DSD HW1

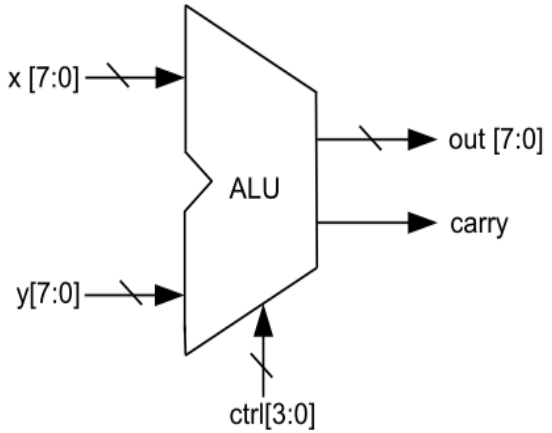
Speaker: Alex

Date: 2025/03/06

Problem 1: 8-bit Arithmetic Logic Unit

- Implement 8-bit ALU with following functions

Control Signal(ctrl)	Description	Function
0000	Add(signed)	out = x + y
0001	Sub(signed)	out = x - y
0010	Bitwise And	out = and(x, y)
0011	Bitwise Or	out = or(x , y)
0100	Bitwise Not	out = not(x)
0101	Bitwise Xor	out = xor(x , y)
0110	Bitwise Nor	out = nor(x , y)
0111	Shift left logical variable	out = y << x[2:0]
1000	Shift right logical variable	out = y >> x[2:0]
1001	Shift right arithmetic	out ={x[7],x[7:1]}
1010	Rotate left	out = {x[6:0] , x[7]}
1011	Rotate right	out = {x[0] , x[7:1]}
1100	Equal	out = (x==y)?1:0



Ex. Add(signed)
x= 8'b10010110(-106)
y= 8'b00101101(45)
Ans: 9'b1_11000011(-61)
→ carry = 1, out = 8'b11000011

Problem 1: 8-bit Arithmetic Logic Unit

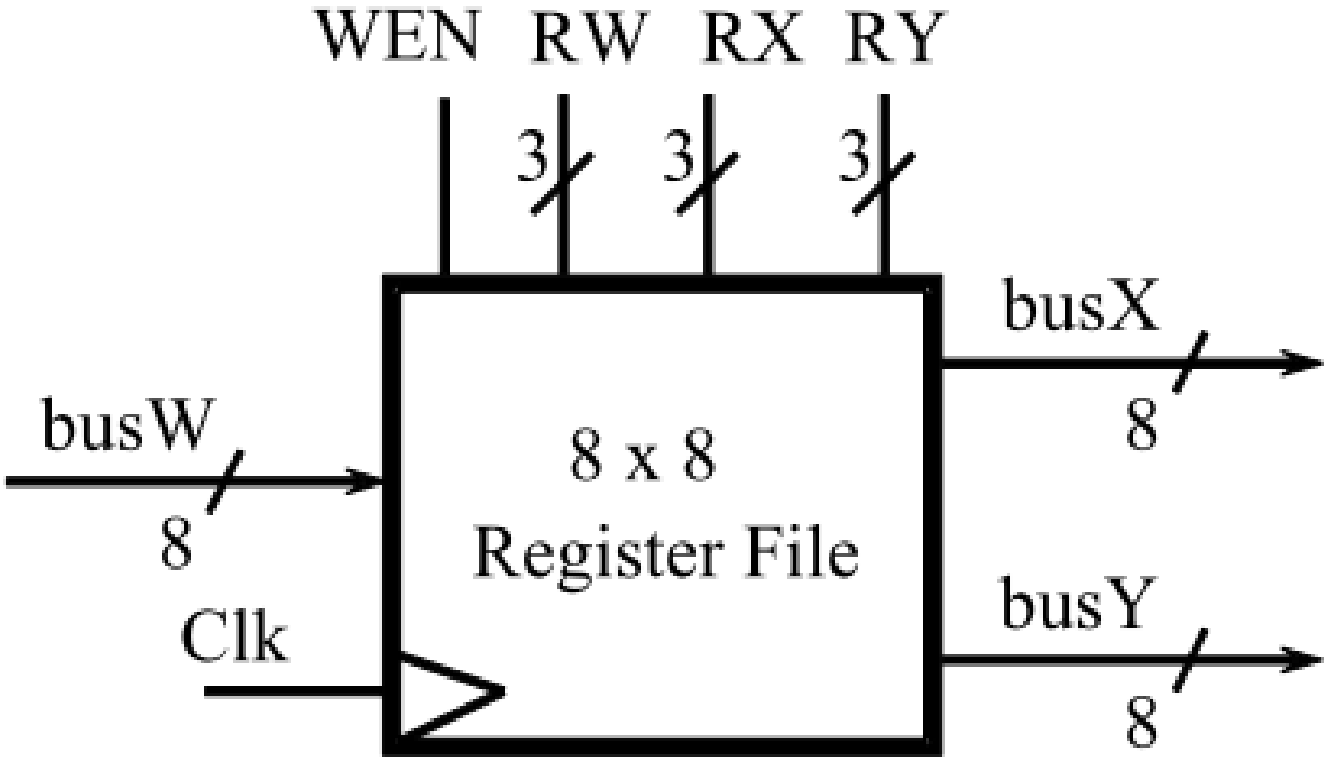
- (1) Implement RTL (***use continuous assignment, assign***) model of the ALU
- (2) Implement RTL (***use procedural assignment, always block***) model of the ALU
- (3) Modify the given testbench to verify all functions in your design are correct

