디지털시스템설계 FINAL PROJECT

20210498 김동한

20220741 구동현

20220923 이윤혁

주제 : 4~6자리 비밀번호 도어락 제작

목적 : 여러 state를 가지는 FSM 을 설계함을 통해 sequential logic(counter, flip-flop 등)에 대한 이해를 높이고 simplification 을 연습해본다.

I. 이론적 배경 :

1) Register and RAM

레지스터와 램은 컴퓨터의 주요한 구성 요소로서, 데이터를 저장하고 처리하는데에 사용된다.

➔ 레지스터 : flip flop의 group들로 구성되며 프로세서에서 일시적인 저장소로 사용된다. 또한 매우 빠른 속도로 데이터에 액세스할 수 있게 해주며 산술 레지스터, 주소 레지스터 등이 존재한다.

➔ 램 : 컴퓨터에서 데이터를 일시적으로 저장하는데 사용되는 주기억 장치로, 실행중인 프로그램, 데이터를 저장한다. SRAM, DRAM 등으로 구분할 수 있으며 SRAM은 전원이 켜진 이상 데이터가 계속 저장되는 램이고, DRAM은 주기적으로 값을 읽고, 다시 쓰는 작업을 반복하는 ram 이다.

2) counter

계수기는 숫자를 세는데 사용되는 장치로, 클럭 신호에 의해 동작하며 특정 순서로 숫자를 증가시키거나 감소시킨다. 타이머, 주파수 계수기 등 다양한 분야에서 유용하게 사용되는 순차 논리 회로이다.

II. 구현과 관련된 수업 내용 및 배경 지식

키패드의 output 7개(4 row, 3 column) 을 BCD encoder 의 input으로 입력 -> 4 bit BCD로 변환한다. 이때 버튼이 2개 이상 눌리면 동작을 제대로 하지 못하는 encoder의 문제를 해결하기 위해 priority encoder 형태로 설계하였다.

(우선 순위 : high <- 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 -> low)

한편 키패드의 버튼을 입력할 때마다 counter의 clk 으로 주입하여 counter가 현재까지 입력된 숫자의 개수를 나타내도록 한다.

또한 select input 으로 위 counter의 output을, data input으로 BCD encoder의 output을, 6개의 output으로 각 4-bit register 6개를 연결하여 1:6 decoder 를 구현한다. (즉 맨 위 4-bit reg 에는 첫 번째 숫자가, 두 번째 4-bit reg에는 두 번째 숫자가 저장 ...)

이후 * 이 눌릴 때 각 자리의 숫자와 자리수를 의미하는 reg와 이에 대응되는 answer reg 들을 bitwise XOR 연산을 수행하여 답이 일치하는지 확인한다.

비밀번호를 재설정 할 때는 번호를 입력받은 register 의 값과 자릿수를 의미하는 reg의 값을 각 자리에 대응하는 answer register로 shifting 하여 재설정한다.

III. 실험 준비 과정 : (state transition diagram 및 state machine 설계)

각 state 간 이동에 쓰이는 input bit 5개 –

1. #(on-off toggle 버튼)을 눌렀는지 여부
2. 4~6개의 번호가 눌린 후 * 을 눌렀는지 여부
3. 비밀번호 재설정 버튼이 눌렸는지 여부
4. 눌린 번호가 정답과 같은지 여부
5. 초기화 버튼이 눌렸는지 여부

각 state(3bit)

OFF(000)-> 도어락의 전원이 꺼진 상태

ON(001) -> 도어락의 전원이 켜진 상태, 7-seg에 3 출력(기회가 3회 남았음을 의미)

WRONG1(010) -> 비밀번호를 1회 틀린 상태, 2를 출력

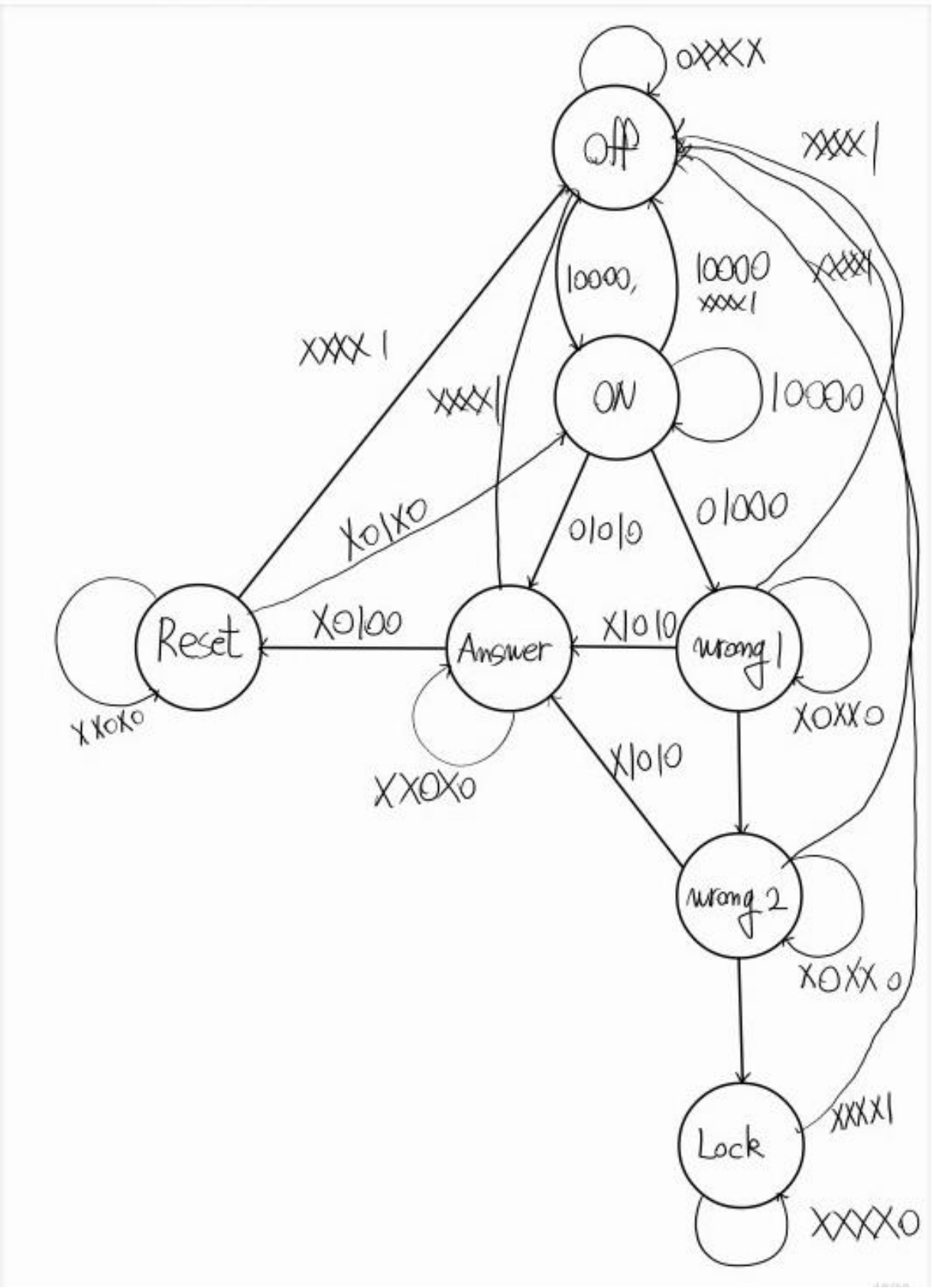
WRONG2(011) -> 비밀번호를 2회 틀린 상태, 1를 출력

ANSWER(100) -> 도어락이 열린 상태, OPEn 을 출력

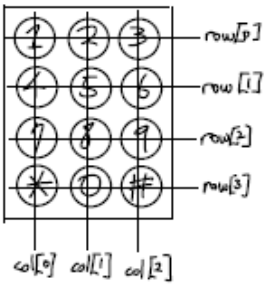
RESET(101) -> 비밀번호를 재설정하고 있는 상태, Chng 를 출력

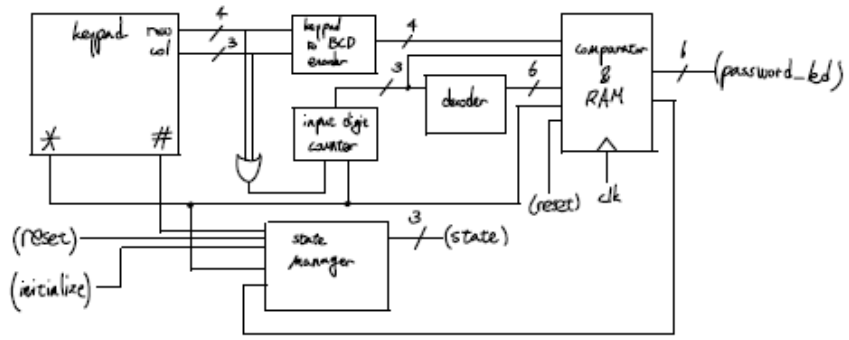
LOCK(111) -> 3회 틀려서 도어락이 아예 잠긴 상태(초기화 버튼으로 off로 변경 가능)

