Final Project: Matrix Multiplication for Image Processing with VHDL (March 2016)

Helen Toma A11727086
Jeremy Scott A11180142

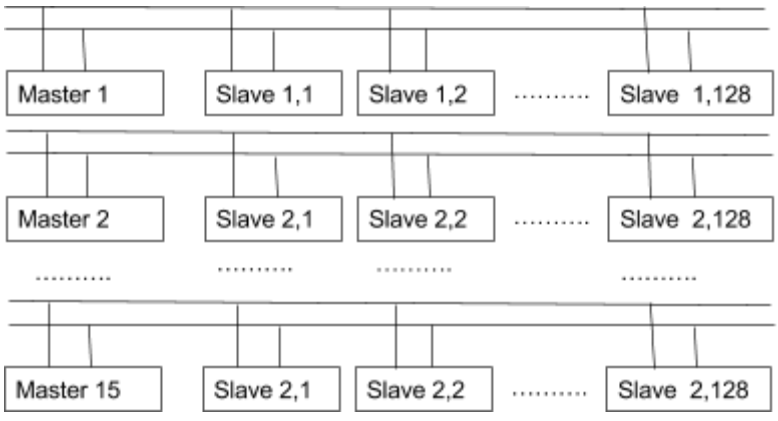
Backstory

We were given a KLT algorithm to build from the ground up. Our ideas regarding design for this project changed considerably over time. It should be noted that our code, and final design ideas here, are not perfectly synchronized. After several scrapped designs, we have failed to solve this problem completely (technical issues, coding skill, etc) and have prepared this report to show our plan, if we were to continue working on this, and progress so far

We attempted to perform matrix multiplication with an I^2C master/slave implementation. Considering Matrix multiplication, we first had to think about what happens mathematically. If Matrix A and B are multiplied together, cell (at row(i),col(j)) of the resulting matrix C denoted $c_{ij} = a_{ik} * b_{kj}$.

There are exactly 1920 such multiplications to do, per c_{ij}. A master can control 128 slaves with 7 bit addressing. So we need 15 masters since 15*128 = 1920.

This was what we envisioned should belong to the coprocessor when we saw that the original reg file had 12 bits, 4 depth, 8 width:



We assumed this meant seven bits for addressing one of one hundred twenty eight slaves, four bits to address fifteen masters (binary 1 to binary 15 where binary 0 meant no master is being addressed). And a final acknowledge bit.

Master Responsibility

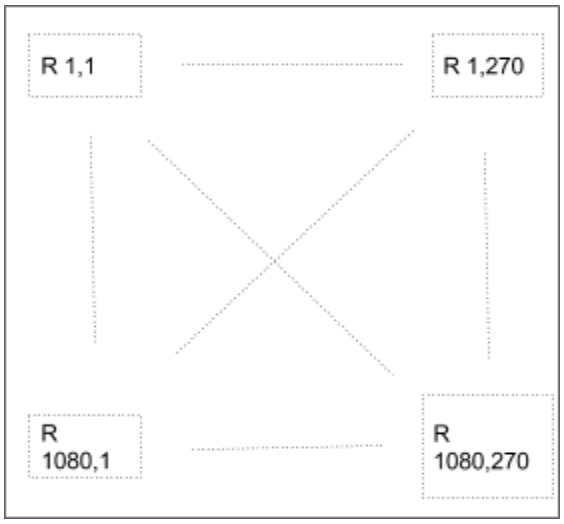
The master is technically a slave to the controller, since it can never work unless the controller grants the master the 'right' to work via 4 bit shift-in addressing. The master is responsible for 1/15th of the total entries in Matrix A and B row/column length. Similar to the controller being responsible for allowing a master to do work, the master is responsible for making his share of slaves to their multiplication work.

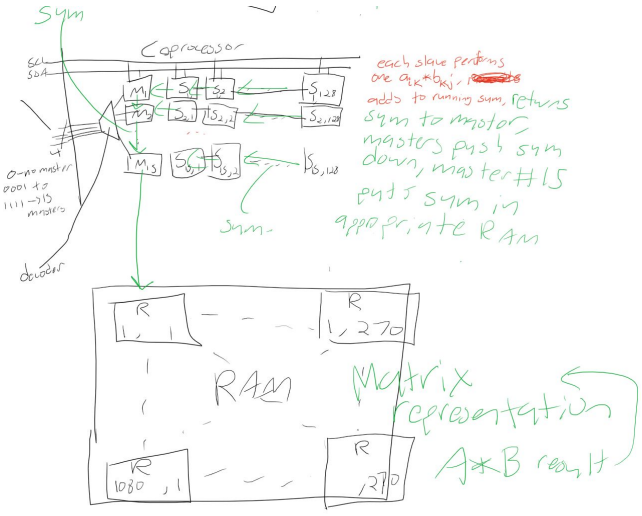
Slave Responsibility

Each slave is responsible for one multiplication and the master keeps track of the sum. As soon as one master is finished with it's 128 slaves, it passes it's sum down to the next master who then calculates products of its 128 slaves. Master 15 is responsible for storing the result into the appropriate cell in the result matrix stored in ram so it should be connected to data_out of coprocessor. There are i*j iterations, (each with 1920 multiplication) to populate the resulting matrix.
I = 1080, J = 270 Ram gets resulting matrix C

