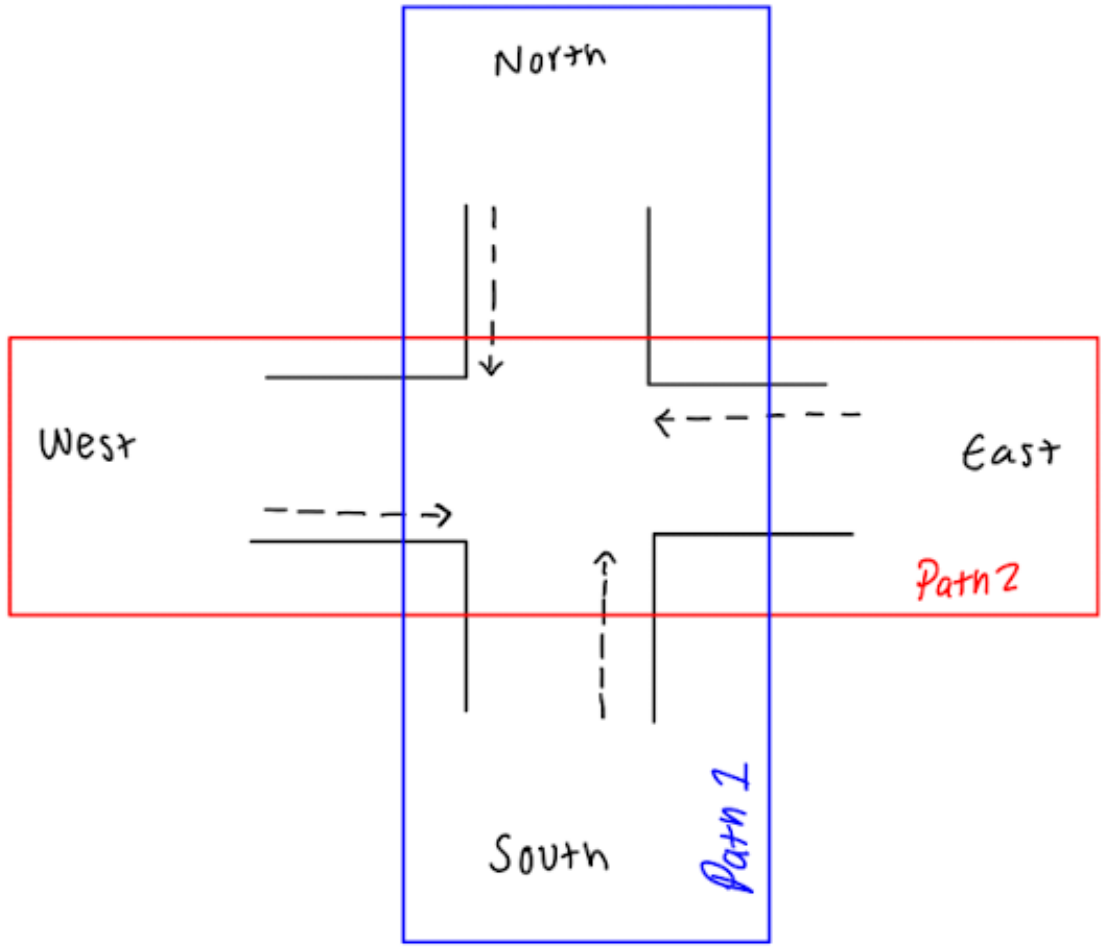
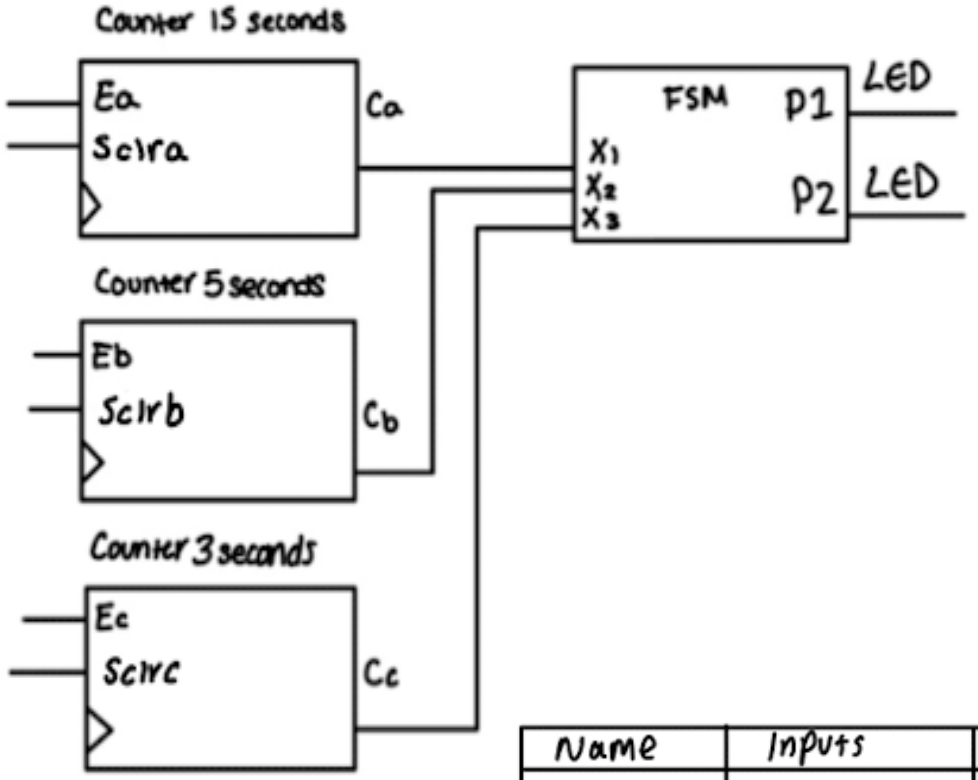
4 WAY TRAFFIC LIGHT CONTROLLER