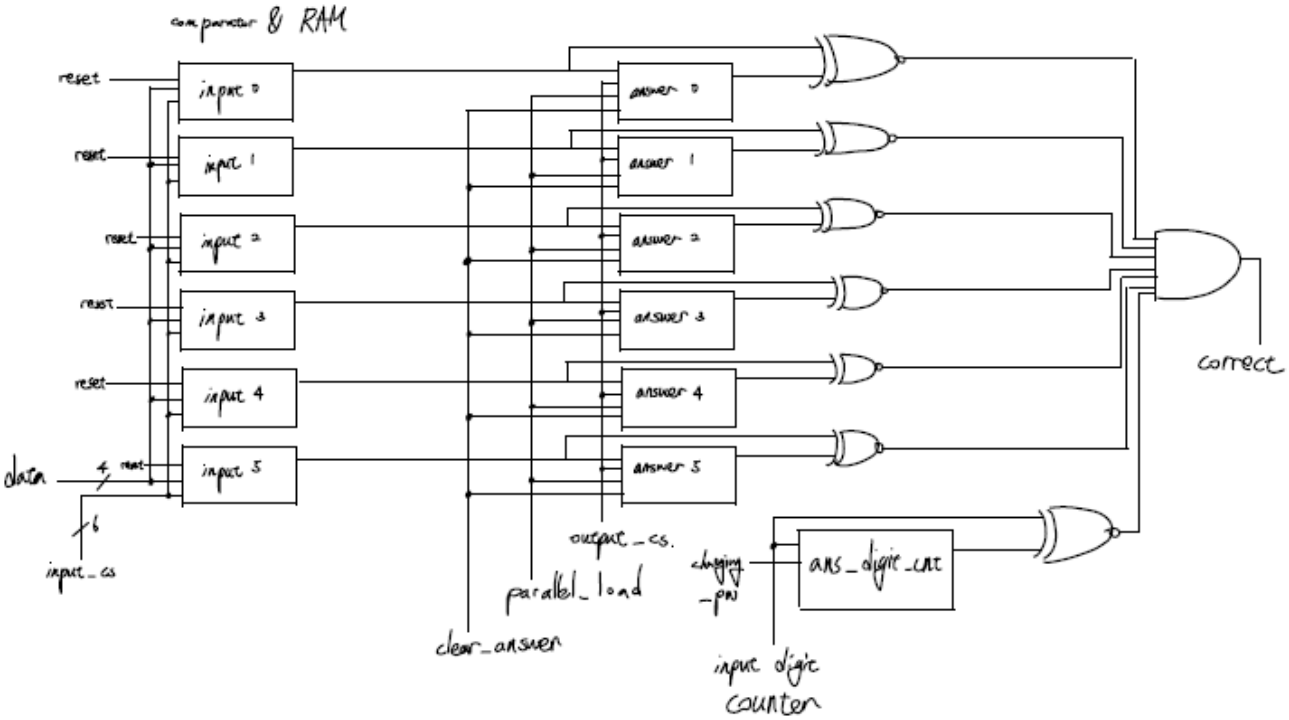
<state diagram>



<회로도>


row[0]
row[1]
row[2]
row[3]
col[0]
col[1]
col[2]


keypad
row
col
keypad to BCD encoder
input digit counter
decoder
comparator & RAM
(password_kd)
(reset)
clk
(reset)
(initialize)
state manager
(state)


comparator & RAM
reset
input 0
input 1
input 2
input 3
input 4
input 5
data
input_cs
answer 0
answer 1
answer 2
answer 3
answer 4
answer 5
output_cs.
parallel_load
clear_answer
changing_pw
ans_digit_cnt
input digit counter
correct

IV. 전체적인 동작 설명 및 입출력

(언제든지 초기화 버튼을 누르면 모든 register 의 값이 0으로 설정되며 Off State가 된다.)

4x3 키패드를 input으로 사용한다.

초기상태에서 #을 누르면 번호를 입력받을 수 있는 상태(State : On) 가 되며 FPGA의 7 segment 가 3을 나타낸다. (기회가 3회 남았음을 의미) 이 후 비밀번호 4~6자리를 누르고 *를 누르면–

(이 때 번호가 1개 눌릴때마다 FPGA의 LED 가 1개씩 켜진다.)

-> 정답 : 금고 문이 열린다.(State : Ans) (7segment 가 OPEn 을 표현)

-> 오답 : 다시 번호를 입력받는다. (State: Wng1) (7segment 가 2를 표현)

-> 정답 : 금고 문이 열린다. (State: Ans) (green LED on)

-> 오답 : 다시 번호를 입력받는다. (State: Wng2) (7 segment가 1을 표현)

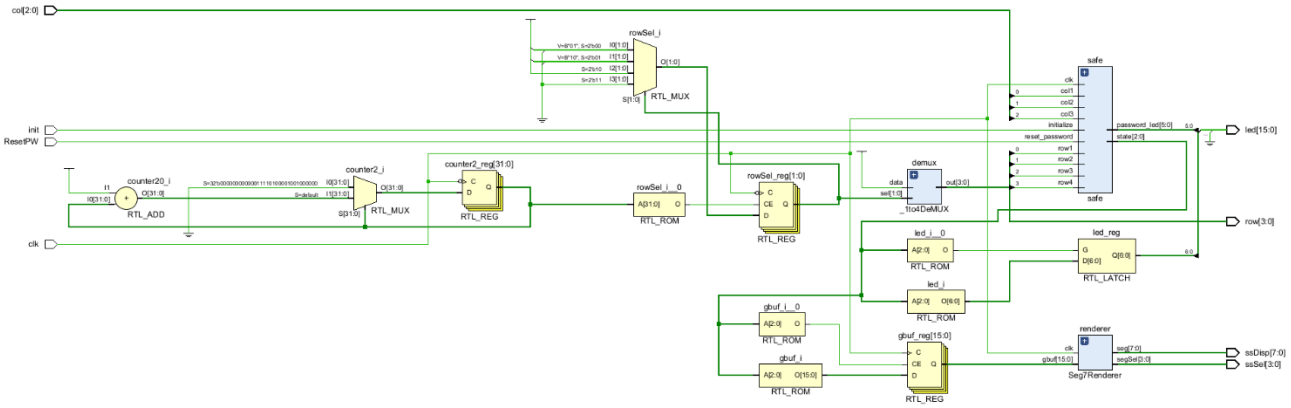
-> 정답 : 금고 문이 열린다. (State: Ans) (green LED on)

-> 오답 : 다시 번호를 입력받지 않으며 더 이상 키패드가 작동하지 않는다. (State : Lock) (7segment가 FAIL 을 표현)

한편 (State : Ans) 상태에서 키패드 뒷변의 비밀번호 재설정 버튼을 클릭하면 새로운 정답 번호를 입력받는 상태가 된다. (이때도 번호를 입력 후 *표를 눌러야 한다.) (State: Reset) (7segment 가 Chng 을 표현)

