

Group Name:

Guapos

Names:

Angel Zuniga

Long Nguyen

Josue Garcia

[CS 3650] Project 0: Verilog Waves

Prompt:

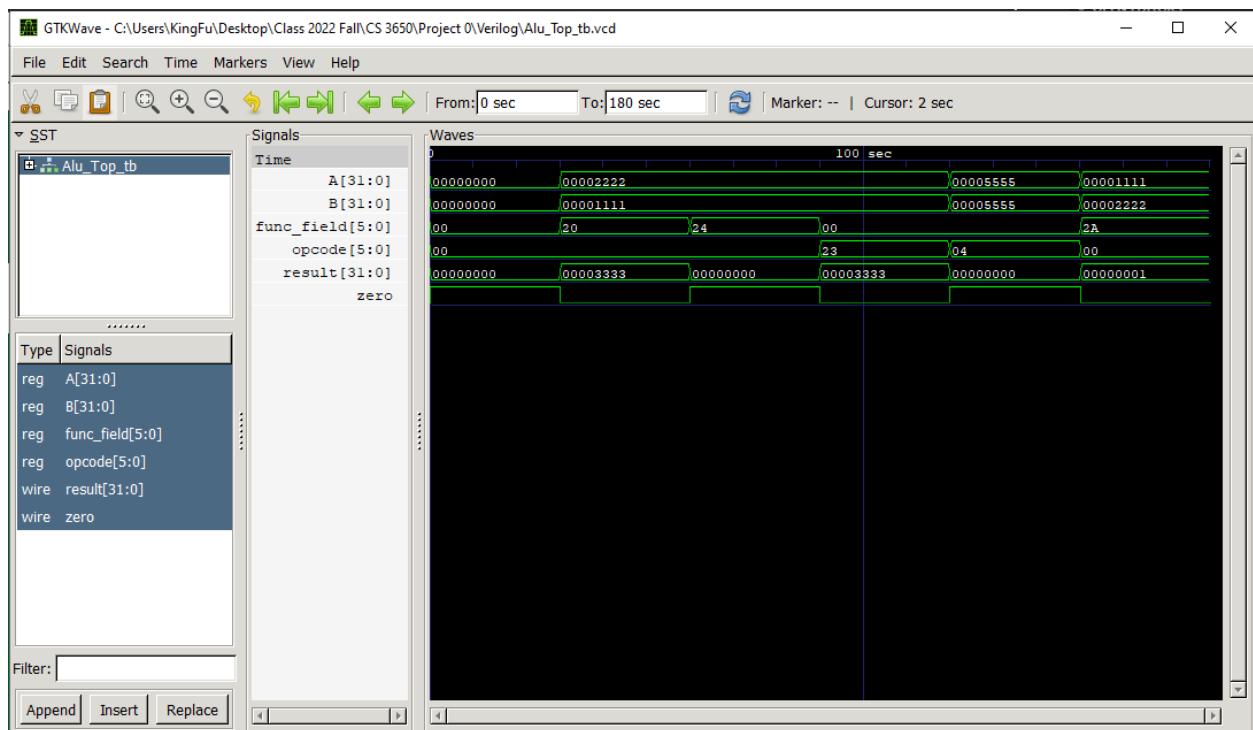
- Submit your notes of learning, including 1 or more pictures of the waves. You can also include the traffic light controller (8.10) waves for 10 extra points. You will need to write your own test bench for the traffic light controller.

Notes of learning:

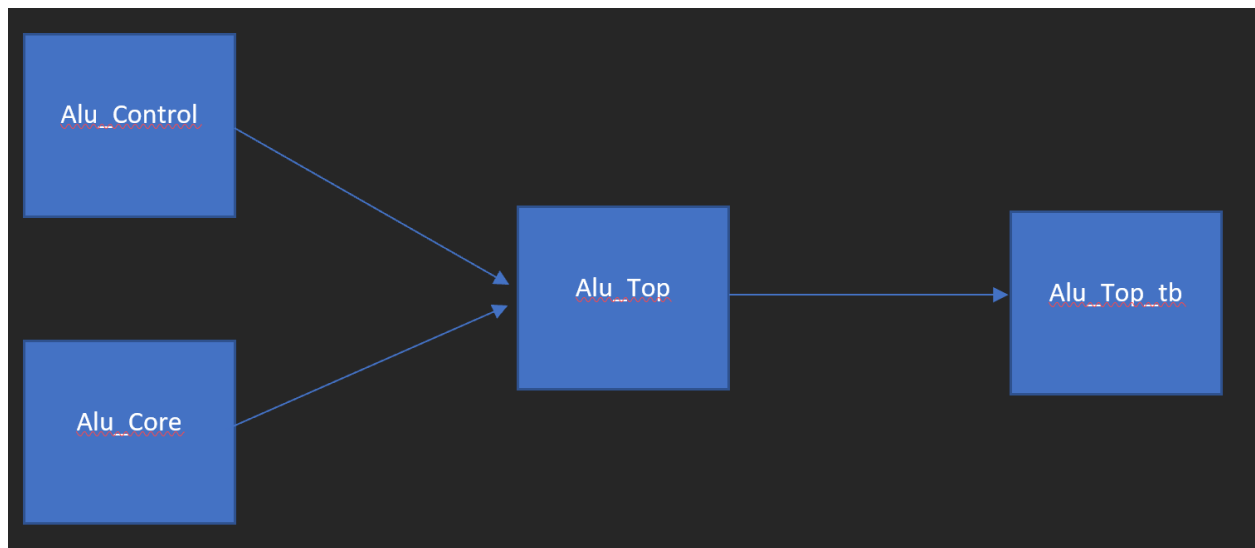
Picture(s) of the waves:

Code: <https://github.com/KingFuGitHub/Verilog> (including extra credit codes)

Image:



The four Verilog files from the blog post are Alu_Control.v, Alu_core.v, Alu_top.v, and Alu_Top_tb.v.



Just from our observations we saw that Alu_Control and Alu_Core were connected to Alu_Top and then Alu_Top was connected to Alu_Top_tb. (Observation above)

In Alu_Control there are 4 ports. Two inputs that have a 6 bit vector set, opcode and func_field. Then there is one output register alu_control that has a 3 bit vector set and a regular register 3 bit vector set called func_code. We observed that for the singular expression func_field, it is assigning 6 hexadecimal bit numbers 20 to 2A to the func_code with 3 hexadecimal bit numbers 0 to 5. The same pattern is also evident with the singular expression opcode. However, it uses opcode as its singular expression and sets alu_control to various 6 hexadecimal bit numbers to 3 hexadecimal bit numbers.

Alu_Core has 4 different ports. This time we have three inputs A and B have a 32 bit vector set, a 3 bit vector set called alu_control and two output ports. One being a 32 bit vector set register called result and a wire called zero with no vector set. However, zero is later assigned the logical not bitwise OR of the port result. A big difference we have noticed is that Alu_Core only has one case block consisting of arithmetic and logical expressions for the singular expression Alu_Control.

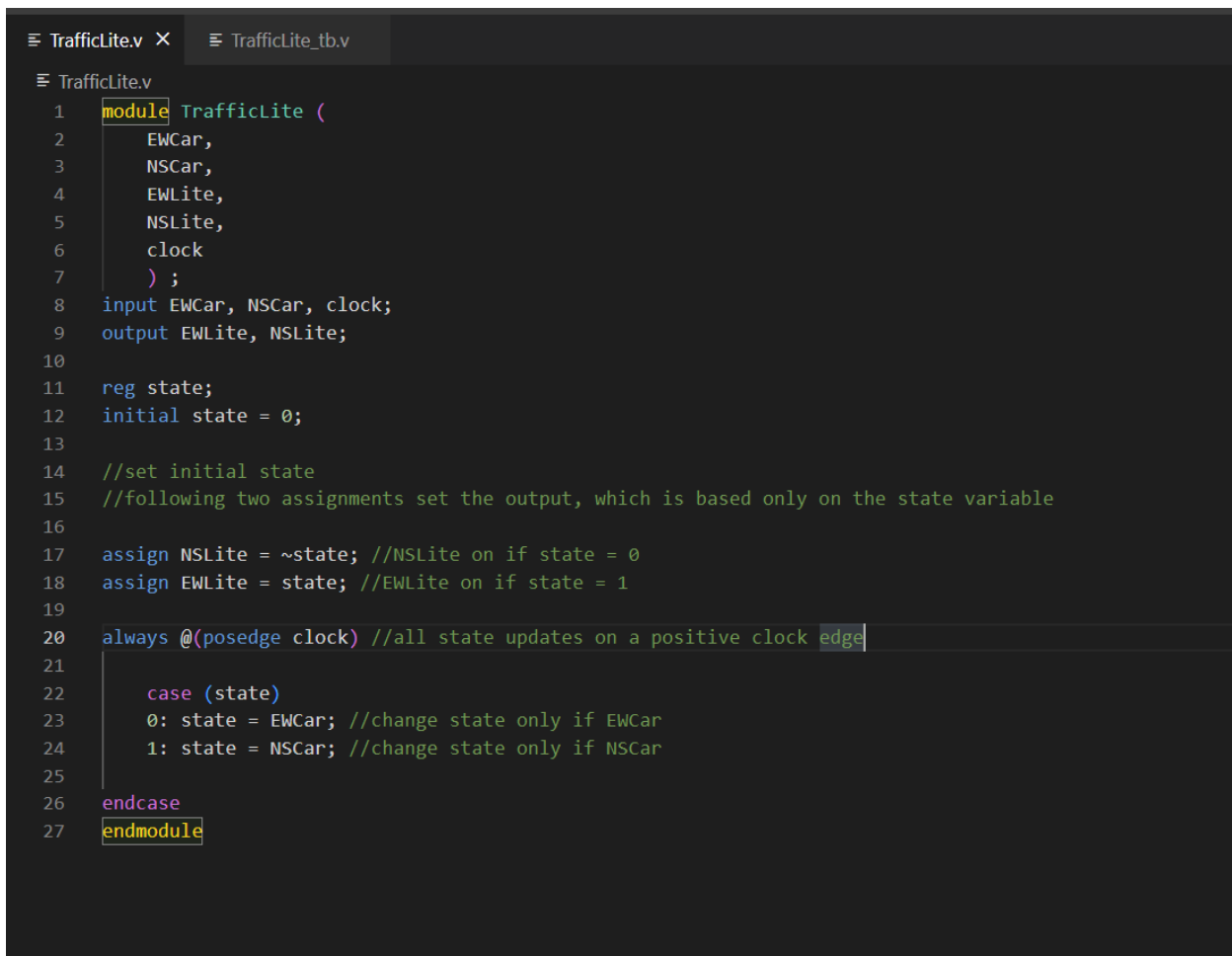
Now, Alu_Top has all the ports from Alu_Control and Alu_Core. Along with instantiation blocks for those Verilog files as well. These ports are connecting or

