

Subject Name Solutions

4321102 – Winter 2023

Semester 1 Study Material

Detailed Solutions and Explanations

Question 1(a) [3 marks]

$(726)_{10} = (\text{_____})_2$

Solution

Table 1: Decimal to Binary Conversion

Step	Calculation	Remainder
1	$726 \div 2 = 363$	0
2	$363 \div 2 = 181$	1
3	$181 \div 2 = 90$	1
4	$90 \div 2 = 45$	0
5	$45 \div 2 = 22$	1
6	$22 \div 2 = 11$	0
7	$11 \div 2 = 5$	1
8	$5 \div 2 = 2$	1
9	$2 \div 2 = 1$	0
10	$1 \div 2 = 0$	1

Reading from bottom to top: $(726)_{10} = (1011010110)_2$

Mnemonic

“Divide By Two, Read Remainders Up”

Question 1(b) [4 marks]

- 1) Convert binary number $(10110101)_2$ into gray number.
- 2) Convert gray number $(10110110)_{gray}$ into binary number.

Solution

Binary to Gray Conversion:

Binary: 1 0 1 1 0 1 0 1
 ↓ ↓ ↓ ↓ ↓ ↓ ↓
XOR: $1 \oplus 0$ $0 \oplus 1$ $1 \oplus 1$ $1 \oplus 0$ $0 \oplus 1$ $1 \oplus 0$ $0 \oplus 1$
 ↓ ↓ ↓ ↓ ↓ ↓ ↓
Gray: 1 1 0 1 1 1 1

Therefore: $(10110101)_2 = (1101111)_{gray}$

Gray to Binary Conversion:

Gray: 1 0 1 1 0 1 1 0
 ↓
Binary: 1
 $1 \oplus 0 = 1$
 $1 \oplus 1 = 0$
 $0 \oplus 1 = 1$
 $1 \oplus 0 = 1$
 $1 \oplus 1 = 0$
 $0 \oplus 1 = 1$

$$1 \oplus 0 = 1$$

Therefore: $(10110110)_{\text{gray}} = (10110101)_2$

Mnemonic

“First bit same, rest XOR with previous binary”

Question 1(c) [7 marks]

Explain NAND as a universal gate.

Solution

Diagram: NAND as Universal Gate

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph TD
    subgraph "NOT Gate using NAND"
        A1(A){-{-}{N1((NAND))}}
        A1(A){-{-}{N1}}
        N1{-{-}{Z1("A{")}}
    end

    subgraph "AND Gate using NAND"
        A2(A){-{-}{N2((NAND))}}
        B2(B){-{-}{N2}}
        N2{-{-}{N3((NAND))}}
        N2{-{-}{N3}}
        N3{-{-}{Z2("A·B")}}
    end

    subgraph "OR Gate using NAND"
        A3(A){-{-}{N4((NAND))}}
        A3(A){-{-}{N4}}
        B3(B){-{-}{N5((NAND))}}
        B3(B){-{-}{N5}}
        N4{-{-}{N6((NAND))}}
        N5{-{-}{N6}}
        N6{-{-}{Z3("A+B")}}
    end
{Highlighting}
{Shaded}
```

- **Universal Property:** NAND gate can implement any Boolean function without needing any other type of gate
- **NOT Implementation:** Connecting both inputs of NAND together creates NOT gate
- **AND Implementation:** NAND followed by another NAND creates AND gate
- **OR Implementation:** Two NAND gates with single inputs, followed by NAND creates OR gate

Table 2: NAND Gate Implementations

Logic Function	NAND Implementation
NOT(A)	NAND(A,A)
AND(A,B)	NAND(NAND(A,B),NAND(A,B))
OR(A,B)	NAND(NAND(A,A),NAND(B,B))

Mnemonic

“NAND can STAND as All gates”

Question 1(c) OR [7 marks]

Explain NOR as a universal gate.

Solution

Diagram: NOR as Universal Gate

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph TD
    subgraph "NOT Gate using NOR"
        A1(A){--}{N1((NOR))}
        A1(A){--}{N1}
        N1{--}{Z1("A{")})
    end

    subgraph "OR Gate using NOR"
        A2(A){--}{N2((NOR))}
        B2(B){--}{N2}
        N2{--}{N3((NOR))}
        N2{--}{N3}
        N3{--}{Z2("A+B{")})
    end

    subgraph "AND Gate using NOR"
        A3(A){--}{N4((NOR))}
        A3(A){--}{N4}
        B3(B){--}{N5((NOR))}
        B3(B){--}{N5}
        N4{--}{N6((NOR))}
        N5{--}{N6}
        N6{--}{Z3("A·B{")})
    end
{Highlighting}
{Shaded}
```

