INSTITUTE OF PHYSICS PUBLISHING NANOTECHNOLOGY
Nanotechnology 12 (2001) 60–67 www.iop.org/Journals/na PII: S0957-4484(01)13454-9

Boltzmann machine neural network devices using single-electron tunnelling

Takashi Yamada, Masamichi Akazawa, Tetsuya Asai and Yoshihito Amemiya

Department of Electrical Engineering, Hokkaido University Kita 13, Nishi 8, Sapporo, 060-8628, Japan

E-mail: yamada@sapiens-ei.eng.hokudai.ac.jp

Received 25 April 2000, in final form 2 January 2001

Abstract

We proposed a method of *implementing the Boltzmann machine neural network on electronic circuits by making use of the single-electron tunnelling phenomenon*. The single-electron circuit shows stochastic behaviour in its operation because of the probabilistic nature of the electron tunnelling phenomenon. It can therefore be successfully used for implementing the stochastic neuron operation of the Boltzmann machine. The authors developed a single-electron neuron circuit that can produce the function required for the Boltzmann machine neuron. A method for constructing Boltzmann machine networks by combining the neuron circuits was also developed. The simulated-annealing operation can be performed easily by regulating an external control voltage for the network circuits. A sample network was designed that solves an instance of a combinatorial optimization problem. Computer simulation demonstrated that, through the simulated-annealing process, the sample network can converge to the global minimum energy state that represents the correct solution to the problem.

1. Introduction

One of the challenges in nanoelectronics is the development of novel electronic devices that can perform neural computing by utilizing functional properties of quantum phenomena. We have proposed one such computation device: *a Boltzmann machine neural circuit that utilizes the inherent stochastic nature of single-electron tunnelling*.

The Boltzmann machine is a kind of recurrent neural network that can solve various problems in areas such as combinatorial optimization, classification, and learning. It consists of a large network of processing units (neurons) interconnected bidirectionally with signal connections having various connection weights. Each neuron receives input signals from every other neuron and sends output signals to every other neuron. The neuron has a binary output state and changes its state in response to the inputs, according to a stochastic transition rule. All neurons operate in parallel and each one adjusts its own state to the states of all the others; as a consequence, the whole network converges into an optimal configuration. The structure of mathematical problems such as combinatorial optimization can be mapped onto the structure of a Boltzmann machine by determining the connection weights between the neurons. In this way, finding the optimal solution to a problem can be reduced to finding the optimal configuration of the Boltzmann machine. The unique and important feature of the Boltzmann machine is its stochastic neuron operation combined with simulated-annealing algorithms. This allows the Boltzmann machine to reach a globally optimal configuration (and thereby an optimal solution) without falling into local minimum configurations (for detailed explanations, see [1,2]).

A Boltzmann machine large-scale integration (LSI) circuit for practical use must integrate thousands of neurons on a chip. The crucial problem in developing such LSIs is how to implement the generation of randomness for the stochastic neuron operation. Every neuron has to have its own randomness because stochastic independence between the neurons is required. Electronic circuits that are currently available for generating randomness—such as thermal noise amplifiers and random bit generators—consist of many device elements and, consequently, require a large area. They, therefore, cannot be used for LSI implementation and so there is a need for a novel device for constructing Boltzmann machine LSIs.

A few years ago, the authors suggested that the single-electron circuit, a quantum electronic circuit based on the Coulomb blockade effect in electron tunnelling, can be utilized

for generating randomness for stochastic operations [3]. We postulated that the single-electron circuit would concisely produce the stochastic neuron device required for the Boltzmann machine. To give a practical form to this idea, we propose in this paper an actual circuit construction for the single-electron Boltzmann machine network.

In the following sections, first, the operation of the stochastic neuron is outlined. We present an idea for implementing the stochastic neuron using the single-electron circuit. The single-electron circuit is an electronic circuit utilizing tunnel transport of electrons and has inherently stochastic properties in its operation (section 2). We then develop an actual circuit structure for a neuron element and demonstrate by computer simulation that the developed neuron circuit can successfully produce a random binary-bit stream in response to the input signals according to the probability function regulated by a control voltage (section 3). By combining a number of the neuron circuits, we construct a Boltzmann machine network; as an example, we design a sample network circuit that represents an instance of the max cut problem (section 4). The problem-solving behaviour of the sample network circuit is then demonstrated by computer simulation. It is shown that the network circuit, under the simulated-annealing operation, does converge successfully to its globally optimal configuration that corresponds to the correct solution, without falling into local minimum configurations (section 5). The authors hope that this paper will be useful to researchers who are aiming to create neural computing devices that utilize quantum phenomena.

2. The Boltzmann machine network and the single-electron circuit

2.1. *Function of neurons required for Boltzmann machine operation*

The Boltzmann machine consists of a network of many identical neurons that are interconnected with each other. The configuration of the network is illustrated in figure 1. The output of each neuron feeds back into inputs of other neurons, and each neuron exchanges signals with others to update its own output. Denoted by W_{ij} is the connection weight to neuron i from neuron j, T_i is the threshold connection weight to neuron i from a bias that is fixed at the value of 1, and x_i is the output of neuron i. The connection weights W_{ij} and T_i can be given any desired value under the restrictions that $W_{ij} = W_{ji}$ and $W_{ii} = 0$. The output of each neuron is binary, and a set of neuron outputs (x_1, x_2, x_3, etc) is called the state of the network. We hereafter represent the neuron output by 1 or -1 (i.e. $x_i = 1$ or -1) because the neuron circuit we have developed (described in section 3.1) produces a *bipolar-mode* output.

