ECE 4250 Lab 8

Traffic Light Controller State Machine

Sterling LaBarbera

14096504

# Objective

In this lab, we designed a state machine that accepted user inputs to activate the walk signal after a certain delay. We used the clock from Lab 7 for the state changing process.

# Lab Work

Using provided code to control the FPGA lighting, we implemented a traffic light controller that was green for at least 32 seconds, changed to yellow for 4 seconds after 32 seconds if the button was pressed, then red for 20 seconds. A light for walk/no walk would come on when the car light was red. During the last 4 seconds of red, the walk light flashed to indicate the light would change soon, and it changed to no walk when the traffic light was green or yellow. If the button was not pressed, the light stayed green until it was pressed. Figure 1 shows simulation results with all lights, clocks, and states.

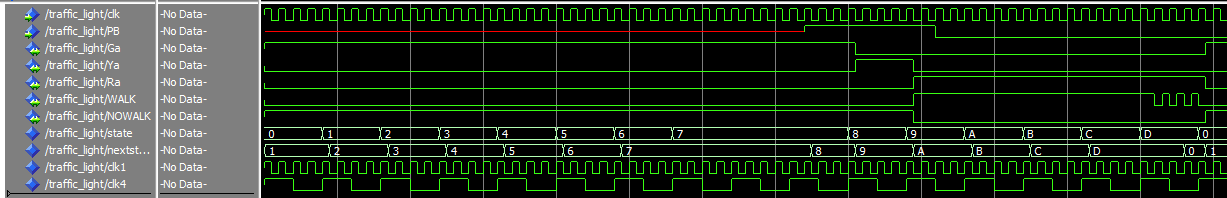


Figure . Simulation results for Lab 8.

We used the generic clock from lab 7, shown in Figure 5, to control the state lengths. Each state was 4 seconds long, with a total of 14 states. As seen in Figure 2 , States 0 – 6 were similar, just green light and no walk light on. State 8 had the same lights, but started checking the button press, to determine if it needed to go to state 8. State 8 had the yellow light and no walk light on. State 9 – 12 had the red light and walk light on, and state 14 flashed the walk light every second while keeping the red light on.

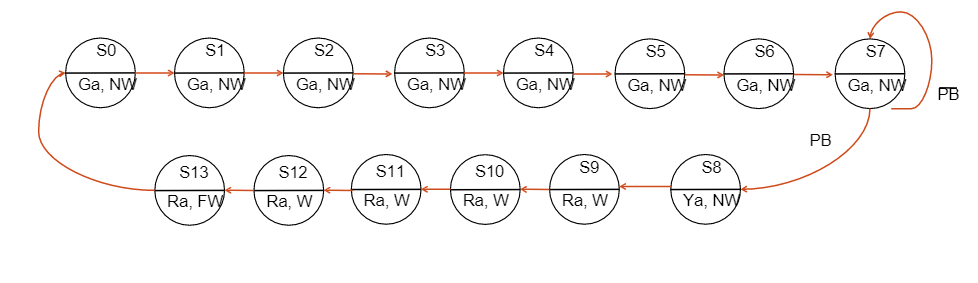


Figure . State diagram for Lab 8.

Using the GenClock, I made changed states based on the 4 second output and used the one second output to flash the walk light in state 13.

# Conclusions

Since the state diagram and supporting code were provided, this lab was straightforward. We only had to implement the traffic light file using the entity definition given to make it work with everything. Once the ModelSim simulation was working, I had one error in Vivado, but it provided a line shown in Figure 3 to add to the constraint file that would fix functionality. On the FPGA, the lights worked, except that the state 13 blinking happened on the 7-segment display instead of the walk light. It is possible this had to do with the change I made for constraints.



Figure . Added line in .XDC file.

