An Accelerator Architecture for Combinatorial Optimization Problems

● Sanroku Tsukamoto ● Motomu Takatsu ● Satoshi Matsubara
● Hirotaka Tamura

In today's world, there are many situations in which difficult decisions must be made under such constraints as a limited resource and a limited amount of time. These situations include disaster response planning, economic policy decision-making, and investment portfolio optimization. In such situations, it is often necessary to solve a "combinatorial optimization problem," which involves evaluating different combinations of various factors and selecting the optimum combination. Since the number of combinations increases explosively as the number of factors increases, it becomes difficult to find the best answer in a realistic amount of time using a von Neumann type processor. To give a solution for such problems, we have developed two schemes to speed up the 1024-bit Ising model and implemented them in a field-programmable gate array (FPGA). Testing demonstrated that a system using this architecture can solve a 32-city traveling salesman problem 12,000 times faster than the same algorithm running on a 3.5-GHz Intel Xeon E5-1620 v3 processor.

1. Introduction

Moore's law, which has predicted the growth of computer systems over the last 50 years by updating the performance approximately 100-fold per decade, will end within another decade.[1),2)] This implies that something innovative computer architecture is needed to maintain the progress without relying on semiconductor device performance (**Figure 1**).

The architecture should feature scaling-independent performance, power efficiency, and high speed.[3)] For example, an architecture optimized for a specific domain by using general-purpose computing on graphics processing units (GPGPUs) or field-programmable gate arrays (FPGAs) would be widely accepted for the next ten years. After that, a non von-Neumann architecture will be applied to suppress data transfer and memory access to achieve power efficient systems. Neural networks are expected to be used in future computing systems because of their superior power efficiency. Quantum computers and coherent computers are expected to be applied to NP-hard problems, which are basically combinatorial optimization problems, because they are much faster than ones based on the conventional von-Neumann architecture.

The search for ways to revolutionize optimization computations in the post-Moore era, when von-Neumann architectures may fail to provide efficient solutions has led to increased interest in quantum-annealing (QA) hardware,[4)] coherent computing,[5)] and neural networks.[6),7)]

However, whether and for which problem instances QA or its classical counterpart performs

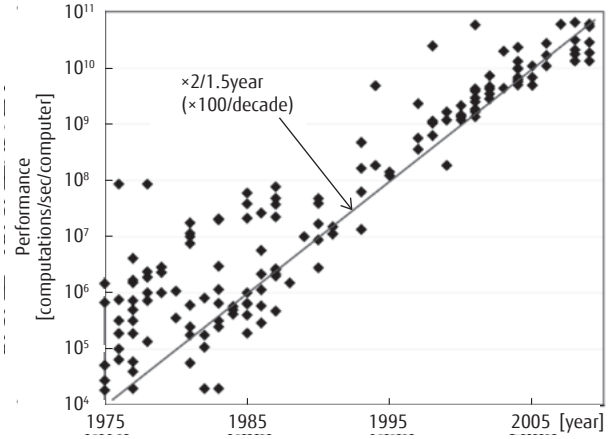

Figure 1
Trend in computer performance.

better than conventional processors are still under debate. Answering these questions and achieving meaningful processor speedups by using system- and architecture-level innovations are of great importance to our ultimate goal, which is to enhance performance so that the ever-increasing amount of data can be handled with reasonable cost performance.

In this paper, we present an architecture for optimizing a fully connected 1024-bit Ising model implemented in an FPGA. A system using this architecture solved a 32-city traveling salesman problem 12,000 times faster than a simulated annealing program running on a 3.5-GHz Intel Xeon E5-1620 v3 processor. This speedup can be increased by a factor greater than 100 in the near future by using a hierarchy of speed-up methods, ranging from ones for an optimization engine to those for a multi-ensemble system[8] composed of a multiple of such engines, each on a custom-designed IC.

2. Operating principle

The architecture we propose consists of a PC and multiple engines, each of which performs a Markov-chain Monte Carlo (MCMC) stochastic search and thereby minimizes the Ising energy:

$$E(X) = -\sum_{\{i,j\}} W_{ij} x_i x_j - \sum_i b_i x_i ,$$
$$x_i \in \{0,1\}\ (i=1,2,\cdots,N),\ W_{ij} = W_{ji} , \quad (1)$$

where $x_i \in \{0,1\}$ is the state variable or bit, N the number of bits, W_{ij} the connection weight between x_i and x_j, and b_i the bias term (**Figure 2**). The operating cycle is divided into two phases, a trial phase in which a state-variable change that meets an acceptance criterion is selected and an update phase in which the selected variable is flipped and relevant signals that depend on the variable are updated accordingly.

