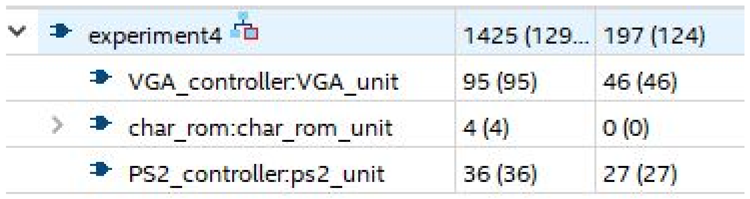
Lab 3 Report

Frank Shan
Mark Naguib

For this lab, there are 197 registers used in total. 124 registers are from experiment 4 file, 46 registers are used from VGA_controller, and 27 registers used from PS2_controller. The registers in VGA_controller and PS2_controller are exactly the same as the ones in experiment 4. As for the 124 registers in experiment 4 file, PS2_reg has 8 bits for each one, and since there are 15 PS2_reg thus there are 120 registers from PS2_reg. The other 4 registers are from data_counter.

The quartus report:



experiment4	1425 (129...	197 (124)
VGA_controller:VGA_unit	95 (95)	46 (46)
char_rom:char_rom_unit	4 (4)	0 (0)
PS2_controller:ps2_unit	36 (36)	27 (27)

Figure 1 from the lab manual shows that the vertical blanking interval occurs from 480us to 523us and this is what pixel_x_pos and pixel_y_pos are based off of. While Figure 2 shows what's in the actual VGA where Vcont and Hcont are without the added delay to create the interval. The board elements file was manipulated so that only during the vertical blanking interval, keys were actually pressed and let go in that timeframe.

For the code portion of the exercise, it was instructed that the final iteration of code had to show from the 15-character message which numerical key appeared the most. If a non-numerical key was pressed, it would be considered as a space being inserted. To implement this, there were case statements to distinguish between what was pressed by looking at the PS2_reg variable. This determined whether a number or a space should be asserted to that position in the register. After that was done, an always comb block was implemented to find which key was pressed the most and had the higher value if they were pressed the same amount. Firstly, counters for each number were set to 4 bit zeros, then a for loop was created to go through the PS2_reg and go through if statements to see which counter should be updated. Finally, another mass of if statements were used to assert the hierarchy of the key values so that the highest key value would take precedence over others(i.e 9>8>7…>1).

If we were to do something differently, we would have gotten rid of the for loop that goes through the register to update each number counter as well as the pile of if statements at the end. WE would have wanted to do this to use less logic elements as well as adding a larger propagation delay to the code. We would try to implement sequential logic but at this time we were unable to fully understand how to do so. It was verified in class that for loops are only useful in certain situations and we are always looking for a more efficient and effective method to better our results.