

# Memory Optimization of a Quantum-Inspired Processor for High-Speed Computation

Moulisvaran U<sup>1</sup>, Pramoth K.P<sup>1</sup>, Rithika Naveencharan<sup>1</sup>, Sheena Christabel Pravin<sup>1</sup> and Reena Monica P<sup>1\*</sup>

<sup>1</sup>School of Electronics Engineering, Vellore Institute of Technology Chennai Campus, India

**Abstract.** The Quantum-like Accelerator for Tangled (QAT) Processor is a high-performance computational unit built to simulate quantum superposition states. It does this by processing multiple bits in parallel, similar to the working of SIMD (Single Instruction, Multiple Data) architectures, while avoiding wavefunction collapse. Designed to work alongside the SHAKTI C-class processor, the QAT improves both data transfer efficiency and overall computational throughput through custom Instruction Set Architecture extensions. One of its more pressing challenges is its large memory footprint which is largely a consequence of its extensive Array of Bits (AoB) structure. To address this challenge, bit compression techniques such as quantization have been implemented, effectively reducing memory usage from 65,536 AoB to 32,768 AoB while maintaining computational fidelity. This enables the QAT to store twice the amount of data, significantly improving efficiency in quantum-inspired computing environments. The proposed architecture paves the way for enhanced quantum-classical hybrid processing, offering a scalable and efficient solution for high-performance computing applications.

## 1 Introduction

Quantum-inspired computing seeks to emulate key principles of quantum mechanics, such as superposition and entanglement, using classical hardware. This approach enables the development of algorithms and architectures that harness computational advantages similar to those of quantum computers, without relying on quantum hardware. One such architecture is the Tangled/QAT (Quantum-like Accelerator for Tangled) architecture, designed to replicate quantum-like processes using an innovative Array of Bits (AoB) representation [1]. Quantum coprocessors, particularly Quantum Processing Units (QPUs), have emerged as critical components of quantum computing, developed by companies like IBM, Google, and Rigetti. QPUs are designed to efficiently handle quantum simulations, machine learning, and combinatorial optimization tasks that are intractable for classical processors. Their growing significance extends to fields such as cryptography, optimization, and scientific computing, making them indispensable in next-generation computing. In hybrid computing systems, quantum coprocessors are employed for quantum-specific computations, while classical processors handle general-purpose tasks, resulting in optimized system performance.

---

\* Corresponding author: Reena Monica P, [reenamonica@vit.ac.in](mailto:reenamonica@vit.ac.in)

Research in quantum computing has predominantly focused on the development of coherent qubits and the improvement of gate fidelities, with less emphasis on programmability and the integration of quantum processors into existing computing infrastructure. Significant challenges, such as error correction, qubit connectivity, and routing, need to be addressed for a fully operational quantum system. These challenges highlight the necessity of a flexible Quantum Instruction Set Architecture (QISA) that can adapt to the limitations of current quantum technologies. Additionally, the difficulties of scaling up quantum devices, particularly the increasing demand for quantum operations, have put a strain on the instruction issue rate, further complicating the development of scalable quantum processors.

The QAT architecture operates on large AoB values, with each AoB represented as a 65,536-bit array, capable of encoding multiple entangled states. However, storing and manipulating such large AoB values requires numerous registers, significantly increasing memory consumption. The QAT coprocessor in this implementation uses 256 registers, each 65,536 bits wide, resulting in a total of 2 MB register memory, posing a considerable challenge on FPGAs with limited on-chip memory capacity (typically between 4–6 MB). This poses a challenge in memory-constrained environments, limiting the architecture's scalability and practical deployment. To mitigate these memory constraints, bit compression techniques have been integrated into the QAT architecture. Specifically, normal quantization and mid-rise quantization are employed to effectively compress AoB values, achieving approximately 50% reduction in memory usage while maintaining computational fidelity. The system improvements lead to enhanced memory capacity, which enables better execution of extensive quantum-inspired simulations.

The study presents the integration of the C-Class processor, an RTL implementation of the RISC-V specification and a key member of the SHAKTI processor family [2]. The C-Class processor functions as a suitable platform which combines multiple instruction set architectures (ISAs) with high configurability to enable advanced computational model embedding. Its open-source nature combined with its adaptable design and optimization features for various computational tasks makes it a better choice for quantum-inspired architecture development. The Functional simulation of the Verilog-based design was performed using ModelSim SE 2020.1. The synthesis and RTL analysis were carried out using Intel Quartus Prime Lite Edition 21.1. The post-synthesis results showed that the QAT coprocessor utilized approximately 40% of the available logic elements and 60% of embedded memory blocks which was tested on Intel Cyclone IV FPGA platform. The design reached its highest operating frequency at 58 MHz which proved the system's ability to function effectively with mid-range FPGA systems. The study solved major obstacles of high memory usage and scalability issues by combining QAT's memory-optimized quantum-inspired processing with C-Class processor. The integrated system shows how quantum-inspired designs can function in traditional computing systems which will further enable new computational methods to solve problems in scientific and industrial fields.

