# pc-COP: An Efficient and Configurable 2048-p-Bit Fully-Connected Probabilistic Computing Accelerator for Combinatorial Optimization

Kiran Magar*, Shreya Bharathan† and Utsav Banerjee*
*Electronic Systems Engineering, Indian Institute of Science, Bengaluru, India
†National Institute of Technology, Tiruchirappalli, India
Email: kirankailash@iisc.ac.in, shreyabharat2306@gmail.com, utsav@iisc.ac.in

***Abstract*—Probabilistic computing is an emerging quantum-inspired computing paradigm capable of solving combinatorial optimization and various other classes of computationally hard problems. In this work, we present pc-COP, an efficient and configurable probabilistic computing hardware accelerator with 2048 fully connected probabilistic bits (p-bits) implemented on Xilinx UltraScale+ FPGA. We propose a pseudo-parallel p-bit update architecture with speculate-and-select logic which improves overall performance by $4\times$ compared to the traditional sequential p-bit update. Using our FPGA-based accelerator, we demonstrate the standard G-Set graph maximum cut benchmarks with near-99% average accuracy. Compared to state-of-the-art hardware implementations, we achieve similar performance and accuracy with lower FPGA resource utilization.**

***Index Terms*—probabilistic computing, fully connected, FPGA, hardware accelerator, max cut, combinatorial optimization, quantum-inspired computing, Ising machine.**

## I. Introduction

Quantum computing [1], [2], [3], [4] is a pioneering paradigm in computation which applies the principles of quantum mechanics to revolutionize problem-solving capabilities. While quantum computing holds tremendous potential, it is still in the nascent stages of development and faces significant challenges on its path to practicality and widespread adoption. These challenges have motivated the emergence of many new physics-inspired computing models such as probabilistic computing [5], stochastic computing [6], simulated annealing [7], probabilistic annealing [8] and parallel tempering [9].

Probabilistic computing is one of the emerging quantum-inspired computing paradigms and it involves the manipulation of unstable stochastic units known as *probabilistic bits* or *p-bits*. Multiple p-bits are interconnected together to construct *probabilistic circuits* or *p-circuits*. Unlike classical bits which are deterministically either 0 or 1 and quantum bits (qubits) which can exist in a superposition of 0 and 1, p-bits rapidly fluctuate between 0 and 1. While qubits require near-absolute-zero temperatures for accurate functionality, such p-bits and p-circuits can be realized at room temperature, thus enabling many novel applications at the intersection of classical and quantum hardware using existing as well as emerging technologies [10], [11]. Recent literature has demonstrated the immense potential of probabilistic computing using various software and hardware implementation platforms such as micro-controllers [12], general purpose micro-processors (CPUs) and graphics processing units (GPUs) [13], [14], [15], field-programmable gate arrays (FPGAs) [16], magnetic tunnel junctions (MTJs) [17], resistive random-access memories (RRAMs) [18], ferro-electric field-effect transistors (FeFETs) [19] and threshold switch devices (TSDs) [20]. However, these hardware implementations have been limited to either small-scale p-circuits using emerging nano-devices or preliminary architectures using FPGAs. They have limited circuit-level analysis, thus leaving plenty of room for design space exploration and algorithm-specific architectural improvement.

Combinatorial optimization [21], [22] is an important class of hard problems which can be solved efficiently using probabilistic computing. In this work, we present pc-COP, an efficient and configurable probabilistic computing hardware accelerator with 2048 fully connected p-bits implemented on state-of-the-art Xilinx UltraScale+ FPGA [23]. It is capable of solving large-scale graph maximum cut combinatorial optimization problems [7], [24] with high accuracy. Our logarithmic adder tree design for sum-of-products computation boosts overall performance. We approximate the activation function and tune the precision of the annealing schedule to reduce FPGA resource utilization. Our proposed pseudo-parallel p-bit update architecture with speculate-and-select logic improves performance by $4\times$ compared to the traditional sequential p-bit update. We implement pc-COP on a Xilinx Zynq UltraScale+ MPSoC ZCU104 Evaluation Board and demonstrate near-99% average accuracy across various G-Set maximum cut benchmarks up to 2,000 nodes [25].

## II. Background

### A. Probabilistic Computing

Probabilistic computing is an emerging quantum-inspired computing paradigm capable of solving many interesting problems such as combinatorial optimization, machine learning, quantum emulation and integer factorization [5], [26], [27], [28], [29]. The operation of a p-circuit with several p-bits is described in Algorithm 1. The system state is defined as $m = \{m_i \,|\, 1 \leq i \leq N_m\}$, where each $m_i \in \{-1, +1\}$ denotes the corresponding p-bit value. Each p-bit $m_i$ is updated sequentially based on the weights $J_{i,j}$ and bias values $h_i$. A

**Algorithm 1** Overview of p-circuit operation [5], [26]

**Require:** number of p-bits $N_m$, interconnection weight matrix $\mathbf{J} = [J_{i,j}]_{N_m \times N_m}$ and bias vector $\mathbf{h} = [h_i]_{N_m \times 1}$ for application-specific p-circuit, number of samples $N_s$

**Ensure:** final p-bit state $m$

```
1: define p-bit state m = {m_i | 1 ≤ i ≤ N_m}
2: randomly initialize p-bits m_i ∈ {−1, +1} ∀ 1 ≤ i ≤ N_m
3: for (s = 1; s ≤ N_s; s = s + 1) do
4:     for (i = 1; i ≤ N_m; i = i + 1) do
5:         I_i ← β × ( h_i + Σ_{j=1}^{N_m} J_{i,j} m_j )
6:         m_i ← sgn( rand(−1, +1) + tanh( I_i ) )
7:     end for
8: end for
9: return m = {m_i | 1 ≤ i ≤ N_m}
```

global constant $\beta$ is used to control the overall strength of p-bit interconnections. A complete sequence of updating all the $N_m$ p-bits in a p-circuit is referred to as a *sample*.

