MINI INTERN
최종보고서

기업명: TSN LAB

작성자: 노지훈

INDEX

본과제: 7 SEGMENT 구현하기

- 7 segment 한 자리 표기하기 (0초부터 9초까지)
- 9초를 넘으면 0초부터 시작
- 7 segment가 없으면? -> GPIO로 구현
- 보드가 없으면? -> Simulation 으로 구현 (해당 O)

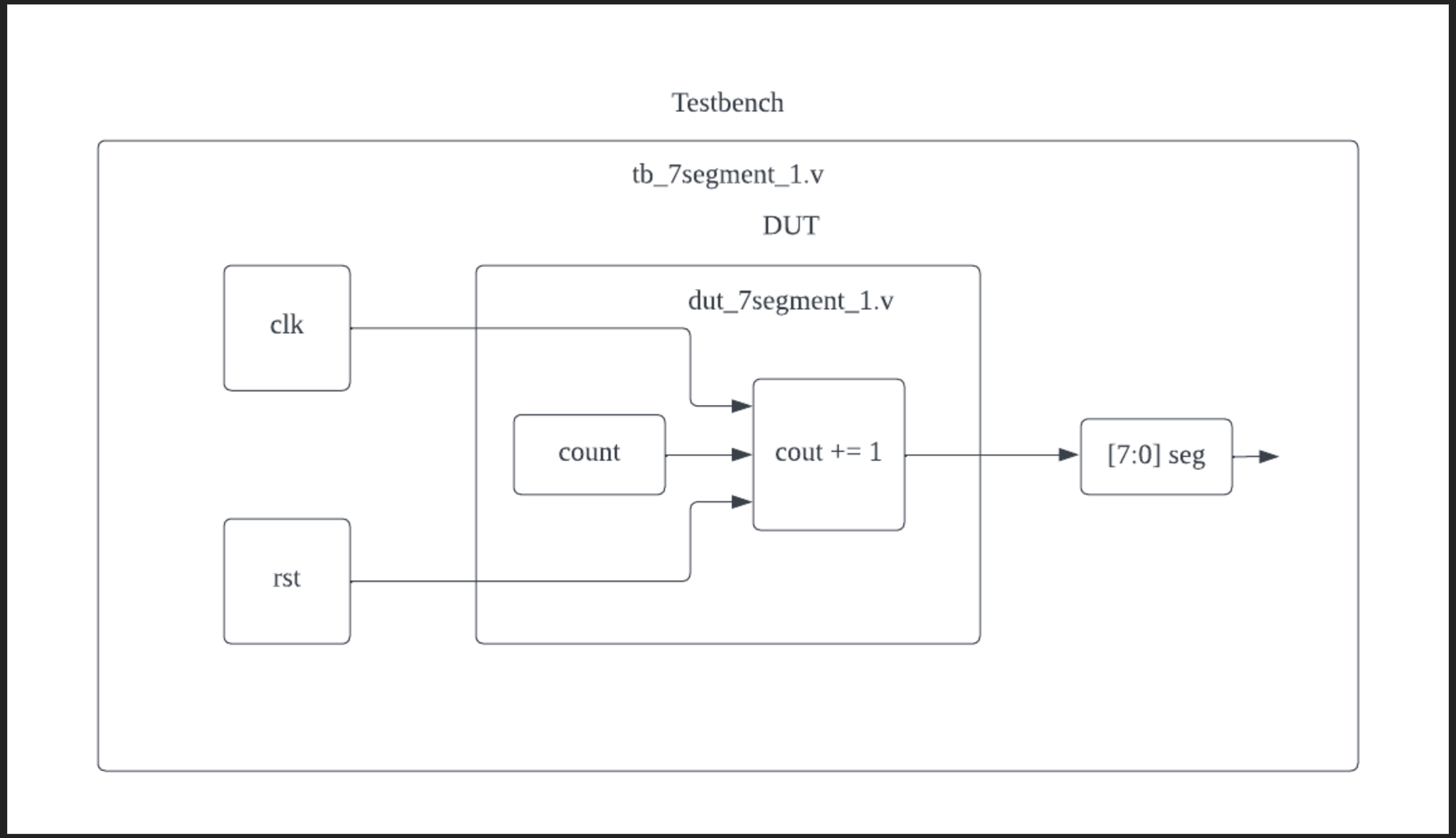
추가 과제 1: 3자리수 표기하기

- 7 segment 3 자릿수 표기하기
- 000 ~ 999를 표기
- 999를 넘으면 000부터 다시 시작하기

추가 과제 2: 분과 초 표기하기

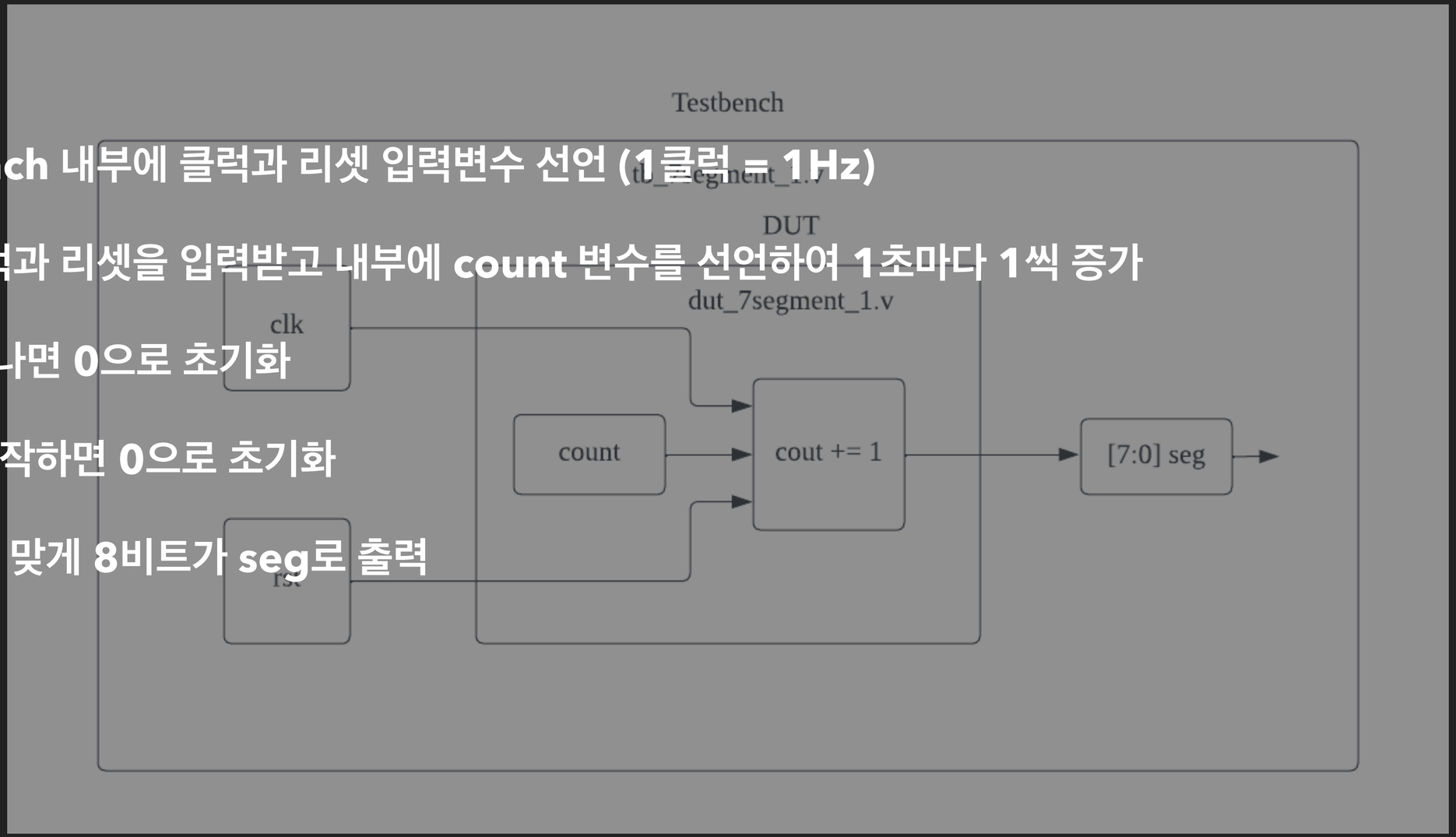
- 타이머 분과 초 표기하기
- EX1) 000은 0분 00초
- EX2) 123은 1분 23초
- 9분 59초 후에 다시 0분 00초부터 시작하기

설계 계획 - 본과제

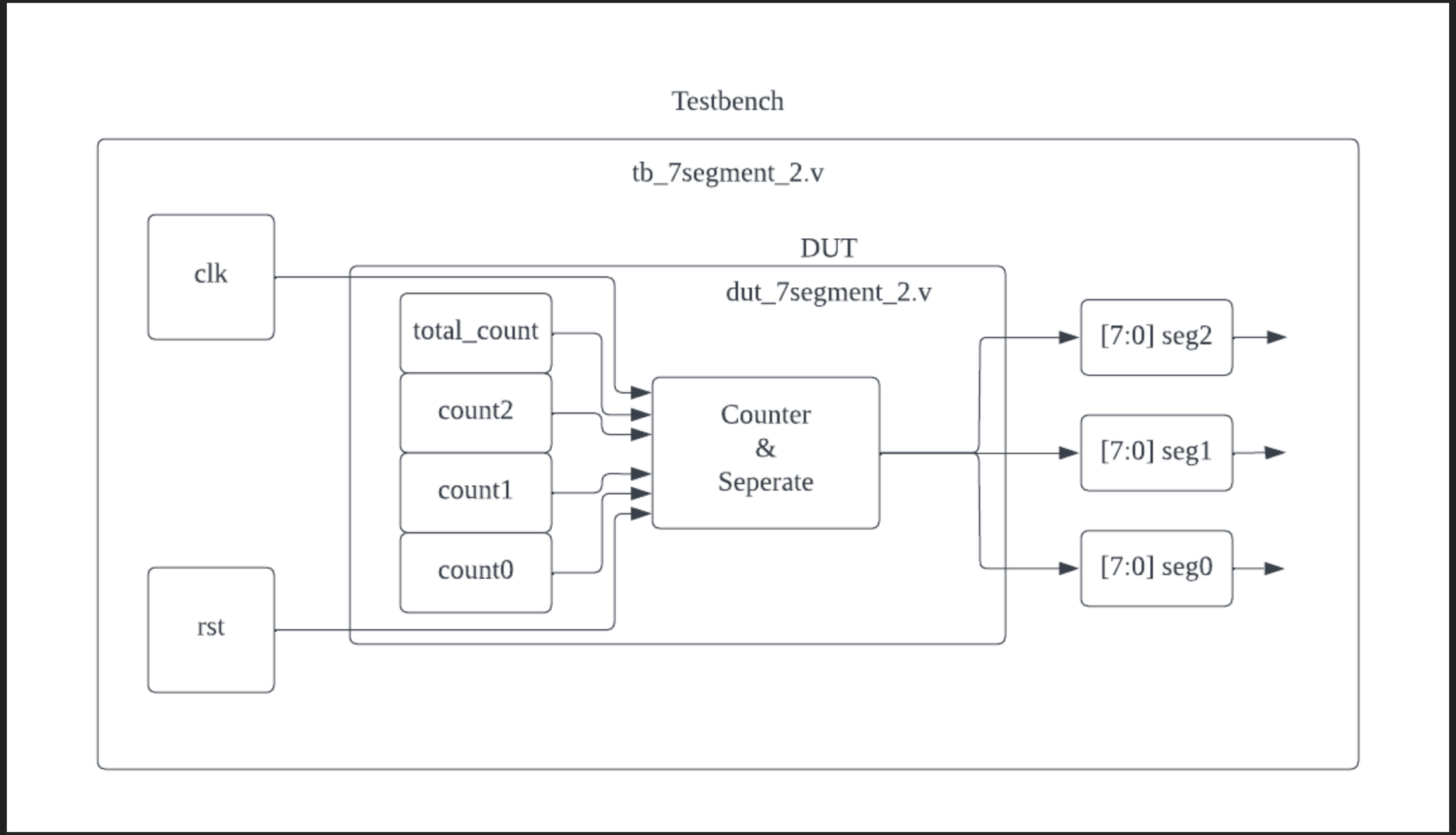

Testbench
tb_7segment_1.v
DUT
dut_7segment_1.v
clk
rst
count
cout += 1
[7:0] seg

설계 계획 - 본과제

- **Testbench 내부에 클럭과 리셋 입력변수 선언 (1클럭 = 1Hz)**
- **DUT 클럭과 리셋을 입력받고 내부에 count 변수를 선언하여 1초마다 1씩 증가**
- **9초가 지나면 0으로 초기화**
- **리셋이 동작하면 0으로 초기화**
- **각 숫자에 맞게 8비트가 seg로 출력**

