Lab 2B: Binary Coded Decimal (BCD) Converter

Objective

The objective of this lab is to design and verify a **Binary to BCD Converter** that converts an **8-bit binary number (0–255)** into a **3-digit BCD (000–255)** representation. This experiment helps understand the **Double-Dabble algorithm**, a widely used method for binary-to-BCD conversion, and its implementation as a purely combinational circuit.

Design Requirements

- **Input**:
    - 8-bit binary number (`in_bits`)
- **Output**:
    - 12-bit BCD output (`bcd_bits`), representing 3 decimal digits (hundreds, tens, ones)
- **Constraints**:
    - Range: 0–255 decimal
    - Purely combinational logic (no clocks or sequential elements)
- **Algorithm**:
    - Use the **Double-Dabble (Shift-and-Add-3)** algorithm

Algorithm Understanding

The **Double-Dabble algorithm** is an efficient way to convert binary numbers into BCD:

1. **Initialization**:
    - Place the binary number in the least significant bits of a shift register.

    - Reserve space for the BCD digits (hundreds, tens, ones).

2. **Shift and Correct**:
    - For each bit of the binary number (8 iterations for 8-bit input):
        - If any BCD digit is **greater than or equal to 5**, add 3 to correct it.
        - Perform a **left shift** across the entire register.
3. **Extraction**:
    - After all shifts, the register contains the valid BCD digits.

Example (binary 37 = 0010 0101):

- After applying the Double-Dabble steps, the output becomes:
    - Hundreds = 0, Tens = 3, Ones = 7 → **BCD = 037**.

Design Methodology

1. **Input Placement**:
    - The 8-bit binary input was loaded into the lower part of a shift register.
2. **Correction Logic**:
    - Before each shift, BCD digits were checked.
    - If ≥ 5, **3 was added** to that digit.
3. **Shifting Operation**:
    - The register was shifted left by one bit each iteration.
4. **Output Extraction**:
    - After 8 iterations, the **12-bit BCD result** was taken from the upper part of the register.

Simulation and Verification

Test Setup

- Multiple binary input values (0, 5, 9, 12, 37, 99, 123, 255) were applied.
- The corresponding BCD outputs were observed and compared against expected decimal values.

Simulation Results

Binary Input	Expected Decimal	BCD Output (Hundreds Tens Ones)
0	000	000
5	005	005
9	009	009
12	012	012
37	037	037
99	099	099
123	123	123
255	255	255

- All test cases matched the expected results.
- The conversion was **fast and purely combinational**.

Results and Discussion

- The **Double-Dabble algorithm** was successfully implemented for binary-to-BCD conversion.
- The circuit correctly handled all binary values in the 0–255 range.
- Since the design is combinational, it provides **instantaneous conversion** without requiring clock cycles.
- This method can be extended to larger binary inputs (e.g., 16-bit or 32-bit) by increasing the number of iterations and BCD digits.

Conclusion

In this lab, an **8-bit Binary to 3-digit BCD Converter** was designed and tested using the Double-Dabble algorithm. The design correctly converted binary values into BCD format for the entire input range (0–255). The experiment reinforced the understanding of **binary-to-decimal conversion, correction logic, and combinational circuit design techniques**.