FPGA-based implementaation of a controlled phase-rotation computational accelerator for quantum Fourier transforms

Chung-ang University, Seoul, South Korea
SeungJun Lee

Poster Session C
24th

Abstract

In this paper, we implement the controlled phase rotation operation, a key operation in the quantum Fourier transform (QFT), on an FPGA. We propose a hybrid approach to accelerate certain mathematical operations with classical hardware while preserving the superposition and entanglement that are inherent properties of quantum computers. Using Xilinx Artix-7 FPGAs, we demonstrate improved performance over existing software implementations and verify the feasibility of interfacing with real quantum system

Keywords— FPGA, Quantum Fourier Transform (QFT), Controlled Phase Rotation, Hybrid Quantum-Classical System, Real-Time Processing, Quantum Computing Acceleratio

Introduction

Quantum computing offers a revolutionary approach to solving complex problems that are beyond the reach of classical computers. A cornerstone of many influential quantum algorithms, such as Shor's algorithm, is the Quantum Fourier Transform (QFT). The QFT achieves an exponential speedup over classical methods by efficiently converting quantum data into the frequency domain, which is key to its computational power.

However, today's quantum hardware faces significant challenges, including limited qubit counts, high susceptibility to noise, and operational instability. To overcome these limitations, quantum-classical hybrid systems have emerged as a promising solution. These systems offload critical quantum operations to specialized classical hardware, enhancing both performance and reliability.

Building on this hybrid approach, we propose a system that implements controlled phase rotation—a fundamental operation within the QFT—using a Field-Programmable Gate Array (FPGA). We selected the FPGA for its inherent parallelism, low power consumption, and reconfigurable architecture, making it ideal for high-performance hardware acceleration. Our design specifically utilizes the FPGA's DSP slices and fixed-point arithmetic to implement these operations with high accuracy and real-time processing capabilities, thereby optimizing the hardware implementation of a critical quantum operation.

System Design

1) Complex Arithmetic Module

This module performs complex number multiplication using a 16-bit fixed-point format. By utilizing DSP slices, high-speed computations are enabled, and state-machine-based control logic ensures stable operation.

2) CORDIC (Coordinate Rotation Digital Computer) Module

The CORDIC algorithm is implemented for trigonometric function calculations. A 16-stage iterative process was used to achieve sufficient precision and
a pipelined architecture was applied to optimize the processing speed.

3) Controlled Phase Rotation Computation Module

This module integrates the complex arithmetic and CORDIC modules to perform the final controlled phase rotation operation. Conditional operations based on the control qubit state were optimized to eliminate unnecessary computations, and LUT-based calculations were used

A. Implementation of Complex Arithmetic Mod

The Complex Arithmetic Module executes the multipli-cation of two complex numbers, each represented in a 16-bit fixed-point format. It implements the standard complex multiplication formulas: (areal · breal) − (aimag · bimag) for the real part, and (areal · bimag) + (aimag · breal) for the imaginary part. High-speed computation is achieved by utilizing dedicated DSP slices on the FPGA, and fixed-point arithmetic provides hardware optimization. State-machine-based control logic ensures stable operation

B. Implementation of the CORDIC Algorithm

The CORDIC (Coordinate Rotation Digital Computer) module performs trigonometric function calculations. It uses iterative micro-rotations, employing add, subtract, and bit-shift operations. At each of the 16 iterations, coordinate and angle accumulator registers (x, y, and z) are updated based on the sign of the remaining angle in z, using pre-calculated arctangent values from a lookup table (atan_table). a pipelined architecture is applied to optimize processing speed while ensuring sufficient precision

Test

A. Test Cases and Results

To validate our design, we evaluated the accuracy and performance of the FPGA-implemented controlled phase rotation, comparing it against an ideal software implementation in Python. Our tests confirmed that the 16-bit fixed-point arithmetic on the FPGA provided high accuracy, precisely matching the outputs from Python's floating-point computations in all test cases.

B. Performance Analysis

In our performance analysis, the FPGA implementation achieved a processing speed of just 585 ns per operation. This is a significant 4.3-fold speedup over the software equivalent, which required 2.5 μs, thereby verifying the system's capability for real-time processing. Furthermore, the hardware design proved exceptionally efficient, consuming minimal resources: 3.7% of LUTs, 1.5% of FFs, and 3.3% of DSP slices. This low resource footprint demonstrates the design's scalability for more complex quantum circuits.

Conclusion

In this study, we successfully demonstrated the feasibility of implementing the controlled phase rotation—a core component of the Quantum Fourier Transform (QFT)—on an FPGA. Our work validates a hybrid quantum-classical approach, where specific quantum operations are accelerated using classical hardware without compromising the fundamental quantum properties of superposition and entanglement.

Our FPGA-based implementation delivered significant performance gains, achieving a processing time of just 585 ns. This represents a more than fourfold speed improvement compared to a standard Python software implementation (which took approximately 2.5 μs), confirming the system's capacity for real-time processing.

Furthermore, we achieved this performance with remarkable efficiency and accuracy. Despite using 16-bit fixed-point arithmetic, our system maintained an acceptable level of precision compared to Python's floating-point calculations. The implementation was also highly resource-efficient, utilizing only 3.7% of LUTs, 1.5% of FFs, and 3.3% of DSP slices, which demonstrates clear potential for building more complex and scalable architectures.

These results not only highlight the promise of quantum-classical hybrid systems but also validate the effectiveness of our implementation and optimization techniques for practical quantum computing applications.