Spintronics-based Reconfigurable Ising Model Architecture

Ankit Mondal and Ankur Srivastava
University of Maryland College Park
College Park, Maryland, United States
amondal2@terpmail.umd.edu,ankurs@umd.edu

ABSTRACT

The Ising model has been explored as a framework for modeling NP-hard problems, with several diverse systems proposed to solve it. The Magnetic Tunnel Junction (MTJ)-based Magnetic RAM is capable of replacing CMOS in memory chips. In this paper, we propose the use of MTJs for representing the units of an Ising model and leveraging its intrinsic physics for finding the ground state of the system through annealing. We design the structure of a basic MTJ-based Ising cell capable of performing the functions essential to an Ising solver. A technique to use the basic Ising cell for scaling to large problems is described. We then go on to propose Ising-FPGA, a parallel and reconfigurable architecture that can be used to map a large class of NP-hard problems, and show how a standard Place and Route tool can be utilized to program the Ising-FPGA. The effects of this hardware platform on our proposed design are characterized and methods to overcome these effects are prescribed. We discuss how two representative NP-hard problems can be mapped to the Ising model. Simulation results show the effectiveness of MTJs as Ising units by producing solutions close/comparable to the optimum, and demonstrate that our design methodology holds the capability to account for the effects of the hardware.

KEYWORDS

Ising Model, NP-hard Problems, Magnetic Tunnel Junctions, Reconfigurable Architectures (FPGA), Simulated Annealing

1 INTRODUCTION

Computing efficiency is becoming increasingly limited by memory bandwidth, which lags far behind processor computing speeds. Several real world problems come under the category of combinatorial optimization and are NP-hard, for eg. the travelling salesman problem, graph coloring, etc. This means that the problems are not computationally scalable with traditional von Neumann computing methods [25]. The capabilities provided by non-von Neumann architectures have motivated research [13, 17, 31] on accelerating the process of solving such problems.

The Ising model [11], a mathematical model to describe interactions between magnetic spins, can be leveraged to express and formulate many NP-hard problems due to the combinatorial nature of the model. It consists of a system of spins which can take one of 2 possible values $\{1, -1\}$. These spins interact with one another in such a way that the system gradually evolves to a minimum energy state, representing a solution to the NP-hard problem that it encodes.

The computational complexity of the Ising model has long been explored and investigated, and so has been the search for efficient hardware systems [8–10, 18, 24, 33] for solving combinatorial problems. For example, the process of quantum annealing [6, 18] naturally holds the capability to solve the Ising model, which requires the system to move out of local minima so as to continue converging to the ground state. However, the quantum technology is far from reaching maturity in terms of a large-scale commercial use due to its requirement of operating superconducting devices at very low temperatures. CMOS-based implementations [33] of Ising solvers have also been looked at, including the use of GPUs [12] for exploiting the inherent parallelism of Ising computations. However, some of these have made use of extra hardware [16, 24] or memory [12] for generating random numbers to simulate annealing properties in the model. Further, the Ising model often requires a large number of connections among Ising spins, which has led to the use of techniques such as cell cloning in fixed 2-D spin arrays [16], or to retaining only the nearest neighbor connections [33] leading to sub-optimal outcomes.

Recent work [29–31] has investigated the use of spintronic (nanomagnetic) devices for emulating the behavior of Ising spins by exploiting their natural physics. The work in [31] demonstrates through simulations such capability in stochastic nanomagnets operating at very high speeds; but these had very low energy barriers, implying that in reality they can suffer from fabrication complexity, read disturbs, and inability to write to several other Ising spins. Shim et al. [30] have used Magnetic Tunnel Junctions (MTJs) with higher energy barriers as Ising spin devices. Such stable MTJs form the central component of Spin Transfer Torque Magnetic RAM (STT-MRAM), the spintronic non-volatile memory which is replacing CMOS technology in cache and embedded memories [32]. However, they limit Ising spin connectivity to only the (four) nearest neighbors, and restrict their interactions to binary. Although this strategy yields a simple design, it severely limits the nature and size of NP-hard problems that can be encoded onto the hardware. The work in [29] does not detail how the influences from different units, in the form of voltages, would be added up.

In our work, we propose to evaluate an Ising model computing platform based on stable MTJs which tackles simultaneously several of the aforementioned issues not addressed in previous work. Our contributions are as follows:

- We design the hardware of an Ising cell, where an MTJ represents an Ising unit, and show how it can perform Ising computations.
- We demonstrate how a cell with fixed no. of inputs can be slightly modified to make it scalable to large problems.
- We then propose Ising-FPGA, a parallel and reconfigurable architecture composed of several of these Ising cells, and having an interconnect topology similar to an FPGA.
- We analyze the degradation in signals in the hardware platform to get a more realistic picture of such implementations, and attempt to take them into account while mapping an NP-hard problem.

2 PRELIMINARIES

2.1 The Ising Model

The Ising model was originally developed to study the behavior of ferromagnets and consists of a number of spin units (ferromagnetic elements) with pairwise interactions [11]. The energy of the system is described by the Ising Hamiltonian

$$H(x) = -\sum_{i,j}^{N} J_{ij} x_i x_j - \sum_{i}^{N} h_i x_i \tag{1}$$

where N is the no. of units, x_i is the spin of the i^{th} unit and can assume one of 2 values, say '+1' (up spin) and '−1' (down spin), J_{ij} is the coefficient of pairwise interaction between the i^{th} and the j^{th} units, and h_i is a bias term accounting for external fields. The model considers a symmetric J, implying a reciprocal nature of the interactions. Also, there are no self-interactions, thus $J_{ii} = 0$.

