

**دانشکده مهندسی برق و کامپیوتر**

**برنامه نویسی GPU**

**گزارش کار تمرین دوم**



**محمد مصطفایی 9432181**

**حسین قاسمی رامشه 9433590**

زمستان 97

# **مقدمه**

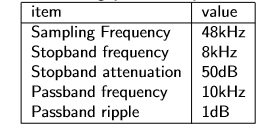
در مبحث کار با GPU ها و کدزنی روی آن ها یکی از چالش برانگیزترین و مهم ترین قسمت ها تقسیم قسمت های مختلف کار بین thread ها می باشد، به گونه ای که هم نتایج به دست آمده صحیح و قابل قبول باشد و هم performance بالایی نسبت به حالت سری دریافت نماییم. به جرعت می توان گفت اصل کار در پیاده سازی تقسیم صحیح block ها و grid ها بین SMها و البته مدیریت صحیح منابع مورد نیاز آنهاست. یکی از مسائل معروف و جالب توجه برای تمرین و کار با blockها و gridها و البته به طور کلی کار با GPU انجام پردازش بر روی عکس هاست، چرا که هم دیتای فراوان و بزرگی را در اختیار ما میگذارد که پردازش غیر موازی بر روی آنها به شدت هزینه بر است و هم فرم و قالب ماتریس گونه ی آنها ما را بطور کامل با چالش تقسیم کارها بین coreها و threadها دست به گریبان می کند و صد البته نتیجه کار آن به دلیل قالب عکس بودن آن به راحتی قابل مشاهده و جذاب و هم چنین پرکاربرد در بخش های مختلف است.

# **شرح مسئله**

مسئله ای که در این تمرین با آن مواجه هستیم مات کردن عکس می باشد به نحوی که مقدار هر کدام از پیکسل ها را با توجه به مقادیر پیکسل های اطرافش به وسیله ی تابعی به نحوی تغییر دهیم که نهایتا عکس وضوح خود را از دست داده و مات به نظر برسد

# **پیاده سازی Highpass Filter**

این فیلتر به گونه ای طراحی می شود که قابلیت عبور دادن فرکانس های بالا را دارد. و باقی فرکانس ها را عبور نمی دهد به منظور پیاده سازی آن ابتدا به کمک نرم افزار matlab و قسمت fdatool ضرایب فیلتر را خواهیم ساخت. مقادیر مربوط به پارامتر های فیلتر مربوطه در جدول زیر آمده است:



در ادامه ضرایب تولید شده را به صورت فایل coe به IPCore مربوطه می دهیم. باید توجه داشت که با توجه به محدودیت DAC باید ورودی فیلتر را 14 بیتی در نظر بگیریم که به این ترتیب با توجه به آن که ضرایب فیلتر 16 بیتی اند، خروجی IPCore که عملا کاری ضرب کردن ضرایب در ورودی را به عهده دارد، 30 بیت خواهد بود که به صورت fixed point است. برای گرفتن ورودی از ADC و دادن خروجی به DAC برای نمایش حاصل روی اسیلوسکوپ به دو ماژول ADC و DAC نیازمند خواهیم بود.

## 

## **ماژول ADC**

این ماژول دارای ورودی های SysClk (clock سیستم)، [13:0]ADCDataIn (داده ی ورودی از سمت پورت ADC برد) و خروجی های [13:0]ADCDataOut (داده ی خروجی نمونه برداری شده) و همچنین Clk (clock کند شده) است.

این ماژول SysClk را برحسب نرخ نمونه برداری با استفاده از ماژول Divider کند کرده و مقدار جدید clock (Clk) را در اختیار ADC برد گذاشته تا با هربار یک شدن Clk مقدار ولتاژ تولید شده در آن لحظه را از طریق ADCDataIn به ماژول پاس داده و با هربار یک شدن Clk، ADCDataIn به ADCDataOut منتقل شود.

## **2-1- ماژول DAC**

این ماژول دارای ورودی های SysClk (clock سیستم)، DACDataIn [13:0] )داده ی ورودی به سمت پورت DAC از سمت برد) و خروجی های Clk(clock کند شده) و همچنین [13:0] DACDataOut (داده ی خروجی به سمت DAC) است.

این ماژول SysClk را برحسب نرخ مورد نظربرای داده گذاری روی خروجی با استفاده از ماژول Divider کند کرده و مقدار جدید clock را در اختیار DAC برد گذاشته تا با هربار یک شدن Clk مقدار ولتاژ مورد نظر را از ADCDataOut دریافت کند به این ترتیب با هربار یک شدن Clk، ADCDataIn به ADCDataOut منتقل می کنیم.