In the trial phase, in each neuron i that generates state variable x_i, the increment in energy E when the state moves to a neighboring state $X^{(i)}$ is calculated. Neighboring state $X^{(i)}$ is generated from current state $X=(x_1, x_2, \cdots, x_N)$ by flipping state variable x_i to $1-x_i$. The resulting energy $E(X)$ increment is given by

$$\Delta E_i = -(1-2x_i)\, h_i , \quad (2)$$

$$h_i = \sum_j W_{ij} x_j + b_i , \quad (3)$$

where h_i is the local field of the i-th neuron. We use an architecture in which at most one state variable changes its value during the update phase, and local field values h_i (i=1,2,⋯,N) are stored in registers (**Figure 3**). When a state variable is updated, each local

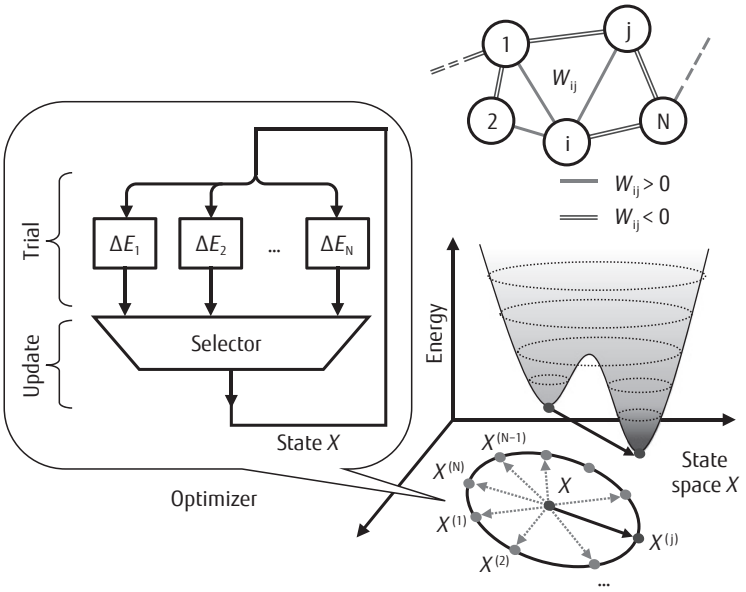

Figure 2
Optimizing Ising model using parallel trial scheme.

field is updated accordingly by adding its increment δh_i to its current value. For example, when variable x_j is updated to $1-x_j$ in the update phase, the resulting δh_i is

$$\delta h_i^{(j)} = W_{ij}(1-2x_j) \ . \qquad (4)$$

In our design, the number of bits N on a chip is 1024. These 1024 bits are fully connected; i.e., any two bits, x_i and x_j, among the 1024 bits can be connected with weight W_{ij}, independent of other bit-pair connections. The weight values are expressed in 16-bit fixed-point signed binary code, the local field h_i in 27-bit signed binary, and the bias term b_i in 26-bit signed binary. The bias term is implicitly given by setting the initial values of h_i and x_i to satisfy Eq. (3).

The criterion used in the trial phase is selectable from either Metropolis-Hastings or Gibbs ones:

$$A(\Delta E_i) = \begin{cases} \min[1, \exp(-\beta \Delta E_i)] & (\text{Metropolis-Hastings}) \\ 1/[1+\exp(\beta \Delta E_i)] & (\text{Gibbs}) \end{cases}, \qquad (5)$$

where $A(\Delta E_i)$ is the acceptance probability of x_i flipping to $1-x_j$, and β $(=1/T)$ is the inverse temperature. In a parallel trial scheme, an acceptance decision block (ADB) in each neuron compares the value of ΔE_i with a numerical noise value to produce a binary flag that becomes "1" with the probability given by Eq. (5). The numerical noise is generated by a table lookup from a random number r_i uniformly distributed between 0 and 1 so that the table outcome is $A^{-1}(r_i)$ (Figure 3). The resulting flag bit indicates whether the corresponding state variable is a candidate that would change its value if selected.