This process is repeated multiple times, thus generating $N_s$ samples. The stochastic neural network representation is similar to Boltzmann machines [5]. The p-bit update equation ($I_i = \beta \times ( h_i + \sum_{j=1}^{N_m} J_{i,j}\, m_j )$) resembles Ising machines [27] while the sequential nature of the update resembles the iterative evolution in Gibbs sampling [30]. The energy of the system with state $m$ is defined as $E(\{m\}) = -( \sum_{i<j} J_{i,j} m_i m_j + \sum h_i m_i )$ which again resembles the quadratic energy model in Ising machines [30]. The inherent stochasticity and non-linear activation function in each p-bit, which is a unique feature of probabilistic computing, ensures that various states $m$ are visited according to their corresponding Boltzmann probability $p_{\{m\}} \propto exp[-\beta E(\{m\})]$, where $\beta$ acts as an *inverse pseudo-temperature* which can enhance or suppress probabilities based on energy minima. The system of p-bits evolves over consecutive samples to converge towards a low-energy state corresponding to an optimum or near-optimal solution of the problem encoded in the p-circuit. The value of $\beta$ can be tuned across samples to achieve better convergence, which bears resemblance to simulated annealing [27], [30].

Recent literature has explored the use of emerging technologies to efficiently realize the p-bit functionality in hardware [5], [26], [17], [18], [31], [11], [20], [19]. But, large-scale implementations of p-circuits using such emerging nano-devices are yet to be demonstrated experimentally. Therefore, FPGA-based implementations offer a promising near-term alternative. Preliminary FPGA-based architectures have been demonstrated in [16], [27], [30], [32], [33]. However, detailed circuit-level analysis and architectural optimizations for probabilistic computing are yet to be explored.

### B. Combinatorial Optimization Problems

Combinatorial optimization is a sub-field of mathematical optimization which involves finding the optimal solution out of a finite but large set of possibilities where exhaustive search is intractable [21], [22]. In this work, we explore the efficacy of probabilistic computing in solving the prototypical combinatorial optimization problem (COP) of graph maximum cut, also known as *max-cut* [7]. The objective of the max-cut problem is to partition the vertices $V$ of a graph $G = (V, E)$ into two complementary sets $S$ and $T$ such that the number of edges ($\in E$) between $S$ and $T$ is as large as possible. The corresponding p-circuit can be constructed by assigning a p-bit $m_i$ corresponding to each vertex $v_i \in V$. Then, the optimal max-cut solution will result in $m_i = +1$ if $v_i \in S$ and $m_i = -1$ if $v_i \in T$ such that the objective function $\sum_{i<j} w_{i,j} m_i m_j$ is minimized, where $w_{i,j}$ is the weight of the edge connecting vertices $v_i$ and $v_j$ [24]. Therefore, the p-bit interaction coefficients are obtained as $J_{i,j} = -w_{i,j}$ and $h_i = 0$. The Stanford G-Set benchmark dataset [25], containing various random, toroidal and planar graphs, is typically used to evaluate max-cut solver implementations. G-Set contains graphs with $w_{i,j} \in \{0, 1\}$ as well as $w_{i,j} \in \{-1, 0, +1\}$, therefore 2-bit interaction coefficients $J_{i,j}$ are required.

## III. Hardware Architecture

Fig. 1 shows the top-level architecture of pc-COP, our proposed hardware accelerator for solving large-scale max-cut COPs using probabilistic computing. It supports max-cut instances up to 2048 nodes using 2048 p-bits stored in the 2048-bit register m_Reg, where -1 and +1 p-bit values are encoded as 0 and 1 respectively. According to step 2 of Algorithm 1, the initial random state of the p-bit register m_Reg is configured using a 2048-bit external input. The corresponding 2048×2048 matrix **J** of 2-bit interaction coefficients (-1, 0 and + 1 encoded as 11, 00 and 01 respectively) is stored in the $2048 \times 4096$-bit = 8 Mb memory J_Mem. Although J_Mem is implemented using 256 physical Block RAM (BRAM) slices of 32 Kb each (excluding error correction coding bits) in FPGA, they are organized such that an entire 4096-bit row of **J** can be read in a single cycle, as required in step 5 of Algorithm 1. The J_Mem memory is configured with the problem-specific **J** matrix 32 bits at a time using the 18-bit address input and an address decoder. The functionality of Algorithm 1 is implemented in the *p-Bit Update Core* and managed by a finite state machine (FSM) and control registers. A 512-bit seed input is used to configure the stochasticity in the p-bit update circuitry. The 24-bit inputs $\beta_{initial}$ and $\beta_{anneal-rate}$ are used to configure the inverse pseudo-temperature $\beta$. A 32-bit instruction input is used to program the accelerator as well as configure the number of p-bits $N_m \leq 2048$ and the number of samples $N_s$. The instruction format is shown in Fig. 1 and its length is set to 32 bits to conform with the 32-bit interface between the processing system (PS) and the programmable logic (PL) in Xilinx Zynq MPSoC as discussed in Section IV. After the accelerator completes $N_s$ samples of the specified max-cut instance with $N_m$ nodes, the final state of the p-bit register is available as output along with several status bits.

### A. Inverse Pseudo-Temperature and Annealing Schedule

Algorithm 1 is controlled by two key hyper-parameters: the inverse pseudo-temperature $\beta$ and the number of samples

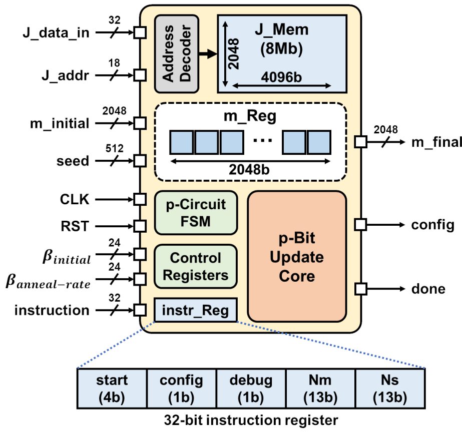


Fig. 1. Top-level architecture of the proposed pc-COP accelerator.

