COMPENG 3DQ5: Digital Systems Design Take-Home Exercise 4

10/20/2021 - Thursday - Group 55
William Siddeley 400245905
Mohamed Al-Asfar 400262489

The first exercise involved modifying the program in experiment 3 in order to design the circuit that computes the arrays Y and Z, as defined in the lab document. We began by modifying the state machine and adding in new states as a result of the additional capacity within 3-bits (S_READ, S_WRITE, and S_LAST_READ). We then added in logic variables to store the address data being read and written (address_ram and address_ram_shifted). The variable address_ram_shifted represents the value of address_ram except incremented by an additional 9’d256, so that when reading / writing, we simultaneously read / write to address *k* and *k* + 256. Next, the given write and read data variables from the RAM instances were assigned to define the Y and Z arrays (e.g. write_data_a[0] = read_data_b[0] - read_data_a[1]). The rest of the exercise involved using the always_ff block in order to either read or write by assigning write_enable either high or low depending on which state was being cycled through (S_READ or S_WRITE). Finally, the states S_LAST_WRITE and S_LAST_READ were used to complete the final 2 clock cycles to store the last index of data.

In terms of register use for this lab, we estimate that our register use is low, since unlike the previous lab, to read and write data to the dual-port RAM, we do not need very many registers to hold our data. Our estimate of the registers that were used is 22 registers. This accounts for the 18 registers required to hold the 9 bits (each) of address_ram and address_ram_shifted. In addition, the 4 registers that are needed for write_enable_a and write_enable_b. The Quartus compilation summary reports that there are 27 registers in use, which is 5 more registers than our estimate.

To inspect the critical path, we used the timing analyzer to inspect the worst case timing paths for clk_50. However, when looking at the timing paths, all the slack values are positive. Slack is the difference in time when data is expected and the actual time it arrives. Since our slack values are positive (13ns - 18ns), this implies that our design does not have timing violations and actually has a margin of time where data is expected before it arrives. Therefore, our critical path is the worst case slack, which is ~13ns.

Finally, our design uses 88 logic elements according to the Quartus report. Our most reasonable explanation is that when looking at the Netlist Viewer, we can see that our design has many adder elements as well as both write_data variables (which is 8 bits * 4 logic elements), as well as address_ram and address_ram_shifted, (which are 2 * 9 bit logic elements) that are used as input for the dual-port RAM instances. Lastly, with the multiplexers that serve as if statements in our design, and the addition of other elements like the 9 green LEDs on the board, our estimate begins to approach a number close to what Quartus provides for the number of logic elements in this design.