MU-MIMO Detection Using Oscillator Ising Machines

Shreesha Sreedhara
UC Berkeley, CA, USA.
shreesha@berkeley.edu

Jaijeet Roychowdhury
UC Berkeley, CA, USA.
jr@berkeley.edu

Joachim Wabnig
Nokia Bell Labs, Cambridge, UK.
joachim.wabnig@nokia-bell-labs.com

Pavan Koteshwar Srinath
Nokia Bell Labs, Paris, France.
pavan.koteshwar@nokia-bell-labs.com

2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD) | 979-8-3503-2225-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCAD57390.2023.10323680

***Abstract*—Over the last several years, Oscillator Ising Machines (OIMs) have been shown to heuristically solve NP-hard combinatorial optimization (CO) problems, most notably MAX-CUT. In this paper, we show that OIMs are capable of solving Multi-User Multiple-Input-Multiple-Output (MU-MIMO) detection, an important real-world problem in telecommunications, achieving near-optimal Symbol Error Rates (SERs). Our results are obtained using CPU- and GPU-based simulation; the latter features a parallelizable event-based algorithm for the generalized Kuramoto equations that reduces OIM simulation times by about $6\times$ without losing accuracy. We also find that good SER results are obtained if 6 or more bits are used to quantize the Ising problem's coupling weights. We provide runtime, throughput and energy consumption comparisons of different implementations and algorithms for MU-MIMO detection, including an OIM emulator chip we had reported earlier. Our results provide useful guidance for designing analog OIM ICs tailored for MU-MIMO detection.**

I. Introduction

Combinatorial optimization (CO) is an enabling technology in many fields that impact modern life, including communication networks, drug/vaccine design, healthcare, delivery logistics, smart grids, *etc.*. However, practical problem sizes have kept outpacing available computational power by large margins. As a result, there has long been interest in ways to speed up CO.

Many practically-important CO problems are computationally difficult (*e.g.*, NP-complete [1]). Such problems can be recast [2] in a standard mathematical form, the Ising model [3–5]. The model is simply a weighted graph, *i.e.*, a collection of nodes/vertices and branches/edges between pairs of nodes, with each branch having a real-number weight. Each node (termed a "spin" in this context) is allowed to take one of two values, either +1 or -1. Associated with this graph is a cost function, called the Ising Hamiltonian, obtained by multiplying the weight of each branch by the values of the two spins it connects to, and summing over all branches. The "Ising problem" is to find spin configurations with the minimum possible Hamiltonian value. For many practical problems, finding a spin configuration with a Hamiltonian close to the minimum possible is also useful.

Note that the Ising problem is NP-hard/complete [6, 7]. Many long-standing problems such as protein folding in biology, finding the optimal artificial neural network for a given set of training data, optimal strategies for playing the game Go, the discrete maximum-likelihood (M.L.) problem, and all 21 of Karp's list of NP-complete problems [1] have Ising incarnations [2].

Over the last decade, a class of hardware Ising solvers (known as Ising machines) has emerged as a promising means to accelerate solutions to these classically difficult computational problems. The premise of Ising machines is that specialized hardware implementing the Ising computational model can solve many classes of NP-complete problems faster than classical algorithms (such as semidefinite programming [8] and simulated annealing [9, 10]) run on digital computers. Ising machines first came into prominence with the D-Wave quantum annealer and the Coherent Ising Machine (CIM). A D-Wave quantum annealer [11] with 2000 spins has been available commercially for several years, with a 5000-spin version recently announced. CIM [12, 13] with 2000 spins has been successfully demonstrated at NTT Research Labs [14], with larger systems under active development. Though without question *tours-de-force* of technology and science that have established the field of Ising machines and inspired follow-on technologies, D-Wave quantum annealers and CIM are not ideally suited for all applications, being physically large, expensive, and difficult to miniaturize or scale to larger problems. For example, the CIM/DOPO scheme involves pulsed lasers and frequency doubling crystals, and is about the size of a rack for a size-2000 machine [14]; D-Wave machines require an operating temperature under 80mK, are the size of several large racks, and are said to cost in the range of $15M.

In 2016, Wang and Roychowdhury discovered that networks of oscillators represented by Generalized-Kuramoto (or Gen-K) models can solve Ising problems (heuristically) [15]. In this scheme, each of the N binary variables (spins) of the Ising problem is encoded by the phase (relative time delay) of an oscillator; the weighted edges of the Ising model are embedded as couplings between the oscillators. They showed that such systems naturally minimize a Lyapunov function that closely approximates the Ising Hamiltonian. This ability stems from the collective behaviour of the oscillators involving two types of *injection locking* (Fundamental and Sub-Harmonic Injection Locking—FHIL and SHIL), a generic synchronization-inducing property exhibited by the oscillators.

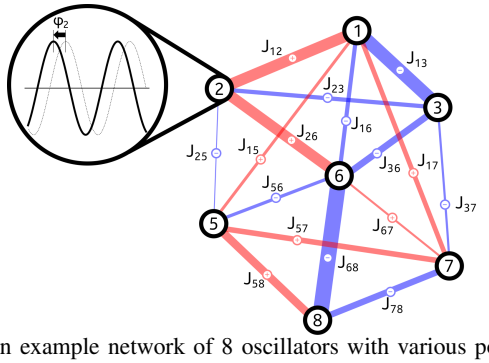

Fig. 1: An example network of 8 oscillators with various positive or negative coupling strengths J_{ij}. Each oscillator can be characterized by its phase relative to a reference oscillator (*e.g.*, oscillator 8).

From a practical deployment perspective, OIM has compelling advantages over previous Ising machines. It can be implemented using plain electronics (in particular, standard CMOS in non-cutting-edge technologies) in very small form factors, especially

compared to CIM and quantum annealers. Indeed, prototype electronic hardware implementations of OIMs have been built that find good solutions of Ising problems in milliseconds [16, 17]. OIMs are orders of magnitude cheaper than prior Ising machines—this, together with small size and easy mass production, greatly broadens the potential applicability of Ising machines.

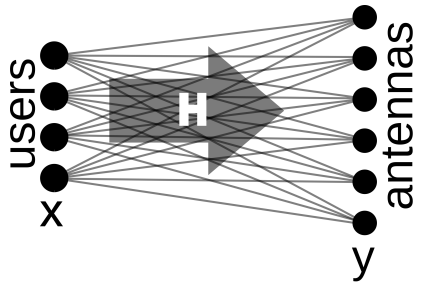

Fig. 2: An illustration of a Multiple User Multiple In Multiple Out (MU-MIMO) setup. Multiple users transmit their data, $\vec{x}$, to a receiver with multiple antennae, where the signal $\vec{y}$ is measured. Transmission occurs over several paths, characterized by the channel matrix H.

In this paper, we report the performance of OIMs on an important problem in wireless communications, the MU-MIMO (Multi-User Multiple-Input-Multiple-Output) detection problem. As shown in Figure 2, modern wireless communication settings involve multiple users with single/multiple transmit antennae, using the same resources (time and frequency) to transmit to a receiver equipped with multiple receive antennae. As a result, each received signal consists of a noisy superposition of several users' transmitted symbols. MU-MIMO 'detection' is the problem of recovering the originally-sent symbols given the set of signals received. Solving exactly for the most likely transmitted symbols, *i.e.*, the M.L. (Maximum Likelihood) solution involves solving a hard CO problem [18, 19]. This is too computationally expensive to be practical; hence, heuristic methods that use much less computation, such as LMMSE ("linear minimum mean-square error"), are universally used even though they do not recover transmitted symbols as accurately as M.L.. Here, we report OIM simulation and hardware performance results on the MU-MIMO detection problem, and show that OIMs have the potential to achieve near-optimal symbol error rates.

The rest of the paper is organized as follows. We first provide background information regarding the Ising problem, FHIL and SHIL, OIMs, and the MU-MIMO problem (Sec. II). In Sec. III, we present CPU and GPU simulation methods of OIMs for MU-MIMO detection. Note that the coupling weights in an OIM need to be quantized for a hardware implementation; we explore the impact of finite-resolution couplings on the performance of OIMs in this section as well.[1]

This is followed by an event-based algorithm that can further reduce simulation times (Sec. IV). Conventionally, an ODE system is solved at incremental time steps starting from a given initial condition [21]. We present a parallel, event-based Gen-K ODE solver which exploits properties intrinsic to the Gen-K model to reduce the number of required time steps. Given the solution to the ODE at a time step, we estimate a time interval during which the derivatives of all the variables are constant.[2] We then directly jump to the end of the interval, *i.e.*, jump to the next *event*, and analytically integrate the variables exploiting the fact that the derivatives were constant throughout the interval.