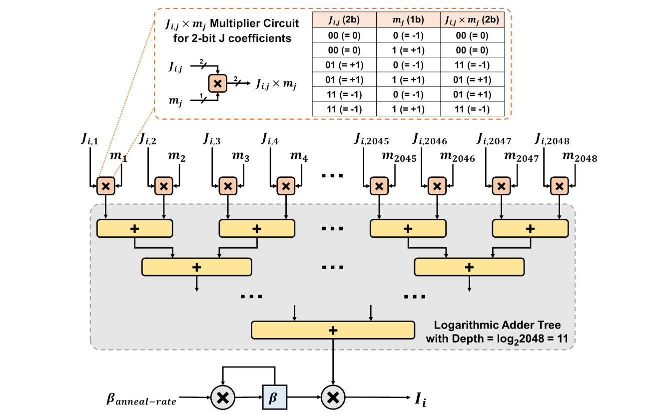


Fig. 2. Logarithmic adder tree and multiplier circuits for p-bit weight logic.

$N_s$ [27], [30]. In our implementation, the inverse pseudo-temperature for each sample (each iteration in step 3 of Algorithm 1) is obtained according to $\beta_s = \beta_{initial} \times \beta_{anneal-rate}^{s-1}$ for $1 \leq s \leq N_s$ which gives the *annealing schedule*. Based on the number of samples $N_s$, these hyper-parameters are tuned according to the analysis presented in [15], [34]. For $N_s = 1000$, we use $\beta_{initial} = 0.01$ and $\beta_{anneal-rate} = 1.005$. For $N_s = 100$, we use $\beta_{initial} = 0.01$ and $\beta_{anneal-rate} = 1.05$. Our accelerator uses a 24-bit register to store the value of $\beta_s$ with a 4-bit integer part and a 20-bit fractional part. Based on Python-based software simulation of Algorithm 1, we observed that average accuracy remains almost the same for various fractional bit precision of $\beta$ ranging from 4-bit to 32-bit. We chose the fractional bit precision of $\beta$ as 20-bit for our design as it requires the least number of DSP slices (2 DSP multipliers) for implementing the annealing schedule in FPGA.

### *B. Logarithmic Adder Tree*

A logarithmic adder tree and associated multiplier circuitry implements step 5 of Algorithm 1. For 2-bit interaction coefficients, the $J_{i,j} \times m_j$ multiplication is implemented using simple Boolean logic and the corresponding truth table is

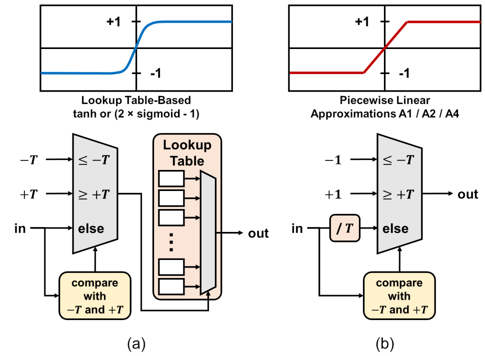


Fig. 3. Implementations of the activation function: (a) lookup table-based $tanh$ and $2 \times sigmoid - 1$ (threshold $T = 4$), and (b) piece-wise linear approximations $A_1$, $A_2$ and $A_4$ (threshold $T$ = 1, 2 and 4 respectively).

shown in Fig. 2. Using a logarithmic adder tree [33] instead of cascaded adders helps reduce the critical path delay by two orders of magnitude [35], [36]. The output of the adder tree is then multiplied with the inverse pseudo-temperature $\beta$ ($= \beta_s$ for the $s$-th sample) to get $I_i = \beta \times (\sum_{j=1}^{N_m} J_{i,j}\, m_j)$. The annealing schedule from Section III-A is implemented using another multiplier which updates the value of $\beta$ after each sample, as shown in Fig. 2.

### *C. Activation Function and Random Number Generation*

Previous work [16], [33] has implemented the $tanh$ activation function using a lookup table. In this work, we have explored various approximations of the activation function. Fig. 3 shows the circuit diagrams for lookup table and piece-wise linear approximation implementations. For lookup table, both $tanh$ and its approximation using $2 \times sigmoid - 1$ are analyzed, and the lookup tables consist of 1024 entries with 20-bit fractional precision (consistent with the discussion in Section III-A). The piece-wise linear approximations of the activation function are implemented as:

$$activation\ output \approx \begin{cases} -1 & \text{if } input \leq -T \\ input\,/\,T & \text{if } -T < input < +T \\ +1 & \text{if } input \geq +T \end{cases}$$

where division by the threshold $T \in \{1, 2, 4\}$ can be easily implemented using bit-shifts (only wiring). We observed that the piece-wise linear approximation with $T = 1$ requires the least number of FPGA LUTs ($\approx 5\times$ smaller than lookup table-based implementation) while achieving accuracy similar to the original $tanh$ activation function. Therefore, we implement this simple approximation of the activation function throughout our design. The output size of the activation function, which always lies in $[-1, +1]$, is 22-bit with 1 sign bit, 1 integer bit and 20 fractional bits.

The $rand(-1, +1)$ and $sgn(\,.\,)$ functions in step 6 of Algorithm 1 represent the p-bit stochasticity. The $rand(-1, +1)$ function is implemented using a 21-bit Fibonacci-style linear feedback shift register (LFSR) [37] with 1 sign bit and 20 fractional bits (consistent with the discussion in Section III-A).

