

Breaking the BRAM Wall: Scalable Vina FPGA Acceleration via Distributed Grid Storage and Cross-Board Long-Ring Pipelines

Ankun Tian¹, Shidi Tang¹, Ruiqi Chen², Ming Ling^{1,*}

¹School of Integrated Circuits, Southeast University, Nanjing, China ²ETRO, Vrije Universiteit Brussel, Belgium

*Corresponding author: trio@seu.edu.cn

Abstract—AutoDock Vina (Vina), a gold standard for molecular docking, is hampered by computational expense. Previous FPGA hardware accelerators were fundamentally constrained by an on-chip memory bottleneck, where storing large, pre-computed energy grids consumes the majority of BRAM resources. This leads to imbalanced resource utilization, as the exhausted on-chip memory makes it impossible to further increase the degree of intra-node parallelism by instantiating more processing units. This paper introduces a novel, scalable multi-FPGA architecture that systematically removes this limitation.

Our architecture’s innovation is a synergistic combination of three mechanisms. First, Distributed Grid Storage partitions the energy grid across all nodes to break the BRAM bottleneck. Second, a Cross-Board Long-Ring Pipeline creates a high-throughput dataflow for distributed energy calculations. Third, a dynamic intra-node scheduler unlocks massive fine-grained parallelism within each node. Together, these mechanisms create a powerful synergistic effect where adding nodes not only increases aggregate throughput but also enhances the performance of each individual node. This intrinsic amplification of per-node capacity is the direct driver of the system’s super-linear performance scaling.

Implemented on a three-ZCU102 FPGA system and without sacrificing accuracy, our single-board normalized performance is $7.6\times$ faster than the Vina-FPGA and $1.95\times$ faster than the state-of-the-art Vina-FPGA-Cluster. Critically, the architecture demonstrates super-linear performance scaling: the three-board system achieves a $3.7\times$ speedup over a single node, outperforming the ideal linear $3\times$ speedup.

I. INTRODUCTION

Molecular docking is an integral part of modern structure-based drug design, serving as a computational lens to scrutinize interactions between ligands and their macromolecular targets [1]. This procedure simulates the binding process to predict the optimal conformation and binding affinity of a ligand, proving pivotal for understanding biomolecular recognition and accelerating therapeutic development.

Among these tools, AutoDock Vina [2] is a reliable and scientifically validated candidate, established as a gold standard due to its robust hybrid scoring function and superior performance in benchmarks. However, Vina’s primary weakness is its significant computational expense, which obstructs its use in ultra-large virtual screening (ULVS). Previous acceleration strategies have encountered critical limitations. Vina-GPU [3], [4] increased concurrency and energy use without speeding up the core algorithm. Vina-FPGA [5] was fundamentally constrained by on-chip memory (BRAM) limitations from storing

entire energy grid tables. The subsequent Vina-FPGA-Cluster [6] was scaled out by distributing jobs across multiple boards, achieving task-level parallelism at the Exhaustiveness Level of the Vina algorithm. This design introduces a Bidirectional-AG module to create data-level parallelism within the innermost search loop, but was still constrained by requiring a full energy grid copy per node. General FPGA cluster architectures are often tailored for different classes of scientific computing problems. For example, accelerators for Molecular Dynamics (MD) [7]–[10] may use a spatial decomposition of the simulation space across nodes connected in a hyper-ring topology, while frameworks for Neural Network Quantum States (NNQS) might employ data-parallel training schemes where nodes process different molecules and synchronize gradients [11]–[15]. These architectural models are not directly transferable to Vina’s workload, as its primary bottleneck is neither spatial communication intensity nor gradient aggregation, but rather the rapid, low-latency access to large, pre-computed energy grid tables required during its iterative search process.

To overcome the memory bottleneck, this paper’s architecture on interconnected FPGAs has three core innovations. First, a distributed energy grid breaks the BRAM bottleneck by partitioning data across the cluster nodes. Second, a cross-board ring pipeline is designed to sustain numerous concurrent tasks. Third, by liberating local BRAM, we unlock unprecedented fine-grained parallelism within each node. The result is a synergistic, highly efficient, and high-throughput architecture.

The key contributions of this work are as follows.

- **Throughput-Optimized Distributed Storage:** Our foundational contribution is a spatial partitioning strategy that decomposes the monolithic energy grid and distributes its sub-grids into the local BRAM of each FPGA node. This approach directly solves the primary memory capacity bottleneck, liberating critical BRAM resources for further parallelization.
- **Cross-Board Long-Ring Pipeline:** Building upon this distributed data model, we designed a computational pipeline unrolled across the entire FPGA cluster. In this architecture, a task packet traverses the ring, accumulating partial energy values from each node’s local grid partition. This transforms memory access conflicts into a high-throughput, pipelined dataflow.
- **Dynamic Intra-Node Parallelism:** Within each FPGA, we implement a hybrid PS-PL architecture where a dynamic scheduler dispatches tasks to a deep, reusable pool

This work is supported in part by the National Natural Science Foundation of China under Grants 92464301. This research work is also supported by the Big Data Computing Center of Southeast University.

of fine-grained pipeline modules. This massively parallel design is dedicated to accelerating the most computationally dominant part of Vina.

