ELEC 391 Demo 2

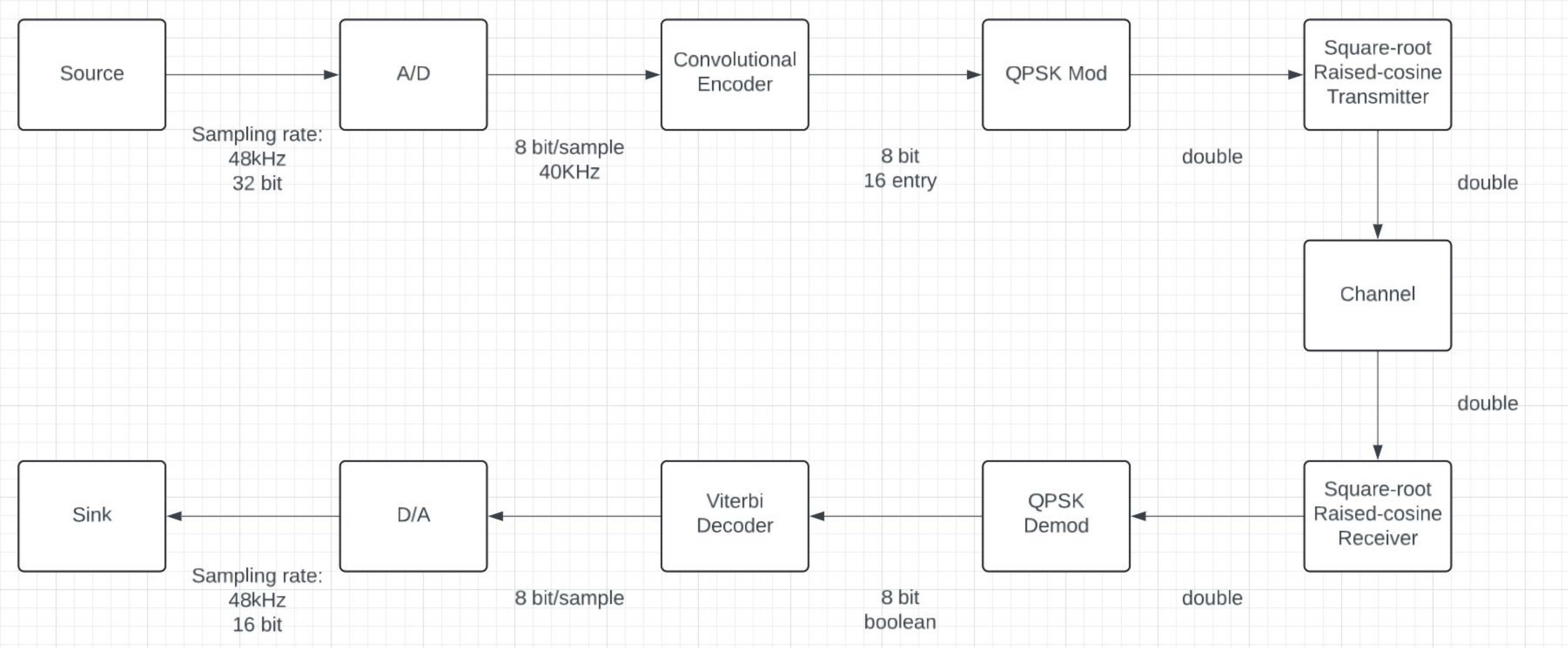
Team 6
Yuqian Song, Chloe Sun, Pengyu Ji

