Introduction to Hierarchical Trigger Systems: From FPGA Level-1 to High-Level Software

Álvaro Navarro Tobar
Javier Prado
Cristina F. Bedoya
Santiago Folgueras

Oviedo
25 de junio de 2026

Índice

1. Links para la práctica

1. Jupyter notebook. Haz una copia!!!! Link
2. Repo HLT. Clónalo a tu PC Link

2. Overview de la práctica

1. Monta y prueba el setup de la práctica.
2. Instala Arduino y el repositorio HLT en tu ordenador.
3. Ejecuta el Jupyter notebook hasta la parte de Analysis.
4. Programa el algoritmo L1 en la FPGA.
    a) Una vez tengas la FPGA programada y el archivo bitstream generado, llámanos para programar la placa
5. Programa el HLT en Arduino

3. Conectando el setup

Solo es necesario conectar un cable USB-C al ESP32. Tanto la FPGA como el ESP32 vienen con el firmware básico ya cargado.

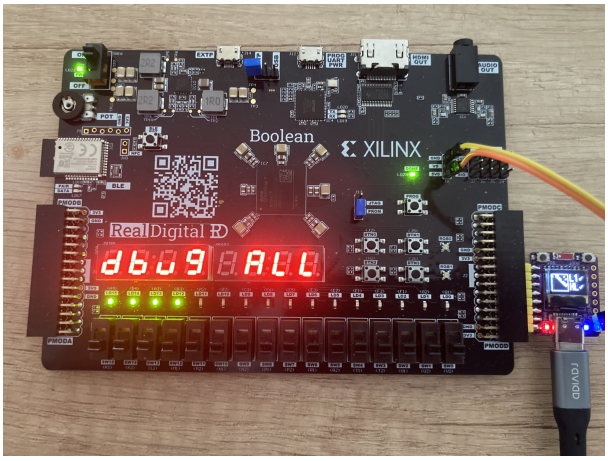

Figura 1: Setup de laboratorio

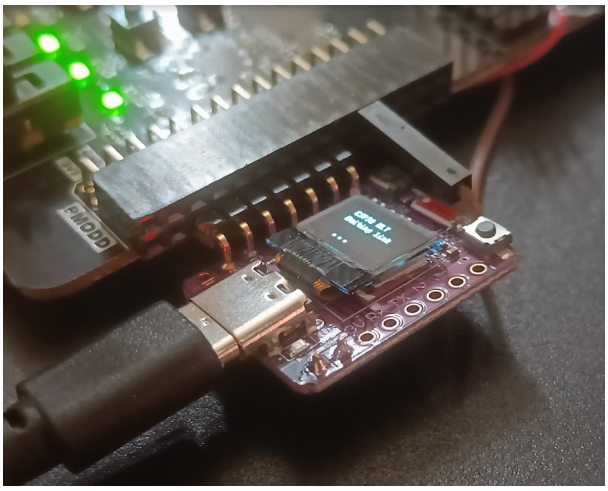

Figura 2: Closeup de la conexión del ESP32 con la FPGA

El primer switch (izquierda) escogerá entre los modos:

- Debug: Al pulsar uno de los botones, enviará una sola muestra para comprobar el funcionamiento de los dos algoritmos: L1T y HLT. Utilizando el resto de switches podrás seleccionar el tipo de partícula que quieres probar: Ruido, muón, alpha, muon de alto momento. alpha+muon y alpha + muon de alto momento.
- Run: Al pulsar el botón, enviará la muestra completa a través de los dos algoritmos y devolverá la eficiencia total de identificación de muones de alto momento. Además, enviará por el puerto serie los datos a analizar.

4. Arduino IDE

Arduino IDE es el entorno de desarrollo integrado oficial para programar placas Arduino y compatibles. Esta guía cubre la instalación y configuración del IDE en su versión 2.x, la más actual y recomendada, así como la configuración necesaria para con la placa ESP32-C3 de Espressif.