- **High System Performance and Super-Linear Scalability:** Our implemented three-node system demonstrates superior performance, achieving a $1.95\times$ speedup over the state-of-the-art single-board Vina-FPGA-Cluster. The architecture exhibits robust super-linear scalability; the performance of the three-node system is $3.7\times$ that of our single-node implementation, exceeding the ideal linear speedup of $3\times$.

II. BACKGROUND

A. The Vina Computational Workflow

The AutoDock Vina computational process is guided by a hybrid algorithm, as illustrated in Figure 1. A global search generates new candidate conformations through random mutations (Mutate Con), each of which is then refined by the computational core: the Broyden-Fletcher-Goldfarb-Shanno (BFGS) local search algorithm [2], [5]. The BFGS process is iterative, repeatedly using an Armijo-Goldstein (AG) line search to find an optimal conformation, followed by an update to its internal Hessian matrix approximation (H-update) to guide the next iteration.

The AG search is the most computationally intensive part of this workflow. As detailed on the left of Figure 1, a single AG iteration involves a sequence of steps: a conformation update (POT update), conversion to coordinates (POT2Coords), the core Energy Calculation (summing Inter- and Intra-molecular energies), and finally, the gradient calculation (Derivation). This sub-procedure may loop multiple times until a sufficient energy decrease is found, after which the final result is evaluated by the global search’s Metropolis acceptance criterion.

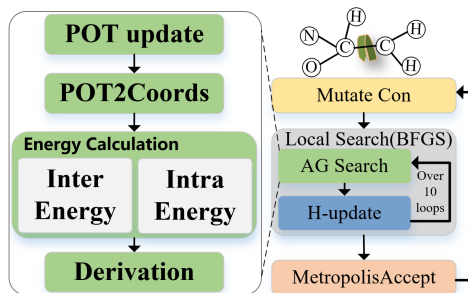


Fig. 1: Workflow of Vina algorithm

B. The Vina Scoring Function and Energy Grid

The Vina scoring function estimates binding affinity as the sum of intermolecular and intramolecular energies. Calculating intermolecular energy, which is the primary component, via direct pairwise summation $\mathcal{O}(N_{ligand} \cdot N_{receptor})$, where N_{ligand} and $N_{receptor}$ are the number of atoms in the ligand and receptor, respectively, is computationally prohibitive for large systems. To overcome this, Vina employs pre-calculated 3D affinity grids for each ligand atom type, a strategy fundamental to its efficiency. During the docking search, the energy for a given ligand conformation is determined via rapid *trilinear interpolation* from surrounding points on the grid,

which reduces the computational complexity to a manageable $\mathcal{O}(N_{ligand})$. Figure 2 illustrates the concept of representing a large molecule (such as a receptor) using an energy table. While crucial for performance, these grids can require several megabytes of storage. For hardware accelerators, this large memory footprint creates a significant challenge, as it consumes a substantial portion of the available on-chip memory (BRAM), creating a major performance bottleneck that limited previous FPGA-based designs.

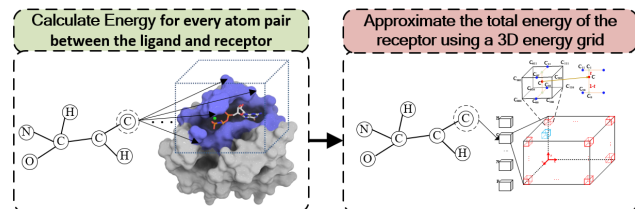


Fig. 2: Using a pre-computed energy grid to approximate a receptor’s energy.

Vina’s core computational loops require millions of iterative grid lookups. To maintain high-throughput and avoid stalling deep FPGA pipelines, these frequent lookups demand extremely low-latency memory access. Consequently, the grids must be stored in fast on-chip BRAM, as the significant latency of off-chip DRAM would create a primary performance bottleneck.

C. Challenges in Parallelism and Scalability

While AutoDock Vina has coarse-grained parallelism suitable for multi-core CPUs, achieving fine-grained acceleration on FPGAs presents three fundamental challenges, as illustrated in the top panel of Figure 3.

1) BRAM Resource Bottleneck: The first and most critical challenge is the on-chip memory capacity. A typical Vina energy grid can exceed 30Mb in size, which, for low-latency access, must be stored entirely in BRAM. This single requirement can consume the vast majority of a high-end FPGA’s BRAM resources, creating a hard memory wall and leading to memory access conflicts when multiple processing units are deployed.