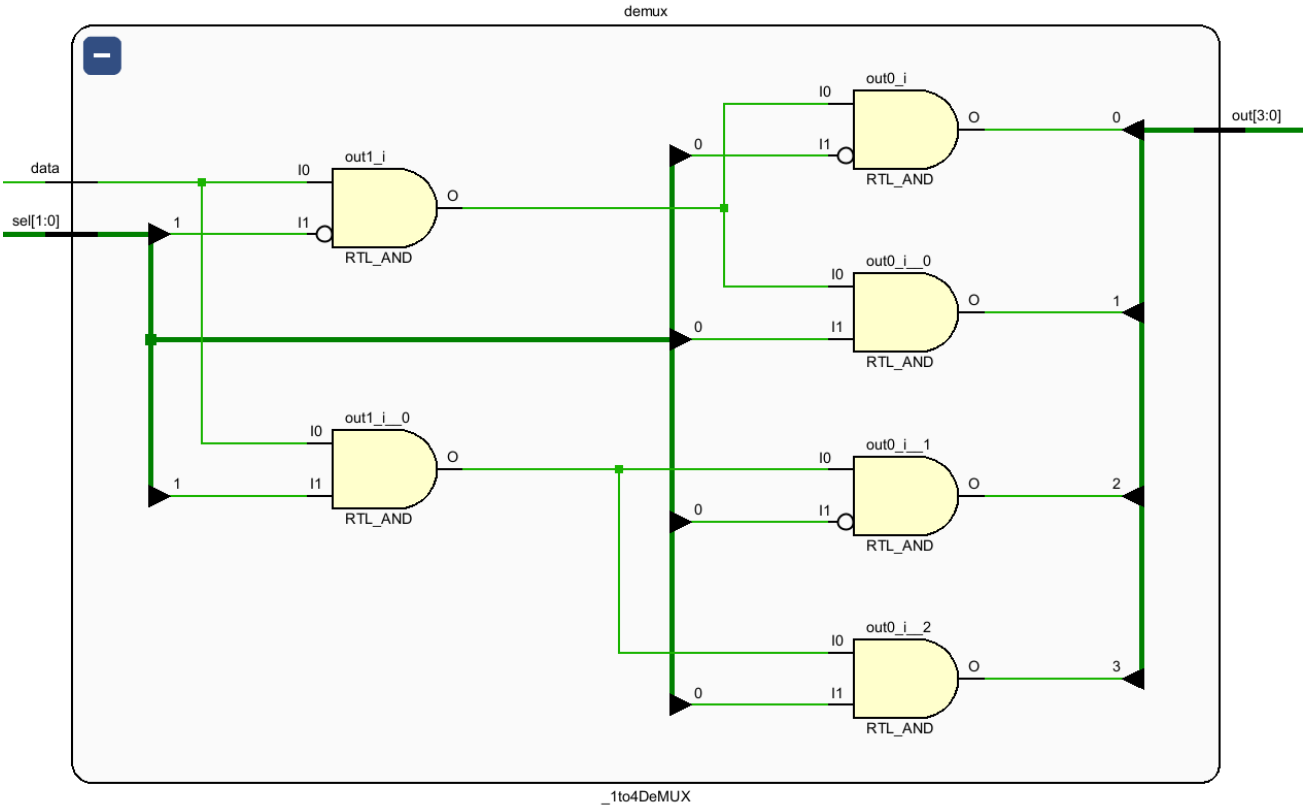
V. 결과

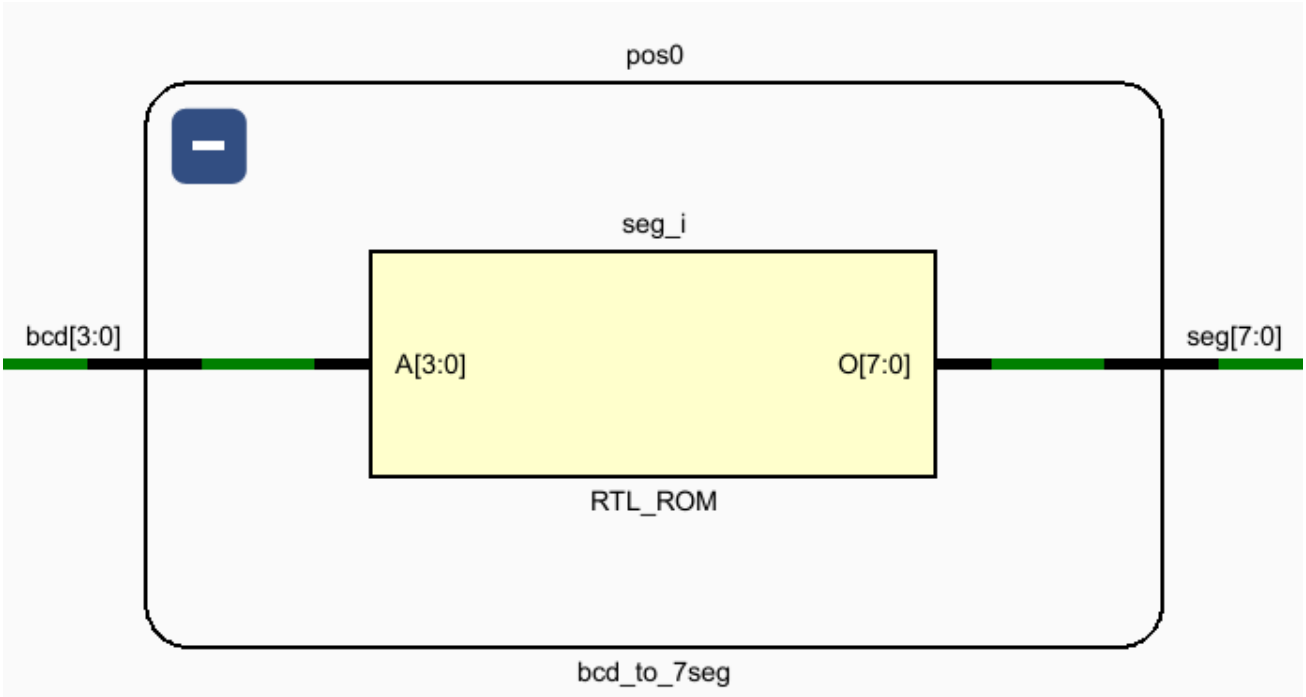
Schematic -

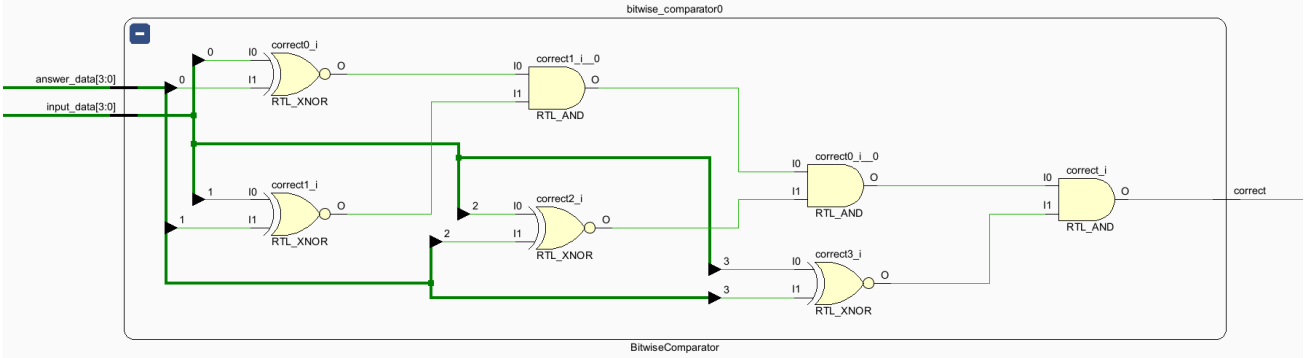


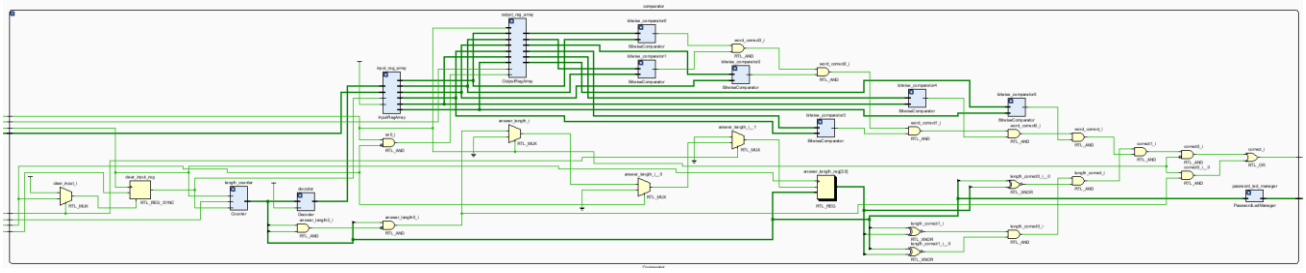
<main.v>

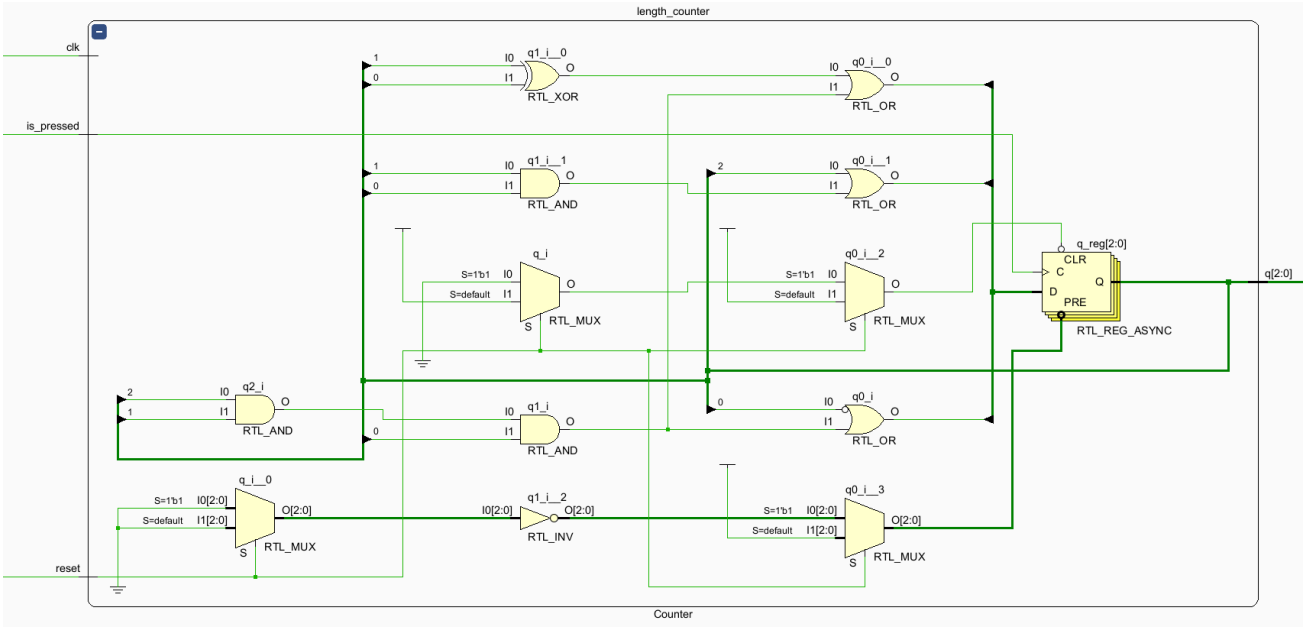
이하 schematic 은 main schematic 속 module 들에 대한 세부 회로도이다.

