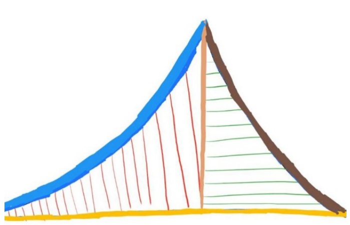
DIGITAL DESIGN

Through Embedded Programming

G. V. V. Sharma



Contents

Introduction

This book introduces digital design through using the arduino framework.

Chapter 1

Installation

1.1. Termux

1. On your android device, install fdroid apk from

```
https://www.f-droid.org/
```

2. Install Termux from apkpure

3. Install basic packages on termux

```
#Give termux access to your user directory in android
termux-setup-storage


#Upgrade packages
apt update && apt upgrade
apt install build-essential openssh


#Mandatory packages
apt install curl git wget subversion proot proot-distro python nmap neovim ranger
#-------------------End Install Termux
```

```
—————————————————————————————
```

4. Install Ubuntu on termux

```
proot−distro install ubuntu
proot−distro login ubuntu
```

1.2. Platformio

1. Install Packages

```
apt update && apt upgrade
apt install apt−utils build−essential cmake neovim
apt install git wget subversion imagemagick nano
apt install avra avrdude gcc−avr avr−libc
#−−−−−−−−−−−−−−−−−−−End Installing ubuntu on termux
    −−−−−−−−−−−−−−−−−−−−−−−−−−−−−


#−−−−−−−−−−−−−−−−−−− Installing python3 on termuxubuntu
    −−−−−−−−−−−−−−−−−−−−−−−−−−−−−
apt install python3−pip python3−numpy python3−scipy python3−matplotlib
    python3−mpmath python3−sympy python3−cvxopt
#−−−−−−−−−−−−−−−−−−− End installing python3 on termuxubuntu
    −−−−−−−−−−−−−−−−−−−−−−−−−−−−−


#−−−−−−−−−−−−−−−−−−− Installing platformio on termuxubuntu
```

```
——————————————————————————————
pip3 install platformio
#——————————————————— End installing python3 on termuxubuntu
——————————————————————————————
```

2. Execute the following on ubuntu

```
cd ide/piosetup/codes
pio run
```

3. Connect your arduino to the laptop/rpi and type

```
pio run –t nobuild –t upload
```

4. The LED beside pin 13 will start blinking

1.3. Arduino Droid

1. Install ArduinoDroid from apkpure

2. Open ArduinoDroid and grant all permissions

3. Connect the Arduino to your phone via USB-OTG

4. For flashing the bin files, in ArduinoDroid,

```
Actions–>Upload–>Upload Precompiled
```

then go to your working directory and select

```
pio/build/uno/firmwarehex
```

for uploading hex file to the Arduino Uno

5. The LED beside pin 13 will start blinking

Chapter 2

Seven Segment Display

We show how to control a seven segment display.

2.1. Components

Component	Value	Quantity
Breadboard		1
Resistor	$\geq 220\Omega$	1
Arduino	Uno	1
Seven Segment Display	Common Anode	1
Jumper Wires		20

Table 2.1:

2.1.1. Breadboard

The breadboard can be divided into 5 segments. In each of the green segements, the pins are internally connected so as to have the same voltage. Similarly, in the central segments, the pins in each column are internally connected in the same fashion as the blue columns.

2.1.2. Seven Segment Display

The seven segment display in Fig. 2.2 has eight pins, a, b, c, d, e, f, g and *dot* that take an active LOW input, i.e. the LED will glow only if the input is connected to ground. Each of these pins is connected to an LED segment. The *dot* pin is reserved for the $\cdot$ LED.

2.1.3. Arduino

The Arduino Uno has some ground pins, analog input pins A0-A3 and digital pins D1-D13 that can be used for both input as well as output. It also has two power pins that can generate $3.3V$ and $5V$. In the following exercises, only the GND, $5V$ and digital pins will be used.

2.2. Display Control through Hardware

2.2.1. Powering the Display

1. Plug the display to the breadboard in Fig. 2.1 and make the connections in Table 2.2. Henceforth, all 5V and GND connections will be made from the breadboard.

Arduino	**Breadboard**
5V	Top Green
GND	Bottom Green

Table 2.2:

2. Make the connections in Table 2.3.

3. Connect the Arduino to the computer. The DOT led should glow.

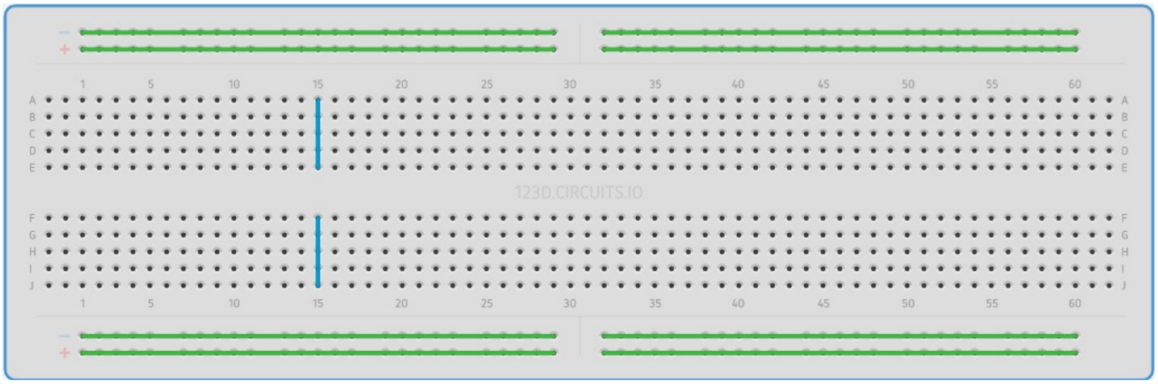


Figure 2.1:

Breadboard		**Display**
5V	Resistor	COM
GND		DOT

Table 2.3:

2.2.2. Controlling the Display

Fig. 2.3 explains how to get decimal digits using the seven segment display. GND=0.

1. Generate the number 1 on the display by connecting only the pins *b* and *c* to GND (=0). This corresponds to the first row of 2.4. 1 means not connecting to GND.

2. Repeat the above exercise to generate the number 2 on the display.

3. Draw the numbers 0-9 as in Fig. 2.3 and complete Table 2.4

a	**b**	**c**	**d**	**e**	**f**	**g**	**decimal**
0	0	0	0	0	0	1	0

Table 2.4:

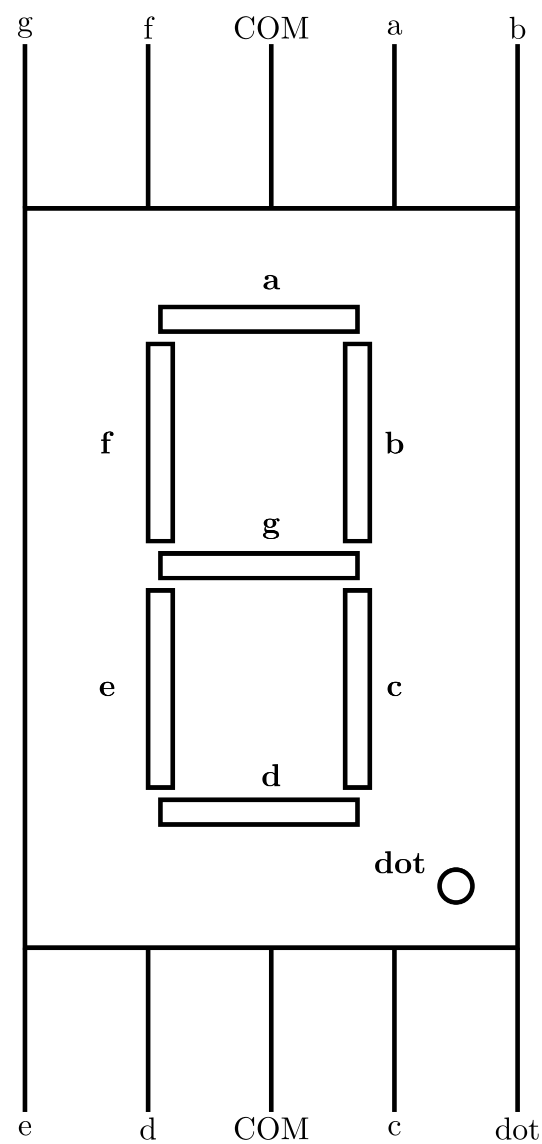


Figure 2.2:

2.3. Display Control through Software

1. Make connections according to Table 2.5

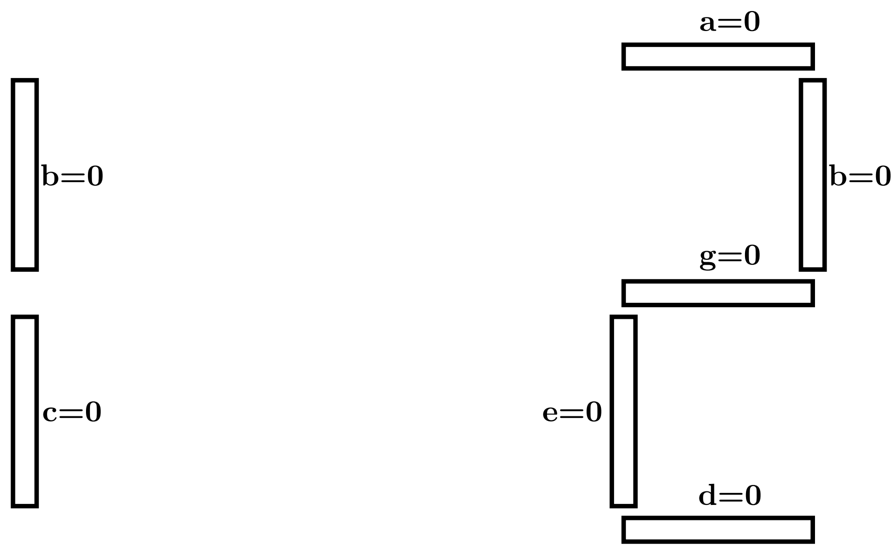


Figure 2.3:

Arduino	2	3	4	5	6	7	8
Display	a	b	c	d	e	f	g

Table 2.5:

2. Download the following code using the arduino IDE and execute

   ide/sevenseg/codes/sevenseg/sevenseg.ino

3. Now generate the numbers 0-9 by modifying the above program.

Chapter 3

7447

Here we show how to use the 7447 BCD-Seven Segment Display decoder to learn Boolean logic.

3.1. Components

Component	Value	Quantity
Resistor	220 Ohm	1
Arduino	UNO	1
Seven Segment Display		1
Decoder	7447	1
Jumper Wires	M-M	20
Breadboard		1

Table 3.1:

3.2. Hardware

1. Make connections between the seven segment display in Fig. 2.2 and the 7447 IC in Fig. 3.1 as shown in Table 3.2

7447	$\bar{a}$	$\bar{b}$	$\bar{c}$	$\bar{d}$	$\bar{e}$	$\bar{f}$	$\bar{g}$
Display	a	b	c	d	e	f	g

Table 3.2:

2. Make connections to the lower pins of the 7447 according to Table 3.3 and connect $V_{CC} = 5$V. You should see the number 0 displayed for 0000 and 1 for 0001.

D	**C**	**B**	**A**	**Decimal**
0	0	0	0	0
0	0	0	1	1

Table 3.3:

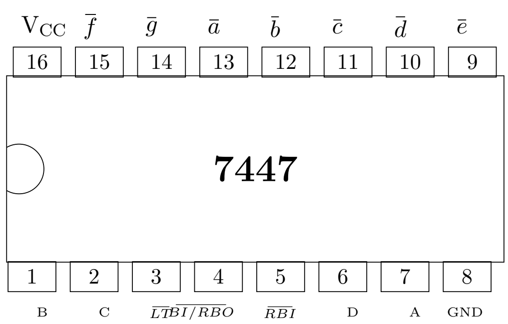


Figure 3.1:

3. Complete Table 3.3 by generating all numbers between 0-9.

3.3. Software

1. Now make the connections as per Table 3.4 and execute the following program

```
ide/7447/codes/gvv_ard_7447/gvv_ard_7447.cpp
```

7447	D	C	B	A
Arduino	5	4	3	2

Table 3.4:

In the truth table in Table 3.5, W, X, Y, Z are the inputs and A, B, C, D are the outputs. This table represents the system that increments the numbers 0-8 by 1 and resets the number 9 to 0 Note that $D = 1$ for the inputs 0111 and 1000. Using boolean logic,

$$D = WXYZ^{'} + W^{'}X^{'}Y^{'}Z \tag{3.1}$$

Note that 0111 results in the expression $WXYZ^{'}$ and 1000 yields $W^{'}X^{'}Y^{'}Z$.

2. The code below realizes the Boolean logic for B, C and D in Table 3.5. Write the logic for A and verify.

```
ide/7447/codes/inc_dec/inc_dec.ino
```

Z	Y	X	W	**D**	**C**	**B**	**A**
0	0	0	0	**0**	**0**	**0**	**1**
0	0	0	1	**0**	**0**	**1**	**0**
0	0	1	0	**0**	**0**	**1**	**1**
0	0	1	1	**0**	**1**	**0**	**0**
0	1	0	0	**0**	**1**	**0**	**1**
0	1	0	1	**0**	**1**	**1**	**0**
0	1	1	0	**0**	**1**	**1**	**1**
0	1	1	1	**1**	**0**	**0**	**0**
1	0	0	0	**1**	**0**	**0**	**1**
1	0	0	1	**0**	**0**	**0**	**0**

Table 3.5:

3. Now make additional connections as shown in Table 3.6 and execute the following code. Comment.

```
ide/7447/codes/ip_inc_dec/ip_inc_dec.cpp
```

Solution: In this exercise, we are taking the number 5 as input to the arduino and displaying it on the seven segment display using the 7447 IC.

	Z	Y	X	W
Input	0	1	0	1
Arduino	9	8	7	6

Table 3.6:

4. Verify the above code for all inputs from 0-9.

5. Now write a program where

   (a) the binary inputs are given by connecting to 0 and 1 on the breadboard

   (b) incremented by 1 using Table 3.5 and

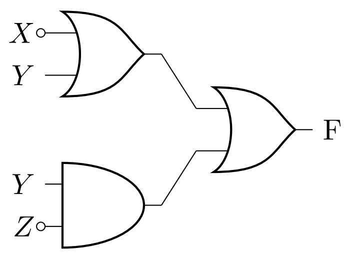
   (c) the incremented value is displayed on the seven segment display.

6. Write the truth table for the 7447 IC and obtain the corresponding boolean logic equations.

7. Implement the 7447 logic in the arudino. Verify that your arduino now behaves like the 7447 IC.

3.4. Problems

1. Obtain the Boolean Expression for the Logic circuit shown below (CBSE 2013)



2. Verify the Boolean Expression (CBSE 2013)

$$A + C = A + A'C + BC \tag{3.2}$$

3. Draw the Logic Circuit for the following Boolean Expression (CBSE 2015)

$$f(x, y, z, w) = (x' + y)z + w' \tag{3.3}$$

4. Verify the following (CBSE 2015)

$$U' + V = U'V' + U'V + UV \tag{3.4}$$

5. Draw the Logic Circuit for the given Boolean Expression (CBSE 2015)

$$(U + V')W' + Z \tag{3.5}$$

6. Verify the following using Boolean Laws (CBSE 2015)

$$X + Y' = XY + XY' + X'Y' \tag{3.6}$$

7. Write the Boolean Expression for the result of the Logic Circuit as shown in Fig. 3.2 (CBSE 2016)

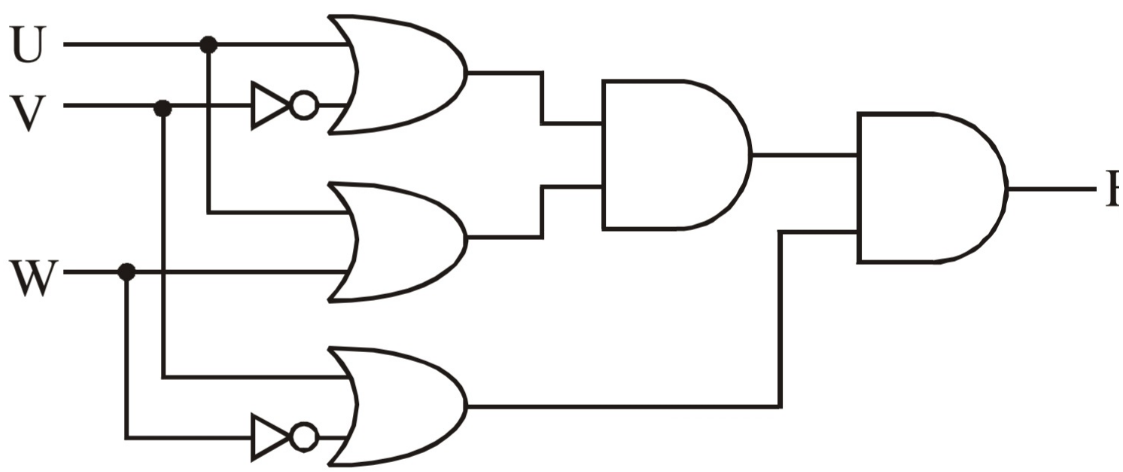


Figure 3.2:

8. Draw the logic circuit of the following Boolean Expression using only NAND Gates. (CBSE 2017)

$$XY + YZ \tag{3.7}$$

9. Draw the Logic Circuit of the following Boolean Expression using only NOR Gates (CBSE 2017)

$$(A + B)(C + D) \tag{3.8}$$

10. Draw the Logic Circuit of the following Boolean Expression (CBSE 2018)

$$(U' + V)(V' + W') \tag{3.9}$$

11. Derive a Canonical POS expression for a Boolean function F, represented by Table 3.7 (CBSE 2019)

X	Y	Z	F(X,Y,Z)
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Table 3.7:

12. For the logic circuit shown in Fig.3.3, find the simplified Boolean expression for the output. (GATE EC 2000)

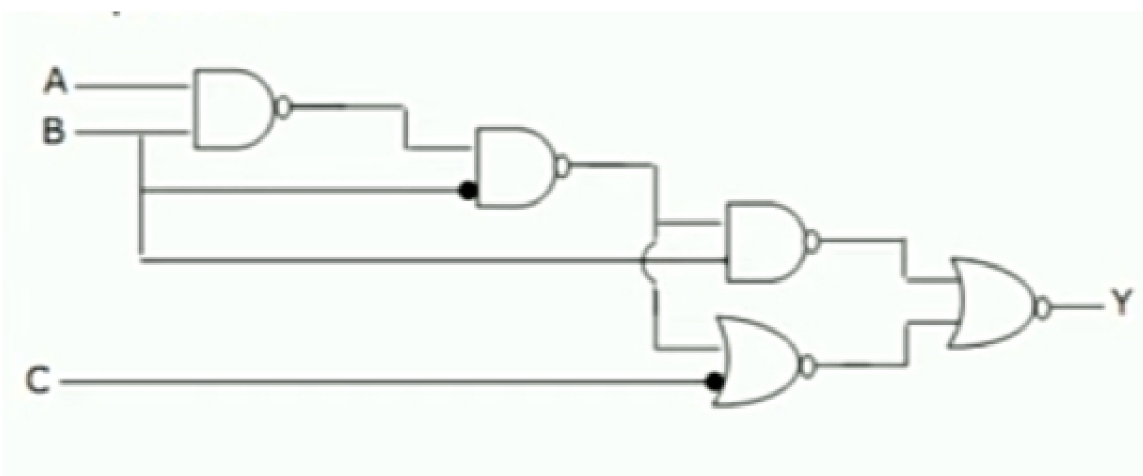
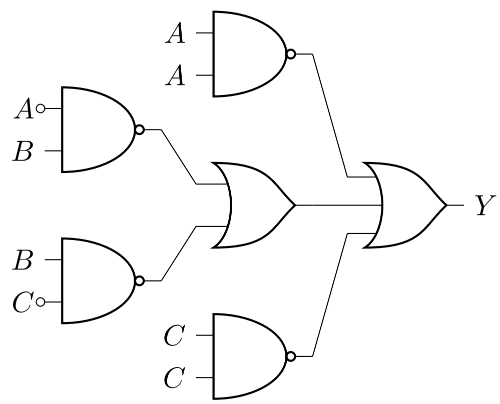


Figure 3.3:

13. Obtain the Boolean Expression for the Logic circuit shown below (GATE EC 1993)



14. Implement Table 3.8 using XNOR logic. (GATE EC 1993)

A	**B**	**Y**
0	0	1
0	1	0
1	0	0
1	1	1

Table 3.8:

15. For a binary half-sub-tractor having two inputs A and B, find the correct set of logical expressions for the outputs D (=A minus B) and X (=borrow). (GATE EC 1999)

16. Find X in the following circuit in Fig. 3.4 (GATE EC 2007)

17. A logic circuit implements the boolean function F=X'.Y+X.Y'.Z'. It is found that the input combination X=Y=1 can never occur. Taking this into account, find a simplified expression for F. (GATE IN 2007)

18. Find the Boolean logic realised by the following circuit in Fig. 3.5 (GATE EC 2010)

19. Find the logic function implemented by the circuit given below in Fig. 3.6 (GATE EC 2011)

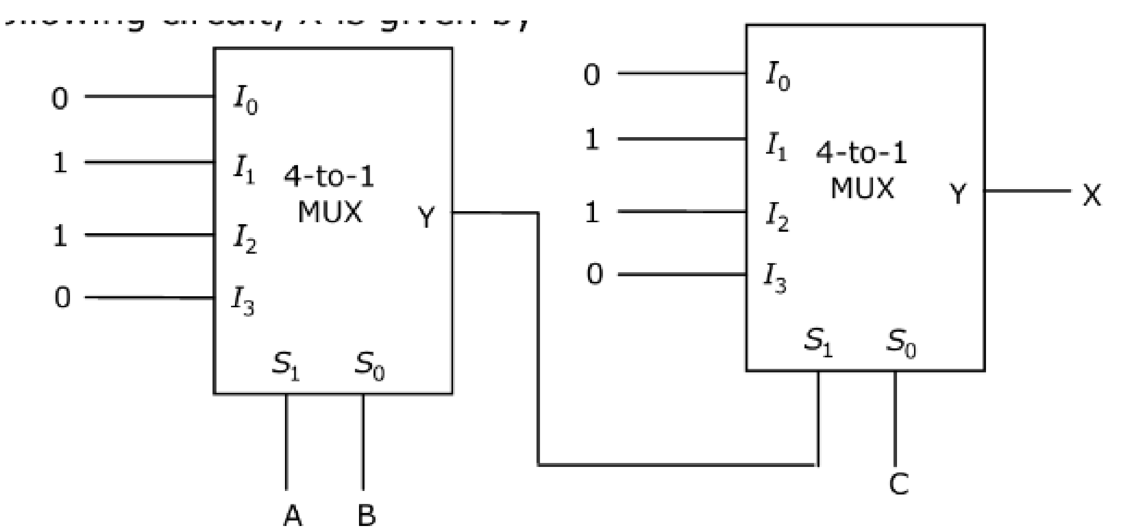


Figure 3.4:

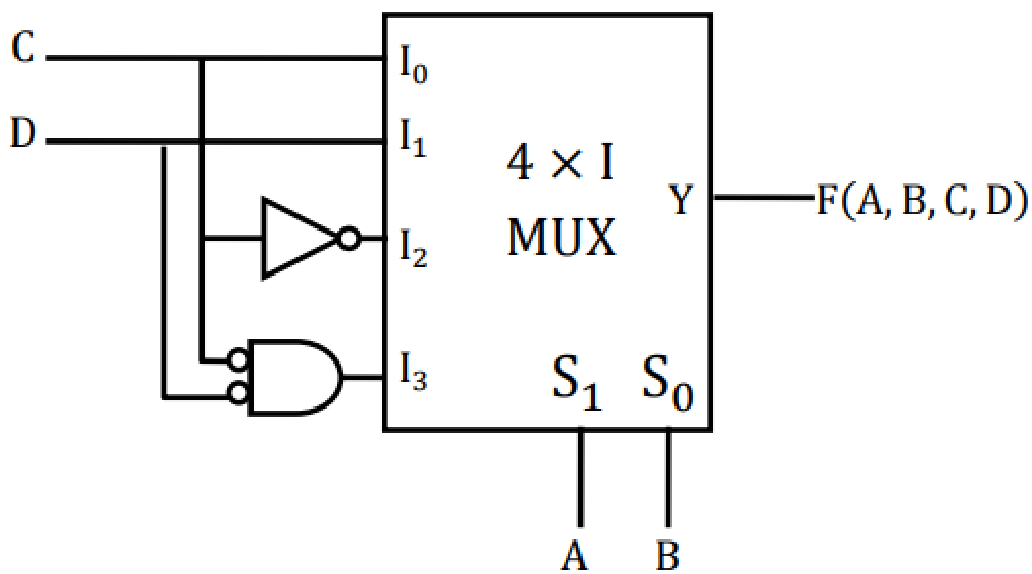


Figure 3.5:

20. Find F in the Digital Circuit given in the figure below in Fig. 3.7. (GATE IN 2016)

21. Find the logic function implemented by the circuit given below in Fig. 3.8 (GATE

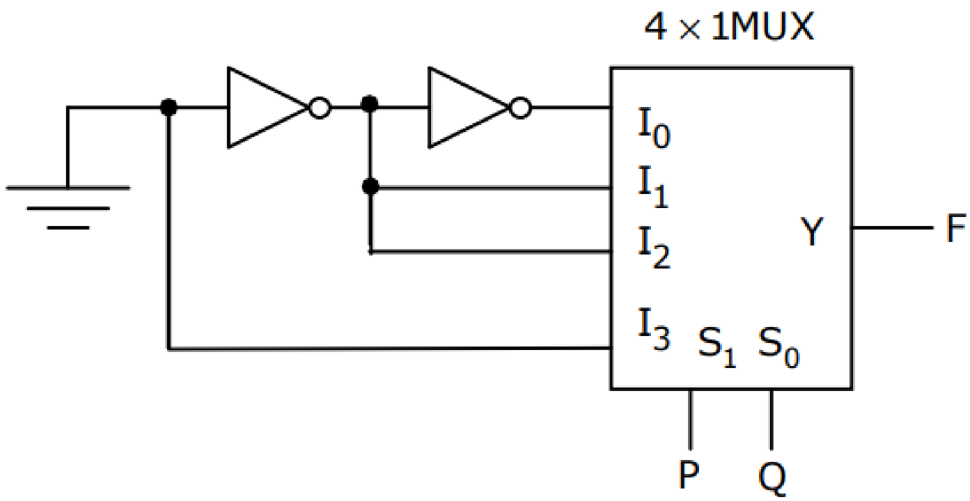


Figure 3.6:

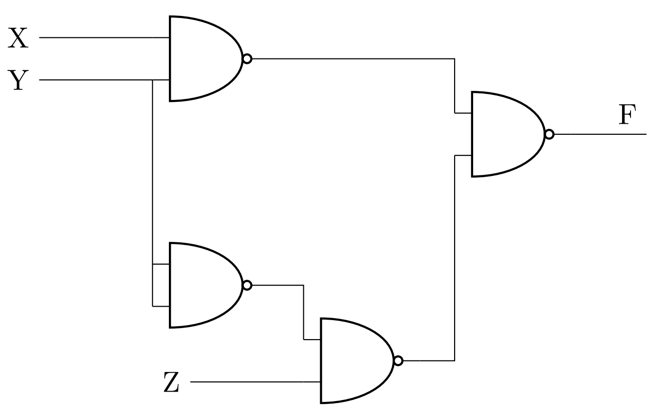


Figure 3.7:

EC 2017)

22. Find the logic function implemented by the circuit given below in Fig. 3.9 (GATE EC 2018)

23. Find the logic function implemented by the circuit given below in Fig. 3.10 (GATE EE 2018)

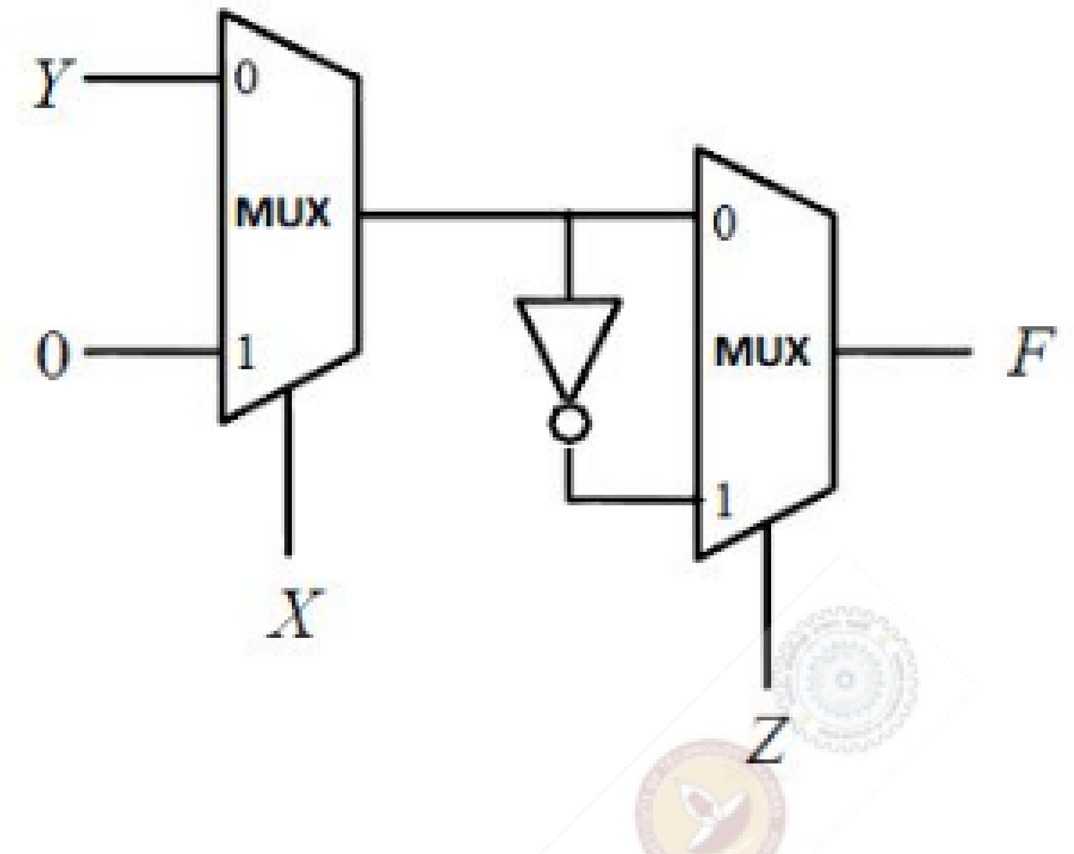


Figure 3.8:

24. Find the logic function implemented by the circuit given below in Fig. 3.11 (GATE EE 2019)

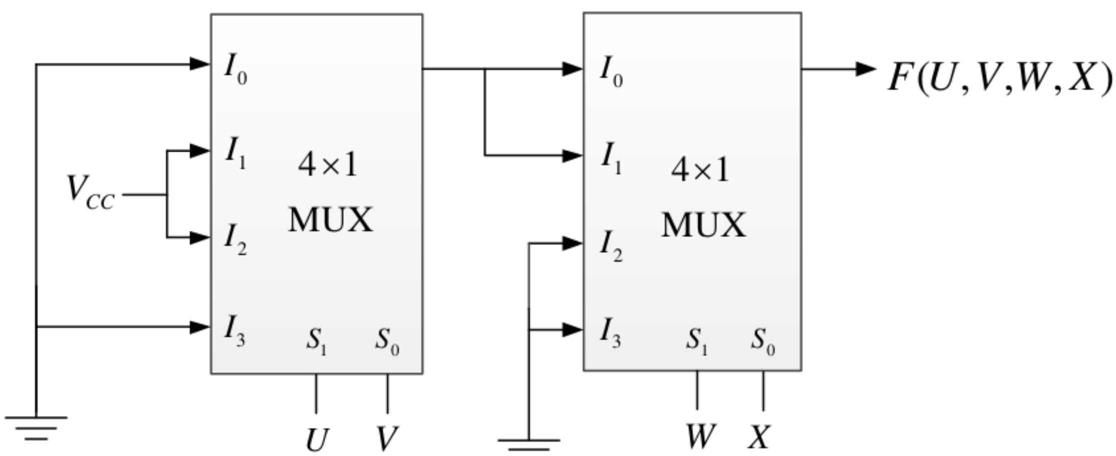


Figure 3.9:

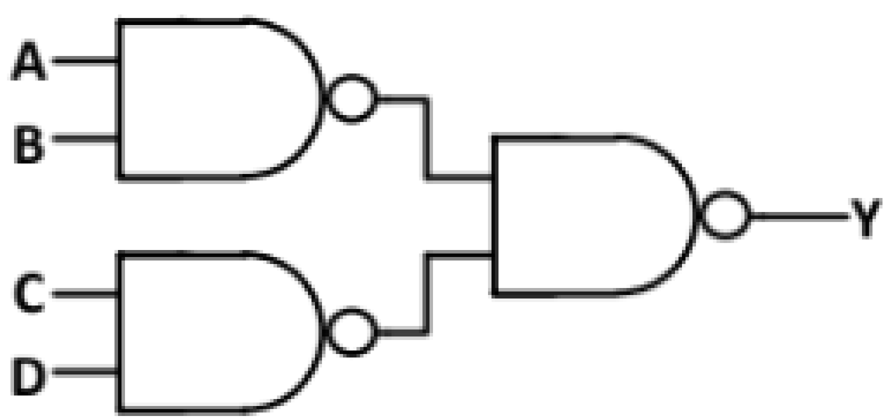


Figure 3.10:

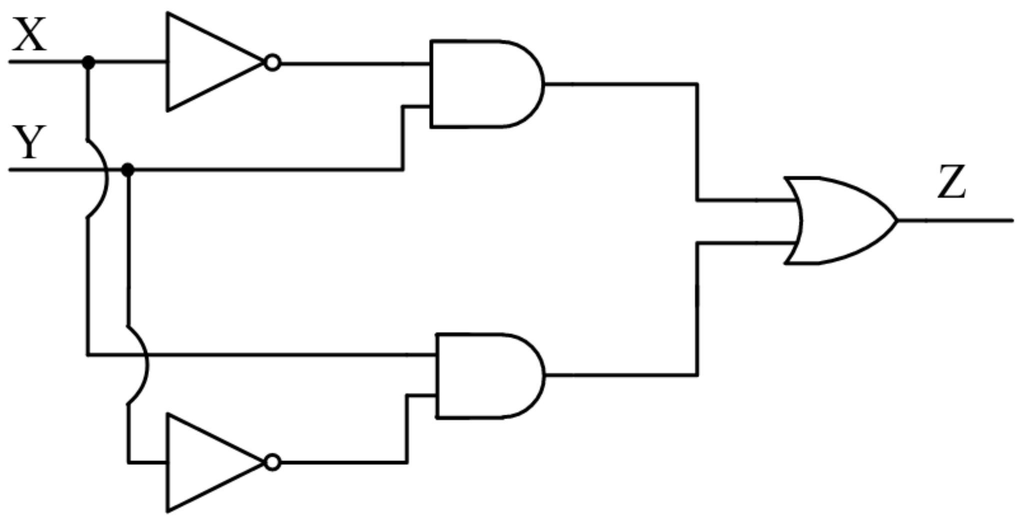


Figure 3.11:

Chapter 4

Karnaugh Map

4.1. Introduction

We explain Karnaugh maps (K-map) by finding the logic functions for the incrementing decoder

4.2. Incrementing Decoder

The incrementing decoder takes the numbers $0, , \ldots, 9$ in binary as inputs and generates the consecutive number as output The corresponding truth table is available in Table 4.1

4.3. Karnaugh Map

Using Boolean logic, output A in Table 4.1 can be expressed in terms of the inputs W, X, Y, Z as

$$A = W'X'Y'Z' + W'XY'Z' + W'X'YZ' + W'XYZ' + W'X'Y'Z \quad (4.1)$$

Z	Y	X	W	**D**	**C**	**B**	**A**
0	0	0	0	**0**	**0**	**0**	**1**
0	0	0	1	**0**	**0**	**1**	**0**
0	0	1	0	**0**	**0**	**1**	**1**
0	0	1	1	**0**	**1**	**0**	**0**
0	1	0	0	**0**	**1**	**0**	**1**
0	1	0	1	**0**	**1**	**1**	**0**
0	1	1	0	**0**	**1**	**1**	**1**
0	1	1	1	**1**	**0**	**0**	**0**
1	0	0	0	**1**	**0**	**0**	**1**
1	0	0	1	**0**	**0**	**0**	**0**

Table 4.1:

1. K-Map for A: The expression in (4.1) can be minimized using the K-map in Fig 4.1 In Fig 4.1, the implicants in boxes 0,2,4,6 result in $W'Z'$ The implicants in boxes 0,8 result in $W'X'Y'$ Thus, after minimization using Fig 4.2, (4.1) can be expressed as

$$A = W'Z' + W'X'Y' \tag{4.2}$$

   Using the fact that

$$\begin{aligned} X + X' &= \\ XX' &= 0, \end{aligned} \tag{4.3}$$

   derive (4.2) from (4.1) algebraically

2. K-Map for B: From Table 4.1, using boolean logic,

$$B = WX'Y'Z' + W'XY'Z' + WX'YZ' + W'XYZ' \tag{4.4}$$

   Show that (4.4) can be reduced to

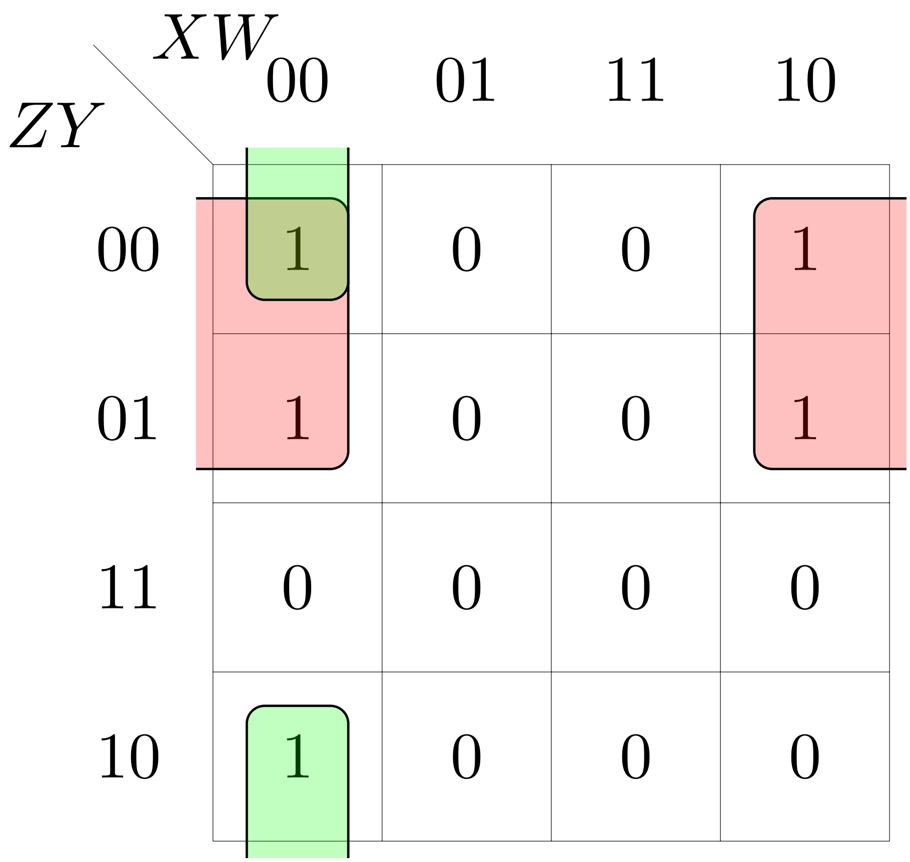


Figure 4.1: K-map for A

$$B = WX'Z' + W'XZ' \tag{4.5}$$

using Fig 4.2

3. Derive (4.5) from (4.4) algebraically using (4.3)

$ZY \backslash XW$	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	0	0	0
10	0	0	0	0

Figure 4.2: K-map for B

4. K-Map for C: From Table 4.1, using boolean logic,

$$C = WXY'Z' + W'X'YZ' + WX'YZ' + W'XYZ' \tag{4.6}$$

Show that (4.6) can be reduced to

$ZY \backslash XW$	00	01	11	10
00	0	0	1	0
01	1	1	0	1
11	0	0	0	0
10	0	0	0	0

Figure 4.3: K-map for C

$$C = WXY'Z' + X'YZ' + W'YZ' \tag{4.7}$$

using Fig 4.3

5. Derive (4.7) from (4.6) algebraically using (4.3)

6. K-Map for D: From Table 4.1, using boolean logic,

$$D = WXYZ' + W'X'Y'Z \tag{4.8}$$

$ZY \backslash XW$	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	0	0	0	0
10	1	0	0	0

Figure 4.4: K-map for D

7. Minimize (4.8) using Fig 4.4

D	C	B	A	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**Decimal**
0	0	0	0	**0**	**0**	**0**	**0**	**0**	**0**	**1**	0
0	0	0	1	**1**	**0**	**0**	**1**	**1**	**1**	**1**	1
0	0	1	0	**0**	**0**	**1**	**0**	**0**	**1**	**0**	2
0	0	1	1	**0**	**0**	**0**	**0**	**1**	**1**	**0**	3
0	1	0	0	**1**	**0**	**0**	**1**	**1**	**0**	**0**	4
0	1	0	1	**0**	**1**	**0**	**0**	**1**	**0**	**0**	5
0	1	1	0	**0**	**1**	**0**	**0**	**0**	**0**	**0**	6
0	1	1	1	**0**	**0**	**0**	**1**	**1**	**1**	**1**	7
1	0	0	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	8
1	0	0	1	**0**	**0**	**0**	**1**	**1**	**0**	**0**	9

Table 4.2: Truth table for display decoder.

8. Execute the code in

```
ide/7447/codes/inc_dec/inc_dec.cpp
```

and modify it using the K-Map equations for A,B,C and D Execute and verify

9. Display Decoder: Table 4.2 is the truth table for the display decoder in Fig. 3.1. Use K-maps to obtain the minimized expressions for a, b, c, d, e, f, g in terms of A, B, C, D with and without don't care conditions

4.4. Dont Care

We explain Karnaugh maps (K-map) using don't care conditions

4.5. Don't Care Conditions

1. Don't Care Conditions: 4 binary digits are used in the incrementing decoder in Table 4.1. However, only the numbers from 0-9 are used as input/output in the decoder and we don't care about the numbers from 0-5 This phenomenon can be addressed by revising the truth table in Table 4.1 to obtain Table 4.3

Z	Y	X	W	**D**	**C**	**B**	**A**
0	0	0	0	**0**	**0**	**0**	**1**
0	0	0	1	**0**	**0**	**1**	**0**
0	0	1	0	**0**	**0**	**1**	**1**
0	0	1	1	**0**	**1**	**0**	**0**
0	1	0	0	**0**	**1**	**0**	**1**
0	1	0	1	**0**	**1**	**1**	**0**
0	1	1	0	**0**	**1**	**1**	**1**
0	1	1	1	**1**	**0**	**0**	**0**
1	0	0	0	**1**	**0**	**0**	**1**
1	0	0	1	**0**	**0**	**0**	**0**
1	0	1	0	**-**	**-**	**-**	**-**
1	0	1	1	**-**	**-**	**-**	**-**
1	1	0	0	**-**	**-**	**-**	**-**
1	1	0	1	**-**	**-**	**-**	**-**
1	1	1	0	**-**	**-**	**-**	**-**
1	1	1	1	**-**	**-**	**-**	**-**

Table 4.3:

2. The revised K-map for A is available in Fig 4.5. Show that

$$A = W' \tag{4.9}$$

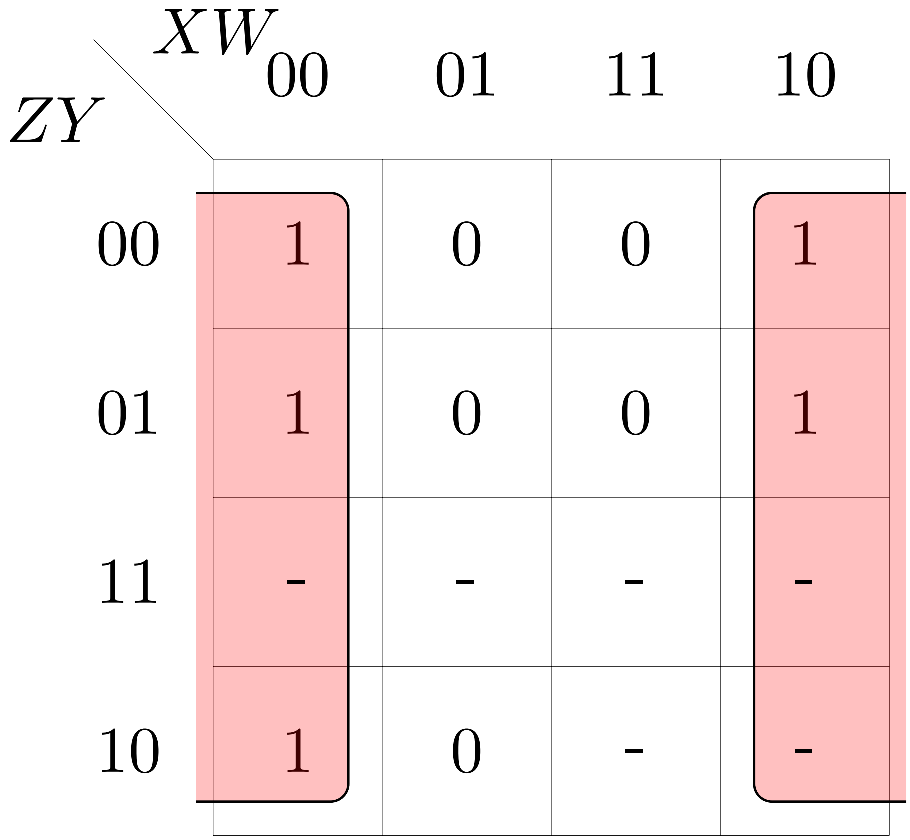


Figure 4.5: K-map for A with don't cares

3. The revised K-map for B is available in Fig 4.6 Show that

$$B = WX'Z' + W'X \tag{4.10}$$

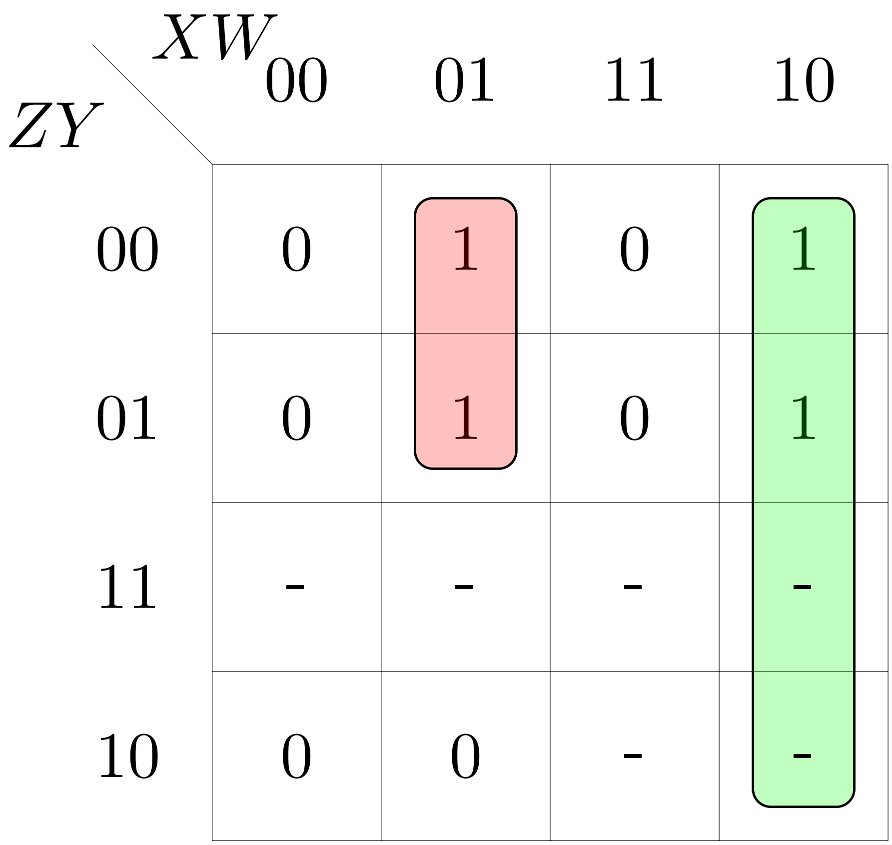


Figure 4.6: K-map for B with don't cares

4. The revised K-map for C is available in Fig 4.7 Show that

$$C = X'Y + W'Y + WXY' \tag{4.11}$$

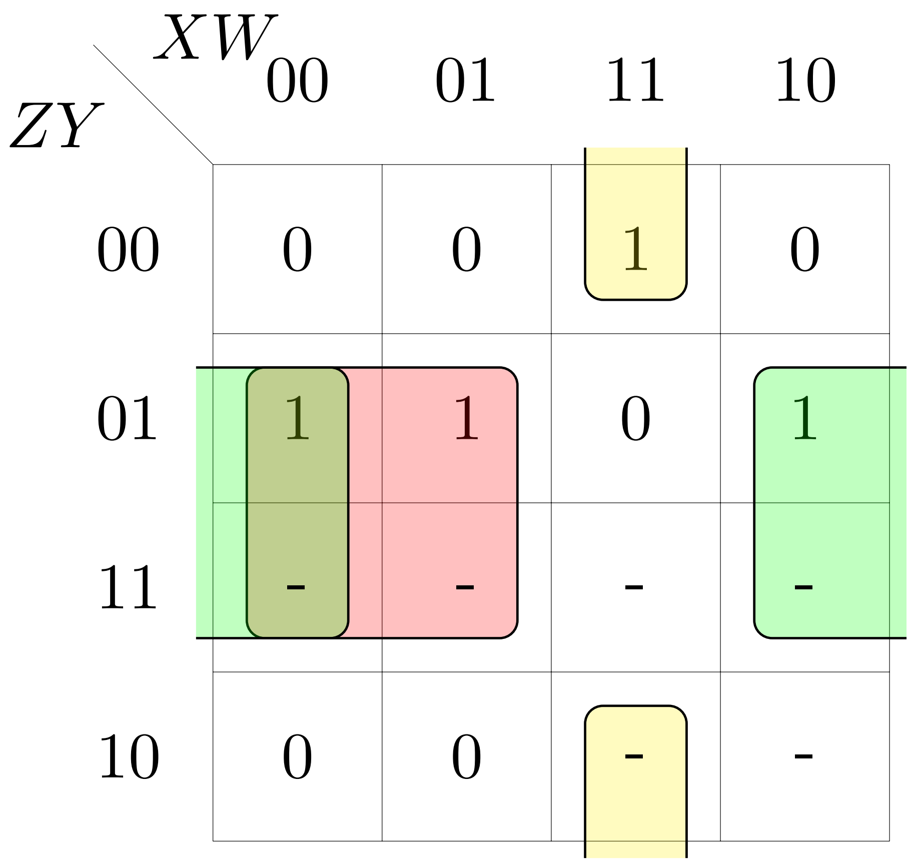


Figure 4.7: K-map for C with don't cares

5. The revised K-map for D is available in Fig 4.8 Show that

$$D = W'Z + WXY \tag{4.12}$$

6. Verify the incrementing decoder with don't care conditions using the arduino

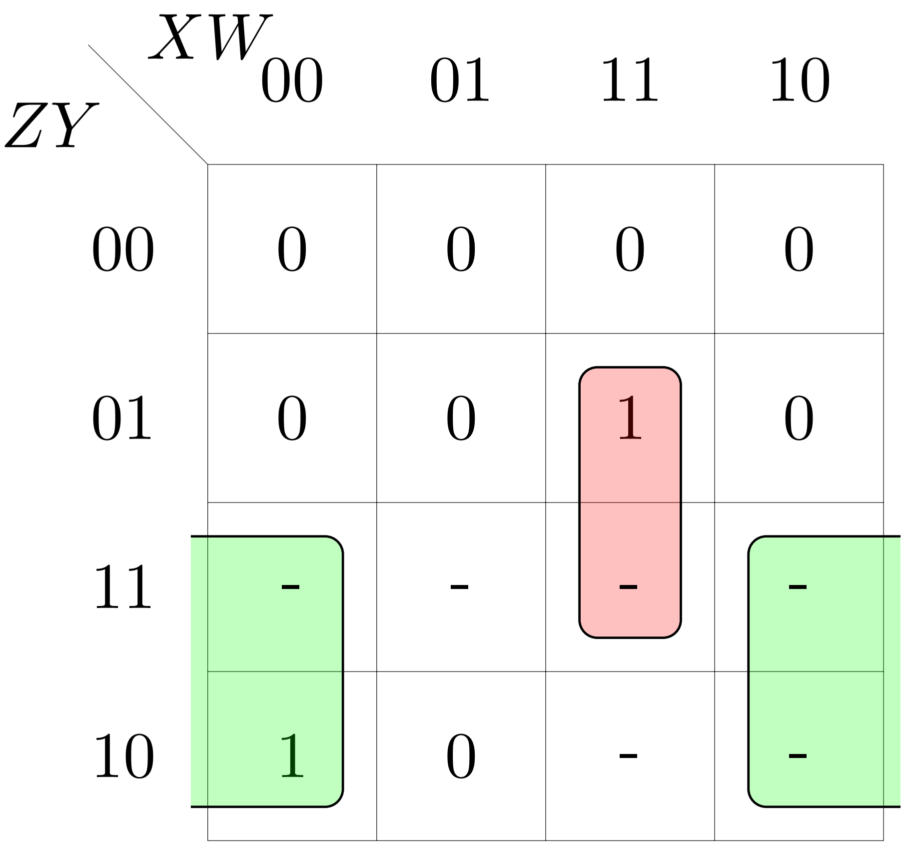


Figure 4.8: K-map for D with don't cares

7. Display Decoder: Use K-maps to obtain the minimized expressions for a, b, c, d, e, f, g in terms of A, B, C, D with don't care conditions