- **Universal Property:** NOR gate can implement any Boolean function without needing any other type of gate
- **NOT Implementation:** Connecting both inputs of NOR together creates NOT gate
- **OR Implementation:** NOR followed by another NOR creates OR gate
- **AND Implementation:** Two NOR gates with single inputs, followed by NOR creates AND gate

Table 3: NOR Gate Implementations

Logic Function	NOR Implementation
NOT(A)	NOR(A,A)
OR(A,B)	NOR(NOR(A,B),NOR(A,B))
AND(A,B)	NOR(NOR(A,A),NOR(B,B))

Mnemonic

“NOR can form ALL logic cores”

Question 2(a) [3 marks]

$$(11011011)_2 \times (110)_2 = (\text{_____})_2$$

Solution

Table: Binary Multiplication

$$\begin{array}{r}
 11011011 \\
 \times 110 \\
 \hline
 11011011 \quad (\times 0) \\
 11011011 \quad (\times 1) \\
 11011011 \quad (\times 1) \\
 \hline
 100000000110
 \end{array}$$

Therefore: $(11011011)_2 \times (110)_2 = (10000001110)_2$

Mnemonic

“Multiply each bit, shift left, add all rows”

Question 2(b) [4 marks]

Prove DeMorgan's theorem.

Solution

Table 4: DeMorgan's Theorem Proof

A	B	A'	B'	A+B	(A+B)'	A' · B'
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

DeMorgan's Theorems: 1. $(A+B)' = A' \cdot B'$ 2. $(A \cdot B)' = A' + B'$
 Truth table proves that $(A+B)' = A' \cdot B'$ since both columns match.

Mnemonic

“Break the line, change the sign”

Question 2(c) [7 marks]

Explain full adder using logic circuit, Boolean equation and truth table.

Solution

Diagram: Full Adder Circuit

Mermaid Diagram (Code)

```

{Shaded}
{Highlighting}[]
graph LR
    A(A){-{-}{}}XOR1(XOR){}
    B(B){-{-}{}}XOR1
    XOR1{-{-}{}}XOR2(XOR){}
    Cin(Cin){-{-}{}}XOR2
    XOR2{-{-}{}}Sum(Sum){}
  
```

```

A{-{-}{-}{-}}AND1(AND){}
B{-{-}{-}{-}}AND1{}
AND1{-{-}{-}{-}}OR1(OR){}

A{-{-}{-}{-}}AND2(AND){}
Cin{-{-}{-}{-}}AND2{}
AND2{-{-}{-}{-}}OR1{}

B{-{-}{-}{-}}AND3(AND){}
Cin{-{-}{-}{-}}AND3{}
AND3{-{-}{-}{-}}OR1{}

OR1{-{-}{-}{-}}Cout(Cout){}
{Highlighting}
{Shaded}

```

Table 5: Full Adder Truth Table

A	B	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- **Boolean Equations:**
 - **Sum** = $A \oplus B \oplus Cin$
 - **Cout** = $(A \cdot B) + (B \cdot Cin) + (A \cdot Cin)$

Mnemonic

“Sum needs XOR three, Carry needs AND then OR”

Question 2(a) OR [3 marks]

Divide $(11010010)_2$ with $(101)_2 = (\text{_____})_2$

Solution

Table: Binary Division

```

      1 0 1 0 1 1
      -----
101 ) 1 1 0 1 0 0 1 0
      1 0 1
      ----
        1 1 0
        1 0 1
        ----
          0 1 0
          0 0
          ----
            1 0 0
            1 0 1
            ----
              1 1 0
              1 0 1
              ----

```

$$\begin{array}{r}
 0\ 1\ 0 \\
 0\ 0 \\
 \hline
 1\ 0 \\
 0 \\
 \hline
 0
 \end{array}$$

Therefore: $(11010010)_2 \div (101)_2 = (101011)_2 \text{ with remainder } (0)_2$

Mnemonic

“Divide like decimal, but use binary subtraction”

Question 2(b) OR [4 marks]

