Research on Annealing Processors for Large-Scale Combinatorial Optimization Problems

Kasho Yamamoto

A thesis presented for the degree of
Doctor of Engineering

Graduate School of Information Science and Technology
Hokkaido University
Japan

March, 2020

Research on Annealing Processors for Large-Scale Combinatorial Optimization Problems

Kasho Yamamoto

Abstract

This research is concerned with an annealing processor for efficiently solving a combinatorial optimization problem.

In order to realize a smart society, optimization of people and things such as transportation networks, power networks, human resource allocation and routes is indispensable. Many of these problems are called combinatorial optimization problems and are known as very important problems in modern society. However, it is known that the combinatorial optimization problem is NP-hard and it is difficult to efficiently solve it with a conventional Neumann computer.

In recent years, the miniaturization of the process of manufacturing computer chips is approaching the physical limit. Therefore, especially in the solution by the full search, the improvement in speed due to the calculation speed of the computer cannot be expected. From such a background, dedicated hardware capable of solving a combination optimization problem at high speed and with low power consumption has been attracting attention.

The annealing processor is a general-purpose combination optimization solver that utilizes the property that the ground state of the optimization problem expressed by using the Ising model and the optimal solution of the combination optimization problem match. The Ising model is a ferromagnetic model in statistical physics expressed by spins having an upward and downward state, spin-spin interactions occurring between the spins, and an external magnetic field acting on each spin. The Ising model can express a combinatorial optimization problem by expressing a control target and a cost using these parameters. The annealing processor updates the spin state while

slowly lowering the temperature parameters based on simulated annealing, which is an optimization method learned from natural phenomena generally called annealing. Eventually, the spin will reach the ground state with a high probability and obtain the optimal solution.

Annealing processors are classified into two types, the nearest neighbor type and the fully connected type, depending on the shape of the Ising model simulated on the hardware. In the nearest neighbor type, the coupling between spins is limited to only between adjacent spins, and in the full coupling type, coupling exists between arbitrary spins. The nearest neighbor type realizes high parallel processing by reducing the data spin required by simulated annealing by reducing the proximity spin. In addition, since data transfer is limited to adjacent spins, the scalability is high. However, since the spin-to-spin interaction is sparse, there is a problem that the class of combinatorial optimization problems that can be dealt with is limited, or that the problem requires deformation processing to match the Ising model of hardware. The fully connected type can solve any class of problems as long as the number of spins permits, however it requires connections between arbitrary spins. Therefore, scalability is low. Because there is a dependency on all spins, the number of spins that can be updated at one time is always limited to only one regardless of the total number of spins. Therefore, it takes time to converge to the ground state.

In this research, we propose a method of increasing the number of spins for this annealing processor by using (1) a spatially-expanded time-division processing mechanism architecture based on a loosely-coupled type, (2) An algorithm for applying a fully coupled Ising network to a more complex Ising network using the nearest neighbor Ising network and an annealing method that can be updated in parallel, (3) We conducted a study on the architecture of a fully coupled annealing processor for parallel updating annealing.

As described above, the research described in this paper contributes to the improvement of the efficiency of the annealing processor for solving the large-scale combinatorial optimization problem from both the fundamental algorithm and the hardware architecture. I hope that this research will solve the combinatorial optimization problem that is expected to increase in demand in the future and contribute to the development of human society.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

1.1.1 Data Mining and Optimization Problem

In order to overcome the problems that exist in modern society such as energy problems and labor shortage, it is necessary to minimize costs and waste while efficiently utilizing limited physical and human resources in sociery. In addition, with the advent of the IoT era in recent years, an environment capable of storing and acquiring an enormous amount of data has been constructed. Such accumulated data is called big data. A method of observing trends in big data (data mining, frequent itemset mining) and optimizing and improving the target based on the observation results (combination optimization problem) has emerged as an important academic issue in modern society.

Frequent itemset mining (FIsM) is an important and fundamental problem in data mining with regards to association and correlation. FIsM finds frequent itemsets by counting the occurrence of subgroups called itemsets in a transactional database. It is possible to know the relationship between items from frequent itemsets because items contained in frequent itemsets have a high probability of occurring at the same time. The knowledge gained from this technique is widely used in data analytics, such as in the analysis of sensor networks, network traffic, stock markets, and fraud detection.

Optimization is a method of maximizing (minimizing) the objective function represented by profit (cost) under given constraints. Combinatorial optimization is one type of such optimization, and finds an optimal combination from a large number of alternatives.

In our lives, we take actions that increase our own interests and satisfaction under the constraints of time and physical strength. The choice is made consciously or unconsciously, which is also the result of solving the combinatorial optimization problem. Such problems exist not only in our immediate surroundings but also in society. For example, manufacturing also supports our daily life base, even if we are not directly visible, such as order planning, searching for the shortest logistics route, staffing, and energy supply.

However, it is known that many of the very important combination optimization problems are NP-hard, and these problems are difficult to efficiently solve with a conventional computer in the theory of computational complexity.

One of the methods for solving the optimization problem is an enumeration method. This is a solution method that considers all possible combinations and finds the optimal solution from among them. Although it is possible to find a global optimal solution by this method, the number of combinations increases exponentially. If the number of elements is N, there are 2^N combinations. Further, the computation time explosively increases, and if an attempt is made to solve a real problem using enumeration, the computation may not be completed in a realistic time even with a supercomputer.

1.1.2 Post Moore and Demand of ASIC

With the advent of the IoT era, the amount of real-time data that is processed in data centers has increased explosively. As a result, stream mining, extracting useful knowledge from a huge amount of data in real time, is attracting more and more attention. It is said, however, that real-time stream processing will become more difficult in the near future, because the performance of processing applications continues to increase at a rate of $10-15\%$ each year, while the amount of data to be processed is increasing exponentially [1]. Therefore, data mining and solving NP-hard problems with

conventional computers requires considerable time and energy.

In order to overcome such bottlenecks in standard computers for combinatorial optimization, alternative computing mechanisms [3] [5] are aggressively pursued. One such hardware that is already in wide use is the Fast Fourier Transform (FFT) hardware accelerator [2]. Another example is that dedicated circuits mounted on smartphones are increasing year by year.

Compared to general-purpose processors that have evolved to handle various applications in general, dedicated hardware (ASICs) dedicated to certain applications can process at higher speeds and lower speeds at the expense of versatility. It can be executed with power consumption. On the other hand, however, developing hardware dedicated to a specific application has a disadvantage that it costs more in terms of time and money than developing an application of a general-purpose processor (with software). Once a circuit is created, ASICs cannot be changed later and cannot be modified in the event of a design error. In addition, it is not cost-effective to implement a dedicated circuit for each problem. In addition, the ASIC takes a huge amount of time since it has to solve physical problems such as chip layout after deciding the logical operation. On the other hand, FPGAs use chips that have already been made, so physical design does not take much time. Therefore, when solving the combination optimization problem with a dedicated circuit, it is not realistic to make hardware for each problem. In order to solve such a problem, there are two existing approaches.

One is a hardware called Field Programmable Gate Array [4], which consists of a Look Up Table that reproduces all logic circuits, and a switch matrix that allows them to be connected and changed arbitrarily. By preparing a circuit for each problem, it is possible to solve the problem. The FPGA can be rewritten many times, but due to its redundancy, the area of the chip (substantially equivalent to the number of constituent elements) is larger than that of an application specific integrated circuit (ASIC). Therefore, power consumption increases. On the other hand, compared to CPUs, FPGAs have a larger circuit area and therefore lower operating frequencies. However, it is said that FPGAs may have higher processing power than CPUs. This is because the FPGA can configure a circuit in a form most suitable for the intended operation.

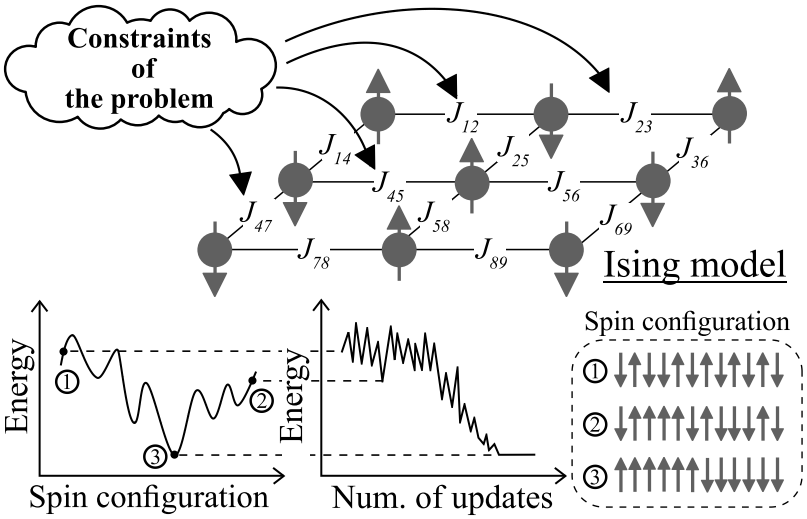


Figure 1.1: Ising model

1.1.3 Annealing Processor

Instead of solving combinatorial optimization problems for each problem, another method uses the dedicated circuit that searches the state of the minimum energy of one physical model and use the fact that ground state of each problem embeded into this physical model is equivalent to optimized solution. Various types of combinatorial optimization problems can be processed by creating one dedicated circuit.

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. 1.1. The annealing processor, based on the Ising model and "annaling', is one approach to natural computing. The annealing processor exploits a combinatorial optimization method called simulated annealing.

A conventional computer called a von Neumann computer is mainly composed of a storage unit, an operation unit, a control unit, and an I/O unit. It solves problems by storing data and calculation procedures in a storage unit and executing them sequentially. On the other hand, a computing architecture called the annealing processor 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 calculation based on this natural phenomena enables more efficient solution searching for many problems than searching for an exponentially expanding solution space one by one.

1.1.4 Subject of This Study

It is not hard to imagine that the data to be processed will increase explosively as the technology for the IoT, the Internet of Everything (IoE) and the smart society develops in the future. Also, as the types of devices connected to the Internet increase, it is considered important to perform data observation and optimization by associating data between different devices.

In frequent itemset mining, high-speed processing has been achieved in the past research [6] on the increase of data types. On the other hand, in the combinatorial optimization problem, an increase in the number of data types not only complicates the problem itself but also increases the number of parameters. There is no approach that can solve the combinatorial optimization problem efficiently. Therefore, research on an annealing processor that solves the combinatorial optimization problem efficiently is important.

1.2 Organization

In the chapter 2, the class of the combinatorial optimization problem targeted in this study and its solution are explained. Chapter 3 summarizes related research on annealing processors and summarizes their characteristics. In Chapter 4, we propose a hardware architecture for an annealing processor that can search for the ground state of a large-scale loosely coupled Ising model. In Chapter 5, based on the loosely coupled Ising model, we propose an Ising model with more complex connections, and an architecture that can handle the fully coupled Ising model. For a complex combinatorial optimization problem, the conventional loosely coupled Ising model uses a combinatorial optimization problem with a network that is more complex than that model. Although it was necessary to perform graph transformation that causes a decrease in accuracy, this method makes it possible to solve

a model that is logically more complex than the Ising model on hardware by utilizing the architecture of the loosely coupled Ising model. Chapter 6 proposes a new annealing technology that can be updated in parallel and an architecture that handles the fully coupled Ising model. With this method, it is possible to efficiently process the fully coupled Ising model, which was a problem in principle with conventional annealing processors, and it is expected that it will be able to process more practical problems. In particular, regarding this architecture, not only the logic verification by simulation, but also a dedicated circuit was actually manufactured and evaluated.

Finally, in Chapter 7, we will summarize our work and discuss what should be done in the future.

Bibliography

[1] T. Skotnicki, J. A. Hutchby, T. J. King, H. 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.

[2] Yuehai Xu, Eitan Frachtenberg, Song Jiang, and Mike Palecezny. Very fast fourier transform algorithms hardware for implementation. *IEEE Transactions on Computers*, C-28:333–341, 1979.

[3] M. Brodowicz and T. Sterling, A non von neumann continuum computer architecture for scalability beyond Moore' s law in Proceedings of the ACM International Conference on Computing Frontiers, ser. CF ' 16. New York, NY, USA, ACM, 2016, pp. 335–338.

[4] F.Belletti,M.Cotallo,A.Cruz,L.A.Fernandez,A.Gordillo-Guerrero, M. Guidetti, A. Maiorano, F. Mantovani, E. Marinari, V. Martin-Mayor, A. Munoz-Sudupe, D. Navarro, G. Parisi, S. Perez-Gaviro, M. Rossi, J. J. Ruiz-Lorenzo, S. F. Schifano, D. Sciretti, A. Tarancon, R. Tripiccione, J. L. Velasco, D. Yllanes, and G. Zanier Janus: An FPGA-based system for high-performance scientific computing Computing in Science Engineering, vol. 11, no. 1, pp. 48– 58, Jan 2009.

[5] A. Vajda, Programming Many-Core Chips New York, NY, USA: Springer, 2011.

[6] K. Yamamoto, M. Ikebe, T. Asai, and M. Motomura FPGA-based Stream Processing for Frequent Itemset Mining with Incremental Multiple Hashes Circuits Syst., 7(10), 3299-3309, 2016.

Chapter 2

Combinatorial Optimization Problem

2.1 Definition of Optimization Problem

Combinatorial optimization is a kind of optimization. Optimization is to make an optimal selection under the some conditions for problem. The method of defining the request to be optimized by various formulas and finding the optimal answer by mathematical calculation is called mathematical optimization. A problem to be solved is represented by a mathematical expression, which is called a mathematical model, and creating this mathematical model is called formulation. Thus, it is necessary to formulate the problem to be solved as an optimization problem, and to describe how much the purpose required by the problem has been achieved. This is called an objective function, and the variables determined to achieve this objective are called decision variables. In the optimization problem, there are often various constraints in achieving the purpose. These are called constraints, which limit the domain of the decision variables. In summary, it can be said that an optimization problem is a problem of finding a variable x (optimal solution) that maximizes (minimizes) the value of an objective function among decision variables x satisfying constraints. When the value of the variable x takes a discrete value such as an integer, it is particularly called a combination optimization problem. Further, for a certain condition, a problem

of a format for outputting a judgment of acceptance or rejection (whether the condition is satisfied or not satisfied) for each input is called a decision problem.

When the objective function f is formulated under the variables $x = (x_1, x_2, ..., x_n)$ that satisfy the constraints, the optimization problem can be expressed as follows.

$$\begin{aligned} &\text{Object function : } f(x) \rightarrow \text{Maximum (minimum)} \\ &\text{Constraints : } x \in S \end{aligned}$$

Here, when the decision variable x takes a discrete value, that is, when $S \subseteq Z$, this problem is called a combinatorial optimization problem. Even if the objective function is maximized, the objective function can be changed to a problem that seeks the minimum by setting the objective function to $-f(x)$.

In an optimization problem, a decision variable x that satisfies the following conditions is called a global optimal solution.

$$f(x) \leq f(x'), \quad \forall x' \in S \tag{2.1}$$

If the neighborhood of x is expressed as $N(x)$ $subset S$, a decision variable satisfying the following conditions is called a local optimal solution.

$$f(x) \leq f(x'), \quad \forall x' \in N(x) \tag{2.2}$$

In the optimization, it is desirable to find a global optimal solution. However, as a practical problem, the formulation becomes complicated, and the calculation time exponentially increases due to an increase in the number of variables.

2.1.1 Classification by nature of problem

The optimization problem can be classified into the following three points.

1. Decision variables: continuous, discrete

2. Objective function: convex function, non-convex function

3. Objective function: linear, nonlinear

Continuity

When the decision variables take continuous values, they are called continuous optimization problems, and when they take discrete values, they are called combination optimization problems (discrete optimization). Among the combinatorial optimization problems, the problem in which the decision variables are integers is called an integer optimization problem, and in particular, if the decision variables are restricted to binary values of 0 and 1, the problem is called the 0-1 integer optimization problem. Called.

Convexity

Here, particularly, a classification method regarding convex and non-convex is considered. In the case of a continuous optimization problem, the convexity of the objective function can be defined for any $x, y \in R^n$, $0 \leq \alpha \leq 1$ by the following formula.

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y) \tag{2.3}$$

A function f that satisfies the above conditions is called a convex function. For any set $S \subseteq R^n$, if any $x, y \in S$, $0 \leq \alpha \leq 1$ satisfies the following conditions, S is called a convex set .

$$(\alpha x + (1-\alpha)y) \in S \tag{2.4}$$

In the discrete function, the following is satisfied for an arbitrary decision variable $x \in Z$.

$$2f(x) \leq f(x+1) + f(x-1), \quad \forall x \in Z \tag{2.5}$$

The biggest feature of the convex optimization problem defined above is that the global optimal solution and the local optimal solution described in the previous section match. Therefore, the problem of convexity is easier to handle than the problem of nonconvexity. When trying to find the optimal solution in the neighborhood $N(x) \subset S$ of x, the solution obtained may differ depending on convexity and nonconvexity.

Linearity

If the objective function is a linear function and the constraints are given by linear equations or inequalities, it is called a linear optimization problem, Otherwise, it is called a non-linear optimization problem.

2.1.2 Classification by calculation difficulty

Combinatorial optimization problems are also classified according to their difficulty. Here, the difficulty of a problem refers to the form of a function of the amount of computation, and such a class of problems is called a complexity class from the viewpoint of computational complexity theory.

Usually, even if the same algorithm is used, the amount of calculation often changes depending on the scale of the problem. Therefore, in order to solve any problem of that scale, it is fundamental to estimate the number of necessary operations in the worst case. The number of operations is indicated in order (O) notation.

When represented by a multiplier or a factorial by a functional system of this calculation amount, it is called an exponential time algorithm. When expressed by a constant K like N^K, it is called a polynomial time algorithm.

The exponential time algorithm is difficult to apply to real problems because the amount of calculation increases exponentially depending on the problem size. On the other hand, polynomial time is a more practical algorithm than the exponential time algorithm, but its application is difficult depending on its constant k.

Is a polynomial time algorithm (one that can be represented by $O(n^k)$ for a variable n and a constant k), that is, a set of relatively easy-to-solve problems is a class P. There are classes such as NP, NP complete, and NP difficult, including the class P.

The NP class, like the P class, is a class of problems that can be represented by a polynomial time algorithm, but the difference between the two is computation method. The P class is performed by a deterministic Turing machine, and the NP class is performed by a non-deterministic Turing machine. In a non-deterministic Turing machine, a calculation is performed in which an output value is not uniquely determined for an input. In the deter-

Class	Problem
P	Liniar Optimization Problem
	Hamiltonian Cycle Problem
	Shortest Path Problem
	Maximum Matching Problem
	Maximum Flow Problme
NP-Hard	Traveling Salesman Problem (Dicision Problem)
	Set Partitioning Problem (Dicision Problem)
	Knapsack Problem (Dicision Problem)
	Maximal Clique Problem (Dicision Problem)
	Minimal Vertex Cover Problem (Dicision Problem)
NP-Complete	Integer Optimization Problem
	Traveling Salesman Problem
	Set Partitioning Problem
	Maximal Clique Problem (Dicision Problem)
	Minimal Vertex Cover Problem

Table 2.1: Combinatorial optimization problems and their classification

ministic Turing machine, the calculation method is determined for the input, but in the non-deterministic Turing machine, there are multiple calculation methods (candidate answers) for the input. In a non-deterministic Turing machine, it is assumed that it has the ability to execute branches to multiple cases in parallel in an algorithm, and can execute multiple calculation processes simultaneously by repeating the branches.

By giving the parallelism to the calculation in this way, it is possible to perform the calculation in a polynomial time even if the calculation becomes enormous when performed by the deterministic Turing machine.

Thus, it can be seen that the capacity of a non-deterministic Turing machine is higher than that of determinism. Therefore, relationship between the calculation difficulty of the class P and the calculation difficulty of the class NP is $N \subseteq NP$.

Table 2.1 shows examples of specific combinatorial optimization problems and classification from the viewpoint of computational complexity.

2.2 Typical Optimization Problems and Conventional Approach

2.2.1 Max-Cut Problem

The max cut problem [1] finds two vertex groups that maximize the weight of the edge for each endpoint that belongs to a different group. Applications of this problem include noise removal of binary images. The obfective function is expressed as follows.

$$H = max\frac{1}{2}\sum_{i,j\in N, i<j} G_{ij}(1 - x_i x_j) \tag{2.6}$$

$x_i \in \{-1, 1\}$ is the group to which vertex i belongs, and G_{ij} represent the weight of edge ij. In conventional computers, this problem is often solved by a method called a greedy method [2]. In the greedy method, if the objective function score is improved by changing the decision variables of the current solution, it is accepted as the current solution. By repeating this process many times, the method finds an approximate solution.