8. Verify the display decoder with don't care conditions using arduino

4.6. Problems

1. Obtain the Minimal Form for the Boolean Expression (CBSE 2013)

$$H(P, Q, R, S) = \sum(0, 1, 2, 3, 5, 7, 8, 9, 10, 14, 15) \tag{4.13}$$

2. Write the POS form for the function G shown in Table 4.4. (CBSE 2013)

U	V	W	G
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Table 4.4:

3. Reduce the following Boolean Expression to its simplest form using K-Map (CBSE 2015)

$$F(X, Y, Z, W) = (0, 1, 4, 5, 6, 7, 8, 9, 11, 15) \tag{4.14}$$

4. Derive a Canonical POS expression for a Boolean function F, represented by the following truth table (CBSE 2015)

5. (CBSE 2015) Reduce the following Boolean Expression to its simplest form using

X	Y	Z	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Table 4.5:

K-map

$$F(X, Y, Z, W) = \sum(0, 1, 6, 8, 9, 10, 11, 12, 15) \tag{4.15}$$

6. Reduce the following Boolean Expression to its simplest form using K-map. (CBSE 2016)

$$F(X, Y, Z, W) = \sum(2, 6, 7, 8, 9, 10, 11, 13, 14, 15) \tag{4.16}$$

7. Derive a Canonical POS expression for a Boolean function F, represented in Table 4.6 (CBSE 2016)

8. Verify the following (CBSE 2016)

$$A' + B'C = A'B'C' + A'BC' + A'BC + A'B'C + AB'C \tag{4.17}$$

9. Reduce the following boolean expression to it's simplest form using K-Map (CBSE

P	Q	R	F(P, Q, R)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Table 4.6:

2017)

$$F(X, Y, Z, W) = \sum(0, 1, 2, 3, 4, 5, 10, 11, 14) \quad (4.18)$$

10. Reduce the following Boolean Expression to its simplest form using K-Map. (CBSE 2017)

$$E(U, V, Z, W) = (2, 3, 6, 8, 9, 10, 11, 12, 13) \quad (4.19)$$

11. Derive a canonical POS expression for a Boolean function G, represented by Table 4.7 (CBSE 2017)

12. Derive a canonical POS expression for a Boolean function FN, represented by Table 4.8. (CBSE 2018)

13. Reduce the following Boolean expression in the simplest form using K-Map.

$$F(P, Q, R, S) = \sum(0, 1, 2, 3, 5, 6, 7, 10, 14, 15) \quad (4.20)$$

X	Y	Z	G(X,Y,Z)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Table 4.7:

X	Y	Z	FN(X,Y,Z)
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Table 4.8:

(CBSE 2019)

14. Fig. 4.9 below shows a muliplexer where S0 and S1 are the select lines, I0 to I3 are the input lines, EN is the enable line and F(P,Q,R) is the output. Find the boolean expression for output F as function of inputs P,Q,R using K-map. (GATE EC 2020)

15. The four variable function f is given in terms of min-terms as

$$f(A,B,C,D)=\sum m(2,3,8,10,11,12,14,15) \tag{4.21}$$

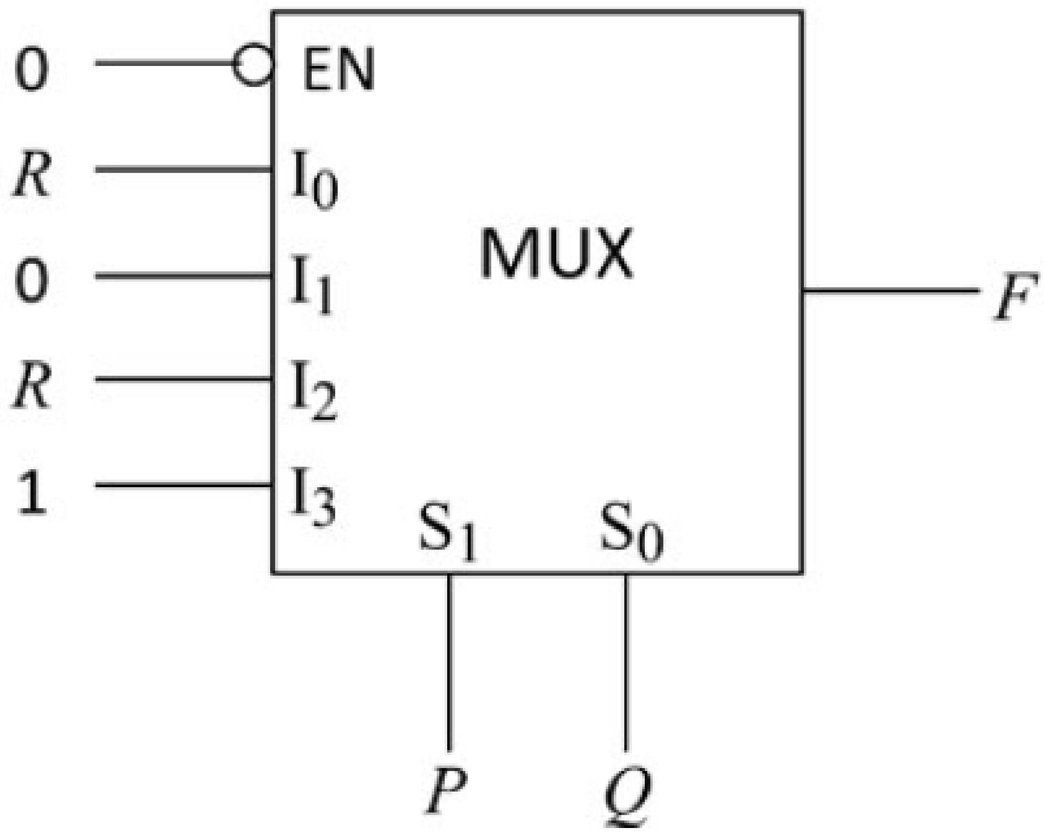


Figure 4.9:

Using the K-map minimize the function in the sum of products form. (GATE EC 1991)

16. Find the logic realized by the circuit in Fig. 4.10. (GATE EC 1992)

17. A combinational circuit has three inputs A, B and C and an output F. F is true only for the following input combinations. (GATE EC 1992)

    (a) A is false and B is true

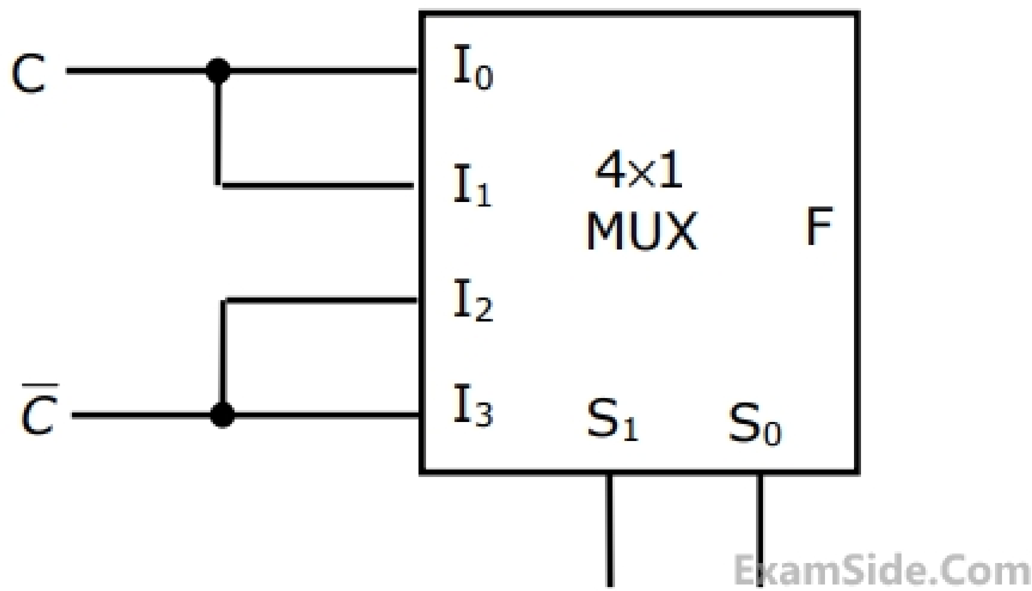


Figure 4.10:

(b) A is false and C is true

(c) A, B and C are all false

(d) A, B and C are all true

(a) Write the truth table for F. use the convention, true = 1 and false = 0.

(b) Write the simplified expression for F as a Sum of Products.

(c) Write the simplified expression for F as a product of Sums.

18. Draw the logic circuit for Table 4.9 using only NOR gates. (GATE EC 1993)

C	B	A	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Table 4.9:

19. Implement the following Boolean function in a 8x1 multiplexer. (GATE EC 1993)

$$Q = BC + ABD' + A'C'D \tag{4.22}$$

20. Minimize the following Boolean function in 4.23.

$$F = A'B'C' + A'BC' + A'BC + ABC' \tag{4.23}$$

21. Find the Boolean expression for Table 4.10. (GATE EC 2005)

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Table 4.10:

22. Minimize the logic function represented by the following Karnaugh map. (CBSE 2021)

X \ YZ	00	01	11	10
0	1	1	1	0
1	0	0	1	0

23. Find the output for the Karnaugh map shown below (GATE EE 2019)

RS \ PQ	00	01	11	10
00	0	1	1	0
01	1	1	1	1
11	1	1	1	1
10	0	0	0	0

24. The propogation delays of the XOR gate, AND gate and multiplexer (MUX) in the circuit shown in the Fig. 4.11 are 4 ns, 2 ns and 1 ns, respectively. If all the inputs P, Q, R, S and T are applied simultaneously and held constant, the maximum propogation delay of the circuit is (Gate EC-2021)

(a) 3 ns

(b) 5 ns

(c) 6 ns

(d) 7 ns

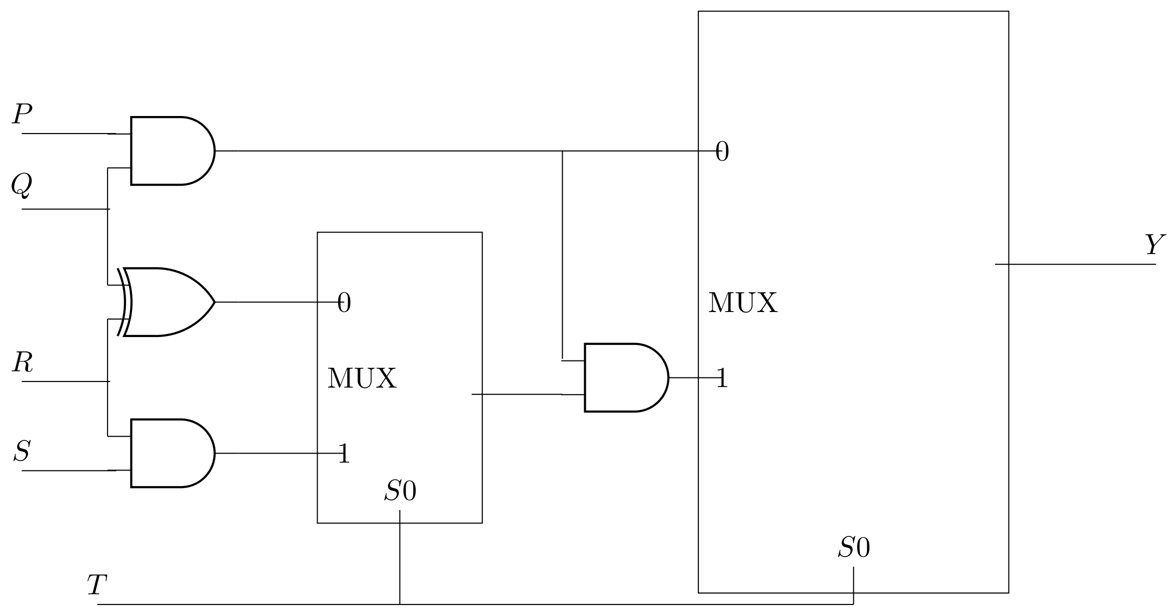


Figure 4.11:

25. Consider the 2-bit multiplexer(MUX) shown in the figure. For output to be the XOR of R and S, the values for W, X, Y and Z are ? (GATE EC-2022)

    (a) $W = 0, X = 0, Y = 1, Z = 1$

    (b) $W = 1, X = 0, Y = 1, Z = 0$

    (c) $W = 0, X = 1, Y = 1, Z = 0$

    (d) $W = 1, X = 1, Y = 0, Z = 0$

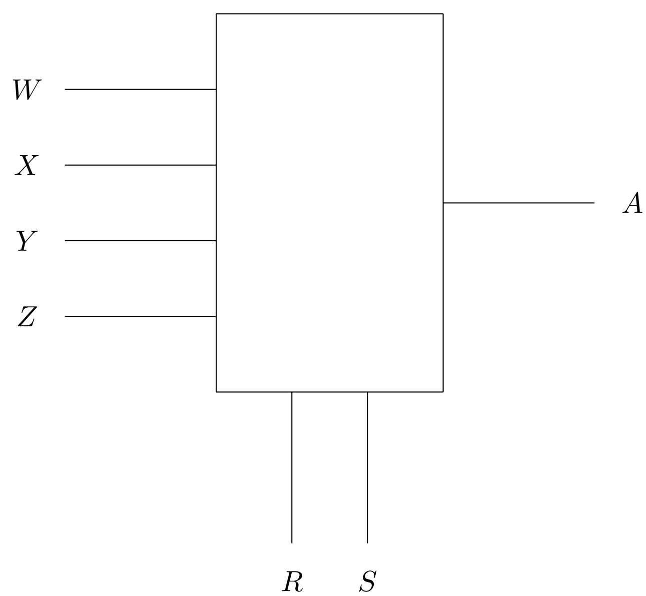


Figure 4.12:

Chapter 5

7474

We show how to use the 7474 D-Flip Flop ICs in a sequential circuit to realize a decade counter.

