UNIVERSITY OF CALIFORNIA—DAVIS

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

EEC180 — DIGITAL SYSTEMS II — WINTER 2023

Tobin Joseph

HOMEWORK 4

1) IEEE 784 Standard (50 points)
Consider the two numbers A = 3.0 and B = -3.5. We would like to use the IEEE 754 standard representation. In this problem, we assume an 8-bit representation with 1 bit for the sign, 3 for the exponent in excess-3 notation (which means the bias is 3), and 4 bits for the mantissa.
1. What is the minimum positive number that can be represented if the significand is normalized? (8 points)

$0\ 000\ 0000$

$X = (-1)^{S} \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$

$X = (-1)^{0} \times (1 + .0000) \times 2^{(000 - 3)}$

$X = 1 \times (1.0000) \times 2^{(-3)}$

$X = 2^{-3}$

$X = 0.125$

2. What is the maximum positive number that can be represented? (8 points)

$0\ 111\ 1111$

$X = (-1)^{S} \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$

$X = (-1)^{0} \times (1 + .1111) \times 2^{(7 - 3)}$

$X = 1 \times (1.1111) \times 2^{(4)}$

$X = (11111)_{b}$

$X = 31$

3. What decimal number is represented by the bit pattern 10000001? (8 points)

$1\ 000\ 00001$

$X = (-1)^{S} \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$

$X = (-1)^{1} \times (1 + .0001) \times 2^{(000 - 3)}$

$X = (-1) \times (1.0001) \times 2^{(-3)}$

$X = -(0.0010001)_{b}$

$X = -0.1328125$

4. Find the IEEE 754 binary representation of the numbers A and B. (10 points)

$A = 3.0 = (11)_b = 1.1000 \times 2^1$
Positive = 0
Exponent = 1 + Bias = 1 + 3 = 4
Mantissa = 1000

A = 01001000

$B = -3.5 = -11.1 = -1.1100 \times 2^1$
Negative = 1
Exponent = 1 + Bias = 1 + 3 = 4
Mantissa = 1100

B = 11001100

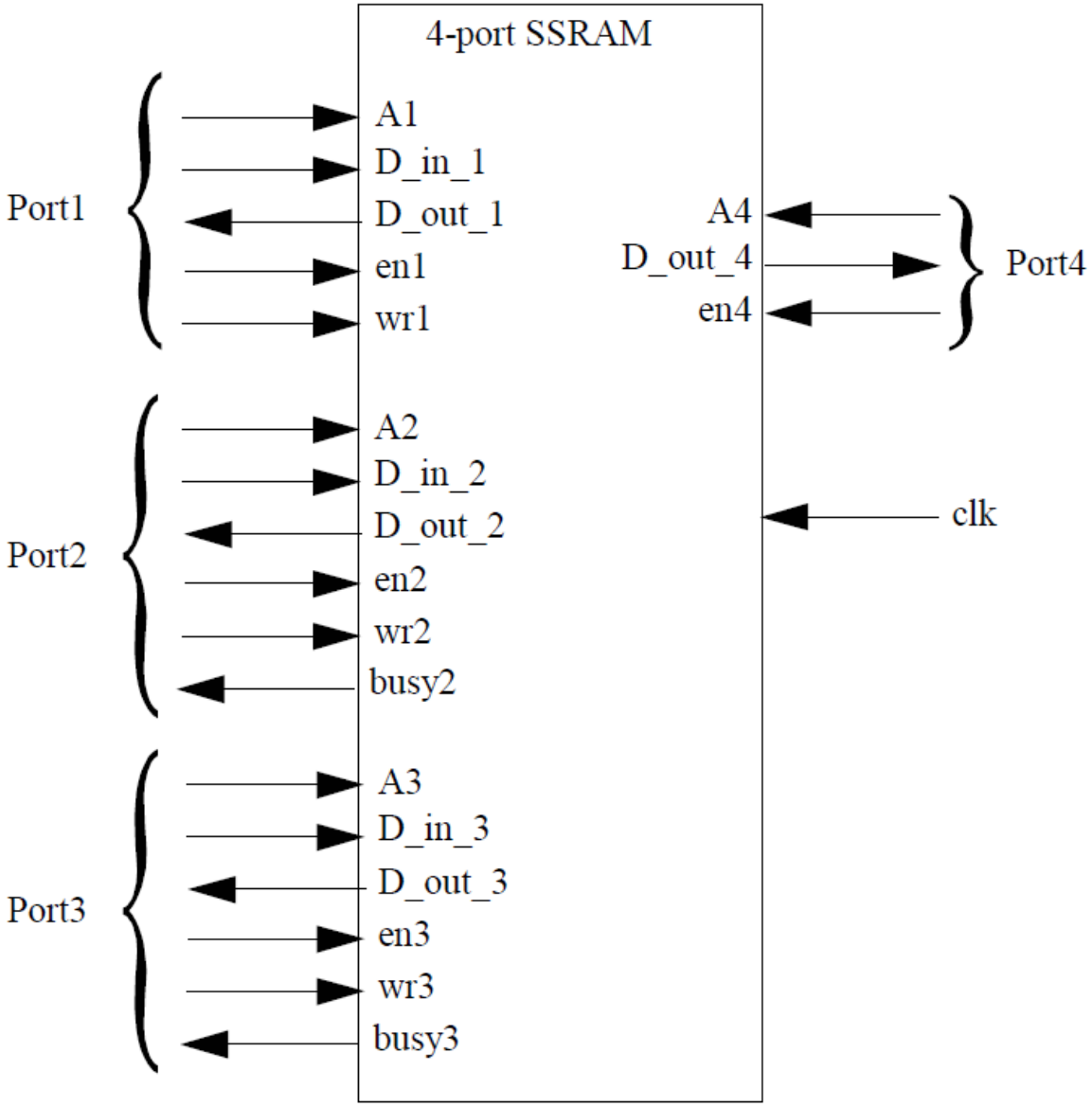
5. Compute P = A × B using the binary (not decimal) floating-point representations of A and B. (16 points)

