PAPER Special Section on Parallel and Distributed Computing and Networking

FPGA-Based Annealing Processor with Time-Division Multiplexing

Kasho YAMAMOTO[†a)], ***Nonmember***, **Masayuki IKEBE**[†], **Tetsuya ASAI**[†], **Masato MOTOMURA**[††], ***and*** **Shinya TAKAMAEDA-YAMAZAKI**[†††,††††], ***Members***

SUMMARY An annealing processor based on the Ising model is a remarkable candidate for combinatorial optimization problems and it is superior to general von Neumann computers. CMOS-based implementations of the annealing processor are efficient and feasible based on current semiconductor technology. However, critical problems with annealing processors remain. There are few simulated spins and inflexibility in terms of implementable graph topology due to hardware constraints. A prior approach to overcoming these problems is to emulate a complicated graph on a simple and high-density spin array with so-called minor embedding, a spin duplication method based on graph theory. When a complicated graph is embedded on such hardware, numerous spins are consumed to represent high-degree spins by combining multiple low-degree spins. In addition to the number of spins, the quality of solutions decreases as a result of dummy strong connections between the duplicated spins. Thus, the approach cannot handle large-scale practical problems. This paper proposes a flexible and scalable hardware architecture with time-division multiplexing for massive spins and high-degree topologies. A target graph is separated and mapped onto multiple virtual planes, and each plane is subject to interleaved simulation with time-division processing. Therefore, the behavior of high-degree spins is efficiently emulated over time, so that no dummy strong connections are required, and the solution quality is accordingly improved. We implemented a prototype hardware design for FPGAs, and we evaluated the proposed method in a software-based annealing processor simulator. The results indicate that the method increased the spins that can be deployed. In addition, our time-division multiplexing architecture improved the solution quality and convergence time with reasonable resource consumption.

key words: *ising model, annealing processor, simulated annealing*

1. Introduction

Combinatorial optimization is a fundamental and important problem with various applications. The goals are generally to find the best combination from numerous candidates within limited computing time and resources. The number of available candidate combinations generally grows exponentially according to the number of problem factors. Thus, finding an exact solution is difficult on standard Neumann computers within a feasible amount of time. Even when approximated solutions instead of exact ones are permitted, they tend to consume ample computing time when high-quality solutions are needed.

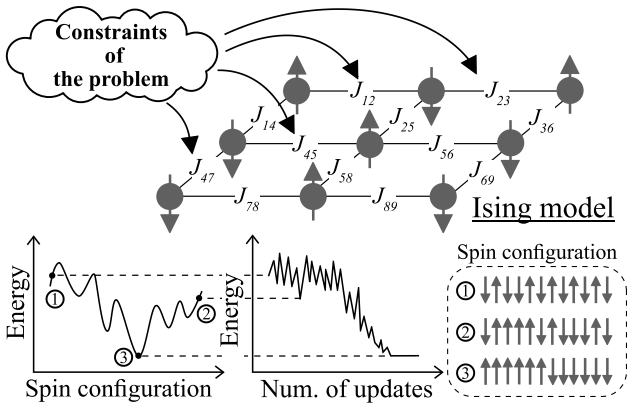


Fig. 1 Ising model

In order to overcome such bottlenecks in standard computers for combinatorial optimization, alternative computing mechanisms are aggressively pursued. One such computing paradigm is so-called natural computing, inspired by systems in nature. Unlike conventional computers, natural computing is based on the mechanism that the computing state eventually reaches the state of lowest energy, as shown in Fig. 1. The annealing processor, based on the Ising model, is one approach to natural computing. The annealing processor exploits a combinatorial optimization method called simulated annealing.

Among annealing processors, the CMOS annealing processor (CMOS-AP) was proposed by Yamaoka et al. [1] as a powerful combinatorial optimization method. The advantage of CMOS annealing is its high parallelism. It can update many spin states—the elements constituting the system of the Ising model—at the same time. Thus, the computing state converges faster than other CMOS approaches.

A disadvantage of the conventional CMOS-AP is that there are a limited number of spin counts that can be deployed due to hardware resource constraints. Our previous memory-based architecture [2] solves this problem by time-division multiplexing. The architecture stored an entire large graph of problem representation in BRAM and made it possible to solve the problem of many spins by dividing the processing in the time direction.

Because the spin-to-spin connections in hardware are

Manuscript received January 7, 2019.
Manuscript revised May 20, 2019.
Manuscript publicized September 20, 2019.

[†]The authors are with Hokkaido University, Sapporo-shi, 060–0814 Japan.
[††]The author is with Tokyo Institute of Technology, Yokohama-shi, 226–8502 Japan.
[†††]The author is with The University of Tokyo, Tokyo, 113–8656 Japan.
[††††]The author is with JST PRESTO, Kawaguchi-shi, 332–0012 Japan.
a) E-mail: yamamoto.kasho.sc@ist.hokudai.ac.jp

DOI: 10.1587/transinf.2019PAP0002

usually sparse in various problems, if the problem graph into an executable graph for the hardware is more complicated than spin-to-spin connectivity as a hardware circuit, it is necessary to convert the problem graph into an executable graph for the hardware by duplicating the spins in the graph of the problem to match the hardware. Our previous work suggested that this deformation leads to a decrease in solution accuracy.

In this paper, we propose the architecture that solves the above problems based on our previous research. The contributions of this paper are as follows:

1. We present a time-division multiplexing architecture for an annealing processor on an FPGA that can solve both insufficient spin counts and the problem of sparse connections in the hardware. It enables a flexible expansion of the spin count and connections with BRAMs and the newly introduced techniques: Continuous Processing and Reverse Order Processing.
2. We evaluated the resource consumption of our proposal using commercial FPGA synthesis tools and the accuracy and convergence speed of our architecture, especially for more complex problems than hardware graphs by using our software simulator.

2. Related Work

Combinatorial optimization methods for conventional computers have been studied widely [3]. However, such problems are known as NP-hard [4]. Consistent with Moore's Law, improvements to the performance of conventional computers is slow with respect to the increasing amount of data [5]. Therefore, solving NP-hard problems with conventional computers requires considerable time and energy.

A computing architecture called the annealing machine has been proposed to map combinatorial optimization problems and find the ground state of the Ising model, the statistical model representing the behavior of the spins of magnetic material. The energy function of the Ising model is represented as follows:

$$H = -\sum_{<i,j>} J_{ij}\sigma_i\sigma_j - \sum_i h_i\sigma_i \quad (1)$$

where σ_i denotes an individual spin state, J_{ij} denotes the interaction coefficient that represents the strength of the interactions between different pairs of spin states and h_i denotes the external magnetic field coefficient. Controlled objects , "vertices" in the max-cut problem and "cities and orders" in the traveling salesman problem, and the cost of the problem is expressed by the spin states and interactions, respectively.

D-wave Systems proposed D-wave [6], which exploits quantum computing technology. Unlike a general-purpose quantum computer, D-wave is a quantum computer specialized for combinatorial optimization problem by expressing the Ising model using superconducting elements. D-wave operates based on an optimization technique called quantum annealing. In quantum annealing, weakening the quantum effect maximizes the probability of observing a qubit state that minimizes the total energy of the system. However, a large-sized cooler is needed to cool the superconducting element to an extremely low temperature. In addition, there is a lack of spin numbers to express real problems.

