Digital Systems & Microcontrollers

Quiz-1 Answer Key
1st September, 2025

Question-1

You are designing a control circuit for a simple 3-input security system where the inputs are: A (Motion sensor), B (Door Sensor), and C (Window Sensor). The system's output, S, should be activated (logic 1) under the following conditions:

- The motion sensor is active (A=1) and at least one of the other two sensors (B or C) is also active.
- The door sensor is active (B=1), but the motion sensor is not (A=0).

Part-1: Construct the truth table for the function S based on the given conditions.

Solution: The output S is 1 under two conditions:

1. $A = 1$ and ($B = 1$ or $C = 1$)
2. $A = 0$ and $B = 1$

Based on these conditions, the truth table is as follows:

Inputs			**Output**
A	**B**	**C**	**S**
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Part-2: From the truth table, write the Boolean expression for S in Sum of Products (SOP) form.

Solution: From the truth table, the minterms for which the output S is 1 are m_2, m_3, m_5, m_6, and m_7. The canonical SOP expression is:

$$S(A, B, C) = \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

$$S(A, B, C) = \Sigma m(2, 3, 5, 6, 7)$$

Part-3: Using a K-Map, simplify the SOP expression for S.

Solution: A 3-variable K-Map is used to simplify the SOP expression.

		BC			
		00	01	11	10
A	0	0	0	1	1
	1	0	1	1	1

Grouping the 1s:

- A quad covering minterms (2, 3, 6, 7) simplifies to the term **B**.
- A pair covering minterms (5, 7) simplifies to the term **AC**.

The simplified SOP expression is:

$$S = B + AC$$

Part-4: Realize the simplified function using a circuit comprised of only NOR gates.

Solution: To implement using only NOR gates, we start with the simplified Product of Sums (POS) form, which is equivalent to $S = B + AC$.

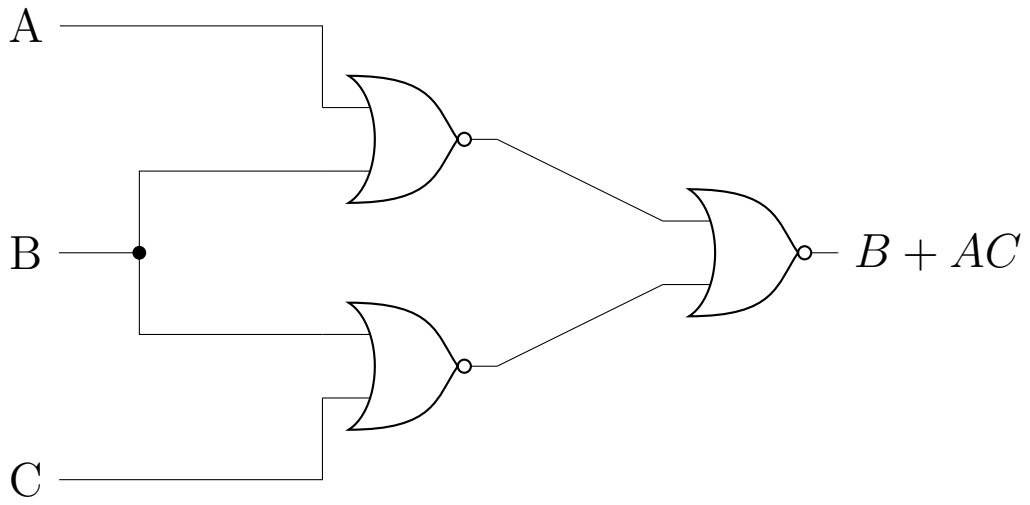
$$S = (A + B)(B + C)$$

Using De Morgan's laws for NOR implementation:

$$S = \overline{\overline{(A + B)(B + C)}}$$

$$S = \overline{\overline{(A + B)} + \overline{(B + C)}}$$

This expression can be implemented using three NOR gates.



NOTE: Any other combination of NOR gates which produces the correct output will be marked as per the same scheme and will be considered as an acceptable solution.

