UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

TEMA:

"DESARROLLO DE APLICACIONES DE SISTEMAS EMBEBIDOS BASADOS EN FPGA"

Previa la obtención del Título

INGENIERO EN TELECOMUNICACIONES

ELABORADO POR:
Holger Geovanny Ávila Espinoza

Guayaquil, 20 de Septiembre del 2013

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

CERTIFICACIÓN

Certifico que el presente trabajo fue realizado en su totalidad por el Sr. **Holger Geovanny Ávila Espinoza** como requerimiento parcial para la obtención del título de INGENIERO EN TELECOMUNICACIONES.

Guayaquil, 20 de Septiembre del 2013

DIRECTOR

_____

MsC. Edwin Palacios Meléndez

REVISADO POR

_____

Ing. Marcos Montenegro Tamayo.
Revisor Metodológico

_____

MsC. Luis Pinzón Barriga.
Revisor de Contenido

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

INGENIERÍA EN TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

HOLGER GIOVANNY ÁVILA ESPINOZA

"DESARROLLO DE APLICACIONES DE SISTEMAS EMBEBIDOS BASADOS EN FPGA"

DECLARAMOS QUE:

El proyecto de tesis denominado "Desarrollo de Aplicaciones de Sistemas Embebidos basados en FPGA" ha sido desarrollado con base a una investigación exhaustiva, respetando derechos intelectuales de terceros conforme las citas que constan al pie de las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de mi autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance científico del proyecto de grado en mención.

Guayaquil, 20 de Septiembre del 2013

EL AUTOR

HOLGER GEOVANNY ÁVILA ESPINOZA

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

INGENIERÍA EN TELECOMUNICACIONES

AUTORIZACIÓN

Yo, HOLGER GEOVANNY ÁVILA ESPINOZA

Autorizó a la Universidad Católica de Santiago de Guayaquil, la publicación, en la biblioteca de la institución del proyecto titulado: "Desarrollo de Aplicaciones de Sistemas Embebidos basados en FPGA", cuyo contenido, ideas y criterios es de mi exclusiva responsabilidad y autoría.

Guayaquil, 20 de Septiembre del 2013

EL AUTOR

HOLGER GEOVANNY ÁVILA ESPINOZA

DEDICATORIA

Dedico este trabajo principalmente a Dios, por haberme dado la vida y haberme permitido culminar mis estudios universitarios .

A mis padres por ser el pilar fundamental en todo lo que soy, en toda mi educación, tanto académica, como de la vida, por su total apoyo en todos estos años . Por sus consejos, comprensión, amor, ayuda en los momentos difíciles, y por ayudarme con los recursos necesarios para estudiar. Me han dado todo lo que soy como persona, mis valores, mis principios, mi carácter, mi empeño, mi perseverancia ,mi coraje para conseguir mis objetivos.

Todo este trabajo ha sido posible gracias a ellos.

EL AUTOR

HOLGER GEOVANNY ÁVILA ESPINOZA

AGRADECIMIENTO

Primeramente me gustaría agradecerte a ti Dios por bendecirme para llegar hasta donde he llegado, porque hiciste realidad este sueño anhelado.

A mis padres por brindarme los recursos necesarios y apoyarme siempre para culminar mis estudios , por los valores que me han inculcado y por haberme dado la oportunidad de haber tenido una excelente educación en el transcurso de mi vida.

A la Facultad Técnica para el desarrollo de la Universidad Católica de Santiago de Guayaquil por darme la oportunidad de estudiar y ser un profesional.

A mi director de tesis, Ingeniero Fernando Palacios por su esfuerzo y dedicación, quien con sus conocimientos, su experiencia, su paciencia y su motivación ha logrado en mí que pueda terminar mis estudios con éxito.

Me gustaría tambien agradecer a mis profesores durante toda mi carrera profesional porque todos han aportado a mi formación universitaria.

Muchas gracias y que Dios los bendiga

EL AUTOR

HOLGER GEOVANNY ÁVILA ESPINOZA

Índice General

Índice de Figuras

Capítulo 2

Capítulo 3

Resumen

Para el presente documento se dan a conocer en forma general el funcionamiento de los Sistemas Embebidos, en la cual se pueden desarrollar prácticas descritas en el capítulo 3 y también en los anexos 1 y 2. Lo interesante del trabajo de titulación, fue la búsqueda de información debido a que no se ha manejado temas de titulación que involucren los sistemas embebidos, ni del manejo de la tarjeta DE0-Nano de Altera, siendo esta muy completa con respecto a las tarjetas disponibles en el Laboratorio de Electrónica de la Carrera de Ingeniería en Telecomunicaciones de la Facultad de Educación Técnica para el Desarrollo.

En el Capítulo 1, se detallan la justificación, antecedentes y definición del problema de investigación, así como también, el objetivo general, objetivos específicos, idea a defender y la metodología empleada,

En el Capítulo 2, se analiza el describe y explora el Estado del Arte de los Sistemas Embebidos, para que el lector se familiarice con el tema.

En el Capítulo 3, se valida el trabajo de titulación mediante dos prácticas en la que el lector podrá hacer uso del mismo en el Laboratorio de Electrónica, donde va a reposar la tarjeta DE0-Nano.

En el Capítulo 4, se finaliza con las conclusiones y recomendaciones que se dan una vez concluida el trabajo de titulación.

CAPÍTULO 1: GENERALIDADES DEL TRABAJO DE TITULACIÓN

1.1. Introducción.

Los Sistemas Embebidos integran tanto dispositivos electrónicos como plataformas de programación. En el caso de los dispositivos electrónicos, tenemos a los Sistemas Electrónicos Digitales, denominados también como Sistemas Digitales, los mismos que han tenido durante muchos años una gran acogida, pero que en la actualidad cuentan con mayores recursos, de poder integrarse a otras herramientas que antes no se podía lograr por la incompatibilidad con otros equipos.

Es decir, que la evolución de los Sistemas Digitales permitió que se expandan a otros campos, para lo cual lo llamamos actualmente Sistemas Embebidos dando paso a equipos tecnológicos con mayor velocidad y capacidad para el procesamiento de información.

1.2. Antecedentes.

En la actualidad hay diversidad de trabajos de graduación de tercer nivel que han desarrollado trabajos de investigación que involucran temas como NIOS II, Quartus II, tarjeta DE0-Nano, entre otros dispositivos electrónicos. A continuación se exponen algunas tesis que emplean estas herramientas:

a. Tesina de Seminario denominada: ROBOT OMNIDIRECCIONAL CONTROLADO CON NIOS II. Realizada por los estudiantes de la ESPOL: **Liliana Patricia Imbaquingo Quy Yon** y **Richard Eduardo Salavarría Quiroz**. Fecha de realización: año 2013. Disponible en el repositorio de la ESPOL cuya página web es: http://www.cib.espol.edu.ec/digipath/d_tesis_pdf/d-83224.pdf

b. Tesina de Seminario denominada: CONTROL DE LOS MOVIMIENTOS DE UN ROBOT USANDO UN ACELERÓMETRO Y NIOS II. Realizado por los estudiantes de la ESPOL: **Segundo**

Jeancarlos Bastidas Carrillo y **Pablo Geovanny Palacios Játiva**. Fecha de realización: año 2013. Disponible en el repositorio de la ESPOL cuya página web es: http://www.cib.espol.edu.ec/Digipath/D_Tesis_PDF/D-83222.pdf

c. Tesis de Grado denominada: DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA EMBEBIDO DE CONTROL DE ACTITUD PARA AERONAVES NO TRIPULADAS. Realizado por el estudiante: **Alan Kharsansky**. Fecha de realización: 31 de mayo del 2013. Disponible en el repositorio de la Universidad de Buenos Aires (UBA) cuya página web es: http://laboratorios.fi.uba.ar/lse/tesis/LSE-FIUBA-Tesis-Grado-Alan-Kharsansky-2013.pdf

d. Proyecto de graduación denominada: SÍNTESIS DE CIRCUITOS DIGITALES UTILIZANDO VHDL (VHSIC HARDWARE DESCRIPTION LANGUAGE) Y FPGAs (FIELD PROGRAMMABLE GATE ARRAYS). Realizado por la estudiante: **Yohana Elizabeth Alverca Maza**. Fecha de realización: Febrero del 2008. Disponible en el repositorio de la EPN cuya página web es: http://bibdigital.epn.edu.ec/bitstream/15000/764/1/CD-1250.pdf

Solo se han nombrado unas cuántas del último quinquenio, como se mencionó anteriormente, son diversos los trabajos de grado que involucran los sistemas embebidos o empotrados, inclusive en el extranjero, tales como Argentina, Brasil, Colombia y Perú, estos países siempre nos han llevado la delantera en conocimientos que van de acuerdo a los avances tecnológicos.

1.3. Justificación del Problema.

Con la integración de conocimientos de los sistemas embebidos, se ha logrado tener pequeñas aplicaciones didácticas que mejoren el aprendizaje de los alumnos de la FETD en la UCSG. La Carrera de Ingeniería en Telecomunicaciones tiene disponible en su Laboratorio de Electrónica, tarjetas

FPGA tanto de Altera (tarjeta DE1) como Xlinx, pero que no han sido actualizados desde el 2007. La programación VHDL aprendida en Sistemas Digitales II y en Laboratorio de Digitales, permiten que podamos utilizar sin inconveniente la tarjeta DE0-Nano del mismo fabricante Altera.

1.4. Definición del Problema.

Necesidad de desarrollar aplicaciones de Sistemas Embebidos basados en FPGA, lo que permitirá incrementar el nivel de conocimiento y aprendizaje tanto de docentes como estudiantes de la FEDT en la Carrera de Ingeniería en Telecomunicaciones.

1.5. Objetivos del Problema de Investigación.

1.5.1. Objetivo General.

Desarrollar aplicaciones de Sistemas Embebidos basados en FPGA, a través de la integración de la tarjeta DE0-Nano, Quartus II, NIOS II.

1.5.2. Objetivos Específicos.

1. Fundamentar el Estado del Arte de los Sistemas Embebidos.
2. Aplicar la programación en VHDL en Quartus II de Altera y su integración con NIOS II de Eclipse.
3. Desarrollar prácticas que permita evidenciar el correcto funcioanmiento del trabajo de titulación.

1.6. Idea a Defender.

A través del desarrollo de aplicaciones de Sistemas Embebidos basados en FPGA, permitirá que los estudiantes de la Carrera de Ingeniería en Telecomunicaciones profundicen e investiguen temas relacionados con la práctica, así como la del aprendizaje de programación en VHDL en QUARTUS II.

1.7. Metodología de Investigación.

El diseño de investigación, es del tipo exploratorio que trata de examinar un tema o problema de investigación poco abordado, del cual todavía generan dudas. Es decir que solo hay guías no investigadas y vagamente relacionadas con respecto al problema de investigación. También es explicativo, porque se pretende establecer las causas del evento estudiado.

CAPÍTULO 2: Estado del Arte de Sistemas Embebidos.

2.1. Introducción a los Sistemas Embebidos.

Actualmente, ante los retos que presentan la globalización y la fuerte presión de los mercados emergentes, todos los sectores están inmersos en una espiral de esfuerzos que les permitan aumentar sus cuotas de competitividad. La gran aplicabilidad de los Sistemas Embebidos en cualquier ámbito sectorial, así como el valor añadido que aportan los mismos a los productos que los contienen, hace que el desarrollo de estos sistemas sea un área estratégica preferente para muchas empresas que buscan precisamente este aumento de su competitividad.

Así, los Sistemas Embebidos van a jugar un papel vital en nuestra sociedad y se supone revolucionarán los sectores de actividad, como son el sector médico, el de medios de transporte o el de automatización industrial, entre otros.

2.2. Significado de un sistema embebido.

Un sistema embebido es un sistema de procesamiento de información de uso específico integrado en otro sistema de mayor tamaño y conformado por componentes hardware y software (Marwedel, 2003).

Los sistemas embebidos tienen algunas particularidades como la integración de componentes hardware y software (Marwedel, 2003), y su relación de jerarquía con un súper-sistema que se encarga de controlar la comunicación entre sistemas del mismo nivel, tal como se ilustra en la figura 2.1:

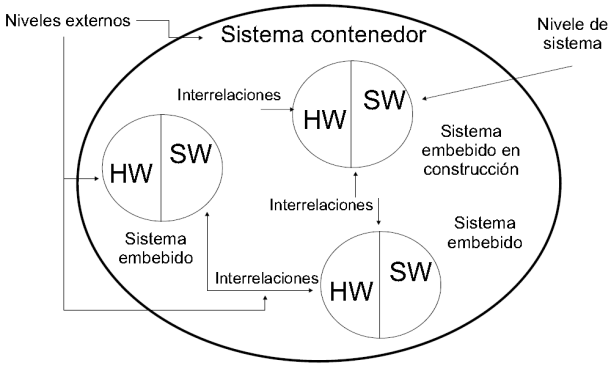


Figura 2. 1: Arquitectura básica de un súper-sistema de varios sistemas embebidos.
Fuente: Liliana González Palacio

2.3. Fundamento de un Sistema Embebido

Un sistema embebido consiste en un sistema de computación cuyo hardware y software están específicamente diseñados y optimizados para resolver un problema concreto eficientemente. El término "embebido" (también se le conoce como “empotrado”) hace referencia al hecho que la electrónica o el sistema electrónico de control es una parte integral del sistema en que se encuentra. La característica principal que diferencia a los “embebidos” de los demás sistemas electrónicos es que, por estar insertados dentro del dispositivo que controlan, están sujetos en mayor medida a cumplir requisitos de tamaño, fiabilidad, consumo y coste, y su existencia puede no ser aparente. Algunos ejemplos de Sistemas Embebidos son los sistemas de información integrados en automóviles, trenes o aviones, y controladores de procesos en sistemas de producción industrial.

2.4. Importancia de los Sistemas Embebidos en las Sociedades Avanzadas.

La importancia que están adquiriendo los Sistemas Embebidos es indiscutible. De cumplirse los pronósticos actuales, el volumen de mercado mundial de estos sistemas será mayor a los 194 billones de euros en el 2013.El campo de los Sistemas Embebidos ha sido considerado de una importancia estratégica para Europa. Estos Sistemas aportan valor añadido a los productos y, cada vez

más, son los responsables de las mejoras introducidas en términos de innovación y competitividad.

Actualmente, Europa es el máximo representante en este campo, donde se espera que este año 2013 el porcentaje de inversión en I+D en Sistemas Embebidos sobre el total de la inversión en I+D sea mayor al 14%. Aun así, esta posición ventajosa puede perderse a favor de los Estados Unidos o de algunos países asiáticos. Los Estados Unidos tienden a utilizar los resultados de los Sistemas Embebidos obtenidos en las aplicaciones militares e industriales y los países asiáticos poseen un amplio mercado nacional y el *knowhow* tecnológico en fabricación que ponen en peligro la posición de liderazgo europea. Europa, por otro lado, posee una elevada cualificación profesional en este ámbito, unos mercados desarrollados y una buena infraestructura.

Una parte de esta infraestructura la conforman las plataformas europeas relacionadas con los Sistemas Embebidos. Las más destacadas son la Plataforma ARTEMIS y la Plataforma ENIAC, cuyos contenidos son los más horizontales dentro de este sector.

De este modo se ha conseguido crear una gran red de contactos que facilita la transferencia de conocimiento y potencia la colaboración a nivel europeo. Su importancia en el marco europeo es tal que llega a influir en la toma de decisiones de las líneas de investigación.

2.5. Tendencias emergentes de los Sistemas Embebidos.

Es precisamente por la importancia que tiene el tema tratado para la industria a nivel mundial y local, que ya existen estudios el cual identifica las tendencias emergentes que probablemente serán de mayor relevancia en el campo de los Sistemas Embebidos en los próximos años.

El objetivo principal de los estudios realizados ha sido el desarrollar una visión de futuro de los Sistemas Embebidos desde un enfoque tecnológico y de sus ámbitos de aplicación, determinando las principales tendencias a corto, medio y largo plazo.

Un estudio realizado en España por un Panel de Expertos recoge de 112 tendencias que pueden considerarse importantes Pese a ello, y con el propósito de realizar un análisis en mayor profundidad de las mismas, se hace necesario seleccionar aquellas tendencias que destacan sobre las demás en cuanto al grado de importancia concedido.

Entre las Tendencias Emergentes más importantes de los Sistemas Embebidos se destacan las siguientes:

2.5.1. Tendencia Tecnológica

Los aspectos referentes a tecnologías son totalmente transversales y tienen impacto en todas las aplicaciones de Sistemas Embebidos.
Las tendencias correspondientes se han dividido en tres áreas:

- Diseños de referencia y arquitecturas
- Conectividad y Middleware
- Métodos, herramientas y procesos para el diseño de sistemas

2.5.1.1. Diseños de referencia y arquitecturas

Las tendencias recogidas en este ámbito tecnológico hacen referencia a aspectos de plataformas hardware, sistemas operativos, software e interfaces. Asimismo también hacen referencia a las características que deberán tener estos sistemas para satisfacer las necesidades que se plantean hoy en día con los Sistemas Embebidos. Así podemos ver cómo aspectos como la seguridad y confiabilidad de los sistemas o la capacidad para trabajar en tiempo real y en estados degradados, destacan como requerimientos que se cumplirán en un futuro más o menos próximo.

El hardware de un Sistema Embebido consiste en uno o más procesadores programables que permiten hacer funcionar la parte software de la aplicación, un subsistema de memoria y bloques específicos para el funcionamiento dependiendo de la aplicación, incluyendo todas las interfaces de entrada y salida así como sensores y actuadores que establecen la conectividad con el dispositivo en que se encuentran embebidos o con el entorno.

Una tendencia muy importante en cuanto a hardware es que éste tenderá a ser abierto (con diseño e interfaces abiertos y modificables por el usuario), de forma que se pueda utilizar para todo tipo de aplicaciones (arquitecturas interoperables). Asimismo, también se generalizará el uso de sistemas operativos y software abierto para programas y aplicaciones que deban funcionar en tiempo real, permitiendo darle un correcto funcionamiento de acuerdo con los conceptos de determinismo, sensibilidad, controlabilidad, fiabilidad y tolerancia a fallos. En la figura 2.2 se muestra la arquitectura de un sistema embebido.

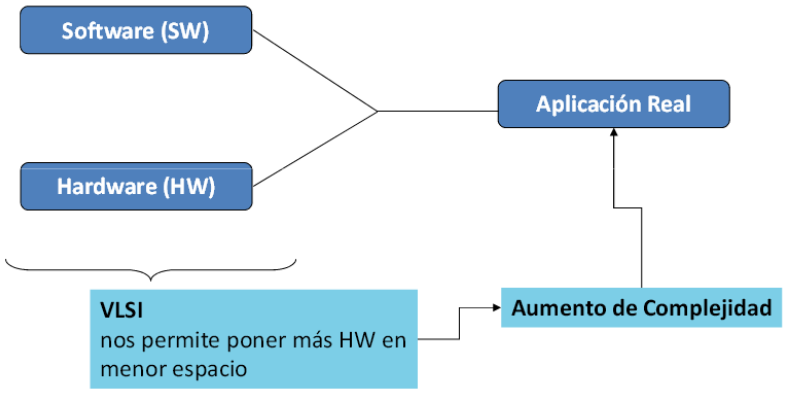


Figura 2. 2:Arquitectura de un Sistema Embebido

Fuente: Guillermo Carpintero del Barrio

Como consecuencia de la mejora de la tolerancia a fallos (confiabilidad) se podrá trabajar en condiciones degradadas, alargando la vida útil de los sistemas con inteligencia embebida. El hecho que las arquitecturas puedan soportar cualquier modo de trabajo en estado degradado puede convertirse en

la clave para el éxito comercial de muchos productos, ya que se asegura el funcionamiento de los mismos pese a las malas condiciones.

Otro tema destacable de este apartado es el que hace referencia a la evolución de los chips ligada a su uso en Sistemas Embebidos. Se prevé que entre el 2015-2020 los chips tendrán integrados métodos de autoconfiguración y autodiagnóstico que les permitirá adaptarse de forma óptima a las tareas en cada situación y trabajar en estado degradado.

Asimismo, el continuo progreso en la tecnología microelectrónica está permitiendo que la tendencia actual sea la de integrar todos los componentes de un computador en un solo circuito. Es el llamado “Systemon Chip (SoC)”, que se prevé que se implante en todos los ámbitos en el período 2015-2020. Paralelamente, se conseguirá un despliegue creciente del “Network on Chip”, que podemos definir como un “Systemon Chip” con capacidades de comunicación integradas. Los Sistemas Embebidos con soluciones SoC implementadas presentan una serie de ventajas que podrían resumirse en: una elevada fiabilidad, tamaño reducido, mayor rendimiento, menor consumo energético y menor coste, en la figura 2.3 se muestra un ejemplo de un sistema embebido.

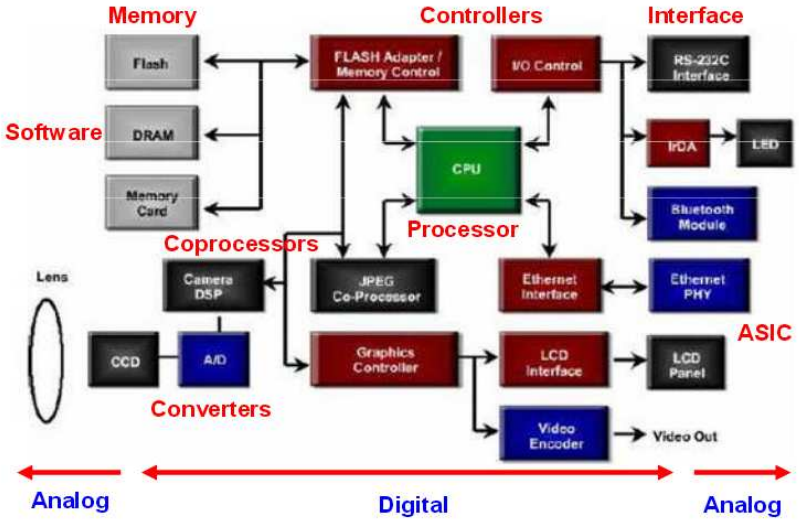


Figura 2. 3:Ejemplo de un Sistema embebido

Fuente: Guillermo Carpintero del Barrio

En cuanto a las interfaces de comunicación entre estos sistemas y su entorno, cabe destacar que se desarrollarán interfaces Humano-Máquina adaptadas a cualquier uso para interactuar con equipos que contengan Sistemas Embebidos. Asimismo, se prevé que en el período 2021-2025 las interfaces de los Sistemas Embebidos tengan elementos de traducción automática y de síntesis de voz totalmente fiables y fáciles de integrar.

2.5.1.2. Conectividad y Middleware

El término middleware se refiere a una capa de software entre los servicios de la red y las aplicaciones, encargada de proporcionar servicios como identificación, autenticación, autorización, directorios y movilidad. El uso de esta capa permite a las aplicaciones sacar un mayor provecho de la Red e interoperar por medio de interfaces normalizadas, ofreciendo así a los usuarios servicios más avanzados con un menor esfuerzo.

El gran número de tecnologías en despliegue para proporcionar una conectividad global refuerza la importancia del middleware. En este campo, las redes de comunicación tendrán un papel muy destacado a todos los niveles.

A fin de proveer cobertura a toda la extensión territorial, las redes ad-hoc (MESH) se integrarán automáticamente con redes de área local, metropolitana y de banda ancha y serán una solución común y generalizada que permitirá la comunicación en zonas donde no existe una infraestructura fija instalada. Las redes ad-hoc son aquéllas que se crean de forma espontánea, sin una infraestructura específica y funcionando en un espacio y tiempo limitados.

En relación a las infraestructuras, se integrarán automáticamente redes inalámbricas de corto alcance y los servicios que ofrecen con Internet. Para la difusión de estos servicios se utilizarán web services en tiempo real de forma generalizada. Se considera un web service un conjunto de protocolos y estándares que sirven para intercambiar datos entre distintas aplicaciones de

software desarrolladas en lenguajes de programación diferentes y ejecutadas sobre cualquier plataforma.

Gracias a la infraestructura dispuesta, los dispositivos elegirán en cada momento la tecnología de comunicación más adecuada para transmitir con máxima seguridad la información en función del tipo y la cantidad de información, la distancia, los interlocutores y otros factores que puedan condicionar el éxito, la rapidez u otra medida de calidad de la transmisión.

Esta tendencia está relacionada con el concepto ABC (AlwaysBestConnected), consistente en la posibilidad de estar comunicado en cualquier lugar, en cualquier momento y siempre a través de la tecnología más apropiada, es decir, emplear el modo de conexión más eficiente entre todos los disponibles en cada momento. Algunos dispositivos ya incluyen sistemas similares, como por ejemplo el IPhone, que dispone de una jerarquía de comunicación por la cual puede pasar de sistema Wi-fi (primero en la jerarquía) al sistema 3G automáticamente en caso de fallo del primero.

La fecha de materialización estimada de esta tendencia está fijada entre el 2009-2015. Aun así, dependiendo del grado y del sector de aplicación, el horizonte temporal se podrá alargar hasta más allá de 2026.

2.5.1.3. Métodos, herramientas y procesos para el diseño de sistemas

Los métodos, herramientas y procesos para el diseño de sistemas engloban, como su nombre indica, todas las etapas referentes al desarrollo de sistemas. Estas etapas experimentarán una mejora considerable gracias a la evolución y aportación de los Sistemas Embebidos.

La tendencia inicial con fecha de materialización prevista más cercana (2009-2018) está relacionada con las técnicas de modelización y simulación. La aplicación de estas técnicas para los Sistemas Embebidos permitirá una

reducción del time tomarket y gestionar la complejidad creciente de los sistemas y, de este modo, reducir costes de desarrollo.

La gran reducción del time tomarket será debida a la disminución del tiempo utilizado para la integración y las pruebas, típicamente las etapas que más tiempo consumen en el proceso de desarrollo.

Para evitar errores durante la fabricación de los sistemas debidos a fallos en el diseño, se generalizará el uso de métodos formales de verificación que permitan validar el sistema en fase de diseño, así como modelos específicos del dominio. Estas herramientas de verificación unidas a las herramientas del diseño basado en modelos supondrán una mejora importante en la optimización del tiempo de trabajo y de los costes de desarrollo.

Otra de las tendencias a destacar y que se está extendiendo en todos los sectores de aplicación es la certificación de calidad de los sistemas. Si bien es cierto que en la actualidad muchos de los sectores no requieren de una certificación para poder realizar sus servicios, se observa un incremento en las demandas de este tipo de acreditaciones que demuestren la confiabilidad de los sistemas. Por este motivo, se generalizará la demanda y oferta de certificación de calidad del software empotrado, así como del proceso de desarrollo y de las capacidades de los desarrolladores.

Debido a la existencia de Sistemas Embebidos críticos, que requieren de certificación para ser considerados como tal y así acreditar una alta confiabilidad para poder realizar ciertas tareas, existirán modelos, metodologías y herramientas que faciliten el proceso de certificación para que se realice en modo eficaz y eficiente.

La utilización de protocolos a todos los niveles es una tendencia generalizada. En concreto, en el ámbito del modelado en tiempo real, se desarrollarán nuevos estándares parecidos a los estándares ya existentes como SysML y

UML-MARTE, y que destacarán por ser soportados por herramientas industriales abiertas.

La tendencia general a aunar todas las funcionalidades en un mismo conjunto propiciará la aparición de herramientas de diseño que cubran, de forma integrada, todas las etapas del ciclo de vida del desarrollo del software, hardware y del sistema. Esta tendencia está englobada en el concepto IDE (Entorno Integrado de Desarrollo), consistente en un entorno de programación empaquetado como un programa de aplicación donde, partiendo de unas funcionalidades estándar, existe la posibilidad de añadir plugins que aporten funcionalidades extra.

Además, las herramientas, métodos y procesos deberán prepararse ante el uso de nuevas arquitecturas aplicadas a Sistemas Embebidos, como por ejemplo el MPSoC (multiprocessorsystemon chip) o las arquitecturasMany-Core.

Estas nuevas arquitecturas acabarán imponiéndose a las arquitecturas actuales y requerirán el desarrollo de nuevos métodos y herramientas de diseño, síntesis, compilación, debugging y despliegue de servicios colaborativos, es decir, una revisión y rediseño de los métodos y herramientas aplicados con las anteriores arquitecturas, y su adaptación a las nuevas.

2.5.2. Tendencias en diversas Áreas de Aplicación

El impacto de los Sistemas Embebidos es elevado y afecta a todos los sectores sin excepción. En los siguientes apartados se identifican las principales tendencias en diversas áreas de aplicación definidas por el Panel de Expertos.

2.5.2.1. Medios de TransporteAeroespacial

El aeroespacial es un sector que está avanzando mucho tecnológicamente, siendo el sector industrial que alcanza una mayor cuota de gasto en I+D respecto a la facturación.

Además, este sector ha estado adquiriendo una importancia notable en los últimos tiempos, debido, entre otros motivos, a la creciente necesidad de movilidad en nuestra sociedad. De hecho, las previsiones para los próximos 20 años apuntan hacia un crecimiento sostenido del sector en nuestro país.

Sin duda, los Sistemas Embebidos jugarán un papel muy importante en la evolución de esta industria, que se materializará, según los expertos, en soluciones tecnológicas como las que se analizan a continuación y que pretenden dar solución a problemáticas y situaciones con las que el sector ha de enfrentarse.

Una de estas problemáticas es precisamente el aumento del volumen de pasajeros, que está llevando a un aumento del número de aeronaves, haciendo necesaria la mejora del control del tráfico aéreo. En este caso concreto, los Sistemas Embebidos podrán ayudar mediante la implantación de sistemas globales de tráfico aéreo interactivos que permitan este incremento de la densidad del tráfico, así como su seguridad.

El aumento del volumen de usuarios de este medio de transporte también estará ligado a la capacidad de ofrecer un valor añadido durante el vuelo. Este valor añadido puede materializarse en forma de servicios que aporten una sensación de confort y bienestar, sobre todo en los vuelos de larga duración.

Así, aparecerán Sistemas Embebidos en los aviones de transporte de viajeros que permitirán ofrecer nuevos servicios a los viajeros (conexión a internet, telefonía, etc.).

A nivel general se prevé que la evolución tecnológica que sufrirán los Sistemas Embebidos permita conseguir niveles de confiabilidad muy elevados a costes muy competitivos, haciendo que la implantación de dichos sistemas sea rentable para el sector.

Los Sistemas Embebidos también tendrán influencia en aspectos relacionados con la seguridad. Con el objetivo de fortalecer la confianza y la sensación de seguridad, se generalizará la utilización de hardware y software abierto de uso comercial para Sistemas Embebidos que ejecutarán funciones con implicaciones en la seguridad de vuelo y, por tanto, certificables.

2.5.2.2. Sector Ferroviario

El sector ferroviario español está muy ligado a las tendencias europeas, y en ese sentido, la creación e implantación del sistema ERTMS (Sistema de Gestión de Tráfico Ferroviario Europeo), también afectará a nuestro sistema ferroviario.

El ERTMS consiste en un sistema de control y otro de señalización diseñados según la normativa europea con el objetivo de aumentar la interoperabilidad del transporte ferroviario en Europa. La implantación de este sistema permitirá controlar la seguridad en la conducción de forma dinámica, así como compartir infraestructuras entre los diversos países con independencia de los operadores ferroviarios.

El incremento de la seguridad debido al uso de los Sistemas Embebidos es una tendencia que se repite en todos los sectores de transporte, y no es una excepción en el sector ferroviario. En este sentido se realizarán varias acciones para su mejora. Por ejemplo, se implantarán sistemas de conducción automática en el transporte público ferroviario (tren y metro), gracias a que los sistemas expertos serán cada vez más inteligentes y fiables. Esto permitirá aumentar la seguridad, evitando los fallos humanos; por tanto, la confiabilidad se convierte en una pieza clave para el desarrollo de los sistemas embebidos para esta área de aplicación.

Los sistemas de control ferroviario también serán muy seguros, incluyendo comunicaciones tren-tierra, control de velocidad, distancia entre vehículos y

gestión de flotas. La comunicación entre vehículos, es una tendencia a nivel general para el sector de transportes.

2.5.2.3. Sector Automoción

La industria del automóvil es uno de los sectores más importantes de la economía de nuestro país, a la que contribuye notablemente en términos de producción, empleo y desarrollo tecnológico.

En este sector, los recientes avances en la tecnología de sistemas, así como la continua demanda de mejoras en la conducción y en la seguridad activa, han obligado a los fabricantes y suministradores a perseguir el desarrollo de subsistemas electromecánicos controlados por ordenador. En este sentido, los sistemas X-by-Wire reemplazarán los tradicionales enlaces mecánicos e hidráulicos, entre los cuales se encuentran los controles del conductor, los mecanismos de dirección o los frenos con elementos electromecánicos.

Así, los componentes tradicionales de los sistemas de frenado y dirección, tales como la columna de dirección, eje intermedio, bomba, manguitos, fluidos, cinturones y los cilindros de potencia de frenada, serán eliminados por completo.

La aplicación de las tecnologías de la información y comunicaciones al ámbito del transporte por carretera permite el desarrollo de una nueva generación de sistemas autónomos que en un futuro permitirán la automatización de determinadas maniobras en automóviles, incrementando tanto el confort del conductor como la seguridad en la carretera. En este sentido, la conducción autónoma se convertirá en una realidad, gracias a que los Sistemas Embebidos serán cada vez más inteligentes y fiables, convirtiéndose la confiabilidad en una pieza clave para el desarrollo de los sistemas embebidos para esta área de aplicación.

Esto permitirá reducir el número de accidentes.Aunque la conducción autónoma de vehículos se considera el último escalón en el desarrollo de los sistemas avanzados de asistencia a la conducción (ADAS), la investigación en este campo contribuye, sin ninguna duda, al desarrollo de toda una gama de herramientas y aplicaciones que servirán de base científica al desarrollo de los futuros productos comerciales en el ámbito de la automoción.

Otra aplicación que mejorará las prestaciones ofrecidas a los usuarios estará encaminada a implementar servicios multimedia que ayuden a optimizar la eficiencia vial (información del tráfico, etc.). Para la implementación de estos servicios se requerirá de tecnologías java y OSGi, que serán usadas en entornos de automoción. Actualmente ya existe algún modelo de vehículo que incorpora servicios multimedia, aunque no se prevé su uso generalizado hasta más allá del año 2012. La incorporación de estos servicios multimedia supondrá un paso más en la mejora de la interacción coche-usuario, que sigue siendo una tendencia clara en este sector.

Las arquitecturas abiertas también tendrán una gran importancia en este sector. Concretamente, se prevé que la arquitectura AUTOSAR tenga un papel muy relevante en el mismo. AUTOSAR (Arquitectura de Sistemas Abierta para el sector del Automóvil) es una arquitectura de software abierta y estandarizada desarrollada conjuntamente por fabricantes de automóviles, proveedores y desarrolladores de herramientas de software. El objetivo de esta colaboración es crear y establecer estándares abiertos para arquitecturas de componentes electrónicos en el sector automotriz, que provean una infraestructura básica a partir de módulos, interfaces al usuario, y control para los diferentes dominios. Esto incluye la estandarización de funciones de sistema básicas, la portabilidad a diferentes variantes de vehículos y plataformas, la portabilidad a lo largo de la red, la integración de múltiples proveedores y el mantenimiento y nuevas versiones del software a lo largo del ciclo de vida del vehículo. Por este motivo, AUTOSAR se impondrá como estándar, de momento en procesadores de 32

bits. Asimismo, se creará una red de proveedores de módulos AUTOSAR, fabricantes de herramientas AUTOSAR, etc. que hoy es incipiente.

La última tendencia a analizar está relacionada con el futuro uso masivo de los vehículos híbridos. En este marco, la electrónica de potencia entrará de forma definitiva, y con ésta la electrónica de control y software asociados, con algoritmos cada vez más sofisticados que se focalizarán en minimizar el consumo y las emisiones.

2.5.2.4. Salud

Las tendencias recogidas en este sector de aplicación hacen referencia a aspectos de dispositivos clínicos, procesos y productos relacionados con el ámbito médico y sanitario.

El desarrollo de las tecnologías de la información ha hecho posible que cualquier ciencia o disciplina se beneficie de ellas. En concreto, los Sistemas Embebidos han sido considerados como “facilitadores” de la tecnología médica, siendo un componente crucial en el desarrollo del sector médico-sanitario, especialmente en aplicaciones de diagnóstico y terapia.

En este campo de aplicación, los Sistemas Embebidos pueden estar incorporados en dispositivos electrónicos que formen parte de equipos médicos con el objetivo de mejorar las características de los mismos y optimizar los procesos. En esta línea, aparecerán sistemas multisensoriales que permitirán mejorar el diagnóstico, el tratamiento y el post-seguimiento de las enfermedades a partir de dispositivos que incorporen la detección y el tratamiento en los mismos.

Asimismo, la transversalidad de los Sistemas Embebidos generará muchas posibilidades que mejorarán los servicios sanitarios. Por ejemplo, los sensores de señales vitales unidos a la localización de personas permitirán una atención mucho más rápida y efectiva en situaciones de emergencia.

Otro aspecto a tener en cuenta, y que ha quedado reflejado en las tendencias anteriormente listadas, es la transmisión de datos. A fin de evitar filtraciones de datosy asegurar su privacidad, las redes inalámbricas de trasmisión de datos sanitarios serán seguras, con garantía de funcionamiento y tendrán un acceso restringido.

La consecución de este hito permitirá otras aplicaciones que necesiten de este tipo de trasmisiones, como son los sistemas de autodiagnóstico para ser usados en cualquier entorno sin la necesidad de asistencia por parte de personal clínico especializado. Otra de las tendencias importantes en este campo hace referencia a los dispositivos implantables, en concreto las prótesis humanas. En este caso, se prevé que éstas estén provistas de inteligencia, de forma que mejorarán su funcionalidad, y por tanto la calidad de vida de los pacientes. Debido a la gran complejidad que conlleva el desarrollo de este tipo de dispositivos, no se prevé que éstos estén disponibles hasta después del año 2026.

Cabe destacar la reticencia general hacia los temas con fuerte implicación de la privacidad de los pacientes.Dichos temas, han quedado fuera de este análisis, ya que su IGI ha sido menor que el de la media del sector.

2.5.2.5. Automatización industrial

Las tendencias recogidas en esta apartado hacen referencia al uso de los Sistemas Embebidos en aplicaciones de automatización industrial, entendiendo la automatización industrial como el conjunto de técnicas que involucran la aplicación e integración de diferentes sistemas para operar y controlar procesos productivos de forma autónoma.

En este sector, los Sistemas Embebidos están cobrando una gran relevancia y se espera que su aplicación y evolución conlleve cambios positivos que afectarán a distintos ámbitos de la automatización.

Uno de los beneficios más claros que se obtendrán gracias a su uso será la optimización de la gestión y el control de productos y procesos.La gestión de almacenes y de logística de la empresa se hará mediante Sistemas Embebidos en los productos y en los sistemas de transporte internos para permitir hacer un seguimiento de las existencias y hacer la gestión de pedidos de forma automática.

Los Sistemas Embebidos juegan en este caso un papel decisivo en la consecución de la trazabilidad de los productos, entendiendo por trazabilidad el conjunto de acciones, medidas y procedimientos técnicos que permiten identificar y registrar cada producto desde su fabricación hasta el final de la cadena de comercialización, y en muchos casos hasta el final de su ciclo de vida. Estos sistemas, además de optimizar la logística de las empresas fabricantes, garantizarán la calidad de los productos, aportando una información completa y veraz sobre su historial. Esta información será de gran interés, tanto para el fabricante como para el consumidor.

A nivel de sistemas de fabricación, éstos incorporarán Sistemas Embebidos dotados de sensores que permitirán conocer su comportamiento, realizar autodiagnósticos y almacenar su historial de fabricación y mantenimiento, facilitando así la construcción de grandes sistemas de fabricación y su control. De este modo, se conseguirá dar un valor añadido sobre el mantenimiento y fabricación de grandes equipos, instalados en grandes plantas, donde es necesario realizar un mantenimiento en tiempo real y donde es muy importante saber cuál es el comportamiento de estos equipos para una mayor optimización del tiempo en operación y de los resultados. Además, la implementación de Sistemas Embebidos permitirá avanzar hacia la flexibilidad y auto configurabilidad de los sistemas de fabricación, permitiendo que éstos se adapten a configuraciones de fabricación distintas.

Otra de las tendencias destacables es que los sistemas de control industrial podrán ser controlados en tiempo real. En este sentido, se utilizarán servicios

Web en tiempo real, que incluso permitirán realizar estas acciones a kilómetros de distancia.

2.6. Identificación de las características propias de los sistemas embebidos.

La característica más importante de los sistemas embebidos es su interacción con el mundo exterior en función del tiempo o en función de la presencia de estímulos. Para garantizar una interacción exitosa con el ambiente, el sistema debe incorporar algunas características, tales como la disponibilidad, fiabilidad y seguridad. Otras características propias de un sistema embebido son (Marwedel, 2003) (Lavi y Kudish 2005):

- Compuesto por hardware y software, con la característica de que el software tiene una interacción directa con los elementos hardware, pues se encarga de controlarlos y comunicarlos. Para esta composición debe ser posible representar: comportamiento (estados, eventos, y señales) y estructura o composición física.

- Relaciones jerárquicas, en las cuales se incluyen las interrelaciones entre el sistema embebido y su súper-sistema, el sistema embebido y sistemas del mismo nivel que se encargan de otras funciones específicas.

- Comportamiento basado en el estado de las componentes, por ejemplo, si X puerto no está disponible, entonces no se podrá enviar la señal que activa el proceso Y.

- Manejo de eventos, que son los que permiten constatar el cambio de estado de las componentes. Un evento puede ser externo (causado por el ambiente) o interno (causado por componentes del sistema).

- Recursos limitados en cuanto al tamaño, el consumo de energía, la memoria, y demás recursos que permitan garantizar la portabilidad del sistema embebido.

- Mínima interacción con el usuario, por lo tanto, son sistemas que deben funcionar durante años sin errores y ser capaces de recuperarse por sí mismos en caso de que estos ocurran. Deben ser sistemas con un alto grado de autonomía.

- Presencia de sincronización y comunicación, para permitir el flujo de información entre los diferentes sistemas embebidos que hacen funciones específicas y contribuyen a la realización de la función del súper-sistema.

