Subject Name Solutions

4321102 – Summer 2024

Semester 1 Study Material

Detailed Solutions and Explanations

Question 1(a) [3 marks]

Convert: $(110101)_2 = (\_\_\_\_\_)_{10} = (\_\_\_\_\_)_8 = (\_\_\_\_\_)_{16}$

Solution

Step-by-step conversion of $(110101)_2$:

Binary $(110101)_2$	**Decimal**	**Octal**	**Hexadecimal**
$1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$	**32+16+0+4+0+1 = 53**	$6 \times 8^1 + 5 \times 8^0 = 48 + 5 = 53$	$3 \times 16^1 + 5 \times 16^0 = 48 + 5 = 35$
$(110101)_2$	$(53)_{10}$	$(65)_8$	$(35)_{16}$

Mnemonic

"Binary Digits Out Here" (BDOH) for Binaryconversion.

Question 1(b) [4 marks]

**Perform: (i) $(11101101)_2 + (10101000)_2 (ii) (11011)_2 * (1010)_2 **$

Solution

Table for binary addition and multiplication:

(i) Binary Addition	**(ii) Binary Multiplication**
11101101 | **11011**	
+ **10101000**	$\times 1010$
———-	——-

```
110010101|         00000
| |                    |        11011
| |                    |       00000
| |                    |      11011
| |                    |
--------             | |
|      11101110
```

Decimal verification:

- 1. $(11101101)_2 = 237, (10101000)_2 = 168, Sum = 405 = (110010101)_2$
- 1. $(11011)_2 = 27, (1010)_2 = 10, Product = 270 = (11101110)_2$

Mnemonic

"Carry Up Makes Sum" for addition and "Shift Left Add Product" for multiplication.

Question 1(c) [7 marks]

(i) Convert: $(48)_{10} = (\_\_\_\_\_)_2 = (\_\_\_\_\_)_8 = (\_\_\_\_\_)_{16}$

Question 1(c) OR [7 marks]

Explain Codes: ASCII, BCD, Gray

Solution

Table of Common Digital Codes:

Code	Description	Example
ASCII (American Standard Code for Information Interchange)	7-bit code representing 128 characters including alphabets, numbers, and special symbols	A = 65 $(1000001)_2$
BCD (Binary Coded Decimal)	Represents each decimal digit (0-9) using 4 bits	42 = 0100 0010
Gray Code	Binary code where adjacent numbers differ by only one bit	(0,1,3,2) = (00,01,11,10)

Diagram: Gray Code Generation:

```
flowchart LR
    A["Binary Code"] {-{-} B["Gray Code"]}
    B {-{-} C["MSB remains same}
    Each bit XORed with previous"]
    D["Binary: 0011"] {-{-} E["Gray: 0010"]}
```

Mnemonic

"Always Binary Generates" - first letter of each code (ASCII, BCD, Gray).

Question 2(a) [3 marks]

Simplify using Boolean Algebra: Y = A B + A' B + A' B' + A B'

Solution

Step-by-step simplification:

Step	Expression	Boolean Law
Y = A B + A' B + A' B' + A B'	Initial expression	-
Y = A(B + B') + A'(B + B')	Factoring	Distributive law
Y = A(1) + A'(1)	Complement law	B + B' = 1
Y = A + A'	Simplification	-
Y = 1	Complement law	A + A' = 1

Mnemonic

"Factor, Simplify, Finish" for Boolean simplification steps.

Question 2(b) [4 marks]

Simplify the following Boolean function using K-map: f(A,B,C,D) = Σm (0,3,4,6,8,11,12)

Solution

K-map Solution:

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

Grouping:

- Group 1: m(0,8) = A'C'D'
- Group 2: m(4,12) = BD'
- Group 3: m(3,11) = CD
- Group 4: m(6) = A'B'CD'

Simplified expression: f(A,B,C,D) = A'C'D' + BD' + CD + A'B'CD'

Mnemonic

"Group Powers Of Two" for K-map grouping strategy.

Question 2(c) [7 marks]

Explain NOR gate as a universal gate with neat diagrams.

