**DISEÑO DE SISTEMAS DIGITALES**

**ACTIVIDAD # 1**

**CAPÍTULO DEL CURSO:** LENGUAJE C++ PARA SISTEMAS EMBEBIDOS.

**TEMA DE LA ACTIVIDAD:** USO DEL LENGUAJE C++ PARA SISTEMAS EMBEBIDOS.

**NOTA**

**OBJETIVOS DE APRENDIZAJE:**

* Elaborar un algoritmo en lenguaje C++ utilizando el microprocesador Nios II, para la verificación de Clases orientada a la programación de objetos mediante la salida por consola.

**DURACIÓN:** 60 minutos

**MATERIALES Y HERRAMIENTAS:**

* Software Quartus 17.0 versión Estándar.
* Software Eclipse Kepler.
* Tarjeta de desarrollo DE10-Standard.

**INTRODUCCIÓN**

El lenguaje C++ representa el resultado de los esfuerzos realizados para proporcionar las ventajas de la programación Orientada a Objetos a un lenguaje clásico, muy extendido en la programación de sistemas. Se trata de una extensión del lenguaje C que presenta claras influencias del lenguaje Simula, que a su vez puede considerarse como el precursor de los lenguajes POO.

En C++ un programa consta de objetos. Los objetos son como los tipos abstractos de datos: · Son un tipo de datos definido por el programador junto con un conjunto de operaciones que se pueden realizar sobre ellos. · Son abstractos para diferenciarlos de los tipos de datos básicos definidos, por ejemplo, int, char y double. Un objeto es una unidad que contiene datos y las funciones que operan sobre esos datos. A los elementos de un objeto se los conoce como miembros: las funciones que operan sobre los objetos se denominan métodos (o funciones miembro) y los datos se denominan miembros datos. Los objetos soportan una serie de características específicas de los mismos:

* Se agrupan en tipos denominados clases.
* Contienen datos internos que definen su estado actual.
* Soportan ocultación de datos.
* Pueden heredar propiedades de otros objetos.
* Pueden comunicarse con otros objetos enviando o pasando mensajes.
* Tienen métodos que definen su comportamiento.

Los objetos pueden ser activados mediante la recepción de mensajes. Un mensaje es una petición de un objeto a otro para que éste se comporte de una determinada manera, ejecutando uno de sus métodos, es decir los objetos de un programa se comunican entre sí mediante el paso o envío de mensajes [1].

**Clases**

El lenguaje C++ entiende por clase un tipo de dato definido por el programador que contiene toda la información necesaria para construir un objeto de dicho tipo y el conjunto de operaciones que permiten manejarlo (métodos). Las clases de C++ le permiten disfrutar de los beneficios de la modularidad, de la misma manera que en lenguajes tales como Ada o Modula-2. La construcción de las clases se hace de tal manera que se asegura el encapsulamiento de los objetos. Las clases en C++ soportan los conceptos:

* Encapsulamiento de datos.
* Tipos abstractos de datos.

El encapsulamiento de datos se produce cuando la representación interna de un objeto junto con sus operaciones, se encierran en la misma estructura (la clase). La clase define un tipo abstracto de dato.

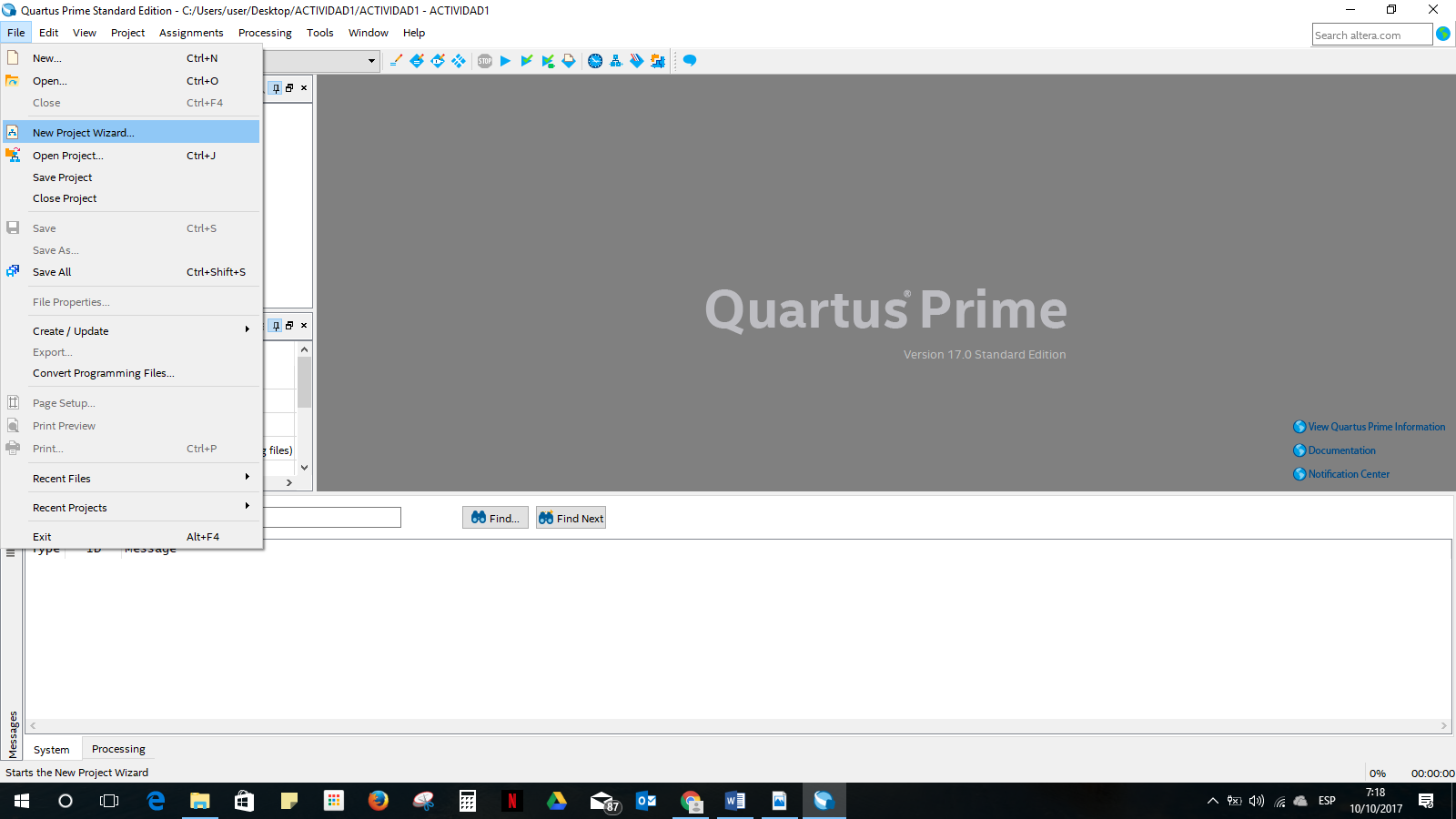
Una clase contiene elementos dato llamado miembros dato, y funciones que manipulan esos datos llamados ***miembros función*** o ***funciones miembro***.

Una clase se puede definir con una estructura (struct), una unión (union) o una clase (class). En el tipo class todos los miembros son de manera predeterminada privados. Las estructuras y uniones son clases en las que todos sus miembros son predeterminadamente public, a no ser que se especifique lo contrario utilizando los especificadores de acceso [1].

**DESCRIPCIÓN DE LA PRÁCTICA:** Se realizará la codificación de un algoritmo en lenguaje C++ y se ejecutará el mismo para mostrar la salida por consola.

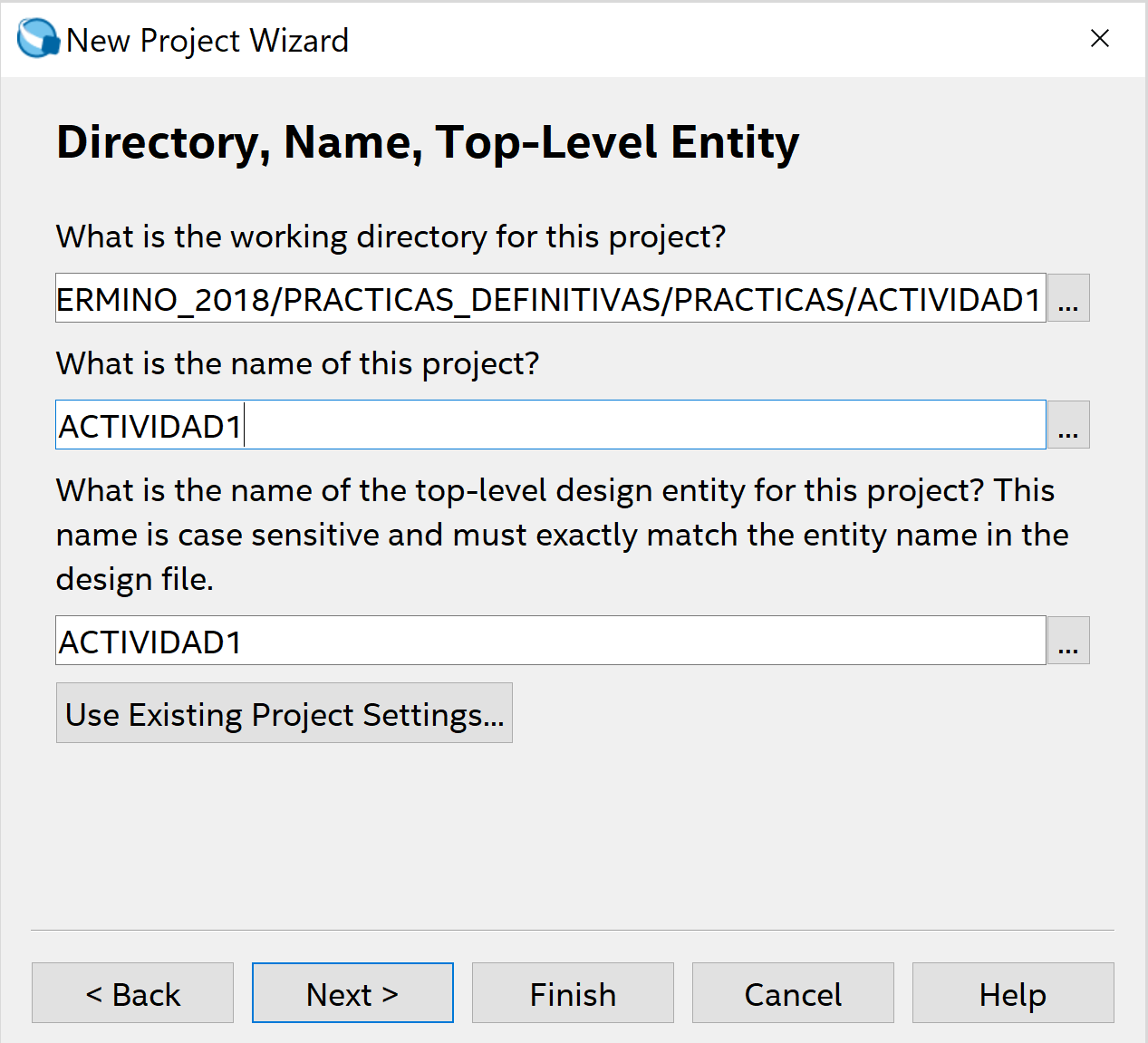
**PROCEDIMIENTO:**

1. Proceda a crear una carpeta en el escritorio con el nombre ACTIVIDAD1 (escriba sin espacios). Luego copie y pegue todos los archivos que se encuentran en la carpeta compartida, en la carpeta que fue creada en el escritorio.
2. Ejecute el programa Quartus Prime 17.0 Standard Edition que se encuentra en el escritorio representado por el ícono .
3. Proceda a crear un nuevo Proyecto seleccionando la opción **File→New Project Wizard** como se observa en la Figura 1. Aparecerá la ventana de Introducción, luego haga clic en **Next**.



