Ada Zağyapan

EEE102-02

9 April 2025

Lab-7: Finite State Machine

PURPOSE

The purpose of this lab is to design a finite state machine (FSM) on the breadboard by using D flip-flops and logic gates.

METHODOLOGY

This design consists of two inputs, two states and one output. To design this finite state machine, a state diagram is firstly constructed. Then, the number of flip flops that should be used is determined. After that, a state table is created. Using this state table, the function of next state Q^+ is determined as a function f(x_0, x_1, Q). After the Boolean equation of the next state is determined, the finite state machine is designed on the breadboard using D flip-flops, an AND gate, an OR gate and a hex inverter. On the Basys3 board, the inputs x_1 ("Treat") and x_0 ("Scold") were wired to the on-board switches SW2 and SW1, respectively, and the clock signal was provided by the center push-button BTN_C. All I/O lines were routed through the JA Pmod header to the breadboard.

DESIGN SPECIFICATIONS

This finite state machine works as a "tail-wag controller" for a robotic puppy. x_1 is a "Treat" push button which you press to give a treat to the puppy. x_0 is a "Scold" push button which you

press to scold the puppy. Q is the tail-wag current state, which is either 1 (wagging) or 0 (resting). When the the user presses Treat ($x_1 = 1$), regardless of whether the tail was wagging or not, Q^+ becomes 1. And on the next clock, Q becomes 1 and the tail starts wagging, which makes the green LED turn on. After Treat is released, x_1 returns to 0, but because Scold is not pressed ($x_0 = 0$), the term $\overline{x_0}q$ remains 1, hence Q^+ becomes 1 again. On every clock the machine stays in $Q = 1$, so the tail keeps wagging and the LED stays lit even though the user has let go of the Treat button as long as the Scold button isn't pressed. When the user presses Scold ($x_0 = 1$), $\overline{x_0}$ goes to 0, hence $\overline{x_0}q = 0$. With $x_1 = 0$, $Q^+ = 0$, so on the next clock, $Q = 0$, the tail stops wagging and the red LED turns on. If neither button is pressed while the tail is resting, $Q = 0$ again, so nothing changes and the red LED stays on. When both Treat and Scold buttons are pressed at the same time, Treat has priority over Scold. Hence, if the tail was resting, it transitions to wagging, and if it was wagging it stays in wagging.