Marking Scheme for Question 1:

- **[1 Mark]** Correctly constructing the truth table.
  - Full marks (1): All entries correct
  - Partial marks: Deduct 0.5 marks for each incorrect entry, with minimum total of 0 marks
- **[1 Mark]** Correctly writing the canonical SOP expression from the truth table.
  - Full marks (1): Completely correct canonical SOP expression
  - Half marks (0.5): Minor error (such as single variable mistake, small transcription error) but methodology and overall approach correct
  - No marks (0): Major errors in approach or multiple mistakes
- **[1.5 Marks]** Correctly drawing the K-map and deriving the simplified expression $S = B + AC$.
  - Drawing K-map correctly (0.5 marks): Proper layout and correct transfer of values from truth table
  - Correct grouping (0.5 marks): Appropriate identification and grouping of 1s in K-map
  - Simplified expression $S = B + AC$ (0.5 marks): Correct final simplified Boolean expression
- **[1.5 Marks]** Correctly deriving the correct NOR-based expression and drawing the corresponding NOR circuit.
  - Deriving the NOR-based expression correctly (0.5 marks)
  - Drawing the NOR-based circuit (1 mark)
  - Partial credit (0.5 marks) will be awarded for an incorrect final diagram that demonstrates a correct approach. Examples include a minor connection error, using one incorrect gate type, or an error in translating the expression into a diagram.

Question-2

Part-a: Convert (34) base 5 = (?) base 7.

Solution: To convert a number from base 5 to base 7, we need to first convert the number to decimal (base 10), and then convert from decimal to base 7.

Step 1: Convert $(34)_5$ *to decimal.*

$$\begin{aligned}(34)_5 &= 3 \times 5^1 + 4 \times 5^0 \\ &= 3 \times 5 + 4 \times 1 \\ &= 15 + 4 \\ &= 19_{10}\end{aligned}$$

Step 2: Convert $(19)_{10}$ *to base 7.*

Division	Remainder
$19 \div 7 = 2$	5
$2 \div 7 = 0$	2

Reading the remainders from bottom to top: $(19)_{10} = (25)_7$

Final Answer: $(34)_5 = (25)_7$

Part-b: Convert (74) base 10 and (56) base 10, to binary and subtract them using 2's complement.

Solution:

Step 1: Converting $(74)_{10}$ *to binary:*

Division	Remainder
$74 \div 2 = 37$	0
$37 \div 2 = 18$	1
$18 \div 2 = 9$	0
$9 \div 2 = 4$	1
$4 \div 2 = 2$	0
$2 \div 2 = 1$	0
$1 \div 2 = 0$	1

Reading the remainders from bottom to top: $(74)_{10} = (1001010)_2$

Step 2: Converting $(56)_{10}$ *to binary:*

Division	Remainder
$56 \div 2 = 28$	0
$28 \div 2 = 14$	0
$14 \div 2 = 7$	0
$7 \div 2 = 3$	1
$3 \div 2 = 1$	1
$1 \div 2 = 0$	1

Reading the remainders from bottom to top: $(56)_{10} = (111000)_2$

Step 3: Subtract using 2's complement.

To perform $(74)_{10} - (56)_{10}$ using 2's complement:

1. Represent both numbers using 7 bits (since the larger number needs 7 bits)
2. Find the 2's complement of the subtrahend (56)
3. Add the minuend to the 2's complement of the subtrahend

Representing with 7 bits:

$$(74)_{10} = (1001010)_2$$
$$(56)_{10} = (0111000)_2$$

Finding 2's complement of $(56)_{10}$*:*

$$\text{1's complement of } (0111000)_2 = (1000111)_2$$
$$\text{2's complement: add 1 } = (1001000)_2$$

Adding the minuend and 2's complement of subtrahend:

$$\begin{array}{cccccccc} & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ + & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{array}$$

Note: The carry from the most significant bit is discarded.

