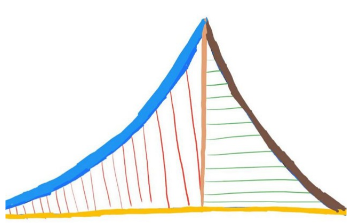
DIGITAL DESIGN

Through Embedded Programming

G. V. V. Sharma



Contents

Introduction

This book introduces digital design through using the arduino framework.

Chapter 1

Installation

1.1. Termux

1. On your android device, install fdroid apk from

```
https://www.f–droid.org/
```

2. Install Termux from apkpure

3. Install basic packages on termux

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


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


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

```
——————————————————————————————
```

4. Install Ubuntu on termux

```
proot—distro install ubuntu
proot—distro login ubuntu
```

1.2. Platformio

1. Install Packages

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


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


#——————————————————— Installing platformio on termuxubuntu
```

```
——————————————————————————
python3 —m venv gvenv
source gvenv/bin/activate
pip3 install platformio
#——————————————————— End installing python3 on termuxubuntu
——————————————————————————
```

2. Execute the following on ubuntu

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

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

```
pio run —t nobuild —t upload
```

4. The LED beside pin 13 will start blinking

1.3. Arduino Droid

1. Install ArduinoDroid from apkpure
2. Open ArduinoDroid and grant all permissions
3. Connect the Arduino to your phone via USB-OTG
4. For flashing the bin files, in ArduinoDroid,

