Week 3

**Ex1:**

**Initial and always block**

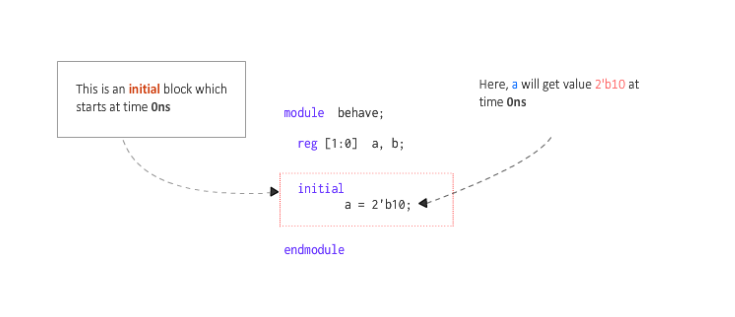
Those are procedural block, all the statements inside initial anad always blocks are executed sequentially.

Those can have more than 1 initial or always block in our design. Those should be used when more than one variable are assigned in initial or always blocks.

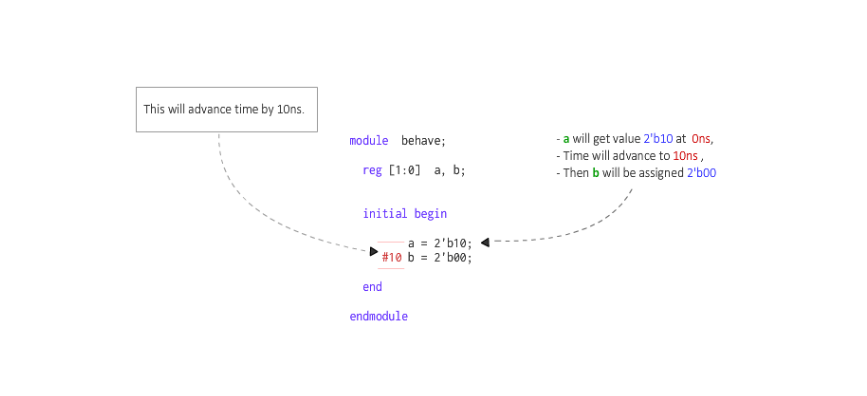
**Initial block**

An initial block is not synthesizable and hence cannot be converted into a hardware schematic with digital elements. Hence initial blocks do not serve much purpose than being used in simulations as these are primarily used to initialize variables and drive design ports with specific values.

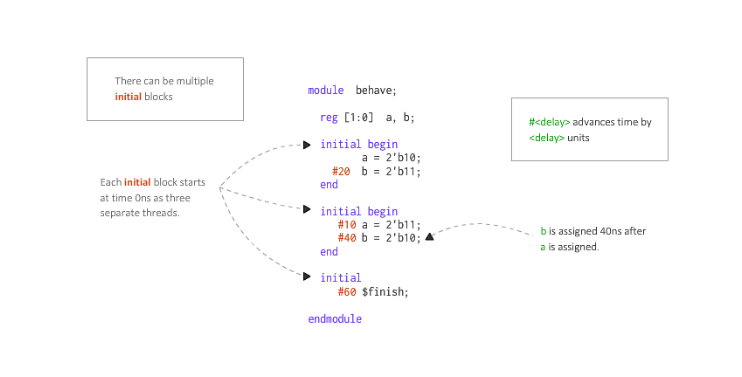
An initial block started at the beginning of a simulation at time 0 unit. This block will be executed only once during the entire simulation. Execution of an initial block finishes once all the statements within the block are executed.



Supposing that we add a delay to each gate:

  
This means that after 10 time units from execution from prev statement, a is assigned 1st with the given value and then after 10 time units, b is assigned to 0.

There are no limits to the number of initial blocks that can be defined in the module. In this example we have 3 initial blocks at the same time.



Initial is not synthesizable for ASIC( synthesizable in some FPGA)

**Always blocks**

Always block is one of the procedural blocks in Verilog. Statements inside an always block are executed sequentially. The always block is executed at some particular event as this event is defined by a sensitivity list

Sensitivity list is the expression that defines when the always block should be executed and is specified after the @ operator within the parentheses(). This list may contain either 1 or a group of signals whose value change will execute the always block. Particularly, all statements inside the always block get executed whenever the value of signals a or b change.

This block can be used to realize combinational or sequential elements. A sequential element like flip flop becomes active when it’s provided with a clock and reset. Similarly, a combinational block becomes active when one of its input values change. These hardware blocks are all working concurrently independent of each other. The connection between each is what determines the flow of data. To model this behavior, an always block is made as a continuous process that gets triggered and performs some action when a signal within the sensitivity list becomes active.

