Bilkent University EEE102-02 Lab 7 Report

Nizam Ercan 22302317

Purpose:

The purpose of this lab session is to design a finite state machine and implement this design on a breadboard with real logic gates and flip-flops.

Methodology:

First of all, I designed the finite state machine with 3 states, 2 inputs and 1 output and its circuit design with its state diagram in paper. Then I implemented this design to my breadboard with the available logic gates(SN74HC00N NAND gates) and D flip-flops at the lab. Then I tested my circuit with all the possible inputs and states to see whether it works fine or not.

Design Specifications:

The design represents a child being tickled. You can see the meanings of the states, inputs and outputs below:

- State A: The kid not being tickled.
- State B: The kid being tickled and having fun.
- State C: The kid being tickled and not having fun.
- Input: Not tickling the kid in any place(“00”)
- Input: Tickling kid’s left foot(“01”)
- Input: Tickling kid’s right foot(“10”)
- Input: Tickling kid’s both feet(“11”)
- Output: Kid not laughing(“0”)
- Output: Kid laughing(“1”)

The scenario starts with the kid not being tickled. This kid’s right foot is much more sensible to tickling so the only way of making him laugh is tickling his right foot. This transitions us from state A to state B where the kid starts laughing. In this state, if we keep tickling only one foot, he keeps laughing and having a great time and if we stop tickling, we turn back into state A. However, if we tickle both of his feet at any state, we get to state C, where the kid keeps laughing but feels uncomfortable and wants the tickling to stop completely. Therefore, the only way of getting out of this state is to stop the tickling completely. The state diagram for this finite state machine is given in Figure 1 where the most significant bit of the input represents the right foot and the least significant bit of the input represents the left foot.

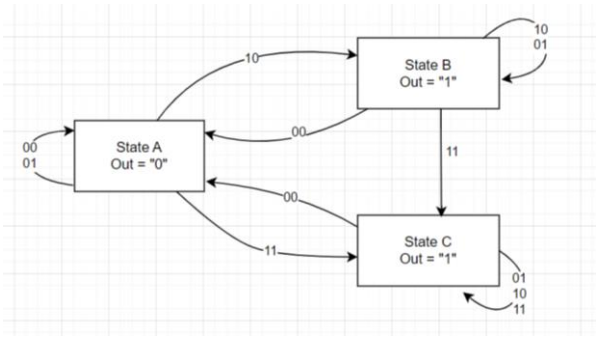

Figure 1: the state diagram

I am representing State A with “00”, state B with “01” and state C with “10”. You can see the state transition tables for both representations in Table 1 and Table 2.

State	RL = “00”	RL = “01”	RL = “10”	RL = “11”
A	A	A	B	C
B	A	B	B	C
C	A	C	C	C

Table 1: State transformation table

State	RL = “00”	RL = “01”	RL = “10”	RL = “11”
“00”	“00”	“00”	“01”	“10”
“01”	“00”	“01”	“01”	“10”
“10”	“00”	“10”	“10”	“10”

Table 2: State transformation table

I can now determine the next state logic from these tables with 2 4-variable Karnaugh maps. Since the state “11” doesn’t exist in this design I can take these values as “don’t cares”. Two Karnaugh maps for the two bits of the next state can be seen in Figure 2 and Figure 3.

