Ising-FPGA: A Spintronics-based Reconfigurable Ising Model Solver

ANKIT MONDAL and ANKUR SRIVASTAVA, University of Maryland College Park

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 article, 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. The hardware overhead of the Ising model is analyzed, and 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 three representative NP-hard problems can be mapped to the Ising model. Further, we suggest ways to simplify these problems to reduce the use of hardware and analyze the impact of these simplifications on the quality of solutions. 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.

CCS Concepts: • **Hardware → Spintronics and magnetic technologies**; *Hardware accelerators*; • **Computer systems organization** → *Architectures*; • **Theory of computation → Probabilistic computation**;

Additional Key Words and Phrases: Combinatorial optimization, np-hard problems, magnetic tunnel junction, reconfigurable architecture, simulated annealing

ACM Reference format:
Ankit Mondal and Ankur Srivastava. 2020. Ising-FPGA: A Spintronics-based Reconfigurable Ising Model Solver. *ACM Trans. Des. Autom. Electron. Syst.* 26, 1, Article 4 (September 2020), 27 pages.
https://doi.org/10.1145/3411511

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

A preliminary version of the article [1] has appeared in the International Symposium on Quality Electronic Design (ISQED), March 2020.
This work is supported by the National Science Foundation (NSF) under Grant 1642424.
Authors' addresses: A. Mondal and A. Srivastava, University of Maryland College Park, 8223 Paint Branch Drive, College Park, Maryland 20742; emails: amondal2@terpmail.umd.edu, ankurs@umd.edu.

1084-4309/2020/09-ART4 $15.00
https://doi.org/10.1145/3411511

combinatorial optimization and are NP-hard, for, e.g., the travelling salesman problem, graph coloring, and so on. This means that the problems are not computationally scalable with traditional von Neumann computing methods [2]. The lack of a known polynomial time algorithm forces us to resort to approximate methods for large-scale problems, often sacrificing the quality of solution for faster computations. The capabilities provided by non–von Neumann architectures have motivated research [3–5] on accelerating the process of solving such problems.

The Ising model [6], 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 that can take one of two possible values $\{1, -1\}$. These spins interact with one another in such a way that decreases the energy of the system. By reducing the temperature of the system sufficiently slowly, one can reach the global optimum, representing the best solution to the NP-hard problem that it encodes.

The computational complexity of the Ising model has long been explored and investigated [7] and so has been the search for efficient hardware systems [8–13] for solving combinatorial problems. For example, the process of quantum annealing [14, 15] 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 [9] of Ising solvers have also been looked at, including the use of GPUs [16] for exploiting the inherent parallelism of Ising computations. However, some of these have made use of extra hardware [10, 17] or memory [16] 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 two-dimensiona (2D) spin arrays [17], or to retaining only the nearest-neighbor connections [9] leading to sub-optimal outcomes.

Recent work [3, 18, 19] has investigated the use of spintronic (nanomagnetic) devices for emulating the behavior of Ising spins by exploiting their natural physics. These devices have only two stable states, and their stochastic switching dynamics make them a potential candidate for the hardware realization of Ising spins. The work in Reference [3] 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. [18] 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 that is replacing CMOS technology in cache and embedded memories [20]. 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. Both Reference [3] and Reference [19] assume a fully connected network of these magnetic devices without any consideration for the cost or feasibility of such high connectivity. The work in Reference [19] 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 that tackles simultaneously several of the aforementioned issues not addressed in previous work. Our contributions can be summarized 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 number of inputs can be slightly modified to make it scalable to large problems.

(a) A general Ising graph **(b)** Ising Hamiltonian

Fig. 1. (a) An Ising graph with five spin units. (b) The system transitions from state 1 to 2, a local minima. Random perturbations can take it to state 3 so that it transitions to 4 and can eventually reach global optimum 5.

- 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. Allowing flexibility in the connections on the hardware platform would enable the mapping of different kinds of problems on the same piece of hardware.
- 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.
- In addition, we also propose approaches to smartly simplify the Ising connectivity graph for different NP-hard problems to reduce hardware usage or to satisfy any resource constraint, and characterize the resulting tradeoff with solution quality.

2 PRELIMINARIES

In this section, we provide the background of the Ising model and the Magnetic Tunnel Junction and mention related work in this field.

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 [6]. The energy of the system is described by the Ising Hamiltonian and is given as

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

where N is the number of units; x_i is the spin of the ith unit and can assume one of two values, say "+1" (up spin) and "−1" (down spin); J_{ij} is the coefficient of pairwise interaction between the ith and the jth units; and h_i is a bias term accounting for external fields. Figure 1(a) shows a complete Ising graph with $five$ units. Note that 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 [21]. The ground state of the spins represents the solution of the NP-hard problem it encodes.

Theoretically, the probability of finding the system in a particular state x is given as [6]

$$P(x) = \frac{e^{-H(x)/(k_B T)}}{\sum_y e^{-H(y)/(k_B T)}}, \tag{2}$$

(a) Stable states of an MTJ **(b)** MTJ switching probability

Fig. 2. (a) Schematic of an MTJ in P and AP states. (b) Switching probabilities of the MTJ with 2-ns pulse width. Note that current polarities would be opposite for $P \rightarrow AP$ and $AP \rightarrow P$. Data obtained from MTJ Stochastic LLG simulations [22, 23] in HSPICE, at steps of 0.1 μA with 10,000 points per step. Device parameters are listed in Table 1.

where k_B is the Boltzmann constant and T is the temperature of the system. At high temperatures, the system explores the solution space and is almost equally likely to be found in any state [6], whereas at low temperatures, states with lower energy would dominate. Ideally, the system should start from a high temperature and be slowly cooled down—a process known as annealing—so that it eventually reaches the ground state.

The underlying parallelism in the model can be exploited while searching for the ground state. The energy due to a single unit x_i and its connections, called the local Hamiltonian, is expressed as [16]

$$H(x_i) = -\sum_{j}^{N} J_{ij} x_j x_i - h_i x_i, \tag{3}$$

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. Figure 1(b) depicts the energy landscape with the local and global minima, and demonstrates the effect of 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 [20]. It has demonstrated high write speeds, good scalability, high endurance, and compatibility with CMOS. The MTJ essentially comprises two ferromagnetic layers (CoFeB) and a thin tunnel barrier (typically MgO) sandwiched between them. The magnetic orientation of one of the ferromagnetic layers is *pinned* in a particular direction, whereas that of the other is *free*, as illustrated in Figure 2(a).

MTJs possess 2 stable states where the relative magnetizations of the free and pinned layers (FL and PL respectively) are Parallel (P) and Anti-Parallel (AP), respectively, with the P state exhibiting a lower resistance than the AP state ($R_P < R_{AP}$). It is this difference in resistance that allows a single-bit value to be encoded in the MTJ.

The magnetic orientation of the MTJ's FL can be changed by passing spin-polarized current of appropriate polarity [24]. This spin current exerts a torque on the magnetization of the FL, a mechanism known as spin-transfer torque [25], and can flip its direction. The magnetization

dynamics of the FL is governed by the stochastic Landau-Lifshitz-Gilbert (LLG) equation[1]:

$$\frac{d\hat{m}}{dt} = -\gamma\left(\hat{m}\times\vec{H}_{eff}\right) + \alpha\left(\hat{m}\times\frac{d\hat{m}}{dt}\right) + \frac{1}{qN_s}\left(\hat{m}\times\hat{m}\times I_s\hat{M}\right). \tag{4}$$

It is a time-differential equation capturing the effect of the net magnetic field and spin current on the FL magnetization vector [18, 25]. It is common to have a heavy metal layer below the MTJ, which converts charge current to spin current with high efficiency through the Spin Hall Effect [18]. Regardless of the exact method, the LLG equation always holds.

The LLG equation accounts for the effect of the spin current I_s and also includes the random field H_{th} due to thermal noise [18], 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. In addition, the random thermal noise H_{th} causes fluctuations in the initial magnetization angle and also affects the switching behavior [25].

Therefore, the switching process is stochastic in nature, implying that a current pulse of given amplitude and duration has only a certain probability of successfully changing the state. This stochasticity does not arise from defects or variations in the device, but is an intrinsic property of the STT switching. Figure 2(b) illustrates the probabilistic switching characteristics for $P \rightarrow AP$ and $AP \rightarrow P$ transitions for a current pulse of 2 ns. The asymmetry in the current requirements for the two directions is because the spin transfer efficiency (which characterizes how much charge current is converted to spin current) for $AP \rightarrow P$ is higher than that for $P \rightarrow AP$ [24].

Under specific conditions, the MTJ switching behavior can be different from what is suggested by the LLG equation. Here we list two such deviations that are commonly referenced and well known in the literature.

- Subvolume Effect in MTJs: When MTJs are scaled below a size of 40 nm diameter, a new phenomena other than the conventional macrospin theory arises. This results in a higher than expected switching spin-torque efficiency [26, 27]. It is due to the slightly different magnetization properties at the interface between the free layer and the oxide barrier. As per the macrospin limit, efficiency $\kappa = E_b/I_{c0}$, where E_b is the thermal energy barrier and I_{c0} is the critical switching current. Reduction in MTJ size beyond a certain limit tends to increase the energy barrier (as compared to what it would have been given its size) and decrease the critical current slightly. This leads to more efficient switching activity, with power savings and longer data retention and improves the scope of MTJ-based memories (MRAMs) to take off in the future.
- Thermal activation switching: The LLG equation for the MTJ magnetization dynamics is typically used when the switching current is above the critical current I_{c0}. It is known as the precessional (high-speed) regime. But, for currents below the critical value, thermal activation plays an important role. The switching is governed by the Neel-Brown law [25, 26], and the switching probability distribution is given as [28, 29]

$$P_{sw}(I_s, t) = 1 - exp\left[-\frac{t}{\tau_0}exp\left\{-\Delta\left(1-\frac{I_s}{I_{c0}}\right)^2\right\}\right], \tag{5}$$

  where t is the switching pulse width, τ_0 is the attempt time ($\approx$1 ns), and Δ is the thermal stability factor.

