**DISEÑO DE SISTEMAS DIGITALES**

**AVANCE #2**

**TEMA DE LA ACTIVIDAD:** Creación de un sistema embebido para el control de un panel solar por medio de la construcción de un PID en un procesador NIOS II

**Materiales y Herramientas**

* Software Quartus 16.1 versión Estándar.
* Software Eclipe Kepler.
* Tarjeta de desarrollo DE10-Standard.

INTRODUCCIÓN

PID. – Un controlador PID es empleado generalmente en una gran variedad de control de sistemas debido a que presenta una estructura sencilla y de gran versatilidad, el diseño del controlador se basa en las características de cada componente del mismo, La componente “P” de proporcionalidad permite realizar la corrección del error durante el proceso, la componente integral “I” permite decrementar el error estacionario que es provocado por el proporcional, y el Derivador “D” que la corrección del error se ejecute en el menor tiempo posible.[

MULTIPROCESADORES. – Tiene como característica principal ejecutar diferentes operaciones o procesos al mismo tiempo, una de las ventajas de emplear varios procesadores a tareas específicas es el incremento de la eficiencia de la velocidad de respuesta a la transferencia de información

LENGUAJE C. – Es un lenguaje que permite la flexibilidad al momento de programar, debido a que es un lenguaje estructurado y permite declaración de subrutinas dentro de la estructura, además la cantidad de datos que permite manejar lo hace idóneo para enfrentar las diversas complejidades de un proyecto a realizar.

PUERTOS GPIO. – son los puertos destinados para entrada y salida de información, se va a emplear los puertos para enviar la información del PWM para poder regular el control de giro del panel solar conforme a la incidencia de luz.

PUERTOS PARALELOS

DESCRIPCION DE LA PRACTICA. – Se realizará el diseño de la arquitectura, así como la implementación del sistema embebido que será construido en la FPGA, y por medio del software Eclipse para la programación de los procesadores NIOS II

**PROCEDIMIENTO:**

1. Proceda a crear una carpeta en el escritorio con el nombre PID. A continuación, copie y pegue los archivos que se encuentran en la carpeta compartida.
2. Ejecute el programa **Quartus Prime 16.1 Standard Edition** que se localiza en el escritorio y se representa por el icono .
3. En la ventana principal de Quartus proceda a crear un nuevo proyecto seleccionando la opción **File🡪 New Project Wizard** tal como se muestra en la Figura 1.



Figura 1: Ventana para la creación de un proyecto en Quartus Prime

1. Aparecerá la ventana de la Figura 2, donde se requiere escoger la ruta donde se creará el proyecto, la ruta designada será la carpeta PID creada en el paso 1 situada en el escritorio. En la siguiente opción se escoge el nombre del proyecto PID, tal como se muestra en la Figura 2. Clic en **Next**

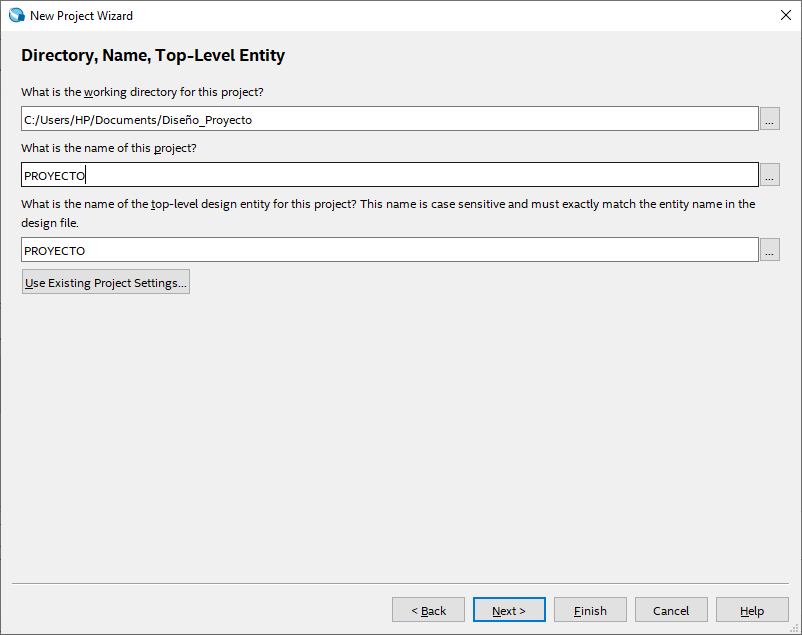


Figura 2: Selección del nombre de proyecto

1. Se visualizará la siguiente ventana donde se escogerá la opción de Proyecto vacío (**Empty project**) como se muestra en la Figura 3. Luego seleccione **Next**.

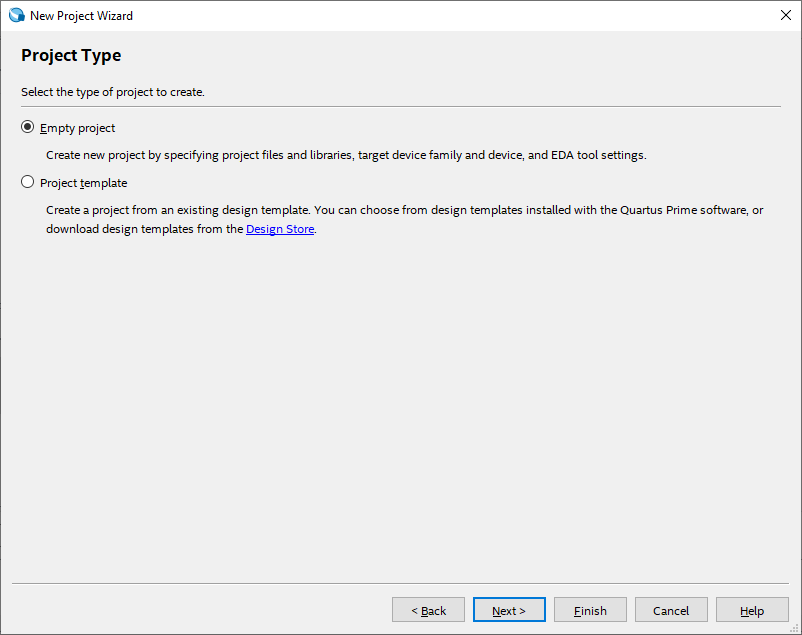


Figura 3: Selección del tipo de proyecto

1. A continuación, se mostrará la ventana de la Figura 4, que corresponde a la adhesión de archivos. Como no existen archivos previos para agregar al proyecto se da clic en **Next.**

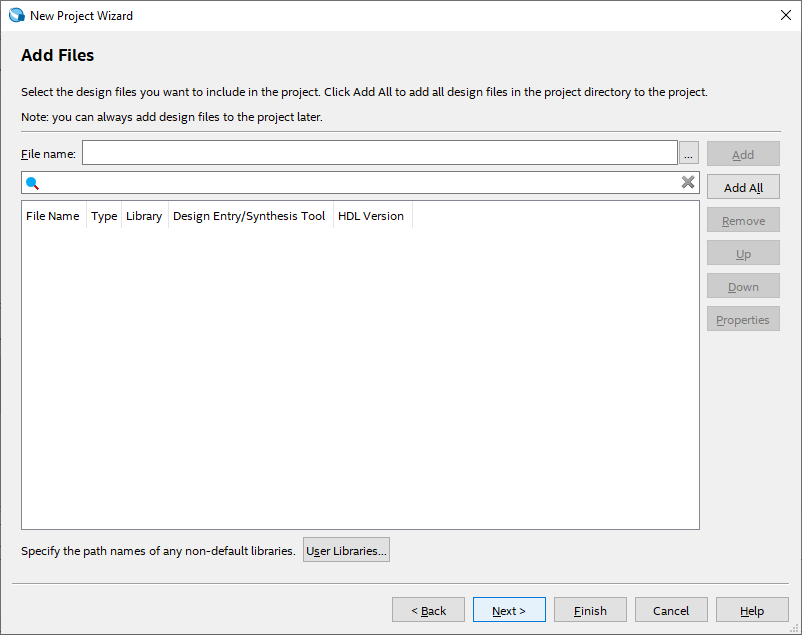


Figura 4: Ventana para adhesión de archivos

1. Aparecerá la venta de la Figura 5, que corresponde a las familias de chips FPGA, donde se escogerá la familia **Cyclone V (E/GX/GT/SX/SE/ST**) y el chip FPGA **SoC** **5CSXFC6D6F31C6**. Clic en **Next.**



Figura 5: Ventana de Familia y modelo del Chip FPGA SoC

1. Aparecerá una ventana que corresponde a las herramientas de Simulación tal como se muestra en la figura 6, debido a que no se realizará simulación en este proyecto no se selecciona ninguno y se da clic en **Next**

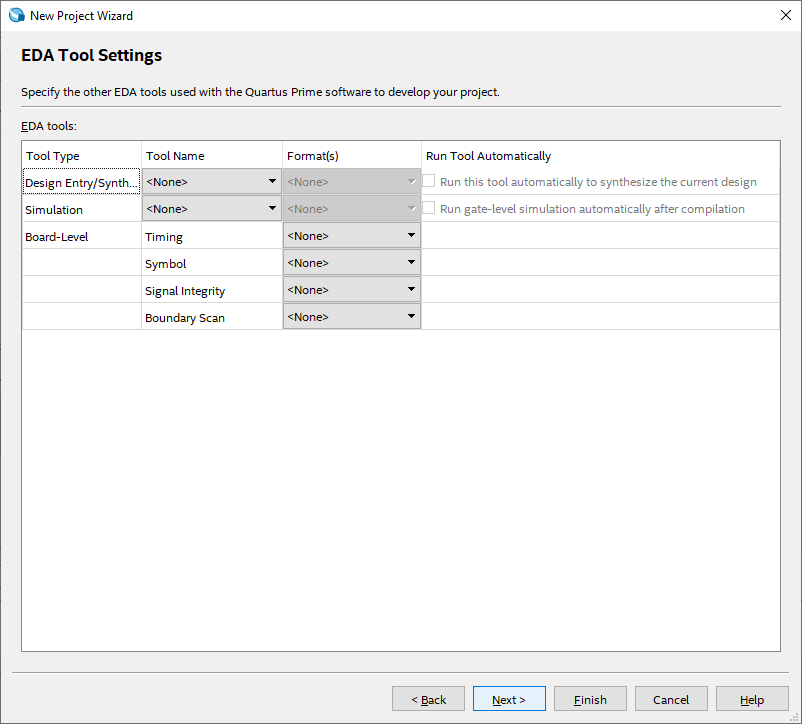


Figura 6; Ventana de herramientas de simulación

1. Una vez concluida la creación de proyecto, se mostrará la ventana **Summary** que corresponde a las especificaciones que hemos dado a nuestro proyecto, clic en **finish**.

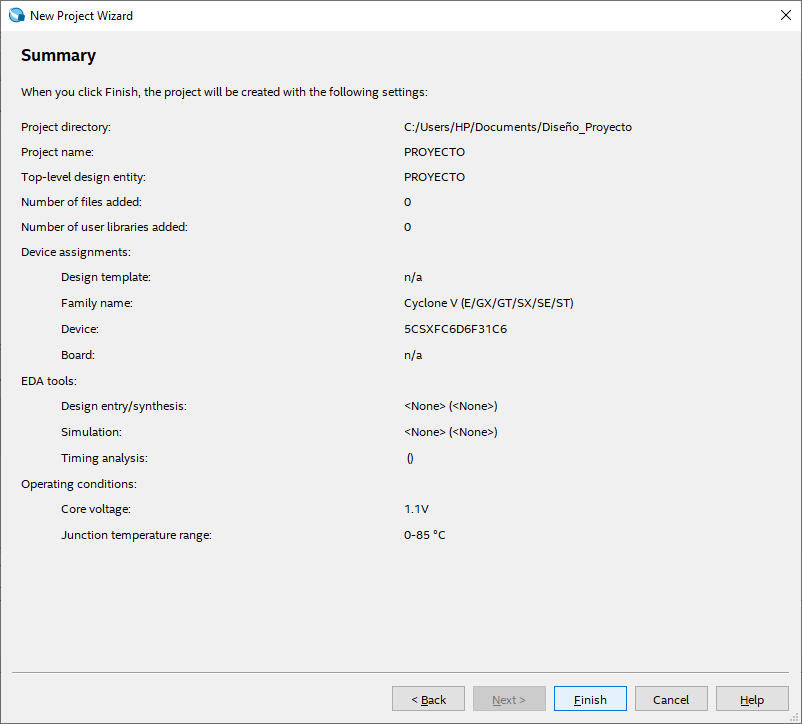


Figura 7: Sumario de proyecto

1. En la ventana principal de Quartus nos dirigiremos a la barra de herramienta donde seleccionaremos Tools, y después escogeremos Qsys, como se observa en la Figura 7.

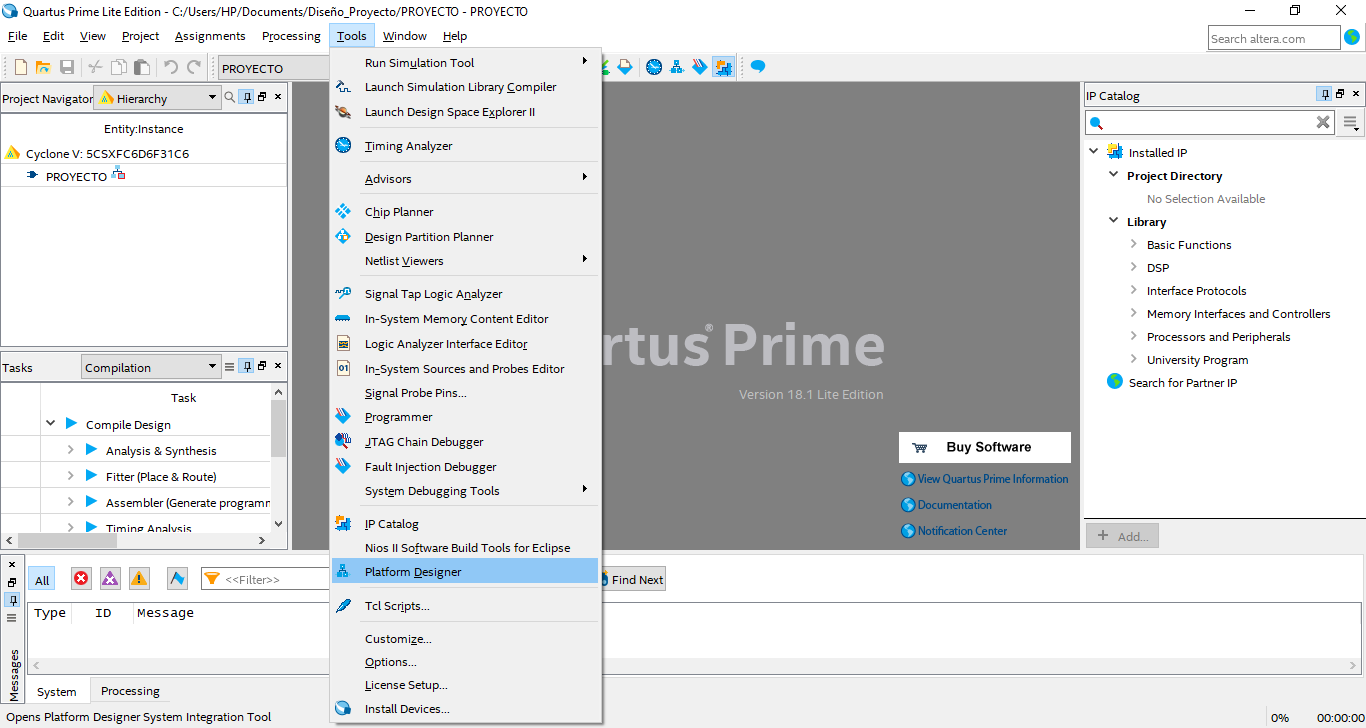


Figura 8:Selección de la herramienta Qsys

1. A continuación, se mostrará la ventana donde se crearán los diferentes componentes de la arquitectura de nuestro proyecto, el primer componente que se mostrará es la interfaz de reloj, cuya función es conectar las señales externas hacia los demás componentes dentro de la arquitectura. Se escoge el componente y con clic derecho sobre el mismo, se procede a renombrar por medio de la opción **Rename,** donde se colocará el nombre **“CLOCK\_50”.**

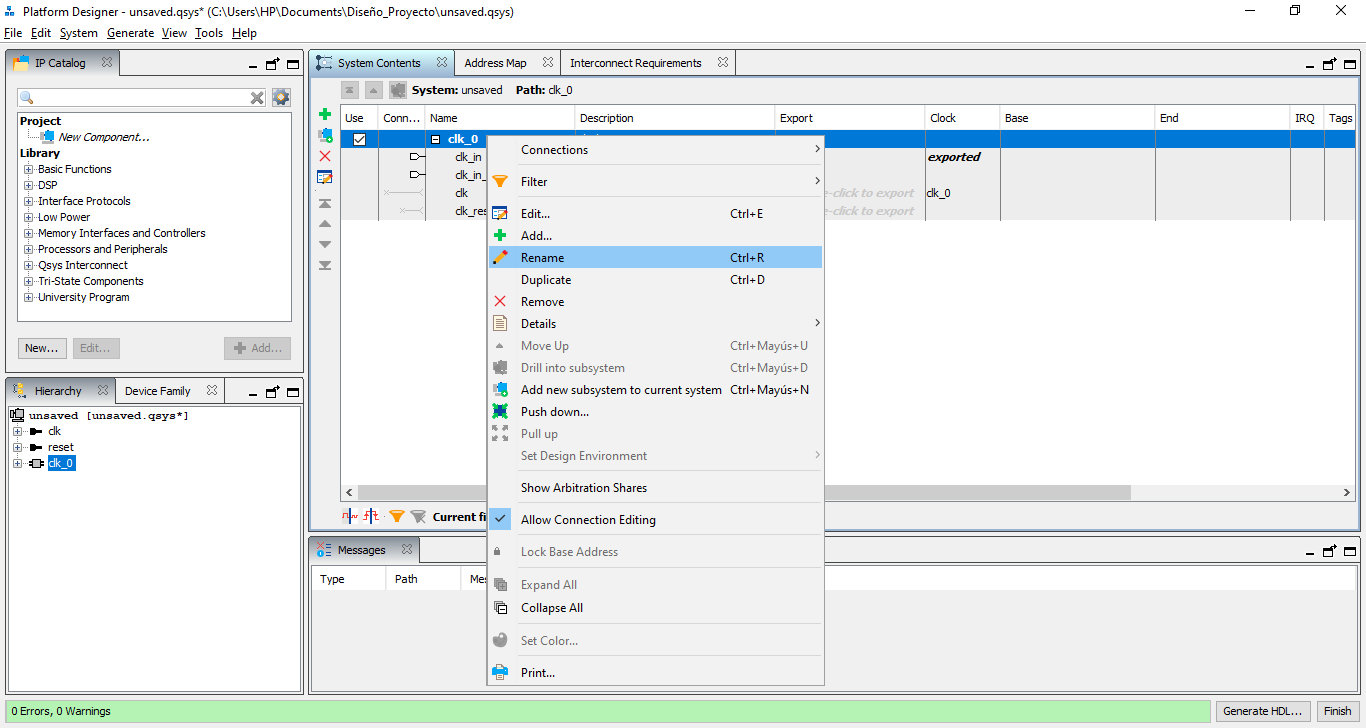


Figura 9: Cambio de nombre al componente Reloj del sistema

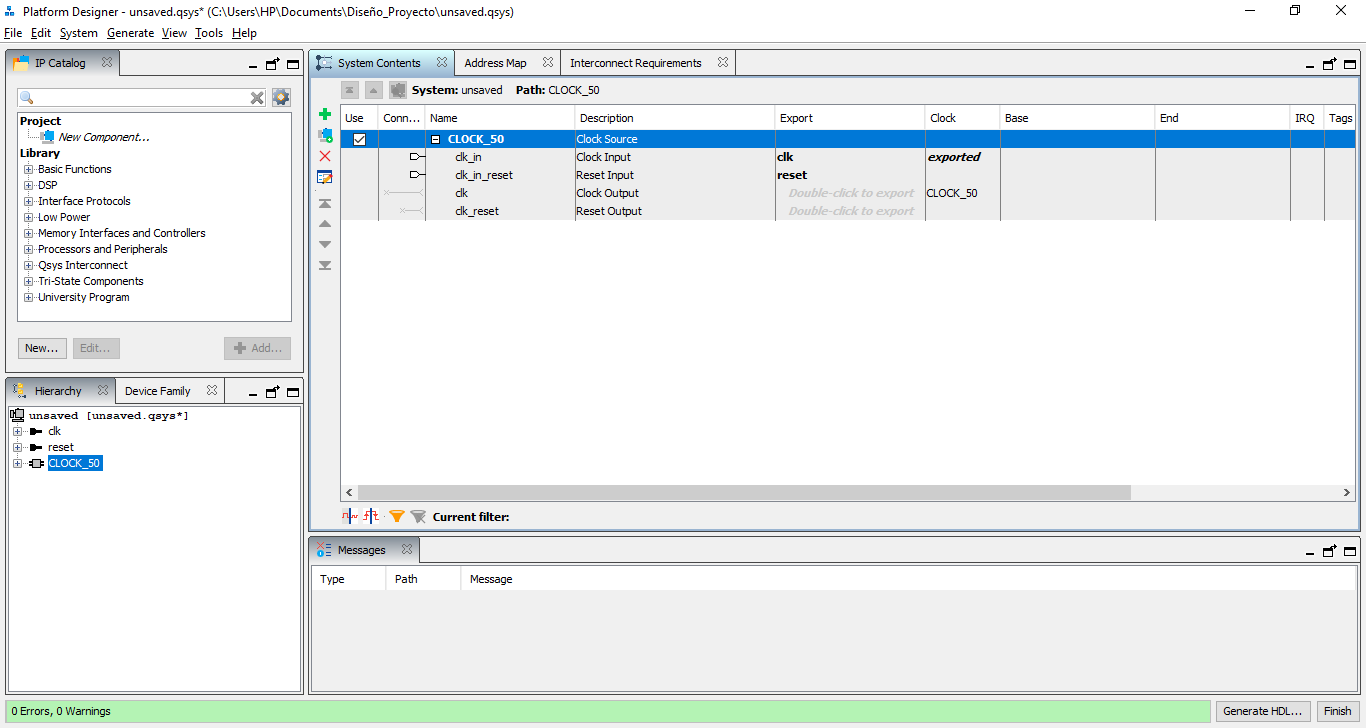


Figura 10: cambio realizado al componente Reloj del sistema por CLOCK\_50

1. Observaremos en la ventana del **Qsys** en la parte izquierda, se muestra la subventana **IP CATALOG**, donde se localizan los bloques que emplearemos en el desarrollo de la arquitectura del sistema, por medio del buscador escribimos el nombre **NIOS II Processor**, se lo elegirá dando doble clic sobre el componente como se observa en la Figura 9.

