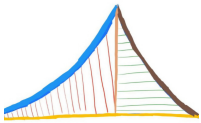
Digital Design Through Arduino



G. V. V. Sharma

About this book

This book provides a simple introduction to digital design using the Arduino framework, assembly and embedded C. It is suitable for students ranging from primary school to college. The content is sufficient for industry jobs. There is no copyright, so readers are free to print and share.

All the boards introduced in this book are well supported by linux toolchains on termux-debian, allowing readers to explore the power of free software and low cost hardware on their android phones.

This book is dedicated to my teachers at IIT Guwahati, Prof. Anil Mahanta and Prof. Anup Gogoi. They somehow managed to teach me circuits.

January 13, 2025

Github:https://github.com/gadepall/digital-design

First manual appeared in 2015
First edition published on July 10, 2024
In this edition, content on the vaman board has been added. This covers ESP 32, ARM Cortex-M4 as well as the Quiclogic EOS-S3 FPGA. In addition, some content on getting started with the Raspberry Pi Pico has been added. Also, the STM32F103C8T6 (blue pill) has been introduced in some detail.

Contents

1 Installation

1.1 Termux

1. On your android device, follow the instructions in

```
https://github.com/gadepall/fwc-1
```

   to setup and install Debian on Termux.

1.2 Platformio

1. Install Packages

```
apt install avra avrdude gcc-avr avr-libc
```

2. Follow the instructions in

```
https://docs.platformio.org/en/stable/core/installation/methods/installer-script.html#
    super-quick-macos-linux
```

   to install platformio.
3. Execute the following on debian

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

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

```
pio run -t nobuild -t upload
```

5. The LED beside pin 13 will start blinking. See Fig. 1.1 for the Arduino pin diagram.

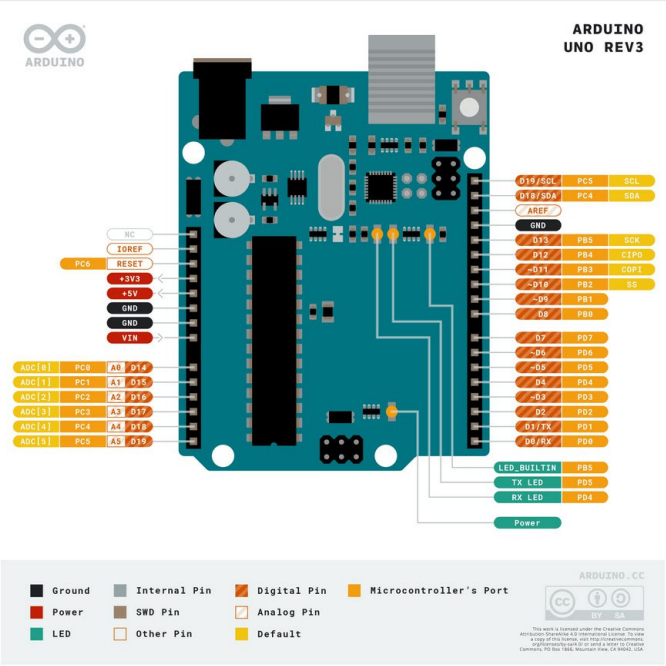


Fig. 1.1

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/firmware.hex

   for uploading hex file to the Arduino Uno
5. The LED beside pin 13 will start blinking

2 Seven Segment Display

We show how to control a seven segment display.

2.1 Components

Component	Value	Quantity
Resistor	220 Ohm	1
Arduino		1
Seven Segment Display		1
Decoder	7447	1
Flip Flop	7474	2
Jumper Wires		20

TABLE 2.1: Components

1. Breadboard:

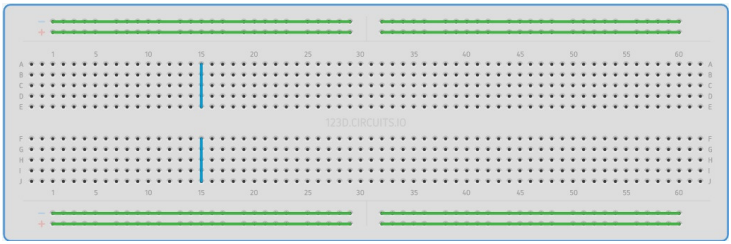


Fig. 2.1: Bread board connnections

   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. 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.
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.

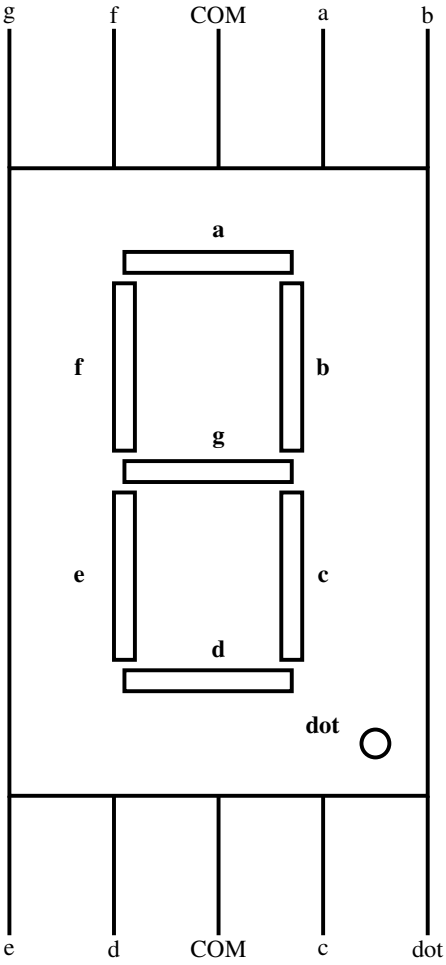


Fig. 2.2: Seven Segment pins

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

TABLE 2.2: Supply for Bread board

2. Make the connections in Table 2.3.

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

TABLE 2.3: Connecting Seven segment display on Bread board

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

TABLE 2.5

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

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

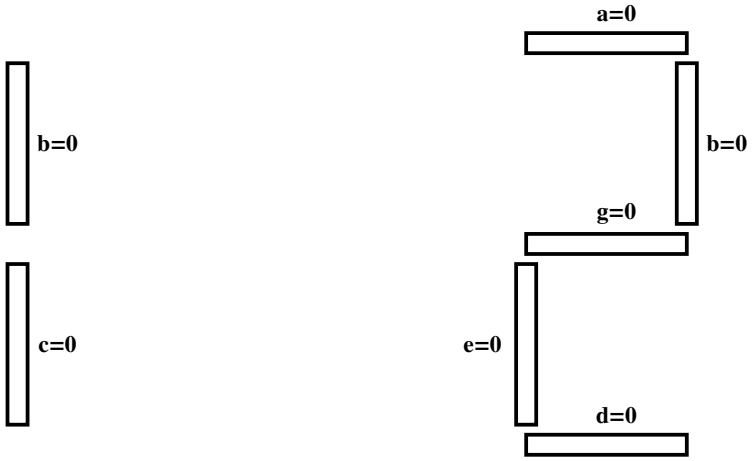


Fig. 2.3: Seven Segment connections

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

2.3 Display Control through Software

1. Make connections according to Table 9.1
2. Download the following code using the arduino IDE and execute

ide/sevenseg/codes/sevenseg/sevenseg.cpp

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

3 7447

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

3.1 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 3.1

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

TABLE 3.1

2. Make connections to the lower pins of the 7447 according to Table 3.2 and connect V_{CC} = 5V. 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 3.2

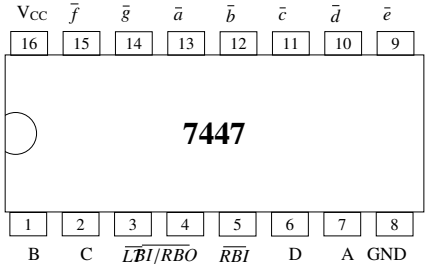


Fig. 3.1: 7447 IC

3. Complete Table 3.2 by generating all numbers between 0-9.

3.2 Software

1. Now make the connections as per Table 9.2 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 3.3

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**

TABLE 3.4: Truth table for incrementing Decoder.

In the truth table in Table 3.4, 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 3.4. Write the logic for A and verify.

ide/7447/codes/inc_dec/inc_dec.ino

3. Now make additional connections as shown in Table 9.3 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 3.5

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 3.4 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.

3.3 Problems

Verify all logic using the Arduino

1. Obtain the Boolean Expression for the logic circuit shown below in Fig. 3.2. (CBSE 2013)

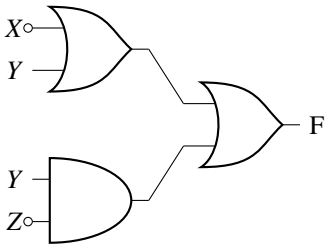


Fig. 3.2

2. Verify the Boolean Expression (CBSE 2013)

$$A + C = A + A'C + BC$$

3. Draw the logic circuit for the following Boolean Expression (CBSE 2015)

$$f(x, y, z, w) = (x' + y)z + w'$$

4. Verify the following (CBSE 2015)

$$U' + V = U'V' + U'V + UV$$

5. Draw the logic circuit for the given Boolean Expression (CBSE 2015)

$$(U + V')W' + Z$$

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

$$X + Y' = XY + XY' + X'Y'$$

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

$$XY + YZ$$

8. Draw the logic circuit of the following Boolean Expression using only NOR Gates. (CBSE 2017)

$$(A + B)(C + D)$$

9. Draw the logic circuit of the following Boolean Expression (CBSE 2018)

$$(U' + V)(V' + W')$$

10. Write the Boolean expression for the result of the logic circuit as shown in Fig. 3.3 (CBSE 2016)

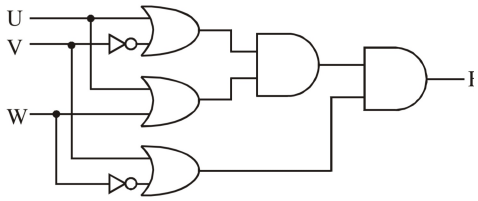


Fig. 3.3

11. Obtain the Boolean Expression for the logic circuit shown below in Fig. 3.4. (GATE EC 1993)

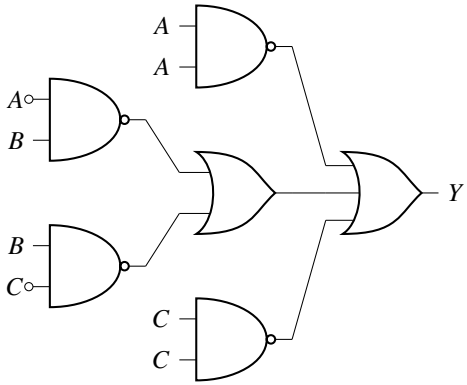


Fig. 3.4

12. Implement Table 3.6 using XNOR logic. (GATE EC 1993)

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

TABLE 3.6

13. 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)
14. Find F in the Digital Circuit given in the figure below in Fig. 3.5. (GATE IN 2016)

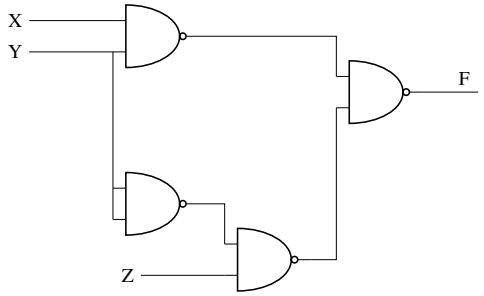


Fig. 3.5

15. Let $\oplus$ and $\odot$ denote the Exclusive OR and Exclusive NOR operations, respectively.Which one of the following is NOT CORRECT ?

(GATE CS 2018)

(A) $\overline{P \oplus Q} = P \odot Q$
(B) $\overline{P} \oplus Q = P \odot Q$
(C) $\overline{P} \oplus \overline{Q} = P \oplus Q$
(D) $(P \oplus \overline{P}) \oplus Q = (P \odot \overline{P}) \odot \overline{Q}$

16. Select the Boolean function(s) equivalent to $x + yz$, where x, y, and z are Boolean variables, and + denotes logical OR operation. (GATE EC 2022)

(A) $x + z + xy$
(B) $(x + y)(x + z)$
(C) $x + xy + yz$
(D) $x + xz + xy$

17. Consider a Boolean gate D where the output Y is related to the inputs A and B as $Y = A + B$, where + denotes logical OR operation. The Boolean inputs 0 and 1 are also available separately. Using instances of only D gates and inputs 0 and 1, select the correct option(s). (GATE EC 2022)

a) NAND logic can be implemented
b) OR logic cannot be implemented
c) NOR logic can be implemented
d) AND logic cannot be implemented.

18. Find the logic function implemented by the circuit given below in Fig. 3.6

(GATE EE 2018)

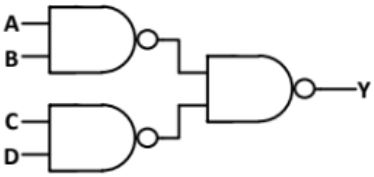


Fig. 3.6

19. Find the logic function implemented by the circuit given below in Fig. 3.7 (GATE EE 2019)

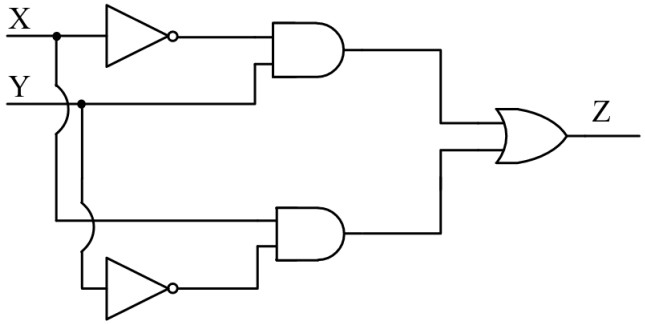


Fig. 3.7

20. Which one of the following options is CORRECT for the given circuit in Fig. 3.8? (GATE PHYSICS 2023)

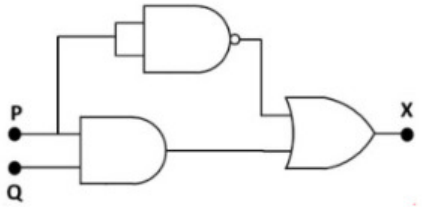


Fig. 3.8

(A) P = 1, Q = 1 ; X = 0
(B) P = 1, Q = 0 ; X = 1
(C) P = 0, Q = 1 ; X = 0
(D) P = 0, Q = 0 ; X = 1

21. Let $R1$ and $R2$ be two 4-bit registers that store numbers in 2's complement form. For the operation $R1 + R2$, which one of the following values of $R1$ and $R2$ gives an arithmetic overflow? (GATE CS 2022)

a) $R1 = 1011$ and $R2 = 1110$
b) $R1 = 1100$ and $R2 = 1010$
c) $R1 = 0011$ and $R2 = 0100$
d) $R1 = 1001$ and $R2 = 1111$

22. The logic block shown in Fig. 3.9 has an output F given by _____.

a) $A + B$
b) $A\bar{B}$
c) $A + \bar{B}$

d) $\bar{B}$

(GATE IN 2021)

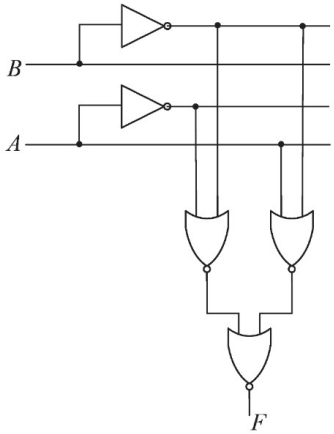


Fig. 3.9

23. Consider the following Boolean expression

$$F = (X + Y + Z)(\bar{X} + Y)(\bar{Y} + Z)$$

Which of the following Boolean expressions is/are equivalent to $\overline{F}$?

a) $(\bar{X} + \bar{Y} + \bar{Z})(X + \bar{Y})(Y + \bar{Z})$
b) $X\bar{Y} + \bar{Z}$
c) $(X + \bar{Z})(\bar{Y} + \bar{Z})$
d) $X\bar{Y} + Y\bar{Z} + \bar{X}\bar{Y}\bar{Z}$

(GATE CS 2021)

24. The following combination of logic gates in Fig. 3.10 represents the operation

a) OR
b) NAND
c) AND
d) NOR

(GATE PH 2021)

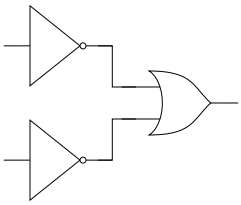


Fig. 3.10

25. In the circuit shown in Fig. 3.11, what are the values of F for $EN = 0$ and $EN = 1$, respectively?

a) 0 and D
b) $Hi - Z$ and D
c) 0 and 1
d) $Hi - Z$ and $\overline{D}$

(GATE EC 2019)

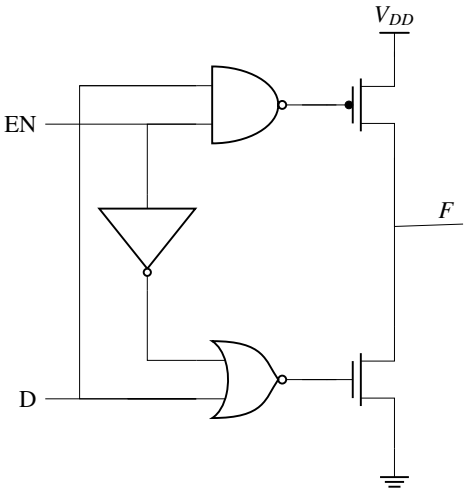


Fig. 3.11: Circuit Diagram

26. In the circuit shown below in Fig. 3.12, assume that the comparators are ideal and all components have zero propagation delay. In one period of the input signal $V_{in} = 6\sin(\omega t)$, the fraction of the time for which the output OUT is in logic HIGH is

a) $\frac{1}{12}$
b) $\frac{1}{2}$
c) $\frac{2}{3}$
d) $\frac{5}{6}$

(GATE IN 2019)

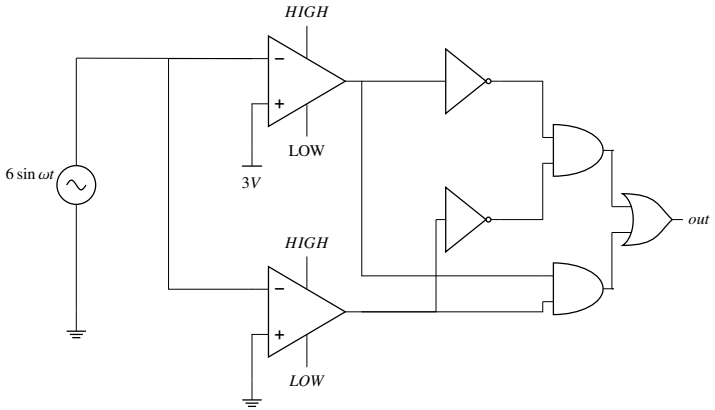


Fig. 3.12: Circuit Daigram

27. Fig. 3.13 shows the *ith* full-adder block of a binary adder circuit. C_i is the input carry and $C_{i+1}$is the output carry of the circuit. If the inputs A_i, B_i; are available and stable throughout the carry propagation, find the outputs S_i and C_{i+1}.

(GATE IN 2019)

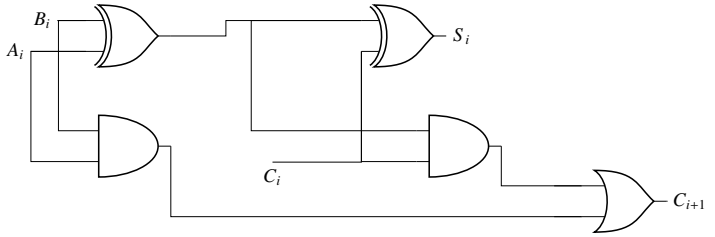


Fig. 3.13: Full Adder

28. The chip select logic for a certain DRAM chip in a memory system design is shown below in Fig. 3.14. Assume that the memory system has 16 address lines denoted by A_{15} to A_0. What is the range of addresses (in hexadecimal) of the memory system that can get enabled by the chip select (CS) signal?

a) $C800$ to $CFFF$
b) $CA00$ to $CAFF$
c) $CA00$ to $C8FF$
d) $DA00$ to $DFFF$

(GATE CS 2019)

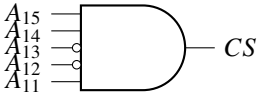
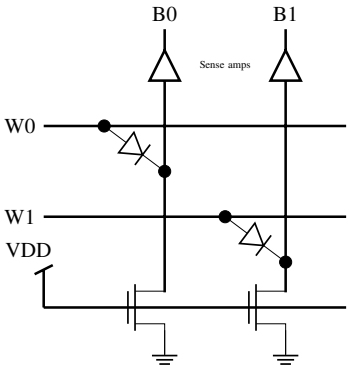


Fig. 3.14

29. A 2×2 ROM array is built with the help of diodes as shown in the circuit below in Fig. 3.15. Here $W0$ and $W1$ are signals that select the word lines and $B0$ and $B1$ are signals that are output of the sense amps based on the stored data corresponding to the bit lines during the read operation.



$$\begin{array}{c} \\ W_0 \\ W_1 \end{array} \begin{array}{c} \begin{array}{cc} B_0 & B_1 \end{array} \\ \begin{bmatrix} D_{00} & D_{01} \\ D_{10} & D_{11} \end{bmatrix} \end{array}$$

Bits stored in the ROM Array

Fig. 3.15: 2×2 ROM array

During the read operation, the selected word line goes high and the other word line is in a high impedance state. As per the implementation shown in the circuit diagram above, what are the bits corresponding to D_{ij} (where $i = 0$ or 1 and $j = 0$ or 1) stored in the ROM? (GATE EC 2018)

a) $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ b) $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ c) $\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$ d) $\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$

30. A and B are logical inputs and X is the logical output shown in Fig. 3.16. The output X is related to A and B by

a) $X = \overline{A}B + \overline{B}A$
b) $X = AB + \overline{B}A$
c) $X = AB + \overline{BA}$
d) $X = \overline{AB} + \overline{B}A$

(GATE IN 2017)

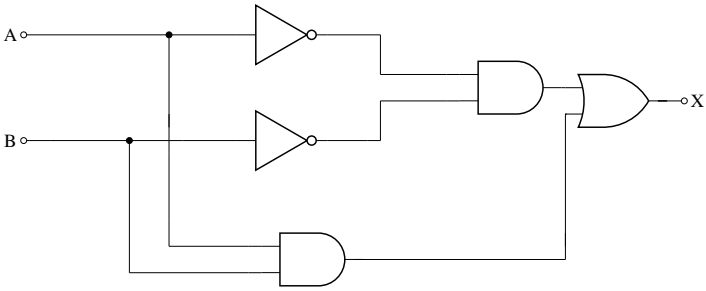


Fig. 3.16

31. Which one the following is not a valid identity?
    a) $(x \oplus y) \oplus z = x \oplus (y \oplus z)$
    b) $(x + y) \oplus z = x \oplus (y + z)$
    c) $x \oplus y = x + y, xy = 0$
    d) $x \oplus y = (xy + x'y')'$

(GATE CS 2019)

32. Let p and q be two propositions. Consider the following two formulae in propositional logic.

$$S_1 : (\rightharpoonup p \vee (p \wedge q)) \rightarrow q$$
$$S_2 : q \rightarrow (\rightharpoonup p \vee (p \wedge q))$$

Which one of the following choices is correct?
    a) Both S_1 and S_2 are tautologies.
    b) S_1 is a tautology but S_2 is not a tautology.
    c) S_1 is not a tautology but S_2 is a tautology.
    d) Neither S_1 nor S_2 is a tautology.

