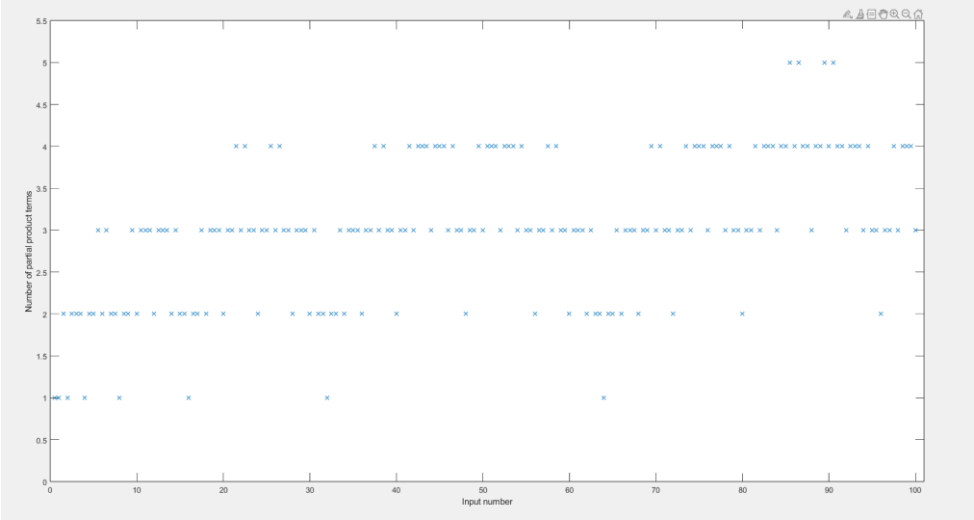
1. Using matlab, write a function which calculates the minimum number of partial products needed to calculate the product of a number multiplied by a fixed number. Both +multiplicand and –multiplicand partial products are allowed. Consider positive and negative numbers with a resolution of 0.5 (e.g., 0, 0.5, 1.0, ...). For example, a multiplicand of +5.5 should result in 3 partial products (either +4, +2, –0.5; or +4, +1, +0.5).

a) Write the described function in matlab.

```
% numppterms.m
function num_products = numppterms(X)
    num_products = 0;
    while X ~= 0
        % Find the nearest power of two (positive or negative) to the remainder
        abs_remainder = abs(X);
        k = ceil(log2(abs_remainder));
        larger = 2^k;
        smaller = 2^(k-1);

        % Choose the closer power of two
        if abs(larger - abs_remainder) < abs(abs_remainder - smaller)
            chosen = larger * sign(X);
        else
            chosen = smaller * sign(X);
        end

        % Subtract the chosen power of two from the remainder
        X = X - chosen;
        % Increment the number of partial products
        num_products = num_products + 1;
    end
end
```

b) Assuming your function is called numppterms(), run the following bit of matlab code (a few points need fixing), submit the figure, and report the Total Sum for all numbers 0.5 – 100.00 .

Total Sum = 605




```
% prob1.m
StepSize = 0.5;
NumTermsArrayPos = zeros(1, 100/StepSize);    % small speedup if init first
for k = 0.5 : StepSize : 100
  NumTermsArrayPos(round(k/StepSize)) = numppterms(k);
end

fprintf('Total sum = %i\n', sum(NumTermsArrayPos));

figure(1); clf;
plot(0.5:StepSize:100, NumTermsArrayPos, 'x');
axis([0 101 0 1.1* max(NumTermsArrayPos)]);
xlabel('Input number');
ylabel('Number of partial product terms');
```

2. An FIR filter has the coefficients:

coeff = [–3 –31 –88 +212 284 +212 –88 –31 –3]

Assume the coefficients cannot be scaled smaller, but they can be scaled up to 2× larger. This implementation works with integers only, so round() all scaled coefficients.

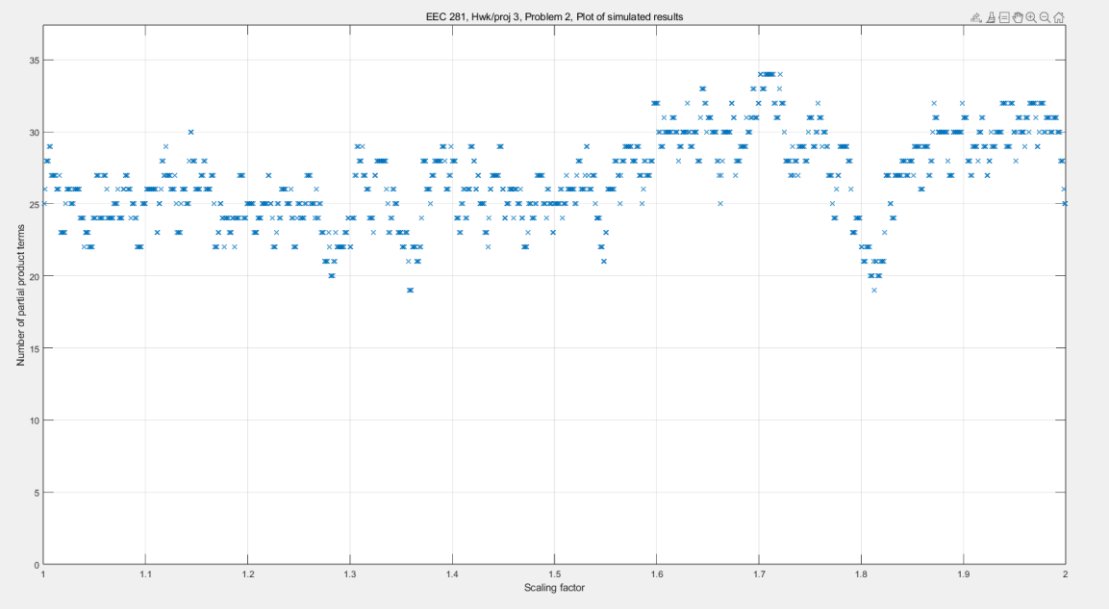
a) How many partial products are necessary to implement the FIR filter with the given coefficients?

25

b) Find the scaling for the coefficients that yields the minimum number of partial products.

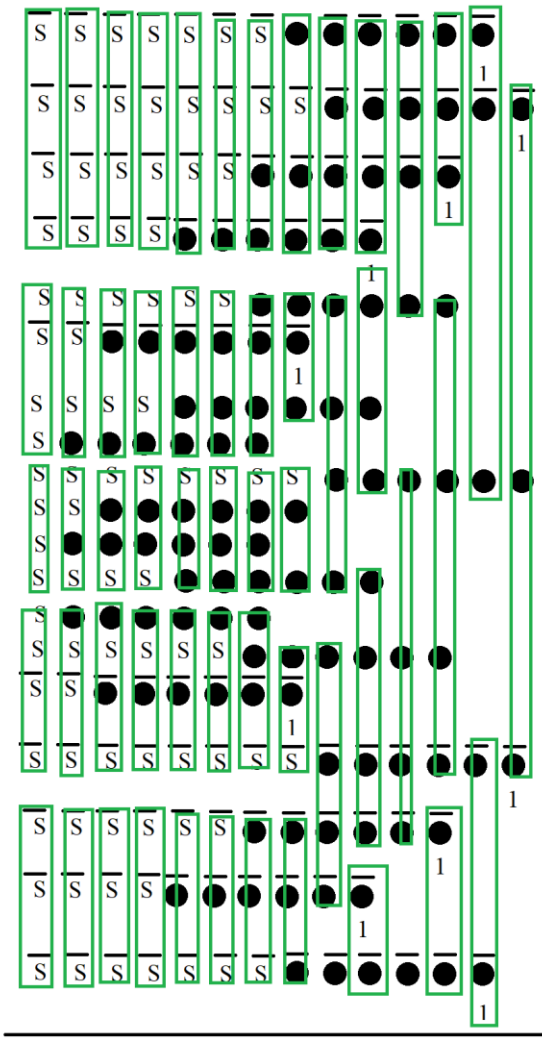
1.358

c) Turn in a plot of the number of required partial products vs. the scaling factor. The plot should look something like the fake results this matlab code generates.



d) Draw a dot diagram showing how the partial products would be added (include sign extension) for the optimized coefficients you found in (b), using the FIR architecture shown below and a 6-bit 2's complement input word. Use 4:2, 3:2, and half adders as necessary and no need to design the final stage carry-propagate adder.

round(coeff * 1.358) = [-4 -42 -120 288 386 288 -120 -42 -4]



h(0) = -4

h(1) = -42 = -32 -8 -2

h(2) = -120 = -128 + 8

h(3) = 288 = 256 + 32

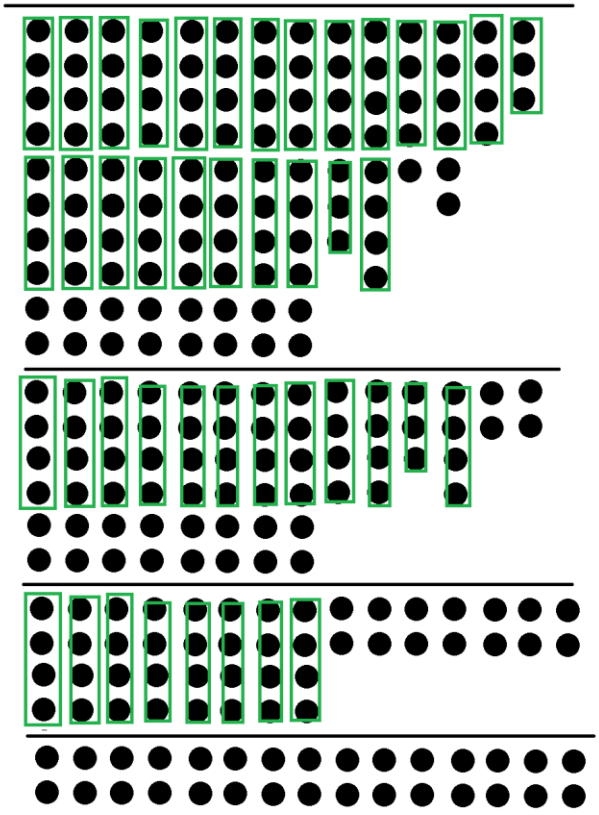
h(4) = 386 = 256 + 128 + 2

h(5) = 288 = 256 + 32

h(6) = -120 = -128 + 8

h(7) = -42 = -32 -8 -2

h(8) = -4



```
% prob2.m
coeff = [-3 -31 -88 +212 284 +212 -88 -31 -3];
sumOfProduct = zeros(1, 1 / 0.001);
stepSize = 0.001;
minAns = 0x7fffffff;
minScale = 0;
for scale = 1.001 : stepSize : 2.000
    tmp = zeros(1, 9);
    for i = 1 : 9
        tmp(i) = numppterms(round(scale * coeff(i)));
    end
    sumOfProduct(round((scale - 1) / stepSize)) = sum(tmp);
    if(sum(tmp) < minAns)
        minAns = sum(tmp);
        minScale = scale;
    end
end
fprintf("minScale : %d\n", minScale);
figure(1); clf;
plot(1.001:0.001:2, sumOfProduct, 'x');
axis([1 2 0 1.1* max(sumOfProduct)]);  grid on;
xlabel('Scaling factor');
ylabel('Number of partial product terms');
title('EEC 281, Hwk/proj 3, Problem 2, Plot of simulated results');
```

3. Design of an area-efficient low-pass FIR filter. The filter must meet the following specifications when its sample rate is 100 MHz.

