



Universidad de Alcalá

ESCUELA POLITÉCNICA SUPERIOR

**MÁSTER UNIVERSITARIO EN INGENIERÍA DE
TELECOMUNICACIÓN**

**PRÁCTICA ENTREGABLE 1
TRANSMISOR 16 QAM**

Diseño de Circuitos Electrónicos para Comunicaciones

Autor:

Paula Bartolomé Mora

20 de diciembre de 2023

Índice general

1. Introducción	1
2. Captura XADC y memoria FIFO	2
2.1. Obtención de los datos convertidos del ADC	3
2.1.1. Configuración del XADC	3
2.1.2. Simulación de la señal de entrada	3
2.2. Lógica de control de lectura y escritura de los datos de la FIFO	5
2.2.1. Configuración del bloque FIFO	5
2.2.2. Comprobación del bloque FIFO	7
3. Modulador 16-QAM	8
3.1. Mapeado 16-QAM	8
3.2. Zero Padding	10
3.3. Bloque de filtrado pulse shaping - RRC	10
4. Bloque DDS	11
5. Conclusiones	12

Capítulo 1

Introducción

El presente trabajo tiene como objetivo principal el desarrollo de un sistema de comunicaciones basado en 16-QAM (Modulación de Amplitud en Cuadratura de 16 niveles). Para ello, se propone el empleo en conjunto de los diferentes bloques electrónicos IP estudiados en la asignatura y se divide el desarrollo del entregable en una serie de fases de configuración:

- **Captura XADC y Memoria FIFO:** En esta primera etapa se realizará la captura de una señal triangular y se procederán a almacenar los datos de entrada en una memoria FIFO.
- **Modulador 16-QAM:**
 - **Mapeado QAM y Zero Padding:** Como siguiente paso a seguir, se generará el mapeado de 16-QAM y se implementará un Zero-Padding 1:32.
 - **Filtrado Root Raised Cosine:** Tras aplicar Zero Padding, se deberá generar y aplicar a cada rama I/Q el filtrado pulse shaping del “root raised cosine” (RRC).
 - **Mezclador DDS y Multiplicadores:** ————
 - **Sumador:** ————
 - **Rx:** ————

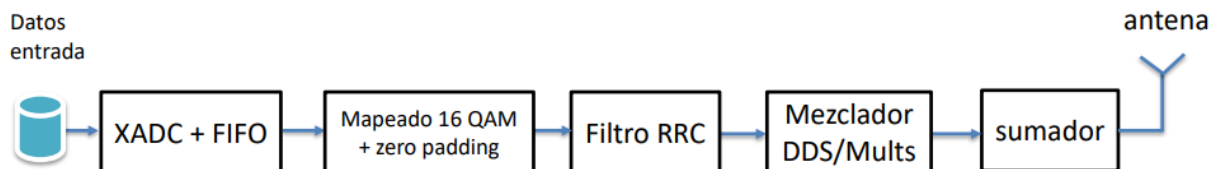


Figura 1.1: Distribución de bloques del sistema de comunicaciones

Para lograr tanto un correcto funcionamiento de cada bloque electrónico como la interoperabilidad entre ellos al integrarlos en el sistema, será preciso un estudio en profundidad de la documentación proporcionada por el fabricante y un análisis cuantitativo y cualitativo de cada uno de los resultados obtenidos en las simulaciones funcionales.

Capítulo 2

Captura XADC y memoria FIFO

Como se ha introducido anteriormente, en este capítulo se definirá el proceso de captura y conversión de una señal de entrada triangular y el posterior almacenamiento de la misma en una memoria FIFO. En la Figura 2.5 se muestra el diseño de los bloques necesarios para esta fase y la definición de sus correspondientes puertos de entrada y salida.