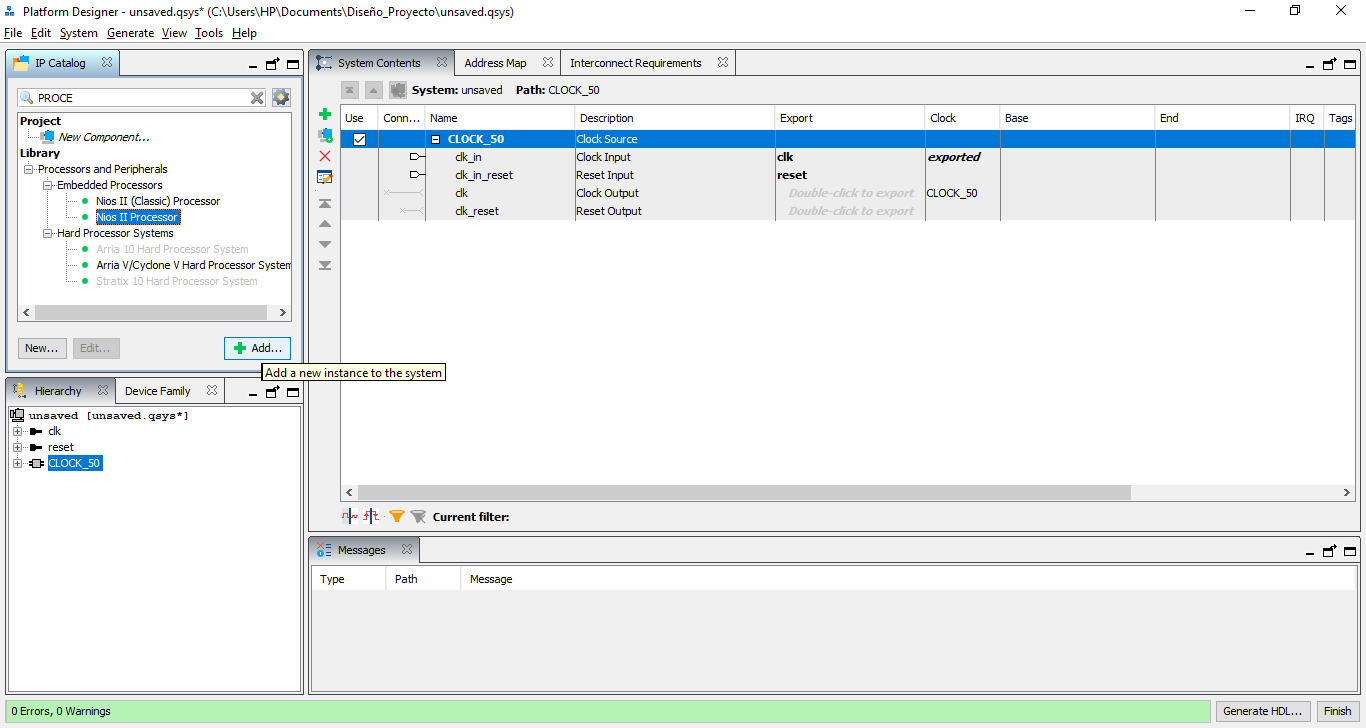


Figura 11: Adhesión del componente NIOS II PROCESSOR

1. Aparecerá la ventana donde seleccionaremos la configuración del procesador a emplear y seleccionaremos **Nios II/e**, como se observa en la figura 10, luego seleccionar **Finish.**

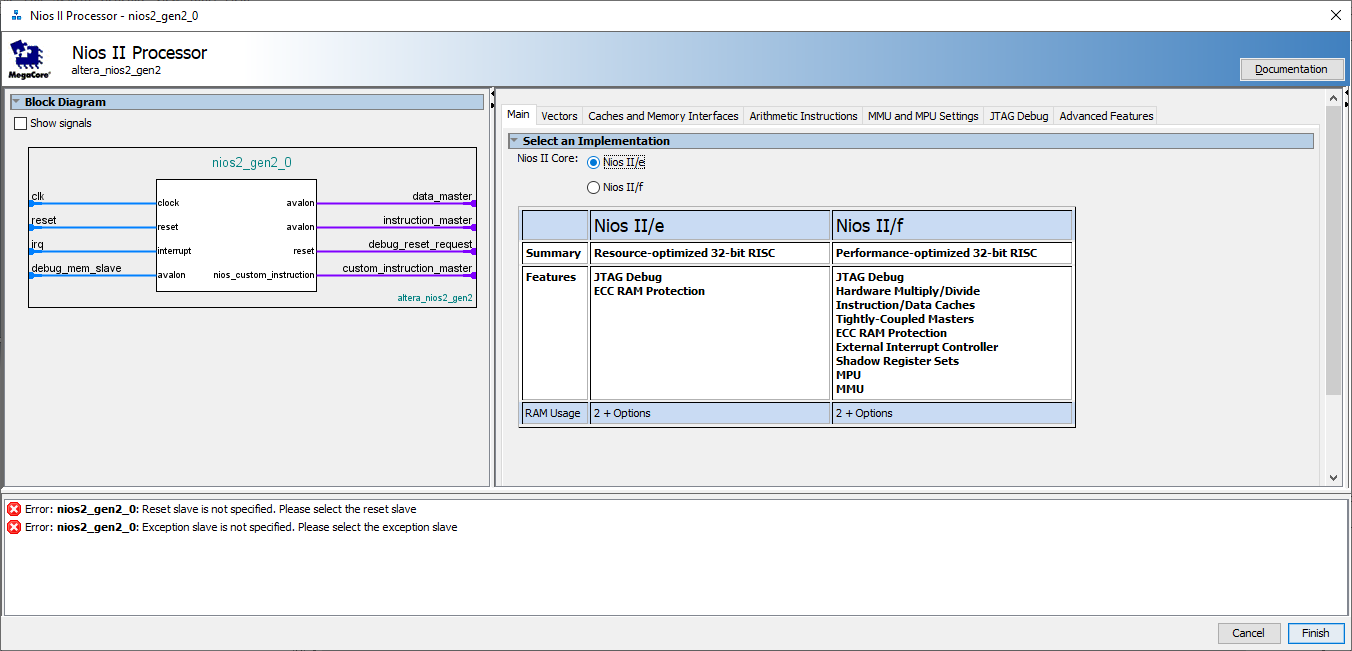


Figura 12: Selección de la configuración del componente Nios II

1. A continuación, se mostrará el componente dentro de la arquitectura, ahora se procede a renombrar el componente **nios2\_gen2\_0**, dando clic derecho y seleccionado la opción **Rename,** el nombre a emplear es “PID”.

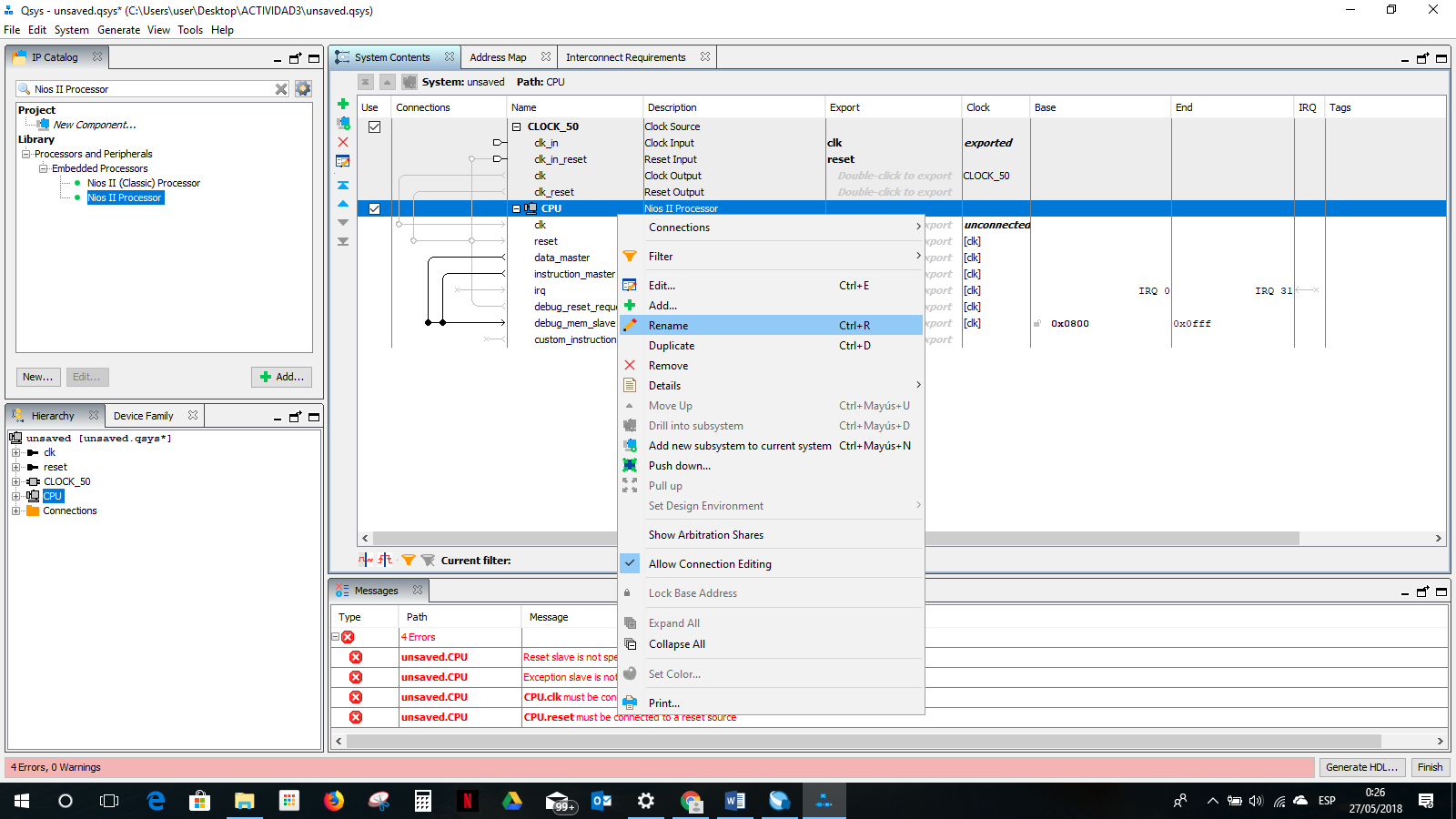


Figura 13: Reasignación de nombre del componente NIOS II PROCESSOR

1. Nuevamente en el buscador de la subventana **IP CATALOG**, se busca el componente **JTAG UART**, y se lo selecciona dicho componente haciendo doble clic sobre el mismo como se muestra en la figura 12.

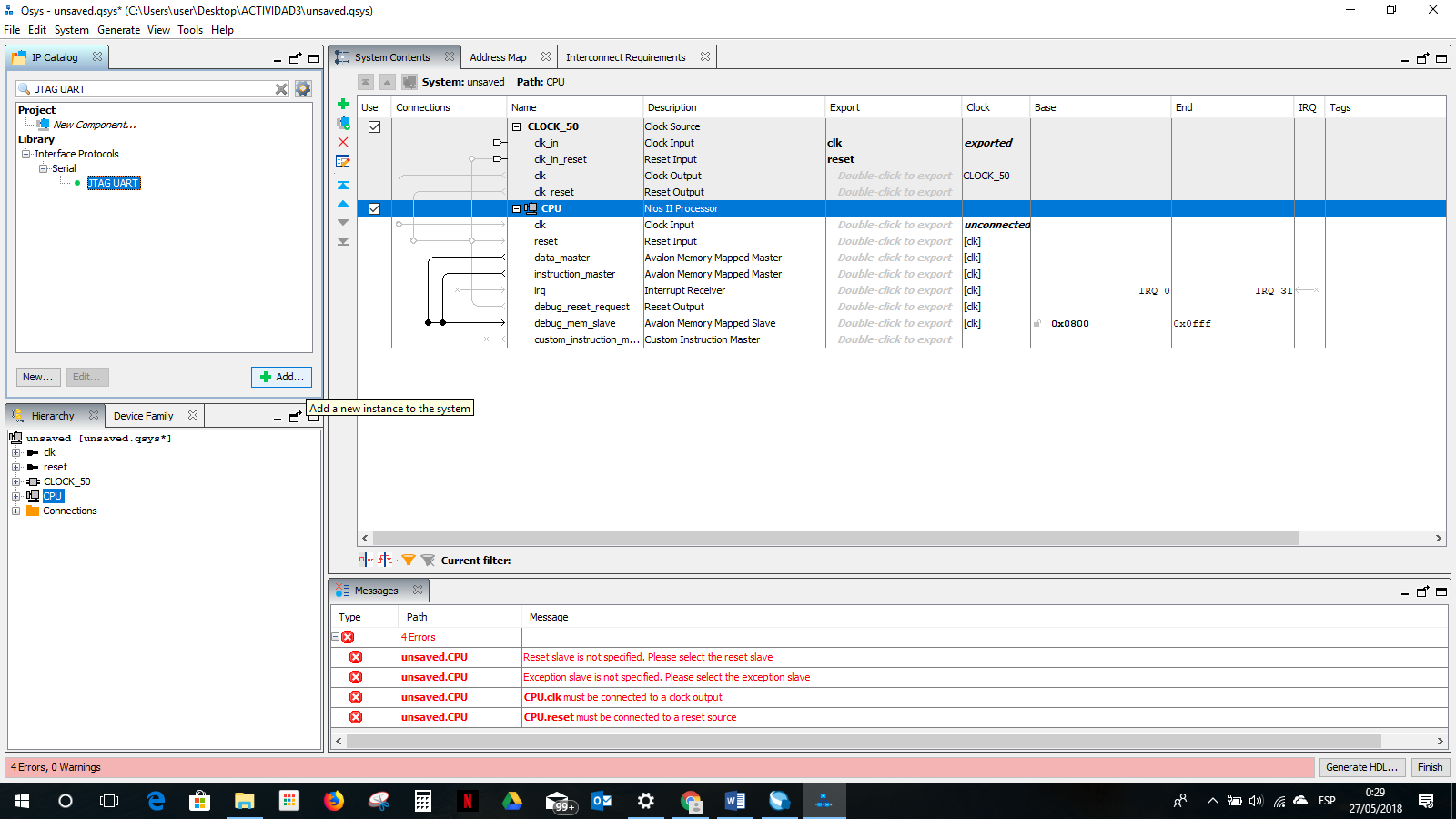


Figura 14: Adhesión del componente JTAG UART

1. Se mostrará la ventana de configuración del componente **JTAG UART**, de clic en **Finish** para anexarlo como se muestra en la Figura 13.

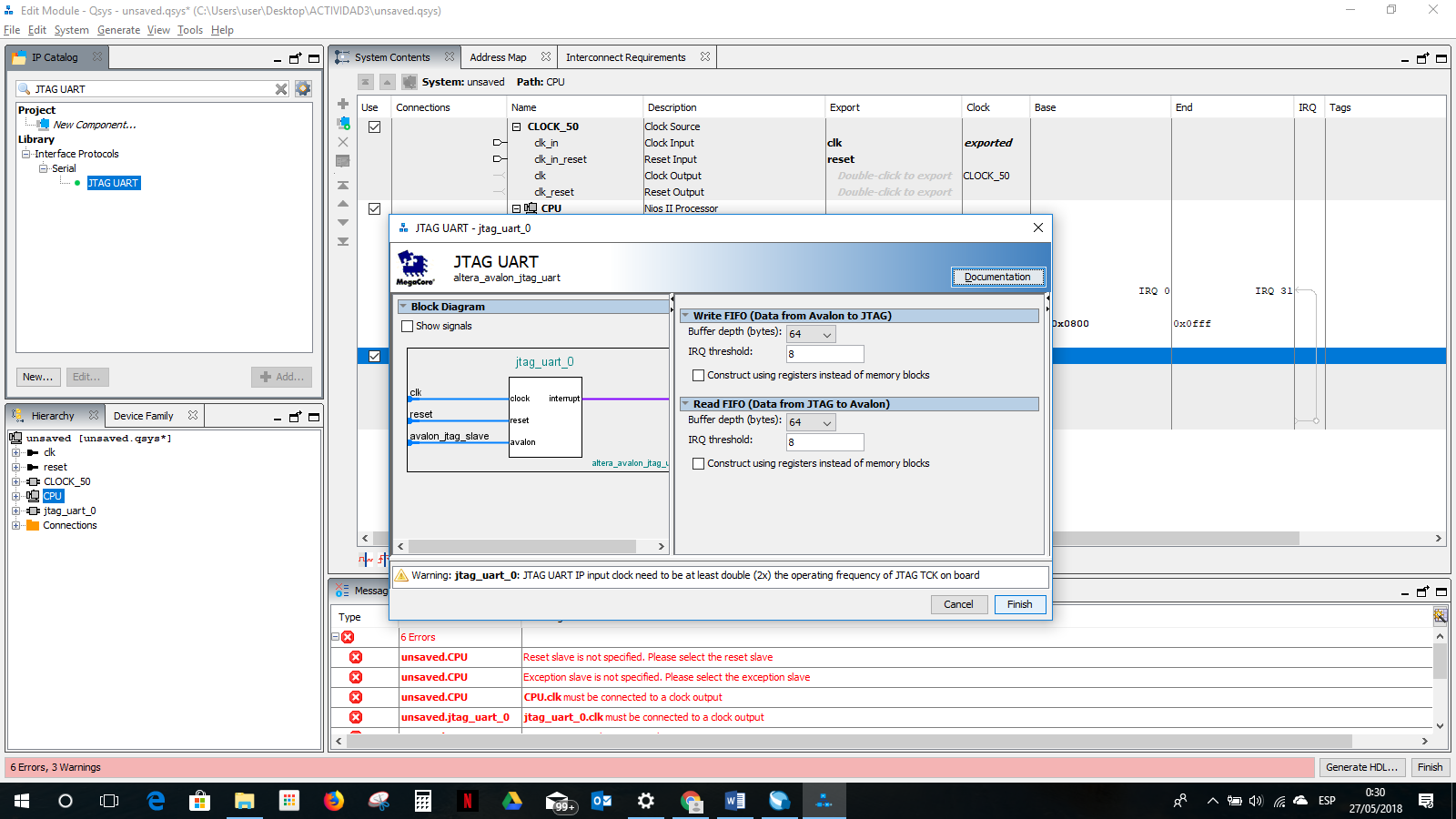


Figura 15: Ventana de configuración del componente JTAG UART

1. Se procede a renombrar el componente **jtag\_uart\_0**, seleccionando con clic derecho sobre el mismo y en la opción **Rename** se escribirá el nombre “**JTAG\_UART**”. Figura 14.