Performance Table

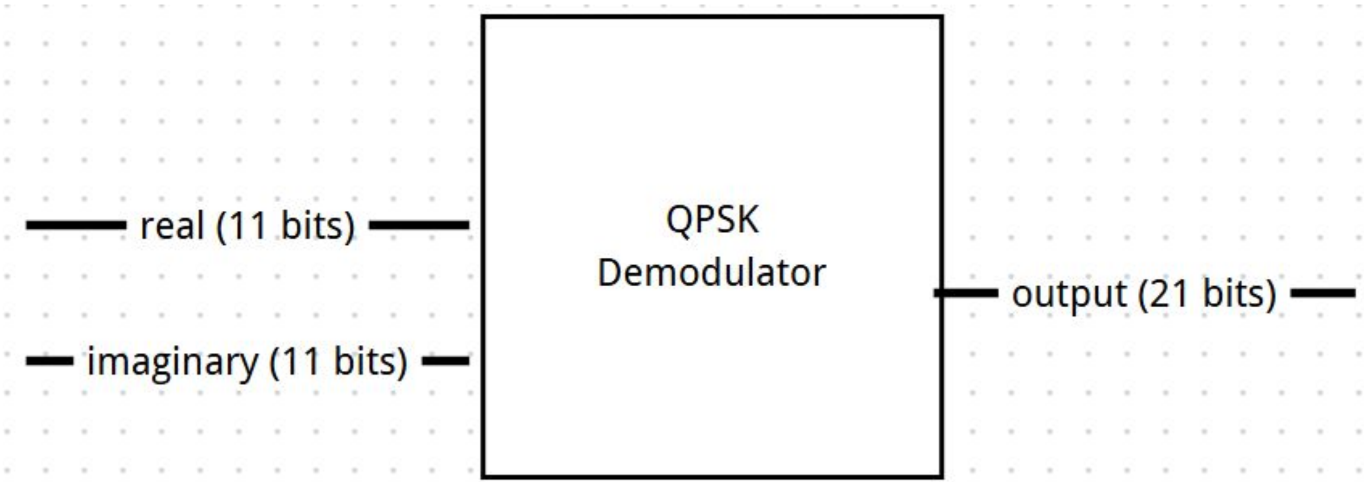
Audio BW (kHz)	Target BER	Spectral Mask (kHz)	Target Channel	Delay (ms)
20	10e-3	200	ß	25

Criterion	Simulink Perf.	FPGA Perf.
Message Transmission (bit rate)	320 kbps	640 kbps
Transmission Reliability (bit error probability)	3.1e-6	
Processing Delay	1 ms	29 ns
Channel Bandwidth	210 kHz	480 kHz