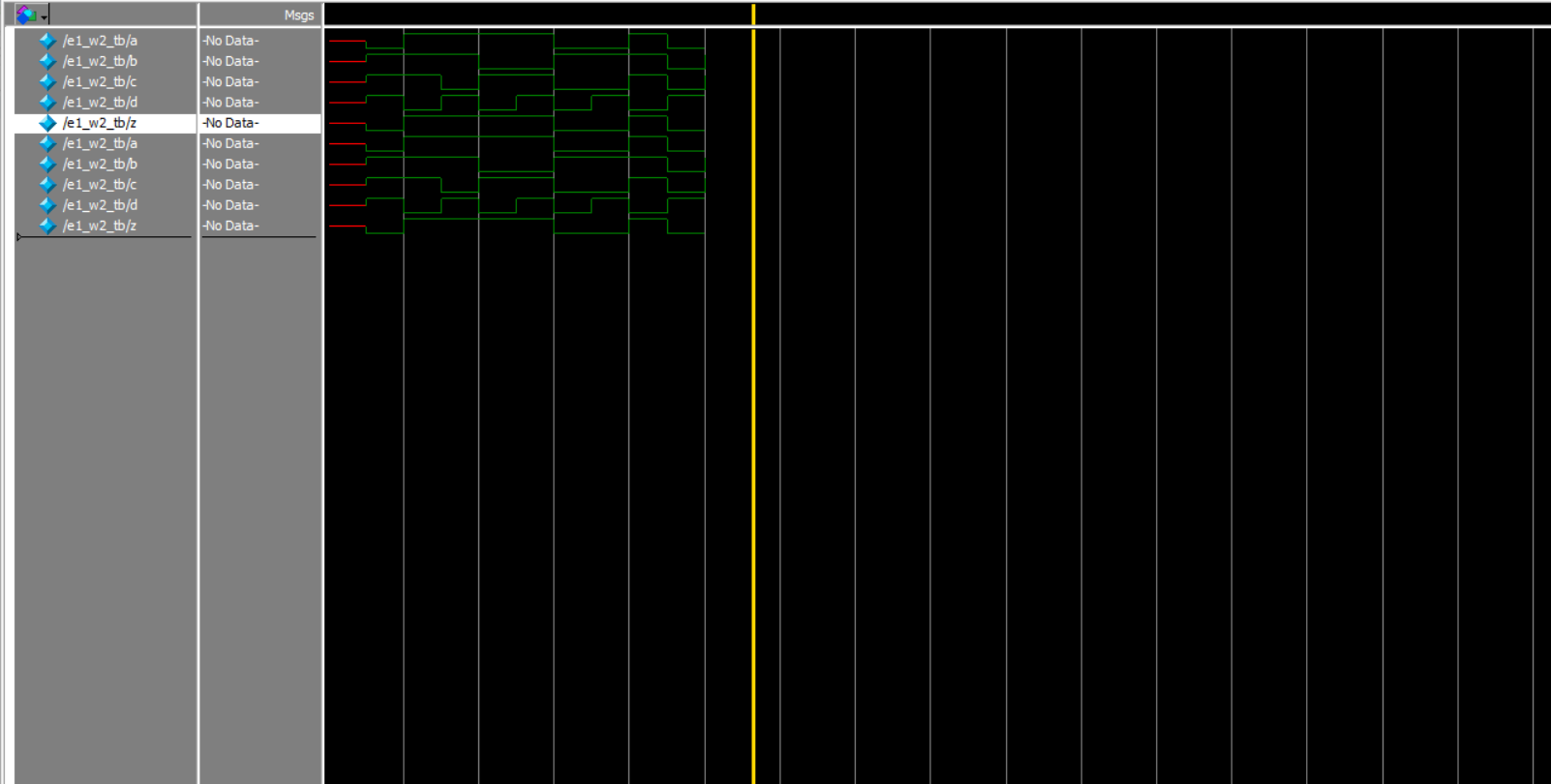
**Ex2**

**K-map ex**

Testbench

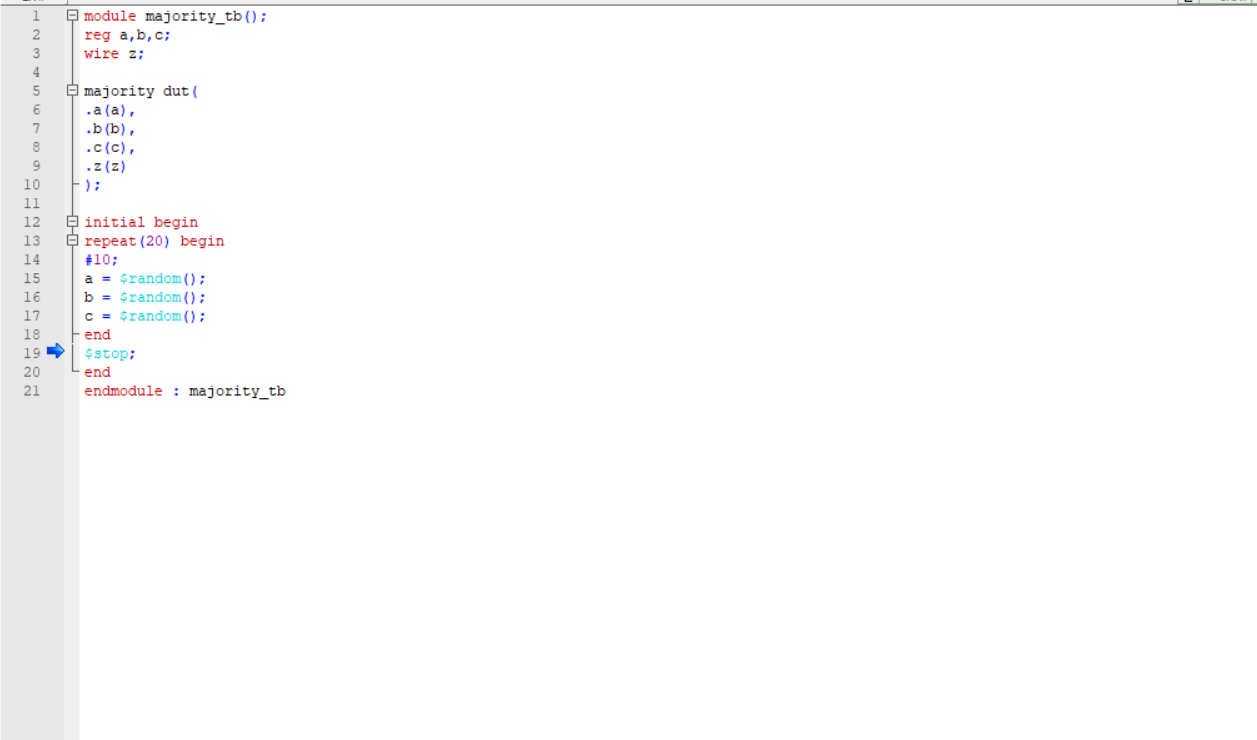


Wave

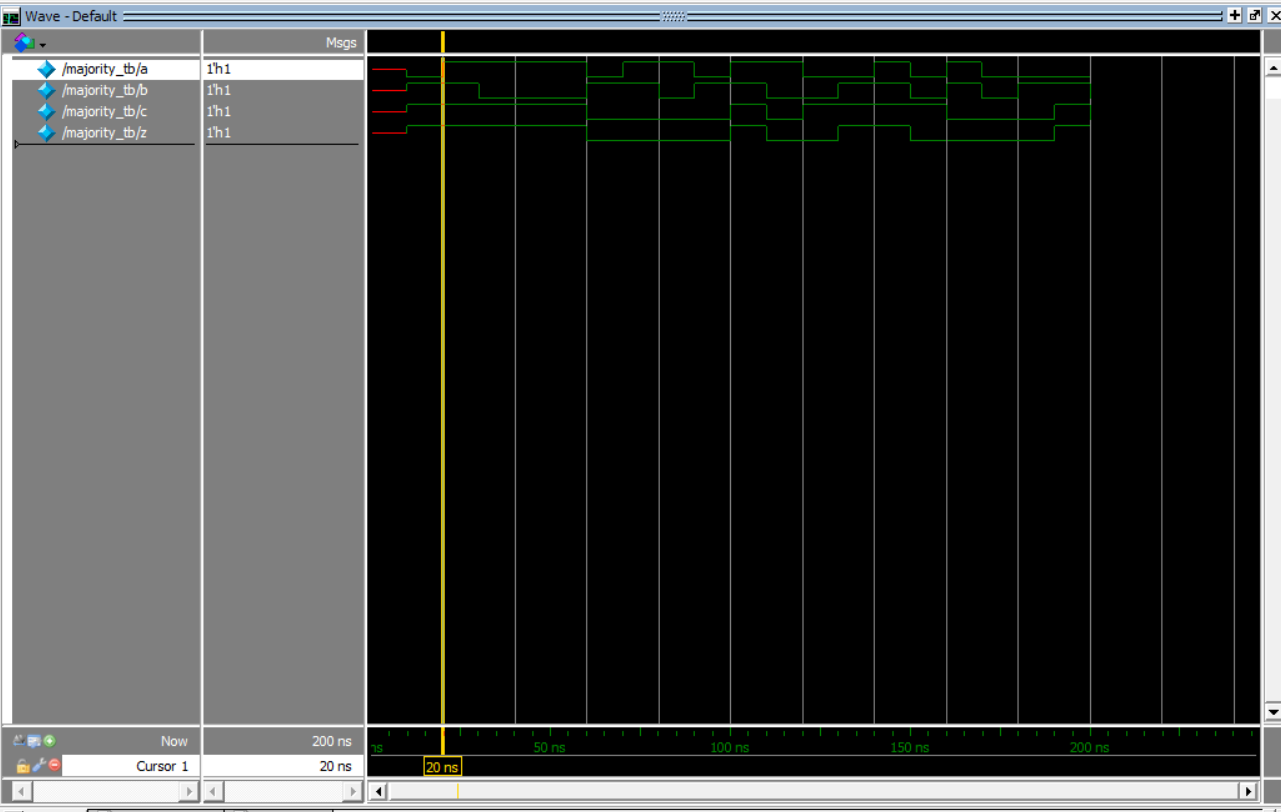


**Majority**

Testbench

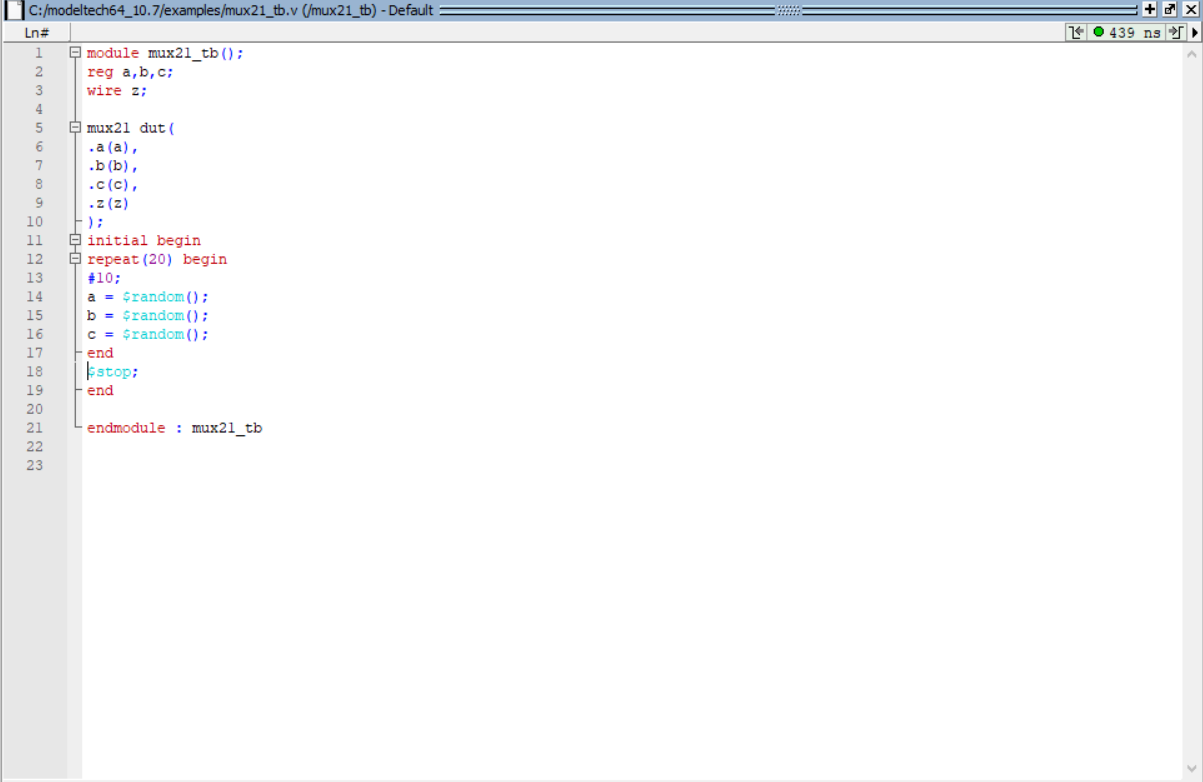


Wave

