AES-128 Decryption Module Implementation Report

Ishan Rehal, Tejaswa Singh Mehra

November 10, 2024

1 Introduction

This report documents the design and implementation of an AES-128 decryption module in VHDL, targeting a Basys3 FPGA. The module integrates multiple compute units, organized hierarchically, to perform AES decryption. Key components include a top-level controller (`TopModule`) that orchestrates memory management, decryption, and display operations.

2 Approach and Design Hierarchy

The decryption system operates under a hierarchical approach, where each component fulfills a specialized role in the overall AES decryption process. The main components are:

- **TopModule (Top-level Controller)**: Manages high-level decryption and display phases, including memory and display control.
- **Compute Units**: These child modules perform specific AES transformations or display functions:
    a. `AES_decryptor`: Executes the core AES transformations in a sequence controlled by a finite state machine (FSM).
    b. `Output_RAM`: Stores intermediate results during decryption and supplies data for the display module.
    c. `DigitalDisplay`: Displays decrypted data on a multiplexed seven-segment display, driven by the `Timing_block`.

3 ROM/RAM used

This section details the ROM/RAM used

- **ROM Units**: These child ROM units store the inputs:
    a. `key_ROM (blk_mem_gen_1)`: 8-bit width, 160 depth, stores round keys
    b. `ciphertext_mem (blk_mem_gen_2)`: 8-bit width, 128*n depth, stores ciphertext.

c. `inv_sbox (dist_mem_gen_0)`: 8-bit width, 256 depth, stores inverse sub-bytes.

- **RAM Units**: These child RAM units store the inputs:

  a. `Intermediate_RAM_port (Intermediate_RAM)`: 8-bit width, 16 depth, stores intermediate steps

  b. `ram_inst (Output_RAM)`: 8-bit width, 128*n depth, stores decrypted cipher of 128 bit block sequentially

4 Module Explanation and Hierarchy

4.1 TopModule

The `TopModule` serves as the primary control interface, managing all decryption and display activities. It coordinates data flow between the compute units based on the FSM states and provides timing controls to ensure signal stability. `TopModule` first initiates decryption through `AES_decryptor`, stores results in `Output_RAM`, and transitions to display mode upon completion.

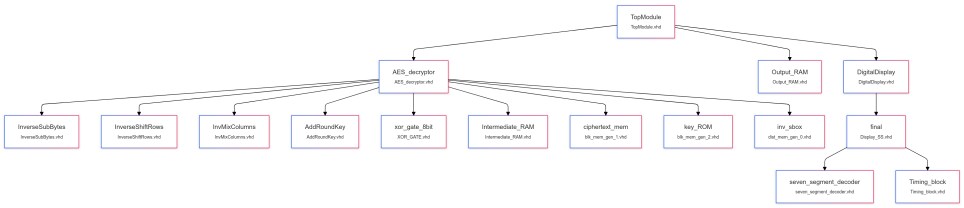

Figure 1: Hierarchical Block Diagram of AES-128 Decryption System

4.2 Compute Units and Child Modules

The compute units are organized into a hierarchy where each module performs specific transformations, memory operations, or display control.

1. **AES_decryptor (Core AES Compute Unit)**

   - **Description**: The `AES_decryptor` handles all decryption transformations following the AES decryption process. This module is controlled by an FSM with states including IDLE, LOAD_CIPHERTEXT, ADD_ROUND_KEY, INVERSE_SUB_BYTES, INVERSE_SHIFT_ROWS, INVERSE_MIX_COLUMNS, and DONE_FULLY.
   - **Submodules**:
     (a) `InverseSubBytes`: Performs inverse S-box substitution for AES decryption.
     (b) `InverseShiftRows`: Shifts rows of the state matrix to reverse the row mixing from encryption.
     (c) `InverseMixColumns`: Applies matrix multiplication in GF(2^8) to reverse column mixing.
     (d) `AddRoundKey`: XORs each byte with the corresponding byte from the round key.

- **Functionality**: Processes each transformation sequentially under FSM control and outputs four 32-bit decrypted blocks.

2. Output_RAM (Memory Management)

- **Description**: This RAM component stores 128-bit decrypted text from the decryption process, byte by byte, and supplies data for the display module.
- **Functionality**: Provides addressable 8-bit storage for decrypted data and intermediates, read by the display module for output.

3. DigitalDisplay (Output Display Unit)

- **Description**: Manages visual display of decrypted data using four seven-segment displays.
- **Submodules**:
  (a) `Timing_block`: Divides clock and controls multiplexing for the seven-segment displays.
  (b) `seven_segment_decoder`: Decodes each 4-bit nibble into seven-segment display patterns.
  (c) `final`: Controls the display anodes, passing decoded values to the segment outputs.
- **Functionality**: Cycles through four nibbles of a 32-bit word for hexadecimal display, synchronized with the clock using `Timing_block`.