Simplify the Boolean expression $Y = A'B + AB' + A'B' + AB$

Solution

Table 6: Boolean Simplification

Step	Expression	Rule Applied
1	$Y = A'B + AB' + A'B' + AB$	Original
2	$Y = A'(B + B') + A(B' + B)$	Factoring
3	$Y = A'(1) + A(1)$	$B + B' = 1$
4	$Y = A' + A$	Simplifying
5	$Y = 1$	$A' + A = 1$

Therefore: $Y = 1$ (Always TRUE)

Mnemonic

“Factor first, apply identities, combine like terms”

Question 2(c) OR [7 marks]

Explain full subtractor using logic circuit, boolean equation and truth table.

Solution

Diagram: Full Subtractor Circuit

Mermaid Diagram (Code)

```

{Shaded}
{Highlighting}[]
graph LR
    A(A){--}{XOR1(XOR)}
    B(B){--}{XOR1}
    XOR1{--}{XOR2(XOR)}
    Bin(Bin){--}{XOR2}
    XOR2{--}{D(Difference)}

    A(A){--}{NOT1(NOT)}
    NOT1{--}{AND1(AND)}
    B{--}{AND1}
    AND1{--}{OR1(OR)}

    XOR1{--}{NOT2(NOT)}
    NOT2{--}{AND2(AND)}

```

Bin{--}{ }AND2}
AND2{--}{ }OR1}

B{--}{ }AND3(AND)}
Bin{--}{ }AND3}
AND3{--}{ }OR1}

OR1{--}{ }Bout(Borrow Out)}
{Highlighting}
{Shaded}

Table 7: Full Subtractor Truth Table

A	B	Bin	Difference	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

- Boolean Equations:
 - Difference = $A \oplus B \oplus Bin$
 - Bout = $(A' \cdot B) + (A' \cdot Bin) + (B \cdot Bin)$

Mnemonic

“Difference uses triple XOR, Borrow when input is greater”

Question 3(a) [3 marks]

Using 2's complement subtract $(1011001)_2$ from $(1101101)_2$.

Solution

Table 8: 2's Complement Subtraction

Step	Operation	Result
1	Number to subtract:	1011001
2	1's complement:	0100110
3	2's complement:	0100111
4	$(1101101) + (0100111) =$	10010100
5	Discard carry:	0010100

Therefore: $(1101101)_2 - (1011001)_2 = (0010100)_2 = (20)_{10}$

Mnemonic

“Flip bits, add one, then add numbers”

Question 3(b) [4 marks]

Simplify the Boolean equation using Karnaugh map (K' map) method: $F(A,B,C,D) = \Sigma m(0,1,2,6,7,8,12,15)$

Solution

Table: Karnaugh Map

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

Diagram: K-map Grouping

+	-	{	-	{	-	{	-	+	-	{	-	{	-	{	-	+	-	{	-	{	-	+	-	{	-	{	-	+	-	{	-	{	-	+
	1		1		0		1																											
	A		A				A																											
+	-	{	-	{	-	{	-	+	-	{	-	{	-	{	-	+	-	{	-	{	-	+	-	{	-	{	-	+	-	{	-	{	-	+
	0		0		1		1																											
					B		B																											
+	-	{	-	{	-	{	-	+	-	{	-	{	-	{	-	+	-	{	-	{	-	+	-	{	-	{	-	+	-	{	-	{	-	+
	0		0		1		0																											
					B																													
+	-	{	-	{	-	{	-	+	-	{	-	{	-	{	-	+	-	{	-	{	-	+	-	{	-	{	-	+	-	{	-	{	-	+
	1		0		0		0																											
	C																																	
+	-	{	-	{	-	{	-	+	-	{	-	{	-	{	-	+	-	{	-	{	-	+	-	{	-	{	-	+	-	{	-	{	-	+

Group A: A'B'C' (4 cells) Group B: BCD (3 cells) Group C: A'B'CD' (1 cell)
Simplified expression: $F(A,B,C,D) = A'B'C' + BCD + A'B'CD'$

Mnemonic

“Find largest groups of 2^n , *use minimal terms*”

Question 3(c) [7 marks]

Explain 3 to 8 decoder using logic circuit and truth table.