Angelo Esho, Jason Mikho, and Ethan Talampas



Procedure

- We designed a 4-way traffic light controller to simulate a real-life scenario of how a traffic light would work at an intersection.
- We divided the routes into two paths Path 1 (North and South) and Path 2 (East and West).
- We constructed different times for each path to turn green, yellow, or red.
- For our hardware we used an Artix-7 50T board, breadboard, and wiring utensils.
- For software we used VHDL code in Vivado.



Name	Inputs	outputs
counter 1	Ea	C_1
	Sclra	
counter 2	Eb	C_2
	Sclrb	
counter 3	Ec	C_3
	Sclrc	
FSM	X1, X2, X3	P1, P2 LEDS

Circuit Design

- We used 3 counters and one FSM machine for our circuit.

The FSM will represent the system with a set number of states. In our circuit and code, we will have 6 states (S0-S5). These states will control the LEDs.

INPUT and OUTPUTS of both parts

- The 3 counters will be required to count 15 seconds, 5 seconds, and 3 seconds.
- The counters contain 6 inputs (Ea, Eb, Ec, Sclra, Sclrb, and Sclrc)
- The counters contain 3 outputs (Ca, Cb, and Cc) that run into the FSM machine.
- The FSM machine has 3 inputs (X1, X2, and X3) that come in from the counters.
- The FSM contains only 2 outputs (P1 and P2) that will produce the LED lights onto our breadboard.

Path 1	Green	Yellow
Path 2	Red	
Time seconds	15	5

Path 2	Green	Yellow
Path 1	Red	
Time seconds	15	5

Path 1	Red
Path 2	Red
Time seconds	3

Timing

- Path 1 will be green for 15 seconds and then change to yellow for 5 seconds while path 2 will be red for 20 seconds.
- This will also occur, but it will be for Path 2.
- The next scenario both paths will be red for 3 seconds. This will allow the switch between the paths to occur.