Solving the system involves finding a configuration x of the spin units that minimizes the energy H. Obtaining this ground state is an NP-hard problem due to the discrete nature of x_i, and this property of the Ising model has enabled the mapping of several combinatorial optimization problems to it [22]. The ground state of the spins represents the solution of the NP-hard problem it encodes.

The energy due to a single unit x_i and its connections, called the local Hamiltonian, is expressed as[12]

$$H(x_i) = -\sum_{j}^{N} J_{ij} x_j x_i - h_i x_i \tag{2}$$

which considers the interactions with its neighbors and its bias. Each step in the process of finding the ground state of the system involves lowering the local Hamiltonian of each unit *in parallel* which can be done by simply changing the state of x_i if that helps lower $H(x_i)$. However, the system would soon get stuck in a local minima rather than converging to the global optimum. The way out of this is to randomly perturb the system and allow it to go to a higher energy state for the time being - a popular concept known as (simulated) annealing.

2.2 Magnetic Tunnel Junction

The MTJ is an emerging non-volatile spintronic device technology. The Magnetic RAM, which is based on the MTJ, is a viable candidate for replacing CMOS as the basic element of future embedded memory [32]. MTJs possess 2 stable states depending on the relative magnetizations of its ferromagnetic layers - Parallel (P) and Anti-Parallel (AP). The P state exhibits a lower resistance than the AP state ($R_P < R_{AP}$).

The state of the MTJ can be changed by passing spin-polarized current of appropriate polarity [14] via the mechanism of spin-transfer torque [21]. The magnetization dynamics of the MTJ is governed by the stochastic Landau-Lifshitz-Gilbert (LLG) equation [21, 30]. It accounts for the effect of the spin current I_s, and also includes the random field H_{th} due to thermal noise [30] which is uncorrelated in all 3 directions. Thus, the time required to switch the MTJ from the P state to the AP state (or vice versa) is heavily dependent on the magnitude of the switching current. Not only that, the random thermal noise H_{th} causes fluctuations in the initial magnetization angle and also affects the switching behavior [21]. Therefore, the switching process is stochastic in nature, implying that a current pulse of given amplitude and duration has only a certain probability to successfully change the state. Fig. 1 illustrates the probabilistic switching characteristics for $P \rightarrow AP$ and $AP \rightarrow P$ for a current pulse of $2ns$. [1]

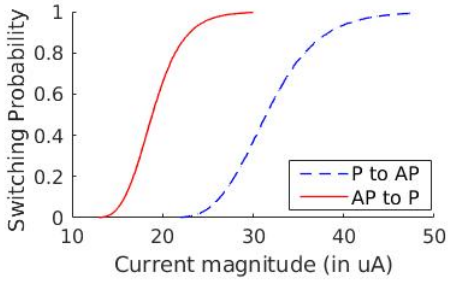


Figure 1: Switching probabilities of the MTJ with $2ns$ pulse width. Note that current polarities would be opposite for $P \rightarrow AP$ and $AP \rightarrow P$.

3 ISING-FPGA FRAMEWORK

An NP-hard problem with N variables requires N Ising units, implying an $O(N^2)$ connectivity among the units. Also, the specific nature/type of the connections depends on the problem itself. We therefore envision a reconfigurable MTJ-based architecture which allows a large class of Ising models to be implemented. To this end, we leverage the advancements made in the FPGA technology to propose a similar architecture for our Ising-model hardware platform, and call it the *Ising-FPGA*. In this paper, we present the design of such an

[1] The asymmetry in the current requirements for the 2 directions is because the spin transfer efficiency for $AP \rightarrow P$ is higher than that for $P \rightarrow AP$ [14].

MTJ-based Ising-FPGA possessing a routing network similar to regular FPGAs. We develop techniques which account for the effects of the hardware platform in the Ising model.

It must be noted that the Ising-FPGA is only an architecture, consisting of an array of MTJs, which exhibits reconfigurability and has a routing topology similar to FPGAs. It serves the purpose of mapping problems which can be formulated as the Ising model. The Ising-FPGA is *not* a standard FPGA, with some components are made of MTJs, and which is to be used for mapping digital logic functions.

3.1 Solving the Ising model

The local Hamiltonian in eqn. 2 tells us how the spin of an Ising unit should be modified towards lower energy. Taking the negative of derivative of both sides, we get

$$-\frac{\partial H(x_i)}{\partial x_i} = \sum_{j}^{N} J_{ij}x_j + h_i = \beta_i \ (say) \tag{3}$$

where β_i represents the cumulative *influence* on the i^{th} unit by the other units (all x_j). The sign of β_i at a certain time step decides the direction in which x_i should be updated to lower the local energy. For eg. if $x_i = -1$, and $\beta_i > 0$, x_i should be switched to +1 (otherwise it should remain at −1). Algorithm 1 summarizes the process of solving the Ising model.

Algorithm 1 Annealing process for the Ising model

```
Initialize all x_i randomly from {−1, 1}
for n = 1 to iters do                          ▷ perform iters iterations
    for i = 1 to N do                          ▷ do parallely for each Ising unit
        Calculate β_i from eqn. 3. Assign x'_i = sign(β_i)
        x'_i = −x'_i with probability p << 1   ▷ flipping randomly with a small probability
    end for
    Assign x = x' and reduce p.
end for
```

3.2 MTJ as an Ising spin unit

In our work, we propose using an MTJ to realize an Ising spin unit since it has 2 stable states, just as is required of an Ising unit. Other non-volatile devices such as RRAMs and PCMs tend to have several intermediate states [34], and therefore, the MTJ is a better choice. It forms the central component of a basic cell of our MTJ-based Ising-FPGA. We exploit its probabilistic switching characteristics to guide the entire system of spins through the states which reduce the energy of the system ($H(x)$ in eqn. 1), with the goal of reaching the ground state.

