

COMP311: Logic Circuit Design  
Spring 2022, Prof. Taigon Song  
Project 2. Due: May 26 7:59pm [Total:90 points]

[ID : 2018114143]

Name : 권명섭(Kwon Myeong Seop)

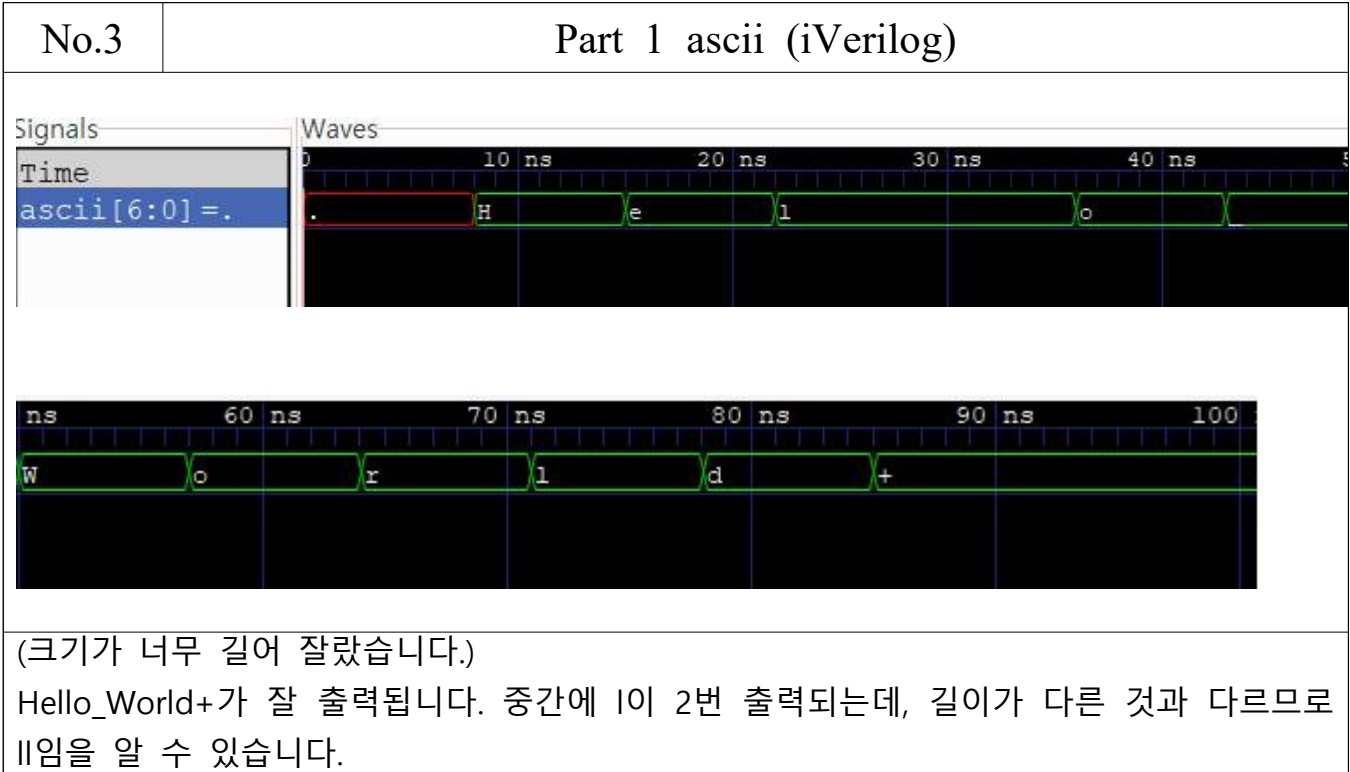
No.	Checksheet item	Done?	Points
1.	ascii code (part 1)	Hello_world+	10
2.	Base64 code (part 2)	hz5tBmxY39PO/m67g	10
3.	Part 1 ascii (iVerilog)	Y	20
4.	Part 2 6-bit printing	Y	25
5.	Part 2 Base64 printing	Y	15
6.	Synthesizable	N	0