2) Logic Resource Underutilization: The BRAM bottleneck directly causes a severe imbalance in resource usage, as depicted in the "Logic Resource Underutilization" panel of Figure 3. For a typical memory-bound design, the chart shows BRAM usage reaching a near-saturation level of **84.13%** while other critical logic resources are utilized at a much lower rate. This disparity prevents the instantiation of more parallel processing elements, which leads to inefficient hardware utilization.

3) Unbalanced Pipeline Latency: The third challenge arises from the disparity in computational delays between different tasks. The latency for memory-intensive grid lookups can be more than double that of simple arithmetic stages. This imbalance stalls the entire pipeline, creating bubbles where faster modules are idle while waiting for the bottleneck stage to complete, severely degrading overall throughput.

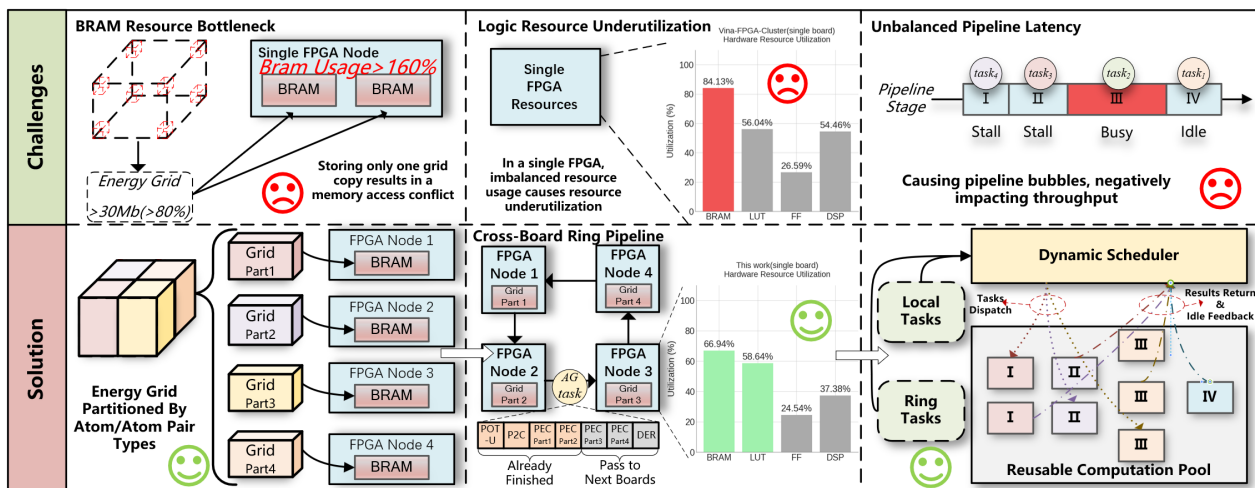


Fig. 3: Top: Key challenges in single-FPGA acceleration including BRAM bottleneck, resource underutilization, and pipeline imbalance. Bottom: Our solutions featuring a distributed energy grid, a ring pipeline architecture, and a dynamic task scheduler.

III. RING-BASED AND DISTRIBUTED-DATA ARCHITECTURE

The fundamental limitation of prior single-device accelerators was that their finite on-chip memory capacity created a performance bottleneck long before logic resources were fully utilized. The architecture detailed herein orchestrates a fundamental shift by re-architecting the relationship between computation, memory, and communication. Figure 3 outlines this solution and its contrast to the challenges faced by previous designs.

A. System Overview

Our overall architecture is a decentralized cluster of N homogeneous FPGA nodes arranged in a high-speed ring topology, as shown in Figure 3. Each node, whose internal architecture is detailed in Figure 5, uses two SFP optical ports [16], [17] to form a closed, unidirectional communication loop with its neighbors via point-to-point links.

B. Distributed Energy Grid

The co-design of our data distribution strategy and our circulating computation model is the central mechanism for eliminating the BRAM bottleneck. A typical Vina energy grid can occupy several megabytes, and storing this entire structure in the limited on-chip BRAM of a single FPGA was the defining scalability wall of prior accelerators.

Our architecture circumvents this limitation through a spatial partitioning strategy. The 3D energy grid is decomposed into N non-overlapping sub-volumes. During initialization, each node $FPGA_i$ loads only its corresponding partition i into its local BRAM. This strategy immediately reduces the on-chip memory requirement for grid storage on any single device by a factor of N , liberating the resources required for a high density of computational PEs in the PL.

This data partitioning scheme enables a “computation-follows-data” paradigm. Lightweight task packets circulate to the nodes where the necessary data resides locally. This guarantees that all high-bandwidth memory accesses for the

computationally intensive grid lookups are local to the processing node, eliminating cross-chip contention and resolving the BRAM capacity issue that crippled prior art.

