1. 구현 내용

구현한 verilog file은 bitcounter.v와 tb_bitcounter.v 입니다.

1) bitcounter.v

4 bit shift register에 입력 받은 4 bit 숫자에서 1의 개수가 몇 개인지 카운트하는 bitcounter를 구현하였습니다. shift register와 counter의 기능을 활용하여 최종적으로 bitcounter 모듈을 구현하였습니다.

```
module bitcounter(w, LA, EA, clk, Data, rst, A, B, LB, EB, z, a0);
input w, LA, EA, clk, rst, LB, EB;
input [3:0] Data;


output reg [3:0] A;
output reg [2:0] B;
output reg a0;
output z;


assign z = ~(A[0] | A[1] | A[2] | A[3]);

always @(posedge clk or negedge rst)
begin
    if (!rst) begin
        A <= 4'b0;
    end



   else begin
       if(LA) A <= Data;
       else if(EA) begin
           a0 <= A[0];
           A[0] <= A[1];
           A[1] <= A[2];
           A[2] <= A[3];
           A[3] <= w;
       end
   end
end
```

```
always @(posedge clk or negedge rst or posedge a0)
begin
    if(!rst) begin
        B <= 3'b0;
    end
    else if(!z) begin
        if (LB) B <= 3'b0;
        else if(EA) begin
            if(a0) B <= B + 1;
        end
    end
end
endmodule
```

```
module bitcounter(w, LA, EA, clk, Data, rst, A, B, LB, EB, z, a0);
// input과 output을 각각 정의함
input w, LA, EA, clk, rst, LB, EB;
input [3:0] Data;
output reg [3:0] A;
output reg [2:0] B;
output reg a0;
output z;

// z에 A[0]부터 A[3]까지를 nor 연산을 통해 값을 할당
assign z = ~(A[0] | A[1] | A[2] | A[3]);

always @(posedge clk or negedge rst)        // shift register
begin
    if (!rst) begin
        A <= 4'b0;     // 리셋이 0이면 A에 0 대입
    end
    else begin
        if(LA)
            A <= Data;         // LA가 1이 되면 A에 Data 넣음
        else if(EA) begin            // EA가 1이면 A[0]값을 a0에 넣고 나머지 shift
            a0 <= A[0];
            A[0] <= A[1];
            A[1] <= A[2];
```

```
            A[2] <= A[3];
            A[3] <= w;
        end
    end
end

always @(posedge clk or negedge rst or posedge a0) // counter
begin
    if(!rst) begin
        B <= 3'b0;              // 리셋이 0이 되면 B에 0 대입
    end
    else if(!z) begin
        if (LB)
            B <= 3'b0;          // LB가 1이면 B에 0 대입
        else if(EA) begin       // 그렇지 않고 EA가 1이고, a0가 1이면 B를 1씩 증가 시킴
            if(a0)              // shfit된 bit가 1이면 B를 1씩 증가 시킴
                B <= B + 1;
        end
    end
end
endmodule
```

2) tb_bitcounter.v

bitcounter 모듈의 기능을 테스트하기 위한 테스트벤치 파일입니다.

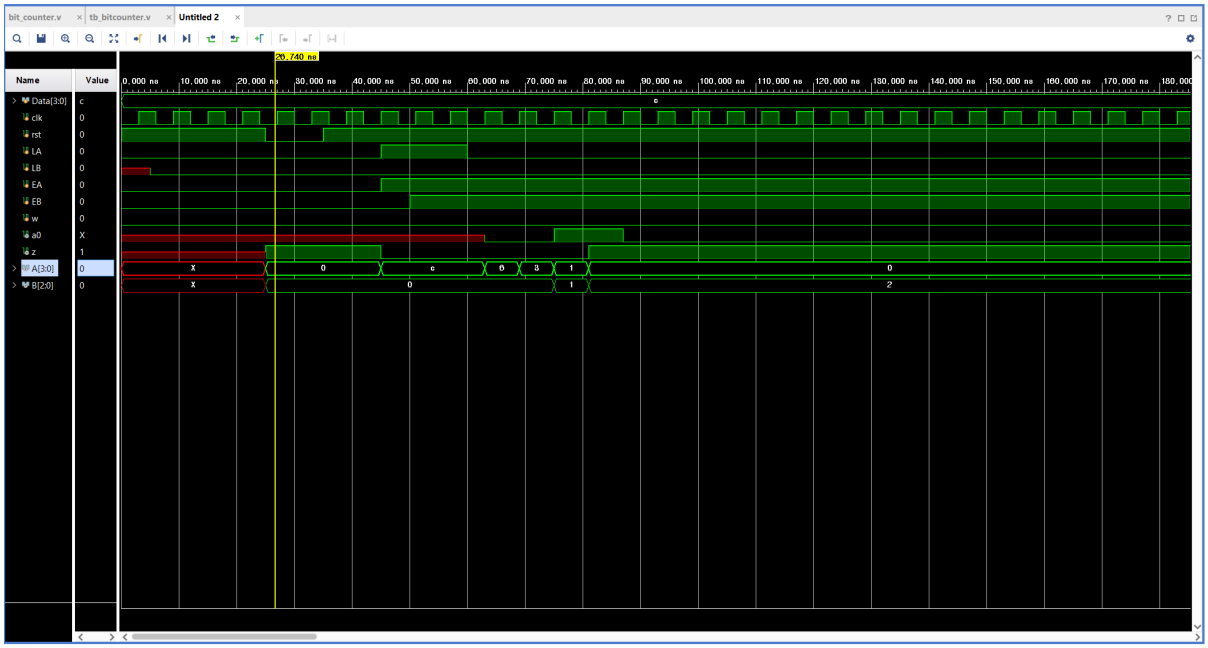
```
module tb_bitcounter();
reg [3:0] Data;
reg clk, rst, LA, LB, EA, EB, w;
wire a0, z;
wire [3:0] A;
wire [2:0] B;
bitcounter mybitcount(w, LA, EA, clk, Data, rst, A, B, LB, EB, z, a0);

always #3 clk = ~clk;
initial begin
clk = 0; rst = 1; w = 0; Data = 4'b1011; LA = 4'b0000; EB = 0; EA = 0;
#5 LB = 0; EB = 0;
#20 rst = 0;
#10 rst = 1;
#10 LA = 1; EA = 1;
#5 EB = 1;
#10 LA = 0;
end
endmodule
```

```
module tb_bitcounter();
reg [3:0] Data;
reg clk, rst, LA, LB, EA, EB, w;
wire a0, z;
wire [3:0] A;
wire [2:0] B;
bitcounter mybitcount(w, LA, EA, clk, Data, rst, A, B, LB, EB, z, a0);

always #3 clk = ~clk; // clk은 3 단위로 바뀌게 설정함
initial begin
clk = 0; rst = 1; w = 0; Data = 4'b1011; LA = 4'b0000; EB = 0; EA = 0;
// 리셋은 초기 1로, Data는 7로 설정한 뒤 1이 3개인 출력값을 확인하도록 함
#5 LB = 0; EB = 0;
#20 rst = 0;   // 리셋이 0이 될 때의 값 변화 확인
#10 rst = 1;   // 다시 리셋을 1로 만들고 정상 진행
#10 LA = 1; EA = 1;
#5 EB = 1;
#10 LA = 0;
end
endmodule
```