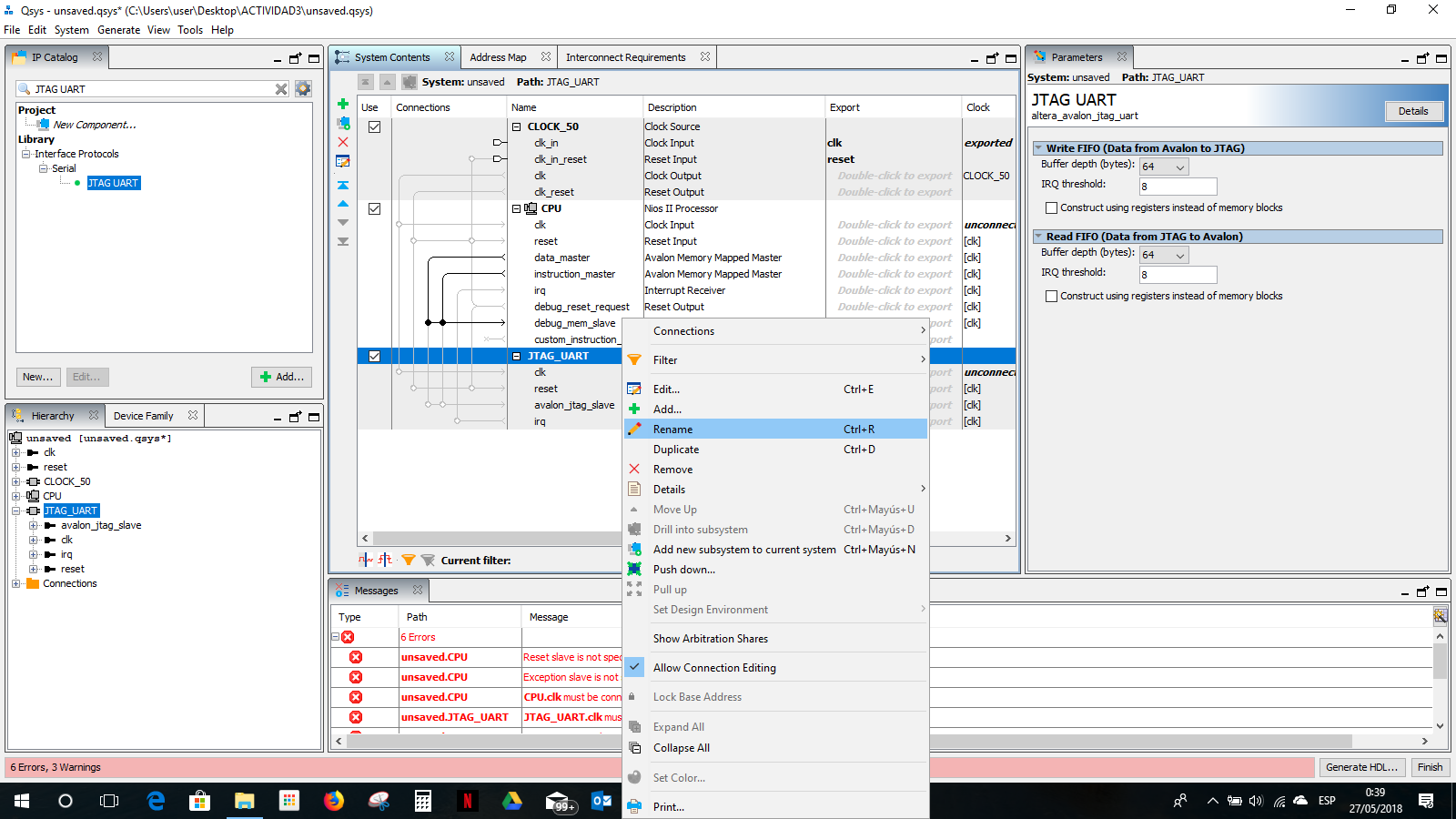


Figura 16: Renombre del componente JTAG\_UART

1. En la subventana **IP Catalog,** se procede a buscar el componente “**System ID**” y elegimos el componente “**System ID Peripheral**” dando doble clic sobre dicho componente. Como se muestra en la Figura 15.

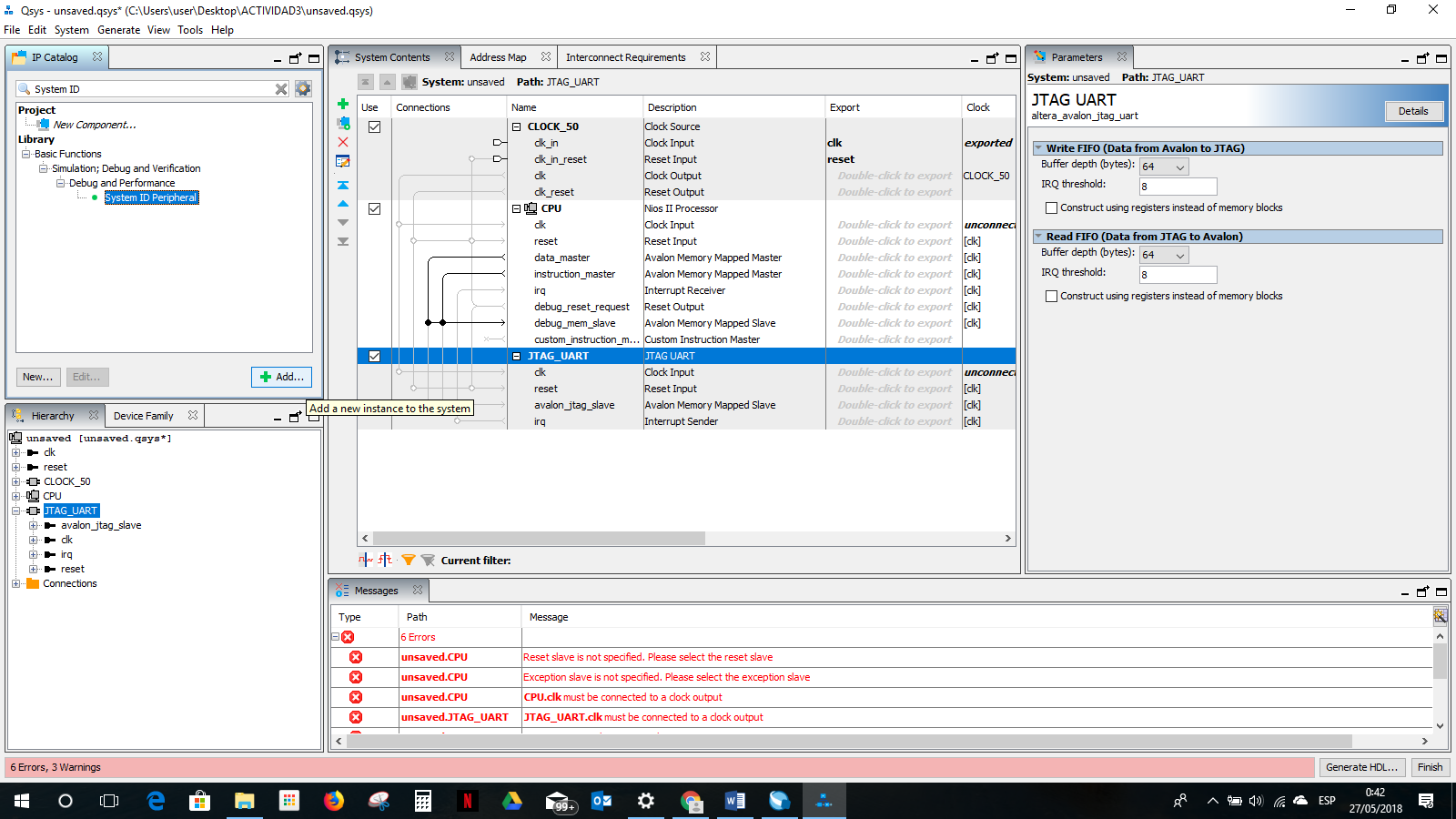


Figura 17: Búsqueda del componente System ID Peripheral

1. Se mostrará la ventana de configuración del componente **System ID Peripheral**, donde seleccionaremos la opción **Finish** para incluirlo a la arquitectura, como se muestra en la Figura 16.

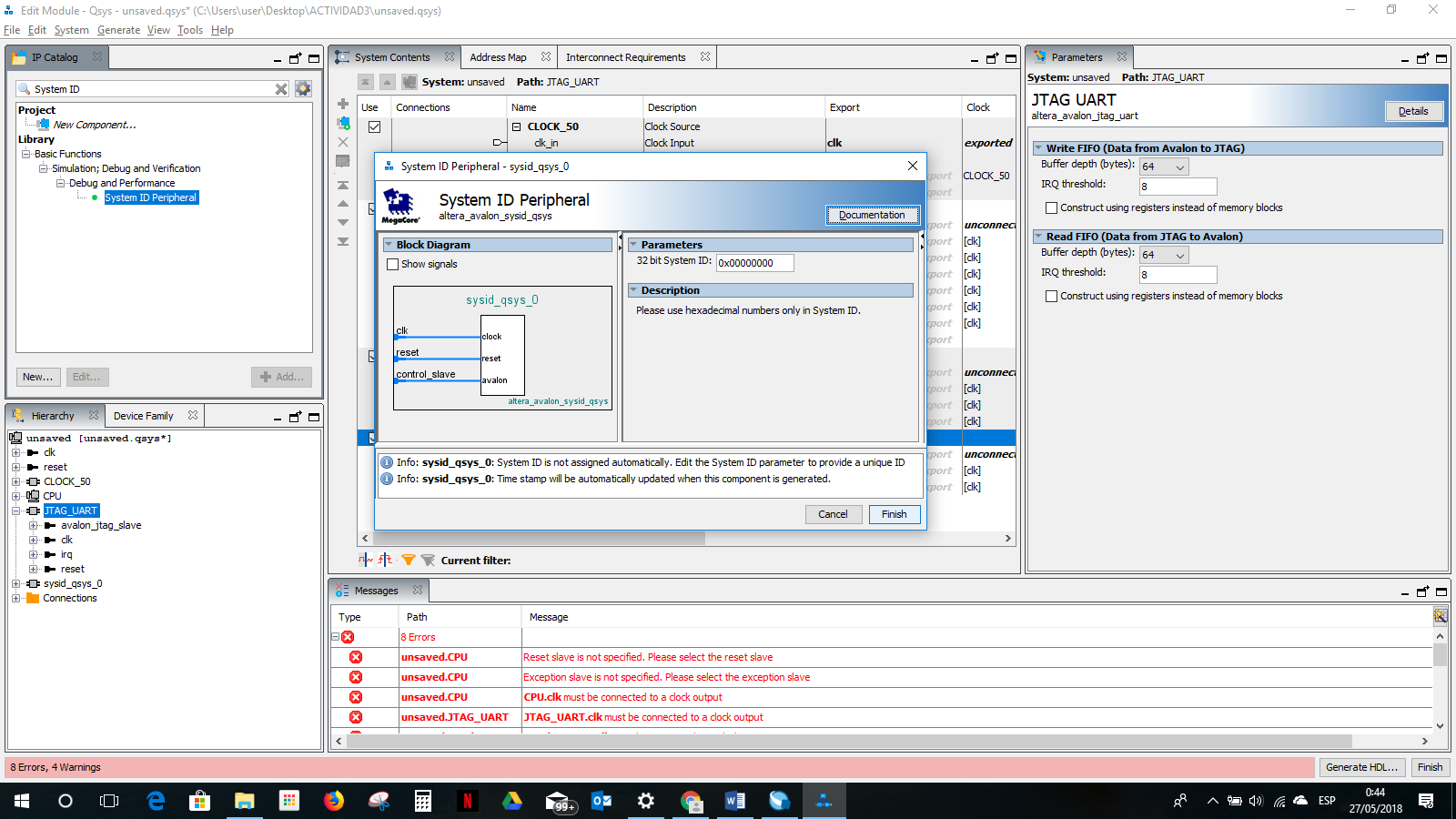


Figura 18: Ventana de configuración del componente System ID Peripheral

1. Procedemos a renombrar el componente **Sysid\_qsys\_0**, dando clic derecho sobre el componente y seleccionando la opción **Renanme**, el nombre del componente será “**SYS\_ID**”, como se muestra en la Figura 17.

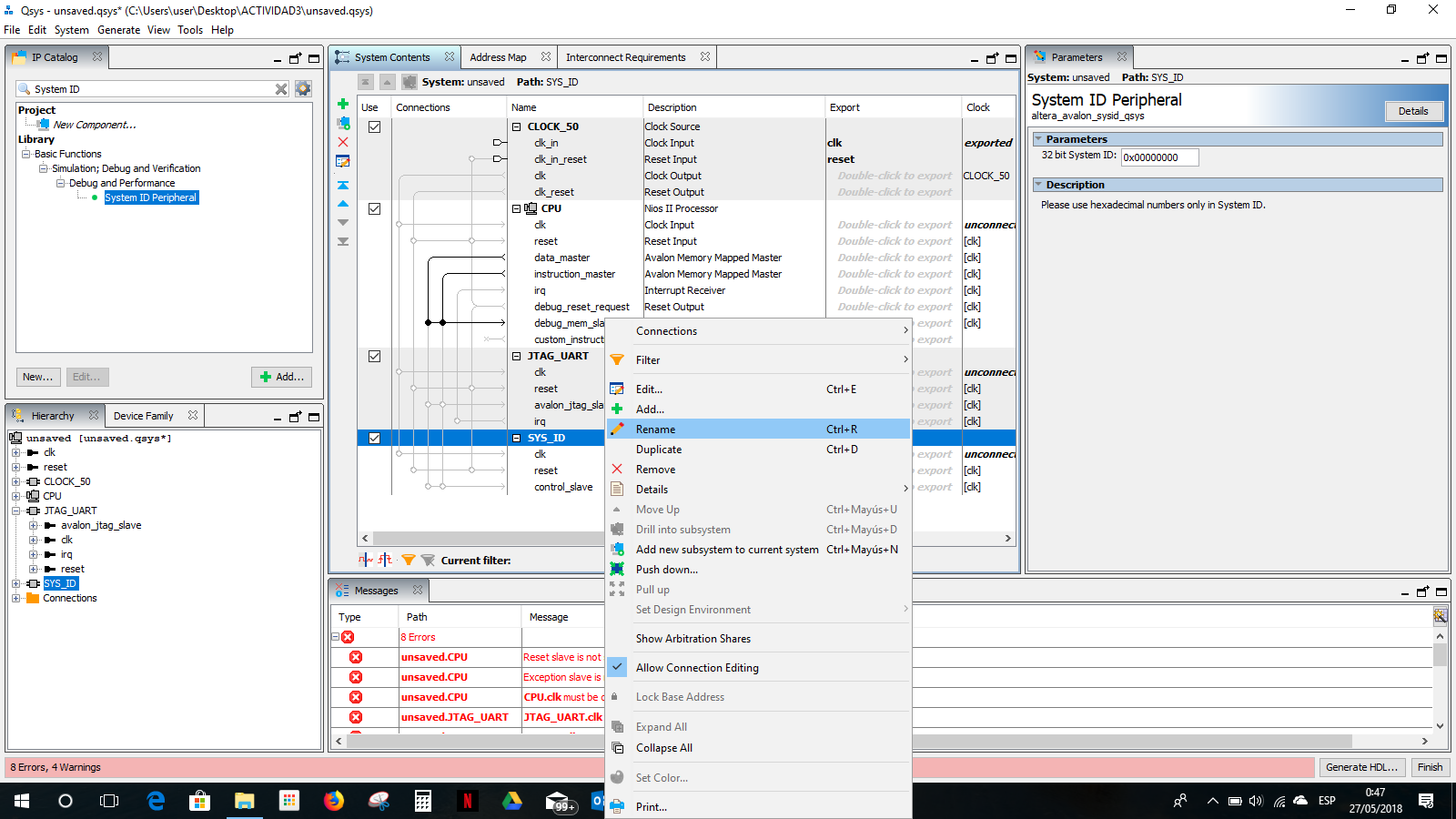


Figura 19: Renombre del componente System ID Peripheral

1. En la subventana **IP Catalog**, se buscará el componente “**On-Chip Ram**”, y se escoge el componente **On-chip Memory** (Ram o Rom), dando doble clic sobre el mismo como se muestra en la Figura 18.

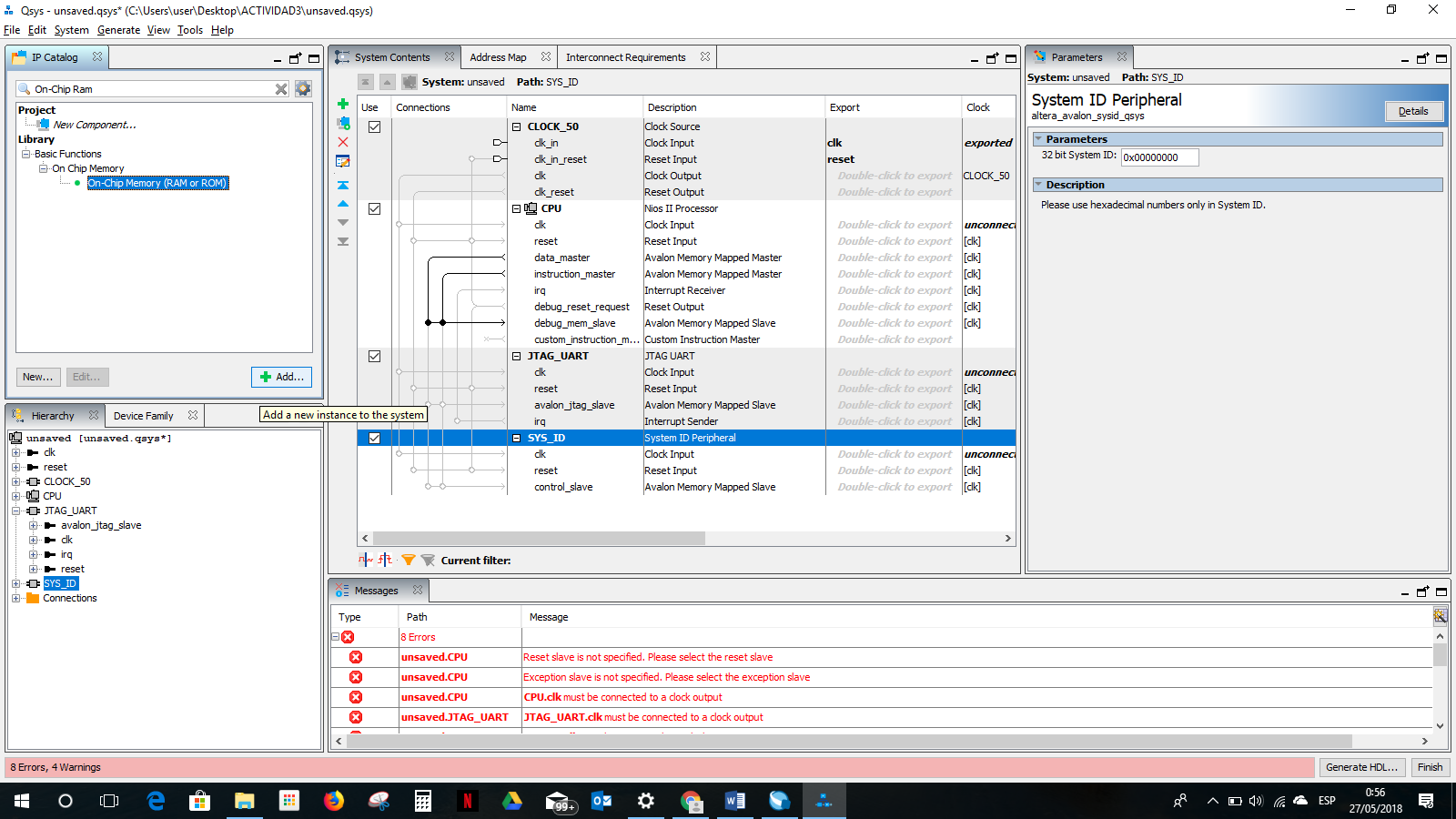


Figura 20: Adhesión del componente On- Chip Memory

1. Aparecerá la ventana de configuración del componente **On-Chip Memory (Ram or Rom),** donde nos dirigiremos a la categoría **Size** y modificaremos el campo **Total memory size** escribiendo el valor 20480 bytes, para finalizar se da clic en **Finish** para anexarlo, como se muestra en la Figura 19.