Solution

NOR as Universal Gate:

Function	Implementation using NOR	Truth Table
NOT Gate		A
		0
		1
AND Gate		A B
		0 0
		0 1
		1 0
		1 1
OR Gate		A B
		0 0
		0 1
		1 0
		1 1

Diagram: NOR Implementation:

```
flowchart TD
    A["NOT: A {-1{-} A"]}
    B["AND: A {-{-}1{-}{-}|}
           |    1{{-}{-} A•B}
         B {-{-}1{-}{-}|"]}
    C["OR: A {-{-}1{-}{-}|}
          |    |{{-}{-} A+B}
        B {-{-}1{-}{-}|"]}
```

Mnemonic

"NOT AND OR, NOR does more" for remembering NOR gate implementations.

Question 2(a) OR [3 marks]

Draw logic circuit for Boolean expression: Y = (A + B') . (A' + B') . (B + C)

Solution

Logic Circuit Implementation:

```
flowchart TD
    A["A"] {-{-} D["OR"]}
    B["B{"] {-}{-} D}
    D {-{-} G["AND"]}
    A1["A{"] {-}{-} E["OR"]}
    B1["B{"] {-}{-} E}
    E {-{-} G}
    B2["B"] {-{-} F["OR"]}
    C["C"] {-{-} F}
    F {-{-} G}
    G {-{-} Y["Y"]}
```

Truth Table Verification:

- Term 1: (A + B')
- Term 2: (A' + B')
- Term 3: (B + C)
- Output: Y = Term1 • Term2 • Term3

Mnemonic

"Each Term Separately" for breaking complex expressions.

Question 2(b) OR [4 marks]

State De-Morgan's theorems and prove it.

Solution

De-Morgan's Theorems and Proof:

Theorem	Statement	Proof by Truth Table
Theorem 1	(A • B)' = A' + B'	A B 0 0 0 1 1 0 1 1
Theorem 2	(A+B)' = A' • B'	A B 0 0 0 1 1 0 1 1

Diagram: De-Morgan's Law Visualization:

```
flowchart TB
    A["(A•B){ = A+B"] {-}{-} B["Invert Operation}
                              AND  OR
                              Invert Variables"]
    C["(A+B){ = A•B"] {-}{-} D["Invert Operation}
                              OR  AND
                              Invert Variables"]
```

Mnemonic

"Break BAR, Change Operation, Invert Inputs" for applying De-Morgan's law.

Question 2(c) OR [7 marks]

Explain all the Logic Gates with the help of Symbol, Truth table and equation.

Solution

Logic Gates Summary:

Gate	Symbol	Truth Table	Equation	Description
AND		A B	Y	Y = A • B
		0 0	0	
		0 1	0	
		1 0	0	
		1 1	1	
OR		A B	Y	Y = A+B
		0 0	0	
		0 1	1	
		1 0	1	
		1 1	1	
NOT		A	Y	Y = A'
		0	1	
		1	0	
NAND		A B	Y	Y = (A • B)'
		0 0	1	
		0 1	1	
		1 0	1	
		1 1	0	
NOR		A B	Y	Y = (A+B)'
		0 0	1	
		0 1	0	
		1 0	0	
		1 1	0	
XOR		A B	Y	Y = A
		0 0	0	
		0 1	1	
		1 0	1	
		1 1	0	
XNOR		A B	Y	Y = (A)'
		0 0	1	
		0 1	0	
		1 0	0	
		1 1	1	

Mnemonic

"All Operations Need Necessary eXecution" (first letter of each gate - AND, OR, NOT, NAND, NOR, XOR).

Question 3(a) [3 marks]

Briefly explain 4:2 Encoder.

Solution

4-to-2 Encoder Overview:

Function	Description	Truth Table
4:2 Encoder	Converts 4 input lines to 2 output lines	$\mathrm{I}_0 I_1 I_2 I_3$
	Only one input active at a time	1 0 0 0
	Input position encoded in binary	0 1 0 0
		0 0 1 0
		0 0 0 1

Diagram: 4:2 Encoder:

```
flowchart TD
    I0["I_{0"] {-}{-} E["4:2 Encoder"]}
    I1["I_{1"] {-}{-} E}
    I2["I_{2"] {-}{-} E}
    I3["I_{3"] {-}{-} E}
    E {-{-} Y1["Y_{1}"]}
    E {-{-} Y0["Y_{0}"]}
```

Mnemonic

"Input Position Creates Output" for encoder function.

Question 3(b) [4 marks]

Explain 4-bit Parallel adder using full adder blocks.