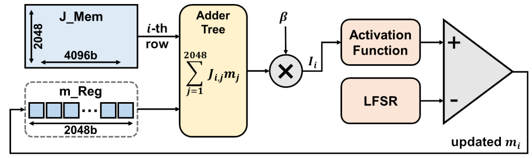


Fig. 4. Architecture of the baseline sequential p-bit update core.

The $sgn$ function is implemented using a signed comparator whose output is 0 or 1 (equivalent to -1 or +1 respectively), denoting the updated value of the p-bit in that sample.

### D. Pseudo-Parallel p-Bit Update with Speculate-and-Select

Fig. 4 shows a baseline architecture of the p-Bit Update Core which integrates the J_Mem, m_Reg, adder tree, $\beta$-multiplier, activation function, LFSR and comparator to update the p-bit state sequentially one p-bit at a time according to Algorithm 1. It takes $N_m + 1$ cycles to update all the $N_m$ p-bits in each sample, and hence requires $(N_m+1)N_s$ cycles to complete $N_s$ samples and reach the final p-bit state. However, this sequential architecture limits the overall performance of the system as it can update only one p-bit per clock cycle.

The sequential nature of the p-bit updates is inspired by Gibbs sampling which does not allow independently updating multiple p-bits in parallel. However, it is possible to speculatively compute all possible combinations of multiple updated p-bit values in parallel and then select the appropriate ones at the end. This technique resembles carry-select logic in high-performance adders [35], [36], [38]. For example, let us consider the computation of the updated value of $m_i$ as:

$$m'_i = sgn(\, rand(-1,+1) + tanh(\, \beta \times (\sum_{j=1}^{N_m} J_{i,j}m_j\,)\,)\,)$$

which has two possibilities: $m'_i = -1$ or $m'_i = +1$. Then, we can also simultaneously pre-compute the following two values:

$$\begin{aligned} m'_{i+1}|_{m'_i=-1} = sgn(\, rand(-1,+1) \,+ \\ tanh(\, \beta \times (\sum_{1\le j\le N_m, j\ne i} J_{i+1,j}m_j - J_{i+1,i}\,)\,)\,) \\ m'_{i+1}|_{m'_i=+1} = sgn(\, rand(-1,+1) \,+ \\ tanh(\, \beta \times (\sum_{1\le j\le N_m, j\ne i} J_{i+1,j}m_j + J_{i+1,i}\,)\,)\,) \end{aligned}$$

which are the two possible updated values of $m_{i+1}$ if the updated value of $m_i$ is -1 and +1 respectively. All three values $m'_i$, $m'_{i+1}|_{m'_i=-1}$ and $m'_{i+1}|_{m'_i=+1}$ are computed in parallel, and then the correct $m'_{i+1}$ is selected as:

$$m'_{i+1} = \begin{cases} m'_{i+1}|_{m'_i=-1} & \text{if } m'_i = -1 \\ m'_{i+1}|_{m'_i=+1} & \text{if } m'_i = +1 \end{cases}$$

Therefore, the p-bits can be updated two at a time for $i \in \{1, 3, 5, \cdots, 2047\}$. Note that the dependency of p-bit updates is maintained, that is, Algorithm 1 is still followed. Therefore, we refer to this technique as *pseudo-parallel update* with *speculate-and-select*. In particular, the above equations describe the *2-way pseudo-parallel architecture* where 2 p-bits are updated per cycle, that is, overall $(\frac{N_m}{2}+1)N_s$ cycles are required to complete $N_s$ samples. To enable the pseudo-parallel computation of two p-bit updates, the J_Mem is split into two banks so that both the $i$-th and $(i+1)$-th rows of matrix **J** (denoted as $J[i]$ and $J[i+1]$ respectively) can be read simultaneously in the same cycle. The overall architecture of the 2-way pseudo-parallel p-Bit Update Core is shown in Fig. 5a. Note that the speculative update of $m_{i+1}$ requires only one adder tree whose output is then adjusted according to the different speculations, e.g., $(\sum_{1\le j\le N_m, j\ne i} J_{i+1,j}m_j - J_{i+1,i}\,)$ can be calculated in one path for $m'_{i+1}|_{m'_i=-1}$, and it can be adjusted by simply adding $2J_{i+1,i}$ in the other path for $m'_{i+1}|_{m'_i=+1}$. This idea can be further extended to a *4-way pseudo-parallel architecture* where 4 p-bits are updated per cycle, that is, overall $(\frac{N_m}{4}+1)N_s$ cycles are required to com-

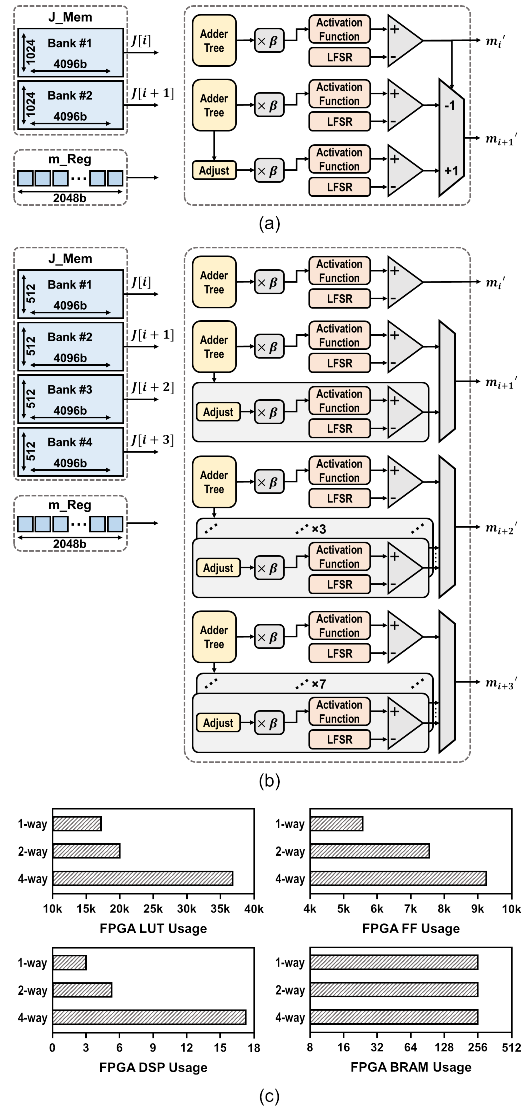


Fig. 5. Architectures of the proposed (a) 2-way and (b) 4-way pseudo-parallel p-bit update cores with speculate-and-select logic, and (c) comparison of FPGA resource utilization of 1-way, 2-way and 4-way architectures.