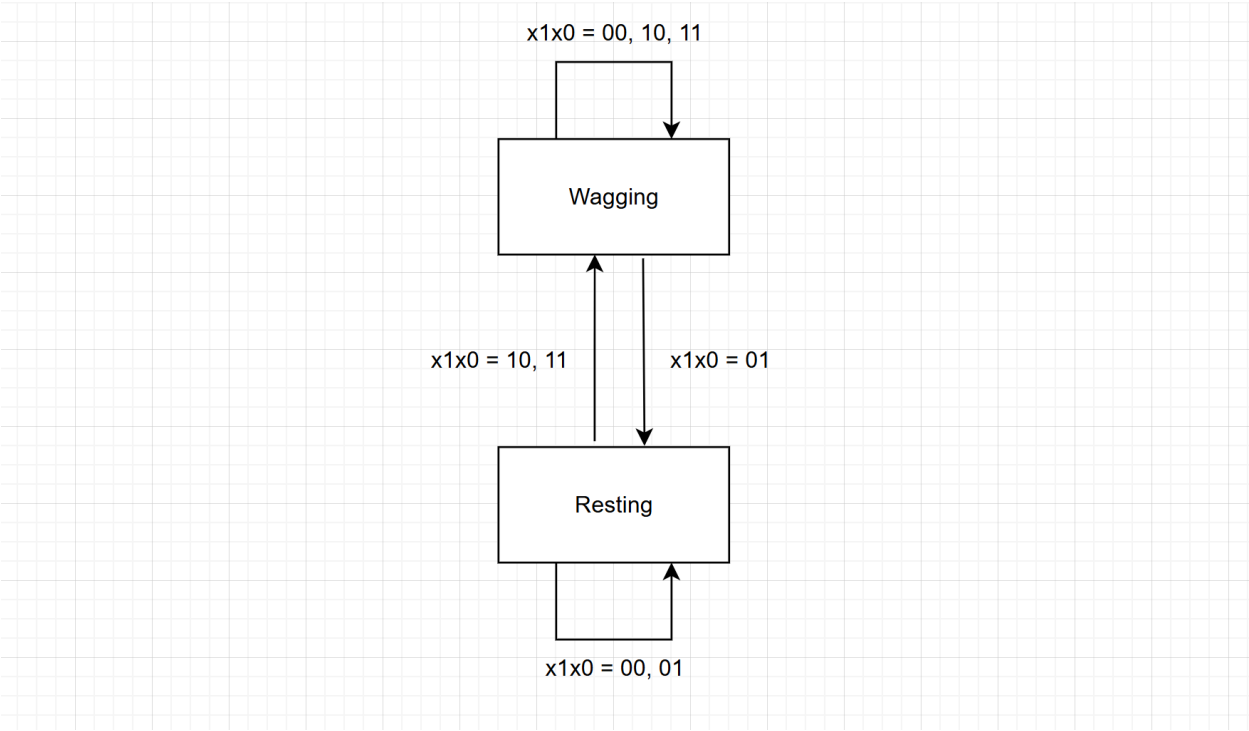

Figure 1: State diagram of the finite state machine

x_1	x_0	Q	Q^+	Green LED (Q^+)	Red LED ($\overline{Q^+}$)
0	0	0	0	0	1
0	0	1	1	1	0
0	1	0	0	0	1
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	1	0

Figure 2: State table of the finite state machine

Using Karnaugh Maps, the Boolean equation for the output of the flip-flop Q^+ is determined as $Q^+ = f(x_0, x_1, Q) = x_1 + (\overline{x_0} \cdot Q)$.

SN74HC08 AND gate: This IC has four independent two-input AND gates; each gate outputs a logic high only when both of its inputs are high.

SN74HC04 inverter: This IC has six inverters (NOT gates). Each one takes a single input and outputs its logical complement.

SN74HC32 OR gate: This IC has four separate two-input OR gates; each outputs high if either or both inputs are high.

SN74HC74 D flip-flop: This IC samples its D input on each rising clock edge and holds that value on Q until the next clock. It also includes asynchronous set and reset pins to initialize or clear the stored state.

RESULTS

After the Boolean equation was determined, the FSM was designed on the breadboard and tested using different x_0 and x_1 values. The observed outputs were consistent with the intended results seen on the state table. The outputs can be observed in Figures 3-6.