**1. 실행 결과.....2~3**

**2. 모듈 설명.....4~6**

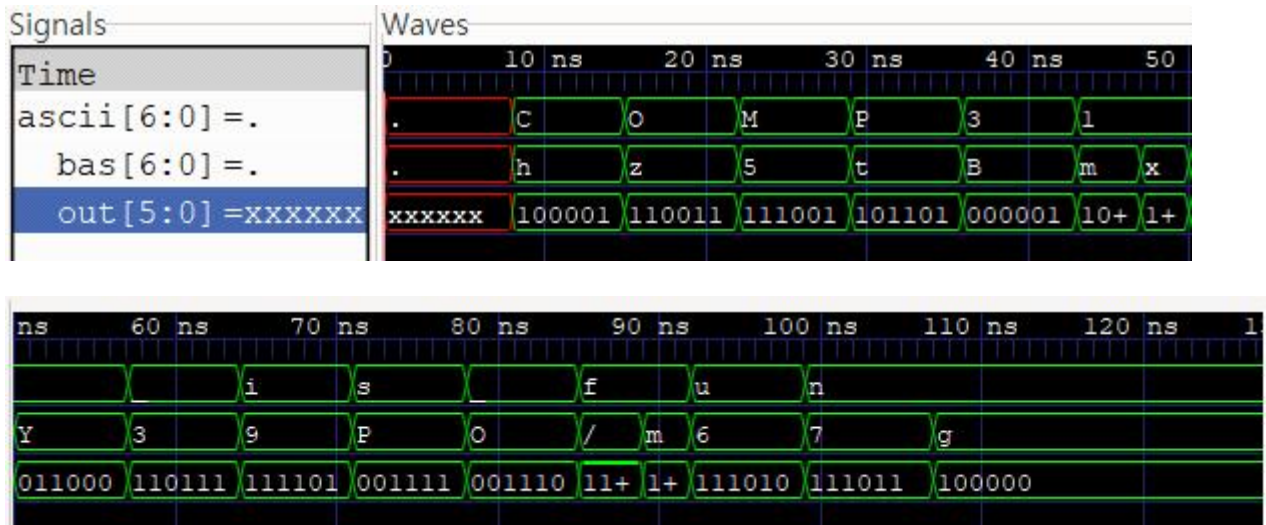
**3. 코드.....7~16**

# 1. 실행 결과



```
C:\work\project2>run.bat
C:\work\project2>del output*
C:\work\project2>iverilog -o output.vvp rtl/*.v
C:\work\project2>vvp output.vvp
VCD info: dumpfile output.vcd opened for output.
Hello_World+rtl/p1_tb.v:18: $finish called at 1500 (100ps)
C:\work\project2>gtkwave output.vcd
GTKWave Analyzer v3.3.108 (w)1999-2020 BSI
[0] start time.
[150000] end time.
```

cmd를 이용한 출력도 잘 나옴을 알 수 있습니다.  
\$display문을 이용하면 한 글자 출력 후 줄 바꿈이 일어나서 \$write를 사용했습니다.



(크기가 너무 길어 잘랐습니다.)

아래 out이 6-bit 출력, 중간 bas가 base64입니다.

bitstream의 길이가 98이라 2자리가 마지막에 남는데 0 4개를 추가해서 g가 나옵니다.

이에 따라 bas는 hz5tBmxY39PO/m67g가 잘 출력됩니다.

