

# HPQEA: A Scalable and High-Performance Quantum Emulator with High-Bandwidth Memory for Diverse Algorithms Support

Tran Van Duy<sup>1</sup>, Tuan Hai Vu<sup>1</sup>, Vu Trung Duong Le<sup>1</sup>, Hoai Luan Pham<sup>1</sup>, and Yasuhiko Nakashima<sup>1</sup>

<sup>1</sup> Nara Institute of Science and Technology, 8916-5 Takayama-cho, Ikoma, Nara 630-0192, Japan.

Email: tran.van\_duy.tu0@naist.ac.jp

**Abstract**—In recent years, there has been a growing interest in the development of quantum emulation. However, existing studies often struggle to achieve broad applicability, high performance, and efficient resource and memory utilization. To address these challenges, we provide HPQEA, a quantum emulator based on the state-vector emulation approach. HPQEA includes three main features: a high-performance computing core, an optimized controlled-NOT gate computation strategy, and effective utilization of high-bandwidth memory. Verification and evaluation on the Alveo U280 board show that HPQEA can emulate quantum circuits with up to 30 qubits while maintaining high fidelity and low mean square error. It outperforms comparable FPGA-based systems by producing faster execution, supporting a wider range of algorithms, and requiring low hardware resources. Furthermore, it exceeds the Nvidia A100 in normalized gate speed for systems with up to 20 qubits. These results demonstrate the scalability and efficiency of HPQEA as a platform for emulating quantum algorithms.

**Index Terms**—quantum emulator, state vector, FPGA, SoC, high bandwidth memory, parallel computing.

## I. INTRODUCTION

Recently, quantum computing [1] has emerged as a promising solution to complex problems, challenging the limits of Moore’s law [2]. The major technology companies, including IBM, Google, Microsoft, and Fujitsu [3], are actively developing quantum computers. However, the extreme sensitivity of quantum states confines current systems to controlled laboratory environments. In addition, quantum algorithms play a vital role in exploring applications in fields such as data fitting [4], optimization [5], and machine learning [6]. However, limited access to practical hardware hinders their development and testing, motivating the use of quantum emulation approaches.

In contrast to actual quantum computers, a quantum bit (qubit) can be emulated on classical computers. Current emulation techniques, such as state-vector-based simulations, are computationally and memory-intensive as the number of qubits (#Qubits) increases. High-performance computing (HPC) systems, such as DGX A100 and H100 [7], can address this issue but introduce extremely high power consumption. In addition, software improvements [8] can eliminate unnecessary processing on a small-scale #Qubits [9]. Other methods, such as tensor networks [10], stabilizer formalism [10], and density matrices [11], provide polynomial or linear time computation for specific circuits, but support only limited types of circuits. Therefore, developing an emulation system with high

processing performance while minimizing hardware resources and power consumption is still necessary.

FPGAs are now widely recognized as an efficient platform for quantum emulation [12]–[19], enabling deep hardware-level optimization compared to CPUs, GPUs, and HPC. However, existing FPGA-based studies frequently lack thorough hardware optimization [12], [13], [15], support only specific algorithms such as Quantum Fourier Transform (QFT), Quantum Haar Transform (QHT), or Quantum Support Vector Machine (QSVM) [14], [17], use low-memory usage strategies [16], or are limited to small-scale systems ( $\leq 17$  qubits) [18], [19].

To address these limitations, we propose the High-Performance Quantum Emulation Accelerator (HPQEA), which utilizes the state vector-based simulation method to design a hardware architecture that enables deeply optimized computation and storage, high performance with low cost, broad algorithm compatibility, and scalability up to 30 qubits. To gain these goals, HPQEA includes:

- **High Performance Dual Processing Element Arrays (PEAs):** Accelerate computation and optimize memory usage.
- **Optimized CX Swapper:** Improve data transfer scheduling to reduce CX gate execution latency.
- **Optimized High Bandwidth Memory (HBM) Based Bulk Data Transfer:** Leverage high-bandwidth memory for large-scale data access and storage.

The AMD Alveo U280 FPGA board was used [20] to validate and evaluate HPQEA. The rest of the paper is organized as follows: Section II provides background knowledge; Section III explores the hardware architecture of HPQEA; Section IV presents implementation and evaluation; and Section V concludes the study. The detailed implementation can be found at [https://github.com/d2k-daniel/HPQEA\\_2\\_Core](https://github.com/d2k-daniel/HPQEA_2_Core).