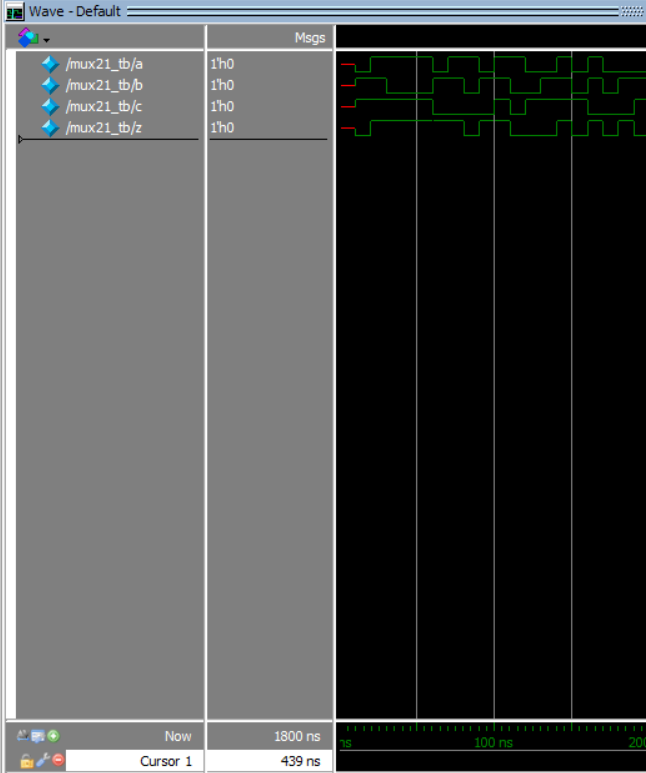


**MUX**

Testbench

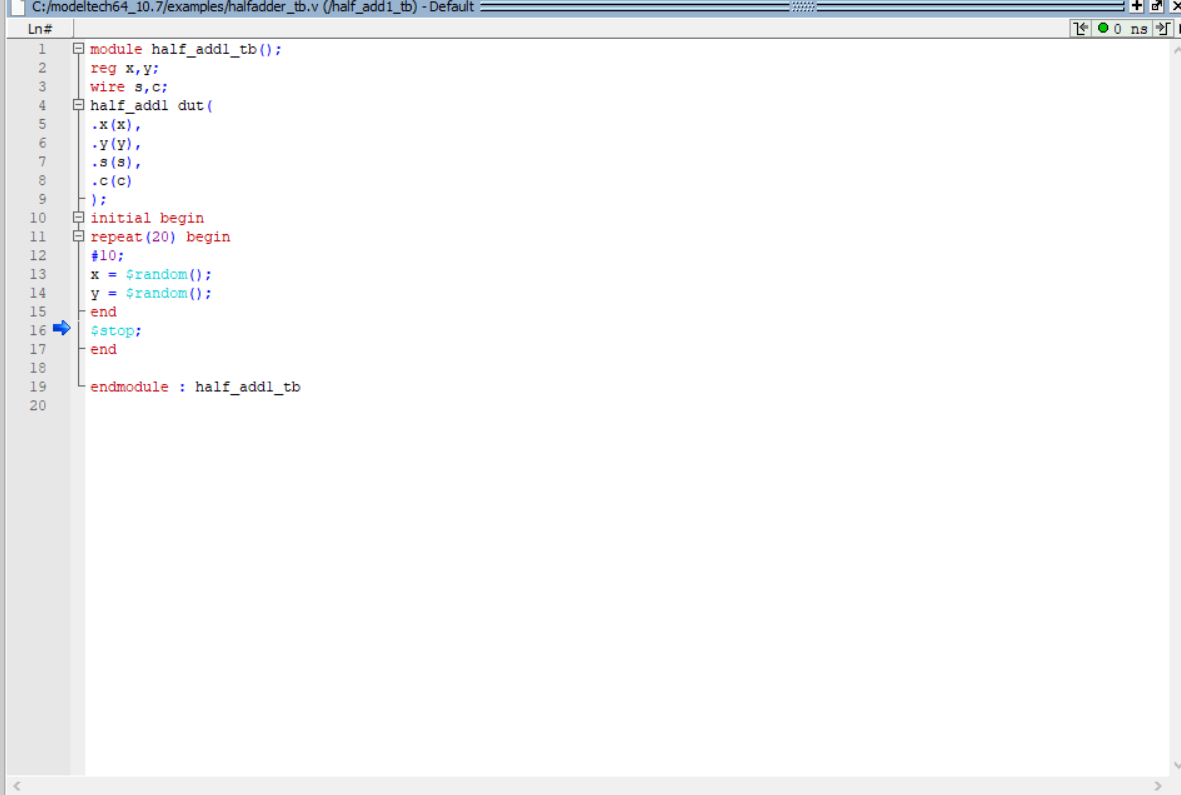


Wave

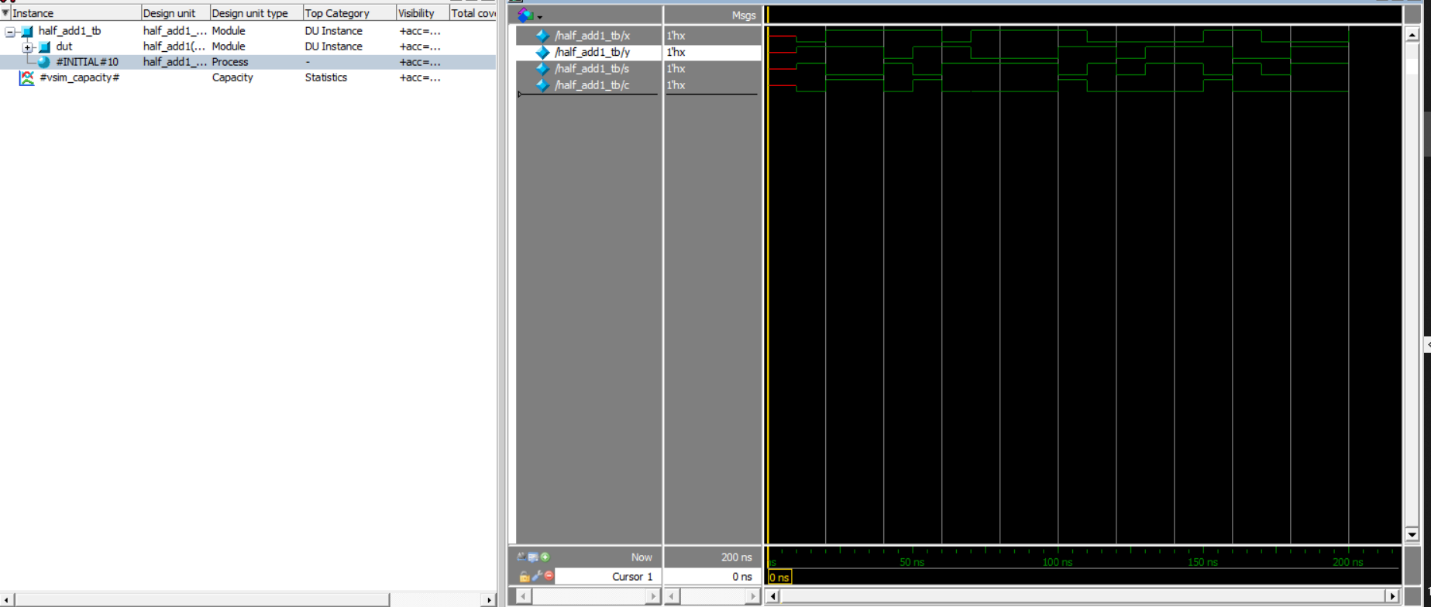


**Half Adder**

Testbench

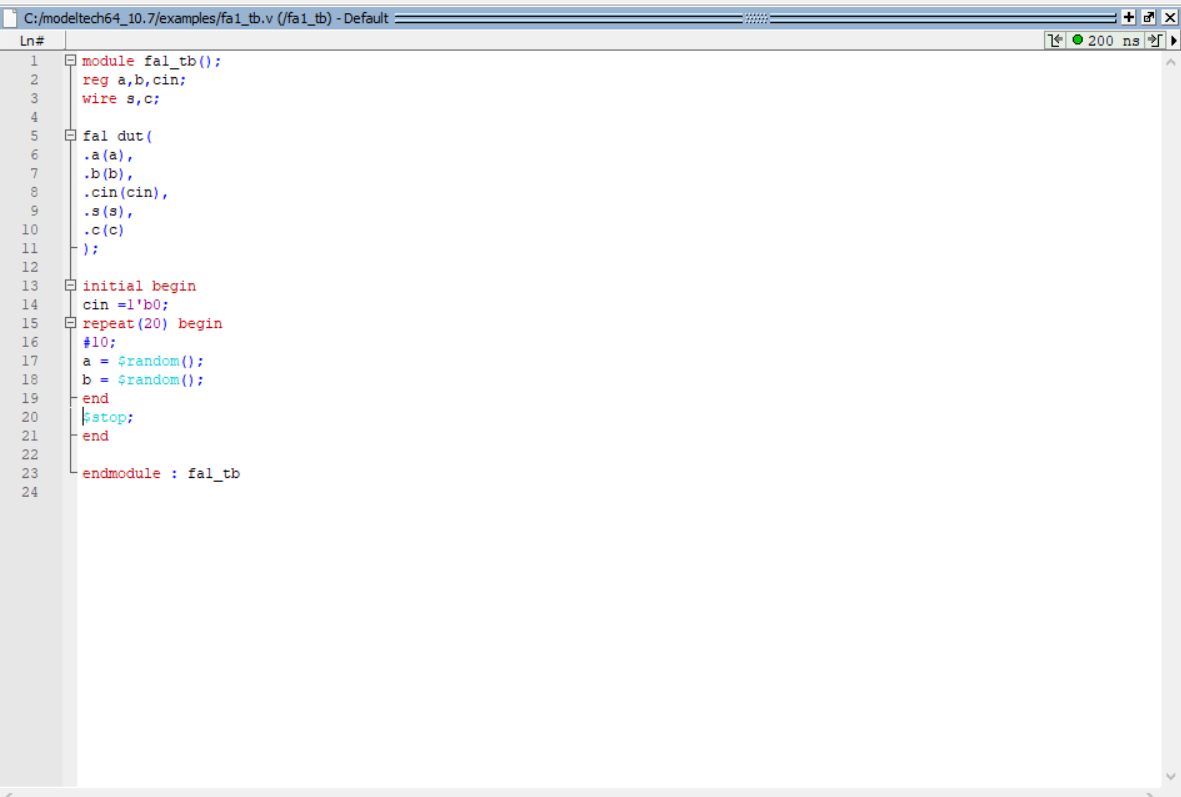


Wave