Figure 4 shows the block diagram for the full lab. The Lab7Bench block contains the top level. It has components AnodeControl and LEDDisplay for the LED functions, and traffic\_light as defined above. The AnodeControl I am uncertain of its functioning. The LEDDisplay contains a component dec\_7seg that translates a 4-bit number to a 7-bit code for the 7-segment display. LEDDisplay controls what is shown on 7-segment displays on the FPGA. It also appears to run the LEDs below the 7-segment display that represented the different lights.

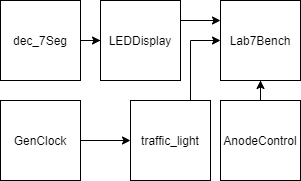


Figure . Lab 8 block diagram.

FPGA results are shown in Figure 8, Figure 9, and Figure 10. The small green LEDs from right to left are green, yellow, red, walk, and no walk.

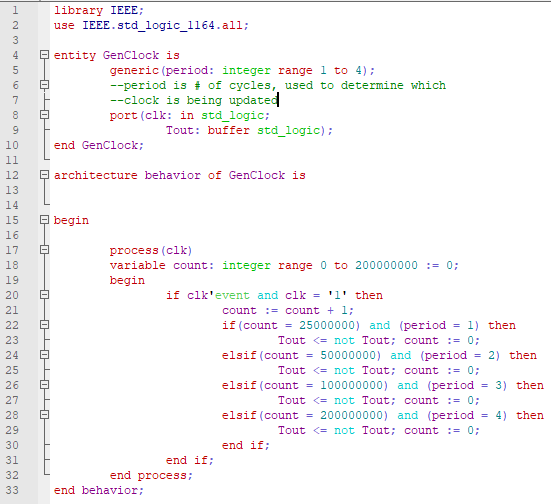


Figure 5. GenClock code.

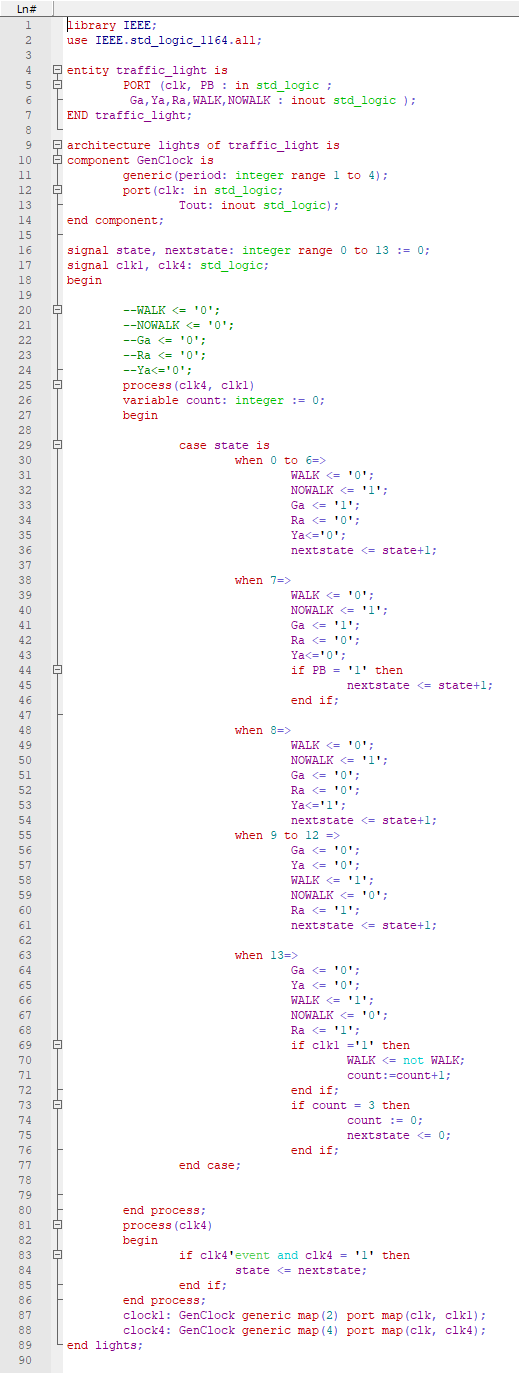


Figure . Traffic light code.