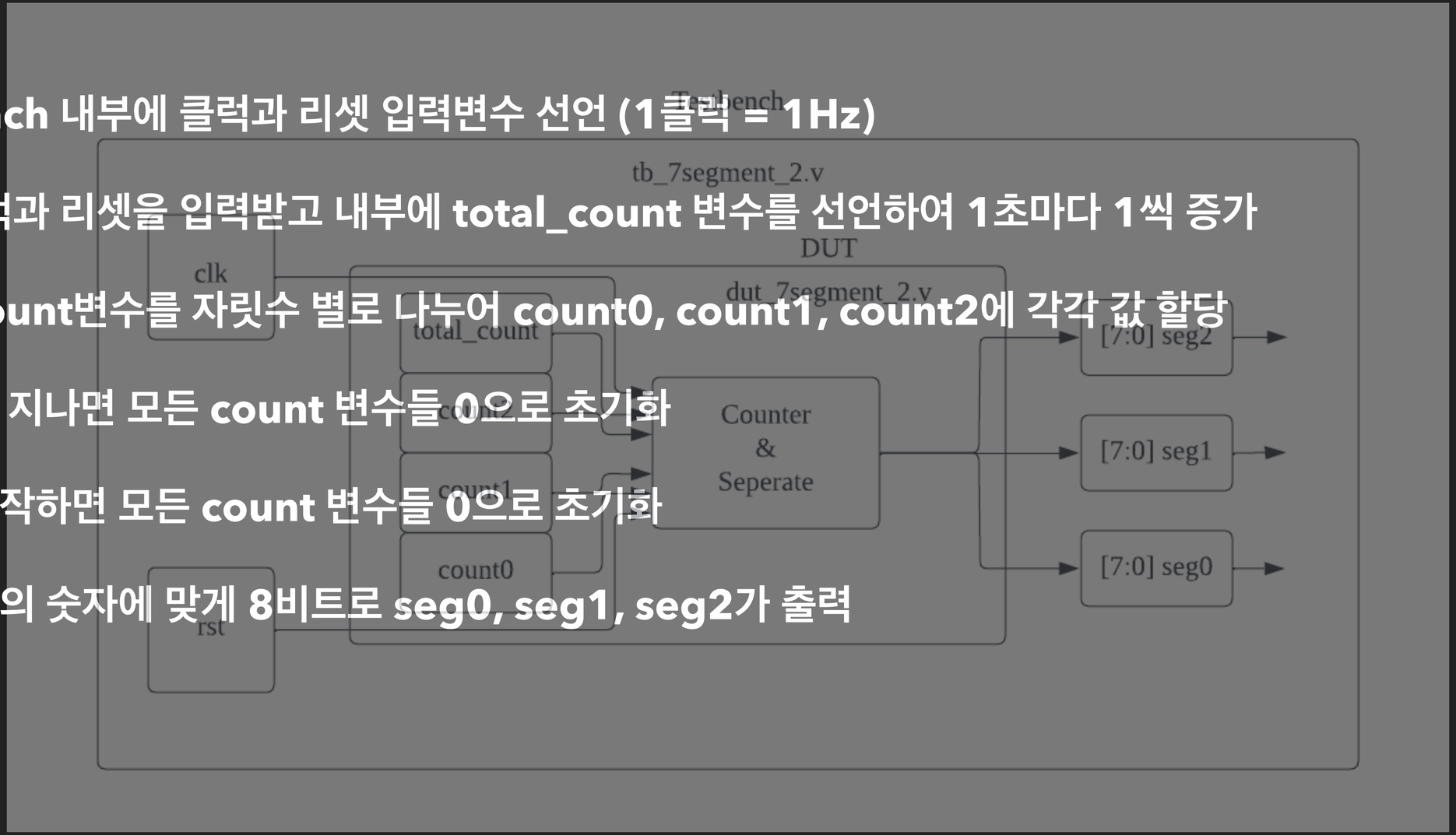


설계 계획 - 추가과제 1

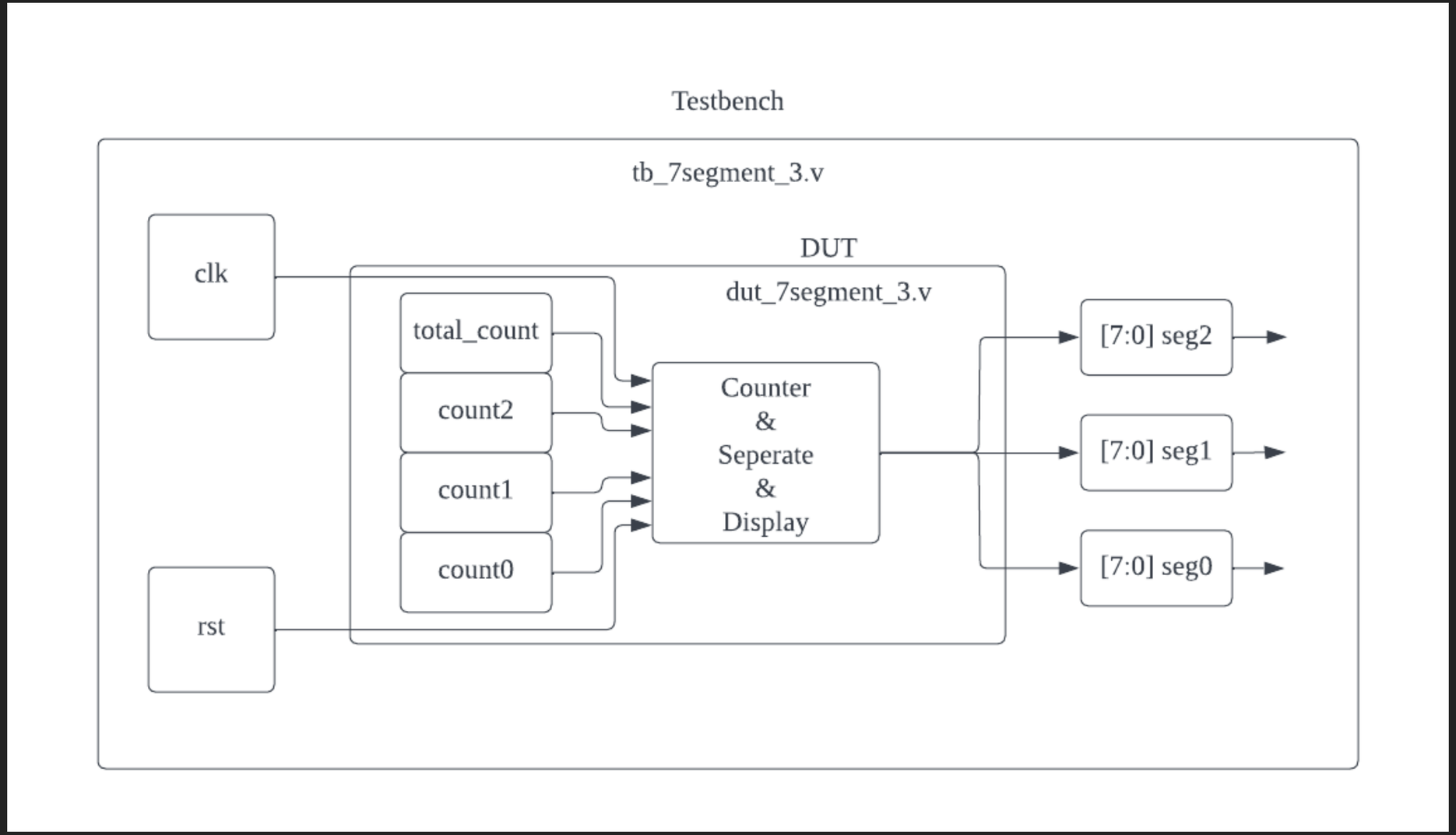

Testbench
tb_7segment_2.v
clk
rst
DUT
dut_7segment_2.v
total_count
count2
count1
count0
Counter
&
Seperate
[7:0] seg2
[7:0] seg1
[7:0] seg0

설계 계획 - 추가과제 1

- **Testbench 내부에 클럭과 리셋 입력변수 선언 (1클럭 = 1Hz)**
- **DUT 클럭과 리셋을 입력받고 내부에 total_count 변수를 선언하여 1초마다 1씩 증가**
- **total_count변수를 자릿수 별로 나누어 count0, count1, count2에 각각 값 할당**
- **999초가 지나면 모든 count 변수들 0으로 초기화**
- **리셋이 동작하면 모든 count 변수들 0으로 초기화**
- **각 자릿수의 숫자에 맞게 8비트로 seg0, seg1, seg2가 출력**

