Elevator Controller

1. Introduction

- Elevator controllers are essential digital systems designed to manage the safe and efficient movement of elevators within buildings. They handle user requests, control the elevator cabin's motion, and ensure reliable operation under various conditions.

- This project focuses on designing, implementing, and verifying a N-floor (Parameterized) elevator controller. The design emphasizes request prioritization, stop/resume functionality, and timing accuracy. Simulation and verification are used to confirm correct behavior under multiple operating scenarios.

- The elevator should manage multiple requests simultaneously by assigning priority. The elevator should prioritize requests in its current direction of movement, servicing requests in the upward direction while moving up, and downward while moving down. If the elevator is idle (not moving), priority is given to the first floor requested. If multiple requests are received at the same time while the elevator is idle, priority is given to the highest floor among the requests.

- The elevator includes a **stop feature**: a push button inside the elevator allows passengers to stop it immediately. When pressed, the elevator halts and retains all input, output, and control states. Pressing the button again releases the stop, allowing the elevator to resume from where it left off, with all signals and states preserved for a seamless continuation.

2. System-level Architecture

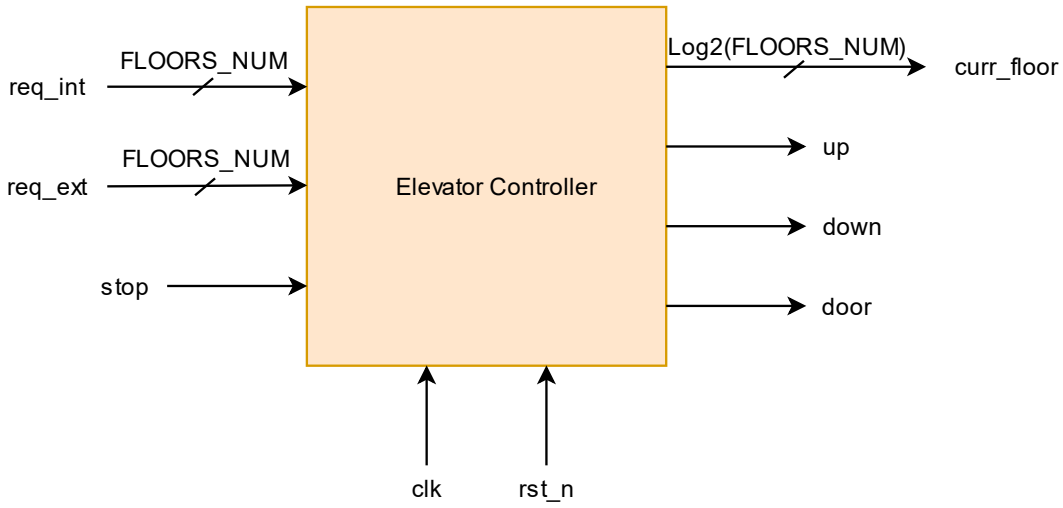

Figure 1 : Elevator Controller System Block

2.1 Parameters Declaration

Parameter	Default value	Description
FLOORS_NUM	5	Number of Floors in the residential building
TRAVEL_FLOOR_CYCLES	100,000,000	Number of cycles the elevator takes to travel between two consecutive floors
DOOR_OPEN_CYCLES	100,000,000	Number of cycles the door still opened when reaching the target floor

2.2 Inputs & Outputs Declaration

Signal	Direction	Width	Description
clk	Input	1	Positive edge system clock
rst_n	Input	1	Negative edge asynchronous system reset
req_int	Input	FLOORS_NUM	Internal elevator requests where each bit represents a push button inside the elevator for a specific floor.
req_ext	Input	FLOORS_NUM	External requests where each bit represents the push button outside the elevator for its corresponding floor.
stop	Input	1	A stop button inside the elevator allows People to immediately halt the elevator.
curr_floor	Output	Log2(FLOORS_NUM)	An output indicating the current floor number. For example, if the elevator is on the third floor, ***curr_floor*** will be set to 'd3.
up	Output	1	1: elevator is moving up 0: elevator is moving down or not moving
down	Output	1	1: elevator is moving down 0: elevator is moving up or not moving
door	Output	1	1: elevator door opens 0: elevator door closed

