Cinvestav

Reporte Análisis punto fijo e implementación

Jhonatan Alexander Gomez Gamboa

2023-03-21

1. Introducción experimental.

En el presente reporte se presenta el análisis de punto fijo e implementación de un módulo top el cual contiene un bloque sumador y un bloque multiplicador.

El análisis de punto fijo se realizó utilizando la libreria fxpmath, la cúal fue concebida para este tipo de análisis, despues de realizar dicho análisis se implementó utilizando la herramienta Quartus prime, donde se implementara y simulara las diferentes etapas de la operación $A\cos(2\pi t) + C$, tanto en la herramienta de análisis como en el ambiente de quartus, para complementar el caso de estudio se caluló el SQNR (relación señal-ruido de cuantización) con una cantidad de 30 dB.

Se presentaran al final del reporte las evidencias de la simulación y la comparación de los resultados obtenidos donde se espera que las salidas de Quartus sean exactamente los mismos con las salidas del análisis.

2. Ambiente experimental.

Quartus (Quartus Prime 17.7 lite Edition)

ModelSim

Spyder (IDE de python 3)

Libreria fxpmath (libreria especializada en el análisis de punto fijo)

3. Programa.

3.1 Analisis en punto fijo utilizando Spyder y la liberia fpxmath

El análisis de realizo en Python utilizando el IDE Spyder. A continuación se presentaran las etapas que nos llevaron a obtener el número de bits para cada representación y el SQNR de la salida.

1. Conversión de la constante A=3.7 a punto flotante utilizando la libreria fxpmath

```
a = Fxp(3.7, signed = True, n_word = 6, n_frac=3)
```

2. Obtención del coseno cuantizado y su conversión a punto fijo utilizando fxpmath

```
time = np.linspace(0, 1, 15)
cos_ana = np.cos(2*math.pi*time)
cos_cuan = Fxp(cos_ana, signed=True, n_word=6, n_frac=5)
```

3. Obteniendo el producto de $A \cdot B = A\cos(2\pi t)$ para posteriormente realizarle un truncamiento.

```
acos_cuan = a * cos_cuan

acos_cuan.resize(True, 10, 5)
```

4. Representación y conversión de la constante C=13.7 a punto flotante utilizando la libreria fxpmath

```
c = Fxp(13.2, signed = True, n_word = 10, n_frac=5)
```

5. Obtención de la suma de $A\cos(2\pi t) + C$ y su respectivo truncamiento.

```
dataout = c + acos_cuan

dataout.resize(True, 7, 1)
```

6. Calculo del utilizando la siguiente formula.

```
#P_x
v_out = cos_ana*3.7 + 13.2
p_x = 0
for i in range(len(v_out)):
    p_x = p_x + math.pow(abs(v_out[i]), 2)
p_x = p_x/len(v_out)

#Pn
v_diff = v_out - dataout
p_n = 0
for i in range(len(v_diff)):
    p_n = p_n + math.pow(abs(v_diff[i]), 2)
p_n = p_n/len(v_diff)

SQNR = 10*math.log(p_x / p_n, 10)
print(SQNR)
```

$$SQNR = 10\log_{10}(\frac{\sum_{i=1}^{n}|x_i|^2}{\sum_{j=1}^{n}|x_j|^2})$$

donde:

$x_i,\ i = 1, 2, \ldots, n$ son los valores del vector de la salida real de $A\cos(2\pi t) + B$ en punto flotante (golden reference)

$x_j,\ j = 1, 2, \ldots, n$ son los valores del vector obtenido por la diferencia de la golden reference y el vector de salida con nuestros valores en formato punto fijo.

Table 1: Resultados obtenidos del análisis de punto fijo en python.

Variable	**tamaño**	**int**	**frac**
A_top	6	2	3
B_top	6	0	5
C_top	10	4	5

Table 2: Datos en formato table de los valores que toma el coseno en punto fijo y la salida ACos(n)+C en punto fijo.

Valores de coseno en fxp	**Valores de la salida del top**
011111	0100001
011100	0100000
010011	0011110
000111	0011011
111001	0011000
101101	0010110
100100	0010100
100000	0010011
100100	0010100
101101	0010110
111001	0011000
000111	0011011
010011	0011110
011100	0100000
011111	0100001

Utilizamos este análisis para poder conocer los valores maximos y minimos que puede tomar la salida en punto flotante y tener en cuenta los bits necesarios para representar estos valores, los cúales nos bastaba con mantener 5 bits en la parte entera, uno de signo y uno de decimal, ya que el valor maximo que podra tomar la salida del módulo en punto flotante sera de $x_{sup} = 16.9$ y para el limite inteferior tenemos $x_{inf} = 9.5$.