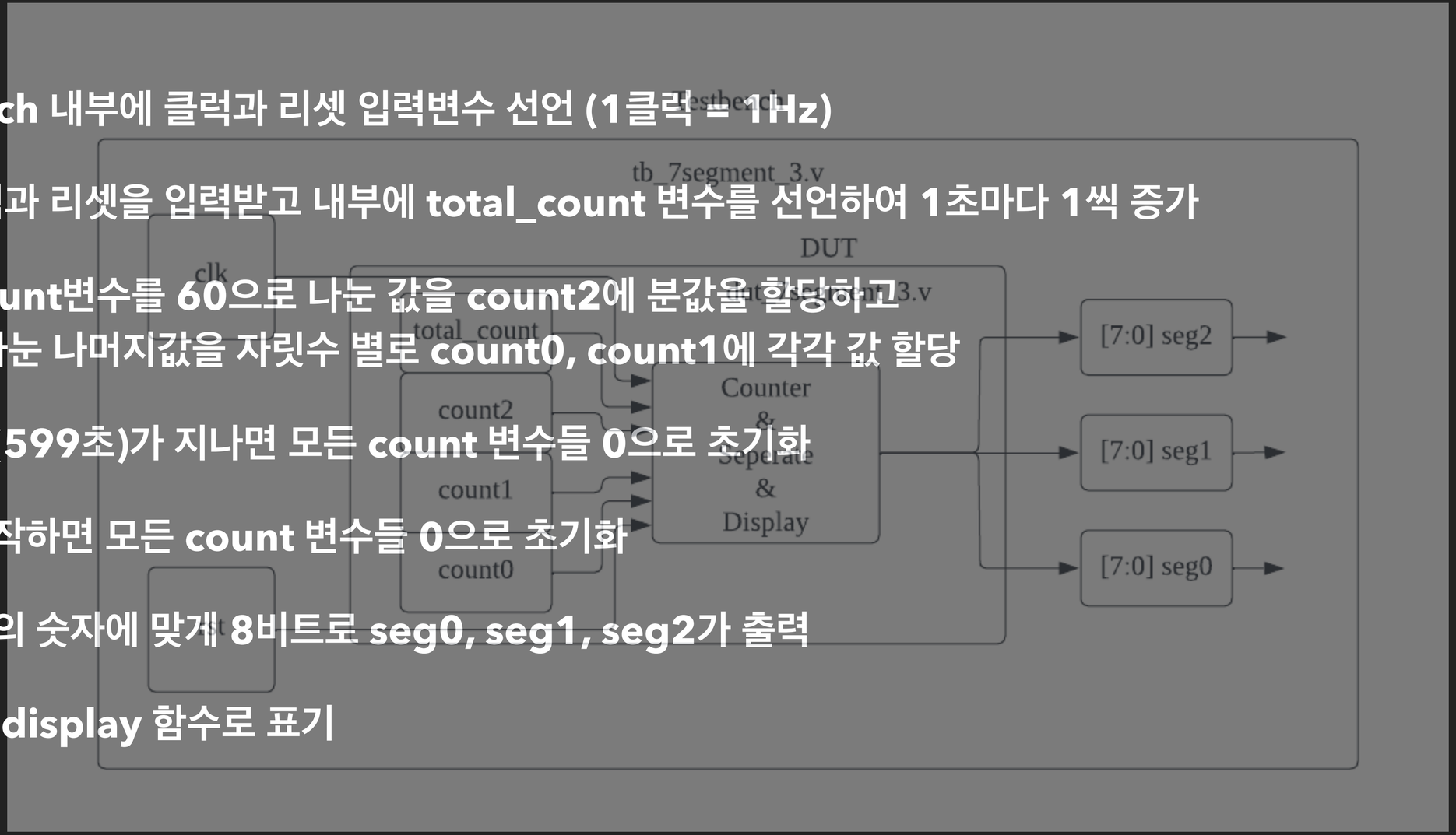


설계 계획 - 추가과제 2


Testbench
tb_7segment_3.v
clk
rst
DUT
dut_7segment_3.v
total_count
count2
count1
count0
Counter
&
Seperate
&
Display
[7:0] seg2
[7:0] seg1
[7:0] seg0

설계 계획 - 추가과제 2

- **Testbench 내부에 클럭과 리셋 입력변수 선언 (1클럭 = 1Hz)**
- **DUT 클럭과 리셋을 입력받고 내부에 total_count 변수를 선언하여 1초마다 1씩 증가**
- **total_count변수를 60으로 나눈 값을 count2에 분값을 할당하고 60으로 나눈 나머지값을 자릿수 별로 count0, count1에 각각 값 할당**
- **9분59초(599초)가 지나면 모든 count 변수들 0으로 초기화**
- **리셋이 동작하면 모든 count 변수들 0으로 초기화**
- **각 자릿수의 숫자에 맞게 8비트로 seg0, seg1, seg2가 출력**
- **분과 초는 display 함수로 표기**



TESTBENCH

- segment의 dot부분도 포함하여 8비트의 출력 설정
- 처음 20초는 타이머 동작
- 그 후 20초는 리셋에 의한 0으로 초기화
- 그 후 20초는 다시 타이머 동작 후 종료

```
8   // 현실시간과 매칭을 위해 1초 단위로 설정
9   `timescale 1s/1ms
10
11  module tb_7segment;
12  reg clk;
13  reg rst;
14  wire [7:0] seg;
15
16  // 1Hz 클럭 생성
17  always
18      #0.5 clk = ~clk;
19  initial begin
20      // 초기값 생성
21      clk = 0;
22      rst = 0;
23  #20
24      // 리셋 동작 확인
25      rst = 1;
26  #20
27      // 재동작 확인
28      rst = 0;
29  #20
30  $finish;
31  end
32
33  // dut 파일 연결
34  dut_7segment DUT(
35      .clk    (clk),
36      .rst    (rst),
37      .seg    (seg)
38  );
39  endmodule
```

DUT -1

- Testbench와 마찬가지로 timescale 및 변수 선언
- 내부에서 사용할 count와 s 변수를 int와 reg로 선언
- 리셋이 1이면 count = 0으로 선언하여 초기화
- 리셋이 0이면 count는 1클럭(상승 엣지)마다 1씩 증가
- 내부 카운트는 상승 에지, segment는 하강 에지에서 출력되므로 count가 10이 아닌 9일때 0으로 초기화

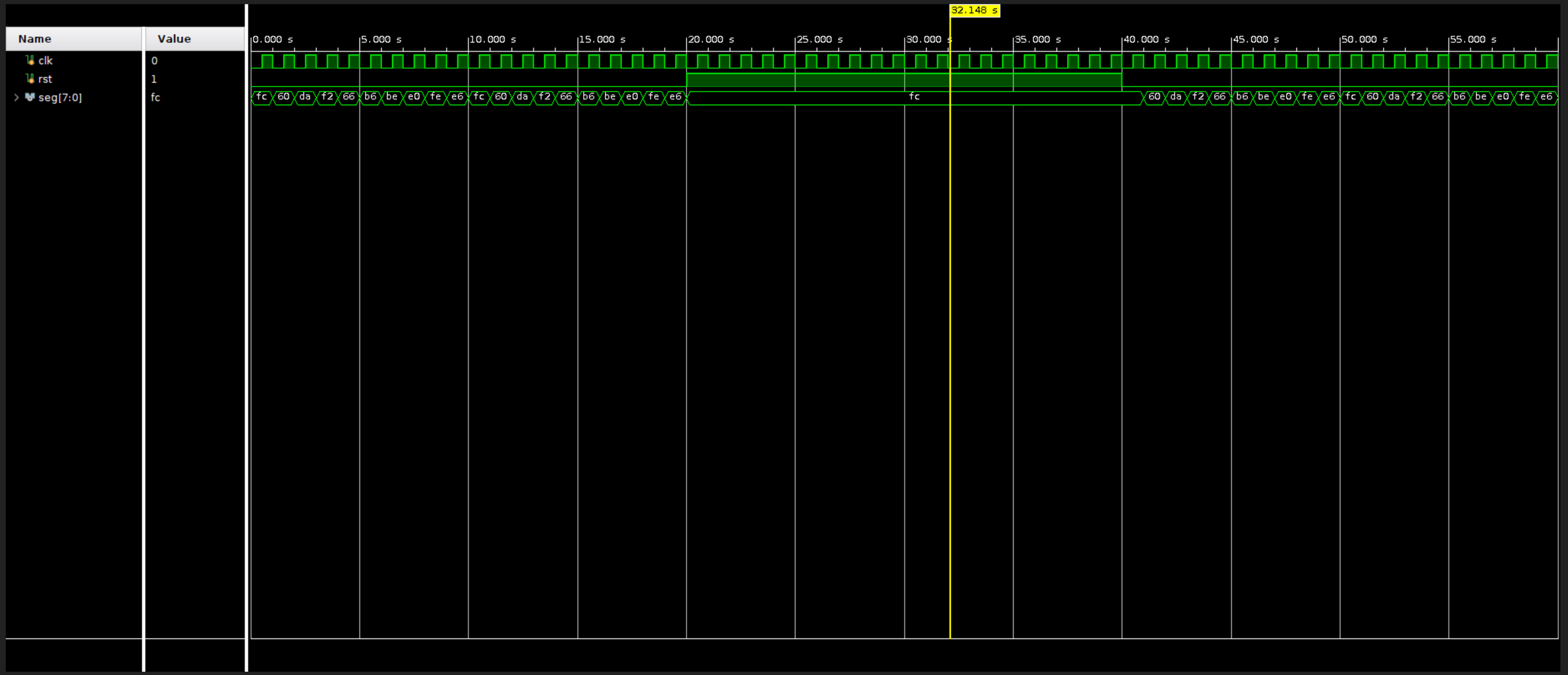
```
8   // 현실시간과 매칭을 위해 1초 단위로 설정
9   `timescale 1s/1ms
10
11  module dut_7segment(
12      input clk,
13      input rst,
14      output [7:0] seg
15  );
16      integer count = 0;
17      reg [7:0] s;
18
19  // 리셋 여부에 대한 내부 count 동작 선언
20  always @ (posedge clk) begin
21      if (rst == 1) begin
22          count = 0;
23      end else begin
24          if (count == 9) begin
25              count = 0;
26          end else begin
27              count = count + 1;
28          end
29      end
30  end
```

DUT -2

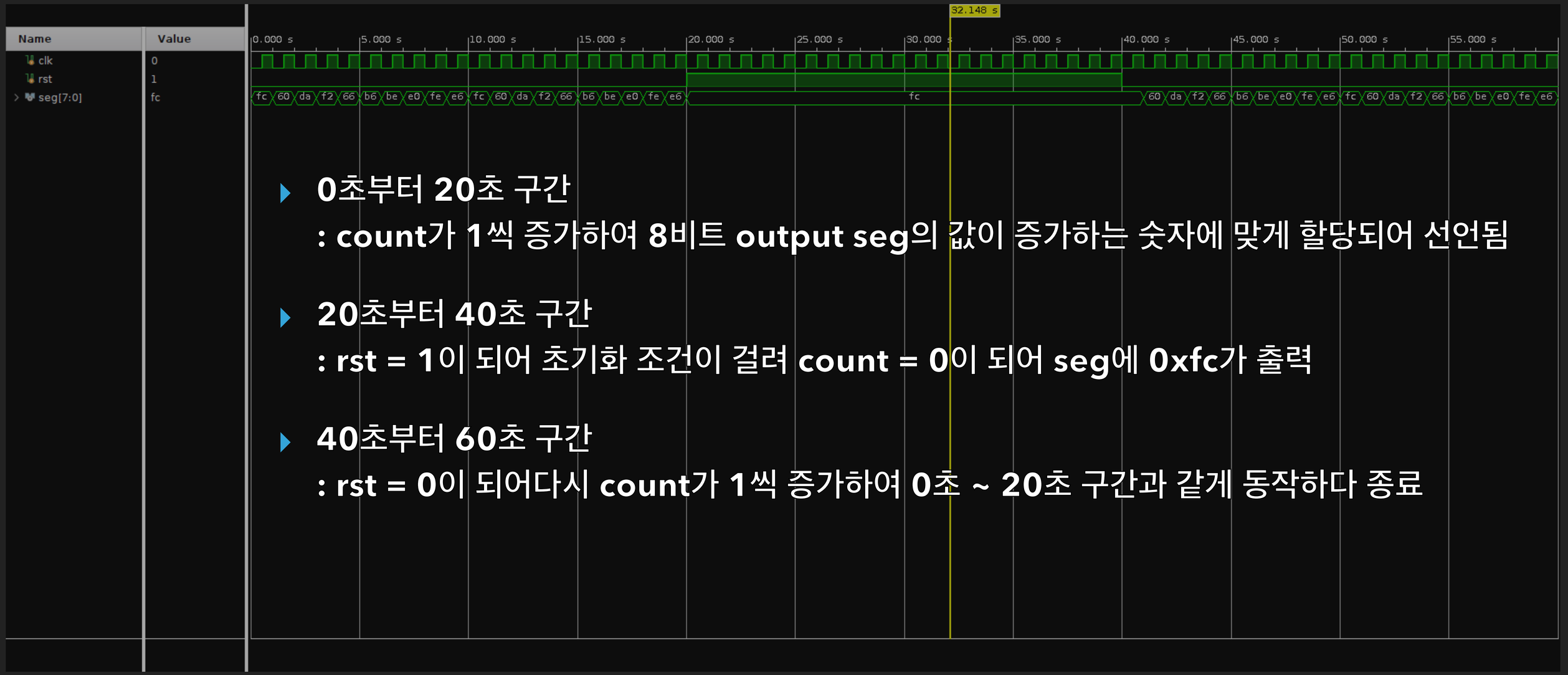
- 하강 에지에서 현재의 count값에 따라 segment 선언
- 최종 output인 seg에 현재 s값 할당하여 출력

```
32  // count 값에 따른 7-segment 선언
33  always @ (negedge clk) begin
34      s <=    (count == 0) ? 8'b11111100: //0
35              (count == 1) ? 8'b01100000: //1
36              (count == 2) ? 8'b11011010: //2
37              (count == 3) ? 8'b11110010: //3
38              (count == 4) ? 8'b01100110: //4
39              (count == 5) ? 8'b10110110: //5
40              (count == 6) ? 8'b10111110: //6
41              (count == 7) ? 8'b11100000: //7
42              (count == 8) ? 8'b11111110: //8
43              (count == 9) ? 8'b11100110: 8'b00000000; // 9, error: 0
44  end
45
46  // 최종 output인 seg에 값 할당
47  assign seg = s;
48
49  endmodule
```

