Preliminary Design Review

Perceptron – ECE 5020

Braden Redick

Gage Farmer

Jason Forrester

William Huang

Summary

The perceptron is comprised of three major building blocks: the SRAM memory array, the 8-bit full adder, and an 8-bit output multiplier. To divide the work evenly, each building block will be individually completed by one of the members. Since Team 16 has four team members, the most difficult block will be assigned a team of two. During the design, each block will be tested for functionality and performance. Once each building block's designs are complete, they will be integrated together at the top level. After successful testing and evaluation at the top level, the project will be complete.

The team decided the SRAM memory array will likely be the hardest block to design. Therefore, the SRAM memory array will be assigned to the team of two. Jason Forrester and Braden Redick are tasked with completing the SRAM memory array. This leaves Gage Farmer to complete the 8-bit full adder, and William Huang to complete the 8-bit output multiplier.

To stay on schedule, each member must complete their assigned block by **November 19th**. This will give the team one week to integrate the blocks at the top level. If a team member becomes unable to meet this deadline, they must notify the team as soon as possible. Weekly meetings will be held to coordinate the design and ensure the team stays on schedule.

SRAM Memory Array

The SRAM memory array is shown in Figure 1. The SRAM memory array requires five control variables: A0-2, Row/Column_Bar, and Read/Write_Bar. A0-2 are memory addresses that select which memory block is being accessed. Row/Column_Bar is used to toggle between the row and column decoders when reading in the address data. And Read/Write_Bar toggles between reading from and writing to the selected memory block. The 6-bit data bus is used to transfer the weights stored in the memory blocks to the multipliers.

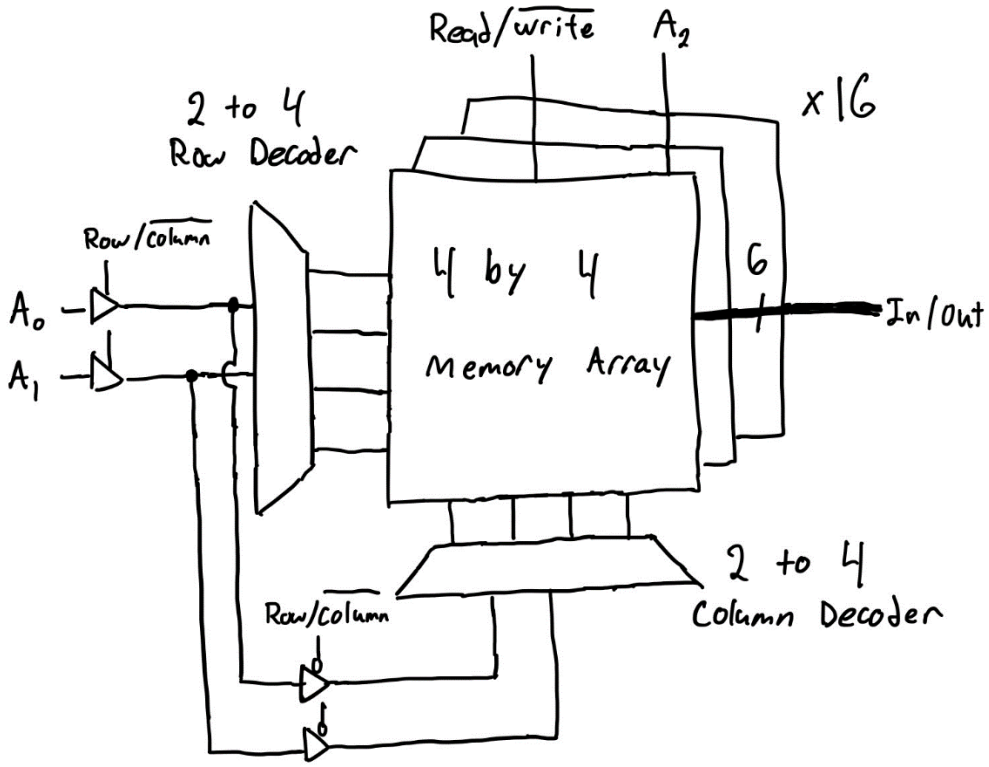

Figure 1. SRAM Memory Array – Top Level Schematic

As shown in Figure 2, a 6-bit bus is used to transfer the weights. Since each weight is only 3-bits, 2 weights are transferred at a time. This also means four of the 4 by 4 memory arrays are unused. This is an unavoidable inefficiency of this design, because the project requires the memory array to be able to store thirty-two, 8-bit words. Since there are only sixty-four, 3-bit weights used in this design, one quarter of the memory array will always be unassigned regardless of the architecture.