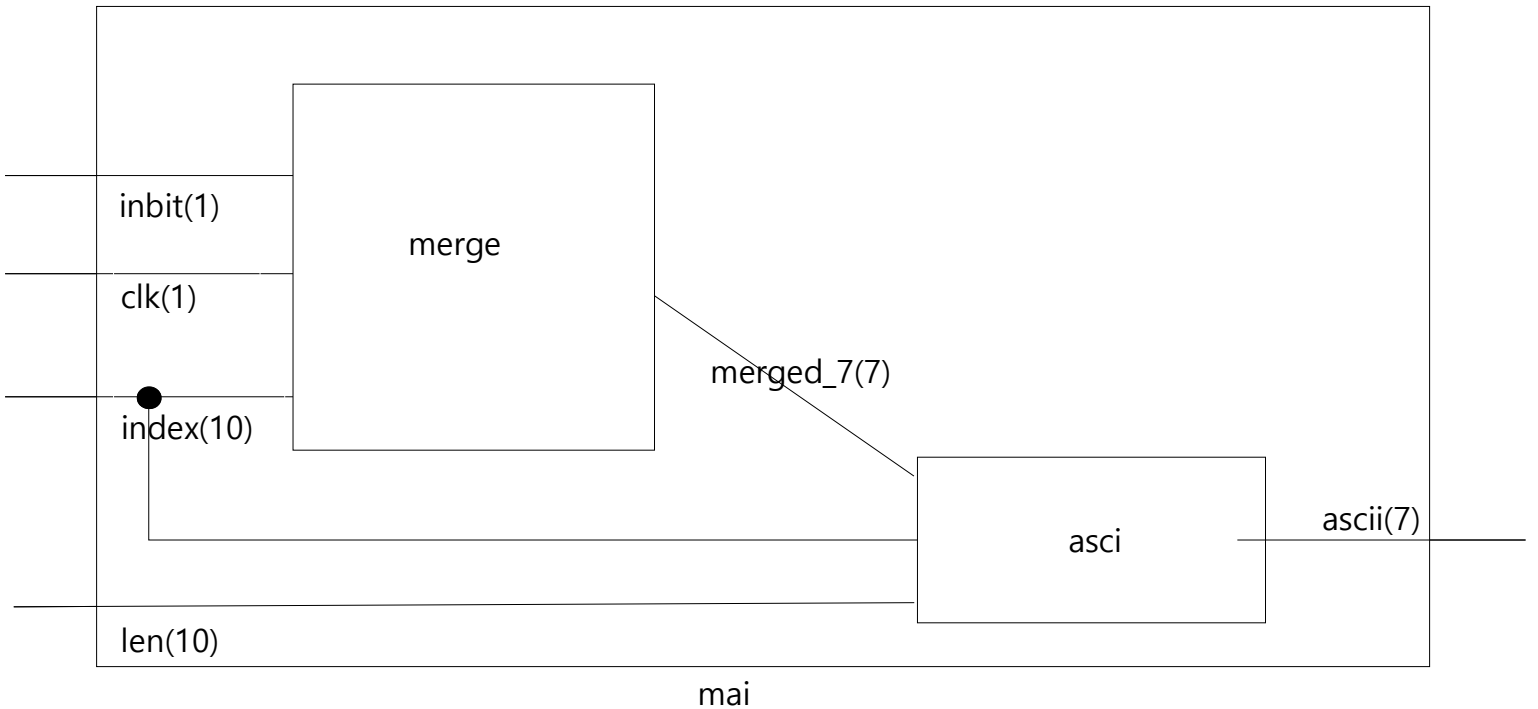
```
C:\work\project2>run.bat
C:\work\project2>del output*
C:\work\project2\output*을(를) 찾을 수 없습니다.
C:\work\project2>iverilog -o output.vvp rtl/*.v
C:\work\project2>vvp output.vvp
VCD info: dumpfile output.vcd opened for output.
hz5tBmxY39PO/m67g rtl/p2_tb.v:20: $finish called at 1500 (100ps)
C:\work\project2>gtkwave output.vcd
GTKWave Analyzer v3.3.108 (w)1999-2020 BSI
[0] start time.
[150000] end time.
```

cmd에서도 잘 나옵니다.