- Passband below 12.0 MHz: no more than 3dB ripple from minimum to maximum levels
- Passband below 18.0 MHz: no more than 3dB attenuation below gain level at DC
- Stopband above 26.0 MHz: at least 16dB attenuation below gain level at DC
- Stopband above 34.0 MHz: at least 27dB attenuation below gain level at DC

a) Write a matlab function lpfirstats(H) or lpfirstats(H,W) that takes a frequency response vector H from [H,W] = freqz(coeffs) (or a vector of filter coefficients directly) as an input and returns the four critical values listed above (passband ripple, etc.).

```
%lpfirstats.m
function [ripple, minpass, maxstoplo, maxstophi] = lpfirstats(H, W)
    % LPFIRSTATS Calculate filter performance metrics
    % H: Frequency response magnitude
    % W: Frequency vector (optional)

    if nargin < 2
        W = linspace(0, 1, length(H)); % Assume normalized frequency range [0, 1]
    end

    % Passband Ripple (Below 12 MHz)
    passband_range = W <= 0.24 * pi; % Normalized frequency range
    passband = H(passband_range);
    ripple = 20*log10(max(passband)) - 20*log10(min(passband));

    % Passband Max Attenuation (Below 18 MHz)
    passband2_range = W <= 0.36 * pi;
    passband2 = H(passband2_range);
    minpass = 20*log10(min(passband2));

    % Stopband Max Attenuation (Above 26 MHz)
    stopbandlo_range = W >= 0.52 * pi;
    stopbandlo = H(stopbandlo_range);
    maxstoplo = 20*log10(max(stopbandlo));

    % Stopband Max Attenuation (Above 34 MHz)
    stopbandhi_range = W >= 0.68 * pi;
    stopbandhi = H(stopbandhi_range);
    maxstophi = 20*log10(max(stopbandhi));
end
```

b) Either by hand or with a matlab function, repeatedly call lpfirstats(H,W) to find a reasonable small area filter. There is no need to write a sophisticated optimization algorithm, just something reasonable that does more than simple coefficient scaling. For example, making small perturbations to the frequency and amplitude values that remez() uses such as using 0.01 and other small values instead of 0.00 in the stopband.

```
%prob3.m
freqs = [0 0.24 0.36 0.52 0.68 1];
amps = [0.99 1 0.99 0.01 0. 0.01];
numtaps = 17;
scale = 2^11;
coeffs1   = firpm(numtaps-1, freqs, amps);
coeffs2   = coeffs1*scale;
coeffs    = round(coeffs2);
[H,W]     = freqz(coeffs);
H_norm    = abs(H) ./ abs(H(1));
[ripple, minpass, maxstoplo, maxstophi] = lpfirstats(H_norm,W);
fprintf("ripple: %f, minpass: %f, maxstoplo: %f, maxstophi: %f\n", ...
    ripple, minpass, maxstoplo, maxstophi);
plot_one_lpfir(coeffs)
stem(coeffs)
```

c) Provide the following for your smallest-area filter in your paper submission.

i) Filter coefficients

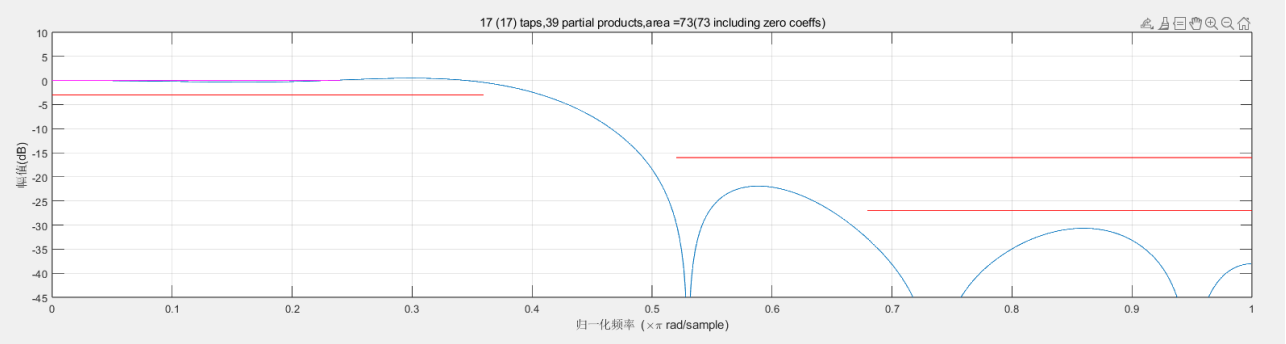
```
%coeffs.m
[-24 -2 66 80 -107 -191 128 638 898 638 128 -191 -107 80 66 -2 -24]
```

ii) The number of taps, number of required partial products, area estimate, and the attained values for the four filter criteria in dB.

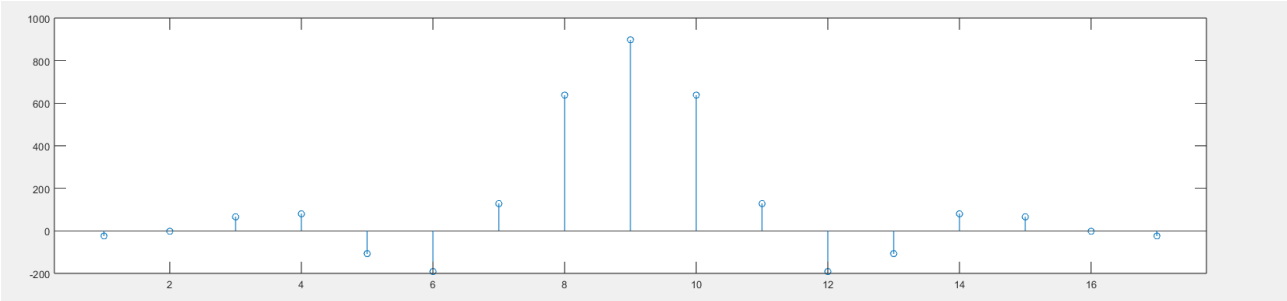
17 (17) taps,39 partial products,area =73(73 including zero coeffs)

ripple: 0.420367 dB, minpass: -0.390044 dB, maxstoplo: -21.911612 dB, maxstophi: -30.667907 dB

iii) A plot made by: plot_one_lpfir.m (that requires updating) to show the filter's frequency response.



iv) A stem() plot of the filter's coefficients.



d) Include in your submission:

i) Your modified version of plot_one_lpfir.m

```
% plot_one_lpfir.m
function plot_one_lpfir(coeffs)
   % Set these values (MHz), and also the "axis" parameters below
   FreqSamp       = 100;
   Ripple         = 12;   Ripple_dB =  3;  % placed at 0 dB for no good reason
   Pass1          = 18;   Pass1_dB =  -3;
   Stop1          = 26;   Stop1_dB = -16;
   Stop2          = 34;   Stop2_dB = -27;
   AxisValues     = [0 1 -45 10];

   % Probably do not change this constant
   NumSamplesFreqz = 2^11;                % 512 is default of freqz()

   figure(1); clf;

   coeffs_scaled   = coeffs/sum(coeffs);    % scale coeffs so H(0)=1
   len            = length(coeffs);

   freqz(coeffs_scaled, 1, NumSamplesFreqz);% plot freq response
   hold on;                                % don't erase following plots
   % only a single line for passband ripple because the vertical location
   % depends on the exact filter characteristics
   plot([0     Ripple]     /(FreqSamp/2), [Ripple_dB Ripple_dB], 'm'); % ripple
   plot([0     Pass1]      /(FreqSamp/2), [Pass1_dB Pass1_dB], 'r');  % passbnd
   plot([Stop1 (FreqSamp/2)]/(FreqSamp/2), [Stop1_dB Stop1_dB], 'r');  % stop1
   plot([Stop2 (FreqSamp/2)]/(FreqSamp/2), [Stop2_dB Stop2_dB], 'r');  % stop2

   axis(AxisValues);                       % set window view

   % Calculate number of partial products
   numpps = 0;
   for k = 1:len,
      numpps = numpps + numppterms(coeffs(k));
   end

   % Calculate area and areaeff
   area    = numpps + 2*len;
   leneff  = len;
   while (coeffs(1) == 0 & coeffs(leneff) == 0),
      coeffs = coeffs(2:(leneff-1));
      leneff = leneff -2;
   end
   areaeff   = numpps + 2*leneff;
```

