Tyche: A Compact and Configurable Accelerator for Scalable Probabilistic Computing on FPGA

Yashash Jain and Utsav Banerjee
Electronic Systems Engineering
Indian Institute of Science, Bengaluru, India
Email: jainyashash24@gmail.com, utsav@iisc.ac.in

***Abstract*—Probabilistic computing is an emerging computing paradigm which involves the systematic control and manipulation of unstable stochastic units called p-bits. Multiple p-bits are connected together to implement p-circuits which have been shown to be capable of solving interesting computationally hard problems. In this work, we present Tyche, a compact and configurable hardware accelerator for scalable probabilistic computing on FPGA. Our architecture allows p-circuits requiring different number of p-bits to be implemented using the same hardware. The use of a single p-bit computing core instead of an array of processing elements provides significant logic resource savings. A logarithmic adder tree is used for single-cycle weight logic computation while ensuring reasonable performance even for large number of p-bits. Various application-specific p-circuits are experimentally demonstrated using our proposed hardware accelerator implemented on Xilinx UltraScale+ FPGA, thus emphasizing the viability of practical scalable probabilistic computing on modern FPGAs.**

***Index Terms*—FPGA, probabilistic computing, configurable accelerator, binary stochastic neuron, p-bit, p-circuit, invertible logic, Boltzmann machine, Ising machine, integer factorization.**

I. Introduction

The rise of quantum computing technologies has not only opened up a plethora of new opportunities in cryptography, machine learning, bio-informatics, mathematical optimization and other applications [1], [2], [3] but also led to the emergence of several new non-conventional computing paradigms such as stochastic computing [4], probabilistic computing [5] and probabilistic annealing [6]. Probabilistic computing involves the systematic control and manipulation of unstable stochastic units, also known as probabilistic bits or p-bits, interconnected together, also known as probabilistic circuits or p-circuits. The behaviour of p-bits and p-circuits resembles networks of binary stochastic neurons similar to Boltzmann machines and Ising machines [5], [7]. Probabilistic computing has been shown to be capable of solving interesting computationally hard problems such as combinatorial optimization, quantum emulation, Bayesian inference, integer factorization and invertible logic [5], [8], [7], [9], [10].

Fig. 1 summarizes the key differences between the three computing paradigms - digital (classical), probabilistic and quantum, and an immediate observation is that probabilistic computing lies somewhere in the middle of digital and quantum [11]. Digital computing involves the manipulation of bits which are deterministically either 0 or 1. Quantum computing involves the manipulation of qubits which can exist in a superposition of 0 and 1, a phenomenon unique to quantum physics. In contrast, p-bits in probabilistic computing fluctuate rapidly between 0 and 1. Unlike qubits which require near-absolute-zero temperatures for correct functionality, it is possible to realize p-bits and p-circuits at room temperature thus providing a methodology for practical quantum emulation. Although probabilistic computers are not expected to be a replacement for quantum computers, they enable various interesting applications using existing hardware technologies at the intersection of classical and quantum computing [12].

Previous literature has demonstrated the potential of probabilistic computing by implementing p-bits and p-circuits using a variety of technologies such as magnetic tunnel junctions [13], resistive random access memories [14], micro-controllers [15] and FPGAs [16], [7]. The FPGA-based implementations were fixed p-circuit demonstrations of applications such as invertible logic, optimization and integer factorization using limited number of p-bits. In this work, we propose an efficient and configurable architecture for scalable probabilistic computing on FPGA. We present our hardware accelerator Tyche and its experimental demonstration using Xilinx UltraScale+ FPGA. Our configurable architecture allows p-circuits requiring different number of p-bits to be evaluated using the same hardware. We implement a single p-bit computing core instead of an array of processing elements to achieve significant logic resource savings. We use a logarithmic adder tree for single-cycle arithmetic computation to ensure reasonable performance even for large number of p-bits. Using our configurable accelerator, we implement application-specific p-circuits up to 52 p-bits for invertible logic, max cut, travelling salesman and integer factorization, thus demonstrating practical scalable probabilistic computing on a modern FPGA platform.

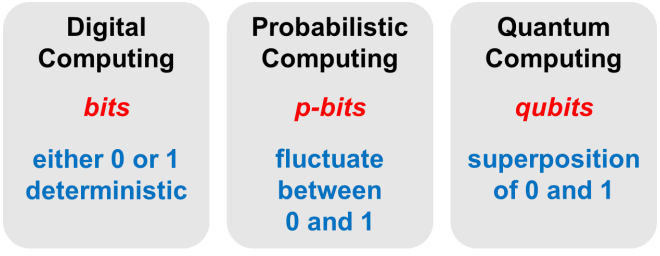

Fig. 1. Three computing paradigms: digital, probabilistic and quantum.

