ADIC Assignment
(Automated Lift Controller)

Name :- Rudrajyoti Roy
Roll :- 18EC10047
Date of Submission :- 8th Nov, 2021

Problem Statement

Design a lift-controller for a 7 storey building.

System Definition – Input

A 7-storey buiding has a single 2-way lift. There are buttons on every floor for requesting to go up or go down. The switches are of pushbutton type, which are OFF in normal state and ON only when pushed. Additionally, there are floor buttons on the elevator.

In total, for an N storey building, there are N switches for going up (US) - one at each floor, and similarly N switches for going down (DS) - one at each floor, and N switches for floor selection (SS) on the elevator.

The UP switch and DN switch at top floor and ground floor may be assumed to be disabled.

Other switches such as alarm, force stop, door open(force) can be ~~door~~ added, but are not shown here. For door open (forced), a button added to the timer for async reload would be sufficient.

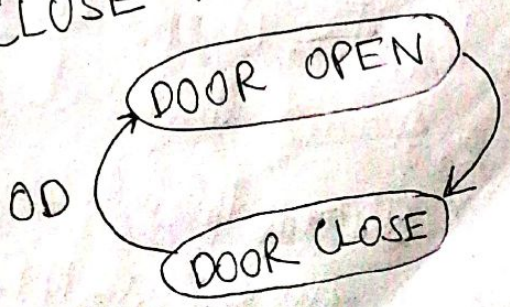
System Definition - Output

(Actual Relay to motor connections are not shown)

1) Elevation Control :- The outputs for controlling elevation are two bits MOVE and DIRECTION (MODE). The behaviour of the elevator would be : (This is combinational design using relays)

MOVE	MODE	ELEVATOR
0	X	STOP
1	0	UP
1	1	DN

2) Door Control :- The output for controlling door is FSM based with inputs "OPEN DOOR" and "CLOSE DOOR"



OD	CD	Q_{next}
0	0	Q
1	0	Door open
0	1	Door close
1	1	Not Allowed

Request Array (Register)

The pushbutton switches are used to create a request, but storing the requests require register bank.

In this approach, we use separate registers for storing STOP Request (SR), UP Request (UR), Down Request (DR). Each one is N-bit register.

The registers are connected to SS, US and DS switches respectively, but the switches can only be used to set a request, while clearing a request is done internally.

Suppose persons inside the elevator want to deboard at floors (1, 3, 5, 7). People want to go up at floors (2, 3) and down at (3, 4).

In that case, SR = (1, 0, 1, 0, 1, 0, 1),

UR = (0, 1, 1, 0, 0, 0, 0), DR = (0, 0, 1, 1, 0, 0, 0)

Algorithm Level Design (MODE = DN if 1 = UP if 0)

- **MOVE if and only if**

a) Not at any floor ($\forall i \; F[i] == 0$)

OR

b) i) At any floor ($\exists i$ s.t $F[i] == 1$)

AND door closed AND

ii) No request at floor i to STOP or go in current direction

$$[SR[i] + (MODE \cdot DR[i] + \overline{MODE} \cdot UR[i])] == 0$$

AND

iii) At least one request at floor j (further in the same direction) for STOP, UP or DN.

$$MODE \cdot \sum_{j=0}^{i-1} (SR[j] + UR[j] + DR[j])$$

$$+ \overline{MODE} \cdot \sum_{j=i+1}^{N} (SR[j] + UR[j] + DR[j]) == 1$$

- **TIMER START and DOOR OPEN if**

a) Elevator has stopped at any floor (NEGEDGE or MOVE (stopped condition))

OR

b) Elevator is at any floor, and somebody presses SS, US or DS corresponding to that floor.

(POSEDGE of $\sum_{i=1}^{N} \overline{MOVE} \cdot [SR[i] + UR[i] + DR[i]] \cdot F[i]$

~~Not correct~~

c) Explicit door open switch pressed.

- WHEN TIMER STOPS (OVERFLOWS)

i) (⊓) Pulse on overflow flag as timer overflows and reloads (but does not restart again) T_{wait}.

ii) Overflow pulse clears SR[i] and UR[i] or DR[i] as per selected mode.

iii) Overflow pulse also closes door (door close)

- MODE SWITCH (MODE = $\overline{MODE}$) if and only if

i) At any floor i ($\exists$ i s.t. F[i] == 1) AND

ii) No request at floor i to stop or go in current direction.

