```
// AlU module
module ALU #(parameter W=4)
      (input [W-1:0] A,
      input [W-1:0] B,
      input [2:0] ALUcntrl,
      output reg [W-1:0] O,
      output reg CO,
      output reg OVF,
      output reg N,
      output reg Z);
// Since switch case block can be implemented only in always block
// I created always block with default sensitivity list
always @(*)
begin
   OVF=0;// Assign default parameters
   CO=0;// These parameters are meaningful in only addition and subtraction
   // Doing operations according to value of control signals
   case(ALUcntrl)
      // Addition
      3'b000:
         begin
         {CO,O} = A+B;
         Z = (O==0) ? 1:0;
         N=O[W-1];
         // Overflow occurs when A and B are positive however result is negative
         if (~A[W-1] & ~B[W-1])
            begin
            if(O[W-1])
               OVF=1;
            end
         // Overflow occurs when A and B are negative however result is positive
         if (A[W-1] & B[W-1])
            begin
            if(~O[W-1])
               OVF=1;
            end
         end
      // Subtraction
      3'b001:
         begin
         {CO,O} = A-B;
         Z = (O==0) ? 1:0;
         // Overflow occurs when A is Negative and B is positive however result is positiv
         if (A[W-1] & ~B[W-1])
            begin
            if(~O[W-1])
               OVF=1;
            end
         // Overflow occurs when A is Positive and B is negative however result is negativ
         if (~A[W-1] & B[W-1])
            begin
            if(O[W-1])
               OVF=1;
            end
         N=O[W-1];
         end
      // Subtraction
      3'b010:
         begin
         {CO,O} = B-A;
         Z = (O==0) ? 1:0;
         // Overflow occurs when A is positive and B is negative however result is positiv
         if (~A[W-1] & B[W-1])
            begin
            if(~O[W-1])
               OVF=1;
            end
         // Overflow occurs when A is negative and B is positive however result is negativ
         if (A[W-1] & ~B[W-1])
            begin
            if(O[W-1])
               OVF=1;
            end
         N=O[W-1];
         end
      //Bit Clear
      3'b011:
         begin
         O = A & ~B;
         Z = (O==0) ? 1:0;
         N=O[W-1];
         end
      //AND operator
      3'b100:
         begin
         O = A & B;
         Z = (O==0) ? 1:0;
         N=O[W-1];
         end
      //OR operator
      3'b101:
         begin
         O = A | B;
         Z = (O==0) ? 1:0;
         N=O[W-1];
         end
      //XOR operator
      3'b110:
         begin
         O = A ^ B;
         Z = (O==0) ? 1:0;
         N=O[W-1];
         end
      //XNOR operator
      3'b111:
         begin
         O = ~(A ^ B);
         Z = (O==0) ? 1:0;
         N=O[W-1];
         end
   endcase
end
endmodule
```