Department of Computer Science & Engineering
Indian Institute of Technology Delhi
Semester I, 2025-26

COL215 - Digital Logic and System Design

Lab Assignment 7:

Linked List Implementation

Developed By:
Bharmal Bhakar (2024CS10403)
Ritik Raj (2024CS10086)

Introduction

The objective of this assignment was to implement a **singly linked list** data structure using **Verilog HDL**. Due to the lack of dynamic memory allocation in hardware description languages, the linked list structure was simulated using **register arrays** to store node data and pointers. A Finite State Machine (FSM) was designed to manage all list operations and control the system I/O, including switches, LEDs, and the 7-segment display.

1. Design Structure and Components

1.1 Node Structure and Memory Allocation

The linked list memory is implemented using parallel arrays parameterised by MAX NODES (default value is 32). The pointer width is derived using $clog2(MAX NODES).

- Node_data_mem: Stores the 8-bit data for each node.
- Next_ ptr_mem: Stores the index (pointer) to the next node.

A **Free List** mechanism is implemented using next_ptr _mem indices that are not currently part of the active list. The free_ptr register always points to the head of the available memory pool, and NULL _PTR is defined as MAX _NODES.

1.2 I/O Pin Functions

The module handles all control and data inputs as per the assignment specifications.

Table 1: Pin and Switch Function Assignments

Pin / Switch	Function
SW7-SW0	8-bit Data to be written/deleted
SW15-SW13	Operation Code (e.g., 100 for Insert Head
LED0	Overflow condition
LED1	Underflow condition
BTNC	Reset (Clears list and displays '-rSt')
Seg/an	Seg and anodes for controlling the Seven_segment display

Finite State Machine (FSM) Implementation

The core logic resides in the linked list fsm module, which uses a 10-state FSM to orchestrate the linked list operations, ensuring sequential and safe memory manipulation.

1.3 FSM States and Transitions

The FSM states are:

- S_IDLE (4'd0): Waiting for a new, stable operation input (SW op).
- S_INSERT_H (4'd1): Allocates a node from the free list and inserts it as the new head.
- S_INSERT_T_START (4'd2): Checks for an empty list (to route to S INSERT H) or proceeds to find the tail.
- S_INSERT_T_FIND (4'd3): Traverses the list until a node with a NULL PTR next pointer is found.
- S_INSERT_T_FINISH (4'd4): Links the old tail's next pointer to the new node.

- S_DELETE_START (4'd5): Initializes pointers for traversal or flags an underflow error.
- S_DELETE_FIND (4'd6): Traverses the list to find the first node matching SW data.
- S_DELETE _FINISH (4'd7): Updates list pointers (head or previous node) to bypass the deleted node, then returns the node to the free list.
- S_TRAVERSE_START (4'd8): Initializes the traverse_ptr or returns to IDLE if the list is empty.
- S_TRAVERSE_SHOW (4'd9): Holds the current node's data on the 7-segment display using a frequency divider (traverse display counter) before moving to the next node.
- S_ERROR_DISPLAY (4'd10): Displays the error LED (LED0 for Overflow, LED1 for Underflow/Not Found) for a fixed duration defined by TWO _SECONDS COUNT.

Design Decisions

Single-Cycle Operation Per Node

Insertion and deletion operations are designed to be completed within one or a few clock cycles (S INSERT H, S _INSERT _T FINISH, S DELETE FINISH). However, S _INSERT _T FIND and S DELETE FIND traverse the list one node per clock cycle. This trade-off balances complexity with throughput: while the maximum latency is proportional to $O(N)$ (list length), the design avoids complex multi-cycle controller logic for memory access.

Error Handling

The error condition (Overflow/Underflow) is handled by transitioning to S ERROR DISPLAY. This state is clocked by a large counter (error timer) to ensure the error LED is visible for a sufficient period, independent of the main operation flow.