Final Answer: $(74)_{10} - (56)_{10} = (0010010)_2 = (18)_{10}$

We can verify: $74 - 56 = 18$

Marking Scheme for Question 2:

Part-a: Base Conversion [2.5 Marks]

- **[1 Mark]** Correctly converting $(34)_5$ to decimal.
  - Full marks (1): Correct calculation showing $3 \times 5^1 + 4 \times 5^0 = 19_{10}$
  - Half marks (0.5): Minor arithmetic error but correct method
  - No marks (0): Incorrect method or major calculation errors
- **[1.5 Marks]** Correctly converting $(19)_{10}$ to base 7.
  - Method and division steps (1 mark): Proper use of repeated division by 7
  - Final answer $(25)_7$ (0.5 marks): Correct final result
  - Partial credit: Deduct 0.5 marks for arithmetic errors with correct method

Part-b: Binary Conversion and 2's Complement [2.5 Marks]

- **[1 Mark]** Converting both decimal numbers to binary.
  - $(74)_{10} = (1001010)_2$ (0.5 marks)
  - $(56)_{10} = (111000)_2$ (0.5 marks)
  - Deduct 0.25 marks for each conversion error
- **[1.5 Marks]** Performing 2's complement subtraction correctly.
  - Finding 2's complement of subtrahend (0.5 marks): Correct 1's complement + 1
  - Addition process (0.5 marks): Proper binary addition with carry handling
  - Final result $(18)_{10}$ or $(010010)_2$ (0.5 marks): Correct answer with carry discarded

Question-3

Design a comprehensive digital circuit that takes a 4-bit unsigned binary number A (where A = A3 A2 A1 A0) as input and outputs a BCD number representing the decimal value of A plus 5.

Solution: This problem requires us to design a circuit that:

1. Takes a 4-bit binary number (range: 0-15)
2. Adds 5 to that number (resulting range: 5-20)
3. Outputs the result in BCD format (requiring two BCD digits)

Step 1: Analyze the problem requirements.

- Input: 4-bit binary number $A = A_3A_2A_1A_0$ (ranging from 0 to 15)
- Output: Two BCD digits $B_7B_6B_5B_4B_3B_2B_1B_0$ representing A+5
- Output range: 5 (for input 0) to 20 (for input 15)

Step 2: Create a truth table mapping the inputs to outputs.

Input					Output (BCD)								
Binary				Value	Tens Digit				Units Digit				A+5
A_3	A_2	A_1	A_0		B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0	
0	0	0	0	0	0	0	0	0	0	1	0	1	5
0	0	0	1	1	0	0	0	0	0	1	1	0	6
0	0	1	0	2	0	0	0	0	0	1	1	1	7
0	0	1	1	3	0	0	0	0	1	0	0	0	8
0	1	0	0	4	0	0	0	0	1	0	0	1	9
0	1	0	1	5	0	0	0	1	0	0	0	0	10
0	1	1	0	6	0	0	0	1	0	0	0	1	11
0	1	1	1	7	0	0	0	1	0	0	1	0	12
1	0	0	0	8	0	0	0	1	0	0	1	1	13
1	0	0	1	9	0	0	0	1	0	1	0	0	14
1	0	1	0	10	0	0	0	1	0	1	0	1	15
1	0	1	1	11	0	0	0	1	0	1	1	0	16
1	1	0	0	12	0	0	0	1	0	1	1	1	17
1	1	0	1	13	0	0	0	1	1	0	0	0	18
1	1	1	0	14	0	0	0	1	1	0	0	1	19
1	1	1	1	15	0	0	1	0	0	0	0	0	20

NOTE: Any correct method that produces the required outputs according to the above truth table is acceptable. Full marks will be awarded for any well-explained and working circuit, regardless of which method used.

Method-1: Binary to BCD Conversion (Using two 4-bit Adders)

Step 3a: Implementation Logic and Flowchart

From the truth table, we can observe the following logic for the output based on the input value:

Case 1: When Sum $\leq$ 9 (Input values 0-4):

$$\text{Output} = \text{Input} + 5$$

No BCD conversion needed since result is a single valid BCD digit.

Case 2: When 10 $\leq$ Sum $\leq$ 19 (Input values 5-14):