2.2.2 N-Quenn Problem

The N-Queen problem [3] is to find patterns that N queens can not attack each other on the $N \times N$ chessboard. That is, another queen cannot exist in the same column, row, or diagonal where one queen exists. In a case where this problem is solved by a conventional computer, a queen arrangement that satisfies the above constraint is found by exhaustive search method.

2.2.3 Travelling Salesman Problem

The traveling salesman problem is a problem in which, when a salesman departs from a certain city, visits all the cities and returns to the starting point, finds the order in which the shortest route to go around the city is reached. The traveling salesman problem can be applied to delivery planning problems and VLSI design. When turning around a n city, there are $\frac{(n-1)!}{2}$ streets around the city considering the reverse direction. Even if you try

to solve the problem by brute force, the combination becomes more and more exponential (factorial) as the number of cities increases, so it becomes difficult. In other words, the traveling salesman problem is a NP difficult class problem, and it is difficult to find a deterministic polynomial-time algorithm for the size of a problem example.

2.3 Summary

There are many combinatorial optimization problems in human society, and techniques for solving such problems are indispensable in terms of advanced decision-making and control of people and objects. These problems can be categorized in terms of decision variables, features of the objective function, and computational complexity. In particular, there are classes that are difficult to solve with conventional computers in terms of computational complexity. In this paper, the following chapters describe research on dedicated hardware for solving combinatorial optimization problems involving such classes.

Bibliography

[1] E. Boros and P.L. Hammer, "The max-cut problem and quadratic 0-1 optimization; polyhedral aspects, relaxations and bounds," Annals of Operations Research 33 151 (1991).

[2] S. Kahruman, E. Kolotoglu, S. Butenko, and I. V. Hicks "On Greedy Construction Heuristics for the MAX-CUT problem," Int. J. of Computational Science and Engineering

[3] J. Bell and B. Stevens, "A survey of known results and research areas for n-queens," Discrete Mathematics, Volume 309, Issue 1, 2009, Pages 1-31,

[4] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys The Traveling Salesman Problem : A Guided Tour of Combinatorial Optimization 1st ed. Wiley, 9 1985

Chapter 3

Annealing Processor

3.1 Introduction

In this chapter, two problems, "the slowdown of the performance improvement of the conventional computer" and "the conventional computer is not suitable for solving the combination optimization problem", which are problems when solving the combination optimization problem with the conventional computer, are presented. The explanation about the new computing technology which solves is described. A conventional von Neumann computer including a calculation unit, a storage unit, and an input / output unit solves a desired problem by writing (programming) a command for data by a user. On the other hand, the annealing processor described in this chapter associates a natural phenomenon model called natural computing with the problem to be solved (equivalent to conventional programming). Then, by inputting a problem to be solved into a processor simulating the behavior of nature, the processor automatically outputs a solution according to a natural phenomenon. The annealing processor uses the Ising model as a model and annealing as a natural phenomenon, The optimal solution (ground state of the Ising model) of the combinatorial optimization problem converted to the Ising model is obtained efficiently by using an annealing processor. The following describes the annealing processor and its related research.

3.2 Annealing Processors

3.2.1 Ising Model

The Ising model is when a huge number of elements interact with each other and each element is given an external force. This is a model that expresses the behavior as a whole. This is a model where the spins are arranged in a grid as shown in ref fig: ising. This is a very simple model in statistical mechanics. An Ising model is defined on an undirected graph. In the undirected graph $G = (V, E)$, V indicates a vertex and E indicates a set of edges. In the Ising model, each vertex has a spin state at each vertex, and each side has an interaction coefficient $J_{ij}(i \neq j, 0 \leq i \leq n, 0 \leq j \leq n)$. Assuming that the spin state of the vertex $v_i \in V$ is σ_i, there are two upward and downward states $\sigma_i = \pm 1$. Regarding this J_{ij}, if $J_{ij} > 0$, it shows a ferromagnetic interaction and is stable when adjacent spins have the same direction. When $J_{ij} < 0$, anti-ferromagnetic interaction is exhibited, and it becomes stable when adjacent spins are in opposite directions.

3.2.2 Simulated Annealing

Sampling method of Ising model

The energy and Hamiltonian of the whole system of the Ising model are given by the following functions.

$$H = \Gamma - \sum_{i,j} J_{ij}\sigma_i\sigma_j - \sum_i b_i\sigma_i \tag{3.1}$$

Here, b_i is a magnetic field applied to vertex v_i. Regarding the magnetic field, it is stable when $\sigma_i = +1$ when $b_i > 0$ and when $\sigma_i = -1$ when $b_i < 0$. The spin arrangement when the energy becomes minimum is called a ground state.

Now, σ_j in the expression 3.1 indicates a spin adjacent to σ_i. In other words, j exists in the 2 direction in the 1 dimension and in the 2 direction in the 2 dimension. In this Ising model, when it is in a statistically equilibrium

state, its distribution function (state sum) is expressed as follows.

$$Z = \sum_{\sigma_i} exp\{-\beta H(\sigma_i)\} \tag{3.2}$$

An attempt is made to sample a state that follows this statistical mechanical distribution function by the Monte Carlo method. Now consider a N state sequence $X_i, i \in N$. If the sequence is random, the following equation holds.

$$P_N(\{X_1, X_2, ..., X_N\}) = P_1(X_1)P_1(X_2)...P_1(X_N) \tag{3.3}$$

Here, $P(\{X\{)$ is the probability that N, the event X, holds. When this sequence is a Markov chain, the state is changed while satisfying the following equation.

$$P_N(X_1, X_2, ..., X_N) = P_1(X_1)T(X_1, X_2)T(X_2, X_3)...T(X_{N-1}, X_N) \tag{3.4}$$

Here, $T(X, X')$ is the transition probability from state X to X'. $T(X, X')$ is normalized by the following equation.

$$\sum_{X'} T(X, X') = 1 \tag{3.5}$$

In a certain Monte Carlo step t, the probability of being in the state X is represented as $\rho(X, t)$. At this time, the change in the probability at the transition from X to X' at $t + 1$ is the decrease in the probability at the transition from X to X', Based on the increase in the probability of transition from X' to X, it can be expressed as follows.

$$\rho(X, t-1) - \rho(X, t) = -\sum_{X'} \rho(X, t)T(X, X') + \sum_{X'} \rho(X', t)T(X', X) \tag{3.6}$$

A Boltzmann distribution is assumed as a steady-state distribution function.

$$\rho(X, t) = \rho(X, t+1) = exp(-\beta H(X)), \quad \rho(X', t) = exp(-\beta H(X')) \tag{3.7}$$

β indicates the inverse temperature and is expressed as follows using the Boltzmann constant k_B.

$$\beta = \frac{1}{k_B T} \tag{3.8}$$

By transforming the expression 3.6 using this expression, the following expression is obtained.

$$\sum_{X'} T(X, X') exp(-\beta H(X)) = \sum_{X'} T(X', X) exp(-\beta H(X')) \tag{3.9}$$

With the special solution of this equation as a sufficient condition, a solution that satisfies this relationship is obtained for all X and X'. This solution is referred to as a detailed balance solution.

$$T(X, X') exp(-\beta H(X)) = T(X', X) exp(-\beta H(X')) \tag{3.10}$$

Using the detailed balance solution, a metropolis sampling widely used in the Monte Carlo method is formulated. The transition probability $T(X, X')$ is expressed as follows using the symmetric matrix $\omega_{XX'}$ and the adoption probability $A_{XX'}$.

$$T(X, X') = \omega_{XX'} A_{XX'} \tag{3.11}$$

$\omega_{XX'}$ indicates the probability of selecting X' as the next transition destination in the next step when the state is X. By using $\omega_{XX'} = \omega_{X'X}$ because it is a symmetric matrix, the following equation can be derived from the detailed balance equation.

$$\frac{A_{XX'}}{A_{X'X}} = exp[-\beta\{H(X) - H(X')\}] \tag{3.12}$$

$A_{XX'}$ that satisfies this equation is as follows.

$$A_{XX'} = \begin{cases} exp[-\beta\{H(X) - H(X')\}] & (H(X) > H(X')) \\ 1 & (H(X) \leq H(X')) \end{cases}$$

The procedure of metropolis sampling is as follows.

1. In the state X, the next state X' is determined using $\omega_{XX'}$.
2. Transition to state X with probability $A_{XX'}$ (no transition with probability $1 - A_{XX'}$)

All of these operations are controlled using random numbers.

In the Ising system, in the case of d dimensions, one specific spin is selected from among N^d spins. Depending on the change in spin energy when

the transition from state X to X' is selected, Select whether to transition with a probability of 1 or $exp(-\beta\Delta H)$. As a similar sampling method, there is Gibbs sampling. In Gibbs sampling, transition occurs with the following probability without depending on the state before the transition.

$$A_{XX'} \propto exp\{-\beta H(X')\} \tag{3.13}$$

Since the spin state transition of the Ising model is basically performed by focusing on each spin, the parallelism is extremely low.

Annealing for Ising Model

In metal engineering, the process of gradually lowering the temperature after a substance is kidnapped is called annealing. In annealing, atoms are detached from their initial state and change to a higher energy state. Slow cooling in this state increases the likelihood of a change to a lower energy state than the initial state. Simulated annealing (SA method) is a highly versatile algorithm that imitates this annealing. One of the solutions to the NP complete problem is to use this SA method. This is a method in which a virtual temperature is introduced on a computer and a state close to a global optimal solution is obtained by appropriately controlling the temperature with time. The SA method randomly finds the neighborhood of the current solution. By repeating this operation, an optimal solution is obtained. In the process of finding the optimal solution, even if the value of the cost function when the combination is changed does not become small (it does not approach the optimal solution), it is selected as the next state with a certain probability. This control is performed using a temperature variable, and when the temperature is high, the transition of the combination becomes intense. When the temperature is low, the value of the cost function changes only to a low state, and it is possible to search for an optimal solution while preventing falling into a local solution. Now, when solving the combinatorial optimization problem using this SA method, some objective function must be defined. Given input $x = x_0, x_1, x_2, ..., x_n$, The following objective function needs to be minimized.

$$f(x) = -\sum_{i} C(x) \tag{3.14}$$

This objective function varies depending on the problem when creating an annealing machine having an optimal model according to the purpose. The advantages of the SA are as follows.

- It is hardly trapped a local solution and it is possible to find a global optimal solution (or a local optimal solution close to the global optimal solution)
- It is possible to set arbitrary constraints, and there is high flexibility for problems
- There are no restrictions on the objective function. High versatility because it can be applied to continuous functions and discrete functions
- Very simple to implement, very easy to implement

However, there are some disadvantages as follows.

- Simple to implement, but computationally intensive, requiring enormous computation time for some problems
- Optimal parameters need to be set depending on the problem

These two shortcomings come to the fore when trying to find a good solution. Although it is possible to obtain a good solution by increasing the amount of calculation, it is almost impossible to solve the problem in a realistic time. Also, there are various problems regarding parameter setting, such as how to update the temperature parameter. The algorithm of the SA method is as follows.

1. Initial state
2. Create next state
3. Acceptance of state by energy difference and temperature
4. Updating temperature parameters

5. Judgment of termination condition

For 2, 3, the process is repeated until the termination condition is satisfied. The termination condition includes various conditions, such as repetition a predetermined number of times and a sufficient decrease in temperature.

There is also a technique called quantum annealing that changes the quantum effect instead of temperature. In a state where the quantum effect is strong, all combinations are superimposed. The state transition due to the tunnel effect occurs as the quantum effect becomes stronger, but the state becomes stable when the quantum effect is reduced.

In order to apply the Ising model to the optimization problem, it is necessary to formulate the Ising spin glass ground state search problem. Here, how to apply some optimization problems will be described.

3.2.3 Optimization on Annealing Processors

Application to max-cut problem

The maximum weight cut problem is a problem in which vertices of a weighted graph are divided into two groups. There is a problem of maximizing the sum of the weights of the sides to be cut when grouping is performed.

This problem is formulated as follows When a graph $G = (V, E)$ with a vertex set V and an edge set E is input, the edge weight w_{ij} is considered, Find $V_1 \subseteq V$ and $V_2 \subseteq V(V \backslash V_1)$ that satisfy the following.

$$max. \sum_{v_i \in V_1, v_j \in V_2} w_{ij} \tag{3.15}$$

The value of each vertex is $v_x = \{-1, 1\}$, and if $v_x \in V_1$, the value of the vertex is $x_n = 1$, $v_x \in V_2 If$ is $x_n = -1$, The objective function is as follows, and finds x_i and x_j that minimize H.

$$H = \frac{1}{2} \sum_{v_i, v_j \in V(i<j)} w_{ij}(1 - x_i x_j) \tag{3.16}$$

When this equation is applied to the Ising model, it can be applied only by eliminating the constant term. That is, when applied to the Ising model, the

following equation may be used.

$$H = \frac{1}{2} \sum_{v_i, v_j \in V(i<j)} w_{ij} x_i x_j \tag{3.17}$$

Here, when compared with the Hamiltonian of the Ising model, the Ising model conversion of the maximum cut problem is It is possible to set $J_{ij} = -w_{ij}$ for the interaction coefficient J_{ij} of the Ising model.

3.2.4 Optimization on Annealing Processors

Now, in order to apply the traveling salesman problem to the Ising model, the problem is formulated. Now, assuming that the number of cities is n, and the distance between cities i and j is d_{ij}, the total distance when passing through all cities once is minimized.

$$D = \sum_{i \neq j} d_{ij} x_{ij} \tag{3.18}$$

Here, x_{ij} takes 1 when moving from city i to city j, and 0 when not moving. The constraints are as follows.

1. At the same time there is only one salesman (at any time, in some city)
2. Visit all cities once (no city has 0 visits)

If the above conditions are met, the traveling salesman problem of "visiting all cities once" is satisfied. At a certain time t, an expression satisfying the above requirements is as follows.

$$\sum_i x_{t,i} = 1 \tag{3.19}$$

$$\sum_t x_{t,i} = 1 \tag{3.20}$$

$x_{t,i}$ indicates whether or not it is in city i at time t ($x_{t,i} \in 0, 1$). When trying to find a solution that satisfies these two constraints, there is a method of "do not search for a state that does not satisfy the constraints". However, annealing cannot add such arbitrary states. Therefore, the objective

function needs to be modified. What an annealing machine can do is to minimize the objective function. When trying to find a solution that satisfies the constraints described above, a penalty term may be added to the objective function. This is a term that prevents an optimal solution from being obtained if the constraints are not satisfied. Here, in order to incorporate the two constraints into the objective function, the constraint 3.19 is modified as follows.

$$(\sum_{i} x_{t,i} - 1)^2 = 0 \tag{3.21}$$

In addition, since it must hold for all t, it is further modified as follows.

$$H_A = \sum_{t} (\sum_{i} x_{t,i} - 1)^2 \tag{3.22}$$

Also, the condition 2 is similarly transformed into the following expression.

$$H_B = \sum_{i} (\sum_{t} x_{t,i} - 1)^2 \tag{3.23}$$

Regarding the two expressions, H_A $geq0$ and H_B $geq0$ regardless of the value of x. H_A and H_B take the minimum value 0 only when the constraint condition is satisfied. When these two conditions are added to the objective function, it becomes as follows.

$$H = \sum_{i \neq j} d_{ij} x_{ij} + AH_A + BH_B \tag{3.24}$$

A, B are large enough numbers. Although it is possible to perform annealing using this objective function, the accuracy greatly changes depending on the values of the input parameters A and B.

3.3 Related Work

D-wave Systems proposed D-wave [1], 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 [9] [5] [11]. However, these are simulators designed for physics research, rather than combinatorial optimization problems. With the FPGA-based AP approach, hardware topologies, embedding [12], and random number generators [6] are mainly studied. In terms of the hardware topology, two approaches are mainly studied: the all-to-all type [2], [13], in which all spins are connected to each other; and the nearest-neighbor type [3], 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

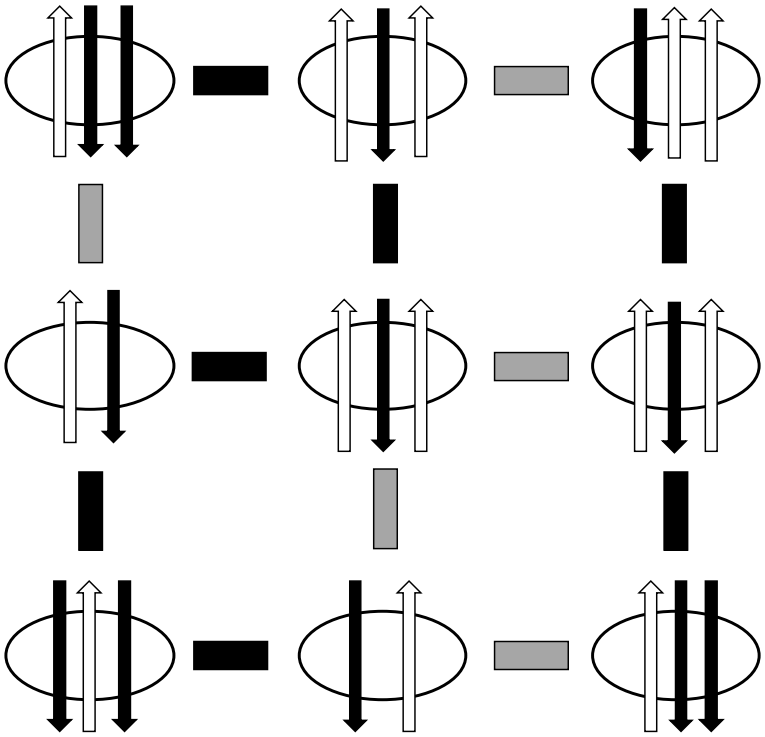


Figure 3.1: Quantum annealing in D-wave machine

hardware, additional conversions are required for the embedding process.

3.3.1 D-Wave

The Dwave machine is a quantum computer based on quantum annealing developed by D-wave Systems. The superconducting chip inside operates based on quantum annealing. The chip is packed with 2000 rings, which are connected by a superconductor. Each of these rings is in a quantum superimposed state. In a normal copper wire, if the same amperage current is passed clockwise and counterclockwise at the same time, the two cancel each other out to 0. However, in a superconducting element, both can exist simultaneously without canceling out in a very short time. By regarding the reverse current that exists at the same time as a spin state, it can be regarded that two states exist at the same time. If there are two rings, 4 combinations of spins can be expressed simultaneously. In the Ising model, the Hamiltonian based on the combination of spins is calculated, and the

minimum value is calculated.

As shown in Figure 3.1, each ring has upward and downward spins (actually clockwise and counterclockwise currents). Due to the stability of the downward and upward relationships, each interaction can be represented in gray and black in the figure to cope with the problem. While gradually weakening quantum superposition, he repeatedly determines his own spin state according to the interaction with the neighbor.

3.3.2 CMOS Annealing

CMOS annealing is a technique that simulates the Ising model with a CMOS circuit. In CMOS Annealing Processor (CMOS-AP) 3.2, 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 3.2 shows the operator unit for updating the states of spins. In this circuit, the operation shown in Equation 5.15 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.3.2(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.3.2(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.