(GATE CS 2021)

33. The functionality implemented by the circuit below in Fig. 3.17 is
    a) 2-to-1 multiplexer
    b) 4-to-1 multiplexer
    c) 7-to-1 multiplexer
    d) 6-to-1 multiplexer

(GATE 2016 EC)

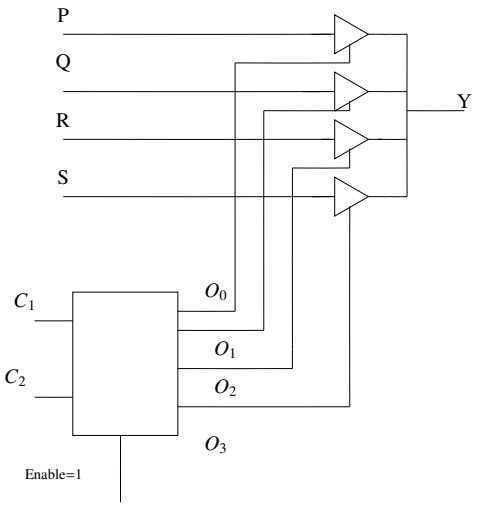


Fig. 3.17: Multiplexer

34. A 3-input majority gate is defined by the logic function

$$M(a, b, c) = ab + bc + ca.$$

Which one of the following gates is represented by the function

$$M(\overline{M(a, b, c)}, M(a, b, \overline{c}), c)?$$

a) 3-input NAND gate
b) 3-input XOR gate
c) 3-input NOR gate
d) 3-input XNOR gate

(GATE EC 2015)

35. A 2-bit flash Analog to Digital Converter (ADC) is given in Fig. 3.18. The input is $0 \leq V_{IN} \leq 3$ Volts. The expression of the LSB of the output B_0 as a boolean function of X_2, X_1, and X_0 is

a) $X_0 \left[\overline{X_2 \oplus X_1}\right]$
b) $\overline{X_0} \left[\overline{X_2 \oplus X_1}\right]$
c) $X_0 [X_2 \oplus X_1]$
d) $\overline{X_0} [X_2 \oplus X_1]$

(GATE EE 2016)

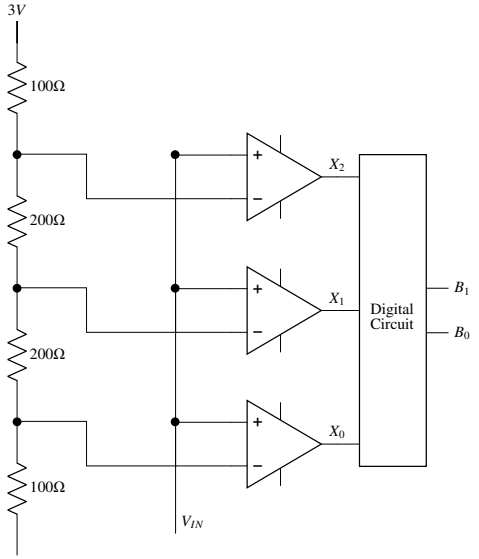


Fig. 3.18

36. In Fig. 3.19, the number of distinct values of $X_3X_2X_1X_0$ (out of the 16 possible values) that give $Y = 1$ is _____. (GATE EC 2018)

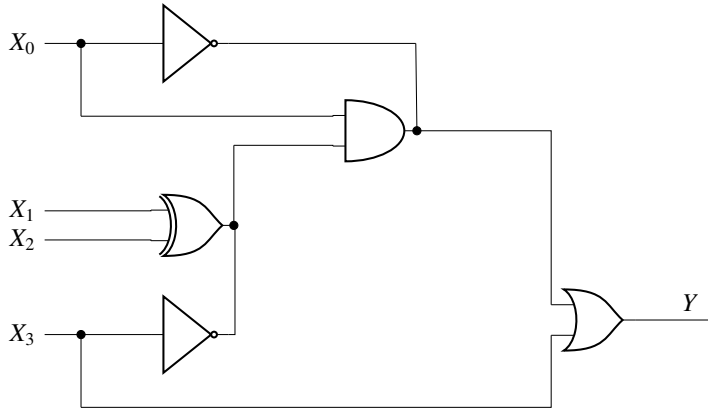


Fig. 3.19

37. In the circuit shown in Fig. 3.20, if $C = 0$, the expression for Y is
    a) $Y = A\overline{B} + \overline{A}B$
    b) $Y = \overline{A} + B$
    c) $Y = \overline{A} + \overline{B}$
    d) $Y = AB$

(GATE EC 2014)

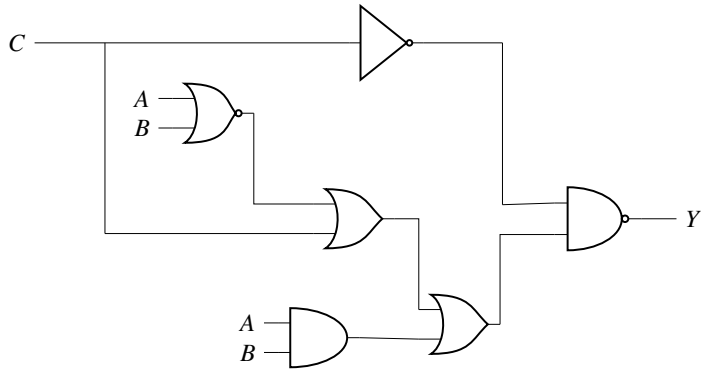


Fig. 3.20

38. In the logic circuit shown in Fig. 3.21, Y is given by (GATE EE 2018)
    a) $Y = ABCD$
    b) $Y = (A + B)(C + D)$
    c) $Y = A + B + C + D$
    d) $Y = AB + CD$

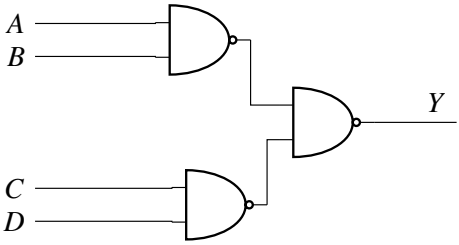


Fig. 3.21

39. The logic function $f(X, Y)$ realised by the given circuit in Fig. 3.22 is
    a) NOR
    b) AND
    c) NAND
    d) XOR

(GATE EC 2018)

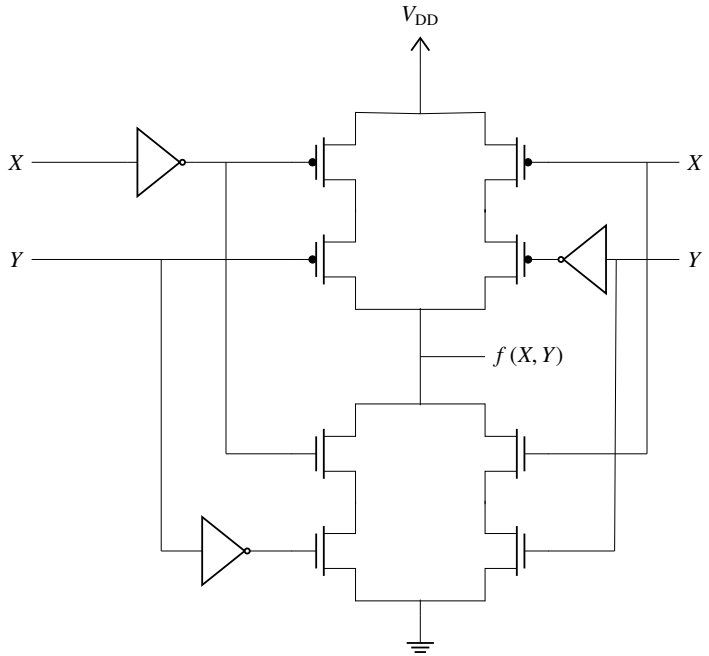


Fig. 3.22

40. P, Q and R are the decimal integers corresponding to the 4-bit binary number 1100 considered in single magnitude, 1's complement and 2's complement representations, respectively. The 6-bit 2's complement representation of $(P + Q + R)$ is
    a) 110101

b) 110010
c) 111101
d) 111001

(GATE EC 2020)

41. In the circuit shown below in Fig. 3.23, P and Q are the inputs. The logical function realized by the circuit is _____. (GATE EC 2023)

a) Y=PQ
b) Y=P+Q
c) Y= $\overline{PQ}$
d) Y= $\overline{P+Q}$

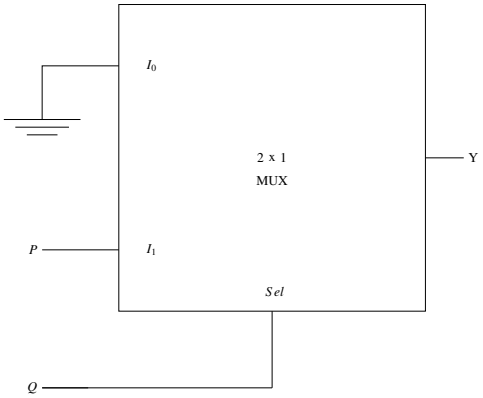


Fig. 3.23

42. In Fig. 3.24, which of the following is correct? (GATE PH 2023)

a) P = 1, Q = 1; X = 0
b) P = 1, Q = 0; X = 0
c) P = 0, Q = 1; X = 0
d) P = 0, Q = 0; X = 1

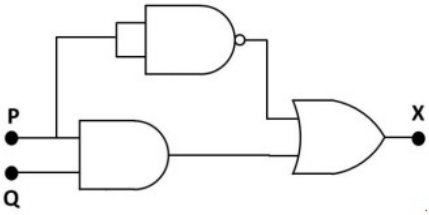


Fig. 3.24

43. The logic block shown in Fig. 3.25 has an output F given by _____.

a) $A + B$

b) $A\bar{B}$
c) $A + \bar{B}$
d) $\bar{B}$

(GATE IN 2022)

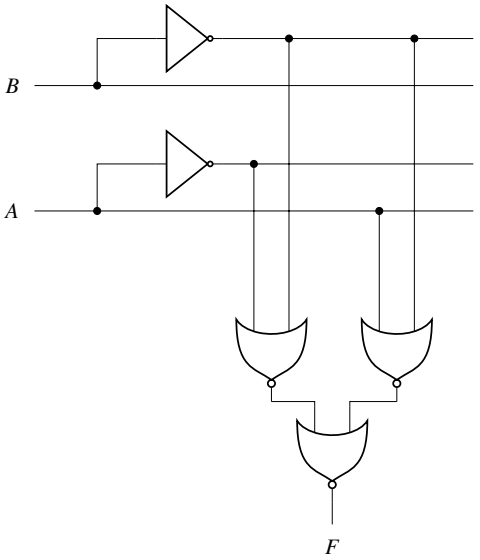


Fig. 3.25

44. In the circuit shown below in Fig. 3.26, X and Y are digital inputs, and Z is a digital output. The equivalent circuit is a (GATE EE 2019)
    a) NAND gate
    b) NOR gate
    c) XOR gate
    d) XNOR gate

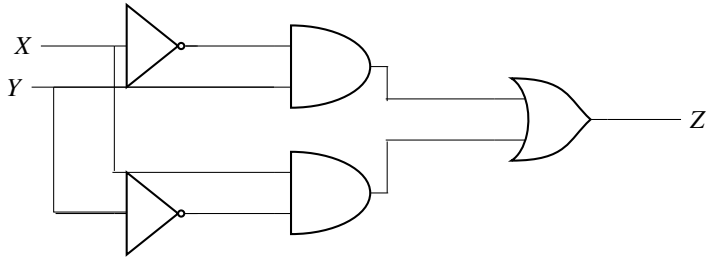


Fig. 3.26

45. The minimum number of two-input NAND gates required to implement the Boolean

expression

$$Y = \left[A\bar{B}(C + BD) + \bar{A}\bar{B}\right]C$$

is _____. (GATE-PH-2022)

46. The Boolean expression for the shaded regions as shown in Fig. 3.27 is _____. ($GATE\ \ IN2020 - 11$)

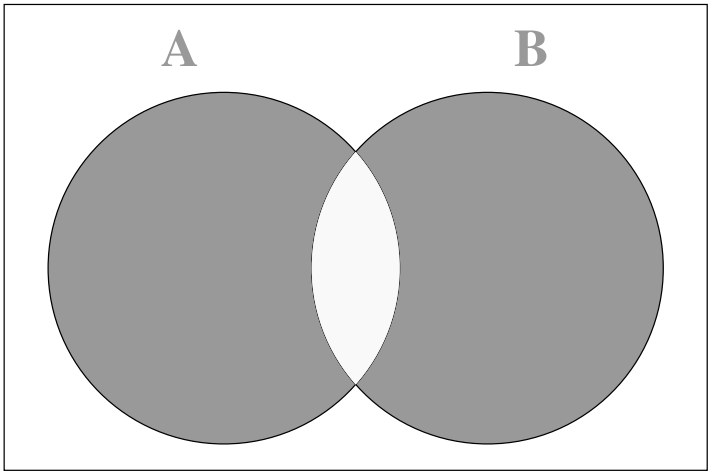


Fig. 3.27: Venn Diagram

a) $(A + B) \bullet (\overline{A} + \overline{B})$
b) $(A + \overline{B}) \bullet (\overline{A} + B)$
c) $(\overline{A} + B) \bullet (\overline{A} + \overline{B})$
d) $(\overline{A} + \overline{B}) \bullet (A + \overline{B})$

47. The Boolean operation performed by the following circuit in Fig. 3.28 at the output O is _____.

a) $O = S_1 \oplus S_0$
b) $O = S_1\overline{S_0}$
c) $O = S_1 + S_0$
d) $O = S_0\overline{S_1}$

(GATE IN 2020)

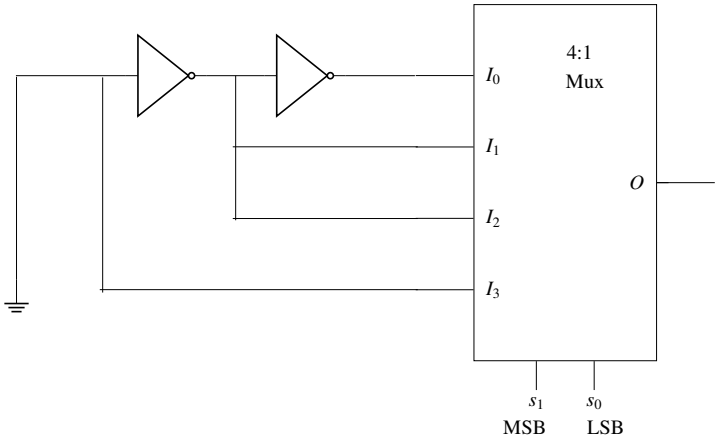


Fig. 3.28

4 Karnaugh Map

4.1 Incrementing Decoder

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

1. 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 3.4
2. Using Boolean logic, output A in Table 3.4 can be expressed in terms of the inputs W, X, Y, Z as

$$A = W'X'Y'Z' + W'XY'Z' + W'X'YZ' + W'XYZ' + W'X'Y'Z \tag{4.1}$$

3. K-Map for A: The expression in (4.1) can be minimized using the K-map in Fig. 4.1

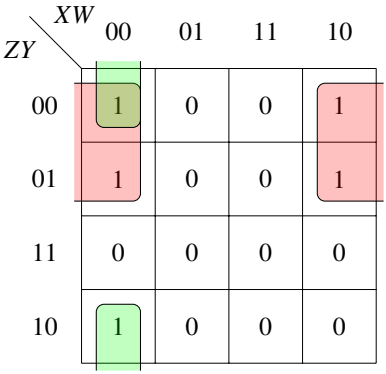


Fig. 4.1: K-map for A

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. 4.1, (4.1) can be expressed as

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

Using the fact that

$$\begin{aligned} X + X' &= 1 \\ XX' &= 0, \end{aligned} \tag{4.3}$$

derive (4.2) from (4.1) algebraically

4. K-Map for B: From Table 3.4, 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

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

using Fig 4.2

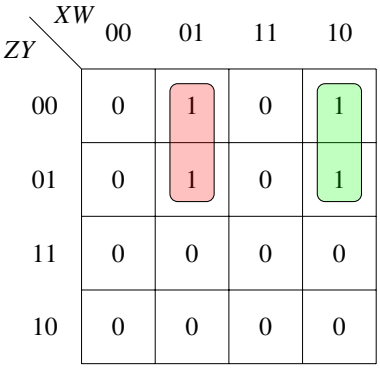


Fig. 4.2: K-map for B

5. Derive (4.5) from (4.4) algebraically using (4.3)
6. K-Map for C: From Table 3.4, using boolean logic,

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

Show that (4.6) can be reduced to

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

using Fig. 4.3.

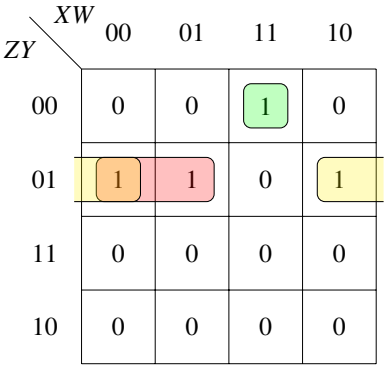


Fig. 4.3: K-map for C

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

8. K-Map for D: From Table 3.4, using boolean logic,

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

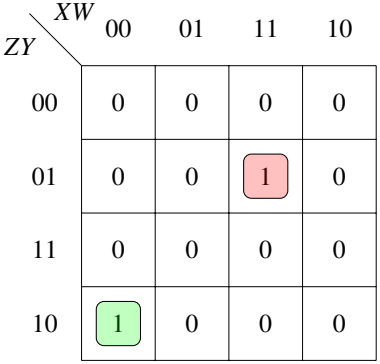


Fig. 4.4: K-map for D

9. Minimize (4.8) using Fig 4.4
10. 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 for each case.

11. Display Decoder: Table 4.1 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.

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 4.1: Truth table for display decoder.

4.2 Dont Care

We explain Karnaugh maps (K-map) using 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 10-15. This phenomenon can be addressed by revising the truth table in Table 4.1 to obtain Table 4.2.

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 4.2

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

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

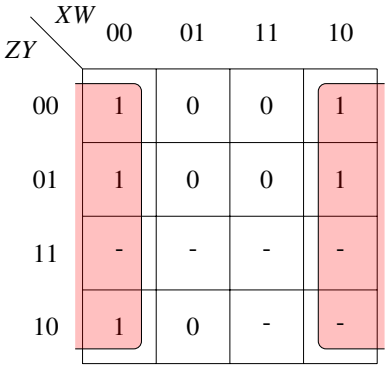


Fig. 4.5: K-map for A with don't cares

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

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

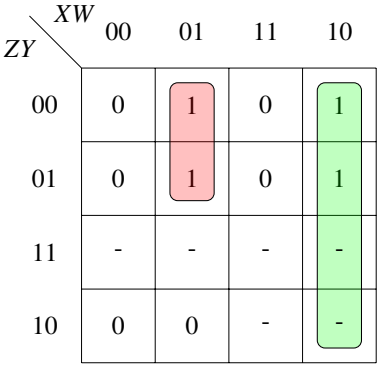


Fig. 4.6: K-map for B with don't cares

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

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

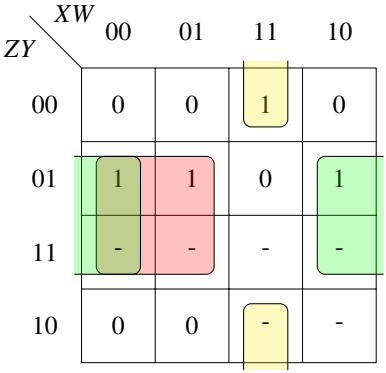


Fig. 4.7: K-map for C with don't cares

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

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

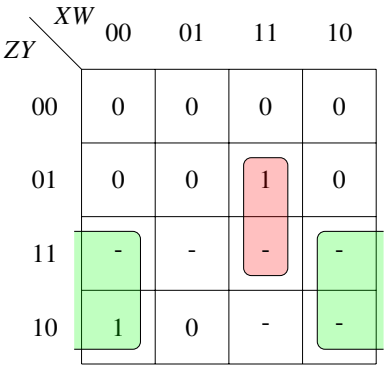


Fig. 4.8: K-map for D with don't cares

6. Verify the incrementing decoder with don't care conditions using the arduino.
7. Display Decoder: In Table 4.1, use K-maps to obtain the minimized expressions for a, b, c, d, e, f, g in terms of A, B, C, D with don't care conditions. Verify using arduino.

4.3 Problems

Verify all your solutions using arduino.

1. Obtain the minimal form for the Boolean expression (CBSE 2013)

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

2. Write the POS form for the function G shown in Table 4.3. (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 4.3

3. Reduce the following Boolean Expression to its simplest form using K-Map.

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

(CBSE 2015)

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

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

5. Reduce the following Boolean Expression to its simplest form using K-map.

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

(CBSE 2016)

6. Derive a Canonical POS expression for a Boolean function F, represented in Table 4.4. (CBSE 2016)

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 4.4

7. Verify the following (CBSE 2016)

$$A' + B'C = A'B'C' + A'BC' + A'BC + A'B'C + AB'C$$

8. Reduce the following boolean expression to its simplest form using K-Map.

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

(CBSE 2017)

9. Reduce the following Boolean Expression to its simplest form using K-Map.

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

(CBSE 2017)

10. Derive a canonical POS expression for a Boolean function G, represented by Table 4.5. (CBSE 2017)

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 4.5

11. Derive a canonical POS expression for a Boolean function FN, represented by Table 4.6. (CBSE 2018)

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 4.6

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

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

(CBSE 2019)

13. 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 is the output. Find the boolean expression for output F as function of inputs P, Q, R using K-map. (GATE EC 2020)

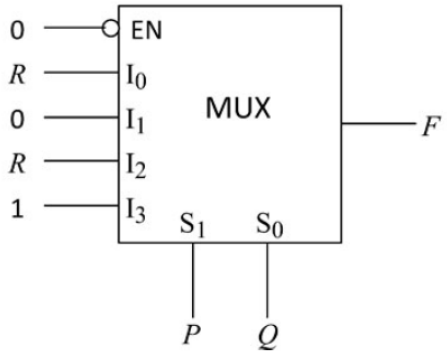


Fig. 4.9

14. The four variable function f is given in terms of min-terms as (GATE EC 1991)

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

Using the K-map minimize the function in the sum of products form.

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

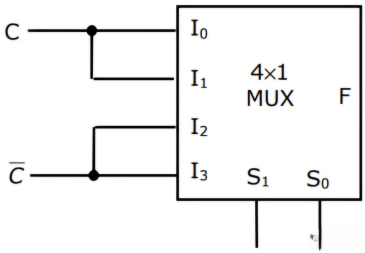


Fig. 4.10

16. 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
    b) A is false and C is true
    c) A, B and C are all false
    d) A, B and C are all true

    Now, do the following.
    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.
17. Draw the logic circuit for Table 4.7 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 4.7

18. Implement the following Boolean function in a 8×1 multiplexer. (GATE EC 1993)

$$Q = BC + ABD' + A'C'D$$

19. Minimize the following Boolean function.

$$F = A'B'C' + A'BC' + A'BC + ABC' \tag{4.13}$$

20. Find the Boolean expression for Table 4.8. (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 4.8

21. Minimize the logic function represented by the following Karnaugh map in Fig. 4.11. (CBSE 2021)

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

Fig. 4.11

22. Find the output for the Karnaugh map shown below in Fig. 4.12 (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

Fig. 4.12

23. If all the inputs P, Q, R, S and T are applied simultaneously and held constant in Fig. 4.13, find the output Y. (GATE EC 2021)

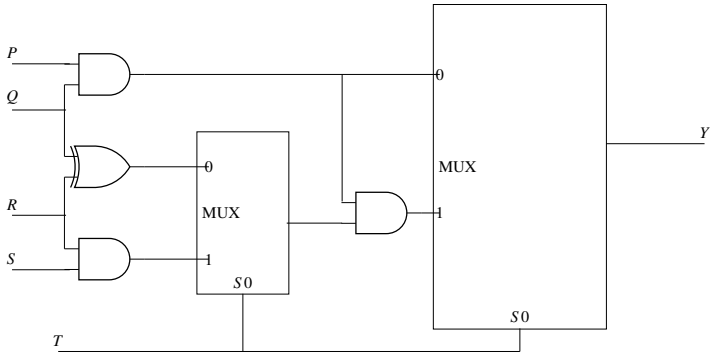


Fig. 4.13

24. Consider the 2-bit multiplexer (MUX) shown in Fig. 4.14. 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) $W = 1, X = 1, Y = 0, Z = 0$

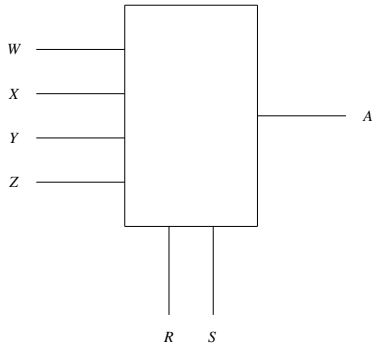


Fig. 4.14

25. $A = a_1a_0$ and $B = b_1b_0$ are two 2-bit unsigned binary numbers. If $F(a_1, a_0, b_1, b_0)$ is a Boolean function such that $F = 1$ only when $A > B$, and $F = 0$ otherwise, then F can be minimized to the form _____. (GATE IN 2022)
26. A 4×1 multiplexer with two selector lines is used to realize a Boolean function F having four Boolean variables X, Y, Z, and W as shown below in Fig. 4.15. S_0 and S_1 denote the least significant bit (LSB) and most significant bit (MSB) of the selector lines of the multiplexer, respectively. I_0, I_1, I_2, I_3 are the input lines of the multiplexer. The canonical sum of product representation of F is (GATE IN 2021)
    a) $F(X, Y, Z, W) = \Sigma m(0, 1, 3, 14, 15)$
    b) $F(X, Y, Z, W) = \Sigma m(0, 1, 3, 11, 14)$
    c) $F(X, Y, Z, W) = \Sigma m(2, 5, 9, 11, 14)$

d) $F(X, Y, Z, W) = \Sigma m(1, 3, 7, 9, 15)$

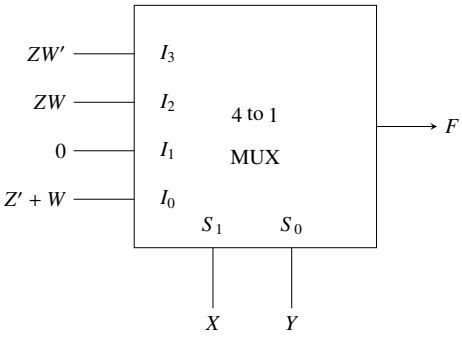


Fig. 4.15

27. $X = X_1X_0$ and $Y = Y_1Y_0$ are 2-bit binary numbers. The Boolean function S that satisfies the condition "If $X>Y$, then $S = 1$", in its minimized form, is
    a) $X_1Y_1+X_0Y_0$
    b) $X_1\overline{Y_1} + X_0\overline{Y_0}\overline{Y_1} + X_0\overline{Y_0}X_1$
    c) $X_1\overline{Y_1}X_0\overline{Y_0}$
    d) $X_1Y_1 + X_0\overline{Y_0}Y_1 + X_0\overline{Y_0X_1}$

    (GATE IN 2019)

28. A function $F(A, B, C)$ defined by three Boolean variables A, B and C when expressed as sum of products is given by

$$F = \left(\overline{ABC}\right) + \left(\overline{A}B\overline{C}\right) + \left(A\overline{BC}\right)$$

    where, $\overline{A}, \overline{B}$ and $\overline{C}$ are the complements of the respective variables. The product of sums (POS) form of the function F is (GATE EC 2018)
    a) $(A + B + C)\left(A + \overline{B} + C\right)\left(\overline{A} + B + C\right)$
    b) $\left(\overline{A} + \overline{B} + \overline{C}\right)\left(\overline{A} + B + \overline{C}\right)\left(A + \overline{B} + \overline{C}\right)$
    c) $\left(A + B + \overline{C}\right)\left(A + \overline{B} + \overline{C}\right)\left(\overline{A} + B + \overline{C}\right)\left(\overline{A} + \overline{B} + C\right)\left(\overline{A} + \overline{B} + C\right)$
    d) $\left(\overline{A} + \overline{B} + C\right)\left(\overline{A} + B + C\right)\left(A + \overline{B} + C\right)\left(A + B + \overline{C}\right)(A + B + C)$

29. The product of sun expression of a Boolean function is given by

$$F(A, B, C) = (A + B + \overline{C})(A + \overline{B} + \overline{C})(\overline{A} + B + C)(\overline{A} + \overline{B} + \overline{C}) \tag{4.14}$$

    The canonical sum of product expression of $F(A, B, C)$ is given by (GATE IN 2018)
    a) $\overline{ABC} + \overline{A}BC + A\overline{BC} + ABC$
    b) $\overline{ABC} + \overline{A}B\overline{C} + A\overline{B}C + AB\overline{C}$
    c) $AB\overline{C} + A\overline{BC} + \overline{A}BC + \overline{ABC}$
    d) $\overline{ABC} + \overline{A}BC + AB\overline{C} + ABC$

30. A four-variable Boolean function is realized using 4×1 multiplexers as shown in the Fig. 4.16. The minimized expression for F is (GATE EC 2018)
    a) $\left(UV + \bar{U}\bar{V}\right)\bar{W}$

b) $\left(UV + \bar{U}\bar{V}\right)\left(\bar{W}\bar{X} + \bar{W}X\right)$
c) $\left(U\bar{V} + \bar{U}V\right)\bar{W}$
d) $\left(U\bar{V} + \bar{U}V\right)\left(\bar{W}\bar{X} + \bar{W}X\right)$

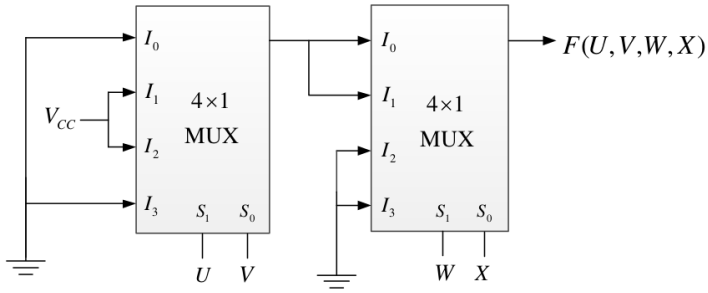


Fig. 4.16

31. In the Karnaugh map shown below in Fig. 4.17, X denotes a don't care term. What is the minimal form of the function represented by the Karnaugh map?
    a) $b'd' + a'd'$
    b) $a'b' + b'd' + a'bd'$
    c) $b'd' + a'bd'$
    d) $a'b' + b'd' + a'd'$

(GATE EC 2008)

cd \ ba	00	01	11	10
00	1	1	0	1
01	X	0	0	0
11	X	0	0	0
10	1	1	0	X

Fig. 4.17

32. Consider the minterm list form of a Boolean function given below.

$$F(P, Q, R, S) = \sum m(0, 2, 5, 7, 9, 11) + d(3, 8, 10, 12, 14)$$

Here, m denotes a minterm and d denotes a don't care term. The number of essential prime implicants of the function is _____. (GATE CS 2018)

33. The simplified form of the Boolean function $F(W, X, Y, Z) = \sum(4, 5, 10, 11, 12, 13, 14, 15)$ with the minimum number of terms and smallest number of literals in each terms is _____. (GATE IN 2023)
    a) $WX + \bar{W}X\bar{Y} + W\bar{X}Y$
    b) $WX + WY + X\bar{Y}$
    c) $X\bar{Y} + WY$
    d) $\bar{X}Y + \bar{W}\bar{Y}$
34. Q, R, S are Boolean variables $\oplus$ and is the XOR operator. Select the CORRECT option(s). (Gate BM 2023)
    a) $(Q \oplus R) \oplus S = Q \oplus (R \oplus S)$
    b) $(Q \oplus R) \oplus S = 0$ when any of the Boolean variables (Q, R, S) are 0 and the third variable is 1
    c) $(Q \oplus R) \oplus S = 1$ when Q = R = S = 1
    d) $((Q \oplus R) \oplus (R \oplus S)) \oplus (Q \oplus S) = 1$
35. The output F of the digital circuit shown in Fig. 4.18 can be written in the form(s) _____.

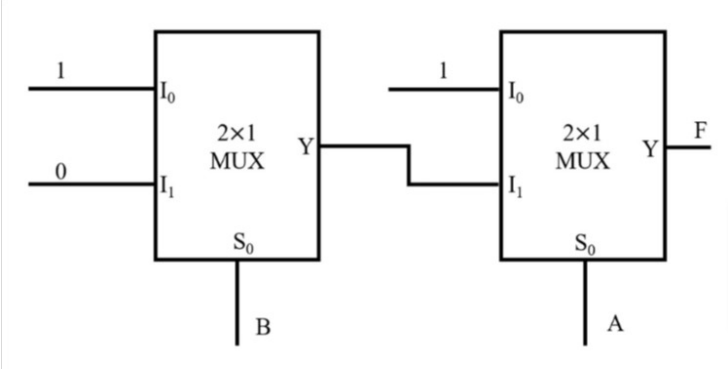


Fig. 4.18

    a) $\overline{A.B}$
    b) $\bar{A}+\bar{B}$
    c) $\overline{A + B}$
    d) $\bar{A}.\bar{B}$

(GATE IN 2022)

36. $\mathbf{A} = a_1a_0$ and $\mathbf{B} = b_1b_0$ are two 2-bit unsigned binary numbers. If $\mathbf{F}(a_1, a_0, b_1, b_0)$ is a Boolean function such that $\mathbf{F} = 1$ only when $\mathbf{A} > \mathbf{B}$, and $\mathbf{F} = 0$ otherwise, then $\mathbf{F}$ can be minimized to the form _____
    a) $a_1\bar{b_1} + a_1a_0\bar{b_0}$
    b) $a_1\bar{b_1} + a_1a_0\bar{b_0} + a_0\bar{b_0}\bar{b_1}$
    c) $a_1a_0\bar{b_0} + a_0\bar{b_0}\bar{b_1}$

d) $a_1\bar{b_1} + a_1a_0\bar{b_0} + a_0\bar{b_0}b_1$

(GATE IN 2022)

37. In the circuit shown below in Fig. 4.19, Y is a 2-bit (Y_1Y_o) output of the combinational logic. What is the maximum value of Y for any given digital inputs, A_1A_0 and B_1B_0 ? (GATE BM 2021)

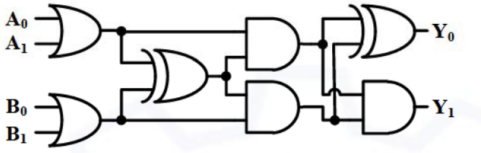


Fig. 4.19

(A) 01
(B) 10
(C) 00
(D) 11

38. Match the Boolean expression with its minimal realization in Table 4.9

	Boolean expression		Minimal realization
P	$\bar{X}\bar{Y}\bar{Z} + \bar{X}Y\bar{Z} + \bar{X}YZ$	K	$X(Y + Z)$
Q	$XYZ + X\bar{Y}Z + XY\bar{Z}$	L	$\bar{X}(Y + \bar{Z})$
R	$XY + XYZ + XY\bar{Z} + \bar{X}YZ$	M	Z
S	$\bar{X}\bar{Y}Z + \bar{X}YZ + X\bar{Y}Z + XYZ$	N	$Y(X + Z)$

TABLE 4.9

(GATE BM 2020)

(A) $P - K$, $Q - L$,$R - N$, $S - M$

(B) $P - L$, $Q - K$, $R - N$, $S - M$

(C) $P - L$,$Q - N$, $R - M$, $S - K$

(D) $P - M$, $Q - K$, $R - L$, $S - N$

39. A 4×1 multiplexer with two selector lines is used to realize a Boolean function F having four Boolean variables X, Y, Z and W as shown below in Fig. 4.20. S_0 and S_1 denote the least significant bit (LSB) and most significant bit (MSB) of the selector lines of the multiplexer respectively. I_0, I_1, I_2, I_3 are the input lines of the multiplexer. (GATE IN 2021)

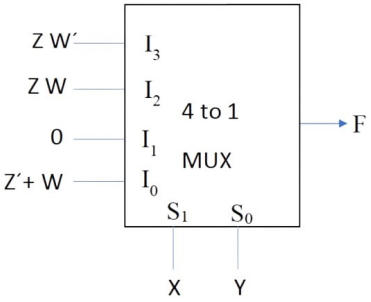


Fig. 4.20: Multiplexer

The canonical sum of product representations of F is

a) $F(X, Y, Z, W) = \sum m(0, 1, 3, 14, 15)$
b) $F(X, Y, Z, W) = \sum m(0, 1, 3, 11, 14)$
c) $F(X, Y, Z, W) = \sum m(2, 5, 9, 11, 14)$
d) $F(X, Y, Z, W) = \sum m(1, 3, 7, 9, 15)$

40. The output expression of the Karnaugh map shown below in Fig. 4.21 is (GATE EE 2017)

AB \ CD	00	01	11	10
00	0	0	0	0
01	1	0	0	1
11	1	0	1	1
10	0	0	0	0

Fig. 4.21

a) $B\overline{D} + BCD$
b) $B\overline{D} + AB$
c) $\overline{B}D + ABC$
d) $B\overline{D} + ABC$

41. A Boolean function F of three variables $X, Y,$ and Z is given as $F(X, Y, Z) = (X' + Y + Z)(X + Y' + Z')(X' + Y + Z')(X'Y'Z' + X'YZ' + XYZ')$.
Which one of the following is true? (GATE IN 2021)

a) $F(X,Y,Z) = (X + Y + Z')(X' + Y' + Z')$
b) $F(X,Y,Z) = (X' + Y)(X + Y' + Z')$
c) $F(X,Y,Z) = X'Z' + YZ'$
d) $F(X,Y,Z) = X'Y'Z + XYZ$

42. The product of sum expression of a Boolean function $F(A,B,C)$ of three variables is given by

$$F(A,B,C) = \left(A + B + \bar{C}\right)\left(A + \bar{B} + \bar{C}\right)\left(\bar{A} + B + C\right)\left(\bar{A} + \bar{B} + \bar{C}\right)$$

The canonical sum of product expression of $F(A,B,C)$ is given by

a) $\bar{A}\ \bar{B}\ C + \bar{A}\ B\ \bar{C} + A\ \bar{B}\ C + A\ B\ C$
b) $\bar{A}\ \bar{B}\ \bar{C} + \bar{A}\ B\ \bar{C} + A\ \bar{B}C + A\ B\ \bar{C}$
c) $A\ B\ \bar{C} + A\ \bar{B}\ \bar{C} + \bar{A}\ B\ C + \bar{A}\ B\ C + \bar{A}\ \bar{B}\ \bar{C}$
d) $\bar{A}\ \bar{B}\ \bar{C} + \bar{A}\ B\ C + A\ B\ \bar{C} + A\ B\ \bar{C} + A\ B\ C$

(GATE IN 2018)

43. Digital input signals A, B, C with A as the MSB and C as the LSB are used to realize the Boolean function $F = m_0 + m_2 + m_3 + m_5 + m_7$, where m_i denotes the i^{th} minterm. In addition, F has a don't care for m_1. The simplified expression for F is given by

a) $\bar{A}\bar{C} + \bar{B}C + AC$
b) $\bar{A} + C$
c) $\bar{C} + A$
d) $\bar{A}C + BC + A\bar{C}$

(GATE EE 2018)

44. Consider the minterm list form of a Boolean function F given below.

$$F(P,Q,R,S) = \sum m(0,2,5,7,9,11) + d(3,8,10,12,14)$$

Here, m denotes a minterm and d denotes a don't care term.The number of essential prime implicants of the function F is _____. (GATE CS 2018)

45. Derive a Canonical POS expression for a Boolean function F, represented by Table 4.10 (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 4.10

46. Find X in the following circuit in Fig. 4.22 (GATE EC 2007)

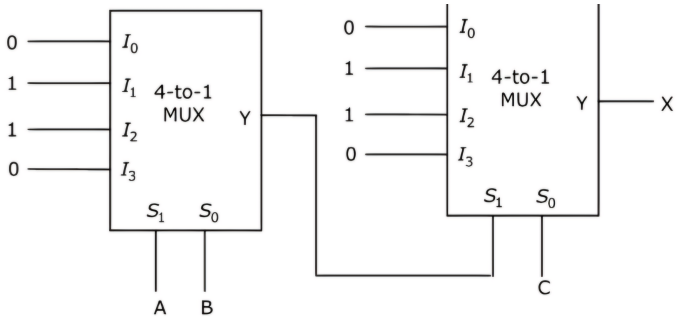


Fig. 4.22

47. A logic circuit implements the boolean function $F = X'Y + XY'Z'$. 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)
48. Find the Boolean logic realised by the following circuit in Fig. 4.23. (GATE EC 2010)

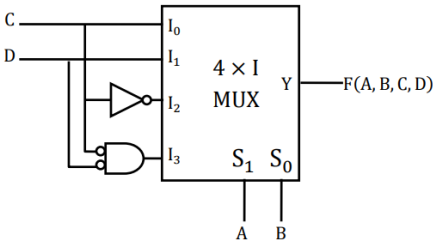


Fig. 4.23

49. Find the logic function implemented by the circuit given below in Fig. 4.24. (GATE EC 2011)

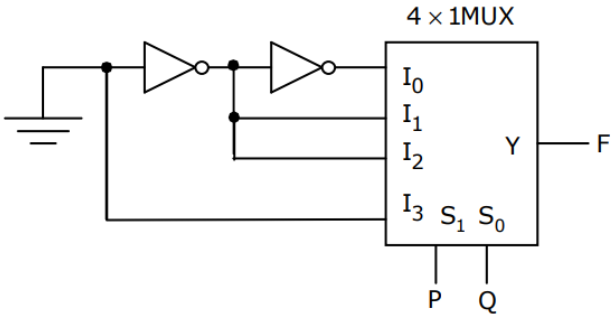


Fig. 4.24

50. The circut shown in Fig. 4.25 comprises of XOR, AND gates and multiplexers. If all the inputs P, Q, R, S and T are applied simultaneously and held constant, find Y. (GATE EC 2021)

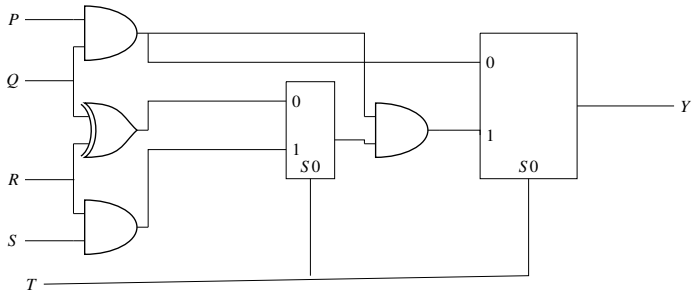


Fig. 4.25

51. Find the logic function implemented by the circuit given below in Fig. 4.26. (GATE EC 2017)

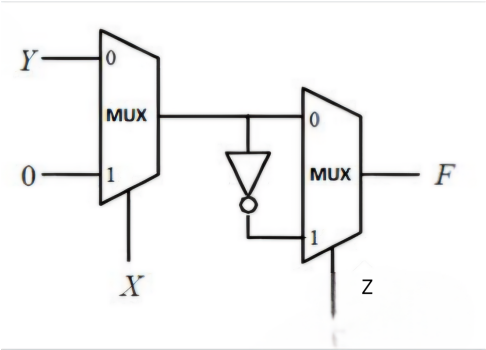


Fig. 4.26

52. A Boolean digital circuit is composed using two 4-input multiplexers $M1$ and $M2$ and one 2-input multiplexer $M3$ as shown in the Fig. 4.27. $X0$–$X7$ are the inputs of the multiplexers $M1$ and $M2$ and could be connected to either 0 or 1. The select lines of the multiplexers are connected to Boolean variables A, B and C as shown. Which one of the following set of values of $(X0, X1, X2, X3, X4, X5, X6, X7)$ will realise the Boolean function $\overline{A} + \overline{AC} + A\overline{B}C$? (GATE CS 2023)

a) (1, 1, 0, 0, 1, 1, 1, 0)
b) (1, 1, 0, 0, 1, 1, 0, 1)
c) (1, 1, 0, 1, 1, 1, 0, 0)
d) (0, 0, 1, 1, 0, 1, 1, 1)

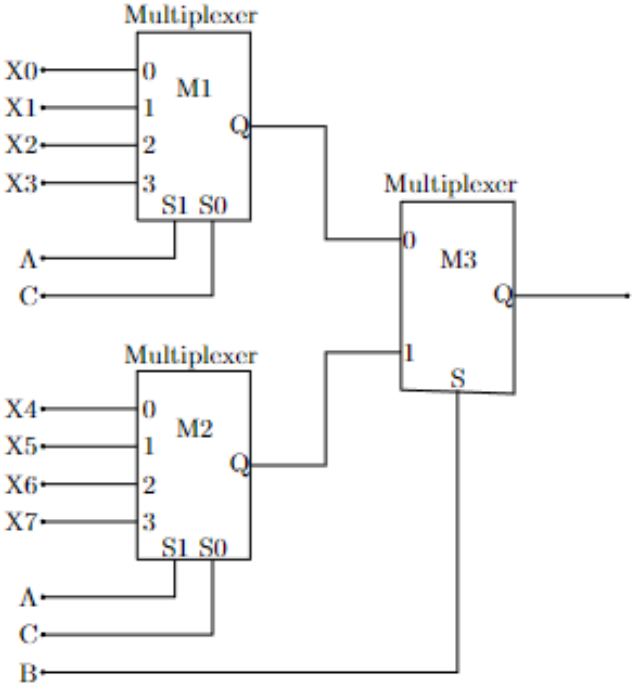


Fig. 4.27: Digital Circuit

53. Consider the boolean Function $z(a, b, c)$ from Fig. 4.28 below. Which of the following minterm lists represent the circuit given above?

a) $z = \Sigma(0, 1, 3, 7)$
b) $z = \Sigma(1, 4, 5, 6, 7)$
c) $z = \Sigma(2, 4, 5, 6, 7)$
d) $z = \Sigma(2, 3, 5)$

(GATE CS 2020)

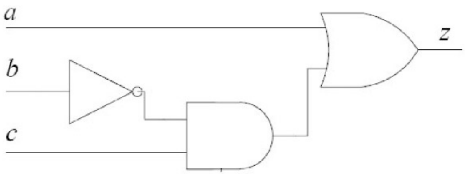


Fig. 4.28

54. Consider three 4-variable functions f_1, f_2,and f_3, which are expressed in sum-of-minterms as

$$f_1 = \sum(0, 2, 5, 8, 14), f_2 = \sum(2, 3, 6, 8, 14, 15), f_3 = \sum(2, 7, 11, 14).$$

For the following circuit in Fig. 4.29, with one AND gate and one XOR gate, the output function f can be expressed as

a) $\sum(7, 8, 11)$
b) $\sum(2, 7, 8, 11, 14)$
c) $\sum(2, 14)$
d) $\sum(0, 2, 3, 5, 6, 7, 8, 11, 14, 15)$

(GATE CS 2019)

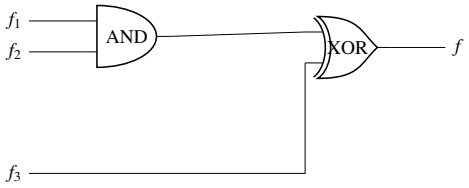


Fig. 4.29

55. For the logic circuit shown in Fig. 4.30, find the simplified Boolean expression for the output. (GATE EC 2000)

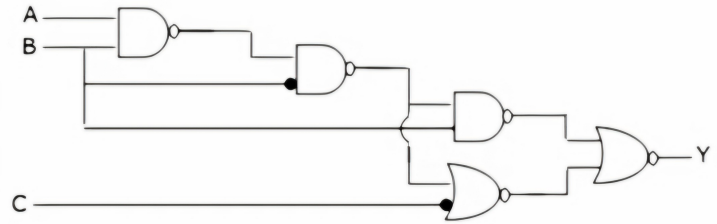


Fig. 4.30

56. Consider the Boolean function Z(a, b, c). Which one of the following minterm lists represents the circuit given below in Fig. 4.31?

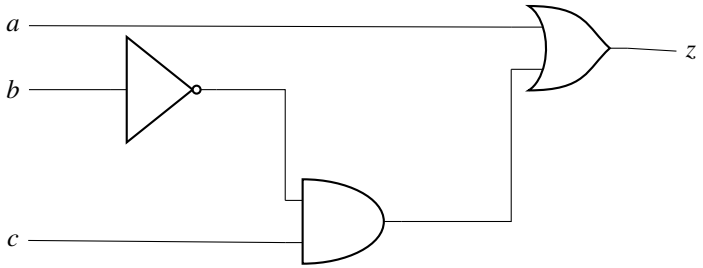


Fig. 4.31

A. $z = \sum(0, 1, 3, 7)$

B. $z = \sum(1, 4, 5, 6, 7)$
C. $z = \sum(2, 4, 5, 6, 7)$
D. $z = \sum(2, 3, 5)$

(GATE CS 2020)

57. The Boolean expression $F(X, Y, Z) = \overline{X}Y\overline{Z} + X\overline{Y}\overline{Z} + XY\overline{Z} + XYZ$ converted into the canonical product of sum (POS) form is
a) $(X + Y + Z)\left(X + Y + \overline{Z}\right)\left(X + \overline{Y} + \overline{Z}\right)\left(\overline{X} + Y + \overline{Z}\right)$
b) $\left(X + \overline{Y} + Z\right)\left(\overline{X} + Y + \overline{Z}\right)\left(\overline{X} + \overline{Y} + Z\right)\left(\overline{X} + \overline{Y} + \overline{Z}\right)$
c) $(X + Y + Z)\left(\overline{X} + Y + \overline{Z}\right)\left(X + \overline{Y} + Z\right)\left(\overline{X} + \overline{Y} + \overline{Z}\right)$
d) $\left(X + \overline{Y} + \overline{Z}\right)\left(\overline{X} + Y + Z\right)\left(\overline{X} + \overline{Y} + Z\right)(X + Y + Z)$

(GATE EC 2015)

58. Consider the 2-bit multiplexer (MUX) shown in Fig. 4.32. For OUTPUT to be the XOR of C and D, the values for A_0, A_1, A_2 and A_3 are _____. (GATE EC 2022)
a) $A_0 = 0, A_1 = 0, A_2 = 1, A_3 = 1$
b) $A_0 = 1, A_1 = 0, A_2 = 1, A_3 = 0$
c) $A_0 = 0, A_1 = 1, A_2 = 1, A_3 = 0$
d) $A_0 = 1, A_1 = 1, A_2 = 0, A_3 = 0$

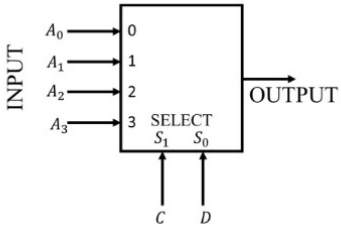


Fig. 4.32: MUX

5 7474

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

5.1 Decade Counter

1. Generate the CLOCK signal using the **blink** program in the arduino.
2. Connect the Arduino, 7447 and the two 7474 ICs according to Table 9.4 and Fig. 5.2. The pin diagram for 7474 is available in Fig. 5.1

	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 5.1

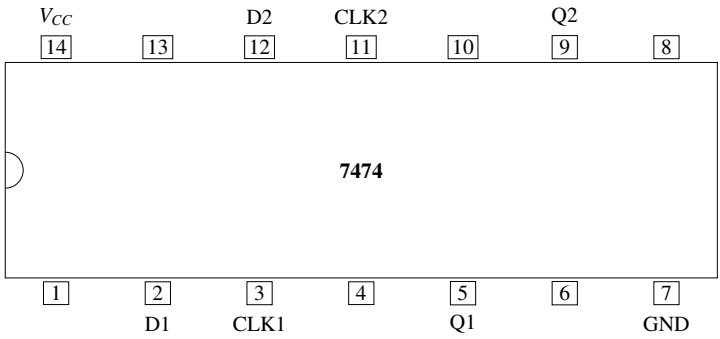


Fig. 5.1

3. Intelligently use the codes in

ide/7447/codes/inc_dec/inc_dec.ino

and

ide/7447/codes/inc_dec/ip_inc_dec.ino

to realize the decade counter in Fig. 5.2.

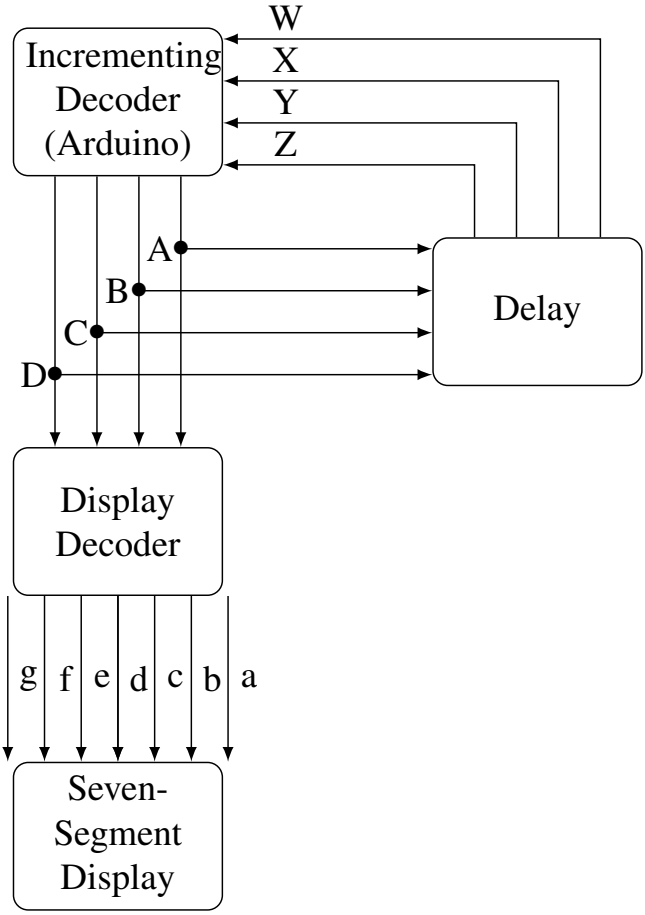


Fig. 5.2

5.2 Problems

1. A counter is constructed with three D flip-flops. The input-output pairs are named (D_0, Q_0), (D_1, Q_1), and (D_2, Q_2), 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 $Q_2\ Q_1\ Q_0$ format. The combinational logic expression for D_1 is

   a) $Q_2Q_1Q_0$
   b) $Q_2Q_0 + Q_1\bar{Q}_0$
   c) $\bar{Q}_2Q_0 + Q_1\bar{Q}_0$
   d) $Q_2Q_1 + \bar{Q}_2\bar{Q}_1$

   (GATE EE 2021)

2. Implement a 4-stage ripple counter utilizing flip-flops. (GATE EE 2022)

3. In the circuit in Fig. 5.3, the clock(Clk) frequency provided to the circuit is 500MHz. Starting from the initial value of the flip-flop outputs $Q2Q1Q0 = 111$ with $D2 = 1$, fine the time after which $Q2Q1Q0 = 100$. (GATE EC 2021)

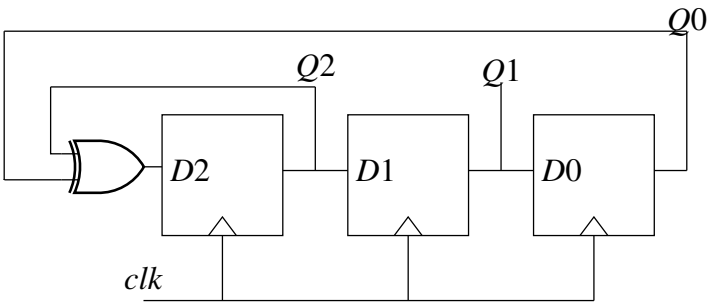


Fig. 5.3

4. For the 3-bit binary counter shown in Fig. 5.4, the output increments at every positive transition in the clock (CLK). Assume ideal diodes and the starting state of the counter as 000. If output high is $1V$ and output low is $0V$, find the current I (in mA) flowing through the 50Ω resistor during the 5th clock cycle (up to one decimal place). (GATE IN 2018)

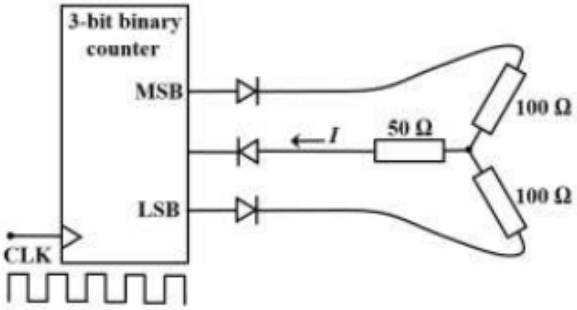


Fig. 5.4

5. Consider the sequential circuit shown in Fig. 5.5, where both flip-flops used are positive edge-triggered D flip-flops. The number of states in the state transition diagram of this circuit that have a transition back to the same state on some value of "in" is _____.

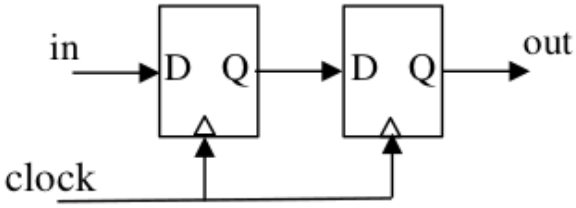


Fig. 5.5

(GATE IN 2018)

6. Implement the synchronous sequential circuit shown below in Fig. 5.6 with a clock frequency of $500Hz$. (GATE EC 2023)

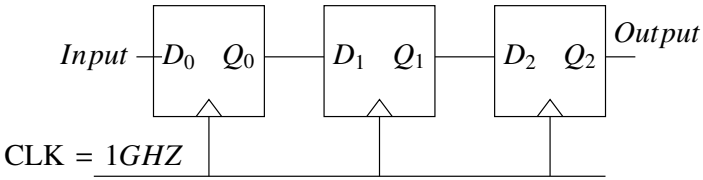


Fig. 5.6

7. In a given sequential circuit in Fig. 5.7, initial states are $Q1 = 1$ and $Q2 = 0$. For a clock frequency of $1MHz$, find the frequency of signal $Q2$ in kHz (rounded off to the nearest integer). (GATE EC 2023)

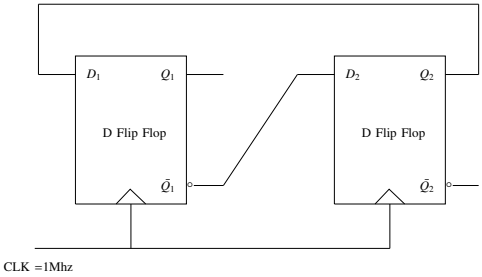


Fig. 5.7

8. Consider a sequential digital circuit consisting of T flip-flops and D flip-flops as shown in Fig. 5.8. CLKIN is is the clock input to the circuit. At the beginning,Q1,Q2 and Q3 have values 0,1 and 1, respectively. Which of the given values of (Q_1, Q_2, Q_3) can NEVER be obtained with this digital circuit? (GATE CS 2023)

a) $(0, 0, 1)$
b) $(1, 0, 0)$

c) $(1, 0, 1)$
d) $(1, 1, 1)$

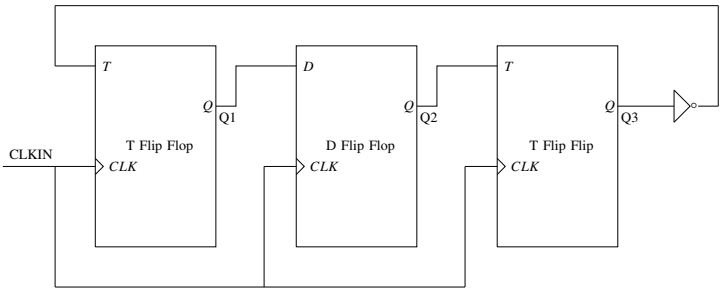


Fig. 5.8

9. Find the decimal equivalent of $ABCD$ in Fig. 5.9. (EE GATE 2023)

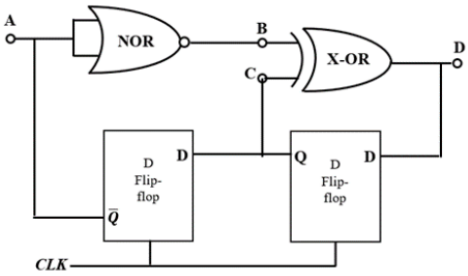


Fig. 5.9

10. In the circuit shifter in Fig. 5.10, the initial binary content of the shift register A is 1101 and that of shift register B is 1010 The shift registers are positive edge triggered, and the gates have no delay. when the shift control is high, which of the following are legitimate combinations of binary content of the shift registers A and B in any clock pulse? (GATE IN 2023)
    a) $A = 1101, B = 1101$
    b) $A = 1110, B = 1001$
    c) $A = 0101, B = 1101$
    d) $A = 1010, B = 1111$

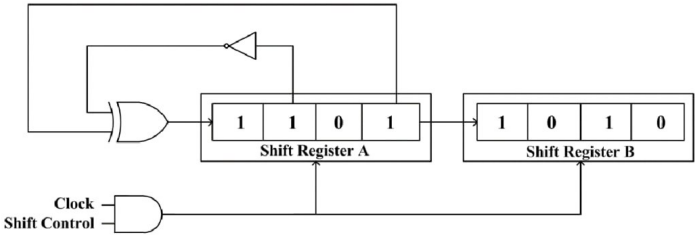


Fig. 5.10

11. For the circuit shown in Fig. 5.11, the clock frequency is f_0 and the duty cycle is 25%. For the signal at the Q output of the Flip-Flop, (GATE EC 2022)
    a) frequency is $\frac{f_0}{4}$ and duty cycle is 50%
    b) frequency is $\frac{f_0}{4}$ and duty cycle is 25%
    c) frequency is $\frac{f_0}{2}$ and duty cycle is 50%
    d) frequency is f_0 and duty cycle is 25%

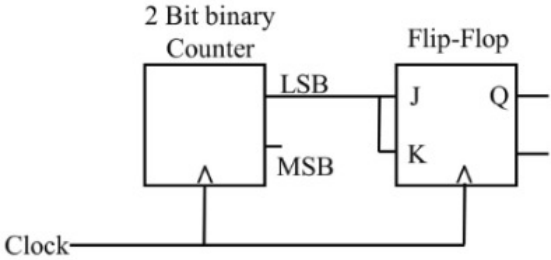


Fig. 5.11: Circuit

12. The digital circuit shown in Fig. 5.12 (GATE IN 2022)

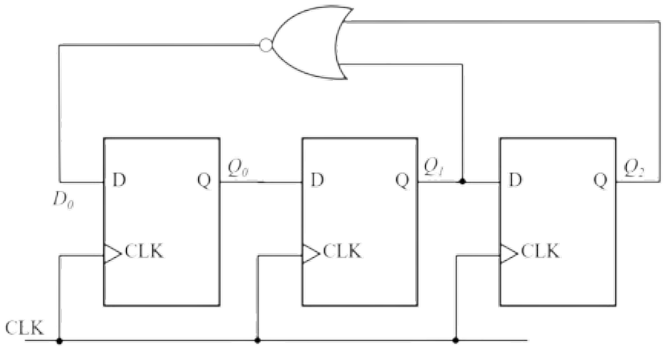


Fig. 5.12

a) is a divide-by-5 counter
b) is a divide-by-7 counter
c) is a divide-by-8 counter
d) does not function as a counter due to disjoint cycles of states

13. Given below in Fig. 5.13 is the diagram of a synchronous sequential circuit with one $J-K$ flip-flop and one T flip-flop with their outputs denoted as A and B respectively, with $J_A = (A' + B')$, $K_A = (A + B)$ and $T_B = A$. Starting from the initial state ($AB = 00$), the sequence of states (AB) visited by the circuit is (GATE IN 2021)

a) $00 \rightarrow 01 \rightarrow 10 \rightarrow 11 \rightarrow 00\ldots$
b) $00 \rightarrow 10 \rightarrow 01 \rightarrow 11 \rightarrow 00\ldots$
c) $00 \rightarrow 10 \rightarrow 11 \rightarrow 01 \rightarrow 00\ldots$
d) $00 \rightarrow 01 \rightarrow 11 \rightarrow 00\ldots$

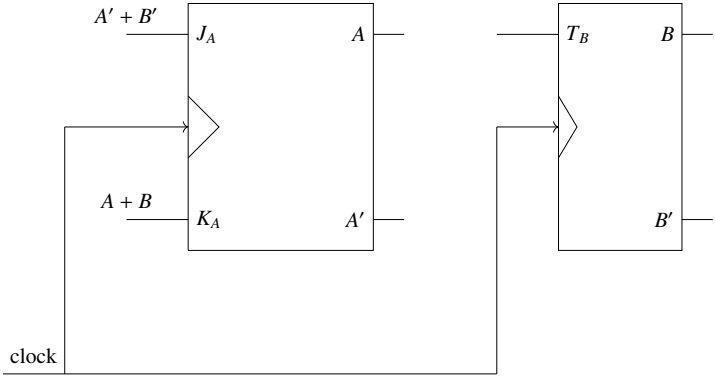


Fig. 5.13

14. Consider the D-Latch shown in Fig. 5.14, which is transparent when its clock input CK is high and has zero propagation delay. In the figure, the clock signal $CLK1$ has

50% duty cycle and $CLK2$ is a one fifth period delayed version of $CLK1$. The duty cycle at the output of the latch in percentage is _____. (GATE-EC2017)

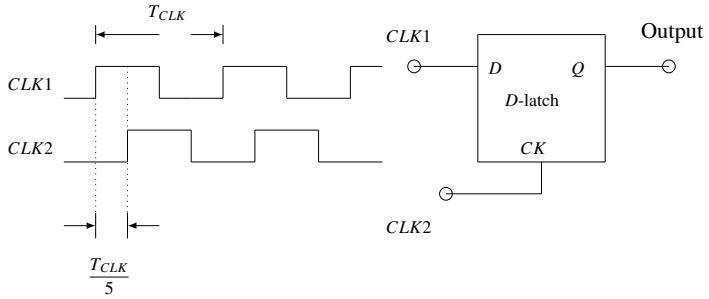


Fig. 5.14

15. A 4-bit shift register circuit configured for right-shift operation, i.e. $D_{in} \rightarrow A, A \rightarrow B, B \rightarrow C, C \rightarrow D$, is shown in Fig. 5.15. If the present state of the shift register is $ABCD = 1101$, the number of clock cycles required to reach the state $ABCD = 1111$ is _____. (GATE EC 2017)

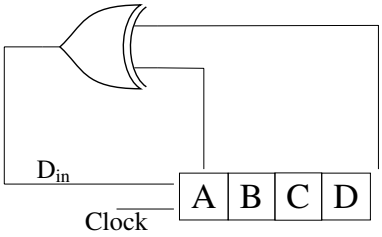


Fig. 5.15

16. In the circuit shown in Fig. 5.16, the clock frequency is 12 KHz. The frequency of the signal at $Q2$ is _____ KHz.

(GATE EC 2019)

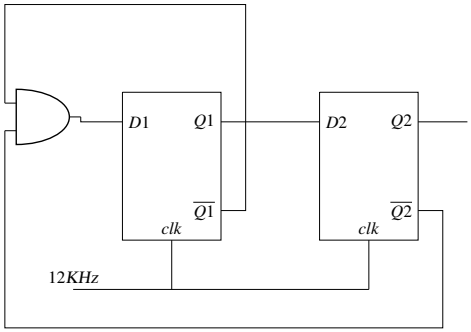


Fig. 5.16

17. The circuit shown in Fig. 5.17 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)
    a) $X = 0, Y = 0$
    b) $X = 0, Y = 1$
    c) $X = 1, Y = 0$
    d) $X = 1, Y = 1$

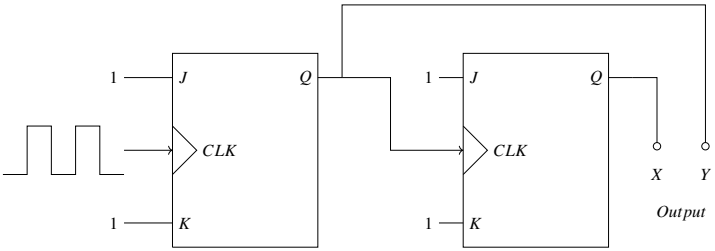


Fig. 5.17

18. The digital circuit shown in Fig. 5.18 generates a modified clockpulse at the output. Sketch the output waveform. (GATE EE 2004)

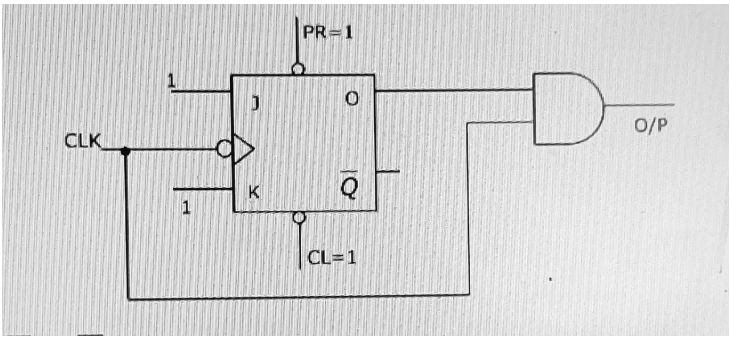


Fig. 5.18

19. Consider a 4-bit counter constructed out of four flip-flops. It is formed by connecting the J and K inputs to logic high and feeding the Q output to the clock input of the flip-flop in Fig. 5.19. The input signal to the counter is a series of square pulses and the change of state is triggered by the falling edge. At time $t = t_0$ the outputs are in logic low state (Q0 = Q1 = Q2 = Q3 = 0). Then at $t = t_1$, the logic state of the outputs is (GATE PH 2020)

a) Q0 = 1, Q1 = 0, Q2 = 0 and Q3 = 0
b) Q0 = 0, Q1 = 0, Q2 = 0 and Q3 = 1
c) Q0 = 1, Q1 = 0, Q2 = 1 and Q3 = 0
d) Q0 = 0, Q1 = 1, Q2 = 1 and Q3 = 1

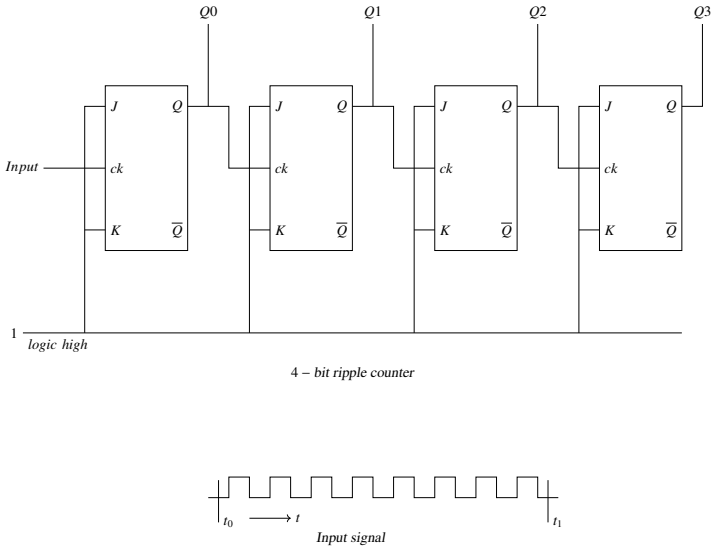


Fig. 5.19

20. A 2-bit synchronous counter using two J-K flip flops is shown in Fig. 5.20. The

expression for the inputs to the J-K flip flops are also shown in the figure. The output sequence of the counter starting from $Q_1Q_2 = 00$ is (GATE IN 2018)

a) $00 \rightarrow 11 \rightarrow 10 \rightarrow 01 \rightarrow 00\ldots$
b) $00 \rightarrow 01 \rightarrow 10 \rightarrow 11 \rightarrow 00\ldots$
c) $00 \rightarrow 01 \rightarrow 11 \rightarrow 10 \rightarrow 00\ldots$
d) $00 \rightarrow 10 \rightarrow 11 \rightarrow 01 \rightarrow 00\ldots$

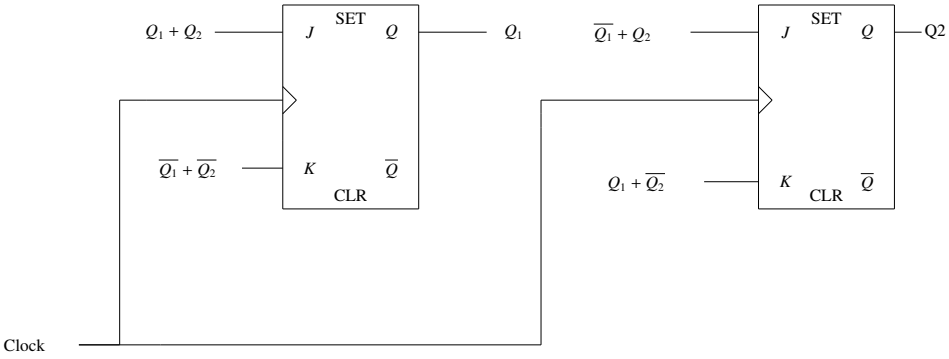


Fig. 5.20

21. Which of the following statements is true about digital circuit shown in Fig. 5.21? (GATE EE 2018)

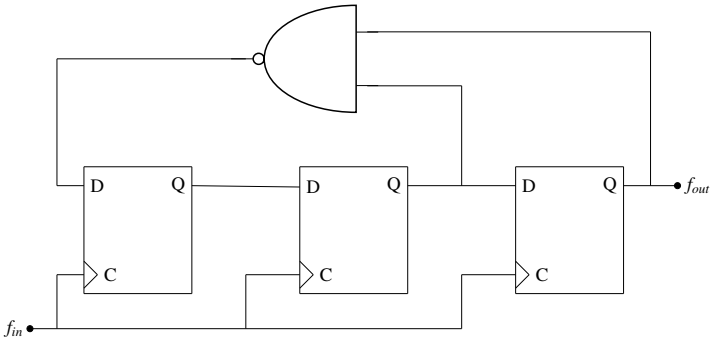


Fig. 5.21

a) It can be used for dividing the input frequency by 3.
b) It can be used for dividing the input frequency by 5.
c) It can be used for dividing the input frequency by 7.
d) It cannot be reliably used as frequency divider due to disjoint internal cycles.

22. In the circuit shown below, a positive edge-triggered D Flip-Flop is used for sampling input data D_{in} using clock CK. The XOR gate outputs 3.3 volts for logic HIGH and 0 volts for logic LOW levels. The data bit and clock periods are equal and the value of $\triangle T/T_{CK} = 0.15$, where the parameters $\triangle T$ and T_{CK} are shown in Fig. 5.22. Assume

that the Flip-Flop and the *XOR* gate are ideal. If the probability of input data bit (D_{in}) transition in each clock period is 0.3, the average value (in volts, accurate to two decimal places) of the voltage at node X, is _____. (GATE EC 2018)

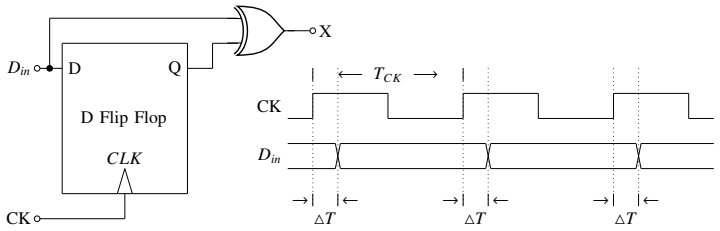


Fig. 5.22

23. Assume that all the digital gates in the circuit shown in Fig. 5.23 are ideal,the resistor R=10$k\Omega$ and the supply voltages is $5V$. The D flip-flops D_1, D_2, D_3, D_4 and D_5 are intialized with logic values $0, 1, 0, 1$, and 0, respectively. The clock has a 30% duty cycle. The average power dissipated in the resistor R is _____. (GATE EC 2016)

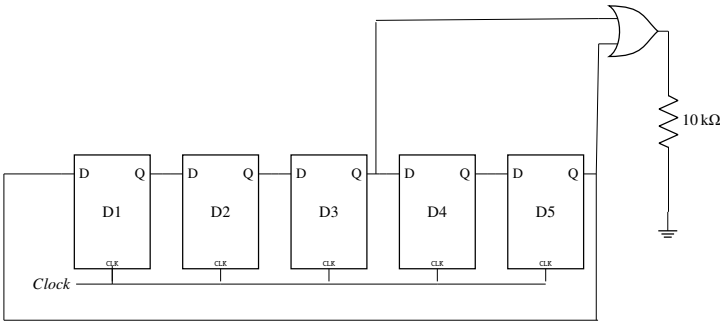


Fig. 5.23

24. 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)
25. Two T-flip flops are interconnected as shown in Fig. 5.24. The present state of the flip flops are: $A = 1, B = 1$. The input x is given as $1, 0, 1$ in the next three clock cycles. The decimal equivalent of $(ABy)_2$ with A being the MSB and y being the LSB, after the 3rd clock cycle is _____. (GATE IN 2020)

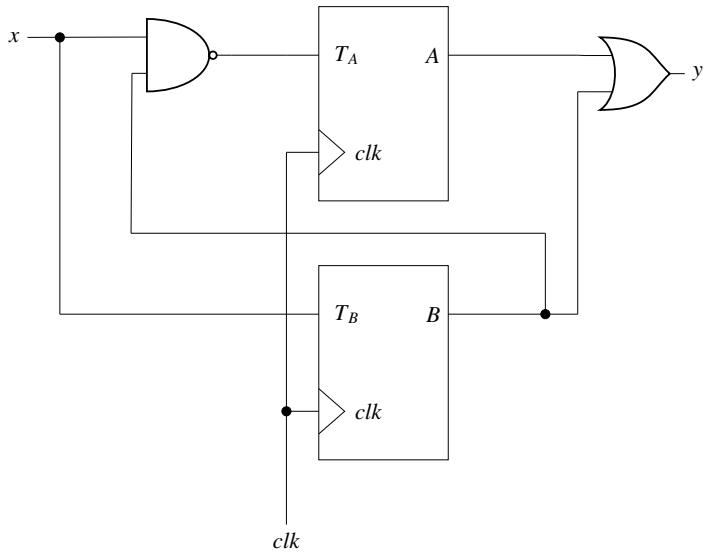


Fig. 5.24

26. A 16-bit synchronous binary up-counter is clocked with the frequency f_{CLK}. The two most significant bits are OR-ed together to form an output Y. Measurements show that Y is periodic, and the duration for which Y remains high in each period is 24 ms. Find the clock frequency. (GATE EE 2021)
27. The sequence of states (Q_1Q_0) of the given synchronous sequential circuit in Fig. 5.25 is _____. (GATE EC 2024)

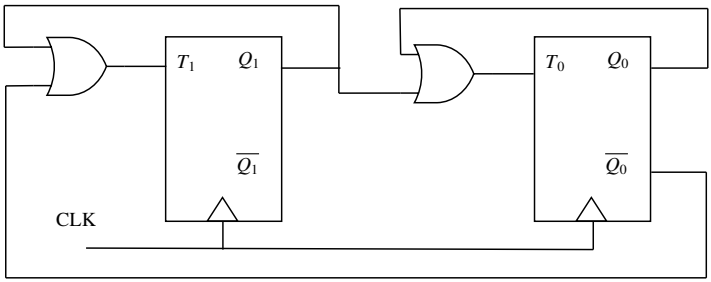


Fig. 5.25

a) $00 \rightarrow 10 \rightarrow 11 \rightarrow 00$
b) $11 \rightarrow 00 \rightarrow 10 \rightarrow 01 \rightarrow 00$
c) $01 \rightarrow 10 \rightarrow 11 \rightarrow 00 \rightarrow 01$
d) $00 \rightarrow 01 \rightarrow 10 \rightarrow 00$

28. A $6\frac{1}{2}$ digit time counter is set in the time period mode of operation and range is set as 'ns'. For an input signal the time-counter displays 1000000. With the same input

signal, the time counter is changed to 'frequency' mode of operation and the range is set as Hz. The display will be show the number _____. (GATE IN 2020)

29. The two inputs A and B are connected to to an R-S latch via two AND gates as shown in Fig. 5.26. If $A = 1$ and $B = 0$, the output $Q\bar{Q}$ is (GATE IN 2017)
    a) 00
    b) 10
    c) 01
    d) 11

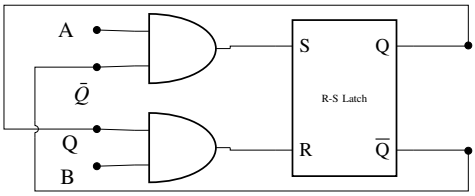


Fig. 5.26

30. For the fallowing circuit Fig. 5.27, the correct logic values for the entries $X2$ and $Y2$ in the truth table in Table 5.2 are (GATE PH 2019)
    a) 1 and 0
    b) 0 and 0
    c) 0 and 1
    d) 1 and 1

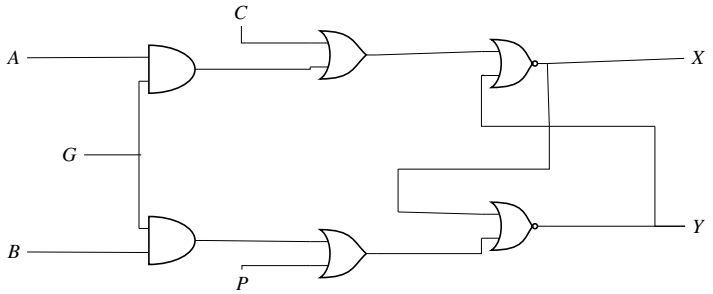


Fig. 5.27

G	A	B	P	C	X	Y
1	0	1	0	0	0	1
0	0	0	1	0	X2	Y2
1	0	0	0	1	0	1

TABLE 5.2

We explain a state machine by deconstructing 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.

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 3.4, where the present state is denoted by the variables W, X, Y, Z and the next state by A, B, C, D.

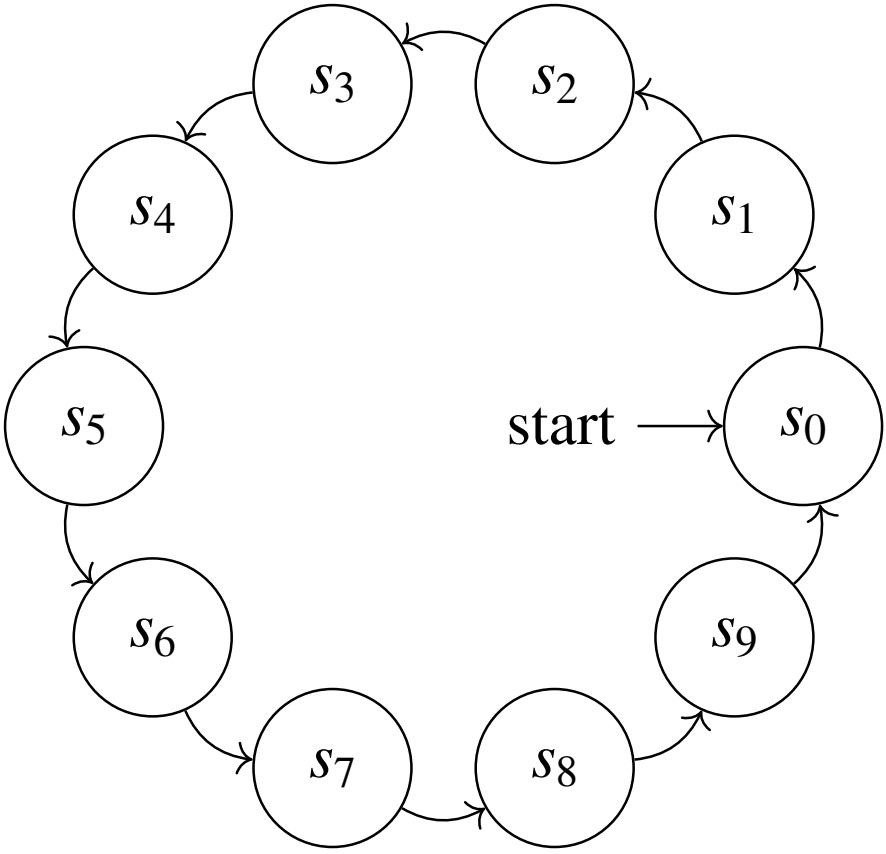


Fig. 6.1: FSM for the decade counter

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

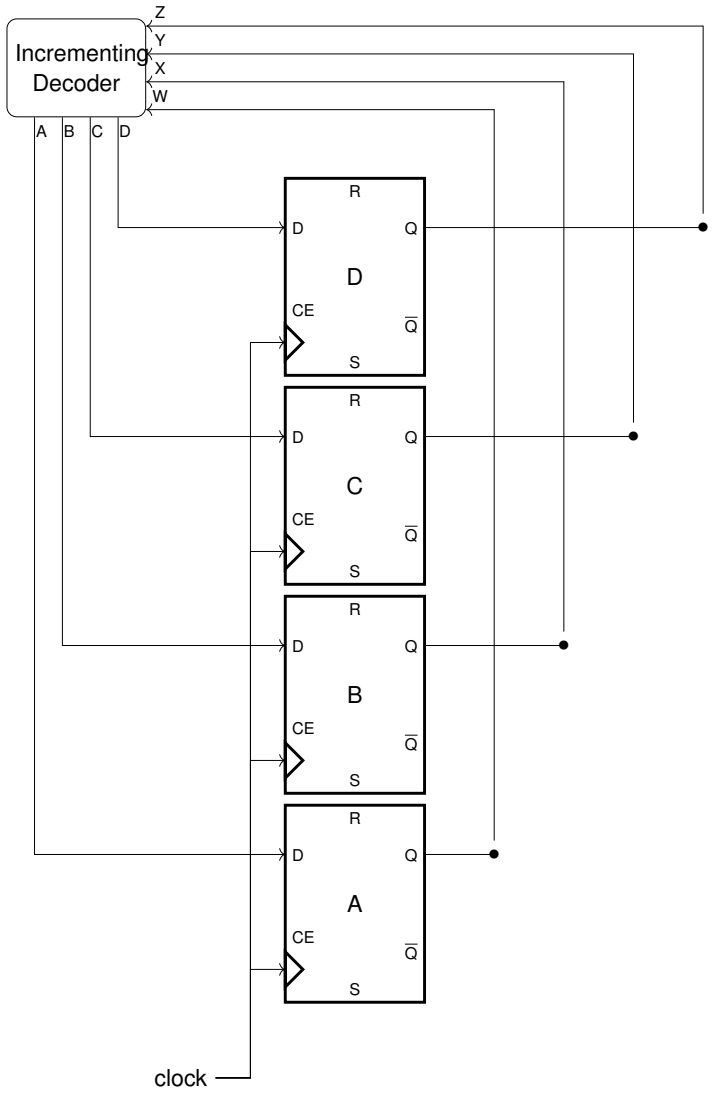


Fig. 6.2: Decade counter FSM implementation using D-Flip Flops

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 of flip flops 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.

6.1 Problems

1. The state diagram of a sequence detector is shown in Fig. 6.3. State S_0 is the initial state of the sequence detector. If the output is 1, then (GATE EC 2020)
   a) the sequence 01010 is detected
   b) the sequence 01011 is detected
   c) the sequence 01110 is detected
   d) the sequence 01001 is detected

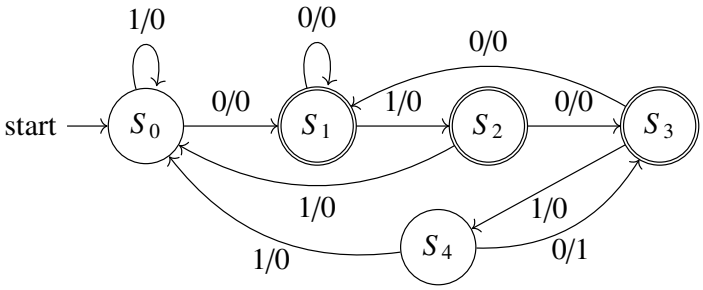


Fig. 6.3

2. A sequence detector is designed to detect precisely 3 digital inputs, with overlapping sequences detectable. For the sequence $(1,0,1)$ and input data $(1,1,0,1,0,0,1,1,0,1,0,1,1,0)$, what is the output of this detector?
   a) 1,1,0,0,0,0,1,1,0,1,0,0
   b) 0,1,0,0,0,0,0,1,0,1,0,0
   c) 0,1,0,0,0,0,0,1,0,1,1,0
   d) 0,1,0,0,0,0,0,0,1,0,0,0

   (GATE EE 2020)
3. Consider a 3-bit counter, designed using T flip-flops, as shown below in Fig. 6.4. Assuming the initial state of the counter given by *PQR* as 000,what are the next three states? (GATE CS 2021)
   a) $011, 101, 000$
   b) $010, 101, 000$
   c) $010, 101, 000$
   d) $010, 101, 000$

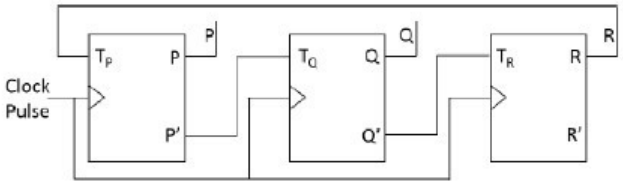


Fig. 6.4

4. The state transition diagram for the circuit shown in Fig. 6.5 is (GATE IN 2019)

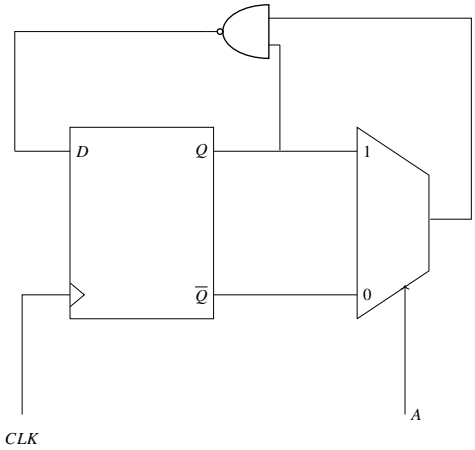


Fig. 6.5

a) Fig. 6.6

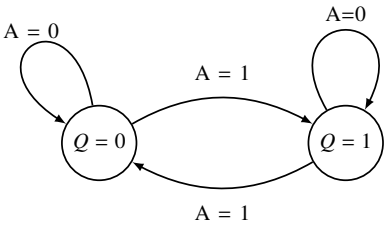


Fig. 6.6

b) Fig. 6.7

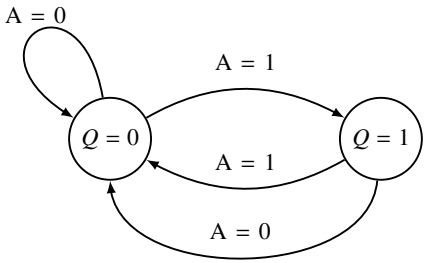


Fig. 6.7

c) Fig. 6.8

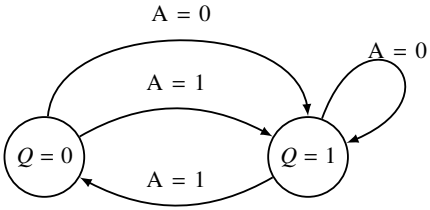


Fig. 6.8

d) Fig. 6.9

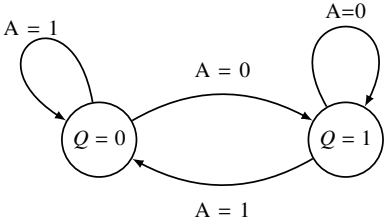


Fig. 6.9

5. A finite state machine (FSM) is implemented using the D flip-flops A and B, and logic gates, as shown in Fig. 6.10 below. The four possible states of the FSM are $Q_A Q_B$ = 00, 01, 10 and 11. Assume that X_{IN} is held at a constant logic level throughout the operation of the FSM. When the FSM is initialized to the state $Q_A Q_B$ = 00 and clocked, after a few clock cycles, it starts cycling through (GATE EC 2017)
   a) all of the four possible states if $X_{in} = 1$
   b) three of the four possible states if $X_{in} = 0$
   c) only two of the four possible states if $X_{in} = 1$

d) only two of the four possible states if $X_{in} = 0$

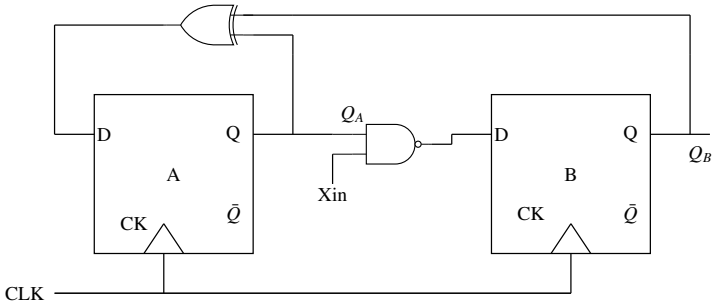


Fig. 6.10

6. Find the states in Fig. 6.11. (GATE EC 2020)

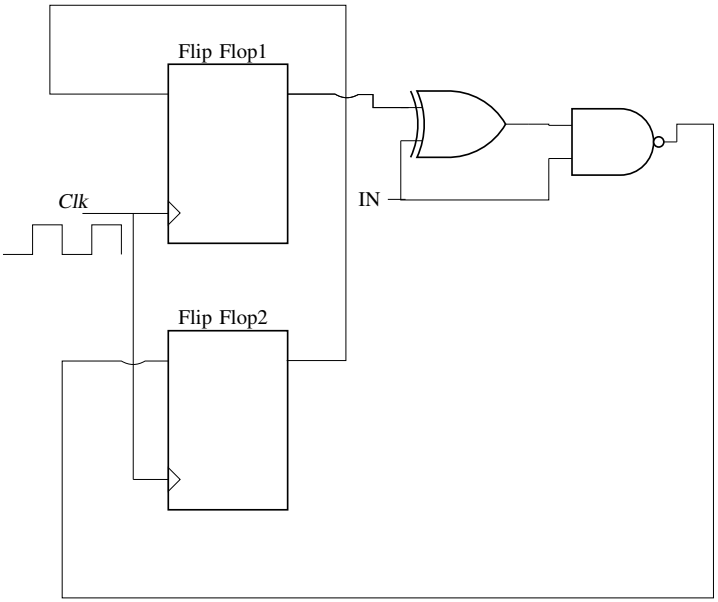


Fig. 6.11

7 Assembly Programming

7.1 Setup

We show how to setup the assembly programming environment for the arduino.

1. Copy the .inc file to your home directory

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

2. Execute

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

   Make sure that the path to **m328Pdef.inc** is correctly given in **hello.asm**.

3. Then flash the .hex file

```
hello.hex
```

4. You should see the led beside pin 13 light up.
5. Now edit **hello.asm** by modifying the line to

```
ldi r17,0b00000000
```

   Save and execute. The led should turn off.

6. 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).

7. What do the following instructions do?

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

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

7.2 Seven Segment Display

We show how to control a seven segment display through AVR-Assembly.

1. See Table 2.1 for components.
2. Complete Table 7.1 for all the digital pins using Fig. 7.1.

Port Pin	Digital Pin
PD2	2
PB5	13

TABLE 7.1

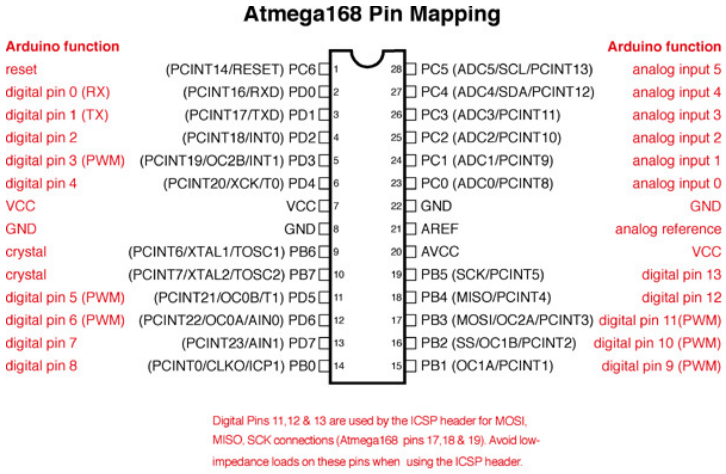


Fig. 7.1

3. Make connections according to Table 7.2.

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

TABLE 7.2

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

assembly/sevenseg/codes/sevenseg.asm

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

7.3 7447

We show how to program the 7447 BCD-Seven segment display decoder through AVR-Assembly.

1. Verify the AND,OR and XOR operations in assembly using the following code and making pin connections according to Table 9.2

assembly/7447/count/codes/and_or_xor.asm

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.

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.

$$A = W' \tag{7.1}$$
$$B = WX'Z' + W'X \tag{7.2}$$
$$C = WXY' + X'Y + W'Y \tag{7.3}$$
$$D = WXY + W'Z \tag{7.4}$$

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

7.4 Display Control

We show how to program the 7447 BCD-Seven segment display decoder through AVR-Assembly.

1. Connect the 7447 IC to the seven segment display.
2. Make connections between the 7447 and the arduino according to Table 9.2.
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 7.3. 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 7.3

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.

7.5 Blink through TIMER

We show how to use the Atmega328p timer to blink the builtin led with a delay.

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

$$TCCR0B = 0b101 \tag{7.5}$$

$$\implies CLK = \frac{SYSCLK}{1024} \tag{7.6}$$

$$= \frac{16M}{1K} = 16kHz. \tag{7.7}$$

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

$$delay = \frac{CLK}{lp2 \times PAUSE} seconds \tag{7.8}$$

$$= \frac{16 \times 1024}{64 \times 256} seconds = 1 second \tag{7.9}$$

7.6 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

$$delay \approx 80 \times 256 \times 256 cycles \tag{7.10}$$

$$= \frac{80 \times 256 \times 256}{2^4 \times 2^20} seconds \tag{7.11}$$

$$= 0.3125 seconds \tag{7.12}$$

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.

7.7 Memory

This 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.

7.8 Problems

1. In a given 8-bit general purpose micro-controller there are following flags. *C*-Carry, *A*-Auxiliary Carry, *O*-Overflow flag, *P*-Parity (0 for even, 1 for odd) *R*0 and *R*1 are the two general purpose registers of the micro-controller. After execution of the following instructions, the decimal equivalent of the binary sequence of the flag pattern [*CAOP*] will be _____. (GATE EE 2023)

```
MOV R0,+0x60
MOV R1,+0x46
ADD R0,R1
```

2. Consider the given C-code and its corresponding assembly code, with a few operands U1-U4 being unknown. Some useful information as well as the semantics of each unique assembly instruction is annotated as inline comments in the code.

```
int a[10],b[10],i;
//int is 32−bit
for (i=0;i<10;i++)
a[i]=b[i]∗8;
```

```
;r1−r5 are 32−bit integer registers
;initialize r1=0,r2=0
;initialize r3,r4 with base address of a,b

L01:jeq r1,r2,end ;if(r1==r2) goto end
L02:lw r5,0(r4) ;r5<−Memory[r4+0]
L03:shl r5,r5,U1 ;r5<−r5<<U1
L04:sw r5,0(r3) ;Memory[r3+0]<− r5
L05:add r3,r3,U2 ;r3<−r3+U2
L06:add r4,r4,U3
L07:add r1,r1,1
L08:jmp U4 ;goto U4
L09:end
```

Which one of the following options is a CORRECT replacement for operands in the position (U1,U2,U3,U4) in the above assembly code?

a) (8,4,1,L02)
b) (3,4,4,L01)
c) (8,1,1,L02)
d) (3,1,1,L01)

3. An 8085 microprocessor accesses two memory locations (2001H) and (2002H), that contain 8-bit numbers 98H and B1H, respectively.The following program is executed:

```
LXI H,2001H
MVI A,21H
INX H
ADD M
INX H
MOV M,A
HLT
```

At the end of this program, the memory location 2003H contains the number in decimal form _____. (GATE EE 2020)

4. Which of the following is the correct binary equivalent of the hexadecimal F6C? (GATE PH 2020)

a) 011011111100
b) 111101101100
c) 110001101111

d) 011011000111

5. A portion of an assembly language program written for an 8-bit microprocessor is given below along with explanations. The code is intended to introduce a software time delay. The processor is driven by a 5 MHz clock. The time delay (in μs) introduced by the program is (GATE IN 2018)

```
MVI B, $64$H; Move immediate the given byte into register B. Takes 7 clock
      periods.
LOOP: DCR B ; Decrement register B. Affects Flags. Takes 4 clock periods.
JNZ LOOP ; Jump to address with Label LOOP if zero flag is not set. Takes 10
      clock periods when jump is performed and 7 clock periods when jump is not
      performed.
```

6. A $10\frac{1}{2}$ digit Counter-timer is set in the 'frequency mode' of operation (with $T_s = 1s$). For a specific input, the reading obtained is 1000. Without disconnecting this input, the Counter-timer is changed to operate in the 'Period mode' and the range selected is microseconds (μs, with $f_s = 1$ MHz).The Counter will then display

a) 0
b) 10
c) 100
d) 1000

(GATE IN 2021)

7. Consider three registers R1 ,R2 ,R3 that store numbers in IEEE-754 single precision floating point format. Assume that R1 and R2 contain the values (in hexadecimal notation) 0x42200000 and 0xC1200000, respectively. If R3=$\frac{R1}{R2}$, what is the value stored in R3? (GATE EC 2021)

a) 0x40800000
b) 0xC0800000
c) 0x83400000
d) 0xC8500000

8. The content of the registers are $R_1 = 25H$, $R_2 = 30H$ and $R_3 = 40H$. The following machine instructions are executed.

```
PUSH R1
PUSH R2
PUSH R3
POP R1
POP R2
POP R3
```

After execution, the content of registers R1, R2, R3 are (GATE EC 2021)

a) $R_1 = 40H, R_2 = 30H, R_3 = 25H$
b) $R_1 = 25H, R_2 = 30H, R_3 = 40H$
c) $R_1 = 30H, R_2 = 40H, R_3 = 25H$
d) $R_1 = 40H, R_2 = 25H, R_3 = 30H$

8 Embedded C

8.1 Blink

We show how to control an led using AVR-GCC. AVR-GCC is a C compiler for the Atmega328p.

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 << *PB*5)) 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.

8.2 Display Control

We show how to control a seven segment display using AVR-GCC with arduino

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.
4. Now connect the arduino to the seven segment display through 7447.
5. Execute the following code

```
avr−gcc/input/codes/main.c
```

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

8.3 GCC-Assembly

We show how to 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.

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 8.1. 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 8.1: 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.

8.4 LCD

We show 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.

1. The required components are listed in Table 8.2

Component	Value	Quantity
Resistor	220 Ohm	1
	1K	1
Arduino	Uno	1
Jumper Wires		20

TABLE 8.2

2. Plug the LCD in Fig. 8.1 to the breadboard.
3. Connect the Arduino pins to LCD pins as per Table 8.3.

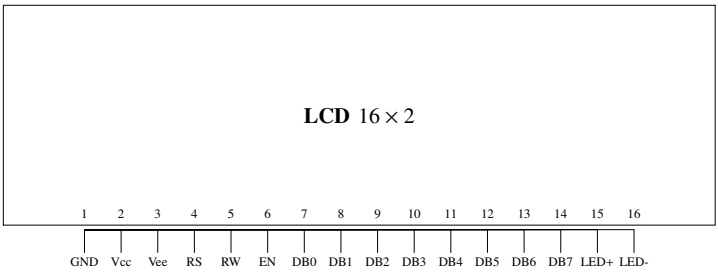


Fig. 8.1: LCD

TABLE 8.3: 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
D12	4	RS	Register Select
GND	5	R/W	Read/Write
D11	6	EN	Enable
D5	11	DB4	Serial Connection
D4	12	DB5	Serial Connection
D3	13	DB6	Serial Connection
D2	14	DB7	Serial Connection
5V	15	LED+	Backlight
GND	16	LED-	Backlight

4. Execute

```
cd avr−gcc/lcd/codes
make
```

5. Modify the above code to display a string.
6. Modify the above code to obtain a decade counter so that the numbers from 0 to 9 are displayed on the lcd repeatedly.
7. Repeat the above exercies to display a string on the first line and a number on the second line of the lcd.
8. Write assembly routines for driving the lcd.

8.5 Problems

1) The representation of the decimal number $(27.625)_{10}$ in base-2 number is

(A) 11011.110
(B) 11101.101
(C) 11011.101
(D) 10111.110

(GATE IN 2018)

9 ESP32

9.1 Arduino Droid

For flashing the ESP32 through OTG follow the below steps:

1) Install ArduinoDroid from apkpure.
2) Open ArduinoDroid and grant all permissions
3) Connect the ESP32 to your phone via USB-OTG and select the board **DOIT ESP32 DEVKIT V1** in ArduinoDroid using the below path.

```
Settings−>Board type−>ESP32−>DOIT ESP32 DEVKIT V1
```

See Fig. 9.1 for DOIT ESP32 DEVKIT V1. For ESP32 **NodeMCU** (see Fig. 9.2),

```
Settings−>Board type−>ESP32−>NodeMCU−32S
```

4) Run the blink program present in the below link

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

The in-built led will start blinking.

9.2 Platformio

1) In termux excecute the following to generate the bin file.

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

2) Copy the bin file generated to ArduinoDroid/precompiled directory

```
cp .pio/build/esp32doit−devkit−v1/firmware.bin /sdcard/ArduinoDroid/precompiled/
```

3) Flash the bin file to ESP32 using ArduinoDroid. Open ArduinoDroid,

```
Actions−>Upload−>Upload Precompiled−>firmware.bin
```

After the upload is finished we get the below error in ArduinoDroid terminal. This indicates that the code is uploaded.

```
Error: open failed: ENOENT (No such file or directory)
```

Disconnect the power supply from ESP32 and reconnect it. The onboard LED should blink.

9.3 OTA: Wireless Flashing

1) Setup your mobile hotspot with

```
username: npal
password: npal1234
```

2) Connect the esp32 to your mobile through USB.
3) Execute the following commands using platformio

```
cd esp32/ide/ota/setup
pio run
```

and follow the instructions from Item 2 onwards.

4) Check the connected devices in your hotspot settings. You should see the ESP as a connected device. Obtain the IP address of your ESP from the hotspot settings. Now execute the following in termux.

```
cd esp32/ide/ota/blink
pio run
pio run −t nobuild −t upload −−upload−port 192.168.xx.xx #replace xx with IP of
      ESP32
```

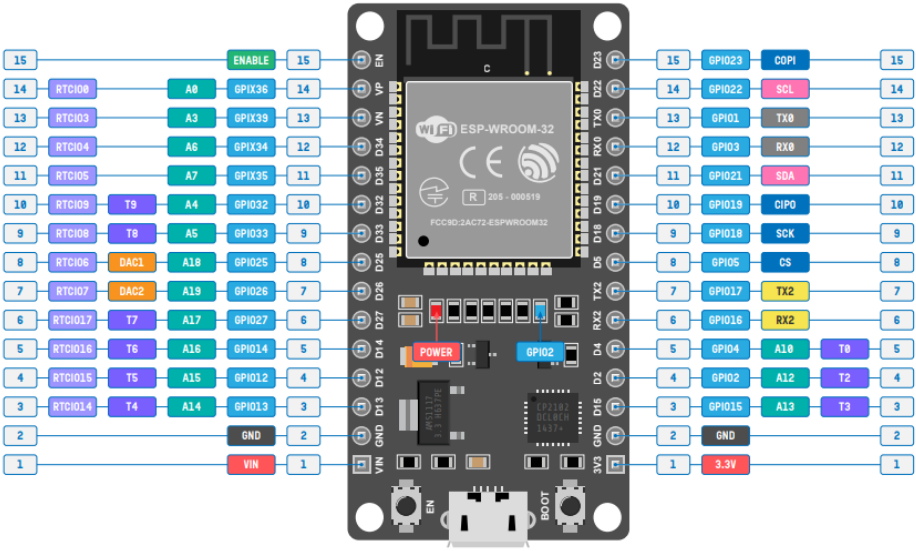


Fig. 9.1: ESP32 Devkit-v1

In the following, we will use the OTA method for flashing the ESP32 using platformio.

9.4 Seven Segment Display

1. Make connections according to Table 9.1

ESP32	13	12	14	27	26	25	33
Display	a	b	c	d	e	f	g

TABLE 9.1

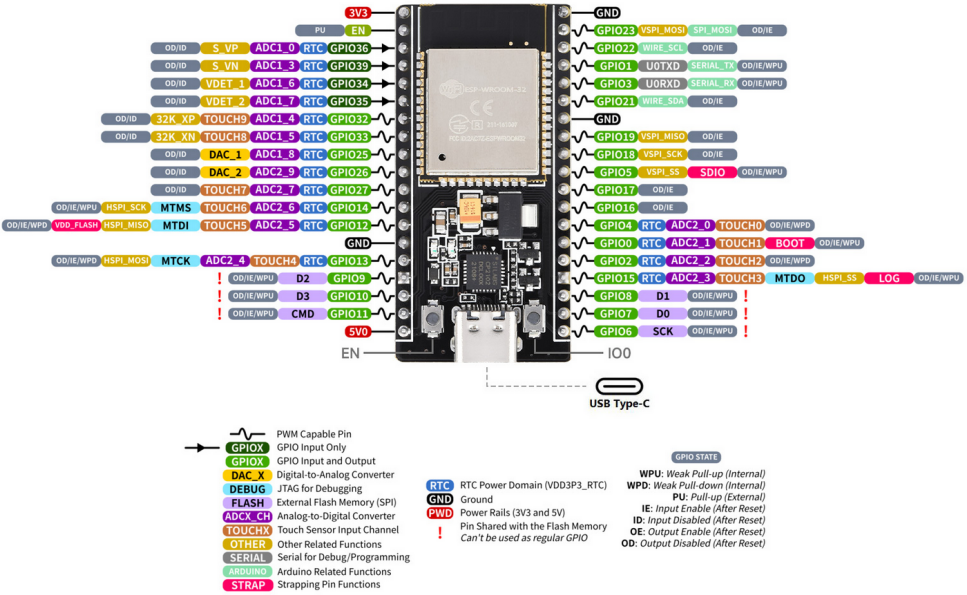


Fig. 9.2: ESP32 NodeMCU

2. Execute the following

esp32/ide/sevenseg/src/main.cpp

The number 5 should be displayed.

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

9.5 7447

1) Make the connections as per Table 9.2 and execute the following program. You should see the number 5 displayed.

/esp32/ide/7447/codes/display/src/main.cpp

7447	D	C	B	A
ESP32-Devkit-v1	27	14	12	13

TABLE 9.2

2) Now execute the following code. You should see the number 2 being displayed. This code increments the input by 1.

esp32/ide/7447/codes/inc_dec/src/main.cpp

3) Make additional connections as shown in Table **??**. You should see the number 6 displayed. The code increments the manually given input to the 7447 IC by 1.

esp32/ide/7447/codes/ip_inc_dec/src/main.cpp

	Z	Y	X	W
Input	0	1	0	1
ESP32	32	33	25	26

TABLE 9.3

9.6 7474

1) Generate the CLOCK signal using the **blink** program in the SP32.
2) Connect the ESP32, 7447 and the two 7474 ICs according to Table 9.4.

	INPUT				**OUTPUT**				CLOCK		3.3V			
	W	X	Y	Z	A	B	C	D	D2					
ESP32	D32	D33	D25	D26	D13	D12	D14	D27						
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 9.4

3) The code for Decade counter is given in the below link

https://github.com/pradyumnasv9/esp32/blob/psv/ide/7447/codes/ip_inc_dec.ino

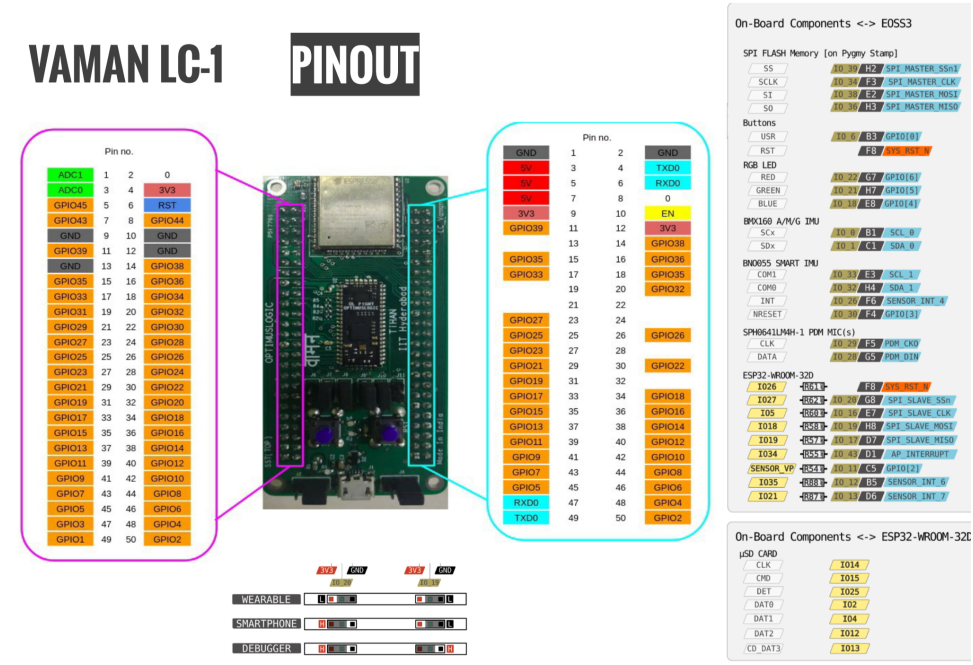


Fig. 10.1: Vaman pins. Right side reperesents ESP32 and left side M4-FPGA (Pygmy).

10 Vaman

10.1 ESP

Here we show how to program the ESP32 on the Vaman using the Arduino framework.

1. Make sure that Vaman board does not power any devices.
2. Make connections as shown in Table 10.1.
3. The Vaman pin diagram is available in Fig. 10.1.

VAMAN-ESP	UART PINS
5V	5V
GND	GND
TXD0	TXD
RXD0	RXD
0	GND

TABLE 10.1

4. Connect the UART to raspberry pi through USB.
5. Connect the pins between Vaman-ESP32 and Vaman-PYGMY as per Table 10.2
6. On termux on your phone,

ESP32	Vaman
GPIO2	GPIO18
GPIO4	GPIO21
GPIO5	GPIO22

TABLE 10.2

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

7. Transfer the ini and bin files to the rpi

```
scp platformio.ini pi@192.168.50.252:./hi/platformio.ini

scp .pio/build/esp32doit-devkit-v1/firmware.bin pi@192.168.50.252:./hi/.pio/build/
    esp32doit-devkit-v1/firmware.bin
```

8. On rpi,

```
cd /home/pi/hi
pio run -t nobuild -t upload
```

   You should see the blue led blinking.
9. Now disconnect pin 2 from pin 18 and connect to pygmy GPIO pin 21.
10. Repeat the above exercise using GPIO pin 22.
11. On your phone, open

```
src/main.cpp
```

    and change the delay to

```
delay(100);
```

    and execute the code by following the steps above.
12. Flash the following code.

```
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
13. 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

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

```
vaman/esp32/codes/ide/ota/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.

15. Flash the following code OTA

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

You should see the onboard LEDs blinking.

16. Change the blink duration to 100 ms.
17. Execute the code in

```
vaman/esp32/codes/ide/ota/sevenseg/static/
```

to control the seven segement display.

10.2 FPGA

We show how to program the Vaman FPGA/microcontroller board.

1. Follow the instructions available in the video at

```
https://github.com/whyakari/TermuxDisableProcces?tab=readme−ov−file
```

to ensure that termux is not killed during the following installation process.

2. On termux-debian,

```
wget https://raw.githubusercontent.com/gadepall/fwc−1/main/scripts/setup.sh
bash setup.sh
```

3. Login to termux-debian on the android device and execute the following commands

```
cd vaman/fpga/setup/codes/blink
source ~/.vamenv/bin/activate
ql_symbiflow −compile −src vaman/fpga/setup/codes/blink −d ql−eos−s3 −P
    PU64 −v helloworldfpga.v −t helloworldfpga −p pygmy.pcf −dump binary
scp blink/helloworldfpga.bin pi@192.168.0.114:
```

Make sure that the appropriate IP address for the raspberry pi is given in the above command.

4. Put the Vaman board in download mode. For this, you need to first press the button to the right of the usb port and immediately press the button to the left. The green led should now flash and you can go to the next step.
5. Now execute the following commands on the raspberry pi.

```
python3 -m venv ~/.vamenv
source ~/.vamenv/bin/activate
git clone --recursive https://github.com/QuickLogic-Corp/TinyFPGA-
    Programmer-Application.git
pip3 install tinyfpgab
deactivate
sudo reboot
source ~/.vamenv/bin/activate
python3 TinyFPGA-Programmer-Application/tinyfpga-programmer-gui.py --
    port /dev/ttyACM0 --appfpga /home/pi/helloworldfpga.bin --mode fpga --
    reset
```

6. Make sure that the correct USB port address is given in the above command. After some time, the LED will start blinking red.
7. Replace the following line in the code in instruciton 10

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

with

```
assign blueled = led;
```

and execute the code.

8. In the following .pcf file,

```
codes/blink/pygmy.pcf
```

the pin numbers for the 3 colour-leds are defined. See Table 10.2 and Fig. 10.2. The IO locations in Fig. 10.2 can be found in pygmy.pcf while the aliases (GPIO) are printed on the board.

9. Now modify the helloworldfpga.v file to get the green led blinking.
10. In the following verilog program,

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

pay attention to the following lines

```
delay = delay+1;
if(delay > 20000000)
begin
delay=25'b0;
led=!led;
end
```

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

11. In instruction 10, replace

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

Type	Vaman Pin	Connection
Input	IO_28	3.3V

TABLE 10.3: Vaman Input/Output.

with

```
if(delay==25'b10011000100101101000000000)
```

and execute the verilog code.

12. Since the delay is 20 MHz, the blink period is 1 second. Modify the verilog code so that the blink period becomes 0.5s.
13. Find the bit length of 20 MHz.
    Solution:

$$\log_2(20000000) \approx 25 \tag{10.1}$$

14. Obtain the above answer using a Python code.
    Solution: Execute the following code and compare with instruction 11.

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

15. Ensure that the LED stays on in green colour.
    Solution: Execute the following code

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

16. Execute the following code and make pin connections as per Table 10.3. Take out the input pin connect to 3.3V. Plug it again. Do this repeatedly.

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

17. Connect an external LED and repeat.
18. Execute the codes in

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

to control the seven segment display.

10.3 ARM

We show how to control an LED using the M4 on Vaman.

1. Check your path

```
cd vaman/arm/setup/blink/GCC_Project
nvim config.mk
```

and

2. and modify so that you have the following lines

PD64		
IO Locatic	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 Locatic	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 Locatic	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

Fig. 10.2: Pin Definitions

```
#export PROJ_ROOT=/data/data/com.termux/files/home/pygmy-dev/pygmy-sdk
export PROJ_ROOT=/root/pygmy-dev/pygmy-sdk
```

3. Now execute

```
cd vaman/arm/setup/blink/GCC_Project
make -j4
scp output/bin/blink.bin pi@192.168.0.114:
```

Appropriately modify the above ip address before sending blink.bin to the pi.

4. Now log on to the RPi and execute the following

```
sudo python3 /home/pi/Vaman-dev/Vaman-sdk/TinyFPGA-Programmer-
    Application/tinyfpga-programmer-gui.py --port /dev/ttyACM0 --m4app
    blink.bin --mode m4-fpga
```

5. Enter the appropriate USB device port above while executing. Press the button to the right after the above command is successfully executed. The LED will start blinking.
6. 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.

7. Replace the following line in 6

```
        HAL_DelayUSec(2000000);
```

with

```
        HAL_DelayUSec(1000000);
```

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

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

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

Now obtain blue colour.

9. Now obtain green colour without blink.
Solution: Execute the following code.

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

10. Using Table 10.4 and Fig. 10.1, use an input pin to control the onboard LED.
Solution: Execute the following code. You should see the green LED on. Connecting IO_5 to GND will turn the green LED off.

Type	Pin	Destination
Input	IO_5	5V

TABLE 10.4: Vaman control through external input.

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

11. Execute the codes in

vaman/arm/codes/sevenseg/loop/

to control the seven segment display.

11 PicoW

11.1 Arduino Framework

1. The following instructions are for the picow toolchain in termux-debian

```
cd .platformio/packages
git clone https://github.com/earlephilhower/arduino-pico
cd arduino-pico
git submodule update --init
cd pico-sdk
git submodule update --init
cd ../tools
python3 ./get.py
```

2. Downlaod EtchDroid from F-DROID.
3. Now execute the following

```
cd picow/ide/codes/
pio run
cp .pio/build/rpipicow/firmware.uf2 /sdcard/Download/
```

4. Keep BOOTSEL pressed on picow and connect it to your phone.
5. Now flash picow using EtchDroid.
6. The Onboard LED will start blinking.
7. Change the blink frequency.
8. If Item 1 is unsuccessful,
   a) Execute Item 3 on your rpi
   b) Copy relevant platformio packages from your raspberry pi to your phone .platformio/packages directory using scp.
9. See Fig. 11.1 for the picow pin diagram. You may reuse all the Arduino codes for pico by changing only the platformio.ini file.

12 Pico

12.1 Setup

1. There is a button adjacent to the USB port of the pico. Keep this button (BOOTSEL) pressed while connecting the RPi to the Pico through the USB cable.
2. Login to termux-debian and execute the following commands.

```
cd pico/trunk/arm/codes/setup/blink
mkdir build #Only once
cd build
cmake ..
make -j4
scp main.uf2 pi@192.168.0.114:
```

   Suitably modify the above ip address before sending main.uf2 .
3. Now login to the raspberry pi and execute the following commands.

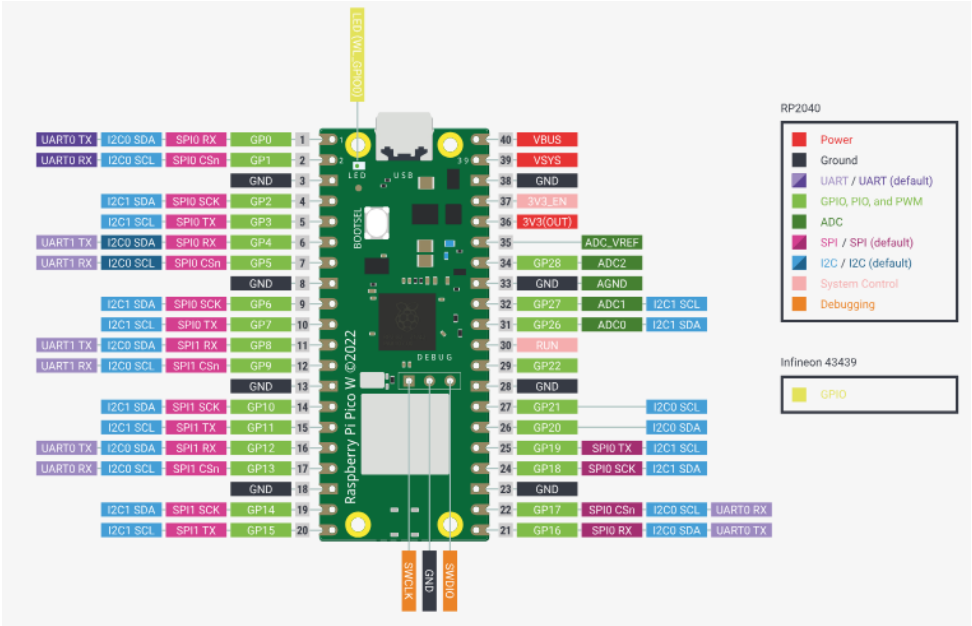


Fig. 11.1: PICOW Pin Diagram

```
sudo mkdir /mnt/pico #Only once
sudo fdisk –l
sudo mount /dev/sda1 /mnt/pico
sudo mv /mnt/pico
sudo umount /mnt/pico
```

You should now see the LED to the right of the USB port blinking.

4. Connect RUN on pico to GND. Keep pressing BOOTSEL while removing the RUN-GND wire from GND. The LED stops blinking. Pico is now ready to be flashed.

12.2 Delay

1. Note the followign lines in the C code below:

codes/setup/blink/main.c

```
        gpio_put(LED_PIN, 1);
        sleep_ms(250);
        gpio_put(LED_PIN, 0);
        sleep_ms(250);
```

It is obvious that the blink period is 500ms = 0.5s

2. Replace the below instruction in the C program

```
sleep_ms(250);
```

with

```
sleep_ms(500);
```

Can you see any difference in the LED blinking frequency?

3. Now modify the above code to keep the LED on permanently.
   Solution: Execute the following code.

pico/codes/setup/onoff/main.c

4. Use GP2 as an output pin and drive an LED.
   Solution: Connect the LED to pico as per Table 12.1 and Fig. 12.1

pico/codes/setup/gpio/main.c

Type	pico pin	Destination
Output	3V3	LED
Output	GP2	LED

TABLE 12.1: Connection between LED and pico

5. Execute the following code. Connect a wire to GND and touch GP2 on the pico. The onboard LED will turn off. Repeat this exercise to blink the LED manually.

pico/codes/setup/input/main.c

6. Execute the following code to drive the seven segment display.

pico/arm/codes/sevenseg/static/main.c

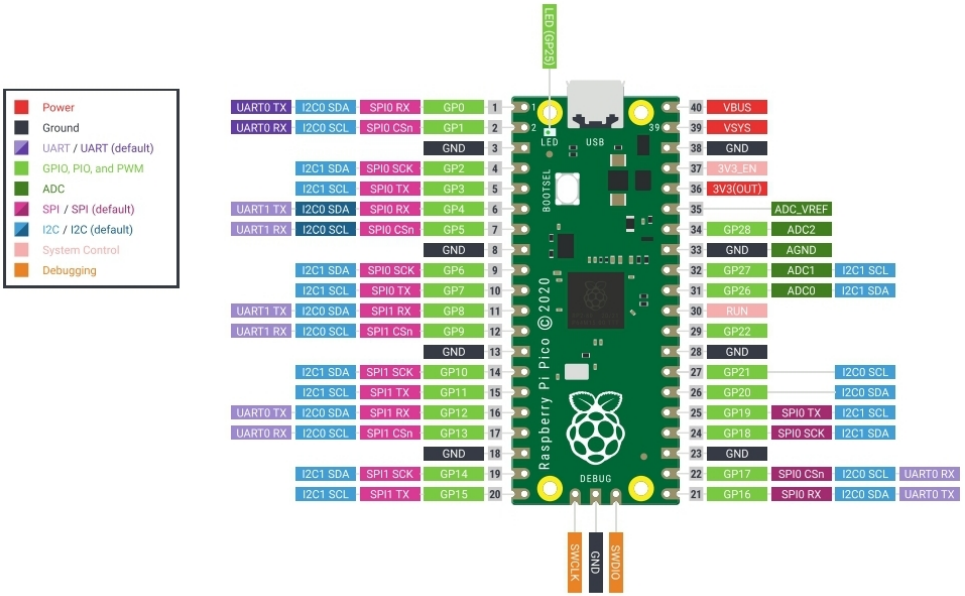


Fig. 12.1: Pin diagram

13 STM32-Arduino

13.1 Arduino Framework

1. The STM32F103C8T6 micro-controller in Fig. 13.1 has two ground pins, few analog input pins and few digital pins that can be used for both input as well as output. It has one Vcc (3.3V) pin that can generate $3.3V$. In the following exercises, only the GND, $3.3V$ and digital pins will be used.
2. Make the connections as per the Table 13.1

STM32F103C8 PINS	USB2UART PINS
3.3	3.3
GND	GND
A9(TXD)	RXD
A10(RXD)	TXD

TABLE 13.1: STM32 to USB-UART connections

3. Basic onchip led blink program is available at

```
stm32/ide/setup/codes/src/blink.c
```

4. Make sure that platformio.ini file has the following lines

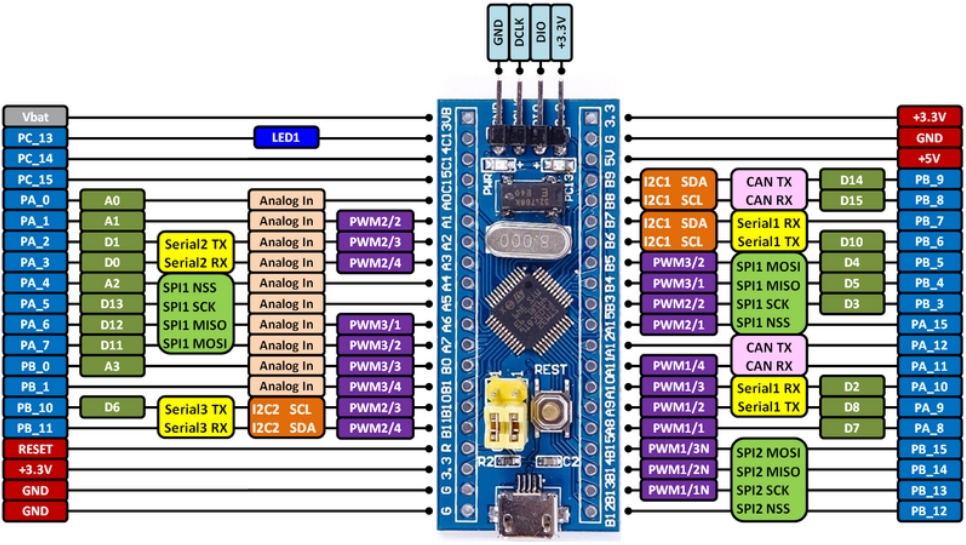


Fig. 13.1: Pin Connections of STM-32

```
[env:bluepill_f103c8]
platform = ststm32
framework = arduino
board = bluepill_f103c8
```

5. Now, complile the blink program

```
cd stm32/ide/setup/codes
pio run
```

6. With this .bin file will be generated and all the necessary packages will be installed.
7. Before flashing, make sure that STM32 board is in Programming mode
8. See how to enable programming mode in Fig. 13.2. This configuration will ensure that the microcontroller enters the system memory programmer on reset.
9. After executing

```
Install STM32-Utils app from playstore
Give the necessary permissions
Connect the USB-UART to OTG
Upload the .bin file to the STM32 board
```

10. The inbuilt LED present on the STM32F103C8T6 will start blinking.
11. Change the STM32 board to operating mode as shown in Fig 13.2 in order to run your code from the main flash memory.

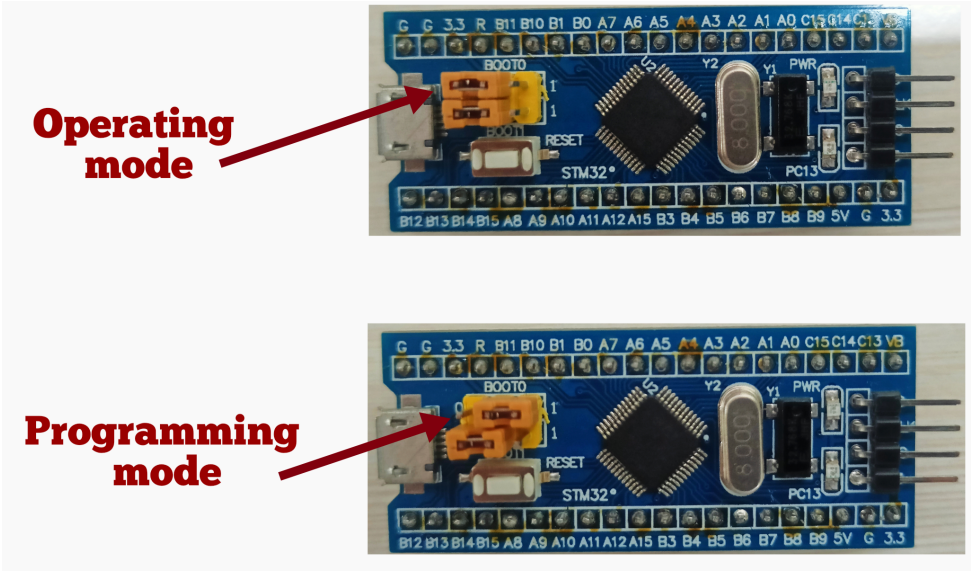


Fig. 13.2: STM32 Operating and Programming mode

14 STM32-GCC

14.1 Embedded C

1. Copy your codes directory to the termux-debian root directory.

```
cp −r STM32F103C8T6 ~/STM32F103C8T6/
cd ~/STM32F103C8T6
make bin
```

2. This will generate main.bin. Flash it using the android app or platformio.
3. Modify main.c in the STM32F103C8T6 directory and modify the code to keep the LED on. Flash it to the STM32 and verify.

14.2 Seven Segment Display

1. Make the pin connections in Table 14.2 using Figs. 2.2 and 13.1.

STM32	**PB9**	**PB8**	**PB7**	**PB6**	**PB5**	**PB4**	**PB3**	**3.3V**
DISPLAY	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**COM**
0	0	0	0	0	0	0	1	
1	1	0	0	1	1	1	1	
2	0	0	1	0	0	1	0	
3	0	0	0	0	1	1	0	
4	1	0	0	1	1	0	0	
5	0	1	0	0	1	0	0	
6	0	1	0	0	0	0	0	
7	0	0	0	1	1	1	1	

5. The previous instructions set the bits in the unused ports PB15-PB10 and PB2-PB0. This may be undesirable in some cases. Generate 0 by not disturbing the unused pins.
   Solution: The following instructions help accomplish this. The first instruction resets PB4-PB9. The second instruction sets the PB3 pin. The other pins are undisturbed.

```
GPIOB->BRR = (1<<4)|(1<<5)|(1<<6)|(1<<7)|(1<<8)|(1<<9); // (Led ON)
GPIOB->BSRR = (1<<3); // (Led OFF)
```

6. Write a program to take a 4-bit BCD as input from hardware (GND or VDD) and show the next number on the seven segment display.
   Solution: The following program takes 4 bits as input from pins PB12-PB15 and displays the output on a seven segment display. The next number can be displayed by slightly modifying the code.

```
stm32/sevenseg/codes/bin2dec_example.c
```

14.3 GPIO Output

1. Connect the USB-UART to STM32. The hardware connections between the USB-UART, STM32 and LED are available in Table 14.3. Make sure that the LED is connected to PB4 through the resistor.

STM32	UART	LED
GND	GND	-
3.3V	3.3V	
A9(Txd)	Rxd	
A10(Rxd)	Txd	
PB4		+

TABLE 14.3: USB-UART to STM32 connections

2. Execute

```
gpio/codes/gpio_example.c
```

3. Modify the above program to turn the LED off.
4. Explain the following line.

```
AFIO->MAPR = AFIO_MAPR_SWJ_CFG_JTAGDISABLE;
```

   Solution: By default, the PB3 and PB4 pins in the STM32F103C8T6 board cannot be used as GPIO pins. The above command allows these pins to be configured for GPIO.
5. What does the following command do?

```
GPIOB->CRL = 0x00030000;
```

Solution: The STM32F103C8T6 has ports A and B, each having 16 pins that can be used as GPIO output. The above command enables the pin B4 of port B as an output pin See Tables 14.5.

Pin	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB70
Nibble	0	0	0	3	0	0	0	0

TABLE 14.5: GPIO Enable pins

6. Explain the significance of the number (nibble) 0x3 corresponding to PB4 in Table 14.5.
   Solution: The nibble 0x3 = 0b0011. From Table 14.6, The first two bits are CNF1=0, CNF0=0 which means that PB4 is configured as a general purpose push-pull output. The last two bits are 11, denoting the mode, which says that PB4 is capable of a maximum output speed of 50 MHz.

Configuration Mode		**CNF1**	**CNF0**	**Mode**	**Max Speed (Mhz)**
General Purpose Output	Push-pull	0	0	01	10
	Open-drain		1	10	2
				11	50
				00	reserved

TABLE 14.6: GPIO Output configuration

7. What does the following instruction do?

```
GPIOB->BRR = (1<<4);
```

   Solution: BRR is the Bit Reset Register. The least significant 16 bits are used to atomically set pin values to GND whereas the most significant 16 bits are used to atomically clear pin values to VDD. The above command clears PB4.

8. What does the following instruction do?

```
GPIOB->BSRR = GPIO_BSRR_BR4;
```

   Solution: BSRR is the Bit Set/Reset Register. GPIO_BSRR_BR4 is used to reset PB4. The result is the same as the previous problem.
9. Modify your program to control an LED using PA2.

14.4 GPIO Input

10. Connect PB7 to GND and execute the following code

```
cd /gpio/gpio_input_example.c
```

11. What does the following instruction do?

```
GPIOB->CRL = 0x80030000;
```

Solution: This instruction declares PB7 as an input pin. 0x8=0b1000. According to Table 14.7, the mode 00 is used only for input and the first two bits 10 are used for pull-up/pull-down.

Configuration Mode		CNF1	CNF0	Mode
Input	Analog	0	0	00
	Input Floating		1	
	Input pull-down	1	0	
	Input pull-up			

TABLE 14.7: GPIO Input configuration

12. Explain the following instruction.

```
#define B7 0x0080
if (((GPIOB->IDR & B7) == 0 ))
        GPIOB->BRR = (1<<4); //PB4 = 0 (Led ON)
else
        GPIOB->BSRR = (1<<4); //PB4 = 1 (Led OFF)
```

Solution: The instruction checks whether the 8th bit of GPIOB− > IDR, i.e. the input from PB7 is 0. If so, then the LED connected to PB4 should be ON.

14.5 Clocks

1. List all available clocks in the STM32F103C8T6 blue pill.
   Solution: See Table 14.8.

Clock	Location	Type	Frequency
HSI	Internal	RC	8Mhz
LSI	Internal	RC	32.768 kHz
HSE	External	Crystal	8Mhz

TABLE 14.8: STM32F103C8T6 Clock Types

2. Make connections as shown in Table 14.9.

STM32	Seven Segment Display
3.3V	COM (through resistor)
PA1	DOT

TABLE 14.9: Pin Connections

14.6 HSE

3. Execute the following program

clocks/codes/hse_systick_blink.c

4. Explain the following instruction

RCC−>CR =0x00010000;

Solution: Fig. 14.1 shows the RCC− >CR register. The above instruction enables the HSE crystal, which is 8 MHz for the STM32F103C8T6.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						PLL RDY	PLLON	Reserved				CSS ON	HSE BYP	HSE RDY	HSE ON
						r	rw					rw	rw	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7:0]								HSITRIM[4:0]					Res.	HSI RDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

Fig. 14.1: RCC Clock Control Register (RCC− >CR)

5. Explain the following instruction

RCC−>CFGR =0x00000001;

Solution: Fig. 14.2 shows the RCC− >CFGR register. The above instruction makes the HSE as the system clock through SW = 01.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					MCO[2:0]			Res.	USB PRE	PLLMUL[3:0]				PLL XTPRE	PLL SRC
					rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCPRE[1:0]		PPRE2[2:0]			PPRE1[2:0]			HPRE[3:0]				SWS[1:0]		SW[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw

Fig. 14.2: RCC clock Configuration Register (RCC− >CFGR)

6. Verify that HSE is the system clock by checking that SWS = 01.

14.7 PLL

7. Make the PLL as the system clock.
Solution:

RCC−>CFGR =0x00000010;

8. Choose the PLL input as HSE.

RCC−>CFGR =0x00010010;

9. Enable PLL
   Solution:

```
RCC->CR =0x01010000;
```

10. Execute the following code.

```
cd /clocks/pll_systick_blink.c
```

11. Make the PLL output = 24 MHz.

14.8 Timers

1. List all the available timers in the STM32F103C8T6 blue pill.
   Solution: See Table 14.10

Timer	**Type**	Counter Resolution
Systick	Default	24 bit
Independent	Watchdog	12 bit
Window	Watchdog	7 bit
TIM1	Advanced	16 bit
TIM2	General Purpose	
TIM3		
TIM4		

TABLE 14.10: STM32F103C8T6 Timer Types.

14.9 Systick timer

2. The Systick timer is the default timer available on all ARM chips.
3. Make connections as shown in Table 14.11.

STM32	**Seven Segment Display**
3.3V	COM (through resistor)
PA1	DOT

TABLE 14.11: Pin Connections

4. Execute the program in

```
timers/codes/blink_systick.c
```

5. The default clock is the HSI 8MHz RC. Find the number of clock cycles required for a 1 s delay.
   Solution: The time period is

$$T = \frac{1}{8}\mu s = 1 \text{ cycle} \tag{14.1}$$

Thus, the number of cycles required for 1 s delay is

$$1 \text{ second} = 8000000 \text{ cycles} \tag{14.2}$$

6. List the SysTick registers.
   Solution: See Table 14.12.

Register	Command	Purpose
SysTick Control and Status	SysTick->CTRL	Timer control
SysTick Reload Value	SysTick->LOAD	Timer Count
SysTick Current Value	SysTick->VAL	Timer Initialize
SysTick Calibration Value		

TABLE 14.12: Systick Registers

7. What do the following instructions do?

```
SysTick->LOAD = 4000000;
SysTick->VAL = 0;
```

   Solution: See Table 14.12 for details. These two instructions ask the SysTick timer to count down from 4000000 to 0.
8. Explain the following instruction.

```
while(!(SysTick->CTRL & 0x00010000));
```

   Solution: Fig. 14.3 shows the SysTick CTRL register. 0x00010000 is used in the above command to mask all the bits except for bit 16, which is the COUNTFLAG. The **while** loop will stop once COUNTFLAG = 0. The while loop is used for the delay.

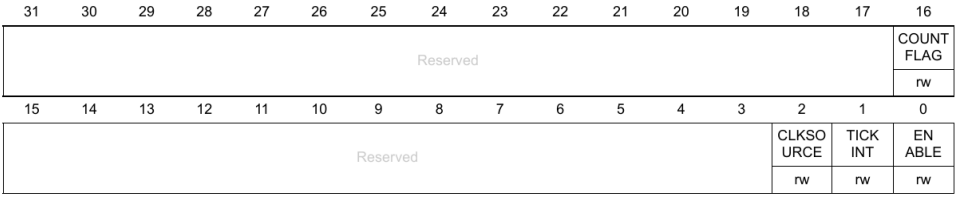


Fig. 14.3: Control Register (CTRL)

9. What does the following instruciton do?

```
SysTick->CTRL = 0x00000005; //8MHz clock
```

   Solution: From Fig. 14.3, ENABLE = 1 enables the counter (for delay) and CLKSOURCE = 1 enables the 8 MHz internal RC clock.
10. Obtain a 1 MHz clock.
    Solution: CLKSOURCE = 1 results in the $\frac{\text{Processor Clock}}{8}$ = 1 MHz clock.

```
SysTick->CTRL = 0x00000001; //1MHz clock
```

11. Obtain a delay of 1 second using the 1 MHz clock.

12. Make the connections according to Table 14.11. Execute the following program

```
cd /timers/timer1_blink.c
```

13. Enable Timer1 through RCC.
Solution:

```
RCC->APB2ENR |= RCC_APB2ENR_TIM1EN;
```

14. Select the HSI clock of 8 MHz as TIM1 clock.
Solution: See Fig. 14.4 for register. SMS=000 implies that the TIM1 will be controlled by the internal clock.

```
TIM1->SMCR = 0;
```

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	Res.	rw	rw	rw

Fig. 14.4: Slave Mode Control Register (SMCR)

15. What does the following instruction do?

```
TIM1->CR1  = 0x0001;
```

Solution: Fig. 14.5 shows the control register 1 (CR1). CEN=1 enables the counter.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Fig. 14.5: Control Register 1 (CR1)

16. Make TIM1 clock = 2 KHz.
Solution: Through the following command,

```
TIM1->PSC = 3999;
```

$$TIM1\_CLK = \frac{HSI\_CLK}{TIM1->PSC + 1} = \frac{8000000}{4000} \tag{14.3}$$

Fig. 14.6 shows the TIM1− >PSC (prescalar) register.
17. What is the maximum value that can be stored in TIM1− >PSC?
18. Make TIM1 count 1000 cycles of the 2 KHz TIM1 clock.
Solution:

```
TIM1->ARR  = 999;
```

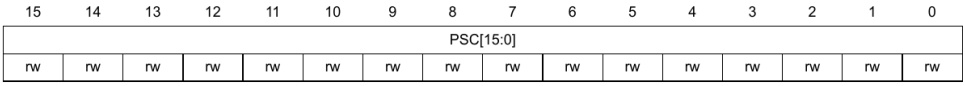


Fig. 14.6: Prescalar (PSC)

19. Like the PSC, the ARR (auto reload register) is also of length 16 bits and used for factoring the clock.
20. What do the following instructions do?

```
if(TIM1->SR & 0x0001)//check if ARR count complete
{
        TIM1->SR &= ~0x0001;//clear status register SR
        GPIOA->ODR ^= (1 << 1);//blink LED through PA1
}
```

Solution: Once the TIM1 counter counts from 0 to TIM1− >ARR=999, it resets and starts counting again to 999. At the time of reset, the LSB of TIM1− >SR = 1. The **if** command checks this and when this condition is satisfied, TIM1− >SR is cleared and PA1 is toggled. This process keeps repeating. This results in a PA1 output of 1 and 0 with frequency

$$\frac{HSI\_CLK}{(TIM1->PSC+1)(TIM1->ARR+1)} = \frac{8000000}{4000\times 1000} = 2 \text{ Hz} \qquad (14.4)$$

21. How would you use the repetition counter (RCR) to do the above?
Solution: The following instructions

```
TIM1->SMCR = 0; //Internal clock, 8MHz
TIM1->PSC = 999; //Prescalar, dividing clock by 1000
TIM1->CR1  = 0x0001; //enable Timer1
TIM1->ARR  = 999; //Load Count
TIM1->RCR  = 3; //Load Repetition Count
```

lead to

$$\frac{HSI\_CLK}{(TIM1->PSC+1)(TIM1->ARR+1)(TIM1->RCR+1)} = \frac{8000000}{4000\times 1000} = 2 \text{ Hz} \qquad (14.5)$$

TIM1− >RCR keeps track of the number of times the counter has overflowed. Fig. 14.7 shows the repetition counter register.

22. What is the maximum value of TIM1− >RCR?
23. What is the function of TIM1− >SR?
Solution: The status register (SR) is shown in Fig. 14.8. The UIF flag is 1 if the RCR overflows.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Fig. 14.7: Repition Counter (RCR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0	Res.	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Fig. 14.8: Staus Register (SR)

14.11 TIMER-2

24. Blink an LED with TIM2.
Solution:

timers/codes/timer2_blink.c

25. In the above code,

```
RCC->APB1ENR |= RCC_APB1ENR_TIM2EN;
```

is used for enabling TIM2. Note that for TIM1, APB2 was used instead of APB1 in Problem 12. Explain.
Solution: Advanced Peripheral Bus 1 (APB1) and Advanced Peripheral Bus 2 (APB2) are connected to the Direct Memory Access (DMA) module, SRAM, peripherals like GPIOs, ADC, timers, etc. and Cortex core. APB1 has a maximum operating frequency of 36MHz while the maximum operating frequency of APB2 is 72MHz. That is why GPIOs are connected to APB2 bus instead of APB1 or any other bus and STM32 GPIOs can achieve 50MHz switching speed. APB1 bus mostly serves communication and timer modules of the STM32.
26. Mention any major difference between TIM1 and TIM2.
Solution: TIM1 is an advanced timer while TIM2 is a general purpose timer. One major difference between the two is that TIM2 does not have RCR.

14.12 Master-Slave Configuration

27. Execute the following program.

timers/codes/master1_slave2_blink.c

28. List the instructions for setting up TIM1 as master and TIM2 as slave. TIM1 should be a prescalar for TIM2.
Solution: The MMS bits can be seen in the CR2 register is shown in Fig. 14.9. The TS and SMS bits are visible in the SMCR register in Fig. 14.4.

```
TIM1->CR2 = 0x0020;//MMS = 010
TIM2->SMCR = 0x0007;//TS = 000, SMS = 111
```

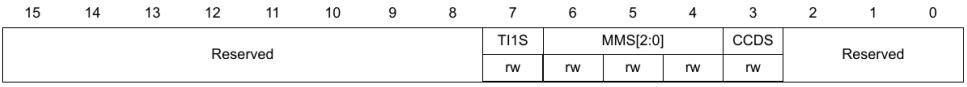


Fig. 14.9: Control Register 2 (CR2)

29. Execute the following program.

timers/codes/decade_counter.c

30. If TIM2− >ARR = 9, TIM2− >CNT = ?
Solution: TIM2− >CNT = 0, 1, . . . , 9, 0,. . . , 9,. . . The counting rate depends on the PCR, ARR, RCR registers of TIM1 and the PCR and ARR registers of TIM2.

14.13 LCD

We show how to interface the 16×2 HD44780-controlled LCD using STM32F103C8T6.

1. Make connections as shown in Table 14.14.

STM32	**LCD Pins**	**LCD Pin Label**	**LCD Pin Description**
GND	1	GND	
3.3V	2	Vcc	
GND	3	Vee	Contrast
A0	4	RS	Register Select
GND	5	R/W	Read/Write
A1	6	EN	Enable
A2	11	DB4	Serial Connection
A3	12	DB5	Serial Connection
A4	13	DB6	Serial Connection
A5	14	DB7	Serial Connection
3.3V	15	LED+	Backlight
GND	16	LED-	Backlight

TABLE 14.14: Pin Connections

2. Execute the following program

lcd/codes/lcd_example.c

The following problems explain how to display the string **0** on the screen using the above code.

3. Write the ASCII code for the 0 character.
Solution: The code for 0 is **48 =0b00110000 = 0x30**.

4. How is 0 written by the STM32 to the LCD controller.
Solution: For the number 0, the upper nibble 0011 is first written followed by the lower nibble 0000. This is done by

```
void SendByte (byte data)
{

 SendNibble(data >> 4); // send upper 4 bits 0011
 SendNibble(data); // send lower 4 bits 0000
}
```

5. How is the nibble 0011 written to the LCD.
   Solution: This is done by the following function where data = 0011.

```
void SendNibble(byte data)
{
GPIOA->BRR = ~(data << 2) & 0b00111100;
GPIOA->BSRR = (data << 2) & 0b00111100;

 PulseEnableLine(); // clock 4 bits into controller
}
```

   The expression

$$GPIOA->BSRR = (data << 2)\&0b00111100 = 0b00001100. \tag{14.6}$$

   This ensures that 11 is written to the pins A2-A3. Note that << indicates 2 left shifts. Similarly,

$$GPIOA->BRR = (data << 2)\&0b00111100 \tag{14.7}$$

6. ensures that 00 is written to the pins A4-A5. PulseEnableLine() provides a clock pulse used to write the nibble 0011 to the LCD.
7. Which pins of the STM32 are used for what purpose?
   Solution: The A2-A5 pins of the STM32 are used for pushing the upper/lower data nibble to the DB4-DB7 pins of the LCD using the BRR and BSRR registers. The A0-A1 pins are used for Register Select and EN for the LCD.
8. What is Register Select?
   Solution: Register Select = 0 implies that LCD configuration commands are being written. For example, cursor on/off, clearing display, number of lines, etc... Register Select = 1 implies that characters are being writen to the LCD.
9. Develop an arithmetic calculator using the STM32 along with the LCD.

14.14 ADC

We show how to use the Internal Temperature Sensor to measure the CPU temperature through the ADC.

1. Make connections as shown in Table 14.14.
2. Execute the following program

```
stm32/adc/codes/internal_temp.c
```

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP 15_0	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Fig. 14.10: SMPR1

3. What do you observe?
   Solution: You should observe a number between 1750-1760. This is the output of the internal temperature sensor, captured in $ADC1->DR$.
4. Find an expression for V_{SENSE}
   Solution:
   $$V_{SENSE} = 3.3 \times \frac{ADC1->DR}{4095} \tag{14.8}$$
5. Obtain the formula for finding the temperature of the STM32 and list the values of the various parameters.
   Solution: The desired formula is
   $$T = (V_{25} - V_{SENSE})/AvgSlope + 25 \tag{14.9}$$
   where the typical values of the above parameters are

V_{25}	AvgSlope
1.43	4.3

6. What is the default ADC frequency?
   Solution: The ADC operates at 14 MHz by default and is indpendent of the processor frequency (8 MHz in this case). It can, however be synchronized with the processor clock for some real time applications.
7. Explain the significance of the following instruction

```
ADC1->SMPR1 |= ADC_SMPR1_SMP16;
```

   Solution: Through this command, $ADC1->SMPR1$ = 0x001C0000 where the SMPR1 register is shown in Fig. 14.10. Note that this makes SMP16 = 111 which means that channel 16 sample time = 239.5 cycles. Channel 16 is reserved for the internal temperature sensor and is connected to ADC1.
8. What is the sampling time?
   Solution: Since the sample time is 239.5 cycles and the ADC frequency is 14 MHz,
   $$T_s = 239.5 \times \frac{1}{14}\mu s = 17.1\mu s \tag{14.10}$$
9. Explain the following instruction.

```
ADC1->SQR3 |= ADC_SQR3_SQ1_4;
```

   Solution: ADC_SQR3_SQ1_4 = 0x00000010. This implies that SQ1=0b10000 in the ADC regular sequence register 3 ($ADC1->SQR3$)shown in Fig. 14.11. Since

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ6[4:0]					SQ5[4:0]					SQ4[4:1]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ4_0		SQ3[4:0]					SQ2[4:0]					SQ1[4:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Fig. 14.11: SQR3

SQ1=16, this means that the ADC input in channel 16 will be the first in the queue for conversion. The ADC is capable of converting analog 16 inputs one after the other. The inputs are called *channels* and the sequence number corresponding to the channel is decided according to the 5 bit entry in SQ.

14.15 Measuring an Unkown Resistance

10. Configure SQR3 so that the 9th channel for ADC1 is 2nd in sequence.
Solution: This implies that SQ2=1001. Thus,

$$ADC1 -> SQR3 = 0x000000120 \tag{14.11}$$

11. List the various pin numbers corresponding to the different channels of the ADC.
Solution: See Fig. 14.15

TABLE 14.15: ADC Analog Input Pins

Channel	0	1	2	3	4	5	6	7	8	9
Pin	PA0	PA1	PA2	PA3	PA4	PA5	PA6	PA7	PB0	PB1

12. Use the 9th channel of ADC1 in SQ2 to measure 3.3V.
Solution: Connect PB1 (Fig. 14.15) to 3.3 V of the STM 32 and execute the following code.

stm32/adc/codes/3_3V.c

13. Measure an unkown resistance using the STM32 and display the result on the LCD.
14. Display the output of the internal temperature sensor as well the unknown resistance on the LCD.

Through this manual, we learn how to measure an unknown resistance through arduino and display it on an LCD.

1. Let R_1 be the known resistor and R_2 be the unknown resistor. Connect R_1 and R_2 in series such that R_1 is connected to GND and R_2 is connected to V_{cc}. Refer to Fig. 15.1

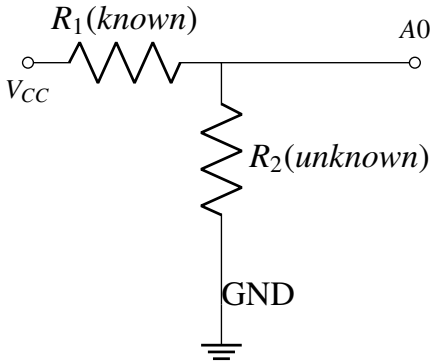


Fig. 15.1: Voltage Divider

2. Connect the junction between the two resistors to the A0 pin on the Arduino.
3. Connect the arduino to the computer so that it is powered.
4. Open the Arduino IDE and type the following code. Open the *serial monitor* to view the output.

ide/lcd/codes/resistance/resistance.ino

5. Plug the LCD in Fig. 8.1 to the breadboard.
6. Connect the 220Ω resistance from V_{cc} to pin 15 (Led+) of the LCD.
7. Connect the Arduino pins to LCD pins as per Table 8.3.
8. Include the instructions for the LCD in the code for measuring the resistance. **Solution:**

ide/lcd/codes/lcd/resistance_lcd.ino

9. We create a variable called analogPin and assign it to 0. This is because the voltage value we are going to read is connected to analogPin A0.
10. The 10-bit ADC can differentiate 1024 discrete voltage levels, 5 volt is applied to 2 resistors and the voltage sample is taken in between the resistors. The value which we get from analogPin can be between 0 and 1023. 0 would represent 0 volts falls across the unknown resistor. A value of 1023 would mean that practically all 5 volts falls across the unknown resistor.
11. V_{out} represents the divided voltage that falls across the unknown resistor.

12. The Ohm meter in this manual works on the principle of the voltage divider shown in Fig. 15.1.

$$V_{out} = \frac{R_1}{R_1 + R_2} V_{in} \tag{15.1}$$

$$\Rightarrow R_2 = R_1 \left(\frac{V_{in}}{V_{out}} - 1 \right) \tag{15.2}$$

In the above, $V_{in} = 5\text{V}$, $R_1 = 220\Omega$.