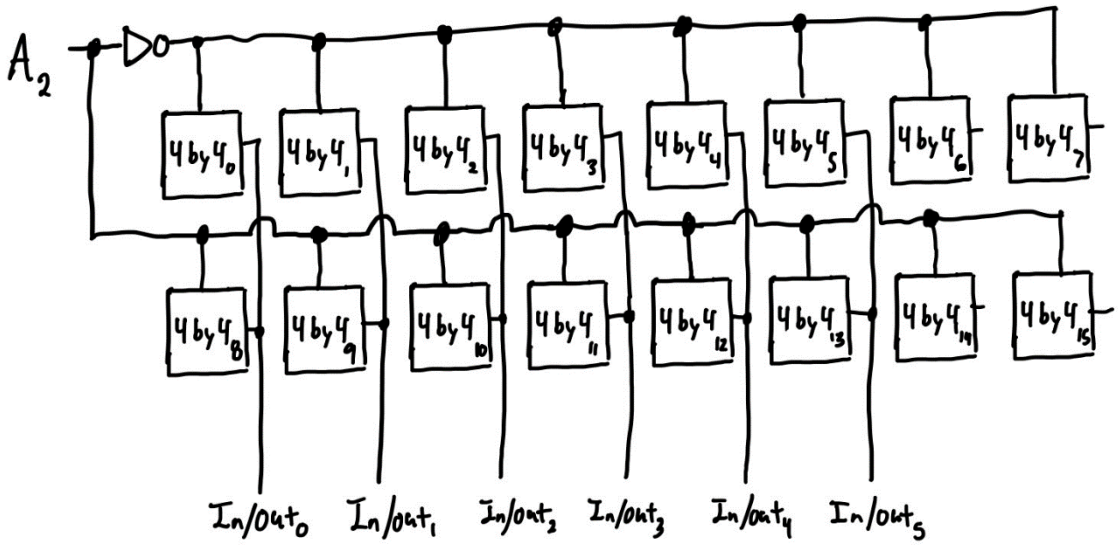

Figure 2. SRAM Memory Array – Input / Output Data Bus Configuration

Figure 3 shows the symbol for the memory blocks. Each memory block will have pins: Read/Write_Bar, Row_En, Column_En, Depth_En, and In/Out. Read/Write_Bar will toggle between reading and writing data. Row_En, Column_En, and Depth_En are all enable pins that determine if the memory block is being accessed. Row_En connects to the row decoder, Column_En connects to the column decoder, and Depth_En connects to A2. Lastly, In/Out is used to access the data within the memory block. The In/Out pin will connect to the In/Out data bus.

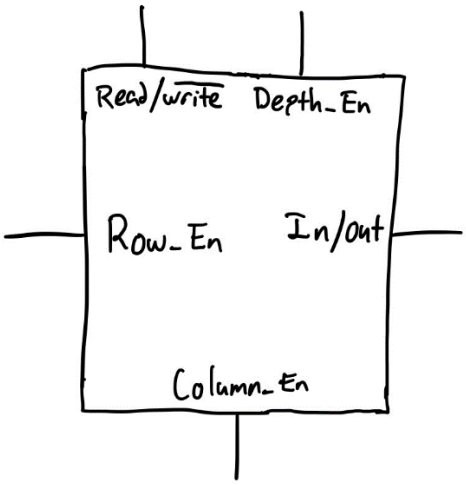

Figure 3. SRAM Memory Block – Symbol

Figure 4 shows a schematic drawing of the memory blocks. Each memory block is made up of two, 4-input AND gates, two tri-state buffers, two inverters, and an SR latch. Each AND gate is assigned to a tri-state buffer on the input or the output of the SR latch. The In/Out pin is connected to the input of the input tri-state buffer, and the output of the output tri-state buffer. The input tri-state buffer will only be on if Row_En = Column_En = Depth_En = 1 and Read/Write_Bar = 0. The output tri-state buffer will only be on if Row_En = Column_En = Depth_En = Read/Write_Bar = 1. This allows the memory block to toggle between reading a writing, which allows for more stable operation.

