Lab 8 : Elevator Control System

Objective:

Design an elevator control system for 2 elevators which moves across 4 floors.

Specifications:

* The number of floors = 4. For each floor, a pair of slide switches represent UP and DOWN request buttons on that floor. Total no. of slide switches used for this = 8. An equal number of LEDs are used to indicate registration of these requests. Total no. of LEDs used for this = 8.

* For each lift, 4 slide switches represent floor request buttons inside that lift. Total no. of slide switches used for this = 8. An equal number of LEDs are used to indicate registration of these requests. Total no. of LEDs used for this = 8.

* Slide switches are used in a way that they are turned on, then immediately turned off (by the user).

* For each lift, one 7-seg display shows the floor on which that lift is and another 7-seg display shows the status of the lift - moving up (u) / moving down(d) / halted with door closed(c) / halted with door open(o). Use the small letters to display the status of the lift on the Seven Segment Display (SSD). Total no. of 7-seg displays used = 4. You can keep time delays for elevator going up(between adjacent floors)/down(between adjacent floors)/door opening and door closing as 2s,2s, 0.5s and 0.5s respectively. Once the lift reaches the destination floor, the doors are opened. If the lift has to serve more requests the door closes after 1 second (can be overridden by door_closed, door_open) else, it remains open.

* The lift number displayed on the Seven Segment Display (SSD) should change after the lift has reached the certain floor, but before the door is opened.

* For each lift, two push buttons are used for door open/close requests. Total no. of push buttons used for this = 4. The door takes 0.5s to close after the door_close button is pressed and 0.1s to open if door_open is pressed. If the lift has to serve more requests the door closes after 1 second (can be overridden by door_closed, door_open) else, it remains open.

* One push button may be used as a reset button to facilitate testing by bringing the system to a known state (Both lifts at floor 0, all requests reset, doors open).

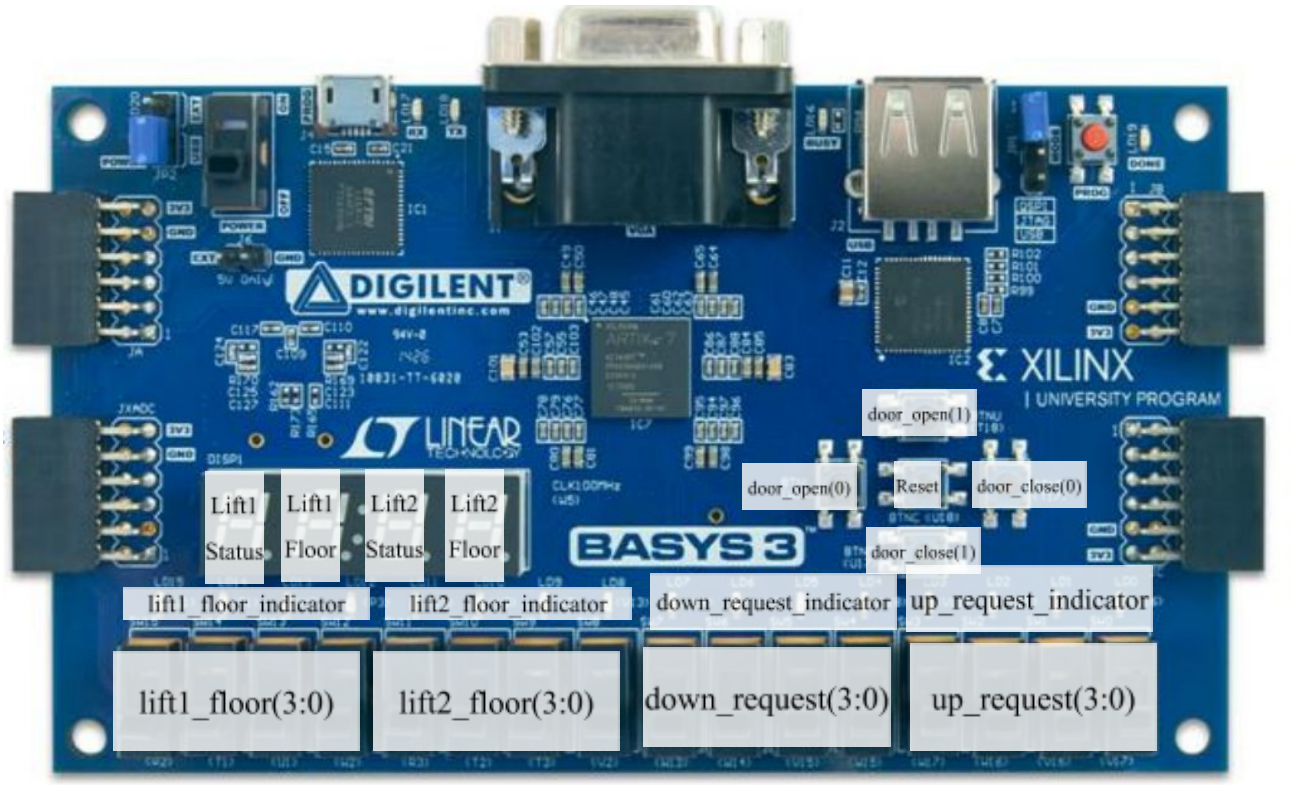
* Direction must be maintained (A lift going up goes up first and a lift going down goes down first before reversing direction). If a user requests a floor from inside the lift in the opposite direction of movement, the request is registered but served only after completing requests in the original direction. The strategy is described in the end.

