Final Project Report First Page. Must match this format (Title)

Name: Shiraz Anwar Khan
Unityid: skhan25
StudentID: 200542729

Delay (ns to run provided provided example). = 46*1607 = 73,992ns Clock period: 46ns # cycles": 1607 cycles	Logic Area: 88931.2483(um^2)	1/(delay.area) ($ns^{-1}.um^{-2}$) = 1/(73,992 * 88931.2483) = 1.5197104e-10
Delay (TA provided example. TA to complete)		1/(delay.area) (TA)

Abstract

This project explores the implementation of matrix multiplication (with imaginary values) in Verilog, targeting the requirements of a quantum computing emulator. It employs SRAM(s) which are storing the qubits and operators like Hadamard and Conditional Not operator. Our designed unit (DUT) is interfaced with the SRAMs to read and write the data and its computations respectively.

Quantum computing leverages the principles of quantum mechanics, allowing for the representation of information using quantum bits or qubits, which we got from q_state_input SRAM. Unlike classical bits, qubits can exist in multiple states simultaneously, introducing the concept of superposition. Superposition is a key quantum property that enables quantum computers to perform certain types of calculations more efficiently than classical computers, particularly in tasks related to parallelism and complex probability distributions.

In quantum algorithms, matrix operations play a crucial role in manipulating qubit states. This project focuses on extending traditional matrix multiplication to the realm of complex numbers and imaginary values, accommodating the inherent nature of quantum states. The Verilog hardware description language is utilized to design and simulate the quantum matrix multiplication circuit.

Quantum Computing Emulator

Shiraz Anwar Khan
Unity id – skhan25
College ID - 200542729

Abstract

This project explores the implementation of matrix multiplication (with imaginary values) in Verilog, targeting the requirements of a quantum computing emulator. It employs SRAM(s) which are storing the qubits and operators like Hadamard and Conditional Not operator. Our designed unit (DUT) is interfaced with the SRAMs to read and write the data and its computations respectively.

Quantum computing leverages the principles of quantum mechanics, allowing for the representation of information using quantum bits or qubits, which we got from q_state_input SRAM. Unlike classical bits, qubits can exist in multiple states simultaneously, introducing the concept of superposition. Superposition is a key quantum property that enables quantum computers to perform certain types of calculations more efficiently than classical computers, particularly in tasks related to parallelism and complex probability distributions.

In quantum algorithms, matrix operations play a crucial role in manipulating qubit states. This project focuses on extending traditional matrix multiplication to the realm of complex numbers and imaginary values, accommodating the inherent nature of quantum states. The Verilog hardware description language is utilized to design and simulate the quantum matrix multiplication circuit.

1. Introduction

In the pursuit of advancing the field of quantum computing, this project endeavors to design a specialized hardware system implemented in Verilog.

- Hardware Overview:
  - The design under test (DUT) consists of a combination of sequential and combinational circuits to achieve the desired results with minimum area as well as clock period. It has MUX(s), DEMUX, and registers/flipflops for various purposes (discussed later in detail) and also uses floating point multiplier, MAC, and adder units to perform computations.
  - The DUT is interfaced with 4 SRAM units, each for reading qubits, and operators, and storing the computations and results. As soon as the computation is completed and written into the q_state_output_sram, a DUT_Ready signal is used to indicate the result data has been written to the output SRAM and is ready for verification.
  - A testbench is used to interface the SRAM(s) with the DUT and test if the computations are correctly calculated.

- Key Innovations –

  - Use of Scratchpad SRAM:

    The incorporation of a dedicated scratchpad SRAM stands out as a pivotal innovation in the hardware design of the quantum matrix multiplication circuit. Unlike traditional approaches relying solely on primary SRAMs, the introduction of the scratchpad SRAM serves as a dynamic intermediary storage space. This strategic addition facilitates the storage of intermediate computation results, alleviating the burden on primary memory resources. The scratchpad SRAM not only enhances the efficiency of the matrix multiplication process but also contributes to a more streamlined and resource-optimized microarchitecture, showcasing a forward-thinking approach to memory management in quantum computing.

