**Class 12 class**

**Chapter 2**

**Computer hardware**

**1. What is a logic gate? Name the three basic logic gates.**

* **Logic Gate:** A logic gate is an electronic circuit having one or more inputs and only one output. The relationship between the input and output is based on a certain logic.
* **Basic Logic Gates:**
  + AND gate
  + OR gate
  + NOT gate

**2. Which gates implement logical addition, logical multiplication, and complementation?**

* **Logical Addition:** OR gate
* **Logical Multiplication:** AND gate
* **Complementation:** NOT gate

**3. What is the other name of the NOT gate?**

* Inverter

**4. Why are NAND and NOR gates called Universal gates?**

* NAND and NOR gates are called Universal gates because any other logic gate (AND, OR, NOT, XOR, XNOR) can be constructed using only NAND gates or only NOR gates.

**5. Which gates are called Universal gates and why?**

* NAND and NOR gates. As explained in question 4, they can implement any other logic gate.

**6. State the purpose of reducing the switching functions to the minimal form.**

* The purpose is to simplify the Boolean expression, which leads to a simpler and less expensive logic circuit. It reduces the number of gates required, thus reducing cost, size, and power consumption.

**7. Draw a logic circuit diagram using NAND or NOR only to implement the Boolean function F(a, b) = a'b' + ab'**

* **Using NAND gates:**
  1. a' = a NAND a
  2. b' = b NAND b
  3. a'b' = (a NAND a) NAND (b NAND b)
  4. ab' = a NAND (b NAND b)
  5. a'b' + ab' = (a'b') NAND (ab') (followed by a NOT which is a NAND with both inputs tied together.)
* **Logic Circuit Diagram (NAND):**

a --- NAND ---o

|

b --- NAND ---o--- NAND ---o--- NOT --- F

|

a -----------o--- NAND ---o

|

b --- NAND ---o

**8. How does a half adder differ from a full adder?**

* **Half Adder:** Adds two single binary digits and produces a sum and carry output.
* **Full Adder:** Adds three single binary digits (two inputs and a carry-in) and produces a sum and carry-out output.

**9. How is gray code different from normal binary code?**

* **Binary Code:** Each successive number differs by one or more bits.
* **Gray Code:** Each successive number differs by only one bit. This property is useful in reducing errors in digital systems.

**10. How many variables are reduced by a pair, quad, and octet respectively?**

* **Pair:** Reduces 1 variable
* **Quad:** Reduces 2 variables
* **Octet:** Reduces 3 variables

**11. What is an inverted AND gate called? What is an inverted OR gate called?**

* **Inverted AND:** NAND gate
* **Inverted OR:** NOR gate

**12. When does an XOR gate produce a high output? When does an XNOR gate produce a high output?**

* **XOR:** High output when the inputs are different.
* **XNOR:** High output when the inputs are the same.

**13. Write some applications of logic networks.**

* Arithmetic logic units (ALUs)
* Memory circuits
* Control circuits
* Data transmission
* Digital signal processing

**14. How is a half adder different from a full adder?**

* (Same as question 8)

**15. What is the circuit that converts from binary to decimal called?**

* Binary to Decimal Decoder

**16. What is the circuit that converts from decimal to binary called?**

* Decimal to Binary Encoder

**17. How many output lines are needed to convert 1...50 decimal numbers into binary?**

* We need to find the smallest 'n' such that 2^n >= 50.
* 2^5 = 32 (not enough)
* 2^6 = 64 (enough)
* Therefore, 6 output lines are needed.

**18. What is a multiplexer? Where is it used?**

* **Multiplexer:** A multiplexer is a combinational circuit that selects one of several input signals and forwards it to a single output line.
* **Use:** Data selection, communication systems, data routing.

**19. Draw logic circuit diagrams for the following:**

**(i) xy + x'z**

x --- AND ---o

/ |

y ---o |

|

x --- NOT ---o--- AND ---o--- OR --- (xy + x'z)

/

z -----------o

**(ii) (A + B)(B + C)(C + A)**

A ---o--- OR ---o

/ |

B ---o |

|

B ---o--- OR ---o--- AND --- (A + B)(B + C)(C + A)

/ |

C ---o |

|

C ---o--- OR ---o

/

A ---o

**(iii) A'B + BC**

A --- NOT ---o--- AND ---o

/ |

B -----------o |

|

B -----------o--- AND ---o--- OR --- (A'B + BC)

/

C -----------o

**(iv) xyz + x'y'z'**

x --- AND ---o

/ |

y ---o |

/ |

z ---o |

|

x --- NOT ---o--- AND ---o--- OR --- (xyz + x'y'z')

/ |

y --- NOT ---o |

/

z --- NOT ---o

**20. Design a circuit (3 input) which gives a high input when there is an even number of low inputs.**

* **Truth Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **C** | **Output** |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

Export to Sheets

* **Output = A'B'C' + A'BC + AB'C + ABC'**
* **Logic Circuit:** (Similar to XOR with a NOT on the final output)

**21. Design a circuit (3 input) which gives a high input only when there is an even number of low or high inputs.**

* This is the XNOR function.
* **Output = A'B'C' + A'BC + AB'C + ABC**
* **Logic Circuit:** (Similar to XOR with a NOT on the final output)

**22. Design a logic circuit to realize the Boolean function f(x, y) = xy + x'y'.**

* This is the XNOR function.
* **Logic Circuit:** (Similar to XOR with a NOT on the final output)

**23. Draw logic circuits for the following using k-maps:**

**(i) F(A, B, C, D) = Σ(1, 3, 5, 9, 10)**

* **K-Map:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CD\AB** | **00** | **01** | **11** | **10** |
| 00 | 0 | 0 | 0 | 1 (10) |
| 01 | 1 (1) | 1 (3) | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 1 (5) | 0 | 1 (9) |

Export to Sheets

* **Simplified SOP:** A'CD' + A'B'D + AB'CD' + AB'C'D
* **Logic Circuit:** (Draw the circuit based on the SOP expression)

**(ii) F(A, B, C) = Π(0, 2, 4, 5)**

* **K-Map (0s at 0, 2, 4, 5):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **C\AB** | **00** | **01** | **11** | **10** |
| 0 | 0 (0) | 0 (2) | 1 | 1 |
| 1 | 0 (4) | 0 (5) | 1 | 1 |

Export to Sheets

* **Simplified POS:** (B + C)(A + C')
* **Logic Circuit:** (Draw the circuit based on the POS expression)

**(iii) F(A, B, C) = Σ(1, 2, 4, 6, 7)**

* **K-Map:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **C\AB** | **00** | **01** | **11** | **10** |
| 0 | 0 | 1 (2) | 0 | 1 (4) |
| 1 | 1 (1) | 0 | 1 (7) | 1 (6) |

Export to Sheets

* **Simplified SOP:** AC + A'B'C + A'BC'
* **Logic Circuit:** (Draw the circuit based on the SOP expression)

**(iv) F(A, B, C) = Π(1, 3, 5, 7)**

* **K-Map (0s at 1, 3, 5, 7):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **C\AB** | **00** | **01** | **11** | **10** |
| 0 | 1 | 0 (1) | 1 | 1 |
| 1 | 1 | 0 (3) | 0 (7) | 0 (5) |

Export to Sheets

* **Simplified POS:** (B + C)(A + C)
* **Logic Circuit:** (Draw the circuit based on the POS expression)

**24. Draw the logic circuit for this Boolean equation y = A'B'CD' + AB'C'D + ABC'D + ABCD'**

* **Logic Circuit:** (Draw the circuit based on the SOP expression)

**25. Draw the AND-OR circuit for y = A'B'CD + ABC'D' + ABCD'**

* **Logic Circuit:** (Draw the circuit based on the SOP expression)

**26. Derive a Boolean expression for the output F at the network shown below:**

**Network Analysis:**

1. The first gate is a NOR gate with inputs A and B. Its output is A+B​.
2. The second gate is a NOR gate with inputs C and D. Its output is C+D​.
3. The final gate is a NOR gate with inputs A+B​ and C+D​. Its output is A+B​+C+D​​.

**Boolean Expression:**

F=A+B​+C+D​​

**Simplification using De Morgan's Law:**

F=A+B​​⋅C+D​​ F=(A+B)⋅(C+D)

**Therefore, the Boolean expression for F is:**

**F = (A + B)(C + D)**

**27. Convert the above circuit into a NAND-to-NAND logic circuit.**

**Conversion using NAND gates:**

1. **Replace NOR gates with equivalent NAND implementations:**
   * **NOR gate using NAND:** A NOR gate can be implemented using NAND gates as follows:
     1. Invert each input using NAND (inputs tied together).
     2. Take the outputs of the inverters and feed them into a NAND gate.
2. **Apply De Morgan's Law to simplify:**
   * We already found the expression for F is (A + B)(C + D).
   * We can implement this using NAND gates.

**NAND-to-NAND Logic Circuit:**

1. **A + B using NAND gates:**
   * A⋅B=A+B
   * This requires:
     + NOT(A) using NAND(A, A)
     + NOT(B) using NAND(B, B)
     + NAND(NOT(A), NOT(B))
2. **C + D using NAND gates:**
   * C⋅D=C+D
   * This requires:
     + NOT(C) using NAND(C, C)
     + NOT(D) using NAND(D, D)
     + NAND(NOT(C), NOT(D))
3. **(A + B)(C + D) using NAND gates:**
   * (A+B)​+(C+D)​​
   * This requires:
     + NAND((A + B), (C + D))
     + NOT(NAND((A + B), (C + D)))

**NAND Circuit Diagram:**

A --- NAND ---o--- NAND ---o

/

A ---o------

B --- NAND ---o

/

B ---o------

C --- NAND ---o--- NAND ---o

/

C ---o------

D --- NAND ---o

/

D ---o------

o--- NAND ---o--- NOT --- F

/

o--- NAND

**Simplified NAND Circuit Diagram:**

A --- NAND ---o

\

B --- NAND ---o--- NAND ---o--- NAND --- F

/

C --- NAND ---o

\

D --- NAND ---o

**Explanation:**

* The first two NAND gates with their inputs tied together act as NOT gates, inverting A and B.
* The next NAND gate implements A + B.
* The next two NAND gates with their inputs tied together act as NOT gates, inverting C and D.
* The next NAND gate implements C + D.
* The final NAND gate implements (A + B)(C + D).