3. Block-level Architecture

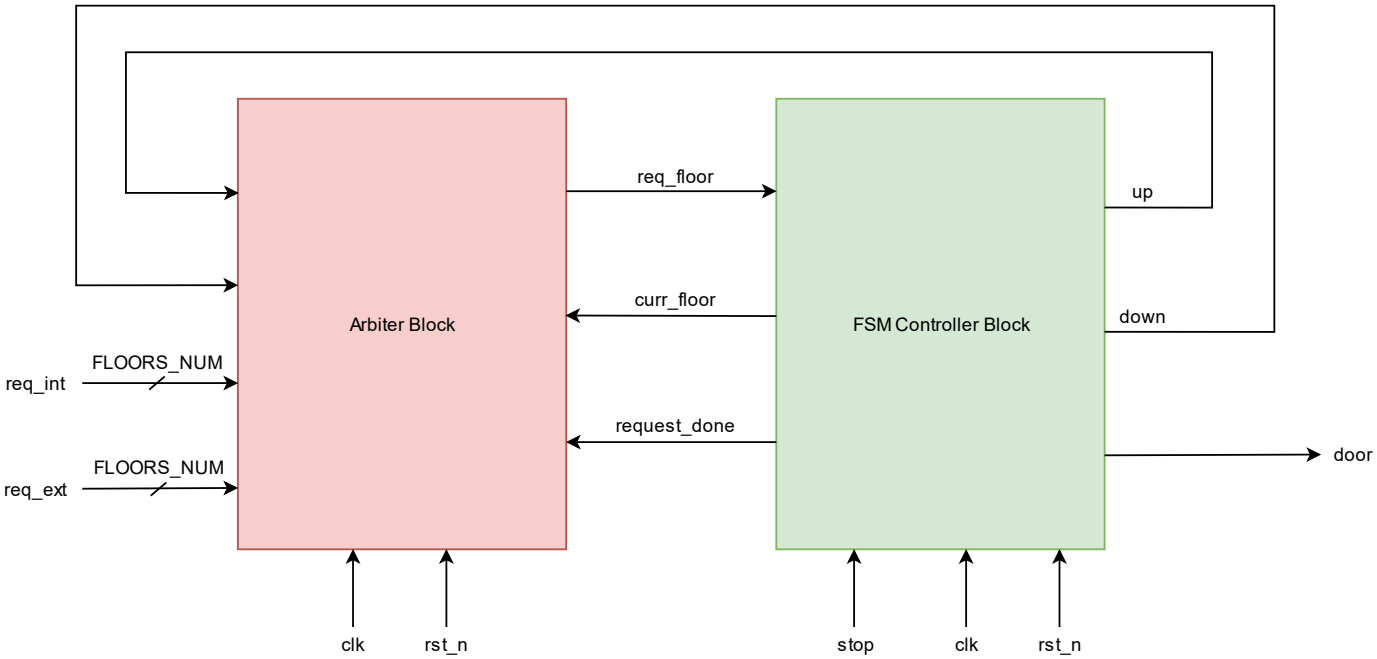

Figure 2 : Top Wrapper that instantiate the two modules (Arbiter, FSM)

3.1 Arbiter Block

- The module acts as the intelligent decision-making core of the system. Its primary role is to analyze all incoming service requests, both from inside the elevator cabin (***req_int***) and from the hallways on each floor (***req_ext***), and determine the most optimal next floor for the elevator to serve (***req_floor***).

- It intelligently prioritizes requests based on the elevator's current direction of travel (***up/down***). When moving up, it selects the nearest requested floor above the current position; when moving down, it selects the nearest floor below. If the elevator is idle, it defaults to serving the highest pending request.

- The module also maintains a register of (***req_pending***) to ensure no requests are lost before they are serviced, clearing them only when a (***request_done***) signal is received.