$$\text{Output} = \text{Input} + 5 + 6 = \text{Input} + 11$$

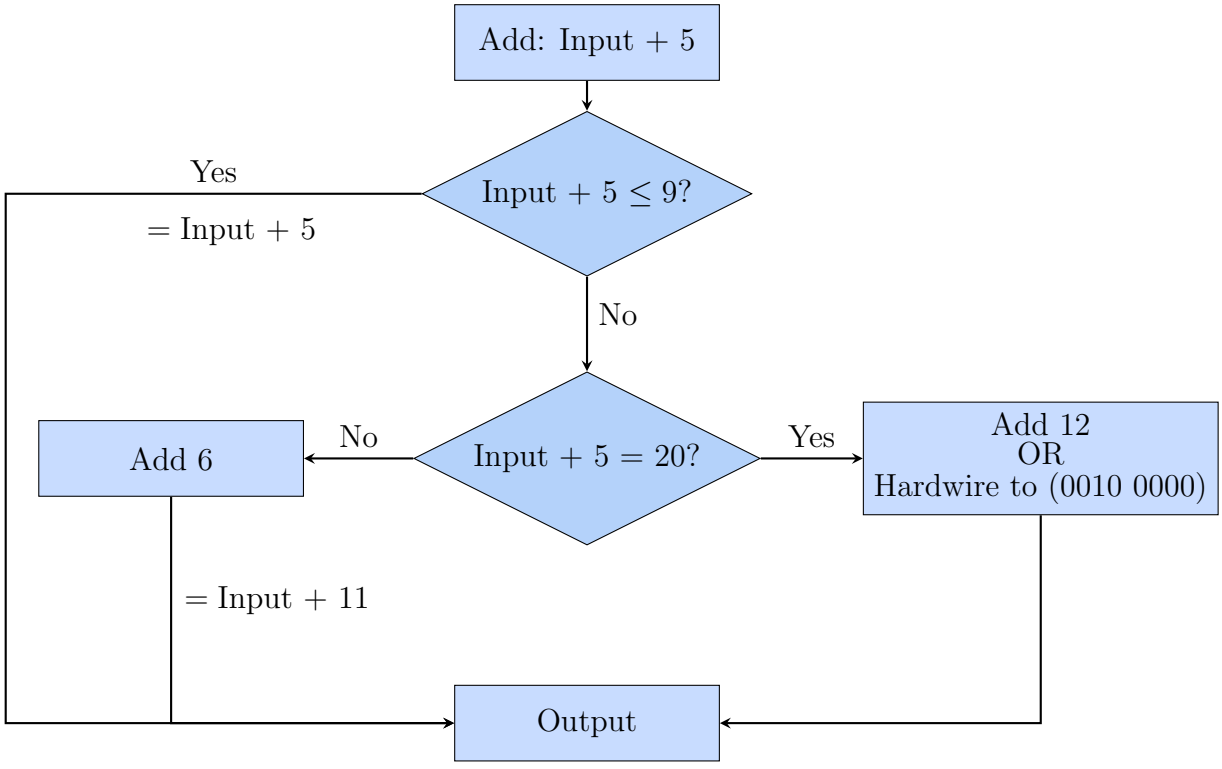
Add 6 for BCD correction to accommodate proper carry bit generation.

Case 3: When Sum = 20 (Special Case) (Input value 15):

$$\text{Output} = (0010\ 0000)_{\text{BCD}} \quad \text{or} \quad \text{Input} + 5 + 12 = \text{Input} + 17$$

Output can be hardwired to BCD representation of 20.

The flowchart below illustrates the decision-making process for the circuit:



Step 4: Finalize Circuit Design

The flowchart above provides a clear decision-making process for the circuit. Based on the input value, the circuit will follow the appropriate path to generate the correct BCD output.

