An Efficient Simulated Oscillator-based Ising Machine on FPGAs

Bailiang Liu, Tingting Zhang, Xingjian Gao, and Jie Han

***Abstract*— Ising model-based computing has emerged as an efficient approach for solving combinatorial optimization problems. In particular, oscillator-based Ising machines (OIMs) have shown promising performance in such metrics as solution quality, time-to-solution, and energy-to-solution. Existing designs, however, require customized chips and use sparsely connected topologies with limited precision for the spin coupling coefficients. In this paper, a simulated oscillator-based Ising machine (SOIM) is designed with a fully connected topology and high coefficient precision for an efficient implementation on field-programmable gate arrays (FPGAs). To this end, an FPGA-oriented model is first developed to describe the underlying mechanism of an oscillator-based Ising machine based on revised differential equations. To save hardware, periodic functions that are expensive to implement are replaced by piecewise linear functions. Moreover, Gaussian noise is omitted in the system to further save hardware. An OIM simulation algorithm is then proposed to solve the new differential equations using the Euler integration method. From experiments on solving 800-node max-cut problems, the results reach a level of 99% of the best-known values. SOIMs of different sizes are then developed and synthesized on a Zynq UltraScale+ board. Compared with state-of-the-art FPGA-based Ising machines, the SOIM is expected to utilize fewer hardware resources to efficiently solve complex combinatorial optimization problems by leveraging a high coefficient precision and a fully connected topology.**

I. Introduction

Combinatorial optimization has become the foundation of many modern technologies, such as electronic design automation [1] and pharmaceutical product design [2]. The non-deterministic polynomial time-hard combinatorial optimization problem (COP) is challenging to solve due to the exponential growth of the search space when using an enumerating approach. Ising machines have emerged as an efficient COP solver to find a near-optimal solution within a much shorter time frame. A heuristic algorithm [3], such as simulated annealing (SA) [4] or simulated bifurcation (SB) [5], makes the energy of the Ising model converging to the ground state. An oscillator-based Ising machine (OIM) uses an oscillator network to implement the Ising model. By its analog nature, an OIM often shows promising results in solution quality, time-to-solution, and energy-to-solution measures. Numerous OIM designs, such as the RC-Tank OIM [6], the Schmitt-trigger OIM [7], and the Ring OIM [8], have been proposed over the years. Most implementations are sparsely connected and the precision of coupling coefficients is very limited due to the use of resistive or capacitive coupling. Based on the complementary metal-oxide semiconductor technology, a digitally emulated OIM (DEOIM) supports 33 fully connected spins [9]. However, a significant loss occurs to the solution quality due to the use of approximation for efficient digital implementations.

A field-programmable gate array (FPGA) is an efficient alternative to custom chip design. It offers significant advantages in terms of cost efficiency, scalability, and precision. It also provides the flexibility to alter functionality post production, allowing for on-the-fly adjustments to meet evolving computational demands. This reconfigurability ensures that FPGA-based implementations are easily scaled up or modified for meeting specific requirements without costly redevelopment. Moreover, the ability to reconfigure hardware enables FPGAs to achieve high precision in computation, thus potentially capable of achieving a high solution quality for solving COPs.

In this paper, an efficient simulated OIM design, referred to as an SOIM, is proposed for FPGA implementations. To this end, an FPGA-oriented model is first developed to capture the evolution behavior of oscillator phases in the OIM by using revised differential equations. Specifically, the numerical range is first scaled for ease in hardware implementation. The complex periodic functions are modified to FPGA-friendly piece-wise linear functions to further reduce hardware. Gaussian noise is omitted in the system, based on the observation from experiments that it has little influence on the tested Gset benchmarks. An OIM simulation algorithm is then proposed for the FPGA-oriented model by solving the differential equations using the Euler integration approach. An SOIM with a fully connected topology is developed for the proposed algorithm. The precision of coupling coefficients is set to 5 bits to provide a relatively high accuracy for number representation. The simulation results for solving max-cut problems and synthesis results in FPGAs show that the SOIM achieves a high hardware efficiency with a great potential to solve complex COPs.

The remainder of this paper is organized as follows. Section II introduces the basics of the Ising model and OIMs. The design of the proposed SOIM is presented in Section III. Section IV reports the experimental results on solving max-cut problems and evaluates the hardware performance. Conclusions are given in Section V.

II. Preliminaries

A. The Ising Model and OIMs

The Ising model mathematically describes the ferromagnetism in a spin glass. The coupling between spins induces a spin into a ferromagnetism or anti-ferromagnetism state. The energy of the spin glass, i.e., the Hamiltonian (H), without external magnetic fields, is given by [10]:

$$H = -\sum_{i,j} J_{ij}\sigma_i\sigma_j, \tag{1}$$

The authors are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada (e-mail: bailiang@ualberta.ca; ttzhang@ualberta.ca; xg6@ualberta.ca; jhan8@ualberta.ca).

where σ_i (or σ_j) is the state of the ith (or jth) spin with either an upward ($+1$) or downward (-1) state, and J_{ij} is the coupling coefficient for the ith and the jth spins.