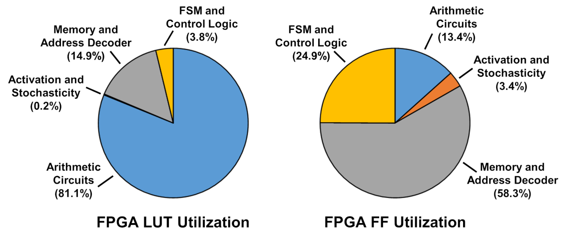


Fig. 6. Distribution of FPGA resource utilization of our proposed accelerator with 4-way pseudo-parallel p-bit update and speculate-and-select logic.

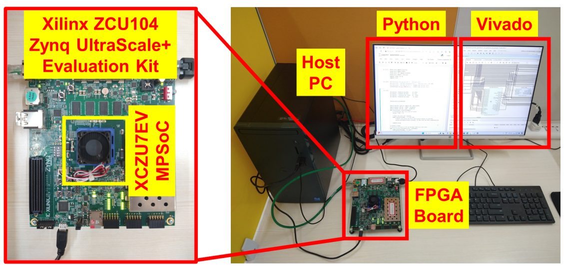


Fig. 7. Measurement setup with Zynq UltraScale+ ZCU104 FPGA board.

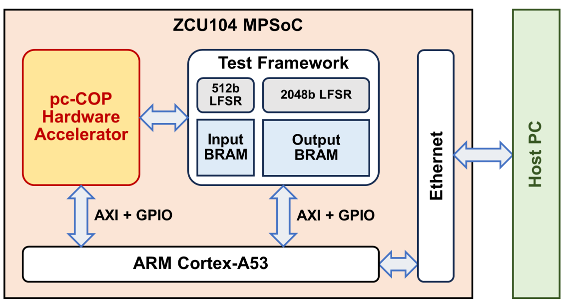


Fig. 8. Experimental validation framework consisting of proposed FPGA-based hardware accelerator interfaced with ARM processor in Zynq MPSoC.

plete $N_s$ samples. To enable the pseudo-parallel computation of four p-bit updates, the J_Mem is split into four banks so that the $i$-th, $(i+1)$-th, $(i+2)$-th and $(i+3)$-th rows of matrix $\mathbf{J}$ (denoted as $J[i]$, $J[i+1]$, $J[i+2]$ and $J[i+3]$ respectively) can be read simultaneously in the same cycle. The p-bits are updated four at a time for $i \in \{1, 5, 9, \cdots, 2045\}$. Fig. 5b shows the overall architecture of the 4-way pseudo-parallel p-Bit Update Core. Fig. 5c compares the proposed 2-way and 4-way pseudo-parallel architectures with the baseline 1-way sequential architecture in terms of FPGA resource utilization (LUTs, FFs, DSPs and BRAMs). In general, a $k$-way pseudo-parallel p-bit update architecture will require $k$ instances of the adder tree, $2^k - 1$ instances each of the $\beta$-multiplier, the activation function, the LFSR and the comparator, along with J_Mem (split into $k$ banks), m_Reg, multiplexors and control circuitry. The logic resource utilization increases exponentially with increasing $k$ and our proposed architecture can be scaled to support more pseudo-parallel updates, e.g., 8-way, 16-way, etc, based on resources available in the target FPGA.

## IV. Implementation Results

We implement and validate our proposed accelerator on a Xilinx Zynq UltraScale+ MPSoC ZCU104 Evaluation Board with an XCZU7EV-2FFVC1156E device [23] using Verilog HDL and Xilinx Vivado Design Suite version ML 2022.2. Our accelerator (with 4-way pseudo-parallel p-bit update) operates at a clock frequency of 100 MHz, and utilizes 37k LUTs, 9.5k FFs, 17 DSPs (equivalent to $\approx$ 7k LUTs [39]) and 256 BRAMs (total 8 Mb) in UltraScale+ FPGA. The distribution of FPGA resource utilization is shown in Fig. 6. Our experimental setup is shown in Fig. 7 with the FPGA board and the host PC running Python and Vivado interfaces.

Fig. 8 provides an overview of our experimental validation framework consisting of the Zynq board and the host PC connected through Ethernet. The ARM processor in the Zynq PS is used to configure the J_Mem with a problem-specific $\mathbf{J}$ matrix as well as provide the initial p-bit state, LFSR seed, annealing parameters and instruction through input ports of the accelerator implemented in the Zynq PL. A 2048-bit LFSR is used to generate the initial p-bit state, while another 512-bit LFSR is used to seed the pc-COP internal LFSRs. The ARM processor is programmed using the open-source Python-based PYNQ software framework provided by Xilinx.