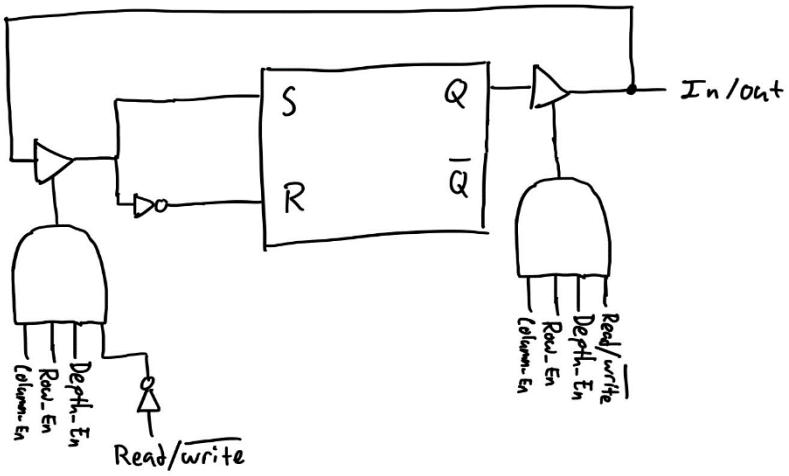

Figure 4. SRAM Memory Block – Schematic

Barrel Shifter

The 8-bit adder has two 8-bit inputs A(0:7) and B(0:7), a carry input Cin, a carry output Cout, and finally 8-bit output S(0:7). The corresponding A and B input for each binary position is sent to its own 1-bit full adder, as shown below in Figure 5, along with a Cin if applicable.

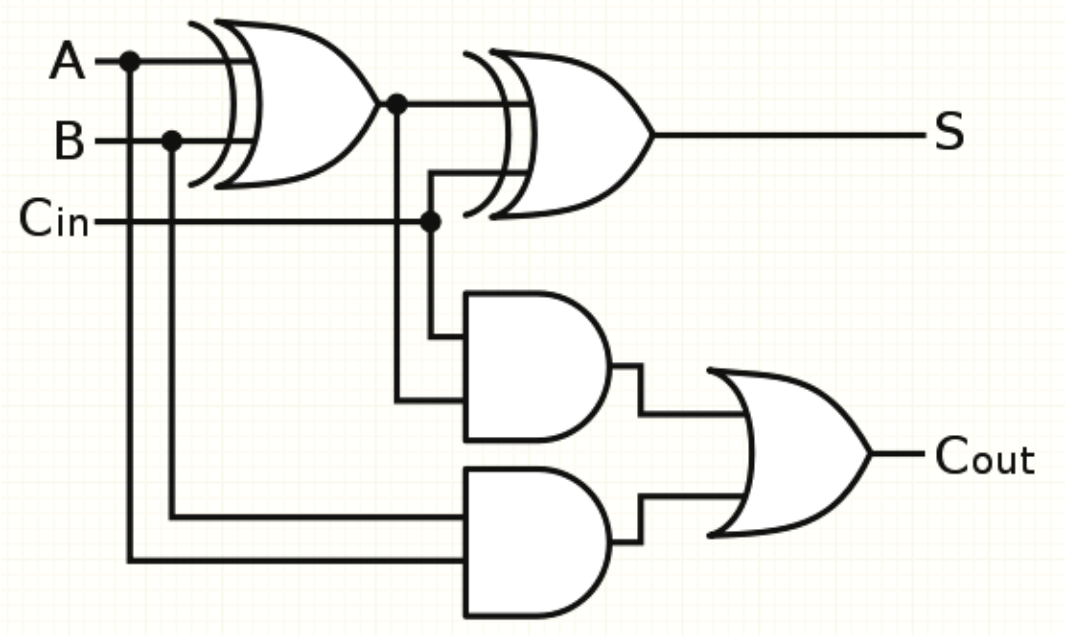

Figure 5. 2-Bit Full Adder Schematic

The full adder shown above will be used for each bit of the 8-bit adder, in which Cin will enter through the LSB adder. The Cout will move down the line towards the MSB in which it will be Cout of the entire 8-bit adder, as shown below in Figure 6. This adder is made up of two XOR gates, two AND gates, and an OR gate.