## 2. 모듈 설명

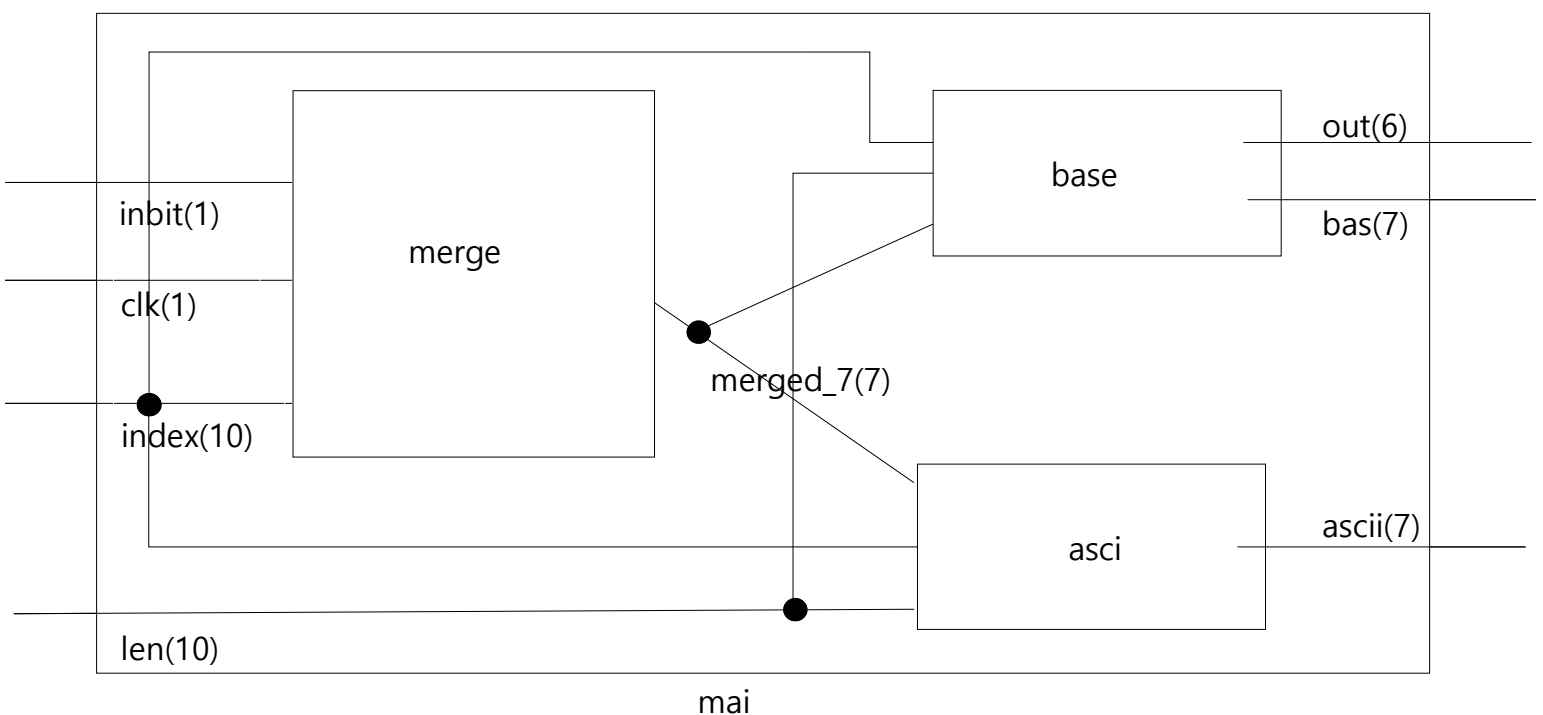
### 2.1 모듈 관계도

#### 2.1-1 part 1



tb에서 호출하는 유일한 모듈인 mai 안에 다음과 같이 연결되어 있습니다.  
괄호 안의 수는 비트 수이며, 모듈 기준 왼쪽이 입력, 오른쪽이 출력입니다.

#### 2.1-2 part2



part1에서 base 모듈이 추가되었습니다.

## 2.2 모듈 설명

in,output에 대해서는 input은 초록색, 아웃풋은 보라색으로 하였으며, 그것들을 만든 이유를 적었습니다.

mai- 모든 모듈을 감싸서 tb에서 볼 때 하나의 모듈만 보이게 했습니다.

inbit(1) : 1bit bitstream input입니다. 기본 조건이었습니다.

clk:(1) : 일정한 간격으로 신호를 받기 위해 clk을 썼습니다.

index(10) : 내부 모듈 셋에서 다 쓰입니다.

ascii : 아웃풋입니다. 합쳐진 7bit입니다.

merge- 1bit씩 신호를 받아(inbit) merged\_7에 저장합니다. 항상 유의미한 값은 아니지만, 다른 모듈에서 7비트가 되었을 때만 이 값을 사용하기 때문에 문제없습니다.

input, output에 관한 설명:

inbit(1) ,clk:(1) : mai 모듈과 같은 이유입니다.

index(10) : 값을 부여하는 데 필요하고, 다른 모듈에서도 계속 쓰여서, 현재 처리 중인 자릿수로 썼습니다.

merged\_7(7) : 아웃풋입니다. 합쳐진 7bit입니다. 항상 비트를 갱신해서 의미없는 값일때도 있습니다.