Moving onward from CPU and GPU simulations, the next step is hardware implementation of OIMs for MU-MIMO detection. A prime contributor to analog OIM performance degradation is the variability of components. In [22], this was addressed by emulating OIMs in digital hardware; we summarize it in Sec. V. Note that OIM emulation is accomplished by directly solving the underlying ODE (*i.e.*, the Gen-K ODE) in hardware. Moreover, fixed point operations are exploited to make the implementation more efficient.

We consolidate the results from each of the above sections in Sec. VI. Here, we show that OIMs achieve near-optimal Symbol Error Rates (SERs) as opposed to other heuristics such as LMMSE whose performance is up to 20× worse than M.L.. We also show that the event-based Gen-K ODE solver reduces simulation times by up to 6× while maintaining near-optimal SERs. With regards to real world implementations, we find that 6-7 or more bits of coupling resolution are sufficient to guarantee negligible degradation in OIM performance; this makes analog OIMs feasible. Finally, we summarize the performance of the digital OIM emulator IC. The preliminary CPU and GPU simulations along with the emulator IC set up a baseline for future analog OIM implementations; an analog OIM must match the detection accuracy of the emulator IC while being more efficient with power in order to be competitive.

II. Background

A. The Ising Problem

The cost function of the Ising problem, *i.e.*, the Ising Hamiltonian is defined as

$$C(s_1,\cdots,s_n) \triangleq -\frac{1}{2}\sum_{i,j=1}^{n} J_{ij}\,s_i s_j, \tag{1}$$

where $s_i \in \{-1,\ +1\}$, $i = 1,\cdots,n$, are the n spins, J_{ij} are the weights; J_{ij} obey $J_{ij} = J_{ji}$ and $J_{ii} = 0$. An alternative version of the Ising Hamiltonian uses so-called "external magnetic field" terms comprised of a linear combination of the spins, *i.e.*,

$$\tilde{C}(s_1,\cdots,s_n) \triangleq -\left[\frac{1}{2}\sum_{i,j=1}^{n} J_{ij}\,s_i s_j + \sum_{i=1}^{n} B_i s_i\right]. \tag{2}$$

By adding one more spin, $s_{n+1} \equiv 1$ and defining

$$J_{n+1,i} = J_{i,n+1} \triangleq B_i, \quad i = 1,\cdots,n, \quad \text{and } J_{n+1,n+1} \triangleq 0, \tag{3}$$

it is easily shown that (2) is equivalent to (1), *i.e.*,

$$\tilde{C}(s_1,\cdots,s_n) \equiv C(s_1,\cdots,s_n,s_{n+1}=1). \tag{4}$$

Thus the form (1), which we use here, is general enough to capture external magnetic field terms.

B. FHIL, SHIL, and Oscillator Ising Machines

OIMs (Oscillator Ising Machine) are networks of mutually coupled, self-sustaining, nonlinear ([23]) of oscillators. They employ phenomena known as Fundamental and Sub-Harmonic Injection Locking—FHIL and SHIL.

If an oscillator is disturbed by an external input with a frequency ω_D close to ω_0 (as illustrated by the waveform at the top left of Figure 3) it will spontaneously change its natural frequency to exactly match that of the external input. Moreover, the external input and the oscillator's output waveform become synchronized ("phase locked") to each other, as illustrated at the right of Figure 3. This phenomenon, which has a long and rich history dating back to at least 1672 [24], is known today as injection locking, or more precisely, as fundamental-harmonic

[1] A preprint of our CPU-based simulation results was put out earlier [20].

[2] This is possible only for specific choices of the parameters of the Gen-K model.

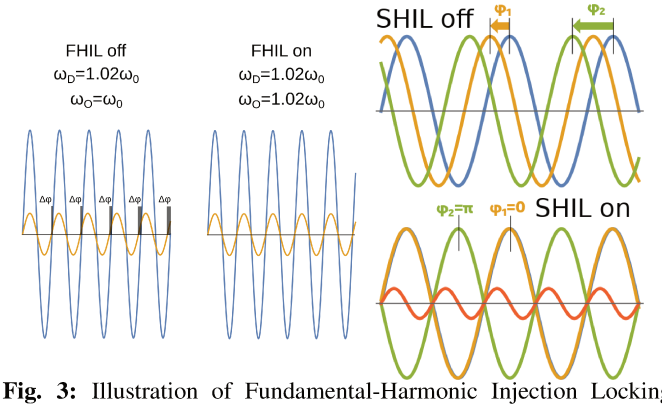

Fig. 3: Illustration of Fundamental-Harmonic Injection Locking (FHIL) and Sub-Harmonic Injection Locking (SHIL). The figure on the left shows the interaction of a self-sustaining nonlinear oscillator (of free-running frequency ω_0) with a driving signal of a slightly higher frequency ($\omega_D = 1.02\omega_0$). It leads to a shift in frequency of the oscillator ($\omega_O = \omega_D$) and locking of the oscillator's and the signal's phases. Figure on the right is an illustration of 2$^{\text{nd}}$-subharmonic injection locking (2-SHIL). Without the SYNC signal, interacting self-sustaining oscillators settle in a fixed phase relationship according to their coupling. When a 2-SHIL signal of sufficient amplitude is introduced, the phases lock at either π or 0.

injection locking (FHIL).

It can be shown [25] that if FHIL occurs, the phase difference between the injection and the oscillation waveforms will be a single fixed number, *i.e.*, there cannot be two or more different phases at which the waveforms lock stably.

FHIL is only one possible type of injection locking; interesting synchronization behaviours also manifest when the injected signal's frequency is near an integral multiple of the oscillator's natural frequency ω_0. For example, if the injection frequency is close to twice the natural frequency, *i.e.*, $\omega_D \simeq 2\omega_0$, frequency- and phase-locking can also occur; this is called 2-SHIL (2$^{\text{nd}}$ sub-harmonic injection locking). In 2-SHIL, the oscillator changes its natural frequency to precisely *half* of ω_D; the resulting waveform is also phase locked to the injection signal, as illustrated in Figure 3. A key difference between FHIL and 2-SHIL is that in the latter, there are *two* possible values of relative phase between the injection and oscillation waveforms at which (stable) lock can occur [25]; moreover, these two phase locks are always separated by 180°. In OIMs, the two 180°-separated phase locks in 2-SHIL correspond to Ising spins $+1$ and -1. FHIL and 2-SHIL are both crucial for making networked oscillator systems function as Ising machines.

In an OIM system (such as Figure 1), the oscillators are coupled to each other (for example, using resistors) with weights obtained from the given Ising problem. If the frequencies of the oscillators are close enough to each other, FHIL will force all oscillators to lock to a common frequency [26]. An additional common external signal, of fixed frequency set to about twice that of the average natural frequency of the oscillators, is also injected into each oscillator. This injection, termed SYNC, is used to induce 2-SHIL, *i.e.*, phase lock at one of two binary values separated by 180°.

A useful mathematical model for the coupled oscillator system with SYNC injection is the Generalized Kuramoto (Gen-K) equation [27],

$$\frac{1}{\omega_0}\frac{d\theta_i}{dt} = -K_s \,\mathrm{F_s}\left(2\theta_i(t)\right) - \sum_{j=1}^{N} J_{ij}\,\mathrm{F_c}\left(\theta_i(t) - \theta_j(t)\right), \quad (5)$$

shown for the simplified case where all oscillators have the same natural (angular) frequency ω_0. N is the number of oscillators in the system; $\theta_i(t)$ is the phase of the i^{th} oscillator; $\mathrm{F_c}(\cdot)$ is a 2π-periodic function that captures the FHIL dynamics of the system; similarly, $\mathrm{F_s}(\cdot)$ is a function that captures 2-SHIL dynamics (K_s represents the amplitude of the SYNC signal); and J_{ij} is the coupling strength between the i^{th} and j^{th} oscillator, the same as in (1).

If $\mathrm{F_c}(.)$ is odd and K_s is kept constant with time, it can be shown that the phases in (5) always evolve to naturally minimize the Lyapunov function

$$L(\vec{\theta}) \triangleq \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{N} J_{ik}\,\mathrm{I_c}(\theta_i - \theta_k) + \frac{K_s}{2}\sum_{i=1}^{N}\mathrm{I_s}(2\theta_i) \quad (6)$$

where $\mathrm{I_s}(\cdot)$ and $\mathrm{I_c}(\cdot)$ are integrals of $\mathrm{F_s}(\cdot)$ and $\mathrm{F_c}(\cdot)$, respectively [27]. Crucially, it can be shown that when K_s is high, the Lyapunov function approximates the Ising Hamiltonian. Note that we vary K_s periodically to (heuristically) obtain lower minima of the Lyapunov function. The coupled oscillator system, with periodic variation of K_s, evolves to find good solutions of the Ising problem.