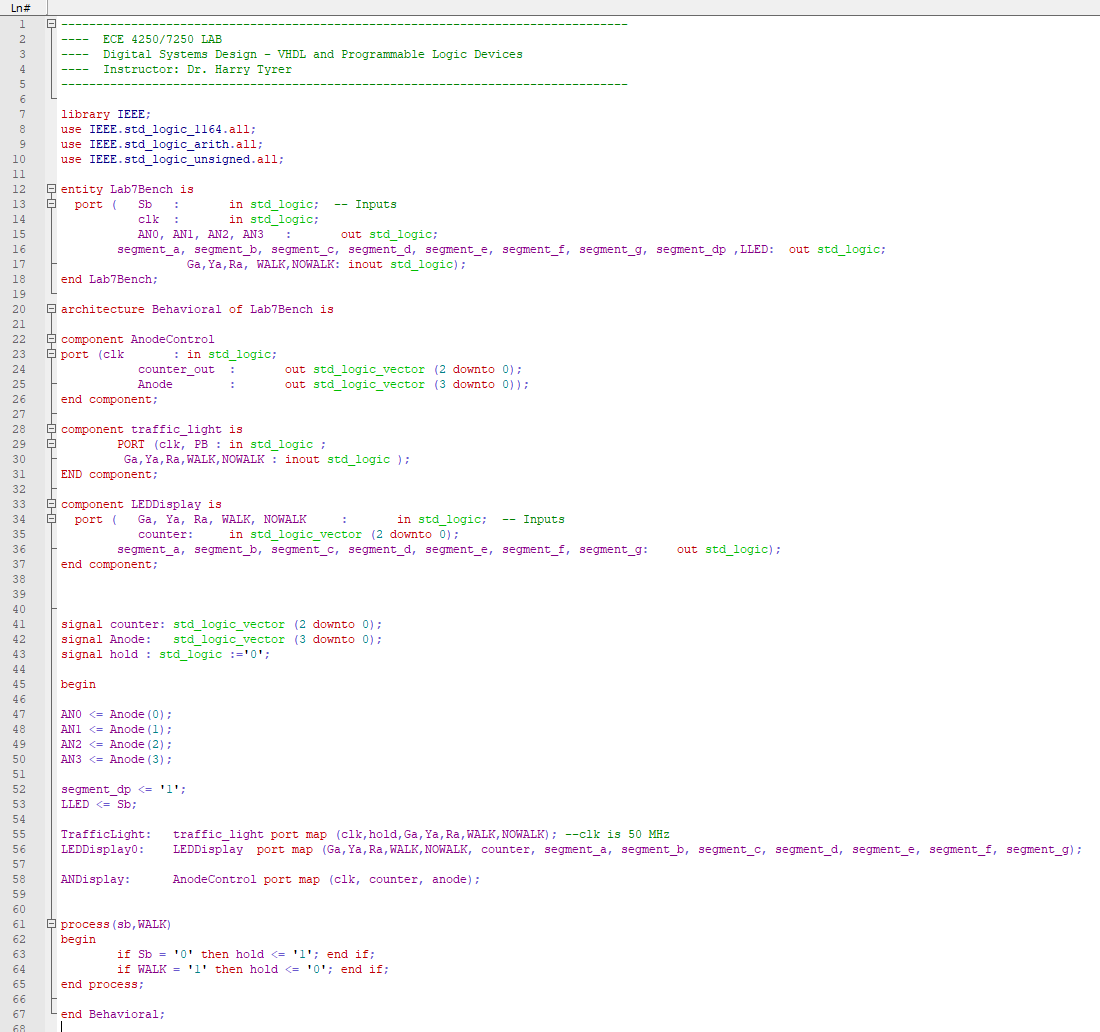


Figure . Lab 8 bench code.

