Report for the implementation of an instruction demultiplexer and a CPU

Specification verification for the instruction demultiplexer

To prove the specification of the demultiplexer, the five commands given in the assignment description were executed and the outputs tested. As can be seen in figure 1, all the tests are successfully completed. For easy readability every action was stored as a string. With this feature, it is very easy to see when which test happens and therefore what the resulting output should be.

For the test bench all the expected outputs got packed in a constant. After the setup of the demuxer inputs the outputs get checked with these constants. All the relevant data of the demuxer is also packed to bit arrays for easy reading and comparing the values given in the specification.

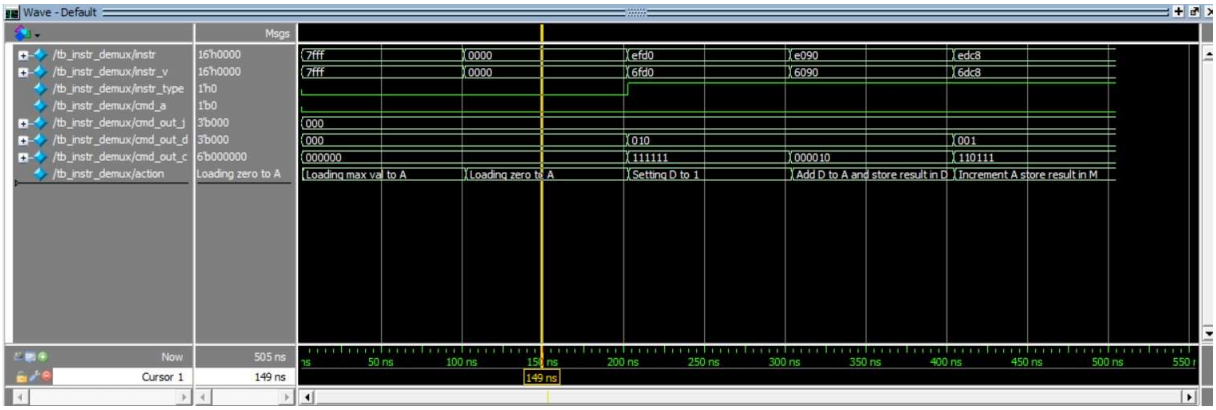

Figure 1: Instruction demultiplexer testbench waveform

Specification verification for the CPU

To verify this specification all the commands used in the demultiplexer testbench were also used here. A sixth test was introduced because of the following: when using the order as said in the specification, the register A is first set to zero, D is set to one and then they are both added and stored in M. Now if there is any failure, it is possible that A and D did not get summed up and just D is outputted to M because A was zero in the first place. Therefore, before they get summed up, A is set to d42 or 0x2a. So, the signal can be tested for the functioning summation of the registers. Same is true for the last test “INC_A_STORE_IN_M”. Here also just D could be putted out.

As can be seen in figure 2, every test follows the same timing structure. On the first negative edge of the 25MHz clock the action string is set and can be seen in the wave form diagram. On the next negative edge, the input is set for the CPU. Afterwards the expected output is written on the next positive edge and on the next negative edge the signal is probed and compared with the expected output.

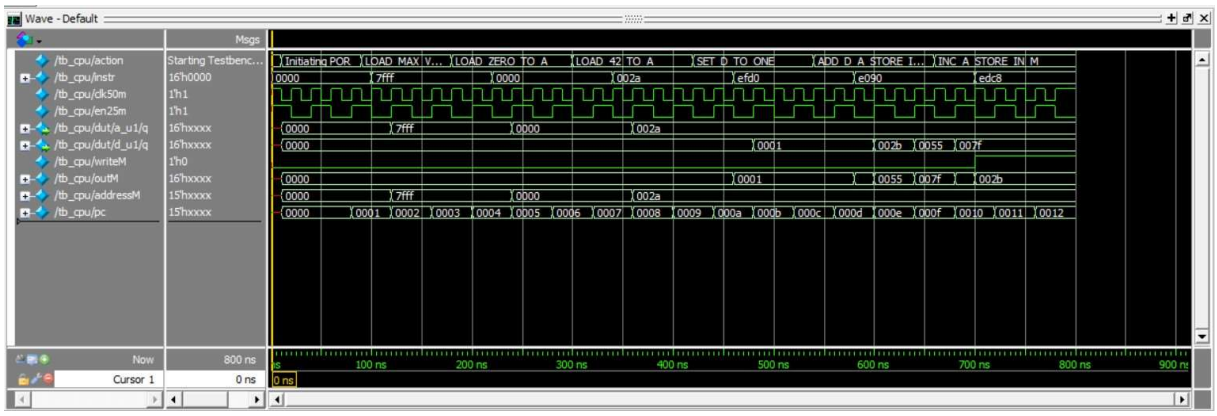

Figure 2: CPU testbench waveform

Used implementation method

For the instruction demultiplexer combinatorial logic was used as the demuxer does not have a dependency on the past.

For the CPU the only functional code was to implement the three multiplexers which choose the input of the A register and the ALU. Everything else was to instantiate the modules and interconnect them with wires. For the wires mostly the same name as the module's outputs were used.

Even if not required, the testbench for both modules CPU, and demuxer, is automatically testing and comparing the results printing an error or success message in the end of the testbench.