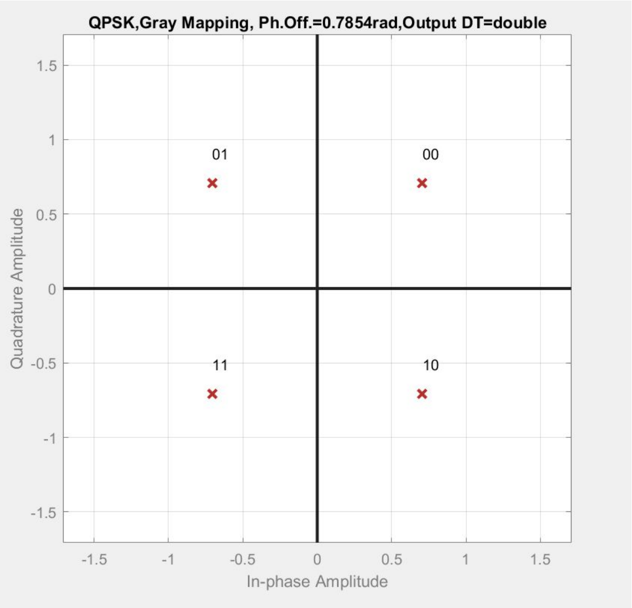
System Block Diagram

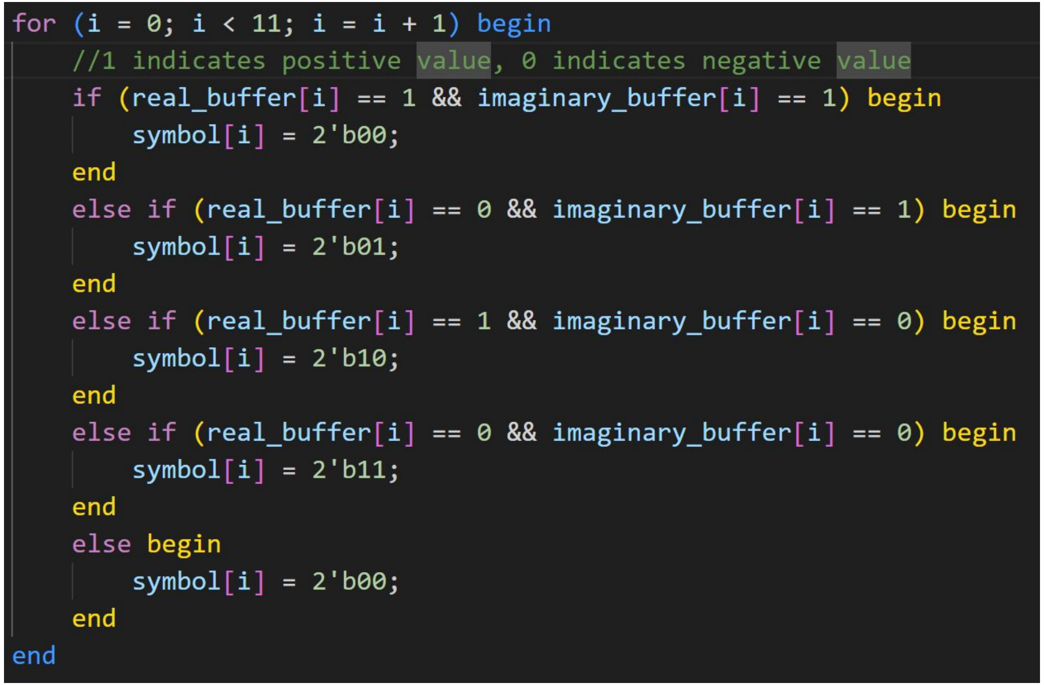


Validation NO.1: QPSK Demodulator



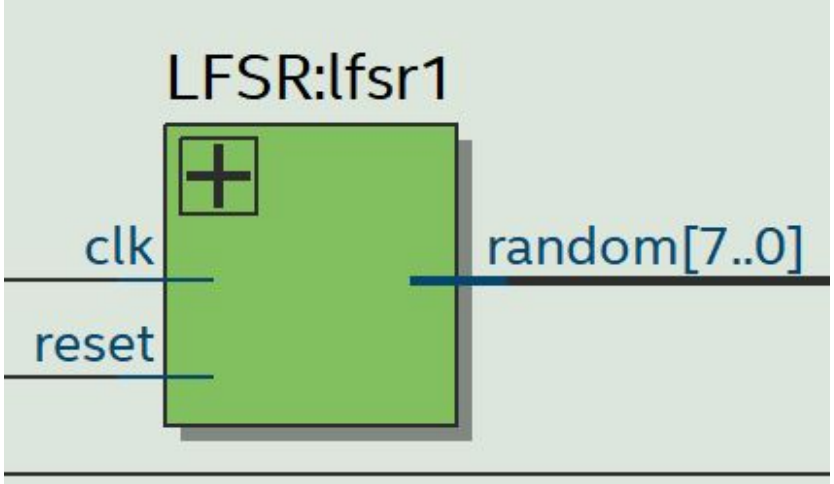
QPSK Demodulator

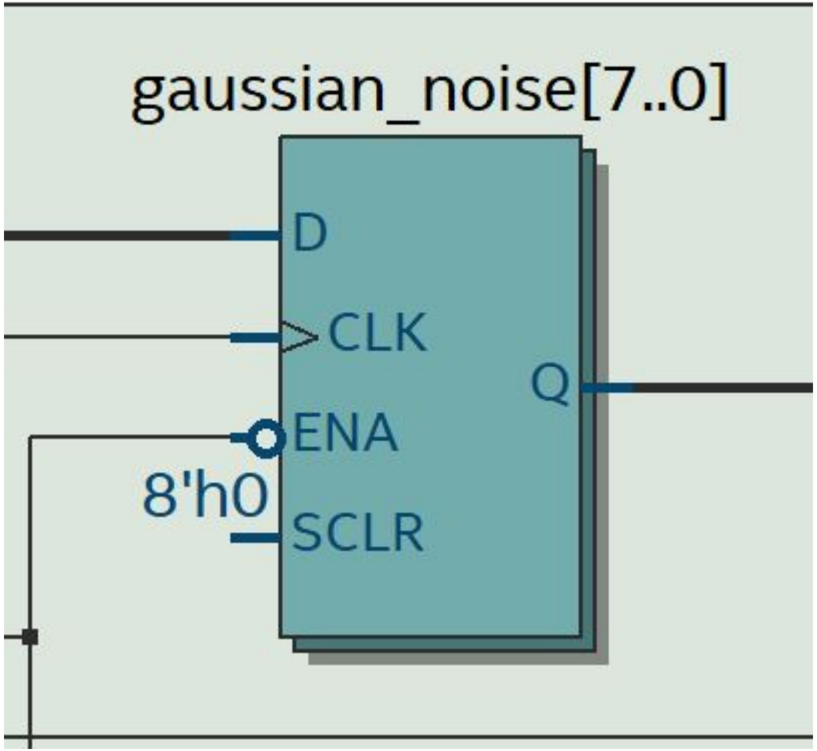




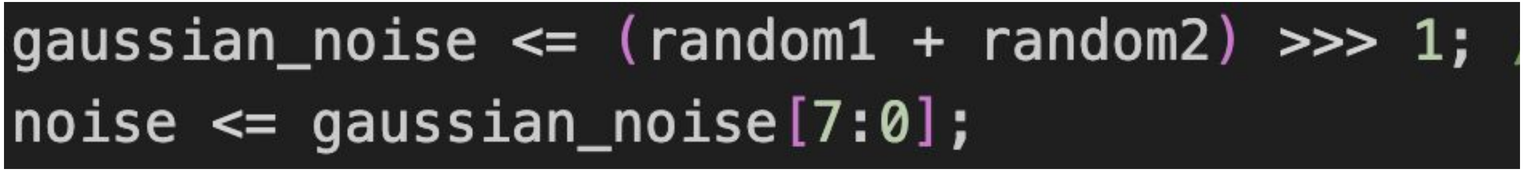
```
for (i = 0; i < 11; i = i + 1) begin
    //1 indicates positive value, 0 indicates negative value
    if (real_buffer[i] == 1 && imaginary_buffer[i] == 1) begin
        symbol[i] = 2'b00;
    end
    else if (real_buffer[i] == 0 && imaginary_buffer[i] == 1) begin
        symbol[i] = 2'b01;
    end
    else if (real_buffer[i] == 1 && imaginary_buffer[i] == 0) begin
        symbol[i] = 2'b10;
    end
    else if (real_buffer[i] == 0 && imaginary_buffer[i] == 0) begin
        symbol[i] = 2'b11;
    end
    else begin
        symbol[i] = 2'b00;
    end
end
```