C. Cross-Board Long-Ring Pipeline

The core of our architecture is the Cross-Board Long-Ring Pipeline, a single, deep computational pipeline spatially unrolled across the entire N nodes cluster. This model fundamentally redefines the AG search by physically distributing its most intensive part, the intermolecular energy calculation, across the ring, synergistically maximizing both throughput and resource utilization.

A task packet must complete a full traversal of this long ring to be finalized. At each node it visits, it accumulates a partial energy value calculated using that node’s local grid partition. The pipeline’s depth, equal to the number of nodes, is the key to throughput: it allows numerous independent docking tasks to be in-flight simultaneously, each occupying a different physical stage of the ring. This massively parallel workflow is a key design choice for large-scale screening where aggregate throughput is paramount.

Crucially, this cross-board execution model is what unlocks the system’s compelling scaling characteristics. By partitioning the grid, substantial BRAM on each node is liberated from storing the monolithic dataset. This freed resource is immediately repurposed to instantiate a greater number of parallel Processing Elements and their dedicated grid replicas, a feat impossible in prior memory-bound designs

The data flow for a single AG iteration on a system with more than four FPGAs, initiated at an arbitrary node $FPGA_i$, as described in figure 4, exemplifies this model.

The process unfolds as follows:

- 1) **Initiation (on Node i):** An AG iteration begins on a starting node, $FPGA_i$. One of its local AG pipelines performs the *Con Update* and *Con convert to coords* steps and calculates the intramolecular energy.
- 2) **Pipelined Energy Calculation (Nodes i through $(i + N - 1) \pmod{N}$):** The task packet begins a traversal of the ring. At each node $FPGA_j$, the packet is routed to

one of the multiple parallel Partial Energy Calculation modules. This module accesses its dedicated BRAM copy of the local grid partition G_j to calculate a partial intermolecular energy, which is added to the packet's running total. The packet is then immediately forwarded to the next node, $FPGA_{((j+1) \pmod N)}$.

- 3) **Finalization (on Node $(i + N - 1) \pmod N$):** After a full circle, the packet arrives at $FPGA_{((i+N-1) \pmod N)}$. One of its AG pipelines performs the *Derivation* and the AG condition check using the now-complete total energy.
- 4) **Continuation:** If the AG conditions are not met, the state is updated, and the next AG iteration begins on this same node, $FPGA_{((i+N-1) \pmod N)}$. This naturally permutes the roles of the FPGAs, ensuring dynamic load balancing.

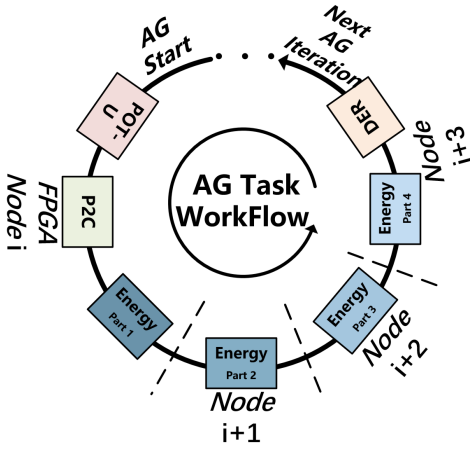


Fig. 4: A single task packet circulates the ring, accumulating partial energy calculations at each node.

This mechanism creates a powerful synergistic effect: adding a new node enhances the computational capacity of all existing nodes. The performance of a single node, $P(N)$, thus becomes an increasing function of the total cluster size N . This intrinsic property predetermines a super-linear growth phase, where the system's aggregate performance, approximately $N \times P(N)$, outpaces the node count. This phase is bounded, however, as the system's bottleneck inevitably shifts from memory to the on-chip logic and communication, a transition we will quantify in our evaluation.

IV. INTRA-NODE ARCHITECTURE

While the ring interconnect enables system-level scaling, it is the intra-node architecture, depicted in Figure 5, that truly dismantles the BRAM contention bottleneck. This design achieves massive fine-grained parallelism by synergistically combining a hybrid PS-PL model with a sophisticated on-chip dataflow management system. As shown, the architecture leverages a dynamic scheduler to manage tasks from both local and ring sources, feeding them into a deep, reusable computation pool. The following sections detail these key components.

A. Hybrid PS-PL Computational Model

Each node is a self-contained, hybrid processing system leveraging the distinct capabilities of a Processing System (PS) and Programmable Logic (PL), as depicted in Figure 5.

- **Processing System (PS):** The ARM-based PS orchestrates the high-level BFGS algorithm, handling sequential tasks like the Hessian Matrix Update and the Metropolis Accept criterion.
- **Programmable Logic (PL):** The PL accelerates the core Armijo-Goldstein (AG) line search. The PS offloads AG tasks to a Task Scheduler, which dispatches them to parallel modules: POT-U (POT Update), P2C (POT to Coordinates), PEC (Partial Energy Calculation), and DER (Derivation).

