Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ingeniería Eléctrica
IEE2463 Sistemas Electrónicos Programables

Ayudantía 04 2025-2

ZYBO Z7-10

Ayudante: Juan Ignacio Lorca juan.lorca@uc.cl

Prof. Dr.-Ing. Félix Rojas - felix.rojas@uc.cl

---

1. Objetivos de la Ayudantía

La siguiente ayudantía ejercitará el uso de los periféricos (IP Cores) ILA y VIO para visualizar sinusoides ($\sin(2\pi fn)$ y $\cos(2\pi fn)$) generadas internamente por la tarjeta Zybo Z7. Para ello, se empleará el IP Core *Block Memory Generator* obtenido desde el *IP Catalog* y configurado como *Read-Only Memory* (ROM), en la cual se almacenarán muestras discretas de un ciclo de las funciones seno y coseno. Dichas memorias serán instanciadas en código VHDL y leídas de manera secuencial a una frecuencia estable de reloj, de modo de reconstruir las señales periódicas.

En particular, la ayudantía se enfocará en los siguientes objetivos:

- Seleccionar y configurar un IP Core *Block Memory Generator* como ROM a partir del *IP Catalog* de Vivado.
- Instanciar el IP Core generado dentro de un módulo VHDL mediante la declaración de componentes.
- Implementar un mecanismo de direccionamiento secuencial para la lectura periódica de muestras de la ROM.
- Visualizar y verificar las señales generadas a través de los IP Cores de depuración ILA y VIO.

2. Actividades previas a la Ayudantía

Para el desarrollo de esta actividad, **no se utilizará el proyecto base ejecutado en el video de la ayudantía 04, pero sí es necesario haber visto este último (para comprender el funcionamiento de los bloques ILA y VIO)**. En cambio, esta sección se encargará de guiarlos para que puedan cumplir con los objetivos de enunciado.

2.1. Pasos a seguir

1. Descargar el archivo `.coe` disponible en la sección de ayudantías en los módulos de Canvas. Este archivo contiene $N = 256$ muestras de un ciclo de sinusoide codificadas en 8 bits con signo, en formato hexadecimal.

2. Crear un nuevo proyecto en Vivado asociado a la tarjeta Zybo Z7-10. Si la tarjeta no aparece en la lista de *boards*, actualizar los *board files* (como se muestra en la Ayudantía 00) y reiniciar Vivado. No olvidar añadir el archivo `.xdc` con los *constraints* correspondientes a la tarjeta.

3. Integrar en el proyecto el IP Core *Block Memory Generator* disponible en el *IP Catalog*. Configurarlo como ROM de un puerto y cargar el archivo `.coe` previamente descargado. Procure en la generación*output products* **definir una síntesis global** (seleccionando *Global* y después *Apply*) y **no generar los** *output products*, ya que esto puede derivar en errores de síntesis o al querer incluir un empaquetado de código a un *Block Design*. El procedimiento se ilustra a continuación:

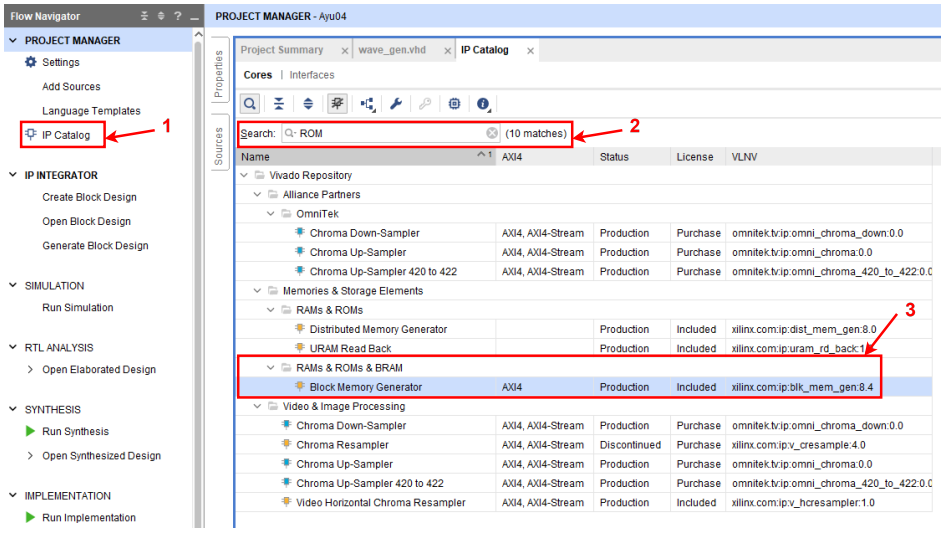

Figura 1: Selección del IP Core Block Memory Generator en el IP Catalog

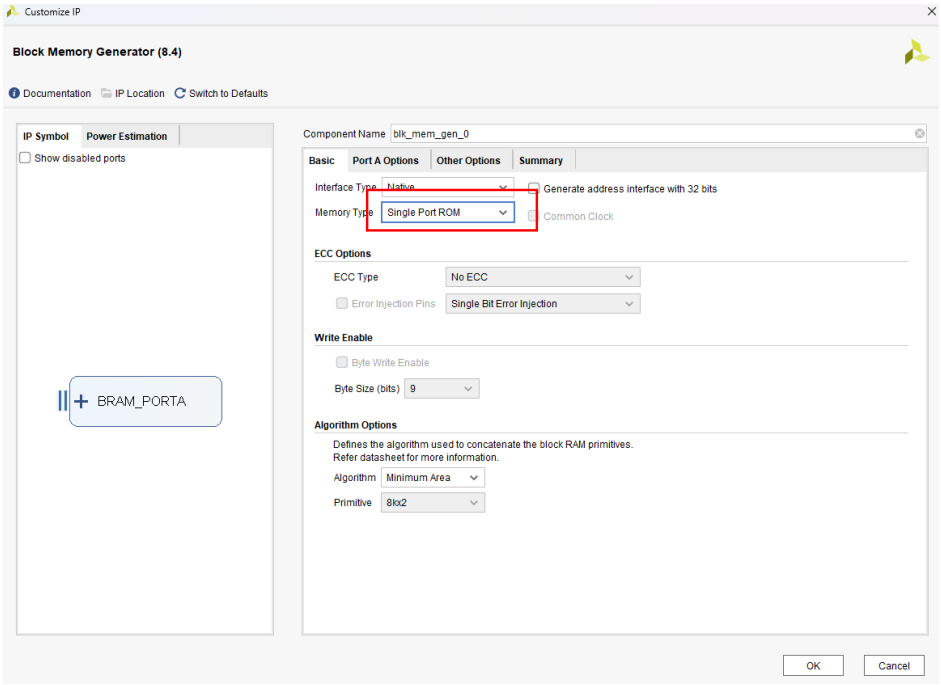

Figura 2: Configuración del IP como memoria ROM de un puerto

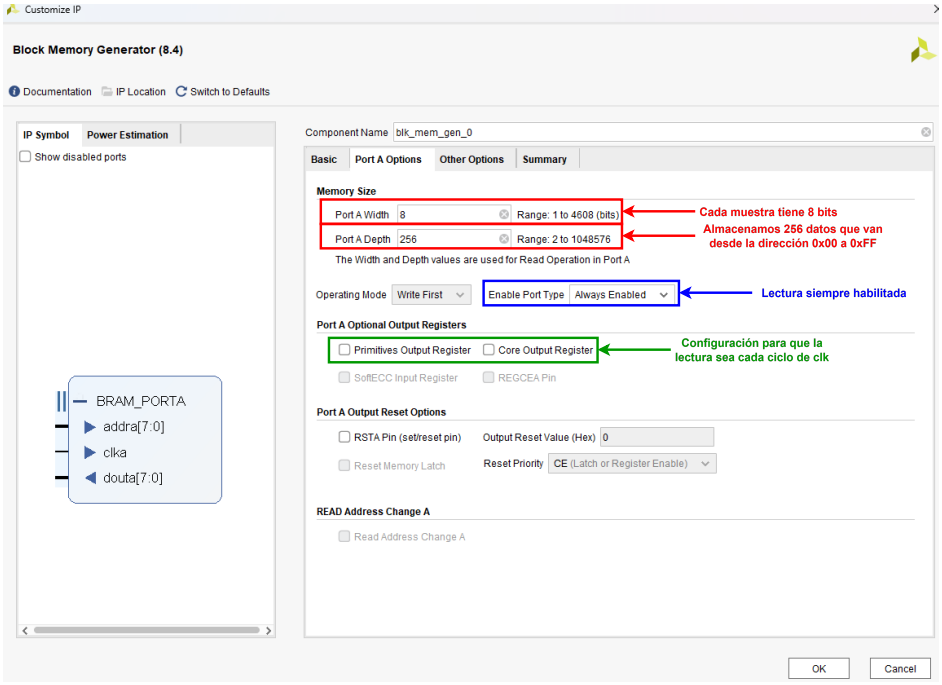

Figura 3: Configuración del tamaño de la ROM

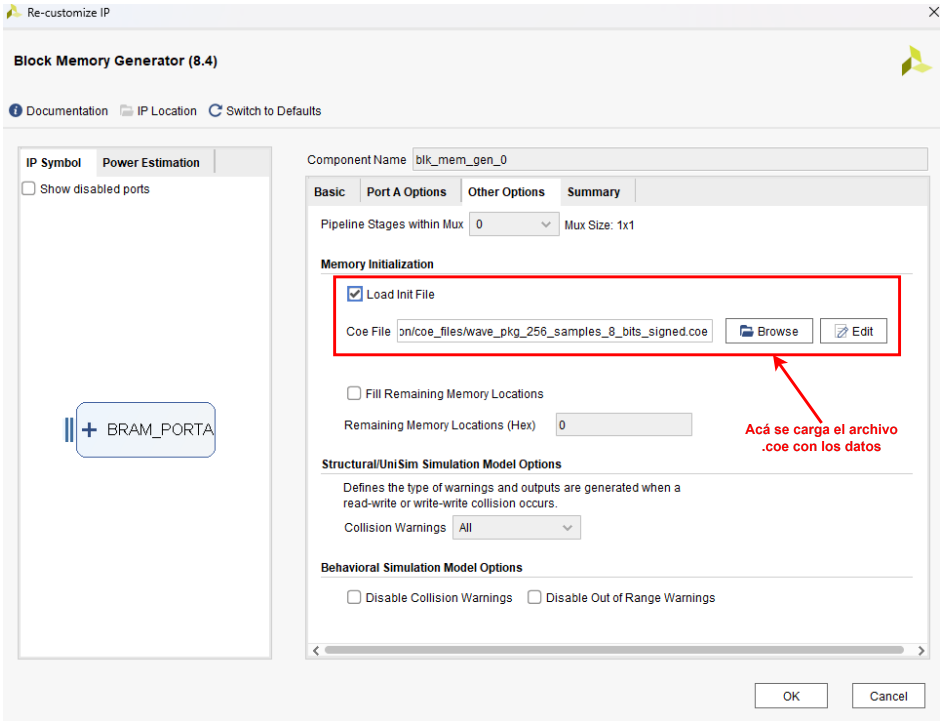