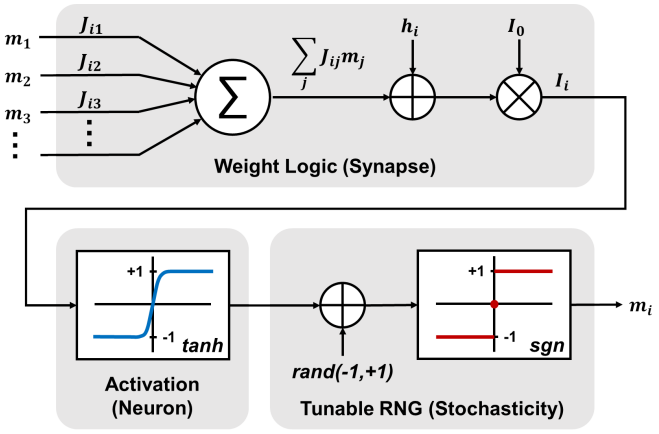

Fig. 2. Overview of p-bit operation as a binary stochastic neuron.

II. Background

Probabilistic computing, also known as probabilistic spin logic, is an emerging computing paradigm which involves the systematic control and manipulation of unstable stochastic units called probabilistic bits or *p-bits*. Multiple p-bits are connected together to implement probabilistic circuits or *p-circuits* which have been shown to be capable of solving interesting computationally hard problems such as combinatorial optimization, quantum emulation, Bayesian inference, integer factorization and various Boolean functions represented as invertible logic [5], [8], [7], [9], [10].

The operation of a p-bit can be represented as a *binary stochastic neuron*, as shown in Fig. 2. For a p-circuit with N_m p-bits $m_i \in \{-1, +1\} \forall 1 \leq i \leq N_m$, each p-bit is updated sequentially according to the following equation:

$$m_i = sgn(\, rand(-1, +1) + tanh(\, I_i \,) \,)$$

where $I_i = I_0 \times (\, h_i + \sum_{j=1}^{N_m} J_{ij}\, m_j \,)$ for $1 \leq i \leq N_m$. The synaptic weights J_{ij} which determine the interconnections between different p-bits are obtained from an $N_m \times N_m$ matrix **J**. The bias values h_i which determine the local contributions for different p-bits are obtained from an $N_m \times 1$ column vector **h**. I_0 is a constant which can be used to control the strength of the p-bit interconnections. The above equations resemble the behaviour of Boltzmann machines [5] and the stochastic neural network representation is similar to Ising machines [7].

Recent literature has explored the use of emerging technologies such as magnetic tunnel junction and non-volatile memory devices to efficiently realize the binary stochastic neuron functionality in hardware [5], [8], [13], [7], [17], [14], [12]. However, large-scale implementations of p-circuits using such emerging devices are yet to be experimentally demonstrated. Therefore, FPGA-based implementations provide a promising near-term alternative [16], [7], [4]. In this work, we present an FPGA-based compact hardware accelerator for scalable probabilistic computing which can be configured to implement various application-specific p-circuits with different number of p-bits. Our hardware architecture is described in Section III and implementation results are discussed in Section IV.

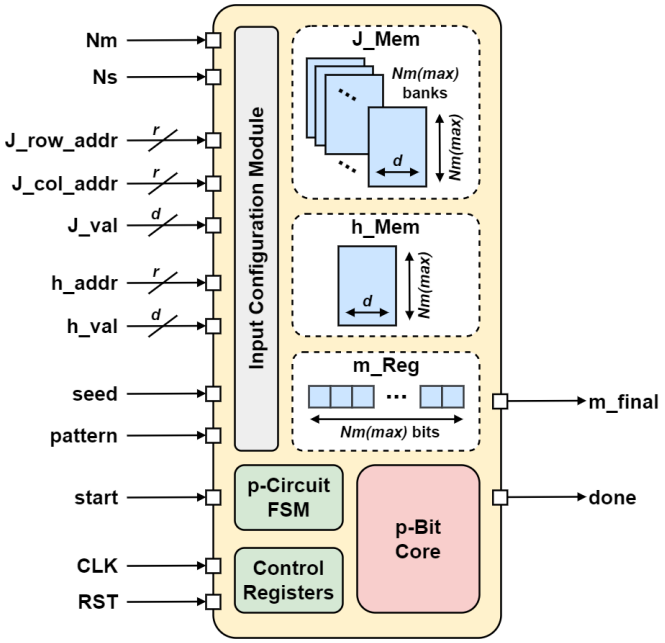

Fig. 3. Top-level architecture of the proposed compact, configurable and scalable probabilistic computing hardware accelerator.

