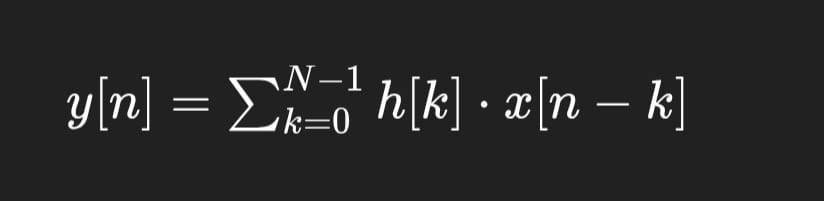
1. **FIR Filter Design Overview**

A Finite Impulse Response (FIR) filter has an output defined by a weighted sum of current and past input samples:



**Where:**

* h[k]h[k]h[k] are the filter coefficients (impulse response),
* x[n−k]x[n-k]x[n−k] are the input samples,
* NNN is the number of taps.

1. **Verilog Code (FIR Filter)**

module fir\_filter (

input clk,

input rst,

input signed [7:0] x\_in,

output reg signed [15:0] y\_out

);

reg signed [7:0] shift\_reg [0:3];

wire signed [15:0] mul [0:3];

wire signed [15:0] sum;

**// Coefficients scaled by 2^8 = 256 for fixed-point (0.25 \* 256 = 64)**

parameter signed [7:0] h [0:3] = {64, 64, 64, 64};

**// Shift Register**

always @(posedge clk or posedge rst) begin

if (rst) begin

shift\_reg[0] <= 0;

shift\_reg[1] <= 0;

shift\_reg[2] <= 0;

shift\_reg[3] <= 0;

end else begin

shift\_reg[3] <= shift\_reg[2];

shift\_reg[2] <= shift\_reg[1];

shift\_reg[1] <= shift\_reg[0];

shift\_reg[0] <= x\_in;

end

end

assign mul[0] = shift\_reg[0] \* h[0];

assign mul[1] = shift\_reg[1] \* h[1];

assign mul[2] = shift\_reg[2] \* h[2];

assign mul[3] = shift\_reg[3] \* h[3];

assign sum = mul[0] + mul[1] + mul[2] + mul[3];

**// Output with bit shift to adjust scaling**

always @(posedge clk or posedge rst) begin

if (rst)

y\_out <= 0;

else

y\_out <= sum >>> 8**; // scale down result**

end

endmodule

1. **Testbench (Verilog)**

module tb\_fir\_filter;

reg clk, rst;

reg signed [7:0] x\_in;

wire signed [15:0] y\_out;

fir\_filter uut (

.clk(clk),

.rst(rst),

.x\_in(x\_in),

.y\_out(y\_out)

);

always #5 clk = ~clk**; // Clock generation**

initial begin

$monitor("Time = %0d, Input = %d, Output = %d", $time, x\_in, y\_out);

clk = 0;

rst = 1;

x\_in = 0;

#10 rst = 0;

**// Apply test inputs**

x\_in = 10; #10;

x\_in = 20; #10;

x\_in = 30; #10;

x\_in = 40; #10;

x\_in = 50; #10;

x\_in = 60; #10;

x\_in = 70; #10;

x\_in = 80; #10;

#50 $finish;

end

endmodule

**4. Simulation Results**

* Look for waveform outputs (e.g., x\_in vs y\_out) to verify filtering effect
* You should see the output representing a smoothed version of the input signal

**5. Performance Analysis**

Key performance metrics:

* **Latency**: 4 clock cycles (for 4-tap filter)
* **Throughput**: 1 sample per clock (after pipeline filled)
* **Resource Usage**:
  + 4 multipliers (can be optimized using MAC units)
  + 4 adders
  + Shift register for past inputs