**You have to submit this report via Moodle.**

|  |  |  |
| --- | --- | --- |
| Digital Design and Computer Architecture: Lab Report | | |
| Lab 8: Full System Integration (Session I) | | |
| Date | 04.06.2021 | Grade |
| Names | Berner, Zheng |  |
|  |  | Lab session / lab room |
|  |  | Friday 08:15  Zoom breakout room 60 |

**Use a zip file or tarball that contains the report and any other required material. Only one member from each group should submit the report. All members of the group will get the same grade.**

**The name of the submitted file should be *Lab8\_LastName1\_LastName2.zip* (or *.tar*), where *LastName1* and *LastName2* are the last names of the members of the group.**

**Note 1: Please include all the required material. No links/shortcuts are accepted.**

**Note 2: The deadline for the report is a hard deadline and it will not be extended.**

**Exercise 1**

Which MIPS instructions do you think would produce wrong outputs if the ControlUnit signal *RegWrite* is *'stuck at 0'*, i.e., *RegWrite* always has value 0? In other words, which MIPS instructions depend on the control signal *RegWrite*?

**All instructions that write to a register would produce wrong outputs in this case.**

**Meaning all instructions except SW, J, JR and branch instructions.**

**Exercise 2**

Explain why a 6-bit address is enough for the instruction and data memory. *(Hint: size of the memory.)*

**The modules InstructionMemory and DataMemory define arrays DataArr with 64 entries [63:0].**

**The address needs to specify the entry (for example DataArr[A] <= WD;).**

**Therefore, the biggest specified entry needs to be 63.**

**A 6-bit 111111 is 63. Which means 6-bit is enough to address all entries.**

**Exercise 3**

As you might have noticed, there are three different counters used in this lab. One is present in the *snake\_patterns.asm* file, the second is in the clock\_div module and the third is the DispCount signal for the 7-segment display. Explain the functions of each of these three counters/dividers in a sentence or two each.

**The counter in the *snake\_patterns.asm* file is intended to get some waiting time before the snake moves to the next LED-Part.**

**After a loop (when the counter is done) $t4 is increased by 4 causing the snake to move because there are 12 LED-Parts used for the snake pattern. 12\*4 = 48, which is the initialized length of the display pattern stored in $t5.**

**Without the loop, the snake would move on immediately after the previous move and would just be racing.**

**The counter in the clock\_div module has 2 bits and increases from 0 to 3.**

**The values get assigned with non-blocking assignments, so the first value is also assigned even though clk\_count was increased.**

**Those 4 values are passed through a reduction AND-operator that only outputs a 1 if all the individual bits are 1. Therefore, it only outputs a 1 after every 4th clock cycle, slowing down the clock.**

**The two most significant bits of the DispCount counter determine which of the four LEDs is active at a given moment.**

**The most significant bits are chosen because then you only have LEDs light up in 4 out of 65536 possible values of DispCount, which gives you the desired light up time described in a top module comment (“Use the MSB of the Disp count so that each digit lights up for 1.6ms == 65536/4 \* 100ns”).**

**Feedback**

If you have any comments about the exercise please add them here: mistakes in the text, difficulty level of the exercise, or anything that will help us improve it for the next time.