SR[i] + MODE · DR[i] + $\overline{MODE}$ · UR[i] == 0 AND

iii) No request at any floor j (further in the same direction) for STOP, UR or DR

$$MODE \cdot \sum_{j=1}^{i-1} (SR[j] + UR[j] + DR[j]) + \overline{MODE} \cdot \sum_{j=i+1}^{N} (SR[j] + UR[j] + DR[j]) == 0$$

- Consequences of MODE SWITCH

i) MODE = $\overline{MODE}$

ii) Re-trigger overflow pulse (clear SR[] and UR[] or DR[])

NOTE: It is trivially ensured that MODE is not switched while elevator is moving.

Specialized Hardware Blocks (Logical Design)

1) Switch to Request Register Connection

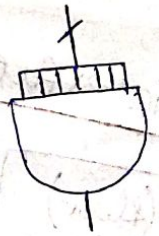
The switches can only be used to set a request. While clearing is done by elevator when the request is serviced.

First, three types of gates

i) **Bus bitwise gate** :- Bit by bit operation on all bits individually.

ii) **Bus-with-bit gate** :- One bit is operated against every bit of a bus. Output is also a bus.

iii) **Bus-to-bit gate** :- N-input gates that operated on all the lines of a bus. Input is a bus and output is a single bit.



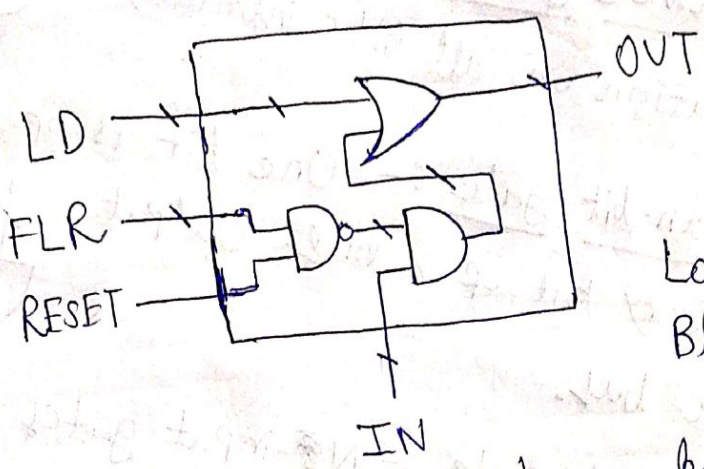
Now, back to the block, for clearing a register bit, we have unique reset signal (bit) which is bitwise AND-ed with the current floor array, to get the bit that needs to be reset.

NOTE :- We can attach LEDs to indicate the status of the registers for visual feedback to consumers (whether their requests are registered or not)

LD	RESET	OUT (Q) (Corresponding bit line)
0	0	IN(Q)
0	1	0
1	X	1

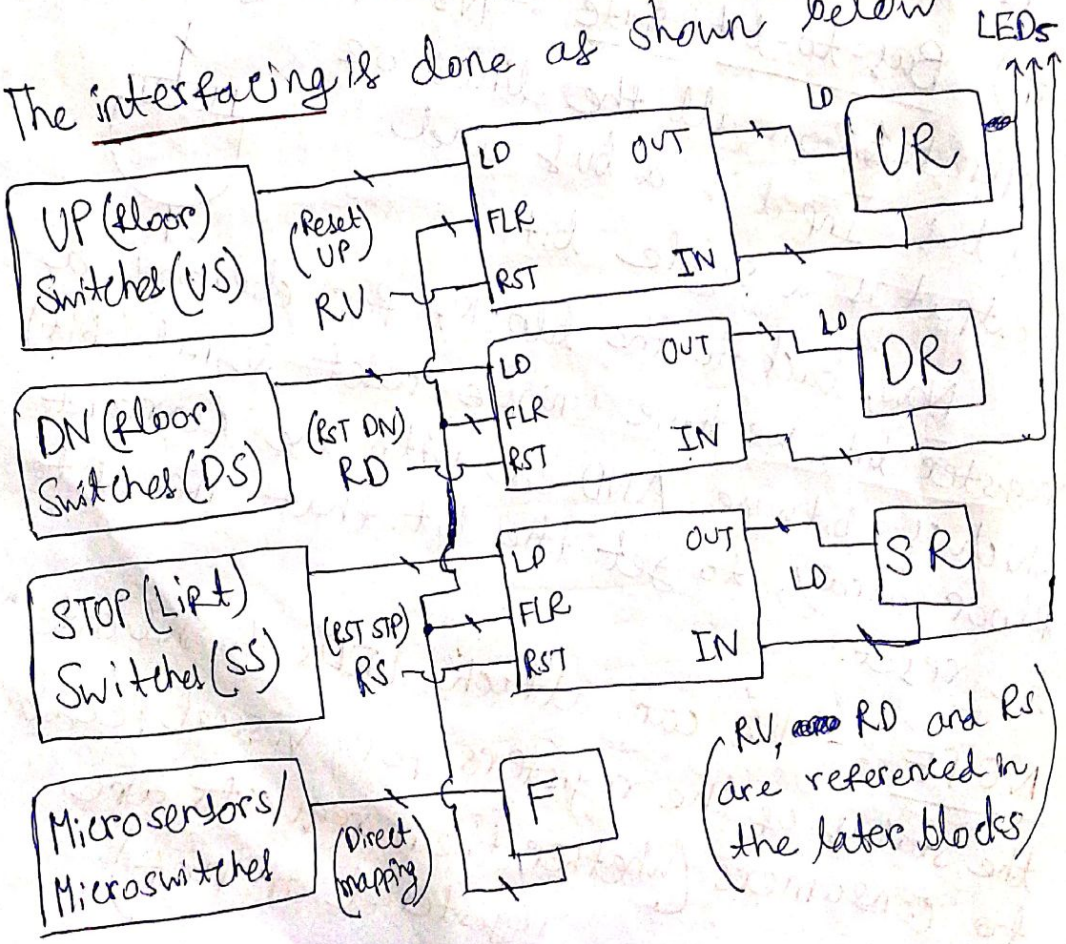
Here reset is converted to array by bitwise AND-ing with floor indicator array.