B. The Oscillator-based Ising Machine

The OIM implements the Ising model by an oscillator network and encodes spins into the oscillator phases. By injecting a second harmonic signal, oscillators are set under second harmonic injection locking (SHIL). In this way, oscillator phases are polarized, which can be used to represent the spins in an Ising model. The dynamics of the oscillator network under SHIL is governed by the generalized Adler's equation, as [6]:

$$\begin{aligned}\frac{d\phi_i(t)}{dt} = \quad & \omega_i - \omega^* + \omega_i K_s s_i(2\phi_i(t)) \\ & + \omega_i K \textstyle\sum_j [J_{ij} c_{ij}(\phi_i(t) - \phi_j(t))],\end{aligned} \tag{2}$$

where $\phi_i(t)$ is the phase of the ith oscillator at time t; ω_i is the frequency of the ith oscillator; ω^* is the central frequency of the oscillator network; $c_{ij}(\cdot)$ is a periodic function for the coupling between the ith and jth oscillators, referred to as the coupling function; $s_i(\cdot)$ is a periodic function for the second harmonic signal injected to the ith oscillator, referred to as the second harmonic injection function; and K and K_s are the modulation parameters for the coupling coefficient and the second harmonic signal, respectively.

For an RC-tank based oscillator network, $c_{ij}(\cdot)$ and $s_i(\cdot)$ are defined as sinusoidal functions. All the oscillators in the system are expected to share the same frequency without considering circuit and device variability, hence $\omega_i = \omega^*$. Since ω_i is a constant, it is merged into the modulation parameters, K and K_s, for simplicity. Then, the global Lyapunov function of the network, $E(\vec{\phi}(t))$, is given by [6]:

$$\begin{aligned}E(\vec{\phi}(t)) = & -K \textstyle\sum_{i,j} J_{ij} \cos(\phi_i(t) - \phi_j(t)) \\ & - K_s \textstyle\sum_i \cos(2\phi_i(t)).\end{aligned} \tag{3}$$

When an oscillator is under SHIL, its phase can only be 0 or π. Therefore, $\cos(\phi_i(t) - \phi_j(t))$ outputs -1 or $+1$, respectively depending on whether $\phi_j(t)$ and $\phi_i(t)$ are out of phase ($\phi_i(t) \neq \phi_j(t)$) or not ($\phi_i(t) = \phi_j(t)$), and $\cos(2\phi_i(t))$ always produces 1. When $K = 1$, (3) is equivalent to the Hamiltonian of the Ising model, as given in (1), only with a constant offset.

To deal with the scalability issue in simulation, Wang and Roychowdhury utilized a smoothened square function, i.e., $\tanh(10\sin(\cdot))$, as the coupling function, instead of using $\sin(\cdot)$ [6]. To help the system escape from local minima, a time-variant modulation parameter is used for the coupling coefficient, and Gaussian noise is introduced to the system. Thus, the modified Adler's equation is obtained as [6]:

$$\begin{aligned}d\phi_i(t) = \quad & [-K \textstyle\sum_j J_{ij} \tanh(10\sin(\phi_i(t) - \phi_j(t))) \\ & - K_s \sin(2\phi_i(t))]dt + K_n dW_t,\end{aligned} \tag{4}$$

where dW_t is a time-variant Gaussian noise with the mean 0 and the variance dt, and K_n is the modulation parameter. The parameters K, K_s, and K_n are usually given by [6]:

$$\begin{cases} K = 1 + \dfrac{t}{20} \\ K_s = 1 + 2\tanh(10\cos(\pi t)) \\ K_n = 0.8 \end{cases}. \tag{5}$$

III. The Simulated Oscillator-based Ising Machine

A. An FPGA-Oriented Model of OIMs

For an efficient implementation of OIMs on an FPGA, the modified Adler's equation in (4) is revised by following the observations below:

1) An experimental evaluation shows that omitting the Gaussian noise in (4) only decreases the solution quality by a mere 0.8% for solving max-cut problems (although the details are not shown due to limited space). As shown in [11], using a random number generator for each spin to introduce Gaussian noise in a 1024-spin FPGA-based annealer increases hardware costs by at least tenfold. Therefore, Gaussian noise is omitted in this work to save on hardware usage.
2) All the periodic functions in (4) have a period of 2π, which is accurate for describing physical models but inefficient for an FPGA implementation. To address this issue, the period is reduced to 2, thus eliminating the multiplication and division involving π.
3) For ease in implementation, the complex coupling function ($\tanh(10\sin(\cdot))$) and the second harmonic injection function ($\sin(\cdot)$) in (4) are approximated by simple piece-wise linear functions, which can be implemented using bit-wise operations with simple addition and subtraction. Moreover, a function, called an evolution bound function, is used to constrain the phase values to be within $[-2, 2]$. The proposed coupling function, the second harmonic injection function, and the evolution bound function are defined as follows:

- Coupling Function (C): The coupling function in (4) utilizes the functions $\tanh(x)$ and $\sin(x)$, which are hardware-consuming for FPGA implementations. The $\tanh(10\sin(x))$ is approximated by a piece-wise function with a period of 2, where the definition over $[0, 2)$ is defined as:

$$C(x) = \begin{cases} 0 & x = 0 \; or \; 1 \\ 1 & 0 < x < 1 \\ -1 & 1 < x < 2 \end{cases}. \tag{6}$$

  Fig. 1a presents the comparison between the original and proposed coupling function.
- Second Harmonic Injection Function (S): Equation (4) uses the $\sin(x)$ function as the second harmonic injection function. As shown in Fig. 1b, to facilitate hardware implementation while preserving its periodic nature along with its ascending and descending trends, the following piece-wise linear function with a period of 2 is proposed:

$$S(x) = \begin{cases} 2x & 0 \leq x < 0.5 \\ 2 - 2x & 0.5 \leq x < 1.5 \\ 2x - 4 & 1.5 \leq x < 2 \end{cases}. \tag{7}$$

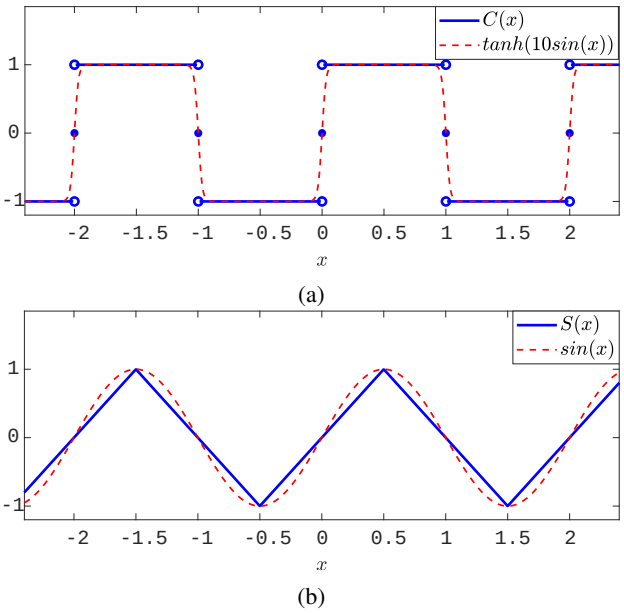

Fig. 1: The comparisons between the original and the proposed approximate (a) coupling function, and (b) second harmonic injection function.

- Evolution Bound Function (B): Since all functions exhibit a period of 2 in the proposed FPGA-oriented model, modulo and addition by a power of 2 to the oscillator phases do not affect the results. An evolution bound function with period of 4 is introduced to limit the oscillator phases to be within $[-2, 2]$, given by:

$$B(x) = \begin{cases} x & 0 \le x < 2 \\ x - 4 & 2 \le x < 4 \end{cases}. \quad (8)$$

Therefore, for an N-spin Ising model, the proposed FPGA-oriented model searches for a solution by solving the following differential equation for each spin:

$$d\phi_i(t) = [-K \sum_{j=1}^{N} J_{ij} C(\phi_j(t) - \phi_i(t)) - K_s S(2\phi_i(t))]dt, \quad (9)$$

where K is defined in (5) and K_s is given by:

$$K_s = 1 + 2 \times M(x), \quad (10)$$

where the function $M(x)$ is modified from the coupling function $C(x + 0.5)$ by setting $M(1.5)$ to 1 and $M(0.5)$ to -1, so excluding 0 as a functional output. It is obtained by shifting $C(x)$ to the left by 0.5 in order to compensate the difference between $tanh(sin(\cdot))$ in (4) and $tanh(cos(\cdot))$ in (5). The oscillator phase is updated by first calculating the new oscillator phase and then limiting it to be within $(-2, 2]$ as

$$\phi_i(t + 1) = B(\phi_i(t) + d\phi_i(t)), \quad (11)$$

where $\phi_i(t+1)$ is the phase of the ith oscillator at time $t+1$.