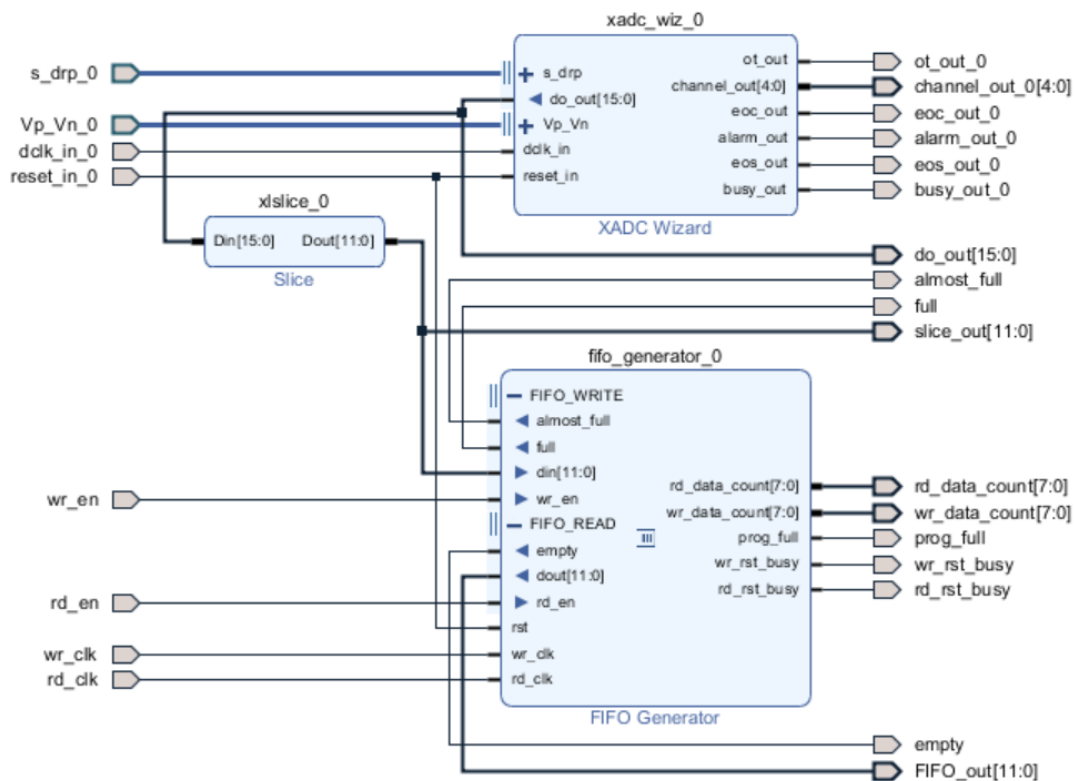


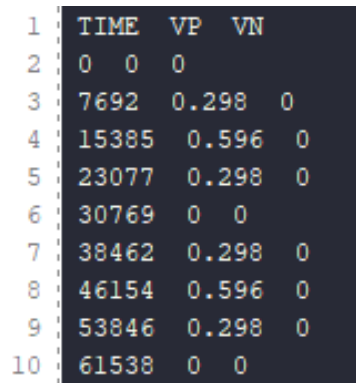
Figura 2.1: Diseño del bloque XADC+FIFO

2.1. Obtención de los datos convertidos del ADC

2.1.1. Configuración del XADC

En primer lugar, teniendo en cuenta que nuestro puesto de laboratorio es el 8, se debe generar a la entrada una señal triangular con un valor de frecuencia igual a 13KHz y que opere en un rango de valores de tensión entre 0 y 0.596 voltios. Los valores que toma en cada instante temporal serán definidos en el fichero `design.txt`, el cual tendrá que ser importado al proyecto para llevar a cabo la conversión.

Como se puede ver en la Figura 2.2 se definen valores de tensión para dos entradas (VP y VN), que corresponden a las entradas analógicas diferenciales del XADC. Por simplificación, se configura el valor absoluto de tensión para la entrada positiva (VP) y un valor nulo para la negativa (VN).



	TIME	VP	VN
1	0	0	0
2	7692	0.298	0
3	15385	0.596	0
4	23077	0.298	0
5	30769	0	0
6	38462	0.298	0
7	46154	0.596	0
8	53846	0.298	0
9	61538	0	0
10			

Figura 2.2: Valores de tensión (V) de la señal triangular extraídos de `design.txt`

El XADC se configura en modo continuo y con un único canal de entrada para capturar las muestras, por lo que se ha seleccionado el canal 3 para registrar el valor convertido de tensión (*s_drp_daddr*). La conversión se realizará a una velocidad de 1 MS/seg con un reloj de 52 MHz y se establecerá un tiempo de adquisición igual a 4 para que la señal de captura sea estable.