In this network, each neuron i takes a weighted sum of the inputs of the other neurons and the threshold connection weight according to the following equation

$$s_i = \sum_{j \neq i} W_{ij} x_i + T_i. \tag{1}$$

Each neuron generates an output, 1 or -1, updating the output state continuously, following the logistic-sigmoid probability

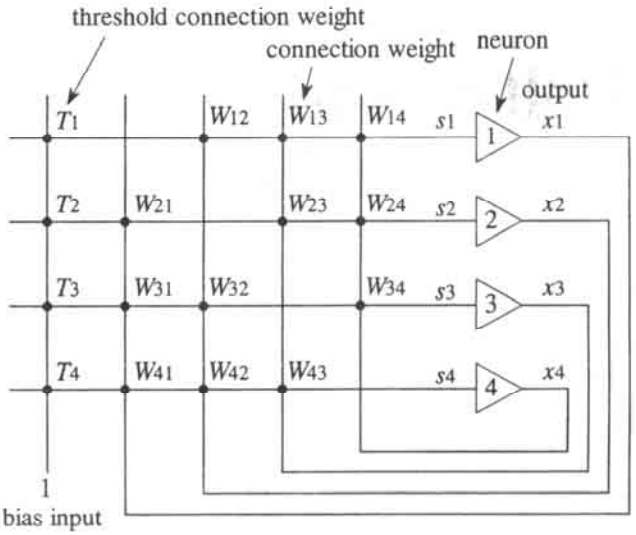

Figure 1. Concept of the Boltzmann machine. It is a recurrent network consisting of many identical neurons with signal connections. Each neuron produces a binary stochastic output.

function given by

$$f(s_i) = 1/(1 + \exp(s_i/c)), \tag{2}$$

where $f(s_i)$ is the probability for generation of an output 1. A *control parameter*, denoted by c, is for regulating the probability function. It is decreased slowly from a large positive value to zero during the simulated-annealing process. Through this process, the Boltzmann machine network changes its state to minimise the 'energy' function defined by

$$-\tfrac{1}{2} \sum_i \sum_j W_{ij} x_i x_j - \sum_i T_i x_i. \tag{3}$$

By adjusting connection weights W_{ij} and T_i, we can relate the energy function of the network to the objective function of a given optimization problem. In this way, we can find the solution to a problem simply by observing the final state that the network reaches.

2.2. *The concept of the single-electron circuit*

The single-electron circuit is an electronic circuit consisting of tunnel junctions and capacitors designed to manipulate electronic functions by controlling the transport of individual electrons (for a detailed explanation, see [4]). A single-electron circuit has a number of nodes that are interconnected by the tunnel junctions. Electrons in each node can tunnel to another node through the tunnel junction. The internal state of the circuit is determined by the configuration of its electrons (i.e. the pattern in which the electrons are distributed among the nodes). The circuit changes its electron configuration through electron tunnellings in response to the inputs and, thereby, changes its output voltage as a function of the inputs. A change of the electron configuration caused by a tunnelling event can occur, at low temperatures, only when the free energy of the circuit decreases during the tunnelling event. This phenomenon is called the *Coulomb blockade effect*. Utilizing this phenomenon, the single-electron circuit controls the transport of individual electrons in order to produce various functions such as digital logic and memory operations.

The single-electron circuit has been receiving increasing attention because it can be used to produce LSIs that combine large integration and ultra-low power dissipation. The key point for constructing single-electron devices is to fabricate the circuit elements (tunnel junctions and capacitors) in very minute dimensions (50 nm or less) because the Coulomb blockade effect emerges only when the capacitance of each element is reduced to femtofarads or less. The technology for such nanofabrication is still immature but is making steady progress. Several elementary circuits, such as logic gates and memory cells, have been produced in recent years, and a prototype of single-electron LSI can be expected in the near future.

2.3. Stochastic properties of single-electron circuits

A conspicuous property of the single-electron circuit is that the circuit shows stochastic behaviour in its operation. This is caused by the fact that the electron tunnelling is a probabilistic phenomenon. A single-electron circuit changes its electron configuration in response to the inputs through a sequence of electron tunnellings, where the waiting time for each tunnel event shows a probabilistic fluctuation (i.e. a tunnelling might occur after a very short time, or it might not occur for a long time; see the appendix). If we construct a single-electron circuit such that its output is modulated by the probabilistic fluctuation in the tunnel waiting time, then we can obtain novel electronic devices showing stochastic operation. Since the stochastic property is inconvenient for ordinary digital applications, single-electron circuits have been designed in such a configuration so that the stochastic property will be suppressed as thoroughly as possible. In contrast, however, the stochastic properties can be well utilized for the present purpose; that is, creating a Boltzmann machine device with a compact design.

3. Single-electron circuit for the Boltzmann machine neuron

3.1. Creating the stochastic neuron using a single-electron circuit

The point of our idea is that *the operation of a Boltzmann machine neuron can be produced by a digital oscillator consisting of a single-electron circuit.* A digital oscillator is a circuit that generates an output of a 1/−1 bit stream; and if the oscillator consists of a single-electron circuit, then the duration of an output 1 (or an output −1) will fluctuate randomly because of the probabilistic nature of electron tunnelling. As a consequence we will be able to obtain an output of a *random* 1/−1 bit stream required for the Boltzmann machine operation. To produce the complete function of the neuron, the digital oscillator must be designed so that the probability for an output 1 can be modulated by a signal input (s_i in equation (1)), according to the logistic-sigmoid probability function regulated by a control input (c in equation (1)). For this purpose, we take *Tucker's single-electron inverter* and modify its circuit configuration to create the neuron function (for details of Tucker's original circuit, see [5]).