2. Seven-Segment Driver

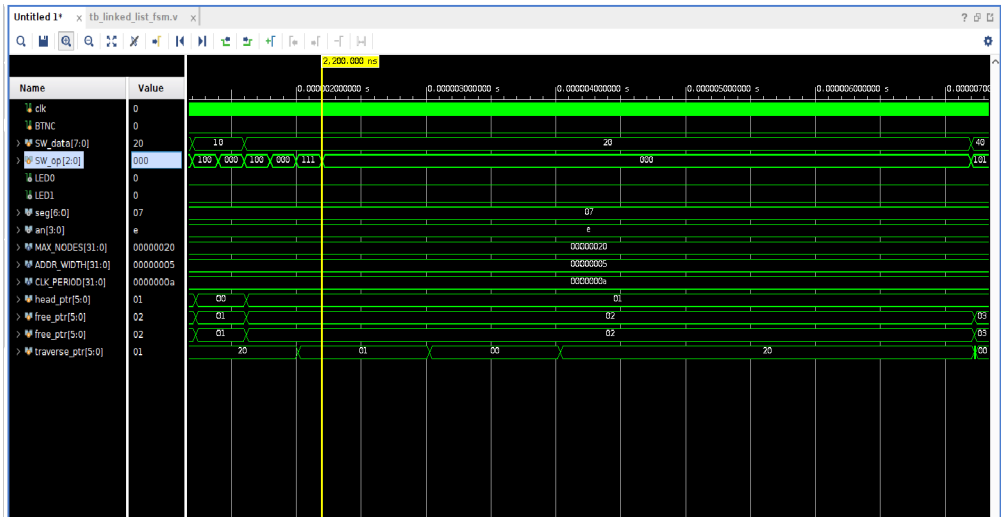
The seven segment driver module handles two primary display modes:

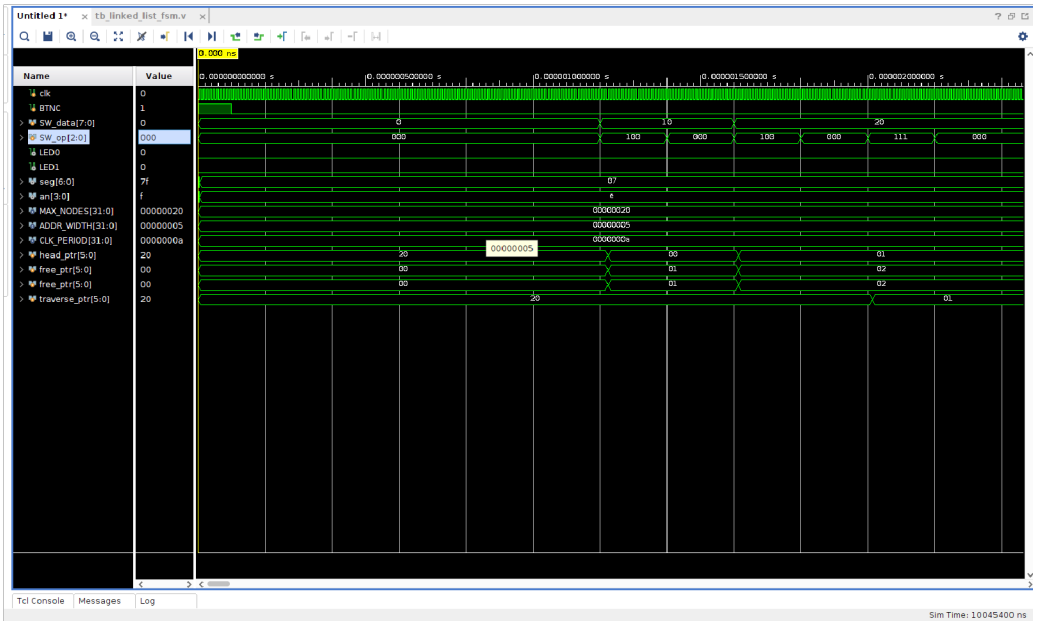
1. **Data Display**: Shows the 8-bit node data in hexadecimal format during traversal.
2. **Reset Display**: Upon activation of BTNC, the module enters a reset state and displays the string '-rSt' on the 7-segment display for the required 5-second duration, controlled by the reset _timer.

Simulation Results and Waveform Analysis

The TB linked list_fsm testbench verified all supported operations.The MAX DISPLAY COUNT parameter was overridden to 100 cycles for faster simulation visualization.

