Final Project Report.

Name: Jujjavarapu Raj Mouli
Unityid: rjujjav
StudentID: 200536086

Delay (ns to run provided provided example). 257640 ns
Clock period: 60
# cycles”: 4294 (test1)

Delay (TA provided example. TA to complete)

Logic Area: (um^2)
134527.3733

Memory: N/A

1/(delay.area) (ns^{-1}.um^{-2})
0.00000743 * 0.0000038

= 2.8234e(-11)

1/(delay.area) (TA)

Abstract

With the number of complex matrices being a variable and their dimensions being a power of two, this project essentially does matrix multiplication. Four SRAMs are included as additional modules; two of them supply the module with input data, and one SRAM stores the final output. We were provided extra SRAM to use for the intermediate-stage storage requirement. To eliminate the stalling of operations necessary for reading and writing from SRAM, I developed a packed array instead of using this SRAM in my design.

Four floating point multiply and accumulate units (fp_Mac) are used in this architecture. These modules are taken out of Synopsys' design-ware library. This design has a number of benefits and drawbacks, which are covered in more detail later in the document. Because of the nature of this design, it is capable of scaling to superscalar execution by adding an even number of Mac modules. The architecture has been limited to four Mac modules due to the SRAM constraint of reading one data in each cycle.

QUANTUM EMULATOR

Jujjavarapu Raj Mouli

Abstract

With the number of complex matrices being a variable and their dimensions being a power of two, this project essentially does matrix multiplication. Four SRAMs are included as additional modules; two of them supply the module with input data, and one SRAM stores the final output. We were provided extra SRAM to use for the intermediate-stage storage requirement. To eliminate the stalling of operations necessary for reading and writing from SRAM, I developed a packed array instead of using this SRAM in my design.

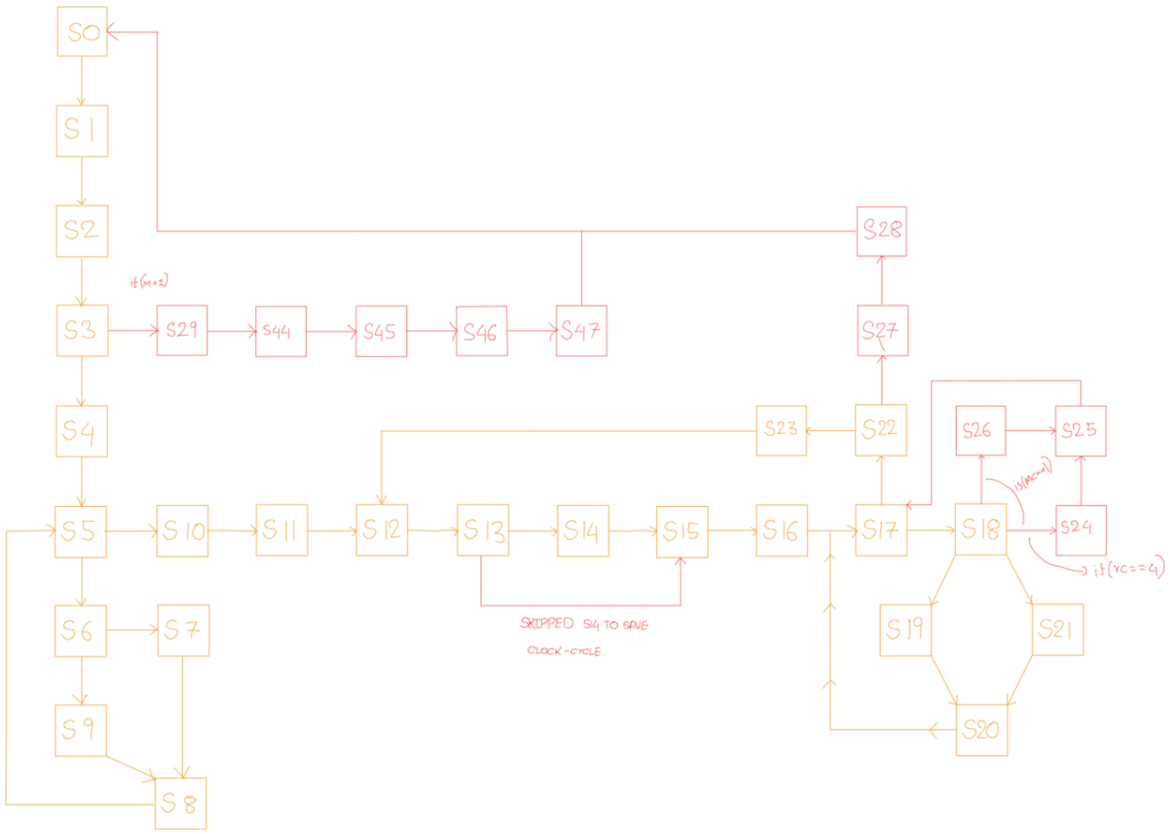
Four floating point multiply and accumulate units (fp_Mac) are used in this architecture. These modules are taken out of Synopsys' design-ware library. This design has a number of benefits and drawbacks, which are covered in more detail later in the document. Because of the nature of this design, it is capable of scaling to superscalar execution by adding an even number of Mac modules. The architecture has been limited to four Mac modules due to the SRAM constraint of reading one data in each cycle.