Only test Boolean not

```
initial begin
    ctrl = 4'b1101;
    x    = 8'd0;
    y    = 8'd0;

    #(`CYCLE);
    // 0100 boolean not
    ctrl = 4'b0100;

    #(`HCYCLE);
    if( out == 8'b1111_1111 ) $display( "PASS --- 0100 boolean not" );
    else $display( "FAIL --- 0100 boolean not" );

    // finish tb
    #(`CYCLE) $finish;
end
```

Problem 2. 8x8 Register File

- Implement a 8 x 8 register file which can support read and write operation



Flip-flop

- Combinational circuit

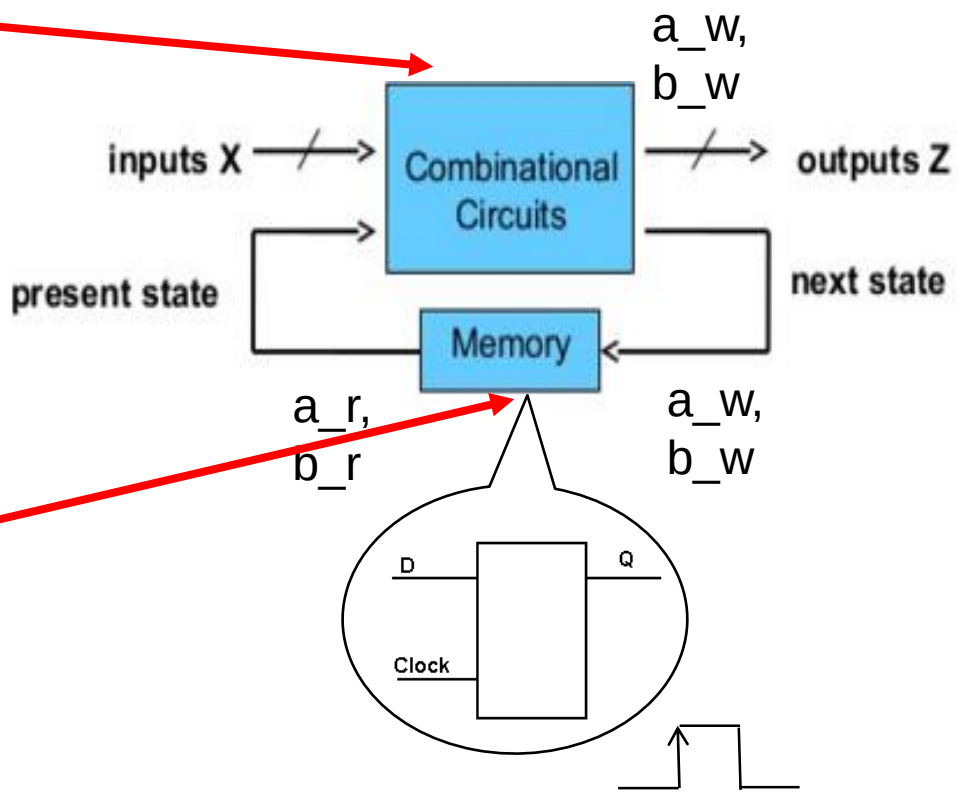
```
always@(*) begin
    a_w = a_r;
    b_w = a_r + 1;
end
```