$$Q'_{next} = LD + \overline{RESET} \cdot Q$$



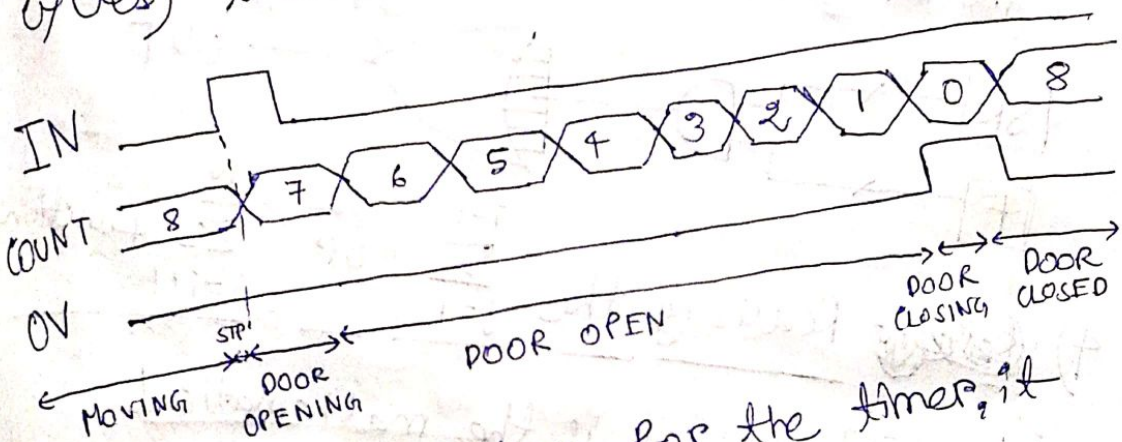
Loading Block

The interfacing is done as shown below



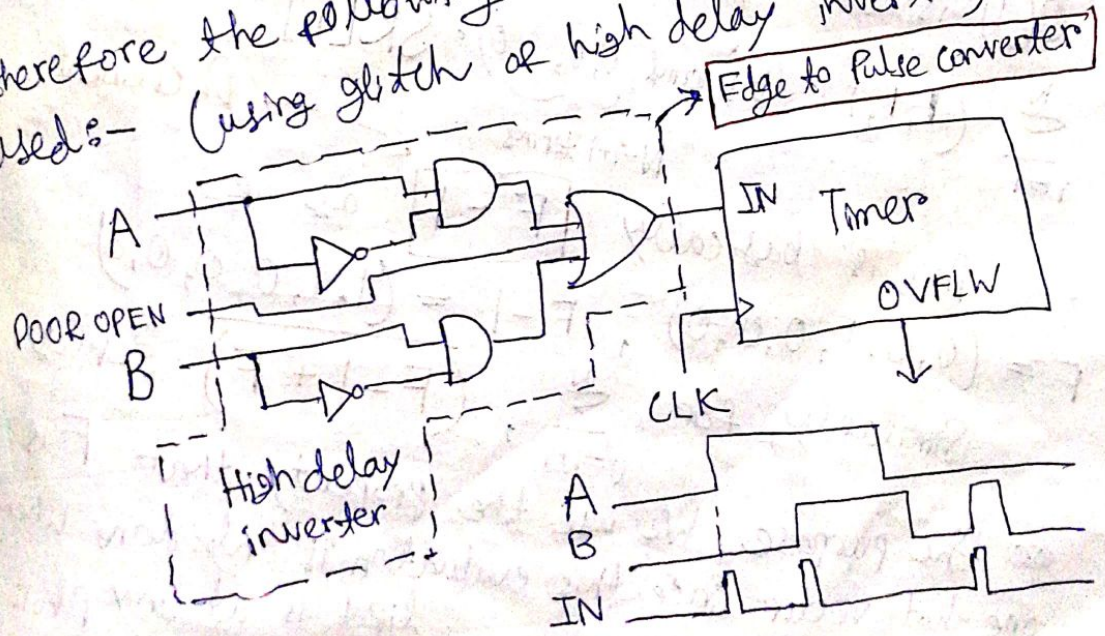
2) The Timer :- The timer is used to add a finite time (T_{wait}) of the order ~ 10 sec between door open and door close.

It is triggered by positive edge of input signal. When it reaches 0, it overflows (holds the overflow flag high for few clock cycles) then reloads the same value.



As discussed earlier, for the timer, it should be started if positive edge occurs for $\overline{MOVE}(A)$ or $\sum_{i=1}^{N} \overline{MOVE} \cdot F[i]\left[SR[i] + UR[i] + DR[i]\right]$