The Array of Bits (AoB) is a fundamental data structure central to the quantum-inspired computations in this study. An AoB represents a large bit-vector encoding multiple entangled states or superpositions, with size typically expressed as  $2^E$  for an E-way superposition. This structure enables classical hardware to simulate quantum phenomena such as superposition and entanglement by applying bitwise SIMD operations. Prior work by Dietz has explored AoB representations, establishing their utility in efficiently modeling quantum-like behavior in classical systems. This paper builds upon these foundations by integrating AoB-based quantum-inspired computation within a hybrid RISC-V and QAT architecture.

## 2 Related Works

Quantum-inspired algorithms are executed on classical machines to accelerate optimization and learning without using actual qubits [3]. In practice, this includes specialized analog hardware such as Ising machines and digital accelerators. Fujitsu's Digital Annealer, a classical CMOS device which emulates quantum annealing with fully-connected bits, uses massive parallel updates to solve optimization problems beyond standard hardware [4]. Likewise, analog computing approaches using continuous physical variables such as memristor-based crossbar arrays have been used to implement a quantum-inspired parallel annealing method, achieving full parallelism and significantly improved speed and energy efficiency on Max-Cut problems [5].

Apart from dedicated accelerators, many quantum-inspired methods are purely algorithmic but target classical processors which include quantum-inspired evolutionary algorithms, tensor-network algorithms for machine learning, and kernel methods that leverage superposition concepts to improve optimization and ML tasks. A recent study explored how quantum-inspired evolutionary algorithms (QIEAs) and tensor-network kernels exploit qubit-like operations on classical hardware to outperform traditional algorithms in cybersecurity and optimization benchmarks [3].

Hybrid systems that tightly integrate quantum processing units (QPUs) with classical hosts are increasingly studied to reduce overheads and maximize performance. On the hardware side, recent proposals move beyond treating quantum devices as remote cloud resources. Ramsauer and Mauerer propose a vertically integrated architecture with a Quantum Abstraction Layer (QAL) at the OS kernel, so that QPUs appear as peripheral co-processors. QAL enables real-time, low-latency scheduling and resource management between CPUs and QPUs, supporting error correction and tight feedback loops [6]. A large-scale gate-based QPU control system is implemented utilizing a multi-chassis PXIe hardware hierarchy with synchronized AWGs for pulses, and a SIMD-based quantum instruction pipeline that broadcasts gates across qubit groups in parallel [7].

In high-performance computing (HPC) environments, embedding QPUs as accelerators follows the GPU paradigm. For example, the XACC framework treats quantum hardware as an accelerator resource in the classical workflow [8]. Just as HPC nodes use GPUs with CUDA, XACC uses an intermediate representation and compiler plugins to target either quantum hardware or classical simulators transparently. This abstraction enables executing quantum circuits across heterogeneous clusters, effectively virtualizing quantum parallelism on classical resources. Parallelism is also pursued at the quantum instruction level. Existing architectures provide SIMD mechanisms for qubit operations, where a single instruction can broadcast a gate to many qubits simultaneously, reducing instruction overhead [7]. Circuit cutting, where a large quantum circuit is partitioned into smaller ones that can run separately but require extra measurements and classical recombination [7], is another approach. The trade-off in all these cases is between coherence, time, and repetition.

RISC-V ISA was extended with quantum-specific instructions to efficiently encode quantum circuits, where the analysis showed that a tailored RISC-V control processor could improve code density and increase execution efficiency for algorithms like QFT and Grover's, compared to standard ISA designs [9]. In parallel, recent proposals add specialized hardware around a RISC-V nucleus. A recent survey describes a quantum control processor (QCP) that processes classical instructions on a simple RISC-V core, while handling qubit operations via a quantum eQASM-like format [10]. The recent trends in quantum-accelerated architectures highlight that tightly integrated hybrid architectures embed QPUs as accelerators in classical compute nodes, using layered software such as compiler plugins to unify programming [11]. The trade-offs in these approaches reveal that improved throughput and integration often come with increased hardware complexity or reduced accuracy.