- Propiedades no funcionales, tales como la tolerancia a fallas, el tamaño, el consumo de potencia, el peso, la disponibilidad, la seguridad, la fiabilidad, deben definirse desde etapas tempranas de la construcción del sistema.

- La controlabilidad es otra característica de los sistemas embebidos, pues son sistemas pensados, en su mayoría, para el control, y además, por sus restringidas y muy específicas funciones, también son fácilmente controlables.

2.7. La Evolución de los Componentes Electrónicos de Sistemas Digitales.

Desde la aparición del circuito integrado, existen dos alternativas para realizar un hardware digital: codificación del algoritmo en un microprocesador o mapeo directo del algoritmo en hardware. Los microprocesadores junto con los microcontroladores y DSPs permiten resolver eficazmente la mayor parte de los problemas electrónicos. Tal como ocurre en la naturaleza, donde los insectos sencillos son mayoría, en el mundo digital existen infinidad de problemas que

se resuelven con un procesador sencillo. En efecto, aún hoy los micros de 8 bits son los más vendidos, seguidos de dispositivos de 4 bits. Sin embargo, existe una serie de problemas donde los micros no son suficientes. Tal es el caso cuando la E/S de datos combina gran cantidad y gran velocidad, o cuando el número de operaciones por muestra es elevado, la tabla 2.1 se muestra la evolución de los dispositivos electrónicos. En tal caso, la única opción es la realización de un ASICs (Circuito Integrado de Aplicación Específica), una tecnología que aparece en la década de los 80.

Tabla 2. 1:Evolución de los componentes electrónicos para la materialización de SistemasDigitales.

Componente	Época aproximada	Orden de Velocidad (Multiplicación)
Engranajes	1400-1930	seg.
Relés	1930-1940	mseg
Válvulas	1940-1960	µseg
Transistores	1950-	µseg
Circuito Impreso	1950-	µseg
Circuito Integrado	1960-	µseg - nseg
Microprocesadores	1971-	µseg - nseg
ASICs	1980-	nseg-pseg
Full-custom	1960-	nseg-pseg
Gate Arrays	1980-	nseg-pseg
Sea-of-Gates	1980-	nseg-pseg
Standard-Cells	1980-	nseg-pseg
Laser PGA	1996-	nseg-pseg
FPGAs	1985	nseg-pseg

Fuente: EDUARDO BOEMO SCALVINONI

2.7.1. LOS ASICS (Circuito Integrado de Aplicación Específica),

Las principales características de los ASICs se pueden resumir en la siguiente lista:

- Area-time-power mínimo;
- Mixtos;

- Alta fiabilidad;
- Alta Confidencialidad;
- Alto Costo;
- Vulnerabilidad a errores;
- Problemas con el stock;
- Problemas de ventana de mercado;
- Alta complejidad;
- Elevado número de transistores;
- Elevada frecuencia de operación;
- Herramientas EDA complejas.

De todos los puntos anteriores son dos los determinantes: 1º el costo que suele descartar esta opción tecnológica y 2º el consumo de potencia, que suele ser el factor principal que puede obligar a utilizar esta tecnología.La figura 2.4 muestra el estado actual de la tecnología ASIC, que lleva a que su utilización quede restringida a grandes circuitos.

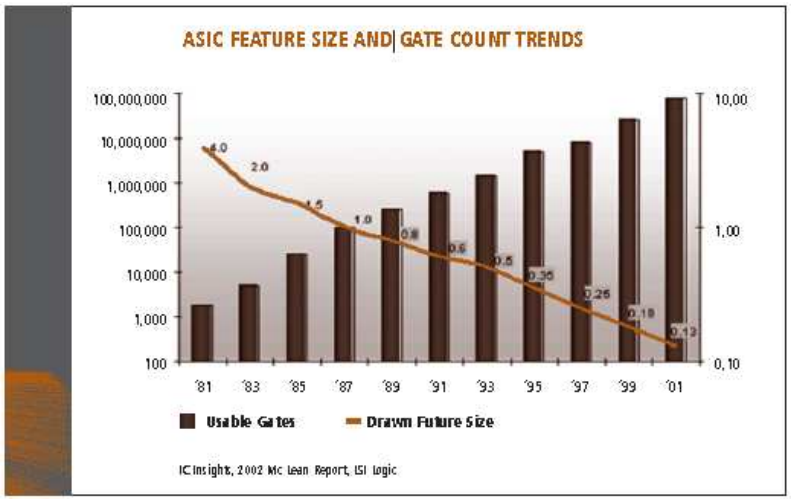


Figura 2. 4:Estado actual de la Tecnología ASIC

Fuente: EDUARDO BOEMO SCALVINONI

LaFigura 2.5 se muestra el proceso de fabricación de un ASIC, el cual se separa en dospartes: la superior (estándar) es común a todos los circuitos

mientras que la interior (custom) depende de cada diseño en particular. De allí la denominación de semicustom para este tipo de tecnología.

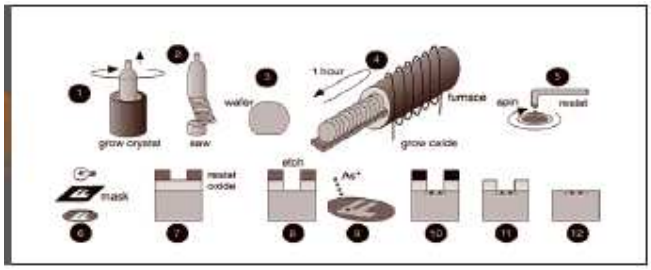


Figura 2. 5: Proceso de fabricación de una ASIC
Fuete: EDUARDO BOEMO SCALVINONI

Dentro de los ASICs destacan los Gate Arrays, de los cuales deriva el nombre de FPGA.Están formados por filas de transistores sin interconexión junto con canales intermedios para rutado. Permiten una utilización de hasta un 90% del área de silicio y se fabrica en diferentes tamaños. La idea central de los Gate Arrays es que con 4 transistores e interconexión, puede construirse cualquier puerta lógica.

Los transistores son estándares y las interconexiones customs. El proceso es el siguiente:

1. Se preparan las obleas con los arrays de transistores y los buffers de I/O,

2. Se almacenan en una "sala blanca", y finalmente

3. Se completa los pasos restantes cuando el cliente finaliza en diseño.

En la figura 2.6 se muestra el arreglo básico de transistores y las conexiones para crear una puerta.

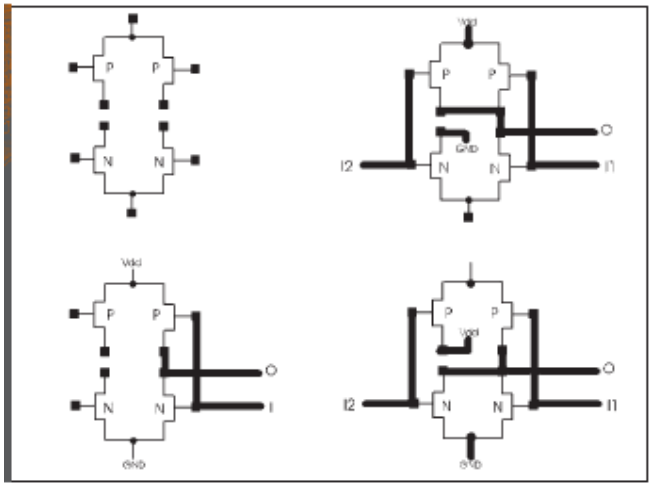


Figura 2. 6:Arreglo Básico de Transistores

Fuete: EDUARDO BOEMO SCALVINONI

En la figura 2.7 se muestra un esquema de un Gate Array y la INV. NAND y NOR. Disposición de los canales de rutado.

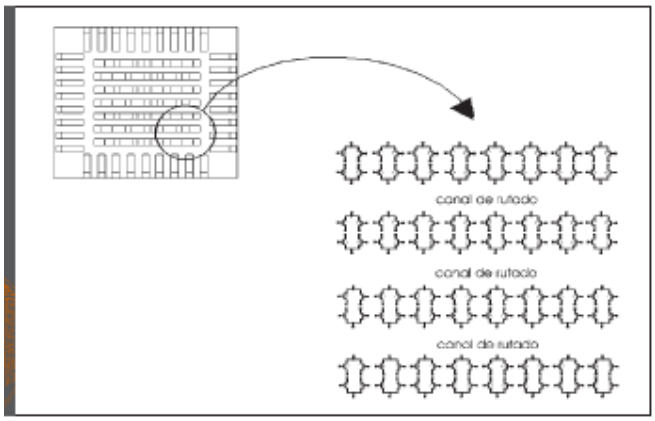


Figura 2. 7:Esquema de un Gate Array

Fuete: EDUARDO BOEMO SCALVINONI

Aunque los *Gate Arrays* y opciones tecnológicas posteriores como los *Sea-of-Gates* y los *Standard Cells* permiten realizar circuitos con área-retardo y consumo mínimo, el costo de fabricación y la vulnerabilidad a errores de diseño han limitado en la actualidad su utilización a grandes productos de consumo.

2.7.2. FPGA.

Introducidas por Xilinx en 1985. Son los dispositivos programables por el usuario de aplicación más general. Estos chips tienen unos componentes básicos que se pueden unir según las necesidades de diseño, Esta configuración se encuentra almacenada en una memoria ram interna, y se carga desde el exterior del chip. De igual forma que en los microcontroladores se carga el software, en las FPGA's se carga la configuración que determina en qué circuito se va a convertir.

Una FPGA's es un chip que según cómo se configure, puede realizar cualquier circuito digital. Una FPGA más grande, con más recursos internos, alcanza a implementar diseños más complejos. Pero al final se tiene una manera de poder crear diseños digitales sin tener que utilizar componentes externos. Y lo interesante es que una vez configurada la FPGA, lo que tenemos en su interior es hardware.

Cuando se aborda el diseño de un sistema electrónico y surge la necesidad de implementar una parte con hardware dedicado son varias las posibilidades que hay. En la figura 2.8 se han representado las principales aproximaciones ordenándolas en función de los parámetros coste, flexibilidad, prestaciones y complejidad.

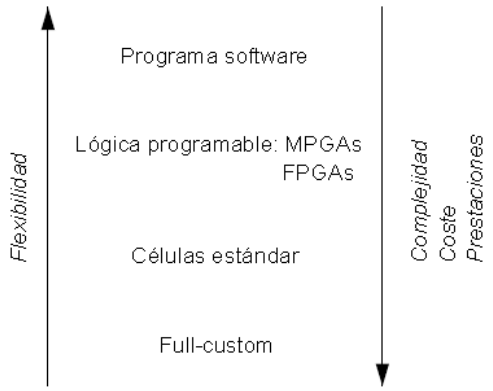


Figura 2. 8:Diferentes soluciones para el diseño de circuitos digitales
Fuente: M. L. López Vallejo

Como se puede apreciar de la figura 2.8, las mejores prestaciones las proporciona un diseño full-custom, consiguiéndose a costa de elevados costes y enorme complejidad de diseño. En el otro extremo del abanico de posibilidades se encuentra la implementación software, que es muy barata y fexible, pero que en determinados casos no es válida para alcanzar un nivel de prestaciones relativamente alto.

Entre estas dos opciones se puede elegir la fabricación de un circuito electrónico realizado mediante diseño semi-custom, utilizando células estándar, o recurrir a un circuito ya fabricado que se pueda programar “ in situ”, como son las FPGAs. De estas dos opciones la primera proporciona mejores prestaciones, aunque es más cara y exige un ciclo de diseño relativamente largo.

Por otro lado, los dispositivos lógicos programables constituyen una buena oferta para realizar diseños electrónicos digitales con un buen compromiso coste-prestaciones. Y lo que es mejor, permiten obtener una implementación en un tiempo de diseño asombrosamente corto (con la consiguiente reducción del parámetro).

Otro aspecto que se debe tener en cuenta para decidirse por este tipo de implementación es que el coste de realización es muy bajo, por lo que suele ser una buena opción para la realización de prototipos.

En este marco teórico de tesis se describirá de forma muy somera en qué consisten estos dispositivos, particularizando para una familia del fabricante Xilinx. La información contenida en ellas se basa en gran medida en las siguientes fuentes: sobre arquitectura de FPGAs [BFRV92, Xil91,CSR+99a]; sobre diseño de circuitos [WE94, CSR+99b].

2.7.2.1. Evolución de los dispositivos programables

Se entiende por dispositivo programable aquel circuito de propósito general que posee una estructura interna que puede ser modificada por el usuario final (o a petición suya, por el fabricante).

Para implementar una amplia gama de aplicaciones. El primer dispositivo que cumplió estas características era una memoria PROM, que puede realizar un comportamiento de circuito.

Para ello el fabricante proporciona las herramientas de diseño adecuadas. Los elementos básicos constituyentes de una FPGA como las de Xilinx se pueden ver en la figura 2.9 y son los siguientes:

1. Bloques lógicos, cuya estructura y contenido se denomina arquitectura. Hay muchos tipos de arquitecturas, que varían principalmente en complejidad (desde una simple puerta hasta módulos más complejos o estructuras tipo PLD). Suelen incluir biestables para facilitar la implementación de circuitos secuenciales. Otros módulos de importancia son los bloques de Entrada/Salida,
2. Recursos de interconexión, cuya estructura y contenido se denomina arquitectura de rutado.
3. Memoria RAM, que se carga durante el RESET para configurar bloques y conectarlos.

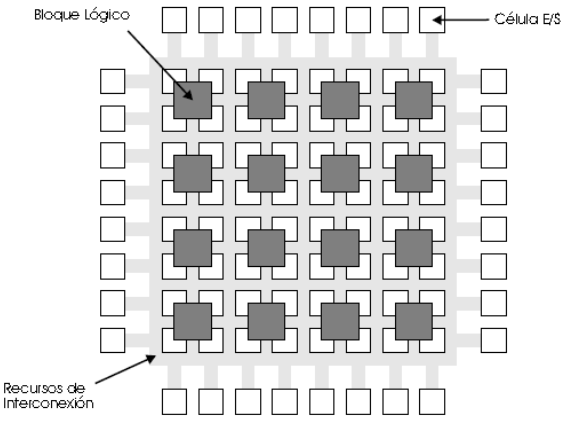


Figura 2. 9:Estructura general de una FPGA (en concreto de XILINX)

Fuente: M. L. López Vallejo

Entre las numerosas ventajas que proporciona el uso de FPGAs dos destacan principalmente:

El bajo coste de prototipado y el corto tiempo de producción. No todo son ventajas. Entre los inconvenientes de su utilización están su baja velocidad de operación y baja densidad lógica (poca lógica implementable en un solo chip). Su baja velocidad se debe a los retardos introducidos por los conmutadores y las largas pistas de conexión.

Por supuesto, no todas las FPGA son iguales. Dependiendo del fabricante nos podemos encontrar con diferentes soluciones. Las FPGAs que existen en la actualidad en el mercado se pueden clasificar como pertenecientes a cuatro grandes familias, dependiendo de la estructura que adoptan los bloques lógicos que tengan definidos. Las cuatro estructuras se pueden ver en la figura 2.10, sin que aparezcan en la misma los bloques de entrada/salida.

1) Matriz simétrica, como son las de XILINX
2) Basada en canales, ACTEL
3) Mar de puertas, ORCA
4) PLD jerárquica, ALTERA o CPLDs de XILINX.

En concreto, para explicar el funcionamiento y la estructura básica de este tipo de dispositivos programables solo se consideraran las distintas familias de XILINX.

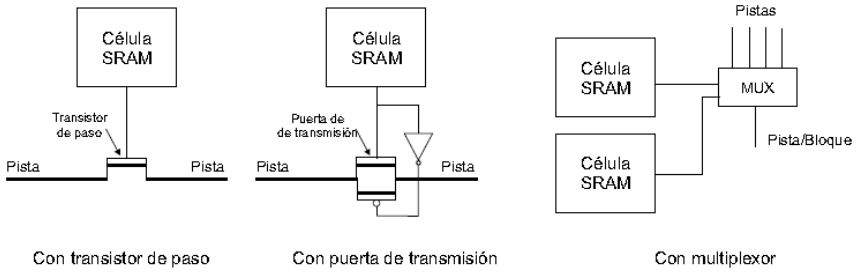


Figura 2. 10:Tipos de conectores utilizados por XILINX

Fuente: M. L. López Vallejo

Es importante destacar que si se utilizan células SRAM la conjuración de la FPGA será válida únicamente mientras esté conectada la alimentación, pues es memoria volátil. En los sistemas finales está claro que hace falta algún mecanismo de almacenamiento no volátil que cargue las células de RAM. Esto se puede conseguir mediante EPROMs o disco.

Este elemento de programación es relativamente grande (necesita por lo menos 5 transistores), pero se puede implementar en el proceso normal de fabricación del circuito (es CMOS). Además, permite reconfigurar la FPGA de una forma muy rápida.

2.7.2.2. Descripción de las principales familias

Hay múltiples familias lógicas dentro de XILINX. Las primeras que surgieron son: XC2000 (descatalogada en el año 1999), XC3000 y XC4000, correspondientes respectivamente a la primera, segunda y tercera generación de dispositivos, que se distinguen por el tipo de bloque lógico configurable (CLB) que contienen. En la actualidad existen también las familias de FPGA SpartanII, SpartanIII, Virtex, VirtexII y VirtexPro. La tabla 2.2 muestra la cantidad de CLBs que puede haber en cada FPGA de las familias base y ese mismo valor expresado en puertas equivalentes.

Tabla 2. 2: Familias del fabricante XILINX

SERIE	TIPO CLB	Nº DE CLBs	PUERTAS EQUIVALENTES
XC2000	1 LUT, 1FF	64-100	1200-1800
XC3000	1 LUT, 2FF	64-484	1500-7500
XC4000XL	3 LUT, 2FF	64-3136	1600-180000

Fuente: M. L. López Vallejo

2.7.2.3. Metodología de diseño con FPGA's

La metodología de diseño es similar a la cualquier sistema digital, salvo que al final se obtiene un archivo ejecutable que se descarga a la FPGA para que se reconfigure, implementando así el diseño esperado. Primero hay que tener una descripción del circuito a realizar. Tradicionalmente en las ingenierías se realizan planos o esquemas para esta descripción, de forma similar a como un arquitecto diseña un edificio. Sin embargo es posible realizar una descripción del hardware utilizando algún lenguaje de descripción de hardware, como VHDL o Verilog, con esta descripción se pueden realizar simulaciones del circuito, para comprobar que lo diseñado trabaja correctamente de lo contrario se volverá a modificar la descripción (esquemas o programa) hasta que la simulación sea satisfactoria. Hasta aquí sólo se ha utilizado el computador y no se ha tocado hardware. Sin embargo en el caso del software, la propia simulación es la ejecución del programa. Se observa directamente el resultado del programa y se modifican el código fuente hasta que se eliminen los errores.

En el caso del hardware hay que construir el circuito. Y aquí es donde vienen las FPGA's para hacerlo. A partir de la especificación hardware y utilizando un compilador especial, obtenemos un archivo binario, llamado bitstream que contiene toda la información necesaria para configurar la FPGA. Este archivo, que es el equivalente a un programa ejecutable en el caso del software, es el que hay que cargar en la FPGA. Se carga este archivo en la FPGA y listo. Ya se tiene el hardware que queríamos situado en el interior de un chip. No se ha tenido que soldar, ni comprar componentes, ni perder tiempo haciendo un prototipo. Ahora los cambios en el diseño se pueden hacer igual de rápidos que en el caso de software. Sólo hay que cambiar la especificación del diseño, volver a compilar y reconfigurar la FPGA con el nuevo bitstream generado.

CAPÍTULO 3: DESARROLLO EXPERIMENTAL

3.1. Práctica 1: Encendido de LEDs usando la Tarjeta DE0 Nano de Altera programada bajo la plataforma NIOS II de Eclipse.

La presente práctica se trabaja con la Tarjeta DE0 Nano de Altera (véase figura 3.1), que es la que nos permite integrar varios dispositivos electrónicos, denominándose Sistemas Embebidos.

Objetivos de la Práctica 1:

- Configurar el banco de LEDs de la Tarjeta DE0 Nano de Altera por medio del Quartus II.
- Desarrollar un algoritmo en la plataforma Nios II de Eclipse para realizar el encendido de LEDs de la Tarjeta DE0 Nano de Altera.

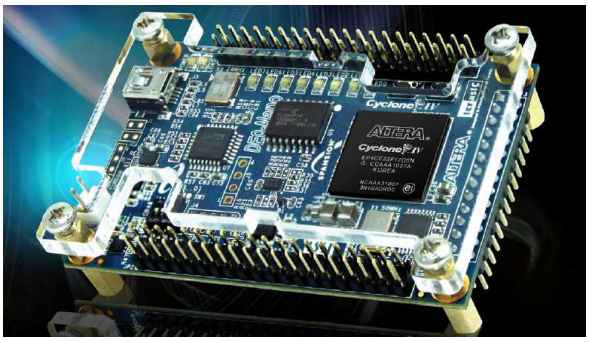


Figura 3. 1: Tarjeta DE0 Nano de Altera.
Fuente: El Autor.

Descripción:

Para la práctica #1, se va a proceder a realizar el encendido de LEDs pertenecientes a la Tarjeta DE0 Nano. En la figura 3.2 se puede observar el diagrama de bloques.

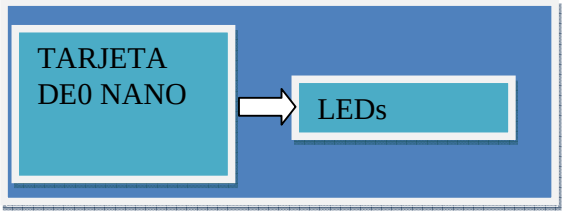


Figura 3. 2: Diagrama de bloques de la práctica #1.
Fuente: El Autor.

Desarrollo:

La práctica se desarrolla inicialmente haciendo una copia del archivo cuyo nombre es DE0-Nano, como se muestra en la figura 3.3:

Figura 3. 3: Copia del archivo DE0-Nano-copia.
Fuente: El Autor.

Una vez realizado este procedimiento, se procede a cambiarle el nombre a “DE0-Nano-LEDs”como se muestra en la figura 3.4.

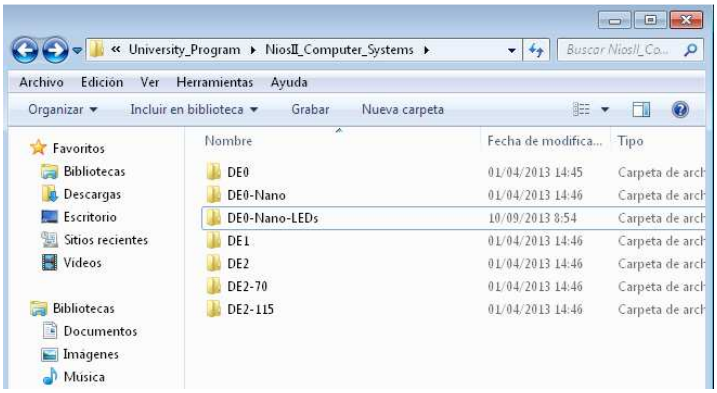


Figura 3. 4: Archivo DE0-Nano_LEDs (modificado).
Fuente: El Autor.

QUARTUS II DE ALTERA

Posteriormente, se debe tener previamente instalados la plataforma de programación en VHDL denominada QUARTUS II (véase la figura 3.5):

Figura 3. 5: Ventana de inicio de QUARTUS II.
Fuente: El Autor.

Una vez abierto el QUARTUS II, se debe realizar lo siguiente: File, Open Project (ver figura 3.6) y después seleccionar el archivo "DE0_Nano_Basic_Computer" (ver figura 3.7). O a su vez seguir la ruta: C:\altera\12.1\University_Program\NiosII_Computer_Systems\DE0-Nano-LEDs\DE0-Nano_Basic_Computer\verilog.

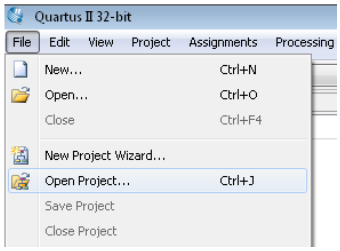


Figura 3. 6: Selección para abrir un proyecto existente.
Fuente: El Autor.

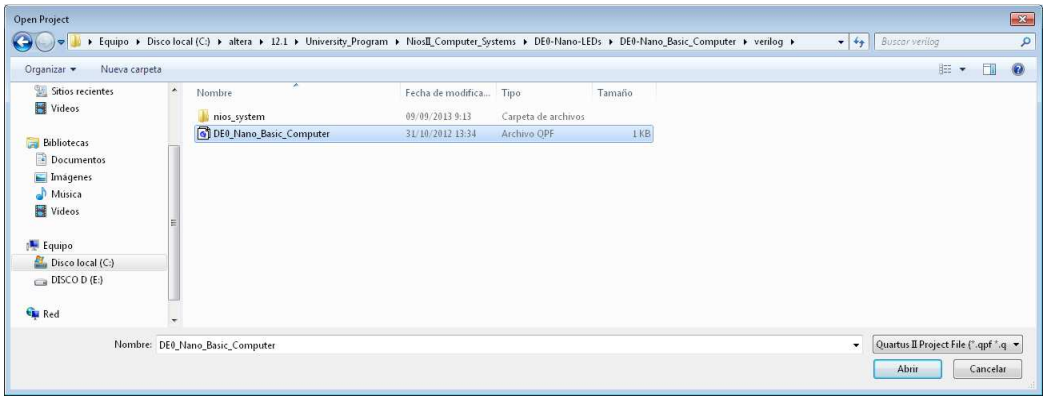


Figura 3. 7: Selección del archivo DE0_Nano_Basic_Computer.
Fuente: El Autor.

Después se muestra el archivo listo para ser procesado, desde QUARTUS II (ver figura 3.8).

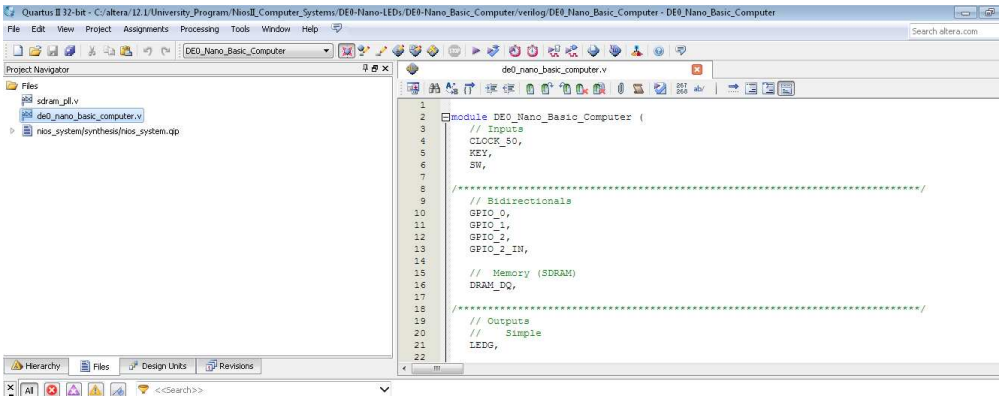


Figura 3. 8: Ventana de Quartus II listo para su procesamiento.
Fuente: El Autor.

QSYS DEL QUARTUS II DE ALTERA

De la figura 3.8, después escogemos el icono “Editor de elementos en Qsys” de Altera (ver figura 3.9) y con esta selección aparecerá la ventana de inicio de Qsys (ver figura 3.10):

Figura 3. 9: Selección del icono Qsys de Altera.
Fuente: El Autor.

Figura 3. 10: Ventana de Qsys de Altera.
Fuente: El Autor.

En la figura 3.10, la ventana que se abre nos indica que se debe de abrir el archivo de extensión .qsys, él mismo que se encuentra ubicado en la carpeta de la práctica “DE0-Nano_LEDs” con directorio:C:\altera\12.1\University_Program\NiosII_Computer_Systems\DE0-Nano-LEDs\DE0-Nano_Basic_Computer\verilog.Se selecciona el archivo: nios_system.qsys(ver figura 3.11) y se hace clic en abrir.

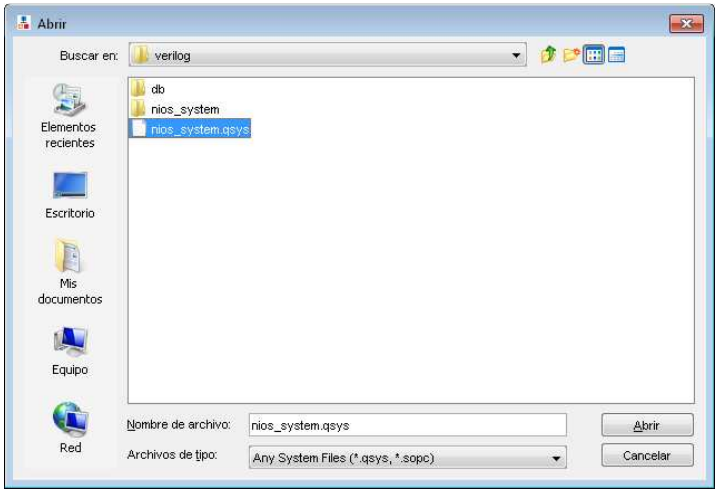


Figura 3. 11: Selección del archivo nios_system.qsys.
Fuente: El Autor.

Posterior a la selección que se visualizó en la figura 3.11, se mostrará una nueva ventana con mensajes de advertencias, lo que indica que no hay problemas, y hacemos clic en close (ver figura 3.12).

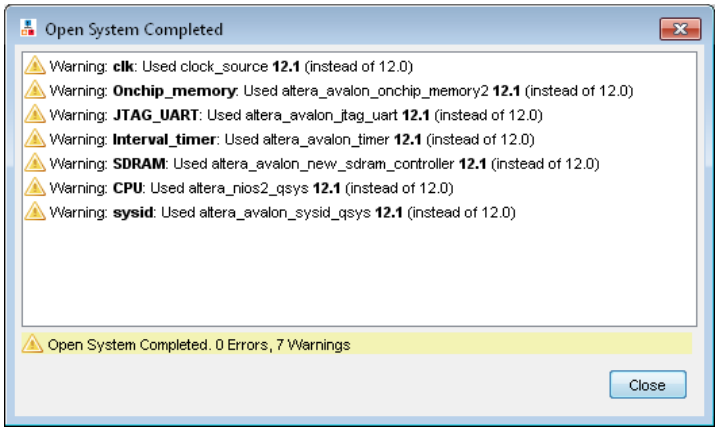


Figura 3. 12: Ventana del Open SystemCompleted sin errores.
Fuente: El Autor.

En la figura 3.13 se muestra la ventana principal de Qsys, que mediante la barra de desplazamiento vertical, se puede apreciar todos los componentes pertenecientes a la tarjeta DE0 Nano.Se debe buscar y comprobar que el banco de LEDs, forma parte de los componentes, así como se muestra en la siguiente 3.13.

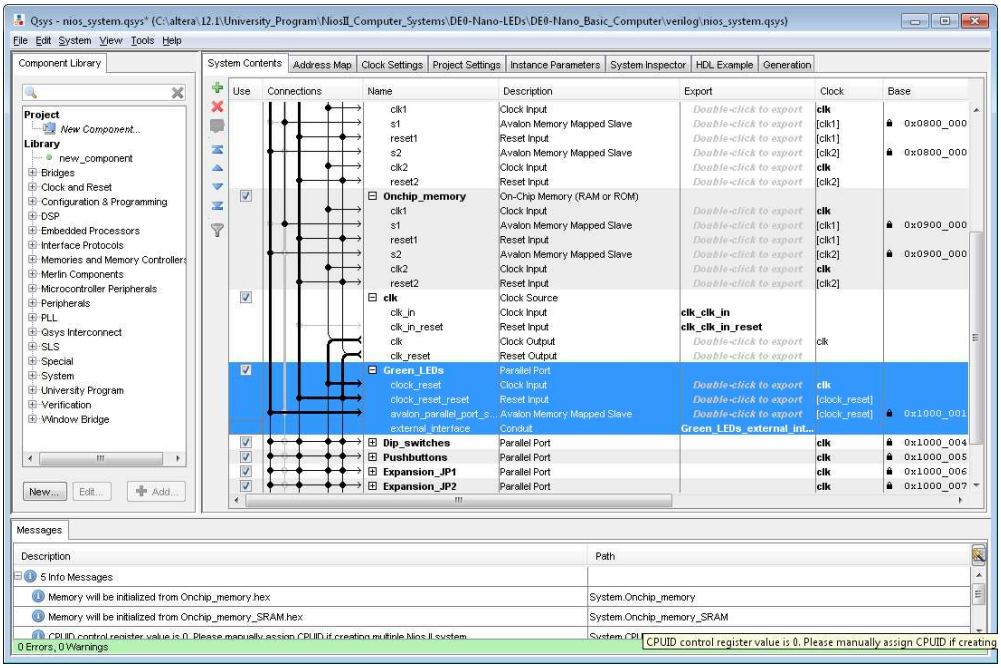


Figura 3. 13: Configuración del banco de LEDs en Qsys de Altera.
Fuente: El Autor.

En el siguiente paso procedemos a generar la máquina (ver figura 3.14):

- Seleccionamos el menú Generation.
- Hacer clic en Generate para generar el sistema.

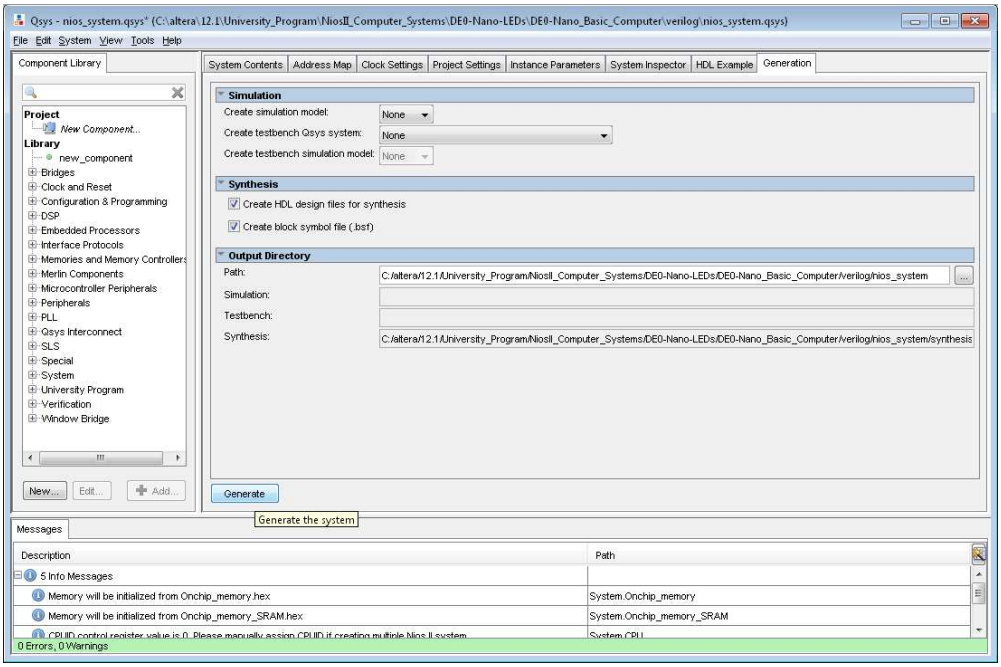


Figura 3. 14: Ventana para generar máquina.
Fuente: El Autor.

Una vez realizada “Generate” se procede a guardar los cambios, y se espera hasta que termine de realizar la compilación respectiva, para lo cual la figura 3.15 nos muestra la compilación sin errores para la construcción de la máquina.

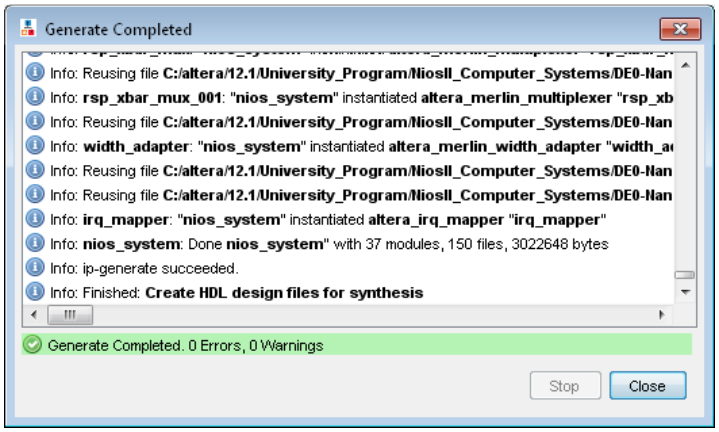


Figura 3. 15: Ventana de compilación Qsyspara generar máquina.
Fuente: El Autor.

Posteriormente, al regresar a QUARTUS II, se selecciona el ícono StartCompilation, el cual se encargará de compilar la máquina en Quartus II,

hasta que su valor llegue a un 100% y sin errores, como se indica en la figura 3.16.

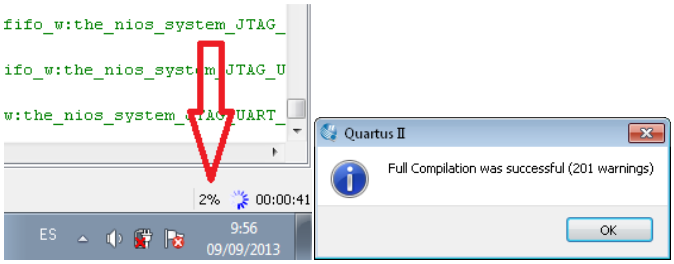


Figura 3. 16: Ventana de compilación QUARTUS II práctica 1.
Fuente: El Autor.

PLATAFORMA NIOS II DE ECLIPSE.

Ahora se utiliza la plataforma NIOS II de Eclipse (ver figura 3.17), la que nos permite integrar como un sistema embebido.

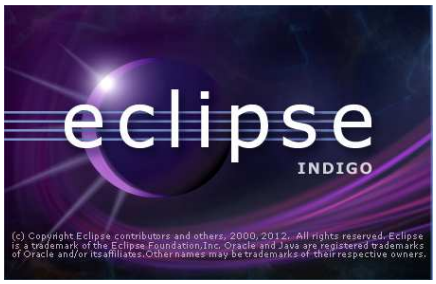


Figura 3. 17: Ventana de inicio de plataforma NIOS II de Eclipse.
Fuente: El Autor.

En esta plataforma, selecionamos un espacio de trabajo dentro de la carpeta de ubicación de la práctica de endendido de LEDs, como se muestra en la figura 3.18.

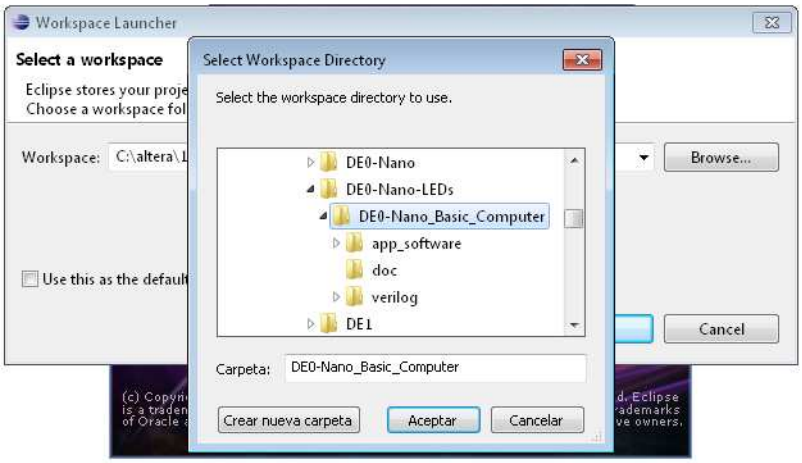


Figura 3. 18: Ventana de Select Workspace Directory.
Fuente: El Autor.

Una vez realizado lo anterior (figura 3.18), aparecerá la ventana principal del NIOS II de Eclipse.

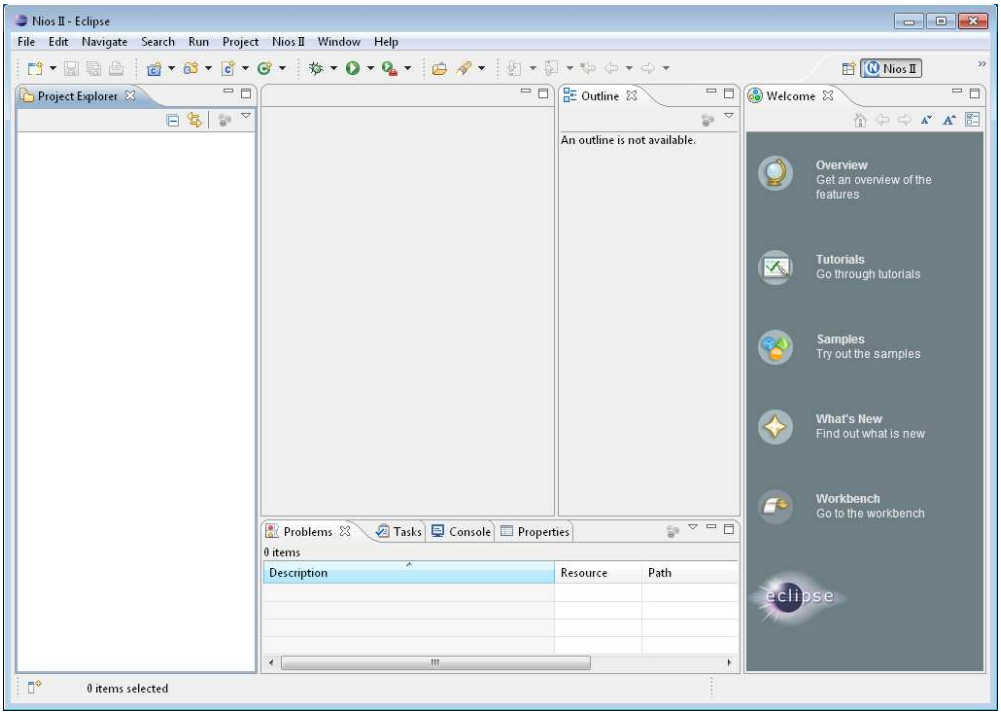


Figura 3. 19: Ventana principal de NIOS II – Eclipse.
Fuente: El Autor.