For the MTJ-based Ising unit, we can encode the direction and probability with which it should switch in the polarity and magnitude respectively of the switching current provided to it. Considering the gradient in eqn. 3, the write current passed through the i^{th} unit may be written as

$$I_i = I_{min} + \frac{\beta_i}{k}(I_{max} - I_{min}) \tag{4}$$

where I_{min} is the minimum current provided to overcome the soft threshold below which the switching probability is negligible, k is a normalizing factor to ensure that I_i is bounded by a maximum current I_{max}. We choose values of I_{min} and I_{max} that correspond to probabilities of roughly 0.1% and 98% respectively for a $2ns$ pulse duration. For $P \rightarrow AP$, $I_{min} = -22\mu A$, $I_{max} = -44\mu A$, and for $AP \rightarrow P$, $I_{min} = 13\mu A$, $I_{max} = 26\mu A$.

Once the Ising unit's MTJ is updated probabilistically using the write current in eqn. 4, we can allow the magnetization a while to settle, and then read the value stored in the MTJ by passing a small current (say $< 5\mu A$) through it and sensing the potential drop across it [19]. This value read would then be used to update the states of the *other* spins in the next iteration. The effect of random noise in the system can be realized by passing a small current I_{RF} which flips the MTJ with a small probability and, once again, letting it relax. Fig. 2 depicts the timeline of these stages where the Random Flip of a spin unit is done according to its own value read in the *previous* iteration, but before the write stage to avoid another readout.

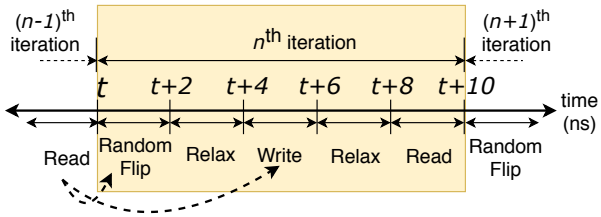


Figure 2: Different stages of an iteration in the process of finding the ground state of an Ising model. Each stage is of duration $2ns$, and hence an iteration takes $10ns$. The dashed arrows show where the spin value read is utilized.

3.3 MTJ-based Ising-FPGA cell

Let us now describe the structure of an Ising spin cell, which is the basic unit of our hardware platform, and show how eqn. 4 would be realized. Each Ising cell corresponds to one spin variable and houses the MTJ whose state represents the value of the spin. It is responsible for (a) receiving the states of the other spin units and writing to its MTJ with a certain current, (b) reading the state of its MTJ, and also (c) flipping it randomly.

The coefficients of interactions (J_{ij}) between spin units can be represented by variable resistors, and the summation in eqn. 3 can be obtained through an op-amp with $N-1$ inputs. Fig. 3 shows the Ising cell in a system with 5 variables. In this figure, we specifically illustrate the Ising cell of variable x_1. It receives binary voltage signals $V_2 \ldots V_5 \in \{-V_m, V_m\}$ from the cells of the other variables $x_2 \ldots x_5$, where the voltage polarity represents their spin values (V_m for +1 and $-V_m$ for −1). These input voltages are modulated by the resistors $R_{12} \ldots R_{15}$ and fed to the positive terminal of an op-amp OA1, along with an internal bias voltage V_{h1} through R_{h1}. The output V_o of the op-amp OA1, with feedback resistor R_f, is provided to

the MTJ write control circuit shown within the dashed box. It regulates the direction of current I_1 through the MTJ with the help of a pair of switches. These are controlled by the output of comparator OA2 (in open loop configuration) which turns on one and only one of the two switches. The switch controlled by WR is turned on in the Write stage. Voltages V_+ and V_- of opposite polarity are added to V_o to offset it and obtain the minimum current I_{min} for $AP \rightarrow P$ and $P \rightarrow AP$ respectively.

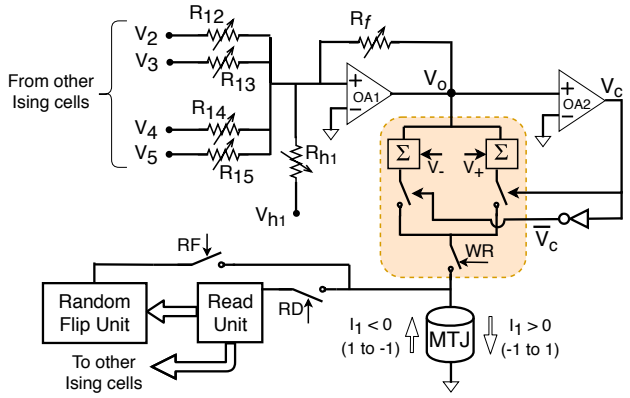


Figure 3: The proposed Ising spin cell. Switches WR, RD and RF are turned on in the Write, Read and Random Flip stages respectively. Parameters $V_+ = 0.227V$, $V_- = -0.172$, obtained with HSPICE simulations using $V_m = 0.4V, R_P = 5.2k\Omega, R_{AP} = 13.7k\Omega$.

