**Thiết kế bộ lọc FIR bằng SystemVerilog**

Bộ lọc FIR được thiết kế fully parameterizable.

Các thông số tùy chỉnh trong thiết kế:

N\_TAPS: số lượng tap của bộ lọc FIR.

COEFF\_W: bit width của thông số bộ lọc (coefficients)

DATA\_W: bit with của dữ liệu (input và output)

COEFFS: mảng chứa các giá trị thông số bộ lọc

Inputs:

clk: tín hiệu clock của mạch, tích cực cạnh lên.

reset\_n: tín hiệu reset, tích cực mức thấp.

data\_in: dữ liệu đầu vào, mặc định 24 bit.

Outputs:

data\_out: dữ liệu ngõ ra, mặc định 24 bit.

Code được push lên git: <https://github.com/Stork1323/DSPonFPGA/blob/main/FIR_filter/FIR.sv>

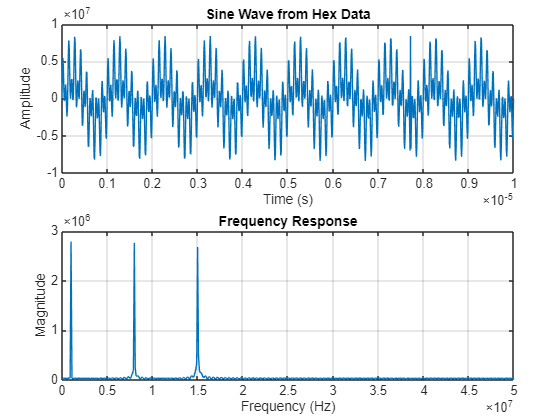
Ở các samples, sử dụng bộ lọc FIR 32 bậc.

**Sample 1:**

Dữ liệu đầu vào là sóng sine bị nhiễu. Đầu tiên ta phải phân tích được phổ tần số của tín hiệu sóng sine, từ đó xác định được loại bộ lọc cần thiết kế. Đoạn code matlab dưới đây sẽ giúp phân tích phổ tần số của sóng sine đầu vào.

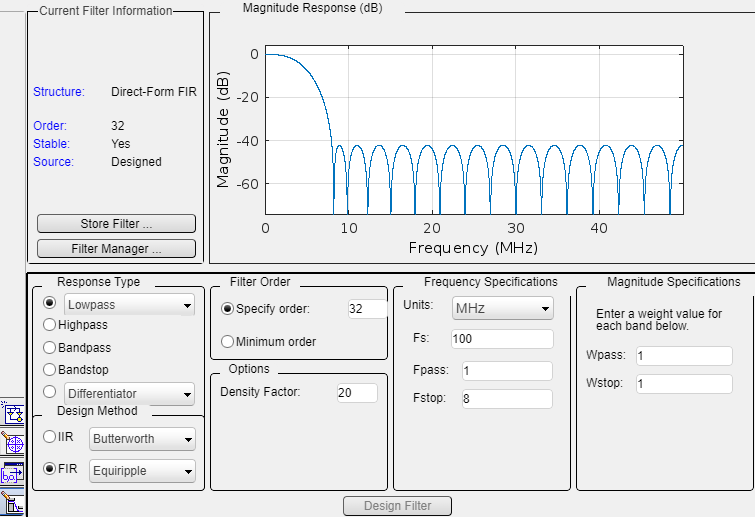
|  |
| --- |
| % Parameters  filename = 'sine.hex'; % Hex file name  sampling\_frequency = 100e6; % 100 MHz  duration = 1; % Duration in seconds  % Read the hex file  fileID = fopen(filename, 'r');  hexData = textscan(fileID, '%s'); % Read hex values as strings  fclose(fileID);  % Convert hex to decimal (24-bit signed integer)  data = hex2dec(hexData{1});  data = data - (data > 2^23) \* 2^24; % Convert to signed integers  % Create a time vector  numSamples = length(data);  timeVector = (0:numSamples-1) / sampling\_frequency;  % Save the waveform data in a variable  waveform = data; % Now 'waveform' holds the sine wave data  % Optionally, plot the data  figure;  subplot(2, 1, 1);  plot(timeVector, waveform);  xlabel('Time (s)');  ylabel('Amplitude');  title('Sine Wave from Hex Data');  grid on;  % Frequency response analysis using FFT  N = length(waveform); % Number of samples  Y = fft(waveform); % Compute the FFT  f = (0:N-1)\*(sampling\_frequency/N); % Frequency vector  % Compute the magnitude  magnitude = abs(Y)/N; % Normalize the magnitude  magnitude = magnitude(1:N/2+1); % Take the positive frequencies  magnitude(2:end-1) = 2\*magnitude(2:end-1); % Adjust magnitude for single-sided spectrum  f = f(1:N/2+1); % Take the positive frequencies  % Plot the frequency response  subplot(2, 1, 2);  plot(f, magnitude);  xlabel('Frequency (Hz)');  ylabel('Magnitude');  title('Frequency Response');  grid on;  % Optionally save the data to a .mat file  save('sine\_wave.mat', 'waveform', 'timeVector', 'f', 'magnitude'); |