### 3 Experimental Setup

The Verilog-based Finite State Machine (FSM) models were simulated using ModelSim SE 2020.1 which was the primary simulation environment for functional verification. Comprehensive testbenches were developed to verify the state transitions, output responses, and timing behavior under various input conditions. Waveform analysis and simulation logs were utilized to verify the correct operation of the FSM through all its designed test cases.

The design was executed using Intel Quartus Prime Lite Edition 21.1 for synthesis purposes. Register Transfer Level (RTL) schematics were generated to verify logical correctness and assess the structural representation of the FSM after its synthesis. Quartus Prime also enabled timing analysis and provided resource utilization reports, offering insight into the efficiency and feasibility of the design for hardware implementation. The RTL Viewer and Technology Map Viewer features were particularly useful for examining the low-level logic structure generated from the Verilog code.

### 4 Methodology

The Tangled/QAT architecture leverages the Tangled processor as the host CPU, which interfaces tightly with the QAT coprocessor to perform complex computations on large datasets represented as an Array of Bits (AoB). The design of the Tangled processor is focused on maximizing efficiency in handling operations that involve entangled superpositions. The QAT coprocessor is central to this architecture, tasked with processing large 65,536-bit AoB values. These values, which can represent up to 16-way entanglement, are integral to the quantum-inspired computations that the architecture is designed to handle. The QAT coprocessor utilizes 256 registers to hold these AoB values. The core design of the SHAKTI C-class processor was developed using Bluespec SystemVerilog (BSV), an open-source high-level synthesis language that facilitates modular hardware design and generates synthesizable Verilog code for both FPGA and ASIC targets.

#### 4.1 Memory Optimization Techniques: Quantization Approaches

To mitigate the high memory usage inherent in such large data representations, the QAT coprocessor employs two key bit compression techniques: normal quantization and mid-rise quantization.

##### 4.1.1 Normal Quantization

Normal quantization, also referred to as uniform quantization, is a process that maps a wide range of input values into a smaller set of discrete levels. This mapping is achieved by dividing the entire range of possible values into equal intervals, with each interval representing a quantized value. The first step in uniform quantization involves identifying the range of input values. This range is defined by the minimum and maximum values of the signal, encompassing the full span of possible values [12]. The quantized value can be represented by the midpoint, lower boundary, or upper boundary of the interval [13]. However, the effectiveness of normal quantization is limited by its uniform intervals, which may not align well with the actual distribution of data within the AoB. This misalignment can lead to significant quantization errors especially when the data is unevenly distributed. Thus, reducing the efficiency of compression in certain scenarios. Uniform quantization process inherently has loss of information because it maps a range of values to a single

quantized level. The loss is irreversible and can be significant in applications requiring high precision.

#### *4.1.2 Mid-Rise Quantization*

Mid-rise quantization is an improved version of uniform quantization which uses midpoints of intervals as its quantization levels instead of using their endpoint positions. The method technique is particularly advantageous because it provides a closer approximation of the original data by focusing on variations near the midpoint of each interval. Mid-rise quantization achieves better accuracy in representing data from its original dataset through its method of handling data which exists in specific range clusters. The process of compressing Array of Bits (AoB) data relies on mid-rise quantization as its essential component. The method uses midpoint-based quantization to create data encoding which decreases bit width while maintaining data accuracy. The method achieves significant benefits when data distribution matches midpoint-centered intervals as it results in higher data preservation during compression.

Mid-rise quantization provides multiple benefits which exceed standard uniform quantization as it achieves better performance in reducing quantization errors. The likelihood of large deviations between the original and quantized values reduces as it captures the central tendencies of the data more effectively [14]. This makes it a more accurate and efficient method for compressing AoB data, particularly in scenarios where preserving data fidelity is critical. The experimental results show that mid-rise quantization outperforms regular quantization by providing a better balance between compression efficiency and data accuracy.

This approach not only enhances the precision of the compressed data but also optimizes memory usage making it a preferred choice in applications requiring high data integrity. The framework's ability to minimize errors while compressing large-scale bit arrays like those in the Tangled/QAT architecture ensures that computational processes remain robust when operating under memory constraints.

## **4.2 Integrating the Bit Compressed QAT Architecture with C-Class SHAKTI Processor**

With the increasing demand for computing, the integration of quantum-inspired accelerators with classical processors is vital to enhance computational efficiency. The C-Class SHAKTI processor which an implementation of the RISC-V ISA serves as a flexible and scalable platform for such integration. By embedding the bit-compressed QAT architecture within this processor, quantum-like computation is enabled to complement classical processing. Figure 1 shows the integrated architecture of C-Class SHAKTI processor with QAT coprocessor. The quantum coprocessor enables the Shakti processor to tackle problems that are practically impossible for classical processors, including problems in cryptography, large-scale optimization, and complex simulations in material science or drug discovery [15]. This integration allows for optimized data transfer and handling, as the QAT coprocessor manages the complexities of quantum data formats and memory management internally, ensuring efficient and seamless communication with the Shakti processor [16]. By independently handling quantum specific instructions, the QAT coprocessor reduces latency and enables both processors to operate concurrently on their respective tasks, maximizing parallel processing and overall system throughput [17].



