Verilog and FPGA Basics

Calvin Reese
cjreese@fortlewis.edu

1/31/22

1 Introduction

This lab we simply got introduced to Verilog and vim. here is the evidence that I participated.

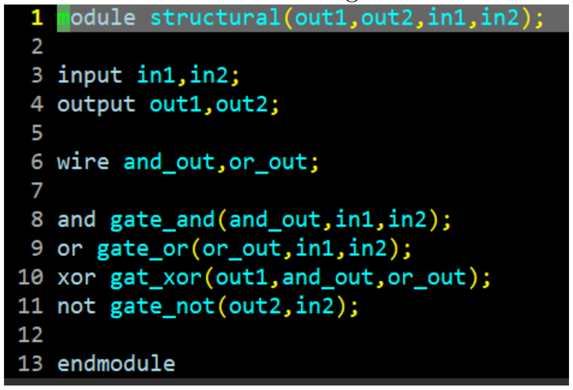
2 Materials and Methods

The tutorial for making these examples are in `http://www.yilectronics.com/Courses/CE433/s2022/lectures/week1_Basics/week1_Basics.html`

3 Results

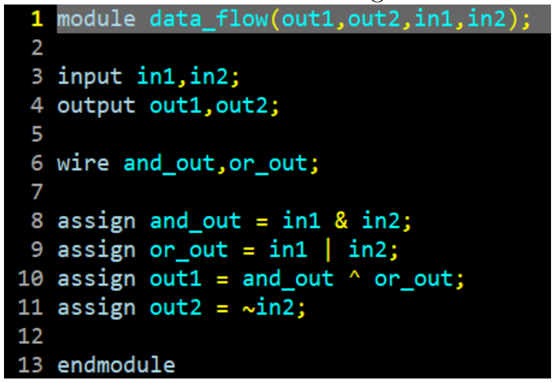
3.1 Task 1

2.1 Code: Structural modeling



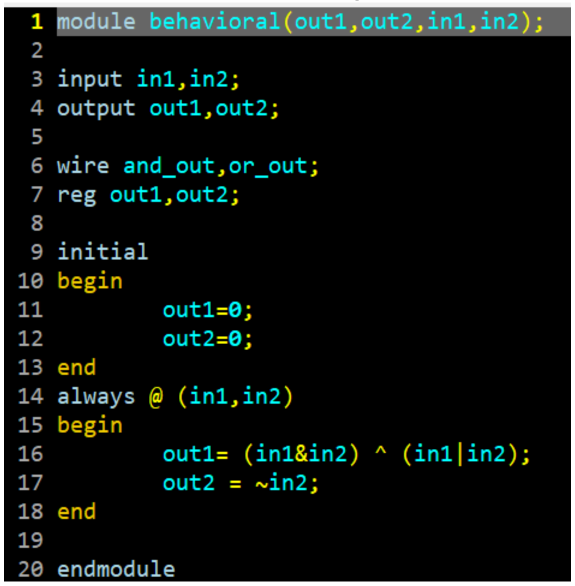
```
module structural(out1,out2,in1,in2);

input in1,in2;
output out1,out2;

wire and_out,or_out;

and gate_and(and_out,in1,in2);
or gate_or(or_out,in1,in2);
xor gat_xor(out1,and_out,or_out);
not gate_not(out2,in2);

endmodule
```

2.2 Code: Dataflow modeling



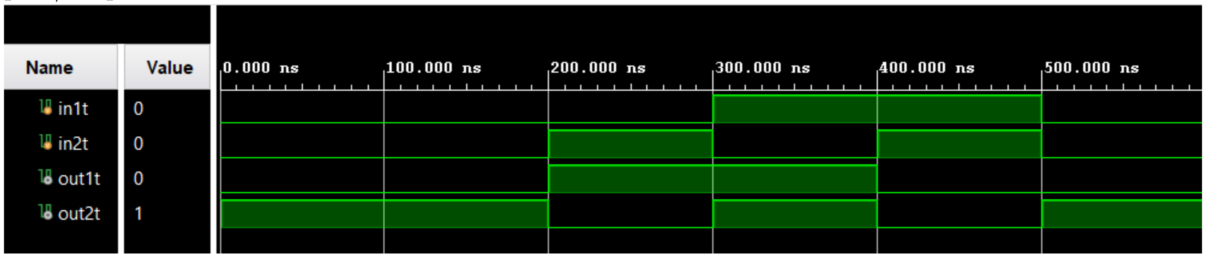
```
module data_flow(out1,out2,in1,in2);

input in1,in2;
output out1,out2;

wire and_out,or_out;

assign and_out = in1 & in2;
assign or_out = in1 | in2;
assign out1 = and_out ^ or_out;
assign out2 = ~in2;

endmodule
```