III. Accelerator Architecture

In this work, we design, implement and demonstrate a compact and configurable hardware accelerator for scalable probabilistic computing on Xilinx UltraScale+ FPGA [18]. Fig. 3 shows the top-level architecture of Tyche, our proposed hardware accelerator. The architecture is designed to support p-circuits comprised of maximum $N_{m(max)}$ p-bits, where $N_{m(max)}$ is a design parameter which can be set prior to FPGA implementation. After implementation, the accelerator can be easily configured to implement p-circuits with $N_m \leq N_{m(max)}$ p-bits by providing N_m as an external input. In order to support up to $N_{m(max)}$ p-bits, the accelerator is required to store the $N_{m(max)} \times N_{m(max)}$ matrix **J** and the $N_{m(max)} \times 1$ column vector **h**, where each element of **J** and **h** is a signed fixed-point d-bit value. As shown in Fig. 3, the **J** matrix memory J_Mem is organized in the form of $N_{m(max)}$ banks of single-port RAMs with each bank having $N_{m(max)}$ words of d-bit each. The i-th word in the j-th bank $(1 \leq i, j \leq N_{m(max)})$ stores the (i, j)-th element of **J**, that is, the element in the j-th column of the i-th row. As shown in Fig. 3, the **h** vector memory h_Mem is organized as one single-port RAM having $N_{m(max)}$ words of d-bit each. The i-th word $(1 \leq i \leq N_{m(max)})$ stores the i-th element of **h**, that is, the element in the i-th row. This memory organization ensures that all $N_{m(max)} + 1$ elements required for the weight logic computation $h_i + \sum_{j=1}^{N_{m(max)}} J_{ij}\, m_j$ corresponding to the i-th p-bit update can be read in parallel in a single clock cycle, as discussed later. The **J** and **h** values are written to these memories using a d-bit data input and a pair of r-bit address inputs (for row and column) for J_Mem and a d-bit data input and an r-bit address input for h_Mem, where $r =$

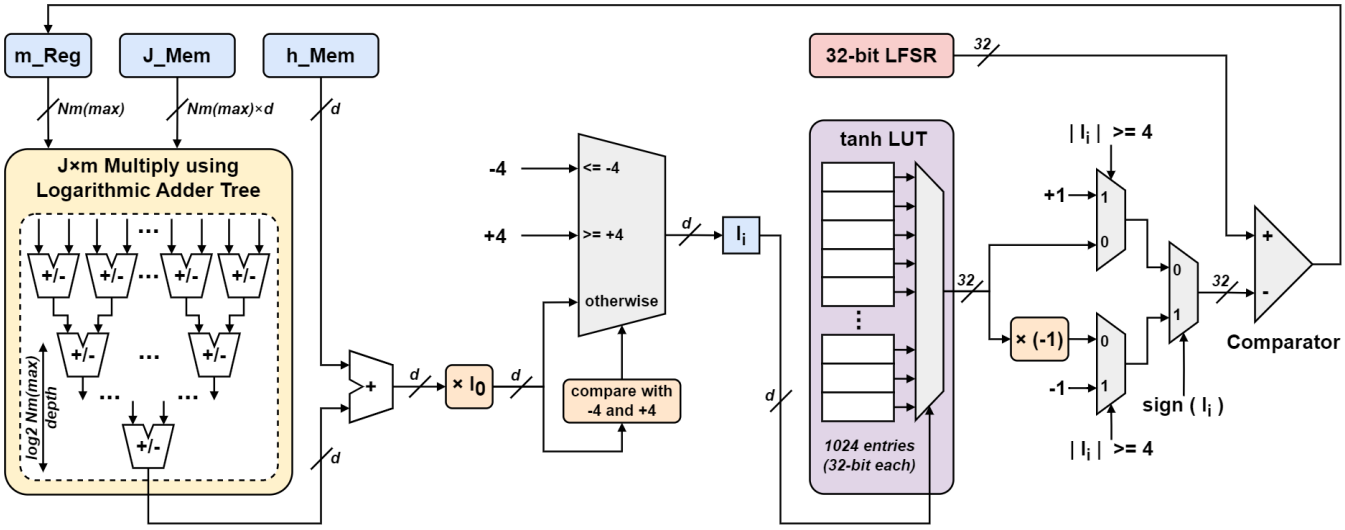

Fig. 4. Architecture of the p-bit core in the proposed probabilistic computing hardware accelerator.

$\lceil \log_2 N_{m(max)} \rceil$. The input configuration module contains address decoder logic to translate these r-bit external addresses into appropriate internal addresses of the single-port RAMs. It also contains control circuitry to enable the appropriate memory banks and ensure that the inputs are written to the correct memory locations in J_Mem and h_Mem.

The p-bit values are stored in an $N_{m(max)}$-bit register m_Reg with a -1 p-bit value stored as 0 and a +1 p-bit value stored as 1. Unlike previous work which has implemented arrays of p-bit processing elements [16], [7], our design contains only a single p-bit core as shown in Fig. 3. This is based on the observation that most p-circuits require sequential update of p-bits for correct functionality [5], [8], that is, multiple p-bits are not updated simultaneously and having an array of p-bit cores would mean that only one core would be active at a time and others would be idle. Therefore, our proposed architecture is able to support a large set of application-specific p-circuits while reducing the computation logic to $\approx 1/N_{m(max)}$-th.

