A Parallel FPGA Architecture for Data Clustering using the Potts Model

Xingjian Gao
Department of Electrical and Computer Engineering
University of Alberta
Edmonton, Canada
xg6@ualberta.ca

Jie Han
Department of Electrical and Computer Engineering
University of Alberta
Edmonton, Canada
jhan8@ualberta.ca

***Abstract*—Efficient data clustering on edge devices with limited resources requires balancing algorithm complexity with hardware constraints. This paper introduces a parallel accelerator in field programmable gate arrays (FPGAs) for real-time clustering based on the Potts model. We present a deterministic solver that uses a 32-core single-instruction multiple-data (SIMD) architecture with interleaved memory banks to allow conflict-free parallel updates. By mapping the Hamiltonian to an integer-only formulation suitable for hardware implementation, it reduces the requirements for digital signal processor modules and improves logic efficiency and parallelism. Implemented on a Xilinx Kintex UltraScale+ FPGA, the solver converges in 3.07 ms for 4096 nodes, reaching a throughput of 1.33 million samples per second. Experimental results show that this architecture achieves accurate clustering on the evaluated synthetic benchmark and outperforms K-Means clustering in topological robustness, providing a scalable, high-accuracy solution for edge data analysis.**

***Index Terms*—Potts Model, Clustering Algorithm, Field Programmable Gate Array (FPGA)**

I. Introduction

As the internet of things (IoT) ecosystem expands, edge devices are expected to handle complex tasks beyond basic data processing, such as network routing and resource scheduling [1]. Although data clustering and combinatorial optimization (CO) focus on different tasks, both can be expressed as energy-minimization problems on graphs, making them suitable for domain-specific hardware implementations.

To meet this need, we consider the Potts model [2], a generalization of the Ising model [3], where each state represents a cluster label, and clustering is formulated as an energy-minimization problem. This physics-inspired approach offers clear benefits over conventional geometric techniques such as K-Means clustering [4]. Specifically, the energy landscape helps identify non-convex clusters based on local connectivity, whereas K-Means clustering relies on distances to global centroids. This feature improves performance on datasets with complex local structures and non-convex shapes.

Deploying Potts model solvers on resource-limited field programmable gate arrays (FPGAs) is challenging. Typical software solutions use simulated annealing (SA) to avoid local minima [5], but it involves costly floating-point computations and high-quality random number generators (RNGs) to mimic thermal fluctuations [6]. The incurred high latency and large area make them unsuitable for real-time edge use [7], [8]. In this work, we move away from physical simulation and focus instead on engineering efficiency. We propose a simplified digital update rule that replaces SA's sequential, stochastic updates with a highly parallel deterministic greedy method. This approximation is then assessed through on-chip experiments and bit-accurate software emulation. This approach circumvents the need for complex RNGs and exploits FPGA parallelism to quickly explore solutions.

Our design presents a 32-core parallel FPGA architecture that approximates the Potts model's energy minimization using lightweight, parallel-update logic and simple integer arithmetic. It employs a banked memory system to avoid write conflicts. Experimental results show that this approach outperforms K-Means in clustering accuracy on non-linear datasets while maintaining low digital signal processor (DSP) usage, making it suitable for resource-limited edge devices.

II. Preliminaries

A. The Potts Model

The Potts model describes a system of N interacting spins, where each spin $s_i \in \{0, \ldots, q-1\}$ represents the cluster assignment of a data point i [2]. The system is described by a Hamiltonian energy function $H(\mathbf{s})$ that combines a local similarity term with a global balancing penalty. The Hamiltonian is given as:

$$H(\mathbf{s}) = -\sum_{\langle i,j \rangle} J_{ij}\delta(s_i, s_j) + \gamma \sum_{k=0}^{q-1} \left(n_k - \frac{N}{q} \right)^2 . \quad (1)$$

The first term represents the interaction energy, where $\delta(s_i, s_j)$ is the Kronecker delta. The coupling strength J_{ij} describes pairwise data similarity. To preserve local relationships in the data, a sparse K-nearest-neighbor (K-NN) graph is built, and edge weights are assigned using a Gaussian kernel [9]:

$$J_{ij} = \begin{cases} e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}} & \text{if } j \in \mathcal{N}_i \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

979-8-3315-8229-6/26/$31.00 ©2026 IEEE

where $\mathbf{x}_i$ denotes the feature vector of node i, and σ is the width parameter. Minimizing this term favours neighbours in the high-dimensional space to share the same label.