1.1 Insertion and Traversal





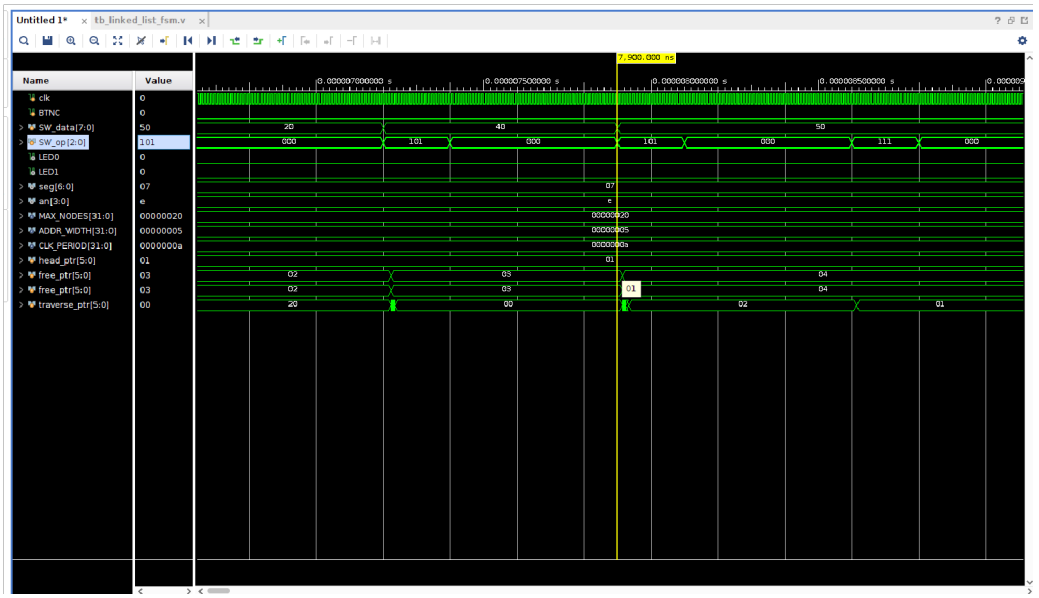


Figure 1: Simulation Waveform for Insert-Head (10, 20, 30)
Figure 2: Simulation Waveform for Initial Traverse (Expected: 30 → 20 → 10).
Figure 3: Simulation Waveform for Insert at Tail.

Snapshot Explanation: The waveform confirms that setting SW op = 3'b100 (Insert Head) correctly pushes the new data (SW _data) to the head of the list. During the Traverse operation (SW op = 3'b111), the ssd _data output cycles through the stored values in reverse insertion order, confirming the LIFO behavior of Insert Head and the correct list traversal.

3.2 Delete Operation and Pointer Update

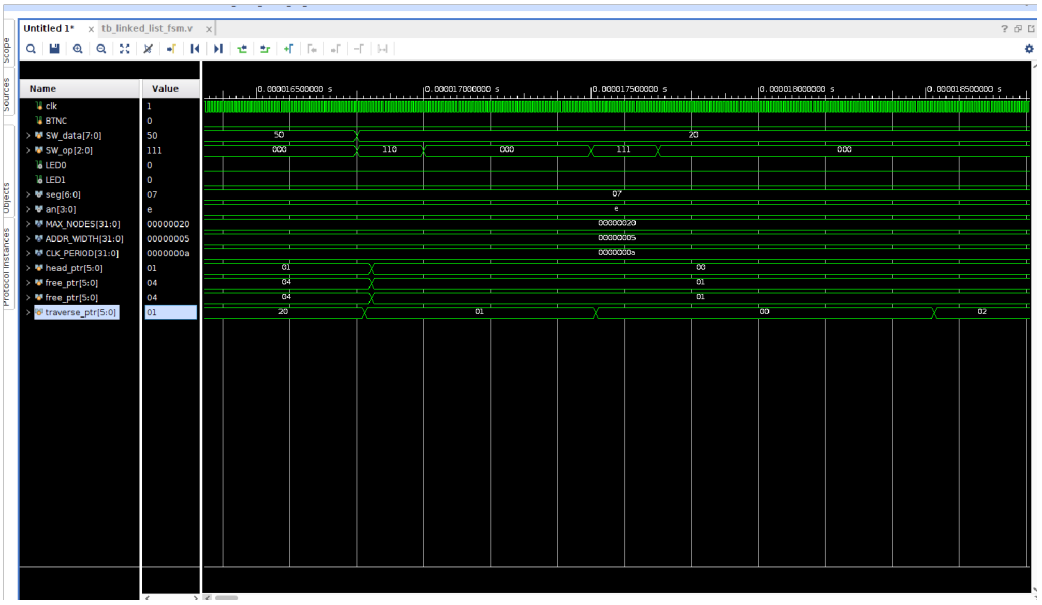


Figure 2: Simulation Waveform for Delete Operation (Data = 20) and subsequent list state.
Snapshot Explanation: The Delete operation (SW _op = 3'b110, SW data = 20) correctly transitions through S _DELETE _FIND (where traverse ptr and prev ptr are updated) and enters

S DELETE FINISH. In this state, the next ptr mem for the previous node is updated to bypass the deleted node, and the deleted node's index is prepended to the free list.