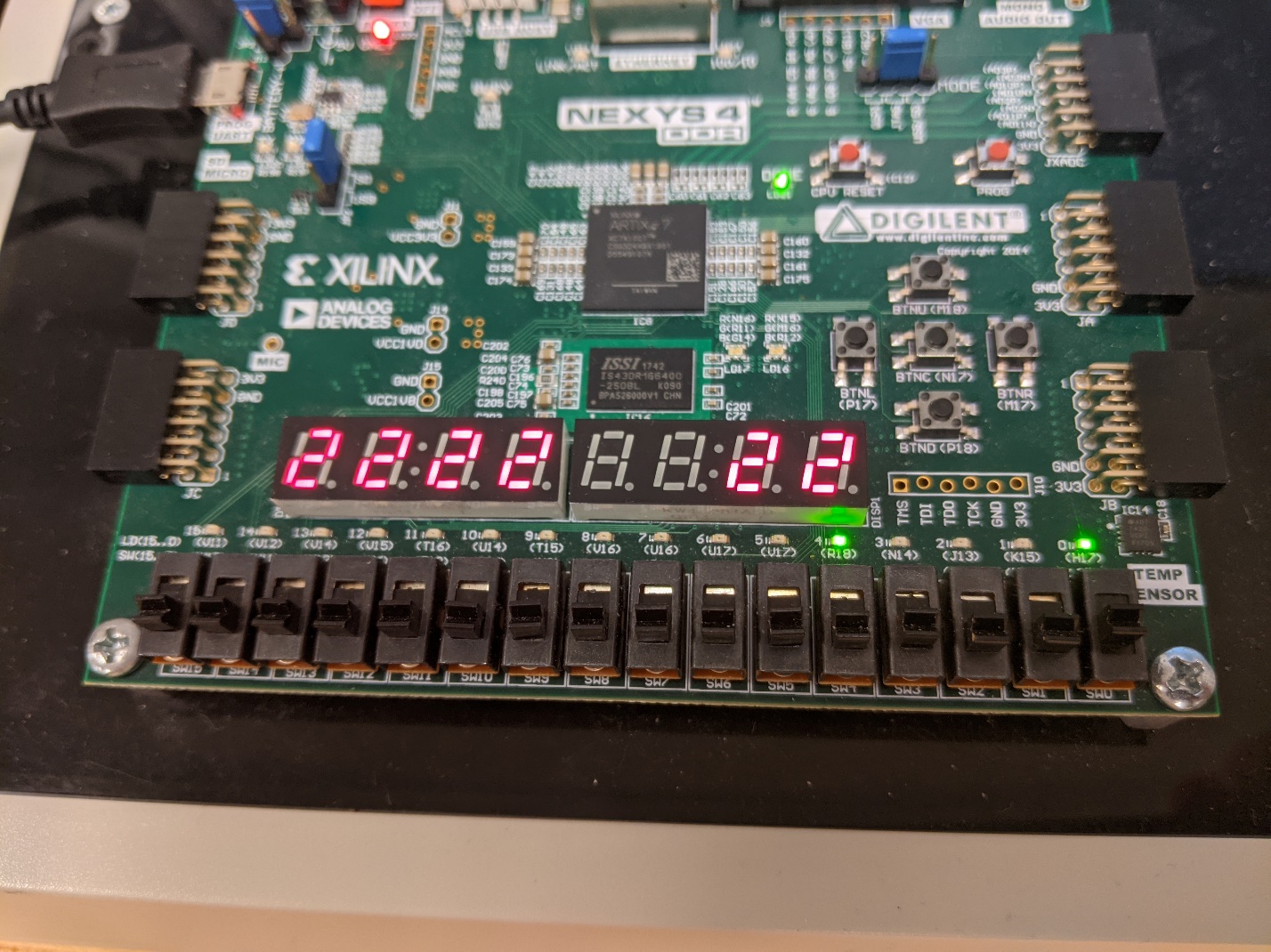


Figure . State 0 with no walk and green light.