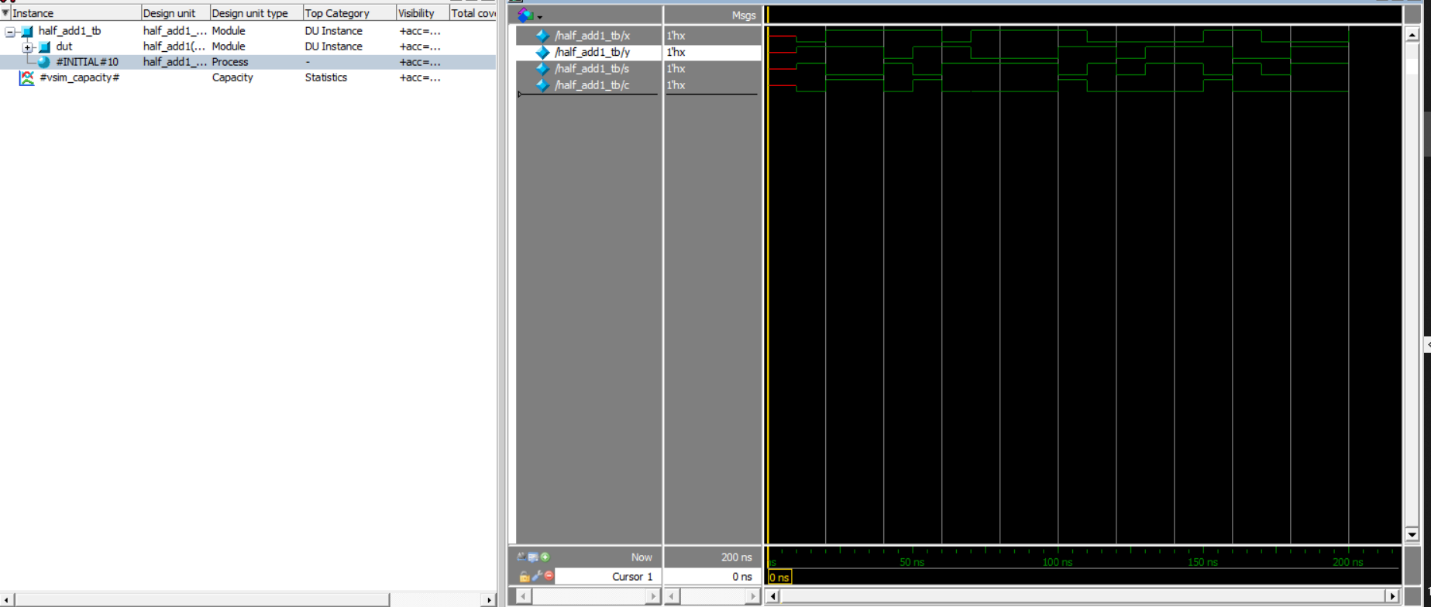


**Full Adder**

Testbench

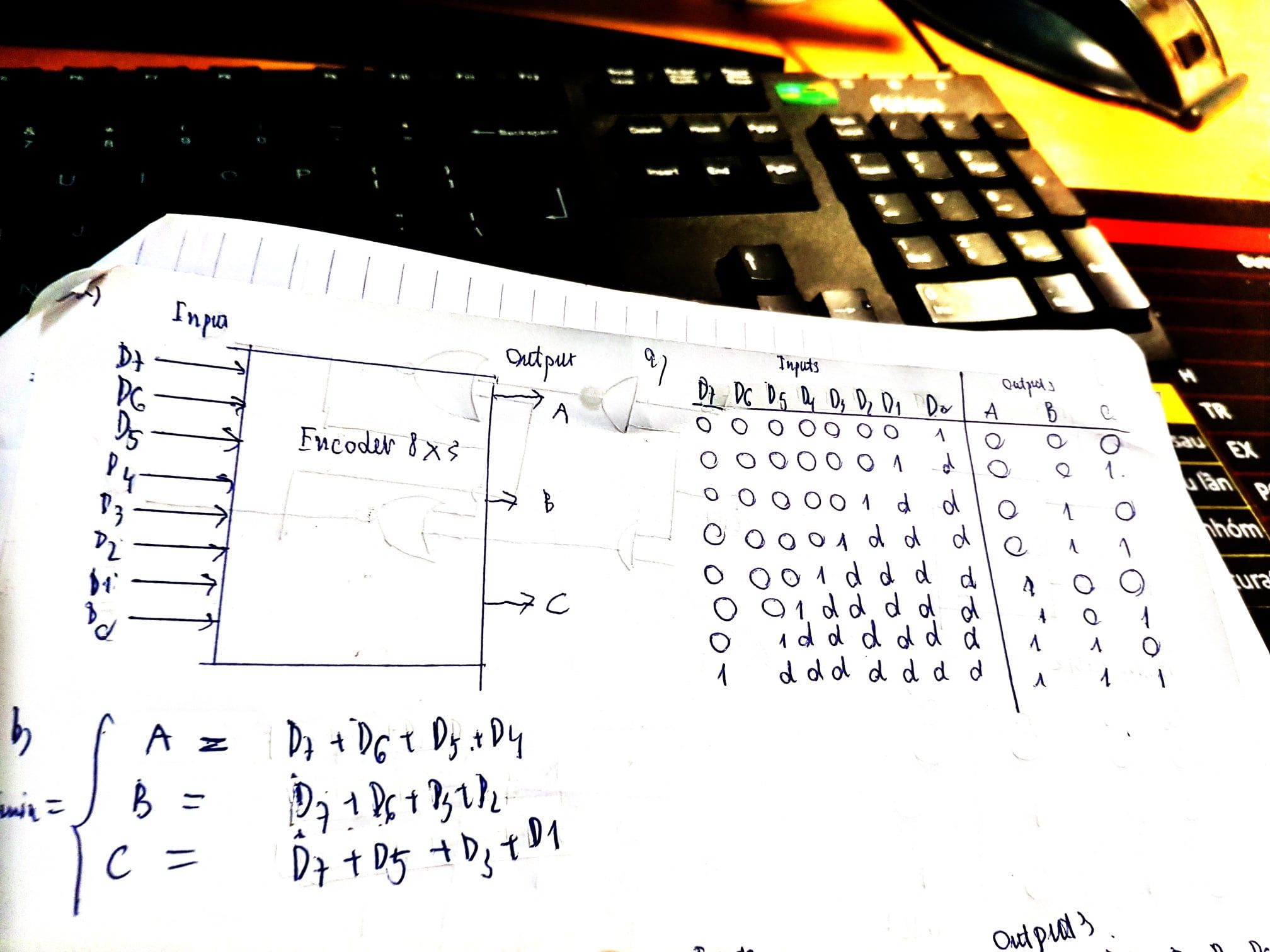


Wave

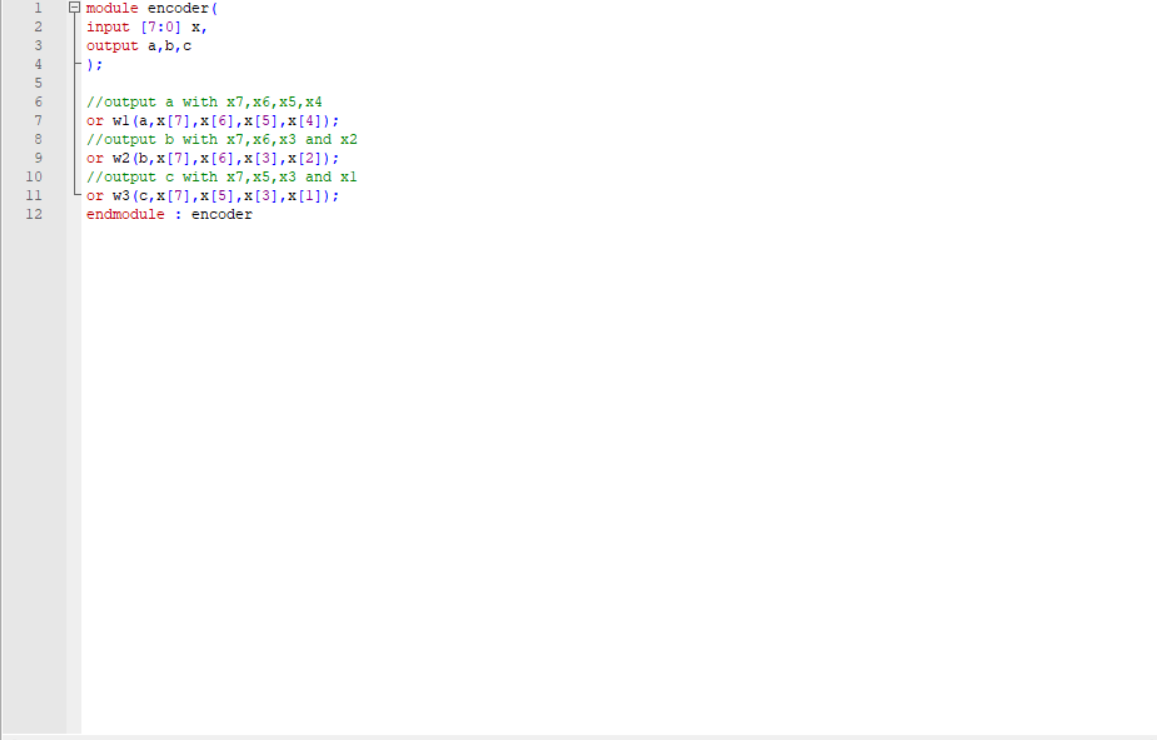


**Ex3**

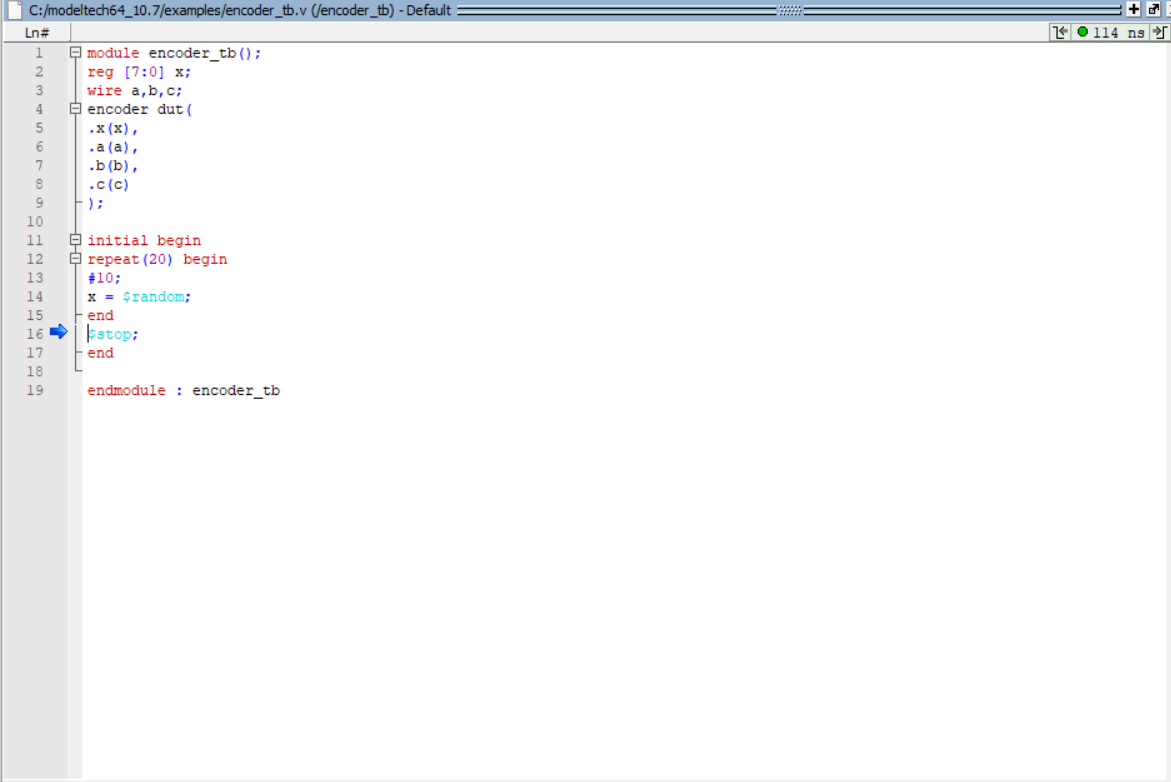
**Encoder 8x3**



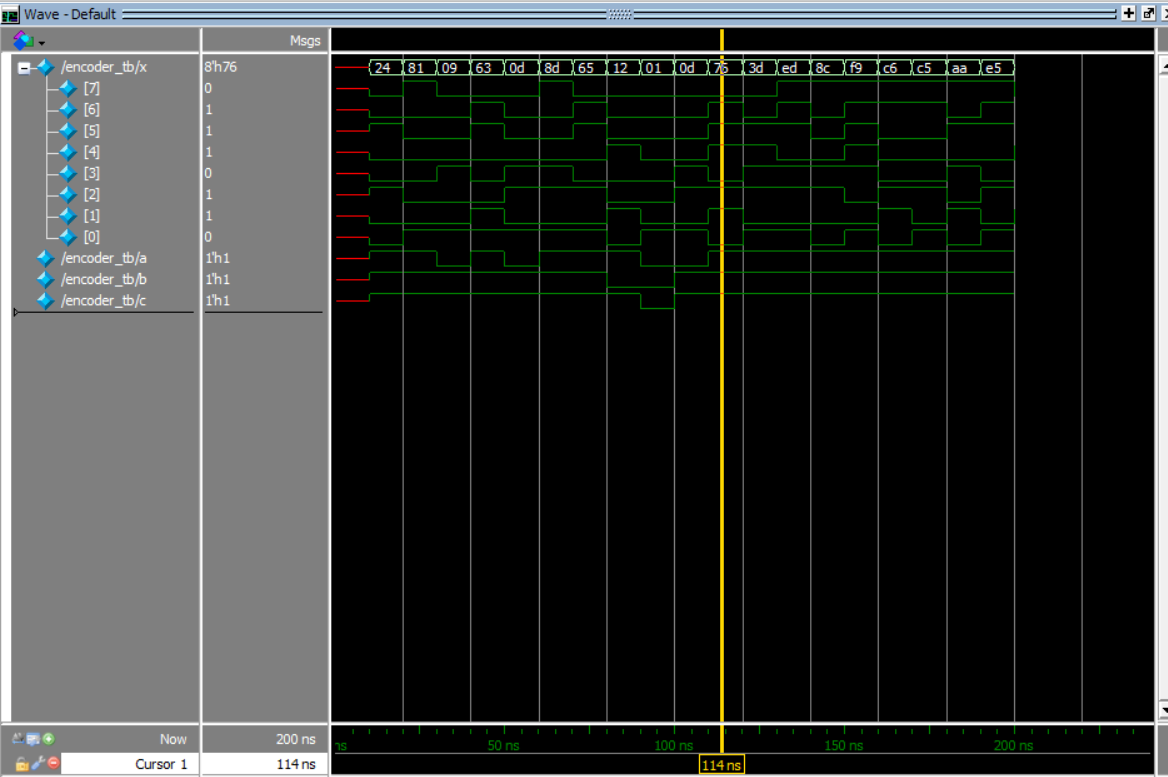