결과 - 본과제


32.148 s
Name
Value
clk
0
rst
1
seg[7:0]
fc
0.000 s
5.000 s
10.000 s
15.000 s
20.000 s
25.000 s
30.000 s
35.000 s
40.000 s
45.000 s
50.000 s
55.000 s
fc 60 da f2 66 b6 be e0 fe e6 fc 60 da f2 66 b6 be e0 fe e6
fc
60 da f2 66 b6 be e0 fe e6 fc 60 da f2 66 b6 be e0 fe e6

결과 - 본과제



- **0초부터 20초 구간**
  : count가 1씩 증가하여 8비트 output seg의 값이 증가하는 숫자에 맞게 할당되어 선언됨
- **20초부터 40초 구간**
  : rst = 1이 되어 초기화 조건이 걸려 count = 0이 되어 seg에 0xfc가 출력
- **40초부터 60초 구간**
  : rst = 0이 되어다시 count가 1씩 증가하여 0초 ~ 20초 구간과 같게 동작하다 종료

TESTBENCH

- 3개의 segment 출력 변수 선언
- 999이상의 동작을 보기 위해 처음 1200초는 타이머 동작
- 그 후 20초는 리셋에 의한 0으로 초기화
- 그 후 다시 1200초는 다시 타이머 동작 후 종료

```
8   // 현실시간과 매칭을 위해 1초 단위로 설정
9   `timescale 1s/1ms
10
11  module tb_7segment_2;
12  reg clk;
13  reg rst;
14  wire [7:0] seg0;
15  wire [7:0] seg1;
16  wire [7:0] seg2;
17
18  // 1Hz 클럭 생성
19  always
20      #0.5 clk = ~clk;
21  initial begin
22      // 초기값 생성
23      clk = 0;
24      rst = 0;
25  #1200 // 999이상까지 테스트
26      // 리셋 동작 확인
27      rst = 1;
28  #20
29      // 재동작 확인
30      rst = 0;
31  #1200
32  $finish;
33  end
34
35  // dut 파일 연결
36  dut_7segment_2 DUT(
37      .clk    (clk),
38      .rst    (rst),
39      .seg0   (seg0),
40      .seg1   (seg1),
41      .seg2   (seg2)
42  );
43  endmodule
```

DUT -1

- Testbench와 마찬가지로 timescale 및 변수 선언
- 3자릿수에 맞게 seg 변수도 3개 선언
- 리셋이 1이면 total_count = 0으로 선언하여 초기화
- 리셋이 0이면 total_count는 1클럭마다 1씩 증가
- 10으로 나눈 몫과 나머지를 이용하여 각 자릿수를 분리하여 각 count 변수에 할당

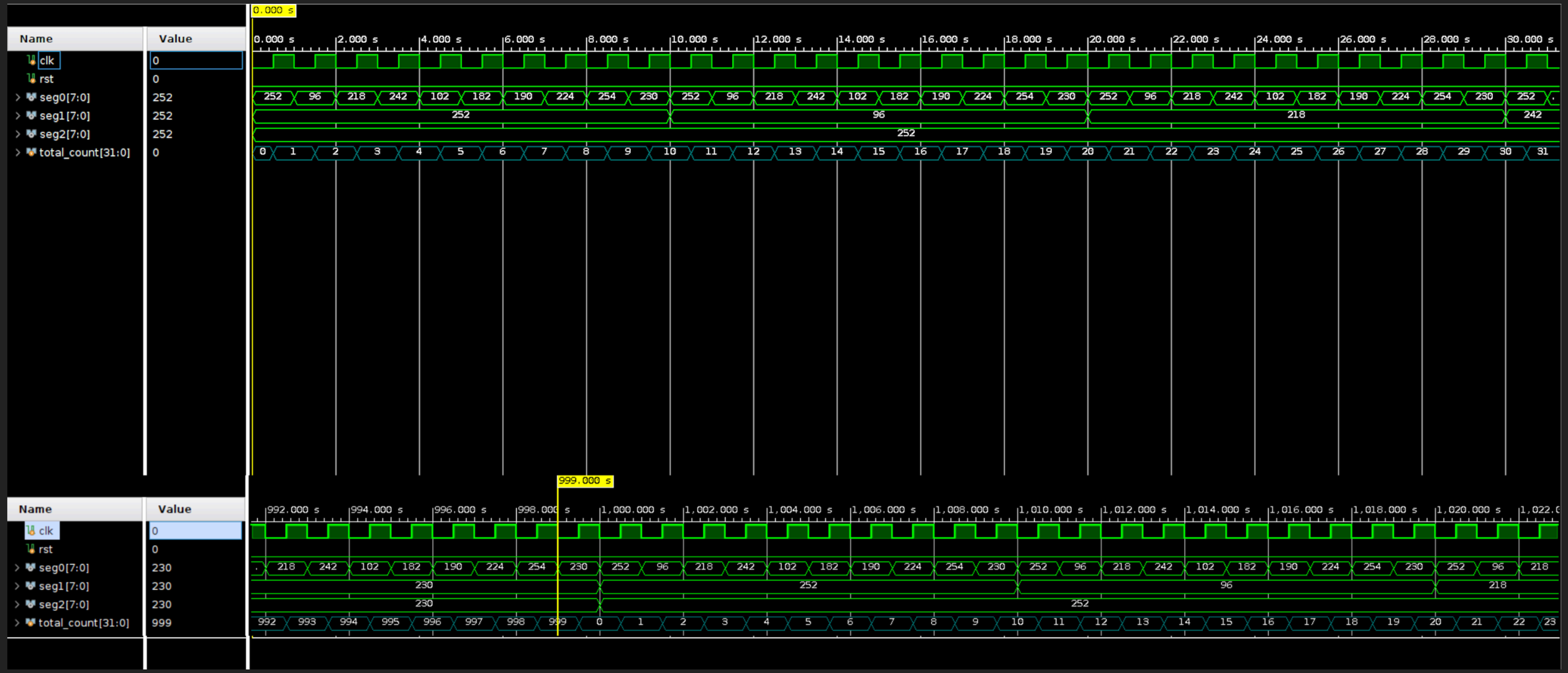
```
9   // 현실시간과 매칭을 위해 1초 단위로 설정
10  `timescale 1s/1ms
11
12  module dut_7segment_2(
13      input clk,
14      input rst,
15      // 3자릿수 Segment 인풋
16      output [7:0] seg0,
17      output [7:0] seg1,
18      output [7:0] seg2
19  );
20      // 000 ~ 999까지 세는 카운터용 변수 선언
21      integer total_count = 0;
22      // 자릿수 별 값 할당을 위한 변수 선언
23      integer count0 = 0;
24      integer count1 = 0;
25      integer count2 = 0;
26      reg [7:0] s0;
27      reg [7:0] s1;
28      reg [7:0] s2;
29
30  // 리셋 여부에 대한 내부 count 동작 선언
31  always @ (posedge clk) begin
32      // 리셋 초기화 조건
33      if (rst == 1) begin
34          total_count = 0;
35      end else begin
36          // 999 넘을 경우 초기화 조건
37          if (total_count == 999) begin
38              total_count = 0;
39              count0 = 0;
40              count1 = 0;
41              count2 = 0;
42          // 전체 카운터에서 각 자릿수로 할당하기 위해 숫자 분리
43          end else begin
44              total_count = total_count + 1;
45              count0 = total_count % 10;
46              count1 = ((total_count) / 10) % 10;
47              count2 = ((total_count) / 100) % 10;
48          end
49      end
50  end
```