Our p-bit core architecture is shown in Fig. 4. Since $m_j \in \{-1, +1\} \forall 1 \leq j \leq N_{m(max)}$, the sum of products $\sum_{j=1}^{N_{m(max)}} J_{ij}\, m_j$ can be computed simply using additions and subtractions instead of multiplications. If $m_j = +1$, then the corresponding J_{ij} is added. If $m_j = -1$, then the corresponding J_{ij} is subtracted. A set of 2's complement adder-subtractor modules are used for this computation. The straight-forward implementation would require $N_{m(max)} - 1$ such adder-subtractor modules in cascade to process the $N_{m(max)}$ J_{ij} inputs. However, the cascade architecture leads to longer critical paths which adversely affect the overall system performance for larger p-circuits. Therefore, we implement the adder-subtractor modules in a logarithmic tree architecture [19], [20] as shown in Fig. 4. The logarithmic tree requires the same number of adder-subtractor modules but reduces the circuit depth, thus improving performance. If the propagation delay for each adder-subtractor module is t_{ADD}, then the overall propagation delay for the logarithmic tree is $\lceil \log_2 N_{m(max)} \rceil\, t_{ADD}$ as opposed to $(N_{m(max)} - 1)\, t_{ADD}$ for the cascade architecture. The variation in propagation delay of the logarithmic tree versus the cascade architecture with increasing $N_{m(max)}$ is shown in Fig. 5, where each adder-subtractor module is implemented as an optimized 24-bit ripple carry circuit in Xilinx UltraScale+ FPGA [18].

All arithmetic with **J** and **h** is performed with d = 24-bit signed fixed-point values, where the most significant bit accounts for the sign, the next 11 bits for the integral part and the last 12 bits for the fractional part. Although previous work [15], [16], [4], [14] has used only integer values with smaller d for **J** and **h**, our simulations have shown that the accuracy of the p-circuit improves significantly with larger d and with fixed-point values instead of integers.

The $I_i = I_0 \times (h_i + \sum_{j=1}^{N_{m(max)}} J_{ij}\, m_j)$ value is computed using another adder followed by a fixed-constant multiplier ($I_0 = 1$ for the p-circuits demonstrated in this work). It is then restricted to the range $[-4, +4]$ using a comparator and a multiplexer. A 1024×32-bit lookup table for the $tanh$

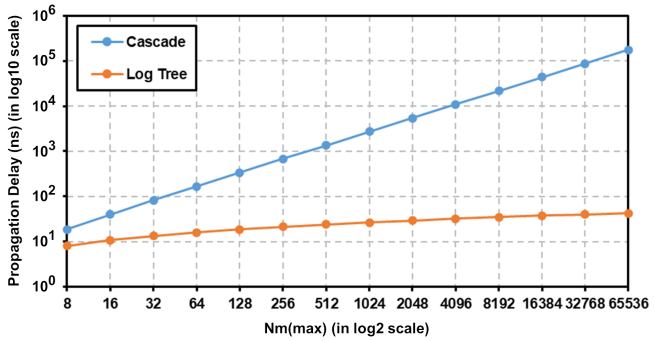

Fig. 5. Scaling of adder propagation delay for logarithmic tree and cascade architectures with maximum number of p-bits $N_{m(max)}$.

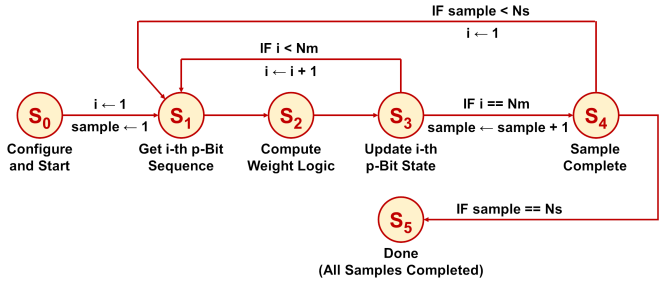

Fig. 6. Top-level state machine of the proposed p-circuit hardware accelerator.

activation function followed by a thresholding circuit (using a set of multiplexers) is used to compute $tanh(I_i)$ as a 32-bit value. A 32-bit linear feedback shift register (LFSR) [21], [16] is used the generate the p-bit stochasticity, where the LFSR seed is configured through external input. Finally, a 32-bit comparator is used to compute the updated p-bit value $m_i = sgn(\, rand(-1, +1) + tanh(\, I_i \,)\,)$ and written back to the i-th bit in m_Reg.