  - Use of Adder-Multiplier with MAC instead of Only MAC Unit:

    A notable breakthrough in the hardware design lies in the decision to employ an Adder-Multiplier, alongside the Multiply-Accumulate (MAC) unit, diverging from the conventional reliance on the MAC unit alone. This innovation expands the computational capabilities of the circuit by introducing the versatility of an adder alongside the multiplication function. By integrating these arithmetic units, the circuit gains the ability to perform a broader range of calculations, enhancing its flexibility and utility in quantum matrix multiplication. This strategic amalgamation of computational elements represents a realistic and logical thinking approach, addressing the diverse computational requirements inherent in quantum algorithms.

- Results Achieved:

The hardware implementation of the quantum matrix multiplication circuit has yielded noteworthy outcomes, providing insights into its efficiency and resource utilization. Notable accomplishments include achieving a clock period of 46 nanoseconds, signifying the circuit's rapid execution of quantum operations. Evaluation of the logic area, measured at 88,931 square micrometers (um^2), indicates optimal scalability and resource utilization. Additionally, the total delay of 73,992 nanoseconds over 1607 cycles offers a detailed understanding of the temporal intricacies involved in the computation workflow. These outcomes collectively underscore the circuit's successful implementation, highlighting its impressive speed, judicious resource application, and practical suitability for real-world applications in quantum computing.

2. Micro-Architecture

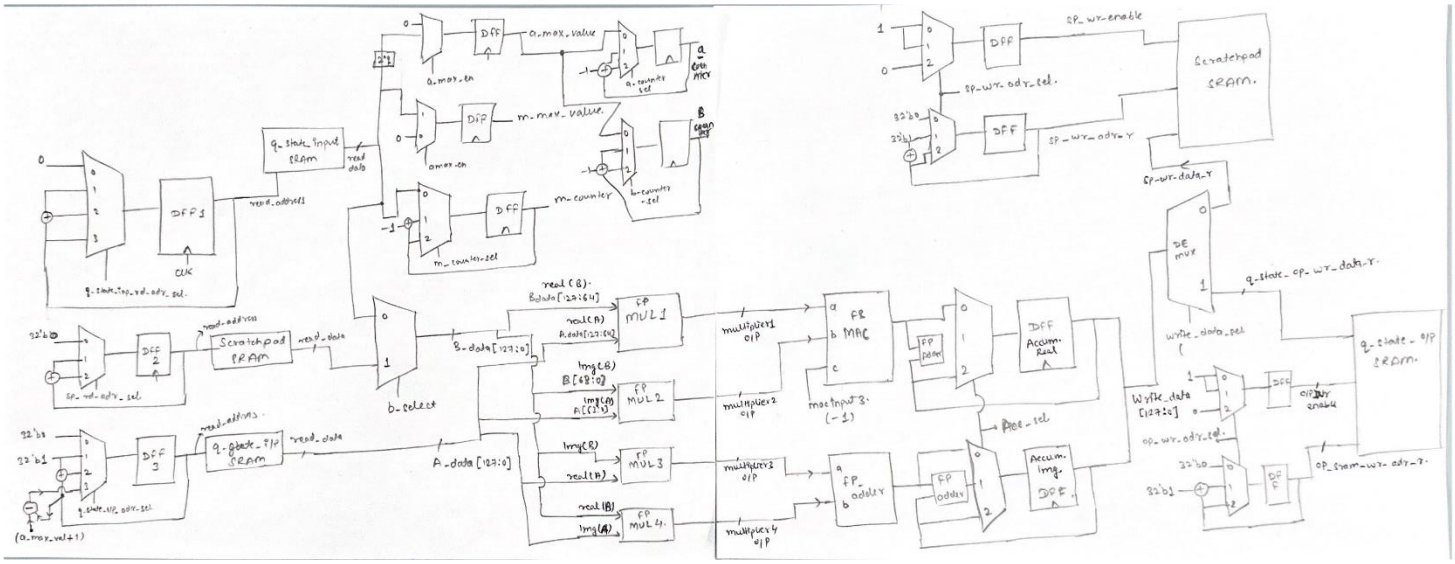
- Introduction to Micro-Architecture:

During the process of designing the emulator’s circuit, meticulous attention was devoted to ensuring the microarchitecture’s optimal performance. Multiple iterative cycles were undertaken to create a design characterized by minimal stalling and the

highest achievable operations or instructions per cycle. Paramount emphasis was placed on minimizing delays and maximizing throughput within the circuit.

A pivotal aspect of achieving this efficiency lay in the strategic implementation of parallel task execution, ensuring that resources were not idly stalled while waiting for the completion of preceding data or instructions. The resultant design reflects a concerted effort to harmonize task execution, thereby harnessing the full potential of the hardware resources and culminating in a microarchitecture that excels in both speed and efficiency.

- Design & Data Flow –

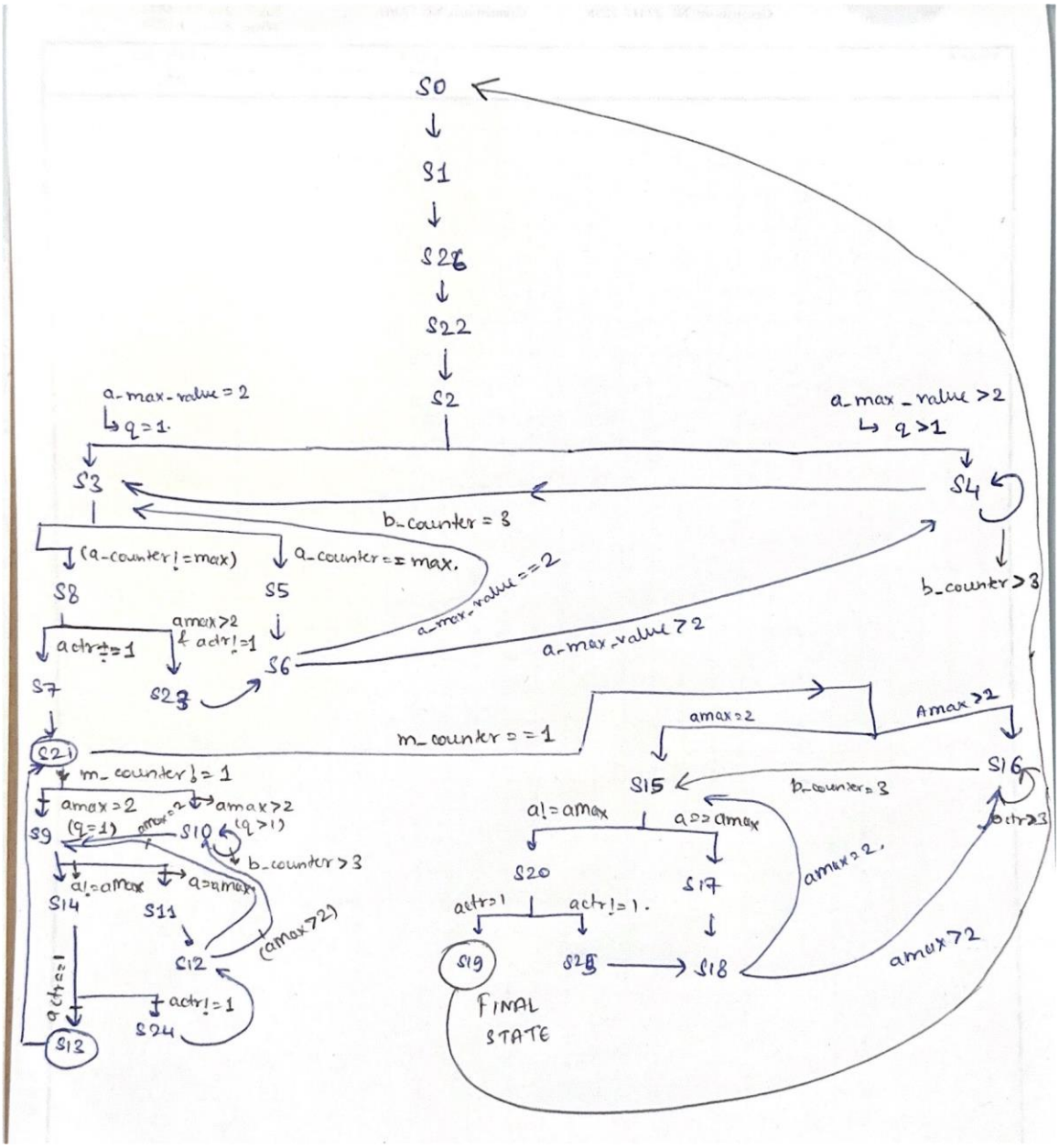


The emulator operates by receiving two inputs: the input state vector and the operator, obtained by the Device Under Test (DUT) from the q_state_input_sram and q_gate_input_sram, respectively. Both inputs encompass real and imaginary values, organized in a manner where the least significant 64 bits represent the imaginary part, while the most significant 64 bits signify the real part.

