IEICE TRANS. INF. & SYST., VOL.E101–D, NO.2 FEBRUARY 2018

PAPER Special Section on Reconfigurable Systems

Area Efficient Annealing Processor for Ising Model without Random Number Generator

Hidenori GYOTEN[†a)], ***Student Member***, **Masayuki HIROMOTO**[†], ***and*** **Takashi SATO**[†], ***Members***

SUMMARY An area-efficient FPGA-based annealing processor that is based on Ising model is proposed. The proposed processor eliminates random number generators (RNGs) and temperature schedulers, which are the key components in the conventional annealing processors and occupying a large portion of the design. Instead, a shift-register-based spin flipping scheme successfully helps the Ising model from stucking in the local optimum solutions. An FPGA implementation and software-based evaluation on max-cut problems of 2D-grid torus structure demonstrate that our annealing processor solves the problems 10–10^4 times faster than conventional optimization algorithms to obtain the solution of equal accuracy.

key words: *combinatorial optimization problem, max-cut problem, Ising model, annealing, FPGA*

1. Introduction

Efficient social infrastructures and systems are essential to our daily lives. It is desirable to reduce the logistics cost of supply chain, to shorten travel time, to minimize the distribution loss of energy resources, etc. Most of these practical problems are known to be formulated as combinatorial optimization problems. However, it is very hard to obtain the optimal solutions in a practical time as the size of the problems becomes large, mostly due to the exponential nature of their computational complexity. Although efficient heuristic algorithms have been actively developed [1]–[3], they still require unacceptably large computation time to obtain solutions, or the quality of the solutions becomes insufficient.

Besides these conventional heuristic algorithms, an alternative solver that utilizes the *Ising model* is gaining increasing attention. The Ising model is a mathematical model of ferromagnetism in statistical mechanics [4]. The model consists of *spins*, each of which takes one of two discrete states $\{+1, -1\}$. The spins are generally arranged in a lattice form as shown in Fig. 1. The spin's value is determined such that the local energy of the spin decreases according to the states of the adjacent spins considering the interactions with adjacent spins, as shown in Fig. 2. This local update decreases the energy of the entire Ising model, which facilitates to solve the problems in a highly parallel manner, thus achieving a fast optimization.

A quantum annealing computer, D-Wave [5], is one of the Ising-model-based solvers, which emulates the behavior of the Ising model by quantum annealing [6], [7]. It is reported that the weak-strong cluster pair problem, which is one of the combinatorial optimization problems, has been solved 1.8×10^8 times faster than simulated annealing [8]. However, a superconducting flux qubit, the key element of the D-Wave, requires cryogenic temperature during the operation. This makes the computer system significantly complex and expensive. Other approach includes CMOS annealing [9]. It simulates the annealing of the Ising model on a general CMOS logic circuit, which realizes the circuit operation at a room temperature. This method has advantages of inexpensive implementation and easier integration with conventional computer systems. Recent research includes an implementation of the CMOS annealing on FPGA [10] and a hardware architecture for fully-connected Ising model [11].

In this paper, we focus on the architecture based on the concept of CMOS annealing and propose an area-efficient annealing processor that can be implemented using fully digital circuits, such as in an FPGA. One of the most important challenges for the annealing processor is area efficiency. The performance of such processors directly depends on the parallelism achieved, i.e., the number of spins that can be implemented on a single chip. The spin itself can be easily realized by a simple small circuit. However, the annealing processor requires an additional function to simulate the

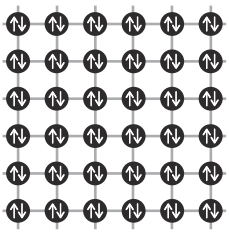

Fig. 1 Typical structure of the Ising model.

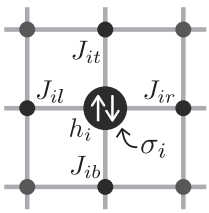

Fig. 2 A spin in the Ising model. Each spin has interactions with adjacent spins.

Manuscript received May 8, 2017.
Manuscript revised September 8, 2017.
Manuscript publicized November 17, 2017.

[†]The authors are with Department of Communications and Computer Engineering, Graduate School of Informatics, Kyoto University, Kyoto-shi, 606–8501 Japan.
a) E-mail: paper@easter.kuee.kyoto-u.ac.jp

DOI: 10.1587/transinf.2017RCP0015

probabilistic behavior of the Ising model. Random number generators (RNGs) and temperature scheduler are typically adapted for this purpose, which have been occupying a large hardware resource, as in the case of [10].

In this work, we propose a novel shift-register-based spin flipper (SRSF) that helps the annealing process to converge without using random numbers. This approach significantly improves the area efficiency of the annealing processor by eliminating large RNGs, and thus improves its scalability. Our proposed annealing processor has the following three features: (1) a simple SRSF circuit that can emulate the annealing behavior, (2) scalable architecture that can be configurable for the size of the problems to solve, and (3) fully synthesizable digital design that can be easily implemented on an FPGA. Owing to these features, we can successfully implement a 2000-spin Ising model on an FPGA of moderate capacity to use as the accelerator of the combinatorial optimization solver. We solve max-cut problems of 2D-grid torus structures by the proposed annealing processor and compare its efficiency and accuracy to the existing software solvers. The experimental results show that our processor can find quasi-optimum solutions 10–10^4 times faster than the existing software solvers.

The contribution of this paper is summarized as follows:

- An area-efficient annealing processor for the Ising model is proposed, which eliminates the resource-consuming random number generators, by employing a simple shift-register-based spin flipper with a negligible area overhead.
- The proposed processor is implemented on an FPGA and its performance is quantitatively evaluated using max-cut benchmark problems [12]. The results are also compared with software implementations.
- Our annealing processor can solve the max-cut problems several orders faster than the existing fastest approximation algorithm [1].

The rest of this paper is organized as follows. Section 2 provides preliminary knowledge about the Ising model. Section 3 proposes the architecture of our annealing processor and the SRSF methods to improve the area efficiency without using RNGs. Section 4 shows the evaluation and the results, and finally, Sect. 5 concludes the paper.