- IDLE: Set done  $\leftarrow$  0; if start, store data\_in in data\_reg
- QUANTIZE: compute quantized\_data  $\leftarrow$  data\_reg / STEP\_SIZE
- DEQUANTIZE: compute dequantized\_data  $\leftarrow$  quantized\_data  $\times$  STEP\_SIZE
- DONE: Set done  $\leftarrow$  1

#### 4. Return dequantized\_data, done

The implemented quantization process operates on 16-bit input data, reducing its bit-width to an 8-bit representation. The system functions as a state machine with distinct states for idle, quantization, dequantization, and completion. Initially, the input data is stored in a register and processed by the quantization block, which compresses it into an 8-bit format, improving storage and transmission efficiency. The quantized data is subsequently output through a designated register. The system also includes a dequantization process that reconstructs the 8-bit data back to its original 16-bit format for fidelity evaluation. Control logic, including multiplexers and state registers, ensures correct operation sequencing, all synchronized by a clock signal. The done signal indicates process completion, signalling that the quantized or dequantized data is ready for further processing.

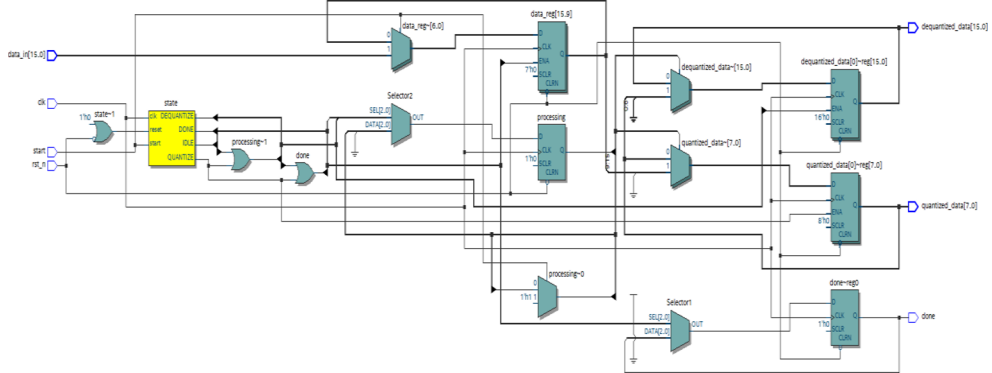


Fig. 2. RTL Schematic of normal quantization process.

#### Algorithm 2: Mid-Rise Quantization

##### Input:

- clk: Clock signal
- rst\_n: Active-low reset signal
- start: Start signal
- data\_in: 16-bit input data
- STEP\_SIZE: Step size for quantization

##### Output:

- quantized\_data: 8-bit quantized output
- dequantized\_data: 16-bit reconstructed output
- done: Completion flag

##### Initialize State Machine:

if rst\_n = 0 then state  $\leftarrow$  IDLE; else state  $\leftarrow$  next\_state

##### Determine Next State:

if state == IDLE and start == 1 then next\_state  $\leftarrow$  QUANTIZE  
 else if state == QUANTIZE then next\_state  $\leftarrow$  DEQUANTIZE  
 else if state == DEQUANTIZE then next\_state  $\leftarrow$  DONE  
 else if state == DONE then next\_state  $\leftarrow$  IDLE

##### Process Data:

if rst\_n = 0: quantized\_data  $\leftarrow$  0, dequantized\_data  $\leftarrow$  0, done  $\leftarrow$  0  
 IDLE: done  $\leftarrow$  0; if start == 1 then data\_reg  $\leftarrow$  data\_in, processing  $\leftarrow$  1

QUANTIZE:  $\text{quantized\_data} \leftarrow \text{floor}((\text{data\_reg} / \text{STEP\_SIZE}) + 0.5)$   
 DEQUANTIZE:  $\text{dequantized\_data} \leftarrow \text{quantized\_data} \times \text{STEP\_SIZE}$ ,  $\text{processing} \leftarrow 0$   
 DONE:  $\text{done} \leftarrow 1$   
**return quantized\_data, dequantized\_data, done**

This algorithm shows the structured approach to mid-rise quantization using a finite state machine (FSM). It efficiently processes a 16-bit input, compresses it into an 8-bit quantized value, and reconstructs it back to 16-bit resolution. The FSM transitions through four states: (1) IDLE: Waits for the start signal. (2) QUANTIZE: Divides the input by STEP\_SIZE, rounding to the nearest integer. (3) DEQUANTIZE: Multiplies the quantized value by STEP\_SIZE to reconstruct the signal. (4) DONE: Signals completion.

