universidad del país vasco

euskal herriko unibertsitatea

Facultad de Informática
Informatika Fakultatea

TITULACIÓN: Ingeniería Técnica en Informática de Sistemas

Hardware basado en FPGA para síntesis de audio mediante muestras grabadas y control MIDI

Alumno/a:	**D./Dña. Imanol Gil Fernandez**
Director/a:	**D./Dña. Andoni Arruti Illarramendi**

Proyecto Fin de Carrera, julio de 2012

RESUMEN

El presente documento recoge la memoria de un proyecto fin de carrera presentado dentro del Departamento de Arquitectura y Tecnología de Computadores (ATC).

En este proyecto se ha desarrollado y construido una plataforma hardware que permite manejar mediante software diferentes dispositivos. Se partía de un PFC anterior el cual tenía preparados diferentes módulos en Quartus para poder trabajar con diferentes componentes como por ejemplo un teclado MIDI. Se ha utilizado el SoPC Builder y el Nios II para modificar el módulo principal del PFC del cual partimos para poder ampliar la gama y variedad de sistemas que se podrán desarrollar mas adelante.

El resultado ha sido una plataforma sobre la cual se pueden desarrollar múltiples sistemas utilizando los componentes incluidos (por ejemplo, interruptores y Leds de la placa DE2 y un teclado MIDI).

ÍNDICE

1.- INTRODUCCIÓN

1.1.- Motivación del proyecto

Han sido tres los principales motivos de la elección de este proyecto:

El primero de ellos, el gusto por las asignaturas de Laboratorio de Diseño Digital y Microcontroladores. Ambas asignaturas me han parecido de lo mas interesante de la carrera por lo que decidí buscar un Proyecto Fin de Carrera (PFC a partir de ahora) relacionado con dichas asignaturas.

Otro de los motivos de la elección ha sido que desde hace unos años me ha venido interesando mucho el diseño de aplicaciones del tipo Hardware/Software codesign. Siempre me ha gustado el mundo de la electrónica y el hecho de andar con Hardware y que la aplicación no se trate solo de implementar software hace que me resulte mas entretenido y que por ello trabaje mas y mejor en el proyecto.

El último motivo que me ha llevado a realizar este proyecto a sido que cada día los proyectos que se realizan son mas variados y complejos por lo que aprender a utilizar nuevas tecnologías y herramientas para el desarrollo de aplicaciones puede que resulte útil y enriquecedor no solo para mi satisfacción personal sino para mi futuro laboral. Ampliar los conocimientos siempre es bueno puesto que podrá abrirte nuevas puertas por el camino.

1.2.- Organización de la memoria

Siguiendo a esta introducción, en el capítulo 2 presentaremos el Documento de Objetivos del Proyecto. En el capítulo 3 se mostrará una breve descripción de las tecnologías y herramientas utilizadas a lo largo de este proyecto. Después de esta breve descripción, en el capítulo 4 situaremos el punto de partida para el desarrollo del PFC actual, para lo cual explicaremos un poco mas al detalle el funcionamiento de los módulos del proyecto anterior y los cambios que realizaremos en el. A continuación, en el capítulo 5, comentaremos que pasos se han dado y como se ha desarrollado la aplicación final de este proyecto. Para finalizar, dedicaremos un capítulo a la reflexión y a las conclusiones a las que se a llegado en la finalización del mismo y otro a la bibliografía utilizada en dicho PFC, estos serán los capítulos 6 y 7 respectivamente.

"En el CD adjunto con la memoria se puede encontrar todo el código y archivos generados a lo largo de este PFC. Desde los ficheros generados por el Quartus II hasta los generados por el SoPC Builder o Nios II."

2.- DOCUMENTO DE OBJETIVOS DEL PROYECTO

2.1.- Objetivos

En este proyecto se persiguen dos objetivos principales. El primero de ellos es aprender a desarrollar aplicaciones Hardware/Software codesign (H/S-c en adelante) combinando para ello las diferentes herramientas y tecnologías de Quartus II, SoPC Builder y Nios II. Para llevar a cabo este objetivo, aprenderemos a utilizar dichas tecnologías empezando desde cero, mediante la lectura de manuales y la realización de diferentes ejercicios y tutoriales. El segundo es desarrollar una plataforma hardware que permita manejar mediante software diferentes dispositivos para poder desarrollar diferentes sistemas. Para poder realizar este objetivo, utilizaremos las tecnologías mencionadas con anterioridad.

Una vez adquiridos esos conocimientos y para comprobar que los objetivos principales de este proyecto se han cumplido, como objetivo secundario del proyecto, realizaremos una aplicación software en la cual sustituiremos el módulo principal del proyecto ya existente, el cual está implementado en VHDL mediante Quartus II (Hardware), por uno nuevo desarrollado haciendo uso de SoPC Builder y Nios II (Software). Se intentará que la nueva aplicación disponga de la misma funcionalidad que el PFC en el cual esta basado. De esta manera mostraremos un ejemplo del tipo de sistemas que se pueden desarrollar sobre la plataforma creada.

2.2.- Método de trabajo

En el caso de que surgiera un problema o duda a resolver con urgencia se realizará una reunión en el momento. De no ser necesario se concertarán reuniones cada vez que se realice algún paso importante en el proyecto. En las reuniones se expondrá el trabajo realizado, se discutirán posibles mejoras y se resolverán dudas y problemas.

En lo referido a las entregas se adoptará la estrategia de terminar el trabajo en el 75% del tiempo disponible, para que de esta forma se puedan solucionar imprevistos y resolver problemas de última hora que seguro surgirán.

Para realizar los objetivos descritos en el apartado anterior, he decidido basar la realización de este PFC en tres sencillas etapas:

2.2.1.- Situación inicial:

En esta primera etapa se ha situado el punto de partida del proyecto. Para ello se ha estudiado el proyecto previo en el cual esta basado el actual. Una vez comprendido se han establecido los módulos que se van a modificar y con que tecnologías y herramientas se van a realizar dichas modificaciones.

2.2.2.- Formación y aprendizaje:

Una vez decididas las tecnologías y herramientas a utilizar y al ver que no se tiene conocimiento alguno de ellas, toca formarse para poder realizar la aplicación anteriormente descrita.

2.2.3.- Desarrollo:

Para poder finalizar el proyecto y una vez adquiridos los conceptos y conocimientos necesarios para la realización de aplicaciones utilizando estas nuevas herramientas y tecnologías, solo queda desarrollar y dejar preparada la plataforma para posteriormente intentar desarrollar una aplicación software de funcionamiento similar al anterior..

2.3.- Alcance

A continuación se analizarán las tareas principales que se llevarán a cabo en el proyecto, así como la duración estimada de las mismas.