2. Ising Model

2.1 Definition

The Ising model [4] consists of a set of spins that are interconnected with each other with a weight. The connection of the spins can take any topology, representatively a lattice form as shown in Fig. 1, which is a most typical example. For each spin i, the spin has a discrete spin value $\sigma_i \in \{+1, -1\}$ and interaction weight J_{ij} with an adjacent spin j, as shown in Fig. 2. The local energy of spin i is given by

$$H_i(\sigma_i) = -\sum_j J_{ij}\sigma_i\sigma_j - h_i\sigma_i, \tag{1}$$

where h_i is an external magnetic field (or a bias). Thus, the total energy of the entire model is given by

$$H = -\sum_{\langle i\,j\rangle} J_{ij}\sigma_i\sigma_j - \sum_i h_i\sigma_i. \tag{2}$$

Note that $\langle i\,j\rangle$ indicates to take sum of all spin pairs. The value of each spin σ_i is determined according to the interactions with adjacent spins, such that the local energy H_i is minimized. By repeatedly updating all spins independently, the total energy of the Ising model, H, decreases and converges to a minimum value.

2.2 Optimization Utilizing the Ising Model

Combinatorial optimization problems can be solved by utilizing the Ising model by the following four steps:

Step 1: Formulation Represent the real-world problem of interest as a combinatorial optimization problem and formulate equations in the form of the energy minimization of the Ising model in Eq. (2).

Step 2: Mapping Assign parameters (interactions and biases) to the model according to the formulation.

Step 3: Annealing Update spin values repeatedly until convergence is obtained.

Step 4: Interpretation Convert the final spin values back into the original optimization problem.

In the following subsections, each step will be explained in detail.

2.2.1 Formulation

The first step of the formulation is to express the real-world problem as a combinatorial optimization problem. The main part of this formulation step is to convert the combinatorial optimization problem to the form of the energy minimization of the Ising model of Eq. (2). We consider the max-cut problem, which is one of the NP-hard problems, but other problems can be also formulated as summarized in [13].

The objective of the general max-cut problem is to obtain a vertex set such that the total weight of the edges between the vertices in the subset and the complementary subset becomes the largest. In the example of Fig. 3, the maximum cut is 25, when {1, 2, 4} is chosen as the subset.

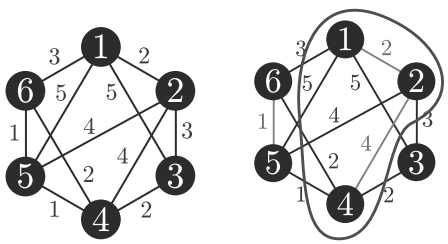

Fig. 3 An example of the max-cut problem and its solution.

The max-cut problem is formulated as follows:

$$\max \frac{1}{2} \sum_{i,j \in V, i<j} w_{ij}(1 - x_i x_j), \tag{3}$$

where V is a set of vertices, w_{ij} is a weight of the edge between the vertices i and j, and $x_i = \{-1, 1\}$ is an indicator that represents the vertex i belongs to the subset or not. Here, our objective is to convert the max-cut problem represented by Eq. (3) to the energy minimization of the Ising model in Eq. (2). By comparing these equations, they become equivalent when we associate the variables as $J_{ij} = -w_{ij}$, $\sigma_i = x_i$, and $h_i = 0$. The spin corresponds to the cut, and the minimum energy of the Ising model gives the maximum cut.

2.2.2 Mapping

In the mapping step, according to the formulation obtained in the previous step, parameters (interactions and biases) are assigned to the hardware Ising model. This process is called "embedding" in [14]. In general, both the formulated problem and the structure of the hardware Ising model can be treated as graphs. The formulated problem can be represented as a graph $G = (V, E)$, where V is a set of vertices that represents logical variables $\{\sigma_i\}$ and E is a set of edges that represents interactions $\{J_{ij}\}$. We can also obtain the graph $G' = (V', E')$ that represents the structure of the hardware Ising model. In order to embed G into G', G should be a subgraph of G'. In this case, it is possible to simply embed G into G', if the degree of all V is equal to or smaller than that of V' and the graph G' is sufficiently large. Otherwise, particularly when the maximum degree of V exceeds that of V', a vertex $\sigma_i \in V$ should be represented using a group of duplicated multiple vertices, called "a clone," such that the all clones attain sufficient number of edges that realize the necessary connections of all vertices in the problem to solve, as shown in Fig. 4. Here, interactions between the vertices in a clone are determined so that the vertices tend to take the same spin value. This process is called "graph minor embedding," which is known as an NP-hard problem [15].

The reduction of the execution time of graph minor embedding, the minimization of the number of the clones to be generated, and the determination of the optimal interactions between clones are currently the topics of intense researches. The detail of the general mapping algorithm is beyond the scope of this paper, so we limit ourselves to refer the algorithms proposed in [14], [15]. Similarly, the problems solved in the later section are limited to be a 2D-grid torus graph structure.

2.2.3 Annealing

In this step, each spin is iteratively updated so that the local energy of the spin becomes smaller. Since a change of a spin value will affect its neighbors, the iteration is required so that the change propagates in the Ising model. The energy

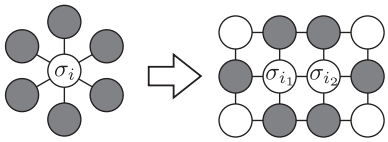

Fig. 4 An example of minor embedding for a six-edge vertex to two four-edge vertices. The spin σ_i is duplicated using two clones, σ_{i_1} and σ_{i_2}.

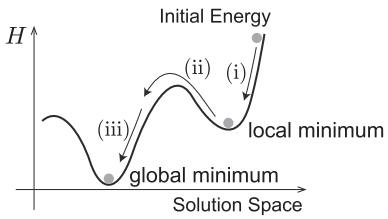

Fig. 5 Optimization by annealing.

Algorithm 1 Annealing Process of the Ising Model

```
N_RF ← K
Initialize σ_i
for n = 1 to N do
    for all i ∈ 𝔾_n do
        σ_i ← argmin H_i(σ_i)
    end for
    Flip N_RF spins (random flip)
    N_RF ← α_n N_RF (α_n < 1)
end for
```

reduction of each spin basically decreases the total energy of the Ising model, but the reduction may stuck in a local minimum as shown in Fig. 5 (i). To get out of the local minimum to continue to find the global minimum, temporary energy increase is necessary. In the Ising model, this is realized by a "random flip (RF)," in which a set of spins is randomly chosen and their values are flipped [9]. By the RF, the energy of the entire Ising model once increases as shown in Fig. 5 (ii), but eventually decreases further as the local minimization is repeated. This is how the global minimum is obtained (Fig. 5 (iii)).