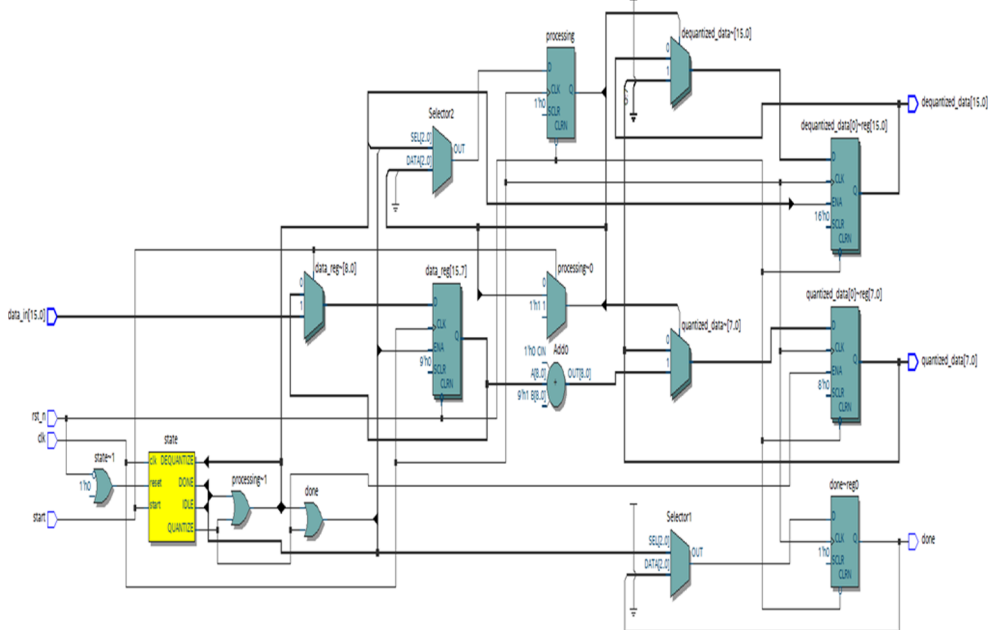


Fig. 3. RTL Schematic of mid-rise quantization process.

#### 4.2.1 Procedural Implementation Flow

The implementation of the Quantum-like Accelerator for Tangled (QAT) system focuses on integrating the QAT coprocessor with the SHAKTI C-class processor to enable efficient quantum-inspired computations. This flow comprises the design of a custom Instruction Set Architecture (ISA), modifications to the decode stage, and interaction with the QAT coprocessor.

The process begins by extending the SHAKTI processor's ISA to include new instructions specifically tailored for tasks optimized for the QAT coprocessor such as manipulating large Array of Bits (AoB) structures for quantum-like operations. These instructions are encoded with unique opcodes to differentiate them from standard SHAKTI instructions, enabling the processor to invoke specialized QAT operations.

The decode stage of the SHAKTI processor is modified to recognize and handle these new instructions. The decode logic is updated to detect the custom opcodes associated with the QAT coprocessor, ensuring accurate identification during instruction fetching. The system executes recognized instructions by sending them to the QAT coprocessor which implements the instructions without using the main execution pipeline of SHAKTI. The

SHAKTI processor uses this forwarding method because it enables the system to process general-purpose instructions while avoiding main core bottlenecks which leads to decreased system processing efficiency.

The QAT coprocessor executes the forwarded instructions which define specialized operations through the custom ISA. The operations require the coprocessor to process 65,536-bit AoB values which represent 16-way entangled states through its optimized hardware. The results of these computations are stored in the QAT's dedicated memory, which consists of 256 registers designed to hold these extensive bit arrays. The Internal storage ensures that results are readily available for subsequent retrieval and maintains isolation from the SHAKTI's register file to streamline data management.

### 4.3 Results

The results section discusses the efficiency and effectiveness of the optimization technique in the proposed framework. Input data at 16-bit resolution is quantized to 8 bits where values are mapped to discrete levels. The process compresses the signal but introduces precision loss, leading to a dequantized output that differs from the original. The trade-off in normal quantization balances compression efficiency against accuracy. Figure 4 illustrates the waveform representing the timing and values of the normal quantization process.