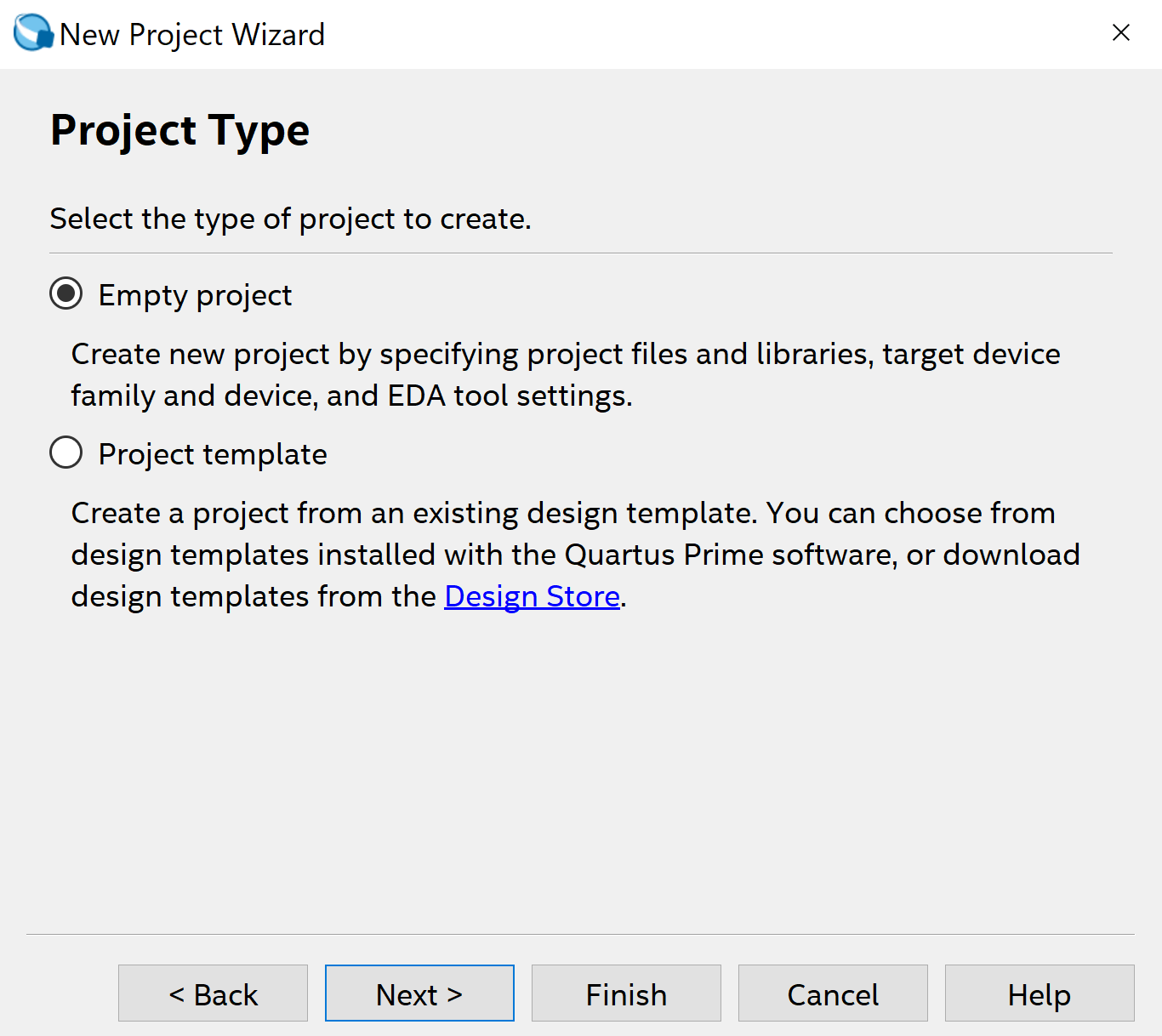
**Figura 1. Creación de un nuevo proyecto.**

1. En la primera línea escoge la ruta donde va a crear su proyecto, en este caso, la ruta será la carpeta ACTIVIDAD1 que fue creada en el procedimiento 1 y que se encuentra en el escritorio. En la siguiente línea escoja como nombre del proyecto ACTIVIDAD1, tal como se visualiza en la Figura 2. Luego haga clic en Next.



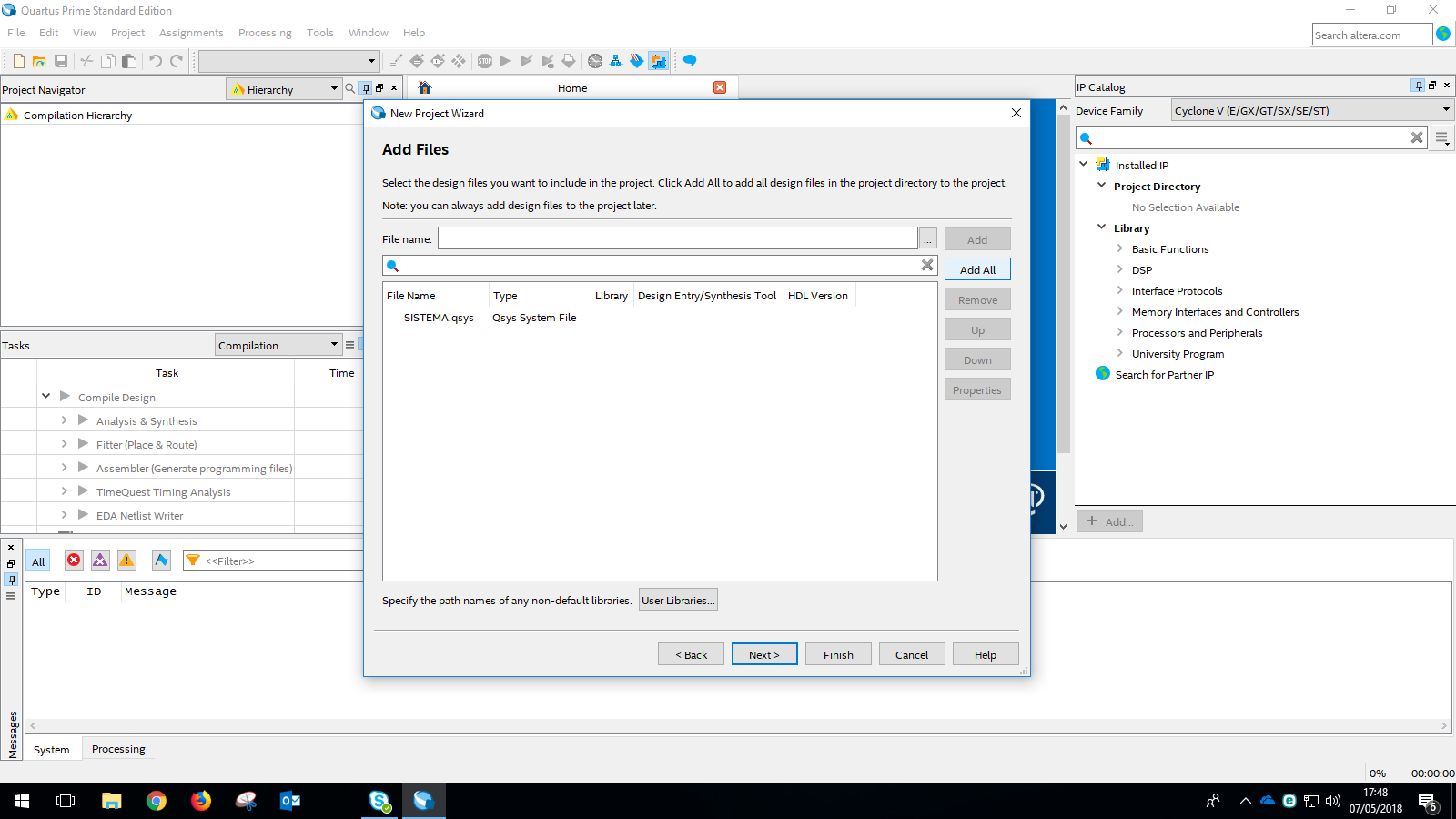
**Figura 2. Nombre del proyecto.**

1. Se mostrará la ventana para seleccionar el tipo de proyecto. Se escogerá la opción de Proyecto vacío (**Empty project**), tal como se visualiza en la Figura 3. Luego haga clic en **Next**.



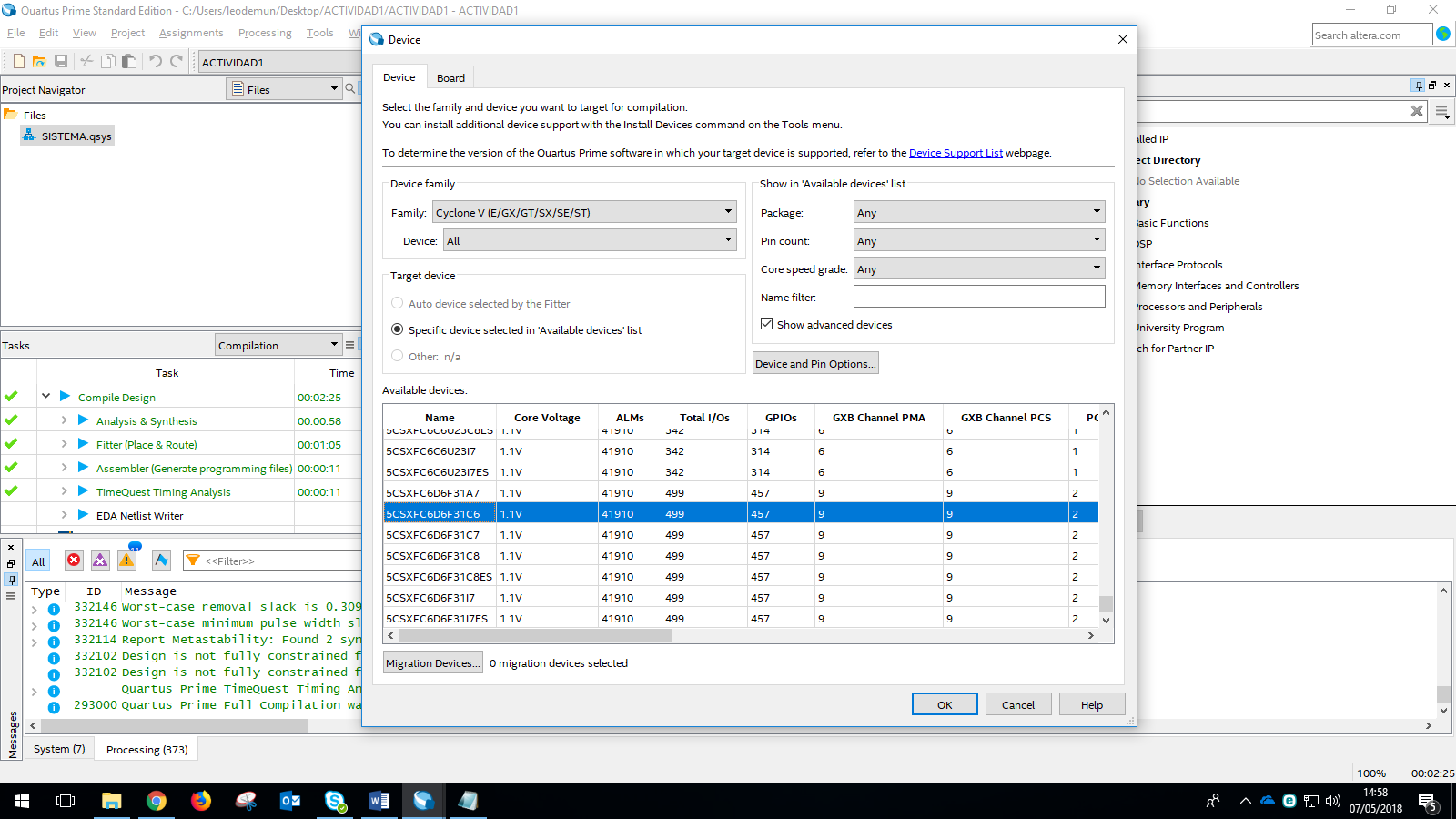
**Figura 3. Tipo del proyecto**

1. Se mostrará la ventana para adhesión de archivos. En esa ventana seleccione la opción **Add All**, para que se agreguen todos los archivos que fueron copiados a la carpeta del proyecto y que se realizó en el procedimiento 1. En la Figura 4 se observan los archivos añadidos. Por último, haga clic en **Next**.