3. Acceleration scheme 1: parallel trials

The update selector selects a single state variable from the state variables having a flag value of "1." It then generates a flag indicating whether there is a candidate bit or not and an index for the selected variable. Ten stages of two-to-one selectors perform the selection and the flag and index generation (**Figure 4**). If both inputs of the two-to-one selection stage are eligible for the next flip, one of them is randomly selected using a random binary number. If there is no candidate for the state-variable change, the last-stage selector produces a flag value of "0," and no update is made in the next update phase. The index of the selected candidate is

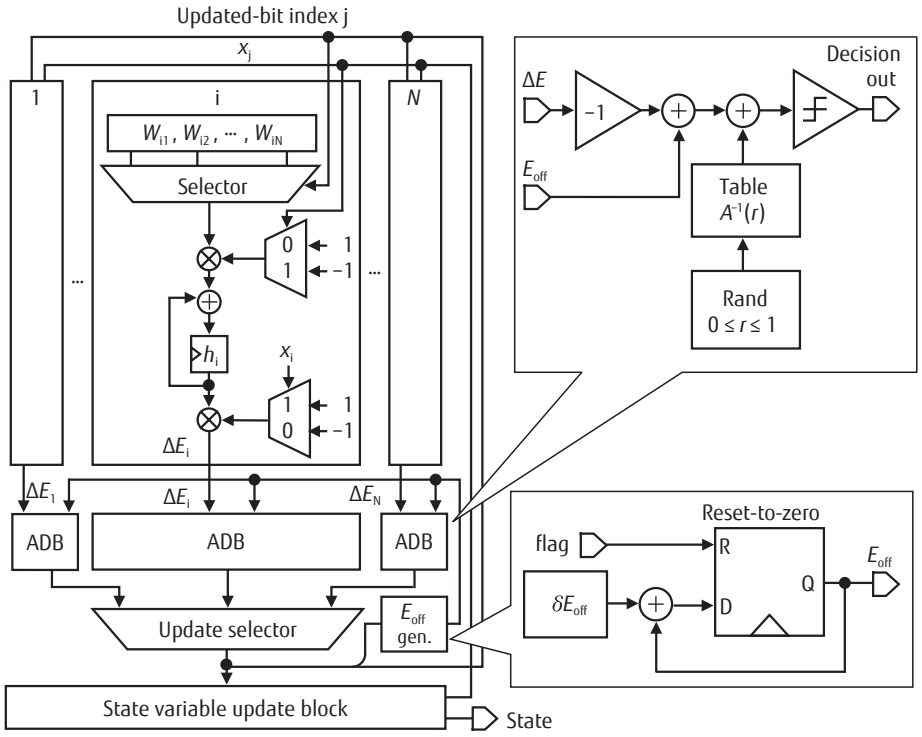

Figure 3
Optimizer architecture.

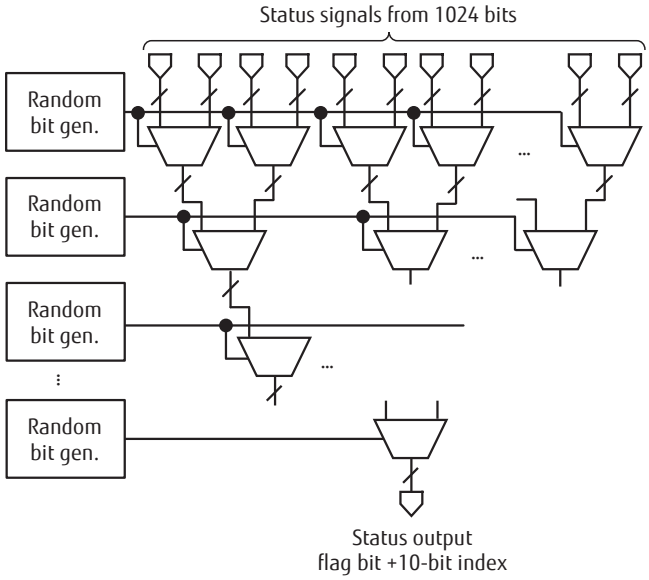

Figure 4
Update selector.

