# Implementation of finite state machine

M Sai Sarath Chandra
chandu.4567890@gmail.com
FWC22117
IIT Hyderabad-Future Wireless Communication Assignment

March 2023

## Contents

# 1 Problem

GATE EC-2020
Q.39. The state diagram of a sequence detector is shown below.State S0 is the initial state of the seuence detector.If the output is 1,then

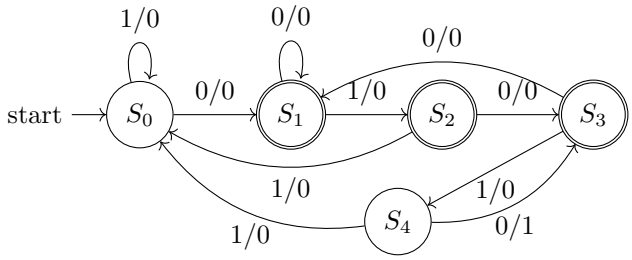


Figure 1: FSM

1. the sequence 01010 is detected.
2. the sequence 01011 is detected.
3. the sequence 01110 is detected.
4. the sequence 01001 is detected.

# 2 Introduction

The aim is to implement the above finite state machine using 7447 and 7474 IC's.Finite state machine is a device which stores a state at a given time,the state will then change based on inputs,provding the resulting output for the implemented changes.IC 7474 is a dual positive-edge-triggered D-type flip flop,which means it has two separate flip-flop that are triggered by the rising edge of a clock signal.

# 3 Components

| COMPONENTS | | |
|---|---|---|
| Component | Value | Quantity |
| Resistor | =220 Ohm | 1 |
| Arduino | UNO | 1 |
| Seven Segent Display | Common Anode | 1 |
| Decoder | 7447 | 1 |
| Flip Flop | 7474 | 2 |
| Jumper Wires | | 20 |
| Breadboard | | 1 |

Table 1: Components

# 4 Hardware

The IC 7474 is a type of flip-flop integrated circuit that is commonly used in digital electronics applications.It is a dual positive-edge-triggered by

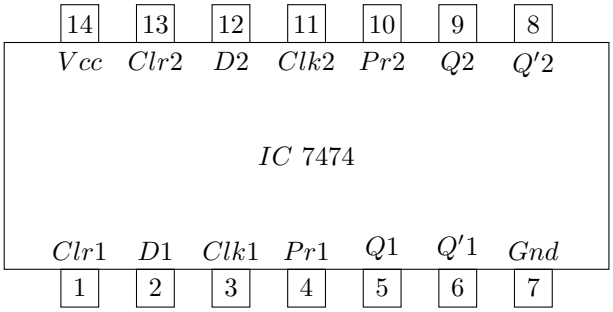


Figure 2: 7474

The connections between arduino and two 7474 IC's and one 7447 IC is:

| | INPUT | | | OUTPUT | | | CLOCK | | VCC | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ARDUINO | D9 | D10 | D11 | D2 | D3 | D4 | D13 | | 5V | | | |
| 7474 | 5 | 9 | | 2 | 12 | | 3 | 11 | 1 | 4 | 10 | 13 |
| 7474 | | | 5 | | | 2 | 3 | 11 | 1 | 4 | 10 | 13 |
| 7447 | | | | 7 | 1 | 2 | | | | | 16 | |

Table 2: Arduino-7474

The transition table of the finite state machine is given in the below table

| INPUT | | | | OUTPUT | | | |
|---|---|---|---|---|---|---|---|
| X | Y | Z | I | A | B | C | D |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

Table 3: Truth table

K-map for A is

| $ZI$ \ $XY$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 1 | 0 |
| 11 | - | - | - | - |
| 10 | 0 | 0 | - | - |

Figure 3: kmap

The resultant expression for $A$ is $A = YZI$.