## II. BACKGROUND KNOWLEDGE

### A. Quantum computing

Quantum computing leverages fundamental principles of quantum mechanics to process information in ways fundamentally different from classical computing, such as superposition, entanglement, and interference. Generally, the state of an

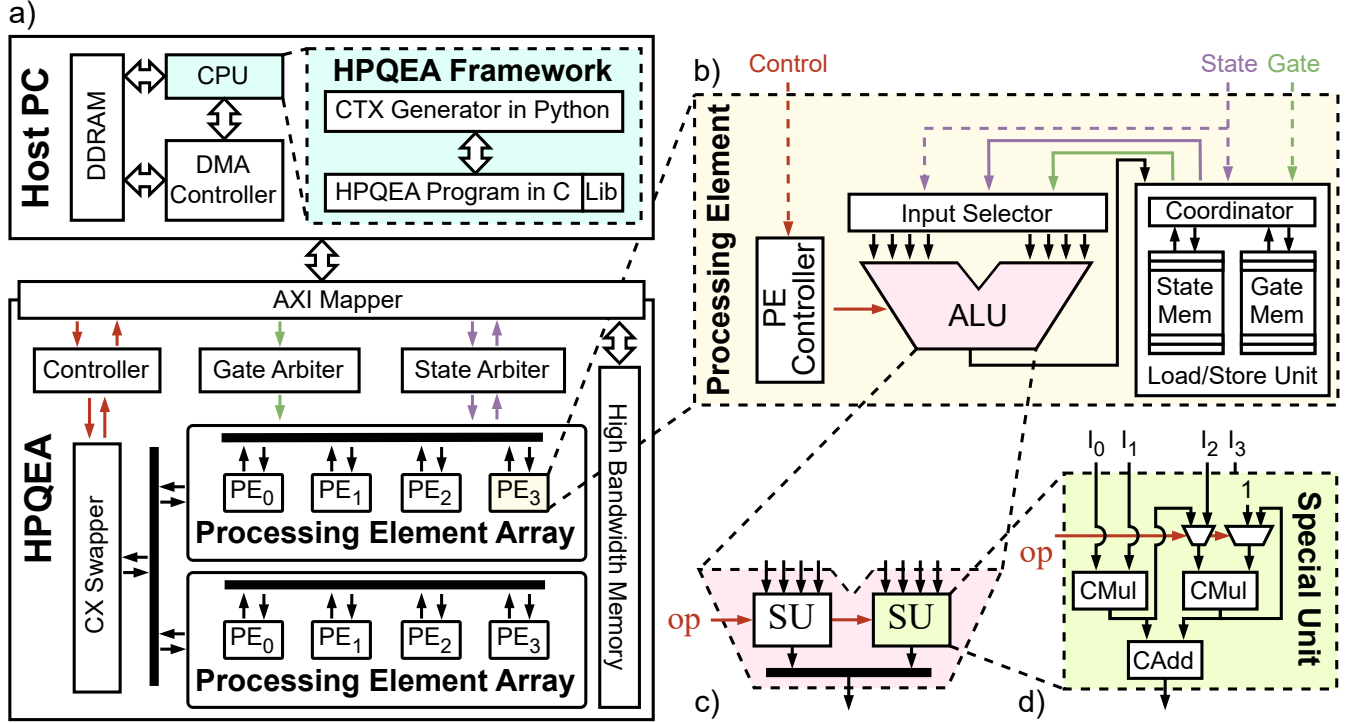


Figure 1: The details of the HPQEA: (a) The overview of the HPQEA system, (b) (Inset) Processing Element, (c) (Inset) Arithmetic Logic Unit, and (d) (Inset) Special Unit.

$n$ -qubit system at a given time is represented by a  $2^n$ -dimensional complex vector  $|\psi\rangle = [\alpha_0 \ \alpha_1 \ \dots \ \alpha_{2^n-1}]$  where  $\{\alpha_j\}$  is called amplitudes.

In addition to the quantum state, quantum circuits update the state vector using quantum gates, which are represented as unitary matrices. In this work, the universal gate set Clifford +  $R_i$  with  $i \in \{x, y, z\}$  was used following the Gottesman-Knill theorem [21]. That means any quantum circuit can be presented by the set  $\{H, S, R_x(\cdot), R_y(\cdot), R_z(\cdot), CX\}$ . A quantum circuit can be broken down into low-level gates or built up into high-level gates. As an example, the control rotation  $CR_x(\cdot)$  can be decomposed into  $\{S, CX, R_x(\cdot), R_y(\cdot), R_z(\cdot)\}$  following a transpiler [22]. A transpiler can recover the original state by  $|\psi_{\text{tran}}\rangle = e^{i\phi}|\psi_{\text{origin}}\rangle$  where  $\phi$  is the global phase born from the transpilation process.

### B. Quantum simulation algorithm

A quantum circuit is theoretically simulated by applying quantum gates to an initial state  $|\psi^{(0)}\rangle$  sequentially. The result state  $|\psi^{(m)}\rangle$  after applying a quantum operator (quantum circuit) with  $m$  gates is expressed as  $|\psi^{(m)}\rangle = \mathcal{U}|\psi^{(0)}\rangle$ . The quantum operator  $\mathcal{U}$  can be decomposed into a series of matrix multiplications between  $p$  smaller sub-operators  $u$ , as well as a sequence of  $g$  quantum gate operations (single-qubit and two-qubit gates) from each sub-operator  $u$ . These gates can be applied gate by gate to the initial state in sequence, resulting in the same final state as shown in Eq. 1.

$$\mathcal{U} \equiv \prod_{j=1}^p u_j = \prod_{j=1}^p \left( \bigotimes_{k=1}^q g_k \right), \quad |\psi^{(j+1)}\rangle = u_j |\psi^{(j)}\rangle \quad (1)$$

