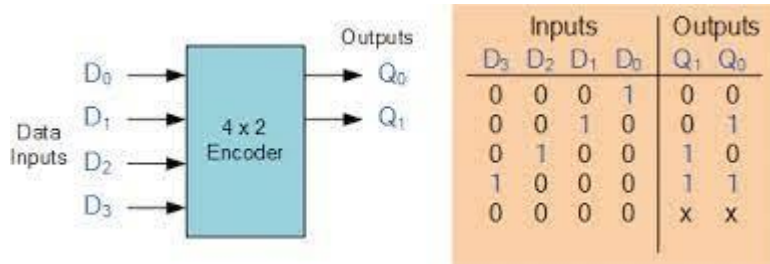


< 실습과제 4 >

2017253041_홍성우

1.

(1)



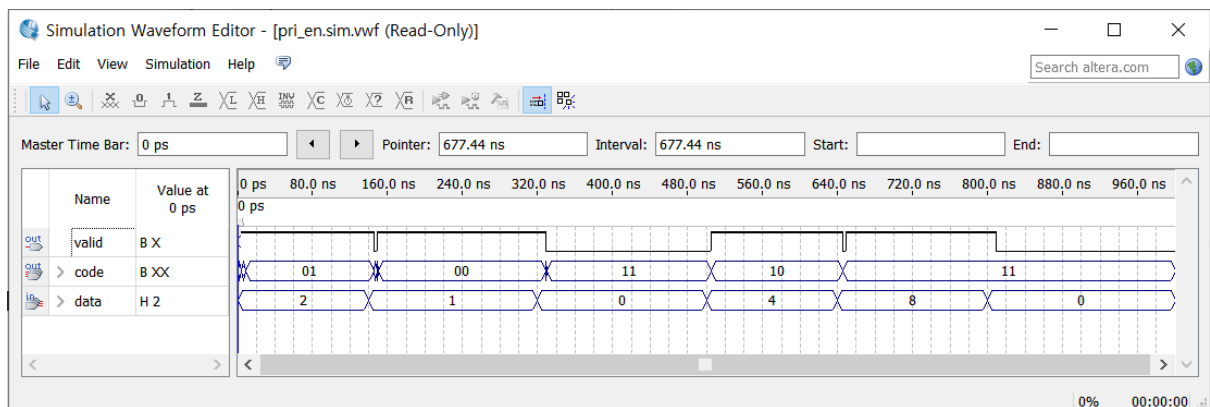
인코더와 다르게 한 입력만이 1인 제한조건이 없으며, 여러 개의 입력이 동시에 1인 경우에는 가장 우선순위가 높은 입력에 대한 코드를 출력한다. 또, 입력에 1인 bit가 존재할 때 valid = 1 이다.

(2)

```

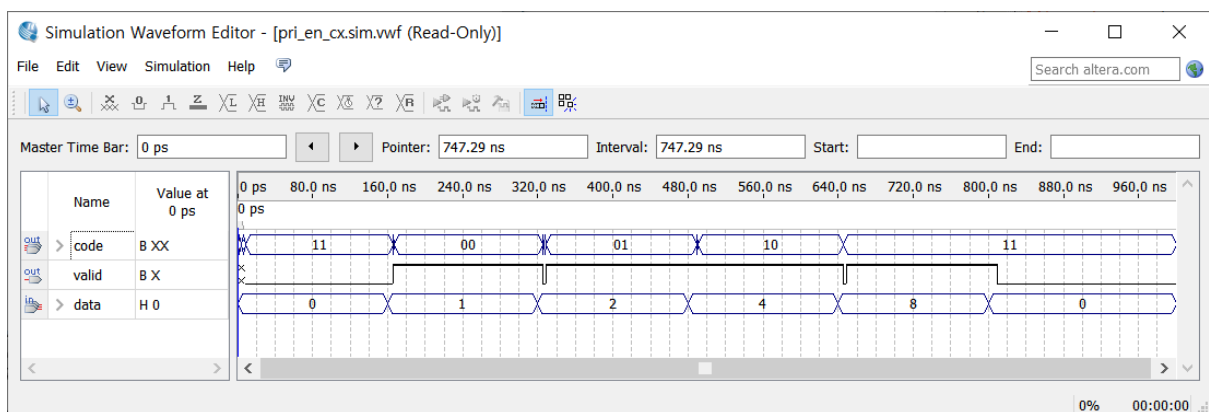
1 module pri_en(code,data,valid);
2   output [1:0] code;
3   input [3:0] data;
4   output valid;
5   reg [1:0] code;
6
7   always @(data) begin
8     if (data[0]) code = 0;
9     else if (data[1]) code = 1;
10    else if (data[2]) code = 2;
11    else if (data[3]) code = 3;
12    else code = 2'bx;
13  end
14  assign valid = | data;
15 endmodule