The circuit we developed for the neuron device is illustrated in figure 2. It consists of four tunnel junctions (C_{j1} through C_{j4}) and seven capacitors (C_1 through C_7), and is supplied with two bias voltages, i.e., a positive voltage V_{dd} and a negative voltage $-V_{ss}$. The offset voltage V_b is used to adjust the operating point of the circuit. The circuit, with the appropriate set of parameters given later, operates as an unstable multi-vibrator or a square-wave oscillator. That is, the circuit repeats a cycle of transferring one electron from $-V_{ss}$ to V_{dd} by making an electron tunnel from one node to another in the following sequence: (tunnelling 1) node $L \rightarrow V_{dd}$, (tunnelling 2) node $M \rightarrow$ node L, (tunnelling 3) $-V_{ss} \rightarrow$ node N, and (tunnelling 4) node $N \rightarrow$ node M (see section (3.2)). The output voltage x_i of the circuit is nearly equal to V_{dd} (an output 1) during a period between tunnelling 2 and tunnelling 4 because, in this period, an electron is extracted from node M (output node) and, consequently, node M is charged positive. In the remaining period, the output is nearly equal to $-V_{ss}$ (an output −1). The time that the node M is charged positive depends on the waiting time for tunnellings, so the output randomly alternates between 1 and −1 according to the probabilistic fluctuation in the tunnel waiting time. The probability for an output 1 can be controlled by external voltage s_i. Thus the circuit accepts voltage input s_i (the weighted sum of inputs) and produces the corresponding voltage output x_i in a form of a random 1/−1 bit stream. We hereafter call this circuit a *unit-neuron circuit.* (This circuit does not include a subcircuit for calculating the weighted sum of inputs s_i. For calculating s_i according to equation (1), we will use an additional capacitor subcircuit, as described in section 4.)

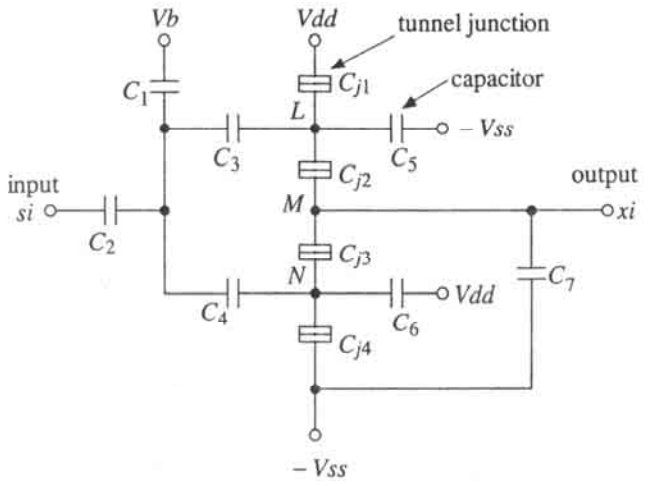

Figure 2. Configuration of the unit-neuron circuit for single-electron Boltzmann machines.

To create the stochastic neuron operation, we set the circuit parameters so that the circuit can operate under oscillating conditions. In determining the optimum parameters, we used the *stability diagram* of the circuit as a guide map. (The stability diagram illustrates the internal states of a single-electron circuit in a multi-dimensional space of circuit variables—namely, the voltages of powers and inputs, and the capacitances of tunnel junctions and capacitors.) An example set of the capacitance parameters is

$$\begin{aligned} C_{j1} = C_{j4} = 1\ \text{aF}, \qquad & C_{j2} = C_{j3} = 2\ \text{aF}, \\ C_1 = C_2 = 12\ \text{aF}, \qquad & C_3 = C_4 = 4\ \text{aF}, \\ C_5 = C_6 = 10\ \text{aF}, \qquad & C_7 = 24\ \text{aF}. \end{aligned} \tag{4}$$

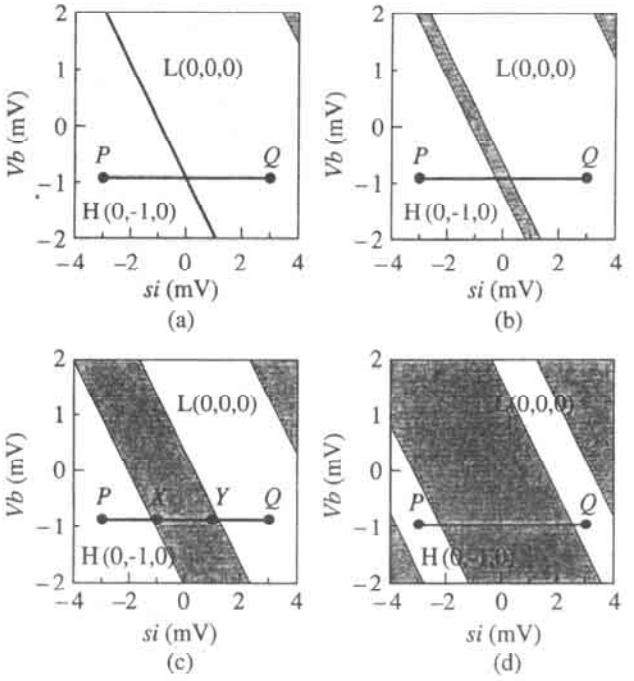

Figure 3. Stability diagrams of the unit-neuron circuit given in figure 2, plotted on a plane of input voltage s_i and offset voltage V_b. For capacitance parameters, see the text. The shaded regions are unstable regions. Figures (*a*) through to (*d*) correspond to a gradual increase in V_{dd} and V_{ss}. The value of $(V_{dd}, -V_{ss})$ is: (*a*) (2.72 mV, −3.29 mV), (*b*) (2.74 mV, −3.30 mV), (*c*) (2.80 mV, −3.36 mV), (*d*) (2.87 mV, −3.43 mV).