Solution

4-bit Parallel Adder:

Component	Function
Full Adder	Adds 3 bits (A, B, Carry-in) producing Sum and Carry-out
Parallel Adder	Connects 4 full adders with carry propagation

Diagram: 4-bit Parallel Adder:

```
flowchart LR
    A0["A_{0"] {-}{-} FA0["FA"]}
    B0["B_{0"] {-}{-} FA0}
    C0["C_{0=0"] {-}{-} FA0}
    FA0 {-{-} S0["S_{0}"]}
    FA0 {-{-} "C_{1}" {-}{-} FA1["FA"]}

    A1["A_{1"] {-}{-} FA1}
    B1["B_{1"] {-}{-} FA1}
    FA1 {-{-} S1["S_{1}"]}
    FA1 {-{-} "C_{2}" {-}{-} FA2["FA"]}

    A2["A_{2"] {-}{-} FA2}
    B2["B_{2"] {-}{-} FA2}
    FA2 {-{-} S2["S_{2}"] }
    FA2 {-{-} "C_{3}" {-}{-} FA3["FA"]}

    A3["A_{3"] {-}{-} FA3}
    B3["B_{3"] {-}{-} FA3}
    FA3 {-{-} S3["S_{3}"]}
    FA3 {-{-} C4["C_{4}"]}
```

Mnemonic

"Carry Always Passes Right" for the carry propagation in parallel adder.

Question 3(c) [7 marks]

Describe 8:1 Multiplexer with truth table, equation and circuit diagram.

Solution

8:1 Multiplexer:

Component	Description	Function
8:1 MUX	Data selector with 8 inputs, 3 select lines, 1 output	Selects one of 8 inputs based on select lines

Truth Table:

Select Lines	Output
$S_2S_1S_0$	Y
0 0 0	D_0
0 0 1	D_1
0 1 0	D_2
0 1 1	D_3
1 0 0	D_4
1 0 1	D_5
1 1 0	D_6
1 1 1	D_7

Boolean Equation: $Y = S_2'S_1'S_0'D_0 + S_2'S_1'S_0D_1 + S_2'S_1S_0'D_2 + S_2'S_1S_0D_3 + S_2S_1'S_0'D_4 + S_2S_1'S_0D_5 + S_2S_1S_0'D_6 + S_2S_1S_0D_7$
Diagram: 8:1 MUX:

```
flowchart TD
    D0["D_{0"] {-}{-} MUX["8:1 MUX"]}
    D1["D_{1"] {-}{-} MUX}
    D2["D_{2"] {-}{-} MUX}
    D3["D_{3"] {-}{-} MUX}
    D4["D_{4"] {-}{-} MUX}
    D5["D_{5"] {-}{-} MUX}
    D6["D_{6"] {-}{-} MUX}
    D7["D_{7"] {-}{-} MUX}
    S0["S_{0"] {-}{-} MUX}
    S1["S_{1"] {-}{-} MUX}
    S2["S_{2"] {-}{-} MUX}
    MUX {-{-} Y["Y"]}
```

Mnemonic

“Select Decides Data Output” for multiplexer operation.

Question 3(a) OR [3 marks]

Draw the logic circuit of half Subtractor and explain its working.

Solution

Half Subtractor:

Function	Description	Truth Table
Half Subtractor	Subtracts two bits producing Difference and Borrow	A B
		0 0
		0 1
		1 0
		1 1

Logic Circuit:

```
flowchart TD
    A["A"] {-{-} XOR[""]}
    B["B"] {-{-} XOR}
    XOR {-{-} D["D = A"]}

    A1["A{"] {-}{-} AND["•"]}
    B1["B"] {-{-} AND}
    AND {-{-} Bout["Bout = A•B"]}
```

Equations:

- Difference (D) = A $\oplus B$
- Borrow out (Bout) = A' • B

Mnemonic

"Different Bits Borrow" for half subtractor operation.

Question 3(b) OR [4 marks]

Explain 3:8 Decoder with truth table and circuit diagram.