4.1. Instalación

Accede a la página oficial de Arduino para descargar la versión más reciente del IDE: https://www.arduino.cc/en/software

Figura 3: Arduino install page

4.1.1. Instalación en Windows

1. Ejecuta el instalador descargado (.exe) como administrador.
2. Acepta el acuerdo de licencia.
3. Selecciona los componentes a instalar (se recomienda mantener los predeterminados).
4. Elige la carpeta de instalación y haz clic en Instalar.
5. Acepta la instalación de drivers USB cuando el sistema lo solicite.
6. Haz clic en Finalizar al completar la instalación.

4.1.2. Instalación en Mac

1. Abre el archivo .dmg descargado.
2. Arrastra el icono de Arduino IDE a la carpeta Aplicaciones.
3. La primera vez que lo abras, confirma que deseas ejecutar la aplicación.

4.1.3. Instalación en Linux

1. Descarga el archivo AppImage.
2. Dale permisos de ejecucion con el comando: chmod +x arduino-ide_*.AppImage
3. Ejecuta el AppImage haciendo doble clic o desde terminal: ./arduino-ide_*.AppImage

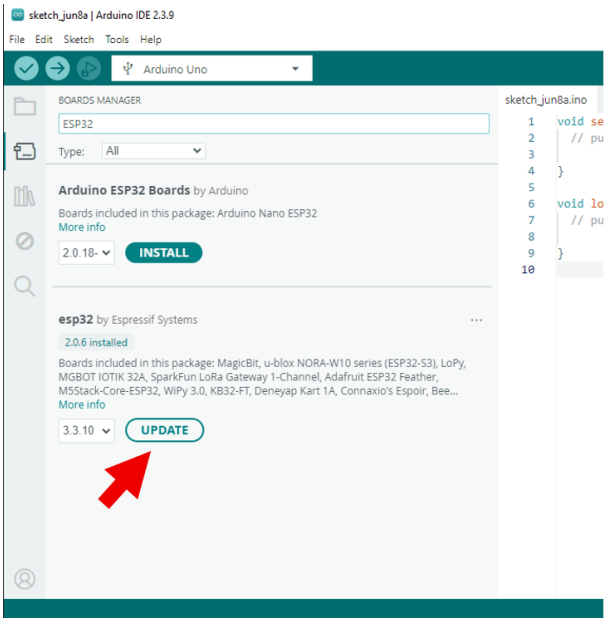

Figura 4: Librería de la placa ESP32C3

4.2. Instalar librería para gestión de lapantalla OLED

1. Ve a Sketch > Include Libraries > Manage Libraries...
2. Escribe 'U8g2' en el cuadro de búsqueda.
3. Localiza el paquete 'U8g2 by oliver' y haz clic en Instalar. (Si no sale nada, ir a la sección de problemas comunes)
4. Espera a que se descargue e instale el paquete (puede tardar varios minutos).

4.3. Instalar dispositivo ESP32

1. Ve a Herramientas > Placa > Gestor de tarjetas.
2. Escribe 'esp32' en el cuadro de búsqueda.
3. Localiza el paquete 'esp32 by Espressif Systems' y haz clic en Instalar. (Si no sale nada, ir a la sección de problemas comunes)
4. Espera a que se descargue e instale el paquete (puede tardar varios minutos).

5. ESP32

5.1. Descripción general

La ESP32-C3 es una placa de desarrollo de Espressif Systems basada en el microcontrolador ESP32-C3, que integra Wi-Fi 802.11 b/g/n y Bluetooth 5.0 (BLE) en un chip de arquitectura RISC-V de 32 bits. Precio $\approx$ 3€!

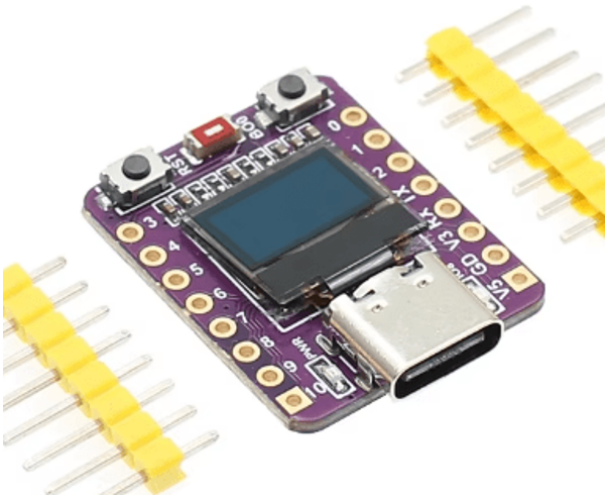

Figura 5: Placa ESP32

Característica	Valor
Microcontrolador	ESP32-C3 (RISC-V 32 bits)
Velocidad de reloj	160 MHz
Memoria Flash	4 MB
SRAM	400 KB
Wi-Fi	802.11 b/g/n (2.4 GHz)
Bluetooth	BLE 5.0
Pines GPIO	22 pines (multifunción)
Pantalla Oled	0.42”