Let me know if you need any more clarification or assistance!

Can you draw the image

A ---o---¬

| |

| NOR ---o

| | \

B ---o---¬ >--- NOR --- F

/

C ---o---¬

| |

| NOR ---o

| |

D ---o---¬

**NAND Circuit Diagram:**

A --- NAND ---o--- NAND ---o

/

A ---o------

B --- NAND ---o

/

B ---o------

C --- NAND ---o--- NAND ---o

/

C ---o------

D --- NAND ---o

/

D ---o------

o--- NAND ---o--- NOT --- F

/

o--- NAND

**Simplified NAND Circuit Diagram:**

A --- NAND ---o

\

B --- NAND ---o--- NAND ---o--- NAND --- F

/

C --- NAND ---o

\

D --- NAND ---o

**28. Why are NAND and NOR gates more popular?**

* They are universal gates, meaning they can implement any other logic function.
* They are generally faster and cheaper to manufacture compared to other gates.

**29. Draw the logical circuits for the following using NAND gates only:**

**(i) xy + xyz + x'yz**

* **Logic Circuit:** (Draw the circuit using NAND gates based on the expression)

**(ii) ABC + A'B'C + A'BC'**

* **Logic Circuit:** (Draw the circuit using NAND gates based on the expression)

**30. Draw the logical circuits for the following using NOR gates only:**

**(i) (X + Y).(X' + Y).(X + Y')**

* **Logic Circuit:** (Draw the circuit using NOR gates based on the expression)

**(ii) (X + Y + Z).(X + Y' + Z')**

* **Logic Circuit:** (Draw the circuit using NOR gates based on the expression)

**31. Draw the logical circuit for the following function using NAND gates only: F(a, b, c) = Σ(0, 3, 4, 7)**

* **Simplified SOP:** b'c' + bc (from question 30)
* **Logic Circuit:** (Draw the circuit using NAND gates based on the expression)

**32. Draw the simplified logic diagram using only NAND gates to implement the three input function denoted by the expression: F = Σ(0, 1, 2, 3)**

The minterms are: m0 = ABC m1 = ABC m2 = ABC m3 = ABC

The Boolean expression in SOP form is: F=ABC+ABC+ABC+ABC F=AB(C+C)+AB(C+C) F=AB(1)+AB(1) F=AB+AB F=A(B+B) F=A(1) F=A

To implement F=A using only NAND gates:

A ----o----¬ | | | NAND ---- F | | A ----o----¬

**33. Draw a circuit diagram and truth table for an encoder to convert 0-7 decimal numbers into binary.**

A 0-7 decimal to binary encoder has 8 input lines (D0-D7) and 3 output lines (A, B, C) representing the binary equivalent.

**Circuit Diagram:**

D0 ---

D1 ---

D2 ---o-------o--- C

D3 ---o---o---|

D4 ---o---|---|--- B

D5 ---o---|---o---

D6 ---o---|---|---|--- A

D7 ---o-------o---o---

**Boolean Expressions for Outputs:** A=D4+D5+D6+D7 B=D2+D3+D6+D7 C=D1+D3+D5+D7

**Truth Table:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **D7** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** | **A** | **B** | **C** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 (assuming no input active means 0) |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

Export to Sheets

**Note:** Only one input line should be active (HIGH) at a time for a standard encoder. If multiple inputs are HIGH, the output might be ambiguous.

**34. Draw a circuit diagram for 4 by 16 decoder.**

A 4-to-16 decoder has 4 input lines (A, B, C, D) and 16 output lines (Y0-Y15). Each output line corresponds to one of the 24=16 possible input combinations.

The circuit consists of 16 AND gates. Each AND gate has 4 inputs, connected to the input lines (A, B, C, D) or their complements (A,B,C,D) in such a way that each AND gate produces a HIGH output only for a unique combination of the input variables.

**Conceptual Diagram:**

A ----

B ----

C ----

D ----

|

|--- Decoder Logic ---|

| |

| (Combinations of |

| AND gates with |

| inverted/non-inverted|

| inputs) |

| |

-----------------------

/|\ /|\ /|\ ... /|\ /|\ /|\

Y0 Y1 Y2 ... Y13 Y14 Y15

**Each output Yi will be HIGH only when the binary value of ABCD corresponds to i.** For example:

* Y0 = ABCD
* Y1 = ABCD
* Y2 = ABCD
* ...
* Y15 = ABCD

**35. Implement the following Boolean function with an 8x1 multiplexer.** **F(A, B, C, D) = Σ(0, 3, 5, 6, 8, 9, 14, 15)**

An 8x1 multiplexer has 3 select inputs and 8 data inputs. Since the function has 4 variables, we can connect three variables to the select inputs and the fourth variable (or its complement, or constants 0 or 1) to the data inputs. Let's use A, B, C as the select inputs (S2=A, S1=B, S0=C). The data inputs (D0-D7) will be determined by the value of D for each minterm.

**Truth Table and Multiplexer Inputs:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **F** | **Minterm** | **Data Input (D)** |
| 0 | 0 | 0 | 0 | 1 | 0 | D0 = 1 |
| 0 | 0 | 0 | 1 | 0 | 1 |  |
| 0 | 0 | 1 | 0 | 0 | 2 |  |
| 0 | 0 | 1 | 1 | 1 | 3 | D1 = 1 |
| 0 | 1 | 0 | 0 | 0 | 4 |  |
| 0 | 1 | 0 | 1 | 1 | 5 | D2 = 1 |
| 0 | 1 | 1 | 0 | 1 | 6 | D3 = 1 |
| 0 | 1 | 1 | 1 | 0 | 7 |  |
| 1 | 0 | 0 | 0 | 1 | 8 | D4 = 1 |
| 1 | 0 | 0 | 1 | 1 | 9 | D5 = 1 |
| 1 | 0 | 1 | 0 | 0 | 10 |  |
| 1 | 0 | 1 | 1 | 0 | 11 |  |
| 1 | 1 | 0 | 0 | 0 | 12 |  |
| 1 | 1 | 0 | 1 | 1 | 13 |  |
| 1 | 1 | 1 | 0 | 1 | 14 | D6 = 1 |
| 1 | 1 | 1 | 1 | 1 | 15 | D7 = 1 |

Export to Sheets

**Multiplexer Connections:**

* Select Inputs: S2 = A, S1 = B, S0 = C
* Data Inputs: D0 = 1, D1 = 1, D2 = 1, D3 = 1, D4 = 1, D5 = 1, D6 = 1, D7 = 1
* Output: F

**Circuit Diagram:** Draw an 8x1 multiplexer with A, B, C connected to the select lines. Connect all the data input lines (D0 through D7) to a logic HIGH (1). The output of the multiplexer will be F(A, B, C, D) as specified.

**Explanation:** When A, B, C select a particular data input line, if the corresponding minterm is in the function's sum of minterms, the data input is effectively a '1', making F '1'. If the minterm is not present, the data input should be '0'. Let's correct the data inputs based on the value of D.

**Corrected Truth Table and Multiplexer Inputs:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **Minterm** | **F** | **D (for F=1)** | **Multiplexer Input (Di)** |
| 0 | 0 | 0 | 0 | 1 | D (irrelevant) | D0 = 1 |
| 0 | 0 | 1 | 1 | 0 |  |  |
| 0 | 1 | 0 | 2 | 0 |  |  |
| 0 | 1 | 1 | 3 | 1 | D (irrelevant) | D3 = 1 |
| 1 | 0 | 0 | 4 | 0 |  |  |
| 1 | 0 | 1 | 5 | 1 | D (irrelevant) | D5 = 1 |
| 1 | 1 | 0 | 6 | 1 | D (irrelevant) | D6 = 1 |
| 1 | 1 | 1 | 7 | 0 |  |  |

Export to Sheets

This approach of directly connecting constants doesn't work when the function depends on D. We need to express F in terms of A, B, C and D for each select input combination.

Let's use a different approach for the multiplexer inputs:

Consider the cases for each combination of A, B, C:

* **ABC = 000:** F = D (minterm 0) => D0 = D
* **ABC = 001:** F = 0 (no minterm 1 or 2) => D1 = 0
* **ABC = 010:** F = D (minterm 5) => D2 = D
* **ABC = 011:** F = D (minterm 6) => D3 = D
* **ABC = 100:** F = D (minterm 8) => D4 = D
* **ABC = 101:** F = D (minterm 9) => D5 = D
* **ABC = 110:** F = D (minterm 14) => D6 = D
* **ABC = 111:** F = D + D = 1 (minterm 15) => D7 = 1

**Multiplexer Connections (Corrected):**

* Select Inputs: S2 = A, S1 = B, S0 = C
* Data Inputs: D0 = D, D1 = 0, D2 = D, D3 = D, D4 = D, D5 = D, D6 = D, D7 = 1
* Output: F

**Circuit Diagram:** Draw an 8x1 multiplexer. Connect A to S2, B to S1, and C to S0. Connect the data inputs D0-D7 as determined above (using a NOT gate for D). The output will be F.

**36. An 8x1 multiplexer has inputs A, B, C connected to the selection inputs S2, S1, S0 respectively. The data inputs I0 through I7 are as follows: I1 = D, I2 = 1, I3 = D', I5 = 0, I6 = D, I7 = D'. Determine the Boolean function that the multiplexer implements.**

The output F of the multiplexer is given by: F=ABCI0+ABCI1+ABCI2+ABCI3+ABCI4+ABCI5+ABCI6+ABCI7

Substituting the given values of I0-I7 (assuming I0 and I4 are 0 if not specified): F=ABC(0)+ABCD+ABC(1)+ABCD′+ABC(0)+ABC(0)+ABCD+ABCD′ F=ABCD+ABC+ABCD′+ABCD+ABCD′

**Boolean function implemented by the multiplexer: F=ABCD+ABC+ABCD′+ABCD+ABCD′**

**37. Prove that x⊕y=xy+xy​**

Truth Table:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **x** | **y** | **x** | **y​** | **xy** | **xy​** | **xy+xy​** | **x⊕y** |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

Export to Sheets

The truth tables for xy+xy​ and x⊕y are identical.