In order to ensure convergence, "temperature scheduling" during the optimization process is important. The temperature is the metaphor of the energy input to the spins. When the temperature is high, a large number of spins are randomly flipped to increase the probability of escaping out of possible local minima. On the other hand, when the temperature is low, few spins are flipped and it is easy to converge to the steady state. To find the global minima, gradually decreasing the temperature from high to low is effective. This temperature scheduling is called annealing.

Algorithm 1 shows the detailed procedure of the annealing. N_{RF} is the number of spins to be randomly flipped at a time, and K is its initial value. The following operations are repeated for N times. First, a set of spins to be updated, $\mathbb{G}_n$, is determined. It can be a single spin, an entire set of spins, or a subset of spins partitioned like a checkerboard [10]. Second, each spin in $\mathbb{G}_n$ is updated so that its

local energy is minimized. Finally, the RF is executed and the temperature is lowered by multiplying α_n (< 1) to N_{RF}.

Note that an update of spin values can be executed in a fully parallel manner because the individual spin depends only on the adjacent spins. This is a great advantage for implementing the Ising model on highly parallel hardware architecture.

2.2.4 Interpretation

This step converts the final spin states with minimum energy after the annealing process to the solution of the original optimization problem. It is realized just by the reverse process of the formulation. For example, in the case of the max-cut problem, the maximum cut can be obtained by calculating Eq. (3) with $x_i = \sigma_i$.

3. Annealing Processor for the Ising Model

Recently, several specialized processors motivated by the Ising model have been proposed, such as [5], [9]–[11]. Similarly to the design in [10], our proposed annealing processor is fully synthesizable, is targeting an FPGA implementation, but our processor specifically aims to achieve better area efficiency. This section first describes the base architecture of the proposed annealing processor with the details of its key components, and then describes the proposed two techniques to improve the area efficiency.

3.1 Base Architecture of Annealing Processor

The architecture of the proposed annealing processor is presented in Fig. 6. The basic units called *Ising cells* (`ISC_ij`) are arranged in an $m \times n$ lattice form. Though the proposed architecture is not bound to a limited network topology, the target Ising model here is assumed to be in a lattice form, in which each spin has interactions with the left, right, top, and bottom adjacent spins.

Each Ising cell corresponds to a single spin and performs operations for spin update, which will be detailed in Sect. 3.1.1. The subscripts `i` and `j` of each cell are indices that represent the cell location in the array. Each cell `ISC_ij` is connected with its neighbors: left `ISC_i(j-1)`, right `ISC_i(j+1)`, top `ISC_(i-1)j`, and bottom `ISC_(i+1)j`. Interaction weights between cells are stored in the register `J_ijk`. For the index variable k, "v" represents the vertical interaction weights J_{ijv} between `ISC_ij` and `ISC_(i+1)j`, and "h" represents the horizontal interaction weights J_{ijh} between `ISC_ij` and `ISC_i(j+1)`.

All the registers for the variables J_{ijk}, σ_{ij}, and h_{ij} (the last two are inside the Ising cell) form independent scan chains for initialization and result output. Note that the scan chains are not depicted in the figure for clarity. A controller module changes operation mode of the processor, to control I/O and annealing behavior, which will be described in Sect. 3.1.3.

The annealing processor is fully synthesizable, and hence it can easily be implemented on an FPGA. Although the example architecture in Fig. 6 is a simple 2-D array structure, we can configure the processor to fit various problems by changing the number of connections between spins or adopting torus structure. A synthesized instance of the annealing processor with a certain configuration can be applicable to solve various problems without resynthesizing the instance as long as the numbers of spins and connections do not exceed its capacity. When solving the other problems, only an initialization of the registers that are storing the variables J_{ijk}, σ_{ij}, and h_{ij} are required, which can be carried out in a very short time. However, if we want to solve the problems that do not fit the instance, resynthesis and reconfiguration of the FPGA are required.

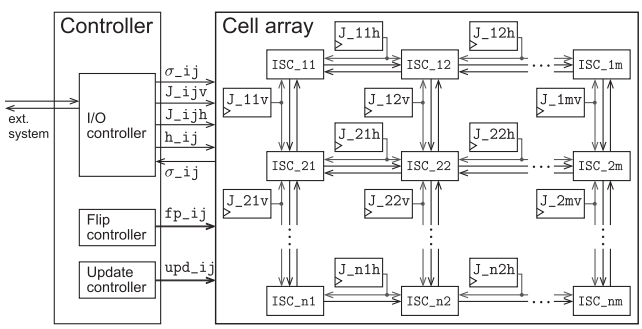

Fig. 6 Overall architecture of the proposed annealing processor. It consists of a controller and a 2D-array of Ising cells (`ISC`s), each of which is connected to the adjacent `ISC`s with interactions `J`.

3.1.1 Ising Cell

The Ising cell, `ISC_ij`, holds the corresponding spin value σ_{ij} in the register σ`_ij` and the bias h_{ij} in `h_ij`. It performs local energy minimization through the spin update operation. The internal structure of the cell is shown in Fig. 7. The inputs of the cell are the adjacent spin values, σ`_(i-1)j`, σ`_(i+1)j`, σ`_i(j-1)`, and σ`_i(j+1)`, and the interaction weights to the adjacent cells, `J_(i-1)jv`, `J_(i+1)jv`, `J_i(j-1)h`, and `J_i(j+1)h`. According to Eq. (1), the Ising cell calculates the next spin direction and updates its spin under the control of the update signal `upd_ij` and the flip signal `fp_ij`.