explicitly linking to the Alu_Control and Alu_Core design. By making the port connection by name within a module instantiations of Alu_Control and Alu_Core, Alu_Top connects to those two files.

Finally, we have the Alu_Top_tb. This file imports from Alu_Top since the instantiations of the other two files are already in that file, there is no need to import those files. There is a single instantiation called Unit Under Test which uses port connection by name to assign different values to each different port. We did add a dumpfile and a dumpvars to dump the state of the design to a vcd file to simulate what is happening in gtkwave.

Extra points: Traffic Light Controller

For the TrafficLite extra credit we created a file called TrafficLite_tb.v which is the test bench for the TrafficLite.v. Furthermore, we included the TrafficLite.v in the TrafficLite_tb.v file to use the module. Next, we initialize and instantiate the registers and wire to set it to 0 or 1. Lastly, we repeated what we did for the Alu and generated the vvp file then vcd file to display the waves in gtkwave.

A screenshot of a Verilog code editor with two tabs: 'TrafficLite.v' and 'TrafficLite_tb.v'. The 'TrafficLite.v' tab is active, showing the module definition for 'TrafficLite'. The code is as follows:

```
1 module TrafficLite (  
2     EWCAR,  
3     NSCAR,  
4     EWLITE,  
5     NSLITE,  
6     clock  
7 ) ;  
8 input EWCAR, NSCAR, clock;  
9 output EWLITE, NSLITE;  
10  
11 reg state;  
12 initial state = 0;  
13  
14 //set initial state  
15 //following two assignments set the output, which is based only on the state variable  
16  
17 assign NSLITE = ~state; //NSLITE on if state = 0  
18 assign EWLITE = state; //EWLITE on if state = 1  
19  
20 always @(posedge clock) //all state updates on a positive clock edge  
21  
22     case (state)  
23     0: state = EWCAR; //change state only if EWCAR  
24     1: state = NSCAR; //change state only if NSCAR  
25  
26 endcase  
27 endmodule
```

≡ TrafficLite.v ≡ TrafficLite_tb.v X ≡ Alu_Control.v

≡ TrafficLite_tb.v

```
1  `timescale 1ns / 1ns
2  `include "TrafficLite.v"
3
4  module TrafficLite_tb;
5
6  reg EWCAR;
7  reg NSCAR;
8  reg clock;
9
10 wire EWLite;
11 wire NSLite;
12
13 TrafficLite uut(
14     .EWCAR(EWCAR),
15     .NSCAR(NSCAR),
16     .EWLite(EWLite),
17     .NSLite(NSLite)
18 );
19
20 initial begin
21     $dumpfile("TrafficLite_tb.vcd");
22     $dumpvars(0, TrafficLite_tb);
23
24     EWCAR = 0;NSCAR = 1;
25     #10;
26
27     NSCAR = 0;EWCAR = 1;
28     #10;
29
30     EWCAR = 0;NSCAR = 1;
31     #10;
32
33     NSCAR = 0;EWCAR = 1;
34     #10;
35
36     EWCAR = 0;NSCAR = 1;
37     #10;
38
39     NSCAR = 0;EWCAR = 1;
40     #10;
41
42     EWCAR = 0;NSCAR = 1;
43     #10;
44
45     NSCAR = 0;EWCAR = 1;
46     #10;
47 end
48 endmodule
```