3.1.1 Calculó del SQNR

Como parte de la practica, se nos solicitó realizar el análisis del punto fijo con un SQNR de 30 dB, por lo cual utilizando la fórmula presentada en el punto anterior la calculamos, primero obtendremos el vector que contiene los valores de la salida real en punto flotante la cúal utilizaremos como nuestra golden reference:

$V = [16.9, 16.53358481, 15.5069122, 14.02332746, 12.37667254, 10.89308773, 9.86641519, 9.5, 9.86641519, 10.89308773, 12.37667$

Con este vector podremos calcular la potencia de la señal:

$$P_x = \sum_{i=1}^{n} |V_i|^2/n$$

despues obtendremos el vector de la salida del módulo implementado en Quartus, dado por los parametros en punto fijo:

$U = [16.5, 16.0, 15.0, 13.5, 12.0, 11.0, 10.0, 9.5, 10.0, 11.0, 12.0, 13.5, 15.0, 16.0, 16.5]$.

con este ultimo y el vector V de los valores en punto flotante, podremos calcular la potencia del error aplicando la fórmula anterior para calcular la potencia de la señal pero con el vector de diferencias $W = V - U$

Con este vector podremos calcular la potencia del error:

$$P_n = \sum_{i=1}^{n} |W_i|^2/n$$

y por ultimo calcularemos la relación señal a ruido como sigue:

$$SQNR = 10\log_{10}(\frac{P_x}{P_n}) = 30.898918190001005$$

Con esto finalizamos el análisis de punto fijo, en el siguiente punto, se implementara en Quartus para poder obtener la salida y compararla con los valores obtenidos aqui

3.2 Implementación del modulo en Quartus.

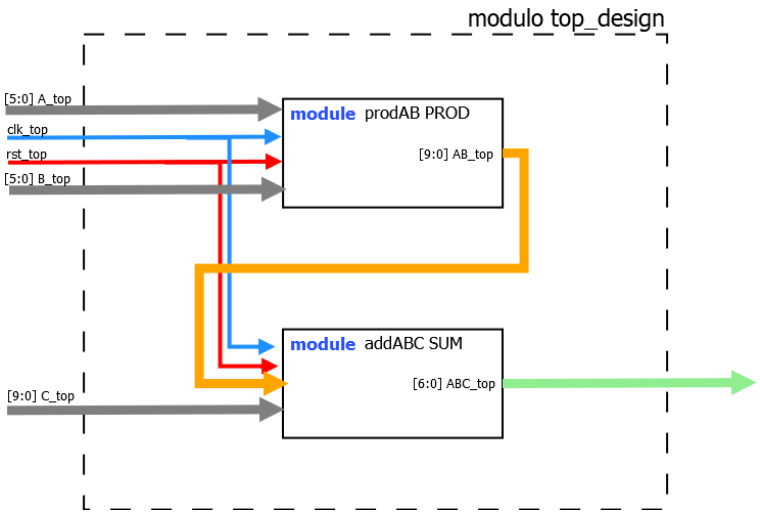

Figure 1: *Diagrama de bloques que representa la arquitectura del módulo implementado para el análisis de punto fijo*

El modulo top consta de un módulo multiplicador de 6×6 bits en formado signado y decimal, el producto es resultado de las entradas **A_top** la cual es una constante con valor *3.7*, al momento de representarla en fórmato punto fijo su valor real es de *3.625* lo representamos con un ancho de palabra de 6 bits, de los cuales 3 bits son utilizados para representar la parte fraccionaria y 2 bits son utilizados para representar la parte entera, mas el bit de signo.

El segundo número es **B_top** el cuál tomara los valores de [-1: 0.96875], estos valores son los limites que podremos representar en fórmato punto fijo para representar estos valores se utilizarón 6 bits de ancho de palabra, de los cuales 5 bits son para representar la parte fraccionaria, 0 bits para la parte entera y 1 bit para representar el bit de signo.

Estos valores fueron calculados al realizar el análisis de punto fijo utilizando la herramienta de Spyder, la cual es un IDE de Python para el análisis de datos. Dichos resultados se presentan en la siguiente sección.

El diseño consta de un módulo top llamado **top_design** como se puede ver apreciar en la (figura 1) consta de las siguientes entradas y salidas descritas en la tabla 2.

Table 3: Entradas y salidas del modulo TOP design.