Input Ca, Cb, & Cc			Present state	Next state	output (P1 & P2)	
1	x	x	000	001	010	100
x	1	x	001	010	110	100
x	x	1	010	011	100	100
1	x	x	011	100	100	010
x	1	x	100	101	100	110
x	x	1	101	000	100	100

Input Ca, Cb, & Cc			Present state	Next state	output (P1 & P2)	
1	x	x	S0	S1	010	100
x	1	x	S1	S2	110	100
x	x	1	S2	S3	100	100
1	x	x	S3	S4	100	010
x	1	x	S4	S5	100	110
x	x	1	S5	S0	100	100

Excitation and State Tables

- Three inputs are used to control which states will be active. These inputs are (Ca, Cb, and Cc). However only one input is needed at one time.
- The inputs 1 will control if the states move on.
- When the input Ca, Cb, or Cc is 1 it will make the state change .

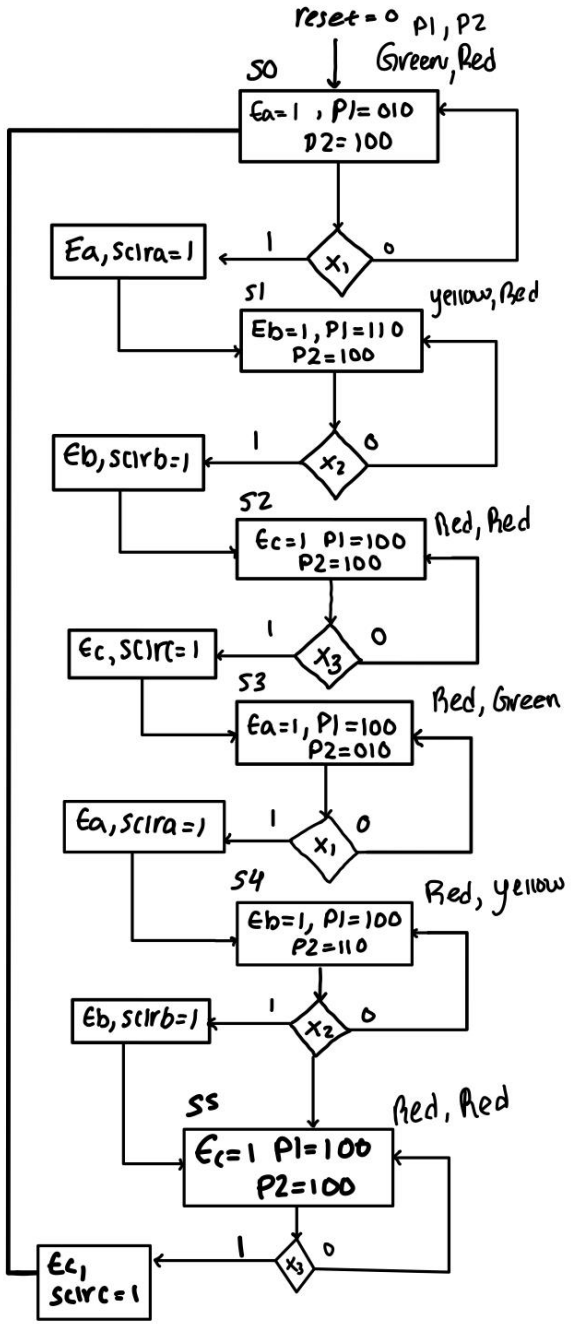
Number meanings:

010 = green, 110 = yellow, 100 = red

States

S0 = 000, S1 = 001, S2 = 010, S3 = 011, S4 = 011,

S5 = 101



State Explanations

S0

- When X1 = 0 it will remain in S0, When X1 = 1 the state changes to S1 and the counter will run for 15 seconds.

S1

- When X2 = 0 it stays in S1, X2 = 1 it will change to S2 and the counter will run for 5 seconds.

S2

- When X3= 0 it stays in S2, X2 = 1 it will change to S3 and the counter will run for 3 seconds.

S3

- When X1 = 0 it will remain in S3, When X1 = 1 the state changes to S4 and the counter will run for 15 seconds.