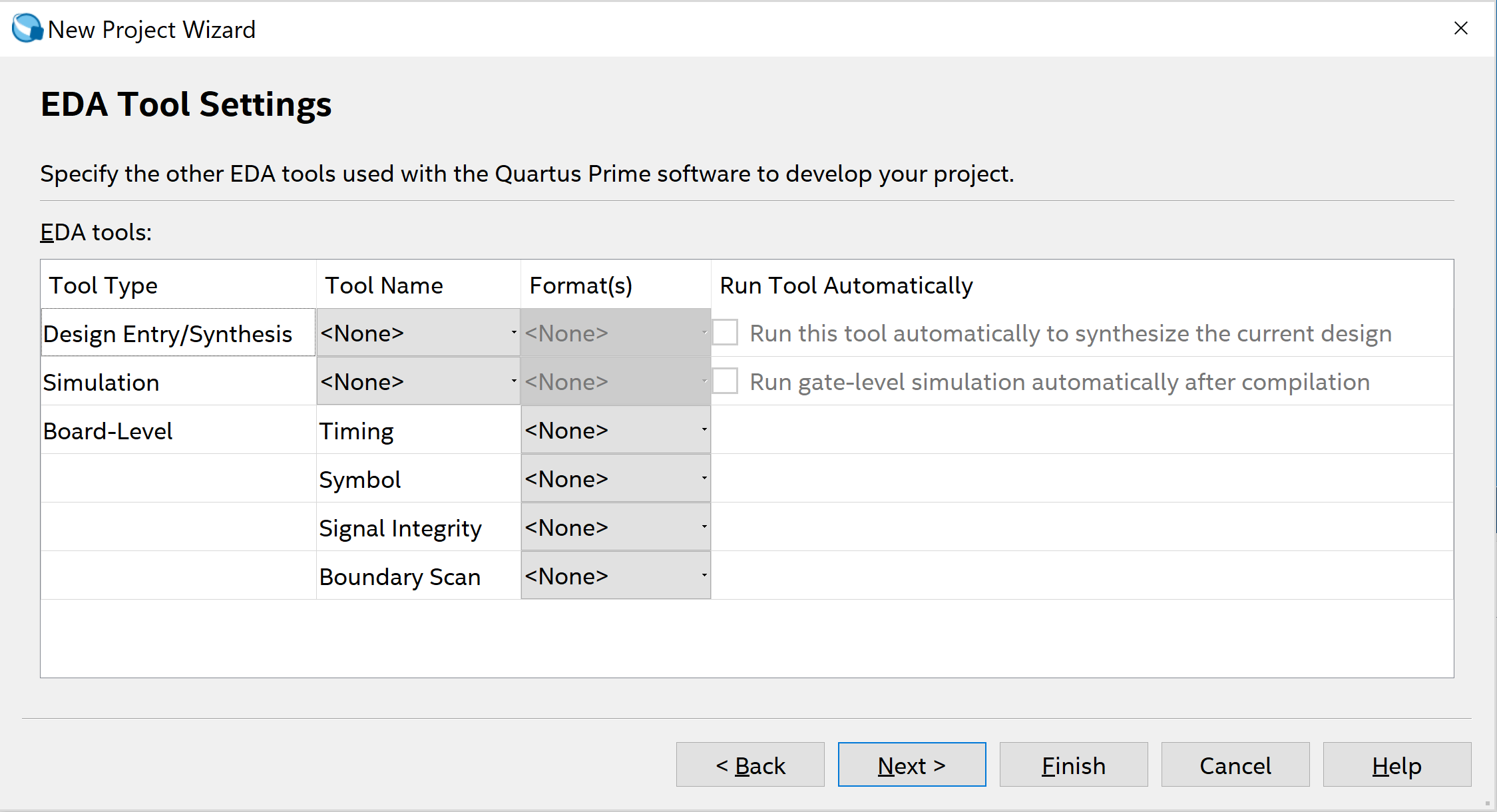
**Figura 4. Adhesión de archivos**

1. En la siguiente ventana aparecerán todas las familias de chips FPGA. Proceda a escoger la familia **Cyclone V (E/GX/GT/SX/SE/ST)** y seleccione el nombre del chip FPGA SoC **5CSXFC6D6F31C6**, tal como se muestra en la Figura 5. Luego haga clic en **Next**.



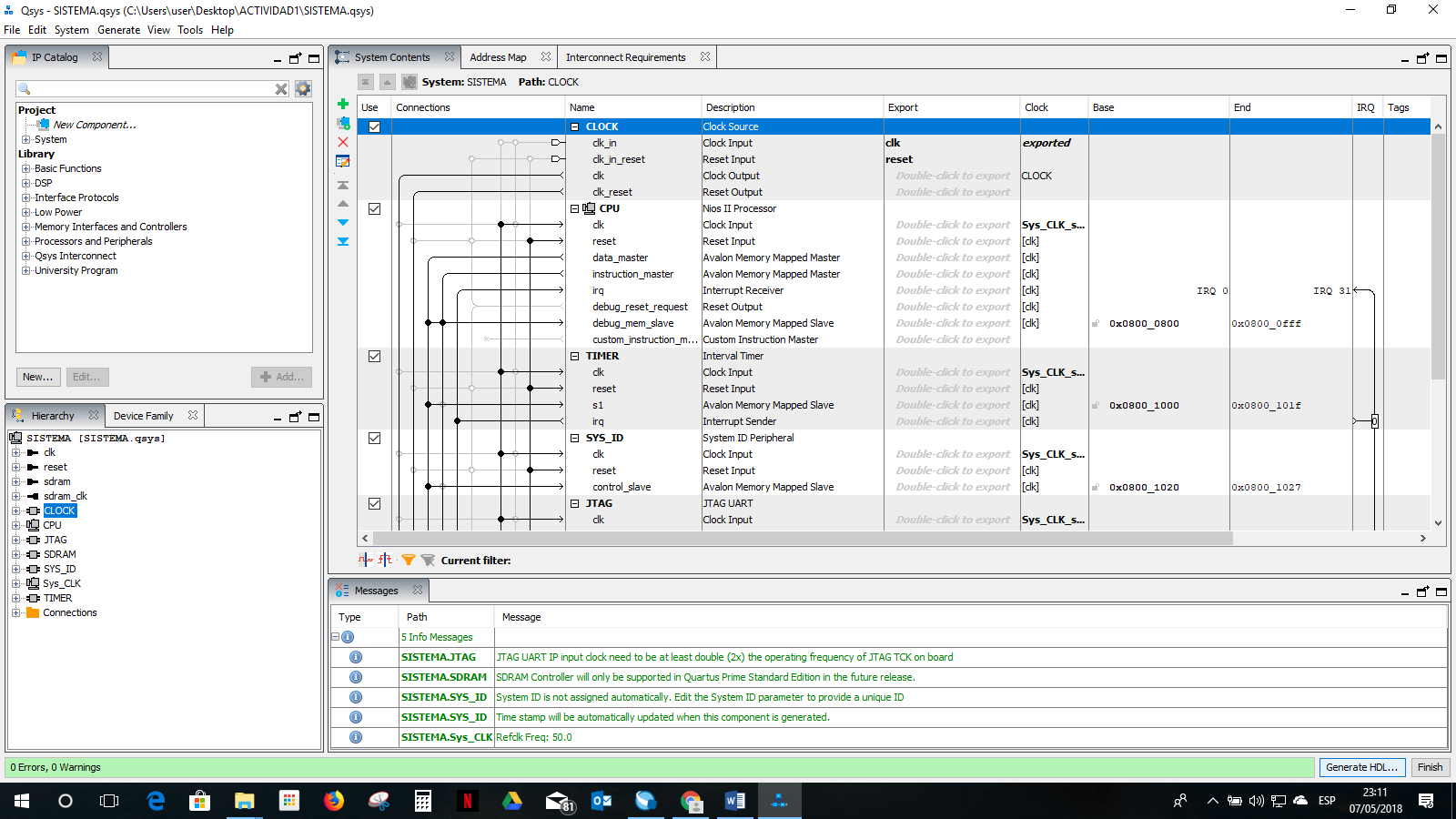
**Figura 5. Familia y modelo del chip FPGA SoC**

1. Debido a que no se realizarán simulaciones en la presente actividad, no se seleccionará alguna herramienta de simulación como se muestra en la Figura 6. Haga clic en **Next**.



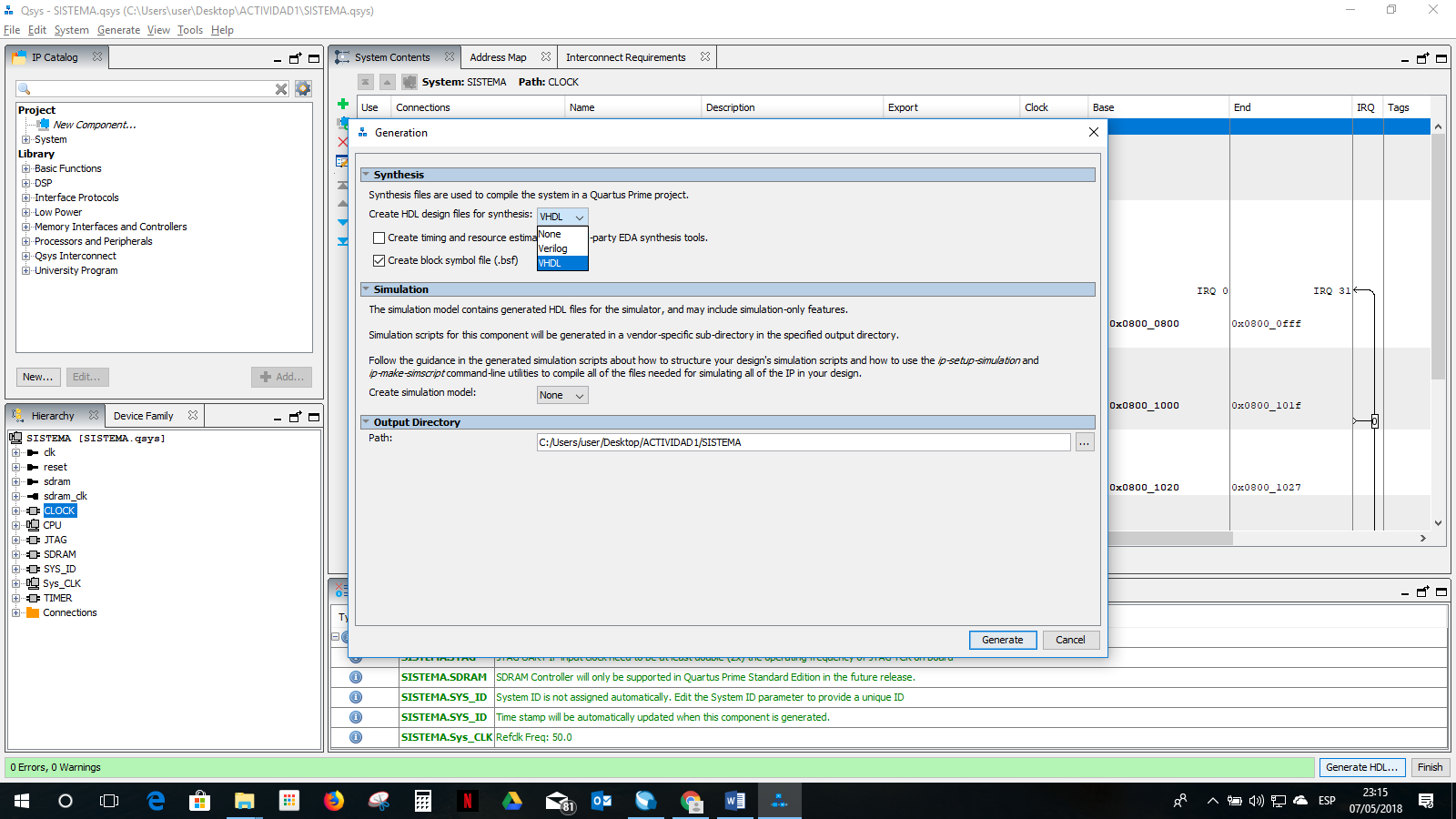
**Figura 6. Herramientas para simulación**

1. Al finalizar la creación del proyecto, aparecerá la ventana **Summary** para resumir todas las especificaciones que tendrá nuestro proyecto. Para finalizar haga clic en **Finish**.
2. Haga doble clic al archivo **SISTEMA.qsys** dentro del programa **Quartus Prime**. Dentro de la ventana **Qsys**, haga clic en **Generate HDL** que se encuentra en la parte inferior derecha, tal como se muestra en la Figura 7.



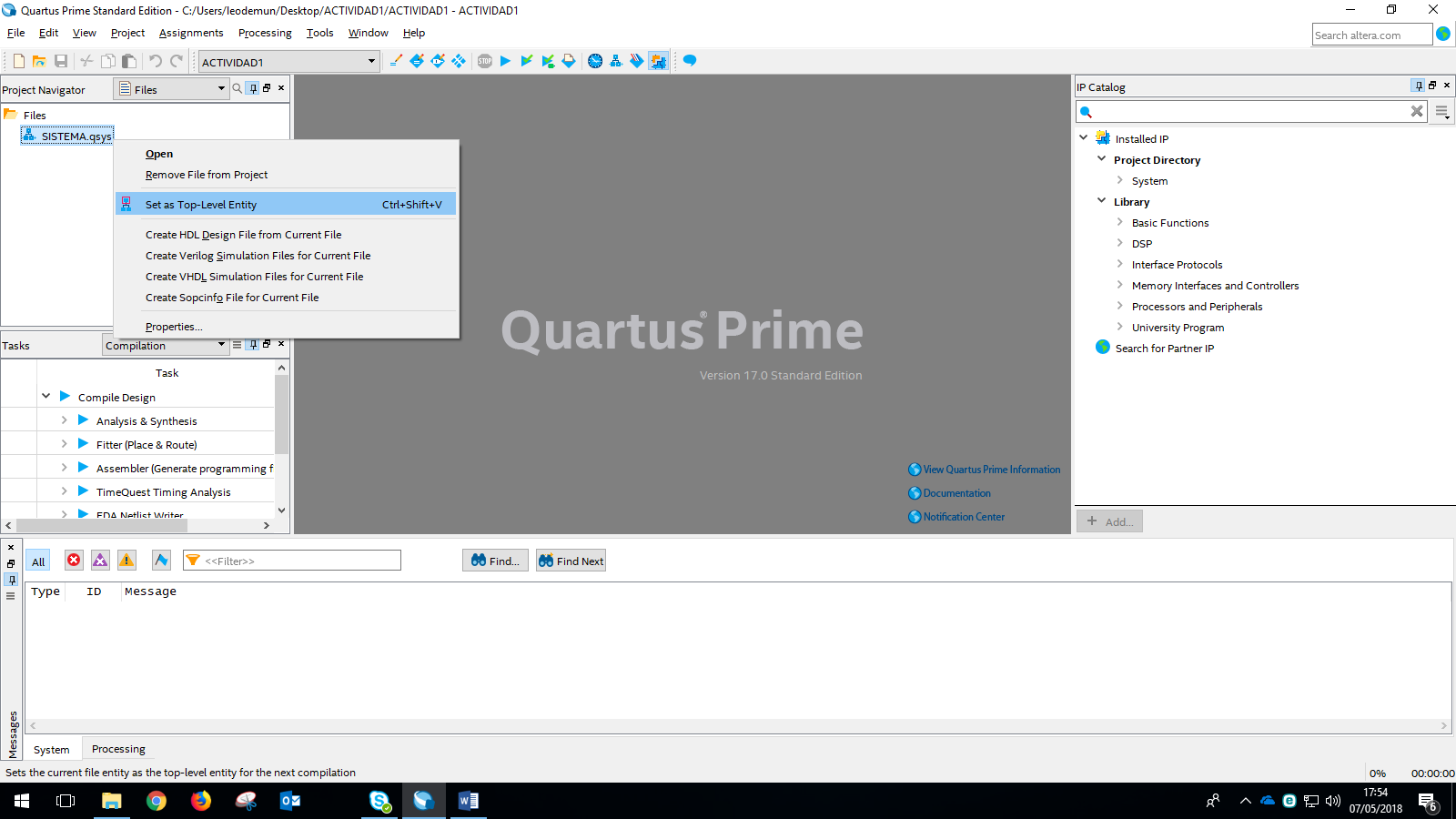
**Figura 7. Ventana Qsys**

1. En la ventana **Generation**, seleccione la creación de archivos VHDL en la sección **Synthesis**, tal como se muestra en la Figura 8. Por último, haga clic en **Generate**, y cuando finalice haga clic en **Finish**.