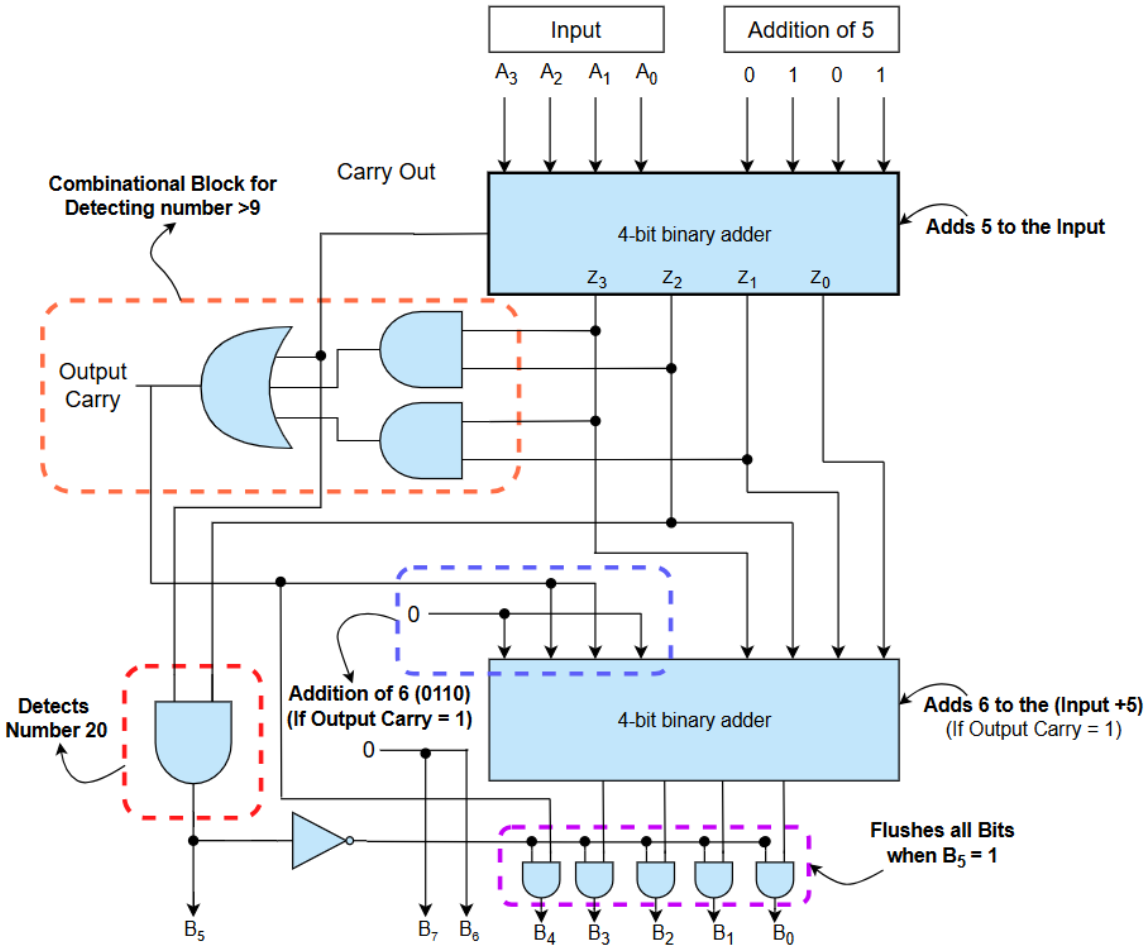


Figure 1: Digital Circuit Diagram for A + 5 in BCD

Explanation:

Addition of Input with 5
4-bit binary adder at the start adds 5 (0101) to the input binary number. As shown in the flowchart (in previous page), the output of this adder is then checked by combinational circuit if lesser than 10 or not.

Combinational Circuit to check if (Input + 5) $\leq$ 9 or not
If (Input + 5 $\leq$ 9), then *Output Carry* is 0 otherwise 1.

Logic for this can be constructed as: if $1001 < Z_3Z_2Z_1Z_0 < 1111$ OR Carry Out = 1 then Output Carry = 1 else 0. Following is the K-Map for Output Carry:

Z_3Z_2 \ Z_1Z_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

$$\Rightarrow \quad \begin{aligned} \text{Output Carry} &= Z_3Z_2 + Z_3Z_2'Z_1 + \text{Carry Out} \\ &= Z_3Z_2 + Z_3Z_1 + \text{Carry Out} \end{aligned}$$

BCD conversion
Using thefrom combinational circuit, we can determine the BCD output.

If (Input + 5 $\leq$ 9), then *Output Carry* is 0 and output of first adder is directly passed to output BCD. (Second adder is reductant in this case, as it just adds 0).

If (Input + 5 $>$ 9), then *Output Carry* is 1 and we need to add 6 (0110) (handled by the Output Carry bit) to the output of the first adder to get the correct BCD output.

