Lab Assignment Report 7

Course: COL215 – Digital Logic & System Design

Department of Computer Science & Engineering, IIT Delhi

Semester III, 2025-26

Students: Aryan Patel (2024CS50489)

Rohit Shakya (2024CS10410)

Introduction:

In this assignment, we aim to design and implement a singly linked list in **Verilog HDL**. Since dynamic memory allocation is not supported in hardware, the linked list is implemented using arrays to store node data and pointers. The design supports insertion at both the head and the tail of the list, deletion of any node by value, and traversal of the entire list.

Additionally, hardware indicators are used to represent special conditions — if an insertion is attempted when the list is full, **LED[0]** lights up to indicate an **overflow** condition, and if a deletion is attempted when the list is empty, **LED[1]** lights up to indicate an **underflow** condition.

Design Decision:

- **NULL**: A localparam named NULL is used to represent the absence of a next pointer. It is assigned the value NUM_NODES, indicating that no valid node follows.
- **Width:** The width of the pointer array and all pointer-related signals is defined as a global parameter, ensuring consistency across the design.
- **Storing data and pointer to next node:** Two arrays are used to represent the linked list:

1. data_array: stores the 8-bit data for each node.
2. next_array: stores the pointer to the next node.
   Additionally, next_array helps track and return the pointer of the next available free node.

- **Initialization (btnC):** The **BTNC** button is pressed before any operation to initialize the list. During initialization:

1. next_array[i] is set to i + 1 for all nodes, and next_array[NUM_NODES-1] is set to NULL.
2. head and tail pointers are set to NULL.
3. Next ptr, LEDs, and all registers are reset to 0.

- **Edge Detection:** Positive-edge detection is implemented for all operations (insert, delete, and traverse). This ensures that each button press triggers the corresponding operation only once, preventing multiple unintended executions.
- **Insertion:** Before inserting, the design checks whether free space is available by examining the next pointer.
    1. If next equals NULL, it indicates that the list is full, and **overflow** is signaled via **LED[0]**.
    2. Otherwise, the data is written into data_array, the node's next pointer is updated, and the head and tail pointers are adjusted accordingly.
    3. When inserting the first element, both head and tail are updated to point to the new node.
- **Deletion:**
    1. If head equals NULL, it means the list is empty, and **underflow** is indicated via **LED[1]**.
    2. Otherwise, the deletion process begins with current = head and previous = NULL.
    3. The design sequentially searches for the node containing the target value.
    4. Once found, the previous node's next pointer is updated to skip the deleted node, and the deleted node is added back to the free list by updating next_reg.
    5. If not found, traversal continues until the end of the list.