Tabla 1: Caracteristicas ESP32

5.2. Seleccionar la placa ESP32-C3

1. Ve a Herramientas > Placa > ESP32 Arduino.
2. Selecciona ’ESP32C3 Dev Module’
3. Conecta la ESP32-C3 al ordenador y selecciona el puerto en Herramientas > Puerto.
4. Para poder ver el puerto serie: Herramientas > USB CDC on boot: Enabled

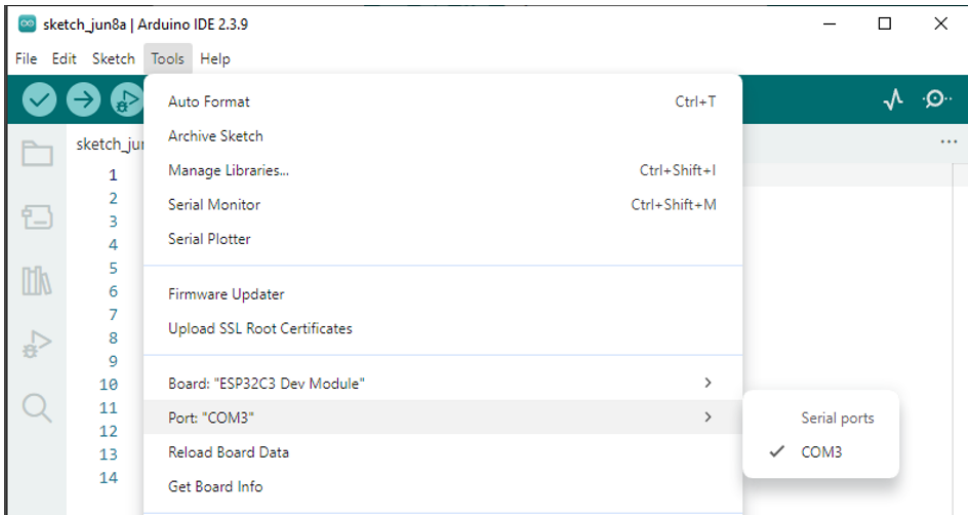

Figura 6: Seleccionar puerto de comunicación

5.3. Programacion del ESP32

El siguiente código hace parpadear el LED integrado de la placa ESP32-C3. Pégalo en el IDE para comprobar el correcto funcionamiento.

```
const int LED_PIN = 8;  // GPIO8 en ESP32-C3

void setup() {
  Serial.begin(115200);
  pinMode(LED_PIN, OUTPUT);
  Serial.println("ESP32-C3 iniciado correctamente");
}

void loop() {
  digitalWrite(LED_PIN, HIGH);  // LED encendido
  Serial.println("LED ON");
  delay(500);
  digitalWrite(LED_PIN, LOW);   // LED apagado
  Serial.println("LED OFF");
  delay(500);
}
```

1. Verifica que la placa y el puerto están correctamente seleccionados.
2. Haz clic en Cargar.
3. Si la carga falla, mantén pulsado el botón BOOT de la placa mientras el IDE intenta conectar, luego suéltalo.
4. Abre el Monitor Serie (Herramientas > Monitor Serie) a 115200 baudios para ver los mensajes.

Deberías ver el led azul parpadear y un mensaje en el puerto serie de comunicación.

Figura 7: Cómo cargar el programa en el ESP32

5.4. Problemas comunes

Problemas comunes	
El puerto COM no aparece	Instala el driver USB correcto (CH340 o CP2102) sección 6.1. Reinicia el ordenador.
Error 'Failed to connect'	Mantén pulsado el botón BOOT al iniciar la carga. Reduce la velocidad de Upload a 115200.
Compilación falla (ESP32)	Comprueba que el paquete esp32 está instalado y actualizado en el Gestor de tarjetas.
Monitor Serie muestra basura	Asegúrate de que la velocidad del monitor (baudrate) es 115200
Error de permisos en Linux	Ejecuta: sudo usermod -a -G dialout $USER y cierra sesión.