In order to solve such problems, methods using optical parametric amplification [7] and methods using CMOS technology have been proposed. CMOS-AP is an annealing processor based on a Monte Carlo simulation of the classical system (simulated annealing) using a CMOS technique without the quantum effect. It performs simulations of the Ising model in a classical system and searches for the ground state by decreasing the temperature of the system, not the quantum effect.

In recent years, FPGA-based annealing machines have attracted attention to facilitate the development of such systems instead of custom LSIs. There are indeed proposals of FPGA-based Monte Carlo simulations of the Ising model [8]. However, these are simulators designed for physics research, rather than combinatorial optimization problems. With the FPGA-based AP approach, hardware topologies, embedding [9], and random number generators [10] are mainly studied. In terms of the hardware topology, two approaches are mainly studied: the all-to-all type [11], [12], in which all spins are connected to each other; and the nearest-neighbor type [13], in which only adjacent spins are coupled.

With the former, since the hardware allows for all spin coupling (high problem-expression capability), problems converted to the Ising model are easily embedded in the hardware. However, only a few spins can be deployed because of the many interaction coefficients. In addition, due to the restrictions of Glauber dynamics, neighboring spins cannot be updated simultaneously. Therefore, only one spin can be updated at the same time in all coupled spin networks. Furthermore, since it is necessary to share a spin state change with all spins, its scalability is low.

With the latter approach, even under the constraints of Glauber dynamics, non-adjacent spins can be updated at the same time. Further, since the change in the spin state after updating is only the adjacent spin, parallelism and scalability are very high. However, when the combinatorial optimization problem converted to the Ising model does not match the topology of the hardware, additional conversions are required for the embedding process.

This "additional conversions" cause the degradation of the solution quality. As research on solution quality, there is ICAPT [14] that implements parallel tempering approximation processing with high affinity to hardware. On the other hand, this research focuses on maintaining the quality of the solution by eliminating the loss of accuracy due to "additional conversion" by improving problem expression capability. In paper [15], the rectangle packing problem is solved by the all-to-all type annealing machine and the nearest-neighbor type annealing machine. In the evaluation in that paper, the nearest-neighbor type annealing machine

has degraded solution quality compared to the all-to-all type annealing machine.

In this paper, we propose a new approach to overcome the disadvantages to both the all-to-all approach and the nearest-neighbor approach while maintaining high parallelism, scalability, and problem expression capability.

Based on the previous LUT based architecture, we proposed a new BRAM-based architecture to deploy more spins. In addition, by introducing time-division multiplexing, our architecture can flexibly change the hardware topology from the nearest-neighbor to the all-to-all type.

In what follows, details regarding the conventional architecture and problems resulting from mismatches between the converted problem and hardware topology are described.

3. Cmos Annealing Processor

3.1 Conventional Architecture

$$Interaction = \sum_{j=1}^{L-1} J_{ij}\sigma_j + h_i \tag{2}$$

CMOS annealing is a technique that simulates the Ising model with a CMOS circuit. In CMOS-AP [16], the spin states are held in binary $(+1, -1)$ and stored in one bit in the circuit. Both interaction coefficients and external magnetic field coefficients are held in ternary $(+1, 0, -1)$. Each coefficient is stored in two bits. Using these values, the interaction effect between neighboring spins is simulated on the digital circuit.

Figure 2 shows the operator unit for updating the states of spins. In this circuit, the operation shown in Eq. (2) is performed and the state of spin "i" is updated based on the sign of the interaction.

"This unit receives the states of adjacent spins (σ_i) and coefficients (J and H) for spin updates, where $H_k[0]$ and $J_k[0]$ represent the signs, $H_k[1]$ and $J_k[1]$ represent the absolute values and these interactions occur between the spin "i" and its k-th adjacent spin." With these values, calculations are performed in each spin-to-spin connection through XNOR gates (rather than the multiplier), as shown in Fig. 2 (a1), because the spin state and interaction are binary and ternary, respectively. Further, the number of spin-to-spin connections depends on the number of XNOR gates in this architecture.

Then, the results are tabulated by the majority voter circuit shown in Fig. 2 (a2). In this way, the state of the next spin is determined. States of multiple spins can be updated at the same time, provided that they are not connected. Thus, if the number of spins included in the Ising model increases, the number of updated spins processed at the same time increases. In short, the problem size (i.e., the number of spins) has little influence on the update time.

Although the energy decreases according to the interaction operation described above, it is possible that the model becomes trapped in a local minimum that is not the overall minimum. To escape from the local minimum, the states of the spins are randomly destroyed with a thermal fluctuation. This operation is performed by the inverters shown in Fig. 2 (b) with two kinds of random pulses. For details regarding these random pulses, see [13]. This enables the states to escape from the local minimum by randomly transitioning the spin state, making it possible to find a state close to the ground state.

The Fig. 3 shows the spin unit proposed in the previous study. Since the spin unit holds four fully-coupled spins and the adjacent spins cannot be updated at the same time, each spin unit use one operator by switching inputs to reduce resources. A 2-bit counter decides the spins to be updated and its value is used as the address of spin memory and coefficient memory and control signal of selectors. After that, the update spin through the "operator" is written back to the location indicated by the counter.