3.2. *Operation of the unit-neuron circuit*

The internal state of the unit-neuron circuit is expressed by the numbers of excess electrons (l, m, n) on the three nodes (L, M, and N) in the circuit. Assuming the capacitance parameters given by equation (4), we drew the stability diagram in a four-dimensional space of four voltage variables (s_i, V_b, V_{dd}, and $-V_{ss}$). In figures 3(*a*) through to 3(*d*), part of the diagram is illustrated on a plane of two voltage variables, input s_i and offset V_b. The two white regions are stable regions, in which the circuit stabilizes at internal states $(0, -1, 0)$ and $(0, 0, 0)$; the former state produces a positive output voltage (an output 1), while the latter produces negative output voltage (an output −1). The output state for each internal state is illustrated by putting a letter (H for an output 1, or L for an output −1) before the electron number set. The shaded regions are unstable regions in which electron tunnelling frequently occurs and, consequently, the circuit alternates between two or more internal states to output a random 1/−1 bit stream. The width of the unstable region can be controlled by regulating bias voltages V_{dd} and $-V_{ss}$ as shown in figures 3(*a*) through to (*d*).

We operate the unit-neuron circuit on an operating line illustrated by PQ in figures 3(*a*) through to (*d*). It can be expected that the probability for generation of an output 1 can be changed from 1 to 0 continuously by increasing input s_i to move the operating point from the H$(0, -1, 0)$ region to the L$(0, 0, 0)$ region on the line PQ. In addition we will be able to change the control parameter for the probability function by regulating bias voltages V_{dd} and $-V_{ss}$ to change the width of the unstable region. In regulating V_{dd} and $-V_{ss}$, the value of

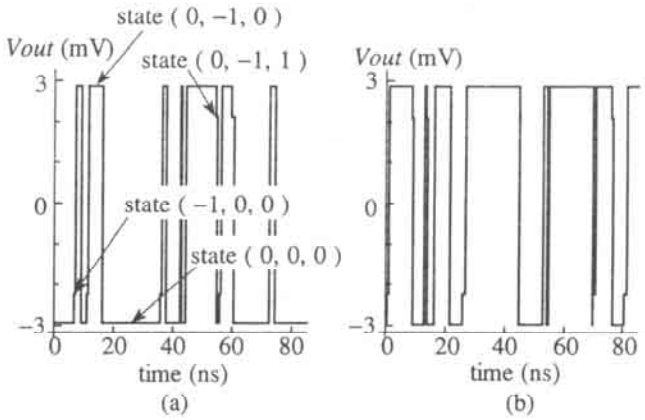

Figure 4. Output voltage waveforms for the unit-neuron circuit with the control-parameter set of (2.80 mV, −3.36 mV, −0.87 mV). Simulated for two input voltages: (*a*) $s_i = 1$ mV and (*b*) $s_i = -1$ mV. For circuit parameters, see the text. Temperature is 0 K.

offset V_b has to be adjusted simultaneously so that the operating point for zero inputs ($s_i = 0$) will be situated on the centre line of the unstable region and, thereby, the probability for an output 1 will exactly be 0.5 at zero inputs. We hereafter call a set of (V_{dd}, $-V_{ss}$, and V_b) the *control-parameter set*.

To confirm our expectation, we simulated the circuit operation; in simulation, we employed the Monte Carlo method combined with the basic equations for electric-charge distribution, charging energy, and tunnelling probability (for this method, see [6] and the appendix). The parameters used here are the same as those given in section 3.3, with tunnel resistances of 100 kΩ for tunnel junctions C_{j1} and C_{j4}, and 5 MΩ for C_{j2} and C_{j3}. The temperature was assumed to be 0 K.

A result of the simulation is illustrated in figure 4 for the control-parameter set of (2.80 mV, −3.36 mV, −0.87 mV) that corresponds to the operating line PQ in figure 3(*c*). The figure shows the output voltage waveform (a random 1/−1 bit stream) for two values of the input voltage: (a) $s_i = 1$ mV (point Y in figure 3(*c*)) and (b) $s_i = -1$ mV (point X in figure 3(*c*)). As we expected, the probability for an output 1 can be changed by input s_i. The state of a negative output (output '−1') is dominant for a negative value of s_i (figure 4(*a*)), while the state of a positive output (output '1') is dominant for a positive value of s_i (figure 4(*b*)). Intermediate states are also generated (i.e. states $(0, -1, 1)$ and $(-1, 0, 0)$ in figure 4(*a*)), but this is not a problem because their duration is always short regardless of the input voltage value. In this example, the circuit changes its internal state in a cycle of: L$(0, 0, 0)$ → L$(-1, 0, 0)$ → H$(0, -1, 0)$ → H$(0, -1, 1)$ → L$(0, 0, 0)$. A similar operation is observed in other control-parameter sets.

The probability for an output 1 is illustrated in figure 5(*a*) as a function of input s_i, for various control-parameter sets $(V_{dd}, -V_{ss}, V_b)$. It is obtained by observing the output 1/−1 stream for 10 μs and measuring the total duration of an output 1. The probability function required for the stochastic neuron can be obtained easily. The probability can be controlled by regulating the control-parameter set; the number (#1 through #4) for each curve indicates a specific control-parameter set that is required for producing the characteristic of the curve. Figure 5(*b*) illustrates a diagram for setting the