The output V_o of op-amp OA1 can be expressed as

$$V_o = -R_f \left(\sum_{j=2}^{5} \frac{V_j}{R_{1j}} + \frac{V_{h1}}{R_{h1}} \right) = -R_f \left(\sum_{j=2}^{5} V_j G_{1j} + V_{h1} G_{h1} \right) \quad (5)$$

where G denotes the respective conductances. The above relation resembles eqn. 3 suggesting that a weighted sum of the outputs from other Ising cells can be easily obtained through an op-amp and resistors. The conductances $G_{ij} \in [G_{min}, G_{max}]$ would be directly proportional to the magnitude of the interaction coefficient, $|J_{ij}|$. If all J_{ij} are normalized such that $|J_{ij}| \leq 1$, then $G_{ij} = |J_{ij}| G_{max}$. To implement bipolar J_{ij}, we can simply add an inverter to each of the $(N-1)$ inputs of x_i's cell to make both V_j and $-V_j$ available, and choose from between the two.

The state of the MTJ is sensed by and stored in the Read unit which then provides voltage signals to the other cells (in the next cycle) accordingly. The Random Flip unit sends current I_{RF} to the MTJ to flip it with a small probability, the direction of the current being dependent on the state stored in the Read Unit.

3.4 Splitting inputs to multiple cells

Any non-von Neumann hardware platform designed for solving an Ising-like problem would have a fixed number of inputs per Ising cell, however might it be implemented - spintronics-based [29–31] or otherwise [24, 33]. Even our Ising-FPGA has a fixed number of inputs per cell. As the problem grows in size, this is going to pose a limitation to the no. of connections made from/to the Ising cells.

Our approach to dealing with limited fan-in Ising cells is a cascading of several of these cells to accommodate as many inputs as required. The analog nature of the computation in eqn. 5 allows for this divide-and-conquer approach with only a small addition to the basic Ising cell. This is in the form of another op-amp OA3. Fig. 4(a) shows the modified Ising cell with $I = 4$ inputs $V_a \ldots V_d$. It can output either from its OA3 or from its Read Unit as required.

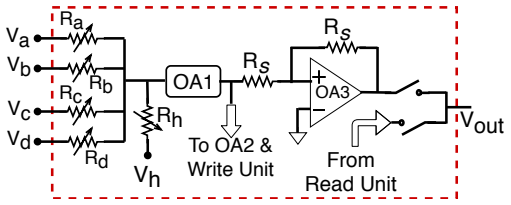
Now, considering a fan-in of $I = 4$ per cell, let us show how we can split inputs to a spin variable into multiple Ising cells for an Ising system consisting of 9 variables. The idea is to have several layers/levels (named A,B,…) of the basic Ising cell connected in a tree-like sequence, with outputs from the cells of one level fed into inputs of a cell in the next level until the number of inputs remaining is less than or equal to the fan-in of each cell. The programmable quantities in these cells would be set as required (depending on their level). Fig. 4(b) shows how we can split the inputs $V_2 \ldots V_9$ into 2 Ising cells (at level A) which then feeds into the last level cell of variable x_1.

The outputs of the cells shown in fig. 4(b) would be

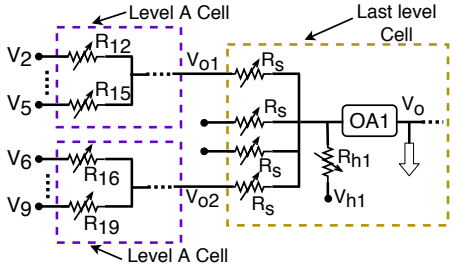
$$V_{o1} = R_s \left(V_2 G_{12} + \ldots V_5 G_{15} \right) \quad \& \quad V_{o2} = R_s \left(V_6 G_{16} + \ldots V_9 G_{19} \right) \quad (6)$$

$$V_o = -R_f \left(\frac{V_{o1}}{R_s} + \frac{V_{o2}}{R_s} + \frac{V_{h1}}{R_{h1}} \right) = -R_f \left(\sum_{j=2}^{9} V_j G_{1j} + V_{h1} G_{h1} \right) \quad (7)$$

where V_o is the output of last level cell's OA1, and is as desired.



(a) OA3 added to the Ising cell



(b) Splitting 8 inputs into 2 cells

Figure 4: (a) The Modified Ising cell. (b) Multi-level Ising cells. Observe that $R_{12} \ldots R_{19}$ are in level A, but R_{h1} is in the last level cell.

4 ARCHITECTURE OF THE ISING-FPGA

Field Programmable Gate Arrays (FPGAs) are integrated circuits that offer easy re-programmability, allowing the implementation of any desired logic function [26]. VTR/VPR [5, 23] is an open-source platform for modeling and analyzing FPGA architecture and CAD. The reconfigurable routing topology of the FPGA is a good match for the kind of network connectivity exhibited by an Ising model-based platform such as the one proposed above.

4.1 Reconfigurable Ising model hardware

Let us discuss the analogous of the FPGA's hardware for our Ising-model solver (that is, the Ising-FPGA) and then explain how part of VPR's software flow can be used for configuring the design.

Ising-FPGA: Each Configurable Logic Block (CLB) of an FPGA corresponds to an Ising cell with multiple inputs and one output which can be either the output of OA3 or from the Read Unit. The no. of inputs to the CLB is set to the no. of inputs to the Ising cell. Thus, for eg. fig. 4(b) shows 3 CLBs, each with $I = 4$ inputs. The architecture file (*.xml*) of the FPGA was used to describe certain parameters of the Ising-FPGA.