The optimal network of the spin unit for hardware de-

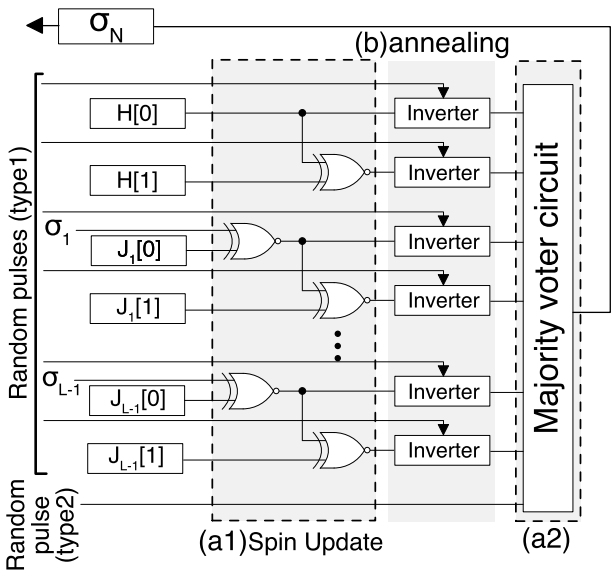


Fig. 2 Operator unit

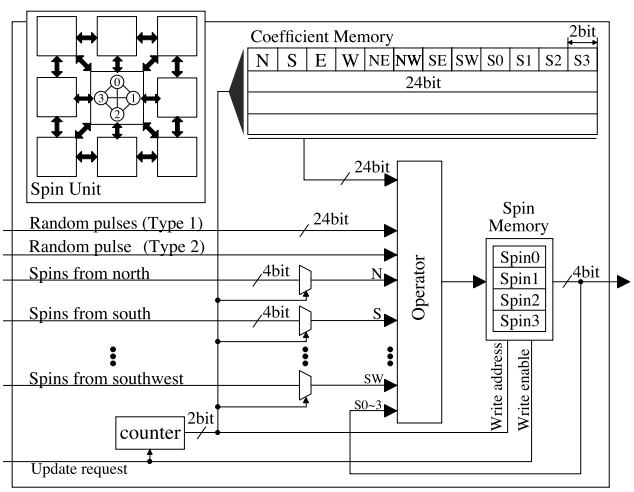


Fig. 3 Spin unit

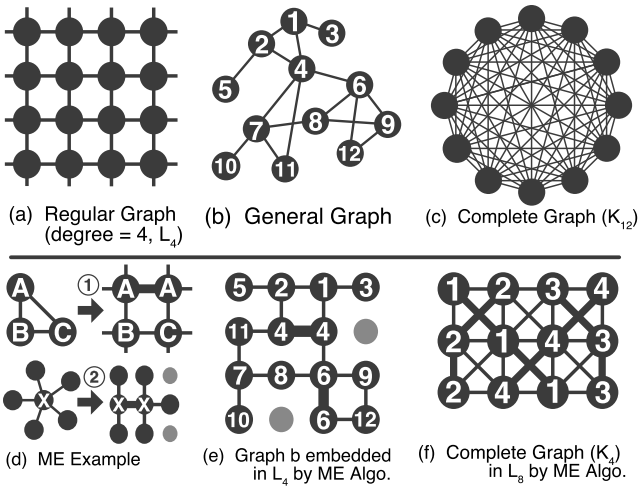


Fig. 4 Minor embedding

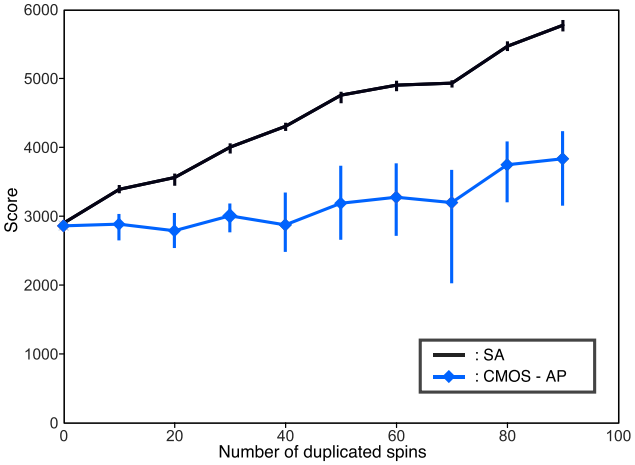


Fig. 5 Decline in accuracy due to duplicated spins

pends on problem. There are some structures for well-known optimization problems such as a three-dimensional mesh structure, a chimera topology, and so on. In this paper, we utilize the king's graph and chimera topology. In the king's graph, spins are placed on a square lattice and spin-to-spin connections are formed on the edge of the square and its diagonal. Therefore, each spin has eight connections. In the chimera topology, four complete spins are contained within one spin unit. Each internal spin is coupled to the corresponding spin in the same position of the adjacent spin units. In short, one spin is coupled with the other three spins in the same spin unit and eight spins at the same position of the adjacent spin units as shown in the upper left of the Fig. 3. Since adjacent spins cannot be updated at the same time, as explained above, all the spin states are updated over four clocks in both topologies.

3.2 Minor Embedding

Minor Embedding (ME) [17] is an algorithm for embedding arbitrary graphs in hardware. In the conventional architecture, there are graphs that cannot be directly embedded in hardware due to hardware and space constraints and because the hardware topology is fixed. In such cases, it is necessary to transform the problem graph so that it can be embedded in the hardware. ME duplicates spins with reverse-minor processing to virtually increase the degree of hardware while maintaining the connection relationship.

We will explain ME for CMOS-AP [18] using the three graphs shown in Fig. 4. The lattice graph in (a) and the complete graph in (c) are denoted L_n and K_m, respectively, where n is the degree of nodes and m is the number of nodes. If it is possible to solve the problem expressed in a complete graph K_m, the Ising machine can solve any kind of problems consisting of m nodes. However, not all problems are expressible as complete graphs. Further, some edges are deleted beforehand as unnecessary. Therefore, we mainly focus on graphs similar to graph (b)—i.e., graphs that are slightly more complicated than lattice graphs.

Given that the spin unit array on the hardware is configured as shown in Fig. 4 (a), subgraphs of L_4 can be embedded without ME. If the problem graph does not satisfy the space and hardware constraints, such graphs are embedded as (d1) and (d2), respectively. When graph (b) is embedded in L_4, vertices 4 and 6 are duplicated for the above reasons. Then, graph (b) is embedded as shown in (e).

Complete graphs K_m can be embedded in a lattice graph L_8 whose size is $m \times (m-1)$. The number assigned to the vertex of row y and column x is denoted by $\#(y, x)$. Vertices of the first row are assigned numbers as follows:

$$\#(1, x) = x \tag{3}$$

In the second and subsequent lines, numbers are allocated as follows:

$$\#(y, x) = \begin{cases} \#(y-1, max(1, x-1)) & (x+y \; is \; even) \\ \#(y-1, min(n, x+1)) & (x+y \; is \; odd) \end{cases} \tag{4}$$

In this way, graph (f) is obtained by converting graph (b) to match king's graph(L_8) which is a kind of graph (a) with ME. When embedding an arbitrary graph, duplicated spins are removed from the converted complete graph so that it has the minimum necessary spins.

3.3 Difficulties with Minor Embedding

In the previous section, we explained how arbitrary graphs can be embedded by ME. However, a new interaction (connection) is required between duplicated spins so that the copying spins stay in the same state. If there is no self-loop, the new interaction between spin i and the duplicated spin is set as follows:

$$new \; connection \; i = \sum |J_{ij}| \tag{5}$$

The newly added interaction is set such that it is stronger than any interaction originally occurring with spin i. Therefore, the replicated spin updates its state based on

the only state of another replicated spin. This means that updating the duplicated spin state is not performed without a random number (heat).

Figure 5 shows the influence on the accuracy of the solution due to the number of duplicated spins in the problem. This graph shows the results from solving the max-cut problem with both simulated annealing and CMOS-AP with ME. When solving a problem that includes a replicated spin, the accuracy of the solution decreases and the solution becomes more dispersed with CMOS-AP compared to simulated annealing.

3.4 Motivation

Based on the results of this preliminary evaluation, we discovered a problem: the interaction between the replicated spins caused by ME deteriorates the convergence speed and the accuracy of the solution. We address this problem using time-division multiplexing.

4. Time-Division Multiplexing Architecture

4.1 Processing in Conventional Architecture for Duplicate Spins

In this section, we explain our solution to the decline in performance from interactions between duplicated spins. As shown in Fig. 6 (a), when there is a mismatch in the problem graph and the topology of the hardware (L_4), the graphs converted by ME are obtained. Figures 6 (a-1) and (a-2) show the process of updating the duplicated spin in the conventional architecture [16] and the time-division multiplexing architecture. When the values of the interaction occurring between spin 4 and its adjacent spins are all 1, an interaction whose value is 5 occurs between duplicated spins, according to Eq. (5).

