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

Name: Tzu-Ching Yeh
Unity_id: tyeh6
Student ID: 200542364

Delay (ns to run provided provided example). Clock period: 20 ns # cycles: 1566	Logic Area: (um^2) 94420.9567 Memory: N/A	1/(delay.area) ($ns^{-1}.um^{-2}$): 3.38150357e-10
Delay (TA provided example. TA to complete)		1/(delay.area) (TA)

Abstract

Quantum computing is a type of computing that uses qubits, which can represent multiple states at once due to superposition. This allows quantum computers to perform certain calculations faster than classical computers. Qubits can also be entangled, meaning their states are instantaneously connected, enabling complex interactions. Quantum computers excel at specific problems like factorization, simulation of quantum systems, and optimization. This project aims to design a nearly fully functional Quantum Emulator with complex values.

Project title: Quantum Computing Emulator

Student name: Tzu-Ching Yeh

Abstract

Quantum computing is a type of computing that uses qubits, which can represent multiple states at once due to superposition. This allows quantum computers to perform certain calculations faster than classical computers. Qubits can also be entangled, meaning their states are instantaneously connected, enabling complex interactions. Quantum computers excel at specific problems like factorization, simulation of quantum systems, and optimization. This project aims to design a nearly fully functional Quantum Emulator with complex values

1. Introduction

This Quantum Emulator module contains initial state vector, and operator matrices. The majority of cases the initial state of the state vector would be that all qubits are initialized to zero giving us this matrix on the left which corresponds to, this just means that when the quantum circuit is measured, it will give us a result of "00" 100% of the time. If there is n qubits in this system, then the initial state vector of it will be 2^n rows, and the qubit operator will be 2^n x 2^n matrix. Hence, once the multiplication of the first operator matrix and initial state vector is finished, we will get a 2^n x 1 matrix which will be used to multiplied with the following operator. After finishing all multiplication, we will get the result. To accelerate the calculation, I implemented and designed a pipelined structure allowing the ASIC to process more data than traditional method.

The following three sections illustrate the details of this design. Section 2 showing the micro-architecture, Section 3 focusing on the specification of the interface, and Section 4 illustrating the technical implementation of this project. Moreover, Section 5 and Section 6 explaining the verification method and achieved results. Finally, Section 7 summarizing the key results of this project.

2. Micro-Architecture

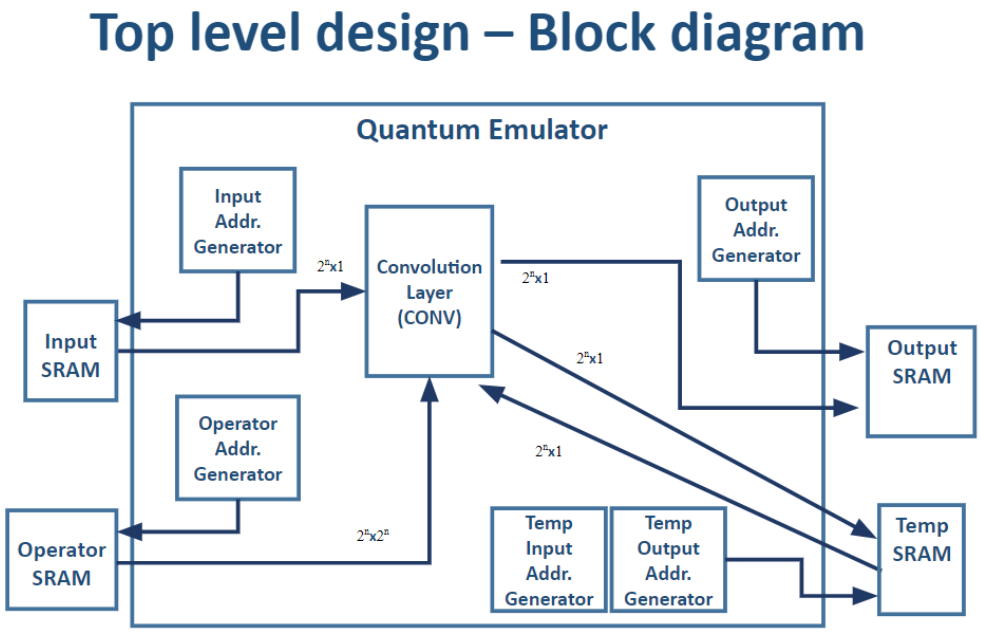

Figure 1. High level architecture block diagram.

Data flow:

A. The input address generator creates a specific sequence of addresses to let the multiplication module read the element in the initial state vector.

B. Similarly, the operator generator generates a sequential address and lets the multiplication module read the element in a 2^n x 2^n matrix.

C. The multiplication module performs pipelined, it will continuing generating the result, once the first result is generated.

