دانشکده مهندسی برق

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دستور کار

آزمایشگاه پردازش سیگنال دیجیتال

آزمایش اول

مقدمه

MATLAB یکی از نرم افزارهای انجام محاسبات ریاضی است. واژه MATLAB هم به معنی محیط محاسبات رقمی و هم به معنی خود زبان برنامه‌نویسی مربوطه است که از ترکیب دو واژه Matrix و Laboratory ایجاد شده است. این نام حاکی از رویکرد ماتریس محور برنامه است، که در آن حتی اعداد عادی هم به عنوان ماتریس در نظر گرفته می‌شوند.

با نرم‌افزار MATLAB می‌توان ماتریس‌ها را به راحتی تغییر داد، توابع یا داده‌ها را ترسیم کرد، الگوریتم‌ها را اجرا کرد و هم‌چنین صفحات رابط میان کاربر و رایانه ایجاد کرد.

برنامه‌های MATLAB اکثرا متن‌باز هستند و در واقع MATLAB مفسر است، نه کامپایلر. قدرت MATLAB از انعطاف‌پذیری و راحت بودن کار با آن ناشی می‌شود، هم‌چنین شرکت سازنده و گروه‌های مختلف، از جمله دانشگاه‌های سراسر دنیا و برخی شرکت‌های مهندسی هرساله جعبه‌ابزارهای خاص کاربردی به آن می‌افزایند که باعث افزایش کارایی و محبوبیت آن شده است.

در این آزمایش شما با دستورات مقدماتی پردازش سیگنال در حوزه زمان و فرکانس آشنا می‌شوید که می‌توانید با مراجعه به Help نرم‌افزار ویژگی‌های پیشرفته‌تر و مثال‌های بیش‌تری را یاد بگیرید.

مراحل آزمایش

دستورات stem و plot

1-1) یک سیگنال سینوسی با طول sec 2، طول گام sec 0.01، دامنه‌ی 5 واحد و فرکانس Hz 1 بسازید و با دستورات stem و plot نمایش دهید. (توجه داشته باشید که نمودار باید دارای عنوان مناسب باشد و هم‌چنین محورها به درستی نام‌گذاری شوند، برای این کار می‌توانید از دستورات xlabel، ylabel و title استفاده کنید.)

دستور subplot

2-1) اکنون یک سیگنال تصادفی با توزیع یکنواخت در بازه‌ی (0.5 ,0.5-) را با سیگنال بالا جمع کرده و هردو سیگنال سینوسی نویزی و بدون نویز را با استفاده از دستور subplot در یک پنجره زیر هم نمایش دهید.

دستور conv

3-1) با استفاده از تابع conv سیگنال نویزی را از یک سیستم moving average با طول 21 عبور دهید ($\frac{1}{M_1+M_2+1}\sum_{k=-M_1}^{M_2} x[n-k]$ با مقادیر $M_1 = 0$ و $M_2 = 20$) و خروجی را در یک نمودار نمایش دهید. در مورد مشاهدات خود و علت تاخیر ایجادشده در خروجی توضیح دهید. برای مقادیر $M_1 = 10$ و $M_2 = 10$ چگونه می‌توان از دستور conv استفاده کرد؟

دستور filter

4-1) مرحله‌ی قبل را این بار با دستور filter تکرار نمایید. آیا خروجی با قسمت قبل یکسان است؟

تعریف تابع

5-1) تابعی به نام singen(ω,n) تعریف کنید که ω و n به ترتیب فرکانس سینوسی و تعداد نمونه‌ها می‌باشد ($0<\omega<\pi$). می‌توانید از دستور filter استفاده نمایید.

$$x[n] = \sin(\omega_0 n)\, u_{-1}[n] \overset{Z.T.}{\Longrightarrow} X(z) = \frac{\sin(\omega_0) z^{-1}}{1 - 2\cos(\omega_0)\, z^{-1} + z^{-2}} \quad |z| > 1$$

بررسی aliasing درحوزه زمان

6-1) سیگنال $x(t) = \cos(2\pi t) + \cos(8\pi t) + \cos(12\pi t)$ را در نظر بگیرید، که دارای مولفه‌های فرکانسی 1 KHz، 4 KHz و 6 KHz می‌باشد (زمان به ms بیان شده است). این سیگنال را در بازه 0 تا 4 ms رسم کرده و سپس با نرخ 5 kHz نمونه برداری کنید و نمونه‌ها را با دایره روی نمودار مشخص کنید. اکنون سیگنال نمونه برداری شده را با فیلتر ایده‌آل بازسازی کرده و خروجی را روی همان نمودار قبل نشان دهید. انتظار دارید سیگنال بازسازی شده به چه شکلی دربیاید؟ حدس خود را اثبات کنید.

سوال: محدود کردن طول فیلتر ایده‌آل چه تاثیری در سیگنال بازسازی شده دارد؟

بررسی aliasing در حوزه فرکانس

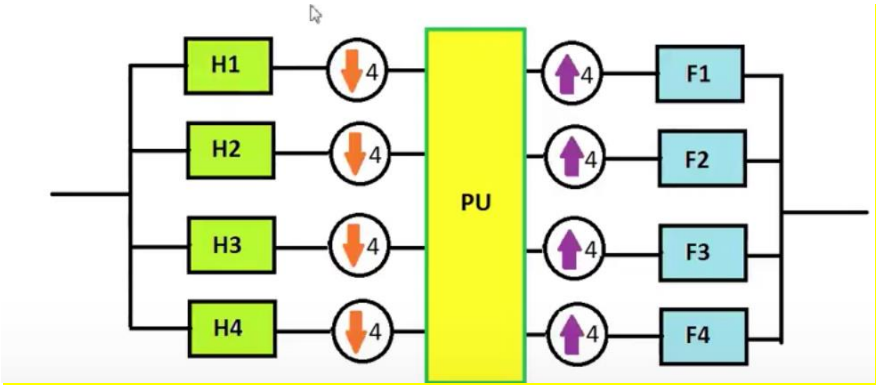
7-1) سیگنال $x(t) = sinc^2(5t)$ را در بازه‌ی [-5, 5] با طول گام‌های 0.01 رسم کنید. سپس این سیگنال را با نرخ‌های 20 Hz، 10 Hz، 5 Hz و 4 Hz نمونه‌برداری کنید. طیف هر دو سیگنال را در هر حالت در یک پنجره در دو نمودار زیرهم رسم کرده و با هم مقایسه کنید. در مورد علت تفاوت احتمالی بحث کنید.

بررسی aliasing در حوزه فرکانس

8-1) سیگنال $x(t) = \text{sinc}(2t)$ را در بازه‌ی [-5, 5] در 256 نقطه نمونه‌برداری شده‌است. نرخ نمونه‌برداری را با نسبت 1/2، 3 و 3/2 تغییر داده و سیگنال اصلی و نمونه‌برداری شده و طیف نرمالیزه‌ی آن‌ها در هر حالت را در یک نمودار رسم کنید.

بانک فیلتری چند نرخی

بلوک دیاگرام زیر را در نظر بگیرید:



این ساختار یک بانک فیلتری چهارکاناله با یک واحد پردازش را نشان می‌دهد. در این قسمت با یکی از کاربردهای ساده‌ی این ساختارها آشنا می‌شویم.

9-1-الف) سیگنال زیر را در نظر بگیرید که از چهار سینوسی با فرکانس‌های مختلف تشکیل شده است:

$$x(t) = \sum_{i=1}^{4} \cos(2\pi f_i t), \quad f_1 = \frac{\pi}{16} \quad f_2 = \frac{5\pi}{16} \quad f_3 = \frac{9\pi}{16} \quad f_4 = \frac{13\pi}{16}$$

سیگنال x(t) را با استفاده از دستورهای fft و fftshift در حوزه فرکانس رسم کنید.

9-1-ب) با استفاده از دستور xlsread ضرایب فیلترهای تجزیه و ترکیب را import کنید.

توجه: ضرایب مربوط به فیلترهای تجزیه در sheet 1 و ضرایب مربوط به فیلترهای ترکیب در sheet 2 می‌باشد. هرکدام متشکل از یک ماتریس 32×4 می‌باشد که هرسطر شامل ضرایب فیلتر متناظر خود است.

8-1-ج) با استفاده از بانک فیلتری مولفه‌ی فرکانسی اول را 2 برابر، دومی را حذف، سومی بدون تغییر و مولفه‌ی چهارم را با ضریب 0.5 تضعیف نمایید. نتیجه را با رنگی متفاوت روی نموداری که در قسمت 9-1-الف رسم کردید نمایش دهید.

آزمایش دوم

مقدمه

بسته به کاربرد و سخت‌افزار، عملیات فیلترینگ دیجیتال می‌تواند به صورت بلوکی یا نمونه به نمونه اعمال شود. در حالت پردازش بلوکی، سیگنال ورودی به صورت یک بلوک بزرگ از نمونه‌های سیگنال در نظر گرفته می‌شود. سیگنال فیلتر شده، حا صل کانولو شن این بلوک و فیلتر ا ست که خود یک بلوک از نمونه‌ها ست. اگر سیگنال ورودی خیلی بزرگ و یا با طول بی‌نهایت باشد این روش به تغییراتی جزئی نیاز دارد، به طور مثال شکستن ورودی به چندین بلوک.

یک روش دیگر، پردازش نمونه به نمونه می‌باشد، بدین صورت که هرگاه یک نمونه ورودی می‌رسد فیلتر شده و نمونه خروجی متناظر تولید می شود. این روش در کاربردهای بلادرنگ که ورودی بسیار طولانی ا ست سودمند می‌باشد. این روش هم‌چنین در فیلترینگ وفقی (adaptive) که خود فیلتر بعد از هر عمل فیلترینگ تغییر می‌کند کاربرد دارد.

در این آزمایش ابتدا روش فیلترینگ با کانولو شن برر سی شده و سپس مقدمه‌ای پردازش سیگنال در حوزه‌ی فرکانس مطرح خواهد شد.

مراحل آزمایش

کانولوشن

کانولوشن یک فیلتر $h_n, n = 0, 1, 2, \dots, M$ با یک سیگنال علی زمان‌محدود $x_n، n = 0, 1, 2, \dots, L$، با معادله‌ی زیر محاسبه می‌شود:

$$y_n = \sum_{m=0}^{\min(-l+n,M)} h_m x_{n-m} \qquad n = 0, 1, 2, \dots, M + L - 1 \qquad (1-2)$$

2-1-الف) تابعی به نام myconv (برای جلوگیری از ابهام با تابع conv داخلی متلب) بنویسید که معادله بالا را پیاده‌سازی کند. این تابع باید به صورت زیر باشد:

$$y = myconv(h, x) \qquad (2-2)$$

که h و x ورودی‌ها و y خروجی فیلتر می‌باشد.

2-1-ب) یک فیلتر شبه‌انتگرال‌گیر را که با رابطه ورودی-خروجی زیر تعریف می‌شود، درنظر بگیرید:

$$y[n] = 0.1(x[n] + x[n-1] + x[n-2] + \cdots + x[n-9]) \qquad (3-2)$$

یک سیگنال ورودی موج‌مربعی را با طول L=200 و دوره تناوب K=50 نمونه را به آن اعمال کنید. با استفاده از تابع myconv، خروجی y[n] را محاسبه کرده و نمودار میله‌ای ورودی و خروجی را رسم کرده و در مورد مشاهدات خود توضیح دهید.

2-1-ج) قسمت قبل را برای فیلتر زیر تکرار نمایید.

$$h[n] = \begin{cases} 0.25(075)^n, & 0 \le n \le 14 \\ 0, & otherwise \end{cases} \qquad (4-2)$$

2-1-د) قسمت (2-1-ب) را برای فیلتر با تابع تبدیل زیر تکرار نمائید:

$$H(z) = \frac{1}{5}(1 - z^{-1})^5 \qquad (5-2)$$

این فیلتر به عنوان یک مشتقگیر مرتبه 5 عمل می‌کند.

فیلترینگ سیگنال‌های نویزی

سیگنال x[n] مجموع یک سیگنال مطلوب s[n] و تداخل v[n] می‌باشد:

$$x[n] = s[n] + v[n] \qquad (6-2)$$

$$s[n] = \sin(\omega_2 n) \qquad (7-2)$$

$$v[n] = \sin(\omega_1 n) + \sin(\omega_3 n) \qquad (8-2)$$

$$\omega_1 = 0.05\pi$$

$$\omega_2 = 0.20\pi$$

$$\omega_3 = 0.35\pi$$

به منظور حذف v[n]، سیگنال x[n] با یک فیلتر میان‌گذر FIR فیلتر می‌شود، که به گونه‌ای طراحی شده‌است که فرکانس ω_2 را عبور داده و فرکانس‌های ω_1 و ω_3 را عبور ندهد.

یک نمونه از چنین فیلتری از مرتبه M=100 می‌تواند با استفاده از پنجره Hamming طراحی شود که پاسخ ضربه زیر را داشته‌باشد:

$$h[n] = w[n]\frac{\sin\left(\omega_b\left(n-\frac{M}{2}\right)\right)-\sin\left(\omega_a\left(n-\frac{M}{2}\right)\right)}{\pi\left(n-\frac{M}{2}\right)} \qquad 0 \le n \le M \qquad (9-2)$$

که $\omega_a = 0.15\pi$ و $\omega_b = 0.25\pi$ و باند موثر $[\omega_a, \omega_b] = [0.15\pi, 0.25\pi]$ است. w[n] پنجره Hamming می‌باشد که به‌صورت زیر تعریف می‌شود:

$$w[n] = \begin{cases} 0.54 - 0.46\sin\left(\frac{2\pi n}{M}\right), & 0 \le n \le M \\ 0 \quad , & otherwise \end{cases} \qquad (10-2)$$

برای جلوگیری از مشکلات محاسباتی در $n = M/2$ می‌توانید از تابع sinc متلب استفاده کنید.

2-2-الف) در یک نمودار x[n] و s[n] را بر حسب n رسم کنید.

2-2-ب) x[n] را از طریق فیلتر h[n] با دستور filter متلب، فیلتر کرده و s[n] و خروجی فیلتر شده y[n] را در یک نمودار رسم کنید. جدای از یک تاخیر ناشی از فیلتر، y[n] باید تقریبا شبیه s[n] باشد.

2-2-ج) با استفاده از جعبه‌ابزار fdatool یک فیلتر میان‌گذر با ویژگی‌های زیر طراحی کرده و قسمت قبل را تکرار نمایید.

$\omega_{s1} = 0.1$

$\omega_{p1} = 0.15$

$\omega_{p2} = 0.25$

$\omega_{s2} = 0.3,$

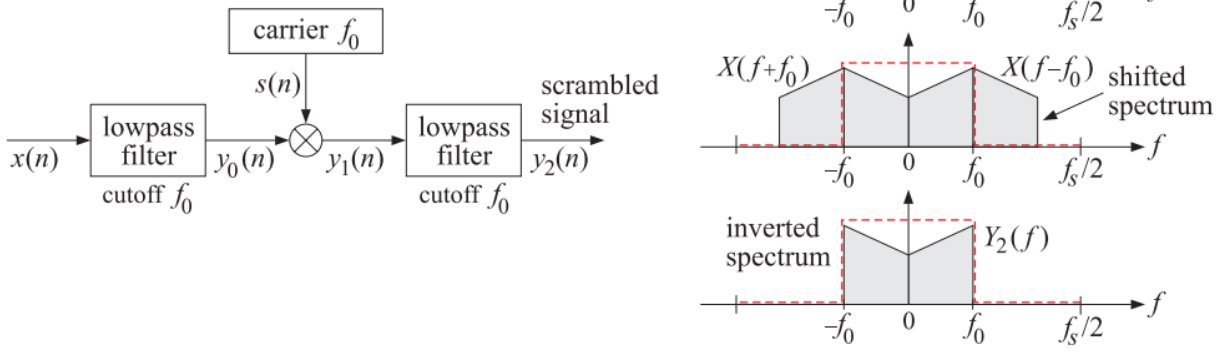
$A_{s1} = 60dB$

$A_p = 0dB$

$A_{s2} = 60dB$

درهم‌سازی صوت(voice scrambling)

یک درهم‌ساز صوت ساده از طریق معکوس کردن طیف کار می‌کند. اگرچه این روش امن‌ترین راه کد کردن نیست اما این روش را به عنوان کاربردی از فیلترهای پایین‌گذر و مدولاسیون AM در نظر می‌گیریم. عملیات کلی در زیر نشان داده شده‌است:



ابتدا سیگنال صدای نمونه‌برداری‌شده‌ی x[n] با فیلتر پایین‌گذر h[n] فیلتر می‌شود (فرکانس قطع آن f_0 به اندازه‌ی کافی بالا هست تا اعوجاج ایجاد نکند). نرخ نمونه‌برداری f_s به گونه‌ای انتخاب می‌شود که $f_s > 4f_0$. عمل فیلترینگ می‌تواند با رابطه‌ی کانولوشن نمایش داده شود:

$$y_0[n] = \sum_m h[m]x[n-m] \qquad (11-2)$$

سپس خروجی فیلتر با یک حامل کسینوسی با فرکانس f_0 مدوله می‌شود:

$$y_1[n] = s[n]y_0[n] \quad , \quad s[n] = 2\cos(\frac{2\pi f_0}{f_s}n) \qquad (12-2)$$

این ضرب در سیگنال حامل باعث می‌شود که طیف سیگنال شیفت پیدا کند و مرکز آن $\pm f_0$ شود. نهایتا سیگنال مدوله شده $y_1[n]$ دوباره از همان فیلتر پایین‌گذر عبور داده می شود که مولفه‌های طیف با $|f| > f_0$ را حذف می‌کند. نتیجه سیگنال $y_2[n]$ با طیف معکوس نسبت به $y_0[n]$ می‌شود. عملیات آخر مطابق زیر است:

$$y_2[n] = \sum_m h[m]y_1[n-m] \qquad (13-2)$$

برای بازیابی صدای اصلی از صدای درهم‌ریخته می‌توان عملیات بالا را دوباره روی آن اعمال کرد. چون طیف معکوس‌شده دوباره معکوس می‌شود و می‌توانیم طیف اصلی را بازیابی کنیم.