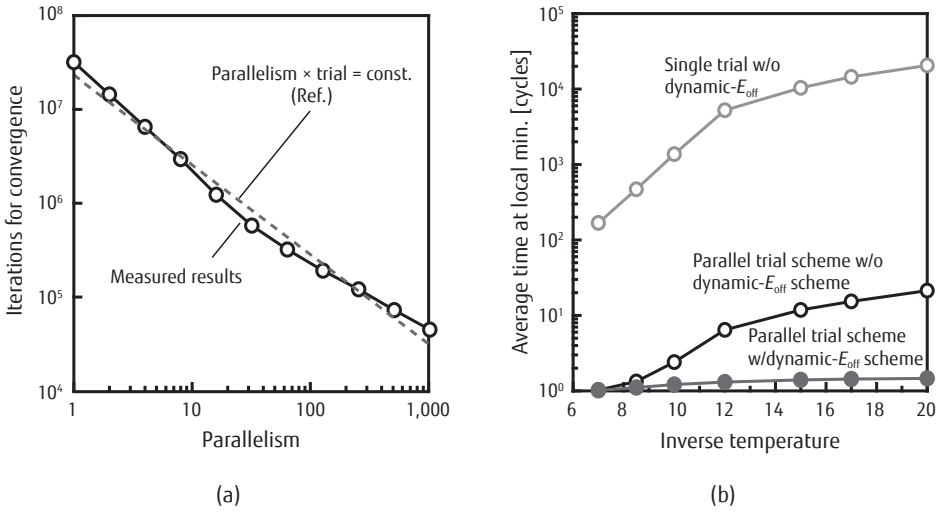

Figure 5
Effects of using acceleration schemes: (a) speedup using parallel trial scheme and (b) reduction in average time spent at local minimum.

used to generate the new state.

This parallel trial scheme accelerates convergence by increasing the probability of finding a state to which the system can move in the next update phase. Simulation showed that the number of cycles needed to reach the global minimum for a 32-city traveling salesman problem (TSP) is inversely proportional to the parallelism of the trials [**Figure 5(a)**]. Since the trials are executed in parallel for all 1024 variables, there is no additional speed penalty due to serial execution of multiple trials. Unlike a parallel update scheme in which several state variables are updated in parallel, this scheme guarantees convergence without requiring knowledge of the problem structure.

4. Acceleration scheme 2: dynamic offset

When the state is at a local minimum of the Ising energy function, the probability of moving to a new state is much smaller than that even with the parallel trial scheme. The system thus stays at the same local-minimum state for many cycles, which slows convergence. In the rejection-free Metropolis scheme used by Zhu, *et al.*,[9] the probability of moving to another state is made equal to 1 by normalizing the state acceptance probability so that the sum of the normalized acceptance probabilities is 1, resulting in a time-warp feature whereby the number of cycles during which the system stays at a local minimum is reduced to one.

Although this time-warp feature is effective, the computational overhead for normalizing the acceptance probability is high. To reduce the overhead while retaining the effects of the time-warp feature, we implemented a scheme to subtract a positive offset E_{off} from the energy increment (Figure 3), which is approximately equivalent to multiplying a common constant factor $\exp(\beta \cdot E_{off}) > 1$ by the state-flip acceptance probabilities. This scheme shortens the time the system spends at a local minimum, as shown in **Figure 5(b)**. This is achieved by having an offset generator generate the offset dynamically by adding a constant increment to the offset value when there is no move to a new state. When there is a state-variable flip, the flag output from the update selector becomes one, resetting the offset value to zero.

5. Connection between bits

To search for the shortest path using the proposed architecture, the TSP is mapped onto an Ising model. A 32-city TSP can be represented by a 32×32 matrix, in which the rows are the visiting order and the columns are the cities. We thus need 1024 bits for a 32-city TSP. W_{ij} represents the distance between two cities. Penalties are used to restrict the visits to one per city. Many connections between each bit are needed to efficiently implement these constraint conditions. In general, a large number of connections between bits with enough resolution in W_{ij} is required to solve a TSP. It is particularly hard to solve the TSP using an Ising model architecture with fewer connections or lower resolution in W_{ij}. The proposed architecture has both fully connectable and 16-bit resolution W_{ij}, making it well suited for solving TSPs. It is also well suited for the max-cut problem, a standard graph problem for evaluating the performance of algorithms, as such problems require much connectivity but very low resolution in W_{ij} (i.e., −1, 0, or 1).