En la cual, procedemos a la creación de un nuevo proyecto en NIOS II, realizando los siguientes pasos:File - New – Nios II Application And BSP fromTemplate.

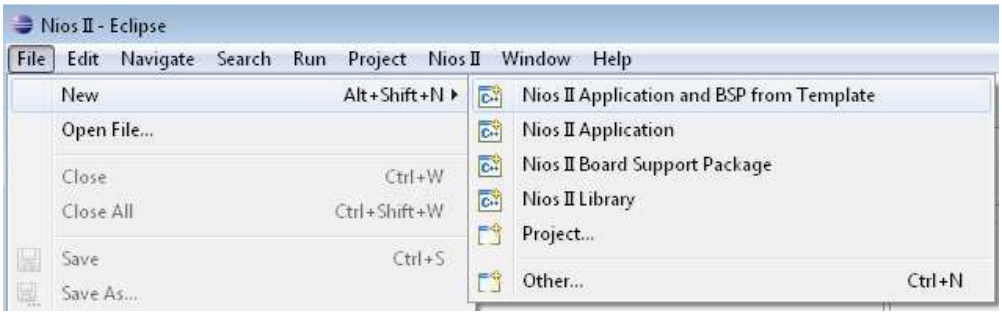


Figura 3. 20: Nuevo proyecto para el NIOS II de Eclipse.
Fuente: El Autor.

Una vez creado el proyecto, seleccionamos el archivo nios_system.sopcinfo haciendo clic en SOPC Information File Name.

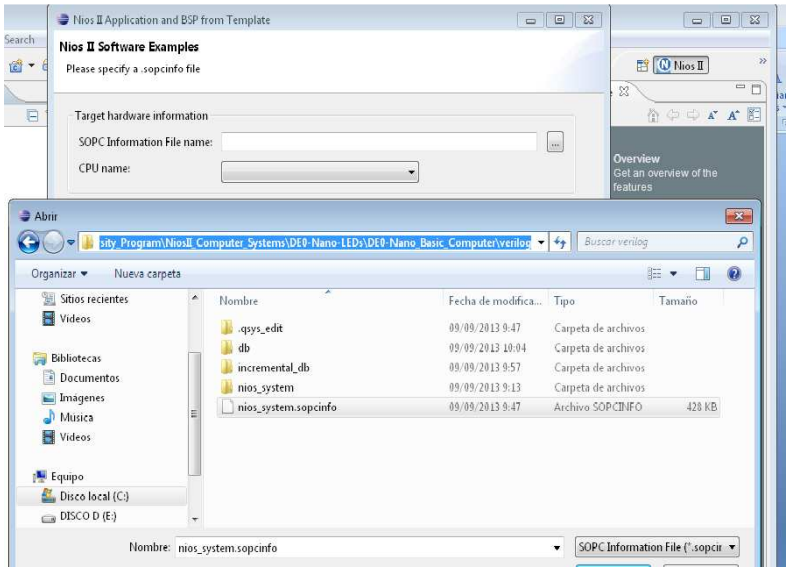


Figura 3. 21: Ventana para la creación final de la máquina.
Fuente: El Autor.

El directorio debeestablecer y mostrar una máquina lista para ser utilizada en el menú de CPU namey a la vez escribimos el nombre del proyecto (ver figura 3.22):

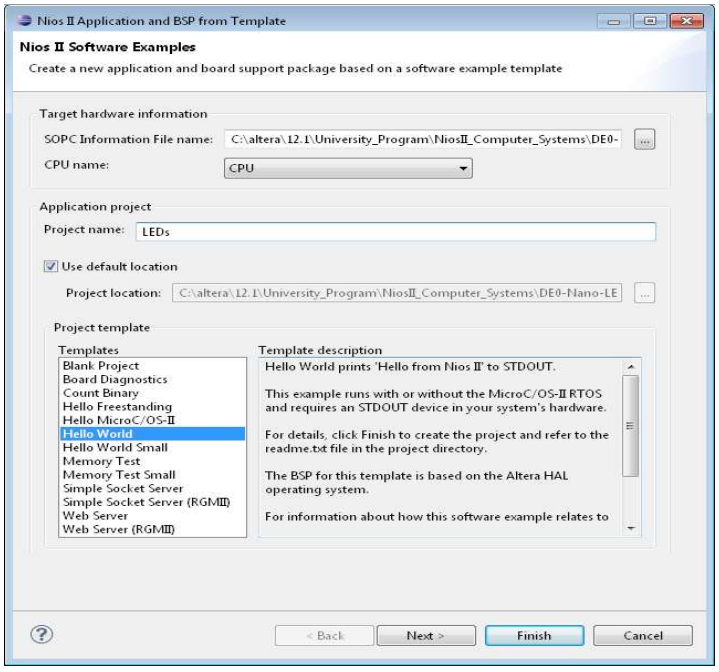


Figura 3. 22: Máquina creada para práctica 1.
Fuente: El Autor.

PROGRAMACION DE LA FPGA

Para procedes a la programación de la FPGA, debemos tener conectado la tarjeta DE0 Nano al computador/laptop.Antes de realizar código alguno, se debe realizar la programación de la máquina construida anteriormente haciendo los siguientes pasos:Menú Nios II – Quartus II Programmer (ver figura 3.23).

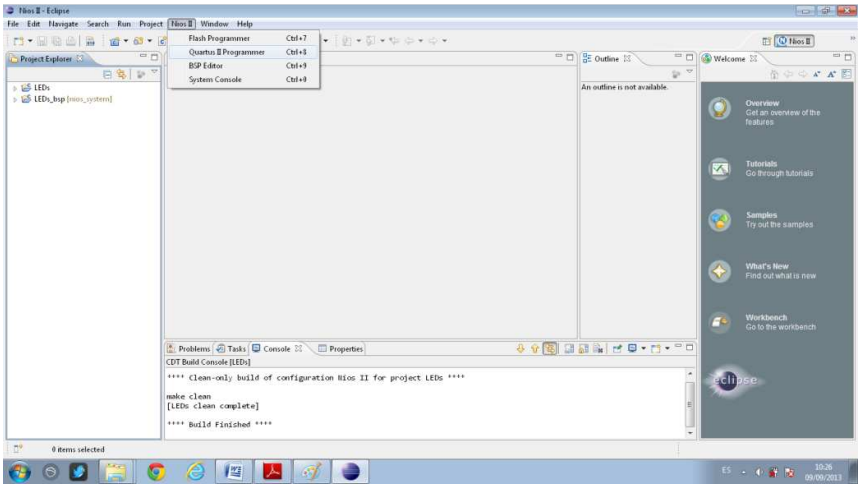


Figura 3. 23: Ventana para programación de la máquina.
Fuente: El Autor.

Una vez más estamos ubicados en la ventana principal del Programador de Quartus II, ahora se procede a añadir el archivo DE0_Nano_Basic_Computer.sof y se hace clic en Open (ver figura 3.24).

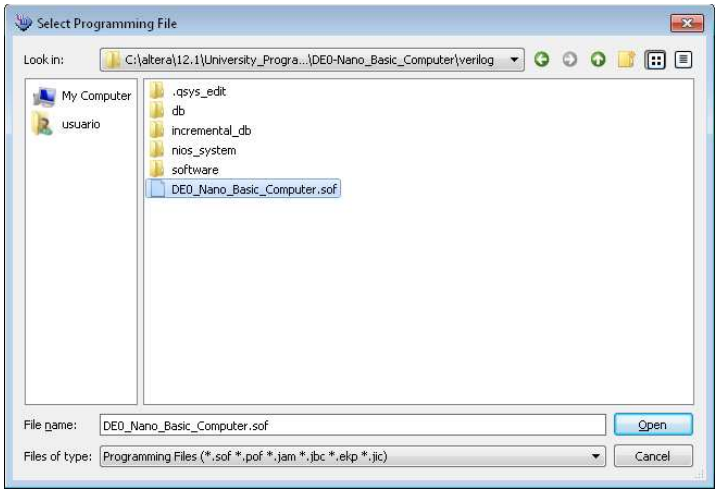


Figura 3. 24: Ventana para la selección del archivo de programa Quartus II.
Fuente: El Autor.

Una vez realizado lo indicado con anterioridad, se muestra la figura 3.25 unanueva vista,que nos presenta la ventana principal indicándonos que el archivo se ha cargado exitosamente.

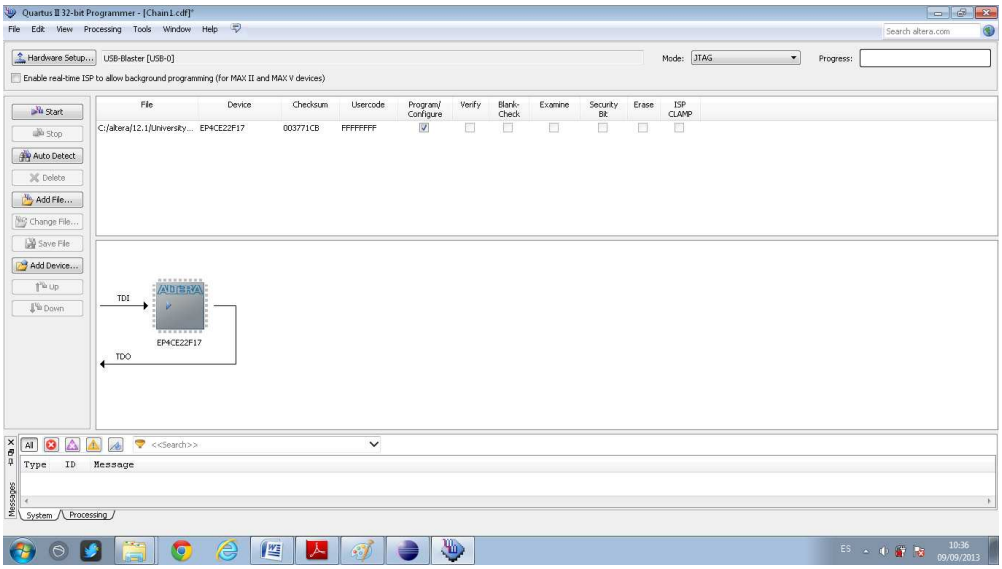


Figura 3. 25: Ventana para la creación final de la máquina.
Fuente: El Autor.

En la ventana superior derecha se mostrará un indicador del estado de la programación de la tarjeta (ver figura 3.26).

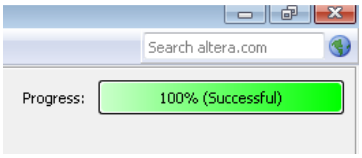


Figura 3. 26: Estado de la programación en la FPGA Nano Station.
Fuente: El Autor.

La tarjeta esta lista para ser programada en NIOS II, a continuación se muestra la programación del sistema embebido al NIOS II de Eclipse.

PROGRAMACION EN NIOS II DE ECLIPSE.

Código:

```
/*
 *  Creado: 09/09/2013
*/
/*DescripciónEncendidodeleds
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
#define GREEN_LEDS_BASE 0x10000010 //Ubicación del puerto de LEDs
intmain ()
{
        int i=0xFF;    //Variable quesemostraraen el puertode LEDs
        //fp_led = nombredepuntero
        //GREEN_LEDS_BASE = Direcciónde lbancode LEDs,
        //              palabrareservadaubicadaen el archivosystem.h
        volatileint    *    fp_led    =    (int    *)    GREEN_LEDS_BASE;    //
        Punteroasignadoalbancode LEDs
        printf ("ENDENDIDO DE LEDS...\n"); // ImprimemensajeenConsole.
        *(fp_led)=i; //Encendidode LEDs según el valor escritoenla   variable i
        return 0;
}
```

Después, copiamos el código correspondiente en la ventana de desarrollo como se muestra en la figura 3.27.

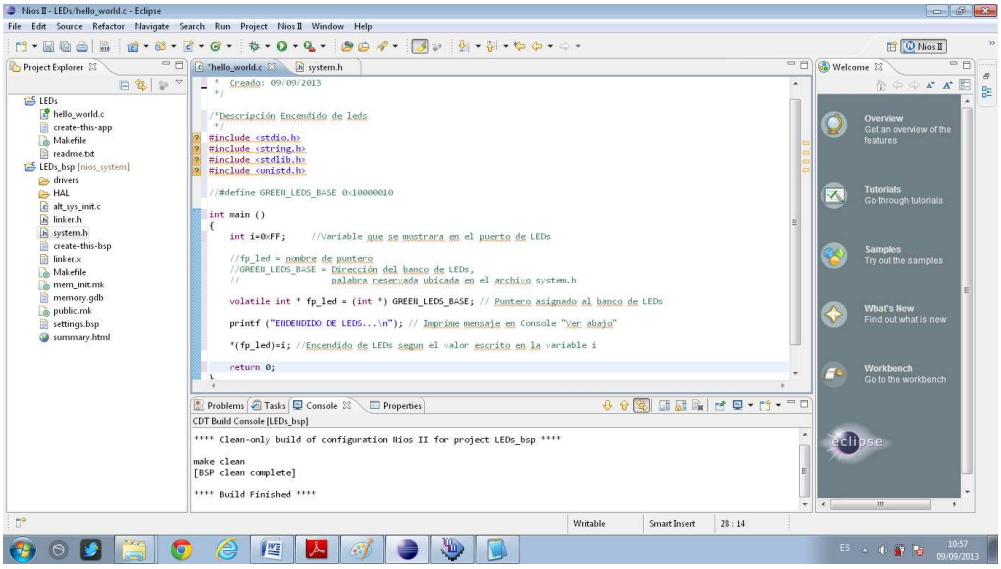


Figura 3. 27: Estado de la programación en la FPGA Nano Station.
Fuente: El Autor.

CONSTRUCCION DEL PROYECTO.

Es necesario el archivo de extensión .elf para su compilación y se deben realizar los siguientes pasos: Project – BulidAll (ver figura 3.28):

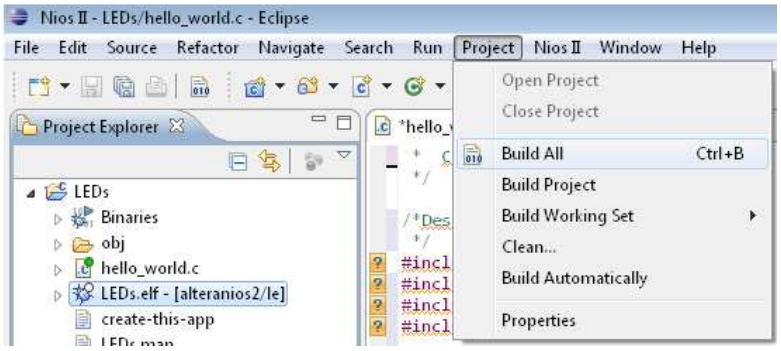


Figura 3. 28: Ventana para Construcción del Proyecto (BuildAll).
Fuente: El Autor.

Entonces aparecerá un archivo de extensión .elf. y un mensaje como se muestra en la siguiente figura en el menú de console "BuildFinished" (ver figura 3.29).

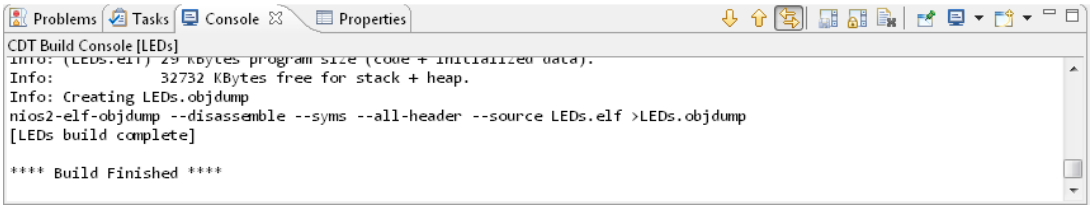


Figura 3. 29: Estado del BuildConsole
Fuente: El Autor.

COMPILACION DEL CODIGO.

Para su respectiva compilación nos dirigimos al RunConfiguration para realizar los cambios respectivos como se muestra en la figura 3.30.

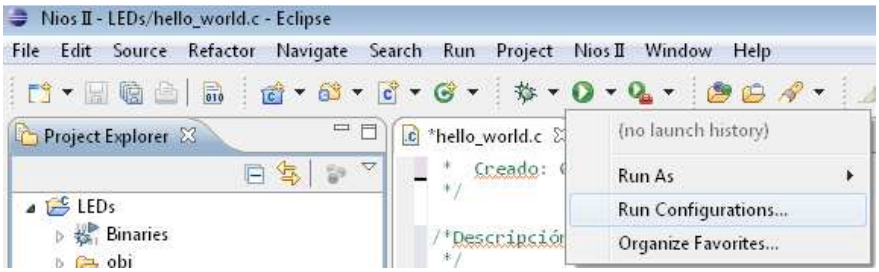


Figura 3. 30: Configuración para ejecución de compilación.
Fuente: El Autor.

Al seleccionar RunConfiguration, nos aparece una ventana en la cual debemos escoger la opción de Nios II Hardware (ver figura 3.31).

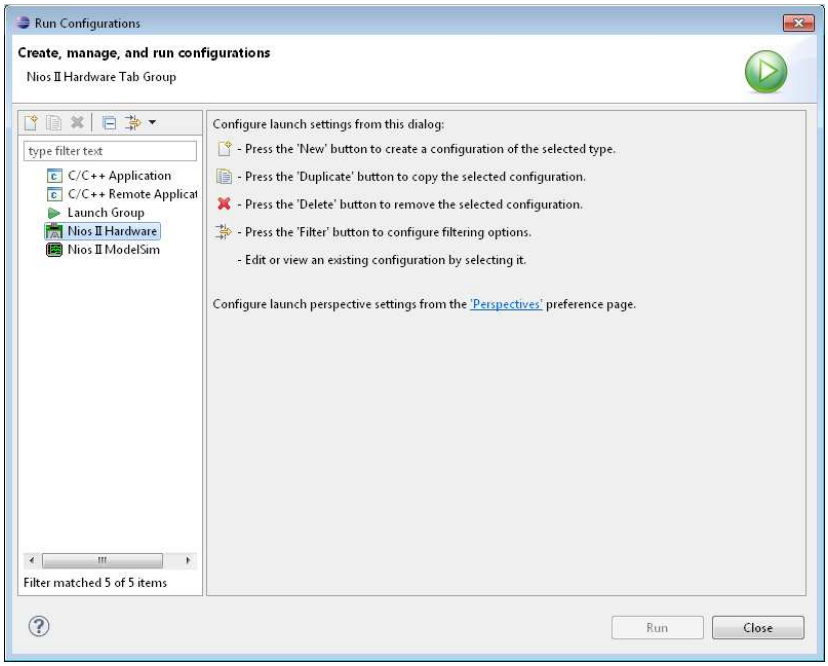


Figura 3. 31: Ventana RunConfiguration del Hardware.
Fuente: El Autor.

Una ventana nos pide guardar los cambios realizados en el código, seleccionamos YES. En el menú de Nios II Console se despliega y nos muestra el mensaje de “Encendido de LEDs” (ver figura 3.32).

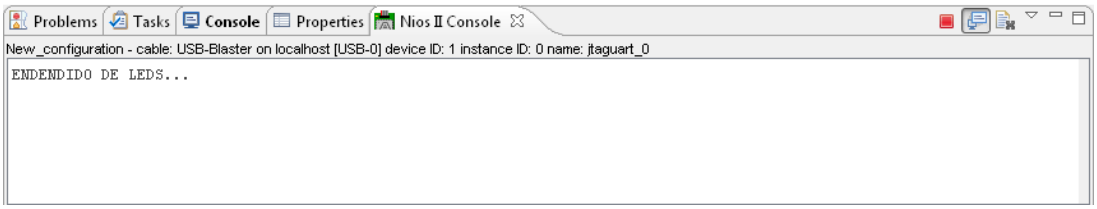


Figura 3. 32: Ventana que muestra el encendido de LEDS.
Fuente: El Autor.

3.2. Práctica 2: Conversión Analógica Digital (ADC) con la Tarjeta DE0 Nano de Altera programada bajo la plataforma NIOS II de Eclipse.

Para la presente práctica se trabajará en forma similar a la práctica 1, con la diferencia que ahora se realiza una conversión Analógica Digital (ADC).

Objetivos de la Práctica 2:

- Configurar el componente de Conversión Analógica Digital (ADC) de la Tarjeta DE0 Nano de Altera por medio de Qsys del Quartus II de Altera.
- Desarrollar un algoritmo en la plataforma Nios II de Eclipse para realizar la lectura de valores provenientes de un sensor de luz con la Tarjeta DE0 Nano de Altera.

Descripción:
En la siguiente práctica se va a proceder a realizar un configuración desde el Qsys del Quartus II de Altera para realizar la lectura de datos que manifiesta un sensor de luz utilizando la Conversión Analógica Digital (ADC).

En la figura 3.33 se muestra el diagrama de bloques de la práctica 2, en la que se trabaja con la tarjeta, laptop y convertidor A/D.

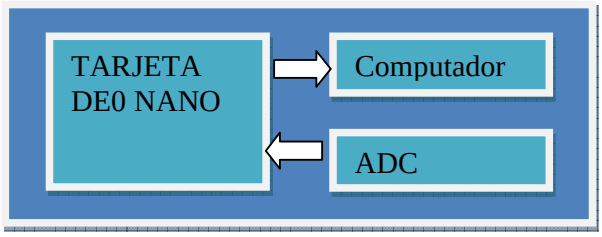


Figura 3. 33: Diagrama de bloques de la práctica #1.
Fuente: El Autor.

Desarrollo:

Los pasos para el desarrollo de la práctica 2, en general son los mismos de la práctica anterior, aquí se explicará cuáles son las modificaciones o excepciones que dependen del ADC.

En la práctica 1 se modificó un archivo, aquí es lo mismo, pero el archivo se debe escribir como DE0-Nano-ADC, tal como se muestra en la figura 3.34.

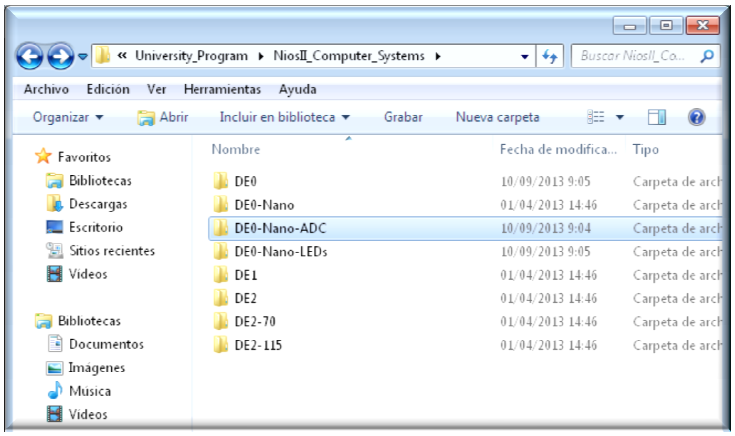


Figura 3. 34: Archivo DE0-Nano-ADC (modificado).
Fuente: El Autor.

Para la creación de un nuevo proyecto en QUARTUS II de Altera, se deben seguir los pasos de la práctica 1, pero adaptado al archivo DE0-Nano-ADC. Así mismo, se debe realizar con el Qsys de Altera.Para hacer uso del convertidor analógico digital, nos dirigimos al menú de Component Library, para incorporar una librería ADC. Se digita el nombre del componente ADC y el buscador automáticamente nos presentará una librería con el nombre de DE0-Nano-ADC Controller, como se muestra en la figura 3.35.

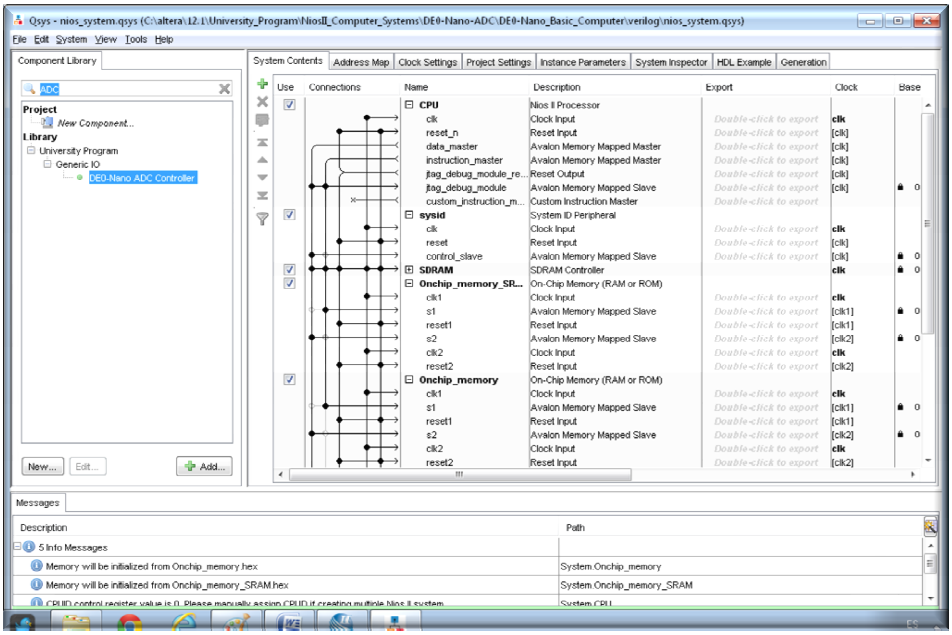


Figura 3. 35: Librería del Qsys para DE0-Nano-ADC.
Fuente: El Autor.

Posterior a esto se muestra una ventana del controlador ADC del DE0-Nano-ADC, él mismo muestra una configuración de hasta 8 canales de lectura ADC, tal como se muestra en la figura 3.36.

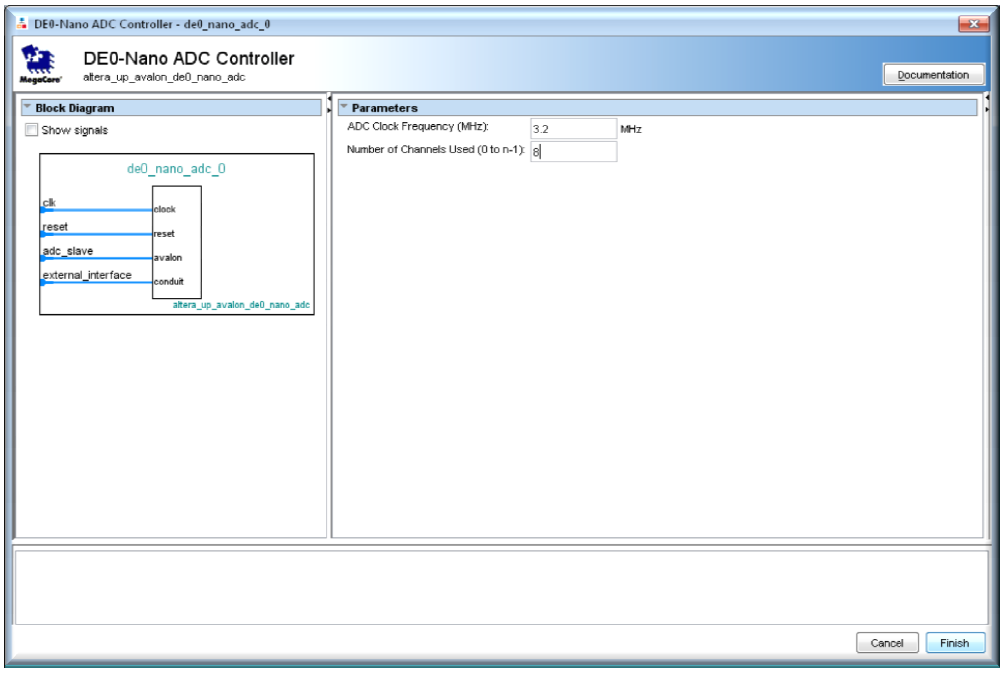


Figura 3. 36: Configuración de librería DE0-Nano-ADC.
Fuente: El Autor.

En la figura 3.37 se puede apreciar que al añadir una nueva librería, aparecen errores descritos en la ventana inferior de “Description”.También se puede observar que la librería de ADC ha sido incorporada con éxito.

Para configurar la nueva librería se debe empezar haciendo las conexiones respectivas, así como se observa en la figura 3.38.

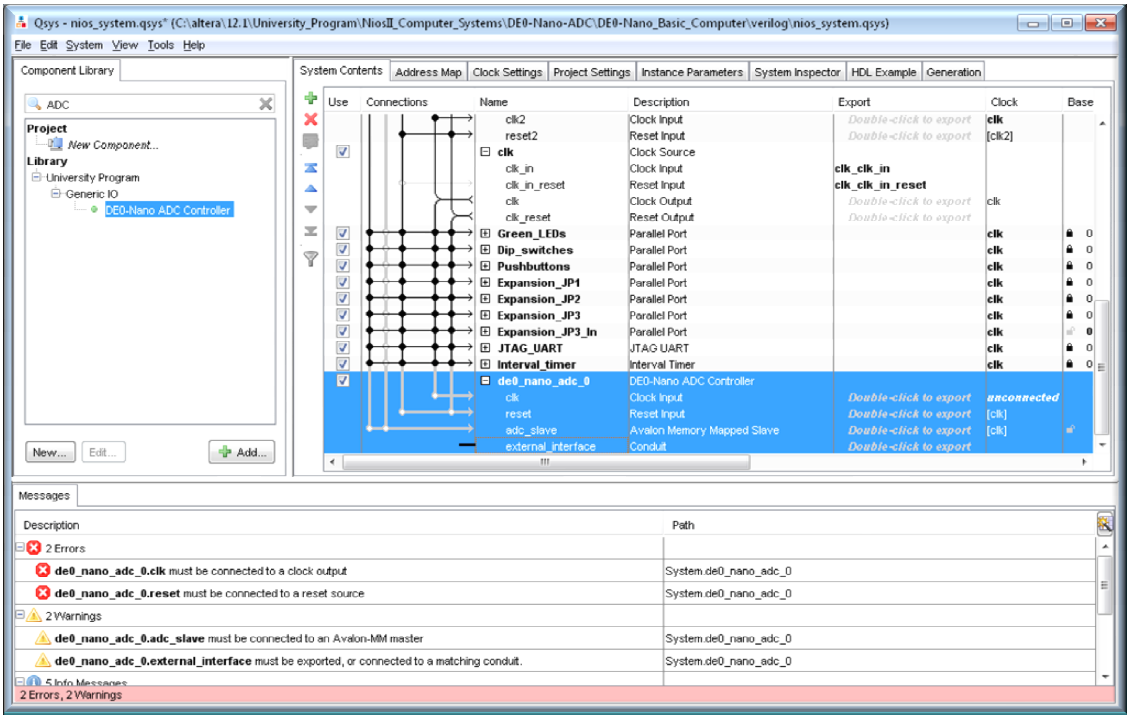


Figura 3. 37: Compilación de la librería DE0-Nano-ADC.
Fuente: El Autor.

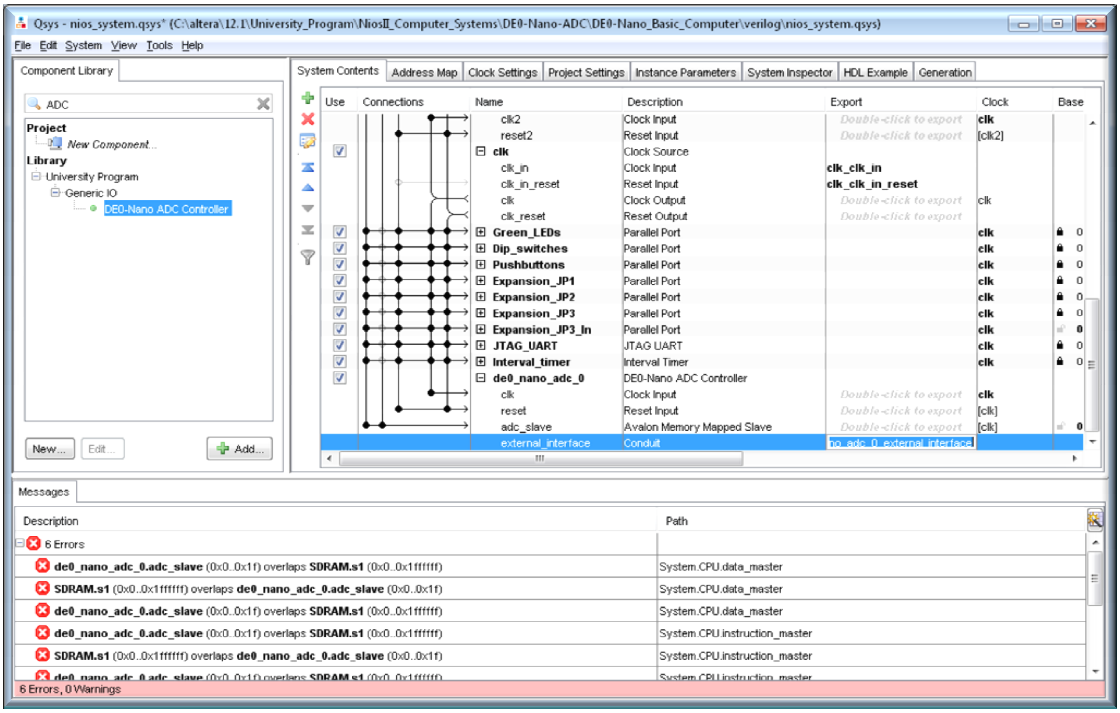


Figura 3. 38: Conexiones de la librería DE0-Nano-ADC.
Fuente: El Autor.

Una vez realizada las conexiones, es importante exportar la señal de External Interface, haciendo doble clic en el menú de Export. Adicionalmente, cuando las conexiones han finalizado, debemos proceder a la eliminación de errores, realizando los siguientes pasos para asignar espacio de memoria: Click en System y Click en Assign Base Addresses (ver figura 3.39).

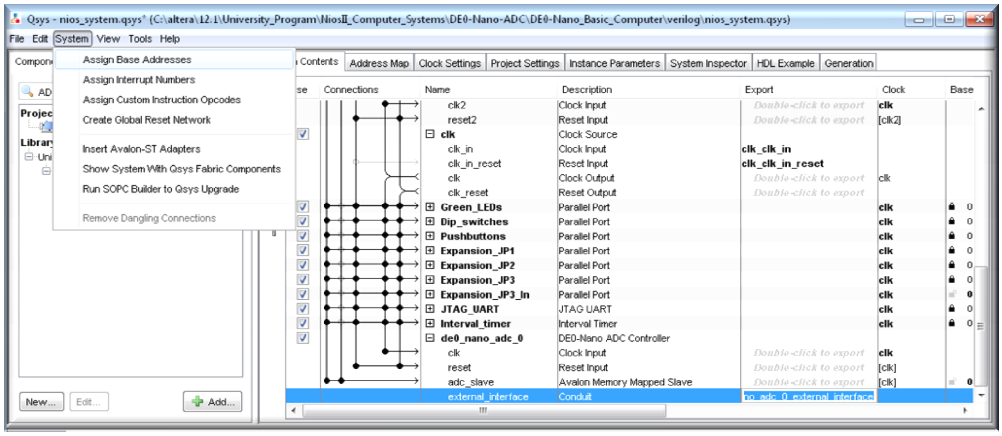


Figura 3. 39: Eliminación de errores en la configuración de la librería DE0-Nano-ADC.
Fuente: El Autor.

Los errores se borraran y obtendremos una ventana libre de errores en el menú de “Description”. 0 Errors, 0 Warnings, tal como se muestra en la figura 3.40.

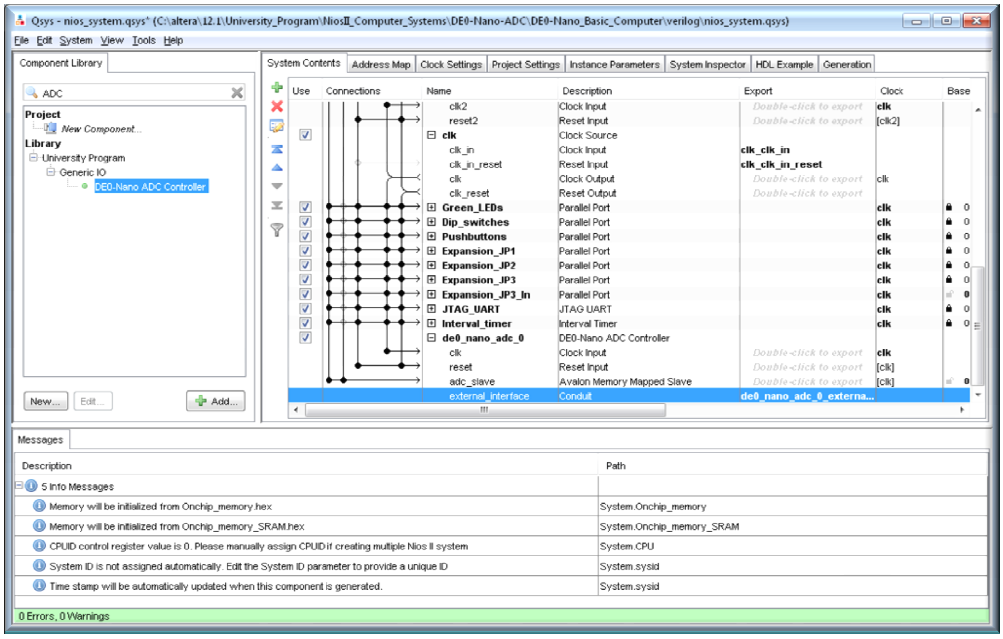


Figura 3. 40: Compilación sin errores de DE0-Nano-ADC.
Fuente: El Autor.

Después, debemos ir al menú de HDL Example, para copiar en código de la librería correspondiente (ver figura 3.41).

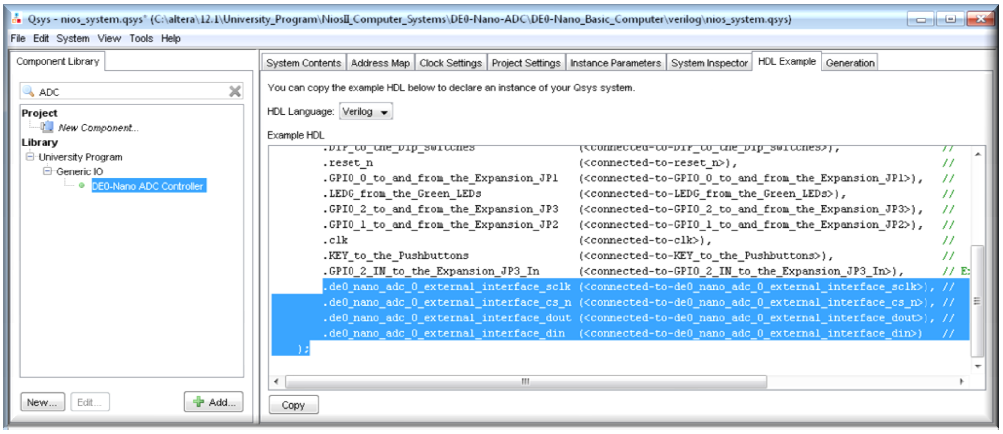


Figura 3. 41: Copia del código correspondiente de la librería DE0-Nano-ADC.
Fuente: El Autor.

En el siguiente paso, se procede a generar la máquina cuyo procedimiento es el mismo de la práctica 1. De manera similar ocurre con la plataforma NIOS II

de Eclipse, pero la programación no es la misma. A continuación se escribe la programación para el conversor A/D.

PROGRAMACION EN NIOS II DE ECLIPSE.

Código:

```
/*
 *  Creado: 10/09/2013
 */
/*DescripciónConversiónAnalógica Digital
#include<io.h>
#include<stdio.h>
#include<system.h>
#include"altera_up_avalon_parallel_port.h"
#include"altera_up_avalon_de0_nano_adc.h"

//funciónpararetardo
voiddelay(int tiempo)
{
        int contador=0;
        while(contador!=tiempo)
                contador++;
}

intmain (void)
{
        alt_up_parallel_port_dev * led; //Punteroparaleds
        alt_up_de0_nano_adc_dev * adc;    //Punteropara ADC

        // Variable paralecturadedatos
        unsignedint data0;
        unsignedint channel0;
```

```
        //Inicializa variables
        data0 = 0;
        channel0=0;

        //Asignacióndepunteroparabancodeleds
        led =alt_up_parallel_port_open_dev ("/dev/Green_LEDs");

        //Asignacióndepunteroparaconvertidoranalógico digital
        adc = alt_up_de0_nano_adc_open_dev ("/dev/de0_nano_adc_0");

        //Imprime mensajeenconsola
        printf("ConversiónAnalógica Digital");
        while (led!=NULL&&adc!=NULL)
        {
                // LECTURA ADC 0
                alt_up_de0_nano_adc_update (adc);

                //Lectura valor desde sensor
                data0 = alt_up_de0_nano_adc_read (adc, 0);

                //Desplazamientode 4 bits
                data0 = (data0/16);

        //Imprimir  valor enconsola
                printf("Valor = %d \n",data0);

        //Muestra valor enBancodeLeds
                alt_up_parallel_port_write_data (led, data0);
                delay(50000);
        }
return 0;
```

Después copiamos el código correspondiente en la ventana de desarrollo como se muestra en la figura 3.42.