The Ising cell works as follows. The spin determination module `Spin_Det` determines the spin value `s_det` so

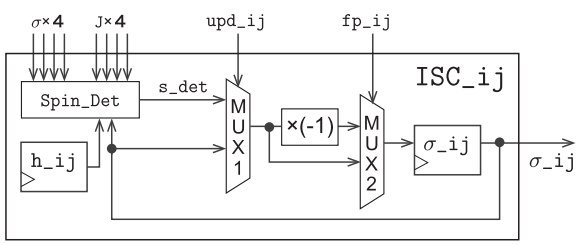

Fig. 7 The internal structure of the Ising cell. It calculates the next spin state from the adjacent spins σ, interactions J, and bias h_{ij}, and updates its spin under the control of the update signal `upd_ij` and flip signal `fp_ij`.

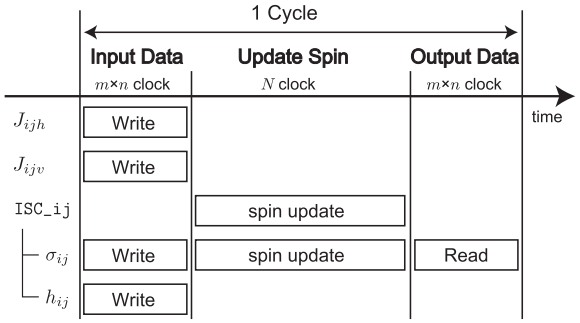

Fig. 8 Operation flow of the proposed annealing processor to solve a problem.

that the local energy is minimized. The multiplexer `MUX1` determines whether to update the spin or not, by selecting the output of `Spin_Det` or the current state. Then the multiplexer `MUX2` determines whether the spin is flipped or not according to the flip signal `fp_ij`. If `fp_ij` = 1, the flipped (multiplied by −1) spin value is selected, otherwise, the original value is selected. Finally, the output of the multiplexer is stored to the register σ`_ij`, which is the final output of this module.

3.1.2 Spin Determination Module

This module is inside the Ising cell and calculates the spin value by $\sigma_i = \mathrm{argmin}\, H_i(\sigma_i)$. From Eq. (1), the local energy H_i can be calculated by

$$H_i = -\left(\sum_j J_{ij}\sigma_j + h_i\right)\sigma_i = -\mathtt{sum} \times \sigma_i. \tag{4}$$

This means that σ_i can be determined just from the sign of the value, i.e., if `sum` > 0, $\sigma_i = +1$, if `sum` < 0, $\sigma_i = -1$.

3.1.3 Controller

The controller controls the operation mode of the annealing processor according to the operation flow in Fig. 8. In the "Input Data" phase, bit stream for the problem is shift-in to initialize the registers of J_{ijk} and h_{ij} using the scan chains. The initial values of the spins σ_{ij} are also loaded. The shift-inputs are executed in parallel using $m \times n$ clocks, which is equal to the number of the spins. Then in the "Update Spin" phase, the spin update is repeated for N times. The spin update takes one clock cycle, and thus this phase uses N clocks in total. After the update phase, all spin values are read out via the scan chain. This operation also needs $m \times n$ clocks, which is the same as the input operation.

In addition to the mode control, the controller also controls the spin update signals `upd_ij` and the flip signals `fp_ij` to emulate the annealing behavior during the "Update Spin" phase. These operations are performed by "Update controller" and "Flip controller" modules, respectively.

3.2 Methods for Improving Area-Efficiency

In order to simulate the probabilistic annealing behavior of the Ising model, random numbers are required for two purposes: to determine of the spin value when the interactions are "balanced," and to realize the random flip to increase energy temporally to avoid local optima. However, random number generators (RNGs) consume large hardware resources having significant impact on the area efficiency. In this work, the following two novel annealing methods without using RNGs are proposed.

3.2.1 RNG-Less Determination of Balanced Spins

The spin determination module `Spin_Det` calculates Eq. (4) to determine a spin value. If the interactions are balanced, i.e., `sum` happened to be 0, the spin can take either direction. Ideally, the spin value should be randomly determined to avoid spatial or temporal bias. Instead, we propose a simple flipping operation that approximates the random behavior without using RNGs:

$$\begin{cases} \sigma_i \leftarrow -\sigma_i & \text{if } \mathtt{sum} = 0, \\ \sigma_i \leftarrow \mathrm{argmin}\, H_i(\sigma_i) & \text{otherwise.} \end{cases} \tag{5}$$

With this scheme, the spin is flipped whenever the interactions are balanced. Otherwise, a spin value that gives lower local energy is chosen.

3.2.2 Shift-Register-Based Spin Flipper

To realize the RF on the annealing processor, the circuit that satisfies the following properties has to be considered: 1) a random pulse generator whose 0/1 probability is controllable according to the annealing temperature, and 2) a mechanism to deliver the generated random pulse to the flip signals `fp_ij` for each Ising cell. The simplest approach that satisfies the above requirements is to include an RNG and a temperature scheduler for each and every Ising cell [10]. However, it will result in an unacceptably large circuit area. Another idea is to deliver pre-calculated random sequences for all cells for every cycle, by giving up online random number generation. In this case, the distribution of the signal to all the Ising cells requires long time or requires a lot of wire resources.

In this work, we resolve these issues with the proposed shift-register-based spin flipper (SRSF). The SRSF consists of the registers `fp_ij` that are connected in series to form a long shift register as shown in Fig. 9, or we can subdivide registers into multiple shift registers. The SRSF works as follows:

1. Load an initial bit sequence $\boldsymbol{C}_{\mathrm{fp}} = \{\mathtt{fp\_11}, \mathtt{fp\_21}, \cdots, \mathtt{fp\_mn}\}$ into the shift register with $m \times n$ clocks.
2. During each annealing, apply k clocks to the shift registers and fill the vacant bits by zeros, i.e., $\boldsymbol{C}_{\mathrm{fp}}$ becomes $\{0, \cdots, 0, \mathtt{fp\_11}, \mathtt{fp\_21}, \cdots\}$.

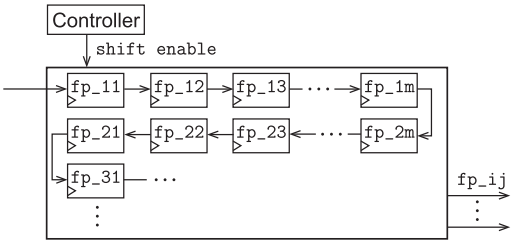

Fig. 9 Proposed shift-register-based spin flipper (SRSF). It consists of `fp_ij` registers connected in serial like a scan chain.

