Oblig 3 – Signals, Variables and Process Sensivity

Task a:

Upon simulating the entity *delay*, we see that the signal of *out_data* is not updated until after 450 ns have passed. The main reason for this so called ‘delay’ in the change of the signal is caused by the rules according to which signals are updated in simulation in VHDL. One of those rules clearly states that signals are updated once in a process invocation, and only at the exit of a process. If we inspect the sensivity list of the *delay* entity and its testbench, we see that a new process is triggered by the changes to either the clock signal *mclk* or the reset signal *rst_n.* For the first 100 ns *rst_n* holds the value ‘0’, before chaning to ‘1’ , meaning that upon the exit of the second process, signals *data1, data3* and *data5,* and variables *data2* and *data4* are no longer assigned value ‘00000000’. The process then runs for another 100 ns before *indata* changes to ‘11111111’, but the clock signal is on the falling egde at this particular moment, meaning that the simulation has to run for another 50 ns before any changes to *data1* signal can be applied. Upon reaching the 250 ns marker, a simultanous assignment of values takes place in the simulation, where *signal data1* is assigned the value of *indata*, and variable *data2* is assigned the value of *data1*. Then at 350 ns, at the exit of the seventh process, signal *data3* receives the value stored in variable *data2*, and variable *data4* is given the value of *data3.* Finally, after 450 ns, at the exit of the ninth process, the value of variable *data4* is assigned to signal *data5* at the rising edge of the clock, and the value is then given to *out_data.*

Task b:

After applying the changes to both the enitity *delay* and its testbench *tb_delay*, the *out_data* signal is undefind for the first 50 ns, because all variables have been replaced with signals. This means that any update to any singal in the simulation will happen at the exit of a process, and not immediately as is the case for variables. Hence, since the first process exits at 50 ns, the *out_data* signal will remain undefined for the duration od the first process. The next change in the *out_data* happens at 750 ns, because the first change in *in_data* does not happen until after 300 ns have passed. This change is, however, applied at the falling edge of the clock, hence the new value of *in_data* is assigned to *data1* signal at the exit of the seventh process, when the clk is at the rising egde again. Since the process only contains signals, the value of *in_data* assigned to *data1* will be assigned to each consecutive *data* signal after 100 ns. This means that it will take 400 ns before the value of *in_data* will show in the *out_data*. This is also true for the change in *in_data* at 450 ns, when *in_data* receives the value “00001111”, which is visible at the output at 850 ns (the last change in *out_data*).