We use the standard G-Set max-cut benchmark graphs [25] with 800, 1000 and 2,000 nodes to evaluate the performance and accuracy of our design with the 4-way pseudo-parallel p-bit update explained in Section III-D. We conduct 1000 trials for each G-Set benchmark for both $N_s = 1000$ and $N_s = 100$ (with the annealing hyper-parameters discussed in Section III-A) to obtain a reasonable distribution of the accuracy of results (accuracy calculated relative to best known cut values from state-of-the-art [24]). Each trial takes 2.01 ms, 2.51 ms and 5.01 ms respectively for $N_m$ = 800, 1000 and 2000 with $N_s = 1000$, and 201 $\mu$s, 251 $\mu$s and 501 $\mu$s respectively for $N_m$ = 800, 1000 and 2000 with $N_s = 100$. pc-COP achieves an average accuracy of 98.49% and 95.99% across all the 51 evaluated G-Set graphs for $N_s = 1000$ and $N_s = 100$ respectively. Table I shows the average accuracy for different graph sizes and types of graphs. We note that pc-COP is able to reach near-99% for most of the benchmarks, thus highlighting its potential for solving large-scale max-cut and other combi-

TABLE I
pc-COP Measured Performance and Accuracy

| No. of Nodes | Type of Graphs | Benchmark Graphs | Average Accuracy † | |
|---|---|---|---|---|
| | | | $N_s = 1000$ | $N_s = 100$ |
| 800 | G-Set Random | G1 - G10 | 99.30% | 97.33% |
| 800 | G-Set Toroidal | G11 - G13 | 95.65% | 86.24% |
| 800 | G-Set Planar | G14 - G21 | 98.33% | 94.46% |
| 1000 | G-Set Random | G43 - G47 | 99.63% | 98.93% |
| 1000 | G-Set Planar | G51 - G54 | 99.05% | 98.43% |
| 2000 | G-Set Random | G22 - G31 | 99.02% | 97.22% |
| 2000 | G-Set Toroidal | G32 - G34 | 95.43% | 91.12% |
| 2000 | G-Set Planar | G35 - G42 | 98.21% | 96.72% |
| 2000 | Fully Connected | K2000 | 98.89% | 97.99% |

† measured results averaged over 1000 trials

TABLE II
COMPARISON OF PC-COP WITH STATE-OF-THE-ART FPGA-BASED G-SET MAX-CUT COP HARDWARE ACCELERATORS

| Design | Type | Tech | No. of Nodes | Connection Topology | Weight Precision | Resource Utilization | Op. Freq. | G-Set Avg. Accuracy | Time to Solution |
|---|---|---|---|---|---|---|---|---|---|
| [7] | Digital Annealing | 22nm CPU | 800 - 20000 | Fully Connected | 2 bits | 32 Xeon cores + 72 GB DRAM | 2.3 GHz | 95.61% | 170 ms - 19.89 s |
| [7] | Digital Annealing | 28nm GPU | 800 - 20000 | Fully Connected | 2 bits | 2880 CUDA cores + 12 GB DRAM | 745 MHz | 95.61% | 110 ms - 390 ms |
| [40] | Coherent Ising Machine | Optics | 2000 | Fully Connected | 1 bit | – | – | 97.92% | 5 ms |
| [38] | Digital Annealing | 16nm FPGA | 1024 | Fully Connected | 4 bits | 40k LUTs + 12k FFs + 4 Mb BRAM | 100 MHz | 99.07% | 373 $\mu$s - 5.38 ms |
| [41] | Digital Annealing | 16nm FPGA | 1024 | Fully Connected | 4 bits | 75k LUTs + 12k FFs + 4 Mb BRAM | 100 MHz | 99.19% | 186 $\mu$s - 1.35 ms |
| [42] | Digital Annealing | 20nm FPGA | 4096 | Fully Connected | 2 bits | – | – | 98.50% | 5 ms - 25 ms |
| [9] | Parallel Tempering | 16nm FPGA | 1024 (×8 replicas) | Fully Connected | 2 bits | 99k LUTs + 74k FFs + 7.125 Mb BRAM | 200 MHz | 99.43% | 0.5 ms - 1 ms |
| [13] | Probabilistic Computing | 14nm CPU | 800 - 3000 | Fully Connected | 2 bits | 2 Core-i7 cores | 2.5 GHz | ≈ 97.00% | – |
| **This Work** | **Probabilistic Computing** | **16nm FPGA** | **2048** | **Fully Connected** | **2 bits** | **37k LUTs + 9.5k FFs + 17 DSPs (≈ 7k LUTs †) + 8 Mb BRAM** | **100 MHz** | **for $N_s = 1000$** | |
| | | | | | | | | **98.49%** | **2.01 ms - 5.01 ms** |
| | | | | | | | | **for $N_s = 100$** | |
| | | | | | | | | **95.99%** | **201 $\mu$s - 501 $\mu$s** |

† 1 DSP is equivalent to ≈ 51.2 logic slices [39] and 1 logic slice contains 8 LUTs in UltraScale+ FPGA [23]

natorial optimization problems. Fig. 9 shows the evolution of the system energy and convergence towards the best known cut size of 11624 (with final energy $E\{m\} = -4050$) for the G1 benchmark, as measured from our experimental setup. We also evaluate max-cut performance with the K2000 benchmark [15] which is a fully-connected graph where all nodes are connected to each other with $\{0, \pm 1\}$ weights. Across 1000 trials, we achieve average accuracy of 98.89% and 97.99% for $N_s = 1000$ and $N_s = 100$ respectively.