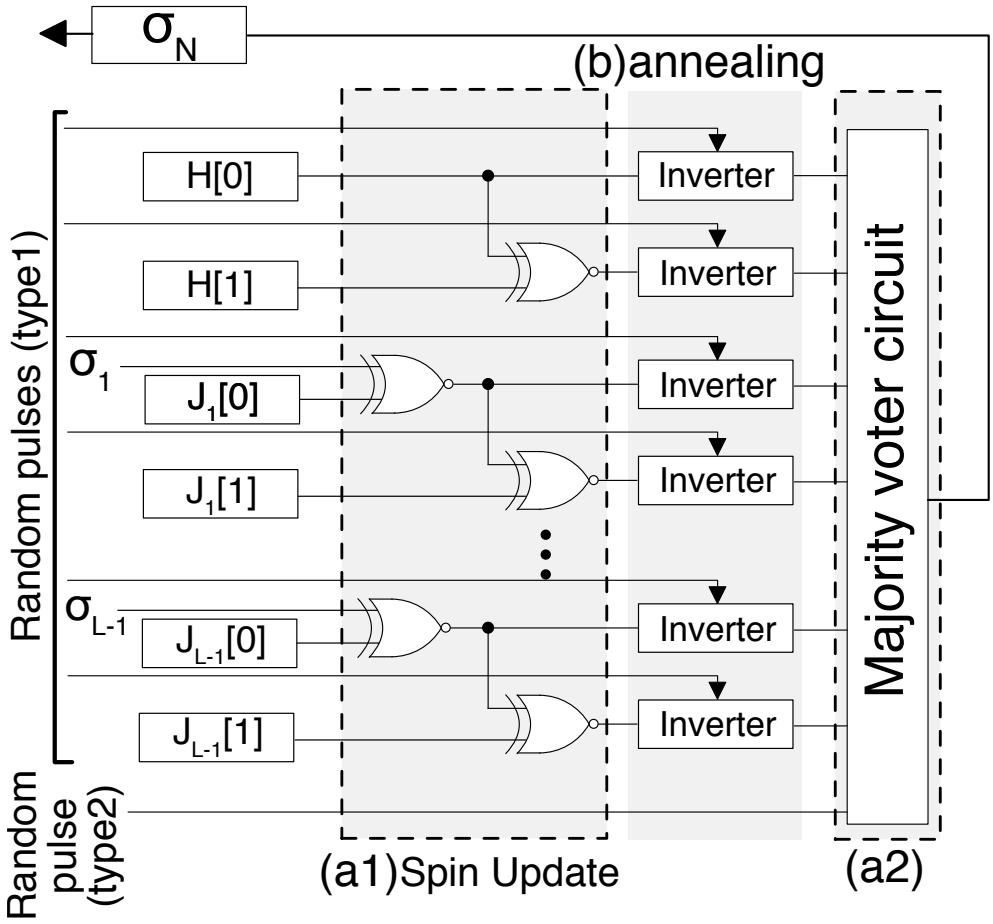


Figure 3.2: Operator unit

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. 3.2(b) with two kinds of random pulses. For details regarding these random pulses, see [14]. 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.3.3 shows the spin unit proposed in the previous study. Since the spin unit holds four fully-coupled spins and the adjacent spins cannot

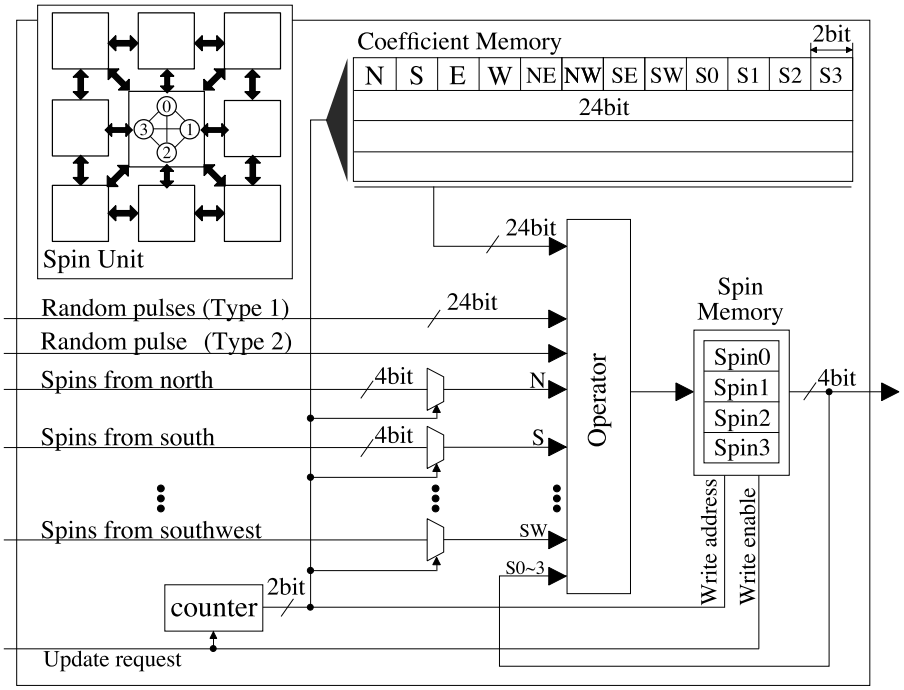


Figure 3.3: Spin unit

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 depends 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

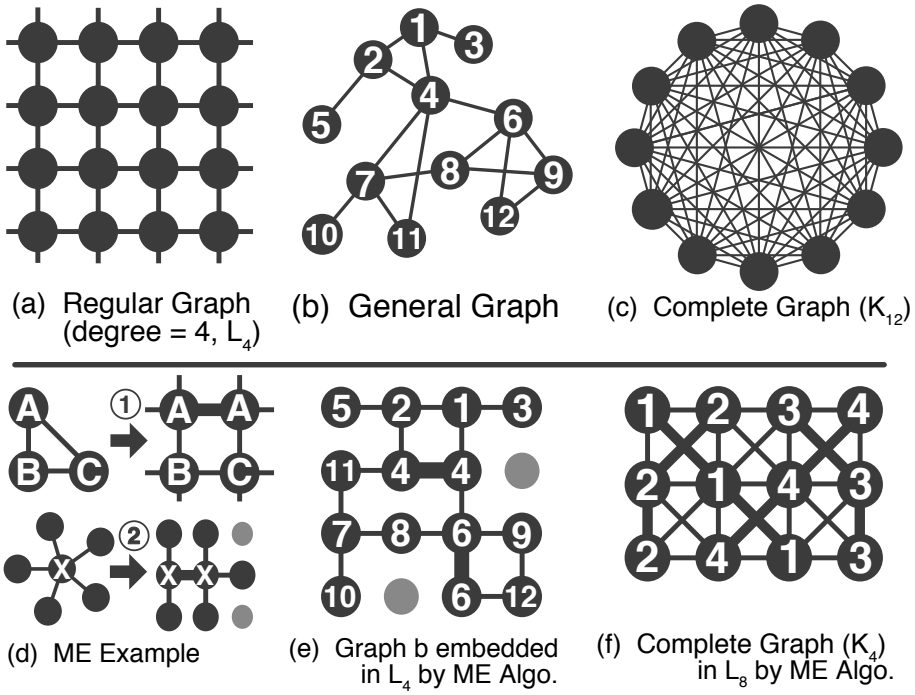


Figure 3.4: Minor embedding

position of the adjacent spin units as shown in the upper left of the Fig.3.3. 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.

Minor Embedding

Minor Embedding (ME) [15] 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 [12] using the three graphs shown in Fig. 3.4. 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

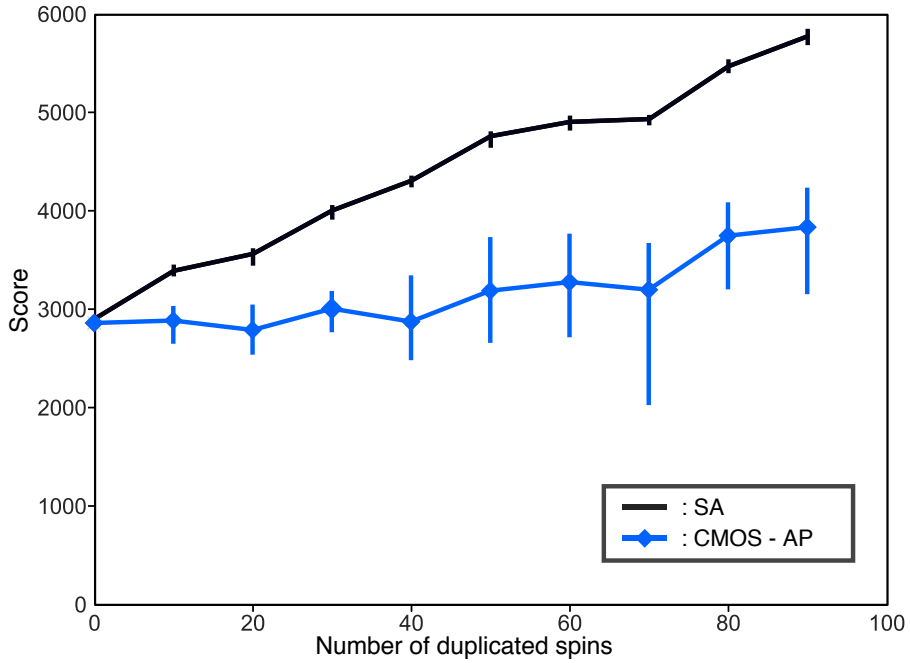


Figure 3.5: Decline in accuracy due to duplicated spins

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. 3.4(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:

$$\#(1, x) = x \tag{3.25}$$

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

$$\#(y,x) = \begin{cases} \#(y-1, max(1, x-1)) \& (x+y\ is\ even) \\ \#(y-1, min(n, x+1)) \& (x+y\ is\ odd) \end{cases} \tag{3.26}$$

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.

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:

$$new\ connection\ i = \sum |J_{ij}| \tag{3.27}$$

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 3.5 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.

3.3.3 Digital Annealer

Digital Annealer is an FPGA-based annealing processor that calculates the ground state of a fully coupled Ising model using simulated annealing. The most significant feature of the fully-coupled annealing processor is that the

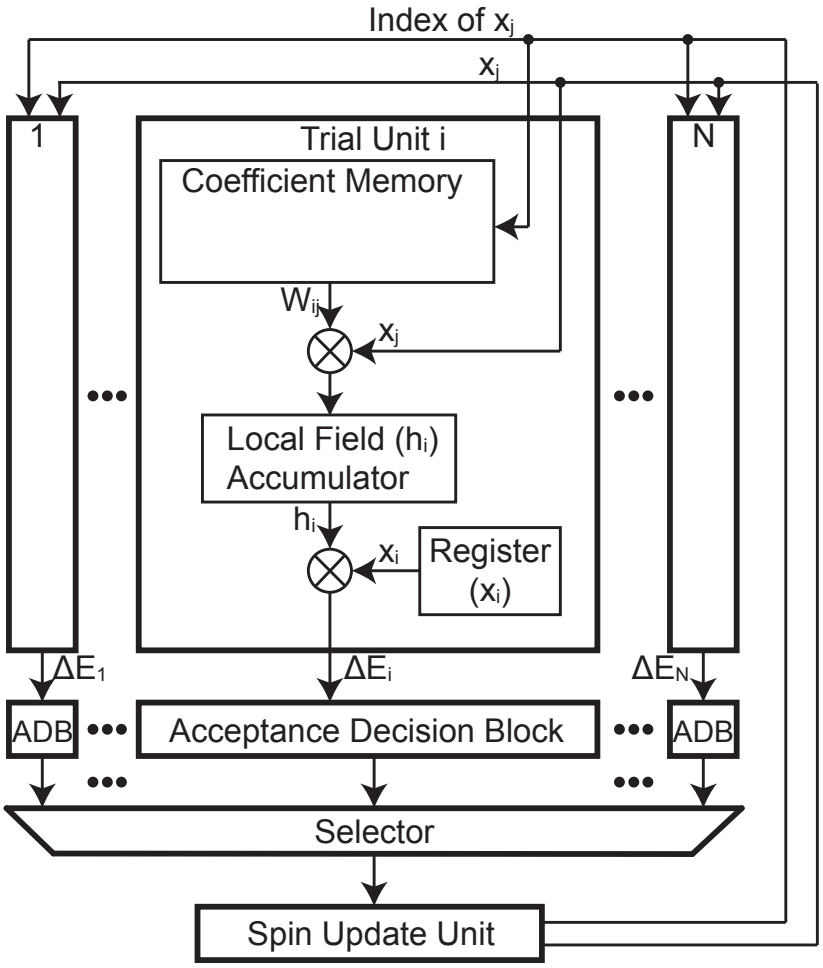


Figure 3.6: Architecture of Digital Annealer

number of spins that can be updated at a time is limited to one regardless of the number of spins at a time due to the limitation of the Glauber dynamics, resulting in low parallelism of hardware. The problem of becoming a problem. Specifically, when each spin unit inverts its own spin state with respect to the current spin arrangement, it is determined in parallel whether or not the state is accepted by Simulated Annealing. Select one and update the spin state. The advantage of this process is that the state of spins often changes because N next states can be verified from the current spin arrangement. If the fully coupled Ising model is implemented intuitively with CMOS Annealing, only one spin can be verified for N spins.If the verified spin state is not accepted,

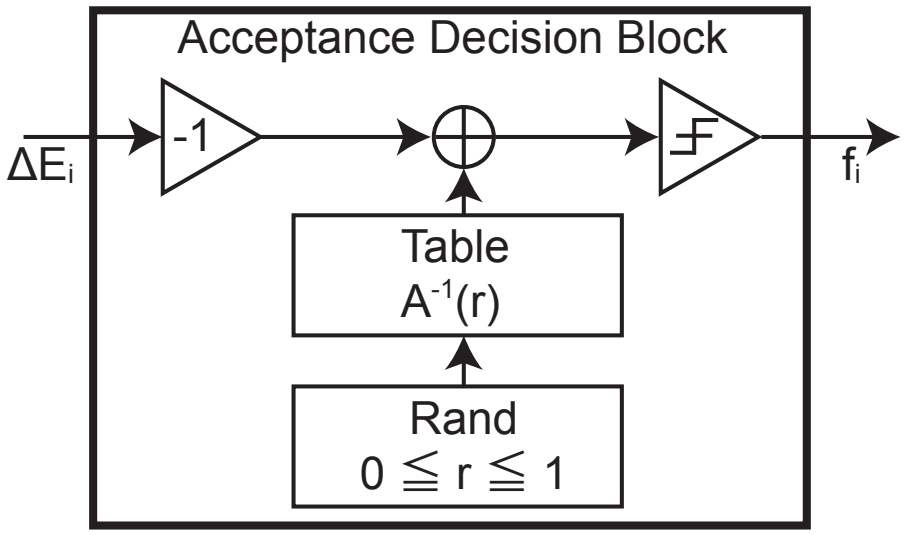


Figure 3.7: Acceptance Decision Block

the spin state changes almost even with the number of steps. There is a problem that it takes much time to reach the ground state.

Figure 3.6 shows the architecture of Digital Annealer. In this architecture, the "trial unit" that calculates and holds the local field of each spin, determines whether the next state can be accepted from the local field, and outputs an Acceptance Dicision Block that outputs flags, and the output N flags And a Spin Update Unit that updates the spin state of the selected spins.

Figure 3.7 shows the details of ADB. Normally, when determining whether or not the next spin state candidate is acceptable, the determination is made by calculating the reversal probability based on ΔE and comparing it with a random number. On the other hand, ADB compares the generated random number with ΔE by using a memory storing an inverse function of the function for calculating the inversion probability. Thereby, it is possible to simplify the hardware of the function necessary for the probability calculation. However, since a memory is required for the square of the bit precision, it is possible that this does not lead to a significant decrease in computational resources.

3.3.4 Coherent Ising Machine and Simulated Bifurcation

The Coherent Ising Machine and Simulated Bifurcation reproduce the fully coupled Ising model, but are not strictly annealing processors. These two approaches are common to annealing processors in that they simulate the behavior of the Ising model, but do not use annealing as the operating principle. The greatest feature is that the spin state can be updated in parallel even in the fully coupled Ising model. These two approaches make use of a phenomenon called bifurcation, in which the two branches correspond to an upward spin and a downward spin, respectively. The Coherent Ising Machine uses an oscillator called degenerate parametric oscillators (DPO) to simulate the behavior of the Ising model using a laser and an FPGA. On the other hand, in the Simulated Bifurcation, all calculations are performed on the digital circuit using the FPGA for the branching phenomenon.

However, since CIM uses an optical system, the system becomes larger than CMOS Annealing or SB, and in order to increase the number of spins, it is necessary to increase the length of the optical fiber. In order to calculate the branch state, SB needs to keep the spin state, which is stored as binary in CMOS Annealing, as a continuous value. Further, since a product-sum operation of the interaction coefficient and its spin state is required, there is a problem that the data path and the calculation unit tend to be more complicated than the Simulated Annealing based approach.

3.4 Goal of This Research

As mentioned in Chapter 2, 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. In addition, no significant improvement in computer performance can be expected with the end of Moore's law.

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. 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.

In the Chapter 3, existing annealing processors and their features are summarized. Based on these, the issues of existing annealing processors are as follows.

1. 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.

2. In the annealing processors which has the nearest-neighbor Ising model, the accuracy and convergence speed of the solution decrease if there is a mismatch between the hardware Ising model and the problem.

3. When calculating the ground state of the fully coupled Ising model by simulated annealing, the number of update spins is limited to one in principle. Therefore, high parallelism cannot be obtained unlike the nearest-neighbor annealing processor.

In the following chapters, I propose solutions to each of these three issues.

Bibliography

[1] M. Johnson, M. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. Berkley, J. Johansson, P. Bunyk et al Quantum annealing with manufactured spins, Nature, vol. 473, no. 7346, pp. 194–198, 2011.

[2] K. Someya, R. Ono, and T. Kawahara Novel ising model us- ing dimension-control for high-speed solver for ising machines 2016 14th IEEE International New Circuits and Systems Conference (NEWCAS), pp.1–4, June 2016

[3] M. Yamaoka, C. Yoshimura, M. Hayashi, T. Okuyama, H. Aoki, and H. Mizuno 20k-spin Ising chip for combinational optimization problem with CMOS annealing ISSCC Dig. Tech. Papers, pp. 432–433, 2015.

[4] A. Lucas Ising formulations of many NP problems Frontiers in Physics, vol. 2, no. 5, 2014.

[5] W. Hastings, Monte carlo sampling methods using Markov chains and their applications Biometrika, vol. 57, pp. 97–109, 1970.

[6] H. GYOTEN, M. HIROMOTO, and T. SATO Area efficient annealing processor for ising model without random number generator IEICE Transactions on Information and Systems, 2018.

[7] T. Inagaki et al A Coherent Ising Machine for 2000-Node Optimization Problems Science, vol. 354, no. 6312, pp. 603-606, Nov. 2016.

[8] 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, Sep 2011.

[9] 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.

[10] F. Barahona, On the computational complexity of Ising spin glass models, Journal of Physics A: Mathematical and General, vol. 15, no. 10, p. 3241, 1982.

[11] A. Gilman, A. Leist, and K. Hawick, 3D lattice Monte Carlo simulations on FPGAs, in Proceedings of the Inter- national Conference on Computer Design (CDES). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (World- Comp), 2013.

[12] C.Y. Takuya Okuyama, M. Hayashi, S. Tanaka, and M. Yamaoka, Contractive graph-minor embedding for cmos ising computer(in japanese), tech. rep.

[13] S. Tsukamoto, M. Takatsu, S. Matsubara, and H. Tamura Anaccelerator architecture for combinatorial optimization problems 2017

[14] C. Yoshimura, M. Hayashi, T. Okuyama, and M. Yamaoka FPGA-based Annealing Processor for Ising Model. In 2016 Fourth International Symposium on Computing and Networking (CANDAR). 436–442. https://doi.org/10.1109/ CANDAR.2016.0081

[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.

Chapter 4

Time-Division Multiplexing Architecture for Large-Scale Sparse Ising Model

4.1 Introduction

In this chapter, I explore a novel FPGA-based Isign machine architecture for increasing the simulated spin count. I found that the prior FPGA-based implementations did not utilize on-chip memory blocks as known as Block RAM (or BRAM in short) in Xilinx FPGAs) on an FPGA effectively. Thus I focus on employing the on-chip memory block to increase the spin count.

1. I present a time-division multiplexing architecture of Ising machine on an FPGA that utilizes on-chip memory blocks for the spin count increase. It enables to flexibly expand the spin count independently of the available logic circuit resources of an FPGA, such LUTs and registers.

2. I evaluated the efficiency of our proposal by using commercial FPGA synthesis tools. The evaluation results show that our architecture can handle 64 times more spin than previous one.

4.2 Time-Division Multiplexing Architecture

4.2.1 Overview

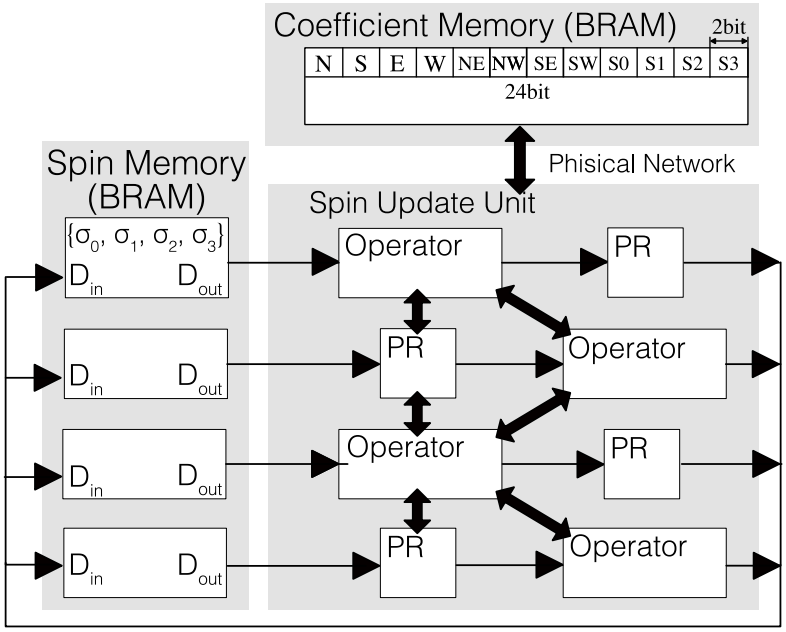


Figure 4.1: Time-Division Multiplexing Architecture

We propose a time-multiplexing architecture that processes large combinational optimization problems on limited hardware resources by separating spins of a target problem into both spacial and temporal directions.