3. Repeat steps 1 and 2 until $\boldsymbol{C}_{\mathrm{fp}}$ becomes all zeros.

The initial bit sequence $\boldsymbol{C}_{\mathrm{fp}}$ is initialized randomly so that 0 and 1 are approximately equally contained. The random numbers here can be generated offline and in advance using any algorithms. The above shift operation reduces the number of 1s in $\boldsymbol{C}_{\mathrm{fp}}$ (N_{RF} in Algorithm 1). By repeating the shifts, it reduces the number of 1s further, which corresponds to the gradual decrease of the annealing temperature. By adjusting the parameter k, the speed of the temperature lowering (α_n in Algorithm 1) can be controlled.

The advantage of the proposed method is the small area overhead by completely eliminate the RNGs. In addition, the initialization step (step 1) of the proposed method does not require additional clock cycles because it can be executed in parallel with the existing initializations for `J_ijk`, `h_ij`, and σ`_ij`. Moreover, the proposed method can help the Ising model from being stuck in the local optimum solutions having several regions of conflicting spin directions. The behavior of the SRSF causes spacial imbalance of the annealing temperature due to its shift operation, e.g., the upper area with low temperature and the lower with high temperature. Since this situation is similar to the physical phenomenon of crystalizing from the edge of the material, the proposed method is considered effective for avoiding the local optima caused by uniform global cooling.

4. Evaluation

In this section, the effect of eliminating RNGs is first evaluated, then the performance of the proposed annealing processor implemented on an FPGA is evaluated. The max-cut benchmark problems used for the evaluation of the proposed processor are listed in Table 1. The problems G11, G12, G13, G32, G33, and G34 are taken from G-Set benchmark [12], while the others, Gx_y, ($x \in \{11, 12, 13\}$, $y \in \{1, 2, 3\}$), are generated by "rudy" [16], which is a generator program of G-Set. Gx_y has the same graph structure as Gx, but they are different in edge weights.

4.1 Effectiveness of SRSF

4.1.1 RNG-Less Determination of Balanced Spins

In order to verify the effectiveness of the proposed RNG-less determination of balanced spins, the solution accuracy

Table 1 List of max-cut problems used in evaluation

Problem	Structure	Nodes	(H×W)	Edges	Weight
G11	2D-grid torus	800	(8×100)	1600	$\{1, -1\}$
G11_1	2D-grid torus	800	(8×100)	1600	$\{1, -1\}$
G11_2	2D-grid torus	800	(8×100)	1600	$\{1, -1\}$
G11_3	2D-grid torus	800	(8×100)	1600	$\{1\}$
G12	2D-grid torus	800	(16×50)	1600	$\{1, -1\}$
G12_1	2D-grid torus	800	(16×50)	1600	$\{1, -1\}$
G12_2	2D-grid torus	800	(16×50)	1600	$\{1, -1\}$
G12_3	2D-grid torus	800	(16×50)	1600	$\{1\}$
G13	2D-grid torus	800	(32×25)	1600	$\{1, -1\}$
G13_1	2D-grid torus	800	(32×25)	1600	$\{1, -1\}$
G13_2	2D-grid torus	800	(32×25)	1600	$\{1, -1\}$
G13_3	2D-grid torus	800	(32×25)	1600	$\{1\}$
G32	2D-grid torus	2000	(20×100)	4000	$\{1, -1\}$
G33	2D-grid torus	2000	(25×80)	4000	$\{1, -1\}$
G34	2D-grid torus	2000	(40×50)	4000	$\{1, -1\}$

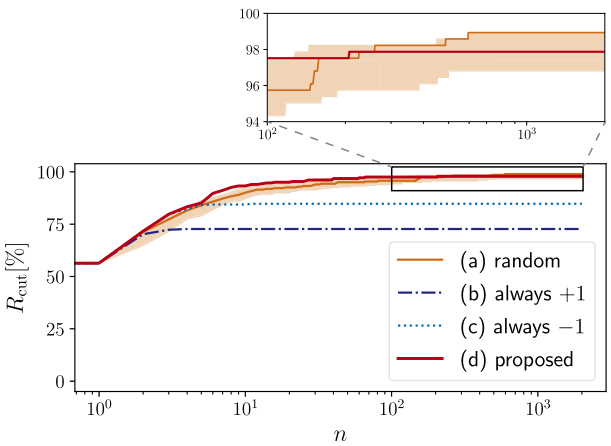

Fig. 10 Development of solution accuracy of G11 as a function of iterations n. The proposed method (d) shows the comparable accuracy to that of the random method (a).

is compared with the following methods:

(a) $\sigma_i \leftarrow \{+1, -1\}$ (random)
(b) $\sigma_i \leftarrow +1$ (always +1)
(c) $\sigma_i \leftarrow -1$ (always −1)
(d) $\sigma_i \leftarrow -\sigma_i$ (proposed)

In this evaluation, a software-implemented Ising solver is used. The parameters in the spin update (Algorithm 1) are set as follows:

- Initial spin values σ_i: all 1
- Maximum number of iterations N: 3000 (G32, G33, G34), 2000 (otherwise)
- Random flips: disabled ($N_{\mathrm{RF}} = 0$)

The checkerboard-like update, in which black and white cells are alternatively updated, is adopted in this experiment.

Figure 10 shows the solution accuracy of the four methods (a)–(d) as the function of iterations when solving G11. The solution accuracy is evaluated by R_{cut}, which is defined as a percentage of the obtained cut to the exact solution. Since the method (a) uses random numbers, the upper and the lower bounds of the results in 20 runs are shown as a

Table 2 Solution accuracies R_{cut} obtained by the different spin-determination methods [%]