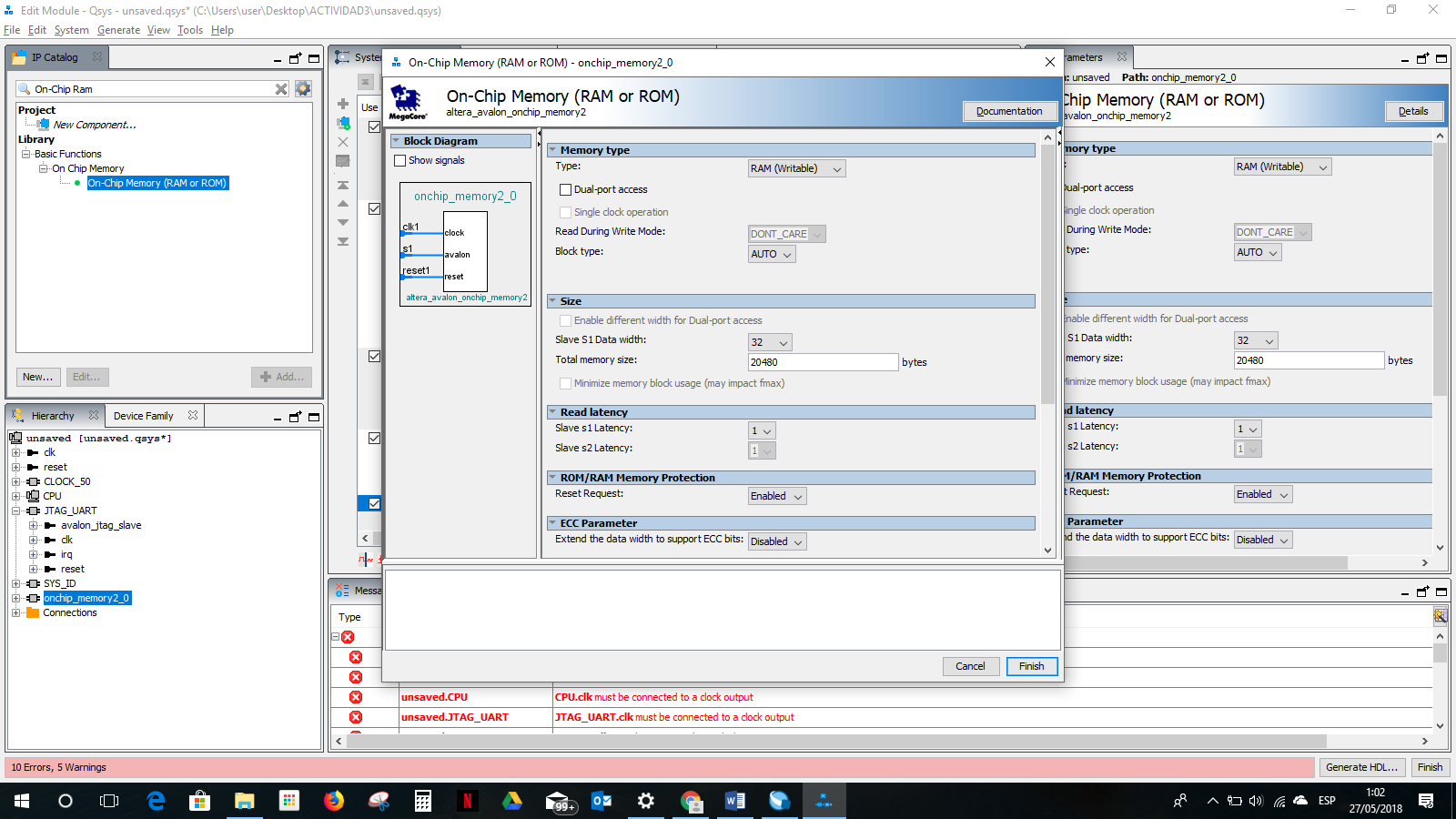


Figura 21: Ventana de configuración del componente On-Chip Memory (Ram or Rom)

1. Se renombrará el componente **onchip\_memory2\_0,** dando clic derecho sobre el componente y seleccionando la opción **Rename,** el nombre a asignar será “**SRAM\_PID**”, como se muestra en la Figura 20.

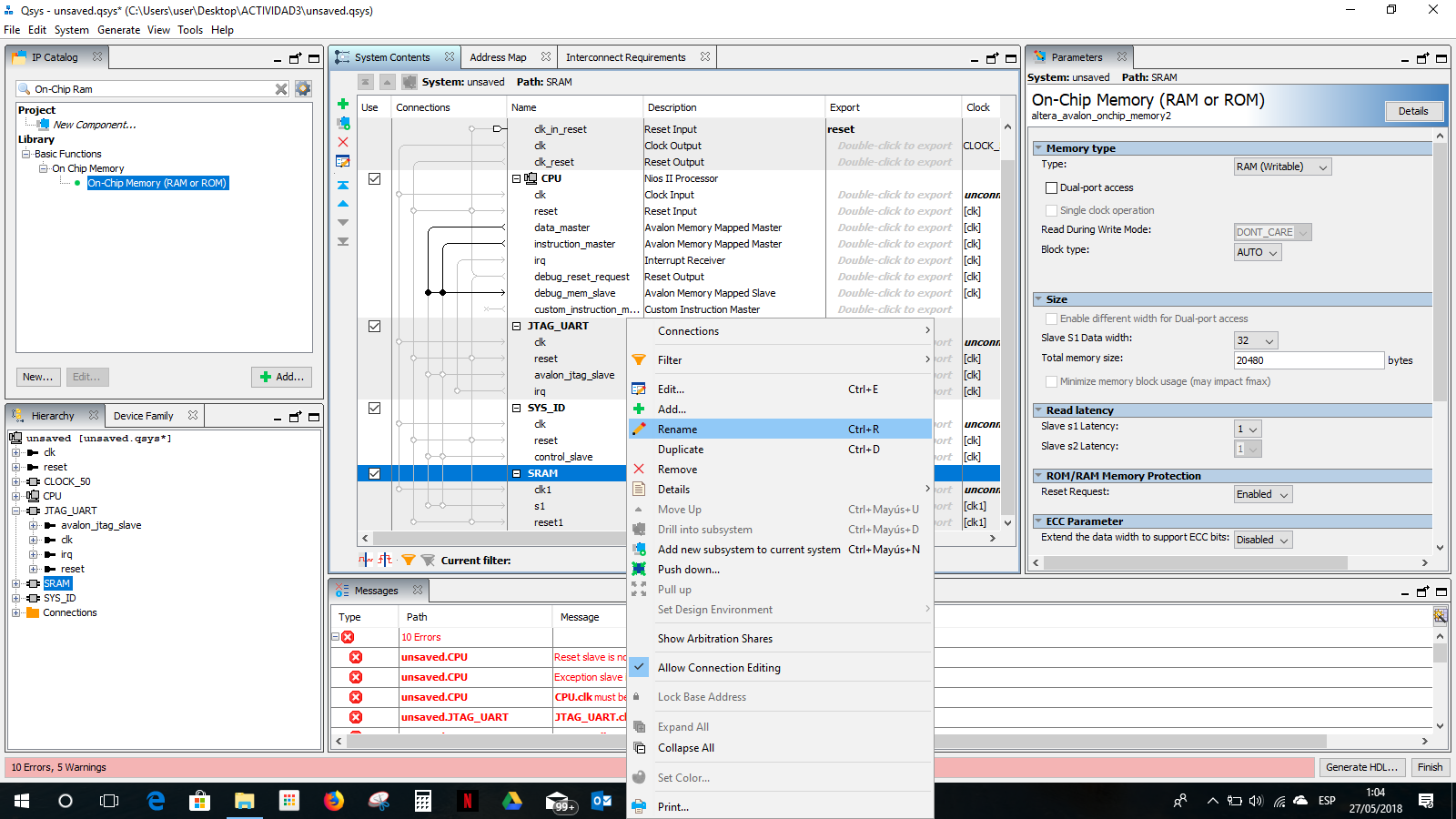


Figura 22: Renombre del componente On-Chip Memory (Ram or Rom)

1. En la subventana IP Catalog, se busca “**Timer”,** y se selecciona el componente “**Interval Timer**”, dando clic sobre el componente para anexarlo como lo muestra la Figura 21.

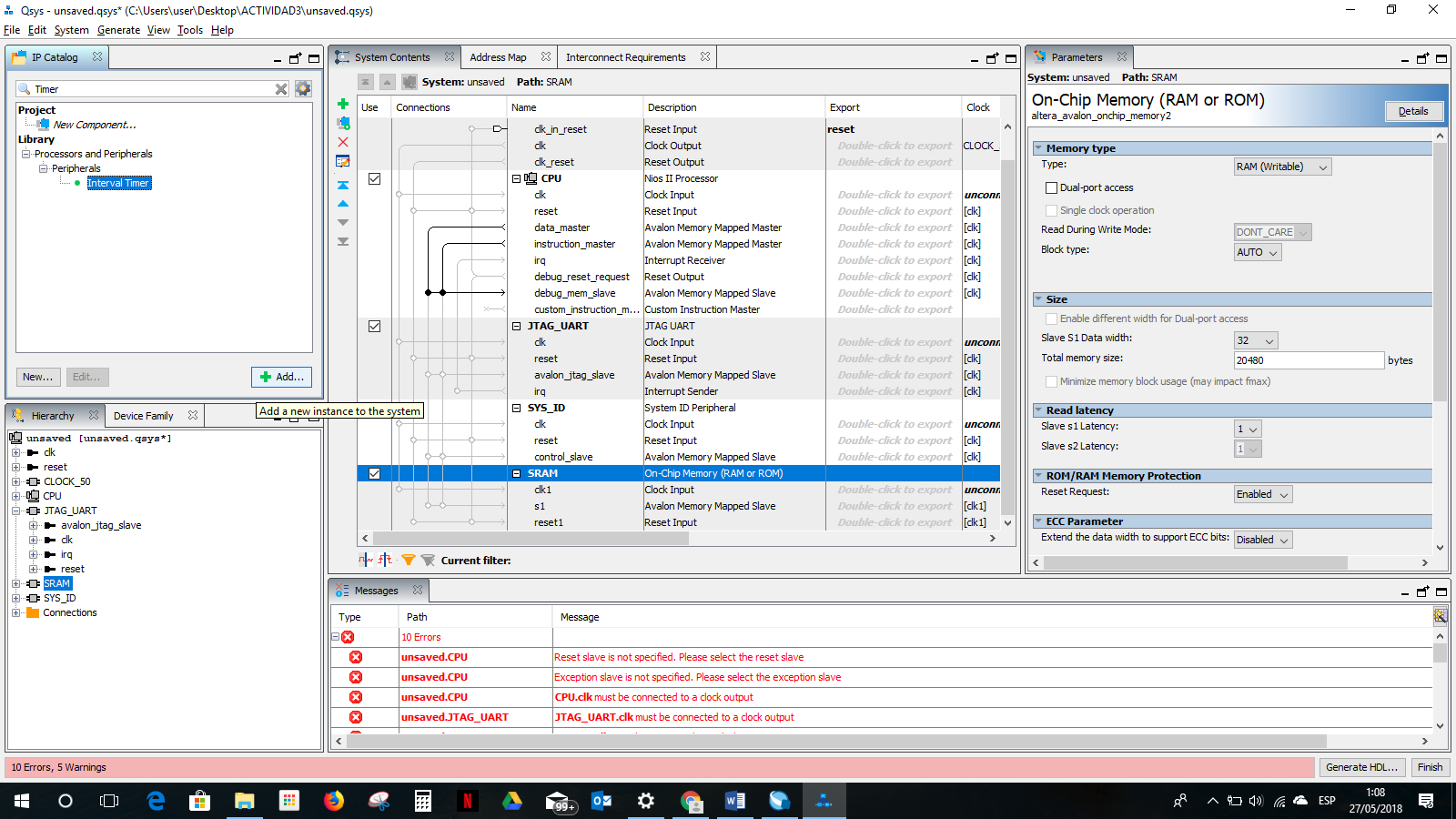


Figura 23:Búsqueda del componente Interval Time

1. Se mostrará la ventana de configuración del componente **timer\_0**, procederemos a dar clic en **Finish** para añadirlo como se muestra en la Figura 22.

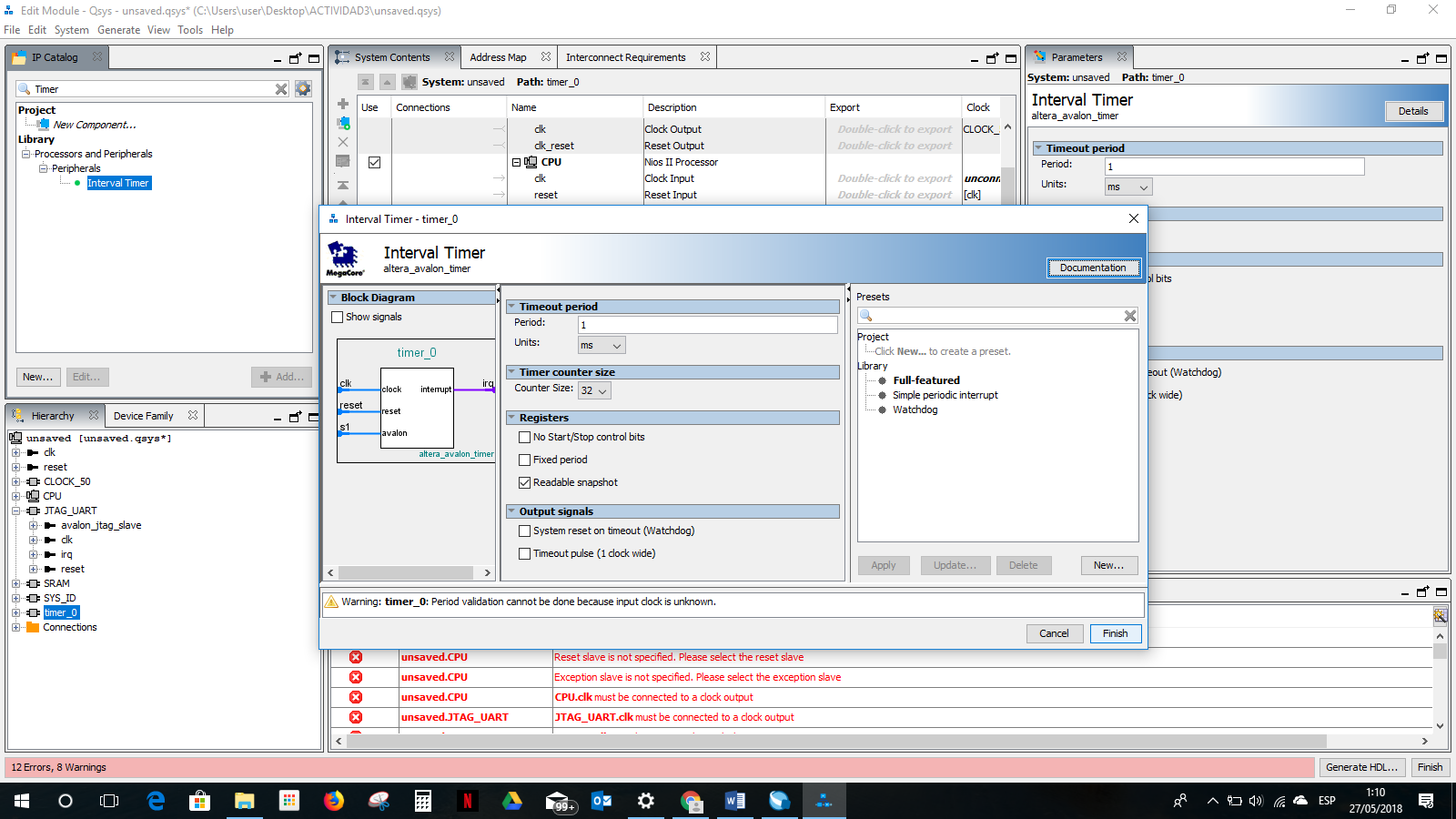


Figura 24: Ventana de configuración del componente Interval Timer

1. Se procede a renombrar el componente **timer\_0** a “**TIMER**”, dando clic derecho sobre el componente y seleccionando la opción **Rename** como lo muestra la Figura 23.



Figura 25: Renombre del componente timer\_0

1. En la subventana IP Catalog buscamos el componente PLL Intel FPGA IP, lo anexamos dando doble clic sobre el mismo como lo muestra la Figura 26.

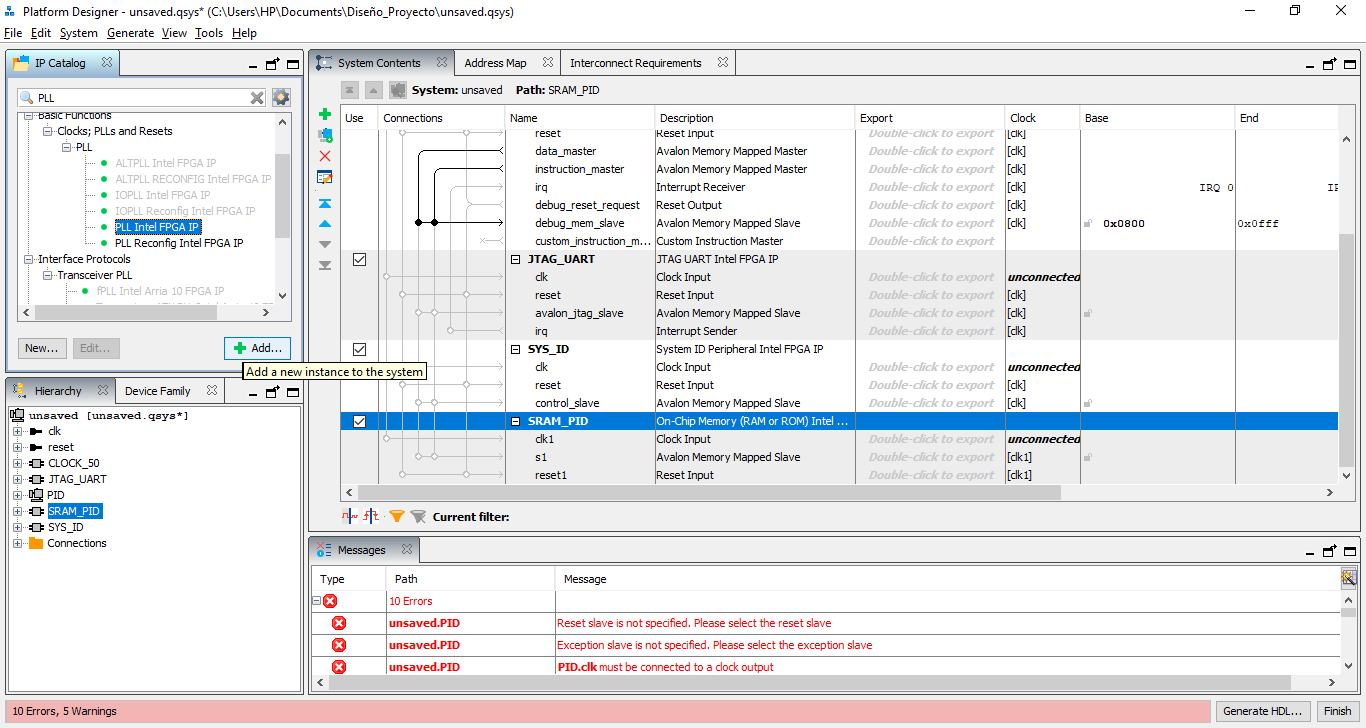


Figura 26:Busqueda del componente PLL Intel FPGA IP

1. Se mostrará la ventana de configuración del componente donde se realizará el cambio de las opciones para cada **outclk**, tal como se muestra en la Figura 27. Una vez concluido el cambio se selecciona **Finish**.

