## Project Overview

### Encryption Algorithm Used

Your project uses a **substitution-based encryption algorithm**. Specifically, it uses a custom substitution table (5x5 matrix) where each letter is mapped to a unique position. The encryption involves combining the positions of plaintext characters with those of a secret key using row-column arithmetic. Special characters are handled separately by modifying their ASCII values based on the key positions.

- **Encryption Process:**
  1. **Substitution Table Initialization**: A 5x5 table is populated with characters.
  2. **Key Setup**: A fixed key `["N", "E", "D", "E", "L", "C", "U"]` is used.
  3. **Character Handling**:
     - For letters, their positions are identified in the table and combined with the key's position values.
     - For special characters, their ASCII values are directly modified by adding key position values.
- **Decryption Process:**
  1. **Key and Table Initialization**: The same key and substitution table are used.
  2. **Position Extraction**: The encrypted values are split back into their original row-column positions by subtracting the key position values.
  3. **Character Restoration**:
     - For letters, positions are mapped back to characters.
     - For special characters, ASCII values are adjusted to restore the original characters.

---

## Test Benches Overview

Your project includes multiple test benches to validate different scenarios in the encryption-decryption pipeline:

1. **`tb_enc_dec_repetitive_letters.sv`**:
   Tests how repetitive letters (like "AABBCC...") are encrypted and decrypted to ensure consistent handling.
2. **`tb_word_enc_dec.sv`**:
   Encrypts and decrypts a word (like "HELLOW"), checking if the row-column mapping is correctly applied.
3. **`special_characters_tb.sv`**:
   Ensures that special characters (`@`, `#`, `$`, etc.) are properly encrypted and restored during decryption.
4. **`tb_enc_dec_name.sv`**:
   Encrypts and decrypts a name ("NEDELCU"), validating if key-based encryption performs correctly on names.
5. **`tb_single_letter.sv`**:
   Tests encryption-decryption for a single letter, ensuring edge cases like minimal input work correctly.

6. **`mixed_case_tb.sv`**:
   Verifies that mixed case inputs ("HeLLoW") are handled correctly, converting to uppercase for encryption and restoring the original case after decryption.
7. **`tb_enc_dec_long_message.sv`**:
   Tests a long message ("HELLOTHISISATESTMESSAGE") to validate the algorithm's scalability and performance for larger texts.

---

## Potential Questions & Answers for Presentation

1. **Q: What kind of encryption are you using?**
   **A:** The project uses a substitution-based encryption algorithm inspired by classical cipher techniques. It combines the positions of characters from a substitution table with key-derived positions to create an encrypted output.
2. **Q: How do you handle special characters?**
   **A:** Special characters are not found in the substitution table. Instead, their ASCII values are modified by adding a numeric representation of the key's position, ensuring they're securely encrypted and can be accurately restored during decryption.
3. **Q: How does the decryption process reverse the encryption?**
   **A:** The decryption subtracts the key's position values from the encrypted text, retrieves the original row-column positions, and maps them back to the corresponding characters from the substitution table.
4. **Q: How do you ensure the encryption is consistent across different text cases?**
   **A:** Mixed case inputs are converted to uppercase during encryption, and the original casing is restored during decryption using a case map.
5. **Q: What happens if the message length exceeds the key length?**
   **A:** The key is reused cyclically using modulo operations (`i % SEC_LEN`), ensuring every character in the message is encrypted with a key character.
6. **Q: Are there any limitations to this encryption method?**
   **A:** While effective for basic text encryption, this method doesn't offer high-level security like modern cryptographic algorithms (AES, RSA). It's primarily educational and demonstrates fundamental encryption techniques.