C. Reformulating the MU-MIMO detection problem in Ising form

A succinct development of the relation between the MU-MIMO and Ising problems follows (a more detailed exposition can be found in [19]). Given a BPSK MU-MIMO system with N_t transmitters (users) and N_r receivers, define a vector of transmitted symbols to be

$$\vec{x} = [x_1, \cdots, x_{N_t}]^T, \quad (7)$$

where $x_i \in \{\pm 1\}$ are N_t simultaneously transmitted symbols. Define $H \in \mathbf{R}^{N_r \times N_t}$ to be the channel transmission matrix, and $\vec{y} \in \mathbf{R}^{N_r}$ to be the vector of received signals. The vector of received signals can be modeled as

$$\vec{y} = H\vec{x} + \vec{w}, \quad (8)$$

where $\vec{w}$ represents additive white Gaussian noise (AWGN).

The optimal solution of the MU-MIMO detection problem, *i.e.*, the Maximum Likelihood (M.L.) solution, is the transmitted symbol vector $\vec{x}^*$ that minimizes the error from the ideally-received signal, *i.e.*

$$\vec{x}^* = \underset{\vec{x}\in\{\pm 1\}^{N_t}}{\arg\min} \|\vec{y} - H\vec{x}\|^2. \quad (9)$$

To frame the MU-MIMO detection problem in Ising form, we first augment the number of transmitted symbols by one to define the spin vector

$$\vec{s} \triangleq \left[\underbrace{x_1}_{s_1}, \cdots, \underbrace{x_{N_t}}_{s_{N_t}}, \underbrace{1}_{s_{N_t+1}}\right]^T = \begin{bmatrix}\vec{x}\\ 1\end{bmatrix}, \quad (10)$$

where we use the terminology $s_i \equiv x_i$, $i = 1, \cdots, N_t$ to emphasize that the transmitted symbols serve as spins for the Ising version of the problem. Note that the last spin of $\vec{s}$ (*i.e.*, s_{N_t+1}) is fixed at 1. Next, define $\hat{H} \triangleq [H, -\vec{y}] \in \mathbf{R}^{N_t \times (N_t+1)}$ and set $J = -\hat{H}^T\hat{H} \in \mathbf{R}^{(N_t+1)\times(N_t+1)}$. With the above definitions, it is easy to see that the ground state of the Ising Hamiltonian (1) is the same as the M.L. solution from (9). Note that if a solution of the Ising problem has $s_{N_t+1} = -1$, it is easily converted to a solution with $s_{N_t+1} = +1$ by flipping all the spins (this does not change the value of the Hamiltonian).

This concludes the section on prerequisite materials.

III. Simulation of OIMs

As stated in the introduction (Sec. I), we evaluate OIMs on MU-MIMO problems (Sec. II). The test problems consist of 11 sets; a set corresponds to a specific SNR (signal-to-noise ratio) at the receiving antennae. Each set contains 1000 different

channel matrices H, and for each channel there exist 50 pairs of transmitted and received signal vectors ($\vec{x}$ and $\vec{y}$ respectively). Thus, there are a total of $11 \times 1000 \times 50 = 550000$ problems in the database. Note that $N_t = 16$ and $N_r = 64$, and QPSK (Quadrature Phase Shift Keying, [28]) modulation was used. Since there are 2 bits per symbol in QPSK, this translates to a size-33 Ising problem (Sec. II). Note that QPSK encoding needs to be taken into account in the calculation of symbol error rates (SER), *i.e.*, any change to a symbol (a single or double bit error) should be counted as a single "symbol error".

The channel matrices H (generated as described in [29, Section IV-A]) capture correlations between users in a fading environment more realistically than the commonly-used independent, identically distributed (i.i.d.) Rayleigh fading model in the literature—in essence, [29] takes into account the fact that users closer to one another tend to have more-correlated channels than further-away users.

We simulate OIMs by numerically solving (5) using a method known as Forward Euler (FE) [21]. Given an initial value $\vec{\theta}(0)$ and a time step parameter h, $\vec{\theta}((n+1)h)$ for $n \in \{0,1,\ldots\}$ are calculated successively [21] using

$$\frac{\mathrm{d}\theta_i(nh)}{\mathrm{d}t} \approx \frac{\theta_i((n+1)h) - \theta_i(nh)}{h} = \\ -\sum_{j=1}^{N} \mathrm{J}_{i,j} \cdot \mathrm{F_c}(\theta_i(nh) - \theta_j(nh)) - K_s \mathrm{F_s}(\theta_i(nh)) \,, \tag{11}$$

where $i \in \{1,\ldots,N\}$. FE is an *explicit* method, *i.e.*, $\vec{\theta}((n+1)h)$ can be calculated from $\vec{\theta}(nh)$ in one single iteration as shown above [21]. Note that the total number of time steps is given by t_{stop}/h, where $[0, t_{stop}]$ denotes the time interval of the simulation. Example $\mathrm{F_s}(.)$ and $\mathrm{F_c}(.)$ are shown in Figure 4 and Figure 5. The results of the above simulations are plotted in Figure 8. As shown, OIMs achieve near-optimal Symbol Error Rates (SERs) over the entire dataset. Other heuristics such as LMMSE and Zero-Forcing Equalization (ZF) [30] fail to achieve this accuracy, and are up to 20× worse than the M.L. solution.

As noted in the introduction, the coupling weights need to be quantized for hardware implementations of OIMs. To this end, we repeat the above simulations for many different coupling resolutions. The results are summarized in Figure 10. As shown, OIMs achieve near-optimal SERs when the coupling resolution is greater than approximately 8-9 bits.

Finally, we note that the OIM simulations can be easily parallelized. The i^{th} phase of the next time step (*i.e.*, $\theta_i((n+1)h)$ in (11)) can be calculated independently of the other time steps. Thus, (11) can be evaluated in N separate threads (which are to be synchronized once per FE time step). The run-times of various simulation methods are listed in Table I.

Next, we explore a technique to shorten the simulation times.

IV. The Event-Based Algorithm

In this section, we provide an event-based algorithm to efficiently solve the Gen-K ODE (5).

We first choose $\mathrm{F_c}(\theta) \triangleq K_c \tanh\big(G\sin(2\pi\theta)\big)$ and $\mathrm{F_s}(\theta) \triangleq \tanh\big(G\sin(2\pi\cdot 2\theta)\big)$, where G is a positive real number. If G is 'large', the above functions reduce to[3]

$$\mathrm{F_c}(\theta) = \begin{cases} +1 & \theta \bmod 1 < 0.5, \quad (12a) \\ -1 & \theta \bmod 1 \geq 0.5, \quad (12b) \end{cases}$$

and

$$\mathrm{F_s}(\theta) = \begin{cases} +1 & \theta \bmod 0.5 < 0.25, \quad (13a) \\ -1 & \theta \bmod 0.5 \geq 0.25. \quad (13b) \end{cases}$$

Observe that the $\mathrm{F_c}(.)$ and the $\mathrm{F_s}(.)$ functions are *constants* except at the crossover points. We exploit this property to find time intervals during which $\mathrm{F_c}(.)$ and $\mathrm{F_s}(.)$ remain constant.

For instance, consider the case when $\forall t \geq 0$, $\mathrm{d}\theta_1(t)/\mathrm{d}t \triangleq -K_s \mathrm{F_s}(\theta_1)$. Let the initial value be $\theta_1(0) \triangleq 0.65$ (dashed vertical line at $t = 0$ in Figure 4). Applying (13a), we get $\mathrm{d}\theta_1(t)/\mathrm{d}t = -K_s$ at $t = 0$. Moreover, it remains constant at $-K_s$ until θ_1 reaches a crossover point of $\mathrm{F_s}(\theta_1)$, which is equal to 0.5 in this case. Thus, the solution to the above ODE is $\theta_1(t) = \theta_1(0) - K_s t$ for $t \in [0, \delta_{1,1}]$, where $\delta_{1,1} \triangleq 0.15/K_s$ (this can be easily checked by solving the above example ODE). In other words, the derivative of $\theta_1(t)$ is constant until t reaches $\delta_{1,1}$.

The above observation forms the core of the event-based algorithm. As opposed to FE where 'small' time steps are taken in each iteration, we can directly jump to the next *event* of the ODE system. We define an event as follows. If an $\mathrm{F_c}(.)$ (or an $\mathrm{F_s}(.)$) function remains constant for the next δ seconds and changes its value immediately after, then an event is said to occur in the $\mathrm{F_c}(.)$ (or the $\mathrm{F_s}(.)$) function after a delay of δ seconds. Concretely, we estimate the 'δ' of an event as follows.