Positive edges cannot be combined as such, therefore the following edge-to-pulse conversion is used:- (using glitch of high delay inverter)

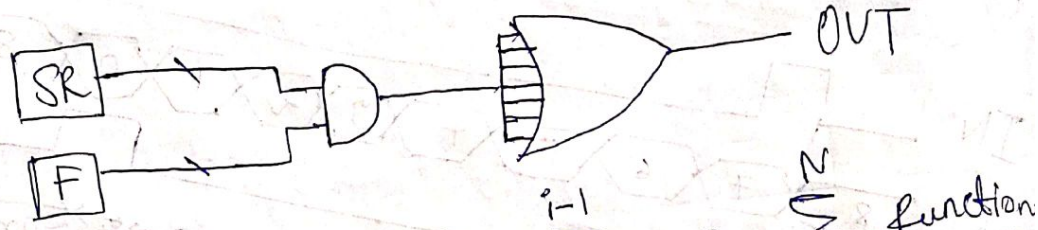


3) Realising the $\sum_{i=1}^{N}$ function

This can be done using N-input OR gate (bus-to-bit). To identify whether there is a stop request at the current floor, the O(1) approach is $= \sum_{i=1}^{N} SR[i] \cdot F[i]$

↑ One-hot vector

The implementation :-



4) Realizing the $\sum_{j=1}^{i-1}$ or $\sum_{j=i+1}^{N}$ function

$\sum_{i=1}^{N} \sum_{j=1}^{i-1} F[i] \cdot SR[i]$ is the mathematical expression for checking whether there is any stop requests at any floor below the current floor. The method of converting this $O(N^2)$ search to O(1) is :-