Problem	(a) random	(b) +1	(c) −1	(d) proposed
G11	**98.21**	72.70	84.75	97.87
G11_1	**97.99**	71.74	81.16	97.83
G11_2	**98.00**	70.69	84.14	97.93
G11_3	100	100	100	100
G12	**99.26**	67.99	82.01	97.48
G12_1	97.53	70.99	83.96	**97.61**
G12_2	**97.46**	71.67	79.86	97.27
G12_3	100	100	100	100
G13	97.58	72.85	80.76	**97.94**
G13_1	97.27	75.87	85.31	**97.90**
G13_2	**97.15**	68.68	81.14	**97.15**
G13_3	100	100	100	100
G32	**97.45**	74.33	81.84	97.02
G33	97.55	72.21	83.36	**97.68**
G34	**97.59**	70.09	81.36	97.40

shaded region, and the run that achieved the best result is shown using a line. The proposed method (d) obtains R_{cut} of 97.9% accuracy, which is comparable to a 99.3% obtained by method (a) that uses random numbers. Table 2 shows the accuracies of final solutions when the problems in Table 1 are solved. The best result for a problem is shown in bold font. Here, all the values of (a) are the mean of 20 runs as (a) includes random process, while those of (b), (c), and (d) are the result of one run as those methods are deterministic. The proposed method (d) obtained equal or very close solution to the random flipping. There is no noticeable difference in the solution accuracies between the problems of different sizes. Hence, the proposed method is considered effective for solving the max-cut problems evaluated. The methods (b) and (c), which use a fixed spin value, could only obtain worse solutions. Our spin determination method works well without using resource-consuming RNGs.

4.1.2 Shift-Register-Based Spin Flipper

In order to verify the effectiveness of the SRSF, solution accuracy is compared with the conventional flipping method that uses a random number generator. A software-implemented Ising solver and a logic simulator of Verilog HDL are used for the conventional and the proposed method, respectively. The parameters in the spin update are set as follows:

- Initial spin values σ_i: random
- Maximum number of iterations N: 3000 (G32, G33, G34), 2000 (otherwise)
- Random flip: K = 0.5× (spin size) and α_n = 0.996 (G32, G33, G34), α_n = 0.993 (otherwise) for the conventional method, and decrement N_{RF} by 1 for each iteration for the proposed method. α_n is determined so that N_{RF} becomes 0 at about the same time as the proposed method.

In both methods, the checkerboard-like update is adopted, and the optimizations are run for 100 times to obtain averaged performance. Figure 11 compares how the solution

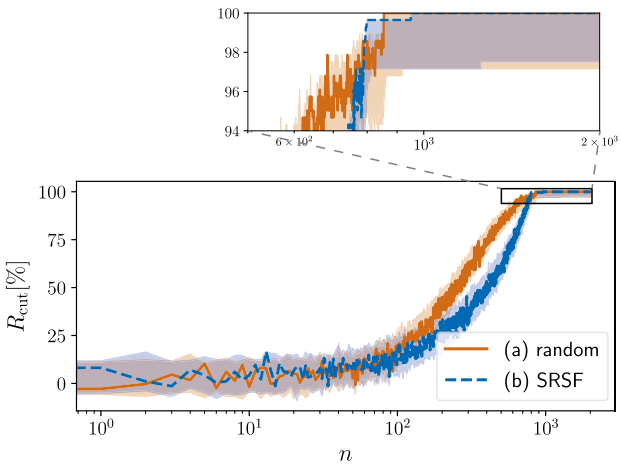

Fig. 11 Development of solution accuracy of G11 as a function of iterations n. The proposed SRSF (b) shows the comparable accuracy to that of the random method (a).

Table 3 Solution accuracies R_{cut} of spin-flipping methods [%]

Problem	(a) random		(b) SRSF	
	mean	max	mean	max
G11	**98.73**	100	98.48	100
G11_1	**98.79**	100	98.44	100
G11_2	**98.68**	**99.66**	98.16	99.31
G11_3	98.69	100	**99.99**	100
G12	**98.75**	**100**	98.68	99.64
G12_1	**98.36**	**99.66**	97.89	98.98
G12_2	**98.46**	**100**	97.98	99.32
G12_3	98.74	100	**100**	100
G13	**98.46**	**99.66**	97.10	98.63
G13_1	**98.44**	**99.65**	97.55	99.30
G13_2	**98.50**	**100**	97.33	98.93
G13_3	**99.74**	100	99.60	100
G32	**98.64**	**99.29**	98.16	99.01
G33	95.70	**99.42**	**97.30**	98.41
G34	**98.79**	99.28	98.46	**99.42**

accruracies changed as a function of iteration count. Since the initial spin values are randomly determined, the upper and the lower bounds of the results are shown as a shaded region, and the run that achieved the best result is shown using a line. Both methods obtain optimal solution almost at the same iteration counts. Table 3 shows the mean and the best accuracies, in which bold font indicates better results between the corresponding columns. Again, the proposed SRSF achieved very close solution accuracy to the conventional method. By using proposed SRSF, good optimization results are expected without using resource-consuming RNGs.

4.2 Evaluation of Hardware-Implemented Annealing Processor

The proposed annealing processor is implemented on an FPGA to evaluate its circuit area and speed to solve max-cut problems. The spin value σ is expressed using 1 bit (0 for "+1" and 1 for "−1") and both the interaction weight J and bias h are represented by using 2 bits so that it can take three values $\{-1, 0, +1\}$.

Table 4 FPGA resource usage of the annealing processor with 800 spins (G11)

Module	Slices		LUTs		Registers	
Cell array	10,975	(21.2%)	19,190	(9.3%)	5,600	(2.7%)
Ising cell (avg.)	13.7		24.0		7.0	
Controller	229	(0.4%)	80	(0.0%)	848	(0.4%)
Flip controller	202	(0.4%)	2	(0.0%)	801	(0.4%)
Others	27	(0.0%)	78	(0.0%)	47	(0.0%)
Overall	11,204	(21.3%)	19,270	(9.3%)	6448	(3.1%)

Table 5 FPGA resource usage of the annealing processor with 2000 spins (G32)

Module	Slices		LUTs		Registers	
Cell array	34,185	(65.94%)	52,010	(25.08%)	14,000	(6.75%)
Ising cell (avg.)	17.1		26.0		7.0	
Controller	527	(1.0%)	79	(0.0%)	2046	(1.0%)
Flip controller	502	(1.0%)	3	(0.0%)	2001	(1.0%)
Others	25	(0.0%)	76	(0.0%)	45	(0.0%)
Overall	34,712	(67.0%)	52,089	(25.1%)	16,046	(7.7%)

4.2.1 Circuit Area and Maximum Delay

The proposed annealing processor was written in Verilog HDL and synthesized/implemented by Xilinx ISE Design Suite 14.7 for a target FPGA, Xilinx Virtex5 XC5VLX330T. Tables 4 and 5 show the breakdown of the resource usage after synthesis for the problems of G11 with 800 nodes and for G32 with 2000 nodes, respectively. Synthesis results for the other problems are omitted since the resource usages are determined by the number of nodes. The percentages to the total resources are also shown in the parentheses. The maximum delays of the designs for all the problems are less than 5.0 ns, meaning that the proposed processor can operate at 200 MHz. As shown in Table 4, the proposed SRSF used 202 slices, which is much smaller than the array of the Ising cells. By eliminating the RNGs, the area efficiency of the annealing processor has been greatly improved. According to Tables 4 and 5, the resource usage of slices and LUTs increases linearly to the number of nodes, indicating that the proposed annealing processor is scalable as long as the maximum number of edges is bounded to a small number.