El proyecto de fin de carrera de la Ingeniería Técnica en Informática de Sistemas tiene una carga lectiva de 6 créditos, lo que equivale a 120 horas. El proyecto se puede dividir en 3 tareas, que a su vez se dividirán en múltiples subtareas que se analizarán más adelante.

TAREA	TIEMPO
Formación en el uso de la tecnología SoPC Builder	25 horas
Formación en el uso de la tecnología Nios II	25 horas
Desarrollo de la aplicación	40 horas
Documentación del Proyecto	40 horas
TOTAL	**130 horas**

2.3.1.- Formación en el uso de la tecnología SoPC Builder

Parte de la aplicación a desarrollar se va a realizar con SoPC Builder. Para poder llevarla a cabo se deben conocer las características y posibilidades de la tecnología, los entornos de desarrollo, el funcionamiento de los diferentes entornos, etc.

Una vez asentadas las bases de la tecnología se realizarán pruebas y pequeñas aplicaciones que sirvan a modo de práctica para ir dominando la tecnología.

El apartado de formación se ha dividido en 3 tareas más concretas requiriendo unas 25 horas en total:

- Búsqueda de información acerca de SoPC Builder (1 hora).
- Lectura y comprensión de los diferentes manuales encontrados (12 horas).
- Pruebas y realización de pequeñas aplicaciones (12 horas).

2.3.2.- Formación en el uso de la tecnología Nios II

El resto de la aplicación a desarrollar se va a realizar con Nios II. Al igual que en el caso del SoPC Builder, para poder llevarla a cabo se deben conocer las características y posibilidades de la tecnología, los entornos de desarrollo, el funcionamiento de los diferentes entornos, etc.

Una vez asentadas las bases de la tecnología se realizarán pruebas y pequeñas aplicaciones que sirvan a modo de práctica para ir dominando la tecnología.

El apartado de formación se ha dividido en 3 tareas más concretas requiriendo unas 25 horas en total:

• Búsqueda de información acerca de Nios II (1 hora).

• Búsqueda y prueba de entornos de desarrollo (2 horas).

• Lectura y comprensión de los diferentes manuales encontrados (11 horas).

• Pruebas y realización de pequeñas aplicaciones (11 horas).

2.3.3. Desarrollo de la aplicación

Esta tarea consiste en la creación de la aplicación. Se comenzará con una fase de análisis en la que se analizará el PFC anterior y sus módulos para comprender el funcionamiento interno de la aplicación a desarrollar.

A continuación se realizará el diseño de la aplicación. En el cual se decidirá como será el componente que sustituiremos en la aplicación. Una vez diseñado se pasará a implementar y probar la aplicación final.

Se ha estimado que se requerirán unas 40 horas divididas de la siguiente manera:

• Análisis de la aplicación (8 horas).

• Diseño (6 horas).

• Implementación (20 horas).

• Pruebas (6 horas).

2.3.4.- Documentación del proyecto

Además de la documentación del proyecto, en esta tarea se va a incluir lo referido a la presentación del proyecto. La documentación se va a realizar de manera continuada a lo largo del proyecto y al final habrá que organizar y componer la memoria. Se tendrá que disponer de tiempo para realizar correcciones y cambios.

Para la documentación se estima que se necesitarán unas 40 horas dividas de la siguiente forma:

- Borrador de la memoria (15 horas).
- Primera versión de la memoria (6 horas).
- Correcciones y cambios (6 horas).
- Impresión de la memoria y realización de las copias de los CD's con la aplicación (1 hora).
- Preparación de la defensa del proyecto (9 horas).
- Ensayos de la exposición (3 horas).

2.4.- Plan de contingencia y factibilidad

Uno de los riesgos que más impacto puede causar en el proyecto puede ser el derivado de las diferentes tecnologías a utilizar dado que no se dispone de conocimiento previo alguno sobre ellas. Si con los medios disponibles (manuales, ejercicios, ejemplos...) no se consigue llegar al nivel de conocimiento deseado, impidiendo el avance del proyecto, será necesaria la ayuda de algún profesor

con experiencia en dichas tecnologías. Al tratarse de tecnologías completamente desconocidas se prevé que surja algún imprevisto a lo largo del desarrollo del proyecto con su correspondiente demora temporal .

En cuanto a recursos temporales, no se dispone de mucho mas tiempo del previsto por lo que habrá que conseguir cumplir plazos y llevar las tareas planificadas al día para poder reservar ese tiempo extra del que se dispone para posibles imprevistos o mejoras.

Otro de los riesgos importantes que pueden surgir (en lo que al objetivo secundario se refiere) a lo largo de la realización de la aplicación es que el paso de una aplicación de "ejecución paralela" (hardware) a "ejecución secuencial" (software) resulte mas complicado de lo esperado y haga que no se consiga el objetivo secundario propuesto en este proyecto. Puesto que esta complicación puede que haga que se alargue mas de la cuenta el tiempo de realización agotando el tiempo del que se dispone y haciendo el proyecto inviable.

3.- ANÁLISIS DE HERRAMIENTAS Y TECNOLOGÍAS

A continuación explicaremos brevemente las herramientas y tecnologías utilizadas durante el desarrollo del PFC:

3.1.- Altera DE2 (Development and Education Board)

El propósito de esta placa es proporcionar el material ideal para el aprendizaje de la lógica digital, y FPGAs. La tarjeta ofrece un rico conjunto de características que lo hacen adecuado para poder desarrollar una gran gama de diseños, desde simples circuitos, hasta sofisticados y complejos sistemas digitales y multimedia. Aparte de la placa en si, Altera proporciona un conjunto de materiales de apoyo para la placa DE2, que incluye tutoriales, ejercicios de laboratorio "ready-to-teach" y demostraciones ilustradas.

Especificaciones Técnicas de la placa:

FPGA

• FPGA Cyclone II EP2C35F672C6

Dispositivos I/O

• USB Blaster para la configuración de la FPGA

• Puertos 10/100 Ethernet, RS-232 e Infrarrojo.
• Salida de Video (VGA 10-bit DAC)
• Entrada de Video (NTSC/PAL/Multi-formato)
• USB 2.0 (tipo A y tipo B)
• Conector de ratón o teclado PS/2
• CODEC de audio de 24 bits (Line-in, Line-out, entrada de micrófono)
• Cabezales de expansión (76 pines)