## III. PROPOSED HARDWARE ARCHITECTURE

### A. Overview

Figure 1 (a-d) shows the overall architecture of the proposed HPQEA system, consisting of two main parts: the host PC (software) and the HPQEA (hardware). On the host PC, the HPQEA framework generates the context data and control signals for the hardware. In detail, the initial quantum state and the quantum circuit context are created using the Python-based Qiskit program, processed by a C program, and transferred to the HPQEA via Direct Memory Access (DMA) over a 256-bit AXI bus for high-throughput communication. The HPQEA hardware includes six main components:

- **AXI Mapper:** Connects the external AXI bus to the internal memory bus.
- **High Bandwidth Memory:** Provides large storage and high-throughput data access.
- **Controller:** Manages all hardware modules based on host PC commands.
- **Gate Arbiter and State Arbiter:** Distribute quantum state data and gate instructions from the host PC and HBM to internal memory.
- **CX Swapper:** Executes two-qubit CNOT operations.
- **Dual Processing Element Arrays (PEAs):** Update the state vector for single-qubit gate operations.

The following sections provide a detailed discussion of the hardware architecture, including high-performance dual PEAs, optimized CX Swapper, and optimized HBM-based bulk data transfer.

### B. High-Performance Dual PEAs

Computational performance is critical in quantum simulation, as processing demand grows rapidly with #Qubits. However, as noted in Section I, recent studies often lack deep hardware optimization, broad algorithm support, and efficient resource use. To address these limitations, the proposed HPQEA employs two parallel PEAs, adapted from [19], effectively doubling the computational ability while preserving wide support for algorithm compatibility and other advantages.

Each PE (Figure 1(b)) contains an Arithmetic Logic Unit (ALU) and a Load/Store Unit (LSU). The ALU (Figure 1(c)) integrates two Special Units (SUs) (Figure 1(d)) optimized for BRAM bandwidth and computation. These SUs are reconfigurable for sparse or dense single-qubit gate computation [19], using two complex multipliers, one complex adder, and an operation signal (*op*). The LSU overwrites updated quantum states in the same location as the previous quantum states, avoiding extra memory usage. Moreover, a full  $n \times n$  quantum gate matrix is replaced with a  $2 \times 2$  [19], reducing resource cost. Unlike QEA [19], HPQEA utilizes and connects two PEs for data sharing, thereby preventing conflicts and enabling true parallelism.

Beyond computation, efficient data usage and storage are also important for accuracy and simplified control logic. Following the evaluation in [19], HPQEA adopts a 32-bit fixed-point format (2 integer bits, 30 fractional bits), providing high fidelity with low mean square error (MSE). To adapt to the dual-PEA structure and support any #Qubits, the state vector is divided into two parts, one for each PEA, and each part is further divided into four segments for the four PEs in each PEA. The relative positions of state values are preserved without reordering in each PE. This organization allows each PE to process its assigned data independently under two access modes: (1) loading from its own LSU or (2) combining its data and data from another PE. In the second scenario, only the current loaded values of PEs are exchanged over a shared bus, eliminating redundant memory access and avoiding conflicts. This is because the data loaded from the current PE will be the value needed from the other PE, thanks to the proposed data arrangement. This method simplifies data flow control, prevents access conflicts, and improves overall system efficiency.

### C. Optimized CX Swapper

Contrary to other quantum gates, the CX gate just swaps values in the state vector, without performing arithmetic computations [19], [23]. However, the design in [19] is inefficient for data scheduling, as illustrated in Fig. 2(a). In particular, one cycle is used to compute swap indices, two cycles to load data from the State Memory into the PEs, and two cycles to save the swapped data back. For an  $n$ -qubit system, this

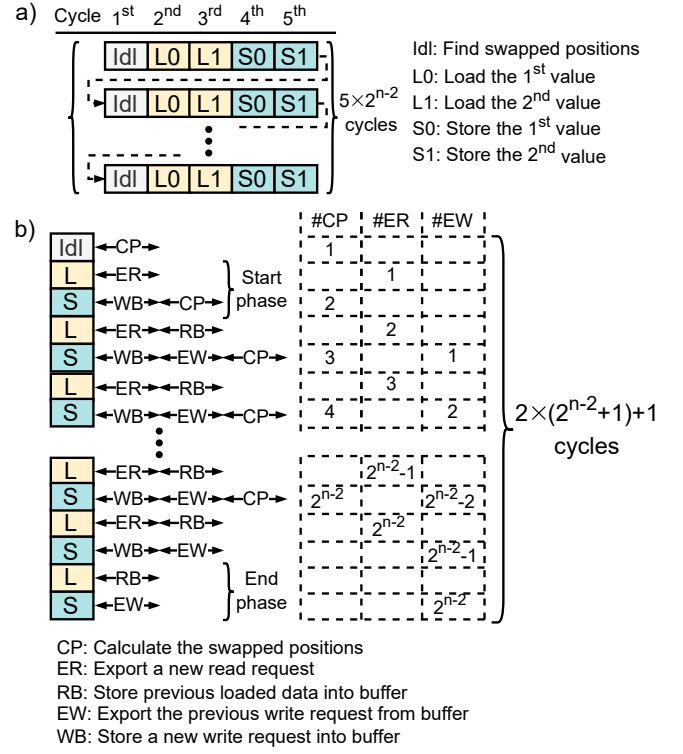


Figure 2: CX working flow: (a) Old version, (b) New version

results in  $2^{n-2} \times 5$  total cycles, where  $2^{n-2}$  represents the number of data pairs swapped. The sequential structure of the process further affects BRAM bandwidth utilization and overall efficiency.