Figura 4: Carga del archivo `.coe` en la memoria

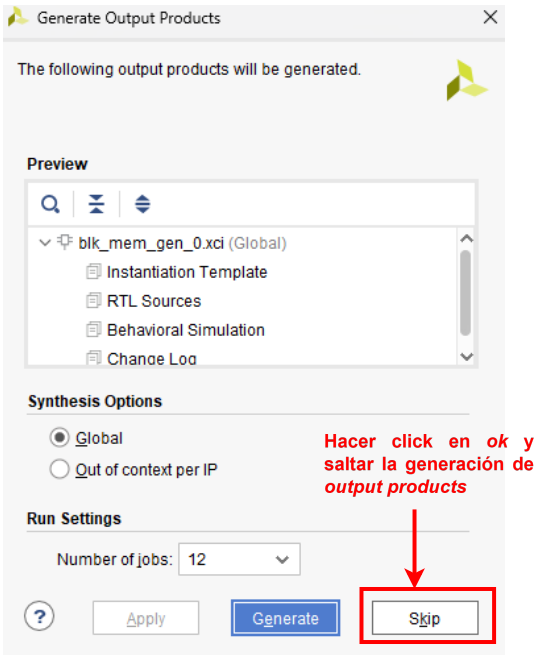

Figura 5: Confirmación de la configuración IP

4. Incluir en el proyecto el archivo `wave_gen.vhd`, disponible en Canvas también. Este módulo permite leer el contenido almacenado en la ROM y generar las señales sinusoidales `sin[7:0]` y `cos[7:0]`.