3-2-الف) با اسفاده از دستور audioread متلب سیگنال صوت با نام 'Audio01.wav'را بارگذاری کنید.

3-2-ب) با استفاده از fdatool متلب یک فیلتر پایین‌گذر با مشخصات زیر طراحی کنید:

$$f_p = 10000, f_s = 12000, A_p = 0.5, A_s = -60$$

و سپس ضرایب آن را در قالب MAT-File و نام متغیر filter ذخیره کنید.

2-3-ج) فیلتر طراحی شده در ق سمت قبل را به سیگنال صوت اعمال کرده و سیگنال حا صله را به سیگنال حامل ضرب کنید. سپس دوباره از همان فیلتر عبور دهید. برای شنیدن نتیجه می‌توانید از دستور sound استفاده کنید.

2-3-د) با تکرار عملیات فوق سیگنال صوت اصلی را بازیابی کرده و به آن گوش دهید. آیا با سیگنال صدای اصلی مطابقت دارد؟

تبدیل فوریه‌ی گسسته و تبدیل فوریه‌ی زمان کوتاه

سیگنالی در نظر بگیرید که جمع یک سینوسی، یک سیگنال Chirp و یک ضربه است. فرکانس سیگنال سینو سی 100 Hz ا ست که در بازه‌ی [0, 2] با فرکانس 500 Hz نمونه‌برداری شده‌ا ست. فرکانس سیگنال Chirp هم به‌صورت خطی در بازه‌ی [0, 2] از 400 Hz به 200 Hz تغییر می‌کند. سیگنال ضربه در نقطه‌ی 250 ام با دامنه‌ی 50 است.

2-4-الف) طیف این سیگنال را برحسب فرکانس با استفاده از دستور fft رسم کنید.

2-4-ب) با استفاده از دستور spectrogram تبدیل فوریه‌ی زمان کوتاه آن را رسم کنید. برای این کار از پنجره‌ی Hamming استفاده کنید. طول پنجره را 256 و 512 قرار دهید. با توجه به مشاهدات خود تفاوت تبدیل فوریه گسسته و تبدیل فوریه‌ی زمان کوتاه را شرح دهید. با تغییر طول پنجره رزولوشن زمانی و رزولوشن فرکانسی چه تغییری می‌کنند؟

تبدیل موجک (Wavelet Transform)

از دیدگاه تئوری سیگنال، تبدیل موجک تصویر کردن سیگنال روی مجموعه‌ای از توابع پایه به نام موجک‌هاست. تبدیل موجک به‌عنوان جایگزینی برای STFT ارائه شد تا بر مشکلات مربوط به رزولوشن زمانی و فرکانسی آن غلبه کند. برخلاف STFT که رزولوشن زمانی یکنواختی برای همه‌ی فرکاتس‌ها دارد، تبدیل موجک برای فرکانس‌های بالا، رزولوشن زمانی بالا و رزولوشن فرکانسی پایین و برای فرکانس‌های پایین، رزولوشن زمانی پایین و رزولوشن فرکانسی بالا فراهم می‌کند. در حالت عمومی، موجک یک تابع با میانگین صفر است:

$$\int_{-\infty}^{+\infty} \psi(t)dt = 0$$

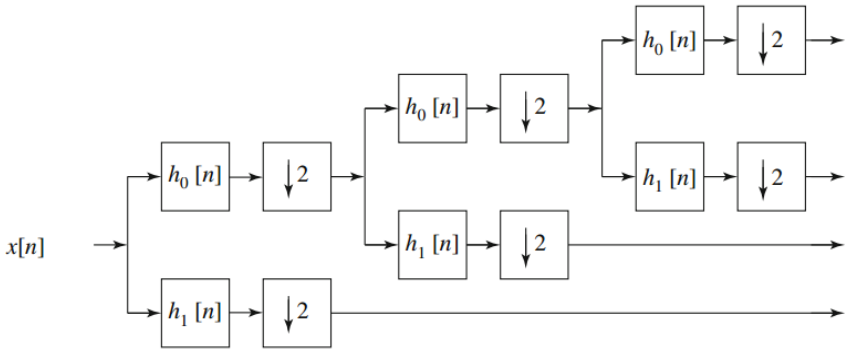
که شروطی را ارضا می‌کند تا اطمینان حاصل شود که می‌تواند در تجزیه‌ی چند رزولوشنه استفاده شود. می‌توان با جابجایی و تغییر مقیاس یک موجک مادر ψ خانواده‌ای از موجک‌ها داشته‌باشیم:

$$\psi_{s,u}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-u}{s}\right)$$

تبدیل موجک تابع f در زمان u و مقیاس s به‌صورت زیر تعریف می‌شود:

$$\mathcal{W}(f,s,u) = \int_{-\infty}^{+\infty} f(t)\psi_{s,u}(t)dt$$

موجک‌های گسسته هم از جابجایی و تغییر مقیاس با گام‌های گسسته از موجک مادر حاصل می شوند. اصول تبدیل موجک گسسته مشابه روش Subband Coding است. بلوک دیاگرام زیر تبدیل موجک سه سطحی را نشان می‌دهد.



همان طور که مشاهده می‌کنید سیگنال x[n] به سه سطح تجزیه شده است. پروسه‌ی بالا یک بانک فیلتری(filter bank) با سه سطح است. تبدیل موجک یک سیگنال گسسته مطابق زیر اجرا می‌شود:

ابتدا x[n] از هر دو فیلتر پایین‌گذر $h_0[n]$ و بالاگذر $h_1[n]$ عبور می‌کند. سپس سیگنال‌های حاصله با فاکتور 2، downsample می‌شوند. سیگنال‌های حاصله یا دو مجموعه ضرایب، تبدیل سطح 1 سیگنال نامیده می‌شوند. اگر این پروسه مکررا روی حاصل خط پایین‌گذر اعمال شود، تبدیل‌های سطح بالاتر به دست می‌آیند.

5-2)یکی از کاربردهای تبدیل موجک Denoising است. با استفاده از تابع wnoise از جعبه ابزار موجک، سیگنال نویزی ایجاد کرده و با استفاده از دستور dwt چند سطح از آن تبدیل موجک بگیرید. مشاهدات خود را شرح دهید.

آزمایش سوم

مقدمه

قسمت اول این آزمایش اصول بهبود سیگنال و کاهش نویز را با یک مثال ساده فیلتر IIR نشان می‌دهد. قسمت دوم فعل و انفعال بین پاسخ گذرا و مانا و trade off بین ثابت زمانی و تیزی فیلتر را بیان می‌کند. قسمت سوم نگاهی است بر مثال‌هایی از پردازنده‌های دینامیک برای سیگنال‌های صوتی، مانند فشرده سازها، محدودکننده‌ها، منبسط کننده‌ها و گیت‌های نویز.

مراحل آزمایش

بهبود سیگنال و کاهش نویز(Signal Enhancement and Noise Reduction)

یک سیگنال سینوسی نویزی با فرکانس $f_0 = 500\ Hz$ را در نظر بگیرید که با نرخ $f_s = 10\ KHz$ نمونه‌برداری شده است:

$$x[n] = \cos(\omega_0 n) + v[n] \qquad (1-3)$$

که $\omega_0 = 2\pi f_0 / f_s$ فرکانس دیجیتال و $v[n]$ نویز می‌باشد.

مطلوب، طراحی فیتر $H(j\omega)$ به منظور استخراج سیگنال مطلوب $s[n]$ از سیگنال نویزی $x[n]$ می‌باشد. چنین فیلتری باید دو ویژگی دا شته با شد: اولا باید مولفه نویز را تا حد امکان حذف کند، ثانیا باید به سیگنال مطلوب اجازه دهد که بدون تغییر عبور کند، به جز احتمالا با یک تاخیر زمانی.

خواسته دوم با طراحی یک فیلتر میان‌گذر که باند عبورش با سیگنال مطلوب تلاقی دارد برآورده می‌شود. مولفه‌ی نویز معمولا یک نویز سفید ا ست که توان آن به صورت یکنواخت روی کل محور فرکانسی توزیع شده است. بعد از فیلترینگ، توان مجموع نویز کاهش خواهد یافت زیرا تنها، توانی که در باند عبور قرار گرفته باقی خواهد ماند.

در مثال ما باند عبور سیگنال مطلوب فقط فرکانس ω_0 می‌باشد. بنابراین می‌بایست فیلتر میانگذری طراحی کنیم که در فرکانس ω_0 بهره واحد دا شته‌با شد، یعنی $|H(j\omega_0)| = 1$. ساده‌ترین انتخاب ممکن برای چنین فیلتری تشدیدکننده دوطبقه با قطب‌هایی در $Re^{\pm j\omega_0}$ می‌باشد که R باید برای پایداری به گونه‌ای انتخاب شود که $0 < R < 1$. تابع تبدیل، به فرم زیر خواهد بود:

$$H(z) = \frac{G}{(1 - Re^{j\omega_0}z^{-1})(1 - Re^{-j\omega_0}z^{-1})} = \frac{G}{1 + a_1z^{-1} + a_2z^{-2}} \qquad (2-3)$$

که $a_1 = -2Rcos(\omega_0)$ و $a_2 = R^2$ می‌باشد. بهره G برای اطمینان از $|H(\omega_0)| = 1$ زیر می‌باشد:

$$G = (1 - R)(1 - 2Rcos(2\omega_0) + R^2)^{\frac{1}{2}} \qquad (3-3)$$

در حوزه زمان این فیلتر با معادله دیفرانس زیر توصیف می‌شود:

$$y[n] = -a_1y[n-1] - a_2y[n-2] + Gx[n] \qquad (4-3)$$

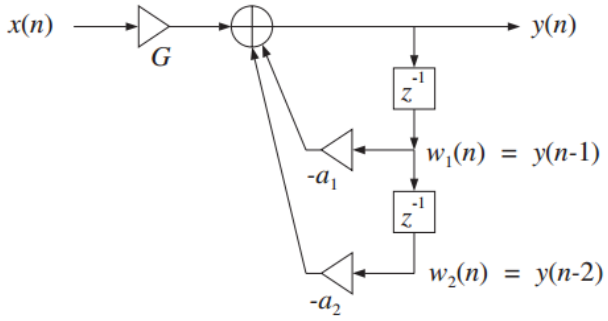
پاسخ ضربه این فیلتر می‌تواند به صورت زیر نشان داده شود:

$$h[n] = \frac{G}{\sin(\omega_0)}R^n sin(\omega_0 n + \omega_0)\,, \quad n = 0, 1, 2, \ldots \qquad (5-3)$$

در نهایت توان دوم پاسخ دامنه می‌تواند به صورت زیر نشان داده شود:

$$|H(j\omega)|^2 = \frac{G^2}{[1 - 2Rcos(\omega - \omega_0) + R^2][1 - 2Rcos(\omega + \omega_0) + R^2]} \qquad (6-3)$$

یک بلوک‌دیاگرام برای پیاده‌سازی معادله دیفرانس و تابع تبدیل بالا در زیر نشان داده شده‌است:



شکل 3-4

ابتدا روابط بالا را اثبات کنید. ســـپس به ازای هرکدام از مقادیر R=0.8، R=0.90 و R=0.99 موارد زیر را انجام دهید:

3-1-الف) توان دوم پاسخ دامنه $\left|H(e^{j\omega})\right|^2$ را در بازه‌ی فرکانسی $0 < \omega < \pi$ رسم کنید.

پیشنهاد:برای رسم می‌توانید $\left|H(e^{j\omega})\right|^2$ را در 500 نقطه با فاصله‌های مساوی در بازه مذکور محاسبه کنید.

توجه:$\omega = \frac{2\pi f}{f_s}$

3-1-ب) پاسخ دامنه h[n] را با ارسال ضربه واحد در ورودی معادله دیفرانس و تکرار در زمان(با شرایط اولیه صفر) محاسبه کنید. مقادیر محاسبه شده را با مقادیری که از فرمول 3-5 به دست آمده مقایسه کنید. مقادیر h[n]/G را به ازای $0 \le n \le 300$ رسم کنید.

3-1-ج) با استفاده از تابع تولید عدد تصادفی متلب، randn، 301 نمونه سیگنال تصادفی تولید کنید. سپس با استفاده از الگوریتم پردازش نمونه زیر x[n] را با H(z) فیلتر کرده و سیگنال خروجی y[n] را به ازای n در بازه‌ی $0 \le n \le 300$ محاسبه و رسم کنید. در همان نمودار سیگنال بدون نویز s[n] را قرار دهید. بده‌بستان بین تیزی فیلتر میان‌گذر و سرعت پاسخ (سرعت رسیدن به حالت مانا) را مورد بحث قرار دهید.

For each input sample x do:

$y = -a_1 w_1 - a_2 w_2 + Gx$

$w_2 = w_1$

$w_1 = y$

3-1-د) سیگنال نویز v[n] را جداگانه از این فیلتر میان‌گذر عبور داده و نویز خروجی فیلتر شده متناظر $y_v[n]$ را حساب کنید. در دو نمودار مجزا v[n] و $y_v[n]$ را برجسب n رسم کنید. توضیح دهید که چرا نویز فیلترشده بیشتر شبیه سینوسی است تا نویز.

3-1-ه) می‌توان نشان داد که برای یک ورودی نویز سفید میانگین صفر v[n] با واریانس σ_v^2 که از یک فیلتر پایدار علی h[n] عبور می‌کند، سیگنال نویز خروجی $y_v[n]$ واریانس $\sigma_{y_v}^2$ را خواهد داشت که:

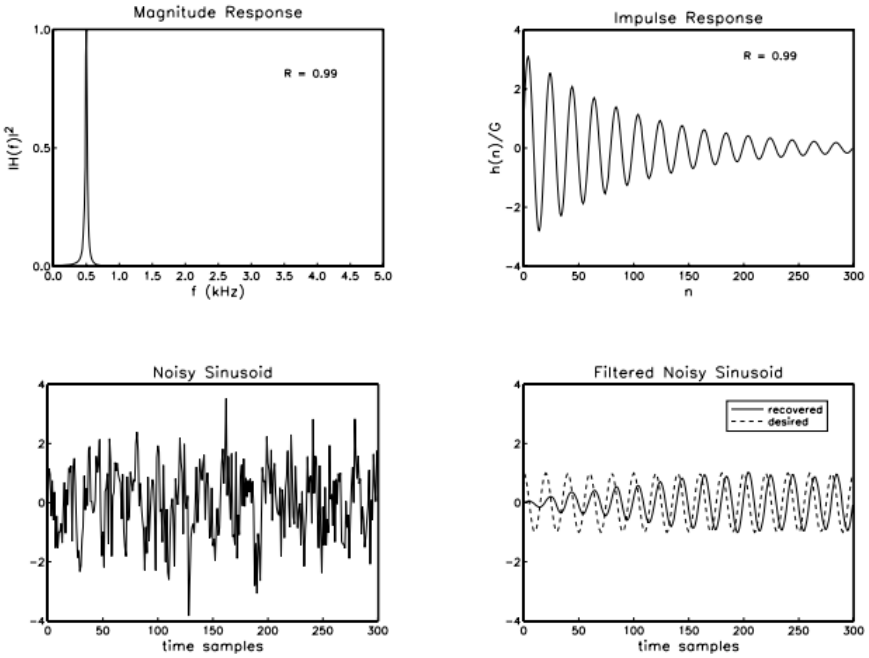
$$\frac{\sigma_{y_v}^2}{\sigma_v^2} = \sum_{n=0}^{\infty} h^2[n] \qquad (7-3)$$

سمت چپ معادله نسبت کاهش نویز (NRR) می‌باشد، زیرا تاثیر فیلتر را بر نویز ورودی تعیین می‌کند. برای پاسخ ضربه خصوصی 3-5 اثبات کنید NRR مطابق رابطه زیر است:

$$\sum_{n=0}^{\infty} h^2[n] = \frac{1+R^2}{(1+R)(1+2R\cos(\omega_0)+R^2)} \qquad (8-3)$$

(راهنمایی: h[n] را به فرم $h[n] = A_1 p_1^n + A_2 p_2^n$ بنویسیدکه p_1 و p_2 قطب‌های فیلتر هستند.)