This hybrid model allows high-throughput AG calculations to be offloaded to custom hardware, while the PS efficiently manages the less parallelizable, control-flow-heavy parts of the algorithm.

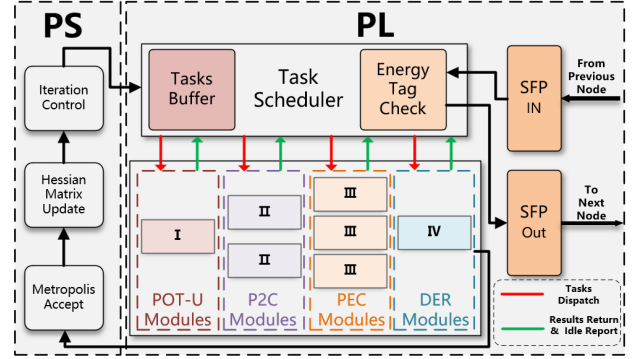


Fig. 5: Internal architecture of a single FPGA node. A dynamic Task Scheduler in the PL dispatches both ring and local tasks to a reusable pool of parallel processing modules.

B. Throughput-Oriented Fine-Grained AG Pipeline

The PL fabric is populated with P identical Processing Elements (PEs), each architected as a fine-grained, four-stage pipeline to process AG search tasks. The modules within each PE are connected in a streaming fashion, allowing a new operation to enter the pipeline while previous ones are still being processed. To support dynamic scheduling and high throughput, each computational stage is instantiated with multiple parallel units. The four core modules, as shown in The Figure 5 and their replication counts are as follows:

- M_{POTU} : The *POT Update* module, with C_{POTU} parallel units, calculates the potential energy terms.
- M_{P2C} : The *POT to Coordinates* module, with C_{P2C} parallel units, converts the potential representation to Cartesian coordinates.
- M_{PEC} : The *Partial Energy Calculation* module, with C_{PEC} parallel units, computes the molecular energy using the local grid partition.
- M_{DER} : The *Derivation* module, with C_{DER} parallel units, computes the gradient of the energy function with respect to the atomic coordinates.

A task flows through these modules sequentially. The replication of each module ($C_{POTU}, C_{P2C}, C_{DER} > 1$) ensures that no single stage becomes a rigid bottleneck and allows for flexible, dynamic allocation of tasks to available hardware resources.

C. Dynamic Task Flow

The key to the node’s high efficiency is its ability to dynamically manage and schedule a concurrent mix of local and ring-based tasks. As shown in Figure 4, this is orchestrated by a central Task Scheduler within the PL, which intelligently manages the flow of data through the reusable computation pool.

Tasks originate from two distinct sources. Local AG search tasks, initiated by the PS, are first placed into a Tasks Buffer. Concurrently, in-flight tasks arriving from the ring network via the SFP port [16], [17] are vetted by an Energy Tag Check module. This module inspects metadata within each task packet specifically, a counter tracking which energy partitions have been processed to determine if the task requires computation on this node.

The Task Scheduler acts as the control hub, constantly monitoring the idle/busy status of every parallel processing module (POT-U, P2C, PEC, and DER). Based on real-time resource availability and the validation from the tag check, it dispatches tasks to the appropriate computation units. A new local task is dispatched to an available POT-U module to begin its pipeline, while a validated ring task is sent directly to an available PEC module for its partial energy calculation.

After a task completes the PEC stage, its routing is critical. A decision is made based on its completion status. If the energy calculation is incomplete (i.e., the counter is less than N), the task is forwarded to the SFP output port for transmission to the next node in the ring. If the calculation is finished, the task is retained locally and passed to an available Derivation (DER) module for finalization. This centralized, dynamic scheduling ensures maximal resource utilization and maintains a high-throughput data flow, which is critical for the cluster’s overall performance.

V. EVALUATION

This section presents a comprehensive evaluation of our proposed ring-based FPGA cluster architecture. We first detail the experimental setup and baselines. We then validate the scientific accuracy of our implementation, followed by a rigorous analysis of single-board throughput and multi-board scalability, and finally report the hardware resource utilization.

A. Evaluation Setup

1) Implementation: Our experimental platform consists of a three-node FPGA cluster. Each node is a Xilinx Zynq UltraScale+ MPSoC ZCU102 development board. The nodes are interconnected in a unidirectional ring topology using onboard SFP optical ports operating at 12.5 Gbps. The entire design is implemented in Verilog HDL, utilizing an 18-bit fixed-point representation to maintain scientific accuracy.

2) Baseline: To comprehensively assess the performance, we compare it against state-of-the-art baselines, including the official AutoDock Vina on CPU [2], the Vina-FPGA accelerator [5], the Vina-FPGA-Cluster [6] and the Vina-GPU [3], [4].