Memoria

• SDRAM de 8-MB, SRAM de 512-KB, Flash de 4-MB

• Ranura para tarjeta de memoria SD

Interruptores, LEDs, Displays, y Relojes

• 18 interruptores de palanca

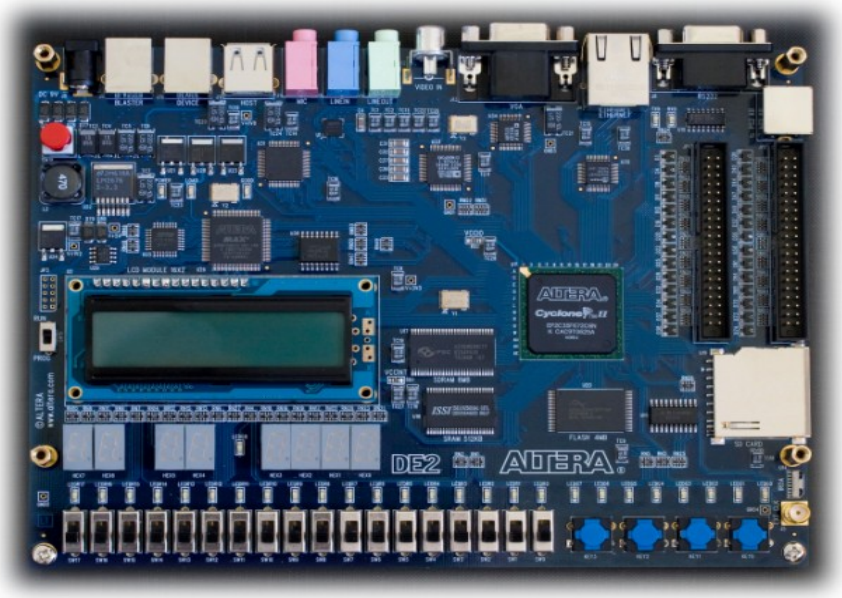
• 4 pulsadores de boton

• 18 LEDs rojos y 9 LEDs verdes

• 8 Displays de 7 segmentos

• Display LCD de 16 x 2

• Osciladores de 27-MHz y 50-MHz para fuentes de reloj



3.2.- Teclado Oxygen 8 v2 (MIDI)

Oxygen 8 v2 es un completo teclado controlador MIDI dotado de un mecanismo de tecla excelente y de ocho botones con los que podremos controlar cualquier parámetro MIDI de nuestros dispositivos hardware o software. Entre las funciones incorporadas, se incluye el soporte completo para mensajes MIDI con asignación de canal para cada controlador, además de 6 controles de transporte que se pueden reasignar a cualquier parámetro MIDI. Oxygen 8 v2 también tiene 10 memorias no volátiles, y es compatible con Enigma, el software de edición gratuito para ordenador que permite almacenar, recuperar y gestionar un número ilimitado de patches.

Como la placa DE2 no dispone de un puerto de salida MIDI para poder realizar la conexión con el teclado, utilizaremos un circuito capaz de transformar la señal procedente del puerto de salida MIDI del teclado en niveles de tensión de 5 o 0 voltios. De esta manera, conectando dicho circuito a uno de los cabezales de extensión de la placa podremos reconocer los valores 1 (5 voltios) o 0 (0 voltios) y así trabajar con ellos.

3.3.- Quartus II

Quartus II es una herramienta de software producida por Altera para el análisis y la síntesis de diseños realizados en HDL (*Hardware Description Language)*. Permite al desarrollador compilar sus diseños, realizar análisis temporales, examinar diagramas RTL y configurar el dispositivo de destino con el programador.

Para la realización de este proyecto se ha utilizado la versión gratuita de Quartus II, la denominada "Web Edition". Esta edición permite la compilación y la programación de un número limitado de dispositivos Altera. La familia de FPGAs de bajo coste Cyclone está soportada por esta edición (la placa DE2 utilizada contiene el dispositivo Cyclone II EP2C35F672C6), por lo que los pequeños desarrolladores no tendrán problemas por el coste del desarrollo de software.

Se requiere un registro de licencia para utilizar la Edición Web de Quartus II, la cual es gratuita y puede ser renovada ilimitadamente. Eso si, se debe pagar una licencia para utilizar todas las funciones de la aplicación.

3.4.- VHDL

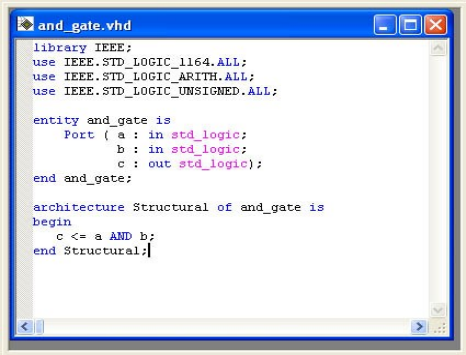
VHDL es el acrónimo que representa la combinación de VHSIC y HDL, donde VHSIC es el acrónimo de *Very High Speed Integrated Circuit*y HDL es a su vez el acrónimo de *Hardware Description Language*.

Se trata de un lenguaje de descripción de hardware, definido por el IEEE (*Institute of Electrical and Electronics Engineers*) (ANSI/IEEE 1076-1993), esto significa que mediante él se puede describir la forma de comportarse de un circuito electrónico. El comportamiento puede ser llevado a algún dispositivo que dispondrá de sus propios componentes con los que lograr ese comportamiento deseado. La forma de comportarse es independiente del hardware donde se implementará.

Aunque puede ser usado de forma general para describir cualquier circuito se usa principalmente para programar PLD (*Programable Logic Device*), FPGA (*Field Programmable Gate Array*), ASIC y similares.

Sus ventajas son:

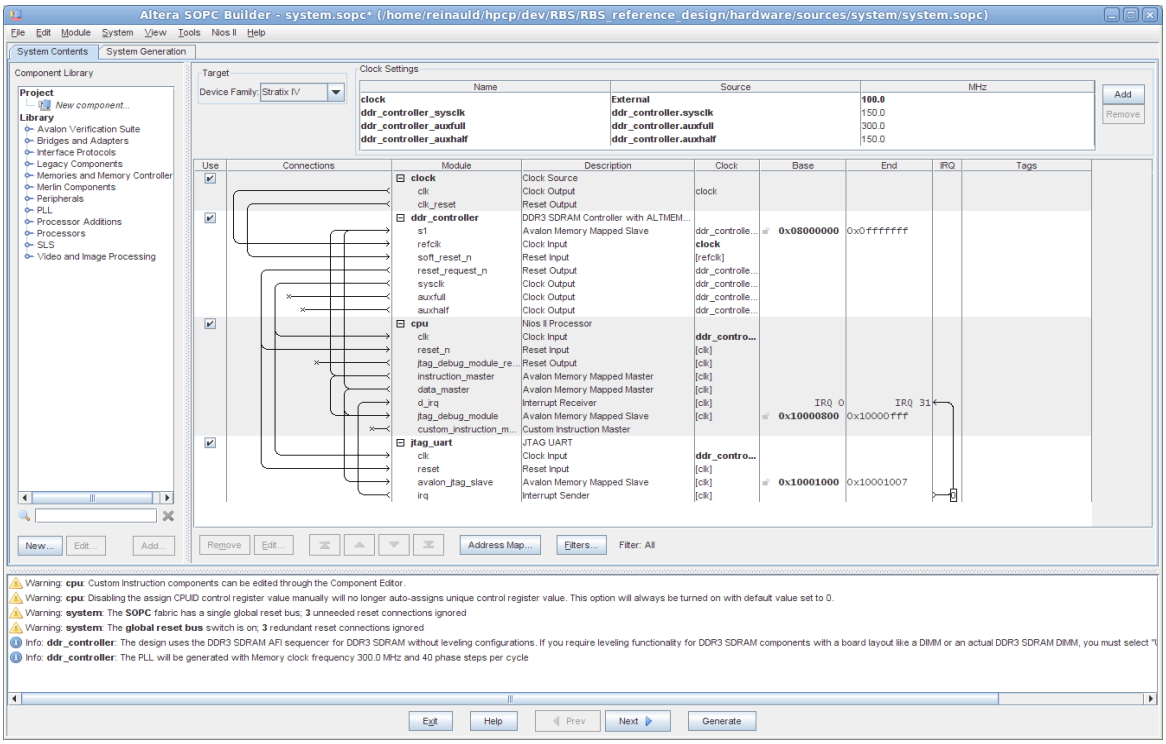
- Una disponibilidad pública
- Independencia de dispositivos y fabricantes
- Reutilización
- Diseño jerárquico



