Lab 8: LED CPU with ROM & RAM

What is LedCPU

This simple CPU is one that has a simple mechanism that we will describe below.

The instructions are 16 bits. These are kept in a ROM containing up to 256 instructions. The CPU has 2 possible instructions:

- ***Output***
- ***Jump***

These operations are divided within the instructions as follows:

BIT #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Output	OUTPUT PATTERN								DELAY≠0							
Jump	JUMP ADDRESS								DELAY=0							

So, we have the following logic:

- DELAY = 0 => Do Jump operation BIT[15:8] is the address we jump to.
- DELAY ≠0 => Do Output operation
  BIT[15:8] is the pattern shown in the LEDs.

  BIT[7:0] is the amount of time this particular pattern should be shown in the LEDs.

Then it moves to the next address

The CPU simply starts from address 0 and executes the following instructions.

Example

Let's say we wish to make the rotating dot example with this CPU and consider the original version where the dot rotated from left to right.

Let's divide the rotating dot to LED outputs, so we should see:

BIT #	7	6	5	4	3	2	1	0
STAGE0	1	0	0	0	0	0	0	0
STAGE1	0	1	0	0	0	0	0	0
STAGE2	0	0	1	0	0	0	0	0
STAGE3	0	0	0	1	0	0	0	0
STAGE4	0	0	0	0	1	0	0	0
STAGE5	0	0	0	0	0	1	0	0
STAGE6	0	0	0	0	0	0	1	0
STAGE7	0	0	0	0	0	0	0	1

So the corresponding LedCPU's ROM, i.e. its memory, should be:

BIT # / Address	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	7	6	5	4	3	2	1	0
0	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	1	0	0	0	0	0	0	0
1	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	1	0	0	0	0	0	0	0
2	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	1	0	0	0	0	0	0	0
3	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	1	0	0	0	0	0	0	0
4	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	1	0	0	0	0	0	0	0
5	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	1	0	0	0	0	0	0	0
6	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	1	0	0	0	0	0	0	0
7	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	1	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

So, between addresses 0 and 7, we give new LED patterns that are in the instruction's BIT[15:8] and ask the delay to be 8'b10000000 in BIT[7:0]. At the last address (address 8), the CPU jumps back to the instruction at the address 0.

Question

What does this 8'b10000000 delay value in the example mean? Yes, we all know 8'b10000000 is 8'd128 but then again how long in terms of seconds does this delay mean? Do not just give the number, explain how you obtained / calculated that value. (Handwrite the solution for the question and take a picture)

(Note: For answering this question, check the Verilog code and remember that we use a master clock of 50 MHz. Also remember to use the commented frequency value for this question.)

ASSIGNMENTS

Part 1:

1. Download lab8 files.
2. Create new project in Xilinx ISE Design Suite and add copy of source project files.
3. Open LedCPUCore.v file and modify as required.
4. After the synthesis take a screenshot of the whole screen.
5. Open Virtual FPGA program. (VFPGA.bat) and add all .v files and top_wrapper.v files.
6. When the design is complete, you should see a rotating dot running from left to right on the FPGA.

Part 2:

1. Open ROM.v file
2. Modify ROM code for right to left rotation with ¼ seconds.
3. Run the VFPGA program.

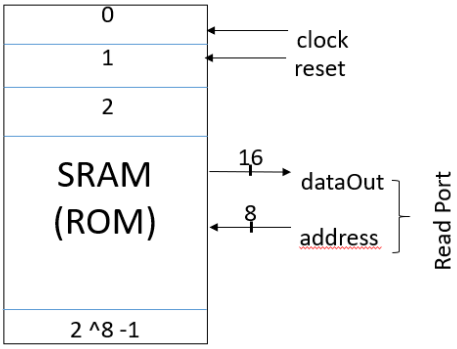

Figure. ROM I/O

Part 3:

1. Open all LedCPUwithRAM files.
2. Create new project in Xilinx ISE Design Suite and add copy of source project files.
3. You can take the LedCPUcore.v file from part1.
4. Modify the progLogic.v file as required.
7. After the synthesis take a screenshot of the whole screen.
5. Run the Virtual FPGA program again.

Submission

Submit **progLogic.v**, **LedCPUcore.v** and **ROM.v** file that you modified in part2. files and all **synthesis screenshots** (for part1 and part3) to LMS under assignment LAB8.

Handwrite the solution for the question, take a picture of it and add to the submission files.