4.2.2 Speed and Accuracy

In order to evaluate the efficiency of solving optimization problem on an FPGA, the synthesized netlist is implemented on the same FPGA board, and a PC with Intel Core i7-6700K @4.00GHz is used as a host. The FPGA board and the PC are connected via PCI Express, and the input/output data are transferred by using a DDR2 memory on the FPGA board. A control program on the host PC written in C language controls the following execution flow:

(a) PC converts a problem to the Ising model (Mapping)
(b) PC writes data to DDR2 memory on FPGA board
(c) FPGA executes annealing of the Ising model
(d) PC reads data from DDR2 memory on FPGA board

The process (c) includes all the operations on the FPGA, which are data input/output between the DDR2 memory and the FPGA and the spin update operation as shown in Fig. 8. The execution time of each step to solve G11 and G32 is summarized in Table 6. Numbers of iterations in the spin update are $N = 2000$ for G11 and $N = 3000$ for G32. We omit the runtime of the other problems because their runtimes are completely equal to those with the same number of spins. The processing time to solve a single problem with the proposed annealing processor only takes about 1 ms in total, including communications between PC.

Table 6 Runtime of the proposed annealing processor for solving max-cut problems (G11 and G32)

Process	Time (G11)	Time (G32)
(a) Mapping on PC	690 μs	980 μs
(b) Write data from PC to FPGA board	80 μs	80 μs
(c) Annealing on FPGA	80 μs	100 μs
(d) Read data from FPGA board to PC	70 μs	70 μs
Total	0.92 ms	1.23 ms

We also compare the execution time with other max-cut solvers: (A) SW Ising: software-implemented Ising model solver, (B) CPLEX: mixed-integer programming (MIP) solver [17], and (C) SG3: existing fastest heuristic algorithm for max-cut solver [1]. Here, (A) is the same as the software Ising solver in Sect. 4.1.2. For the experiments, SW Ising and SG3 were run on a PC with Intel Xeon E5-1650 @3.5 GHz, and CPLEX was run using 32 threads on a PC with Intel Xeon E5 @2.6 GHz. SW Ising and SG3 were written in C++ language.

Figures 12, 13, and Table 7 show the comparison of the solving time by the proposed FPGA-based annealing processor (D) and the other software solvers (A)–(C). Note that t in Table 7 means runtime of the algorithms and t_{eq} is the time required for CPLEX to obtain a solution with an equal accuracy to that of HW Ising. As shown in Fig. 12, HW Ising by the proposed processor is 2.0×10^2 times faster than SW Ising to obtain the solution. When compared with SG3, which achieved 96.0% solution and takes 1.0 s, HW Ising is still faster and obtained solutions with better accuracy. On

Table 7 Solution accuracy R_{cut} and runtime of four algorithms

Problem	(A) SW Ising		(B) CPLEX			(C) SG3		(D) HW Ising	
	R_{cut} [%]	t [s]	R_{cut} [%]	t [s]	t_{eq} [s]	R_{cut} [%]	t [s]	R_{cut} [%]	t [ms]
G11	98.73	0.18	100	10.27	6.66	95.92	1.01	98.48	0.92
G11_1	98.79	0.19	100	6.14	5.59	95.83	1.27	98.44	0.92
G11_2	98.68	0.18	100	9.11	5.41	94.66	1.21	98.16	0.92
G11_3	98.69	0.18	100	0.03	0.03	99.94	1.49	99.99	0.92
G12	98.75	0.19	100	11.30	10.48	95.86	1.06	98.68	0.92
G12_1	98.36	0.19	100	8.65	7.41	95.05	1.23	97.89	0.92
G12_2	98.46	0.18	100	9.37	7.48	92.32	1.22	97.98	0.92
G12_3	98.74	0.19	100	0.03	0.03	99.94	1.44	100	0.92
G13	98.46	0.15	100	14.62	9.01	96.05	1.13	97.10	0.92
G13_1	98.44	0.16	100	8.64	6.93	94.58	1.34	97.55	0.92
G13_2	98.50	0.15	100	7.92	7.09	94.13	1.34	97.33	0.92
G13_3	99.74	0.15	100	3.86	0.03	99.94	1.39	99.60	0.92
G32	98.64	0.73	100	41.97	33.74	94.68	16.46	98.16	1.23
G33	95.70	0.61	100	72.89	46.40	93.85	16.78	97.30	1.23
G34	98.79	0.73	100	76.75	48.06	95.01	17.25	98.46	1.23

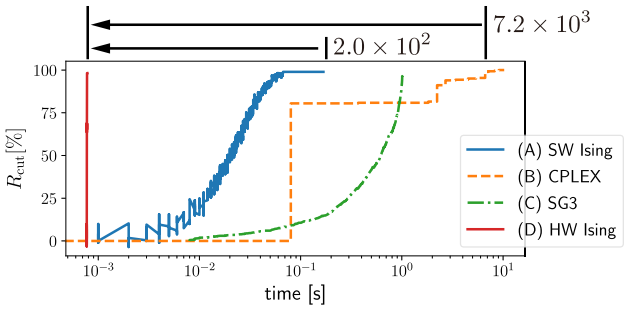

Fig. 12 Solution accuracy of G11 as a function of runtime.