Từ đoạn code trên, cho ra được hình ảnh như sau:



Từ kết quả trên, nhận thấy sóng sine là sự kế hợp của 3 tần số là 1MHz, 8MHz, 15MHz. Từ đó, thiết kế bộ lọc đơn giản và hiệu quả nhất thì ta sẽ dùng bộ lọc thông thấp để loại bỏ 2 tần số 8MHz và 15MHz, giữ lại tần số 1MHz.

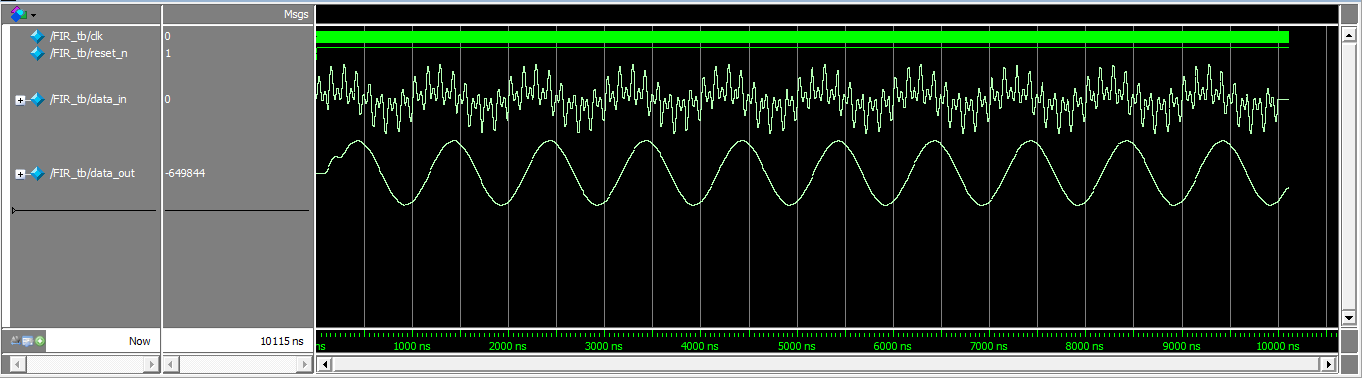
Sử dụng phần mềm Filter Designer trên Matlab, thiết kế bộ lọc như hình:



Ta có được các hệ số của bộ lọc như sau:

-0.006000, -0.004700, -0.005600, -0.005600, -0.004300, -0.001300, 0.003700, 0.010800, 0.020000, 0.030800, 0.042900, 0.055300, 0.067200, 0.077700, 0.085900, 0.091100, 0.092900, 0.091100, 0.085900, 0.077700, 0.067200, 0.055300, 0.042900, 0.030800, 0.020000, 0.010800, 0.003700, -0.001300, -0.004300, -0.005600, -0.005600, -0.004700, -0.006000

Từ các hệ số của bộ lọc, thu được kết quả như hình:

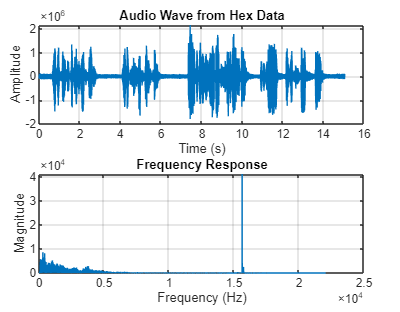


Trong quá trình thiết kế bộ lọc áp dụng cho sóng sine ở trên, nhận thấy khi Fstop nằm ở khoảng 8MHz thì đạt được sóng sine mượt nhất. Nếu Fstop thấp hơn thì bộ lọc có độ suy hao ít nhiều đến tần số 1MHz, trường hợp cao hơn 8MHz thì bộ lọc chưa hoàn toàn lọc được tần số 8MHz và 15MHz nên sóng sine sẽ bị ảnh hưởng ít nhiều.

