Task 4 - FIR Filter Design (Verilog)

Objective: Design and simulate a digital FIR (Finite Impulse Response) filter using Verilog.

Verilog Code (fir_filter.v)

```
module fir_filter #(
    parameter DATA_WIDTH = 8,
    parameter COEF_WIDTH = 8,
    parameter TAP_NUM    = 3
)
(
    input  clk,
    input  reset,
    input  signed [DATA_WIDTH-1:0] x,
    output reg signed [DATA_WIDTH+COEF_WIDTH-1:0] y
);
reg signed [DATA_WIDTH-1:0] shift_reg [0:TAP_NUM-1];
wire signed [DATA_WIDTH+COEF_WIDTH-1:0] mult [0:TAP_NUM-1];
reg signed [DATA_WIDTH+COEF_WIDTH-1:0] acc;
integer i;
localparam signed [COEF_WIDTH-1:0] coef0 = 1;
localparam signed [COEF_WIDTH-1:0] coef1 = 2;
localparam signed [COEF_WIDTH-1:0] coef2 = 1;
assign mult[0] = shift_reg[0] * coef0;
assign mult[1] = shift_reg[1] * coef1;
assign mult[2] = shift_reg[2] * coef2;
always @(posedge clk or posedge reset) begin
    if (reset) begin
         for (i = 0; i < TAP_NUM; i = i + 1)
             shift_reg[i] <= 0;
         y <= 0;
    end else begin
         shift_reg[2] <= shift_reg[1];
         shift_reg[1] <= shift_reg[0];
         shift_reg[0] <= x;
         acc = 0;
         for (i = 0; i < TAP_NUM; i = i + 1)
             acc = acc + mult[i];
         y <= acc;
    end
end
endmodule
```

Testbench Code (fir_filter_tb.v)

```
module fir_filter_tb;
  reg clk;
  reg reset;
  reg signed [7:0] x;
  wire signed [15:0] y;
  fir_filter uut (
      .clk(clk),
      .reset(reset),
      .x(x),
      .y(y)
  );
  initial begin
      clk = 0;
      forever #5 clk = ~clk;
  end
  initial begin
      reset = 1;
      x = 0;
      #10;
      reset = 0;
      x = 8'd1;  #10;
      x = 8'd2;  #10;
      x = 8'd3;  #10;
      x = 8'd4;  #10;
      x = 8'd5;  #10;
      $finish;
  end
  initial begin
      $monitor("Time %t: x = %d, y = %d", $time, x, y);
  end
endmodule
```