DUT -2

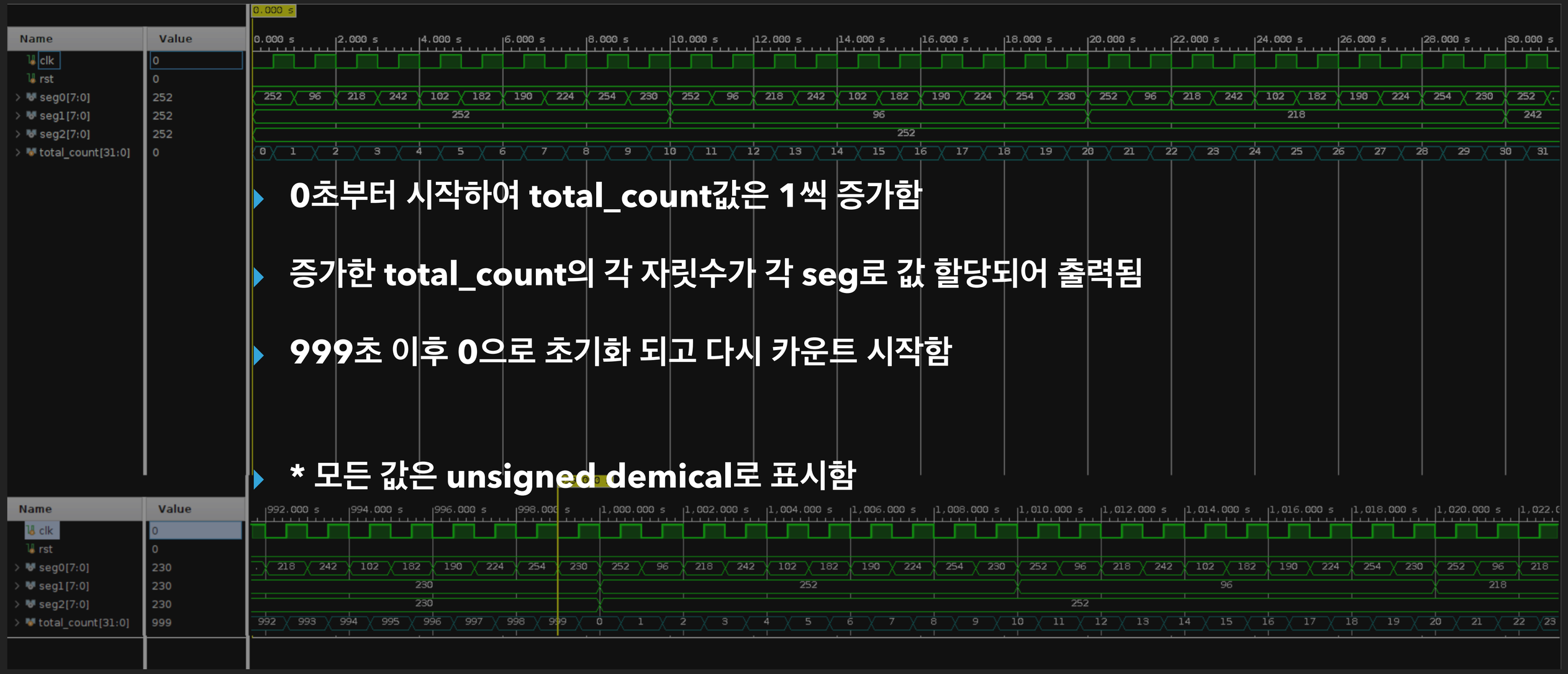
- 각 자릿수값을 세그먼트 표기값으로 변환하기 위해 task를 이용하여 반복되는 코드 단축
- 클럭의 하강에지에서 세그먼트 표기값을 할당해주는데 이 경우에도 에러 방지를 위해 리셋 동작 초기화 조건을 삽입
- 최종 output인 각 seg에 현재 각 s값을 할당하여 출력

```
52  task convert(
53      input [31:0] count,
54      output reg [7:0] s
55  );
56      case (count)
57          0 : s = 8'b11111100; //0, d252
58          1 : s = 8'b01100000; //1, d96
59          2 : s = 8'b11011010; //2, d218
60          3 : s = 8'b11110010; //3, d242
61          4 : s = 8'b01100110; //4, d102
62          5 : s = 8'b10110110; //5, d182
63          6 : s = 8'b10111110; //6, d190
64          7 : s = 8'b11100000; //7, d224
65          8 : s = 8'b11111110; //8, d254
66          9 : s = 8'b11100110; //9, d230
67          default: s = 8'b00000000; //error
68      endcase
69  endtask
70
71  // 각 count 값에 따른 7-segment 선언
72  always @ (negedge clk) begin
73      if (rst == 1) begin
74          // 리셋 초기화
75          total_count = 0;
76          count0 = 0;
77          count1 = 0;
78          count2 = 0;
79          s0 <= 8'b11111100;
80          s1 <= 8'b11111100;
81          s2 <= 8'b11111100;
82      end else begin
83          // 자릿수 별 값 할당
84          convert(count0, s0);
85          convert(count1, s1);
86          convert(count2, s2);
87      end
88  end
89
90  // 최종 output인 seg에 값 할당
91  assign seg0 = s0;
92  assign seg1 = s1;
93  assign seg2 = s2;
94
95  endmodule
```

결과 - 추가과제 1 (1)

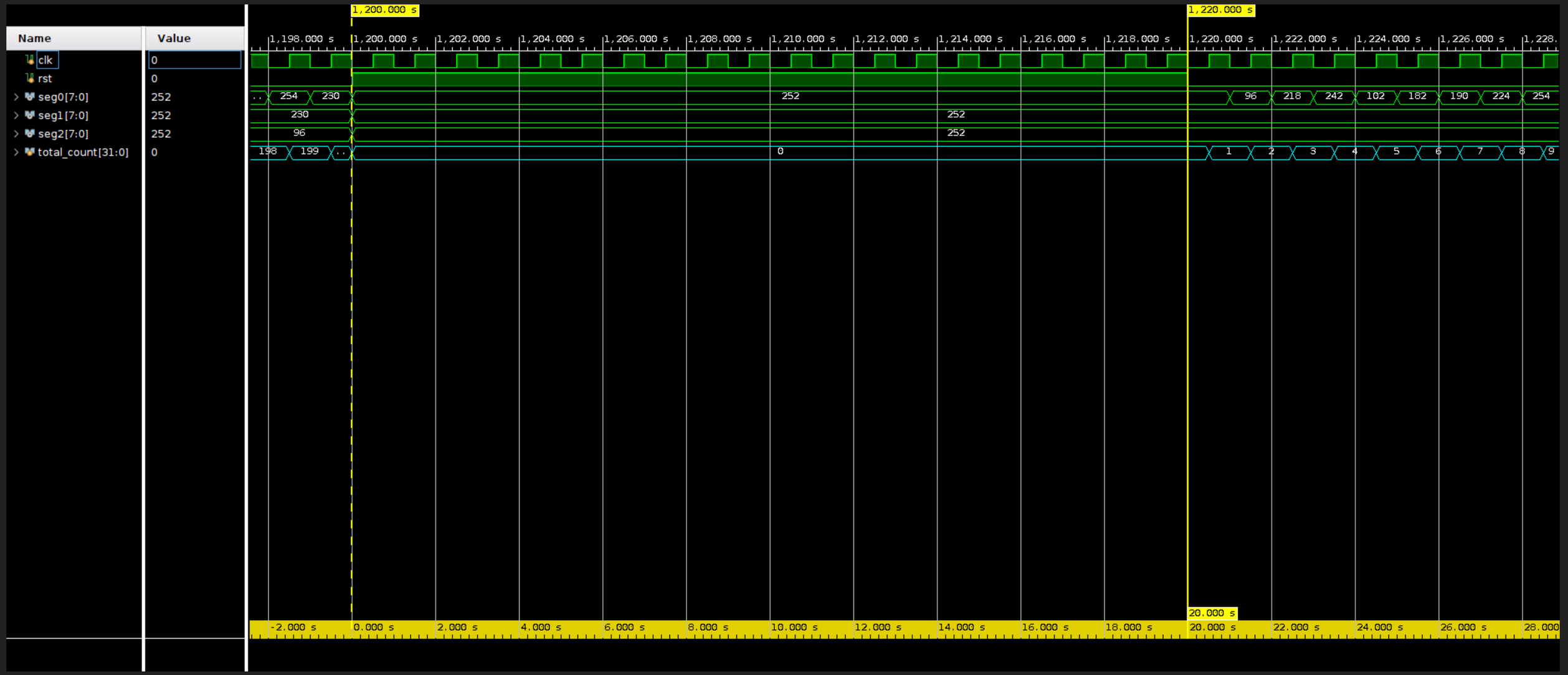

Name
Value
clk
0
rst
0
seg0[7:0]
252
seg1[7:0]
252
seg2[7:0]
252
total_count[31:0]
0
0.000 s
999.000 s
230
999

결과 - 추가과제 1 (1)

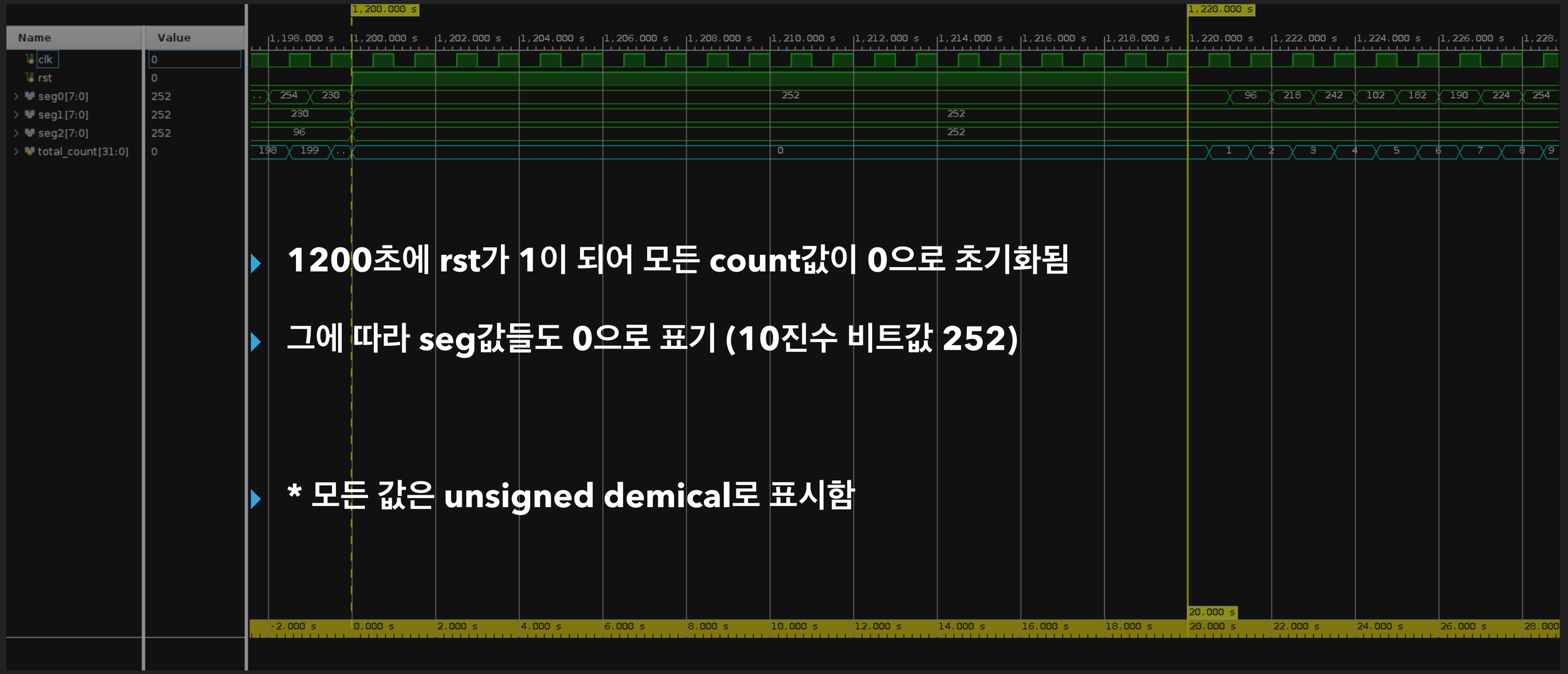


- **0초부터 시작하여 total_count값은 1씩 증가함**
- **증가한 total_count의 각 자릿수가 각 seg로 값 할당되어 출력됨**
- **999초 이후 0으로 초기화 되고 다시 카운트 시작함**
- *** 모든 값은 unsigned demical로 표시함**

결과 - 추가과제 1 (2)


Name
Value
clk
0
rst
0
seg0[7:0]
252
seg1[7:0]
252
seg2[7:0]
252
total_count[31:0]
0
1,200.000 s
1,220.000 s
1,198.000 s
1,200.000 s
1,202.000 s
1,204.000 s
1,206.000 s
1,208.000 s
1,210.000 s
1,212.000 s
1,214.000 s
1,216.000 s
1,218.000 s
1,220.000 s
1,222.000 s
1,224.000 s
1,226.000 s
1,228.
254
230
252
96
218
242
102
182
190
224
254
230
252
96
252
198
199
0
1
2
3
4
5
6
7
8
9
20.000 s
-2.000 s
0.000 s
2.000 s
4.000 s
6.000 s
8.000 s
10.000 s
12.000 s
14.000 s
16.000 s
18.000 s
20.000 s
22.000 s
24.000 s
26.000 s
28.000

결과 - 추가과제 1 (2)



