April 30th 2025

Alp Efe Kılıçarslan, 22402390

EEE 102-2

LAB-07: Finite State Machine

Purpose:

The purpose of this lab assignment is to design and implement a finite state machine using integrated circuits and jumper cables. A state table and a state diagram will be created for the designed machine and documented.

Design Specifications:

The finite state machine that was decided on was a “Day/Night Cycle Parity Checker”. For each clock, the design will take a 2-bit length logic input and have a 1-bit output. The machine’s states, inputs and outputs were decided as:

-state: ABNORMAL = “00”, “01”, “10”

-state: NORMAL = “11”

-input: NIGHT_NIGHT = “00”

-input: NIGHT_DAY = “01”

-input: DAY_NIGHT = “10”

-input: DAY_DAY = “11”

-output: INCORRECT = “0”

-output: CORRECT = “1”

The intended function of this design was to check if two consecutive days had DAY_NIGHT’s (“10”) and if this was observed, it would output a logic high (“1”). For every other case, the output would be a logic low (“0”). Each bit of a specific state would indicate the expected behavior of said state, a “1” indicating standard behavior and “0” implying abnormal measurements. These bits would be assigned in accordance to the previous state and current inputs. Using these definitions, a state table (Table 1) and a state diagram (Figure A) was created to show the mechanism of the finite state machine.

Previous State	Input	Next State	Output
00	00	00	0
00	01	00	0
00	10	01	0
00	11	00	0
01	00	10	0
01	01	10	0
01	10	11	1
01	11	10	0
11	00	10	0
11	01	10	0
11	10	11	1
11	11	10	0
10	00	00	0
10	01	00	0
10	10	01	0
10	11	00	0

Table 1: State Table of the Day/Night Parity Checker showing each possible combination of the design