Figure 4.1 presents the proposed architecture and processing mechanism. The architecture consists of a spin memory (Spin Memory) that stores the state of all spins, an interaction coefficient memory (Coefficient Memory) that stores the interaction coefficient between adjacent spins, and Spin Update Unit wihch updates the state of spins row by row including Operator / Pipeline Register (PR) array. In the Spin Update Unit , there is a constraint that adjacent spins can not be updated simultaneously [**?**]. therefore the pipeline registers are sandwiched between Operators.

The behavior of the architecture is as follows. As mentioned above, four complete spins (local spin) are contained within one spin unit in the chimera topology. This architecture updates sequentially from the local spin 0 in

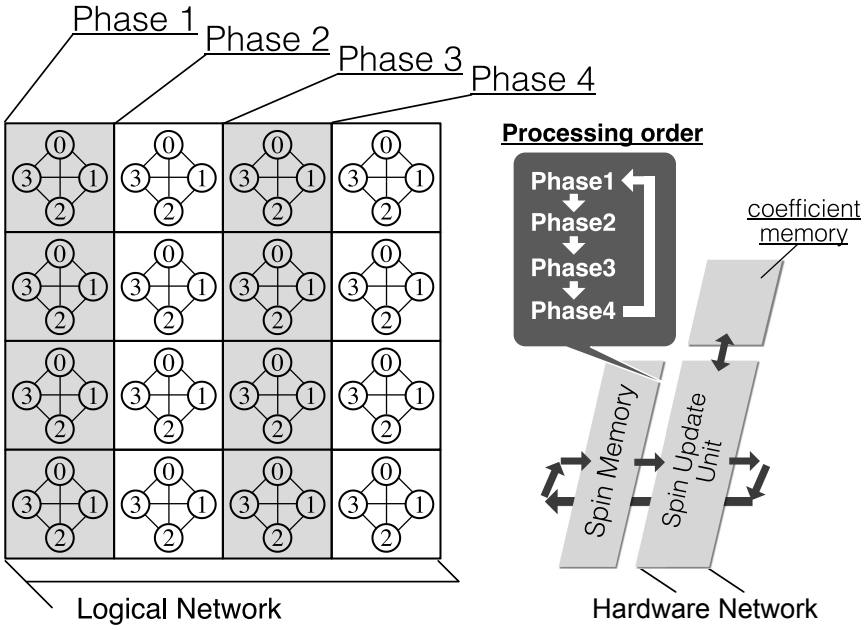


Figure 4.2: Time-Division Multiplexing Process

the chimera topology. A target Ising network is separated into 4 sections; We hereinafter call the section "Phase". First, spins in phase1 shown at the top of the figure 4.2 are read and transferred to the Spin Update Unit. the interaction coefficients of Phase 0 and Local spin 0 of Phase 2 are read from Spin Memory and Coefficient Memory, respectively. Then, they are transferred to the Spin Update Unit. Based on the read value, the Spin Update Unit updates the local spin 0 of phase 1. The back operator whose PR is inserted in the preceding stage receives the updated spin state from the front operator. This process is similarly performed up to Phase 4. After that, processing for local spin 1 is performed from phase 1 and it is repeated until all the local spins are updated. The architecture repeats this until the Ising model converges.

4.2.2 Scalability

It is possible to increase the spin count by dividing the Ising structure as long as it can be stored in BRAM. However, the more the process is divided, the more time it takes to update once. Thus, in order to improve the processing speed, it is necessary to allocate many Spin Units. Since the Ising model

Table 4.1: Resource Utilization of Entire Design

	#. Spin Unit	LUT	FF	BRAM
Available		303600	607200	1030
conventional CMOS annealing	4 (2×2)	682 (0.22%)	833 (0.13%)	0.5
	16 (4×4)	1305 (0.42%)	905(0.14%)	0.5
	64 (8×8)	3897 (1.28%)	1193(0.19%)	0.5
time division CMOS annealing	4×4	663 (0.22%)	825 (0.14%)	4 (0.39%)
	8×8	832 (0.28%)	831 (0.14%)	4 (0.39%)

can update using information of adjacent spins, parallelism can be increased by arranging Operator / PR arrays in the column direction as long as the memory bandwidth of BRAM and circuit resources allows.

In this implementation, the processing unit is assumed to be one line, but depending on the problem it may be divided into rectangles. In such a case, it is necessary to consider the sharing of the boundary part of the processing unit. For example, when processing 8×4 networks on this architecture, First, the upper 4 rows are processed in the horizontal direction, then the lower 4 rows are processed in the same way. In processing the 4th and 5th rows, the states of spins of the 5th and 4th rows are required, respectively. Therefore, processing can not be performed because there is no mechanism sharing the boundary part of the vertical processing in the architecture described above. This problem can be solved by connecting the top and bottom (torus) operators in the Spin Update Unit so that the data at the boundary can be shared.

4.3 Evaluation

4.3.1 Setup

We evaluated the resource utilization of the conventional architecture and our architecture. Our target borad is Xilinx Virtex-7 FPGA VC707 evaluation kit (FPGA : Virtex-7 XC7VX485T). Both architectures were synthesized in

Table 4.2: Resource Utilization of Random Pulse Generator

	Random Pulse Generator			
	#. Spin Unit	LUT	FF	BRAM
conventional CMOS annealing	4 (2×2)	501	809	0.5
	16 (4×4)	597	809	0.5
	64 (8×8)	981	809	0.5
time division CMOS annealing	4×4	509	809	0.5
	8×8	547	809	0.5

Table 4.3: Resource Utilization of Spin Unit

	Spin Unit			
	#. Spin Unit	LUT	FF	BRAM
conventional CMOS annealing	4 (2×2)	45	6	0
	16 (4×4)	42	6	0
	64 (8×8)	42	6	0
time division CMOS annealing	4×4	33	0	1.5
	8×8	34	0	1.5

verilog HDL and Vivado Design Suite 2016.2.

The dotted line in the figure 4.3 and 4.4 is extrapolated based on a small scale implementation. All spin spin units update at each clock. In the conventional CMOS annealing machine, all spins are updated in 4 clocks, whereas in the proposed architecture, it takes 4 clocks for one phase.

4.3.2 Resource Utilization

Tables 5.1 and 4.3 show the resource usage of the conventional architecture and our proposed architecture. Although the resource amount in the paper of the conventional architecture differs from the resource amount shown in this paper, it is thought that the minimum implementation was done in order to focus on the resource usage amount in this evaluation. As shown in the table, when the conventional architecture and the proposed architecture for

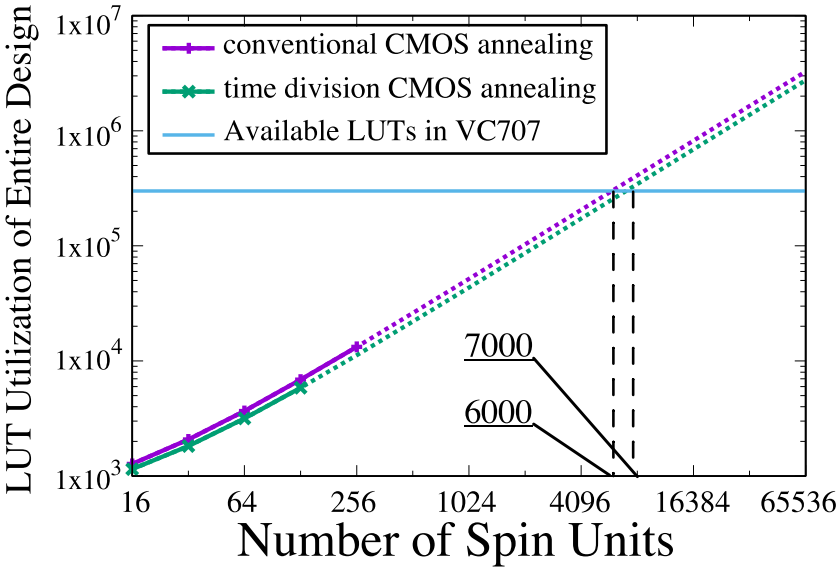


Figure 4.3: Estimation of LUT consumption

the same problem size are compared, the number of Spin Units required for processing is reduced by the time division processing. Therefore, the total resource consumption of the proposed architecture is smaller than that of the conventional architecture Even when Spin Units are of the same size, the proposed architecture has less LUT and FF. it is considered that LUT and FF consumption per Spin Unit of our architecture are reduced from the resource consumption amount of each module in the table. One of the factors to reduce the resource consumption of the Spin Unit is to store the spin state and interaction coefficient in the BRAM. The other is the difference of the method of supplying the spin states and the interaction coefficients to the Operator. In the conventional architecture, data related to the spin to be updated is selected by the selector for supplying data to the Operator. On the other hand, since our architecture enables to acquire necessary data by just accessing the BRAM, selectors are unnecessary in front of each Operator in the proposed architecture.

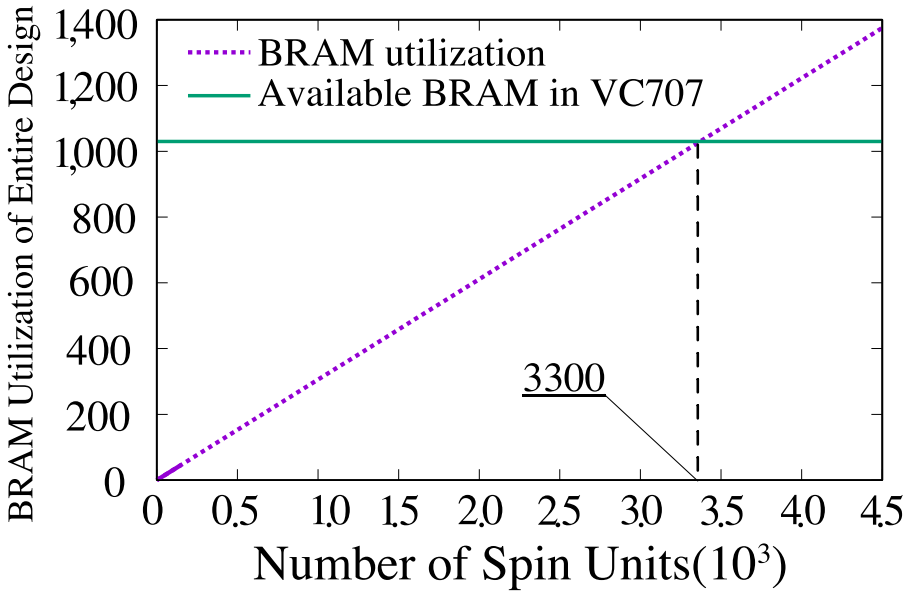


Figure 4.4: Estimation of BRAM consumption

4.3.3 Spin Count

Based on these results of syntheses, we estimated the number of Spin Units that can be deployed on two architectures on the target board. The results for the LUT are shown in figure 4.3, and the results for the BRAM are shown in figure 4.4. Note that the horizontal axis of figures is not the number of spins, but the Spin Unit (4 spins).

As can be seen from the figure, since the LUT consumption of the proposed architecutre is smaller than that of conventional method, the proposed architecture can allocate many Spin Units. However, even with the maximum use of 2000 BRAMs of 18 kb (1000 in the case of using 36 kb), there are about 3,300 Spin Units that can be deployed considering BRAM consumption in figure 4.4. In other words, compared to the conventional architecture, our architecture has the same number of spins by dividing the process into two, and it can handle the spin number over the conventional architecture with division of three or more. Further, if the Ising network is devided by 128, our architecture can hold about 211,000 spins 64 times larger than the conventional architecture.

4.4 Conclusion

In this study, we proposed an annealing machine of Ising model using BRAM dedicated to combinatorial optimization problem. Compared with the architecture of the previous research, the number of spins that can be processed at the same time is halved because the read bit width of the BRAM is limited. however, by maximizing the use of BRAM, it is possible to deploy 64 times the spin units of the previous research.

In addition, we discussed efficient use of BRAM and advantage of time division multiplexing with BRAM. In the case where there is a need to solve the problem that can not be handled by the conventional architecture, the BRAM based architecture is indispensable. Therefore, research that alleviates the overhead due to time division processing is necessary.

Bibliography

[1] T. Okuyama, C. Yoshimura, M. Hayashi, and M. Yamaoka Computing architecture to perform approximated simulated annealing for Ising models In 2016 IEEE International Conference on Rebooting Computing (ICRC). 1–8. https://doi.org/10.1109/ICRC.2016.7738673

[2] K. Sano, Y. Hatsuda, and S. Yamamoto Scalable Streaming-Array of Simple Soft-Processors for Stencil Computations with Constant Memory-Bandwidth In 2011 IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines. 234–241. https://doi.org/10.1109/FCCM.2011.12

[3] G. Marsaglia Xorshift RNGs, Journal of Statistical Soft- ware, vol. 8, no. 14, pp. 1–6, 2003.

[4] M. Hayashi, M. Yamaoka, C. Yoshimura, T. Okuyama, H. Aoki, and H. Mizuno Accelerator chip for ground-state searches of ising model with asynchronous random pulse distribution International Journal of Networking and Computing, vol. 6, no. 2, pp. 195–211, 2016.

[5] 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.

[6] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Optimization by simulated annealing," Science 220, pp. 671–680, 1983.

[7] R. Y. Li, R. Di Felice, R. Rohs, and D. A. Lidar, Quantum annealing versus classical machine learning applied to a simplified computational biology problem NPJ quantum information 4, 14, 2018.

[8] M. Hernandez, and M. Aramon Enhancing quantum annealing performance for the molecular similarity problem Quantum Information Processing 16, 133, 2017.

Chapter 5

Time-Division Multiplexing Architecture for Multi-Layered Sparse Ising Model

5.1 Introduction

In this chapter, the architecture for more complicated graph of a problem is proposed. Because the spin-to-spin connections in hardware are 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. My previous work in chapter 4 suggested that this deformation leads to a decrease in solution accuracy. The contributions of this paper are as follows:

1. I 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.

5.2 Stochastic Cellular Automaton Annealing

5.2.1 Definition

SCA (Stochastic cellular automaton) is one of the automata. SCA is an automaton using a discrete-time Markov chain in the product space S^V. An automaton is a virtual machine that transitions between a plurality of internal states according to certain rules. Using the combination of the current state and the input, transit to the specified next state. If there is no combination that satisfies the condition, the current state is maintained. This rule can be expressed as a state transition diagram. There are several types of automata. There is a finite automaton having a finite number of inputs and states, and a cellular automaton in which a plurality of automata are arranged and interacted. Among the finite automata, there are deterministic finite automata in which the next state is uniquely determined by the current state and input, and nondeterministic finite automaton in which the next state is not uniquely determined. SCA is a kind of non-deterministic finite automaton.

DFA

Deterministic Finite Automaton (DFA) is a finite automaton in which the next transition state is uniquely determined by the current state and input. A deterministic finite automaton has the following 5 elements.

- Q : Finite set of states
- $\sum$: Finite set of inputs
- δ : State transition function

- q_0 : Initial state
- F : Set of final states

An automaton with these 5 elements is uniquely determined (defined). The defined automaton is expressed as follows

$$M = \{Q, \sum, \delta, q_0, F\} \tag{5.1}$$

An example of the expression of this automaton M is shown below.

- $Q = \{a, b\}$
- $\sum = \{0, 1\}$
- $\delta(a, 0) = a, \delta(a, 1) = b, \delta(b, 0) = b, \delta(b, 1) = a$
- $q_0 = a$
- $F = \{b\}$

This can be represented by a state transition diagram as shown in Figure 5.2.1. Using DFA, it is possible to determine whether the set of input symbol strings $\sum^*$ is accepted or not (rejected).

Cellular automaton (CA) is an automaton that changes its state according to a grid of cells and rules, and is a parallelization of deterministic finite automata as described above. State transition is performed discretely. Each cell changes according to its state transition function, but also considers the simultaneous interaction of each cell. The SCA changes the state of each cell according to a certain probability distribution at the time of state transition. A deterministic finite state automaton has a state transition that is uniquely determined, whereas an SCA shows non-determinism, so a state cannot be uniquely determined. The state is changed according to a certain probability distribution.

SCA transitions states according to the discrete-time Markov chain Monte Carlo method in the product space S^V. A discrete-time Markov chain means that on a non-continuous (discrete value) time axis, the next state depends

only on the current state. When predicting the state of the next time $n+1$ at a certain time n, Markov states that the state of the time $i \in (0, 1, ..., n-1)$ is irrelevant. It is called sex. At time n, a random variable taking a value on S is set as X_n. X_n is a discrete-time Markov chain with transition probability matrix $p(i, j)$, given arbitrary states $i_0, i_1, ..., i_{i-1}, i_n, j$ Then, it can be expressed as follows.

$$P(X_{n+1} = j | X_n = i, X_{n-1} = i-1, ..., X_0 = 0) = P(X_{n+1} = j | X_n = i) \quad (5.2)$$

This transition probability can also be expressed as follows.

$$P(X_{n+1} = j | X_n = i) = p(i, j) \quad (5.3)$$

It has been mathematically proven that SCA behaves similarly to the Ising model when it is sampled using Gibbs measure in a finite space. Then the Hamiltonian looks like this:

$$H(\sigma) = \sum_{i \in V} h_i(\sigma)\sigma_i \quad (5.4)$$

$$h_i(\sigma) = -\sum_j J_{ij}\sigma_j \quad (5.5)$$

When the probability of state transition is defined by Gibbs scale, it is expressed by the following equation.

$$\pi^G(\sigma) = \frac{e^{-H(\sigma)}}{Z^G} \equiv \frac{\omega^G(\sigma)}{\sum_\sigma \omega^G(\sigma)} \quad (5.6)$$

$$Z^G = \sum_\sigma e^{-H(\sigma)}, \qquad \omega^G(\sigma) = -H(\sigma) \quad (5.7)$$

When represented by the above equation, the transition probability is represented by the following equation.

$$P^G_{\sigma,\tau} = \begin{cases} \frac{1}{|V|}\frac{e^{h_i(\sigma)\sigma_i}}{e^{h_i(\sigma)\sigma_i} + e^{-h_i(\sigma)\sigma_i}} & (\tau = \sigma^i) \\ 1 - \sum_{i \in V} P^G_{\sigma,\sigma^i} & (\sigma = \tau) \\ 0 & (otherwise) \end{cases}$$

However, SCA does not focus on single spins, but focuses on multiple spins and selects whether to make transitions. When implementing in SCA, the Hamiltonian is defined as follows.

$$H(\sigma, \sigma') = \sum_{i \in V} [h_i(\sigma)\sigma'_i + q(1 - \sigma_i\sigma'_i)] \quad (5.8)$$

When $q > 0$ and the spin set follows a Markov chain, the transition probability for SCA is defined as follows.

$$P^{sca}_{\sigma,\sigma'} = \frac{e^{-H(\sigma,\sigma')}}{Z_\sigma} \tag{5.9}$$

$$Z_\sigma = \sum_\tau e^{-H(\sigma,\tau)} = \omega^{sca}(\sigma) \tag{5.10}$$

Considering the same as applying the Gibbs scale, the scale for SCA is as follows.

$$\pi^{Sca}(\sigma) = \frac{\sum_\tau e^{-H(\sigma,\tau)}}{\sum_{\tau,\tau'} e^{-H(\tau,\tau')}} \equiv \frac{\omega^{sca}(\sigma)}{\sum_\tau \omega^{sca}(\tau)} = \frac{\omega^{sca}(\sigma)}{Z^{sca}} \tag{5.11}$$

From the above, the transition probability for SCA is as follows.

$$P(\sigma,\sigma') = \prod_{i\in V} P(\sigma'_i|\sigma) \tag{5.12}$$

$$P(\sigma'_i|\sigma) = \frac{e^{\sigma'_i(h_i(\sigma)+q\sigma_i)}}{2cosh(h_i(\sigma)+q\sigma_i)} \tag{5.13}$$

5.3 Time-Division Multiplexing Architecture

5.3.1 Processing in Conventional Architecture for Duplicate Spins

$$new\ connection\ i = \sum |J_{ij}| \tag{5.14}$$

In this section, we explain our solution to the decline in performance from interactions between duplicated spins as mentioned in 3.3.2. As shown in Fig 5.1(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 5.1(a-1) and (a-2) show the process of updating the duplicated spin in the conventional architecture [2] 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. (5.14).

