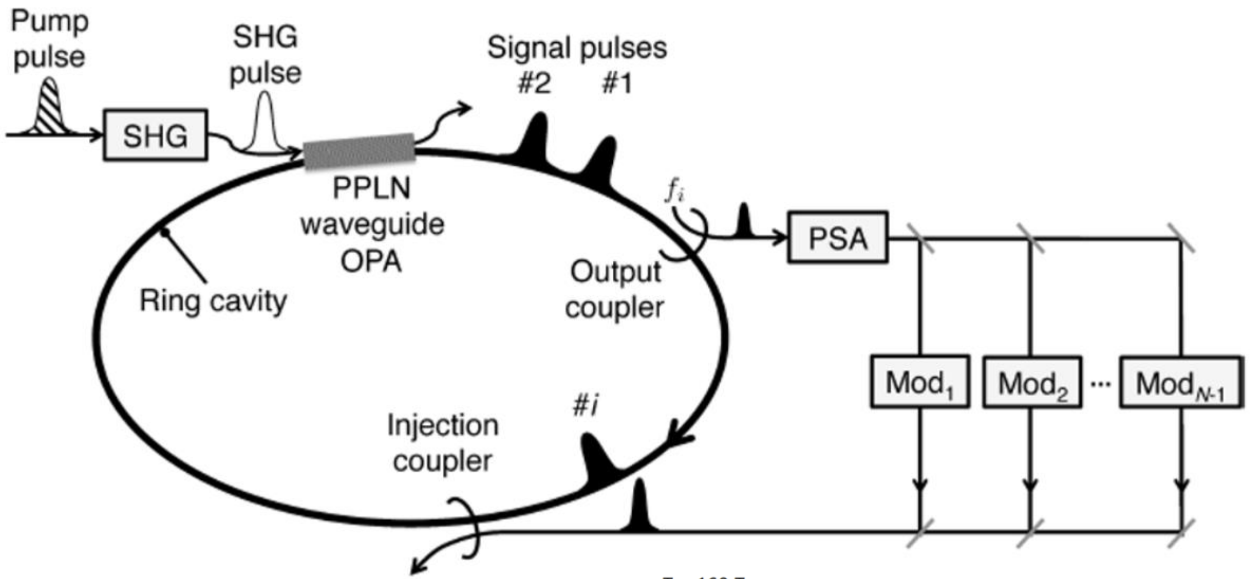
	Fujitsu Digital Annealer [25]	Hitachi CMOS Annealer [15], [26]	Hitachi CMOS Annealer [16], [26]	This work
Maximum number of spins	8192	61,952	6,400	32,768
Type of coupling	Total coupling	King graph	King graph	Total Coupling
Number of couplings	67 million	0.37 million	0.4 million	1 billion
Computation	64-bit fixed-point	not mentioned	8-bit fixed point	32-bit floating-point
Implementation	ASIC	ASIC	FPGA	2-FPGA connected via fiber

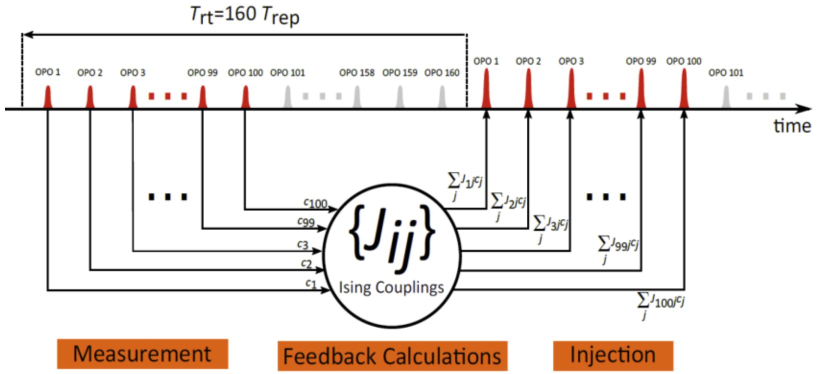


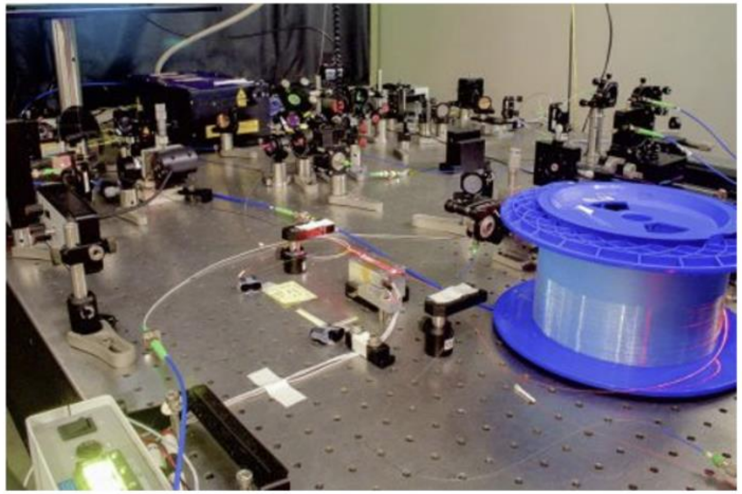
Category	FP	U accelerator
Speed-up		lementation
Accuracy		ation
Problem size		
Power consumption		ely large
Power-efficiency		ely small
Availability		m PCs to supercomputers
Programmability	Requi	DA, OpenCL, OpenAcc, etc.
Compilation time		n a minute
Design time		ely small