The delay before an event occurs in the $\mathrm{F_s}(.)$ function of the i^{th} spin (denoted as $\delta_{i,i}$) is defined as

$$\delta_{i,i} \triangleq \begin{cases} \dfrac{\theta_i \bmod 0.25}{|\mathrm{d}\theta_i/\mathrm{d}t|} & \mathrm{d}\theta_i/\mathrm{d}t < 0, \quad (14a) \\ \dfrac{0.25 - \theta_i \bmod 0.25}{|\mathrm{d}\theta_i/\mathrm{d}t|} & \mathrm{d}\theta_i/\mathrm{d}t > 0, \quad (14b) \\ \infty & \mathrm{d}\theta_i/\mathrm{d}t = 0. \quad (14c) \end{cases}$$

Next, consider the example shown in Figure 5; this is a Gen-K system with two negatively coupled oscillators. Let $(\theta_i(0) - \theta_j(0)) = 0.1$ (dashed vertical line at $t = 0$ in Figure 5), and let $\mathrm{d}(\theta_i(t) - \theta_j(t))/\mathrm{d}t$ at $t = 0$ be positive. Note that $\delta_{1,2}$ is the amount of time it takes for $(\theta_1(t) - \theta_2(t))$ to reach 0.5 (a crossover point of $\mathrm{F_c}(.)$) starting from the initial condition 0.1. Thus, $\mathrm{F_c}\big(\theta_1(t) - \theta_2(t)\big)$ remains constant until $\delta_{1,2}$. (15) merely extends this idea to the general case.

$$\delta_{i,j} \triangleq \begin{cases} \dfrac{(\theta_i - \theta_j) \bmod 0.5}{|\mathrm{d}(\theta_i - \theta_j)/\mathrm{d}t|} & \mathrm{d}(\theta_i - \theta_j)/\mathrm{d}t < 0, \quad (15a) \\ \dfrac{0.5 - (\theta_i - \theta_j) \bmod 0.5}{|\mathrm{d}(\theta_i - \theta_j)/\mathrm{d}t|} & \mathrm{d}(\theta_i - \theta_j)/\mathrm{d}t > 0, \quad (15b) \\ \infty & \mathrm{d}(\theta_i - \theta_j)/\mathrm{d}t = 0. \quad (15c) \end{cases}$$

In the above equation, $\delta_{i,j}$ is the time delay before an event occurs in the coupler connected between the i^{th} and the j^{th} spins. Note that we ignore the case when i and j are the same since $\mathrm{J}_{i,i} = 0$ for all i.

We present the event-based algorithm in two parts. Sec. IV-A contains a simple event-based algorithm. In Sec. IV-B, we analyze a major drawback of this algorithm where two phases 'lock' to each other and cause tightly spaced events. We address this issue by 'merging' the locked phases. In Sec. IV-C, we provide the final event-based algorithm.

[3]Except at the crossover points such as $\theta = 0, 0.5$, *etc.*

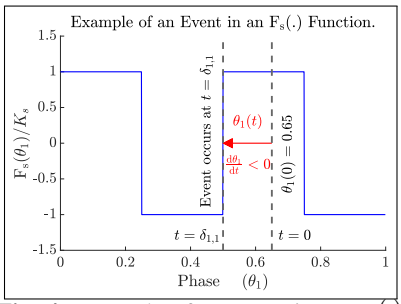

Fig. 4: Example of an event in an $F_s(.)$ function.

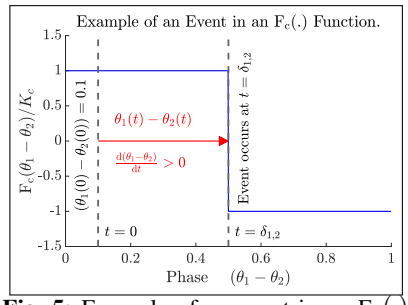

Fig. 5: Example of an event in an $F_c(.)$ function.

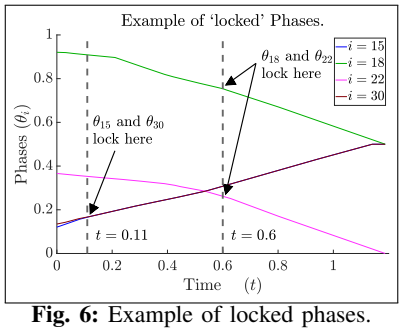

Fig. 6: Example of locked phases.

A. Part-1 of the Event-Based Algorithm

Step-1 (1-10 in Alg. 1): Perform the following initialization steps.

- Set $\mathscr{H}_{min} \leftarrow \infty$ ($\mathscr{H}_{min}$ keeps track of the lowest $\mathscr{H}$ found).
- Create a coupling matrix $\tilde{\mathbf{J}}$, and set $\tilde{\mathrm{J}}_{i,j} = \mathrm{J}_{i,j}$ for $i, j \in \{1,\ldots,N\}$. Note that we assume $\mathrm{J}_{i,i} = 0$ for $i \in \{1,\ldots,N\}$.
- Create $\mathscr{A}$—a set of phases that are still 'active', and $\mathscr{A}^{\mathsf{C}}$—the complement of $\mathscr{A}$. Initially, $\mathscr{A} = \{1,\ldots,N\}$. Note that $\mathscr{A}$ will not be modified until we perform 'merge' operations.
- Initialize every phase in $\mathscr{A} \setminus \{N\}$ to a random number in $[0,1]$; the N^{th} phase is set to 0.[4] Note that the N^{th} phase will be held constant at 0 throughout the simulation.
- Initialize **L**, **B**, and 'mergedTo' (explained in Step-5 and Step-6).
- Set t_{sim}, the time variable in the simulator to zero. Set K_s (the constant used in $\mathrm{F_s}(.)$ functions) to zero. This is carried out to allow parameter cycling of K_s. Specifically, we first solve the ODE with $K_s = 0$, then set K_s to a nonzero value to binarize the phases.

Step-2 (12-13 in Alg. 1): Evaluate the derivatives. Calculate $\mathrm{d}\theta_i(t_{sim})/\mathrm{d}t$ using (5) for $i \in \mathscr{A} \setminus \{N\}$. Note that we use (12) and (13) to evaluate $\mathrm{F_c}(.)$ and $\mathrm{F_s}(.)$ (respectively).

Step-3 (14-23 in Alg. 1): Find the next event. Calculate $\delta_{i,j}$ (and $\delta_{i,i}$, if $K_s \neq 0$) for $i \in \mathscr{A} \setminus \{N\}$ and $j \in \mathscr{A} \setminus \{i\}$. Find the minimum over the above $\delta_{i,j}$ (and $\delta_{i,i}$, if $K_s \neq 0$) and denote it as δ_{min}.

Let (i_m, j_m) be an ordered pair such that $\delta_{i_m,j_m} = \delta_{min}$. In other words, (i_m, j_m) are the phases whose coupling function's event is the nearest in the future. If there are multiple such (i_m, j_m), choose any. We call (i_m, j_m) as the bottleneck causing ordered pair, and regard i_m as the phase that caused the bottleneck. [5]

Step-4 (24-33 in Alg. 1): Integrate the phases if δ_{min} is finite.

(a) δ_{min} is infinite and $K_s = 0$. This means that no $\mathrm{F_c}(.)$ function changes value in the future. We set K_s to a nonzero value[6] and restart from Step-2.

(b) δ_{min} is infinite and $K_s \neq 0$. Here too, no $\mathrm{F_c}(.)$ or $\mathrm{F_s}(.)$ changes value in the future. We merely exit and go to Step-9.

(c) δ_{min} is finite. By construction of δ_{min}, the derivatives of all the phases in $\mathscr{A} \setminus \{N\}$ remain constant until t_{sim} increases by δ_{min}. Therefore, we analytically integrate the phases in $\mathscr{A} \setminus \{N\}$. First, increment δ_{min} by a small number to step over the discontinuous $\mathrm{F_c}(.)$ and $\mathrm{F_s}(.)$ functions, *i.e.*, $\delta_{min} \leftarrow \delta_{min} + t_{small}$. Then, integrate the phases according to $\theta_i \leftarrow \theta_i + \delta_{min} \cdot \mathrm{d}\theta_i(t_{sim})/\mathrm{d}t$, where $i \in \mathscr{A} \setminus \{N\}$. Update t_{sim} to reflect the progress.

At this stage of the simulator, we could repeat the above procedure starting from Step-2 until all the phases settle to steady state values[7] or until t_{sim} reaches a predetermined value. However, such a strategy has a major drawback, which is analyzed next.

B. The issue of 'locked' phases

Let C_i denote the total number of iterations during the simulation where the i^{th} phase caused the bottleneck (see Step-3 for terminology). Figure 6 shows the top four phases of an example Gen-K model whose C_i counts are the highest. Consider the following two pairs of phases $(\theta_{15}, \theta_{30})$ and $(\theta_{18}, \theta_{22})$ plotted in this figure. We observe the following.