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*,

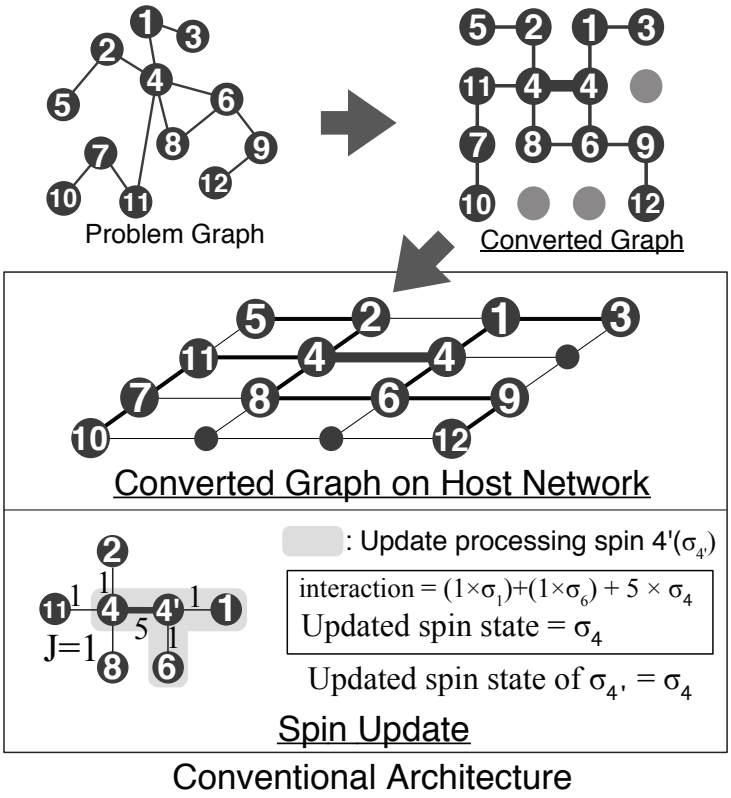


Figure 5.1: Updating process of duplicated spins by conventional architecture

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.

5.3.2 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 the memory to the spin unit is switched with time-division multiplexing. This

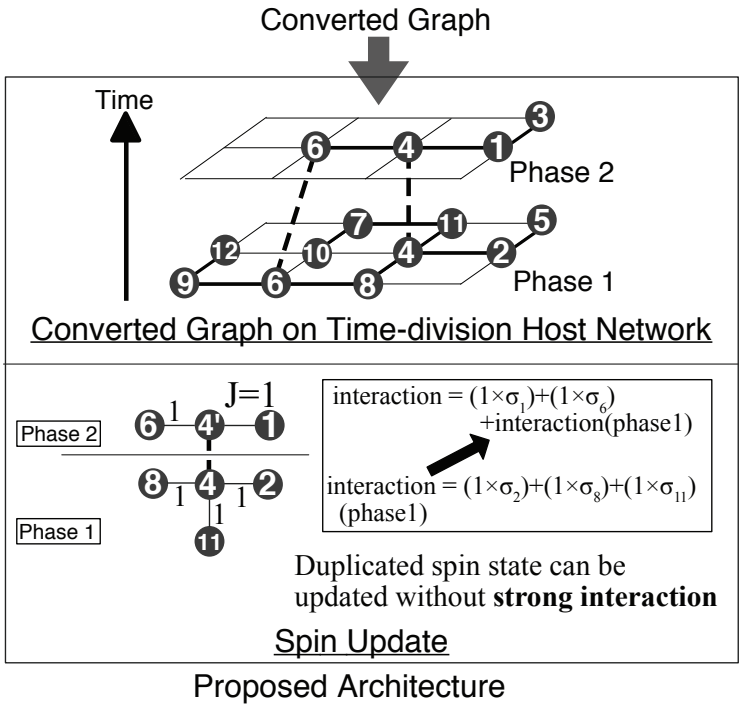


Figure 5.2: Updating process of duplicated spins by proposed architecture

allows one spin unit to process multiple spins.

$$Interaction = \sum_{j=1}^{L-1} J_{ij}\sigma_j + h_i \tag{5.15}$$

Our previous architecture [5] used this property to process large-scale graphs in the spatial direction. Unfortunately, according to Eq. 5.15, 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. 5.1(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

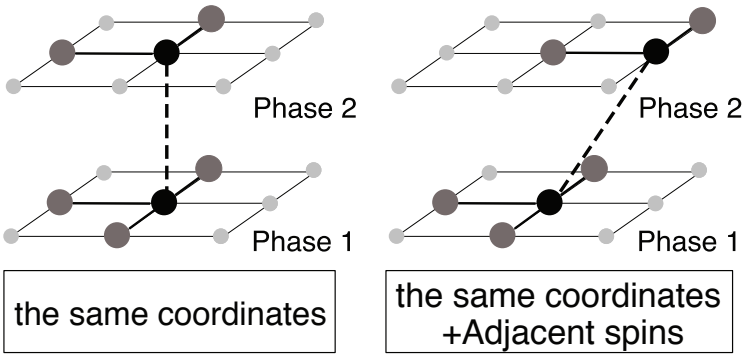


Figure 5.3: Continuous processing

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 equation 5.16.

$$Interaction = \sum^{phase} (\sum_{j \in phase\, N}^{L-1} J_{ij}\sigma_j) + h_i \tag{5.16}$$

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. 5.1(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. 5.1(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. 5.1(b). This makes it possible to allocate graphs with arbitrary

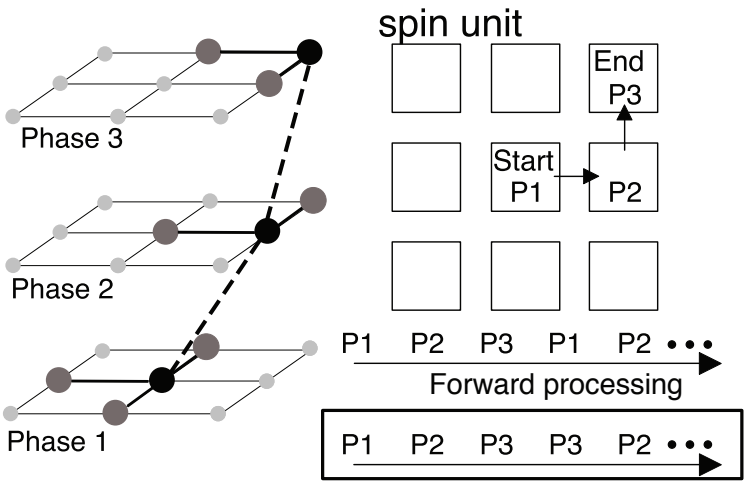


Figure 5.4: Reverse order processing

connections, as shown in Fig. 3.4(f) in the time direction.

5.3.3 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 **??**(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. 5.5 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. 5.5, where phase $1 > 2$ is defined as a forward order and phase

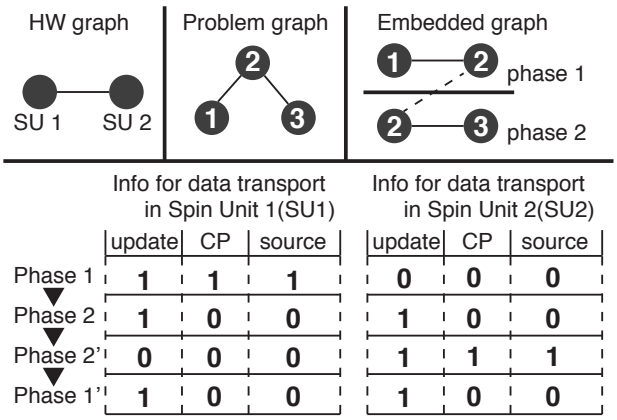


Figure 5.5: Example for Continuous Processing and Reverse-order Processing

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.

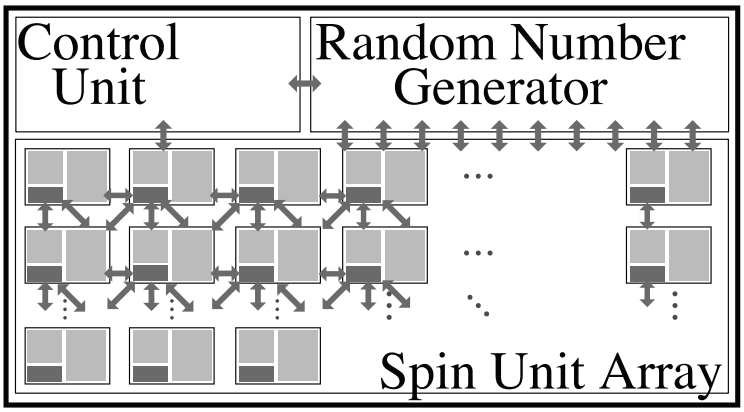


Figure 5.6: Time-division multiplexing architecture

5.3.4 Architecture Overview

We propose a time-division multiplexing architecture that processes complex and large combinational optimization problems with limited hardware resources by separating the 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. 5.8. 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 [2]. With the proposed architecture, they are constituted on the premise of BRAM (Block RAM in Xilinx FPGAs), as shown on the left in Fig.5.8. 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

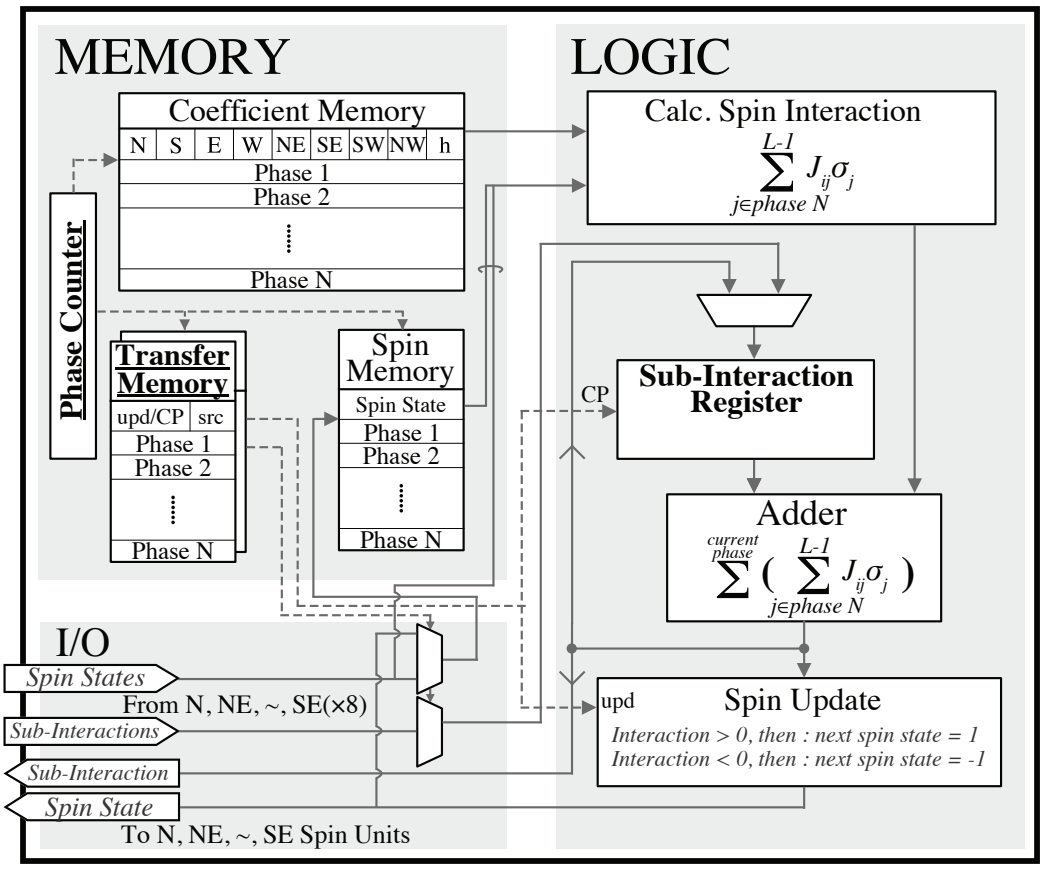


Figure 5.7: Spin unit

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

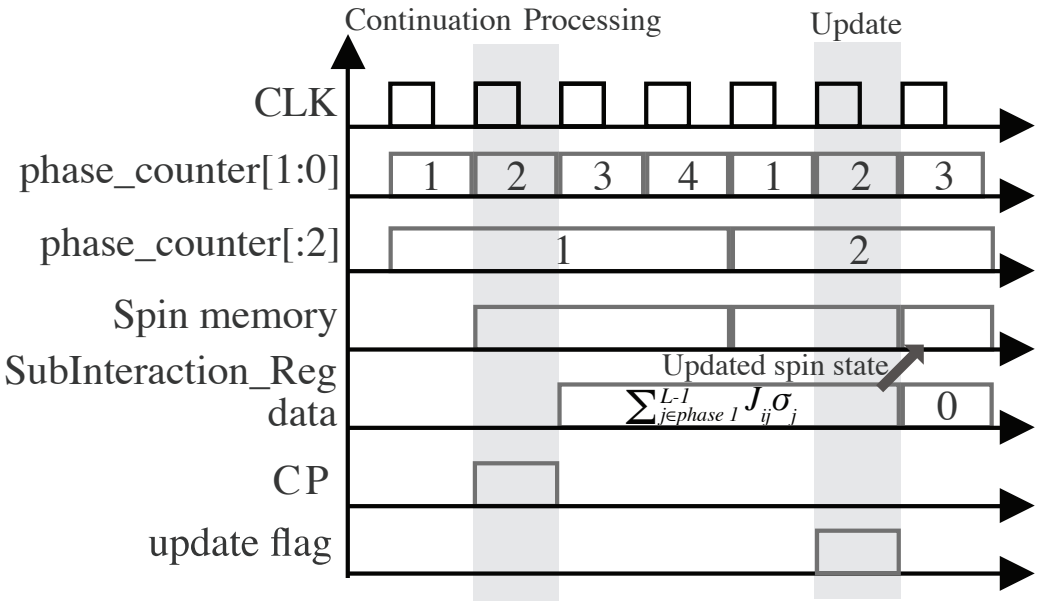


Figure 5.8: Timing Chart

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. 5.8. 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.

5.4 Evaluation

5.4.1 Setup

We evaluated the resource utilization of our architecture and the conventional CMOS annealing processor (**CMOS-AP**) because our architecture was devised based on (**CMOS-AP**). The proposed architecture with continuous processing is hereafter referred to as **TDM with CP**, and our previous architecture [5] 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

Table 5.1: 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

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.

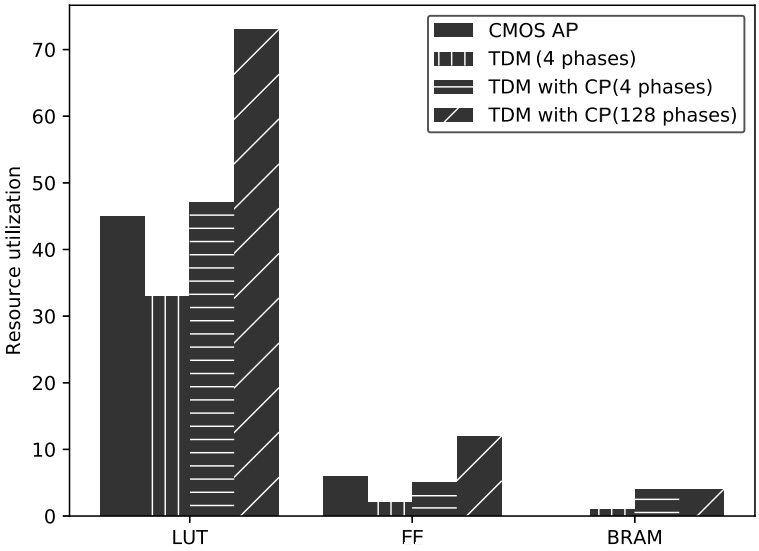


Figure 5.9: Resource Utilization per Spin Unit

Table 5.2: 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

5.4.2 Resource Utilization

Table 5.1 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. 5.9. 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 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 5.2 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.

5.4.3 Performance

Accuracy Comparison

Figure 5.10 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

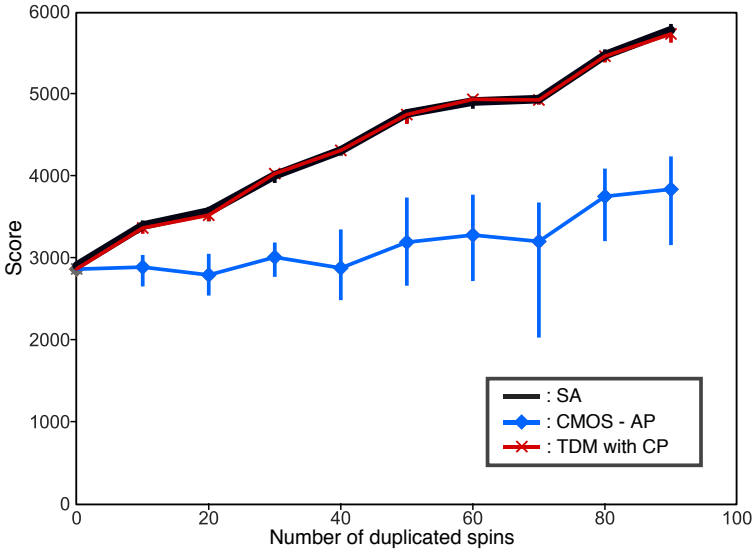


Figure 5.10: Influence of duplicated spins on the score

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

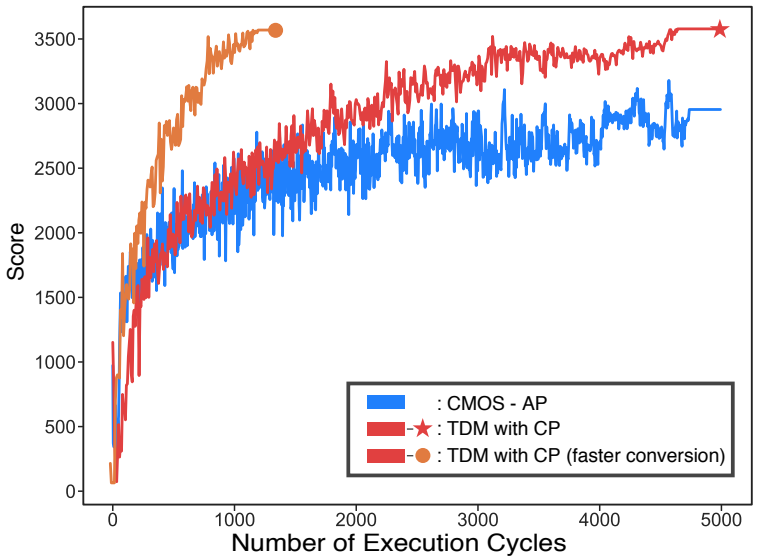


Figure 5.11: Score transition in CMOS annealing and TDM with CP

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 5.11 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 conventional 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

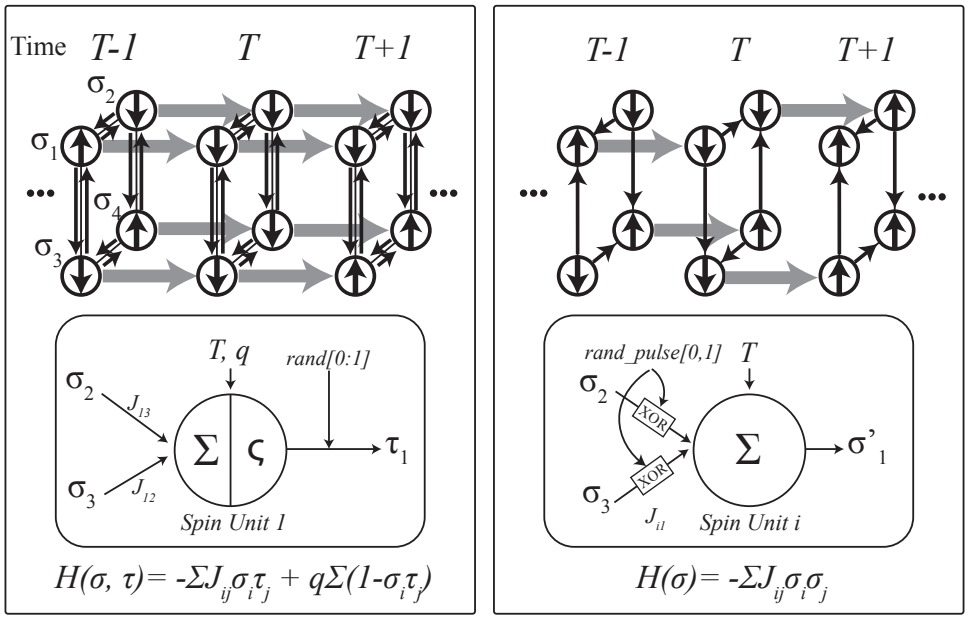
TDM with CP with two types of temperature schedules. One was set so as to converge to a solution with the same execution cycles in both CMOS-AP and TDM with CP. In this work, the annealing schedule is determined according to the number of spins in the problem. The total number of spins in the embedded 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.