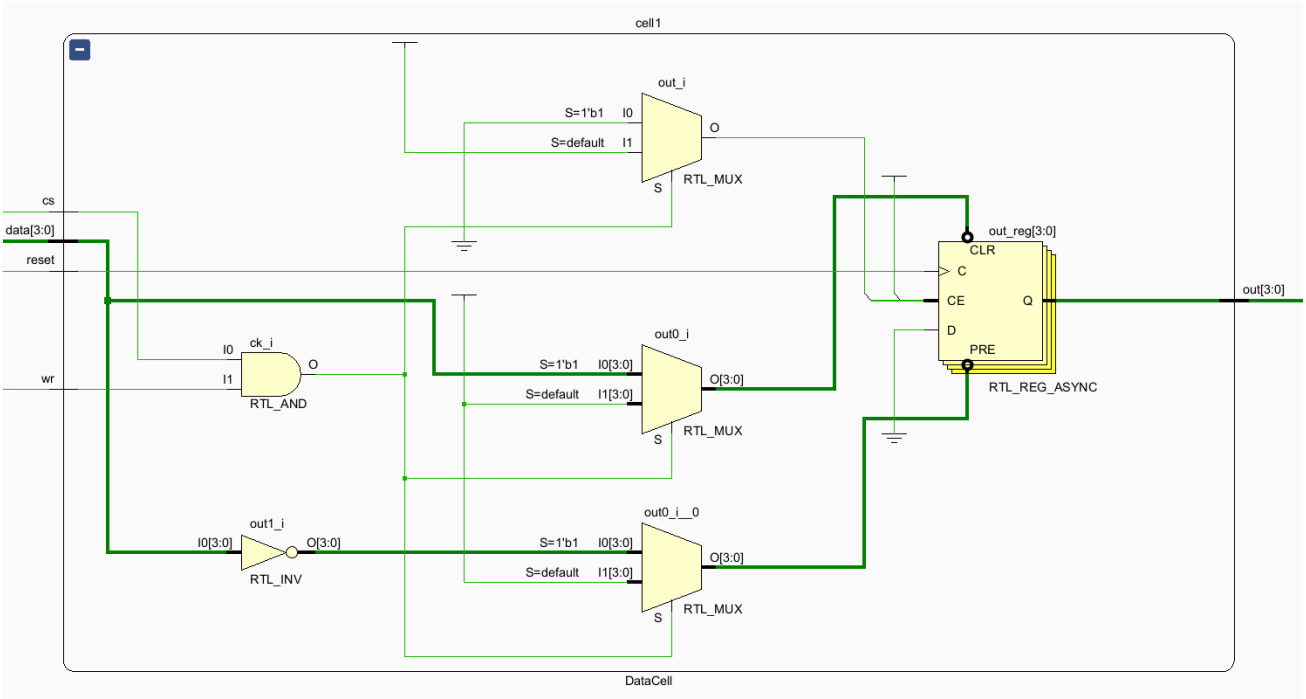


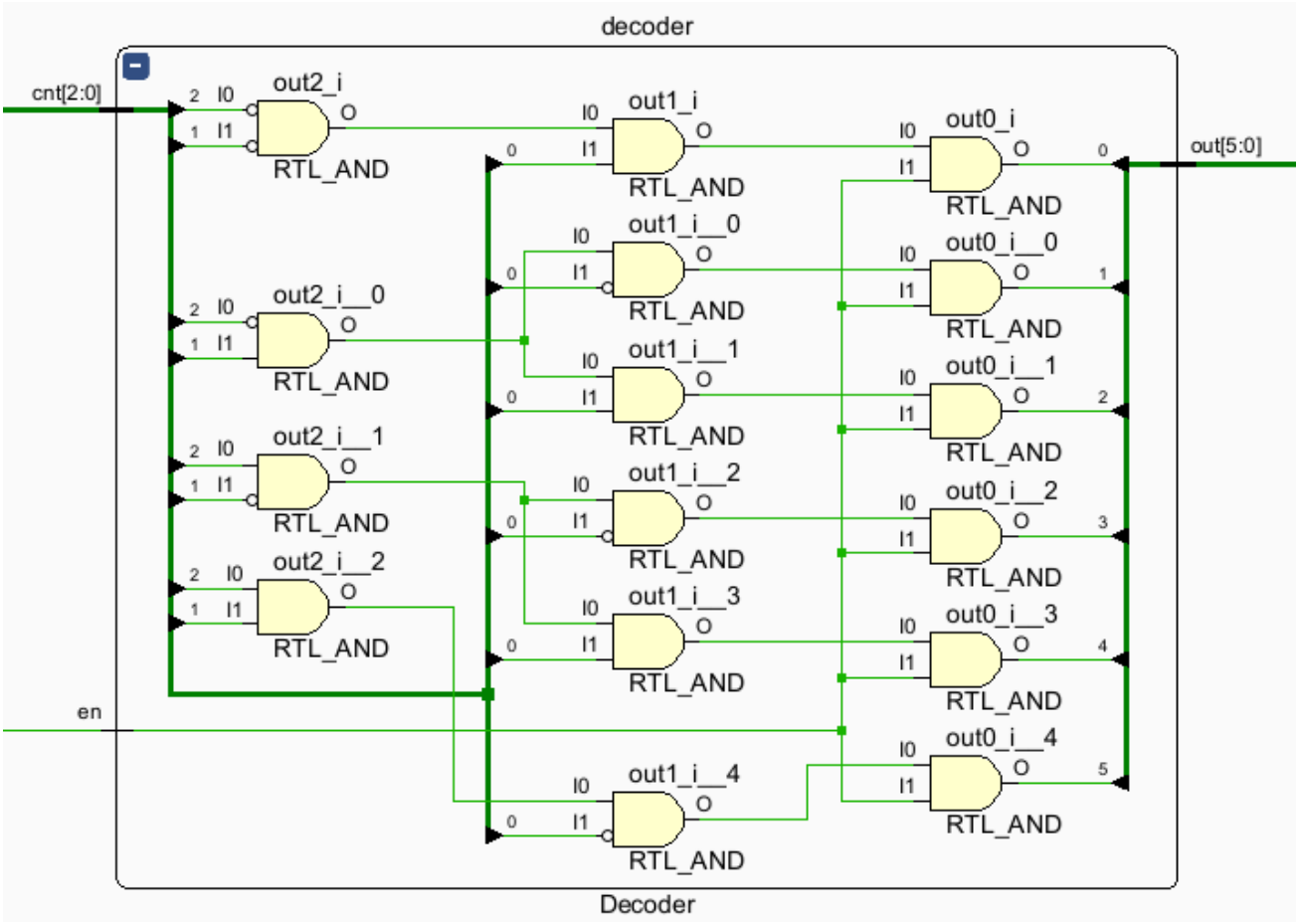

pos0
seg_i
bcd[3:0]
A[3:0]
O[7:0]
seg[7:0]
RTL_ROM
bcd_to_7seg

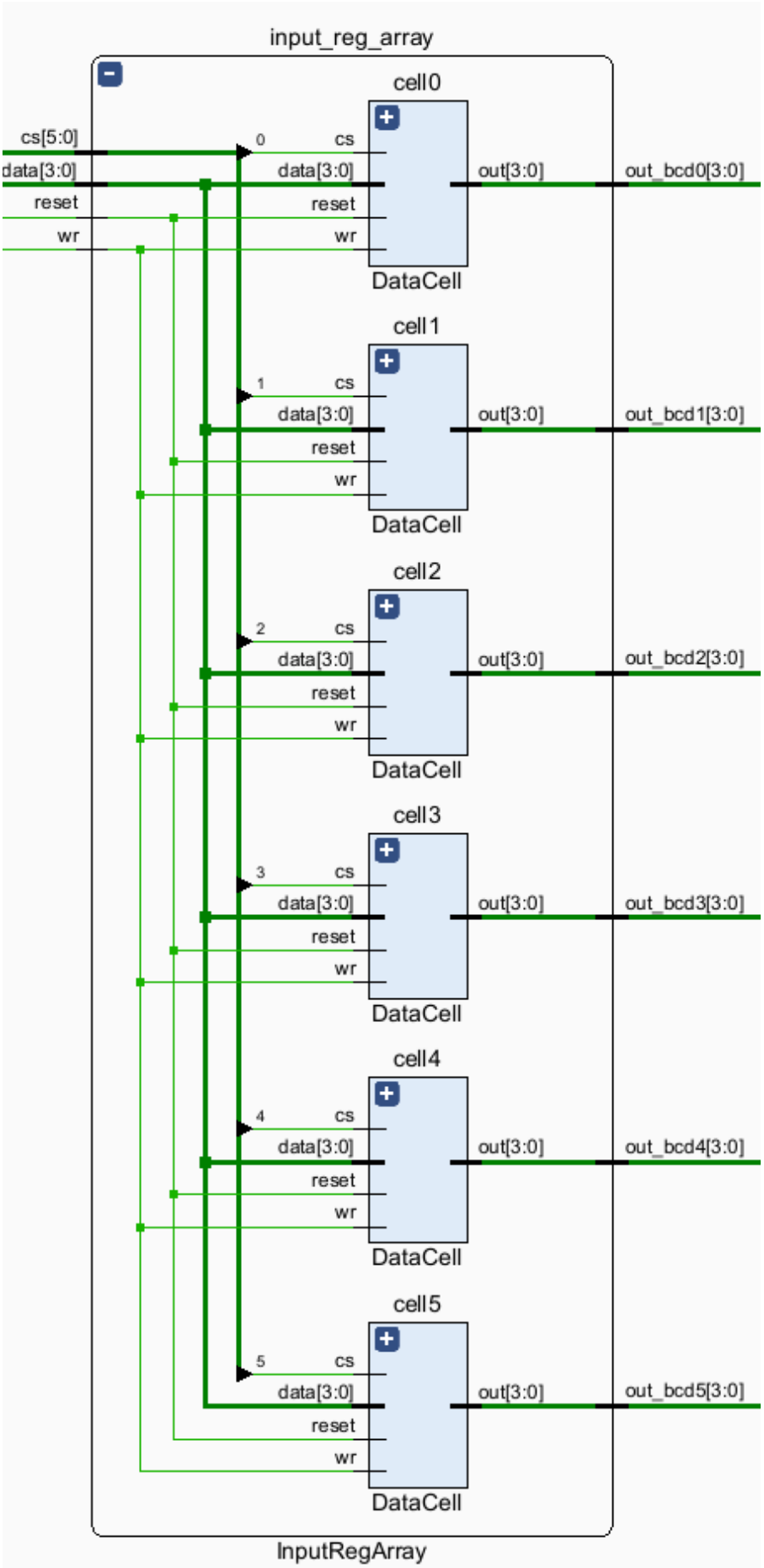

bitwise_comparator0
answer_data[3:0]
input_data[3:0]
correct0_i
correct1_i
correct2_i
correct3_i
correct0_i__0
correct1_i__0
correct_i
RTL_XNOR
RTL_AND
correct