3.5.- SoPC Builder

SoPC Builder (System on a Programmable Chip Builder) es un software hecho por Altera, que automatiza la conexión entre componentes de software y hardware para crear un sistema completo que se ejecuta en cualquiera de sus diversas FPGA.

A su vez, incorpora una biblioteca de componentes predefinidos, (incluyendo su buque insignia que es el procesador Nios II, controladores de memoria, interfaces y periféricos) y una interfaz que nos permite la incorporación de otros personalizados. Las interconexiones se realizan mediante el bus Avalon. El arbitraje del bus, la coincidencia de ancho de bus, e incluso el reloj de controlador de dominio se manejan de forma automática cuando SoPC Builder genera el sistema. Una simple interfaz gráfica de usuario es lo único que utilizaremos tanto para configurar los componentes (que a menudo tienen muchas opciones) como para especificar la topología de bus.

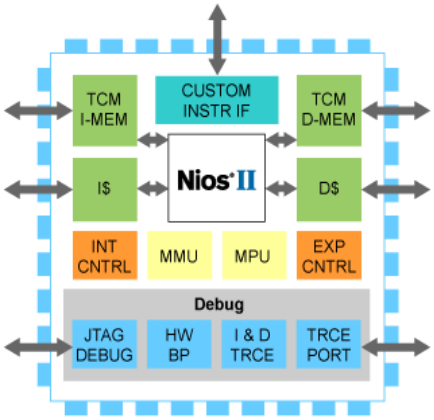


El resultado es un sistema "virtual", el cual puede ser conectado con el mundo exterior a través de pines programables de la FPGA o conectado internamente a otros componentes software. Los Pines de la FPGA se enrutan a los conectores, como para el bus PCI o DDR, o (como ocurre a menudo en los sistemas empotrados) a otros chips montados en el mismo PCB *(Process Control Block)*.

La utilización de recursos de una FPGA que aloja un sistema de SoPC Builder es muy modesto para los estándares modernos. La lista de dispositivos FPGA que soportan a los sistemas SoPC incluye casi todas las FPGAs de Altera (e incluso algunos CPLDs).

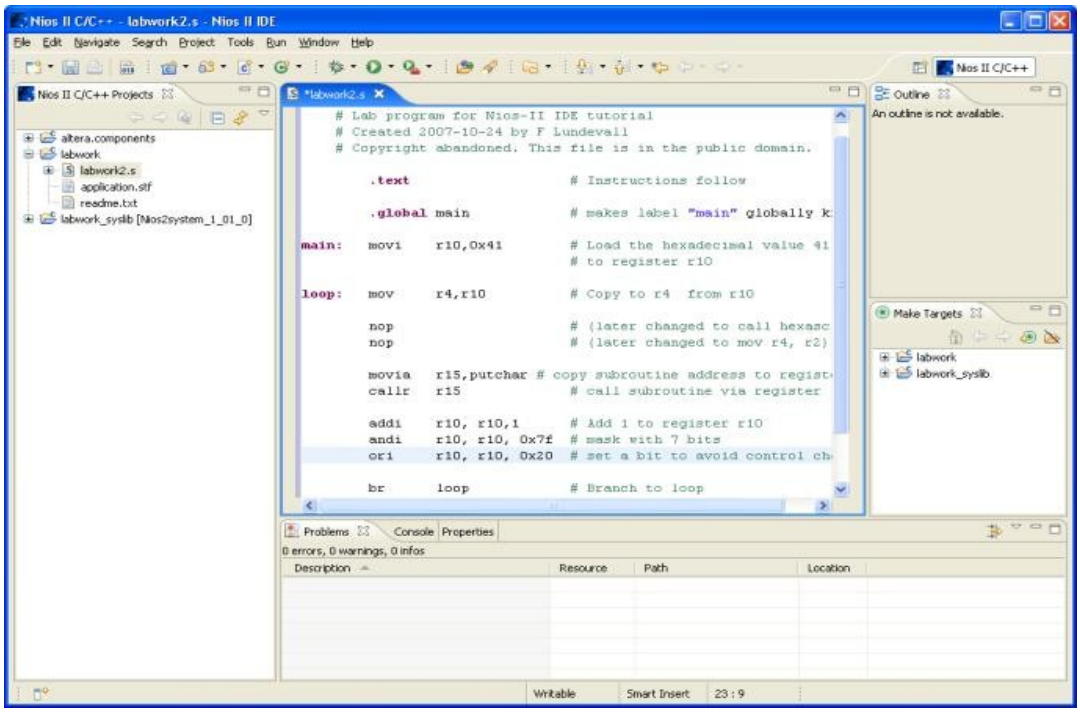
3.6.- Nios II

Es un procesador embebido de arquitectura de 32 bits diseñada específicamente para la familia de FPGA-s de Altera. Nios II incorpora muchas mejoras sobre la arquitectura Nios original, por lo que es más adecuado para desarrollar una amplia gama de aplicaciones empotradas, desde DSP (*Proceso Digital de Señales*) hasta sistemas de control.



Nios II es el procesador más versátil del mundo, según Gartner Research, es el procesador más utilizado en la industria de las FPGA. Este procesador ofrece una flexibilidad sin precedentes para sus necesidades como los costes razonables, tiempo real, seguridad crítica (DO-254) y ASIC optimizado.

