MVP Busyness/Functional Requirements

1) The elevator must serve a building with more than one floor. (floor quantity is not fixed).
2) The elevator must be controllable by users from inside and outside going up and down to transport people between floors.
3) The doors of elevator are always open in every floor while there is no any command from inside.
4) External calls must be ignored if the elevator is already called or is performing its task delivering the passenger target floor.

MVP Technical Requirements

1) Input devices
- **a) Sensors** on every floor to detect elevator's position.
- **b)** Internal floor-selection **buttons**.
- **c)** Single external call **button** per floor.

2) Output devices
- **a) Engine** for elevator movement (up/down).
- **b) Engine** for door open/close mechanism.

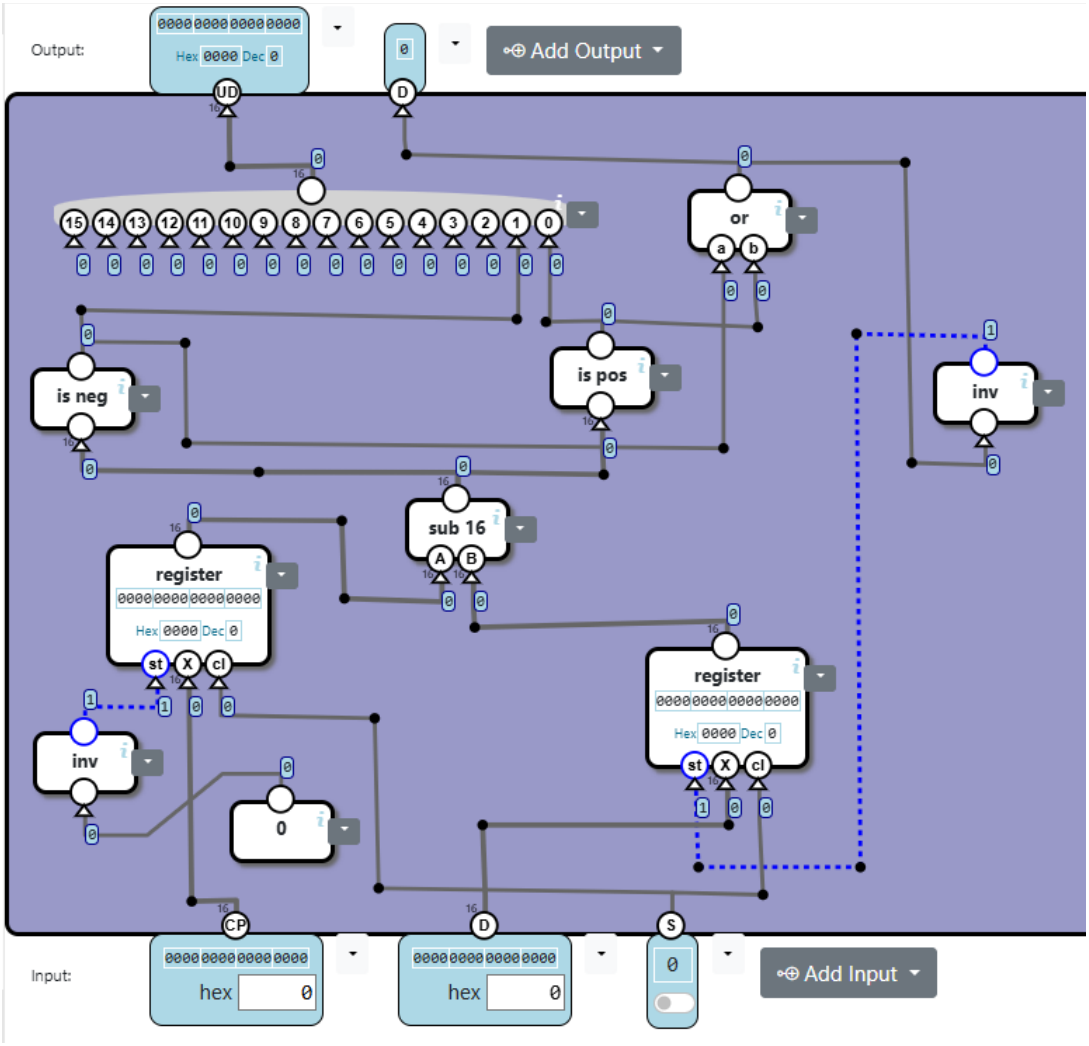
3) Controller which must be designed by me working with electric current to manage and control the functionality of the elevator system.

Controller Design Details

Controller has 3 input ports:
1) CP – Current position of elevator. It is 16-bit input port which receives the floor number (in binary system) from sensors identifying the current position of elevator.
2) D – Direction where to elevator must go. It is 16-bit input port which receives the direction floor number (in binary system) from internal or external buttons pressed by passenger.
3) S – Signal. It is 1 bit input port which gets 1 the moment when any internal/external button is pressed, or a signal comes from a sensors, and then it resets to zero.

And it has 2 output ports:
1) UD – Up/Down. It is 16-bit output port which that indicates the movement direction of the elevator. 01 - the elevator goes down, 10 – the elevator goes up, 00 – the elevator stays in place.
This signal is sent to the elevator engine.
2) D – Door is closed/open. It is one bit output port which sends to engine 1 to close the door, and 0 to open it.

The implemented scheme details is below:



We have 2 registers to store the signals from ‘CP’ and ‘D’. They are for synchronization, so that the signals reach to ‘sub’ simultaneously which provides ‘cl’ entry of registers. And also the registers help to handle 4th point mentioned on “**MVP Busyness/Functional Requirements**”.
The data is stored in register when ‘cl’ = ‘st’ = 1, but stored value is not yet output and the moment ‘cl’ changes to 0 the previous stored value is output.
For leftmost register ‘st’ is always 1, because we don’t have problem to handle signals from sensors.
The ‘st’ entry of the rightmost register depends on ‘inv’ which depends on ‘or’.
The ‘or’ component checks if the elevator is working (goes up/down) or not. If it is, then it sends 1 to ‘inv’. In this case ‘inv’ will send 0 to ‘st’, so every new external call will be ignored not to be stored in register.

‘sub’ is for understanding the destination the elevator should go. Then the decision data is stored in first 2 bits of 16-bits placer.
‘D’ output port receives data from ‘or’ component. So when the elevator is moving , it gets 1 to make engine close the door and when elevator reaches the target, it gets 0 to make engine open the door.

Future consideration

1) The min weight the elevator can transport is 40 kg.
2) The max weight the elevator can transport is 400 kg.
3) The light inside elevator must be on if and only if there is a passenger inside.
4) The doors must be open if and only if the elevator reaches the target (to take/leave passenger).
5) The elevator doesn’t contains emergency button.
6) The speed of movement of the elevator is fixed (there is no any button to change it).