[1]Where $\hat{m}$ is the unit vector along the direction of FL magnetization, γ is the electron gyromagnetic ratio, $\vec{H}_{eff}$ is the effective magnetic field acting on the FL, α is the Gilbert damping constant, I_s is the spin current, $\hat{M}$ is the PL magnetization, and $N_s = M_s V/\mu_B$ is the number of spins in the FL, with M_s being the saturation magnetization of the FL material, V the FL volume, and μ_B the Bohr magneton.

2.3 Related Work

The idea of leveraging the Ising model to represent combinatorial optimization problems and developing specific hardware systems for solving them (or similar problems) has been explored in several works. Here we summarize some of the recent ones (done in the last few years), most of which tend to depart from conventional computing methods.

A Fully CMOS-based design: Yamaoka et al. [9] design an Ising chip with 20,000 spins in CMOS technology and fabricated using a 65*nm* process. They discuss its potential to obtain better solutions than heuristic algorithms implemented on von-Neumann computers. The topology is a 3D lattice implemented physically as a 2D array of CMOS SRAM circuits, with each spin unit being connected to five5 others. Interactions are ternary $(-1, 0, +1)$ in nature, and Ising computations are realized using XOR gates and analog majority voter circuits. Randomness for annealing is achieved by occasionally reducing the supply voltage of SRAM cells storing the spin values. The authors demonstrate their proposed circuit by running a custom max-cut problem.

Non-conventional source of randomness: Cook et al. [16] consider the implementation of an Ising model solver on a GPU. The natural randomness of GPU thread scheduling is exploited during the annealing process to randomize the update of Ising spins and improve GPU resource utilization. The authors of Reference [17] propose a novel shift-register-based spin flipper (SRSF) that helps the annealing process to converge without using area-expensive random number generators. However, the SRSF introduces a bias in the annealing process by introducing different temperatures for different Ising spins. Only binary interaction values and 2D grid torus structures are considered.

Exploiting RRAM crossbars: In Reference [10] the use of an RRAM crossbar architecture is proposed for a part of Ising computations involving dot products. Remaining operations are performed using digital CMOS components; hence the architecture is an RRAM/CMOS hybrid. It is shown to achieve better power efficiency than the fully digital architecture of Gyoten et al. [17]. A hardware accelerator based on Boltzmann machine was proposed in Reference [13] to solve NP-hard problems and realize Deep Belief Networks. It too exploited the in-situ computing capability of a 1-Transistor 1-Resistor memristive array.

Using Neuromorphic Computing: Corder et al. [4] demonstrate the use of a neuromorphic processor, which had erstwhile been used mostly for implementing spiking neural networks or enabling learning-based systems to operate, for mapping and solving the Ising model. The mapping requires some pre-processing in terms of satisfying fan-in and fan-out constraints, which was done via approximate graph-partitioning, and solving time-dependency constraints (scheduling node update), which was done through approximate graph coloring. Experimental results for the vertex cover problem were shown to be competitive with that of an approximate algorithm.

Alom et al. [8] also perform NP-hard optimizations of the Quadratic Unconstrained Binary Optimization (QUBO) type, which can be expressed using the Ising model, on the IBM Neurosynaptic TrueNorth system. Experiments were performed on general graph problems and potential applications are discussed such as image segmentation, density estimation of vehicles and cellular network users, and monitoring of weather and super-computing centers. Fonseca et al. [30] develop a software framework to demonstrate the solving of constraint satisfaction problems on the SpiNNaker neuromorphic chip through stochastic search of the solution space.

FPGAs for invertible circuits: Unstable stochastic units called probabilistic bits (p-bits) can be used to construct probabilistic circuits that solve a large class of optimization and inference problems. Each p-bit produces an output m that is related to its input I, with some randomness, as

$$m(t) = sign\{rand[-1, 1] + tanh(I(t))\}, \tag{6}$$

where I is obtained through a dot product, a weighted sum of other p-bits. The authors of Reference [31] propose the implementation of such a circuit on FPGAs. Weight values are represented using fixed point arithmetic and the activation function $tanh$ is realized with an LUT. Thresholding of input I is done to avoid exceeding the input range of the LUT with the help of comparators and a MUX. A 32-bit LFSR is used for the random number generation. 16 weighted p-bits can be implemented in a 4×4 tile that will allow the realization of a 16×16 weight matrix. Each of the 16 p-bits are updated sequentially for proper system operation. An AND gate and a full adder are demonstrated within a tile in floating and invertible modes, whereas larger circuits such as a 32-bit RCA required a cascade of several system tiles.

Use of spintronic devices: Recent works [3, 12, 18, 19] have considered nanomagnetic devices for realizing Ising spin units to take advantage of their intrinsic physics. A conceptual demonstration was given by Sutton et al. [3] by simulating stochastic nanomagnets with very low thermal stability; emphasis was not laid on implementation beyond the device level. Multiferroic oxide devices exhibiting magnetoelectric effect were utilized for Ising spins in the work by Sharmin et al. [19]. Voltage controlled spin units were envisioned to reduce current flow in the design and improve scalability. LLG simulations of Travelling Salesman Problem were performed, but circuit level details were not discussed. In Reference [18], MRAMs were the candidates for Ising units due to their stability and their possibility for being integrated with CMOS circuits. These works have proved that nanomagnetic devices are suitable for representing Ising units. However, they had their own limitations, as described in Section 1, which we try to overcome in our work.

Other works in the literature (not counting those based on the quantum phenomena) include taking inspiration from the dynamics of Cellular Neural Networks (CNNs) [32], developing analog circuits implementing a continuous-time dynamical system [33], and using networks of coupled oscillators [34].

3 ISING-FPGA FRAMEWORK

An NP-hard problem with N variables requires N Ising units, implying an $O(N^2)$ connectivity among the units. Such full connectivity was assumed by some prior works [3, 19] demonstrating Ising model hardware; even if we build ASICs to solve a specific NP-hard problem, such an overhead is substantial and could be impractical.

We envision a reconfigurable MTJ-based architecture that 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 article, we present the design of such an MTJ-based Ising-FPGA possessing a routing network similar to regular FPGAs. We develop techniques that account for the effects of the hardware platform in the Ising model. Also, due to limited computation units and routing resources, problems beyond a specific size are unable to be exactly implemented on such a platform (just as in a regular FPGA). Therefore, we propose Ising model simplification strategies that allow for the best use of the FPGA resources.

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 and solving problems that can be formulated using the Ising model. The Ising-FPGA is *not* a standard FPGA, with some components being made of MTJs, which is to be used for mapping digital logic functions.

3.1 Solving the Ising Model

The local Hamiltonian in Equation (3) tells us how the spin of an Ising unit should be modified toward lower energy. Taking the negative of derivative of both sides, we get

$$-\frac{\partial H(x_i)}{\partial x_i} = \sum_{j}^{N} J_{ij}x_j + h_i = \beta_i \ (say), \tag{7}$$

where β_i represents the cumulative *influence* on the ith 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 example, if $x_i = -1$, and $\beta_i > 0$, then x_i should be switched to $+1$ (otherwise it should remain at -1). This is similar to a gradient descent approach, although it must be noted that x_i can only be binary. Algorithm 1 summarizes the process of solving the Ising model. After all spin units are initialized randomly (line 1), each iteration involves calculating influence β_i (line 4), modifying the spin value accordingly (line 7), and then flipping it randomly with a small probability (line 8) to enable escaping from local minima (Figure 1(b)). Observe that both the inner **for** loops can be executed in parallel for the N units.

ALGORITHM 1: Annealing process for the Ising model

```
1:  Initialize all x_i randomly from {−1, 1}
2:  for n = 1 to iters do
3:      for i = 1 to N do
4:          Calculate β_i from Equation (7)
5:      end for
6:      for i = 1 to N do
7:          x'_i = sign(β_i)
8:          x'_i = −x'_i with probability p << 1
9:      end for
10:     Assign x = x' and reduce p.
11: end for
```

3.2 MTJ as an Ising Spin Unit

The stochastic switching characteristics of the MTJ has been an impediment to the realization of energy-efficient STT-MRAM based memory chips [20]. However, many applications where computations can be non–von Neumann in nature, particularly neuromorphic computing, have leveraged this same characteristic of spintronics to obtain better performance than traditional CMOS-based methods [35]. The absence of stochasticity in CMOS memory/logic necessitates the use of pseudo-random number generators to mimic probabilistic behavior.

In our work, we propose using an MTJ to realize an Ising spin unit, since it has two stable states, just as is required of an Ising unit. Other non-volatile devices such as Resistive RAMs and Phase Change Memory devices tend to have several intermediate states [36], 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 that reduce the energy of the system ($H(x)$ in Equation (1), with the goal of reaching the ground state. Additionally, when the system gets stuck in a local energy minima, the same characteristic would also be used to get it out of the minima. The system of Ising spin units realized with the MTJs would involve interactions among units through voltages and currents that would depend on the parameters of the encoded NP-hard problem and the present state of the system. Details of the implementation shall be discussed shortly.

