PD 0

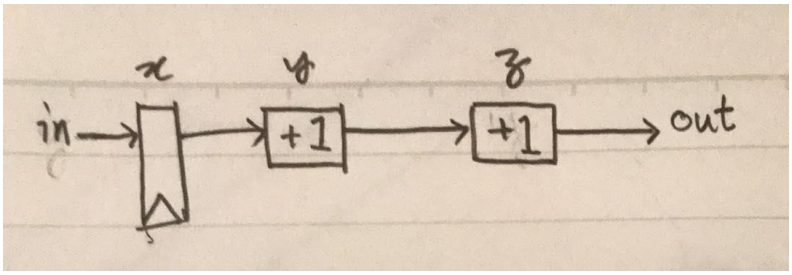
Bu Fan Wang

20656402

Exercise 3.1) Please explain why the order in which the three non-blocking statements specified within the always is irrelevant? Use no more than 5 sentences.

By definition, a non-blocking assignment defers its assignments until all right-hand sides have been evaluated at the end of the simulation step. This means that the values of the right-hand sides in a non-blocking assignment, at the end of the simulation step, will be equivalent to the left-hand side values at the beginning of the simulation step. Therefore, the order of three non-blocking statements within an always does not matter, as changing the order does not affect the values of the left-hand sides at the beginning of the simulation step.

Exercise 3.2) Draw a hardware diagram (using similar blocks as Figure 3) to describe the resultant hardware for the blocking version of the pipeline specification as shown in Listing 15. Using no more than 5 sentences, please explain what the resultant hardware does.



The implementation in Listing 15 is not a pipeline implementation. Because blocking assignments are used, line 13 is immediately dependent on line 12, and line 12 is immediately dependent on line 11. As a result, only one register is used, and the expected output is shown after only one simulation step.

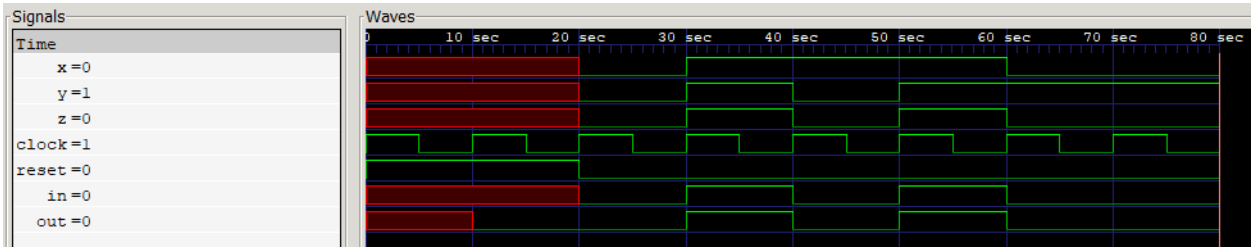
Exercise 3.3) Write the Verilog specification of the example in Listing 16 to implement the registered AND gate with an asynchronous reset. Using no more than 5 sentences, describe the difference between asynchronous and synchronous resets.

Change reg_and to:

```
module reg_and
      (
       input wire clock,
       input wire reset,
       input wire in,
       output     out
      );
      reg out;
      always @(posedge clock or negedge reset)  begin
            out <= in & (~reset);
      end
endmodule
```

An asynchronous reset resets the state of the hardware as soon as the reset signal is given. It would not be coordinated with the clock of the system in a simulation. Meanwhile, a synchronous reset has its reset signal coordinated with the clock of the system, and the resetting of the state would only happen at the end of the simulation step.

Exercise 3.4) Write the Verilog specification using the example in Listing 16 to register both the inputs and outputs to the AND gate. Make sure to also show the output of the simulation. Explain whether the simulation output is the expected behaviour or not.




```
VCD info: dumpfile reg-and.vcd opened for output.
time=                   10, reset=1, x=x, y=x, z=x, out=x
Reset complete
time=                   20, reset=1, x=0, y=0, z=0, out=0
set 1 1
time=                   30, reset=0, x=1, y=1, z=1, out=0
time=                   40, reset=0, x=1, y=0, z=0, out=1
time=                   50, reset=0, x=1, y=1, z=1, out=0
time=                   60, reset=0, x=0, y=1, z=0, out=1
time=                   70, reset=0, x=0, y=1, z=0, out=0
time=                   80, reset=0, x=0, y=1, z=0, out=0
```

In the simulation, the AND gate and reset signals work as expected. However, there's a difference between the logs of the simulation vs. the visual simulation from GTKWave. In the logs, the output seems to be slightly delayed. I take this to be because the output assignment is registered, and thus only becomes equal to z at the end of the simulation step. Being that GTKWave is displaying the data over

the span of 80 seconds, the size of the simulation step is negligible, so the output is shown to have the same result as z, as the clock's positive edge coincides with the change in x or y signals.

To verify this hypothesis, I modified the simulation log:

```
VCD info: dumpfile reg-and.vcd opened for output.
time=                   10, reset=1, x=x, y=x, z=x, out=x
time=                   11, out2=0
Reset complete
time=                   20, reset=1, x=0, y=0, z=0, out=0
time=                   21, out2=0
set 1 1
time=                   30, reset=0, x=1, y=1, z=1, out=0
time=                   31, out2=1
time=                   40, reset=0, x=1, y=0, z=0, out=1
time=                   41, out2=0
time=                   50, reset=0, x=1, y=1, z=1, out=0
time=                   51, out2=1
time=                   60, reset=0, x=0, y=1, z=0, out=1
time=                   61, out2=0
time=                   70, reset=0, x=0, y=1, z=0, out=0
time=                   71, out2=0
time=                   80, reset=0, x=0, y=1, z=0, out=0
```