COL215P Assignment 2 Stage 1

Aniruddha Deb 2020CS10869

Sachit Sachdeva 2020CS10840

September 2022

Entities

ROM

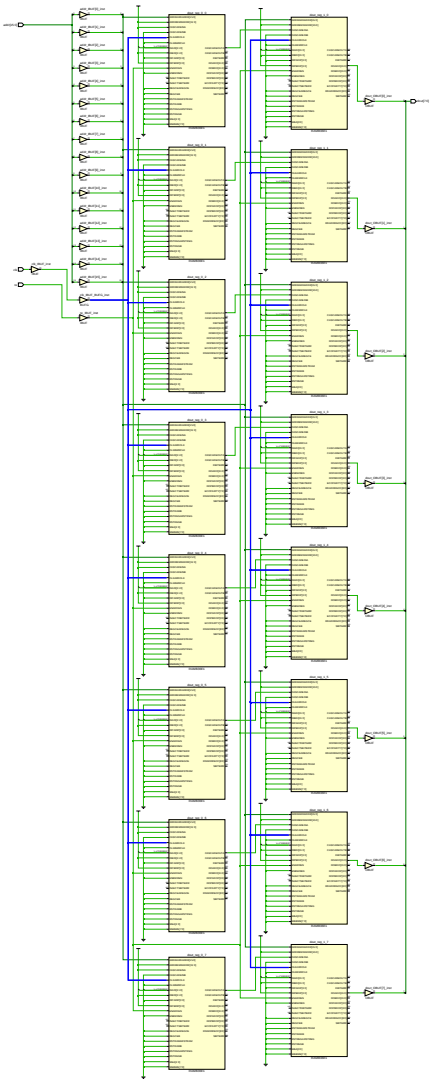


Figure 1: ROM synthesis

A 64 kB ROM was designed, split into two parts: 1kB for the image and the remainder for the weights and biases. The ROM was synthesized using 16 BRAM modules, as shown on the left. To syntesize to BRAM, a synchronous design was chosen for the ROM. Reading from it takes 1 clock cycle, and this design decision will be implemented in the controller when we construct it.

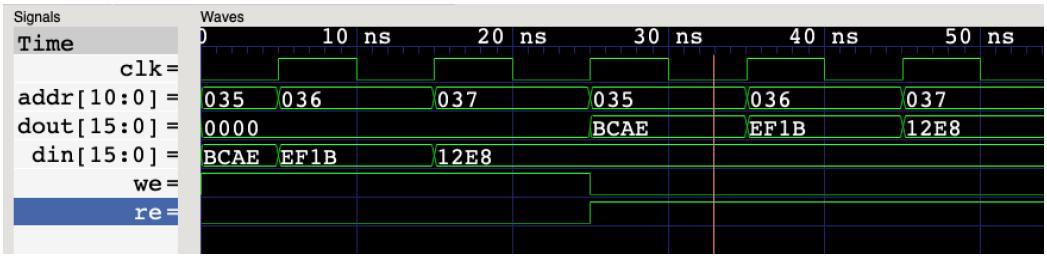
As informed in the problem statement, the image file starts at address 0x0000 and takes 784 bytes. The weights and biases start at 0x0010 and take 50890 bytes. The remainder of the ROM is empty. To initialize the ROM using two separate MIF files, the following code was used:

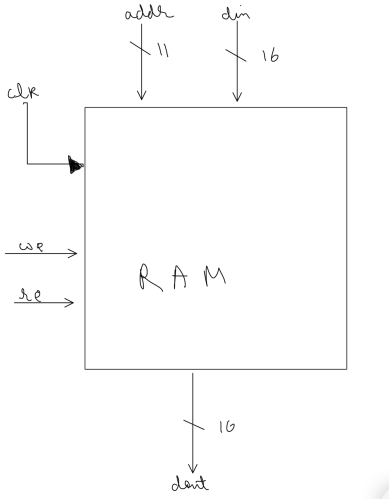
```
impure function read_rom(
        img_file_name : in string;
        wt_file_name : in string
    )
    return rom_arr is
    file img_file : text open read_mode is img_file_name;
    file wt_file : text open read_mode is wt_file_name;
    variable mif_line : line;
    variable temp_bv : bit_vector(DATA_WIDTH-1 downto 0);
    variable temp_mem : rom_arr := (others => x"00");
begin
    for i in 0 to 783 loop
        readline(img_file, mif_line);
        read(mif_line, temp_bv);
        temp_mem(i) := to_stdlogicvector(temp_bv);
    end loop;
    for i in 1024 to 51913 loop
        readline(wt_file, mif_line);
        read(mif_line, temp_bv);
        temp_mem(i) := to_stdlogicvector(temp_bv);
    end loop;
    return temp_mem;
end function;
```

.

RAM

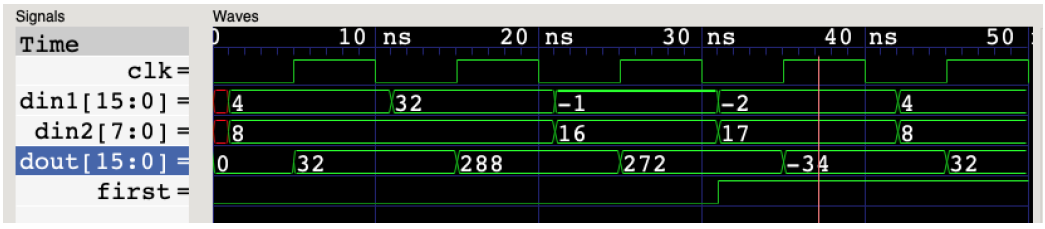
Ram is internally made as an array of 2048 words(each word 16 bit). The ram has synchronous write(hence the clk is given as input) and asynchronous read. There is a read and write enable signal given to ram to enable/disable read and writes. The addr is an input signal of size 12 bits to index for a word in RAM. The ram takes input the 16 bit word it needs to store, and gives 16 bit read output

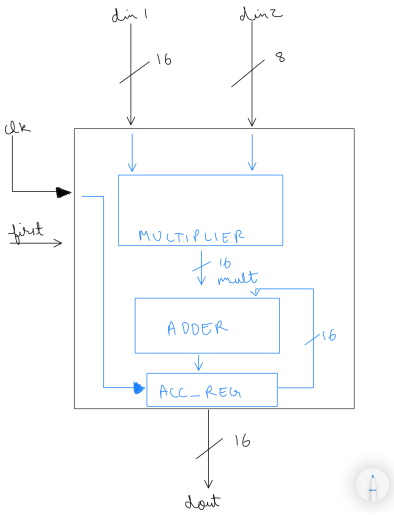




MAC

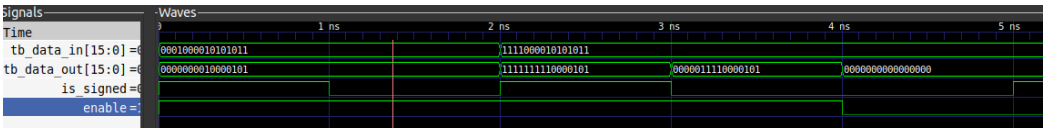
MAC, internally consists of a multiplier, an accumulator, and an accumulator register. The multiplier takes 16 bit and 8 bit input, does a signed multiplication and gives a 16 bit output (after truncation). This output and the current value of accumulator register are added by the adder. The adder's output is fed to the the accumulator register, which would be updated at the next rising edge