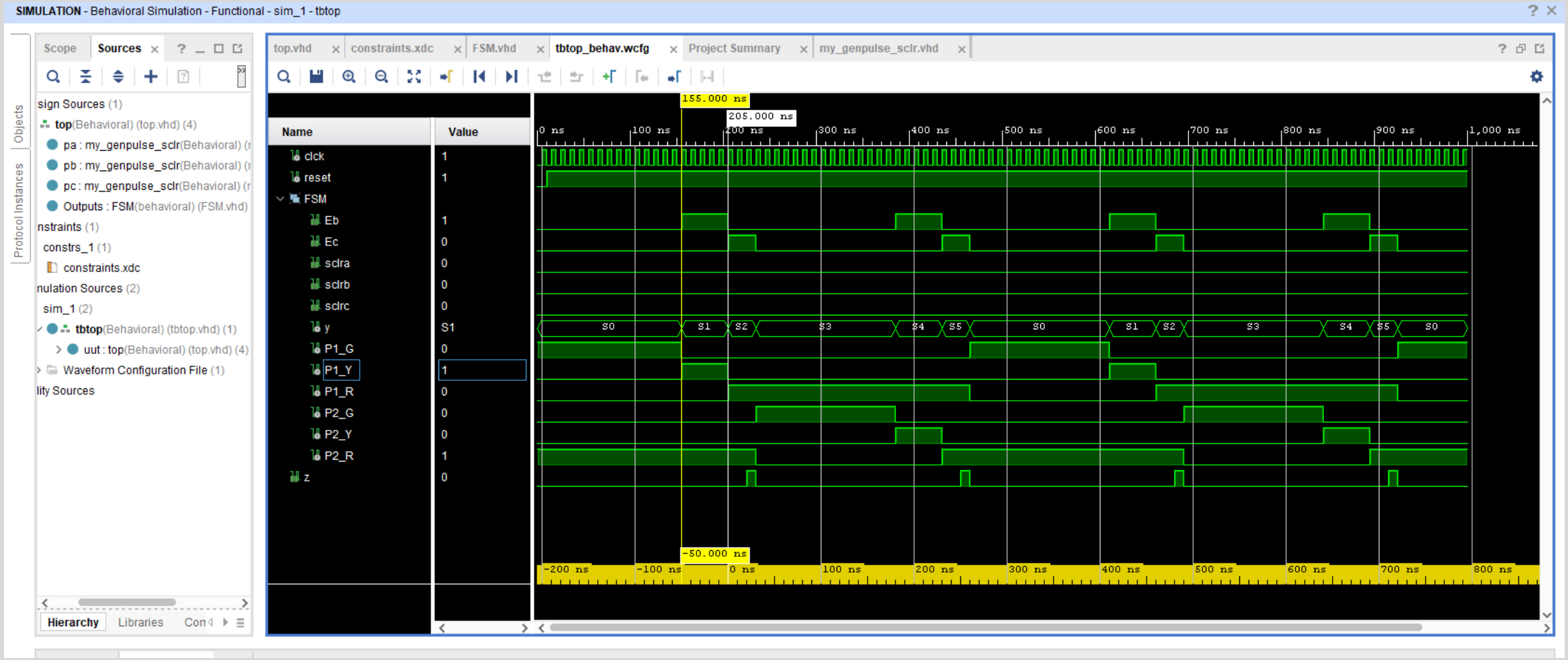
S4

- When X2 = 0 it stays in S4, X2 = 1 it will change to S5 and the counter will run for 5 seconds.