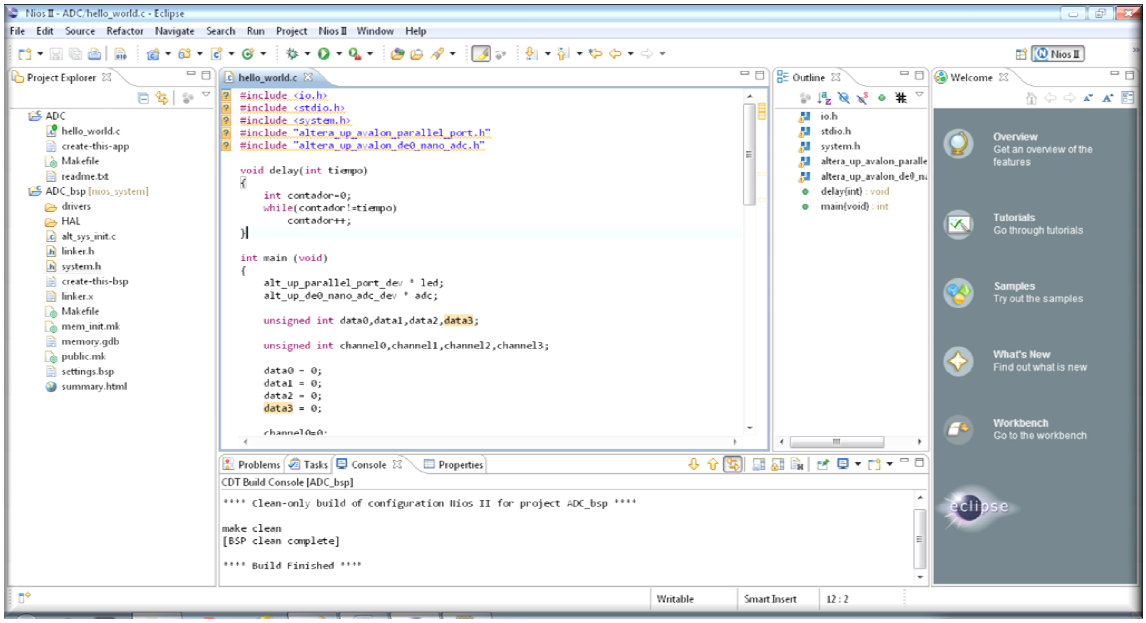


Figura 3. 42: Código del DE0-Nano-ADC en NIOS II.
Fuente: El Autor.

CONSTRUCCION DEL PROYECTO.

Es necesario el archivo de extensión .elf para su compilación y se deben realizar los mismos pasos de la práctica 1: Project – BuildAll. Entonces aparecerá un archivo de extensión .elf, y un mensaje que se muestra en lafigura 3.43 en el menú de console "BuildFinished".

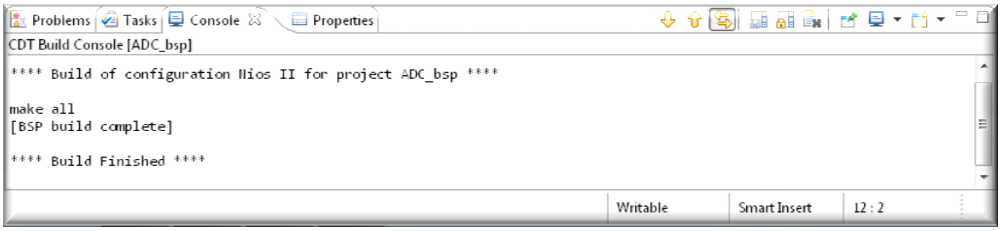


Figura 3. 43: Menú del BuildConsole del DE0-Nano-ADC.
Fuente: El Autor.

COMPILACION DEL CODIGO.

Para su respectiva compilación nos dirigimos al RunConfiguration para realizar los cambios respectivos, tal como se ilustra en la figura 3.44.

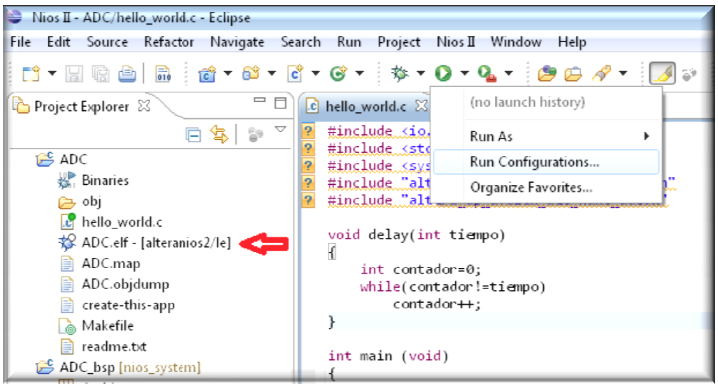


Figura 3. 44: Código del DE0-Nano-ADC en NIOS II.
Fuente: El Autor.

De la figura 3.44, podemos observar la presencia del archivo ADC.elf con la flecha roja, la misma que denota que el proyecto se ha compilado exitosamente. Posterior a esto, aparece una ventana y nos dirigimos a seleccionar la opción de Nios II Hardware (véase figura 3.45).

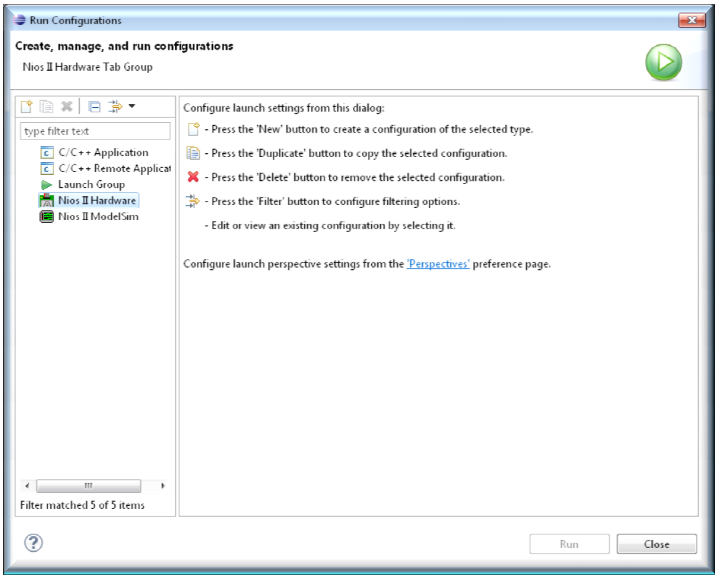


Figura 3. 45: Configuración de ejecución del NIOS II.
Fuente: El Autor.

Entonces se mostrará una ventana de nombre RunConfiguration, como se muestra en la figura 3.46.

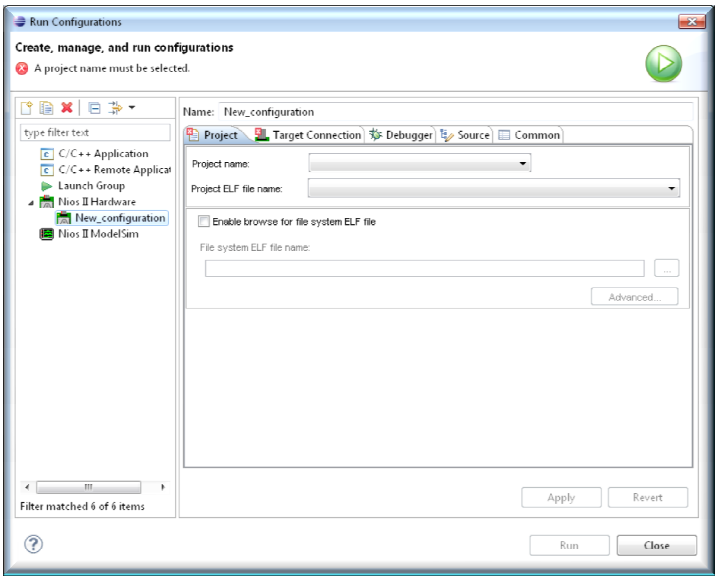


Figura 3. 46: RunConfiguration del DE0-Nano-ADC.
Fuente: El Autor.

En el menú Project – Project name – Seleccionamos el nombre del proyecto, tal como se ilustra en la figura 3.47.

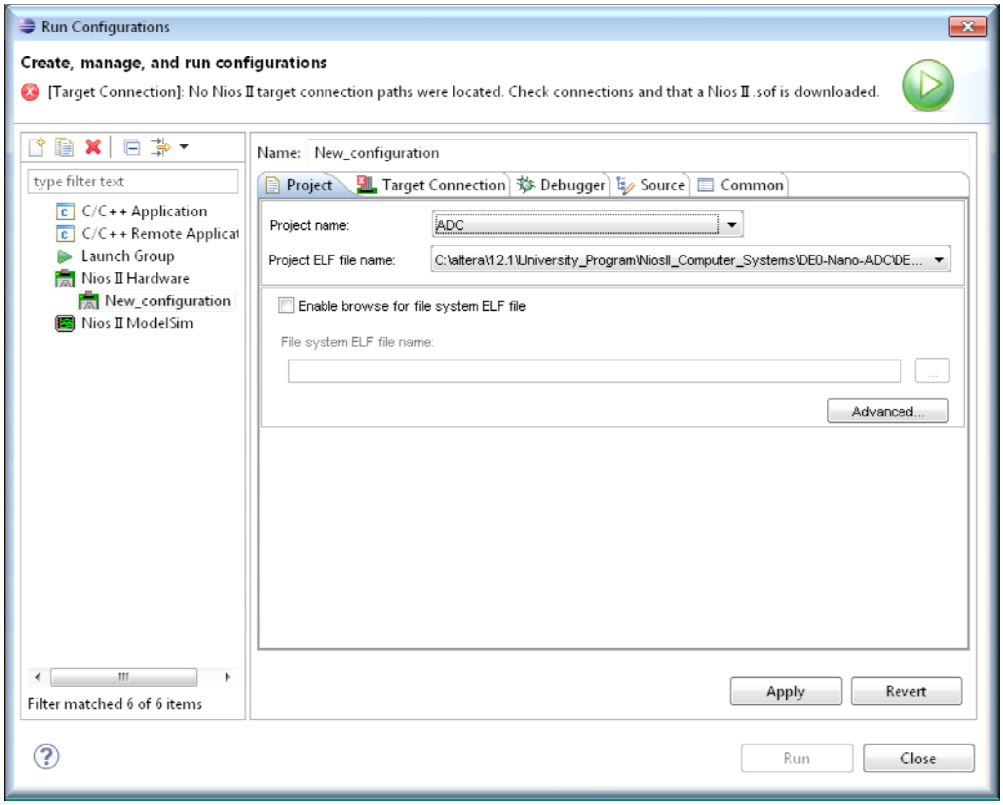


Figura 3. 47: Creación y configuración de la máquina del DE0-Nano-ADC.

Fuente: El Autor.

Para finalizar el presente capítulo, donde se expusieron dos prácticas mediante sistemas embebidos, los cuales integran tanto hardware y software para que funcione correctamente estas aplicaciones.

En los anexos 1 y 2, se muestran en detalle dos prácticas adicionales preparadas para el aprendizaje de los alumnos de materias como Sistemas Digitales II, Laboratorio de Digital, Microprocesadores y Diseño Electrónico Digital.

CAPÍTULO 4: CONCLUSIONES Y RECOMENDACIONES.

4.1. Conclusiones.

- Se ha cumplido con el primer objetivo específico, en la cual se describió el Estado del Arte o Fundamentación Teórica de Sistemas Embebidos, los cuales permiten integrar dispositivos electrónicos (Hardware) y plataformas de programación (Software).

- Se pudo comprobar que la mejor herramienta o plataforma de programación es QUARTUS II de Altera, que trabaja de manera conjunta con la tarjeta DE0-Nano Development and EducationBoard de Altera mismo.

- A través de las dos prácticas descritas en el capítulo 3, se pudo comprobar que el proceso de un sistema embebido, no es fácil de lograr, debido a la integración de varias herramientas, pero lo más importante fue el aprendizaje logrado y la idea de transmitir el conocimiento básico de manejar la tarjeta DE0-Nano.

4.2. Recomendaciones.

- Poner en funcionamiento las dos prácticas que se dejan en los Anexos 1 y 2, la misma que se encuentra explicada paso a paso, para que el usuario o estudiante, capte de manera inmediata.

- Si bien es cierto que se deja constancia de la entrega de los equipos utilizados en el presente trabajo de titulación, sugiero que la FETD compre tarjetas de desarrollo DE0-NANO para Sistemas Digitales Avanzados, y que puedan salir temas de investigación para propuestas de trabajo de graduación.

- Capacitar tanto a docentes como estudiantes en el uso de la tarjeta DE0-Nano, y su aplicación en los programas de estudios de materias como Digitales I, Digitales II, Laboratorio de Digitales, Sistemas Microprocesador.

REFERENCIAS BIBLIOGRÁFICAS

Acevedo Lara, C., & Rueda Blanco, R. (2010). *Implementación de LabView como Sistema Scada para la arquitectura de control SNAC PAC OPTO 22, mediante una aplicación OPC.* Bucaramanga: Universidad Pontificia Bolivariana.

Amaro Soriano, J. E. (2012). *Android: Programación de dispositivos móviles a través de ejemplos* (1era ed.). Barcelona: Marcombo.

Ciscar Martinez, V. A. (2010). *Diseño e Implementación de un Sistema Automático para la Caracterización Dinámica de probetas y elementos estructurales de construcción.* Valencia: Universidad Politécnica de Valencia.

Developers. (27 de 05 de 2013). *Developers Android*. Obtenido de http://developer.android.com/sdk/index.html

Developers1. (20 de 05 de 2013). *Developers Android*. Obtenido de http://developer.android.com/tools/sdk/ndk/index.html

Developers2. (21 de 05 de 2013). *Developer Android*. Obtenido de http://developer.android.com/tools/devices/index.html

Dogan, I. (2008). *Programación de Microcontroladores PIC.* Barcelona: Marcombo S.A.

Dogan, I. (2011). *PIC Basic Projects: 30 Projects using PIC BASIC and PIC BASIC PRO.*Burlington: Newnes.

Gargenta, M. (2011). *Learning Android.* Estados Unidos: O´Reilly.

Gironés, J. T. (2013). *El Gran Libro de Android* (3era ed.). Barcelona: Marcombo.

González Cayufilo, R., & Pradines Pino, R. (2007). *Análisis de Software para Desarrollo entorno Gráfico LabView y Propuesta de Implementación para Laboratorio en el Instituto de Electricidad y Electrónica en Universidad Austral de Chile.* Valdivia: Universidad Austral de Chile.

Iskandar Morine, R. J. (2013). *Estudio comparativo de alternativas y frameworks de programación, para el desarrollo de aplicaciones móviles en entorno Android.* Barcelona: Universidad Politécnica de Catalunya.

Lajara Vizcaíno, J. R., & Pelegrí Sebastía, J. (2011). *LabView: Entorno gráfico de programación* (Segunda ed.). Barcelona: Marcombo, S.A.

López Hernández, I., & Zuñiga Castro, D. (2009). *Protocolo para crear un sistema para reducir energía mediante el control de temperatura en casas habitación.* Juárez: Repositorio Universidad Autónoma de Ciudad de Juárez.

Melchor Hernández, N. (2009). *Tarjeta de Desarrollo para Microcontroladores PIC.* México D.F.: Repositorio Instituto Politécnico Nacional (IPN).

MIT. (25 de 05 de 2013). *AppInventor MIT.* Obtenido de http://appinventor.mit.edu/explore/sites/teach.appinventor.mit.edu/files/MIT%20App%20Inventor%20Development%20Overview_0.pdf

Reyes, C. A. (2008). *Microcontroladores PIC Programación en Basic.* Quito, Ecuador: RISPERGRAF.

Santamaría, J. (2009). *Herramientas Computacionales para Control Sesión LabView.* Bucaramanga: UPB - Facultad de Ingeniería Electrónica.

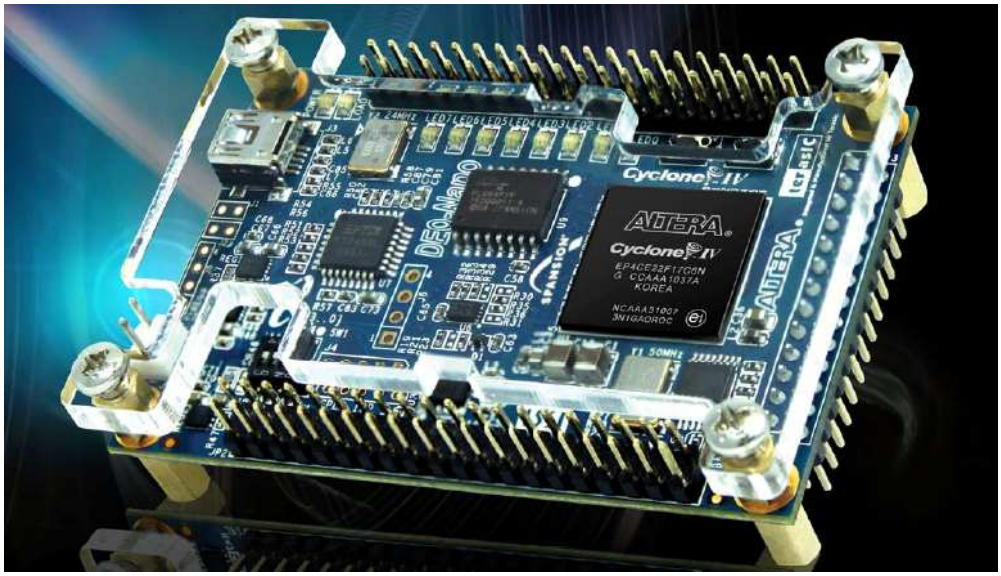
Terven Salinas, J. (17 de 06 de 2013). *Tervenet.* Obtenido de http://www.tervenet.com/itmaz/micros2012/01%20Introduccion.pdf

Valdivieso Noroña, D. (2013). *Construcción de un Controlador de temperatura ambiental y huedad del suelo de un invernadero de tomate riñón orgánico*

utilizando el Microcontrolador PIC 16F877A . Quito: Repositorio de la Escuela Politécnia Nacional (EPN).
Verle, M. (2010). *PIC Microcontrollers - Programming in Basic.*Belgrado: mikroElektronika.

ANEXO #1

Práctica #3: Comunicación Serial Uart con la Tarjeta DE0 Nano de Altera programada bajo la plataforma NIOS II de Eclipse.



Tarjeta DE0 Nano de Altera

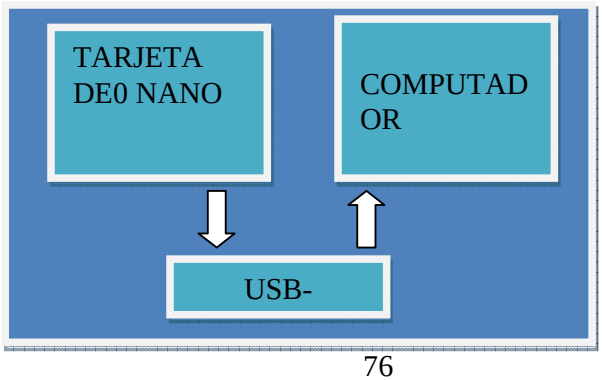
Objetivos:

- Configurar el componente de Comunicación Serial Uart en la Tarjeta DE0 Nano de Altera por medio de Qsys del Quartus II de Altera.
- Desarrollar un algoritmo en la plataforma Nios II de Eclipse para realizar la lectura y escritura de caracteres con la Tarjeta DE0 Nano de Altera a través de un dispositivo convertidor USB_SERIAL.

Descripción:

En la siguiente práctica se va a proceder a realizar un configuración desde el Qsys del Quartus II de Altera para realizar la Comunicación Serial Uart y realizar las pruebas mediante el software de comunicación serial AccessPort.

Diagrama de bloques:

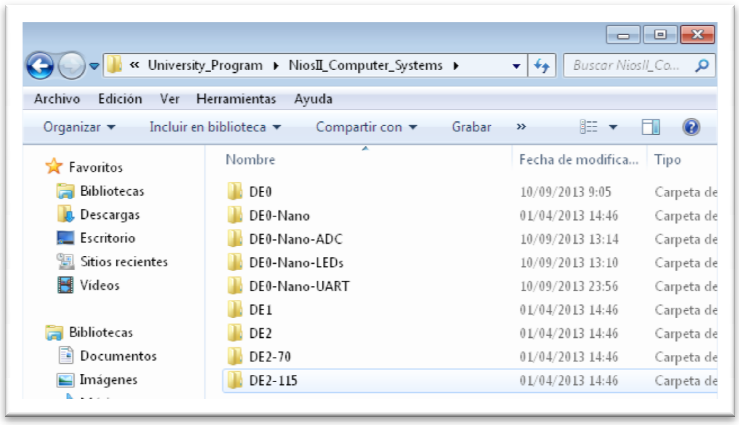


Desarrollo:

La práctica se desarrolla inicialmente haciendo una copia del archivo con el siguiente directorio:

En el cual se realiza una copia del archivo cuyo nombre es “DE0-Nano”, como se muestra en la imagen.

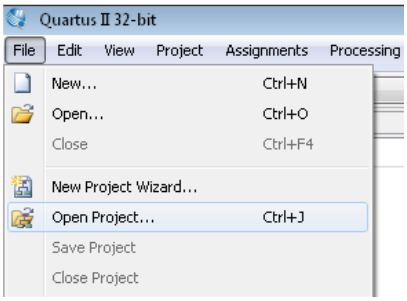
Una vez realizado este procedimiento, se procede a cambiarle el nombre como se muestra en la siguiente figura “DE0-Nano-UART”.



QUARTUS II DE ALTERA

Abrir el Quartus II de Altera:

Se realiza los siguientes procedimientos para abrir el proyecto ubicado en la carpeta "DE0-Nano-UART".



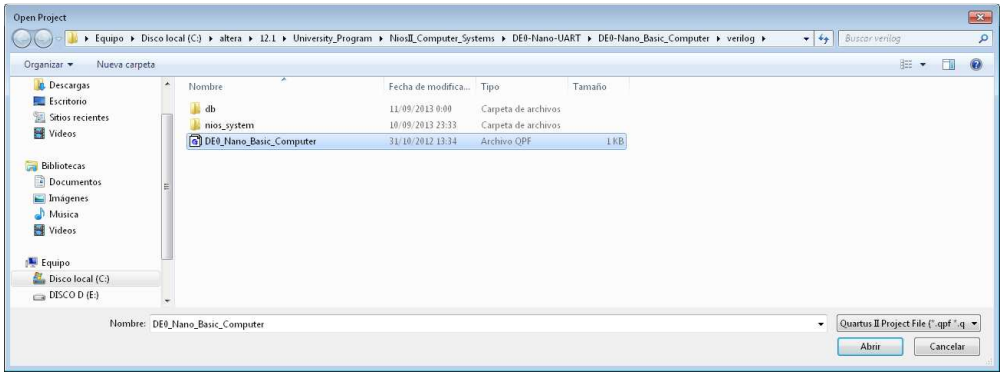
File

Open Project…

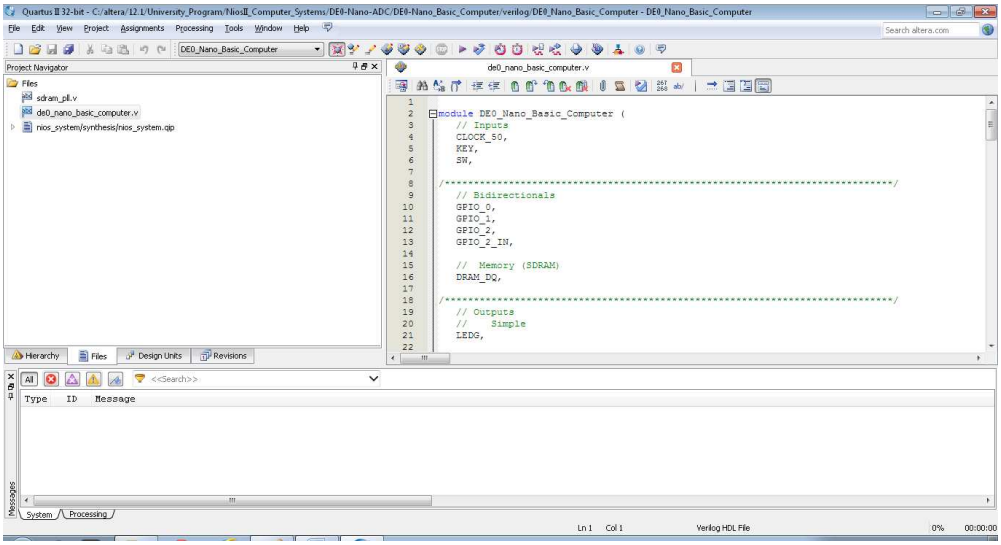
Se selecciona el archivo “DE0_Nano_Basic_Computer” ubicado en el siguiente directorio:

C:\altera\12.1\University_Program\NiosII_Computer_Systems\DE0-Nano-UART\DE0-Nano_Basic_Computer\verilog.

Click en Abrir.



Ventana principal de Quartus II de Altera con archivo listo para su procesamiento.



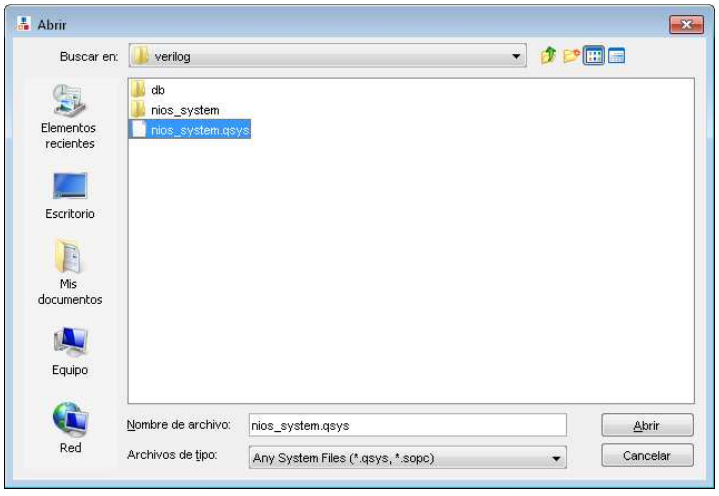
QSYS DEL QUARTUS II DE ALTERA

Editor de elementos en Qsys:

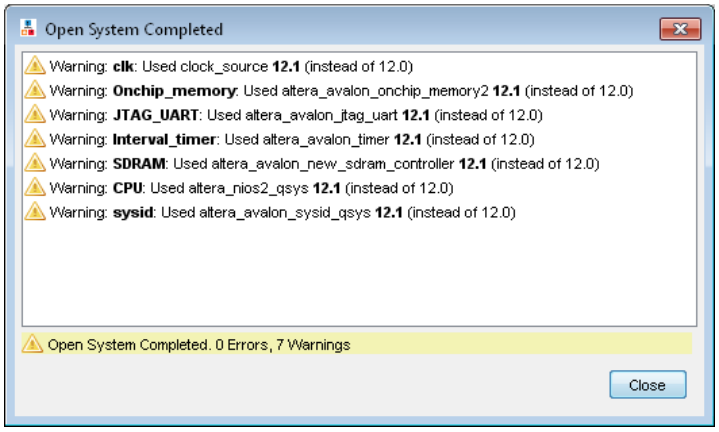
Se procede a abrir la ventana de Qsys haciendo clic en el icono que se muestra en la figura anterior.

La ventana que se abre nos indica que se debe de abrir el archivo de extensión .qsys ubicado en la carpeta de la práctica “DE0-Nano-UART” con directorio: C:\altera\12.1\University_Program\NiosII_Computer_Systems\DE0-Nano-UART\DE0-Nano_Basic_Computer\verilog.

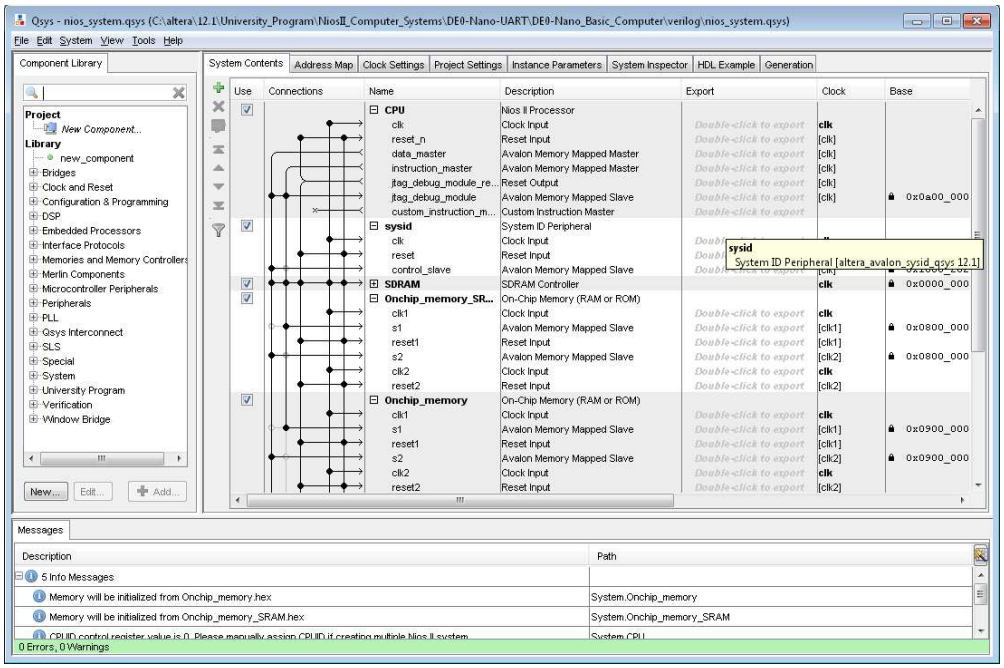
Se selecciona el archivo: nios_system.qsys y se hace clic en abrir.



Se mostrará un nueva ventana y hacemos clic en close.



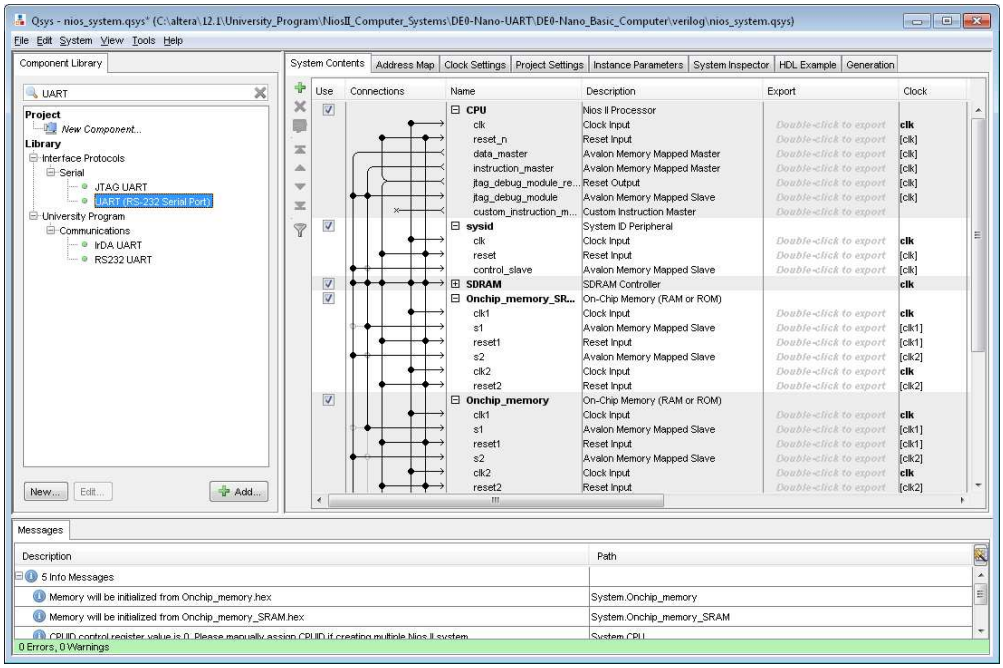
VENTANA PRINCIPAL DE QSYS.



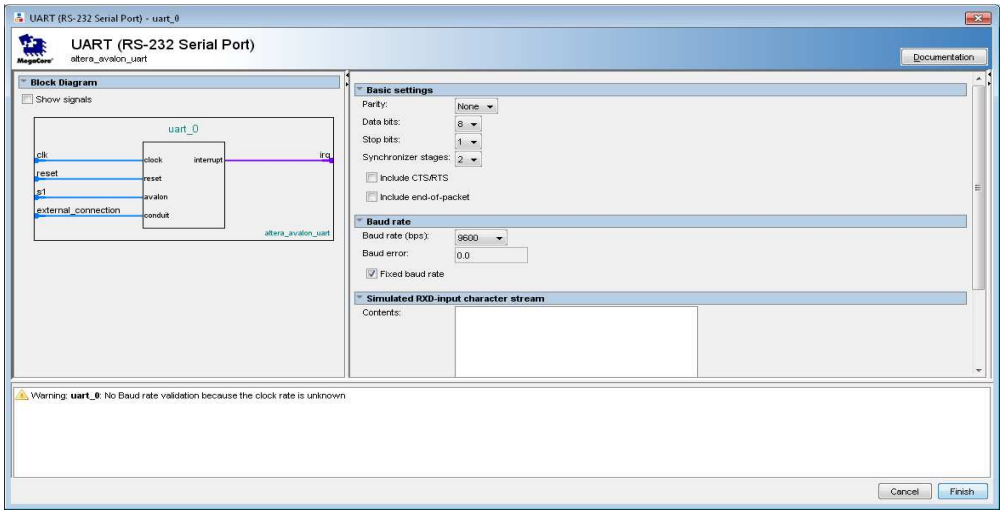
Por medio de la barra de desplazamiento vertical, se puede apreciar todos los componentes pertenecientes a la tarjeta DE0 Nano.

COMPONENETES DE QSYS

Para hacer uso de la Comunicación Serial Uart, nos dirigimos al menú de Component Library, para incorporar una librería Comunicación Serial Uart. Se digita el nombre del componente y el buscador automáticamente nos presentará una librería con el nombre de DE0-Nano ADC Controller, como se muestra en la siguiente figura.



Seguidamente se muestra una ventana después de hacer doble clic en dicha librería "UART (RS – 232 Serial Port)". Seguidamente de muestra una figura para el cual se ha configurado Baud rate (bps) = 9600.

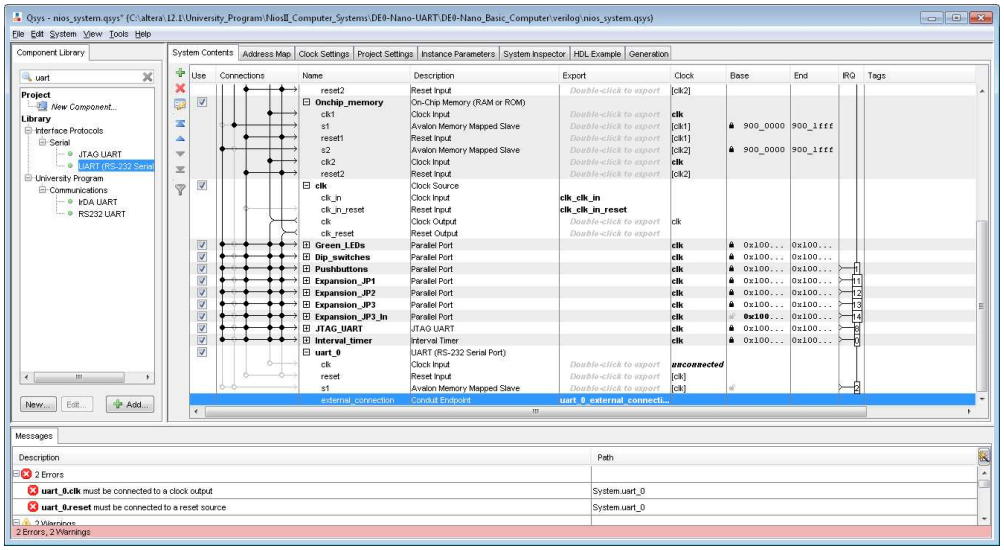


Esta ventana permite realizar la configuración de este protocolo de comunicación.

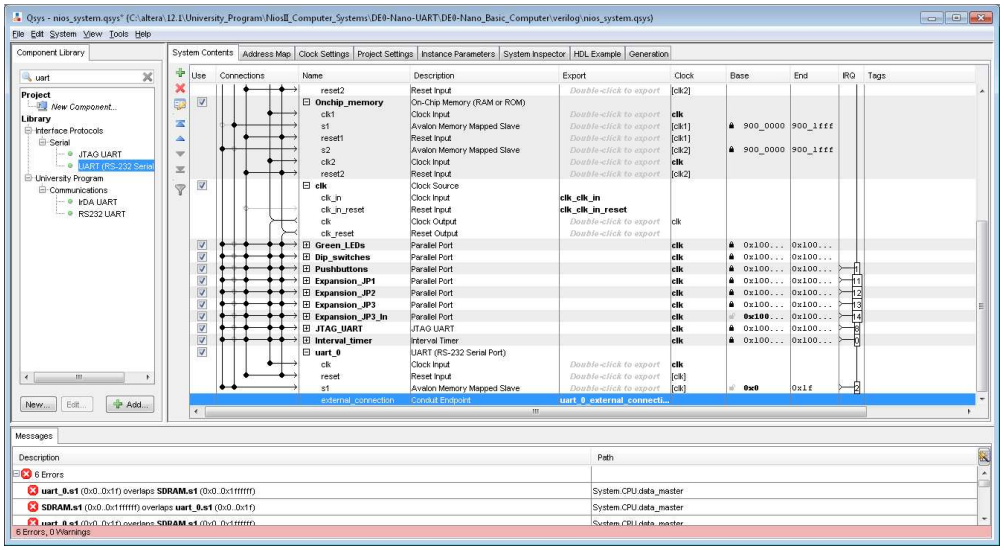
Hacer clic en Finish.

En la siguiente figura se puede apreciar que al añadir una nueva librería, aparecen errores descritos en la ventana inferior de “Description”.

También se puede observar que la librería de UART (RS – 232 Serial Port) ha sido incorporada con éxito.



Para configurar la nueva librería se debe empezar haciendo las conexiones respectivas, así como se observa en la siguiente imagen.

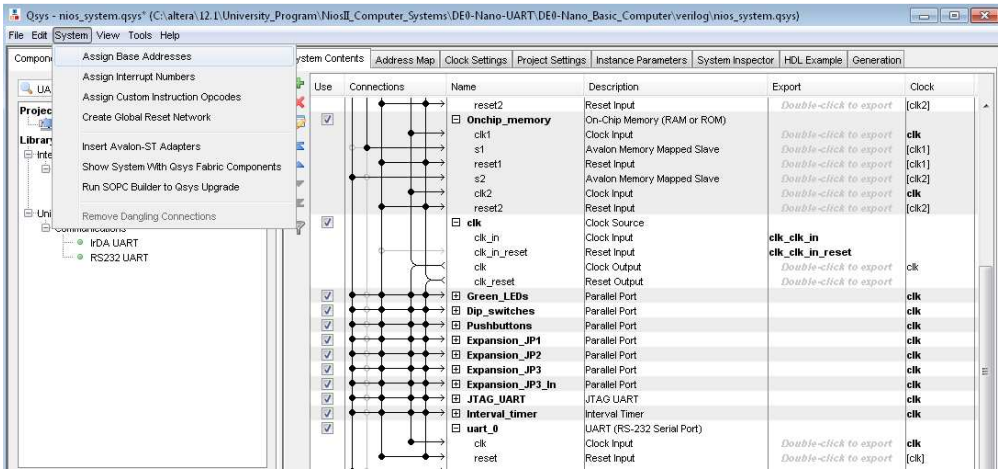


Una vez realizada las conexiones, es importante exportar la señal de External Interface, haciendo doble clic en el menú de Export.

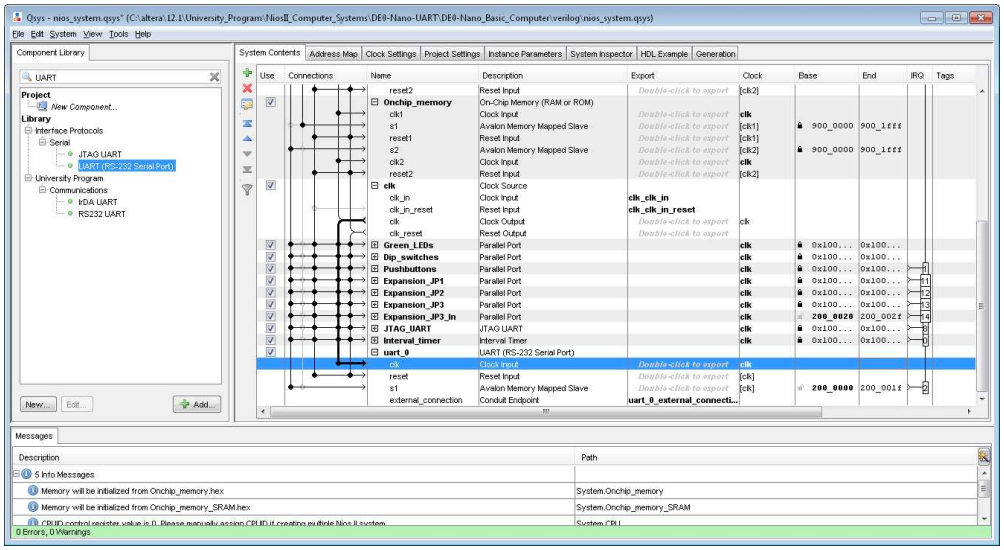
Una vez realizadas la conexión ahora se procede a eliminar los errores, realizando los siguientes pasos para asignar espacio de memoria.

Click en System.

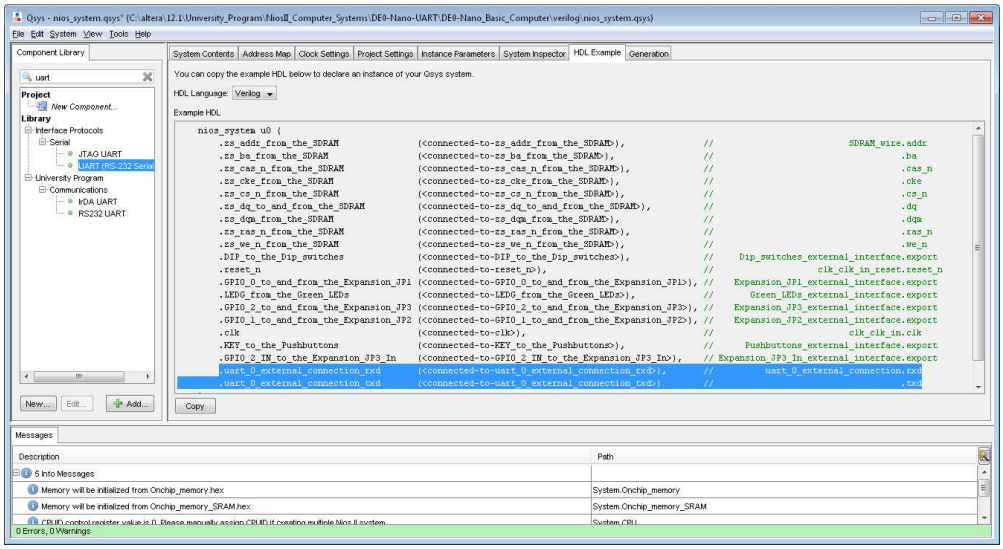
Click en Assign Base Addresses.