6. Benchmark results

The proposed architecture was implemented in an Altera Arria 10 GX FPGA with a 1-GB DDR4 SDRAM (double-data-rate fourth-generation synchronous dynamic random-access memory) development board clocked at 100 MHz. It took five clock cycles per trial phase, resulting in a search of 20.4 G (=1024×100 MHz/5) trials/s. As a benchmark, time-to-solution metrics with 99% confidence for 32-city TSPs were evaluated. When the parallel trial and energy offsetting were not used, the system was about two times faster than ones using an in-house simulated annealing algorithm performing 13.7 M trials/s, running on a 3.5-GHz Intel Xeon E5-1620 v3 processor. When the 1024-parallel trial scheme was applied, an additional speedup of about 1,000 was obtained. When the dynamic-energy offset scheme was used in addition to the parallel trial scheme, another speedup of about 6 was obtained, resulting in a 12,000× speedup in processor performance (**Figure 6**).

7. Conclusion

Two schemes were developed for optimizing a fully connected 1024-bit Ising model and were implemented in an FPGA. Testing demonstrated that

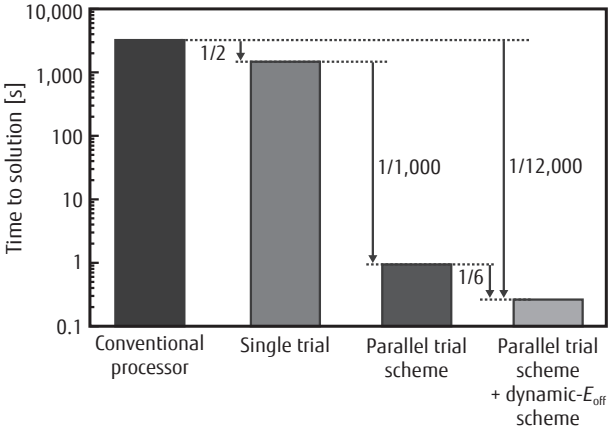

Figure 6
Speed comparison for 32-city traveling salesman problem.

a system using this architecture can solve the 32-city traveling salesman problem 12,000 times faster than one running on a conventional CPU using the same algorithm.

References

1) R. Colwell: The Chip Design Game at the End of Moore's Law. Hot Chips, Vol. 27, 2015.
2) J. G. Koomey et al.: IEEE Annals of the History of Computing, July–September, pp. 46–54, 2011.
3) M. Horowitz: Computing's Energy Problem: (and what we can do about it). ISSCC2014, 1-1, 2014 (referring to Markovic, EE292 Class, Stanford, 2013).
4) P. Bunyk et al.: Architectural Considerations in the Design of a Superconducting Quantum Annealing Processor. IEEE Trans. Applied Superconductivity, Vol. 24, No. 4, 2014.
5) S. Utsunomiya: Mapping of Ising models onto injection-locked laser systems. Optics Express, Vol. 19, No. 19, Sep. 2011.
6) P. A. Merolla et al.: A million spiking-neuron integrated circuit with a scalable communication network and interface. Science 8 August 2014, Vol. 345 No. 6197 pp. 668–673.
7) Y. Chen et al.: DaDianNao: A Machine-Learning Supercomputer. 47th IEEE/ACM Int. Symp. on Microarchitecture, pp. 609–622, 2014.
8) K. Hukushima and K. Nemoto: Exchange Monte Carlo Method and Application to Spin Glass Simulations. J. Phys. Soc. Jpn. Vol. 65, pp. 1604–1608, 1996.
9) H. Zhu et al.: A Boltzmann Machine with Non-rejective Move. IEICE Trans. Fundamentals of Electronics, Vol. E85-A, pp. 1229–1235, Jun. 2002.

Sanroku Tsukamoto
Fujitsu Laboratories Ltd.
Dr. Tsukamoto is currently engaged in research and development on systems for combinatorial optimization problems.

Motomu Takatsu
Fujitsu Laboratories Ltd.
Dr. Takatsu is currently engaged in research and development on systems for combinatorial optimization problems.

Satoshi Matsubara
Fujitsu Laboratories Ltd.
Mr. Matsubara is currently engaged in research and development on systems for combinatorial optimization problems.

Hirotaka Tamura
Fujitsu Laboratories Ltd.
Dr. Tamura is currently engaged in research and development on systems for combinatorial optimization problems.