Verilog



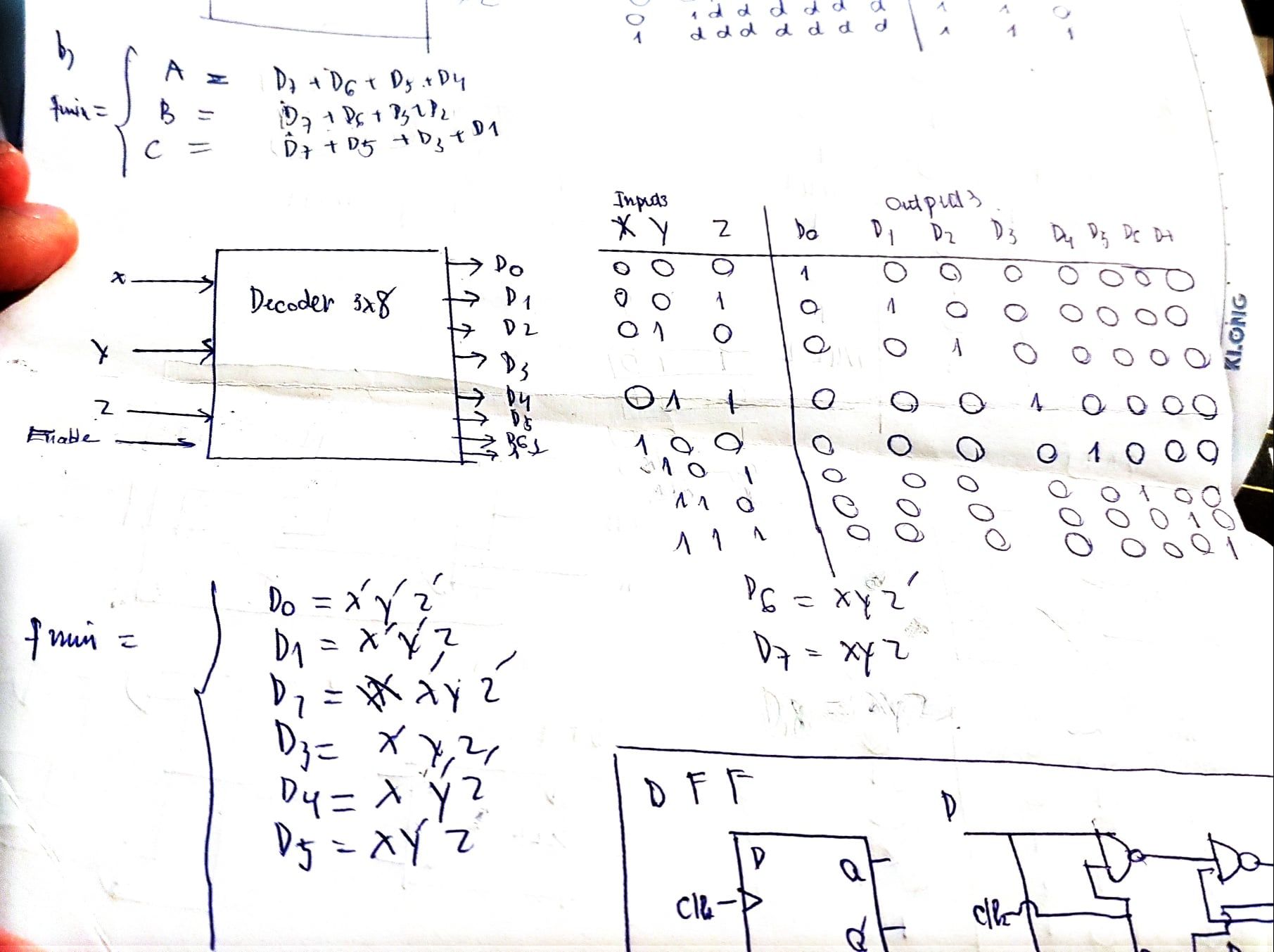
Testbench



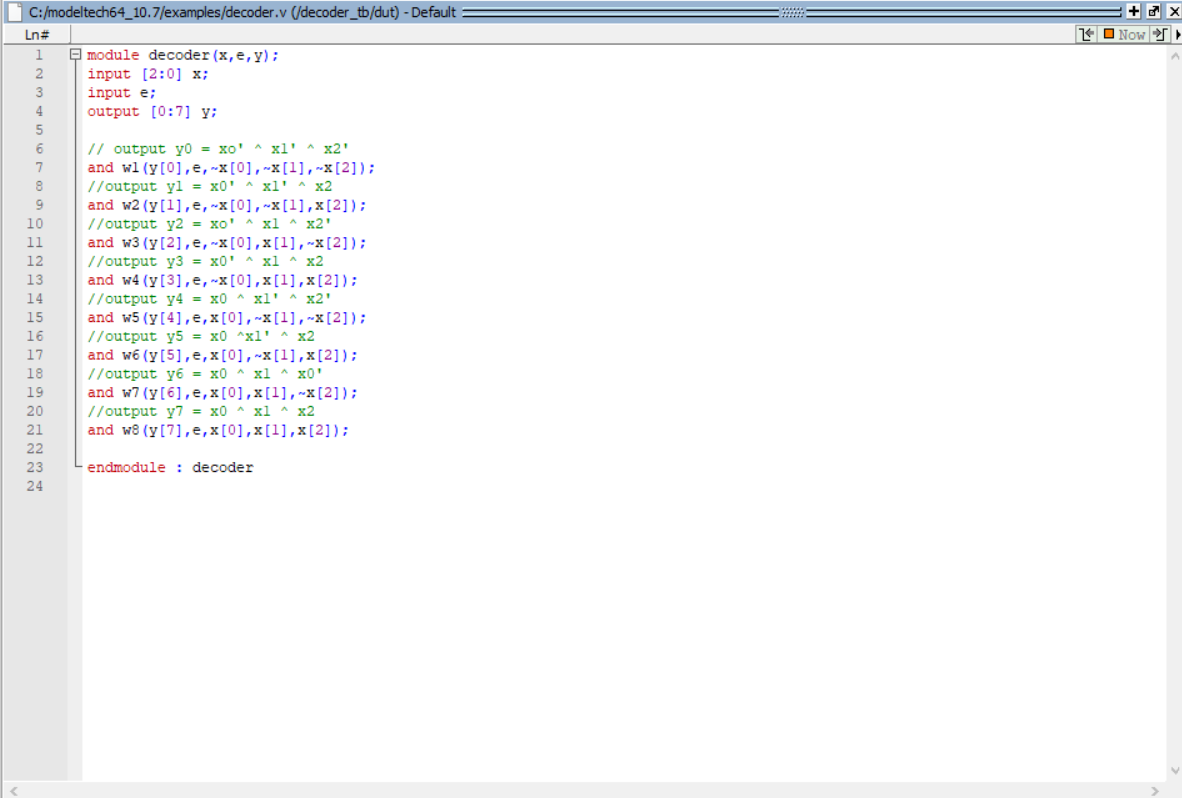
Wave



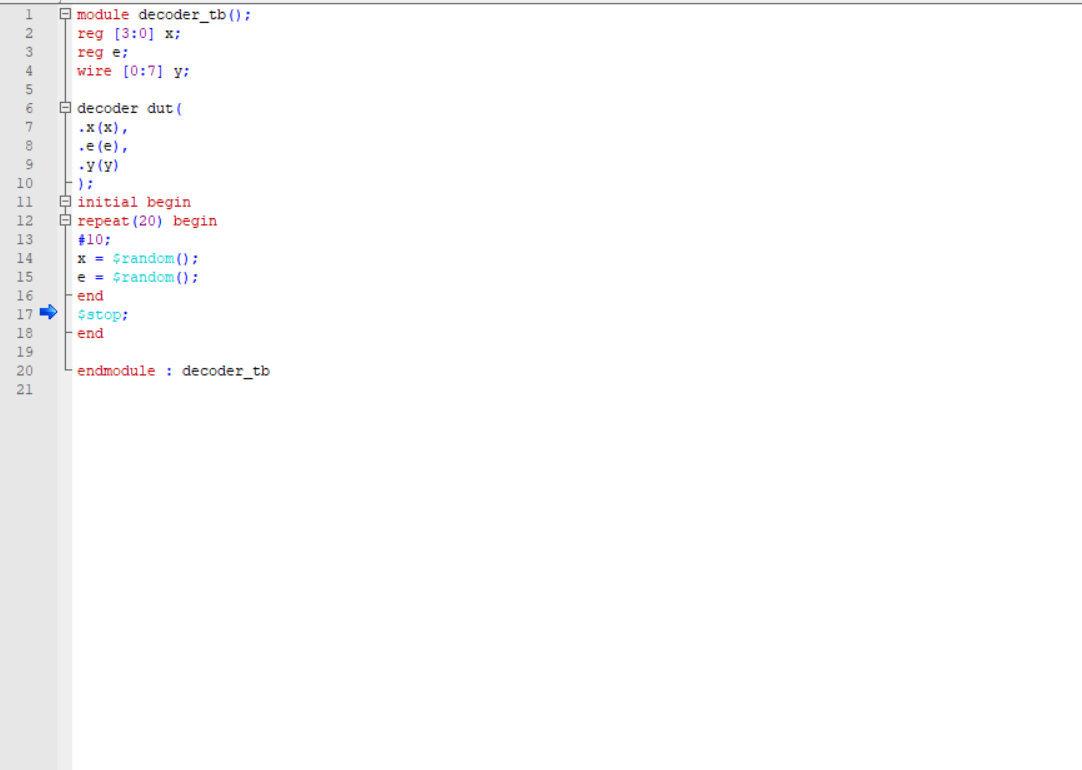
**Decoder 3x8**



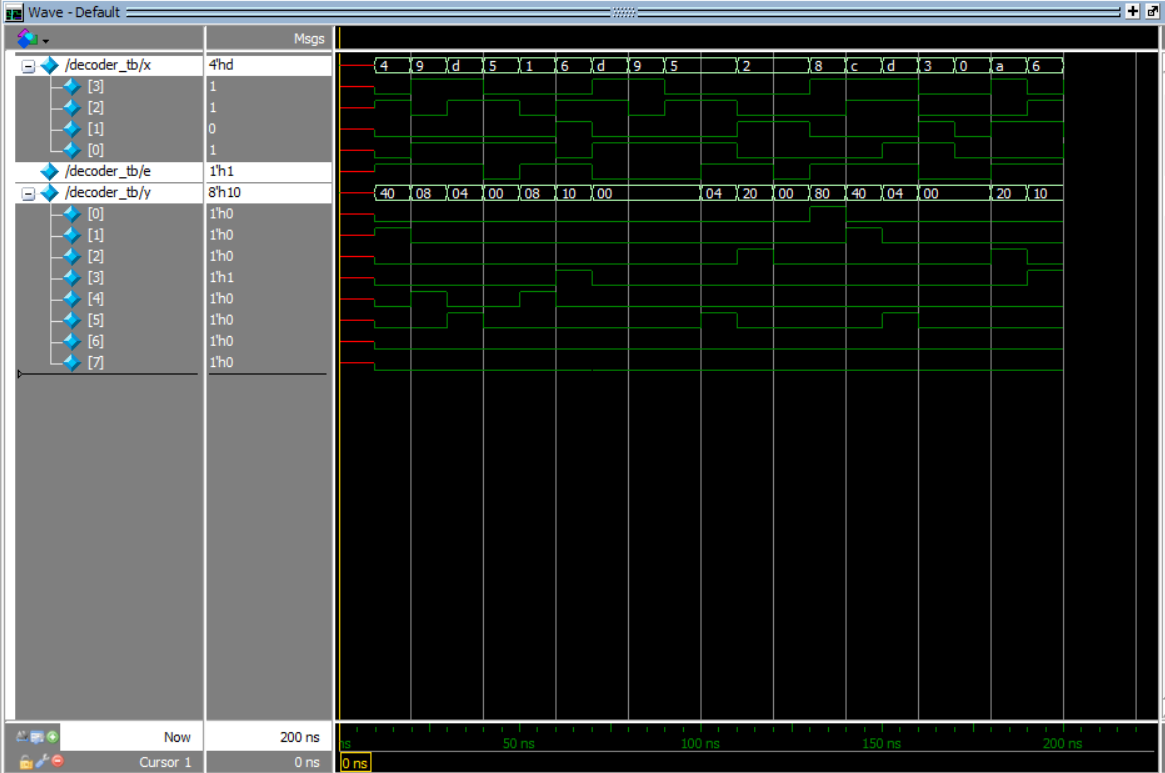
Verilog



