RTL Design, Validation, and Implementation of Elevator Controller

Project Documentation

Summer 2022

Prepared by:
Shehab Bahaa

Table of Contents

Elevator Controller

1. Objectives

The objectives of this project are:

- RTL Design of an elevator controller using VHDL.
- Validation of the design using self-checking testbenches and simulation.
- The hardware implementation and validation of the elevator controller using the Terasic DE0-CV development board.

2. Design Requirements

Move the elevator either up or down to reach the requested floor. Once on the requested floor, open the door for at least 2 seconds. Ensure the door is never open while moving. Don't change directions unless there are no higher requests when moving up or no lower requests when moving down. Assume that the elevator moves from one floor to another in 2 seconds. The controller should use the 50 MHz clock on the DE0-CV board. Use a 1-sec clock enable to the timer for the elevator movement and the opening of the door. The inputs and outputs of the controller are shown in the figure below. Also, the controller can be broken into a Request Resolver that resolves various floor requests into a single requested floor and a Unit Control that moves the elevator to this requested floor as shown in figure 1. Design the controller to serve up to 10 floors (0 is the ground floor and 9 is the highest floor). Use VHDL generics for the number of floors. The floor output should be connected to a seven-segment display. All control signals and status signals will be connected to push button switches and LEDs.

3. HW Validation Setup

The hardware validation setup of the elevator_ctrl.vhd module is shown in Figure 2. The setup is limited to 4 floors only. The buttons are limited to the ones inside the elevator only the other buttons are hardwired to '0' to be inactive during the test. The 4 control buttons are assumed to be KEY0, KEY1, KEY2, and KEY3. Those key correspond to bn0, bn1, bn2, and bn3. The reset_n is active low and is tied to push button KEY4. The floor count is connected to the units SSD. The status output signal mv_up, mv_down, and door_open are connected to LED0, LED1, and LED3; respectively.