**Sample 2:**

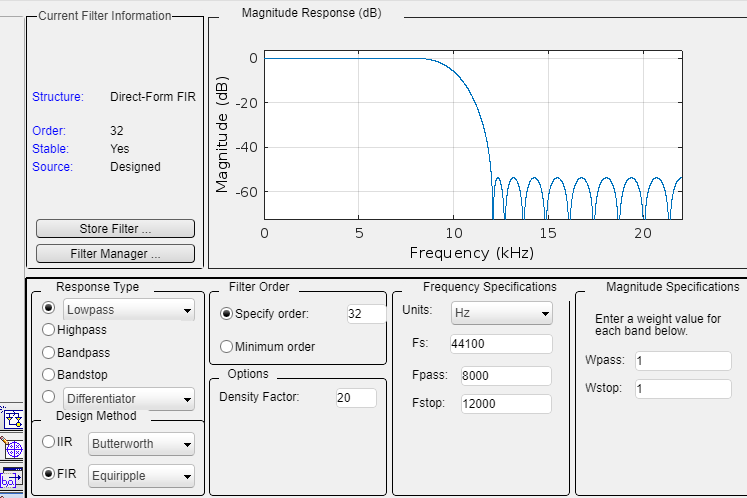
Dữ liệu đầu vào là đoạn audio có tiếng rít, độ dài đoạn audio là 15 giây và tần số lấy mẫu là 44.1KHz.

Phân tích phổ như sample 1, có được:



Nhận thấy phổ tần số của đoạn audio có chứa một hài có biên độ rất lớn ở tần số 15.625Khz. Từ đó, thiết kế bộ lọc thông thấp để loại bỏ tần số này.

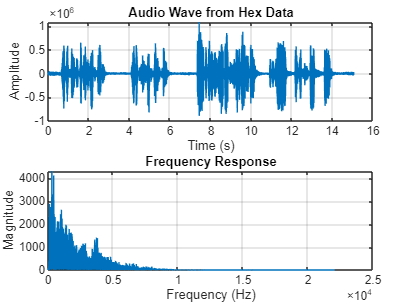
Thông số của bộ lọc được mô tả như hình:



Từ đó, có được các hệ số của bộ lọc như sau:

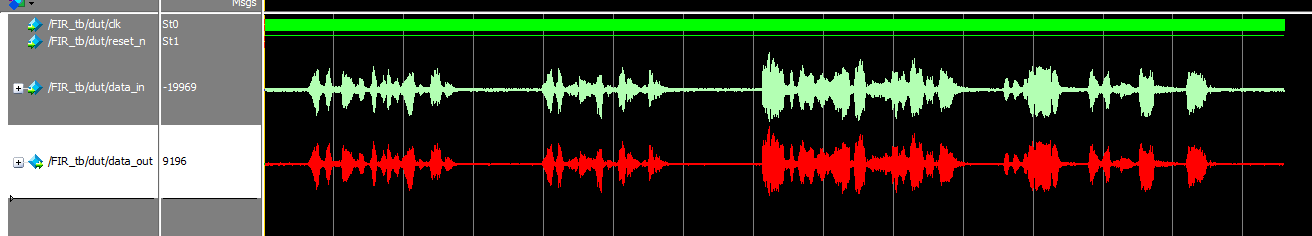
-0.001200, 0.001900, 0.003800, -0.001600, -0.007900, 0.000200, 0.014300, 0.005000, -0.022300, -0.016600, 0.030900, 0.039600, -0.038700, -0.090100, 0.044100, 0.312700, 0.453900, 0.312700, 0.044100, -0.090100, -0.038700, 0.039600, 0.030900, -0.016600, -0.022300, 0.005000, 0.014300, 0.000200, -0.007900, -0.001600, 0.003800, 0.001900, -0.001200

Khi áp dụng bộ lọc, phổ tần số của đoạn audio:



Từ đó ta thấy âm thanh ở tần số trên 15KHz đã bị lọc bỏ.