B. The Proposed Algorithm

In this work, we consider a simple Euler integration method to solve the differential equation in (9). As shown in Algorithm 1, it requires external inputs including the coupling coefficient matrix $\boldsymbol{J}$, the simulation stop time T_{stop}, and the time step for each iteration ΔT. The output is a matrix for oscillator phases, $\boldsymbol{\Phi}$. The oscillator phases are initialized by using uniformly distributed random numbers between -1 and $+1$ (Line 1). At the qth iteration and at time $t = \Delta T \cdot q$, K and K_s are first updated according to (5) and (10) (Lines 3 and 4). For the ith oscillator, the phase difference between the ith and other oscillators ($\boldsymbol{pd_i}$) is computed for the coupling function (Line 6). Then, the output from the coupling function is multiplied with the coupling coefficient ($\boldsymbol{J_i}$) and its modulation parameter K to compute the local field on the ith oscillator (lf_i) (Line 7). The second harmonic injection signal (ss_i) is then generated depending on the ith oscillator phase at the $(q-1)$th iteration ($\Phi_{q-1,i}$) (Line 8). The cumulative effect from lf_i and ss_i on the ith oscillator, $\Delta\Phi_{q,i}$, is then computed (Line 9). Finally, all the oscillator phases at the qth time step ($\boldsymbol{\Phi_q}$) are updated depending on the oscillator phases at the $(q-1)$th time step ($\boldsymbol{\Phi_{q-1}}$) and the cumulative effect multiplied by ΔT based on the evolution bound function (Line 11). The oscillator phases will continue being updated until T_{stop} is reached.

Algorithm 1 Oscillator-based Ising Machine Simulation Algorithm

Input: $\boldsymbol{J}$, T_{stop}, **and** ΔT
Output: $\boldsymbol{\Phi}$

1: Initialize oscillator phases
2: **for** $q = 1$ to $T_{stop}/\Delta T$ **do**
3: $K \Leftarrow 1 + q\Delta T/20$
4: $K_s \Leftarrow 1 + 2 \times M(q\Delta T)$
5: **for** each oscillator, i, **do**
6: $\boldsymbol{pd_i} \Leftarrow \Phi_{q-1,i} \cdot ones(1, N) - \boldsymbol{\Phi_{q-1}}$
7: $lf_i \Leftarrow -K(\boldsymbol{J_i} \cdot C(\boldsymbol{pd_i}))$
8: $ss_i \Leftarrow -K_s S(2\Phi_{q-1,i})$
9: $\Delta\Phi_{q,i} \Leftarrow lf_i + ss_i$
10: **end for**
11: $\boldsymbol{\Phi_q} \Leftarrow B(\boldsymbol{\Phi_{q-1}} + \Delta T \cdot \boldsymbol{\Delta\Phi_q})$
12: **end for**

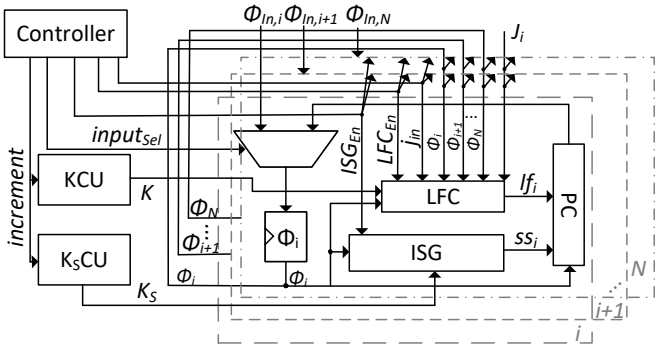

Fig. 2: The top-level circuit architecture of SOIM.

C. Circuit Design

Fig. 2 shows the top-level architecture of the proposed SOIM. An N-spin SOIM consists of two computing units for different parameters, one for K (KCU) and the other for K_s (K_sCU), as well as N injection signal generators (ISGs) for SHIL, N local field calculators (LFCs), and N phase calculators (PCs). Note that the SOIM uses the fixed point

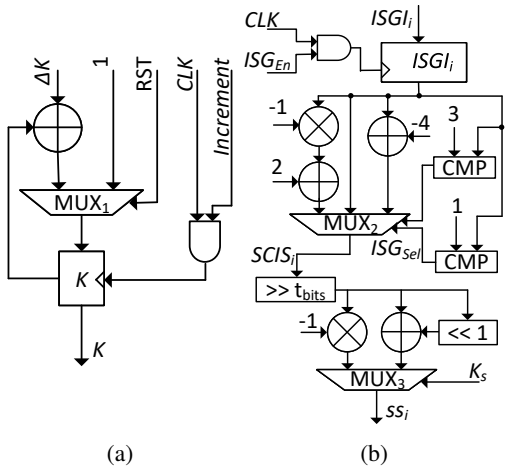

Fig. 3: The block diagrams for (a) KCU, and (b) ISG.

number representation with a total of $(int + frac)$ bits, int bits for the integer part and $frac$ bits for the fractional part. The modules in the SOIM are designed as follows:

1) Computation Unit for K (KCU): As shown in Fig. 3a, the KCU computes the time-variant parameter K in (5) (Algorithm 1, Line 3). When $RST = 1$, the register is initialized to decimal 1. When all the oscillator phases have been updated, the $increment$ is set to high to enable the accumulation. Each accumulation increases K by a constant step ΔK, where $\Delta K = \Delta T/20$. The output from the accumulator is used for the lf computation in the LFC.