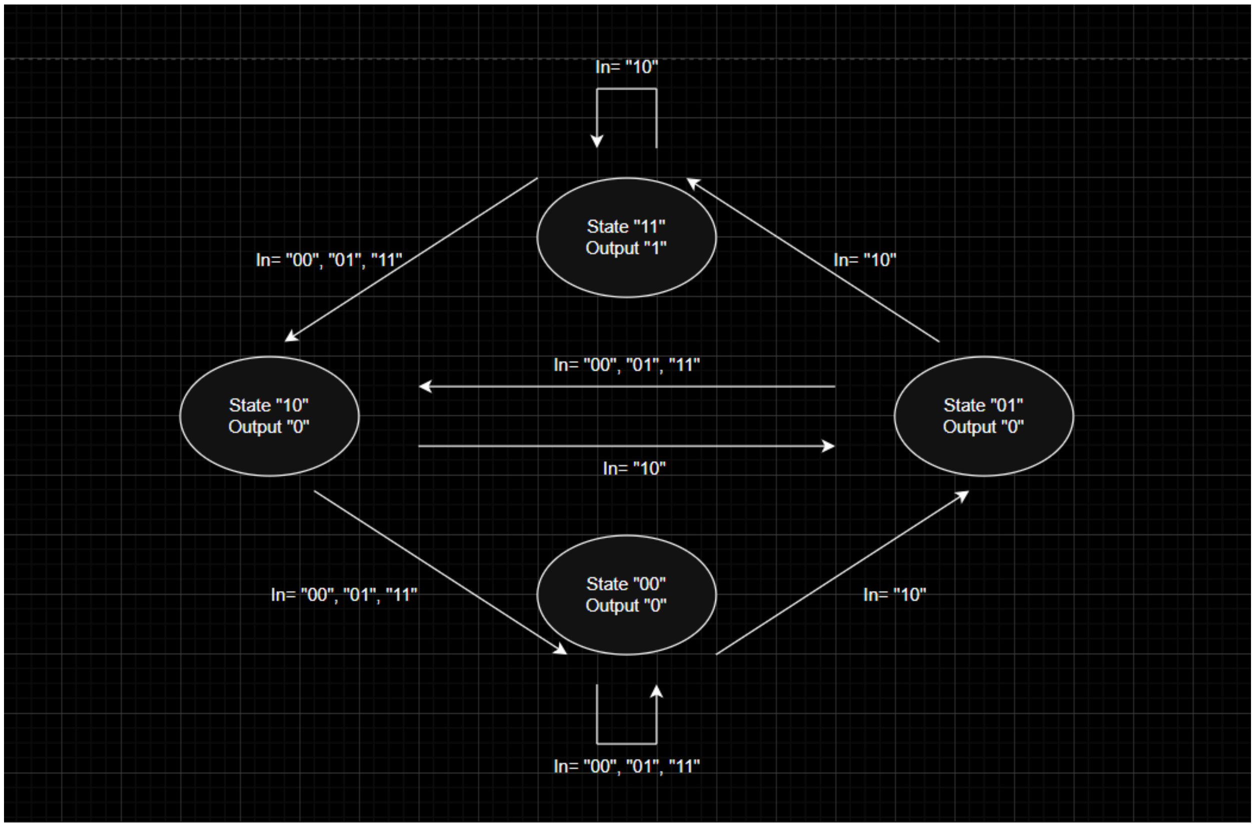

Figure A: State Diagram of the Day/Night Parity Checker

After analyzing these visual descriptions, the logic of the finite state machine was formed. The circuit would take the rightmost bit of the input and invert it. It would take the inverted bit and the leftmost input bit and AND them. The output of this operation would be fed to a D Flip-Flop. The output of the initial

Flip-Flop would be fed to another D Flip-Flop. The outputs of both Flip-Flops would be taken and put through an AND gate. The output of this operation would be finally fed to another D Flip-Flop. The last Flip-Flop's output would be connected to an LED and the compliment of the output would also be connected to an LED. A logic high would be connected to the PRE and CLR inputs of each Flip-Flop to ensure changes with inputs. A clock would be connected to each Flip-Flop to make a synchronous Moore-type Finite State Machine. Using this design, a logic diagram was drawn (Figure B).

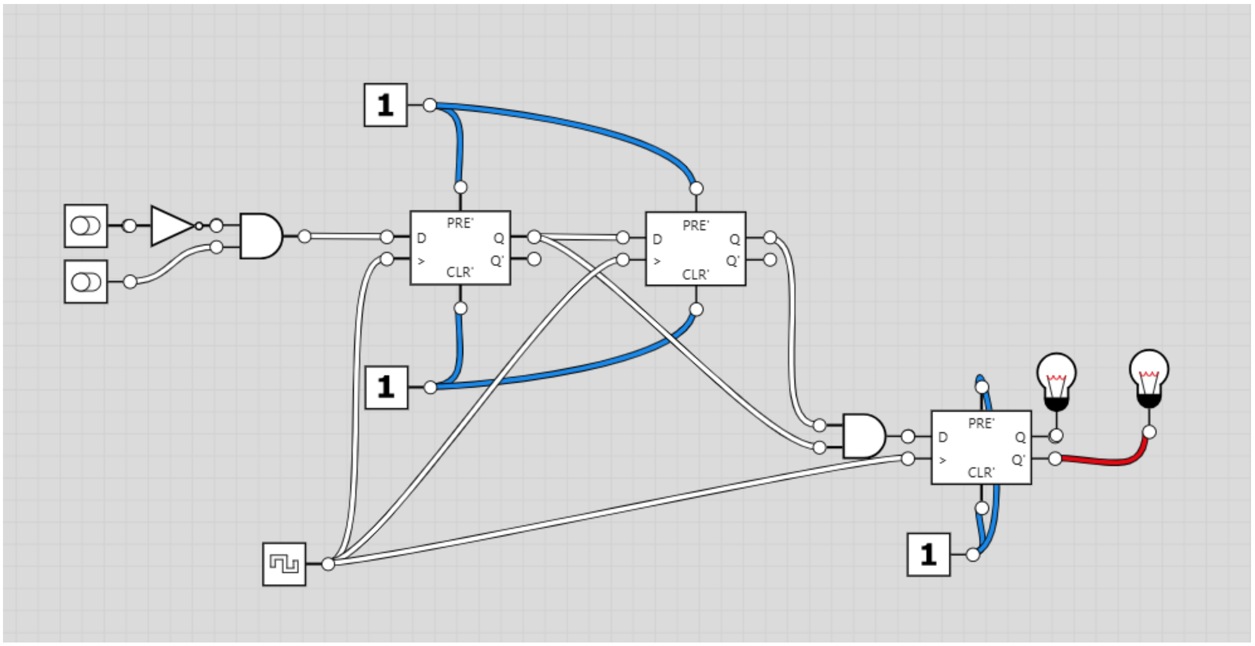

Figure B: Logic Diagram of the Day/Night Parity Checker

To implement this design in real-life, two 74 LS/HC 74's (Dual DFF), one 74 LS/HC 04 (Hex Inverter) and one 74 LS/HC 08 (Quad 2-input AND) were used. In order to give the circuit the required clock and inputs, the BASYS 3 FPGA was used. The two rightmost switches and the center button were assigned as the input and the clock signal respectively. These signals were outputted to the JA PMOD Port of the BASYS 3 as required. The required power and ground for the system were also supplied using the specific PMOD Ports of the FPGA. Using all of these parameters, the circuit schematic was formed.