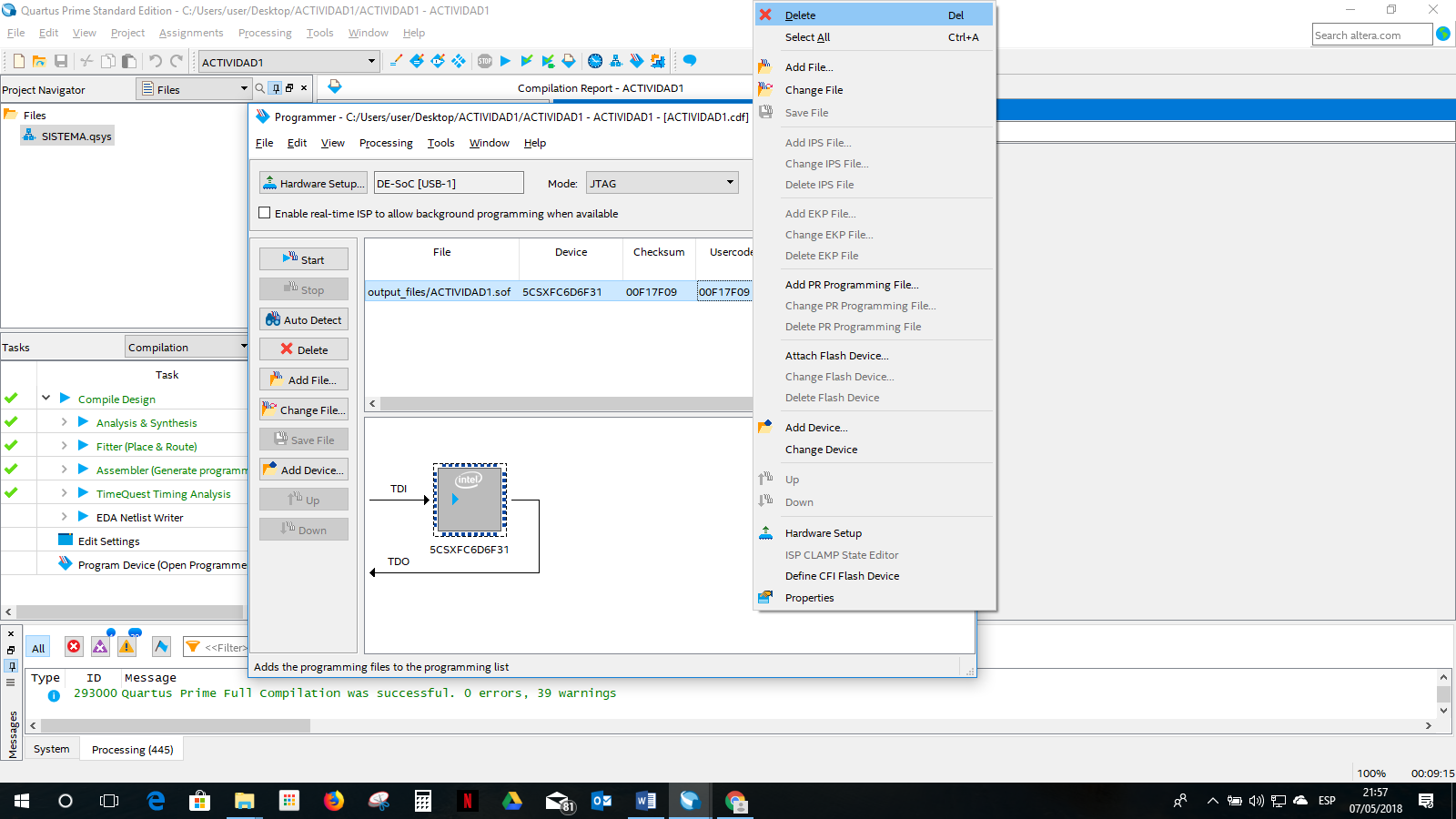
**Figura 8. Ventana Generation**

1. Proceda a compilar el archivo **SISTEMA.qsys** estableciéndolo previamente como más alta jerarquía (dando clic derecho en el mismo y seleccionando la opción **Set as Top-Level Entity**), como se muestra en la Figura 9. Luego seleccione el ícono .



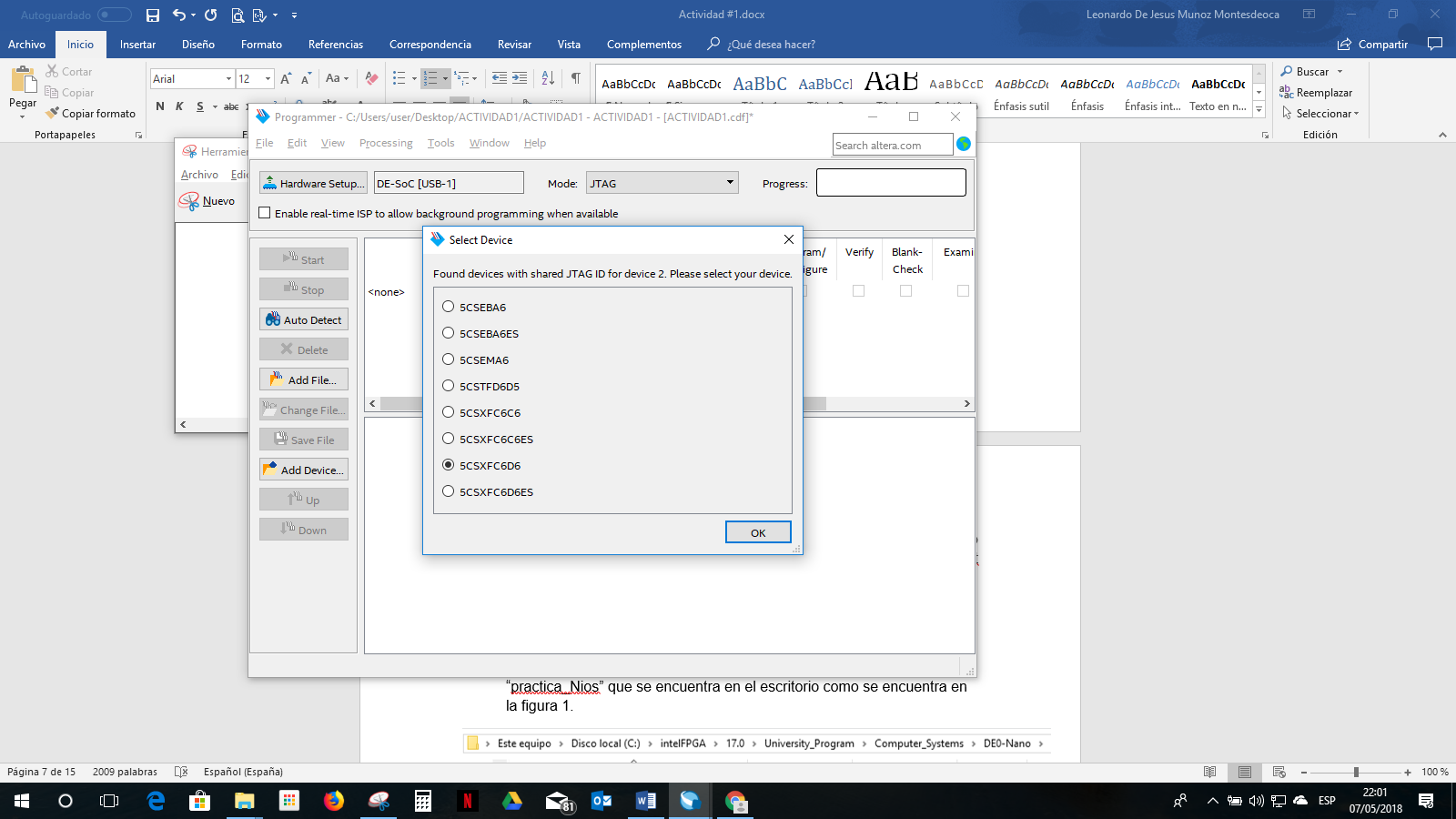
**Figura 9. Compilación del proyecto**

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.



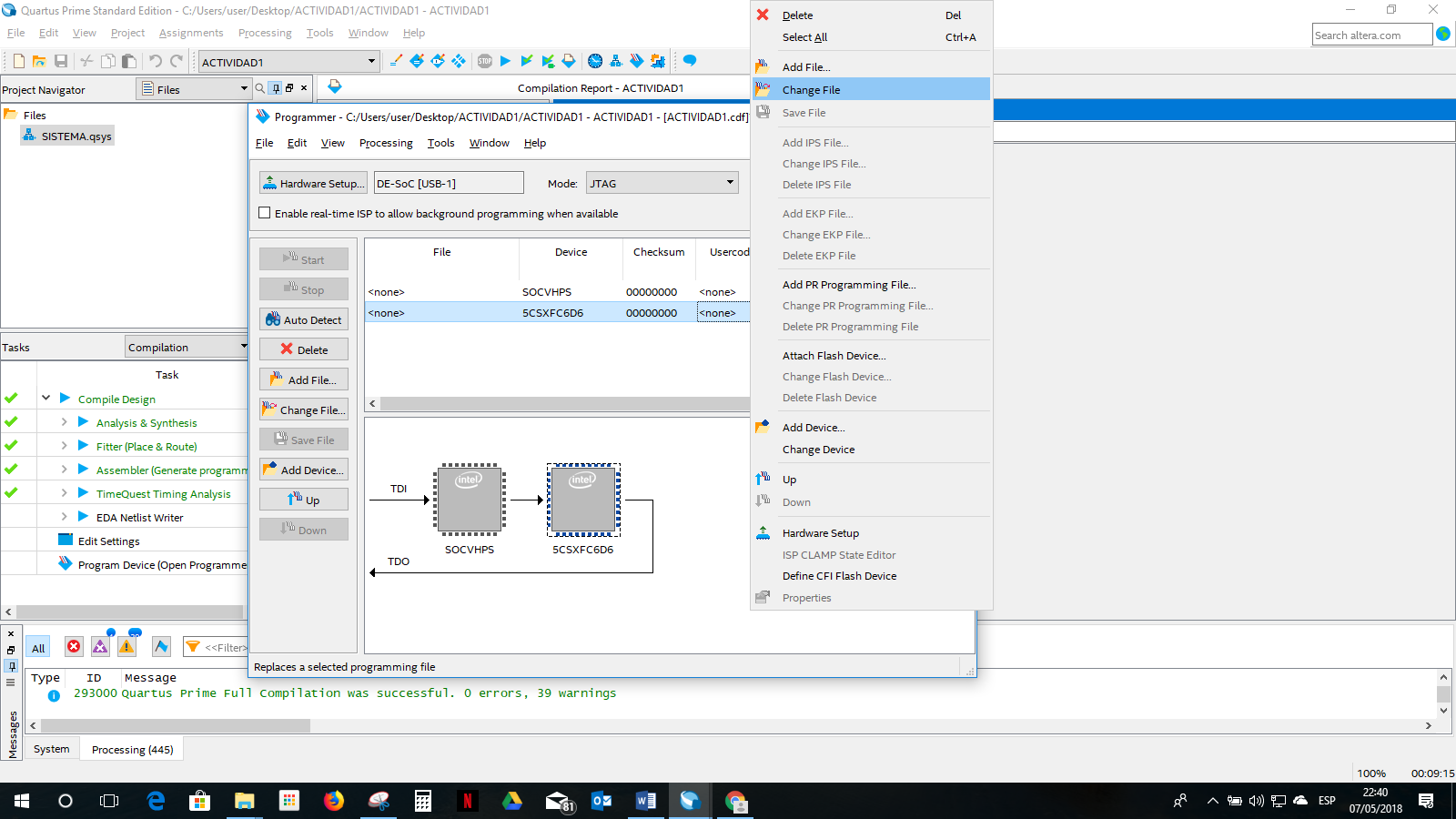
**Figura 10. Ventana Programmer**

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.