Testbench

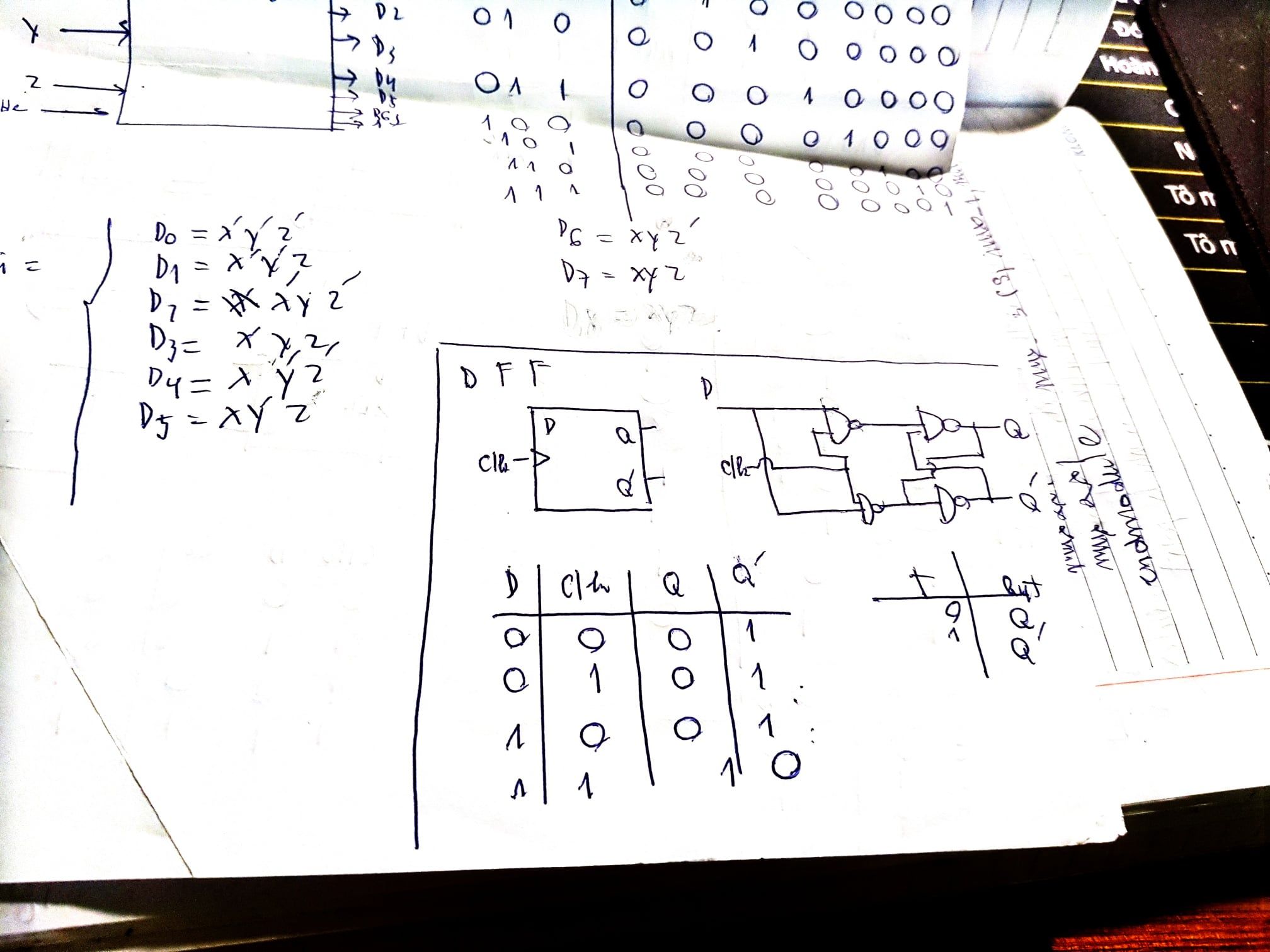


Wave



**Ex4**

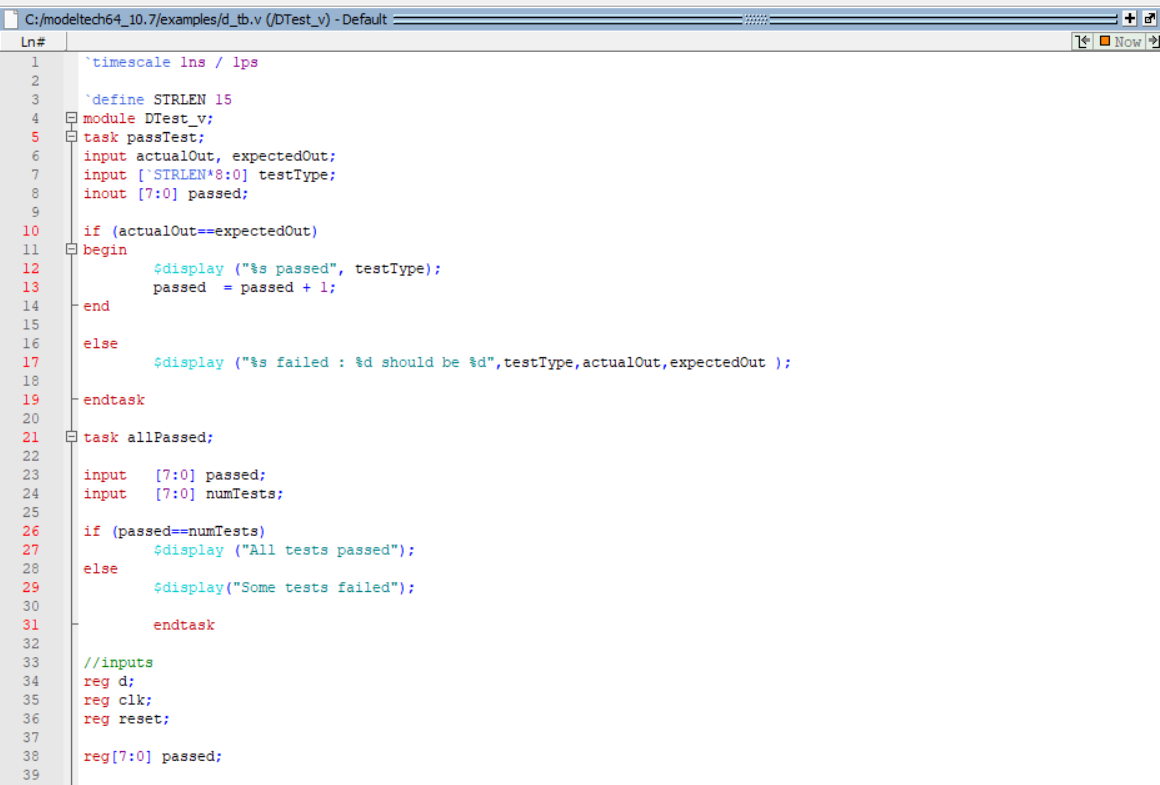
**DFF**



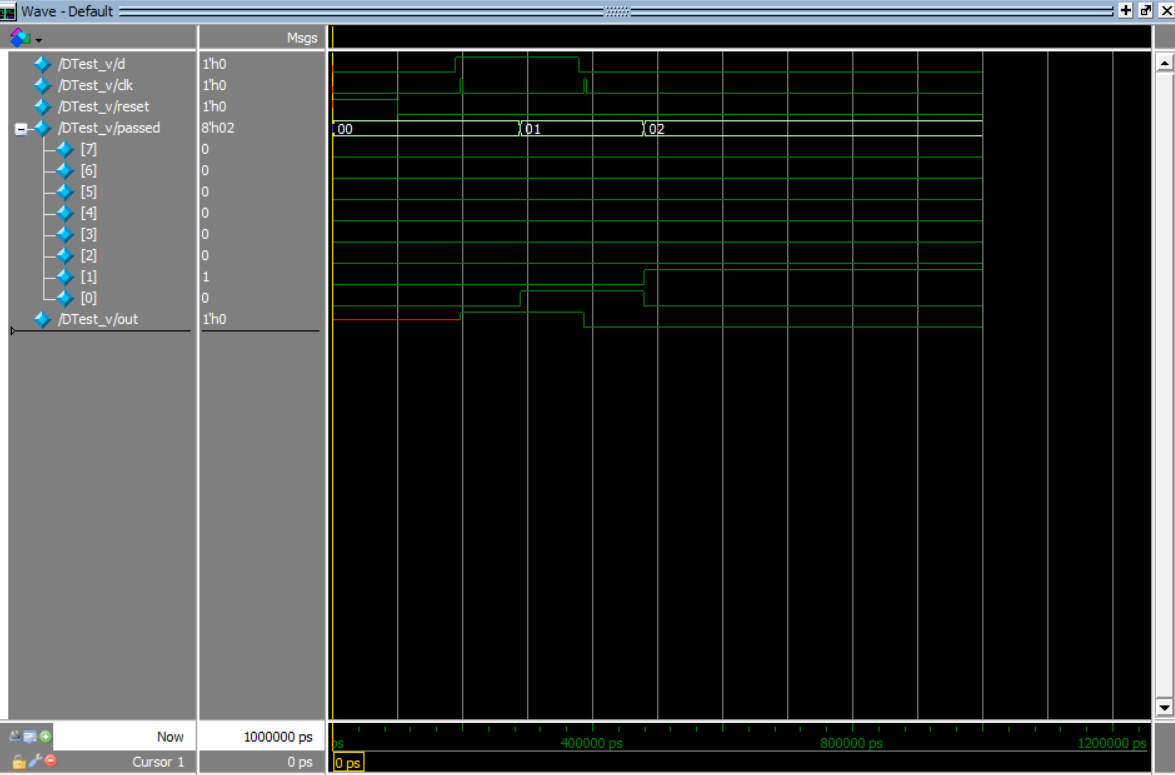
Verilog



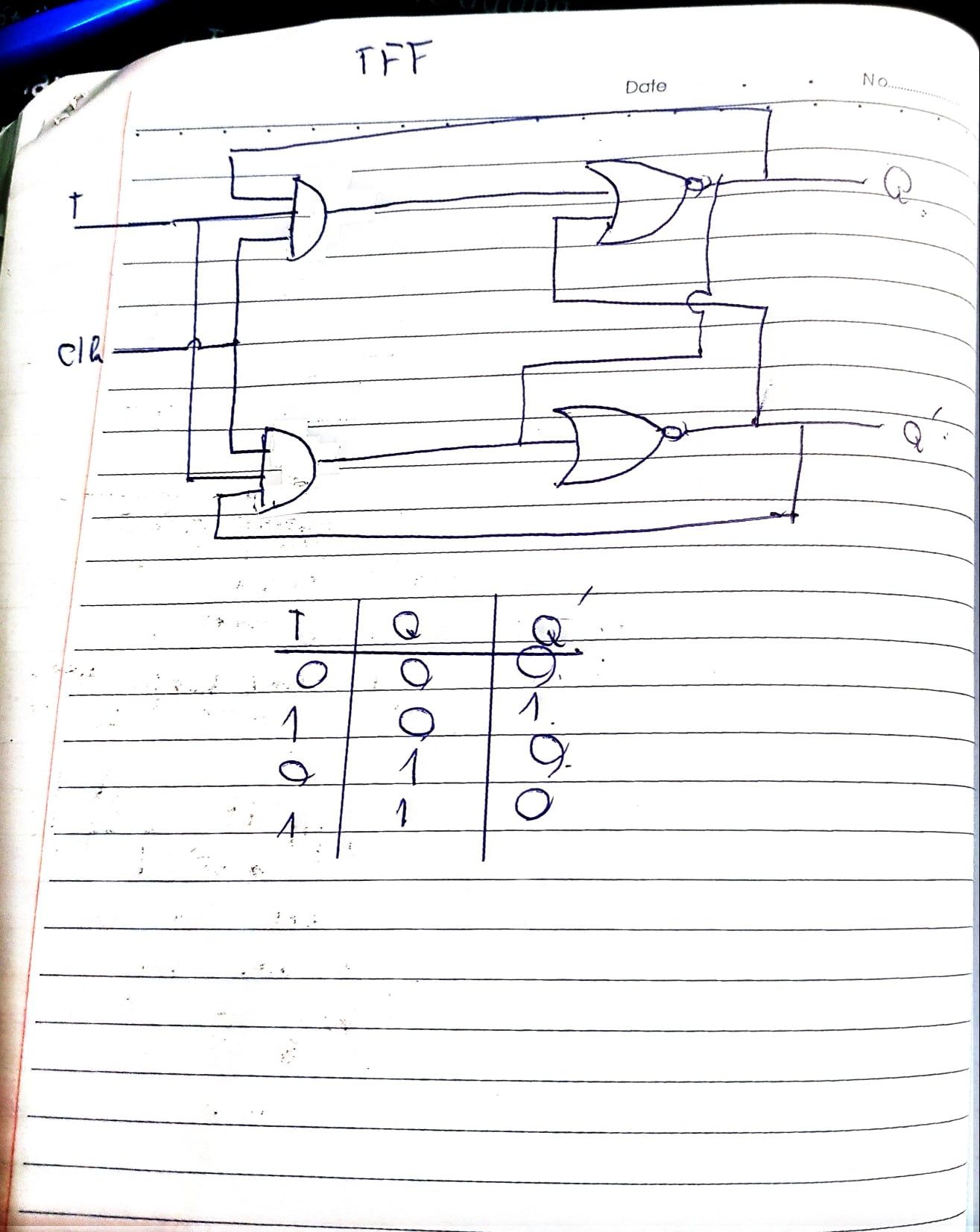
Testbench



Wave