```
Actions—>Upload—>Upload Precompiled
```

then go to your working directory and select

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

for uploading hex file to the Arduino Uno

5. The LED beside pin 13 will start blinking

Chapter 2

Seven Segment Display

We show how to control a seven segment display.

2.1. Components

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

Table 2.1:

2.1.1. Breadboard

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

2.1.2. Seven Segment Display

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

2.1.3. Arduino

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

2.2. Display Control through Hardware

2.2.1. Powering the Display

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

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

Table 2.2: Supply for Bread board

2. Make the connections in Table 2.3.

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

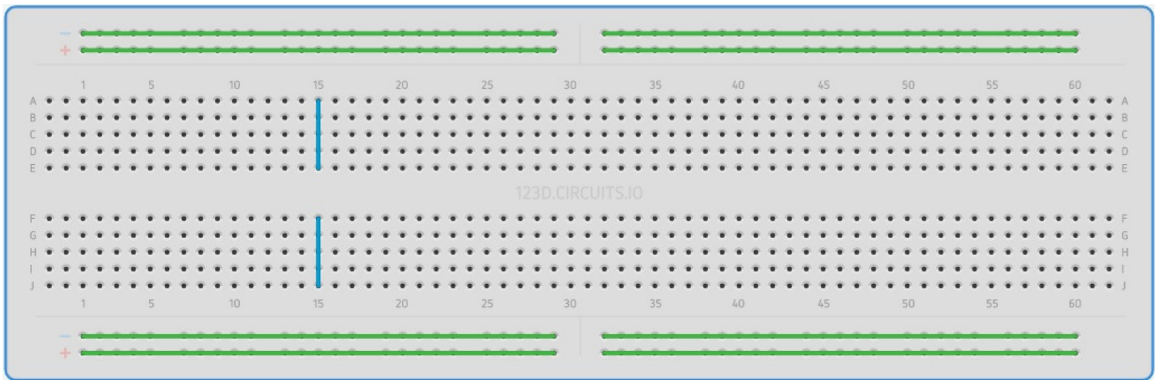


Figure 2.1: Bread board connnections

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

Table 2.3: Connecting Seven segment display on Bread board

2.2.2. Controlling the Display

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

1. Generate the number 1 on the display by connecting only the pins *b* and *c* to GND (=0). This corresponds to the first row of 2.4. 1 means not connecting to GND.
2. Repeat the above exercise to generate the number 2 on the display.
3. Draw the numbers 0-9 as in Fig. 2.3 and complete Table 2.4

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

Table 2.4:

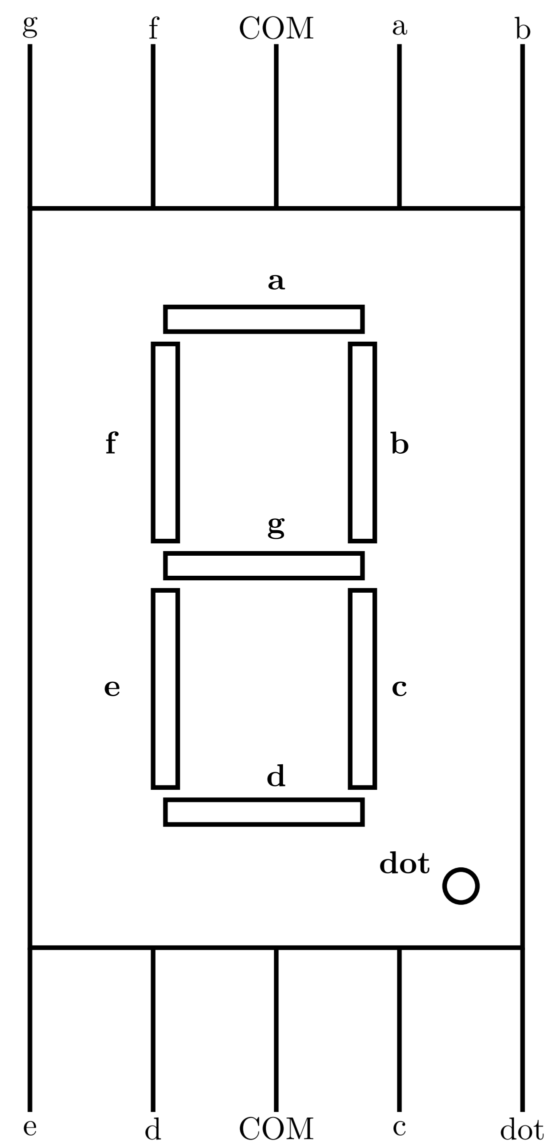


Figure 2.2: Seven Segment pins

2.3. Display Control through Software

1. Make connections according to Table 2.5

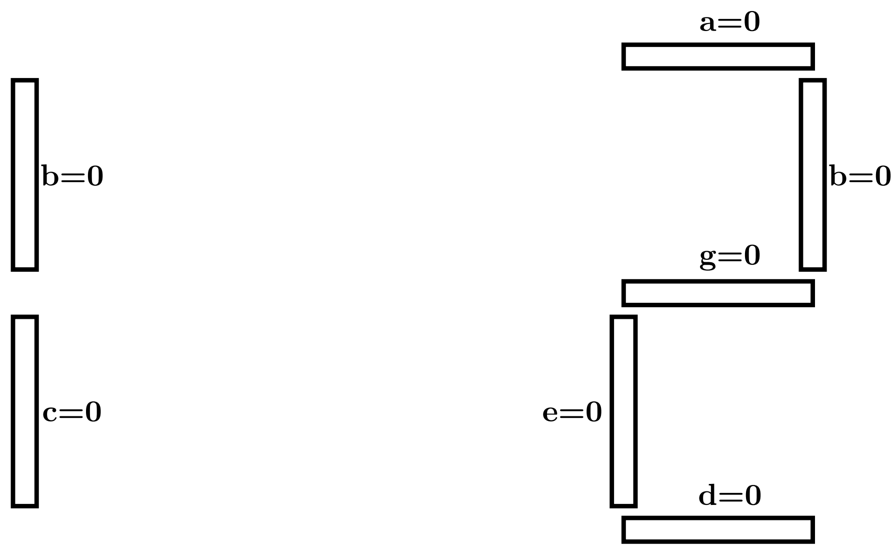


Figure 2.3: Seven Segment connections

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

Table 2.5:

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

ide/sevenseg/codes/sevenseg/sevenseg.ino

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

Chapter 3

7447

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

3.1. Components

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

Table 3.1:

3.2. Hardware

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

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

Table 3.2:

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

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

Table 3.3:

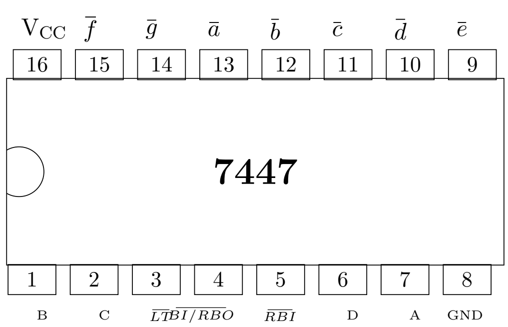


Figure 3.1:

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

3.3. Software

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

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

7447	D	C	B	A
Arduino	5	4	3	2

Table 3.4:

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

Table 3.5: Truth table for incrementing Decoder.

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

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

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

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

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

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

ide/7447/codes/ip_inc_dec/ip_inc_dec.cpp

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

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

Table 3.6:

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

5. Now write a program where

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

   (b) incremented by 1 using Table 3.5and

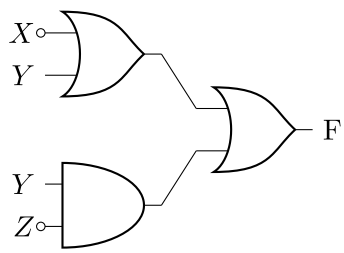
   (c) the incremented value is displayed on the seven segment display.

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

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

3.4. Problems

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



2. Verify the Boolean Expression (CBSE 2013)

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

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

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

4. Verify the following (CBSE 2015)

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

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

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

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

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

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

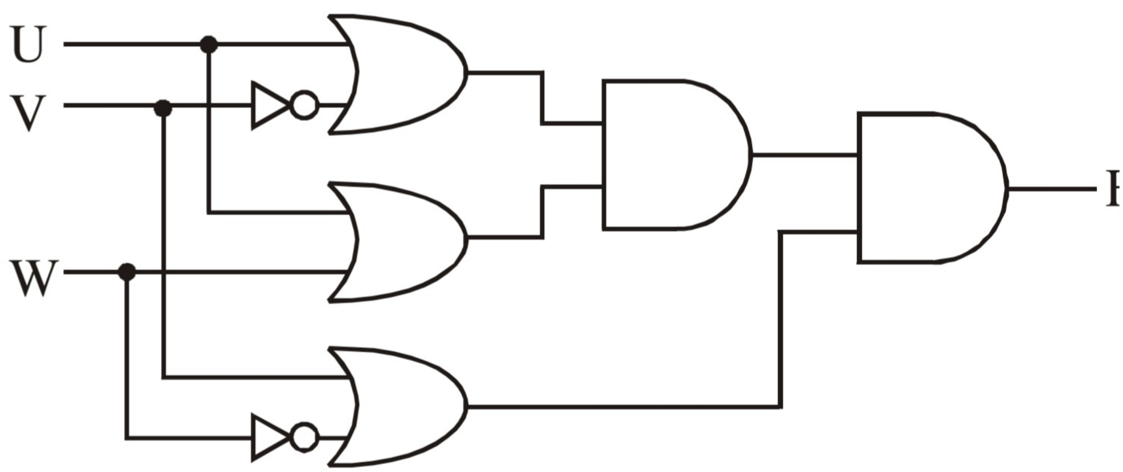


Figure 3.2:

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

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

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

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

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

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

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

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

Table 3.7:

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

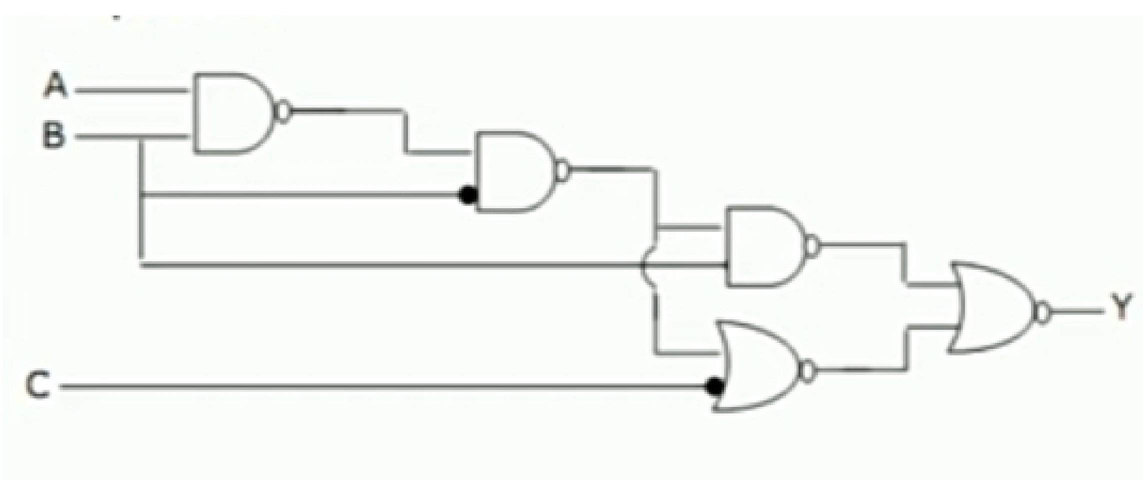
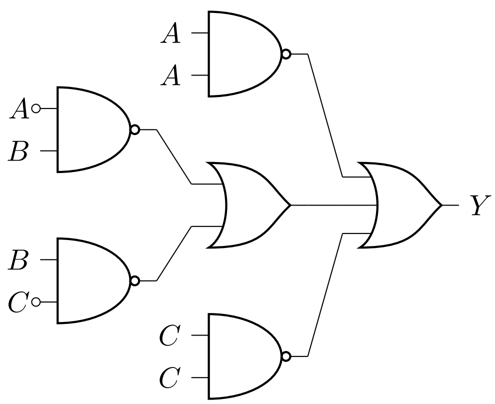


Figure 3.3:

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



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

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

Table 3.8:

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

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

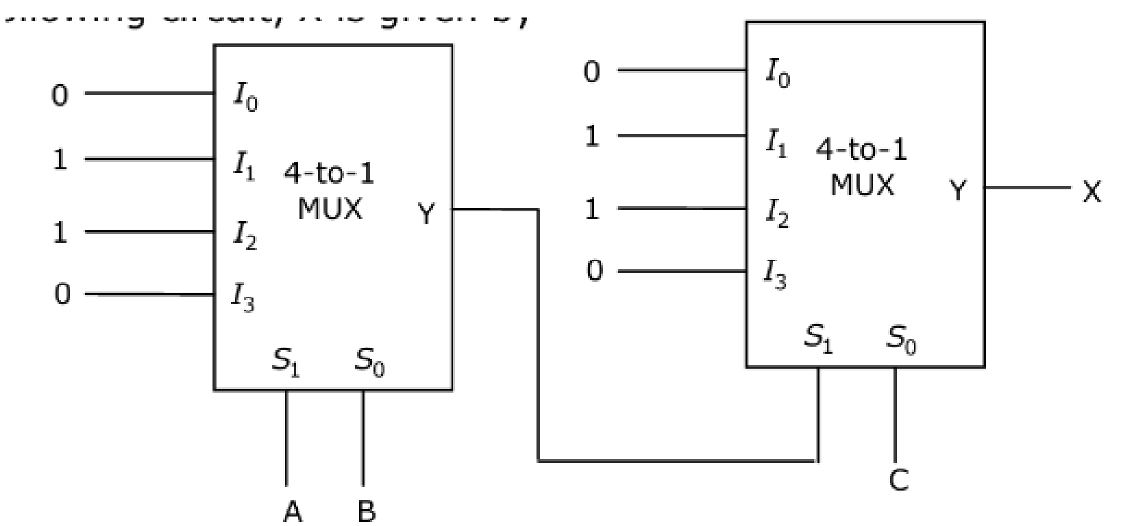


Figure 3.4:

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

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

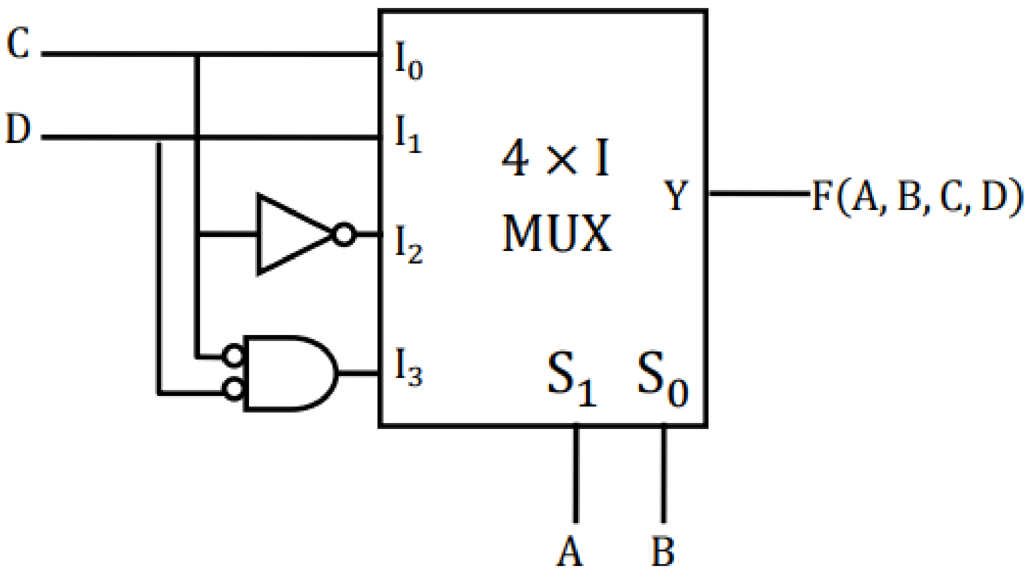


Figure 3.5:

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

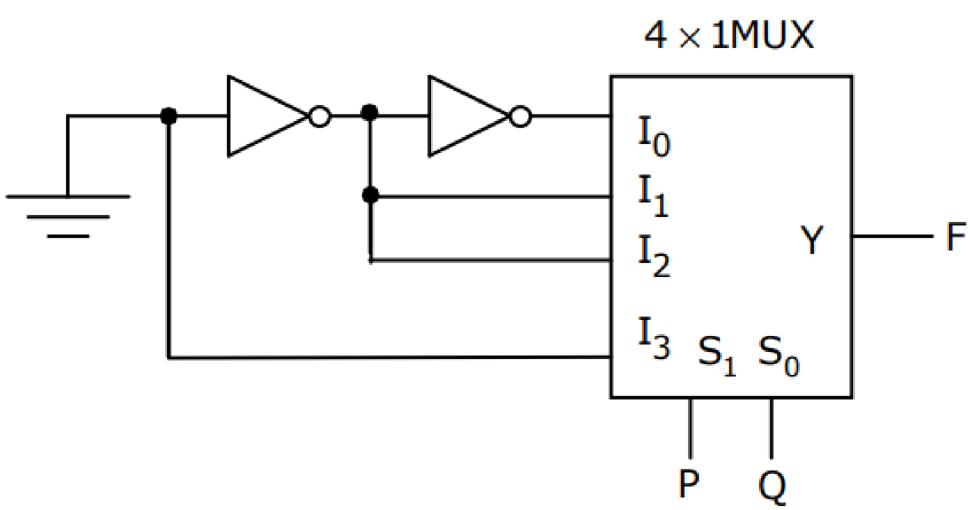


Figure 3.6:

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

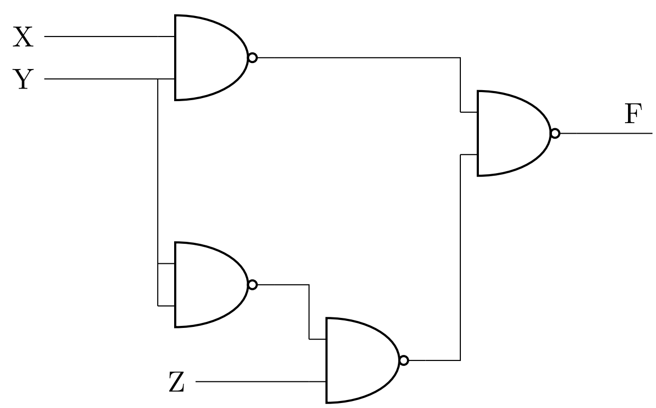


Figure 3.7:

21. Find the logic function implemented by the circuit given below in Fig. 3.8 (GATE EC 2017)

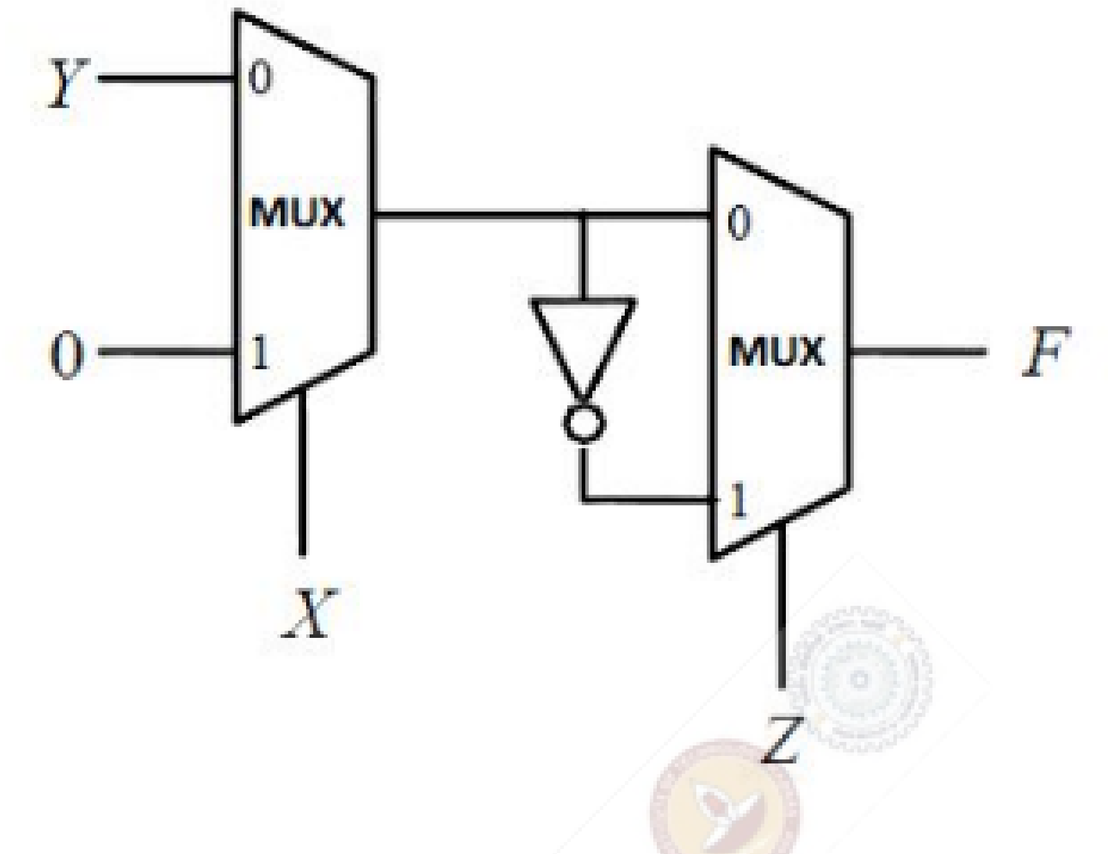


Figure 3.8:

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

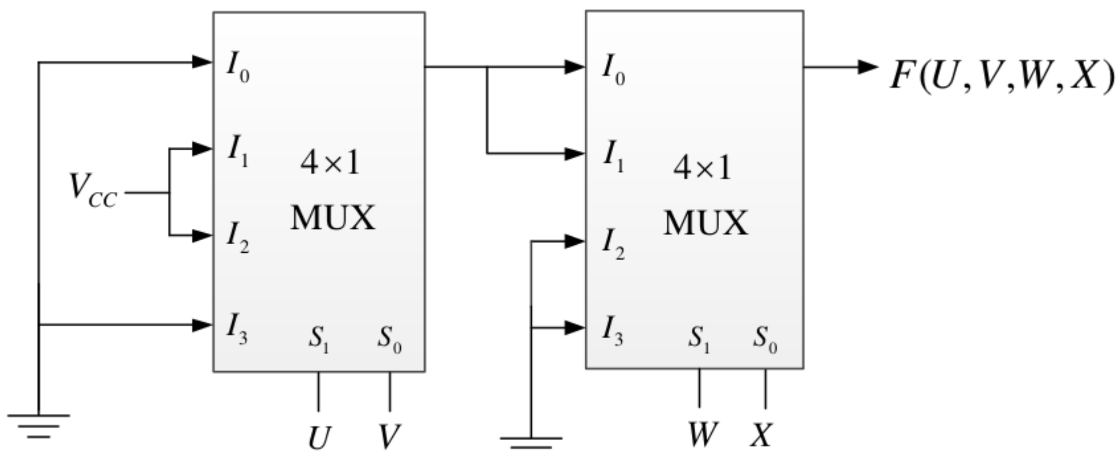


Figure 3.9:

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

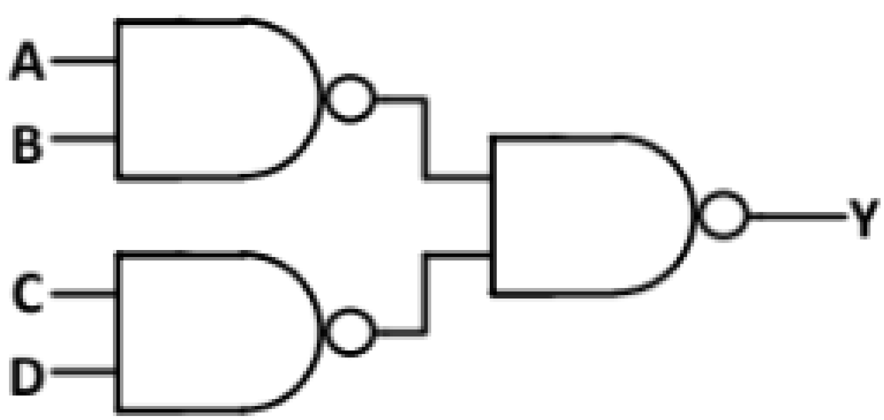


Figure 3.10:

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

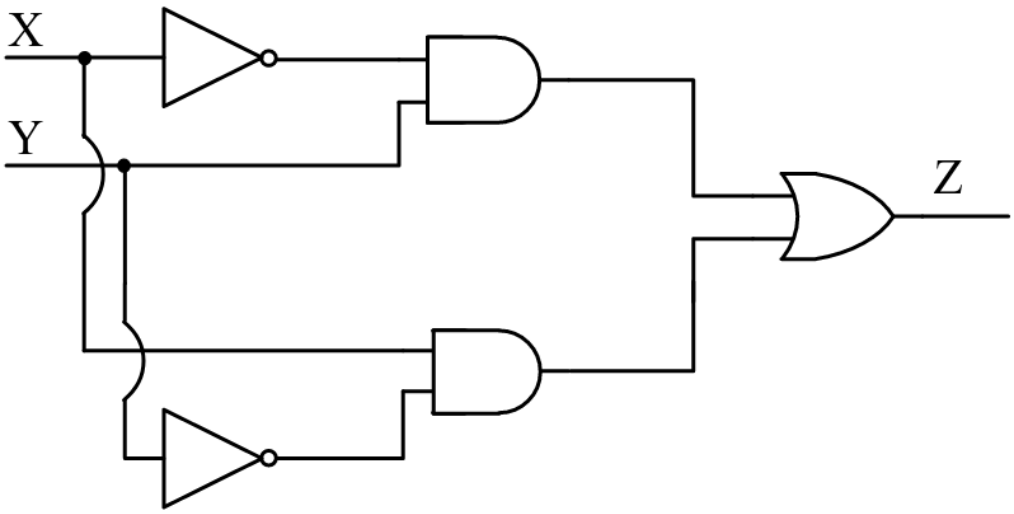


Figure 3.11:

25. Let $\oplus$ and $\odot$ denote the Exclusive OR and Exclusive NOR operations, respectively.Which one of the following is NOT CORRECT ? 25 (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) $(P \oplus \overline{P}) \oplus Q = (P \odot \overline{P}) \odot \overline{Q}$

26. A Boolean digital circuit is composed using two 4-input multiplexers ($M1andM2$) and one 2-input multiplexer ($M3$) as shown in the figure. $X0$–$X7$ 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.(GATE CS2023,44)

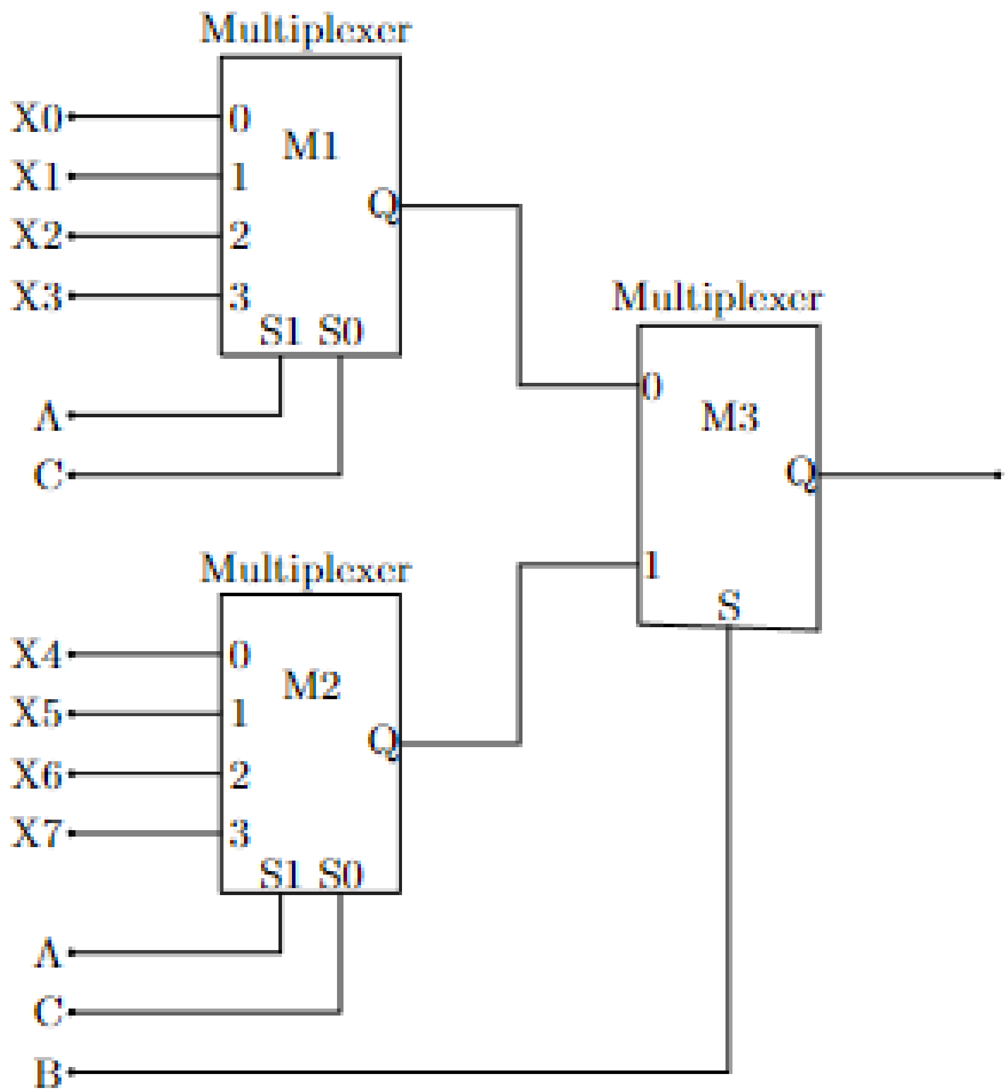


Figure 3.12: Digital Circuit

Which one of the following set of values of $(X0, X1, X2, X3, X4, X5, X6, X7)$ will realise the Boolean function $\overline{A} + \overline{A}.\overline{C} + A.\overline{B}.C$?

(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)

27. For the given digital circuit, $A = B = 1$. Assume that AND, OR, and NOT gates have propagation delays of 10ns,10ns, and 5ns respectively. All lines have zero propagation delay. Given that $C = 1$ when the circuit is turned on, the frequency of steady-state oscillation of the output Y is ______. (GATE IN 2023)

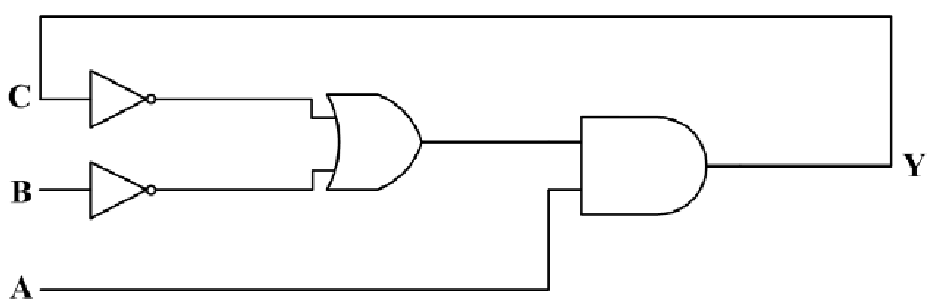


Figure 3.13: Image

(a) 20MHz

(b) 15MHz

(c) 40MHz

(d) 50MHz

28. 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) $x+xz+xy$

29. Which one of the following options is CORRECT for the given circuit ? (GATE PHYSICS 2023)

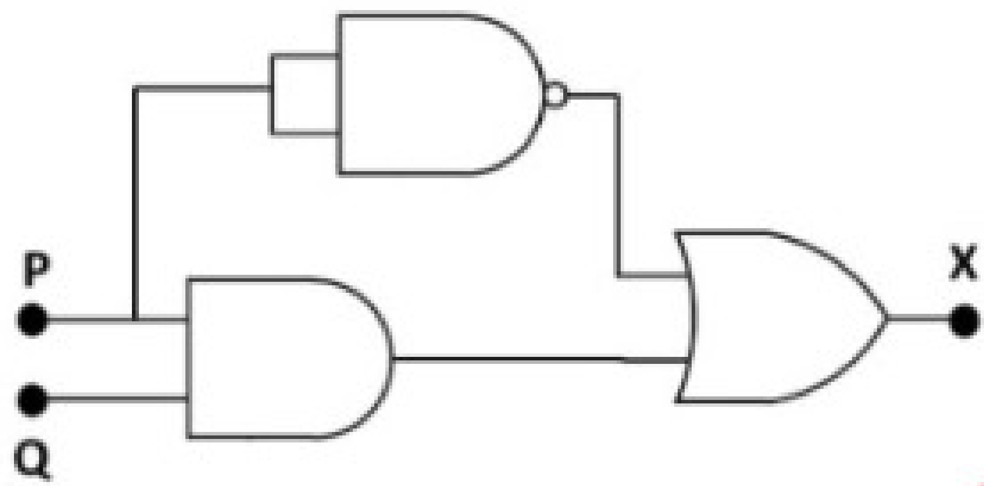


Figure 3.14:

(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

30. In the circuit diagram shown below, the logic gates operate with a supply voltage of $1V$. NAND and XNOR have 200ps and 400ps input-to-output delay, respectively.

At time $t = T.A(t) = 0, B(t) = 1 and Z(t) = 0$. When the inputs are changed to $A(t) = 1, B(t) = 0 at t = 2T$, a 1 V pulse is observed at Z. the pulse width of the $1V$ pulse is ps.

(GATE BM 2022)

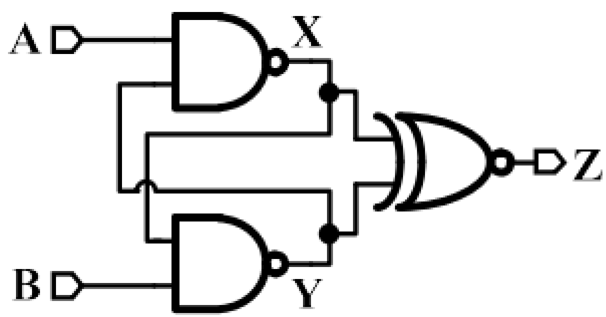


Figure 3.15:

(a) 100

(b) 200

(c) 400

(d) 600

31. 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.

32. 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 $R2 = 1110$

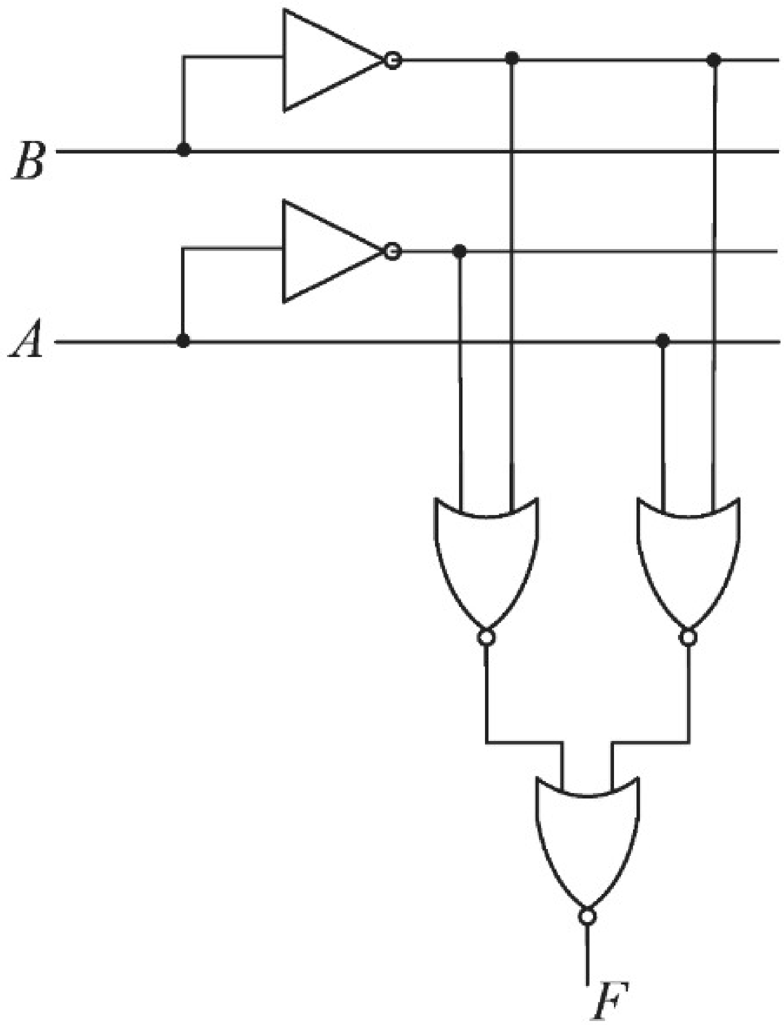
(b) $R1 = 1100$ and $R2 = 1010$

(c) $R1 = 0011$ and $R2 = 0100$

(d) $R1 = 1001$ and $R2 = 1111$

33. The maximmunm clock frequeccy in MHz of a 4-stage ripple counter, utilize flip-flops, with each flip-flop having a propagation delay of 20 ns, is __________.
(*round off to one decimal place*) (GATE EE 20222)

34. The logic block shown has an output F given by __________ (GATE IN 2021)



(a) $A + B$

(b) $A.\bar{B}$

(c) $A + \bar{B}$

(d) $\bar{B}$

35. 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 $\overline{F}$ (complement of F)?

(Gate CS 2021,42)

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

36. The propagation delays of the XOR gate, AND gate and multiplexer (MUX) in the circuit shown in the figure are $4ns$, $2ns$ and $1ns$, respectively. If all the inputs P, Q, R, S and T are applied simultaneously and held constant, the maximum propagation delay of the circuit is (GATE-EC2021,31)

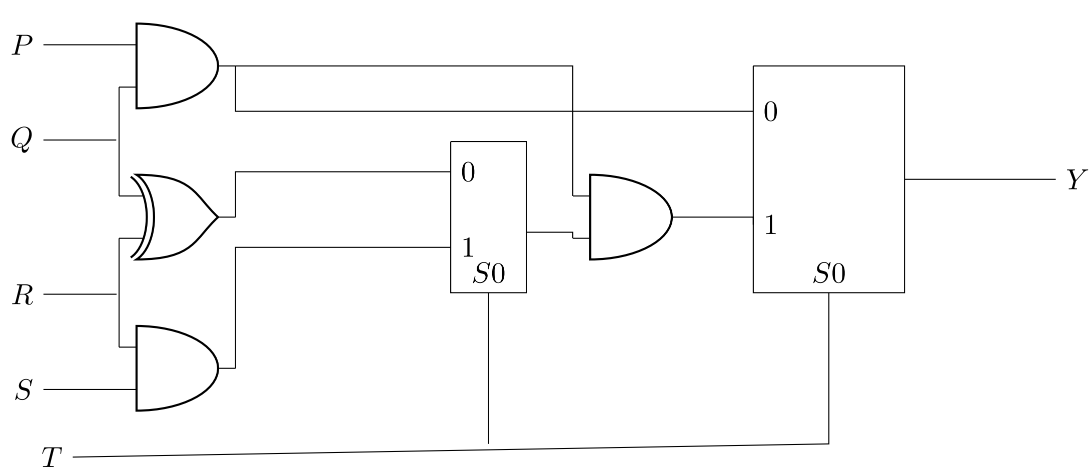


Figure 3.16: circuit daigram

(a) $3ns$

(b) $5ns$

(c) $6ns$

(d) $7ns$

37. The above combination of logic gate represent the operation (GATE PH 2021)

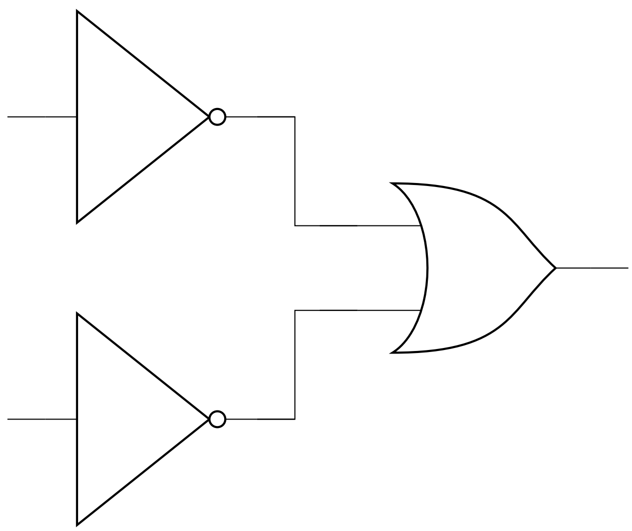


Figure 3.17: combination circuit

(a) OR

(b) NAND

(c) AND

(d) NOR

38. Consider the boolean Function $z\,(a,b,c)$ from below .

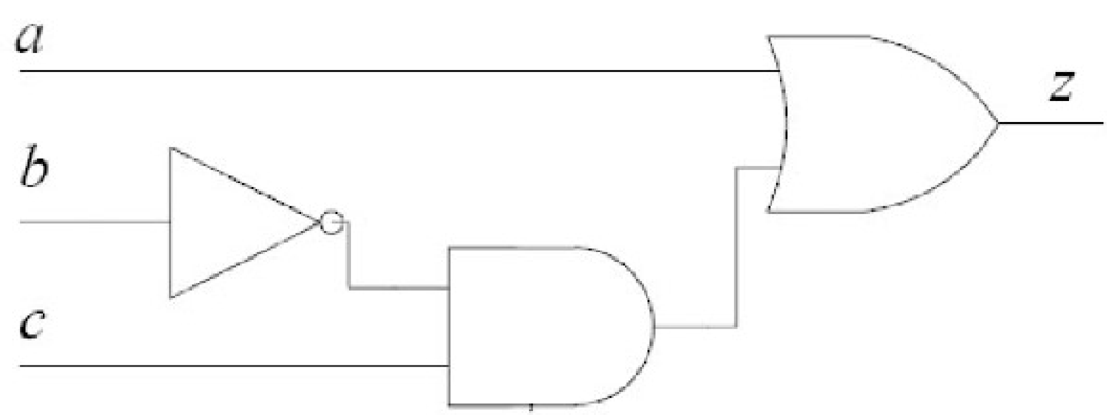


Figure 3.18: circuit diagram

(Gate CS-2020)

Which of the following minterm lists represent the circuit given above?

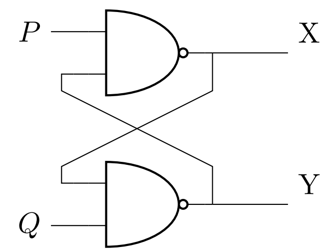
(a) $z = \Sigma\,(0, 1, 3, 7)$

(b) $z = \Sigma\,(1, 4, 5, 6, 7)$

(c) $z = \Sigma\,(2, 4, 5, 6, 7)$

(d) $z = \Sigma\,(2, 3, 5)$

39. In the latch circuit shown, the NAND gates have non-zero but unequal propagation delays. The present input condition is: $P = Q =$ '$0'$. If the input condition is changed simultaneously to $P = Q =$ '$1'$,the outputs X and Y are



(a) $X = `1', Y = `1'$

(b) either $X = `1', Y = `0'$ or $X = `0', Y = `1'$

(c) either $X = `1', Y = `1'$ or $X = `0', Y = `0'$

(d) $X = `0', Y = `0'$

(GATE EC 2017)

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

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

For the following circuit with one AND gate and one XOR gate, the output function f can be expressed as: (GATE-CS2019,30)

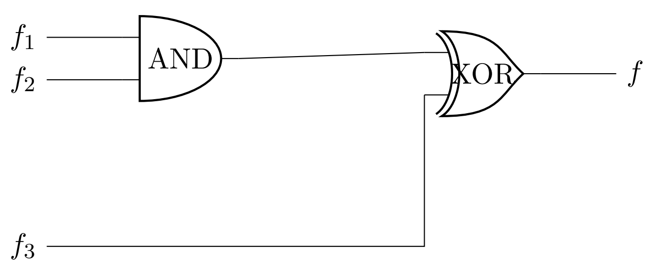


Figure 3.19: Circuit Daigram

(a) $\sum(7, 8, 11)$

(b) $\sum(2, 7, 8, 11, 14)$

(c) $\sum(2, 14)$

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

41. In the circuit shown, what are the values of $F$for $EN = 0$ and $EN = 1$, respectively? (GATE-EC2019,14)

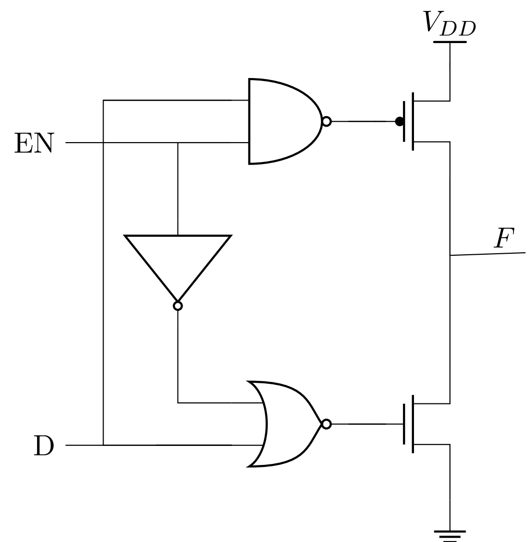


Figure 3.20: Circuit Diagram

(a) 0 and D

(b) $Hi - Z$ and D

(c) 0 and 1

(d) $Hi - Z$ and $\overline{D}$

42. In the circuit shown, A and B are the inputs and F is the output. What is the functionality of the circuit? (GATE-EC2019,15)

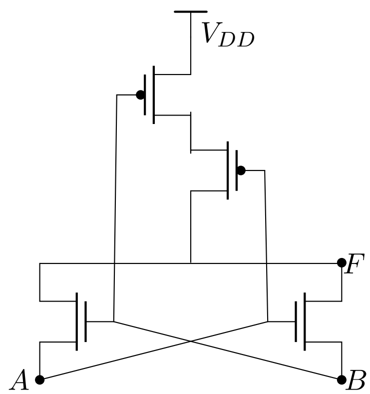


Figure 3.21: Circuit Diagram

(a) Latch

(b) XNOR

(c) SRAM Cell

(d) XOR