Los errores se borraran y obtendremos una ventana libre de errores en el menú de “Description”. 0 Errors, 0 Warnings.

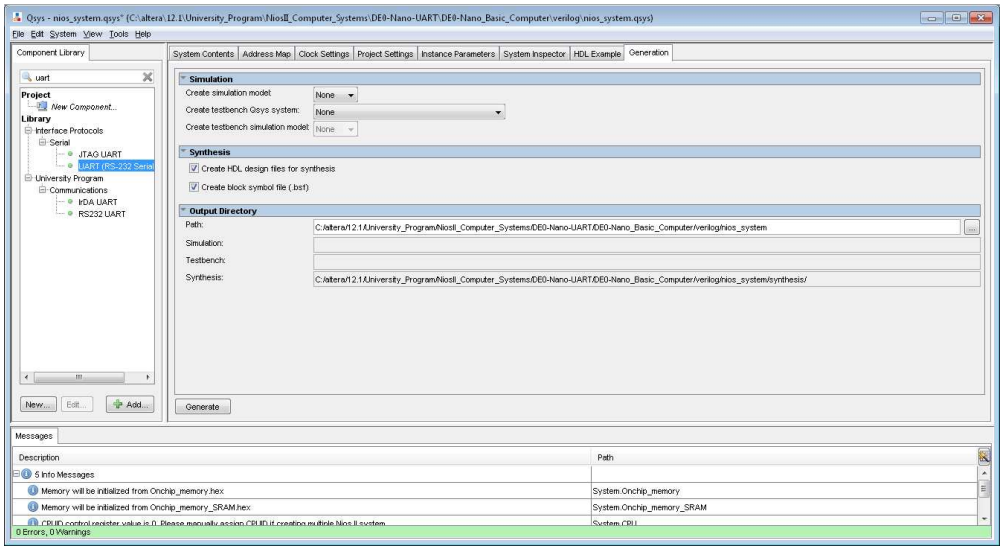


Como siguiente paso debemos ir al menú de HDL Example, para copiar en código de la librería correspondiente.

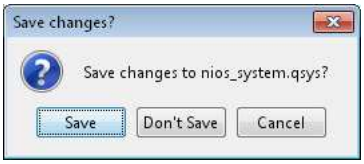


En el siguiente paso procedemos a generar la máquina:

- Seleccionamos el menú Generation.
- Hacer clic en Generate para generar el sistema.

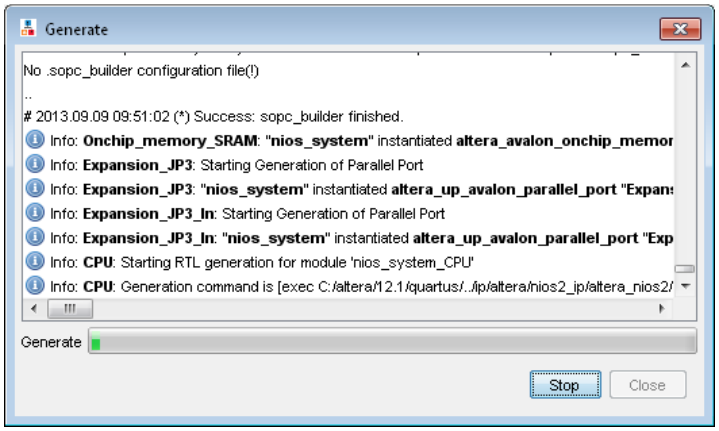


Una ventana para guardar los cambios aparece y aceptamos los cambios realizados.

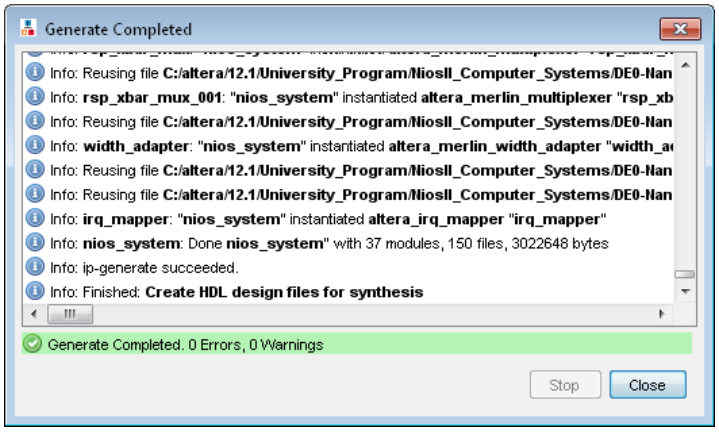


Esperamos hasta que se realice la compilación respectiva:

Proceso de construcción de la máquina se manifiesta en la figura siguiente.



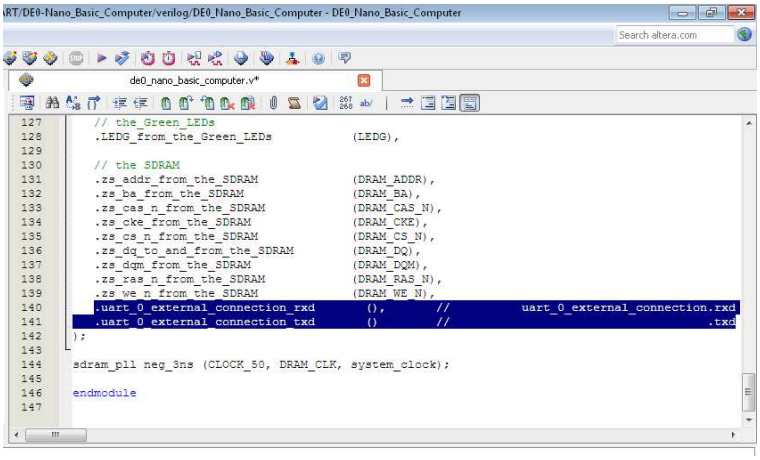
Una ventana nos manifiesta que la construcción se ha realizado con éxito “0 Errors”.



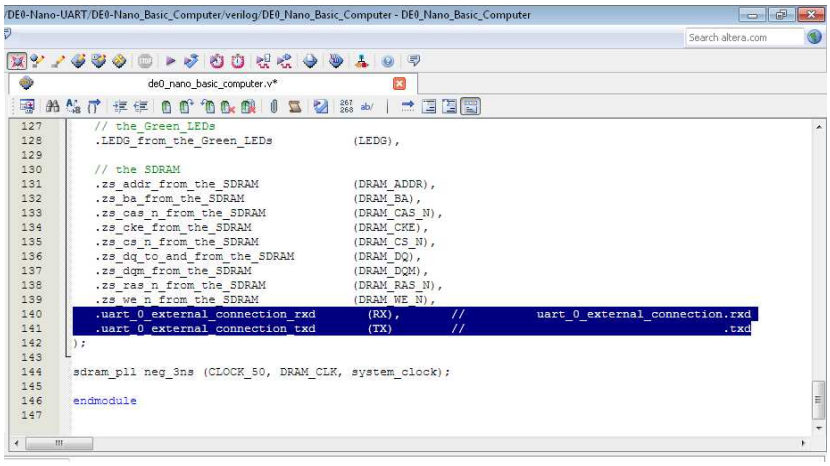
- Click en close.
- Cerrar la ventana de Qsys.

Nota: No olvidar de copiar las dos líneas de código generado por la librería de c Comunicaion Serial Uart.

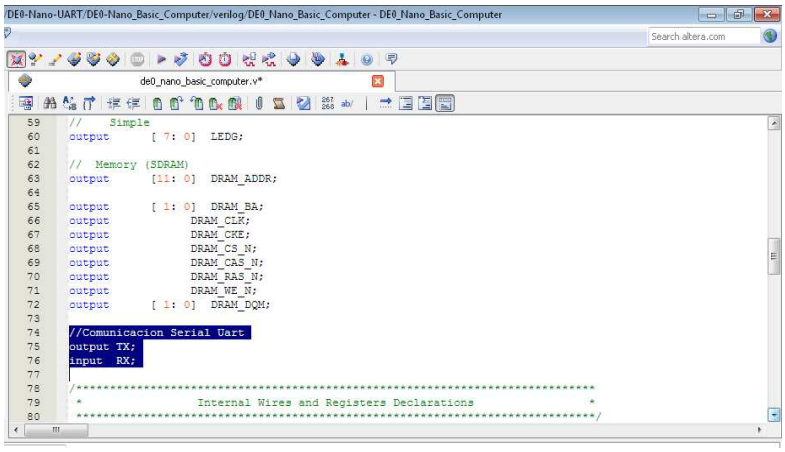
Una vez generada la máquina, se procede a pegar el código copiado de QSYS en la parte del código perteneciente a la máquina básica DE0 Nano.



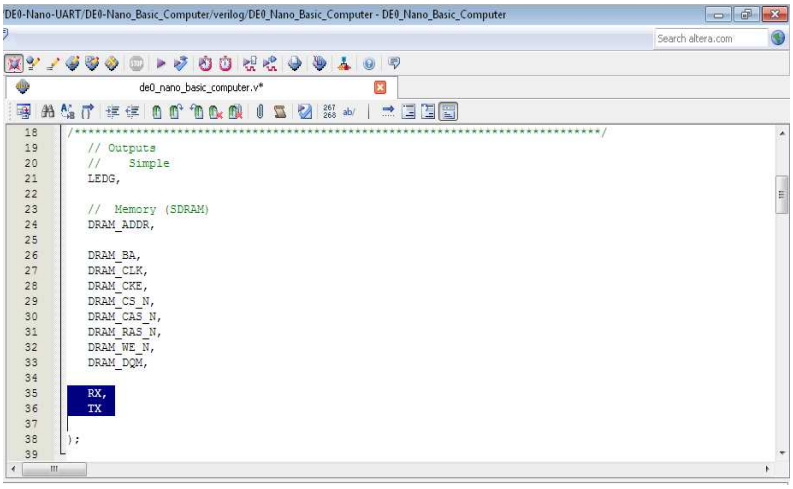
Posteriormente se necesita escribir un nombre para el cual van a ser las señales de control de la librería correspondiente a la Comunicaion Serial Uart, así como se muestra de figura.



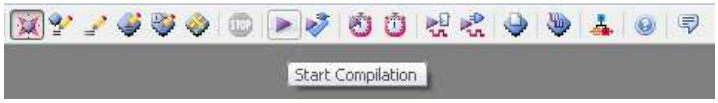
El nombre de cada señal seleccionada debe de ser declarada como entrada y salida respectivamente como se muestra en la siguiente figura.



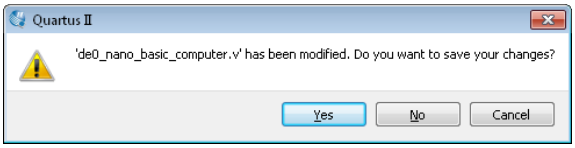
Así también debe de ser mencionada cada una de las señales de control en la sección de module DE0_Nano_Basic_Computer.



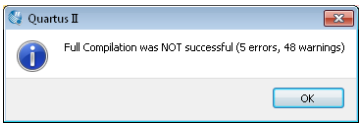
En la ventana de Quartus II, se hace clic en icono de Start Compilation.



Se guardan los cambios respectivos haciendo clic en YES.

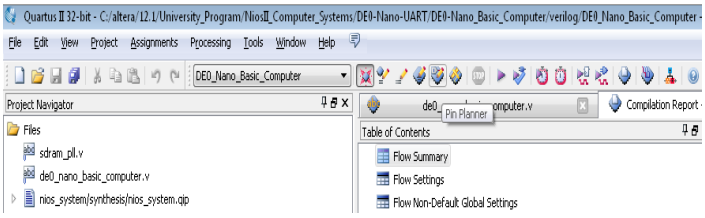


Una ventana similar a la siguiente figura aparece con el motivo de anunciar de que las nuevas señales generadas aun no han sido configuradas en el Pin Planner.

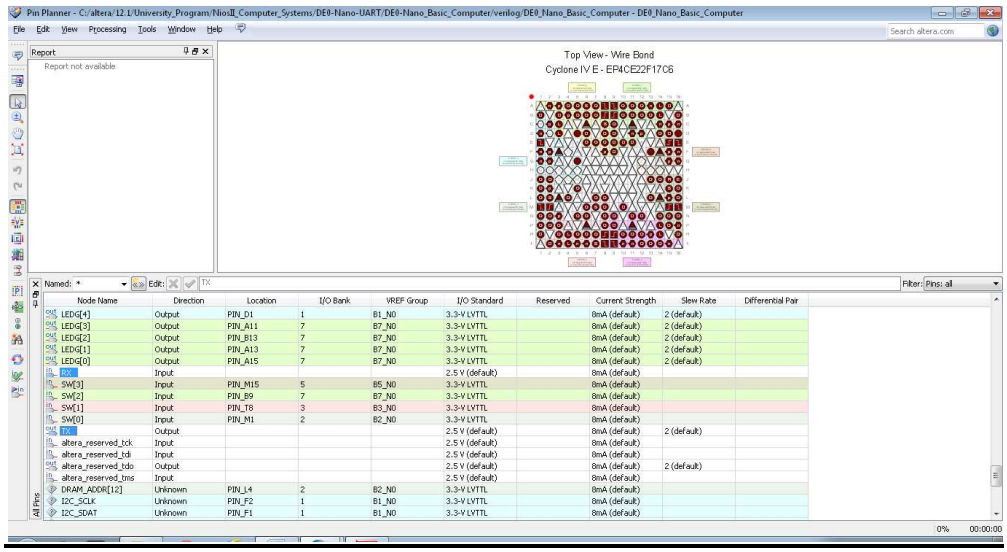


Clic Ok

Nos introducimos en el Pin Planner, haciendo clic en su icono correspondiente.



VENTANA PRICIPAL DE PIN PLANNER.

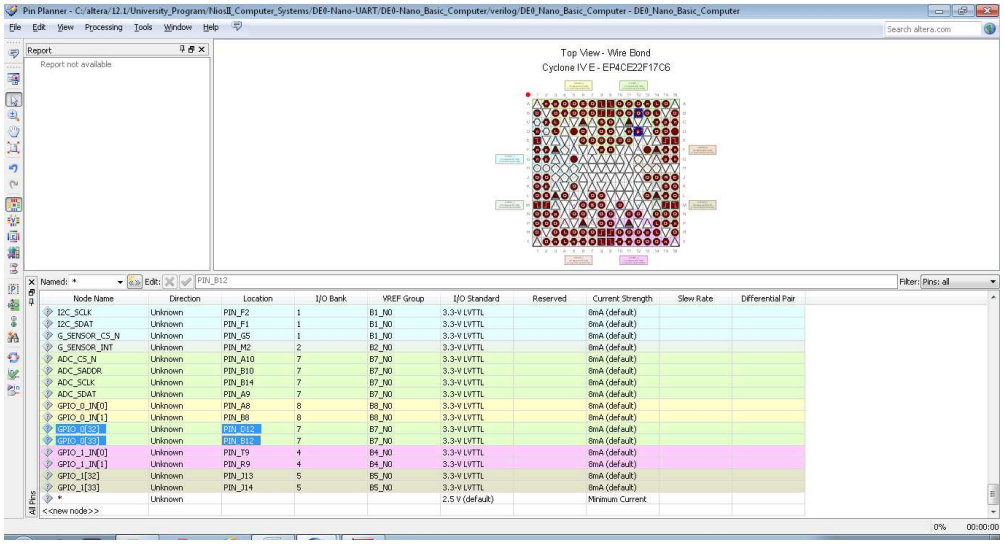


Se puede observar las dos señales RX y TX, y edita el menú de Location e I/O Bank, para que finalmente quede configurado de la siguiente manera t ubicado en los siguientes pines de la Tarjeta DE0 Nano de Altera.

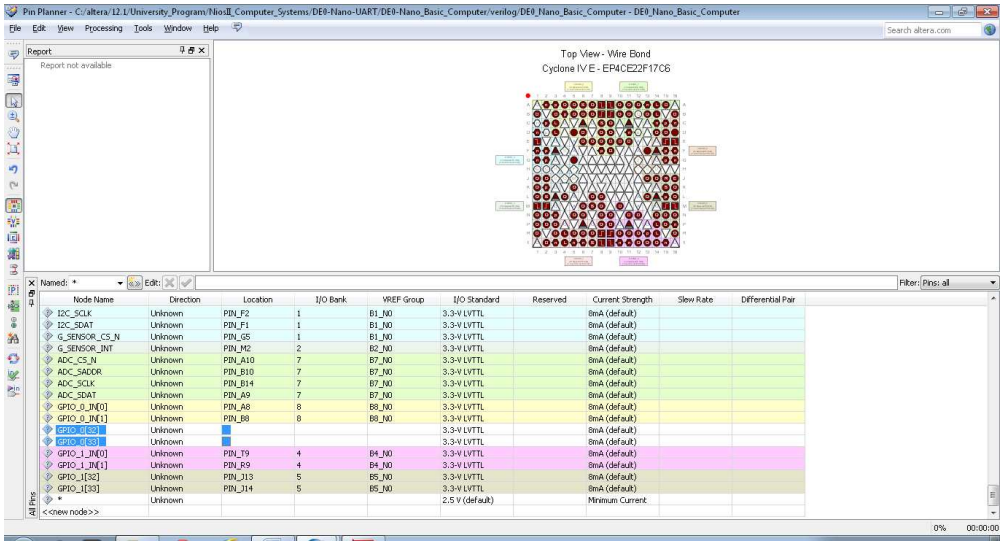
Las señal de transmisión TX ha sido asignada para este caso en el PIN_D12 y la señal de recepción ha sido asignada en el PIN_B12, correspondientes a los pin de GPIO_032 y GPIO_033 de la tarjeta.

Nota: El PIN_D12 y PIN_B12 están ocupados originalmente, por el cual es necesario ubicarlos en el Pin Planner y eliminar su asignación de la siguiente manera.

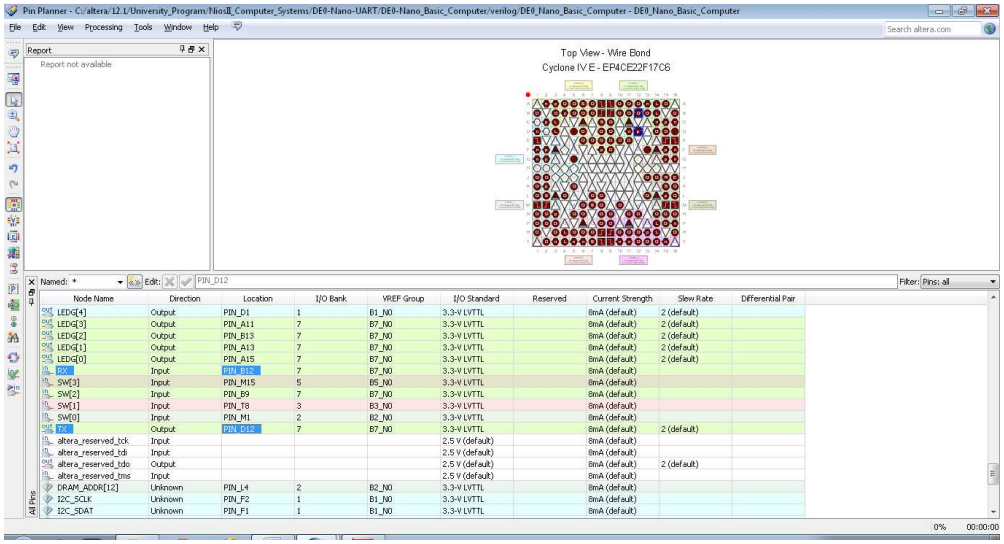
Ubicación de los pines PIN_D12 y PIN_B12.



Eliminación de de su asignación original PIN_D12 y PIN_B12 para reincorporar las señales de RX y TX.

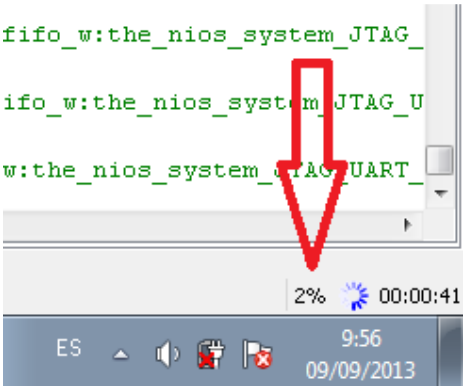


Asignacion de señales de control RX y TX.

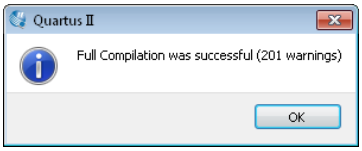


En la ventana de Quartus II, se hace clic en icono de Start Compilation nuevamente.

Se toma un tiempo en compilar la máquina en Quartus II, hasta que su valor llegue a un 100% como se indica en la figura.

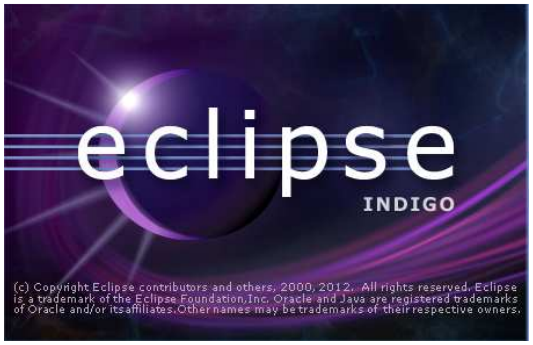


Una ventana aparece y nos indica que no existen errores durante su compilación.

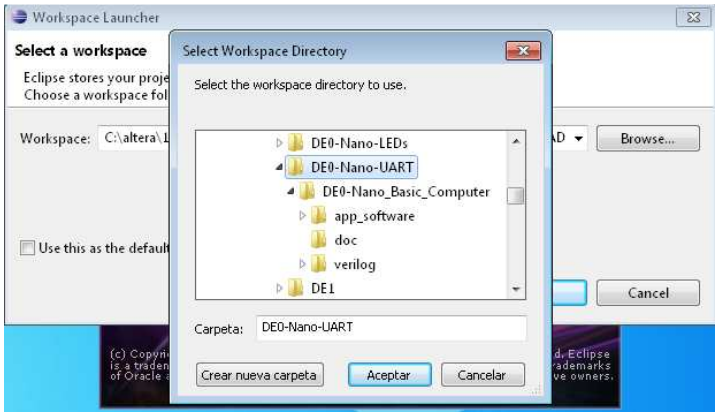


Click en OK

PLATAFORMA NIOS II DE ECLIPSE.



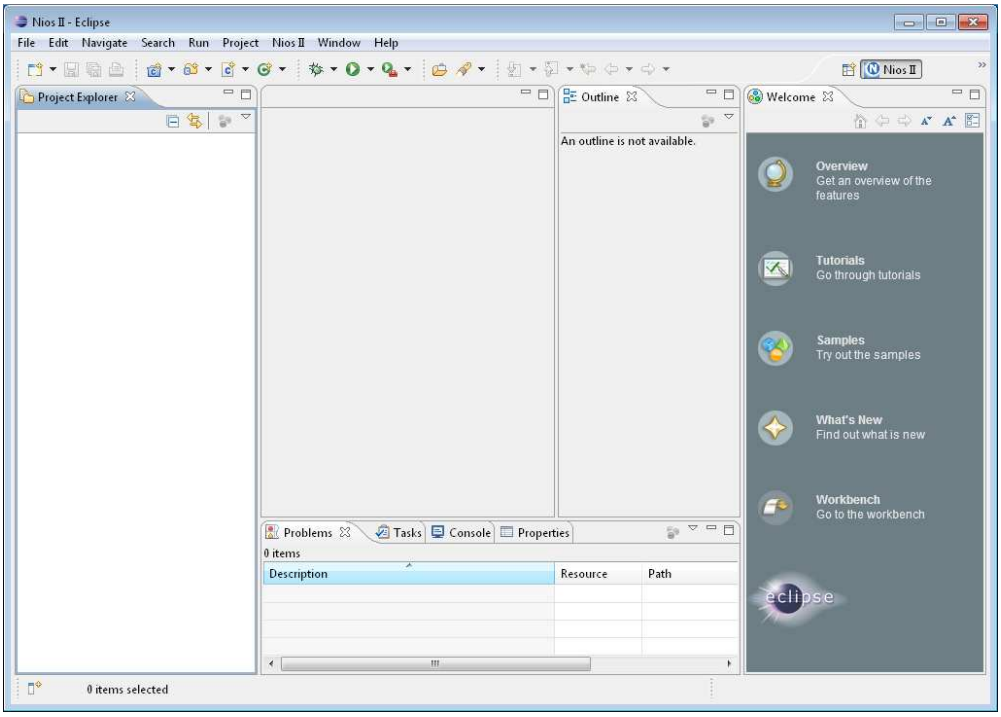
Se selleciona un espacio de trabajo, en este caso se decidió hacerlo dentro de la carpeta de ubicación de la practica Comunicación Serial Uart.



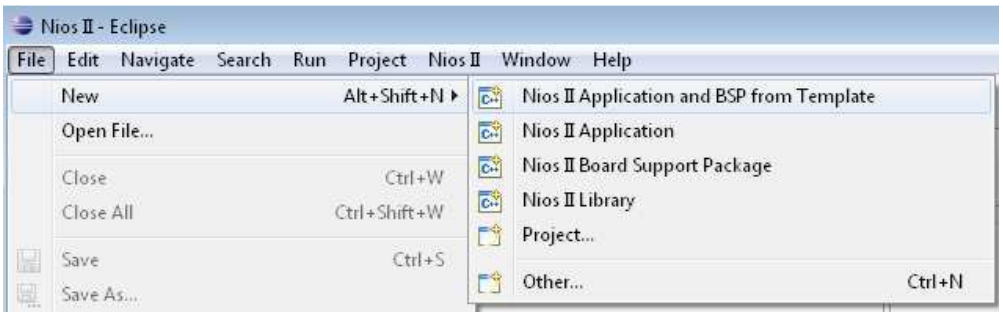
Buscamos la dirección de la carpeta de nuestro archivo.

Click en aceptar.

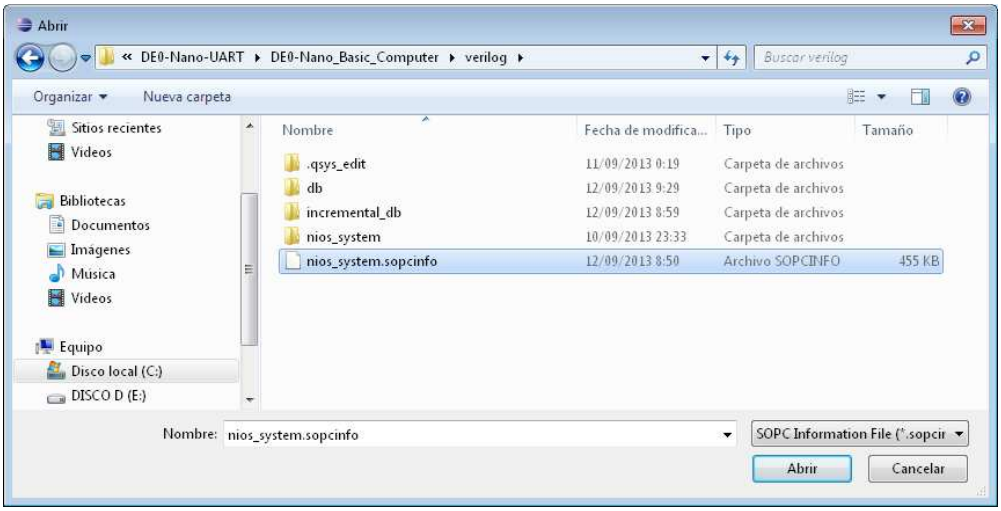
VENTANA PRICIPAL DE NIOS II DE ECLIPSE.



Se procede a crear un nuevo proyecto realizando los siguientes pasos:

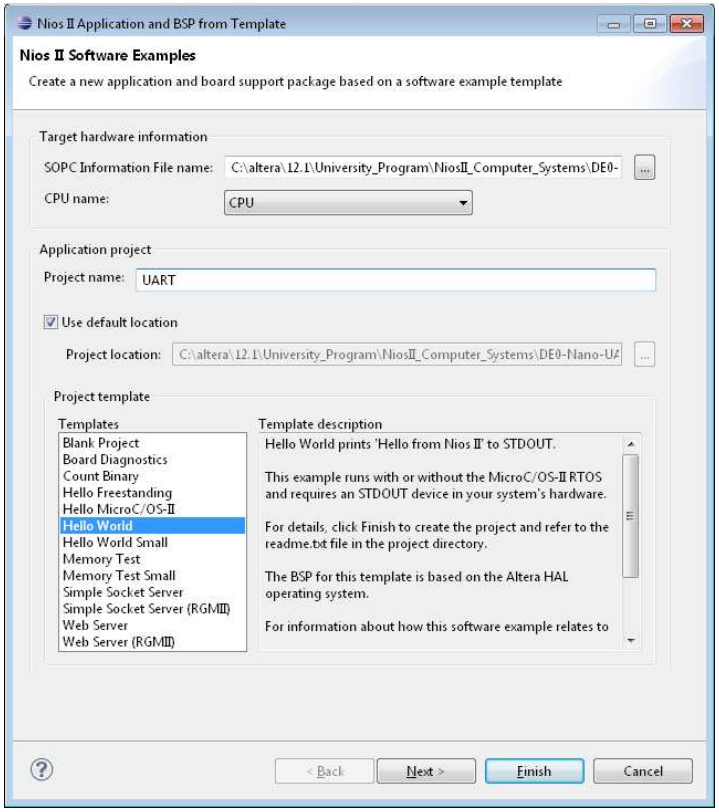


File - New – Nios II Application And BSP from Template.



Se selecciona el archivo nios_system.sopcinfo haciendo clic en SOPC Information File Name.

El directorio de establece y muestra nuestra maquina lista para ser utilizada en el menú de CPU name:



Escribimos un nombre en el menú de Project name:

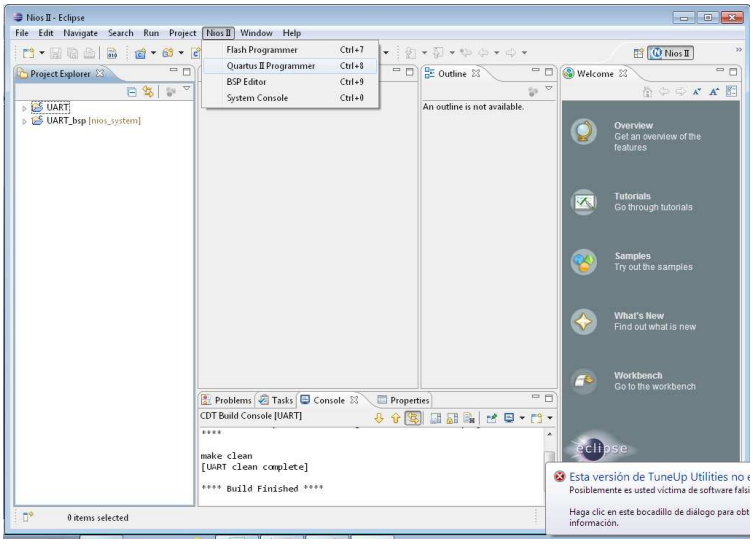
Por último hacemos Click en finish.

PROGRAMACION DE LA FPGA

Nota: Conectar la Tarjeta DE0 Nano al computador.

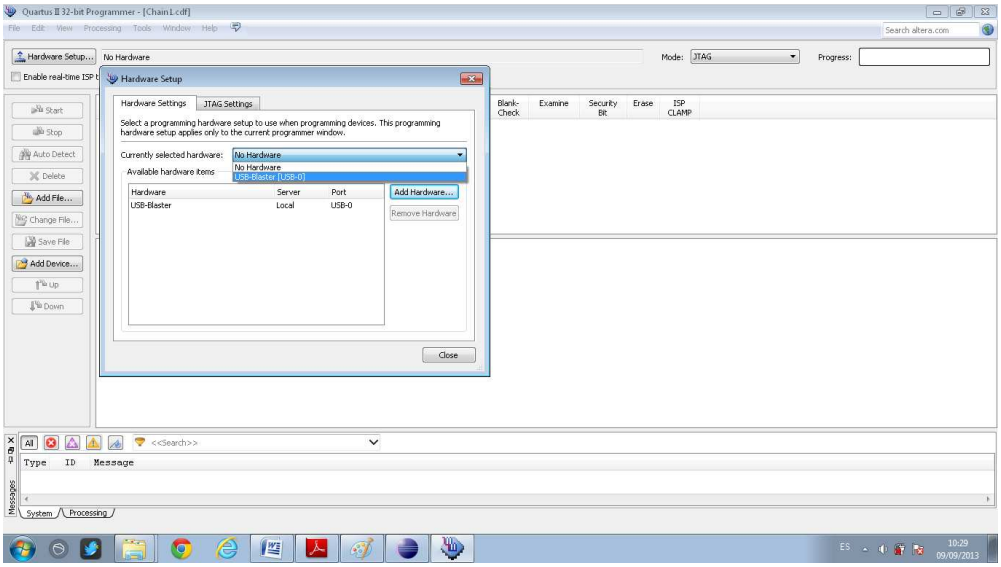
Antes de realizar código alguno, se debe realizar la programación de la maquina que se ha construido haciendo los siguientes pasos:

Menú Nios II – Quartus II Programmer.

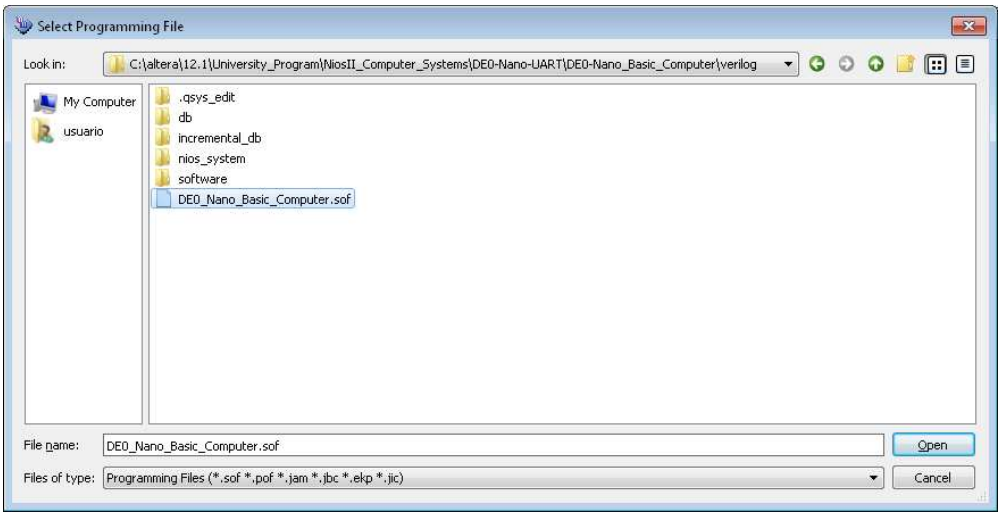


La ventana del programador del Quartus II aparece y realizamos los siguientes pasos.

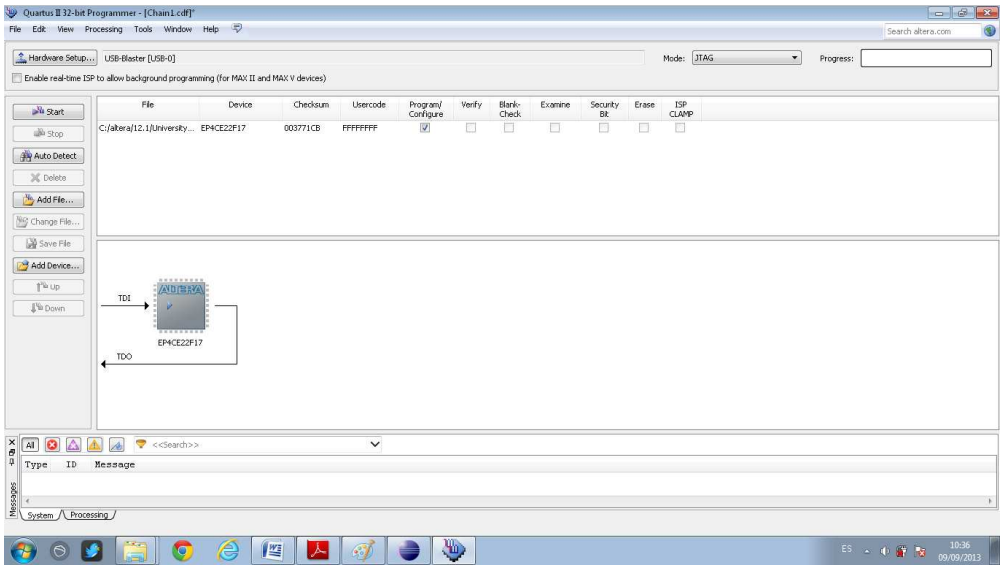
Hardware Setup – Currently Selected Harware – USB Blaster [USB 0] – click en close.



Una vez más nos ubicamos en la ventana principal del Programador de Quartus II, ahora se procede a añadir el archivo DE0_Nano_Basic_Computer.sof y se hace clic en Open.

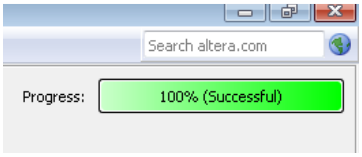


Una nueva vista nos presenta la ventana principal indicándonos que el archivo se ha cargado exitosamente.



Clic en START

En la ventana superior derecha se mostrara un indicador del estado de la programación de la tarjeta.



La tarjeta esta lista para ser programada en NIOS II.

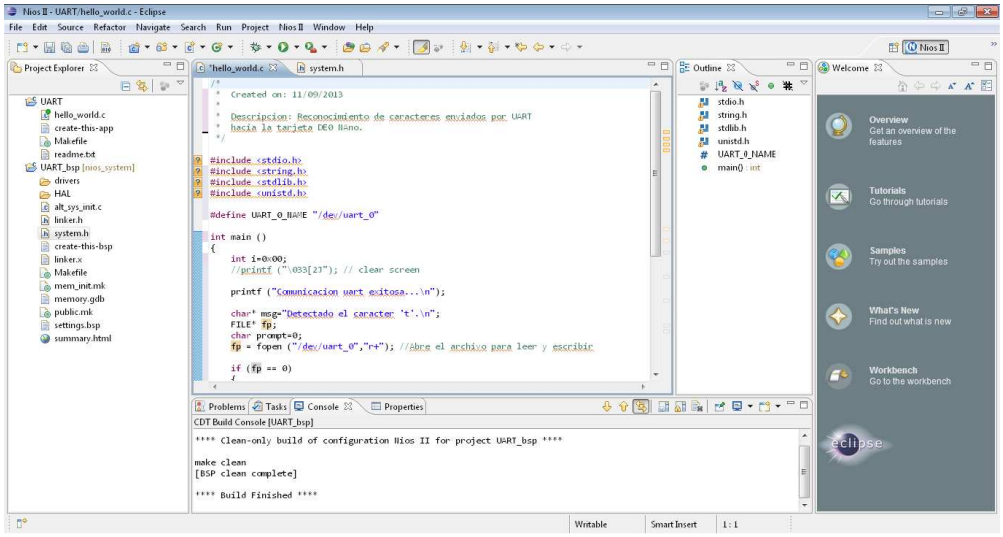
Codigo transmisión:

```
/*
 * principal.c
 *  Creado: 11/09/2013
 */
// Codigo para escribir un mensaje por comunicacion serial

#include <stdio.h>
#include "system.h"

#define UART_0_NAME "/dev/uart_0"

int main()
{
        FILE *fp_uart=0;

        printf("Hola desde el Nios II!\n");

        fp_uart = fopen("/dev/uart_0","r+");  // abre el puerto de comunicacion

        if(fp_uart==0)
                printf("\nError Opening %s\n\n",UART_0_NAME);   //mensaje de error si no se ha
                encontrado el puerto de comunicacion
        else
        {
                printf("Comunicacion Uart lista\n");
                fprintf(fp_uart,"Hola desde NIOS II ...");
        }
        fclose(fp_uart);

        return 0;
}
```

Código Recepción:

```
/*
 *
 * Creado: 11/09/2013
*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>

#define GREEN_LEDS_BASE 0x10000010
#define PUSHBUTTONS_BASE 0x10000050

int main ()
{
        int i;
        volatile int * fp_boton = (int *) PUSHBUTTONS_BASE;
        volatile int * fp_led = (int *) GREEN_LEDS_BASE;
        FILE *fp;
        char prompt=0;

        fp = fopen ("/dev/uart_0","r+"); //Abre el archivo para leer y escribir

        printf ("HOLA DESDE NIOS II...\n");
        *(fp_led)=0x00;
        if (fp == 0)

                printf ("Error...\n");
        else
        {
                while (prompt !='*')
                {
                        prompt = getc(fp);
                        printf ("%c",prompt);
                        if (prompt =='I')
                        {
                                *(fp_led)=0xFF;
                                printf (" CARACTER RECIBIDO\n");
                        }
                }
                fclose (fp);
        }

        return 0;
}
```

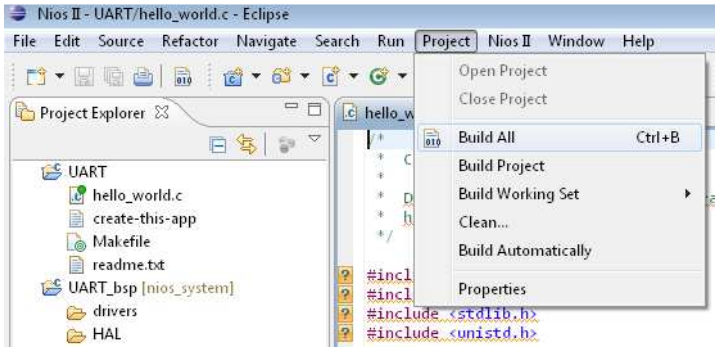
Copiamos el código correspondiente en la ventana de desarrollo como se muestra en la siguiente figura.