Señal	**Dirección**	**Tamaño**	**Descripción**
clk_top	in	1	Señal de reloj interna del FPGA de 50MHz
rst_top	in	1	Señal que permite inicializar todos los registros
A_top	in	6	Constante en formato punto fijo con valor 3.625
B_top	in	6	Valores de coseno muestreado en punto fijo signado
C_top	in	10	Constante en formato punto fijo con valor 13.1875
AB_top	out	10	Resultado del produco $A \cdot B$
ABC_top	out	7	Resultado de la suma $AB + C$

Una vez teniendo el módulo top creado y los valores del coseno cuantizados para 15 valores, estos se guardaron en un archivo txt para posterioemente ser cargados en Quartus y poder ingresarlos al diseño en una de sus entradas **B_top**.

los valores del coseno cuantizado se presentan en la siguiente tabla.

Table 4: Valores del coseno en punto fijo

n	**valor**
0	01_1111
1	01_1100
2	01_0011
3	00_0111
4	11_1001
5	10_1101
6	10_0100
7	10_0000
8	10_0100
9	10_1101
10	11_1001
11	00_0111
12	01_0011
13	01_1100
14	01_1111

3.2.1 Diseño de la cama de pruebas

Se crean las respectivas entradas con sus tamaños previamente calculados en el análisis d epunto fijo, se hace la instanciación del módulo top (figura 1) el cúal internamente contiene los módulos de suma y multiplicación.

Las entradas **A_top** y **C_top**, son constante por lo cuál no cambian su valor durante la ejecución de la simulación. Utilizando la función del sistema $readmemb() podemos cargar un archivo de texto y guardarlo en un registro de palabras que tendra una longitud de palabra de 7 bits y 15 slots para datos o direciones de memoria, el archivo con los datos del coseno en punto fijo se llama “datacoseno.txt” y debe de colocarse en la carpeta de la simulación.

se itera sobre este registro, por cada slot de memoria y esos datos se van inyectando mediante la entrada **B_top**, para, posteriormente terminar la simuluación despues de haberlos ingresados todos.

4. Resultados experimentales.

En esta sección se presentaran los resultados experimentales de la simulación y la comparación de los valores obtenidos en el análisis de punto fijo y lo datos obtenidos por la herramienta de Quartus.

Primero comenzaremos mostrando las diferentes variables y sus nombres, asi como el nombre de las constantes y sus valores en simulación (figura 2).

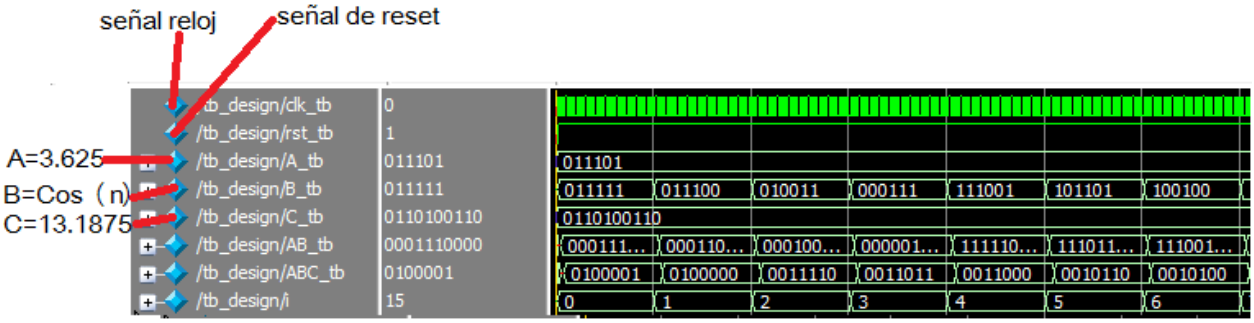

Figure 2: *Figura que nos muetra los primeros 7 valores de B siendo operado, y los nombres de las variables de interes*

Podemos apreciar en la (figura 3) los valores que toma el coseno en punto fijo, lo cual corresponde a las 15 muestras de este mismo, podemos apreciar como la función corresponde a un coseno y la salida del módulo sera tambien un coseno pero de con una amplitud mayot y un offset positivo (figura 3).