5.5 Discussion

This section discusses the issues with the simulated annealing-based time-division multiplexing architecture and suggestions for improvements. When



(a) Stochastic Cellular Automata (b) CMOS Annealing

Figure 5.12: The difference between SCA and CMOS Annealing

simulated annealing is applied to the Ising model, there is a problem that the processing speed is slow because the spin update is a serial update. To solve this problem, CMOS Annealing using a nearest-negighbor connected Ising model to reduce data dependencies has been proposed. When solving a complex problem with a time-division multiplexing architecture based on CMOS Annealing, dependencies on spin updates occur in the connection of the problem graph instead of the host graph. Therefore, when solving a complex problem such as a fully connected graph with a time-division multiplexing architecture, Until one spin is updated, the other spin states cannot be updated. The time-division architecture that requires many phases for one spin update increases the number of required phases and decrease the processing speed.

To solve this problem, we consider a ttime-division multiplexing architecture based on the parallel spin update method. In this disscussion, we adopt stochastic cellular automata (SCA) as a method of parallel spin update. As shown in the figure 5.12, unlike the SA, the SCA can update all spin states

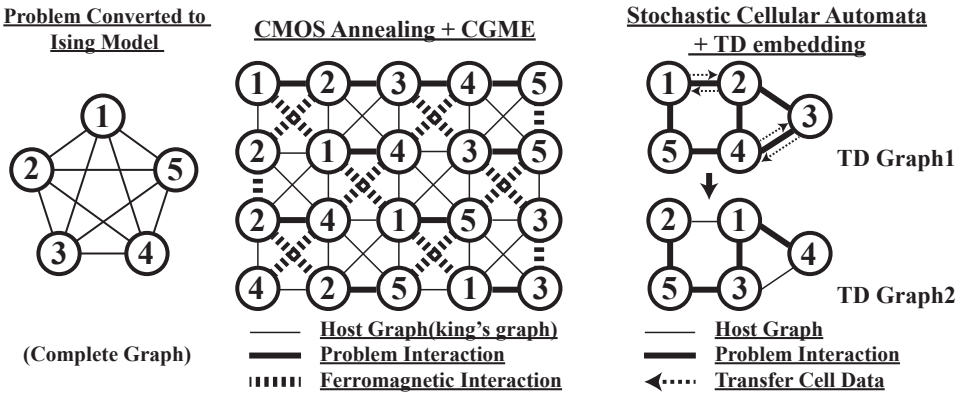


Figure 5.13: The embedding method for multi-layerd Ising model

simultaneously regardless of the shape of the Ising model. Therefore, there is no dependency on the spin update order.

We only need to consider using multiple neareset-neighbor host graphs and embedding to reproduce the connection of the original problem graph.

5.5.1 Time-Division Embedding

Time-division embedding is an extension of the CGME-based algorithm for time-division architectures. In the case of CGME, when embedding the N-spin fully-connected Ising model, a King's graph Ising model of size N times N-1 is created as shown in Figure 5.13. the spin corresponding to each spin unit is arranged as in the following formula.

$$\#(1, x) = x \tag{5.17}$$

$$\#(y, x) = \begin{cases} \#(y-1, max(1, x-1)) \& (x+y\ is\ even) \\ \#(y-1, min(n, x+1)) \& (x+y\ is\ odd) \end{cases} \tag{5.18}$$

This makes it possible to reproduce the fully connected Ising model with the nearest neighbor Ising model.

Unlike CGME, TD embedding is performed using a ladder-like graph as shown on the right of the figure 5.13. The transfer of spin information

between spin units is consistent with the CGME y-direction treated as time. In this embedding, the N-spin nearest neighbor host graph can reproduce the fully connected graph of N-spins with N / 2.

5.5.2 SCA-Based Time-Division Multiplexing Architecture

Figure 5.14 shows the overview of SCA-based architecture. SCA-based architecture also consists of control unit, annealing controller and spin unit array like SA-based architecture (as mentioned in section 4.3).

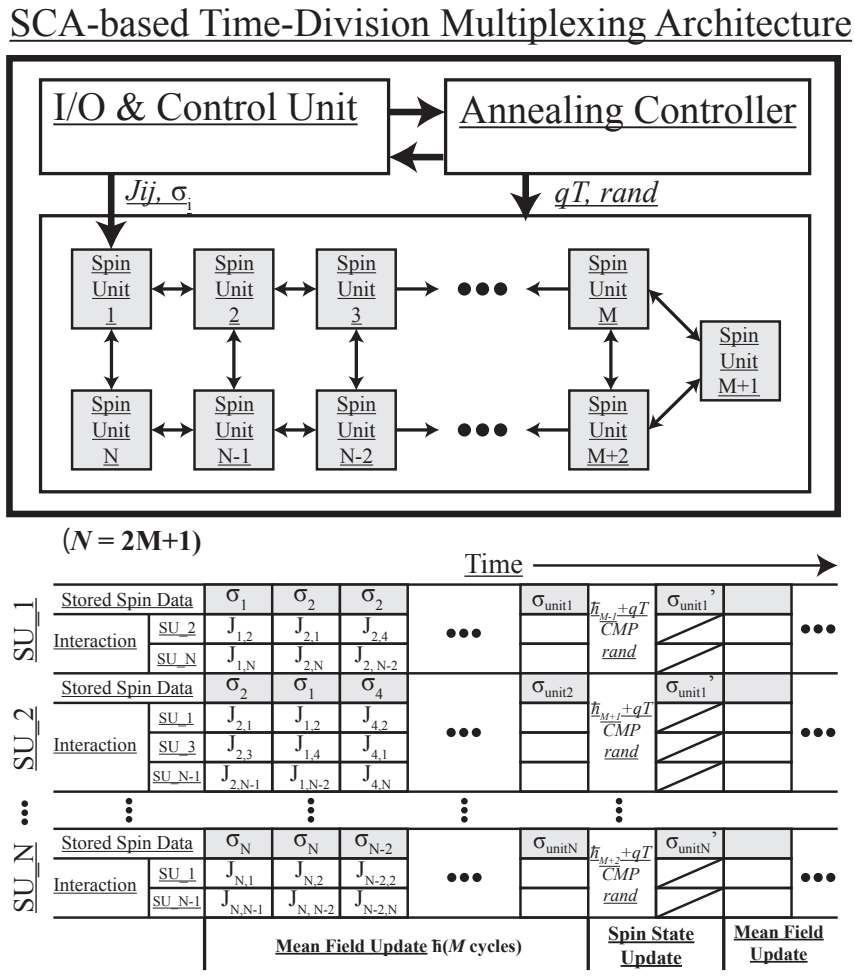


Figure 5.14: SCA-based time-division multiplexing architecture

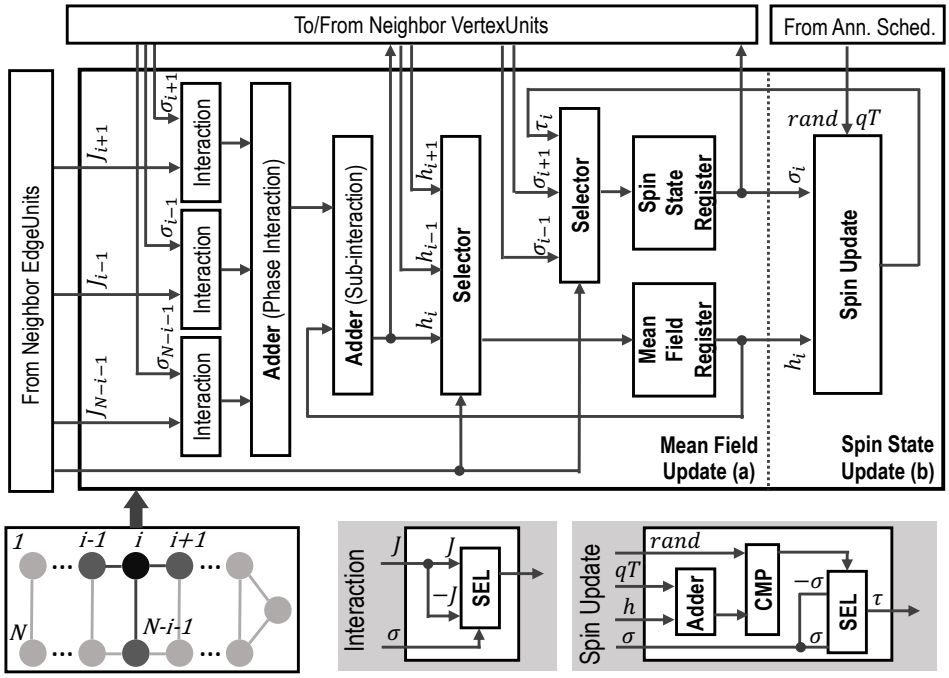


Figure 5.15: Spin unit of SCA-based architecture

Hereafter, different points from SA-based architecture will be described. In annealing controller, it is necessary to control the penalty q in addition to the temperature T. Parameter q is controlled by fixed point multiplication as well as temperature.

In spin unit array in SCA-based architecture, each spin unit is connected like a ladder instead of King's graph in SA-based architecture. In addition, all spin are update at the same time after the Mean field Update as shown in the timing chart of figure 5.14.

This architecture can solve N-spin fully-connected Ising model in N/2 phase with $N^2/2$. As a open research topic, it takes about N / 2 cycles to solve the fully coupled Ising model of N spins in this architecture. In order to speed up this process, it is necessary to reduce the phase including extra connections for complicated graphs that are not completely connected.

5.6 Conclusion

In this research, 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.

Bibliography

[1] Jun Cai, William G. Macready, and Aidan Roy A practical heuristic for finding graph minors CoRR abs/1406.2741 (2014). http://arxiv.org/abs/1406.2741

[2] T. Okuyama, C. Yoshimura, M. Hayashi, and M. Yamaoka Computing architecture to perform approximated simulated annealing for Ising models In 2016 IEEE International Conference on Rebooting Computing (ICRC). 1–8. https://doi.org/10.1109/ICRC.2016.7738673

[3] 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.

[4] 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.

[5] 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.

[6] 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.

Chapter 6

Parallel Spin Update Architecture for Fully-Connected Ising Model

6.1 Introduction

In this chapter, an architecture which conducts parallel SA-type spin updates of fully connected Ising models based on a rigorous mathematical approach, a stochastic cellular automata (SCA) is proposed. The SCA, a generalized version of a probabilistic cellular automata (PCA) sampler [5], prepares replicas τ_x for each spin σ_x and combines them in an SCA Hamiltonian (5.13), which recovers the energy of the Ising Hamiltonian when $\sigma_x = \tau_x$. In this Hamiltonian the τ_x do not interact with each other and can therefore be updated with a spin-parallel Monte Carlo (MC) step. Ultimately, SCA finds the ground state of the Ising Hamiltonian, much faster than SA, by simultaneously increasing the interaction ($q \to \infty$) [which enforces $\sigma_x = \tau_x$] and cooling the system ($T \to 0$) into the lowest energy state.

The main contributions of this research are three-fold: we (1) create an annealing architecture for Ising models with all-to-all connected interactions, named STATICA, which realizes fully parallel spin updates in a single MC step, (2) propose a delta-driven simultaneous spin update (DDSS) scheme that reduces annealing cycles further, and (3) establish light-weight circuits

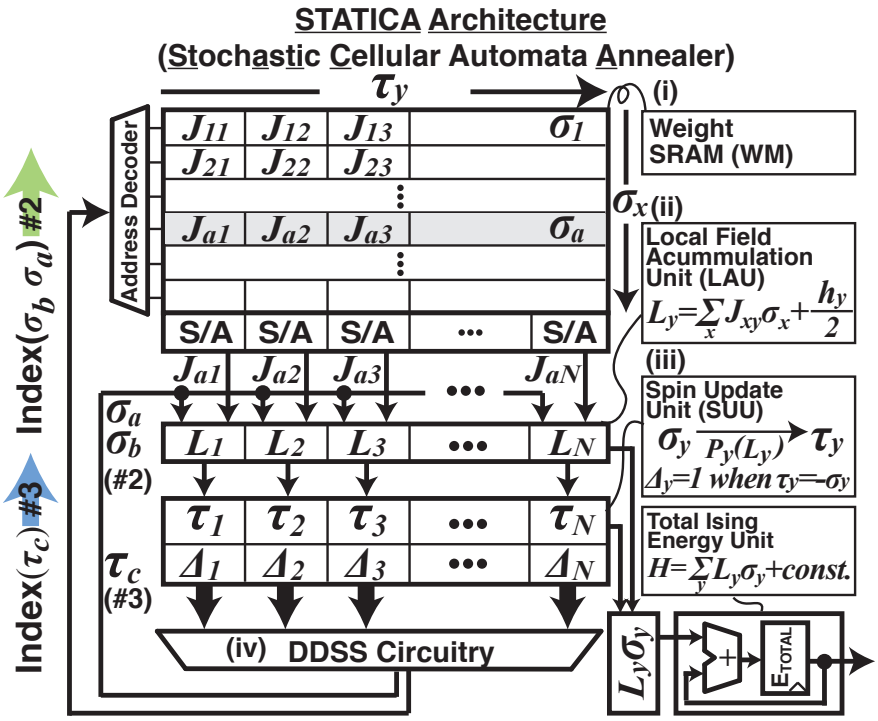


Figure 6.1: STATICA near-memory all-spin-at-once architecture

for STATICA, including key elements for random number generation (RNG) and DDSS. STATICA is also the first silicon annealer that can process fully-connected spins.

6.2 STATICA Architecture

STATICA's datapath architecture (Fig. 6.1) consists of the following blocks that reflect the SCA spin-parallel MC steps: (i) a weight SRAM (WM) that stores interaction weights J_{xy}, where row (column) addresses correspond to σ_x (τ_y), ((ii)) local field accumulation units (LAUs) that compute local fields L_y, ((iii)) spin update units (SUUs) which compute spin flip probabilities $P_y(L_y)$, replica spins τ_y, as well as flip flags Δ_y, and ((iv)) DDSS circuitry. For the initial MC step #1, because initial value in LAUs is zero, all the σ_x (i.e. WM rows) are swept vertically, while L_ys are accumulated in a column-and-pipeline parallel manner in the LAUs (timing diagram, Fig. 6.3). At the end, the SUUs produce τ_y, Δ_y all at once.

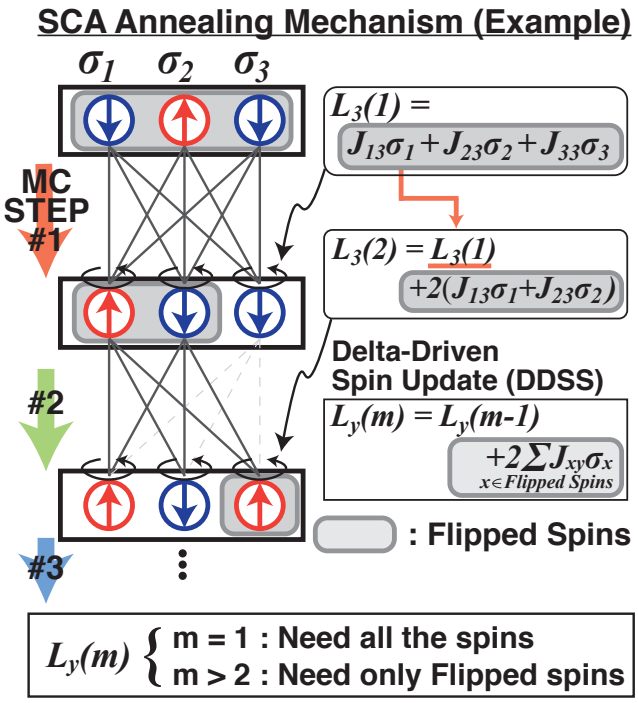


Figure 6.2: the DDSS cycle-reduction scheme.

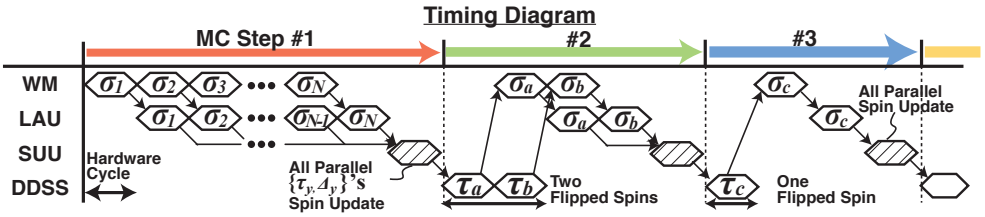


Figure 6.3: STATICA timing chart

STATICA uses the local field update scheme DDSS (Fig. 6.2) to reduce the annealing cycles. As is illustrated in the three-spin example, the local field of the mth MC step $L_y(m)$ is computed by accumulating only the $2J_{xy}\sigma_x\sigma_y$ of the flipped spins into that of the previous MC step as $L_y(m) = L_y(m-1) + 2\Sigma_{y\in flipped}J_{xy}\sigma_x\sigma_y$. In MC step2, σ_1 and σ_2 is flipped in previous step (MC step1). Therefore, local field is updated by using those spins.

Especially, as $(T \to 0)$, where only a few spins flip, DDSS brings significant acceleration, as compared to re-computing L_ys from scratch in every MC step (see also Fig. 6.14). The SUUs use Δ flags to manage flipped spins and the DDSS circuitry reads Δ flags to issue read addresses to the WM and

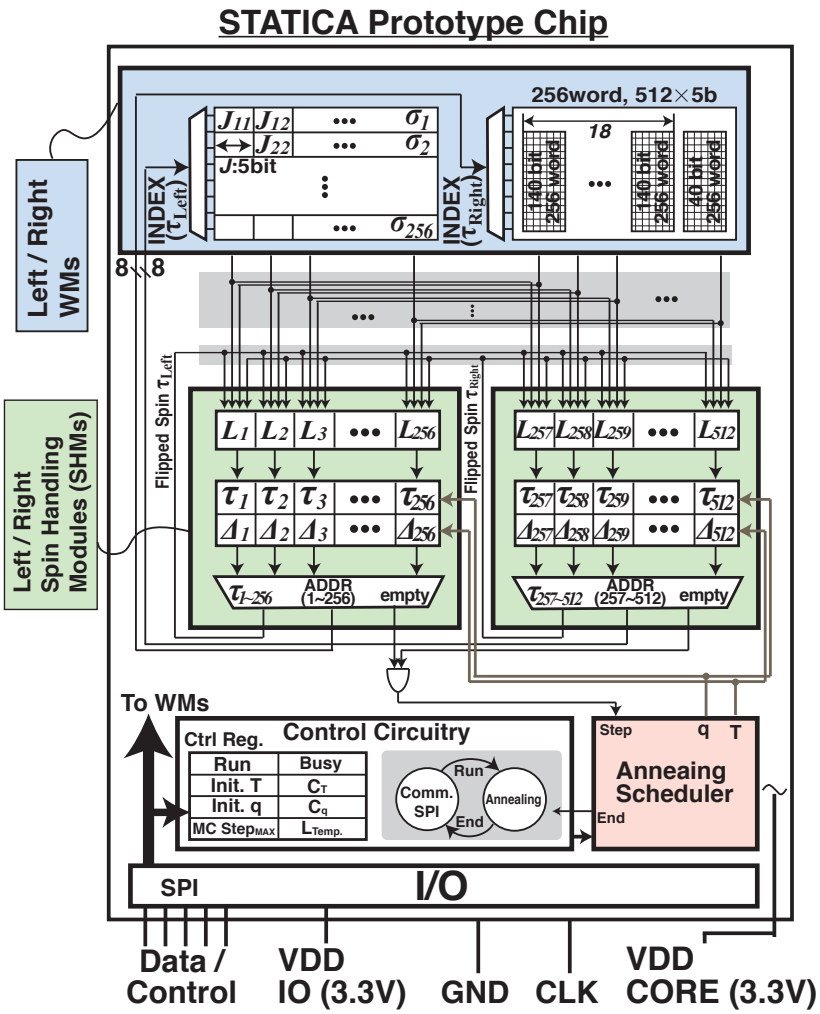


Figure 6.4: A block diagram of a STATICA prototype chip

then provide weights to the LAUs (further details in Fig. 6.8).