3.2.1 Finding a Local Optimum. The magnitude of β_i in Equation (7) provides the extent to which x_i can lower the energy of the system, and is thus indicative of the probability with which x_i should change its state (if necessary). 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

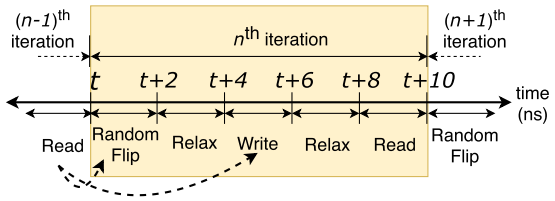


Fig. 3. Different stages of an iteration in the process of finding the ground state of an Ising model. Each stage is of duration 2 ns, and hence an iteration takes 10 ns. The dashed arrows show where the spin value read is utilized.

switching current provided to it. Although the switching characteristics vary non-linearly with the current as per Figure 2(b), we can perform a linear mapping for simplicity as follows. Considering the gradient in Equation (7), the write current passed through the ith unit may be written as

$$I_i = I_{min} + \frac{\beta_i}{k}(I_{max} - I_{min}), \tag{8}$$

where I_{min} is the minimum current provided to overcome the soft threshold below which the switching probability is negligible and k is a normalizing factor to ensure that I_i is bounded by a maximum current I_{max}. Naturally,

$$k = \max_i(\max|\beta_i|) = \max_i\left(\sum_j^N |J_{ij}| + |h_i|\right), \tag{9}$$

which is largest possible influence on any unit.

Once the Ising unit's MTJ is updated probabilistically using the write current in Equation (8), 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 μA) through it and sensing the potential drop across it [37]. 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} that flips the MTJ with a small probability and, once again, letting it relax. Ideally, the probability with which this random flip occurs should go down with time to maintain an equivalence with the theoretical notion of annealing, which is "cooling the system." Hence the current I_{RF} must also reduce in magnitude after each cycle.

Figure 3 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. Thus in each cycle/iteration, some of the spin units get updated depending on their interactions with the rest. The entire system evolves through several iterations and the Ising energy reduces over time when observed on a large scale, since occasional increases must be expected due to the random flipping.

3.2.2 Device and Circuit Parameters. We choose values of I_{min} and I_{max} that correspond to switching probabilities of roughly 0.1% and 98%, respectively, for a 2-ns pulse duration. For $P \rightarrow AP$, $I_{min} = -22$ μA, $I_{max} = -44$ μA, and for $AP \rightarrow P$, $I_{min} = 13$ μA, $I_{max} = 26$ μA. Note that directly feeding the current obtained from the analog dot product to the MTJ eliminates the use of ADCs.

The parameters of the MTJ chosen are summarized in Table 1. The critical switching current of the MTJ was calculated to be $I_{c0}^{P\rightarrow AP} = 17.93$ μA and $I_{c0}^{AP\rightarrow P} = 8.49$ μA using the following formula

Table 1. MTJ Device Parameters

Quantity	Symbol	Value
Size of free layer	$a \times a \times t_F$	22 nm × 22 nm × 1.5 nm
Damping constant	α	0.01
Saturation Magnetization	M_s	800 emu/cm^3
Magnetic Anisotropy	H_k	2.25×10^3 Oe
Polarization factor	P	0.6
MTJ resistances	R_P and R_{AP}	5.2 and 13.7 kΩ, respectively
TMR ratio	$(R_{AP} - R_P)/R_P$	1.63

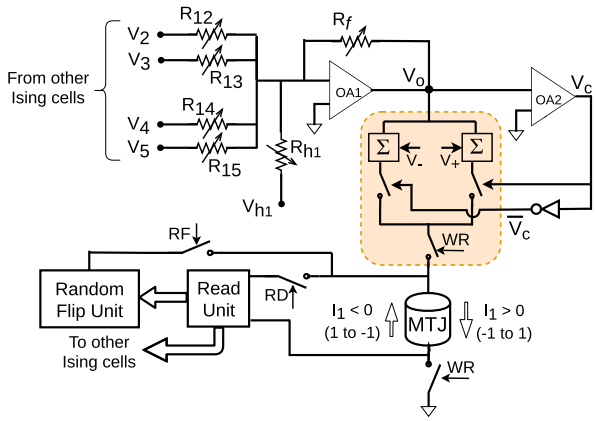


Fig. 4. The proposed Ising spin cell. Switches WR, RD, and RF are turned on in the Write, Read, and Random Flip stages, respectively. The dashed boundary includes the MTJ write control circuit. Power rails exist for positive, negative, and ground ($0V$) voltages.

for the critical switching current density:

$$J_{c0} = \frac{2e\mu_0\alpha M_s H_k t_F}{\hbar\eta}, \tag{10}$$

where e is the charge of an electron, μ_0 is the permittivity of vacuum, and η is the spin transfer efficiency given as $\eta^{P\to AP} = 0.5P/(1+P^2)$ and $\eta^{AP\to P} = 0.5P/(1-P^2)$ [24].

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 Equation (8) 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 Equation (7) can be obtained through an op-amp with $N-1$ inputs. Figure 4 shows the Ising cell in a system with five 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. Assume without loss of generality that the MTJ can be switched (probabilistically) from

- $AP \rightarrow P$ (that is $-1 \rightarrow 1$) if $V_o > 0$ ($\Rightarrow I_1 > 0$), and
- $P \rightarrow AP$ (that is $1 \rightarrow -1$) if $V_o < 0$ ($\Rightarrow I_1 < 0$)

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

$$V_o = -R_f \left(\sum_{j=2}^{5} \frac{V_j}{R_{1j}} + \frac{V_{h1}}{R_{h1}} \right) = -R_f \left(\sum_{j=2}^{5} V_j G_{1j} + V_{h1} G_{h1} \right), \tag{11}$$

where G denotes the respective conductances. The above relation resembles Equation (7) 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 $G_{ij} = |J_{ij}| G_{max}$. The minus sign in Equation (11) due to the inverting nature of the op-amp shall be taken care of in a moment.

The value of the feedback resistance R_f is dependent on the number of inputs to the Ising cell and the desired maximum current I_{max}. For an N-variable Ising model, with each Ising cell having $(N-1)$ inputs, we can use the maximum influence in Equation (9) to calculate the largest possible magnitude of V_o as

$$V_o^{max} = -R_f (k \times (-V_m) G_{max}) = R_f V_m k G_{max}. \tag{12}$$

Thus, R_f would depend on V_o^{max}, which is in turn decided by I_{max}. In Figure 4, parameters $V_+ = 0.227V$, $V_- = -0.172$, $V_o^{max} = 0.184V$, obtained with HSPICE simulations using $V_m = 0.4V$, and values of I_{max} and I_{min} mentioned previously.

The state of the MTJ is sensed by and stored in the Read unit in the cell that then provides voltage signals to the other cells (in the next cycle) accordingly. The Read Unit would be similar to a conventional read circuit (say, with a pre-charge sense amplifier). The Random Flip unit sends the current I_{RF} to the MTJ to flip it with a small probability, wherein the direction of the current is dependent on the state stored in the Read Unit.

3.3.1 Realizing the Sign of J. The interaction between any 2 Ising spin variables x_i and x_j can be either ferromagnetic, with $J_{ij} > 0$, favouring same spin values of x_i and x_j, or anti-ferromagnetic, with $J_{ij} < 0$, favouring opposite spin values of x_i and x_j. But the resistors that represent these interactions are always positive values. To implement bipolar J, 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. Figure 5 shows how this is done for the example in Figure 4.

3.4 Splitting Inputs to Multiple Cells

The Ising model thus involves each variable communicating its value to all other variables. This suggests an $O(N)$ growth in the number of inputs to each Ising cell, and $O(N^2)$ growth in the total number of connections in the most general case. This may be manageable for a small-sized problem, but wiring congestion would definitely be a major hindrance to scalability for large problems.

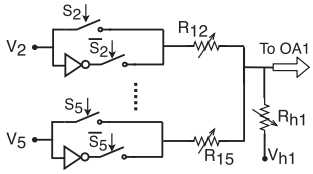
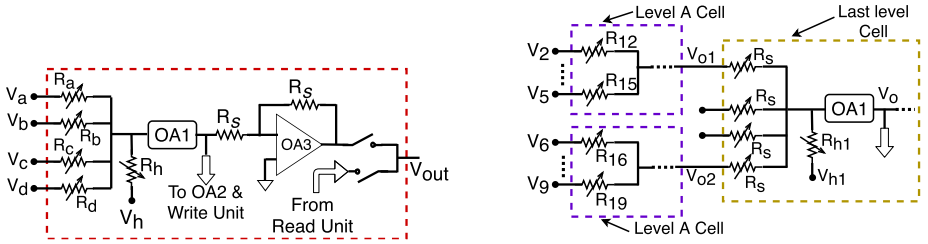


Fig. 5. Inverters at cell inputs for capturing both +ve and -ve interactions.



(a) OA3 has been added to the Ising cell **(b)** Splitting the 8 inputs into 2 cells

Fig. 6. (a) The Modified Ising cell. (b) Multi-level Ising cells. Observe that $R_{12} \ldots R_{19}$ are in Level A, whereas R_{h1} is in the last level cell.

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 [3, 18, 19] or otherwise [9, 10]. 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 number of connections made from/to the Ising cells, even if routing may not be an issue.