- The two phases in a pair have approximately the same phase difference starting from some time point. For example, $(\theta_{15} - \theta_{30}) \approx 0$ for $t \geq 0.11$, and $(\theta_{18} - \theta_{22}) \approx 0.5$ for $t \geq 0.6$.
- The derivatives of the two phases in a pair are also approximately the same starting from some time point.

We consider such phases to be *locked* to each other. To summarize, two variables in the Gen-K ODE system may lock to each other in phase and evolve with approximately the same derivative starting from some time point. We have observed (in simulations) that such locked phases cause δ_{min} to be unnecessarily small, slowing down the event-based simulator. We fix this issue by *merging* the two offending phases. Essentially, we regard the locked phases as a single phase, and solve a modified ODE system that has one less phase variable. This is explained in the next subsection.

C. Part-2 of the Event-Based Algorithm

Recall that in Steps 1-4, we initialized variables, calculated the derivatives, estimated the time delay before the next event occurs, jumped to the earliest event and integrated the phases.

Step-5 (34-43 in Alg. 1): Detect locked phases.

Let the locked phases in Figure 6 be θ_a and θ_b (for example, $a = 15$ and $b = 30$). Observe that θ_a and θ_b lock such that the phase difference is $0.5k$ for some $k \in \mathbb{Z}$. We do not have a proof that this always occurs in the Gen-K model; however, we will assume that it holds true. In other words, if θ_a and θ_b lock at t^*, we assume that $\forall t \geq t^*$, $(\theta_a(t) - \theta_b(t)) = 0.5 \cdot k_{a,b}$

[4] $\mathscr{A} \setminus \mathscr{B}$ denotes the set obtained by subtracting set $\mathscr{B}$ from set $\mathscr{A}$.

[5] Note that if δ_{min} was due to the $\mathrm{F_s}(.)$ function of the i^{th} phase, we approximate the ordered pair to be (i_m, N).

[6] Usually we choose $K_c = K_s = 1$, assuming $\max_{i,j} |J_{ij}| = 1$.

[7] Use a suitably chosen criterion for detecting if a phase has settled; this is not discussed for brevity.

for some $k_{a,b} \in \mathbb{Z}$.

The criterion for detecting locked phases is as follows. Given the above assumption, we consider θ_a to be locked to θ_b if $(\theta_a - \theta_b)$ remained 'close' to $0.5k_{a,b}$ (for some $k_{a,b} \in \mathbb{Z}$) for the last r iterations where (a,b) was the bottleneck-causing ordered pair.[8] Note that the above r iterations need not be consecutive. Moreover, if there are less than r iterations where (a,b) was the bottleneck, we consider θ_a and θ_b to not be locked by default.

This criterion is implemented using matrices $\mathbf{L}$ and $\mathbf{B}$. $\mathbf{L}$ stores the 'limiting' phase differences, *i.e.*, $\mathbf{L}$ stores $0.5\,k_{i,j}$ for each possible ordered pair (i,j). $\mathbf{B}$ keeps track of the number of iterations where the locking criterion described above was satisfied. In other words, $\mathrm{B}_{i,j}$ is the number of iterations where (i,j) was the bottleneck and $(\theta_i - \theta_j)$ remained 'close' to $0.5k_{i,j}$. Note that in each iteration, there is only one bottleneck causing ordered pair (i_m, j_m) (by definition). Thus, in each iteration, we need only check if θ_{i_m} has locked to θ_{j_m}.

Step-6 (45-48): Merge θ_{i_m} to θ_{j_m}. We move θ_{i_m} to $\mathscr{A}^{\complement}$, *i.e.*, mark θ_{i_m} as inactive. From now on, θ_{i_m} stores the offset of the i_m^{th} phase *w.r.t.* the j_m^{th} phase. Since $(\theta_{i_m} - \theta_{j_m}) = 0.5k_{i_m,j_m}$, we set $\theta_{i_m} \leftarrow 0.5k_{i_m,j_m}$. In addition, we also keep track of the index of the phase to which i_m has been merged. This is implemented using the array 'mergedTo'. Since i_m is merged to j_m, we set $\text{mergedTo}(i_m) \leftarrow j_m$. Moreover, we update the offsets and the 'mergedTo' entries of the phases which were merged to i_m. Note that $(\theta_p - \theta_{j_m}) = (\theta_p - \theta_{i_m}) - (\theta_{i_m} - \theta_{j_m})$. Thus, we update the offsets as $\theta_p \leftarrow \theta_p + 0.5k_{i_m,j_m}$ where p is a phase which was originally merged to i_m.

Step-7 (49-52): Update $\tilde{\mathbf{J}}$. We update the coefficients of the coupling matrix $\tilde{\mathbf{J}}$ to reflect the merge operation as explained below. An example is shown in Figure 7. Here, θ_2 is being merged to θ_4 (*i.e.*, $i_m = 2$ and $j_m = 4$), and $\mathscr{A}^{\complement} = \{3\}$. Consider the Gen-K equation of the q^{th} phase (where $q \in \mathscr{A}$), repeated below for reference. Note that $i_m \neq j_m$ (by construction), and let $q \in \mathscr{A}$ be distinct from i_m and j_m.

$$\begin{aligned}\frac{\mathrm{d}\theta_q(t)}{\mathrm{d}t} = &-\tilde{\mathrm{J}}_{q,i_m}\cdot \mathrm{F_c}\big(\theta_q(t)-\theta_{i_m}(t)\big)\\ &-\tilde{\mathrm{J}}_{q,j_m}\cdot \mathrm{F_c}\big(\theta_q(t)-\theta_{j_m}(t)\big) - \mathrm{F_s}\big(\theta_q(t)\big)\\ &-\sum_{p\in\mathscr{A}\setminus\{i_m,j_m\}} \tilde{\mathrm{J}}_{q,p}\cdot \mathrm{F_c}\big(\theta_q(t)-\theta_p(t)\big)\,,\end{aligned} \tag{16}$$

where the terms coupling θ_q to θ_{i_m} and θ_{j_m} are explicitly written out. Replacing θ_{i_m} by $(\theta_{j_m} + 0.5\cdot k_{i_m,j_m})$ and applying (12), we see that

$$\mathrm{F_c}\big(\theta_q - \theta_{i_m}\big) = \begin{cases} +\mathrm{F_c}\big(\theta_q - \theta_{j_m}\big) & k_{i_m,j_m} \text{ is even}, \quad (17a)\\ -\mathrm{F_c}\big(\theta_q - \theta_{j_m}\big) & k_{i_m,j_m} \text{ is odd}\,. \quad (17b)\end{cases}$$

In other words, for any value of $\theta_q(t)$, $\mathrm{F_c}\big(\theta_q(t) - \theta_{i_m}(t)\big)$ is the same as $\mathrm{F_c}\big(\theta_q(t) - \theta_{j_m}(t)\big)$ except possibly for the sign. Therefore, the first two terms in the RHS of (16) can be combined.

Thus, we set $(q \in \mathscr{A}\setminus\{i_m, j_m, N\})$

$$\tilde{\mathrm{J}}_{q,j_m} \leftarrow \begin{cases} \tilde{\mathrm{J}}_{q,j_m} + \tilde{\mathrm{J}}_{q,i_m} & \text{if } k_{i_m,j_m} \text{ is even}, \quad (18a)\\ \tilde{\mathrm{J}}_{q,j_m} - \tilde{\mathrm{J}}_{q,i_m} & \text{if } k_{i_m,j_m} \text{ is odd}. \quad (18b)\end{cases}$$

Note that the i_m^{th} phase is marked as inactive; thus, the coupling term between q and i_m will be ignored starting from the next iteration. Examples of coefficient updates carried out according to (18) are highlighted as blue arrows in Figure 7 where $q \in \{1,5\}$.

[8] r is a tunable parameter; for example, $r = 3$.

Next, we edit the coefficients of the j_m^{th} phase (*i.e.*, the j_m^{th} row of $\tilde{\mathbf{J}}$). As stated previously, the phases i_m and j_m evolve with approximately the same derivative when locked. Thus, we set the derivative of the j_m^{th} phase to the *average* of the i_m^{th} and the j_m^{th} derivatives (and ignore the i_m^{th} phase starting from the next iteration).