Solution

Diagram: 3-to-8 Decoder

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph TD
    A(A){-{-}{}}NOT1(NOT)}
    B(B){-{-}{}}NOT2(NOT)}
    C(C){-{-}{}}NOT3(NOT)}

    NOT1{-{-}{}}AND0(AND)}
    NOT2{-{-}{}}AND0}
    NOT3{-{-}{}}AND0}
    AND0{-{-}{}}DO(DO)}

    NOT1{-{-}{}}AND1(AND)}
    NOT2{-{-}{}}AND1}
    C{-{-}{}}AND1}
    AND1{-{-}{}}D1(D1)}

    NOT1{-{-}{}}AND2(AND)}
    B{-{-}{}}AND2}
    NOT3{-{-}{}}AND2}
```

AND2{--}{D2(D2)}

NOT1{--}{AND3(AND)}

B{--}{AND3}

C{--}{AND3}

AND3{--}{D3(D3)}

A{--}{AND4(AND)}

NOT2{--}{AND4}

NOT3{--}{AND4}

AND4{--}{D4(D4)}

A{--}{AND5(AND)}

NOT2{--}{AND5}

C{--}{AND5}

AND5{--}{D5(D5)}

A{--}{AND6(AND)}

B{--}{AND6}

NOT3{--}{AND6}

AND6{--}{D6(D6)}

A{--}{AND7(AND)}

B{--}{AND7}

C{--}{AND7}

AND7{--}{D7(D7)}

{Highlighting}

{Shaded}

Table 9: 3-to-8 Decoder Truth Table

Inputs			Outputs							
A	B	C	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

- **Function:** Activates one of 8 output lines based on 3-bit binary input
- **Applications:** Memory addressing, data routing, instruction decoding
- **Boolean Equations:** $D0 = A' \cdot B' \cdot C'$, $D1 = A' \cdot B' \cdot C$, etc.

Mnemonic

“One hot output at binary address”

Question 3(a) OR [3 marks]

Do as directed. 1) $(101011010111)_2 = (\text{_____})_8$

Solution

Table: Binary to Octal Conversion

Binary:	1	010	110	101	11
	↓	↓	↓	↓	↓
Octal:	1	2	6	5	3

Therefore: $(101011010111)_2 = (12653)_8$

Mnemonic

“Group by threes, right to left”

Question 3(b) OR [4 marks]

Simplify the Boolean equation using Karnaugh map (K' map) method: $F(A,B,C,D) = \sum m(1,3,5,7,8,9,10,11)$

Solution

Table: Karnaugh Map

Table: Hamming Map				
	CD			
AB	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	0	0	0
10	1	1	1	1

Diagram: K-map Grouping

Diagram: A map Grouping

+{-}{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}

0	1	1	0
A	A		

+{-}{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}

0	1	1	0
A	A		

+{-}{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}

0	0	0	0

+{-}{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}

1	1	1	1
B	B	B	B

+{-}{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}+{-}{-}{-}{-}{-}{-}

Group A: A'CD (4 cells) Group B: AB' (4 cells)
Simplified expression: $F(A,B,C,D) = A'CD + AB'$

Mnemonic

“Group powers of 2, minimize variables”

Question 3(c) OR [7 marks]

Explain 8 to 1 multiplexer using logic circuit and truth table.

Solution

Diagram: 8-to-1 Multiplexer

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph TD
    D0(D0){-{-}{}}AND0(AND)}
    D1(D1){-{-}{}}AND1(AND)}
    D2(D2){-{-}{}}AND2(AND)}
    D3(D3){-{-}{}}AND3(AND)}
    D4(D4){-{-}{}}AND4(AND)}
```