The connections between cells is captured by the reprogrammable connectivity of the Ising-FPGA. For our analog design, we can use muxes based on transmission gates (TGs) as switches in the Switch Box (SB), in a way very similar to directional SBs [20]. Thus, a connection between 2 cells has one TG for each SB that it passes through.

Using VPR for Ising-FPGA: VPR produces a *.blif* file that describes the netlist of the synthesized network, and uses it to perform place and route of the design. We build a BLIF File Generator (BFG) which takes in the no. of spin variables (N) and the fan-in of each Ising cell (I) as inputs, and creates a *.blif* file by connecting Ising cells in a hierarchical way as demonstrated earlier. Since the *.blif* file should specify only those connections that exist, the BFG also takes in the Ising graph, which lists the pairs of variables (i, j) which have a non-zero interaction ($J_{ij} \neq 0$). VPR uses this *.blif* netlist to pack, place and route the design. It outputs a *.route* file (among many others) that contains the design's routing information.

4.2 Signal Degradation and Recovery

The use of TGs for switches in our analog design implies that their finite resistance will result in a potential drop across it, and also bring down the current that was supposed to flow into an Ising cell. We estimate this degradation in every path (from each source cell to its destinations) of the circuit and show how we can recover the original signal.

We consider a linear model for the signal degradation, in the sense that the total resistance offered by a path is directly proportional to its length and is independent of the length of other paths (if any) from the same source cell. We use the information provided in the *.route* file to find the length of the path for each $(src, dest)$ pair in the design.

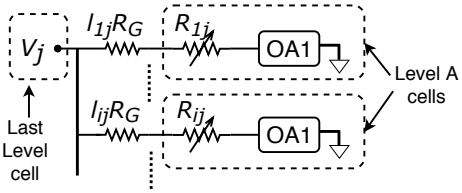


Figure 5: Signal degradation model for paths from a last level cell (source) to level A cells (destinations).

Let us look at the degradation in current before the Level A of Ising cells. Fig. 5 shows a net from the last level cell of x_j to many level A cells, all with different path lengths. Consider for now the path to level A cell of x_i having length l_{ij}. The current flowing through the input resistance R_{ij} should ideally be V_j/R_{ij}. The presence of TGs, each with resistance R_G, in the path means that this current is now going to be $V_j/(l_{ij}R_G+R_{ij})$. To get back the original current level, we can simply reduce the input resistance R_{ij} by $l_{ij}R_G$ subject to a minimum. The new resistance $\overline{R}_{ij}$ is given as

$$\overline{R}_{ij} = \begin{cases} (R_{ij} - l_{ij}R_G) & \text{if } (R_{ij} - l_{ij}R_G) \geq R_{min}/J_{max} \\ R_{min}/J_{max} & \text{otherwise} \end{cases} \quad (8)$$

where $R_{min} = 1/G_{max}$ and $J_{max} \geq 1$ is the largest interaction coefficient for the equivalent of the smallest possible $\overline{R}_{ij}$. Because $\overline{R}_{ij}$ may not still be low enough, we can increase the magnitude of V_j for recovering the desired current. Since different destinations would have different path lengths from the source, they would require to boost V_j by different amounts. Let δ_i^j be the increment in V_j required by the i^{th} destination. Equating the desired and obtained currents,

$$\frac{V_j(1+\delta_i^j)}{\overline{R}_{ij} + l_{ij}R_G} = \frac{V_j}{R_{ij}} \quad \Rightarrow \quad \delta_i^j = \frac{\overline{R}_{ij} + l_{ij}R_G}{R_{ij}} - 1 \quad (9)$$

For any source j, the amount of boosting is decided by the destination having the highest value of δ ($\delta_{max}^j = \max_i \delta_i^j$). This boosting can be performed by amplifying the output voltage of the source cell's Read unit through suitable circuits. No extra routing is required for this modification.

Now that V_j has been boosted by δ_{max}^j, the new connection resistances can be obtained yet again by substituting δ_{max}^j in eqn. 9. This gives us the final value of the resistors as

$$\overline{R}_{ij} = R_{ij}(1 + \delta_{max}^j) - l_{ij}R_G \quad (10)$$

For the next level of signal propagation, that is from the output of level A cell to the input of next level's cell, the source connects to only a single destination. Thus, any modifications at the source will depend only on the path for this $(src, dest)$ pair, and can be done by increasing the feedback resistance R_s of the OA1 in the level A cell of the *src*.

5 ISING GRAPHS OF NP-HARD PROBLEMS

5.1 Maximum Cut

Given an undirected graph $G(V, E)$, the Max-cut problem's objective can be stated mathematically as [22]

$$maximize \ \frac{1}{2} \sum_{i,j \in V} W_{ij}(1 - x_i x_j) \tag{11}$$

where W_{ij} is the weight of the edge between the i^{th} and j^{th} vertices, and $x_i, x_j \in \{-1, 1\}$ indicate which partition they belong to. Clearly, this objective can be mapped to the Ising Hamiltonian in eqn. 1 by choosing $J_{ij} = -W_{ij}/\max_{ij} |W_{ij}|$.