To address these limitations, the proposed HPQEA implements an improved data schedule, illustrated in Fig. 2(b). The operation is organized into three stages: IDLE, LOAD, and STORE. The IDLE stage runs once at the start to compute swap positions for the first paired data. In the LOAD stage, read requests are issued to the State Memory using the precomputed indices to load the swapped data values. During the STORE stage, retrieved data values are written back while the positions for the next swapped values are recalculated. The LOAD and STORE stages repeat until all swaps are completed. Two additional control phases, Start and End, were inserted to ensure correct execution: Start initializes read requests, and End guarantees completion of all write operations. This approach reduces execution time, requiring only  $2 \times (2^{n-2} + 1) + 1$  cycles, achieving substantial speedup over the baseline sequential design.

### D. Optimized HBM-Based Bulk Data Transfer

For small-scale systems, such as those with 10 qubits, data transfer is not a critical bottleneck since the state vector can be stored entirely in internal BRAMs. This significantly reduces transfer latency compared to designs that rely on conventional external memory, such as DDR SDRAM. However, as the system scales to higher #Qubits counts as 25, BRAM capacity becomes insufficient, while DDR SDRAM introduces substan-

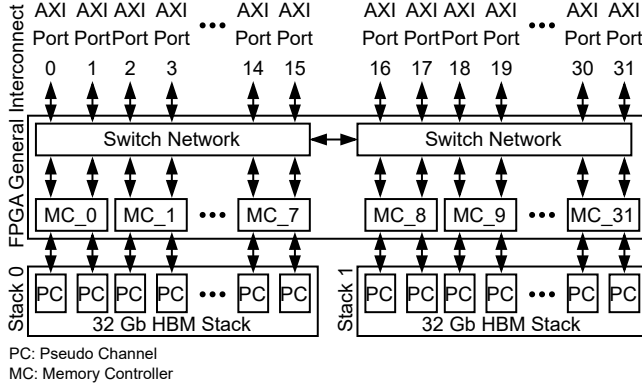


Figure 3: High Bandwidth Memory of the Alveo U280 FPGA board.

tial delays due to limited bandwidth. In this context, HBM offers a more suitable solution. As shown in Fig. 3, HBM provides more interface ports than DDR SDRAM and is built on more advanced manufacturing technology. Specifically, the Alveo U280 FPGA integrates two HBM stacks, each with a total of 16 AXI ports, offering a combined 32 GB of storage. Then, the high number of AXI ports enables HBM to deliver significantly greater bandwidth and lower latency, making it ideal for large-scale quantum simulations.

With the use of HBM, HPQEA can overcome the scalability limits of the original QEA [19], which could handle only up to 17 qubits. Although HBM outperforms DDR SDRAM, it still incurs some overhead for initiating and finalizing data transactions via the AXI interface. For systems with fewer than 20 qubits and more than 20 qubits, HPQEA employs a hybrid memory architecture to maximize efficiency. In the first scenario, only internal BRAMs are utilized, thereby eliminating the overhead of external transfers. In the second scenario, the design automatically switches to using HBM. The 19-qubit barrier reflects the resource limitations of the Alveo U280 FPGA used in HPQEA implementations. Then, HPQEA can reduce transfer overhead while effectively scaling any kind of quantum systems.

#### IV. VERIFICATION AND EVALUATION

##### A. Implementation and Verification

To implement and evaluate HPQEA (Fig. 1 (a)), we used the Vivado 2020.2 tool to run the synthesis and implementation on the 16 nm Alveo U280 FPGA board. Table I shows the post-implementation results, including the AXI Mapper, Controller, Gate Arbiter, State Arbiter, Dual PEAs, and CX Swapper. In detail, exclusively for HBM, HPQEA requires 43,667 lookup tables (LUTs), 22,634 flip-flops (FFs), 956.5 Block RAMs (BRAMs), and 512 digital signal processors (DSPs), yet it consumes only 0.939 W of power. Additionally, the HBM uses considerably fewer resources, but it consumes 7.4 W due to the management of a large amount of memory (8GB). To ensure the reliability of HPQEA, we also conducted numerous experiments ranging from 3 to 30 qubits, as shown in Section IV-C. In our scenario, 30 is the upper limit due to

the memory constraint of the HBM on the Alveo U280 FPGA board. The extensive experiments demonstrated that HPQEA can produce accurate results with very low MSE and high fidelity.