43. In the circuit shown below, assume that the comparators are ideal and all components have zero propagation delay. In one period of the input signal $Vin = 6\sin(\omega t)$, the fraction of the time for which the output OUT is in logic HIGH is (GATE-IN2019,34)

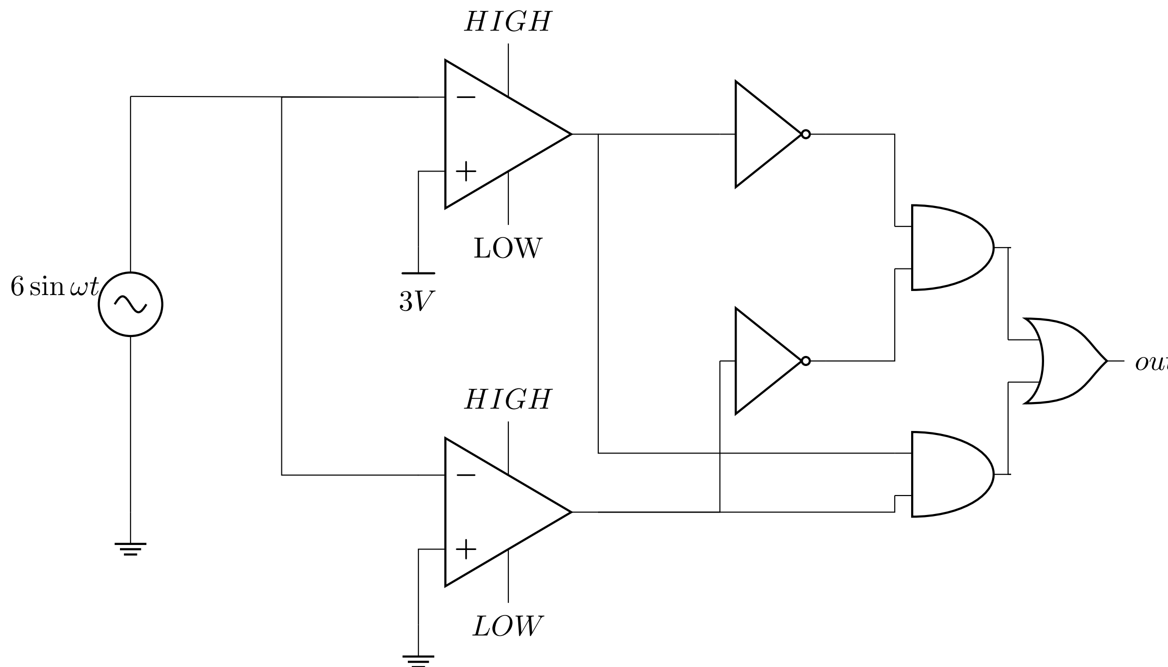


Figure 3.22: Circuit Daigram

(a) $\frac{1}{12}$

(b) $\frac{1}{2}$

(c) $\frac{2}{3}$

(d) $\frac{5}{6}$

44. The figure below 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. Assume that each logic gate has a delay of 2 nanosecond, with no additional time delay due to the interconnecting wires. If the inputs A_i , B_i; are available and stable throughout the carry propagation, the maximum time taken for an input C_i, to produce a steady-state output C_{i+1} is ___ nanosecond. (GATE-IN2019,22)

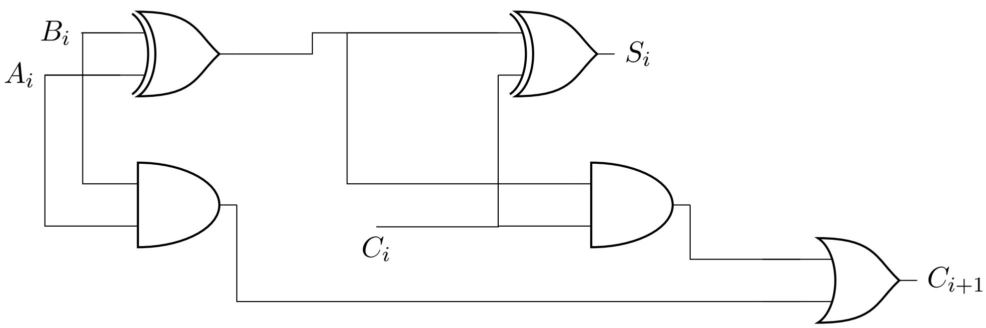


Figure 3.23: Full Adder

45. The Boolean operation performed by the following circuit at the output O is ___________
$(GATE\ IN2020-12)$

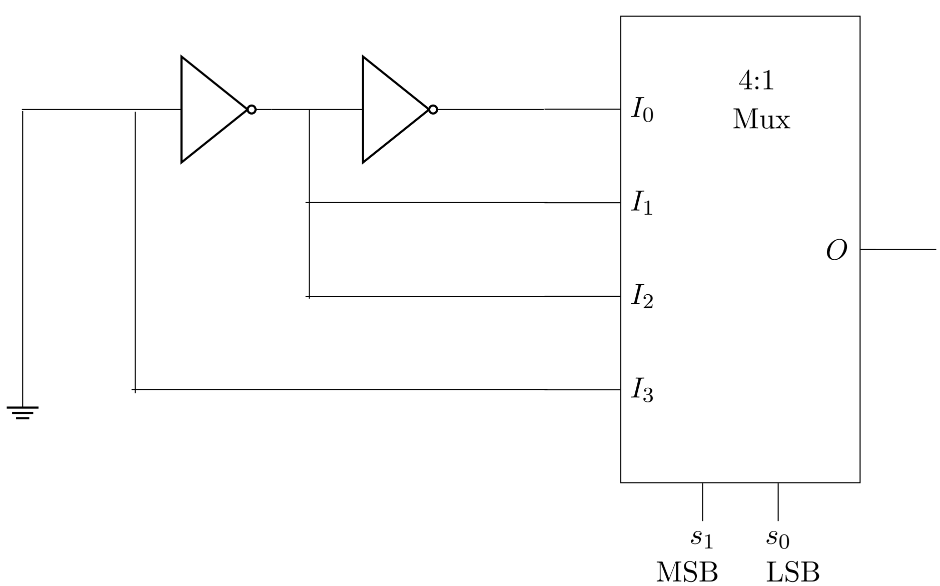


Figure 3.24: Circuit Diagram

(a) $O = S_1 \oplus S_0$

(b) $O = S_1 \bullet \overline{S_0}$

(c) $O = S_1 + S_0$

(d) $O = S_0 \bullet \overline{S_1}$

46. The chip select logic for a certain DRAM chip in a memory system design is shown below. 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? $(GATE\ CS2019-2)$

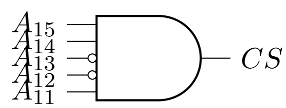


Figure 3.25: Logic Diagram

(a) $C800$ to $CFFF$

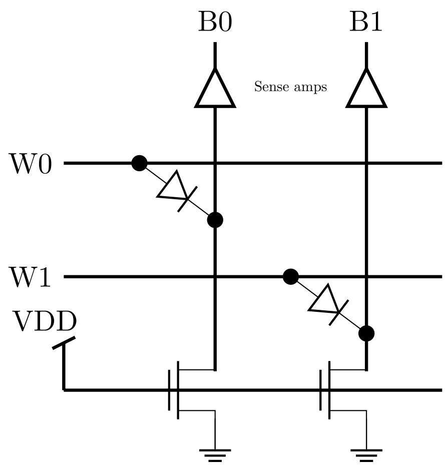
(b) $CA00$ to $CAFF$

(c) $CA00$ to $C8FF$

(d) $DA00$ to $DFFF$

47. 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 signalthe time-counter displays 1000000. with the same input signal,The time countr is changed to 'frequency' mode of operation and the range is set as 'HZ'.The display will be show the number__________. $(GATE\ IN2020-43)$

48. A 2×2 ROM array is built with the help of diodes as shown in the circuit below. 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.



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

Bits stored in the ROM Array

Figure 3.26: 2×2 ROM array

(GATE EC2018,32)

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?

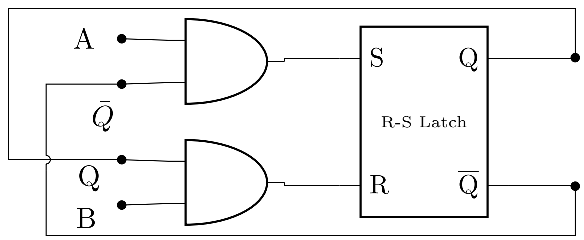
(a) $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

(b) $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

(c) $\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$

(d) $\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$

49. The two inputs A and B are connected to to an R-S latch via two AND gates as shown in the figure. If $A = 1$ and $B = 0$, the output $Q\bar{Q}$ is



(GATE IN2017,43)

(a) 00

(b) 10

(c) 01

(d) 11

50. A and B are logical inputs and X is the logical output shown in the figure. The output X is related to A and B by (GATE IN 2017)

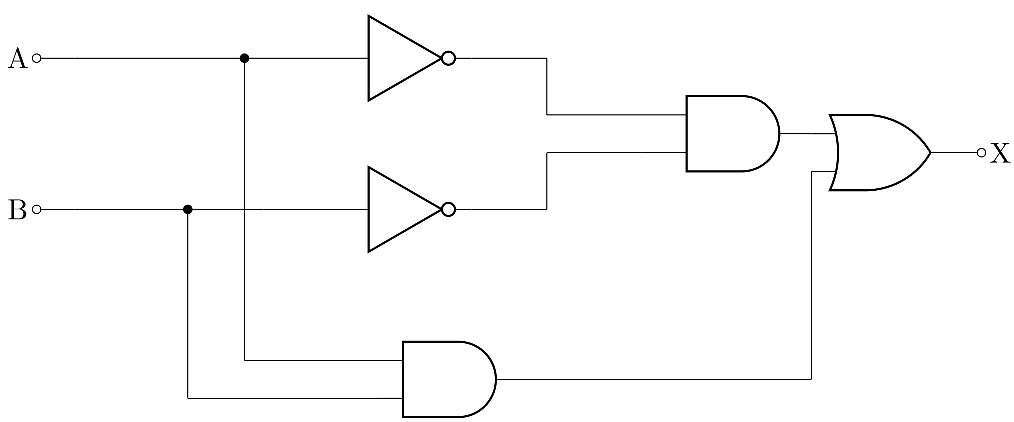


Figure 3.27: Logic Gate Structure

A. $X = \overline{A}B + \overline{B}A$

B. $X = AB + \overline{B}A$

C. $X = AB + (\overline{B})(\overline{A})$

D. $X = (\overline{A})(\overline{B}) + \overline{B}A$

51. The functionality implemented by the circuit below is (GATE 2016 EC)

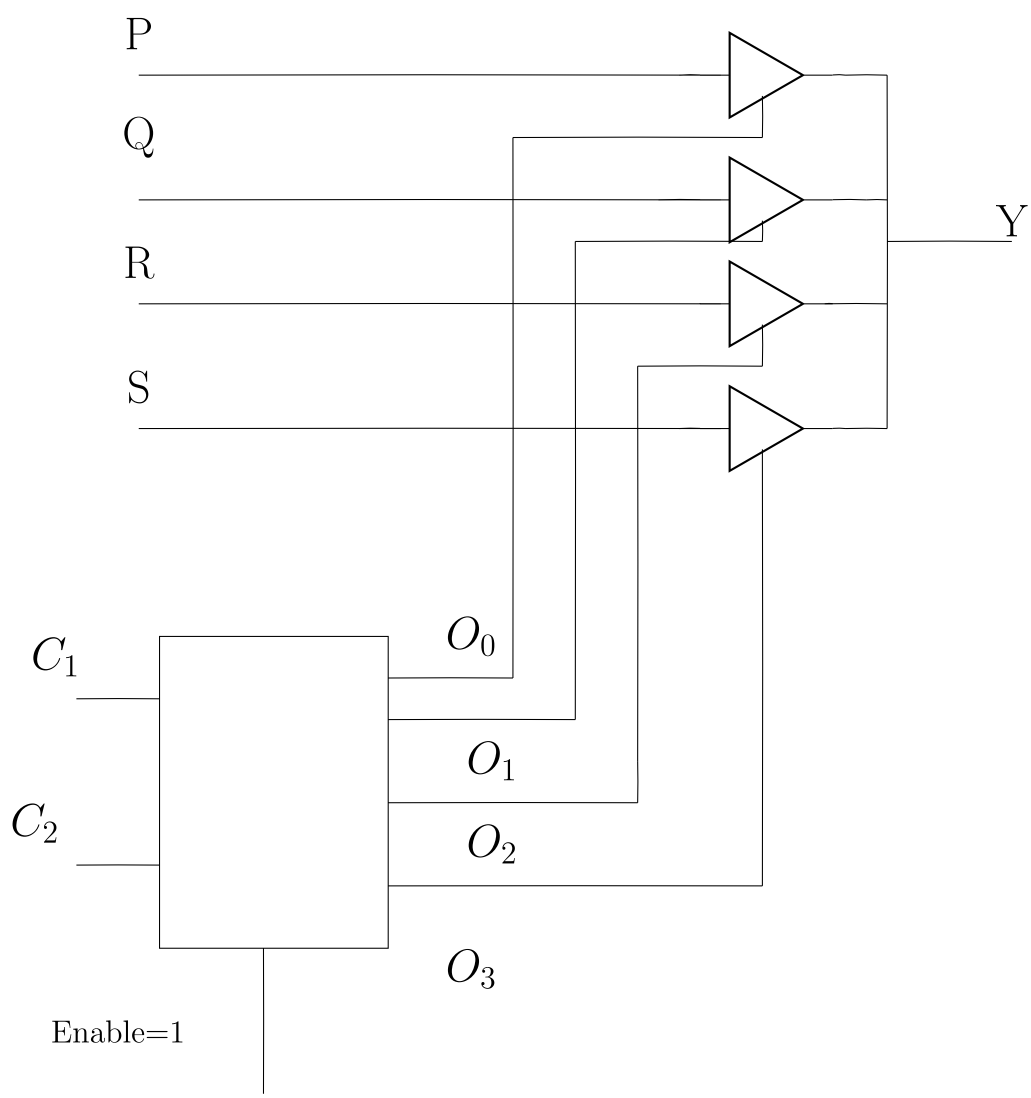


Figure 3.28: Multiplexer

A. 2-to-1 multiplexer

B. 4-to-1 multiplexer

C. 7-to-1 multiplexer

D. 6-to-1 multiplexer

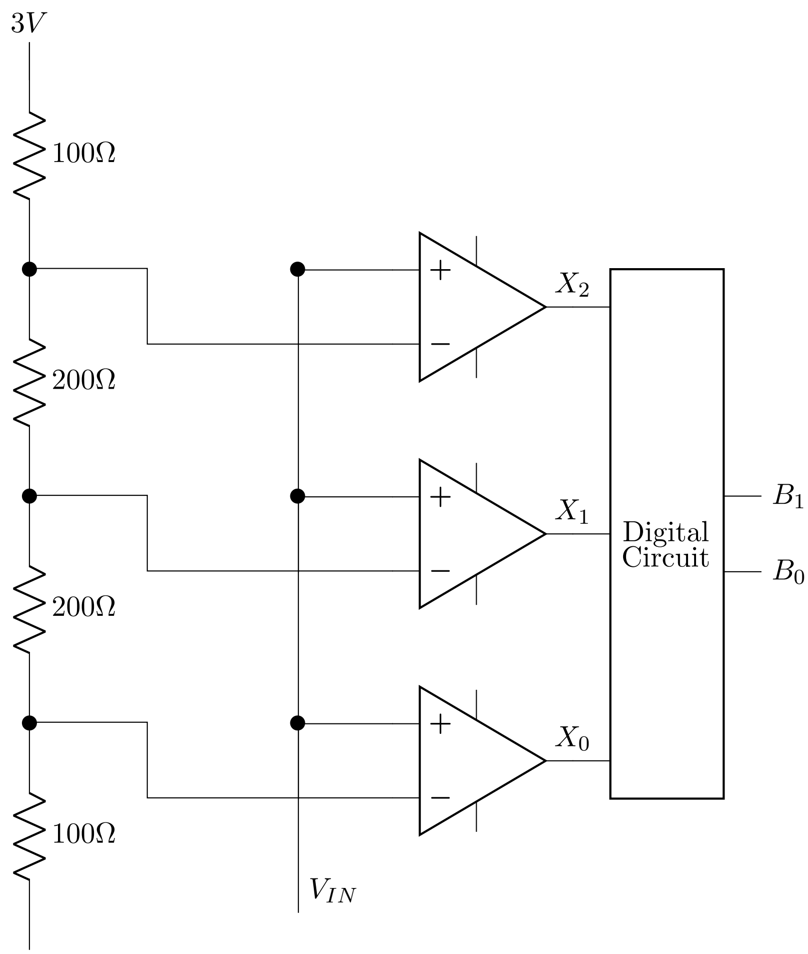
52. A 2-bit flash Analog to Digital Converter (ADC) is given in Fig. 3.29. The input is $0 \le V_{IN} \le 3$ Volts. The expression of the LSB of the output B_0 as a boolean function of X_2, X_1, and X_0 is (GATE EE2016 37)

(a) $X_0\left[\overline{X_2 \oplus X_1}\right]$

(b) $\overline{X_0}\left[\overline{X_2 \oplus X_1}\right]$

(c) $X_0\left[X_2 \oplus X_1\right]$

(d) $\overline{X_0}\left[X_2 \oplus X_1\right]$



$3V$
100Ω
200Ω
200Ω
100Ω
V_{IN}
X_2
X_1
X_0
Digital Circuit
B_1
B_0

Figure 3.29:

Chapter 4

Karnaugh Map

4.1. Introduction

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

4.2. Incrementing Decoder

The incrementing decoder takes the numbers $0, , \ldots, 9$ in binary as inputs and generates the consecutive number as output The corresponding truth table is available in Table 3.5

4.3. Karnaugh Map

Using Boolean logic, output A in Table 3.5 can be expressed in terms of the inputs W, X, Y, Z as

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

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

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

   Using the fact that

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

   derive (4.2) from (4.1) algebraically

2. K-Map for B: From Table 3.5, 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

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

4. K-Map for C: From Table 3.5, using boolean logic,

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

   Show that (4.6) can be reduced to

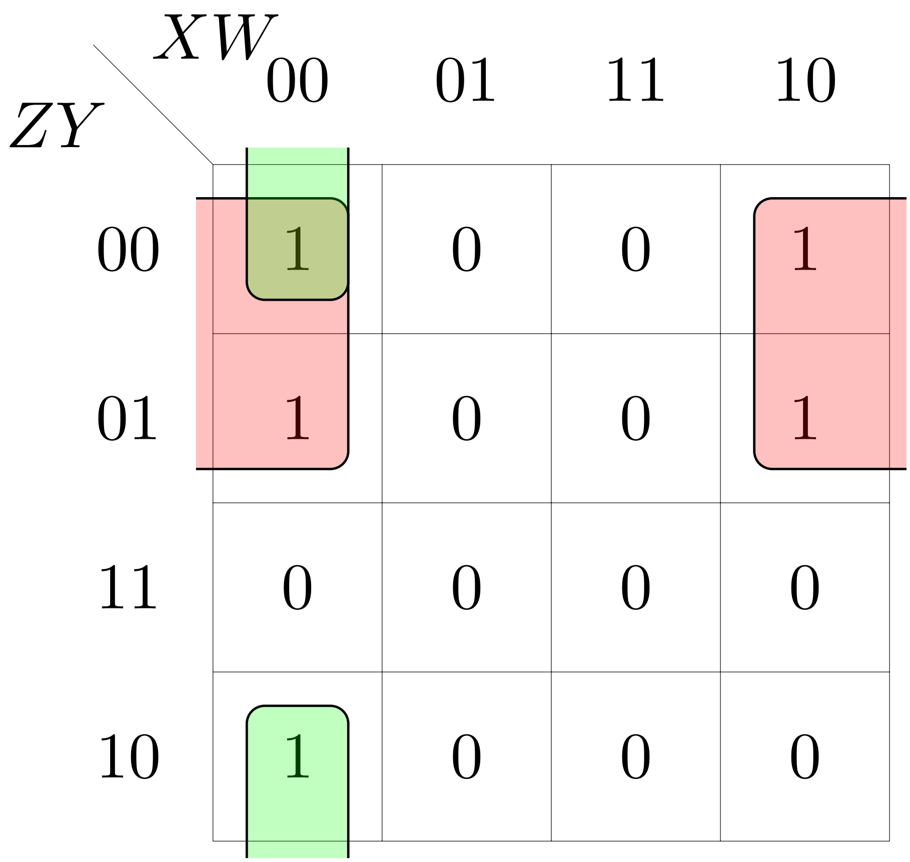


Figure 4.1: K-map for A

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

using Fig 4.3

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

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

Figure 4.2: K-map for B

6. K-Map for D: From Table 3.5, using boolean logic,

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

7. Minimize (4.8) using Fig 4.4

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

Figure 4.3: K-map for C

8. Execute the code in

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

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

9. Display Decoder: Table 4.1 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

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

Figure 4.4: K-map for D

with and without don't care conditions

4.4. Dont Care

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

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.

4.5. Don't Care Conditions

1. Don't Care Conditions: 4 binary digits are used in the incrementing decoder in Table 4.1 However, only the numbers from 0-9 are used as input/output in the decoder and we don't care about the numbers from 0-5 This phenomenon can be addressed by revising the truth table in Table 4.1 to obtain Table 4.2

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

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

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

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

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:

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

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

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

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

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

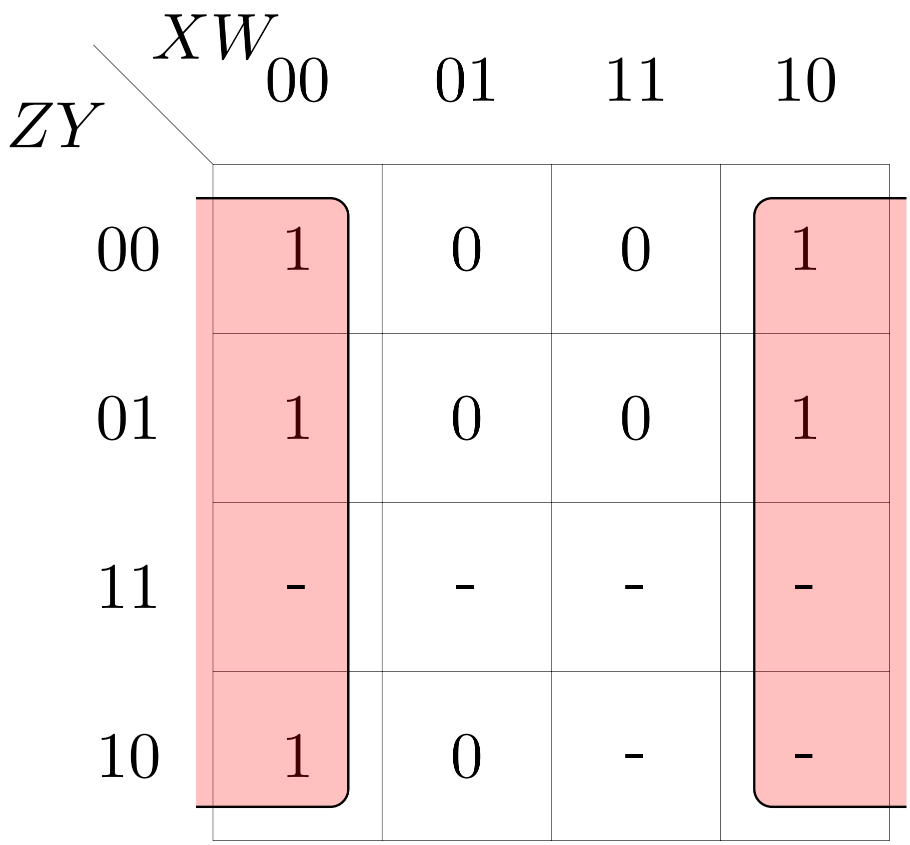


Figure 4.5: K-map for A with don't cares

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

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

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

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

4.6. Problems

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

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

2. Write the POS form for the function G shown in Table 4.3. (CBSE 2013)

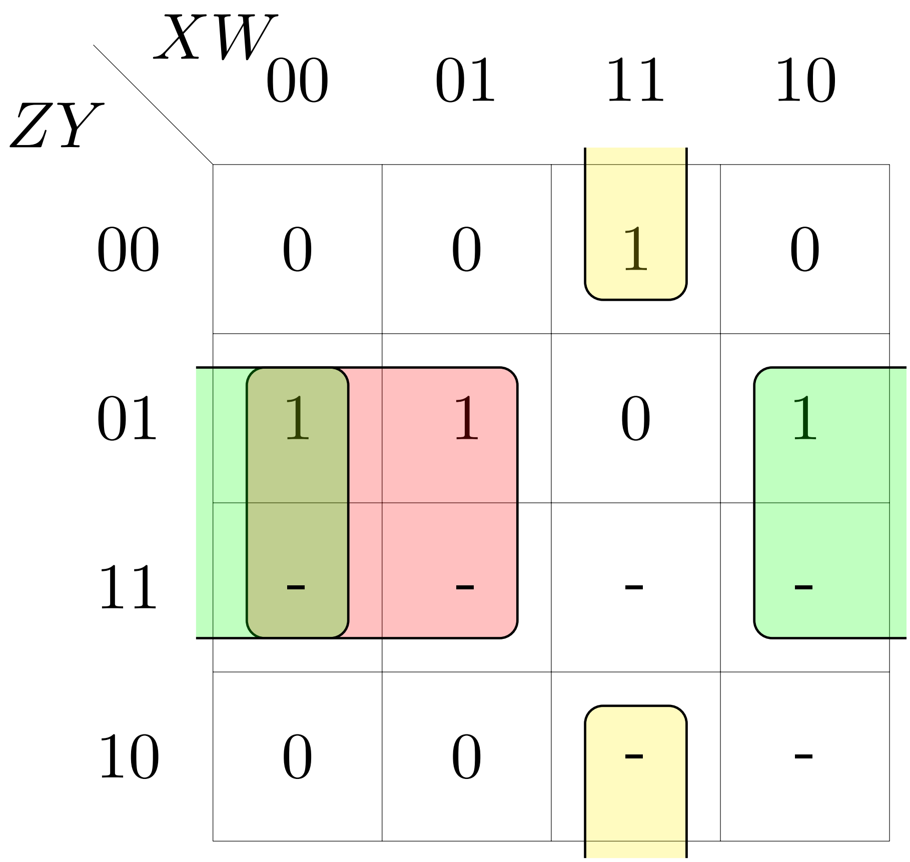


Figure 4.7: K-map for C with don't cares

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

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

4. Derive a Canonical POS expression for a Boolean function F, represented by the

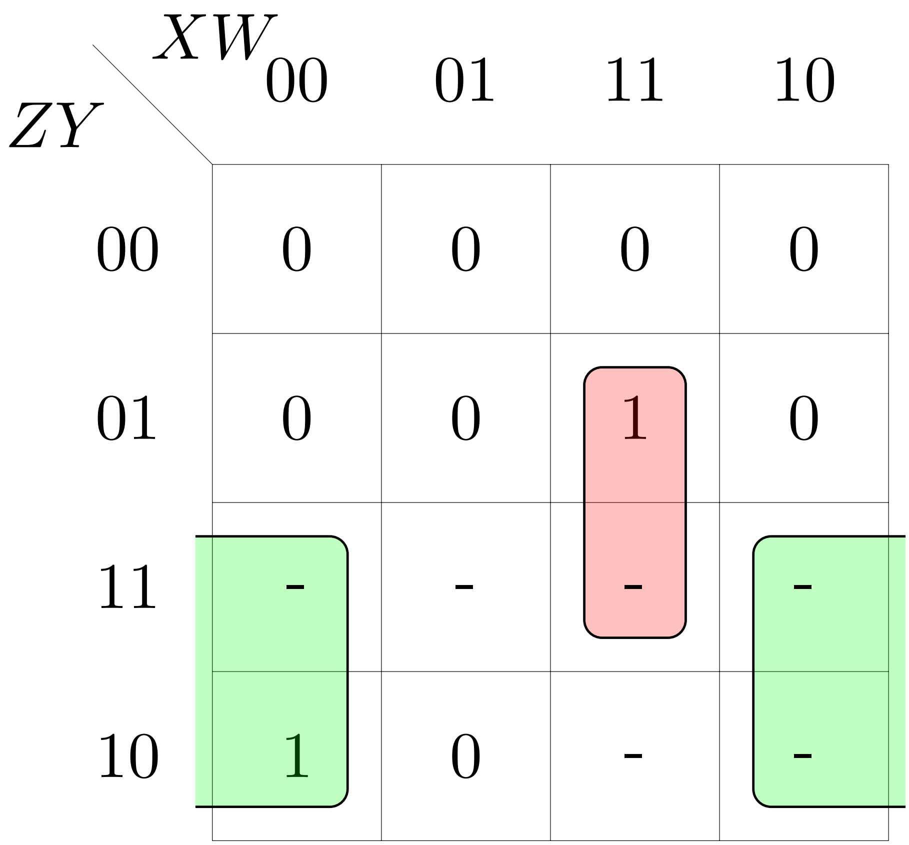


Figure 4.8: K-map for D with don't cares

following truth table (CBSE 2015)

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

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

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:

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

Table 4.4:

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

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

7. Derive a Canonical POS expression for a Boolean function F, represented in Table 4.5 (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 4.5:

8. Verify the following (CBSE 2016)

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

9. Reduce the following boolean expression to it's simplest form using K-Map (CBSE 2017)

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

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

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

11. Derive a canonical POS expression for a Boolean function G, represented by Table 4.6 (CBSE 2017)

12. Derive a canonical POS expression for a Boolean function FN, represented by Table

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 4.6:

4.7. (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 4.7:

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

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

(CBSE 2019)

14. Fig. 4.9 below shows a muliplexer where S0 and S1 are the select lines, I0 to I3 are the input lines, EN is the enable line and F(P,Q,R) is the output. Find the boolean

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

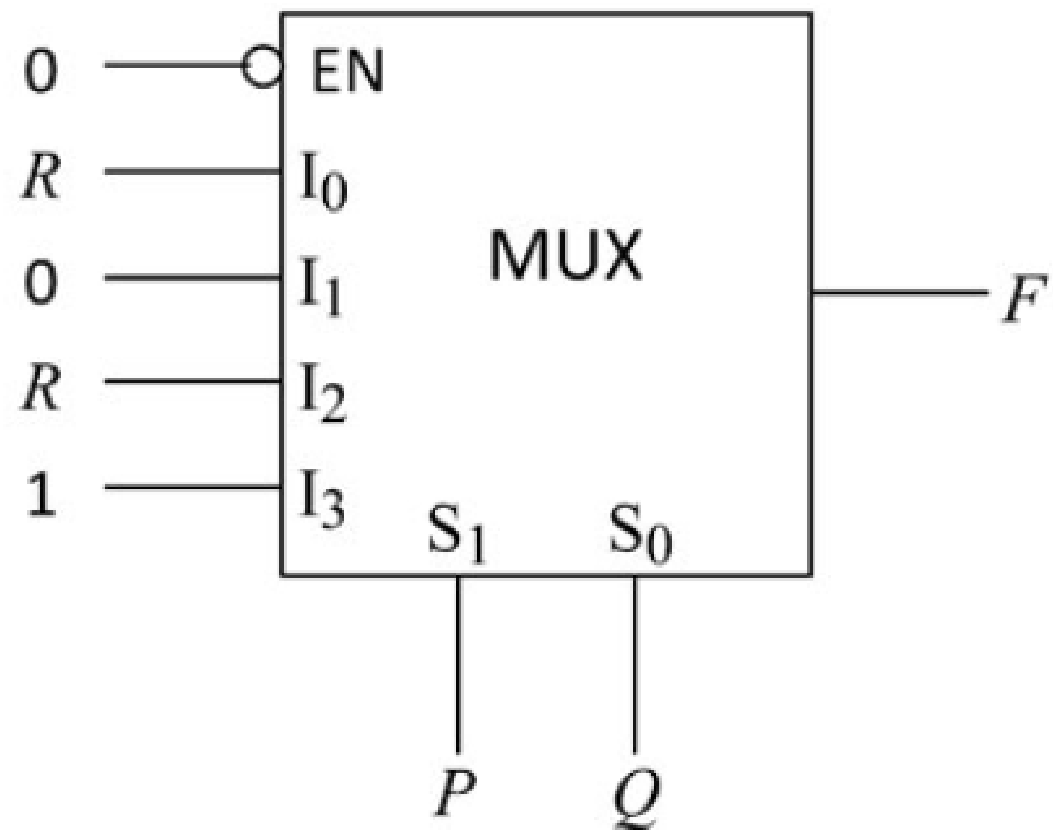


Figure 4.9:

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

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

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

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

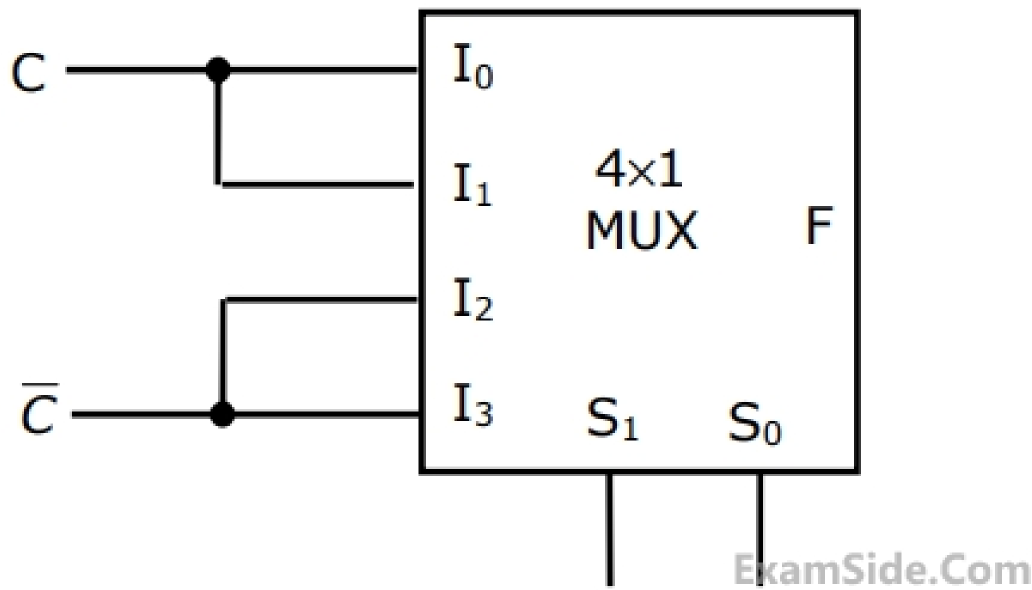


Figure 4.10:

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

    (a) A is false and B is true

    (b) A is false and C is true

    (c) A, B and C are all false

    (d) A, B and C are all true

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

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

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

18. Draw the logic circuit for Table 4.8 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 4.8:

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

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

20. Minimize the following Boolean function in 4.23.

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

21. Find the Boolean expression for Table 4.9. (GATE EC 2005)

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

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 4.9:

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

2021)

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

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

24. The propogation delays of the XOR gate, AND gate and multiplexer (MUX) in the

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

(a) 3 ns

(b) 5 ns

(c) 6 ns

(d) 7 ns

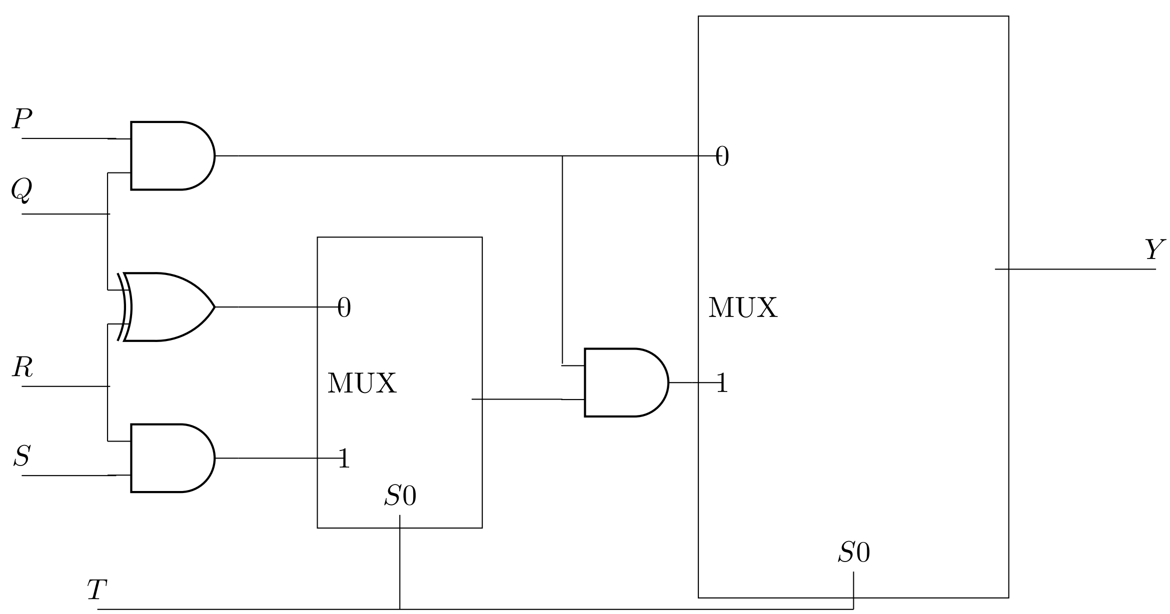


Figure 4.11:

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

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

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

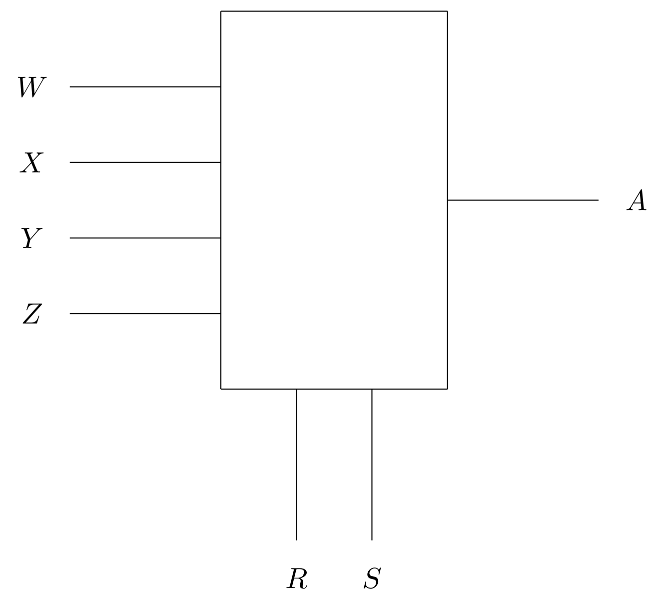


Figure 4.12:

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

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

26. $A = a_1a_0$ 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)

27. The logic block shown has an output F given by _____. (GATE IN 2022)

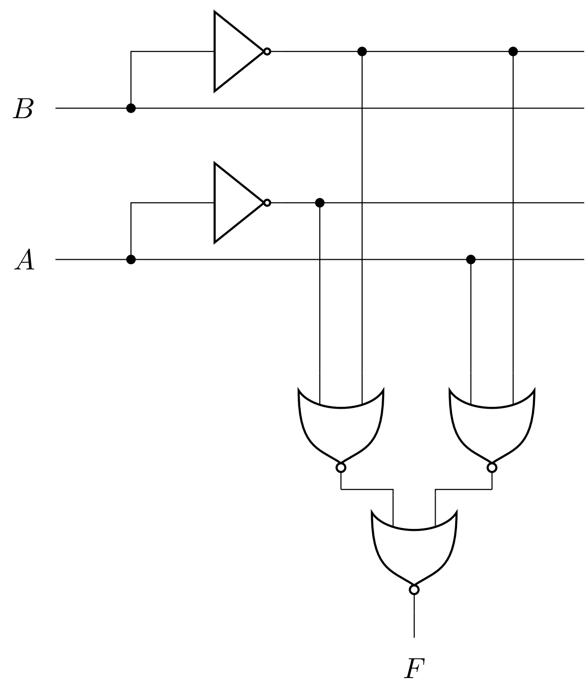


Figure 4.13: Circuit

(a) $A + B$

(b) $A.\bar{B}$

(c) $A + \bar{B}$

(d) $\bar{B}$

28. 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. 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.

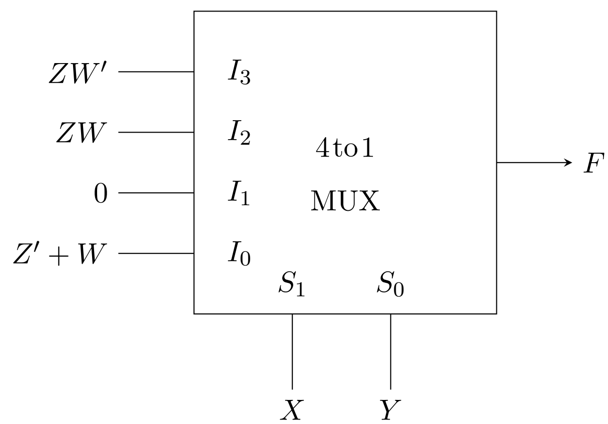