2.1.2. Simulación de la señal de entrada

Una vez configurado el bloque IP XADC se procede a simular la conversión de la señal de entrada. En la figura 2.3 se puede visualizar cómo se obtiene la salida digital de 16 bits por el puerto *do_out*.

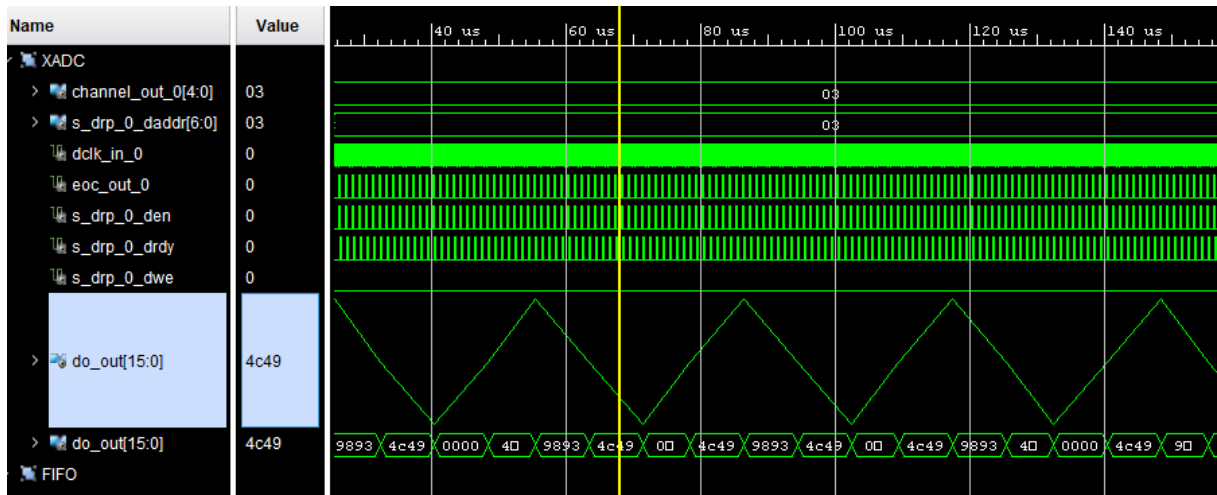


Figura 2.3: Conversión analógico-digital de la señal triangular de entrada (I)

Por otro lado, en la Figura 2.4 se aprecia de una forma más clara los valores que toman cada uno de los puertos del XADC en cada conversión. La señal de fin de conversión *eoc_out* se activa y como consecuencia, también lo hará la de enable del Puerto de Reconfiguración Dinámico (DRP) *eoc_out*.

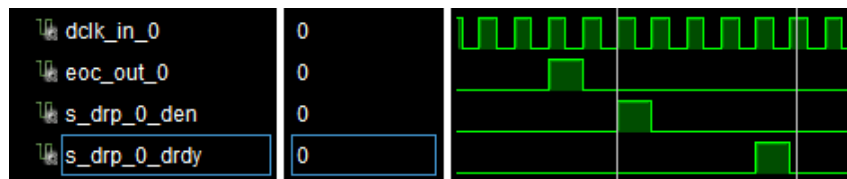


Figura 2.4: Fin de conversión y dato preparado (III)

La operación de lectura se completa cuando la señal de dato ready se activa *s_drp_drdy*, indicando que se ha sacado el dato a la salida y que se puede proceder a convertir el siguiente.

```

1 -- frec. conversion aprox 1MS -> frec 52 KHz (modo continuo)
2 for i in 0 to 200 -- a la espera de 200 datos
3 loop
4     -- End of Conversion signal
5     wait until eoc_out_0 <= '1' ;
6     -- Data ready signal for the dynamic reconfiguration port
7     wait until s_drp_0_drdy = '1';
8 end loop;

```

Listing 2.1: Bucle de conversión de los 200 datos

2.2. Lógica de control de lectura y escritura de los datos de la FIFO

2.2.1. Configuración del bloque FIFO

El bloque de memoria FIFO almacenará cada uno de los datos capturados por el bloque XADC. Consta de una capacidad de 256 posiciones de 12 bits cada una, por lo que a la entrada del bloque FIFO será precisa la configuración de un bloque auxiliar Slice para transformar los 16 bits de la señal de salida del XADC en sus 12 bits de mayor peso.