2.3 Code: Behavioral modeling



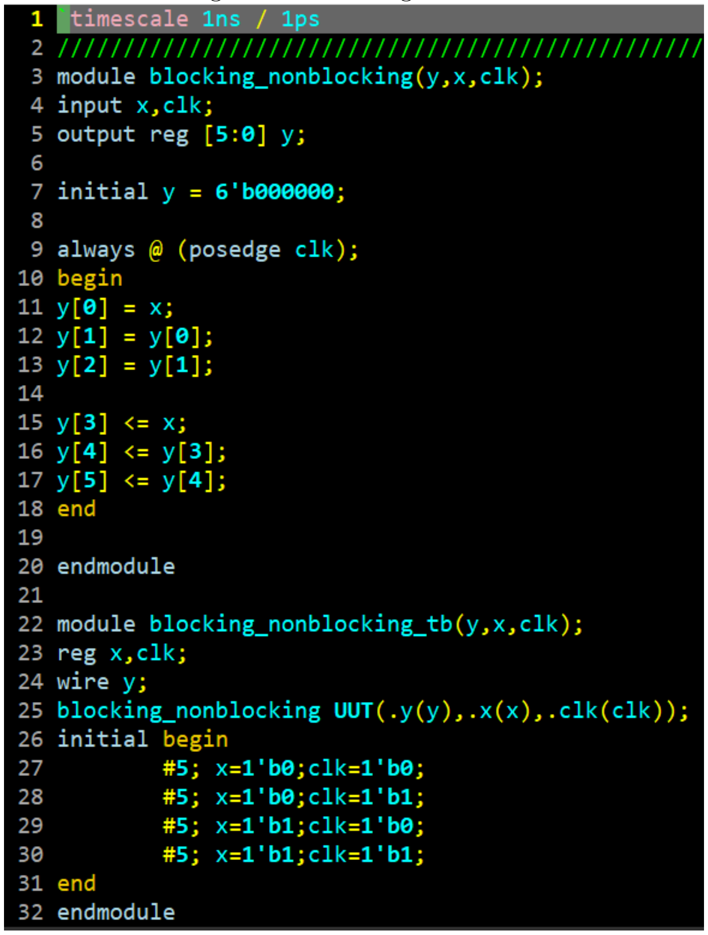
```
module behavioral(out1,out2,in1,in2);

input in1,in2;
output out1,out2;

wire and_out,or_out;
reg out1,out2;

initial
begin
        out1=0;
        out2=0;
end
always @ (in1,in2)
begin
        out1= (in1&in2) ^ (in1|in2);
        out2 = ~in2;
end

endmodule
```

Sim Results for 2.1, 2.2, 2.3. It is the same output for each because it is just the and operation completed in 3 different methods and with the same inputs/outputs.



3.2 Task 2

2.4 Code: blocking vs non-blocking

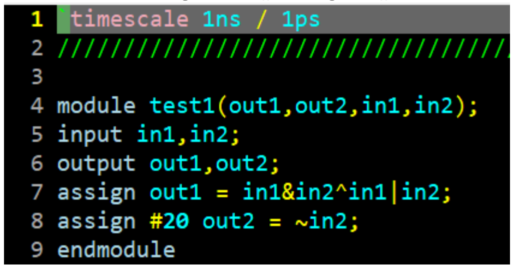


```
`timescale 1ns / 1ps
//////////////////////////////////////////////////
module blocking_nonblocking(y,x,clk);
input x,clk;
output reg [5:0] y;

initial y = 6'b000000;

always @ (posedge clk);
begin
y[0] = x;
y[1] = y[0];
y[2] = y[1];

y[3] <= x;
y[4] <= y[3];
y[5] <= y[4];
end

endmodule

module blocking_nonblocking_tb(y,x,clk);
reg x,clk;
wire y;
blocking_nonblocking UUT(.y(y),.x(x),.clk(clk));
initial begin
        #5; x=1'b0;clk=1'b0;
        #5; x=1'b0;clk=1'b1;
        #5; x=1'b1;clk=1'b0;
        #5; x=1'b1;clk=1'b1;
end
endmodule
```

2.4 Sim: error
The sim refused to run giving the compiling error "[Vivado 12-12986] Compiled library path does not exist: "". However, I know the code would show y[0:3] would be HIGH when x was HIGH and y[4:5] would be LOW regardless of x.