D5(D5){-{-}{}}AND5(AND)}
D6(D6){-{-}{}}AND6(AND)}
D7(D7){-{-}{}}AND7(AND)}

S0(S0){-{-}{}}NOT0(NOT)}
S1(S1){-{-}{}}NOT1(NOT)}
S2(S2){-{-}{}}NOT2(NOT)}

NOT0{-{-}{}}AND0}
NOT1{-{-}{}}AND0}
NOT2{-{-}{}}AND0}

S0{-{-}{}}AND1}
NOT1{-{-}{}}AND1}
NOT2{-{-}{}}AND1}

NOT0{-{-}{}}AND2}
S1{-{-}{}}AND2}
NOT2{-{-}{}}AND2}

S0{-{-}{}}AND3}
S1{-{-}{}}AND3}
NOT2{-{-}{}}AND3}

NOT0{-{-}{}}AND4}
NOT1{-{-}{}}AND4}
S2{-{-}{}}AND4}

S0{-{-}{}}AND5}
NOT1{-{-}{}}AND5}
S2{-{-}{}}AND5}

NOT0{-{-}{}}AND6}
S1{-{-}{}}AND6}
S2{-{-}{}}AND6}

S0{-{-}{}}AND7}
S1{-{-}{}}AND7}
S2{-{-}{}}AND7}

AND0{-{-}{}}OR1(OR)}
AND1{-{-}{}}OR1}
AND2{-{-}{}}OR1}
AND3{-{-}{}}OR1}
AND4{-{-}{}}OR1}
AND5{-{-}{}}OR1}
AND6{-{-}{}}OR1}
AND7{-{-}{}}OR1}

OR1{-{-}{}}Y(Output Y)}

{Highlighting}
{Shaded}

Table 10: 8-to-1 Multiplexer Truth Table

Select Lines			Output
S2	S1	S0	Y
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4

1	0	1	D5
1	1	0	D6
1	1	1	D7

- **Function:** Selects one of 8 input data lines and routes it to output
- **Applications:** Data routing, function generation, parallel-to-serial conversion
- **Boolean Equation:** $Y = S2' \cdot S1' \cdot S0' \cdot D0 + S2' \cdot S1' \cdot S0 \cdot D1 + \dots + S2 \cdot S1 \cdot S0 \cdot D7$

Mnemonic

“Select bits route one input to output”

Question 4(a) [3 marks]

Draw the logic circuit for binary to gray convertor.

Solution

Diagram: Binary to Gray Code Converter

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph TD
    B3(B3){-{-}{}}G3(G3)
    B3{-{-}{}}XOR1(XOR)
    B2(B2){-{-}{}}XOR1
    XOR1{-{-}{}}G2(G2)
    B2{-{-}{}}XOR2(XOR)
    B1(B1){-{-}{}}XOR2
    XOR2{-{-}{}}G1(G1)
    B1{-{-}{}}XOR3(XOR)
    B0(B0){-{-}{}}XOR3
    XOR3{-{-}{}}G0(G0)
{Highlighting}
{Shaded}
```

- **Binary Inputs:** B3, B2, B1, B0 (most to least significant bits)
- **Gray Outputs:** G3, G2, G1, G0 (most to least significant bits)
- **Conversion Rule:** $G3 = B3, G2 = B3 \oplus B2, G1 = B2 \oplus B1, G0 = B1 \oplus B0$

Mnemonic

“First bit same, rest XOR neighbors”

Question 4(b) [4 marks]

Explain working of Serial in Serial out shift register.

Solution

Diagram: Serial-In Serial-Out Shift Register

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    Din(Data In){-{-}{}}FF0(FF0)
    CLK(Clock){-{-}{}}FF0
    CLK{-{-}{}}FF1(FF1)
```

```

CLK{-{-}}{}FF2(FF2)}
CLK{-{-}}{}FF3(FF3)}
FF0{-{-}}{}FF1}
FF1{-{-}}{}FF2}
FF2{-{-}}{}FF3}
FF3{-{-}}{}Dout(Data Out)}
{Highlighting}
{Shaded}

```

Table 11: Serial-In Serial-Out Operation

Clock Cycle	FF0	FF1	FF2	FF3	Data Out
Initial	0	0	0	0	0
1 (Din=1)	1	0	0	0	0
2 (Din=0)	0	1	0	0	0
3 (Din=1)	1	0	1	0	0
4 (Din=1)	1	1	0	1	1

- Operation: Data bits enter serially at input, shift through all flip-flops, and exit serially
- Applications: Data transmission, time delay, serial-to-serial conversion
- Features: Simple design, requires fewer I/O pins but more clock cycles

Mnemonic

“One bit in, shift all, one bit out”

Question 4(c) [7 marks]

Explain workings of D flip flop and JK flip flop using circuit diagram and truth table.