In the conventional architecture (a-1), the converted graph is allocated to the hardware such that the spin unit and problem spins correspond one-by-one. Then, the next state (*spin 4'*) is determined by calculating *spin 1*, *spin 6*, and another duplicated *spin 4* without *spin 2*, *spin 8*, and *spin 11*. Therefore, the state of the duplicated spin is determined by the state of the other duplicated spin since the newly added interaction is stronger than the interaction of the originally connected spins. This also holds for the update of the other duplicated spin.

4.2 Continuous Processing

Here, we propose a method to solve the above problem with a time-division multiplexing architecture. As noted above, the conventional architecture is a fully expandable approach that requires spin units and spins to correspond in a one-to-one manner. On the other hand, the data supplied from

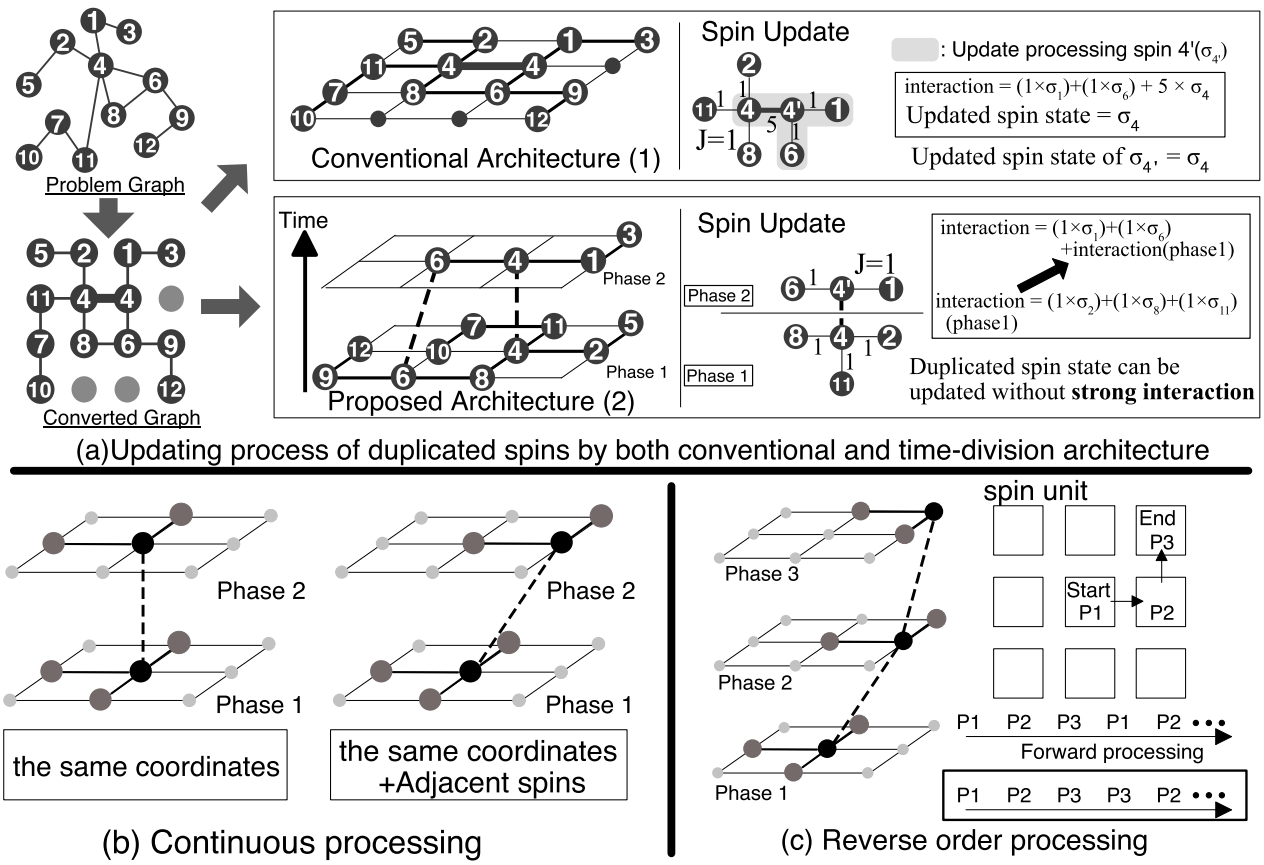


Fig. 6 Time-division multiplexing

the memory to the spin unit is switched with time-division multiplexing. This allows one spin unit to process multiple spins.

Our previous architecture [2] used this property to process large-scale graphs in the spatial direction. Unfortunately, according to Eq. (2), it is difficult to increase the maximum number of degree (L-1) with this approach. It is because increasing the maximum number of degree causes hardware complexity. Therefore, ME is necessary. To solve this problem, we extend the idea of ME for temporal direction, in addition to the spatial direction. In short, different graphs are allocated in each "time" in the same hardware. We refer to this step as the phase, as shown in Fig. 6 (a2).

In the proposed architecture, duplicated spins are not in the same phase (conventional approach), but rather in a different phase. We call this "Continuous Processing." In continuous processing, duplicated spins are processed by the same spin unit over multiple phases. The hardware holds the calculation result only when processing duplicate spins and adds it to the calculation result of the next phase. This virtually increases the maximum number of degree without the strong interaction caused by ME as in the following Eq. (6).

$$Interaction = \sum^{phase} (\sum_{j \in phase\, N}^{L-1} J_{ij}\sigma_j) + h_i \tag{6}$$

where $j \in phase\, N$ means spins existing in phase N, especially spins adjacent to spin i to be updated.

The right side of Fig. 6 (a2) shows processing updates for the duplicated spin 4. In phase 1, interactions with *spins 2, 8, and 11* are calculated and carried to phase 2. Then, interactions with *spins 1* and *6* and the interaction in phase 1 are added in phase 2. Finally, the next state of *spin 4* is determined according to the computation result. No strong interaction is required for this calculation.

Next, we consider two methods propagating calculation results to the next phase. In the method shown on the left side of Fig. 6 (b), the position of spins connected to the high-degree spin is restricted because the high-degree spin remains at the same coordinates. Therefore, we extend the allocation position of the duplicated spin to the adjacent spin, as shown on the right side of Fig. 6 (b). This makes it possible to allocate graphs with arbitrary connections, as shown in Fig. 4 (f) in the time direction.

4.3 Reverse-Order Processing

When the range of continuous processing is expanded to adjacent spins, the spin unit where updating starts can differ from the one where updating finishes, as shown in Fig. 6 (c). In that case, it is necessary to synchronize the updated spin information when processing phase 1, after processing phase 3. However, writing back from an arbitrary point to an arbitrary point increases hardware cost.

Reverse-order processing solves this problem. In forward processing, after processing phases 1 to 3, it returns to phase 1. On the other hand, after processing phase 3, the processing is performed from phases 3 to 1 with reverse-order processing. This increases the memory locality with little additional resources.