**Proof using Boolean Algebra:** x⊕y=(x+y)(xy)​ =(x+y)(x+y​) =xx+xy​+yx+yy​ =0+xy​+xy+0 =xy+xy​

**38. Prove that x⊕1​=x⊕0**

LHS: x⊕1​=(x1+x1)​=(x.0+x.1)​=(0+x)​=x=x

RHS: x⊕0=x0+x0=x.1+x.0=x+0=x

Since LHS = RHS = x, the expression is proven.

**39. Show that if xy=0, then x⊕y=x+y.**

x⊕y=xy+xy​ =xy+xy​+xy−xy (Adding and subtracting xy) =y(x+x)+x(y​+y)−xy (Factoring) =y(1)+x(1)−xy =y+x−xy

Given xy=0: x⊕y=x+y−0=x+y

Therefore, if xy=0, then x⊕y=x+y.

**40. Determine the Boolean functions for outputs F and G as a function of our inputs A, B, C, and D in Fig.**

Unfortunately, the "Fig." you are referring to is not provided in the text. To answer this question, I would need the logic circuit diagram showing how A, B, C, and D are connected to produce outputs F and G.

**Please provide the logic circuit diagram or a description of the connections for Figure 40.**

**41. Explain all the logic gates you have learnt so far. Also explain their functioning along with their truth tables.**

**Basic Logic Gates:**

* **AND Gate:**
  + **Symbol:** (Looks like a 'D')
  + **Boolean Expression:** Y=A⋅B or Y=AB
  + **Functioning:** The output Y is HIGH (1) only if both inputs A and B are HIGH (1). Otherwise, the output is LOW (0).
  + **Truth Table:** | A | B | Y | |---|---|---| | 0 | 0 | 0 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 1 |
* **OR Gate:**
  + **Symbol:** (Curved shape)
  + **Boolean Expression:** Y=A+B
  + **Functioning:** The output Y is HIGH (1) if at least one of the inputs A or B is HIGH (1). The output is LOW (0) only if both inputs are LOW (0).
  + **Truth Table:** | A | B | Y | |---|---|---| | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 1 |
* **NOT Gate (Inverter):**
  + **Symbol:** (Triangle with a bubble at the output)
  + **Boolean Expression:** Y=A or Y=A′
  + **Functioning:** The output Y is the logical inverse (complement) of the input A. If A is HIGH (1), Y is LOW (0), and if A is LOW (0), Y is HIGH (1).
  + **Truth Table:** | A | Y | |---|---| | 0 | 1 | | 1 | 0 |

**Universal Logic Gates:**

* **NAND Gate (NOT-AND):**
  + **Symbol:** (AND gate with a bubble at the output)
  + **Boolean Expression:** Y=A⋅B or Y=(AB)′
  + **Functioning:** The output Y is LOW (0) only if both inputs A and B are HIGH (1). Otherwise, the output is HIGH (1). It's the complement of the AND gate.
  + **Truth Table:** | A | B | Y | |---|---|---| | 0 | 0 | 1 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 0 |
* **NOR Gate (NOT-OR):**
  + **Symbol:** (OR gate with a bubble at the output)
  + **Boolean Expression:** Y=A+B​ or Y=(A+B)′
  + **Functioning:** The output Y is HIGH (1) only if both inputs A and B are LOW (0). Otherwise, the output is LOW (0). It's the complement of the OR gate.
  + **Truth Table:** | A | B | Y | |---|---|---| | 0 | 0 | 1 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 0 |

**Exclusive Logic Gates:**

* **XOR Gate (Exclusive OR):**
  + **Symbol:** (OR gate with an extra curved line at the inputs)
  + **Boolean Expression:** Y=A⊕B=AB+AB
  + **Functioning:** The output Y is HIGH (1) if the inputs A and B are different. The output is LOW (0) if the inputs are the same.
  + **Truth Table:** | A | B | Y | |---|---|---| | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 0 |
* **XNOR Gate (Exclusive NOR):**
  + **Symbol:** (XOR gate with a bubble at the output)
  + **Boolean Expression:** Y=A⊙B=A⊕B​=AB+AB
  + **Functioning:** The output Y is HIGH (1) if the inputs A and B are the same. The output is LOW (0) if the inputs are different. It's the complement of the XOR gate.
  + **Truth Table:** | A | B | Y | |---|---|---| | 0 | 0 | 1 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 1 |

**42. Explain an encoder that can convert (0-9) decimal numbers into binary. Also design a decoder that does the exact opposite work i.e. converts binary numbers (0-9) into decimal numbers (0-9).**

**(a) Decimal (0-9) to Binary Encoder:**

A decimal-to-binary encoder has 10 input lines (one for each decimal digit 0-9) and 4 output lines (representing the binary equivalent). We assume that only one input line is active (HIGH) at any given time.

**Logic:** If input Di (where i is the decimal digit) is HIGH, the binary output should be the 4-bit binary representation of i.

**Boolean Expressions for Outputs (A, B, C, D - MSB to LSB):**

* **A (MSB is 1 for 8 and 9):** A=D8+D9
* **B (Next bit is 1 for 4, 5, 6, 7):** B=D4+D5+D6+D7
* **C (Next bit is 1 for 2, 3, 6, 7):** C=D2+D3+D6+D7
* **D (LSB is 1 for 1, 3, 5, 7, 9):** D=D1+D3+D5+D7+D9

**Circuit Diagram:** Would involve OR gates with the appropriate decimal input lines connected to produce the binary outputs according to the expressions above.

**Truth Table:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **D9** | **D8** | **D7** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** | **A** | **B** | **C** | **D** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

Export to Sheets

**(b) Binary (0-9) to Decimal (0-9) Decoder:**

A binary-to-decimal decoder has 4 input lines (representing the BCD code for decimal digits 0-9) and 10 output lines (one for each decimal digit 0-9).

**Logic:** If the binary input (A, B, C, D) represents a decimal digit i (0-9), then the output line Di should be HIGH.

**Boolean Expressions for Outputs:**

* D0=ABCD
* D1=ABCD
* D2=ABCD
* D3=ABCD
* D4=ABCD
* D5=ABCD
* D6=ABCD
* D7=ABCD
* D8=ABCD
* D9=ABCD

The binary inputs 1010 through 1111 are invalid BCD codes, and the outputs for these combinations would ideally be LOW or could be used to indicate an error.

**Circuit Diagram:** Would involve 10 AND gates. Each AND gate has 4 inputs connected to the binary input lines (A, B, C, D) or their complements to decode a specific binary combination and activate the corresponding decimal output line.

**Truth Table:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **D9** | **D8** | **D7** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

Export to Sheets

**43. Design a circuit that converts hexadecimal numbers (0-9, A-F) into their binary equivalents.**

A hexadecimal-to-binary encoder has 16 input lines (one for each hexadecimal digit 0-9 and A-F) and 4 output lines (representing the 4-bit binary equivalent).

**Logic:** If input Hi (where i is the hexadecimal digit) is HIGH, the binary output should be the 4-bit binary representation of i.

**Boolean Expressions for Outputs (A, B, C, D - MSB to LSB):**

* **A (MSB is 1 for 8, 9, A, B, C, D, E, F):** A=H8+H9+HA+HB+HC+HD+HE+HF
* **B (Next bit is 1 for 4, 5, 6, 7, C, D, E, F):** B=H4+H5+H6+H7+HC+HD+HE+HF
* **C (Next bit is 1 for 2, 3, 6, 7, A, B, E, F):** C=H2+H3+H6+H7+HA+HB+HE+HF
* \*\*D (LSB is 1

Please solve 43 to 49

Let's solve questions 43 to 49.

**43. Design a circuit that converts hexadecimal numbers (0-9, A-F) into their binary equivalents.**

A hexadecimal-to-binary encoder has 16 input lines (H0-HF) and 4 output lines (A, B, C, D - MSB to LSB).

**Boolean Expressions for Outputs:**

* **A:** H8+H9+HA+HB+HC+HD+HE+HF
* **B:** H4+H5+H6+H7+HC+HD+HE+HF
* **C:** H2+H3+H6+H7+HA+HB+EF
* **D:** H1+H3+H5+H7+H9+HB+HD+HF

**Circuit Diagram:** This would involve four OR gates. Each OR gate's inputs are connected to the hexadecimal input lines corresponding to the positions of '1' in the binary representation of that hexadecimal digit.

**Truth Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Hex** | **H\_in** | **A** | **B** | **C** | **D** |
| 0 | H0=1 | 0 | 0 | 0 | 0 |
| 1 | H1=1 | 0 | 0 | 0 | 1 |
| 2 | H2=1 | 0 | 0 | 1 | 0 |
| 3 | H3=1 | 0 | 0 | 1 | 1 |
| 4 | H4=1 | 0 | 1 | 0 | 0 |
| 5 | H5=1 | 0 | 1 | 0 | 1 |
| 6 | H6=1 | 0 | 1 | 1 | 0 |
| 7 | H7=1 | 0 | 1 | 1 | 1 |
| 8 | H8=1 | 1 | 0 | 0 | 0 |
| 9 | H9=1 | 1 | 0 | 0 | 1 |
| A | HA=1 | 1 | 0 | 1 | 0 |
| B | HB=1 | 1 | 0 | 1 | 1 |
| C | HC=1 | 1 | 1 | 0 | 0 |
| D | HD=1 | 1 | 1 | 0 | 1 |
| E | HE=1 | 1 | 1 | 1 | 0 |
| F | HF=1 | 1 | 1 | 1 | 1 |

Export to Sheets

**44. What is an Encoder? Draw the Encoder circuit to convert hexadecimal numbers to binary numbers.**

An **Encoder** is a combinational logic circuit that converts a set of active input signals into a coded output signal. Typically, it has 2n input lines and n output lines. Only one input line is usually active at a time, and the output produces the binary code corresponding to the active input.

The circuit diagram for a hexadecimal-to-binary encoder would consist of four OR gates, as described in the solution to question 43. The inputs to each OR gate would be the hexadecimal input lines corresponding to the '1' bits in the respective binary output position.