Tabla 2: Problemas comunes

6. Programación del L1 trigger

Ordenador a utilizar					
G0	T1	tical43.ific.uv.es	G1	T1	tical43.ific.uv.es
G0	T2	tical43.ific.uv.es	G1	T2	tical43.ific.uv.es
G0	T3	tical43.ific.uv.es	G1	T3	tical43.ific.uv.es
G0	T4	tical53.ific.uv.es	G1	T4	tical53.ific.uv.es
G0	T5	tical53.ific.uv.es	G1	T5	tical53.ific.uv.es

Tabla 3: Ordenador a utilizar

1. Conéctate por ssh a cnid@tical43.ific.uv.es o cnid@tical53.ific.uv.es con contraseña: ######. Habrá una carpeta del tipo CNID_GX_TX
2. Dirígete a la carpeta /home/cnid/[TUCARPETA]/cnid_trigger_practicum_materials/fpga/src
3. El archivo que tienes que modificar es l1t_algorithm.vhd utiliza VIM o tu programa favorito para editarlo.

   El lenguaje de programación es VHDL, utiliza el cheatsheet para ayudarte a programarlo. Existe un archivo l1t_extra_algorithms.vhd en el que existe una solución. Úsalo si estás atascado... no hagas trampas...

Esta será la entrada a tu módulo de L1 trigger:

- **CLK** será la señal de sincronización con la que llegarán las señales al módulo.
- **event_image_in** es tu señal de entrada. En este caso, una matriz de 12x12 (11 downto 0)(11 downto 0)
- **l1t_accept_out** es la señal de aceptación o rechazo. 1=>Aceptado 0=>Rechazado

Para programar la FPGA primero debemos generar el archivo de configuración. Para automatizar la síntesis e implementación del archivo, vamos a crear un archivo TCL que automatice este paso. Para ello:

1. dirígete a la carpeta /home/cnid/[TUCARPETA]/cnid_trigger_practicum_materials/ y ejecuta el código de abajo en el terminal. El archivo build.tcl se debería haber creado.

```
cat > build.tcl << 'EOF'
open_project /home/cnid/[TUCARPETA]/cnid_trigger_practicum_materials/fpga/project/
    project.xpr

reset_run synth_1
launch_runs synth_1 -jobs 4
wait_on_run synth_1

if {[get_property PROGRESS [get_runs synth_1]] != "100%"} {
    error "ERROR: la síntesis falló"
}

launch_runs impl_1 -to_step write_bitstream -jobs 4
wait_on_run impl_1

if {[get_property PROGRESS [get_runs impl_1]] != "100%"} {
    error "ERROR: la implementación/bitstream falló"
}

puts \"\textexclamdown Bitstream generado correctamente!\"
EOF
```

Para ejecutar la construcción del proyecto, ejecuta el siguiente código:

```
source /opt/Xilinx/Vivado/2023.2/settings64.sh
vivado -mode batch -source build.tcl -log build.log -journal build.jou
```

Este proceso puede tardar unos minutos. En este tiempo, el programa está convirtiendo tu diseño (código) a un circuito digital (síntesis) y colocándolo en la FPGA (implementación). El archivo para programar la fpga se guardará en:

```
/home/cnid/[TuCarpeta]/cnid_trigger_practicum_materials/fpga/project/project.runs/impl_1
    /top.bit
```

Descárgalo a tu ordenador utilizando scp, desde tu ordenador, ejecuta el siguiente código (acuerdate de cambiar el ordenador al que estás conectado (tical43 o tical53), TuCarpeta y la carpeta de destino en tu ordenador

```
scp cnid@tical53.ific.uv.es:/home/cnid/[TuCarpeta]/cnid_trigger_practicum_materials/fpga
    /project/project.runs/impl_1/top.bit [Carpeta destino]
```

Si no tienes VivadoLab, llamanos y programaremos tu fpga desde nuestro ordenador. Si tienes vivadoLab ve a la seccion 8.2