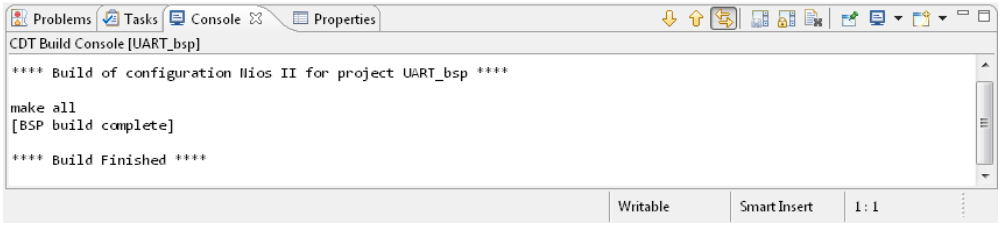
CONSTRUCCION DEL PROYECTO.

Es necesario el archivo de extensión .elf para su compilación y se deben realizar los siguientes pasos.

Project – Build All

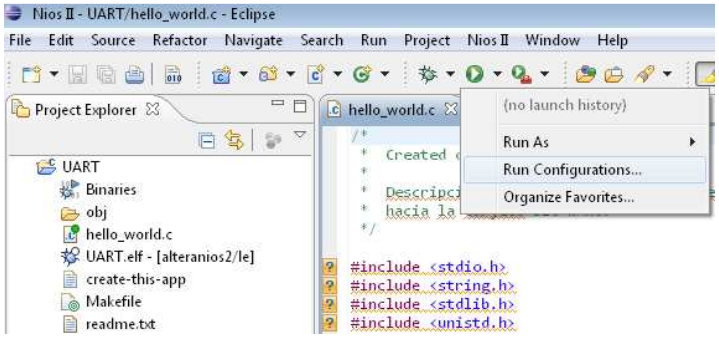


Entonces aparecerá un archivo de extensión .elf. y un mensaje como se muestra en la siguiente figura en el menú de console "Build Finished".



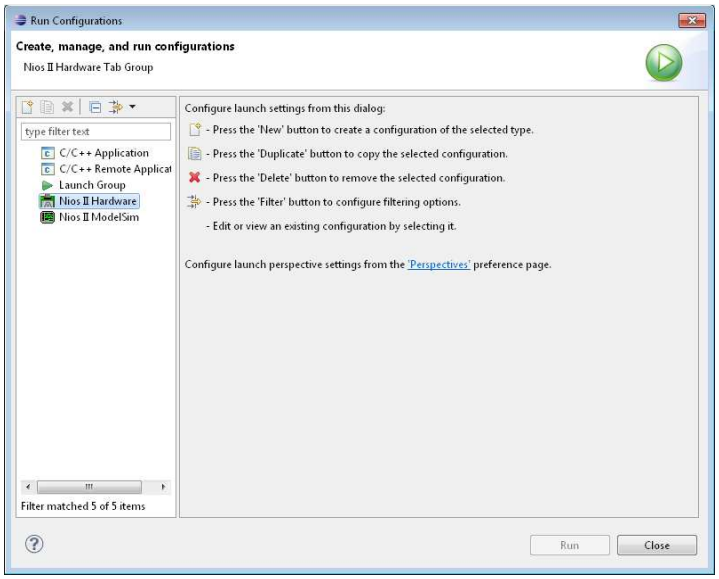
COMPILACION DEL CODIGO.

Para su respectiva compilación nos dirigimos al Run Configuration para realizar los cambios respectivos como se muestra en la siguiente figura.

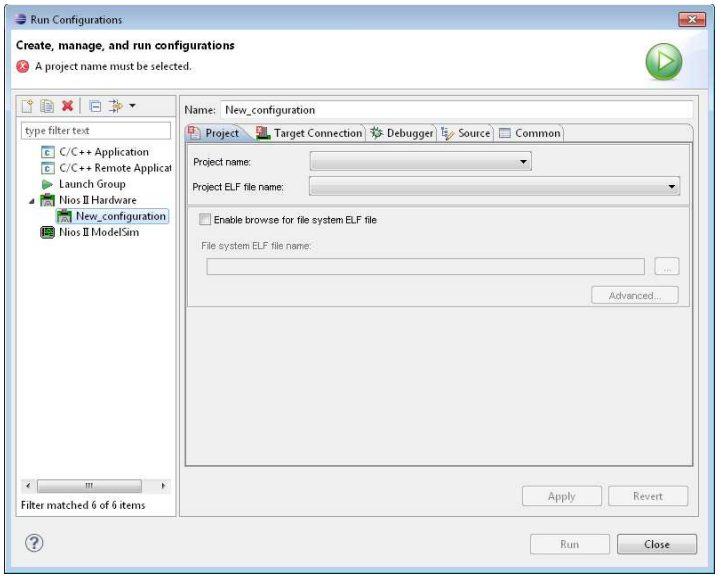


Nota: Observar la presencia del archivo UART.elf con la flecha roja, denota que el proyecto se ha compilado exitosamente.

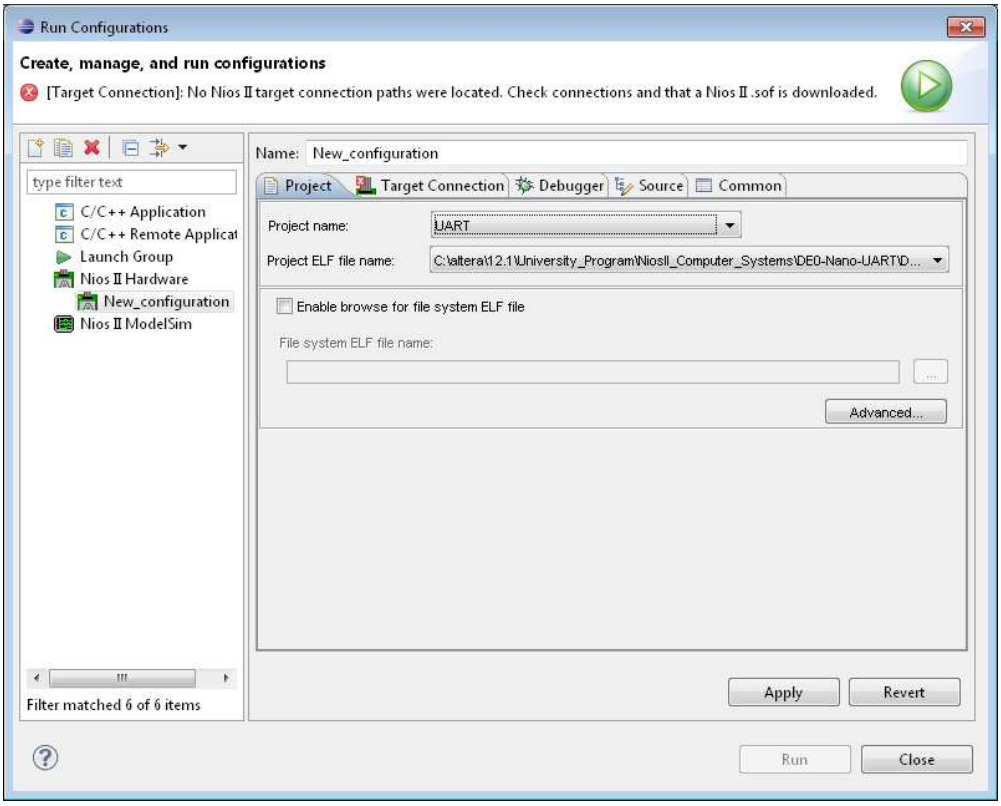
Una ventana aparece y nos dirigimos a seleccionar la opción de Nios II Hardware.



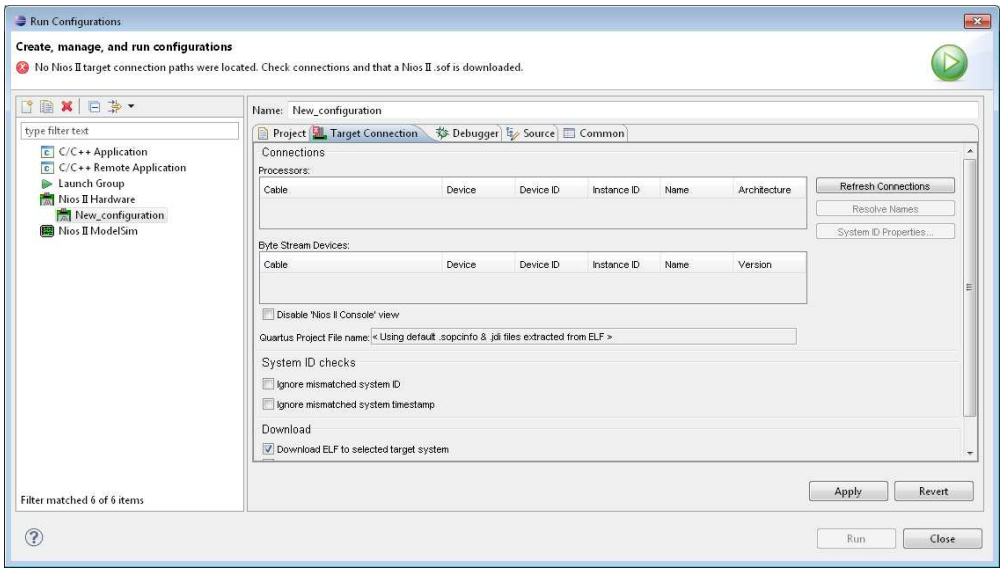
Entonces se mostrara una ventana de nombre Run Configuration.



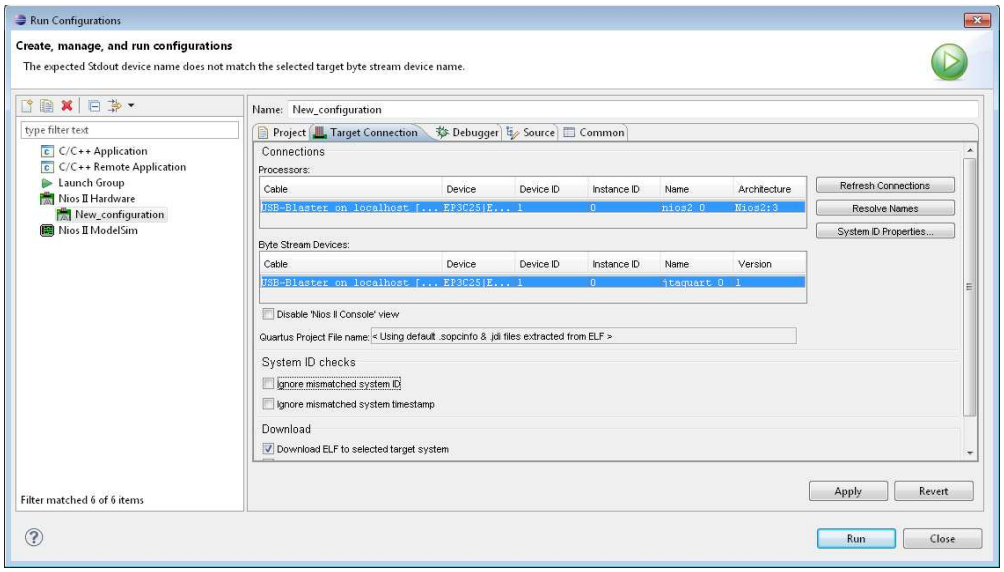
En el menú Project – Project name – Seleccionamos el nombre del proyecto.



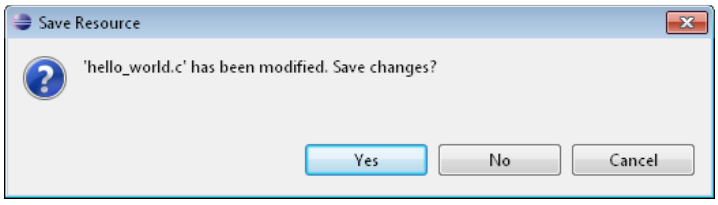
Menú Target Connection.



Hacemos clic en Refresh Connections, para verificar la presencia de la tarjeta DE0 Nano de Altera.



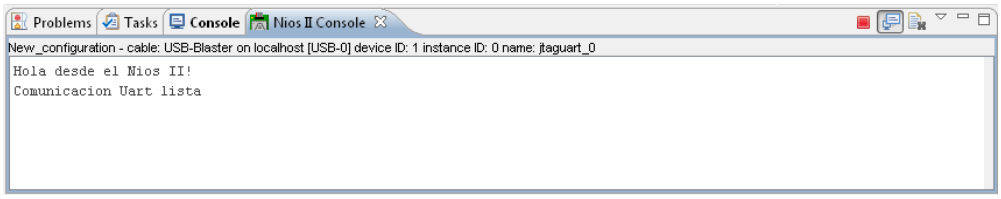
Finalmente se procede hacer clic en RUN.



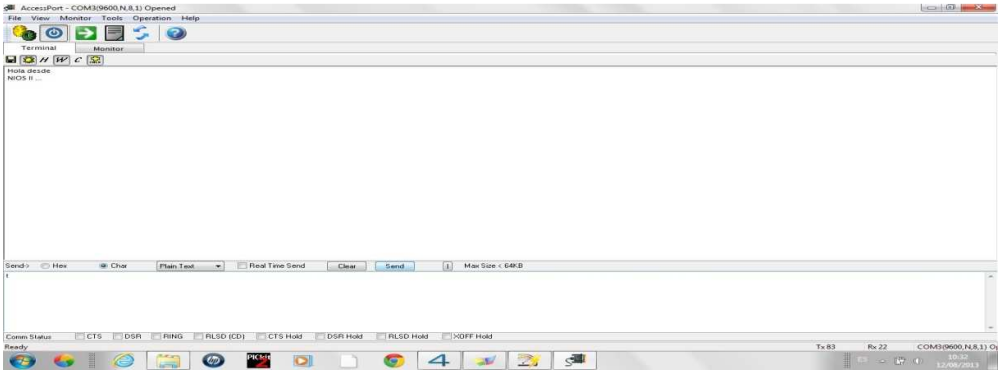
Una ventana nos pide guardar los cambios realizados en el código, seleccionamos YES.

Un menú de Nios II Console se despliega y nos muestra el mensaje siguiente “Comunicación serial exitosa”.

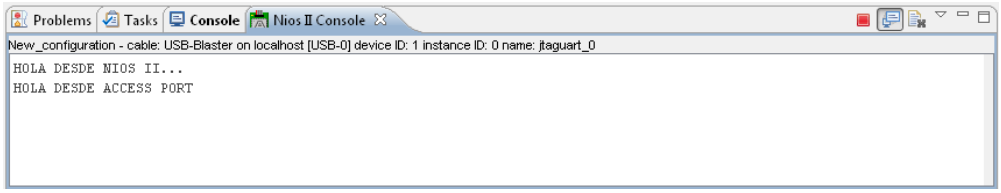
Mensaje de transmisión desde la Tarjeta DE0 Nano:



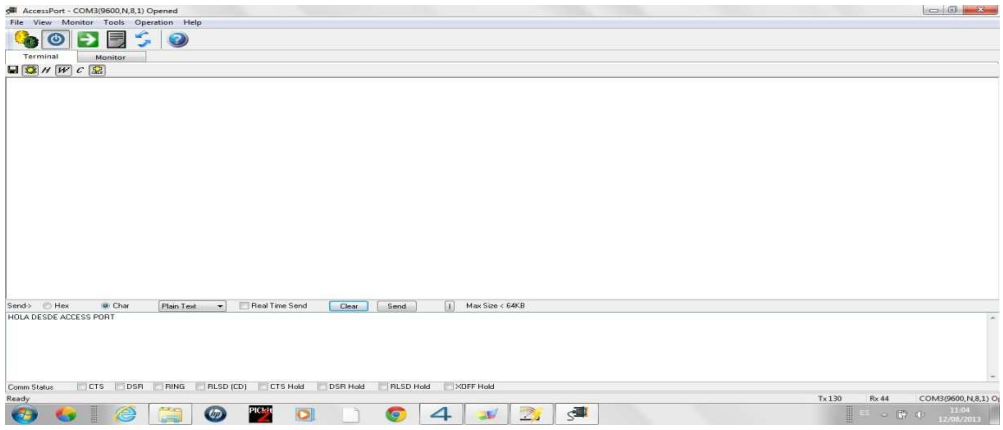
Mensaje de recepción desde Access Port:



Mensaje de recepción desde la Tarjeta DE0 Nano:

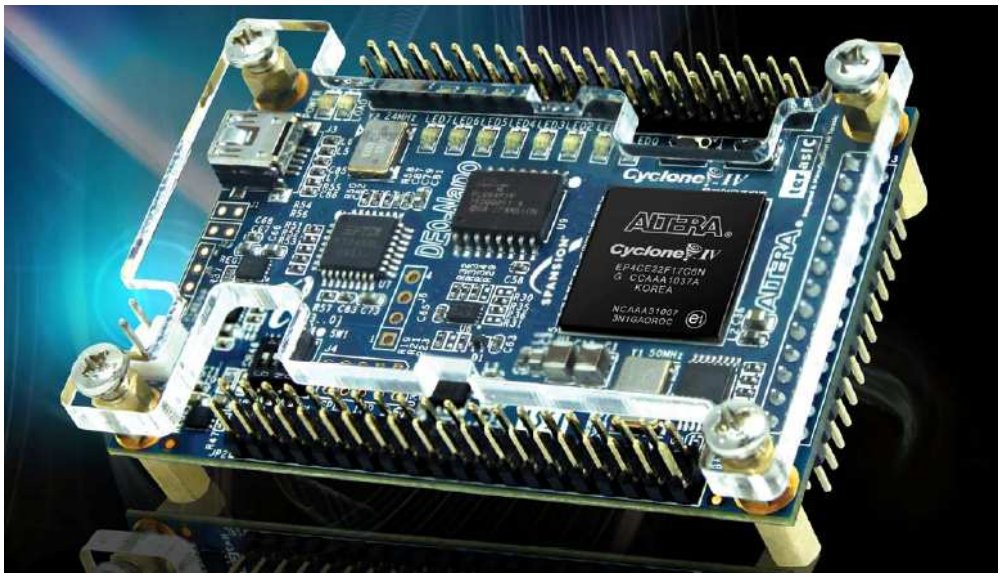


Mensaje de transmisión desde Access Port.



ANEXO #2

Práctica #4: Control velocidad de motor por PWM con la Tarjeta DE0 Nano de Altera programada bajo la plataforma NIOS II de Eclipse.



Tarjeta DE0 Nano de Altera

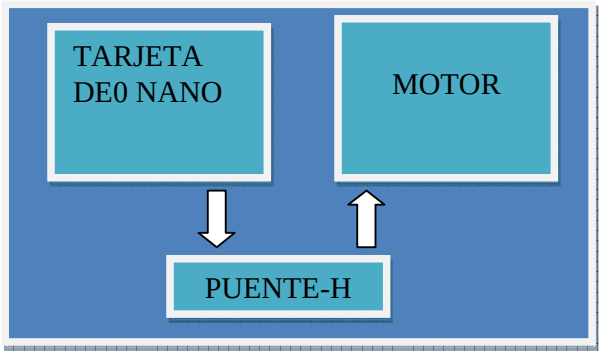
Objetivos:

- Configurar el componente de PWM en la Tarjeta DE0 Nano de Altera por medio de Qsys del Quartus II de Altera.
- Desarrollar un algoritmo en la plataforma Nios II de Eclipse para controlar la velocidad de un motor por medio de la Tarjeta DE0 Nano de Altera.

Descripción:

En la siguiente práctica se va a proceder a realizar un configuración desde el Qsys del Quartus II de Altera para realizar el control por PWM a través de un Puente I&T.

Diagrama de bloques:

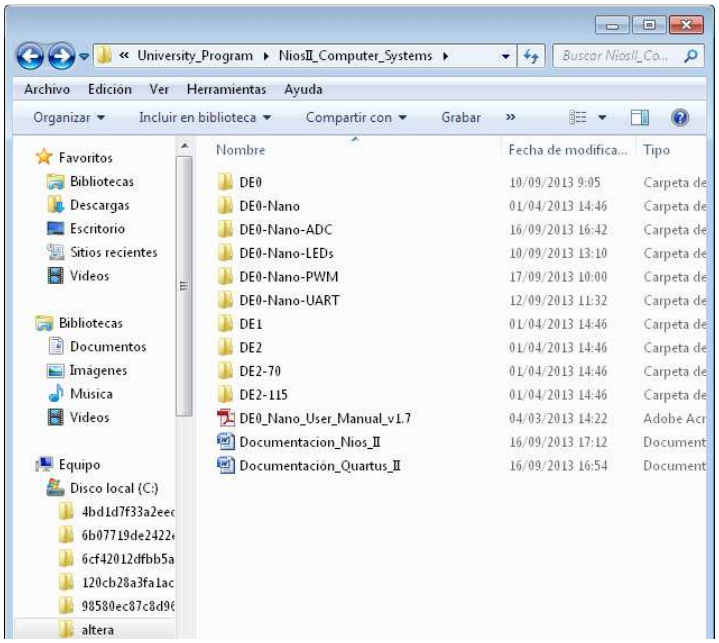


Desarrollo:

La práctica se desarrolla inicialmente haciendo una copia del archivo con el siguiente directorio:

En el cual se realiza una copia del archivo cuyo nombre es “DE0-Nano”, como se muestra en la imagen.

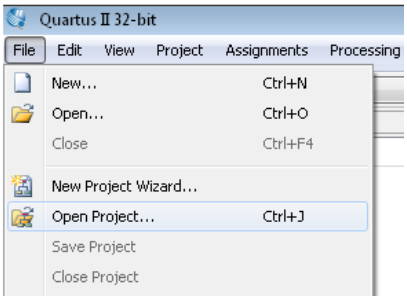
Una vez realizado este procedimiento, se procede a cambiarle el nombre como se muestra en la siguiente figura “DE0-Nano-PWM”.



QUARTUS II DE ALTERA

Abrir el Quartus II de Altera:

Se realiza los siguientes procedimientos para abrir el proyecto ubicado en la carpeta "DE0-Nano-PWM".



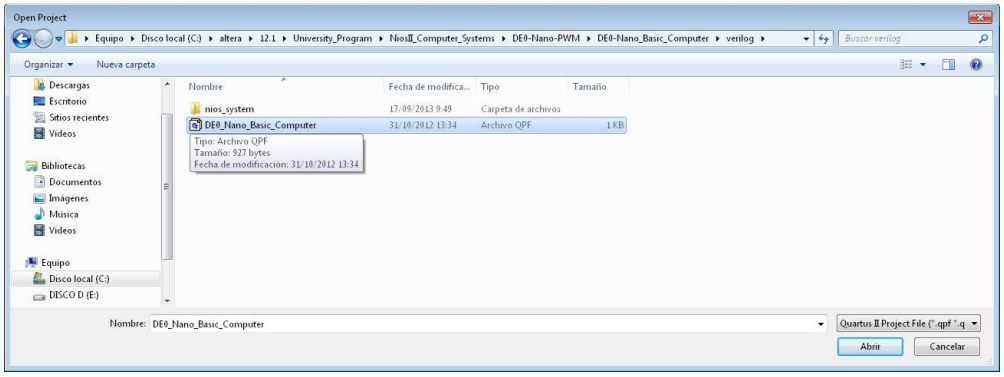
File

Open Project…

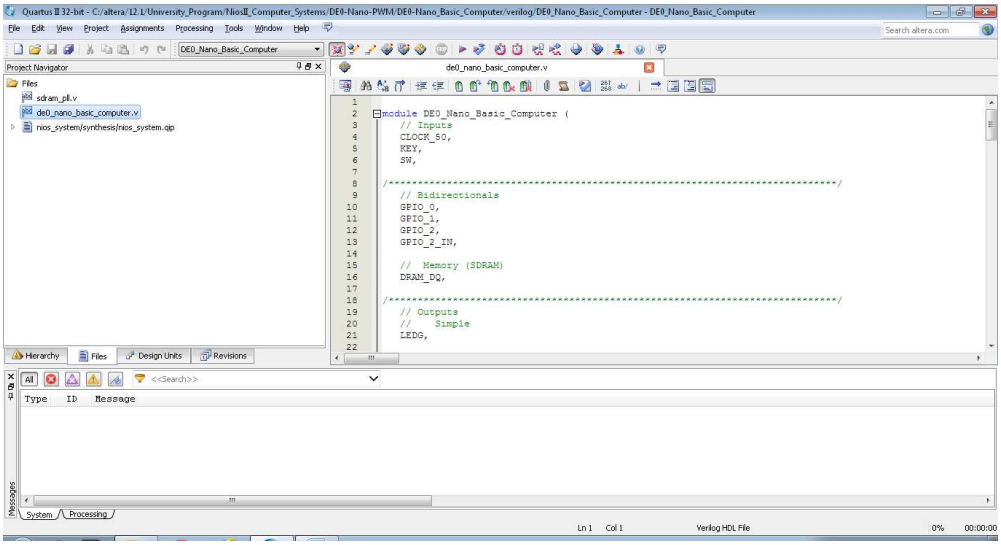
Se selecciona el archivo “DE0_Nano_Basic_Computer” ubicado en el siguiente directorio:

C:\altera\12.1\University_Program\NiosII_Computer_Systems\DE0-Nano-PWM\DE0-Nano_Basic_Computer\verilog.

Click en Abrir.



Ventana principal de Quartus II de Altera con archivo listo para su procesamiento.



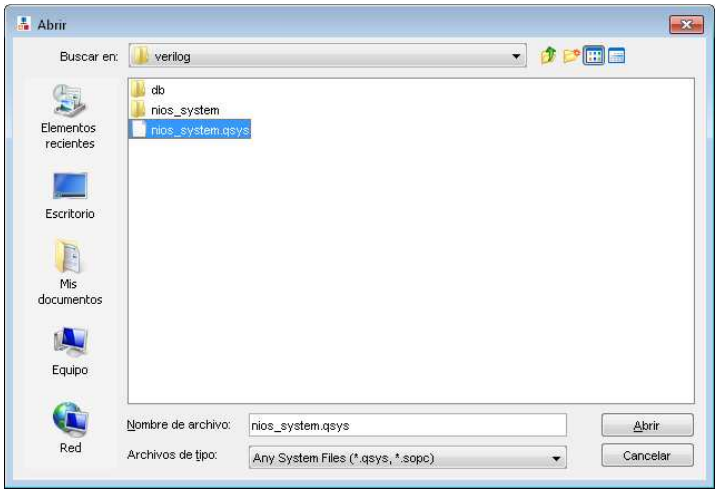
QSYS DEL QUARTUS II DE ALTERA

Editor de elementos en Qsys:

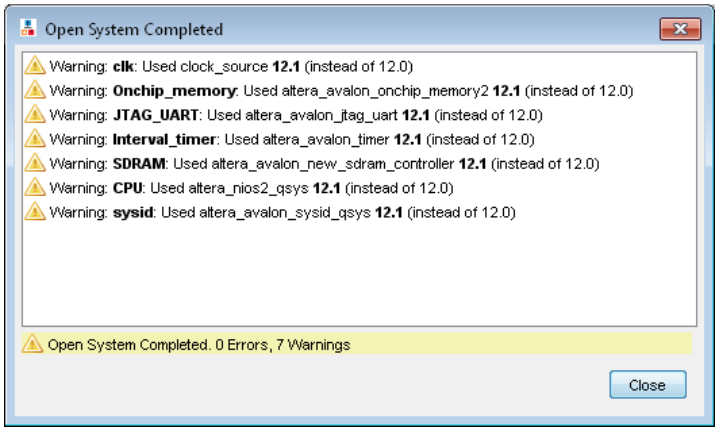
Se procede a abrir la ventana de Qsys haciendo clic en el icono que se muestra en la figura anterior.

La ventana que se abre nos indica que se debe de abrir el archivo de extensión .qsys ubicado en la carpeta de la práctica “DE0-Nano-UART” con directorio: C:\altera\12.1\University_Program\NiosII_Computer_Systems\DE0-Nano-PWM\DE0-Nano_Basic_Computer\verilog.

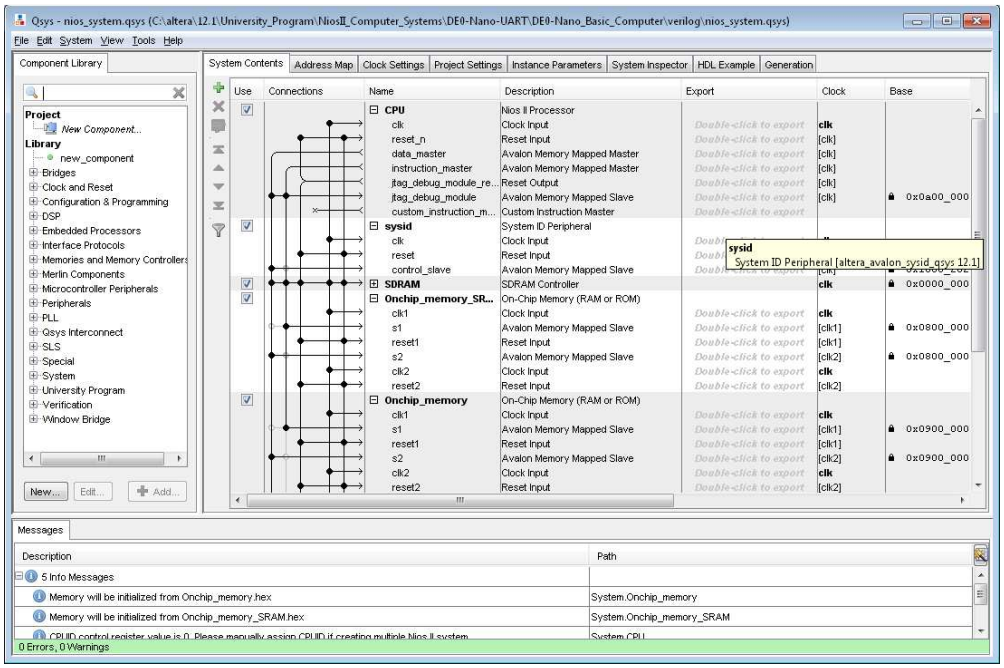
Se selecciona el archivo: nios_system.qsys y se hace clic en abrir.



Se mostrará un nueva ventana y hacemos clic en close.



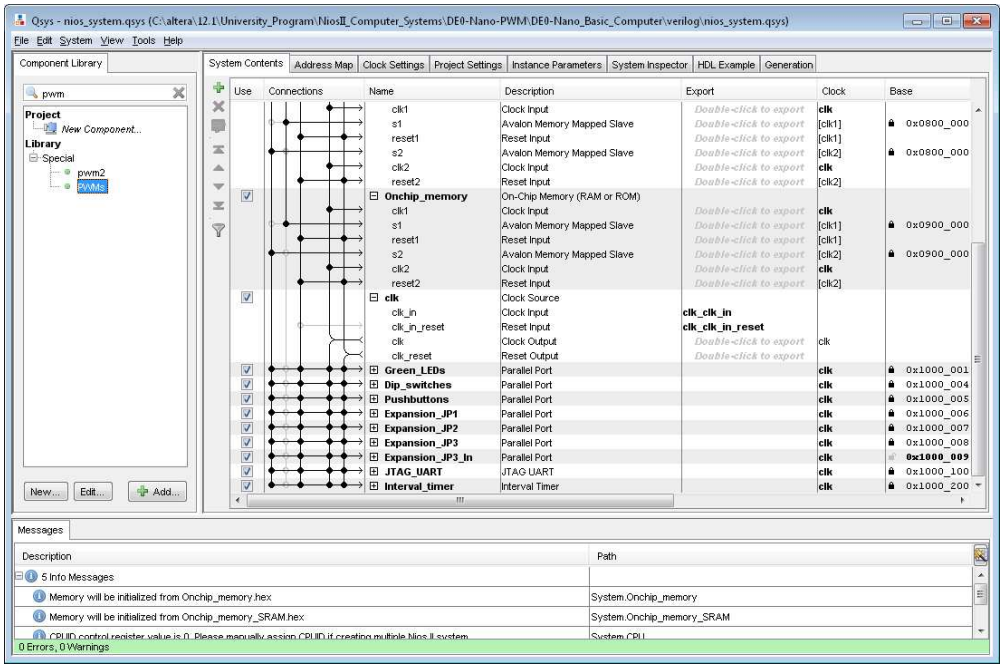
VENTANA PRINCIPAL DE QSYS.



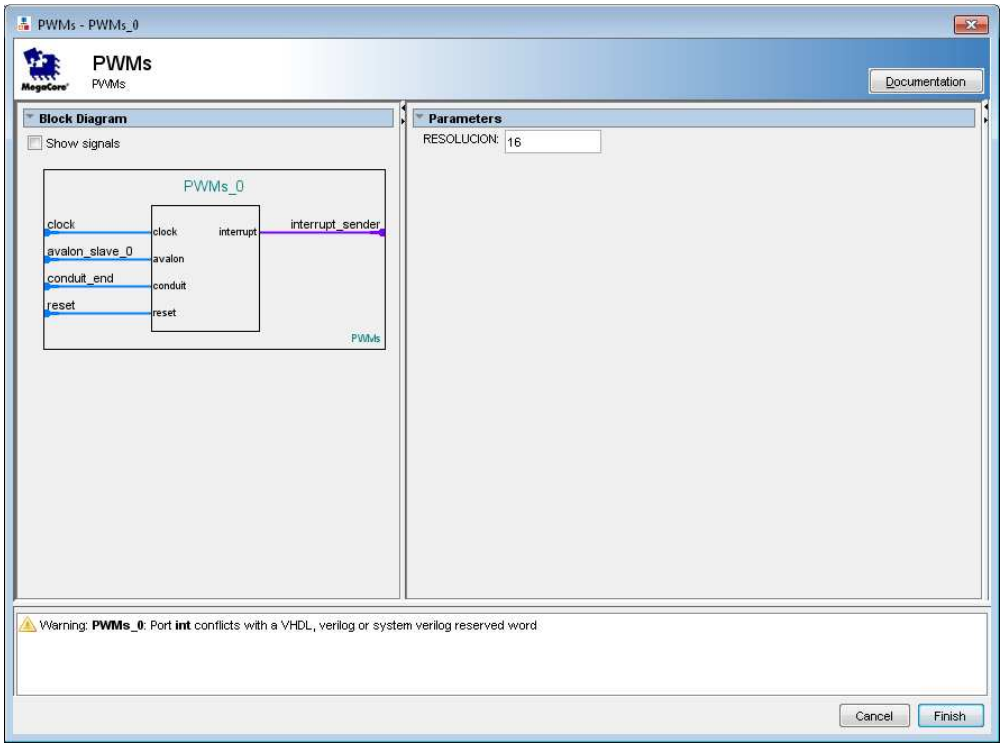
Por medio de la barra de desplazamiento vertical, se puede apreciar todos los componentes pertenecientes a la tarjeta DE0 Nano.

COMPONENETES DE QSYS

Para hacer uso del componente de PWM, nos dirigimos al menú de Component Library, para incorporar la librería respectiva. Se digita el nombre del componente y el buscador automáticamente nos presentará una librería con el nombre de pwm, como se muestra en la siguiente figura.



Seguidamente se muestra una ventana después de hacer doble clic en dicha librería "PWMs". Seguidamente de muestra una figura de la siguiente manera.

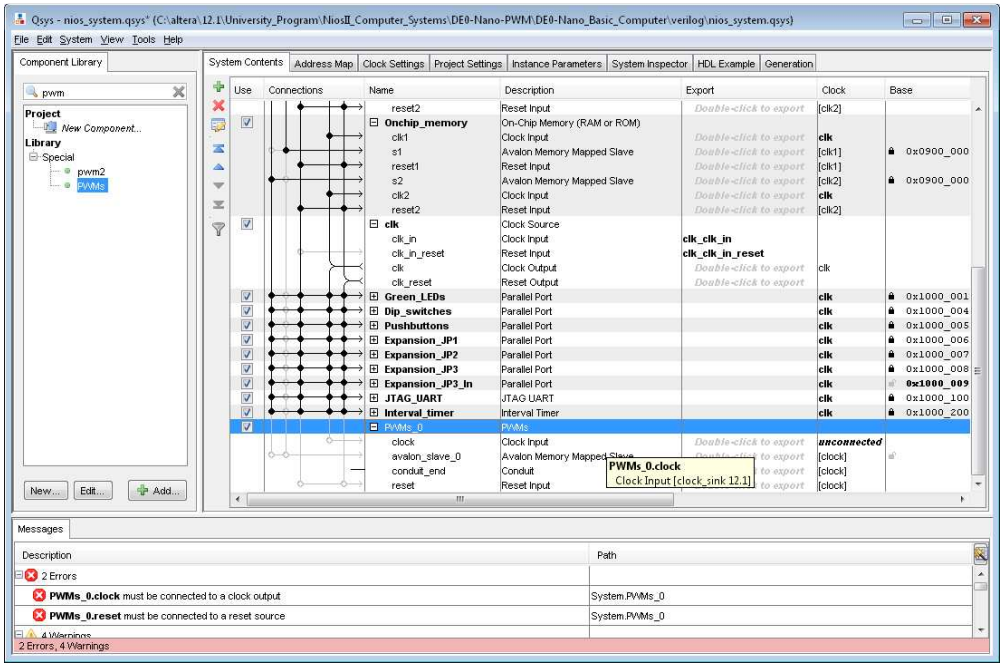


Esta ventana permite visulizar una resolución de 16 bits.

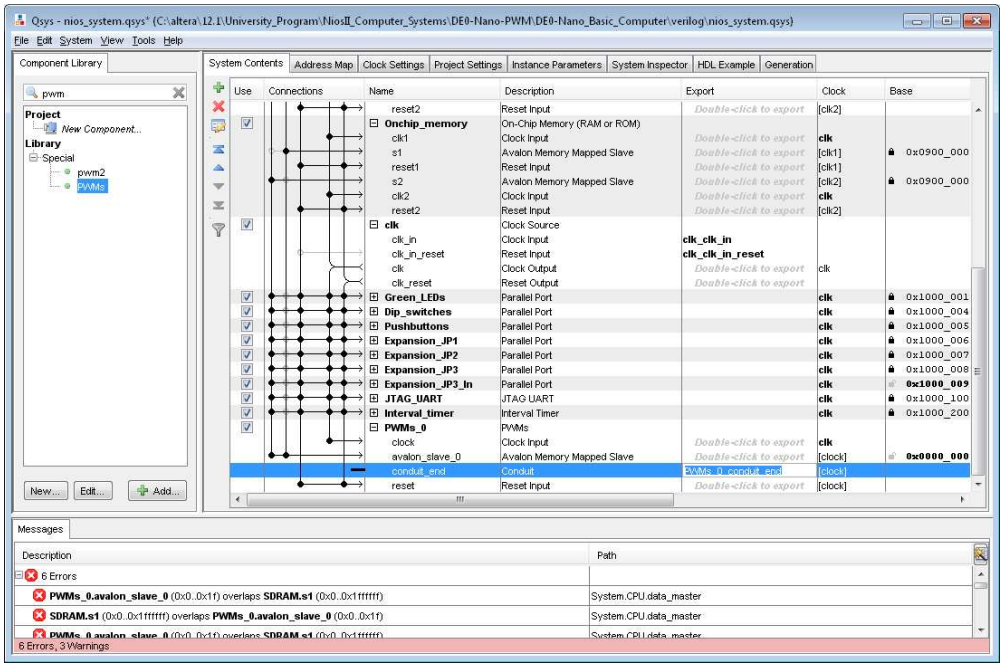
Hacer clic en Finish.

En la siguiente figura se puede apreciar que al añadir una nueva librería, aparecen errores descritos en la ventana inferior de “Description”.

También se puede observar que la librería de PWMs que ha sido incorporada con éxito.



Para configurar la nueva librería se debe empezar haciendo las conexiones respectivas, así como se observa en la siguiente imagen.

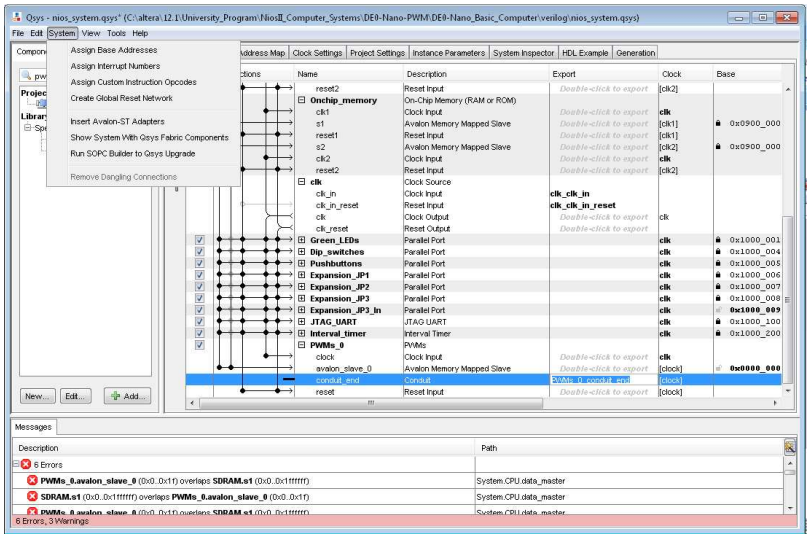


Una vez realizada las conexiones, es importante exportar la señal de External Interface, haciendo doble clic en el menú de Export.