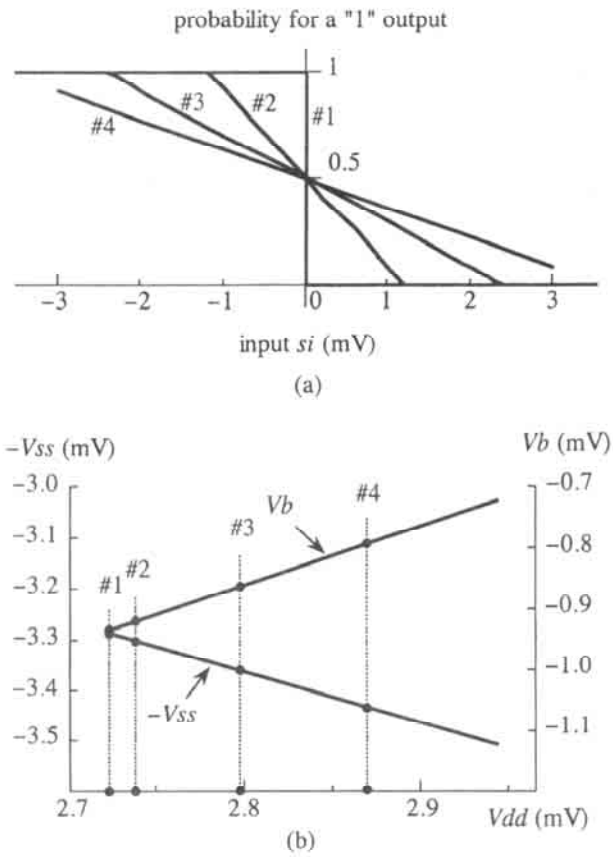

Figure 5. Probability function of the unit-neuron circuit with the device parameters in the text: (*a*) the probability of generating an output 1 for various control-parameter sets, and (*b*) a diagram for setting the control-parameter set. Curves #1 through #4 in (*a*) are obtained for the control-parameter sets #1 through #4 in (*b*).

control-parameter set (V_{dd}, $-V_{ss}$, V_b), where the numbers #1 through #4 indicate the sets for producing the curve #1 through #4 in figure 5(*a*) (e.g. the set for curve #3 can be obtained for $V_{dd} = 2.80$ mV, $-V_{ss} = -3.36$ mV, and $V_b = -0.87$ mV). In curve #1, the circuit acts as a simple threshold element without stochastic operation, which corresponds to the condition of $c = 0$ in equation (1). (Strictly speaking, the obtained characteristic is somewhat different from that of equation (1); i.e. the characteristic is a line sigmoid function rather than a logistic sigmoid. This point leaves room for improvement, but for now we will use this neuron circuit.)

4. Designing the Boltzmann machine network

The Boltzmann machine can be constructed by combining the unit-neuron circuits into a network. The overall configuration of the network circuit is illustrated in figure 6. The network consists of a number of unit-neuron circuits with buffer inverters, negative-weight inverters, and connection capacitors. The buffer inverter is added to each unit-neuron circuit for intensifying the power of load drivability. We define the output of neurons as the voltage of buffer inverter outputs (1+, 2+, 3+, etc). The negative-weight inverters produce voltage signals (1−, 2−, 3−, etc) that are complementary to the neuron outputs (1+, 2+, 3+, etc) and the complementary signals are used for obtaining negative-weight connections. (For the buffer inverters and the negative-weight inverters, we used a single-electron inverter illustrated in figure 7.) Both the output and its complement of each unit-neuron circuit feed back into inputs for other unit-neuron circuits. The connection between two neurons is established by a coupling capacitor C_{ij}. The threshold for each neuron is set up by positive- bias voltage V_1 (or by negative voltage $-V_2$) with a coupling capacitor C_i.

Connection weights between neurons can be set at any values by choosing the capacitances of the coupling capacitors. Each weight (W_{ij} and T_i in equation (3)) is given by

$$|W_{ij}| = \frac{C_{ij}}{\sum_j C_{ij}+C_i}, \qquad \text{and} \qquad |T_i| = \frac{C_i}{\sum_j C_{ij}+C_i}. \tag{5}$$

For a positive-weight ($W_{ij} > 0$), the coupling capacitor is connected with the input node of neuron i and the output node (1+, 2+, 3+, etc) of neuron j, and for a negative-weight ($W_{ij} < 0$), with the input node and the complementary-output node (1−, 2−, 3−, etc). The threshold coupling is made between the input node and positive-bias node $V1$ if $T_i > 0$, and between the input node and the negative-bias node $-V2$ if $T_i < 0$. The capacitances C_{ij} and C_i have to be set at such values that the symmetry in connection (i.e. $W_{ij} = W_{ji}$ for all i, j) can be established. Under these conditions, the network circuit will operate as a complete Boltzmann machine.

5. Problem-solving operation of the network circuit

By setting appropriate values for the coupling capacitances, we can implement various optimization problems on the network circuit. As an example, we construct here a sample network circuit that solves an instance of the *max cut problem*, and demonstrate by computer simulation the problem-solving operation of the network circuit.

5.1. Implementing the max cut problem on the network circuit

The max cut problem is stated as follows: given a graph $G = (V, E)$ with positive-weights on the edges, find a partition of the vertices $V = \{1, 2, \ldots, n\}$ into two disjoint sets V_0 and V_1 such that the sum of the weights of the edges that have one endpoint in V_0 and one endpoint in V_1 is maximal. To formulate the objective function for this problem, we define here a number of variables. Let d_{ij} be the weight associated with the edge $\{i, j\}$ (by definition, $d_{ij} = d_{ji}$) and let x_i be a $1/-1$ variable defined as

$$x_i = 1 \ (\text{if } i \in V_1)$$
$$x_i = -1 \ (\text{if } i \in V_0), \tag{6}$$

then the max cut problem can be formulated as maximise

$$\sum_i \sum_j (d_{ij}/4)\,(x_i - x_j)^2, \tag{7}$$

which can be rewritten using ${x_i}^2 = {x_j}^2 = 1$, as minimise

$$-\tfrac{1}{2} \sum_i \sum_j d_{ij} x_i x_j. \tag{8}$$

The max cut problem can then be implemented by designing a network circuit such that the output of each neuron represents