Table II compares our design with previous work on FPGA-based hardware accelerators demonstrating max-cut with G-Set benchmarks. Most of the previous work are digital annealers and Ising computers implemented using CPU and GPU [7], optics [40] and FPGA [38], [41], [42], [9]. While there are many other implementations of FPGA-based and ASIC-based digital annealers in recent literature [43], [44], [45], we only include those which have demonstrated G-Set benchmarks for fair comparison. [13] is a CPU-based demonstration of G-Set max-cut with probabilistic computing. Compared to previous

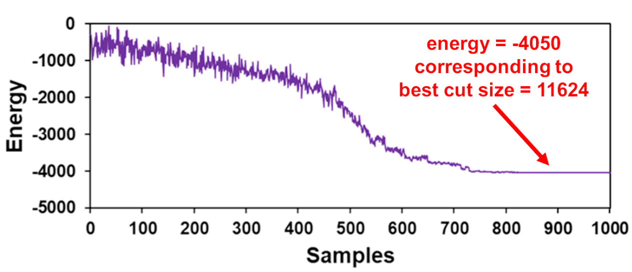


Fig. 9. Evolution of system energy and convergence towards solution for G1 benchmark with $N_s = 1000$, as measured from our experimental setup.

CPU-based and GPU-based implementations, we achieve 3 orders of magnitude speedup while maintaining similar accuracy levels. Compared to previous FPGA-based digital annealer implementations, we achieve reasonably comparable performance and accuracy with the new probabilistic computing paradigm while having lower FPGA resource utilization. This clearly demonstrates that hardware-accelerated probabilistic computing is an excellent candidate for realizing efficient and large-scale combinatorial optimization problem solvers.

## V. CONCLUSION

Probabilistic computing is an emerging quantum-inspired computing paradigm capable of solving various classes of computationally hard problems such as combinatorial optimization. In this work, we present pc-COP, an efficient and configurable probabilistic computing hardware accelerator with 2048 fully connected p-bits implemented on Xilinx UltraScale+ FPGA and demonstrate the standard G-Set graph maximum cut benchmarks. Our efficient logarithmic adder tree design for sum-of-products computation reduces critical path delay. We efficiently approximate the activation function and tune the precision of the annealing schedule to save logic resources. Finally, we propose a pseudo-parallel p-bit update architecture with speculate-and-select logic which improves overall performance by $4\times$ compared to the traditional sequential p-bit update. We achieve near-99% average accuracy across various G-Set max-cut benchmarks with 800, 1000 and 2000 nodes. Our experimental results demonstrate that FPGA-based probabilistic computing hardware accelerators are promising practical systems for efficiently solving large-scale combinatorial optimization problems.

## Acknowledgment

This work was supported in part by a seed grant from the Indian Institute of Science and in part by a Ph.D. Scholarship from the Ministry of Education, Government of India. The authors would like to thank Yashash Jain for helpful technical discussions and Dr. Shantharam Kalipatnapu for helping set up the PYNQ interface.

## References

[1] R. P. Feynman, "Simulating Physics with Computers," *International Journal of Theoretical Physics*, vol. 21, no. 6/7, 1982.

[2] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[3] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM Journal of Computing*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997.

[4] F. Arute *et al.*, "Quantum Supremacy using a Programmable Superconducting Processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[5] K. Y. Camsari, R. Faria, B. M. Sutton, and S. Datta, "Stochastic $p$-Bits for Invertible Logic," *Phys. Rev. X*, vol. 7, p. 031014, Jul. 2017.

[6] S. C. Smithson *et al.*, "Efficient CMOS Invertible Logic Using Stochastic Computing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 6, pp. 2263–2274, Jan. 2019.

[7] C. Cook *et al.*, "GPU-Based Ising Computing for Solving Max-Cut Combinatorial Optimization Problems," *Integration*, vol. 69, pp. 335–344, 2019.

[8] H. Jung *et al.*, "A Quantum-Inspired Probabilistic Prime Factorization based on Virtually Connected Boltzmann Machine and Probabilistic Annealing," *Nature Scientific Reports*, vol. 13, no. 1, p. 16186, 2023.

[9] Y. Zhang *et al.*, "A Parallel Tempering Processing Architecture with Multi-Spin Update for Fully-Connected Ising Models," in *IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2024, pp. 1–6.

[10] K. Y. Camsari and S. Datta, "Dialogue Concerning the Two Chief Computing Systems: Imagine Yourself on a Flight Talking to an Engineer About a Scheme that Straddles Classical and Quantum," *IEEE Spectrum*, vol. 58, no. 4, pp. 30–35, Apr. 2021.

[11] S. Chowdhury *et al.*, "A Full-Stack View of Probabilistic Computing with p-Bits: Devices, Architectures and Algorithms," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 9, no. 1, pp. 1–11, Mar. 2023.

[12] A. Z. Pervaiz *et al.*, "Hardware Emulation of Stochastic p-Bits for Invertible Logic," *Nature Scientific Reports*, vol. 7, no. 10994, Sep. 2017.

[13] M. Khan and O. Hassan, "Benchmarking of Probabilistic-Bit Based Algorithm for Max-Cut Problem," in *International Conference on Electrical and Computer Engineering (ICECE)*. IEEE, 2022, pp. 453–456.

[14] N. Onizawa *et al.*, "Fast-Converging Simulated Annealing for Ising Models Based on Integral Stochastic Computing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 12, pp. 10 999–11 005, Dec. 2023.

[15] N. Onizawa and T. Hanyu, "Enhanced Convergence in p-Bit Based Simulated Annealing with Partial Deactivation for Large-Scale Combinatorial Optimization Problems," *Nature Scientific Reports*, vol. 14, no. 1, p. 1339, 2024.

[16] A. Z. Pervaiz *et al.*, "Weighted $p$-Bits for FPGA Implementation of Probabilistic Circuits," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 6, pp. 1920–1926, Oct. 2019.

[17] W. A. Borders, A. Z. Pervaiz, S. Fukami, K. Y. Camsari, H. Ohno, and S. Datta, "Integer Factorization using Stochastic Magnetic Tunnel Junctions," *Nature*, vol. 573, no. 7774, pp. 390–393, Sep. 2019.

[18] Y. Liu *et al.*, "Probabilistic Circuit Implementation Based on p-Bits using the Intrinsic Random Property of RRAM and p-Bit Multiplexing Strategy," *Micromachines*, vol. 13, no. 6, p. 924, Jun. 2022.