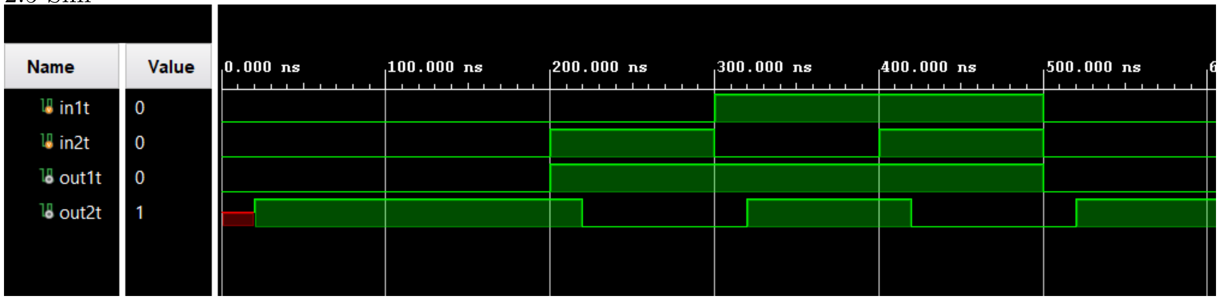
3.3 Task 3

2.5 Code: Using multiple logic operators using dataflow modeling



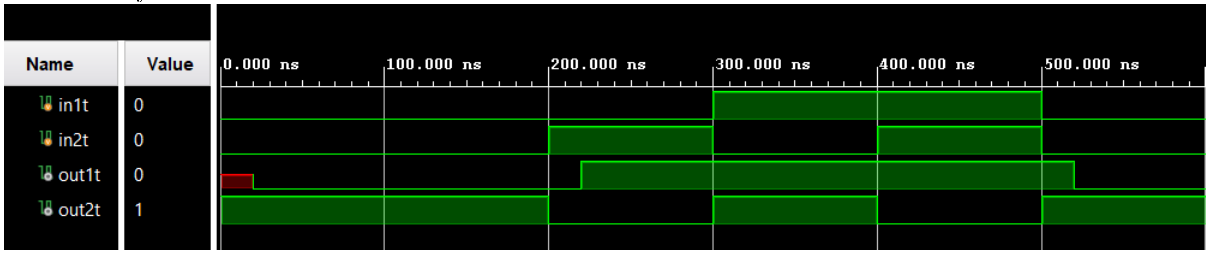
```
`timescale 1ns / 1ps
/////////////////////////////////

module test1(out1,out2,in1,in2);
input in1,in2;
output out1,out2;
assign out1 = in1&in2^in1|in2;
assign #20 out2 = ~in2;
endmodule
```