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 Equation (11) 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 that lets us revert back to the original sign of the dot product in Equation (11). Figure 6(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 nine variables. The idea is to have several layers/levels (named A, B,...) of the basic Ising cell connected in a treelike 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. Figure 6(b) shows how we can split the inputs $V_2 \ldots V_9$ into two Ising cells (at Level A) that then feeds into the last level cell of variable x_1.

Each of the Ising cells would have identical structure and still retain the Write, Read and Random Flip (WR&RF) units, (not shown for simplicity). But there are certain differences in how the programmable quantities in the cells are set, and how each cell is operated, depending on its level as detailed in Table 2. The outputs of the cells shown in Figure 6(b) would be

$$V_{o1} = R_s\,(V_2G_{12} + \ldots V_5G_{15}) \quad \& \quad V_{o2} = R_s(V_6G_{16} + \ldots V_9G_{19}), \tag{13}$$

$$V_o = -R_f\left(\frac{V_{o1}}{R_s} + \frac{V_{o2}}{R_s} + \frac{V_{h1}}{R_{h1}}\right) = -R_f\left(\sum_{j=2}^{9} V_jG_{1j} + V_{h1}G_{h1}\right), \tag{14}$$

Table 2. Configuration of Ising Cells as per Their Level

Component/ Quantity	Level A,B... cells	Last level cell
WR&RF units & MTJ	Disabled (inactive)	Active (MTJ stores x_1)
Bias voltage (V_h)	$0V$	V_{h1} (desired value for x_1)
Output from	OA3	Read Unit
Output sent to	Next level cell	Level A cells of $x_2 \ldots x_9$
Feedback of OA1	Any value (say R_s)	R_f from Equation (12)
Input Resistors	For Level A: R_{ij}, Others: R_s	R_s
Bias Resistor (R_h)	Any value (don't care)	R_{h1} (desired bias for x_1)

Entries of last column specifically for variable x_1.

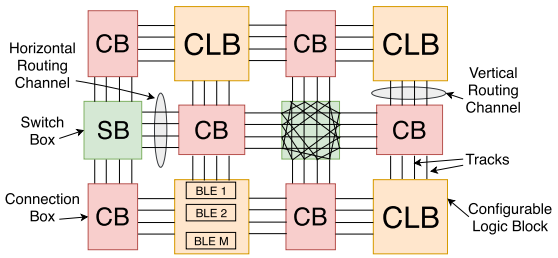


Fig. 7. The FPGA architecture. CLBs are connected through CBs and SBs.

where V_o is the output of OA1 in the last level cell. Note that Equation (14) is identical to Equation (11) indicating the same voltage to the Write control unit as required.

Ising-FPGA size: Thus, with the proposed approach of splitting the fan-in to several cells, the total number of levels for every spin variable is $\lceil log_I(N-1) \rceil$, and the number of cells dedicated to a single variable is $\approx (N-1)/(I-1)$. Hence, the total number of cells required (with N variables) is *quadratic* in N.

4 ARCHITECTURE OF THE ISING-FPGA

Field Programmable Gate Arrays (FPGAs) are integrated circuits that offer easy reprogrammability, allowing the implementation of any desired logic function [38], and have thus found use in different application domains. VTR/VPR [39, 40] is an open source platform for modeling and analyzing FPGA architecture and CAD. The 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. The flexibility of connections required by an Ising solver such as the Ising-FPGA can be fulfilled by the reconfigurability provided by an FPGA-like architecture.

4.1 Architecture of an FPGA

Let us first go over the basics of the FPGA architecture before describing how the proposed Ising-FPGA relates to it and how problems can be mapped to the latter. Figure 7 shows the architecture and the traditional interconnect topology of an FPGA. It consists of Configurable Logic Blocks (CLBs) each of which contains a cluster of Basic Logic Elements (BLEs). A BLE is made up of a k-input LUT and provides the LUT's output either directly or through a flip-flop. The interconnects in the FPGA are arranged in several horizontal and vertical channels all around the CLBs, each channel consisting of multiple tracks. The I/O pins of the CLBs are connected to the tracks of the adjacent channels through Connection Boxes (CBs). At the intersection of

Fig. 8. (a) The BFG creates the *.blif* file. (b) an excerpt from the *.blif* file responsible for variable x_1 specifying connections in Figure 6(b).

a vertical and a horizontal channel lies a Switch Box (SB) that is responsible for connecting the tracks of the channels incident on it, thereby facilitating communication between CLBs.

VTR/VPR [39, 40] is an open source platform for modeling and analyzing FPGA architecture and CAD. It takes in an architecture file (*.xml*) describing the FPGA, and the circuit's behavioral description in Verilog HDL, and produces an optimized netlist in the Berkeley Logic Interchange Format (BLIF) [41]. In the *.blif* file, a logic gate is declared with a *.names* keyword followed by its inputs and its output. The flip-flops in the BLEs are declared by a *.latch* statement. VPR uses this netlist to pack, place, and route the design. It outputs a *.route* file (among others) listing the size of the FPGA, the number of CLBs in use, and the connections (that is, the nets) from each Source to its Sinks, including the channels that the net passes through.

4.2 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: Herein each BLE of an FPGA corresponds to an Ising cell with multiple inputs and one output that can be either the output of OA3 or from the Read Unit, and each CLB contains only one BLE. The size of the LUTs, which in our case would be same as the number of inputs to the CLB, is set to the number of inputs to the Ising cell. Thus, for example, Figure 6(b) shows 3 BLEs (or CLBs), each with $I = 4$ inputs. The architecture file (*.xml*) of the FPGA was used to describe certain parameters of the Ising-FPGA.

Recall that each Ising cell in the last level outputs from its Read Unit, whereas cells in other levels output from their OA3. Thus, there is a continuous flow of signal (current) from the last level cell of a spin variable to that of another variable. The equivalent of this for an FPGA is that the BLEs representing last level cells were chosen to output from their flip-flops, while the rest could output straight from their LUTs. 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, in a way very similar to directional SBs [42]. Thus, a net has one TG for each SB that it passes through (similar to regular FPGAs).

Using VPR for Ising-FPGA: We build a BLIF File Generator (BFG) (Figure 8(a)) that takes in the number 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) that have a non-zero interaction ($J_{ij} \neq 0$). Using this Ising graph and value of I, the BFG determines how many spin variables each variable x_i is connected to, and how many Ising cells are required for making those connections. VPR uses this output *.blif* netlist to place and route the mapped Ising design.

Figure 8(b) shows a fragment of the *.blif* file generated for the connections[2] pertaining to Figure 6(b). Therein, $FF \sim j$ refers to the output from the last level cell of the jth spin variable,

[2]The latch does not indicate a connection from one cell to another and only serves the purpose of marking the end of the combinational circuit.

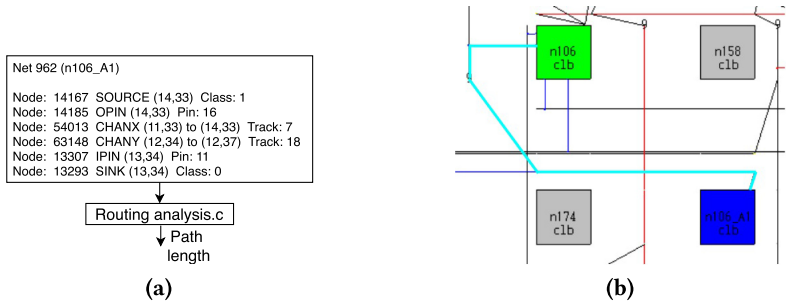


Fig. 9. Routing of nets. (a) The source, sink, and path of a net in textual form in the *.route* file. (b) View of the routing. The source is in dark blue, and the sink is in green. The net has been highlighted by us in sky blue.

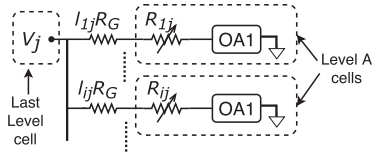


Fig. 10. Signal degradation model for paths from a last level cell (source) to Level A cells (destinations).

$n1\_A0$ and $n1\_A1$ are outputs of the Level A cells of x_1, and $n1$ is that of the last level cell of x_1. Hence, the first line describes the I/O nets of the Level A purple cell of Figure 6(b) taking inputs from the last level cells of x_2, x_3, x_4, x_5, and so on. Since the BFG only connects variable pairs specified in the Ising graph, if $J_{ij} = 0$, then $FF \sim i$ is not an input to any Level A cell of x_j (and vice versa) in the *.blif* file.

4.3 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 propose methods to recover the original signal.

Figure 9(a) shows the description of one net in the *.route* file provided by VPR. It specifies the location of the source, $n106\_A1$ at $(14, 33)$ in this example, the sections of X- and Y-channels that the net passes through, and the sink/destination ($n106$ at $(13, 34)$). The net is depicted in Figure 9(b). It crosses two SBs and hence has two TGs in its path, and we say it has a path length of 2. We use the information provided in the *.route* file to find the length of the path for each (*src, dest*) pair in the design.

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.