```



(3)

```
pri_en_cx.v  Compilation Report - pri_en_cx
1  module pri_en_cx(code,data,valid);
2  output [1:0] code;
3  output valid;
4  input [3:0] data;
5  reg [1:0] code;
6
7  always @(data) begin
8  case(data)
9  4'bxxx1 : code = 0;
10 4'bxx10 : code = 1;
114'bx100 : code = 2;
124'b1000 : code = 3;
13default : code = 2'bx;
14endcase
15end
16
17 assign valid = (data != 0);
18 endmodule
```

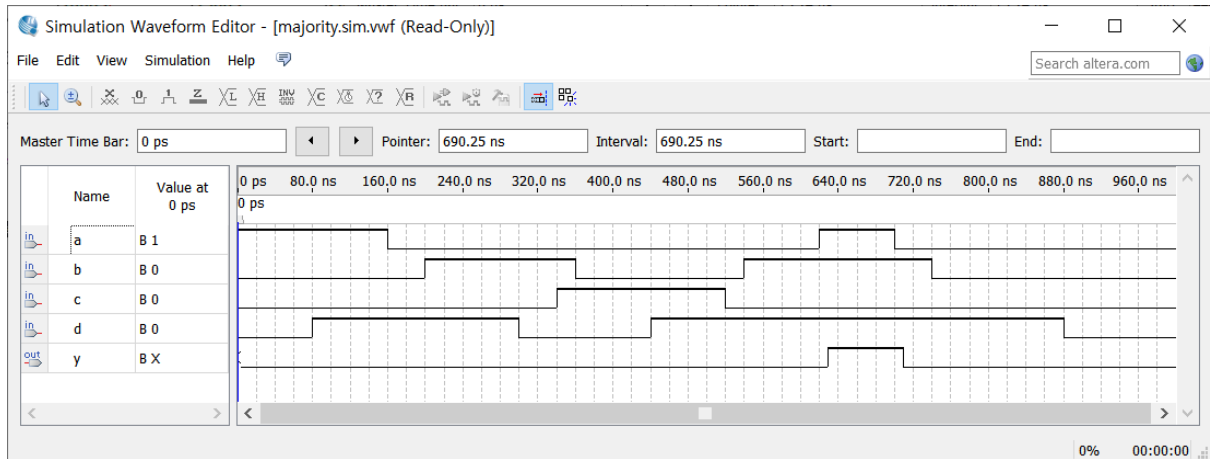


case문은 0,1,x,z 의 값을 정확히 비교하므로 여기서는 사용할 수 없음

2.

(1)

```
majority.v  Compilation Report - majority
1  module majority(y,a,b,c,d);
2  output y;
3  input a,b,c,d;
4  reg y;
5
6  always @(*) begin
7  case({a,b,c,d})
8  7,11,13,14,15 : y = 1;
9  default : y = 0;
10 endcase
11 end
12 endmodule
```



(2)