D. Temp sram stores the elements of the 2^n x 1 matrix generated by the multiplication module. Therefore, it need temp input address generator, and temp output address generator tp create specific sequence of addresses to let the multiplication module read the element in the 2^n x 1 matrix, and send out the computation results.

E. The output address generator generates the corresponding address to send out the computation results multiplication module.

3. Interface Specification

Figure 2 below shows the interface between peripheral devices (i.e., SRAMs and control unit) and the Quantum Emulator hardware module.

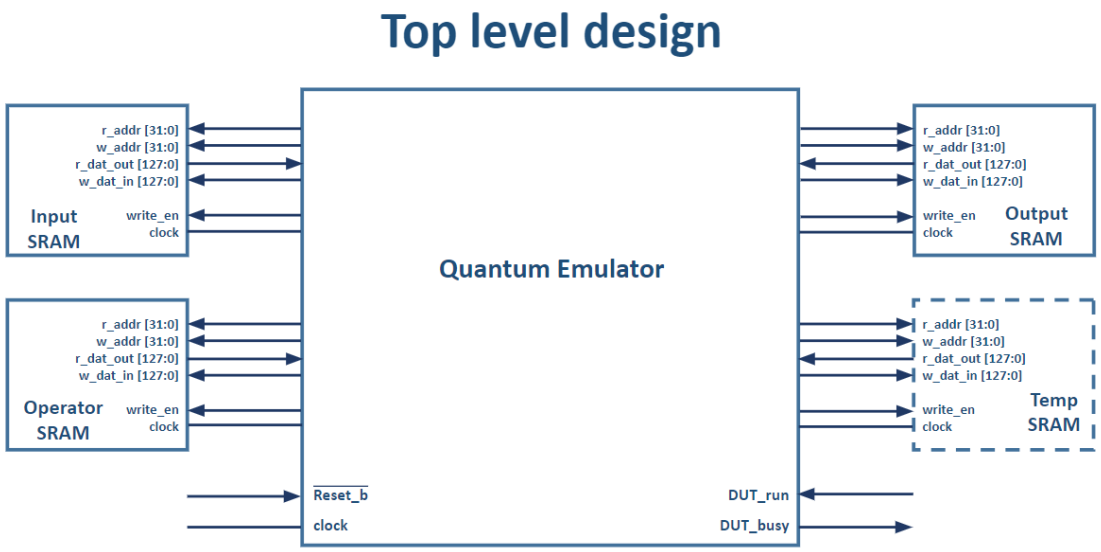

Figure 2. Top level design hardware interface.

The peripheral device contains a input SRAM storing the initial state vector, a operator SRAM for storing the operator matrix, a output SRAM saving the computation results from Quantum Emulator, and a temp SRAM saving the computation results from Quantum Emulator which will be used as a new input to multiply with operator.

Figure3 shows the timing diagram of SRAM interface.

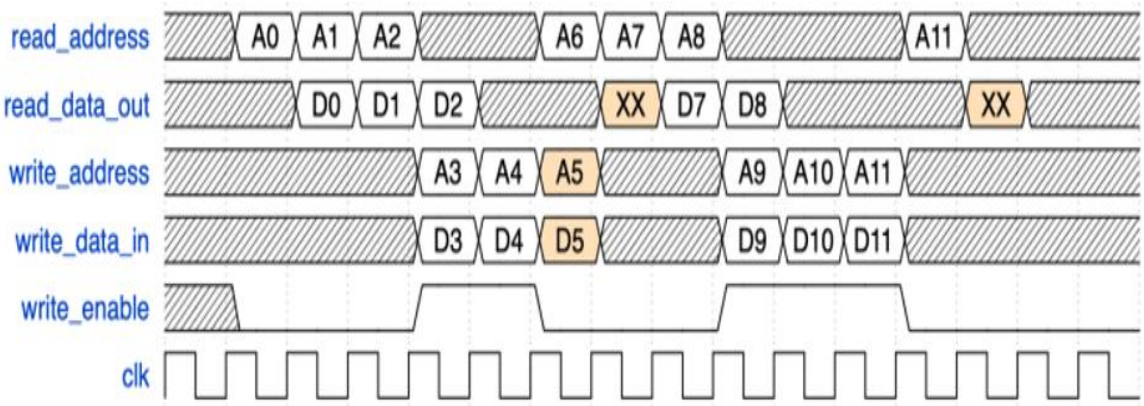

Figure 3. SRAM interface timing diagram.

Table 1. Global signals

Name	Source	Width	Description
clock	Clock source	1	Global clock signal
reset_b	Reset source	1	Global reset signal, active LOW.
DUT_valid	Control source	1	Signal for Ready to run, active HIGH
DUT_ready	Quantum Emulator	1	Ready for a new run to start, active LOW.