B. Docking Accuracy

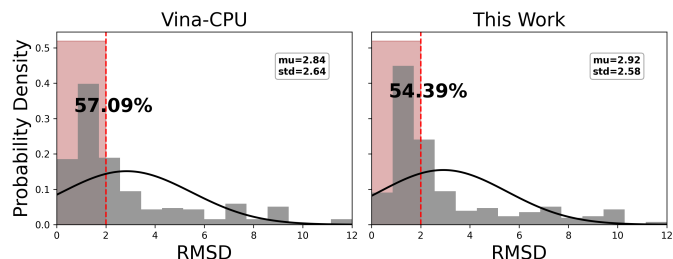


Fig. 6: RMSD distribution comparison between our work and Vina-CPU on the PDBbind v2020 refined set.

To ensure scientific validity, we compared our accelerator against AutoDock Vina on the PDBbind v2020 refined set [18]. As shown in Figure 6, the proportion of high-quality poses (RMSD < 2.0 Å) [19] is nearly identical to the Vina-CPU (54.39% vs. 57.09%). This confirms our accelerator produces scientifically equivalent results, with minor deviations arising from the acceptable use of fixed-point quantization required for hardware implementation.

C. Single-Node Performance Analysis

Figure 7a summarizes our performance against state-of-the-art baselines. Our three-node cluster, with its performance normalized to a single board, achieves a computation time of 6.08s. This represents a $29.7\times$ speedup over Vina-CPU and is notably faster than both Vina-FPGA-Cluster (11.88s) and Vina-GPU (7.27s).

Critically, the figure also reveals a powerful multiplier effect on single-node throughput: a board’s performance is not fixed but is amplified as the cluster grows. A standalone node takes 8.38s, which improves to 6.66s within a 2-node system, and further to 6.08s in a 3-node system. This directly validates the architectural principle outlined in Section III.C: as adding nodes reduces the BRAM storage burden on each device, the liberated memory is repurposed to instantiate more parallel Processing Elements (PEs), thus enhancing the computational capacity of every node in the cluster.

D. System Scalability

The amplification of single-node performance is the engine that drives the super-linear scalability of the overall cluster. Figure 7b presents our scalability model, plotting the aggregate system speedup against the number of nodes. The curve, anchored by experimental data, exhibits a distinct super-linear trajectory, achieving a $2.5\times$ speedup with two nodes and a $4.13\times$ with three nodes, significantly outperforming the linear ideal. This is the result of a powerful synergy between the additive effect of deepening the task pipeline and the multiplicative effect of enhancing each node’s internal throughput.

However, this explosive growth is bounded. The model in Figure 7b correctly projects a transition towards saturation as the system’s bottleneck dynamically shifts from BRAM to on-chip logic resources (LUTs). The curve clearly shows diminishing returns, providing critical insight into selecting an optimal cluster size. The most significant performance gains occur up to 6 nodes ($8.7\times$ speedup). Beyond this, the

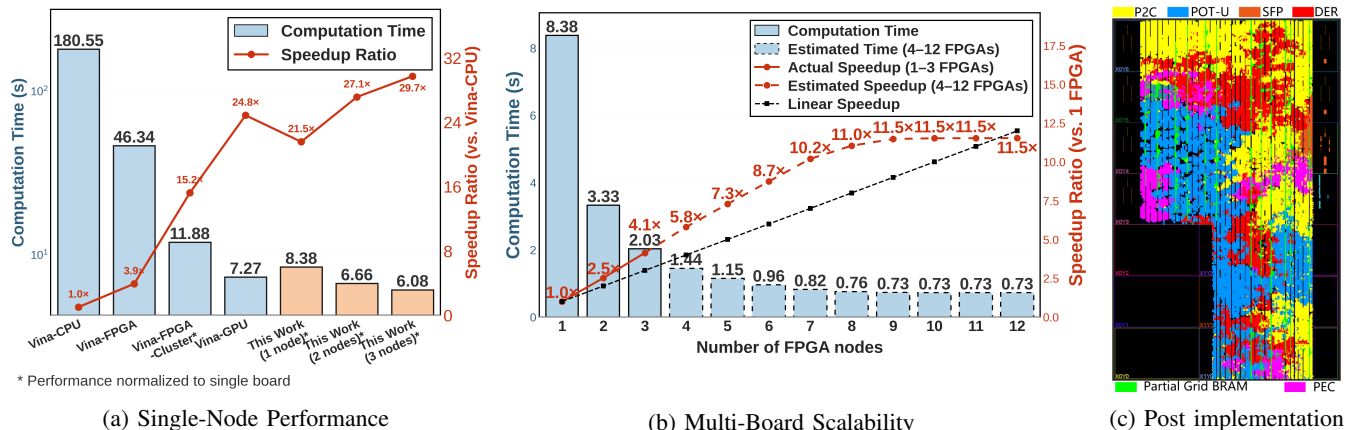


Fig. 7: Overall performance evaluation and hardware implementation.