7. Programación del ESP32

Si todavía no has instalado el software Arduino con las otras instrucciones, porfavor, hazlo ahora. Clona el repositorio de gitlab si aun no lo has hecho. Para abrir el archivo: File > Open

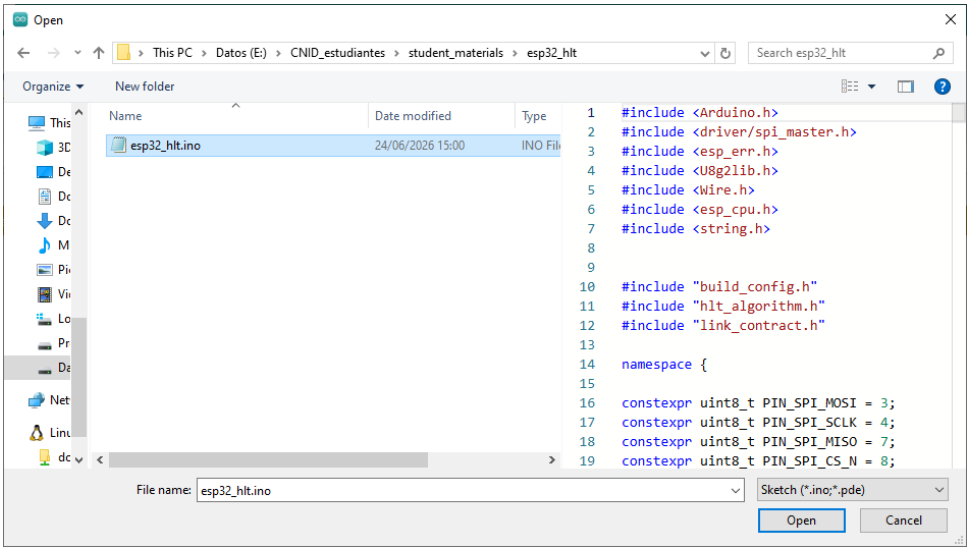

Figura 8: Cargar proyecto

Cuando cargues el proyecto en arduino, se abriran varios archivos como enseña la figura 9. La función

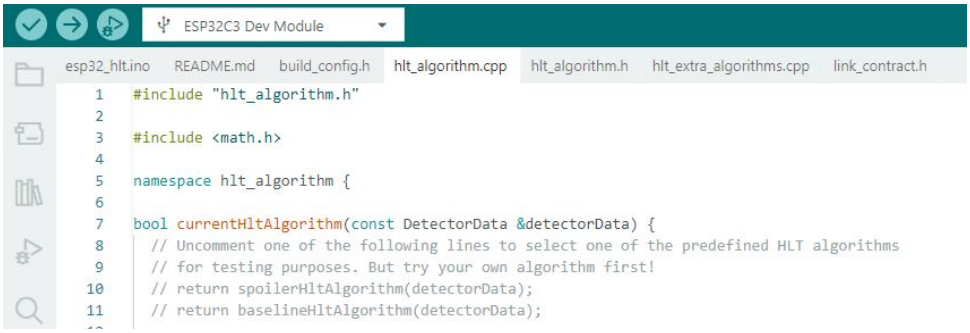

Figura 9: Archivos y funciones HLT

que debes editar es *currentHltAlgorithm* , que retornará 1 si la señal se acepta por el HLT o devolverá 0 si la rechaza.

De nuevo, tenemos dos funciones ya programadas que podéis utilizar para comparar con vuestro L1:

- **baselineAlgorithm**: Una función que funciona, pero no es perfecta.
- **spoilerHLT**: La mejor función que hemos conseguido, a ver si la mejoras...

Cuando ejecutes tu código, abre el monitor serial para ver los resultados de forma detallada. Esta información la puedes pegar en jupyter para ver las estadisticas.