Methodology:

Task-1) Using the specified design, the finite state machine was implemented on a breadboard using the integrated circuits, jumper cables and LEDs. The BASYS 3 was connected accordingly and the system was initialized (Figure 1.1). A red LED was used to show when system output was "0" and a green led was use for cases when the output was "1". Different inputs for specific states were tested to see if the circuit's real-world output matched the state table and diagram. The outputs for certain states were recorded.

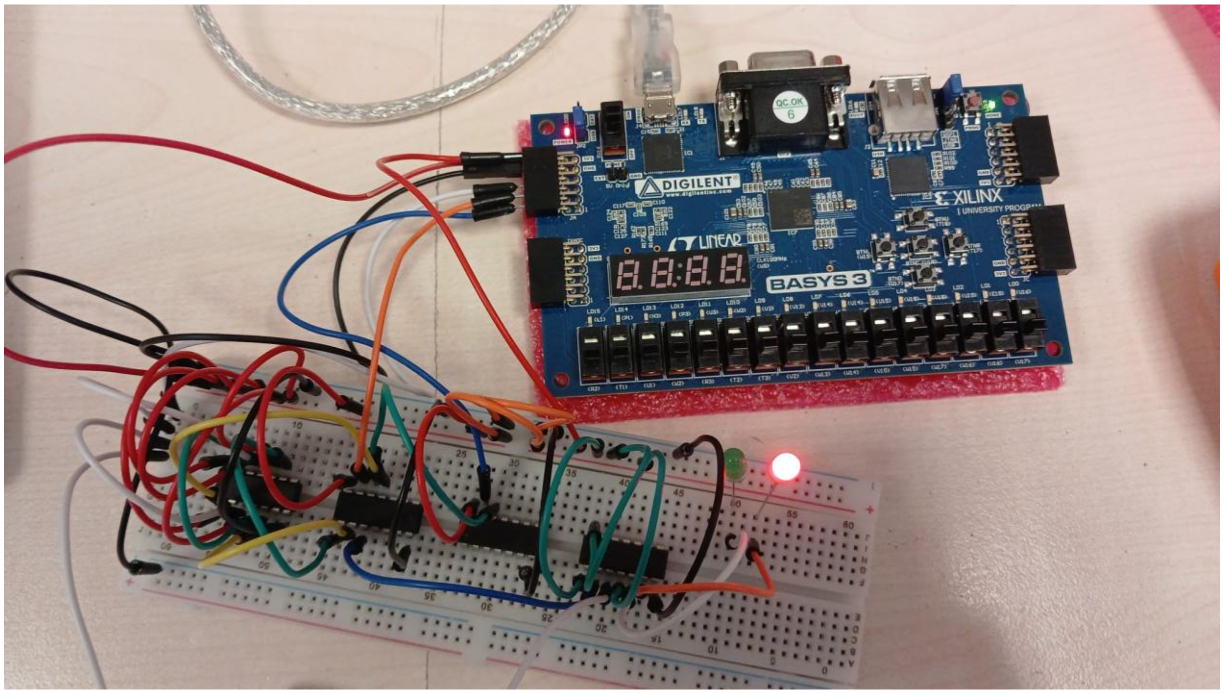

Figure 1.1: Default State of the Day/Night Parity Checker when no inputs are given. For the BASYS 3, reds are Vcc, black is ground, white is clock, orange is the inputs leftmost bit and blue is the rightmost bit of the input. All other connections are as described in the "Design Specifications" and Figure B.

Results:

Task-1) Different states and inputs were given to the Day/Night Parity checker and the outputs were compared to the state table (Figure 1.2-1.4).