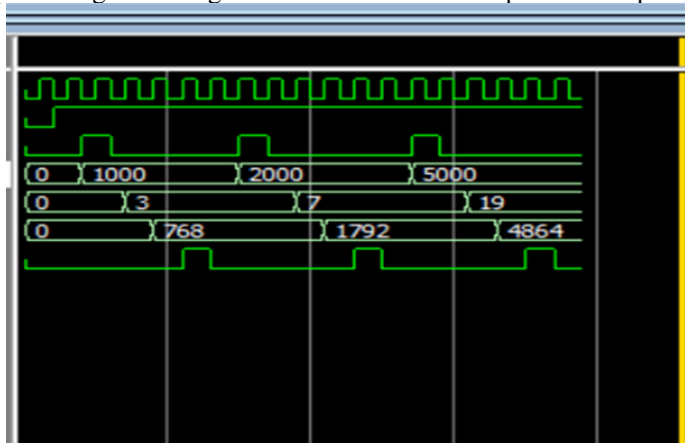


Fig. 4. Waveform representing the timing and values of normal quantization process.

The mid-rise quantization waveform demonstrates synchronization between the start signal and the output. The original input can be reconstructed from dequantized values which show that the quantization method works effectively. While mid-rise quantization introduces some dequantization error that is typically around 9–10%, this error becomes negligible when dealing with significantly larger data sizes, such as [65535:0]. Figure 5 presents the waveform of mid-rise quantization, where symmetric levels around a midpoint minimize quantization error.

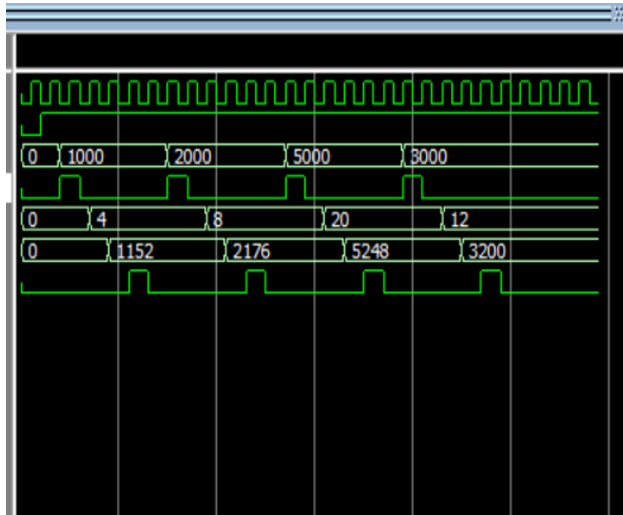


Fig. 5. Mid-rise quantization with symmetric levels around a midpoint to minimize quantization error.

The impact of quantization error diminishes when compressing large AoB values from 65,536 bits to 32,768 bits. While mid-rise quantization technique is not entirely lossless, it strikes a practical balance by delivering highly efficient compression without compromising the integrity of the original data to any significant degree. The technique permits efficient compression of extensive bit arrays because its dequantization error decreases when data volume rises. Tables 1 and 2 present the latency performance and memory footprint of the proposed framework.

**Table 1.** Latency performance of the proposed method.

Method	Latency ( $\mu$ s, 100 MHz)	Throughput (ops/sec)
Normal Quantization	3	$3.33 \times 10^7$
Mid-Rise Quantization	4	$2.50 \times 10^7$

**Table 2.** Register usage and memory footprint of the proposed framework.

Metric	Pre-Compression	Post-Compression	Reduction
Bits per AoB	65,536	32,768	50%
Total Bits (256 Registers)	16,777,216	8,388,608	50%
Memory (MB)	16.78	8.39	50%

Applications needing high-speed data processing can benefit greatly from the module's low latency, which is further confirmed by the timing intervals between the start and completion signals. These findings demonstrate how mid-rise quantization can be used to maximize memory utilization while preserving data integrity in quantum-inspired computing architectures.

The results indicate that the bit compression techniques utilized in QAT substantially reduce the memory footprint, leading to enhanced processing speeds and reduced latency when interfaced with the Shakti processor. This integration facilitates scalable simulations of complex quantum systems, offering a practical approach to quantum-like computations without necessitating a full-scale quantum computer.

## 4.4 Discussion

The proposed architecture envisions a hybrid quantum-inspired classical processor in which a conventional RISC-V core (SHAKTI C-class) is tightly coupled with a quantum-like accelerator (QAT). The SHAKTI C-class processor designed for mid-range embedded and IoT applications serves as the host CPU controlling the system. The QAT unit, by contrast, implements the parallel bit pattern (PBP) model of quantum-like computation. In PBP, an entangled pbit is represented as an Array-of-Bits (AoB) of size  $2^E$  for an E-way superposition. Operations on these pbit states are carried out by conventional SIMD bitwise operations, so that superposition and entanglement are simulated in a purely classical logic array.