Figure 4.14: 4×1 multiplexer

(GATE IN-2021)

The canonical sum of product representation of F is:

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

29. The output expression for the Karnaugh map shown below is (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

Figure 4.15:

(a) QR'+S

(b) QR+S

(c) QR'+S'

(d) QR+S'

30. In the circuit shown below , X and Y are digital inputs, and Z is a digital output. The quivalent circuit is a (GATE EE 2019)

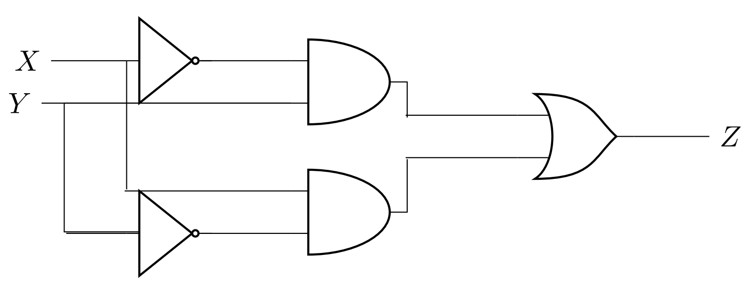


Figure 4.16:

(a) NAND gate

(b) NOR gate

(c) XOR gate

(d) XNOR gate

31. The output F of the digital circuit shown can be written in the form(s)__________
(GATE IN 2022)

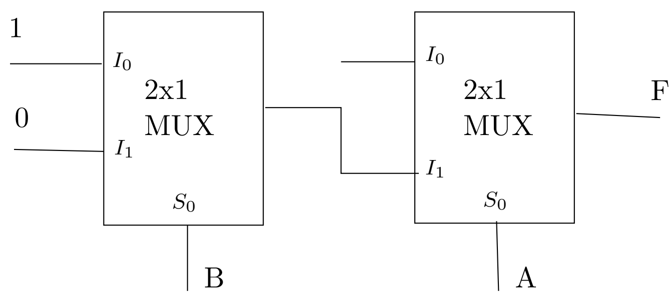


Figure 4.17:

(a) $\overline{A \cdot B}$

(b) $\overline{A}+\overline{B}$

(c) $\overline{A + B}$

(d) $\overline{A} \cdot \overline{B}$

32. If $X = X_1X_0$ 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)

33. The figure below shows the i^{th} 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. Assume that each logic gate has a delay of 2 nanosecond, with no additional time delay due to the interconnecting wires. Of the inputs A_i , B_i are available and stable throughout the carry propagation, the maximum time taken for an input C_i to produce a steady-state output C_{i+1} is __________ nanosecond. (GATE IN 2019)

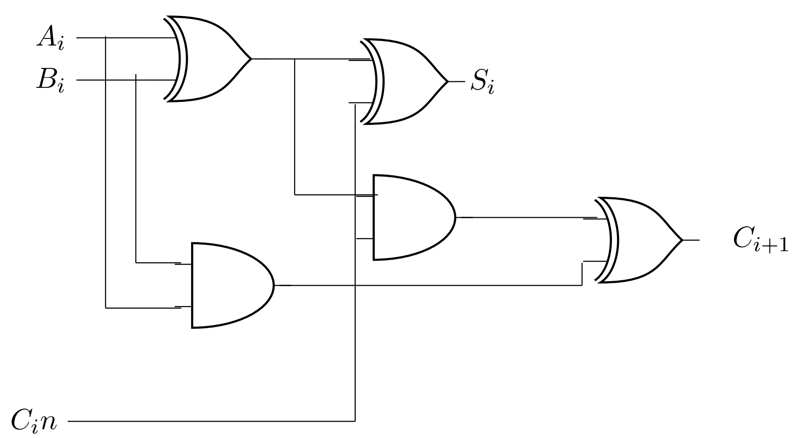


Figure 4.18:

34. The Product of sun expression of a Boolean function $F(A, B, C)$ three variables is given by

$$F(A, B, C) = (A + B + \overline{C}) \times (A + \overline{B} + \overline{C} \times (\overline{A} + B + C) \times (\overline{A} + \overline{B} + \overline{C}) \quad (4.24)$$

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

(a) $\overline{AB}C + \overline{A}BC + A\overline{BC} + ABC$

(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) $\overline{ABC} + \overline{A}BC + AB\overline{C} + ABC$

(GATE IN 2018)

35. A four-variable Boolean function is realized using 4×1 multiplexers as shown in the figure.

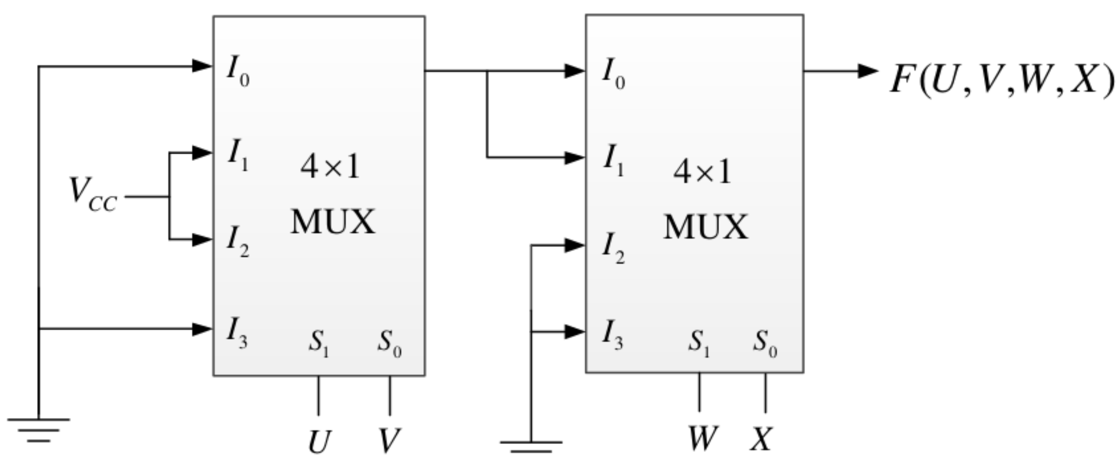


Figure 4.19:

The minimized expression for

(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) $\left(U\bar{V} + \bar{U}V\right)\left(\bar{W}\bar{X} + \bar{W}X\right)$

(GATE EC 2018)

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

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

(A) $(A+B+C)\cdot\left(A+\overline{B}+C\right)\cdot\left(\overline{A}+B+C\right)$

(B) $\left(\overline{A}+\overline{B}+\overline{C}\right)\cdot\left(\overline{A}+B+\overline{C}\right)\cdot\left(A+\overline{B}+\overline{C}\right)$

(C) $\left(A+B+\overline{C}\right)\cdot\left(A+\overline{B}+\overline{C}\right)\cdot\left(\overline{A}+B+\overline{C}\right)\cdot\left(\overline{A}+\overline{B}+C\right)\cdot\left(\overline{A}+\overline{B}+C\right)$

(D) $\left(\overline{A}+\overline{B}+C\right)\cdot\left(\overline{A}+B+C\right)\cdot\left(A+\overline{B}+C\right)\cdot\left(A+B+\overline{C}\right)\cdot(A+B+C)$

(GATE EC 2018)

37. In the Karnaugh map shown below, X denotes a don't care term. What is the minimal form of the function represented by the Karnaugh map?

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

(A) $b'd' + a'd'$

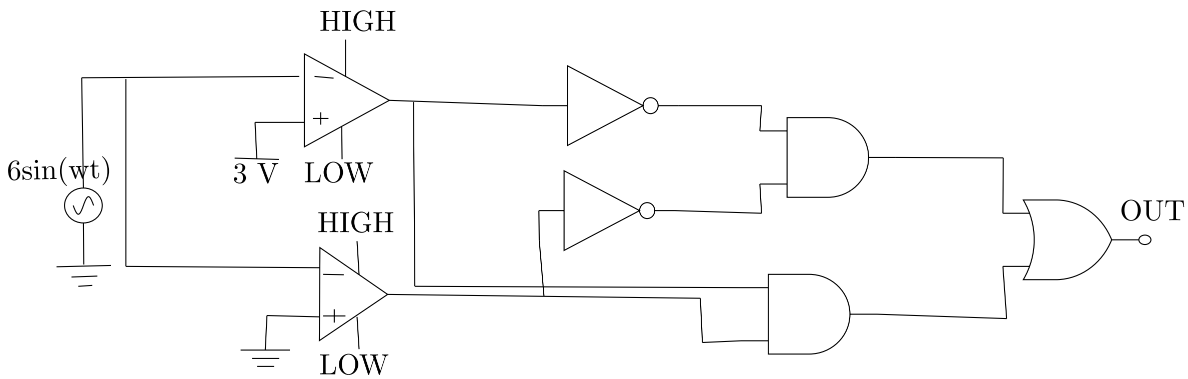
(B) $a'b' + b'd' + a'bd'$

(C) $b'd' + a'bd'$

(D) $a'b' + b'd' + a'd'$

(GATE EC 2008)

38. In the circiut shown below , assume that the comparators are ideal and all components have zero propagation delay . In one period of the input signal Vin=6sin(wt),the fraction of the time which the output OUT is in login state HIGH is



(a) $\frac{1}{12}$

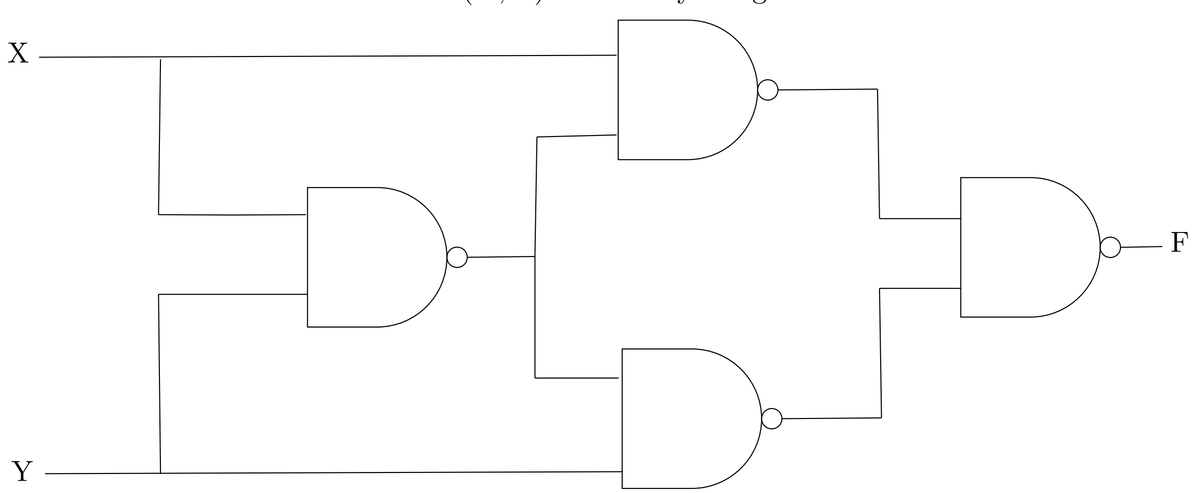
(b) $\frac{1}{2}$

(c) $\frac{2}{3}$

(d) $\frac{5}{6}$

(GATE EC 2018)

39. The Boolean function $F(X, Y)$ realized by the given circuit is



(a) $\bar{X}Y + X\bar{Y}$

(b) $\bar{X}\bar{Y} + XY$

(c) $X + Y$

(d) $\bar{X}.\bar{Y}$

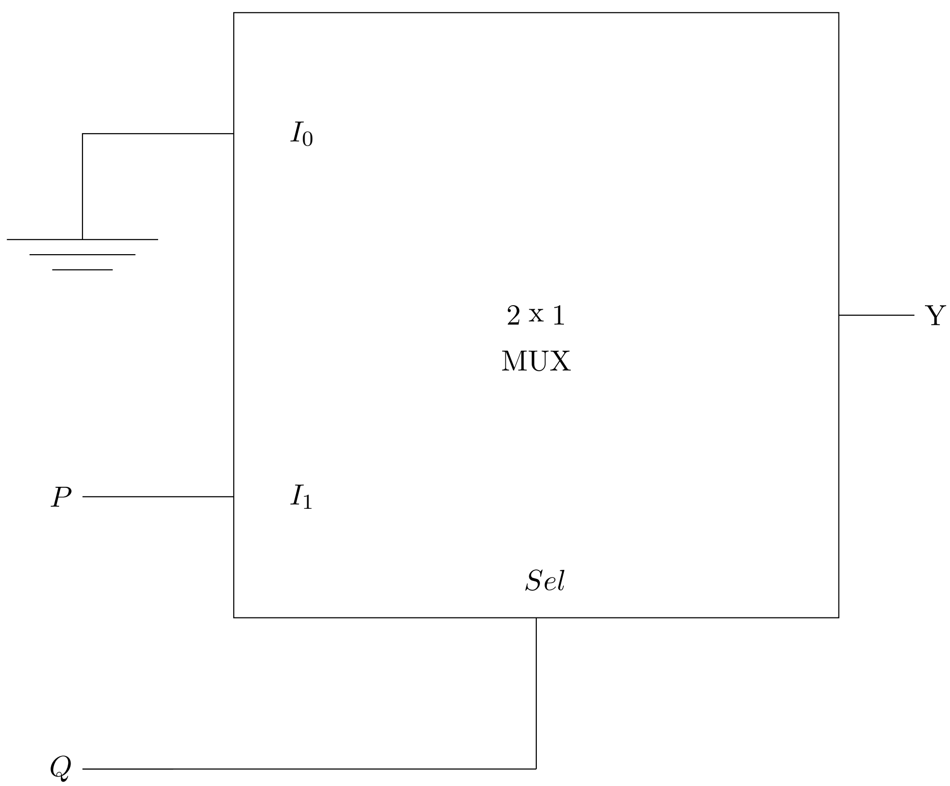
(GATE IN 2019)

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

41. In the circuit shown below, P and Q are the inputs. The logical function realized by the circuit shown below



(a) Y=PQ

(b) Y=P+Q

(c) Y= $\overline{PQ}$

(d) Y= $\overline{P+Q}$

(GATE EC2023,23)

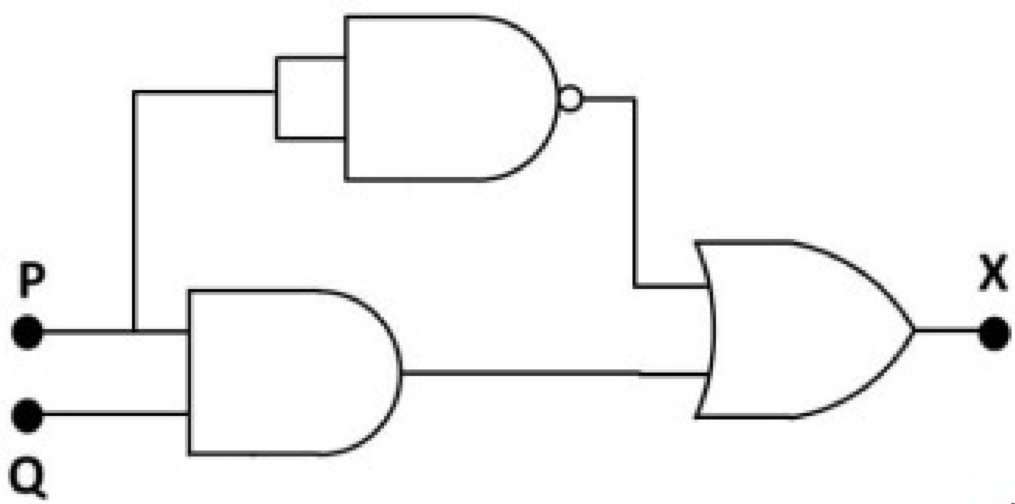


Figure 4.20: k-maps

Fig. 1

(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

(GATE PH2023,24)

42. Consider the 2-bit multiplexer(MUX) shown in the figure. For OUTPUT to be the XOR of C and D, th values for A_0, A_1, A_2 and A_3 are ____ (GATE EC 2022)

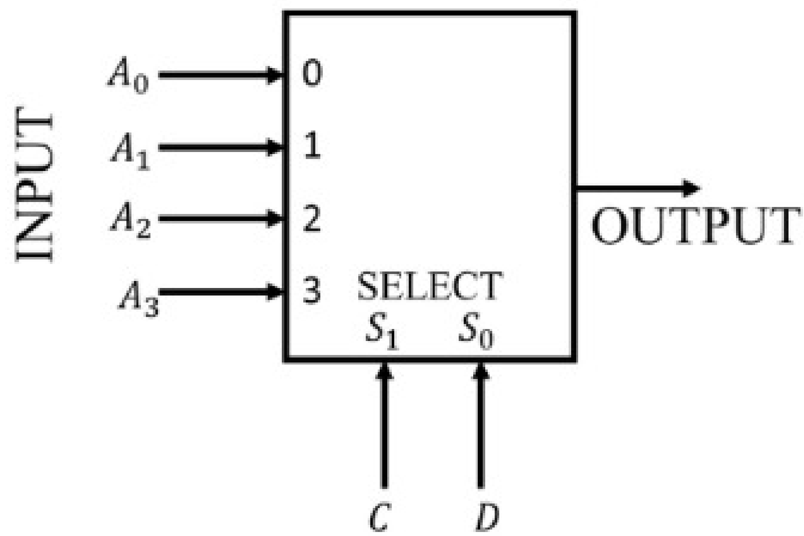


Figure 4.21: MUX

(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) $A_0 = 1, A_1 = 1, A_2 = 0, A_3 = 0$

43. 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 ____________

(a) $WX + \bar{W}X\bar{Y} + W\bar{X}Y$

(b) $WX + WY + X\bar{Y}$

(c) $X\bar{Y} + WY$

(d) $\bar{X}Y + \bar{W}\bar{Y}$

(GATE IN 2023)

44. Q, R, S are Boolean variables $\oplus$ and is the XOR operator. Select the CORRECT option(s).

(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) $((Q \oplus R) \oplus (R \oplus S)) \oplus (Q \oplus S) = 1$

(Gate BM 2023)

45. The output F of the digital circuit shown can be written in the form(s)__________

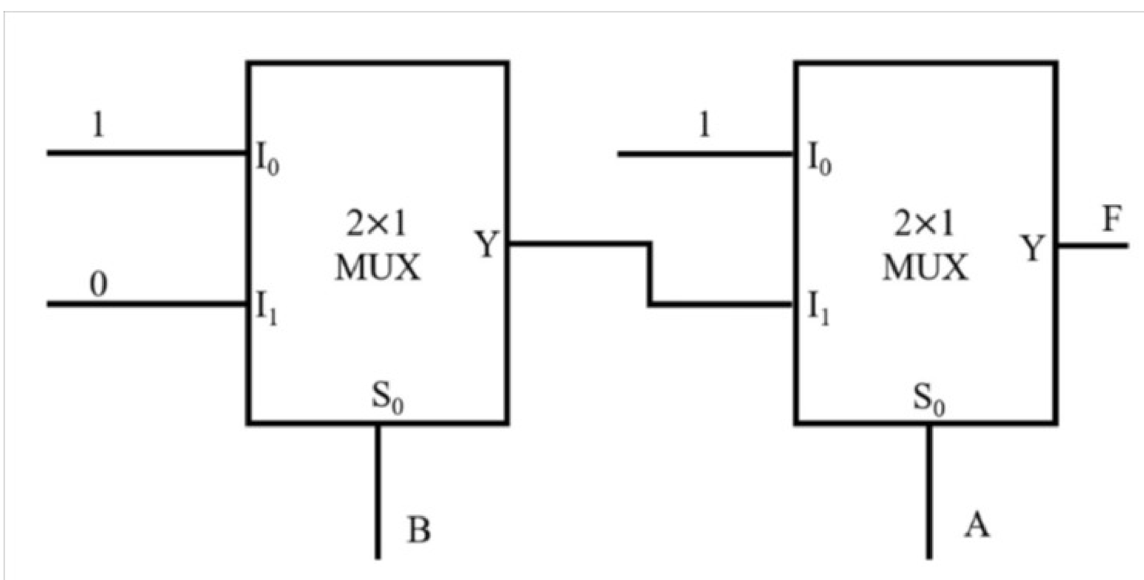


Figure 4.22:

(a) $\overline{A.B}$

(b) $\bar{A}+\bar{B}$

(c) $\overline{A+B}$

(d) $\bar{A}.\bar{B}$

(GATE IN 2022)

46. $\mathbf{A} = a_1a_0$ 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)

47. The minimum number of two-input NAND gates required to implement the following Boolean expression is ____________________

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

(GATE-PH-2022)

48. In the circuit shown below, 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-BM2021)

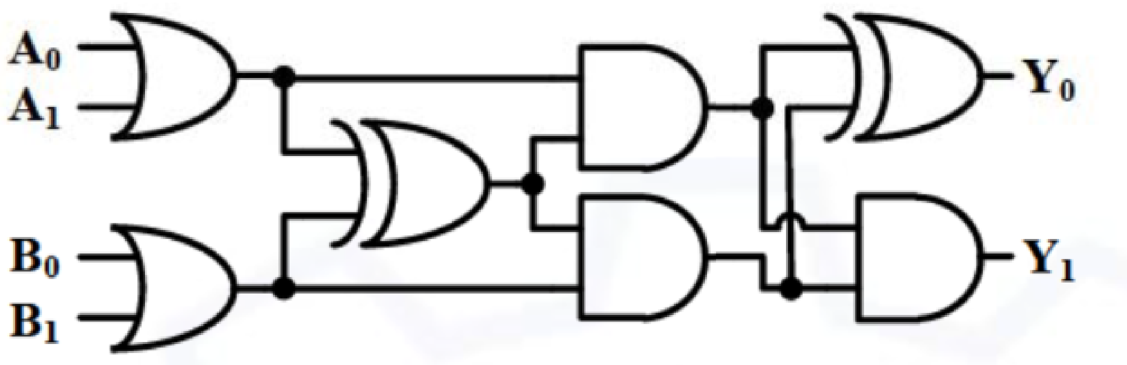


Figure 4.23:

(A) 01

(B) 10

(C) 00

(D) 11

49. Match the Boolean expression with its minimal realization. (GATE BM 2020)

	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	$Y(X+Z)$

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

50. 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. 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 Is are the input lines of the multiplexer. (GATE-IN2021,35)

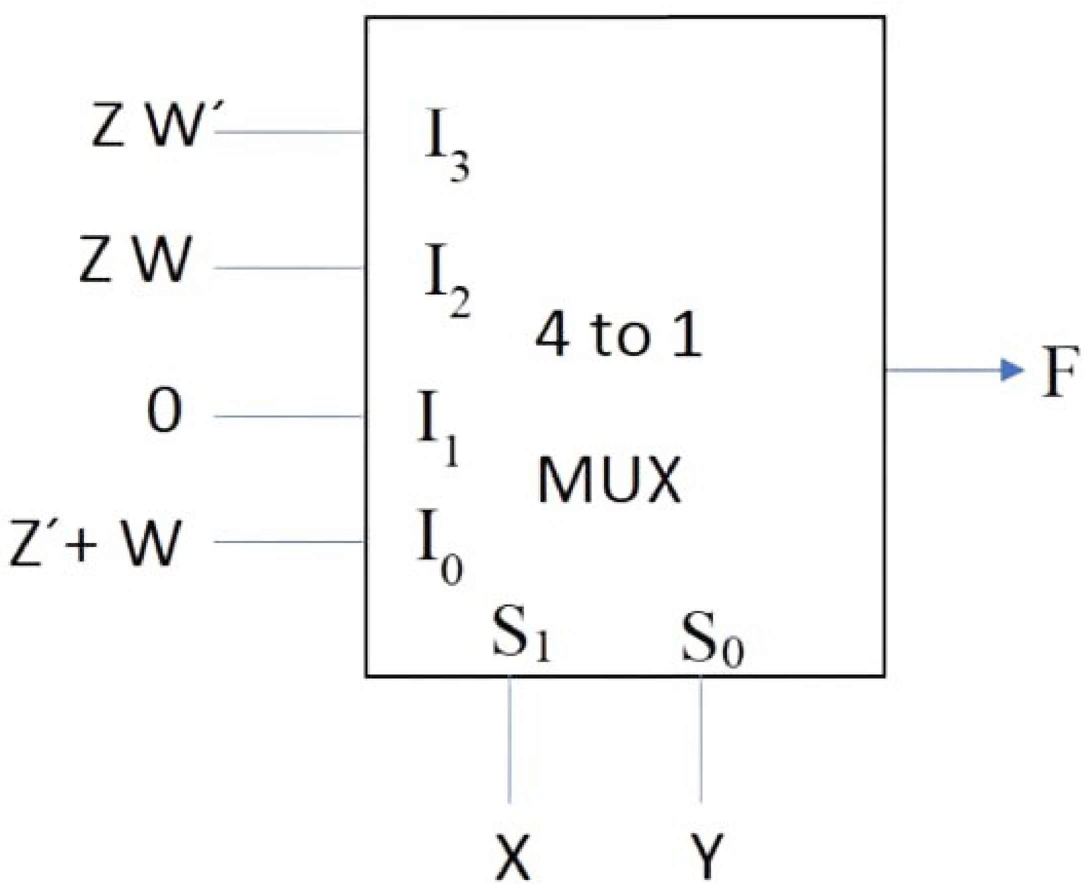


Figure 4.24: Multiplexer

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

51. The output expression for the Karnaugh map shown below is (GATE-EE2019,35)

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

Figure 4.25: K-MAP

(a) $Q\overline{R} + S$

(b) $Q\overline{R} + \overline{S}$

(c) $QR + S$

(d) $QR + \overline{S}$

52. The Boolean expression for the shaded regions as shown in the figure is __________.

$(GATE\ IN2020 - 11)$

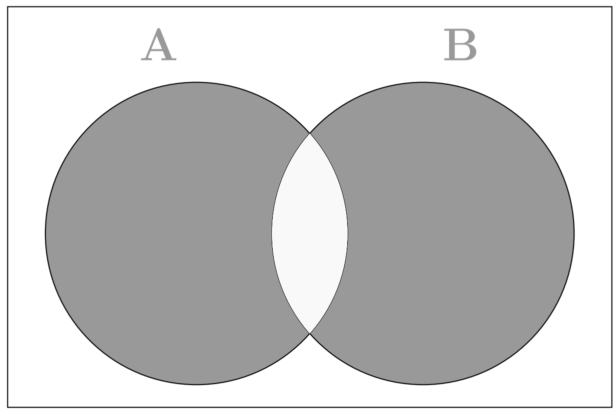


Figure 4.26: Venn Diagram

(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) $(\overline{A}+\overline{B})\bullet(A+\overline{B})$

53. The output expression of the Karnaugh map shown below is (GATE EE-2017, 36)

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

s

A. $B\overline{D}+BCD$

B. $B\overline{D} + AB$

C. $\overline{B}D + ABC$

D. $B\overline{D} + ABC$

54. A Boolean function F of three variables $X, Y,$ and Z is given as

$F(X,Y,Z) = (X'+Y+Z) \cdot (X+Y'+Z') \cdot (X'+Y+Z') \cdot (X'Y'Z'+X'YZ'+XYZ')$.

Which one of the following is true? (GATE IN2021,31)

(a) $F(X,Y,Z) = (X+Y+Z') \cdot (X'+Y'+Z')$

(b) $F(X,Y,Z) = (X'+Y) \cdot (X+Y'+Z')$

(c) $F(X,Y,Z) = X'Z' + YZ'$

(d) $F(X,Y,Z) = X'Y'Z + XYZ$

55. The figure below shows a multiplier where S_1 and S_0 are select lines, I_0 to I_3 are the input data lines, EN is the enable line, and $F\,(P,Q,R)$ is the output. F is

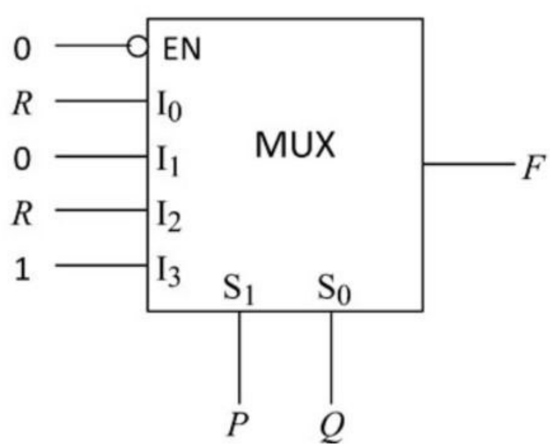


Figure 4.27:

(GATE EC 2020)

(A) $PQ + \bar{Q}R$.

(B) $P + Q\bar{R}$.

(C) $P\bar{Q}R + \bar{P}Q$.

(D) $\bar{Q} + PR$.

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

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

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

(GATE IN 2018)

(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) $\bar{A}\ \bar{B}\ \bar{C} + \bar{A}\ B\ C + A\ B\ \bar{C} + A\ B\ \bar{C} + A\ B\ C$

57. 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

(GATE-2018,EE,37)

(a) $\bar{A}\bar{C} + \bar{B}C + AC$

(b) $\bar{A} + C$

(c) $\bar{C} + A$

(d) $\bar{A}C + BC + A\bar{C}$

Chapter 5

7474

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

5.1. Components

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

Table 5.1:

5.2. Decade Counter

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

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

Table 5.2:

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

   The pin diagram for 7474 is available in Fig. 5.1

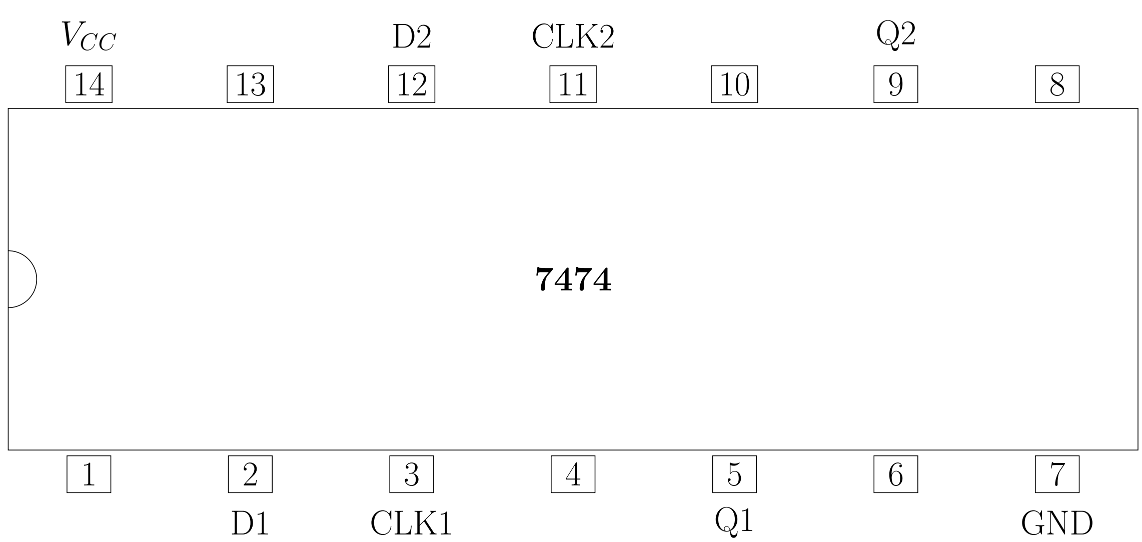


Figure 5.1:

3. Realize the decade counter in Fig. 5.2.

5.3. 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 $000, 001, 011, 010, 110, 111, 10$ 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 (GATE-EE2021,37)

   (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) $Q_2Q_1 + \bar{Q}_2\bar{Q}_1$

2. The propagation delay of the exclusive- OR(XOR) gate in the circuit in the figure is 3 ns. The propagation delay of all the flip-flops is assumed to be zero. The Clock(clk) frequency provided to the circuit is 500 MHz.

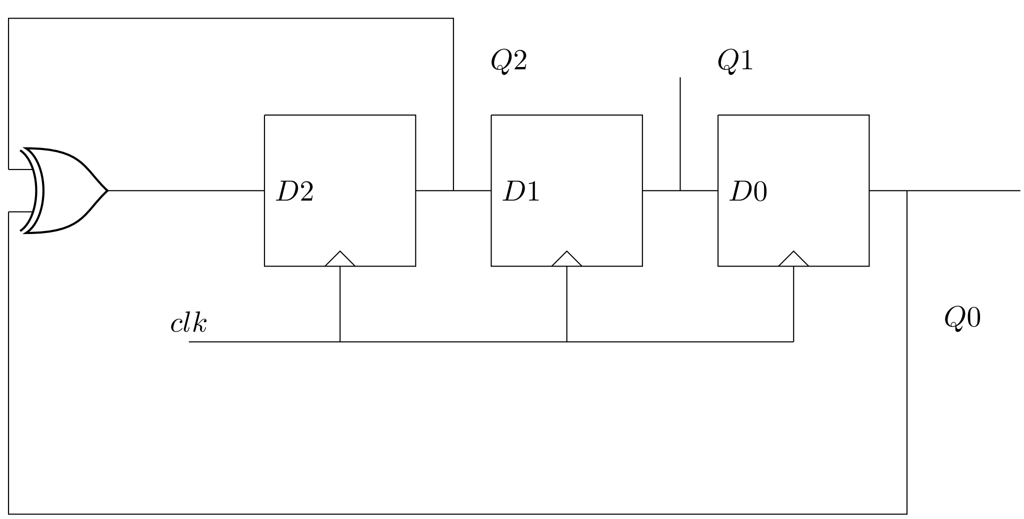


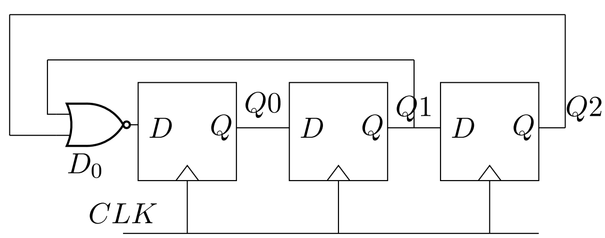
Figure 5.3: Propagation Delay

Starting from the initial value of the flip-flop outputs $Q2Q1Q0 = 111$ with $D2 = 1$,the minimum number of triggering clock edges after which the flip-flop outputs $Q2Q1Q0$ becomes 1 0 0*(*in integer) is ___

(GATE-EC2021,46)

3. The maximum clock frequency in MHz of a 4-stage ripple counter, utilizing flip-flops, with each flip-flop having a propagation delay of 20 ns, is ______. (round off to one decimal place) (GATE EE 2022)

4. The digital circuit shown ______



(A) is a divide-by-5 counter

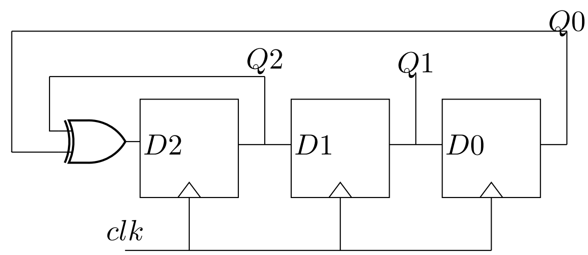
(B) is a divide-by-7 counter

(C) is a divide-by-8 counter

(D) does not function as a counter due to disjoint cycles of states

GATE IN 2022

5. The propogation delay of the exclusive-OR(XOR) gate in the circuit in the figure is 3ns.The propogation delay of all the flip-flops is assumed to be zero.The clock(Clk) frequency provided to the circuit is 500MHz. (GATE EC 2021)



Starting from the initial value of the flip-flop outputs $Q2Q1Q0 = 111$ with $D2 = 1$,the minimum number of triggering clock edges after which the flip-flop outputs $Q2Q1Q0$ becomes 1 0 0*(in integer)* is __

6. For the 3-bit binary counter shown in the figure, 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 $0V$, the current I(in mA) flowing through the 50Ω resistor during the 5th clock cycle is (up to one decimal place) (GATE IN 2018)

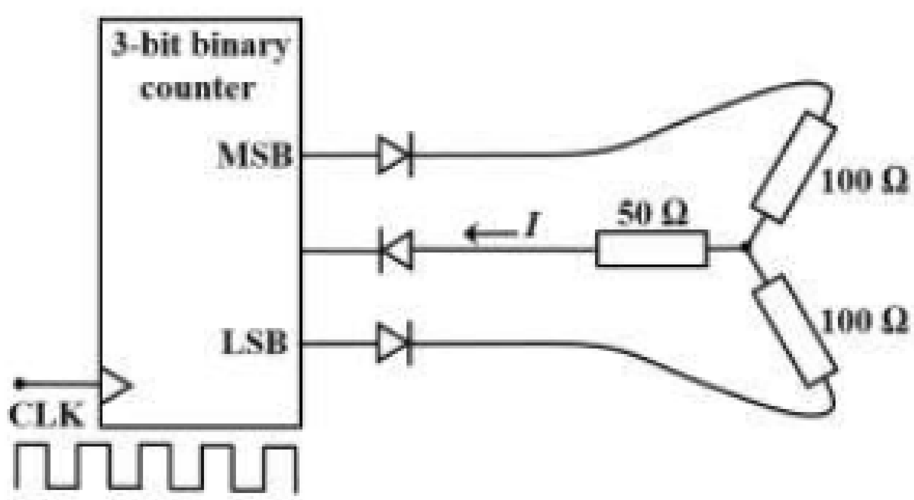


Figure 5.4: circuit

7. Consider the sequential circuit shown in the figure, where both flip-flops used are positive edge-triggered D flip-flops.

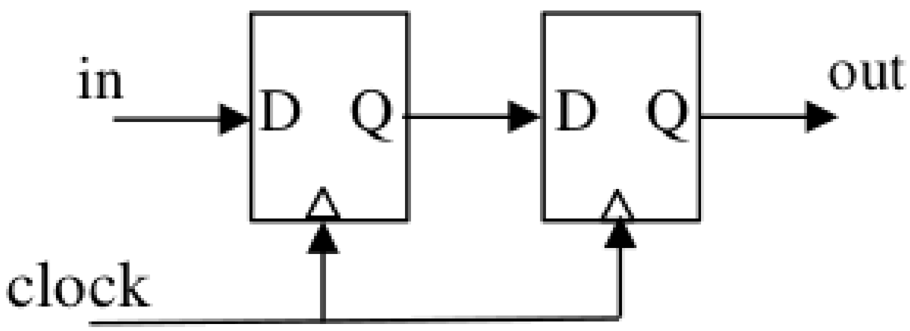
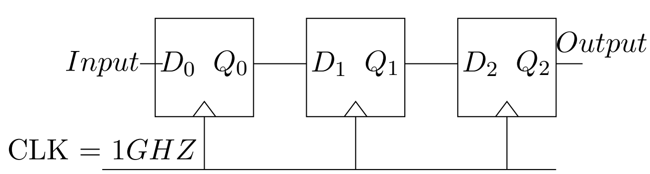


Figure 5.5: ckt

8. 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 _____. (GATE IN 2018)

9. The synchronous sequential circuit shown below works at a clock frequency of $1GHz$. The throughput, in $Mbits/s$,and the latency, in ns, respectively, are



(a) $1000, 3$

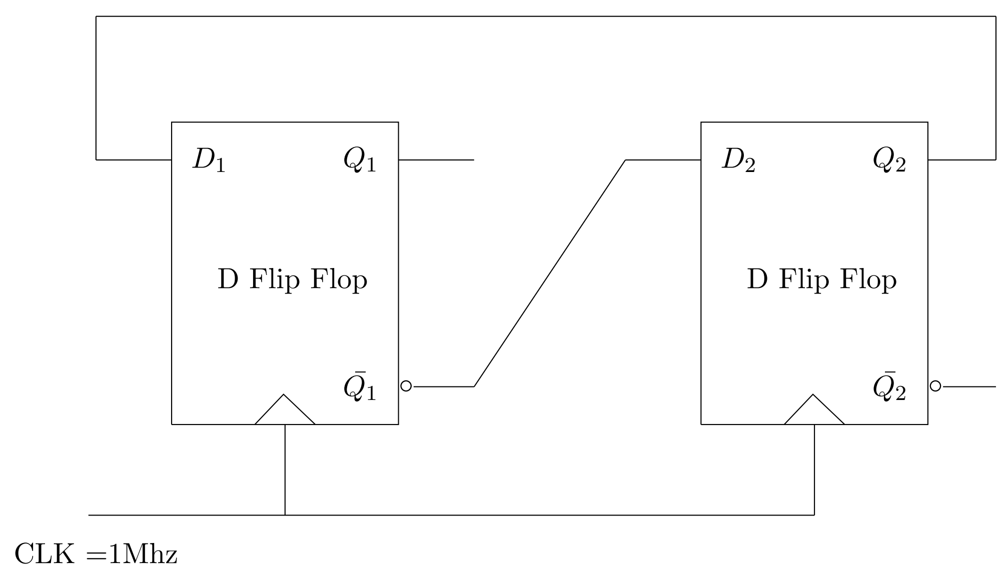
(b) $333.33, 1$

(c) $2000, 3$

(d) $333.33, 3$

(GATE EC 2023)