با استفاده از سری‌های v[n] و $y_v[n]$ که در قسمت 3-1-د تولید کردید و با تابع متلب std یک مقدار تقریبی برای سمت چپ معادله 3-7 حساب کنید و آن را با مقدار تئوری 3-8 مقایسه کنید.



شکل 3-2

ویژگی‌های حالت گذرا و حالت مانا

پاسخ سینوسی فیلتر مرتبه‌ی دوم با قطب‌های p_1 و p_2 فرم دقیق زیر را خواهد داشت:

$$x[n] = \cos(\omega_0 n) \Rightarrow \; y[n] = |H(\omega_0)| \cos(\omega_0 n + \theta_0) + B_1 p_1^n + B_2 P_2^n \qquad (9-3)$$

که شیفت فاز θ_0 مقدار فاز پاسخ فیلتر در فرکانس ω_0 می‌باشد و B_1 و B_2 به پاسخ‌های خصوصی تابع تبدیل بستگی دارد. در این بخش شما خواهید دید که ترم مانا به خوبی خروجی یک سیگنال زمان‌کوتاه را تعیین می‌کند و تاثیر ترم‌های گذرا بر ثابت زمانی فیلتر را مشاهده خواهید کرد.

سیگنال چندضابطه‌ای زیر را در نظر بگیرید:

$$x(t) = \begin{cases} \cos(2\pi f_1 t), 0 \le t < 2sec \\ \cos(2\pi f_2 t), 2 \le t < 4sec \\ \cos(2\pi f_3 t), 4 \le t < 6sec \end{cases} \qquad (10-3)$$

که $f_1 = 4Hz$، $f_2 = 8Hz$ و $f_3 = 12Hz$ می‌باشد.

این سیگنال با نرخ $f_s = 400\frac{\text{sample}}{\text{second}}$ نمونه‌برداری می‌شود. دو فیلتر زیر که در نرخ f_s عمل می‌کنند notch filter می‌باشند که در فرکانس $f_2 = 8Hz$ یک شکاف دارند. بنابراین مشخصه میانی $x(t)$ را از بین می‌برند:

$$H_1(z) = \frac{0.969531 - 1.923772z^{-1} + 0.969531z^{-2}}{1 - 1.923772z^{-1} + 0.939063z^{-2}} \qquad (11-3)$$

$$H_2(z) = \frac{0.996088 - 1.976468z^{-1} + 0.996088z^{-2}}{1 - 1.976468z^{-1} + 0.992177z^{-2}} \qquad (12-3)$$

فیلتر اول پهنای باند 3db برابر $\Delta f = 4Hz$ و دومی پهنای $\Delta f = 0.5Hz$ را دارد. پاسخ‌های دامنه $|H(f)|$در دو صحفه بعد نشان داده شده‌اند.

در این آزمایش ما رابطه‌ی بین پهنای شکاف و ثابت زمانی حالت گذرا را بررسی خواهیم کرد. فیلتر اول پهنای زیاد و ثابت زمانی کوتاه دارد و فیلتر دوم پهنای کم و ثابت زمانی طولانی دارد. Notch filter های مورد بحث ما، فرم کلی زیر را دارند:

$$H(z) = \frac{1}{1+\beta}.\frac{1 - 2\cos(\omega_0)\,z^{-1} + z^{-2}}{1 - \frac{2\cos(\omega_0)}{1+\beta}z^{-1} + \frac{1-\beta}{1+\beta}z^{-2}} \qquad (13-3)$$

که فرکانس شکاف و پهنای شکاف به ترتیب f_0 و Δf_0 می‌باشند و:

$$\omega_0 = \frac{2\pi f_0}{f_s} \qquad \Delta\omega = \frac{2\pi\Delta f}{f_s} \qquad \beta = \tan(\frac{\Delta\omega}{2})$$

3-2-الف) تحقق کانونی هر فیلتر را رسم کنید و الگوریتم پردازش نمونه را با بافرهای خط تاخیر بنویسید.

3-2-ب) زمان نشست 1% هردو فیلتر را حساب کنید.

3-2-ج) اگر $x(t_n)$ ورودی نمونه‌برداری شده‌ی $x(t)$ باشد، $x(t_n)$ را برحسب t_n روی یک دوره‌ی 6 sec رسم کنید. این ورودی را با فیلتر $H_1(z)$ فیلتر کنید و خروجی $y(t_n)$ را برحسب t_n رسم کنید.

3-2-د) دقت کنید که با چه سرعتی مشخصه‌ی میانی $x(t_n)$ برداشته می‌شود. هم‌چنین توجه کنید که م شخ صه‌های f_1 و f_3 دیگر دامنه‌ی واحد ندارند. ثابت کنید که دامنه(های) حالت گذرا به ترتیب با اعداد پا سخ دامنه $|H(f_1)|$ و $|H(f_2)|$ داده شده‌اند. (شما می‌توانید این کار را با ارتفاع‌های $|H(f_1)|$ و $|H(f_3)|$ روی

مشخصات سیگنال متناظر انجام دهید یا با استفاده از تابع max بیش‌ترین مقدار دو مشخصه را معلوم کرده و سپس آن‌ها را با مقادیر پاسخ محاسبه شده، مقایسه کنید.)

آیا یک شیفت فاز را مشاهده می‌کنید؟ آیا پاسخ گذرای مشاهده شده با ثابت زمانی قسمت 3-2-ب مطابقت دارد؟

3-2-ه) سوال‌های 3-2-ج و 3-2-د را برای فیلتر دوم $H_2(z)$ تکرار نمایید.

3-2-و) روی دو نمودار مجزا پاسخ دامنه‌ی $H_1(f)$ و $H_2(f)$ را برحسب f در بازه‌ی $0 < f < 20$ رسم کنید. نمودارهای مطلوب در صفحه‌ی بعد نشان داده شده‌اند. مقادیر f_1 و f_3 نیز روی نمودارها مشخص شده‌اند.

3-2-ز) برای هر کدام از فیلترها فرکانس 3db پایین و بالا را حساب کرده، f_L و f_H بنامید و آن را روی نمودار قسمت 3-2-و نشان دهید. ثابت کنید که اختلاف f_L و f_H به ترتیب برابر 4 Hz و 0.5 Hz می‌باشد.

3-2-ح) دو peak filter زیر را در نظر بگیریدکه مکمل notch filter های بالا هستند. این‌ها یک قله در f_2 و همان پهنای باند 4 Hz و 0.5 Hz را دارند:

$$H_1(z) = \frac{0.030469(1 - z^{-2})}{1 - 1.923772z^{-1} + 0.939063z^{-2}} \qquad (\text{الف} - 14 - 3)$$

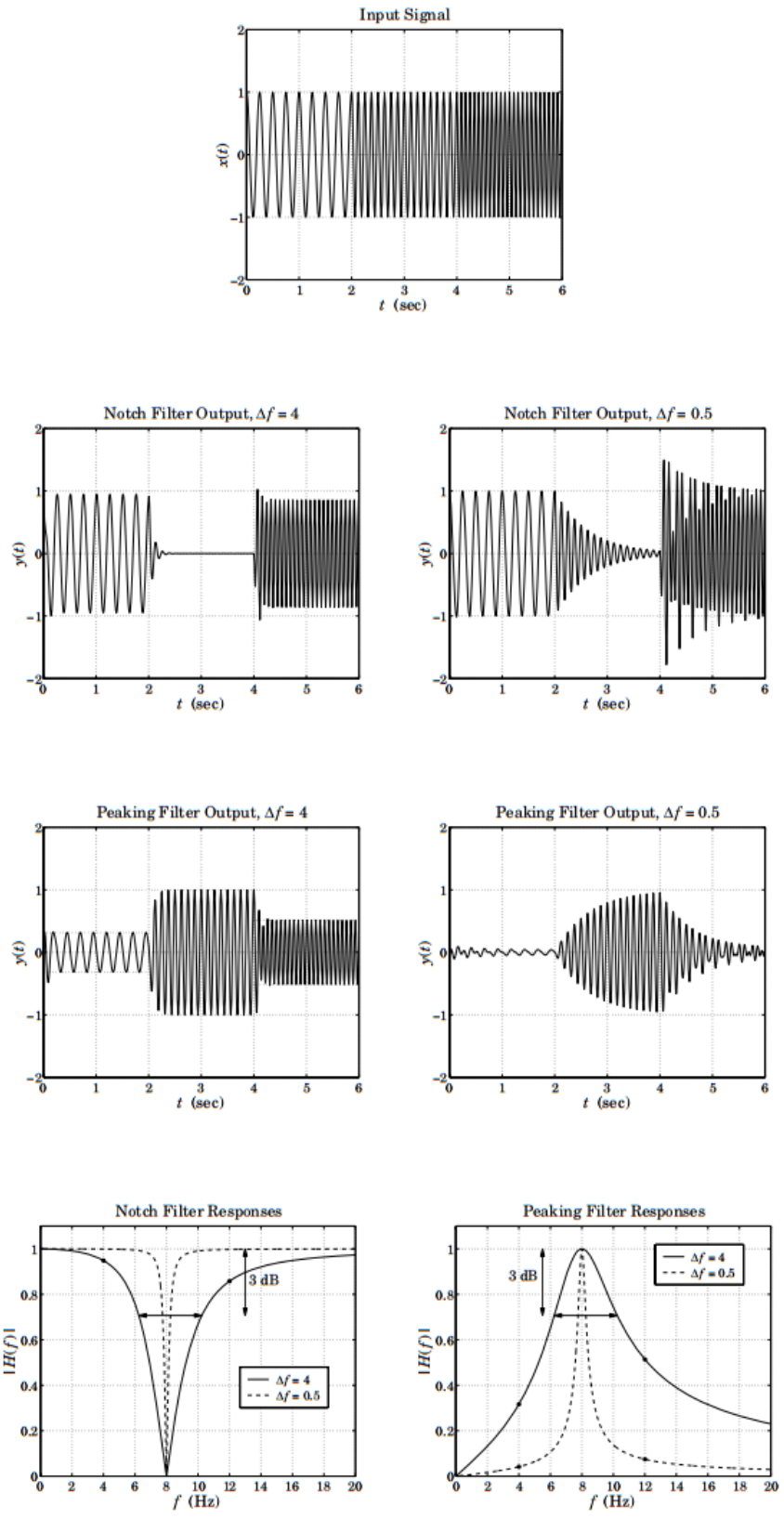
$$H_2(z) = \frac{0.003912(1 - z^{-2})}{1 - 1.976468z^{-1} + 0.992177z^{-2}} \qquad (\text{ب} - 14 - 3)$$

معادله‌ی کلی برای طراحی این فیلترها مطابق زیر است که f_0 و Δf فرکانس قله و پهنای 3db قله می‌باشند:

$$H(z) = \frac{\beta}{1+\beta} \cdot \frac{1 - z^{-2}}{1 - 2\frac{\cos(\omega_0)}{1+\beta} + \frac{1-\beta}{1+\beta}z^{-2}} \qquad (15 - 3)$$

$$\omega_0 = \frac{2\pi f_0}{f_s} \qquad \Delta\omega = \frac{2\pi\Delta f}{f_s} \qquad \beta = \tan(\Delta\omega)$$

بخش‌های 3-2-الف تا 3-2-ز را برای این فیلترها تکرار کنید. انتظار داریم که peak filter مشخصه‌ی میانی ورودی را استخراج کرده و مشخصه‌های f_1 و f_3 را حذف کند.


Input Signal
x(t)
t (sec)
Notch Filter Output, Δf = 4
y(t)
t (sec)
Notch Filter Output, Δf = 0.5
y(t)
t (sec)
Peaking Filter Output, Δf = 4
y(t)
t (sec)
Peaking Filter Output, Δf = 0.5
y(t)
t (sec)
Notch Filter Responses
|H(f)|
3 dB
Δf = 4
Δf = 0.5
f (Hz)
Peaking Filter Responses
|H(f)|
3 dB
Δf = 4
Δf = 0.5
f (Hz)

شکل 3-3

فشرده‌سازها، محدودکننده‌ها، منبسط کننده‌ها و گیت‌های نویز

فشـــرده‌ســـازها، محدودکنندها، منبســـط‌کننده‌ها و گیت‌های نویز کاربردهای وســـیعی در پردازش صـــوت دارند. فشرده‌سازها سیگنال‌های قوی را تضعیف می‌کنند و چون روی بازه‌ی دینامیکی سیگنال‌ها اثر می‌گذارند به آن‌ها پردازنده‌های دینامیکی می‌گویند.

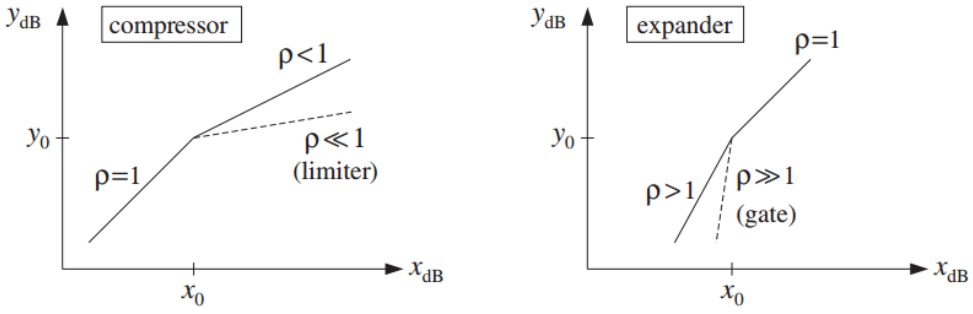
فشرده‌سازها غالبا به منظور کاهش بازه‌ی دینامیکی به کار می‌روند تا بازه‌ی دینامیکی پخش یا سیستم رسانه را مناســـب کنند، مثلا ضـــبط صـــدا روی یک نوار صـــوتی. چندین کاربرد دیگر هم وجود دارد، مانند موســـیقی پس‌زمینه‌ی پنهان‌ساز گزارشگرها، de-essing برای حذف صفیر بیش از حد میکروفون و اثرات خاص دیگر.

منبســـط‌کننده‌ها برای افزایش بازه‌ی دینامیکی ســـیگنال‌ها به منظور کاهش نویز و اثرات خاص گوناگون به کار می‌روند.

یک رابطه‌ی نوعی ورودی-خروجی حالت مانا برای فشرده‌سازها و منبسط‌کننده‌ها برحسب اندازه و دامنه، مطابق زیر است:

$$y = y_0 \left(\frac{x}{x_0}\right)^{\rho} \Rightarrow 20\log_{10}\left(\frac{y}{y_0}\right) = \rho 20\log_{10}\left(\frac{x}{x_0}\right) \qquad (16-3)$$

که x در این‌جا یک ورودی ثابت و x_0 یک مقدار آســـتانه‌ی دلخواه می‌باشـــد و هم‌چنین ρ نســـبت انبســـاط یا فشرده‌سازی را تعریف می‌کند. یک فشرده‌ساز زمانی اثر می‌گذارد که $x \geq x_0$ باشد و $\rho < 1$ است. درحالی که برای منبسط‌کننده‌ها باید $x \leq x_0$ بوده و $\rho > 1$ می‌باشد. شکل 3-4 این روابط را به دسی‌بل نشان می‌دهد، به این ترتیب که 1db تغییر در ورودی باعث ρ dB تغییر در خروجی می‌شود که ρ شیب خط مستقیم بین ورودی و خروجی است.