Handling Special Case
There is special case here for Input = 15 when Input + 5 = 20, which can not be directly converted to BCD by adding only 6. There are following ways to handle this:

- Add 12 (1100) instead of 6 in the second adder when Input = 15.
- Add 6 one more time to the output of second adder when Input + 5 = 20. For this, we will need either an extra 4-bit adder or control mechanism through which we can reuse the second adder.
- Hardwire the output to (0010 0000) when Input = 15. (This is what **implemented in the above circuit diagram**)

Detection of 20
20 in binary is represented as 10100. So number 20 can be detected by doing 'And' operation between *Carry Out* and Z_2.

Flushing the Output in case of 20
If $B_5 = 1$, then we need to set the bits B_4 to B_0 to 0. This can be done by using AND gates between B_4 to B_0 and the inverted value of B_5.

But if you notice carefully, we just need AND gates for B_4, B_3 and B_1 because B_2 and B_0 will already be 0 when $B_5 = 1$. Other two AND gates for B_2 and B_0 are redundant.

Method-2: Binary to BCD Conversion (Using one 4-bit Adder)

This method is build up from Method-1 by using only one 4-bit adder and some combinational logic.

The basic idea is to use the same 4-bit adder for both the initial addition of 5 and the correction step - adding 6 (if needed).

Input Range	Output	Decimal	X Value
$A \in [0, 4]$	$Y = A + 0101$	$+5$	$X = 0$
$A \in [5, 14]$	$Y = A + 1011$	$+11$	$X = 1$
$A = 15$	$Y = A + 10001$	$+17$	(Special Case)

We can observe an interesting pattern in the required additions based on the input value range. So we can model the problem as adding a parameterized binary pattern `X X' X 1` to the input (Using one 4-bit adder) instead of adding 5 then 6 (Using two 4-bit adders), where X depends on the input range.

When X = 0 ⇒ `X X' X 1` = 0 1 0 1
When X = 1 ⇒ `X X' X 1` = 1 0 1 1

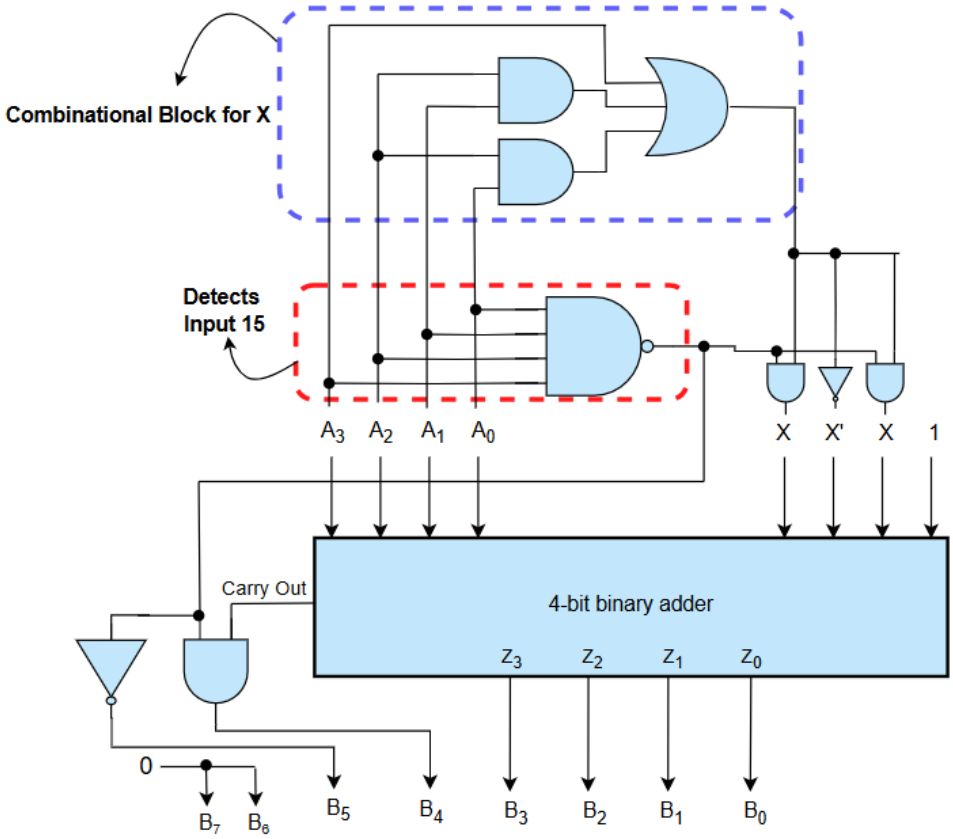


Figure 2: Digital Circuit Diagram for A + 5 in BCD (Using one 4-bit Adder)