Let us first look at the degradation in current before the Level A of Ising cells. Figure 10 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

$$\overline{R}_{ij} = \begin{cases} (R_{ij} - l_{ij}R_G) & \text{if } (R_{ij} - l_{ij}R_G) \geq R_{min}/J_{max} \\ R_{min}/J_{max} & \text{otherwise,} \end{cases} \tag{15}$$

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 ith destination. Equating the desired and obtained currents,

$$\frac{V_j(1+\delta_i^j)}{\overline{R}_{ij} + l_{ij}R_G} = \frac{V_j}{R_{ij}} \quad \Rightarrow \quad \delta_i^j = \frac{\overline{R}_{ij} + l_{ij}R_G}{R_{ij}} - 1. \tag{16}$$

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 Equation (16). This gives us the final value of the resistors as

$$\overline{R}_{ij} = R_{ij}\left(1 + \delta_{max}^j\right) - l_{ij}R_G. \tag{17}$$

For the next level of signal propagation, that is from the o/p 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 GRAPH SIMPLIFICATION

In this section, we describe the combinatorial optimization problems that were mapped to the Ising model and demonstrated using our proposed architecture. A sizeable NP-hard problem has a large number of parameters and constraints that translate to lots of connections among Ising spin variables. In Section 3.4, the total number of Ising cells was calculated to be $O(N^2)$ for a general Ising graph with N nodes. Therefore, we develop strategies to simplify Ising graphs for reducing the hardware usage on the Ising-FPGA. Such simplification would have to done in a way that causes minimal changes to the solution to the problems.

5.1 Maximum Cut

Given an undirected graph $G(V, E)$, the Max-cut problem requires partitioning the vertices of G into 2 subsets S and $\overline{S}$ such that the total weight of the edges having one end in S and the other in $\overline{S}$ is maximized. Mathematically, this can be stated as [21]

$$maximize \ \frac{1}{2} \sum_{i,j \in V} W_{ij}(1 - x_i x_j), \tag{18}$$

where W_{ij} is the weight of the edge between the ith and jth vertices and $x_i, x_j \in \{-1, 1\}$ indicate whether these vertices belong to S or $\overline{S}$. Clearly, this objective can be mapped to the Ising Hamiltonian in Equation (1) by choosing $J_{ij} = -W_{ij}/\max_{ij}|W_{ij}|$ (recall that H is minimized, whereas the cut is maximized). This normalizes all the interactions to the range $[-1, 1]$. The bias terms h would be 0.

For any graph, edge weights having larger magnitude can affect the maxcut value to a higher extent by their presence or absence in the cut. Whereas edges having small weights cannot much

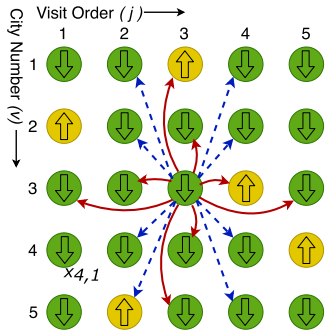


Fig. 11. Arrangement of Ising spin units for a five-city TSP. Arrows show interactions of $x_{3,3}$: Solid red ones enforce constraints, whereas dashed blue ones promote progress of tour. Current spin configuration corresponds to the tour of cities 2,5,1,3,4 in that order.

affect the maxcut value. Therefore, we can simplify the graph by ignoring edges with relatively smaller $|W_{ij}|$ (or $|J_{ij}|$). If f_s denotes the fraction of edges that are to be removed for simplification, then we propose to get rid of the bottom f_s fraction of the edges when sorted in decreasing order of $|W_{ij}|$. However, we always ensure that each vertex is connected to at least 1 other, lest it would be isolated from the graph and would not be influenced by any other vertex.

However, for a graph with no weights assigned to edges, no simplification is possible on this basis. Therefore, we choose to remove edges randomly, again maintaining a minimum degree of 1 for every vertex.

5.2 Travelling Salesman Problem

The TSP is another well-known NP-hard problem that, 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 as shown in Figure 11. Each row corresponds to a particular city and each column to a particular visit order. Thus $x_{v,j} = 1$ means that city v is visited in the jth order, whereas $x_{v,j} = 0$ (not -1, note the difference) implies it was not visited in the jth order. The Ising Hamiltonian is given as [21]

$$H = \sum_{v=1}^{N}\left(1 - \sum_{j=1}^{N} x_{v,j}\right)^2 + \sum_{j=1}^{N}\left(1 - \sum_{v=1}^{N} x_{v,j}\right)^2 + \lambda \sum_{uvj} W_{uv} x_{u,j} x_{v,j+1}. \tag{19}$$

Here the first two terms ensure that the constraints on the solution to a problem (each city visited exactly once) are satisfied. 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.

Any two spin units lying in the same row would not be in a "+1" state at the same time in any valid tour, because it violates the restriction that each city should not be visited more than once. Thus if $x_{v,j} = 1$, it should attempt to turn all $x_{v,i}$ off. Whereas if $x_{v,j} = 0$, it should not influence $x_{v,i}$ in any way (the latter would be decided by other terms of the Hamiltonian) [19]. A similar restriction is also imposed on all pairs of spin units lying in the same column. We therefore choose $J_{(v,j)(u,i)} = -1$ whenever $u = v$ or $i = j$.

The last term of H adds up the distances between the cities visited consecutively. If city v is travelled at the jth order, then it prompts all other cities to be visited just before and after it;

that is, order $j-1$ and $j+1$. The influence in this case is proportional inversely to the distance between the cities, because cities close by are more likely to be visited consecutively in an optimal solution. We therefore choose $J_{(v,j)(u,i)} = d_{min}/W_{uv}$, whenever $i = j-1$ or $j+1$, where d_{min} is the minimum distance between any pair of cities.

While trying to simplify the TSP Ising graph, we cannot do away with the interactions that enforce the restrictions for a valid tour (solid arrows in Figure 11). But we can try to remove those that prompt the travelling between cities (dashed arrows), which is equivalent to removing the paths between some pairs of cities. The best candidates for this are pairs of cities located far away, since they are unlikely to be travelled consecutively in the optimal tour. Therefore, we can sort the $N(N-1)/2$ inter-city distances in decreasing order and remove the *top* f_s fraction (f_s again denotes the extent of simplification).

However, care must be taken so that the removal of paths between cities does not lead to the formation of two or more (isolated) clusters. Hence, we first find the Minimum Spanning Tree (MST) of the graph of cities (with edge weights equal to pairwise distances) and ensure the edges of the MST are included in the simplified graph. Only after considering the MST, do we include as many edges as allowed by the value of f_s. Thus, if the edge between cities u and v are removed, then $J_{(v,j)(u,i)} = J_{(u,j)(v,i)} = 0$ even for $i = j-1$ and $j+1$.

5.3 Binary Quadratic Programming

The Binary Quadratic Programming (BQP) is another NP-hard problem that seeks to optimize a quadratic objective function with linear constraints in place. It is a special case of the Mixed Integer Quadratic Programming (MIQP) with the solution space being binary. The BQP finds use in portfolio optimization and has an objective that can be expressed as follows:

$$minimize \;\; ||Ax-b||_2 \;\; \text{subject to} \;\; Cx = d, \tag{20}$$

where x is the n-dimensional binary variable, A and C are matrices of dimension $m_1 \times n$ and $m_2 \times n$, respectively, and b and d are vectors with m_1 and m_2 components, with m_2 being the number of constraints. The Ising Hamiltonian of this problem with n spins can be written in the form of a Lagrange function:

$$H = ||Ax-b||_2^2 + \lambda||Cx-d||_2^2, \tag{21}$$

with λ being the Lagrange multiplier. The interaction matrix J and bias h of the Ising model can be obtained as

$$J_{ij} = -\frac{\partial H}{\partial x_i \partial x_j} = -\sum_{k=1}^{m_1} A_{ki}A_{kj} - \lambda \sum_{l=1}^{m_2} C_{li}C_{lj}, \tag{22}$$

$$h_i = \text{Constant terms in } -\frac{\partial H}{\partial x_i} = \sum_{k=1}^{m_1} A_{ki}b_k + \lambda \sum_{l=1}^{m_2} C_{li}d_l. \tag{23}$$

Both J and h are made up of terms from the objective function as well as the constraint (unlike TSP where non-zero interactions were dependent on either problem parameters or tour constraints). For performing graph simplification, it is best to remove connections having a small magnitude of J_{ij} maintaining at least one non-zero element in each row/column of J as earlier. This ensures a sparser graph at the cost of doing away with only the weak interactions.

6 SIMULATION SETUP AND RESULTS

6.1 Methodology

Figure 12 depicts the entire flow for simulation and evaluation. First, the nature and parameters of the NP-hard problem are input to the Graph Simplifier, along with f_s, which is the fraction of the

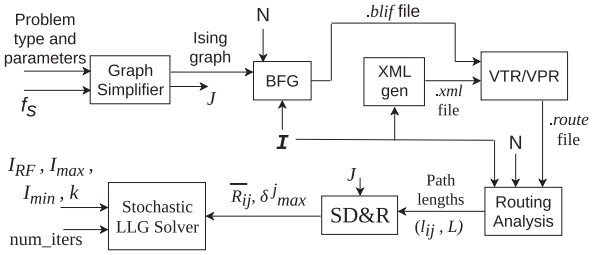


Fig. 12. Steps performed in the simulations. We start with the Graph Simplification and end with the Stochastic LLG simulations.

non-zero interactions that are to be removed (Section 5). It outputs the effective interaction matrix J and also the Ising graph. The latter is input to the BLIF File Generator (BFG) that generates the $.blif$ file according to the number of variables (N) and the number of inputs per Ising cell (I) (Section 4.2). 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 (see Section 4.3). 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, and so on.

The LLG simulations of the MTJ were performed using an HSPICE model[3] [22, 23] that was imported into MATLAB for scalability. The current I_{RF} for Random Flipping (Section 3.2) was chosen in a way that it corresponds to roughly 1% switching probability at the beginning of the simulations (at the first 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." For flipping

- $AP \rightarrow P$, $I_{RF} = 14\ \mu$A (first iteration) down to 13 μA (last iteration)
- $P \rightarrow AP$, $I_{RF} = -23.5\ \mu$A down to $-22\ \mu$A.