Each p-bit update takes 3 clock cycles and a complete update of all $N_{m(max)}$ p-bits requires $3N_{m(max)}$ cycles, also known as a sample. The accelerator repeats this for N_s samples, where N_s is configured as external input. The sequence of p-bit update in consecutive samples must be different in order to prevent correlation between consecutive states leading to incorrect output [5], [8]. This is ensured by using an $N_{m(max)} \lceil \log_2 N_{m(max)} \rceil$-bit sequence register (as part of the control registers in Fig. 3) which generates a new update sequence for each sample and the update pattern is configured using external input. The accelerator operation is controlled by a finite state machine (FSM) circuit, as shown in Fig. 3, which implements the six main states shown in Fig. 6. Overall, our proposed configurable hardware accelerator takes $3 \times N_m \times N_s$ clock cycles to process N_s samples of a p-circuit with $N_m \leq N_{m(max)}$ p-bits.

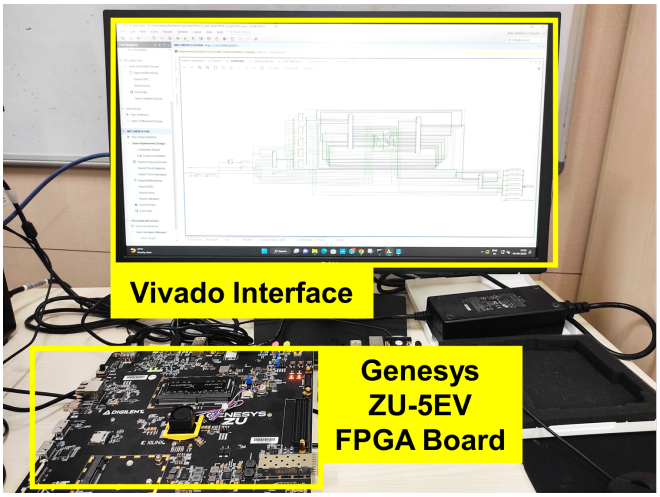

Fig. 7. Experimental validation setup with Genesys ZU-5EV FPGA board.

IV. Implementation Results

A. FPGA Implementation

We implement our configurable and scalable probabilistic computing accelerator design on Xilinx UltraScale+ FPGA [18]. The accelerator is designed using Verilog HDL (Hardware Description Language) and the Xilinx Vivado Design Suite version ML 2022.2 is used for FPGA synthesis, implementation and simulation. We experimentally validate our implementation on a Digilent Genesys ZU-5EV Zynq UltraScale+ MPSoC Development Board [22] with an XCZU5EV-SFVC784-1-E device, and our setup is shown in Fig. 7. It contains 117k look-up tables (LUTs) and 234k flip-flops (FFs) in configurable logic blocks (CLBs), 144 block RAMs (BRAMs) and 1248 digital signal processing (DSP) slices. Fig. 8 shows how the maximum frequency of the configurable p-circuit accelerator scales with the maximum number of p-bits $N_{m(max)}$. As expected with our logarithmic adder tree design discussed in Section III, the maximum frequency decreases approximately linearly (that is, the critical path delay increases approximately linearly) with $\lceil \log_2 N_{m(max)} \rceil$. Figs. 9, 10 and 11 summarize the post-implementation FPGA resource utilization for configurable p-circuit designs with $N_{m(max)} \in \{8, 16, 32, 64, 128, 256\}$ and $d = 24$. As expected, resource utilization (LUTs, FFs and BRAMs) increases with larger $N_{m(max)}$. However, it is the number of BRAMs, determined by the size of **J** and **h**, which limits the maximum number of p-bits that can be supported in a particular device. The Zynq UltraScale+ device in Genesys ZU-5EV has 144 BRAMs, therefore allowing p-circuit implementation up to $\approx$ 286 p-bits. As the number of available BRAMs is increased, the maximum number of supported p-bits $N_{m(max)}$ can also be increased, that is, the p-circuit can be scaled as discussed later. Note that the design does not require any DSP slices since the adder tree is implemented exclusively using LUTs and fast addition carry chains inside CLBs. Next, we discuss the different probabilistic computing applications (up to 52 p-bits) experimentally validated using our FPGA setup.

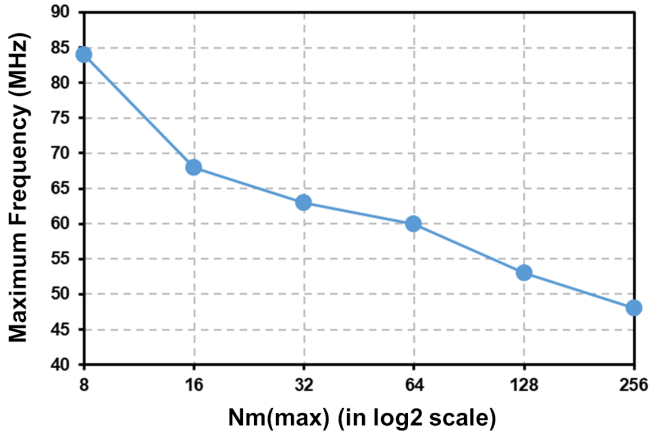

Fig. 8. Scaling of maximum frequency of accelerator implemented on Zynq UltraScale+ MPSoC with maximum number of p-bits.