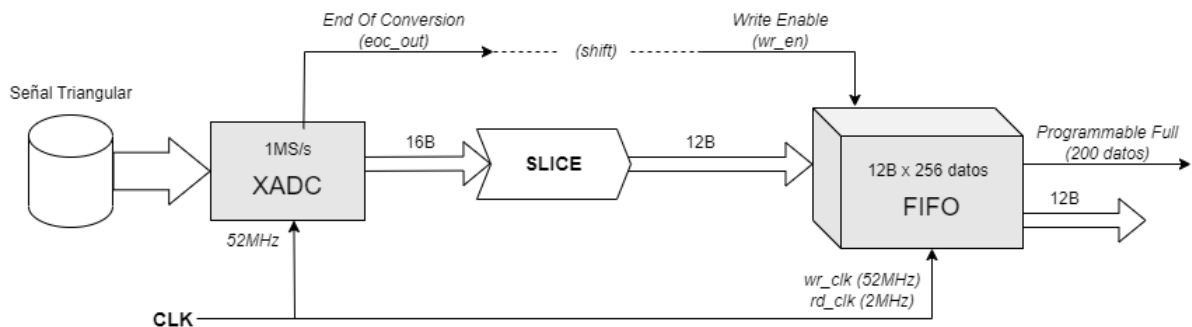


Figura 2.5: Diseño del bloque XADC+FIFO

Se trata de un paso necesario porque el XADC en el fondo maneja una precisión de 12 bits a la salida. En la Figura 2.5 se puede apreciar esta conversión, además de la estructura completa del bloque a implementar.

```

1 process(wr_clk)
2 variable reg : std_logic; -- variable para almacenar dato de salida del
   xadc
3 begin
4 if rising_edge(wr_clk) then
5     if reset_in_0 = '1' then -- reset XADC = 1
6         reg := '0';
7         -- Enable signal for the dynamic reconfiguration port
8         s_drp_0_den <= '0';
9     else
10        s_drp_0_den <= reg ; -- ENABLE = 1 solo durante un ciclo
11        reg := eoc_out_0; -- Se habilita la salida de un nuevo dato
12    end if;
13 end if;
14 end process;
15
16 wr_en <= s_drp_0_drdy; -- se escribe en la FIFO nuevo dato

```

Listing 2.2: Proceso de escritura

Entrando en el proceso de escritura, el funcionamiento sería el siguiente: cada vez que exista un nuevo dato proporcionado por el XADC se escribirá en la memoria FIFO de forma continua a 1 MS/s y con un reloj de 52 MHz (*wr_clk*) hasta llegar a la capacidad máxima.

Una vez la memoria FIFO está llena, comenzará el proceso de lectura de 200 datos a una frecuencia de lectura de 2MHz (*rd_clk*) mediante la activación de la señal (*rd_enable*). En este caso, al desear leer un número de datos menor a la capacidad máxima (256), será necesario configurar una señal auxiliar programable (*programmable_full*).

```

1 process(rd_clk)
2 file outfile: text is out "FIFO_output.txt";
3 variable line_num: line;
4 variable opened, finished: std_logic := '0';
5 begin
6     file_open (outfile, "FIFO_output.txt", write_mode);
7     if rising_edge(rd_clk) then
8         if reset_in_0 = '1' then -- reset XADC = 1
9             rd_en <= '0';
10        else
11            --se comienza a leer la FIFO
12            if prog_full = '1' then
13                rd_en <= '1';
14            end if;
15            if empty = '1' then
16                rd_en <= '0';
17            end if;
18            --proceso de escritura del .txt
19            if rd_en = '1' and finished = '0' then
20                write (line_num, FIFO_out);
21                writeline(outfile, line_num);
22                opened := '1';
23            end if;
24            --Se finaliza la escritura de .txt tras el primer empty
25            if opened = '1' and empty = '1' then
26                finished := '1';
27            end if;
28        end if;
29    end if;
30 end process;

```

Listing 2.3: Proceso de lectura

Adicionalmente, es necesario implementar la escritura de un fichero de salida con los datos almacenados en la FIFO, en el cual cada dato de 12 bits será una nueva línea. Este servirá para simular la lectura de la FIFO desde otro proyecto independiente de Vivado, suponiendo de este modo la entrada del siguiente bloque de mapeado. Se debe limitar la escritura a los primeros 200 datos de salida mediante el uso de variables adicionales (*opened*, *finished*).