5.2 Travelling Salesman Problem

The TSP is another well-known NP-hard problem which, given N cities and their locations, seeks to find a tour of minimum distance such that each city must be visited exactly once. The Ising formulation of the TSP has a system of N^2 spin variables. The Ising Hamiltonian is given as [22]

$$H = \sum_{v=1}^{N} \left(1 - \sum_{j=1}^{N} x_{v,j}\right)^2 + \sum_{j=1}^{N} \left(1 - \sum_{v=1}^{N} x_{v,j}\right)^2 + \lambda \sum_{uvj} W_{uv} x_{u,j} x_{v,j+1} \tag{12}$$

Here the first 2 terms ensure that the constraints on the solution to a problem (each city visited exactly once) are satisfied, for which $J_{(v,j)(u,i)} = -1$ whenever $u = v$ or $i = j$. The last term corresponds to the distance travelled in the tour, with W_{uv} being the distance between cities u and v, and λ is a proportionality constant to make sure that the constraints are never violated in favor of a shorter tour, for which the condition $\lambda < 1/\max W(u, v)$ should be satisfied. We have $J_{(v,j)(u,i)} = d_{min}/W_{uv}$, whenever $i = j - 1$ or $j + 1$, where d_{min} is the minimum distance between any pair of cities.

6 SIMULATION SETUP AND RESULTS

Fig. 6 depicts the entire flow for simulation and evaluation. First, the nature and parameters of the NP-hard problem are input to the Graph Generator which outputs the interaction matrix J and also the Ising graph. The Ising graph is input to the BLIF File Generator (BFG) which creates the $.blif$ file according to the no. of variables (N) and the no. of inputs per Ising cell (I) (sec. 4.1). Then, VPR uses the $.blif$ and $.xml$ files to Place and Route the design. The resultant $.route$ file is analysed to obtain the lengths of the path between each pair of connected Ising cells, which is accordingly used to find the degradation in the signals and the modifications necessary in the design (sec. 4.2 - Signal Degradation and Recovery - SD&R). This information is passed on to the Stochastic LLG solver along with various other parameters such as the number of iterations to perform, various current values, etc. The LLG simulations of the MTJ were performed using an HSPICE model[2] [4, 15] which was imported into MATLAB for scalability.

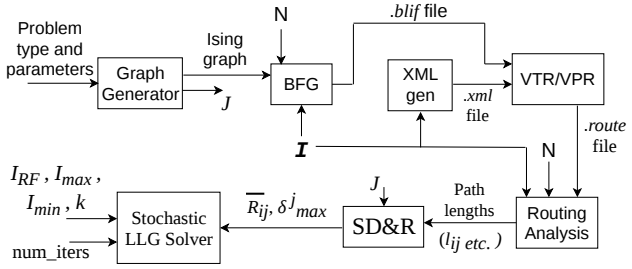


Figure 6: Steps performed in the simulations. We start with the Graph Generator and end with the Stochastic LLG simulations.

The current I_{RF} for Random Flipping (sec. 3.2) was chosen in a way that it corresponds to roughly 1% switching probability at the beginning of the simulations (at the 1^{st} iteration), and was then reduced linearly to a value that corresponded roughly to 0.1% probability at the end. This is equivalent to the theoretical notion of annealing, which requires "cooling the system".

With regard to accounting for the effects of the hardware, simulations were performed for 3 situations:

- Ideal - Not considering the effects of the underlying hardware, i.e. ignoring signal degradation.
- With Signal Degradation (SD) - Considering the effect of the finite resistances of the paths in the Ising-FPGA, taking $R_G = 3.45k\Omega, R_{min} = 50k\Omega$, but not recovering from the issue.
- Recovery (Rec) - The modifications made in the design to recover the original signals (using $\overline{R}_{ij}, \delta^j_{max}$) with $J_{max} = 10$.

Let us now present the results of the simulations performed for the 2 NP-hard problems. For each of these, we mention the usage of the significant hardware components in the Ising-FPGA. These include

(1) the total no. of Ising cells in the Ising-FPGA,
(2) the minimum Channel Width Factor (CWF), (the minimum no. of tracks per channel for successful routing),
(3) the average length of the paths from the last level cells to the Level A cells (average of all l_{ij} - fig. 5) at this CWF.

Max Cut: Table 1 specifies the graphs that were used for benchmarking along with their no. of vertices, the best cut value (obtained using an SDP solver [1]) and the type & range or distribution of edge weights. Table 2 lists the aforementioned Ising-FPGA parameters at the specified Ising cell fan-in (I). Also included is an estimate of the power consumption (in mW) of the system obtained through HSPICE. Fig. 7 shows the obtained cut values for the 4 graphs, each normalized by their respective best cut values in table 1. Each of the graphs was run 10 times, with 1000 iterations of the Ising simulations

[2]Device parameters: MTJ cell dimension - $22nm \times 22nm \times 1.5nm$, damping constant $\alpha = 0.01$, simulation time step $\delta_t = 0.01ns$, saturation magnetization $M_s = 800emu/cm^3$

Name	Source	Verts	Best Cut	Weight Type & Range
$G1$	G1 from G-set [2]	800	11429	Binary ($\{0, 1\}$)
$G2$	Custom	140	2598.65	Fraction: $U \sim [0, 1]$
$G3$	w01_100.0 from Biq mac [3]	100	645	Integer in $[-10, 10]$
$G4$	ising2.5-300_5555 from [3]	300	8.569×10^6	Int in $[-2, 2] \times 10^5$

Table 1: Descriptions of graphs for Maxcut simulations.

per run; all maxcut values are thus average of 10 runs. It is evident that the Ideal maxcut values obtained by simulating the Ising model are very close to the best cut values obtained by heuristics (especially for graphs G1 and G2), thereby revealing the potential of an Ising solver.