10. In a given sequential circuit, initial states are $Q1 = 1$ and $Q2 = 0$. For a clock frequency of $1MHz$, the frequency of signal $Q2$ in kHz, is(rounded off to the nearest integer)



(GATE EC 2023)

11. Neglecting the delays due to the logic gates in the circuit shown in figure, the decimal equivalent of the binary sequence $[ABCD]$ of initial logic states, which will not change with clock, is ___________.

(EE GATE 2023)

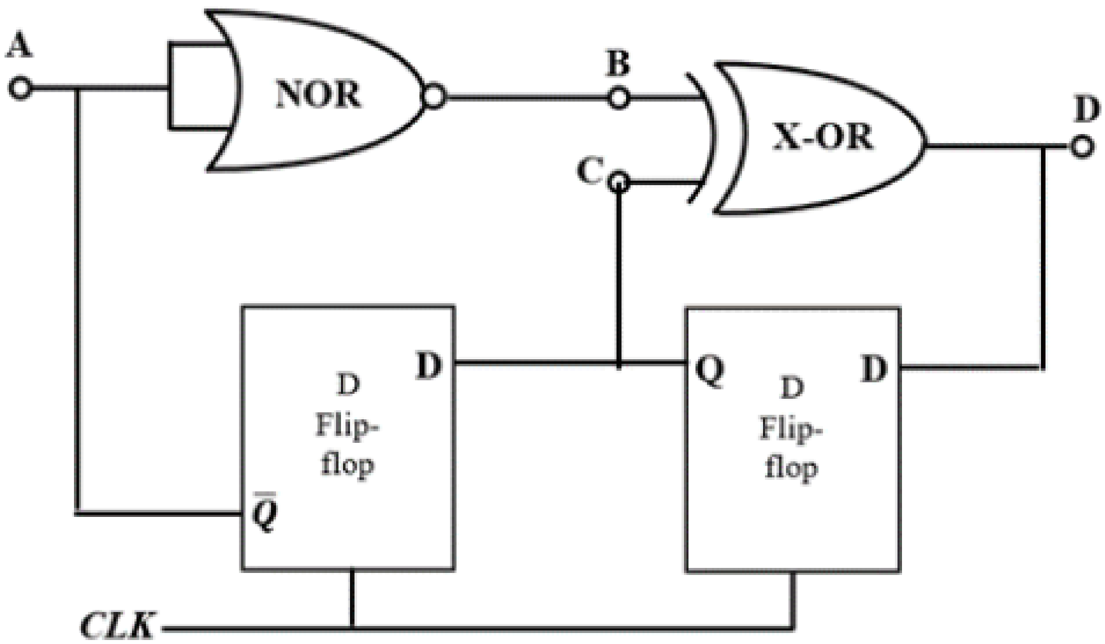


Figure 5.6: Neglecting the delays

12. Consider a sequential digital circuit consisting of T flip-flops and D flip-flops as shown in the figure. CLKIN is is the clock input to the circuit. At the beginning,Q1,Q2 and Q3 have values 0,1 and 1, respectively. (GATE CS2023,43)

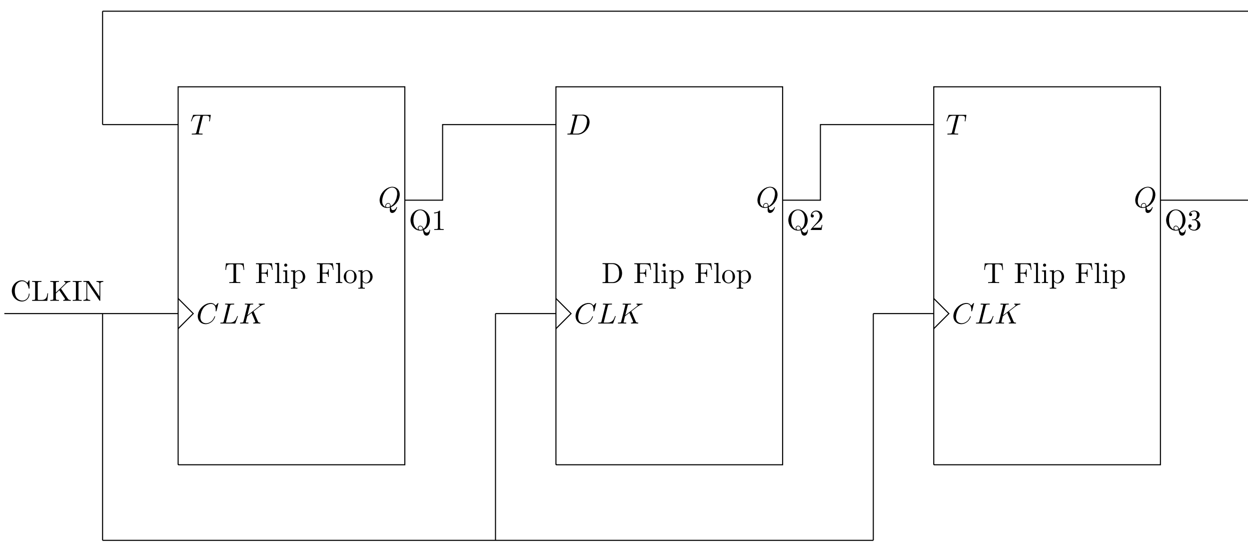


Figure 5.7: Flip-Flop

Which of the given values of (Q_1, Q_2, Q_3) can NEVER be obtained with this digital circuit?

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

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

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

(d) $(1, 1, 1)$

13. In the circuit shift, the initial binary content of the shift register A 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,what will be the binary content of the shift registers A and B after clock pulses?

(GATE IN 2023)

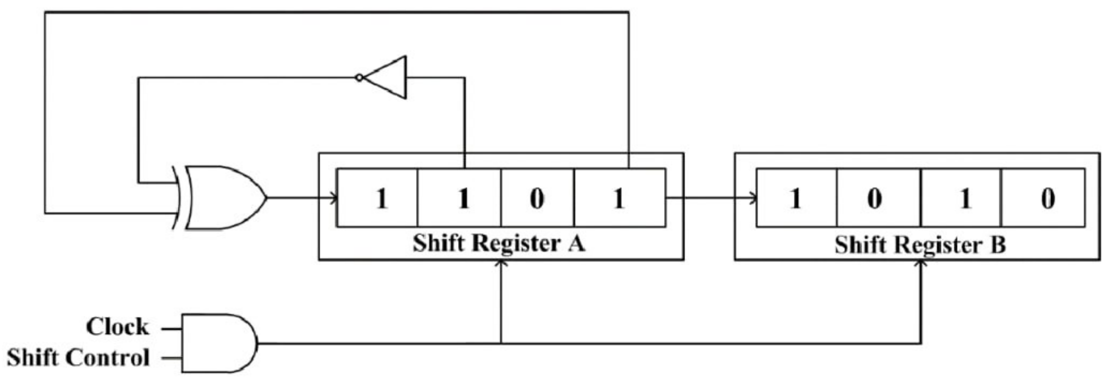


Figure 5.8: circuit Digram

(a) $A = 1101, B = 1101$

(b) $A = 1110, B = 1001$

(c) $A = 0101, B = 1101$

(d) $A = 1010, B = 1111$

14. For the circuit shown, 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%

15. In the circuit shown, the initial binary content of 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,what will be the binary content of the shift registers A and B after four clock pulses? (Gate IN 2023)

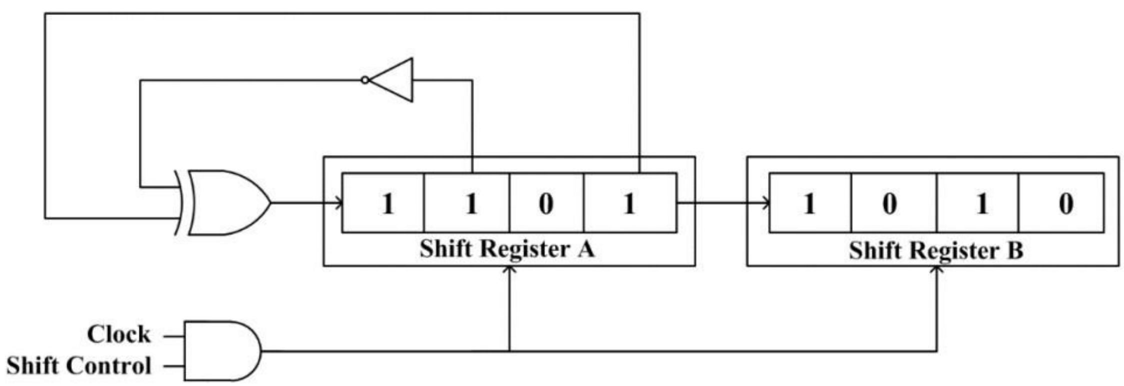


Figure 5.10: Shift register

(a) A=1101,B=1101

(b) A=1110,B=1001

(c) A=0101,B=1101

(d) A=1010,B=1111

16. The digital circuit shown in Fig. 5.11

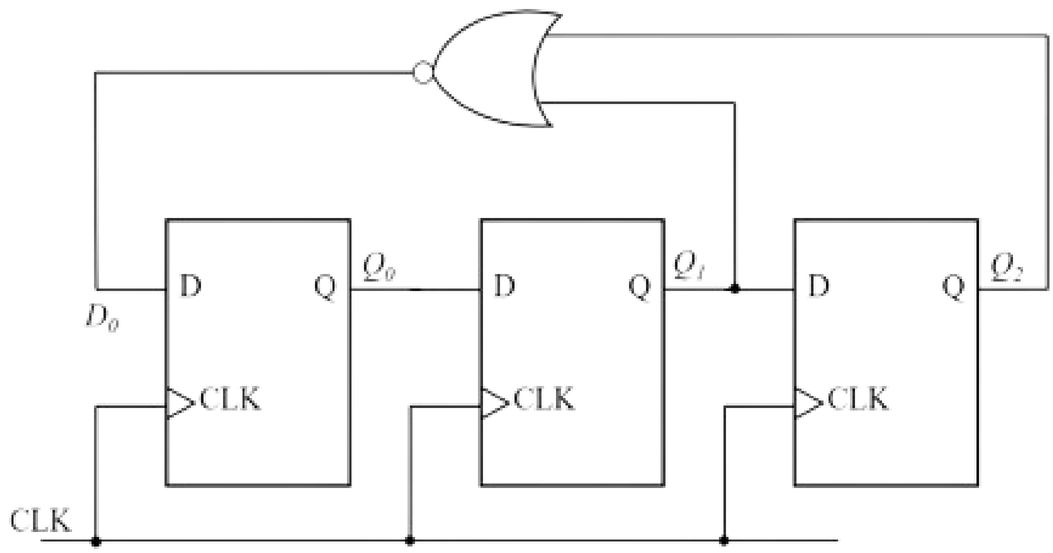


Figure 5.11:

(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

(GATE-IN-2022)

17. Given below 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-IN2021)

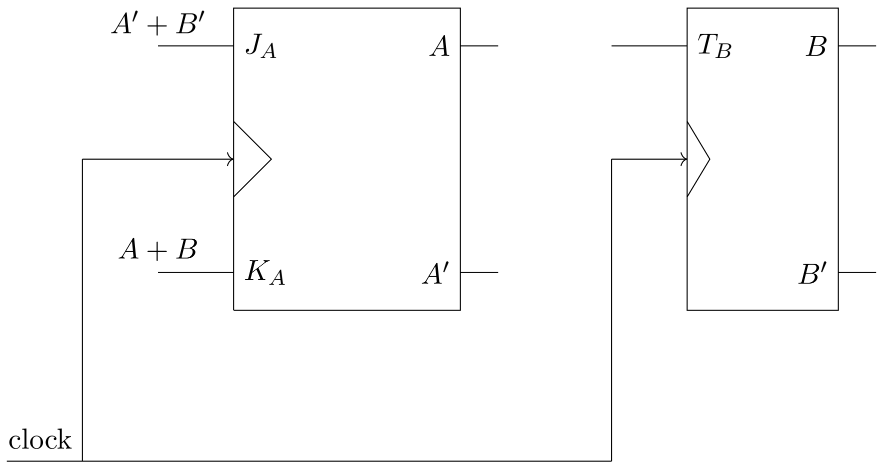


Figure 5.12: synchronous

(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) $00 \rightarrow 01 \rightarrow 11 \rightarrow 00 \ldots$

18. Consider the D-Latch shown in the figure, 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)

19. 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. 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)

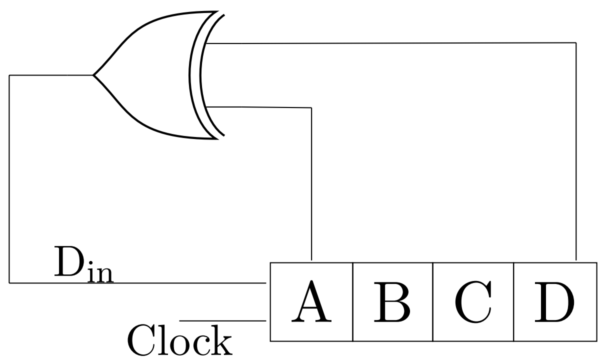


Figure 5.14:

20. In the circuit shown, the clock frequency, i.e., the frequency of the clk signal, is 12 KHz. The frequency of the signal at **Q2** is ___ KHz. (GATE-EC2019,25)

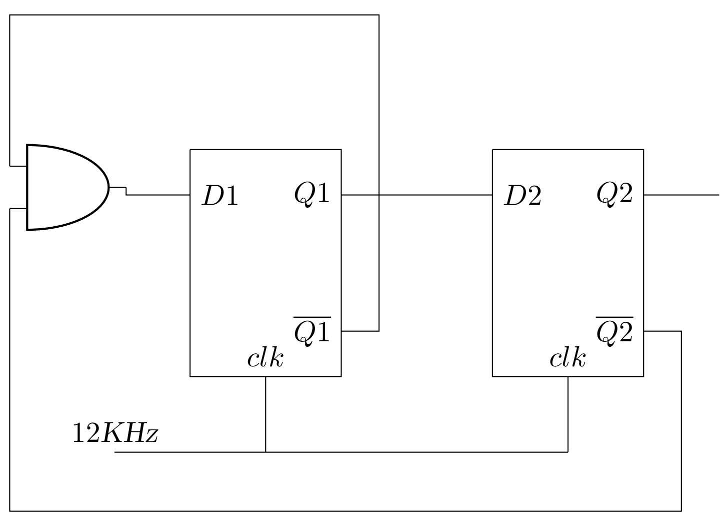
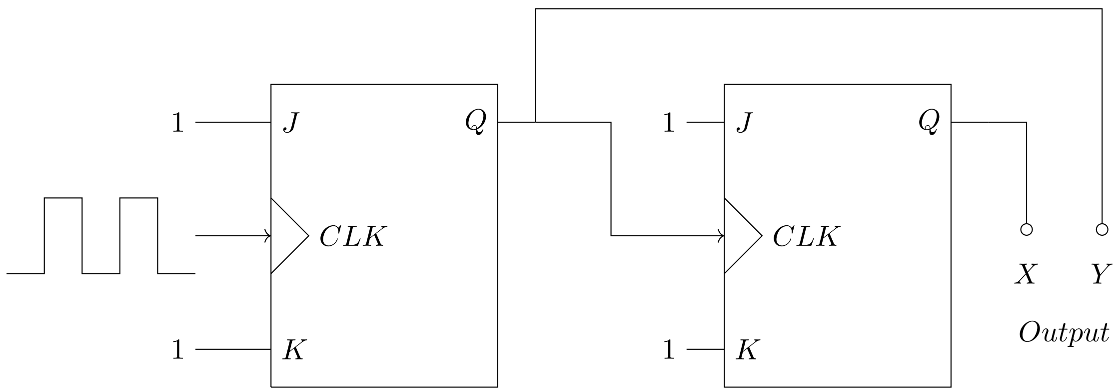


Figure 5.15: Circuit Daigram

21. The circuit shown in the figure below uses ideal positive edge-triggered synchronous $J-K$ flip flops with outputs X and Y. If the initial state of the output is $X=0$ and $Y=0$ just before the arrival of the first clock pulse, the state of the output just before the arrival of the second clock pulse is (GATE-IN2019,12)



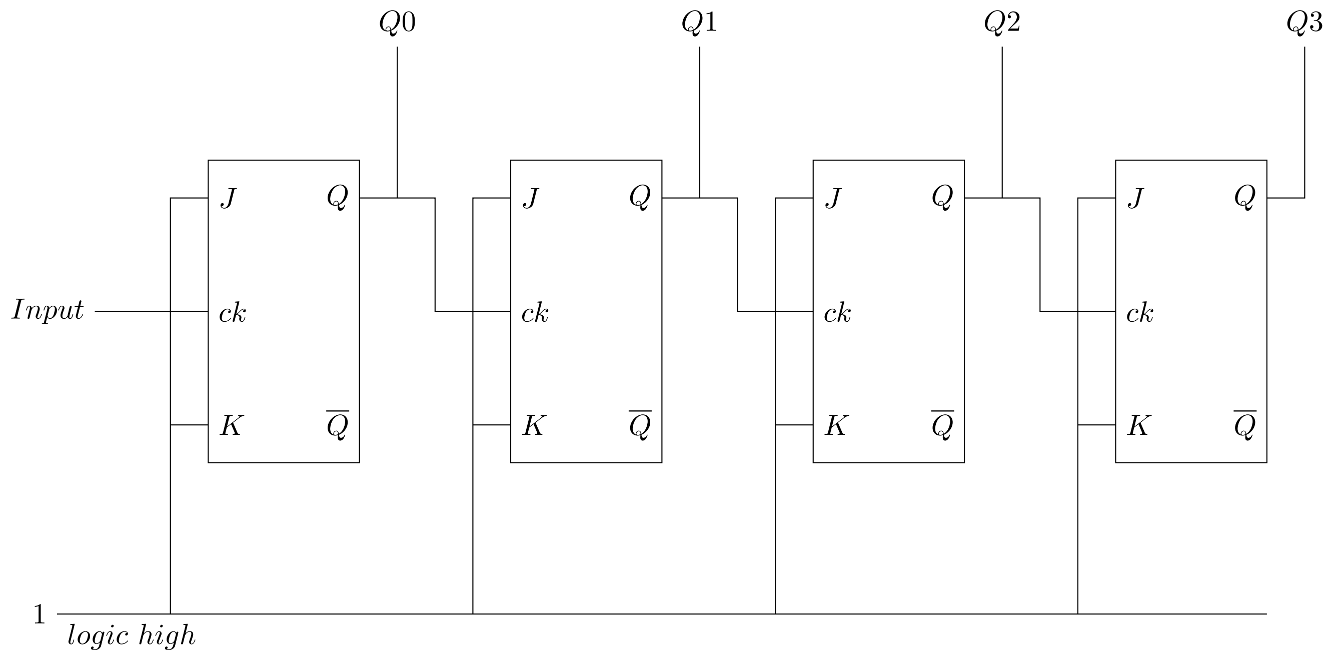
(a) $X = 0, Y = 0$

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

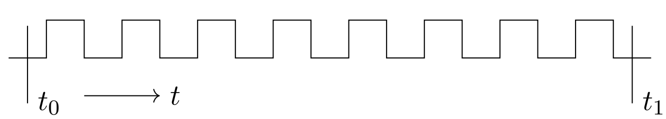
(c) $X = 1, Y = 0$

(d) $X = 1, Y = 1$

22. 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 following flip-flop (see the figure).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=t0 the outputs are in logic low state (Q0 = Q1 = Q2 = Q3 = 0).Then at t=t1,the logic state of the outputs is (GATE-PH2020,30)



4 − bit ripple counter



Input signal

Figure 5.16: Ripple Counter

(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

23. Two T-flip flops are interconnected as shown in the figure. 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 3_{rd} clock cycle is ___________. $(GATE\ \ IN2020-40)$

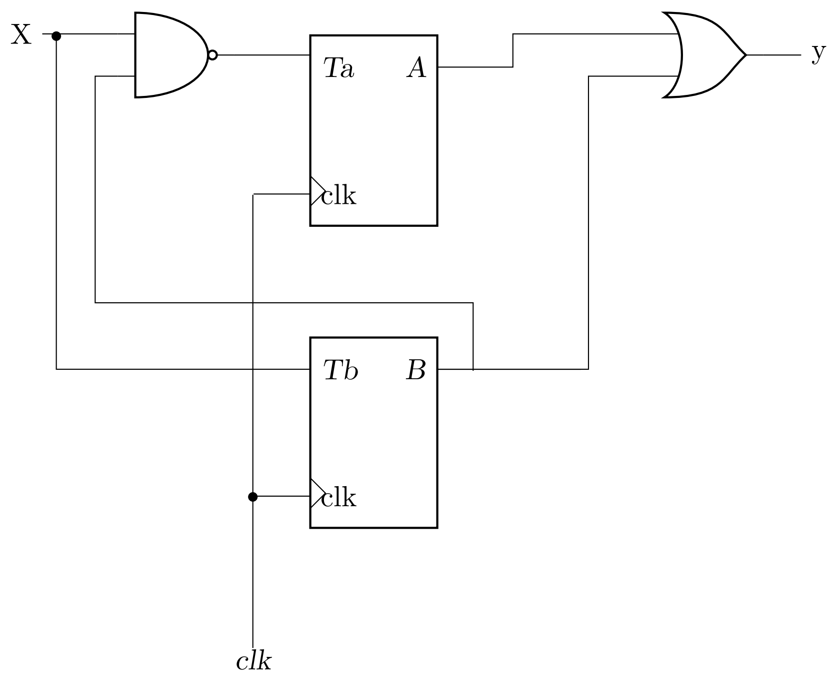


Figure 5.17: Circuit Diagram

24. For the components in the sequential circuit shown below, t_{pd} is the propagation delay, t_{setup} is the setup time, and t_{hold} is the hold time. The maximum clock frequency (rounded off to the nearest integer) at which the given circuit can operate reliably is _____MHZ. $(GATE\ \ EC2020-50)$

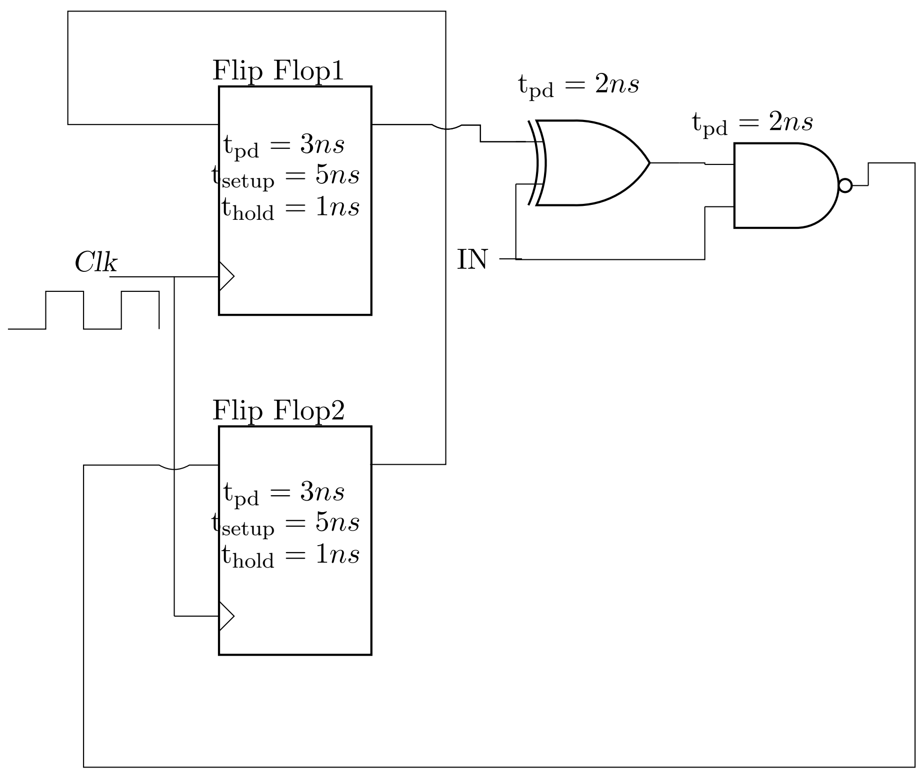


Figure 5.18: Circuit Diagram

25. A 2-bit synchronous counter using two J-K flip flops is shown. 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-IN2018,44

    A. $00 \to 11 \to 10 \to 01 \to 00 \ldots$

    B. $00 \to 01 \to 10 \to 11 \to 00 \ldots$

    C. $00 \to 01 \to 11 \to 10 \to 00 \ldots$

    D. $00 \to 10 \to 11 \to 01 \to 00 \ldots$

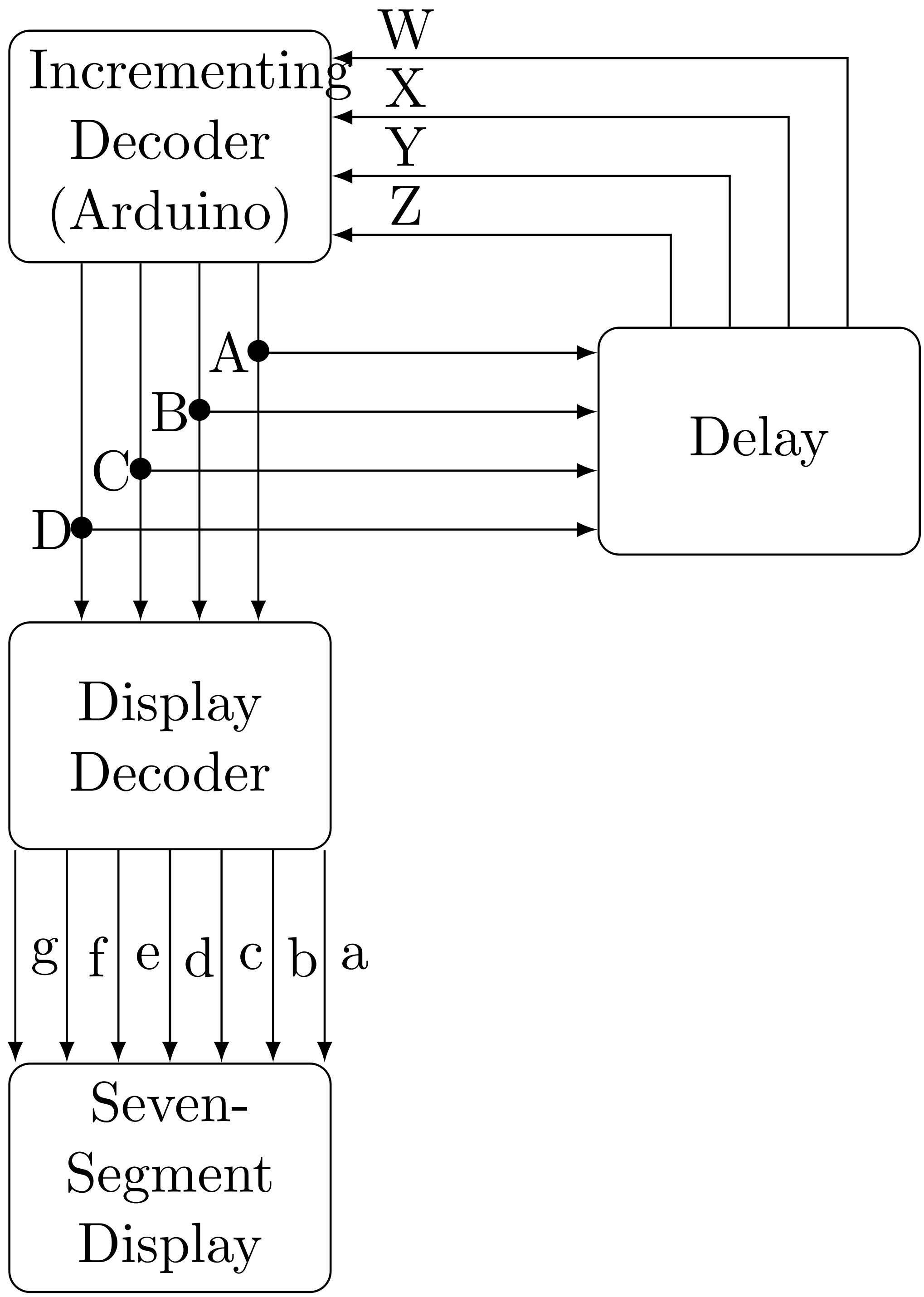

Incrementing Decoder (Arduino)
W
X
Y
Z
A
B
C
D
Delay
Display Decoder
g
f
e
d
c
b
a
Seven-Segment Display

Figure 5.2:

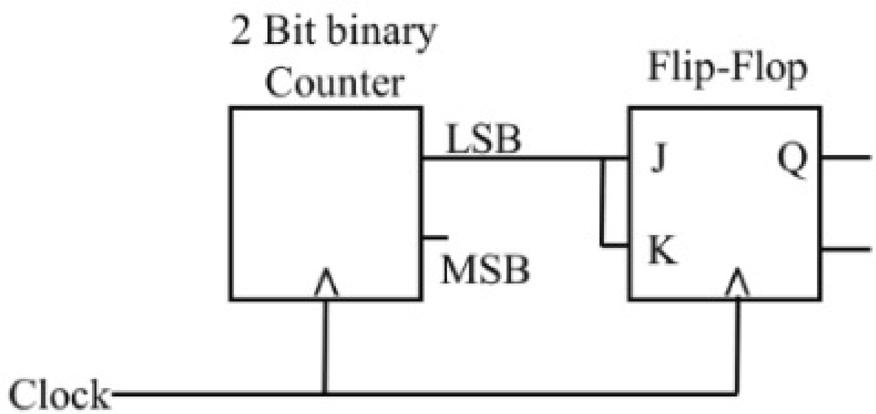


Figure 5.9: Circuit

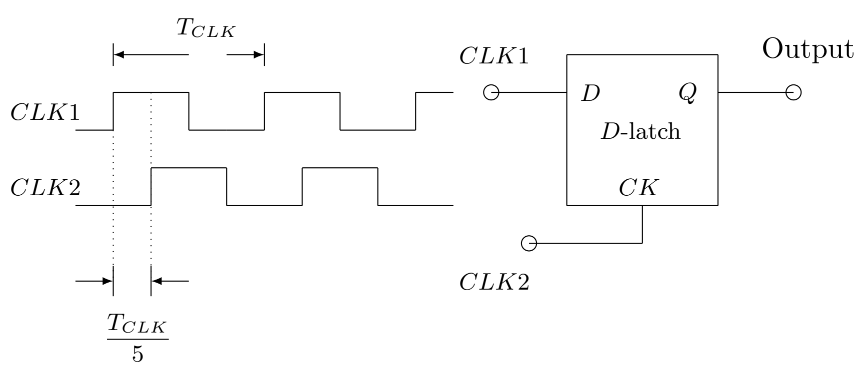
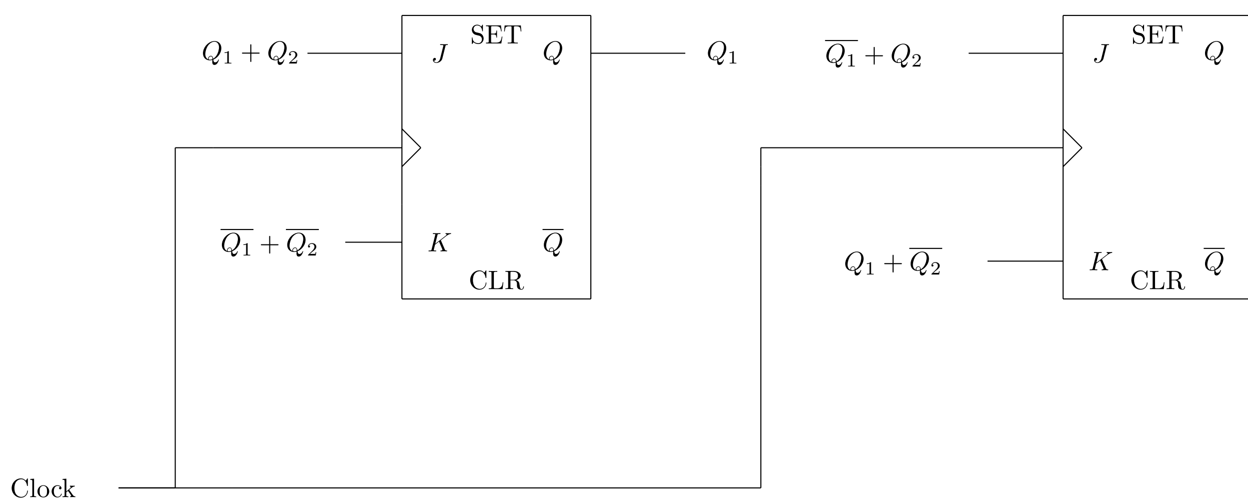


Figure 5.13:


$Q_1 + Q_2$
J
SET
Q
Q_1
$\overline{Q_1} + Q_2$
J
SET
Q
$\overline{Q_1} + \overline{Q_2}$
K
$\overline{Q}$
CLR
$Q_1 + \overline{Q_2}$
K
CLR
$\overline{Q}$
Clock

Chapter 6

Finite State Machine

We explain a state machine by deconstructing the decade counter

6.1. The Decade Counter

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

6.2. Finite State Machine

1. Fig 6.1 shows a finite state machine (FSM) diagram for the decade counter in Fig 5.2 s_0 is the state when the input to the incrementing decoder is 0 The state transition table for the FSM is Table 3.5, where the present state is denoted by the variables W, X, Y, Z and the next state by A, B, C, D.

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

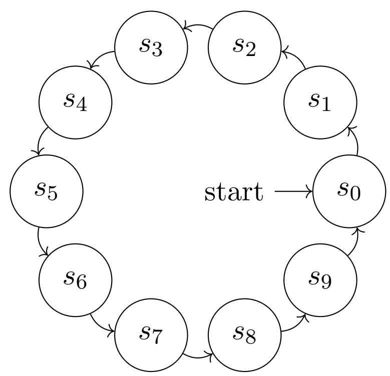


Figure 6.1: FSM for the decade counter

3. The hardware cost of the system is given by

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

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

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

5. Write the state transition table for the down counter

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

7. Verify your design using an arduino

6.3. Problems

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

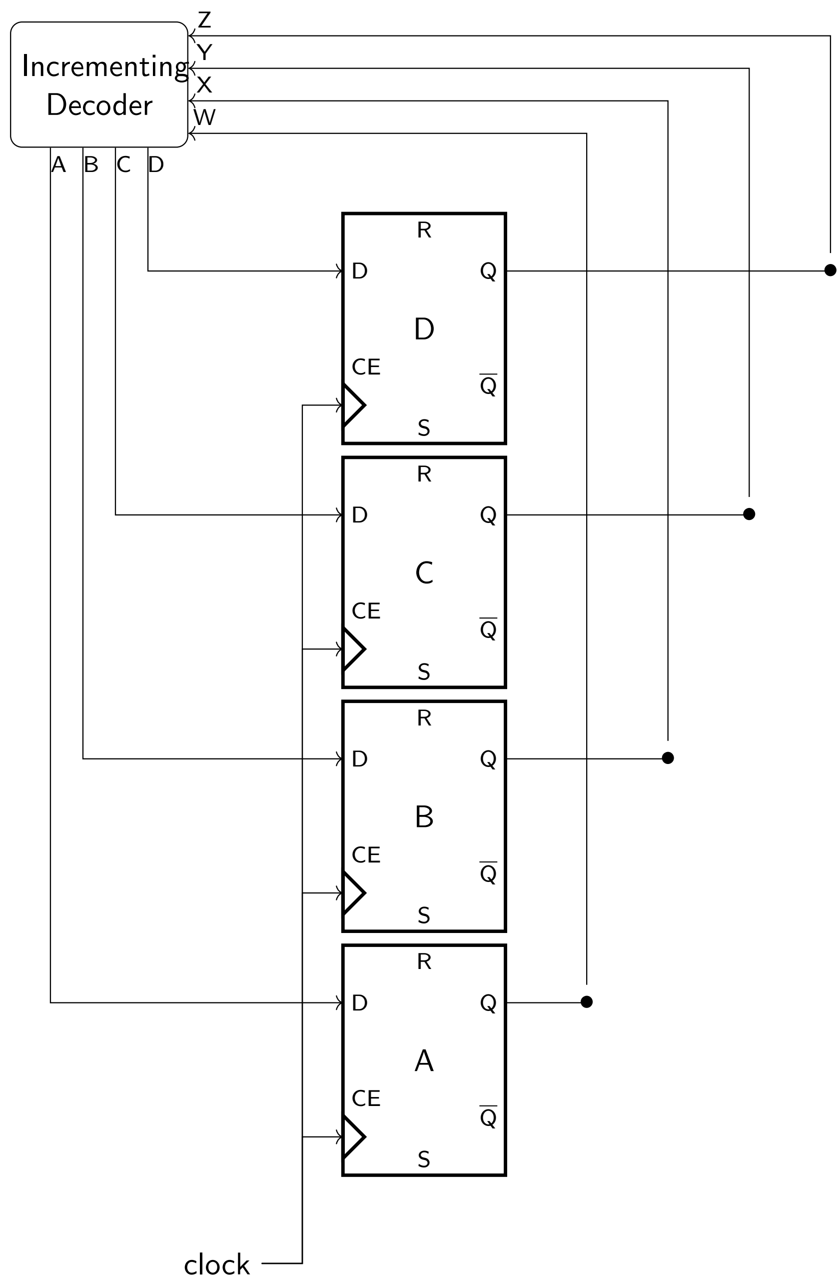


Figure 6.2: Decade counter FSM implementation using D-Flip Flops

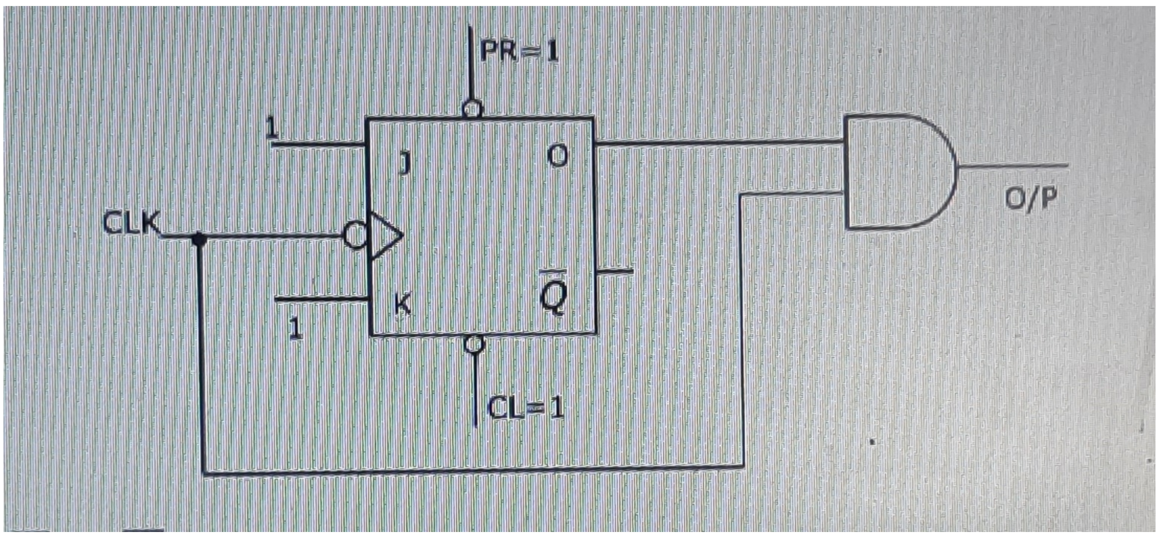


Figure 6.3:

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

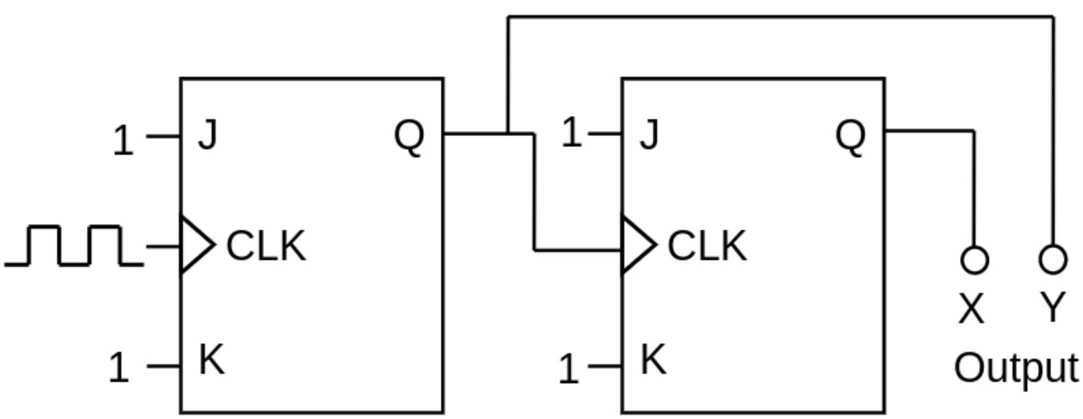


Figure 6.4:

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

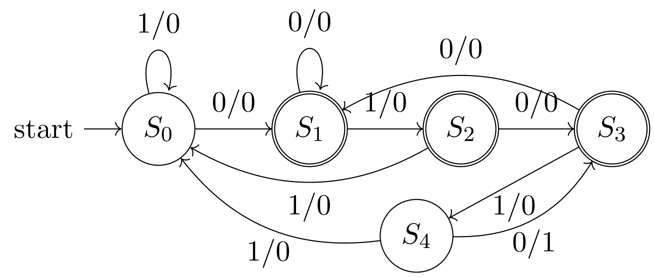


Figure 6.5: State diagram

(a) the sequence 01010 is detected

(b) the sequence 01011 is detected

(c) the sequence 01110 is detected

(d) the sequence 01001 is detected

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

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

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

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

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

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

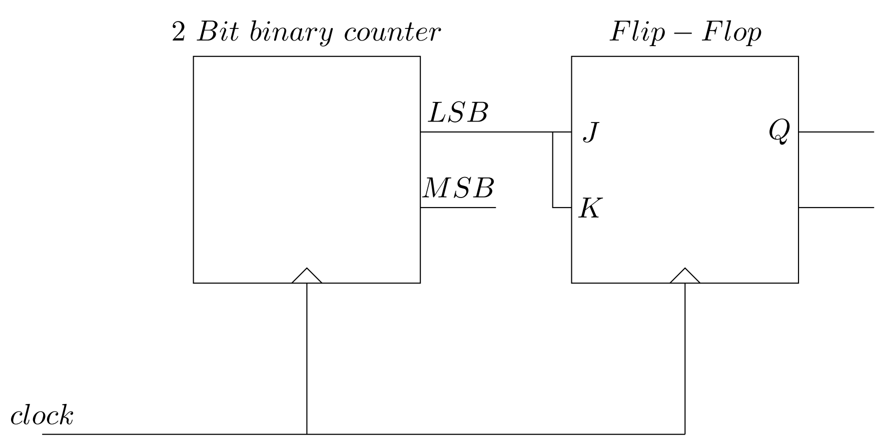


Figure 6.6:

(GATE EC-2022)

6. 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)

7. Two T-flip flops are interconnected as shown in Fig. 6.7. 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 3^{rd} clock cycle is ______

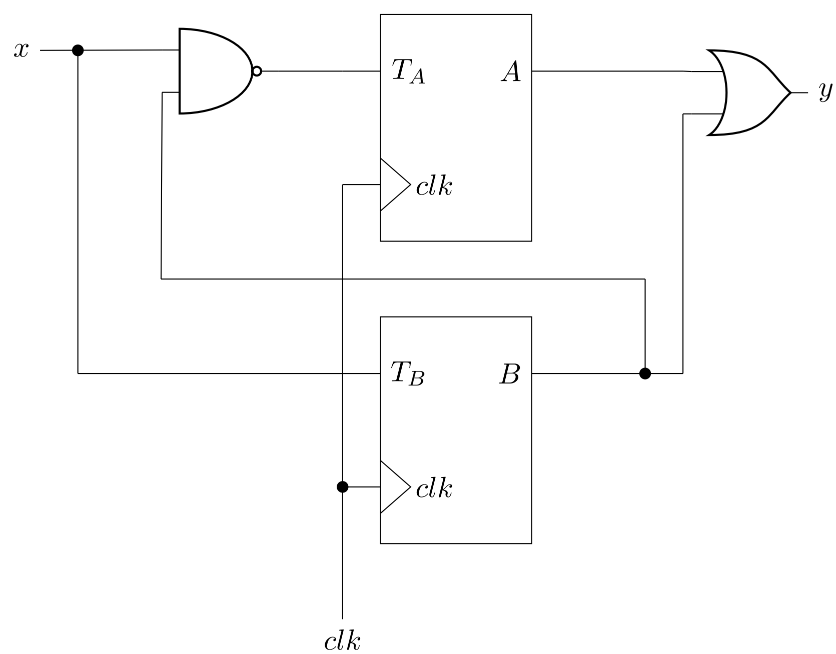


Figure 6.7:

(GATE IN 2020)

8. In the circuit shown below,6.8 a positive edge-triggered D flip-flop is used for sampling input data 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 $\Delta T/T_{ck}$ = 0.15,where the parameters ΔT and T_ck are shown in the figure.Assume that the Flip and the XOR gate are ideal. (GATE EC 2018)

9. A 2-bit synchronous counter using two J-K flip flops is shown. The expressions 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

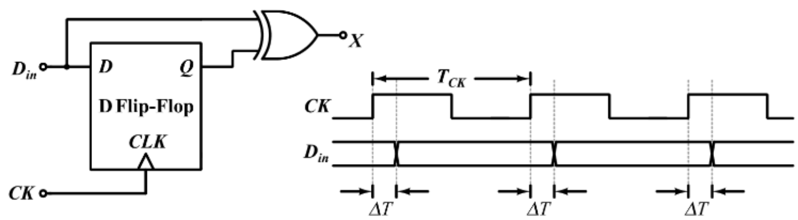


Figure 6.8: image

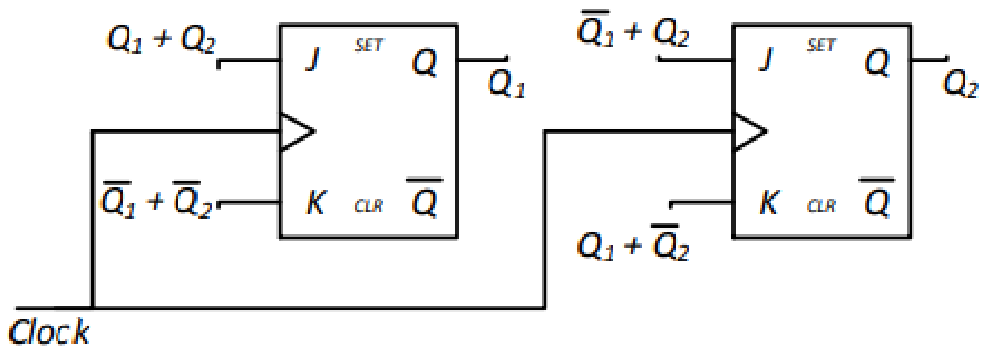


Figure 6.9: 2-bit synchronous counter

(a) $00 \rightarrow 11 \rightarrow 10 \rightarrow 01 \rightarrow 00...$

(b) $00 \rightarrow 01 \rightarrow 10 \rightarrow 11 \rightarrow 00...$

(c) $00 \rightarrow 01 \rightarrow 11 \rightarrow 10 \rightarrow 00...$

(d) $00 \rightarrow 10 \rightarrow 11 \rightarrow 01 \rightarrow 00...$

GATE IN 2018

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

logic.

$$S_1 : (\neg\, p \vee (p \wedge q)) \rightarrow q \tag{6.2}$$

$$S_2 : q \rightarrow (\neg\, p \vee (p \wedge q)) \tag{6.3}$$

Which one of the following choices is correct? (GATE-CS2021)

(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.

11. Consider a 3-bit counter, designed using T flip-flops, as shown below:

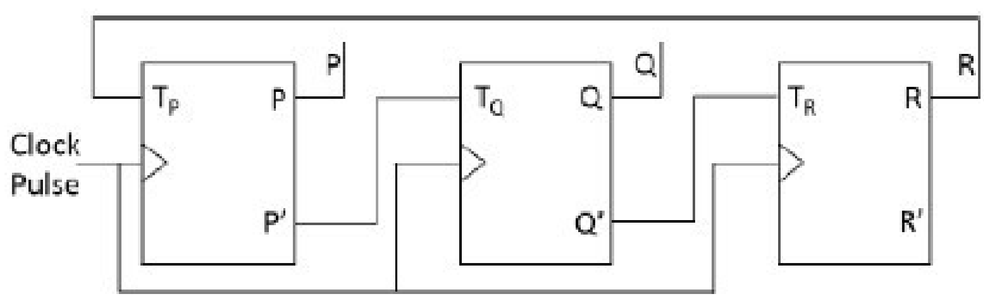


Figure 6.10:

Assuming the initial state of the counter given by PQR as 000,what are the next three states? (GATE-CS2021)

(A) 011, 101, 000

(B) 010, 101, 000

(C) 010, 101, 000

(D) $010, 101, 000$

12. The state diagram of a sequence detector is shown below. state S0 is the initial state of the sequence detector. If the output is 1,then (GATE-EC2020,39)

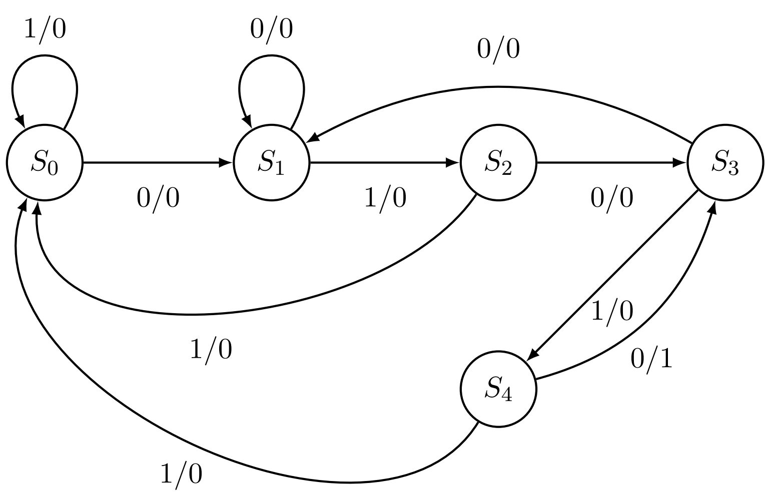


Figure 6.11: State diagram of a sequence detector

(a) the sequence 01010 is detected.

(b) the sequence 01011 is detected.

(c) the sequence 01110 is detected.

(d) the sequence 01001 is detected.

13. The state transition diagram for the circuit shown is (GATE-IN2019,39)

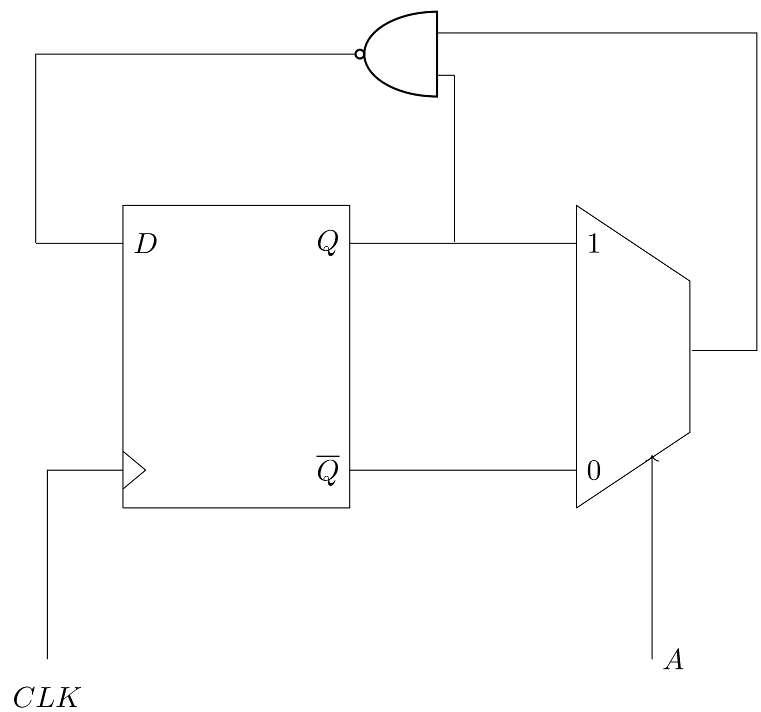
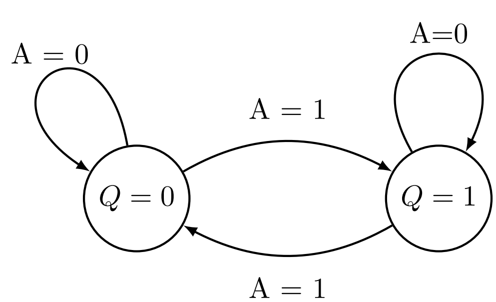
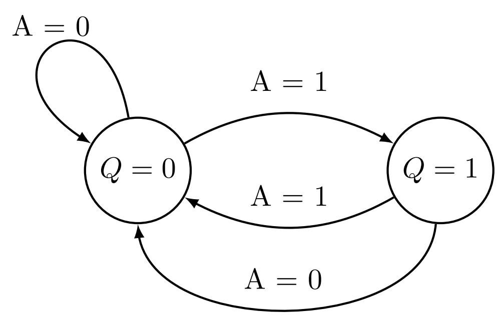


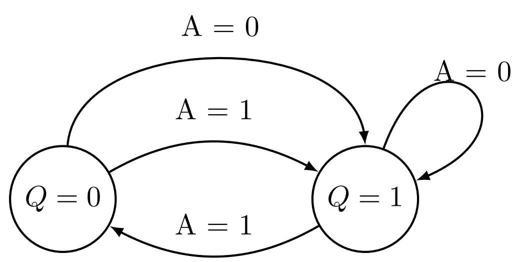
Figure 6.12: Circuit Diagram



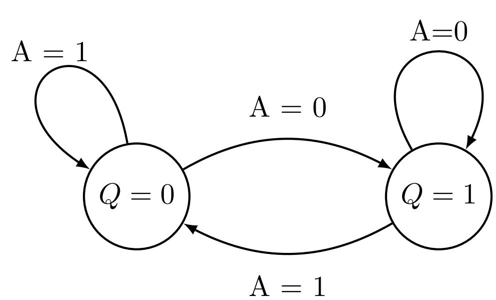
(a)



(b)



(c)



(d)

14. 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 ($base10$) form __________.

$(GATE\ \ EE2020-54)$

15. A sequence detector is designed to detect precisely 3 digital inputs, with overlapping sequences detectable. For the sequence $(1,0,1)$ and input data $(1,1,0,1,0,0,1,1,0,1,0,1,1,0)$. $(GATE\ \ EE2020-15)$

    (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) $(0,1,0,0,0,0,0,0,1,0,0,0)$

16. A 16-bit synchronous binary up-counter is clocked with the frequency $\mathbf{f}_{\text{CLK}}$. 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. The clock frequency__________MHz.
    (Round off to 2 decimal places.) $(GATE\ \ EE2021-22)$

17. which of the following is the correct binary equivalent of the hexadecimal F6C? $(GATE\ PH2020, 6)$

(a) 011011111100

(b) 111101101100

(c) 110001101111

(d) 011011000111

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

(GATE EC-2020,38)

(a) 110101

(b) 110010

(c) 111101

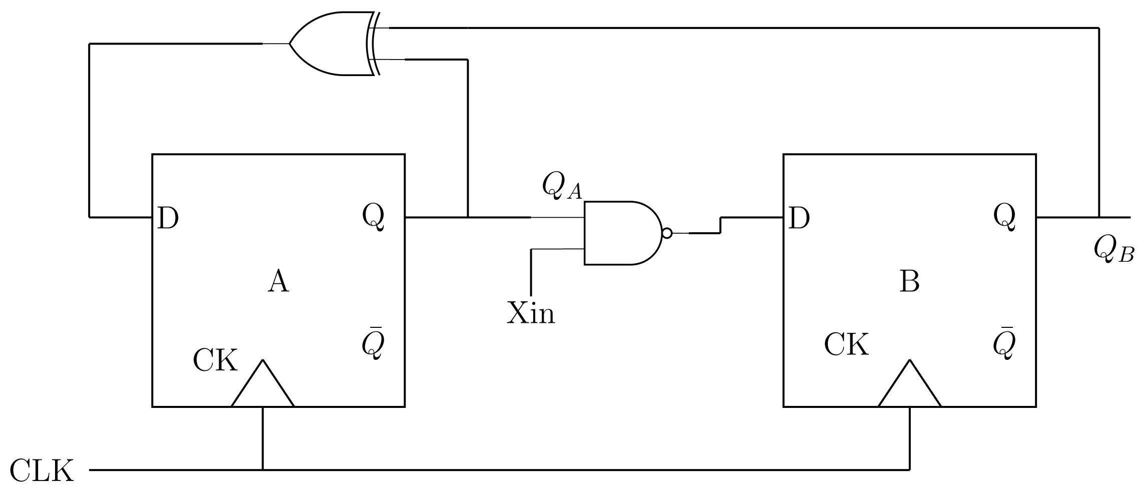
(d) 111001

19. A finite state machine (FSM) is implemented using the D flip-flops A and B, and logic gates, as shown in the figure 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

(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 $X_{IN} = 0$

(GATE EC 2017)

Q.6 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) $x \oplus y = x + y, if xy = 0$

(D) $x \oplus y = (xy + x'y')'$

Chapter 7

Assembly Programming

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

7.1. Software Installation

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