2.2.2. Comprobación del bloque FIFO

En las Figuras 2.6 y 2.7 se pueden visualizar los procesos de escritura y de lectura en la memoria FIFO como se ha descrito en el apartado anterior. A modo de depuración de que se está produciendo un correcto funcionamiento según los tiempos configurados, se añaden a la simulación dos señales de contadores de ciclos de lectura y de escritura.

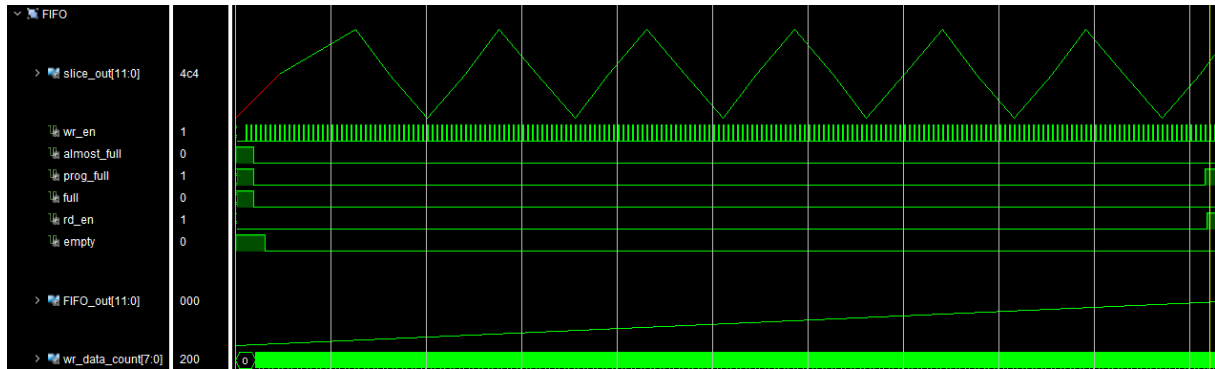


Figura 2.6: Simulación del proceso de escritura en la memoria FIFO

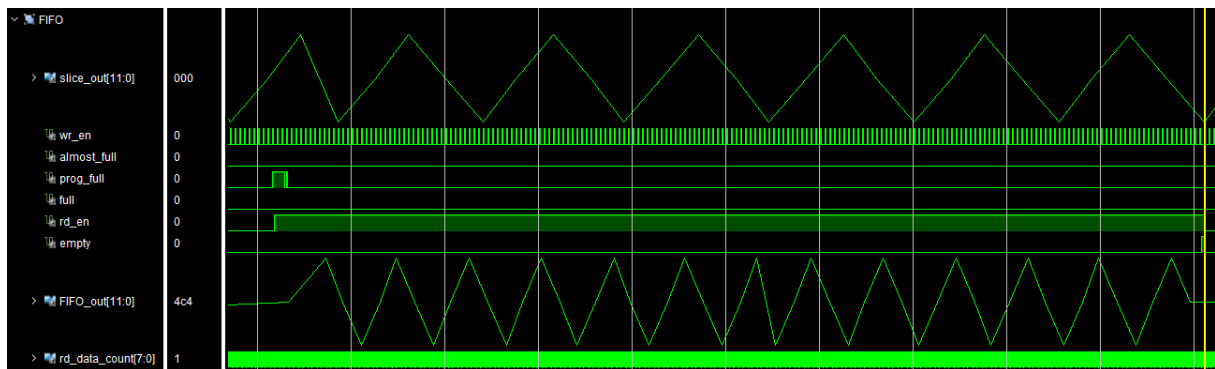


Figura 2.7: Simulación del proceso de lectura en la memoria FIFO

Modulador 16-QAM

En este capítulo se muestra el bloque encargado de la modulación 16-QAM. Este bloque se compone de tres procesos: mapeado 16-QAM, aplicación de Zero Padding y filtrado Root Raise Cosine. En la Figura 3.1 se puede visualizar la estructura de los bloques IP que se han incluido en este segundo proyecto de Vivado.