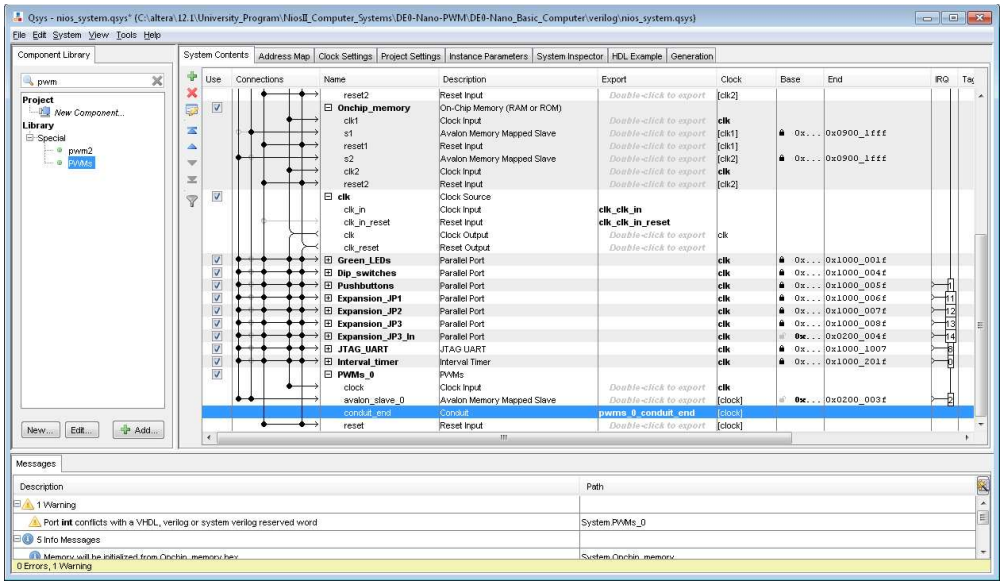
Una vez realizadas la conexión ahora se procede a eliminar los errores, realizando los siguientes pasos para asignar espacio de memoria.

Click en System.

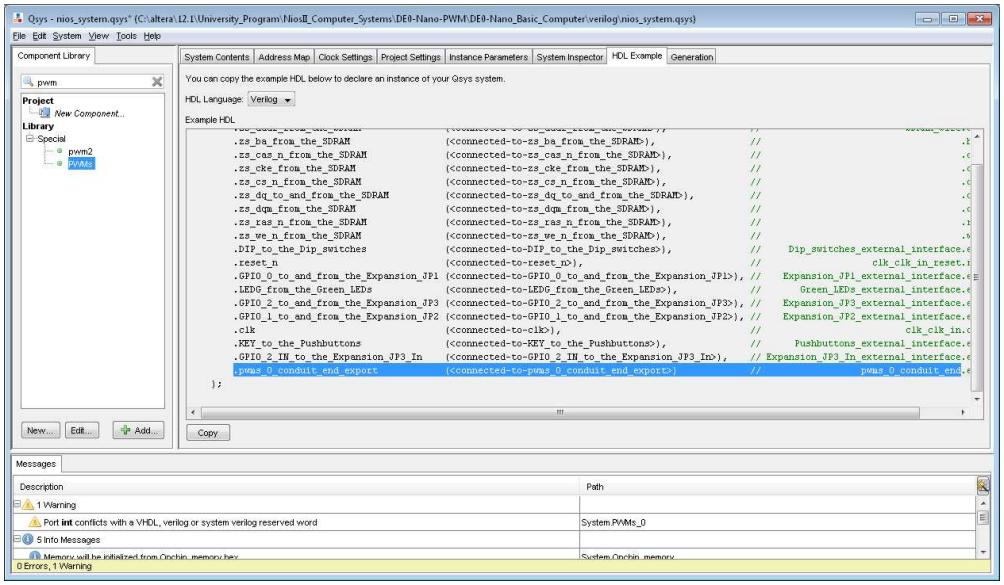
Click en Assign Base Addresses.



Los errores se borraran y obtendremos una ventana libre de errores en el menú de “Description”. 0 Errors, 0 Warnings.

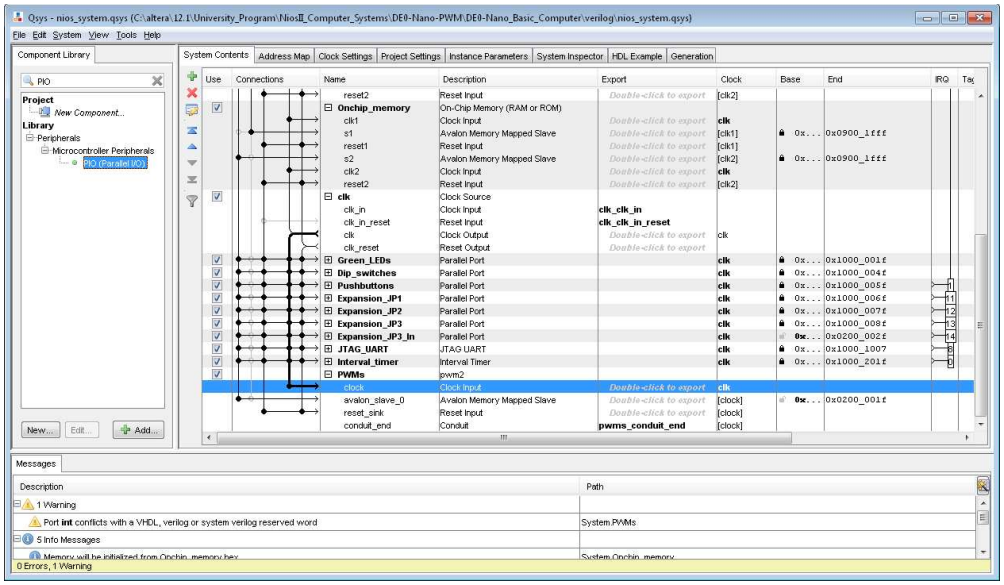


Como siguiente paso debemos ir al menú de HDL Example, para copiar en código de la librería correspondiente.

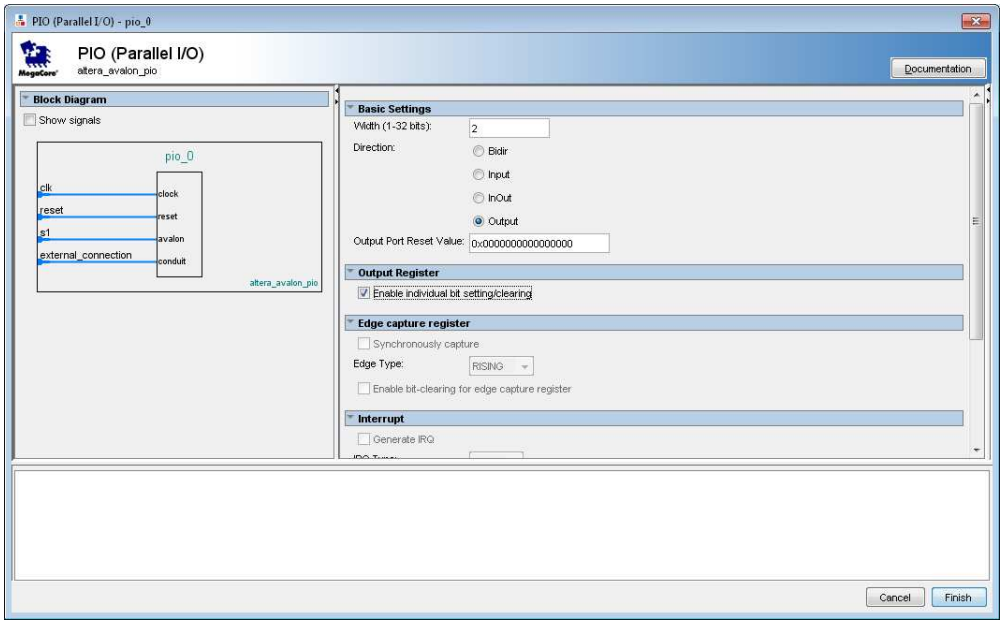


****

Para hacer uso del control de la dirección del motor, nos dirigimos al menú de Component Library, para incorporar la librería respectiva a PIO (Parllel I/O). Se digita el nombre del componente y el buscador automáticamente nos presentará una librería con el nombre de PIO, como se muestra en la siguiente figura.



Seguidamente se muestra una ventana después de hacer doble clic en dicha librería “pio”. Seguidamente de muestra una figura de la siguiente manera.



Esta ventana permite visulizar una y configurar un Width (1-32 bits).

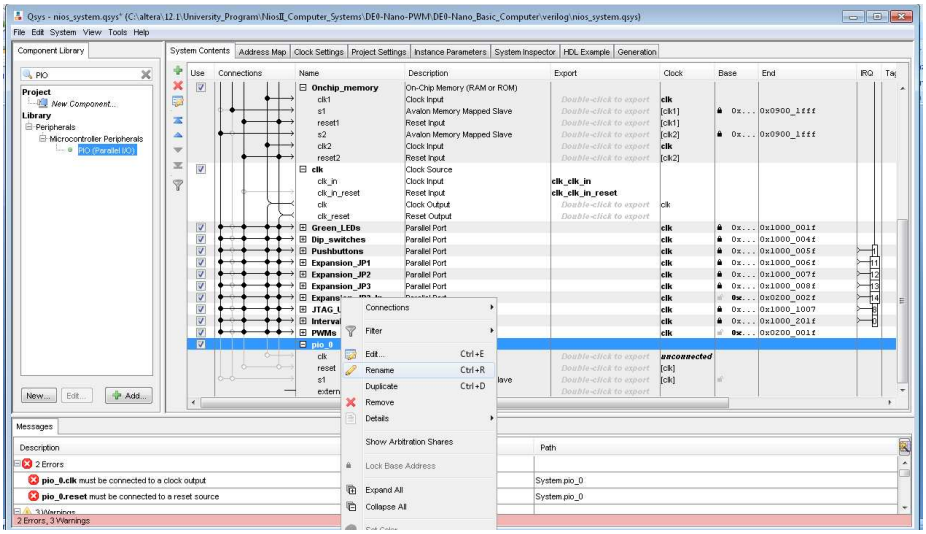
Seleccionar 2,

Seleccionar Output Register.

Hacer clic en Finish.

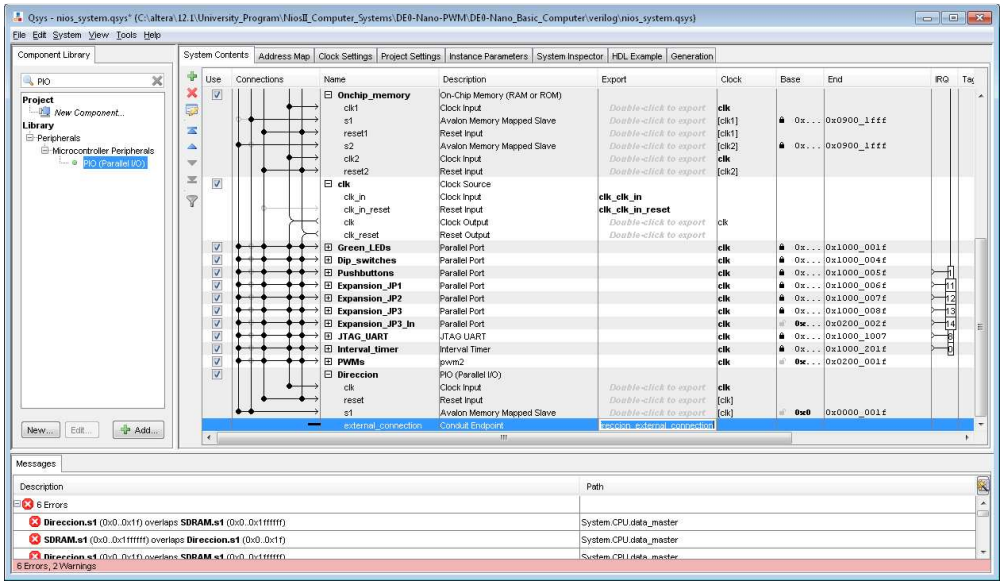
En la siguiente figura se puede apreciar que al añadir una nueva librería, aparecen errores descritos en la ventana inferior de “Description”.

También se puede observar que la librería de PIO que ha sido incorporada con éxito.



Se digita un nombre haciendo clic derecha en el nombre de la librería.

Para configurar la nueva librería se debe empezar haciendo las conexiones respectivas, así como se observa en la siguiente imagen.

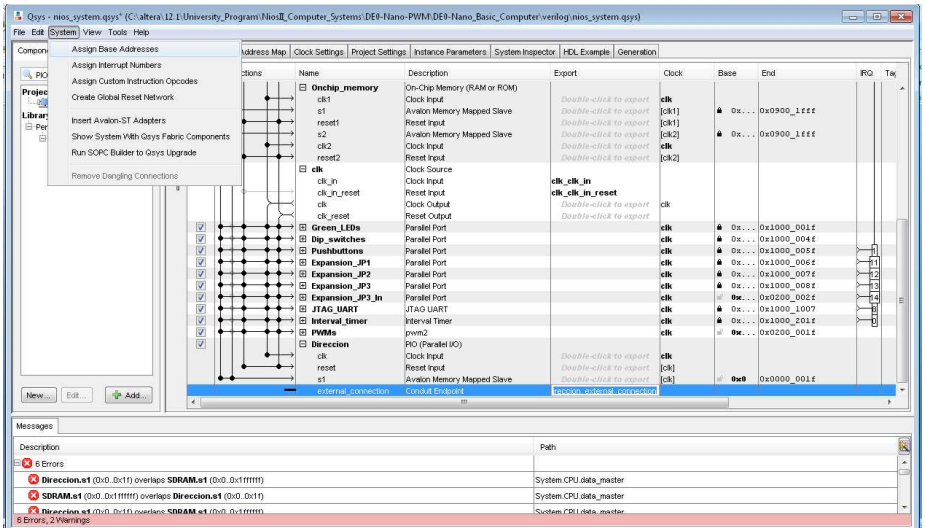


Una vez realizada las conexiones, es importante exportar la señal de External Interface, haciendo doble clic en el menú de Export.

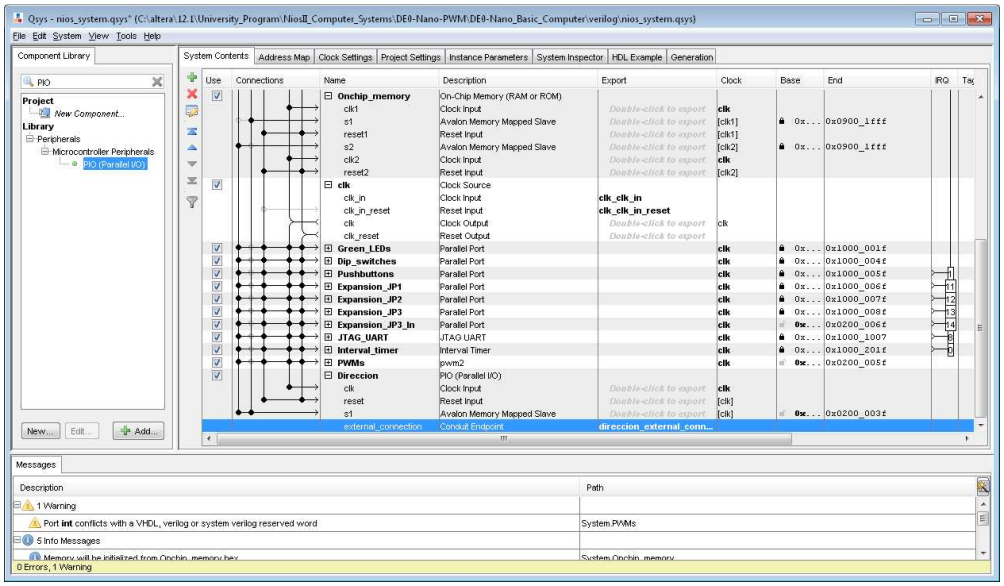
Una vez realizadas la conexión ahora se procede a eliminar los errores, realizando los siguientes pasos para asignar espacio de memoria.

Click en System.

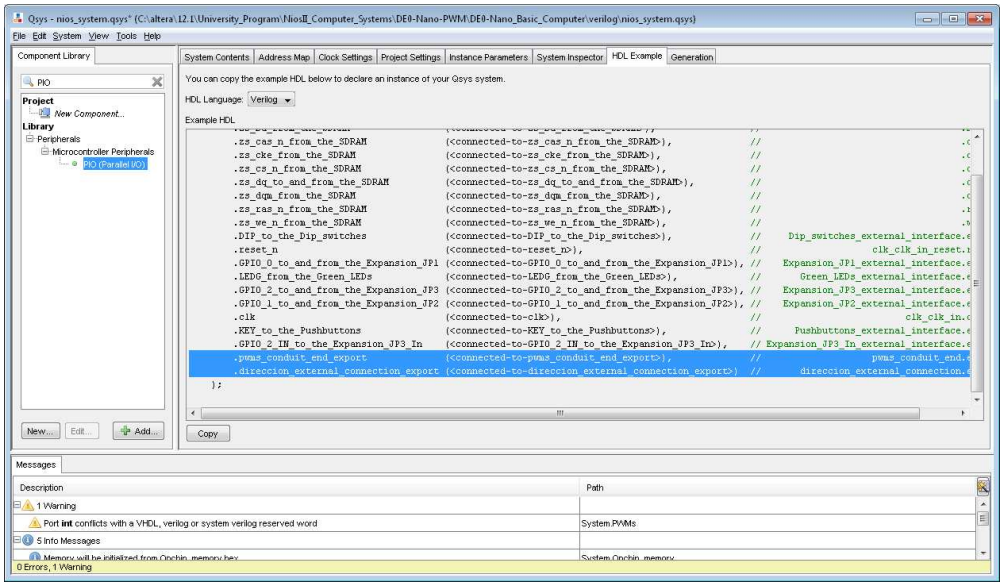
Click en Assign Base Addresses.



Los errores se borraran y obtendremos una ventana libre de errores en el menú de “Description”. 0 Errors, 0 Warnings.



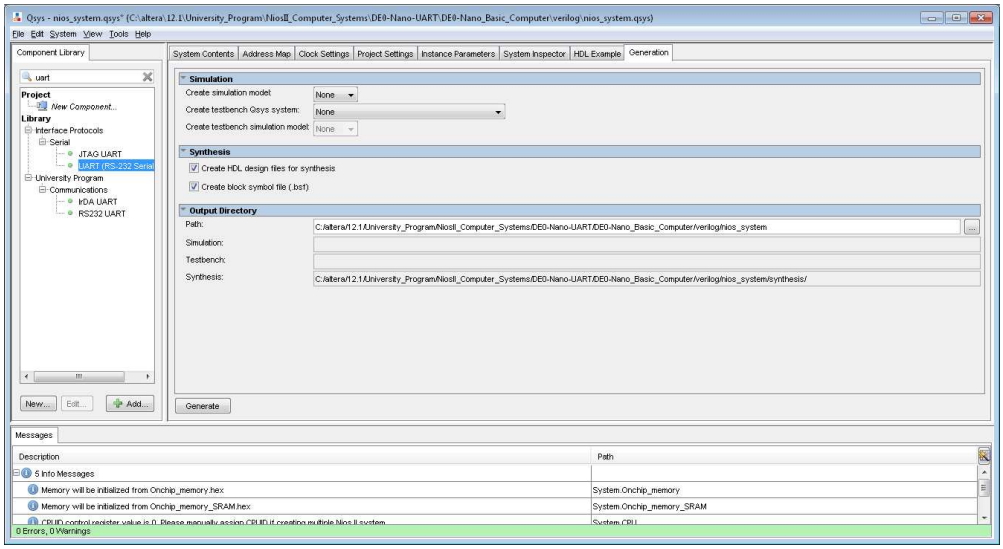
Como siguiente paso debemos ir al menú de HDL Example, para copiar en código de la librería correspondiente.



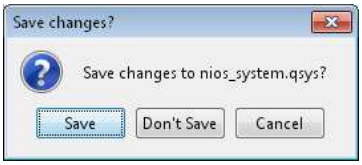
Nota: Copiar el código respectivo para PWM y PIO.

En el siguiente paso procedemos a generar la máquina:

- Seleccionamos el menú Generation.
- Hacer clic en Generate para generar el sistema.

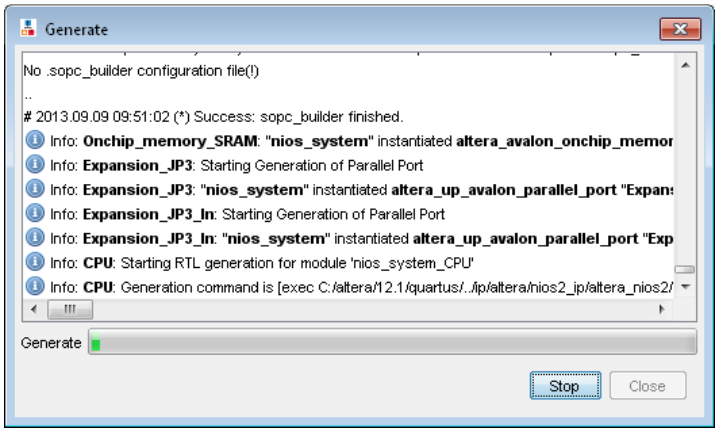


Una ventana para guardar los cambios aparece y aceptamos los cambios realizados.

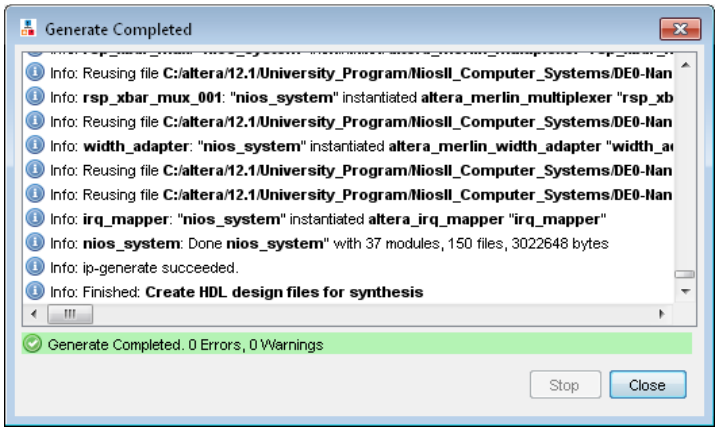


Esperamos hasta que se realice la compilación respectiva:

Proceso de construcción de la máquina se manifiesta en la figura siguiente.

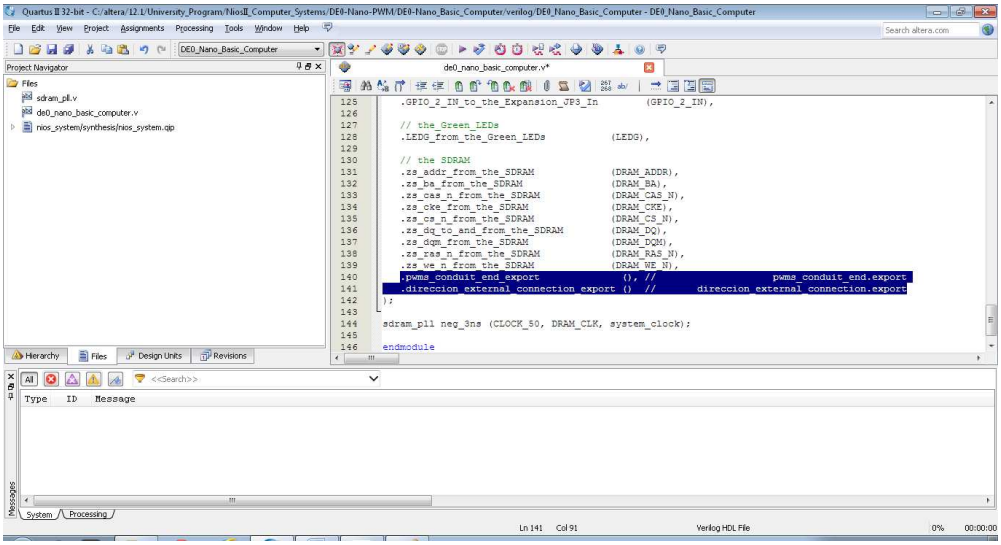


Una ventana nos manifiesta que la construcción se ha realizado con éxito “0 Errors”.

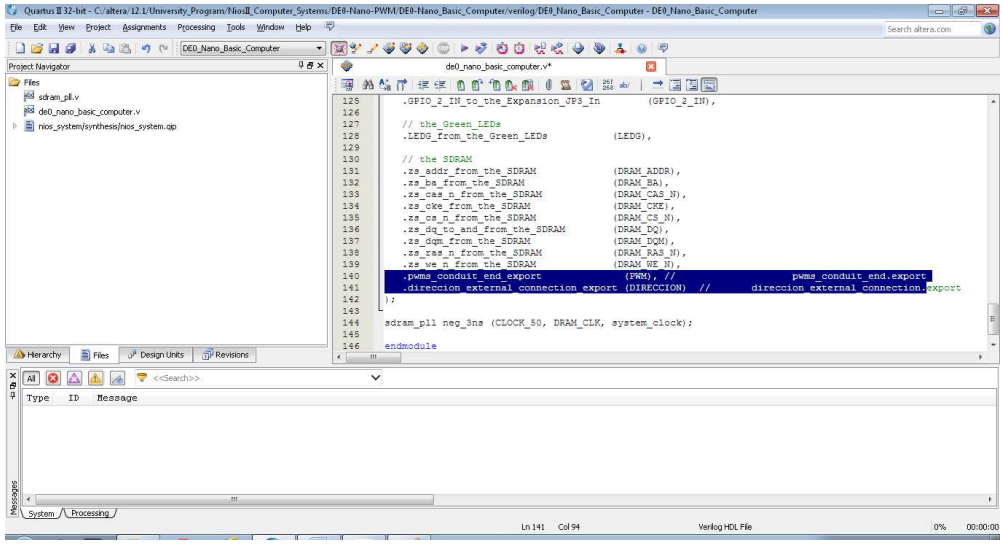


- Click en close.
- Cerrar la ventana de Qsys.

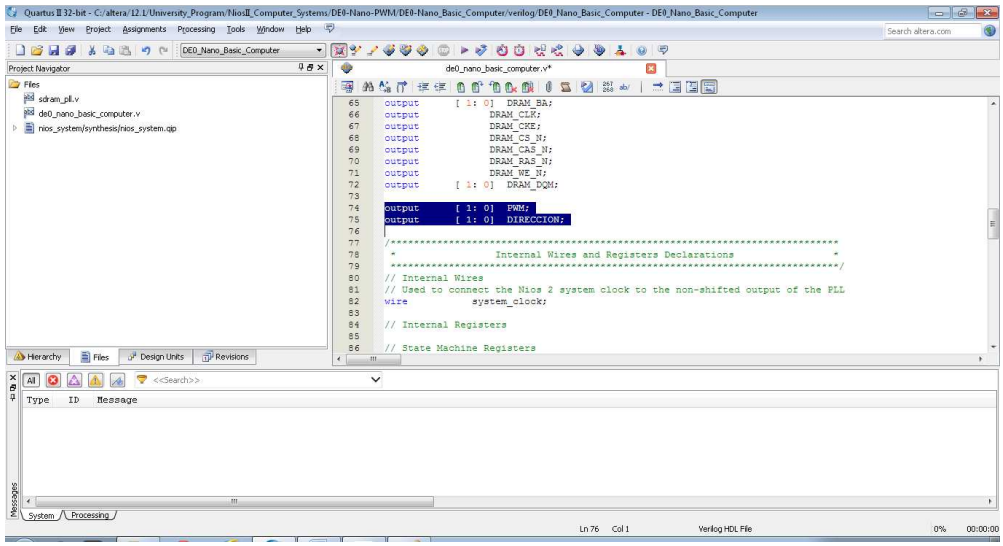
Una vez generada la máquina, se procede a pegar el código copiado de QSYS en la parte del código perteneciente a la máquina básica DE0 Nano.



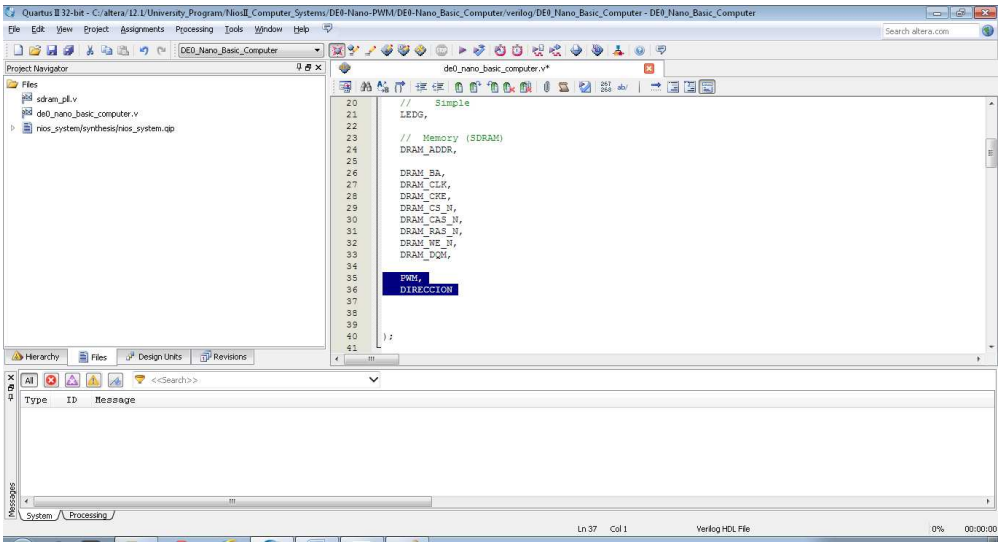
Posteriormente se necesita escribir un nombre para el cual van a ser las señales de control de la librería correspondiente a la Comunicaion Serial Uart, así como se muestra de figura.



El nombre de cada señal seleccionada debe de ser declarada como entrada y salida respectivamente como se muestra en la siguiente figura.

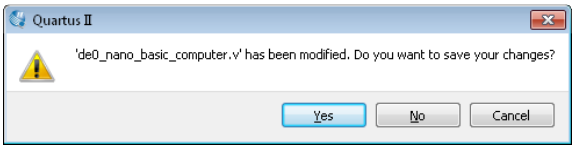


Así también debe de ser mencionada cada una de las señales de control en la sección de module DE0_Nano_Basic_Computer.

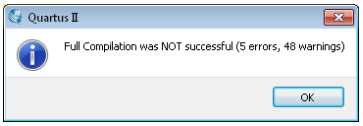


En la ventana de Quartus II, se hace clic en icono de Start Compilation.

Se guardan los cambios respectivos haciendo clic en YES.

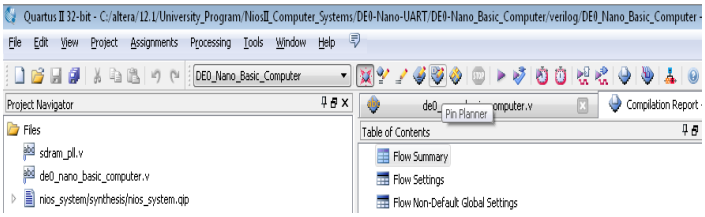


Una ventana similar a la siguiente figura aparece con el motivo de anunciar de que las nuevas señales generadas aun no han sido configuradas en el Pin Planner.



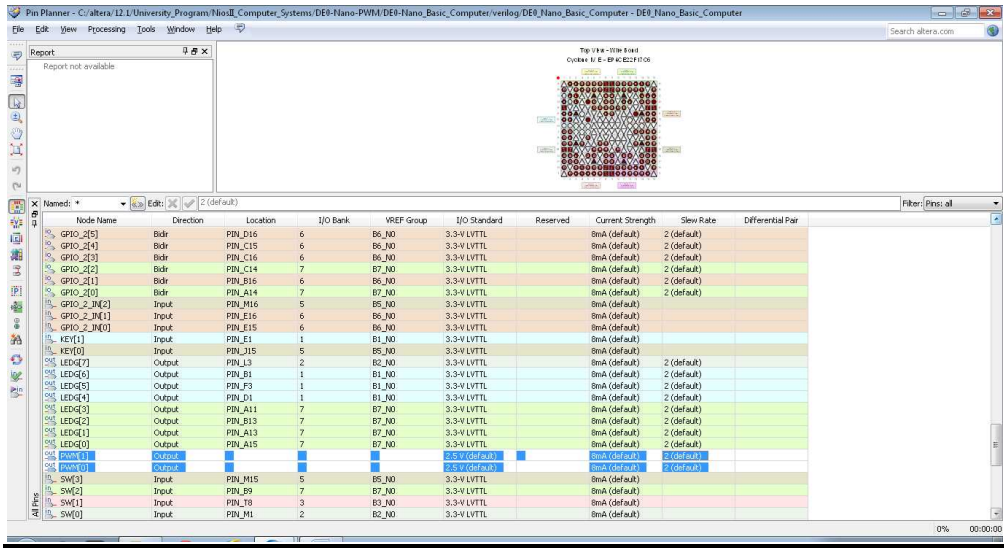
Clic Ok

Nos introducimos en el Pin Planner, haciendo clic en su icono correspondiente.

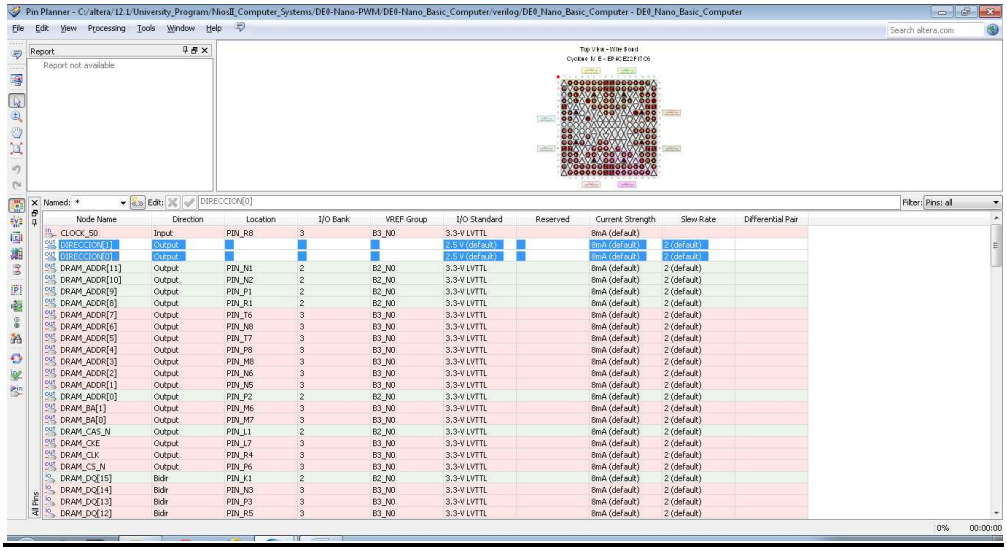

Quartus II 32-bit - C:/altera/12.1/University_Program/NiosII_Computer_Systems/DE0-Nano-UART/DE0-Nano_Basic_Computer/verilog/DE0_Nano_Basic_Computer -
File Edit View Project Assignments Processing Tools Window Help
DE0_Nano_Basic_Computer
Project Navigator
Files
sdram_pll.v
de0_nano_basic_computer.v
nios_system/synthesis/nios_system.qip
Pin Planner
Compilation Report
Table of Contents
Flow Summary
Flow Settings
Flow Non-Default Global Settings

VENTANA PRICIPAL DE PIN PLANNER.

COMPONENTE DE PWM



COMPONENTE DIRECCION DEL MOTOR



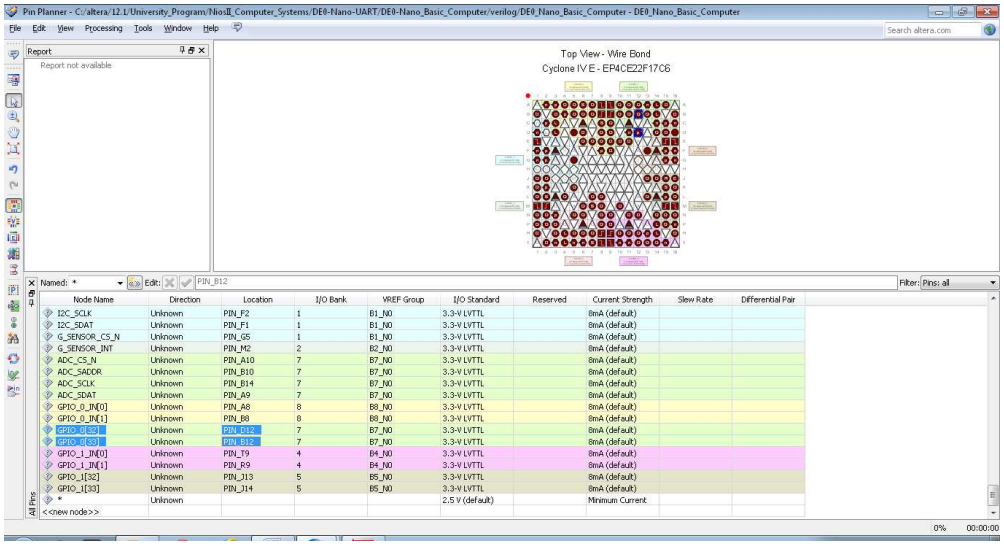
Se puede observar las dos señales PWM y DIRECCION, y edita el menú de Location e I/O Bank, para que finalmente quede configurado de la siguiente manera y ubicado en los siguientes pines de la Tarjeta DE0 Nano de Altera.

La señal de PWM de ha sido asignada para este caso en el PIN_D12 y PIN_B12, correspondientes a los pin de GPIO_032 y GPIO_033 de la tarjeta.

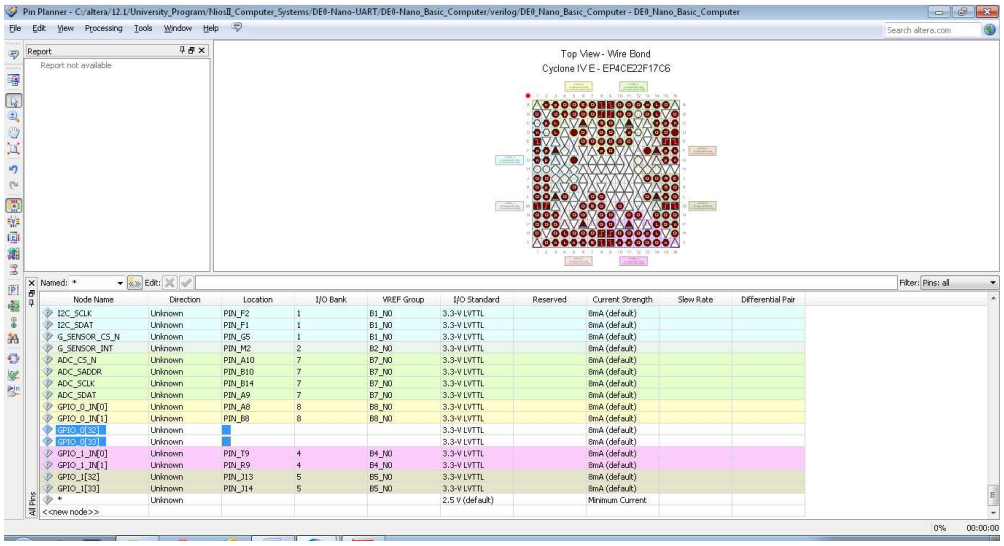
La señal de dirección han sido asignadas en el PIN_A12 y PIN_D11, correspondientes a los pin de GPIO_030 y GPIO_031 de la tarjeta.

Nota: El PIN_D12, PIN_B12, PIN_A12 y PIN_D11 están ocupados originalmente, por el cual es necesario ubicarlos en el Pin Planner y eliminar su asignación de la siguiente manera.

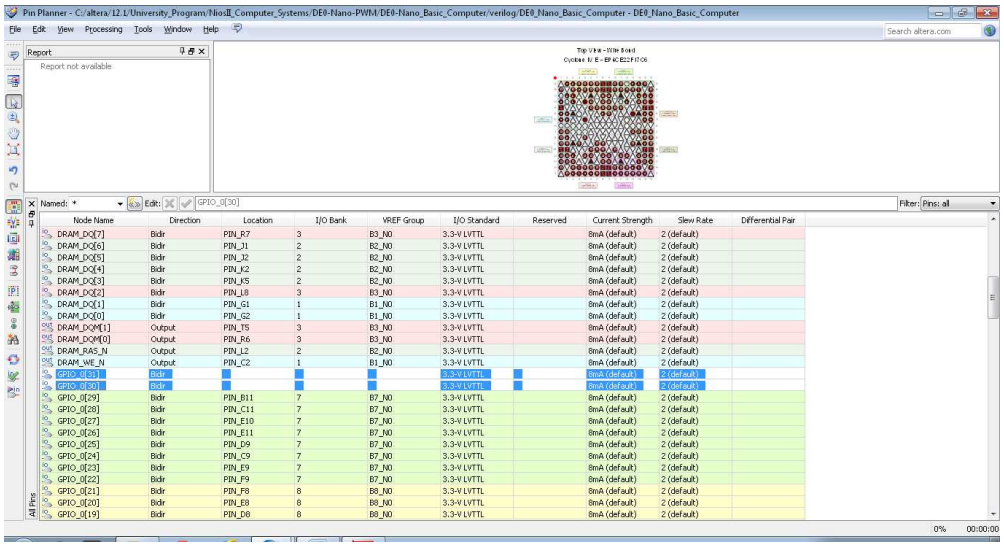
Ubicación de los pines PIN_D12, PIN_B12, PIN_A12 y PIN_D11



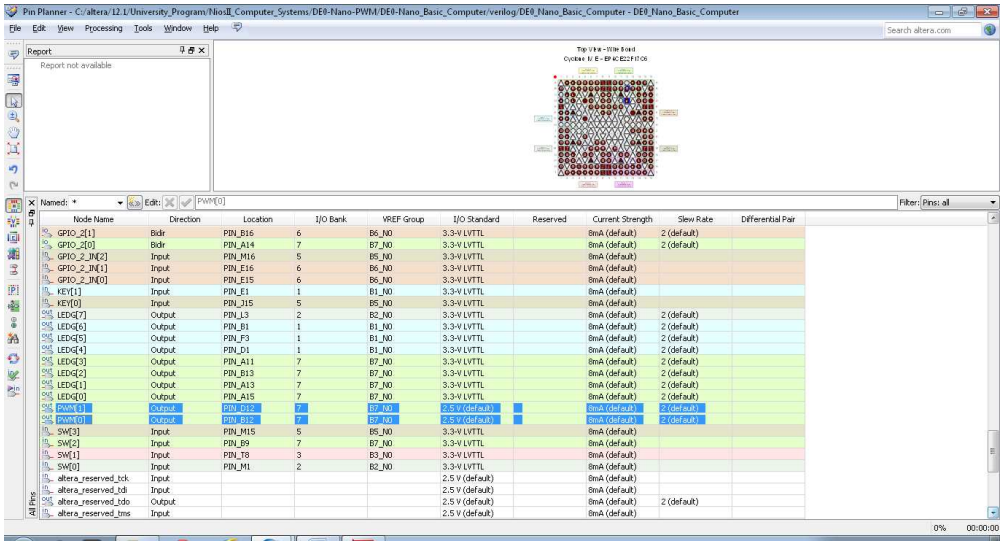
Eliminación de su asignación original PIN_D12 y PIN_B12 para reincorporar la señal de PWM.



