5조 프로젝트 발표

Up Down 게임 머신

2018170914 손명준
2018170920 유하영

목차

CONTENTS

프로젝트 목적

01

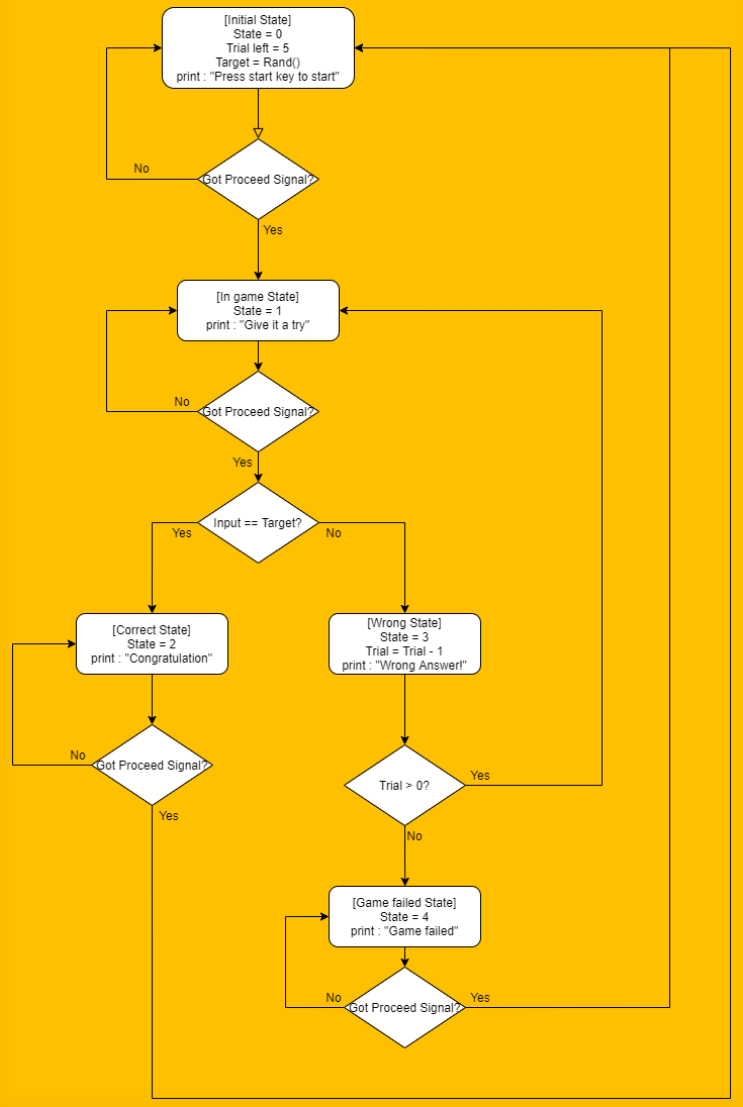
프로젝트 개요

Up Down 게임 규칙처럼 동작하는 Machine을 설계하고 구현하기

01

프로젝트 개요

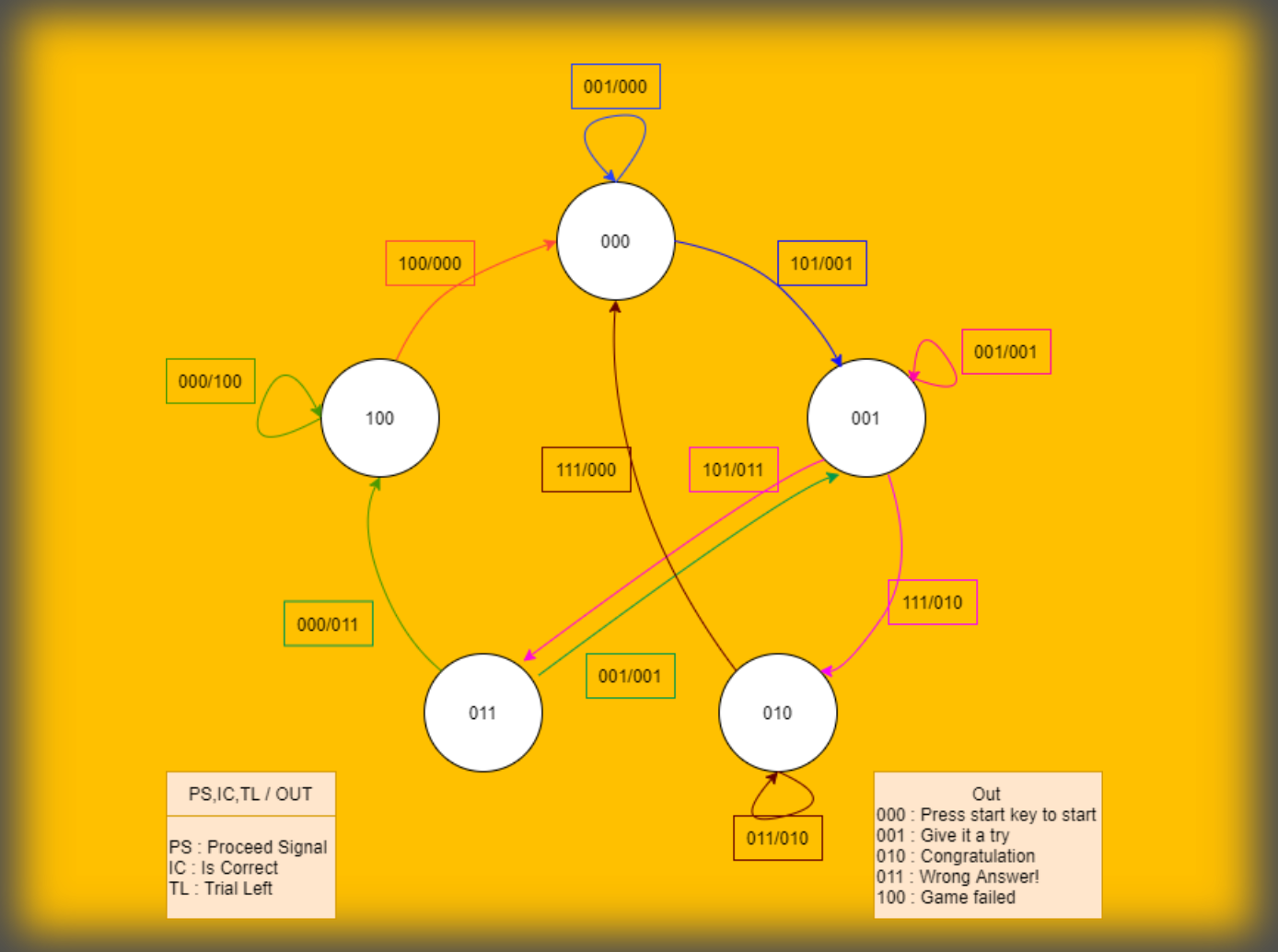
Flow chart



01

프로젝트 개요

State diagram



02

코드 설명

코드 설명

Code explanation

① main 모듈

② 입출력 모듈

③ 내부 처리 모듈

Main 모듈

02

코드 설명

```
module main(
    rst, clk, keypad_in, seg_display, array_en, LCD_E,LCD_RS,LCD_RW, LCD_DATA
);

    parameter   g_init = 3'b000,
                g_ingame = 3'b001,
                g_gameend = 3'b010,
                g_wrong = 3'b011;

    // 7segment input
    input rst,clk;
    input [11:0]keypad_in;

    // 7segment output
    output [6:0]seg_display;
    output [7:0]array_en;

    // textlcd output
    output LCD_E,LCD_RS,LCD_RW;
    output [7:0]LCD_DATA;

    wire [11:0]scan_out;
    wire valid;
    wire [7:0]r0,r1;
    wire [7:0]seg_0,seg_1;
    wire en;

    reg [2:0] output_command; // 000 : correct!,
                              // 001 : game failed!,
                              // 010 : up,
                              // 011 : down,
