**Verilog Design source:**

**/\***

**FIR lowpass filter with a cutoff frequency of -16MHz at 188MHz sampling rate**

**\*/**

**module HPFIR\_7 (**

**input clk,**

**input signed [15:0] noisy\_signal,**

**output signed [15:0] filtered\_signal // Filtered output signal. 1.1.14**

**);**

**integer i,j;**

**// Coefficients for 9-tap FIR**

**// 10MHz cutoff frequency at 185MHz sampling rate**

**/\*reg signed [15:0] coeff [0:6] = { 16'h 0000,**

**16'h FDD5,**

**16'h F956,**

**16'h 7702,**

**16'h F956,**

**16'h FDD5,**

**16'h 0000};\*/**

**reg signed [15:0] coeff [0:6] = { 16'h 0000,**

**16'h 04D8,**

**16'h E747,**

**16'h 2666,**

**16'h E747,**

**16'h 04D8,**

**16'h 0000};**

**reg signed [15:0] delayed\_signal [0:6];**

**reg signed [31:0] prod [0:6];**

**reg signed [32:0] sum\_0 [0:3];**

**reg signed [33:0] sum\_1 [0:1];**

**reg signed [34:0] sum\_2;**

**always @(posedge clk)**

**begin**

**delayed\_signal[0] <= noisy\_signal;**

**for (i=1;i<=6; i=i+1) begin**

**delayed\_signal[i] <= delayed\_signal[i-1];**

**end**

**end**

**// Pipelined multiply and accumulate**

**always @(posedge clk)**

**begin**

**for (j=0; j<=6; j=j+1) begin**

**prod[j] <= delayed\_signal[j] \* coeff[j];**

**end**

**end**

**always @(posedge clk)**

**begin**

**sum\_0[0] <= (prod[0]+ prod[1]);**

**sum\_0[1] <= (prod[2]+ prod[3]);**

**sum\_0[2] <= (prod[4]+ prod[5]);**

**sum\_0[3] <= prod[6];**

**end**

**always @(posedge clk)**

**begin**

**sum\_1[0] <= (sum\_0[0] + sum\_0[1]);**

**sum\_1[1] <= (sum\_0[2] + sum\_0[3]);**

**end**

**always @(posedge clk)**

**begin**

**sum\_2 <= (sum\_1[0] + sum\_1[1]);**

**end**

**// Filtered output signal**

**assign filtered\_signal = $signed (sum\_2[34:13]);**

**endmodule**

**Verilog Test Bench Source:**

**`timescale 1 us / 100 ns**

**module HPFIR\_7\_tb ();**

**localparam CORDIC\_CLK\_PERIOD = 2;**

**// To create 1 GHz CORDIC sampling clock**

**localparam FIR\_CLK\_PERIOD = 1000;**

**// To create 1 kHz FIR Lowpass filter sampling clock**

**localparam signed [15:0] PI\_POS = 16'h 6488;**

**localparam signed [15:0] PI\_NEG = 16'h 9B78;**

**// +pi in fixed-point 1.2.13**

**//-pi in fixed-point 1.2.13**

**localparam PHASE\_INC\_250HZ = 26;**

**localparam PHASE\_INC\_400HZ = 42;**

**// Phase jump for 250Hz sine wave synthesis // Phase jump for 400Hz sine wave synthesis**

**reg cordic\_clk = 1'b0;**

**reg fir\_clk = 1'b0;**

**reg phase\_tvalid = 1'b0;**

**reg signed [15:0] phase\_250HZ=0;**

**// 250Hz phase sweep, 1.2.13**

**reg signed [15:0] phase\_400HZ = 0;**

**// 400Hz phase sweep. 1.2.13**

**wire sincos\_250HZ\_tvalid;**

**wire signed [15:0] sin\_250HZ, cos\_250HZ;**

**// 1.1.14 250Hz sine/cosine**

**wire sincos\_400HZ\_tvalid;**

**wire signed [15:0] sin\_400HZ, cos\_400HZ;**

**// 1.1.14 400Hz sine/cosine**

**reg signed [15:0] noisy\_signal = 0;**

**wire signed [15:0] filtered\_signal;**

**// Resampled 250Hz sine + 400Hz sine. 1.1.14 // Filtered signal output from FIR Lowpass filter**

**// Synthesize 28Hz sine**

**cordic\_0 cordic\_inst\_0 (**

**.aclk (cordic\_clk),**

**.s\_axis\_phase\_tvalid (phase\_tvalid),**

**.s\_axis\_phase\_tdata (phase\_250HZ),**

**.m\_axis\_dout\_tvalid (sincos\_250HZ\_tvalid),**

**.m\_axis\_dout\_tdata ({sin\_250HZ})**

**);**

**// Synthesize 30m2 sine**

**cordic\_0 cordic\_inst\_1 (**

**.aclk (cordic\_clk),**

**.s\_axis\_phase\_tvalid (phase\_tvalid),**

**.s\_axis\_phase\_tdata (phase\_400HZ),**

**.m\_axis\_dout\_tvalid (sincos\_400HZ\_tvalid),**

**.m\_axis\_dout\_tdata ({sin\_400HZ})**

**);**

**// Phase swe??**

**always @(posedge cordic\_clk)**

**begin**

**phase\_tvalid <= 1'b1;**

**// Sweep phase to synthesize 250Hz sine**

**if (phase\_250HZ + PHASE\_INC\_250HZ < PI\_POS) begin**

**phase\_250HZ <= phase\_250HZ + PHASE\_INC\_250HZ;**

**end else begin**

**phase\_250HZ <= PI\_NEG+ (phase\_250HZ + PHASE\_INC\_250HZ - PI\_POS);**

**end**

**// Sweep phase to synthesize 400Hz sine**

**if (phase\_400HZ + PHASE\_INC\_400HZ <= PI\_POS) begin**

**phase\_400HZ <= phase\_400HZ + PHASE\_INC\_400HZ;**

**end else begin**

**phase\_400HZ <= PI\_NEG + (phase\_400HZ + PHASE\_INC\_400HZ - PI\_POS);**

**end**

**end**

**// Create 1 GHz Cordic clock**

**always begin**

**cordic\_clk = #(CORDIC\_CLK\_PERIOD/2) ~cordic\_clk;**

**end**

**// Create 1 kHz FIR clock**

**always begin**

**fir\_clk = #(FIR\_CLK\_PERIOD/2) ~fir\_clk;**

**end**

**// Noisy signal 250Hz sine + 400Hz sine**

**// Noisy signal is resampled at 1 kHz FIR sampling rate**

**always @(posedge fir\_clk)**

**begin**

**noisy\_signal <= (sin\_250HZ + sin\_400HZ) / 2;**

**end**

**// Feed noisy signal into FIR lowpass filter**

**HPFIR\_7 FIR\_filter\_inst (**

**.clk (fir\_clk),**

**.noisy\_signal (noisy\_signal),**

**.filtered\_signal (filtered\_signal)**

**);**

**//initial begin**

**// // Your simulation setup code here**

**// // Run the simulation for 1000 us (1 ms)**

**// repeat (1000000) begin // 1000 us in 1 us time units**

**// // Provide inputs and/or perform other operations here**

**// #1; // Advance time by 1 time unit (1 us)**

**// end**

**// // Terminate the simulation**

**// $finish;**

**//end**

**Endmodule**