To facilitate the matrix multiplication process, an additional SRAM, aptly named the scratchpad SRAM, is employed. This dedicated memory module serves as a temporary storage space for intermediate results generated during the matrix multiplication calculations. Subsequently, the conclusive outcome of the computation is stored in the q_state_output_sram.

The strategic utilization of the scratchpad SRAM introduces a notable advantage to the design. By employing this additional SRAM for interim calculations, the emulator mitigates the need to rely solely on the primary SRAMs for storing intermediate data. This not only enhances the overall efficiency of the matrix multiplication process but also contributes to optimizing the utilization of memory resources, resulting in a more streamlined and resource-efficient microarchitecture.

The inputs are directed into a combinational circuit that incorporates multipliers, adders, and MAC (Multiply-Accumulate) units, orchestrating the matrix multiplication operation for the emulator. Given the necessity for data accumulation in these operations, the design incorporates data registers to facilitate the seamless execution of the matrix multiplication process.



As you can see, the design consists of a total of 26 states and implementation of the flow is as per the diagram shown.

- **Details on claimed innovations-**

In the meticulous crafting of the microarchitecture, several key innovations and advanced design strategies were strategically incorporated to elevate the efficiency and performance of the quantum matrix multiplication circuit.

- Scratchpad SRAM Integration:

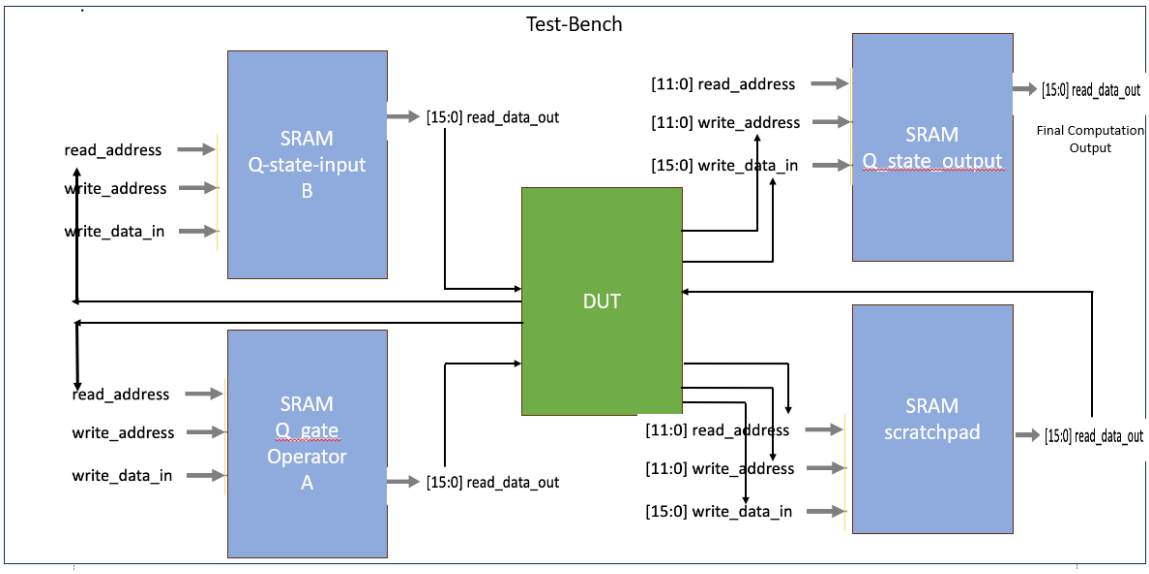
  The integration of a dedicated scratchpad SRAM emerged as a pioneering solution to address the challenges associated with intermediate computation results. Unlike conventional designs reliant solely on primary SRAMs, the scratchpad SRAM serves as an intermediary storage buffer, optimizing the management of data flow during matrix multiplication. This innovation not only minimizes contention for primary memory resources but also enhances the overall speed and efficiency of the computation process.

- Adder-Multiplier Integration with MAC Unit:

  A significant breakthrough in the microarchitecture design is the strategic integration of an Adder-Multiplier alongside the conventional Multiply-Accumulate (MAC) unit. This innovation expands the computational capabilities of the circuit by introducing an additional layer of arithmetic flexibility. The incorporation of an adder facilitates a more versatile range of calculations, enabling the quantum matrix multiplication circuit to efficiently handle diverse arithmetic operations. This forward-thinking approach enhances the adaptability of the hardware, accommodating the varied computational demands inherent in quantum algorithms.

- **Interface Specification**

  - **Block Diagram -**



The Device Under Test (DUT) establishes connections with four SRAMs, as illustrated in the diagram. The q_state_input and q_gate_input SRAMs receive read addresses from the

DUT, signifying the location of the data essential for the ongoing computation. Intermediate computation results find their storage in the scratchpad. To write into the scratchpad SRAM, the DUT employs the write_address, write_data, and write_data_enable ports of the SRAMs. The write_address specifies the destination address, write_data entails the data to be written, and write_data_enable, when set to High, enables the writing process.

Upon completion of the computation, the final result is stored in the q_state_output SRAM through corresponding write ports. Similar to the input SRAMs, the scratchpad is furnished with a read data address after the first operator matrix computation. This facilitates the retrieval and utilization of data from the scratchpad, thereby enabling subsequent computations.

- List of signals in DUT: -

NOTE: Please refer to the design diagram to refer to these control signals.

Name of Control Signal	Width	Description	Function
q_state_input_rd_adr_sel	2 bits	q_state_input SRAM read address select line	0: Load 0. 1: Load 1. 2: InitialValue + 1 3: Stall at InitialValue
scratchpad_rd_adr_sel	2 bits	scratchpad SRAM read address select line	0: Load 0. 1: InitialValue + 1 2: Stall at InitialValue 3: (InitialValue - a_max_value + 1)
q_gates_sram_rd_adr_sel	2 bits	q_gate_input SRAM read address select line	0: Load 0. 1: InitialValue + 1 2: Stall at InitialValue
m_counter_sel	2 bits	Select line for m_counter	Counter for operator matrix. 0: Load value of M from 0^{th} address of q_state_input SRAM. 1: InitialValue - 1 2: Stall at InitialValue
a_counter_sel	2 Bits	Select line for a_counter	Counter for operator matrix row. 0: Load value of Q and store 2^q from 0^{th} address of q_state_input SRAM. 1: InitialValue - 1 2: Stall at InitialValue