The STATICA prototype (Fig. 6.4) is a $3mm \times 4mm$ (65nm CMOS) chip for a 512-spin fully-connected Ising system. It includes an annealing scheduler that controls hyperparameters ($T \to 0, q \to \infty$) as MC steps progress, as well as control circuitry to write weights, start/stop annealing, read spins, etc. The prototype further accelerates the DDSS by processing two flipped spins in parallel with 2 bank SRAM. To this end, the WM, storing 0.25M 5b weights, is divided into identical left/right 640Kb SRAMs. Similarly, the arrays of the LAUs/SUUs are grouped into left/right spin handling modules (SHMs). Weights read from the left (right) WM are multi-cast to both

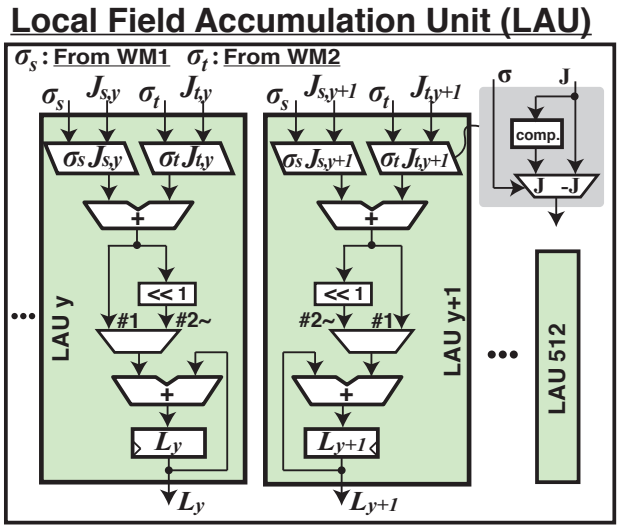


Figure 6.5: A magnified view of the LAU

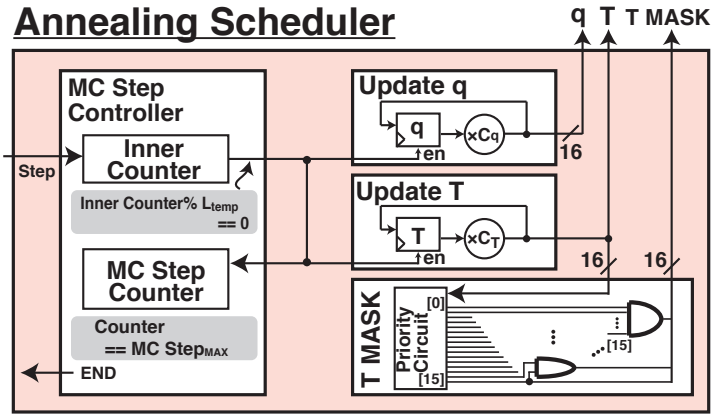


Figure 6.6: A magnified view of the annealing scheduler

SHMs and multiplied with each of the flipped spins from the left (right) block, respectively. Then they are simultaneously summed, doubled, and accumulated on top of the L_ys from the prior MC step (Fig. 6.2).

Fig. 6.5 shows the micro architecture of local field accumulation unit. As mentioned above, two flipped spins are calculated at the same time with 2 bank SRAM in STATICA prototype chip. To calulate the effects of σ_s and σ_t interactions, σ and interaction J multiplication required. Such multiplication is realized as shown in gray part. This multiplication can be calculated without a multiplier by taking the two's complement and selecting a value

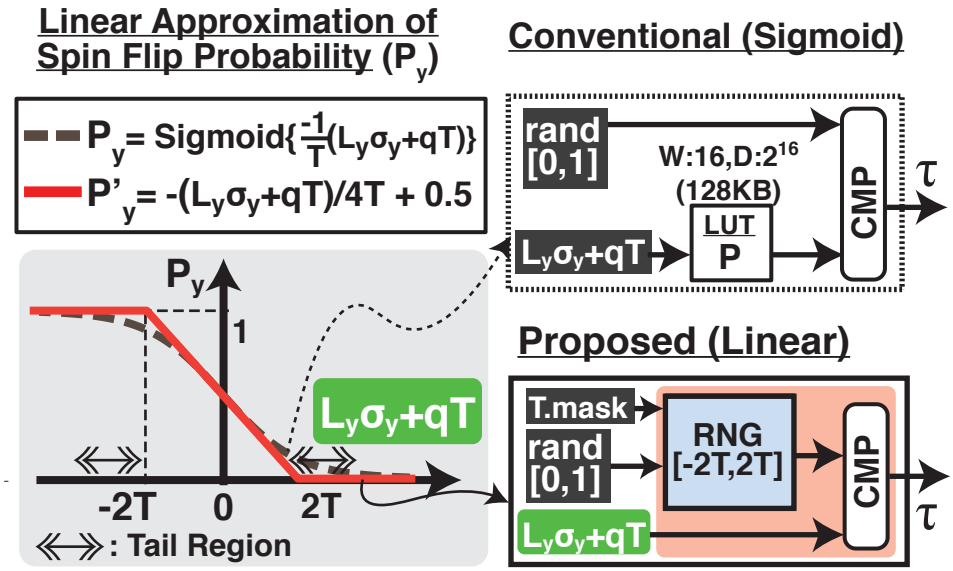


Figure 6.7: Linearized spin-flip determination

based on the value of the spin. After this calculation, two spin interaction is summed and accumlates in accumulator.

Detail of Annealing scheduler is shown in Fig. 6.6. In this module, hyper parameter (q, T) is updated in the timing of $innercounter\%temperaturelength == 0$, where temperature length means the number of MC steps to update the paramter. In addition, the number of Monte Carlo steps is managed to determine the end of annealing. Hyperparamter q and T are kept and calculated at fixed point. T MASK is calculated for random number generator, which indicates priority bit.

To realize light-weight SUUs, STATICA approximates the sigmoidal $P_y(L_y)$ by a piecewise linear function (Fig. 6.7). For $\sigma_x \rightarrow \tau_y$, while a conventional approach would require a look-up table (LUT) whose output is compared with a uniform [0,1] random number, our approach allows for discarding the LUT (128KB) and directly comparing $L\sigma + qT$ with a [-2T, 2T] random number. We share one 32b RNG among two neighboring SUUs and apply 2T-masking for scaling the two 16b independent random numbers to [-2T, 2T]. Altogether, this reduces the SUU area by an order of magnitude. Although RNGs often occupy the major portion of the annealing circuits, our RNG consumes only 11% of the total logic gates since they are present per-

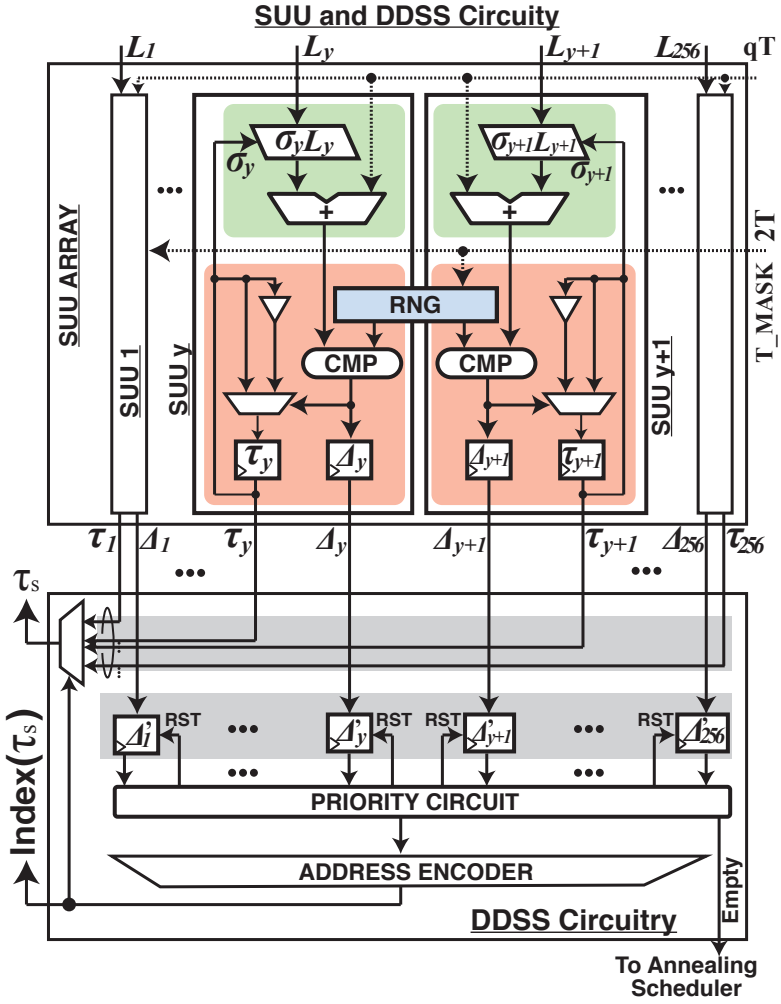


Figure 6.8: (Spin Update Unit SUU) and DDSS circuitry

spin, not per-connection. Finally, the linear approximation of the sigmoid is justified by the low probability ($\leq 10\%$) of having random numbers in the tail region, where both functions exhibit slight deviations (more justification in Fig. 6.14). The DDSS circuitry (Fig. 6.8), on the other hand, generates indices of flipped flags one at a time by way of a priority circuit, then cycles them back as WM read addresses. The flipped spins themselves are broadcasted to all the LAUs as indicated in Fig. 6.8.

In contrast to previous annealers, STATICA can easily compute the Ising energy (Fig. 6.1) which is an important quality metric of the annealing

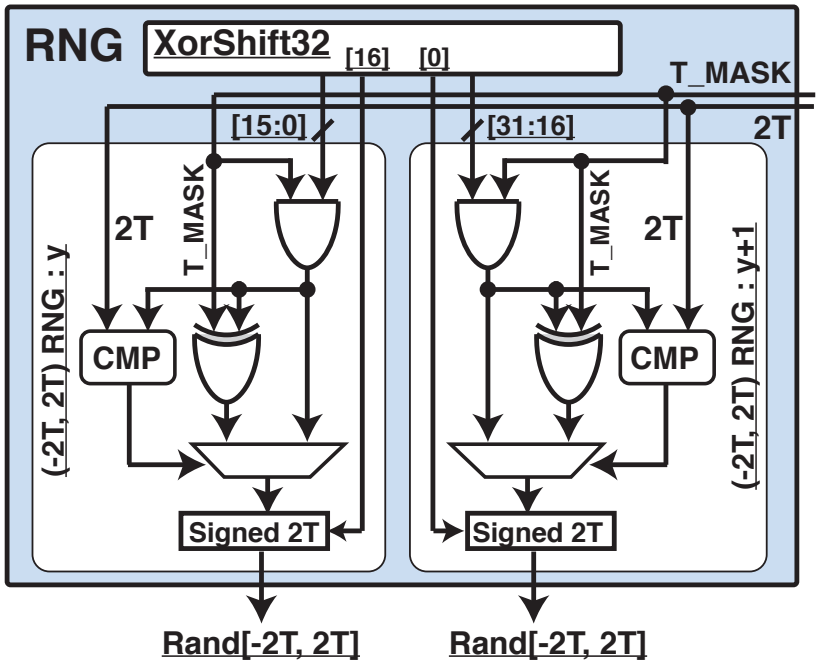


Figure 6.9: A shared Random Number Generator (RNG)

process [using shifts and accumulates of $L_y\sigma_y$s already computed in the LAUs and SUUs]. Hence, STATICA provides an online annealer that can stop annealing autonomously when the quality of the result reaches a predefined threshold.

Fig. 6.9 shows the detail of [-2T, 2T] random number generator. Each random number generator requires only 16-bit precision random numbers. Therefore, two [-2T, 2T] random numbers can be generated from 32-bit precision random numbers generated by XOR-shift. The given 16-bit random number is transformed into 2T + alpha value with T MASK received from the annealing scheduler. If a value exceeding 2T is output, it is inverted to change it to a value within 2T. Finally, a random number is generated the signed value with 1 bit random number of XOR-Shift.

Fig.6.10 shows comparison between conventional fully-connected annealer (FA) and STATICA: FA is based on SA with Glauber dynamics which allows for at most one spin flip per MC step. The algorithm-level parallelism per step is defined by the total number of spins. In each local field accumulate step, LAUs update the local field values in parallel according to the previous

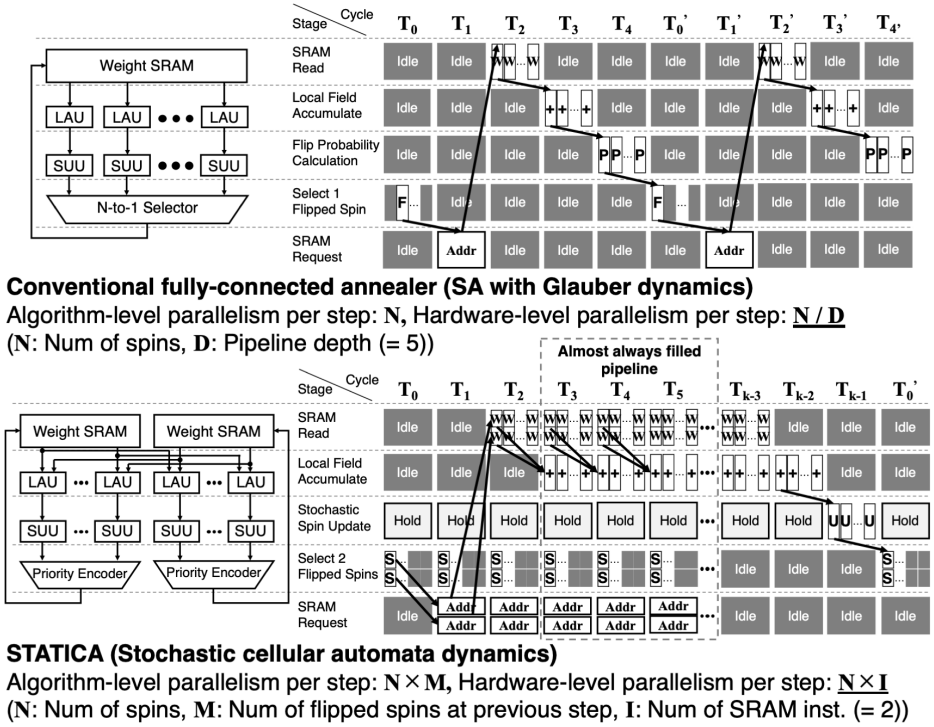


Figure 6.10: The advantage of SCA

flipped spin. In the next step, the flip probability for each spin is calculated in parallel, and flip candidates are determined. Unfortunately, since only one spin from the flip candidates must be selected, the flip probability information of the remaining candidates is finally discarded. The hardware-level parallelism per step is less than the algorithm-level one, due to the computation dependency, namely all micro-operations have a sequential dependency. On the other hand, STATICA is based on SCA (stochastic cellar automata) dynamics which allow multiple spins to flip per step in parallel. The algorithm-level parallelism per step depends on the number of total spins and the number of flipped spins at a previous step, which is generally sufficient to exploit pipeline-level and SRAM-instance-level parallelism. In this prototype chip, two spins are selected from the previous flipped spins for every clock cycle, and the corresponding operations (SRAM access and local field update) are executed in a pipeline manner. Therefore, STATICA achieves a high LAU (PE) utilization rate and high performance.

Name	# Vertices	# Edges	Description	Avg. LAU Util. [%]
G22 [7]	2000	19990	Random-connected sparse graph	92.8
G39 [7]	2000	11778	Planner sparse graph	92.6
K2000 [4]	2000	1999000	Fully-connected graph (slow schedule)	87.6
V512*	512	130816	Fully-connected graph (fast schedule)	89.2
V1024*	1024	523776	↑	94.1
V2048*	2048	2096128	↑	96.1

[7] https://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/306
* These graphs are synthesized by an in-house graph generator for variations of evaluations.

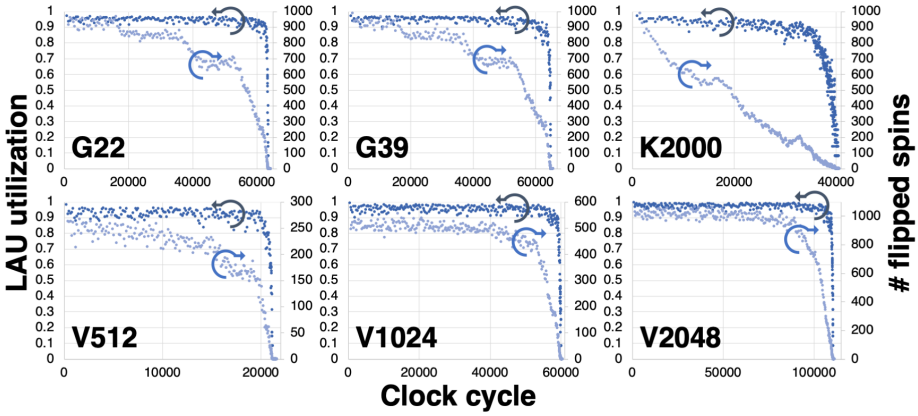
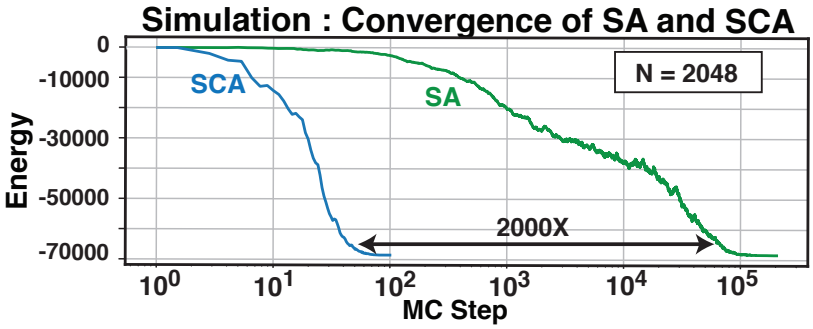


Figure 6.11: Benchmark information and HW utilization for each benchmark

6.3 Evaluation

Top of Fig. 6.11 shows a table of MAX-CUT benchmark graphs and their LAU utilization rates. G22, G39, and K2000 were synthesized using the Rudy graph generator. V512, V1024, and V2048 were synthesized by an in-house graph generator. A MAX-CUT problem for each graph was solved with all spin-spin interactions randomly drawn from +1, -1. Although each graph has a different characteristic, average LAU utilization rates in all the cases are sufficiently high. When compared to the beginning of the annealing, LAU utilization rates become low toward the end of the annealing (in some cases they suddenly drop), because the number of flipped spins at the previous step gets very small. This illustrates why DDSS works well.

As a measurement environment, two source meters (3.3V for clock generator and peripherals, and 1.0V for the STATICA chip) are used. These source meters can measure voltage and current. An external PC assigns all operation parameters, and monitors the behavior of STATICA via SPI. All



Simulation : Sigmoidal vs. Linear Transition Probability
[Max-cut problem score≈ -2 Ising energy]

	N = 512			N = 1024		
	Best	Ave.	Worst	Best	Ave.	Worst
Sigmoid	4584	4514	4380	12558	12355	12104
Linear	4584	4532	4437	12591	12449	12270

Blue is better, Red is worse

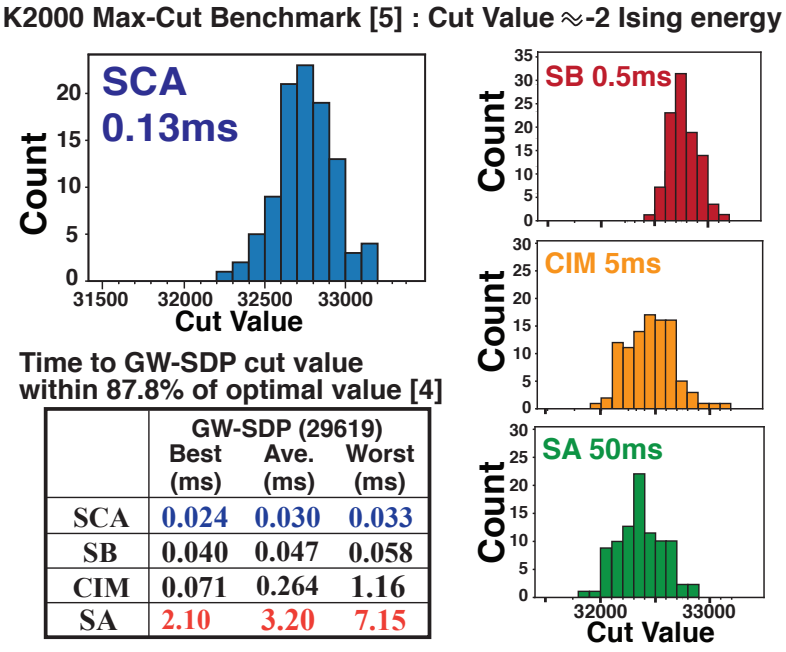
Figure 6.12: Evaluation results: performance of SCA and STATICA and validity of linear approximation

measured results are verified by comparisons with the corresponding expected values on the PC.

In the top of Fig. 6.12, we evaluate STATICA. The typical convergence behavior of SA and SCA on an N=2048 spin Max-Cut problem is compared as a function of MC steps. SCA' s parallel spin update finds a solution about N×faster. This is because SA can update only one spin in one monte carlo step in the serial update, while SCA can update all spins in parallel in one monte carlo step. Therefore, SCA is fundamentally faster than SA. In addition, SCA converges to the solution of SA by increasing q large enough.

The lower diagram in Fig. 6.12 shows the max-cut score on an N = 512 and N = 2048 problem with 100 trials. In the both benchmark, It can be seen that the SCA including the piecewise linear approximation always obtains a better solution than the SCA using the sigmoid function. At N = 512, both methods obtain optimal solutions. Simulations confirm that approximating sigmoidal update probabilities by a linear function does not change the solution quality.

