بسم رب الشهدا و الصدیقین

سیستم های مخابراتی

دکترهادی

مهدی زینلی قهرودی، حسین مقیم، محمدامین انصاری

طراحی بلوک های DAC/ADC

معماری پروژه:

پروژه به صورت دو تابع اصلی مربوط به بلوک های DAC/ADC و تعدادی تابع کوچکتر که تشکیل دهنده بلوک های اصلی خواهند بود هستند.

کتابخانه ها:

در ابتدا باید کتاب‌خانه هایی که برای این پروژه لازم داریم را اضافه کنیم.

Scipy/scipy.io/numpy/math

تابع generate\_final\_signal:

در این تابع صورت مورد اضافه می‌شود و سپس کانال های چپ و راست این صورت با وزن مساوری merge می‌شوند، با استفاده از سیگنال merge شده و با نرخ نمونه برداری مشخص fs یک سیگنال نهایی به صورت vector با کمک کتابخانه numpy می‌سازیم؛ در انتها نیز دو مقدار final\_signal, samplerate را برمی‌گردانیم.

تابع quantization:

در این تایع ما می‌خواهیم سیگنال ها را ب مقادیر مورد نظر که v تعیین کننده آن است map کنیم، برای کار نیاز داریم در ابتدا با روش موجود در اسلایدها، بازه دامنه سیگنال را به بازه های کوچکتر با فاصله delta تقسیم کنیم، سپس هر مقدار از سیگنال را چک کنیم و با مقایسه بین treshhold بازه‌ها آنها را به بازه مورد نظر map کنیم. در نهایت نیز مقدار quantized\_signal را برمی گردانیم.

تابع generate\_bit\_stream:

در این تابع با استفاده از یک refrence که بستگی به مود مورد نظر دارد (nbc, gray) تمامی مقادیر را به عدد باینری خود map می‌کنیم که این refrence نیز با استفاده از تابع هایی که در بالاتر از این تابع وجود دارد پیاده شده است، به صورتی که به یک لیست از bit\_stream ها خواهیم رسید، در نهایت بایکی کردن اعضای این لیست یک رشته از 0,1 خواهیم داشت که این مقدار باید وارد بلوک بعدی شود.

تابع ADC:

در نهایت با استفاده از تابع های بالا اقدام به پیاده سازی تابع نهایی ADC خواهیم کرد. که ۴ ورودی خواهد داشت. ۱-صدا ۲-فرکانس نمونه برداری بعد از load کردن سیگنال صوت ۳-v که نشان دهنده سطح کوانتایز خواهد بود ۴-modelکه بیانگیر مدل map شدن 0,1 ها است.

خروجی این تابع نیز ۳ سیگنال مرحله های قبل، علاوه‌بر refrence, samplerate می‌باشد.

تابع reverse\_bit\_stream:

این تابع همان رشته از 0,1 را دریافت می‌کند و بردار نهایی از داده‌های نرمال شده بین 2\*\*(v-1) و منفی این مقدار برمی گرداند.

تابع generate\_reverse\_signal:

این تابع برای تنظیم دامنه سیگنال خواهد بود، که با استفاده از مقدار ماکزیمم سیگنال که در بلوک قبل استخراج کردیم دامنه سیگنال رو تنظیم ‌می‌کنیم.

تابع low\_pass\_filter:

این تابع یک فیلتر پایین‌گذر است که با گرفتن مقادیر مناسب و یک سیگنال حوزه زمان کار می‌کند، این تابع ابتدا اقدام به گرفتن فوریه و بعد اعمال فیلتر می‌کند، در نهایت نیز سیگنال فیلتر شده را با استفاده از معکوس فوریه به حوزه زمان منتقل می‌کند.

تابع DAC:

این تابع با استفاده از تابع های قبلی یک صورت را در فایلی با نام output\_sound ذخیره می‌کند.

هنگامی که فیلتر موردنظر اعلام می‌شود صدای صوت اصلی نسبت به حالت قبلی کم می‌شود، بنابراین ما دو صوت در فایل خروجی داریم، یک فایل با اعمال lpf و یک فایل هم بدون اعمال‌ کردن lpf.