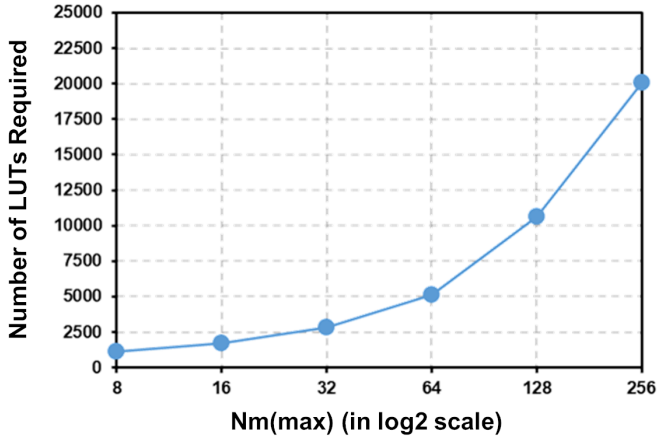

Fig. 9. Scaling of number of LUTs required by accelerator implemented on Zynq UltraScale+ MPSoC with maximum number of p-bits.

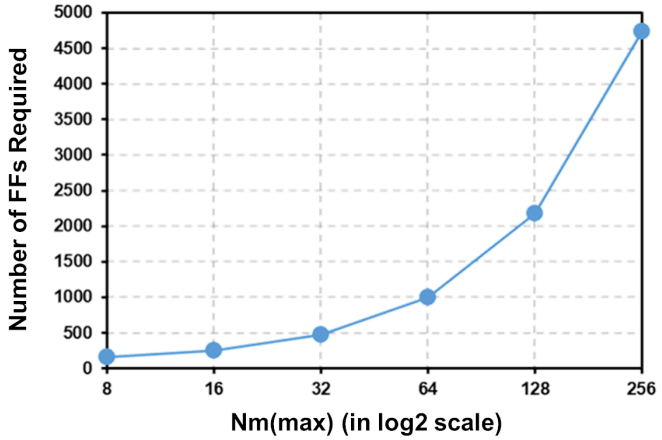

Fig. 10. Scaling of number of FFs required by accelerator implemented on Zynq UltraScale+ MPSoC with maximum number of p-bits.

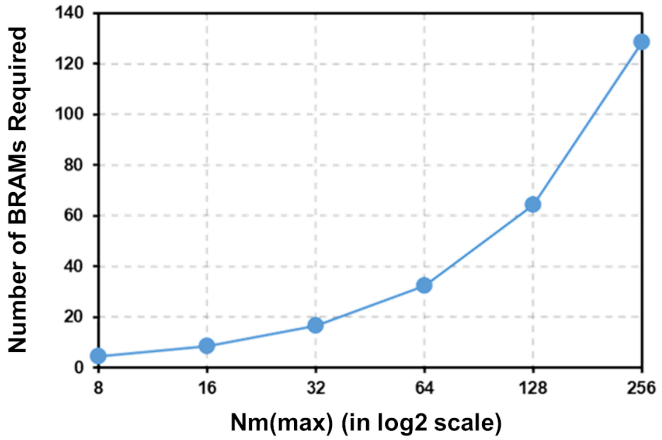

Fig. 11. Scaling of number of BRAMs required by accelerator implemented on Zynq UltraScale+ MPSoC with maximum number of p-bits.

TABLE I
P-BIT REQUIREMENTS OF VARIOUS P-CIRCUITS

Application-Specific p-Circuit	**Weighted p-Bits (N_m)**
Tunable RNG	1
NOT Gate	2
AND Gate	3
1-Bit Full Adder	5
6-Node Max-Cut Problem	6
4-City Travelling Salesman Problem	16
8-Bit Integer Factorization	52

B. Application Demonstration

Table I shows different application-specific p-circuits experimentally validated using our FPGA implementation along with the required number of weighted p-bits (N_m). Note that these p-circuits can be easily implemented using our configurable design (supporting up to $N_{m(max)}$ p-bits) by using appropriate input N_m as long as $N_m \leq N_{m(max)}$. The values of $\mathbf{J}$ and $\mathbf{h}$ for specific p-circuits can be calculated from their truth tables using optimization techniques such as gradient descent [23] or linear programming [24]. It is important to note that the total number of p-bits (including ancillae) required for an application-specific p-circuit must be known to calculate the corresponding $\mathbf{J}$ and $\mathbf{h}$. Furthermore, the optimization procedure becomes exponentially more computationally expensive with increasing number of p-bits.