- 1200초에 rst가 1이 되어 모든 count값이 0으로 초기화됨
- 그에 따라 seg값들도 0으로 표기 (10진수 비트값 252)
- * 모든 값은 unsigned demical로 표시함

TESTBENCH

- 3개의 segment 출력 변수 선언
- 599이상의 동작을 보기 위해 처음 1200초는 타이머 동작
- 그 후 20초는 리셋에 의한 0으로 초기화
- 그 후 다시 1200초는 다시 타이머 동작 후 종료

```
8   // 현실시간과 매칭을 위해 1초 단위로 설정
9   `timescale 1s/1ms
10
11  module tb_7segment_3;
12  reg clk;
13  reg rst;
14  wire [7:0] seg0;
15  wire [7:0] seg1;
16  wire [7:0] seg2;
17
18  // 1Hz 클럭 생성
19  always
20      #0.5 clk = ~clk;
21  initial begin
22      // 초기값 생성
23      clk = 0;
24      rst = 0;
25  #1200 //599이상까지 테스트
26      // 리셋 동작 확인
27      rst = 1;
28  #20
29      // 재동작 확인
30      rst = 0;
31  #1200
32  $finish;
33  end
34
35  // dut 파일 연결
36  dut_7segment_3 DUT(
37      .clk    (clk),
38      .rst    (rst),
39      .seg0   (seg0),
40      .seg1   (seg1),
41      .seg2   (seg2)
42  );
43  endmodule
```

DUT -1

- Testbench와 마찬가지로 timescale 및 변수 선언
- 3자릿수에 맞게 seg 변수도 3개 선언
- 리셋이 1이면 total_count = 0으로 선언하여 초기화
- 리셋이 0이면 total_count는 1클럭마다 1씩 증가
- 60으로 나누어 분과 초를 분리하고 10으로 나누어 초의 각 자릿수를 분리하여 각 count 변수에 할당
- Display 사용 시 공백 제거를 위해 문자열로 변환하고 task를 사용하여 코드 단축

```
10  // 현실시간과 매칭을 위해 1초 단위로 설정
11  `timescale 1s/1ms
12
13  module dut_7segment_3(
14      input clk,
15      input rst,
16      // 3자릿수 Segment 인풋
17      output [7:0] seg0,
18      output [7:0] seg1,
19      output [7:0] seg2
20  );
21      // 000 ~ 599까지 세는 카운터용 변수 선언
22      integer total_count = 0;
23      // 자릿수 별 값 할당을 위한 변수 선언
24      integer count0 = 0;
25      integer count1 = 0;
26      integer count2 = 0;
27      // 분과 초 표기를 위한용
28      reg [7:0] digit0 = "0";
29      reg [7:0] digit1 = "0";
30      reg [7:0] digit2 = "0";
31      reg [7:0] s0;
32      reg [7:0] s1;
33      reg [7:0] s2;
34
35  // 분과 초 표기용 반복작업 task로 정의
36  task digit_converter(
37      input [31:0] count,
38      output [7:0] digit
39  );
40      case (count)
41          0   : digit = "0";
42          1   : digit = "1";
43          2   : digit = "2";
44          3   : digit = "3";
45          4   : digit = "4";
46          5   : digit = "5";
47          6   : digit = "6";
48          7   : digit = "7";
49          8   : digit = "8";
50          9   : digit = "9";
51          default : digit = "0";
52      endcase
53  endtask
```

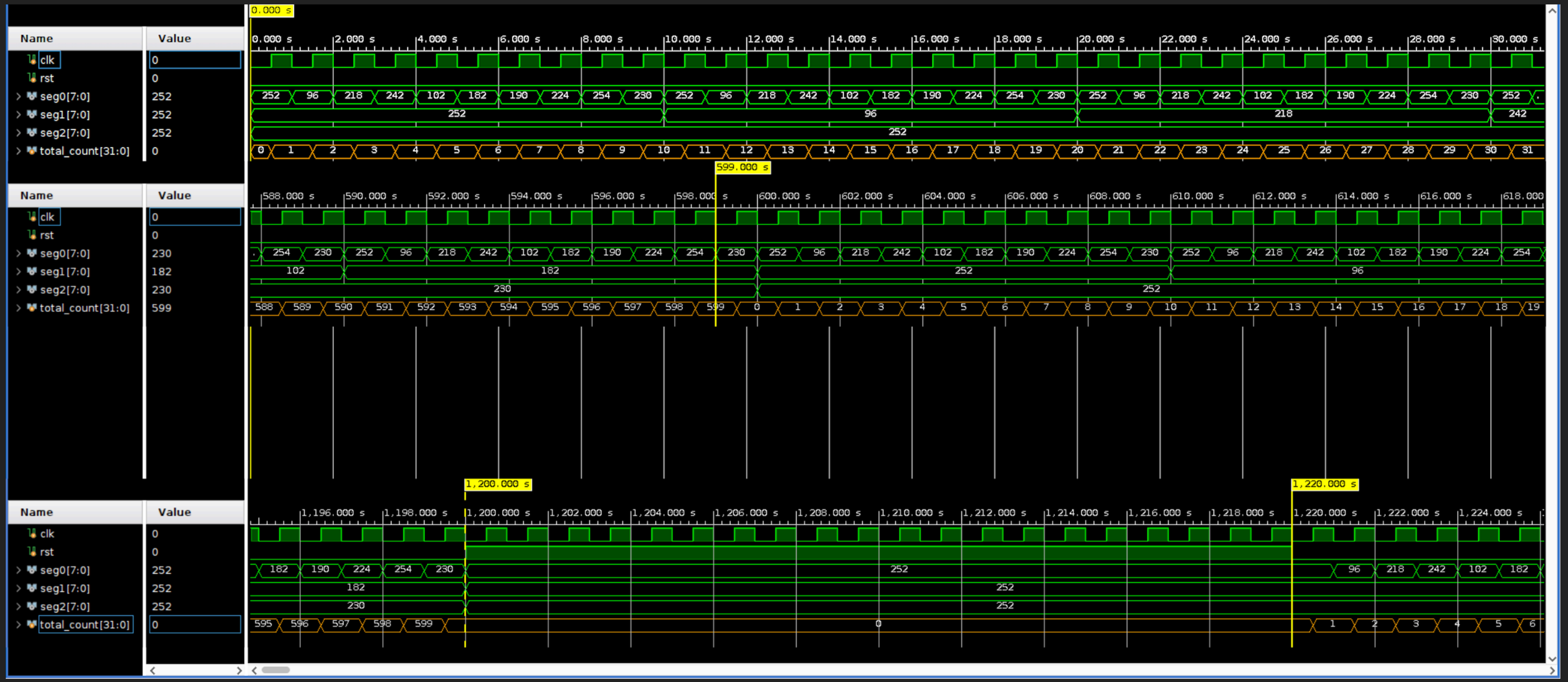
```
54
55  // 리셋 여부에 대한 내부 count 동작 선언
56  always @ (posedge clk) begin
57      // 리셋 초기화 조건
58      if (rst == 1) begin
59          total_count = 0;
60      end else begin
61          // 599 넘을 경우 초기화 조건
62          if (total_count == 599) begin
63              total_count = 0;
64              count0 = 0;
65              count1 = 0;
66              count2 = 0;
67          // 전체 카운터에서 각 자릿수로 할당하기 위해 숫자 분리
68          end else begin
69              total_count = total_count + 1;
70              count2 = total_count / 60;
71              count1 = (total_count % 60) / 10;
72              count0 = (total_count % 60) % 10;
73          end
74      end
75
76      // display를 위해 task 사용
77      digit_converter(count2, digit2);
78      digit_converter(count1, digit1);
79      digit_converter(count0, digit0);
80
81  end
82
```

DUT -2

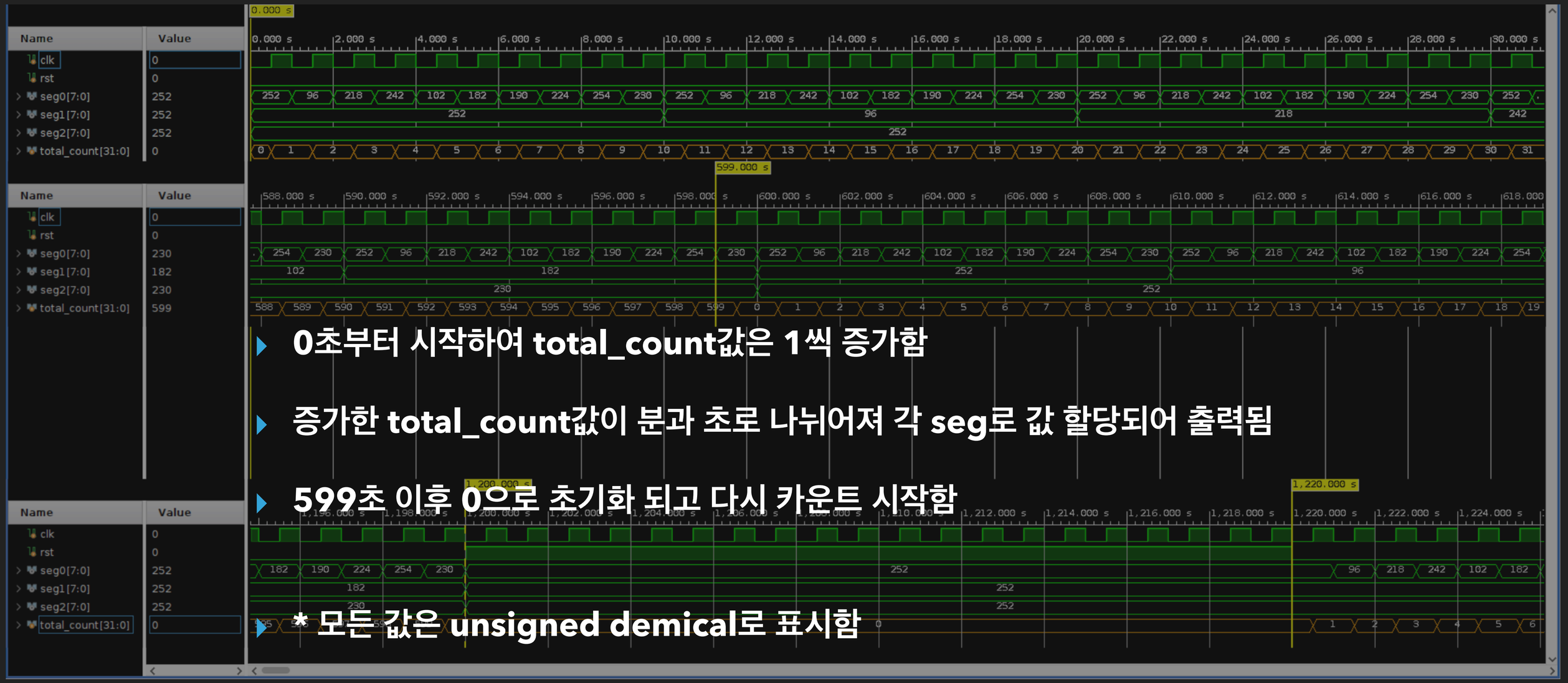
- 각 자릿수값을 세그먼트 표기값으로 변환하기 위해 task를 이용하여 반복되는 코드 단축
- 클럭의 하강에지에서 세그먼트 표기값을 할당해주는데 이 경우에도 에러 방지를 위해 리셋 동작 초기화 조건을 삽입
- Display함수를 사용하여 콘솔창에 타이머 시간 출력 -> 인식문제로 인해 min과 sec로 표기
- 최종 output인 각 seg에 현재 각 s값을 할당하여 출력

```
83  // 7 segment 할당 반복작업 task로 정의
84  task convert(
85      input [31:0] count,
86      output reg [7:0] s
87  );
88      case (count)
89          0 : s = 8'b11111100; //0, d252
90          1 : s = 8'b01100000; //1, d96
91          2 : s = 8'b11011010; //2, d218
92          3 : s = 8'b11110010; //3, d242
93          4 : s = 8'b01100110; //4, d102
94          5 : s = 8'b10110110; //5, d182
95          6 : s = 8'b10111110; //6, d190
96          7 : s = 8'b11100000; //7, d224
97          8 : s = 8'b11111110; //8, d254
98          9 : s = 8'b11100110; //9, d230
99          default: s = 8'b00000000; //error
100     endcase
101 endtask
102
103 // count 값에 따른 7-segment 선언
104 always @ (negedge clk) begin
105     // 자릿수 별 값 할당
106     if (rst == 1) begin
107         total_count = 0;
108         count0 = 0;
109         count1 = 0;
110         count2 = 0;
111         s0 <= 8'b11111100;
112         s1 <= 8'b11111100;
113         s2 <= 8'b11111100;
114     end else begin
115         convert(count0, s0);
116         convert(count1, s1);
117         convert(count2, s2);
118     end
119     // 분과 초 표기, 비바도에서 한글 인식 안돼서 영어로 표기
120     $display("count: [%d] : [%s]min [%s]sec : time: [%d]", total_count, digit2, {digit1,digit0}, $time);
121 end
122 // 최종 output인 seg에 값 할당
123 assign seg0 = s0;
124 assign seg1 = s1;
125 assign seg2 = s2;
126 endmodule
```

