DESIGN OF ELEVATOR CONTROLLER

Self Project

BY

P.Nagasai Goud 203070094

DEPARTMENT OF ELECTRICAL ENGINEERING

IIT BOMBAY

Main Gate Rd, IIT Area, Powai, Mumbai, Maharashtra 400076

Introduction

This project is designed for an eight floor elevator controller. The elevator decides moving direction by comparing request floor with current floor.In the proposed design a VERILOG RTL code is developed to control the lift moment based on the request it will get. For that a finite state machine is developed to know from which state to state the controller is changing based on the requests from the end user. The design is based on the synchronous input which should be operating with a fixed sort of frequency. Finally the RTL is verified and implemented. In this work, the lift controller will be modeled with Verilog HDL code using Finite-State machine (FSM) model to achieve the logic in optimized way.

The Elevator Algorithm

The elevator algorithm, a simple algorithm by which a single elevator can decide where to stop, is summarized as follows:

- Continue traveling in the same direction while there are remaining requests in that same direction.
- If there are no further requests in that direction, then stop and become idle, or change direction if there are requests in the opposite direction.

The elevator controller is based on the concept of finite state machine technology. According to the FSM technology the elevator process can be defined with the help of different states. In the FSM technology there is a change from one state to another state likewise in the elevator there will be a change from one floor to another. Every possible way is assigned a path and the implemented based on FSM concept to write the program code for elevator controller.

