# EGC221
# Class Notes
# 11/13/2018

**Baback Izadi**
Division of Engineering Programs
bai@engr.newpaltz.edu

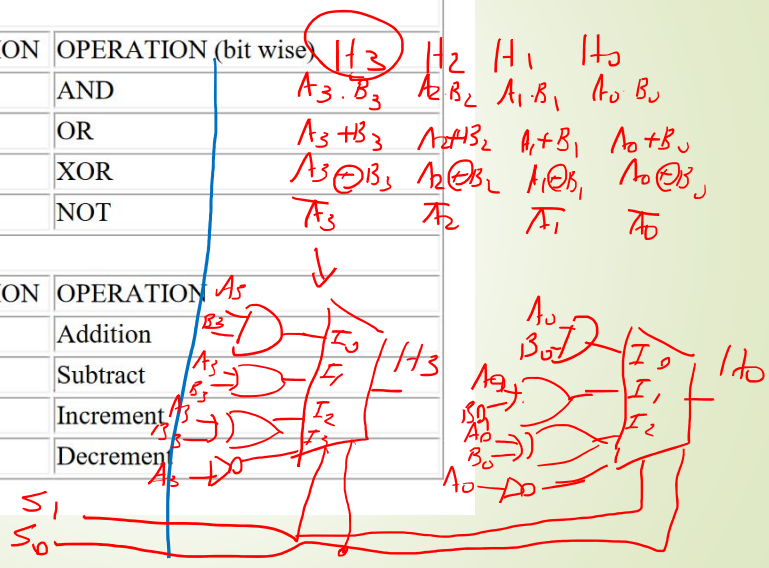
Table 1: Functions of ALU

Logic

| M | S1 | S0 | FUNCTION | OPERATION (bit wise) |
|---|---|---|---|---|
| 0 | 0 | 0 | A · B | AND |
| 0 | 0 | 1 | A + B | OR |
| 0 | 1 | 0 | A ⊕ B | XOR |
| 0 | 1 | 1 | A' | NOT |

Arithmetic

| M | S1 | S0 | FUNCTION | OPERATION |
|---|---|---|---|---|
| 1 | 0 | 0 | A + B | Addition |
| 1 | 0 | 1 | A - B | Subtract |
| 1 | 1 | 0 | A + 1 | Increment |
| 1 | 1 | 1 | A - 1 | Decrement |



# Design of an ALU using Case Statement

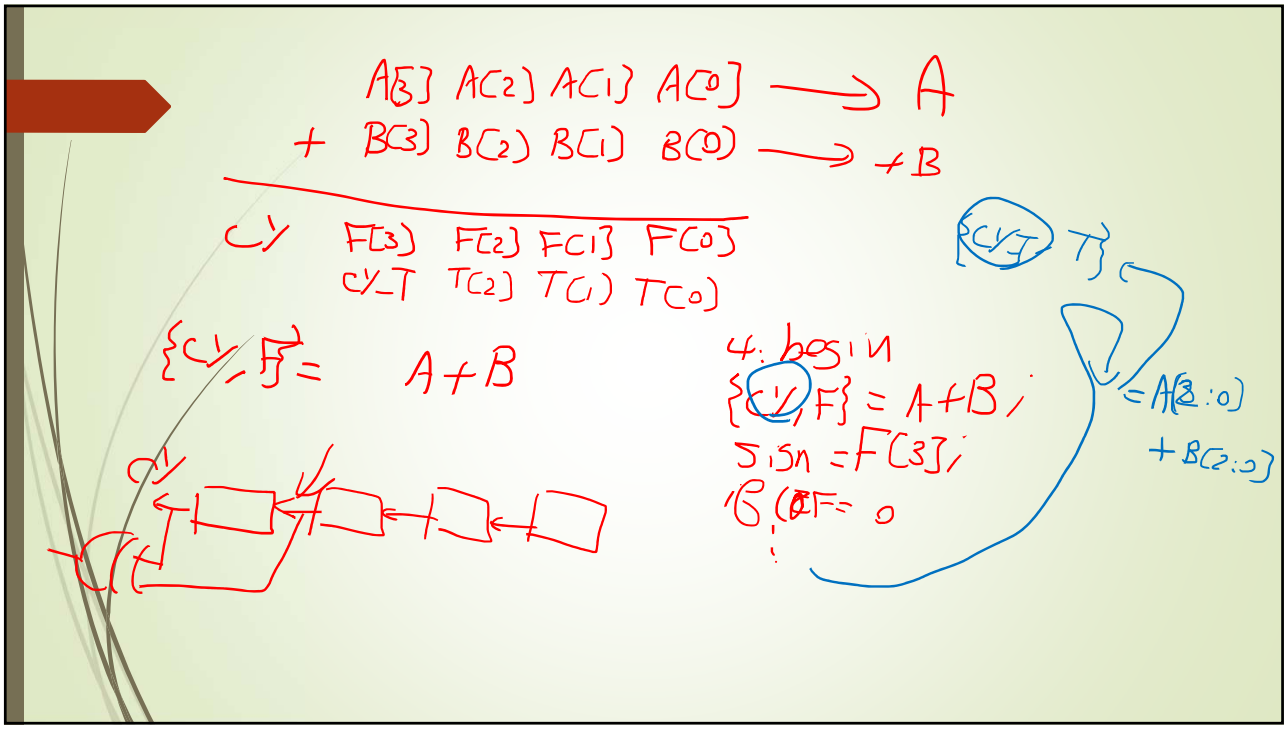
| S | Function |
|---|---|
| 0 | A AND B |
| 1 | A OR B |
| 2 | A XOR B |
| 3 | A' |
| 4 | A+B |
| 5 | A-B |
| 6 | A+1 |
| 7 | A-1 |

```
// 74381 ALU
module alu(s, A, B, F);
input [2:0] s;
input [3:0] A, B;
output [3:0] F;
reg [3:0] F;
always @(s or A or B)
case (s)
0: F = A & B;
1: F = A | B;
2: F = A ^ B;
3: F = ~A;
4: F = A + B;
5: F =A-B;
6: F =A+4'b0001;
7: F = A+4'b1111;
endcase
endmodule
```

# . Concatenation and Replication in Verilog

- The concatenation operator "{ , }" combines (concatenates) the bits of two or more data objects. The objects may be scalar (single bit) or vectored (multiple bit). Multiple concatenations may be performed with a constant prefix and is known as replication.

```
module Concatenation (A, B, Y);
        input [2:0] A, B;
        output [14:0] Y;
        parameter C=3'b011;
        reg [14:0] Y;
        always @(A or B)
        begin
                Y={A, B, {2{C}}, 3'b110};
        end
endmodule
```




## Bit-wise Operations in Verilog

```
module Bitwise (A, B, Y);
        input [6:0] A;
        input [5:0] B;
        output [6:0] Y;
        reg [6:0] Y;
        always @(A or B)
        begin
                Y[0]=A[0]&B[0]; //binary AND
                Y[1]=A[1]|B[1]; //binary OR
                Y[2]=!(A[2]&B[2]); //negated AND
                Y[3]=!(A[3]|B[3]); //negated OR
                Y[4]=A[4]^B[4]; //binary XOR
                Y[5]=A[5]~^B[5]; //binary XNOR
                Y[6]=!A[6]; //unary negation
        end
endmodule
```