K-map for B is

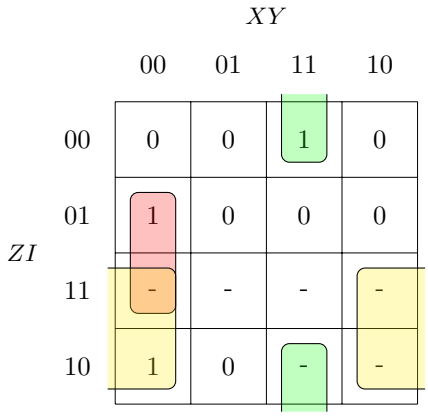


Figure 4: kmap

The resultant expression for $B$ is $B = YZ'I' + Y'ZI + XI'$.

K-map for C is

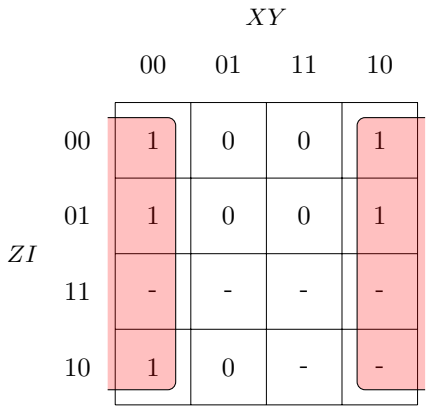


Figure 5: kmap

The resultant expression for $C$ is $C = I'$.

K-map for D is

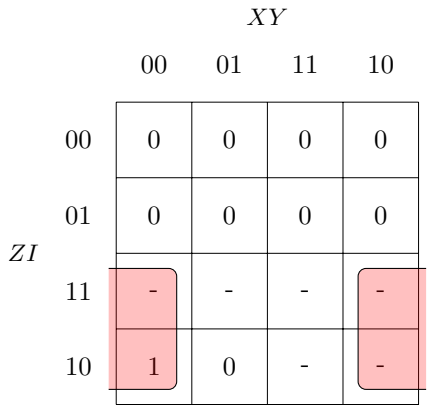


Figure 6: kmap

The resultant expression for $D$ is $D = XI'$.

# 5 Software

The assembly code to implement the above finite state machine is:

```
.include "/sdcard/assembly/assignment/m328Pdef.inc"
.def x=r18
.def y=r19
.def z=r20
.def A=r21
.def B=r22
.def C=r23
.def D=r24
.def i=r27

setup:

ldi r16,0b00111100
out DDRD,r16

ldi r17,0b00100000
out DDRB,r17

ldi r25,0b11111111

loop:
```

```
in r18,pinD
ldi r26,0b01000000
and r26,r25
mov r18,r26
lsr r18

in r19,pinD
ldi r26,0b10000000
and r26,r25
mov r19,r26
lsr r19
lsr r19

in r20,pinB
ldi r26,0b00000001
and r26,r25
mov r20,r26
ldi r30,0b00000101
rcall assembly

out PortB,r17
call delay
;for A
mov r26,r19
mov r28,r20
and r26,r28
mov r28,r27
and r26,r28
mov r21,r26
;for B
mov r26,r19
com r26
and r26,r20
and r26,r27
mov r31,r19
com r20
com r27
and r31,r20
and r31,r27
mov r29,r18
and r29,r20
and r29,r27
or r26,r31
or r26,r29
mov r22,r26
com r20
com r27
;for C
mov r23,r27
com r23
;for D
mov r26,r18
com r27
and r26,r27
mov r24,r26
com r27
```

```
com r17
out portB,r17
call delay
call output
rjmp loop
start:rjmp start

output:ldi r26,0b00100000
and r26,r16
out PortD,r26

assembly:lsl r20
         dec r30
         brne assembly
         ret

delay:ldi r23,100
loop1:
ldi r24,100
loop2:
ldi r29,100
loop3:
dec r29
brne loop2
dec r24
brne loop1
dec r23
brne delay
ret
```