El programa utilizado para incormporar y programar el Nios II en nuestro proyecto es el "Nios II Software Build Tools for Eclipse and Integrated Development Environment (IDE) Support". Los usuarios pueden crear rápida y fácilmente aplicaciones de software para sus sistemas con procesadores Nios II. Al igual que en las tecnologías de Altera anteriormente utilizadas podremos disfrutar de una completa documentación, base de datos de conocimientos, tutoriales sobre como usar el Build Tools for Eclipse o el IDE, y múltiples ejemplos de software.



Como se puede observar la interfaz es similar a la utilizada para programar con Eclipse en Java o en C++. De hecho se puede programar el Nios II tanto en a bajo nivel (Ensamblador-Lenguaje Maquina) como en alto nivel (C++) facilitando así la realización y el desarrollo de diferentes tipos de aplicaciones según se desee.

4.- SITUACIÓN INICIAL

Para poder modificar una parte del proyecto anterior es necesario conocer a la perfección cada una de las partes de dicho proyecto. A continuación se muestra el funcionamiento y descripción de los diferentes componentes del proyecto.

4.1.- Funcionamiento del sistema

Se trata de un sistema digital para la grabación y reproducción de audio a través de un teclado MIDI. Dicho sistema tiene dos modos de funcionamiento. Modo grabación y modo reproducción.

• En modo **grabación** captura y almacena en memoria el sonido proveniente de un micrófono cada vez que se pulse una tecla. Este modo esta activado cuando el primer conmutador de la tarjeta DE2 esté activo, es decir, cuando esté a 1. En los displays de siete segmentos de la tarjeta aparece escrita la palabra REC.

La grabación se inicia cuando, una vez se haya pulsado una tecla del teclado MIDI, la misma se suelta. El tiempo de grabación para cada tecla es de aproximadamente un segundo, así que para ver la duración de la grabación, los 8 leds verdes de la tarjeta DE2 se van encendiendo de forma cíclica hasta el fin de la grabación. Cuando los 8 leds se vuelvan a apagar, esto indicará el fin de la grabación.

A cada tecla le corresponde un sonido distinto, por lo que si queremos grabar sonidos distintos para distintas teclas, deberemos realizar esta operación tantas veces como teclas queramos grabar. Existe la posibilidad de volver a grabar un sonido para una tecla.

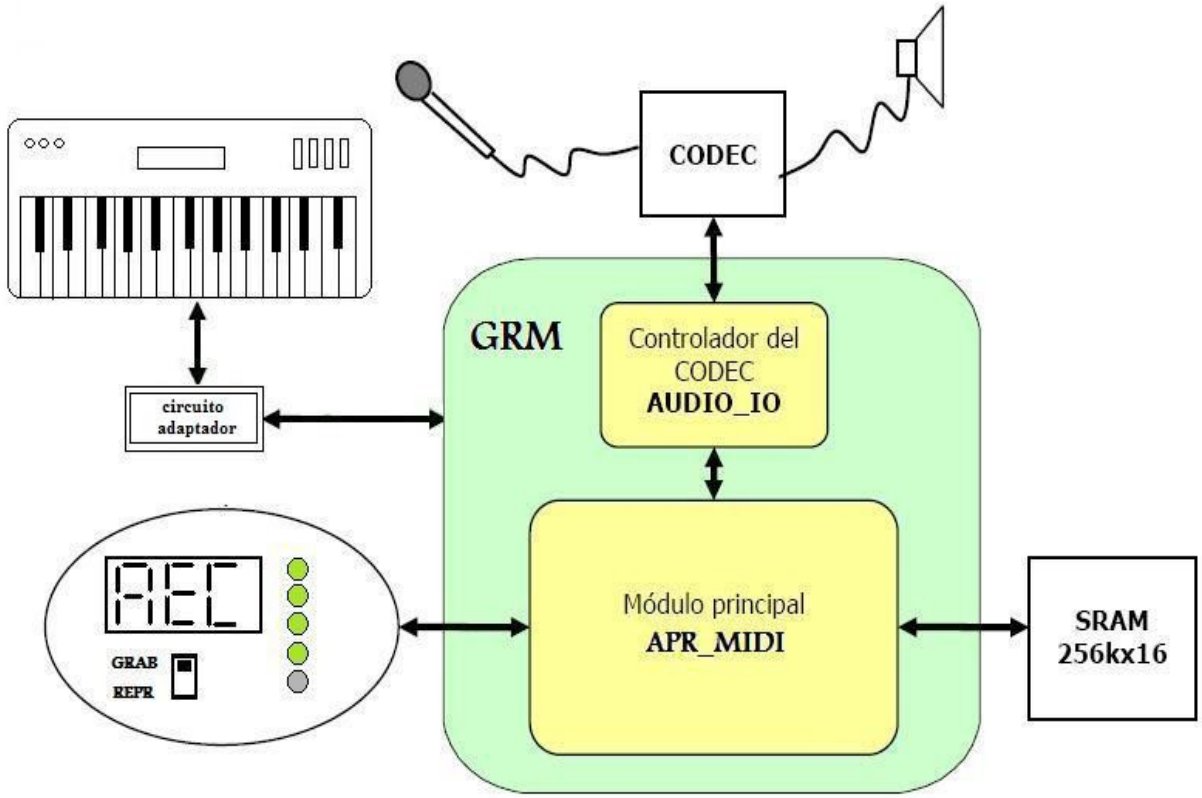
• En modo **reproducción** cada vez que se pulsa una tecla se oirá mediante un altavoz el sonido almacenado en memoria. Este modo estará activo cuando el primer conmutador de la tarjeta DE2 esté inactivo, es decir, cuando esté a 0. En los displays de siete segmentos de la tarjeta aparecerá escrita la palabra PLAY.

La reproducción se iniciará cuando se pulse una tecla del teclado MIDI. El sonido seguirá reproduciéndose mientras la tecla este pulsada y no se suelte. No existe la posibilidad de pulsar dos teclas a la vez.

El sonido es una señal analógica y para poder ser tratado de forma digital son necesarios dispositivos que lo transformen adecuadamente. En concreto, en la grabación es necesario convertir la señal procedente del micrófono de analógica a digital y, en la reproducción hay que reconvertirla de nuevo a analógica para poder enviarla al altavoz. El encargado de dicha transformación es el CODEC configurable disponible en la placa DE2, el WM8731.

4.2.- Estructura general del sistema a diseñar

De ahora en adelante denominaremos al sistema a diseñar **GRM** (Grabador/Reproductor MIDI) y de cara al diseño lo vamos a dividir en dos partes.



• El módulo que denominaremos **AUDIO_IO** se encargará de controlar el CODEC. Por un lado, lo configurará durante la inicialización del sistema y, por otro, intercambiará con él los datos de audio, recogiendo las muestras de audio procedentes del micrófono y entregando las muestras que deben enviarse al altavoz.