결과 - 추가과제 2 (1)

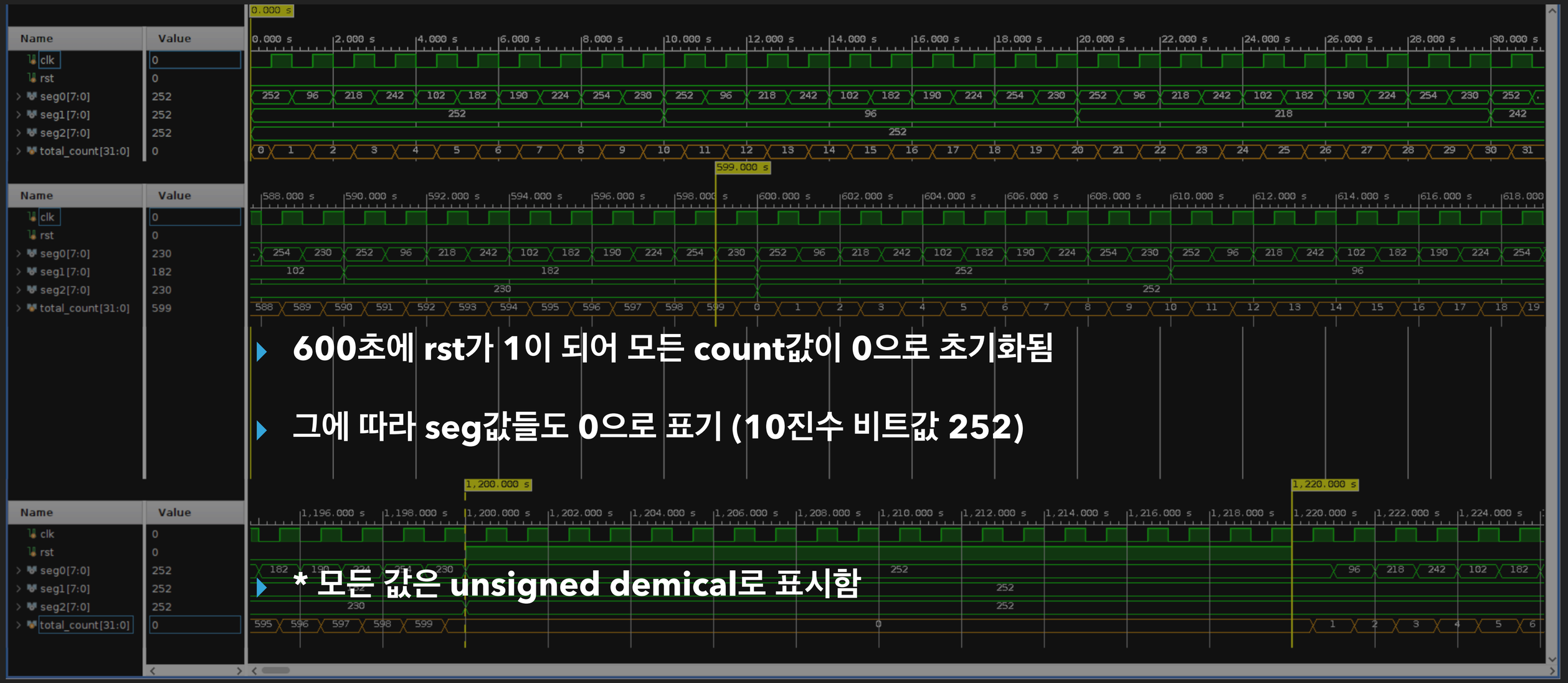

Name
Value
clk
rst
seg0[7:0]
seg1[7:0]
seg2[7:0]
total_count[31:0]
0
0
252
252
252
0
599.000 s
230
182
230
599
1,200.000 s
1,220.000 s
252
0

결과 - 추가과제 2 (1)



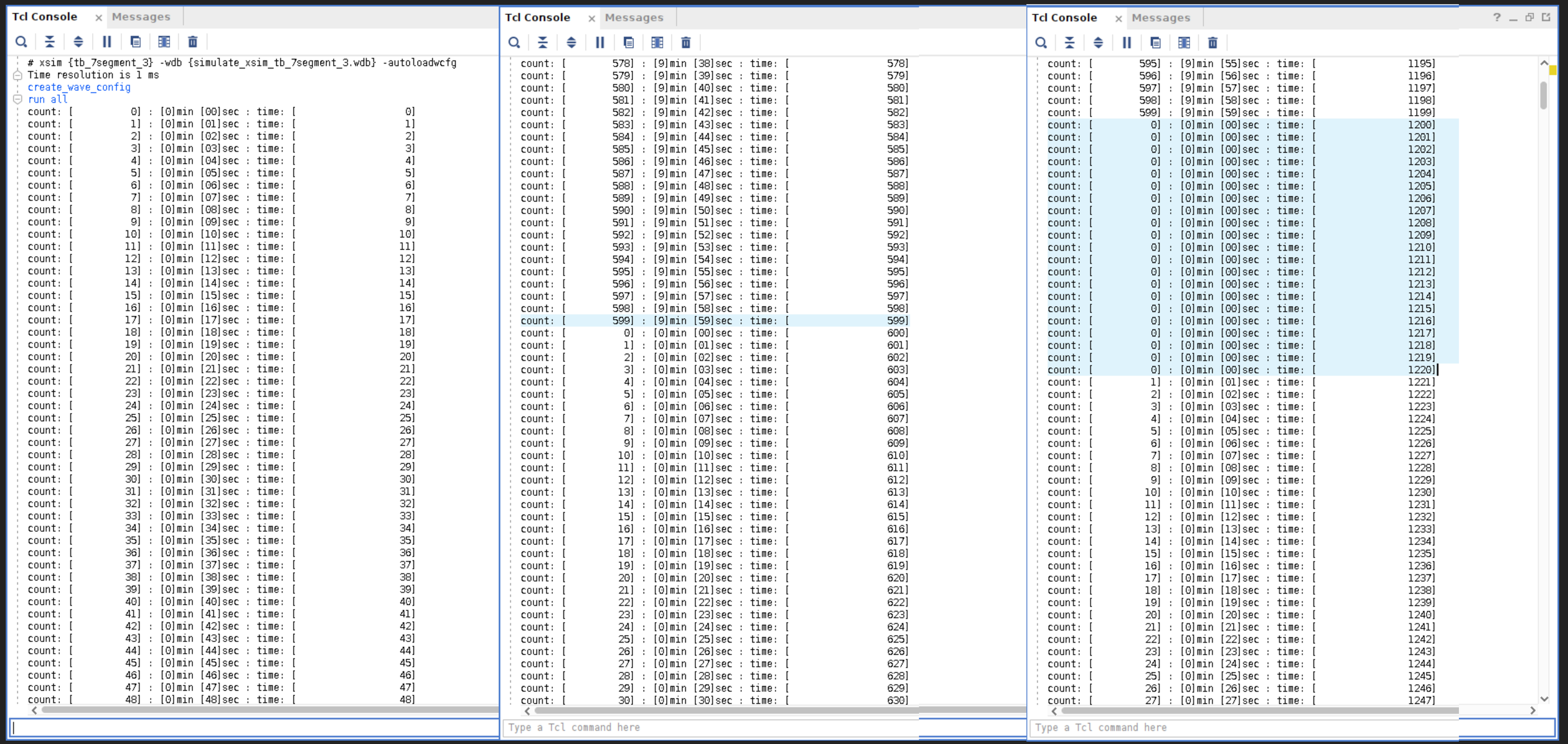
- 0초부터 시작하여 total_count값은 1씩 증가함
- 증가한 total_count값이 분과 초로 나뉘어져 각 seg로 값 할당되어 출력됨
- 599초 이후 0으로 초기화 되고 다시 카운트 시작함
- * 모든 값은 unsigned demical로 표시함

결과 - 추가과제 2 (1)