5.1. Components

Component	Value	Quantity
Breadboard		1
Resistor	$\geq 220\Omega$	1
Arduino	Uno	1
Seven Segment Display	Common Anode	1
Decoder	7447	1
Flip Flop	7474	2
Jumper Wires		20

Table 5.1:

5.2. Decade Counter

1. Generate the CLOCK signal using the **blink** program.

	INPUT				OUTPUT				CLOCK		5V			
	W	X	Y	Z	A	B	C	D						
Arduino	D6	D7	D8	D9	D2	D3	D4	D5	D13					
7474	5	9			2	12			CLK1	CLK2	1	4	10	13
7474			5	9			2	12	CLK1	CLK2	1	4	10	13
7447					7	1	2	6			16			

Table 5.2:

2. Connect the Arduino, 7447 and the two 7474 ICs according to Table 5.2 and Fig. 5.2.

   The pin diagram for 7474 is available in Fig. 5.1

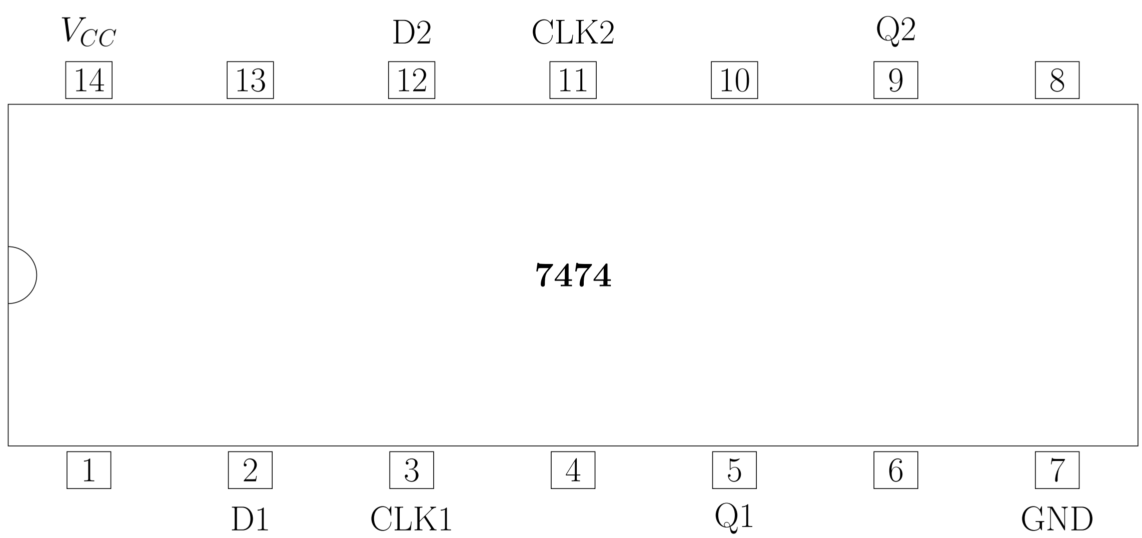


Figure 5.1:

3. Realize the decade counter in Fig. 5.2.

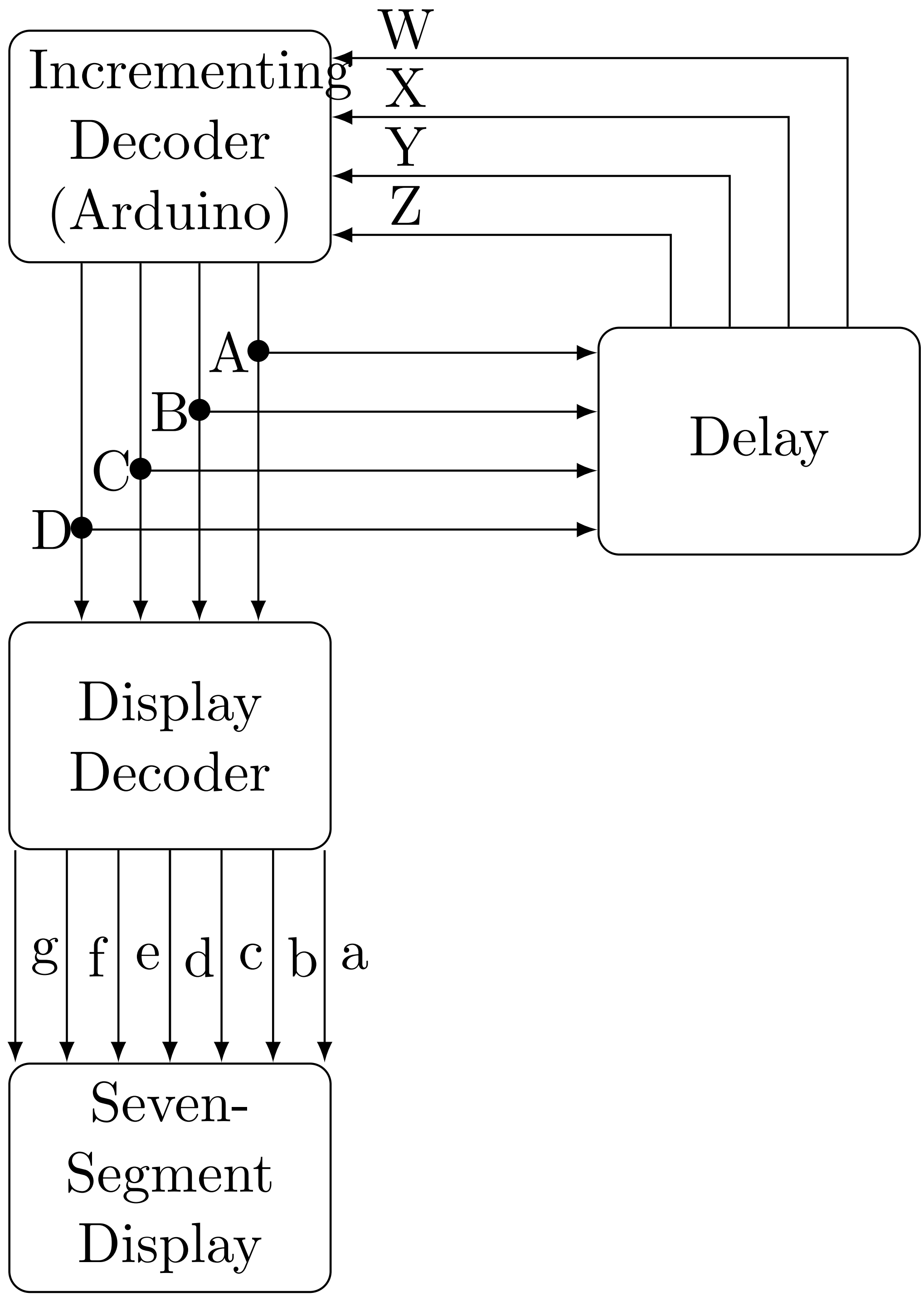

Incrementing
Decoder
(Arduino)
W
X
Y
Z
A
B
C
D
Delay
Display
Decoder
g
f
e
d
c
b
a
Seven-
Segment
Display

Figure 5.2:

Chapter 6

Finite State Machine

We explain a state machine by deconstructing the decade counter

6.1. The Decade Counter

The block diagram of a decade counter (repeatedly counts up from 0 to 9) is available in Fig 5.2 The incrementing decoder and display decoder are part of combinational logic, while the delay is part of sequential logic

6.2. Finite State Machine

1. Fig 6.1 shows a finite state machine (FSM) diagram for the decade counter in Fig 5.2 s_0 is the state when the input to the incrementing decoder is 0 The state transition table for the FSM is Table 4.1, where the present state is denoted by the variables W, X, Y, Z and the next state by A, B, C, D.

2. The FSM implementation is available in Fig 6.2 The flip-flops hold the input for the time that is given by the clock This is nothing but the implementation of the Delay block in Fig 5.2

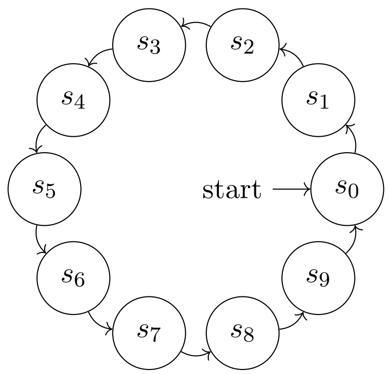


Figure 6.1: FSM for the decade counter

3. The hardware cost of the system is given by

$$\text{No of D Flip-Flops} = \lceil \log_2 (\text{No of States}) \rceil \tag{6.1}$$

   For the FSM in Fig 6.1, the number of states is 9, hence the number flipflops required $= 4$

4. Draw the state transition diagram for a decade down counter (counts from 9 to 0 repeatedly) using an FSM

5. Write the state transition table for the down counter

6. Obtain the state transition equations with and without don't cares

7. Verify your design using an arduino

6.3. Problems

1. The digital circuit shown in Fig. 6.3 generates a modified clockpulse at the output. Sketch the output waveform. (GATE EE 2004)

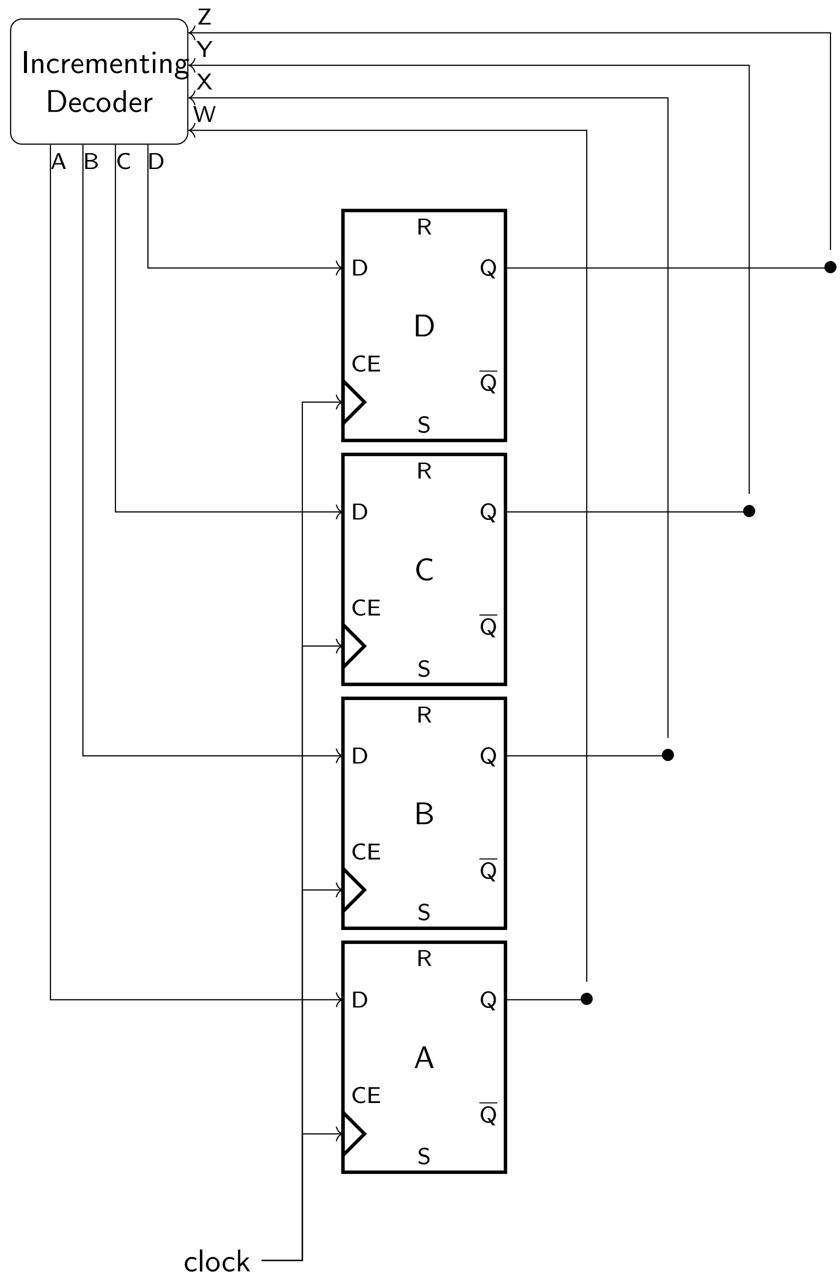


Figure 6.2: Decade counter FSM implementation using D-Flip Flops

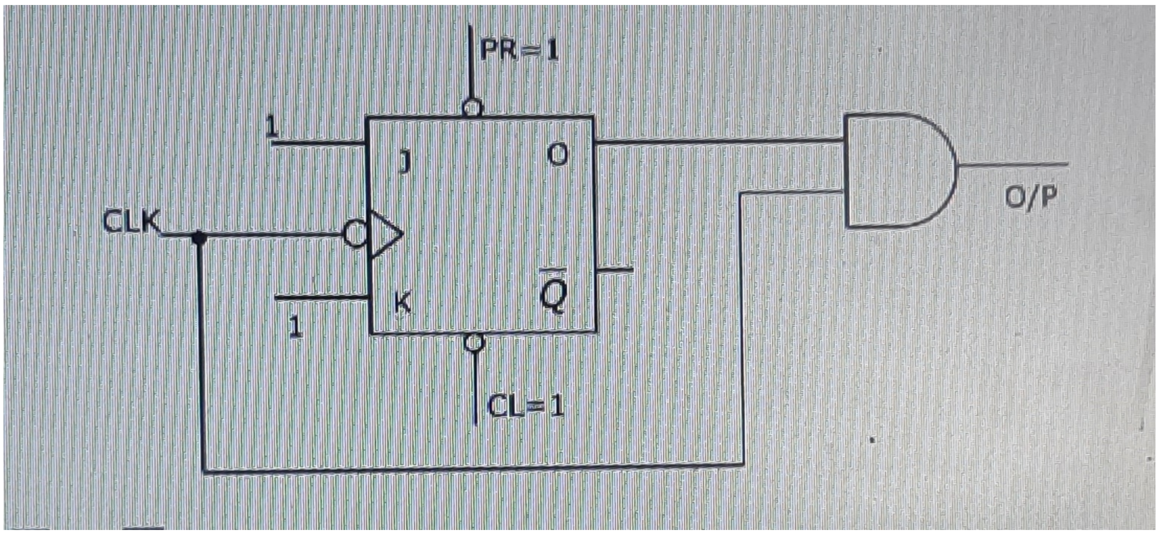


Figure 6.3:

2. The circuit shown in the figure below uses ideal positive edge-triggered synchronous J-K flip flops with outputs X and Y. If the initial state of the output is X=0 and Y=0, just before the arrival of the first clock pulse, the state of the output just before the arrival of the second clock pulse is (GATE IN 2019)

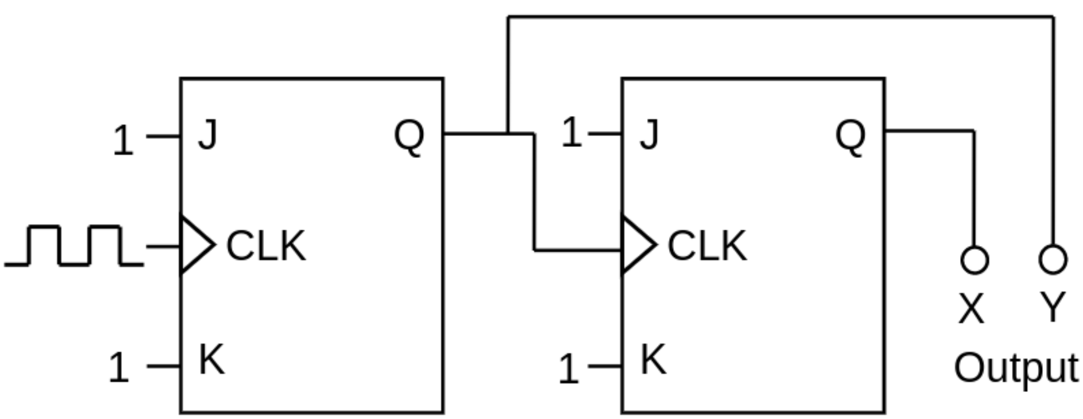


Figure 6.4:

3. The state diagram of a sequence detector is shown in Fig. 6.5 . State S_0 is the initial state of the sequence detector. If the output is 1, then (GATE EC 2020)

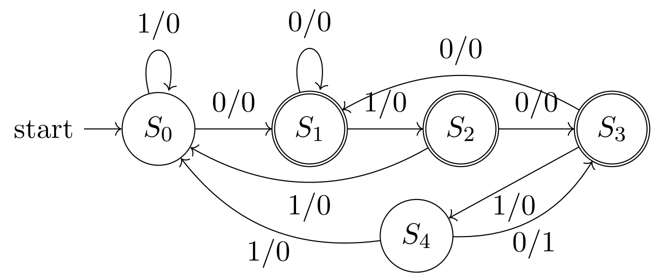


Figure 6.5: State diagram

(a) the sequence 01010 is detected

(b) the sequence 01011 is detected

(c) the sequence 01110 is detected

(d) the sequence 01001 is detected

4. A counter is constructed with three D flip-flops. The input-output pairs are named (D0, Q0), (D1, Q1), and (D2, Q2), where the subscript 0 denotes the least significant bit. The output sequence is desired to be the Gray-code sequence 000, 001, 011, 010, 110, 111, 101, and 100, repeating periodically. Note that the bits are listed in the Q2 Q1 Q0 format. Find the combinational logic expression for D1. (GATE EE 2021)

5. For the circuit shown in Fig. 6.6, the clock frequency is f_0 and the duty cycle is 25%. For the signal at the Q output of the Flip-Flop,

(a) frequency of $\frac{f_0}{4}$ and duty cycle is 50%

(b) frequency of $\frac{f_0}{4}$ and duty cycle is 25%

(c) frequency of $\frac{f_0}{2}$ and duty cycle is 50%

(d) frequency of f_0 and duty cycle is 25%

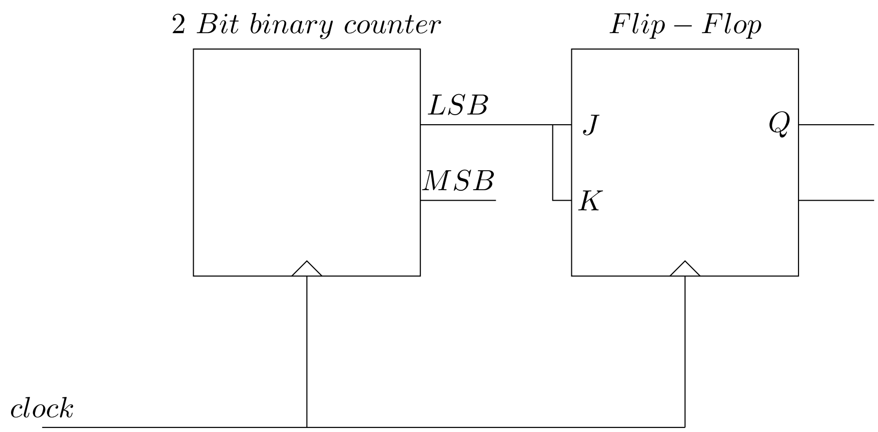


Figure 6.6:

(GATE EC-2022)

Chapter 7

Assembly Programming

T:his manual shows how to setup the assembly programming environment for the arduino.