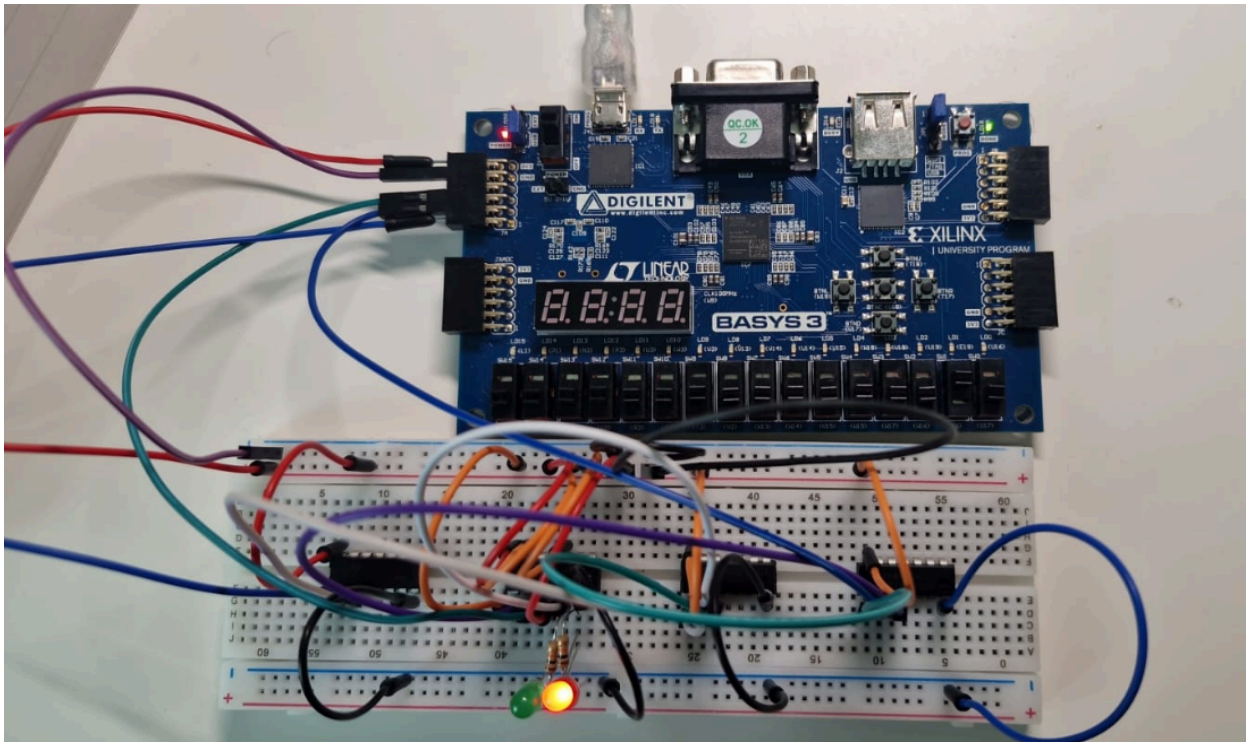

Figure 3: Q=0, x_1=0, x_0, Q^+=0

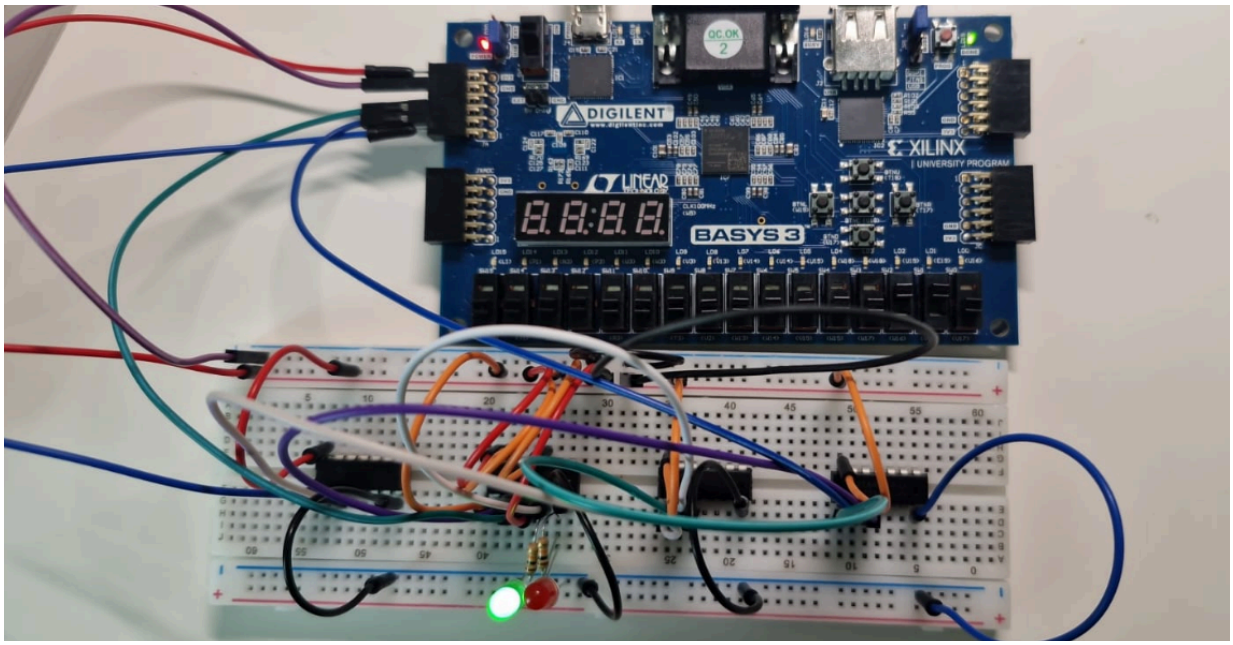

Figure 4: Q=0, x_1=1, x_0=1, Q^+=1

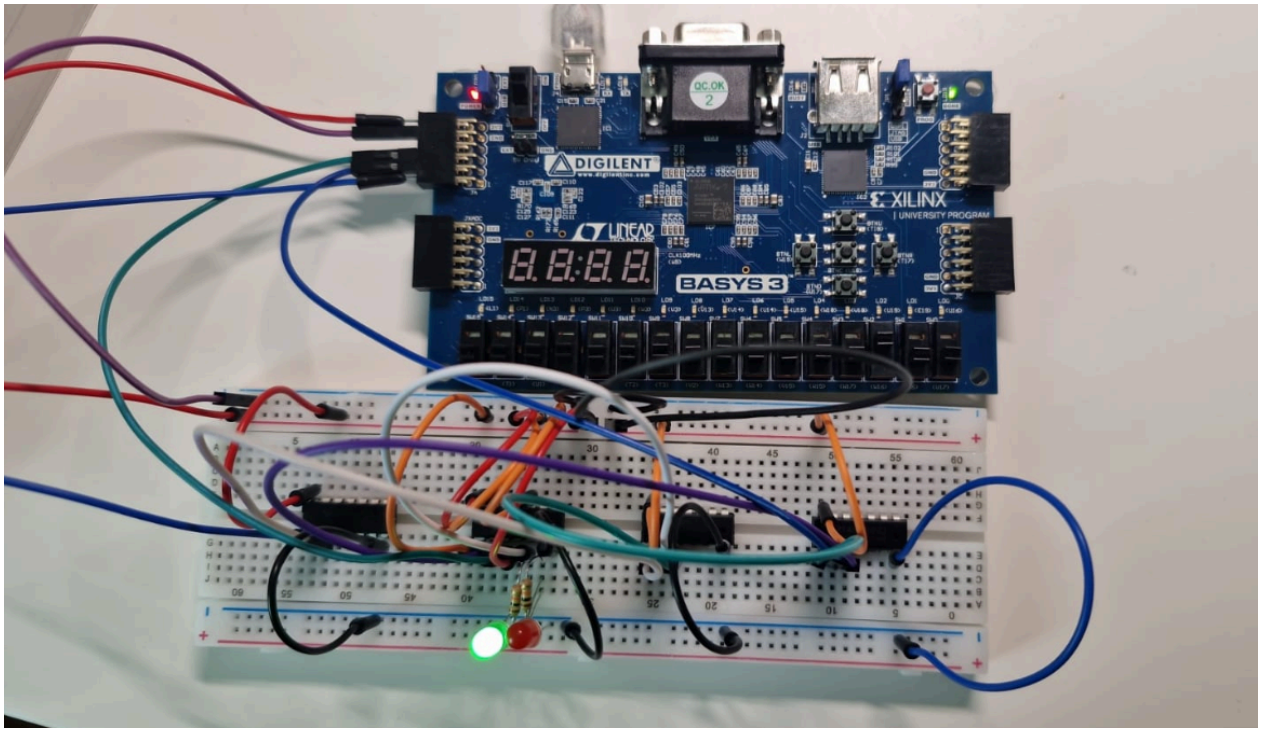

Figure 5: Q=1, x_1=1, x_0=0, Q^+=1

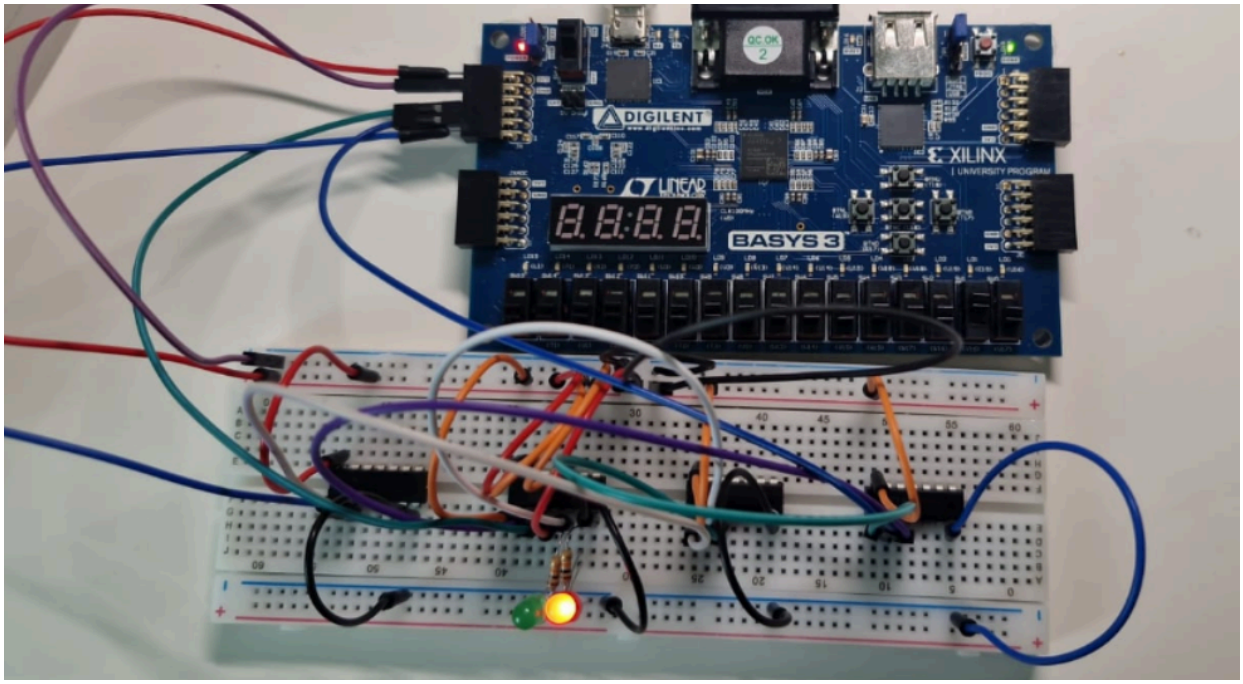

Figure 6: Q=1, x_1=0, x_0=1, Q^+=0

CONCLUSION

In this lab, a two-input, two-state Moore finite state machine was successfully designed, analyzed, and implemented on a breadboard to control a "tail-wag" indicator for a robotic puppy. The circuit was assembled using SN74HC08 AND, SN74HC04 inverter, SN74HC32 OR, and SN74HC74 D-flip-flop ICs. The observed LED outputs matched the state table in every case.