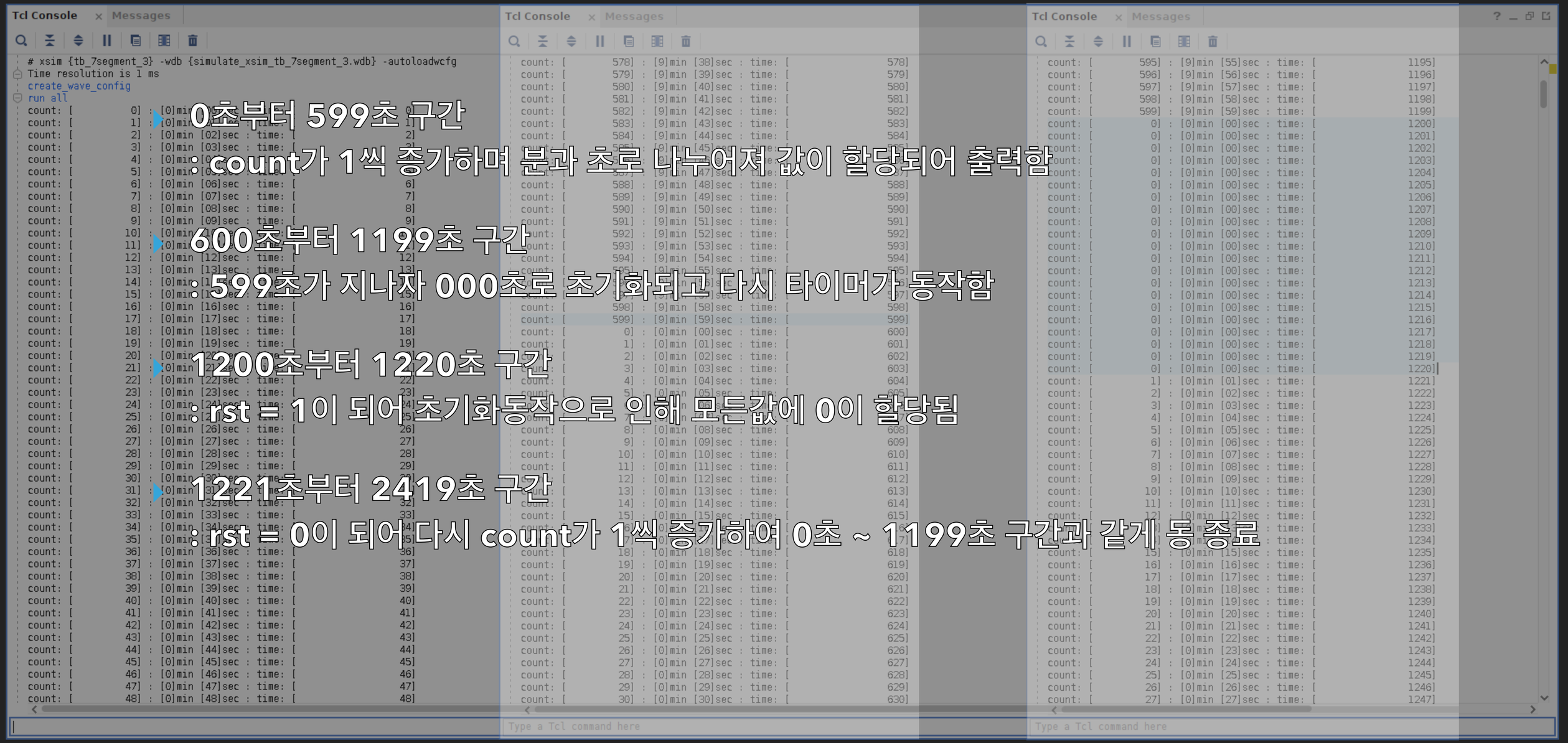


- 600초에 rst가 1이 되어 모든 count값이 0으로 초기화됨
- 그에 따라 seg값들도 0으로 표기 (10진수 비트값 252)
- * 모든 값은 unsigned demical로 표시함

결과 - 추가과제 2 (2)


Tcl Console
Messages
xsim {tb_7segment_3} -wdb {simulate_xsim_tb_7segment_3.wdb} -autoloadwcfg
Time resolution is 1 ms
create_wave_config
run all
count: [0] : [0]min [00]sec : time: [0]
count: [1] : [0]min [01]sec : time: [1]
count: [2] : [0]min [02]sec : time: [2]
count: [3] : [0]min [03]sec : time: [3]
count: [4] : [0]min [04]sec : time: [4]
count: [5] : [0]min [05]sec : time: [5]
count: [6] : [0]min [06]sec : time: [6]
count: [7] : [0]min [07]sec : time: [7]
count: [8] : [0]min [08]sec : time: [8]
count: [9] : [0]min [09]sec : time: [9]
count: [10] : [0]min [10]sec : time: [10]
count: [11] : [0]min [11]sec : time: [11]
count: [12] : [0]min [12]sec : time: [12]
count: [13] : [0]min [13]sec : time: [13]
count: [14] : [0]min [14]sec : time: [14]
count: [15] : [0]min [15]sec : time: [15]
count: [16] : [0]min [16]sec : time: [16]
count: [17] : [0]min [17]sec : time: [17]
count: [18] : [0]min [18]sec : time: [18]
count: [19] : [0]min [19]sec : time: [19]
count: [20] : [0]min [20]sec : time: [20]
count: [21] : [0]min [21]sec : time: [21]
count: [22] : [0]min [22]sec : time: [22]
count: [23] : [0]min [23]sec : time: [23]
count: [24] : [0]min [24]sec : time: [24]
count: [25] : [0]min [25]sec : time: [25]
count: [26] : [0]min [26]sec : time: [26]
count: [27] : [0]min [27]sec : time: [27]
count: [28] : [0]min [28]sec : time: [28]
count: [29] : [0]min [29]sec : time: [29]
count: [30] : [0]min [30]sec : time: [30]
count: [31] : [0]min [31]sec : time: [31]
count: [32] : [0]min [32]sec : time: [32]
count: [33] : [0]min [33]sec : time: [33]
count: [34] : [0]min [34]sec : time: [34]
count: [35] : [0]min [35]sec : time: [35]
count: [36] : [0]min [36]sec : time: [36]
count: [37] : [0]min [37]sec : time: [37]
count: [38] : [0]min [38]sec : time: [38]
count: [39] : [0]min [39]sec : time: [39]
count: [40] : [0]min [40]sec : time: [40]
count: [41] : [0]min [41]sec : time: [41]
count: [42] : [0]min [42]sec : time: [42]
count: [43] : [0]min [43]sec : time: [43]
count: [44] : [0]min [44]sec : time: [44]
count: [45] : [0]min [45]sec : time: [45]
count: [46] : [0]min [46]sec : time: [46]
count: [47] : [0]min [47]sec : time: [47]
count: [48] : [0]min [48]sec : time: [48]
Tcl Console
Messages
count: [578] : [9]min [38]sec : time: [578]
count: [579] : [9]min [39]sec : time: [579]
count: [580] : [9]min [40]sec : time: [580]
count: [581] : [9]min [41]sec : time: [581]
count: [582] : [9]min [42]sec : time: [582]
count: [583] : [9]min [43]sec : time: [583]
count: [584] : [9]min [44]sec : time: [584]
count: [585] : [9]min [45]sec : time: [585]
count: [586] : [9]min [46]sec : time: [586]
count: [587] : [9]min [47]sec : time: [587]
count: [588] : [9]min [48]sec : time: [588]
count: [589] : [9]min [49]sec : time: [589]
count: [590] : [9]min [50]sec : time: [590]
count: [591] : [9]min [51]sec : time: [591]
count: [592] : [9]min [52]sec : time: [592]
count: [593] : [9]min [53]sec : time: [593]
count: [594] : [9]min [54]sec : time: [594]
count: [595] : [9]min [55]sec : time: [595]
count: [596] : [9]min [56]sec : time: [596]
count: [597] : [9]min [57]sec : time: [597]
count: [598] : [9]min [58]sec : time: [598]
count: [599] : [9]min [59]sec : time: [599]
count: [0] : [0]min [00]sec : time: [600]
count: [1] : [0]min [01]sec : time: [601]
count: [2] : [0]min [02]sec : time: [602]
count: [3] : [0]min [03]sec : time: [603]
count: [4] : [0]min [04]sec : time: [604]
count: [5] : [0]min [05]sec : time: [605]
count: [6] : [0]min [06]sec : time: [606]
count: [7] : [0]min [07]sec : time: [607]
count: [8] : [0]min [08]sec : time: [608]
count: [9] : [0]min [09]sec : time: [609]
count: [10] : [0]min [10]sec : time: [610]
count: [11] : [0]min [11]sec : time: [611]
count: [12] : [0]min [12]sec : time: [612]
count: [13] : [0]min [13]sec : time: [613]
count: [14] : [0]min [14]sec : time: [614]
count: [15] : [0]min [15]sec : time: [615]
count: [16] : [0]min [16]sec : time: [616]
count: [17] : [0]min [17]sec : time: [617]
count: [18] : [0]min [18]sec : time: [618]
count: [19] : [0]min [19]sec : time: [619]
count: [20] : [0]min [20]sec : time: [620]
count: [21] : [0]min [21]sec : time: [621]
count: [22] : [0]min [22]sec : time: [622]
count: [23] : [0]min [23]sec : time: [623]
count: [24] : [0]min [24]sec : time: [624]
count: [25] : [0]min [25]sec : time: [625]
count: [26] : [0]min [26]sec : time: [626]
count: [27] : [0]min [27]sec : time: [627]
count: [28] : [0]min [28]sec : time: [628]
count: [29] : [0]min [29]sec : time: [629]
count: [30] : [0]min [30]sec : time: [630]
Type a Tcl command here
Tcl Console
Messages
count: [595] : [9]min [55]sec : time: [1195]
count: [596] : [9]min [56]sec : time: [1196]
count: [597] : [9]min [57]sec : time: [1197]
count: [598] : [9]min [58]sec : time: [1198]
count: [599] : [9]min [59]sec : time: [1199]
count: [0] : [0]min [00]sec : time: [1200]
count: [0] : [0]min [00]sec : time: [1201]
count: [0] : [0]min [00]sec : time: [1202]
count: [0] : [0]min [00]sec : time: [1203]
count: [0] : [0]min [00]sec : time: [1204]
count: [0] : [0]min [00]sec : time: [1205]
count: [0] : [0]min [00]sec : time: [1206]
count: [0] : [0]min [00]sec : time: [1207]
count: [0] : [0]min [00]sec : time: [1208]
count: [0] : [0]min [00]sec : time: [1209]
count: [0] : [0]min [00]sec : time: [1210]
count: [0] : [0]min [00]sec : time: [1211]
count: [0] : [0]min [00]sec : time: [1212]
count: [0] : [0]min [00]sec : time: [1213]
count: [0] : [0]min [00]sec : time: [1214]
count: [0] : [0]min [00]sec : time: [1215]
count: [0] : [0]min [00]sec : time: [1216]
count: [0] : [0]min [00]sec : time: [1217]
count: [0] : [0]min [00]sec : time: [1218]
count: [0] : [0]min [00]sec : time: [1219]
count: [0] : [0]min [00]sec : time: [1220]
count: [1] : [0]min [01]sec : time: [1221]
count: [2] : [0]min [02]sec : time: [1222]
count: [3] : [0]min [03]sec : time: [1223]
count: [4] : [0]min [04]sec : time: [1224]
count: [5] : [0]min [05]sec : time: [1225]
count: [6] : [0]min [06]sec : time: [1226]
count: [7] : [0]min [07]sec : time: [1227]
count: [8] : [0]min [08]sec : time: [1228]
count: [9] : [0]min [09]sec : time: [1229]
count: [10] : [0]min [10]sec : time: [1230]
count: [11] : [0]min [11]sec : time: [1231]
count: [12] : [0]min [12]sec : time: [1232]
count: [13] : [0]min [13]sec : time: [1233]
count: [14] : [0]min [14]sec : time: [1234]
count: [15] : [0]min [15]sec : time: [1235]
count: [16] : [0]min [16]sec : time: [1236]
count: [17] : [0]min [17]sec : time: [1237]
count: [18] : [0]min [18]sec : time: [1238]
count: [19] : [0]min [19]sec : time: [1239]
count: [20] : [0]min [20]sec : time: [1240]
count: [21] : [0]min [21]sec : time: [1241]
count: [22] : [0]min [22]sec : time: [1242]
count: [23] : [0]min [23]sec : time: [1243]
count: [24] : [0]min [24]sec : time: [1244]
count: [25] : [0]min [25]sec : time: [1245]
count: [26] : [0]min [26]sec : time: [1246]
count: [27] : [0]min [27]sec : time: [1247]
Type a Tcl command here

결과 - 추가과제 2 (2)



0초부터 599초 구간
: count가 1씩 증가하며 분과 초로 나누어져 값이 할당되어 출력함

600초부터 1199초 구간
: 599초가 지나자 000초로 초기화되고 다시 타이머가 동작함

1200초부터 1220초 구간
: rst = 1이 되어 초기화동작으로 인해 모든값에 0이 할당됨

1221초부터 2419초 구간
: rst = 0이 되어 다시 count가 1씩 증가하여 0초 ~ 1199초 구간과 같게 동 종료

이상입니다

THANK YOU