Validation No.2 AWGN Channel





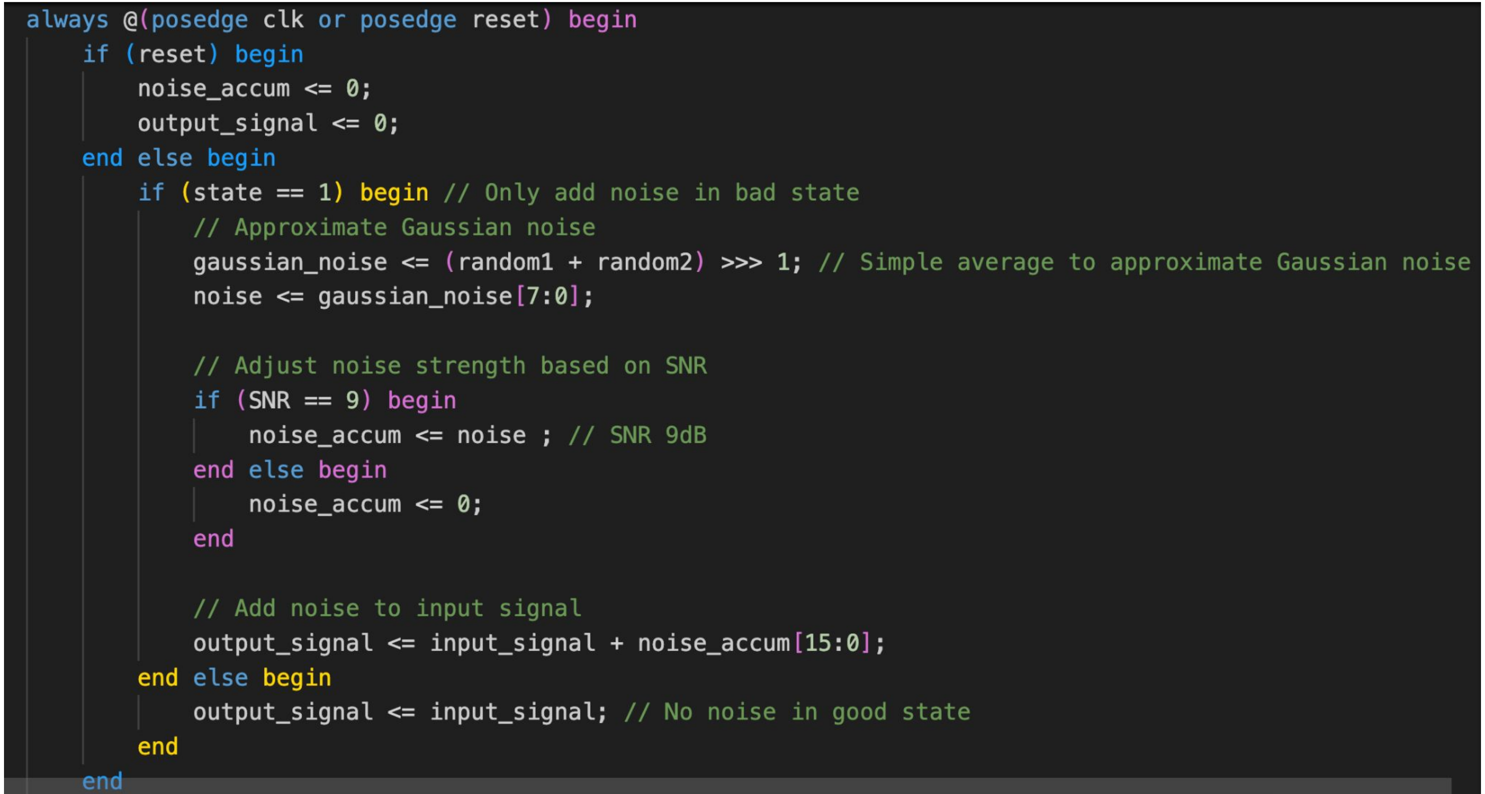
Gaussian noise generator



```
gaussian_noise <= (random1 + random2) >>> 1;
noise <= gaussian_noise[7:0];
```