구체적인 원리 : clk이 posedge일 때 값을 기록해서, 단순히 인풋이 들어올 때마다 기록합니다.

ascii- merge에서 가져온 7bit 신호를 output으로 내보내고, cmd와 wave상에서 출력합니다. part2에서는 base64랑 같이 출력되면 안 되므로, cmd 출력은 주석처리 했습니다.

merged\_7(7) : 당연히 합쳐진 7bit짜리 인풋이 필요합니다.

index(10) : 현재 7bit가 완성됐는지 알려면 필요합니다.

len(10) : 맨 처음에 신호가 안 들어왔을 때 출력하는 사항이 있어 예외 처리에 썼습니다.

ascii(7) : 당연히 필요한 아웃풋입니다.

구체적인 원리 : 현재 처리 중인 자릿수(index)를 봐서 7bit째가 되면 가져와 그것을 아웃풋으로 내보내고, 출력합니다. 출력은 ascii라 바로 할 수 있습니다.

base(part2에서만 존재)- merge에서 가져온 7bit 신호를 6비트 base64로 변환해서, output으로 내보내고(이진 형태와 문자 형태), 이를 cmd(문자만)와 wave(문자와 이진)상에서 출력합니다. 그리고, 비트가 6개가 쌓이면 한 사이클 내에서 2번 출력합니다. 입력이 6bit로 떨어지지 않으면 0을 6bit가 되게 뒤에 채워서 마지막 base64코드를 만듭니다.

merged\_7(7) : 당연히 합쳐진 7bit짜리 인풋이 필요합니다.

index(10) : 현재 7bit가 완성됐는지 알려면 필요합니다. (ascii 모듈과 동일)

len(10) : 맨 처음에 신호가 안 들어왔을 때 출력하는 사항이 있어 예외 처리에 썼습니다. (ascii 모듈과 동일)

bas(7) : 6비트에 해당하는 문자(ascii로 표현)를 나타내는 아웃풋입니다.

out(6) : 6비트 이진 아웃풋입니다.

구체적인 원리 : 7bit가 들어오면, 이전 잉여 비트가 최대 5개까지 있을 수 있으므로 이전 비트 5개랑 새로 들어온 7개의 비트를 합칩니다. 그 후 6개를 떼어내서 base64로 처리한 후 잉여 비트 수를 기록합니다. 이때 잉여 비트 수가 6개가 되면 같은 사이클 안에서 base64로 한 번 더 처리합니다. 한 번 더 처리할 때 cmd출력은 바로 해도 문제없지만, wave 출력은 텀을 줘야 해서(안 그러면 출력 하나가 무시 되므로) 현재 처리 중인 자릿수(index)를 참고해 텀을 주고 값을 바꾸게 했습니다.

cmd 출력은 6bit에 해당하는 숫자 모든 경우를 case 문을 이용해서 write문을 설정하였고, wave 출력도 동일하게 6bit에 해당하는 숫자 모든 경우를 case 문을 이용해서 해당하는 문자를 ascii 코드로 출력할 수 있게 ascii라는 다른 변수에 값을 일일이 대입해서 만들었습니다.

입력 비트의 수가 6의 배수가 아닐 수 있으므로, 입력이 다 들어온 후 시간을 조금 두고 마지막 남은 비트들을 6비트가 되게 0을 뒤에 채워서 마지막 base 64 코드를 만듭니다.

## 3. 코드

### 3.1 part1

#### 3.1-1 tb

```
p1_tb.v x p1.v x
1  `timescale 1ns/100ps
2
3  module asc_tb ();
4      reg clk;
5      reg [83:0] bitstream;
6      reg [9:0] index; //bitstream에서 현재처리하고 있는 자리 수
7      reg [9:0] len; //bitstream의 길이
8
9      wire[6:0] ascii;
10
11     main mai(ascii,clk,bitstream[index],index,len);
12
13     initial begin
14         clk =1;
15         bitstream=84'b10010001100101110110011011001101
16         index=$bits(bitstream);
17         len=$bits(bitstream);
18         #150 $finish;
19     end
20
21     always begin
22         #0.5 clk = ~clk;
23     end
24
25     always begin
26         #1 index=index-1;
27     end
28
29     initial begin
30         $dumpfile("output.vcd");
31         $dumpvars(0);
32     end
33
34 endmodule
```