Name	G1	G2		G3	G4
I	32	16	32	8	8
No. of cells	2398	1400	840	216	1044
Min. CWF	138	48	48	26	20
Avg. Path Lengths	23.2	8.3	8.3	10.6	5.8
Power	52.37	13.39	14.81	1.02	5.065

Table 2: Ising-FPGA hardware usage for Max Cut. Power in mW.

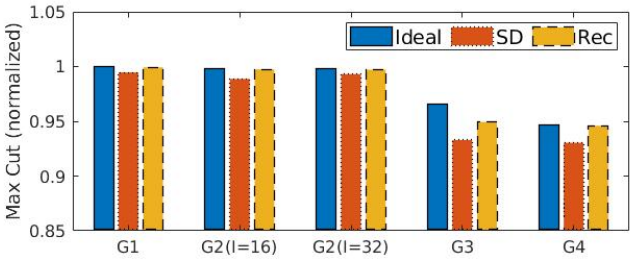


Figure 7: Max cut values (normalized) from the Ising simulations for the 4 graphs, with 2 different values of Ising cell fan-in (I) used for G2.

From the data pertaining to fig. 7, Signal Degradation (SD) leads to an average relative drop of 1.43% in the MaxCut values. If we define the extent of recovery in the maxcut values as $(Rec - SD)/(Ideal - SD)$, the average recovery across graphs was 78.48%. From table 2, we see that a larger fan-in (I) reduces the no. of Ising cells of graph G2 as expected. The minimum CWF and the Average Path Lengths vary in different ways depending on the nature of the graph.

TSP: Three example problems were considered from a dataset [7, 28] - P01, GR17 and FRI26, sets of 15, 17 and 26 cities with optimal tour lengths of 291, 2085 and 937 respectively. Table 3 lists the hardware usage on the Ising-FPGA with $I = 16$. Ising simulations were run 20 times (each having 2000 iterations) for each city set. Table 4 mentions the results in terms of the no. of runs (out of 20) in which at least 1 "valid" tour was discovered and the average of their Minimum Tour Length (MTL). SD results in an increase in the MTL by an average of 5.86% as compared to Ideal, but, more importantly, it reduces the chances of finding a valid tour. With our recovery strategy, the no. of valid tours is almost as many as those in the Ideal case and the MTL is only 3.49% higher than Ideal on an average.

Name	No.of cells	Min. CWF	Avg.Path lengths	Power
P01	1125	48	8.72	8.17
GR17	1445	48	8.59	12.58
FRI26	5408	60	14.3	45.13

Table 3: Ising-FPGA hardware usage for TSP. Power is in mW.

Additionally, fig. 8 compares the average no. of valid tours found in each run for the cases Ideal, SD and Rec. Due to SD, this value dropped by an average of 76.83% compared to the Ideal, again indicating reduced chances of finding a valid tour. We could recover an average of 83.78% of this drop.

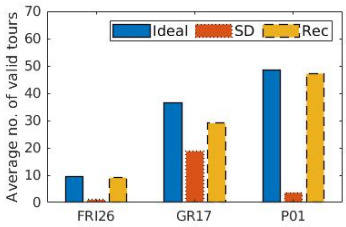


Figure 8: Average (over 20 runs) no. of valid tours found in a run.

City set		P01	GR17	FRI26
Valid	Ideal	20	19	16
	SD	9	12	6
	Rec	20	19	19
MTL	Ideal	443	3448	2262
	SD	450	3765	2416
	Rec	453	3689	2290

Table 4: Results of Ising simulations for TSP.

7 DISCUSSION

Let us now briefly analyze some aspects of our proposed approach and make comparisons with related work.

- **Propagation delay:** Each stage of opamp induces a delay of about $20ps$ (from Cadence Virtuoso simulations). With 3 stages (OA1 & OA3 of level A, and OA1 of last level), the expected propagation delay of Ising spin signals $\pm V_m$ in the write stage is about $0.06ns$. However, this delay could be subsumed within the relax stage just before the write. Further, any minor variations in delay from Ising cell to cell is unlikely to affect the entire system or the final solution, since randomness is an essential part of the Ising computations.
- Resistive RAMs (RRAMs) are a suitable candidate for realizing the variable resistors that capture the interactions between Ising units. These are memristive devices [10, 34] that offer multiple levels of resistance and easy re-programmability.
- Pervaiz et. al. [27] propose the implementation of probabilistic circuits, based on unstable stochastic units called probabilistic bits, on FPGAs. These can be used for Ising and quantum computations. Their entire implementation is on a real FPGA (and is therefore completely based on digital CMOS logic and memory). On the contrary, our work proposes an FPGA-like architecture based on spintronic and memristive devices so that their inherent randomness and in-memory computing capabilities can be harnessed for realizing an Ising solver. It is expected to have a much smaller area footprint than a fully digital implementation such as [27]. Since the authors of that work do not report any figures on the area or power consumption of their design, we are unable to make any detailed analysis.

Research on hardware implementations of Ising model typically focuses on the possibility of mapping such models and on solving the associated optimization problem to

obtain answers. There is not much emphasis on the characterization of system area/power/performance (yet).
- Process variations in MTJs and RRAMs isn't expected to affect the Ising system to any significant extent, again because such variations add to the randomness in the system which it anyway requires.

8 CONCLUSION

In this paper, we proposed an Ising model architecture based on MTJs, which can be used to map and solve NP-hard problems. We discuss realistic hardware implementations in terms of Ising spin cells and their read/write capabilities, network topology, and re-programmability of interactions among spin units to allow different kinds of NP-hard problems to be encoded. We present Ising-FPGA, a parallel and reconfigurable architecture which can be configured using a standard FPGA Place and Route tool, and discuss ways to incorporate the non-idealities in the hardware into the Ising model.