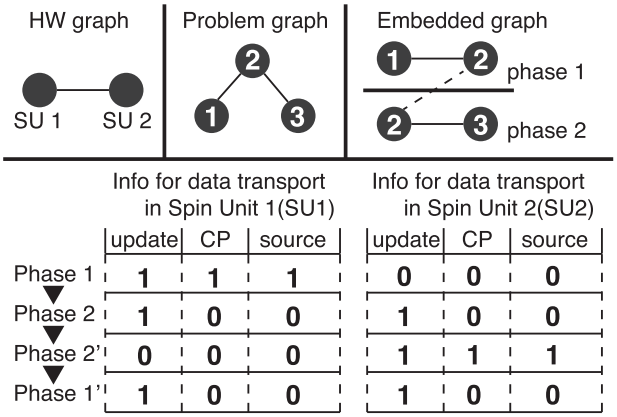


Fig. 7 Example for continuous processing and reverse-order processing

Figure 7 shows an operation example of Continuous Processing and Reverse-order Processing. When solving the problem graph with HW Graph with 2 spin units, The embedding process produces the embedded graph consisting of two phases. The processing order of phase is as shown in the lower left of Fig. 7, where phase 1 > 2 is defined as a forward order and phase 2' > 1' as a reverse order. Each spin unit stores three pieces of information for each phase: update flag, CP flag, source. The update flag indicates whether spin information is updated in that phase. CP flag and source indicate whether CP is performed and from which spin unit data should be received, respectively. In this example, the data source is represented by 1 bit because it is itself or the next spin unit. If the hardware graph is the king's graph, there are nine data sources for each spin unit, it is represented in 4 bits.

Spin unit 1 updates spin 1 and spin 2. In phase 1, spin unit 1 simultaneously updates spin 1 and receives in-progress results of spin 2 from spin unit 2. In phase 2, spin 2 is updated based on the result received on the previous phase and the calculation result between spin 2 and spin 3. Here, if it is attempted to resume calculation from phase 1 again, the updated spin 2 exists in spin unit 1 and spin 2 calculation cannot be performed. Therefore, reverse-order processing is introduced.

The source when forward order processing can be interpreted as the destination during reverse processing. Therefore, it is not necessary to prepare two sources for forward order processing and reverse order processing. Unfortunately, this was omitted to simplify implementation in this paper. We implement forward and reverse order processing with two memories.

4.4 Architecture Overview

We propose a time-division multiplexing architecture that processes complex and large combinational optimization problems with limited hardware resources by separating the

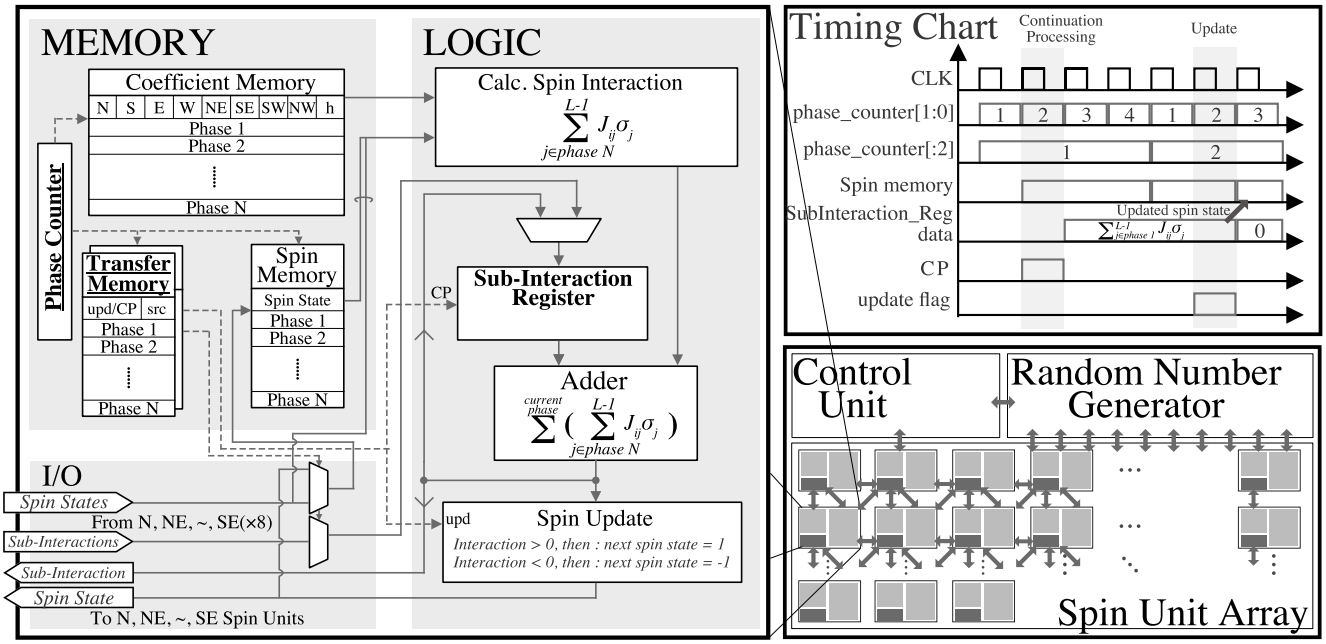


Fig. 8 Time-division multiplexing architecture

spins of a target problem in both spatial and temporal directions.

Our hardware architecture consists of a control unit, a random number generator, and a spin unit array, as shown in the lower right of Fig. 8. The control unit manages the annealing schedule. Based on the initial temperature, the cooling coefficient of temperature, and the number of steps per temperature, it supplies a threshold to control the influence of heat on the random number generator.

The random number generator compares the random number generated by XOR-shift to the threshold value, and random pulses are generated to calculate the interaction of the spin unit.

In the spin unit, the spin memory and coefficient memory are composed of an LUT in the conventional architecture [16]. With the proposed architecture, they are constituted on the premise of BRAM (Block RAM in Xilinx FPGAs), as shown on the left in Fig. 8. By using the phase counter, the data from the spin memory and coefficient memory are switched and supplied to the logic circuit, which calculates for the spin update. The phase counter uses the lower two bits as a counter for spin updates and uses the upper bit for phase management. When the phase counter counts up to a maximum phase (forward processing), the phase counter counts down to perform reverse processing. As mentioned above, there is the constraint that adjacent spins cannot be updated at the same time, so the phase is updated with four clocks.

Continuous processing is realized by the transfer memory and the sub-interaction register in both forward and reverse order processing. Each spin unit has two transfer memories for forward and reverse order processing. When the phase counter is counting up, transfer memory for the forward order processing is read, otherwise transfer memory for the reverse order processing is read After calculating the spin interaction, If the update flag(upd) in the transfer memory is 1, spin update is performed. the spin state is updated through the spin update and the result is written to the spin memory. If Continuous Processing flag (CP) is 1, The intermediate results selected based on the source in the transfer memory from the neighbor spin units and itself are stored in the sub-interaction register. Otherwise, the value of the register is set to 0.

The timing chart for the continuous update is shown in the upper right of Fig. 8. In this example, the spin in the spin unit that is updated secondly in the phase is updated over two phases by continuous processing. When the lower bit of the phase counter is two, this spin unit calculates the interaction based on the values obtained from the interaction coefficient memory and the spin memory. In this phase, the calculated interaction goes through the adder and is stored in the sub-interaction register since the spin state is not updated. The operation is managed by an update flag(upd=0) stored in the transfer memory. In the case of continuous processing, CP in the transfer memory is 1 and the sub-interaction register preserves the value of the interaction based on this value. In phase 2, the newly calculated interaction is added to the value saved in the sub-interaction register by the adder. In this phase, since 1 is stored in the update flag of the transfer memory, the spin update updates the spin state. The value in the sub-interaction register is reset because the CP the transfer memory is 0.