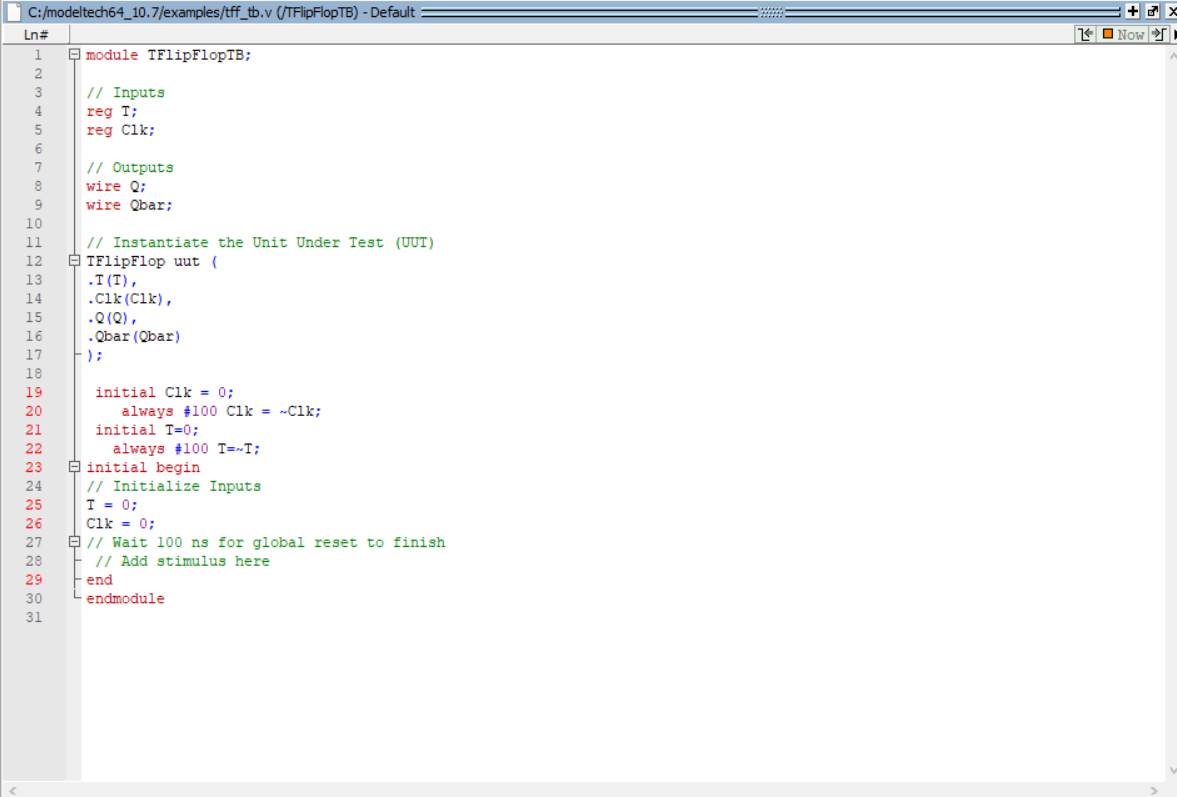
**TFF**



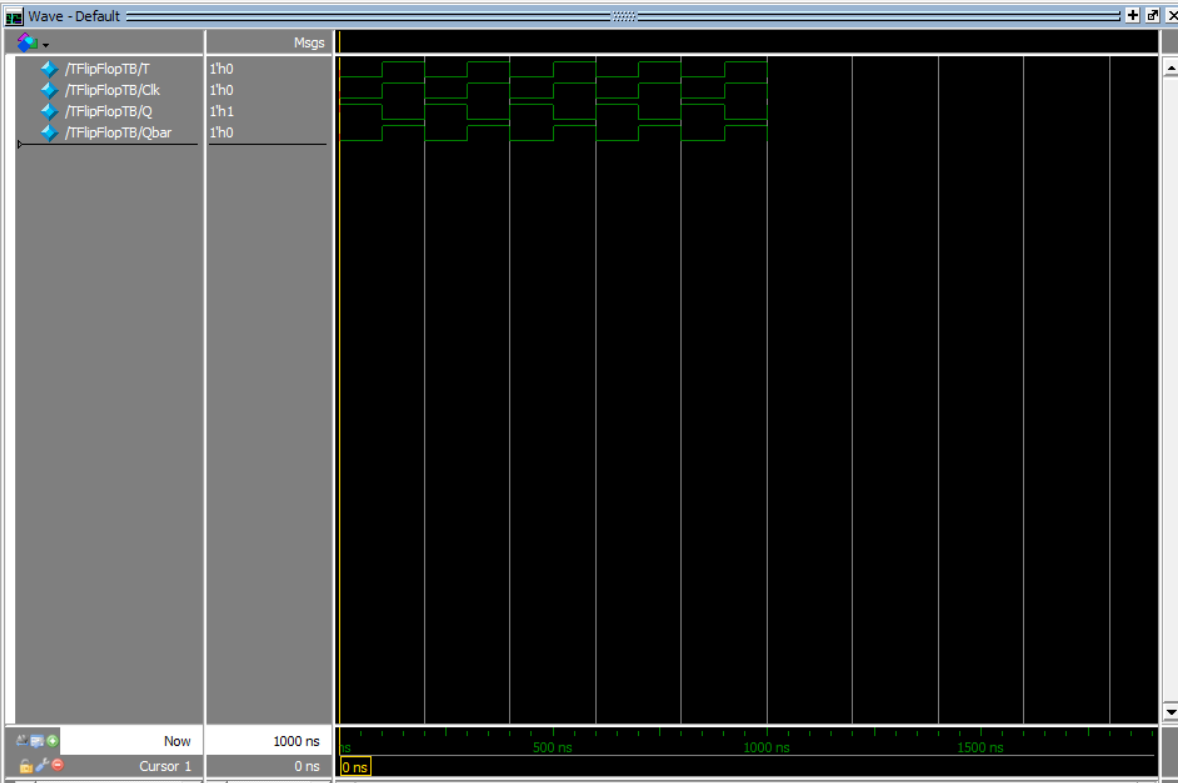
Verilog



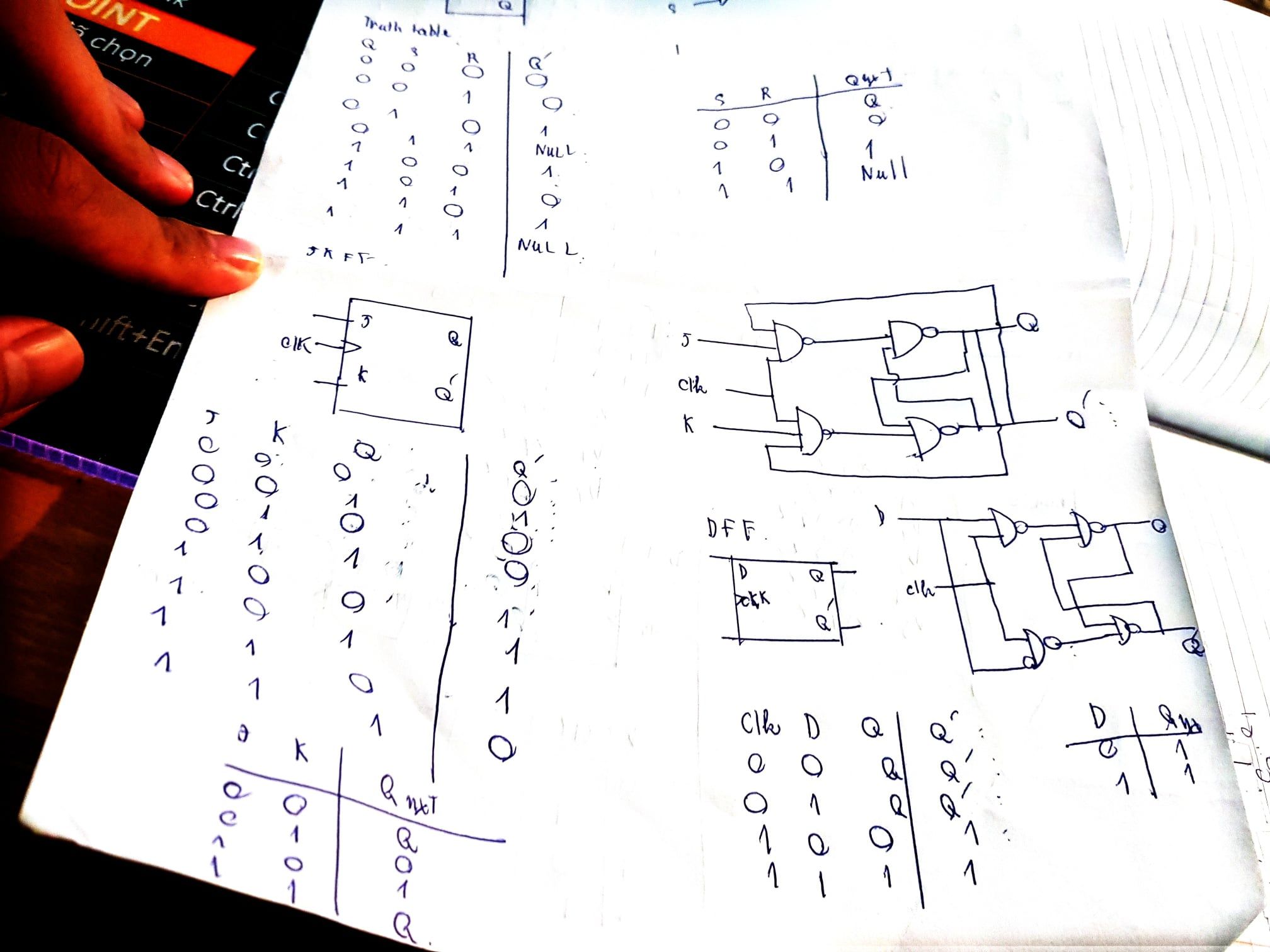
Testbench



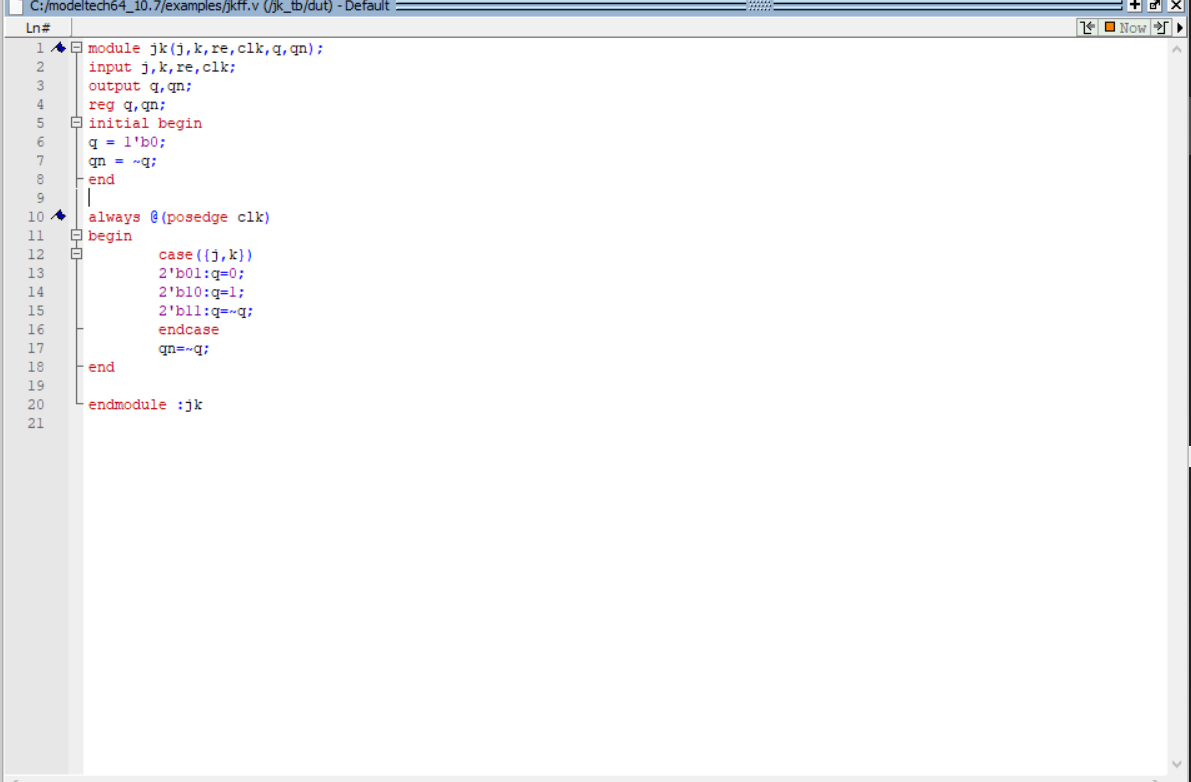
Wave



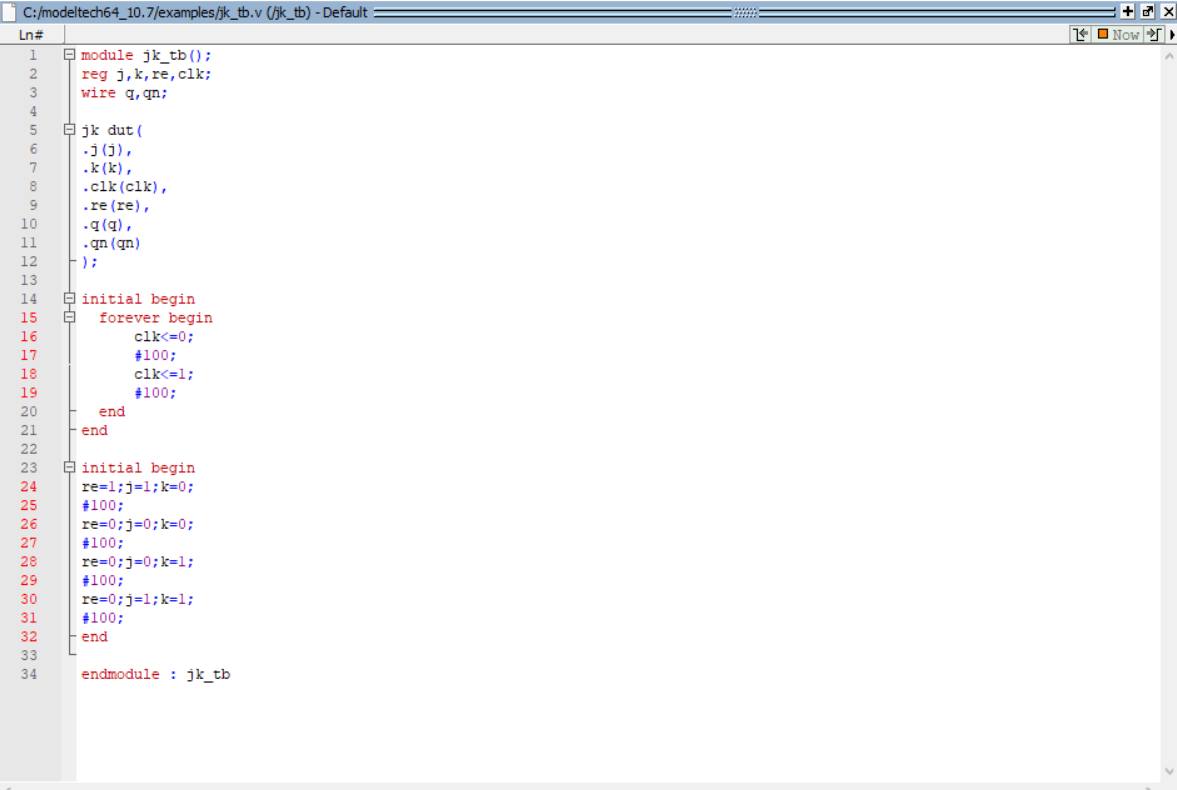
