Harshil Rajesh Patel, Ritam Singal

CPEN 311 LAB 4

1. **Directory / Path of .sof File:**
   ./rtl/rc4-lab4-completed.sof

2. **Status of the Lab:**
   - Finished task 1, where memory is initialized to integers from 0 to 255.
   - Finished task 2, where we can decrypt with given secret_key 0x249 to get the correct message.
   - Finished task 3, where we cycle through all possible keys to decrypt messages in given .mif test files.

3. **FSM State Diagrams:**
   a. *[TASK 1] – Initialize s_memory FSM:*

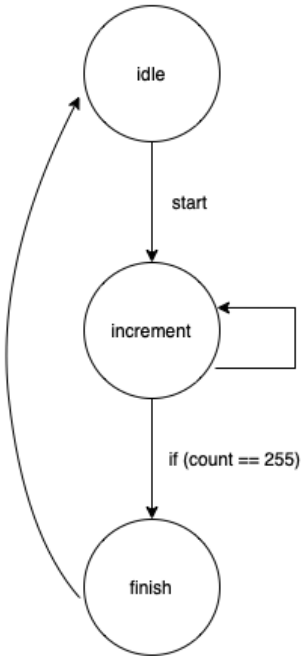

Figure 1: Task 1 State Diagram

b. [TASK 2a] – Task2a FSM:

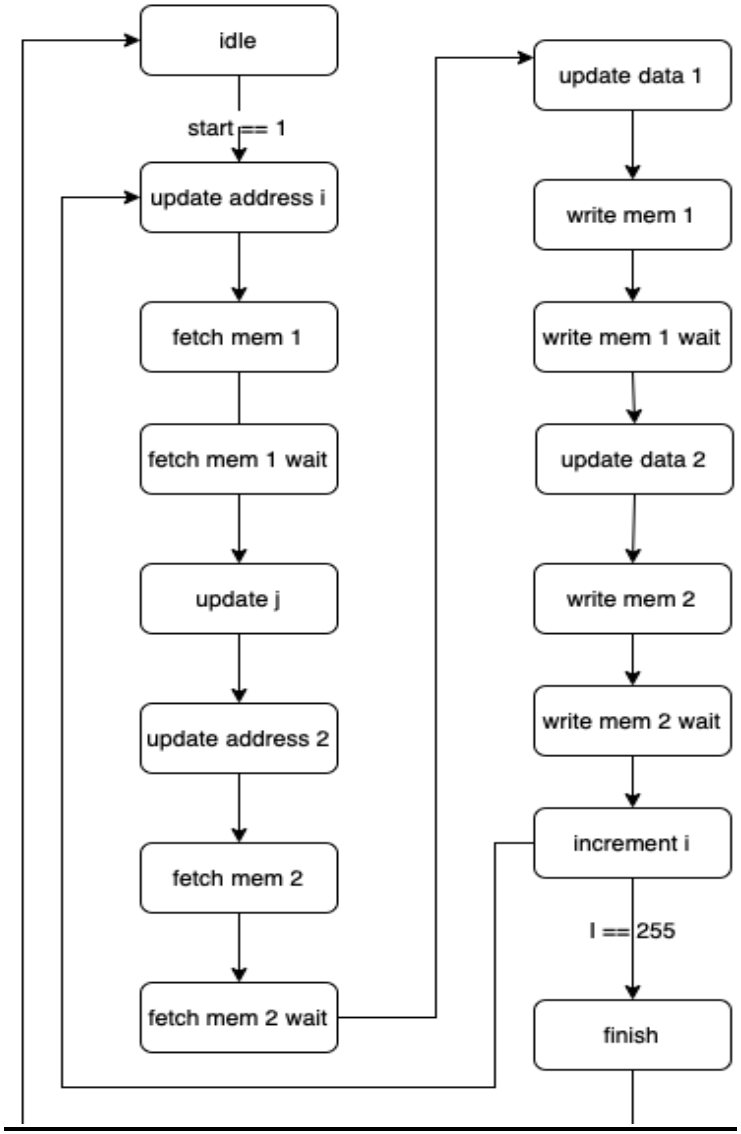

Figure 2: Task2a State Diagram

c. *[TASK 2b] – Task2b FSM:*

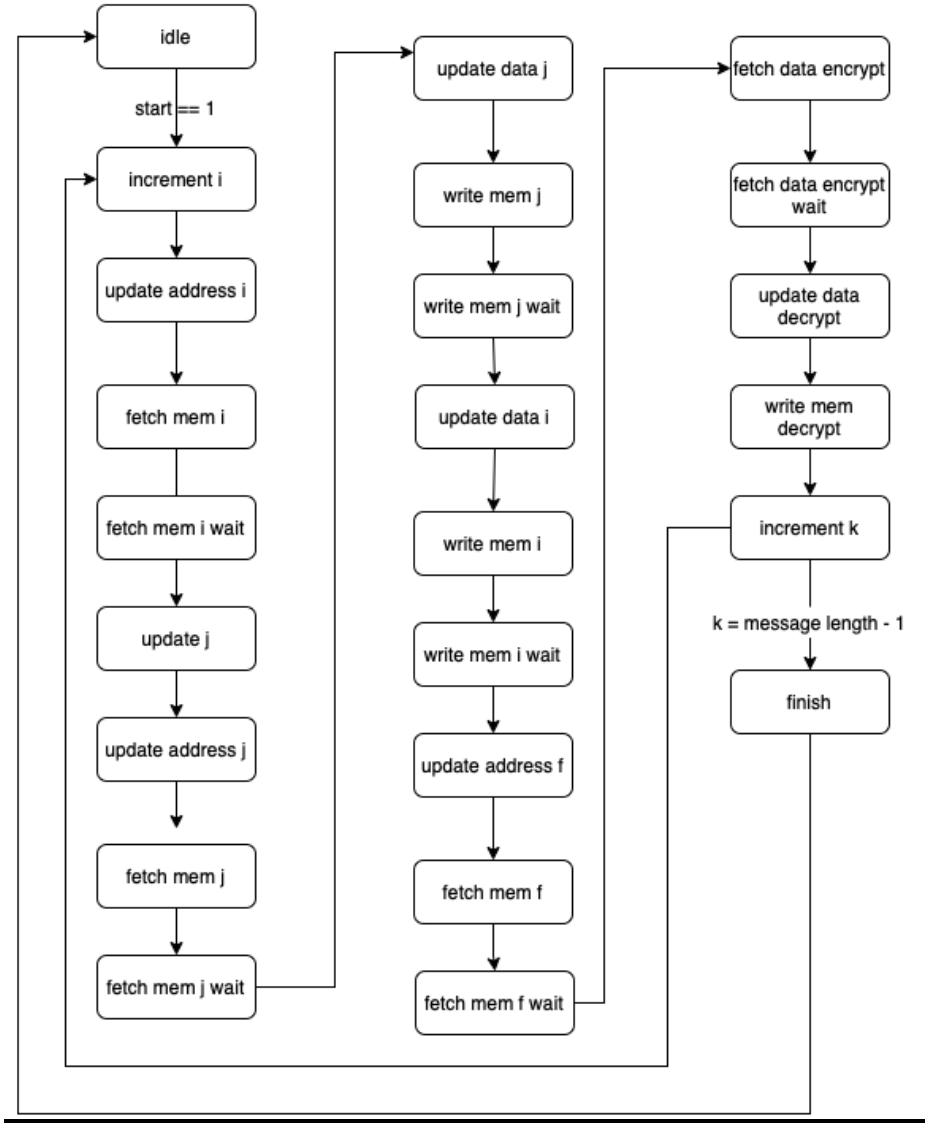

Figure 3: Task 2b State Diagram

d. *[TASK 3] – Task 3 FSM:*

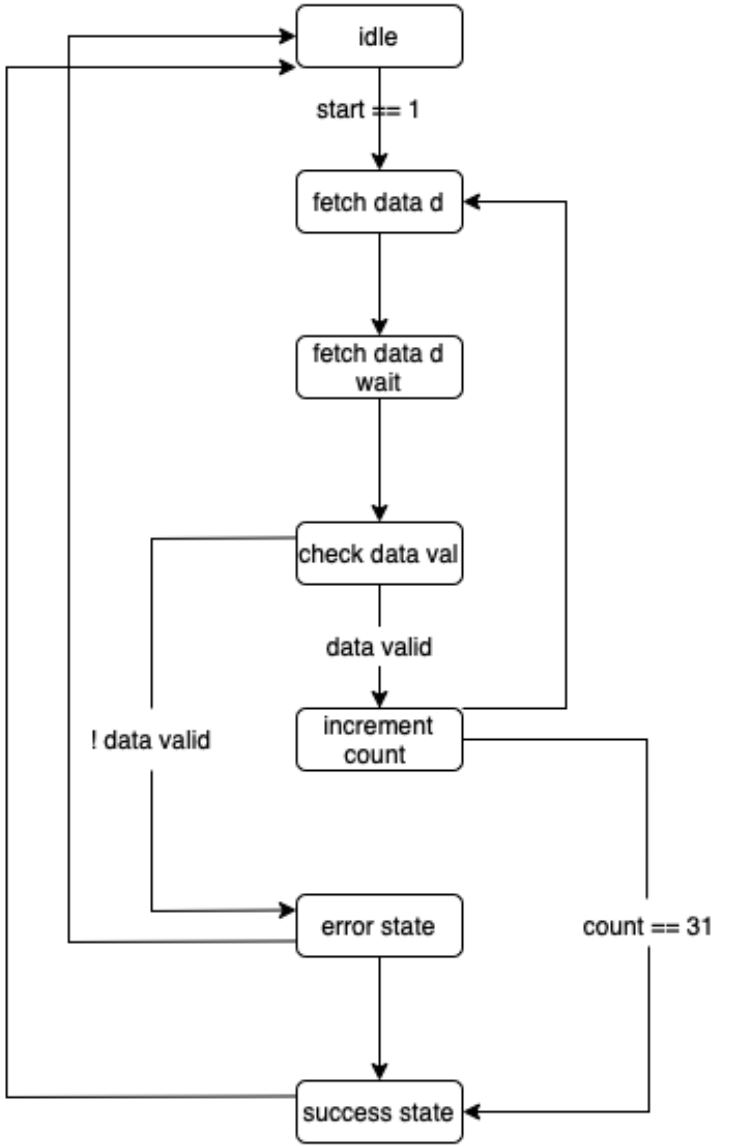

Figure 4: Task 3 State Diagram

4. FSM ModelSim Simulations:

a. *[TASK 1] – Initialize s_memory FSM:*

This FSM is used to perform the code in the first for loop. It is responsible for initializing the memory contents of s_memory to integer values from 0 to 255 at their corresponding addresses. A start signal starts the FSM, and a finished signal indicates that the FSM has completed the task. In the intermediate stages, wen is enabled when writing to memory.