Solution

3:8 Decoder:

Function	Description	Truth Table (Partial)
3:8 Decoder	Converts 3-bit binary input to 8 output lines Only one output active at a time	A$_2A_1A_0$ 0 0 0 0 0 1 ... 1 1 1

Circuit Diagram:

```
flowchart TD
    A0["A_{0"] {-}{-} Dec["3:8 Decoder"]}
    A1["A_{1"] {-}{-} Dec}
    A2["A_{2"] {-}{-} Dec}
    Dec {-{-} Y0["Y_{0}"]}
    Dec {-{-} Y1["Y_{1}"]}
    Dec {-{-} Y2["Y_{2}"]}
    Dec {-{-} Y3["Y_{3}"]}
    Dec {-{-} Y4["Y_{4}"]}
    Dec {-{-} Y5["Y_{5}"]}
    Dec {-{-} Y6["Y_{6}"]}
    Dec {-{-} Y7["Y_{7}"]}
```

Equations:

- Y$_0 = A_2'A_1'A_0'$
- Y$_1 = A_2'A_1'A_0$
- ...
- Y$_7 = A_2A_1A_0$

Mnemonic

"Binary Input Activates Output" for decoder operation.

Question 3(c) OR [7 marks]

Explain Gray to Binary code converter with truth table, equation and circuit diagram.

Solution

Gray to Binary Converter:

Function	Description	Table: Gray to Binary
Gray to Binary	Converts Gray code to Binary code	Gray
	MSB of binary equals MSB of gray	0000
	Each binary bit is XOR of current gray bit and previous binary bit	0001
		0011
		0010
		0110
		...

Circuit Diagram:

```
flowchart LR
    G3["G_{3"] {-}{-} B3["B_{3}"]}
    G3 {-{-} XOR1[""]}
    G2["G_{2"] {-}{-} XOR1}
    XOR1 {-{-} B2["B_{2}"]}
    XOR1 {-{-} XOR2[""]}
    G1["G_{1"] {-}{-} XOR2}
    XOR2 {-{-} B1["B_{1}"]}
    XOR2 {-{-} XOR3[""]}
    G0["G_{0"] {-}{-} XOR3}
    XOR3 {-{-} B0["B_{0}"]}
```

Equations:

- $B_3 = G_3$
- $B_2 = G_3 \oplus G_2$
- $B_1 = B_2 \oplus G_1$
- $B_0 = B_1 \oplus G_0$

Mnemonic

"MSB Stays, Rest XOR" for Gray to Binary conversion.

Question 4(a) [3 marks]

Explain D flip flop with truth table and circuit diagram.

Solution

D Flip-Flop:

Function	Description	Truth Table
D Flip-Flop	Data/Delay flip-flop	CLK
	Q follows D at clock edge	↑
		↑

Circuit Diagram:

```
flowchart LR
    D["D"] {-{-} FF["D Flip{-}Flop"]}
    CLK["Clock"] {-{-} FF}
    FF {-{-} Q["Q"]}
    FF {-{-} Qnot["Q"]}
```

Characteristic Equation:

- Q(next) = D

Mnemonic

"Data Delays one clock" for D flip-flop operation.

Question 4(b) [4 marks]

Explain working of Master Slave JK flip flop.

Solution

Master-Slave JK Flip-Flop:

Component	Operation	Truth Table
Master	Samples inputs when CLK = 1	J K
Slave	Transfers master output when CLK = 0	0 0
		0 1
		1 0
		1 1

Diagram: Master-Slave JK:

```
flowchart LR
    J["J"] {-{-} Master["Master JK"]}
    K["K"] {-{-} Master}
    CLK["Clock"] {-{-} Master}
    CLK{ {-}{-} Slave["Slave JK"]}
    Master {-{-} Slave}
    Slave {-{-} Q["Q"]}
    Slave {-{-} Q["Q"]}
```

Working:

- **Master stage**: Captures input during clock high
- **Slave stage**: Updates output during clock low
- **Prevents race condition** by separating input capture and output update

Mnemonic

"Master Samples, Slave Transfers" for master-slave operation.

Question 4(c) [7 marks]

Classify Shift Registers with the help of Block diagram and Explain any one of them in detail.

