## ECGR 3181 Final Project, Spring 2016

## Due Date : April 26

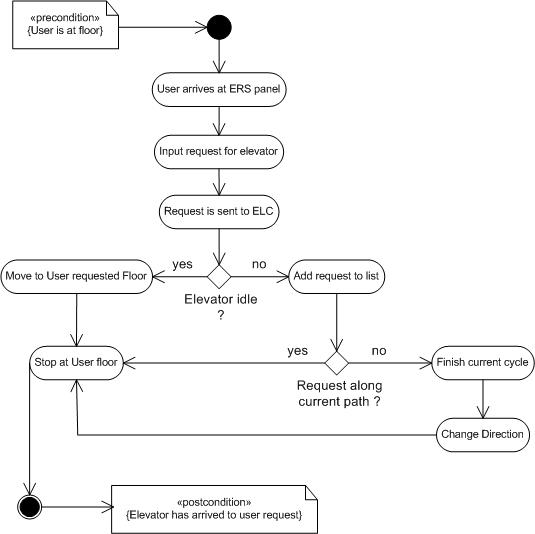
Design an Elevator Controller for a building with 4 floors. Deduce the required inputs and outputs from the following description of the system.

Use Cases with Activity Diagrams

**Use Case 1**: Request Elevator Use Case.

* **Primary Actor(s)**: Elevator User, Elevator Logic Controller   
  **Description**: The Elevator User uses the Elevator Request System (ERS) to request an elevator.  
  **Preconditions**: An Elevator User is at a floor and desires to use the elevator.   
  **Flow of Events**: The Elevator User arrives at the ERS panel on their floor :
  1. User presses “Up” or “Down” button on ERS panel.
  2. The ERS sends a signal to the ELC, detailing which floor the user is on.
  3. The request is added to a list of floors to visit.
  4. The ELC determines which direction to move to service the next request.
  5. The doors are closed, if open, and the elevator begins to move towards the requested floor.
  6. The ELC checks whether to stop as the elevator approaches a floor.
  7. The elevator stops at requested floors that are along the route to the original destination floor.
  8. The ELC opens the elevator door when it comes to a stop.
  9. The elevator arrives to service the request of the user.

**Postconditions**: The elevator has arrived in response to a request.   
**Alternative Flow of Events**: None.   
**Assumptions**: None.



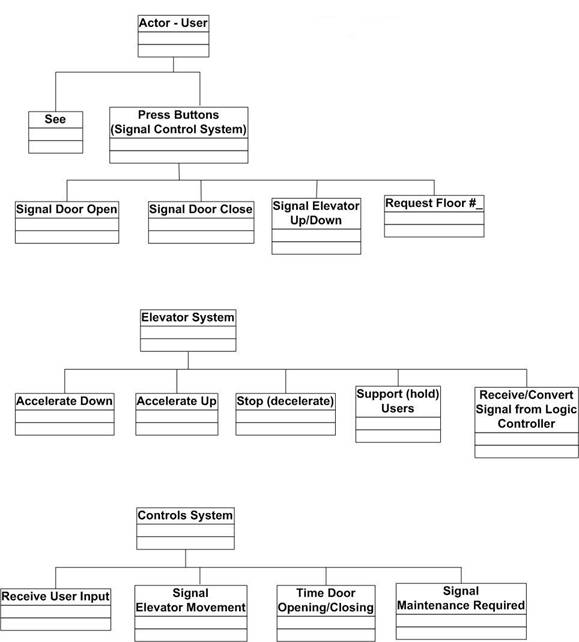
**Figure 1.**Flow of Events for Use Case 1.

**Use Case 2:**Floor Selection Use Case

* **Primary Actor(s)**: Elevator User, Elevator Logic Controller   
  **Description**: The Elevator User uses the Elevator Control System (ECS) to move to a selected floor.  
  **Preconditions**: An Elevator User is inside the elevator.   
  **Flow of Events**: The Elevator User finds the ECS panel in the elevator :
  1. User selects their desired floor button on ECS panel.
  2. The ECS sends a signal to the ELC, detailing which floor the user wants to go to.
  3. The request is added to a list of floors to visit.
  4. The ELC determines which direction to move to service the next request.
  5. The doors are closed, if open, and the elevator begins to move towards the requested floor.
  6. The ELC checks whether to stop as the elevator approaches a floor.
  7. The elevator stops at requested floors that are along the route to the original destination floor.
  8. The ELC opens the elevator door when it comes to a stop.
  9. The elevator arrives to service the selection of the user.