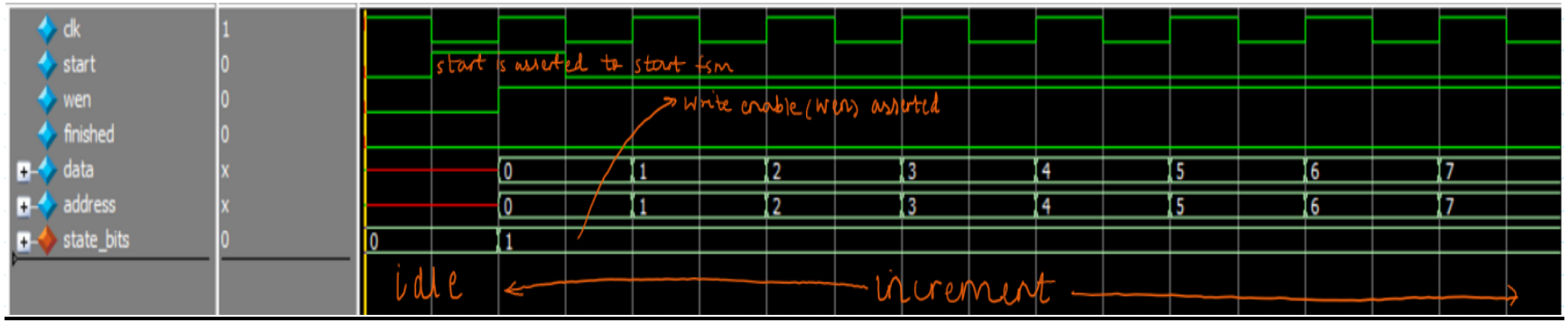

Figure 5: Task 1 FSM Simulation, Part 1

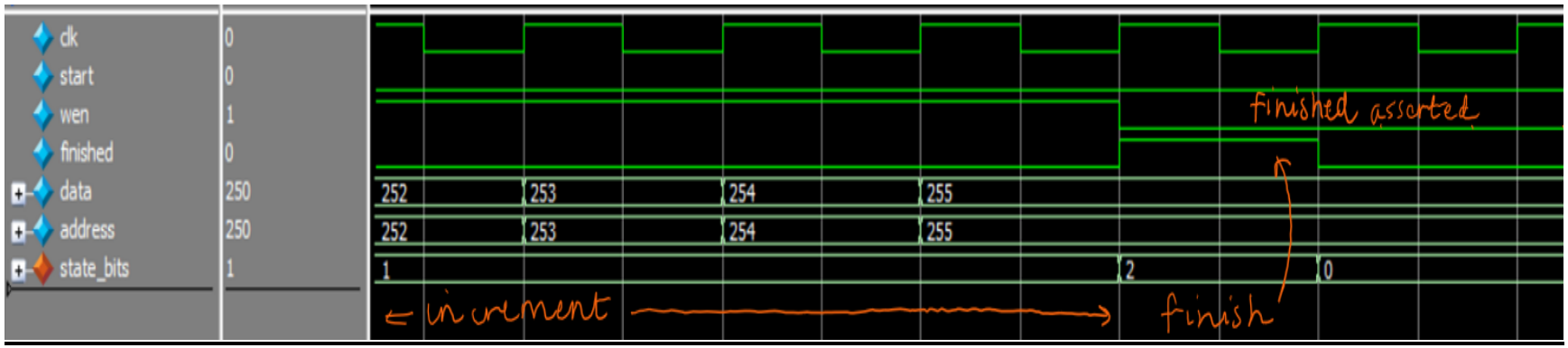

Figure 6: Task 1 FSM Simulation, Part 2

b. *[TASK 2a] – Task2a FSM:*

This FSM is responsible for performing the second for loop in the given code. More specifically, it is responsible for shuffling the contents of the memory that were initialized in the first task. It consists of a start signal that begins the FSM and a finish signal that signifies that it has completed the task. In the intermediate states, we write to addresses i, j and swap values of s[i], s[j]. That requires wen (write enable) to be asserted to write to memory.