3.3 Error Condition (Underflow)

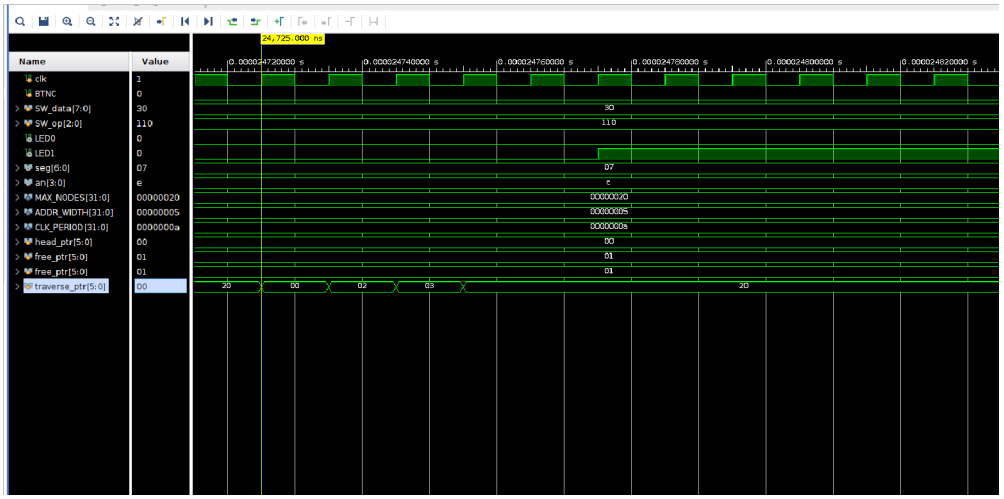
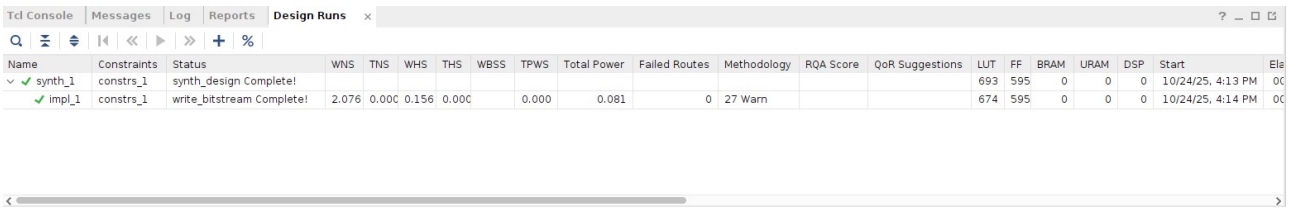


Figure 3: Simulation Waveform for Underflow Test (Delete from an empty list).

Snapshot Explanation: Attempting to delete a node when head ptr is NULL PTR transitions the FSM to S ERROR DISPLAY. The LED1 signal (Underflow) is immediately asserted and remains high until the error _timer expires, confirming the underflow handling requirement.

Hardware Resource Utilization

The following table summarizes the resource usage after synthesis for a typical target device (e.g., Xilinx Artix-7, compatible with the Basys 3 board). This data was obtained from the synthesis report.