BitwiseComparator

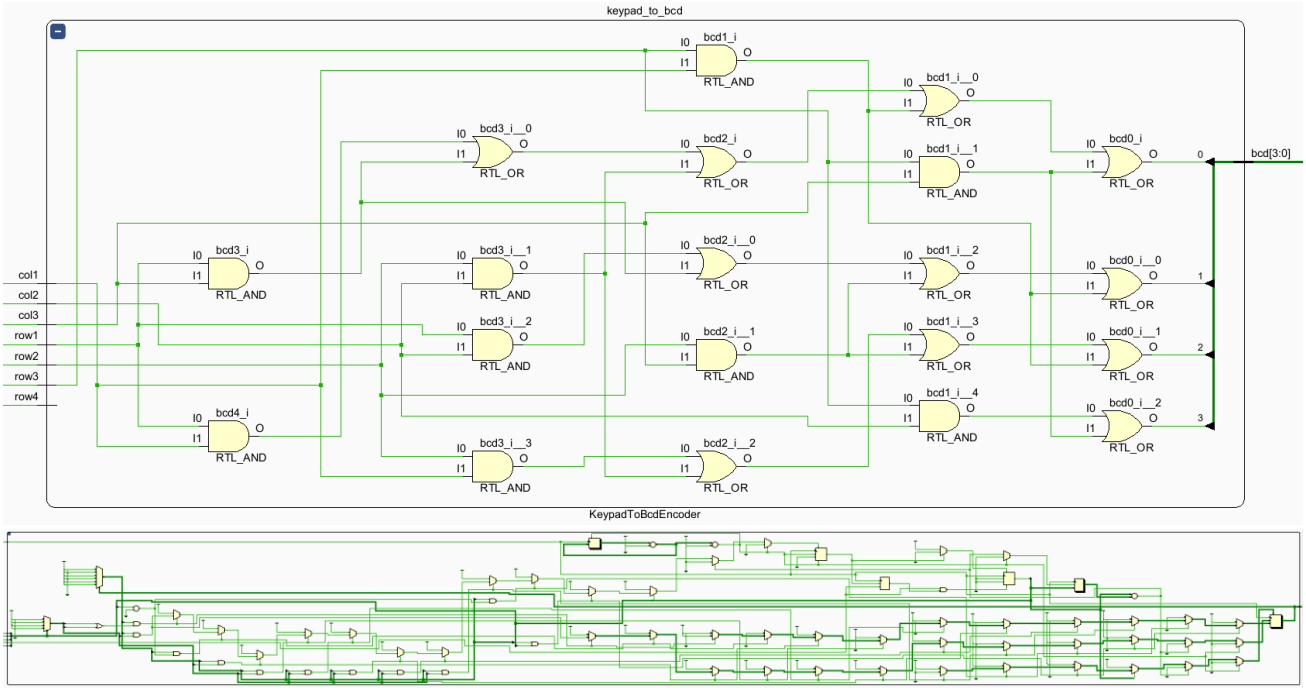


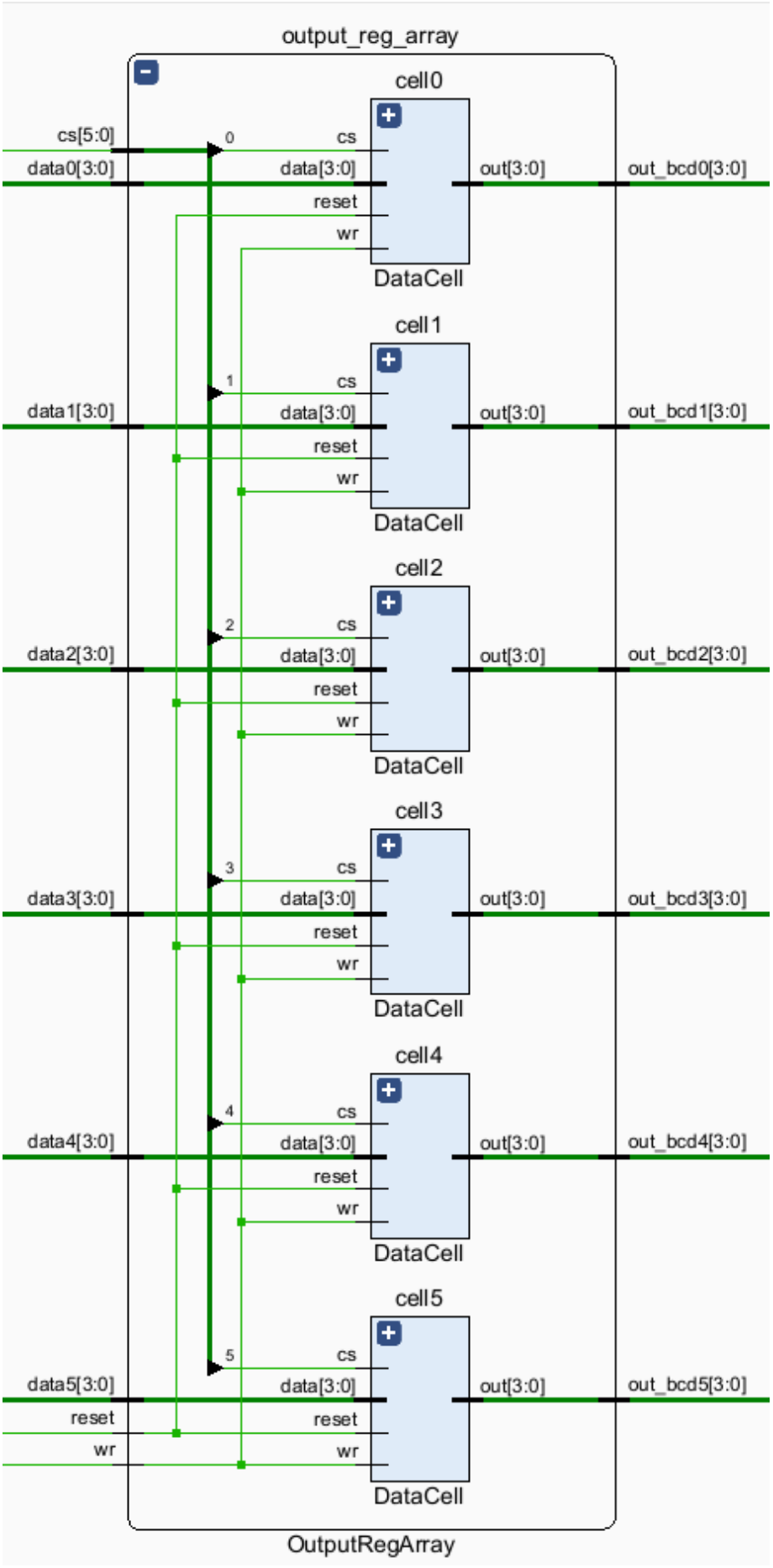

length_counter
clk
is_pressed
reset
q1_i__0
RTL_XOR
q0_i__0
RTL_OR
q1_i__1
RTL_AND
q0_i__1
RTL_OR
q_i
RTL_MUX
q0_i__2
RTL_MUX
q_reg[2:0]
CLR
C
D
Q
PRE
RTL_REG_ASYNC
q[2:0]
q2_i
RTL_AND
q1_i
RTL_AND
q0_i
RTL_OR
q_i__0
RTL_MUX
q1_i__2
RTL_INV
q0_i__3
RTL_MUX
Counter