**45. What is a Decoder? How is it different from an encoder? Draw the logic circuit for a 2 to 4 decoder and explain its working.**

A **Decoder** is a combinational logic circuit that converts a coded input signal into a set of active output signals. It typically has n input lines and 2n output lines. For each unique combination of the input, only one output line is active (HIGH).

**Difference between Encoder and Decoder:**

* **Encoder:** Converts a single active input into a coded output. Fewer outputs than inputs (n outputs for 2n inputs).
* **Decoder:** Converts a coded input into a single active output. More outputs than inputs (2n outputs for n inputs).

**Logic Circuit for a 2 to 4 Decoder:**

A 2-to-4 decoder has 2 input lines (A, B) and 4 output lines (Y0​​,Y1​​,Y2​​,Y3​​). Here, we'll consider active LOW outputs, which is common for decoders.

A --- NOT ---o--- AND --- /Y0

|

B --- NOT ---o---

A --- NOT ---o--- AND --- /Y1

|

B -----------o---

A -----------o--- AND --- /Y2

|

B --- NOT ---o---

A -----------o--- AND --- /Y3

|

B -----------o---

**Boolean Expressions for Outputs (Active LOW):**

* Y0​​=AB
* Y1​​=AB
* Y2​​=AB
* Y3​​=AB

**Working:**

* **Input AB = 00:** A=1,B=1. AND gate for Y0​​ gets inputs 1 and 1, so Y0​​=1 (LOW). All other AND gates will have at least one 0 input, making their outputs 0 (HIGH).
* **Input AB = 01:** A=1,B=1. AND gate for Y1​​ gets inputs 1 and 1, so Y1​​=1 (LOW). All other outputs will be HIGH.
* **Input AB = 10:** A=1,B=1. AND gate for Y2​​ gets inputs 1 and 1, so Y2​​=1 (LOW). All other outputs will be HIGH.
* **Input AB = 11:** A=1,B=1. AND gate for Y3​​ gets inputs 1 and 1, so Y3​​=1 (LOW). All other outputs will be HIGH.

For each input combination, only one output line goes LOW, indicating the decoded binary value.

**46. If you are given a truth table for a binary addition as shown below, can you design an adder circuit for it? If yes, draw the logic circuit for it designed by you.**

The truth table for a binary addition (likely a half-adder, as only two inputs are shown):

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **Sum** | **Carry** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

Export to Sheets

**Yes, we can design an adder circuit for this truth table.** This truth table represents a **half-adder**.

**Boolean Expressions for Outputs:**

* **Sum (S):** S=AB+AB=A⊕B (XOR gate)
* **Carry (C):** C=AB (AND gate)

**Logic Circuit Diagram (Half-Adder):**

A --- XOR --- Sum (S)

/

B --

A --- AND --- Carry (C)

/

B --

**47. A comparator is a circuit that compares two numbers. If you are given a truth table for a comparator that compares two 1-bit numbers A and B and gives the output C as 1 if A is greater than or equal to B, draw the logic circuit for the comparator designed by you.**

Truth Table for A ≥ B:

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **C (A ≥ B)** |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Export to Sheets

**Boolean Expression for Output C:**

From the truth table, C is 1 when:

* A=0, B=0 (AB)
* A=1, B=0 (AB)
* A=1, B=1 (AB)

So, C=AB+AB+AB C=B(A+A)+AB C=B(1)+AB C=B+AB

Alternatively, we can observe that C is 0 only when A=0 and B=1. Therefore, C=AB C=(AB)​=A+B

Let's verify if B+AB=A+B: B+AB=(B+A)(B+B)=(A+B)(1)=A+B

**Logic Circuit Diagram (Comparator A ≥ B):**

B --- NOT ---o--- OR --- C

/

A ---o----------

**48. What is a parity checker? How is it different from a parity generator? Draw the logic circuit for a 3-bit even parity generator.**

A **Parity Checker** is a combinational logic circuit that checks whether the number of 1s in a given binary input is even or odd. It produces an output indicating the parity.

A **Parity Generator** is a combinational logic circuit that adds an extra bit (the parity bit) to a binary input to make the total number of 1s (including the parity bit) either even or odd, depending on the type of parity desired.

**Difference:**

* **Parity Generator:** Adds a parity bit to make the parity of the data even or odd.
* **Parity Checker:** Checks the parity of the received data (including the parity bit) to detect errors.

**Logic Circuit for a 3-bit Even Parity Generator:**

A 3-bit even parity generator takes 3 input bits (A, B, C) and produces one parity bit (P) such that the total number of 1s (A, B, C, P) is even.

Truth Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **Number of 1s** | **P (Even Parity)** |
| 0 | 0 | 0 | 0 (even) | 0 |
| 0 | 0 | 1 | 1 (odd) | 1 |
| 0 | 1 | 0 | 1 (odd) | 1 |
| 0 | 1 | 1 | 2 (even) | 0 |
| 1 | 0 | 0 | 1 (odd) | 1 |
| 1 | 0 | 1 | 2 (even) | 0 |
| 1 | 1 | 0 | 2 (even) | 0 |
| 1 | 1 | 1 | 3 (odd) | 1 |

Export to Sheets

Boolean Expression for P: P=ABC+ABC+ABC+ABC This is the expression for a 3-input XOR gate. P=A⊕B⊕C

**Logic Circuit Diagram (3-bit Even Parity Generator):**

A --- XOR ---o--- XOR --- P

/ /

B -- /

/

C -----------

**49. If you are given a truth table for a 3-bit even parity checker, draw the logic circuit for it.**

A 3-bit even parity checker takes 3 input bits (A, B, C) and produces an output (Error) that is HIGH (1) if the number of 1s in the input is odd (indicating an error in even parity).

Truth Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **Number of 1s** | **Error (Odd Parity)** |
| 0 | 0 | 0 | 0 (even) | 0 |
| 0 | 0 | 1 | 1 (odd) | 1 |
| 0 | 1 | 0 | 1 (odd) | 1 |
| 0 | 1 | 1 | 2 (even) | 0 |
| 1 | 0 | 0 | 1 (odd) | 1 |
| 1 | 0 | 1 | 2 (even) | 0 |
| 1 | 1 | 0 | 2 (even) | 0 |
| 1 | 1 | 1 | 3 (odd) | 1 |

Export to Sheets

Boolean Expression for Error: Error = ABC+ABC+ABC+ABC This is the expression for a 3-input XOR gate. Error = A⊕B⊕C

**Logic Circuit Diagram (3-bit Even Parity Checker):**

A --- XOR ---o--- XOR --- Error

/ /

B -- /

/

C -----------

If the output 'Error' is 0, the parity is even (no error detected). If the output 'Error' is 1, the parity is odd (error detected).

**50. What is a half adder? Write the truth table and derive an SOP expression for sum and carry for a half adder.**

A **half adder** is a combinational logic circuit that adds two single-bit binary numbers. It has two inputs (A and B) and two outputs: Sum (S) and Carry (C).

**Truth Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **A (Augend)** | **B (Addend)** | **Sum (S)** | **Carry (C)** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

Export to Sheets

**SOP Expressions:**

* **Sum (S):** Sum is 1 when A=0, B=1 OR A=1, B=0. S=AB+AB=A⊕B
* **Carry (C):** Carry is 1 when A=1 AND B=1. C=AB

**51. A full adder is a combinational circuit that adds 3 bits, taking into account that 1 may have been carried from a fuller adder in a lower significant stage. Following is the truth table for the full adder. It has 3 inputs: X, Y, Z and has two outputs: Carry (C) and Sum (S).**

**(a) Write the SOP expressions for sum (S) and carry (C).**

From the provided truth table:

* **Sum (S) is 1 for minterms:** 1, 2, 4, 7 S=XYZ+XYZ+XYZ+XYZ
* **Carry (C) is 1 for minterms:** 3, 5, 6, 7 C=XYZ+XYZ+XYZ+XYZ

**(b) Reduce the SOP expressions for sum and carry if possible, using Karnaugh's map.**

**Karnaugh Map for Sum (S):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **YZ\X** | **00** | **01** | **11** | **10** |
| 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |

Export to Sheets

Grouping the 1s, we get four single groupings: S=XYZ+XYZ+XYZ+XYZ=X⊕Y⊕Z The sum expression cannot be simplified further in terms of product terms.

**Karnaugh Map for Carry (C):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **YZ\X** | **00** | **01** | **11** | **10** |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 |

Export to Sheets

Grouping the 1s:

* Group of two (vertical): YZ=11 => XY
* Group of two (horizontal): X=1, Z=1 => XZ
* Group of two (horizontal): Y=1, Z=1 => YZ

C=XY+XZ+YZ

**(c) Draw the logic gate circuit for the reduced expressions for sum and carry if it can be reduced. Otherwise draw the logic gate circuit for the original expression using AND, OR gates. You may use gates with more than two inputs. Assume that variables and their complements are available as inputs.**

**Logic Circuit for Sum (S) = X ⊕ Y ⊕ Z:**

This requires two XOR gates.

X --- XOR ---o--- XOR --- S

/ /

Y -- /

/

Z -----------

**Logic Circuit for Carry (C) = XY + XZ + YZ:**

This requires three AND gates (two-input) and one OR gate (three-input).

X --- AND ---o

/

Y --

X --- AND ---o

/

Z --

Y --- AND ---o

/

Z --

o--- OR --- C

/

o

/

o

**52. (a) What do you understand by a multiplexer? State any one application of a multiplexer.**

A **multiplexer (MUX)**, also known as a data selector, is a combinational logic circuit that selects one of several input signals and forwards the selected input to a single output line. The selection is controlled by a set of select input lines.

**One Application of a Multiplexer:**

* **Data Routing:** Multiplexers can be used to route data from multiple sources to a single destination based on the select inputs. This is useful in communication systems and computer architectures for sharing resources.

**(b) Draw the logic diagram and function table for 4x1 multiplexer and explain its working.**

**Logic Diagram of a 4x1 Multiplexer:**

A 4x1 MUX has 4 data inputs (I0, I1, I2, I3), 2 select inputs (S1, S0), and 1 output (Y).