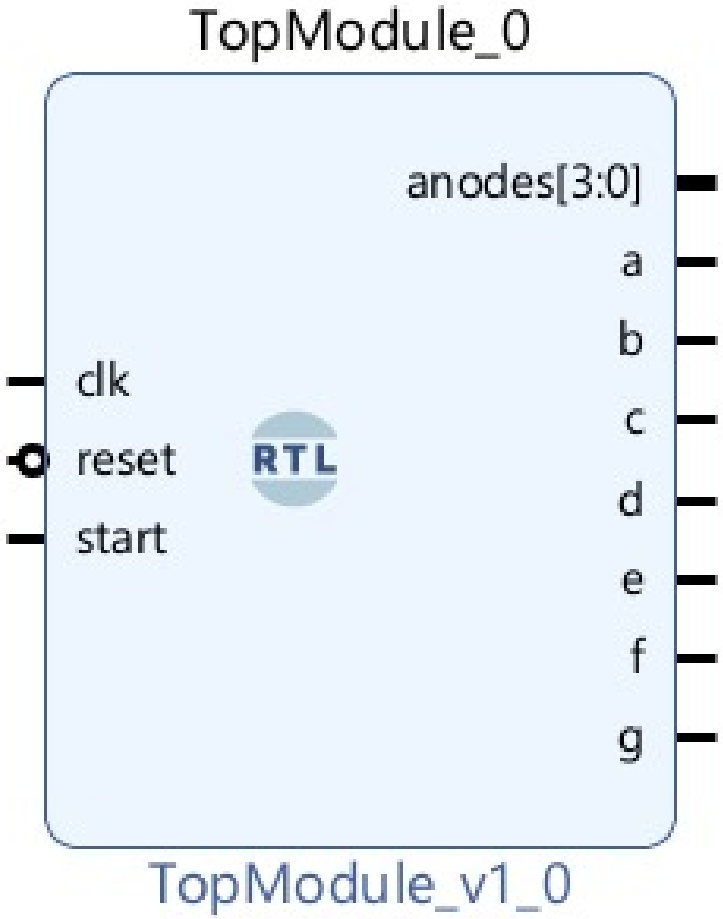

Figure 2: Block Diagram for TopModule

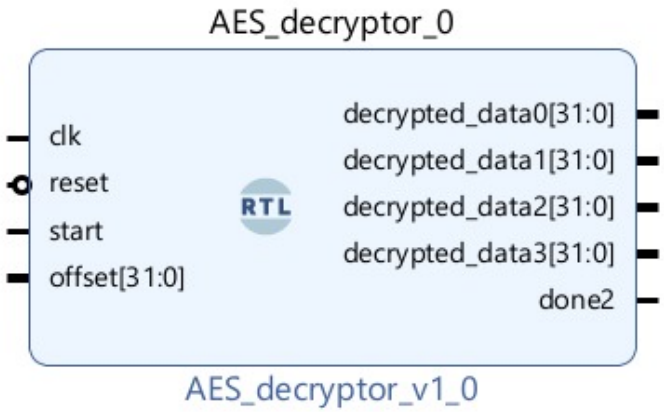

Figure 3: Block Diagram for AES_Decrypter

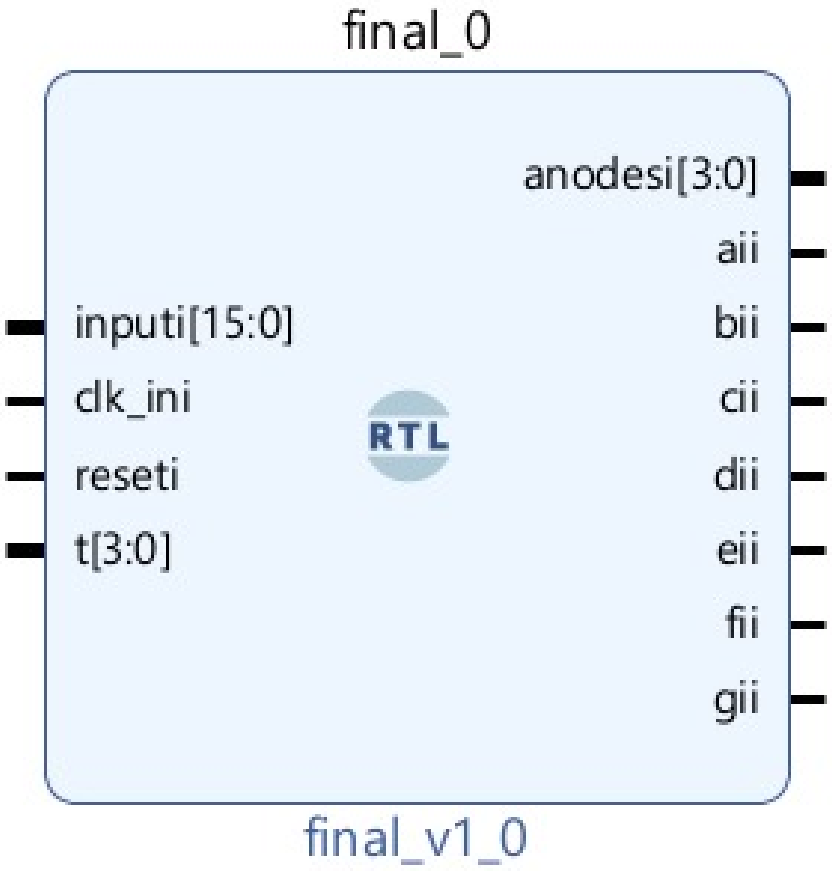

Figure 4: Block Diagram for final

Figure 5: Block Diagram for TimingBlock

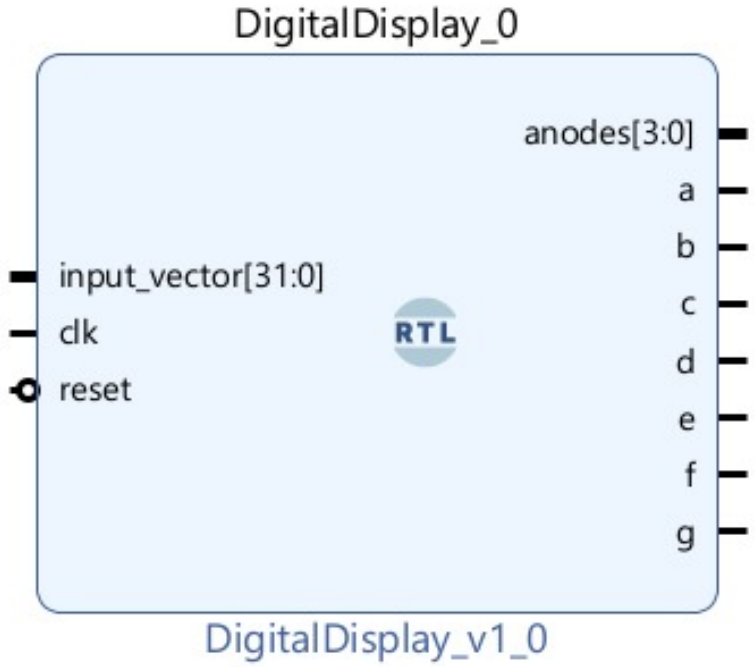

Figure 6: Block Diagram for DigitalDisplay

Figure 7: Block Diagram for InverseShiftRows

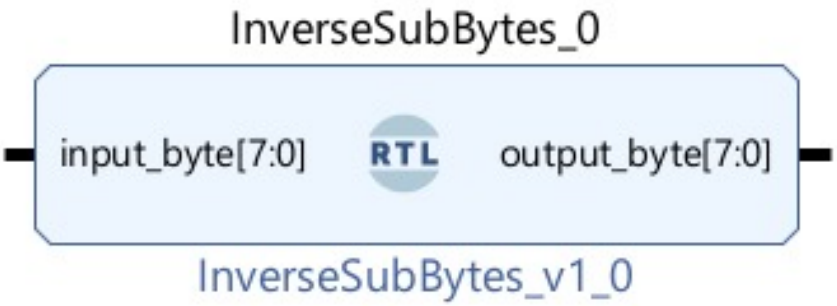

Figure 8: Block Diagram for InverseSubBytes

Figure 9: Block Diagram for InvMixColumns

Figure 10: Block Diagram for xor_gate_8bit

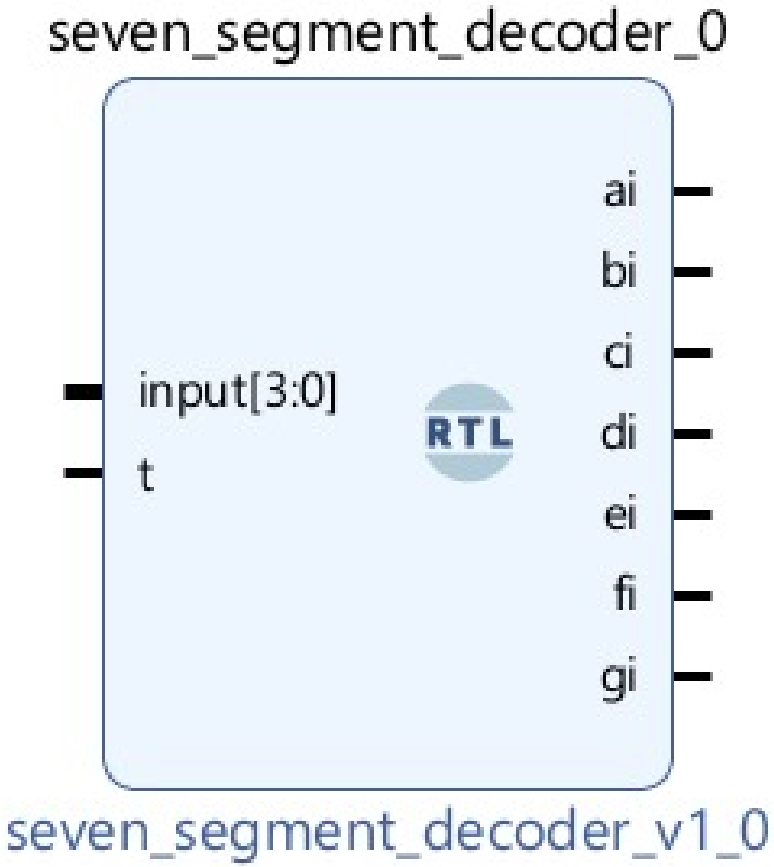

Figure 11: Block Diagram for seven_segment_decoder

5 Finite State Machine (FSM) in AES_decryptor

The FSM in `AES_decryptor` orchestrates decryption transformations. Key states include:

- **IDLE**: Initial state waiting for a start signal.
- **LOAD_CIPHERTEXT**: Loads ciphertext from memory into internal RAM.
- **ADD_ROUND_KEY**: XORs current state with the round key.
- **INVERSE_SUB_BYTES**: Applies inverse S-box substitution.
- **INVERSE_SHIFT_ROWS**: Shifts rows for row reversal.
- **INVERSE_MIX_COLUMNS**: Applies GF(2^8) matrix multiplication to reverse column mixing.
- **DONE_PARTIALLY**: Indicates completion, of decryption, outputs the decrypted contents of Intermediate-RAM.

- **DONE_FULLY**: Indicates completion, signaling `TopModule` to switch modes.

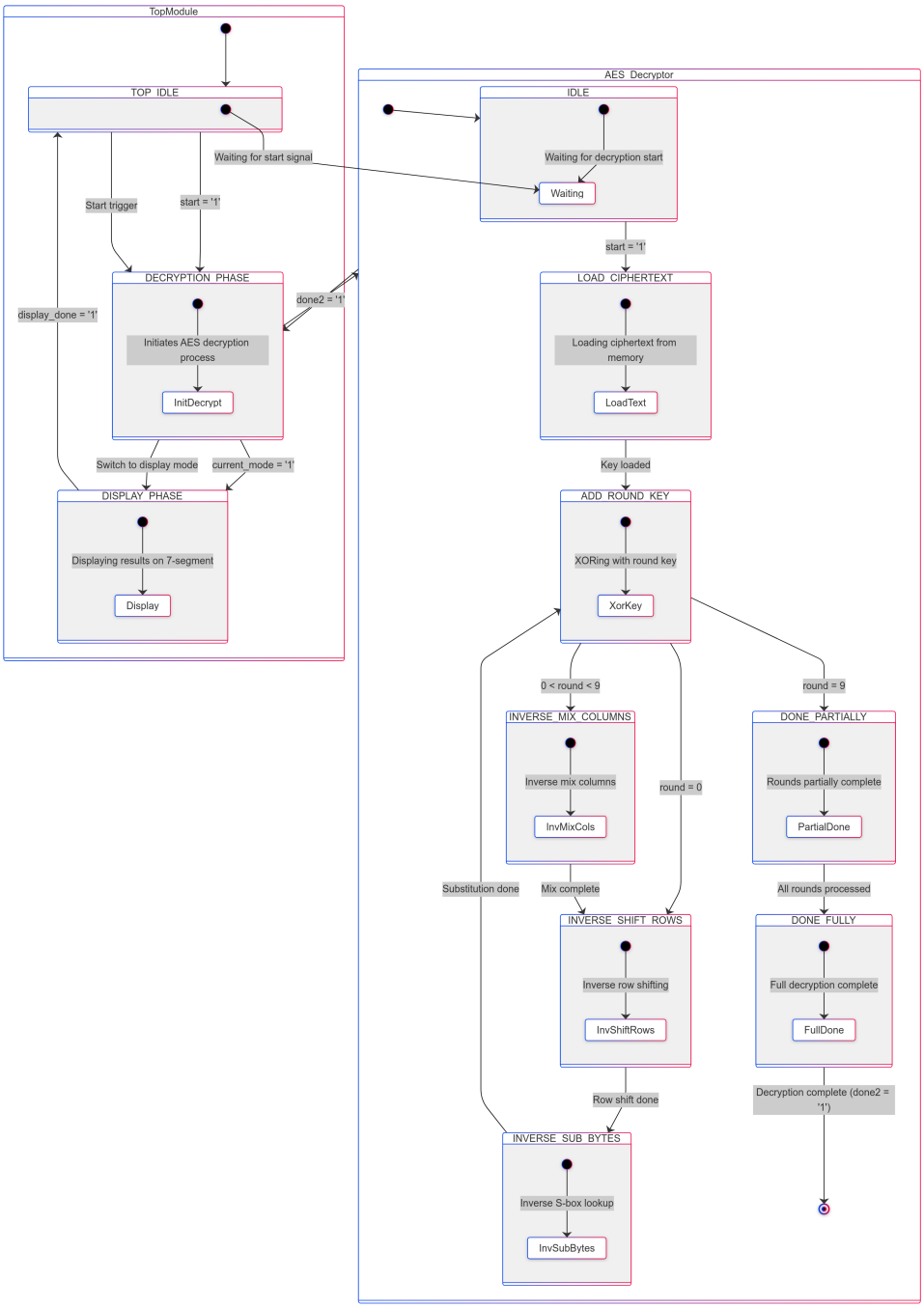

Figure 12: FSM State Diagram for AES Decryption Process

6 Simulation Results and Snapshots

The following waveform demonstrates the behavior of key signals, showing transitions through decryption and display states. Each phase transition is marked by signal changes to ensure stable decryption and display operation.