7.1. Software Installation

1. Find the USB port to which arduino is connected.

```
%Finding the port

sudo dmesg | grep tty
%The output will be something like
[ 6.153362] cdc_acm 1-1.2:1.0: ttyACM0: USB ACM device
%and your port number is ttyACM0
```

2. Copy the .inc file to your home directory

```
cp assembly/setup/m328Pdef/m328Pdef.inc ~/
```

3. Execute

```
avra assembly/setup/codes/hello.asm
```

as

4. Then flash the .hex file

```
hello.hex
```

5. You should see the led beside pin 13 light up.

6. Now edit **hello.asm** by modifying the line to

```
ldi r17,0b00000000
```

Save and execute. The led should turn off.

7. What do the following instructions do?

```
ldi r16,0b00100000
out DDRB,r16
```

Solution: The Atmega328p microcontroller for the arduino board has 32 internal 8-bit registers, R0-R31. R16-R31 can be used directly for i/o. The first instruction loads an 8-bit binary number into R16. The second instruction loads the value in R16 to the DDRB register. Each bit of the DDRB register corresponds to a pin on the arduino. The second instruction declares pin 13 to be an output port. Both the instructions are equivalent to pinMode(13, OUTPUT).

8. What do the following instructions do?

```
ldi r17,0b00100000
out PortB,r17
```

Solution: The instructions are equivalent to digitalWrite(13).

T:he objective of this manual is to show how to control a seven segment display through the AVR-Assembly.

7.2. Seven Segment Display

7.2.0.1. See Table 2.1 for components.

7.2.0.2. Complete Table 7.2.0.2.1 for all the digital pins using Fig. 7.2.0.2.1.

Port Pin	Digital Pin
PD2	2
PB5	13

Table 7.2.0.2.1:

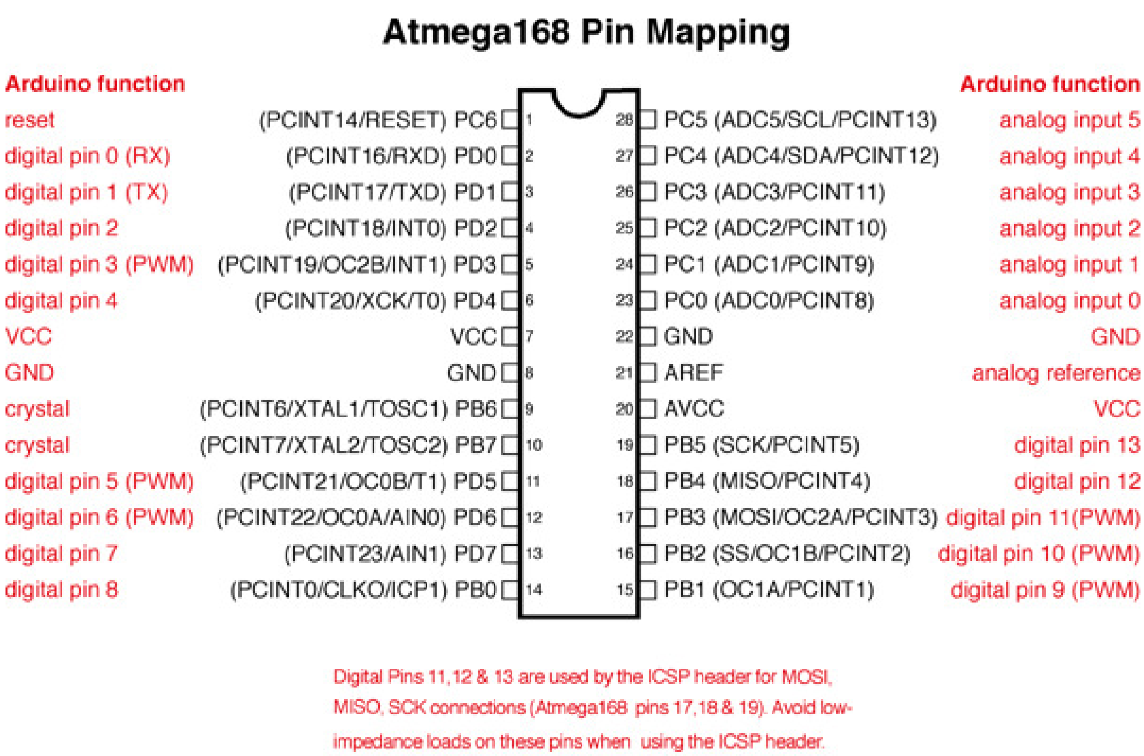


Figure 7.2.0.2.1:

7.2.0.3. Make connections according to Table 7.2.0.3.1.

Arduino	2	3	4	5	6	7	8
	PD2	PD3	PD4	PD5	PD6	PD7	PB0
Display	a	b	c	d	e	f	g
2	0	0	1	0	0	1	0

Table 7.2.0.3.1:

7.2.0.4. Execute the following code. The number 2 should be displayed.

```
;using assembly language for
;displaying number on
;seven segment display

.include "/home/gadepall/m328Pdef.inc"



;Configuring pins 2–7 (PD2–PD7) of Arduino
;as output
  ldi r16,0b11111100
  out DDRD,r16
;Configuring pin 8 (PB0) of Arduino
;as output
  ldi r16,0b00000001
  out DDRB,r16
;Writing the number 2 on the
;seven segment display
  ldi r17,0b10010000
  out PortD,r17
```

```
    ldi r17,0b00000000
    out PortB,r17
Start:
    rjmp Start
```

7.2.0.5. Now generate the numbers 0-9 by modifying the above program.

7.3. 7447

This manual shows how to program the 7447 BCD-Seven segment display decoder through AVR-Assembly.

7.3.1. Components

Component	Value	Quantity
Resistor	220 Ohm	1
Arduino	UNO	1
Seven Segment Display		1
Decoder	7447	1
Jumper Wires	M-M	20
Breadboard		1

7.3.2. Boolean Operations

1. Verify the AND,OR and XOR operations in assembly using the following code and making pin connections according to Table 1.

```
wget https://raw.githubusercontent.com/gadepall/arduino/master/assembly/7447/
    count/codes/and_or_xor.asm
```

7447	D	C	B	A
Arduino	5	4	3	2

Table 1:

2. Suppose R20=0b00000010, R16=0b00000001. Explain the following routine

```
loopw: lsl r16   ;left shift
       dec r20   ;counter --
       brne loopw ;if counter != 0
       ret
```

Solution: The routine shifts R16 by 2 bits to the left (the count in R20=2). At the end of the routine, R16=0b00000100.

3. What do the following instructions do?

```
rcall loopw
out PORTD,r16  ;writing output to pins 2,3,4,5
```

Solution: rcall calls for execution of the **loopw** routine, which shifts R16 by 2 bits to the left and writes R16 to the display through PORTD.

4. Use the following routine for finding the complement of a number.

```
wget https://raw.githubusercontent.com/gadepall/arduino/master/assembly/7447/
    count/codes/complement.asm
```

5. Write an assembly program for implementing the following equations. Note that ZYXW is the input nibble and DCBA is the output nibble. Display DCBA on the seven segment display for each input ZYXW from 0-9.

$$A = W' \tag{5.1}$$

$$B = WX'Z' + W'X \tag{5.2}$$

$$C = WXY' + X'Y + W'Y \tag{5.3}$$

$$D = WXY + W'Z \tag{5.4}$$

6. Repeat the above exercise by getting ZYXW as manual inputs to the arduino from the GND and 5V pins on the breadboard.

T:his manual shows how to program the 7447 BCD-Seven segment display decoder through AVR-Assembly.

7.3.3. Controlling the Display

1. Connect the 7447 IC to the seven segment display.

2. Make connections between the 7447 and the arduino according to Table 2

7447	D	C	B	A
Arduino	5	4	3	2

Table 2:

3. Execute the following program. The number 5 will be displayed.

```
assembly/7447/io/codes/op_7447.asm
```

4. Now generate the numbers 0-9 by modifying the above program.

5. Execute the following program after making the connections in Table 5. The number 3 will be displayed. What does the program do?

```
assembly/7447/io/codes/ip_7447.asm
```

	Z	Y	X	W
Input	0	0	1	1
Arduino	13	12	11	10

Table 5:

Solution: The program reads from pins 10-13 and displays the equivalent decimal value on the display by writing to pins 2-5 of the arduino.

6. Explain the following instructions

```
ldi r17, 0b11000011 ; identifying input pins 10,11,12,13
ldi r17, 0b11111111 ;
out PORTB,r17  ;
in r17,PINB
```

Solution: First define pins 10,11,12 and 13 as input pins. Then ensure that these pins have the input 1 by default. Load the inputs from the pins in port B (which includes pins 10-13) into R17.

7.4. Timer

T:his manual shows how to use the Atmega328p timer to blink the builtin led with a delay.

7.4.1. Components

Component	Value	Quantity
Arduino	UNO	1

7.4.2. Blink through TIMER

1. Connect the Arduino to the computer and execute the following code

```
assembly/timer/codes/timer.asm
```

2. Explain the following instruction

```
sbi DDRB, 5
```

3. What do the following instructions do?

```
ldi r16, 0b00000101
out TCCR0B, r16
```

Solution: The system clock (SYSCLK) frequency of the Atmega328p is 16 MHz. TCCR0B is the Timer Counter Control Register. When

$$TCCR0B = 0b101 \tag{3.1}$$

$$\implies CLK = \frac{SYSCLK}{1024} \tag{3.2}$$

$$= \frac{16M}{1K} = 16kHz. \tag{3.3}$$

4. Explain the PAUSE routine.

```
ldi r19, 0b01000000 ;times to run the loop = 64 for 1 second delay
PAUSE: ;this is delay (function)
lp2: ;loop runs 64 times
            IN r16, TIFR0 ;tifr is timer interupt flag (8 bit timer runs 256
               times)
            ldi r17, 0b00000010
            AND r16, r17 ;need second bit
            BREQ PAUSE
            OUT TIFR0, r17 ;set tifr flag high
      dec r19
      brne lp2
      ret
```

Solution: TIFR0 is the timer interrupt flag and TIFR0=0bxxxxxx10 after every 256 cycles. PAUSE routine waits till TIFR0=0bxxxxxx10, this checking is done by the AND and BREQ instructions above.

5. Explain the lp2 routine.

   Solution: R19 = 64 and is used as a count for lp2. The lp2 routine returns after 64 PAUSE rutines.

6. What is the blinking delay?

Solution: The blinking delay is given by

$$delay = \frac{CLK}{lp2 \times PAUSE} seconds \tag{6.1}$$

$$= \frac{16 \times 1024}{64 \times 256} seconds = 1 second \tag{6.2}$$

7.4.3. Blink through Cycle Delays

1. Connect pin 8 of the Arduino to an led and execute the following code

```
assembly/timer/codes/cycle_delay.asm
```

2. Explain how the delay is obained

```
    ldi r16,0x50
    ldi r17,0x00
    ldi r18,0x00


w0:
    dec r18
    brne w0
    dec r17
    brne w0
    dec r16
    brne w0
    pop r18
    pop r17
    pop r16
```

```
ret
```

Solution: The w0 loop is executed using the counts in R16=2^6+2^4 = 80, R17=R18=2^8 = 256. Thus

$$\begin{aligned} delay &\approx 80 \times 256 \times 256 cycles && (2.1)\\ &= \frac{80 \times 256 \times 256}{2^4 \times 2^2 0} seconds && (2.2)\\ &= 0.3125 seconds && (2.3) \end{aligned}$$

The actual time is slightly more since each instruction takes a few cycles to execute.

3. Should you use timer delay or cycle delay?

   Solution: Timer delay is an accurate method for giving delays. Cycle delay is a crude method and should be avoided.

7.5. Memory

T:his manual shows how to use the Atmega328p internal memory for a decade counter through a loop.

1. Exectute the following code by connecting the Arduino to 7447 through pins 2,3,4,5. The seven segment display should be connected to 7447.

   ```
   assembly/memory/codes/mem.asm
   ```

2. Explain the following instructions

   ```
   ldi xl,0x00
   ```

```
ldi xh,0x01
ldi r16,0b00000000
st x,r16
```

Solution: X=R27:R26, Y=R29:R28, and Z=R31:R30 where R27:R26 represents XH:XL.

The above instructions load 0b00000000 into the memory location X=0x0100.

3. What does the **loop_cnt** routine do?

```
ldi r16,0b00000000
ldi r17,0x09
loop_cnt:
inc r16
inc xl
st x,r16
dec r17
brne loop_cnt
```

Solution: The routine loads the numbers 1-9 in memory locations 0x0101 - 0x0109.

4. Revise your code by using a timer for giving the delay.

Chapter 8

Embedded C

8.1. Blink

T:his manual shows how to control an led using AVR-GCC. AVR-GCC is a C compiler for the Atmega328p.

8.1.1. Components

Component	Value	Quantity
Arduino	UNO	1

8.1.2. Blink

1. Execute the following

```
cd avr-gcc/setup/codes


make
```

2. Now open **main.c**. Explain the following lines.

```
    PORTB = ((0 << PB5));
```

```
        _delay_ms(500);
//turn led on
    PORTB = ((1 << PB5));
    _delay_ms(500);
```

Solution: $((0 << PB5))$ writes 0 to pin 13 (PB5). _delay_ms(500) introduces a delay of 500 ms.

3. Modify the above code to keep the led on.

4. Repeat the above exercise to keep the led off.

T:his manual shows how to control a seven segment display using AVR-GCC with arduino

8.2. Display Control

1. Connect the arduino to the seven segment display

2. Execute the following code

```
avr−gcc/sevenseg/codes/main.c
```

3. Modify the above code to generate numbers between 0-9.

T:his manual shows how to control a seven segment display using AVR-GCC with arduino

8.3. Input

1. Connect the arduino to the seven segment display through 7447.

2. Execute the following code

```
avr−gcc/input/codes/main.c
```

3. Modify the above code to work without the 7447.

8.4. GCC-Assembly

T:his manual shows how write a function in assembly and call it in a C program while programming the ATMega328P microcontroller in the Arduino. This is done by controlling an LED.

8.4.1. Components

Component	Value	Quantity
Breadboard		1
Resistor	$\geq 220\Omega$	1
Arduino	Uno	1
Seven Segment Display	Common Anode	1
Jumper Wires		10

Table 3:

8.4.2. GCC with Assembly

1. Execute

```
cd avr−gcc/gcc−assembly/codes
make
```

2. Modify **main.c** and **Makefile** to turn the builtin led on.

3. Repeat the above exercise to turn the LED off.

4. Explain how the **disp_led(0)** function is related to **Register R24** in **disp_led** routine in **displedasm.S**. **Solution:** The function argument 0 in **disp_led(0)** is passed on to R24 in the assembly routine for further operations. Also, the registers R18-R24 are available for storing more function arguments according to the Table 4. More details are avilable in official ATMEL AT1886 reference.

Register	r19	r18	r21	r20	r23	r22	r25	r24
Function Argument	b7	b6	b5	b4	b3	b2	b1	b0

Table 4: Relationship between Register in assembly and function argument in C

5. Write an assembly routine for controlling the seven segment display and call it in a C program.

6. Build a decade counter with **main.c** calling all functions from assembly routines.

8.5. LCD

T:his manual shows how to interface an Arduino to a 16×2 LCD display using AVR-GCC. This framework provides a useful platform for displaying the output of AVR-Assembly programs.

Component	Value	Quantity
Breadboard		1
Arduino	Uno	1
LCD	16×2	1
Jumper Wires		20

Table 6:

8.5.1. Components

8.5.2. Display Number on LCD

8.5.2.1. Plug the LCD in Fig. 8.5.2.2.1 to the breadboard.

8.5.2.2. Connect the Arduino pins to LCD pins as per Table 8.5.2.2.1.

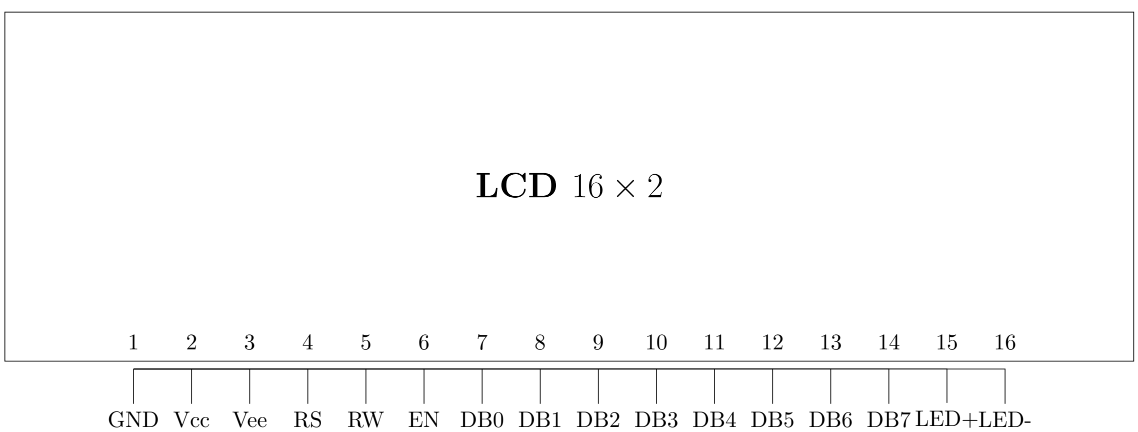


Figure 8.5.2.2.1: LCD

8.5.2.3. Execute

```
cd avr-gcc/lcd/codes
make
```

Table 8.5.2.2.1: Arduino to LCD Pin Connection.