Solution

Diagram: D Flip-Flop

Mermaid Diagram (Code)

```

{Shaded}
{Highlighting}[]
graph LR
    D(D){-{-}}{}DFF(D Flip{-}Flop)}
    CLK(Clock){-{-}}{}DFF}
    DFF{-{-}}{}Q(Q)}
    DFF{-{-}}{}Q(Q)}("Q{")
{Highlighting}
{Shaded}

```

Table 12: D Flip-Flop Truth Table

D	Clock	Q(next)
0	↑	0
1	↑	1

Diagram: JK Flip-Flop

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    J(J){-{-}{}}JKFF(JK Flip{-}Flop)}
    K(K){-{-}{}}JKFF}
    CLK(Clock){-{-}{}}JKFF}
    JKFF{-{-}{}}Q(Q)}
    JKFF{-{-}{}}Q{"Q{}}"}
{Highlighting}
{Shaded}
```

Table 13: JK Flip-Flop Truth Table

J	K	Clock	Q(next)
0	0	↑	Q(no change)
0	1	↑	0
1	0	↑	1
1	1	↑	Q' (toggle)

- D Flip-Flop: Data (D) input is transferred to output Q at positive clock edge
- JK Flip-Flop: More versatile with set (J), reset (K), hold and toggle capabilities
- Applications: Storage elements, counters, registers, sequential circuits

Mnemonic

“D Does what D is, JK Juggles Keep-Toggle-Set”

Question 4(a) OR [3 marks]

Draw the logic circuit for gray to binary convertor.

Solution

Diagram: Gray to Binary Code Converter

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    G3(G3){-{-}{}}B3(B3)}
    G3{-{-}{}}XOR1(XOR)}
    G2(G2){-{-}{}}XOR1}
    XOR1{-{-}{}}B2(B2)}
    XOR1{-{-}{}}XOR2(XOR)}
    G1(G1){-{-}{}}XOR2}
    XOR2{-{-}{}}B1(B1)}
    XOR2{-{-}{}}XOR3(XOR)}
    G0(G0){-{-}{}}XOR3}
    XOR3{-{-}{}}B0(B0)}
{Highlighting}
{Shaded}
```

- Gray Inputs: G3, G2, G1, G0 (most to least significant bits)
- Binary Outputs: B3, B2, B1, B0 (most to least significant bits)
- Conversion Rule: $B3 = G3$, $B2 = B3 \oplus G2$, $B1 = B2 \oplus G1$, $B0 = B1 \oplus G0$

Mnemonic

“First bit same, rest XOR with previous result”

Question 4(b) OR [4 marks]

Explain working of Parallel in Parallel out shift register.

Solution

Diagram: Parallel-In Parallel-Out Shift Register

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph TD
    D0(D0){-{-}{}}FF0(FF0)}
    D1(D1){-{-}{}}FF1(FF1)}
    D2(D2){-{-}{}}FF2(FF2)}
    D3(D3){-{-}{}}FF3(FF3)}
    CLK(Clock){-{-}{}}FF0}
    CLK{-{-}{}}FF1}
    CLK{-{-}{}}FF2}
    CLK{-{-}{}}FF3}
    LOAD(Load){-{-}{}}FF0}
    LOAD{-{-}{}}FF1}
    LOAD{-{-}{}}FF2}
    LOAD{-{-}{}}FF3}
    FF0{-{-}{}}Q0(Q0)}
    FF1{-{-}{}}Q1(Q1)}
    FF2{-{-}{}}Q2(Q2)}
    FF3{-{-}{}}Q3(Q3)}
{Highlighting}
{Shaded}
```

Table 14: Parallel-In Parallel-Out Operation

LOAD	Clock	D0-D3	Q0-Q3 (after clock)
1	↑	1010	1010
0	↑	xxxx	1010 (no change)
1	↑	0101	0101

- Operation: Data loaded in parallel, all bits simultaneously transferred to outputs
- Applications: Data storage, buffering, temporary holding registers
- Features: Fastest register type, requires most I/O pins, no bit shifting

Mnemonic

“All in, all out, all at once”

Question 4(c) OR [7 marks]

Explain workings of T flip flop and SR flip flop using circuit diagram and truth table.

Solution

Diagram: T Flip-Flop

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
```

```
graph LR
    T(T){-}{-}{TFF(T Flip{-}Flop)}
    CLK(Clock){-}{-}{TFF}
    TFF{-}{-}{Q(Q)}
    TFF{-}{-}{Q}{("Q{")}}
    {Highlighting}
    {Shaded}
```

Table 15: T Flip-Flop Truth Table

T	Clock	Q(next)
0	↑	Q (no change)
1	↑	Q' (toggle)

