Krestin Ebraheem, U#50717721
Joseph Daniel Moreno, U#60156204

Assignment 6

Notes on the zipped files:

- Two project folders: “A6_problem1” and “A6_problem1 BEFORE OLED”
  - **“A6_problem1” is the actual submission**; it will generate the bitstream for the ZedBoard.
  - “A6_problem1 BEFORE OLED” has VHDL files set up to show the sorter is working correctly in **simulation**.

- Regarding both project folders…
  - sorter.vhd is the top module.
  - RAM is initialized with numbers in ram_8x16.vhd (“A6_problem1” has different numbers from “A6_problem1 BEFORE OLED”).

- Regarding “A6_problem1”…
  - Two buttons for input: *reset* is BTNC and *go* is BTND.
  - One LED for output: *done* is LD0.
  - **When the bitstream is downloaded on to the board**, press *reset* to set the sorter’s state to s0 then press *go*; *done* will light up when the sorting is done and the sorted numbers will be displayed one by one on the OLED display.

- Regarding simulation in “A6_problem1 BEFORE OLED”…
  - In sorter_tb.vhd, *tempOut* will output the numbers in sorted order once *t_done* == ‘1’.
  - Simulation time has been set to 6μs.

Krestin Ebraheem, U#50717721
Joseph Daniel Moreno, U#60156204

FSMD Table:

Current State	Next State	Condition	Operation(s)

s0	s1	go == 1	count = 0;
			done = 0;
			sort_ready = 0;

s1	s2	count != 15	i = count;
			j = i + 1;
			ram_addr = i;

s1	s9	count == 15	j = 0;
			ram_addr = j;
			done = 1;
			sort_ready = 1;

s2	s3	(no condition)	A = ram_out;
s3	s4	j != 16	ram_addr = j;
s3	s1	j == 16	++count;
s4	s5	(no condition)	B = ram_out

s5	s6	B < A	ram_in = A;
			write_en = 1;

s5	s3	B ≥ A	++j;

s6	s7	(no condition)	ram_addr = i;
			ram_in = B;

s7	s8	(no condition)	A = B;
			++j;

s8	s3	(no condition)	write_en = 0;

s9	s10	(j < oled_r) && (j < 16)	++j;
			sort_ready = 0;

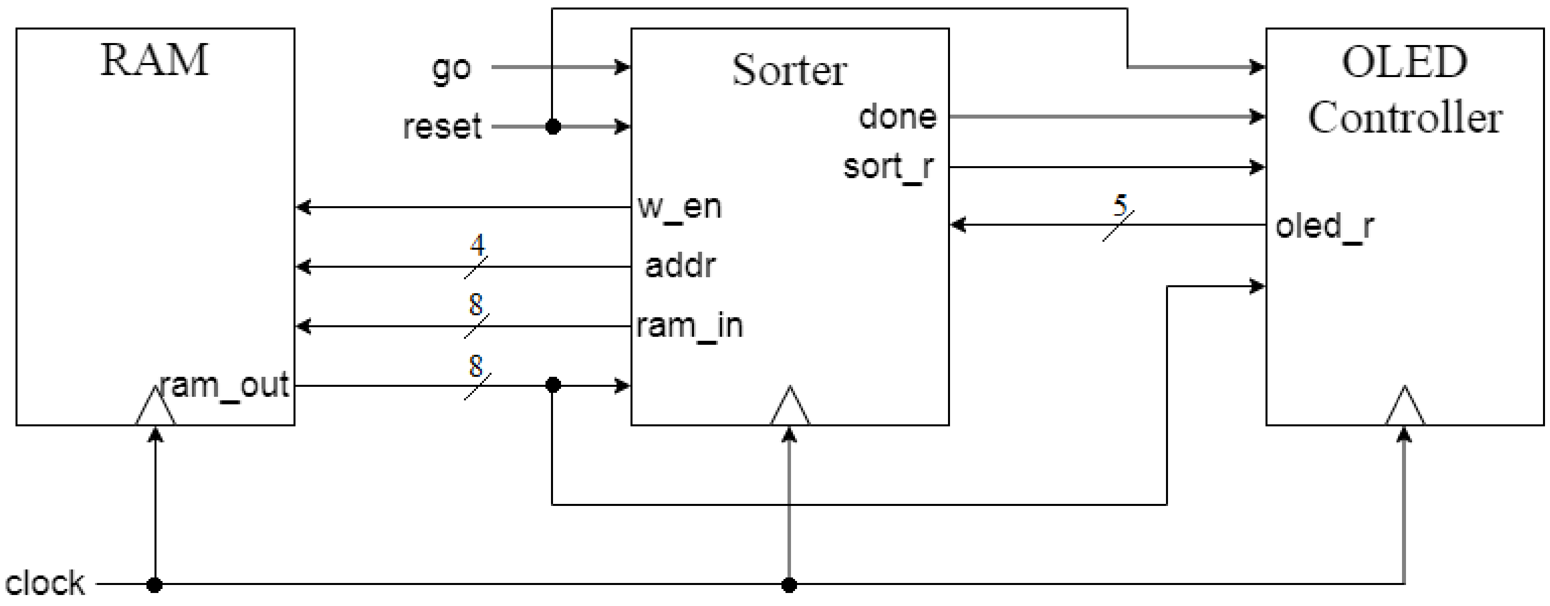
s10	s11	(no condition)	ram_addr = j;
s11	s12	(no condition)	sort_ready = 1;
s12	s9	(no condition)	(no operation)

Side Notes:

- If reset == 1, the next state is s0 regardless of the current state.
- i and j are used to hold memory addresses and load the values from those addresses into A and B, respectively (A = mem[i]; B = mem[j]).
- ram_addr is the memory address currently being read or written to (if write_en == 1).
- oled_r is a 5-bit number used to synchronize the OLED display and the sorter; the sorter needs to wait for the OLED to display the current ram_addr's number before incrementing ram_addr by 1, and after that sort_ready becomes 1 to tell the OLED controller ram_addr is ready to be read from.

Krestin Ebraheem, U#50717721
Joseph Daniel Moreno, U#60156204

Block Diagram:



Interface Explanation:

Inputs	
go	When ‘1’, sorter starts sorting numbers in the RAM
reset	Resets sorter to a halted state and resets OLED to an idle state; done = 0 and sort_r = 0
ram_out	8-bit number read from RAM for the sorter to evaluate
oled_r	5-bit number from OLED controller to synchronize the ram_out address and the OLED’s display
Outputs	
w_en	When ‘1’, ram_in is written to RAM[addr]
addr	Specifies 4-bit address being read/written from/to in RAM
ram_in	When w_en == ‘1’, 8-bit number in ram_in is written to RAM[addr]
done	‘1’ when sorting is done (all 16 numbers in RAM have been sorted)
sort_r	‘1’ when address is ready to be read from (for the OLED controller)