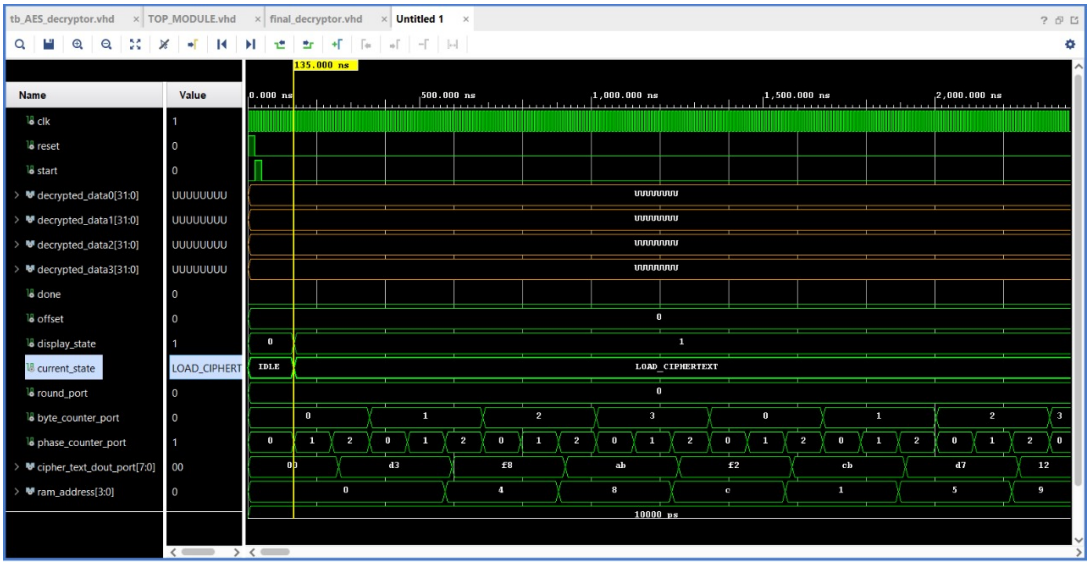

Figure 13: Simulation of FSM with State signals (IDLE to LOAD_CIPHERTEXT

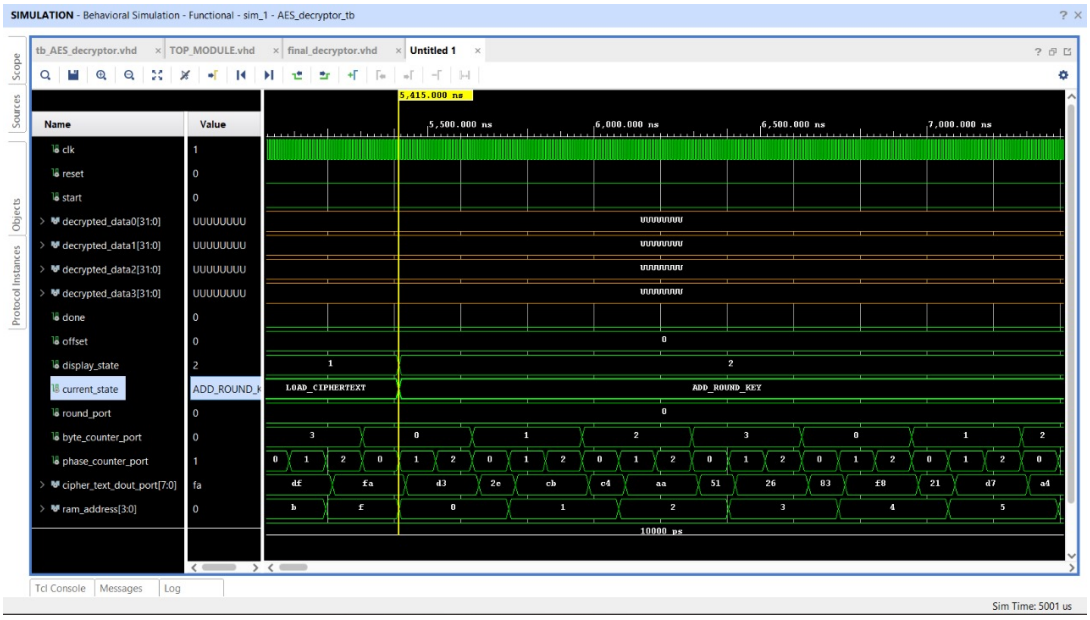

Figure 14: Simulation of FSM with State signals (LOAD_CIPHERTEXT to ADD_ROUND_KEY