## 3.1-2 module

```
p1_tb.v x p1.v x
1  module main(ascii,clk,inbit,index,len);
2
3      output [6:0] ascii;
4
5      input clk;
6      input inbit;
7      input [9:0] index;//bitstream에서 현재처리하고 있는 자리 수
8      input [9:0] len; //bitstream의 길이
9
10     wire [6:0] merged_7; //7개의 비트가 합쳐진 것
11     mergeto7 merge(merged_7,clk,inbit,index);
12     toascii asci(ascii,merged_7,index,len);
13
14 endmodule
15
16
17
18 module mergeto7(merged_7,clk,inbit,index); // posedge마다 1bit씩 merged_7에 값을 가져 오는 모듈
19     output reg [6:0] merged_7; //7개의 비트가 합쳐진 것
20
21     input clk;
22     input inbit;
23     input [9:0] index;//bitstream에서 현재처리하고 있는 자리 수
24
25
26
27     always @(posedge clk) begin
28         if (index !=10'b1111111111) begin //index가 0밀으로 떨어지면 실행 안됨
29             merged_7[index%7]=inbit;
30         end
31     end
32
33 endmodule
34
35 module toascii(ascii,merged_7,index,len); //merged_7이 7비트가 완성되면 출력하는 모듈
36     output reg [6:0] ascii;
37     input [6:0] merged_7;
38     input [9:0] index;
39     input [9:0] len;
40
41     wire [9:0] indexplus;
42
43     assign indexplus=index+1;
44
45     always @(index) begin
46         if(indexplus%7==0&&indexplus<10'b1000000000&&indexplus!=len) begin
47             ascii=merged_7;
48             $write("%c",merged_7); //가로로 표시하기 위해 write를 사용
49         end
50         else begin
51             end
52     end
53
54 endmodule
```



## 3.2 part2

### 3.2-1 tb

```
p2_tb.v x p2.v x
1 `timescale 1ns/100ps
2
3 module asc_tb ();
4     reg clk;
5     reg [97:0] bitstream;
6     reg [9:0] index; //bitstream에서 현재처리하고 있는 자리 수
7     reg [9:0] len; //bitstream의 길이
8
9     wire[6:0] ascii;
10    wire [6:0] bas;
11    wire[5:0] out;
12
13    main mai(ascii,bas,out,clk,bitstream[index],index,len)
14
15    initial begin
16        clk = 1;
17        bitstream=98'b100001110011111100110110101000001100110
18        index=$bits(bitstream);
19        len=$bits(bitstream);
20        #150 $finish;
21    end
22
23    always begin
24        #0.5 clk = ~clk;
25    end
26
27    always begin
28        #1 index=index-1;
29    end
30
31    initial begin
32        $dumpfile("output.vcd");
33        $dumpvars(0);
34    end
35
36 endmodule
```

## 3.2-2 module

```
p2_tb.v p2.v
1  module main(ascii,bas,out,clk,inbit,index,len);
2
3      output [6:0] ascii;
4      output [6:0] bas;
5      output [5:0] out;
6
7      input clk;
8      input inbit;
9      input [9:0] index; //bitstream에서 현재처리하고 있는 자리 수
10     input [9:0] len; //bitstream의 길이
11
12     wire [6:0] merged_7; //7개의 비트가 합쳐진 것
13     mergeto7 merge(merged_7,clk,inbit,index);
14     toascii ascii(ascii,merged_7,index,len);
15     tobase base(out,bas,merged_7,index,len);
16
17 endmodule
18
19
20
21 module mergeto7(merged_7,clk,inbit,index); // posedge마다 1bit씩 merged_7에 값을 가져 오는 모듈
22     output reg [6:0] merged_7; //7개의 비트가 합쳐진 것
23
24     input clk;
25     input inbit;
26     input [9:0] index; //bitstream에서 현재처리하고 있는 자리 수
27
28
29
30     always @(posedge clk) begin
31         if (index != 10'b1111111111) begin //index가 0밑으로 떨어지면 실행 안됨
32             merged_7[index%7]=inbit;
33         end
34     end
35
36 endmodule
```

```

37
38 module toascii(ascii,merged_7,index,len); //merged_7이 7비트가 완성되면 출력하는 모듈
39     output reg [6:0] ascii;
40     input [6:0] merged_7;
41     input [9:0] index;
42     input [9:0] len;
43
44     wire [9:0] indexplus;
45
46     assign indexplus=index+1;
47
48     always @(index) begin
49         if(indexplus%7==0&&indexplus<10'b1000000000&&indexplus!=len) begin
50             ascii=merged_7;
51             //$write("%c",merged_7); //가로로 표시하기 위해 write를 사용
52         end
53         else begin
54             end
55     end
56
57 endmodule
58
59 module tobase(out,bas,merged_7,index,len); //merged_7이 6비트가 완성되면 출력하는 모듈
60     input [6:0] merged_7;
61     input [9:0] index;
62     input [9:0] len;
63     output reg [5:0] out;
64     output [6:0] bas;
65
66     wire [9:0] indexplus;
67
68     reg [5:0] result;
69     reg [11:0] base_char=0;
70     reg [3:0] remain=0; //6비트씩 끊기고, 남은 잉여 비트 수
71     reg [6:0] ascii; //base64에 해당하는 문자를 ascii로 표현한 것
72     reg printmore; //waveform에 한번더 출력해야 할때 1로 바뀜
73
74     assign indexplus=index+1;
75     assign bas=ascii;
76