Simplification of the Ising graph, with the specified extent f_s, was done as described in Sections 5.1, 5.2, and 5.3. With regard to accounting for the effects of the hardware, simulations were performed for three 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 $R_G = 3.45$ kΩ, $R_{min} = 50$ kΩ, but not recovering from the issue.
- Recovery (Rec) - The modifications made in the design to recover the original signals (i.e., using $\overline{R}_{ij}$ and δ^j_{max}) with $J_{max} = 10$. We restrict δ^j_{max} to a maximum of 1 for considerations of power dissipation and any other issue that this approach might introduce.

We also consider the effect of device variations or process variations in the MTJ on the quality of solution obtained through the Ising simulations. We assume variations in the resistances R_P and R_{AP} of the MTJs. We consider a Gaussian distribution for the variation having a mean 0 and standard deviation 10% of the original R_P and R_{AP}.

Power Estimation methodology: We provide an estimate of the power dissipated in the Ising-FPGA for each problem type and parameters. We first obtain the average/expected value from

[3]Device parameters are listed in Table 1, simulation time step $\delta_t = 0.01$ ns.

Table 3. Descriptions of Graphs on Which Simulations Were Performed

Name	Source	Vertices	Best Cut	Weight Type & Range
$G1$	G1 from G-set [44]	800	11,429	Binary ($\{0, 1\}$)
$G2$	Custom	140	2598.65	Fraction: $U \sim [0, 1]$
$G3$	w01_100.0 from Biq mac [45]	100	645	Integer in $[-10, 10]$
$G4$	ising2.5-300_5555 from [45]	300	8.569×10^6	Integer in $[-2, 2] \times 10^5$

simulation in Synopsys HSPICE for one Ising variable, and then multiply that by the number of variables for getting the power for the entire design. The design in HSPICE consisted of connected Ising cells, each with its op-amps, the MTJ and its control unit, and the variable resistors for interconnections. Also included is an op-amp (not shown if Figure 4) that generates the negative voltage supply $-V_m$ from V_m. It adds 4 μW to the power consumption of each last level Ising cell.

The major source of power consumption is the write stage (see Figure 3), since it has current flowing between the cells through the connections and the write control path. For problems with a large number of Level A cells, the total power for a single Ising variable was obtained by extrapolating the values corresponding to a few Level A cells. This is justifiable, since it was observed that the power increases linearly with the number of Level A cells. That is, if P_1, P_2, and P_3 denote the power consumption with 1, 2, and 3 Level A cells (assuming all connected to full capacity), then $P_2 - P_1 \approx P_3 - P_2$. Specific details are mentioned alongside the respective problems.

The power dissipation at the MTJs during the Random Flip stage was also considered by taking the average of the I^2R of $P \rightarrow AP$ and $AP \rightarrow P$ switching. It turned out to be 2.55 μW per last level cell (or Ising variable).

6.2 Results

Let us now present the outcome 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 number of Ising cells in the Ising-FPGA,
(2) the minimum Channel Width Factor (CWF), which is the minimum number of tracks per channel for successful routing, and
(3) the average length of the paths from the last level cells to the Level A cells (average of all l_{ij} - Figure 10) at this CWF.

1) Max Cut: Table 3 specifies the graphs that were used for benchmarking along with their number of vertices, the best cut value (obtained using an SDP solver [43]) and the type & range/distribution of edge weights. Table 4 lists the aforementioned Ising-FPGA parameters at the specified Ising cell fan-in (I) and for select values of f_s. Recall, that $f_s = 0$ implies no graph simplification; and larger values of f_s indicate a sparser graph (Section 5.1). Also included is the estimate of the average power consumption (in mW) of the system for $f_s = 0$ obtained through HSPICE. Since graph G1 is binary, all connection resistances were programmed to represent $|J| = 1$. Whereas for the other graphs, an average value of $|J| = 0.5$ was used.

It is evident from Table 4 that increasing the value of f_s reduces the number of Ising cells in the design. This is expected, since a more simplified graph has lesser number of connections between Ising spins, thereby reducing the number of Level A Ising cells. A larger fan-in (I) also has the same effect. The minimum CWF and the Average Path Lengths vary in different ways depending on the nature of the graph. The total power depends largely on the connectivity of the graphs,

Table 4. Ising-FPGA Hardware Usage for Max Cut Problems

Name	f_s / I	No. of cells				Min. CWF				Avg Path Lengths				Power (in mW)
		0	0.25	0.5	0.75	0	0.25	0.5	0.75	0	0.25	0.5	0.75	0
$G1$	32	2,398	1,916	858	800	138	152	110	64	23.2	23.3	20.3	16.0	55.57
G2	16	1,400	1,118	817	515	48	48	48	44	8.3	7.7	7.9	10.1	13.95
	32	840	668	530	330	48	52	54	62	8.3	8.1	9.3	9.7	15.43
$G3$	8	216	146	106	102	26	22	20	20	10.6	8.5	5.5	3.5	1.42
$G4$	8	1,044	898	544	300	20	20	20	16	5.8	4.8	4.3	4.7	6.625

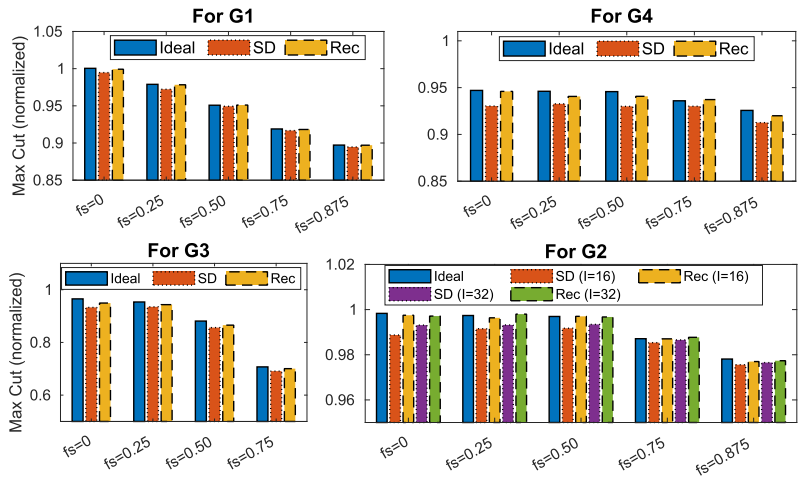


Fig. 13. Max cut values (normalized) from the Ising simulations for the four graphs with different values of f_s.

other than the number of nodes of course. For example, G2 has higher power than G4 in spite of being smaller, because it is fully connected, whereas G4 is much sparser.

Figure 13 shows the obtained cut values for the 4 graphs, each normalized by their respective best cut values in Table 3. Each of the graphs was run 10 times, with 1000 iterations of the Ising simulations per run; all maxcut values are thus average of 10 runs. As we can see, the Ideal maxcut values obtained by simulating the Ising model at $f_s = 0$ 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.

From the data pertaining to Figure 13, Signal Degradation (SD) leads to an average relative drop of 1.03% in the MaxCut values. If we define the extent of recovery in the maxcut values as $(Rec - SD)/(Ideal - SD)$, then the average recovery across graphs was 82.5%. Further, the relative drop in maxcut due to graph simplification varies across graphs and increases with f_s as expected. With $f_s = 0.5$, the maxcut reduced by only 3.57% on an average as compared to $f_s = 0$, and the Ising cell count drops significantly by 48.3%.

Figure 14 shows how the Ising Hamiltonian evolves with time for graph G2. The ragged nature of the plot is due to the annealing through random flips. Last, Table 5 highlights the effect of process variations (PV) in the MTJ device on the maxcut values for all five graphs. We consider the Signal Degradation with recovery (Rec) scenario and assume no graph simplification ($f_s = 0$).

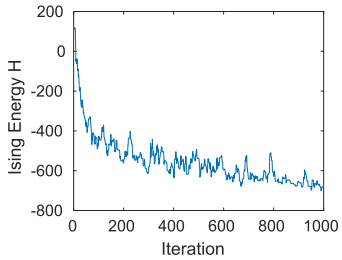


Fig. 14. Ising Hamiltonian H for graph G2 at $f_s = 0$.

Table 5. Maxcut Values with and without Process Variations

Graph Name	$G1$	$G2(I = 16)$	$G2(I = 32)$	$G3$	$G4$
Without PV	11420	2592.1	2591.2	612.4	8.106
With PV	11264	2567.8	2571.0	597.1	7.969

Table 6. Hardware Usage on the Ising-FPGA for TSP. Power is in mW.