2) Computation Unit for K_s (K_sCU): Similarly, the K_sCU uses an accumulator to compute K_s in (10). The accumulator is enabled by the $increment$ signal and is reset when an overflow occurs. According to (10), the value of K_s is either -1 or 3 since $M(t)$ can only be ± 1. To reduce the hardware overhead in the ISG, the K_sCU encodes -1 as "1" and 3 as "0". K_s is then sent to the ISG to generate the second harmonic injection signal for SHIL.

3) Injection Signal Generator (ISG) for SHIL: Fig. 3b shows the block diagram for the ISG. The ISG is enabled by ISG_{En} and it takes two CLK cycles to generate the second harmonic injection signal (Line 8). In the first CLK cycle, it computes the second harmonic injection signal. According to Algorithm 1 (Line 8), Φ_i is first multiplied by 2 to obtain $2\Phi_i$, then it undergoes a modulo 2 operation since $S(2\Phi_i)$ is a periodic function with 2 as the period, as shown in (7). Finally, the remainder is multiplied by 2 again because there is a multiplication with 2 operation in the three candidate values of the result of $S(2\Phi_i)$ in (7). $ISGI_i[frac + 1 : 0] = \Phi_i[frac - 1 : 0]$ & "00" serves as the input to implement the aforementioned operations by using simple bit-wise operation, where & is the concatenation operator. Since $ISGI_i$ carries the $2x$ value in (7), it ranges from 0 to 4 and is compared with 3 and 1 to determine the output of the second harmonic injection function by using MUX_2 from the three candidate values. The output of MUX_2, $SCIS_i$, multiplied by ΔT is implemented by arithmetic right shifting $SCIS_i$ by t_{bits} bits, where $t_{bits} = \log_2 \Delta T$. During the second CLK cycle, $SCIS_i$ is multiplied by K_s to generate ss_i. There are two possible values for ss_i: $-1 \times SCIS_i$ and $3 \times SCIS_i$ implemented as $2 \times SCIS_i + SCIS_i$. Taking these two values as the inputs of MUX_3, ss_i is determined by K_s.

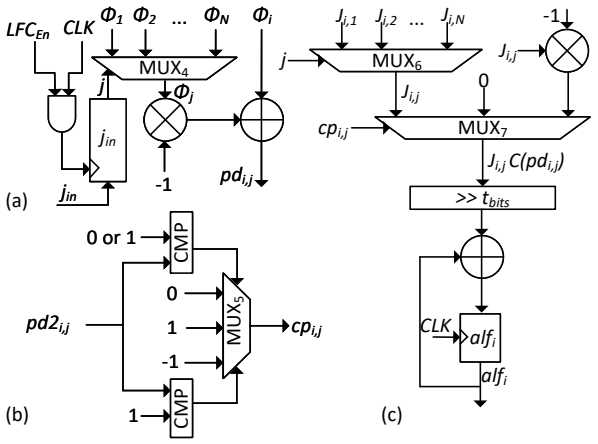

Fig. 4: The block diagram for LFC.

4) Local Field Calculator (LFC): Fig. 4 presents the block diagrams for the LFC. The LFC performs the computation in Algorithm 1, Lines 6 and 7. It can be divided into four steps, as follows:

- The first stage is to compute the phase difference (Line 6). As shown in Fig. 4a, the inputs are the oscillator phases vector $\mathbf{\Phi}$, the index j_{in}, the enable signal LFC_{En}, and the phase of the ith oscillator Φ_i. The j_{in} register outputs the new index j when LFC_{En} and CLK are high. Otherwise, it outputs the previous value stored. The N-to-1 MUX_4 is used to select the jth oscillator phase Φ_j from $\mathbf{\Phi}$. Then, the phase difference $pd_{i,j}$ between the ith and jth oscillators is computed by using subtraction (implemented as the addition of the negative phase in Fig 4a).
- The second stage is to compute the output from the coupling function (6), taking $pd2_{i,j}$ as the input, as shown in Fig. 4b. $pd2_{i,j}$ depends on the phase difference $pd_{i,j}$ calculated in the first step. It is the remainder of $pd_{i,j}$ divided by 2 in order to adapt to the periodic coupling function, such that $pd2_{i,j}[frac : 0] = pd_{i,j}[frac : 0]$, where the most significant bit of $pd2_{i,j}$ is the integer bit and others are fractional bits. The output of the coupling function $C(pd_{i,j})$ $(= C(pd2_{i,j}))$ is encoded into $cp_{i,j}$ using two bits with "00", "01" and "10" indicating the decimal value -1, 0 and 1, respectively. For example, if $pd2_{i,j}$ equals to 0 or 1, $cp_{i,j}$ = "01"; if the decimal value of $pd2_{i,j}$ falls within the range of $(1, 2)$, i.e., $pd2_{i,j}[frac]$ = "1", then $cp_{i,j}$ = "00"; otherwise, $cp_{i,j}$ = "10".
- The third stage is to compute $\sum_j J_{i,j} C(pd_{i,j}) \Delta T$ for the local field. The ΔT is multiplied at this step to prevent overflow and underflow in the multiply-accumulate (MAC) operation. As shown in Fig. 4c, it takes the index j, the coupling coefficient vector for the ith oscillator $\boldsymbol{J_i}$ and $cp_{i,j}$ as inputs. First, the coupling coefficient, $J_{i,j}$, is selected from $\boldsymbol{J_i}$ by using MUX_6. Then, using $cp_{i,j}$ as the select signal for MUX_7, the result of $J_{i,j}C(pd_{i,j})$ in Line 7 is obtained. When $cp_{i,j}$ = "01", MUX_7 outputs 0; when $cp_{i,j}$ = "00", MUX_7 outputs $-J_{i,j}$; otherwise,