شکل 3-4

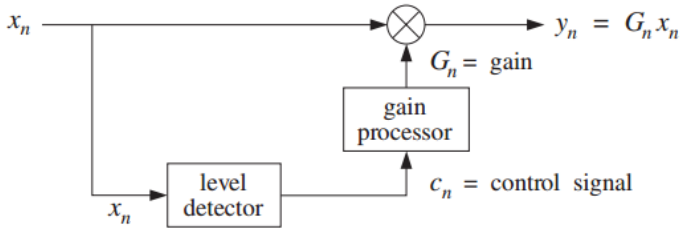
مقادیر عملی نوعی برای فشـــرده‌ســـاز بین $\rho = (1/2$ تا $1/4)$ و برای منبســـط‌کننده‌ها بین $\rho = (2$ تا $4)$ می‌باشد. محدود کننده‌ها فرم‌های اکسترمم فشرده‌سازها هستند که از این‌که سیگنال از یک آستانه حداکثر فراتر رود جلوگیری می‌کنند و شیب خیلی کمی دارند (مثلا 0.1). گیت‌های نویز حالت‌های اکسترمم منبسط کننده‌ها هستند که سیگنال‌های ضعیف را بی‌نهایت تضعیف می‌کنند و بنابراین می‌توانند برای حذف نویز پس‌زمینه به کار روند و شیب خیلی زیادی دارند، مثلا (10).

رابطه‌ی ورودی-خروجی 3-16 تنها برای سیگنال‌های ثابت مناسب است. به ازای $y = Gx$ مشاهده می‌کنیم که بهره‌ی موثر فشرده ساز تابعی غیر خطی از ورودی به فرم $G = G_0 x^{\rho-1}$ می‌باشد. برای سیگنال‌های متغیر با زمان، بهره باید از یک میانگین محلی سیگنال حساب شود که بیانگر سطح سیگنال است.

مدلی از یک فشرده ساز/منبسط‌کننده در شکل 3-5 نشان داده شده‌است. تعیین‌کننده‌ی سطح، یک سیگنال کنترلی c_n تولید می‌کند که بهره‌ی ضرب‌کننده G_n را از طریق یک پردازنده‌ی بهره‌ی غیرخطی کنترل می‌کند. بسته به نوع فشرده ساز سیگنال کنترلی می‌تواند (1) مقدار قله‌ی لحظه‌ای $|x_n|$، (2) پوش x_n و یا (3) مقدار موثر(RMS) x_n باشد. یک مدل ساده از آشکارساز پوش مطابق زیر است:

$$c_n = \lambda c_{n-1} + (1-\lambda)|x_n| \qquad (17-3)$$

معادله‌ی دیفرانس c_n به عنوان یکسوسازی که به دنبال آن یک فیلتر پایین‌گذر است عمل می‌کند. زمان خیز به یک سطح بالای آستانه(جایی که فشرده‌ساز فعال است) ثابت زمانی حمله(attack) نامیده می‌شود. این زمان می‌تواند با یک تاخیر D در ورودی آشکارساز بیش‌تر شود که $|x_{n-D}|$ می‌باشد. زمان افت به یک سطح زیر آستانه (جایی که فشرده‌ساز غیرفعال می‌شود) زمان آزادسازی نامیده می‌شود.



شکل 3-5

به ازای $\lambda = 0$ معادله‌ی 3-17 یک آشـــکارســـاز قله‌ی لحظه‌ای می‌شـــود. این حالت زمانی کاربرد دارد که از فشـرده‌سـاز به عنوان یک محدودکننده اسـتفاده کنیم. اگر در معادله‌ی 3-17 به جای قدر مطلق $|x_n|$ مربع آن $|x_n|^2$ قرار بگیرد سیگنال کنترلی مقدار موثر ورودی را دنبال خواهد کرد.

پردازنده‌ی بهره یک تابع غیرخطی از سیگنال کنترلی است که از رابطه‌ی ورودی-خروجی 3-16 پیروی می‌کند. برای یک فشرده‌ساز می‌توانیم تابع بهره را این‌گونه تعریف کنیم:

$$f(c) = \begin{cases} \left(\frac{c}{c_0}\right)^{\rho-1}, & c \geq c_0 \\ 1, & c < c_0 \end{cases} \qquad (18-3)$$

که c_0 آستانه‌ی دلخواه و $\rho < 1$ می‌باشد.برای یک منبسط‌کننده $\rho > 1$ است و :

$$f(c) = \begin{cases} \left(\frac{c_0}{c}\right)^{\rho-1}, & c < c_0 \\ 1, & c \geq c_0 \end{cases} \qquad (19-3)$$

بنابراین بهره‌ی G_n و سیگنال خروجی نهایی y_n مطابق زیر محاسبه می‌شوند:

$$G_n = f(c_n) \qquad (20-3)$$

$$y_n = G_n x_n \qquad (21-3)$$

فشرده‌سازها/منبسط‌کننده‌ها مثال‌هایی از سیستم‌های پردازش سیگنال وفقی هستند که ضرایب فیلتر(در این‌جا بهره‌ی G_n) وابسته به زمان هستند و خودشان را باطبیعت سیگنال ورودی وفق می‌دهند. آشکارساز سطح 3-17 به عنوان معادله‌ی adaption عمل می‌کند .ثابت زمانی‌های حمله و آزاد سازی، ثابت زمانی‌های آموزش سیستم وفقی هستند. پارامتر λ فاکتور فراموشی سیستم نامیده می‌شود.

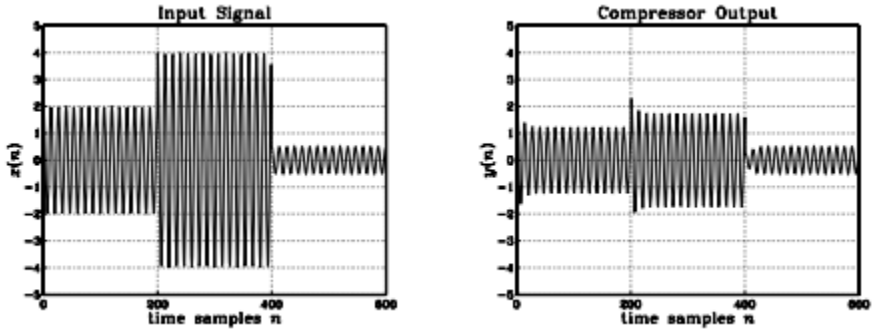
به عنوان یک مثال شبیه‌سازی یک سینوسی با فرکانس $\omega_0 = 0.15\pi \text{ rad/sec}$ را در نظر بگیرید که دامنه‌اش هر 200 نمونه به سه مقدار $A_1 = 2$، $A_2 = 4$ و $A_3 = 0.5$ تغییر می‌کند. همان‌طور که در شکل 3-6 نشان داده شده‌است که $x_n = A_n \cos(\omega_0 n)$ با:

$$A_n = A_1(u_n - u_{n-200}) + A_2(u_{n-200} - u_{n-400}) + A_3(u_{n-400} - u_{n-600}) \qquad (22-3)$$

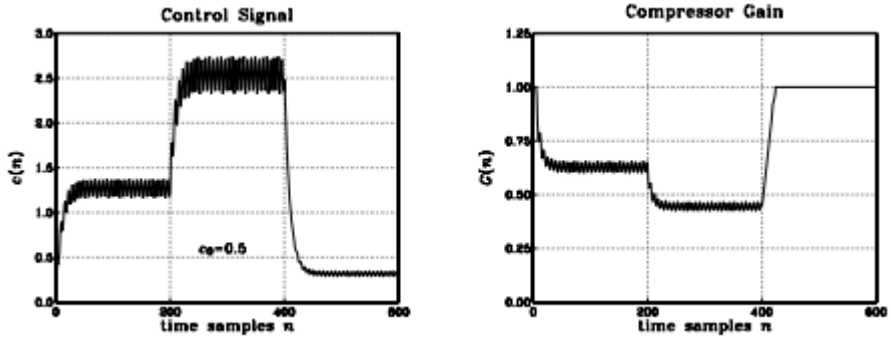
از یک فشرده‌ساز با پارامترهای $\lambda = 0.9$، $c_0 = 0.5$ و $\rho = 0.2$ استفاده می‌شود. خروجی y_n در شکل 3-6 و سیگنال کنترلی c_n و بهره‌ی G_n در شکل 3-7 نشان داده شده‌اند.

در سینوسی اول A_1 و A_2 بالای آستانه قرار می‌گیرند و فشرده می‌شوند. سینوسی، بعد از گذشت زمان آزاد سازی تاثیری نمی‌پذیرد. اگرچه تنها سیگنال‌های قوی‌تر تضعیف می شوند اما کاهش کلی بازه‌ی دینامیکی این‌گونه درک می‌شود که گویا سیگنال‌های ضعیف‌تر نیز تقویت شده‌اند. این ویژگی، ریشه‌ی این عبارت عامه‌پسند، اما گمراه‌کننده است که فشرده سازها سیگنال‌های قوی را تضعیف و سیگنال‌های ضعیف را تقویت می‌کنند.

پرش بین سطوح مانای A_1 و A_2 متناظر تغییر 6db است. چون هردو سطح فشرده می‌شوند سطوح مانای خروجی $6\rho = 3db$ تفاوت خواهند داشت. به منظور حذف برخی از فراجهش‌ها یک تاخیر مناسب در مسیر سیگنال قرار داده می‌شود که خروجی با $y_n = G_n x_{n-d}$ محاسبه می‌شود.



شکل 3-6



شکل 3-7

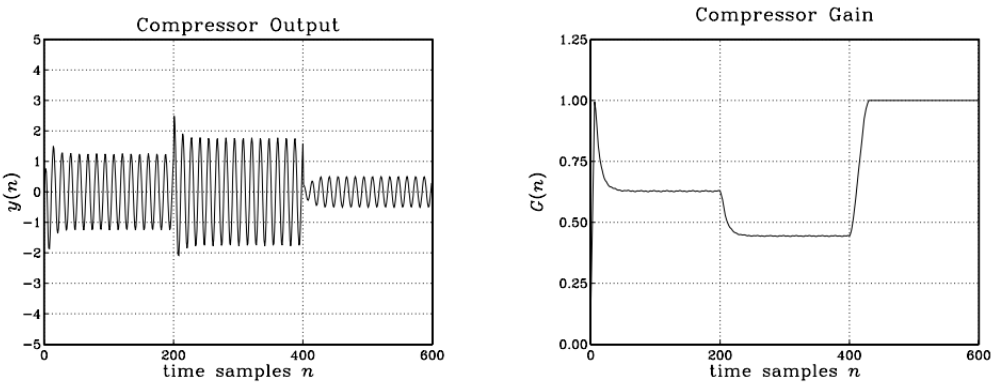
یک بهبودی دیگر هموارکردن بیش‌تر بهره‌ی غیرخطی $g_n = f(c_n)$ با یک فیلتر پایین‌گذر می‌باشد، به گونه‌ای که بهره‌ی نهایی این‌گونه محاسبه شود:

$$G_n = \frac{1}{L}(g_n + g_{n-1} + \cdots + g_{n-L+1}) \qquad (23-3)$$

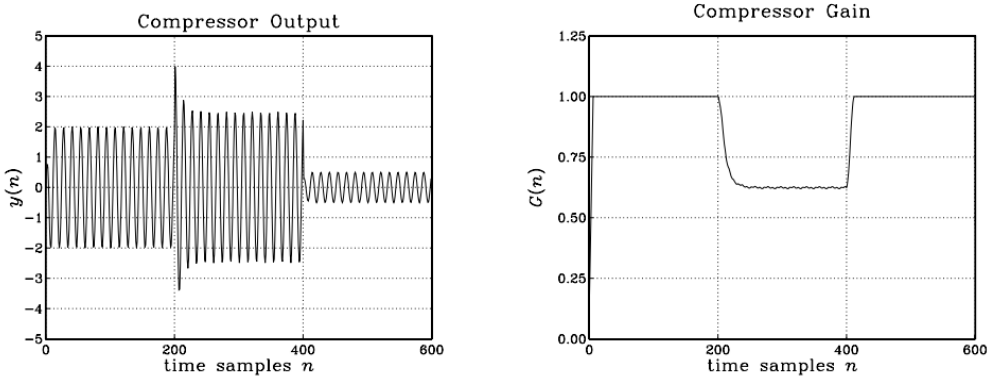
شکل 3-8 سیگنال خروجی و هم‌چنین بهره‌ی یک فشرده ساز را، که با یک هموار ساز 7 نقطه‌ای هموار شده است، نشان می‌دهد. حالت‌های گذرای اولیه‌ی G_n به دلیل حالت گذرای ورودی‌روشن هموارساز ایجاد شده است.

شکل 3-9 سیگنال خروجی و بهره‌ی یک محدودکننده را نشان می‌دهد که نسبت فشرده‌سازی 10:1 دارد، $\rho = 0.1$، و هم‌چنین از یک هموار ساز 7 نقطه‌ای استفاده می‌کند. در این‌جا آستانه تا $c_0 = 1.5$ افزایش یافته تا تنها A_2 بالای آن قرار گیرد و فشرده شود.

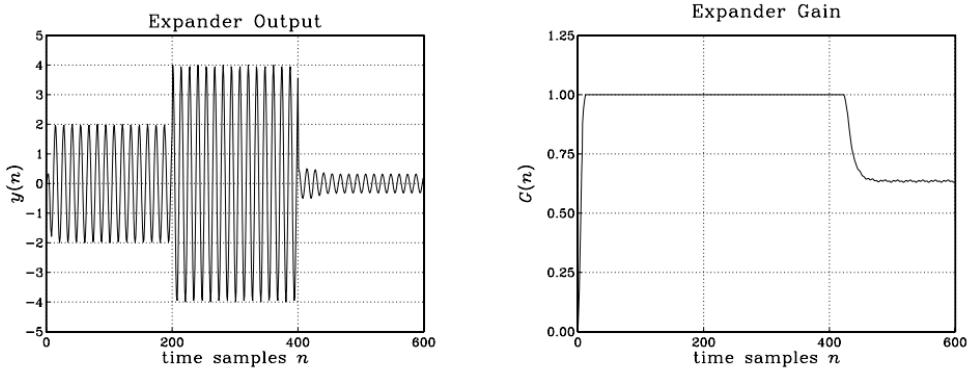
شـــکل 3-10 یک مثال منبســـط‌کننده را با $\lambda = 0.9$، $c_0 = 0.5$، $\rho = 2$ و تابع بهره‌ای که با معادله‌ی 3-19 محا سبه می شود، ن شان می‌دهد که با یک هموار ساز 7 نقطه‌ای هموارتر شده ا ست. تنها A_3 زیر آ ستانه قرار می‌گیرد و ت ضعیف می شود. این باعث می شود که بازه‌ی دینامیکی کلی افزایش یابد. اگرچه منبسط کننده فقط بر روی سیگنال‌های ضعیف‌تر اثر می‌گذارد اما بهره‌ی دینامیکی کلی این‌گونه درک می‌شود که گویا سیگنال‌های قوی‌تر بلندتر شده‌اند و ضعیف‌ترها آهسته‌تر.



شکل 3-8



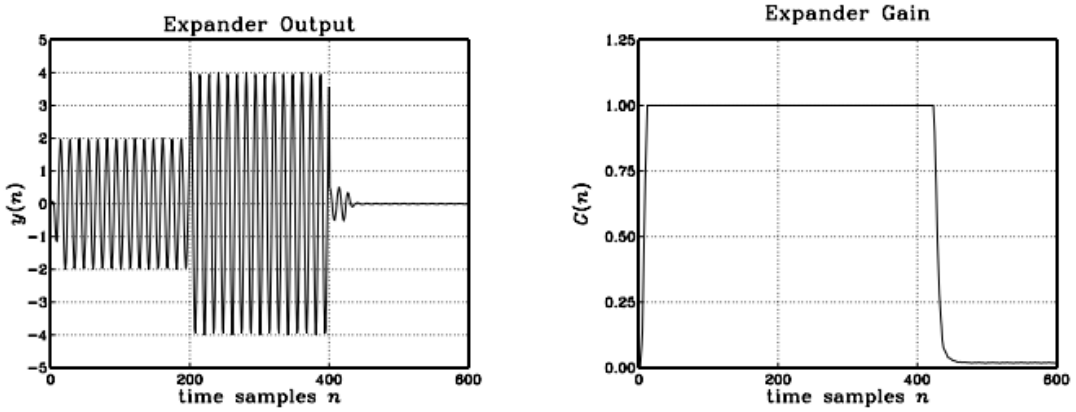
شکل 3-9



شکل 3-10

در نهایت شکل 3-11 یک مثال از گیت نویز را نشان می‌دهد که به عنوان یک منبسط کننده با نسبت انبساط 10:1، $\rho = 10$ پیاده سازی شده است و همان آستانه‌ی شکل 3-10 را دراد و ضرورتا سینوسی A_3 را حذف می‌کند که می‌تواند مربوط به نویز ناخواسته باشد.