• El módulo que denominaremos **APR_MIDI** controlará el funcionamiento de todo el sistema, encargándose de:

▪ Interpretar las señales procedentes de los conmutadores y del teclado MIDI (se utilizará el circuito creado para ese fin).

▪ En modo grabación, almacenar en una memoria las muestras de audio procedentes de AUDIO_IO, y mostrar en el display la palabra REC.

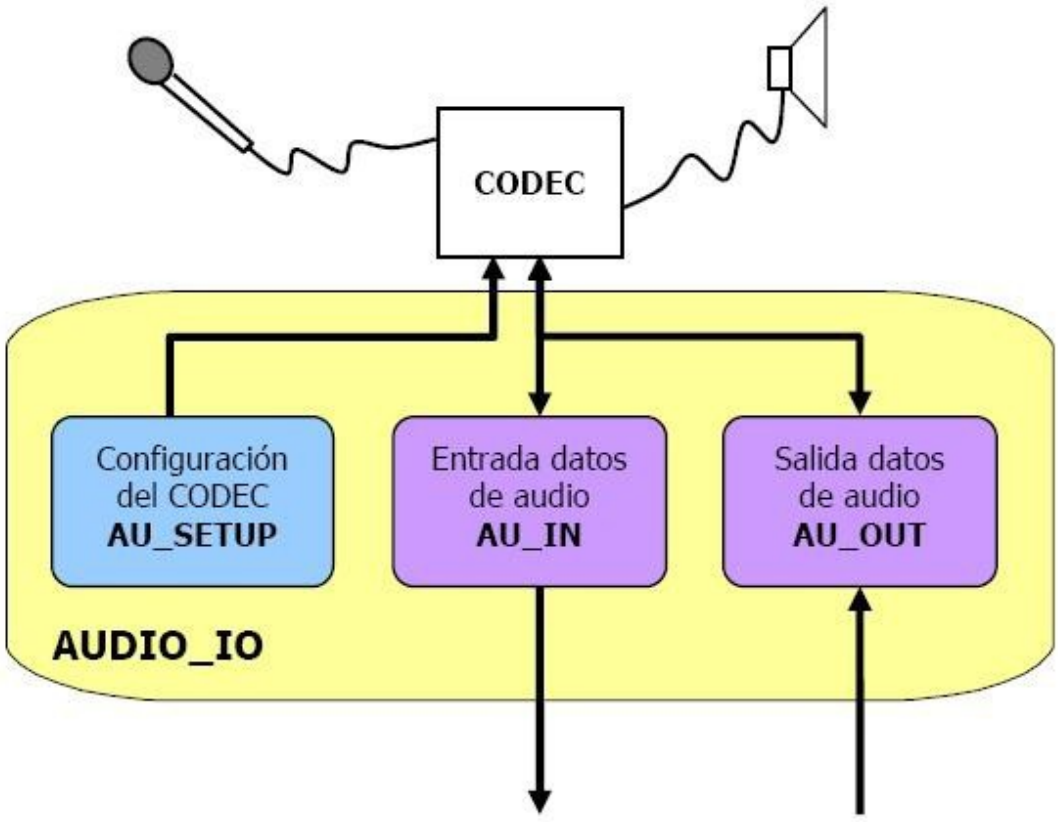
▪ En modo reproducción, leer las muestras de la memoria y entregárselas a AUDIO_IO, y mostrar en el display la palabra PLAY.

4.3.- Módulo AUDIO_IO

Vamos a construir este módulo mediante tres submódulos. Los submódulos se explican con detalle más adelante, y son:

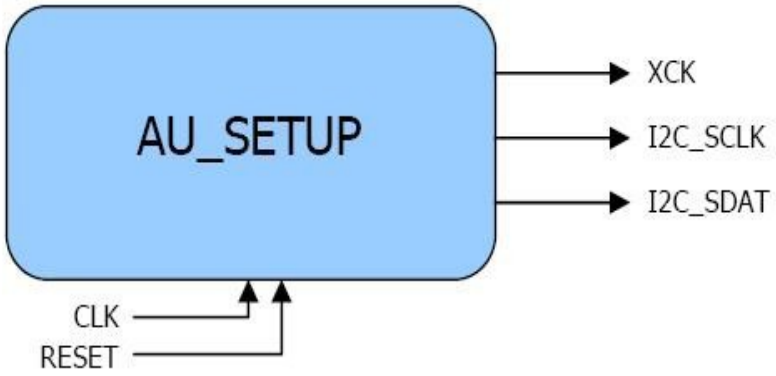
- **AU_SETUP**, que se encarga de configurar el CODEC en la inicialización del sistema.
- **AU_IN**, que se encarga de recibir datos del CODEC.
- **AU_OUT**, que se encarga de enviar datos al CODEC.

El esquema se muestra en la figura siguiente:



4.3.1.- Módulo AU_SETUP

Como ya se ha comentado antes, este módulo se encarga de configurar el CODEC durante la inicialización del sistema y se dará hecho, por lo que sólo se muestra a continuación su esquema:



Salidas:

• **XCK:** señal de reloj para el CODEC.

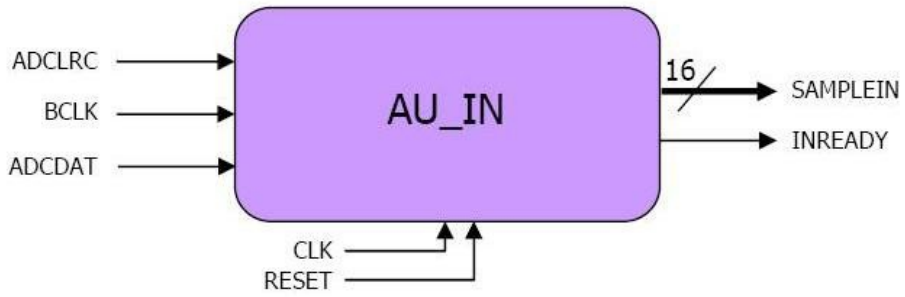
• **I2C_SCLK:** línea de reloj del bus I2C.

• **I2C_SDAT:** línea de datos del bus I2C.

4.3.2.- Módulo AU_IN

Este módulo se encarga de recoger las muestras entregadas por el CODEC, por la línea serie ADCDAT. Para ello analiza las señales ADCLRC y BCLK para capturar los 16 bits de uno de los canales (por ejemplo el izquierdo). Una vez recogido el dato, lo entrega en la salida SAMPLEIN.

El esquema del módulo es el siguiente:



Entradas:

• **ADCLRC**: indica el comienzo del dato procedente del CODEC para cada canal (flanco de subida: left, flanco de bajada: right).

• **BCLK**: el flanco de subida indica el punto medio de un bit.

• **ADCDAT**: entrada serie por la que llegan las muestras desde el CODEC.

