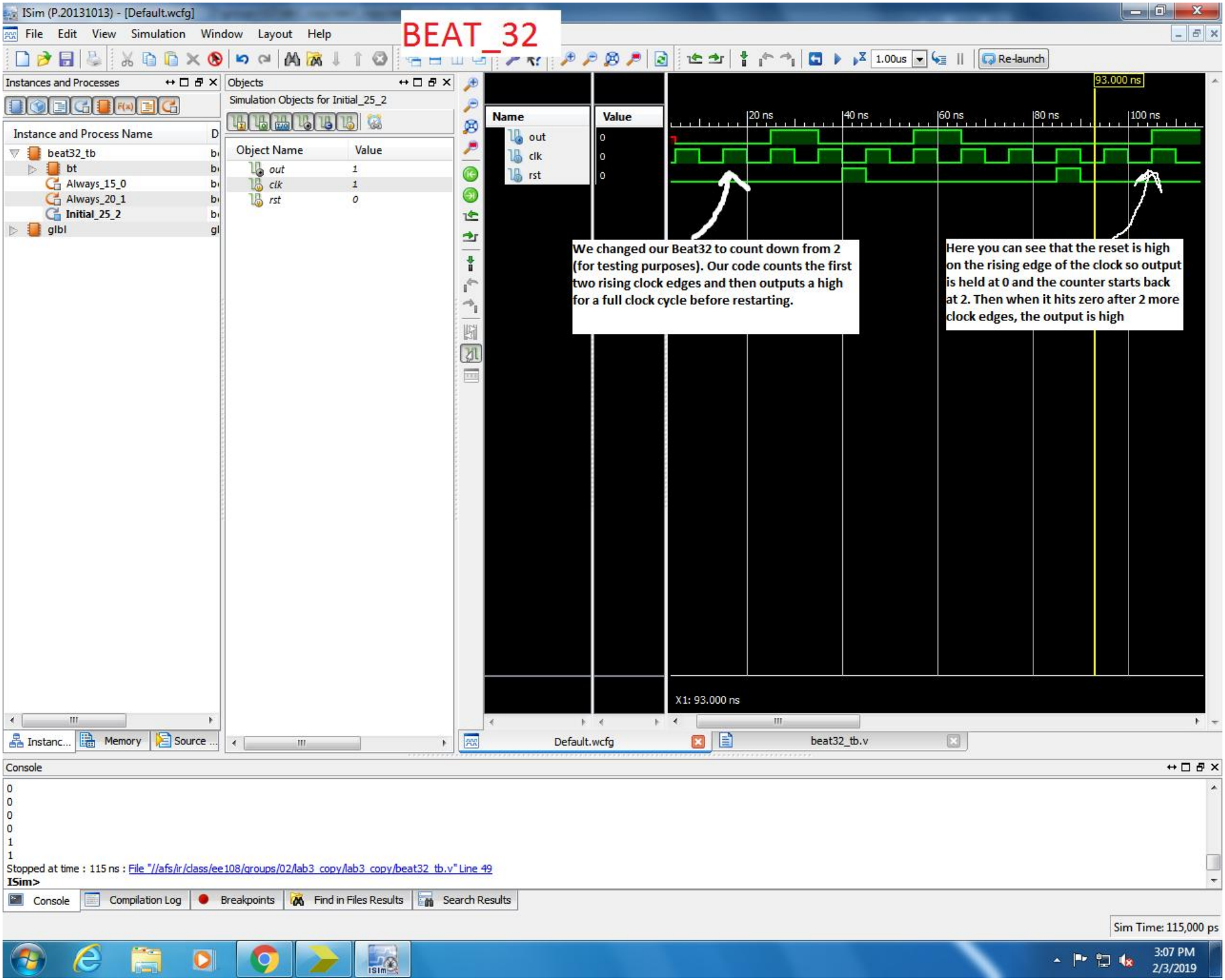
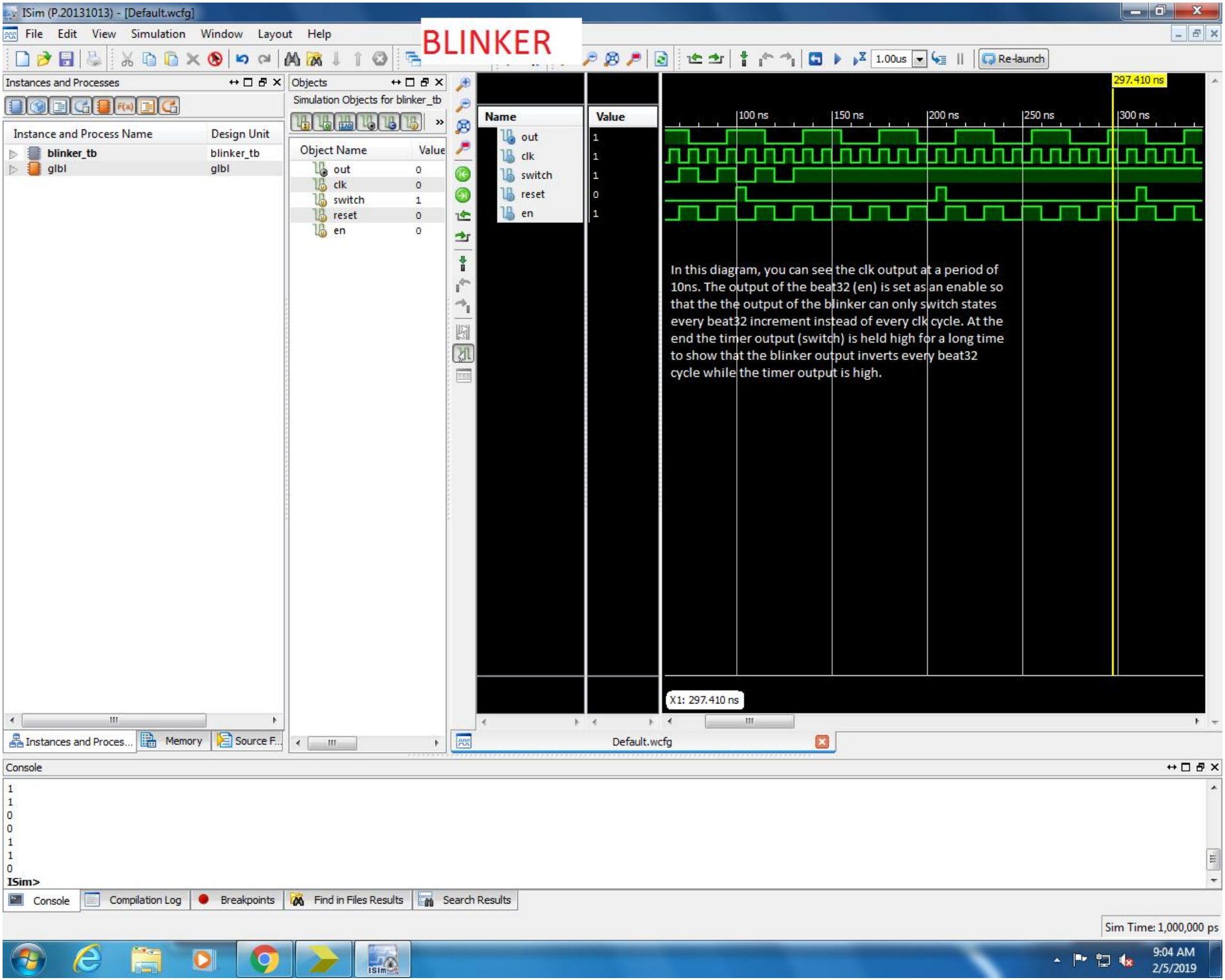
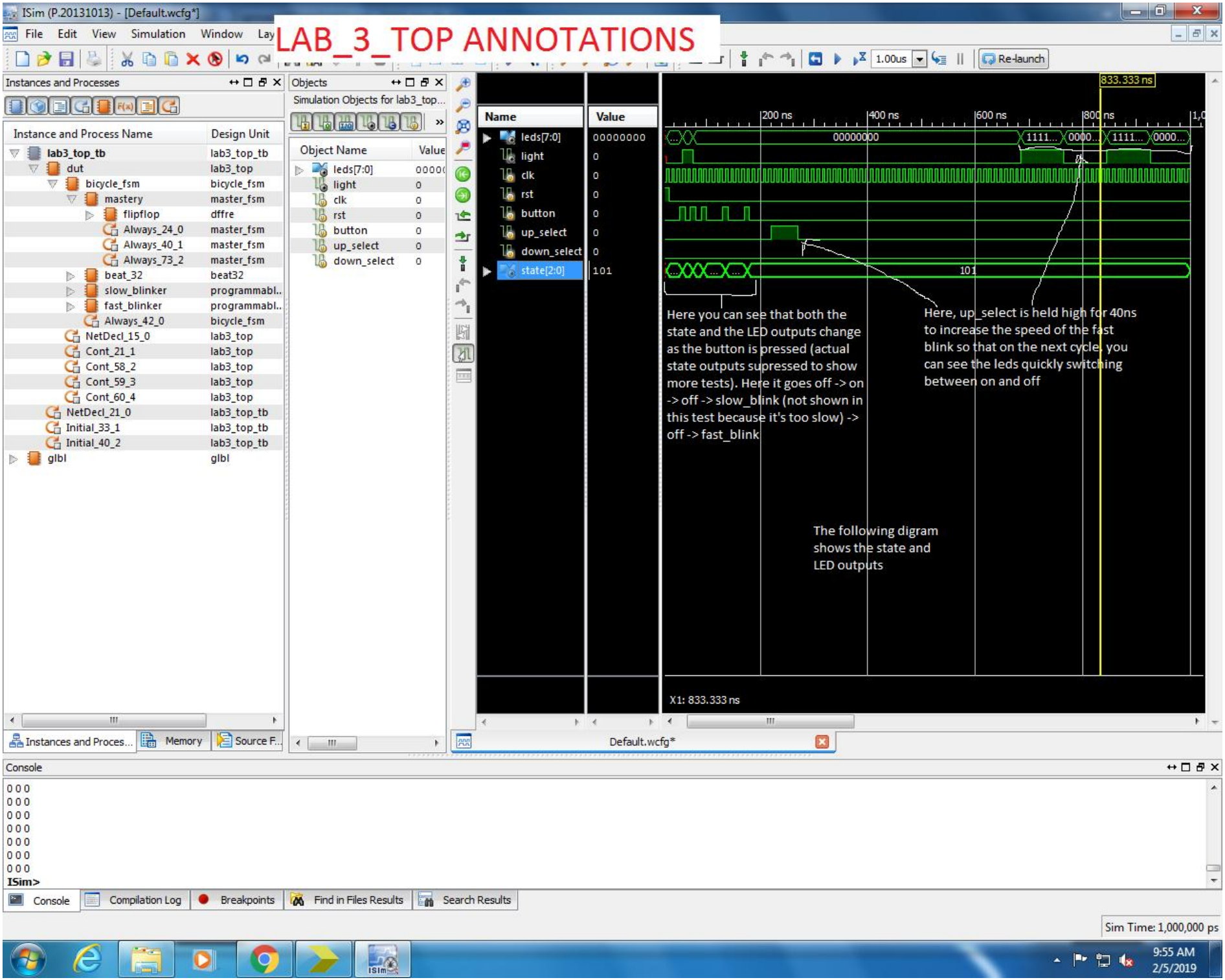
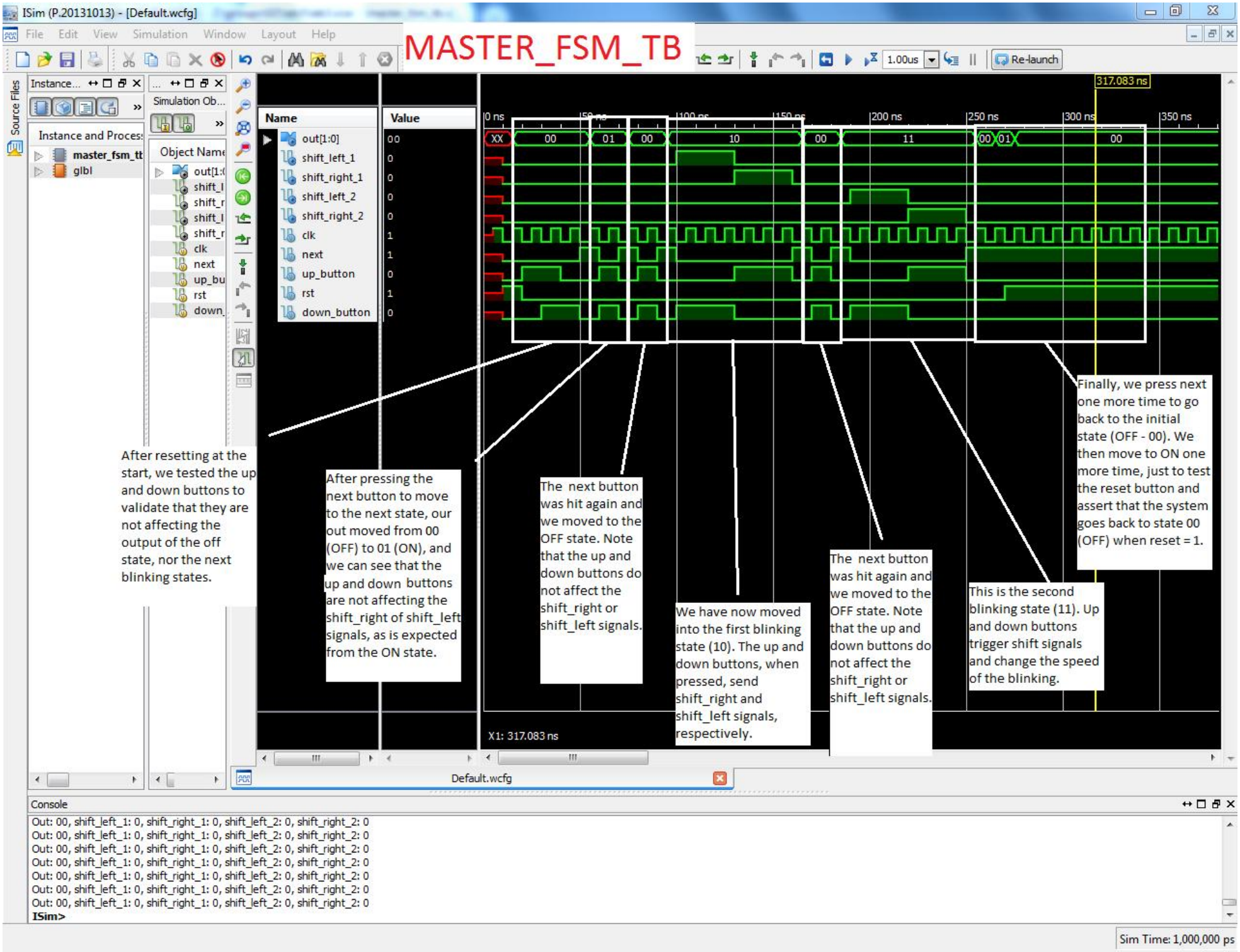
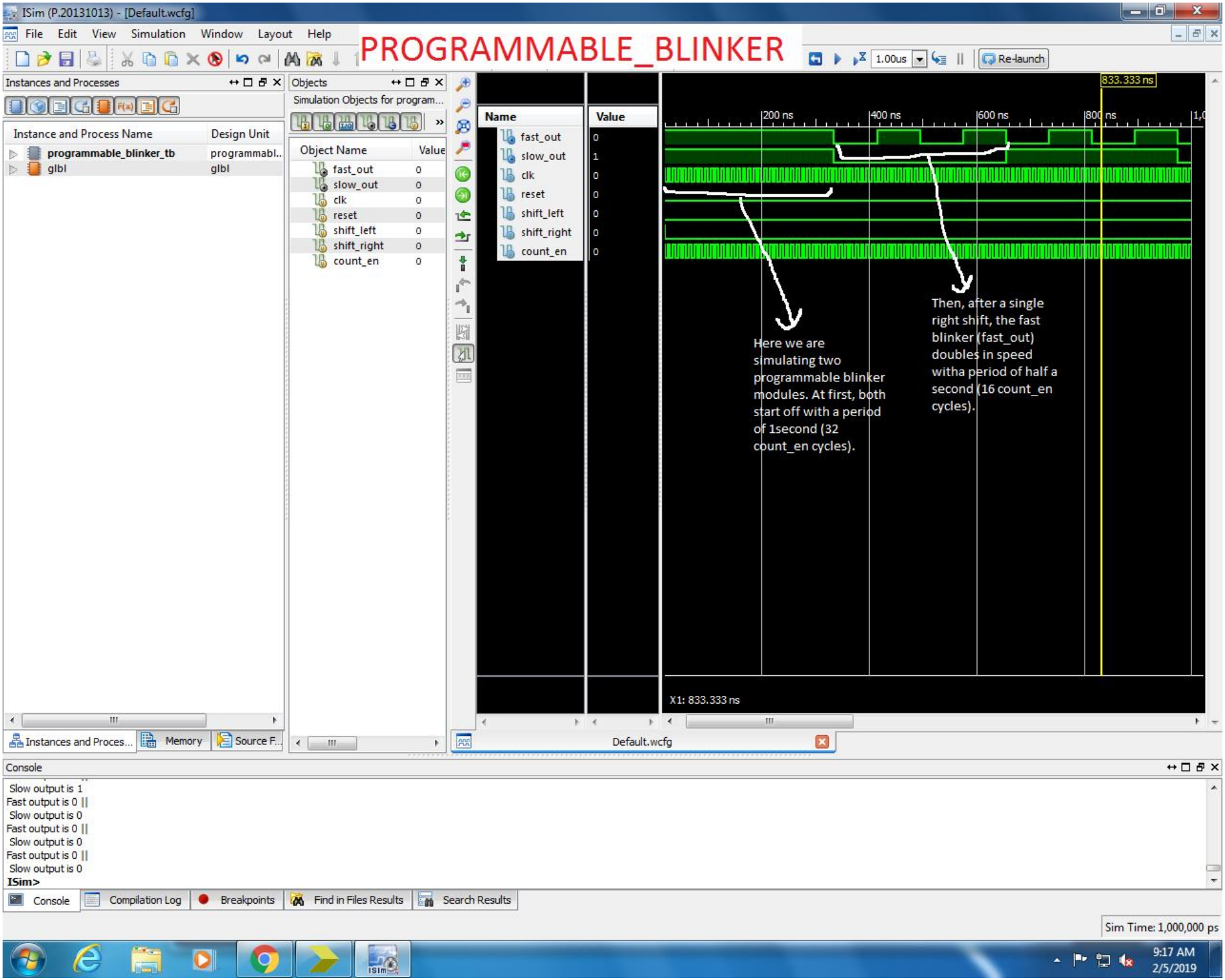

BEAT_32
ISim (P.20131013) - [Default.wcfg]
File Edit View Simulation Window Layout Help
Re-launch
Instances and Processes
Instance and Process Name
beat32_tb
bt
Always_15_0
Always_20_1
Initial_25_2
glbl
Objects