امکان بهبود الگوریتم 3-24 برای دست‌یابی به ثابت زمانی‌های مختلف حمله و آزاد سازی وجود دارد. به هر حال ما در این آزمایش به همین الگوریتم‌ها اکتفا می‌کنیم.



شکل 3-11

3-3-الف)یک سینوسی $x(t) = A\cos(2\pi ft)$ را در نظر بگیرید. نشان دهید که مقدار موثر آن روی یک دوره‌ی تناوب و میانگین قدرمطلق آن به ترتیب به صورت زیر است:

$$\overline{x^2(t)} = \frac{1}{2}A^2 \qquad \overline{|x(t)|} = \frac{2}{\pi}A \qquad (24-3)$$

بنابراین مقادیر قله‌ی $[A_1\ A_2\ A_3] = [2\ 4\ 0.5]$ هر سه جزء سینوسی به مقادیر میانگین قدرمطلق خود نظیر می‌شوند که با آستانه‌ی فشرده‌ساز مفروض مقایسه خواهند شد:

$$[A_1A_2A_3] = [2\ 4\ 0.5] \Rightarrow \frac{2}{\pi}[A_1A_2A_3] = [1.27\ 2.55\ 0.32] \qquad (25-3)$$

3-3-ب)برنامه‌هایی با متلب بنویسید که تمام نمودارهای شکل‌های 3-6 تا 3-11 را تولید کنند.

3-3-ج)نمودارهای فشرده‌ساز شکل‌های 3-6 تا 3-7 را این بار با $\rho = 1/4$ و سایر پارامترهای قبلی انجام دهید.

3-3-د)نمودارهای فشرده‌ساز شکل‌های 3-6 تا 3-7 را این‌بار با $c_0 = 1.3$، $\rho = 1/2$ و $\lambda = 0.9$ توضیح دهید که چرا فقط قسمت میانی فشرده می‌شود.

آزمایش چهارم:آشنایی با پردازش تصویر

مقدمه

پردازش تصاویر امروزه بیشتر به موضوع پردازش تصویر دیجیتال گفته می‌شود که شاخه‌ای از دانش رایانه است که با پردازش سیگنال دیجیتال که نماینده تصاویر گرفته شده با دوربین دیجیتال یا پویش شده توسط پویشگرهستند سر و کار دارد.

پردازش تصویر دارای دو شاخه عمده‌ی بهبود تصاویر و بینایی ماشین است. بهبود تصاویر دربرگیرنده‌ی روش‌هایی چون استفاده از فیلترهای محوکننده و افزایش کنتراست برای بهتر کردن کیفیت دیداری تصاویر و اطمینان از نمایش درست آن‌ها در محیط مقصد(مانند چاپگر یا نمایشگر رایانه) است، در حالی که بینایی ماشین به روش‌هایی می‌پردازد که به کمک آن‌ها می‌توان معنی و محتوای تصاویر را درک کرد تا از آن‌ها در کارهایی چون رباتیک استفاده شود.

در معنای خاص آن پردازش تصویر عبارت است از هر نوع پردازش سیگنال که ورودی یک تصویر است مثل عکس یا صحنه‌ای از یک فیلم. خروجی پردازشگر تصویر می‌تواند یک تصویر یا مجموعه‌ای از نشان‌های ویژه یا متغیرهای مربوط به تصویر باشد. اغلب تکنیک‌های پردازش تصویر شامل برخورد با تصویر به عنوان یک سیگنال دو بعدی و به کار بستن تکنیک‌های استاندارد پردازش سیگنال روی آن‌ها می‌شود. پردازش تصویر اغلب به پردازش دیجیتالی تصویر اشاره می‌کند ولی پردازش نوری و آنالوگ تصویر هم وجود دارند. در این آزمایش با مفاهیم ابتدایی تصویر دیجیتال و هم‌چنین کدهای مقدماتی متلب در پردازش تصویر آشنا خواهید شد.

مراحل آزمایش

1- مبانی تصاویر دیجیتال

4-1-الف) تصویر با نام 'lena.bmp' را با استفاده از دستور imread خوانده، درون یک متغیر بریزید و سپس با استفاده از دستور imshow آن را نمایش دهید.

4-1-ب) دو نوع رایج تصاویر در MATLAB، uint8 (اعداد صحیح 8 بیتی بین 0 تا 255) و double (اعداد حقیقی بین 0 تا 1) هستند. به صورت پیش فرض تصاویر با فرمت uint8 خوانده می‌شوند اما به دلیل نبود مشکل

گرد شدن در هنگام ضرب و تقسیم، اجرای محاسبات بر روی تصاویر double ساده‌تر است. با استفاده از دستور im2double تصویر حاصل شده را به نوع double تبدیل کنید.

4-1-ج) با مطالعه راهنمای متلب برای دستور imhist، هیستوگرام شدت نور تصویر مذکور را رسم کنید.

4-1-د) با مطالعه راهنمای متلب برای دستور histeq، روش یکسانسازی هیستوگرام در تصویر را شرح دهید. به چه دلیل و در کجا از این روش استفاده می‌شود؟ این عمل را بر روی تصویر lena اعمال کنید و تصاویر ورودی و خروجی را مقایسه کنید.

4-1-ه) در بخش قبلی هیستوگرام تصویر حاصل را رسم کنید. چرا هیستوگرام بدست آمده به صورت یکنواخت در نیامده است؟ دلیل آن را شرح دهید.

نویززدایی تصویر(image denoising)

در این بخش با دو فیلتر دیجیتال مهم یعنی میانگین (mean) و میانه (median) آشنا می‌شوید. این فیلترها برای کاهش انواع خاصی از نویز در تصویر استفاده می‌شوند. ایده فیلتر میانگین عوض کردن مقدار هر پیکسل تصویر با میانگین مقادیر همسایگی شامل خود آن است. تاثیر این کار حذف مقادیر پیکسل‌هایی است که در اطراف وجود ندارند. فیلترینگ میانگین معمولا به عنوان فیلتر کانولوشن در نظر گرفته می‌شود. مانند دیگر کانولوشن‌ها این فیلتر بر مبنای یک هسته (kernel) می‌باشد که شکل و سایز همسایگی را هنگام محاسبه میانگین انتخاب می‌شود را مشخص می‌کند. همان طور که در شکل زیر نشان داده شده‌است معمولا از یک هسته مربعی 3x3 استفاده می‌شود، اگرچه در هموار سازی شدید هسته‌های بزرگ‌تر (مثلا 5x5) نیز استفاده می‌شوند.

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

4-2-الف) تصویر با نام 'Image02.jpg' را بارگذاری کنید.

4-2-ب) یک نویز گوسی با میانگین صفر و انحراف استاندارد (σ) برابر 0.2 را به آن اضافه کنید. می‌توانید از دستور متلب imnoise استفاده کنید. نتیجه را نمایش دهید.

4-2-ج) فیلتر میانگین 3x3 را اعمال کرده و خروجی را نمایش دهید. در مورد نتیجه و پارانترهای نویز گوسی حاضر در تصویر بحث کنید.

4-2-د) فیلتر میانگین 5x5 را روی آن اعمال کنید و تصویر منتجه را نمایش دهید. تفاوت بین این قسمت و قسمت قبل چیست؟ آیا یک بده‌بستان دارید؟ اگر پاسخ مثبت است، آن چیست؟

4-2-ه) تصویر اصلی را در نظر بگیرید. این بار یک نویز نمک و فلفل (salt and pepper) با p=10% با استفاده از دستور imnoise به آن اضافه کنید و تصویر خراب را نمایش دهید.

4-2-و) یک فیلتر میانگین 3x3 به این تصویر خراب اعمال کرده و نتیجه را نمایش دهید. آیا این فیلتر برای کاهش نویز نمک و فلفل مفید است؟ چرا؟

4-2-ز) با استفاده از fdatool یک فیلتر پایین گذر FIR طراحی کنید که فرکانس عبور آن حدود فرکانس نرمالیزه 0.5 باشد. سپس با استفاده از دستور ftrans2 فیلتر یک بعدی طراحی شده را به یک فیلتر دو بعدی تبدیل کنید. سپس با استفاده از دو دستور freqz و freqz2 پاسخ فوریه فیلترهای یک بعدی و دو بعدی طراحی شده را با یکدیگر مقایسه کنید.

4-2-ح) فیلتر طراحی شده در بخش قبلی را به دو تصویر حاوی نویز اعمال کنید. نتایج حاصل از این فیلتر را با فیلتر میانگین مقایسه کنید. علت این تفاوت چیست؟

همان‌طور که در بخش قبل دیدید فیلترینگ میانگین در کاهش نویز نمک و فلفل فایده‌ای ندارد، بنابراین ما از یک فیلتر دیگر استفاده می‌کنیم، فیلتر میانه. این فیلتر مانند فیلتر میانگین هر پیکسل تصویر را به نوبت در نظر می‌گیرد و به همسایگی اطراف آن نگاه می‌کند تا تصمیم بگیرد که آیا آن نماینده اطرافش هست یا نه. به جای تعویض مقدار پیکسل با میانگین مقادیر پیکسل‌های همسایگی، آن را با مقدار میانه عوض می‌کند. میانه با ابتدا مرتب کردن عددی همه‌ی مقادیر پیکسل‌های همسایگی اطراف و سپس تعویض پیکسل مورد نظر با مقدار پیکسل میانی محاسبه می‌شود (در صورتی که همسایگی مورد نظر تعداد زوجی پیکسل داشته باشد، از میانگین دو عدد میانی استفاده می‌شود).شکل زیر یک محاسبه‌ی نمونه را نشان می‌دهد:

123	125	126	130	140
122	124	126	127	135
118	120	150	125	134
119	115	119	123	133
111	116	110	120	130

مقادیر همسایگی:

115، 119، 120، 123، 124،
125، 126، 127، 150

مقدار میانه: 124

4-2-ط) تابعی برای پیاده‌سازی میانه بنویسید. تابع شما باید تصویر اصلی و سایز پنجره که طول و عرض آن فرد می‌باشد را به عنوان ورودی بگیرد و تصویر اصلاح‌شده را به عنوان خروجی بدهد و هم‌چنین در صورتی که طول و یا عرض وارد شده عدد فردی نبود پیغام خطا ظاهر شود.

4-2-ک) فیلتر خود را با سایز 3 به تصویر نویزی قسمت 4-2-ه اعمال کنید. نتیجه را نمایش دهید. چگونه این فیلتر می‌تواند به شما کمک کند تا بر نویز نمک و فلفل غلبه کنید؟

4-2-ل) اثر bluring فیلترهای میانه و میانگین را مقایسه کنید. آیا فیلتر میانه بدی‌هایی نیز دارد؟

تبدیل موجک دوبعدی

در این بخش شما استفاده از تبدیل موجک دوبعدی را تمرین خواهید کرد.

4-3-الف) 'Image03.png' را بارگذاری کرده و با اســتفاده از 'db1' به عنوان نام موجک، تبدیل موجک را به این تصــویر اعمال کنید و ســپس تصــاویر تقریبی و جزئیات را نمایش دهید. می‌توانید از تابع متلب dwt2 که تبدیل موجک گسسته دوبعدیی می‌باشد استفاده کنید. ورودی‌ها و خروجی‌های این تابع چه می‌باشند؟ هر زیر تصویر حاوی چه اطلاعاتی است؟

4-3-ب) اکنون در نظر بگیرید که ما می‌خواهیم خطوط افقی را highlight کنیم. شما چه پیشنهادی دارید؟ آن را اعمال کنید.

تارشدگی(Motion Bluring)

یکی از مشکلات عکاسی تار شدن تصویر است. این زمانی اتفاق می‌افتد که جسم در زمان باز بودن شاتر دوربین دیجیتال حرکت کند. ما می‌توانیم با علم به الگوی جسم آن را اصلاح کنیم.

4-4-الف) 'Image04.png' را بارگذاری کرده و نمایش دهید. با اســـتفاده از توابع داخلی متلب یک تاری (motion blur) به اندازه 15 و با زاویه 20° به تصویر اعمال کنید. تصویر تارشده را نمایش دهید.

4-4-ب) تصویر تار شده را با فیلتر wiener اصلاح کنید. شما می‌توانید از تابع deconvwnr استفاده کنید. برای این تابع به Help متلب مراجعه کنید.شما باید پارامتر NSR (Noise to Signal Power Ratio) را تغییر دهید تا به یک نتیجه منطقی برسید. ابتدا NSR را صفر بگیرید و حاصل را نمایش دهید، سپس با تکرار، مقادیر مناسبی برای NSR انتخاب کنید.

4-4-ج) اکنون یک نویز گوسی با میانگین صفر و واریانس 10 به تصویر تار قسمت 4-4-الف اضافه کنید. تصویر تار نویزی را نمایش دهید.

4-4-د) قسمت 4-4-ب را برای تصویر قسمت 4-4-ج تکرار نمایید.

فیلتر Antialiasing

4-5-الف) فایل با نام 'glass.tif' را بارگذاری کرده و نمایش دهید.

4-5-ب) DFT دوبعدی تصویر را با تابع fft2 حساب کنید. فاز و لگاریتم دامنه DFT دوبعدی را در تصاویر مجزا نمایش دهید. از دستور fftshift استفاده کرده و توضیح دهید که این دستور چه کاری انجام می‌دهد.

توجه: فوریه دوبعدی به صورت زیر تعریف می‌شود:

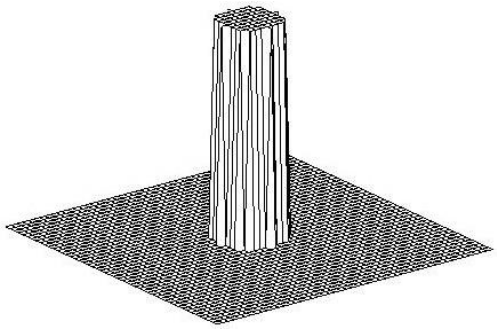
$$F(u,v) = \sum_{m}\sum_{n} f(m,n)e^{j2\pi(mv+nu)}$$

4-5-ج) تابع زیر را که فیلتر پایین‌گذر دوبعدی را پیاده‌سازی می‌کند در نظر بگیرید، تعریف کنید.

Output_image = FFT_LP_2D(input_image, cutoff_frequency)

فرکانس قطع باید در بازه‌ی $[0\ \pi]$ باشد. Ifft2 را اعمال کرده تا معکوس DFT دوبعدی را حساب کنید.

فیلتر باید به فرم دایروی (Circular) باشد. به عبارت دیگر شعاع دایره باید فرکانس قطع باشد.



به شما توصیه می‌شود که از توابع fftshift، ifftshift و meshgrid استفاده کنید. هم‌چنین فراموش نکنید که از گزینه 'symmetric' در ifft استفاده کنید. هم‌چنین به بازه‌ای که DFT دوبعدی در آن حساب می‌شود توجه داشته باشید.

4-5-د) فیلتر پایین‌گذر با فرکانس قطع 0.1π را روی تصویر قسمت 4-5-الف اعمال کرده و نتیجه را نمایش دهید. بر اساس آنچه تا کنون در مورد فیلترهای پایین‌گذر و بالاگذر آموخته‌اید، در مورد عملکرد این فیلتر بحث کنید.

4-5-ه) از تابع downsample به درستی استفاده کنید و ابعاد این تصویر را به 1/4 تصویر اصلی کاهش دهید. خروجی را نمایش دهید و تاثیر aliasing را در تصویر downsampled مشاهده کنید. aliasing گوشه‌های تصویر یا قسمت‌های راه راه را خراب می‌کند. بر اساس دانش خودتان از پردازش سیگنال یک‌بعدی، از فیلتری که تعریف کرده‌اید استفاده کنید و تصویر را به اندازه کافی تغییر دهید تا اثرات aliasing را در تصویر 'downsampled' حذف کنید. خروجی را نمایش دهید.