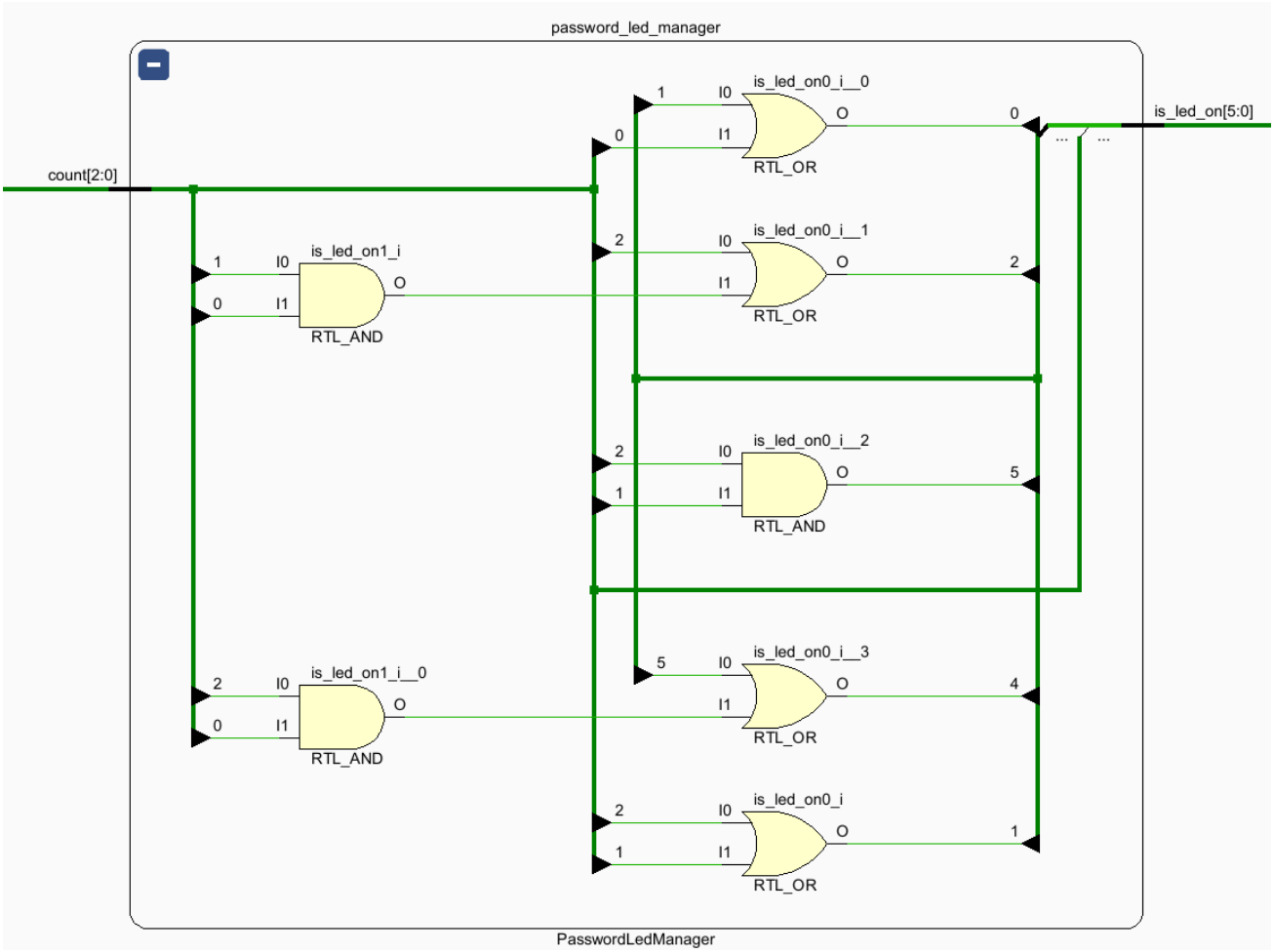

cell1
cs
data[3:0]
reset
wr
out_i
RTL_MUX
out_reg[3:0]
CLR
C
CE
D
Q
PRE
RTL_REG_ASYNC
out[3:0]
ck_i
RTL_AND
out0_i
RTL_MUX
out1_i
RTL_INV
out0_i__0
RTL_MUX
DataCell

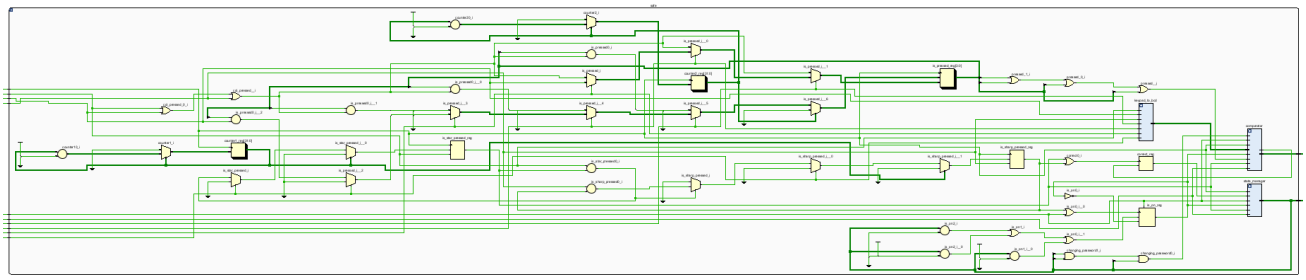

decoder
cnt[2:0]
en
out[5:0]
out2_i
out2_i__0
out2_i__1
out2_i__2
out1_i
out1_i__0
out1_i__1
out1_i__2
out1_i__3
out1_i__4
out0_i
out0_i__0
out0_i__1
out0_i__2
out0_i__3
out0_i__4
RTL_AND
Decoder

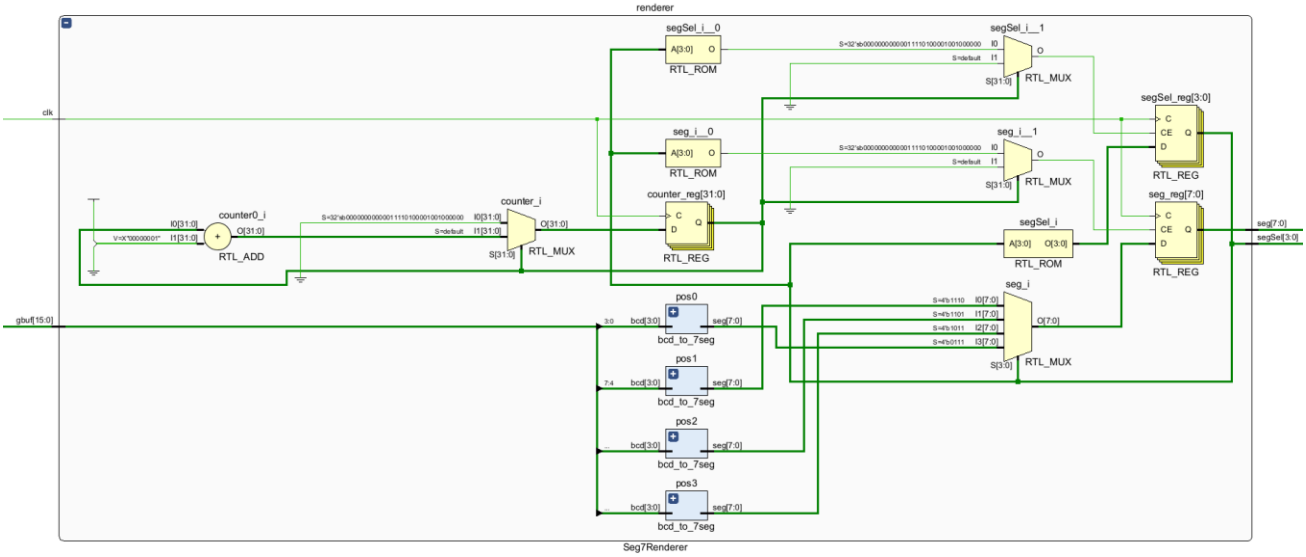

input_reg_array
cell0
cell1
cell2
cell3
cell4
cell5
cs[5:0]
data[3:0]
reset
wr
0
1
2
3
4
5
cs
data[3:0]
reset
wr
out[3:0]
out_bcd0[3:0]
out_bcd1[3:0]
out_bcd2[3:0]
out_bcd3[3:0]
out_bcd4[3:0]
out_bcd5[3:0]
DataCell
InputRegArray


keypad_to_bcd
col1
col2
col3
row1
row2
row3
row4
bcd[3:0]
bcd1_i
bcd3_i__0
bcd2_i
bcd1_i__0
bcd1_i__1
bcd0_i
bcd3_i
bcd3_i__1
bcd2_i__0
bcd1_i__2
bcd0_i__0
bcd3_i__2
bcd2_i__1
bcd1_i__3
bcd0_i__1
bcd4_i
bcd1_i__4
bcd0_i__2
bcd3_i__3
bcd2_i__2
RTL_AND
RTL_OR
I0
I1
O
0
1
2
3
KeypadToBcdEncoder


output_reg_array
cell0
cs[5:0]
0
cs
data0[3:0]
data[3:0]
out[3:0]
out_bcd0[3:0]
reset
wr
DataCell
cell1
1
cs
data1[3:0]
data[3:0]
out[3:0]
out_bcd1[3:0]
reset
wr
DataCell
cell2
2
cs
data2[3:0]
data[3:0]
out[3:0]
out_bcd2[3:0]
reset
wr
DataCell
cell3
3
cs
data3[3:0]
data[3:0]
out[3:0]
out_bcd3[3:0]
reset
wr
DataCell
cell4
4
cs
data4[3:0]
data[3:0]
out[3:0]
out_bcd4[3:0]
reset
wr
DataCell
cell5
5
cs
data5[3:0]
data[3:0]
out[3:0]
out_bcd5[3:0]
reset
reset
wr
wr
DataCell
OutputRegArray