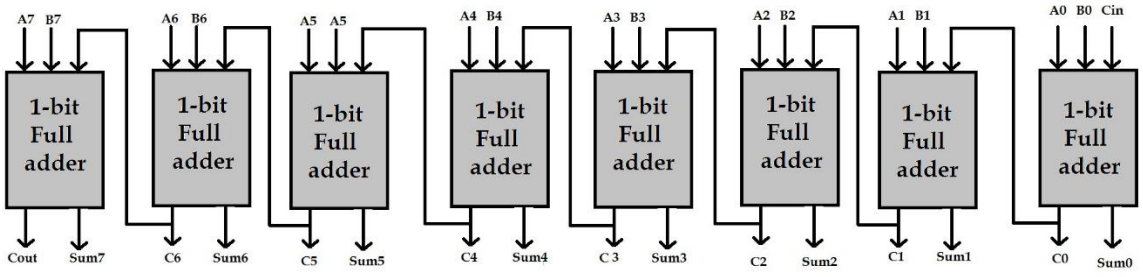

Figure 6. 8-Bit Full Adder Block Diagram

In Figure 7, we show what the full schematic will look like with each individual adder wired together, with the proper path shown for the carry bit from start to finish. In total there will be 16 XOR gates, 16 AND gates, and 8 OR gates. It will take two 8-bit inputs and 1 carry bit and output 1 8-bit sum and 1 carry bit.