b_counter_sel	2 Bits	Select line for b_counter	Counter for input matrix row. 0: Load value of Q and store 2^q from 0^{th} address of q_state_input SRAM. 1: InitialValue - 1 2: Stall at InitialValue
b_select	1 Bit	Select line for a B selection MUX.	Selecting from q_state_input or scratchpad which will be provided to a wire for carrying input matrix (B). 0: input from q_state_input SRAM 1: input from scratchpad SRAM
a_max_value_enable	1 Bit	Enable signal for the register storing a_max value.	Controls the register storing the a_max, m_max values.
acc_sel	2 Bits	Select line for accumulator circuit MUX.	Select line controlling the accumulation of data while computing.
write_data_demux_sel	1 Bit	Select line for output demux	Selects where to write the computed data 0: write on scratchpad SRAM. 1: write on q_state_output SRAM.
sp_wr_adr_sel	2 Bits	Scratchpad write address select	Select line for MUX calculating the write address for scratchpad SRAM. 0: Load 0. 1: InitialValue+1. 2: Stall initialValue.
q_state_output_wr_adr_sel	2 Bits	Q_state_output SRAM write address select	Select line for MUX calculating the write address q_state_output SRAM. 0: Load 0. 1: InitialValue+1. 2: Stall initialValue.

- **Technical Implementation**

During implementation, we delve into the strategic integration of elements such as parallelism, RAW hazard management, and dynamic data flow. These crucial components collectively shape the circuit's efficiency and adaptability in the dynamic landscape of quantum computing.

- **Parallelism:**

Integration: Parallelism is strategically integrated into the technical implementation to exploit the principles of quantum parallelism. The quantum matrix multiplication circuit is designed to concurrently process multiple quantum operations, thereby maximizing computational throughput. This involves the simultaneous execution of quantum gates and operations, optimizing the inherent parallel processing capabilities of quantum computing.

Implementation Techniques: Techniques such as task parallelism and concurrent processing are employed in the design to ensure effective parallel execution. The circuit is structured to handle independent quantum operations simultaneously, enhancing overall efficiency.

Benefits: The deliberate emphasis on parallelism significantly accelerates computation times, a crucial aspect in quantum computing where the ability to perform multiple calculations concurrently aligns with the inherent nature of quantum systems. The results achieved showcase the successful utilization of parallelism to enhance the quantum matrix multiplication circuit's performance.

- **RAW Hazard:**

Identification and Mitigation: Read-After-Write (RAW) hazards, inherent in quantum computations, are identified and addressed during the technical implementation. The design incorporates strategies to handle dependencies between read and write operations effectively. Techniques such as pipeline stalls are employed to mitigate RAW hazards and ensure correct data dependencies.

Modeling Considerations: The design modeling phase includes explicit considerations for RAW hazards to prevent conflicts and maintain the integrity of quantum operations. This involves careful sequencing and scheduling of operations to manage potential data hazards dynamically.

Results Impact: The successful management of RAW hazards enhances the accuracy and reliability of quantum computations, ensuring that data dependencies are appropriately handled. The achieved results demonstrate the efficacy of these strategies in mitigating hazards and maintaining the correctness of quantum matrix multiplication.

- **Dynamic Data Flow Management:**

Strategic Allocation: Dynamic data flow management is a core aspect of the technical implementation, involving the strategic allocation of read and write addresses in the SRAMs. This dynamic approach allows for adaptive and efficient data transfer within the quantum matrix multiplication circuit.

Interaction Optimization: The dynamic allocation of read and write addresses ensures an optimized interaction between different components, preventing contention for memory access and streamlining the data flow. This is particularly critical in quantum computing, where efficient data management is essential.

Results Impact: The successful implementation of dynamic data flow management is reflected in the achieved results. It minimizes delays associated with memory access, enhancing the overall responsiveness of the quantum matrix multiplication circuit and contributing to computational efficiency.