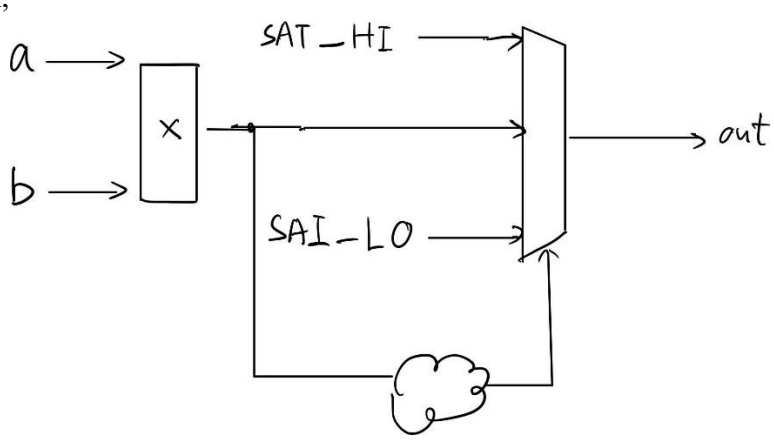
```
   % Generate title on plot
   titlestr = strcat(num2str(leneff), ' (',                    ...
                    num2str(len), ') taps, ',                 ...
                    num2str(numpps), ' partial products,',     ...
                   'area = ', num2str(areaeff),               ...
                   '(', num2str(area), ' including zero coeffs)' );
   title(titlestr);
   hold off;

   % Put title message on Phase Plot
   subplot(2,1,2);
   title('Ignore this phase plot');
   xlabel(' ');
   ylabel(' ');
return;
```

ii) Filter coefficients for your smallest-area filter in a copy-and-pasteable matlab vector;

see c) i)

4. Design a circuit that multiplies two 4-bit 2's complement inputs and saturates the product to 6 bits. Submit the following:

1) a circuit diagram,

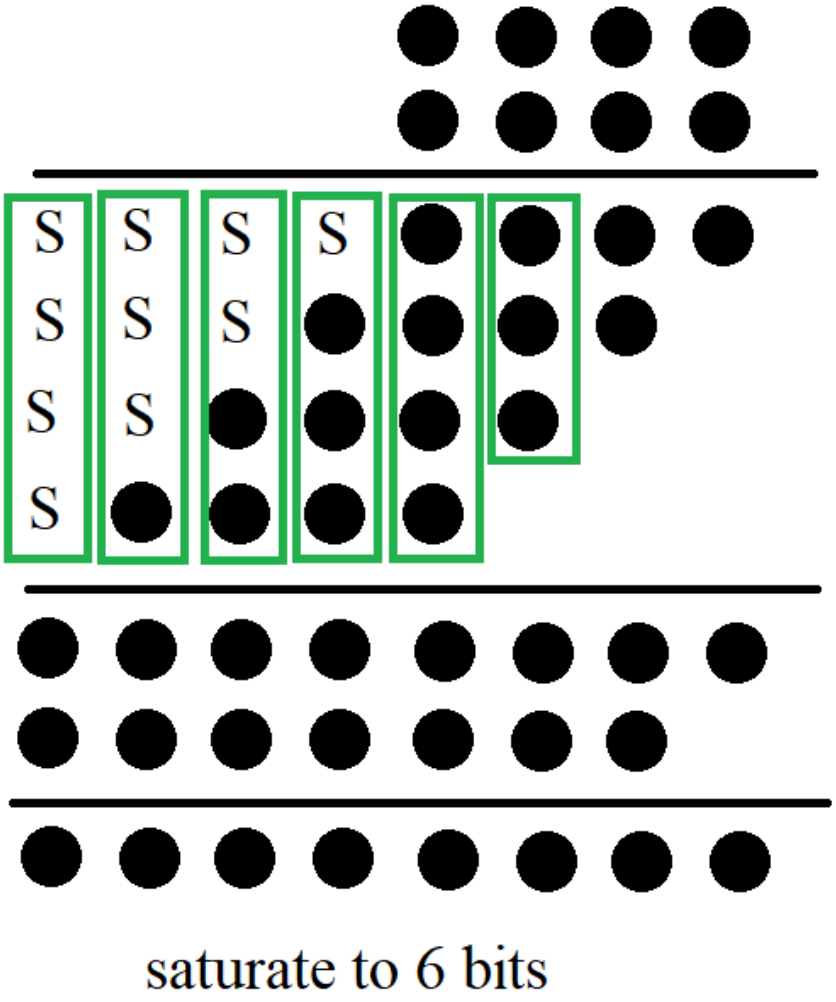


2) calculate the range of the full unsaturated output, and the range of the actual output,

Unsaturated output: [-56, 64]

Actual output: [-32, 31]

3) a dot diagram,



4) verilog for the design using "*" for the multiplier and "+" for the adder(s),

```
// prob4.v

`timescale 10ps/1ps
`celldefine
module prob4(
    a,
    b,
    c
);
    input [3:0]    a;
    input [3:0]    b;
    output reg [5:0]   c;
    wire [7:0]      c_pre;

    assign c_pre = {{4{a[3]}}, a} * {{4{b[3]}}, b};
    always @(*) begin
        if(c_pre[7 : 5] == 3'b000 || c_pre[7 : 5] == 3'b111)
            c = c_pre[5 : 0];
        else if(c_pre[7] == 1'b0)
            c = 6'b011111;
        else
            c = 6'b100000;
    end
endmodule
`endcelldefine
```

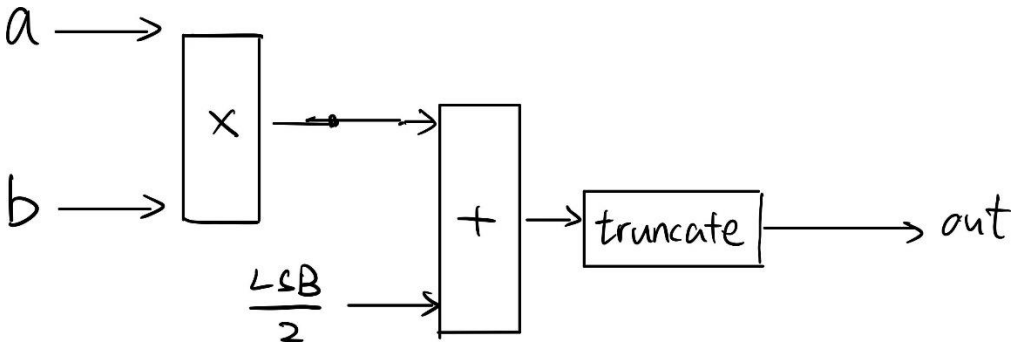
5) test your verilog design with at least 15 test cases including all extreme input cases, and verify using method ***(1).

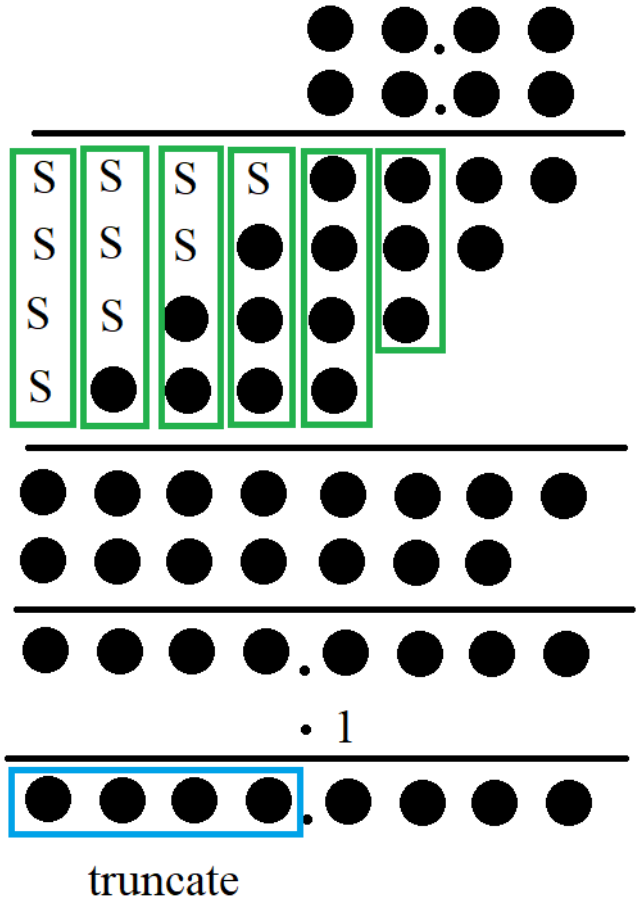
```
[will02@coe-ece-2107-31 hw3]$ make run
ncverilog +access+r -l prob4.logv -f prob4.vfv
ncverilog(64): 15.20-s031: (c) Copyright 1995-2017 Cadence Design Systems, Inc.
Loading snapshot worklib.prob4_tbench:v ..................... Done
ncsim> source /software/Cadence/INCISIVE152/tools/inca/files/ncsimrc
ncsim> run
Time: 0 | a = x, b = x, out_hw = x)       correct
Time: 5100 | a = 0, b = 0, out_hw = 0)    correct
Time: 5200 | a = -1, b = 1, out_hw = -1)  correct
Time: 5300 | a = -8, b = 7, out_hw = -32) correct
Time: 5400 | a = 7, b = 7, out_hw = 31)   correct
Time: 5500 | a = -8, b = -8, out_hw = 31) correct
Time: 5600 | a = 2, b = 4, out_hw = 8)    correct
Time: 5700 | a = 2, b = -8, out_hw = -16) correct
Time: 5800 | a = 6, b = 3, out_hw = 18)   correct
Time: 5900 | a = -4, b = 3, out_hw = -12) correct
Time: 6000 | a = -4, b = 5, out_hw = -20) correct
Time: 6100 | a = -6, b = 5, out_hw = -30) correct
Time: 6200 | a = -2, b = 1, out_hw = -2)  correct
Time: 6300 | a = -2, b = 2, out_hw = -4)  correct
Time: 6400 | a = -2, b = 4, out_hw = -8)  correct
Time: 6500 | a = 5, b = 5, out_hw = 25)   correct
Simulation complete via $finish(1) at time 7500 PS + 0
./prob4_tbench.v:27        $finish;
ncsim> exit
```

```
// prob4_tbench.v
`timescale 10ps/1ps
module prob4_tbench;
    reg [3:0]  a;
    reg [3:0]  b;
    wire [5:0] c;
    integer i, seed;
    prob4 _prob4(.a(a), .b(b), .c(c));
    initial begin
        #500;
        #10 a = 4'b0000; b = 4'b0000;
        #10 a = 4'b1111; b = 4'b0001;
        #10 a = 4'b1000; b = 4'b0111;
        #10 a = 4'b0111; b = 4'b0111;
        #10 a = 4'b1000; b = 4'b1000;
        #10 a = 4'b0010; b = 4'b0100;
        #10 a = 4'b0010; b = 4'b1000;
        #10 a = 4'b0110; b = 4'b0011;
        #10 a = 4'b1100; b = 4'b0011;
        #10 a = 4'b1100; b = 4'b0101;
        #10 a = 4'b1010; b = 4'b0101;
        #10 a = 4'b1110; b = 4'b0001;
        #10 a = 4'b1110; b = 4'b0010;
        #10 a = 4'b1110; b = 4'b0100;
        #10 a = 4'b0101; b = 4'b0101;
        #100
        $finish;
    end

    initial begin
        $monitor("Time: %0t | a = %0d, b = %0d, out_hw = %0d)",
                    $time, $signed(a), $signed(b), $signed(c));
    end

endmodule
```

5. Repeat the previous problem but instead of saturating the output, round it to a 4-bit output using the "add 1/2 LSB and truncate" method. The output may never overflow or underflow.

1,3,4,5) same as the previous problem





```
// prob5.v

`timescale 10ps/1ps
`celldefine
module prob5(
    a,
    b,
    c
);
    input [3:0]    a;
    input [3:0]    b;
    output reg [3:0]   c;
    wire [7:0]     c_pre;

    assign c_pre = {{4{a[3]}}, a} * {{4{b[3]}}, b} + {1'b1,{3{1'b0}}};
    always @(*) begin
        c = c_pre[7 : 4];
        // $display("%d", c);
    end
endmodule
`endcelldefine
```

```
[will02@coe-ece-2107-31 hw3]$ make run
ncverilog +access+r -l prob5.logv -f prob5.vfv
ncverilog(64): 15.20-s031: (c) Copyright 1995-2017 Cadence Design Systems, Inc.
Loading snapshot worklib.prob5_tbench:v ..................... Done
ncsim> source /software/Cadence/INCISIVE152/tools/inca/files/ncsimrc
ncsim> run
Time: 0 | a = 0.00, b = 0.00, out_hw = x)      correct
Time: 5100 | a = 0.00, b = 0.00, out_hw = 0)   correct
Time: 5200 | a = -0.25, b = 0.25, out_hw = 0)  correct
Time: 5300 | a = -2.00, b = 1.75, out_hw = -3) correct
Time: 5400 | a = 1.75, b = 1.75, out_hw = 3)   correct
Time: 5500 | a = -2.00, b = -2.00, out_hw = 4) correct
Time: 5600 | a = 0.50, b = 1.00, out_hw = 1)   correct
Time: 5700 | a = 0.50, b = -2.00, out_hw = -1) correct
Time: 5800 | a = 1.50, b = 0.75, out_hw = 1)   correct
Time: 5900 | a = -1.00, b = 0.75, out_hw = -1) correct
Time: 6000 | a = -1.00, b = 1.25, out_hw = -1) correct
Time: 6100 | a = -1.50, b = 1.25, out_hw = -2) correct
Time: 6200 | a = -0.50, b = 0.25, out_hw = 0)  correct
Time: 6300 | a = -0.50, b = 0.50, out_hw = 0)  correct
Time: 6400 | a = -0.50, b = 1.00, out_hw = 0)  correct
Time: 6500 | a = 1.25, b = 1.25, out_hw = 2)   correct
Simulation complete via $finish(1) at time 7500 PS + 0
./prob5_tbench.v:27        $finish;
ncsim> exit
```

```
// prob5_tbench.v
`timescale 10ps/1ps
module prob5_tbench;
    reg [3:0]  a;
    reg [3:0]  b;
    wire [3:0] c;
    integer i, seed;
    prob5 _prob5(.a(a), .b(b), .c(c));
    initial begin
        #500;
        #10 a = 4'b0000; b = 4'b0000;
        #10 a = 4'b1111; b = 4'b0001;
        #10 a = 4'b1000; b = 4'b0111;
        #10 a = 4'b0111; b = 4'b0111;
        #10 a = 4'b1000; b = 4'b1000;
        #10 a = 4'b0010; b = 4'b0100;
        #10 a = 4'b0010; b = 4'b1000;
        #10 a = 4'b0110; b = 4'b0011;
        #10 a = 4'b1100; b = 4'b0011;
        #10 a = 4'b1100; b = 4'b0101;
        #10 a = 4'b1010; b = 4'b0101;
        #10 a = 4'b1110; b = 4'b0001;
        #10 a = 4'b1110; b = 4'b0010;
        #10 a = 4'b1110; b = 4'b0100;
        #10 a = 4'b0101; b = 4'b0101;
        #100
        $finish;
    end

    initial begin
        $monitor("Time: %0t | a = %2.2f, b = %2.2f, out_hw = %0d)",
                    $time, $signed(a) / 4.0, $signed(b) / 4.0, $signed(c));
    end

endmodule
```

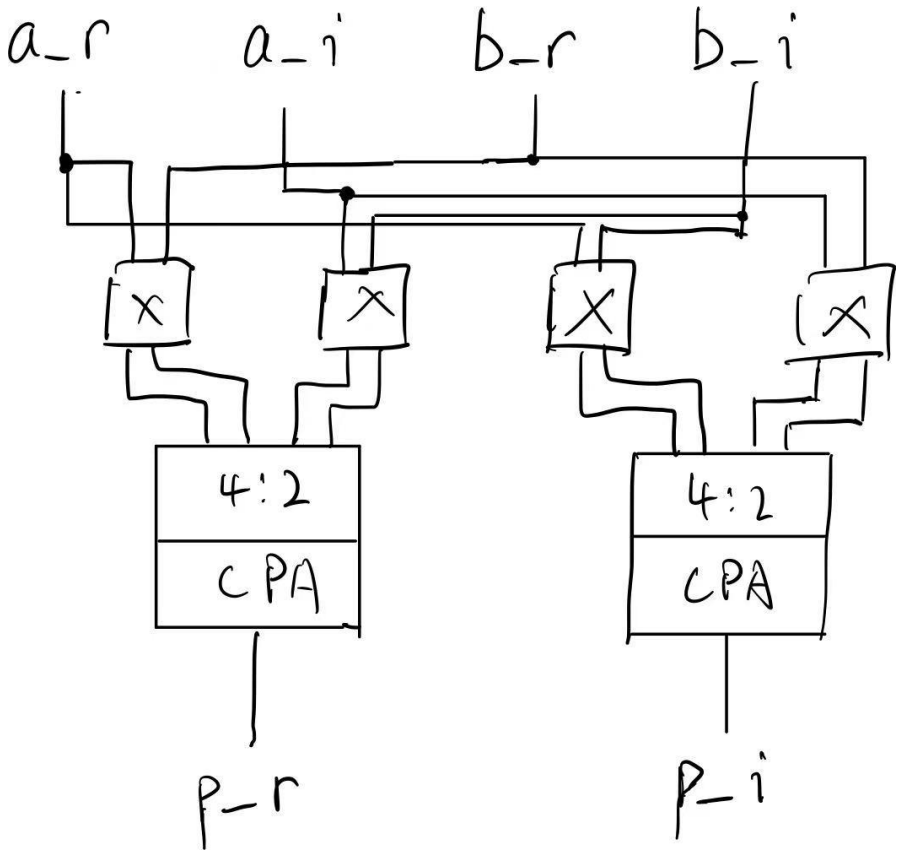
2) calculate the range of the full unrounded output, and the range of the actual output,

Unrounded output: [-3.5, 4]

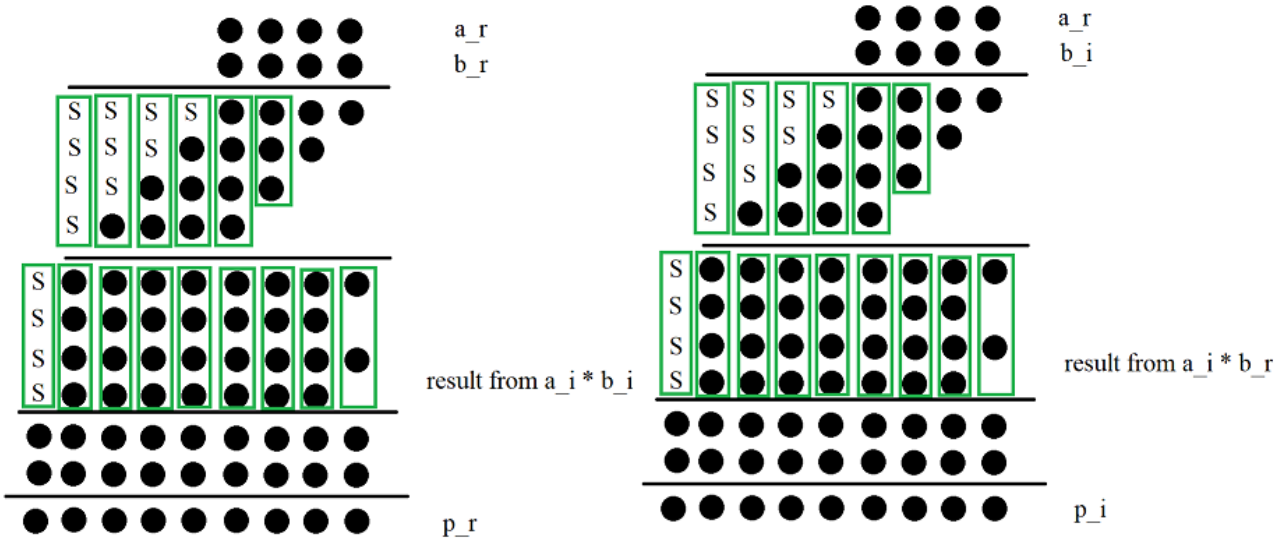
Actual output: [-3, 4]

6. Design a complex 2's complement multiplier p = a × b with 4-bit inputs a_r, a_i, b_r, b_i, and 9-bit outputs p_r, p_i. Register all inputs and outputs. Submit the following:

1) a block diagram,



2) a dot diagram,



3) verilog for the design using "*" for the multipliers and "+" for the adders,

```
// prob6.v

`timescale 10ps/1ps
`celldefine
module prob6(
    clk,
    a_r,
    a_i,
    b_r,
    b_i,
    p_r,
    p_i
);
    input               clk;
    input [3:0]         a_r, a_i;
    input [3:0]         b_r, b_i;
    output reg [8:0]    p_r, p_i;
    reg [3:0]           a_r_r, a_i_r, b_r_r, b_i_r;
    wire [8:0]          p_r_c, p_i_c;

    assign p_r_c = {{5{a_r_r[3]}}, a_r_r} * {{5{b_r_r[3]}}, b_r_r} -
{{5{a_i_r[3]}}, a_i_r} * {{5{b_i_r[3]}}, b_i_r};
    assign p_i_c = {{5{a_r_r[3]}}, a_r_r} * {{5{b_i_r[3]}}, b_i_r} +
{{5{a_i_r[3]}}, a_i_r} * {{5{b_r_r[3]}}, b_r_r};
    always @(posedge clk) begin
        a_r_r <= #1 a_r;
        a_i_r <= #1 a_i;
        b_r_r <= #1 b_r;
        b_i_r <= #1 b_i;
        p_r <= #1 p_r_c;
        p_i <= #1 p_i_c;
    end
endmodule
`endcelldefine
```

5) test your verilog design exhaustively over all 2^16 possible inputs, verifying using the Golden Reference method. State how many errors you have. Submit approximately 100 passing test cases copied & pasted from the most important sections of your exhaustive output, in addition to the required failing case after you purposely make a hardware error: ***(3).

No errors.

Test 65435: a_r = -6, a_i = -7, b_r = -1, b_i = -1, p_r = -1, p_i = 13, p_r_ref = -1, p_i_ref = 13, correct)

Test 65436: a_r = -5, a_i = -7, b_r = -1, b_i = -1, p_r = -2, p_i = 12, p_r_ref = -2, p_i_ref = 12, correct)

Test 65437: a_r = -4, a_i = -7, b_r = -1, b_i = -1, p_r = -3, p_i = 11, p_r_ref = -3, p_i_ref = 11, correct)

Test 65438: a_r = -3, a_i = -7, b_r = -1, b_i = -1, p_r = -4, p_i = 10, p_r_ref = -4, p_i_ref = 10, correct)
Test 65439: a_r = -2, a_i = -7, b_r = -1, b_i = -1, p_r = -5, p_i = 9, p_r_ref = -5, p_i_ref = 9, correct)
Test 65440: a_r = -1, a_i = -7, b_r = -1, b_i = -1, p_r = -6, p_i = 8, p_r_ref = -6, p_i_ref = 8, correct)
Test 65441: a_r = 0, a_i = -6, b_r = -1, b_i = -1, p_r = -6, p_i = 6, p_r_ref = -6, p_i_ref = 6, correct)
Test 65442: a_r = 1, a_i = -6, b_r = -1, b_i = -1, p_r = -7, p_i = 5, p_r_ref = -7, p_i_ref = 5, correct)
Test 65443: a_r = 2, a_i = -6, b_r = -1, b_i = -1, p_r = -8, p_i = 4, p_r_ref = -8, p_i_ref = 4, correct)
Test 65444: a_r = 3, a_i = -6, b_r = -1, b_i = -1, p_r = -9, p_i = 3, p_r_ref = -9, p_i_ref = 3, correct)
Test 65445: a_r = 4, a_i = -6, b_r = -1, b_i = -1, p_r = -10, p_i = 2, p_r_ref = -10, p_i_ref = 2, correct)
Test 65446: a_r = 5, a_i = -6, b_r = -1, b_i = -1, p_r = -11, p_i = 1, p_r_ref = -11, p_i_ref = 1, correct)
Test 65447: a_r = 6, a_i = -6, b_r = -1, b_i = -1, p_r = -12, p_i = 0, p_r_ref = -12, p_i_ref = 0, correct)
Test 65448: a_r = 7, a_i = -6, b_r = -1, b_i = -1, p_r = -13, p_i = -1, p_r_ref = -13, p_i_ref = -1, correct)
Test 65449: a_r = -8, a_i = -6, b_r = -1, b_i = -1, p_r = 2, p_i = 14, p_r_ref = 2, p_i_ref = 14, correct)
Test 65450: a_r = -7, a_i = -6, b_r = -1, b_i = -1, p_r = 1, p_i = 13, p_r_ref = 1, p_i_ref = 13, correct)
Test 65451: a_r = -6, a_i = -6, b_r = -1, b_i = -1, p_r = 0, p_i = 12, p_r_ref = 0, p_i_ref = 12, correct)
Test 65452: a_r = -5, a_i = -6, b_r = -1, b_i = -1, p_r = -1, p_i = 11, p_r_ref = -1, p_i_ref = 11, correct)
Test 65453: a_r = -4, a_i = -6, b_r = -1, b_i = -1, p_r = -2, p_i = 10, p_r_ref = -2, p_i_ref = 10, correct)
Test 65454: a_r = -3, a_i = -6, b_r = -1, b_i = -1, p_r = -3, p_i = 9, p_r_ref = -3, p_i_ref = 9, correct)
Test 65455: a_r = -2, a_i = -6, b_r = -1, b_i = -1, p_r = -4, p_i = 8, p_r_ref = -4, p_i_ref = 8, correct)
Test 65456: a_r = -1, a_i = -6, b_r = -1, b_i = -1, p_r = -5, p_i = 7, p_r_ref = -5, p_i_ref = 7, correct)
Test 65457: a_r = 0, a_i = -5, b_r = -1, b_i = -1, p_r = -5, p_i = 5, p_r_ref = -5, p_i_ref = 5, correct)
Test 65458: a_r = 1, a_i = -5, b_r = -1, b_i = -1, p_r = -6, p_i = 4, p_r_ref = -6, p_i_ref = 4, correct)
Test 65459: a_r = 2, a_i = -5, b_r = -1, b_i = -1, p_r = -7, p_i = 3, p_r_ref = -7, p_i_ref = 3, correct)
Test 65460: a_r = 3, a_i = -5, b_r = -1, b_i = -1, p_r = -8, p_i = 2, p_r_ref = -8, p_i_ref = 2, correct)
Test 65461: a_r = 4, a_i = -5, b_r = -1, b_i = -1, p_r = -9, p_i = 1, p_r_ref = -9, p_i_ref = 1, correct)
Test 65462: a_r = 5, a_i = -5, b_r = -1, b_i = -1, p_r = -10, p_i = 0, p_r_ref = -10, p_i_ref = 0, correct)
Test 65463: a_r = 6, a_i = -5, b_r = -1, b_i = -1, p_r = -11, p_i = -1, p_r_ref = -11, p_i_ref = -1, correct)
Test 65464: a_r = 7, a_i = -5, b_r = -1, b_i = -1, p_r = -12, p_i = -2, p_r_ref = -12, p_i_ref = -2, correct)
Test 65465: a_r = -8, a_i = -5, b_r = -1, b_i = -1, p_r = 3, p_i = 13, p_r_ref = 3, p_i_ref = 13, correct)
Test 65466: a_r = -7, a_i = -5, b_r = -1, b_i = -1, p_r = 2, p_i = 12, p_r_ref = 2, p_i_ref = 12, correct)
Test 65467: a_r = -6, a_i = -5, b_r = -1, b_i = -1, p_r = 1, p_i = 11, p_r_ref = 1, p_i_ref = 11, correct)
Test 65468: a_r = -5, a_i = -5, b_r = -1, b_i = -1, p_r = 0, p_i = 10, p_r_ref = 0, p_i_ref = 10, correct)
Test 65469: a_r = -4, a_i = -5, b_r = -1, b_i = -1, p_r = -1, p_i = 9, p_r_ref = -1, p_i_ref = 9, correct)
Test 65470: a_r = -3, a_i = -5, b_r = -1, b_i = -1, p_r = -2, p_i = 8, p_r_ref = -2, p_i_ref = 8, correct)
Test 65471: a_r = -2, a_i = -5, b_r = -1, b_i = -1, p_r = -3, p_i = 7, p_r_ref = -3, p_i_ref = 7, correct)
Test 65472: a_r = -1, a_i = -5, b_r = -1, b_i = -1, p_r = -4, p_i = 6, p_r_ref = -4, p_i_ref = 6, correct)
Test 65473: a_r = 0, a_i = -4, b_r = -1, b_i = -1, p_r = -4, p_i = 4, p_r_ref = -4, p_i_ref = 4, correct)
Test 65474: a_r = 1, a_i = -4, b_r = -1, b_i = -1, p_r = -5, p_i = 3, p_r_ref = -5, p_i_ref = 3, correct)
Test 65475: a_r = 2, a_i = -4, b_r = -1, b_i = -1, p_r = -6, p_i = 2, p_r_ref = -6, p_i_ref = 2, correct)
Test 65476: a_r = 3, a_i = -4, b_r = -1, b_i = -1, p_r = -7, p_i = 1, p_r_ref = -7, p_i_ref = 1, correct)
Test 65477: a_r = 4, a_i = -4, b_r = -1, b_i = -1, p_r = -8, p_i = 0, p_r_ref = -8, p_i_ref = 0, correct)

Test 65478: a_r = 5, a_i = -4, b_r = -1, b_i = -1, p_r = -9, p_i = -1, p_r_ref = -9, p_i_ref = -1, correct)
Test 65479: a_r = 6, a_i = -4, b_r = -1, b_i = -1, p_r = -10, p_i = -2, p_r_ref = -10, p_i_ref = -2, correct)
Test 65480: a_r = 7, a_i = -4, b_r = -1, b_i = -1, p_r = -11, p_i = -3, p_r_ref = -11, p_i_ref = -3, correct)
Test 65481: a_r = -8, a_i = -4, b_r = -1, b_i = -1, p_r = 4, p_i = 12, p_r_ref = 4, p_i_ref = 12, correct)
Test 65482: a_r = -7, a_i = -4, b_r = -1, b_i = -1, p_r = 3, p_i = 11, p_r_ref = 3, p_i_ref = 11, correct)
Test 65483: a_r = -6, a_i = -4, b_r = -1, b_i = -1, p_r = 2, p_i = 10, p_r_ref = 2, p_i_ref = 10, correct)
Test 65484: a_r = -5, a_i = -4, b_r = -1, b_i = -1, p_r = 1, p_i = 9, p_r_ref = 1, p_i_ref = 9, correct)
Test 65485: a_r = -4, a_i = -4, b_r = -1, b_i = -1, p_r = 0, p_i = 8, p_r_ref = 0, p_i_ref = 8, correct)
Test 65486: a_r = -3, a_i = -4, b_r = -1, b_i = -1, p_r = -1, p_i = 7, p_r_ref = -1, p_i_ref = 7, correct)
Test 65487: a_r = -2, a_i = -4, b_r = -1, b_i = -1, p_r = -2, p_i = 6, p_r_ref = -2, p_i_ref = 6, correct)
Test 65488: a_r = -1, a_i = -4, b_r = -1, b_i = -1, p_r = -3, p_i = 5, p_r_ref = -3, p_i_ref = 5, correct)
Test 65489: a_r = 0, a_i = -3, b_r = -1, b_i = -1, p_r = -3, p_i = 3, p_r_ref = -3, p_i_ref = 3, correct)
Test 65490: a_r = 1, a_i = -3, b_r = -1, b_i = -1, p_r = -4, p_i = 2, p_r_ref = -4, p_i_ref = 2, correct)
Test 65491: a_r = 2, a_i = -3, b_r = -1, b_i = -1, p_r = -5, p_i = 1, p_r_ref = -5, p_i_ref = 1, correct)
Test 65492: a_r = 3, a_i = -3, b_r = -1, b_i = -1, p_r = -6, p_i = 0, p_r_ref = -6, p_i_ref = 0, correct)
Test 65493: a_r = 4, a_i = -3, b_r = -1, b_i = -1, p_r = -7, p_i = -1, p_r_ref = -7, p_i_ref = -1, correct)
Test 65494: a_r = 5, a_i = -3, b_r = -1, b_i = -1, p_r = -8, p_i = -2, p_r_ref = -8, p_i_ref = -2, correct)
Test 65495: a_r = 6, a_i = -3, b_r = -1, b_i = -1, p_r = -9, p_i = -3, p_r_ref = -9, p_i_ref = -3, correct)
Test 65496: a_r = 7, a_i = -3, b_r = -1, b_i = -1, p_r = -10, p_i = -4, p_r_ref = -10, p_i_ref = -4, correct)
Test 65497: a_r = -8, a_i = -3, b_r = -1, b_i = -1, p_r = 5, p_i = 11, p_r_ref = 5, p_i_ref = 11, correct)
Test 65498: a_r = -7, a_i = -3, b_r = -1, b_i = -1, p_r = 4, p_i = 10, p_r_ref = 4, p_i_ref = 10, correct)
Test 65499: a_r = -6, a_i = -3, b_r = -1, b_i = -1, p_r = 3, p_i = 9, p_r_ref = 3, p_i_ref = 9, correct)
Test 65500: a_r = -5, a_i = -3, b_r = -1, b_i = -1, p_r = 2, p_i = 8, p_r_ref = 2, p_i_ref = 8, correct)
Test 65501: a_r = -4, a_i = -3, b_r = -1, b_i = -1, p_r = 1, p_i = 7, p_r_ref = 1, p_i_ref = 7, correct)
Test 65502: a_r = -3, a_i = -3, b_r = -1, b_i = -1, p_r = 0, p_i = 6, p_r_ref = 0, p_i_ref = 6, correct)
Test 65503: a_r = -2, a_i = -3, b_r = -1, b_i = -1, p_r = -1, p_i = 5, p_r_ref = -1, p_i_ref = 5, correct)
Test 65504: a_r = -1, a_i = -3, b_r = -1, b_i = -1, p_r = -2, p_i = 4, p_r_ref = -2, p_i_ref = 4, correct)
Test 65505: a_r = 0, a_i = -2, b_r = -1, b_i = -1, p_r = -2, p_i = 2, p_r_ref = -2, p_i_ref = 2, correct)
Test 65506: a_r = 1, a_i = -2, b_r = -1, b_i = -1, p_r = -3, p_i = 1, p_r_ref = -3, p_i_ref = 1, correct)
Test 65507: a_r = 2, a_i = -2, b_r = -1, b_i = -1, p_r = -4, p_i = 0, p_r_ref = -4, p_i_ref = 0, correct)
Test 65508: a_r = 3, a_i = -2, b_r = -1, b_i = -1, p_r = -5, p_i = -1, p_r_ref = -5, p_i_ref = -1, correct)
Test 65509: a_r = 4, a_i = -2, b_r = -1, b_i = -1, p_r = -6, p_i = -2, p_r_ref = -6, p_i_ref = -2, correct)
Test 65510: a_r = 5, a_i = -2, b_r = -1, b_i = -1, p_r = -7, p_i = -3, p_r_ref = -7, p_i_ref = -3, correct)
Test 65511: a_r = 6, a_i = -2, b_r = -1, b_i = -1, p_r = -8, p_i = -4, p_r_ref = -8, p_i_ref = -4, correct)
Test 65512: a_r = 7, a_i = -2, b_r = -1, b_i = -1, p_r = -9, p_i = -5, p_r_ref = -9, p_i_ref = -5, correct)
Test 65513: a_r = -8, a_i = -2, b_r = -1, b_i = -1, p_r = 6, p_i = 10, p_r_ref = 6, p_i_ref = 10, correct)
Test 65514: a_r = -7, a_i = -2, b_r = -1, b_i = -1, p_r = 5, p_i = 9, p_r_ref = 5, p_i_ref = 9, correct)
Test 65515: a_r = -6, a_i = -2, b_r = -1, b_i = -1, p_r = 4, p_i = 8, p_r_ref = 4, p_i_ref = 8, correct)
Test 65516: a_r = -5, a_i = -2, b_r = -1, b_i = -1, p_r = 3, p_i = 7, p_r_ref = 3, p_i_ref = 7, correct)
Test 65517: a_r = -4, a_i = -2, b_r = -1, b_i = -1, p_r = 2, p_i = 6, p_r_ref = 2, p_i_ref = 6, correct)

Test 65518: a_r = -3, a_i = -2, b_r = -1, b_i = -1, p_r = 1, p_i = 5, p_r_ref = 1, p_i_ref = 5, correct)
Test 65519: a_r = -2, a_i = -2, b_r = -1, b_i = -1, p_r = 0, p_i = 4, p_r_ref = 0, p_i_ref = 4, correct)
Test 65520: a_r = -1, a_i = -2, b_r = -1, b_i = -1, p_r = -1, p_i = 3, p_r_ref = -1, p_i_ref = 3, correct)
Test 65521: a_r = 0, a_i = -1, b_r = -1, b_i = -1, p_r = -1, p_i = 1, p_r_ref = -1, p_i_ref = 1, correct)
Test 65522: a_r = 1, a_i = -1, b_r = -1, b_i = -1, p_r = -2, p_i = 0, p_r_ref = -2, p_i_ref = 0, correct)
Test 65523: a_r = 2, a_i = -1, b_r = -1, b_i = -1, p_r = -3, p_i = -1, p_r_ref = -3, p_i_ref = -1, correct)
Test 65524: a_r = 3, a_i = -1, b_r = -1, b_i = -1, p_r = -4, p_i = -2, p_r_ref = -4, p_i_ref = -2, correct)
Test 65525: a_r = 4, a_i = -1, b_r = -1, b_i = -1, p_r = -5, p_i = -3, p_r_ref = -5, p_i_ref = -3, correct)
Test 65526: a_r = 5, a_i = -1, b_r = -1, b_i = -1, p_r = -6, p_i = -4, p_r_ref = -6, p_i_ref = -4, correct)
Test 65527: a_r = 6, a_i = -1, b_r = -1, b_i = -1, p_r = -7, p_i = -5, p_r_ref = -7, p_i_ref = -5, correct)
Test 65528: a_r = 7, a_i = -1, b_r = -1, b_i = -1, p_r = -8, p_i = -6, p_r_ref = -8, p_i_ref = -6, correct)
Test 65529: a_r = -8, a_i = -1, b_r = -1, b_i = -1, p_r = 7, p_i = 9, p_r_ref = 7, p_i_ref = 9, correct)
Test 65530: a_r = -7, a_i = -1, b_r = -1, b_i = -1, p_r = 6, p_i = 8, p_r_ref = 6, p_i_ref = 8, correct)
Test 65531: a_r = -6, a_i = -1, b_r = -1, b_i = -1, p_r = 5, p_i = 7, p_r_ref = 5, p_i_ref = 7, correct)
Test 65532: a_r = -5, a_i = -1, b_r = -1, b_i = -1, p_r = 4, p_i = 6, p_r_ref = 4, p_i_ref = 6, correct)
Test 65533: a_r = -4, a_i = -1, b_r = -1, b_i = -1, p_r = 3, p_i = 5, p_r_ref = 3, p_i_ref = 5, correct)
Test 65534: a_r = -3, a_i = -1, b_r = -1, b_i = -1, p_r = 2, p_i = 4, p_r_ref = 2, p_i_ref = 4, correct)
Test 65535: a_r = -2, a_i = -1, b_r = -1, b_i = -1, p_r = 1, p_i = 3, p_r_ref = 1, p_i_ref = 3, correct)

Purposely hardware error:
Test 65435: a_r = -6, a_i = -7, b_r = -1, b_i = -1, p_r = -1, p_i = 7, p_r_ref = -1, p_i_ref = 13, wrong)
Test 65436: a_r = -5, a_i = -7, b_r = -1, b_i = -1, p_r = -2, p_i = 6, p_r_ref = -2, p_i_ref = 12, wrong)
Test 65437: a_r = -4, a_i = -7, b_r = -1, b_i = -1, p_r = -3, p_i = 5, p_r_ref = -3, p_i_ref = 11, wrong)
Test 65438: a_r = -3, a_i = -7, b_r = -1, b_i = -1, p_r = -4, p_i = 4, p_r_ref = -4, p_i_ref = 10, wrong)
Test 65439: a_r = -2, a_i = -7, b_r = -1, b_i = -1, p_r = -5, p_i = 3, p_r_ref = -5, p_i_ref = 9, wrong)
Test 65440: a_r = -1, a_i = -7, b_r = -1, b_i = -1, p_r = -6, p_i = 2, p_r_ref = -6, p_i_ref = 8, wrong)
Test 65441: a_r = 0, a_i = -6, b_r = -1, b_i = -1, p_r = -6, p_i = 1, p_r_ref = -6, p_i_ref = 6, wrong)
Test 65442: a_r = 1, a_i = -6, b_r = -1, b_i = -1, p_r = -7, p_i = 0, p_r_ref = -7, p_i_ref = 5, wrong)
Test 65443: a_r = 2, a_i = -6, b_r = -1, b_i = -1, p_r = -8, p_i = -1, p_r_ref = -8, p_i_ref = 4, wrong)
Test 65444: a_r = 3, a_i = -6, b_r = -1, b_i = -1, p_r = -9, p_i = -2, p_r_ref = -9, p_i_ref = 3, wrong)
Test 65445: a_r = 4, a_i = -6, b_r = -1, b_i = -1, p_r = -10, p_i = -3, p_r_ref = -10, p_i_ref = 2, wrong)
Test 65446: a_r = 5, a_i = -6, b_r = -1, b_i = -1, p_r = -11, p_i = -4, p_r_ref = -11, p_i_ref = 1, wrong)
Test 65447: a_r = 6, a_i = -6, b_r = -1, b_i = -1, p_r = -12, p_i = -5, p_r_ref = -12, p_i_ref = 0, wrong)
Test 65448: a_r = 7, a_i = -6, b_r = -1, b_i = -1, p_r = -13, p_i = -6, p_r_ref = -13, p_i_ref = -1, wrong)
Test 65449: a_r = -8, a_i = -6, b_r = -1, b_i = -1, p_r = 2, p_i = 9, p_r_ref = 2, p_i_ref = 14, wrong)
Test 65450: a_r = -7, a_i = -6, b_r = -1, b_i = -1, p_r = 1, p_i = 8, p_r_ref = 1, p_i_ref = 13, wrong)
Test 65451: a_r = -6, a_i = -6, b_r = -1, b_i = -1, p_r = 0, p_i = 7, p_r_ref = 0, p_i_ref = 12, wrong)
Test 65452: a_r = -5, a_i = -6, b_r = -1, b_i = -1, p_r = -1, p_i = 6, p_r_ref = -1, p_i_ref = 11, wrong)
Test 65453: a_r = -4, a_i = -6, b_r = -1, b_i = -1, p_r = -2, p_i = 5, p_r_ref = -2, p_i_ref = 10, wrong)
Test 65454: a_r = -3, a_i = -6, b_r = -1, b_i = -1, p_r = -3, p_i = 4, p_r_ref = -3, p_i_ref = 9, wrong)

Test 65455: a_r = -2, a_i = -6, b_r = -1, b_i = -1, p_r = -4, p_i = 3, p_r_ref = -4, p_i_ref = 8, wrong)
Test 65456: a_r = -1, a_i = -6, b_r = -1, b_i = -1, p_r = -5, p_i = 2, p_r_ref = -5, p_i_ref = 7, wrong)
Test 65457: a_r = 0, a_i = -5, b_r = -1, b_i = -1, p_r = -5, p_i = 1, p_r_ref = -5, p_i_ref = 5, wrong)
Test 65458: a_r = 1, a_i = -5, b_r = -1, b_i = -1, p_r = -6, p_i = 0, p_r_ref = -6, p_i_ref = 4, wrong)
Test 65459: a_r = 2, a_i = -5, b_r = -1, b_i = -1, p_r = -7, p_i = -1, p_r_ref = -7, p_i_ref = 3, wrong)
Test 65460: a_r = 3, a_i = -5, b_r = -1, b_i = -1, p_r = -8, p_i = -2, p_r_ref = -8, p_i_ref = 2, wrong)
Test 65461: a_r = 4, a_i = -5, b_r = -1, b_i = -1, p_r = -9, p_i = -3, p_r_ref = -9, p_i_ref = 1, wrong)
Test 65462: a_r = 5, a_i = -5, b_r = -1, b_i = -1, p_r = -10, p_i = -4, p_r_ref = -10, p_i_ref = 0, wrong)
Test 65463: a_r = 6, a_i = -5, b_r = -1, b_i = -1, p_r = -11, p_i = -5, p_r_ref = -11, p_i_ref = -1, wrong)
Test 65464: a_r = 7, a_i = -5, b_r = -1, b_i = -1, p_r = -12, p_i = -6, p_r_ref = -12, p_i_ref = -2, wrong)
Test 65465: a_r = -8, a_i = -5, b_r = -1, b_i = -1, p_r = 3, p_i = 9, p_r_ref = 3, p_i_ref = 13, wrong)
Test 65466: a_r = -7, a_i = -5, b_r = -1, b_i = -1, p_r = 2, p_i = 8, p_r_ref = 2, p_i_ref = 12, wrong)
Test 65467: a_r = -6, a_i = -5, b_r = -1, b_i = -1, p_r = 1, p_i = 7, p_r_ref = 1, p_i_ref = 11, wrong)
Test 65468: a_r = -5, a_i = -5, b_r = -1, b_i = -1, p_r = 0, p_i = 6, p_r_ref = 0, p_i_ref = 10, wrong)
Test 65469: a_r = -4, a_i = -5, b_r = -1, b_i = -1, p_r = -1, p_i = 5, p_r_ref = -1, p_i_ref = 9, wrong)
Test 65470: a_r = -3, a_i = -5, b_r = -1, b_i = -1, p_r = -2, p_i = 4, p_r_ref = -2, p_i_ref = 8, wrong)
Test 65471: a_r = -2, a_i = -5, b_r = -1, b_i = -1, p_r = -3, p_i = 3, p_r_ref = -3, p_i_ref = 7, wrong)
Test 65472: a_r = -1, a_i = -5, b_r = -1, b_i = -1, p_r = -4, p_i = 2, p_r_ref = -4, p_i_ref = 6, wrong)
Test 65473: a_r = 0, a_i = -4, b_r = -1, b_i = -1, p_r = -4, p_i = 1, p_r_ref = -4, p_i_ref = 4, wrong)
Test 65474: a_r = 1, a_i = -4, b_r = -1, b_i = -1, p_r = -5, p_i = 0, p_r_ref = -5, p_i_ref = 3, wrong)
Test 65475: a_r = 2, a_i = -4, b_r = -1, b_i = -1, p_r = -6, p_i = -1, p_r_ref = -6, p_i_ref = 2, wrong)
Test 65476: a_r = 3, a_i = -4, b_r = -1, b_i = -1, p_r = -7, p_i = -2, p_r_ref = -7, p_i_ref = 1, wrong)
Test 65477: a_r = 4, a_i = -4, b_r = -1, b_i = -1, p_r = -8, p_i = -3, p_r_ref = -8, p_i_ref = 0, wrong)
Test 65478: a_r = 5, a_i = -4, b_r = -1, b_i = -1, p_r = -9, p_i = -4, p_r_ref = -9, p_i_ref = -1, wrong)
Test 65479: a_r = 6, a_i = -4, b_r = -1, b_i = -1, p_r = -10, p_i = -5, p_r_ref = -10, p_i_ref = -2, wrong)
Test 65480: a_r = 7, a_i = -4, b_r = -1, b_i = -1, p_r = -11, p_i = -6, p_r_ref = -11, p_i_ref = -3, wrong)
Test 65481: a_r = -8, a_i = -4, b_r = -1, b_i = -1, p_r = 4, p_i = 9, p_r_ref = 4, p_i_ref = 12, wrong)
Test 65482: a_r = -7, a_i = -4, b_r = -1, b_i = -1, p_r = 3, p_i = 8, p_r_ref = 3, p_i_ref = 11, wrong)
Test 65483: a_r = -6, a_i = -4, b_r = -1, b_i = -1, p_r = 2, p_i = 7, p_r_ref = 2, p_i_ref = 10, wrong)
Test 65484: a_r = -5, a_i = -4, b_r = -1, b_i = -1, p_r = 1, p_i = 6, p_r_ref = 1, p_i_ref = 9, wrong)
Test 65485: a_r = -4, a_i = -4, b_r = -1, b_i = -1, p_r = 0, p_i = 5, p_r_ref = 0, p_i_ref = 8, wrong)
Test 65486: a_r = -3, a_i = -4, b_r = -1, b_i = -1, p_r = -1, p_i = 4, p_r_ref = -1, p_i_ref = 7, wrong)
Test 65487: a_r = -2, a_i = -4, b_r = -1, b_i = -1, p_r = -2, p_i = 3, p_r_ref = -2, p_i_ref = 6, wrong)
Test 65488: a_r = -1, a_i = -4, b_r = -1, b_i = -1, p_r = -3, p_i = 2, p_r_ref = -3, p_i_ref = 5, wrong)
Test 65489: a_r = 0, a_i = -3, b_r = -1, b_i = -1, p_r = -3, p_i = 1, p_r_ref = -3, p_i_ref = 3, wrong)
Test 65490: a_r = 1, a_i = -3, b_r = -1, b_i = -1, p_r = -4, p_i = 0, p_r_ref = -4, p_i_ref = 2, wrong)
Test 65491: a_r = 2, a_i = -3, b_r = -1, b_i = -1, p_r = -5, p_i = -1, p_r_ref = -5, p_i_ref = 1, wrong)
Test 65492: a_r = 3, a_i = -3, b_r = -1, b_i = -1, p_r = -6, p_i = -2, p_r_ref = -6, p_i_ref = 0, wrong)
Test 65493: a_r = 4, a_i = -3, b_r = -1, b_i = -1, p_r = -7, p_i = -3, p_r_ref = -7, p_i_ref = -1, wrong)
Test 65494: a_r = 5, a_i = -3, b_r = -1, b_i = -1, p_r = -8, p_i = -4, p_r_ref = -8, p_i_ref = -2, wrong)

Test 65495: a_r = 6, a_i = -3, b_r = -1, b_i = -1, p_r = -9, p_i = -5, p_r_ref = -9, p_i_ref = -3, wrong)
Test 65496: a_r = 7, a_i = -3, b_r = -1, b_i = -1, p_r = -10, p_i = -6, p_r_ref = -10, p_i_ref = -4, wrong)
Test 65497: a_r = -8, a_i = -3, b_r = -1, b_i = -1, p_r = 5, p_i = 9, p_r_ref = 5, p_i_ref = 11, wrong)
Test 65498: a_r = -7, a_i = -3, b_r = -1, b_i = -1, p_r = 4, p_i = 8, p_r_ref = 4, p_i_ref = 10, wrong)
Test 65499: a_r = -6, a_i = -3, b_r = -1, b_i = -1, p_r = 3, p_i = 7, p_r_ref = 3, p_i_ref = 9, wrong)
Test 65500: a_r = -5, a_i = -3, b_r = -1, b_i = -1, p_r = 2, p_i = 6, p_r_ref = 2, p_i_ref = 8, wrong)
Test 65501: a_r = -4, a_i = -3, b_r = -1, b_i = -1, p_r = 1, p_i = 5, p_r_ref = 1, p_i_ref = 7, wrong)
Test 65502: a_r = -3, a_i = -3, b_r = -1, b_i = -1, p_r = 0, p_i = 4, p_r_ref = 0, p_i_ref = 6, wrong)
Test 65503: a_r = -2, a_i = -3, b_r = -1, b_i = -1, p_r = -1, p_i = 3, p_r_ref = -1, p_i_ref = 5, wrong)
Test 65504: a_r = -1, a_i = -3, b_r = -1, b_i = -1, p_r = -2, p_i = 2, p_r_ref = -2, p_i_ref = 4, wrong)
Test 65505: a_r = 0, a_i = -2, b_r = -1, b_i = -1, p_r = -2, p_i = 1, p_r_ref = -2, p_i_ref = 2, wrong)
Test 65506: a_r = 1, a_i = -2, b_r = -1, b_i = -1, p_r = -3, p_i = 0, p_r_ref = -3, p_i_ref = 1, wrong)
Test 65507: a_r = 2, a_i = -2, b_r = -1, b_i = -1, p_r = -4, p_i = -1, p_r_ref = -4, p_i_ref = 0, wrong)
Test 65508: a_r = 3, a_i = -2, b_r = -1, b_i = -1, p_r = -5, p_i = -2, p_r_ref = -5, p_i_ref = -1, wrong)
Test 65509: a_r = 4, a_i = -2, b_r = -1, b_i = -1, p_r = -6, p_i = -3, p_r_ref = -6, p_i_ref = -2, wrong)
Test 65510: a_r = 5, a_i = -2, b_r = -1, b_i = -1, p_r = -7, p_i = -4, p_r_ref = -7, p_i_ref = -3, wrong)
Test 65511: a_r = 6, a_i = -2, b_r = -1, b_i = -1, p_r = -8, p_i = -5, p_r_ref = -8, p_i_ref = -4, wrong)
Test 65512: a_r = 7, a_i = -2, b_r = -1, b_i = -1, p_r = -9, p_i = -6, p_r_ref = -9, p_i_ref = -5, wrong)
Test 65513: a_r = -8, a_i = -2, b_r = -1, b_i = -1, p_r = 6, p_i = 9, p_r_ref = 6, p_i_ref = 10, wrong)
Test 65514: a_r = -7, a_i = -2, b_r = -1, b_i = -1, p_r = 5, p_i = 8, p_r_ref = 5, p_i_ref = 9, wrong)
Test 65515: a_r = -6, a_i = -2, b_r = -1, b_i = -1, p_r = 4, p_i = 7, p_r_ref = 4, p_i_ref = 8, wrong)
Test 65516: a_r = -5, a_i = -2, b_r = -1, b_i = -1, p_r = 3, p_i = 6, p_r_ref = 3, p_i_ref = 7, wrong)
Test 65517: a_r = -4, a_i = -2, b_r = -1, b_i = -1, p_r = 2, p_i = 5, p_r_ref = 2, p_i_ref = 6, wrong)
Test 65518: a_r = -3, a_i = -2, b_r = -1, b_i = -1, p_r = 1, p_i = 4, p_r_ref = 1, p_i_ref = 5, wrong)
Test 65519: a_r = -2, a_i = -2, b_r = -1, b_i = -1, p_r = 0, p_i = 3, p_r_ref = 0, p_i_ref = 4, wrong)
Test 65520: a_r = -1, a_i = -2, b_r = -1, b_i = -1, p_r = -1, p_i = 2, p_r_ref = -1, p_i_ref = 3, wrong)
Test 65521: a_r = 0, a_i = -1, b_r = -1, b_i = -1, p_r = -1, p_i = 1, p_r_ref = -1, p_i_ref = 1, correct)
Test 65522: a_r = 1, a_i = -1, b_r = -1, b_i = -1, p_r = -2, p_i = 0, p_r_ref = -2, p_i_ref = 0, correct)
Test 65523: a_r = 2, a_i = -1, b_r = -1, b_i = -1, p_r = -3, p_i = -1, p_r_ref = -3, p_i_ref = -1, correct)
Test 65524: a_r = 3, a_i = -1, b_r = -1, b_i = -1, p_r = -4, p_i = -2, p_r_ref = -4, p_i_ref = -2, correct)
Test 65525: a_r = 4, a_i = -1, b_r = -1, b_i = -1, p_r = -5, p_i = -3, p_r_ref = -5, p_i_ref = -3, correct)
Test 65526: a_r = 5, a_i = -1, b_r = -1, b_i = -1, p_r = -6, p_i = -4, p_r_ref = -6, p_i_ref = -4, correct)
Test 65527: a_r = 6, a_i = -1, b_r = -1, b_i = -1, p_r = -7, p_i = -5, p_r_ref = -7, p_i_ref = -5, correct)
Test 65528: a_r = 7, a_i = -1, b_r = -1, b_i = -1, p_r = -8, p_i = -6, p_r_ref = -8, p_i_ref = -6, correct)
Test 65529: a_r = -8, a_i = -1, b_r = -1, b_i = -1, p_r = 7, p_i = 9, p_r_ref = 7, p_i_ref = 9, correct)
Test 65530: a_r = -7, a_i = -1, b_r = -1, b_i = -1, p_r = 6, p_i = 8, p_r_ref = 6, p_i_ref = 8, correct)
Test 65531: a_r = -6, a_i = -1, b_r = -1, b_i = -1, p_r = 5, p_i = 7, p_r_ref = 5, p_i_ref = 7, correct)
Test 65532: a_r = -5, a_i = -1, b_r = -1, b_i = -1, p_r = 4, p_i = 6, p_r_ref = 4, p_i_ref = 6, correct)
Test 65533: a_r = -4, a_i = -1, b_r = -1, b_i = -1, p_r = 3, p_i = 5, p_r_ref = 3, p_i_ref = 5, correct)
Test 65534: a_r = -3, a_i = -1, b_r = -1, b_i = -1, p_r = 2, p_i = 4, p_r_ref = 2, p_i_ref = 4, correct)

Test 65535: a_r = -2, a_i = -1, b_r = -1, b_i = -1, p_r = 1, p_i = 3, p_r_ref = 1, p_i_ref = 3, correct)

```
// prob6_tbench.v
`timescale 10ps/1ps
module prob6_tbench;
    reg         clk;
    reg         reset;
    reg [3:0]  a_r, a_i;
    reg [3:0]  b_r, b_i;
    wire [8:0] p_r, p_i;
    reg [3:0] a_r_old, a_i_old, b_r_old, b_i_old;
    reg [8:0] p_r_ref, p_i_ref;
    reg [15:0] i;
    prob6
_prob6(.clk(clk), .a_r(a_r), .a_i(a_i), .b_r(b_r), .b_i(b_i), .p_r(p_r), .p_i(p_
i));
    initial begin
        reset = 1'b1;
        #500;
        reset = 1'b0;
        @(posedge clk); #10
        for (i = 0; i < 16'b1111111111111111; i = i + 1) begin
            a_r = i[3:0];
            a_i = i[7:4];
            b_r = i[11:8];
            b_i = i[15:12];
            @(posedge clk); #10
            if(i!=0)begin
                if (p_r == p_r_ref && p_i == p_i_ref)
                    $display("Test %0d: a_r = %0d, a_i = %0d, b_r = %0d, b_i
= %0d, p_r = %0d, p_i = %0d, p_r_ref = %0d, p_i_ref = %0d, correct)",
                            i, $signed(a_r_old), $signed(a_i_old),
$signed(b_r_old), $signed(b_i_old), $signed(p_r), $signed(p_i),
$signed(p_r_ref), $signed(p_i_ref));
                else
                    $display("Test %0d: a_r = %0d, a_i = %0d, b_r = %0d, b_i
= %0d, p_r = %0d, p_i = %0d, p_r_ref = %0d, p_i_ref = %0d, wrong)",
                            i, $signed(a_r_old), $signed(a_i_old),
$signed(b_r_old), $signed(b_i_old), $signed(p_r), $signed(p_i),
$signed(p_r_ref), $signed(p_i_ref));
            end
            a_r_old = a_r;
            a_i_old = a_i;
            b_r_old = b_r;
            b_i_old = b_i;
            p_r_ref = {{5{a_r[3]}}, a_r} * {{5{b_r[3]}}, b_r} - {{5{a_i[3]}},
a_i} * {{5{b_i[3]}}, b_i};
            p_i_ref = {{5{a_r[3]}}, a_r} * {{5{b_i[3]}}, b_i} + {{5{a_i[3]}},
a_i} * {{5{b_r[3]}}, b_r};
        end
```

```
        @(posedge clk); #10
        if (p_r == p_r_ref && p_i == p_i_ref)
            $display("Test %0d: a_r = %0d, a_i = %0d, b_r = %0d, b_i = %0d, p_r
= %0d, p_i = %0d, p_r_ref = %0d, p_i_ref = %0d, correct)",
                    i, $signed(a_r_old), $signed(a_i_old), $signed(b_r_old),
$signed(b_i_old), $signed(p_r), $signed(p_i), $signed(p_r_ref),
$signed(p_i_ref));
        else
            $display("Test %0d: a_r = %0d, a_i = %0d, b_r = %0d, b_i = %0d, p_r
= %0d, p_i = %0d, p_r_ref = %0d, p_i_ref = %0d, wrong)",
                    i, $signed(a_r_old), $signed(a_i_old), $signed(b_r_old),
$signed(b_i_old), $signed(p_r), $signed(p_i), $signed(p_r_ref),
$signed(p_i_ref));
        repeat (50) @ (posedge clk);
        $finish;
    end

    always begin
        if(reset == 1'b1)begin
            clk = 1'b0;
            #1;
        end
        else begin
            #100
            clk = ~clk;
        end
    end

endmodule
```