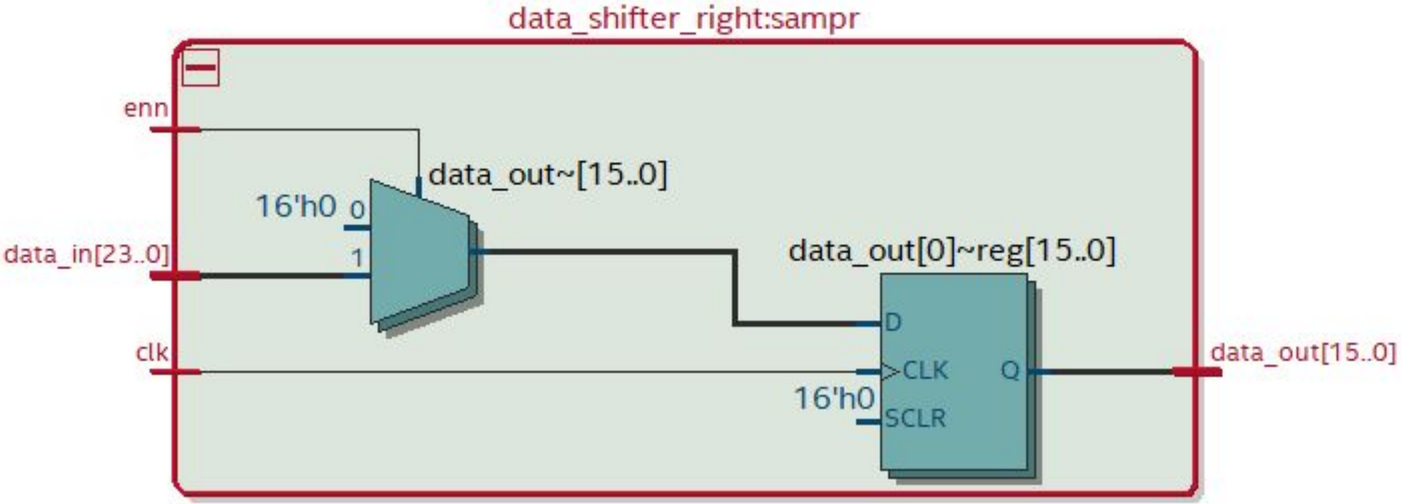
CLT: The Central Limit Theorem is a fundamental statistical principle that states that the distribution of the sum (or average) of a large number of independent, identically distributed random variables approaches a normal (Gaussian) distribution.

Gaussian noise generator

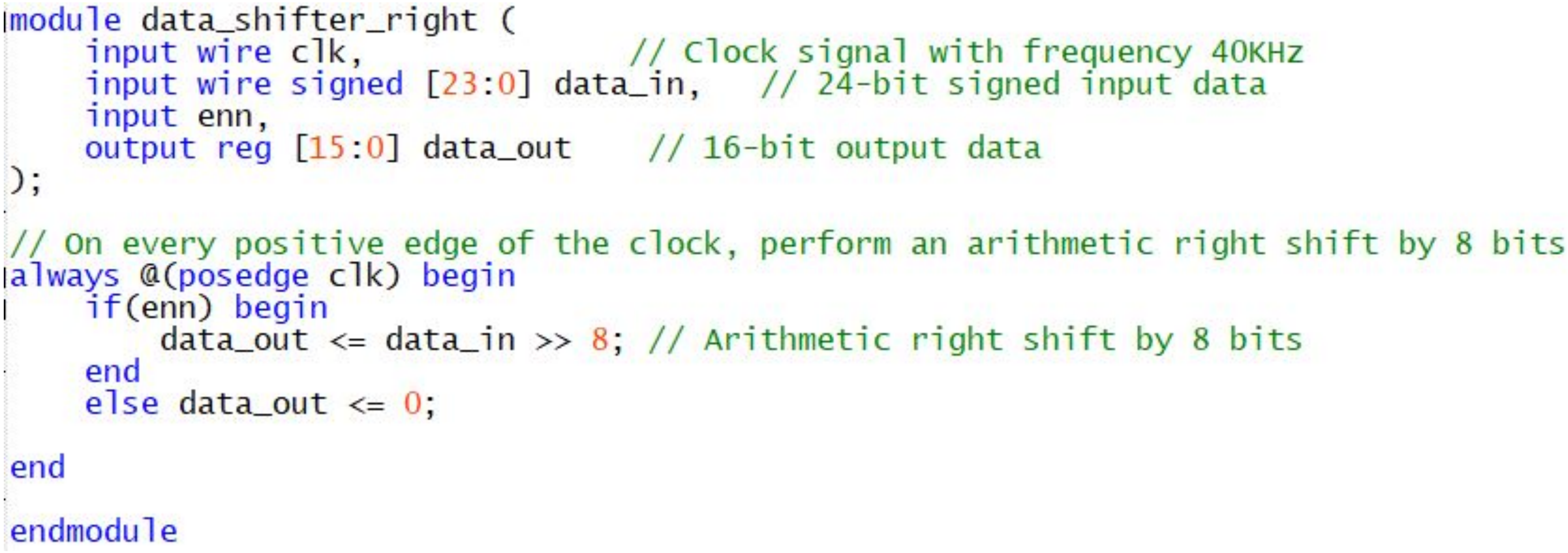


```
always @(posedge clk or posedge reset) begin
    if (reset) begin
        noise_accum <= 0;
        output_signal <= 0;
    end else begin
        if (state == 1) begin // Only add noise in bad state
            // Approximate Gaussian noise
            gaussian_noise <= (random1 + random2) >>> 1; // Simple average to approximate Gaussian noise
            noise <= gaussian_noise[7:0];

            // Adjust noise strength based on SNR
            if (SNR == 9) begin
                noise_accum <= noise ; // SNR 9dB
            end else begin
                noise_accum <= 0;
            end

            // Add noise to input signal
            output_signal <= input_signal + noise_accum[15:0];
        end else begin
            output_signal <= input_signal; // No noise in good state
        end
    end
```