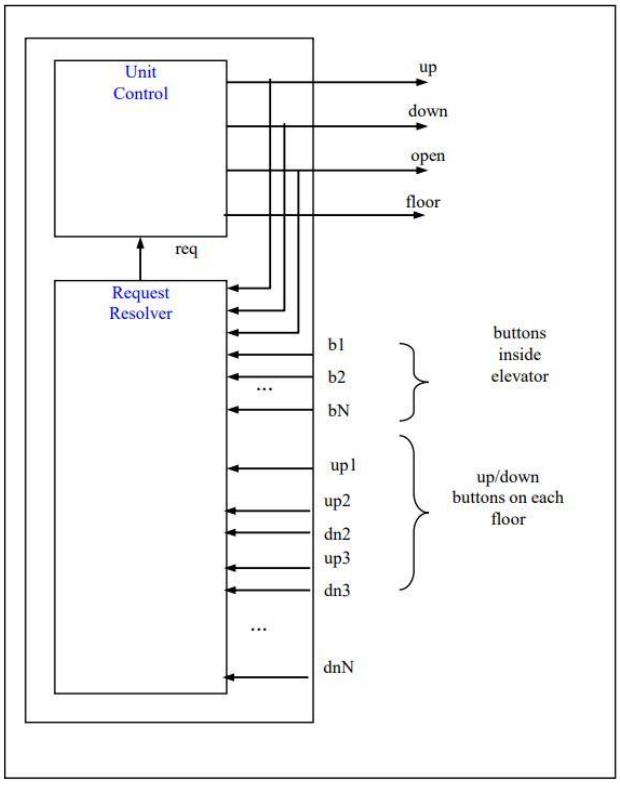


Figure 1: Elevator controller block diagram and interface ports

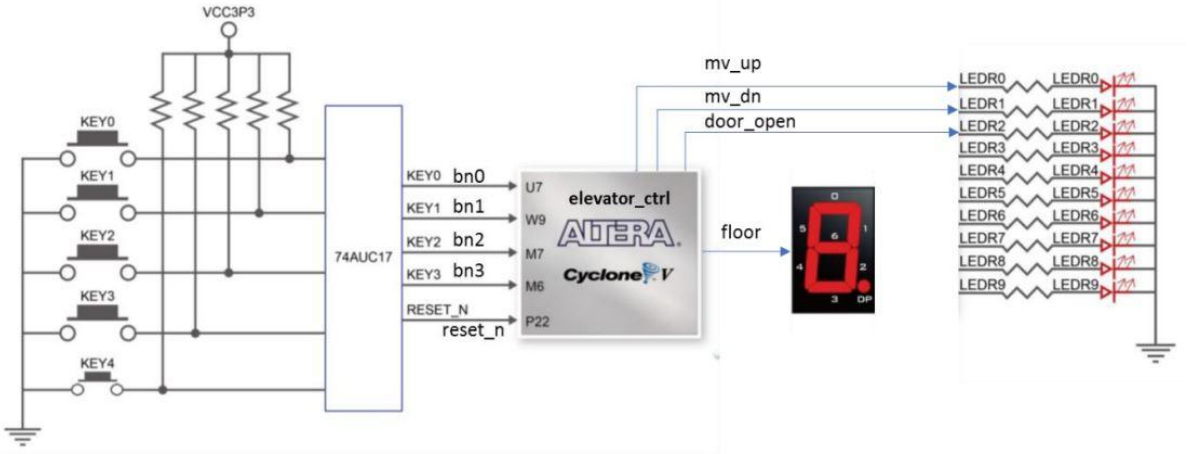


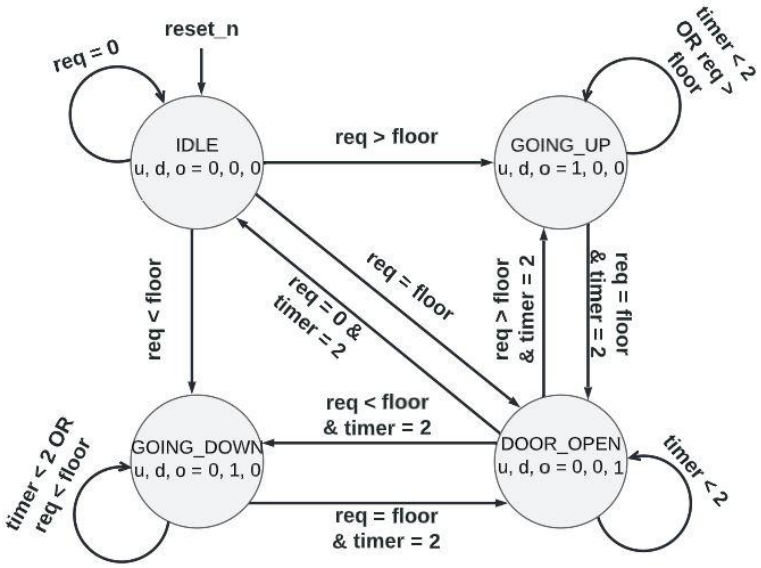
Figure 2: Validation setup for the elevator controller