                              // 100 : retry?
                              // 101 : game start!
                              // 110 : enter any number
    reg [2:0] state;
    wire [31:0] input_dec;
    wire [31:0] random_number;

    reg[31:0] target;

    integer trial_left;
```

현재 State 정보를 나타내는 parameter

7 segement를 control 하기 위한 input

7 segement를 control 하기 위한 output

lcd를 control 하기 위한 output

내부 연산을 위해 필요한 변수들

En : *버튼의 입력에 대한 변수
State : 현재 state를 저장하는 변수
Input_dec : keypad로 입력받은 수를 10진수로 바꿔서 저장한 변수
Random_number : random_generator 모듈에서 생성된 random 숫자
Target : 해당 game에서 플레이어가 찾아야 할 숫자
trial_left : 남은 시도 횟수

Main 모듈

02

코드 설명

```
keypad_scan KS(rst, clk, keypad_in, scan_out, valid);
display DP(rst, clk, scan_out, valid, r0,r1, en);
register RG1(clk, rst, r0, seg_0);
register RG2(clk, rst, r1, seg_1);
seg_controller SC(clk, rst, seg_0, seg_1, seg_display, array_en);
textlcd tlcd(rst, clk, output_command, LCD_E, LCD_RS, LCD_RW, LCD_DATA, trial_left);
random_generator random_gen(clk, random_number);

reg_to_dec rtc(seg_0, seg_1, input_dec); // register에 저장된 7seg 신호를 10진법으로 바꾸는 모듈
```

