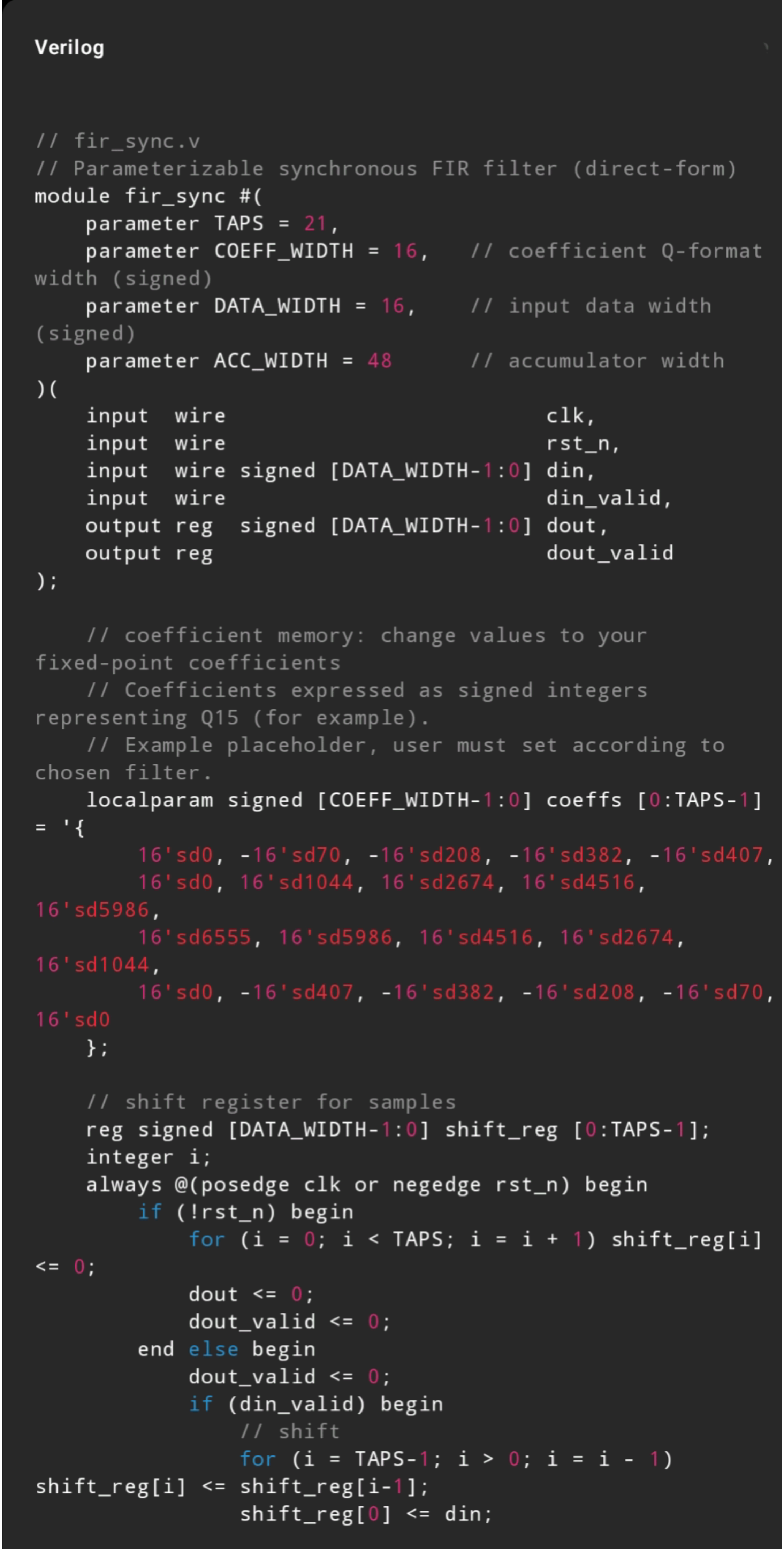
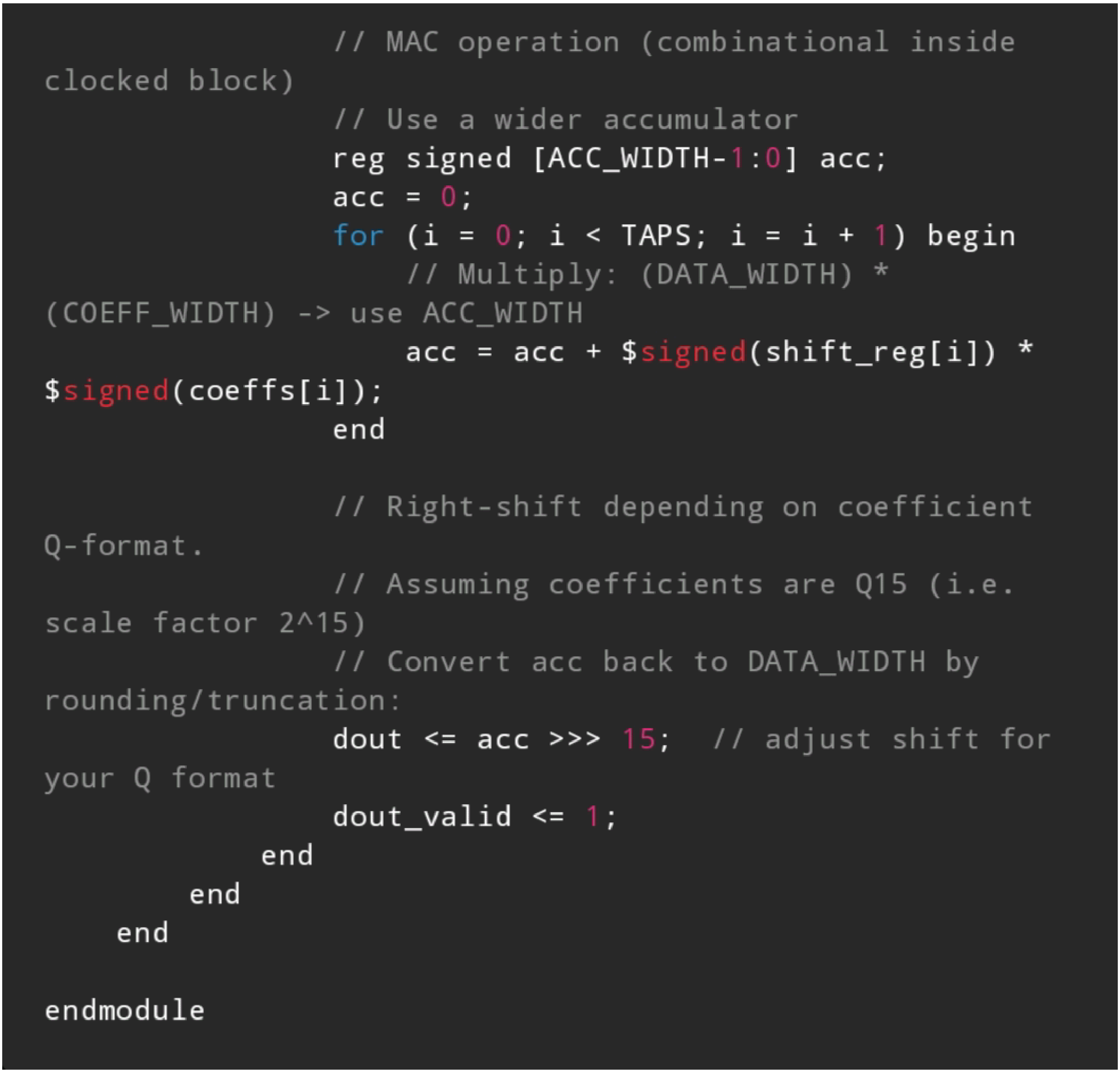
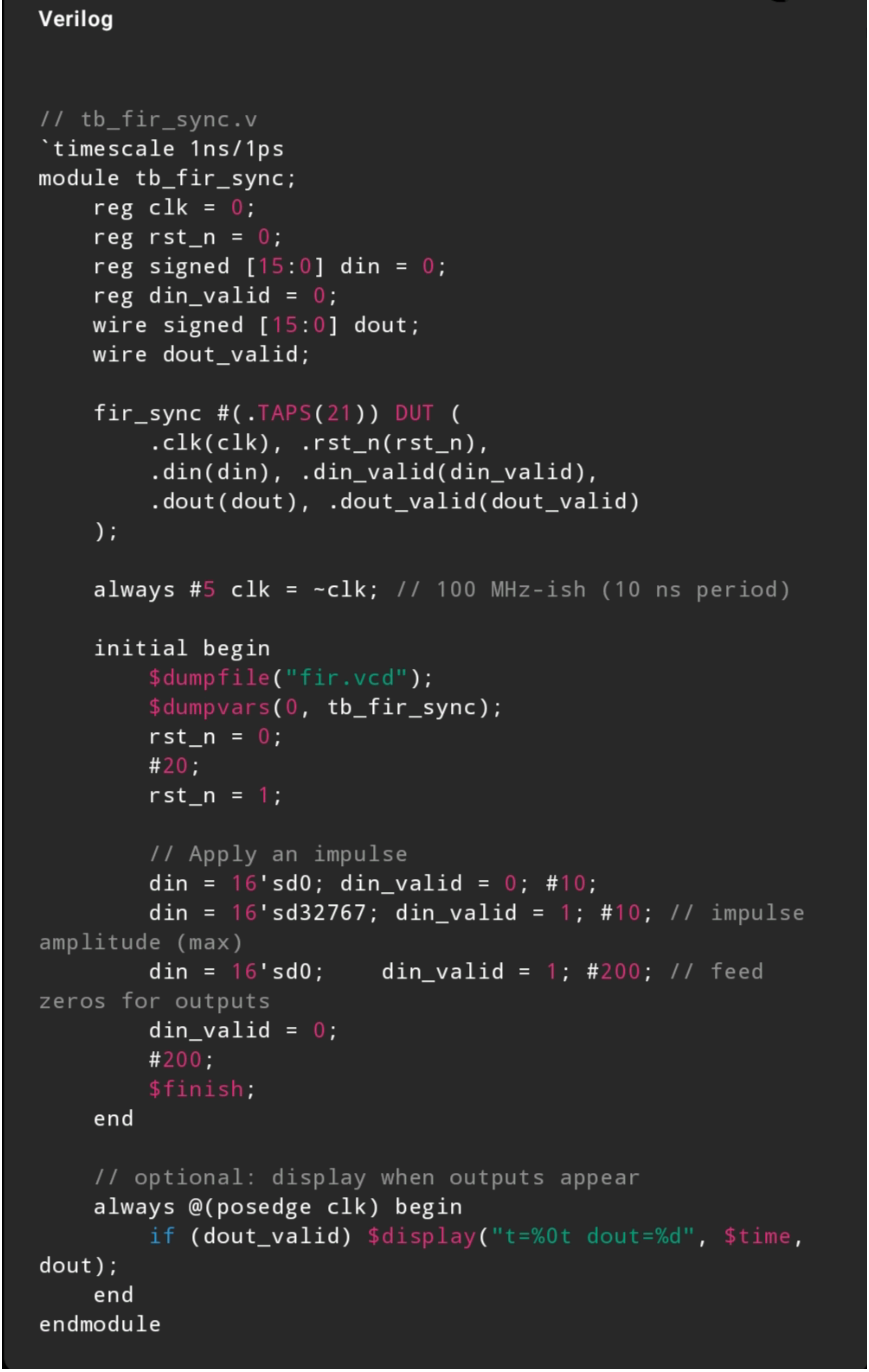


Verilog

```verilog
// fir_sync.v
// Parameterizable synchronous FIR filter (direct-form)
module fir_sync #(
    parameter TAPS = 21,
    parameter COEFF_WIDTH = 16,   // coefficient Q-format width (signed)
    parameter DATA_WIDTH = 16,    // input data width (signed)
    parameter ACC_WIDTH = 48      // accumulator width
)(
    input  wire                         clk,
    input  wire                         rst_n,
    input  wire signed [DATA_WIDTH-1:0] din,
    input  wire                         din_valid,
    output reg  signed [DATA_WIDTH-1:0] dout,
    output reg                          dout_valid
);

    // coefficient memory: change values to your fixed-point coefficients
    // Coefficients expressed as signed integers representing Q15 (for example).
    // Example placeholder, user must set according to chosen filter.
    localparam signed [COEFF_WIDTH-1:0] coeffs [0:TAPS-1] = '{
        16'sd0, -16'sd70, -16'sd208, -16'sd382, -16'sd407,
        16'sd0, 16'sd1044, 16'sd2674, 16'sd4516, 16'sd5986,
        16'sd6555, 16'sd5986, 16'sd4516, 16'sd2674, 16'sd1044,
        16'sd0, -16'sd407, -16'sd382, -16'sd208, -16'sd70, 16'sd0
    };

    // shift register for samples
    reg signed [DATA_WIDTH-1:0] shift_reg [0:TAPS-1];
    integer i;
    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin
            for (i = 0; i < TAPS; i = i + 1) shift_reg[i] <= 0;
            dout <= 0;
            dout_valid <= 0;
        end else begin
            dout_valid <= 0;
            if (din_valid) begin
                // shift
                for (i = TAPS-1; i > 0; i = i - 1) shift_reg[i] <= shift_reg[i-1];
                shift_reg[0] <= din;
```



```
                // MAC operation (combinational inside clocked block)
                // Use a wider accumulator
                reg signed [ACC_WIDTH-1:0] acc;
                acc = 0;
                for (i = 0; i < TAPS; i = i + 1) begin
                    // Multiply: (DATA_WIDTH) * (COEFF_WIDTH) -> use ACC_WIDTH
                    acc = acc + $signed(shift_reg[i]) * $signed(coeffs[i]);
                end

                // Right-shift depending on coefficient Q-format.
                // Assuming coefficients are Q15 (i.e. scale factor 2^15)
                // Convert acc back to DATA_WIDTH by rounding/truncation:
                dout <= acc >>> 15;  // adjust shift for your Q format
                dout_valid <= 1;
            end
        end
    end

endmodule
```

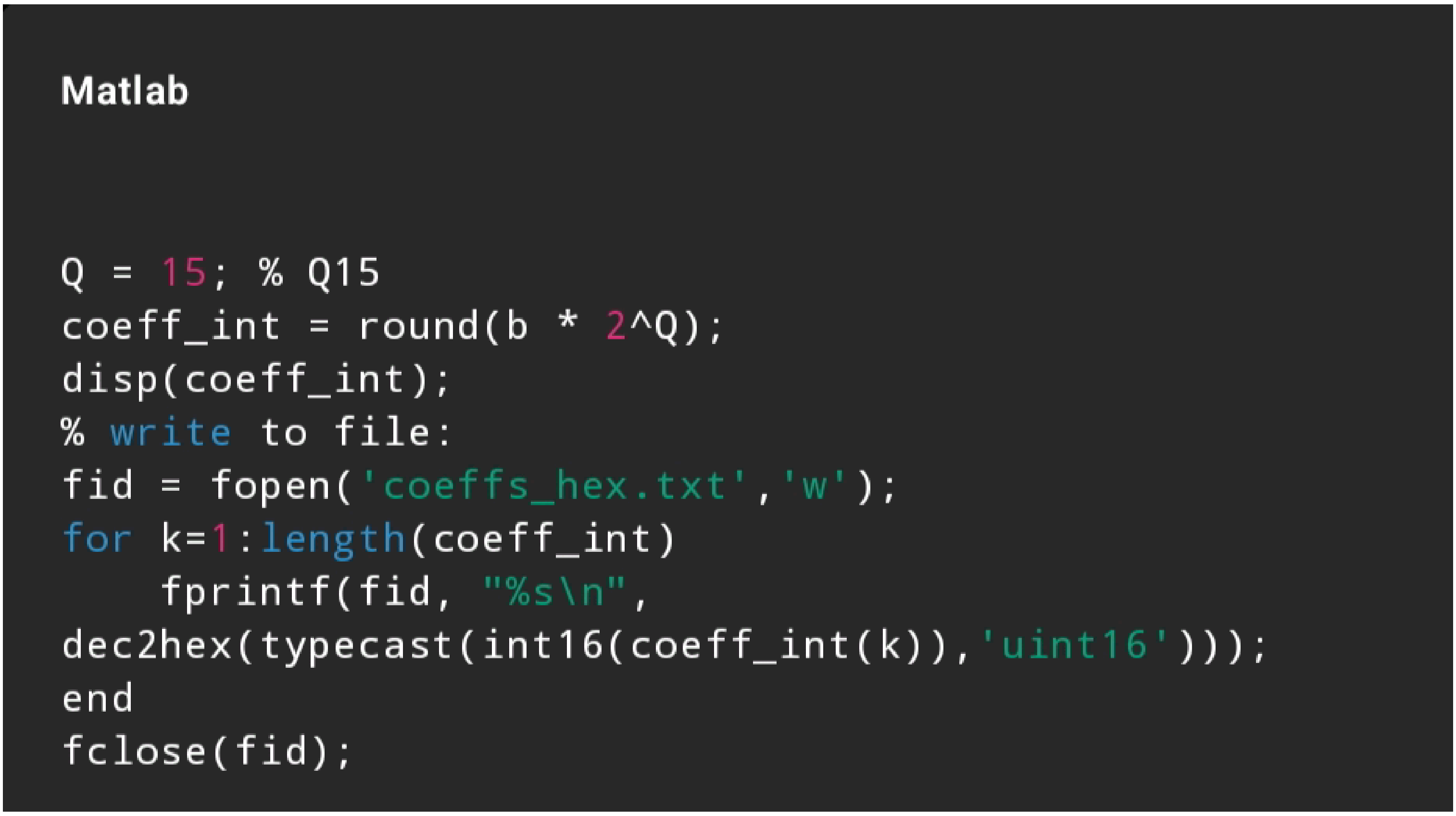


Verilog

```
// tb_fir_sync.v
`timescale 1ns/1ps
module tb_fir_sync;
    reg clk = 0;
    reg rst_n = 0;
    reg signed [15:0] din = 0;
    reg din_valid = 0;
    wire signed [15:0] dout;
    wire dout_valid;

    fir_sync #(.TAPS(21)) DUT (
        .clk(clk), .rst_n(rst_n),
        .din(din), .din_valid(din_valid),
        .dout(dout), .dout_valid(dout_valid)
    );

    always #5 clk = ~clk; // 100 MHz-ish (10 ns period)

    initial begin
        $dumpfile("fir.vcd");
        $dumpvars(0, tb_fir_sync);
        rst_n = 0;
        #20;
        rst_n = 1;

        // Apply an impulse
        din = 16'sd0; din_valid = 0; #10;
        din = 16'sd32767; din_valid = 1; #10; // impulse amplitude (max)
        din = 16'sd0;    din_valid = 1; #200; // feed zeros for outputs
        din_valid = 0;
        #200;
        $finish;
    end

    // optional: display when outputs appear
    always @(posedge clk) begin
        if (dout_valid) $display("t=%0t dout=%d", $time, dout);
    end
endmodule
```

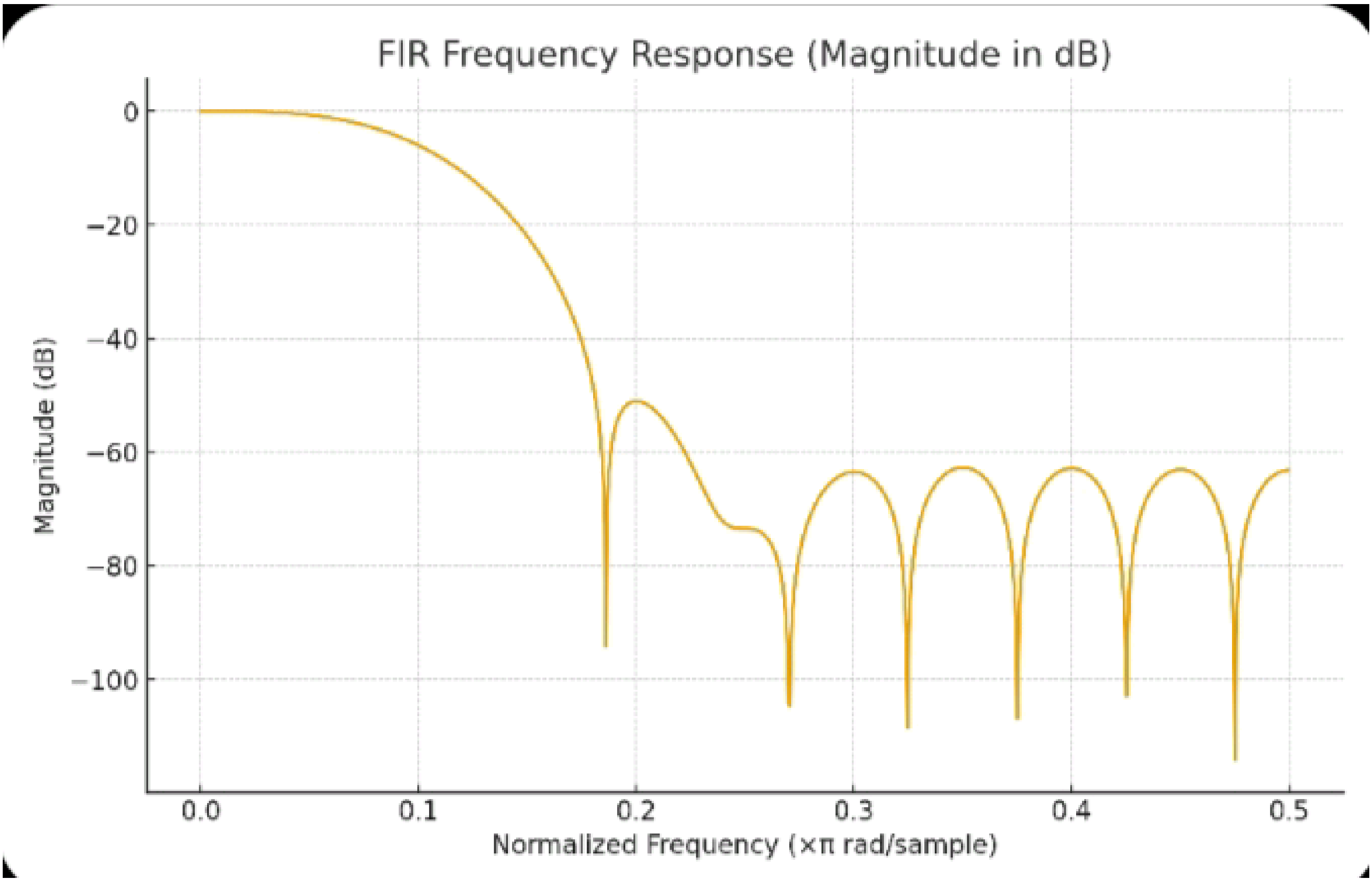
Matlab

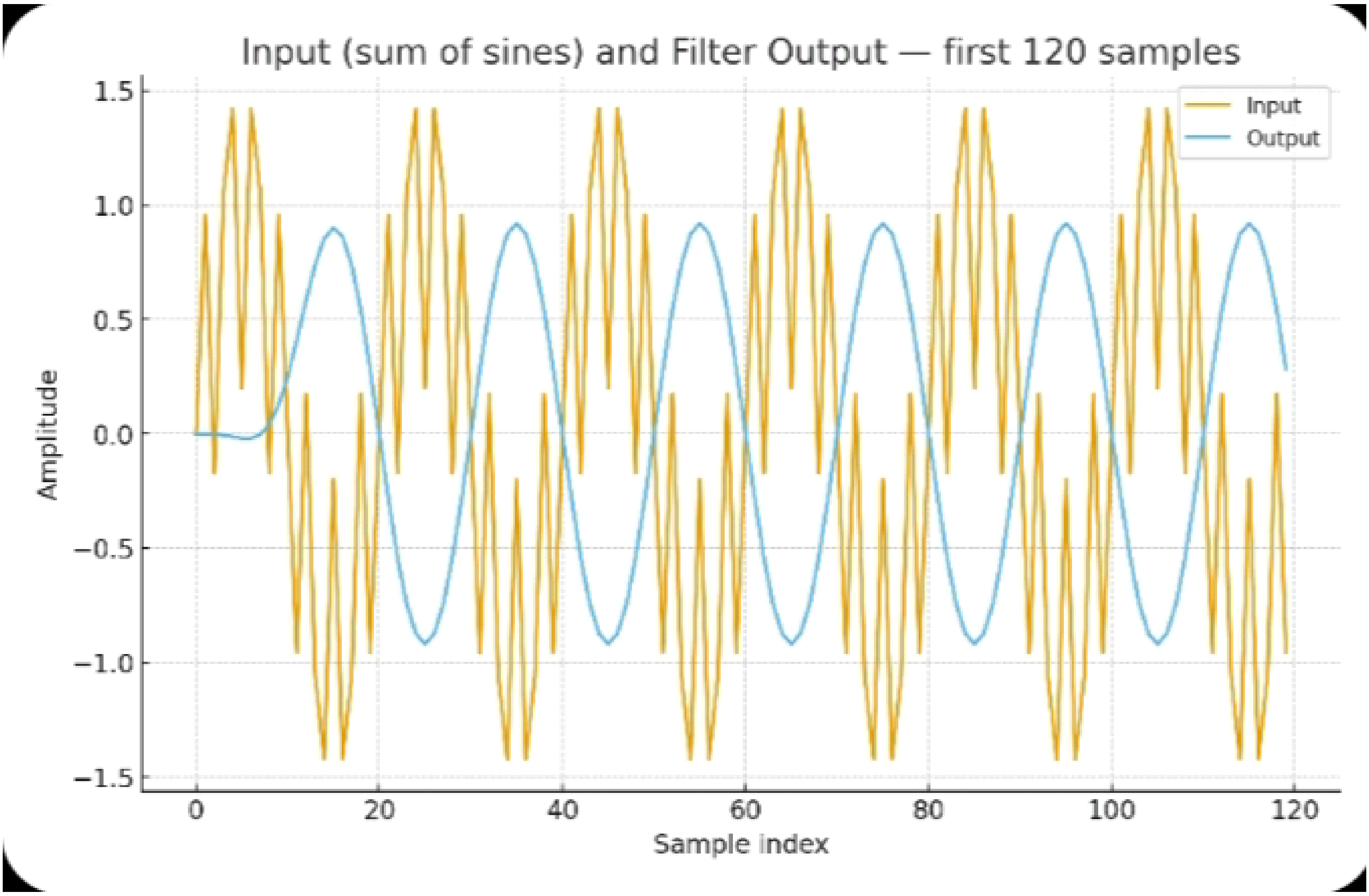
```matlab
% fir_design_and_simulate.m
clear; close all; clc;

% Filter spec
numTaps = 21;
cutoff = 0.2; % normalized cutoff (0.0 - 0.5)
b = fir1(numTaps-1, 2*cutoff, hamming(numTaps)); % fir1 expects normalized 0-1 (1 = Nyquist)

% Plot frequency response
[H,w] = freqz(b,1,1024);
figure; plot(w/pi/2, 20*log10(abs(H))); grid on
title('FIR Frequency Response (Magnitude in dB)')
xlabel('Normalized Frequency (×π rad/sample)')
ylabel('Magnitude (dB)')

% Impulse response
figure; stem(0:length(b)-1, b, 'filled');
title('Impulse Response (coefficients)')
xlabel('n'); ylabel('h[n]'); grid on

% Test signal: sum of two sines
n = 0:199;
x = sin(2*pi*0.05*n) + 0.8*sin(2*pi*0.35*n);
y = filter(b,1,x);

% Plot input vs output (first 120 samples)
figure; plot(n(1:120), x(1:120)); hold on; plot(n(1:120), y(1:120));
legend('Input','Output'); grid on;
title('Input (sum of sines) and Filter Output — first 120 samples')
xlabel('Sample index'); ylabel('Amplitude')
```



Matlab

```
Q = 15; % Q15
coeff_int = round(b * 2^Q);
disp(coeff_int);
% write to file:
fid = fopen('coeffs_hex.txt','w');
for k=1:length(coeff_int)
    fprintf(fid, "%s\n",
dec2hex(typecast(int16(coeff_int(k)),'uint16')));
end
fclose(fid);
```


FIR Frequency Response (Magnitude in dB)
0
−20
−40
−60
−80
−100
Magnitude (dB)
0.0
0.1
0.2
0.3
0.4
0.5
Normalized Frequency (×π rad/sample)


Input (sum of sines) and Filter Output — first 120 samples
Input
Output
Amplitude
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
20
40
60
80
100
120
Sample index