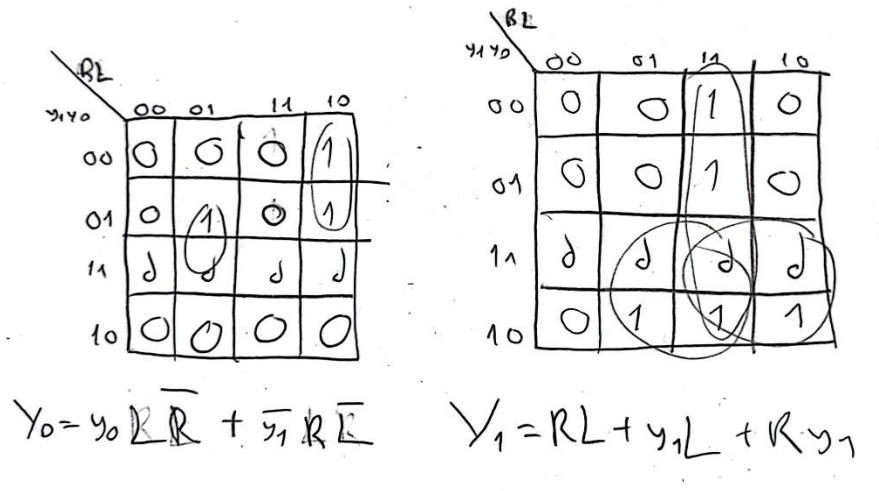

Figure 2-3: Karnaugh maps for the next state

Since my design is a Moore Machine, the output is determined by only the present state variables y1 and y2. Therefore, the output can be determined by a 2-variable Karnaugh map as you can see in Figure 4.

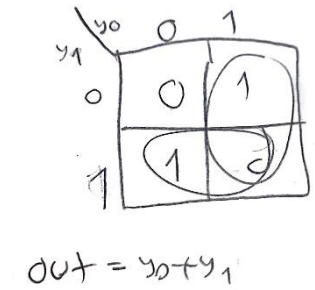

Figure 4: the output Karnaugh map

After obtaining the necessary equations for the next state and the output, I drew the schematics for the whole circuit as you can see in Figure 5 where the inputs are controlled with a switch and the output is shown with an LED.

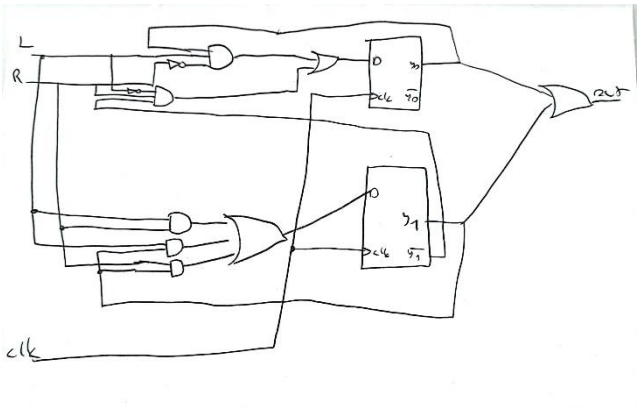

Figure 5: circuit design for the finite state machine

Since I didn't have any access to any "and" or "or" gates or any inverters in the lab, I implemented this design with only nand gates and two D flip-flops as you can see in Figure 6.

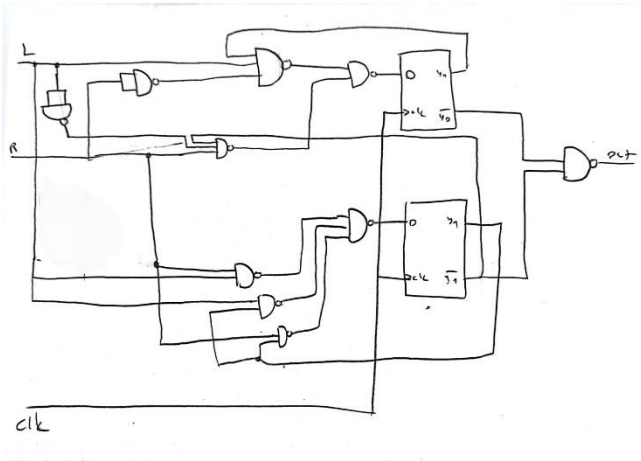

Figure 6: finalized circuit design with nand gates

In order to implement this design on my breadboard, I used a 74HC/LS74 dual D type flip-flop and 3 SN74HC00N type quadruple 2 input Nand gates. I used a signal generator to generate a square wave with a frequency of 1 Hz with amplitude of 5V as the clock input as you can see in Figure 7.