8. Extra

8.1. Instalación de VivadoLab

Dirígete a https://www.amd.com/en/support/downloads/adaptive-socs-and-fpgas/development-tools/2023-2.html y descarga el archivo correspondiente (Linux o Windows) Tendrás que crearte una cuenta en amd:

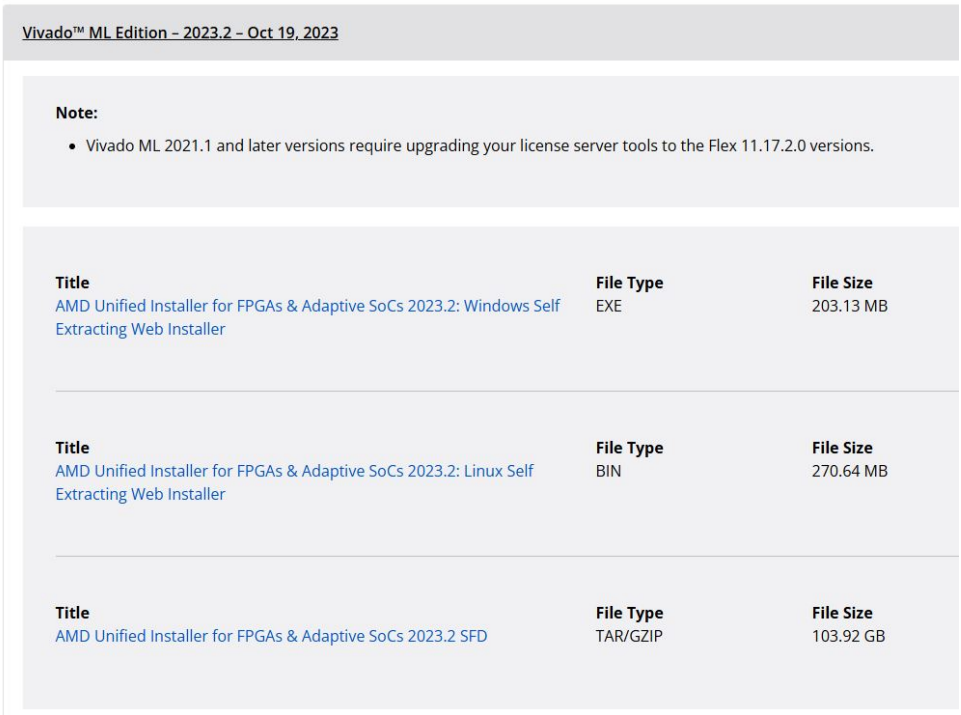

Figura 10: Instaladores de vivado

Selecciona VIVADO LAB EDITION! Y continua con la instalación

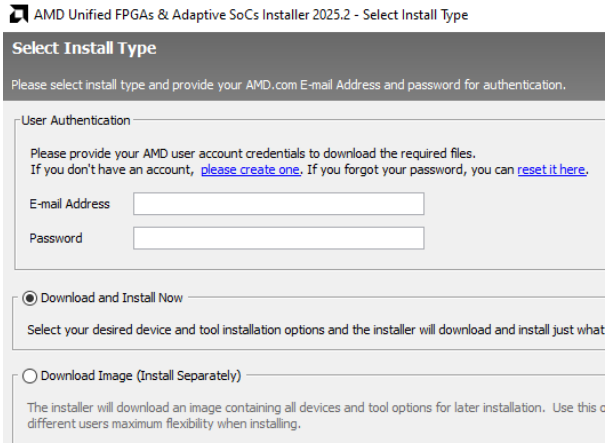

Figura 11: Enter Caption

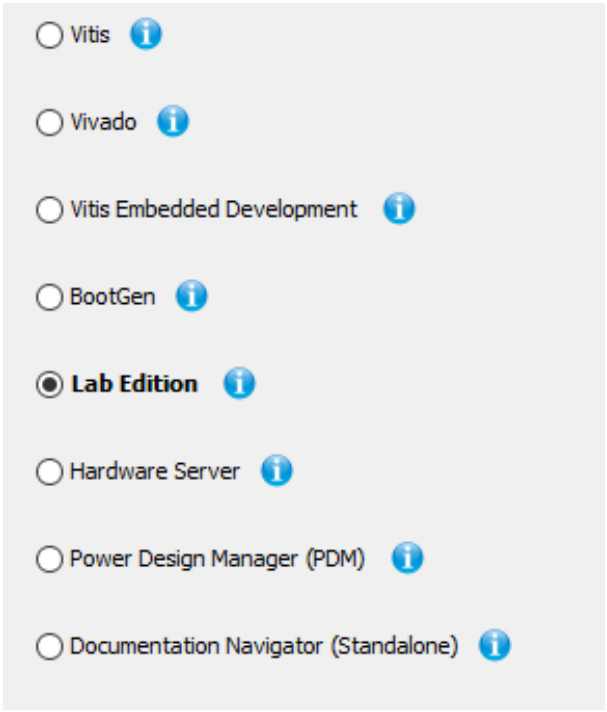

Figura 12: Edición de vivado necesaria

8.2. Setup de programación FPGA

Una vez tengas tu archivo bitstream, abre vivadoLab y selecciona **Open hardware manager**

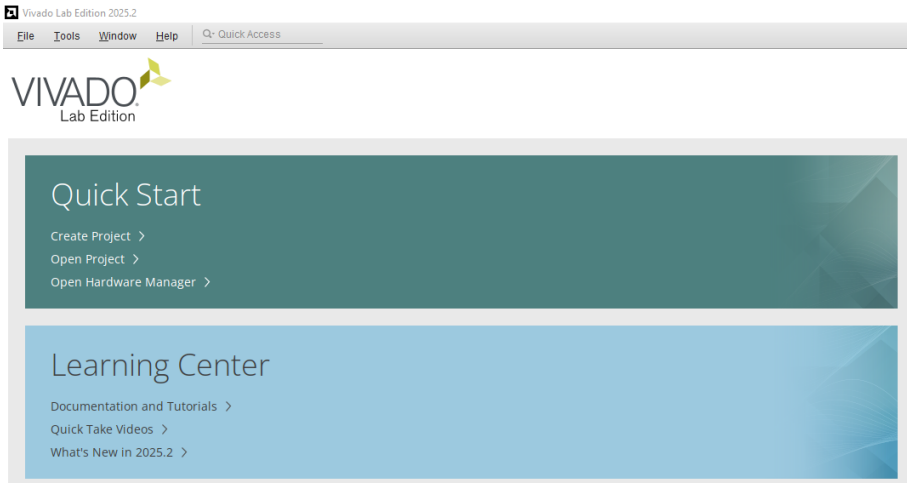

Figura 13: VivadoLab start page

9. Cheatsheet de programación VHDL

9.1. Variables y asignación

```
-- STD_LOGIC type
 signal my_std_logic : STD_LOGIC;
 my_std_logic <= '1';

-- STD_LOGIC_VECTOR type
 signal my_std_logic_vector : STD_LOGIC_VECTOR(3 downto 0);
 my_std_logic_vector <= "1101";

-- UNSIGNED type
 signal my_unsigned : UNSIGNED(7 downto 0);
 my_unsigned <= to_unsigned(255, 8);

-- SIGNED type
 signal my_signed : SIGNED(7 downto 0);
 my_signed <= to_signed(-42, 8);

-- INTEGER type