ACKNOWLEDGEMENT

This work was supported by the National Science Foundation (NSF) under Grant 1642424.

REFERENCES

[1] 2002. Computational Optimization Laboratory. https://web.stanford.edu/~yyye/yyye/coplsdp.zip
[2] 2003. The G-set benchmark. https://web.stanford.edu/~yyye/yyye/Gset/
[3] 2007. The Biq Mac Library. http://biqmac.uni-klu.ac.at/biqmaclib.html
[4] 2016. The LLG module. https://nanohub.org/groups/spintronics/llg_thermal_noise
[5] 2016. VTR documentation. https://docs.verilogtorouting.org/en/latest/
[6] 2018. D-Wave Systems. http://www.dwavesys.com.html
[7] 2018. Data for the Traveling Salesperson Problem. https://people.sc.fsu.edu/~jburkardt/datasets/tsp/tsp.html
[8] Md Z Alom et al. 2017. Quadratic Unconstrained Binary Optimization (QUBO) on neuromorphic computing system. In *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 3922–3929.
[9] S Bhanja et al. 2016. Non-Boolean computing with nanomagnets for computer vision applications. *Nature nanotechnology* 11, 2 (2016), 177.
[10] M N Bojnordi et al. 2016. Memristive boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning. In *2016 IEEE Int. Sym. on High Performance Computer Architecture (HPCA)*. 1–13.
[11] B A Cipra. 1987. An introduction to the Ising model. *The American Mathematical Monthly* 94, 10 (1987), 937–959.
[12] C Cook et al. 2018. GPU Based Parallel Ising Computing for Combinatorial Optimization Problems in VLSI Physical Design. *arXiv preprint:1807.10750* (2018).
[13] K Corder et al. 2018. Solving Vertex Cover via Ising Model on a Neuromorphic Processor. In *2018 IEEE Int. Sym. on Circuits and Systems*. 1–5.
[14] Z Diao et al. 2005. Spin transfer switching and spin polarization in magnetic tunnel junctions with MgO and AlOx barriers. *Applied Physics Letters* 87, 23 (2005).
[15] S Ganguly et al. 2016. Evaluating spintronic devices using the modular approach. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* 2 (2016), 51–60.
[16] H Gyoten et al. 2018. Area efficient annealing processor for Ising model without random number generator. *IEICE Trans. on Information and Systems* 101, 2 (2018).
[17] H Gyoten et al. 2018. Enhancing the solution quality of hardware ising-model solver via parallel tempering. In *2018 IEEE/ACM ICCAD*. 1–8.
[18] V Kumar et al. 2018. Quantum annealing for combinatorial clustering. *Quantum Information Processing* 17, 2 (2018), 39.
[19] H Lee et al. 2015. Design of a fast and low-power sense amplifier and writing circuit for high-speed MRAM. *IEEE Trans. Mag.* 51, 5 (2015), 1–7.
[20] G Lemieux et al. 2004. Directional and single-driver wires in FPGA interconnect. In *2004 IEEE Int. Conf. on Field-Programmable Technology*.
[21] Z Li et al. 2003. Magnetization dynamics with a spin-transfer torque. *Physical Review B* 68, 2 (2003), 024404.
[22] A Lucas. 2014. Ising formulations of many NP problems. *Frontiers in Physics* 2 (2014), 5.
[23] J Luu et al. 2014. VTR 7.0: Next generation architecture and CAD system for FPGAs. *ACM Trans. Reconfigurable Tech. and Systems (TRETS)* (2014).
[24] S Matsumoto et al. 2018. RRAM/CMOS-Hybrid Architecture of Annealing Processor for Fully Connected Ising Model. In *2018 IEEE Int'l Memory Workshop*. IEEE, 1–4.
[25] F Neumann et al. 2010. Combinatorial optimization and computational complexity. In *Bioinspired Computation in Combinatorial Optimization*. Springer, 9–19.
[26] H Parvez et al. 2010. *Application-specific mesh-based heterogeneous FPGA architectures*. Springer Science & Business Media.
[27] A Z Pervaiz et al. 2018. Weighted p-Bits for FPGA Implementation of Probabilistic Circuits. *IEEE trans. Neural Networks & Learning Sys.* (2018).
[28] G Reinelt. 1991. TSPLIB: A traveling salesman problem library. *ORSA journal on computing* 3, 4 (1991), 376–384.
[29] S Sharmin et al. 2017. Magnetoelectric oxide based stochastic spin device towards solving combinatorial optimization problems. *Scientific Reports* 7, 1 (2017), 11276.
[30] Y Shim et al. 2017. Ising computation based combinatorial optimization using spin-Hall effect (SHE) induced stochastic magnetization reversal. *Journal of Applied Physics* 121, 19 (2017), 193902.
[31] B Sutton et al. 2017. Intrinsic optimization using stochastic nanomagnets. *Scientific Reports* 7 (2017), 44370.
[32] L Thomas et al. 2017. Basic Principles, Challenges and Opportunities of STT-MRAM for Embedded Memory Applications. *MSST 2017* (2017).
[33] M Yamaoka et al. 2016. A 20k-spin Ising chip to solve combinatorial optimization problems with CMOS annealing. *IEEE Journal of Solid-State Circuits* 51, 1 (2016).
[34] Mohammed A Zidan et al. 2018. The future of electronics based on memristive systems. *Nature Electronics* 1, 1 (2018), 22.