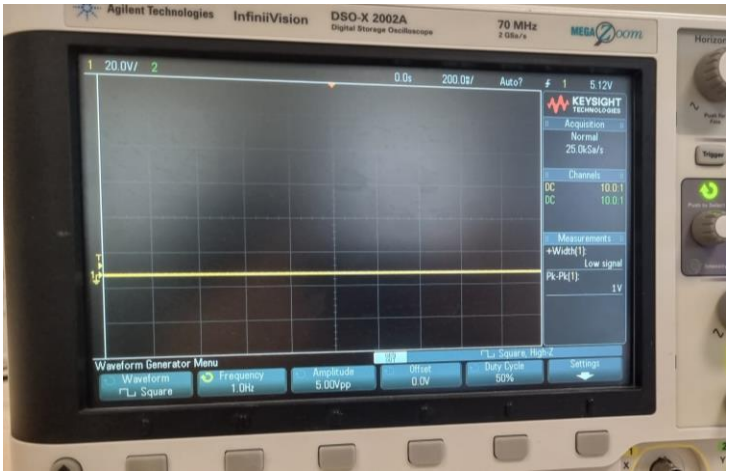

Figure 7: the generated waveform as the clock input

Results:

The implemented design can be seen in Figure 8 where I used a switch for the inputs and an LED for showing the output. I also used pulldown resistors to the ends of the switches. Also, for the power source, I applied a 5V DC current. In addition, I connected every component's ground to the ground of the power supply.

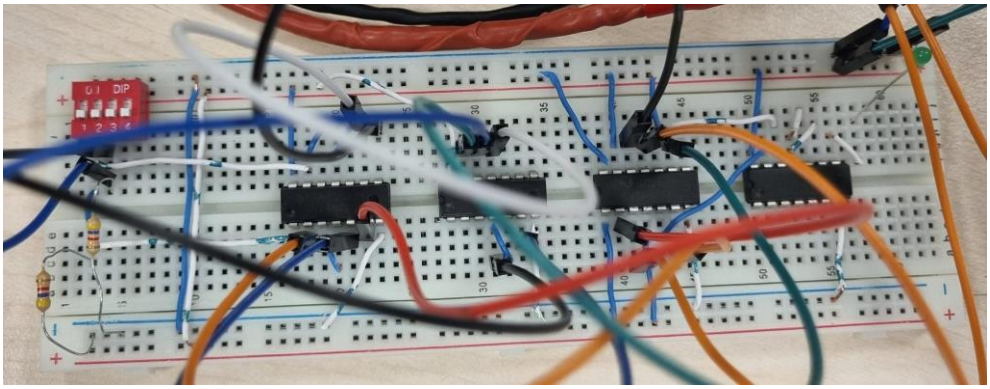

Figure 8: the implemented circuit

After implementing this circuit on my breadboard, I tested it whether it worked correctly or not. I started with the inputs "00" and as you can already see in Figure 8, the LED wasn't giving any light. When I try to turn the second switch on which is the "L" input you can see from Figure 9 that the LED wasn't giving any light as expected.

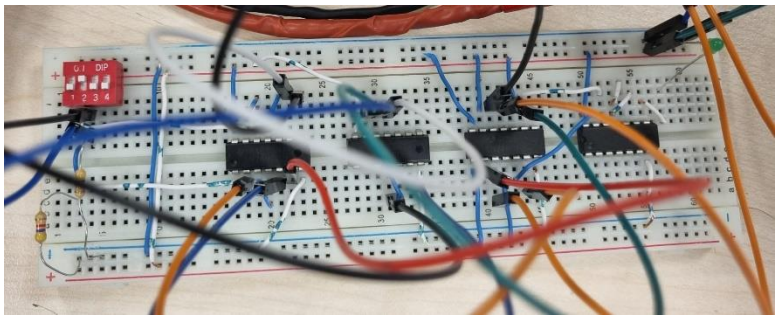

Figure 9: input "01" in State A

However, when I enter “10” as an input the machine transforms into State B and the LED gives light as you can see in Figure 10.