Simulation Objects for Initial_25_2
Object Name
Value
out 1
clk 1
rst 0
Name
Value
out 0
clk 0
rst 0
93.000 ns
20 ns
40 ns
60 ns
80 ns
100 ns
We changed our Beat32 to count down from 2 (for testing purposes). Our code counts the first two rising clock edges and then outputs a high for a full clock cycle before restarting.
Here you can see that the reset is high on the rising edge of the clock so output is held at 0 and the counter starts back at 2. Then when it hits zero after 2 more clock edges, the output is high
X1: 93.000 ns
Instanc...
Memory
Source ...
Default.wcfg
beat32_tb.v
Console
0
0
0
0
1
1
Stopped at time : 115 ns : File "//afs/ir/class/ee108/groups/02/lab3_copy/lab3_copy/beat32_tb.v" Line 49
ISim>
Console
Compilation Log
Breakpoints
Find in Files Results
Search Results
Sim Time: 115,000 ps
3:07 PM
2/3/2019


ISim (P.20131013) - [Default.wcfg]
File Edit View Simulation Window Layout Help
BLINKER
Re-launch
Instances and Processes
Instance and Process Name
Design Unit
blinker_tb
blinker_tb
glbl
glbl
Objects
Simulation Objects for blinker_tb
Object Name
Value
out 0
clk 0
switch 1
reset 0
en 0
Name
Value
out 1
clk 1
switch 1
reset 0
en 1
297.410 ns
100 ns
150 ns
200 ns
250 ns
300 ns
In this diagram, you can see the clk output at a period of 10ns. The output of the beat32 (en) is set as an enable so that the the output of the blinker can only switch states every beat32 increment instead of every clk cycle. At the end the timer output (switch) is held high for a long time to show that the blinker output inverts every beat32 cycle while the timer output is high.
X1: 297.410 ns
Instances and Proces...
Memory
Source F...