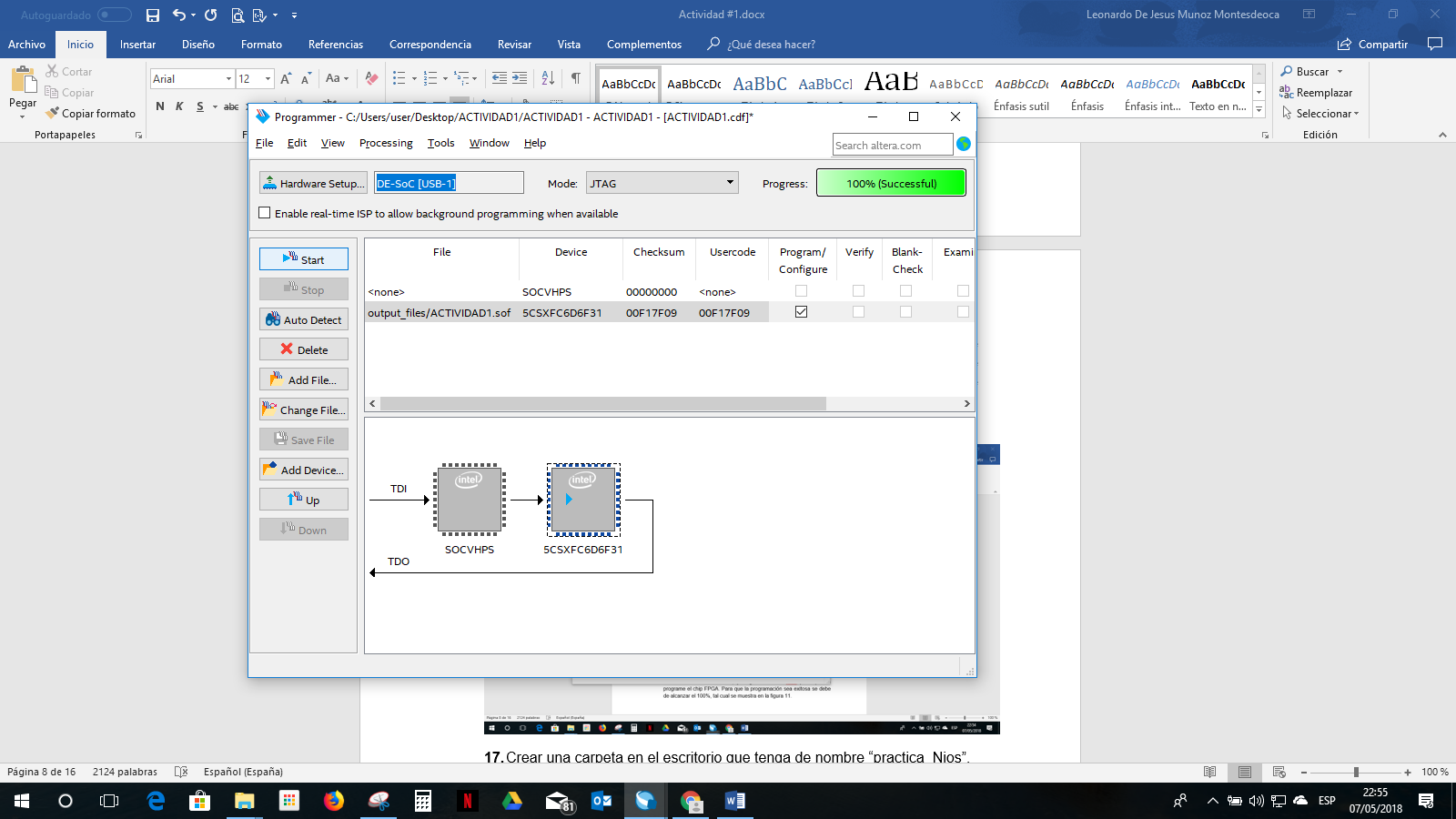
**Figura 11. Ventana Auto Detect**

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**.



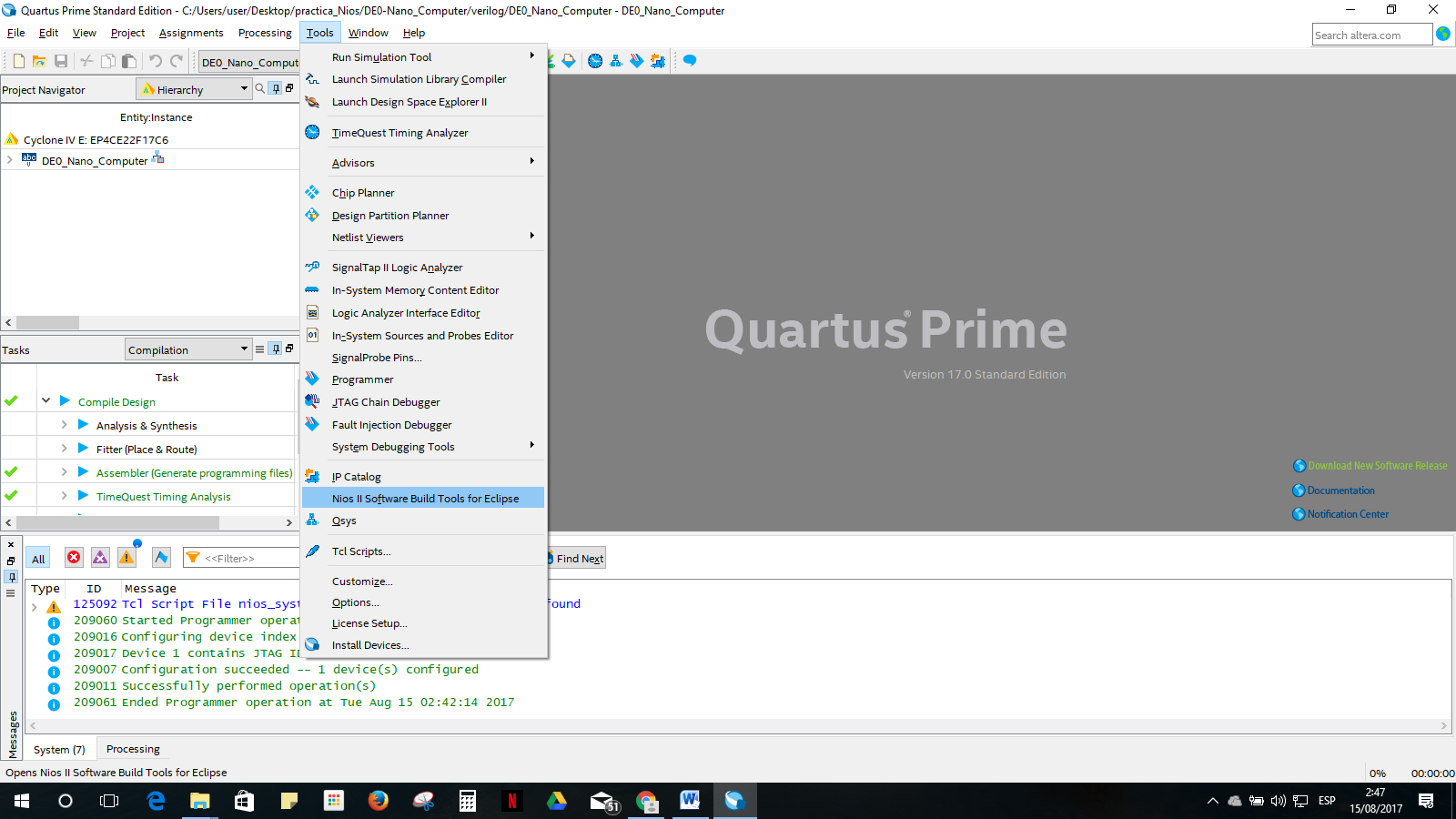
**Figura 12. Cambio de archivo en la ventana Programmer**

1. Haga clic en la casilla correspondiente a la columna **Program/Configure** del archivo **ACTIVIDAD1.sof**, y luego seleccione **Start** para que se programe el chip FPGA. Para que la programación sea exitosa se debe de alcanzar el 100%, tal cual se muestra en la Figura 13.



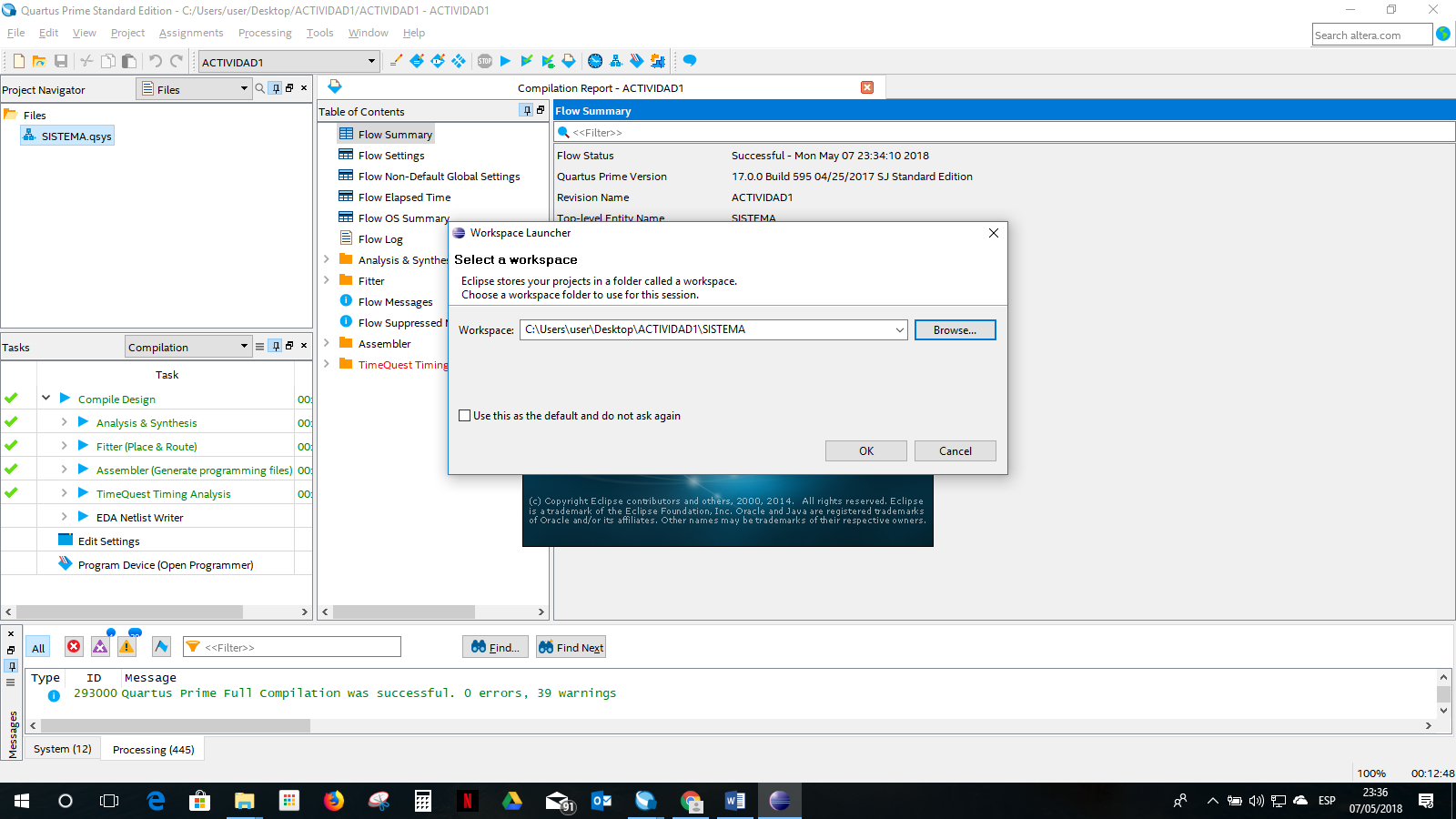
**Figura 13. Programación exitosa del chip FPGA**

1. En este momento se acaba de construir físicamente la computadora embebida con el microprocesador Nios II en el chip FPGA.
2. Lo siguiente que se realizará será la programación del microprocesador **Nios II** utilizando lenguaje C++. En el menú **Tools**, seleccione la opción **Nios II Software Build Tools for Eclipse**, como se muestra en la Figura 14.