Diagram: SR Flip-Flop

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    S(S){-}{-}{SRFF(SR Flip{-}Flop)}
    R(R){-}{-}{SRFF}
    CLK(Clock){-}{-}{SRFF}
    SRFF{-}{-}{Q(Q)}
    SRFF{-}{-}{Q}{("Q{")}}
    {Highlighting}
    {Shaded}
```

Table 16: SR Flip-Flop Truth Table

S	R	Clock	Q(next)
0	0	↑	Q (no change)
0	1	↑	0 (reset)
1	0	↑	1 (set)
1	1	↑	Invalid

- T Flip-Flop: Toggle flip-flop changes state when T=1, maintains state when T=0
- SR Flip-Flop: Basic flip-flop with Set (S) and Reset (R) inputs
- Applications: T flip-flops for counters and frequency dividers, SR for basic memory

Mnemonic

“T Toggles when True, SR Sets or Resets”

Question 5(a) [3 marks]

Compare TTL, CMOS and ECL logic families.

Solution

Table 17: Comparison of Logic Families

Parameter	TTL	CMOS	ECL
Power Consumption	Medium	Very Low	High
Speed	Medium	Low-Medium	Very High
Noise Immunity	Medium	High	Low
Fan-out	10	>50	25
Supply Voltage	+5V	+3V to +15V	-5.2V
Complexity	Medium	Low	High

- **TTL:** Transistor-Transistor Logic - Good balance of speed and power
- **CMOS:** Complementary Metal-Oxide-Semiconductor - Low power, high density
- **ECL:** Emitter-Coupled Logic - Highest speed, used in high-performance applications

Mnemonic

“TCE: TTL Compromises, CMOS Economizes, ECL Excels in speed”

Question 5(b) [4 marks]

Explain decade counter with the help of logic circuit diagram and truth table.

Solution

Diagram: Decade Counter (BCD Counter)

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    CLK(Clock){-{-}{}}JK0(JK FF0)}
    JK0{-{-}{}}Q0(Q0)}
    JK0{-{-}{}}Q0{"Q0{}}"}
    Q0{-{-}{}}JK1(JK FF1)}
    JK1{-{-}{}}Q1(Q1)}
    JK1{-{-}{}}Q1{"Q1{}}"}
    Q1{-{-}{}}JK2(JK FF2)}
    JK2{-{-}{}}Q2(Q2)}
    JK2{-{-}{}}Q2{"Q2{}}"}
    Q2{-{-}{}}JK3(JK FF3)}
    JK3{-{-}{}}Q3(Q3)}
    JK3{-{-}{}}Q3{"Q3{}}"}
    Q3{-{-}{}}NAND1(NAND)}
    Q1{-{-}{}}NAND1}
    NAND1{-{-}{}}CLEAR(Clear)}
    CLEAR{-{-}{}}JK0}
    CLEAR{-{-}{}}JK1}
    CLEAR{-{-}{}}JK2}
    CLEAR{-{-}{}}JK3}
{Highlighting}
{Shaded}
```

Table 18: Decade Counter States

Count	Q3	Q2	Q1	Q0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
0	0	0	0	0

- **Function:** Counts from 0 to 9 (decimal) and then resets to 0
- **Applications:** Digital clocks, frequency dividers, BCD counters
- **Features:** Auto-reset at count 10, synchronous with clock

Mnemonic

“Counts one Decade, resets after nine”

Question 5(c) [7 marks]

Give Classification of Memories in detail.

Solution

Diagram: Memory Classification

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph TD
    M[Memory] --> PM[Primary Memory]
    M --> SM[Secondary Memory]

    PM --> RAM[RAM]
    PM --> ROM[ROM]

    RAM --> SRAM[Static RAM]
    RAM --> DRAM[Dynamic RAM]

    ROM --> MROM[Mask ROM]
    ROM --> PROM[Programmable ROM]
    ROM --> EPROM[Erasable PROM]
    ROM --> EEPROM[Electrically EPROM]

    EEPROM --> FLASH[Flash Memory]