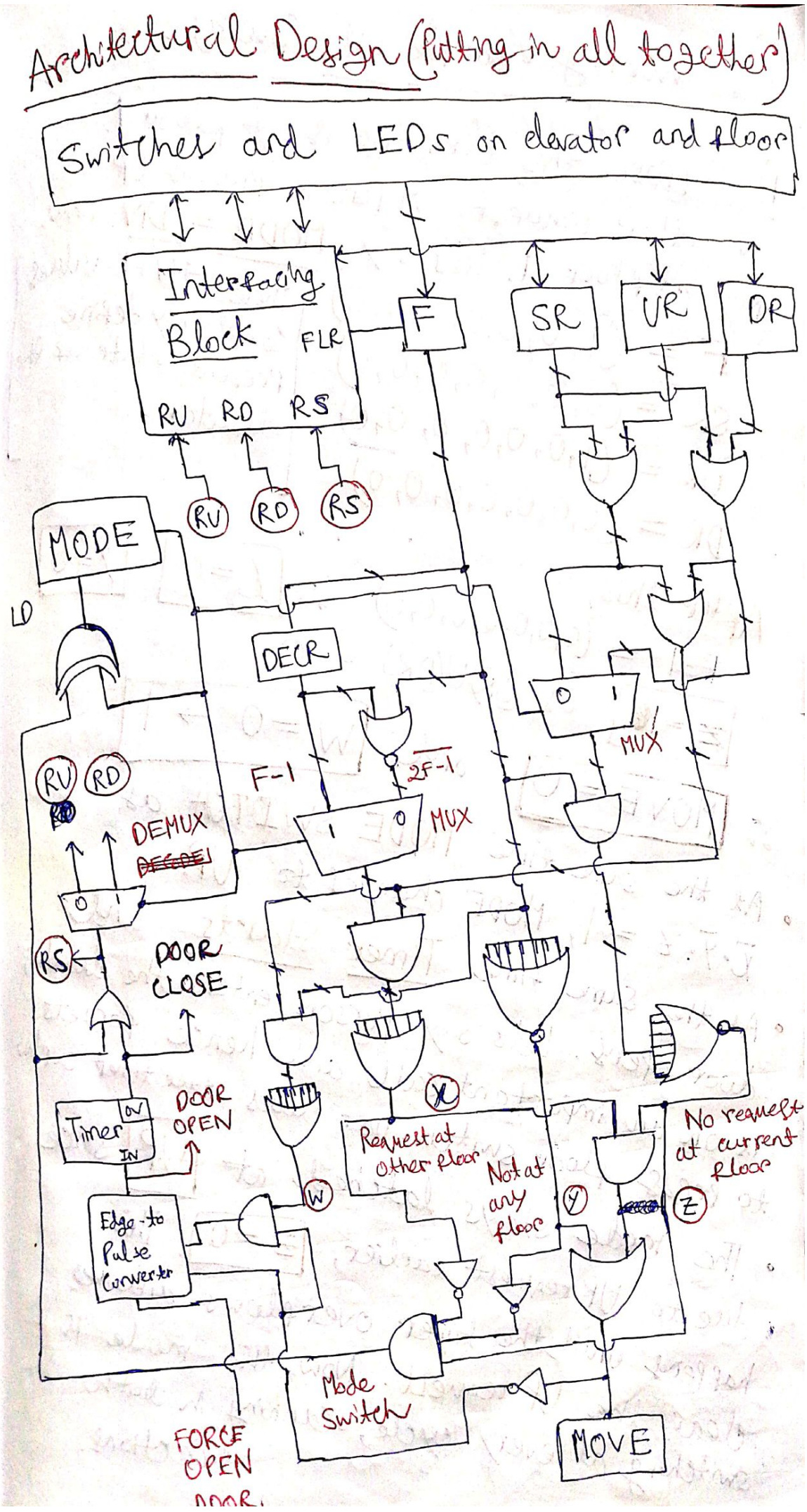
$\sum_{i=1}^{N} (1,1,1 \ldots (i-1) \text{ terms}, \underbrace{0,0,\ldots 0}_{N-i+1 \text{ terms}}) \cdot SR[i]$

This is basically $\boxed{F-1}$ as in $F = (0,0,1,0,0,0)$, $F-1 = (1,1,0,0,0,0)$

← Decrement

Similarly, for $\sum_{j=i+1}^{N}$, $(F-1+F)$ serves the purpose. We use the property that F is one-hot vector since this evaluation is only done when lift is at any floor.

Architectural Design (Putting in all together)



Working Example and Coverage

1) Suppose the elevator is at floor 1. No other request. A person presses UP key at floor 1. Let's say MODE = DN. now.

$F = (1,0,0,0,0,0,0)$
$SR = (0,0,0,0,0,0,0)$
$UR = (1,0,0,0,0,0,0)$
$DR = (0,0,0,0,0,0,0)$

These 4N+1 values completely define present state of the elevator.

After this,

$F-1 = (0,0,0,0,0,0,0)$ $\therefore$ $\boxed{x=0}$ $\boxed{y=0}$

$\boxed{z=1}$ as $(SR)\cup(DR) = 0$

$\therefore$ $\boxed{MOVE = 0}$ and $\boxed{W = 0 \rightarrow 1}$

- At the same time MODE SWITCH as $\bar{x}\cdot\bar{y}\cdot z = 1$, MODE changes to UP.
- At the same time Timer Starts and door opens. Let's say person enters the elevator, is on an important call and hence forgets to press floor switch. Door closes after timer overw.
- The mode stays latched at $\boxed{UP}$ since due to UP request earlier, $\boxed{z=0}$. This happens until the timer overflows and clears the UP request. Now, the mode is switching at every cycle, scanning in both directions.