آزمایش پنجم

نرم‌افزار ISE و آموزش کار با آن و ساخت پروژه

FPGA چیست؟

FPGA از تعداد بسیار زیادی گیت منطقی در داخل خود ساخته می‌شود. این آی سی نسبت به عدم نویز پذیری بسیار مقاوم‌تر از انواع میکروکنترلرها است. ویژگی مثبت FPGAها نسبت به تراشه‌های میکروپروسسورها و میکروکنترلرها و DSPها این است که می‌تواند به صورت موازی دستورات را اجرا کند. هم‌چنین توسط این آی سی می‌توان هر نوع میکروکنترلر و میکروپروسسوری را پیاده‌سازی نمود. به دلیل پردازش موازی این آی سی در صنایع مخابرات جهت سوییچ‌های مخابراتی و لایه‌های شبکه استفاده می شود. هم‌چنین در صنایع نظامی جهت کنترل موشک استفاده می‌شود. این آی سی می‌تواند به عنوان پردازش سیگنال و تصویر نیز استفاده شود که این ویژگی توسط نرم افزار MATLAB پشتیبانی می شود. با استفاده از این آی سی می‌توان تمامی مدارات منطقی را پیاده نمود. ویژگی منفی این آی سی‌ها این است که قیمت آنها نسبت به میکروکنترلرها بالاتر است و برنامه‌نویسی آن نیز از میکروکنترلر بسیار مشکل‌تر می‌باشد.

زبان برنامه نویسی FPGA چیست؟

زبان‌های برنامه نویسی مانند C، Basic و اسمبلی به صورت خط به خط دستورات را اجرا می‌کنند. این زبان‌ها قدرت کافی جهت پردازش موازی و حفظ حالات قبلی مدار را ندارند. در نتیجه زبان‌های توصیف سخت افزار HDL ساخته شد. مشهور ترین این زبان‌ها عبارتند از:

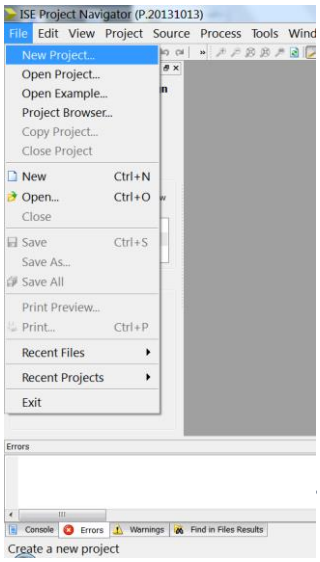
System C: این زبان شباهت بسیار زیادی به زبان C دارد.

System Verilog: این زبان نیز شباهت‌هایی به زبان C دارد. این زبان توسط شرکت Cadence گسترش پیدا کرد و کمی بعد از زبان VHDL استاندارد IEEE شد.

VHDL: این زبان یکی از مشهورترین زبان‌های توصیف سخت افزار می‌باشد که در وزارت دفاع آمریکا با هم‌کاری شرکت‌های Texas Instrument، IBM و Intermetrics ساخته شد و اولین زبان توصیف سخت افزار بسیار قوی می‌باشد. امروزه مشهورترین زبان‌های توصیف سخت‌افزار زبان VHDL و Verilog می‌باشد.

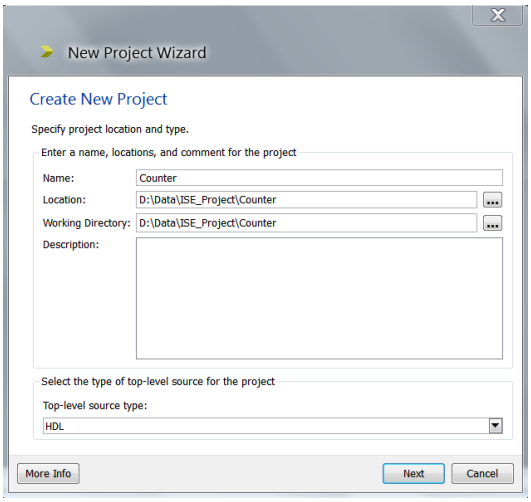
چگونه از نرم افزار ISE استفاده کنیم؟

نخست نرم افزار ISE را باز نمایید. و سپس وارد منوی File شده و New Project را انتخاب کنید. (شکل 1)



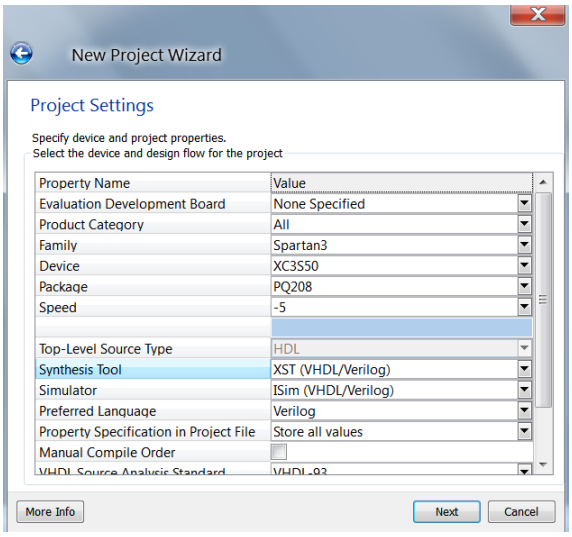
شکل 1

در پنجره باز شده مسیری را برای پوشه برنامه انتخاب کنید و نامی برای پروژه در نظر بگیرید و حالت توصیف سخت افزاری را برای برنامه نویسی به زبان VHDL انتخاب کنید. (شکل 2)



شکل 2

برای انتخاب تراشه مورد استفاده، محیط شبیه سازی و زبان برنامه نویسی به صورت زیر عمل کنید (شکل 3):



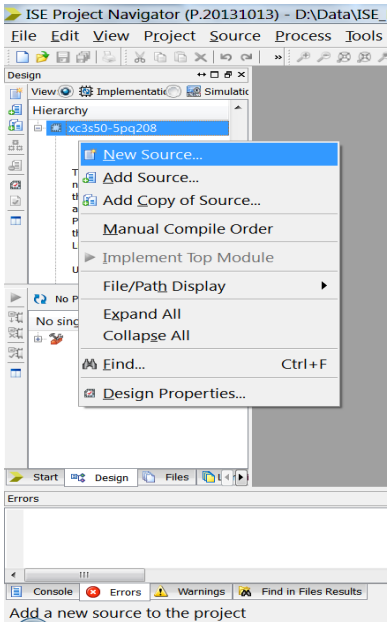
شکل 3

در نهایت، در پنجره آخر finish را انتخاب کنید.

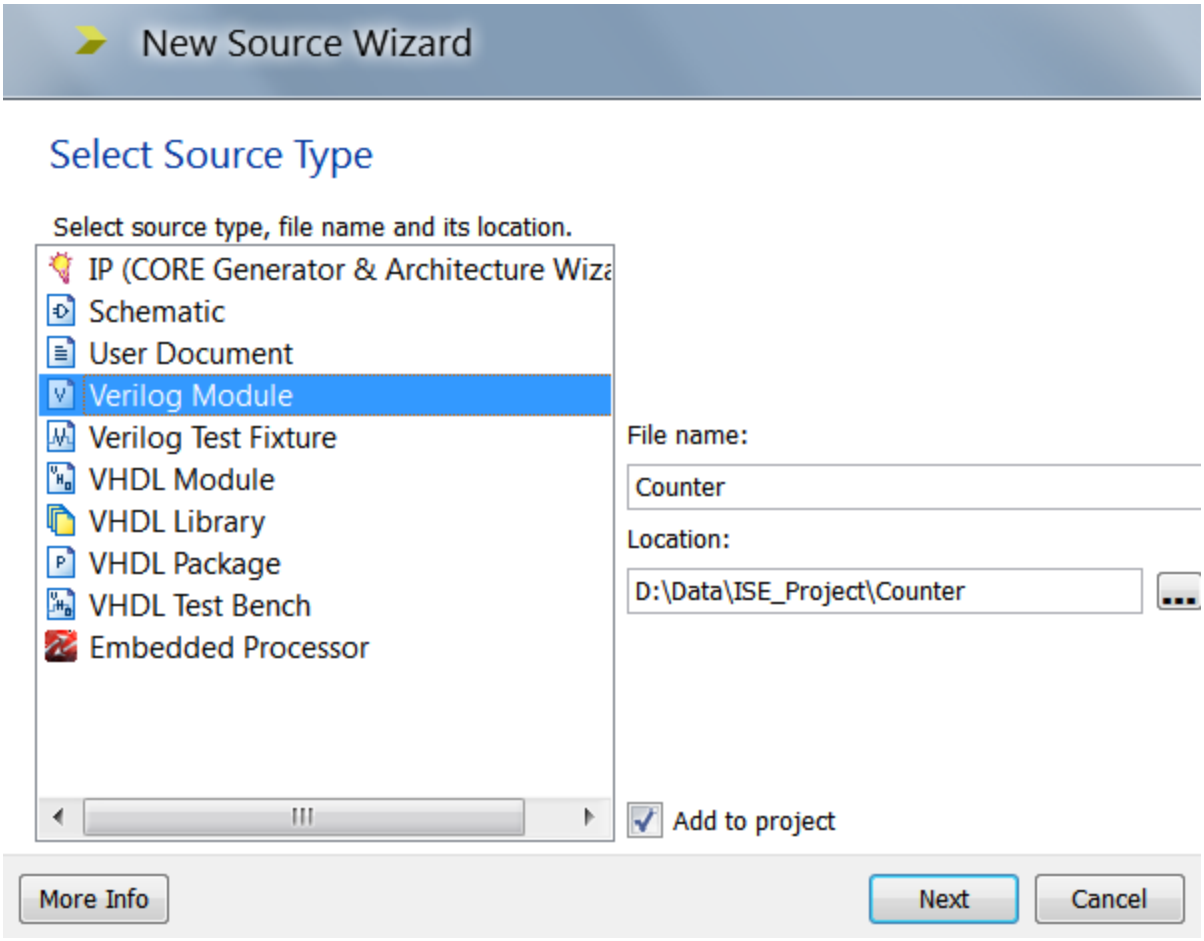
برای نوشتن برنامه نیازمند ایجاد یک فایل برای نوشتن کدهای خود در آن هستیم. برای ایجاد یک فایل جدید به صورت زیر عمل کنید:

روی اسم تراشه خود در سمت چپ پنجره کلیک راست کنید و New source را انتخاب کنبد. (شکل 4)

در پنجره باز شده دقت کنید که ابتدا Verilog Module انتخاب شده باشد و سپس برای فایل خود یک نام انتخاب کنید. (شکل 5)

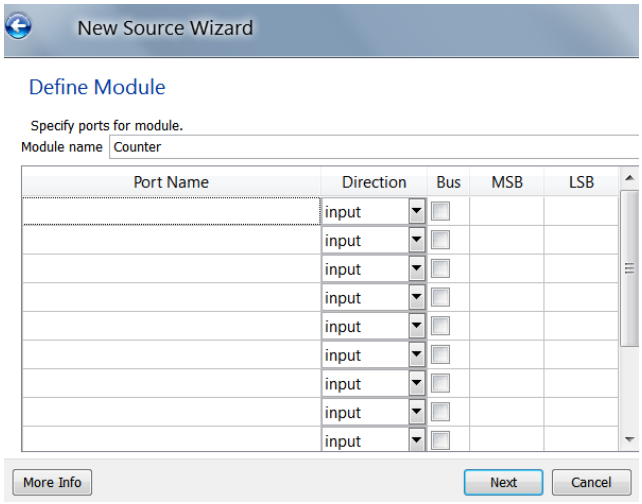


شکل 4



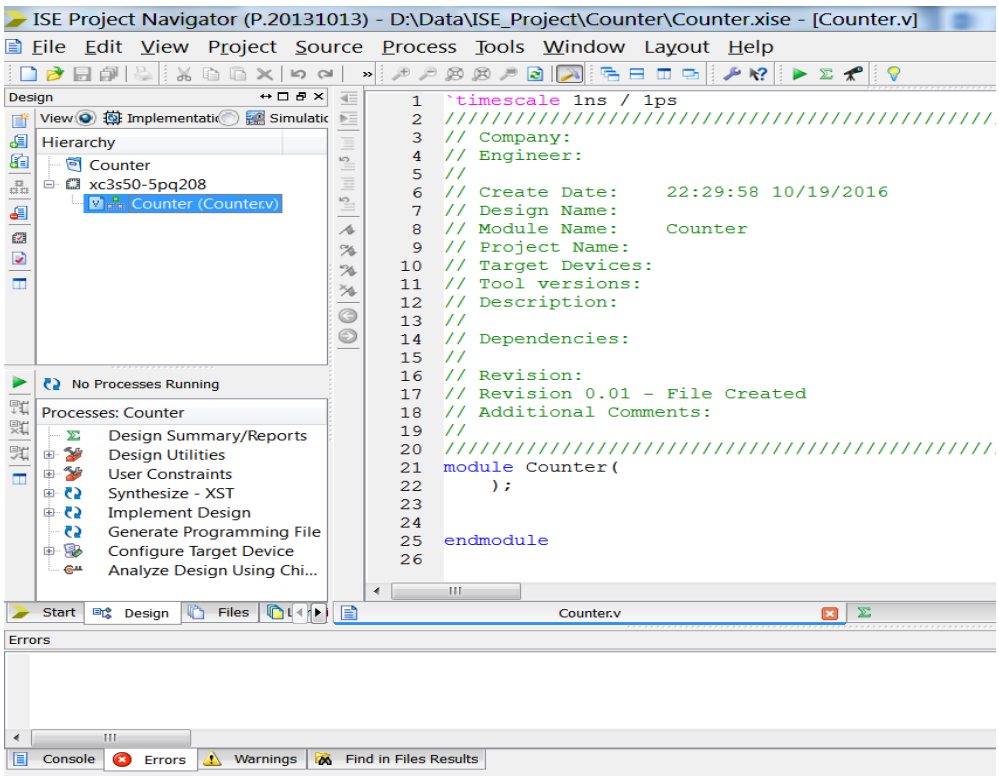
شکل 5

در پنجره بعدی شما باید ورودی و خروجی برنامه خود را تعیین کنید که این کار را در حین برنامه نویسی نیز می‌توان انجام داد و نیازی به پر کردن این پنجره نیست و تنها روی next کلیک کنید. (شکل 6)



شکل 6

و در انتها finish را انتخاب کنید تا فایل مورد نظر به شکل زیر ایجاد شود. (شکل 7)

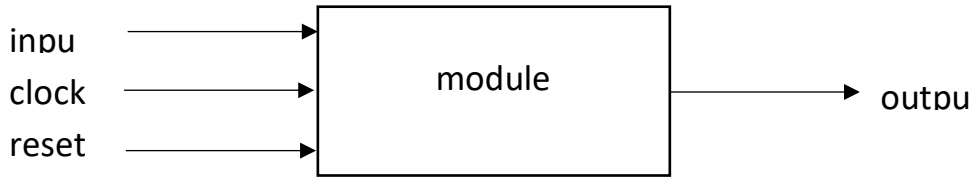


شکل 7

حال باید به معرفی ورودی و خروجی‌های ماژول خود بپردازیم.

یک ماژول ساده به طور معمول دارای ورودی‌هایی به شرح زیر است:

1. سیگنال های ورودی و داده های خام
2. سیگنال reset: تمام حافظه‌ها را به حالت اولیه برمیگرداند.
3. سیگنال clock: شبیه به قلب تپنده‌ی ماژول عمل می‌کند و با هر بار تپش آن، یک سری دستور خاص انجام می‌شود.



شکل 8

به عنوان مثال برای یک ماژول شمارنده، ما نیازمند ورودی‌های زیر هستیم:

- سیگنال clock
- سیگنال reset

خروجی نیز نشان دهنده عدد شمارنده خواهد بود.

برنامه اصلی نیز به این صورت خواهد بود که با هر بار آمدن سیگنال clock، حافظه‌ای که مقدار شمارنده را دارد، یک واحد افزایش می‌یابد.

با توجه به این توضیحات، نحوه تعریف ورودی و خروجی را بصورت زیر بیان میکنیم:

ابتدا اسم خروجی و ورودی‌ها در جلوی نام ماژول و داخل پرانتز می‌نویسیم. بهتر است که ابتدا خروجی‌ها و بعد از آن ورودی‌ها نوشته شوند.