    SM --> HD[Hard Disk]
    SM --> OD[Optical Disk]
    SM --> USB[USB Drives]
    SM --> SD[SD Cards]
{Highlighting}
{Shaded}
```

Table 19: Memory Types Comparison

Memory Type	Volatility	Read/Write	Access Speed	Typical Use
SRAM	Volatile	R/W	Very Fast	Cache memory
DRAM	Volatile	R/W	Fast	Main memory
ROM	Non-volatile	Read-only	Medium	BIOS, firmware
PROM	Non-volatile	Write-once	Medium	Permanent programs
EPROM	Non-volatile	Erasable UV	Medium	Upgradable firmware
EEPROM	Non-volatile	Electrically erasable	Medium	Configuration data
Flash	Non-volatile	Block erasable	Medium-Fast	Storage devices

- RAM (Random Access Memory): Temporary, volatile working memory
- ROM (Read Only Memory): Permanent, non-volatile program storage
- Characteristics: Access time, data retention, capacity, cost per bit

Mnemonic

“RAM Vanishes, ROM Remains”

Question 5(a) OR [3 marks]

Define: Fan out, Fan in and Figure of merit.

Solution

Table 20: Digital Logic Parameters

Parameter	Definition	Typical Values
Fan-out	Number of standard loads a gate output can drive	TTL: 10, CMOS: >50
Fan-in	Number of inputs a logic gate can handle	TTL: 8, CMOS: 100+
Figure of Merit	Speed-power product (propagation delay \times power consumption)	Lower is better

- Fan-out: Maximum number of gate inputs that can be connected to a gate output
- Fan-in: Maximum number of inputs available on a single logic gate
- Figure of Merit: Quality factor for comparing different logic families

Mnemonic

“Out drives many, In accepts many, Merit measures goodness”

Question 5(b) OR [4 marks]

Explain asynchronous up counter with the help of logic circuit diagram and truth table.

Solution

Diagram: 4-bit Asynchronous Up Counter

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    CLK(Clock){-{-}{-}}TFF0(T FF0)}
    TFF0{-{-}{-}}Q0(Q0)}
    TFF0{-{-}{-}}Q0{"Q0{}}"}
    Q0{-{-}{-}}TFF1(T FF1)}
    TFF1{-{-}{-}}Q1(Q1)}
    TFF1{-{-}{-}}Q1{"Q1{}}"}
    Q1{-{-}{-}}TFF2(T FF2)}
    TFF2{-{-}{-}}Q2(Q2)}
    TFF2{-{-}{-}}Q2{"Q2{}}"}
    Q2{-{-}{-}}TFF3(T FF3)}
    TFF3{-{-}{-}}Q3(Q3)}
    TFF3{-{-}{-}}Q3{"Q3{}}"}

    CLR(Clear){-{-}{-}}TFF0}
    CLR{-{-}{-}}TFF1}
    CLR{-{-}{-}}TFF2}
    CLR{-{-}{-}}TFF3}
{Highlighting}
{Shaded}
```

Table 21: 4-bit Asynchronous Counter States

Count	Q3	Q2	Q1	Q0
0	0	0	0	0

1	0	0	0	1
2	0	0	1	0
...
14	1	1	1	0
15	1	1	1	1

- **Operation:** Each flip-flop triggers the next when transitioning from 1 to 0
- **Features:** Simple design but suffers from propagation delay (ripple)
- **Applications:** Frequency division, basic counting applications

Mnemonic

“Ripples Up, Each bit triggers Next”

Question 5(c) OR [7 marks]

Describe steps and the need of E-waste Management of Digital ICs.

Solution

Diagram: E-waste Management Cycle

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    C[Collection]{-}{S[Sorting]}
    S{-}{D[Disassembly]}
    D{-}{R[Recycling]}
    R{-}{M[Material Recovery]}
    M{-}{N[New Products]}
    N{-}{C}
{Highlighting}
{Shaded}
```

Table 22: E-waste Management Steps

Step	Description	Importance
Collection	Gathering obsolete ICs	Prevents improper disposal
Sorting	Categorizing by type	Enables efficient processing
Disassembly	Separating components	Facilitates material recovery
Recycling	Processing materials	Reduces environmental impact
Material Recovery	Extracting valuable metals	Conserves resources
Safe Disposal	Handling toxic components	Prevents contamination

- **Need for E-waste Management:**
 - **Environmental Protection:** Prevents toxic substances from leaching into soil/water
 - **Resource Conservation:** Recovers valuable metals like gold, silver, copper
 - **Health Safety:** Reduces exposure to hazardous materials like lead, mercury
 - **Legal Compliance:** Follows regulations regarding electronic waste

Mnemonic

“Collect, Sort, Disassemble, Recycle, Recover, Reuse”