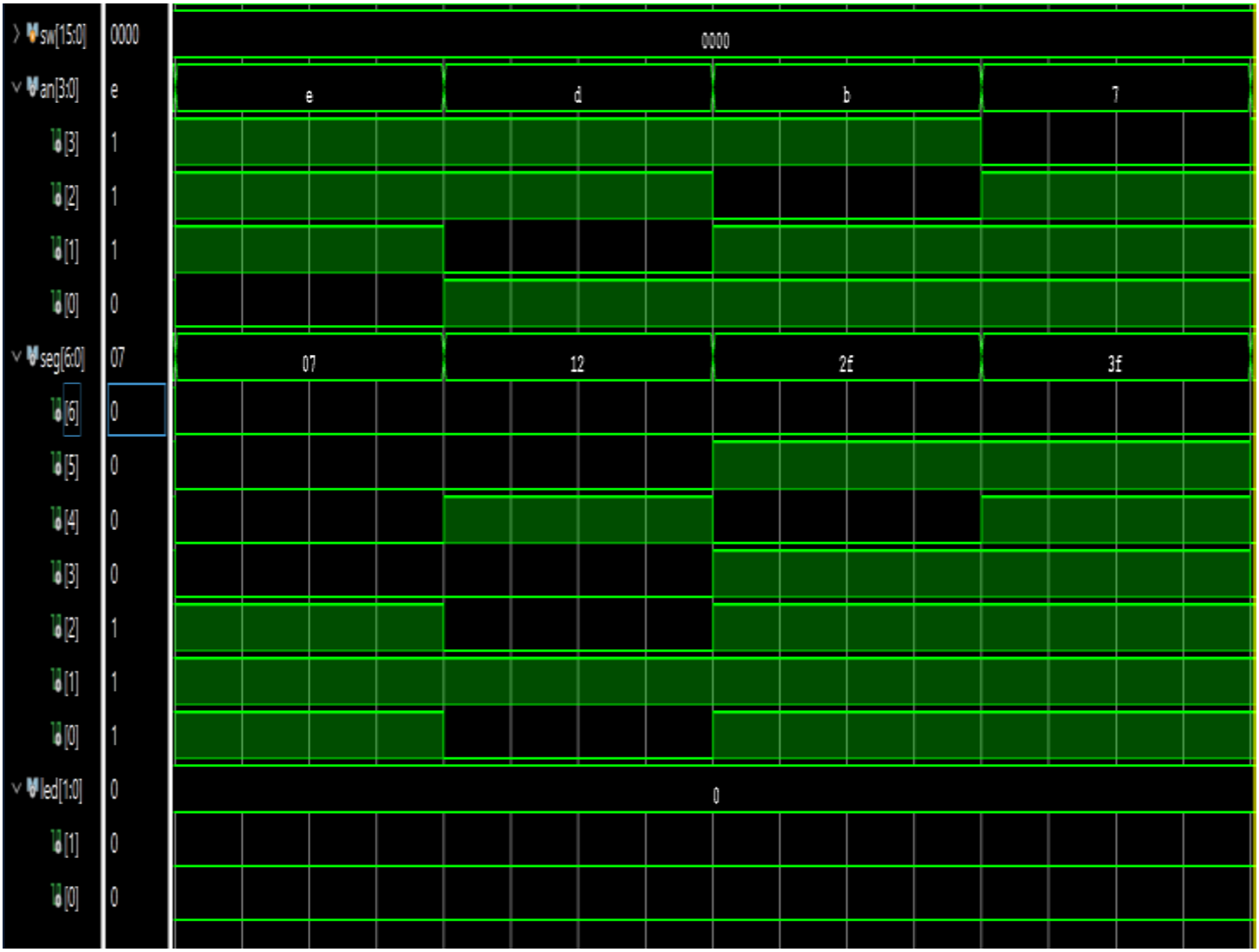
- **Traversal:**

1. If head equals NULL, underflow is indicated since the list is empty.
2. Otherwise, traversal begins from the head node and proceeds through each subsequent node using its next pointer.
3. The data of each node is displayed on the 7-segment display for one second, controlled by an internal counter.
4. Traversal stops once the current pointer becomes NULL.

• Simulation Snapshots:

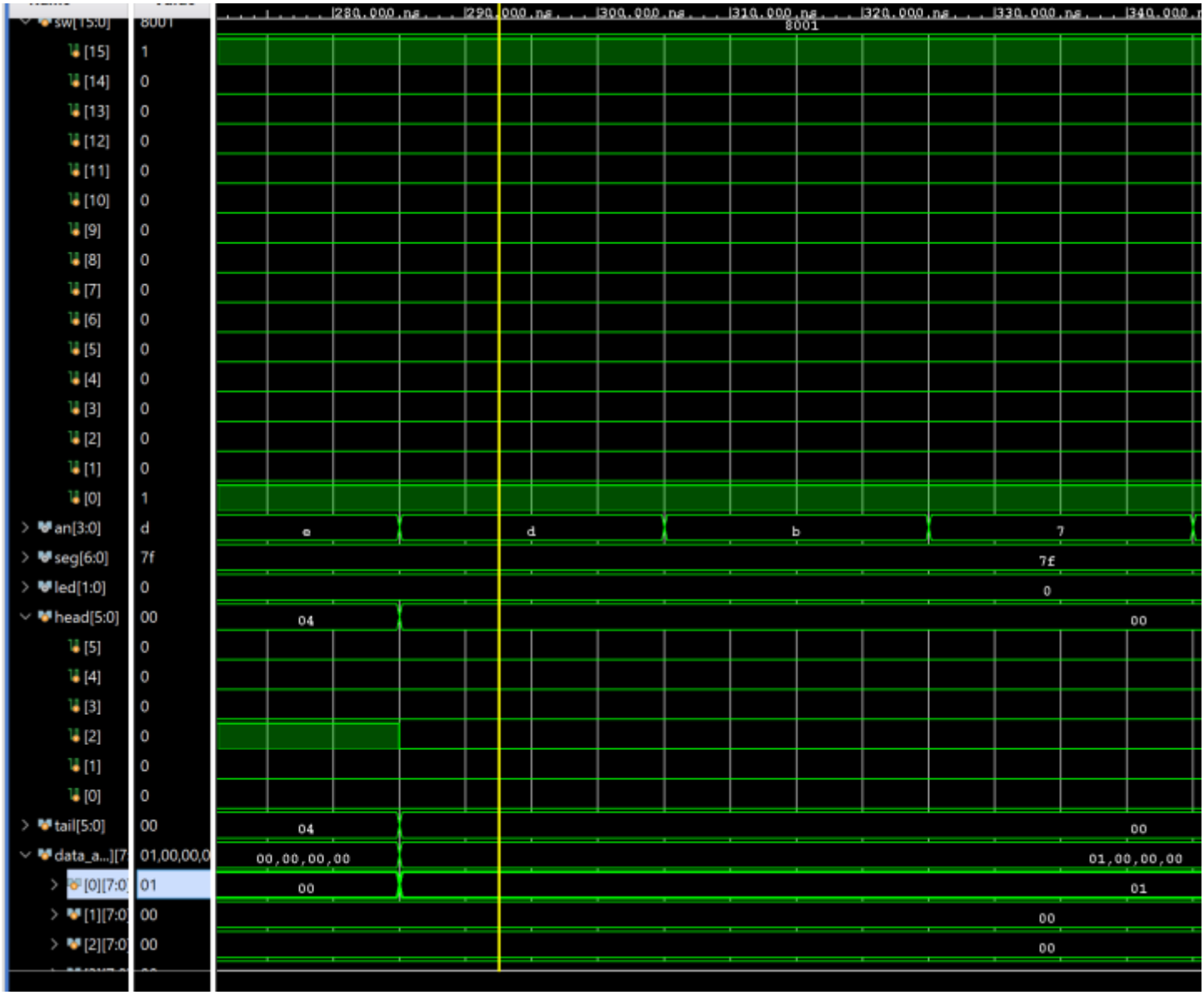
Reset Simulation:

The reset simulation verifies the system's response to the btnC input. As specified, when the reset button is pressed, the seven-segment display should show "-rSt".