[1] Waidyasooriya, Hasitha Muthumala, and Masanori Hariyama. "A GPU-Based Quantum Annealing Simulator for Fully-Connected Ising Models Utilizing Spatial and Temporal Parallelism."
[2] Waidyasooriya, H.M., Hariyama, M., Miyama, M.J. et al. OpenCL-based design of an FPGA accelerator for quantum annealing simulation.
[3] Waidyasooriya, Hasitha Muthumala, and Masanori Hariyama. “Highly-Parallel FPGA Accelerator for Simulated Quantum Annealing”

Coherent Ising Machines







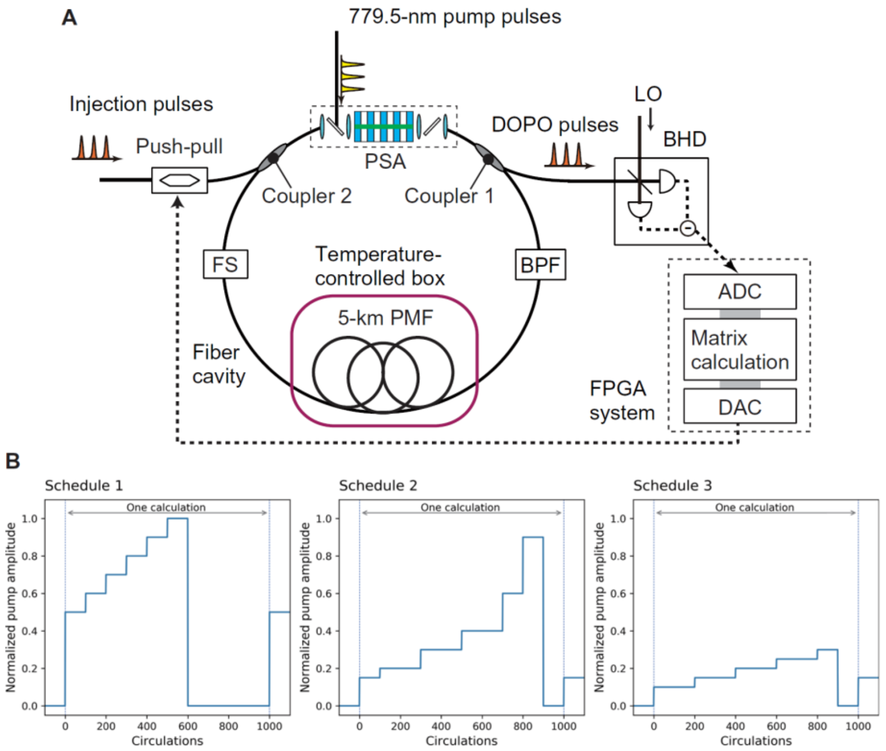
NTT
NIPPON TELEGRAPH AND TELEPHONE CORPORATION

Electrical & Computer ENGINEERING

TEPPER

USRA Universities Space Research Association

NEWS: 100,000 Spins



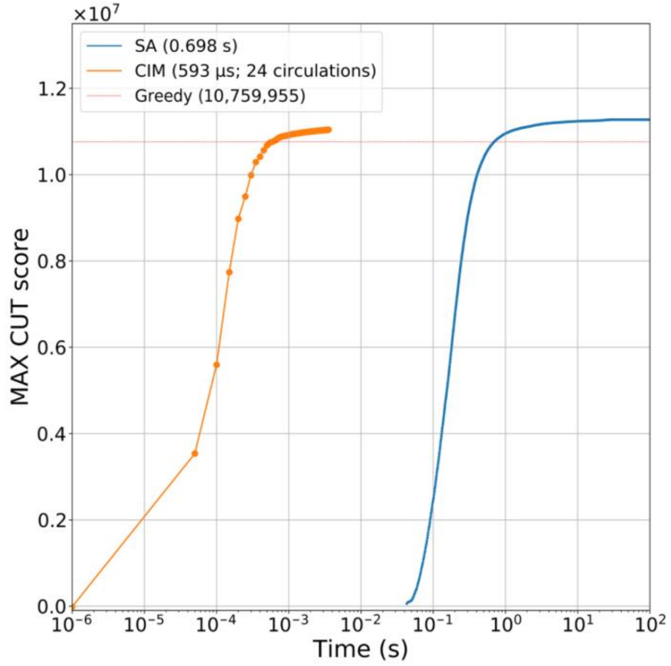


Fig. 3. MAX CUT score as a function of computation time obtained with the CIM (orange line) and SA (blue line). The data points exhibit the scores evaluated at the intermediate steps in the CIM and SA computation. The dotted line denotes the score obtained with SG (10,759,955).

