Lab 5 Report

Frank Shan
Mark Naguib

State diagram

states	0	1	2	3	4	5	6	7	8	9	10	11	12	13	6	7	8	9	10	11	12	13
pixl_x_pos																						
SRAM_addres	red	green	blue_e		red	green	blue_o		red	green	blue_e		red	green	blue_o		red	green	blue_e		red	green
VGA_data[2]				D0(R0,R1)			D3(R2,R3)					D3(R4,R5)			D3(R6,R7)					D3(R8,R9)		
VGA_data[1]					D1(G0,G1)			D4(G2,G3)					D4(G4,G5)			D4(G6,G7)					D4(G8,G9)	
VGA_data[0]						D2(B0,B2)			D5(B1,B3)					D5(B4,B6)			D5(B5,B7)					D5(B8,B10)
red_buff							R1				R3				R5			R7				
green_buff							g1								G5							
blue_buff							B2								B6							
VGA_red							R0			R1	R2	R3			R4				R5	R6	R7	
VGA_green							G0			G1	G2	G3			G4				G5	G6	G7	
VGA_blue							B0			B1	B2	B3			B4				B5	B6	B7	

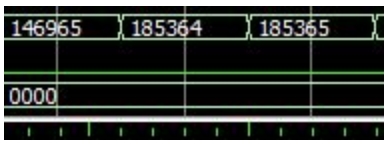
Above is the state table we had created for the exercise different from the experiment 1 table done during the lab. Here we had added in 4 more different states:

S_FETCH_PIXEL_DATA_4,
S_FETCH_PIXEL_DATA_5,
S_FETCH_PIXEL_DATA_6,
S_FETCH_PIXEL_DATA_7,

We had decided to create 3 buffer variables so that we could first take values from the odd blue memory location and then values from the even blue memory locations. We also changed around the original code to match our current state table by making the first state of S_FETCH_PIXEL_DATA_0 to put R1,G1,B1 into respective buffer variables so that they are not overwritten and it also grabs the address for D3 and returns the values R0 G0 B0. Then the next state grabs D4 address and then the next state gets D5. For S_FETCH_PIXEL_DATA_3, this returns the values for R1 G1 B1. Finally in the new states that we created, we had first saved the R3 value in a buffer and returned R2 G2 B2. In the next state, we had grabbed the address for D3 and returned R3 G3 B3 and then D4 address was grabbed and finally D5's address was used. This was then looped until the final SRAM address was reached.

We were not able to get the picture that was required, the issue that we found was that when we write into the data, the data was not equivalent to the data read from the sram_address.

Writing into the data



Read from the data