The second term in (1) is a penalty term controlled by γ, which regularizes the solution by discouraging trivial partitions (e.g., all points in a single cluster), where n_k is the size of cluster k [10].

B. Deterministic Update Rule

Minimizing the Hamiltonian in (1) for a general graph is known to be NP-hard [10]. In software-based physics simulations, stochastic methods such as Metropolis-Hastings or SA are employed to traverse the energy landscape and escape local minima [11]. However, implementing these algorithms on FPGAs incurs considerable overhead due to the need for high-quality pseudorandom number generation (PRNG) and many stochastic iterations before reaching thermal equilibrium.

For hardware efficiency, we use a deterministic update scheme, which is formally related to the iterated conditional modes (ICMs) algorithm [12]. Unlike stochastic sampling, each processing unit computes the energy change ΔH for all possible candidate states and greedily selects the state that minimizes the local energy:

$$s_i^{(t+1)} = \arg \min_{k \in \{0,\dots,q-1\}} \left(\Delta H(s_i \to k)\right). \tag{3}$$

A spin update occurs only when the best candidate state is different from the current state and results in a negative energy change. While this greedy approach does not theoretically guarantee global optimality, it maps efficiently to digital logic by eliminating floating-point probability calculations. In practice, using a relatively dense neighborhood graph often results in more stable local update behavior.

C. Related Work on Clustering Hardware

K-Means clustering is commonly used in hardware acceleration because of its predictable memory access and linear per-iteration cost with respect to the number of data points N for fixed feature and cluster counts. Various FPGA implementations have optimized the Euclidean distance calculation and centroid updates to achieve high throughput [13]. However, K-Means clustering performs best when clusters are compact and clearly separated by their centroids, which limits its effectiveness on non-convex datasets. On the other end of the spectrum, spectral clustering can identify non-linear data structures but involves costly global matrix computations, making it unsuitable for resource-limited edge devices [14]. The proposed Potts-based architecture targets the design space between these two methods, providing the representational benefits of graph-based clustering with a computational structure suitable for parallel hardware acceleration.

III. The Parallel Potts Clustering Solver

To handle the algorithmic complexity of graph-based clustering with limited FPGA resources, we use a hardware-software co-design approach. The process is divided into two phases: graph construction and iterative solving. The host CPU builds the K-NN graph and sends the resulting topology and edge weights to the FPGA. Subsequently, the FPGA performs iterative energy minimization, characterized by extensive parallelism and low-latency memory requirements, using a specialized 32-core design.

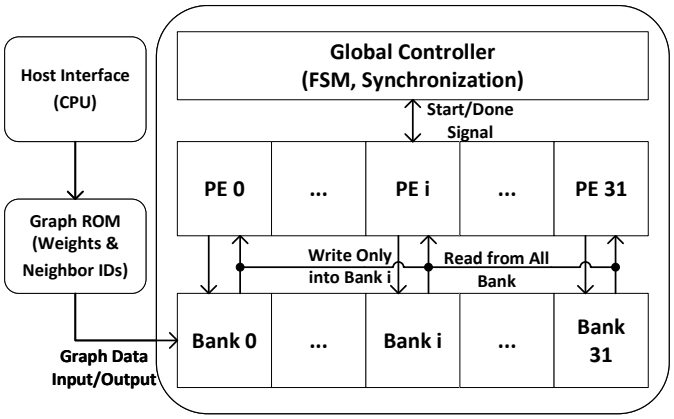

Fig. 1. Top-Level Architecture of the 32-Core Potts Solver. The design features static node partitioning and an interleaved memory subsystem to enable conflict-free parallel updates.