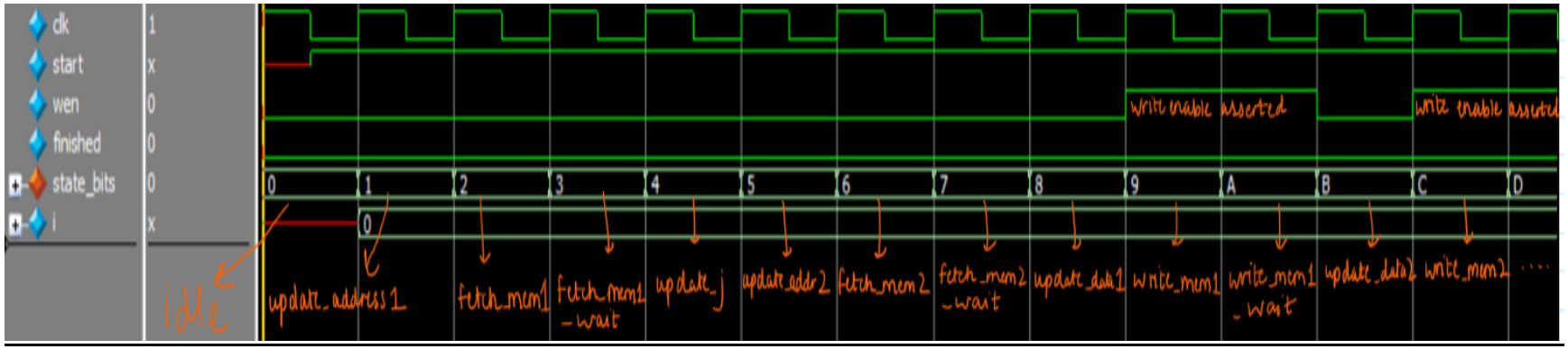

Figure 7: Task 2a FSM Simulation, Part 1

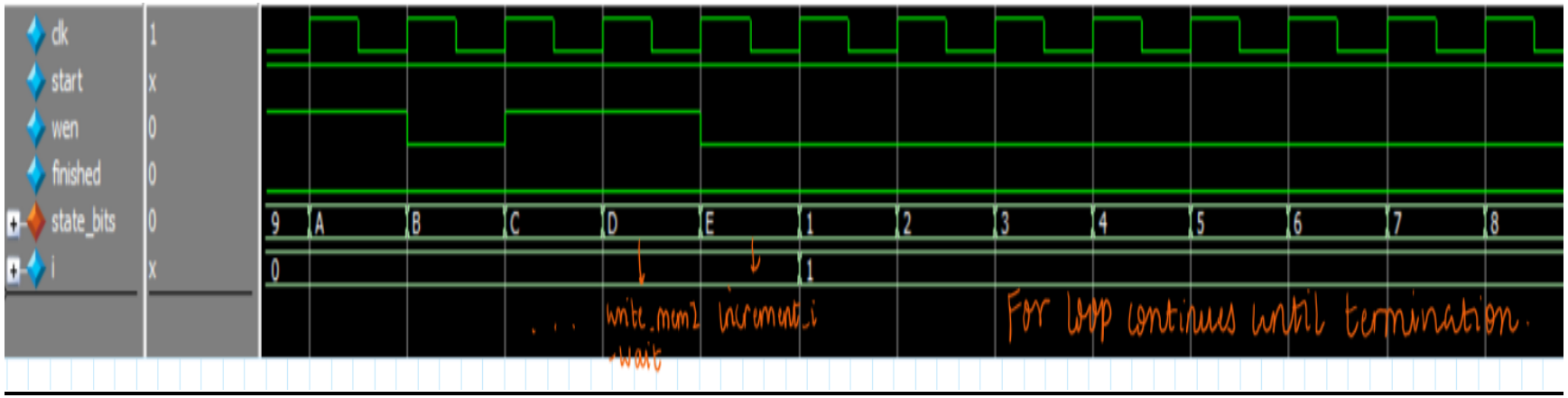

Figure 8: Task 2a FSM Simulation, Part 2

c. *[TASK 2b] – Task2b FSM:*
This FSM is responsible for performing the last part of the given code (the decryption of the message). It deals with data and addresses from the s_memory (Initialized RAM), e_memory (Encryption ROM), and the d_memory (Decrypted RAM). It begins when start is asserted, asserts finished when it completes. In addition, wen is asserted when writing to the s_memory and wren_d is asserted when writing to d_memory.