The 1-p-bit tunable random number generator (RNG) sets $\mathbf{J} = 0$ and $\mathbf{h} = 0$ so that the output state is either 0 or 1 with equal probability. The $\mathbf{J}$ and $\mathbf{h}$ values for 2-p-bit NOT gate, 3-p-bit AND gate and 5-p-bit full adder (FA) are as follows:

$$\mathbf{J}_{NOT} = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} \ , \ \mathbf{h}_{NOT} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\mathbf{J}_{AND} = \begin{pmatrix} 0 & -1 & 2 \\ -1 & 0 & 2 \\ 2 & 2 & 0 \end{pmatrix} \ , \ \mathbf{h}_{AND} = \begin{pmatrix} 1 \\ 1 \\ -2 \end{pmatrix}$$

$$\mathbf{J}_{FA} = \begin{pmatrix} 0 & -1 & -1 & 1 & 2 \\ -1 & 0 & -1 & 1 & 2 \\ -1 & -1 & 0 & 1 & 2 \\ 1 & 1 & 1 & 0 & -2 \\ 2 & 2 & 2 & -2 & 0 \end{pmatrix} \ , \ \mathbf{h}_{FA} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Both the max-cut problem and the travelling salesman problem involve graph computations with the $\mathbf{J}$ and $\mathbf{h}$ values determined by the number of vertices in the graph and the weights of their edges, as obtained from [25]. Solving the max-cut problem with n nodes requires n p-bits and solving the travelling salesman problem (TSP) with n cities requires n^2 p-bits. The 8-bit integer factorization is performed using the p-circuit for 4-bit $\times$ 4-bit integer multiplication in reverse with the output clamped to the 8-bit integer to be factorized. This requires 52 p-bits, where the first 4 p-bits are for the first factor, the next 4 p-bits are for the second factor, the last

TABLE II
COMPARISON WITH STATE-OF-THE-ART INVERTIBLE LOGIC IMPLEMENTATIONS ON KINTEX ULTRASCALE FPGA

Implementation	Computing Paradigm	Configurable Architecture	Clock Cycles per p-Bit Update	AND Gate		Full Adder	
				LUT	FF	LUT	FF
Pervaiz *et al* [16]	Probabilistic	No	3	156	123	1345	586
Smithson *et al* [4]	Stochastic	No	N/A	257	307	400	329
This Work*	Probabilistic	Yes	3	64	72	143	90

*re-implemented with Kintex UltraScale FPGA for fair comparison

8 p-bits are for the product and the remaining 36 p-bits are ancillae used for storing and processing intermediate computation results. The corresponding **J** and **h** values are obtained from [25]. Our implementation of the 8-bit factorization p-circuit requires 156 clock cycles to update all 52 p-bits in a sample, and achieves maximum clock frequency of ≈ 60 MHz, as validated using our experimental setup with the Genesys ZU-5EV Zynq UltraScale+ MPSoC Board described earlier. Considering $N_s = 5000$ samples, it takes only 13 ms to obtain the probability distribution of all possible states using our hardware accelerator. As an example, with $143 = 11 \times 13$ as the input, we are able to obtain the factors as (11, 13) and (13, 11) with high probability.

C. Scalability Analysis

As discussed earlier, the number of BRAMs available in the target FPGA device limits the maximum number of p-bits that can be supported using our configurable p-circuit accelerator. For up to $N_{m(max)}$ p-bits, the total memory required to store **J** and **h** is $N^2_{m(max)}\, d + N_{m(max)}\, d = N_{m(max)}\, (N_{m(max)}+1)\, d$ bits. Therefore, the memory requirement scales quadratically with the number of p-bits, as shown in Fig. 12.

D. Comparison with Previous Work

While most of the existing literature has explored implementation of p-bits and p-circuits using emerging technologies such as magnetic tunnel junctions (MTJs) [5], [8] and resistive random access memories (RRAMs) [14], purely digital FPGA-based implementations of the basic building blocks and small invertible logic circuits have been demonstrated by [16] and [4]. Table II compares the resource utilization of FPGA implementations of probabilistic / stochastic circuits for AND gate and full adder from [16] and [4] with our application-specific p-circuit implementations. Since [16] and [4] were implemented on Xilinx Kintex UltraScale devices, we have also re-implemented our design for Kintex UltraScale FPGA with the same design parameters as [16] and [4] for fair comparison. Clearly, our proposed architecture with a single p-bit computation core (instead of an array of cores) provides significant savings in terms of logic resources compared to previous work. Our compact design still requires 3 cycles to update each p-bit, same as [16], so there is no loss in performance. This further emphasizes the resource-efficiency of our compact design which is particularly beneficial for p-circuits requiring large number of p-bits. Furthermore, unlike [16] and [4], our architecture allows the same accelerator to be configured to support various p-bit requirements.