**Postconditions**: Elevator arrives at floor selected by the user.   
**Alternative Flow of Events**: None.   
**Assumptions**: None.

We view overall behavior of the "elevator system" as the combination of systems -- the users, the elevator system, and the logic control system. The elements of each system are summarized in Figure 2.



**Figure 2.**Elements of behavior for users of the elevator, the elevator system itself, and the elevator control system.

Operational Scenarios

**Scenario 1**

* **Scenario 1.1.** When a user presses the up or down button, the elevator will begin moving towards them.
* **Scenario 1.2.**When the user, already in the elevator car, presses the button for their desired floor, the elevator will begin moving towards that floor.
* **Scenario 1.3.**  When the user presses the Stop button, the elevator will decelerate and stop immediately.
* **Scenario 1.**4.  When the user, inside the car, presses the Door Open or Door Close button, the door will react as conditions permit.

**Scenario 2**

* **Scenario 2.1.**The buttons for the Elevator Request System (Up/Down) must be distinguishable.
* **Scenario 2.2.**The buttons for the Elevator Control System (Floors, Door Open/Close, Start/Stop) must be easy to use.

Elevator Control System

Every elevator system must be equipped with a control system that will receive the rider input and translate that into a signal that will control the actual elevator hardware.

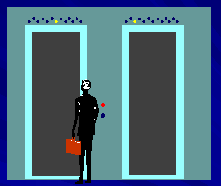
**Logic Controller**

These systems include a logic controller that takes the rider input and translates it into meaningful actions. The logic controller should be given at least three critical pieces of information, namely:

* Where people want to go?
* Where each floor is?
* Where the elevator car is?

The first input comes directly from the riders. The user pushes the floor number inside the elevator car signaling their final destination. When the floor buttons are pushed the logic controller receives the signal and registers the user's request.

The second input "where each floor is" can often be determined by the addition of holes located on a long vertical tape inside the elevator shaft. The elevator car is equipped with a light or magnetic sensor that reads the number of and which holes are being passed by the elevator car as it ascends and descends. The controller is equipped with a means of varying the speed of electric motor (connected to either of the following system designs) so it can slow down the car when it is approaching a floor at which it is to stop.

**Scheduling Elevator Operations**

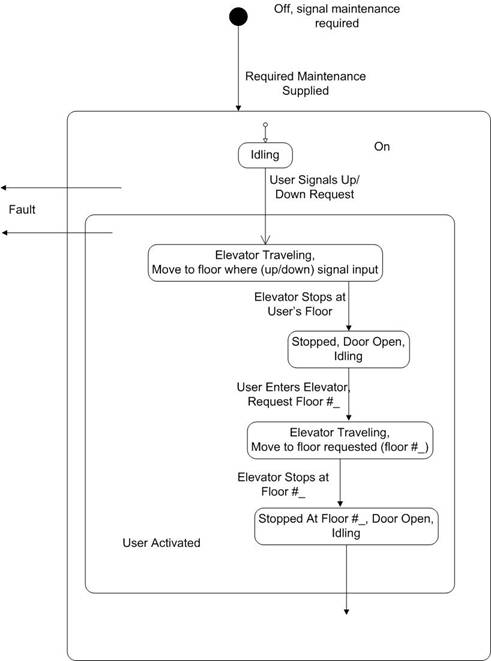
Logic controllers must have some way to determine in what order riders should be picked up and dropped off. Many elevator systems will move in one direction (e.g., upward) and only pick up riders that are also signaling to go in that direction (e.g., upward). When the final floor that has been requested in that direction (e.g., upward) is reached the elevator will turn around and pick up all riders signaling the opposite direction (e.g., downward). Of course, the elevator car also stops at all floors for which riders, already inside the car, have input a requested. A more sophisticated system, often used in hotels and other large buildings with a lot of foot traffic, involves the traffic patterns that reoccur. These systems have logic controllers that are programmed with information about the demand on each floor with respect to the time of day and they route the elevator cars accordingly so as to minimize the wait for all riders. When there are multiple elevator cars, the logic controller bases the movement on each car on that of the others. Often, the elevator car is equipped with a load sensor so that if the elevator is full to capacity it sends a signal to the control system and the logic controller signals the car not to pick up any more passengers until the load is lowered.

