Report for Final Project

이승원(20141440) 정준호(20141575)

2019.06.13 EE32002 Group 7

Objectives

◇ Construct a simple elevator system using Verilog and verify the system with MAX II Demonstration Board.

Observations

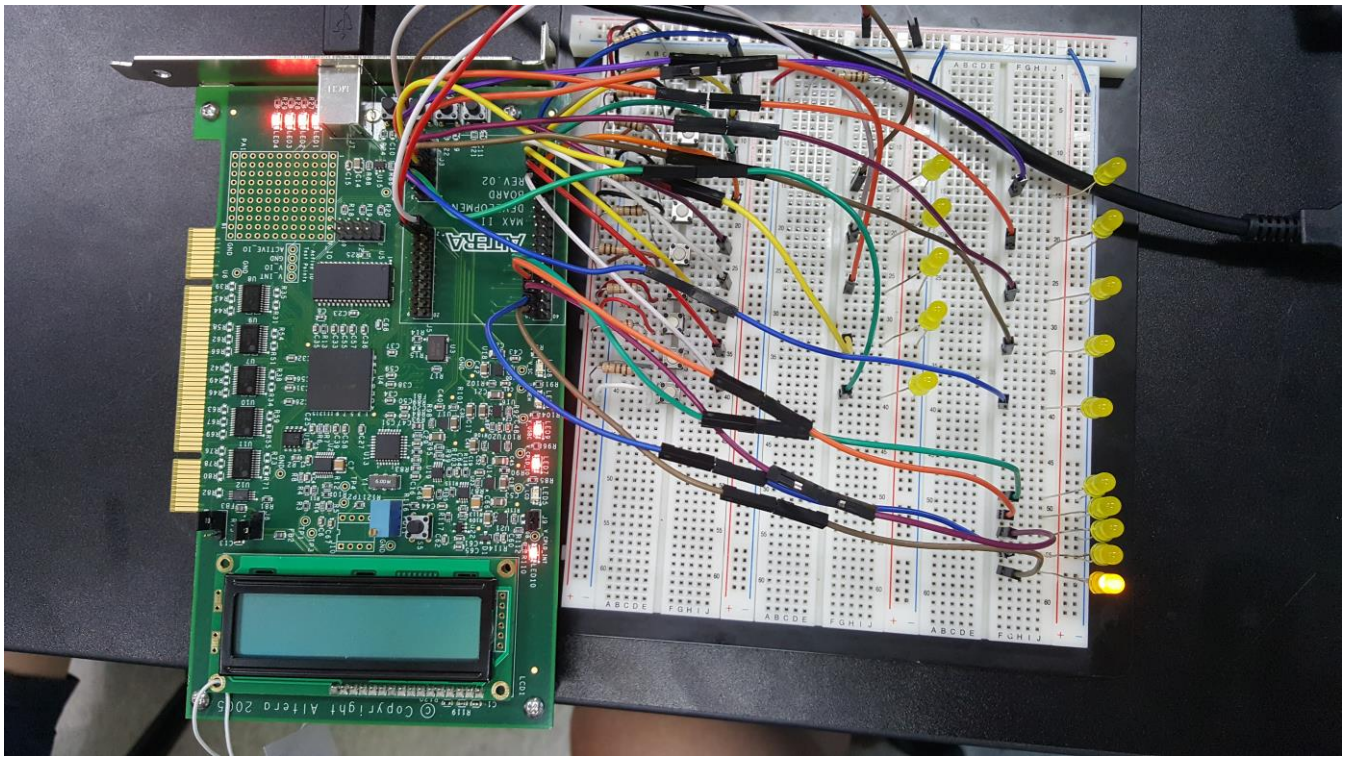

Fig 1) Initial State

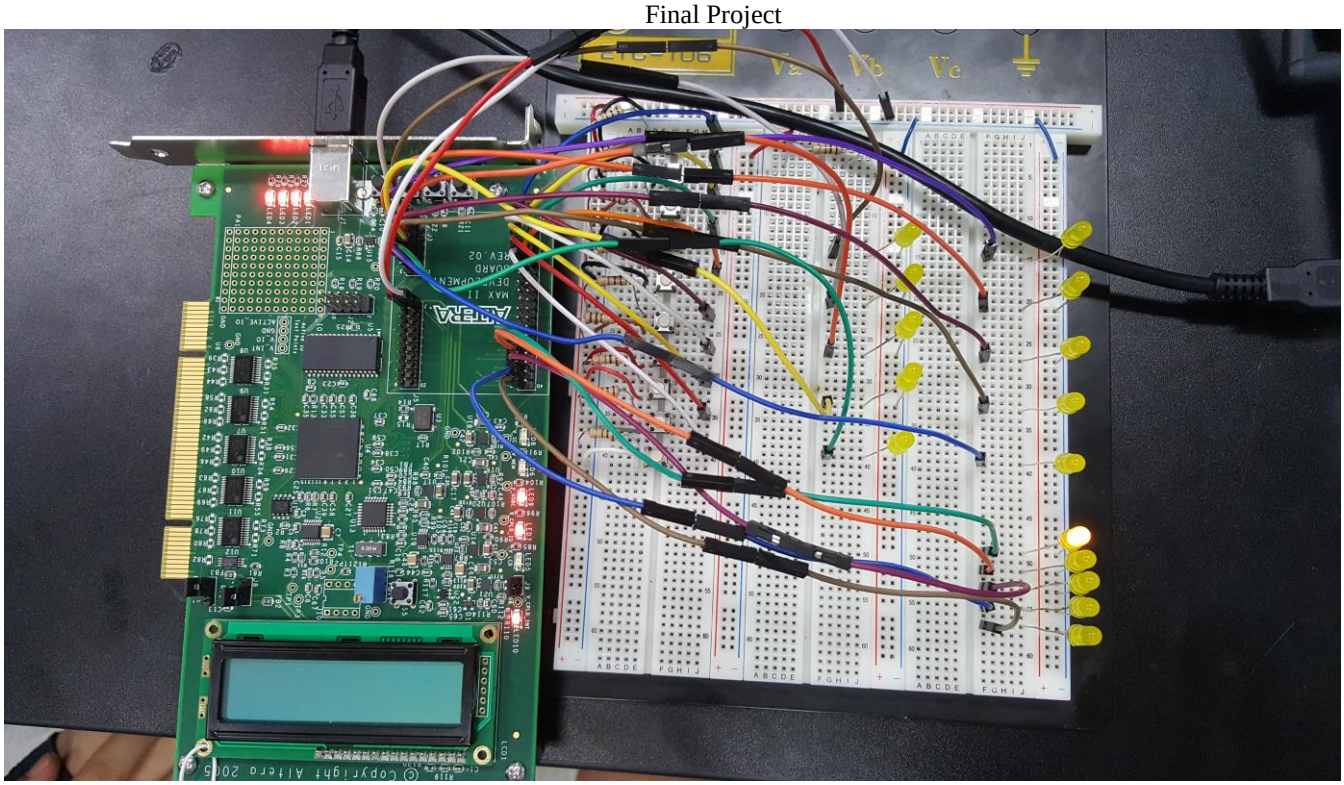

Fig 2) arrive on the 5th floor

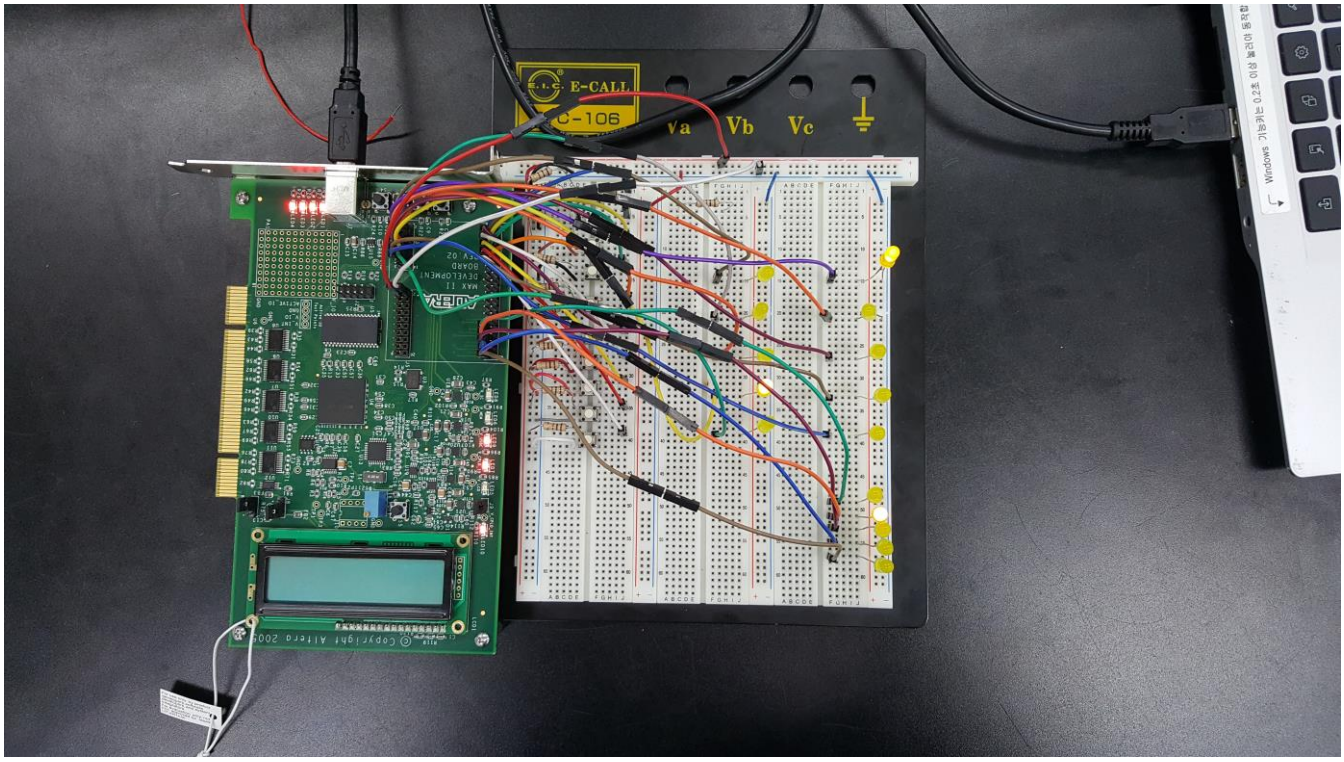

Fig 3) Progress of from 1th to 5th while another request to 2th floor from outside is received

Codes

```
module elevator(
    input request_out1,
    input request_out2,
    input request_out3,
    input request_out4,
    input request_out5,
    input request_in1,
    input request_in2,
    input request_in3,
    input request_in4,
    input request_in5,
    input clk,
    input resetn,
    output reg current_1,
    output reg current_2,
    output reg current_3,
    output reg current_4,
    output reg current_5,
    output reg output_request_out1,
    output reg output_request_out2,
    output reg output_request_out3,
    output reg output_request_out4,
    output reg output_request_out5,
    output reg output_request_in1,
    output reg output_request_in2,
    output reg output_request_in3,
    output reg output_request_in4,
    output reg output_request_in5
);
    reg move = 0;
```

Code 1) Input & Output