```
%Finding the port


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

2. Copy the .inc file to your home directory

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

3. Execute

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

as

4. Then flash the .hex file

```
hello.hex
```

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

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

```
ldi r17,0b00000000
```

   Save and execute. The led should turn off.

7. What do the following instructions do?

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

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

8. What do the following instructions do?

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

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

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

7.2. Seven Segment Display

1. See Table 2.1 for components.

2. Complete Table 2 for all the digital pins using Fig. 2.

Port Pin	Digital Pin
PD2	2
PB5	13

Table 2:

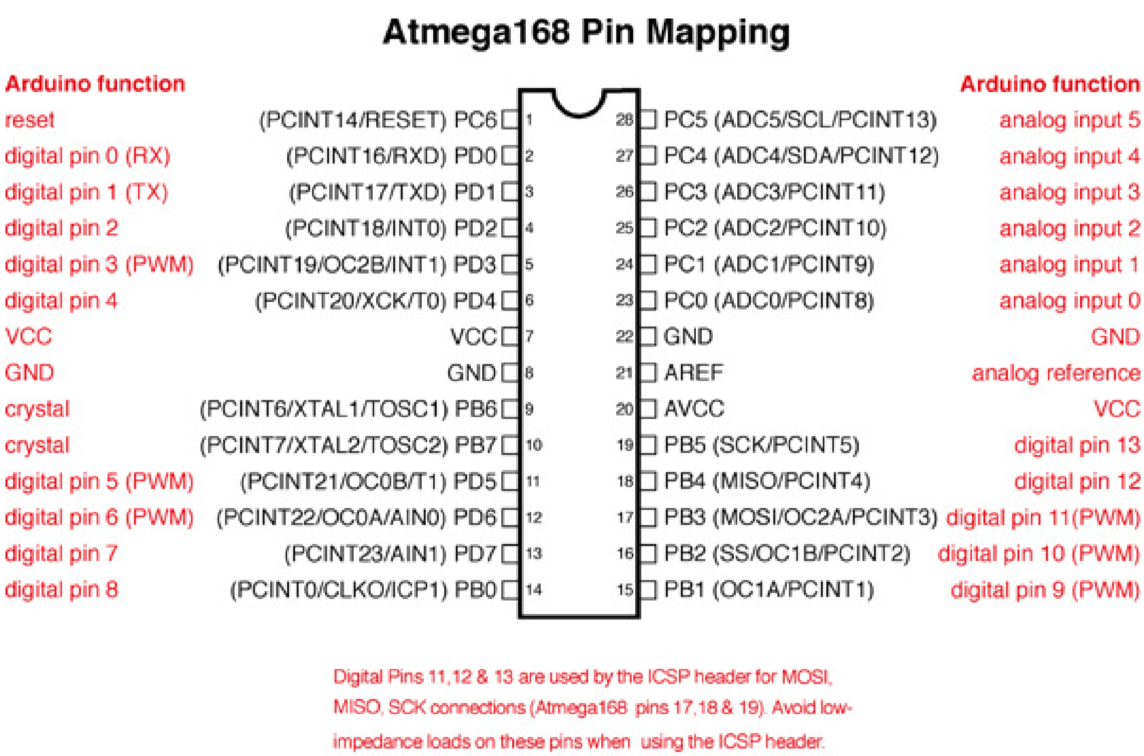


Figure 2:

3. Make connections according to Table 3.

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

Table 3:

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

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

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



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

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

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

7.3. 7447

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

7.3.1. Components

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

7.3.2. Boolean Operations

7.3.2.1. Verify the AND,OR and XOR operations in assembly using the following code and making pin connections according to Table 7.3.2.1.1.

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

7447	D	C	B	A
Arduino	5	4	3	2

Table 7.3.2.1.1: Arduino and 7447 IC connections

7.3.2.2. 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.

7.3.2.3. 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.