Table I: Post-implementation results of the HPQEA design on the AMD Alveo U280 FPGA

Design	Freq. (MHz)	Resources				Power (W)
		LUT	FF	DSP	BRAM	
AXI Mapper	250	470	843	0	0	0.01
Controller		2333	695	0	0	0.008
Gate Arbiter		567	270	28.5	0	0.011
State Arbiter		1328	3420	0	0	0.038
Dual PEAs		27059	14811	928	512	0.805
CX Swapper		11910	2595	0	0	0.067
HBM		965	860	0	4	7.4
<b>HPQEA (Total)</b>		<b>44632</b>	<b>23494</b>	<b>956.5</b>	<b>516</b>	<b>8.339</b>

##### B. Benchmarking Methods

To demonstrate the scalability, efficiency, and flexibility of HPQEA across diverse scenarios, we conducted evaluations using 19 quantum circuit templates [24], each assigned a unique circuit ID from 1 to 19. Each circuit includes various common quantum circuit topologies, including chain, alternating, all-to-all [25], and rotation-based structures, as illustrated in Fig. 4. In detail, the chain topology arranges qubits linearly, allowing interactions only between neighboring qubits. The alternating topology forms a bipartite structure, enabling interconnections between two distinct subsets of qubits. The all-to-all topology represents a fully connected graph, where every qubit can interact directly with all others. Lastly, the rotation  $i$  serves as a parameterized single-qubit operation without entanglement.

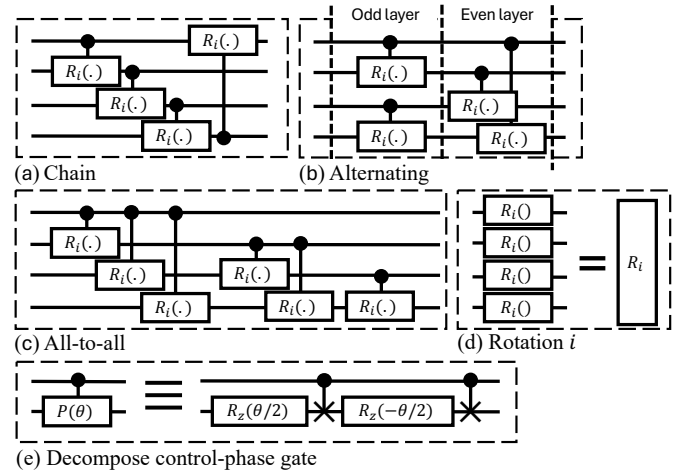


Figure 4: Example of 4-qubit topologies (a-d) and (e) Decomposing of  $CP(\theta)$  as a combination of  $CX$  and  $R_z$  gates [19].

To compare with related works, we also evaluated the HPQEA using the Quantum Fourier Transform (QFT) [26]. QFT is an essential part used in various quantum algorithms,

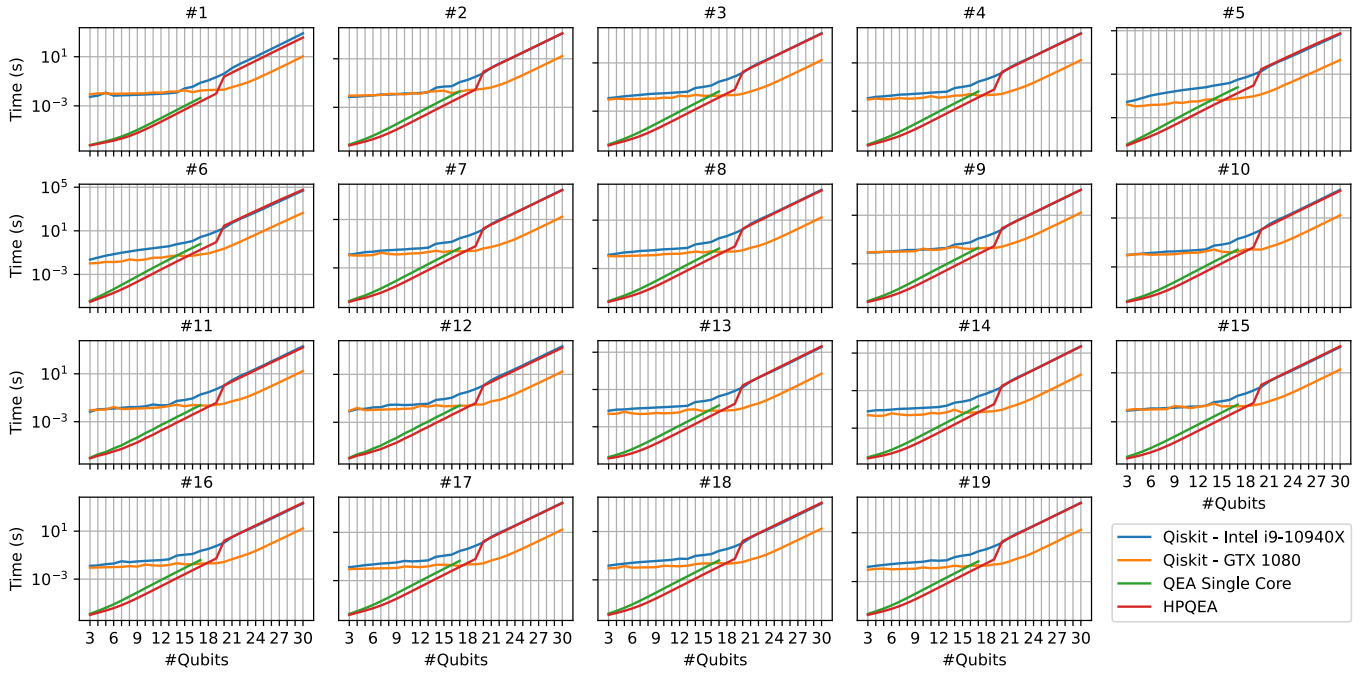


Figure 5: Comparison in execution time between HPQEA and a variety of random quantum circuits indexed from #1 to #19