Table 2. Input SRAM channel signals

Name	Source	Width	Description
input_sram_read_address	Quantum Emulator source	32	Address of read matrix
input_sram_read_data	Input SRAM	128	Read data from Input SRAM.

Table 3. Operator SRAM channel signals

Name	Source	Width	Description
operator_sram_read_address	Quantum Emulator source	32	Address of weights matrix
operator_sram_read_data	Weights SRAM	128	Read data from Operator SRAM.

Table 4. Output SRAM channel signals

Name	Source	Width	Description
output_sram_write_enable	Quantum Emulator source	1	Enable the write function.
output_sram_write_address	Quantum Emulator source	32	Address of output matrix
output_sram_write_data	Quantum Emulator source	128	Write data to Output SRAM.

Table 5. Temp SRAM channel signals

Name	Source	Width	Description
temp_sram_write_enable	Quantum Emulator source	1	Enable the write function.
temp_sram_write_address	Quantum Emulator source	32	Address of output matrix
temp _sram_write_data	Quantum Emulator source	128	Write data to Temp SRAM
temp_sram_read_address	Quantum Emulator source	32	Address of read matrix
temp _sram_read_data	temp SRAM	128	Read data from Temp SRAM.

4. **Technical Implementation**

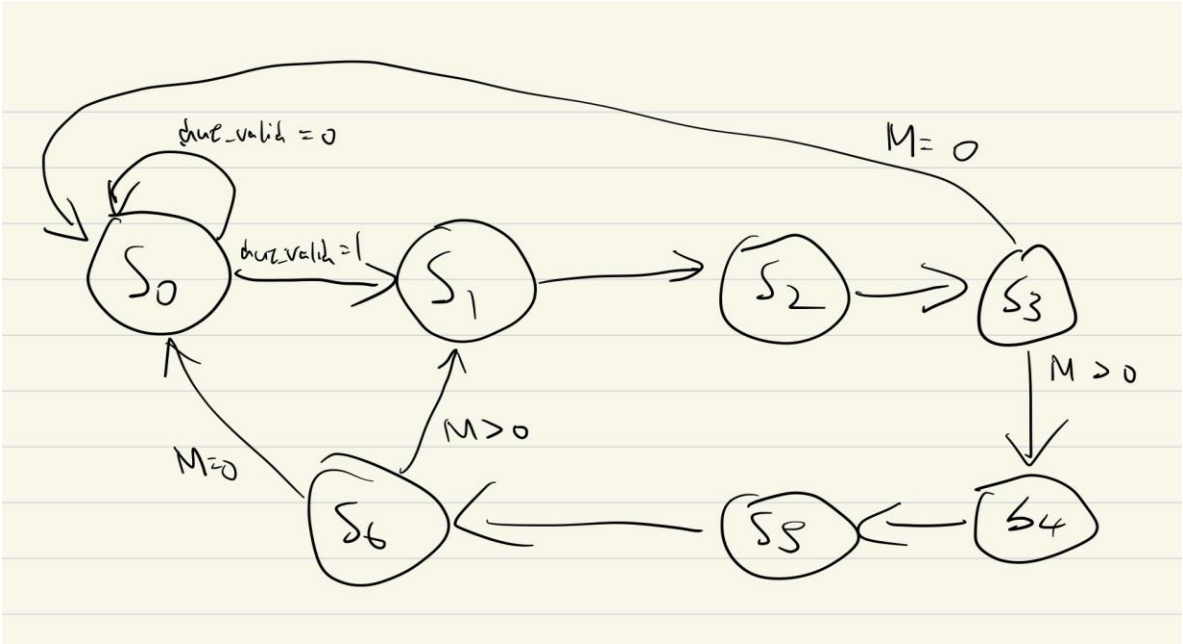

Figure 4. State machine diagram.

Figures 4 demonstrate the implementation structure and state machine diagram. The hardware will continuing read the data from input SRAM, and operator SRAM, the elements obtained from these two SRAM will be calculating in next cycle, in the next cycle the following elements will be obtained. Therefore, we can start process the calculation earlier, even though the former calculation is processing. After finishing the calculation of all the elements in the first row of the operator matrix, we add up these results, and will obtain the first element of the 2^n x 1 matrix which will be saved in temp SRAM. If all the elements in the operator matrix are multiplied, we will get a new 2^n x 1 matrix saved in temp SRAM, Hence, the temp SRAM will replace input SRAM and will be new input multiplying with the following operator matrix. In the end, when we are multiplying the final matrix, the result will be sent out to the output SRAM.