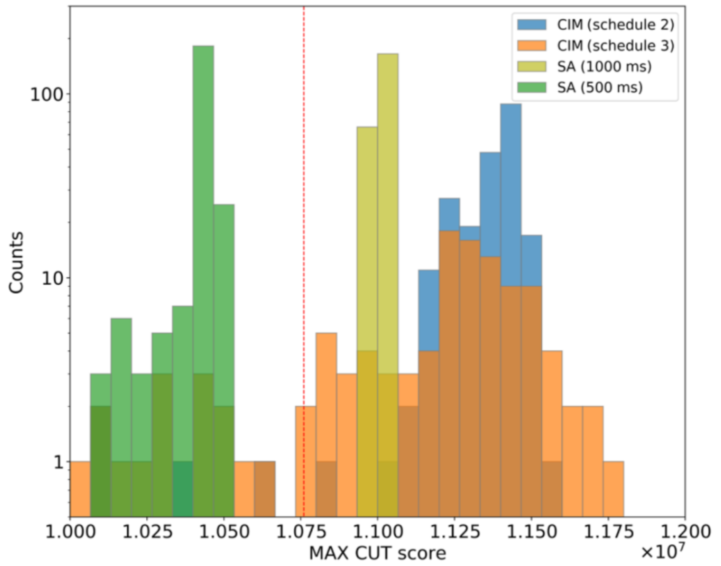


Fig. 6. Histograms of MAX CUT score with CIM and SA. The vertical dashed line shows the SG score (10,759,955).

SCIENCE ADVANCES | RESEARCH ARTICLE

COMPUTER SCIENCE

100,000-spin coherent Ising machine

Toshimori Honjo[1]*, Tomohiro Sonobe[2], Kensuke Inaba[1], Takahiro Inagaki[1], Takuya Ikuta[1], Yasuhiro Yamada[1], Takushi Kazama[3], Koji Enbutsu[3], Takeshi Umeki[3], Ryoichi Kasahara[3], Ken-ichi Kawarabayashi[2], Hiroki Takesue[1]*

Alternatives available

	Fixstars Optigan	D-Wave 2000Q	Hitachi CMOS Annealing	Fujitsu Digital Annealer	Toshiba SBM
Calculation method	GPU	Quantum annealing	Digital circuit	Digital circuit	GPU
Maximum number of bits	Over 100,000	2,048 (16x16x8)	61,952 (352x176)	1,024 / 8,192	10,000
Coefficient parameter	Digital (32 / 64bit)	Analog (about 5bit)	Digital (3bit)	Digital (16/64 bit)	Digital (32bit)
Combined graph	Fully combined	Chimera graph	King Graph	Fully combined	Fully combined
Total number of combined conversion bits	65,536	64	176	1,024 / 8,192	1,000
API endpoint	Fixstars	D-Wave Cloud	Annealing Cloud Web	DA Cloud	AWS

Playing with several platforms

You can check one of the integrated software stack for several of these platforms developed in Japan at https://amplify.fixstars.com/en/

Translated version (but you cannot run it) https://colab.research.google.com/github/bernalde/QuIPML/blob/master/notebooks/Notebook%208%20-%20Amplify%20Tutorials.ipynb

Playing with several platforms

Explore quantum cloud solutions available on Azure Quantum

Azure Quantum assembles and curates some of the most compelling and diverse quantum resources available today from industry leaders—including optimization and quantum hardware solutions—for developers and customers across all industries.

Azure Quantum enables you to learn, build, and deploy impactful solutions at scale, helping you harness quantum computing and benefit from the latest innovations.

QUANTUM COMPUTING

Honeywell

Honeywell quantum solutions

Trapped-ion system with high-fidelity, fully connected qubits, and the ability to perform mid-circuit measurement.

QUANTUM COMPUTING

IONQ

IONQ Trapped-ion quantum computer

Dynamically reconfigurable system for up to 11 fully connected qubits that lets you run a two-qubit gate between any pair.

QUANTUM COMPUTING

qci

Quantum Circuits, Inc

Quantum Circuits, Inc.

Fast and high-fidelity system with powerful real-time feedback to enable error correction.

OPTIMIZATION

1QBit

1Qloud

Connecting intractable industry problems to innovative solutions.

OPTIMIZATION

Microsoft

Microsoft QIO

Ground-breaking optimization algorithms inspired by decades of quantum research.

OPTIMIZATION

TOSHIBA

Toshiba SBM

Toshiba Simulated Bifurcation Machine is a GPU-powered ISING machine that solves large-scale combinatorial optimization problems at high speed.

https://azure.microsoft.com/en-us/services/quantum/#features