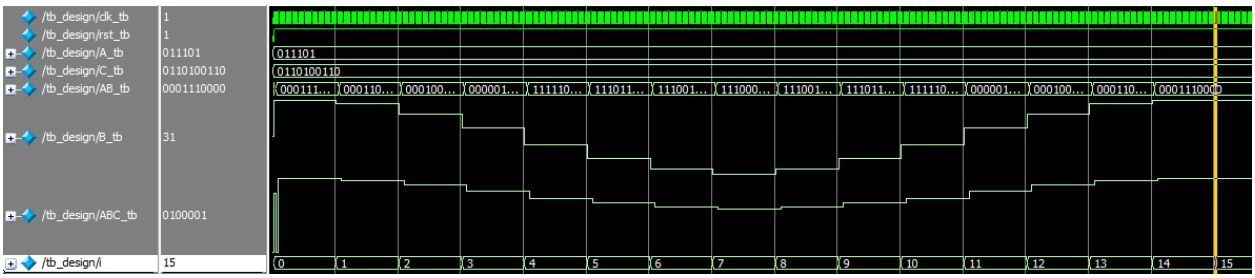

Figure 3: *Representación en forma de onda del coseno en punto fijo y el resultado a la salida del módulo top*

En la siguiente figura es necesario resaltar los valores que van tomando los valores del coseno en punto fijo asi como los valores a la salida del módulo top, ya que posteriormente las utilizaremos para comparar estos valores con los obtenidos en el script de python (figura 4).

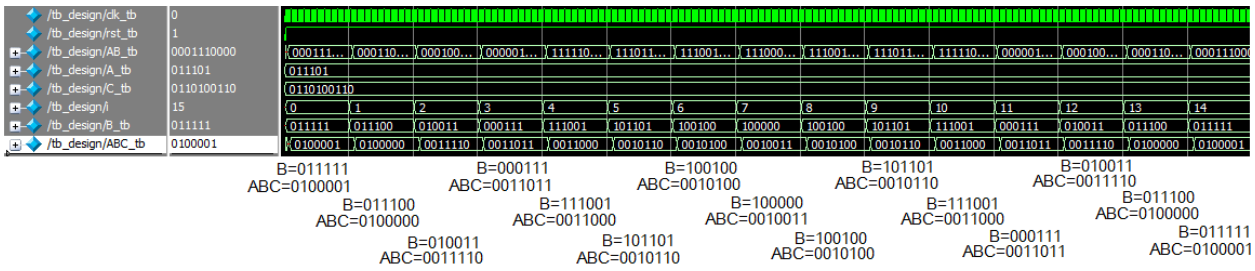

Figure 4: *Figura que permite apreciar los valores que van tomando las variables a lo largo de la simulación*

En la siguiente figura se muestran los valores que toma la entrada del coseno en punto fijo con respecto al valor que toma la salida del módulo top en punto fijo obtenidas en el scriot de python (figura 5).

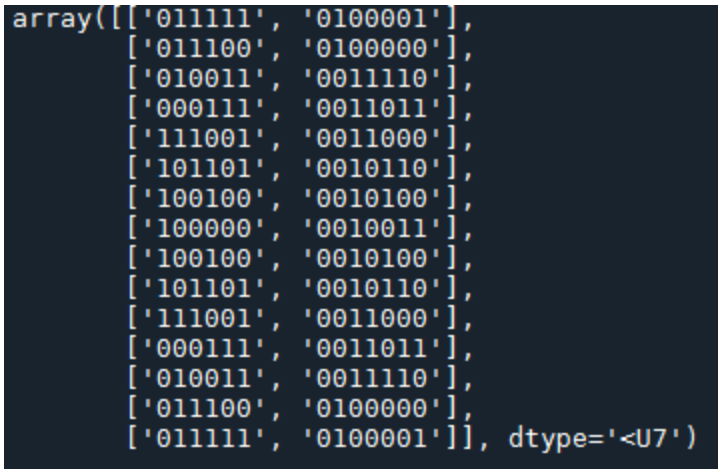

Figure 5: *Salida del Script de python para análisis de punto fijo donde nos muestra los valores que toma el coseno en punto fijo y su respectiva salida del módulo top*

5. Concluciones.

Habiendo presentado los resultados procederemos a plantear las conclusiones.

Es necesario priorizar que el anális de punto fijo se realizó de forma satisfactoria, como podemos observar al comparar la figura 4 y la figura 5, los valores obtenidos tanto en el análisis en python como los datos obtenidos en la simulación son exactamente los mismo, esto forma parte de los requerimientos de la practica, asi como lograr un $SQNR$ de 30 dB, como se puede apreciar en sección 3.1.1 en valor del la relación señal a ruido concuerda con lo solicitado.

La principal dificultad encontrada en esta practica fue el representar los valores en punto fijo dentro de la herramienta de Quartus, ya que a diferencia de python con sy libreria fxpmath que nos permite hacer todo tipo de manipulación a los valores, Quartus no, es necesario adecuar los valores para que sean los correctos al conectarse a otros módulos.