4. Algorithm for Floors' Request Resolution

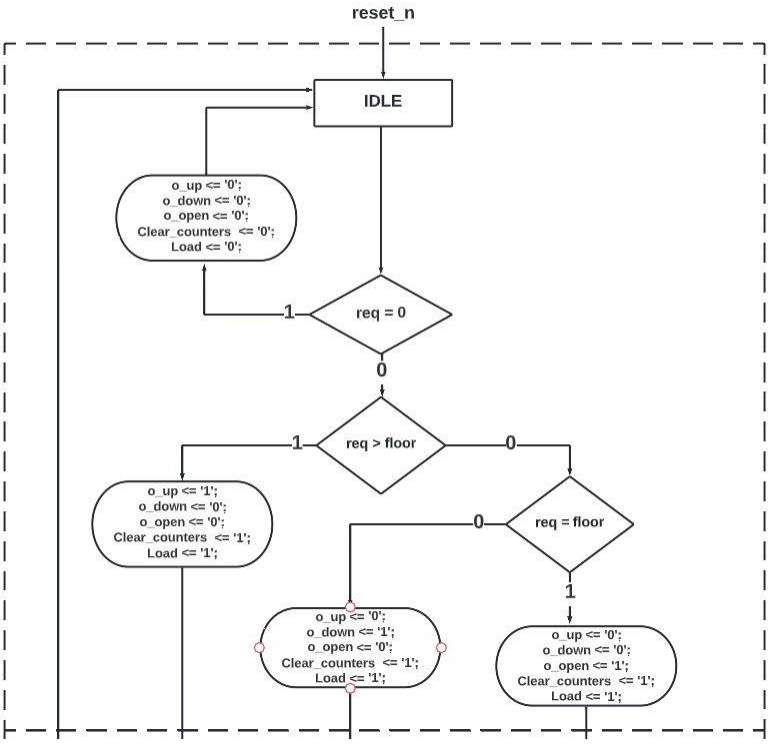
1) **Requests analysis**: the registered requests from both the inside and outside buttons are analysed to determine if there are requests from the same floor of the elevator, floors above the elevator, or/and floors down the elevator, then the results of the analysis is stored in flags (same_floor_request_flag, up_requests_flag, down_requests_flag).
   Implementation: (signal of floor number is one-hot encoded)
   a) For the same floor requests, floor signal is anded with requests.
   b) For the up direction, floor signal is shifted left by one then subtracted by 1 then negated. Afterwards, the result is anded with the requests.
   c) For the down direction, floor signal is subtracted by 1. Afterwards, the result is anded with the requests.
2) **Direction of movement decision**: these flags are then used to decide the direction of movement which can be:
   a) Opening the door (highest priority) if there is a request from the same floor of the elevator while the elevator is not moving.
   b) Going up if there are up requests and the elevator is moving up or there are up requests and the elevator was moving down, but there is no down requests anymore.
   c) Going down if there are down requests and the elevator is moving down or there are down requests and the elevator was moving up, but there is no up requests anymore.
   d) Staying idle if there is no requests at any floor.
3) **Next request resolution**: if the elevator is moving up, the first set bit is detected for the result of the requests analysis (by anding the two's complement with the original signal). if the elevator is moving down, the result of the requests analysis is reversed then the first set bit is detected (by anding the two's complement with the original signal) and then the result is reversed again.