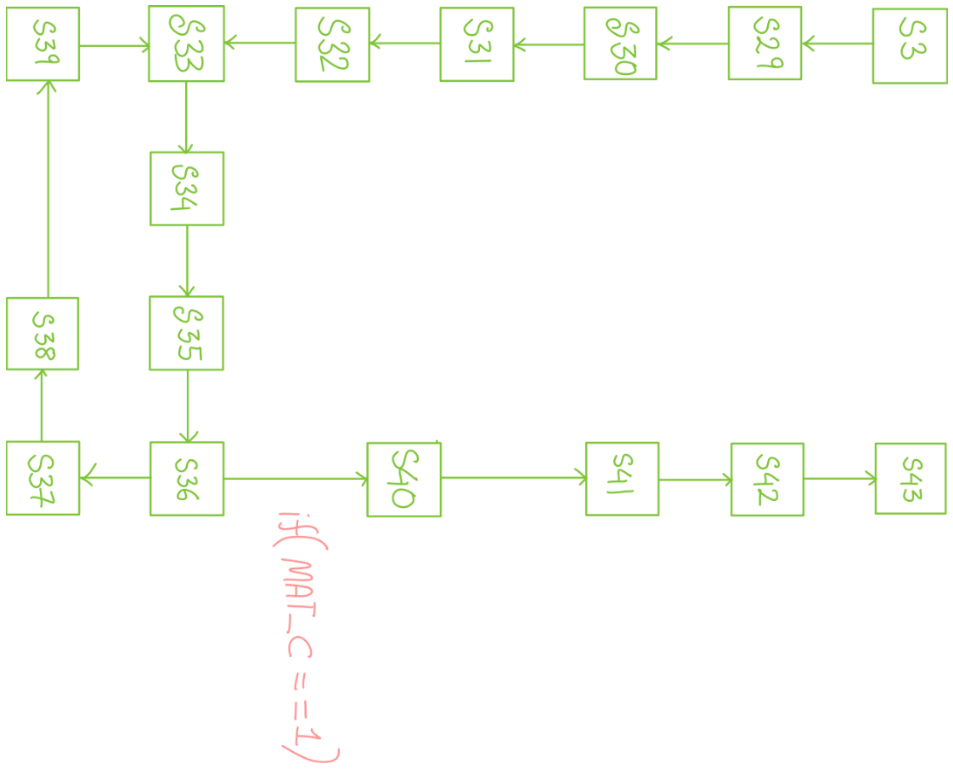
1. Introduction

This design functions as a quantum emulator in terms of applications. All the variables are complex floating point numbers, and at its core, it is a matrix multiplier defined by the hardware. A maximum of 21 matrices with a maximum dimension of 16 were tested, despite the fact that the design can multiply any number of matrices. A 128-bit number, of which the most significant 64 bits are real and the remaining bits are imaginary, is the format in which the data is moved to the module. Since they are floating point values as well, they are further separated into fraction and exponent parts, with the MSB acting as a sign bit.

There are 14 control signals in this design, including those needed to regulate the inputs to packed arrays, counters, and Mac units. Three counters are present in this architecture to monitor multiple matrices, rows, and columns. These counters help the design remember which specific area of these matrices has to be multiplied as well as when to halt the process. The next section includes a comprehensive Finite State Machine-based diagram that shows how the control signals are evolving, along with a list of these signals. Following synthesis, it is evident that the design takes up a lot of area; this is because packed arrays are utilized to prevent stalling. It is more difficult to reduce the clock period further because of the Mac unit's wider critical path. However, we can shorten the clock duration even further by turning on extra registers after each Mac.