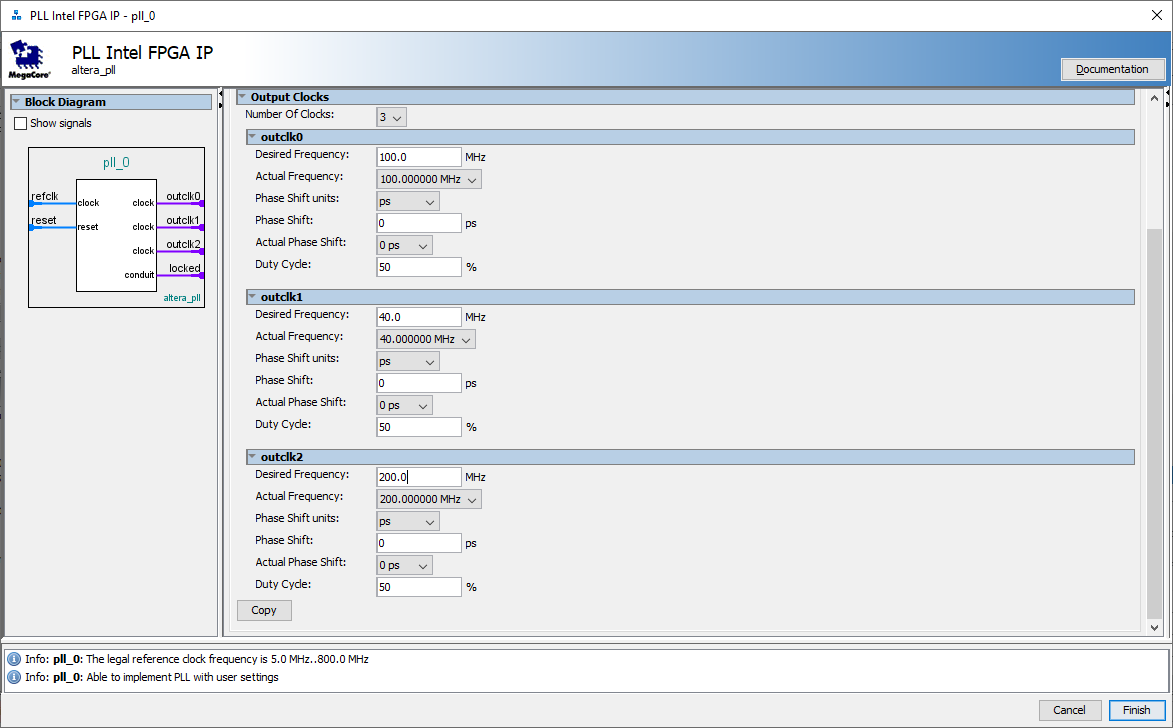


Figura 27:Ventana de configuración del componente PLL

1. Se procede a renombrar el componente dando clic derecho sobre el componente y seleccionamos **Rename**, donde escribiremos “**PLL**” como el nuevo nombre del componente. Como se muestra en la Figura 28.

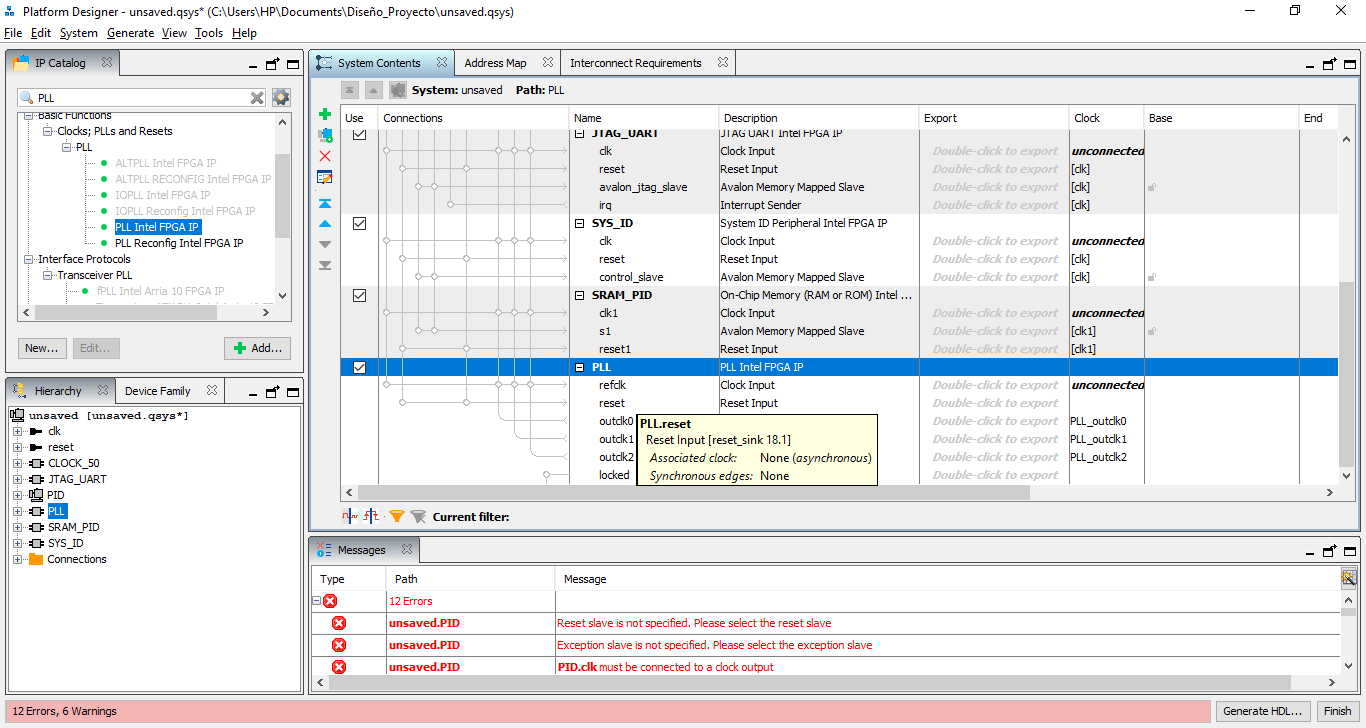


Figura 28: Renombre del componente PLL

1. A continuación, se copia la carpeta **IP** que se encuentra en la carpeta **DE10\_Standard\_ADC** que se encuentra en los archivos provistos de la práctica, copie y pegue la carpeta dentro de la carpeta de su proyecto tal como se, muestra en la figura 29

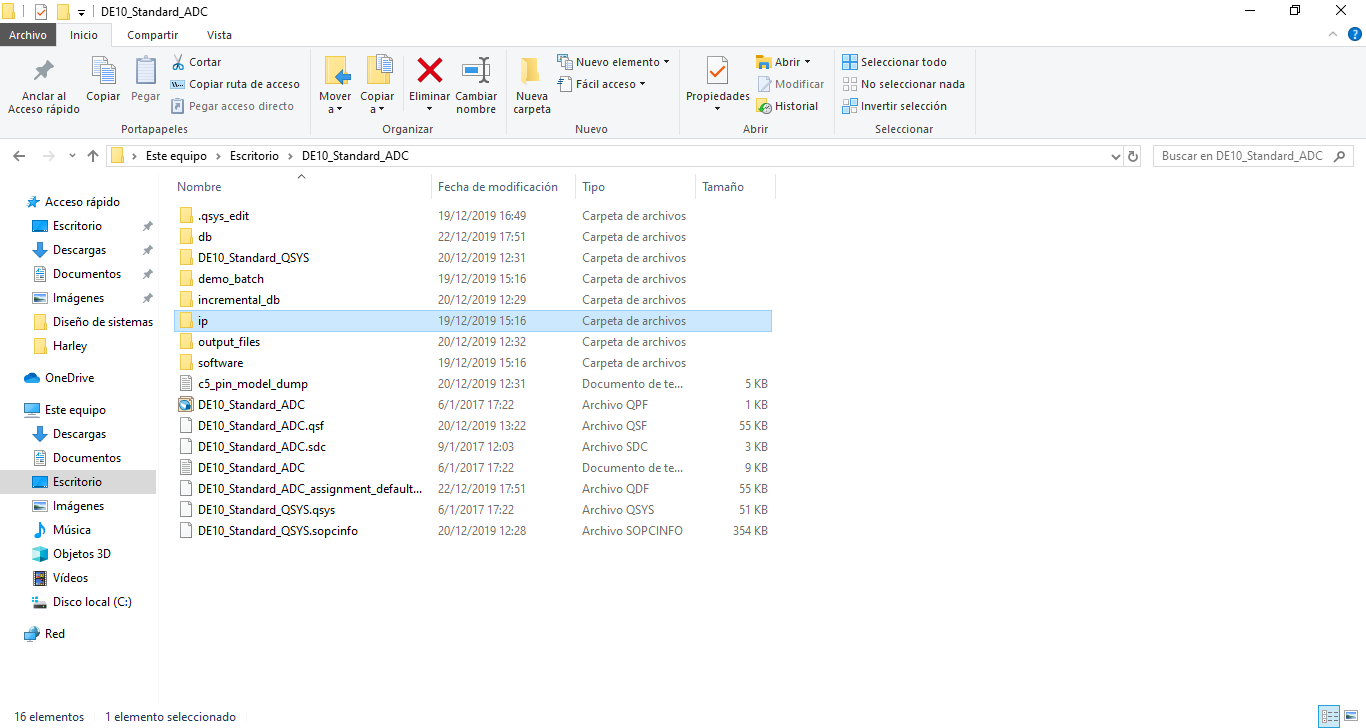


Figura 29: Búsqueda de carpeta IP para su posterior anexión al proyecto

1. Se procede a guardar el proyecto con el nombre **PID,** al realizar esta acción ahora aparecerán las componentes propias que se desarrollan de forma externa al proyecto, es decir componentes creadas por otros usuarios. La Figura 30 muestra el componente **adc\_Itc2308** que se ha añadido a la subventana **IP Catalog .**

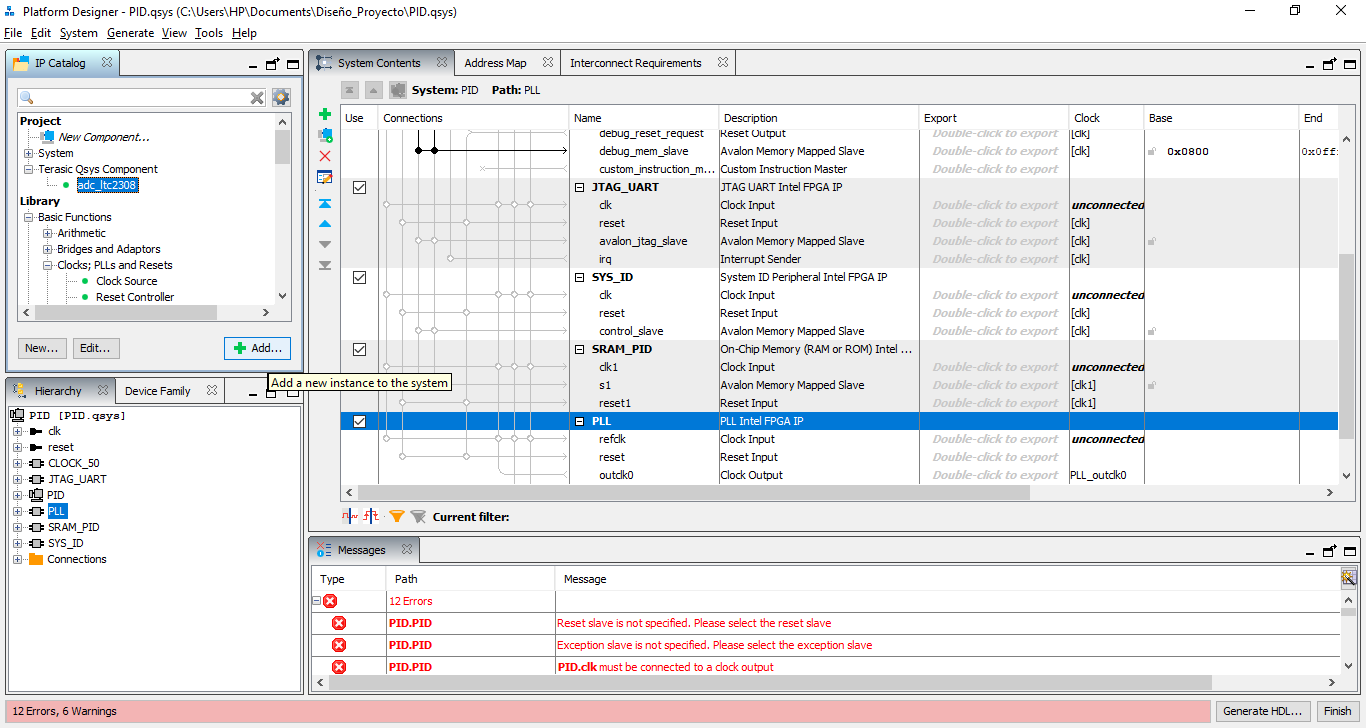


Figura 30: Componente añadida a IP Catalog adc\_Itc2308

1. Se añade el componente al proyecto y se procede a renombrar el nombre del componente, dando clic derecho sobre el mismo y seleccionando la opción **Rename,** el nombre a designar será “**ADC**”, como se muestra en la Figura 31.

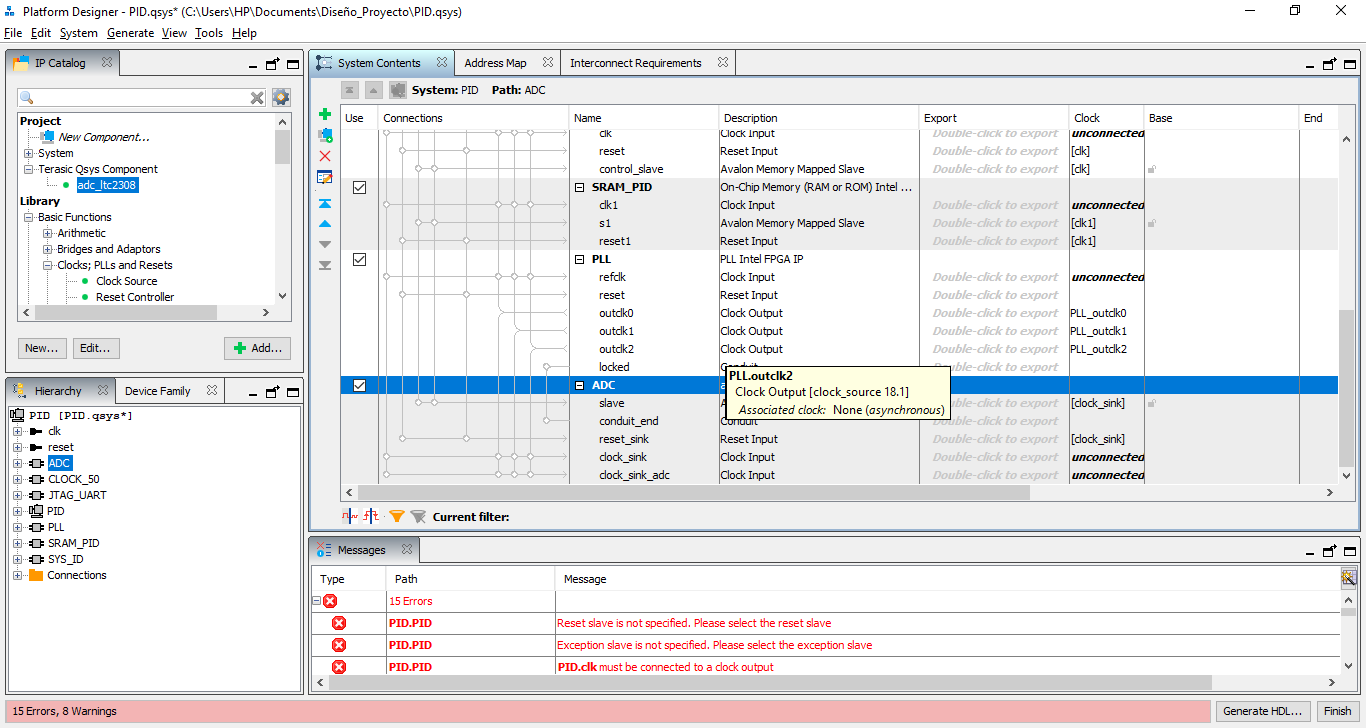


Figura 31: Renombre del componente ADC

1. Procedemos a buscar el componente **PIO Parallel I/O Intel FPGA IP,** por medio de la subventana **IP catalog**, añadimos el componente dando doble clic como se muestra en la Figura 32.



Figura 32: Búsqueda de componente PIO

1. Se mostrará la ventana de configuración del componente PIO y realizaremos la configuración del componente tal y como se muestra en la Figura 33.

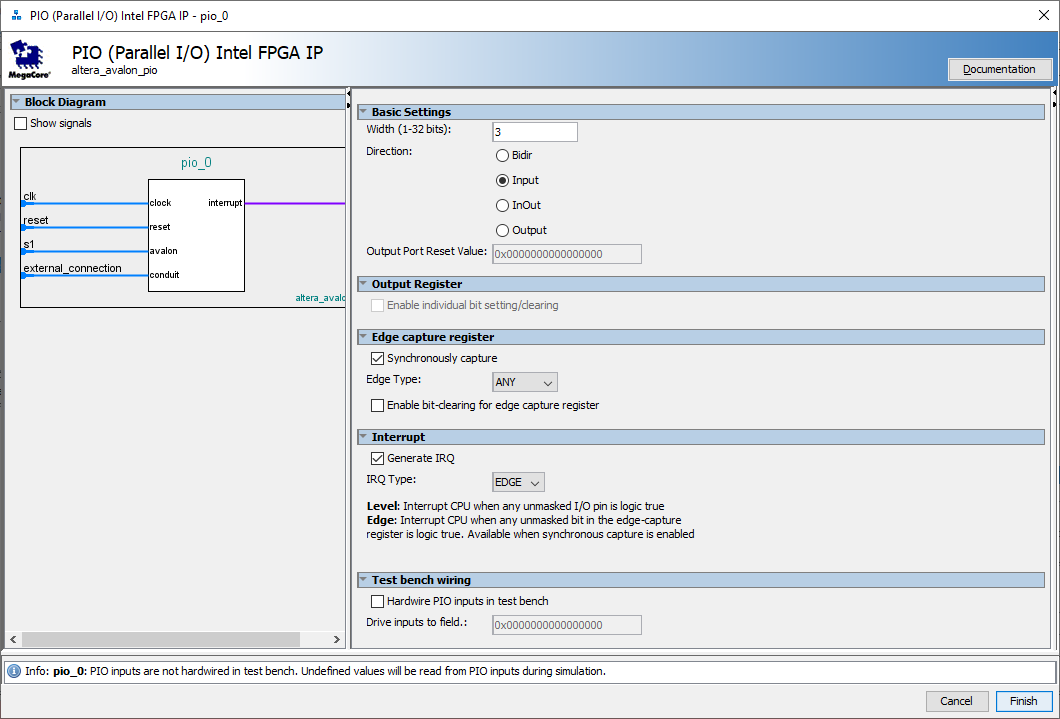


Figura 33: Ventana de configuración del componente PIO

1. Una vez concluida la configuración, se selecciona **Finish**, para dar por terminado la adhesión, procedemos a renombrar el componente con el nombre **PIO\_INPUT**
2. A continuación, en la ventana seleccionaremos en la barra de herramientas la opción **System**, se desplegará una ventana de opciones donde se elegirá la opción **Create Global Reset Network**, tal como se muestra en la Figura 33.