STATICA' s superior annealing speed and accuracy is demonstrated by extending the SA, CIM, and SB K2000 Max-Cut benchmark. SB, CIM, SA



	GW-SDP (29619) Best (ms)	Ave. (ms)	Worst (ms)
SCA	0.024	0.030	0.033
SB	0.040	0.047	0.058
CIM	0.071	0.264	1.16
SA	2.10	3.20	7.15

Figure 6.13: Comparison with other fully-connected solutions

results refer to data in SB papers [3]. In terms of accuracy, histograms shows that the SCA cut value is higher than other methods. In other words, the higher the cut value, the better the quality of the solution, indicating that STATICA converges to a good solution at the highest frequency. The top left of each histogram shows the average convergence time for each method. it can be seen that SCA is also superior to other methods in speed. STATICA's cut values show the best performance, indicating superior accuracy in annealing time of 0.13ms, i.e., 3.8× faster than SB, as estimated based on a critical path analysis of 65nm 2K-spin STATICA@300MHz. The lower right part of the figure shows the convergence time of each method until reaching the solution (87.8%) of K2000 calculated by GW-SDP. In this case as well, we can see that STATICA reaches at the solution faster than other methods. In this case, the reason that the difference from the other methods is small is that when reaching the solution of GW-SDP, is that the solutions with hardware are in the middle of annealing and STATICA consumes only few cycles in the late stage of annealing.

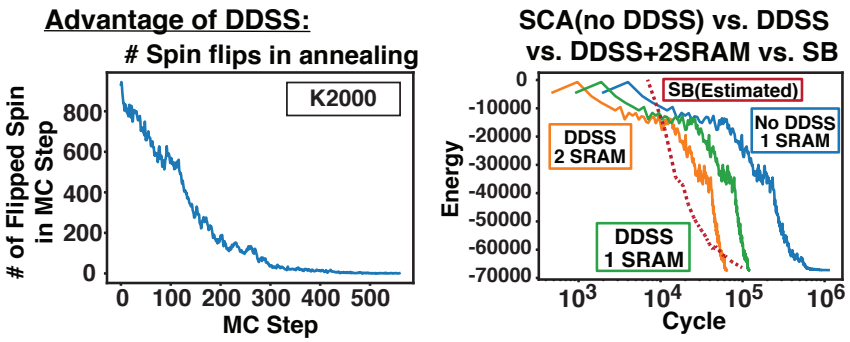


Figure 6.14: Advantage of DDSS

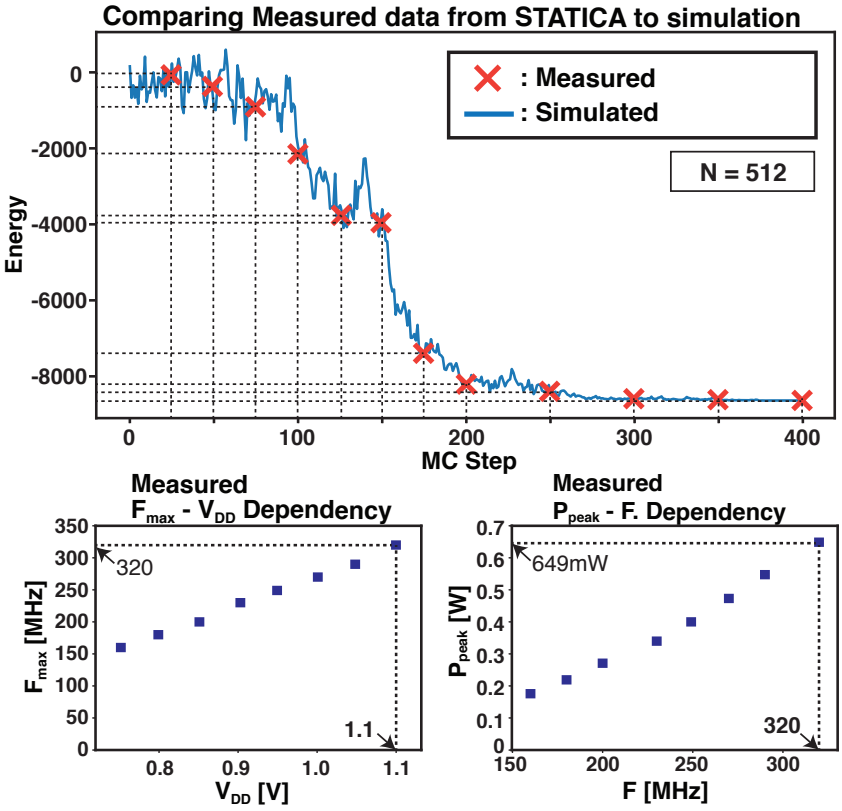


Figure 6.15: Confirmation of chip operation with 320MHz@1.1V.

Next, we illustrate the advantage of DDSS. [left] Since the number of flipped spins decreases as a function of MC steps [right] including DDSS into STATICA reduces the number of required cycles by an order of magnitude. Cycles are further reduced by parallel processing of two updated spins.

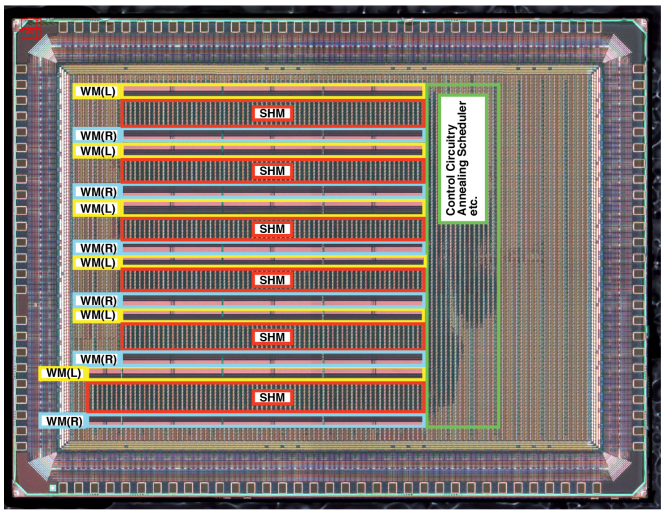
We confirm the correct operation of the STATICA chip by comparing

		D-wave [1]	CMOS Anl. ISSCC'19 [2]	Dig. Anl. [3]	CIM [4]	Sim. Bifur. [5]	This work
Technology		Quantum Circuits	CMOS Chip (40nm)	FPGA	Optics (Laser+FPGA)	FPGA (Arria 10 GX)	CMOS Chip (65nm)
Annealing Method		Quantum Annealing @20mKV	Simulated Annealing	Simulated Annealing	Quantum Adiabatic Bifurcation	Simulated Bifurcation	SCA Annealing
Topology (Implication)		Nearest-Neighbor (Needs graph embedding: Limited Applications)		Fully-Connected (Native graph mapping: Rich Applications)			
# of Spins		2K (qubit)	30K/Chip	1028	2K	2K	512
# of Weights		26K	240K/Chip	1056K	4M	4M	262K
Precision [bit]		N/A	3	16	1	1	5
Op. Freq. [MHz]		N/A	100	100	N/A	229 - 248	320
Voltage [V]		N/A	1.1	N/A	N/A	0.87 - 0.98	1.1
Op. Power [W]		25k	0.05	N/A	N/A	~40	649m
Chip Size [mm²]		N/A	4.3×5.5	N/A	N/A	N/A	3.0×4.0
V512	Time [µs]	N/A	N/A	N/A	N/A	N/A	9.4 (Measured)
	Energy [µJ]				N/A	N/A	6.1 (Measured)
K2000	Time [ms]	N/A	N/A	N/A	5	0.5 (Measured)	0.13 (Estimated)
	Energy [J]				N/A	20m (Estimated)	<1µ (Estimated)

Figure 6.16: Comparison with state-of-the-art annealers: STATICA is the first custom processor chip for fully-connected Ising spins. Among the fully-connected family, it has the fastest annealing speed and lowest energy consumption per annealing run.

the measured Ising energy as a function of MC steps to numerical simulations. Both data are identical. As the measured Fmax-VDD and Ppeak-F curves show, the STATICA prototype operates at 320MHz@1.1V, dissipating 629mW peak power.

Fig. 6.16 compares STATICA with prior art. First, the sparse connectivity of quantum and CMOS annealers limits their utility and their equivalent counts for fully connected spins are inferior. Further, comparing fully-connected architectures is not straightforward, since STATICA is the first silicon prototype. Yet, while exact numbers have not been published, we expect that DA is at least two orders of magnitude slower and consumes more power due to the serial spin updates. On the other hand, STATICA clearly



Technology	TSMC 65nm CMOS
Chip Size	12mm²
Core V_{DD}	1.1V
I/O V_{DD}	3.3V
Frequency	320MHz
Power	649mW
Gate Count	336K
SRAM	256word,140bit×18 256word,40bit×2

Figure 6.17: STATICA prototype chip

outperforms SA, CIM, and SB (Fig. 6.13).

Figure 6.17 shows a STATICA prototype microphotograph and a specification table. Note that increasing STATICA' s spin count in the future is straightforward as the chip size (process node) becomes larger (finer), since SHMs scale linearly with the number of spins (not the number of connections), and also since STATICA' s critical path is determined by the WM read cycle.

6.4 Conclusion

In summary, STATICA aims at revolutionizing annealing processors by solving optimization problems based on the mathematical model of SCA, facilitating three innovations: (1) A parallel, near-memory architecture for fast and monitorable annealing of all-to-all spin connections. (2) Accelerated DDSS field updates removing redundant computations based on flipped spins. (3) New light-weight RNGs. Initial evaluations provided strong results.

Bibliography

[1] T. Takemoto et al. A 2 ×30k-Spin Multichip Scalable Annealing Processor Based on a Processing-In-Memory Approach for Solving Large-Scale Combinatorial Optimization Problems ISSCC, pp. 52-53, 2019.

[2] S. Tsukamoto, M. Takatsu, S. Matsubara, and H. Tamura Anaccelerator architecture for combinatorial optimization problems 2017

[3] H. Goto et al., Combinatorial Optimization by Simulating Adiabatic Bifurcations on Nonlinear Hamiltonian Systems, Science Advances, vol. 5, no. 4, 2019.

[4] K. Tatsumura, A. R. Dixon, and H. Goto FPGA-based Simulated Bifurcation Machine 2019 29th International Conference on Field Programmable Logic and Applications (FPL)

[5] Dai Pra, Paolo, Benedetto Scoppola, and Elisabetta Scoppola Sampling from a Gibbs Measure with Pair Interaction by Means of PCA Journal of Statistical Physics, vol. 149, no. 4, pp. 722–737, 2012

Chapter 7

Conclusion

Combinational optimization is a fundamental and practical method to describe and solve various social problems in our daily lives, such as transportation cost optimization, machine learning, drug discovery and so on. The goals of that problem are to find the best combination that maximize or minimize the objective function defined by user. The difficult point to solve such optimization problems is that they are known as NP, so that they can be resolved in a polynomial time. Even if approximate solutions via heuristic approaches are allowed, it takes certainly long computing time to solve them due to their iterative searches of optimal solution points.

In order to overcome inefficiency of the modern von Neumann computers for such optimization problems, an Ising computer has been proposed based on the "natural computing" paradigm which maps a target problem onto a physical model in nature and observes the obtained status of the physical matters as its computing result. Unlike conventional computers, natural computing is based on the mechanism that the computing state eventually reaches the state of lowest energy. The annealing processor, based on the Ising model, is one approach to natural computing and exploits the fact that combinatorial optimization problem can be mapped to the ground state search of Ising model.

Based on these, the issues of existing annealing processors in Chapter 3.4 are as follows.

1. A disadvantage of the conventional CMOS-AP is that there are a lim-

ited number of spin counts that can be deployed due to hardware resource constraints.

2. In the annealing processors which has the nearest-neighbor Ising model, the accuracy and convergence speed of the solution decrease if there is a mismatch between the hardware Ising model and the problem.

3. When calculating the ground state of the fully coupled Ising model by simulated annealing, the number of update spins is limited to one in principle. Therefore, high parallelism cannot be obtained unlike the nearest-neighbor annealing processor.

In chapter 4, the method of increasing the number of spins based on the nearest neighbor type was studied based on the CMOS Annealing machine, the first CMOS-type annealing processor. In CMOS annealing, since a distributed memory is used to maintain spin states and spin-spin interactions, there was a problem that the number of LUTs became a bottleneck and the scalability of spin numbers was low. In this research, we focused on the drawbacks and created an architecture with a time-division processing mechanism that is presumed to be stored in a higher density block RAM of FPGA. As a result, it was shown that a loosely coupled Ising network formed with a larger number of spins than the conventional method can be implemented at high density.

In chapter 5, based on the architecture for the large-scale combinatorial optimization problem described above, we researched a method to solve the problem that the class of the problem which the nearest neighbor type has is limited in this research. In the case of a loosely coupled Ising network, if there is a mismatch between the network in problem and the hardware graph, it is necessary to perform a graph transformation called embedding processing to match the shape of the Ising model of the annealing processor and the network of the optimization problem. However, since a spin having a strong ferromagnetic connection which does not originally occur due to the graph deformation is added, it is considered that the accuracy of the annealing and the convergence speed are adversely affected. Therefore, in this research, based on the time-division architecture of (1), we developed a

technology to virtually increase the complexity of the Ising network on the hardware, and a class of problems that can be addressed without lowering the accuracy and speed We considered a method to increase the number. As a result, we obtained a result that it is possible to eliminate the decrease in the accuracy and the convergence speed of the annealing due to the embedding process for the problem that the accuracy is reduced in the conventional nearest neighbor Ising network.

However, when trying to reproduce a fully coupled Ising network using a loosely coupled Ising model using a time-sharing architecture, the number of divided networks increases in the time direction to resolve data dependencies, and processing speed decreases. Therefore, a new annealing technique that samples the spin state and searches for the ground state using a stochastic cellular automaton that can update the spin state in parallel even in the fully connected Ising model, and an algorithm that reproduces the fully coupled Ising network with a loosely coupled Ising network Researched. This study shows that the ground state search of a fully coupled Ising network can be realized at high speed with a small number of divisions using a loosely coupled Ising model and a time division architecture.

In chapter 6, I conducted a study on a fully coupled architecture based on the parallel update annealing described above. This architecture solves the problem that if all spins are updated in parallel, many product-sum operations are required for each update. It is possible to obtain the ground state at a higher speed as compared with. In addition, a dedicated circuit was created based on the architecture that was actually created, and the superiority of other annealing processors in terms of convergence speed and power consumption was actually confirmed.

As described above, the research described in this paper contributes to the improvement of the efficiency of the annealing processor for solving the large-scale combinatorial optimization problem from both the fundamental algorithm and the hardware architecture. I hope that this research will solve the combinatorial optimization problem that is expected to increase in demand in the future and contribute to the development of human society.

Acknowledgements

I want to thank Professor Takamaeda-Yamazaki Shinya at the University of Tokyo, Professor Masato Motomura at Tyokyo Institute of Technology and Professor Tetsuya Asai at Hokkaido University for his advice during the writing of this thesis and pursuing my Ph.D. research.

I am also deeply grateful to Professor Kazuhisa Sueoka, Professor Seiya Kasai and Professor Masayuki Ikebe at Hokkaido University for their advice and fruitful discussions.

Dr. Takashi Takemoto at Hitachi, Dr. Mertig, Normann at Hitachi, Professor Hiroshi Teramoto, Professor Hiroki Arimura and Professor Akira Skai provided me with a lot of suggestions on the research.

I also would like to thank the members of my research team and lab mates: Kodai Ueyoshi, Miho Ushida, Dr. Eric-Shun Fukuda, Dr. Kazuyoshi Ishimura, Dr. Itaru Hida, Takanori Ohira, Masafumi Mori, Tsunaki Sadahisa, Toshiyuki Itou, Kentaro Orimo, Kota Ando, Kazutoshi Hirose, Weiqiang Huang, Teruaki Kumazawa, and many others for supporting my research in lab.

This work was supported by JST CREST Grant Number JPMJCR18K3, Japan.

List of Publications

1. Journal Papers

1. K. Yamamoto, M. Ikebe, T. Asai, M. Motomura, S. Takamaeda-Yamazaki, "FPGA-based annealing processor with time-division multiplexing, " IEICE Trans. Inf.& Syst., E102-D(12), 2019.

2. K. Yamamoto, M. Ikebe, T. Asai, M. Motomura, "FPGA-based stream processing for frequent itemset mining with incremental multiple hashes," Circuits Syst., 7(10), 3299-3309, 2016.

2. International Conferences

1. K. Yamamoto, K. Ando, N. Mertig, T. Takemoto, M. Yamaoka, H. Teramoto, A. Sakai, S. Takamaeda-Yamazaki, M. Motomura, "STATICA: A 512-Spin 0.25M-Weight Full-Digital Annealing Processor with a Near-Memory All-Spin-Updates-at-Once Architecture for Combinatorial Optimization with Complete Spin-Spin Interactions," 2020 International Solid-State Circuits Conference (ISSCC 2020), San Francisco Marriott Marquis, San Francisco, US (Feb. 16-20, 2020).

2. K. Yamamoto, W. Huang, S. Takamaeda-Yamazaki, M. Ikebe, T. Asai, M. Motomura, "A Time-Division Multiplexing Ising Machine on FPGAs," International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies (HEART 2017), Bochum, Germany Jun. 7-9, 2017.

3. K. Yamamoto, S. Takamaeda-Yamazaki, M. Ikebe, T. Asai, M. Motomura, “A scalable ising model implementation on an FPGA,” COOL Chips 20, Yokohama, Japan Apr. 19-21, 2017.

4. K. Yamamoto, T. Asai, M. Motomura, “Hardware architecture for online frequent items mining with memory-efficient data structure,” COOL Chips XIX, Yokohama, Japan Apr. 20-22, 2016.

5. K. Yamamoto, E.S. Fukuda, T. Asai, M. Motomura, “An accelerator for frequent Itemset mining from data stream with parallel item tree,” The 19th Workshop on Synthesis And System Integration of Mixed Information Technologies, Yilan, Taiwan Mar. 16-17, 2015.

3. Domestic Conferences (in Japanese)

1. 山本 佳生, 熊澤 輝顕, 池辺 将之, 浅井 哲也, 本村 真人, 高前田 伸也, ”高次数イジングネットワークの時分割処理方式の検討,” 電子情報通信学会コンピュータシステム研究会 (CPSY), 秋田アトリオンビル, (秋田), 2017年7月26-28日.

2. 山本 佳生, 高前田 伸也, 池辺 将之, 浅井 哲也, 本村 真人, ”時分割多重機構を用いた高密度FPGAイジングマシン,” 電子情報通信学会コンピュータシステム研究会 (CPSY), 登別温泉第一滝本館, (登別), 2017年5月23日.

3. 山本 佳生, 池辺 将之, 浅井 哲也, 本村 真人, 高前田 伸也, ”時分割多重機構を用いたイジングプロセッサの解精度向上手法の検討,” LSIとシステムのワークショップ2017, 東京大学, (東京), 2017年5月15-16日.

4. 山本 佳生, 定久 紀基, 浅井 哲也, 本村 真人, ”FPGAによる多重ハッシュを用いた頻出アイテムセットマイニングのストリームプロセッシング,” 電子情報通信学会リコンフィギャラブルシステム研究会, 富士通研究所, (川崎), 2016年5月19-20日.

5. 山本 佳生, 定久 紀基, 金 多厚, 福田 駿 エリック, 浅井 哲也, 本村 真人, ”頻出アイテムセットマイニング高速化のためのストリーム

プロセッサ,” LSIとシステムのワークショップ, 北九州国際会議場, (北九州市), 2015年5月11-13日.

4. Domestic Conferences (Co-authored, in Japanese)

1. 定久 紀基, 山本 佳生, 浅井 哲也, 本村 真人, ”二重ハッシングによる類似検索ハードウェアアーキテクチャのFPGA実装,” LSIとシステムのワークショップ, 北九州国際会議場, (北九州市), 2015年5月11-13日.

2. 定久 紀基, 山本 佳生, 金 多厚, 福田 駿 エリック, 浅井 哲也, 本村 真人, ”Locality-Sensitive Hashingのスケーラブルなハードウェアアーキテクチャ のFPGA実装,” 電子情報通信学会総合大会, 立命館大学びわこ・くさつキャンパス, (草津), 2015年3月10-13日.

3. 定久 紀基, 山本 佳生, 金 多厚, 福田 駿 エリック, 浅井 哲也, 本村 真人, ”類似検索を行うLocality-Sensitive Hashingのスケーラブルなハードウェアアーキテクチャ,” 電子情報通信学会集積回路研究会・コンピュータシステム研究会合同 平成26年度若手研究会, 機械振興会館, (東京), 2014年12月1-2日.