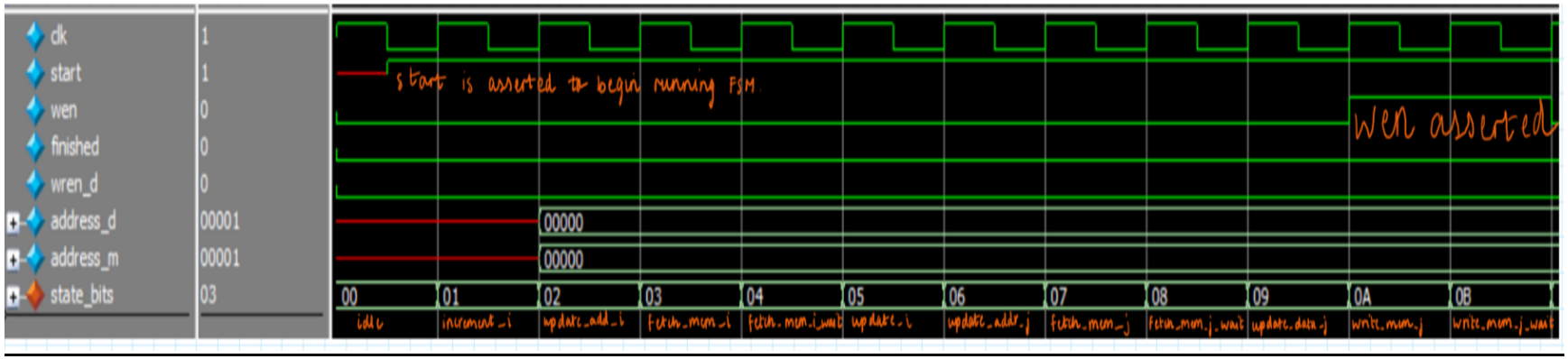

Figure 9: Task 2b FSM Simulation, Part 1

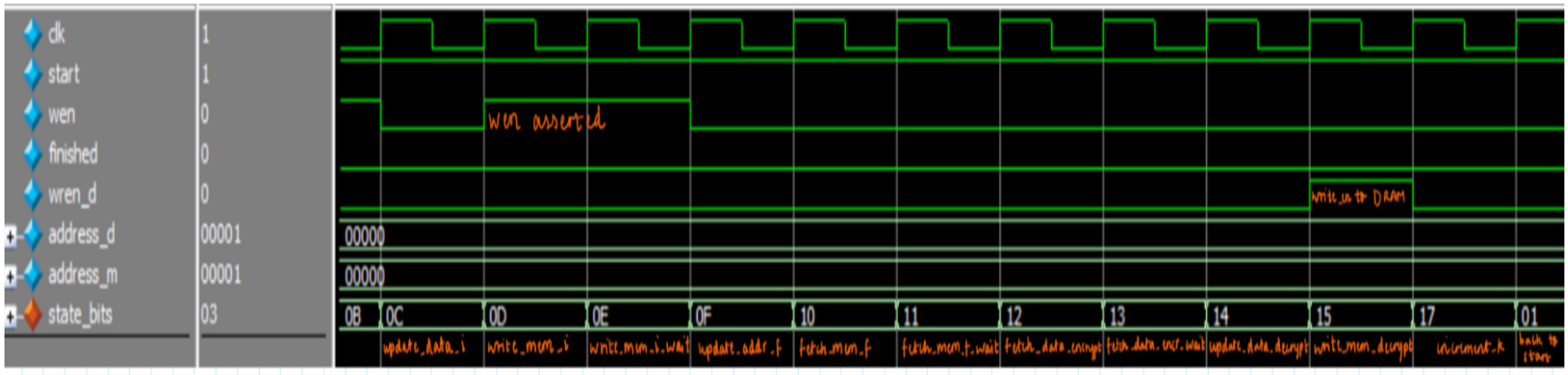

Figure 10: Task 2b FSM Simulation, Part 2

d. *[TASK 3] – Task3 FSM:*
This FSM is responsible for checking if a given secret_key can help decrypt the message into the correct output. It sends the finished signal which turns on LED0 on the DE1_SOC. Otherwise, the error signal is asserted, and we move on to decrypt using the next secret_key or stop at the last iteration.