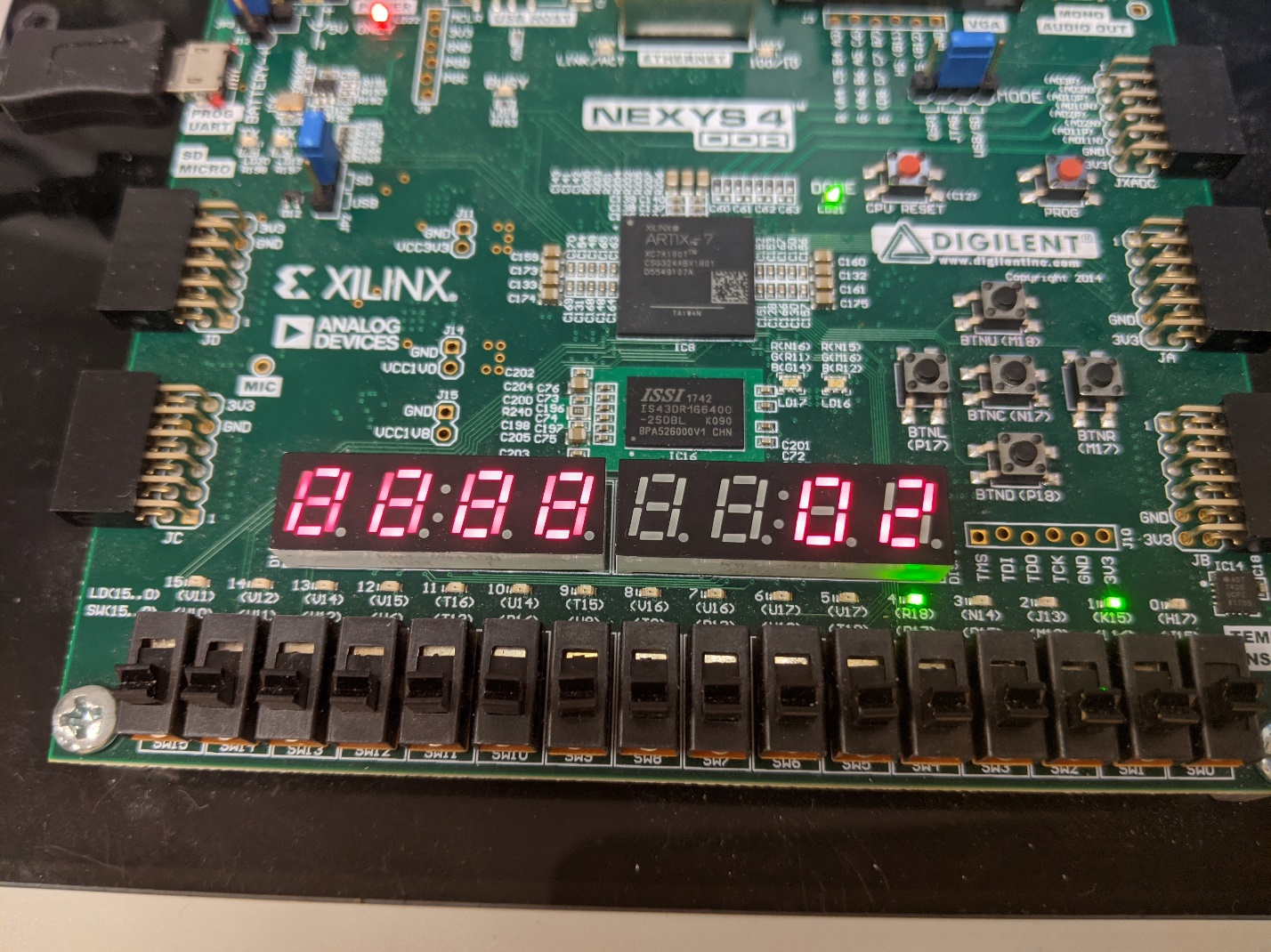


Figure . State 8 with no walk and yellow light.