I0 --- AND (S1', S0') ---o

I1 --- AND (S1', S0) ---o

I2 --- AND (S1, S0') ---o--- OR --- Y

I3 --- AND (S1, S0) ---o

**Function Table (Truth Table):**

|  |  |  |  |
| --- | --- | --- | --- |
| **S1** | **S0** | **Output (Y)** | **Selected Input** |
| 0 | 0 | I0 | I0 |
| 0 | 1 | I1 | I1 |
| 1 | 0 | I2 | I2 |
| 1 | 1 | I3 | I3 |

Export to Sheets

**Working:**

The select inputs (S1, S0) determine which of the four AND gates is enabled. Each AND gate is connected to one of the data inputs and a unique combination of the select inputs (either in their true or complemented form).

* **When S1=0 and S0=0:** The top AND gate (connected to I0, S1, S0) is enabled, and its output is I0. The other AND gates are disabled (their output is 0). The OR gate outputs Y = I0 + 0 + 0 + 0 = I0.
* **When S1=0 and S0=1:** The second AND gate (connected to I1, S1, S0) is enabled, and its output is I1. The OR gate outputs Y = 0 + I1 + 0 + 0 = I1.
* **When S1=1 and S0=0:** The third AND gate (connected to I2, S1, S0) is enabled, and its output is I2. The OR gate outputs Y = 0 + 0 + I2 + 0 = I2.
* **When S1=1 and S0=1:** The bottom AND gate (connected to I3, S1, S0) is enabled, and its output is I3. The OR gate outputs Y = 0 + 0 + 0 + I3 = I3.

Thus, the multiplexer selects one of the data inputs based on the binary value of the select inputs and routes it to the output.

**53. How are decoders different from encoders? Where are they used?**

(Answered in question 45)

**Where are Decoders Used?**

* **Memory Address Decoding:** Selecting specific memory locations in a memory system.
* **Instruction Decoding:** Identifying the operation to be performed by a processor based on the instruction opcode.
* **Data Routing:** Directing data to specific output lines.
* **Seven-Segment Display Drivers:** Converting binary codes to activate the segments of a seven-segment display to show decimal digits.
* **Control Signal Generation:** Generating specific control signals based on a coded input.

**54. Using NOR gates only, draw AND, OR and NOT gate.**

* **NOT Gate using NOR:** Connect the input to both inputs of a NOR gate.
* A ---o---¬
* | |
* | NOR --- NOT(A)
* | |
* A ---o---¬
* **OR Gate using NOR:** Use a NOR gate directly.
* A ---o---¬
* | |
* | NOR --- OR(A, B)
* | |
* B ---o---¬
* **AND Gate using NOR:** Use a NOR gate followed by two NOT gates (implemented using NOR gates as shown above) on the inputs.
* A ---o---¬
* | |
* | NOR ---o---¬
* | | | |
* B ---o---¬ | NOR --- AND(A, B)
* | |
* A ---o---¬
* | |
* | NOR
* | |
* B ---o---¬

**55. Draw the truth table and logic circuit diagram for converting a decimal number to binary. Explain the working principle with an example.**

This question is a bit ambiguous. Converting a single decimal digit to binary is a type of encoding (BCD encoder for 0-9). Converting a multi-digit decimal number to its binary equivalent is a more complex process typically done algorithmically or with specialized hardware for larger numbers.

**Assuming the question refers to encoding a single decimal digit (0-9) to BCD:**

(Truth table and logic circuit would be similar to the encoder discussed in question 42, part a).

**Working Principle with an Example (Decimal 6 to BCD):**

When the input for decimal '6' is HIGH, the logic circuit (OR gates as defined by the Boolean expressions) will activate the output lines corresponding to the binary code for 6, which is 0110. So, the output lines would be A=0, B=1, C=1, D=0.

**56. (a) State the difference between an encoder and a multiplexer circuit.**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Encoder** | **Multiplexer** |
| Function | Converts active input to a coded output | Selects one of several inputs to a single output |
| Inputs | 2n (typically only one active) | n data inputs, log2​(n) select inputs |
| Outputs | n (coded representation of input) | 1 (selected input) |
| Purpose | Data compression, code generation | Data selection, routing |

Export to Sheets

**(b) Using NOR gates only, show how to realize the Boolean function Y=AB+C. Show how it is equivalent to the original circuit.**

**Realization using NOR gates:**

1. **Implement AND (AB) using NOR gates:** (As shown in question 54)
2. **Implement NOT (C) using NOR gates:** (As shown in question 54)
3. **Implement OR (AB+C) using NOR gates:** (As shown in question 54, using the outputs of the AND and NOT implementations as inputs to the final NOR gate).

**Circuit Diagram using NOR gates:**

A ---o---¬

| |

| NOR ---o---¬

| | | |

B ---o---¬ | NOR --- Output (AB + C')'

| |

A ---o---¬

| |

| NOR

| |

B ---o---¬

C ---o---¬

| |

| NOR --- NOT(C) ---o---¬

| | | |

C ---o---¬ | NOR --- Output (AB + C')'' = AB + C'

| |

(Output of AND) ------o---¬

| |

| NOR

| |

(Output of NOT) ------o---¬

**Equivalence:**

The circuit using NOR gates implements the following: NOT( NOT(A NOR B) NOR NOT(C) ) NOT( NOT( NOT(AB) ) NOR C' ) NOT( AB NOR C' ) NOT( NOT(AB + C') ) AB + C'

This is equivalent to the original Boolean function Y=AB+C.

**57. Why is the NOR gate regarded as a Universal Gate? Draw the logic gate symbol and make the truth table for the two input NOR gate.**

(Answered in question 4)

**Logic Gate Symbol for Two-Input NOR Gate:**

A ---o---¬

| |

| NOR --- Y

| |

B ---o---¬

**Truth Table for Two-Input NOR Gate:**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Y (A+B​)** |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

**58. The National College of Journalism is offering courses in three different categories of journalism, which are the print, the web and the broadcasting media.**

**A student is eligible to apply if he/she satisfies any one of the following conditions:**

* The student is a graduate in any discipline with an aggregate percentage of 75 or above and with a record of literary skills. **OR**
* The student is a graduate in Mass Communication with an aggregate percentage of 75 or above. **OR**
* The student is a graduate in Mass Communication with an aggregate percentage of 50 or above and has a service of more than 10 years in a media organization.

**The inputs are:**

* A: Graduate in any discipline
* B: Graduate in Mass Communication
* C: Aggregate percentage of 75 and above
* D: Aggregate percentage of 50 or above
* E: Record of literary skills
* F: Service of more than 10 years in a media organization

**Output:** X: Denotes eligible for the course (1 indicates YES and 0 indicates NO in all cases)

**(a) Draw the truth tables for the inputs and outputs given above and write the SOP expression for X(A, B, C, D, E, F).**

To draw the truth table, we need to consider all possible combinations of the six input variables (A, B, C, D, E, F), which would result in 26=64 rows. Then, for each row, we would evaluate the eligibility conditions to determine the output X.

Let's directly write the SOP expression for X based on the given conditions:

Eligibility Condition 1: (Graduate in any discipline AND Percentage ≥ 75 AND Literary skills) Term 1: A⋅C⋅E

Eligibility Condition 2: (Graduate in Mass Communication AND Percentage ≥ 75) Term 2: B⋅C

Eligibility Condition 3: (Graduate in Mass Communication AND Percentage ≥ 50 AND Service > 10 years) Term 3: B⋅D⋅F

The student is eligible if any of these conditions are met (OR operation). Therefore, the SOP expression for X is:

**X=(A⋅C⋅E)+(B⋅C)+(B⋅D⋅F)**

**(b) Reduce X(A, B, C, D, E, F) using Karnaugh's map.**

A Karnaugh map with 6 variables would be quite complex (64 cells). Reducing this expression algebraically might be more practical here.

X=ACE+BC+BDF

There are no obvious direct algebraic simplifications possible between these terms without more specific relationships or redundancies implied in the problem statement (which are not explicitly given). For example, we don't know if being a graduate in Mass Communication (B) automatically implies being a graduate in any discipline (A) or vice-versa. Assuming these are independent conditions, the SOP expression is already in a relatively simplified form. Further reduction would likely require additional information or constraints.

**(c) Draw the logic gate diagram for the reduced SOP expression. Assume that the variables and their complements are available as inputs.**

Based on the SOP expression X=ACE+BC+BDF, the logic gate diagram would be:

1. **Three AND gates:**
   * AND gate 1 with inputs A, C, E. Output: ACE
   * AND gate 2 with inputs B, C. Output: BC
   * AND gate 3 with inputs B, D, F. Output: BDF
2. **One OR gate:**
   * OR gate with inputs from the outputs of the three AND gates (ACE, BC, BDF). Output: X

**Logic Circuit Diagram:**

A --- AND ---o

/

C ---o

/

E ---o

B --- AND ---o

/

C ---o

B --- AND ---o

/

D ---o

/

F ---o

o--- OR --- X

/

o

/

o

**65. A provisional store announces a special discount on all its products as a festival offer to only those who satisfy any of the following conditions:**

* He/she is an employee of the store and has a service of more than 10 years. **OR**
* He/she is a regular customer of the store whose age is less than 65 years and is not an employee of the store. **OR**
* He/she is a senior citizen but not a regular customer of the store.

**The inputs are:**

* E: Employee of the store
* R: Regular customer of the store
* S: Service of the employee is more than 10 years
* A: Age of the customer is less than 65 years
* C: Senior citizen of 65 years or above

**Output:** X: Denotes eligible for discount (1 indicates YES and 0 indicates NO in all cases)

**(a) Draw the truth tables for the inputs and outputs given above and write the SOP expression for X(E, R, S, A, C).**

Again, a full truth table with 5 input variables would have 25=32 rows. Let's directly write the SOP expression for X:

Eligibility Condition 1: (Employee AND Service > 10 years) Term 1: E⋅S

Eligibility Condition 2: (Regular customer AND Age <65 AND NOT Employee) Term 2: R⋅A⋅E

Eligibility Condition 3: (Senior citizen AND NOT Regular customer) Term 3: C⋅R

The customer is eligible if any of these conditions are met (OR operation). Therefore, the SOP expression for X is:

**X=(E⋅S)+(R⋅A⋅E)+(C⋅R)**

**(b) Reduce X(E, R, S, A, C) using Karnaugh's map.**

A Karnaugh map with 5 variables (32 cells) would be complex to draw and reduce here. Let's try algebraic simplification.