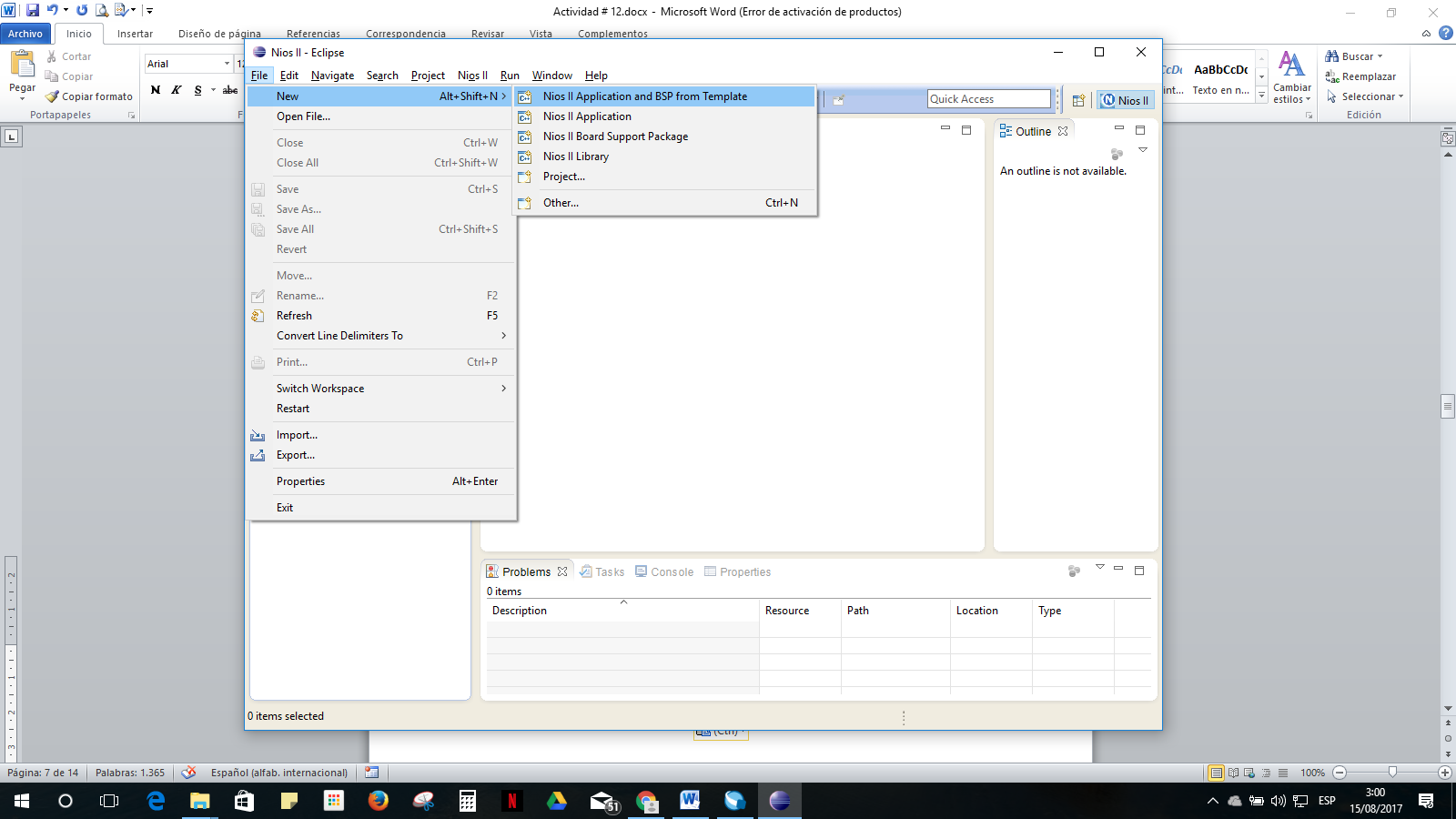
**Figura 14. Selección de la herramienta Eclipse**

1. Lo primero que pedirá Eclipse será ubicar en donde va a estar alojado el proyecto que vayamos a crear, seleccione la carpeta **SISTEMA** que se encuentra dentro de la carpeta del proyecto, como se encuentra en la Figura 15. Haga clic en OK.



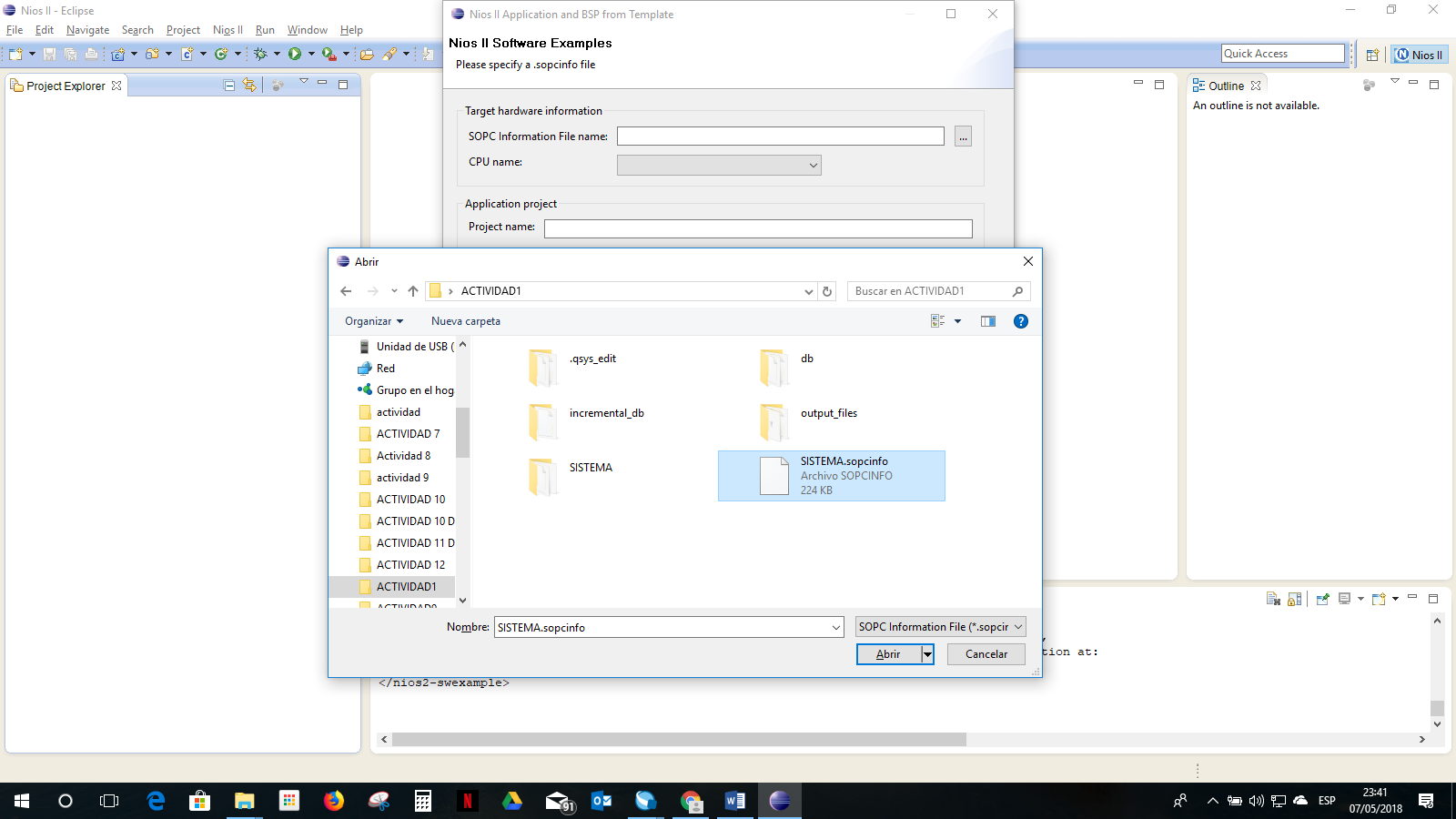
**Figura 15. Ubicación del proyecto en Eclipse**

1. Se abrirá el entorno de programación de Eclipse. Se procederá a crear un nuevo proyecto, para ello seleccione **File→New→Nios II Application and BSP from Template** como se visualiza en la Figura 16.



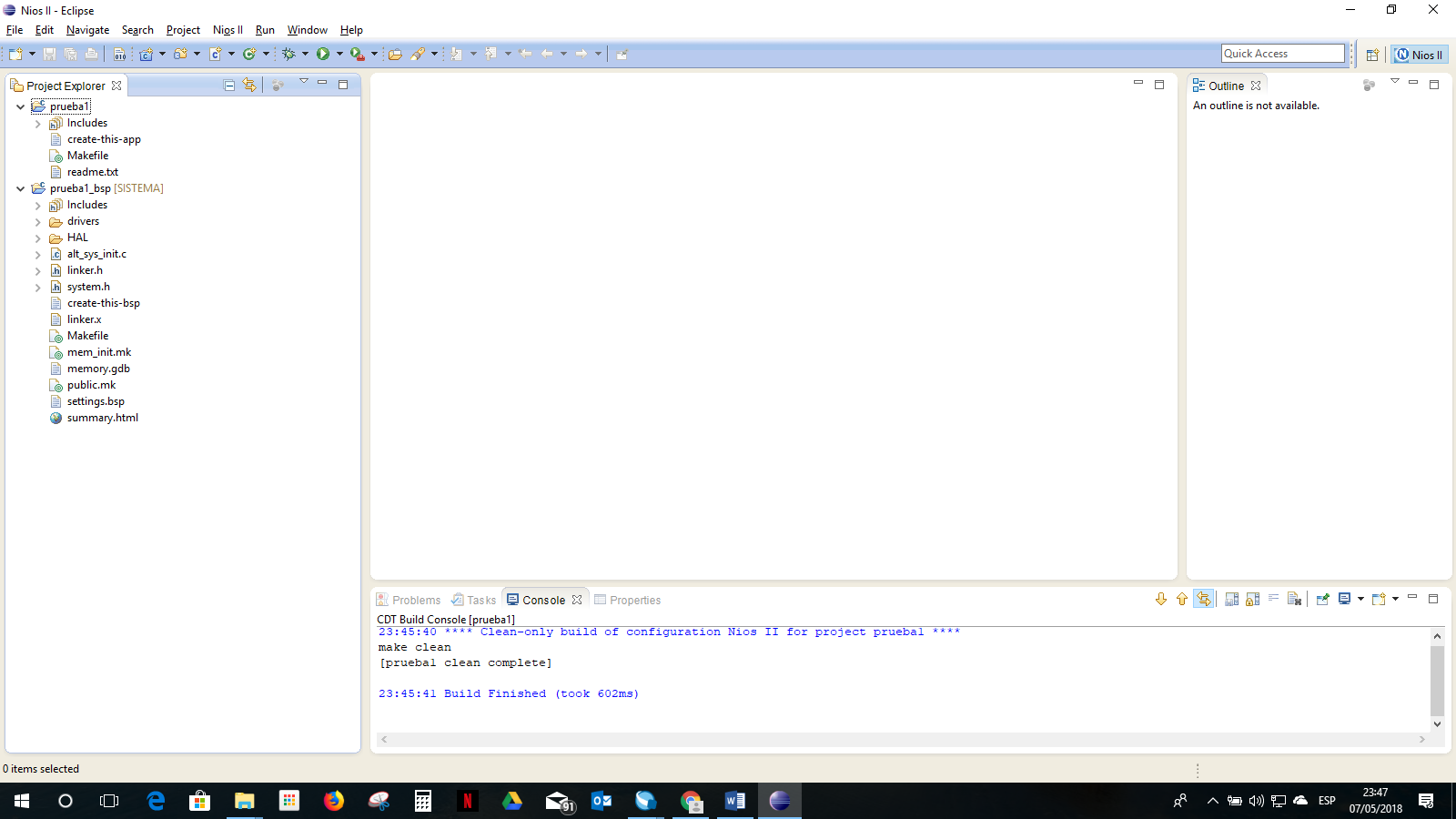
**Figura 16. Selección de una nueva Aplicación para Nios II**

1. En la opción **SOPC Information File name**, proceda a buscar el archivo **Sistema.sopcinfo** que tendrá toda la información de la computadora embebida, como se visualiza en la Figura 17.