## **3-1- ماژول HighpassFilter**

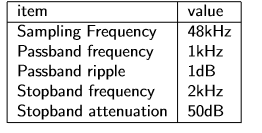
این ماژول، ماژول اصلی این بخش است و دارای ورودی های SysClk (clock سیستم)، [13:0]ADCDataIn (داده ی ورودی از سمت ADC) و هم چنین خروجی های ReadyForData (سیگنالی که هر زمان IPCore آماده ی دریافت داده باشد یک می شود)، Ready (سیگنالی که هر وقت داده ی خروجی از IPCore آماده باشد یک می شود)، DACClk (clock مربوط به DAC)، ADCClk (clock مربوط به ADC) و [13:0]DACDataOut (داده ی خروجی نهایی به سمت DAC) است.

در اینجا پس از instance گرفتن از ماژول ADC و دادن ورودی ماژول اصلی به آن، خروجی ماژول ADC را گرفته و به فیلتر می دهیم. هر زمان که خروجی فیلتر آماده گشت، آن را گرفته و 14 بیت از آن جدا کرده و به DAC می دهیم. به این ترتیب با افزودن فایل ucf و متصل کردن DAC به اسیلوسکوپ می توان حاصل فیلتر مربوطه را مشاهده کرد.

**چالش پیاده سازی:** نحوه یtruncate کردن مقدار خروجی فیلتر و جدا سازی 14 بیت از 30 بیت دشوارترین قسمت این بخش است که باید دید صرفه نظر کردن از msb ممکن است یا خیر. در صورتی می توانیم از msb صرفه نظر کنیم که بیت های msb یکسان باشند و در غیر این صورت باید به سراغ صرف نظر کردن از lsb برویم.

# **پیاده سازی Lowpass Filter**

این فیلتر به گونه ای طراحی می شود که قابلیت عبور دادن فرکانس های پایین را دارد و باقی فرکانس ها را عبور نمی دهد. به منظور پیاده سازی آن ابتدا به کمک نرم افزار matlab و قسمت fdatool ضرایب فیلتر را خواهیم ساخت. مقادیر مربوط به پارامتر های فیلتر مربوطه در جدول زیر آمده است:



در ادامه ضرایب تولید شده را به صورت فایل coe به IPCore مربوطه می دهیم. باید توجه داشت که با توجه به محدودیت DAC باید ورودی فیلتر را 14 بیتی در نظر بگیریم که به این ترتیب با توجه به آن که ضرایب فیلتر 16 بیتی اند، خروجی IPCore که عملا کاری ضرب کردن ضرایب در ورودی را به عهده دارد، 33 بیت خواهد بود که به صورت fixed point است. برای گرفتن ورودی از ADC و دادن خروجی به DAC برای نمایش حاصل روی اسیلوسکوپ به دو ماژول ADC و DAC نیازمند خواهیم بود.

## **3-2- ماژول LowpassFilter**

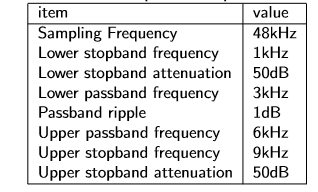
این ماژول، ماژول اصلی این بخش است و دارای ورودی های SysClk (clock سیستم)، [13:0]ADCDataIn (داده ی ورودی از سمت ADC) و هم چنین خروجی های ReadyForData (سیگنالی که هر زمان IPCore آماده ی دریافت داده باشد یک می شود)، Ready (سیگنالی که هر وقت داده ی خروجی از IPCore آماده باشد یک می شود)، DACClk (clock مربوط به DAC)، ADCClk (clock مربوط به ADC) و [13:0]DACDataOut (داده ی خروجی نهایی به سمت DAC) است.

در اینجا پس از instance گرفتن از ماژول ADC و دادن ورودی ماژول اصلی به آن، خروجی ماژول ADC را گرفته و به فیلتر می دهیم. هر زمان که خروجی فیلتر آماده گشت، آن را گرفته و 14 بیت از آن جدا کرده و به DAC می دهیم. به این ترتیب با افزودن فایل ucf و متصل کردن DAC به اسیلوسکوپ می توان حاصل فیلتر مربوطه را مشاهده کرد.

**چالش پیاده سازی:** نحوه یtruncate کردن مقدار خروجی فیلتر و جدا سازی 14 بیت از 33 بیت دشوارترین قسمت این بخش است که باید دید صرفه نظر کردن از msb ممکن است یا خیر. در صورتی می توانیم از msb صرفه نظر کنیم که بیت های msb یکسان باشند و در غیر این صورت باید به سراغ صرف نظر کردن از lsb برویم.