3.2 FSM Controller Block

- The module is the state machine that physically controls the elevator's operation based on the target floor (***req_floor***) provided by the arbiter. It manages the entire journey through five distinct states: **IDLE**, **UP**, **DOWN**, **DOOR**, and **STOP**.

- Moreover, it is responsible for incrementing or decrementing the (***curr_floor***) output, asserting the up and down direction signals, and controlling the door operation and generates the (***request_done***) signal to inform the arbiter that a request has been fulfilled.

- A crucial feature is its ability to handle an emergency stop signal, which freezes operation in a **STOP** state and seamlessly resumes from the saved state once the stop condition is cleared.

4. Simulation Results

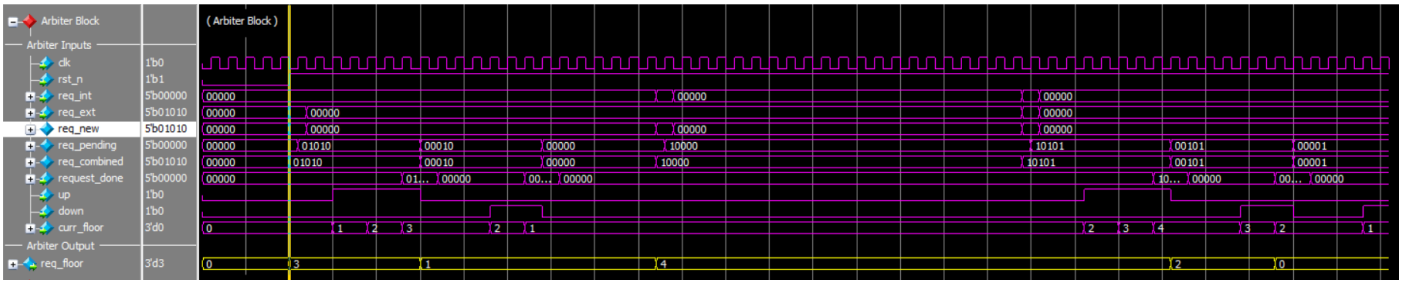

Figure 3 : Arbiter Block Signals

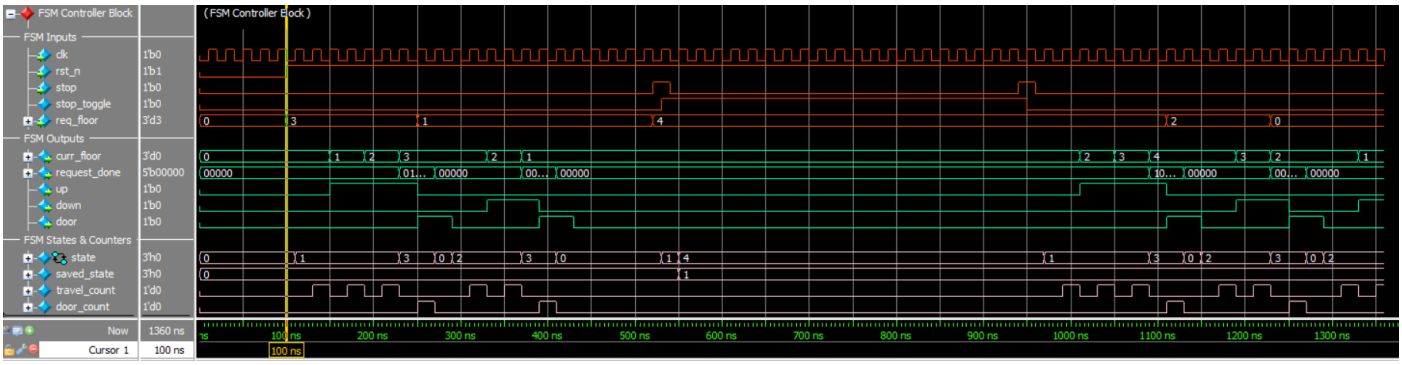

Figure 4 : FSM Controller Block Signals