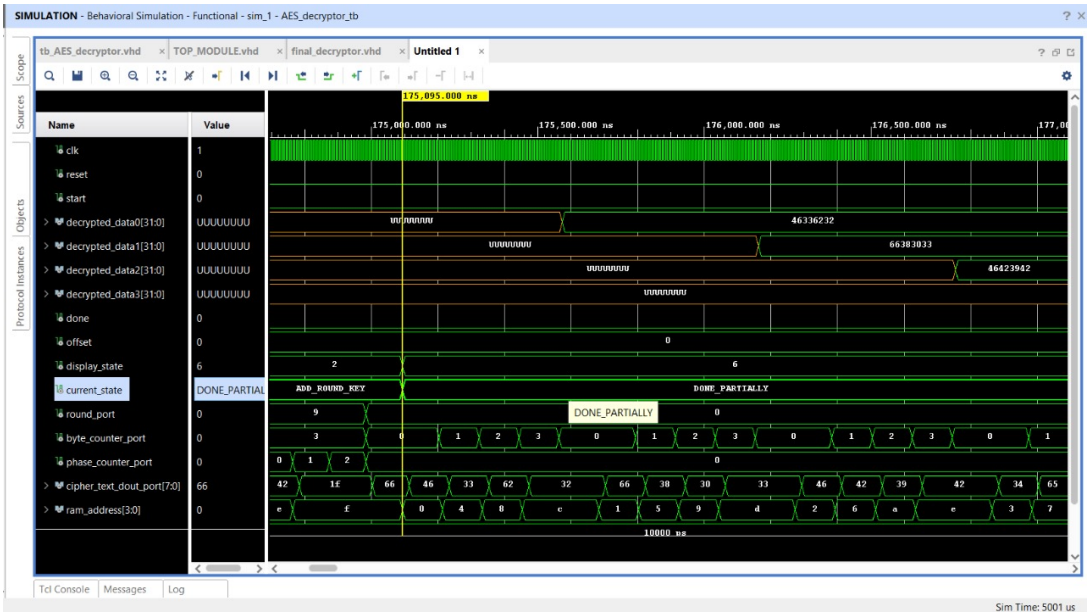

Figure 15: Simulation of FSM with State signals (ADD_ROUND_KEY to DONE_PARTIALLY

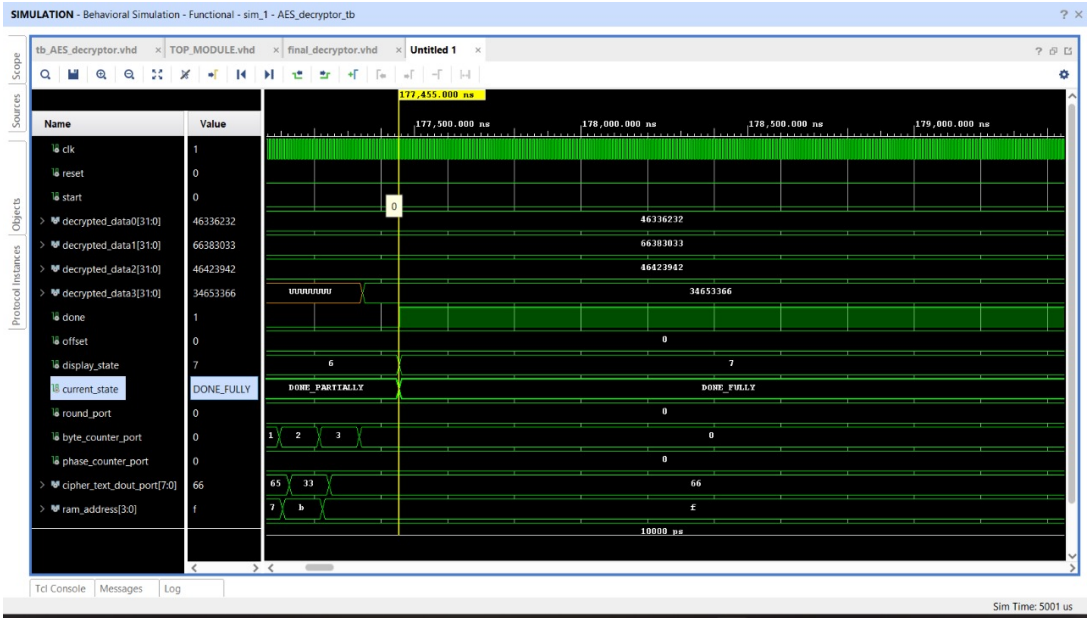

Figure 16: Simulation of FSM with State signals (DONE_PARTIALLY to DONE_FULLY

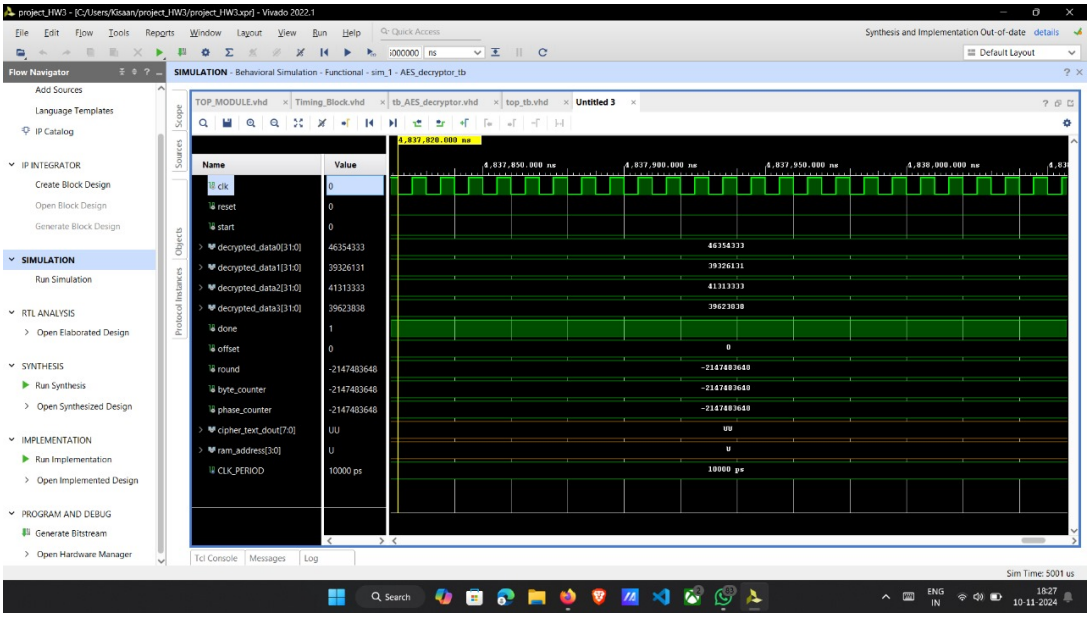

Figure 17: Simulation of FSM with State signals (IDLE to LOAD_CIPHERTEXT

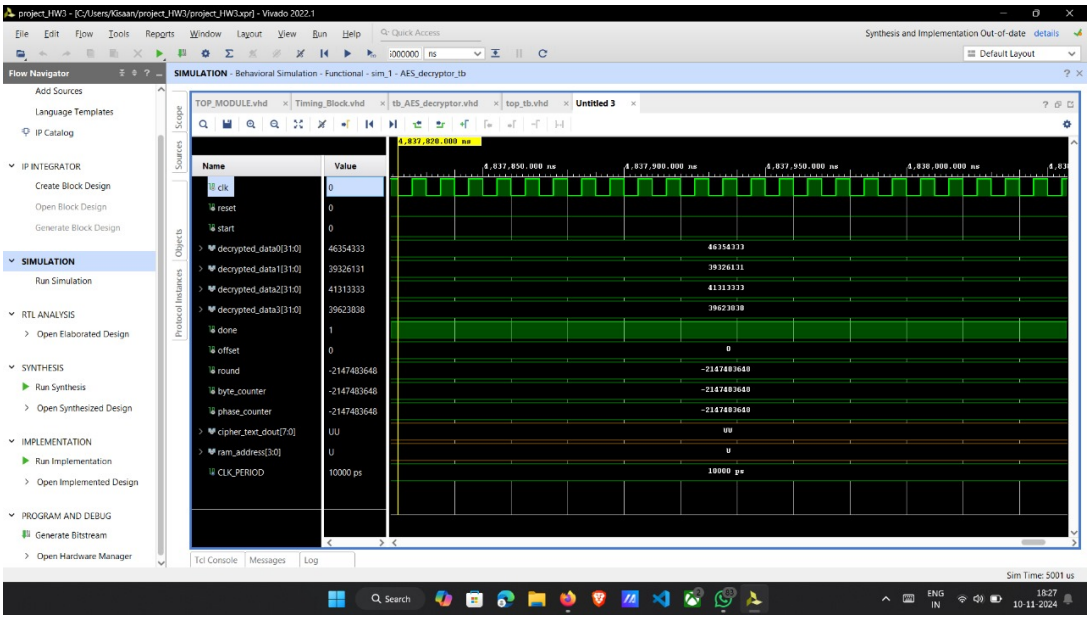

Figure 18: Simulation of AES Decryption System (128-bit input)

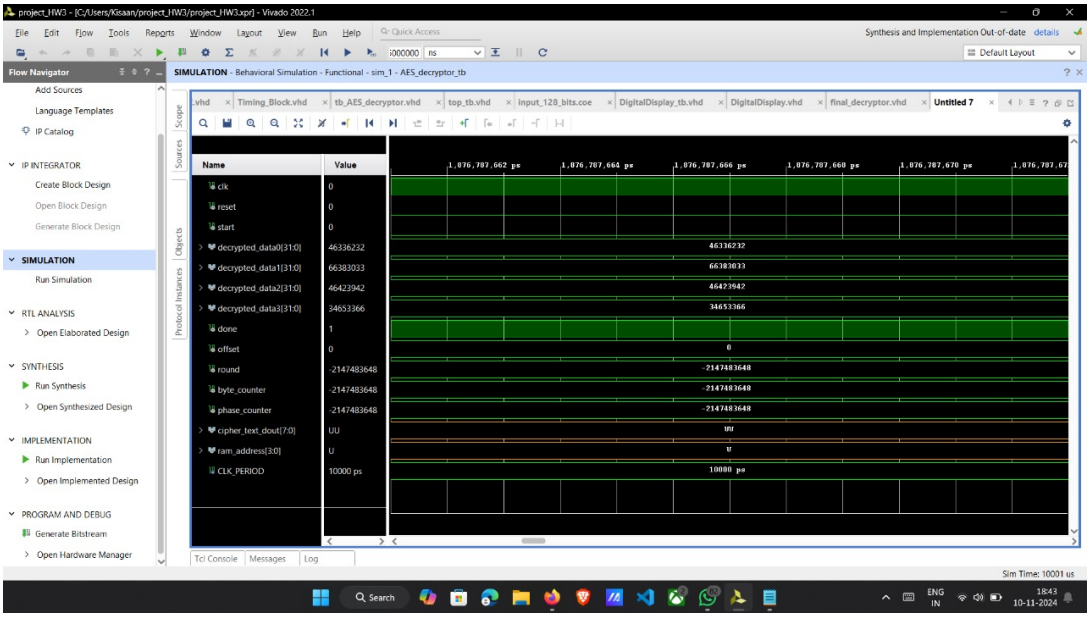

Figure 19: Simulation of AES Decryption System (256-bit input)

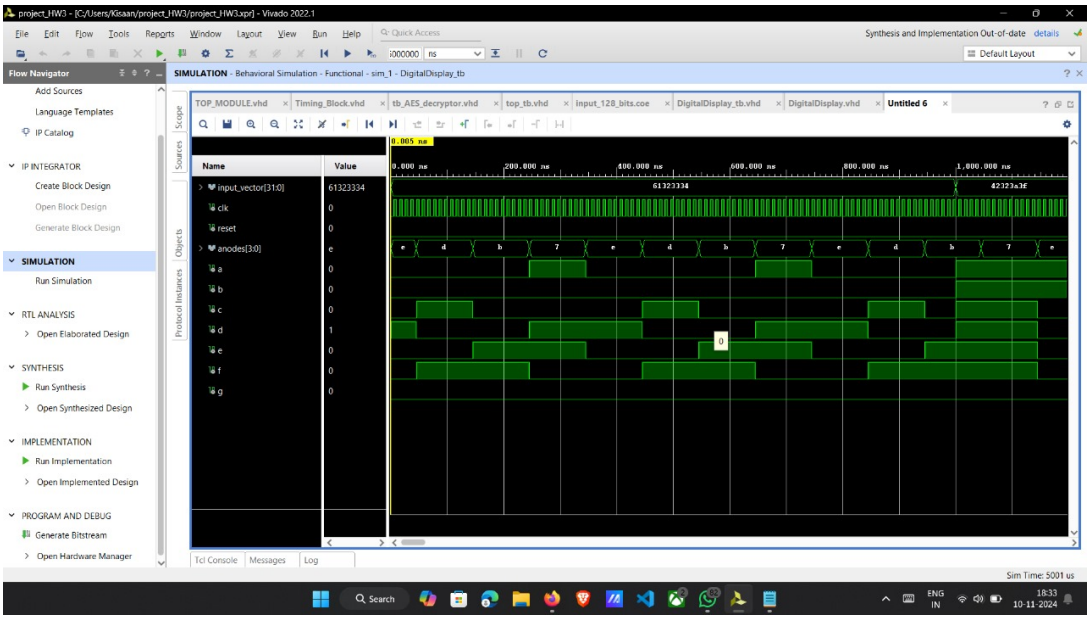

Figure 20: Simulation of Display System (Byte 1)