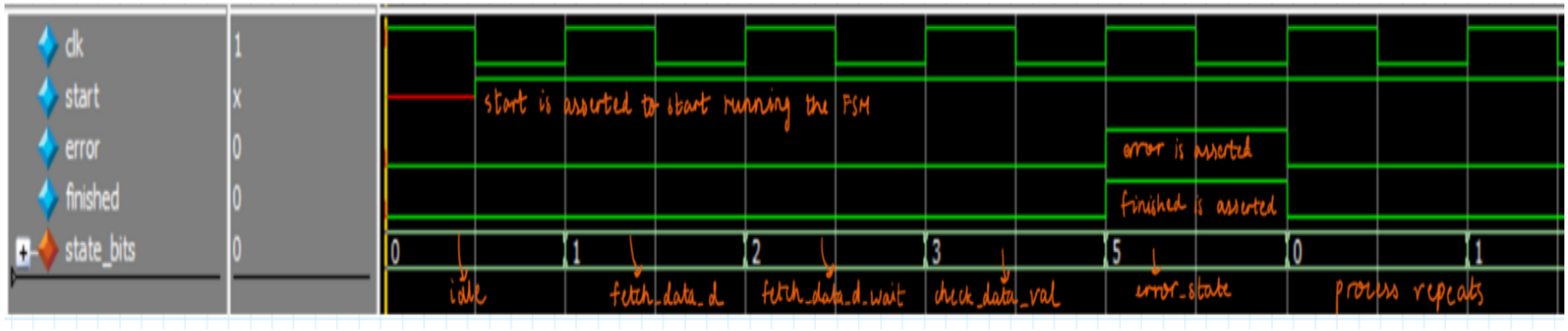

Figure 11: Task 3 FSM Simulation

5. FSM SignalTap:

a. *[TASK 1] SignalTap:*
The first half of the SignalTap instance shows the address, and count being calculated. The second half with the zigzags and the wen asserted show that memory is being written to (too many addresses are being written to, and very fast which is why it appears distorted, we intentionally tried to capture this section on SignalTap).

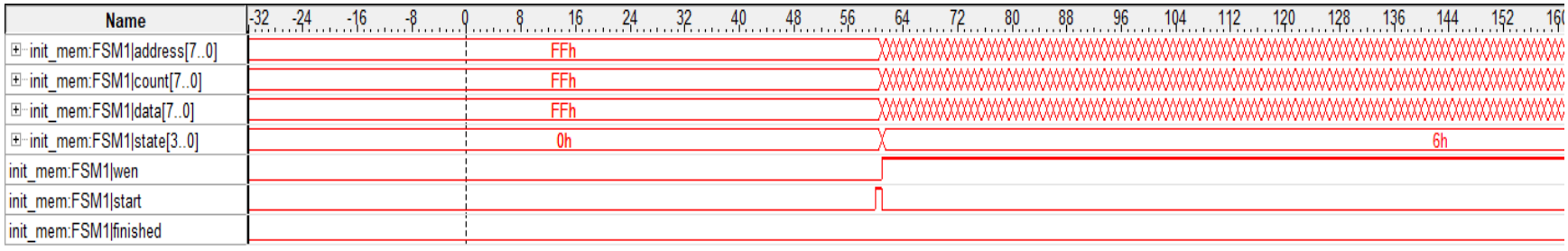

Figure 12: Task 1 FSM SignalTap

b. *[TASK 2a] – SignalTap:*
The following instance shows the signals involved with shuffling the memory contents initialized in Task1,

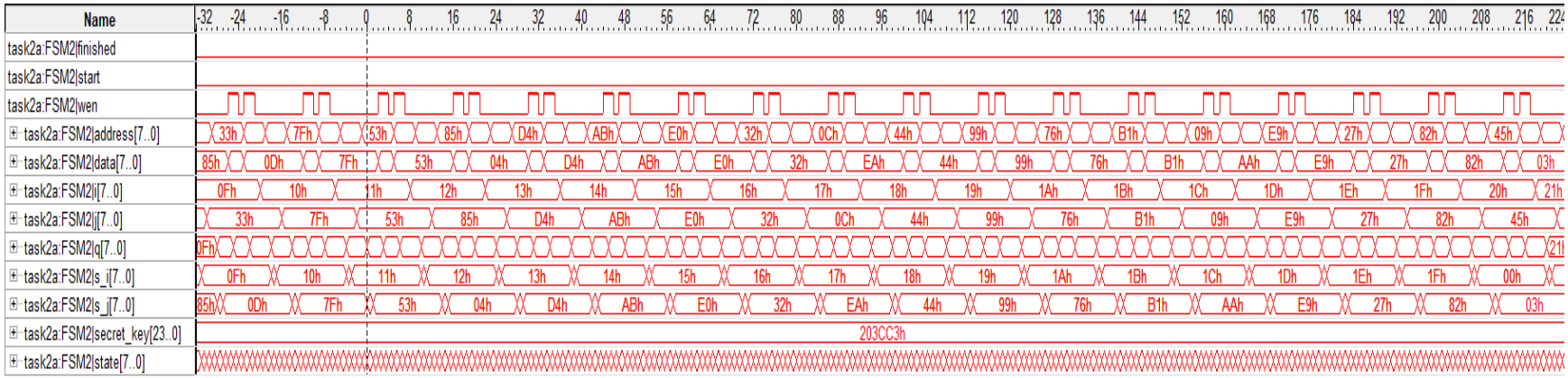

Figure 13: Task 2a FSM SignalTap

c. *[TASK2b] – SignalTap:*
The following instance shows the signals involved with decrypting the encrypted message using a secret_key.