5. Evaluation

5.1 Setup

We evaluated the resource utilization of our architec-

ture and the conventional CMOS annealing processor (**CMOS-AP**) [16] because our architecture was devised based on [16]. The proposed architecture with continuous processing is hereafter referred to as **TDM with CP**, and our previous architecture [2] is referred to as **TDM**. The resource utilization of the Entire design is the result obtained for the hardware in which the spin units are composed of chimera topology. Our target board was the Xilinx Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit (FPGA: Zynq UltraScale XCZU9EG-2FFVB1156). Both architectures were synthesized in Verilog HDL and Vivado Design Suite 2016.2.

In this evaluation, we assume that all the hardware configurations in the evaluation run at 100MHz, and we did not survey the maximum clock frequency, because the main purpose of this work is to confirm the improvement of annealing accuracy and feasibility of our architecture in point of hardware resource utilization. Please note that the proposed architecture requires a few components in addition to the baseline hardware. So, it is expected that the maximum clock frequency of the proposed architecture is comparable to **CMOS-AP**.

For the current implementation, it was not possible to evaluate the total amount of resources for the three methods. Therefore, we used TDM to compare the overall resource consumption of **CMOS-AP**. since TDM and TDM with CP have almost the same configuration except for the spin unit. Then, we compared the resource consumption of spin units in the three approaches.

To illustrate the accuracy and the convergence speed of our architecture for data expressed in graphs that are more complicated than the hardware topology, we evaluated datasets of 10 max-cut problems that were randomly generated. We created a dataset that does not require spin duplication by ME, consisting of 100 vertices with random spin-to-spin interactions and connections. Based on that dataset, nine problems with different numbers of replicated spins were created with randomly added connections exceeding the maximum hardware degree. The number of replicated spins was 10 at intervals of 10 to 90. When there are some duplicate spins, the total number of spins increases accordingly. In this experiment, the total number of spins is increased to 190 from 100.

The max cut problem finds two vertex groups that maximize the weight of the edge for each endpoint that belongs to a different group. Therefore, the desired spin configuration has a high score in terms of figures. The score was calculated based on the cost function of the max-cut problem with the spin state after "burnout" to get the final state: a state where there is no influence of the thermal fluctuation. These datasets were evaluated by software simulation. The simulator used in this study is the in-house simulator written in Python. In the simulator, the number of temperature updates (N), the number of steps at the same temperature (L)), the initial temperature (T), and the temperature decay coefficient (β) are used as parameters specified by the user. The entire annealing step is determined by $N \times L$. The update of the temperature can be obtained by multiplying the current temperature by β.

Table 1 Resource utilization of the entire design

	#. Spin Unit	LUT	FF	BRAM
Available		537600	1075200	1728
CMOS AP	4	682	833	0.5
	16	1305	905	0.5
	64	3897	1193	0.5
TDM	4(×4 phases)	663	825	4
	8(×8 phases)	832	831	4

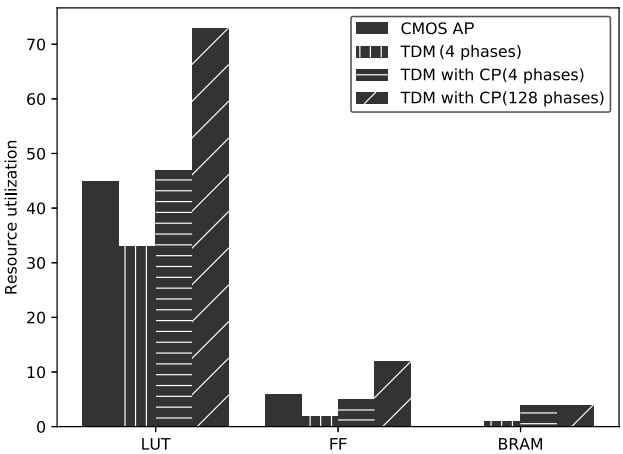


Fig. 9 Resource Utilization per Spin Unit

5.2 Resource Utilization

Table 1 shows the resource consumption of (CMOS-AP) and (TDM). The second column of the table shows the number of spin units. In CMOS-AP, this number matched the number of spins that could be processed. On the other hand, spin units in TDM could process different spins in each phase. Therefore, the number of spins that could actually be processed is the number of spin units multiplied by the number of phases.

As shown in the table, when the number of spins that could be processed was the same, the consumption of the LUT and FF was lower in TDM than in CMOS-AP. This is because the number of spin units required for processing is reduced by time-division multiplexing. Of course, it takes more time to update the spin state with time-division multiplexing. However, it is possible to solve problems that cannot be processed with CMOS-AP.

When the number of spin units of CMOS-AP and TDM was the same (e.g., the third and rows), TDM had a slightly smaller LUT and FF consumption than CMOS-AP. This is because CMOS-AP stores spin interaction coefficients and spin states using an LUT and FF, whereas TDM with CP stores them using BRAMs.

A summary of resources for each spin unit with each method is shown in Fig. 9. Comparing CMOS-AP and TDM, TDM had less resource consumption in terms of the LUT and FF. On the other hand, the resource consumption of the

Table 2 Resource utilization of a random pulse generator and a spin unit

	Random pulse generator			Spin unit		
	LUT	FF	BRAM	LUT	FF	BRAM
CMOS-AP	501	809	0.5	45	6	0
TDM	509	809	0.5	47	5	4

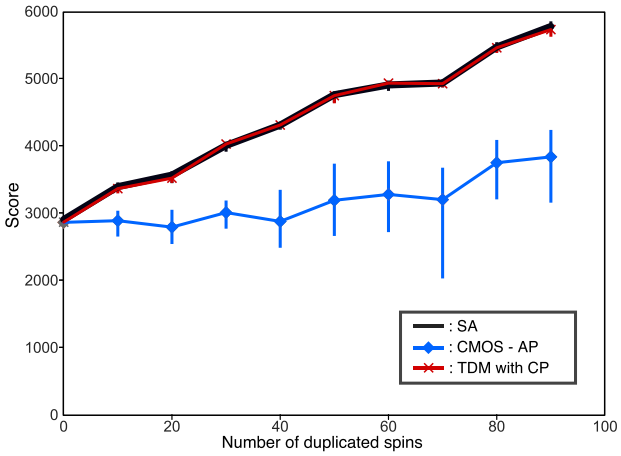


Fig. 10 Influence of duplicated spins on the score

LUT and FF increased as the number of phases increased with TDM with CP. This is likely because increasing the number of phases increases the bit width of the adder for calculating the interactions and the bit width of the register for storing the intermediate result.

Table 2 shows the resource consumption per random number generator, control unit, and spin unit. The control parts are shared by the random number generators. Thus, the hardware resource amount will not increase very little, in contrast to the spin units. Then, the spin unit consumed very little resources in terms of the random number generators and control unit with both methods. Therefore, the increase in the resource consumption of the spin unit does not considerably affect the overall performance.