Arduino Pins	**LCD Pins**	**LCD Pin Label**	**LCD Pin Description**
GND	1	GND	
5V	2	Vcc	
GND	3	Vee	Contrast
D8	4	RS	Register Select
GND	5	R/W	Read/Write
D9	6	EN	Enable
D10	11	DB4	Serial Connection
D11	12	DB5	Serial Connection
D12	13	DB6	Serial Connection
D13	14	DB7	Serial Connection
5V	15	LED+	Backlight
GND	16	LED-	Backlight

8.5.2.4. Modify the above code to display a string.

8.5.2.5. Modify the above code to obtain a decade counter so that the numbers from 0 to 9 are displayed on the lcd repeatedly.

8.5.2.6. Repeat the above exercies to display a string on the first line and a number on the second line of the lcd.

8.5.2.7. Write assembly routines for driving the lcd.

Chapter 9

Vaman-ESP32

9.1. Software

All codes used in this document are available in the following directory

```
vaman/esp32/codes
```

9.2. Flash Vaman-ESP32 using Arduino

9.2.1. Do not power any devices. Make connections as shown in Table 9.2.1.1 and Fig. 9.2.1.1.

The Vaman pin diagram is available in Fig. 9.2.1.2

VAMAN LC PINS	ARDUINO PINS
3.3	3.3
GND	GND
TXD0	TXD
RXD0	RXD
0	GND
EN	GND

Table 9.2.1.1:

9.2.2. For compiling and genrating the bin file

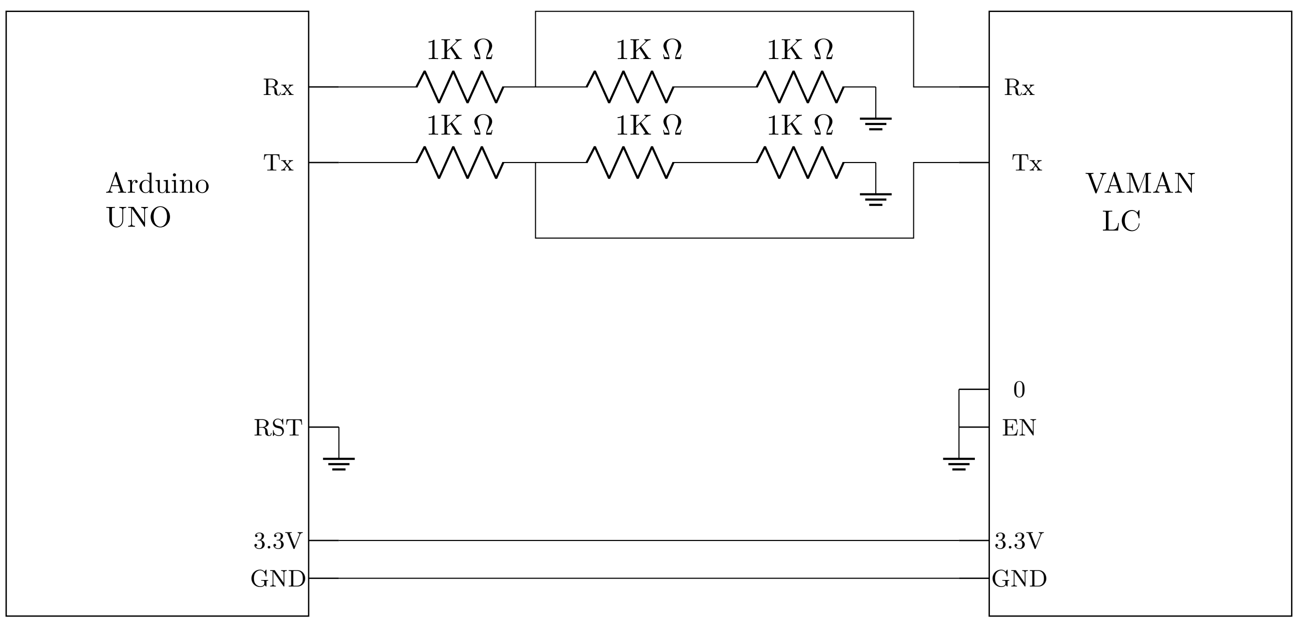


Figure 9.2.1.1: Circuit Connections

```
cd vaman/esp32/codes/ide/blink
pio run
```

9.2.3. make sure that platformio.ini file contains these lines

```
[env:esp32doit-devkit-v1]
platform = espressif32
board = esp32doit-devkit-v1
framework = arduino
platform_packages = toolchain-xtensa-esp32@https://github.com/esphome/
    esphome-docker-base/releases/download/v1.4.0/toolchain-xtensa32.tar.gz
framework-arduinoespressif32@<3.10006.210326
```

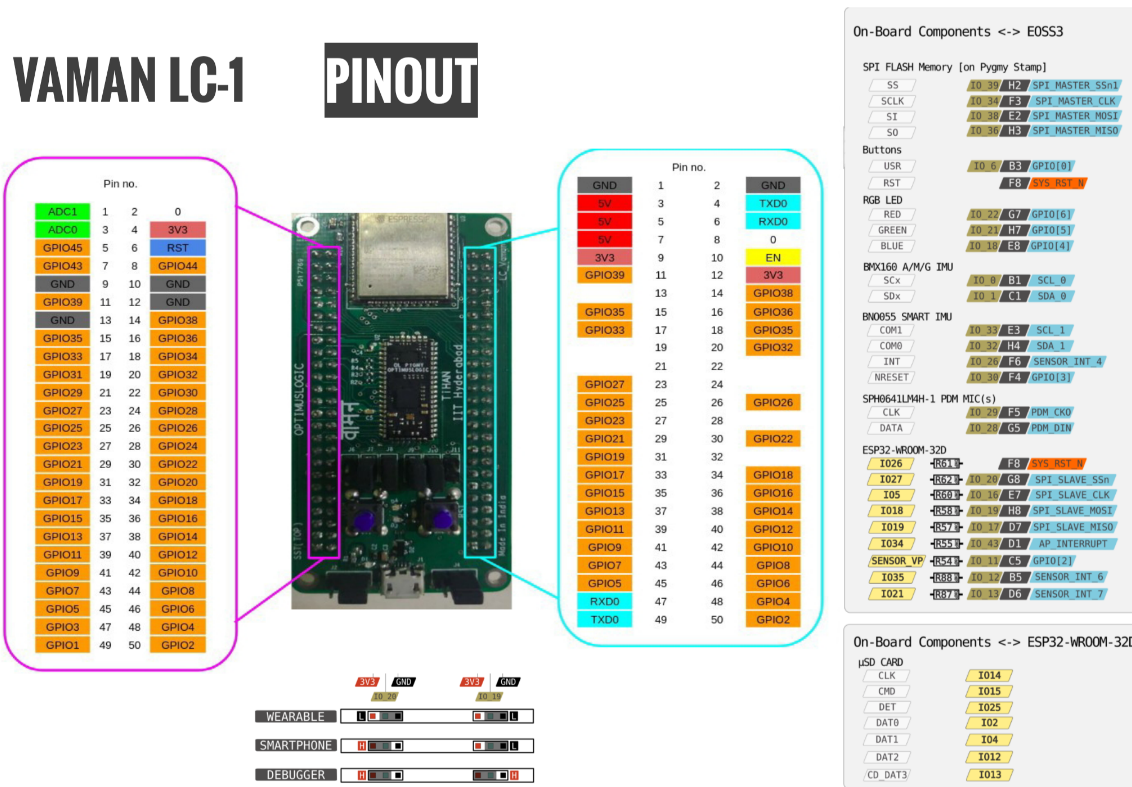


Figure 9.2.1.2: Pin diagram

9.2.4. For uploading bin file to Vaman through ArduinoDroid application

1. Open the Droid Application
2. Click the three dots **in** the top right corner
3. Navigate to Settings —> Board Type
4. Select ESP32 —> DOIT ESP32 DEVKIT V1
5. Change the upload speed to 115200
6. Upload the generated .bin file

While the dots are printed on the screen, disconnect the EN wire from GND. Make sure that the Vaman board is not powering any device while flashing. The Vaman-ESP should now flash.

9.2.5. After flashing, disconnect pin 0 on Vaman-ESP from GND. Power on Vaman and the appropriate LED will blink.

9.2.6. Open

```
vaman/esp32/codes/ide/blink/src/main.cpp
```

and change the delay to

```
delay(2000);
```

and exectute the code by following the steps above.

9.3. OTA

9.3.1. Flash the following code through USB-UART.

```
vaman/esp32/codes/ide/ota/setup
```

after entering your wifi username and password (in quotes below)

```
#define STASSID "..." // Add your network credentials
#define STAPSK "..."
```

in src/main.cpp file

9.3.2. You should be able to find the ip address of your vaman-esp using

```
ifconfig
nmap –sn 192.168.231.1/24
```

where your computer's ip address is the output of ifconfig and given by 192.168.231.x

9.3.3. Assuming that the username is gvv and password is abcd, flash the following code wirelessly

```
vaman/esp32/codes/ide/blink
```

through

```
pio run
pio run -t nobuild -t upload --upload-port 192.168.231.245
```

where you may replace the above ip address with the ip address of your vaman-esp.

9.3.4. Connect pin 2 to an LED to see it blinking.

9.4. Onboard LED

9.4.1. Connect the pins between Vaman-ESP32 and Vaman-PYGMY as per Table 9.4.1.1

ESP32	**Vaman**
GPIO2	GPIO18
GPIO4	GPIO21
GPIO5	GPIO22

Table 9.4.1.1:

9.4.2. Flash the following code OTA

```
vaman/esp32/codes/ide/ota/blinkt
```

You should see the onboard green LED blinking.

9.4.3. Change the blink duration to 100 ms.

Chapter 10

Vaman-FPGA

10.1. Setup

T:his document provides a simple introduction to software and hardware using the Vaman FPGA/microcontroller board. The exercises provided here are suitable for students from primary school till college.

10.1.1. Software

The codes are available at

```
vaman/fpga/setup/codes/
```

10.1.2. Setup

10.1.2.1. Follow the instructions in

```
vaman/installation/termuxdebian/termux_debian_fpga.txt
```

10.1.3. Frequency

10.1.3.1. In the following verilog program,

```
codes/blink/helloworldfpga.v
```

pay attention to the following lines

```
delay = delay+1;
if(delay > 20000000)
begin
delay=27'b0;
led=!led;
end
```

It may be deduced from the above that the blink frequency is 20 MHz.

10.1.3.2. In instruction 10.1.3.1, replace

```
if(delay > 20000000)
```

with

```
if(delay==27'b1001100010010110100000000)
```

and execute the verilog code.

10.1.3.3. Since the delay is 20 MHz, the blink period is 1 second. Modify the verilog code so that the blink period becomes 0.5s.

10.1.3.4. Find the bit length of 20 MHz.

Type	Vaman Pin	Connection
Input	IO_28	GND
Output	IO_11	LED

Table 10.1.3.8.1: Vaman Input/Output.

Solution:

$$\log_2 (20000000) \approx 27 \quad (10.1.3.4.1)$$

10.1.3.5. Obtain the above answer using a Python code.

Solution: Exeucte the following code and compare with instruction 10.1.3.2.

```
codes/blink/freq_count.py
```

10.1.3.6. Replace the following line in the code in instruciton 10.1.3.1

```
assign redled = led; //If you want to change led colour to red,
```

with

```
assign blueled = led;
```

and execute the code.

10.1.3.7. Ensure that the LED stays on in green colour.

Solution: Execute the following code

```
vaman/setup/codes/blink/onoff.v
```

10.1.3.8. Using Table 10.1.3.8.1 and Fig. 9.2.1.2, control the onboard LED through an external input. Connect an external LED and control it using an output pin as well.

Solution: Execute the following code and take out the input pin connect to GND. Plug it again. Do this repeatedly.

```
vaman/setup/codes/input/blink_ip.v
vaman/setup/codes/input/pygmy.pcf
```

10.2. Seven Segment Display

W:e show how to use Vaman as a decade counter.

10.2.1. Software

All codes used in this manual are available at the following link.

```
vaman/fpga/sevenseg/codes
```

10.2.2. Setup

10.2.2.1. The pin sheet for the Vaman is available in Fig. 9.2.1.2. Connect the pins in the bank J5 of the Vaman with the seven segment display shown in Fig. 2.2 according to Table 10.2.2.1.1. Ensure that the COM pin is connected to 3.3V through a resistor.

10.2.2.2. Now execute the following code.

```
codes/static/sevenseg.v
```

Flash the helloworldfpga.bin file to Vaman. You should see the number 5 displayed. The following lines are used for generating numbers on the display.

Display	Pygmy
a	IO_4
b	IO_5
c	IO_6
d	IO_7
e	IO_8
f	IO_10
g	IO_11
COM	3.3 V

Table 10.2.2.1.1: Seven segment display - Vaman connection.

```
assign a=0;
assign b=1;
assign c=0;
assign d=0;
assign e=1;
assign f=0;
assign g=0;
```

10.2.2.3. Modify the above code appropriately with the help of Table 10.2.2.3.1 and Fig. 2.3 to generate the numbers from 0-9 on the display.

a	b	c	d	e	f	g	decimal
1	0	0	1	1	1	1	1
0	0	1	0	0	1	0	2

Table 10.2.2.3.1: Pin values used for generating decimal numbers on the seven segment display.

10.2.3. Examples

10.2.3.1. Table 10.2.2.1.1 and the PU 64 table in Fig. 10.2.3.2.1 explain the pin numbering in the following file.

```
codes/static/Vaman.pcf
```

10.2.3.2. Execute the code below. All the pins in the display are controlled using a 7 bit word.

```
codes/static/sevenseg_word.v
```

The above file is used for generating the number 4 on the display. The process is explained by the completion of Table 10.2.2.3.1.

```
gpio_out=7'b0100100;
```

10.2.3.3. Use a verilog function that takes a decimal input and display it on the seven segement display.

Solution: Execute the following code.

```
codes/static/sevenseg_dec.v
```

10.2.3.4. Use the Vaman as a decade counter.

Solution: Execute the following code.

codes/loop/decade_counter.v

10.3. Boolean Logic

I:n this document we show how to design a decade counter using Vaman and boolean logic.

10.3.1. Software

All codes used in this manual are available at the following link.

vaman/fpga/boolean/codes

10.3.2. Setup

10.3.2.1. Fig. 9.2.1.2 shows the pin diagram for the Vaman. Using the bank J5, connect the pins of the seven segment display in Fig. 2.2 to the Vaman according to Table 10.3.2.1.1. Make sure that the COM pin is connected to 3.3V through a resistor.

10.3.2.2. Implement Table 10.3.2.2.1 using the Vaman and the display.

Solution: In Table 10.3.2.2.1, the output variables a, b, c, d, e, f, g can be expressed

Display	Pygmy
a	IO_4
b	IO_5
c	IO_6
d	IO_7
e	IO_8
f	IO_10
g	IO_11
COM	3.3 V

Input Variable	**Pin**
W	IO_28
X	IO_23
Y	IO_31
Z	IO_12

Table 10.3.2.1.1: Pin connections between Vaman and the display.

in terms of the input variables W, X, Y, Z as

$$a = WX'Y'Z' + W'X'YZ' \tag{10.3.2.2.1}$$

$$b = WX'YZ' + W'XYZ' \tag{10.3.2.2.2}$$

$$c = Z'Y'XW' \tag{10.3.2.2.3}$$

$$d = WX'Y'Z' + W'X'YZ' + WXYZ' + WX'Y'Z \tag{10.3.2.2.4}$$

$$e = WX'Y'Z' + WXY'Z' + W'X'YZ' + WX'YZ' + WXYZ' + WX'Y'Z \tag{10.3.2.2.5}$$

$$f = WX'Y'Z' + W'XY'Z' + WXY'Z' + WXYZ' \tag{10.3.2.2.6}$$

$$g = W'X'Y'Z' + WX'Y'Z' + WXYZ' \tag{10.3.2.2.7}$$

Execute the following program.

```
vaman/fpga/boolean/codes/decoders/dispdec.v
vaman/fpga/boolean/codes/decoders/Vaman.pcf
```

Connect W, X, Y, Z to GND. For different values of the input variables, verify the output in on the display using Table 10.3.2.2.1.

Z	Y	X	W	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**Decimal**
0	0	0	0	**0**	**0**	**0**	**0**	**0**	**0**	**1**	0
0	0	0	1	**1**	**0**	**0**	**1**	**1**	**1**	**1**	1
0	0	1	0	**0**	**0**	**1**	**0**	**0**	**1**	**0**	2
0	0	1	1	**0**	**0**	**0**	**0**	**1**	**1**	**0**	3
0	1	0	0	**1**	**0**	**0**	**1**	**1**	**0**	**0**	4
0	1	0	1	**0**	**1**	**0**	**0**	**1**	**0**	**0**	5
0	1	1	0	**0**	**1**	**0**	**0**	**0**	**0**	**0**	6
0	1	1	1	**0**	**0**	**0**	**1**	**1**	**1**	**1**	7
1	0	0	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	8
1	0	0	1	**0**	**0**	**0**	**1**	**1**	**0**	**0**	9

Table 10.3.2.2.1: Truth table for the display decoder.

10.3.2.3. Table 10.3.2.3.1 describes the properties of the incrementing decoder. Using Boolean logic, express A, B, C, D in terms of W, X, Y, Z. Subsequently, implement this decoder by implementing the the expressions so obtained in the Vaman using verilog.

Solution: The following equations contain the desired expressions.

$$A = W'X'Y'Z' + W'XY'Z' + W'X'YZ' + W'X'Y'Z \tag{10.3.2.3.1}$$

$$B = WX'Y'Z' + W'XY'Z' + WX'YZ' + W'XYZ' \tag{10.3.2.3.2}$$

$$C = WXY'Z' + W'X'YZ' + WX'YZ' + W'XYZ' \tag{10.3.2.3.3}$$

$$D = WXYZ' + W'X'Y'Z \tag{10.3.2.3.4}$$

Execute the following code. The next number should be displayed.

```
vaman/fpga/boolean/codes/decoders/incdec.v
```

Z	Y	X	W	**D**	**C**	**B**	**A**
0	0	0	0	**0**	**0**	**0**	**1**
0	0	0	1	**0**	**0**	**1**	**0**
0	0	1	0	**0**	**0**	**1**	**1**
0	0	1	1	**0**	**1**	**0**	**0**
0	1	0	0	**0**	**1**	**0**	**1**
0	1	0	1	**0**	**1**	**1**	**0**
0	1	1	0	**0**	**1**	**1**	**1**
0	1	1	1	**1**	**0**	**0**	**0**
1	0	0	0	**1**	**0**	**0**	**1**
1	0	0	1	**0**	**0**	**0**	**0**

Table 10.3.2.3.1: Truth table for the incrementing decoder.

10.3.3. Decade Counter

10.3.3.1. Using Fig. 5.2 and modifying the code in Problem 10.3.2.3, design the decade counter.

10.3.3.2. Design and implement the down counter.

10.4. LCD

T:his manual shows how to interface a 16×2 LCD display to Vaman and verilog code for addition of two numbers and display the output on the LCD display.

10.4.1. Display the addition of two numbers on LCD

10.4.1.1. Plug the LCD in Fig. 8.5.2.2.1 to the breadboard.

10.4.1.2. Connect the Vaman Pygmy pins to LCD pins as per Table 10.4.1.2.1.

10.4.1.3. The below code is for displaying the output of addition of two numbers

Pygmy pins	**LCD Pins**	**LCD Pin Label**	**LCD Pin Description**
GND	1	GND	
5V	2	Vcc	
GND	3	Vee	Contrast
10	4	RS	Register Select
GND	5	R/W	Read/Write
9	6	EN	Enable
14	11	DB4	Serial Connection
13	12	DB5	Serial Connection
12	13	DB6	Serial Connection
11	14	DB7	Serial Connection
5V	15	LED+	Backlight
GND	16	LED-	Backlight

Table 10.4.1.2.1: Pin connections between Vaman and the display.

10.4.1.4. Now execute the following code.

```
cd vaman/fpga/lcd/codes/lcd.v
```

10.4.1.5. Flash the helloworldfpga.bin file to Vaman. You should see the result of additon of two numbers

10.4.1.6. Modify the above code to obtaing addition of different numbers of two digits.

10.4.1.7. Repeat the above exercies to add three digit numbers and display the output.

10.4.1.8. Write verilog code for different arithmetic operations.

PD64		
IO Locatio	Alias	IO Type
B1	IO_0	BIDIR
C1	IO_1	BIDIR
A1	IO_2	BIDIR
A2	IO_3	BIDIR
B2	IO_4	BIDIR
C3	IO_5	BIDIR
B3	IO_6	BIDIR
A3	IO_7	BIDIR/CLOCK
C4	IO_8	BIDIR/CLOCK
B4	IO_9	BIDIR
A4	IO_10	BIDIR
C5	IO_11	BIDIR
B5	IO_12	BIDIR
D6	IO_13	BIDIR
A5	IO_14	BIDIR
C6	IO_15	BIDIR
E7	IO_16	BIDIR
D7	IO_17	BIDIR
E8	IO_18	BIDIR
H8	IO_19	BIDIR
G8	IO_20	BIDIR
H7	IO_21	BIDIR
G7	IO_22	BIDIR/CLOCK
H6	IO_23	BIDIR/CLOCK
G6	IO_24	BIDIR/CLOCK
F7	IO_25	BIDIR
F6	IO_26	BIDIR
H5	IO_27	BIDIR
G5	IO_28	BIDIR
F5	IO_29	BIDIR
F4	IO_30	BIDIR
G4	IO_31	BIDIR
H4	IO_32	SDIOMUX
E3	IO_33	SDIOMUX
F3	IO_34	SDIOMUX
F2	IO_35	SDIOMUX
H3	IO_36	SDIOMUX
G2	IO_37	SDIOMUX
E2	IO_38	SDIOMUX
H2	IO_39	SDIOMUX
D2	IO_40	SDIOMUX
F1	IO_41	SDIOMUX
H1	IO_42	SDIOMUX
D1	IO_43	SDIOMUX
E1	IO_44	SDIOMUX
G1	IO_45	SDIOMUX

PU64		
IO Locatio	Alias	IO type
4	IO_0	BIDIR
5	IO_1	BIDIR
6	IO_2	BIDIR
2	IO_3	BIDIR
3	IO_4	BIDIR
64	IO_5	BIDIR
62	IO_6	BIDIR
63	IO_7	BIDIR/CLOCK
61	IO_8	BIDIR/CLOCK
60	IO_9	BIDIR
59	IO_10	BIDIR
57	IO_11	BIDIR
56	IO_12	BIDIR
55	IO_13	BIDIR
54	IO_14	BIDIR
53	IO_15	BIDIR
40	IO_16	BIDIR
42	IO_17	BIDIR
38	IO_18	BIDIR
36	IO_19	BIDIR
37	IO_20	BIDIR
39	IO_21	BIDIR
34	IO_22	BIDIR/CLOCK
33	IO_23	BIDIR/CLOCK
32	IO_24	BIDIR/CLOCK
31	IO_25	BIDIR
30	IO_26	BIDIR
28	IO_27	BIDIR
27	IO_28	BIDIR
26	IO_29	BIDIR
25	IO_30	BIDIR
23	IO_31	BIDIR
22	IO_32	SDIOMUX
21	IO_33	SDIOMUX
20	IO_34	SDIOMUX
18	IO_35	SDIOMUX
17	IO_36	SDIOMUX
15	IO_37	SDIOMUX
16	IO_38	SDIOMUX
11	IO_39	SDIOMUX
13	IO_40	SDIOMUX
14	IO_41	SDIOMUX
10	IO_42	SDIOMUX
7	IO_43	SDIOMUX
8	IO_44	SDIOMUX
9	IO_45	SDIOMUX

WR42		
IO Locatio	Alias	IO Type
A7	IO_0	BIDIR
B7	IO_1	BIDIR
C7	IO_3	BIDIR
A6	IO_6	BIDIR
B6	IO_8	BIDIR/CLOCK
A5	IO_9	BIDIR
B5	IO_10	BIDIR
A4	IO_14	BIDIR
B4	IO_15	BIDIR
E1	IO_16	BIDIR
D1	IO_17	BIDIR
C1	IO_19	BIDIR
F2	IO_20	BIDIR
E2	IO_23	BIDIR/CLOCK
D2	IO_24	BIDIR/CLOCK
D3	IO_25	BIDIR
F3	IO_28	BIDIR
E3	IO_29	BIDIR
F4	IO_30	BIDIR
E4	IO_31	BIDIR
D5	IO_34	SDIOMUX
F5	IO_36	SDIOMUX
E6	IO_38	SDIOMUX
F6	IO_39	SDIOMUX
D7	IO_43	SDIOMUX
E7	IO_44	SDIOMUX
F7	IO_45	SDIOMUX

Figure 10.2.3.2.1: Pin Definitions

Chapter 11

Vaman-ARM

11.1. Setup

11.1.1. Software

All codes used in this document are available at

```
vaman/arm/codes/setup
```

11.1.2. Setup

11.1.2.1. Follow the instructions at

```
vaman/installation/termuxdebian/termuxdebian_arm.txt
```

11.1.3. Delay

11.1.3.1. See the following lines of the code below

```
codes/setup/blink/src/main.c
```

```
PyHal_Set_GPIO(18,1);//blue
PyHal_Set_GPIO(21,1);//green
PyHal_Set_GPIO(22,1);//red
    HAL_DelayUSec(2000000);
PyHal_Set_GPIO(18,0);
PyHal_Set_GPIO(21,0);
PyHal_Set_GPIO(22,0);
```

We may conclude that the blink delay is 2000 000us = 2 s.

11.1.3.2. Replace the following line in 11.1.3.1

```
    HAL_DelayUSec(2000000);
```

with

```
    HAL_DelayUSec(1000000);
```

and execute. Can you see any difference in the blink period?

11.1.3.3. To obtain red colour, execute the following code.

```
vaman/arm/codes/setup/red/src/main.c
```

Now obtain blue colour.

11.1.3.4. Now obtain green colour without blink.

Solution: Execute the following code.

```
vaman/arm/codes/setup/onoff/src/main.c
```

Type	Pin	Destination
Input	IO_5	GND

Table 11.1.3.5.1: Vaman control through external input.

11.1.3.5. Using Table 11.1.3.5.1 and Fig. 9.2.1.2, use an input pin to control the onboard LED.

Solution: Execute the following code. You should see the LED blinking pink. Disconnecting the wire from GND will result in the LED blinking white and green alternately.

```
vaman/arm/codes/setup/gpio/src/main.c
```

11.2. Seven Segment Display

T:his document shows how to implement a decade counter using arm-gcc on Vaman.

11.2.1. Software

All codes used in this document are available at the following link

```
https://github.com/gadepall/vaman/tree/master/arm/vaman/arm/codes/sevenseg/
```

11.2.2. Setup

11.2.2.1. Fig.9.2.1.2 shows all the pin banks of the Vaman. Connect the pins of the display in Fig. 2.2 to bank J5 of the Vaman using Table 11.2.2.1.1. The COM pin should be connected to 3.3V through a resistor.

11.2.2.2. Now execute the following code

Display	Vaman
a	IO_4
b	IO_5
c	IO_6
d	IO_7
e	IO_8
f	IO_10
g	IO_11
COM	3.3 V

Table 11.2.2.1.1: Display-Vaman connection.

```
vaman/arm/codes/sevenseg/static/src/main.c
```

Flash static.bin obtained upon execution of the above code to the Vaman. You should see the number 7 on the display. The following function generates this number.

```
sevenseg(0,0,0,1,1,1,1);
void sevenseg(int a, int b, int c, int d, int e, int f, int g)
{
    //Seven Segment GPIO
    PyHal_GPIO_Set(4,a);//a
    PyHal_GPIO_Set(5,b);//b
    PyHal_GPIO_Set(6,c);//c
    PyHal_GPIO_Set(7,d);//d
```

```
    PyHal_GPIO_Set(8,e);//e
    PyHal_GPIO_Set(10,f);//f
    PyHal_GPIO_Set(11,g);//g
}
}
```

11.2.2.3. Modify the above program using Table 11.2.2.3.1 and Fig. 2.3 to display 0-9.

a	b	c	d	e	f	g	decimal
1	0	0	1	1	1	1	1
0	0	1	0	0	1	0	2

Table 11.2.2.3.1: Decimal number generation on the display.

11.2.3. Examples

11.2.3.1. Table 11.2.2.1.1 and PU 64 Table in Fig. 10.2.3.2.1 show how to use the pins of the Vaman to drive the seven segment display.

11.2.3.2. Use a function taking decimal input in the code in 11.2.2.2 to generate numbers on the display.

Solution: Execute the following file.

```
vaman/arm/codes/sevenseg/decimal/main.c
```

11.2.3.3. Program the Vaman to function as a decade counter.

Solution: Execute the following code.

vaman/arm/codes/sevenseg/loop/main.c

11.3. FSM

T:his document shows how to use the Vaman to design a decade counter using a finite state machine (FSM).

11.3.1. Software

All codes in this document are available at the following links.

https://github.com/gadepall/vaman/tree/master/arm/codes/decoders
https://github.com/gadepall/vaman/tree/master/arm/codes/fsm

11.3.2. Setup

11.3.2.1. Execute Table 11.3.2.1.1 using the Vaman and a seven segment display.

Solution: The outputs a, b, c, d, e, f, g in Table 11.3.2.1.1 are expressed in terms of

D	C	B	A	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**Decimal**
0	0	0	0	**0**	**0**	**0**	**0**	**0**	**0**	**1**	0
0	0	0	1	**1**	**0**	**0**	**1**	**1**	**1**	**1**	1
0	0	1	0	**0**	**0**	**1**	**0**	**0**	**1**	**0**	2
0	0	1	1	**0**	**0**	**0**	**0**	**1**	**1**	**0**	3
0	1	0	0	**1**	**0**	**0**	**1**	**1**	**0**	**0**	4
0	1	0	1	**0**	**1**	**0**	**0**	**1**	**0**	**0**	5
0	1	1	0	**0**	**1**	**0**	**0**	**0**	**0**	**0**	6
0	1	1	1	**0**	**0**	**0**	**1**	**1**	**1**	**1**	7
1	0	0	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	8
1	0	0	1	**0**	**0**	**0**	**1**	**1**	**0**	**0**	9

Table 11.3.2.1.1: Truth table for the display decoder

the inputs A, B, C, D through the following equations.

$$a = AB'C'D' + A'B'CD' \tag{11.3.2.1.1}$$

$$b = AB'CD' + A'BCD' \tag{11.3.2.1.2}$$

$$c = D'C'BA' \tag{11.3.2.1.3}$$

$$d = AB'C'D' + A'B'CD' + ABCD' + AB'C'D \tag{11.3.2.1.4}$$

$$\begin{aligned} e = {} & AB'C'D' + ABC'D' + A'B'CD' + AB'CD' \\ & + ABCD' + AB'C'D \end{aligned} \tag{11.3.2.1.5}$$

$$f = AB'C'D' + A'BC'D' + ABC'D' + ABCD' \tag{11.3.2.1.6}$$

$$g = A'B'C'D' + AB'C'D' + ABCD' \tag{11.3.2.1.7}$$

Now execute the following code.

```
codes/decoders/dispdec/main.c
```

For different values of A, B, C, D, execute the above code to verify Table 11.3.2.1.1.

Z	Y	X	W	**D**	**C**	**B**	**A**
0	0	0	0	**0**	**0**	**0**	**1**
0	0	0	1	**0**	**0**	**1**	**0**
0	0	1	0	**0**	**0**	**1**	**1**
0	0	1	1	**0**	**1**	**0**	**0**
0	1	0	0	**0**	**1**	**0**	**1**
0	1	0	1	**0**	**1**	**1**	**0**
0	1	1	0	**0**	**1**	**1**	**1**
0	1	1	1	**1**	**0**	**0**	**0**
1	0	0	0	**1**	**0**	**0**	**1**
1	0	0	1	**0**	**0**	**0**	**0**

Table 11.3.2.2.1: Truth table for the incrementing decoder

11.3.2.2. Table 11.3.2.2.1 shows the logic for the incrementing decoder. Express A, B, C, D in terms of W, X, Y, Z.

Solution: The desired expressions are available below.

$$
\begin{aligned}
A &= W'X'Y'Z' + W'XY'Z' + W'X'YZ' \\
&\quad + W'XYZ' + W'X'Y'Z \qquad (11.3.2.2.1) \\
B &= WX'Y'Z' + W'XY'Z' \\
&\quad + WX'YZ' + W'XYZ' \qquad (11.3.2.2.2) \\
C &= WXY'Z' + W'X'YZ' \\
&\quad + WX'YZ' + W'XYZ' \qquad (11.3.2.2.3) \\
D &= WXYZ' + W'X'Y'Z \qquad (11.3.2.2.4)
\end{aligned}
$$

Execute the following code. You should see the next number displayed.

```
codes/decoders/incdec/main.c
```

11.3.2.3. Fig. 9.2.1.2 shows the pin diagram for the Vaman. Connect the pins in bank J5 to the seven segment display using Fig. 2.2 and Table 11.3.2.3.1. Do not forget to put a resistor between COM and 3.3V. Then execute the following code

```
codes/fsm/dispdec/main.c
```

Display	Pygmy
a	IO_4
b	IO_5
c	IO_6
d	IO_7
e	IO_8
f	IO_10
g	IO_11
COM	3.3 V

Input Variable	**Pin**
W	IO_28
X	IO_23
Y	IO_31
Z	IO_12

Table 11.3.2.3.1: Pin connection between the Vaman and seven segment display

11.3.2.4. Modify the above code to obtain a decade counter.

11.3.3. Decade Counter

11.3.3.1. Use the Vaman to implement all the decoders in Fig. 5.2. Implement the delay using a flip flop. This is an example of an FSM which is implemented using a sequential circuit.