Concretely, we use

$$\begin{aligned}\frac{\mathrm{d}\theta_{j_m}(t)}{\mathrm{d}t} \triangleq &-\sum_{\substack{p\in\mathscr{A}\\ p\neq i_m\\ p\neq j_m}} \begin{pmatrix}\tilde{\mathrm{J}}_{j_m,p}\cdot \mathrm{F_c}\big(\theta_{j_m}(t)-\theta_p(t)\big)+\\ \tilde{\mathrm{J}}_{i_m,p}\cdot \mathrm{F_c}\big(\theta_{i_m}(t)-\theta_p(t)\big)\end{pmatrix}\Big/2\\ &-\Big(\mathrm{F_s}\big(\theta_{j_m}(t)\big)+\mathrm{F_s}\big(\theta_{i_m}(t)\big)\Big)/2\end{aligned} \tag{19}$$

The above equation can be simplified as follows.

Note that $\mathrm{F_s}(.)$ is 0.5-periodic, which implies $\mathrm{F_s}\big(\theta_{j_m}(t)\big) = \mathrm{F_s}\big(\theta_{i_m}(t)\big)$ since $\theta_{i_m}(t) = (\theta_{j_m}(t) + 0.5\cdot k_{i_m,j_m})$. Thus, the average of $\mathrm{F_s}\big(\theta_{j_m}(t)\big)$ and $\mathrm{F_s}\big(\theta_{i_m}(t)\big)$ in (19) can be simplified to merely $\mathrm{F_s}\big(\theta_{j_m}(t)\big)$. Moreover, the coupling functions satisfy a relation analogous to (17) (omitted for brevity). Therefore, we set $(p \in \mathscr{A}\setminus\{i_m, j_m\})$

$$\tilde{\mathrm{J}}_{j_m,p} \leftarrow \begin{cases} \big(\tilde{\mathrm{J}}_{j_m,p} + \tilde{\mathrm{J}}_{i_m,p}\big)/2 & \text{if } k_{i_m,j_m} \text{ is even}, \quad (20a)\\ \big(\tilde{\mathrm{J}}_{j_m,p} - \tilde{\mathrm{J}}_{i_m,p}\big)/2 & \text{if } k_{i_m,j_m} \text{ is odd}. \quad (20b)\end{cases}$$

Note that the i_m^{th} phase is moved to $\mathscr{A}^{\complement}$; thus, the i_m^{th} row of $\tilde{\mathbf{J}}$ is ignored starting from the next iteration. In Figure 7, the coefficient updates carried out according to (20) are highlighted using red arrows $(p \in \{1,5,6\})$. Note that $\tilde{\mathrm{J}}_{j_m,j_m}$ remains unchanged at 0.

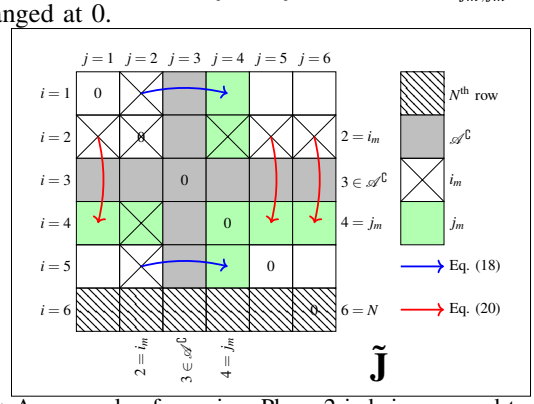

Fig. 7: An example of merging. Phase-2 is being merged to phase-4 (*i.e.*, $i_m = 2$ and $j_m = 4$), and $\mathscr{A}^{\complement} = \{3\}$.

Step-8 (11-52 in Alg. 1): Repeat until $\mathscr{A} = \{N\}$ (*i.e.*, $|\mathscr{A}| = 1$) or until t_{sim} reaches a predetermined value.

Step-9 (53-57 in Alg. 1): If $i \in \mathscr{A}^{\complement}$, θ_i is the offset of the i^{th} phase *w.r.t.* to $\text{mergedTo}(i)$. Thus, we calculate the actual value of the i^{th} phase using $\theta_i \leftarrow \theta_i + \theta_{\text{mergedTo}(i)}$. We then map the phases to spins (+1—even multiple of 0.5, -1—odd multiple of 0.5) and evaluate the Hamiltonian using the original coupling matrix $\mathbf{J}$. Repeat the above procedure by choosing different random initial conditions for $\vec{\theta}$ to find better minima. Note that one can also add noise to the existing $\vec{\theta}$ values instead of generating completely different noise values (not shown in Alg. 1 for brevity).

We also note that the algorithm in Alg. 1 can be parallelized using $N_t - 1$ threads (explanation is omitted for brevity). This concludes the 'event-based' algorithm. We relegate the results of this simulation technique to Sec. VI. Next, we summarize a simulation technique [22] that is of interest particularly to digital hardware implementations of OIMs.

Algorithm 1: The event-based Gen-K ODE solver

// Step-1: Initialization.
$\mathscr{H}_{min} \leftarrow \infty$, $\vec{s}_{opt} \leftarrow \text{rand}(N-1)$; // Heuristic ground state of $\mathscr{H}$.
for $n_{trial} = \{1,2,\ldots,3\}$ **do** // For example, repeat 3 times.
  $\tilde{\mathbf{J}} \leftarrow \mathbf{J}$; // $\tilde{\mathbf{J}}$ is a copy of the connectivity matrix $\mathbf{J}$.
  $\{\theta_1,\ldots,\theta_{N-1}\} \leftarrow \text{rand}(N-1)$, $\{d\theta dt_i,\ldots,d\theta dt_{N-1}\} \leftarrow 0$;
  $\theta_N \leftarrow 0$, $d\theta dt_N \leftarrow 0$; // The N^{th} phase is held constant at 0.
  $\mathscr{A} = \{1,\ldots,N\}$, $\mathscr{A}^{\complement} = \emptyset$, ; // $\mathscr{A}$ is the set of active phases.
  // $\mathbf{L}$ and $\mathbf{B}$ initialized below are used for detecting locked phases.
  $\mathbf{L} \leftarrow \text{rand}(N-1, N)$; // $(N-1)\times N$ matrix of limiting values.
  $\mathbf{B} \leftarrow \mathbf{0}$; // $(N-1)\times N$ matrix of bottleneck counters.
  // mergedTo$(i) = j$ implies θ_i is merged to θ_j.
  mergedTo $\leftarrow \{1,\ldots,(N-1)\}$; // Default is mergedTo$(i) \leftarrow i$.
  $t_{sim} \leftarrow 0$, $K_s \leftarrow 0$; // Other miscellaneous variables.
  while $(|\mathscr{A}| > 1)$ AND $(t_{sim} < t_{stop})$ **do** // Choose suitable t_{stop}.
    // Step-2: Calculate the derivatives.
    for $i \in \mathscr{A} \setminus \{N\}$ **do**
      $d\theta dt_i \leftarrow -\sum_{j\in\mathscr{A}} \tilde{\mathrm{J}}_{i,j} \cdot \mathrm{F_c}(\theta_i - \theta_j) - \mathrm{F_s}(\theta_i)$
    // Step-3: Estimate the next event.
    $\delta_{min} \leftarrow \infty$, $i_m = N$, $j_m = N$;
    for $i \in \mathscr{A} \setminus \{N\}$ **do**
      for $j \in \mathscr{A} \setminus \{i\}$ **do**
        Calculate $\delta_{i,j}$ according to (15) ;
        if $\delta_{i,j} < \delta_{min}$ **then**
          $\delta_{min} \leftarrow \delta_{i,j}$, $i_m \leftarrow i$, $j_m \leftarrow j$
      if $K_s \neq 0$ **then**
        Calculate $\delta_{i,i}$ according to (14) ;
        if $\delta_{i,i} < \delta_{min}$ **then**
          $\delta_{min} \leftarrow \delta_{i,i}$, $i_m \leftarrow i$, $j_m \leftarrow N$
    // Step-4: Integrate the phases if δ_{min} is finite.
    if $\delta_{min} = \infty$ **then**
      if $K_s = 0$ **then**
        $K_s \leftarrow 1$;
        continue // Restart from Step-2.
      else
        break // Nothing much we can do, exit the loop.
    else
      $t_{sim} \leftarrow t_{sim} + \delta_{min} + t_{small}$, $\delta_{min} \leftarrow \delta_{min} + t_{small}$;
      for $i \in \mathscr{A} \setminus \{N\}$ **do**
        $\theta_i \leftarrow \theta_i + d\theta dt_i \cdot \delta_{min}$
    // Step-5: Check for locked phases.
    $\theta_{diff} \leftarrow (\theta_{i_m} - \theta_{j_m})$; // Phase difference.
    $\rho_{diff} \leftarrow \text{round}(\theta_{diff}, 0.5)$; // Round to $0.5k$, $k \in \mathbb{Z}$.
    $\mu \leftarrow$ false ; // To merge or not to merge, this is the question.
    if $|\rho_{diff} - \theta_{diff}| < \varepsilon$ AND $|\mathrm{L}_{i_m,j_m} - \rho_{diff}| < \varepsilon$ **then** // ε is a small number.
      $\mathrm{B}_{i_m,j_m} \leftarrow \mathrm{B}_{i_m,j_m} + 1$;
      if $\mathrm{B}_{i_m,j_m} \geq r$ **then** // For example, $r = 3$.
        $\mu \leftarrow$ true ; // Enable phase merging.
    else
      $\mathrm{L}_{i_m,j_m} \leftarrow \rho_{diff}$; // Update L_{i_m,j_m}.
      $\mathrm{B}_{i_m,j_m} \leftarrow 0$; // Reset B_{i_m,j_m}.
    if μ **then**
      // Step-6: Merge θ_{i_m} to θ_{j_m}.
      // From now on, θ_{i_m} stores the offset of i_m w.r.t. j_m.
      Move i_m to $\mathscr{A}^{\complement}$, $\theta_{i_m} \leftarrow \rho_{diff}$, mergedTo$(i_m) \leftarrow j_m$;
      for $i \in \mathscr{A}^{\complement}$ **do** // Update the inactive phases.
        if mergedTo$(i) = i_m$ **then**
          mergedTo$(i) \leftarrow j_m$, $\theta_i \leftarrow \theta_i + \rho_{diff}$
      // Step-7: Update the coupling matrix $\tilde{\mathbf{J}}$.
      for $q \in \mathscr{A} \setminus \{N\}$ **do** // Update the j_m^{th} column of $\tilde{\mathbf{J}}$
        Update $\tilde{\mathrm{J}}_{q,j_m}$ according to (18).
      for $p \in \mathscr{A} \setminus \{j_m\}$ **do** // Update the j_m^{th} row of $\tilde{\mathbf{J}}$
        Update $\tilde{\mathrm{J}}_{j_m,p}$ according to (20).
    // Step-8: Repeat until $|\mathscr{A} = 1|$ or until $t_{sim} \geq t_{stop}$
  // Step-9: Calculate the (heuristic) ground state of $\mathscr{H}$.
  for $i \in \mathscr{A}^{\complement}$ **do** // Calculate the actual phase values.
    $\theta_i \leftarrow \theta_i + \theta_{\text{mergedTo}(i)}$
  Map $\vec{\theta}$ to $\vec{s}$, calculate $\mathscr{H}$ according to (1) ;
  if $\mathscr{H} < \mathscr{H}_{min}$ **then**
    $\mathscr{H}_{min} \leftarrow \mathscr{H}$, $\vec{s}_{opt} \leftarrow \vec{s}$; // Store the results.