X=ES+RAE+CR

There are no immediate obvious algebraic simplifications. The terms involve different combinations of variables and their complements. Without further information about potential dependencies between these conditions, this SOP expression is likely in a reasonably simplified form.

**(c) Draw the logic gate diagram for the reduced SOP expression. Assume that the variables and their complements are available as inputs.**

Based on the SOP expression X=ES+RAE+CR, the logic gate diagram would be:

1. **Three AND gates:**
   * AND gate 1 with inputs E, S. Output: ES
   * AND gate 2 with inputs R, A, E (requires a NOT gate for E). Output: RAE
   * AND gate 3 with inputs C, R (requires a NOT gate for R). Output: CR
2. **One OR gate:**
   * OR gate with inputs from the outputs of the three AND gates (ES, RAE, CR). Output: X

**Logic Circuit Diagram:**

E --- AND ---o

/

S ---o

R --- AND ---o

/

A ---o

/

E --- NOT ---o

C --- AND ---o

/

R --- NOT ---o

o--- OR --- X

/

o

/

o

**59. (a) Draw the truth tables for the inputs and outputs given above and write the SOP expression for R(A, B, C, D).**

Unfortunately, the "inputs and outputs given above" are not visible in the provided image. To answer this question, I need the truth table showing the relationship between inputs A, B, C, D and output R.

**Please provide the truth table for R(A, B, C, D).**

**(b) Reduce R(A, B, C, D) using Karnaugh's map.**

This step can only be performed once the truth table and the initial SOP expression for R are available from part (a).

**(c) Draw a logic gate circuit for the reduced SOP expression for R(A, B, C, D) using AND & OR gates. You may use gates with two or more inputs. Assume that the variables and their complements are available as inputs.**

This step depends on the reduced SOP expression obtained in part (b).

**60. (a) What is a decoder? Draw the truth table and a logic circuit diagram for a 2 by 4 decoder.**

(Answered in question 45)

**(b) What is a Half Adder? Draw the truth table, derive its Boolean expression and draw a logic diagram for Half Adder.**

(Answered in question 50)

**(c) What is an XOR gate? Draw a truth table representing a 2 input XOR operation. Derive its logic gate diagram.**

(Answered in question 41 for truth table and logic function. Logic diagram also shown there.)

**(d) Draw the logic gate diagram for the Boolean function F(A, B, C) = (A' + B').(B' + C') making use of NOR gates only.**

1. **Implement OR gates using NOR gates:** (As shown in question 54)
   * A′+B′ can be implemented by NOR(A′,B′) = NOR(A, B).
   * B′+C′ can be implemented by NOR(B′,C′) = NOR(B, C).
2. **Implement AND gate using NOR gates:** (As shown in question 54) The function is effectively (A′+B′)⋅(B′+C′). Let X=A′+B′ and Y=B′+C′. We need to implement X⋅Y using NOR gates. This is equivalent to X⋅Y=X+Y​.

**Circuit Diagram using NOR gates:**

A ---o---¬

| |

| NOR --- X'

| |

B ---o---¬

B ---o---¬

| |

| NOR --- Y'

| |

C ---o---¬

X' ---o---¬

| |

| NOR --- F

| |

Y' ---o---¬

**Explanation:**

* The first NOR gate implements A′+B′, and its output is A′+B′​.
* The second NOR gate implements B′+C′, and its output is B′+C′​.
* The final NOR gate takes these two outputs as inputs: F=A′+B′​+B′+C′​​ Using De Morgan's Law: F=(A′+B′′)⋅(B′+C′′)=(A′+B)⋅(B′+C)

There seems to be a misunderstanding in directly replacing OR with NOR. Let's redo the NOR gate implementation.

**Corrected NOR Gate Implementation for F(A, B, C) = (A' + B').(B' + C'):**

1. **Implement NOT gates:**
   * A′: A connected to both inputs of a NOR gate.
   * B′: B connected to both inputs of a NOR gate.
   * C′: C connected to both inputs of a NOR gate.
2. **Implement OR gates:**
   * A′+B′: Output of NOT(A) and NOT(B) connected to a NOR gate, then inverted (another NOT gate).
   * B′+C′: Output of NOT(B) and NOT(C) connected to a NOR gate, then inverted (another NOT gate).
3. **Implement AND gate:** The outputs of the two OR implementations are connected to a NOR gate, then inverted (another NOT gate).

**Circuit Diagram (Corrected):**

A ---o---¬

| |

| NOR --- A' ---o---¬

| | | |

A ---o---¬ | NOR ---o---¬

| | | |

B ---o---¬ | | | NOR --- F

| | | | | |

| NOR --- B' ---o---¬ | |

| | | |

B ---o---¬ o---¬

| |

C ---o---¬ | NOR

| | | |

| NOR --- C' ---o-------------

| |

C ---o---¬

**Explanation:**

* First two NORs: Generate A′ and B′.
* Third NOR: Implements A′+B′, output is A′+B′​.
* Fourth and fifth NORs: Generate B′ and C′.
* Sixth NOR: Implements B′+C′, output is B′+C′​.
* Seventh NOR: Takes A′+B′​ and B′+C′​ as input, output is A′+B′​+B′+C′​​.
* Eighth NOR: Inverts the output of the seventh NOR, giving F=A′+B′​+B′+C′​​​=A′+B′​​⋅B′+C′​​=(A′+B′)⋅(B′+C′).

**61. Name the Universal Gates.**

NAND and NOR gates are the Universal Gates.

**62. Draw the truth table and logic circuit diagram for a Hexadecimal to Binary encoder.**

(Answered in question 43)

**63. State the dual theorem and also draw the logic gate diagram for the dual expression obtained using NOR gate only.**

**Dual Theorem:** The dual of a Boolean expression is obtained by interchanging AND and OR operators, and interchanging 0 and 1. The dual of an expression is equal to the complement of the function represented by the original expression.

**Given expression:** F=A+B⋅C

**Dual Expression (F\_D):** Interchange + and ⋅: FD​=A⋅(B+C)

**Logic Gate Diagram for the Dual Expression using NOR gates only:**

1. **Implement NOT gates:** For B.
2. **Implement OR gate (B+C) using NOR gates:** (As shown in question 54)
3. **Implement AND gate (A⋅(B+C)) using NOR gates:** (As shown in question 54, using A and the output of the OR implementation as inputs).

**Circuit Diagram using NOR gates:**

B ---o---¬

| |

| NOR --- B'

B' ---o---¬

| |

| NOR --- (B' + C)'

| |

C ---o---¬

A ---o---¬

| |

| NOR ---o---¬

| | | |

(B'+C)'---o---¬ | NOR --- F\_D'

| |

A ---o---¬

| |

| NOR

| |

(B'+C)'---o---¬

F\_D' ---o---¬

| |

| NOR --- F\_D

| |

F\_D' ---o---¬

**Explanation:**

* First NOR: Generates B.
* Second NOR: Implements B+C, output is B+C​.
* Third NOR structure: Implements AND of A and B+C​, output is A⋅B+C​​.
* Final NOR structure: Inverts the previous output to get FD​=A⋅B+C​=A⋅(B+C).

**64. (a) F(P, Q, R, S) = P'Q'R'S' + P'Q'RS' + P'QRS' + PQ'RS' + PQ'R'S' + PQ'RS + PQR'S' + PQRS'**

**Use Karnaugh's map to reduce the function F using the SOP form. Draw a logic gate diagram for the reduced SOP form. You may use gate with more than two inputs. Assume that the variables and their complements are available as inputs.**

**Karnaugh Map:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RS\PQ** | **00** | **01** | **11** | **10** |
| 00 | 1 | 0 | 1 | 1 |
| 01 | 1 | 1 | 1 | 1 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

Export to Sheets

**Grouping the 1s:**

* **Quad:** Top row (RS=00) excluding PQ=01: S′R′Q′​+S′R′PQ′​+S′R′PQ′
* **Quad:** Second row (RS=01): SR′P′Q′​+SR′P′Q+SR′PQ+SR′PQ′​

Let's redraw the K-map for clarity:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RS\PQ** | **00 (P'Q')** | **01 (P'Q)** | **11 (PQ)** | **10 (PQ')** |
| 00 | 1 | 0 | 1 | 1 |
| 01 | 1 | 1 | 1 | 1 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

Export to Sheets

**Grouping:**

* **Horizontal Quad:** Bottom row (RS=01) => S
* **Vertical Pair:** Left column (PQ=00), top two cells => PQ​R′
* **Vertical Pair:** Rightmost column (PQ=10), top two cells => PQ​R′
* **Horizontal Pair:** Top row (RS=00), first and third columns => R′S′Q′

**Reduced SOP Expression:** F=S+PQ​R′+PQ​R′+R′S′Q′ F=S+Q​R′(P+P)+R′S′Q′ F=S+Q​R′+R′S′Q′

**Logic Gate Diagram:**

S --- NOT ---o

|

Q --- NOT ---o--- AND ---o

| |

R --- NOT ---o--- |

|

R --- NOT ---o--- AND ---o

| |

S --- NOT ---o--- |

|

Q -----------o--- |

|

o-------------------- OR --- F

**(b) F(A, B, C, D) = Π(0, 1, 2, 3, 5, 7, 9, 13)**

**Use Karnaugh's map to reduce the given function F using the POS form. Draw a logic gate diagram for the reduced POS form. You may use gate with more than two inputs. Assume that the variables and their complements are available as inputs.**

**Karnaugh Map (0s at the minterms listed):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CD\AB** | **00** | **01** | **11** | **10** |
| 00 | 0 | 0 | 1 | 0 |
| 01 | 0 | 1 | 1 | 1 |
| 11 | 1 | 1 | 0 | 1 |
| 10 | 1 | 1 | 1 | 1 |

Export to Sheets

**Grouping the 0s for POS:**

* **Quad:** Top row (CD=00) and second row (CD=01), first column (AB=00) => A+B
* **Quad:** Top row (CD=00) and second row (CD=01), second column (AB=01) => A+B
* **Pair:** Second row (CD=01), last two columns => A+D
* **Pair:** Third row (CD=11), first two columns => C+D

**Reduced POS Expression:** F=(A+B)(A+B)(A+D)(C+D)

**Logic Gate Diagram:**

A ---o--- OR ---o

/ |

B ---o----------|

A ---o--- OR ---o

/ |

B --- NOT ---o--|

A --- NOT ---o--- OR ---o

/ |

D --- NOT ---o----------|

C --- NOT ---o--- OR ---o

/ |

D -----------o----------|

o--- AND --- F

/

o

/

o

**66. From the Logic Circuit diagram given below, name the parts (1), (2), (3) and finally derive the Boolean expression and simplify it:**

**(Image of the logic circuit is needed to answer this question. Please provide the image or describe the logic gates and their connections labeled (1), (2), and (3).)**

Assuming a hypothetical circuit for explanation:

Let's say:

* **(1)** is an AND gate with inputs A and B.
* **(2)** is an OR gate with inputs from the output of (1) and C.
* **(3)** is a NOT gate with input from the output of (2).
* The final output is F(A, B, C).