2. 시뮬레이션 결과

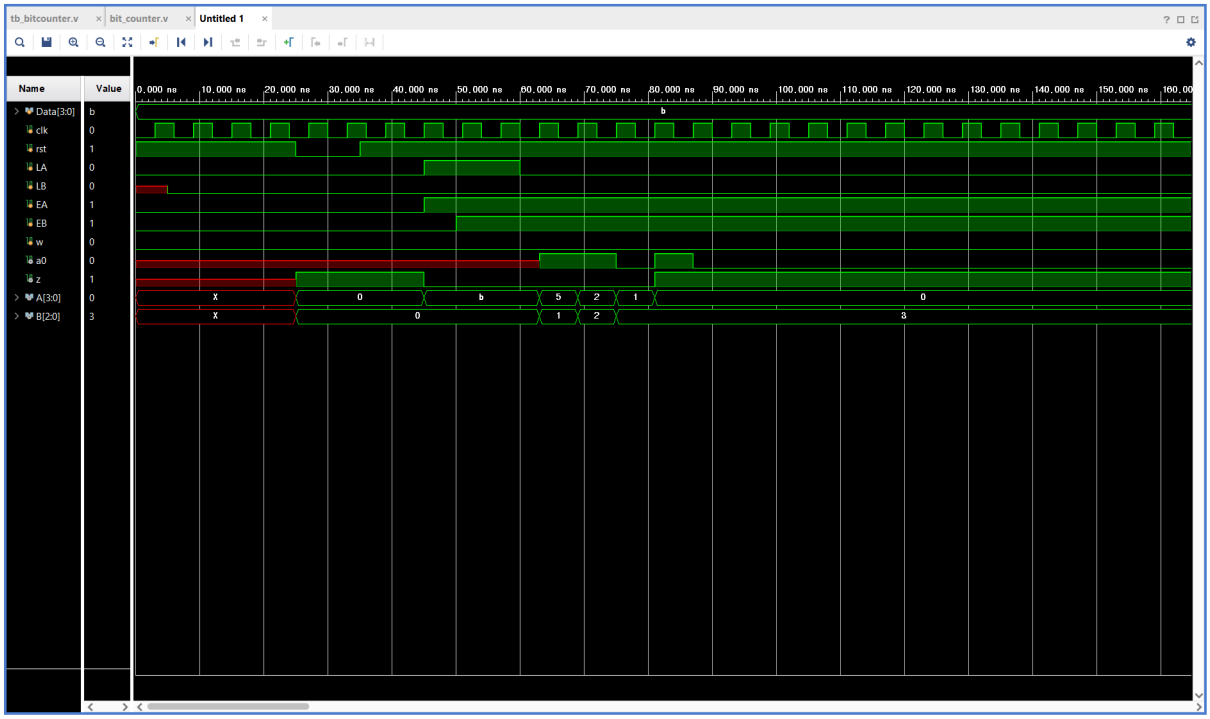


[그림 1] 1011 Bit Counting

[그림 1]은 Data에 1011 값을 입력 후, bitcounting을 한 시뮬레이션입니다.

A[3] bit부터 0을 대입하면서 shifting을 구현하였고, 5 2 1 0 순으로 shifting이 정상적으로 동작한 것을 확인할 수 있습니다.

Shifting을 통하여 나온 bit가 1일 경우, B의 값을 1씩 증가시켰는데 0 1 2 3으로 bit 수를 정상적으로 counting한 것을 확인할 수 있습니다.



[그림 2] 1100 Bit Counting

[그림 2]는 [그림 1]과 같은 방식으로, Data에 1100 값을 입력 후, bitcounting을 한 시뮬레이션입니다.

6 3 1 0 순으로 shifting이 정상적으로 동작한 것을 확인할 수 있습니다.

또한, 0 1 2으로 bit 수를 정상적으로 counting한 것을 확인할 수 있습니다.