```

```

76
77 always @(index) begin
78     if (indexplus%7==0&&indexplus!=len&&indexplus<10'b1000000000) begin
79         base_char={base_char[4],base_char[3],base_char[2],base_char[1],base_char[0],merged_7}; //기존의 잉여 비트와 새로받은 7비트를 합침
80
81         case(remain)
82         0 : begin
83             result = {base_char[6],base_char[5],base_char[4],base_char[3],base_char[2],base_char[1]};
84             remain=1;
85             end
86         1 : begin
87             result = {base_char[7],base_char[6],base_char[5],base_char[4],base_char[3],base_char[2]};
88             remain=2;
89             end
90         2 : begin
91             result = {base_char[8],base_char[7],base_char[6],base_char[5],base_char[4],base_char[3]};
92             remain=3;
93             end
94         3 : begin
95             result = {base_char[9],base_char[8],base_char[7],base_char[6],base_char[5],base_char[4]};
96             remain=4;
97             end
98         4 : begin
99             result = {base_char[10],base_char[9],base_char[8],base_char[7],base_char[6],base_char[5]};
100            remain=5;
101            end
102         5 : begin
103             result = {base_char[11],base_char[10],base_char[9],base_char[8],base_char[7],base_char[6]};
104             remain=6;
105             end
106         default : ;
107         endcase
108         out=result;
109         case(result) //cmd 출력
110
111
112
113
114
115
116
117         case(result) //waveform 출력
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455

```

//cmd 출력과 //waveform 출력이라고 되어 있는 부분은 겹쳐 있는데, 엄청 길고 반복적으로 여러 개 있어서 따로 빼냈습니다.



//cmd 출력

```
403 case(result) //cmd 출력
404 0 : $write("A");
405 1 : $write("B");
406 2 : $write("C");
407 3 : $write("D");
408 4 : $write("E");
409 5 : $write("F");
410 6 : $write("G");
411 7 : $write("H");
412 8 : $write("I");
413 9 : $write("J");
414 10 : $write("K");
415 11 : $write("L");
416 12 : $write("M");
417 13 : $write("N");
418 14 : $write("O");
419 15 : $write("P");
420 16 : $write("Q");
421 17 : $write("R");
422 18 : $write("S");
423 19 : $write("T");
424 20 : $write("U");
425 21 : $write("V");
426 22 : $write("W");
427 23 : $write("X");
428 24 : $write("Y");
429 25 : $write("Z");
430 26 : $write("a");
431 27 : $write("b");
432 28 : $write("c");
433 29 : $write("d");
434 30 : $write("e");
435 31 : $write("f");
436 32 : $write("g");
437 33 : $write("h");
438 34 : $write("i");
439 35 : $write("j");
440 36 : $write("k");
441 37 : $write("l");
442 38 : $write("m");
443 39 : $write("n");
444 40 : $write("o");
```

```


436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470

32 : $write("g");
33 : $write("h");
34 : $write("i");
35 : $write("j");
36 : $write("k");
37 : $write("l");
38 : $write("m");
39 : $write("n");
40 : $write("o");
41 : $write("p");
42 : $write("q");
43 : $write("r");
44 : $write("s");
45 : $write("t");
46 : $write("u");
47 : $write("v");
48 : $write("w");
49 : $write("x");
50 : $write("y");
51 : $write("z");
52 : $write("0");
53 : $write("1");
54 : $write("2");
55 : $write("3");
56 : $write("4");
57 : $write("5");
58 : $write("6");
59 : $write("7");
60 : $write("8");
61 : $write("9");
62 : $write("+");
63 : $write("/");

default : ;
endcase