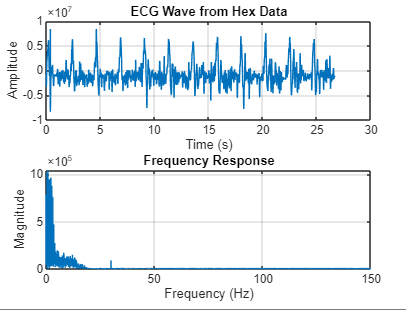
Dưới đây là hình ảnh waveform từ mô phỏng ở phần cứng:



Sample 3:

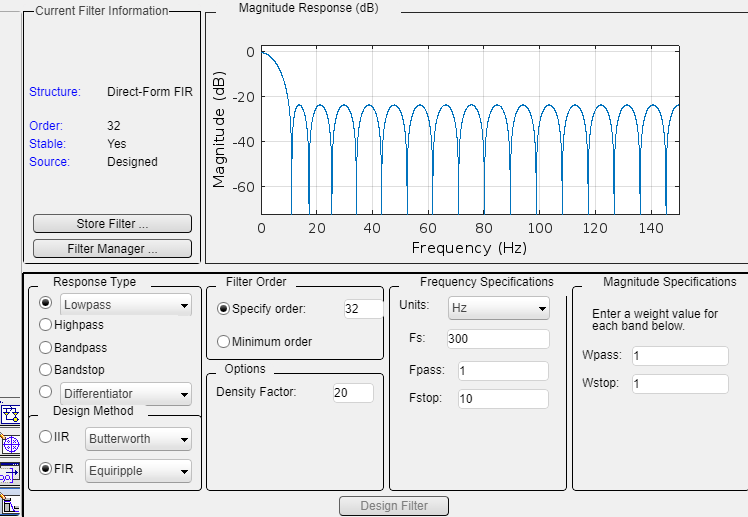
Dữ liệu ngõ vào là tín hiệu ECG chứa nhiễu, thông thường tín hiệu ECG sẽ có khoảng tần số từ 0.5 đến 150 Hz. Thông thường tín hiệu ECG hay bị tác động bởi nhiễu từ các nguồn như điện áp lưới, hoặc nhiễu từ các tín hiệu khi đo đạc điện tim như tín hiệu các cơ khi nhúc nhích,…

Tương tự như 2 ví dụ trên, biểu đồ đáp ứng xung được mô tả như hình:

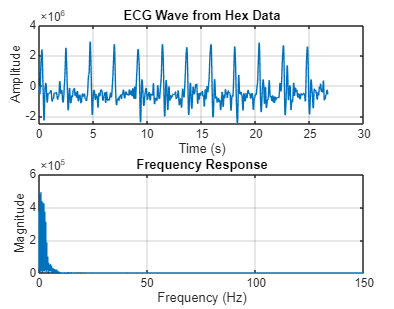


Vì sử dụng tần số lấy mẫu là 300Hz, và từ biểu đồ trên nhận thấy được tín hiệu ECG chủ yếu nằm trong khoảng tần số 0.5 đến 20 Hz. Trong biểu đồ có chứa tín hiệu ở tần số 30Hz. Có thể tín hiệu ở tần số 30Hz này là nhiễu, và có khả năng nhiễu tác động vào khoảng tần 0.5 đến 20 Hz nên việc khả thi nhất là loại bỏ tín hiệu ở tần số 30Hz này.

Sau nhiều lần phân tích và thử nghiệm thì thông số của bộ lọc được thiết kế như sau:

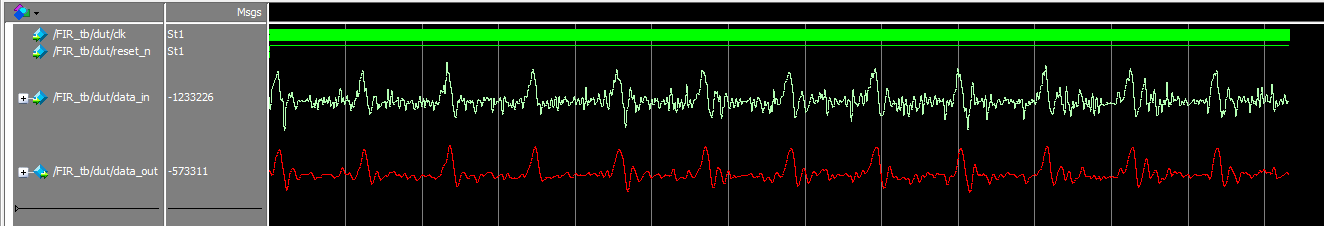


Sau khi áp dụng bộ lọc thì phổ tín hiệu có được như hình:



Như mong muốn hình tín hiệu ở tần số 30Hz đã được loại bỏ.

Kết quả đạt được sau khi mô phỏng:



Nhận thấy kết quả sau khi lọc đã phần nào ít nhiễu hơn dữ liệu đầu vào, nhưng bộ lọc và cách thức loại bỏ nhiễu chưa được thực hiện tối ưu nên nhiễu ở tín hiệu ECG ngõ ra vẫn chưa được loại bỏ hoàn toàn.