 signal my_integer : INTEGER;
 my_integer <= 123;

-- BOOLEAN type
 signal my_boolean : BOOLEAN;
 my_boolean <= TRUE;

-- STRING type
 signal my_string : STRING(1 to 5);
 my_string <= "Hello";

-- CHARACTER type
 signal my_character : CHARACTER;
 my_character <= 'A';

-- Float
 signal my_sfixed : SFIXED(7 downto -4);
 my_sfixed <= to_sfixed(2.5, my_sfixed);

-- Composite types
type custom_array is array (0 to 7) of std_logic; --Matriz de bits
signal custom_signal : custom_array;
```

9.2. IF-ELSE-ELSIF

```
if condition1 then
    -- Code to execute when condition1 is true
  elsif condition2 then
    -- Code to execute when condition2 is true
  else
    -- Code to execute when none of the conditions are true
end if;
```

9.3. With

```
with selector_signal select
  output_signal <=
    value1 when condition1,
    value2 when condition2,
others => default_value;
```

9.4. Case

```
case selector is
    when value1 =>
        -- Code for value1
    when value2 =>
        -- Code for value2
    when others =>
        -- Default code
end case;
```

9.5. For loops

```
for i in 0 to 7 loop
    -- Loop body
 end loop;
```

9.6. While

```
while condition loop
-- Loop body
end loop;
```