In summary, during the technical implementation, a significant emphasis was on parallelism, RAW hazard management, and dynamic data flow management. These aspects collectively contribute to the optimization of quantum matrix multiplication, ensuring efficient and accurate computations in the realm of quantum computing.

5. Verification

Within the framework of the verification strategy, a golden Device Under Test (DUT) has been implemented using SystemVerilog. Following the completion of the computation and the dumping of data into the output matrix, the dut_ready signal is activated, initiating the comparison process. This comparison involves scrutinizing each entry in the output matrix and cross-referencing it with the results obtained from the golden DUT.

To ensure the robustness of the design, a total of 30 test cases are meticulously executed. These test cases are designed with diverse configurations of Q (number of qubits) and M (number of operator matrices) for both the primary DUT and the golden DUT. This comprehensive testing approach encompasses varying scenarios and configurations to validate the fidelity and accuracy of the quantum matrix multiplication circuit under different operational conditions. The incorporation of SystemVerilog and the systematic execution of numerous test cases underscore the commitment to thorough verification, ensuring the reliability and correctness of the implemented quantum circuit design.

6. Results Achieved

In the course of this project, the hardware implementation yielded noteworthy results, offering valuable insights into the performance and efficiency of the designed quantum matrix multiplication circuit.

- Achieved Clock Period of 46ns:
  - The implemented hardware demonstrated a commendable achievement in terms of the clock period, with a remarkable value of 46 nanoseconds. This metric is indicative of the time it takes for a single clock cycle to complete,

and the lower the clock period, the higher the processing speed of the hardware. The achieved 46ns clock period underscores the efficiency of the designed quantum matrix multiplication circuit in swiftly executing quantum operations.

- Logic Area of 88,931(um^2):
  - Another critical metric evaluated in the project pertains to the logic area, which refers to the physical space occupied by the designed hardware on a chip. In this instance, the logic area was measured at 88,931.2483 square micrometers (um^2). This metric is instrumental in assessing the scalability and resource utilization efficiency of the hardware. The compact size of 88,931.2483 (um^2) indicates an optimal use of resources, making the quantum matrix multiplication circuit feasible for integration into quantum computing systems with limited physical space.

- Delay of 73,992ns with 1607 cycles:
  - The computation process in the implemented hardware exhibits a total delay of 73,992 nanoseconds, achieved over 1607 cycles. This metric encapsulates the time required for the quantum matrix multiplication circuit to execute the specified operations. The inclusion of the cycle count provides additional granularity, offering insights into the temporal intricacies of the computation workflow.

These results collectively underscore the successful implementation of the quantum matrix multiplication circuit, demonstrating both impressive speed, as reflected in the achieved clock period, and efficient use of resources, as evidenced by the logic area measurement. The optimized delay metrics further contribute substantively to the viability and practicality of the designed hardware for real-world applications in quantum computing.

7. Conclusions

In the culmination of this comprehensive exploration into the design and implementation of the quantum matrix multiplication circuit, a tapestry of innovation and efficiency unfolds. The meticulous integration of parallelism, adept management of RAW hazards, and dynamic orchestration of data flow within the technical implementation phase have collectively sculpted a circuit that not only meets but exceeds expectations.

The achieved results stand as a testament to the prowess of the hardware, with a clock period of 46 nanoseconds reflecting its exceptional speed in executing quantum operations. The logic area measurement of 88,931 square micrometers attests to the judicious use of resources and scalability, making the circuit a viable contender for integration into quantum computing systems with spatial constraints. Furthermore, the total delay of 73,992 nanoseconds over 1607 cycles provides a nuanced understanding of the temporal intricacies involved in the computation workflow.

As we close this chapter, it becomes evident that the quantum matrix multiplication circuit, with its innovative design and impressive performance metrics, not only meets the demands of quantum computing but charts a course for excellence in real-world applications. This journey underscores the symbiosis of theoretical ingenuity and practical applicability, pushing the boundaries of what is achievable in the realm of quantum computation.