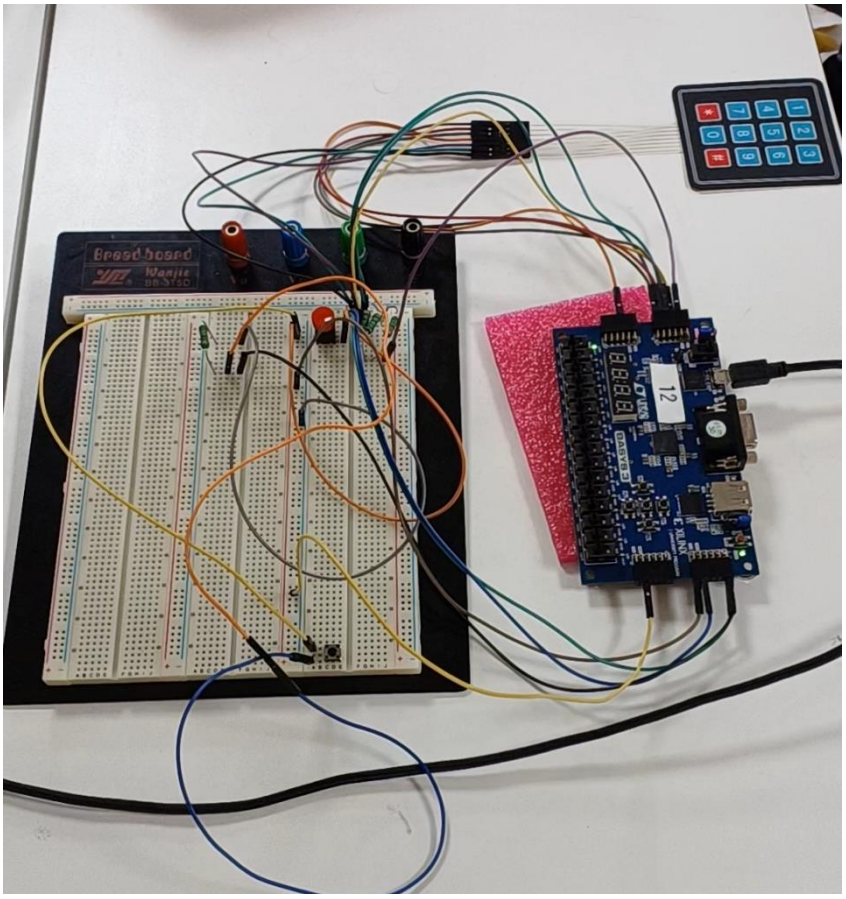

password_led_manager
count[2:0]
is_led_on[5:0]
is_led_on0_i__0
is_led_on0_i__1
is_led_on0_i__2
is_led_on0_i__3
is_led_on0_i
is_led_on1_i
is_led_on1_i__0
RTL_OR
RTL_AND
I0
I1
O
PasswordLedManager



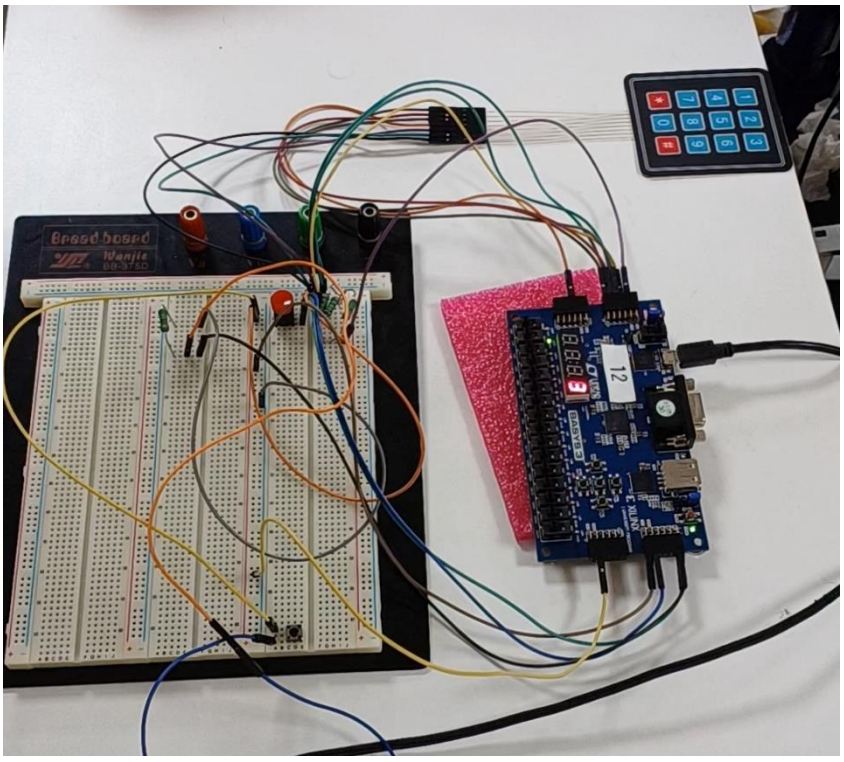

renderer
segSel_i__0
segSel_i__1
seg_i__0
seg_i__1
counter_reg[31:0]
segSel_reg[3:0]
seg_reg[7:0]
segSel_i
seg_i
counter0_i
counter_i
pos0
pos1
pos2
pos3
RTL_ROM
RTL_MUX
RTL_REG
RTL_ADD
bcd_to_7seg
clk
gbuf[15:0]
seg[7:0]
segSel[3:0]
Seg7Renderer