Salidas:

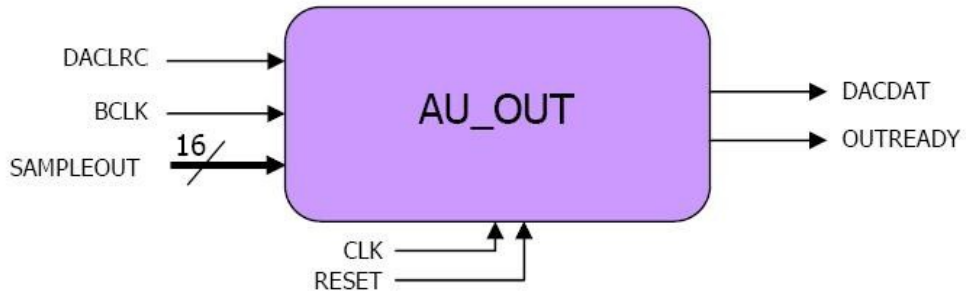
• **SAMPLEIN**: muestra de 16 bits proveniente del CODEC.

• **INREADY**: se activa durante un ciclo de reloj para indicar que se acaba de recibir una muestra.

4.3.3.- *Módulo AU_OUT*

Este módulo se encarga de entregar las muestras al CODEC, por la línea serie DACDAT, para ello analiza las señales ADCLRC y BCLK para poner en la línea DACDAT los 16 bits de la muestra SAMPLEOUT, en el momento adecuado.

El esquema del módulo es el siguiente:



Entradas:

• **DACLRC**: indica cuando iniciar el envio del dato hacia el CODEC para cada canal (flanco de subida: left, flanco de bajada: right).

• **BCLK**: el flanco de bajada indica cuando colocar el bit en la linea de salida.

• **SAMPLEOUT**: muestra de 16 bits para ser enviada al CODEC

Salidas:

• **DACDAT**: salida serie por la que se envían las muestras al CODEC.

• **OUTREADY**: se activa durante un ciclo de reloj para indicar que ya se ha leído la muestra y se ha comenzado a enviar al CODEC.

4.4.- Módulo APR_MIDI

Vamos a construir este módulo mediante dos submódulos. Los submódulos se explican con detalle más adelante, y son:

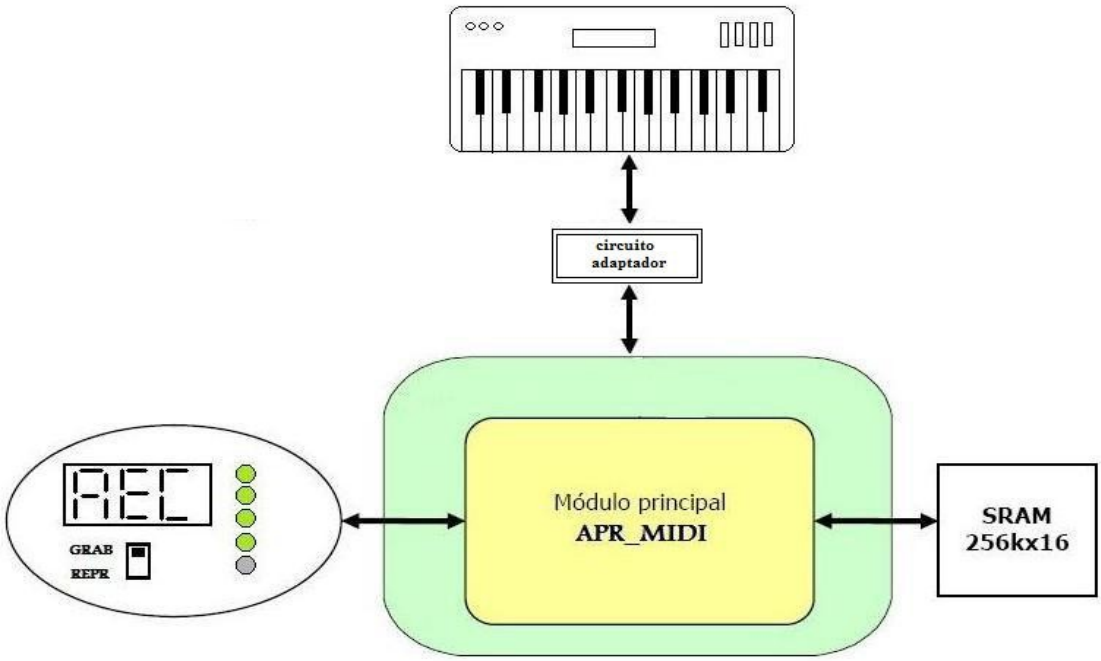
• **MIDI_CONTROL,** que se encarga de interpretar las señales procedentes del teclado MIDI. Las señales pasan a través del circuito creado para poder conectar el teclado y la tarjeta.

• **GRAB_REPR,** que se encarga de interpretar las señales procedentes de los conmutadores.

▪ En modo grabación, almacena en una memoria las muestras de audio procedentes de AUDIO_IO, y muestra en el display la palabra REC.

▪ En modo reproducción, lee las muestras de la memoria y se las entrega a AUDIO_IO, y muestra en el display la palabra PLAY.

El esquema se muestra en la figura siguiente:

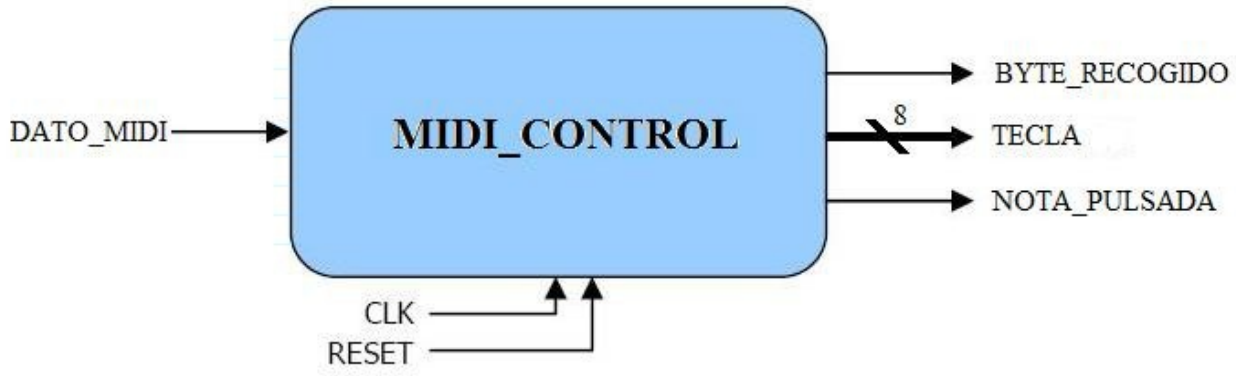


4.4.1.- Módulo MIDI_CONTROL

Este módulo se encarga de recoger los datos provenientes del teclado MIDI. Los datos llegarán a este módulo por el pin GPIO[35] en serie a la linea DATO_MIDI.