a_w -> a_nxt

a_r -> a

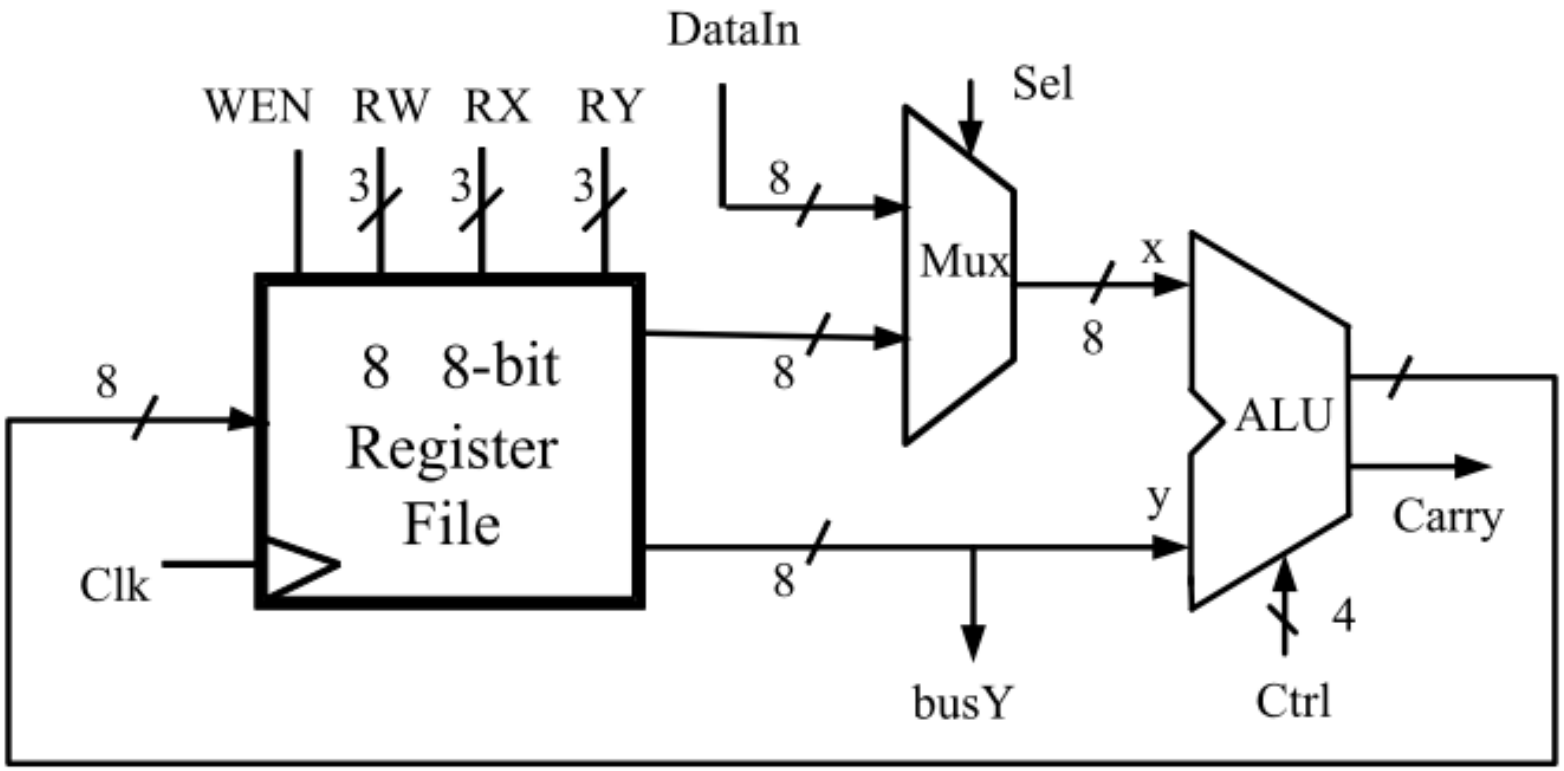
- Sequential circuit

```
always@(posedge Clk) begin
    a_r <= a_w;
    b_r <= b_w;
end
```