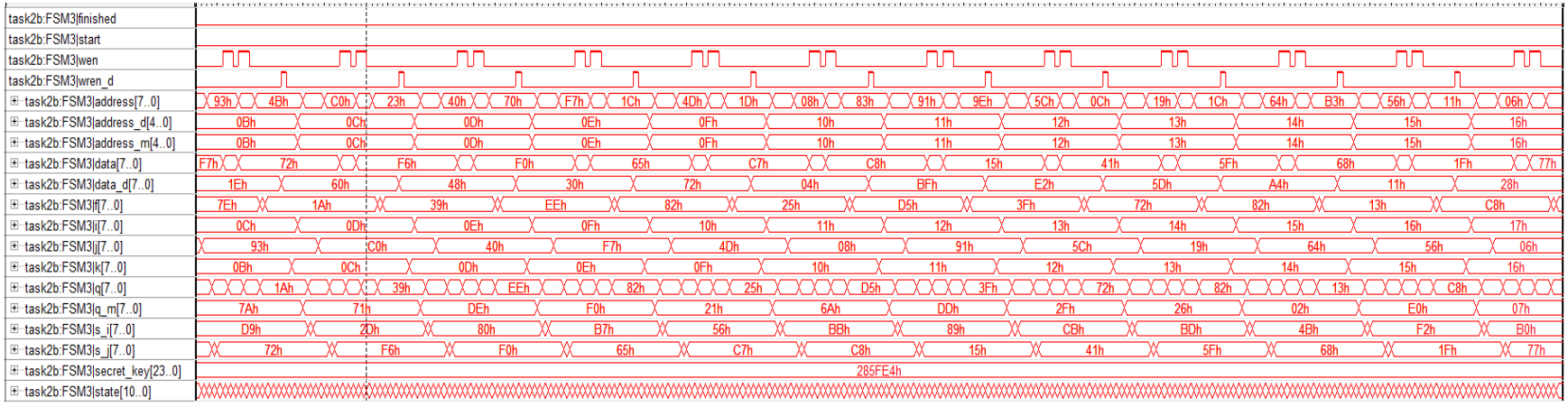

Figure 14: Task 2b FSM SignalTap

d. *[TASK3] – SignalTap:*

The following signals are involved in checking if a decrypted message is correct.

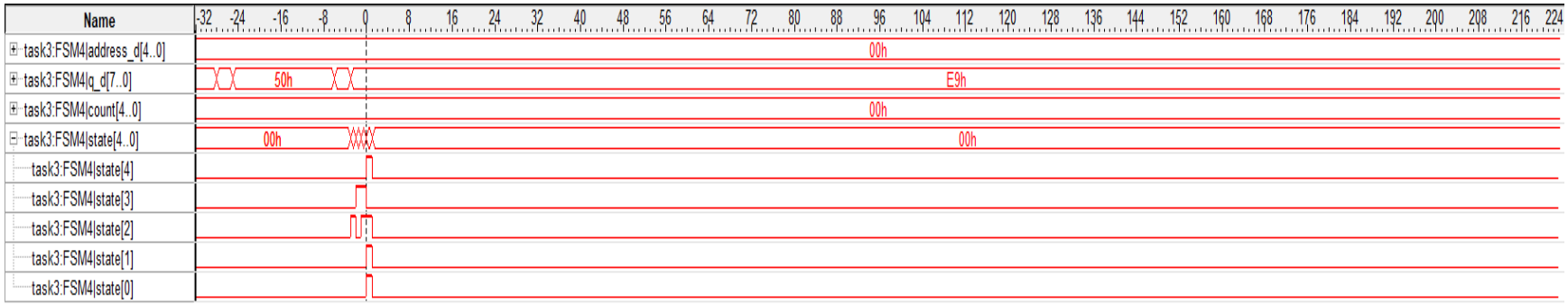

Figure 15: Task 3 FSM SignalTap

6. Additional Info

- All tasks were done using Live Share on VSCode.
- Both partners contributed equally to the completion of this lab.
- Both partners worked together on all tasks, together with paired programming.