DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING

COLLEGE OF E&ME, NUST, RAWALPINDI

Digital System Design

Assignment#2

SUBMITTED TO:

Dr. Muhammad Yasin

SUBMITTED BY:

Amina Qadeer

DE-42 (C&SE)-A

Amina Qadeer 359607 CE42A

DSD ASSIGNMENT 02

Q1 (a) $0 \quad \underbrace{01111011}_{123} \quad 01011100001010001111011$

$(-1)^0 \times 2^{123-127} \times 1.01011100 \ldots\ldots 11011)$

$= 2^{-4} \times (1.0101100 \ldots\ldots 11011)$

$= (0.000101011100001010001111011)_2$

$= 2^{-4} + 2^{-6} + 2^{-8} + 2^{-9} + 2^{-10} + 2^{-15} + 2^{-17} + 2^{-21} + 2^{-22}$

$+ 2^{-23} + 2^{-24} + 2^{-26} + 2^{-27}$

$= 0.085000000\ 894069 67$

$= 8.5 \times 10^{-2}$

(b) $0\ \underbrace{10000111}\ 10011010000100000000000$

$(1)^0 \times 2^{135-127} \times 1.1001101 \ldots\ldots 00000)$

$= 2^8 \times (1.1001101 \ldots\ldots 0000)$

$= (110011010.0001)_2 = 410.0625$

$= 4.1 \times 10^2$

Q2 $16.25_d \rightarrow 10000.01\ b$

$32.5 \longrightarrow 100000.1\ b$

10000.01×2^0

1.000001×2^4

$\boxed{\begin{array}{l} e-8=4 \\ e=12_d=1100_b \end{array}}$

$16.25 \longrightarrow 0\ 1100\ 00000$ (underflow)

100000.1×2^0

1.000001×2^5

$\boxed{\begin{array}{l} e-8=5 \\ e=13_d=1101 \end{array}}$

$32.5 \longrightarrow 0\ 1101\ 00000$ (underflow)

Aligning the exponents by increasing the smalle exponent to match the larges. Adjusting the mantissa accordingly.

Amina Qadeer 359607 CE-42-A

16.25 → 0 1101 0.100000

After 2nd Complement, 1

1.011111

+ 1

1.900000

1.00000 (32.5)

+ 1.10000 (-16.25)

10.10000

Ignore the overflow and subtract

32.5-16.25 → 0.100000$_b \times 2^{15}$

1.0000 $\times 2^{12}$

01100 0000

= $\boxed{(180)_h}$

(b) 1.5d → $1.1_b \times 2^0$ - 1.5d = 2.25

0 1 000 10000

↳ multiplying mantixas

$1.1_b \times 1.1_b = 10.01_b$

↳ adding exponents

```
   1.1
 x 1.1
 -----
    11
 + 11x
 -----
 10.01
```

$1000_b + 1000_b - 1000 - 1000_b = 0000_b$

2.25d → $10.01_b \times 2^0$

1.001×2^1

0 1001 00100

= $(124)_h$

Q3 (a) 2.5d → 010.1000000b

1.25d → 001.0100000b

Amina Qadeer 359607 CE-42-A

1-25d - 2-5d = 1-25 + (-2-5)d

Take 2's complement of 2-5d

```
  101.0111111
            1
  -----------
  101.1000000
```

Add both numbers

```
    001.0100000    (125d)
  + 101.1000000    (-2-5d)
  -------------
    110.1100000    (-1.25d)
```

$\boxed{110.1100000}$

a_1 = 2-5d → 010.10000000b

a_2 = 1-5d → 01.10b

```
   00.00000000    (2-5)
   01.0           (1.5)
  -------------
  100.00000000    (4.0)
```

If Q3.7 is signed → overflow

Solution: put all bits except MSB.

Q4 a- Q9.7 format can be used. Max number it allows 255.9921875 and minimum number it allows is -256.

b→ -7.5d → $-111.1b \times 2^0$

-1.111×2^2

(2.)