V. Efficiently Solving Gen-K ODE Systems Using Fixed Point Operations

In this section, we outline how fixed point arithmetic can be exploited to accelerate OIM simulations. Note that the following was implemented in our publication [22] on a OIM emulator IC.

Recall that the phases of the oscillators in OIMs can be modeled as Gen-K ODE systems (Sec. II). We assume that $\mathrm{F_c}(.)$ and $\mathrm{F_s}(.)$ are as defined in (12) and (13). The above functions can be evaluated efficiently by applying fixed point formats [31] as described below.

We utilize a fixed point format of the form $0 \bullet b_1 b_2 \ldots b_n$ to store the θ_i in (11). Note that the decimal equivalent of the above form is $\sum_{i=1}^{n} b_i 2^{-i}$, which ranges from 0 (when $\forall i$, $b_i = 0$) to ≈ 1 (when $\forall i$, $b_i = 1$). Hence, the above fixed point format stores $(\theta_i \bmod 1)$ with n bits of precision.

Consider $\mathrm{F_c}(\theta_{diff} \triangleq \theta_i - \theta_j)$. As stated in (12), $\mathrm{F_c}(\theta_{diff})$ depends only upon the modulo-1 value of θ_{diff}. Moreover, $\theta_{diff} \bmod 1$ is simply the fixed point subtraction of the two phases since $\theta_{diff} \bmod 1 = (\theta_i - \theta_j) \bmod 1 = ((\theta_i \bmod 1) - (\theta_j \bmod 1)) \bmod 1$. Thus, $\mathrm{F_c}(\theta_{diff})$ can be efficiently evaluated using the MSB of θ_{diff} since $\mathrm{F_c}(\theta_{diff}) = +1$ if $\text{MSB}(\theta_{diff}) = 0$, and -1 if $\text{MSB}(\theta_{diff}) = 1$.

The above can be extended to $\mathrm{F_s}(.)$ as well. Denoting the penultimate MSB of θ as $b_2(\theta)$, we have $\mathrm{F_s}(\theta) = +1$ if $b_2(\theta) = 0$, and $+1$ if $b_2(\theta) = 1$. We can thus efficiently evaluate the RHS of (11) using a fixed point format.

The above idea was demonstrated in a digital OIM emulator with 33 spins and all-to-all programmable connectivity [22]. More details of this prototype IC can be found in [22]; its results are summarized in Sec. VI.

VI. Results

A. Performance of OIMs on practical MU-MIMO detection problems

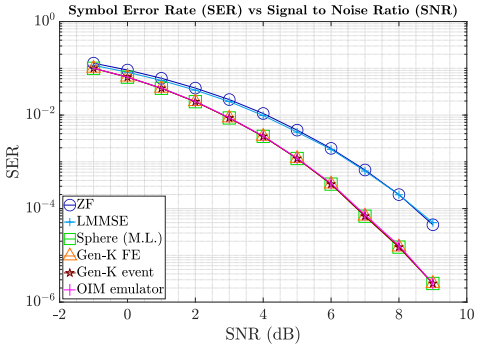

Fig. 8: Detection performance of OIM *vs.* other methods. The symbol error rate (SER) is shown as a function of signal-to-noise ratio (SNR) for M.L., LMMSE, ZF, and OIM detectors.

Figure 8 shows the average SER (over all problems in each SNR set) from ML, LMMSE, ZF, and OIM. Note that M.L. detection was carried out using a Sphere decoder [32]. As stated in Sec. III, SER numbers for OIMs were obtained by numerical simulations of the generalized Kuramoto equations (5).

Examining the data reveals several interesting features:

1) The SER of M.L. for high SNR values is very low; for example, the SER is 2.5e-6 at SNR=9 dB. Thus, high-SNR cases are much more challenging than low-SNR cases.

2) The performance of LMMSE varies from about 16% worse than M.L. at SNR -1 dB, to almost 20× worse for the SNR=9 set.
3) In contrast, SER numbers from OIMs are very close to M.L. for all SNR sets, *i.e.*, not more than 4% over M.L., which is significantly better than LMMSE for every SNR set. Note that OIMs tends to match the SER of M.L. at higher SNRs (which are more challenging).

B. Effect of coupling quantization on OIM performance

Figure 9 shows the absolute values of the 33×33 coupling matrices for an example problem from the SNR=9 dB set. As can be seen, the entries in the last row and column (which stem from the "external magnetic field" terms $-H^T\vec{y}$), are about a factor of 4 larger than the other values in the matrix. Similar patterns are seen in the coupling matrices of all the problems. This suggests that from an accuracy standpoint, it is advantageous to use one set of quantized values for the last row and column, and another set for the remainder of the matrix—this is easy to implement in IC hardware.

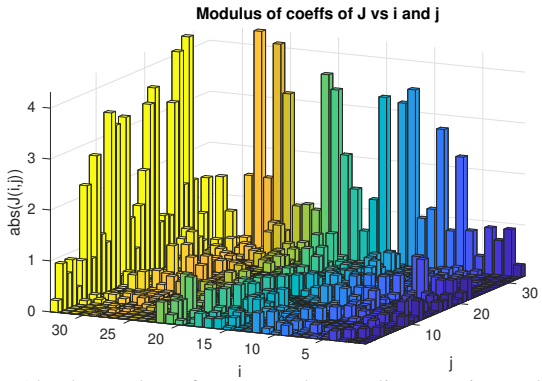

Fig. 9: Absolute value of an example coupling matrix used in the detection (for SNR 9; all the other SNR sets are virtually identical in pattern). The terms in the last row and column are approximately 4× larger than the rest of the terms.

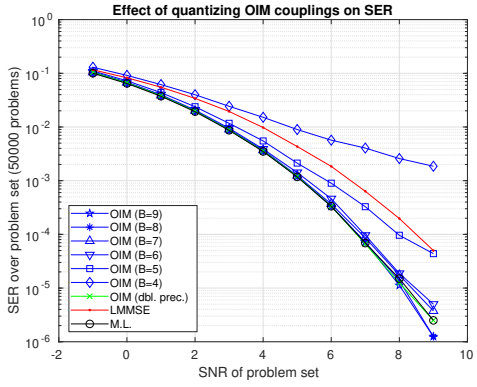

Fig. 10: Symbol error rate as a function of signal-to-noise ratio for different quantizations of the OIM coupling weights, compared to OIM with coupling weights in double precision, LMMSE and M.L. OIM with 9-bit through 6-bit quantizations yields SERs from near-optimal to acceptable. Detection performance deteriorates quickly below 6 bits of quantization.