TABLE I: Overall Comparison of Performance, Power, Energy, and Resource Utilization

Design	Performance		Power & Energy			Resource Utilization(Count/%)			
	Time (s)	Speedup vs. CPU	Power (W)	Energy (J)	Energy Eff. vs. CPU	LUT	FF	BRAM	DSP
CPU	182.28	1.0x	47.34	8629.14	1.0x	—	—	—	—
Vina-GPU [4]	7.27	25.1x	67.2	488.54	17.7x	—	—	—	—
Vina-FPGA [5]	46.34	3.9x	4.7	217.80	39.7x	259586 (78.2%)	281322 (42.4%)	1080 (100%)	390 (14.1%)
Vina-FPGA-Cluster* [6]	11.88	15.3x	4.9	58.21	148.2x	129127 (56.04%)	122530 (26.59%)	262.5 (84.13%)	941 (54.46%)
This Work(3 nodes)*	6.08	29.9x	4.78	29.11	296.4x	160732 (58.64%)	134518 (24.54%)	610.5 (66.94%)	942 (37.38%)

*Performance metrics are normalized to a single board for fair comparison.

incremental benefit of adding hardware decreases sharply; for instance, scaling from 6 to 9 nodes only yields an additional 1.3x in speedup (from 8.7x to 11.0x), indicating a far lower return on investment. The performance effectively plateaus at 10 nodes (11.5x speedup).

Therefore, for practical deployment, our analysis suggests an optimal cost-performance balance is achieved between 6 and 8 nodes. This range maximizes the architectural benefits before the system becomes limited by logic resource saturation and the accumulating communication overhead inherent in a larger ring. This rigorous model preempts questions about the limits of super-linear scaling by demonstrating not only its cause but also its predictable and finite boundaries.

E. Performance and Energy Efficiency

Our architecture’s performance and efficiency are summarized in Table I. With a single-board normalized time of 6.08s, our design is 7.6x faster than Vina-FPGA and 2.4x faster than Vina-FPGA-Cluster. While consuming less than a quarter of the power of Vina-GPU, our system achieves superior energy efficiency, requiring only 29.11J per task. This marks a 7.8x and 21.2x energy reduction compared to Vina-FPGA and Vina-GPU, respectively, highlighting the benefits of our distributed architecture.

F. Resource Utilization

Figure 8 presents an ablation study that visually confirms how our innovations achieve a balanced resource profile, a key differentiator from prior memory-bound accelerators like Vina-FPGA (100% BRAM utilization) and Vina-FPGA-Cluster (84.13%) shown in Table I. The baseline design (based on Vina-FPGA-Cluster, implemented on ZCU102) is severely memory-

bound, with BRAM usage at 99.78%. Our **Distributed Grid** strategy resolves this by partitioning the grid, drastically reducing the BRAM footprint required on each node. With BRAM resources liberated, our **Dynamic Scheduler** then effectively utilizes the freed-up space to instantiate more parallel logic, increasing overall LUT utilization to 58.64%. This results in a final balanced state, quantified by the low 1.14 BRAM/LUT ratio, confirming a successful shift from a memory-centric to a balanced design.

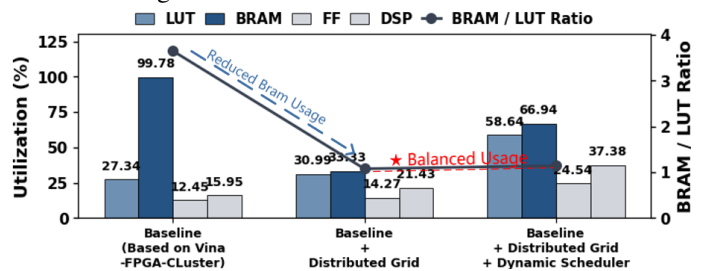


Fig. 8: Ablation Study on Resource utilization.

VI. CONCLUSION

This paper addressed the fundamental BRAM wall that has constrained Vina hardware accelerators. We introduced an architecture built on the synergistic co-design of a distributed grid, a cross-board ring pipeline, and a dynamic intra-node scheduler. This approach not only mitigates the memory bottleneck to unlock massive intra-node parallelism, but also enhances each node’s performance as the cluster grows. The resulting super-linear scaling provides definitive evidence of this scalable design, offering a new blueprint for accelerating memory-bound scientific applications.