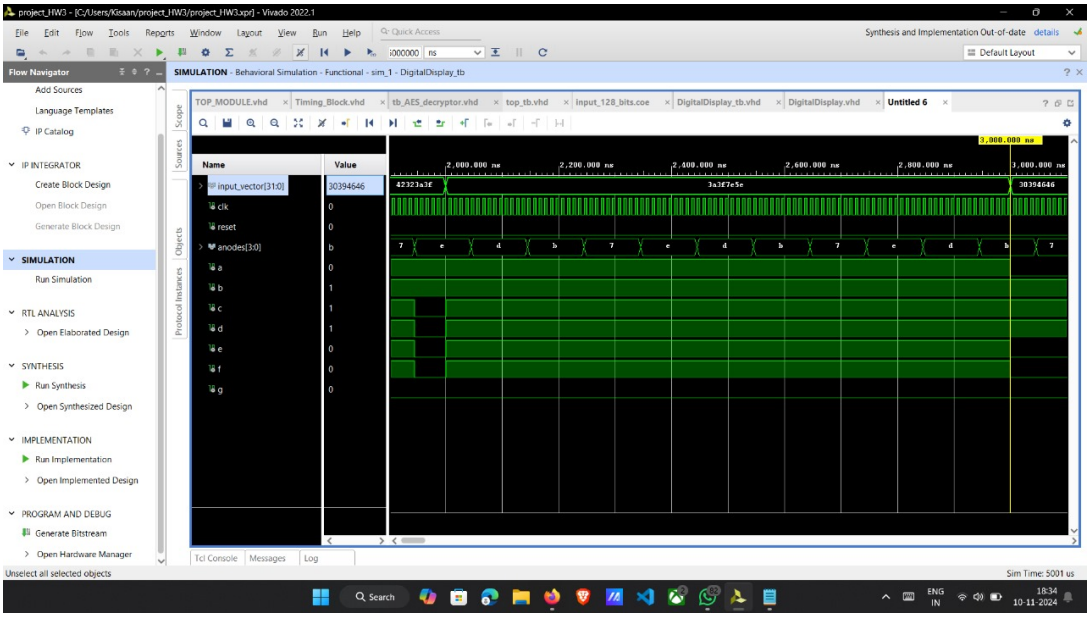

Figure 21: Simulation of Display System (Byte 2)

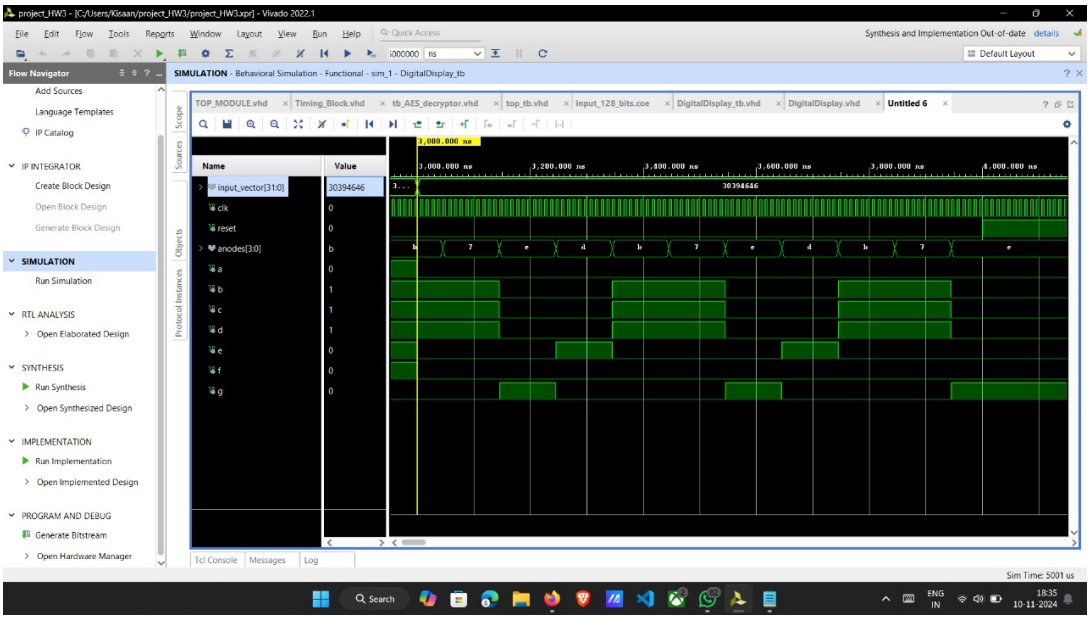

Figure 22: Simulation of Display System (Byte 3)

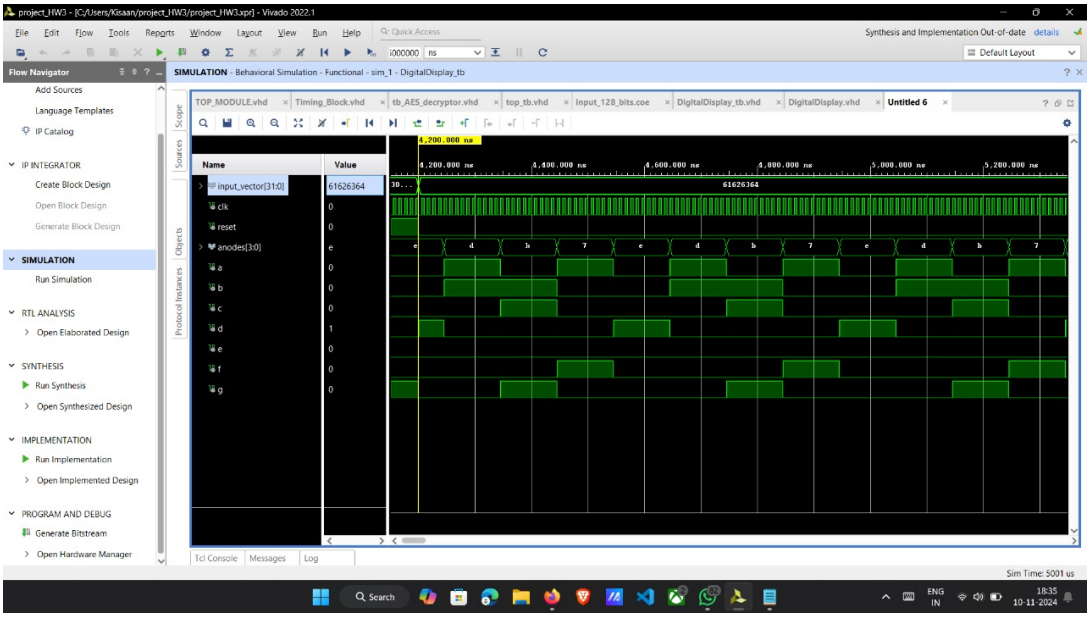

Figure 23: Simulation of Display System (Byte 4)