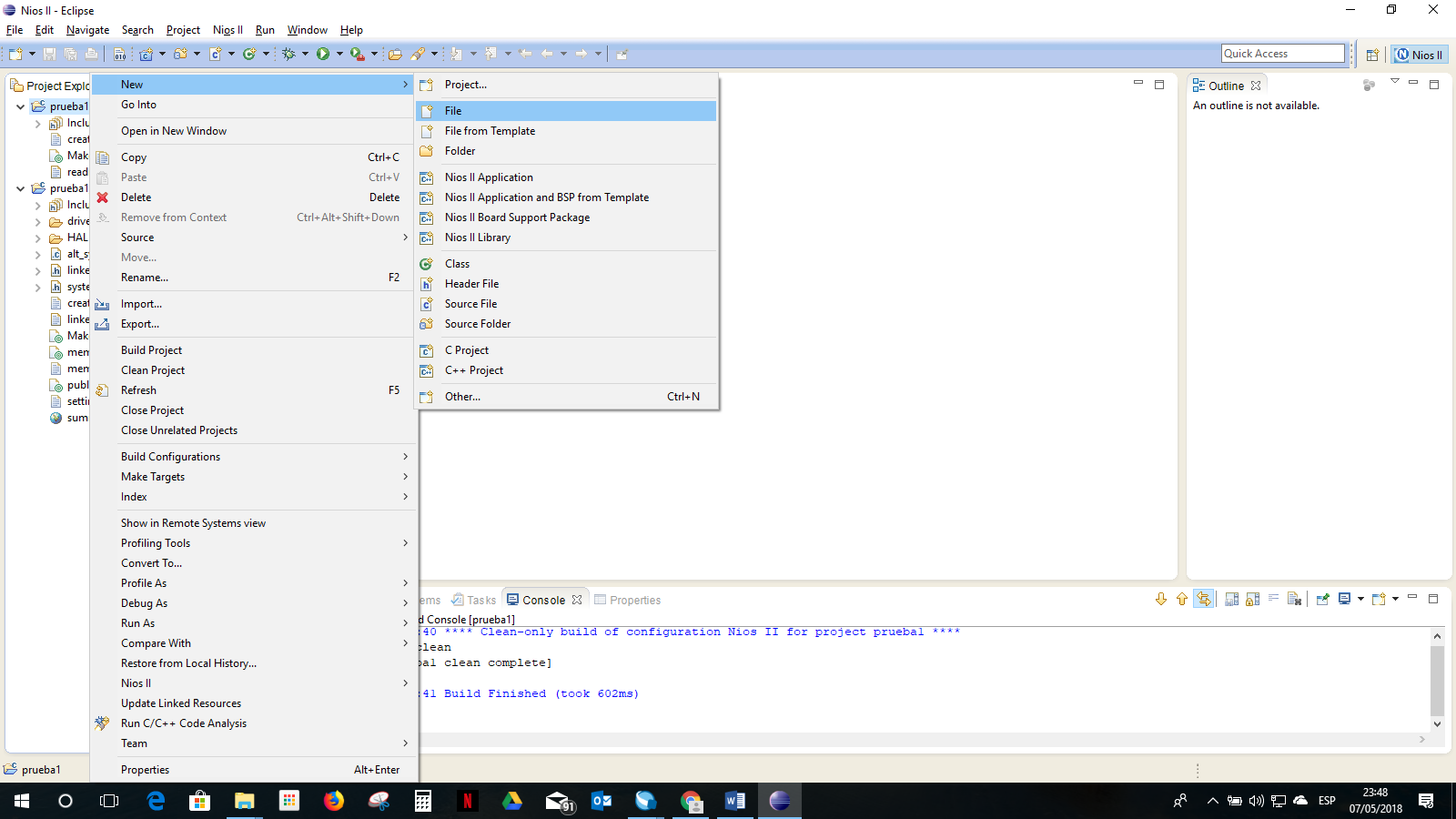
**Figura 17. Ubicación del archivo que contiene la computadora embebida**

1. En la opción **CPU name** se puede utilizar cualquiera de los dos núcleos disponibles en la computadora embebida. Para efectos de la práctica, se utilizará el núcleo **CPU**.
2. En **Project name** se elegirá el nombre del proyecto, utilice el nombre **prueba1** para efectos de la práctica.
3. En **Project Template**, seleccione la opción **Blank Project** para crear nuestro proyecto vacío. Haga clic en **Finish**.
4. En la subventana de Project Explorer podrán visualizar dos carpetas: **prueba1** y **prueba1\_bsp**. En la primera se podrán crear todos los archivos de programación en lenguaje C/C++ para nuestro microprocesador, y la segunda carpeta contiene todos los componentes del hardware, como se visualiza en la Figura 18.



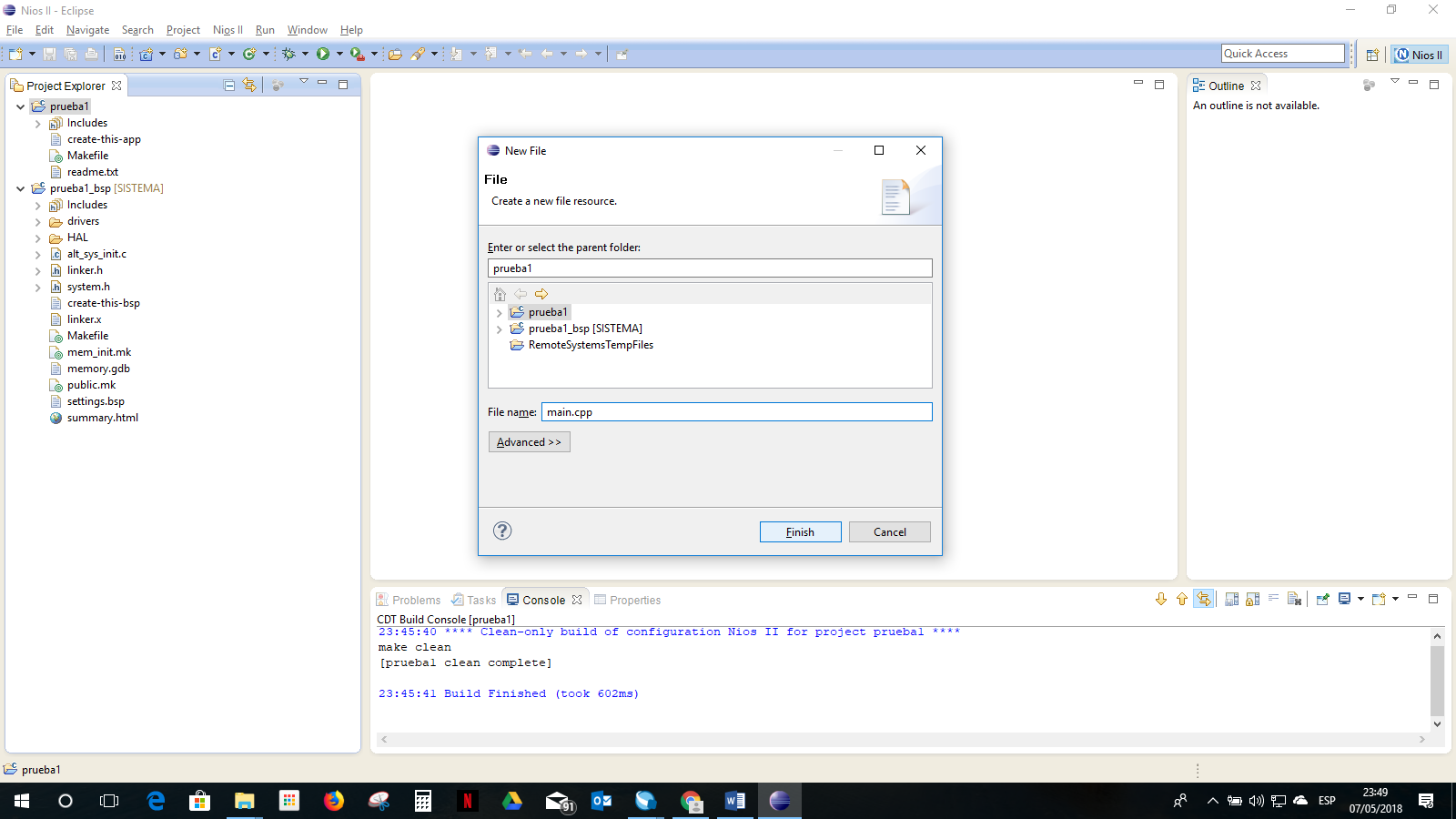
**Figura 18. Ventana del navegador de proyectos**

1. Se procederá a crear un archivo en lenguaje C++, para ello haga clic derecho en la carpeta **prueba1** y seleccione **New → File** como se muestra en la Figura 19.



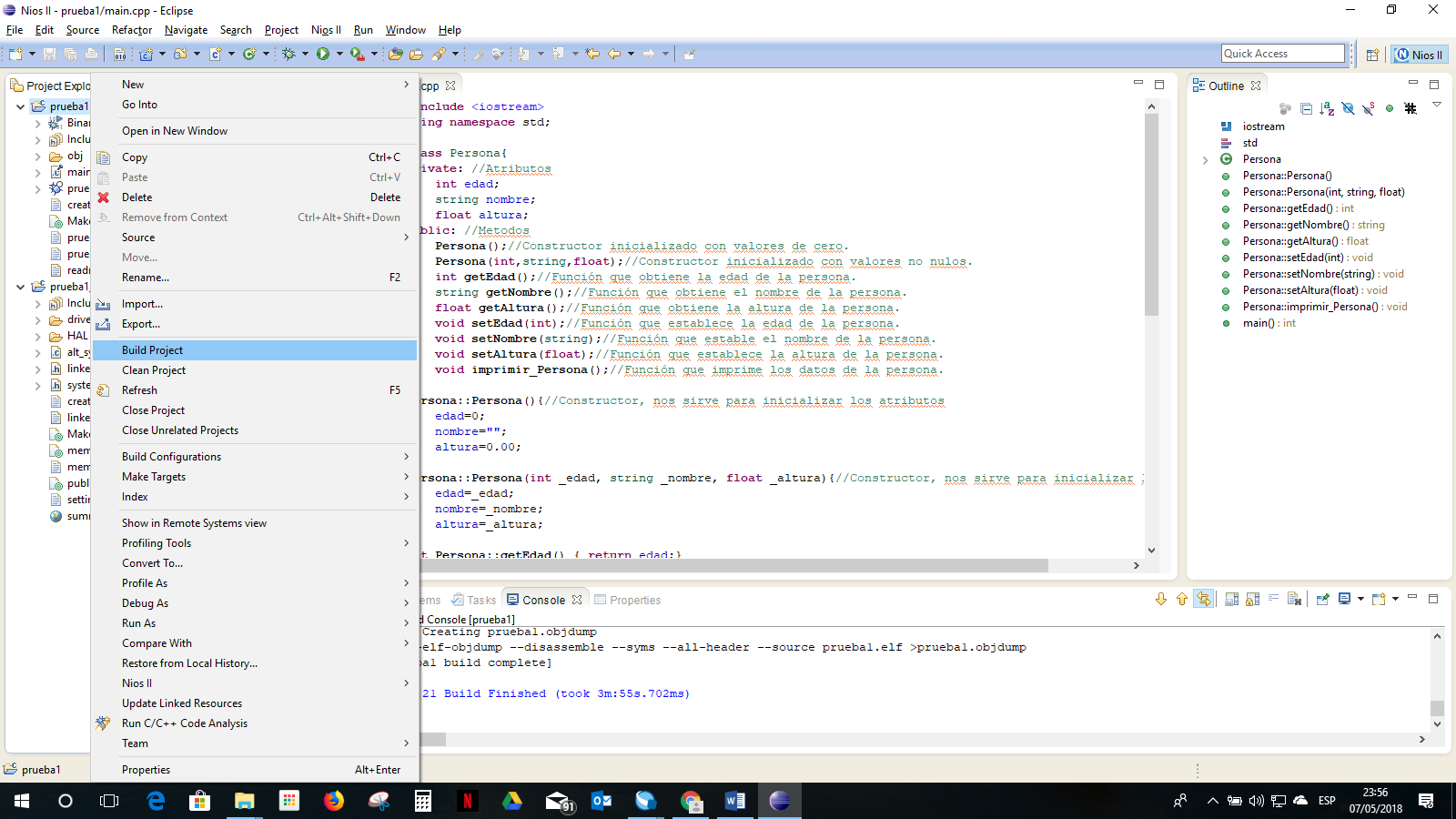
**Figura 19. Creación de un nuevo archivo de programación**