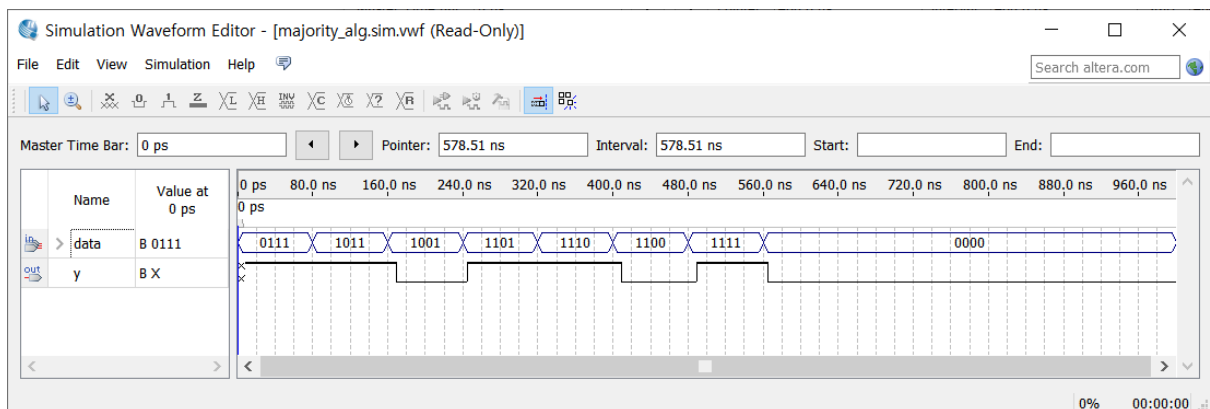
majority_alg.v

Compilation Report - majority_alg

```

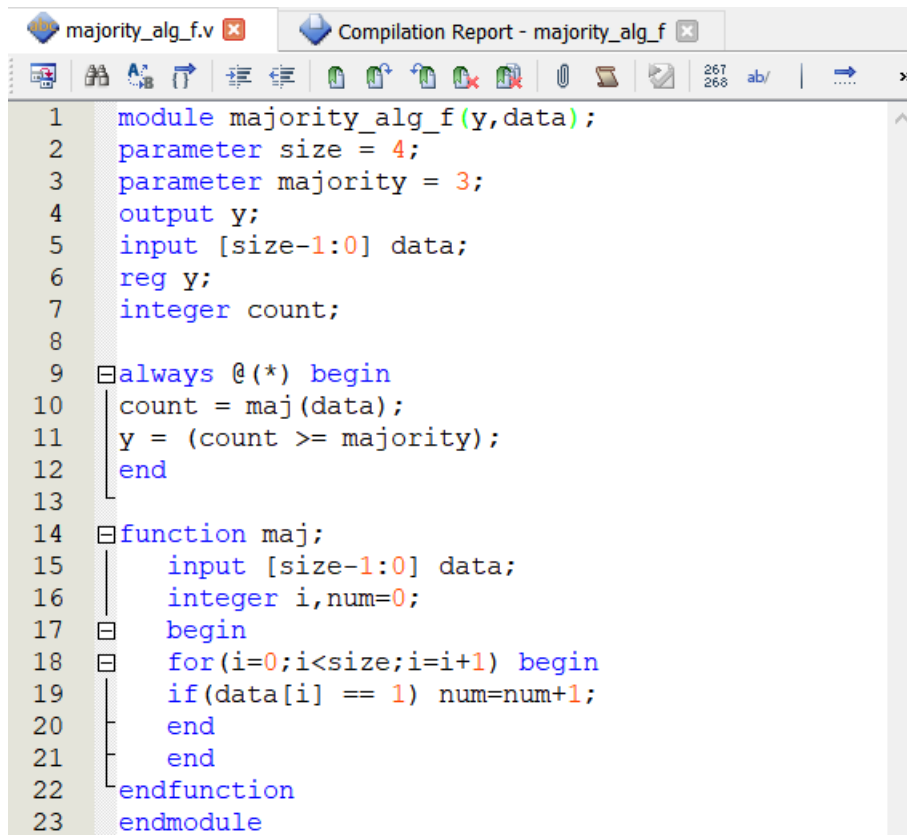
1 module majority_alg(y,data);
2   parameter size = 4;
3   parameter majority = 3;
4   output y;
5   input [size-1:0] data;
6   reg y;
7   integer count,i;
8
9   always @(*) begin
10    count = 0;
11    for(i=0;i<size;i=i+1) begin
12      if(data[i]==1) count=count+1;
13    end
14    y = (count >= majority);
15  end
16 endmodule

```



3.