5. Crear un *Block Design* que incorpore los siguientes IP Cores:

   - ***Clocking Wizard*:** configurado con una frecuencia de salida de 100 MHz.
   - ***VIO*:** con dos puertos de entrada de 8 bits y un puerto de salida de 1 bit.
   - ***ILA*:** con dos puertos de entrada de 8 bits.
   - ***Utility Vector Logic*:** configurado para realizar la operación booleana AND, con dos entradas de 1 bit y una salida de 1 bit.

6. Agregar el módulo `wave_gen.vhd` al *Block Design*. Para ello, hacer click derecho sobre el archivo en el explorador de fuentes y seleccionar *Add Module to Block Design*.

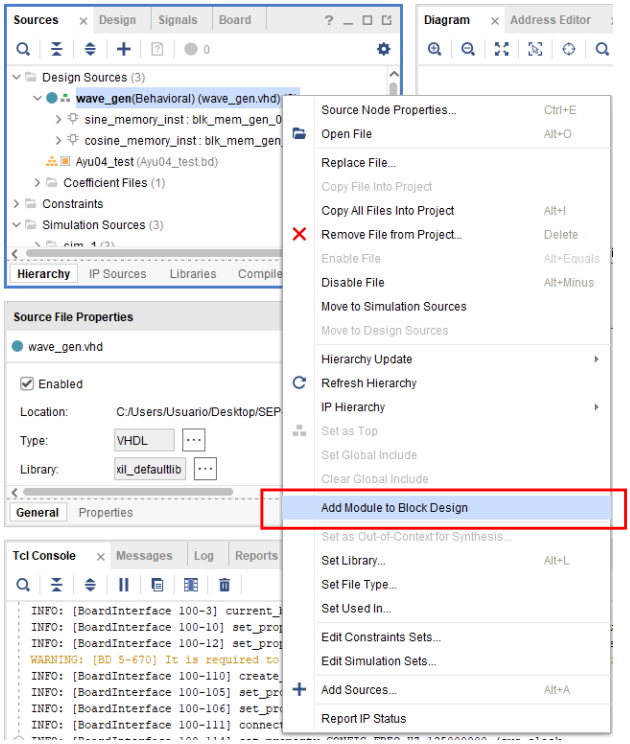

Figura 6: Método para agregar código empaquetado a un *Block Design*

7. Conectar el reloj externo de la tarjeta al *clock* del diseño, de modo que se pueda realizar el mapeo de puertos correspondiente en el archivo de *constraints*.

8. Una vez completados los pasos anteriores, el diseño debería contar con un diagrama de bloques similar al siguiente:

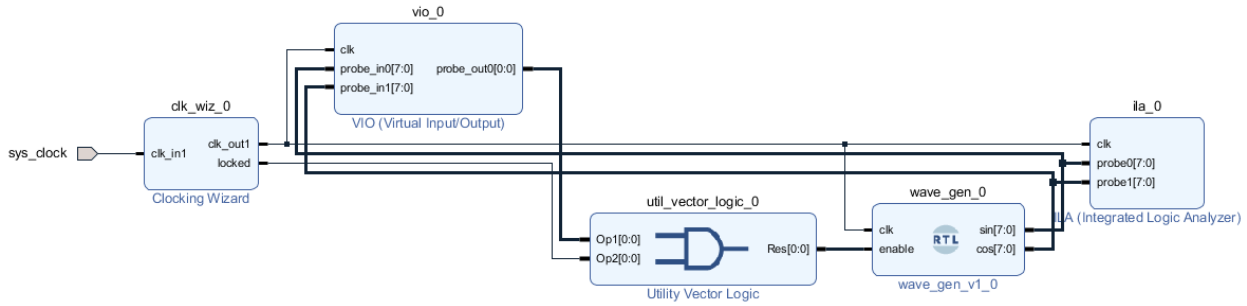

Figura 7: Diagrama de bloques del proyecto con los IP Cores integrados

Con los pasos anteriores concluidos, se puede proceder a generar el *wrapper* del diagrama y fijarlo como *top file* para la síntesis. Posteriormente, sintetizar, implementar y generar el *bitstream* para programar la tarjeta.