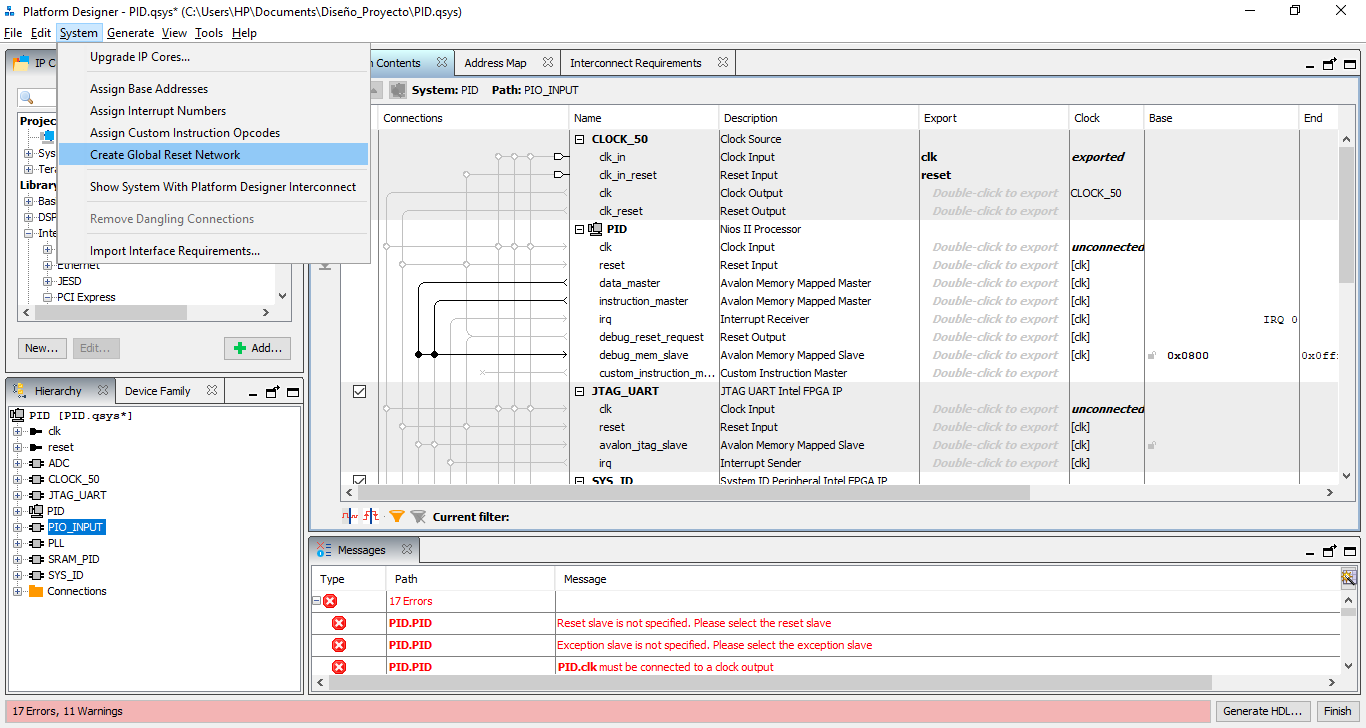


Figura 34: Ventana para la selección del reset Global

1. Se realiza la configuración del procesador del sistema como se muestra en la Figura 35.

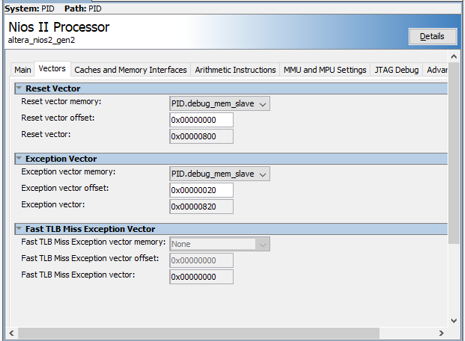
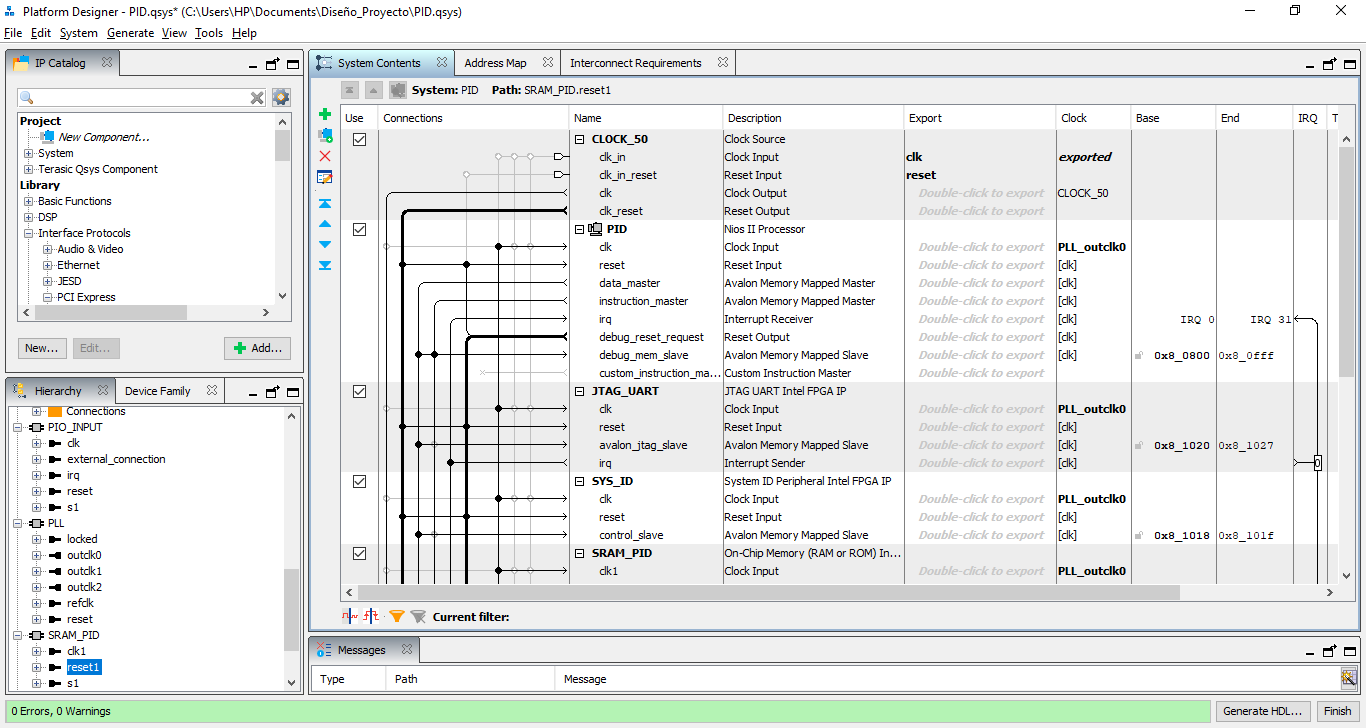


Figura 35:

1. Se procede a realizar las conexiones entre los diferentes componentes que se han añadido. Como se observa en la Figura 36.



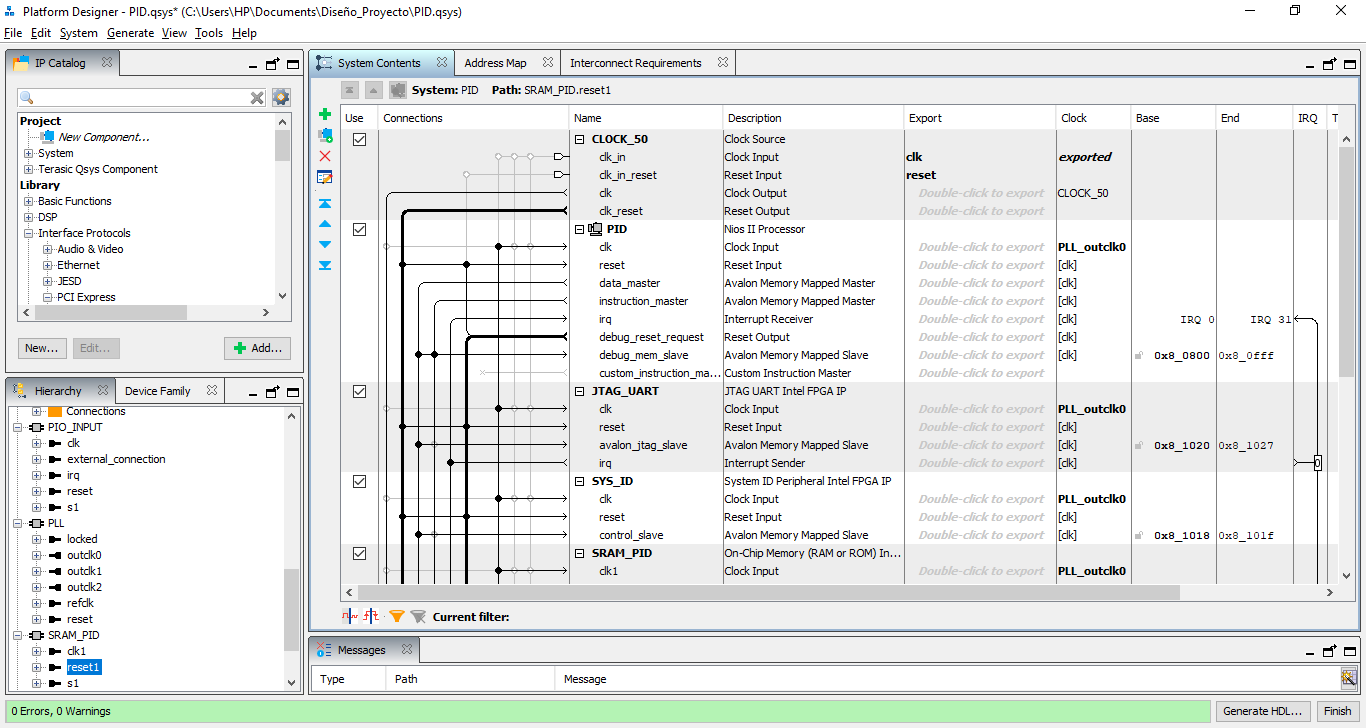


Figura 36: Conexiones entre los diferentes componentes del sistema

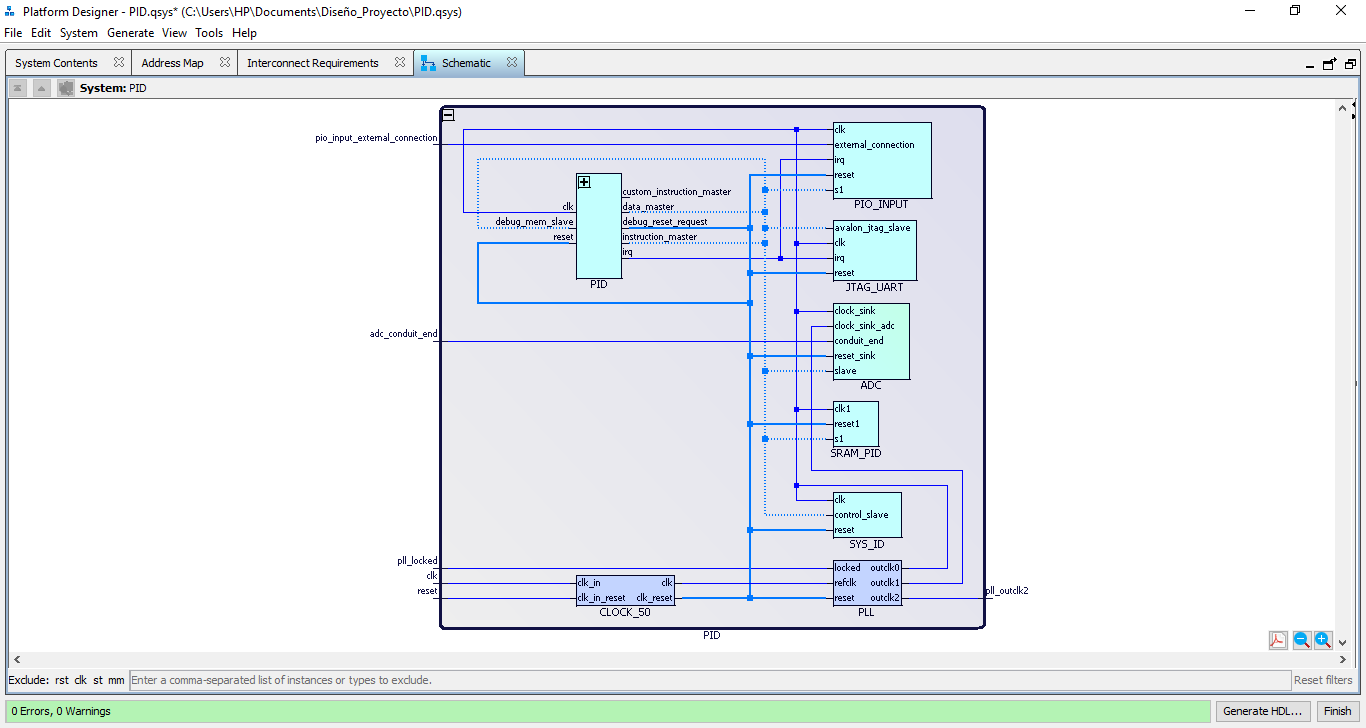


Figura 37: Diagrama esquemático de las conexiones de los componentes

1. Se procederá a realizar la generación del **VHDL,** dando clic en la opción **Generate VHDL,** que se encuentra en la parte inferior de la ventana del **Qsys**

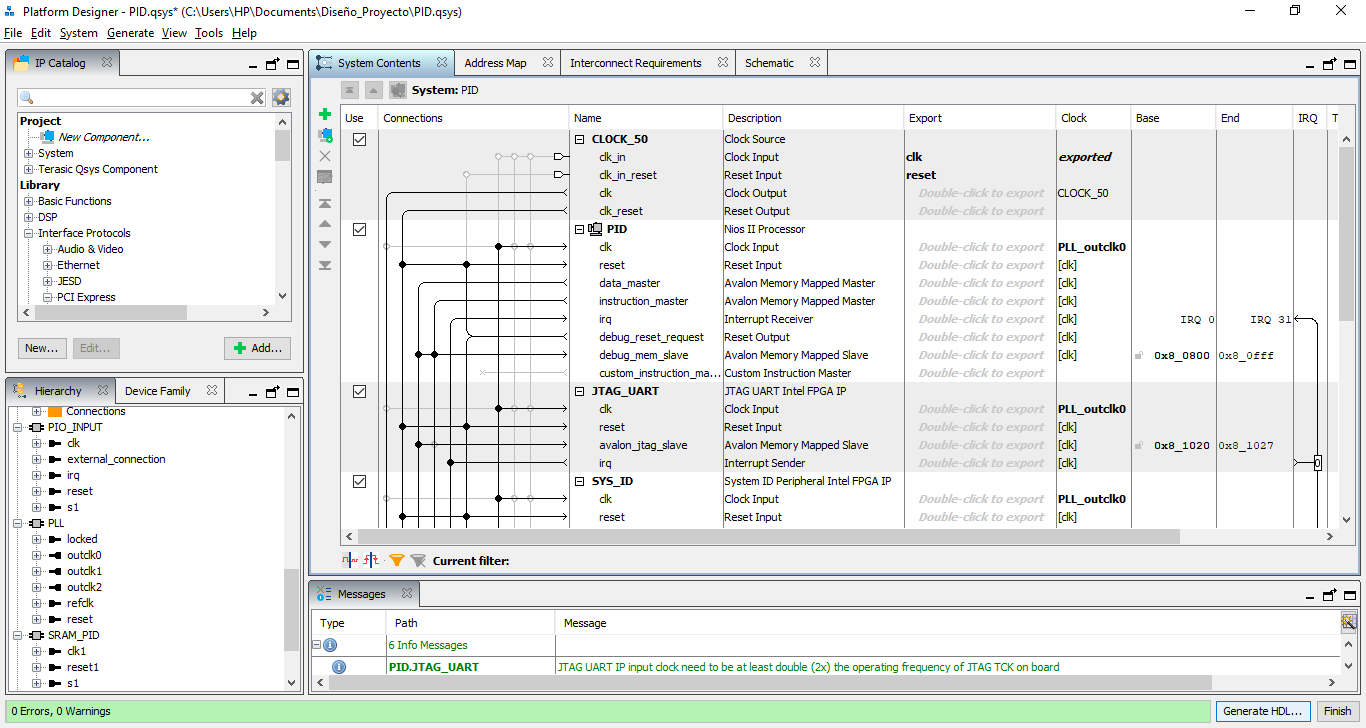
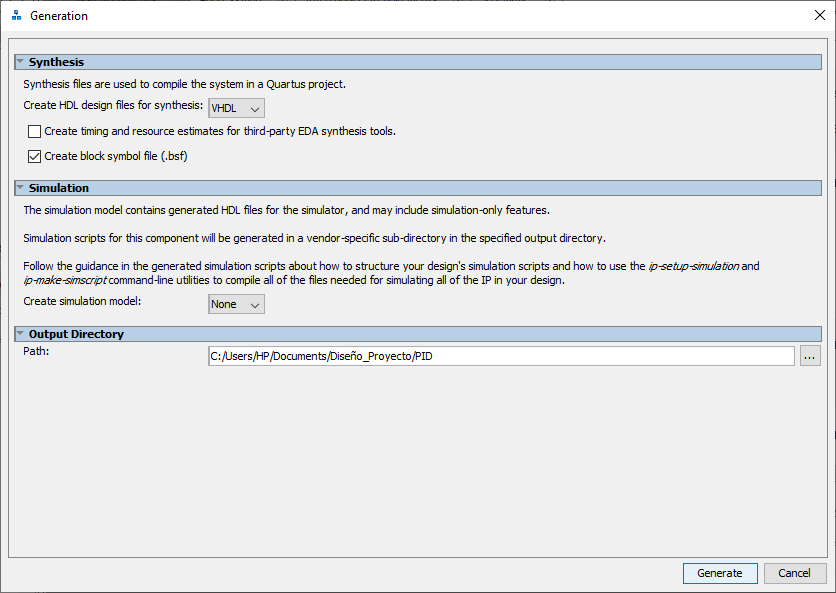


Figura 38; Selección de Generacion de VHDL

1. Se mostrará una ventana donde en la opción **Create HDL design files for synthesis**, seleccionaremos la opción **VHDL**, al concluir se escoge la opción **Generate** que se encuentra en la parte inferior de la ventana como se muestra en la Figura 39.



1. Una vez concluida la generación del vhdl se presentará la ventana donde se mostrará si existe error alguno en el proceso, caso contrario aparecerá un mensaje en la parte inferior de la ventana **completed successfully**.

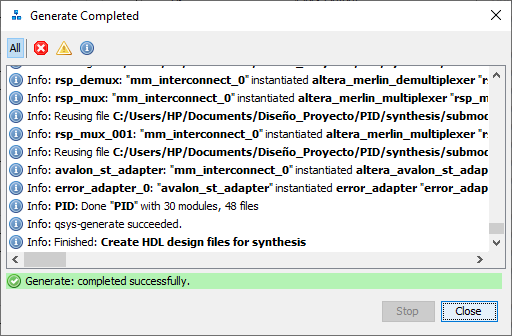


Figura 39: Ventana que muestra la compilación correcta del sistema

1. Una vez concluida la generación de VHDl se procede a cerrar la ventana por medio de la selección de la opción **close**, y en la ventana principal de **Qsys**, se escogerá la opción **Finish** que da como concluida la arquitectura de nuestro sistema.

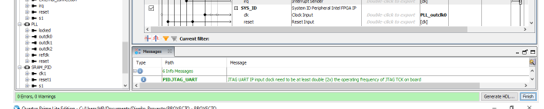


Figura 40: selección de la opción Finish

1. Se coloca como alta jerarquía el Qsys creado y procedemos a compilar por medio del icono 

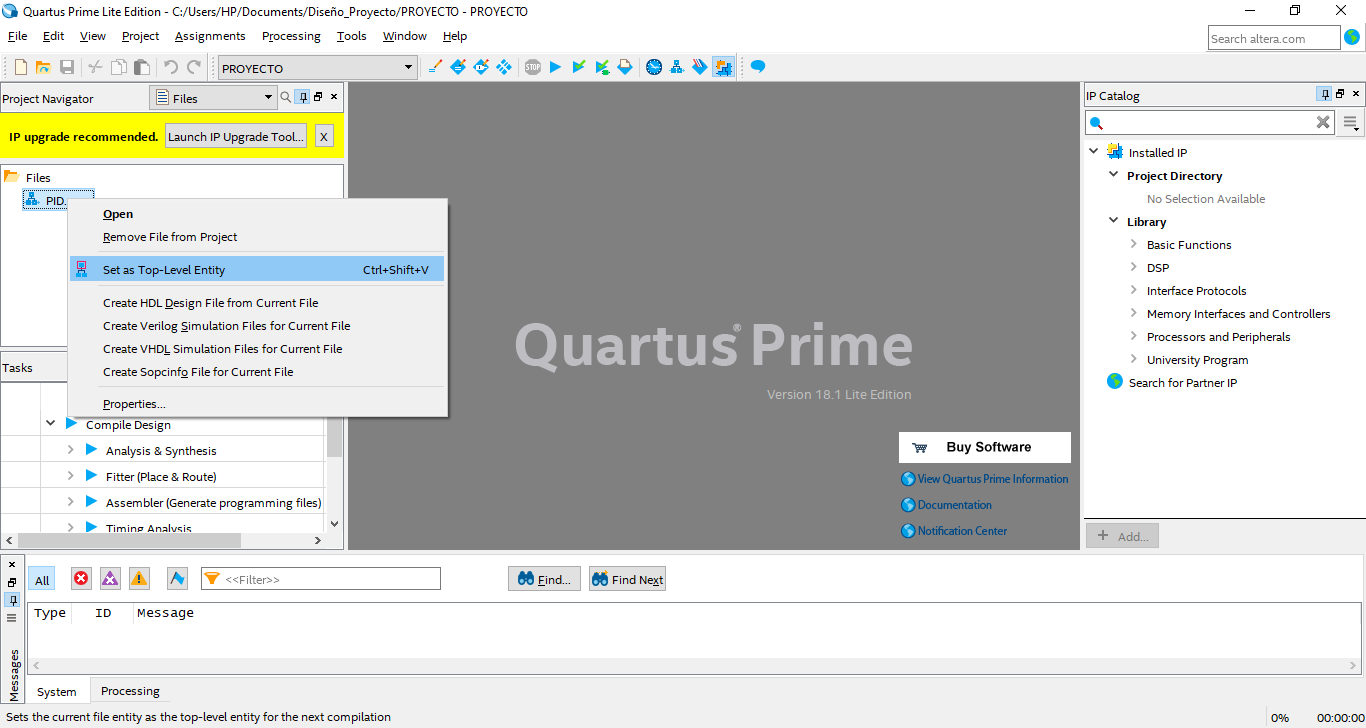


Figura 41: Selección del Qsys como alta jerarquía

1. Proceda a conectar la tarjeta de desarrollo **DE10-Standard** tanto a la PC (cable **USB-Blaster**) como a la fuente de alimentación.
2. Una vez que se haya completado la compilación, diríjase al ícono que se encuentra en la barra de tareas del programa **Quartus Prime**, y servirá para programar la tarjeta de desarrollo **DE10-Standard.**
3. En la ventana Programmer se visualizará un archivo, haga clic derecho en el mismo y elimínelo escogiendo la opción **Delete,** tal como se visualiza en la Figura 10.