A. Top-Level Architecture

The proposed accelerator, shown in Fig. 1, features a single-instruction multiple-data (SIMD) architecture with $P = 32$ parallel processing elements (PEs). To improve throughput and prevent shared-state write coordination among processing elements, we use a static partitioning approach. The N nodes are partitioned into P contiguous segments, each managed by one PE responsible for roughly N/P nodes.

The system operation follows a three-phase finite state machine (FSM):

1) Initialization: The graph topology with neighbor IDs and edge weights is pre-loaded into on-chip memory, while the spin states are randomly initialized and spread across the processing elements.
2) Parallel Execution: Upon the start signal, all 32 PEs operate synchronously. They iterate over their assigned nodes, reading neighbor states and updating their local spins according to the deterministic dynamics.
3) Synchronization & Early Exit: The system monitors the total number of spin flips across all cores. If the total number of flips in an epoch drops to zero (indicating a converged low-energy state), the global controller asserts a Done signal, terminating the process to save power and reduce latency.

B. Interleaved Memory Subsystem

A critical challenge in parallelizing graph algorithms is managing memory access conflicts when multiple cores simultaneously read and write shared state variables. We resolve this issue using an interleaved memory-banking scheme.

The overall spin state S is spread across P separate block random access memory (BRAM) banks. In our hardware design, Core i exclusively writes to Bank i, which holds the

subset of spins assigned to that core. Read access, however, is available to all cores through a shared interconnect. When Core i calculates the energy of a node, it may need to retrieve the spin state of a neighbor managed by Core j. This setup enables:

- Conflict-Free Writes: Each spin is assigned to and updated by one PE, so concurrent write conflicts do not occur, and there is no need for lock-based or atomic read-modify-write mechanisms.
- High-Bandwidth Reads: The BRAM banks work simultaneously, providing enough combined bandwidth to supply data to all 32 PEs.

C. Pipelined Processing Elements (PEs)

To support high-throughput operation, each PE is organized as a four-stage pipeline that evaluates candidate states and applies the minimum-ΔH update rule. The operational process followed by the parallel PEs is detailed in Algorithm 1. This pipeline overlaps memory access with computation, helping to minimize the impact of memory latency on throughput.

Algorithm 1 Hardware-Aware Sparse Potts Solver

Input: Dataset X, Neighbor size K, Clusters Q, Balance γ
Output: Cluster assignments S

1: **Host:** Construct sparse K-NN graph J
2: **FPGA Init:** Randomize spins S; Count clusters N_c
3: **while** not converged **and** iter $<$ max_iter **do**
4: **for** each node i managed by PE (Parallel) **do**
5: $A \leftarrow s_i$ (Current State)
6: **Stage 1-2:** Fetch neighbors $\mathcal{N}_i$ and spins $S_{\mathcal{N}}$
7: **Stage 3:** $E_{int}(c) = \sum_j J_{ij}\delta(s_j, c)$
8: **Stage 4:** $\Delta H_{A\to c} = \Delta E_{int} + \Delta E_{bal}$
9: $B \leftarrow \arg\min_c(\Delta H_{A\to c})$
10: **if** $B \neq A$ **and** $\Delta H_{A\to B} < 0$ **then**
11: $s_i \leftarrow B$
12: Update Local Bank & Global Counts N_c
13: **end if**
14: **end for**
15: **end while**
16: **return** S

1) Stage 1: Graph Fetch (Line 6): In the initial stage of the pipeline, each PE creates the address for the current node's neighbor list and fetches the neighbor ID (n_{id}) along with the pre-computed interaction weight (J_{ij}) from the graph read-only memory (ROM).

2) Stage 2: State Retrieval (Line 6): Using the retrieved n_{id}, the PE accesses the Interleaved Memory Subsystem. It then maps the global ID to a specific memory bank and retrieves the neighbor's current spin state, s_j.