Default.wcfg
Console
1
1
0
0
1
1
0
ISim>
Console
Compilation Log
Breakpoints
Find in Files Results
Search Results
Sim Time: 1,000,000 ps
9:04 AM
2/5/2019

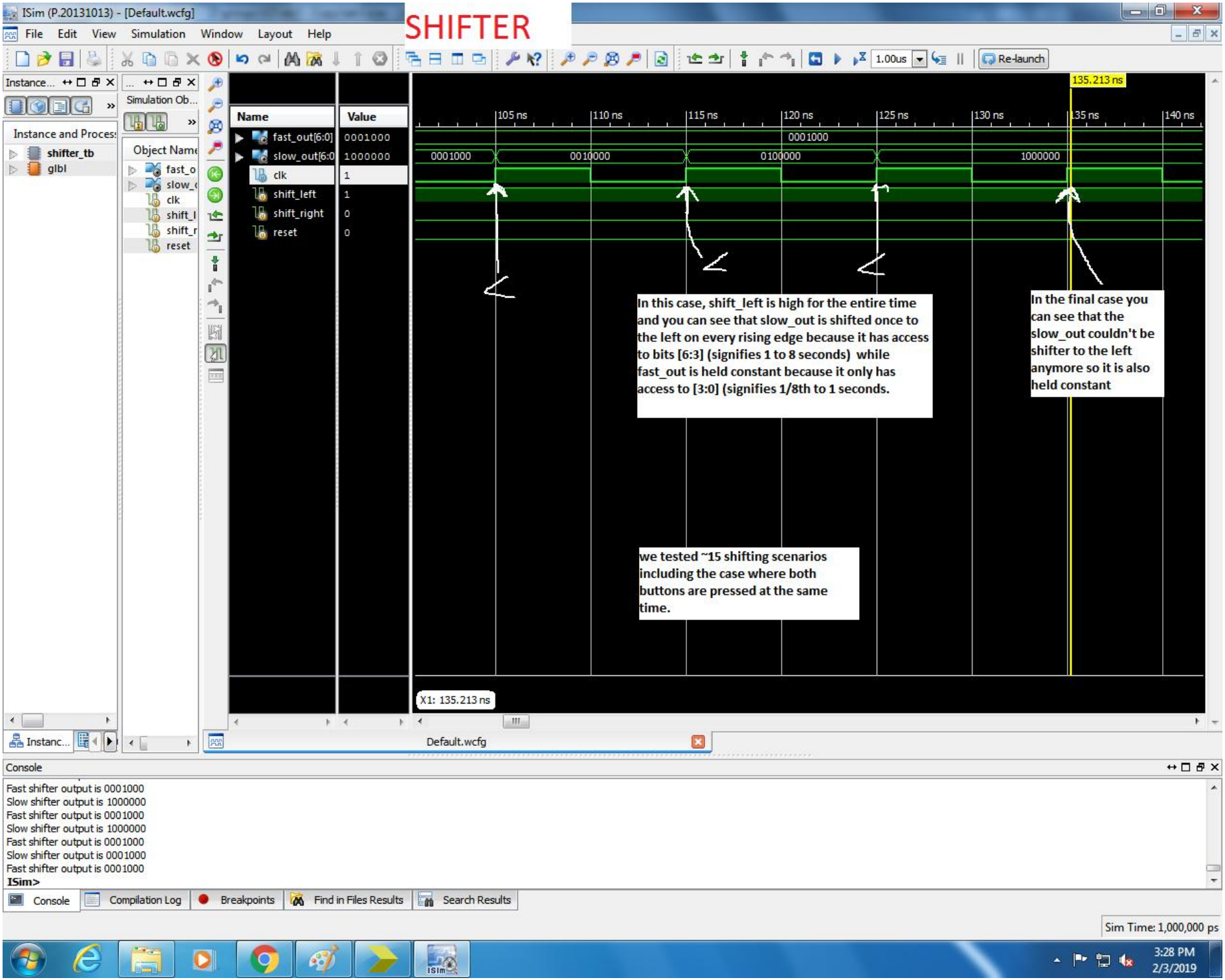
LAB_3_TOP ANNOTATIONS

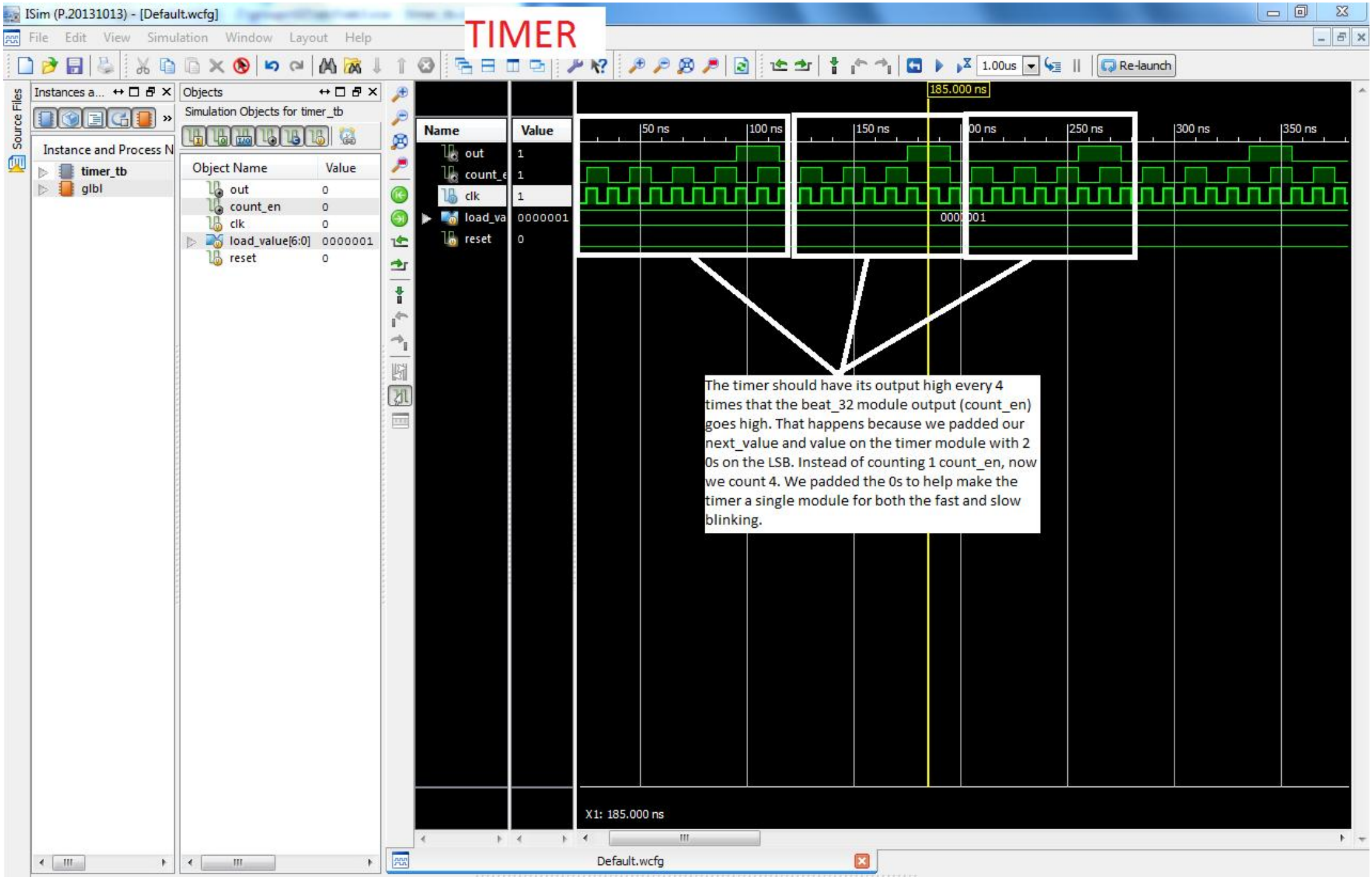



ISim (P.20131013) - [Default.wcfg]
File Edit View Simulation Window Layout Help
MASTER_FSM_TB
Re-launch
1.00us
317.083 ns
Name
Value
out[1:0]
shift_left_1
shift_right_1
shift_left_2
shift_right_2
clk
next
up_button
rst
down_button
00
0
0
0
0
1
1
0
1
0
0 ns
100 ns
200 ns
250 ns
300 ns
350 ns
XX
00
01
00
10
00
11
00
01
00
X1: 317.083 ns
Default.wcfg
Console
Out: 00, shift_left_1: 0, shift_right_1: 0, shift_left_2: 0, shift_right_2: 0
ISim>
Sim Time: 1,000,000 ps
After resetting at the start, we tested the up and down buttons to validate that they are not affecting the output of the off state, nor the next blinking states.