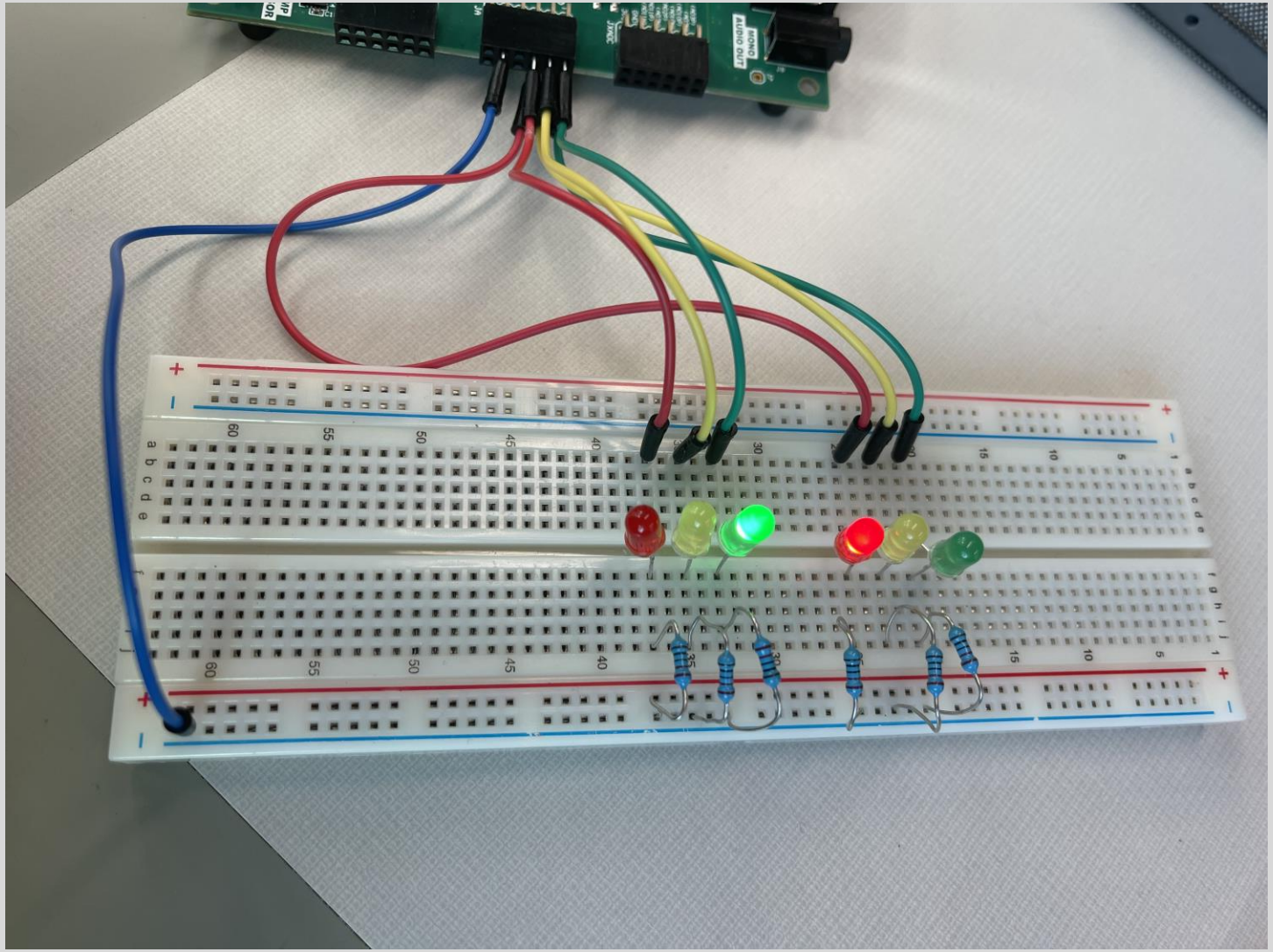
S5

- When X3 = 0 it will remain in S5, X3 = 1 it will change back to S0 restarting the cycle.