Then:

* Output of (1): A⋅B
* Output of (2): (A⋅B)+C
* Output of (3): (A⋅B)+C​

Boolean Expression: F(A,B,C)=AB+C​

Simplification: Using De Morgan's Law, F(A,B,C)=AB⋅C=(A+B)⋅C=AC+BC

**67. (a) State a difference between multiplexers and decoders. Also state a use of each.**

(Difference answered in Question 56a)

**Use of Multiplexer:** Data selection in communication systems. **Use of Decoder:** Address decoding in memory units.

**(b) Write the SOP expression, truth table and the logic circuit of the full adder.**

(Truth table and SOP expressions answered in Question 51a. Logic circuit diagram answered in Question 51c.)

**68. (a) Given the Boolean function F(A, B, C, D) = Σ(0, 2, 3, 4, 6, 7, 9, 13). Use Karnaugh's map to reduce the function F using the SOP form. Draw a logic gate diagram for the reduced SOP form. You may use gates with more than two inputs. Assume that the variables and their complements are available as inputs.**

**Karnaugh Map:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CD\AB** | **00** | **01** | **11** | **10** |
| 00 | 1 (0) | 0 | 1 (3) | 1 (2) |
| 01 | 1 (4) | 0 | 1 (7) | 1 (6) |
| 11 | 0 | 1 (9) | 0 | 1 (13) |
| 10 | 0 | 0 | 0 | 0 |

Export to Sheets

**Grouping the 1s:**

* **Quad:** First column (AB=00): ABD
* **Quad:** Second row (CD=01): D
* **Pair:** Top right (AB=10, CD=00 and AB=10, CD=01): ABC
* **Pair:** Bottom right (AB=10, CD=01 and AB=10, CD=11): ABD
* **Pair:** Middle right (AB=11, CD=01 and AB=11, CD=00 - wrap around): BCD

**Reduced SOP Expression:** F=ABD+D+ABC+ABD+BCD F=D(AB+1+BC)+ABC+ABD F=D+ABC+ABD F=D+AB(C+D)

**Logic Gate Diagram:**

D --- NOT ---o

|

A -----------o

|

B --- NOT ---o--- AND ---o

| |

C --- NOT ---o--- |

|

D -----------o--- |

o-------------------- OR --- F

**(b) Given X(A, B, C, D) = Π(2, 3, 4, 5, 12, 14). Use Karnaugh's map to reduce this function X using the given POS form. Draw a logic gate diagram for the reduced POS form. You may use gates with more than two inputs. Assume that the variables and their complements are available as inputs.**

**Karnaugh Map (0s at the minterms listed):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CD\AB** | **00** | **01** | **11** | **10** |
| 00 | 1 | 1 | 1 (12) | 1 |
| 01 | 0 (4) | 0 (5) | 1 | 1 |
| 11 | 1 | 1 | 0 (14) | 1 |
| 10 | 1 | 0 (2) | 0 (3) | 1 |

Export to Sheets

**Grouping the 0s:**

* **Quad:** Second column (AB=01): A+B
* **Quad:** Third row (CD=11): C+D
* **Pair:** Top right (AB=10, CD=00 and AB=10, CD=01): A+B+C

**Reduced POS Expression:** X=(A+B)(C+D)(A+B+C)

**Logic Gate Diagram:**

A ---o--- OR ---o

/ |

B --- NOT ---o--|

C --- NOT ---o--- OR ---o

/ |

D --- NOT ---o----------|

A --- NOT ---o--- OR ---o

/ |

B --- NOT ---o |

/ |

C --- NOT ---o----------|

o--- AND --- X

/

o

/

o

**69. A combinational circuit with 3 inputs A, B, C detects an error during transmission of code and gives the output D as 1 if any two of the inputs are low (0).**

**(i) Write the truth table with inputs A, B, C and Output D.**

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **C** | **D** |
| 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 |

Export to Sheets

**(ii) Write the SOP expression for D (A, B, C). Implement D using NAND gates only.**

**SOP Expression:** D=ABC+ABC+ABC+ABC

**Implementation using NAND gates only:**

1. **Implement NOT:** A NAND gate with both inputs tied together acts as a NOT gate.
2. **Implement AND:** An AND gate can be implemented using two NAND gates (NOT after NAND).
3. **Implement OR:** An OR gate can be implemented using three NAND gates (NOT before NAND).

**Circuit Diagram using NAND gates:**

A ---o---¬

| |

| NAND ---o---¬

| | | |

B ---o---¬ | NAND --- AB

| |

C ---o---¬ o---¬

| |

| NAND --- ABC

A ---o---¬

| |

| NAND ---o---¬

| | | |

B ---o---¬ | NAND --- AB

| |

C -----------o---¬

| |

| NAND --- AB~C

A ---o---¬

| |

| NAND ---o---¬

| | | |

B -----------o---¬ | NAND --- A~B

| |

C ---o---¬ o---¬

| |

| NAND --- A~BC

A -----------o---¬

| |

| NAND --- ~A

B ---o---¬ o---¬

| | | |

| NAND --- AB ---o---¬

| | | | | |

C ---o---¬ o---¬ | NAND --- ~ABC

| | | |

| NAND --- ~AB~C

(Outputs of the four AND implementations) ---o---¬

| |

| NAND --- D

| |

(Outputs of the four AND implementations) ---o---¬

**(iii) Write the POS expression for D (A, B, C). Implement D using NOR gates only.**

**POS Expression:** D=(A+B+C)(A+B+C)(A+B+C)(A+B+C)

**Implementation using NOR gates only:**

1. **Implement NOT:** A NOR gate with both inputs tied together acts as a NOT gate.
2. **Implement OR:** A NOR gate acts as an OR gate if the inputs are complemented.
3. **Implement AND:** An AND gate can be implemented using three NOR gates (NOT before NOR, then NOT after).

**Circuit Diagram using NOR gates:**

A -----------o

|

B ---o---¬ |

| | |

| NOR ---o

| | |

C ---o---¬ |

| | |

| NOR ---o---¬

| |

| NOR --- Term 1

A ---o---¬

| |

| NOR ---o

| |

B -----------o

| |

| NOR ---o---¬

| |

| NOR --- Term 2

C ---o---¬

| |

| NOR ---o

A ---o---¬

| |

| NOR ---o

| |

B ---o---¬

| |

| NOR ---o---¬

| |

| NOR --- Term 3

C -----------o

A ---o---¬

| |

| NOR ---o

| |

B ---o---¬

| |

| NOR ---o---¬

| |

| NOR --- Term 4

C ---o---¬

| |

| NOR ---o

(Outputs of the four OR implementations) ---o---¬

| |

| NOR --- D

| |

(Outputs of the four OR implementations) ---o---¬

**70. (a) Given the Boolean function F(A, B, C, D) = Σ(1, 6, 7, 8, 9, 10, 14, 15). Use Karnaugh's map to reduce the function F using the SOP form. Draw a logic gate diagram for the reduced SOP form. You may use gates with more than two inputs. Assume that the variables and their complements are available as inputs.**

**Karnaugh Map:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CD\AB** | **00** | **01** | **11** | **10** |
| 00 | 0 | 1 (1) | 1 (3) | 0 |
| 01 | 0 | 1 (5) | 1 (7) | 1 (6) |
| 11 | 1 (8) | 1 (9) | 1 (15) | 1 (14) |
| 10 | 0 | 0 | 1 (11) | 1 (10) |

Export to Sheets

**Grouping the 1s:**

* **Quad:** Bottom row (CD=11): AB
* **Quad:** Right column (AB=10): AB
* **Pair:** Top left (AB=00, CD=01 and AB=01, CD=01): ABCD
* **Pair:** Middle left (AB=01, CD=01 and AB=01, CD=11): ABD

**Reduced SOP Expression:** F=AB+AB+ABCD+ABD F=A(B+B)+ABD(C+1) F=A+ABD=A+BD

**Logic Gate Diagram:**

A -----------o

|

B --- AND ---o--- OR --- F

/

D ---o

**(b) Given X(A, B, C, D) = Π(0, 2, 3, 4, 5, 11, 12, 13). Use Karnaugh's map to reduce this function X using the given POS form. Draw a logic gate diagram for the reduced POS form. You may use gates with more than two inputs. Assume that the variables and their complements are available as inputs.**

**Karnaugh Map (0s at the minterms listed):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CD\AB** | **00** | **01** | **11** | **10** |
| 00 | 0 (0) | 1 | 0 (3) | 0 (2) |
| 01 | 0 (4) | 0 (5) | 1 | 1 |
| 11 | 1 | 1 | 0 (13) | 0 (12) |
| 10 | 1 | 1 (9) | 0 (11) | 1 |