Figure 10 shows OIMs' performance with quantized couplings; B—the number of bits used for quantization is varied from 9 down to 4. It can be seen that SER performance degradation (over Maximum Likelihood with no quantization) is essentially negligible for 9 and 8 bits of quantization. Using 6 bits of quantization still yields significant improvements over LMMSE across all the problems, while $B = 5$ remains competitive against LMMSE. These results, indicating that implementing OIM in IC hardware is practical, can help guide design tradeoffs.

C. Performance of the event-based simulator, and the digital OIM emulator prototype IC

The SER numbers of the event-based algorithm (Sec. IV) and the digital OIM emulator IC (Sec. V and [22]) are plotted in Figure 8. As shown, both the event-based algorithm and the emulator achieve near-optimal SERs over all the signal to noise ratios.

The time taken by various algorithms and the OIM emulator IC are listed in Table I. As expected, the OIM emulator is the fastest since it is a specialized hardware. Furthermore, the event-based algorithm achieves up to 6× speed up as compared to the 'plain' OIM simulator while maintaining a near-optimal detection accuracy.

Finally, Table I also lists estimated energy spent per detection problem. Note that ZF, LMMMSE, and Sphere were run on a general purpose computer that consumes 100 W; Gen-K FE and Gen-K event-based OIM simulations were run on a general purpose Graphics Processing Unit (GPU) that consumes a total of 200 W when using 4864 threads. As before, the specialized hardware consumes the least amount of energy.

TABLE I: Performances of various MIMO detection schemes.

Algorithm	Runtime	Throughput	Energy
ZF	0.16 ms	200 Kbps	16 mJ
LMMSE	1.7 ms	18 Kbps	170 mJ
Sphere decoder	2.3 ms	14 Kbps	230 mJ
Gen-K FE (N_t threads)	50 ms	0.64 Kbps	66 mJ
Gen-K event-based (N_t threads)	8 ms	4 Kbps	11 mJ
OIM emulator IC	1 ms	32 Kbps	250 µJ

VII. Conclusion

In this paper, we applied OIMs to (heuristically) solve the MU-MIMO detection problem. CPU and GPU simulations of OIMs indicate that they achieve near-optimal accuracy, whereas other detectors such as LMMSE and ZF are up to 20× worse. Note that simulation times of OIMs can be shortened by a novel 'event-based' ODE solver presented in this paper. It can achieve up to 6× speed ups while maintaining the same near-optimal accuracy. The next natural step to CPU and GPU simulations is hardware implementation. A major hurdle in implementing analog OIMs is the variability of components. As demonstrated in [22], this issue can be circumvented by emulating OIMs in digital hardware. Note that fixed point operations can be utilized to design efficient digital OIMs emulators. The preliminary simulations and the digital OIM emulator set up a performance baseline for future analog OIMs for MU-MIMO detection.

References

[1] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, pp. 85–103, Springer, 1972.

[2] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics*, vol. 2, p. 5, 2014.

[3] E. Ising, "Beitrag zur Theorie des Ferromagnetismus," *Zeitschrift für Physik*, vol. 31, no. 1, pp. 253–258, 1925.

[4] S. G. Brush, "History of the Lenz-Ising Model," *Rev. Mod. Phys.*, vol. 39, pp. 883–893, Oct 1967.

[5] S.M. Bhattacharjee and A. Khare, "Fifty Years of the Exact solution of the Two-dimensional Ising Model by Onsager," *arXiv preprint cond-mat/9511003*, 1995.

[6] J. Schneider and S. Kirkpatrick, *Stochastic Optimization*. Springer Science & Business Media, 2007.

[7] F. Barahona, "On the computational complexity of Ising spin glass models," *Journal of Physics A: Mathematical and General*, vol. 15, no. 10, p. 3241, 1982.

[8] L. Vandenberghe and S. Boyd, "Semidefinite Programming," *SIAM Review*, vol. 38, no. 1, pp. 49–95, 1996.

[9] B. Gärtner and J. Matousek, *Approximation Algorithms and Semidefinite Programming*. Springer, 2014.

[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *SCIENCE*, vol. 220, no. 4598, pp. 671–680, 1983.

[11] Z. Bian, F. Chudak, W. G. Macready, and G. Rose, "The Ising model: teaching an old problem new tricks," *D-Wave Systems*, vol. 2, 2010.

[12] Z. Wang, A. Marandi, K. Wen, R. L. Byer and Y. Yamamoto, "Coherent Ising machine based on degenerate optical parametric oscillators," *Physical Review A*, vol. 88, no. 6, p. 063853, 2013.

[13] Y. Haribara, S. Utsunomiya and Y. Yamamoto, "Computational principle and performance evaluation of Coherent Ising Machine based on degenerate optical parametric oscillator network," *Entropy*, vol. 18, no. 4, p. 151, 2016.

[14] T. Inagaki, Y. Haribara, K. Igarashi, T. Sonobe, S. Tamate, T. Honjo, A. Marandi, P. L. McMahon, T. Umeki, K. Enbutsu and others, "A Coherent Ising machine for 2000-node Optimization Problems," *Science*, vol. 354, no. 6312, pp. 603–606, 2016.

[15] T. Wang and J. Roychowdhury, "Oscillator-based Ising Machine," *arXiv:1709.08102*, 2017.

[16] T. Wang, L. Wu and J. Roychowdhury, "New Computational Results and Hardware Prototypes for Oscillator-based Ising Machines," in *Proc. IEEE DAC*, pp. 239:1–239:2, 2019.

[17] T. Wang, L. Wu, P. Nobel, and J. Roychowdhury, "Solving combinatorial optimisation problems using oscillator based Ising machines," *Natural Computing*, pp. 1–20, April 2021.

[18] M. O. Damen, H. El Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Transactions on Information Theory*, vol. 49, no. 10, pp. 2389–2402, 2003.

[19] M. Kim, D. Venturelli, and K. Jamieson, "Leveraging quantum annealing for large MIMO processing in centralized radio access networks," in *Proceedings of the ACM Special Interest Group on Data Communication*, pp. 241–255, ACM, 2019.

[20] J. Roychowdhury, J. Wabnig, and K. P. Srinath, "Performance of Oscillator Ising Machines on Realistic MU-MIMO Decoding Problems," *Research Square preprint (Version 1)*, 22 September 2021. Web link to preprint.

[21] K. E. Atkinson, *An Introduction to Numerical Analysis*. New York: John Wiley & Sons, second ed., 1989.

[22] S. Sreedhara, J. Roychowdhury, J. Wabnig, and P. Srinath, "Digital emulation of oscillator ising machines," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–2, 2023.

[23] A. Demir, A. Mehrotra, and J. Roychowdhury, "Phase Noise in Oscillators: a Unifying Theory and Numerical Methods for Characterization," *IEEE Trans. Ckts. Syst.–I: Regular Papers*, vol. 47, pp. 655–674, May 2000.

[24] C. Huygens, *Horologium Oscillatorium*. Paris, France: Apud F. Muget, 1672. (Observations of injection locking between grandfather clocks).

[25] A. Neogy and J. Roychowdhury, "Analysis and Design of Sub-harmonically Injection Locked Oscillators," in *Proc. IEEE DATE*, Mar 2012.

[26] J. Roychowdhury and P. Bhushan, "Hierarchical abstraction of weakly coupled synchronized oscillator networks," *International Journal for Numerical Methods in Engineering*, 2015.

[27] T. Wang and J. Roychowdhury, "OIM: Oscillator-based Ising Machines for Solving Combinatorial Optimisation Problems," in *arXiv:1903.07163*, 2019.

[28] D. Saha and T. G. Birdsall, "Quadrature-quadrature phase-shift keying," *IEEE Transactions on Communications*, vol. 37, no. 5, pp. 437–448, 1989.

[29] M. Goutay, F. A. Aoudia, and J. Hoydis, "Deep Hypernetwork-based MIMO Detection," in *21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–5, IEEE, 2020.

[30] J. G. Proakis and M. Salehi, *Digital Communications*. Boston: McGraw Hill, fifth ed., 2008.

[31] C. Kormanyos, *Real-Time C++: Efficient Object-Oriented and Template Microcontroller Programming*. Springer, 2018. .

[32] B. Hassibi and H. Vikalo, "On the expected complexity of sphere decoding," in *Conf. Record of Thirty-Fifth Asilomar Conf. on Signals, Systems and Computers*, vol. 2, pp. 1051–1055, IEEE, 2001.