RAM

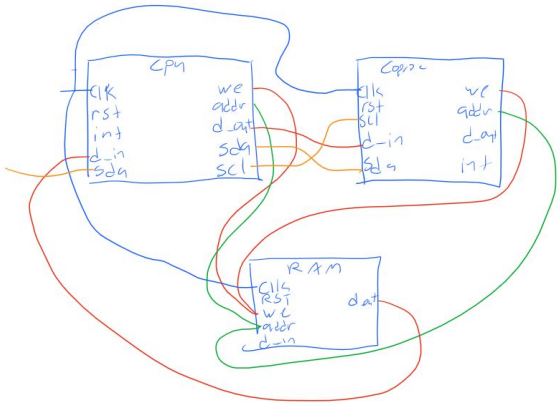
The ram should store the result of the matrix multiplication, so it should have 1080 rows, and 270 columns.
Note that 19 bits is sufficient to address all 1080*270 = 291,600 cells, since 2^{18} = 262144 (not enough) and 2^{19} = 524288(more than enough). Adder is a 23 bit bus, so what are the other 4 bits for? Well there are 15 masters, which could be addressed by these 4 bits, so we can assume perhaps every master is capable of writing to ram after it deals with its row of slaves, and if this is the case, then each register in "RAM" could probably have had an adder inside so that each master just pushes its data into the register that adds the new data to its own value.





Matrix multiplication solved with I²C

We think solving this problem involves having only one register in each slave for one cell in matrix B. The top module should send matrix B's data to the coprocessor to store each cell's data. Next, Matrix A's feeds into each slave one by one, not needing a register, but a multiplier (which could be really fast with a huge lookup table!) so that the previously saved data is multiplied against new data (from matrix A), its sum is added to its the master, and the address to the next slave loads to do the same. By the time we finish with slave 128, this master sends its sum down to the next master who continues the same process. Master 15 finally sends its sum to the appropriate location in RAM.



Masters and Slaves in I²C

Each master and slave read in one bit of information at a time. This requires the use of shift registers with 4 and 8 bits for masters and slaves respectively.

How the master works:

The 4 LSB's are shifted to the right one by one, each master receiving *every* bit. Each master has it's own "address" register, so each master checks the shift register against it's own address. After 4 bits come in, some master opens gates for controlling its slaves.

How the slaves work:

Bits 5 through 11 shift into every slave of one master, one at a time per scl cycle, and is equal to sda. Each slave has a 7 bit address register to compare against the shift register, as soon as the bits match the address the slave only allowed to acknowledge the masters order if the next incoming bit is 1 (acknowledge).

While a slave is performing calculation, it holds down the scl line until it is finished so that its computation is not ignored, and prevents the system from attempting to force more data into it.

Microcontroller

The CPU is our microcontroller and is responsible for loading up the coprocessor with matrix data, for sending data from a new matrix to be multiplied in the coprocessor, and for controlling whether or not ram is being written to or read from. We assume that FSM's are in the microcontroller to control the iteration cycles through the multiplication algorithm. But we are bad at math and coding vhdl, and this is as much as we understand.

Performance comparison: Hardware Vs Software

Matrix multiplication is software is easier to implement, so we would pay these developers less because they can finish their work faster. However programming matrix multiplication with one thread in software does not take advantage of concurrency offered by hardware implementations.

What if it were possible, for example, for a camera to act like a human eyeball, to have thousands or millions of little cameras that could instantly fill a matrix of pixel data. Imagine we have a huge lookup table for every possible multiplication between two pixel data. Then for our purpose here we could perform the almost 560 million calculations in 19 clock cycles.

Here are the total number of multiplications needed (not including summing) 1080*270*1920 = 559,872,000

If 1080*1920 could be filled instantly by snapping a picture with this hypothetical camera, then we are left with 1080*270 multiplications = 291600 of them. Since we have a huge multiplication lookup table, these all take one time step. Then log(291600)/log(2) = 18.15 -> 19 clock cycles to perform matrix multiplication on an image. We could process streaming video playing thousands of times faster than standard speed. With this, it would be easy to feed all videos ever made into a classification algorithm, and have it finish in a reasonable time!

Conclusion

The I^2C protocol allows very clean communication between controllers masters and slaves. The wires are easy to set up and connect between components, however the actual vhdl implementation was very challenging and for us, unsolvable.
The project seemed relatively easy to understand at a high level, but we got so lost in the details, trying to understand the skeleton code and other implementations.

Note: We are both parents with young children, and feel our responsibilities for family and school are too much already to add this type of project to finish in such a short period of time. These types of coding projects that we don't see ourselves ever applying for a job to do. We enjoy the high level details of the class, but the vhdl learning and coding work, was far too much to learn while we were already swamped with other project class for software engineering. To be perfectly honest, VHDL does not resonate with either of us, we are too busy with the family and responsibilities, and cannot learn new tricks in such a short amount of time piled into already too-hard classes. Many other students have more time to put into the project because they do not have children or spouses to take care of while attending school. This is not to say that we did not work on the project. We actually put it so much more time and efforts than we initially planned to because we wanted to get the best out of it and get it working. But even with that, the project was simply too hard and long to finish. Given even another month, we do not feel like we could solve this problem while maintaining other classes and responsibilities.