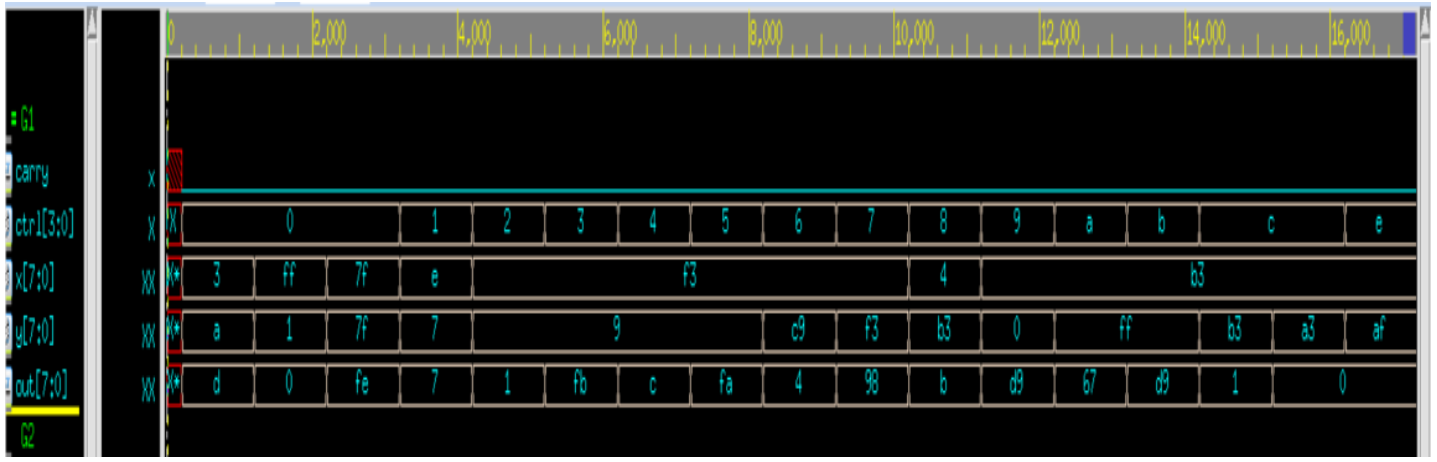
Problem 3. Simple Calculator

- Combine the previous two designs (ALU, Register File) into a simple calculator unit



Report

1. **ALU**: Screenshot the waveform result in nWave of alu_assign & alu_always, describe how you verify the correctness
2. **Register file**: Screenshot the waveform result in nWave of register_file, describe how you verify the correctness.
3. **Bonus (5%)**: Write down what you found. Feel free to share your experience. (Ex: some mistakes you spend a lot of time, your environment, naming method)



Note

- Continuous assignment
  - LHS has to be wire type
- always statement
  - LHS has to be reg type
- Don’t use non-synthesizable syntax
- Port
  - input
    - within the module, input must be declared as type wire(net), externally, they can connect to a type reg or wire
  - output
    - within the module, output can be declared as type reg or wire(default wire), externally they must always connect to net(wire)
    - if output port used in an always statement, the port must be redeclared as type reg, e.g. output reg carry;