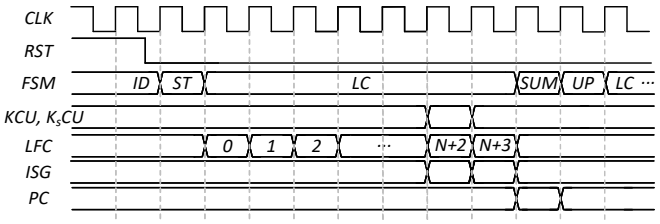

Fig. 5: The timing chart of the SOIM.

MUX_7 outputs $J_{i,j}$. It is then arithmetic-right shifted by t_{bits} bits to implement $J_{i,j}C(pd_{i,j})\Delta T$, which is added to an accumulator to obtain alf_i, an intermediate result of lf_i. This process needs to iterate over $j = 1$ to N cycles to compute the dot product of $\boldsymbol{J_i} \cdot C(\boldsymbol{pd_i})\Delta T$.

- In the last stage, the modulation parameter for the coupling coefficient K generated in the KCU is multiplied by alf_i to compute the $lf_i\Delta T$ (Line 7).

5) Phase Calculator (PC): The PC calculates the new oscillator phase depending on lf_i, ss_i and the current phase (Algorithm 1, Lines 9 and 11). Note that the operations of multiplication with -1 in Lines 7 and 8, are not implemented in the LFC and the ISG, but is implemented in this unit instead. The evolution bound function in (8) is applied to the sum of lf_i, ss_i and the current phase, which is represented by $(int+frac)$ bits. By keeping the least significant $(frac+2)$ bits, the remainder of the sum divided by 4, denoted by x, is obtained. In this way, $0 \leq x < 4$. Then, according to (8), if $0 \leq x < 2$, i.e., the $(frac+2)$th bit is "0", the most significant $(int-2)$ bits of the sum are set to "0"; if $2 \leq x < 4$, i.e., the $(frac+2)$th bit is "1", the remainder needs to be subtracted by 4 which is accomplished by adding -4 in 2's complement, equivalent to setting the most significant $(int-2)$ bits to "1". Thus, these operations can be implemented by copying the $(frac+2)$th bit to the most significant $(int-2)$ bits in the sum.

D. Timing Design

Fig. 5 provides the timing chart of the SOIM for the first iteration. The system is controlled by a finite state machine (FSM) with five states: IDLE (or ID), START (Or ST), LC, SUM, and UP. The SOIM uses an asynchronous $reset$ signal that forces the FSM go into the IDLE state when $reset = 1$. When $reset = 0$, the FSM will go to the START state to load the coupling coefficient and initialize the phases. After one clock cycle, the local field computation begins and the machine jumps to LC, in which the local field is computed in the LFC. The local field computation process uses a four-stage pipeline design, which is controlled by a local FSM as follows:

- The first stage of the pipeline is to compute $pd_{i,j}$.
- The second stage is to compute the coupling function with the $pd_{i,j}$ as the input.
- The third stage is to compute the accumulated local field on the ith oscillator.
- The fourth stage is to multiply the accumulated local field by its modulation parameter, K.

The LC state takes $N+3$ cycles due to the MAC operation. When this local FSM reaches $N+2$ iterations, the controller will send an enable signal ISG_{En} to the ISG, as shown in Fig. 2, and the $increment$ signal is set to high. When finishing $N + 3$ cycles, the FSM goes to the SUM state, where the new oscillator phase is computed by the PC. The FSM is changed to the UP state after one clock cycle to load newly computed oscillator phases to internal registers. After that, the FSM will loop back to the LC and start a new iteration for updating the oscillator phases.