```

//wave 출력

471		<code>case(result) //waveform 출력</code>
472		<code>0 : ascii= 7'b1000001;</code>
473		<code>1 : ascii= 7'b1000010;</code>
474		<code>2 : ascii= 7'b1000011;</code>
475		<code>3 : ascii= 7'b1000100;</code>
476		<code>4 : ascii= 7'b1000101;</code>
477		<code>5 : ascii= 7'b1000110;</code>
478		<code>6 : ascii= 7'b1000111;</code>
479		<code>7 : ascii= 7'b1001000;</code>
480		<code>8 : ascii= 7'b1001001;</code>
481		<code>9 : ascii= 7'b1001010;</code>
482		<code>10 : ascii= 7'b1001011;</code>
483		<code>11 : ascii= 7'b1001100;</code>
484		<code>12 : ascii= 7'b1001101;</code>
485		<code>13 : ascii= 7'b1001110;</code>
486		<code>14 : ascii= 7'b1001111;</code>
487		<code>15 : ascii= 7'b1010000;</code>
488		<code>16 : ascii= 7'b1010001;</code>
489		<code>17 : ascii= 7'b1010010;</code>
490		<code>18 : ascii= 7'b1010011;</code>
491		<code>19 : ascii= 7'b1010100;</code>
492		<code>20 : ascii= 7'b1010101;</code>
493		<code>21 : ascii= 7'b1010110;</code>
494		<code>22 : ascii= 7'b1010111;</code>
495		<code>23 : ascii= 7'b1011000;</code>
496		<code>24 : ascii= 7'b1011001;</code>
497		<code>25 : ascii= 7'b1011010;</code>
498		<code>26 : ascii= 7'b1100001;</code>
499		<code>27 : ascii= 7'b1100010;</code>
500		<code>28 : ascii= 7'b1100011;</code>
501		<code>29 : ascii= 7'b1100100;</code>
502		<code>30 : ascii= 7'b1100101;</code>
503		<code>31 : ascii= 7'b1100110;</code>
504		<code>32 : ascii= 7'b1100111;</code>
505		<code>33 : ascii= 7'b1101000;</code>
506		<code>34 : ascii= 7'b1101001;</code>
507		<code>35 : ascii= 7'b1101010;</code>
508		<code>36 : ascii= 7'b1101011;</code>
509		<code>37 : ascii= 7'b1101100;</code>
510		<code>38 : ascii= 7'b1101101;</code>



```

511      39 : ascii= 7'b1101110;
512      40 : ascii= 7'b1101111;
513      41 : ascii= 7'b1110000;
514      42 : ascii= 7'b1110001;
515      43 : ascii= 7'b1110010;
516      44 : ascii= 7'b1110011;
517      45 : ascii= 7'b1110100;
518      46 : ascii= 7'b1110101;
519      47 : ascii= 7'b1110110;
520      48 : ascii= 7'b1110111;
521      49 : ascii= 7'b1111000;
522      50 : ascii= 7'b1111001;
523      51 : ascii= 7'b1111010;
524      52 : ascii= 7'b0110000;
525      53 : ascii= 7'b0110001;
526      54 : ascii= 7'b0110010;
527      55 : ascii= 7'b0110011;
528      56 : ascii= 7'b0110100;
529      57 : ascii= 7'b0110101;
530      58 : ascii= 7'b0110110;
531      59 : ascii= 7'b0110111;
532      60 : ascii= 7'b0111000;
533      61 : ascii= 7'b0111001;
534      62 : ascii= 7'b0101011;
535      63 : ascii= 7'b0101111;
536  default : ;
537  endcase

```