Importantly, the QAT is conceived as a co-processor, where its instructions are fetched and decoded by the host CPU and integrated into the pipeline. In practice, this means that a program running on Shakti can issue QAT instructions to create or manipulate entangled bit patterns, interleaving classical and quantum-like computation on the same chip. The hybrid design thus merges cutting-edge academic ideas of quantum-inspired computing with a pragmatic, open RISC-V platform.

By embedding PBP support in hardware, the SHAKTI QAT architecture enables quantum-inspired classical algorithms to run at the bit-vector gate level rather than in software, potentially realizing the efficiency gains predicted by compiler-level optimization. In effect, entangled superpositions become bit-pattern operations on an AoB vector, which can be executed with ordinary bitwise SIMD logic. This means one can apply aggressive bit-level compiler optimizations analogous to gate-level reductions to reduce the total number of bit-operations and overall power consumption.

Critical procedural steps significantly influencing the success of the method include the efficient synchronization between the SHAKTI host CPU and the QAT co-processor, precise instruction encoding for QAT operations, and effective memory management to handle large AoB arrays without bottlenecks. Ensuring that the host CPU correctly interleaves classical and quantum-like instructions without pipeline stalls is essential for maintaining performance. Additionally, compiler support that translates high-level quantum-inspired algorithms into optimized QAT instructions is a pivotal factor for realizing practical speedups.

The proposed architecture contains its limitations. The integrated hybrid processor faces constraints when representing very large superposition spaces that could quickly exhaust memory or degrade performance. The study is also largely architectural and theoretical as it has not been fabricated yet, lacking empirical data on real performance, latency, and power. The current QAT designs exist only as Verilog models rather than tape-outs; therefore, issues like timing, area, and energy are unmeasured. To test and validate the proposed architecture, hardware prototyping on FPGA hardware would allow direct measurement of performance and resource usage. Formal verification tools could also be applied to the co-processor design to ensure that entanglement manipulations obey the intended semantics.

Challenges during implementation may arise from timing mismatches between the host CPU and QAT, excessive memory demands from large AoB sizes, and instruction decoding complexity. Troubleshooting requires iterative testing with FPGA prototypes, thorough debugging of synchronization mechanisms, and profiling to identify and alleviate performance bottlenecks. Adjusting AoB size and employing bit compression techniques may be necessary to balance between computational fidelity and resource constraints. Hardware accelerators for post-quantum cryptography (PQC) are currently under development [19]. One recent design uses a Keccak-based core to offload expensive PQC algorithms and improve throughput. Similarly, the quantum co-processor could speed up classical cryptographic primitives and explore quantum-resilient algorithms in hardware [20]. Prior work suggests that gate-level compiler optimizations can dramatically reduce bit-

operations for quantum-like algorithms [21]. Hence, adapting compiler backends to emit QAT-friendly instructions would unlock that potential.

In terms of usability, integrating the hybrid QAT architecture into existing embedded systems and IoT devices may face practical constraints such as limited on-chip memory, power consumption limits, and compatibility with existing software ecosystems. Despite these challenges, the open RISC-V foundation provides flexibility for customization and gradual adoption, making this approach promising for incremental integration into next-generation computational platforms. Future work can proceed in the direction of leveraging the integration of QAT with the Shakti processor, which shows strong potential for accelerating AI model training, especially in domains where quantum-inspired algorithms surpass classical approaches. The low-power design of the C-Class Shakti processor, combined with the quantum-like processing capabilities of QAT, suggests a viable solution for embedding advanced functionalities in IoT and edge computing devices [22].

The authors gratefully acknowledge the infrastructure facilities, research support, and open access funding provided by Vellore Institute of Technology (VIT).

Disclosures: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

**Author Contributions:** **Moulisvaran U:** Methodology, Software, Writing-review & editing, Conceptualization. **Pramoth K.P:** Methodology, Software. **Rithika Naveencharan:** Methodology, Software, Writing-Original draft, review & editing. **Sheena Christabel Pravin:** Formal analysis, Project administration. **Reena Monica P:** Project administration, Conceptualization, Writing-review & editing, Methodology.