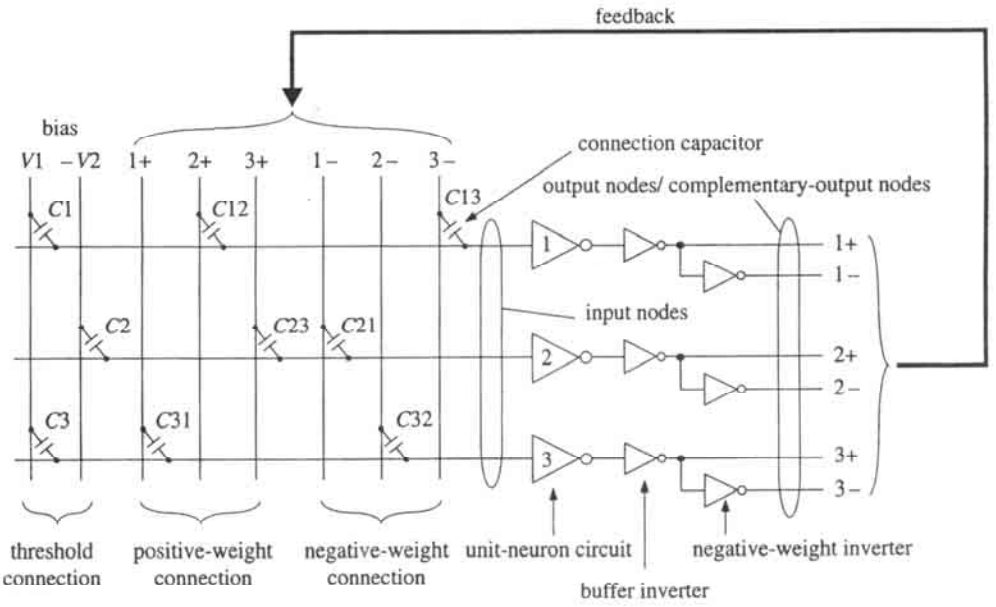

Figure 6. Overall configuration of the single-electron Boltzmann machine network. The neuron outputs (1+, 2+, 3+, etc) and the complementary-outputs (1−, 2−, 3−, etc) feed back to become the inputs for the unit-neuron circuits. The connection between two neurons is established by the coupling capacitor C_{ij}, and the threshold input for each neuron is set by the coupling capacitor C_i.

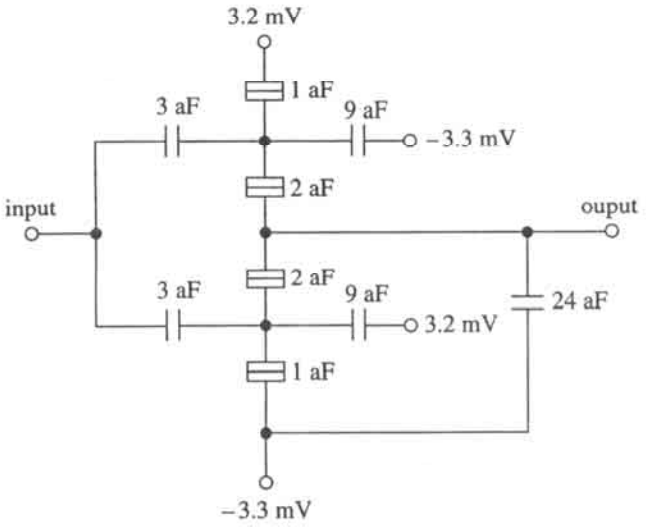

Figure 7. A single-electron inverter used for the buffer inverters and the negative-weight inverters together with its circuit parameters (tunnel resistance = 100 kΩ for all of the four junctions).

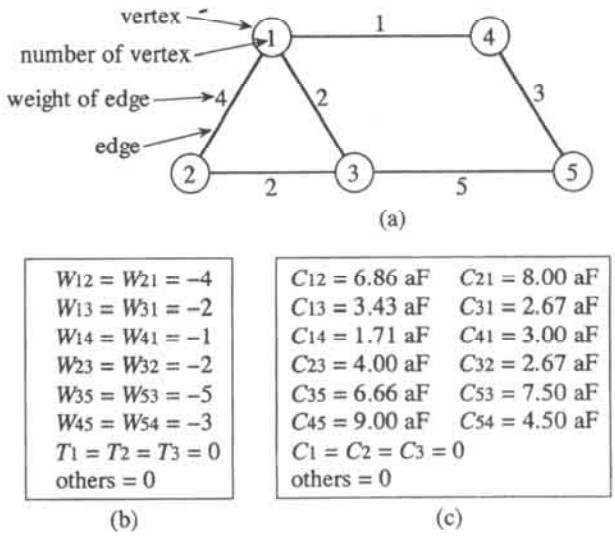

Figure 8. An instance of the max cut problem and the corresponding connection weights and coupling capacitances: (*a*) a weighted graph for the problem; (*b*) connection weights W_{ij} between neurons; and (*c*) coupling capacitances for the network circuit.

each variable x_i. As an example, here we take up a weighted graph given in figure 8(*a*) and design a network circuit whose structure is isomorphic to the graph. To implement this problem instance, we prepare the network circuit with five neurons and represent vertex i of the problem graph by the ith neuron ($i = 1$ through 5). The required connection weights W_{ij} between the neurons can be determined as in figure 8(*b*) by comparing the objective function given by equation (8) with the energy function given by equation (3). From the weight values W_{ij} we can determine a set of the coupling capacitances for the circuit construction by using equation (5). The result is given in figure 8(*c*). The network circuit with this coupling capacitance set will hereafter be called the *sample network*.

5.2. *Energy function and local minima in the sample network*

The internal state of the sample network is expressed by a set of five neuron outputs $(x_1, x_2, x_3, x_4, x_5)$, where x_i is 1 or −1. For simplicity, let us represent the set by a code of signs such as (++−+−), where + denotes '$x_i = 1$' and − denotes '$x_i = -1$'. The sample network has 32 possible internal states. The value of the energy function calculated from equation (3) is plotted in figure 9 for all the states. States (−+++−) and (+−−−+) are the global minimum and represent the correct solution to the problem (that is, the maximal cut for the problem graph of figure 8(*a*) is given by two disjointed sets of vertices {1, 5} and {2, 3, 4}). The network can change its internal state through the transition of a *Hamming distance of 1*. From the second

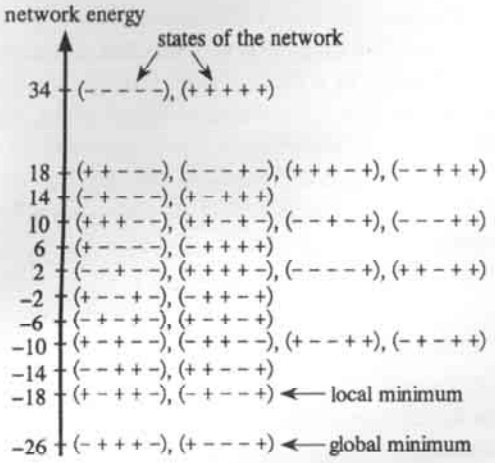

Figure 9. Energy diagram for the sample network circuit corresponding to the problem instance of figure 8(*a*). The notation, such as (+ + − + −), denotes the set of five neuron outputs. States (− + + + −) and (+ − − − +) correspond to the global minimum that represents the correct solution to the problem. The second lowest states (+ − + + −) and (− + − − +) correspond to local minima.

lowest states (−++−+) and (+−−+−) to the global minimum states, there is *no* transition path of a *Hamming distance of 1*—therefore these two states act as a local minimum.

5.3. Problem-solving operation of the sample network

For problem solving, it is essential that, starting with a given initial state, the network circuit should converge to its global minimum energy states. To observe the behaviour of the sample network, we simulated the state transition of the network.

We simulated the behaviour of the sample network, first, without the annealing operation. (In this examination, all the unit-neuron circuits were set at the condition of simple threshold (curve #5 in figure 5(*a*)).) The result is illustrated in figure 10. The network was initially set at state (+ + + + +), then it was allowed to change its state without restraint. After some transition time the circuit stabilized into a final state. This procedure, a *trial*, was repeated many times using a different series of random numbers; the results of three trials are illustrated in the figure. The network was sometimes able to converge into the global minimum state (− + + + −) or (+ − − − +) (as shown by number 1), but frequently became stuck in the local minimum state (+−++−) or (−+−−+) and could not reach the global minimum (as shown by numbers 2 and 3).

We then operated the sample network under the annealing operation. In the annealing, the control-parameter set for the unit-neuron circuits was gradually changed with the advance in time, according to an appropriate schedule. In the present experimentation, we changed the control-parameter set gradually, following a cooling schedule given by $V_{dd} = 2.72+0.88\exp(-t/50)$ (mV) ($-V_{ss}$ and V_b were also changed in accordance with the curves in figure 5(*b*)), where t is time in ns. The simulation result of the circuit operation is illustrated in figure 11. The circuit was initially set at state (+ + + + +), then was allowed to change its state under the annealing operation. The results for one trial are plotted in figure 11. The circuit successfully reached the global minimum state. We

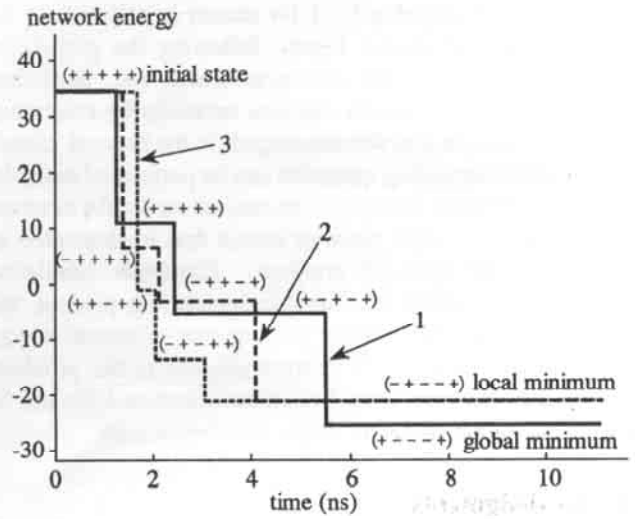

Figure 10. State transition in the sample network without the annealing (computer simulation). The results of three trials are plotted.

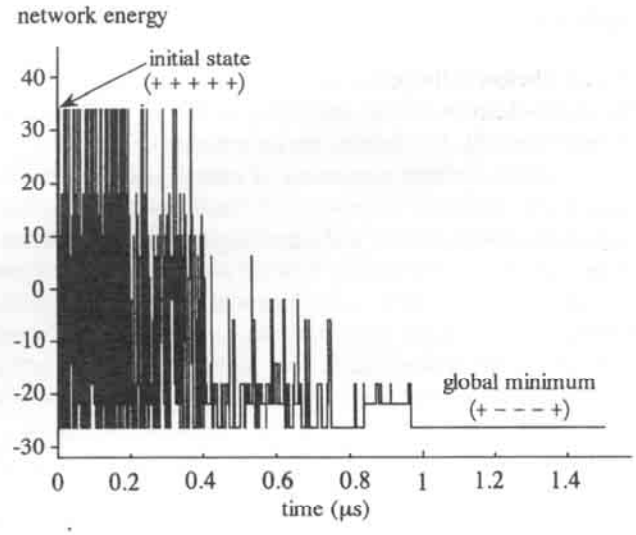

Figure 11. State transition in the sample network under annealing (computer simulation). The result of one trial is plotted and it can be seen that the network successfully reaches the global minimum state.

repeated the same trial many times and confirmed that every trial resulted in successful convergence. In this way, we can find the global minimum state of the network and, thus, the correct solution to the problem.

6. Conclusion

The authors developed *a method of implementing the Boltzmann machine on electronic circuits using single-electron circuit technology*. The single-electron circuit has a stochastic nature of operation because the waiting time for electron tunnelling shows probabilistic fluctuation. Therefore, the operation of the Boltzmann machine can be easily implemented using such a single-electron circuit that modulates its output in response to the fluctuation in the tunnel waiting time. We developed a single-electron neuron circuit that can produce the function required for the Boltzmann machine neuron; the proposed neuron circuit

produces as an output a 1/−1 bit stream in response to the weighted sum of neuron inputs, following the probability function regulated by the control-parameter set. A method of constructing Boltzmann machine networks by combining the neuron circuits was also developed. In the network circuit, the simulated-annealing operation can be performed easily by gradually changing the control-parameter set for the neurons. We designed a sample network circuit that implemented an instance of the max cut problem. Computer simulation showed that, through the simulated-annealing process, the sample network can converge to its global minimum energy state, which represents the correct solution to the problem. Our results show that large Boltzmann machine LSIs can be fabricated compactly using single-electron circuits.

Acknowledgments

This paper was supported by CREST of JST (Japan Science and Technology).

Appendix

Outlined below is the procedure of the Monte Carlo simulation for single-electron circuits employed in this paper. This is an excerpt from [6]. For details, see the reference.

Consider a circuit consisting of tunnel junctions (tunnel capacitors), ordinary or nontunnel capacitors, and voltage sources for power, clocks, and signal inputs. The internal state of the circuit is expressed by a set of numbers that represent the count of *excess* electrons on the nodes in the circuit. (In the following, state means this set of electron numbers.) Choose a starting state of the circuit, and set time = 0. Electron tunnelling or time-dependent behaviour of the circuit for the given values of the source voltages is simulated as follows.

Step 1. Compute electrostatic energy E_0 (the sum of electrostatic energy on the tunnel junctions and ordinary capacitors) for the current state. Then enumerate all possible subsequent states and compute the electrostatic energy E_{i1} for each subsequent state i. (A subsequent state means a state into which the current state can be transformed by one tunnelling of an electron. If the number of the tunnel junctions is N, there are $2N$ possible tunnellings and therefore $2N$ subsequent states.) Also compute the energy E_{i2} that the voltage sources will supply in the transformation of the circuit from the current state to each subsequent state i.

Step 2. Compute the energy difference ΔE_i $(=E_0-E_{i1}+E_{i2})$ for each subsequent state. (It is assumed that, in a tunnel event, ΔE_i is dissipated in a form of heat through the interaction between the electron and the crystal lattice of the conductive material that forms the node of the single-electron circuit.) From the value of ΔE_i, calculate the waiting time for a tunnelling process corresponding to each subsequent state. The waiting time τ_i is given as

$$\tau_i = \frac{1}{\Gamma_i} \ln \frac{1}{\gamma} \tag{A.1}$$

where γ is a uniform random number $(0 < \gamma < 1)$ generated for each tunnel event, and Γ_i is the mean tunnelling rate (the mean number of electrons that tunnel in one second) given by

$$\Gamma_i = \frac{\Delta E_i}{e^2 R_T \{1 - \exp(-\Delta E_i / k_B T)\}} \tag{A.2}$$

where R_T is a tunnelling resistance of the tunnel junction capacitor, k_B is the Boltzmann constant, e is the elementary charge, and T is temperature. (At 0 K, $\Gamma_i = \Delta E_i/(e^2 R_T)$ for $\Delta E_i > 0$, and $\Gamma_i = 0$ for $\Delta E_i < 0$.)

Step 3. After calculating the waiting time τ_i for all possible tunnellings, take the tunnel event that has the *shortest* waiting time, and accept the corresponding subsequent state as the current state. Then put the time *forward* by τ_i and return to step 1 to repeat the iterations.

References

[1] Hinton G E and Sejnowski T J 1986 Learning and relearning in Boltzmann machines *Parallel Distributed Processing—Explorations in the Microstructure of Cognition* vol 1, ed D E Rumelhart, J L McClellend and the PDP Research Group (Cambridge, MA: MIT Press) pp 282–317

[2] Aarts E and Korst J 1989 *Simulated Annealing and Boltzmann Machines—A Stochastic Approach to Combinatorial Optimization and Neural Computing* (New York: Wiley)

[3] Akazawa M and Amemiya Y 1997 Boltzmann machine neuron circuit using single-electron tunneling *Appl. Phys. Lett.* **70** 670–2

[4] Grabert H and Devoret M H 1992 *Single Charge Tunneling—Coulomb Blockade Phenomena in Nanostructures* (New York: Plenum)

[5] Tucker J R 1992 Complementary digital logic based on the Coulomb blockade *J. Appl. Phys.* **72** 4399–413

[6] Kuwamura N, Taniguchi K and Hamakawa C 1994 Simulation of single-electron logic circuits *IEICE Trans. Electron.* **J77-C-II** 221–8