1. En la ventana **New File**, coloque el nombre **main.cpp** y haga clic en **Finish** como se muestra en la Figura 20.



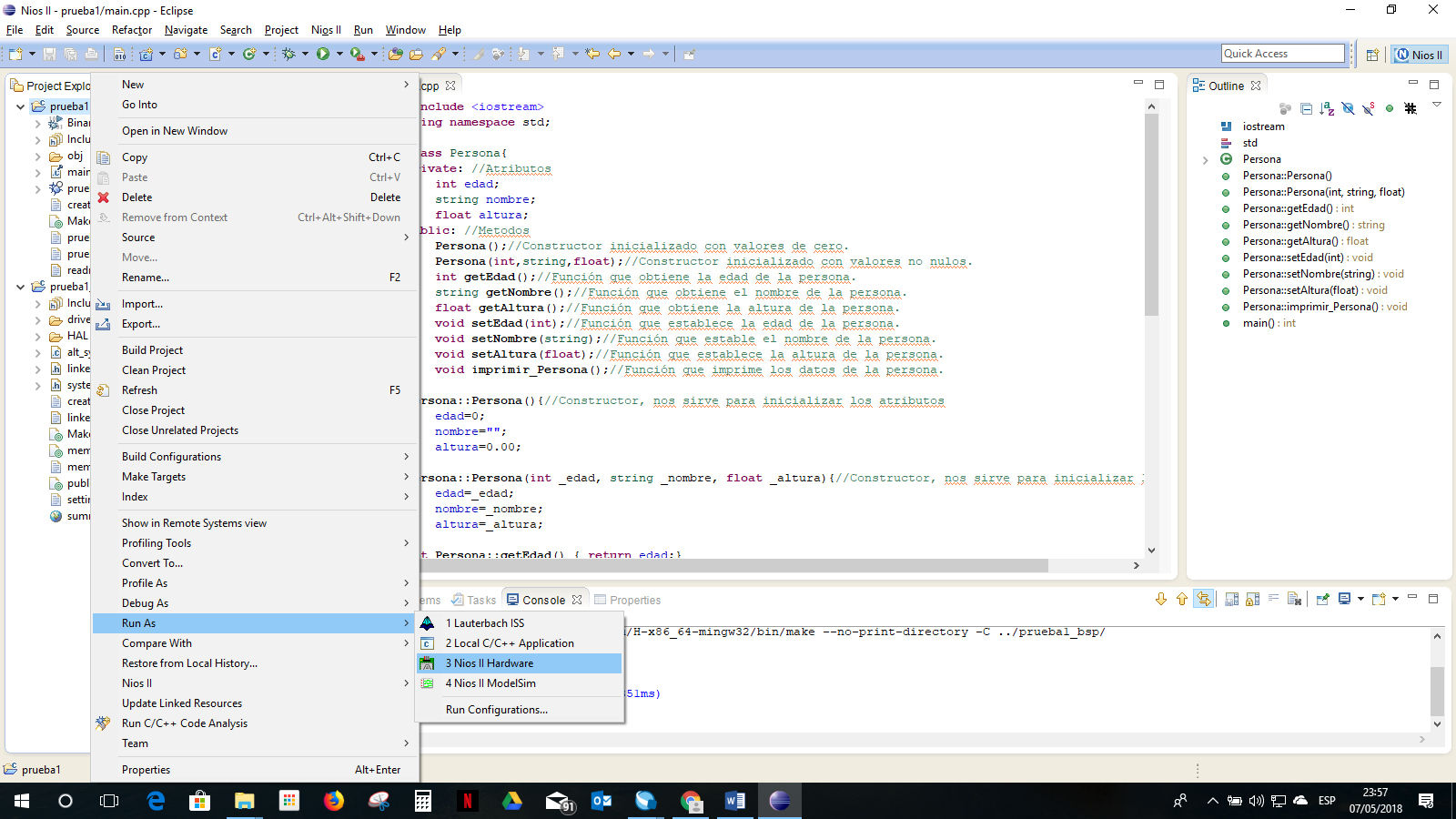
**Figura 20. Ventana de la creación de un archivo en lenguaje C++**

1. Copie la programación del archivo **programa.txt** que se encuentra en la carpeta compartida, en el **main.cpp** de Eclipse.
2. Luego proceda a hacer clic derecho en la carpeta **prueba1** y seleccione la opción **Build Project** como se muestra en la Figura 21.



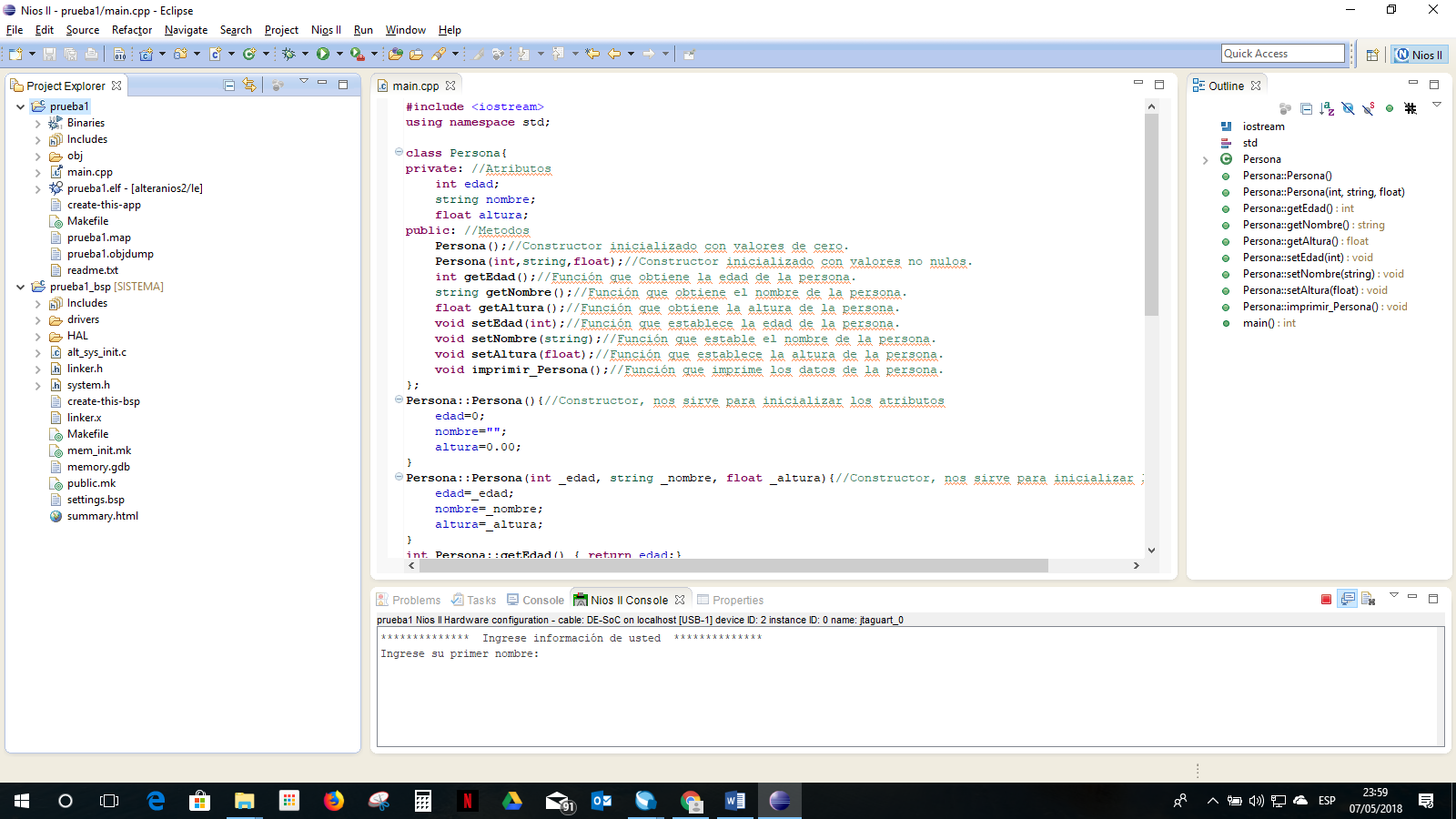
**Figura 21. Compilación del proyecto**

1. Por último, proceda a hacer clic derecho en la carpeta **prueba1** y seleccione la opción **Run As → Nios II Hardware** como se muestra en la Figura 22.



**Figura 22. Programación del microprocesador Nios II en la FPGA**

1. Ingresará por teclado los datos correspondientes para cumplir con el algoritmo deseado, tal como se muestra en la Figura 23.

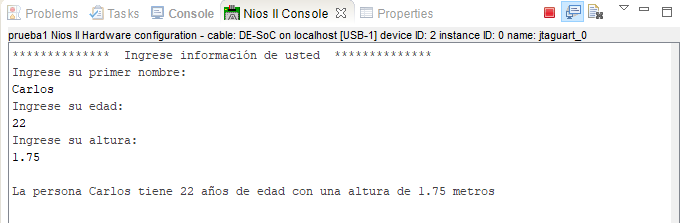


**Figura 23. Salida por consola del programa Eclipse**

1. Conteste la pregunta #1.
2. Conteste la pregunta #2.
3. Conteste la pregunta #3.
4. Conteste la pregunta #4.
5. Describa las conclusiones y recomendaciones de la práctica.

**PREGUNTAS**

1. **Indique lo que realiza el código programa.txt que fue cargado en el microprocesador NIOS II. Adjunte una captura de la salida por consola con sus respectivos datos.**

****

El código comienza importando las librerías a usar, luego define la clase persona.

Dentro de la clase persona define los atributos como privados y los métodos como públicos. Luego establece los constructores de la clase persona y define el main.

El programa establece un loop infinito donde pide al usuario que ingrese datos por teclado sobre su nombre edad y altura, con esos datos crea el objeto persona y muestra en pantalla la salida del método imprimir\_persona().

1. **Indique cuál es la diferencia entre atributos o métodos públicos y privados. Escriba un ejemplo detallando la diferencia.**

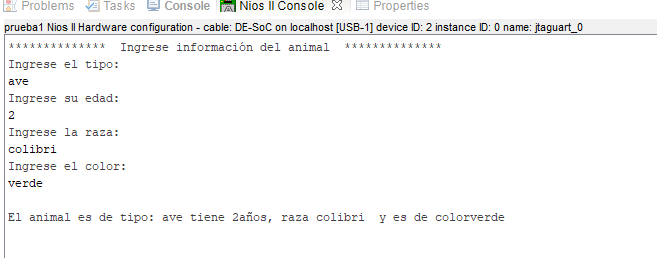
Dentro de los programas se trata de que el usuario final no tenga acceso a todos los datos del sistema, solo los necesarios, es por ello que se definen atributos o métodos privados, los cuales solo pueden ser accedidos dentro de la misma clase. Si se desea que un dato sea accedido por un tercero se crea un parámetro o método público, de esta forma él puede ser llamado fuera de la clase.

Si se desea que los datos sean privados y que alguno pueda ser requerido por un tercero se podría definir un método público, el cual retorne el dato privado. Por ejemplo una clase que represente una cuenta bancaria no retornaría el parámetro de “ saldo” ya que podría ser modificado, en su defecto se crearía un método “consultar\_saldo()” el cual retorna solo el valor numérico del atributo.

1. **Con sus propias palabras indique qué son las Clases en programación orientada a objetos.**

Las clases son la estructura de una representación de un objeto. Un objeto es cualquier elemento que tiene atributos y acciones asociadas a el.

1. **Adjunte un algoritmo en lenguaje C++, en el cual se realice una Clase con atributos privados y métodos públicos. La clase será de tipo animal, el cual tendrá como atributos: (que tipo de animal es, raza del animal, color del animal y edad del animal.). El usuario ingresará por consola los datos de los atributos y al final los visualizará en una frase.**

****

**CONCLUSIONES Y RECOMENDACIONES (**Recuerde que son con respecto a la práctica, mínimo 2 por cada una**)**

Recomendaciones

-Revisar ejercicios de fundamentos de programacion o POO para realizar algoritmos de forma más rapida.

- El document de C++ no se guarda automáticamente, no olvidarse de guarder con cada cambio.

Conclusiones

-La FPGA puede ser usada para realizar procesamiento de datos tal y como un cpu.

- Creando el Hardware apropiado para la FPGA se pueden realizar Proyectos de todo tipo, ya sea sensores, procesamiento de datos, etc.

Codigo Pregunta 4

**#include** <iostream>

**using** **namespace** std;

**class** Animal{

**private**: //Atributos

**int** edad;

string tipo;

string raza;

string color;

**public**: //Metodos

**Animal**();

**Animal**(**int**,string,string,string);

**int** **getEdad**();

string **getTipo**();

string **getRaza**();

string **getColor**();

**void** **setEdad**(**int**);

**void** **setTipo**(string);

**void** **setRaza**(string);

**void** **setColor**(string);

**void** **imprimir\_Animal**();

};

**Animal::Animal**(){//Constructor, nos sirve para inicializar los atributos

edad=0;

tipo="";

raza="";

color="";

}

**Animal::Animal**(**int** \_edad, string \_tipo, string \_raza, string \_color){//Constructor, nos sirve para inicializar los atributos

edad=\_edad;

tipo=\_tipo;

raza=\_raza;

raza=\_color;

}

**int** **Animal::getEdad**() { **return** edad;}

string **Animal::getTipo**() { **return** tipo;}

string **Animal::getRaza**() { **return** raza;}

string **Animal::getColor**() { **return** color;}

**void** **Animal::setEdad**(**int** \_edad) { edad = \_edad;}

**void** **Animal::setTipo**(string \_tipo) { tipo = \_tipo;}

**void** **Animal::setRaza**(string \_raza) { raza = \_raza;}

**void** **Animal::setColor**(string \_color) { color = \_color;}

**void** **Animal::imprimir\_Animal**(){

cout<<"El animal es de tipo: "<<tipo<<" tiene "<<edad<<"años, raza "<<raza<<" y es de color"<<color<<**endl**;

}

**int** **main**()

{

**while**(1){

**int** age=0;string tipo="";string raza="";string color="";

Animal p1= Animal();

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\* Ingrese información del animal \*\*\*\*\*\*\*\*\*\*\*\*\*\*" << **endl**;

cout << "Ingrese el tipo: ";

cin >> tipo;

cout << "Ingrese su edad: ";

cin >> age;

cout << "Ingrese la raza: ";

cin >> raza;

cout << "Ingrese el color: ";

cin >> color;

cout << **endl**;

p1.setEdad(age);

p1.setTipo(tipo);

p1.setRaza(raza);

p1.setColor(color);

p1.imprimir\_Animal();

**for**(**int** i=0;i<8;i++){

cout << **endl**;

}

}

**return** 0;

}

**BIBLIOGRAFIA**

[1]. Universidad Tecnológica Nacional de Buenos Aires. Programación orientada a objetos en C++. Enlace para consulta en línea: http://www.frlp.utn.edu.ar/materias/paradigmas/ApunteCmasmas.pdf. Fecha de consulta: viernes 5 de enero de 2017.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **INICIAL** | **EN DESARROLLO** | **DESARROLLADO** | **EXCELENTE** |

**Firma del Profesor**