## References

1. H. Dietz, Tangled: A conventional processor integrating a quantum-inspired coprocessor. 50th Int. Conf. Parallel Processing Workshop (ICPP Workshops '21) (2021).
2. N. Gala, A. Menon, R. Bodduna, G. S. Madhusudan, V. Kamakoti, SHAKTI processors: An open-source hardware initiative. 29th International Conference on VLSI Design and 15th International Conference on Embedded Systems (VLSID), 461–466 (2016). <https://doi.org/10.1109/VLSID.2016.130>
3. G. Iovane, Quantum-inspired algorithms and perspectives for optimization. *Electronics*. **14** (14), 2839 (2025). <https://doi.org/10.3390/electronics14142839>
4. M. Schönberger, I. Trummer, W. Mauerer, Quantum-inspired digital annealing for join ordering. *Proc. VLDB Endow.* **17** (3), 511–524 (2023). <https://doi.org/10.14778/3632093.3632112>
5. M. Jiang et al., Efficient combinatorial optimization by quantum-inspired parallel annealing in analogue memristor crossbar. *Nat. Commun.* **14**, 5927 (2023). <https://doi.org/10.1038/s41467-023-41647-2>
6. R. Ramsauer, W. Mauerer, Towards system-level quantum-accelerator integration. arXiv:2507.19212 (2025).
7. F. Zhang et al., A classical architecture for digital quantum computers. *ACM Trans. Quantum Comput.* **5** (1), 3 (2024). <https://doi.org/10.1145/3626199>
8. D. Claudino, D. I. Lyakh, A. J. McCaskey, Parallel quantum computing simulations via quantum accelerator platform virtualization. *Future Gener. Comput. Syst.* **160**, 264–273 (2024). <https://doi.org/10.1016/j.future.2024.06.007>

9. A. Butko et al., Understanding quantum control processor capabilities and limitations through circuit characterization. 2020 Int. Conf. Rebooting Computing (ICRC), 66–75 (2020). <https://doi.org/10.1109/ICRC2020.2020.00011>
10. P. Döbler, M. S. Jattana, A survey on integrating quantum computers into high performance computing systems. arXiv:2507.03540 (2025).
11. L. Belostotski, A. Uddin, A. Madanayake, S. Mandal, A survey of analog computing for domain-specific accelerators. *Electronics*. **14** (16), 3159 (2025). <https://doi.org/10.3390/electronics14163159>
12. A. Gersho, R. M. Gray, Vector quantization and signal compression. Springer (1992).
13. A. V. Oppenheim, R. W. Schaffer, J. R. Buck, Discrete-time signal processing (2nd ed.). Prentice-Hall, Inc. (1999).
14. R. M. Gray, D. L. Neuhoff, Quantization. *IEEE Trans. Inf. Theory*. **44** (6), 2325–2383 (1998). <https://doi.org/10.1109/18.720541>
15. J. Preskill, Quantum computing in the NISQ era and beyond. *Quantum*. **2**, 79 (2018). <https://doi.org/10.22331/q-2018-08-06-79>
16. S. Martiel, T. Ayrat, C. Allouche, Benchmarking quantum coprocessors in an application-centric, hardware-agnostic, and scalable way. *IEEE Trans. Quantum Eng.* **2**, 1–11 (2021). <https://doi.org/10.1109/TQE.2021.3090207>
17. A. Kay, Coprocessors for quantum devices. *Phys. Rev. A*. **97**, 032316 (2018). <https://doi.org/10.1103/PhysRevA.97.032316>
18. L. Riesebois et al., Quantum accelerated computer architectures. 2019 IEEE Int. Symp. Circuits Syst. (ISCAS), 1–4 (2019). <https://doi.org/10.1109/ISCAS.2019.8702488>
19. J. W. Bos, J. Renes, C. van Vredendaal, Post-quantum cryptography with contemporary co-processors: Beyond Kronecker, Schönhage-Strassen & Nussbaumer. in 31st USENIX Security Symp. (USENIX Security '22), 3683–3697 (2022).
20. D. Moody, A. Robinson, Cryptographic standards in the post-quantum era. *IEEE Secur. Priv.* **20** (6), 66–72 (2022). <https://doi.org/10.1109/MSEC.2022.3202589>
21. T. Häner, D. S. Steiger, K. M. Svore, M. Troyer, A software methodology for compiling quantum programs. *Quantum Sci. Technol.* **3** (2), 020501 (2018). <https://doi.org/10.1088/2058-9565/aaa5cc>
22. A. G. Putrada, N. Alamsyah, M. N. Fauzan, S. Prabowo, I. D. Oktaviani, Quids: A novel edge-based botnet detection with quantization for IoT device pairing. *Indones. J. Comput. (Indo-JC)*. **8** (3), 29–41 (2023). <https://doi.org/10.34818/INDOJC.2023.8.3.878>