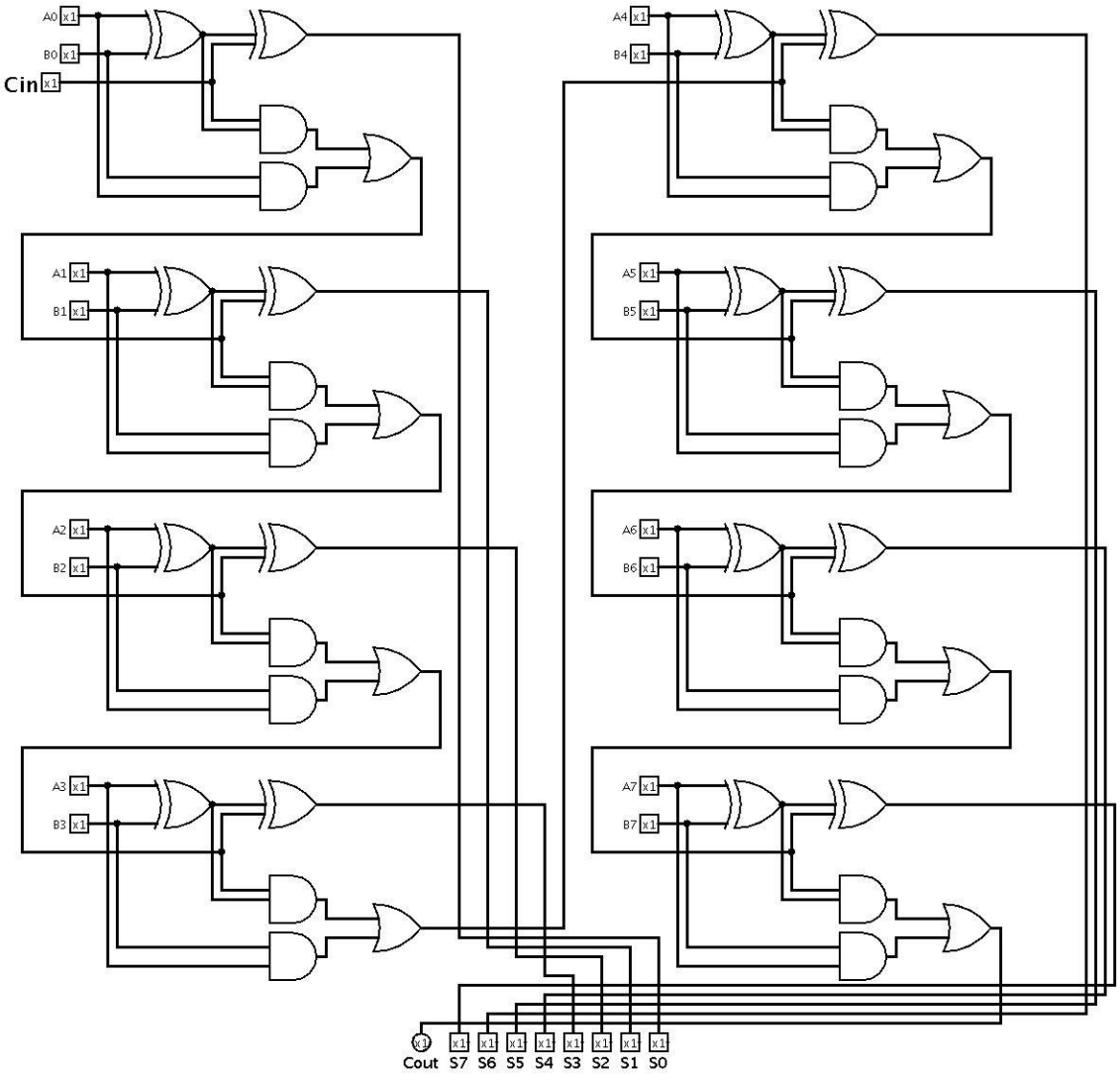

Figure 7. 8-Bit Full Adder Schematic

8-Bit Multiplier

A multiplier is needed for the design. Based on the possible value of the weight, it is possible to employ a shifter in place of an actual multiplier. Based on the material, analysis, and the possible value of the weight shown in class, 3-bits are required for the shift to be properly executed. The design below employs a shifter that can act as the multiplier for this design.

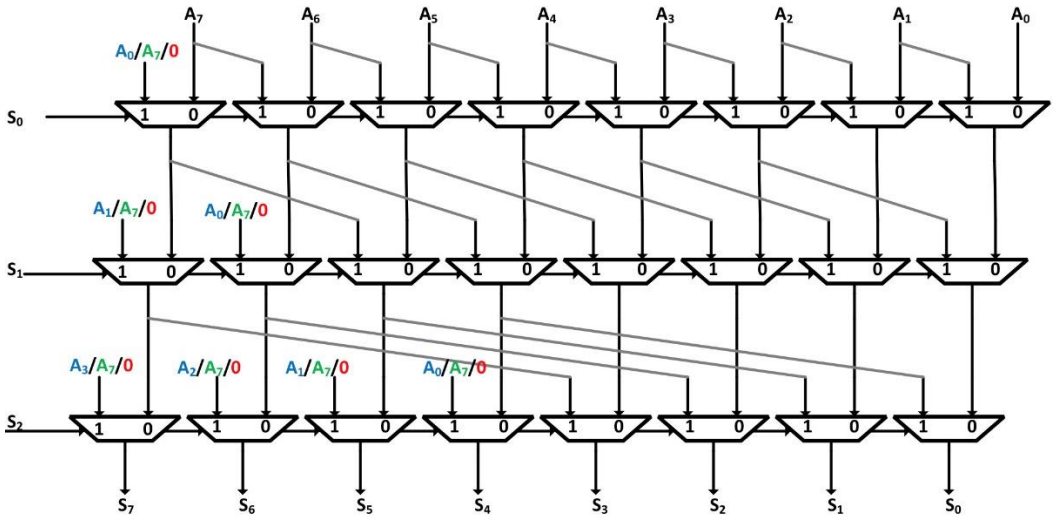

Figure 8. Shifter Block Diagram

Below is the design for the 2:1 MUX used in the shifter.

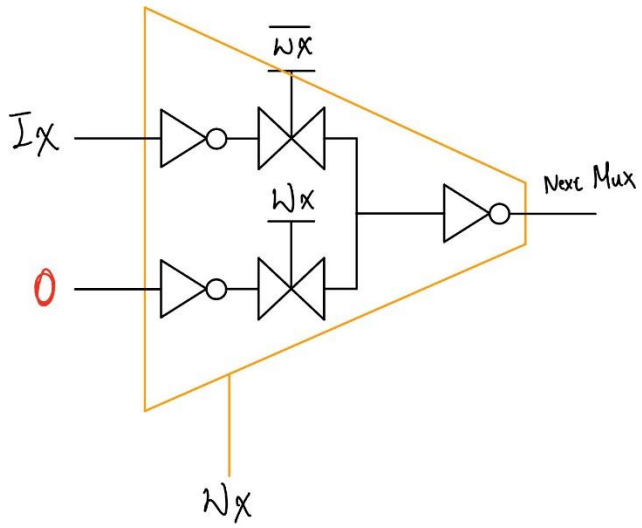

Figure 9. 2:1 MUX