After pressing the next button to move to the next state, our out moved from 00 (OFF) to 01 (ON), and we can see that the up and down buttons are not affecting the shift_right of shift_left signals, as is expected from the ON state.
The next button was hit again and we moved to the OFF state. Note that the up and down buttons do not affect the shift_right or shift_left signals.
We have now moved into the first blinking state (10). The up and down buttons, when pressed, send shift_right and shift_left signals, respectively.
The next button was hit again and we moved to the OFF state. Note that the up and down buttons do not affect the shift_right or shift_left signals.
This is the second blinking state (11). Up and down buttons trigger shift signals and change the speed of the blinking.
Finally, we press next one more time to go back to the initial state (OFF - 00). We then move to ON one more time, just to test the reset button and assert that the system goes back to state 00 (OFF) when reset = 1.


ISim (P.20131013) - [Default.wcfg]
File Edit View Simulation Window Layout Help
PROGRAMMABLE_BLINKER
Re-launch
1.00us
Instances and Processes
Instance and Process Name
Design Unit
programmable_blinker_tb
programmabl..
glbl
glbl
Objects
Simulation Objects for program...
Object Name
Value
fast_out 0
slow_out 0
clk 0
reset 0
shift_left 0
shift_right 0
count_en 0
Name
Value
fast_out 0
slow_out 1
clk 0
reset 0
shift_left 0
shift_right 0
count_en 0
833.333 ns
200 ns
400 ns
600 ns
Here we are simulating two programmable blinker modules. At first, both start off with a period of 1second (32 count_en cycles).
Then, after a single right shift, the fast blinker (fast_out) doubles in speed witha period of half a second (16 count_en cycles).
X1: 833.333 ns
Instances and Proces...
Memory
Source F...
Default.wcfg
Console
Slow output is 1
Fast output is 0
Slow output is 0
Fast output is 0
Slow output is 0
Fast output is 0
Slow output is 0
ISim>
Console
Compilation Log
Breakpoints
Find in Files Results
Search Results
Sim Time: 1,000,000 ps
9:17 AM
2/5/2019

SHIFTER


ISim (P.20131013) - [Default.wcfg]
File Edit View Simulation Window Layout Help
Re-launch
Name
Value
fast_out[6:0]
0001000
slow_out[6:0]
1000000
clk
1
shift_left
1
shift_right
0
reset
0
135.213 ns
105 ns
110 ns
115 ns
120 ns
125 ns
130 ns
135 ns
140 ns
0001000
0010000
0100000
1000000
In this case, shift_left is high for the entire time and you can see that slow_out is shifted once to the left on every rising edge because it has access to bits [6:3] (signifies 1 to 8 seconds) while fast_out is held constant because it only has access to [3:0] (signifies 1/8th to 1 seconds.
In the final case you can see that the slow_out couldn't be shifter to the left anymore so it is also held constant
we tested ~15 shifting scenarios including the case where both buttons are pressed at the same time.
X1: 135.213 ns
Default.wcfg
Console
Fast shifter output is 0001000
Slow shifter output is 1000000
Fast shifter output is 0001000
Slow shifter output is 1000000
Fast shifter output is 0001000
Slow shifter output is 0001000
Fast shifter output is 0001000
ISim>
Console
Compilation Log
Breakpoints
Find in Files Results
Search Results
Sim Time: 1,000,000 ps
3:28 PM
2/3/2019


TIMER
ISim (P.20131013) - [Default.wcfg]
File Edit View Simulation Window Layout Help
Re-launch
1.00us
Instances a...
Instance and Process N
timer_tb
glbl
Objects
Simulation Objects for timer_tb
Object Name
Value
out 0
count_en 0
clk 0
load_value[6:0] 0000001
reset 0
Name
Value
out 1
count_e 1
clk 1
load_va 0000001
reset 0
185.000 ns
50 ns
100 ns
150 ns
250 ns
300 ns
350 ns
The timer should have its output high every 4 times that the beat_32 module output (count_en) goes high. That happens because we padded our next_value and value on the timer module with 2 0s on the LSB. Instead of counting 1 count_en, now we count 4. We padded the 0s to help make the timer a single module for both the fast and slow blinking.
X1: 185.000 ns
Default.wcfg