IV. Experimental Results

A. Solving Max-Cut Problems

In this subsection, the performance and feasibility of the proposed algorithm is evaluated in comparison with the classical SA [12], the OIM simulation [6], and the DEOIM [9] algorithms. We consider twenty benchmark datasets from the Gset [13] of 800-node max-cut problems with varying connectivity.

The average (AVG) and maximum (MAX) of the cut values from ten runs are normalized by the best-known cut values [6] for measuring the solution quality of different algorithms. For the proposed algorithm, the unitless parameters are set as $T_{stop} = 40$ and $\Delta T = 2^{-9}$. Thus, the total number of iterations for updating the spin states is given by $T_{stop}/\Delta T$. The phases from all oscillators at all iterations are evaluated to obtain the maximum cut value. The SOIM and DEOIM simulations are conducted on an AMD processor, Ryzen 7 5800X3D (4.5 GHz), with Python 3.11.7. The simulation results for SA and OIM simulation algorithms are obtained from [6], where the SA was run on Intel Xeon E3-1245v2 (3.4 GHz) and the OIM simulation algorithm was run on Intel Xeon E5-1603v3 (2.8 GHz).

Fig. 6 compares the MAX and AVG results obtained by using the proposed algorithm with the MAX results obtained by using the SA, the OIM simulation, and the DEOIM algorithms. In an evaluation across twenty max-cut problems, the proposed SOIM algorithm returns average values of 99.1% in MAX and 98.2% in AVG. Compared to DEOIM, the SOIM achieves an average 8% increase in MAX. It indicates that an SOIM-based system has a smaller probability of getting stuck in local minima. When compared to the SA and the OIM simulation algorithms, there is a marginal degradation in MAX by approximately 0.9% and 1.8%, respectively. The loss in solution quality is attributed to several key factors as follows:

- The use of piece-wise functions as the coupling and the second harmonic injection functions results in a slightly different evolution of oscillator phases.
- The exclusion of Gaussian noise reduces the probability for the SOIM to escape from local minima.
- Using the Euler integration approach provides a limited computational accuracy for solving the differential equations in the OIM system.

However, the proposed SOIM is expected to have a significant improvement in hardware efficiency.

B. Hardware Performance

The hardware performance of the SOIM is compared with the FPGA Annealer (FA) [11] and SB Machine (SBM) [14],

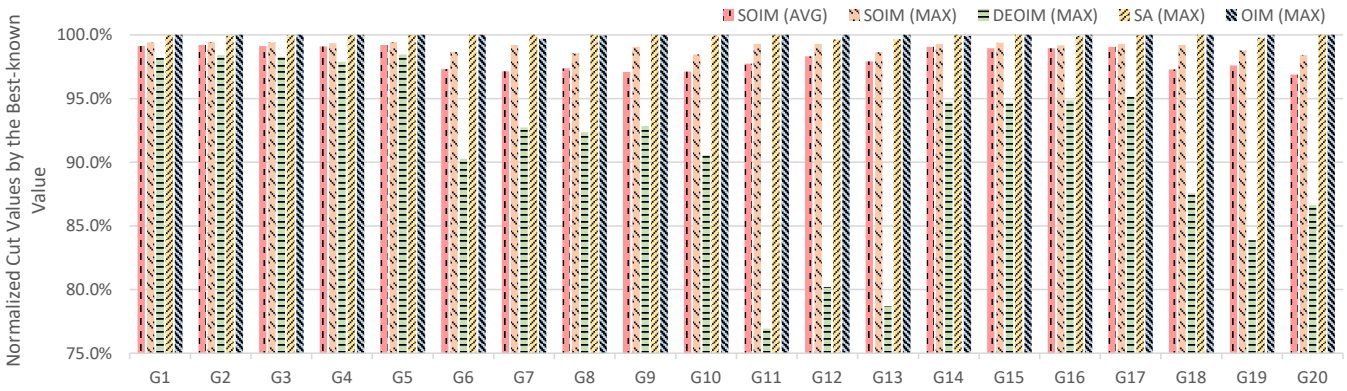

Fig. 6: The performance for solving max-cut problems by using different Ising model-based algorithms.

TABLE I: Comparison of FPGA Implementations for Different Ising Machines

Design	Size	Coefficient Precision	Topology	LUTs	FFs	DSPs
SOIM	50	5 bits	Complete[1]	5,706	3,456	50
SOIM	100	5 bits	Complete[1]	11,622	6,928	100
SOIM	500	5 bits	Complete[1]	58,989	34,631	500
FA [11]	1,024	3 level	Sparse[2]	468,183	-	-
SBM [14]	2,048	1 bit	Complete[1]	260,953	281,274	104

[1] For the complete topology, all the spins are connected to each other.
[2] For the sparse topology, not all the spins are coupled together.

as shown in Table I. The SOIMs were implemented on a Zynq UltraScale+ ZCU104FPGA board with an XCZU7EV-FFVC1156-2-E processor, simulated and synthesised in Vivado 2022.2. This FPGA board is equipped with 230,400 Look Up Tables (LUTs), 460,800 Flip-Flops (FFs), and 1,728 Digital Signal Processing Elements (DSPs).