Solution

Shift Register Classification:

Type	Description	Function
SISO	Serial In Serial Out	Data enters and exits serially, bit by bit
SIPO	Serial In Parallel Out	Data enters serially, exits in parallel
PISO	Parallel In Serial Out	Data enters in parallel, exits serially
PIPO	Parallel In Parallel Out	Data enters and exits in parallel

SIPO Shift Register in Detail:

```
flowchart LR
    Din["Data In"] {-{-} FF1["FF_{1}"]}
    FF1 {-{-} FF2["FF_{2}"]}
    FF2 {-{-} FF3["FF_{3}"]}
    FF3 {-{-} FF4["FF_{4}"]}
    CLK["Clock"] {-{-} FF1}
    CLK {-{-} FF2}
    CLK {-{-} FF3}
    CLK {-{-} FF4}
    FF1 {-{-} Q0["Q_{0}"]}
    FF2 {-{-} Q1["Q_{1}"]}
    FF3 {-{-} Q2["Q_{2}"]}
    FF4 {-{-} Q3["Q_{3}"]}
```

Working of SIPO Shift Register:

- **Serial data** enters at Data-In pin, one bit per clock cycle
- **Each flip-flop** passes its content to the next on clock pulse
- **After 4 clock cycles**, 4-bit data is stored in all flip-flops
- **Parallel output** available from Q0-Q3 simultaneously

Timing Diagram for SIPO:

```
Clock   \_| |\_| |\_| |\_| |\_
Data    \_\_\_|  |\_\_\_|  |\_
Q0      \_\_\_|  |\_\_\_|  |\_
Q1      \_\_\_\_\_|  |\_\_\_|
Q2      \_\_\_\_\_\_\_|  |\_\_\_
Q3      \_\_\_\_\_\_\_\_\_|  |\_
```

Mnemonic

"Serial Inputs Parallel Outputs" for SIPO operation.

Question 4(a) OR [3 marks]

Explain SR flip flop with truth table and circuit diagram.

Solution

SR Flip-Flop:

Function	Description	Truth Table
SR Flip-Flop	Set-Reset flip-flop	S R
	Basic memory element	0 0
		0 1
		1 0
		1 1

Circuit Diagram:

```
flowchart LR
    S["S"] {-{-} NOR1["1"]}
    QN["Q{"] {-}{-} NOR1}
    NOR1 {-{-} Q["Q"]}
    R["R"] {-{-} NOR2["1"]}
    Q {-{-} NOR2}
    NOR2 {-{-} QN}
```

Mnemonic

"Set to 1, Reset to 0" for SR flip-flop operation.

Question 4(b) OR [4 marks]

Describe JK flip flop with truth table and circuit diagram.

Solution

JK Flip-Flop:

Function	Description	Truth Table
JK Flip-Flop	Improved SR flip-flop	J K
	Resolves invalid condition	0 0
		0 1
		1 0
		1 1

Circuit Diagram:

```
flowchart LR
    J["J"] {-{-} AND1["•"]}
    Qn["Q{"] {-}{-} AND1}
    AND1 {-{-} OR["1"]}
    K["K"] {-{-} AND2["•"]}
    Q["Q"] {-{-} AND2}
    AND2 {-{-} OR}
    OR {-{-} FF["D FF"]}
    CLK["Clock"] {-{-} FF}
    FF {-{-} Q}
    FF {-{-} Qn}
```

Characteristic Equation:

- Q(next) = J•Q' + K'•Q

Mnemonic

"Jump-Keep-Toggle" for JK flip-flop states (J=1 K=0: Jump to 1,
J=0
K=0: Keep state,
J=1
K=1: Toggle).

Question 4(c) OR [7 marks]

Describe 4-bit Asynchronous UP Counter with truth table and circuit diagram.

Solution

4-bit Asynchronous UP Counter:

Function	Description	Count Sequence
Asynchronous Counter	Also called ripple counter Clock drives only first FF Each FF triggered by previous FF output	0000 → 0001 → 0010 → 0011 0100 → 0101 → 0110 → 0111 1000 → 1001 → 1010 → 1011 1100 → 1101 → 1110 → 1111

Circuit Diagram:

```
flowchart LR
    CLK["Clock"] {-{-} JK1["JK FF_{0}"]}
    J1["J=1"] {-{-} JK1}
    K1["K=1"] {-{-} JK1}
    JK1 {-{-} Q0["Q_{0}"]}
    JK1 {-{-}"Q_{0}"{-}{-} JK2["JK FF_{1}"]}
    J2["J=1"] {-{-} JK2}
    K2["K=1"] {-{-} JK2}
    JK2 {-{-} Q1["Q_{1}"]}
    JK2 {-{-}"Q_{1}"{-}{-} JK3["JK FF_{2}"]}
    J3["J=1"] {-{-} JK3}
    K3["K=1"] {-{-} JK3}
    JK3 {-{-} Q2["Q_{2}"]}
    JK3 {-{-}"Q_{2}"{-}{-} JK4["JK FF_{3}"]}
    J4["J=1"] {-{-} JK4}
    K4["K=1"] {-{-} JK4}
    JK4 {-{-} Q3["Q_{3}"]}
```

Working:
- **First FF** toggles on every clock pulse
- **Second FF** toggles when first FF goes from 1 to 0
- **Third FF** toggles when second FF goes from 1 to 0
- **Fourth FF** toggles when third FF goes from 1 to 0

Mnemonic

"Ripple Carries Propagation Delay" for asynchronous counter operation.