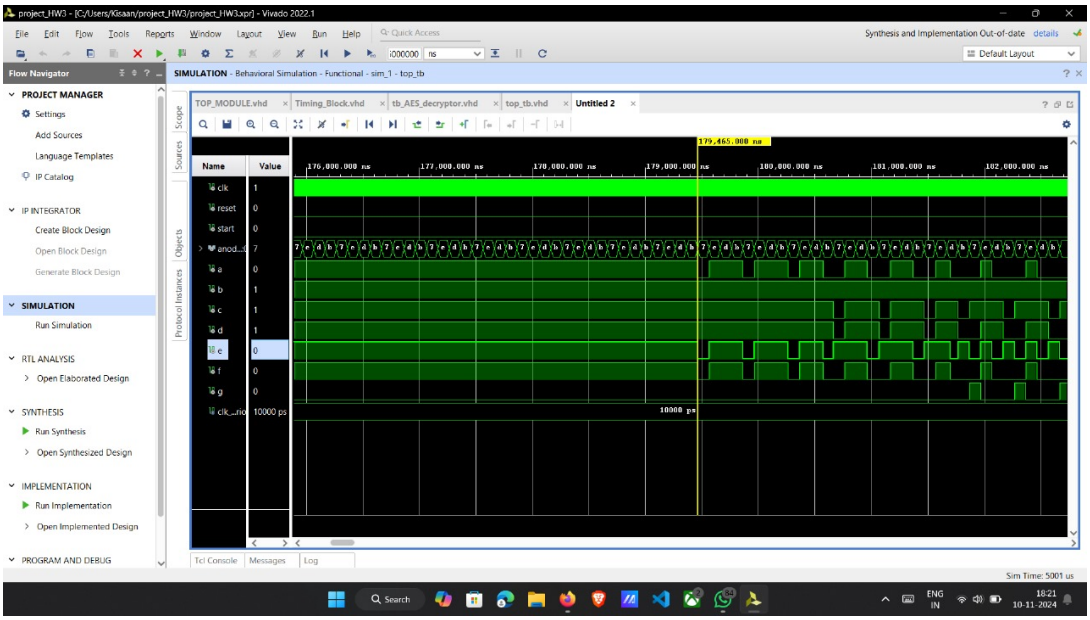

Figure 24: Simulation of TopModule (128-bit) I

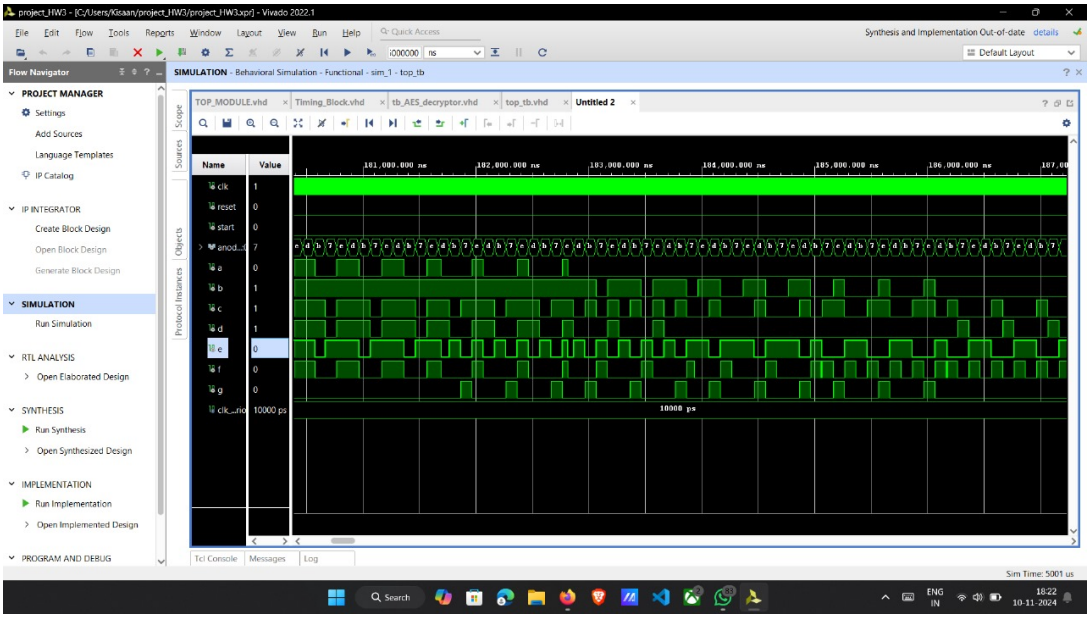

Figure 25: Simulation of TopModule (128-bit) II

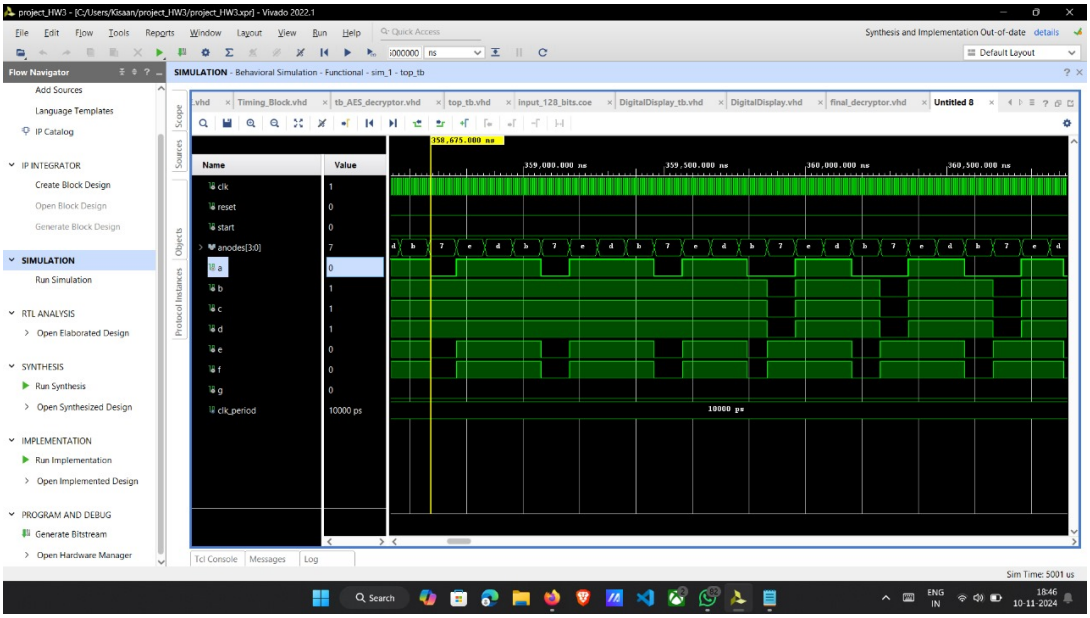

Figure 26: Simulation of TopModule (256-bit) I

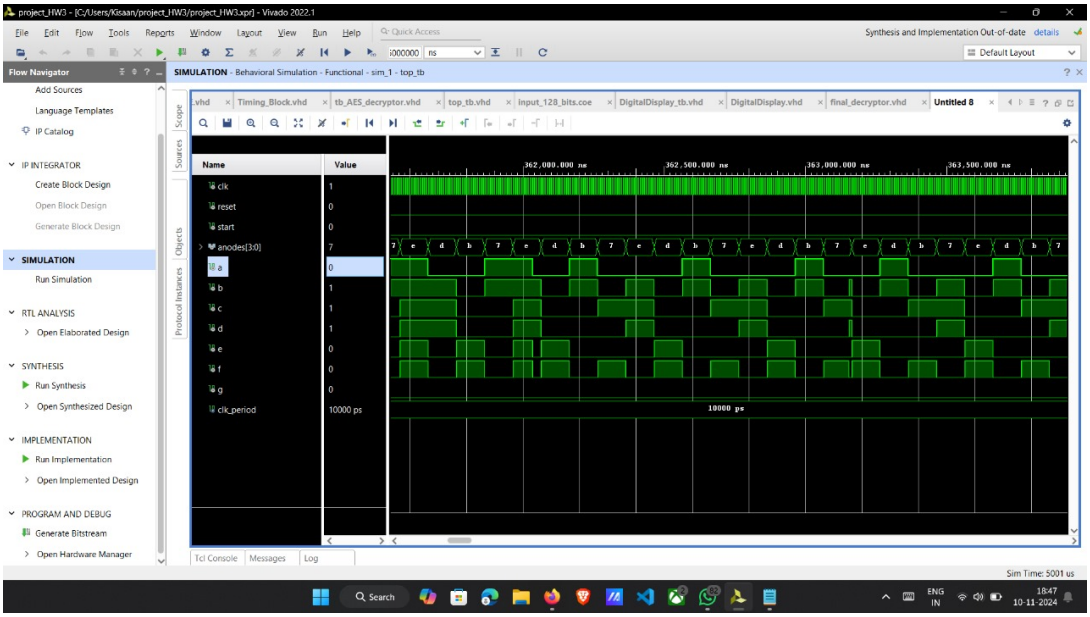

Figure 27: Simulation of TopModule (256-bit) II

7 Schematic Snapshot

The FPGA synthesis schematic verifies correct signal routing and interconnections among the compute units.