Export to Sheets

**Grouping the 0s:**

* **Quad:** First row (CD=00) excluding AB=01: C+D+A
* **Quad:** Second column (AB=01) excluding CD=10: A+B+C
* **Pair:** Bottom row (CD=10), first two columns: C+D+B

**Reduced POS Expression:** X=(A+C+D)(A+B+C)(B+C+D)

**Logic Gate Diagram:**

A ---o--- OR ---o

/ |

C ---o----------|

/ |

D ---o----------|

A --- NOT ---o--- OR ---o

/ |

B --- NOT ---o----------|

/ |

C -----------o----------|

B --- NOT ---o--- OR ---o

/ |

C -----------o----------|

/ |

D --- NOT ---o----------|

o--- AND --- X

/

o

/

o

**71. (a) Given the Boolean function F(A, B, C, D) = Σ(0, 1, 4, 5, 7, 11, 13, 15). Use Karnaugh's map to reduce the function F using the SOP form. Draw a logic gate diagram for the reduced SOP form. You may use gates with more than two inputs. Assume that the variables and their complements are available as inputs.**

**Karnaugh Map:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CD\AB** | **00** | **01** | **11** | **10** |
| 00 | 1 (0) | 1 (1) | 0 | 0 |
| 01 | 1 (4) | 1 (5) | 1 (7) | 0 |
| 11 | 0 | 1 (13) | 1 (15) | 1 (11) |
| 10 | 0 | 0 | 0 | 0 |

Export to Sheets

**Grouping the 1s:**

* **Quad:** First column (AB=00) and second column (AB=01), first two rows (CD=00, 01): A
* **Pair:** Second column (AB=01), second and third rows (CD=01, 11): ABD
* **Pair:** Third column (AB=11), second and third rows (CD=01, 11): BCD
* **Pair:** Fourth column (AB=10) has no 1s.
* **Pair:** Third column (AB=11), third row (CD=11) and second column (AB=01), third row (CD=11): BD

Let's regroup for better simplification:

* **Quad:** First column (AB=00) and second column (AB=01), first row (CD=00): AC
* **Quad:** First column (AB=00) and second column (AB=01), second row (CD=01): AD
* **Pair:** Third column (AB=11), second and third rows (CD=01, 11): BC
* **Pair:** Fourth column (AB=10) has no 1s.
* **Single 1s:** The remaining 1s at (11, 11) which is ABC and (11, 01) which is BCD.

Let's try another grouping:

* **Quad:** First column (AB=00) and second column (AB=01), first row (CD=00): AC
* **Quad:** First column (AB=00) and second column (AB=01), second row (CD=01): AD
* **Pair:** Second column (AB=01), third row (CD=11) and third column (AB=11), third row (CD=11): BD
* **Single 1:** Third column (AB=11), second row (CD=01): BCD

**Reduced SOP Expression:** F=AC+AD+BD+BCD

**Logic Gate Diagram:**

A --- NOT ---o--- AND ---o

| |

C --- NOT ---o--- |

A --- NOT ---o--- AND ---o

| |

D -----------o--- |

B -----------o--- AND ---o

| |

D -----------o--- |

B -----------o--- AND ---o

| |

C -----------o--- |

| |

D --- NOT ---o--- |

o-------------------- OR --- F

/

o

/

o

/

o

**(b) Now given X(A, B, C, D) = Π(0, 3, 4, 7, 9, 10, 11, 12). Use Karnaugh's map to reduce this function X using the given POS form. Draw a logic gate diagram for the reduced POS form. You may use gates with more than two inputs. Assume that the variables and their complements are available as inputs.**

**Karnaugh Map (0s at the minterms listed):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CD\AB** | **00** | **01** | **11** | **10** |
| 00 | 0 (0) | 1 | 1 | 0 (12) |
| 01 | 0 (4) | 1 | 1 | 1 |
| 11 | 1 | 0 (3) | 0 (11) | 0 (10) |
| 10 | 1 | 0 (7) | 1 | 0 (9) |

Export to Sheets

**Grouping the 0s:**

* **Quad:** First row (CD=00), first and last columns: C+D+B
* **Quad:** Third row (CD=11), last two columns: A+B+C
* **Pair:** First column (AB=00), first and second rows: A+B+D
* **Pair:** Second column (AB=01), third and fourth rows: A+B+C

**Reduced POS Expression:** X=(B+C+D)(A+B+C)(A+B+D)(A+B+C)

**Logic Gate Diagram:**

B ---o--- OR ---o

/ |

C ---o----------|

/ |

D ---o----------|

A --- NOT ---o--- OR ---o

/ |

B --- NOT ---o----------|

/ |

C --- NOT ---o----------|

A ---o--- OR ---o

/ |

B ---o----------|

/ |

D --- NOT ---o--|

A --- NOT ---o--- OR ---o

/ |

B -----------o----------|

/ |

C -----------o----------|

o--- AND --- X

/

o

/

o

**72. (a) Draw the logic circuit diagram for an octal to binary encoder and explain its working when a particular digit is pressed.**

**Logic Circuit Diagram for an Octal to Binary Encoder:**

An octal to binary encoder has 8 input lines (D0-D7) and 3 output lines (A, B, C, where A is MSB).

D0 ---

D1 ---o-------o--- C

D2 ---o---o---|

D3 ---o---|---|

D4 ---o---|---|--- B

D5 ---o---|---o---

D6 ---o---|---|---|--- A

D7 ---o-------o---o---

**Boolean Expressions for Outputs:**

* A=D4+D5+D6+D7
* B=D2+D3+D6+D7
* C=D1+D3+D5+D7

**Working Principle:**

When a particular octal digit input line is activated (goes HIGH), the corresponding combination of OR gates will produce the binary equivalent of that digit on the output lines (A, B, C).

**Example: When digit '5' is pressed (D5 is HIGH):**

* Input: D5 = 1, all other D inputs = 0.
* Output A: A=0+0+0+1=1
* Output B: B=0+0+0+1=1
* Output C: C=0+1+0+1=1

The output ABC will be 101, which is the binary representation of the decimal number 5 (octal 5 is also decimal 5).

**Also, state the difference between encoders and decoders.**

(Answered in Question 45)

**(b) Draw the circuit of a two input XOR gate with the help of NOR gates.**

**Circuit Diagram of a 2-input XOR gate using NOR gates:**

A ---o---¬

| |

| NOR --- W

| |

B ---o---¬

A -----------o---¬

| |

| NOR --- Y (A XOR B)

| |

B -----------o---¬

W ---o---¬

| |

| NOR --- Z

| |

Z ---o---¬

**Explanation:**

1. The first NOR gate (inputs A and B) produces output W = A+B​.
2. We feed A and B individually into NOT gates made from NOR gates. Output of NOT(A) is connected to one input of the final NOR gate, and output of NOT(B) is connected to the other input.
3. The final NOR gate has inputs connected as follows:
   * Input 1: A (obtained from A through a NOR-based NOT gate)
   * Input 2: B (obtained from B through a NOR-based NOT gate)

Let's redraw for clarity using the standard construction:

A ---o---¬

| |

| NOR ---o

| | \

B ---o---¬ >--- NOR --- XOR Output (A ⊕ B)

/

A ---o---¬

| |

| NOR ---o

| |

B ---o---¬

**Step-by-step Boolean derivation:**

1. First NOR gate: P=A+B​=AB
2. Second NOR gate (using NOR as NOT): Q=A
3. Third NOR gate (using NOR as NOT): R=B
4. Final NOR gate: XOR=Q+R​=A+B​=AB=AB **(This is wrong, let's correct the connections)**

**Corrected Circuit Diagram of a 2-input XOR gate using NOR gates:**

A ---o---¬

| |

| NOR --- P

| |

B ---o---¬

A -----------o---¬

| |

| NOR --- Y

| |

P -----------o---¬

B -----------o---¬

| |

| NOR --- Z

| |

P -----------o---¬

Y ---o---¬

| |

| NOR --- XOR Output (A ⊕ B)

| |

Z ---o---¬

**Step-by-step Boolean derivation (Corrected):**

1. First NOR gate: P=A+B​
2. Second NOR gate: Y=A+P​=A+A+B​​=A+(AB)​=(A+A)(A+B)​=1⋅(A+B)​=A+B​=AB
3. Third NOR gate: Z=B+P​=B+A+B​​=B+(AB)​=(B+A)(B+B)​=(B+A)⋅1​=B+A​=AB
4. Final NOR gate: XOR=Y+Z​=AB+AB​

There's still an issue. Let's use the standard 4-NOR gate construction:

A ---o---¬

| |

| NOR --- P

| |

B ---o---¬

A -----------o---¬

| |

| NOR --- Q

| |

P -----------o---¬

B -----------o---¬

| |

| NOR --- R

| |

P -----------o---¬

Q ---o---¬

| |

| NOR --- XOR Output (A ⊕ B)

| |

R ---o---¬

**Step-by-step Boolean derivation (Corrected Again):**

1. First NOR gate: P=A+B​
2. Second NOR gate: Q=A+P​=A+A+B​​
3. Third NOR gate: R=B+P​=B+A+B​​
4. Fourth NOR gate: XOR=Q+R​=A+A+B​​+B+A+B​​​

This derivation is getting complex. Let's use the fact that A⊕B=(A+B)(AB)=(A+B)(A+B).

1. NOR(A, B) = A+B​
2. NOR(A, B) = A+B​=AB (We need A and B which can be made from NOR)

**Final Corrected Circuit Diagram of a 2-input XOR gate using NOR gates:**

A ---o---¬

| |

| NOR --- P

| |

B ---o---¬

A ---o---¬

| |

| NOR --- A'

B ---o---¬

| |

| NOR --- B'

A' ---o---¬

| |

| NOR --- Q (A' + B')

| |

B' ---o---¬

P ---o---¬

| |

| NOR --- R

| |

Q ---o---¬

**Boolean Derivation:**

1. P=A+B​
2. A′=A
3. B′=B
4. Q=A′+B′​=A+B​=AB
5. R=P+Q​=A+B​+AB​=(AB)+AB​ Using De Morgan's: R=(A+B)(A+B)​=AA+AB+BA+BB​=0+AB+AB+0​=AB+AB​=AB+AB=A⊕B

This final circuit and derivation are correct.