3) Stage 3: Energy Accumulation (Line 7): The PE gathers the interaction energy for all candidate clusters. Unlike typical Ising machines, which handle only binary states, our accumulator operates on q-state Potts logic. The interaction potential is calculated as a sum of weights for each cluster c:

$$E_{int}(c) = \sum_{j \in \mathcal{N}_i} J_{ij} \cdot \delta(s_j, c). \tag{4}$$

4) Stage 4: Update Logic (Lines 8-12): In the final stage, the PE calculates the total energy change ΔH for assigning the node to each candidate cluster by combining the interaction term and the cluster-balance penalty. To prevent complex squaring operations in hardware, we use a delta-update rule for the balance term:

$$\Delta E_{bal} = 2\gamma \cdot (n_{new} - n_{old} + 1), \tag{5}$$

where n_{new} and n_{old} are the global counts of the target and current clusters. The comparator evaluates if $\Delta H < 0$. If the minimum energy change is negative, the PE updates the local spin state and signals the global counter to update the cluster counts.

IV. Experimental Results

To evaluate the proposed architecture, we implemented the design on a Xilinx Kintex UltraScale+ FPGA (xcku3p-ffva676-3-e). These experiments aim to verify hardware correctness with synthetic data and explore resource trade-offs for dense graph interactions on edge devices.

A. Experimental Setup

The hardware was synthesized and routed with a target frequency of 100 MHz. For on-chip verification, we utilized a synthetic Gaussian Blobs dataset ($N = 4096$ nodes, $Q = 4$ clusters). The solver was configured with a relatively high connectivity degree, $K = 120$, and a balance strength, $\gamma = 20$. Although the synthetic blob dataset is quite simple, we set $K = 120$ to challenge the memory-banking and arithmetic pipelines with dense graph interactions.

B. Resource Utilization & Problem Scale

Table I compares our design with a standard FPGA K-Means implementation [13]. Rather than a direct efficiency comparison, this table highlights the architectural costs of supporting graph-based clustering relative to geometric partitioning.

TABLE I
Hardware Resource Consumption vs. Problem Scale

Metric	K-Means [13]	Proposed Potts Solver
Device	Virtex-6	Kintex UltraScale+
Nodes (N)	4096	4096
Connectivity (K)	~ 4 (Centroids)	120 (Neighbors)
LUTs	11,560	124,017
Flip-Flops (FFs)	11,171	20,266
DSP Blocks	28	30

1) Analysis of Logic Area: The proposed design uses 124,017 look-up tables (LUTs). This order-of-magnitude difference from K-Means stems from the fundamental difference in problem complexity. K-Means computes distances to a few centroids ($C = 4$), whereas our architecture computes interactions with $K = 120$ neighbours per node. To achieve this in real time, each of the 32 cores instantiates a parallel adder tree to accumulate these 120 interactions in a single pipeline pass. This massive logic footprint is necessary to enable dense graph processing on-chip.

2) Arithmetic Efficiency: Despite the high logic density, the design requires only 30 DSP slices, which is remarkably close to the K-Means implementation (28 DSPs). This validates the effectiveness of our integer-only optimization. By precomputing Gaussian kernels into integer weights and using delta updates for the Hamiltonian, we successfully offloaded the computational burden from scarce DSPs to the abundant logic fabric.

C. Performance and Accuracy

The 32-core system consistently converges on the $N = 4096$ dataset in about 3.07 ms on average, providing a throughput of 1.33 million samples per second. This performance highlights the architecture's higher processing efficiency: despite using about $10\times$ more logic resources than K-Means, it processes a $30\times$ larger workload per cycle, that is, 120 neighbor interactions vs. 4 centroid distances. This shows that the extra logic cost supports a much higher interaction workload per cycle. In terms of clustering quality, the hardware solver shows reliable convergence. Starting with random initialization, the deterministic greedy dynamics successfully identify the ground-truth clusters, achieving an adjusted rand index (ARI) of 1.0, where ARI measures the agreement between the predicted clustering and the ground-truth labels after adjusting for chance. This shows that the parallel update logic and integer-based energy formulation preserve correct clustering behavior on the evaluated synthetic benchmark.