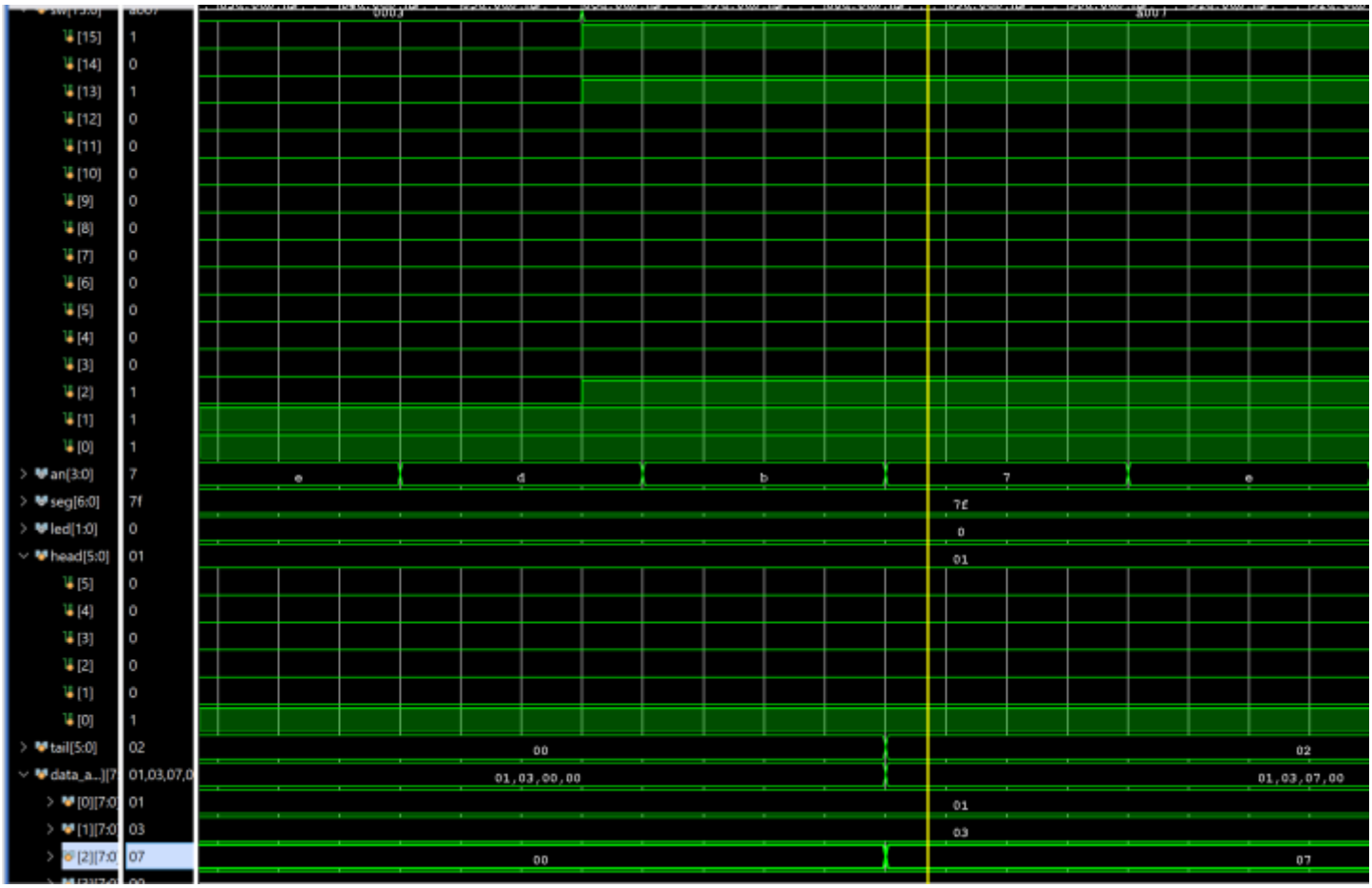
Insertion At Head Simulation:

This simulation shows the addition of 1 at head(i.e NULL(4) before insertion & 0 after insertion) and can be seen in the data_array:



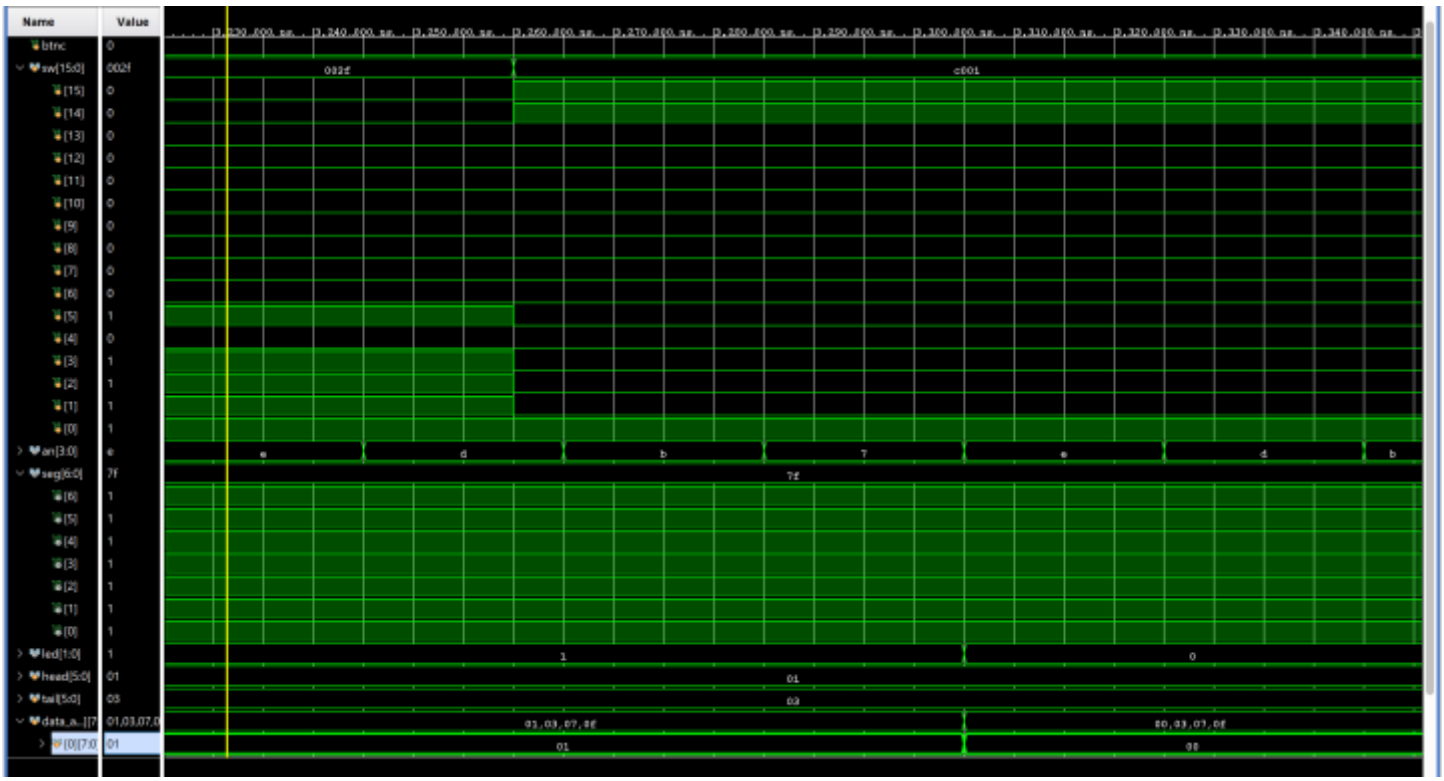
Insertion At Tail Simulation:

This simulation shows the addition of 7 at tail(i.e 0 before insertion & 2 after insertion) and can be seen in the data_array:



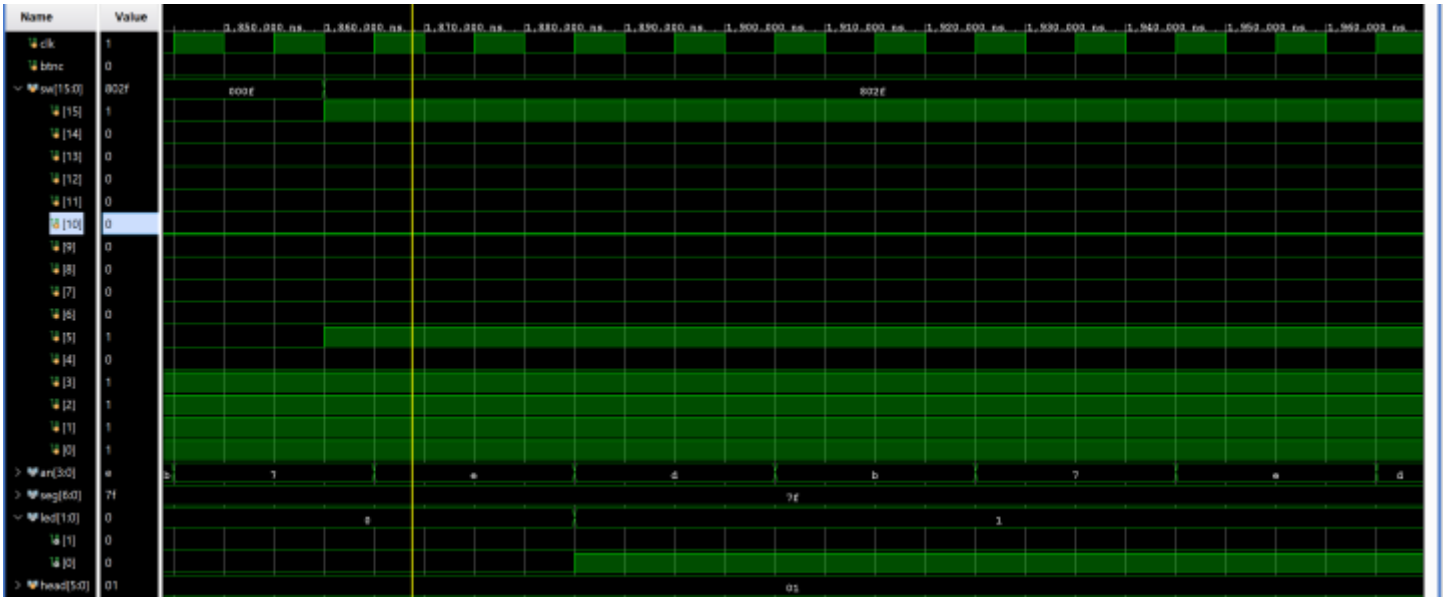
Deleting a number:

The simulation shows that we have 1 in the data array and after few clock cycles 101 operation is given and 1 is deleted:

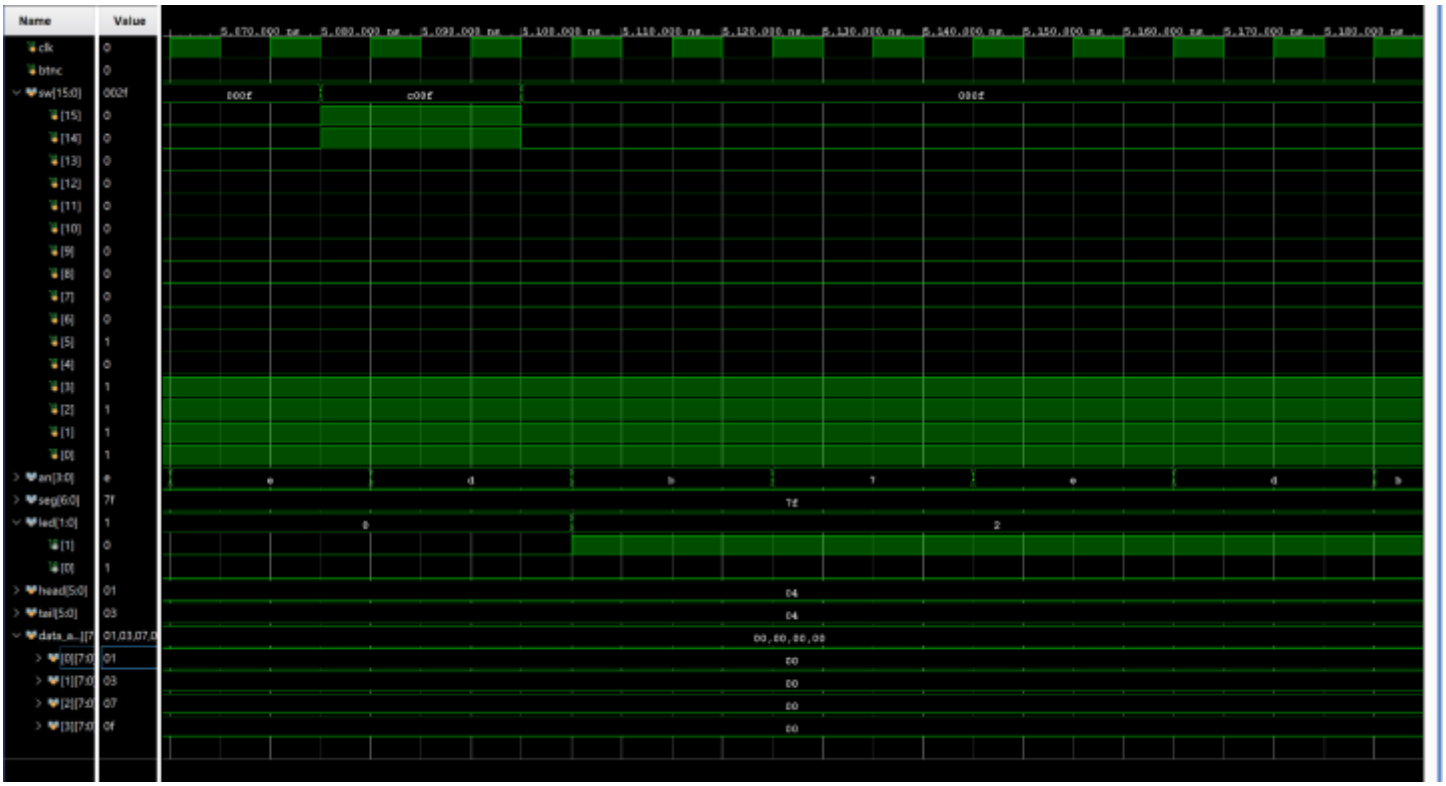


Inserting a number in a fully filled Linked List:

The simulation shows the overflow condition by led[0] is turned on:

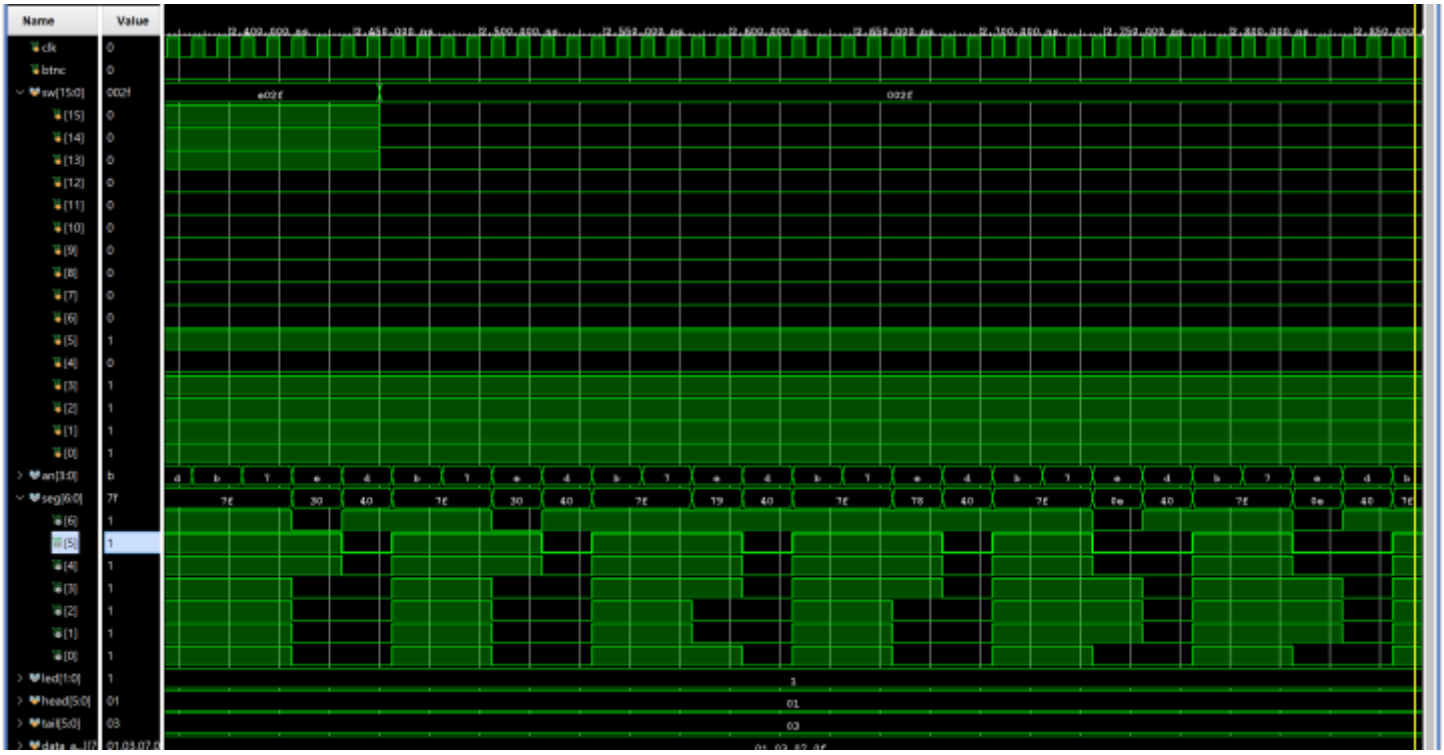


Deleting a number in an empty Linked List:

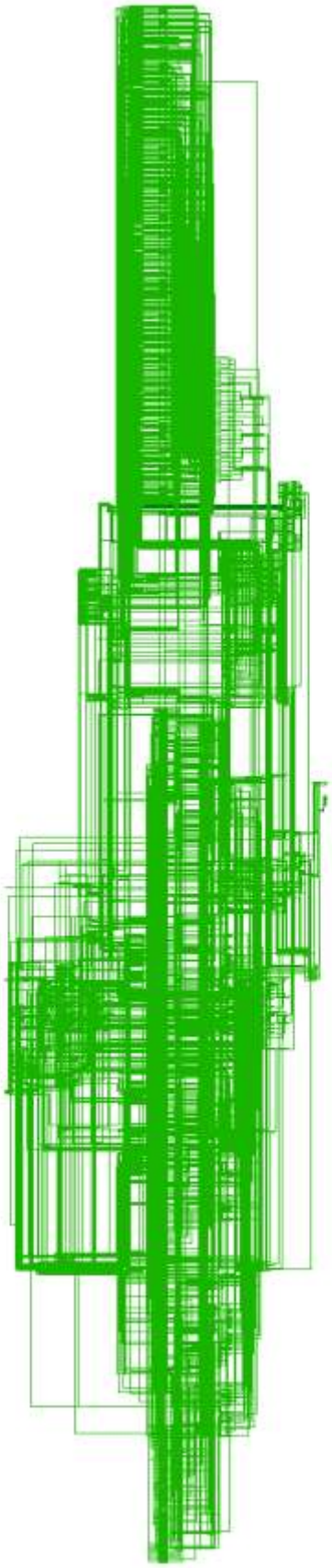


Traversal in a Linked List:

The simulation shows the display of values via seven segment display while traversing the linked list:



Generated Schematics:



Synthesis Report:

Resource	Used	Available	Utilization (%)
LUTs	946	20,800	4.55%
Flip-Flops (FFs)	636	41,600	1.53%
BRAMs	0	50	0.00%
DSPs	0	90	0.00%