APPENDIX

pins.vhd:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;



entity pins is
    port(
        JA   : out std_logic_vector(2 downto 0);
        sw   : in  std_logic_vector(1 downto 0);
        btnC : in  std_logic
        );
end pins;

architecture Behavioral of pins is

begin

JA(1 downto 0) <= sw(1 downto 0);
JA(2)          <= btnC;

end Behavioral;
```

cons.xdc:

```
# Switches
set_property PACKAGE_PIN V16 [get_ports {sw[0]}]

     set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]}]
set_property PACKAGE_PIN W16 [get_ports {sw[1]}]

     set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]}]

#Buttons
set_property PACKAGE_PIN U18 [get_ports btnC]

     set_property IOSTANDARD LVCMOS33 [get_ports btnC]


#Pmod Header JA
#Sch name = JA1
set_property PACKAGE_PIN J1 [get_ports {JA[0]}]

     set_property IOSTANDARD LVCMOS33 [get_ports {JA[0]}]
#Sch name = JA2
set_property PACKAGE_PIN L2 [get_ports {JA[1]}]

     set_property IOSTANDARD LVCMOS33 [get_ports {JA[1]}]
#Sch name = JA3
set_property PACKAGE_PIN J2 [get_ports {JA[2]}]

     set_property IOSTANDARD LVCMOS33 [get_ports {JA[2]}]
```