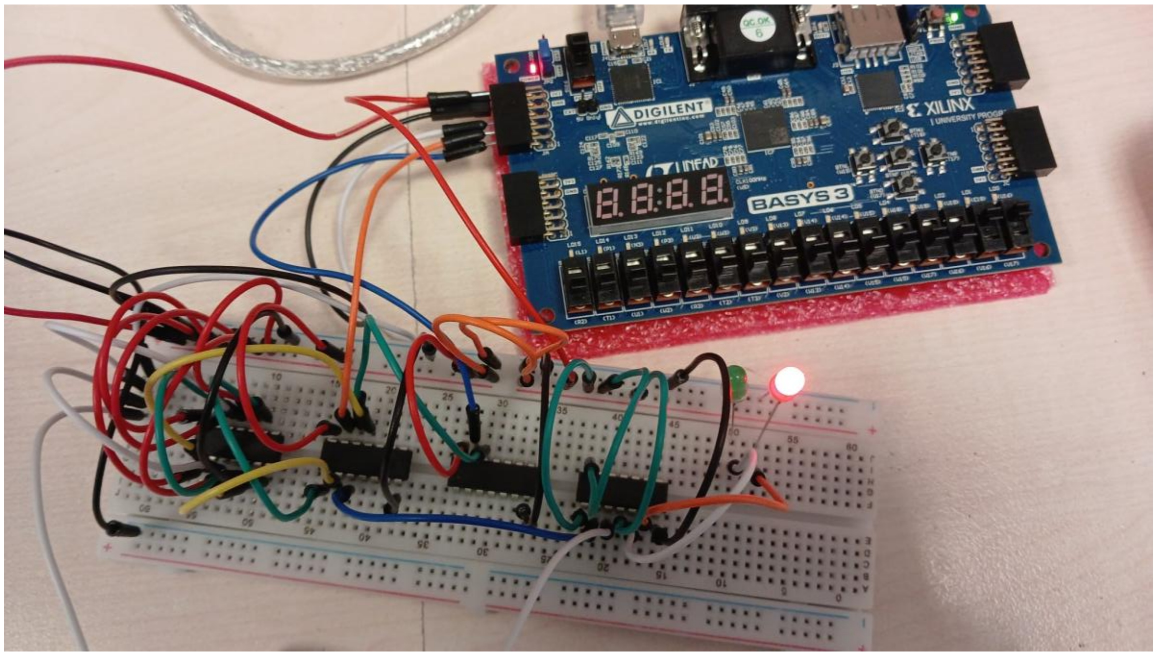

Figure 1.2: FSM for Previous state "00", Input "11", Next State "00", Output "0"

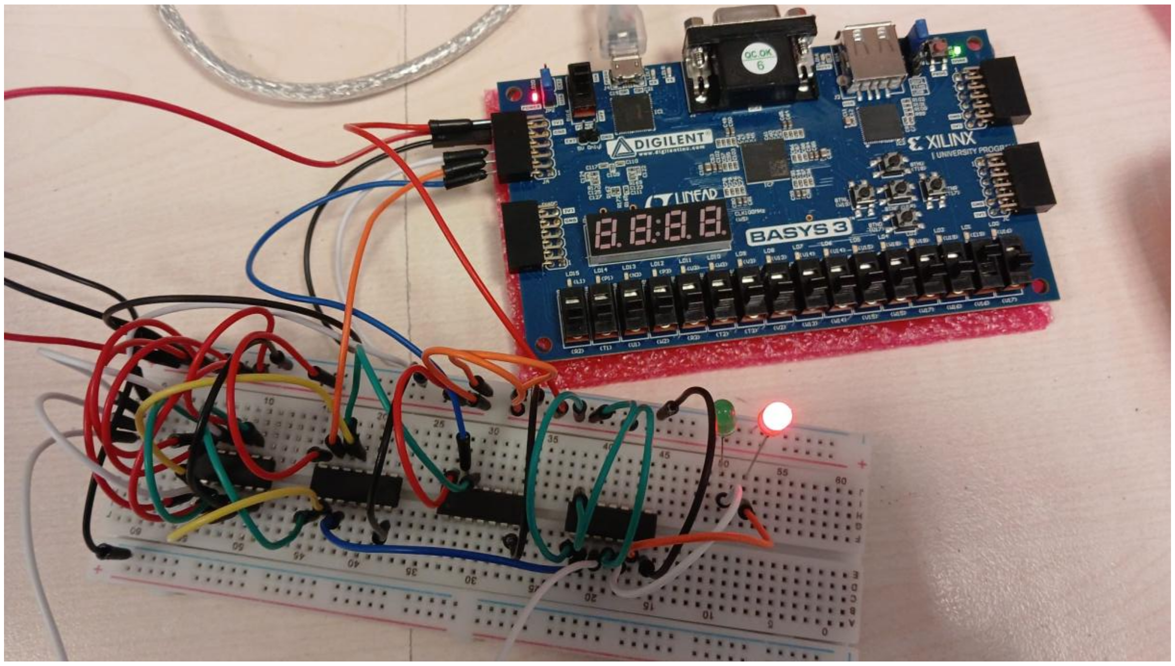
Figure 1.3: FSM for Previous state “00”, Input “10”, Next State “01”, Output “0”