$P = A \times B = 3.0 \times (-3.5)$
$P = (1.1000)_2 \times 2^1 \times (-1.1100)_2 \times 2^1$

Step 1. Add exponents (unbiased)
1 + 1 = 2
Step 2. Multiply significands
$(1.1000)_2 \times (1.1100)_2 = (10.101)_2 \Rightarrow (10.101)_2 \times 2^2$
Step 3. Normalize results & check for over/underflow
$(1.0101)_2 \times 2^3$ (no over/underflow)
Step 4. Round and renormalize if necessary
$(1.0101)_2 \times 2^3$ (no change)
Step 5. Determine sign: =ve x -ve => -ve
$(-1.0101)_2 \times 2^3 = -10.5$

$P = (-1.0101)_2 \times 2^3 = -10.5$

2) Multiple-Port Memory (50 points).
Consider the 4-port SRAM shown below. The SRAM has three read/write ports and one read-only port. All read operations of this SRAM are combinational (asynchronous). The output signal busy2 is set to 1 when the write operation at port2 cannot proceed due to a concurrent write at the same address from port1. The output signal busy3 is set to 1 when the write operation at port3 cannot proceed due to a concurrent write at the same address from port1 or port2.
Write a complete Verilog model of the SRAM. Your model should be able to handle any special cases that may arise when using the SRAM.



```
module mutlti_port_sram(input clk,
                                    // port 1
                                    input [7:0]A1,
                                    input [7:0]D_in_1,
                                    input en1,
                                    input wr1,
                                    output reg [7:0]D_out_1,
                                    // port 2
                                    input [7:0]A2,
                                    input [7:0]D_in_2,
                                    input en2,
                                    input wr2,
                                    output reg [7:0]D_out_2,
                                    output reg busy2,
                                    // port 3
                                    input [7:0]A3,
                                    input [7:0]D_in_3,
                                    input reg en3,
                                    input wr3,
                                    output reg [7:0]D_out_3,
                                    output reg busy3,
                                    // port 4
                                    input [7:0]A4,
                                    input en4,
                                    output reg [7:0]D_out_4);

      reg [7:0] mem [0:255];

      // Synchronous-write RAM, write on posedge clk
      always @(posedge clk) begin
            // Write operations complete one cycle after valid address,
            // data, and wrX = 1 are present at the inputs
            if (wr1 == 1'b1) begin
                  // Write to port 1 rakes precedence
                  mem[A1] <= #1 D_in_1;
                  // default values
                  busy2 = 1'b0;
                  busy3 = 1'b0;
            end
            if (wr2 == 1'b1) begin
                  // The output signal busy2 is set to 1 when the write
                  // operation at port2 cannot proceed due to a
                  // concurrent write at the same address from port1
```

```
            if (wr1 == 1'b1 && A1 == A2) begin
                  busy2 = 1'b1;
            end
            else begin
                  busy2 = 1'b0;
                  mem[A2] <= #1 D_in_2;
            end
      end
      else begin
            busy2 = 1'b0;
      end
      if (wr3 == 1'b1) begin
            // The output signal busy3 is set to 1 when
            // the write operation at port3 cannot proceed
            // due to a concurrent write at the same address
            // from port1 or port2.
            if ((wr1 == 1'b1 && A1 == A3) || (wr2 == 1'b1 && A2 == A3))
            begin
                  busy3 = 1'b1;
            end
            else begin
                  busy3 = 1'b0;
                  mem[A3] <= #1 D_in_3;
            end
      end
      else begin
                  busy3 = 1'b0;
      end

end

// All read operations of this SRAM are combinational (asynchronous)
// Simultaneous read/write to same address will read the current
// contents of the memory
// and then write data to the memory when the clock risesr
// Perform reads asynchronously on enable change to 1
always @(en1 or en2 or en3 or en4) begin
      if (en1 == 1'b1) begin
            D_out_1 = mem[A1];
      end
      if (en2 == 1'b1) begin
            D_out_2 = mem[A2];
      end
```

```
            if (en3 == 1'b1) begin
                  D_out_3 = mem[A3];
            end
            if (en4 == 1'b1) begin
                  D_out_4 = mem[A4];
            end
      end

endmodule
```