1 10000010: 111000100000000000000000000 000

8.0625d → 1000.0001×2^0

1.0000001×2^3

(16)

0 10000011 0000001000000 0000000000.00

9.125d → 1001.001 b × 2^0 = 1.001001 × 2^3

01000001100100100000000000000000

10.75d → 1010.11 b × 2^0

1.01011 × 2^3

01 0000011 01011000000000000000

ii— −7.5d →

− (111.1b) × 2^0

0111.10

1st Complement

1000.01

+1

2nd complement

−7.5d ← 1000.10

8.0625d → 1000.0001 b × 2^0

cannot fit in Q 4.2

0111.11b → 7.75d

9.125d → 1001.001

cannot fit in Q 4.2

0111.11b → 7.75d

10.75d → 1010.11b × 2^0

cannot fit in Q 4.2

0111.11b → 7.75d

MAE with IEEE = 0

MAE with fixed Point = $\frac{1}{4}\Big[(-7.5-(-7.5)) + (8.625-7.75) + (9.125\text{-}$
$+ (10.75 - 7.75)$

= 1.171875

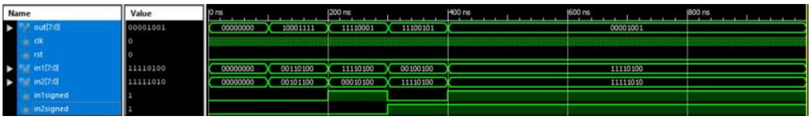
Unified Multiplier:

```
module Multiplier(
    input clk, rst,
    input [7:0] in1, in2,
    input in1signed, in2signed,
    output [7:0] out
);

    reg signed [15:0] intermediate;
    reg signed [7:0] in1temp, in2temp;
    reg [15:0] temp, temp2, temp3, temp4;
    integer i = 2;

    assign out = intermediate[11:4];

    initial begin
        intermediate <= 0;
        in1temp <= 0;
        in2temp <= 0;
        temp <= 0;
        temp2 <= 0;
        temp3 <= 0;
        temp4 <= 0;
    end

    always @(posedge clk or posedge rst) begin
        if (rst) begin
            intermediate = 0;
            in1temp = 0;
            in2temp = 0;
            temp = 0;
        end else begin
            if (~in1signed && ~in2signed) begin
                intermediate <= in1 * in2;
            end
            if (in1signed && ~in2signed) begin
                intermediate = 0;
                for (i = 0; i < 8; i = i + 1) begin
                    temp = in1 * in2[i];
                    temp2 = {{8{temp[7]}}, temp[7:0]};
                    temp3 = (temp2 <<< i);
                    intermediate = intermediate +
temp3;
                end
            end
            if (~in1signed && in2signed) begin
                intermediate = 0;
                for (i = 0; i < 7; i = i + 1) begin
                    temp = in1 * in2[i];
                    temp2 = temp <<< i;
                    intermediate = intermediate +
temp2;
                end
                temp = in1 * in2[7];
                temp2 = (~temp + 1) <<< 7;
                intermediate = intermediate + temp2;
            end
            if (in1signed && in2signed) begin
                intermediate = 0;
                for (i = 0; i < 7; i = i + 1) begin
                    temp = in1 * in2[i];
                    temp2 = {{8{temp[7]}}, temp[7:0]};
                    temp3 = temp2 <<< i;
                    intermediate = intermediate +
temp3;
                end
                temp = in1 * in2[7];
                temp2 = {{8{temp[7]}}, temp[7:0]};
                temp3 = (~temp2 + 1) <<< 7;
                intermediate = (intermediate + temp3)
<< 1;
            end
```

```
        end
    end

endmodule
```

mulTB testbench:

```
module mulTB;
    // Inputs
    reg clk;
    reg rst;
    reg [7:0] in1;
    reg [7:0] in2;
    reg in1signed;
    reg in2signed;

    // Outputs
    wire [7:0] out;

    // Instantiate the Unit Under Test (UUT)
    Multiplier uut (
        .clk(clk),
        .rst(rst),
        .in1(in1),
        .in2(in2),
        .in1signed(in1signed),
        .in2signed(in2signed),
        .out(out)
    );

    always #1 clk = ~clk;

    initial begin
        clk = 0;
        rst = 0;
        in1 = 0;
        in2 = 0;
        in1signed = 0;
        in2signed = 0;
        #100;

        in1signed = 0;
        in2signed = 0;
        in1 = 8'b00110100;
        in2 = 8'b00101100;
        #100;

        in1signed = 1;
        in2signed = 0;
        in1 = 8'b11110100;
        in2 = 8'b00010100;
        #100;

        in1signed = 0;
        in2signed = 1;
        in1 = 8'b00100100;
        in2 = 8'b11110100;
        #100;

        in1signed = 1;
        in2signed = 1;
        in1 = 8'b11110100;
        in2 = 8'b11111010;
    end

endmodule
```


Name
Value
0 ns
200 ns
400 ns
600 ns
800 ns
out[7:0]
00001001
00000000
10001111
11110001
11100101
00001001
clk
0
rst
0
in1[7:0]
11110100
00000000
00110100
11110100
00100100
11110100
in2[7:0]
11111010
00000000
00101100
00010100
11110100
11111010
in1signed
1
in2signed
1