D. Algorithmic Capability: Beyond Synthetic Benchmark

Although the on-chip evaluation utilized synthetic convex blobs to verify functional behavior after synthesis and implementation, the algorithm itself is not limited to these simple datasets.

To quantify this, we performed bit-accurate Python simulations of the proposed hardware logic using the UCI Optical Recognition of Handwritten Digits dataset ($N = 1797$, $D = 64$). These simulations were executed on a standard x86 workstation (Intel Core i7) to evaluate the algorithm without FPGA hardware constraints. The proposed Potts solver ($K = 30, \gamma = 10$) achieved 81.3% clustering accuracy, exceeding standard K-Means (79.4%) and remaining comparable to spectral clustering (80.5%) without the need for the costly global matrix computations that spectral clustering requires.

Because the FPGA design implements the same update logic evaluated in the bit-accurate software model, these results indicate that the proposed approach can go beyond the synthetic on-chip benchmark and remain competitive on a higher-dimensional nonlinear dataset.

V. Conclusion

This paper introduces a 32-core parallel FPGA architecture designed for data clustering based on the Potts model. Replacing stochastic annealing with a deterministic minimum-ΔH update rule and using precomputed integer edge weights along with integer delta updates for the balance term eliminates floating-point arithmetic, while still being suitable for parallel FPGA implementation. The findings reveal a key architectural trade-off: although the graph-based approach uses more logic than K-Means due to the extensive parallelism needed for neighbor processing, it maintains high arithmetic efficiency by using only 30 DSPs and offers the benefits of graph-based clustering for non-convex data. This research demonstrates an efficient architectural design: a $10\times$ increase in logic cost yields a $30\times$ increase in interaction density, enabling low-latency processing of graph-based clustering problems within edge hardware constraints. The proposed architecture also illustrates a reusable design pattern for other graph-based energy-minimization solvers on edge FPGAs.

References

[1] J. Lin, W.-M. Chen, Y. Lin, C. Gan, S. Han *et al.*, "Mcunet: Tiny deep learning on IoT devices," *Advances in neural information processing systems*, vol. 33, pp. 11 711–11 722, 2020.

[2] F.-Y. Wu, "The Potts model," *Reviews of modern physics*, vol. 54, no. 1, p. 235, 1982.

[3] B. A. Cipra, "An introduction to the Ising model," *The American Mathematical Monthly*, vol. 94, no. 10, pp. 937–959, 1987.

[4] J. B. McQueen, "Some methods of classification and analysis of multivariate observations," in *Proc. of 5th Berkeley Symposium on Math. Stat. and Prob.*, 1967, pp. 281–297.

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

[6] T. Zhang, Q. Tao, B. Liu, A. Grimaldi, E. Raimondo, M. Jiménez, M. J. Avedillo, J. Nunez, B. Linares-Barranco, T. Serrano-Gotarredona *et al.*, "A review of Ising machines implemented in conventional and emerging technologies," *IEEE Transactions on Nanotechnology*, vol. 23, pp. 704–717, 2024.

[7] E. Cohen, A. Mandal, H. Ushijima-Mwesigwa, and A. Roy, "Ising-based consensus clustering on specialized hardware," in *International Symposium on Intelligent Data Analysis*. Springer, 2020, pp. 106–118.

[8] B. Liu, T. Zhang, X. Gao, and J. Han, "An efficient simulated oscillator-based Ising machine on FPGAs," in *2024 IEEE 24th International Conference on Nanotechnology (NANO)*. IEEE, 2024, pp. 469–474.

[9] E. Güngör and A. Özmen, "Distance and density based clustering algorithm using Gaussian kernel," *Expert Systems with Applications*, vol. 69, pp. 10–20, 2017.

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

[11] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[12] J. Besag, "On the statistical analysis of dirty pictures," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 48, no. 3, pp. 259–279, 1986.

[13] L. A. Dias, J. C. Ferreira, and M. A. Fernandes, "Parallel implementation of k-means algorithm on FPGA," *IEEE Access*, vol. 8, pp. 41 071–41 084, 2020.

[14] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 14, 2001.