	No. of cells				Min. CWF				Avg Path Lengths				Power
Name \ f_s	0	0.25	0.5	0.75	0	0.25	0.5	0.75	0	0.25	0.5	0.75	0
P01	1,125	990	930	870	48				8.72	8.20	8.63	9.13	14.49
GR17	1,445	1,411	1,309	1,156	48				8.59	9.17	9.35	10.05	20.12
FRI26	5,408	4,628	4,134	3,666	60	60	62	62	14.3	12.5	14.7	13.3	67.62

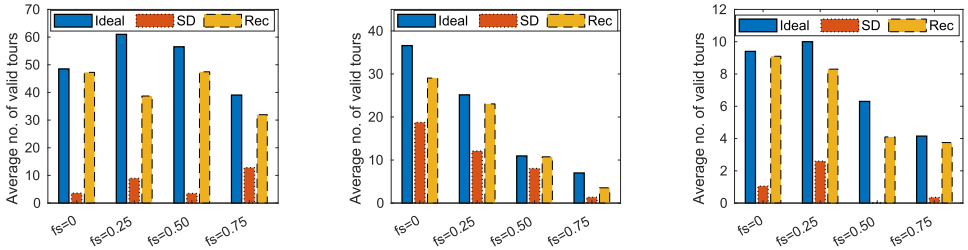
Table 7. Results of Ising Simulations for TSP

City set		$f_s = 0$		$f_s = 0.25$		$f_s = 0.5$		$f_s = 0.75$	
		Valid	MTL	Valid	MTL	Valid	MTL	Valid	MTL
P01	Ideal	20	443.1	19	411	19	391	19	351
	SD	9	450.0	10	424	9	398	15	358
	Rec	20	453.4	20	416	20	389	20	339
GR17	Ideal	19	3,448	19	3,414	15	3002	9	2,896
	SD	12	3,765	12	3,444	14	3481	7	2,977
	Rec	19	3,689	19	3,396	17	3163	9	2,825
FRI26	Ideal	16	2,262	18	2,034	16	1798	8	1,510
	SD	6	2,416	6	2,125	0	—	1	1,572
	Rec	19	2,290	18	2,056	7	1755	4	1,503

"Valid" denotes the number of runs (of 20) in which at least 1 valid tour was found, and MTL denotes the average of the Minimum Tour Lengths obtained in those runs.

On an average, the maxcut reduces by only about 1.45%, indicating that even 10% variations does not have a significant effect.

2) TSP: Three example problems were considered from a dataset [46, 47] (P01, GR17, and FRI26), sets of 15, 17, and 26 cities with optimal tour lengths of 291, 2,085, and 937, respectively. Table 6 lists the hardware usage on the Ising-FPGA for four different values of f_s and with $I = 16$. Ising simulations were run 20 times (each having 2,000 iterations) for each city set and each value of f_s. Table 7 mentions the results in terms of the number of runs (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 4.82% as compared to Ideal, but, more importantly, it reduces the chances of finding a valid tour. With our recovery strategy, the number of valid tours is almost as many as those in the Ideal case and the MTL is only 0.95% higher than Ideal on an average. It



(a) For P01 **(b)** For GR17 **(c)** For FRI26

Fig. 15. Average (over 20 runs) number of valid tours found in a run.

Table 8. Effect of Process Variations on the Results for TSP (Average Number of Valid Tours and MTL)

	P01		GR17		FRI26	
	Valid tours	MTL	Valid tours	MTL	Valid tours	MTL
Without PV	46.25	453.4	29.05	3,689	9.10	2,290
With PV	66.40	448.9	64.05	3,775	14.4	2,206

Table 9. Ising-FPGA Cell Count, Average Level A Wirelengths, and Minimum CWF for BQP

	No. of cells			Min. CWF			Avg Path Lengths			Power (in mW)
Name \ f_s	0	0.25	0.5	0	0.25	0.5	0	0.25	0.5	0
QP1	2,800	2,155	1514	68	62	58	12.1	8.3	8.4	27.28
QP2	11,200	8,754	5661	132	130	112	13.8	11.2	10.8	109.03

must also be noted that simplifying the graph decreases the MTL, because cities located far away have less chances of being travelled consecutively. However, it has the downside of reducing the number of valid tours obtained.

Additionally, Figure 15 compares the average number of valid tours found in each run for the cases Ideal, SD, and Rec. Due to SD, this value dropped by an average of 75.1% compared to the Ideal, again indicating reduced chances of finding a valid tour. We could recover an average of 76.0% of this drop. Simplifying the Ising graph also decreases the average number of valid tours, because each city now tends to be prompted for a visit by a lesser number of cities, and therefore the chances that all cities are visited go down.

In Table 8, we show how PV changes the average number of valid tours (from Figure 15) and the MTL (from Table 7) obtained from the simulations. Once again we take $f_s = 0$ and consider the Rec situation. While the number of valid tours goes up for all 3 city-sets, the MTL reduces slightly by 0.78% on an average. Thus, PV would not have any significant impact on the solution quality.

3) Binary Quadratic Programming: Two custom problems were generated, QP1 and QP2 with $n = 200$ and $n = 400$ spins, respectively. Both had three constraints, that is, $m_2 = 3$, and m_1 was chosen to be 10. Smaller-sized problems with same values of m_1 and m_2 were provided to the ILP-based MIQP solving technique in MATLAB. However, the scalability in terms of time complexity was poor. With $n = 30, 35$, and 40, the runtimes for finding the optimal value of the objective function were 7, 50, and 980s, respectively, indicating that larger problems with n in the range of a few hundreds would take several hours if not days. Table 9 lists the Ising-FPGA resource utilization

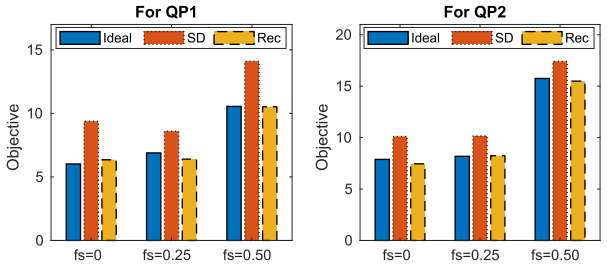


Fig. 16. Best objective value obtained (averaged over 10 runs) for different levels of graph simplification.

Table 10. QP Objective Values in the Presence and Absence of Process Variations

	QP1	QP2
Without PV	6.36	7.45
With PV	6.15	7.33

for both QP1 and QP2 with Ising cell fan-in $I = 16$. The Ising cell count reduces as much as the value of f_s, whereas the CWF and average path length see lesser drops.

The Ising model simulation was run 10 times with 2,000 iterations in each run. Figure 16 provides the average of the minimum of the objective function ($||Ax - b||_2$) achieved (when constraint $Cx = d$ was satisfied). In the presence of SD, this value increases by about 38.1% and 21.0% as compared to the ideal for QP1 and QP2, respectively. After signal recovery, it was restored to be close to the ideal value. The effects of graph simplification in terms of increment in objective value is negligible with $f_s = 0.25$. However, further removal of graph edges ($f_s = 0.5$) leads to an increase of about 65.7% for QP1 and 107% for QP2, the benefit being that the number of Ising cells is almost halved and the CWF also reduces noticeably. Last, with 10% PV, the objective values (as listed in Table 10) decrease by a small margin of 2.45%.

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 20 ps (from Cadence Virtuoso simulations). With three stages (OA1 and 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 0.06 ns. 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 [13, 36] that offer multiple levels of resistance and easy re-programmability with low energy. Such an approach combining memristive and spintronic technology should be possible. Experiments [48] have shown RRAM and MTJ devices placed side by side to be functioning as desired. Our work has the MTJ write control circuit and an op-amp in between the two, and therefore they all must be functioning correctly.
- **Crossbar for Ising model:** A good connectivity structure for an Ising processor, other than the FPGA-like architecture proposed by us, is the crossbar architecture, in which a grid/matrix of RRAMs can represent connections between Ising cells (as suggested in Reference [10]). While the crossbar is indeed an area-efficient way to realize Ising-like connectivity, we highlight two differences from our Ising-FPGA.
  (1) The size of crossbars required for solving moderate/big problems would be large. That would require it to be split into several smaller crossbars and cascading them. For

example, the 26-city TSP would require a 676×676 crossbar, which is not practically feasible.

(2) A sub-optimality in the crossbar architecture would be the lack of its ability to exploit sparsity present in the Ising connectivity matrix J. For example, in the G1 graph in maxcut, each vertex is connected to only 6% of the remaining vertices on an average. Similarly, in an N-city TSP, each Ising spin unit connects only with $4(N-1)$ units rather than $O(N^2)$ units. Further, our proposed graph simplification techniques only decreases the density of connections in the Ising graph. Such sparsity in graph connectivity may remain underutilized in a crossbar architecture. However, the Ising-FPGA design uses hardware (Ising cells) incrementally and utilizes most of the connections physically present.

- 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. Only a few works have put any emphasis on the characterization of system area/power/performance. Shim et al. [18] report on the energy consumption of only the heavy metal layer (attached to the MTJ) during write and read operations as 0.27 and 0.04 pJ, respectively. They state that the energy consumption of the peripheral circuits is small as compared to the spintronic part, perhaps, because each Ising cell is connected to only four others. And Matsumoto et al. [10] report a power consumption of about 10 mW for solving maxcut problems with a graph size of 800 nodes, which is indeed efficient.
- In Reference [31], the authors 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) that exhibits von-Neumann computing. 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 in Reference [31]. 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 thorough comparison.
- Process variations in MTJs does not affect the Ising system to any significant extent, again because such variations add to the randomness in the system that it anyway requires. One difference between the effect of signal degradation and process variations in MTJ may be worth highlighting. SD can change the sign of β (in Equation (7)) and V_o (in Equation (11)), causing current to flow through the MTJ in a direction that is opposite to that desired. Whereas MTJ device variations can only change the magnitude of the current, and not the direction, thereby preventing undesired state changes. Hence, SD has a worse effect on Ising system's convergence than PV.
- Much like commercial FPGAs, the Ising-FPGA can also be heterogeneous [38], consisting of some Ising cells that can be used as last level cells and others that can be used in levels A, B, and so on. The latter type would not contain the MTJ and its write control unit, the OA2, Read unit and Random Flip unit. This will significantly reduce the area of the design, although the placement process will become less flexible.