Code 1 shows the inputs and outputs of our codes. We separated the all individual inputs. Thus, the number of inputs is 10, i.e., 5 for the inside and 5 for the outside of elevator respectively. The number of outputs is 15, which are 5 for each inside, outside and indicator of elevator. At the bottom of Code 1, one variable is declared. It is used for making unable changing destination during moving. We will explain it next the code.

```
always @(posedge clk) begin
   if(!resetn) begin
    current_1 <= 1; current_2 <= 0; current_3 <= 0; current_4 <= 0; current_5 <= 0;
    move <= 0;
   end
   else begin

      if(move == 0 && request_in1 == 0) begin
      output_request_in1 <= 1;
      move <= 1;
      if(current_2 == 1) begin
      #2000000000
          current_2 <= 0;
          current_1 <= 1;
      end

      if(current_3 == 1) begin
        #2000000000
          current_3 <= 0;
          current_2 <= 1;
        #2000000000
          current_2 <= 0;
          current_1 <= 1;
      end
      if(current_4 == 1) begin
        #2000000000
          current_4 <= 0;
          current_3 <= 1;
        #2000000000
          current_3 <= 0;
          current_2 <= 1;
        #2000000000
          current_2 <= 0;
          current_1 <= 1;
      end
      if(current_5 == 1) begin
        #2000000000
          current_5 <= 0;
          current_4 <= 1;
        #2000000000
          current_4 <= 0;
          current_3 <= 1;
        #2000000000
          current_3 <= 0;
          current_2 <= 1;
        #2000000000
          current_2 <= 0;
          current_1 <= 1;
      end
      #5000000000
      move <= 0;
      output_request_in1 <= 0;
      end
```

Code 2) Main Algorithm

Code 2 shows the main algorithm of our codes. When the reset button is pushed, only current floor turns on and others turns off. If not, the algorithm will check which request button is pushed. For example if request button to the 1th floor from inside of elevator(request_in1 == 0) is pushed, then first, it will change ‘move’ variable to 1 to prevent code from executing other conditional statements. That is why there is “move == 0” in condition. After that, it checks where is the current floor. Then it starts to move to destination. Algorithm of other request buttons is same as this.

Results and Conclusions

Through the experiments, we could know how an elevator system works. We succeeded to make elevator go up or down to destination where user want to go.

However, we could not make progress after it. That is the hardest part of this experiment, delaying. We have tried multiple cases, but all of them failed. One of them was re-using counter_100ms.v code we already used. We wanted to check whether the time passes by 2sec using while loop, but Verilog does not allow too many iterations. Though we increased the upper bound of iteration limit, it took too long to compile. Because of this reason, we could not give any delay. That is why we couldn't see the light of inside or outside of elevator while elevator moving. If delay works properly, predicted figure is shown Fig 3. What we want to say is LEDs of inside and outside of elevator are actually work but we can't see because it is too fast.

Doing this experiment, we could feel used to using the Verilog and improving our skills.