Main 모듈과 다른 입출력, 내부 처리 모듈을 연결하는 부분

Main 모듈

02

코드 요약

```
always @(posedge clk) begin
    case(state)  // * 버튼을 누르지 않아도 자동으로 state가 넘어가야 하는 state
        g_init:begin
            output_command = 3'b110; // lcd에 enter any number 출력
            target = random_number; // 랜덤 숫자 생성
            state = g_ingame; // ingame state로 state 변경
                trial_left = 5; //시도 수 5 로 제한
        end
        g_wrong:begin
            state = g_ingame; // ingame state로 state 변경
        end
    endcase
```

입력이 필요 없는 state에 대해 처리하는 부분

init state에서 "enter any number"를 출력하고 target number를 설정
Trial left 초기화

Wrong state에서 다시 ingame state로 이동

Main 모듈

02

코드 설명

```
if(en == 1'b1)begin
    // * 버튼을 눌러야 state가 넘어가야 하는 state -> 입력이 필요한 st
    // ingame state의 경우 정답에 대한 입력이 필요
    // gameend state의 경우 재시작 여부에 대한 입력이 필요
    case(state)
    g_ingame:begin
        if(input_dec == target) begin // 정답일 때
            output_command = 3'b000; // lcd에 correct 출력
            state = g_gameend;
        end
        else begin // 오답일 때
            trial_left = trial_left - 1;
            if(trial_left == 0) begin
                state = g_gameend;
                output_command = 3'b100; // lcd에 retry? 출력
            end
            else if(input_dec > target) begin
                state = g_wrong;
                output_command = 3'b011; // lcd에 down 출력
            end
            else if(input_dec < target) begin
                state = g_wrong;
                output_command = 3'b010; // lcd에 up 출력
            end
            else begin
            $display("something is strange..");
            end
        end
    end

    g_gameend:begin
            output_command = 3'b100;
        if(input_dec == 0) begin // 다시 안하는 경우
        end
        else if(input_dec == 1) begin // 다시 하는 경우
            state = g_init;
            output_command = 3'b101; // lcd에 game start! 출력
        end
    end
endcase
```

입력이 필요한 state에 대해 처리하는 부분

Ingame state에서
입력받은 숫자와 targe을 비교하여 정답 여부 확인

정답인 경우
gameend state로 이동,
오답인 경우
trial left가 0보다 크면
up 또는 down 출력
g_wrong state로 이동,
0이면
gameend state로 이동

Gameend state에서
retry 여부를 물었을 때 01이 입력되면
다시 init state로 전환

입출력 모듈
Textlcd.v

02

코드 설명

```
line1:
begin
    LCD_RW=1'b0;
    case(output_command)
    3'b000 : begin // correct!
        case(CNT)
            0:      begin LCD_RS = 1'b0;      LCD_DATA=8'b10000000;/*address*/ end
            1:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01000011;/*C*/ end // U
            2:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01101111;/*o*/ end // P
            3:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01110010;/*r*/ end
            4:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01110010;/*r*/ end
            5:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01100101;/*e*/ end
            6:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01100011;/*c*/ end
            7:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01110100;/*t*/ end
            8:      begin LCD_RS = 1'b1;      LCD_DATA=8'b00100001;/*!*/ end
            9:      begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/**/ end
            10:     begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/**/ end
            11:     begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/**/ end
            12:     begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/**/ end
            13:     begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/**/ end
            14:     begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/**/ end
            15:     begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/**/ end
            16:     begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/**/ end
            default:begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/**/ end
        endcase
    end
    3'b100 : begin // game failed!
        case(CNT)
            0:      begin LCD_RS = 1'b0;      LCD_DATA=8'b10000000;/*address*/ end
            1:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01100111;/*g*/ end
```

```
line2:      begin
    LCD_RW=1'b0;
    if(output_command == 3'b100 || output_command == 3'b000) begin
    case(CNT)
        0:      begin LCD_RS = 1'b0;      LCD_DATA=8'b11000000;/*address*/ end
        1:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01110010;/*r*/ end
        2:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01100101;/*e*/ end
        3:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01110100;/*t*/ end
        4:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01110010;/*r*/ end
        5:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01111001;/*y*/ end
        6:      begin LCD_RS = 1'b1;      LCD_DATA=8'b00111111;/*?*/ end
        7:      begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/**/ end
        8:      begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/**/ end
        9:      begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/**/ end
        10:     begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/**/ end
        11:     begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/**/ end
        12:     begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/**/ end
        13:     begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/**/ end
        14:     begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/**/ end
        15:     begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/**/ end
        16:     begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/**/ end
        default:begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/**/ end
    endcase
    end
    else begin
        case(CNT)
            0:      begin LCD_RS=1'b0;        LCD_DATA=8'b11000000;/*address*/ end
            1:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01110100;/*t*/ end
            2:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01110010;/*r*/ end
            3:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01101001;/*i*/ end
            4:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01100001;/*a*/ end
            5:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01101100;/*l*/ end
            6:      begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/* */ end
            7:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01101100;/*l*/ end
            8:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01100101;/*e*/ end
            9:      begin LCD_RS = 1'b1;      LCD_DATA=8'b01100110;/*f*/ end
            10:     begin LCD_RS = 1'b1;      LCD_DATA=8'b01110100;/*t*/ end
            11:     begin LCD_RS = 1'b1;      LCD_DATA=8'b00100000;/* */ end
                /* 숫자 */
            12:     begin LCD_RS = 1'b1;
                        case(trial_left)
```