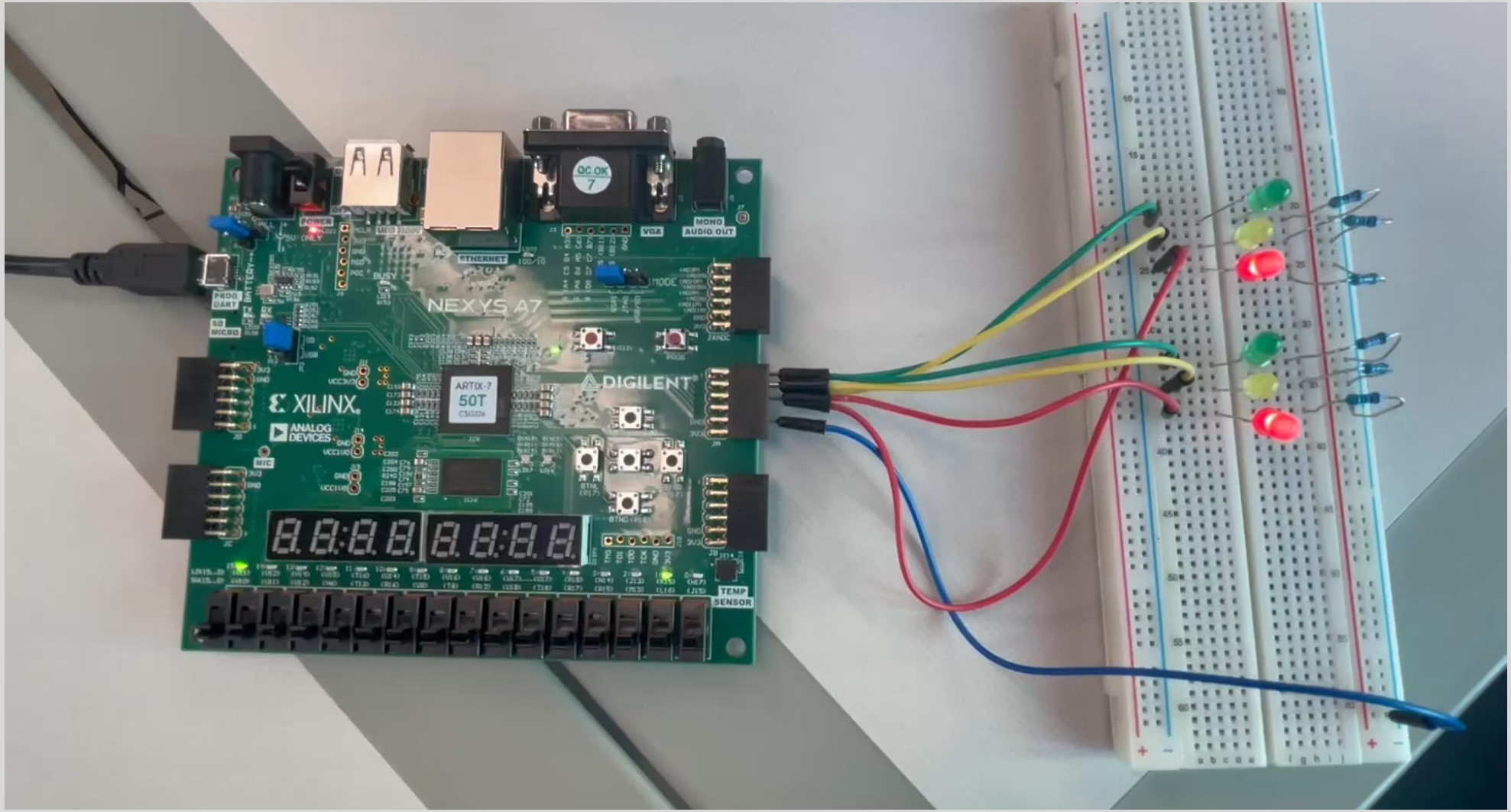
Simulation


SIMULATION - Behavioral Simulation - Functional - sim_1 - tbtop
Scope
Sources
top.vhd
constraints.xdc
FSM.vhd
tbtop_behav.wcfg
Project Summary
my_genpulse_sclr.vhd
Objects
Protocol Instances
sign Sources (1)
top(Behavioral) (top.vhd) (4)
pa : my_genpulse_sclr(Behavioral)
pb : my_genpulse_sclr(Behavioral)
pc : my_genpulse_sclr(Behavioral)
Outputs : FSM(behavioral) (FSM.vhd)
nstraints (1)
constrs_1 (1)
constraints.xdc
nulation Sources (2)
sim_1 (2)
tbtop(Behavioral) (tbtop.vhd) (1)
uut : top(Behavioral) (top.vhd) (4)
Waveform Configuration File (1)
lity Sources
Hierarchy
Libraries
Name
Value
clck
reset
FSM
Eb
Ec
sclra
sclrb
sclrc
y
P1_G
P1_Y
P1_R
P2_G
P2_Y
P2_R
z
155.000 ns
205.000 ns
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
800 ns
900 ns
1,000 ns
S0
S1
S2
S3
S4
S5
-50.000 ns
-200 ns
-100 ns

Breadboard Design Structure



Traffic System on Breadboard



References

1) Llamoca. ”Generic Pulse Generator.” VHDL Coding For FPGAS, http://www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html