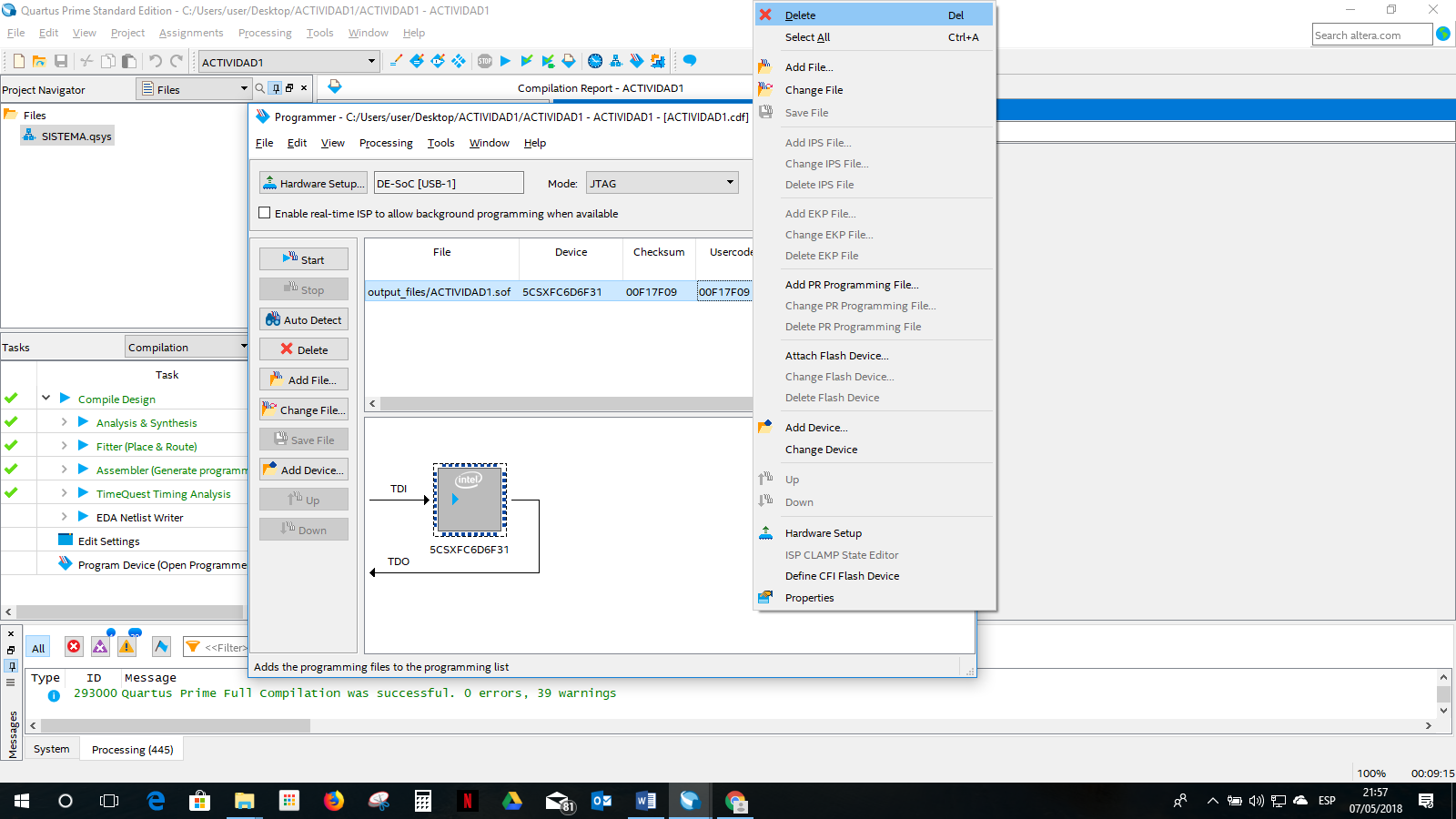


Figura 42: Ventana Programmer

1. Una vez eliminado el archivo y que no se encuentre ningún otro archivo en la ventana Programmer, proceda a escoger la opción **Auto Detect** representada por la siguiente botonera . Seleccione la opción **5CSXFC6D6** tal como se muestra en la Figura 11. Haga clic en OK.

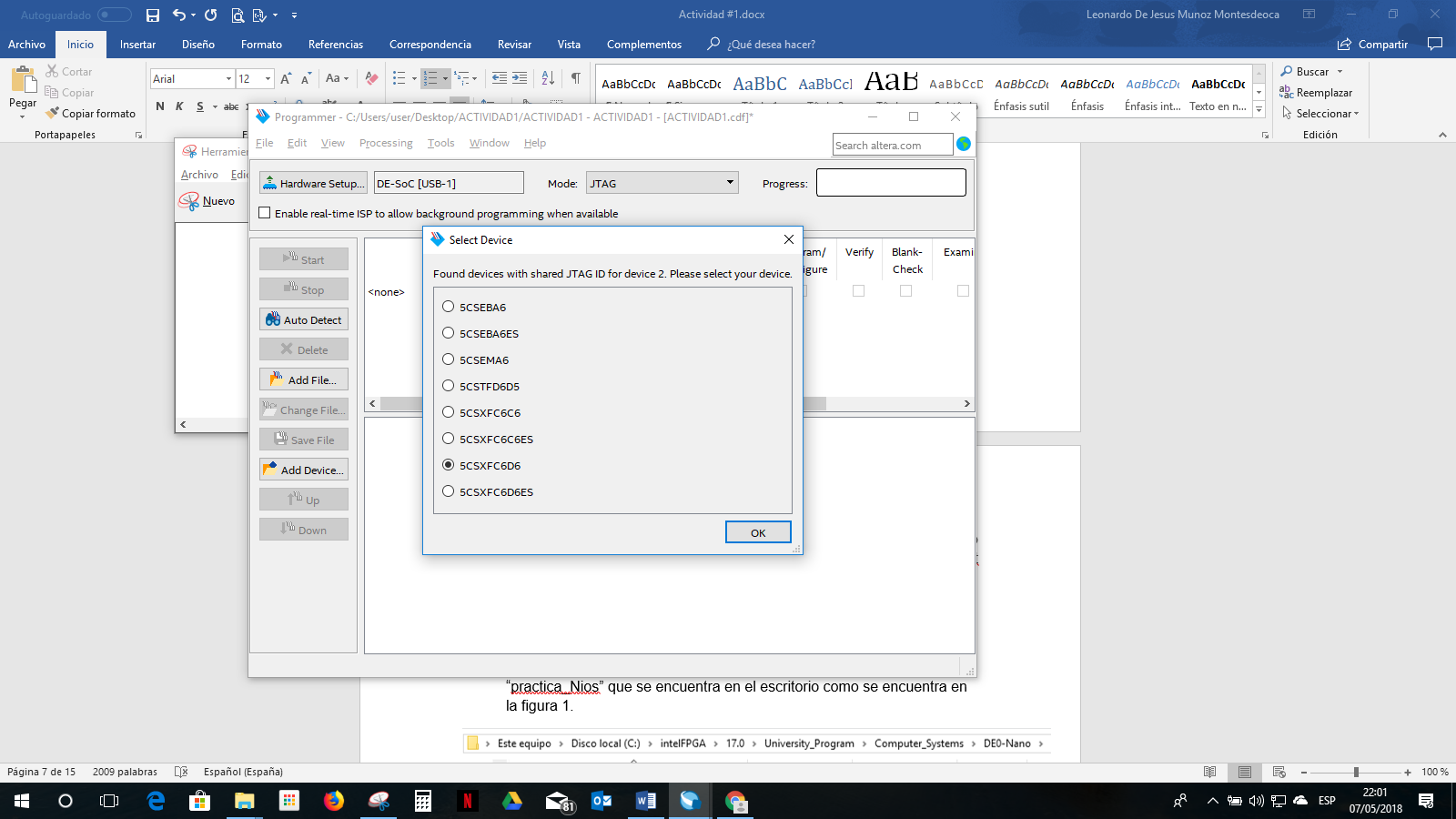


Figura 43: Ventana AUTODETEC

1. Aparecerán dos archivos: **SOCVHPS** y **5CSXFC6D6**. Haga clic derecho en el segundo y escoja la opción Change File tal como se muestra en la Figura 12, y reemplácelo por el archivo **ACTIVIDAD1.sof** que se encuentra en la carpeta **Output\_Files**.

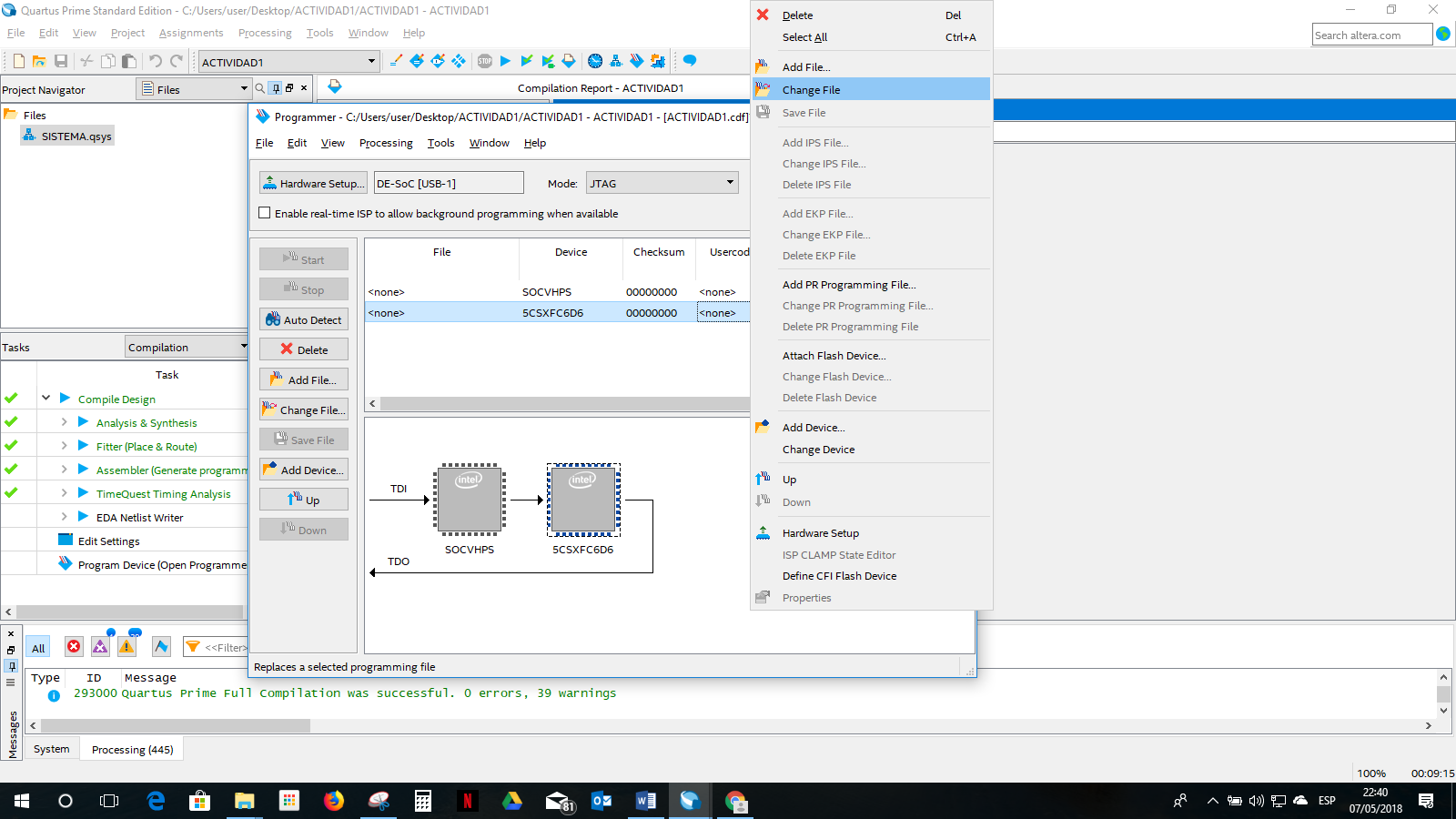


Figura 44: nuevo archivo en la ventana del programmer

1. Haga clic en la casilla correspondiente a la columna **Program/Configure** del archivo **PID.sof**, y luego click en **Start** para que se cargue la programación en el chip FPGA. Cuando la programación sea exitosa podrá apreciar que se muestra el 100 % de éxito en la ventana, tal cual se muestra en la Figura 45

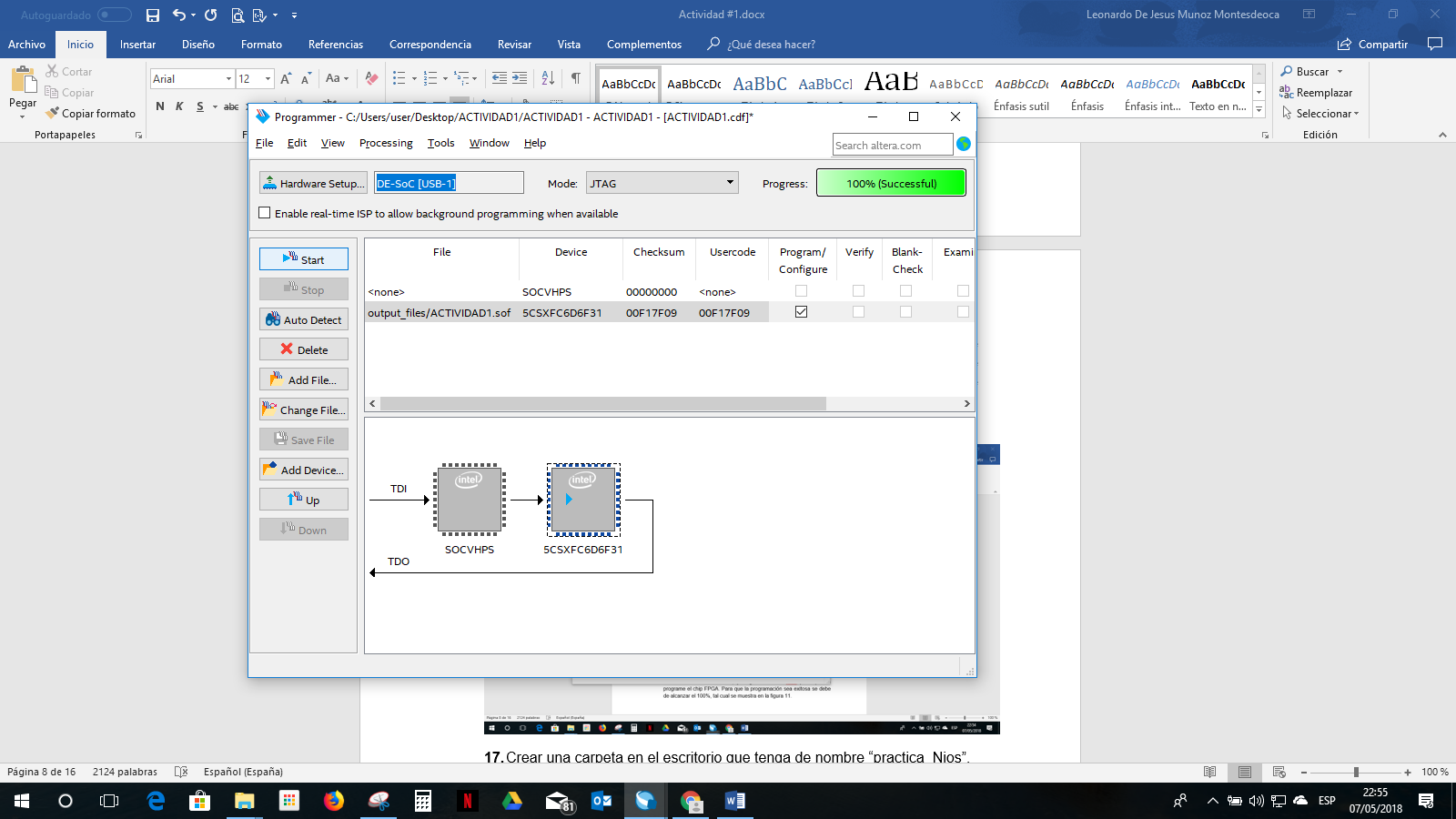


Figura 45: VENTANA DE PROGRAMACION EXITOSA

1. En este momento se realizó la construcción física de un ordenador embebido el microprocesador Nios II en el chip FPGA.
2. El siguiente paso que se realizará será la programación del microprocesador **Nios II** empleando el lenguaje C++. Seleccionaremos el menú **Tools** y elegiremos la opción **Nios II Software Build Tools for Eclipse**, como se muestra en la Figura 46

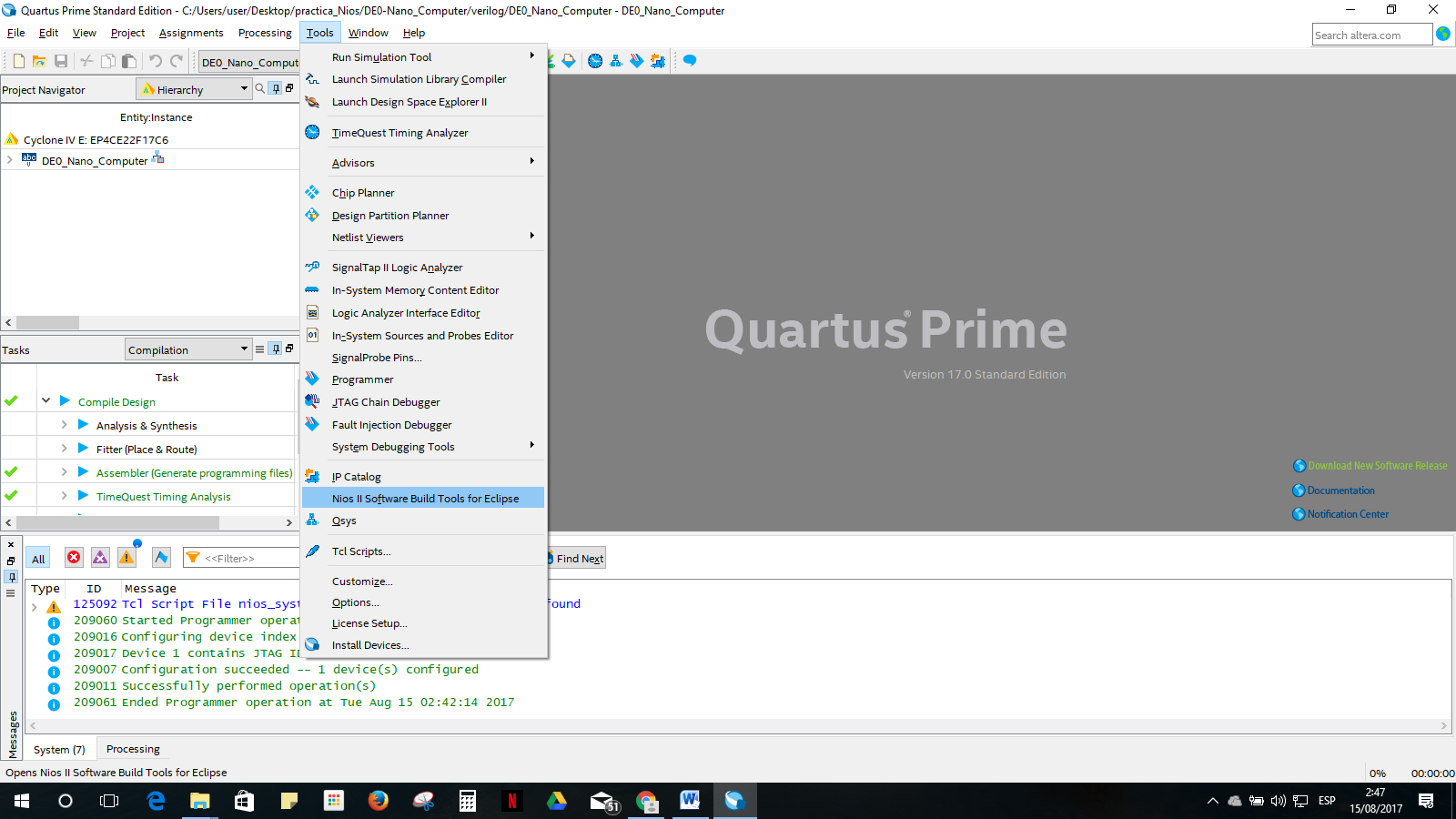


Figura 46: NIOS II ECLIPSE

1. Aparecerá una ventana donde se buscará la ubicación de los archivos del proyecto creado, y seleccionaremos la carpeta **SISTEMA**,
2. Se mostrará una ventana que corresponde al entorno de programación de ECLIPSE. Procederemos a crear un proyecto nuevo, dando clic en **File→New→Nios II Application and BSP from Template** como se visualiza en la Figura 47.