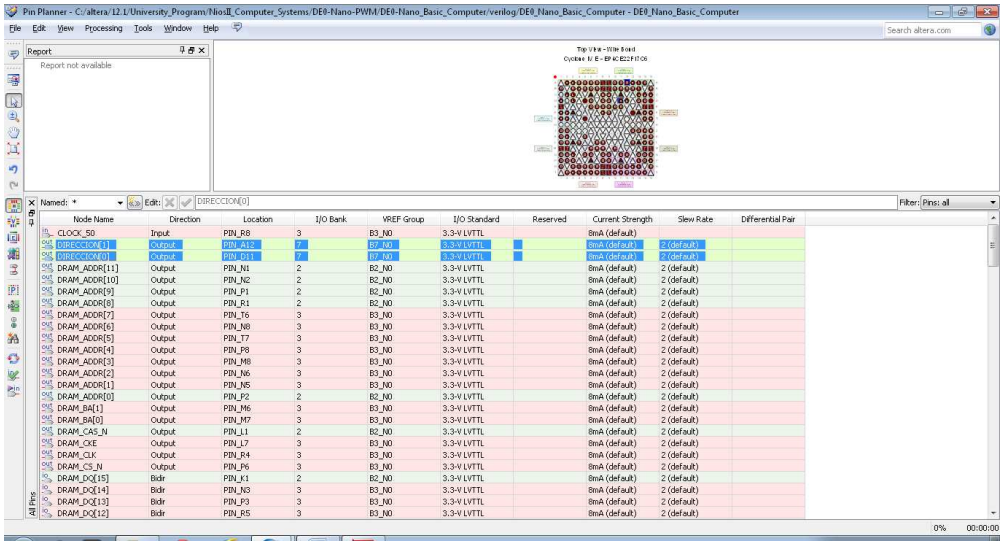
Eliminación de su asignación original PIN_D12 y PIN_B12 para reincorporar la señal de DIRECCION.



Asignacion de señales de control PWM.

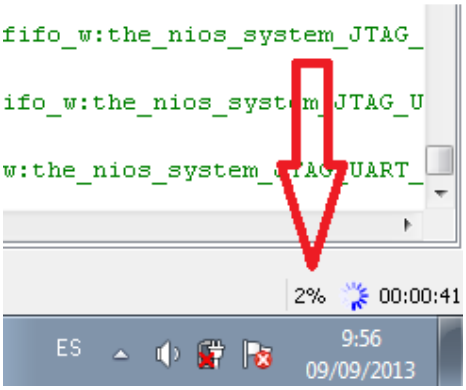


Asignacion de señales de control DIRECCION.

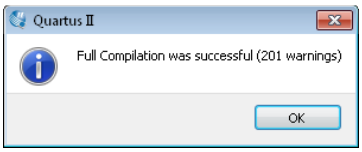


En la ventana de Quartus II, se hace clic en icono de Start Compilation nuevamente.

Se toma un tiempo en compilar la máquina en Quartus II, hasta que su valor llegue a un 100% como se indica en la figura.

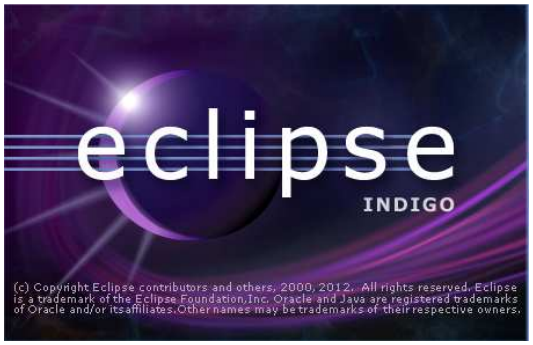


Una ventana aparece y nos indica que no existen errores durante su compilación.

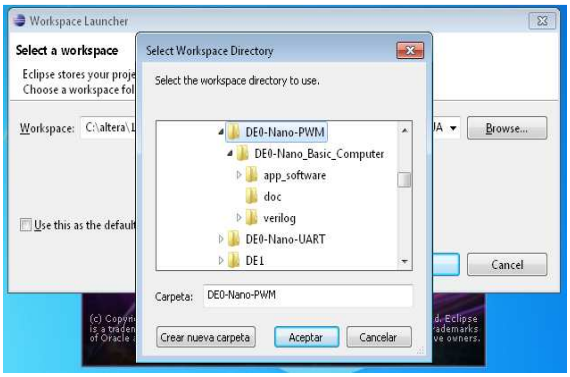


Click en OK.

PLATAFORMA NIOS II DE ECLIPSE.



Se selleciona un espacio de trabajo, en este caso se decidió hacerlo dentro de la carpeta de ubicación de la practica DE PWM.

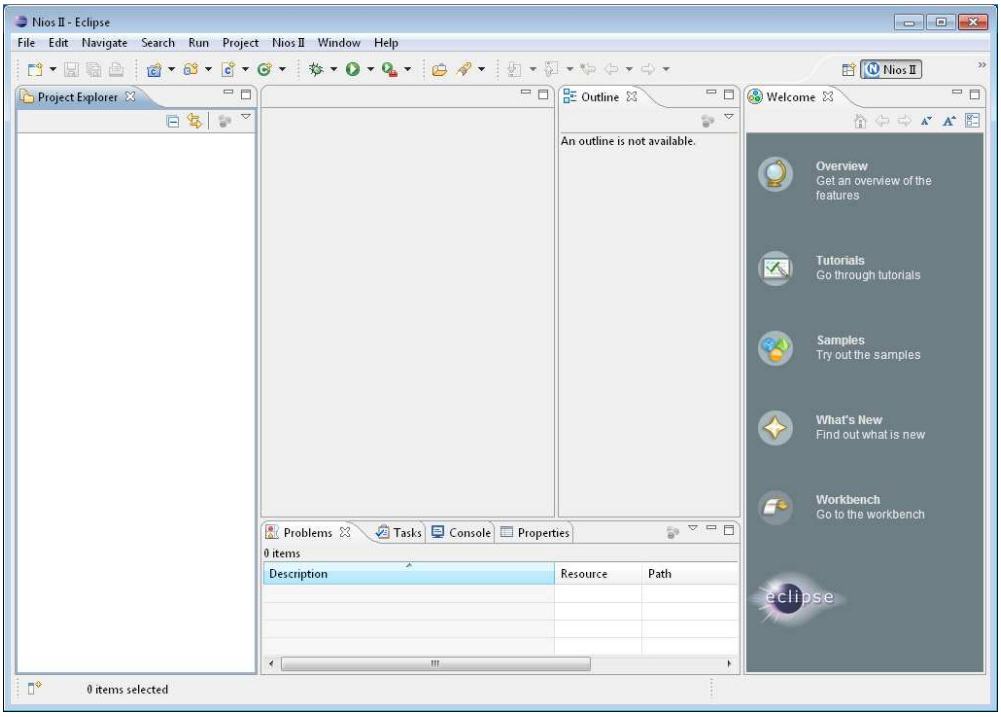


Buscamos la dirección de la carpeta de nuestro archivo.

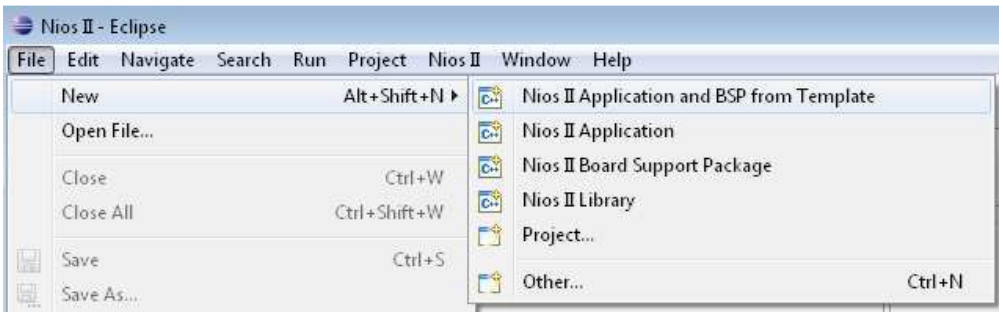
Click en aceptar.

Clic Ok.

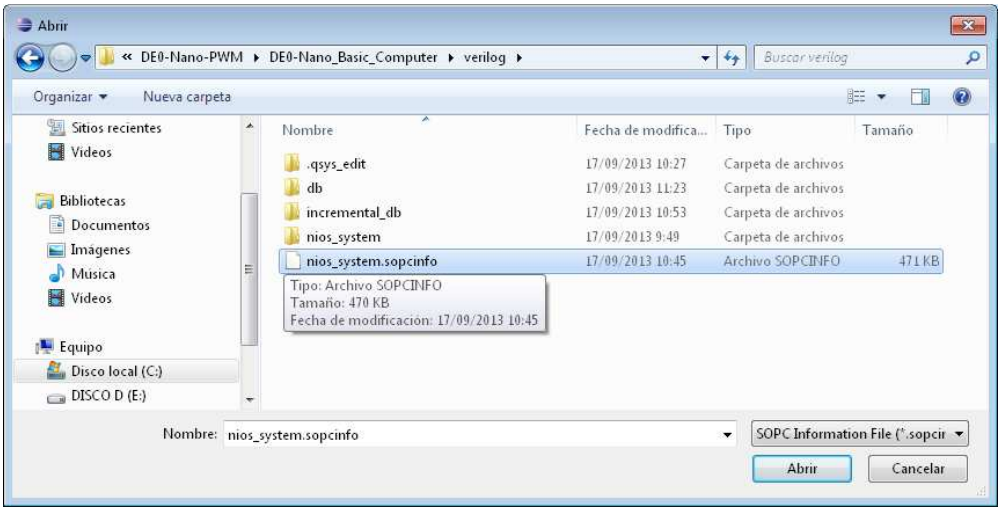
VENTANA PRICIPAL DE NIOS II DE ECLIPSE.



Se procede a crear un nuevo proyecto realizando los siguientes pasos:

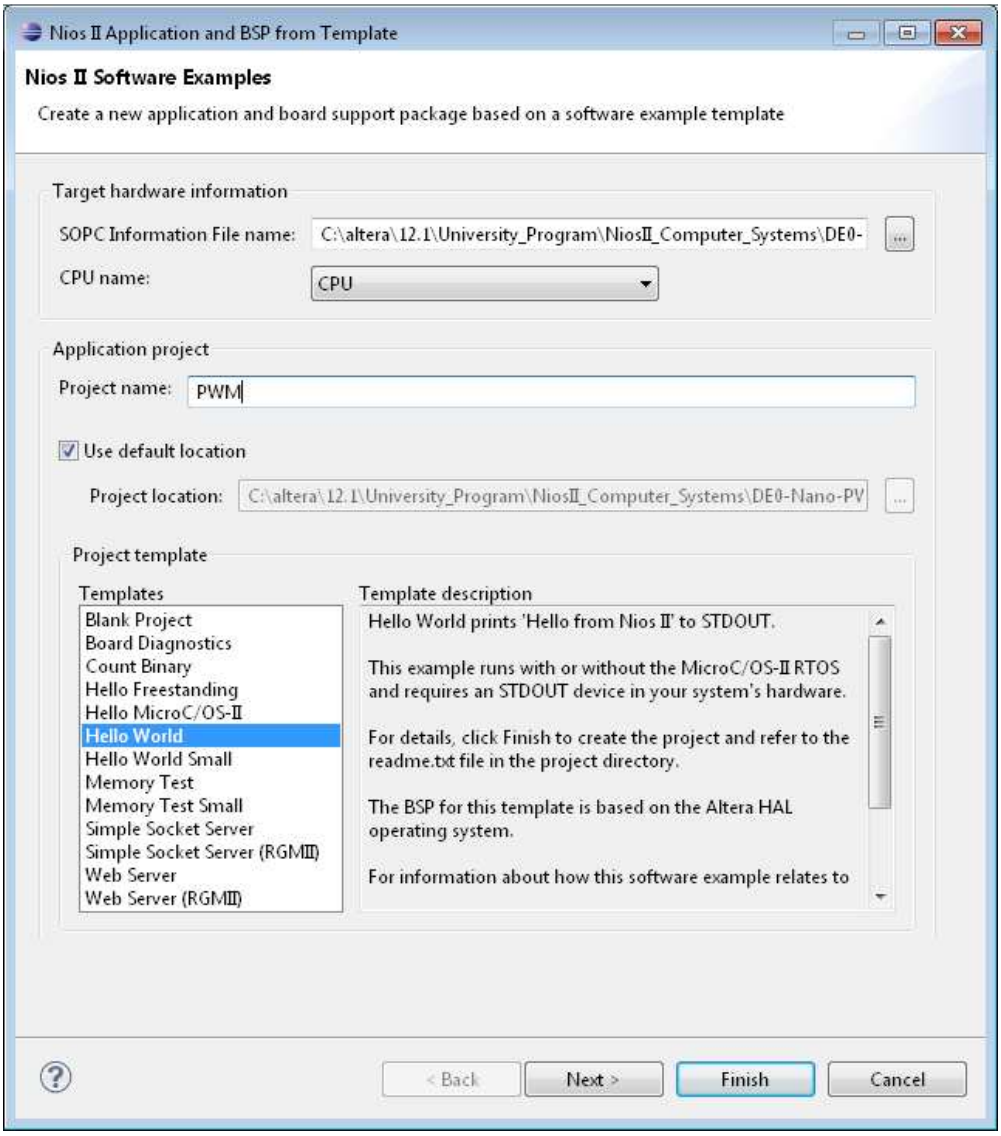


File - New – Nios II Application And BSP from Template.



Se selecciona el archivo nios_system.sopcinfo haciendo clic en SOPC Information File Name.

El directorio de establece y muestra nuestra maquina lista para ser utilizada en el menú de CPU name:



Escribimos un nombre en el menú de Project name:

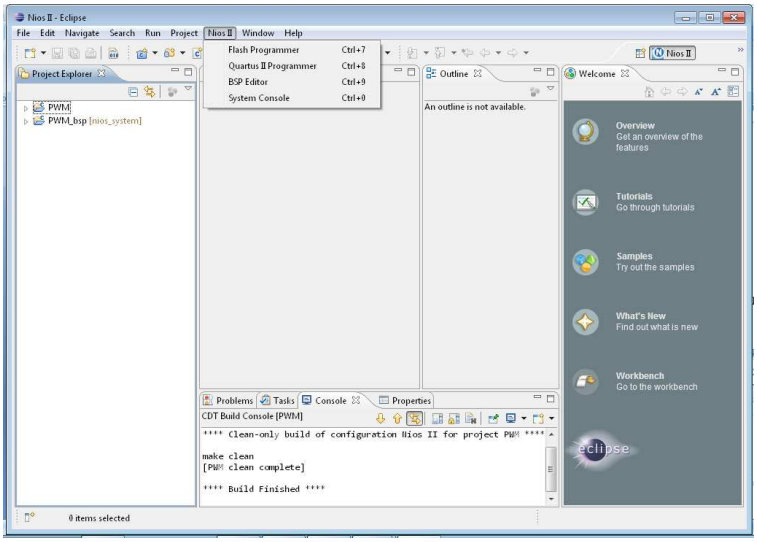
Por último hacemos Click en finish.

PROGRAMACION DE LA FPGA

Nota: Conectar la Tarjeta DE0 Nano al computador.

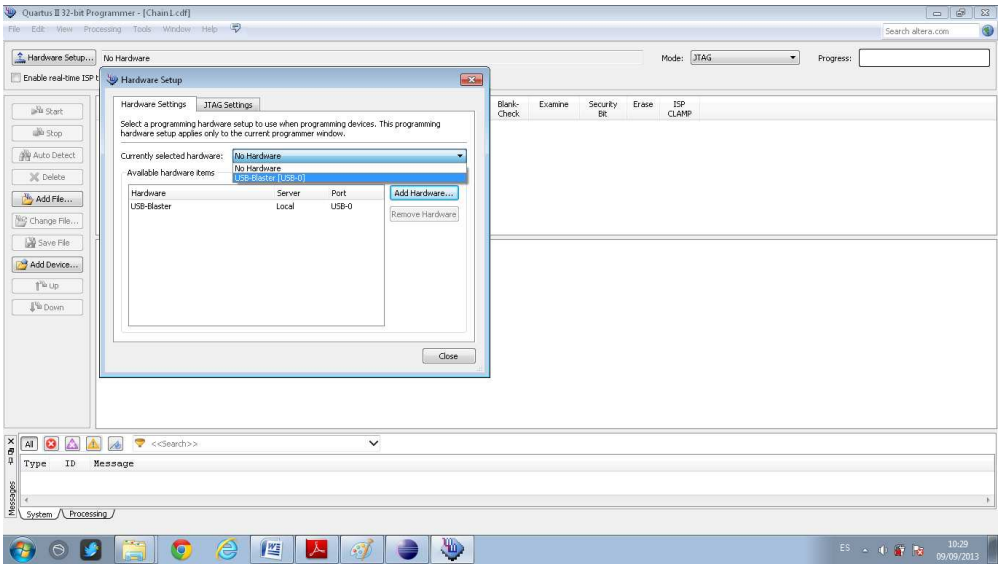
Antes de realizar código alguno, se debe realizar la programación de la maquina que se ha construido haciendo los siguientes pasos:

Menú Nios II – Quartus II Programmer.

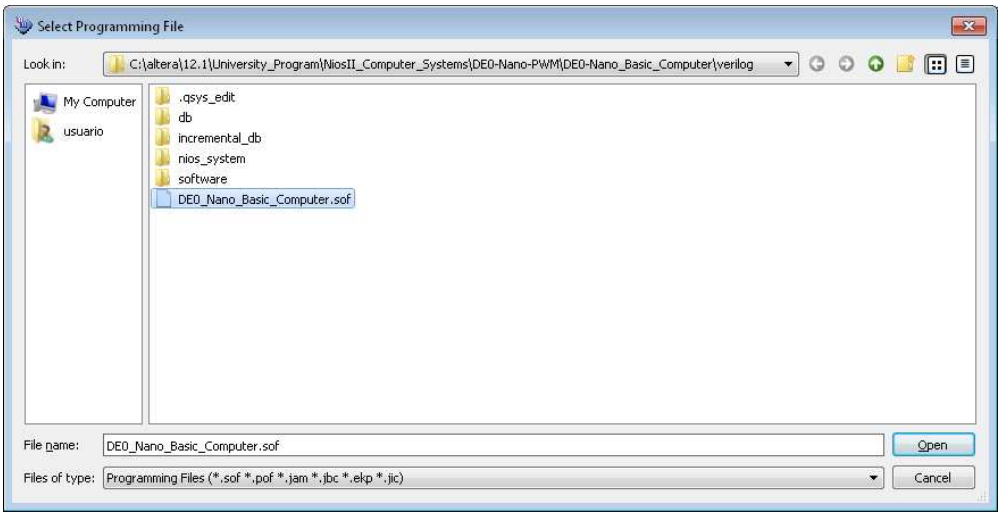


La ventana del programador del Quartus II aparece y realizamos los siguientes pasos.

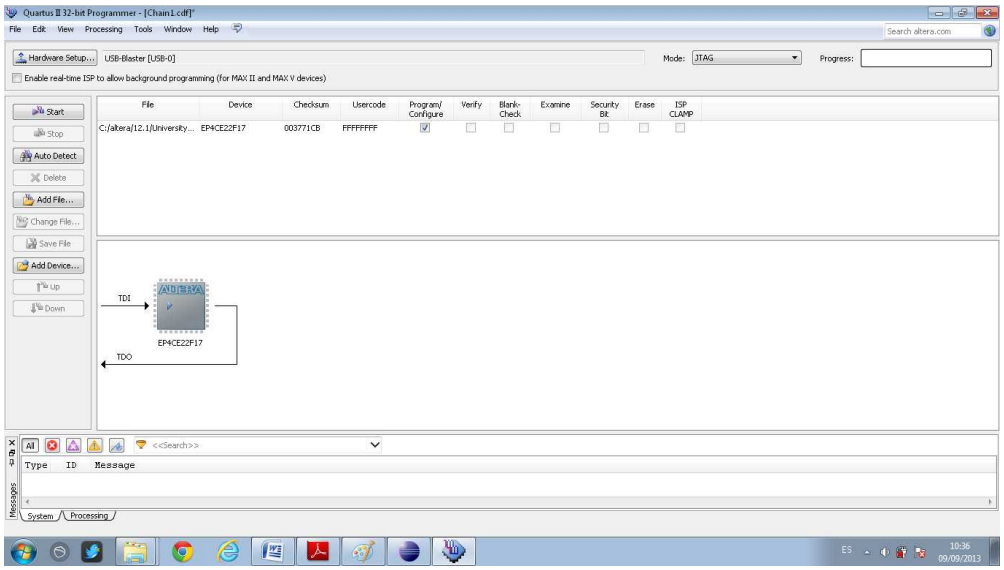
Hardware Setup – Currently Selected Harware – USB Blaster [USB 0] – click en close.



Una vez más nos ubicamos en la ventana principal del Programador de Quartus II, ahora se procede a añadir el archivo DE0_Nano_Basic_Computer.sof y se hace clic en Open.

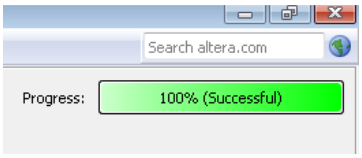


Una nueva vista nos presenta la ventana principal indicándonos que el archivo se ha cargado exitosamente.



Clic en START

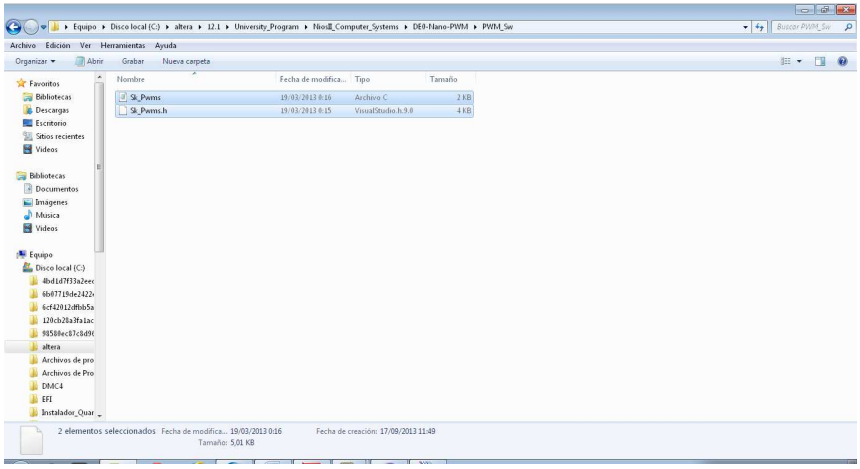
En la ventana superior derecha se mostrara un indicador del estado de la programación de la tarjeta.



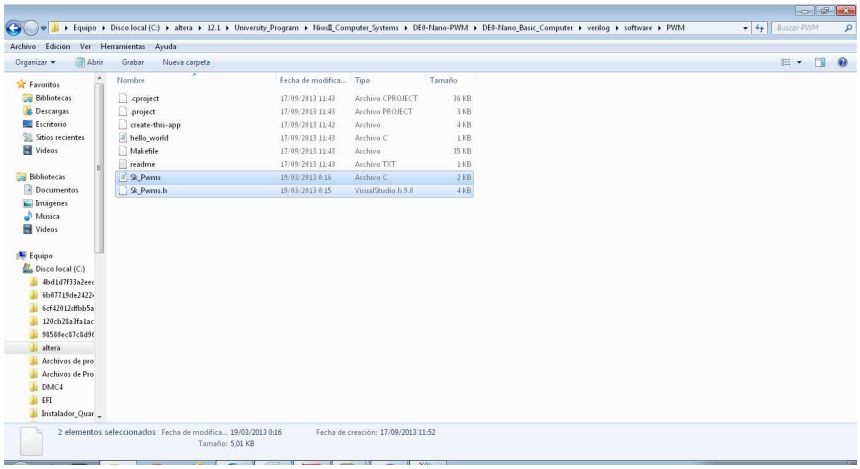
La tarjeta esta lista para ser programada en NIOS II.

Dentro de la carpeta de PWM existe un archivo de manejo de librerías para el control de PWM, es necesario copiarlas y ubicarlas desde y hacia la siguiente ruta denotada en la parte de abajo.

Origen de controlaldores de PWM.

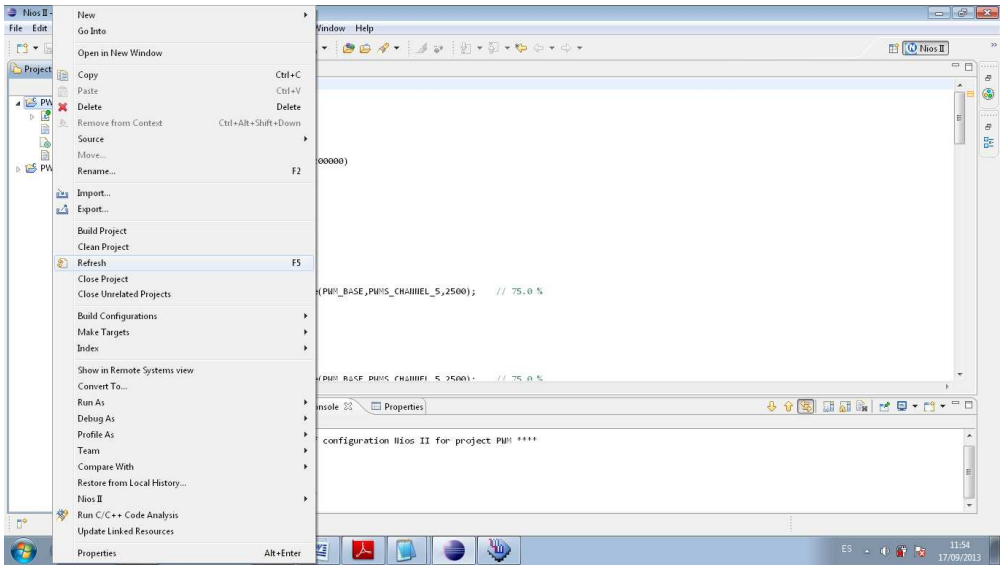


Destino de controladores de PWM



Dirección: C:\altera\12.1\University_Program\NiosII_Computer_Systems\DE0-Nano-PWM\DE0-Nano_Basic_Computer\verilog\software\PWM.

Se procede a dar un refresh, para que los archivos se actualizen y se ubiquen en nuestro directorio de trabajo de NIOS II.

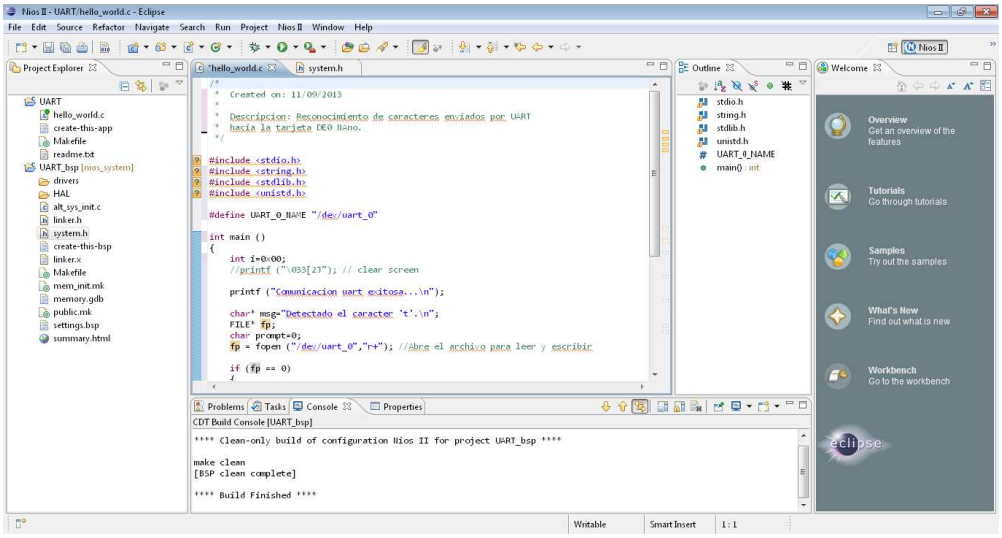


PROGRAMACION EN NIOS II DE ECLIPSE.

Codigo Control por PWM:

```
/*
 * principal.c
 *  Creado: 12/09/2013
 */
// Codigo para control de motorpor PWM

#include <stdio.h>
#include "system.h"
#include "Sk_Pwms.h"

void delay()
{
        int contador=0;
        while(contador!=200000)
        {
                contador++;
        }
}

int main(void)
{

        //Asignacion de puntero a banco de direccion
        volatile int * direccion = (int *) DIRECCION_BASE;
        void delay();
        //Mensaje por consola
    printf("PWM CONTROL DE MOTOR.....\n\n");
    //Inicio de PWM
    PWMS_Init((void*)PWMS_BASE,PWMS_IRQ_INTERRUPT_CONTROLLER_ID,PWMS_IRQ,10,2500);

    while(1)

    {
                //Dirección izquierda
            *(direccion)=2;             // izquierda

            //PWM 50% velocidad
            PWMS_SetDutyCycle(PWMS_BASE,PWMS_CHANNEL_0,1250);    // 50 %  velocidad
            delay();







            //Dirección derecha
            //*(direccion)=1;   //derecha

            //PWM 50% velocidad
            //PWMS_SetDutyCycle(PWMS_BASE,PWMS_CHANNEL_0,2500);    // 100 % velocidad
            //delay();


    }

}
```

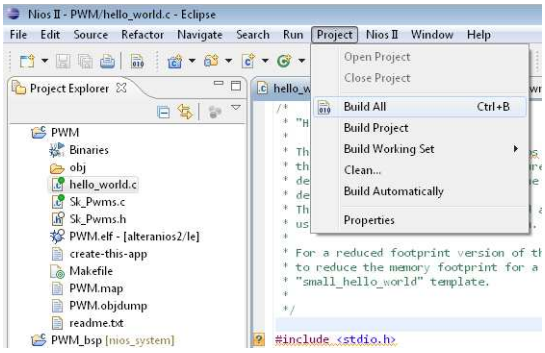
Copiar el código correspondiente en la ventana de desarrollo como se muestra en la siguiente figura.



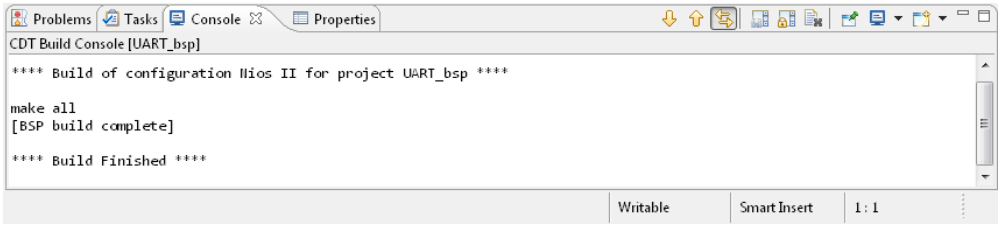
CONSTRUCCION DEL PROYECTO.

Es necesario el archivo de extensión .elf para su compilación y se deben realizar los siguientes pasos.

Project – Build All

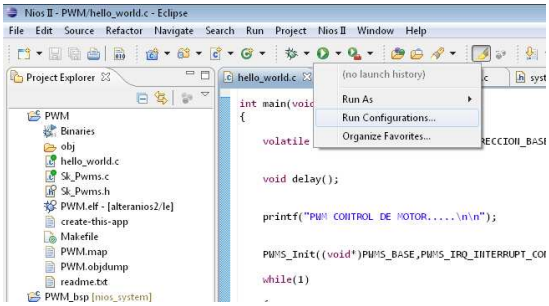


Entonces aparecerá un archivo de extensión .elf. y un mensaje como se muestra en la siguiente figura en el menú de console "Build Finished".



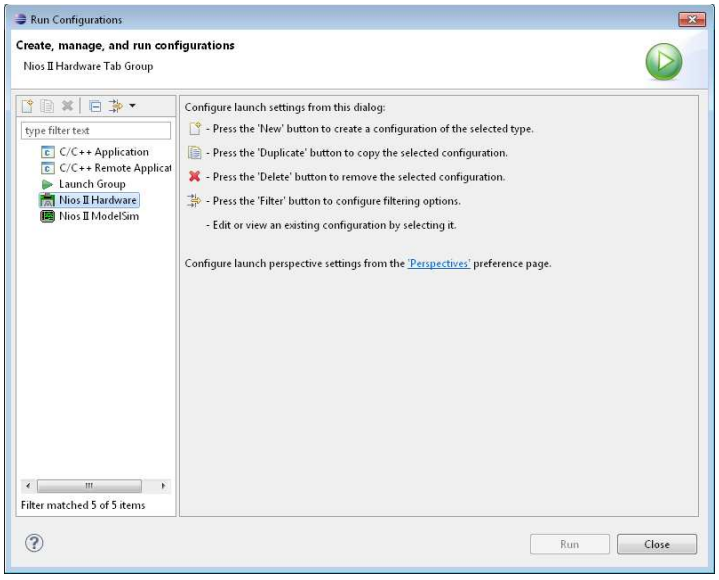
COMPILACION DEL CODIGO.

Para su respectiva compilación nos dirigimos al Run Configuration para realizar los cambios respectivos como se muestra en la siguiente figura.

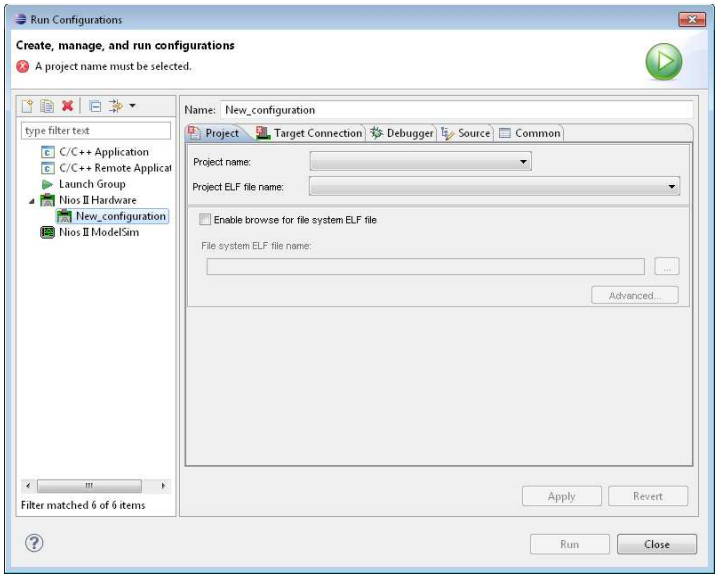


Nota: Observar la presencia del archivo PWM.elf, denota que el proyecto se ha compilado exitosamente.

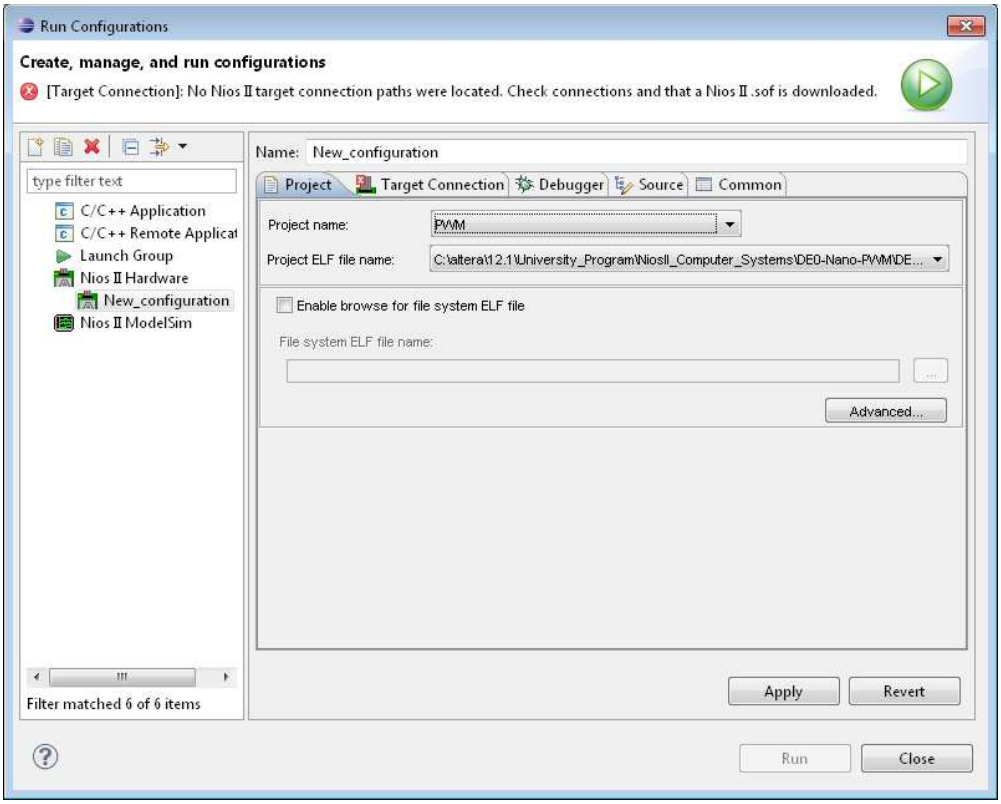
Una ventana aparece y nos dirigimos a seleccionar la opción de Nios II Hardware.



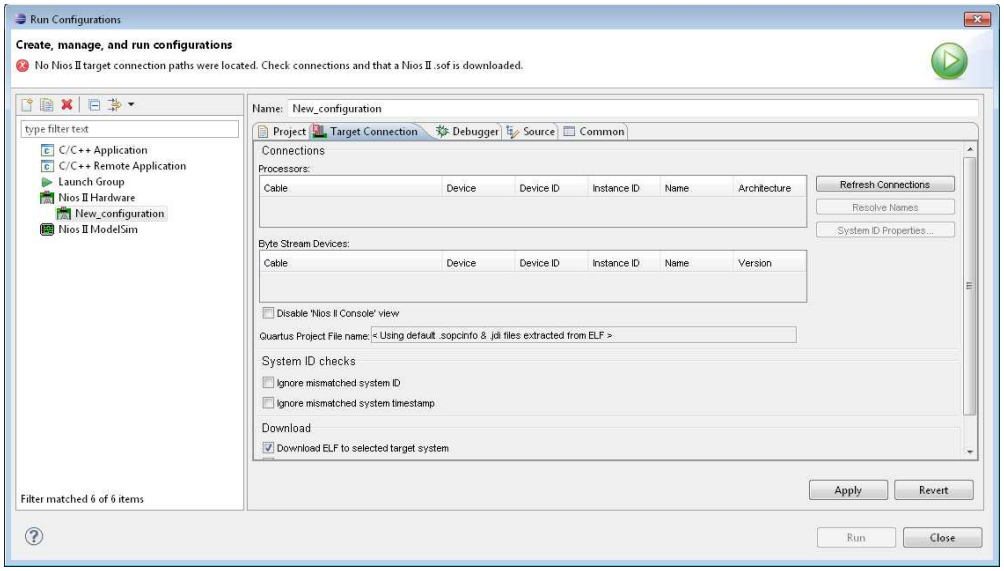
Entonces se mostrara una ventana de nombre Run Configuration.



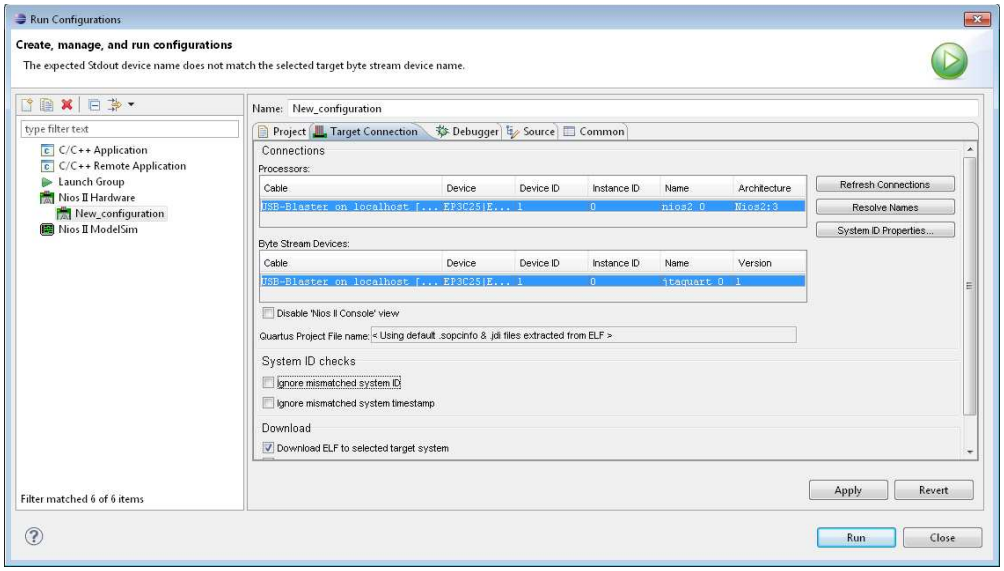
En el menú Project – Project name – Seleccionamos el nombre del proyecto.



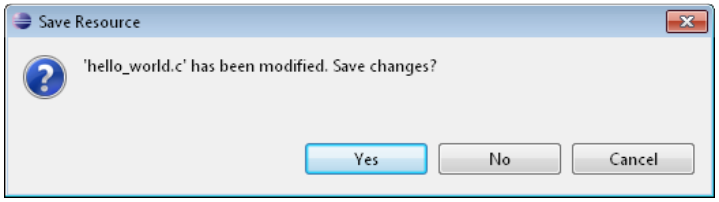
Menú Target Connection.



Hacemos clic en Refresh Connections, para verificar la presencia de la tarjeta DE0 Nano de Altera.



Finalmente se procede hacer clic en RUN.



Una ventana nos pide guardar los cambios realizados en el código, seleccionamos YES.

Un menú de Nios II Console se despliega y nos muestra el mensaje siguiente “”.