2. Finite State Machine



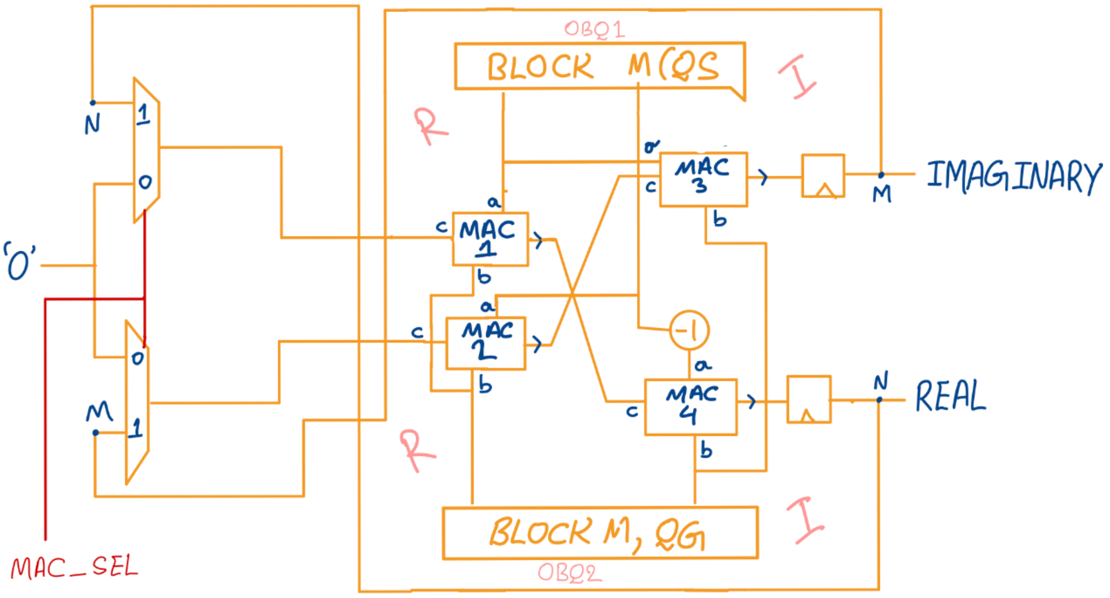


The design's finite state machine has 47 states in total. To make it easier to comprehend, it has been separated into two diagrams using different colors according to the size of Q (matrix dimension-based control). When the matrix dimension is greater than 4, the yellow and pink route is chosen; if it is less than 4, the green route is chosen. If you look at the provided FSM, you will see that I later avoided State 14 because I saw that it was redundant and consumed a cycle. Combining states 13 and 15 also suggests a more optimized strategy, since packed arrays can load and generate data in the same cycle.

A short summary of the control signals utilized is provided here, along with a brief discussion of what components of the hardware they control:

- Q state address select (qs_sel): This is a 2-bit signal that is used to control the Q state SRAM address generation, it increments and resets the address back to 0 or
- Q Gate address select (qg_add_sel): Similar to “qs_sel”, but it can only reset back to 0, as this data is sequentially arranged and that sequence starts from “0”.
- Mac unit selects: There are two slect lines to control the inputs of the mac units. They are “mac_sel” and “mac_i_sel”, these lines enable data from scram and array, while also controlling to accumulate previous value or to discard it.
- Read/write array select: there are 3 control lines under this, they are used to control whether to write data to the array or read from it, or transfer the data between them.
- Counter selects: they include 3 signals, column select, row select, and matrix select. They deal with resetting and decrementing the counter.
- Output SRAM address select and enable: these two signals are required to operate with output SRAM, as our previous SRAMs only require reading of data, their enable is set to default “0”.

3. Main Module Hardware circuit Diagram description



The accompanying diagram depicts a pictorial representation of the matrix multiplication circuit diagram. It has four 64-bit Mac units, with two each for real and imaginary calculations. The MUX controlled by "mac_sel" handles the accumulation of prior output; if set to "1", it accumulates previous output and adds it back to the mac. While "0" indicates that accumulation is not to be performed. Another MUX exists within the "Block MQS" and is used to select between Q state SRAM and the internal packed array.