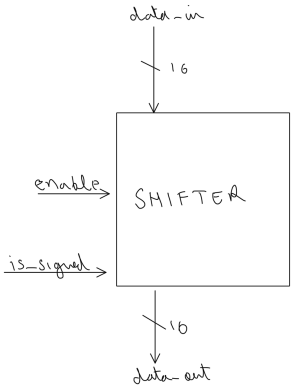




Shifter

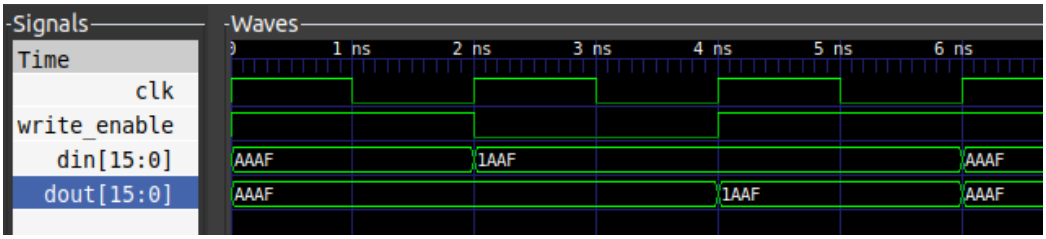
Shifter has an enable signal to enable/disable the shifter. There is another 1 bit signal is_signed which would be '1' in case we want to do signed_shifting. The shift is by a factor of 5 bits (division by 32), with or without sign extension.

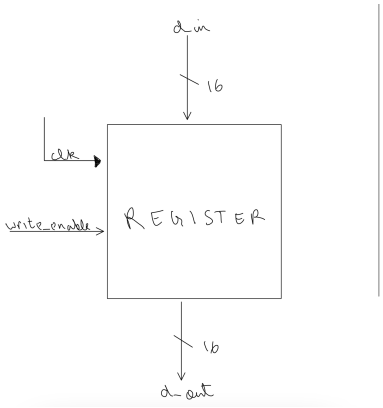




Register

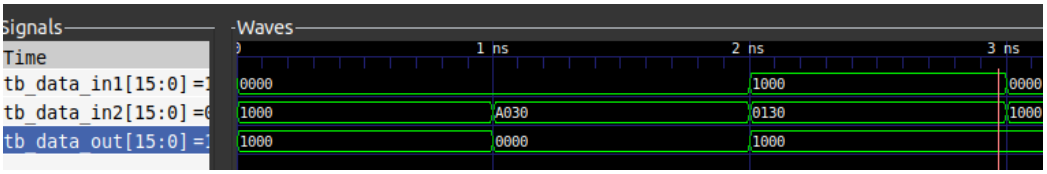
Register takes a signal write_enable, to enable/disable the register. When enabled the register data is updated with the input data on rising edge of clock

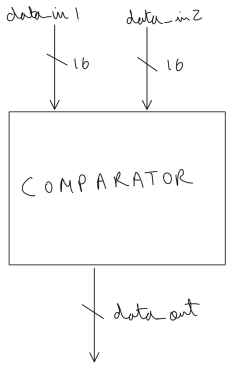




Comparator

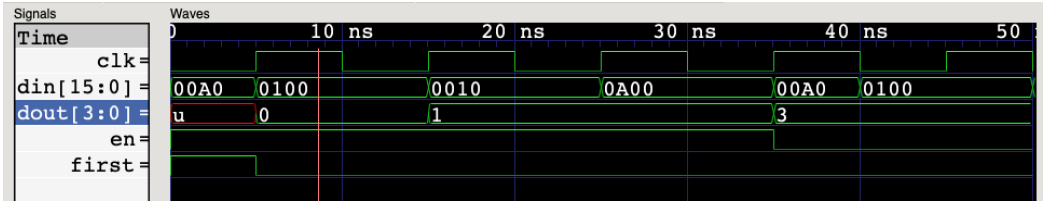
Comparator takes in 2, 16 bit inputs, compares them(signed comparsion) and returns the one that has larger value. This is different from a standard comparator that returns a boolean indicating the result of comparator





Argmax

The Argmax component outputs the index corresponding to the maximum of a stream of numbers passed to it. This is used after all the inference is done, and we need to display the class to which the image belongs.



LED Driver

The LED driver drives a single 7 segment display. If active is high, it displays the digit on it's input on the display. If active is low, it displays a dash (-) to represent that the model is currently processing information.