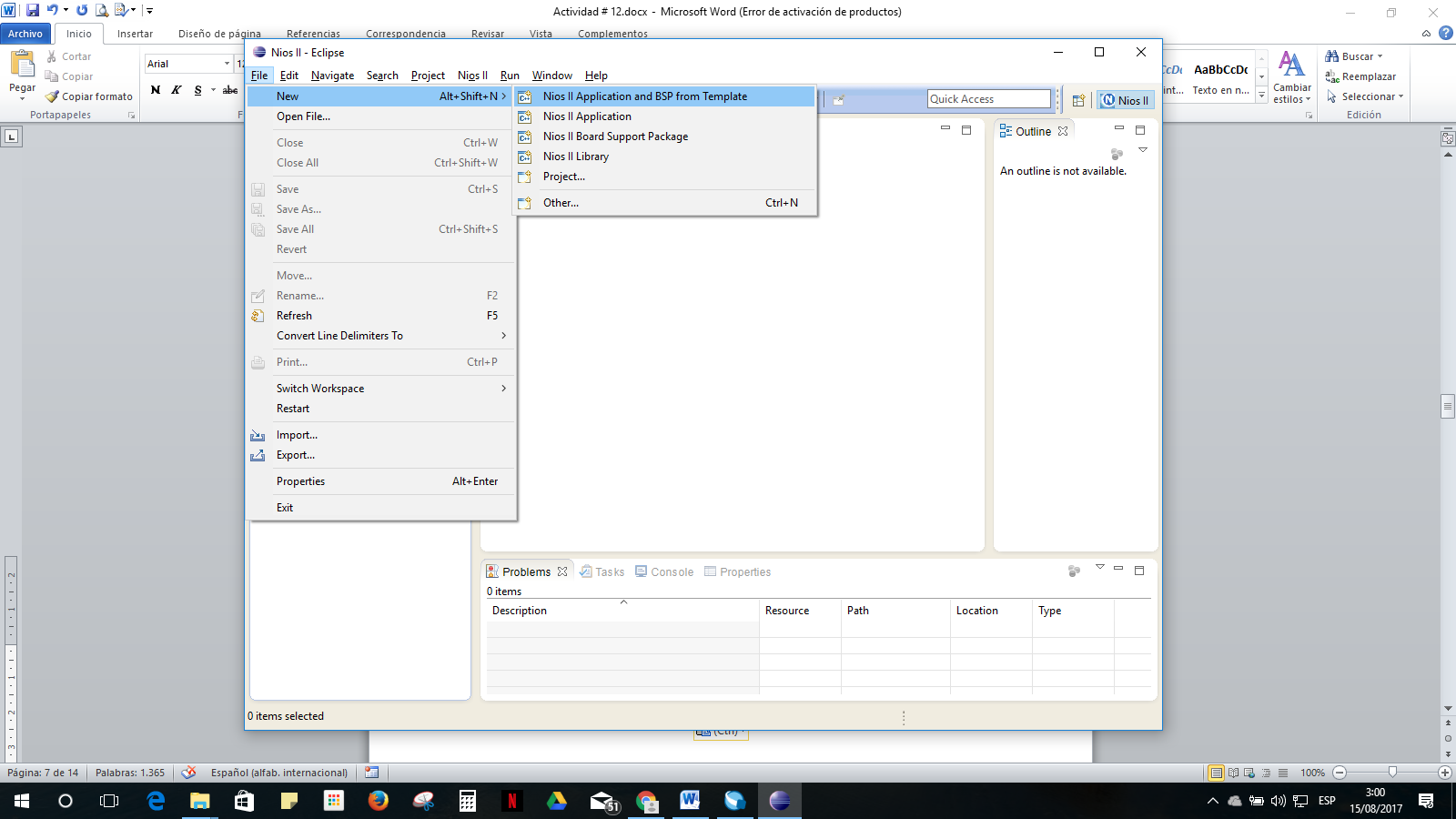


Figura 47:NUEVA APLICACION

1. En la opción **SOPC INFORMATION FILE NAME**, realizaremos la búsqueda de nuestro archivo **.sopcinfo**, que contiene todos los datos de nuestro ordenador embebido,
2. En la opción **CPU name,** utilizaremos el del ordenador embebido
3. En **Project name** elegiremos el nombre del proyecto, en este caso emplearemos **prueba1.**
4. En **Project Template,** elegiremos **Blank Project**. Y clic en finish
5. Aparecerá una subventana que corresponde al Project Explorer, donde se podrá observar dos carpetas: prueba1 y prueba1\_bsp; en la primera se creará los archivos de programación del lenguaje C/C++, de nuestro microprocesador, y la segunda corresponde al contenido del componente de nuestro hardware

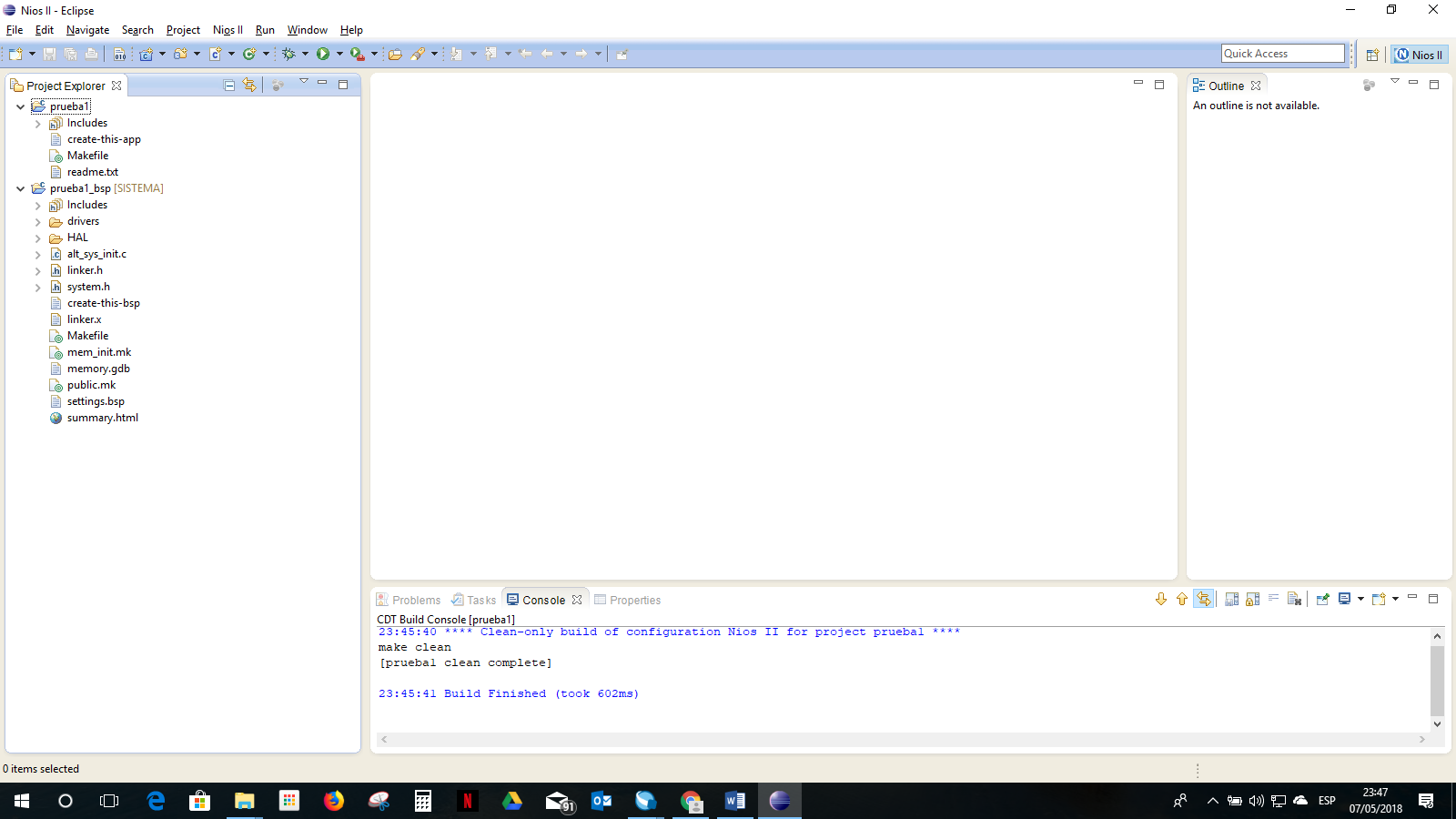


Figura 48: VENTANA PROJECT EXPLORER

1. A continuación, se creará un archivo de lenguaje C++, donde daremos clic derecho en la carpeta prueba1 y seleccionaremos **New🡪File** como se observa en la figura 50.

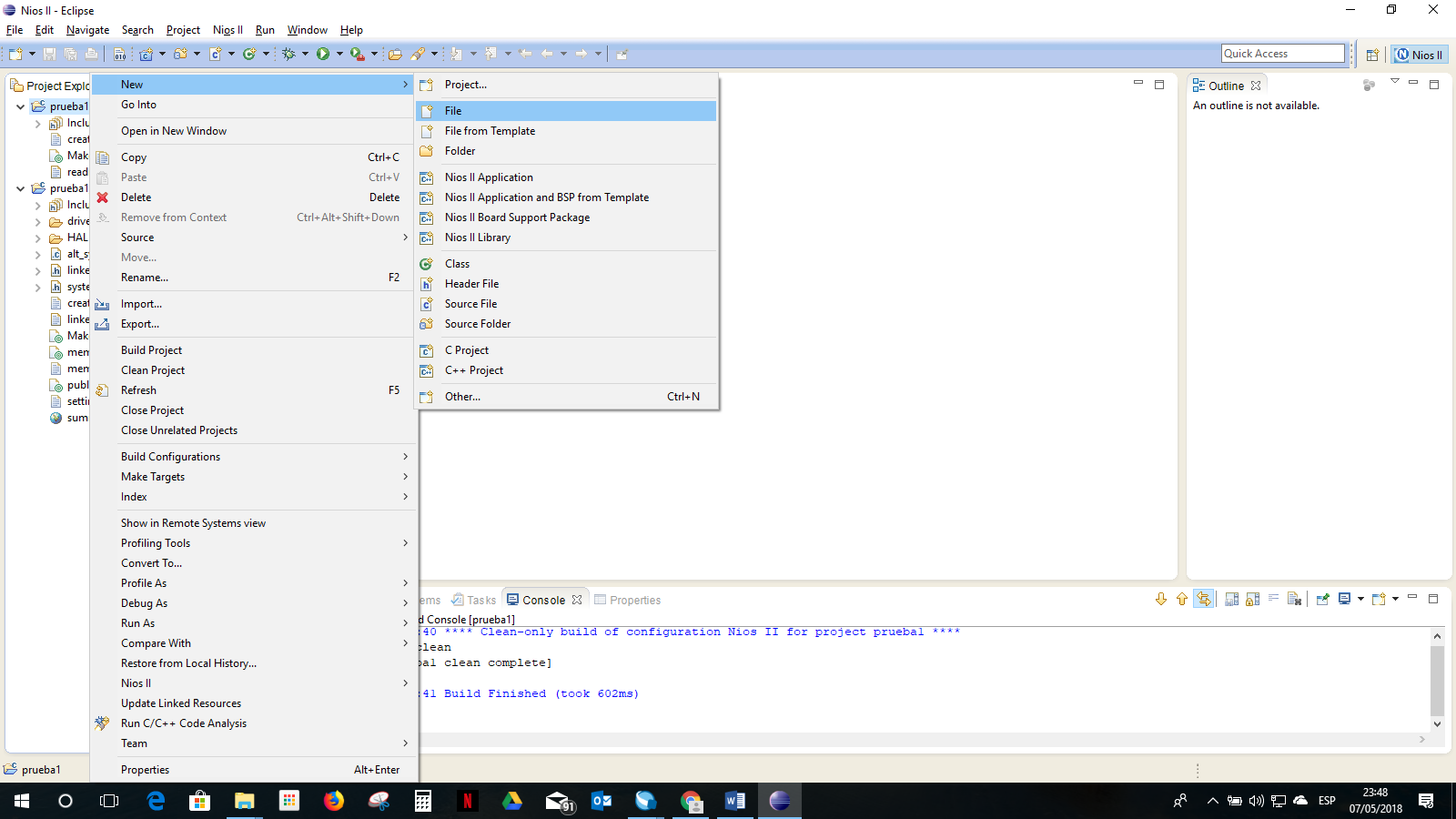


Figura 49: CREACION DE ARCHIVO C

1. EN la ventana **NEW FILE,** se colocará en nombre **main.c,** y daremos clic en **finish**.
2. Se copia el contenido del archivo adc.txt, que se encuentra en los archivos del proyecto
3. Luego clic derecho sobre la carpeta **prueba1** y elegimos la opción **Build Project** como se observa en la figura 50.
4. Para finalizar, clic derecho sobre la carpeta **prueba1** y elegimos la opción **RUN AS🡪 NIOS II HARDWARE**
5. A continuación, se realiza la conexión de los potenciómetros que usaremos para realizar el ADC, en los canales elegidos
6. Observaremos que por consola se muestra los datos correspondientes a los potenciómetros

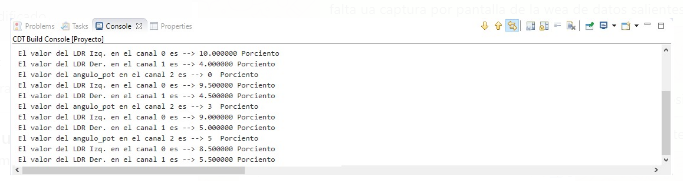


Figura ; DATOS POR CONSOLA DE ADC

**ANEXOS**

**ANEXO 1.-** Datos recogidos por medio de Arduino para encontrar la ecuación de la planta del sistema para su posterior compensación por medio del controlador PID.

Los datos fueron recolectados por medio de la tarjeta Arduino, una vez terminada la recolección se procedió a realizar la identificación del sistema con ayuda del programa Matlab, donde se escogerá la mejor forma de representar el controlador para de esta manera realizar una aproximación de al menos 90 % o superior a este valor.

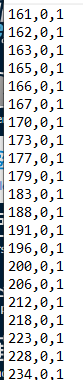


Figura 51; En la siguiente imagen se muestra algunos Datos recolectados mediante arduino

**Anexo 2. –** Respuesta de la función de transferencia de la planta frente. En la figura se muestra dos respuestas del sistema

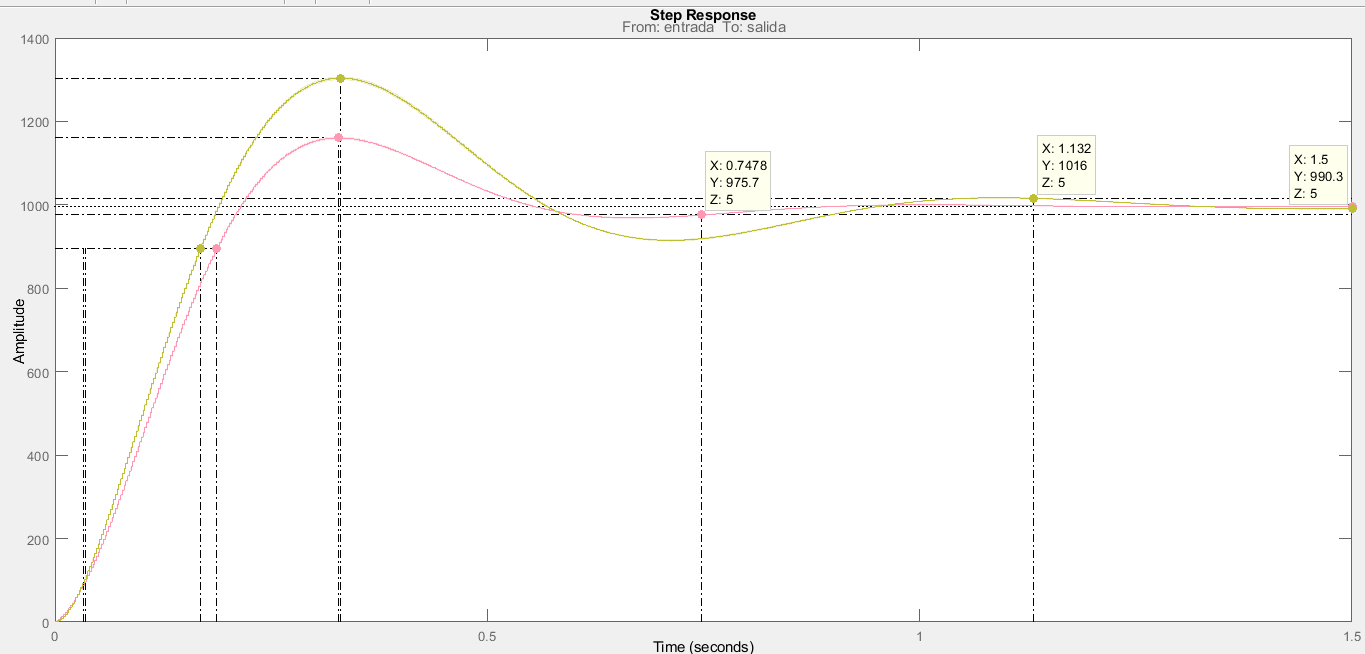


Figura 52: Respuesta de función de transferencia de la planta

**Anexo 3. – Codigo de Arduino empleado**

const int analogInPin = A0;

#define in1 2

#define in2 3

//const unsigned long t\_high = 1000000;

//const unsigned long t\_period = 2000000;

const long interval = 2000;

unsigned long t;

unsigned long previousMillis;

unsigned long startMillis;

unsigned long currentMillis;

int potValue = 0;

void setup() {

Serial.begin(250000);

startMillis = micros();

pinMode(in1, OUTPUT);

pinMode(in2, OUTPUT);

}

void loop() {

currentMillis = micros();

potValue = analogRead(analogInPin);

t = currentMillis - startMillis;

if (potValue < 200) { digitalWrite(in2, HIGH); digitalWrite(in1 , LOW); }

if (potValue > 800) { digitalWrite(in1, HIGH); digitalWrite(in2 , LOW); }

if (currentMillis - previousMillis >= interval) {

Serial.print(potValue);

Serial.print(",");

Serial.print(digitalRead(in1));

Serial.print(",");

Serial.println(digitalRead(in2));

previousMillis = currentMillis;

}

}

**ANEXO 4: código de ADC**

#include <stdio.h>

#include "io.h"

#include <math.h>

#include <system.h>

#include <stdlib.h>

/======= main =========/

int main (void) {

volatile int \* ADC = (int \*) 0x41020;//asignar la memoria del ADC

int \* memoria = (int \*) 0x20000; // SDRAM address

//Variable de lectura

unsigned int ldr1;

unsigned int ldr2;

unsigned int angulo\_pot;

//Canales mediciones

unsigned int canal\_ldr1;

unsigned int canal\_ldr2;

unsigned int canal\_angulo\_pot;

\*(ADC) = 0; //setear valores de los canales en 0

while (1){

//Asignar valores a los canales

canal\_ldr1 = 0;

canal\_ldr2 = 1;

canal\_angulo\_pot = 2;

//ldr izquierdo

ldr1 = (\*(ADC+canal\_ldr1))\*100/4095; //Obtener el valor del ldr

printf("El valor del LDR Izq. en el canal %d es --> %d Porciento \n",canal\_ldr1,ldr1);

//ldr derecho

ldr2 = (\*(ADC+canal\_ldr2))\*100/4095; //Obtener el valor del ldr

printf("El valor del LDR Der. en el canal %d es --> %d Porciento \n",canal\_ldr2,ldr2);

//Potenciometro que permite orientar el sistema

angulo\_pot = ((\*(ADC+canal\_angulo\_pot))\*70/4095)+20;

printf("El valor del angulo\_pot en el canal %d es --> %d Porciento\n",canal\_angulo\_pot,angulo\_pot);

//Escritura en memoria

IOWR(memoria,0x20000,ldr1);

IOWR(memoria,0x20100,ldr2);

IOWR(memoria,0x20200,angulo\_pot);

}

return 0;

}