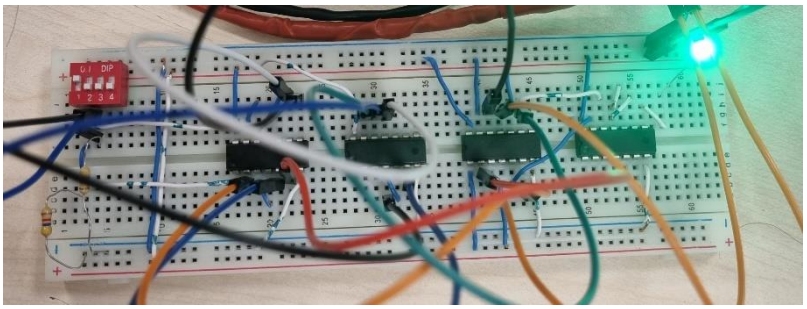

Figure 10: input "10" state transition from A to B

When I enter “11” as an input, I expect the machine to transform from state B to state C. However, I have no way of checking this as you can see in Figure 11, the output is the same and the LED gives light in both state B and C.

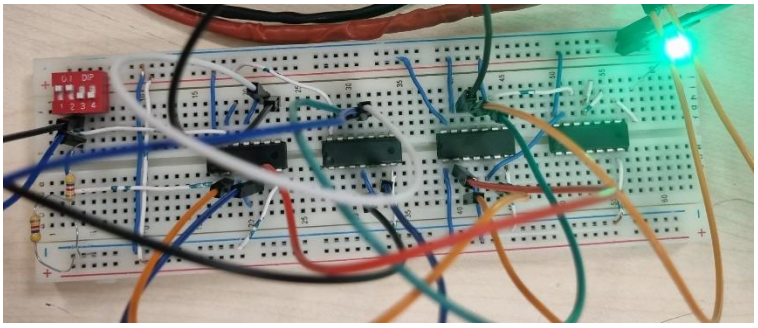

Figure 11: input 11 expected state transition from B to C

I know that in State C, the only input that can turn of the LED is “00”. Therefore, if I enter the input “01”, I expect the LED to still give light unlike state A. When I check it in Figure 12, you can see that the LED is still turned on which means that the machine changes states correctly. This verifies that my implementation is correct.

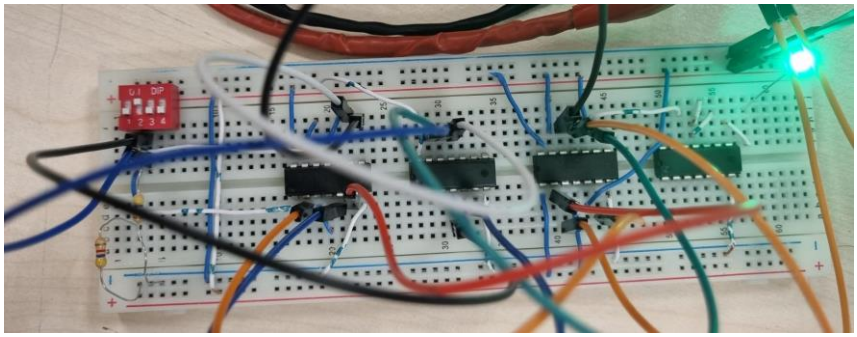

Figure 12: input "01" didn't change the output

Conclusion:

In this lab, our main was to design a custom finite state machine and implement it on a breadboard with logic gates and flip-flops. While designing the finite state machine, I used state transition diagrams and state transition tables. While designing the circuit, I used multiple Karnaugh maps and truth tables which gave me the correct circuit design. When I implemented this design to my breadboard, I made sure that I use pulldown resistors to make sure that the switches are either connected to 5V power supply or the ground of the power supply. In the end, my design worked without any errors. There were some small noise and loose connection of cables but none of these affected the operation of the circuit.

References:

https://www.ti.com/lit/ds/symlink/sn74hc00.pdf?ts=1715089360149&ref_url=https%253A%252F%252Fwww.google.com%252F

https://pdf1.alldatasheet.com/datasheet-pdf/download/27440/TI/74LS74A.html