REFERENCES

- [1] J. M. Paggi, A. Pandit, and R. O. Dror, "The Art and Science of Molecular Docking," *Annual Review of Biochemistry*, vol. 93, no. 1, pp. 389–410, Aug. 2024.
- [2] O. Trott and A. J. Olson, "AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading," *Journal of Computational Chemistry*, vol. 31, no. 2, pp. 455–461, 2010.
- [3] S. Tang, J. Ding, X. Zhu, Z. Wang, H. Zhao, and J. Wu, "Vina-GPU 2.1: Towards further optimizing docking speed and precision of AutoDock vina and its derivatives," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 21, no. 6, pp. 2382–2393.
- [4] J. Ding, S. Tang, Z. Mei, L. Wang, Q. Huang, H. Hu, M. Ling, and J. Wu, "Vina-GPU 2.0: Further Accelerating AutoDock Vina and Its Derivatives with Graphics Processing Units," *Journal of Chemical Information and Modeling*, vol. 63, no. 7, pp. 1982–1998, Apr. 2023.
- [5] M. Ling, Q. Lin, R. Chen, H. Qi, M. Lin, Y. Zhu, and J. Wu, "Vina-FPGA: A Hardware-Accelerated Molecular Docking Tool With Fixed-Point Quantization and Low-Level Parallelism," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 4, pp. 484–497, Apr. 2023.
- [6] M. Ling, Z. Feng, R. Chen, Y. Shao, S. Tang, and Y. Zhu, "Vina-FPGA-Cluster: Multi-FPGA Based Molecular Docking Tool With High-Accuracy and Multi-Level Parallelism," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 18, no. 6, pp. 1321–1337, Dec. 2024.
- [7] C. Wu, S. Bandara, T. Geng, V. Sachdeva, W. Sherman, and M. Herbordt, "System-level modeling of GPU/FPGA clusters for molecular dynamics simulations," in *2021 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, pp. 1–8.
- [8] C. Wu, T. Geng, A. Guo, S. Bandara, P. Haghi, C. Liu, A. Li, and M. Herbordt, "FASDA: An FPGA-aided, scalable and distributed accelerator for range-limited molecular dynamics," *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–14, Nov. 2023.
- [9] Y. Wu, W. Cao, J. Zhao, and H. Shang, "Fast and scalable neural network quantum states method for molecular potential energy surfaces," *IEEE Transactions on Parallel and Distributed Systems*, vol. 36, no. 7, pp. 1431–1443, Jul. 2025.
- [10] Q. Deng, Q. Liu, M. Yuan, X. Duan, L. Gan, J. Yang, W. Zhao, Z. Zhang, G. Wu, W. Luk, H. Fu, and G. Yang, "Acceleration of multi-body molecular dynamics with customized parallel dataflow," *IEEE Transactions on Parallel and Distributed Systems*, vol. 35, no. 12, pp. 2297–2314.
- [11] J. Shen, D. Wang, Y. Huang, M. Wen, and C. Zhang, "Scale-out acceleration for 3d CNN-based lung nodule segmentation on a multi-FPGA system," in *Proceedings of the 56th Annual Design Automation Conference 2019*. ACM, pp. 1–6.
- [12] T. Wang, T. Geng, A. Li, X. Jin, and M. Herbordt, "FPDeep: Scalable acceleration of CNN training on deeply-pipelined FPGA clusters," *IEEE Transactions on Computers*, pp. 1–1.
- [13] Y.-C. Lin, B. Zhang, and V. K. Prasanna, "HitGNN: High-throughput GNN training framework on CPU+multi-FPGA heterogeneous platform," *IEEE Transactions on Parallel and Distributed Systems*, vol. 35, no. 5, pp. 707–719.
- [14] J. Shan, M. T. Lazarescu, J. Cortadella, L. Lavagno, and M. R. Casu, "Power-optimal mapping of CNN applications to cloud-based multi-FPGA platforms," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 12, pp. 3073–3077.
- [15] S. Hong, S. Moon, J. Kim, S. Lee, M. Kim, D. Lee, and J.-Y. Kim, "DFX: A low-latency multi-FPGA appliance for accelerating transformer-based text generation," in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, pp. 616–630.
- [16] R. Fan and Y. Yamaguchi, "A study of FPGA-based cluster computing by high-speed serial-link communication," in *2020 Eighth International Symposium on Computing and Networking Workshops (CANDARW)*. IEEE, pp. 401–405.
- [17] S. Shin, S. Choi, E. Lee, S. Lee, and H. Yoo, "Implementation of aurora interface using sfp+ transceiver," in *2022 19th International SoC Design Conference (ISOCC)*, 2022, pp. 350–351.
- [18] R. Wang, X. Fang, Y. Lu, and S. Wang, "The PDBbind database: Collection of binding affinities for protein-ligand complexes with known three-dimensional structures," *Journal of Medicinal Chemistry*, vol. 47, no. 12, pp. 2977–2980, Jun. 2004.
- [19] G. Jones, P. Willett, R. C. Glen, A. R. Leach, and R. Taylor, "Development and validation of a genetic algorithm for flexible docking," *Journal of Molecular Biology*, vol. 267, no. 4, pp. 1078–1107, Apr. 1997.