سپس در داخل ماژول باید نوع متغیرها را تعریف کنیم.

```
module Counter(count, clock, reset
    );
    output reg [7:0] count;
    input clock;
    input reset;


endmodule
```

در ادامه باید کد مربوط به شمارش و برنامه اصلی را بنوسیم

برای این کار ابتدا یک بلوک always به صورت زیر قرار می‌دهیم. بلوک always بدین صورت عمل می‌کند که با هر با تحریک شدن پارمترهای داخل پرانتز آن، دستورات داخل بلوک یک بار انجام می شود.

```
always @ (      ) begin

end
```

ما می‌خواهیم که با هر بار آمدن لبه بالارونده سیگنال clock، یک واحد به شمارنده ما اضافه شود. در نتیجه، تحریک بلوک always، لبه بالارونده clock خواهد بود. هم‌چنین کد داخل بلوک باید افزایش شمارنده را ایجاد کند. بنابراین برنامه به صورت زیر خواهد بود:

```
module Counter(count, clock, reset
    );
    output reg [7:0] count = 0;
    input clock;
    input reset;

    always @ (posedge clock) begin

      if (reset == 1) begin
         count = 0;
      end

      else begin
         count = count + 1;
      end
    end

endmodule
```

این شمارنده از 0 تا 255 خواهد شمرد و دوباره با سر ریز شدن ریجیستر count دوباره از صفر شروع به شمارش می‌کند.

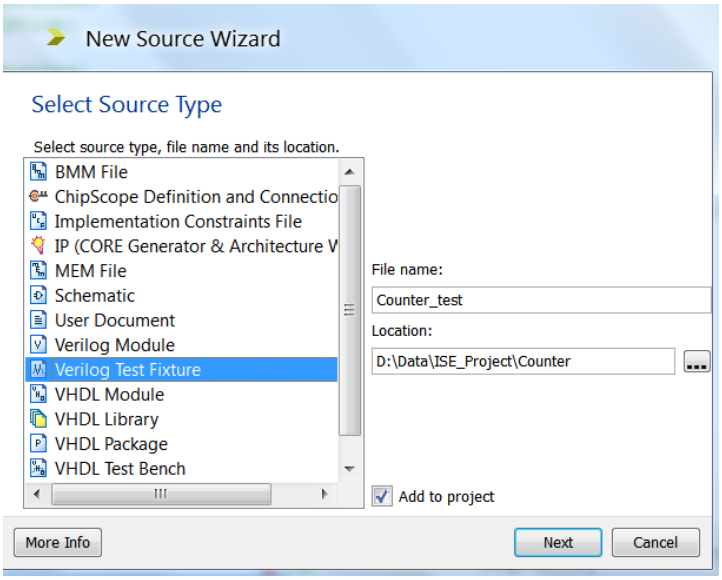
شبیه سازی ماژول (Simulation)

برای استفاده از ابزار شبیه سازی ISim به صورت زیر عمل می‌کنیم:

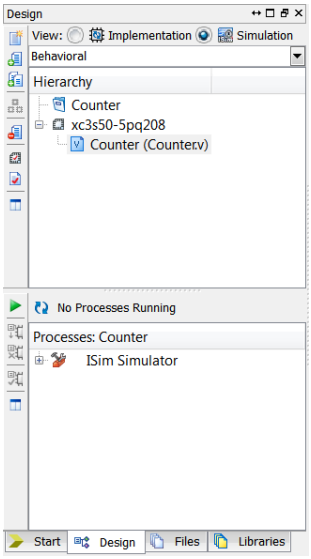
ابتدا از منوی سمت چپ گزینه Simulation را با توجه به شکل زیر انتخاب میکنیم. (شکل 9)

سپس روی ماژول خود کلیک راست کرده و New Source را انتخاب کنید.

در پنجره باز شده Verilog Test Fixture را انتخاب کرده و نامی برای ماژول تست خود انتخاب نمایید. (شکل 10)

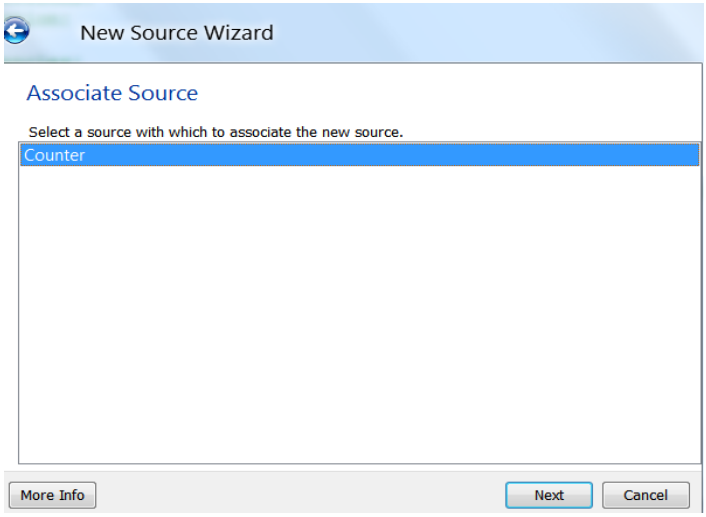


شکل 9



شکل 10

در مرحله بعد ماژولی را که می‌خواهید برای آن تست را بنویسید انتخاب کنید. (شکل 11)



شکل 11

و در پنجره بعدی finish را انتخاب کنید تا ماژول تست ایجاد شود.

همان طور که ملاحظه میکنید محتویات ماژول درست شده به شکل زیر است:

```
module Counter_test;

    // Inputs
    reg clock;
    reg reset;

    // Outputs
    wire [7:0] count;

    // Instantiate the Unit Under Test (UUT)
    Counter uut (
        .count(count),
        .clock(clock),
        .reset(reset)
    );

    initial begin
        // Initialize Inputs
        clock = 0;
        reset = 0;

        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here

    end

endmodule
```

در واقع در این ماژول، باید ورودی‌ها از جمله داده های خام، clock و reset را خودمان به صورت دستی مقدار دهی کنیم.

ایجاد clock در ماژول تست

کلاک یکی از مهم ترین ورودی های ماژول است و باید به درستی ایجاد شود. برای ایجاد یک سیگنال دائمی کلاک متناوب به صورت زیر عمل میکنیم:

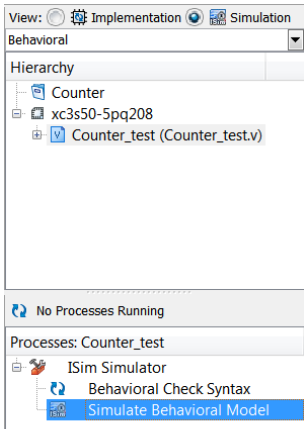
```
always #5 clock =~clock;
```

این دستور در هر 5 واحد زمانی (نانوثانیه) کلاک را not می‌کند.

برای مقدار دهی به ورودی‌ها نیز کد زیر را اضافه می‌کنیم:

```
initial begin
    // Initialize Inputs
    clock = 0;
    reset = 0;

    // Wait 100 ns for global reset to finish
    #100;

    // Add stimulus here
    reset = 1;

    #30;
    reset = 0;

end
```

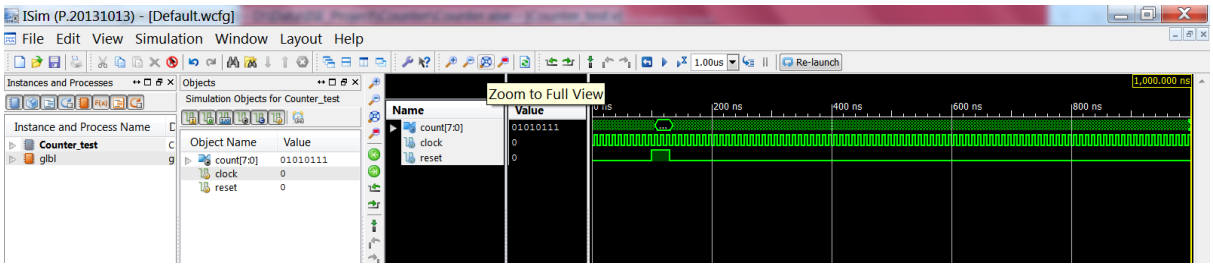
سپس مطابق شکل در قسمت سمت چپ دقت می‌کنیم که ماژول تست انتخاب شده باشد و سپس روی Simulate Behavioral Model دوبار کلیک می‌کنیم تا محیط شبیه سازی باز شود. (شکل 12)



شکل 12

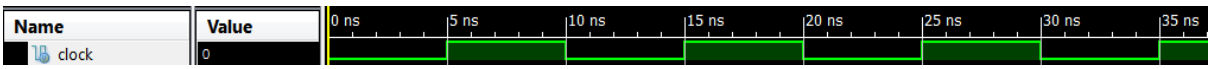
در پنجره باز شده روی Zoom to full view کلیک می‌کنیم تا تمام سیگنال را در بازه زمانی مورد نظر مشاهده کنیم.

به شکل زیر:



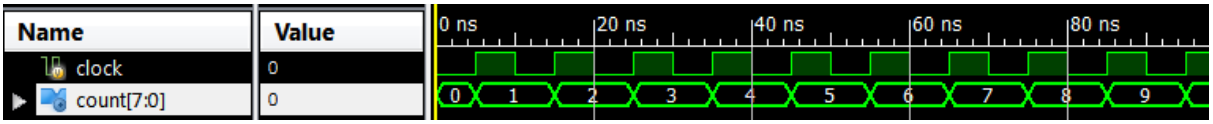
شکل 13

همان طور که مشاهده میکنیم، سیگنال کلاک به شکل زیر در هر 5 واحد زمانی not می‌شود.



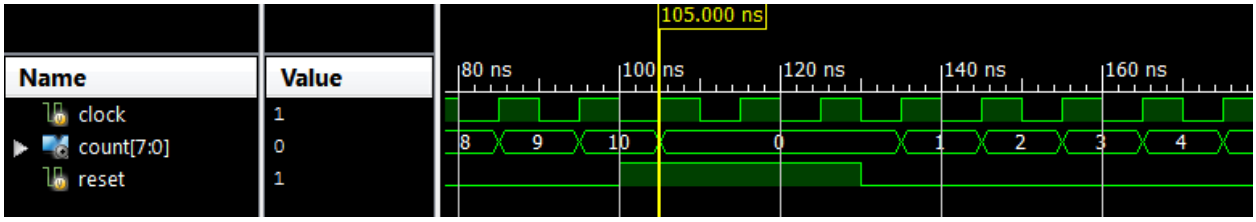
شکل 14

هم‌چنین با توجه به شکل در هر بار لبه بالارونده clock، یک واحد به count اضافه می‌شود.



شکل 15

هم‌چنین با 1 شدن سیگنال reset در لحظه‌ی 100ام و آمدن لبه بالارونده کلاک در زمان 105ام، مقدار count به 0 تغییر می‌کند و تا زمانی که سیگنال reset در مقدار 1 باقی می‌ماند (لحظه 130ام)، مقدار شمارنده 0 باقی می‌ماند. با 0 شدن سیگنال reset در لحظه 130ام و آمدن لبه بالارونده clock در لحظه 135ام، دوباره شمارنده شروع به شمارش از ابتدا می‌کند.



شکل 16

بنابراین کد نهایی ماژول تست به صورت زیر خواهد بود:

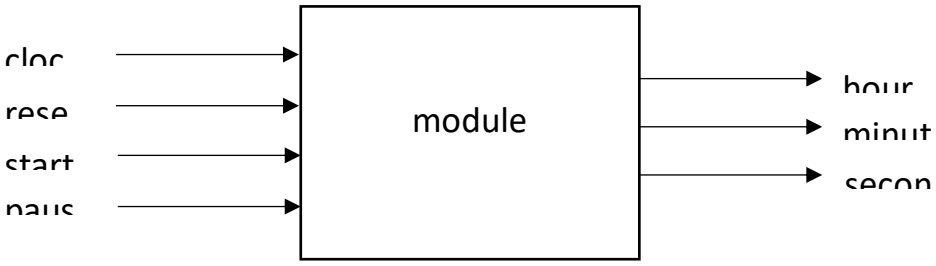
```
`timescale 1ns / 1ps
```

```
24
25  module Counter_test;
26
27     // Inputs
28     reg clock;
29     reg reset;
30
31     // Outputs
32     wire [7:0] count;
33
34     // Instantiate the Unit Under Test (UUT)
35     Counter uut (
36        .count(count),
37        .clock(clock),
38        .reset(reset)
39     );
40
41     always #5 clock =~clock;
42
43     initial begin
44        // Initialize Inputs
45        clock = 0;
46        reset = 0;
47
48        // Wait 100 ns for global reset to finish
49        #100;
50
51        // Add stimulus here
52        reset = 1;
53
54        #30;
55        reset = 0;
56
57     end
58
59  endmodule
60
```

تکلیف:

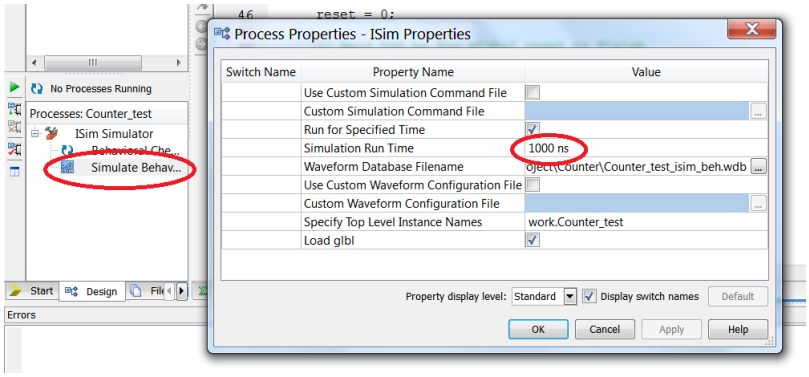
حال با ایجاد یک پروژه جدید، ماژولی به شرح زیر بنویسید و برای آن یک ماژول تست نیز ایجاد کنید:

در این ماژول می‌خوایم یک ساعت به صورت زیر طراحی کنیم:



شکل 17

- با 1 شدن سیگنال start، ساعت شروع به کار میکند.
- با 1 شدن سیگنال reset، ساعت صفر می‌شود.
- با 1 شدن سیگنال pause، ساعت زمان فعلی خود را حفظ می‌کند وثابت می‌ماند. با 0 شدن pause، ساعت شروع به ادامه شمارش می‌کند.
- به تغییر سیگنال‌های ورودی دقت کنید و شرط‌ها و یا سیگنال‌های ورودی را به صورتی در نظر بگیرید که ماژول وارد حالت‌های نامعلوم و یا وارد چند حالت به طور همزمان نشود.
- در نوشتن برنامه به تعداد بیت‌های لازم برای هر ریجیستر دقت کنید.
- در داخل بلوک always از ساختار شرطی if استفاده کنید.
- در شبیه سازی، سیگنال clock را طوری تغییر دهید که با سپری شدن هر 1 ثانیه، ساعت کار کند. دقت کنید، از آنجایی که واحد زمانی در حالت پیش فرض روی 1 ns (1 نانوثانیه) قرار داده شده، شما باید راه حلی برای ایجاد سیگنال clock با تناوب 1 s (1 ثانیه) پیشنهاد دهید.
- با وارد شدن به قسمت properties می‌توانید زمان شبیه سازی را افزایش دهید. (شکل 18)



شکل 18

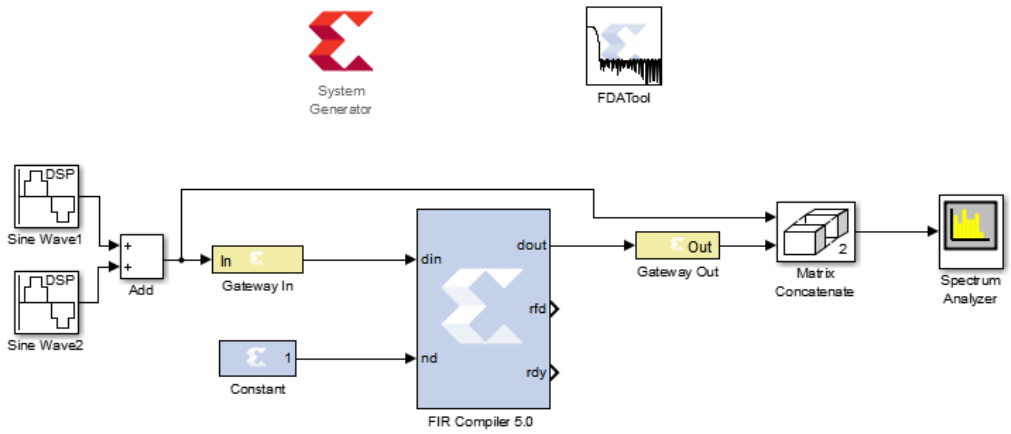
- **طراحی یک فیلتر پایین‌گذر در محیط Simulink با استفاده از System Generator**

وارد محیط MATLAB شوید و دستور simulink را وارد کنید.