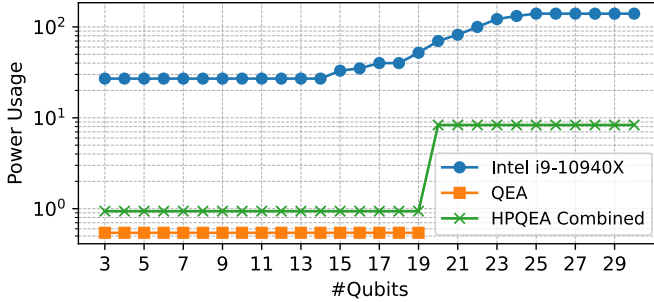


Figure 6: The average power usage comparison between HPQEA and other platforms of random quantum circuits indexed from #1 to #19

such as phase estimation, order-finding, and Shor’s factoring algorithm. In our implementation, QFT circuits were generated utilizing Hadamard ( $H$ ) gates, decomposed controlled phase ( $CP(\theta)$ ) in Fig. 4 (e), and SWAP operations using three consecutive CNOT ( $CX$ ) gates.

The evaluated metrics include memory usage, fidelity, mean square error (MSE), and execution time, clearly defined in [18]. The fidelity and MSE between two state vectors range from 0 to 1, with a fidelity of 1 or an MSE of 0 meaning that two vectors completely overlap and vice versa. The execution time on the CPU (Qiskit) is measured from constructing the circuit to receiving the final state, while the execution time of HPQEA is measured by running HPQEA with a maximum frequency of 250 MHz to calculate the final state from the initial state.

Furthermore, the evaluation considers four key metrics: memory usage, fidelity [18], mean-square error (MSE), and execution time. Fidelity and MSE are used to assess the

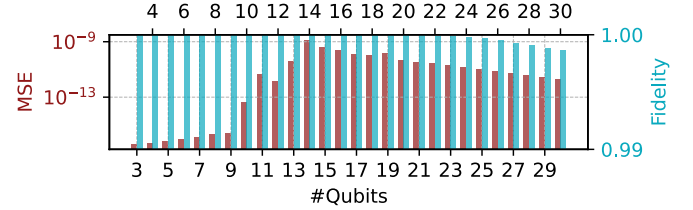


Figure 7: Average MSE and fidelity measurement between HPQEA and Qiskit for a set of templates indexed from 1 to 19.

similarity between two quantum state vectors. Both metrics are normalized between 0 and 1, where a fidelity of 1 or an MSE of 0 indicates perfect overlap between vectors, while lower fidelity or higher MSE indicates increasing divergence. Execution time is measured differently for software and hardware platforms. For the software-based evaluation (Qiskit), the total execution time includes circuit construction, simulation, and retrieval of the final state vector. Besides, the hardware execution time is measured by starting the HPQEA to calculate the final state from the initial state with a fixed maximum operating frequency of 250 MHz.

### C. Comparison with CPUs and GPUs

To validate the correctness and performance, we compared HPQEA with Intel(R) Core(TM) i9-10940X, NVIDIA GTX 1080 (using Qiskit-GPU [27]), and QEA [19] across quantum circuits, mentioned in Section IV-B, indexed from #1 to #19 in Fig. 5. The results indicate that HPQEA surpasses Intel i9-10940X CPU for systems with less than 19 qubits, with an average acceleration of  $4\times$  over QEA. HPQEA also significantly outperforms the GTX 1080 between 3 and 18



Table II: Comparative analysis in post-implementation of HPQEA and existing FPGA-based emulators on QFT's performance.

Works	Device	Freq (MHz)	Reconfig*	Precision	#Qubits	Execution time (s)	#Gates <sup>†</sup>	NGS <sup>††</sup>
[12]	AMD Xilinx Zynq-7000	100	✓	32-bit FX	6	$1.15 \times 10^{-4}$	10	$1.8 \times 10^{-7}$
[13]	Arria 10AX115N4F45E3SG	233	✓	32-bit FP	16	$1.84 \times 10^1$	528	$5.33 \times 10^{-7}$
[14]	Xilinx XCKU115	160	✗	16-bit FX	16	$2.70 \times 10^{-1}$	136	$3.03 \times 10^{-8}$
[15]	2 × Intel Stratix 10 MX2100	299	✗	32-bit FP	30	$4.47 \times 10^0$	465	$8.95 \times 10^{-12}$
[17]	Xilinx XCVU9P	233	✗	18-bit FX	16	$1.20 \times 10^{-3}$	-	-
[18]	Xilinx ZCU102	125	✓	32-bit FX	17	$2.81 \times 10^0$	721	$2.97 \times 10^{-8}$
[19]	AMD Alveo U280	250	✓	32-bit FX	17	$3.29 \times 10^{-1}$	721	$3.48 \times 10^{-9}$
<b>This work</b>	<b>AMD Alveo U280</b>	<b>250</b>	<b>✓</b>	<b>32-bit FX</b>	<b>17</b>	<b><math>9.66 \times 10^{-2}</math></b>	<b>721</b>	<b><math>1.02 \times 10^{-9}</math></b>

\* The quantum emulator is fixed with a application and #Qubits.