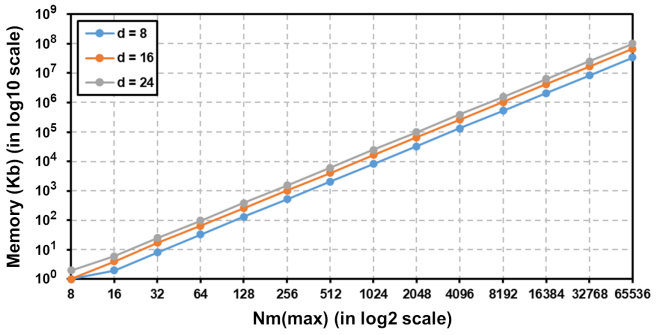

Fig. 12. Scaling of memory requirement (in Kb) for **J** and **h** with maximum number of p-bits $N_{m(max)}$ and $d \in \{8, 16, 24\}$.

V. CONCLUSIONS AND FUTURE WORK

In this work, we have presented the design and implementation of Tyche, a compact and configurable hardware accelerator for scalable probabilistic computing on FPGA. Our hardware architecture is configurable, thus allowing p-circuits requiring different number of p-bits to be evaluated using the same hardware. The use of a single p-bit computing core instead of an array of processing elements provides significant logic resource savings. A logarithmic adder tree is used for single-cycle weight logic computation while ensuring reasonable performance even for large number of p-bits. Various application-specific p-circuits are experimentally demonstrated using our proposed hardware accelerator implemented on Xilinx UltraScale+ FPGA, thus emphasizing the viability of practical scalable probabilistic computing on modern FPGA platforms. Future extensions of this work include the design of memory-efficient architectures for probabilistic computing, demonstration of larger p-circuits and development of efficient algorithms for application-specific p-circuit construction.

ACKNOWLEDGMENT

The authors would like to thank Matam Thrinethra, Kollipara Abhishek Anand, Akshit Agiwal and Kiran Kailash Magar for helpful technical discussions. They also thank Prof. Viveka K. R. and Prof. Debayan Das for their valuable feedback. This work was supported by a seed grant from the Indian Institute of Science and an M.Tech. scholarship from the Ministry of Education, Government of India.

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] S. C. Smithson, N. Onizawa, B. H. Meyer, W. J. Gross, and T. Hanyu, "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.

[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] H. Jung, H. Kim, W. Lee, J. Jeon, Y. Choi, T. Park, and C. Kim, "Probabilistic Prime Factorization based on Virtually Connected Boltzmann Machine and Probabilistic Annealing," *arXiv preprint arXiv:2210.14519*, Oct. 2022.

[7] 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.

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

[9] N. A. Aadit, A. Grimaldi, M. Carpentieri, L. Theogarajan, G. Finocchio, and K. Y. Camsari, "Computing with Invertible Logic: Combinatorial Optimization with Probabilistic Bits," in *IEEE International Electron Devices Meeting (IEDM)*, Dec. 2021, pp. 40–43.

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

[11] 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.

[12] S. Chowdhury, A. Grimaldi, N. A. Aadit, S. Niazi, M. Mohseni, S. Kanai, H. Ohno, S. Fukami, L. Theogarajan, G. Finocchio, S. Datta, and K. Y. Camsari, "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.

[13] 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.

[14] Y. Liu, Q. Hu, Q. Wu, X. Liu, Y. Zhao, D. Zhang, Z. Han, J. Cheng, Q. Ding, Y. Han, B. Peng, H. Jiang, X. Xue, H. Lv, and J. Yang, "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.

[15] A. Z. Pervaiz, L. A. Ghantasala, K. Y. Camsari, and S. Datta, "Hardware Emulation of Stochastic p-Bits for Invertible Logic," *Nature Scientific Reports*, vol. 7, no. 10994, Sep. 2017.

[16] A. Z. Pervaiz, B. M. Sutton, L. A. Ghantasala, and K. Y. Camsari, "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] N. A. Aadit, A. Grimaldi, M. Carpentieri, L. Theogarajan, J. M. Martinis, G. Finocchio, and K. Y. Camsari, "Massively Parallel Probabilistic Computing with Sparse Ising Machines," *Nature Electronics*, vol. 5, no. 7, pp. 460–468, Jul. 2022.

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

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

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

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

[22] Digilent Inc., "Genesys ZU: Zynq UltraScale+ MPSoC Development Board," https://digilent.com/shop/genesys-zu-zynq-ultrascale-mpsoc-development-board.

[23] S. Ruder, "An Overview of Gradient Descent Optimization Algorithms," *arXiv preprint arXiv:1609.04747*, Sep. 2016.

[24] N. Onizawa, K. Nishino, S. C. Smithson, B. H. Meyer, W. J. Gross, H. Yamagata, H. Fujita, and T. Hanyu, "A Design Framework for Invertible Logic," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 4, pp. 655–665, Apr. 2021.

[25] Purdue University, "Purdue-P," https://www.purdue.edu/p-bit.