[19] S. Luo, Y. He, B. Cai, X. Gong, and G. Liang, "Probabilistic-Bits Based on Ferroelectric Field-Effect Transistors for Probabilistic Computing," *IEEE Electron Device Letters*, vol. 44, no. 8, pp. 1356–1359, 2023.

[20] S. Heo *et al.*, "Experimental Demonstration of Probabilistic-Bit (p-bit) Utilizing Stochastic Oscillation of Threshold Switch Device," in *IEEE Symposium on VLSI Technology and Circuits (VLSI)*, 2023, pp. 1–2.

[21] A. Lucas, "Ising Formulations of Many NP Problems," *Frontiers in Physics*, vol. 2, p. 74887, 2014.

[22] N. Mohseni, P. L. McMahon, and T. Byrnes, "Ising Machines as Hardware Solvers of Combinatorial Optimization Problems," *Nature Reviews Physics*, vol. 4, no. 6, pp. 363–379, 2022.

[23] Xilinx Inc., "UltraScale Architecture: Staying a Generation Ahead with an Extra Node of Value," https://www.xilinx.com/products/technology/ultrascale.html.

[24] Y. Matsuda, "Benchmarking the MAX-CUT Problem on the Simulated Bifurcation Machine," Medium, 2019, https://medium.com/toshiba-sbm/benchmarking-the-max-cut-problem-on-the-simulated-bifurcation-machine-e26e1127c0b0.

[25] Stanford University, "G-Set Graph Dataset," https://web.stanford.edu/%7Eyyye/yyye/Gset.

[26] K. Y. Camsari, B. M. Sutton, and S. Datta, "$p$-Bits for Probabilistic Spin Logic," *Applied Physics Reviews*, vol. 6, no. 1, Mar. 2019.

[27] B. Sutton, R. Faria, L. A. Ghantasala, R. Jaiswal, K. Y. Camsari, and S. Datta, "Autonomous Probabilistic Coprocessing With Petaflips per Second," *IEEE Access*, vol. 8, pp. 157 238–157 252, Aug. 2020.

[28] N. A. Aadit *et al.*, "Computing with Invertible Logic: Combinatorial Optimization with Probabilistic Bits," in *IEEE International Electron Devices Meeting (IEDM)*, Dec. 2021, pp. 40–43.

[29] J. Kaiser and S. Datta, "Probabilistic Computing with p-Bits," *Applied Physics Letters*, vol. 119, no. 15, Oct. 2021.

[30] N. A. Aadit *et al.*, "Massively Parallel Probabilistic Computing with Sparse Ising Machines," *Nature Electronics*, vol. 5, no. 7, pp. 460–468, Jul. 2022.

[31] A. Grimaldi *et al.*, "Experimental Evaluation of Simulated Quantum Annealing with MTJ-Augmented p-bits," in *International Electron Devices Meeting (IEDM)*, 2022, pp. 539–542.

[32] N. A. Aadit, M. Mohseni, and K. Y. Camsari, "Accelerating Adaptive Parallel Tempering with FPGA-based p-bits," in *IEEE Symposium on VLSI Technology and Circuits (VLSI)*, 2023, pp. 1–2.

[33] Y. Jain and U. Banerjee, "Tyche: A Compact and Configurable Accelerator for Scalable Probabilistic Computing on FPGA," in *IEEE High Performance Extreme Computing Conference (HPEC)*, 2023, pp. 1–7.

[34] N. Onizawa, K. Kuroki, D. Shin, and T. Hanyu, "Local Energy Distribution Based Hyperparameter Determination for Stochastic Simulated Annealing," *IEEE Open Journal of Signal Processing*, vol. 4, pp. 452–461, 2023.

[35] J. M. Rabaey, A. P. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits: A Design Perspective*. Prentice-Hall, 2002.

[36] N. H. E. Weste and D. M. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*. Addison-Wesley, 2011.

[37] R. Ward and T. Molteno, "Table of Linear Feedback Shift Registers," Department of Physics, University of Otago, Tech. Rep., Oct. 2007.

[38] Z. Huang, D. Jiang, X. Wang, and E. Yao, "An Ising Model-Based Annealing Processor With 1024 Fully Connected Spins for Combinatorial Optimization Problems," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 8, pp. 3074–3078, 2023.

[39] Y. Zhang *et al.*, "Ultra High-Speed Polynomial Multiplications for Lattice-Based Cryptography on FPGAs," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 4, pp. 1993–2005, 2022.

[40] T. Inagaki *et al.*, "A Coherent Ising Machine for 2000-Node Optimization Problems," *Science*, vol. 354, no. 6312, pp. 603–606, 2016.

[41] Z. Huang *et al.*, "An Annealing Processor based on 1k-Spin Fully-Connected Ising Model for Combinatorial Optimization Problems," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2023, pp. 1–5.

[42] H. M. Waidyasooriya and M. Hariyama, "Temporal and Spatial Parallel Processing of Simulated Quantum Annealing on a Multicore CPU," *The Journal of Supercomputing*, pp. 1–18, 2022.

[43] C. Yoshimura *et al.*, "CMOS Annealing Machine: A Domain-Specific Architecture for Combinatorial Optimization Problem," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020, pp. 673–678.

[44] S. Xie *et al.*, "Ising-CIM: A Reconfigurable and Scalable Compute Within Memory Analog Ising Accelerator for Solving Combinatorial Optimization Problems," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 11, pp. 3453–3465, 2022.

[45] J. Bae *et al.*, "CTLE-Ising: A Continuous-Time Latch-Based Ising Machine Featuring One-Shot Fully Parallel Spin Updates and Equalization of Spin States," *IEEE Journal of Solid-State Circuits*, vol. 59, no. 1, pp. 173–183, 2024.