Details of the design

Name	Type	Description
up_request(3:0)	Input	Requesting the lift to go up from i^{th} floor. Note that up_request(3) should not work.
down_request(3:0)	Input	Requesting the lift to go down from i^{th} floor. Note that down_request(0) should not work.
up_request_indicator(3:0)	Output	LED Indicator for up_requests
down_request_indicator(3:0)	Output	LED Indicator for down_requests
reset	Input	Reset system as per the specifications
cathode(6:0)	Output	SSD Cathode output
anode(3:0)	Output	SSD Anode output
door_open(1:0)	Input	Request to keep door open for i^{th} lift
door_closed(1:0)	Input	Request to keep door closed for i^{th} lift
clk	Input	Clock to the system

lift1_floor(3:0)	Input	The floor buttons corresponding to lift1. If a user enters the lift1 and presses lift1_floor(2), the lift should go to the 2nd floor. Also, the current floor should not be registered if pressed. If the lift is going in the opposite direction, the request is registered but the lift completes requests in the original direction before reversing.
lift2_floor(3:0)	Input	The floor buttons corresponding to lift2.
lift1_floor_indicator(3:0)	Output	Registration of lift1_floor button press
lift2_floor_indicator(3:0)	Output	Registration of lift2_floor button press
sim_mode	Input	High for simulation. Details will be provided later.

Entity Name: lab8_elevator_control



Elevator Request Strategy:

Floors are 0-3 and lifts are Lift1,Lift2. Refer to the block diagram below.

request_handler receives the various requests from all floors and assigns the request to one of the lifts. The Lift Controllers are responsible for executing these requests.

The lifts have to be in one of these three states:

1. reqUp: Executing a request where lift has to go up.
2. reqDown: Executing a request where lift has to go down.
3. Idle: Not executing any requests at the moment.

The *request_handler* receives incoming requests from floor, eg. 2D (Go down from 2nd floor). Then it classifies this request with respect to each lift in one of the following categories:

1. UpReqUp : A request from above the lift for going up.
2. UpReqDown: A request from above the lift for going down.
3. DownReqUp: A request from below the lift for going up.
4. DownReqDown: A request from below the lift for going down.

Also, requests from the same floor are classified as requests coming from a lower floor (either as DownReqUp or DownReqDown). The requests in the category UpReqUp and DownReqDown are assigned immediately to the lift with state reqUp and reqDown respectively. In case of UpReqDown/DownReqUp requests, the *request_handler* waits until one of the lifts is in the Idle state and assigns it to the lift which is idle first. Note that the classification might have changed from the time it was received. The classification will have to be computed again, based on the current floor position. In all cases, if both lifts are eligible for assignment, preference is given to lift1. The idle state lift services the requests in the following order (higher to lower priority) - (1) UpReqUp,(2) UpReqDown,(3) DownReqUp,(4) DownReqDown.

When we say assign a request, we mean that the request_handler communicates the destination floor number to the lift controller. The lift controller modifies its existing targets according to this information.

The lift controller also receives requests from people entering the lift. The targets will therefore be a union of requests from people inside the lift and the request_handler.

[Implementation hint]

When the request_handler has to wait, you can use a bit vector to store information about which requests have come (out of all possible requests). Then if any of the lifts change its state to idle, one of these requests can be assigned.

BLOCK DIAGRAM