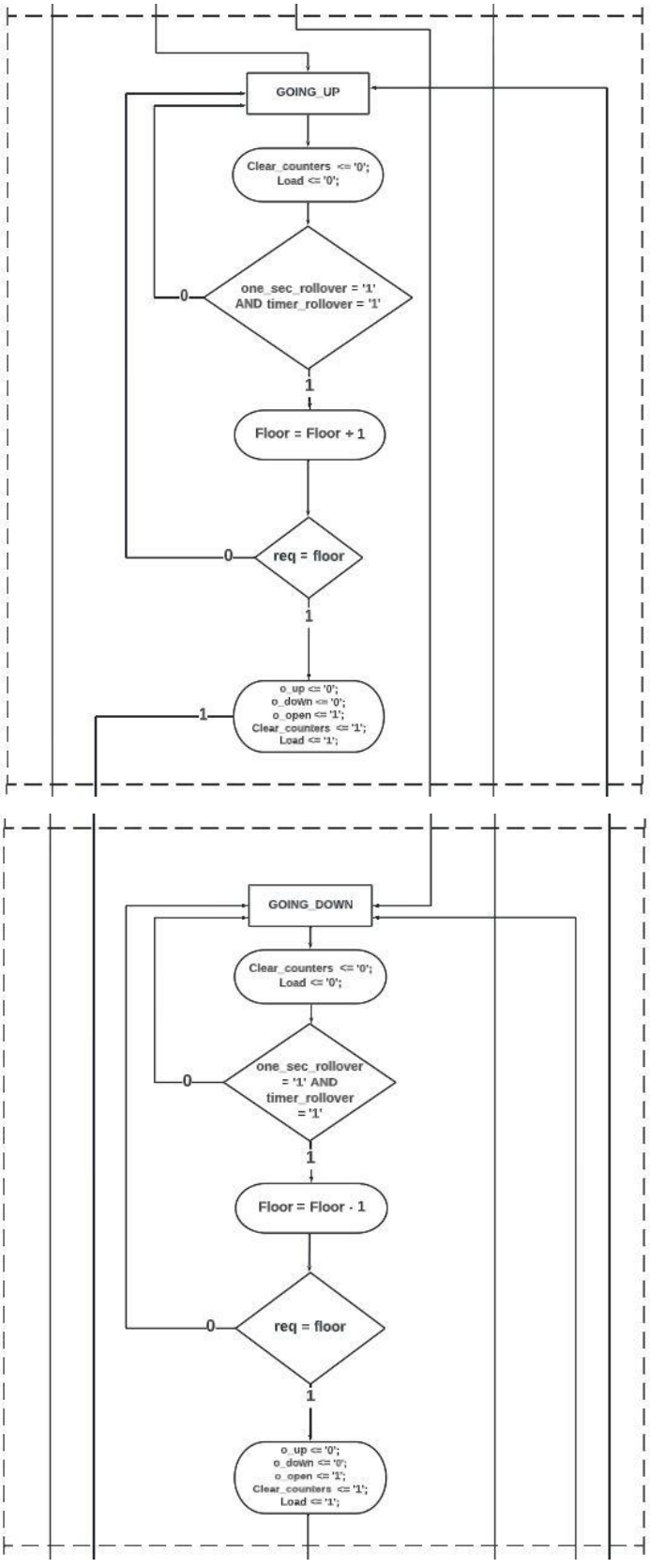
5. Unit Control's FSM

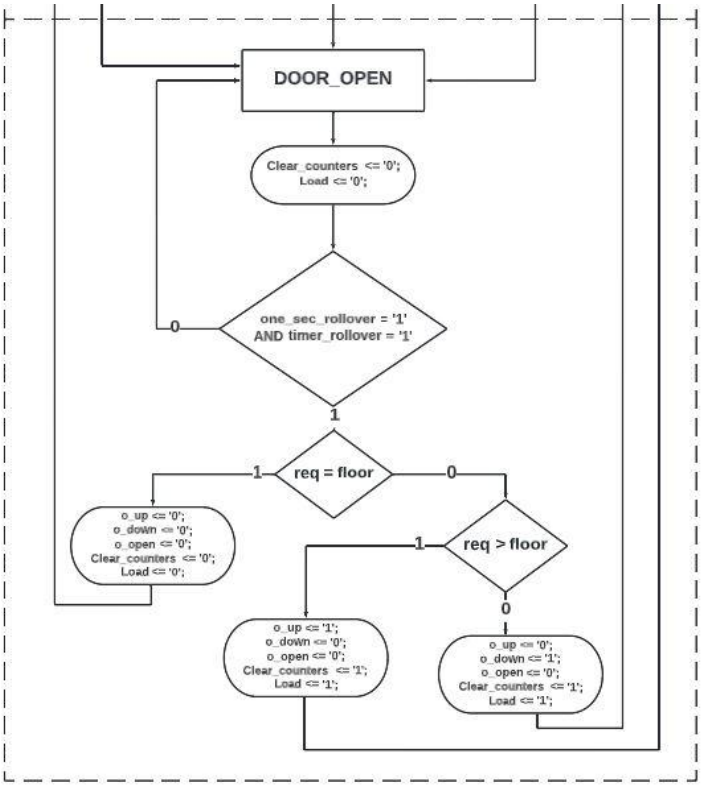
- **FSM State Diagram**



- **ASM Chart**




GOING_UP
Clear_counters <= '0';
Load <= '0';
one_sec_rollover = '1'
AND timer_rollover = '1'
0
1
Floor = Floor + 1
req = floor
0
1
o_up <= '0';
o_down <= '0';
o_open <= '1';
Clear_counters <= '1';
Load <= '1';
1
GOING_DOWN
Clear_counters <= '0';
Load <= '0';
one_sec_rollover
= '1' AND
timer_rollover
= '1'
0
1
Floor = Floor - 1
req = floor
0
1
o_up <= '0';
o_down <= '0';
o_open <= '1';
Clear_counters <= '1';
Load <= '1';


DOOR_OPEN
Clear_counters <= '0';
Load <= '0';
one_sec_rollover = '1'
AND timer_rollover = '1'
0
1
req = floor
1
0
o_up <= '0';
o_down <= '0';
o_open <= '0';
Clear_counters <= '0';
Load <= '0';
req > floor
1
0
o_up <= '1';
o_down <= '0';
o_open <= '0';
Clear_counters <= '1';
Load <= '1';
o_up <= '0';
o_down <= '1';
o_open <= '0';
Clear_counters <= '1';
Load <= '1';