**Elevator Door Control**

The control system's computer also controls the movement of the elevator car doors. The amount of time for which the doors are held open when a floor is reached is programmed into the logic controller. The elevator car doors also have a sensor that detects if someone or something is caught in the door and stops the door closing mechanism from closing the door with the large force that is required. This is also part of the safety system since it ensures that people are not hurt when trying to enter or exit the elevator car.

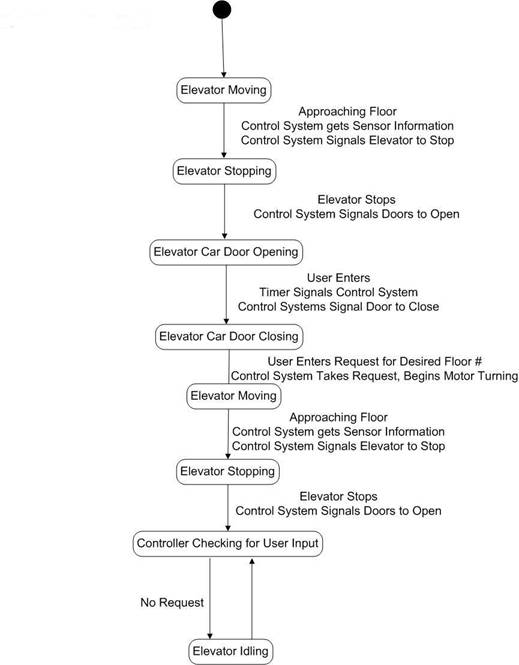
Statecharts for System Behavior

**Statechart for "User" Behavior**



**Figure 3.**Statechart for User Activity

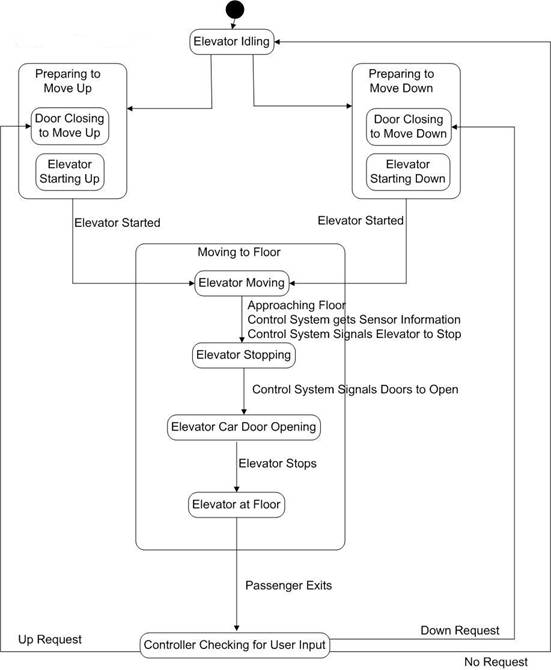
**Statechart for "Elevator System"**



**Figure 4.**Statechart for Elevator System.

The elevator system is comprised of two major subsystems: the control system and the mechanical hardware.

**Statechart for "Controls System"**



**Figure 5.**Statechart of Controls Systems Behavior.

Deliverables

1) Detail descriptions of all your assumptions, inputs, outputs, etc.

2) Diagrams and explanations of all your High-Level and Finite State Machines.

3) System level architecture of your controller.

4) Karnaugh Maps of all state bits and outputs.

5) SOP/POS equations of all state bits and outputs, along with their circuit diagrams.

6) VHDL codes and testbenches of the entire system.

**Submission Guidelines**

1. All files are to be submitted in .vhd format only. You can find your VHDL codes (in .vhd format) in U:\linux\apps\vsim  
  
2. Create a folder named as <last\_name>\_<first\_name>\_3181Elevator\_Project. Create a folder inside this folder named modelsim. Copy all the .vhd files into this folder.

3. Apart from these folders the master folder should also have a PDF document named "Report" which describes the basic methodology involved in writing the code and also the parts that are working and the ones that are not.   
  
4. The report should have the following sections  
  
a) Problem Description

b) Detailed outline of all your assumptions, inputs, outputs, etc.  
c) System level architecture  
d) Implementation and explanations of the HLSM(s), FSM(s) and Datapath(s)

e) Detailed description of simplification process of the FSM using K-Maps/Quine-McCluskey Methods  
f) Detailed explanations for cases that work and the ones that don't  
g) Suggest Future Work and Upgrades  
  
5) Create a zip file (no other compression format will be accepted) of the folder described in (2) and upload the zip file on Moodle.