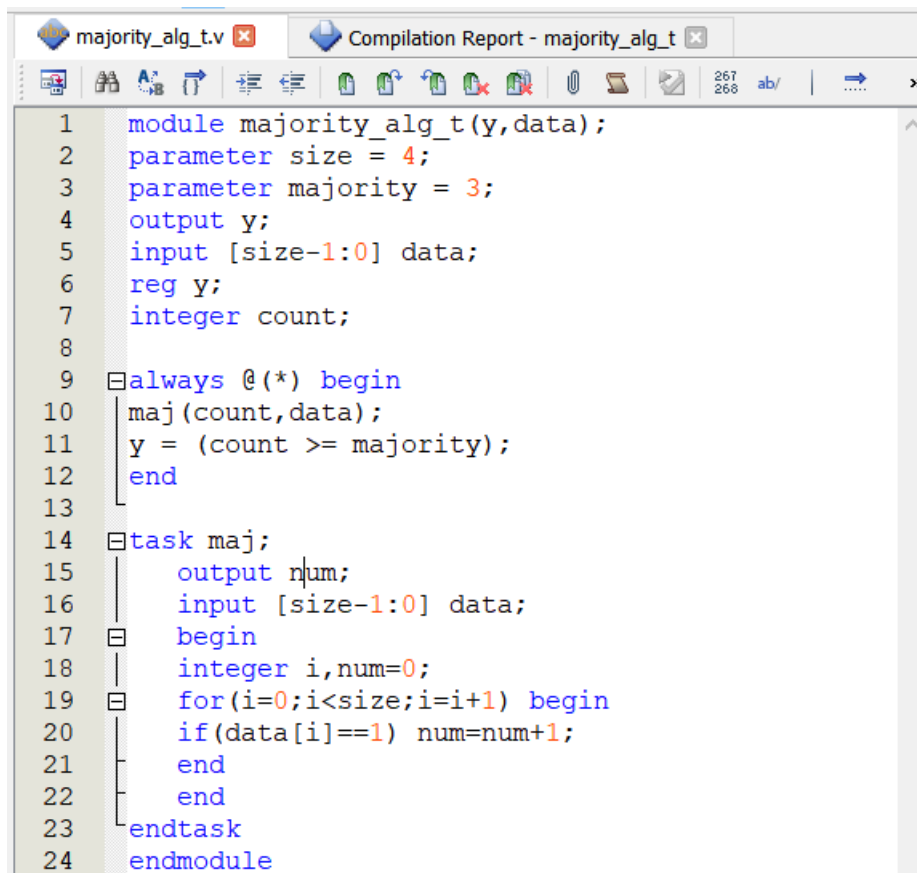
(1)



The screenshot shows a Verilog code editor with two tabs: 'majority_alg_f.v' and 'Compilation Report - majority_alg_f'. The code defines a module 'majority_alg_f' with an output 'y' and an input 'data' of size 4. It includes parameters 'size = 4' and 'majority = 3'. An 'always' block calls a 'maj' function to calculate the majority. The 'maj' function iterates through the 'data' array, counting the number of '1's. The code is as follows:

```
1  module majority_alg_f(y,data);
2  parameter size = 4;
3  parameter majority = 3;
4  output y;
5  input [size-1:0] data;
6  reg y;
7  integer count;
8
9  always @(*) begin
10     count = maj(data);
11     y = (count >= majority);
12 end
13
14 function maj;
15     input [size-1:0] data;
16     integer i,num=0;
17     begin
18     for(i=0;i<size;i=i+1) begin
19         if(data[i] == 1) num=num+1;
20     end
21     end
22 endfunction
23 endmodule
```

(2)



```
1 module majority_alg_t(y,data);
2 parameter size = 4;
3 parameter majority = 3;
4 output y;
5 input [size-1:0] data;
6 reg y;
7 integer count;
8
9 always @(*) begin
10     maj(count,data);
11     y = (count >= majority);
12 end
13
14 task maj;
15     output num;
16     input [size-1:0] data;
17     begin
18         integer i,num=0;
19         for(i=0;i<size;i=i+1) begin
20             if(data[i]==1) num=num+1;
21         end
22     end
23 endtask
24 endmodule
```

4.

이번 실습과제에서는 논리회로설계 시간에 배웠던 인코더,디코더,멀티플렉서 등 베릴로그를 통하여 실습을 해보니 나름 재미있었다.