시연 사진 -

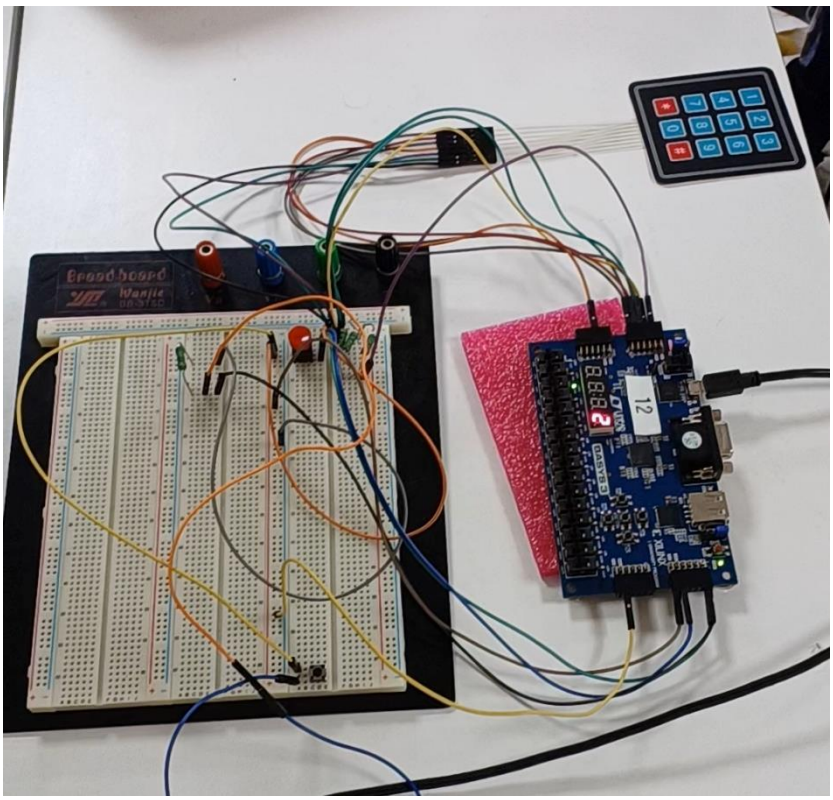
<OFF STATE>



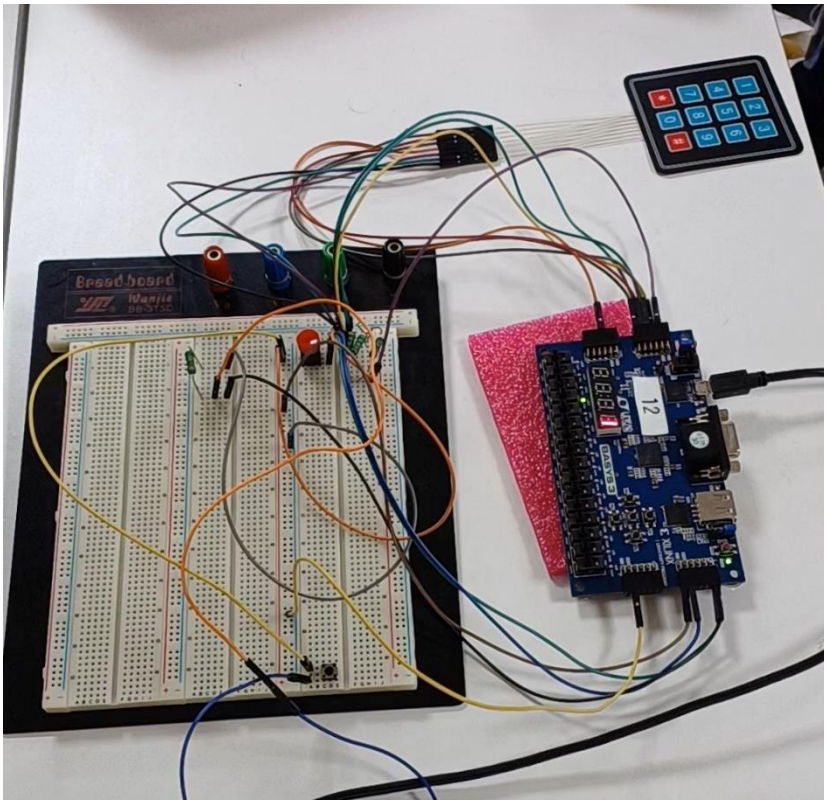
<ON STATE>



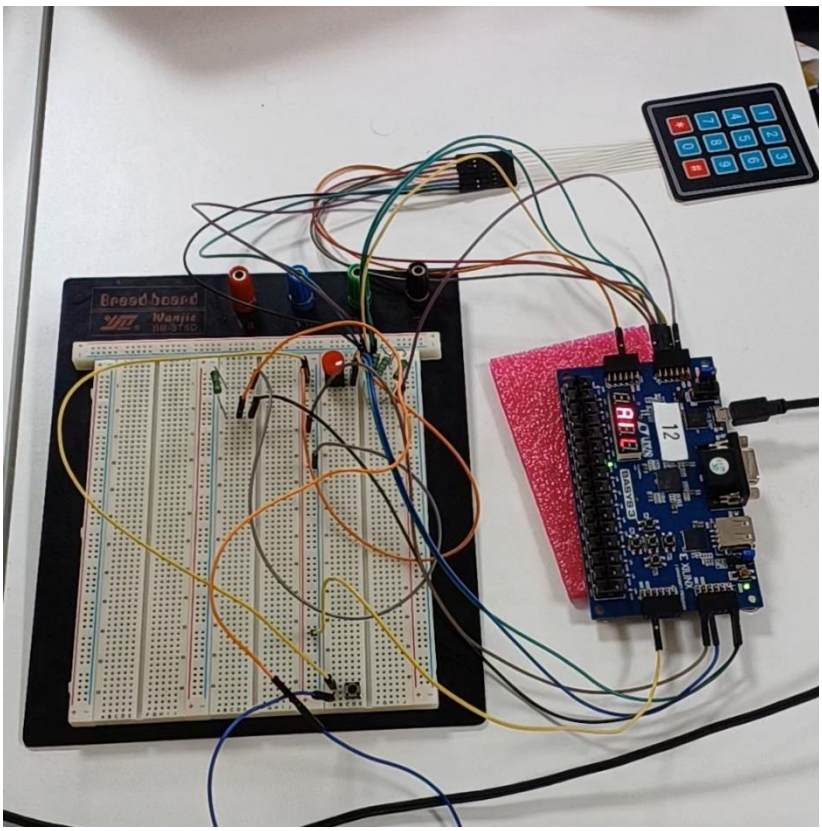
<WRONG1 STATE>



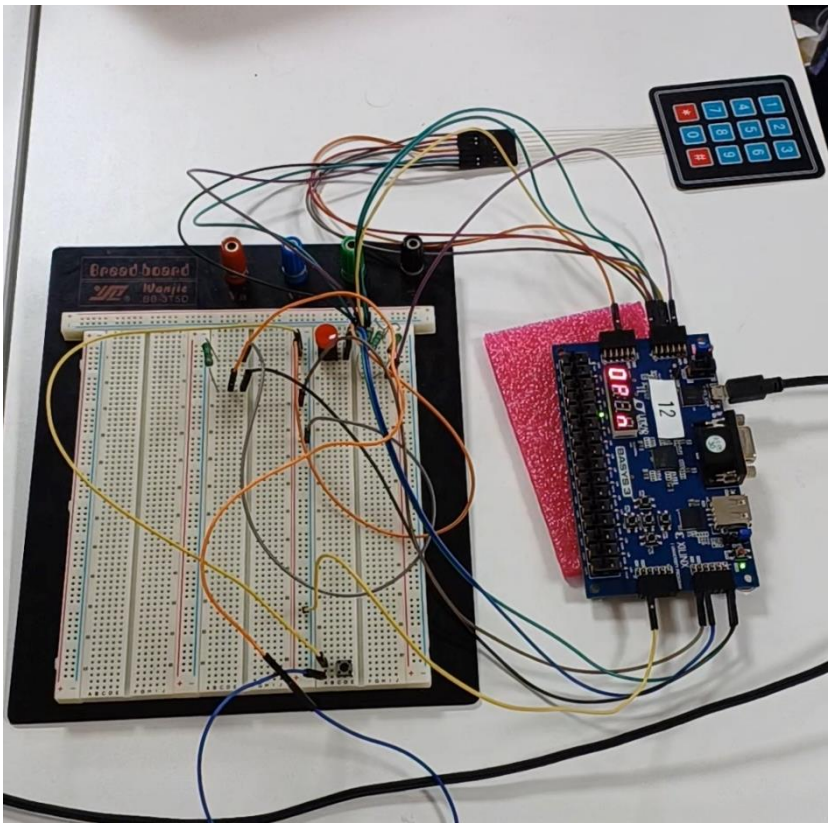
<WRONG2 STATE>



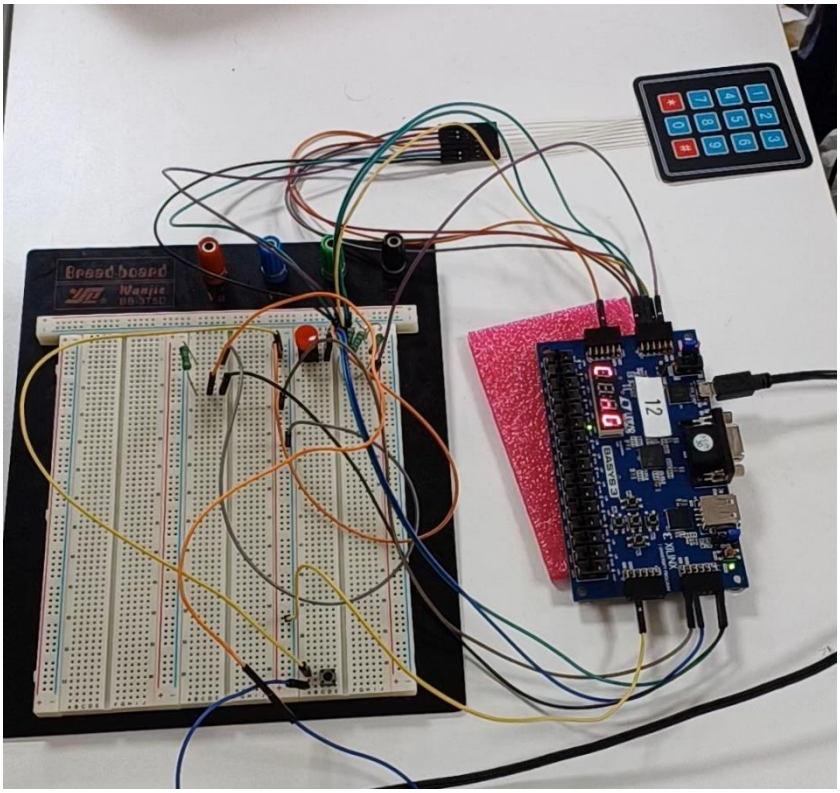
<LOCK STATE>



<ANSWER STATE>



<RESET STATE>



(7segment 4개는 실제로는 빠르게 번갈아가며 점멸하기 때문에 휴대폰 카메라의 한계로 인해 LOCK STATE, RESET STATE, ANSWER STATE는 실제와 다르게 일부만 사진에 찍혔습니다.)

VI 논의

이번 최종 프로젝트를 통해 Verilog의 고급 구문들의 사용법, 복잡한 프로젝트의 설계 방법과 그 구현 방법을 체득할 수 있었다. 이번 프로젝트에서 가장 어려웠던 부분은 소프트웨어의 이상적인 환경과 하드웨어의 현실적인 환경 사이의 차이로 클록 신호를 포함하여 구현이 제대로 되지 않았던 부분인데, 이를 counter 변수를 사용해 딜레이를 인위적으로 부여함으로써 해결할 수 있었다.