5. Verification

After multiplying all of the operator matrices with the 2^n x 1 input matrix, the test bench will compare the final result storing in the Output SRAM, and indicate whether it is match to the golden output. The test result from the testbench can been seen in the ModelSim terminal or in the result log file.

```
VSIM 7> run
# INFO: number of testcases:          4
# +CLASS+464
# INFO: DONE WITH RESETING DUT
# INFO: ######## Running Test: 1 ########
# INFO: Reading memory file: ../inputs/input1/test1_B.dat
# INFO: Reading memory file: ../inputs/input1/test1_A.dat
# INFO: reading ../inputs/output1/test1_C.dat
# INFO: Number of cases        : 2
# INFO: Number of passed cases : 2
# INFO: presentage passed     :  100.00
# INFO: Test: 1, Result:  100.00
#
```

Figure 5. The test results from testbench output in ModelSim terminal.

```
RESULTS.log
# INFO: Test: 1, Result:  100.00
# INFO: Test: 2, Result:  100.00
# INFO: Test: 3, Result:  100.00
# INFO: Test: 4, Result:  100.00
# INFO: Test: 5, Result:  100.00
# INFO: Test: 6, Result:  100.00
# INFO: Test: 7, Result:  100.00
# INFO: Finial Results          : 100.00
# INFO: Finial Time Result      : 28860 ns
# INFO: Finial Cycle Result     : 2886 cycles
```

Figure 6 The test4 results from testbench output in result log file.

6. Results Achieved

Table 6 below lists the cell area, throughput, and performance index. As you can see, when the clock period sets 20 ns, the total delay is 31320 ns. Furthermore, the total cell area of this design setting is 94420.9567 um^2, and the performance index is 3.38150357e-10

Table 6. Final results

Item	Test6
Clock period (ns)	20
Latency (cycles)	1566
Delay (ns)	31320
Cell area (um^2)	94420.9567
Delay * Cell area	2957264363.844
Performance = 1 / (Delay * Cell area)	3.38150357e-10

Figure 7, 8, and 9 show the synthesize final report with best performance in cell area, timing_max_slow, and timing_min_fast.

```
                    DLH_X1          NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm
                                                          2.9260  n
-----------------------------------------------------------------------------
Total 61680 cells                                       94420.9567
1
```

Figure 7. The cell area result from cell_report_final report.

```
U71014/ZN (OAI22_X1)                                  0.2522    17.2977 f
U71015/ZN (NAND2_X1)                                  0.2650    17.5627 r
U71016/ZN (NOR2_X1)                                   0.1169    17.6796 f
U71017/ZN (NAND2_X1)                                  0.1527    17.8323 r
U71032/ZN (NOR2_X1)                                   0.0674    17.8997 f
U18002/ZN (NAND2_X1)                                  0.5100    18.4097 r
U71108/ZN (NOR2_X1)                                   0.1926    18.6023 f
U71109/ZN (XNOR2_X1)                                  0.2826    18.8849 f
U71110/ZN (AOI22_X1)                                  0.5771    19.4621 r
U71681/ZN (NOR2_X1)                                   0.1285    19.5906 f
q_state_input_sram_write_data_r_reg[111]/D (DFF_X2)   0.0000    19.5906 f
data arrival time                                               19.5906

clock clk (rise edge)                                20.0000    20.0000
clock network delay (ideal)                           0.0000    20.0000
clock uncertainty                                    -0.0500    19.9500
q_state_input_sram_write_data_r_reg[111]/CK (DFF_X2)
                                                      0.0000    19.9500 r
library setup time                                   -0.3576    19.5924
data required time                                              19.5924
-------------------------------------------------------------------------
data required time                                              19.5924
data arrival time                                              -19.5906
-------------------------------------------------------------------------
slack (MET)                                                      0.0018
```

Figure 8. The slack result from timing_max_slow_holdfixed report.

```
Point                                        Incr       Path
-----------------------------------------------------------
clock clk (rise edge)                      0.0000     0.0000
clock network delay (ideal)                0.0000     0.0000
counter_2_reg[0]/CK (DFF_X1)               0.0000 #   0.0000 r
counter_2_reg[0]/Q (DFF_X1)                0.0632     0.0632 r
U46988/ZN (NOR2_X1)                        0.0205     0.0837 f
counter_2_reg[0]/D (DFF_X1)                0.0000     0.0837 f
data arrival time                                     0.0837

clock clk (rise edge)                      0.0000     0.0000
clock network delay (ideal)                0.0000     0.0000
clock uncertainty                          0.0500     0.0500
counter_2_reg[0]/CK (DFF_X1)               0.0000     0.0500 r
library hold time                          0.0007     0.0507
data required time                                    0.0507
-----------------------------------------------------------
data required time                                    0.0507
data arrival time                                    -0.0837
-----------------------------------------------------------
slack (MET)                                           0.0330
```

1

Figure 9. The slack result from timing_min_fast_holdcheck report.

7. Conclusions

The result shows this design is a nearly fully functional Quantum Emulator with complex values. if it contains n qubits, it can represent 2**n states. Therefore, the design process the test6 in 1566 cycle with 20 ns clock cycle, and the cell area of this design is 94420.9567 um^2.