8 CONCLUSION

In this article, 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 that 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. Last, we describe, for three NP-hard problems, approaches that can simplify their Ising graphs for reducing the required area footprint on the Ising-FPGA and analyse the loss in solution quality.

REFERENCES

[1] Ankit Mondal and Ankur Srivastava. 2020. Spintronics-based reconfigurable ising model architecture. In *Proceedings of the 21st International Symposium on Quality Electronic Design (ISQED'20)*. 134–140.

[2] Frank Neumann and Carsten Witt. 2010. Combinatorial optimization and computational complexity. In *Bioinspired Computation in Combinatorial Optimization*. Springer, 9–19.

[3] Brian Sutton et al. 2017. Intrinsic optimization using stochastic nanomagnets. *Sci. Rep.* 7 (2017), 44370.

[4] Kevin Corder et al. 2018. Solving vertex cover via ising model on a neuromorphic processor. In *Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS'18)*.

[5] Hidenori Gyoten et al. 2018. Enhancing the solution quality of hardware ising-model solver via parallel tempering. In *Proceedings of the 2018 IEEE/ACM International Conference On Computer Aided Design (ICCAD'18)*. IEEE, 1–8.

[6] Barry A. Cipra. 1987. An introduction to the ising model. *The American Mathematical Monthly* 94, 10 (1987), 937–959.

[7] Carleton James Coffrin et al. 2017. *Ising Processing Units: Potential and Challenges for Discrete Optimization*. Technical Report. Los Alamos National Laboratory, Los Alamos, NM.

[8] Md Zahangir Alom et al. 2017. Quadratic unconstrained binary optimization (QUBO) on neuromorphic computing system. In *Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN'17)*. IEEE, 3922–3929.

[9] Masanao Yamaoka et al. 2016. A 20k-spin ising chip to solve combinatorial optimization problems with CMOS annealing. *IEEE J. Solid-State Circ.* 51, 1 (2016).

[10] Shogo Matsumoto et al. 2018. RRAM/CMOS-hybrid architecture of annealing processor for fully connected ising model. In *Proceedings of the 2018 IEEE International Memory Workshop*. IEEE, 1–4.

[11] Vaibhaw Kumar et al. 2018. Quantum annealing for combinatorial clustering. *Quant. Inf. Proces.* 17, 2 (2018), 39.

[12] Sanjukta Bhanja et al. 2016. Non-boolean computing with nanomagnets for computer vision applications. *Nat. Nanotechnol.* 11, 2 (2016), 177.

[13] Mahdi Nazm Bojnordi and Engin Ipek. 2016. Memristive boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning. In *Proceedings of the 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA'16)*. IEEE, 1–13.

[14] D-Wave Systems. (2018). Retrieved from http://www.dwavesys.com.html.

[15] Ryan Hamerly et al. 2018. Scaling advantages of all-to-all connectivity in physical annealers: The Coherent Ising Machine vs. D-Wave 2000Q. *arXiv preprint* arXiv:1805.05217v1 (2018).

[16] Chase Cook et al. 2018. GPU based parallel ising computing for combinatorial optimization problems in VLSI physical design. *arXiv preprint* arXiv:1807.10750 (2018).

[17] Hidenori Gyoten et al. 2018. Area efficient annealing processor for ising model without random number generator. *IEICE Trans. Inf. Syst.* 101, 2 (2018).

[18] Yong Shim et al. 2017. Ising computation based combinatorial optimization using spin-Hall effect (SHE) induced stochastic magnetization reversal. *J. Appl. Phys.* 121, 19 (2017), 193902.

[19] Saima Sharmin et al. 2017. Magnetoelectric oxide based stochastic spin device towards solving combinatorial optimization problems. *Sci. Rep.* 7, 1 (2017), 11276.

[20] Luc Thomas et al. 2017. Basic principles, challenges and opportunities of stt-mram for embedded memory applications. In *Proceedings of the International Conference on Massive Storage Systems and Technology (MSST'17)*.

[21] Andrew Lucas. 2014. Ising formulations of many NP problems. *Front. Phys.* 2 (2014), 5.

[22] Samiran Ganguly et al. 2016. Evaluating spintronic devices using the modular approach. *IEEE J. Explor. Solid-State Comput. Dev. Circ.* 2 (2016).

[23] The Stochastic LLG module. (2016). Retrieved from https://nanohub.org/groups/spintronics/llg_thermal_noise.

[24] Zhitao Diao et al. 2005. Spin transfer switching and spin polarization in magnetic tunnel junctions with MgO and AlOx barriers. *Appl. Phys. Lett.* 87, 23 (2005).

[25] Z. Li and S. Zhang. 2003. Magnetization dynamics with a spin-transfer torque. *Phys. Rev. B* 68, 2 (2003), 024404.

[26] Yue Zhang, Bonan Yan, Wang Kang, Yuanqing Cheng, Jacques-Olivier Klein, Youguang Zhang, Yiran Chen, and Weisheng Zhao. 2015. Compact model of subvolume MTJ and its design application at nanoscale technology nodes. *IEEE Trans. Electr. Dev.* 62, 6 (2015), 2048–2055.

[27] J. Z. Sun, S. L. Brown, W. Chen, E. A. Delenia, M. C. Gaidis, J. Harms, G. Hu, Xin Jiang, R. Kilaru, W. Kula, et al. 2013. Spin-torque switching efficiency in CoFeB-MgO based tunnel junctions. *Phys. Rev. B* 88, 10 (2013), 104426.
[28] Iman Alibeigi, Abdolah Amirany, Ramin Rajaei, Mahmoud Tabandeh, and Saeed Bagheri Shouraki. 2019. A low-cost highly reliable spintronic true random number generator circuit for secure cryptography. In *Proceedings of the SPIN*. World Scientific, 2050003.
[29] Yuanzhuo Qu, Jie Han, Bruce F. Cockburn, Witold Pedrycz, Yue Zhang, and Weisheng Zhao. 2017. A true random number generator based on parallel STT-MTJs. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE'17)*. IEEE, 606–609.
[30] Fonseca Guerra et al. 2017. Using stochastic spiking neural networks on spinnaker to solve constraint satisfaction problems. *Front. Neurosci.* 11 (2017), 714.
[31] Ahmed Zeeshan Pervaiz et al. 2018. Weighted p-bits for FPGA implementation of probabilistic circuits. *IEEE Trans. Neur. Netw. Learn. Syst.* 30, 6 (2018).
[32] Dóra Babicz and Attila Tihanyi. 2019. Simulation of an analogue circuit solving NP-hard optimization problems. In *Proceedings of the 2019 IEEE International Symposium on Circuits and Systems (ISCAS'19)*. IEEE, 1–5.
[33] Xunzhao Yin et al. 2017. Efficient analog circuits for boolean satisfiability. *IEEE Trans. VLSI Syst.* 26, 1 (2017), 155–167.
[34] Tianshi Wang and Jaijeet Roychowdhury. 2019. OIM: Oscillator-based ising machines for solving combinatorial optimisation problems. In *Proceedings of the International Conference on Unconventional Computation and Natural Computation*. Springer, 232–256.
[35] Nicolas Locatelli et al. 2015. Spintronic devices as key elements for energy-efficient neuroinspired architectures. In *Proceedings of the 2015 DATE Conference & Exhibition*. EDA Consortium.
[36] Mohammed A. Zidan et al. 2018. The future of electronics based on memristive systems. *Nat. Electr.* 1, 1 (2018), 22.
[37] Hochul Lee et al. 2015. Design of a fast and low-power sense amplifier and writing circuit for high-speed MRAM. *IEEE Trans. Magnet.* 51, 5 (2015), 1–7.
[38] Husain Parvez et al. 2010. *Application-specific Mesh-based Heterogeneous FPGA Architectures*. Springer Science & Business Media.
[39] Jason Luu et al. 2014. VTR 7.0: Next generation architecture and CAD system for FPGAs. *ACM Trans. Reconfig. Technol. Syst.* 7, 2 (2014), 6.
[40] Verilog-to-Routing (VTR) documentation. (2016). Retrieved from https://docs.verilogtorouting.org/en/latest/.
[41] 1992. The BLIF format. (1992). Retrieved from http://www.cse.iitb.ac.in/ supratik/courses/cs226/spr16/blif.pdf.
[42] Guy Lemieux et al. 2004. Directional and single-driver wires in FPGA interconnect. In *Proceedings of the 2004 IEEE International Conference on Field-Programmable Technology*. 41–48.
[43] 2002. Computational Optimization Laboratory. (2002). Retrieved from https://web.stanford.edu/ yyye/yyye/coplsdp.zip.
[44] 2003. The G-set benchmark. (2003). Retrieved from https://web.stanford.edu/ yyye/yyye/Gset/.
[45] 2007. The Biq Mac Library. (2007). Retrieved from http://biqmac.uni-klu.ac.at/biqmaclib.html.
[46] 2018. Data for the Traveling Salesperson Problem. (2018). Retrieved from https://people.sc.fsu.edu/ jburkardt/datasets/tsp/tsp.html.
[47] Gerhard Reinelt. 1991. TSPLIB: A traveling salesman problem library. *ORSA J. Compu.* 3, 4 (1991), 376–384.
[48] Yu Zhang et al. 2018. Demonstration of multi-state memory device combining resistive and magnetic switching behaviors. *IEEE Electr. Device Lett.* 39, 5 (2018), 684–687.

Received February 2020; revised May 2020; accepted July 2020