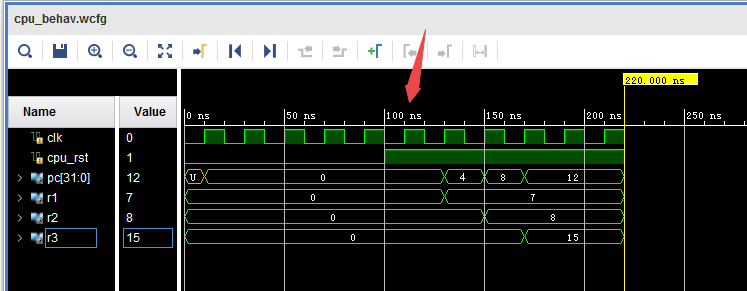
2.3.3 Functional Simulation

(1) Sample code 1

The operations of this code are described in 2.3.2. Now we run a functional simulation to check it.

The sequence of the signals being showed is:

clk, cpu\_rst, pc, r1, r2, r3



We can see that after the cpu\_rst becomes high, the instructions in the inst\_mem are executed one by one. Then 7 is written to r1, 8 is written to r2, and their sum 15 is written to r3.

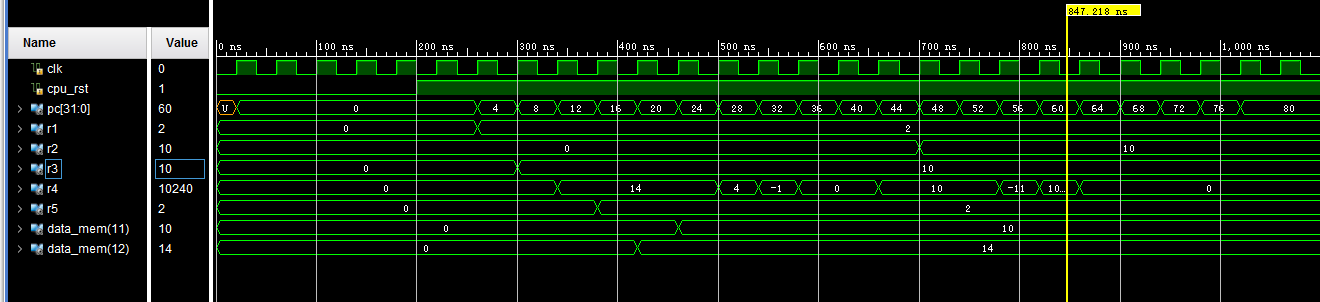
Notice that after cpu\_rst is set to 1, the first instruction with address “0” will be fetched and begin to be executed at the first coming clock rising edge (pointed by the red arrow). Its result will be valid at the same time when the next instruction with address “4” is fetched (e.g. the next clock rising edge), because we perform a synchronous writing operation.

(2) Sample code 2

The operations of this code are described in 2.3.2, too. Now we run a functional simulation to check it.

The sequence of the signals being showed is:

clk, cpu\_rst, pc, r1, r2, r3, r4, r5, data\_mem(11), data\_mem(12)



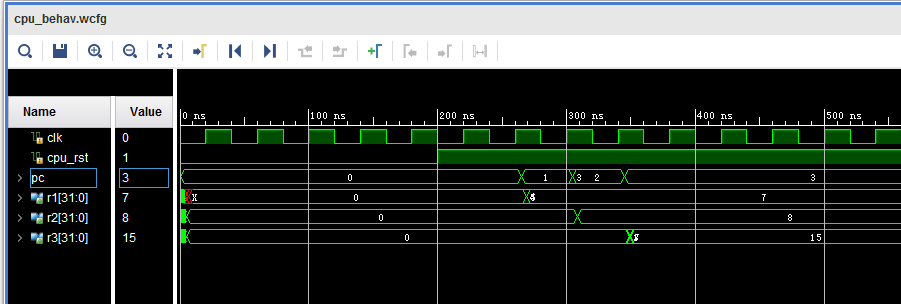
Still, after the cpu\_rst goes high, the instructions are executed and the result for each instruction is valid at the next cycle. Compared to the description in 2.3.2, we can find the simulation results are correct.

2.3.4 Timing Simulation

(1) Sample code 1

This time we should check the timing simulation of this code. The sequence of the signals is the same as 2.3.3.

Notice that here we doesn’t show the two least significant bits of pc (program counter), so the values it gives represent the sequence number of the instructions being executed instead of the addresses (0, 1, 2, 3… instead of 0, 4, 8, 12…)



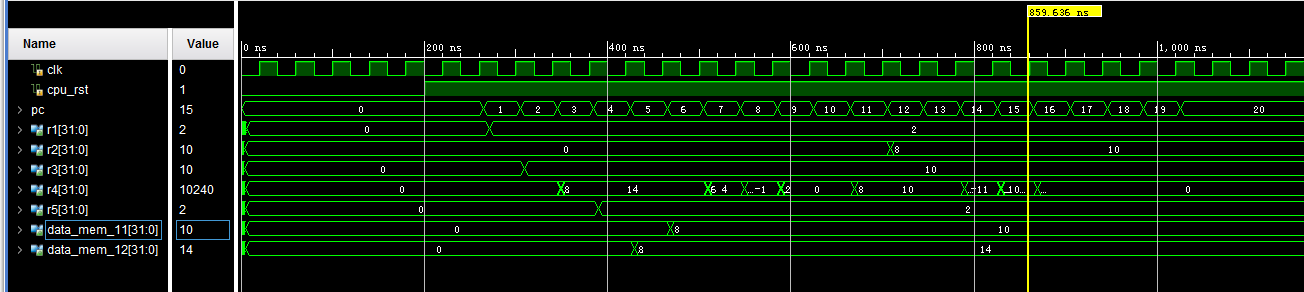
We get the same results as the functional simulation.

Notice that there is a delay between the clock rising edge and the results being valid.

Besides, there are also glitches at the time when the signals change, because the values of different bits in the signal may not arrive at the same time.

(2) Sample code 2

The configurations are the same with 2.3.3 (2), and still we let pc represent the sequence number of the instructions.



Similarly to 2.3.4 (1), this timing simulation is in correspondence with its functional simulation. And we can see delays and glitches in the wave form.