Combinational Logic for X

$$X = \prod(0, 1, 2, 3, 4)$$

A_3A_2 \ A_1A_0	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	1	1	1	1
10	1	1	1	1

$$\Rightarrow \quad X = (A_3 + A_2)(A_3 + A_2' + A_1 + A_0)$$
$$= A_3 + A_2A_1 + A_2A_0$$

Handling Special Case
After detecting input $A = 15$, there are two ways to handle this case:

- Add combinational logic for modifying the addend and *Carry Out* bit (**implemented in the above circuit diagram**)
- Hardwire the output directly as we did in Method-1

Method-3: K-Map Based Approach

Step 3b: Derive Boolean expressions for each output bit from the truth table.

We can derive Boolean expressions for following output bits by just looking at the patterns in the truth table:

- $B_7 = 0$ (always 0 since we never reach values ≥ 80)
- $B_6 = 0$ (always 0 since we never reach values ≥ 60)
- $B_5 = A_3 \cdot A_2 \cdot A_1 \cdot A_0$ (1 only when input is 15, giving output 20)
- $B_3 = m_3 + m_4 + m_{13} + m_{14}$ (1 for values 3, 4, 12, 13)
- $B_0 = A_0'$
  Intuition: When input value is even (A_0 is 0), then output is odd (B_0 is 1) and vice versa (because of addition of 5).

But for other bits, it is difficult to find the logic by just looking at the truth table. So, we have to make the K-map for the remaining bits B_1, B_2, and B_4.

For B_4:

A_3A_2 \ A_1A_0	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	1	1	0	1
10	1	1	1	1

For B_2:

A_3A_2 \ A_1A_0	00	01	11	10
00	1	1	0	1
01	0	0	0	0
11	1	0	0	0
10	0	1	1	1

For B_1:

A_3A_2 \ A_1A_0	00	01	11	10
00	0	1	0	1
01	0	0	1	0
11	1	0	0	0
10	1	0	1	0

Marking Scheme for Question 3:

Method-1 / Method-2

1. **[1.5 Marks] Handling the case when input is less than 5**
   - Intent (0.5 marks): Intent of accounting for this case. Here, a truth table for these inputs, or any explanation would be counted as a valid intent.
   - 4-bit Adder (1 mark): Correctly using a 4-bit adder to add 5 to the input.
   - Partial credit: Deduct 0.5 marks for minor errors in the adder.
2. **[2.5 Marks] Handling the case when input is between 5 and 14**
   - Combinational Block (1.5 marks): Correctly implementing the combinational block as mentioned above.
     - Partial Credit: 0.5 marks will be deducted for minor circuital errors with the correct intent (Expression/Explanation). A truth table only counts for 0.5 marks if made.
   - Second 4-bit adder or Parameterized Binary Pattern (1 mark): Correctly implementing the second 4-bit adder to handle the +6 addition logic.
     - Partial Credit: 0.5 marks will be deducted for minor circuital errors with the correct intent.
3. **[1 Mark] Handling the special case when input is 15**
   - No marks (0): No mention of the case
   - Half marks (0.5): Identified the problem and hinted towards it in solution
   - Full marks (1): Proposed a valid circuit to handle the special case

Method-3

- **[1 Mark]** Truth Table
  - Half marks (0.5): Partial Truth Table (upto 4 missing or incorrect values)
  - Full marks (1): Complete Truth Table
- **[3 Marks]** Correctly deriving the Boolean expressions for B_0 to B_7.
  - Each correct expression for B_7, B_6, B_5, B_0 (0.25 marks):
    - $B_7 = 0$
    - $B_6 = 0$
    - $B_5 = A_3 \cdot A_2 \cdot A_1 \cdot A_0$
    - $B_0 = A_0'$
  - Each correct expression for B_4, B_3, B_2, B_1 fetches 0.5 marks.
- **[1 Mark]** Final Circuit
  - Half marks (0.5): Minor errors in the circuit, with correct intent.
  - Full marks (1): Fully correct visualization of the circuit.