HW Assignment 3

- Report by Aastha Rajani(2021CS10093) and Piya Bundela(2021CS10118)

In this assignment, we are asked to display the results for matrix multiplication of two 128×128 matrix storing 8 bit number in each index. This is to be implemented with the help of various components integrated together. The various modules which we have defined here includes:

- ROM – We have generated two ROM-files in which we have uploaded the sample COE file uploaded on Moodle for our sake of cross checking. The matrixes which needs to be multiplied must be uploaded in these ROMs in the form of communication files.
  This file takes a address(of 14 bits) as input and returns the data stored at that address.
  Here the data will be of 8 bits.
- Register- This component takes read, write and , reg_cmd as inputs and returns some data which is stored in the register or writes the value given to register. We have defined two different types of registers, one which stores 8 bit input(used for storing inputs) and the other which stores 16 bit inputs(used for storing products).
- MAC – we are using this component to give vector product of respective row and column. To perform this multiplication, it further has 3 components – multiplier , accumulator and register. We input two 8 bit vectors that goes to multiplier. Their product goes to the accumulator and gets stored until multiplier has run 128 times (signifies multiplication of whole row with column) . After multiplier has run 128 times this accumulated value gets stored to register. From register, this

value gets stored in RAM. The above mentioned operations are governed by a mac_cmd variable.

- RAM – here we give inputs (d) via register which gives it product of row and column and the address where the given inputs needs to be stored. We have also given a command write enable as input. When write_enable is set to 1 , the given data is stored at the given address in ram, whereas when it is set to 0, the value stored at that particular address is given as output.
- FSM – This component is the controller unit of all the above mentioned components. We take a clock signal clk as input and define a signal named counter which is incremented at every rising edge of the clock. We have then defined several states namely rom_state, reg_state, ram_state, mac_state, off_state. At every increment of the counter , we check its value mod 136. Initially we start at the off-state, if counter mod 136 is 1 we set it to the rom_state and correspondingly rom_cmd to be 1. This is the time when we extract input from the rom. From this state we then move to reg_state where the reg_1_cmd and reg_2_cmd are set to 1 and the extracted input from two roms is loaded to the registers. We set the read and write inputs of the registers accordingly. Once the registers are loaded, on next rising edge of the clock, we send these stored inputs to the mac where it multiplies the two given inputs and stores the product into a defined signal. This cycle is repeated 128 times until one row and one column is completed and the product obtained every time is added to the previously obtained product. Once we finish operating this mac cycle 128 tiems we move to reg_3_state and set reg_3_cmd to 1. At this state, the obtained value of output from the mac is loaded to register which stores 16 bit.On the next rising edge we load this stored value of register to the ram.

For running the mac cycle 128 times we have defined two nested loops, one which traverses through the rows of the matrix and other which traverses to the columns for every row, and changes the inputs accordingly.

- Input This module extracts the data (16 bit) stored in the ram and gives this 16 bits as input to the main_fun named module which displays the inputs on the basys board (defined in assignment 1).