5.3 Performance

Accuracy Comparison

Figure 10 shows the simulation results of the score with respect to the aforementioned 10 max-cut problems. We compared our architecture with CMOS-AP and simulated annealing (SA). The x- and y-axes of the graph indicate the number of replicated spins included in the problem and scores, respectively. The solid line indicates the average score obtained after 16 trials for each dataset. Error bars indicate the range of scores obtained in these 16 trials. The number of trials is 16, which is determined by considering the number of spins in the evaluation data set. Please note that there is a slight variation of obtained scores in our experiments. The parameters of TDM with CP, CMOS-AP and SA were set as $[N = 100 \sim 150, L = 3, T = 100, \beta = 0.96]$, $[N = 200 \sim 300, L = 5, T = 100 \sim 190, \beta = 0.98]$ and

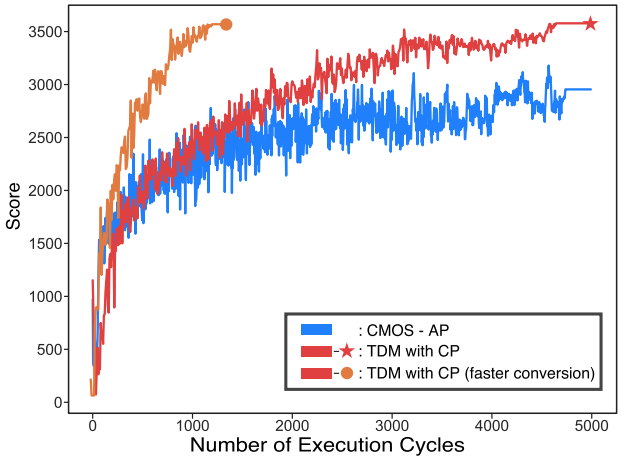


Fig. 11 Score transition in CMOS annealing and TDM with CP

$[N = 100 \sim 150, L = 30, T = 100, \beta = 0.96]$, respectively. In each problem, both CMOS AP and TDM with CP consume 4,500 ~ 5,500 execution cycles.

As shown in the figure, SA, CMOS AP, and TDM with CP obtained solutions with nearly the same precision in the case of the dataset that did not require replicated spins. This indicates that CMOS AP and TDM with CP can search for solutions without compromising the accuracy of the solution when the complexity of the dataset is lower than the hardware topology. Even when TDM with CP processed data with replicated spins, it reached solutions with the same accuracy as SA. On the other hand, CMOS AP resulted in significantly degraded accuracy when processing data that included replicated spins. In addition, the range of the obtained score varied greatly. This became more prominent as the number of replicated spins increased.

There are two causes for the degradation of accuracy. First, CMOS-AP processes replicated spins as separate spins internally. Therefore, the frustration that occurred in the original spin changes. Second, the strong interaction is added to treat replicated spins as the same spin. Since this strong interaction is greater than the sum of the original interactions, the original interaction does not affect the spin state update. In other words, the state of the spin changes only with a random number for the strong interaction. Therefore, as the number of replicated spins increases, the obtained solution varies. TDM solves these problems by internally processing the duplicated spin as the same spin with continuous processing, and by replacing the strong interaction with the constraint regarding the placement of the replicated spin within the adjacent spin of the next phase.

Convergence Speed

Figure 11 shows the transition of the score when a dataset with 20 duplicated spins was processed by CMOS-AP and TDM with CP. The x-axis of the graph shows the number of execution cycles and the y-axis shows the score at that point. All spin units update at each clock. In the con-

ventional CMOS-AP, all spins are updated in four clocks, whereas the proposed architecture requires four clocks for each phase. In this simulation, eight clocks were needed for TDM with CP to process all spins, since all spins were embedded in two phases. In order to compare the convergence speed, we evaluated TDM with CP with two types of temperature schedules. One was set so as to converge to a solution with the same execution cycles in both CMOS-AP and TDM with CP. In this work, the annealing schedule is determined according to the number of spins in the problem. The total number of spins in the embedded problem increased as the number of duplicated spins increased with CMOS-AP while TDM with CP processes duplicated spins as one spin by continuous processing. We heuristically determined the annealing schedules under the condition that the solution accuracy of all 16 trials was 90% or more of **the best score of the 16 trials**. Therefore, we adjusted the scheduling of TDM with CP based on the CMOS-AP scheduling so that the number of execution cycles until the solution converges are the same. The other schedule was adjusted such that the system in TDM converged most quickly under the condition that the solution accuracy of all 16 trials was 90% or more of **the optimal solution**. The endpoints of the score transition in each schedule are represented by circles and stars.

As shown in the graph, even when convergence was performed by applying the same cycles, the accuracy of the solution reached by TDM with CP was higher than that by CMOS-AP. In addition, the score of the spin state obtained during the transition was often better than the ultimate solution from CMOS-AP. Since the duplicated spins were not updated due to the strong interaction, replicated spin states were almost decided by random numbers. Therefore, the score improves when good random numbers happen to be allocated to duplicated spins.

In a tight temperature schedule, TDM with CP converged about four times faster than CMOS-AP without loss in accuracy. Further, CMOS-AP did not arrive at the optimal solution after many trials. In view of convergence speed and solution precision, then, it is preferable to provide a hardware topology that can be embedded without spin replication when complicated problems are solved by APs.

6. Conclusion

In this paper, we proposed a time-division multiplexing architecture that processes complex and large combinational optimization problems. We evaluated our architecture from the viewpoint of resource consumption, solution accuracy, and convergence speed. With the proposed architecture, spin arrays are reused with BRAM-based time-division processing, and there is a virtual increase to the degree of hardware spin from continuously updating the target spin between spin arrays from different times as the same spin.

Our results indicate the following:

- If the duplicated spins by ME are processed on the same spin array, the solution accuracy decreases by 20 to 30%.
- Processing the replicated spin on the spin array from a different time (phase) by time-division processing is not affected by strong interaction between replicated spins. Therefore, it is possible to obtain a solution with the same accuracy as SA.
- There is little effect on hardware resources from introducing time-division processing.

Based on the above, our proposed architecture greatly improves the solution accuracy and convergence speed by introducing time-division processing.

In future research, we shall research an efficient embedding method for the time-division spin array. With time-division processing, the time required for processing increases as the number of phases increases. Therefore, we shall consider both the hardware topology (to facilitate embedding) and the embedding method in which the number of phases decreases by processing simultaneous updatable spins without spin-to-spin connections in the same phase.

Acknowledgments

This work is supported in part by JST PRESTO JPMJPR18M9 and JST CREST JPMJCR18K3.

References

[1] M. Yamaoka, C. Yoshimura, M. Hayashi, T. Okuyama, H. Aoki, and H. Mizuno, "24.3 20k-spin ising chip for combinational optimization problem with cmos annealing," 2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers, pp.1–3, Feb. 2015.

[2] K. Yamamoto, W. Huang, S. Takamaeda-Yamazaki, M. Ikebe, T. Asai, and M. Motomura, "A time-division multiplexing ising machine on fpgas," Proceedings of the 8th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies, HEART2017, New York, NY, USA, pp.3:1–3:6, ACM, 2017.