2.5 Sim

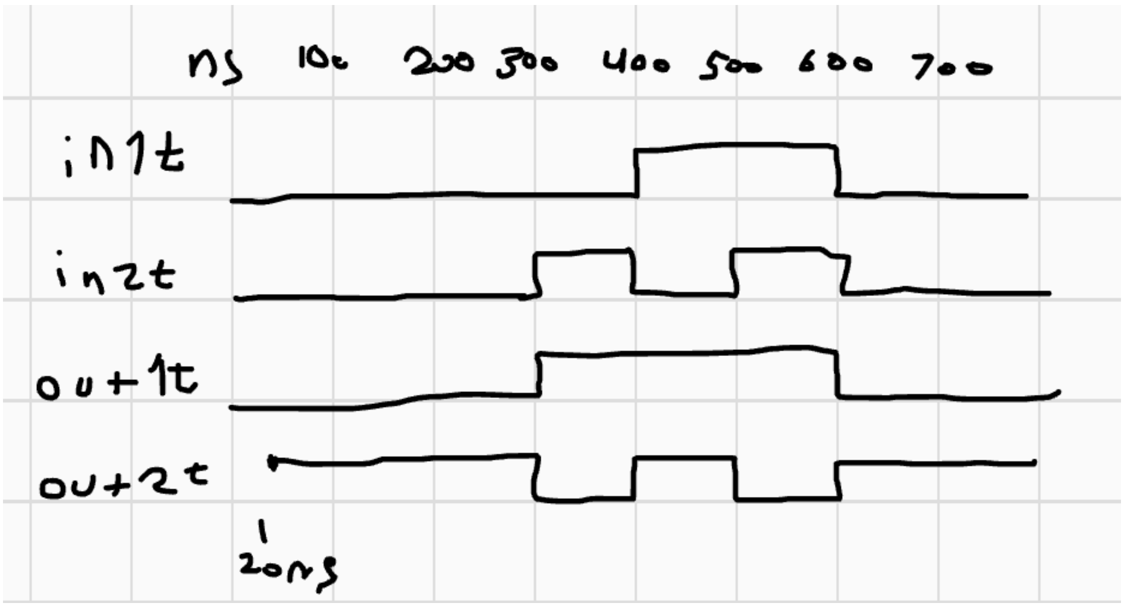


3.4 Task 4

Moved delay for 2.5

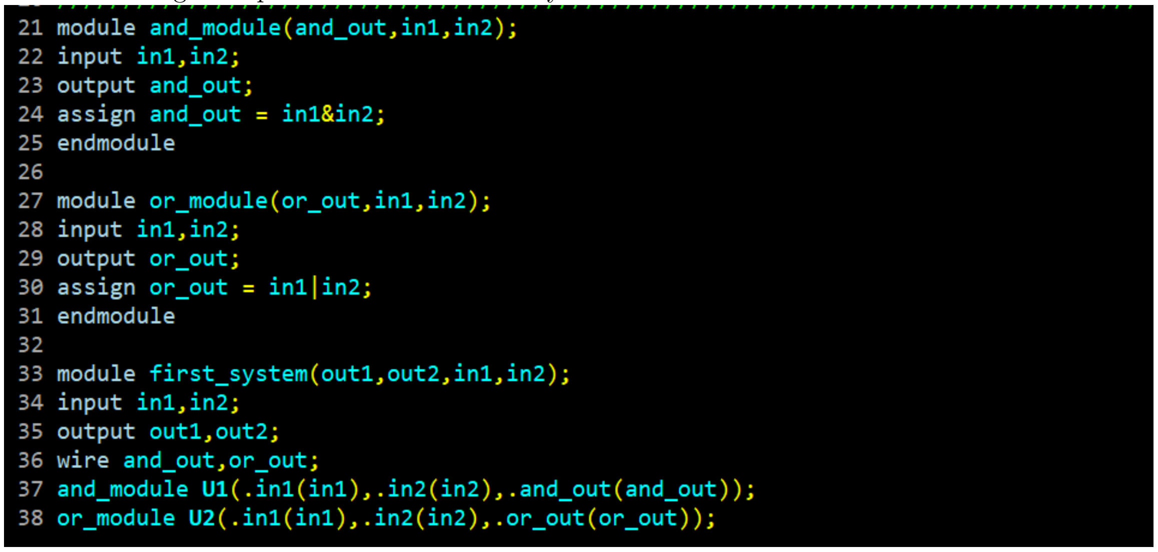


Hand drawn version of the simulation



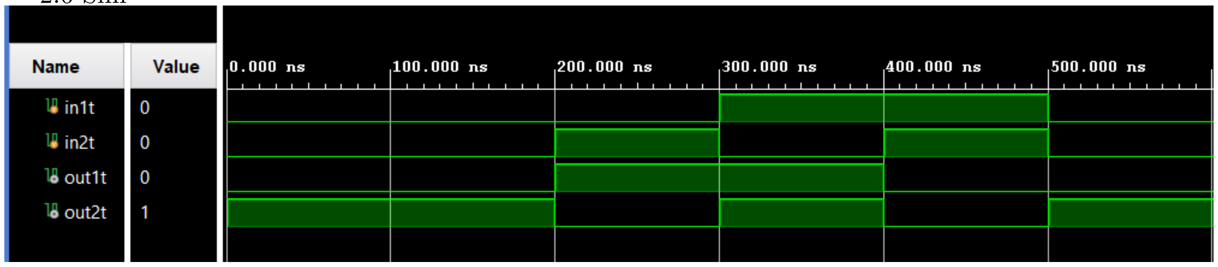
3.5 Task 5

2.6 Code: using multiple modules to create a system



```
module and_module(and_out,in1,in2);
input in1,in2;
output and_out;
assign and_out = in1&in2;
endmodule

module or_module(or_out,in1,in2);
input in1,in2;
output or_out;
assign or_out = in1|in2;
endmodule

module first_system(out1,out2,in1,in2);
input in1,in2;
output out1,out2;
wire and_out,or_out;
and_module U1(.in1(in1),.in2(in2),.and_out(and_out));
or_module U2(.in1(in1),.in2(in2),.or_out(or_out));
```

2.6 Sim



Name	Value
in1t	0
in2t	0
out1t	0
out2t	1

4 Discussion

All of this was fairly simple just to follow your tutorial. The only issue I ran into was that I couldn't run the code for the blocking example (2.4) because it came up with an compiling error. There were no errors on the code, so I eventually decided to just move on.