Line 1에는 up, down, correct, game failed를 출력
line 2에는 trial left 또는 retry?를 출력

입출력 모듈
display.v

02

코드 요약

```
else begin
  en <= 0; // 매 클럭마다 en을 초기화한다 -> *키가 눌린 시점에서만 en이 1이 되도록 하기 위함
  if (valid) begin
    case(scan_data)
    12'b000000000001 : begin w <= 7'b0110000; end // 1
    12'b000000000010 : begin w <= 7'b1101101; end // 2
    12'b000000000100 : begin w <= 7'b1111001; end // 3
    12'b000000001000 : begin w <= 7'b0110011; end // 4
    12'b000000010000 : begin w <= 7'b1011011; end // 5
    12'b000000100000 : begin w <= 7'b1011111; end // 6
    12'b000001000000 : begin w <= 7'b1110010; end // 7
    12'b000010000000 : begin w <= 7'b1111111; end // 8
    12'b000100000000 : begin w <= 7'b1111011; end // 9
    12'b001000000000 : begin en <= 1; end         // * 다음 상태로 넘어가는 특수키
    12'b010000000000 : begin w <= 7'b1111110; end // 0
    12'b100000000000 : begin r9 <= r9 + 1; w <= 7'b1111110; end // #
    endcase
```

*버튼이 눌리면 입력 완료 신호 생성
#버튼이 눌리면 옆자리로 커서 이동

내부 처리 모듈
Reg_to_dec.v

02

프로젝트 개요

```
module reg_to_dec(
    seg_0, seg_1, dec_out
);

input [7:0] seg_0;
input [7:0] seg_1;

output reg [31:0] dec_out;

integer ten, one;

always @ (seg_0 or seg_1) begin
    case(seg_0)
        7'b0110000 : ten = 1;
        7'b1101101 : ten = 2;
        7'b1111001 : ten = 3;
        7'b0110011 : ten = 4;
        7'b1011011 : ten = 5;
        7'b1011111 : ten = 6;
        7'b1110010 : ten = 7;
        7'b1111111 : ten = 8;
        7'b1111011 : ten = 9;
        7'b1111110 : ten = 0;
        default : ten = 0;
    endcase
    case(seg_1)
        7'b0110000 : one = 1;
        7'b1101101 : one = 2;
        7'b1111001 : one = 3;
        7'b0110011 : one = 4;
        7'b1011011 : one = 5;
        7'b1011111 : one = 6;
        7'b1110010 : one = 7;
        7'b1111111 : one = 8;
        7'b1111011 : one = 9;
        7'b1111110 : one = 0;
        default : one = 0;
    endcase
    dec_out = ten*10+one;
end
endmodule
```

7 segment 출력 신호 형태로 저장된 입력값을 decimal로 변환하는 모듈
Segment 0는 10의 자릿수, segment 1은 1의 자릿수로 계산하여 2자리 10진수를 반환

내부 처리 모듈
Random_generator.v

02

코드 작성

```
module random_generator(clk, out);

input clk;
output reg [31:0] out;

integer cnt;

initial cnt = 0;

always @ (posedge clk) begin
    cnt = cnt + 1;
    if(cnt == 50) cnt = 0;
    out = cnt;
end
endmodule
```

경과된 clock을 이용해 target 숫자를 생성하는 모듈
단순한 로직으로 유사 난수를 생성할 수 있음.
게임의 편의를 위해 생성되는 숫자의 상한을 50으로 설정
첫 번째 게임에서는 targe이 0으로 고정되는 단점이 있음.

03

제목을 입력하세요

시연 영상

Examination video

① 정답을 맞힌 경우

② 시도 횟수를 초과한 경우

감사합니다.

THANK YOU.

2018170914 손명준
2018170920 유하영