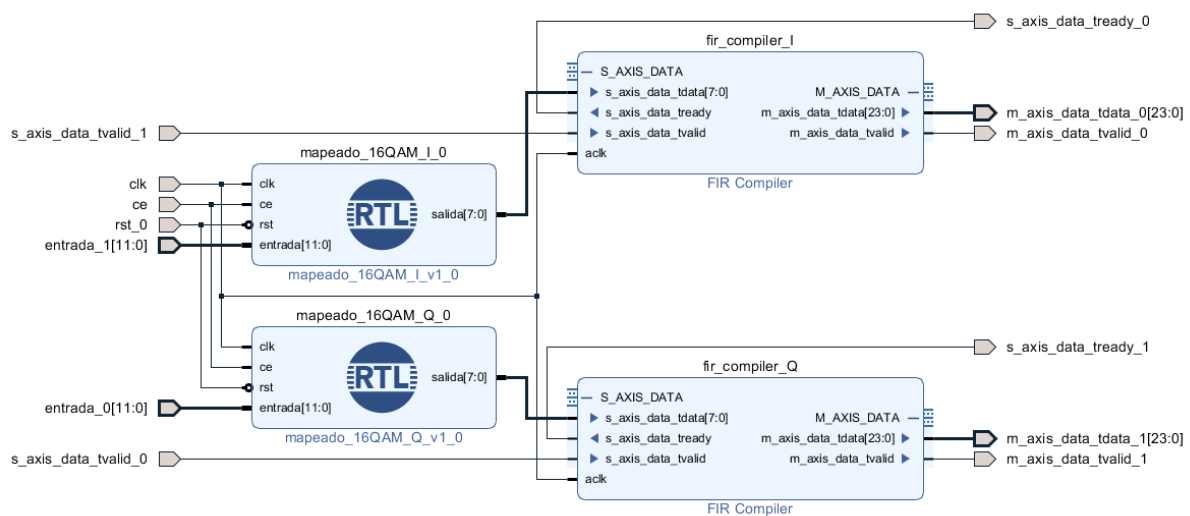


Figura 3.1: Diseño de los bloques mapeado 16-QAM + ZP y filtrado RRC

3.1. Mapeado 16-QAM

Para implementar una modulación 16-QAM (Modulación de Amplitud en Cuadratura) lo primero que se debe realizar es el bloque de mapeado, con el fin de asignar patrones de amplitud y fase a cada símbolo y proporcionar una mayor eficiencia espectral. Para ello, se recibe una señal de entrada de 12 bits, obtenida a partir de la lectura de los 200 valores almacenados en la memoria FIFO.

Como se había descrito en el Capítulo 2, la lectura se produce a una frecuencia de 2MHz. Según las especificaciones de este proyecto, la salida del mapeado debe funcionar a una frecuencia de 6MHz, es decir, el triple de la anterior porque por cada valor a la entrada de 12 bits se obtendrán 3 símbolos 16-QAM. En otros términos, se tratarán los bits de entrada de 4 en 4 para mapear cada símbolo 16-QAM.

El mapeado de cada símbolo 16-QAM supone la generación a la salida de dos caminos independientes: uno para la componente en Fase (I) y otro para la de Cuadratura (Q). Cada símbolo es de 3 bits con signo y puede tomar los valores $(-3, -1, +1, +3)$ como se puede visualizar en la Figura 3.2.