As a preliminary work, we constructed the SOIM with a small size and verified its functionality for a 10-spin system. The coupling coefficient and the initial phase are set as inputs to the system, requiring significant I/O usage. Larger SOIMs are synthesized with the presumption that the I/O constraint is ignored and will be addressed in future work. Although we only present the hardware measurements for the system with 50, 100, and 500 spins, the hardware performance for a larger size can be estimated. The DSPs are used for the multiplication in the LFC, so its usage linearly increases with the number of spins. When scaling up the number of spins to 1024, as can be inferred from Table I, the SOIM is expected to use fewer LUTs than the FA; when scaled up to 2048 spins, it is expected that the SOIM requires fewer FFs than the SBM. Compared with the FA, the SOIM offers a fully connected topology, which makes it easy to embed a given COP to the Ising machine. Compared with the SBM, it has a stronger ability in solving complex COPs since it attains 5-bit precision for the coupling coefficient.

V. Conclusions

As a first FPGA design for a fully connected oscillator-based Ising machine, the SOIM utilizes an FPGA-oriented model to simulate the oscillator network using revised differential equations. For hardware efficiency, the complex periodic functions for describing the coupling between oscillators and the second harmonic signal are both replaced by piecewise linear functions. To further reduce hardware, Gaussian noise is discarded in the system with a negligible loss in solution quality. A heuristic algorithm is then developed for this model to solve the differential equations by using a Euler integration method. Based on the proposed algorithm, SOIMs with a fully connected topology are implemented and synthesized on FPGAs. The experimental results show that the SOIM has an advantage in hardware performance with a slight loss in solution quality, approximately by 1% in the maximum and 1.8% in the average max-cut values found. Based on this preliminary work, a larger SOIM will be implemented and verified on FPGAs by fully exploiting the limited on-board resources. Finally, large-scale COPs with dense connectivity will be investigated in future work.

References

[1] J. A. Roy, S. N. Adya, D. A. Papa, and I. L. Markov, "Min-cut floorplacement," *IEEE TCAD*, vol. 25, no. 7, pp. 1313–1326, 2006.

[2] S. Siddhaye, K. Camarda, M. Southard, and E. Topp, "Pharmaceutical product design using combinatorial optimization," *Comput. Chem. Eng.*, vol. 28, no. 3, pp. 425–434, 2004.

[3] T. Zhang, Q. Tao, B. Liu, and J. Han, "A review of simulation algorithms of classical Ising machines for combinatorial optimization," in *ISCAS*. IEEE, 2022, pp. 1877–1881.

[4] D. Oku, K. Terada, M. Hayashi, M. Yamaoka, S. Tanaka, and N. Togawa, "A fully-connected Ising model embedding method and its evaluation for CMOS annealing machines," *IEICE Trans. Inf. Syst.*, vol. 102, no. 9, pp. 1696–1706, 2019.

[5] T. Zhang and J. Han, "Quantized simulated bifurcation for the Ising model," in *IEEE NANO Conf.* IEEE, 2023, pp. 715–720.

[6] T. Wang and J. Roychowdhury, "OIM: Oscillator-based Ising machines for solving combinatorial optimisation problems," in *UCNC*. Springer, 2019, pp. 232–256.

[7] J. Vaidya, R. Surya Kanthi, and N. Shukla, "Creating electronic oscillator-based Ising machines without external injection locking," *Sci. Rep.*, vol. 12, no. 1, p. 981, 2022.

[8] W. Moy, I. Ahmed, P.-w. Chiu, J. Moy, S. S. Sapatnekar, and C. H. Kim, "A 1,968-node coupled ring oscillator circuit for combinatorial optimization problem solving," *Nat. Electron.*, vol. 5, no. 5, pp. 310–317, 2022.

[9] S. Sreedhara, J. Roychowdhury, J. Wabnig, and P. Srinath, "Digital emulation of oscillator Ising machines," in *DATE*. IEEE, 2023, pp. 1–2.

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

[11] C. Yoshimura, M. Hayashi, T. Okuyama, and M. Yamaoka, "FPGA-based annealing processor for Ising model," in *CANDAR*. IEEE, 2016, pp. 436–442.

[12] T. G. Myklebust, "Solving maximum cut problems by simulated annealing," *arXiv preprint arXiv:1505.03068*, 2015.

[13] S. E. Karisch. [Online]. Available: https://web.stanford.edu/ yyye/yyye/Gset/

[14] K. Tatsumura, A. R. Dixon, and H. Goto, "FPGA-based simulated bifurcation machine," in *FPL*. IEEE, 2019, pp. 59–66.