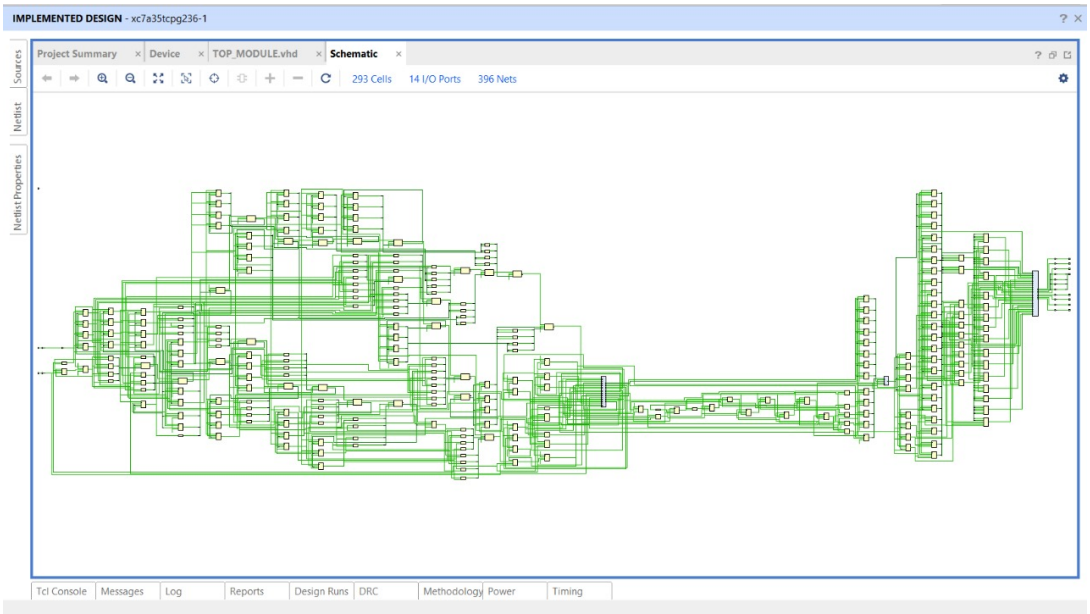

Figure 28: Schematic Snapshot of AES Decryption System

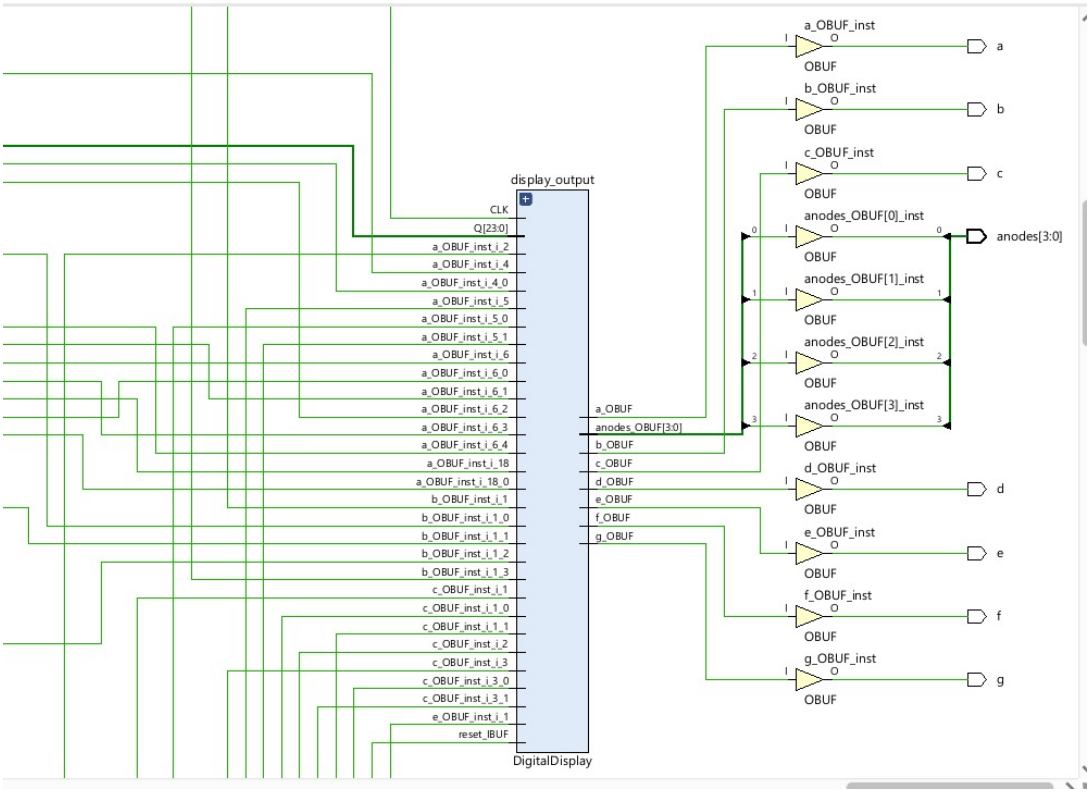

Figure 29: Schematic Zoomed in on Digital Display

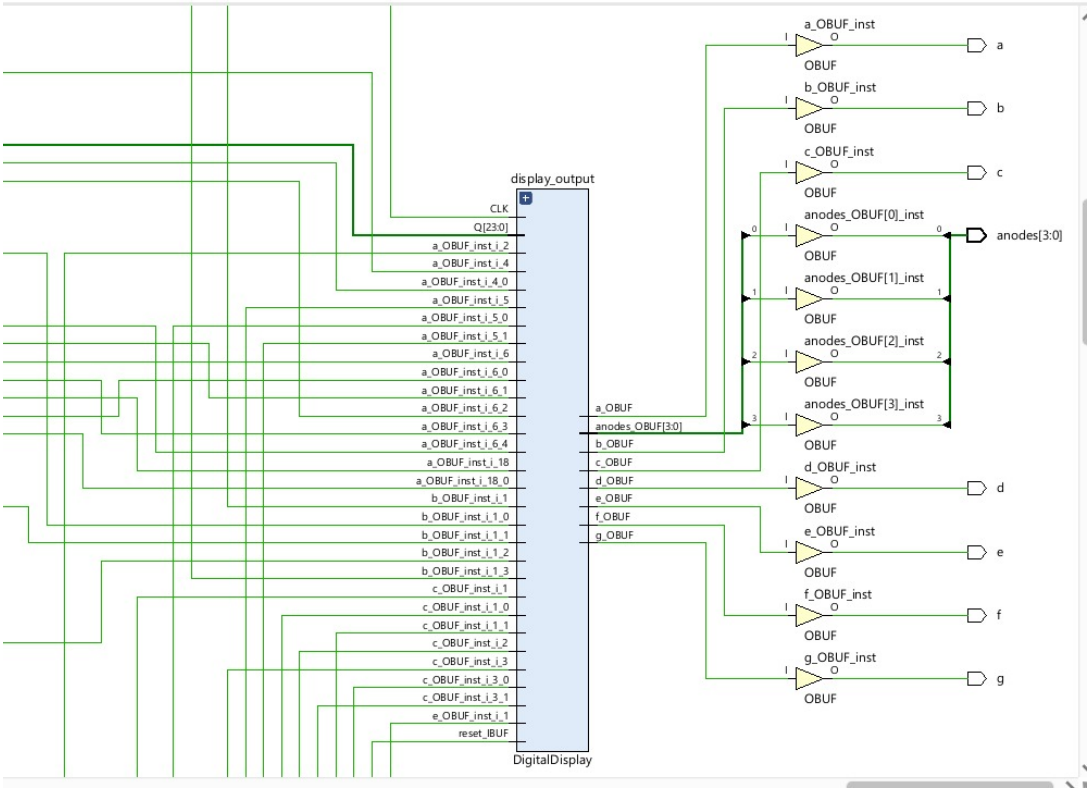

Figure 30: Schematic Zoomed in on AES Decryption System

8 Resource Utilization

The resource usage of the design is summarized in the table below, including FPGA resources used by each compute unit.

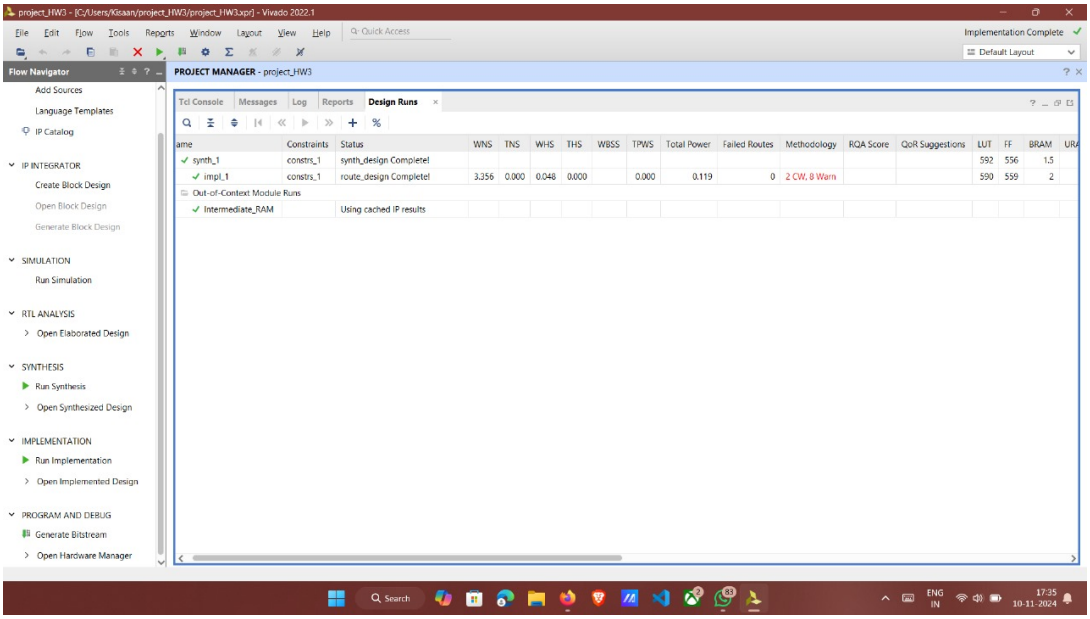

Figure 31: Resource Utilization for Top File (128 bit input)

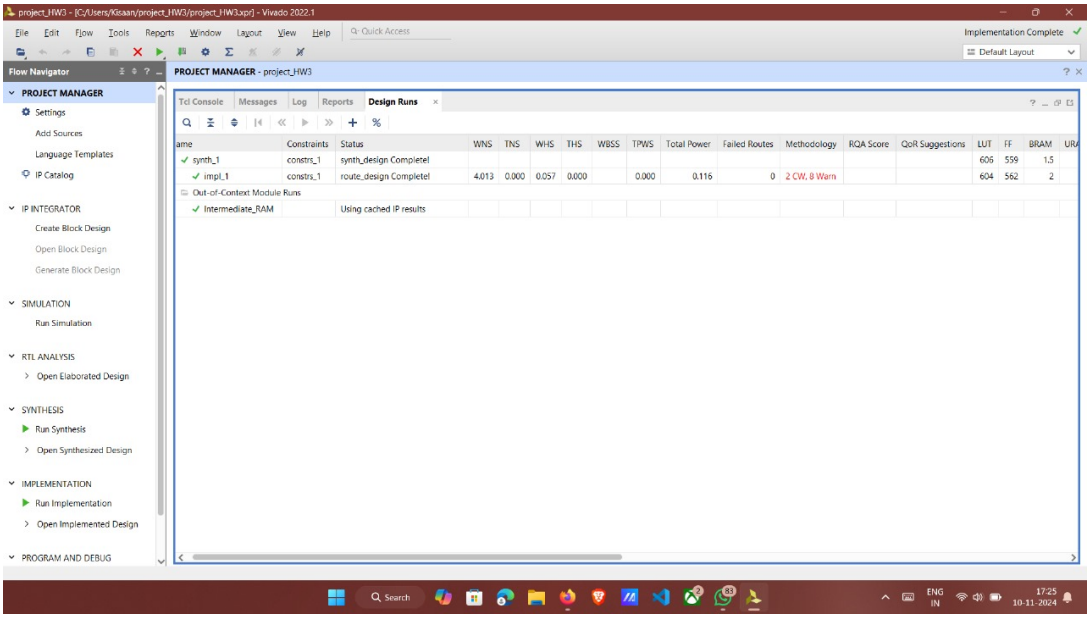

Figure 32: Resource Utilization for Top File (256 bit input)