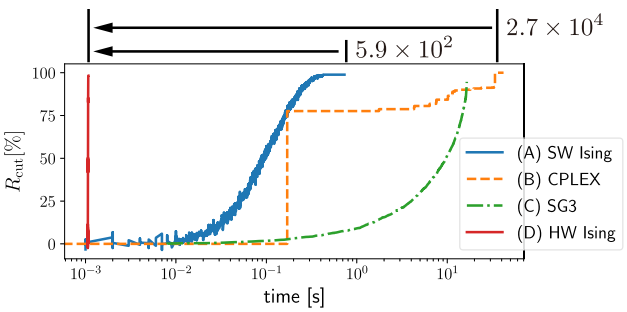

Fig. 13 Solution accuracy of G32 as a function of runtime.

the other hand, CPLEX obtained the optimal solution but it took much longer time to the other solvers. When the calculation time of CPLEX is compared to the HW Ising at a same accuracy, HW Ising is faster than CPLEX by 7.2×10^3. Figure 13 also shows similar trend. HW Ising is 5.9×10^2 times faster than SW Ising and 2.7×10^4 times faster than CPLEX. The results are similar for the other problems except G11_3, G12_3, G13_3, for which, however, our HW Ising is still 10 times faster. This result shows that the proposed architecture is suitable to obtain quasi optimal solution with a very short time.

5. Conclusion

This paper proposed an area-efficient highly parallel annealing processor that utilizes no random number generator. The proposed processor is based on the Ising model, and fully synthesizable, scalable, and configurable for the problems to solve. In order to eliminate random number generators, a spin flipping scheme for balanced spin and a shift-register-based spin flipping scheme in the annealing are introduced. Through an implementation of the proposed architecture on an FPGA and evaluation solving max-cut problems, our annealing processor achieved 10–10^4 times speedup compared with conventional optimization algorithms to obtain the solution of equal accuracy.

Acknowledgments

This work was partially supported by JSPS KAKENHI Grant No. 26280014 and 17H01713. This work was also supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Mentor Graphics, Inc.

References

[1] S. Kahruman, E. Kolotoglu, S. Butenko, and I.V. Hicks, "On greedy construction heuristics for the MAX-CUT problem," International Journal of Computational Science and Engineering, vol.3, no.3, pp.211–218, 2007.
[2] G.A. Kochenberger, J.-K. Hao, Z. Lü, H. Wang, and F. Glover, "Solving large scale max cut problems via tabu search," Journal of Heuristics, vol.19, no.4, pp.565–571, 2013.
[3] L. Grippo, L. Palagi, M. Piacentini, V. Piccialli, and G. Rinaldi, "SpeeDP: an algorithm to compute SDP bounds for very large max-cut instances," Mathematical Programming, vol.136, no.2, pp.353–373, 2012.
[4] H. Nishimori, Statistical Physics of Spin Glasses and Information Processing, Oxford University Press, 2001.
[5] D-Wave. http://www.dwavesys.com/.
[6] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse ising model," Physical Review E, vol.58, no.5, pp.5355–5363, 1998.
[7] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, "A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem," Science, vol.292, no.5516, pp.472–475, 2001.

[8] V.S. Denchev, S. Boixo, S.V. Isakov, N. Ding, R. Babbush, V. Smelyanskiy, J. Martinis, and H. Neven, "What is the computational value of finite-range tunneling?," Physical Review X, vol.6, no.3, 2016.
[9] M. Yamaoka, C. Yoshimura, M. Hayashi, T. Okuyama, H. Aoki, and H. Mizuno, "20k-spin ising chip for combinational optimization problem with CMOS annealing," Digest of Technical Papers of IEEE International Solid-State Circuits Conference (ISSCC), p.24.3, 2015.
[10] 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.
[11] K. Someya, R. Ono, and T. Kawahara, "Novel ising model using dimension-control for high-speed solver for ising machines," Proceedings of 14th IEEE International New Circuits and Systems Conference (NEWCAS), 2016.
[12] G-Set. http://web.stanford.edu/˜yyye/yyye/Gset/.
[13] A. Lucas, "Ising formulations of many NP problems," Frontiers in Physics, vol.2, no.5, pp.1–15, 2014.
[14] A. Zaribafiyan, D.J.J. Marchand, and S.S.C. Rezaei, "Systematic and deterministic graph minor embedding for cartesian products of graphs," Quantum Information Processing, vol.16, no.5, 2017.
[15] J. Cai, W.G. Macready, and A. Roy, "A practical heuristic for finding graph minors," 2014. arXiv:1406.2741.
[16] C. Helmberg and F. Rendl, "A spectral bundle method for semidefinite programming," SIAM Journal on Optimization, vol.10, no.3, pp.673–696, Jan. 2000.
[17] IBM, "IBM ILOG CPLEX Optimization Studio V12.6.2 documentation."

Hidenori Gyoten recieved B.E. degree in Electrical and Electronic Engineering from Kyoto University in 2016. He is a master course student at Department of Communications and Computer Engineering, Kyoto University.

Masayuki Hiromoto received B.E. degree in Electrical and Electronic Engineering and M.Sc. and Ph.D. degrees in Communications and Computer Engineering from Kyoto University in 2006, 2007, and 2009 respectively. He was a JSPS research fellow from 2009 to 2010, and with Panasonic Corp. from 2010 to 2013. In 2013, he joined the Graduate School of Informatics, Kyoto University, where he is currently a senior lecturer. His research interests include VLSI design methodology, image processing and pattern recognition. He is a member of IEEE and IPSJ.

Takashi Sato received B.E. and M.E. degrees from Waseda University, Tokyo, Japan, and a Ph.D. degree from Kyoto University, Kyoto, Japan. He was with Hitachi, Ltd., Tokyo, Japan, from 1991 to 2003, with Renesas Technology Corp., Tokyo, Japan, from 2003 to 2006, and with the Tokyo Institute of Technology, Yokohama, Japan. In 2009, he joined the Graduate School of Informatics, Kyoto University, Kyoto, Japan, where he is currently a professor. He was a visiting industrial fellow at the University of California, Berkeley, from 1998 to 1999. His research interests include CAD for nanometer-scale LSI design, fabrication-aware design methodology, and performance optimization for variation tolerance. Dr. Sato is a member of the IEEE and the Institute of Electronics, Information and Communication Engineers (IEICE). He received the Beatrice Winner Award at ISSCC 2000 and the Best Paper Award at ISQED 2003.