Here it is implemented in FSM with 6 states

- IDLE
- UP
- DOWN
- OPEN
- WAIT
- CLOSE

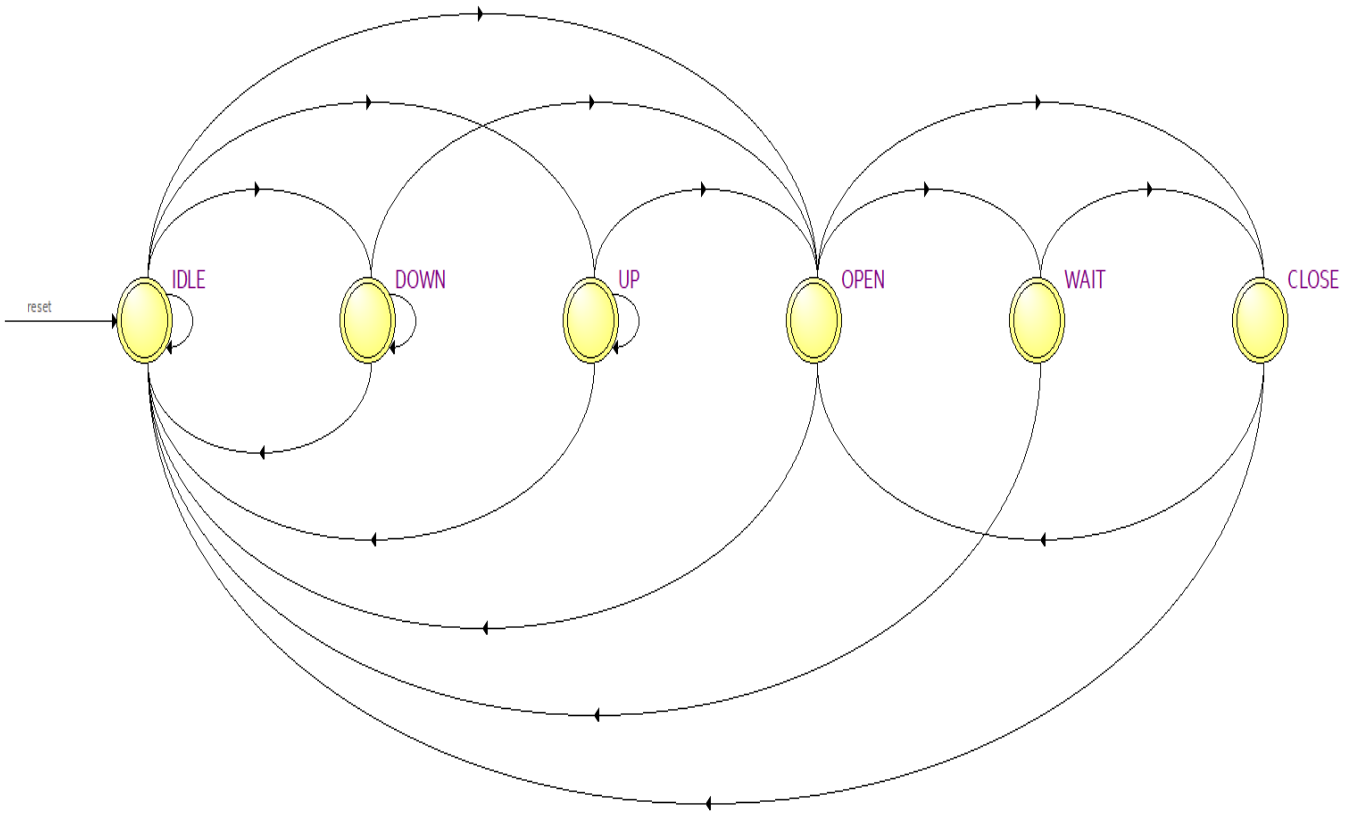

Figure 1: FSM

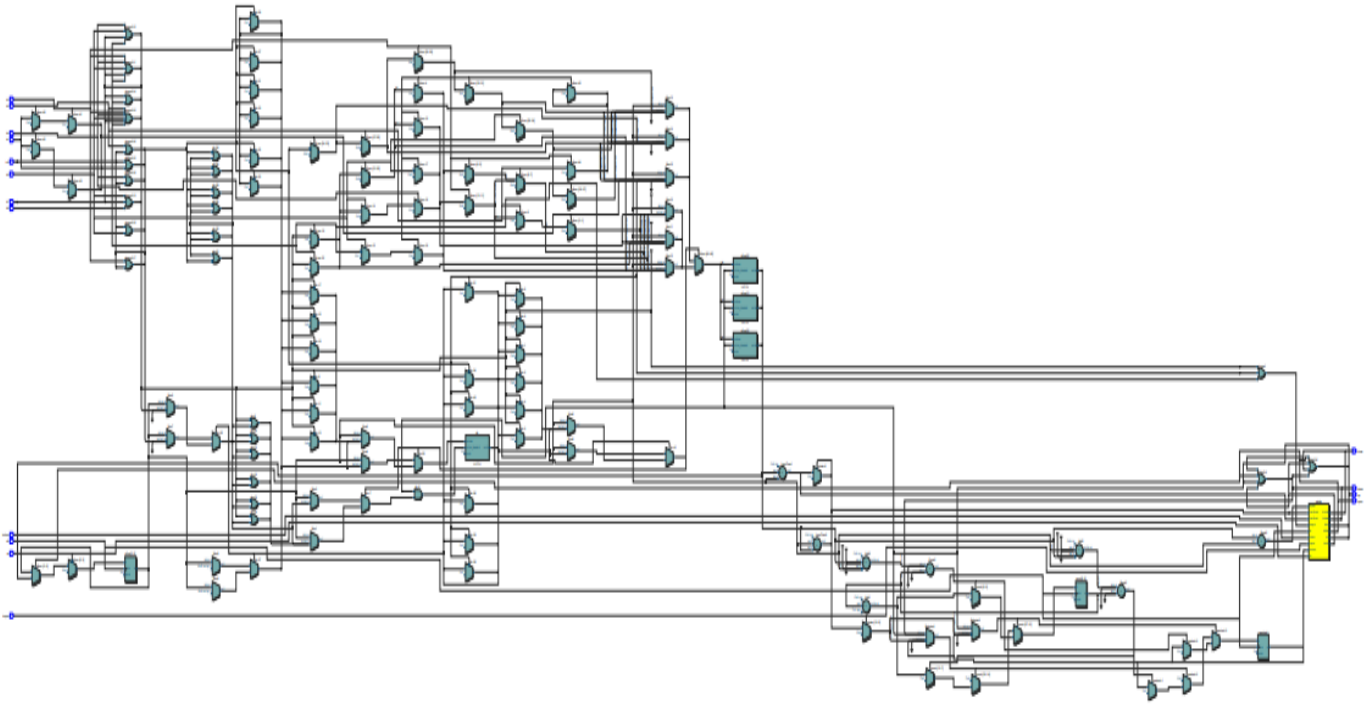

Figure 2: RTL View

```
# next_floor : x , present_floor:x , inputs(g,1,2,3,4,5,6,7) : 00000000
# next_floor : 0 , present_floor:0 , inputs(g,1,2,3,4,5,6,7) : 00000000
# next_floor : 1 , present_floor:0 , inputs(g,1,2,3,4,5,6,7) : 11101111
# next_floor : 1 , present_floor:0 , inputs(g,1,2,3,4,5,6,7) : 10101111
# next_floor : 1 , present_floor:1 , inputs(g,1,2,3,4,5,6,7) : 10101111
# next_floor : 2 , present_floor:1 , inputs(g,1,2,3,4,5,6,7) : 10101111
# next_floor : 2 , present_floor:1 , inputs(g,1,2,3,4,5,6,7) : 10001111
# next_floor : 2 , present_floor:2 , inputs(g,1,2,3,4,5,6,7) : 10001111
# next_floor : 4 , present_floor:2 , inputs(g,1,2,3,4,5,6,7) : 10001111
# next_floor : 4 , present_floor:2 , inputs(g,1,2,3,4,5,6,7) : 10000111
# next_floor : 4 , present_floor:4 , inputs(g,1,2,3,4,5,6,7) : 10000111
# next_floor : 5 , present_floor:4 , inputs(g,1,2,3,4,5,6,7) : 10000111
# next_floor : 5 , present_floor:4 , inputs(g,1,2,3,4,5,6,7) : 10000011
# next_floor : 5 , present_floor:5 , inputs(g,1,2,3,4,5,6,7) : 10000011
# next_floor : 6 , present_floor:5 , inputs(g,1,2,3,4,5,6,7) : 10000011
# next_floor : 6 , present_floor:5 , inputs(g,1,2,3,4,5,6,7) : 10000001
# next_floor : 6 , present_floor:6 , inputs(g,1,2,3,4,5,6,7) : 10000001
# next_floor : 7 , present_floor:6 , inputs(g,1,2,3,4,5,6,7) : 10000001
# next_floor : 7 , present_floor:6 , inputs(g,1,2,3,4,5,6,7) : 10000000
# next_floor : 7 , present_floor:7 , inputs(g,1,2,3,4,5,6,7) : 10000000
# next_floor : 0 , present_floor:7 , inputs(g,1,2,3,4,5,6,7) : 10000000
# next_floor : 0 , present_floor:7 , inputs(g,1,2,3,4,5,6,7) : 00000000
# next_floor : 0 , present_floor:0 , inputs(g,1,2,3,4,5,6,7) : 00000000
```