**JKFF**



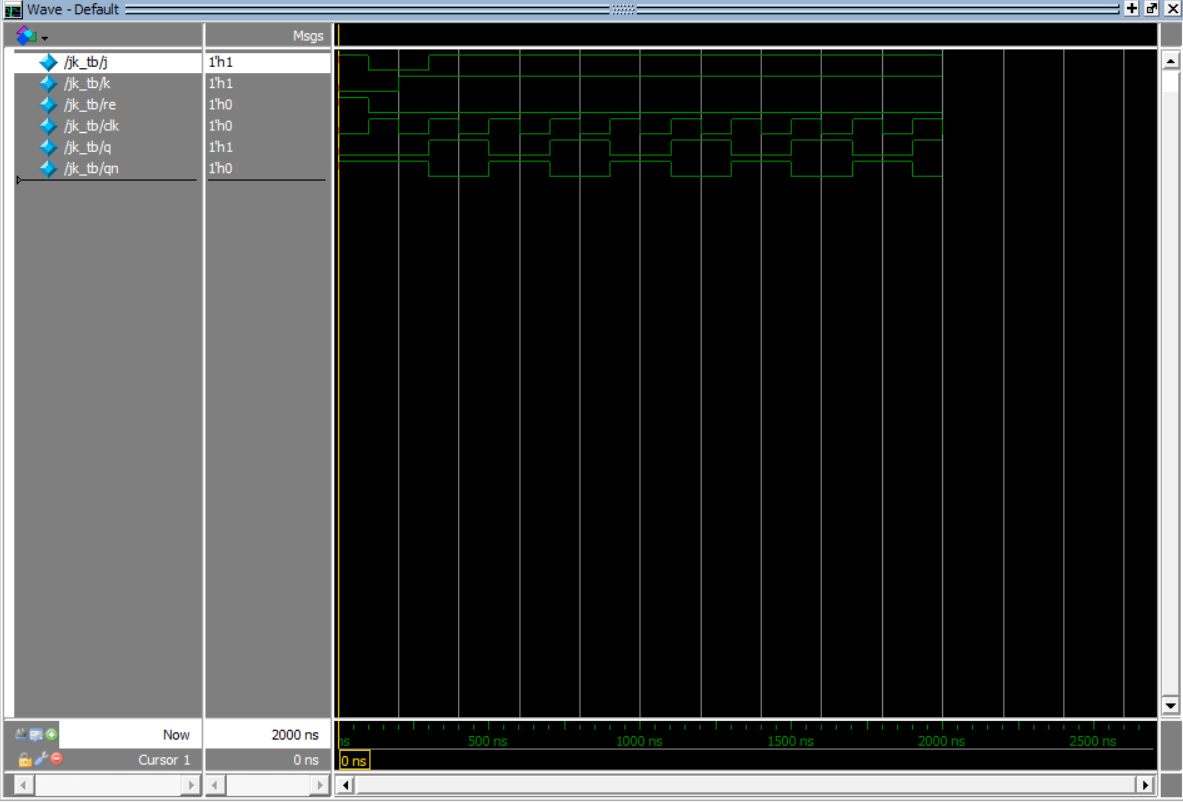
Verilog

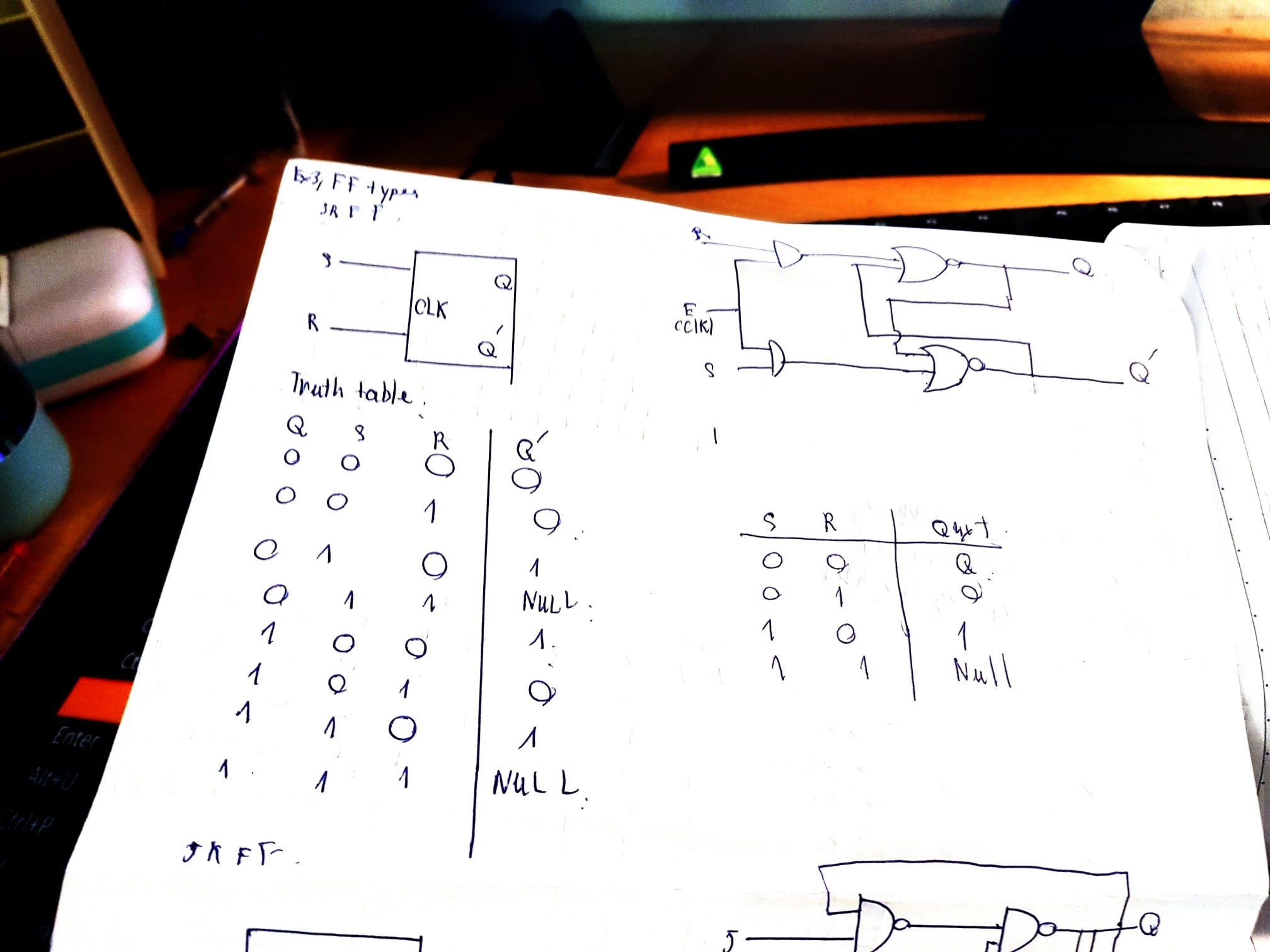


Testbench

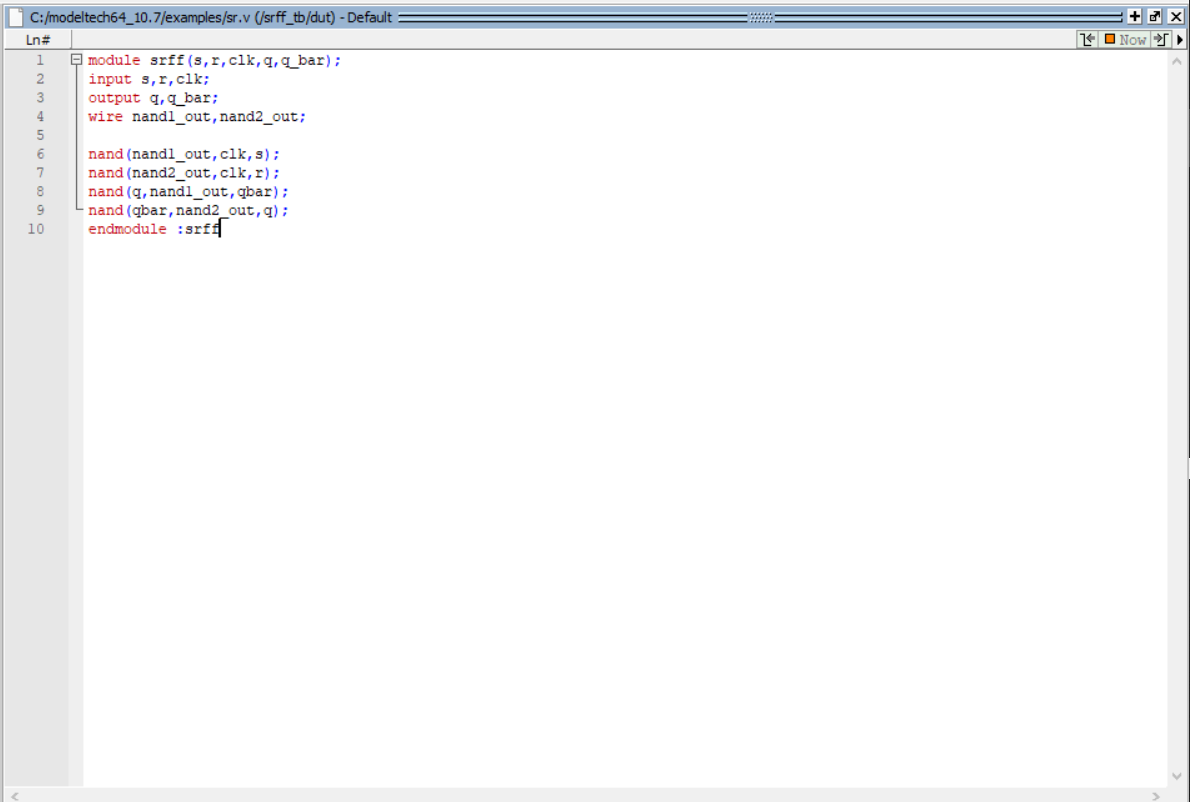


Wave

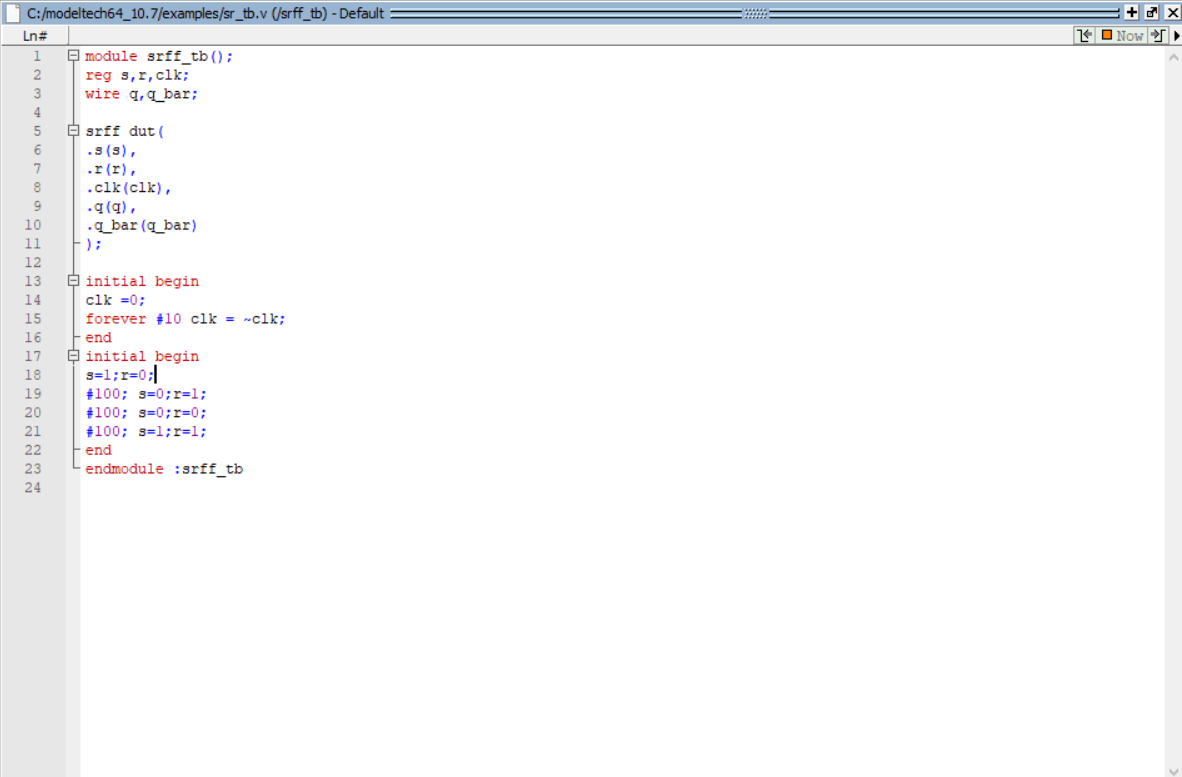
**  
SRFF**



Verilog



Testbench



Wave