# **پیاده سازی Bandpass Filter**

این فیلتر به گونه ای طراحی می شود که قابلیت عبور دادن فرکانس های میانی (نه بالا نه پایین) را دارد و باقی فرکانس ها را عبور نمی دهد. به منظور پیاده سازی آن ابتدا به کمک نرم افزار matlab و قسمت fdatool ضرایب فیلتر را خواهیم ساخت. مقادیر مربوط به پارامتر های فیلتر مربوطه در جدول زیر آمده است:



در ادامه ضرایب تولید شده را به صورت فایل coe به IPCore مربوطه می دهیم. باید توجه داشت که با توجه به محدودیت DAC باید ورودی فیلتر را 14 بیتی در نظر بگیریم که به این ترتیب با توجه به آن که ضرایب فیلتر 16 بیتی اند، خروجی IPCore که عملا کاری ضرب کردن ضرایب در ورودی را به عهده دارد، 31 بیت خواهد بود که به صورت fixed point است. برای گرفتن ورودی از ADC و دادن خروجی به DAC برای نمایش حاصل روی اسیلوسکوپ به دو ماژول ADC و DAC نیازمند خواهیم بود.

## **3-2- ماژول BandpassFilter**

این ماژول، ماژول اصلی این بخش است و دارای ورودی های SysClk (clock سیستم)، [13:0]ADCDataIn (داده ی ورودی از سمت ADC) و هم چنین خروجی های ReadyForData (سیگنالی که هر زمان IPCore آماده ی دریافت داده باشد یک می شود)، Ready (سیگنالی که هر وقت داده ی خروجی از IPCore آماده باشد یک می شود)، DACClk (clock مربوط به DAC)، ADCClk (clock مربوط به ADC) و [13:0]DACDataOut (داده ی خروجی نهایی به سمت DAC) است.

در اینجا پس از instance گرفتن از ماژول ADC و دادن ورودی ماژول اصلی به آن، خروجی ماژول ADC را گرفته و به فیلتر می دهیم. هر زمان که خروجی فیلتر آماده گشت، آن را گرفته و 14 بیت از آن جدا کرده و به DAC می دهیم. به این ترتیب با افزودن فایل ucf و متصل کردن DAC به اسیلوسکوپ می توان حاصل فیلتر مربوطه را مشاهده کرد.

**چالش پیاده سازی:** نحوه یtruncate کردن مقدار خروجی فیلتر و جدا سازی 14 بیت از 31 بیت دشوارترین قسمت این بخش است که باید دید صرفه نظر کردن از msb ممکن است یا خیر. در صورتی می توانیم از msb صرفه نظر کنیم که بیت های msb یکسان باشند و در غیر این صورت باید به سراغ صرف نظر کردن از lsb برویم.

**اشکال:** متاسفانه علی رغم موفق به جواب گرفتن از این بخش نشدیم و علت آن نیز هنوز مشخص نیست.

# **پیاده سازی Moving Average Filter**

این فیلتر به گونه ای طراحی می شود که قابلیت عبور دادن فرکانس های پایین را دارد و باقی فرکانس ها را عبور نمی دهد. به منظور پیاده سازی آن نوعی میانگین گیری بین داده ها صورت می گیرد و در صورتی که تغییرات زیاد باشد داده را صفر کرده و فقط تغییرات کم را عبور می دهد.

برای گرفتن ورودی از ADC و دادن خروجی به DAC برای نمایش حاصل روی اسیلوسکوپ به دو ماژول ADC و DAC نیازمند خواهیم بود.

## **1-4- ماژول Filter1**

این ماژول دارای ورودی های Clk ( clockسیستم)، و [13:0]DataIn (داده ی ورودی از سمت ADC) و خروجی [13:0]DataOut (داده ی خروجی به سمت DAC) است. در این ماژول به وسیله ی 64 خانه ای داده ی های ورودی را ذخیره کرده و بین آن ها میانگین گیری می کند.

## **4-2- ماژول TopMovingAverage**

این ماژول، ماژول اصلی این بخش است و دارای ورودی های SysClk (clock سیستم)، [13:0]ADCDataIn (داده ی ورودی از سمت ADC) و هم چنین خروجی های DACClk (clock مربوط به DAC)، ADCClk (clock مربوط به ADC) و [13:0]DACDataOut (داده ی خروجی نهایی به سمت DAC) است.

در اینجا پس از instance گرفتن از ماژول ADC و دادن ورودی ماژول اصلی به آن، خروجی ماژول ADC را گرفته و به فیلتر می دهیم. و خروجی را به DAC می دهیم. به این ترتیب با افزودن فایل ucf و متصل کردن DAC به اسیلوسکوپ می توان حاصل فیلتر مربوطه را مشاهده کرد.