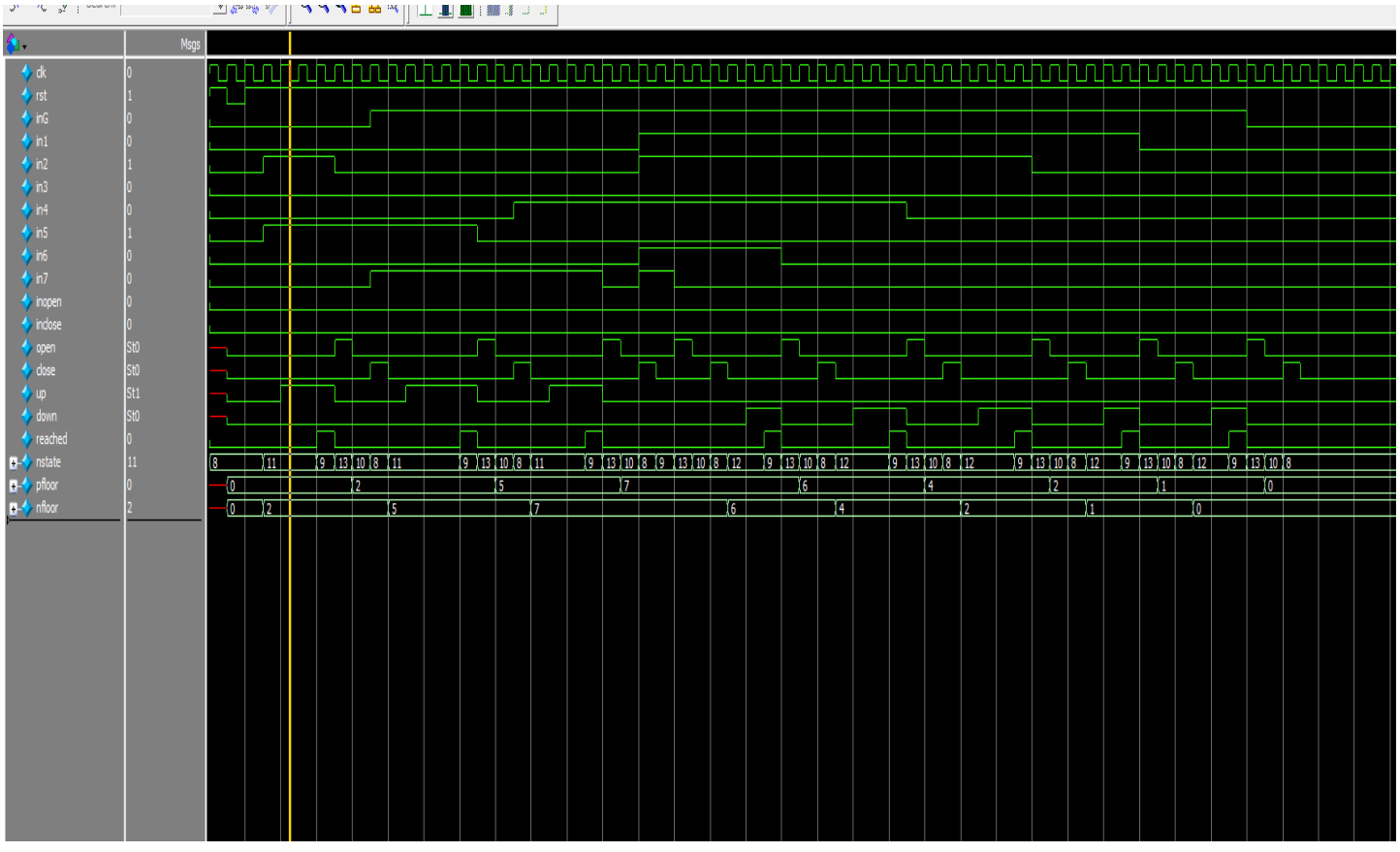

Figure 3: Output in ModelSim