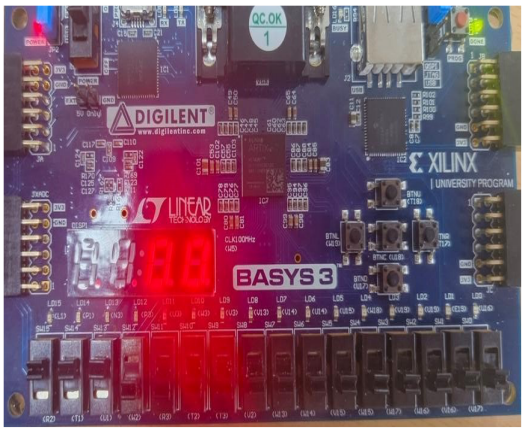


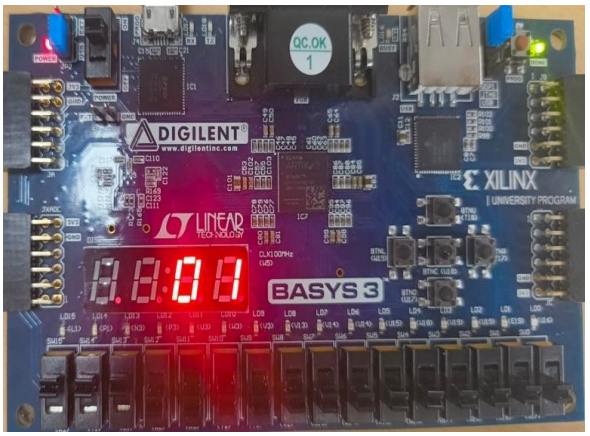
Name	Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power	Failed Routes	Methodology	RQA Score	QoR Suggestions	LUT	FF	BRAM	URAM	DSP	Start	Ela
synth_1	constrs_1	synth_design Complete!												693	595	0	0	0	10/24/25, 4:13 PM	0(
impl_1	constrs_1	write_bitstream Complete!	2.076	0.000	0.156	0.000		0.000	0.081	0	27 Warn			674	595	0	0	0	10/24/25, 4:14 PM	0(

Note: The linked list memory (node data mem, next ptr mem) is implemented using Distributed RAM (LUTs) rather than Block RAM (BRAM), ensuring portability and simpler memory management in this design.

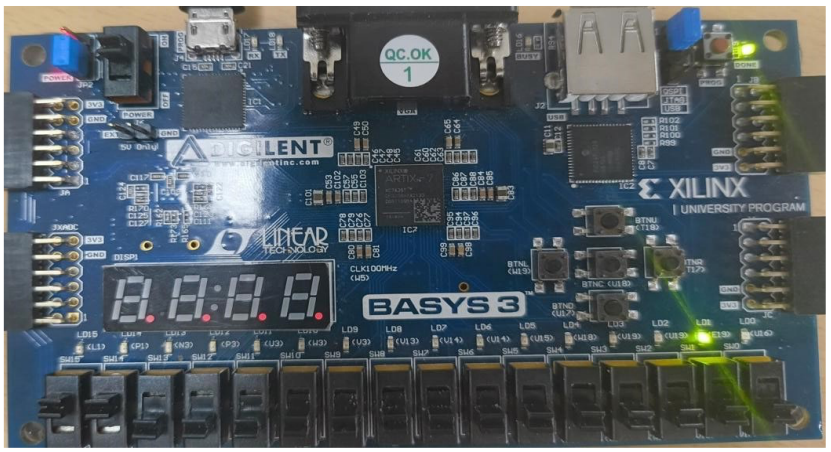
Generated Schematics and Hardware implementation

5.1 Hardware Implementation

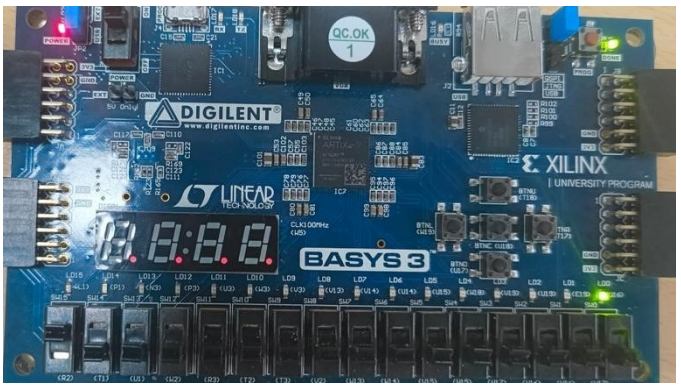




Linked list traversal is being shown when sw15-sw13 are 111(all on)



underflow condition is shown in the photo when the given number is not present in linked list.



overflow condition is shown when linked list is full and we try to insert another number.

Reset shown for 5 secs

Generated Schematics

A high-level view of the entire design, showing the connections between the FSM and the driver. (Pdf format directly attached at the last page.)

Conclusion

The singly linked list was successfully implemented in Verilog HDL using a FSM controlling register arrays for data and pointers. All required operations—**Insert at Head**, **Insert at Tail** , **Delete**, and **Traverse**—were verified through simulation, along with correct handling of **Overflow** (LED0) and **Underflow** (LED1) conditions.

(sechematic is attached at the last page)