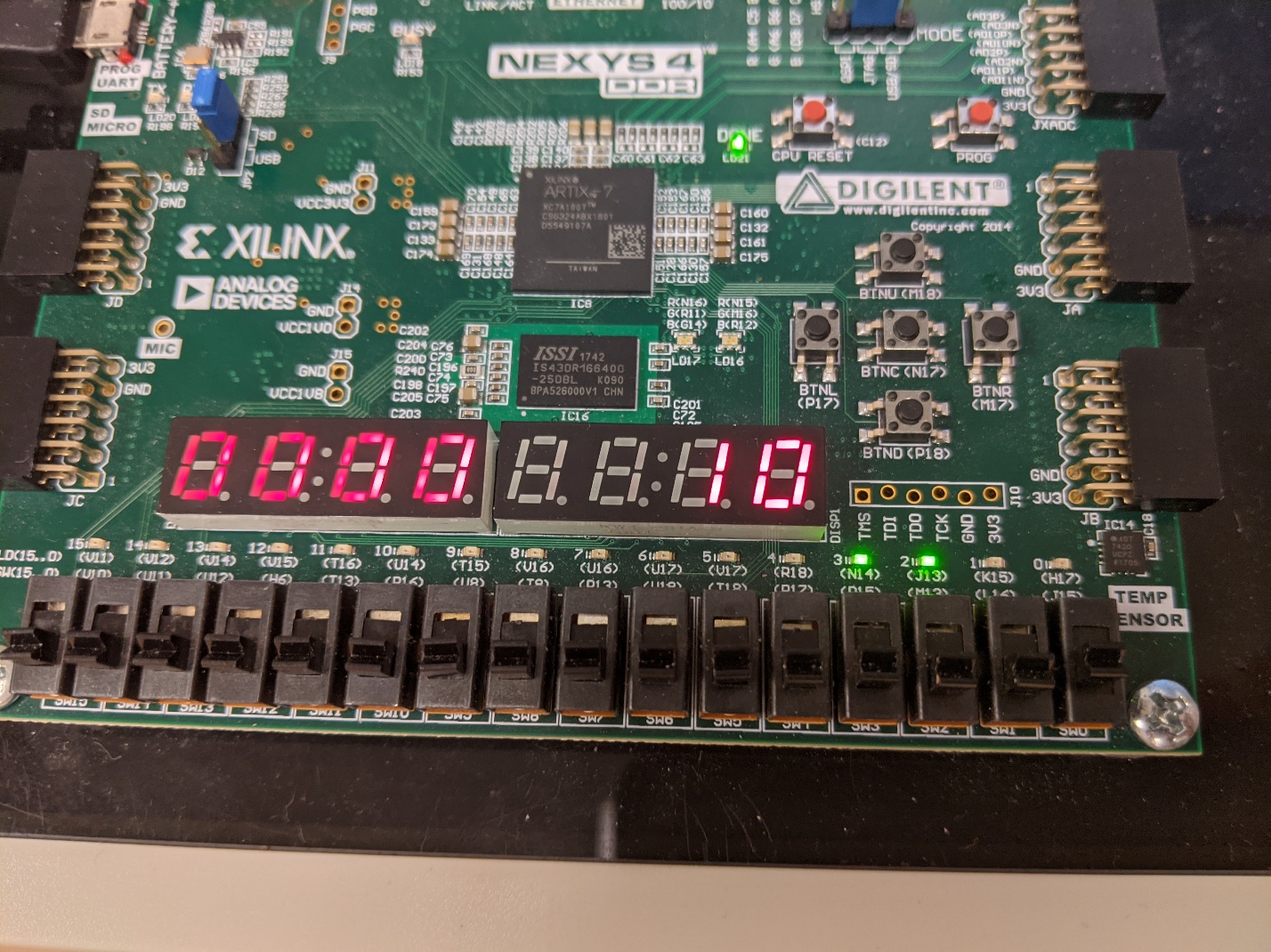


Figure . State 9 with walk and red light.