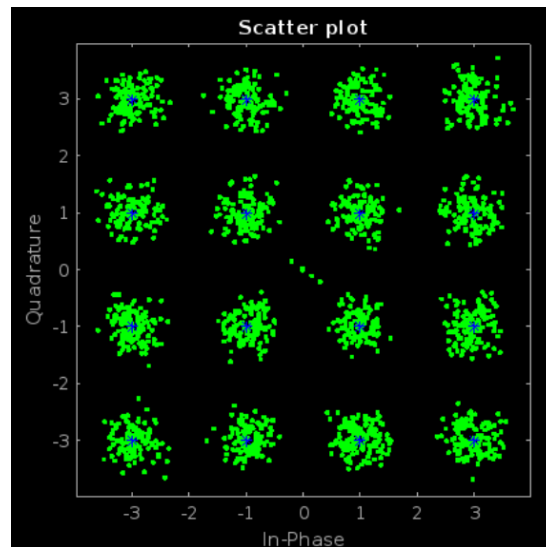


Figura 3.2: Constelación 16-QAM

En este caso, se han decidido ajustar las tablas Q e I para establecer una codificación Gray. Esta se trata de una técnica específica que garantiza que solo un bit cambie entre dos valores consecutivos. Además, consigue minimizar la posibilidad de errores producidos por cambios de múltiples bits en la transmisión. A continuación, se puede visualizar la definición de cada una de las tablas:

```

1 type Array3Bit is array (0 to 15) of STD_LOGIC_VECTOR(2 downto 0);
2 constant tabla_mapeado_Q: Array3Bit :=
3   ("000", "001", "011", "010",
4    "110", "111", "101", "100",
5    "000", "001", "011", "010",
6    "110", "111", "101", "100");

```

Listing 3.1: Definición de la tabla de mapeado Q

```

1 type Array3Bit is array (0 to 15) of STD_LOGIC_VECTOR(2 downto 0);
2 constant tabla_mapeado_I: Array3Bit :=
3   ("000", "000", "001", "001",
4    "011", "011", "010", "010",
5    "110", "110", "111", "111",
6    "101", "101", "100", "100");

```

Listing 3.2: Definición de la tabla de mapeado I

Por tanto, para mapear cada

3.2. Zero Padding

3.3. Bloque de filtrado pulse shaping - RRC

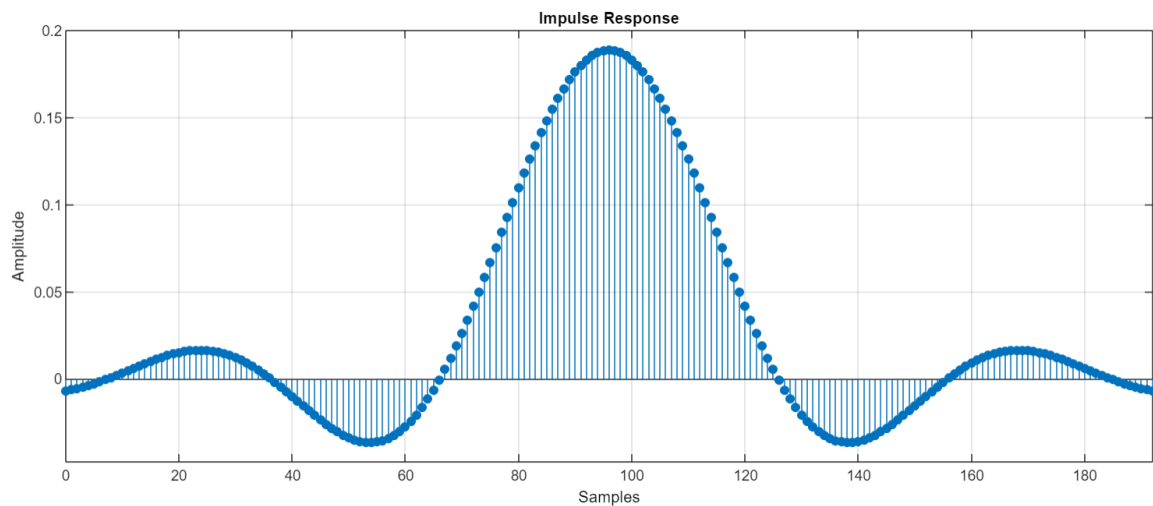


Figura 3.3

Capítulo 4

Bloque DDS

Generación de las sinusoides coseno/-seno mediante bloque DDS a 576 MHz

Capítulo 5

Conclusiones

Bibliografía