2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID)

Realizing Boolean functions using Probabilistic Spin Logic (PSL)

Vaibhav Agarwal
Indraprastha Institute of Information Technology
New Delhi- 110020, India
Email: vaibhav16119@iiitd.ac.in

Sneh Saurabh
Indraprastha Institute of Information Technology
New Delhi- 110020, India
Email: sneh@iiitd.ac.in

***Abstract*—Probabilistic Spin Logic (PSL) is a novel computing model that can be implemented using stochastic units (called p-bits) such as low-barrier nanomagnets. A PSL can exhibit accuracy which is comparable to a conventional digital circuit. Remarkably, a PSL can also be exploited to compute the inverse of a function. In this paper, using simulations, we examine the application of PSL in realizing Boolean functions. We propose a methodology to implement any Boolean function given in Conjunctive Normal Form (CNF) using PSL. Our methodology is based on synthesizing a given function in terms of NOT/AND/OR gates and deriving appropriate interconnections between p-bits. Further, we demonstrate the application of PSL in computing the inverse of a given function.**

***Index Terms*—P-bits, Boolean function.**

I. Introduction

As CMOS technology is approaching its fundamental scaling limits, new technologies based on electron spin or nanomagnets are being actively investigated. Recently, nanomagnets with low energy barrier have been proposed to implement probabilistic-bits or p-bits and realize logic functions [1]. Remarkably, logic based on p-bits or probabilistic spin logic (PSL) is invertible, a property that is absent in standard digital circuits [1]. The input I_i controls the output m_i according to the following equation:

$$m_i(t) = sgn(rand(-1,1) + tanh(I_i(t))) \tag{1}$$

where $rand(-1,+1)$ represents a random number uniformly distributed between the interval $[-1,+1]$. For a nanomagnet-based PSL implementation, I_i could be the spin current that controls the magnetization m_i. When a group of p-bits are interconnected and allowed to fluctuate in a correlated manner, useful functions similar to digital circuits can be obtained. The behavior of the i-th p-bit when interconnected with other p-bits is modeled using the following equation:

$$I_i(t) = I_0 * \{h_i(t) + \sum_j J_{ij} m_j(t)\} \tag{2}$$

where h_i provides a bias to the p-bit i and J_{ij} defines the effect of the p-bit j on the p-bit i, and I_0 sets a global scale for the strength of the interactions.

The key elements that determine the functionality of a PSL are the coefficients h_i/J_{ij} in Eq. (2) (or in matrix notation h/J-matrices). For a given functionality, h/J-matrices are not unique. In this paper, we propose a methodology to derive a PSL implementation for a given Boolean function by finding one of the possible h/J-matrices.

In this paper, a PSL system is simulated using the universal model approach [1]. The rest of this paper is organized as follows: in section II a methodology to derive a PSL implementation for a given Boolean function is presented, in section III the invertibility of a PSL is investigated and in section IV conclusions are made.

II. Realizing Logic Function using PSL

To understand how the interconnections between p-bits are made to realize a Boolean function, several approaches are possible such as Ising Hamiltonian for quantum computers, principles developed for Hopfield networks, etc. [2], [3]. For a Boolean network, the Ising Hamiltonian can be defined as [4]:

$$H_{Ising} = \sum_i h_i m_i + \sum_{<i,j>} J_{ij} m_i m_j \tag{3}$$

where J_{ij} describes the interaction strength between two neighboring p-bits m_i/m_j and h_i provides a bias to the p-bit m_i. Depending on the values assumed by the p-bits, the network can have various possible outcomes or states. These states fluctuate in a correlated manner such that on an average, a functionality similar to a digital circuit is obtained. For example, an AND gate ($Y = A \wedge B$) can be realized using two p-bits (m_1 and m_2) for the inputs and one p-bit (m_3) for the output. The h/J-matrices for AND, OR, and NOT gate are described in [2].

A. XOR Gate

A two-input XOR gate can be implemented as $Y = (A \wedge \bar{B}) \vee (\bar{A} \wedge B)$, where A, B are the inputs and Y is the output. In PSL, a two-input XOR gate can be implemented as shown in Fig. 1(a): the p-bits m_1 and m_2 are connected to the inputs, the p-bit m_7 is connected to the output and the rest four p-bits are the internal p-bits.The h/J-matrices are shown in Eq. (4) and is determined using the matrix for the AND/OR/NOT gates. For example, the p-bit m_2 and p-bit m_3 are connected to p-bit m_5 using an AND gate. Therefore, the 2^{nd}, 3^{rd} and 5^{th} rows/columns of J_X contain elements same as that of the AND gate as in [2]. Fig. 1(b) shows the simulation results of an XOR gate, demonstrating correct functionality.

2380-6923/19/$31.00 ©2019 IEEE
DOI 10.1109/VLSID.2019.00110

$$h_X = \begin{bmatrix} +1 \\ +1 \\ +1 \\ +1 \\ -3 \\ -3 \\ 2 \end{bmatrix} \quad J_X = \begin{bmatrix} 0 & 0 & -1 & -1 & 0 & 2 & 0 \\ 0 & 0 & -1 & -1 & 2 & 0 & 0 \\ -1 & -1 & 0 & 0 & 2 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 2 & 0 \\ 0 & 2 & 2 & 0 & 0 & -1 & 2 \\ 2 & 0 & 0 & 2 & -1 & 0 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 0 \end{bmatrix} \quad (4)$$

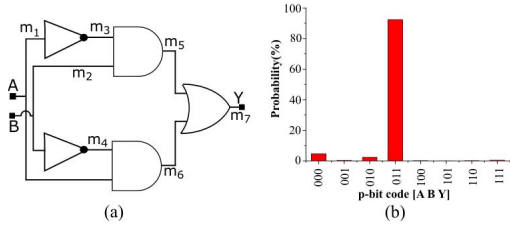

Fig. 1. **XOR gate:** (a) Implementation of two-input XOR gate using PSL. (b) Externally biased XOR gate in which A and B are clamped to '0' and '1' respectively, while Y is left floating. Histogram shows peaks for $(A, B, Y) = (011)$ which is the only valid state as per the truth table of XOR gate. The results are averaged over 1000 time samples.