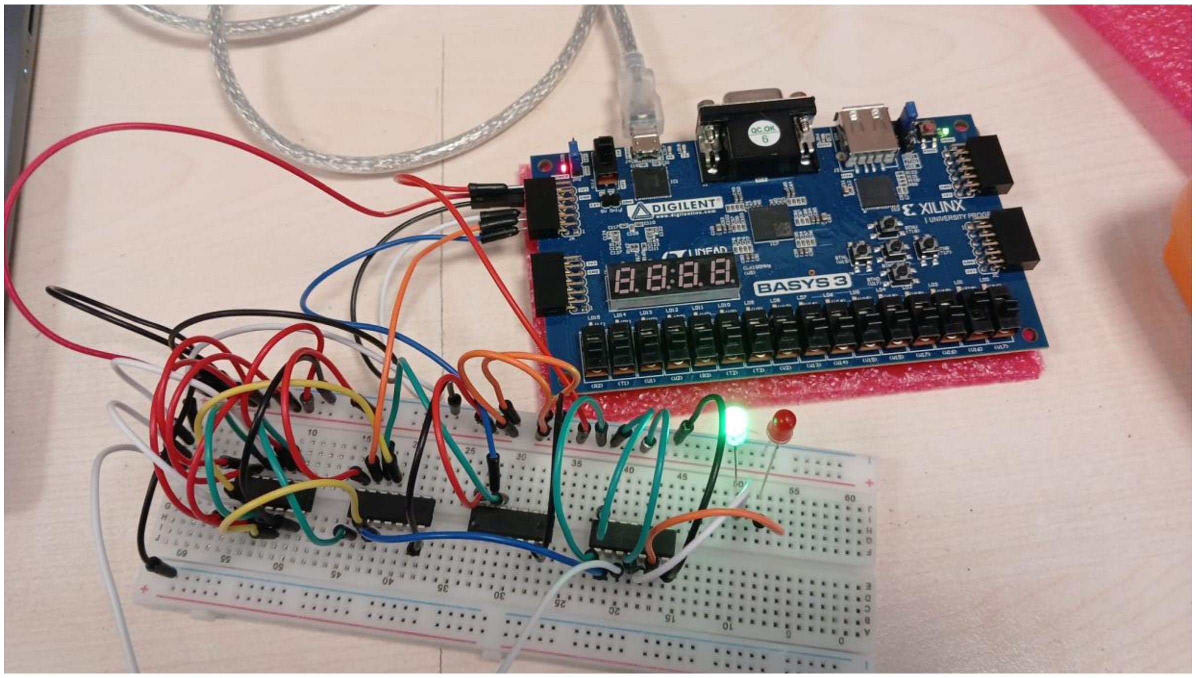
Figure 1.4: FSM for Previous state “01”, Input “10”, Next State “11”, Output “1”

When the results of the images were compared to the state table and diagram, the same behavior was observed for all cases, indicating correct wiring and proper design implementation.

Conclusion:

The purpose of this lab was to design and build a finite state machine. The design was decided upon and it's state diagram was formed. Using this diagram, the logic for the required circuit was made and implemented on a breadboard using integrated circuits, LEDs, jumper cables and the BASYS 3 FPGA. Different states and inputs were tested for the built machine and compared to the state table. No errors were found between the ideal cases and the real-world outcomes. From this assignment, finite state machine design was understood more deeply.

References:

Mano, M. M., & Ciletti, M. D. (2017). *Digital design with an introduction to the Verilog HDL, VHDL, and SystemVerilog* (6th ed.).

Roth, C. H., & Kinney, L. L. (2015). *Fundamentals of logic design* (7th ed.).

Harris, D. M., & Harris, S. L. (2012). *Digital design and computer architecture* (2nd ed.). Morgan Kaufmann.

Appendices:

FSM_Controller.vhd

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



entity FSM_Controller is
  port(
      swin   : in  std_logic_vector(1 downto 0);
      btnC : in  std_logic;
      JA   : out std_logic_vector(2 downto 0)
      );
end FSM_Controller;
```

```
architecture Behavioral of FSM_Controller is

begin

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

end Behavioral;
```

finallabplz.xdc

```
set_property PACKAGE_PIN V17 [get_ports {swin[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {swin[0]}]
set_property PACKAGE_PIN V16 [get_ports {swin[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {swin[1]}]
set_property PACKAGE_PIN U18 [get_ports btnC]
        set_property IOSTANDARD LVCMOS33 [get_ports btnC]
set_property PACKAGE_PIN J1 [get_ports {JA[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {JA[0]}]
set_property PACKAGE_PIN L2 [get_ports {JA[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {JA[1]}]
set_property PACKAGE_PIN J2 [get_ports {JA[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {JA[2]}]
```