

Ejercicio

- Python Fixed Point

El objetivo es aplicar los conceptos de Aritmética de Punto Fijo usando la clase de Python *fixedInt.py*.

- Ejercicio 1

Aplicar los conceptos de aritmética de punto fijo sobre el filtro rcosine del simulador realizado en python en el archivo *tx_rcosine_procom.py*. Para ello completar las siguientes consignas:

1. Generar tres filtros con *rolloff* [0.0,0.5,1.0] en punto flotante con $N_{Baud} = 16$, $F_{Baud} = 1G$ y $OS = 8$.
2. Graficar la respuesta al impulso y frecuencia.
3. Graficar la convolución de los tres filtros con los símbolos a transmitir y la constelación buscando la fase óptima.
4. Sobre cada filtro aplicar las siguiente cuantizaciones: $S(8, 7)$ truncado, $S(8, 7)$ redondeo, $S(3, 2)$ truncado, $S(3, 2)$ redondeo, $S(6, 4)$ truncado y $S(6, 4)$ redondeo. En todos los casos considerar saturación.
5. Realizar con los nuevos filtros las gráficas anteriores y comparar resultados.
6. Obtener la SNR en dB de salida de la convolución para cada caso de cuantización de los coeficientes del filtro.

- RTL Fixed Point

El objetivo es aplicar los conceptos de Aritmética de Punto Fijo sobre operaciones simples a nivel de RTL.

- Ejercicio 2

Completar las siguientes consignas:

1. Escribir un módulo en Verilog que permita realizar una suma de dos entradas en punto fijo con los siguientes formatos: **S16.14 y S12.11**, entregando diferentes salidas en los siguientes formatos:
 - a) Full-resolution
 - b) S11.10 con overflow y truncado
 - c) S11.10 con saturación y truncado
 - d) S9.8 con saturación y redondeo
2. Escribir un módulo en Verilog que permita realizar una multiplicación de dos entradas en punto fijo con los siguientes formatos: **S8.6 y S12.11**, entregando diferentes salidas en los siguientes formatos:

- a)* Full-resolution
 - b)* S12.11 con overflow y truncado
 - c)* S12.11 con saturación y truncado
 - d)* S10.9 con saturación y redondeo
- 3. Para los dos casos anteriores escribir un testbench que permita validar el hardware diseñado obteniendo vector-matching con vectores generados en Python empleando entradas aleatorias cuantizadas con la librería *fixedInt*.
- 4. Para los dos casos anteriores incluir además, un puerto de salida de un bit que indique con un nivel alto la existencia de una saturación en el resultado de salida.