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

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

7.3.2.5. 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.

$$A = W' \tag{7.3.2.5.1}$$

$$B = WX'Z' + W'X \tag{7.3.2.5.2}$$

$$C = WXY' + X'Y + W'Y \tag{7.3.2.5.3}$$

$$D = WXY + W'Z \tag{7.3.2.5.4}$$

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

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

7.3.3. Controlling the Display

1. Connect the 7447 IC to the seven segment display.
2. Make connections between the 7447 and the arduino according to Table 7.3.2.1.1
3. Execute the following program. The number 5 will be displayed.

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

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

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

   assembly/7447/io/codes/ip_7447.asm

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

Table 5:

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

6. Explain the following instructions

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

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

7.4. 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) $R0$ and $R1$ 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 _____.

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

(EE GATE 2023)

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
```

3. 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)

7.5. Timer

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

7.5.1. Components

Component	Value	Quantity
Arduino	UNO	1

7.5.2. Blink through TIMER

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

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

2. Explain the following instruction

```
sbi DDRB, 5
```

3. What do the following instructions do?

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

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

$$TCCR0B = 0b101 \tag{3.1}$$
$$\implies CLK = \frac{SYSCLK}{1024} \tag{3.2}$$
$$= \frac{16M}{1K} = 16kHz. \tag{3.3}$$

4. Explain the PAUSE routine.

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

```
        AND r16, r17 ;need second bit
        BREQ PAUSE
        OUT TIFR0, r17 ;set tifr flag high
    dec r19
    brne lp2
    ret
```

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

5. Explain the lp2 routine.

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

6. What is the blinking delay?

   Solution: The blinking delay is given by

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

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

7.5.3. Blink through Cycle Delays

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

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

2. Explain how the delay is obained

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

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

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

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

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

3. Should you use timer delay or cycle delay?

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

7.6. Memory

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

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

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

2. Explain the following instructions

```
ldi xl,0x00
ldi xh,0x01
ldi r16,0b00000000
st x,r16
```

Solution: X=R27:R26, Y=R29:R28, and Z=R31:R30 where R27:R26 represents XH:XL. The above instructions load 0b00000000 into the memory location X=0x0100.

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

```
ldi r16,0b00000000
ldi r17,0x09
loop_cnt:
inc r16
```

```
inc xl
st x,r16
dec r17
brne loop_cnt
```

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

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

7.7. Problems

1. 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

   MVI B, 64H; 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.

   (GATE IN 2018)

2. 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 $f_s = 1\,\text{MHz}$).The Counter Will then display (GATE IN 2021)

(a) 0

(b) 10

(c) 100

(d) 1000

3. 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) $0\mathbf{x}42200000$ and $0\mathbf{x}C1200000$, respectively. If $\mathbf{R3} = \frac{\mathbf{R1}}{\mathbf{R2}}$, what is the value stored in **R3**? (GATE-EC2021,31)

(a) 0x40800000

(b) 0xC0800000

(c) 0x83400000

(d) 0xC8500000

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

$$PUSH\,\{R_2\}$$
$$PUSH\,\{R_2\}$$
$$PUSH\,\{R_3\}$$
$$POP\,\{R_1\}$$
$$POP\,\{R_2\}$$
$$POP\,\{R_3\}$$

After execution, the content of registers R_1, R_2, R_3 are (GATE-EC2021,32)

(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) $R_1 = 40H, R_2 = 25H, R_3 = 30H$

Chapter 8

Embedded C

8.1. Blink

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

8.1.1. Components

Component	Value	Quantity
Arduino	UNO	1

8.1.2. Blink

1. Execute the following

```
cd avr-gcc/setup/codes


make
```

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

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

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

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

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

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

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

8.2. Display Control

1. Connect the arduino to the seven segment display

2. Execute the following code

```
avr−gcc/sevenseg/codes/main.c
```

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

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

8.3. Input

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

2. Execute the following code

```
avr—gcc/input/codes/main.c
```

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

8.4. GCC-Assembly

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

8.4.1. Components

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

Table 3:

8.4.2. GCC with Assembly

1. Execute

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

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

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

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

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

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

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

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

8.5. LCD

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

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

Table 6:

8.5.1. Components

8.5.2. Display Number on LCD

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

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

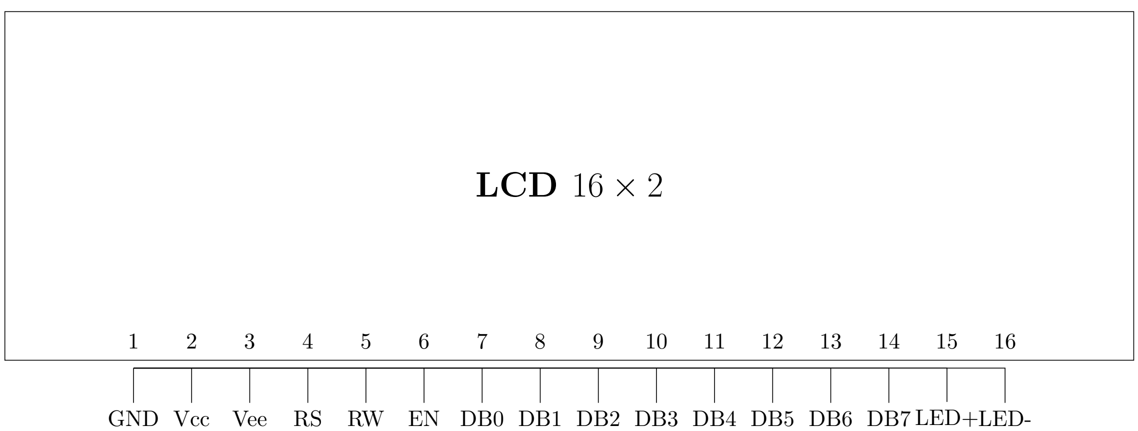


Figure 8.5.2.2.1: LCD

8.5.2.3. Execute

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

Table 8.5.2.2.1: Arduino to LCD Pin Connection.

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

8.5.2.4. Modify the above code to display a string.

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

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

8.5.2.7. Write assembly routines for driving the lcd.

8.6. 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)

Chapter 9

Vaman-ESP32

9.1. Software

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

```
vaman/esp32/codes
```

9.2. Flash Vaman-ESP32 using Arduino

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

The Vaman pin diagram is available in Fig. 9.2.1.2

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

Table 9.2.1.1:

9.2.2. For compiling and genrating the bin file

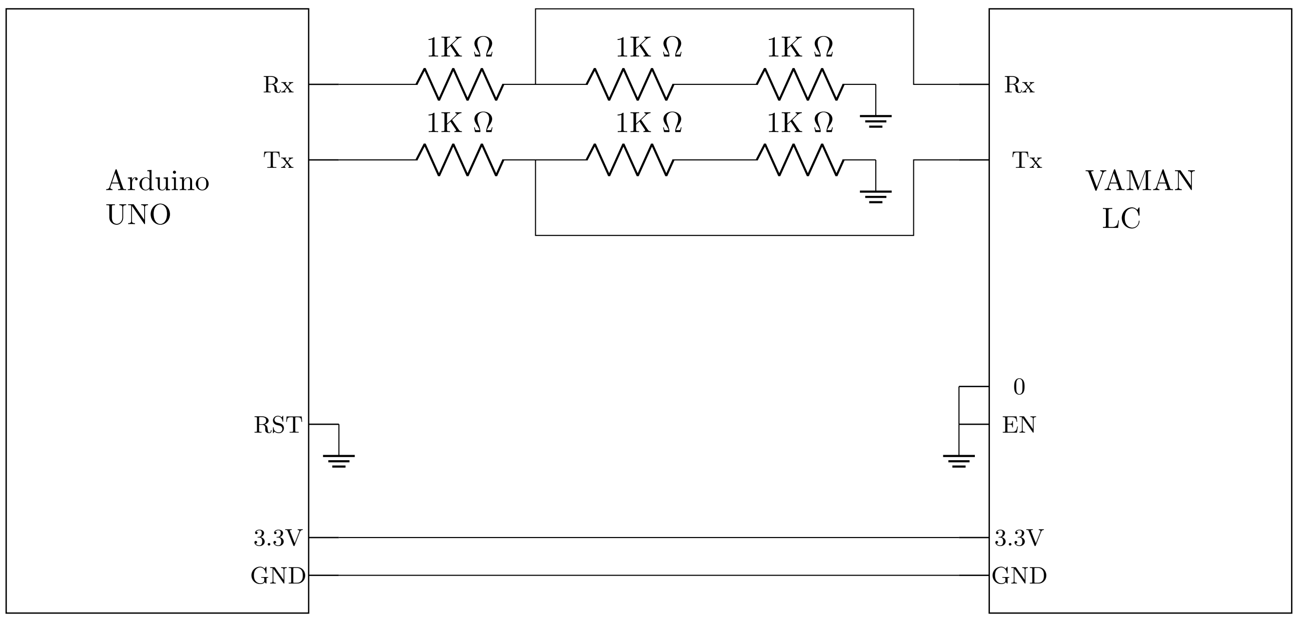


Figure 9.2.1.1: Circuit Connections

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

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

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

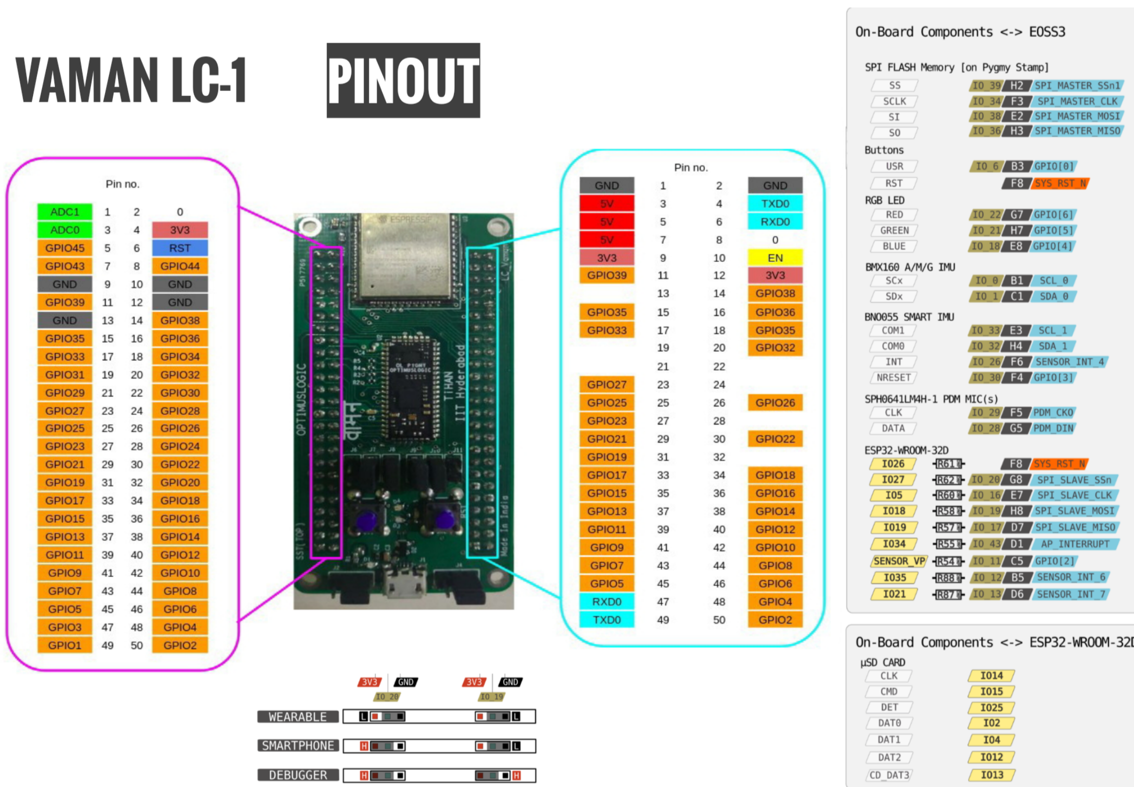


Figure 9.2.1.2: Pin diagram

9.2.4. For uploading bin file to Vaman through ArduinoDroid application

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

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

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

9.2.6. Open

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

and change the delay to

```
delay(2000);
```

and exectute the code by following the steps above.

9.3. OTA

9.3.1. Flash the following code through USB-UART.

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

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

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

in src/main.cpp file

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

```
ifconfig
nmap -sn 192.168.231.1/24
```

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

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

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

through

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

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

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

9.4. Onboard LED

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

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

Table 9.4.1.1:

9.4.2. Flash the following code OTA

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

You should see the onboard green LED blinking.

9.4.3. Change the blink duration to 100 ms.

9.5. LCD

T:hrough this manual, we learn how to measure an unknown resistance through ESP32 and display it on an LCD.

9.5.1. Components

Component	Value	Quantity
Resistor	220 Ohm	1
	1K	1
ESP32	Devkit V1	1
Jumper Wires		20
Bread board		1
LCD	16 X 2	1
Potentiometer	10K	1

Table 9.5.1:

9.5.2. Setting up the Display

9.5.1. Plug the LCD in Fig. 9.5.1.1 to the breadboard.

9.5.2. Connect the ESP32 pins to LCD pins as per Table 9.5.2.1. Make sure that all 5V sources are connected to the LCD through a 220 Ω resistance.

9.5.3. Execute the following code after editing the wifi credentials

```
vaman/esp32/lcd/codes/setup
```

You should see the following message

```
Hi
```

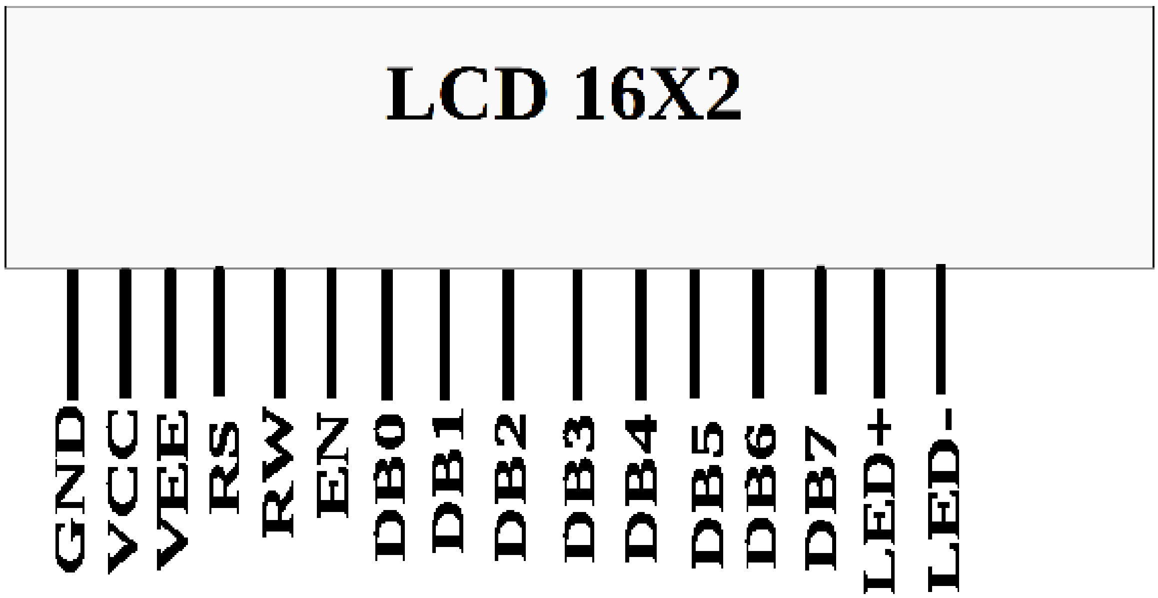


Figure 9.5.1.1: lcd

```
This is CSP Lab
```

9.5.4. Modify the above code to display your name.

9.5.3. Measuring the resistance

9.5.1. Connect the 5V pin of the ESP32 to an extreme pin of the Breadboard shown in Fig. 2.1. Let this pin be V_{cc}.

9.5.2. Connect the GND pin of the ESP32 to the opposite extreme pin of the Breadboard.

9.5.3. 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 V_{cc} and R_2 is connected to GND. Refer to Fig. 9.5.3.1

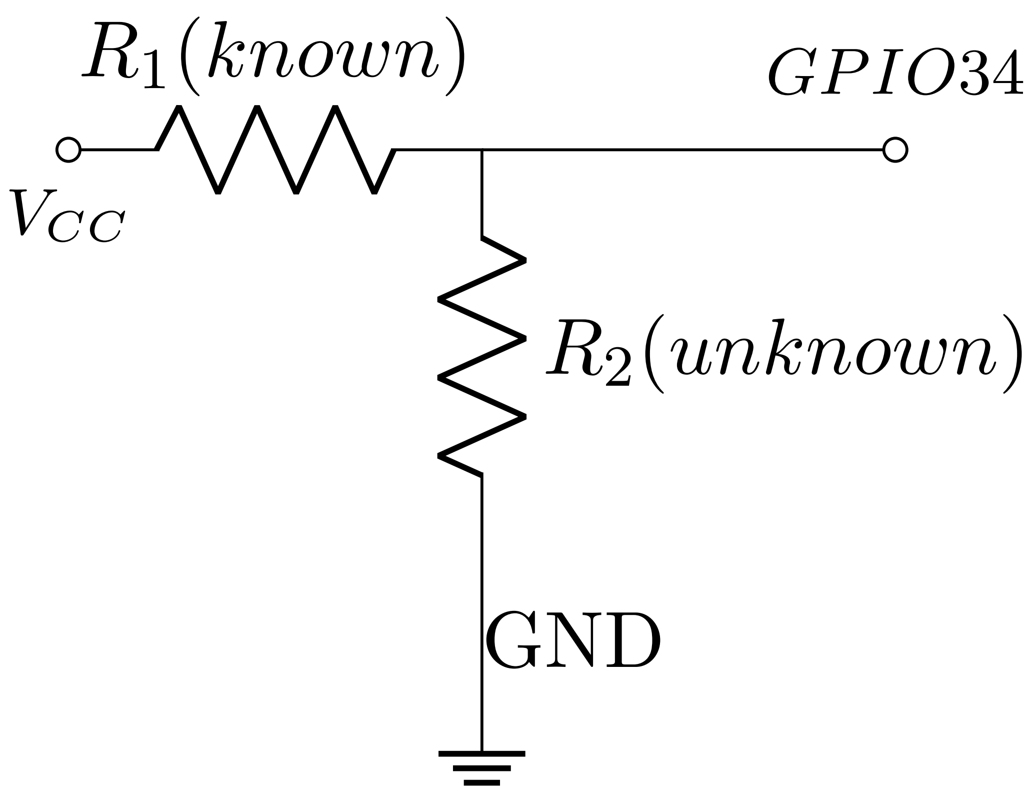


Figure 9.5.3.1: Voltage Divider

ESP32	**LCD Pins**	**LCD Pin Label**	**LCD Pin Description**
GND	1	GND	
5V	2	Vcc	
GND	3	Vee	Contrast
GPIO 19	4	RS	Register Select
GND	5	R/W	Read/Write
GPIO 23	6	EN	Enable
GPIO 18	11	DB4	Serial Connection
GPIO 17	12	DB5	Serial Connection
GPIO 16	13	DB6	Serial Connection
GPIO 15	14	DB7	Serial Connection
5V	15	LED+	Backlight
GND	16	LED-	Backlight

Table 9.5.2.1: Make sure that all 5V sources are connected to the LCD through a 220 Ω resistance.

9.5.4. Connect the junction between the two resistors to the GPIO34 pin on the ESP32.

9.5.5. Connect the ESP32 to the computer so that it is powered.

9.5.6. Execute the following code after editing the wifi credentials

```
vaman/esp32/lcd/codes/resistance
```

9.5.4. Explanation

9.5.1. 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 GPIO34.

9.5.2. The 12-bit ADC can differentiate 4096 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 4095. 0 would represent 0 volts falls across the unknown resistor. A value of 4095 would mean that practically all 5 volts falls across the unknown resistor.

9.5.3. V_{out} represents the divided voltage that falls across the unknown resistor.

9.5.4. The Ohm meter in this manual works on the principle of the voltage divider shown in Fig. 9.5.3.1.

$$V_{out} = \frac{R_1}{R_1 + R_2} V_{in} \tag{9.5.4.1}$$

$$\Rightarrow R_2 = R_1 \left(\frac{V_{in}}{V_{out}} - 1 \right) \tag{9.5.4.2}$$

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

9.5.5. Repeat the exercise with another unknown resistance.

9.6. Problems

1. Write programme to add 8 bit data i.e. B1H and 21H and print the result in decimal in LCD. (GATE EE 2020)

Chapter 10

Vaman-FPGA

10.1. Setup

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

10.1.1. Software

The codes are available at

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

10.1.2. Setup

10.1.2.1. Follow the instructions in

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

10.1.3. Frequency

10.1.3.1. In the following verilog program,

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

pay attention to the following lines

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

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

10.1.3.2. In instruction 10.1.3.1, replace

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

with

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

and execute the verilog code.

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

10.1.3.4. Find the bit length of 20 MHz.

Type	Vaman Pin	Connection
Input	IO_28	GND
Output	IO_11	LED

Table 10.1.3.8.1: Vaman Input/Output.

Solution:

$$\log_2(20000000) \approx 27 \tag{10.1.3.4.1}$$

10.1.3.5. Obtain the above answer using a Python code.

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

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

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

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

with

```
assign blueled = led;
```

and execute the code.

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

Solution: Execute the following code

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

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

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

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

10.2. Seven Segment Display

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

10.2.1. Software

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

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

10.2.2. Setup

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

10.2.2.2. Now execute the following code.

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

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

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

Table 10.2.2.1.1: Seven segment display - Vaman connection.

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

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

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

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

10.2.3. Examples

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

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

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

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

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

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

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

Solution: Execute the following code.

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

10.2.3.4. Use the Vaman as a decade counter.

Solution: Execute the following code.

codes/loop/decade_counter.v

10.3. Boolean Logic

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

10.3.1. Software

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

vaman/fpga/boolean/codes

10.3.2. Setup

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

10.3.2.2. Implement Table 4.1 using the Vaman and the display.

Solution: In Table 4.1, the output variables a, b, c, d, e, f, g can be expressed in terms

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

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

Table 10.3.2.1.1: Pin connections between Vaman and the display.

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

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

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

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

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

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

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

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

Execute the following program.

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

Connect W, X, Y, Z to GND. For different values of the input variables, verify the output in on the display using Table 4.1.

10.3.2.3. Table 3.5 describes the properties of the incrementing decoder. Using Boolean logic, express A, B, C, D in terms of W, X, Y, Z. Subsequently, implement this decoder by implementing the the expressions so obtained in the Vaman using verilog.

Solution: The following equations contain the desired expressions.

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

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

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

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

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

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

10.3.3. Decade Counter

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

10.3.3.2. Design and implement the down counter.

10.4. LCD

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

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

Table 10.4.1.2.1: Pin connections between Vaman and the display.

10.4.1. Display the addition of two numbers on LCD

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

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

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

10.4.1.4. Now execute the following code.

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

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

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

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

10.4.1.8. Write verilog code for different arithmetic operations.

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

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

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

Figure 10.2.3.2.1: Pin Definitions

Chapter 11

Vaman-ARM

11.1. Setup

11.1.1. Software

All codes used in this document are available at

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

11.1.2. Setup

11.1.2.1. Follow the instructions at

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

11.1.3. Delay

11.1.3.1. See the following lines of the code below

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

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

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

11.1.3.2. Replace the following line in 11.1.3.1

```
    HAL_DelayUSec(2000000);
```

with

```
    HAL_DelayUSec(1000000);
```

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

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

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

Now obtain blue colour.

11.1.3.4. Now obtain green colour without blink.

Solution: Execute the following code.

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

Type	Pin	Destination
Input	IO_5	GND

Table 11.1.3.5.1: Vaman control through external input.

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

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

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

11.2. Seven Segment Display

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

11.2.1. Software

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

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

11.2.2. Setup

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

11.2.2.2. Now execute the following code

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

Table 11.2.2.1.1: Display-Vaman connection.

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

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

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

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

11.2.2.3. Modify the above program using Table 10.2.2.3.1and Fig. 2.3 to display 0-9.

11.2.3. Examples

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

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

Solution: Execute the following file.

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

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

Solution: Execute the following code.

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

11.3. FSM

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

11.3.1. Software

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

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

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

11.3.2. Setup

11.3.2.1. Execute Table 4.1 using the Vaman and a seven segment display.

Solution: The outputs a, b, c, d, e, f, g in Table 4.1 are expressed in terms of the inputs A, B, C, D through the following equations.

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

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

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

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

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

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

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

Now execute the following code.

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

For different values of A, B, C, D, execute the above code to verify Table 4.1.

11.3.2.2. Table 3.5 shows the logic for the incrementing decoder. Express A, B, C, D in terms of W, X, Y, Z.

Solution: The desired expressions are available below.

$$A = W'X'Y'Z' + W'XY'Z' + W'X'YZ' + W'XYZ' + W'X'Y'Z \tag{11.3.2.2.1}$$

$$B = WX'Y'Z' + W'XY'Z' + WX'YZ' + W'XYZ' \tag{11.3.2.2.2}$$

$$C = WXY'Z' + W'X'YZ' + WX'YZ' + W'XYZ' \tag{11.3.2.2.3}$$

$$D = WXYZ' + W'X'Y'Z \tag{11.3.2.2.4}$$

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

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

11.3.2.3. Fig. 9.2.1.2 shows the pin diagram for the Vaman. Connect the pins in bank J5 to the seven segment display using Fig. 2.2 and Table 10.3.2.1.1. Do not forget to put a resistor between COM and 3.3V. Then execute the following code

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

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

11.3.3. Decade Counter

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

Chapter 12

STM-32

12.1. Setup

T:his manual shows how to program an STM32 board using USB-UART.

12.1.1. Software

12.1.1.1. Make the connections as per the Table 12.1.1.1.1

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

Table 12.1.1.1.1: STM32 to USB-UART connections

12.1.1.2. Basic onchip led blink program is available at

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

12.1.1.3. Make sure that platformio.ini file has these below lines

```
[env:bluepill_f103c8]
platform = ststm32
framework = arduino
board = bluepill_f103c8
platform_packages=platformio/toolchain-gccarmnoneeabi@^1.90201.0
```

12.1.1.4. Now, complile the blink program

```
cd codes
pio run
```

12.1.1.5. By this .bin file will be genrated and all the necessary packages will be installed

12.1.1.6. Before flashing, make sure that STM32 board is in Programming mode

12.1.1.7. To make the stm32 board into Programming mode, by setting the alligners to programming mode as shown in Fig 12.1.1.7.1. This configuration will ensure that the microcontroller enters the system memory programmer on reset.

12.1.1.8. After executing

```
Install STM-Utils from playstore
Give the necessary permissions
Now connect the USB-UART to OTG
Automatically the the USB will be detected
Now upload the .bin file to the STM32 board
```

12.1.1.9. By these above mentioned steps the blink code will be flashed to the STM32 board

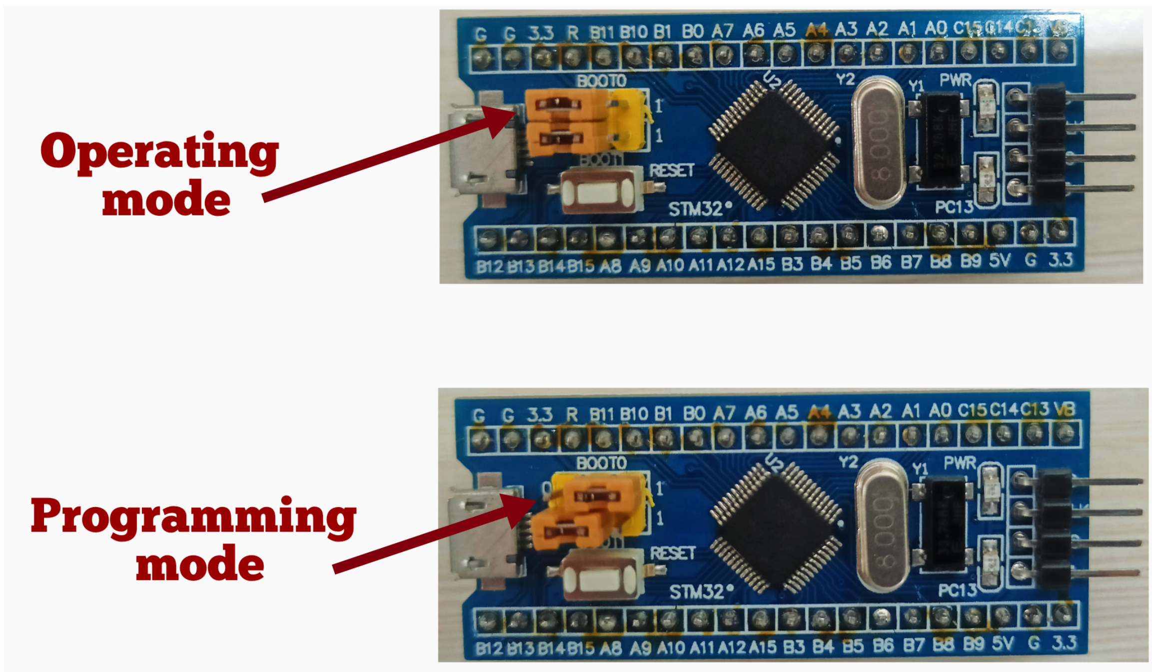


Figure 12.1.1.7.1: STM32 Operating and Programming mode

12.1.1.10. The inbuilt LED present on the stm32f103c8 will start blinking.

12.1.1.11. After flashing, the inbuilt LED present on the stm32f103c8 will start blinking and then change the STM32 board to operating mode as shown in Fig 12.1.1.7.1 inorder run your code from the main flash memory.

12.2. Seven Segment Display

T:he objective of this manual is to introduce beginners to arm embedded programming by powering a seven segment display.

12.2.1. Components

12.2.1.1. The necessary components for this manual are listed in Table 12.2.1.1.1.

Component	**Value**	**Quantity**
Breadboard		1
Resistor	220 Ω	1
STM32F103C8T6		1
Seven Segment Display	Common Anode	1
Jumper Wires		20

Table 12.2.1.1.1: Components

12.2.2. Hardware

12.2.2.2. 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 · LED.

12.2.2.3. Connect one end of the 1K resistor to the COM pin of the display and the other end to an extreme pin of the breadboard.

12.2.2.4. The STM32F103C8T6 micro-controller in Fig. 12.2.2.5.1 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.

12.2.2.5. Make the pin connections in Table 12.2.2.5.2 using Figs. 2.2 and 12.2.2.5.1.

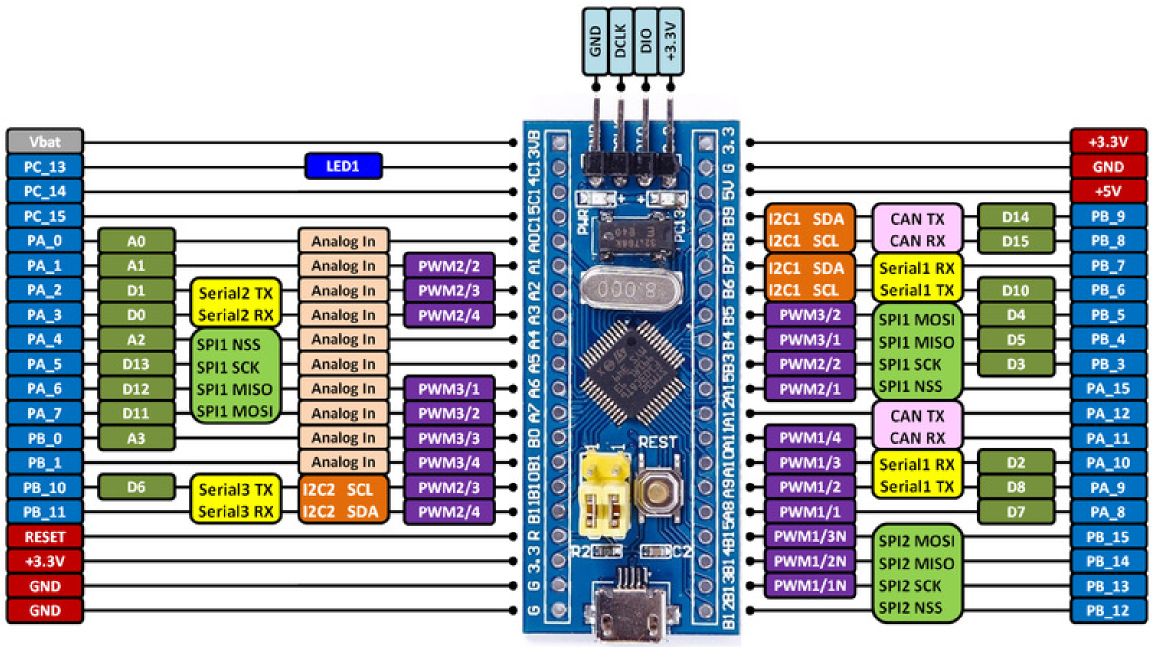


Figure 12.2.2.5.1: Pin Connections of STM-32

12.2.3. Software

12.2.3.6. Before flashing code to STM32 set the STM32 board to Programming mode as shown in Fig. 12.1.1.7.1, then change the mode to Operating mode as shown in Fig. 12.1.1.7.1 after flashing the code inorder to run your program from main memory.

12.2.3.7. Execute the following code

```
cd stm32/sevenseg/codes/sevenseg_example.c
```

12.2.3.8. Explain the process of generating the number 0 using the following instruction.