Validation No.3 Downsampling and Quantization

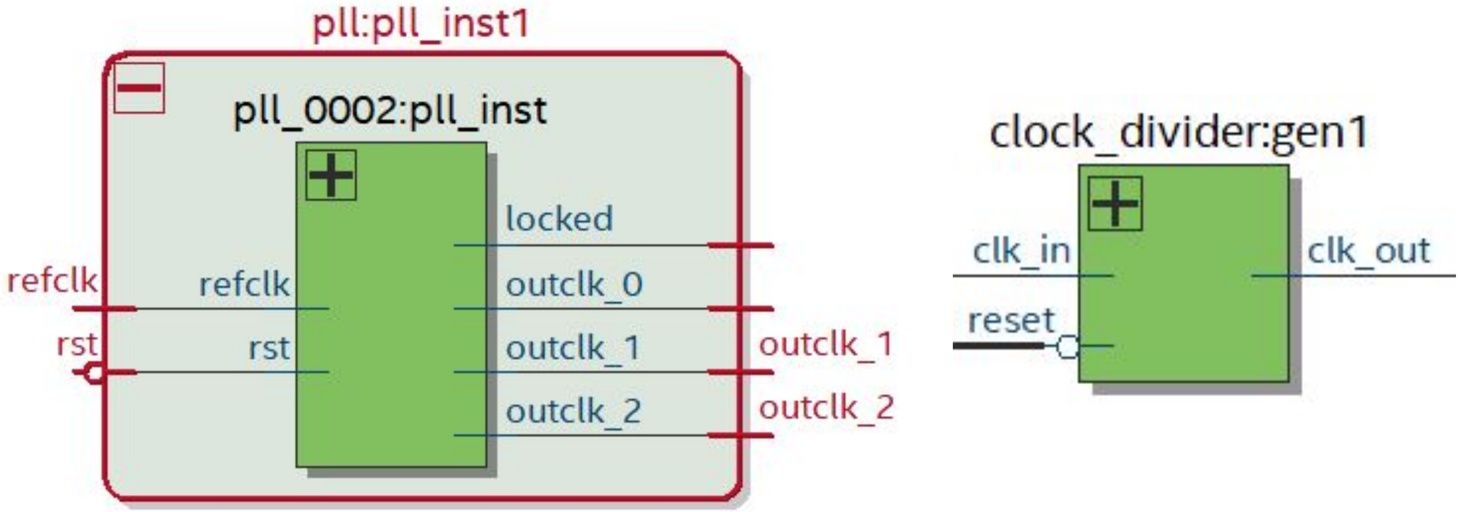


Quantization Verilog Implementation

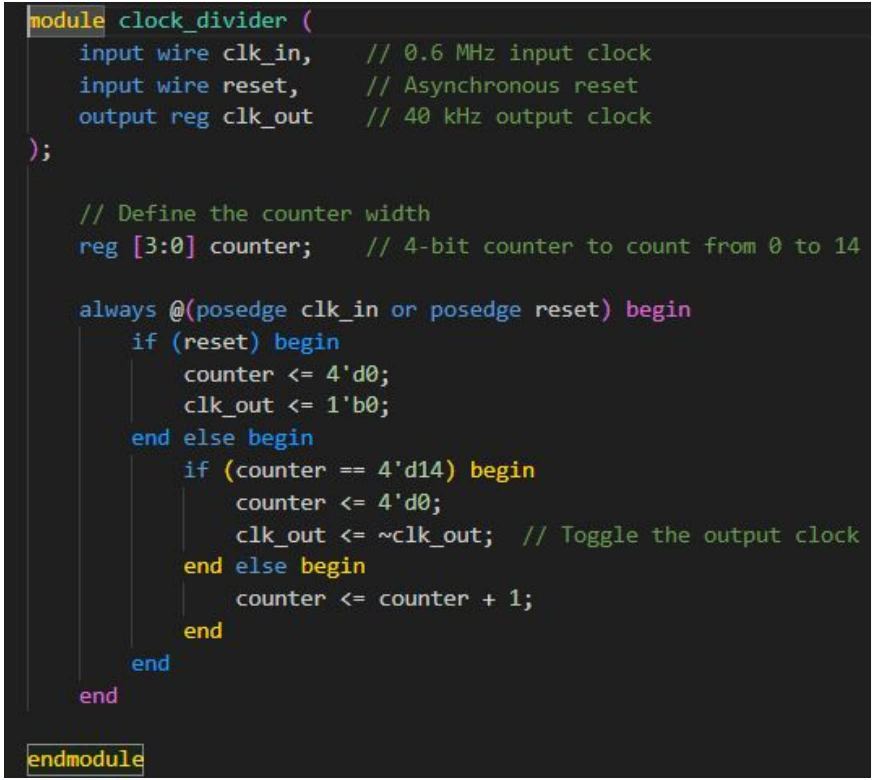


```
module data_shifter_right (
    input wire clk,              // Clock signal with frequency 40KHz
    input wire signed [23:0] data_in,   // 24-bit signed input data
    input enn,
    output reg [15:0] data_out    // 16-bit output data
);

// On every positive edge of the clock, perform an arithmetic right shift by 8 bits
always @(posedge clk) begin
    if(enn) begin
        data_out <= data_in >> 8; // Arithmetic right shift by 8 bits
    end
    else data_out <= 0;

end

endmodule
```

Downsampling



Downsampling Verilog Implementation



```verilog
module clock_divider (
    input wire clk_in,    // 0.6 MHz input clock
    input wire reset,     // Asynchronous reset
    output reg clk_out    // 40 kHz output clock
);

    // Define the counter width
    reg [3:0] counter;    // 4-bit counter to count from 0 to 14

    always @(posedge clk_in or posedge reset) begin
        if (reset) begin
            counter <= 4'd0;
            clk_out <= 1'b0;
        end else begin
            if (counter == 4'd14) begin
                counter <= 4'd0;
                clk_out <= ~clk_out;  // Toggle the output clock
            end else begin
                counter <= counter + 1;
            end
        end
    end

endmodule
```

Demonstration

Q&A