B. Multiplexer

A 2:1 MUX can be described as $Y = (A \wedge \bar{sel}) \vee (B \wedge sel)$, where A and B are the data inputs, sel is the select input and Y is the output. Using the same methodology described for an XOR gate, the h/J-matrices of a 2:1 MUX is determined and Fig. 2 shows the simulation results of a 2:1 MUX when

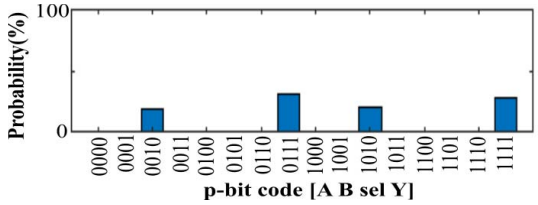

Fig. 2. Simulation results of 2:1 MUX in which sel is clamped to '1'. The response shows peaks for the four states $(A, B, sel, Y) = (0010), (0111), (1010)$, and (1111). Histogram is obtained by averaging over 10000 time samples.

$sel = 1$ is clamped. All the input bits (A and B) and the output Y fluctuate in a correlated manner such that the valid states of the MUX are visited with a high probability.

C. Boolean function in Conjunctive Normal Form(CNF)

Any Boolean function can be represented in conjunctive normal form (CNF). Given a Boolean function in CNF, an algorithm was derived to find the h/J-matrices for its PSL implementation. The algorithm represents the CNF in terms of two-input AND/OR gates and finds h/J-matrices.

The algorithm generalizes the approach described for the XOR gate. Using the derive algorithm, PSL implementations were computed for various Boolean functions represented in CNF. As another illustration, we show the simulation results of a CNF consisting of $n = 8$ variables, $m = 22$ clauses, and $k = 3$ literals per clause in Fig. 3. In this simulation,

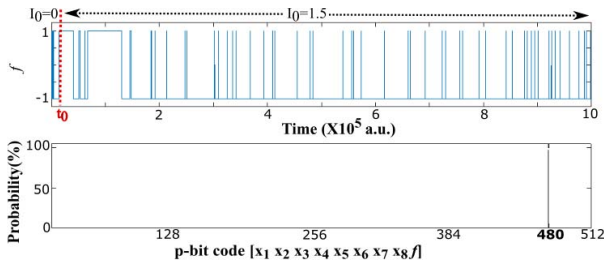

Fig. 3. Simulation results of a CNF with input bits clamped to [11110000]. Peak at the output is observed at '0' corresponding to the state '480'.

the inputs can take $2^8 = 256$ possible patterns. However, we have fixed the input to the following pattern: [11110000]. The expected output for the pattern is '0'. Therefore, a peak at the state $[111100000] = 480$, as observed in Fig. 3, is expected.

III. Computing Inverse of a Logic Function using PSL

A distinguishing feature of a PSL is that it is invertible: given an output, PSL can compute the possible inputs. As an illustration, the results for computing an inverse of a function for an XOR gate and a 2:1 MUX is shown in Fig. 4. The states corresponding to the inputs that result in the given output show distinct peaks.

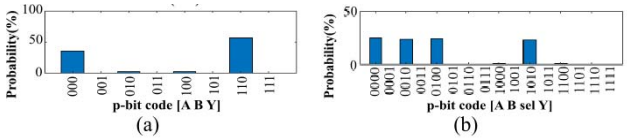

Fig. 4. **Invert mode operation:** (a) **XOR gate:** when Y is clamped to '0', then the inputs fluctuate among two possible states $[(A, B) = (00), (11)]$ with almost equal probabilities. (b) **2:1 MUX:** when Y is clamped to '0', then A, B, sel fluctuate between '0' and '1' such that $(A, B, sel, Y) = (0000), (0010), (0100)$ and (1010) states show peaks. Histogram is obtained by averaging over 10000 samples.

IV. Conclusion

In this work, we have proposed a methodology to implement a given Boolean function in CNF using PSL. We have also demonstrated the capability of a PSL in computing the inverse of a given Boolean function.

Acknowledgment

The authors wish to thank Dr. Supriyo Datta and Dr. Kerem Camsari for stimulating discussions and helpful suggestions.

References

[1] K. Y. Camsari, R. Faria, B. M. Sutton, and S. Datta, "Stochastic p-bits for invertible logic," *Phys. Rev. X*, vol. 7, p. 031014, Jul 2017.

[2] J. D. Biamonte, "Nonperturbative k-body to two-body commuting conversion Hamiltonians and embedding problem instances into Ising spins," *Phys. Rev. A*, vol. 77, p. 052331, May 2008.

[3] L. Personnaz, I. Guyon, and G. Dreyfus, "Collective computational properties of neural networks: New learning mechanisms," *Phys. Rev. A*, vol. 34, pp. 4217–4228, Nov 1986.

[4] B. Sutton, K. Y. Camsari, B. Behin-Aein, and S. Datta, "Intrinsic optimization using stochastic nanomagnets," *Scientific Reports*, vol. 7, p. 44370, Mar 2017.