<sup>†</sup> The #Gate in this work is higher than other work due to the no-use of control-rotation gates.

<sup>††</sup> The Normalized Gate Speed (NGS) (s / (gate × amplitude)) = Execution time / (#Gates × 2<sup>#Qubits</sup>), smaller is better.

qubits systems, but has limited processing capabilities and HBM overhead in systems over 20 qubits.

In addition, power analysis also highlights HPQEA's efficiency in Fig. 6. While it consumes slightly more power than QEA  $1.73\times$  (0.939 vs 0.543 W) but achieves significantly faster execution times. Furthermore, as compared to the Intel i9-10940X CPU, HPQEA helps save from  $28.75\times$  (0.939 vs 27 W) and  $149.1\times$  (0.939 vs 140 W). Fidelity and MSE values, presented in Fig. 7, are almost identical to Qiskit's results, indicating correctness and numerical reliability.

For a more meaningful benchmark, Fig. 8 compares execution times QFT circuits across HPQEA, QEA, and NVIDIA A100. Compared to QEA, it delivers speedups between  $2.24\times$  and  $3.39\times$ , while outperforming A100 by up to  $10^4\times$  at quantum circuits below 20 qubits. Beyond 20 qubits, however, HPQEA's performance decreases, running between  $4.5\times$  and  $8.44\times$  slower than A100. The slowdown is caused by limited computational resources and HBM overhead. This bottleneck could be addressed by increasing computing resources and optimizing HBM usage. In addition, the MSE results remain consistent with previous experiments, verifying the reliability of HPQEA.

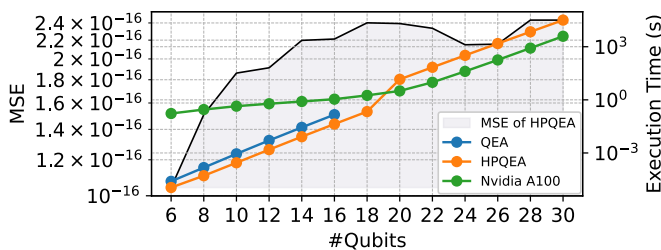


Figure 8: Comparison of QFT circuits in terms of mean square error (MSE) between Qiskit and HPQEA state vectors (Black line, left-y axis), and the execution time between HPQEA and other platforms (right y-axis)

#### D. Comparison with FPGA-based works

To provide a more comprehensive assessment of the efficiency and optimization of HPQEA, this section presents a comparative evaluation against FPGA-based quantum accelerators using the QFT algorithm.

Table II first summarizes the hardware characteristics of HPQEA alongside prior works [12]–[15], [17]–[19], focusing on key metrics such as maximum operating frequency, reconfigurability, supported #Qubits, and Normalized Gate Speed (NGS). The NGS metric enables fair comparisons across platforms with differing hardware resources, precision levels, and quantum circuit sizes. For consistency across evaluations, we used a 17-qubit QFT circuit as a reference benchmark, which is also supported by the comparative designs. The results demonstrated that HPQEA achieves superior performance and flexibility compared to the other works. Specifically, HPQEA outperforms [12]–[14], [17]–[19] in terms of maximum frequency, with improvements ranging from  $1.07\times$  (250 vs. 233 MHz) to  $2.5\times$  (250 vs. 100 MHz). Furthermore, HPQEA supports a wider range of quantum algorithms and larger #Qubits systems, enhancing its adaptability for diverse applications. In terms of NGS, HPQEA exhibits substantial improvements over earlier designs, achieving speedups from  $3.41\times$  ( $1.02 \times 10^{-9}$  vs.  $3.48 \times 10^{-9}$  seconds) up to  $522.55\times$  ( $1.02 \times 10^{-9}$  vs.  $5.33 \times 10^{-7}$  seconds). In comparison with [15], HPQEA could not surpass it in every individual metric, such as operational frequency, supported qubits, or NGS. However, it still offers a significantly wider range of supported algorithms since it is not limited to QFT. In summary, these results highlight HPQEA's superior performance, configurability, and scalability when compared to existing FPGA-based quantum accelerators.

#### V. CONCLUSION

This study introduces a high-performance quantum emulation accelerator (HPQEA) designed to address challenges in terms of broad applicability, scalability, performance, and resource efficiency for emulating quantum circuits. HPQEA incorporates three fundamental concepts: a high-performance

computing core, an optimized controlled-NOT gate computation strategy, and effective utilization of high-bandwidth memory. Experimental results have shown that it has a wide range of applications, is highly reliable, and uses resources efficiently, outperforming CPU, GPU, and FPGA-based techniques by supporting up to 30 qubits. Nonetheless, execution times were impacted by restricted computational resources and inefficient HBM usage. The future study will focus on optimizing HBM usage while discovering efficient resource utilization strategies to address these constraints.