2) Now, say the person presses floor 3.

So, MODE = UP

$F = (1, 0, 0, 0, 0, 0, 0)$

$SR = (0, 0, 1, 0, 0, 0, 0)$

$UR = DR = (0, 0, 0, 0, 0, 0, 0)$

$\overline{2F-1} = (0, 1, 1, 1, 1, 1, 1)$

So, now $\boxed{X = 1}$ $\boxed{Y = 0}$ and

$\therefore$ $\boxed{Z = 1}$ as $(SR+UR)$ $F = (0, 0, 0, 0, 0, 0, 0)$

$\therefore$ After door closes and UR is serviced, the lift starts moving upwards.

As the lift moves UP, $F =$ (all zeroes)

so, $\boxed{X = 0}$ and $\boxed{Z = 1}$ but $\boxed{Y = 1}$ keeps the lift moving.

3) Suppose POWER CUT occurs and lift gets stopped midway. Now, when power comes back, all the values are 0.

$\boxed{Y = 1}$ and default mode = DN, so lift starts coming down.

Once it reaches floor 1 (in general case, the immediate next floor), $\boxed{X = 0}$ $\boxed{Y = 0}$ $\boxed{Z = 0}$. So, lift stops, door opens, timer starts. After door closes again, the lift does NOT move further unless re-instructed.

4)a) Now, the person again presses floor 3, the lift is going up as expected. Now, before floor 2, say somebody presses UP request at floor 2. When lift reaches floor 2, MODE = 0 (UP)

F = (0, 1, 0, 0, 0, 0, 0)

SR = (0, 0, 1, 0, 0, 0, 0)

UR = (0, 1, 0, 0, 0, 0, 0)

DR = all zeroes.

∴ z = 0 y = 0 x = 1 w = 1

So, MOVE = 0 and lift stops at floor 2. As moving stops, we have a positive edge at w = 1. Timer starts. door opens. When door closes, the request is cleared and UR = all zeroes.

4)b) Say instead of UP request, somebody presses DN request. So, when lift reaches floor 2, UR = all zeroes

DR = (0, 1, 0, 0, 0, 0, 0)

z = 1 y = 0 x = 1 w = 1

The elevator doesn't stop, stops at floor 2, after door closes, stop request cleared (x=0) mode switches

3. Door opens, the mode gets latched at 1 as x = 1 again. The lift stops at floor 2 on the way down and clears the down request.

Working Behaviour and Discussion

i) The elevator design does not necessitate use of intermediate switches. The direction and stop/move decisions are only taken when the elevator is at any particular floor and thus F[i] == 1 for some i.

Thus emergency power down scenario is covered as on powerup, all registers would be initialised to 0. Therefore, lift will start moving UP (if it is not any floor). If it is/ as soon as it reaches any particular floor, it will remain stationary.

ii) It is essential that all the floor sensors are in working order. In case any floor sensor output is not held high for more duration than critical path to MOVE output, it is possible that the lift may run out of safe area. Otherwise remaining in safe area is guaranteed.

iii) Requests are serviced in order of closeness in the current direction. The direction only changes after all the requests in the current direction (Stop/Up requests for up direction and also any stop/up/down requests from any floor in up direction, similar for down) are serviced and cleared. If new requests arrive, to service which the lift shall not need to change direction, the new requests would be serviced before changing direction.

iv) When no new requests are pending in any direction, the mode keeps complementing on every cycle, thereby scanning both directions. If any new request arrives:

a) At current floor: door opens timer restarts

b) At/For any other floor: Direction(mode) is latched in that floor's position w.r.t lift and lift starts moving. stops at that floor and opens door.

v) The force door open button is a safety feature to open door and restart timer irrespective of position. It can also be used to keep the door open for longer than usual time.

vi) LEDs can be added to SR, UR, DR registers for visual feedback and diagnosis.

vii) The setup acts well for any N-storey building. Only registers and switches need to be N-bit. This setup is NOT cascadable.

viii) DN requests from passing/stopping floors are ignored while moving UP and vice versa.

ix) With only set-reset and toggle type memory elements, the circuit can be ~~designed~~ asynchronous except for the door open/close clock.