```
GPIOB−>ODR = 0xFC08;
```

Solution: ODR is the Output Data Register, which is used to write outputs to the

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	
8	0	0	0	0	0	0	0	
9	0	0	0	0	1	0	0	

Table 12.2.2.5.2: STM-32 and Seven Segment Display Connections

GPIO pins. The 16 bit number 0xFC08 on the RHS represents the pin configuration for the pins of port B of STM32F103C8T6, which are numbered PB15-PB0 in that order. See Table 12.2.2.5.2.

12.2.3.9. Repeat the above exercise to generate the numbers 1-9 on the display.

12.2.3.10. 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)
```

12.2.3.11. 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.

```
cd codes/bin2dec_example.c
```

12.3. Timers

T:his manual shows how to program timers in arm using STM32F103C8T6.

12.3.1. Components

12.3.1.1. The necessary components for this manual are listed in Table 12.3.1.1.1.

Component	**Value**	**Quantity**
Breadboard		1
Resistor	220 Ω	1
STM32F103C8T6		1
Seven Segment Display	Common Anode	1
Jumper Wires		20

Table 12.3.1.1.1: Components

12.3.1.2. List all the available timers in the STM32F103C8T6 blue pill.

Solution: See Table 12.3.1.2.1

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 12.3.1.2.1: STM32F103C8T6 Timer Types.

12.3.2. Systick timer

12.3.2.3. The Systick timer is the default timer available on all ARM chips.

12.3.2.4. Make connections as shown in Table 12.3.2.4.1.

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

Table 12.3.2.4.1: Pin Connections

12.3.2.5. Before flashing code to STM32 set the STM32 board to Programming mode as shown in Fig. 12.1.1.7.1, then change the mode to Operating mode as shown in Fig. 12.1.1.7.1 after flashing the code inorder to run your program from main memory.

12.3.2.6. Execute the program in

```
cd timers/blink_systick.c
```

12.3.2.7. 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

$$T = \frac{1}{8}\mu s = 1 \text{ cycle} \tag{12.3.2.7.1}$$

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

$$1 \text{ second} = 8000000 \text{ cycles} \tag{12.3.2.7.2}$$

12.3.2.8. List the SysTick registers.

Solution: See Table 12.3.2.8.1.

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 12.3.2.8.1: Systick Registers

12.3.2.9. What do the following instructions do?

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

Solution: See Table 12.3.2.8.1 for details. These two instructions ask the SysTick timer to count down from 4000000 to 0.

12.3.2.10. Explain the following instruction.

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

Solution: Fig. 12.3.2.10.1 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.

12.3.2.11. What does the following instruciton do?

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															COUNT FLAG
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													CLKSO URCE	TICK INT	EN ABLE
													rw	rw	rw

Figure 12.3.2.10.1: Control Register (CTRL)

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

Solution: From Fig. 12.3.2.10.1, ENABLE = 1 enables the counter (for delay) and CLKSOURCE = 1 enables the 8 MHz internal RC clock.

12.3.2.12. Obtain a 1 MHz clock.

Solution: CLKSOURCE = 1 results in the $\frac{\text{Processor Clock}}{8} = 1$ MHz clock.

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

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

12.3.3. TIMER-1

12.3.3.14. Make the connections according to Table 12.3.2.4.1. Execute the following program

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

12.3.3.15. Enable Timer1 through RCC.

Solution:

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

12.3.3.16. Select the HSI clock of 8 MHz as TIM1 clock.

Solution: See Fig. 12.3.3.16.1 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

Figure 12.3.3.16.1: Slave Mode Control Register (SMCR)

12.3.3.17. What does the following instruction do?

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

Solution: Fig. 12.3.3.17.1 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

Figure 12.3.3.17.1: Control Register 1 (CR1)

12.3.3.18. Make TIM1 clock = 2 KHz.

Solution: Through the following command,

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

$$TIM1\_CLK = \frac{HSI\_CLK}{TIM1->PSC+1} = \frac{8000000}{4000} \tag{12.3.3.18.1}$$

Fig. 12.3.3.18.1 shows the TIM1− >PSC (prescalar) register.

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

Figure 12.3.3.18.1: Prescalar (PSC)

12.3.3.19. What is the maximum value that can be stored in TIM1− >PSC?

12.3.3.20. Make TIM1 count 1000 cycles of the 2 KHz TIM1 clock.

Solution:

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

12.3.3.21. Like the PSC, the ARR (auto reload register) is also of length 16 bits and used for factoring the clock.

12.3.3.22. 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

$$\frac{HSI\_CLK}{(TIM1->PSC+1)\,(TIM1->ARR+1)} = \frac{8000000}{4000 \times 1000} = 2 \text{ Hz} \quad (12.3.3.22.1)$$

12.3.3.23. 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

$$\frac{HSI\_CLK}{(TIM1->PSC+1)\,(TIM1->ARR+1)\,(TIM1->RCR+1)} = \frac{8000000}{4000 \times 1000} = 2 \text{ Hz} \quad (12.3.3.23.1)$$

TIM1− >RCR keeps track of the number of times the counter has overflowed. Fig. 12.3.3.23.1 shows the repetition counter register.

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

Figure 12.3.3.23.1: Repition Counter (RCR)

12.3.3.24. What is the maximum value of TIM1− >RCR?

12.3.3.25. What is the function of TIM1− >SR?

Solution: The status register (SR) is shown in Fig. 12.3.3.25.1. 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			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

Figure 12.3.3.25.1: Staus Register (SR)

12.3.4. TIMER-2

12.3.4.26. Blink an LED with TIM2.

Solution:

```
cd timers/timer2_blink.c
```

12.3.4.27. 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 12.3.3.14. 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.

12.3.4.28. 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.

12.3.5. Master-Slave Configuration

12.3.5.1. Blink

12.3.5.29. Execute the following program.

```
cd timers/master1_slave2_blink.c
```

12.3.5.30. 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. 12.3.5.30.1. The TS and SMS bits are visible in the SMCR register in Fig. 12.3.3.16.1.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TI1S	MMS[2:0]			CCDS	Reserved		
								rw	rw	rw	rw	rw			

Figure 12.3.5.30.1: Control Register 2 (CR2)

12.3.5.2. Decade Counter

12.3.5.31. Execute the following program.

```
cd timers/decade_counter.c
```

12.3.5.32. 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.

12.4. Clocks

T:his manual shows how to manage clocks in arm using STM32F103C8T6.

12.4.1. Components

12.4.1.1. The necessary components for this manual are listed in Table 12.4.1.1.1.

Component	**Value**	**Quantity**
Breadboard		1
Resistor	220 Ω	1
STM32F103C8T6		1
Seven Segment Display	Common Anode	1
Jumper Wires		20

Table 12.4.1.1.1: Components

12.4.1.2. List all available clocks in the STM32F103C8T6 blue pill.

Solution: See Table 12.4.1.2.1.

12.4.1.3. Make connections as shown in Table 12.4.1.3.1.

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

Table 12.4.1.2.1: STM32F103C8T6 Clock Types

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

Table 12.4.1.3.1: Pin Connections

12.4.2. HSE

12.4.2.4. Before flashing code to STM32 set the STM32 board to Programming mode as shown in Fig. 12.1.1.7.1, then change the mode to Operating mode as shown in Fig. 12.1.1.7.1 after flashing the code inorder to run your program from main memory.

12.4.2.5. Execute the following program

```
cd /clocks/hse_systick_blink.c
```

12.4.2.6. Explain the following instruction

```
RCC−>CR =0x00010000;
```

Solution: Fig. 12.4.2.6.1 shows the RCC− >CR register. The above instruction enables the HSE crystal, which is 8 MHz for the STM32F103C8T6.

12.4.2.7. Explain the following instruction

```
RCC−>CFGR =0x00000001;
```

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

Figure 12.4.2.6.1: RCC Clock Control Register (RCC− >CR)

Solution: Fig. 12.4.2.7.1 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

Figure 12.4.2.7.1: RCC clock Configuration Register (RCC− >CFGR)

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

12.4.3. PLL

12.4.3.9. Make the PLL as the system clock.

Solution:

```
RCC−>CFGR =0x00000010;
```

12.4.3.10. Choose the PLL input as HSE.

```
RCC−>CFGR =0x00010010;
```

12.4.3.11. Enable PLL

Solution:

```
RCC−>CR =0x01010000;
```

12.4.3.12. Execute the following code.

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

12.4.3.13. Make the PLL output = 24 MHz.

12.5. GPIO

T:his manual shows how to use the GPIO pins on the STM32F103C8T6 to control an LED.

12.5.1. Components

12.5.1.1. The necessary components for this manual are listed in Table 12.5.1.1.1.

12.5.2. Hardware Setup

12.5.2.2. Connect the USB-UART to STM32. Fig. 12.2.2.5.1 shows the blue pill board. The hardware connections between the USB-UART, STM32 and LED are available in Table 12.5.2.2.1. Make sure that the LED is connected to PB4 through the resistor.

Component	Quantity
STM32F103C8T6	1
USB-UART	1
Female-Female Jumper Wires	4
Female-Male Jumper Wires	4
Resistor ($> 220\Omega$)	1
LED	1

Table 12.5.1.1.1: Components

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

Table 12.5.2.2.1: USB-UART to STM32 connections

12.5.3. GPIO Output

12.5.3.3. Before flashing code to STM32 set the STM32 board to Programming mode as shown in Fig. 12.1.1.7.1, then change the mode to Operating mode as shown in Fig. 12.1.1.7.1 after flashing the code inorder to run your program from main memory.

12.5.3.4. Execute

```
cd /gpio/gpio_example.c
```

12.5.3.5. Modify the above program to turn the LED off.

12.5.3.6. 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.

12.5.3.7. 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 12.5.3.7.2.

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

Table 12.5.3.7.2: GPIO Enable pins

12.5.3.8. Explain the significance of the number (nibble) 0x3 corresponding to PB4 in Table 12.5.3.7.2.

Solution: The nibble 0x3 = 0b0011. From Table 12.5.3.8.1, 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.

12.5.3.9. 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

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 12.5.3.8.1: GPIO Output configuration

atomically clear pin values to VDD. The above command clears PB4.

12.5.3.10. 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.

12.5.3.11. Modify your program to control an LED using PA2.

12.5.4. GPIO Input

12.5.4.12. Connect PB7 to GND and execute the following code

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

12.5.4.13. What does the following instruction do?

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

Solution: This instruction declares PB7 as an input pin. 0x8=0b1000. According to Table 12.5.4.13.1, 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 12.5.4.13.1: GPIO Input configuration

12.5.4.14. 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.

12.6. LCD

T:his manual shows how to interface the 16×2 HD44780-controlled LCD using STM32F103C8T6.

12.6.1. Components

12.6.1.1. The necessary components for this manual are listed in Table 12.6.1.1.1.

Component	Value	Quantity
Breadboard		1
STM32F103C8T6		1
LCD	16 × 2 HD44780	1
Jumper Wires	F-F	20
Jumper Wires	M-F	20

Table 12.6.1.1.1: Components

12.6.2. Hardware

12.6.2.2. Make connections as shown in Table 12.6.2.2.2.

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 12.6.2.2.2: Pin Connections

12.6.3. Software

12.6.3.3. Before flashing code to STM32 set the STM32 board to Programming mode as shown in Fig. 12.1.1.7.1, then change the mode to Operating mode as shown in Fig. 12.1.1.7.1 after flashing the code inorder to run your program from main memory. Execute the following program

```
cd /lcd/lcd_example.c
```

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

12.6.3.4. Write the ASCII code for the 0 character.

Solution: The code for 0 is **48 =0b00110000 = 0x30**.

12.6.3.5. 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
}
```

12.6.3.6. 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

$$GPIOA- > BSRR = (data << 2)\&0b00111100$$
$$= 0b00001100. \quad (12.6.3.6.1)$$

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

$$GPIOA- > BRR = \; (data << 2)\&0b00111100 \quad (12.6.3.6.2)$$

12.6.3.7. ensures that 00 is written to the pins A4-A5. PulseEnableLine() provides a clock pulse used to write the nibble 0011 to the LCD.

12.6.3.8. 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.

12.6.3.9. 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.

12.6.4. Project

12.6.4.10. Develop an arithmetic calculator using the STM32 along with the LCD.

12.7. ADC

T:his manual shows how to interface the 16×2 HD44780-controlled LCD using STM32F103C8T6.

12.7.1. Components

Component	**Value**	**Quantity**
Resistor	220 Ohm	1
	1K	1
Breadboard		1
STM32F103C8T6		1
LCD	16 x 2 HD44780	1
Jumper Wires	M-F	20
	F-F	5
	M-M	5

Table 10.1: Components

12.7.2. Internal Temperature Sensor

12.7.2.1. Make connections as shown in Table 12.6.2.2.2.

12.7.2.2. Make sure to set the STM32 board to Programming mode as shown in Fig. 12.1.1.7.1 then after flashing the code set the STM32 board to Operating mode as shown in Fig. 12.1.1.7.1 to run your program from main memory.

12.7.2.3. Execute the following program

```
cd /adc/internal_temp.c
```

12.7.2.4. What do you observe?

Solution: You should observe a number between 1750-1760. This is the output of the internal temperature sensor, captured in $ADC1->DR$.

12.7.2.5. Find an expression for V_{SENSE}

Solution:

$$V_{SENSE} = 3.3 \times \frac{ADC1->DR}{4095} \tag{12.7.2.5.1}$$

12.7.2.6. Obtain the formula for finding the temperature of the STM32 and list the values of the various parameters.

Solution: The desired formula is

$$T = (V_{25} - V_{SENSE})/AvgSlope + 25 \tag{12.7.2.6.1}$$

where the typical values of the above parameters are

V_{25}	AvgSlope
1.43	4.3

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

Figure 12.7.2.8.1: SMPR1

12.7.2.7. 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.

12.7.2.8. 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. 12.7.2.8.1. 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.

12.7.2.9. What is the sampling time?

Solution: Since the sample time is 239.5 cycles and the ADC frequency is 14 MHz,

$$T_s = 239.5 \times \frac{1}{14}\mu s = 17.1\mu s \qquad (12.7.2.9.1)$$

12.7.2.10. Explain the following instruction.

```
ADC1−>SQR3 |= ADC_SQR3_SQ1_4;
```

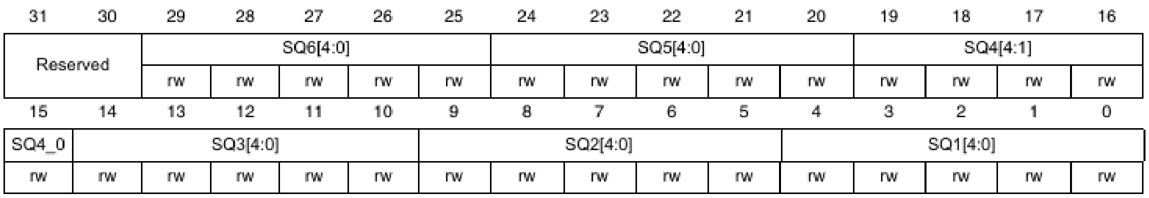


Figure 12.7.2.10.1: SQR3

Solution: ADC_SQR3_SQ1_4 = 0x00000010. This implies that SQ1=0b10000 in the ADC regular sequence register 3 ($ADC1->SQR3$)shown in Fig. 12.7.2.10.1. Since 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.

12.7.3. Measuring an Unkown Resistance

12.7.3.11. Configure SQR3 so that the 9th channel for ADC1 is 2nd in sequence.

Solution: This implies that SQ2=1001. Thus,

$$ADC1->SQR3 = 0x000000120 \tag{12.7.3.11.1}$$

12.7.3.12. List the various pin numbers corresponding to the different channels of the ADC.

Solution: See Fig. 12.7.3.12.1

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

Table 12.7.3.12.1: ADC Analog Input Pins

12.7.3.13. Use the 9th channel of ADC1 in SQ2 to measure 3.3V.

Solution: Connect PB1 (Fig. 12.7.3.12.1) to 3.3 V of the STM 32 and execute the following code.

12.7.3.14. Before flashing code to STM32 set the STM32 board to Programming mode as shown in Fig. 12.1.1.7.1, then change the mode to Operating mode as shown in Fig. 12.1.1.7.1 after flashing the code inorder to run your program from main memory.

```
cd /adc/3_3V.c
```

12.7.3.15. Measure an unkown resistance using the STM32 and display the result on the LCD.

12.7.3.16. Display the output of the internal temperature sensor as well the unknown resistance on the LCD.

Chapter 13

Picoprobe

13.1. Setup

This section enumerates the steps needed to create a debugger probe from a Raspberry Pi Pico, called the Picoprobe. For further details refer to the documentation.

Note that all instructions in this chapter are for Linux systems running Debian-based distributions only. For other operating systems refer to the above link.

1. Setup the Raspberry Pi Pico environment on your laptop by entering the following commands at a terminal window.

```
sudo apt update && sudo apt upgrade
sudo apt install pkg-config
cd
wget https://raw.githubusercontent.com/raspberrypi/pico-setup/master/pico_setup
    .sh
chmod +x ./pico_setup.sh
SKIP_VSCODE=1 INCLUDE_PICOPROBE=1 ./pico_setup.sh
```

2. Download the picoprobe UF2 file from this link.

3. Connect the Raspberry Pi Pico board to be used as the debugger (henceforth referred to as the “debugger”) to your laptop and simultaneously press the BOOTSEL button to boot the debugger in bootloader mode.

4. Flash the downloaded `picoprobe.uf2` file using `picotool` to the debugger as follows.

```
picotool save /path/to/picoprobe.uf2
```

This will convert the Raspberry Pi Pico board into a Picoprobe debugger which we can use to flash and debug programs on other boards.

13.2. Raspberry Pi Pico

This section enumerates the steps needed to use the Picoprobe to debug a Raspberry Pi Pico from the command line using OpenOCD and GDB.

13.2.1. Components

Component	**Value**	**Quantity**
Raspberry Pi	Picoprobe	1
Raspberry Pi	Pico	1
Jumper Wires	Male to Female	10
	Female to Female	10
USB Cable	Type B	1

Table 4.1: List of components required to debug a Raspberry Pi Pico.

13.2.2. Wiring

Wire the debugger to the other Raspberry Pi Pico board by following the wiring diagram in Figure 4.1.

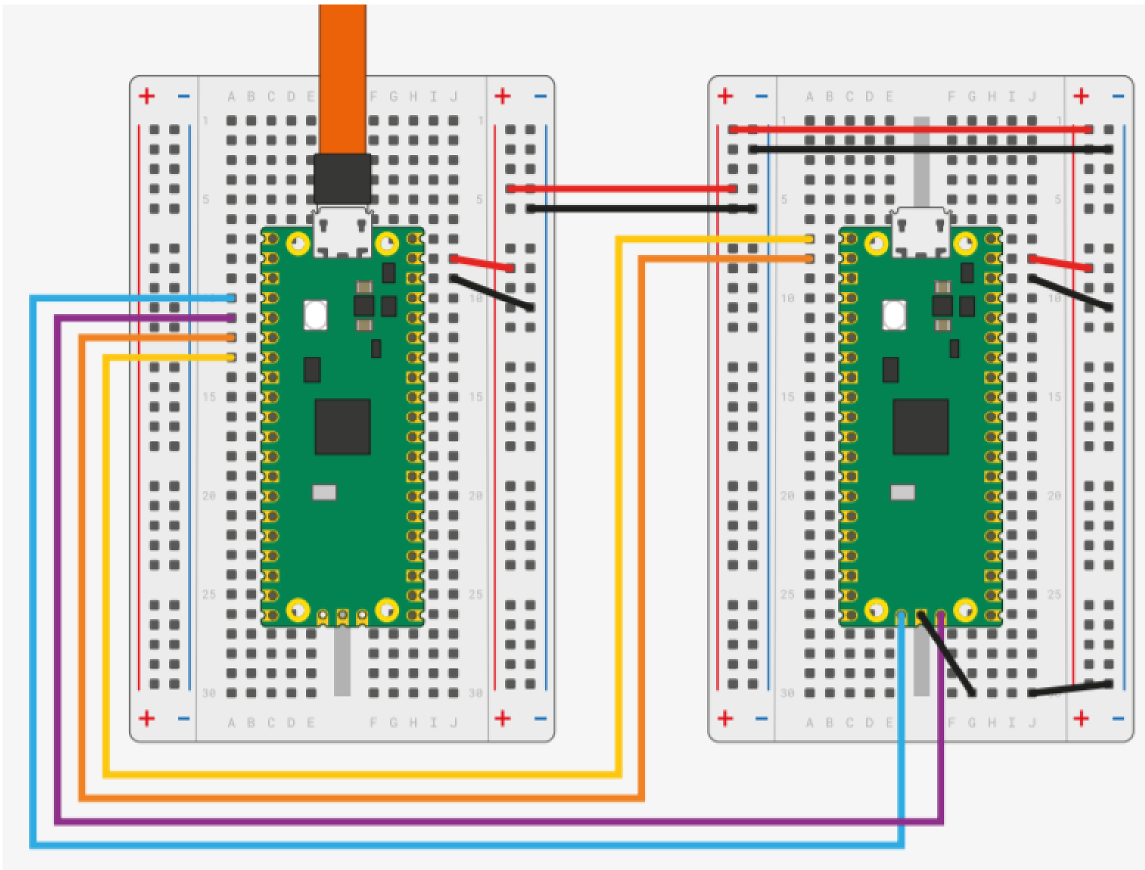


Figure 4.1: Wiring diagram to debug a Pico (right) using another Pico as Picoprobe (left).

13.2.3. Building

1. Build a debuggable ELF file to load onto the Pico by entering the following commands at a terminal window.

```
cd pico/pico-examples/build
rm -rf ./*
source ~/.bashrc
cmake -DCMAKE_BUILD_TYPE=Debug ..
cd hello_world/serial
make -j4
```

2. A successful build should generate the debuggable ELF file

```
pico/pico-examples/build/hello_world/serial/hello_serial.elf
```

13.2.4. Debugging

1. Connect the debugger to the laptop again and simultaneously press the BOOTSEL button on the target Pico board (and *not* the debugger) to boot it in bootloader mode.

2. In a terminal window, start an OpenOCD server by entering the following commands.

```
cd pico/openocd
sudo src/openocd -f tcl/interface/cmsis-dap.cfg -f tcl/target/rp2040.cfg -s tcl -c
    "adapter speed 5000"
```

A successful execution of these commands will result in the following (truncated) output.

```
Info: Listening on port 3333 for gdb connections
```

3. In *another* terminal window, start `gdb` with the ELF file as follows.

```
cd pico/pico-examples/build/hello_world/serial
gdb-multiarch hello_serial.elf
(gdb) target extended-remote localhost:3333
(gdb) monitor reset init
(gdb) break main
(gdb) continue
(gdb) layout src
```

Here, you should see the source code for the program flashed onto the target Pico.

(Optional) You can also use `tmux` instead of two separate terminal instances.

4. You can execute instruction by instruction by typing `next` at the GDB prompt. Alternatively, you can step into functions called from the main function by `step`.

5. To reset to the start of the program and reach the main breakpoint again, type the following.

```
(gdb) monitor reset init
(gdb) continue
```

To quit from gdb, type the following.

```
(gdb) quit
```

For other functionalities type `help` at the GDB prompt.

6. To see serial output, attach a terminal to the debugger by typing the following commands at a terminal window.

```
sudo minicom -D /dev/ttyACM0 -b 115200
```

Note that if the debugger is not present at `/dev/ttyACM0`, then you can find the correct port by inspecting the output produced by the following command.

```
sudo dmesg -w
```

13.3. Vaman

This section enumerates the steps needed to use the Picoprobe to debug an LC Vaman board from the command line using OpenOCD and GDB.

13.3.1. Components

Component	Value	Quantity
Raspberry Pi	Picoprobe	1
Vaman	LC	1
Jumper Wires	Female to Female	10
USB Cable	Type B	2

Table 6.1: List of components needed to debug Vaman with a Picoprobe.

13.3.2. Wiring

Wire the debugger to the other Raspberry Pi Pico board by following the wiring diagram in Table 6.2.

Vaman-Pygmy	Picoprobe
IO_14 (SWDIO)	GP2
IO_15 (SWCLK)	GP3
IO_44 (UART0 TX)	GP5 (UART0 RX)
IO_45 (UART0 RX)	GP4 (UART0 TX)
IO_19	VSYS
GND	GND

Table 6.2: Connections required to debug Vaman with a Picoprobe.

13.3.3. Building

1. Build a debuggable ELF file to load onto the Vaman by entering the following commands at a terminal window.

```
cd vaman/arm/codes/setup/blink/GCC_Project
make -j4
```

2. A successful build should generate the debuggable ELF file

```
vaman/arm/codes/setup/blink/GCC_Project/output/bin
```

13.3.4. Debugging

1. Connect the debugger and Vaman to the laptop simultaneously.

2. In a terminal window, start an OpenOCD server by entering the following commands.

```
cd pico/openocd
sudo src/openocd -f tcl/interface/cmsis-dap.cfg -f tcl/target/eos_s3.cfg -s tcl
```

A successful execution of these commands will result in the following (truncated) output.

```
Info: Listening on port 3333 for gdb connections
```

3. In *another* terminal window, start `gdb` with the ELF file as follows.

```
cd vaman/arm/codes/setup/blink/GCC_Project/output/bin
gdb-multiarch blink.elf
(gdb) target extended-remote localhost:3333
(gdb) break main
(gdb) layout src
(gdb) continue
```

   Here, you should see the source code for the program flashed onto the Vaman

   (Optional) You can also use `tmux` instead of two separate terminal instances.

4. You can execute instruction by instruction by typing `next` at the GDB prompt. Alternatively, you can step into functions called from the main function by `step`.

5. To reset to the start of the program and reach the main breakpoint again, type the following.

```
(gdb) monitor reset init
(gdb) load
(gdb) continue
```

   To quit from gdb, type the following.

```
(gdb) quit
```

   For other functionalities type `help` at the GDB prompt.

6. To see serial output, attach a terminal to the debugger by typing the following commands at a terminal window.

```
sudo minicom −D /dev/ttyACM0 −b 115200
```

Note that if the debugger is not present at `/dev/ttyACM0`, then you can find the correct port by inspecting the output produced by the following command.

```
sudo dmesg −w
```

Appendix A

Raspberry Pi - Vaman ESP32 through UART

A.1. Enable Serial Communication

A.1.1. On the RPi,

```
sudo raspi—config
```

A.1.2. Select Interfacing Options

A.1.3. Then select Serial Port

A.1.4. Reply no to login shell over serial

A.1.5. Say yes to running hardware over serial port.

A.1.6. Connect the rpi tx (pin 8) and rx (pin 10). See Fig. A.1.6.1

A.1.7. Install minicom and start it

```
sudo apt install minicom
minicom —b 115200 —o —D /dev/serial0
```

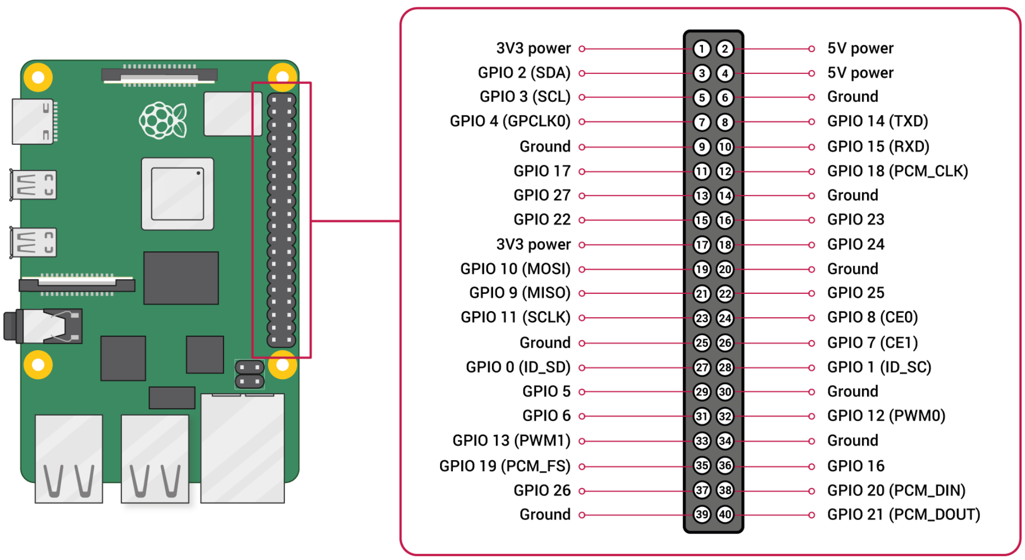


Figure A.1.6.1:

Type namaste. If you see it displayed on screen, your serial port is working.

A.2. Flash Vaman-ESP

A.2.0.1. Since the RPi supports UART through its GPIO pins, a separate USB UART adapter is not required. Make connections according to Table A.2.0.1.1 On RPi, the pin numbers for serial communication are Tx=8, Rx=10.

A.2.0.2. Connect the Vaman-ESP pins to the seven segment display according to Table A.2.0.2.1 The GPIO pins are listed in Table A.2.0.2.2 Note that these pins can be used for several functions, refer to the ESP32 datasheet for details.

A.2.0.3. On termux, execute the blink program.

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

Table A.2.0.1.1:

ESP	SEVEN SEGMENT DISPLAY
5V	COM
2	DOT

Table A.2.0.2.1:

A.2.0.4. 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
```

A.2.0.5. On rpi, modify your platformio.ini file by adding the line

```
upload_port = /dev/serial0
```

A.2.0.6. Now execute the following on the rpi.

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

While the dots and dashes are printed on the screen, disconnect the EN wire from GND. Make sure that the Vaman board is not powering any device while flashing.

GPIO	Input	Others
2	34	1
4	35	3
5	36	6
10	37	7
12	38	8
13	39	9
14		10
15		11
16		
17		
18		
19		
21		
22		
23		
25		
26		
27		
32		
33		

Table A.2.0.2.2:

The Vaman-ESP should now flash.

A.2.0.7. After flashing, disconnect pin 0 on Vaman-ESP from GND. Power on Vaman and the appropriate LED will blink.

A.3. OTA

A.3.1. Connect the pins between Vaman-ESP32 and Vaman-PYGMY as per Table A.3.1.1

ESP32	Vaman
GPIO2	GPIO18
GPIO4	GPIO21
GPIO5	GPIO22

Table A.3.1.1:

A.3.2. Flash the following code OTA

```
https://github.com/gadepall/vaman/tree/master/esp32/codes/ide/ota/blinkt
```

You should see the onboard green LED blinking.

A.3.3. Change the blink duration to 100 ms.