9 Conclusion

The AES-128 decryption module efficiently implements the AES transformations while handling multiple stages in a pipelined manner. The hierarchical design ensures mod-

ularity, with each compute unit responsible for specific AES transformations or display control. Key accomplishments include:

- Accurate AES-128 decryption with modular design.
- Resource-efficient implementation with appropriate timing control.
- Hierarchical organization for clear signal management and easy debugging.

This design provides a robust foundation for AES decryption on FPGA, suitable for real-time applications requiring secure data handling.

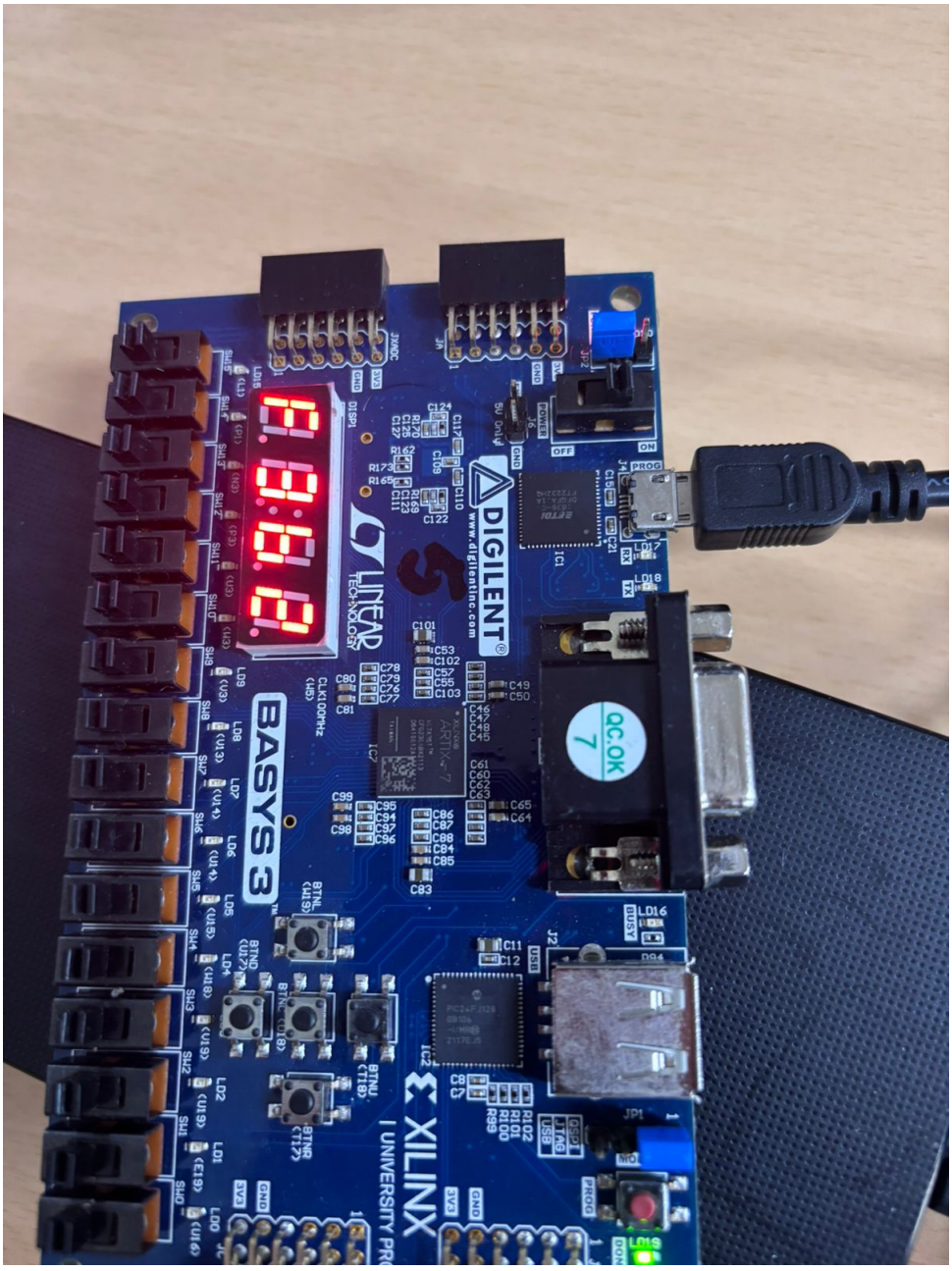

Figure 33: Basys Board Showing Output I

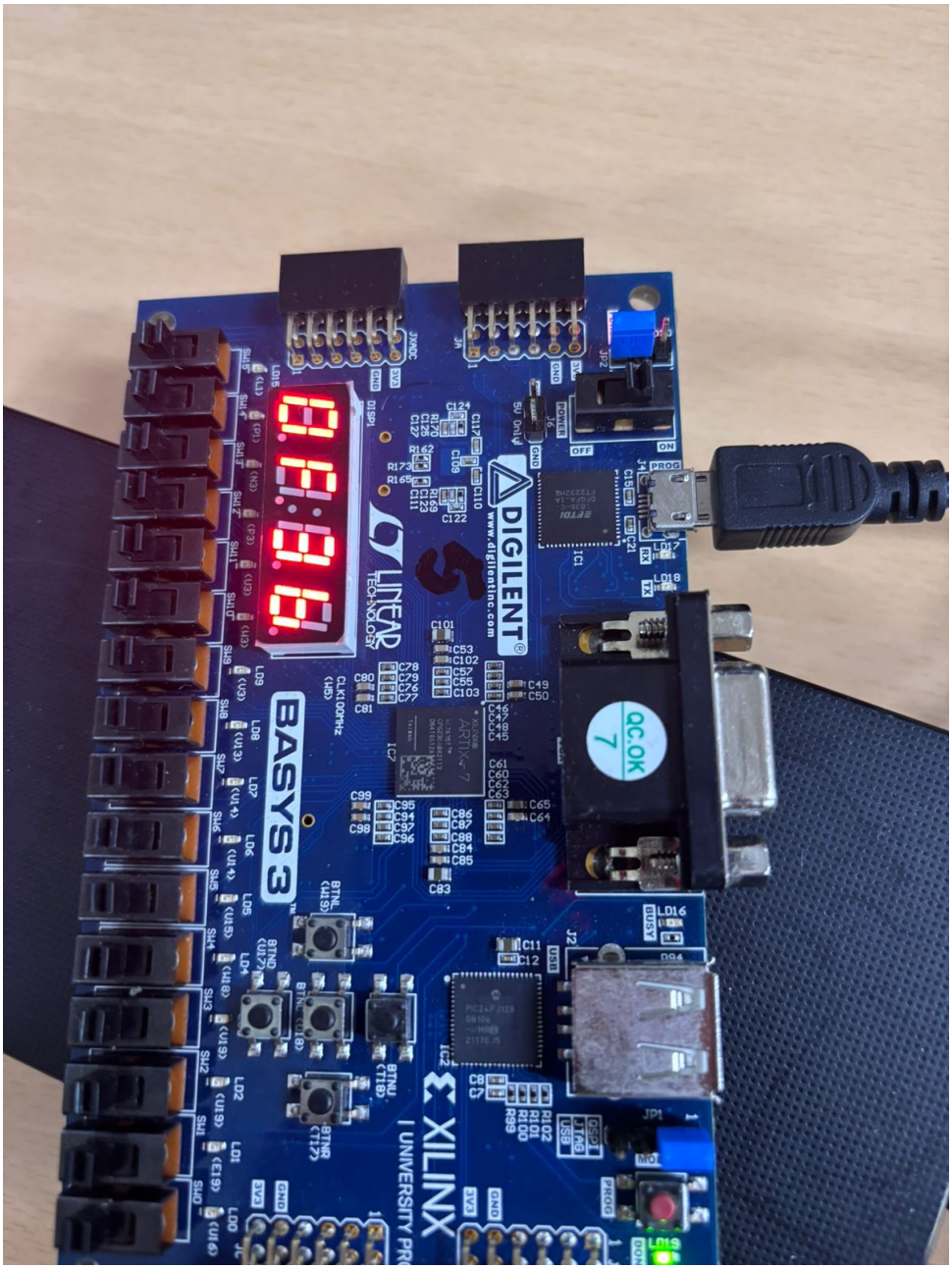

Figure 34: Basys Board Showing Output II