[3] S. Kahruman, E. Kolotoglu, S. Butenko, and I.V. Hicks, "On greedy construction heuristics for the max cut problem," Int. J. Comput. Sci. Eng., vol.3, no.3, pp.211–218, April 2007.

[4] A. Lucas, "Ising formulations of many np problems," Frontiers in Physics, vol.2, p.5, 2014.

[5] T. Skotnicki, J.A. Hutchby, T.-J. King, H.-S.P. Wong, and F. Boeuf, "The end of cmos scaling: toward the introduction of new materials and structural changes to improve mosfet performance," IEEE Circuits and Devices Magazine, vol.21, no.1, pp.16–26, Jan. 2005.

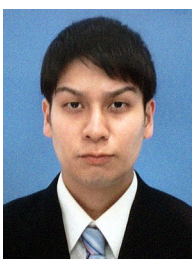
[6] M.W. Johnson, M.H.S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A.J. Berkley, J. Johansson, P. Bunyk, E.M. Chapple, C. Enderud, J.P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M.C. Thom, E. Tolkacheva, C.J.S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose, "Quantum annealing with manufactured spins," Nature, vol.473, pp.194–198, 2011.

[7] S. Utsunomiya, K. Takata, and Y. Yamamoto, "Mapping of ising models onto injection-locked laser systems," Opt. Express, vol.19, no.19, pp.18091–18108, Sept. 2011.

[8] Y. Lin, F. Wang, X. Zheng, H. Gao, and L. Zhang, "Monte carlo simulation of the ising model on fpga," Journal of Computational Physics, vol.237, pp.224–234, 2013.

[9] J. Cai, W.G. Macready, and A. Roy, "A practical heuristic for finding graph minors," CoRR, vol.abs/1406.2741, 2014.

[10] H. Gyoten, M. Hiromoto, and T. Sato, "Area efficient annealing processor for ising model without random number generator," IEICE Transactions on Information and Systems, vol.E101-D, no.2, pp.314–323, 2018.
[11] K. Someya, R. Ono, and T. Kawahara, "Novel ising model using dimension-control for high-speed solver for ising machines," 2016 14th IEEE International New Circuits and Systems Conference (NEWCAS), pp.1–4, June 2016.
[12] S. Tsukamoto, M. Takatsu, S. Matsubara, and H. Tamura, "An accelerator architecture for combinatorial optimization problems," 2017.
[13] T. Okuyama, C. Yoshimura, M. Hayashi, and M. Yamaoka, "Computing architecture to perform approximated simulated annealing for ising models," 2016 IEEE International Conference on Rebooting Computing (ICRC), pp.1–8, Oct. 2016.
[14] H. Gyoten, M. Hiromoto, and T. Sato, "Enhancing the solution quality of hardware ising-model solver via parallel tempering," Proceedings of the International Conference on Computer-Aided Design, ICCAD '18, New York, NY, USA, pp.70:1–70:8, ACM, 2018.
[15] K. Terada, D. Oku, S. Kanamaru, S. Tanaka, M. Hayashi, M. Yamaoka, M. Yanagisawa, and N. Togawa, "An ising model mapping to solve rectangle packing problem," 2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT), pp.1–4, April 2018.
[16] C. Yoshimura, M. Hayashi, T. Okuyama, and M. Yamaoka, "Implementation and evaluation of fpga-based annealing processor for ising model by use of resource sharing," International Journal of Networking and Computing, vol.7, no.2, pp.154–172, 2017.
[17] V. Choi, "Minor-embedding in adiabatic quantum computation: I. the parameter setting problem," Quantum Information Processing, vol.7, no.5, pp.193–209, Oct. 2008.
[18] C.Y. Takuya Okuyama, M. Hayashi, S. Tanaka, and M. Yamaoka, "Contractive graph-minor embedding for cmos ising computer (in japanese)," tech. rep.



Kasho Yamamoto received the B.E. and M.E. degrees from Hokkaido University, Sapporo, Japan, in 2015 and 2017, respectively, where he is currently pursuing the Ph.D. degree, with a focus on efficient hardware architecture for annealing machine based on Ising model. He received a Research Fellowships for Young Scientists from JSPS in 2018.

Masayuki Ikebe received his B.S., M.S., and Ph.D.degrees in electrical engineering from Hokkaido University, Sapporo, Japan, in 1995, 1997, and 2000, respectively. During 2000–2004, he worked for the Electronic Device Laboratory, Dai Nippon Printing Corporation, Tokyo, Japan, where he was engaged in the research and development of wireless communication systems systems and image processing systems. Presently, he is a Professor and leads the Integrated Quantum Systems group at the Research Center For Integrated Quantum Electronics (RCIQE), Hokkaido University. His current research interests are analog and mixed-signal IC design with an emphasis on CMOS image sensor, THz imaging devices and intelligent image processing systems. He is a member of IEICE and IEEE.

Tetsuya Asai received his B.S. and M.S. degrees in electronic engineering from Tokai University, Japan, in 1993 and 1996, respectively, and his Ph.D. degree from Toyohashi University of Technology, Japan, in 1999. He is now a Professor in the Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan. His research interests are focused on developing intelligent integrated circuits and their computational applications. Current topics that he is involved with include emerging research architectures, deep learning accelerators, and device-aware neuromorphic VLSIs.

Masato Motomura received the B.S., M.S.,and Ph.D. degrees in electrical engineering from Kyoto University, Kyoto, Japan, in 1985, 1987, and 1996, respectively. In 1987,he joined the NEC central research laboratories, where he worked on various hardware architectures including string search engines, multi-threaded on-chip parallel processors, embedded DRAM field-programmable gate array (FPGA) hybrid systems, memory-based processors, and reconfigurable systems. From 2001 to 2008, he was with NEC Electronics where he led research and business development of dynamically reconfigurable processor (DRP) that he invented. He was also a visiting researcher at MIT laboratory for computer science from 1991 to 1992. From 2011 to 2019, he was a professor at Hokkaido University. He has been a Professor at Tokyo Institute of Technology, Japan, since 2019. His current research interests include reconfigurable and parallel architectures for deep neural networks,machine learning, annealing machines, and intelligent computing in general. Dr. Motomura is a member of IEICE, IPSJ, and EAJ. He was a recipient of the IEEE JSSC Annual Best Paper Award in 1992, the IPSJ Annual Best Paper Award in 1999, the IEICE Achievement Award in 2011, and the ISSCC Silkroad Award as the corresponding author in 2018, respectively.

Shinya Takamaeda-Yamazaki received the B.E, M.E, and D.E degrees from Tokyo Institute of Technology, Japan in 2009, 2011, and 2014 respectively. From 2011 to 2014, he was a JSPS research fellow (DC1). From 2014 to 2016, he was an assistant professor of Nara Institute of Science and Technology, Japan. From 2016 to 2019, he was an associate professor of Hokkaido University, Japan. Since 2018, he has been a researcher of JST PRESTO. Since 2019, he has been an associate professor of The University of Tokyo, Japan. His research interests include computer architecture, high-level synthesis, and machine learning acceleration. He is a member of IEEE, IEICE, and IPSJ.