Question 5(a) [3 marks]

Compare following logic families: TTL, CMOS, ECL

Solution

Logic Families Comparison:

Parameter	TTL	CMOS	ECL
Technology	Bipolar transistors	MOSFETs	Bipolar transistors
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	3-15V	-5.2V

Mnemonic

"Technology Controls Many Electrical Characteristics" for comparing logic families.

Question 5(b) [4 marks]

Compare Combinational and Sequential Logic Circuits.

Solution

Combinational vs Sequential Circuits:

Parameter	Combinational Circuits	Sequential Circuits
Output depends on	Current inputs only	Current inputs and previous state
Memory	No memory	Has memory elements
Feedback	No feedback paths	Contains feedback paths
Examples	Adders, MUX, Decoders	Flip-flops, Counters, Registers
Clock	No clock required	Clock often required
Design approach	Truth tables, K-maps	State diagrams, tables

Diagram: Comparison:

```
flowchart TB
    A["Combinational Logic"] {-{-} B["Outputs = f(Current Inputs)"]}
    C["Sequential Logic"] {-{-} D["Outputs = f(Current Inputs, Previous State)"]}
```

Mnemonic

"Current Only vs Memory States" for differentiating combinational and sequential circuits.

Question 5(c) [7 marks]

Define: Fan in, Fan out, Noise margin, Propagation delay, Power dissipation, Figure of merit, RAM

Solution

Digital Electronics Key Definitions:

Term	Definition	Typical Values
Fan-in	Maximum number of inputs a logic gate can handle	TTL: 2-8, CMOS: 100+
Fan-out	Maximum number of gate inputs that can be driven by a single output	TTL: 10, CMOS: 50
Noise margin	Maximum noise voltage that can be added before causing error	TTL: 0.4V, CMOS: 1.5V
Propagation delay	Time taken for change in input to cause change in output	TTL: 10ns, CMOS: 20ns
Power dissipation	Power consumed by gate during operation	TTL: 10mW, CMOS: 0.1mW
Figure of merit	Product of speed and power (lower is better)	TTL: 100pJ, CMOS: 2pJ
RAM	Random Access Memory - temporary storage device	Types: SRAM, DRAM

Diagram: Digital Parameter Relationships:

```
flowchart LR
    A["Lower Propagation Delay"]{-{-}"Increases"{-}{-}B["Speed"]}
    C["Lower Power Dissipation"]{-{-}"Increases"{-}{-}D["Efficiency"]}
    B{-{-}"x"{-}{-}E["Figure of Merit"]}
    D{-{-}"x"{-}{-}E}
```

Mnemonic

"Fast Power Needs Proper Figure Ratings" for remembering the parameter terms.

Question 5(a) OR [3 marks]

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

Solution

E-waste Management for Digital ICs:

Step	Description	Importance
Collection	Separate collection of electronic waste	Prevents improper disposal
Segregation	Separating ICs from other components	Enables targeted recycling
Dismantling	Removal of hazardous parts	Reduces environmental harm
Recovery	Extracting valuable materials (gold, silicon)	Conserves resources
Safe disposal	Proper disposal of non-recyclable parts	Prevents pollution

Need for E-waste Management:

- **Hazardous Materials**: ICs contain lead, mercury, cadmium
- **Resource Conservation**: Recovers precious metals and rare materials
- **Environmental Protection**: Prevents soil and water contamination
- **Health Safety**: Reduces exposure to toxic substances

Mnemonic

"Collection Starts Dismantling Recovery Safely" for e-waste management steps.

Question 5(b) OR [4 marks]

Explain working of Ring Counter with circuit diagram.

Solution

Ring Counter:

Function	Description	Count Sequence
Ring Counter	Circular shift register with single 1 Only one flip-flop is set at any time N flip-flops for N states	1000 → 0100 → 0010 → 0001 → 1000