#### ACKNOWLEDGMENT

This work was supported by JST-ALCA-Next (JPM-JAN23F4), the NAIST Senju Monju Project (Daiichi-Sankyo "Habataku" Support Program for the Next Generation of Researchers).

#### REFERENCES

- [1] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge university press, 2010.
- [2] R. R. Schaller, "Moore's law: past, present and future," *IEEE spectrum*, vol. 34, no. 6, pp. 52–59, 2002.
- [3] SpinQ. (2025) How many quantum computers are there in 2025? Accessed: 2025-08-08.
- [4] N. Wiebe, D. Braun, and S. Lloyd, "Quantum algorithm for data fitting," *Physical review letters*, vol. 109, no. 5, p. 050505, 2012.
- [5] Q. Wang, J. Guan, J. Liu, Z. Zhang, and M. Ying, "New quantum algorithms for computing quantum entropies and distances," *IEEE Transactions on Information Theory*, vol. 70, no. 8, pp. 5653–5680, 2024.
- [6] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.
- [7] C.-C. Wang, Y.-C. Lin, Y.-J. Wang, C.-H. Tu, and S.-H. Hung, "Queen: A quick, scalable, and comprehensive quantum circuit simulation for supercomputing," 2024.
- [8] B. et al, "cuQuantum SDK: A High-Performance Library for Accelerating Quantum Science," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 01, 2023, pp. 1050–1061.
- [9] V. Bergholm and et al, "PennyLane: Automatic differentiation of hybrid quantum-classical computations," 2022.
- [10] J. Mei, M. Bonsangue, and A. Laarman, "Simulating quantum circuits by model counting," in *Computer Aided Verification*, A. Gurfinkel and V. Ganesh, Eds. Cham: Springer Nature Switzerland, 2024, pp. 555–578.
- [11] J. Johansson, P. Nation, and F. Nori, "QuTiP 2: A Python framework for the dynamics of open quantum systems," *Computer Physics Communications*, vol. 184, no. 4, pp. 1234–1240, 2013.
- [12] A. Silva and O. G. Zabaleta, "Fpga quantum computing emulator using high level design tools," in *2017 Eight Argentine Symposium and Conference on Embedded Systems (CASE)*. IEEE, 2017, pp. 1–6.
- [13] N. e. a. Mahmud, "Efficient computation techniques and hardware architectures for unitary transformations in support of quantum algorithm emulation," *Journal of Signal Processing Systems*, vol. 92, pp. 1017–1037, 2020.
- [14] Y. Hong, S. Jeon, S. Park, and B.-S. Kim, "Quantum circuit simulator based on fpga," in *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2022, pp. 1909–1911.
- [15] H. M. Waidyasooriya, H. Oshiyama, Y. Kurebayashi, M. Hariyama, and M. Ohzeki, "A scalable emulator for quantum fourier transform using multiple-fpgas with high-bandwidth-memory," *IEEE Access*, vol. 10, pp. 65 103–65 117, 2022.
- [16] T. X. H. e. a. Le, "Theoretical Analysis of the Efficient-Memory Matrix Storage Method for Quantum Emulation Accelerators with Gate Fusion on FPGAs," in *2024 IEEE 17th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoc)*, 2024.
- [17] S. Liang, Y. Lu, C. Guo, W. Luk, and P. H. Kelly, "Pcq: Parallel compact quantum circuit simulation," in *2024 IEEE 32nd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2024, pp. 24–31.
- [18] T. H. Vu, V. T. D. Le, H. L. Pham, Q. C. Nguyen, and Y. Nakashima, "FQsun: A Configurable Wave Function-Based Quantum Emulator for Power-Efficient Quantum Simulations," *arXiv preprint arXiv:2411.04471*, 2024.
- [19] V. D. Tran, T. H. Vu, V. T. D. Le, H. L. Pham, and Y. Nakashima, "Qea: An accelerator for quantum circuit simulation with resources efficiency and flexibility," in *2025 10th International Conference on Integrated Circuits, Design, and Verification (ICDV)*, 2025.
- [20] AMD, "Alveo u280 data center accelerator card," 2020, accessed: 2025-03-18.
- [21] S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits," *Phys. Rev. A*, vol. 70, p. 052328, Nov 2004.
- [22] P. R. et al, "Highly optimized quantum circuits synthesized via data-flow engines," *Journal of Computational Physics*, vol. 500, p. 112756, 2024.
- [23] H. Horii and J. Doi, "Optimization of quantum computing simulation with gate fusion," 2021.
- [24] S. e. a. Sim, "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms," *Advanced Quantum Technologies*, vol. 2, no. 12, p. 1900070, 2019.
- [25] T. Haug, K. Bharti, and M. Kim, "Capacity and quantum geometry of parametrized quantum circuits," *PRX Quantum*, vol. 2, p. 040309, Oct 2021.
- [26] D. Coppersmith, "An approximate Fourier transform useful in quantum factoring," 2002.
- [27] A. Javadi-Abhari and et al, "Quantum computing with Qiskit," 2024.