Nota: no se olvide de utilizar la interfaz de *VIO* para accionar el *enable* del bloque `wave_gen_0` proveniente de la señal `probe_out0`.

2.2. Resultados

A continuación se presentan las formas de onda capturadas en el ILA para las señales `sin[7:0]` y `cos[7:0]`.

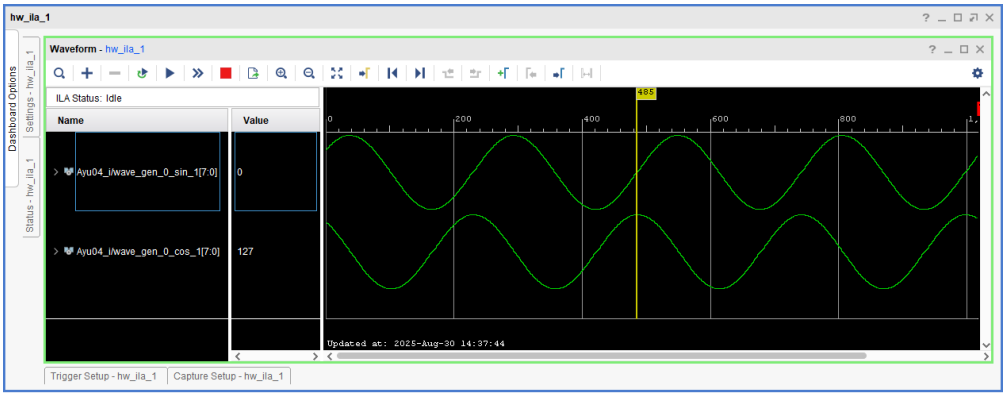

Figura 8: Visualización en el ILA de las señales sinusoidal y cosenoidal.

Nota: para lograr que las formas de onda se visualicen de manera similar a una señal analógica en el ILA, es necesario realizar los siguientes ajustes:

1. Hacer click derecho sobre la señal y cambiar la opción *Waveform Style* a *Analog*.
2. Cambiar la opción *Radix* a *Signed Decimal*, de modo que se interpreten correctamente los valores codificados en complemento a dos.

3. Actividades durante la Ayudantía

El ejercicio propuesto para esta ayudantía consiste en incorporar una señal triangular al bloque utilizado para la generación de formas de onda.

La definición discreta de una señal triangular periódica de N muestras y amplitud A es:

$$x[n] = \begin{cases} A - \dfrac{4A}{N}\, n, & 0 \le n < \frac{N}{2}, \\ -A + \dfrac{4A}{N}\left(n - \frac{N}{2}\right), & \frac{N}{2} \le n < N, \end{cases} \tag{1}$$

con extensión periódica

$$x[n+N] = x[n].$$

La única restricción es que la señal debe provenir desde un bloque instanciado en el código principal que genera las ondas. Por ejemplo, se puede:

- Implementar una memoria ROM inicializada con un archivo `.coe` que contenga las muestras de la triangular (similar a lo hecho con las sinusoides).
- O bien, generar un módulo en VHDL que implemente la lógica de la ecuación 1 y luego instanciar dicho módulo en el diseño principal.

Nota importante: la ecuación 1 toma valores negativos en el segundo semiciclo. Para poder visualizarlos correctamente en el ILA o VIO, la salida debe codificarse en formato *two's complement* (complemento a 2). De este modo, el ILA mostrará la forma triangular en el rango $[-A, A]$.

En la ayudantía se verificará que los estudiantes hayan completado correctamente ambas etapas de la actividad (la guiada y la propuesta). Para ello, deberán mostrar en el ILA las formas de onda correspondientes a `sin[7:0]`, `cos[7:0]` y `triag[7:0]` (señal triangular), además de ser capaces de explicar cómo se determina la frecuencia fundamental de las señales visualizadas.