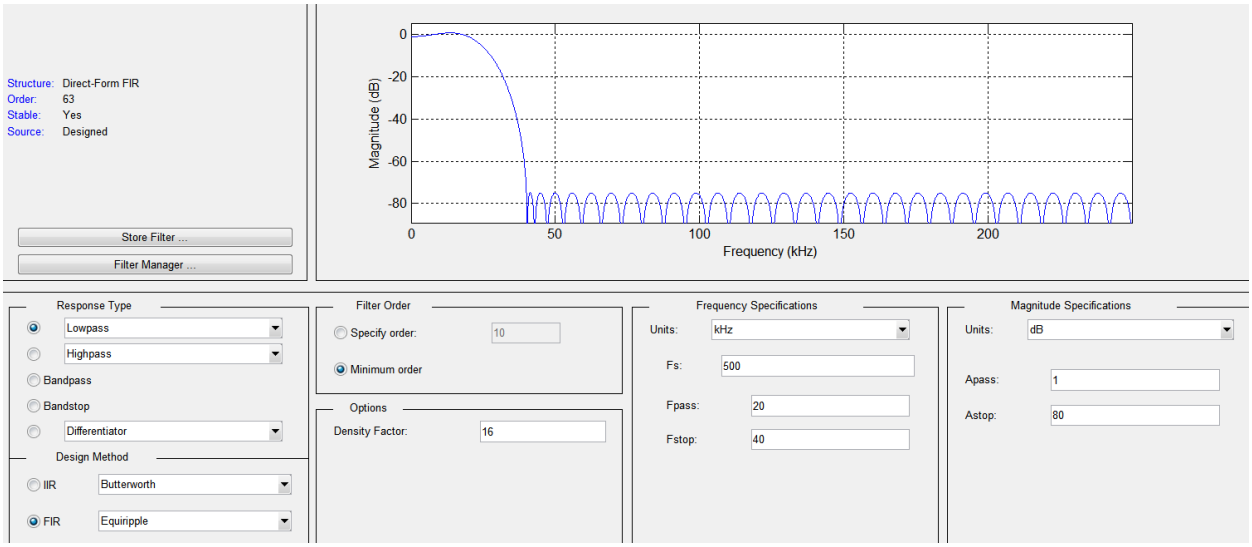
اندکی صبر کنید تا کتابخانه simulink باز شود.

از قسمت File>New>Model یک مدل جدید ایجاد کنید. سپس از کتابخانه simulink بلوک‌های مورد نیاز خود را وارد می‌کنیم:

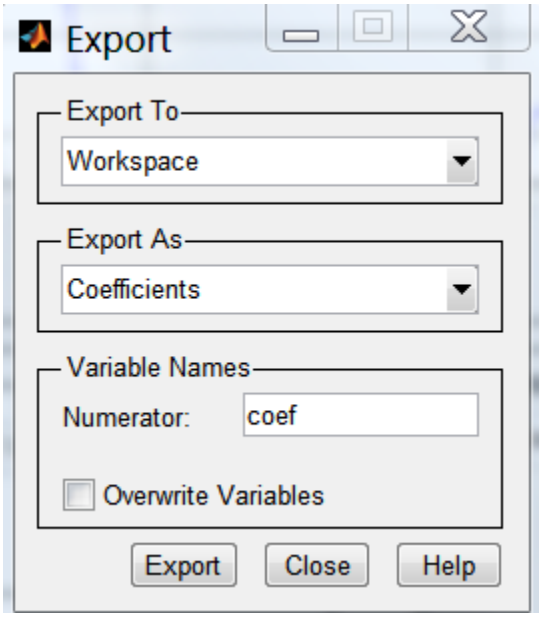
- System Generator
- FDATool
- FIR Compiler 5.0
- Sine Wave (DSP)
- Gateway In
- Gateway Out
- Constant
- Add
- Matrix Concatenate
- Spectrum Analyzer



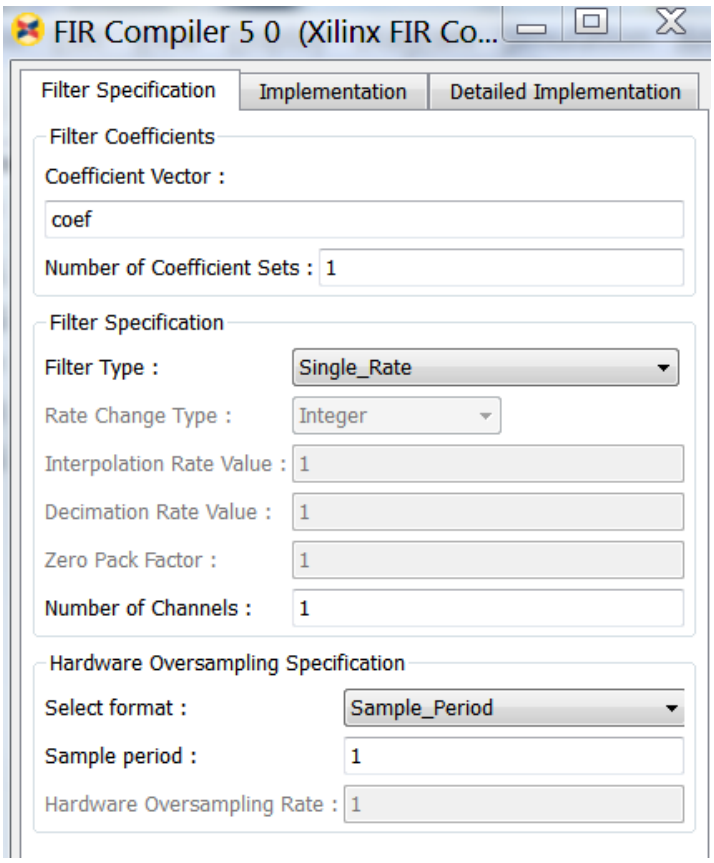
1) روی FDATool دوبار کلیک کنید تا صفحه طراحی فیلتر باز شود و فیلتری با مشخصات زیر طراحی کنید.

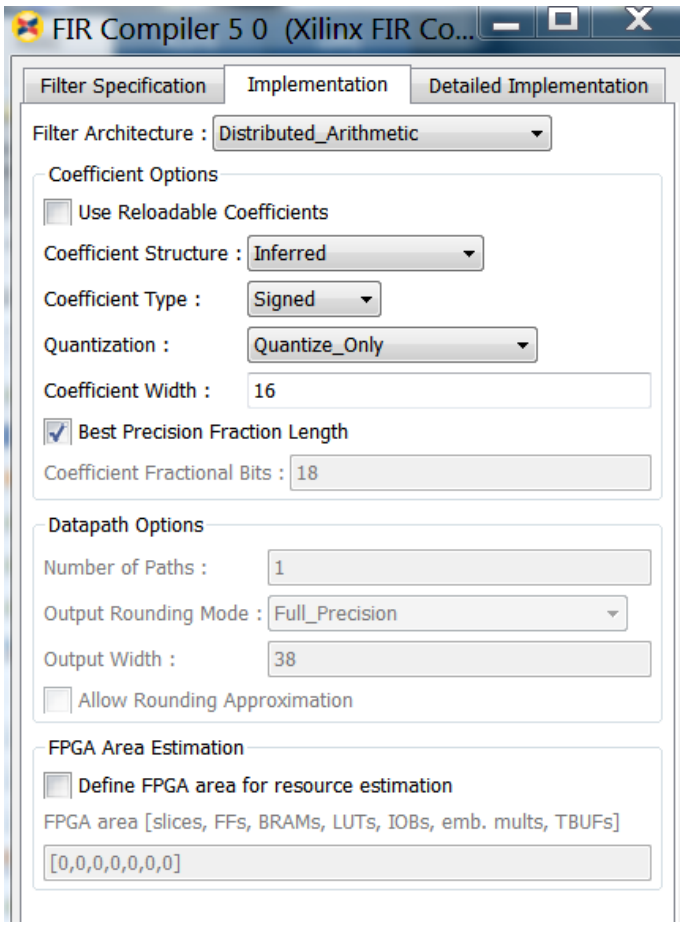


سپس در همین پنجره وارد مسیر File>Export شوید و نامی برای ضرایب فیلتر خود انتخاب کرده و روی Export کلیک کنید.

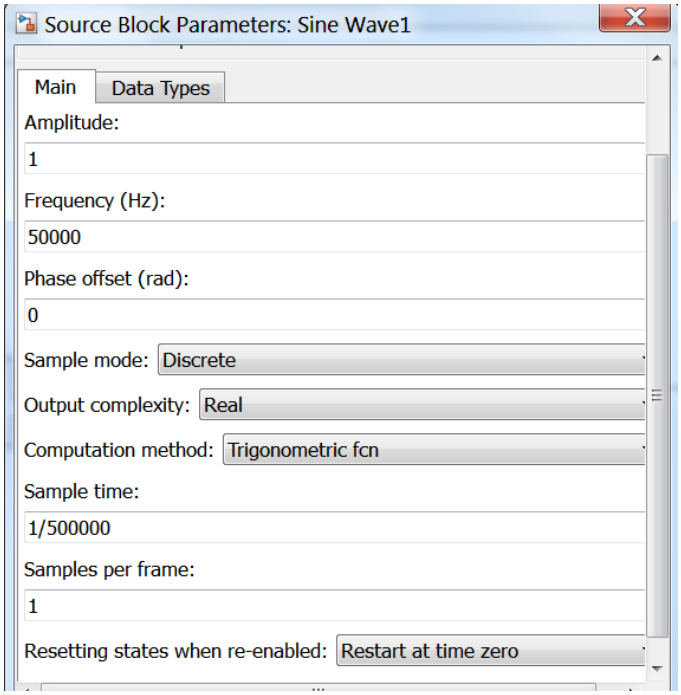


2) سپس با انتخاب FIR Compiler 5.0 تنظیمات آن را مطابق شکل زیر وارد می‌کنیم:

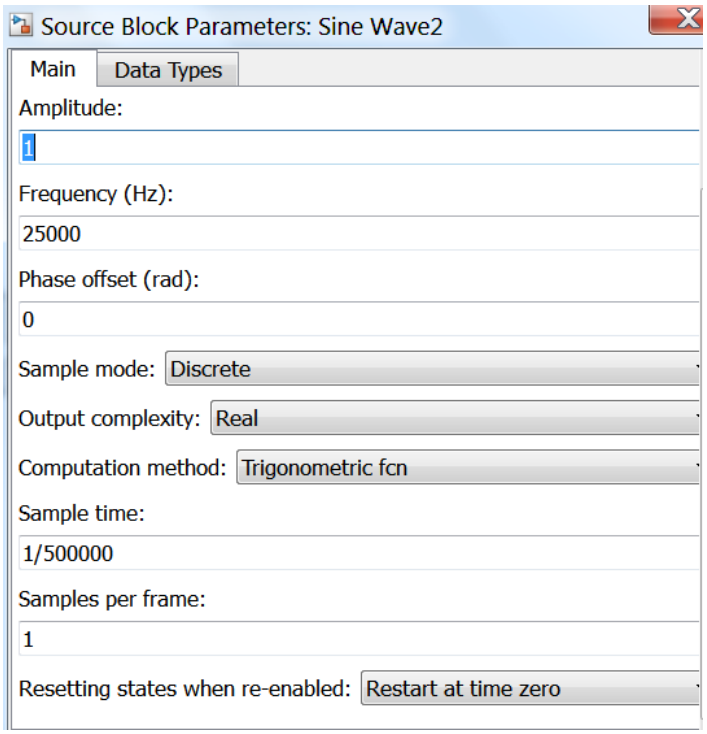




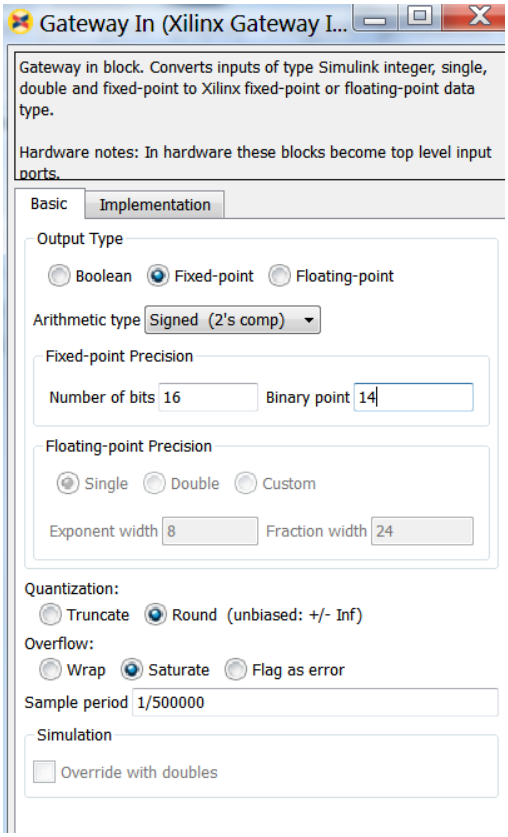
3) تنظیمات بلوک Sine Wave 1:



4) تنظیمات بلوک Sine Wave 2:

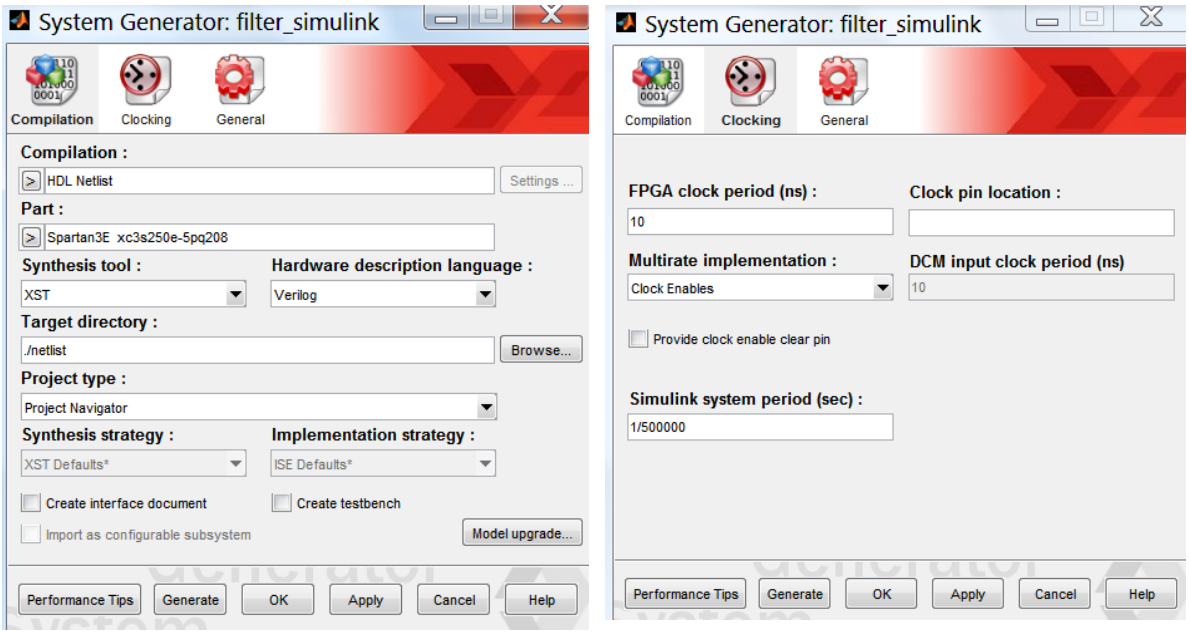


5) بلوک Gateway In:



6) بلوک Constant را نیز در حالت Boolean با اندازه 1 قرار دهید.

7) در بلوک System Generator نیز تنظیمات زیر را لحاظ کنید:



8) حال مدل خود را run کرده و نتیجه را در خروجی ملاحظه کنید: