Quantum Computing Emulator

Name: Nirmal Kumar Sedhumadhavan
Unityid: nsedhum@ncsu.edu
StudentID: 200483323

Delay (ns to run provided provided example). Clock period: 8.7 ns	Logic Area: 23791.839 (um^2) Memory: 335 MBytes	1/(delay.area) $4.8311 * 10^{-6}$ ($ns^{-1}.um^{-2}$)

Introduction
This project focuses on the hardware design of a Quantum Computing Emulator. The primary task involves performing matrix multiplications between an initial quantum state vector and a sequence of quantum operator matrices. Since the matrix elements are complex numbers, the design accounts for both real and imaginary components during computation. The result of these multiplications is a final state vector matrix, which is stored in an output SRAM. This output is subsequently verified for correctness through a series of validation checks.

Implementation
The multiplication process begins by computing the product of the initial quantum gate matrix and the quantum state vector matrix. For each operation, the first element of the quantum gate matrix row and the corresponding element of the quantum state matrix column are fetched and multiplied using a Design ware floating-point Multiply unit. The real and imaginary components of the result are accumulated separately using a Design ware floating-point addition unit.

This process continues element by element along the row of the quantum gate matrix, accumulating the partial sums. Once the entire row has been processed, the final accumulated result is written to a designated location in the scratchpad SRAM. The system then proceeds to the next row of the quantum gate matrix, restarting the process with the first column of the quantum state matrix.

After the complete multiplication of the initial gate and state matrices, the resulting intermediate matrix stored in the scratchpad SRAM is used as the input for multiplication with the next quantum gate matrix in the sequence. This iterative process continues, with each intermediate result being stored back in the scratchpad and used in the next stage.

Upon completing the final matrix multiplication with the last quantum gate, the resulting state vector is written to the output SRAM. This final output is then used for verification and correctness testing.

2. Interface Specification

Signal Name	Width	Function/Description
inst_a	64 bits	Operand for floating-point multiplication and addition.
inst_b	64 bits	Operand for floating-point multiplication and addition.
z_inst_mult	64 bits	Result of the floating-point multiplication.
z_inst_adder	64 bits	Result of the floating-point addition.
status_inst_mult	8 bits	Status output from the floating-point multiplier.
status_inst_adder	8 bits	Status output from the floating-point adder.
current_state	4 bits	Current state of the state machine.
next_state	4 bits	Next state of the state machine.
set_dut_ready	1 bit	Control signal to set the DUT ready status.
get_array_size	1 bit	Control signal to get the size of the array.
save_array_size	1 bit	Control signal to save the size of the array.
input_sram_sel	1 bit	Selects between Input and Scratchpad SRAM.
q_state_output_sram_write_enable_r	1 bit	Write enable for the Q state output SRAM.
scratchpad_sram_write_enable_r	1 bit	Write enable for the scratchpad SRAM.
q_gates_sram_read_address_r	13 bits	Read address for the Q gates SRAM.
scratchpad_sram_read_address_r	13 bits	Read address for the scratchpad SRAM.
scratchpad_sram_write_address_r	13 bits	Write address for the scratchpad SRAM.

q_state_input_sram_read_address_r	5 bits	Read address for the Q state input SRAM.
q_state_output_sram_write_address_r	5 bits	Write address for the Q state output SRAM.
scratchpad_sram_write_data_r	128 bits	Data to be written to the scratchpad SRAM.
q_state_output_sram_write_data_r	128 bits	Data to be written to the Q state output SRAM.
m	5 bits	Number of matrices in Q gate.
m_counter	5 bits	Counter for the matrices.
q_state_input_size	5 bits	Size of the Q state input.
row_counter	5 bits	Counter for the current row being processed.
q_gates_size	8 bits	Size of the Q gates.
counter	8 bits	Counter for the elements being processed.
real_adder	64 bits	Accumulator for the real part of the product.
img_adder	64 bits	Accumulator for the imaginary part of the product.
m_counter_sel	2 bits	Selector for the M counter operation.
row_counter_sel	2 bits	Selector for the row counter operation.
counter_sel	2 bits	Selector for the elements in the matrix.
sram_write_enable_sel	2 bits	Selector for the SRAM write enable operation.
input_read_addr_sel	2 bits	Selector for the input read address operation.
q_gates_read_addr_sel	2 bits	Selector for the Q gates read address operation.
scratchpad_read_addr_sel	2 bits	Selector for the scratchpad read address operation.

output_write_addr_sel	2 bits	Selector for the output write address operation.
scratchpad_write_addr_sel	2 bits	Selector for the scratchpad write address operation.
real_adder_sel	2 bits	Selector for the real adder operation.
img_adder_sel	2 bits	Selector for the imaginary adder operation.
compute_mult_adder	3 bits	Selector for the computation operation (multiplication/adder).
input_sram_flag	1 bit	Flag indicating the current input SRAM.
last_element	1 bit	Flag indicating the last element in the matrix.
last_row	1 bit	Flag indicating the last row in the matrix.
not_last_matrix	1 bit	Flag indicating not the last matrix.
last_matrix	1 bit	Flag indicating the last matrix.
last_matrix_prev	1 bit	Flag indicating the m-1 matrix.

3. Technical Implementation
FSM Attached as JPEG file

4. Results Achieved

Clock Period: 8.7 ns
Area: 23791.838 um^2
Performance: $4.8311 * 10^{-6}$ ($ns^{-1}.um^{-2}$)