Cada vez que reciba un MIDI_byte (10 bits), activará la señal BYTE_RECOGIDO. También se almacenará información acerca de las teclas del teclado MIDI, es decir, si se ha pulsado/soltado una tecla y cuál es la TECLA que se ha pulsado/soltado.

El esquema del módulo es el siguiente:



Entradas:

• **DATO_MIDI:** bit procedente del teclado MIDI. Se trata de una linea serie, por lo que llegarán bites hasta completarse un MIDI_byte (10 bites).

Salidas:

• **BYTE_RECOGIDO:** se activa cada vez que se recibe un MIDI_byte.

• **TECLA:** muestra la tecla sobre la que se actúa; o se ha pulsado esta tecla, o se ha soltado.

• **NOTA_PULSADA:** Si vale 1 es que se ha pulsado la tecla TECLA, en cambio, si vale 0 se ha soltado la tecla.

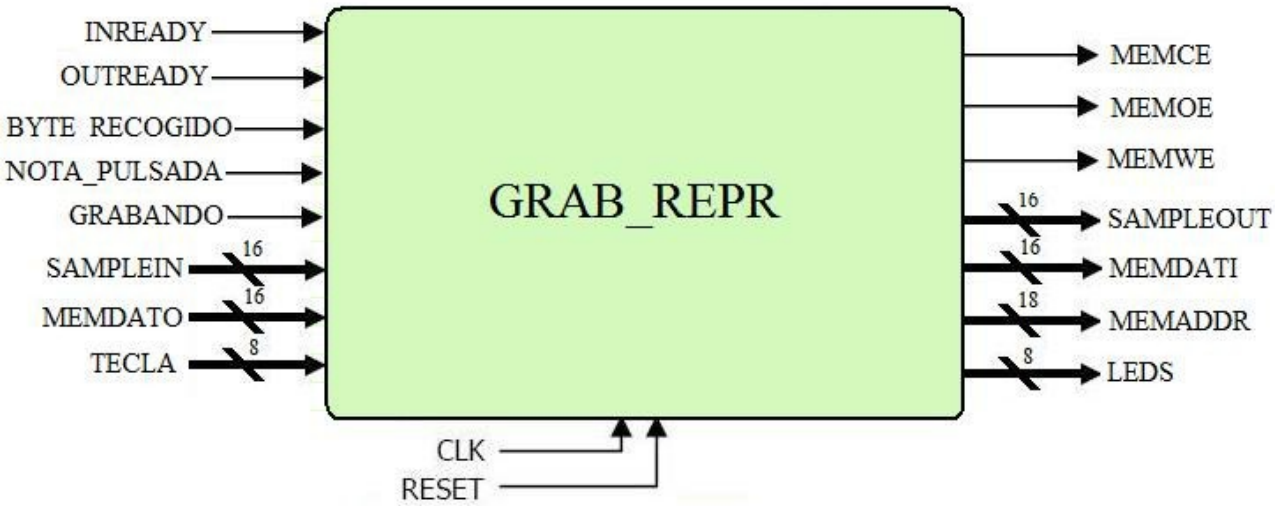
4.4.2.- Módulo GRAB_REPR

Este módulo controlará el funcionamiento de todo el sistema. Se encargará de interpretar la señal procedente del conmutador GRABANDO. A partir de ahí:

• En modo grabación, almacenará en una memoria las muestras de audio procedentes de AUDIO_IO, y mostrará en el display la palabra REC.

• En modo reproducción, leerá las muestras de la memoria y se las entregará a AUDIO_IO, y mostrará en el display la palabra PLAY.

El esquema del módulo es el siguiente:



Entradas:

• **INREADY:** indica que en la salida de AU_IN hay una muestra procedente del CODEC.

• **OUTREADY:** indica que AU_OUT ya ha cargado la muestra y ha comenzado a enviarla.

• **BYTE_RECOGIDO:** procede del módulo MIDI_CONTROL y se activa cada vez que se recibe un MIDI_byte.

• **NOTA_PULSADA:** Si vale 1 es que se ha pulsado la tecla TECLA, en cambio, si vale 0 se ha soltado la tecla.

• **GRABANDO:** Si vale 1, el sistema está en modo grabación. Si vale 0, el sistema estará en modo reproducción.

• **SAMPLEIN:** muestra de 16 bits procedente del CODEC.

• **MEMDATO:** dato de 16 bits procedente de la memoria.

• **TECLA:** Código de la tecla sobre la que se realiza la acción (Grabar/reproducir).

Salidas:

• **MEMCE:** señal de habilitación para la memoria.

• **MEMOE:** señal de habilitación para la salida de la memoria.

• **MEMWE:** señal de habilitación para la escritura en memoria.

• **SAMPLEOUT:** muestra de 16 bits para ser enviada al CODEC.

• **MEMDATI:** dato de 16 bits para escribir en la memoria (triestado, debe poder ponerse en alta impedancia porque el bus de datos de la memoria es bidireccional).

• **MEMADDR:** dirección de 18 bits para la memoria.

• **LEDS:** cada bit de los 8 corresponde con un LED de la placa. Cuando alguno de los bits tenga el valor 1 se encenderá el LED correspondiente, si vale 0, el LED permanecerá apagado.

4.5.- Toma de decisiones

Una vez se ha finalizado la fase de estudio y comprensión del proyecto en el que nos vamos a basar, se ha decidido elegir el módulo GRAB_REPR anteriormente mencionado (apartado 3.4.1.) para llevar a cabo la puesta en práctica de los conocimientos que mas adelante adquiriremos.

En el PFC actual, sustituiremos dicho módulo, el cual esta desarrollado en VHDL, por otro con función similar desarrollado mediante la herramienta **SoPC Builder** y programado mediante la aplicación de **Nios II para Eclipse**.

La intención que se persigue es conseguir que tras sustituir el módulo anterior por el obtenido en este PFC la aplicación resultante tenga la misma funcionalidad o la mas parecida posible al anterior, logrando que el usuario final no note diferencia alguna entre la aplicación desarrollada únicamente en VHDL y la aplicación H/S-c desarrollada en este PFC.

5.- DESARROLLO

5.1.- Formación y aprendizaje

Una vez decididas las tecnologías y herramientas a utilizar en el paso anterior, y al ver que no se tiene conocimiento alguno de ellas, toca formarse para poder realizar la aplicación anteriormente descrita.

Parte de las herramientas descritas no han tenido que aprenderse puesto que en la asignatura cursada de Laboratorio de Diseño Digital se aprendió a utilizar el Quartus II Web Edition y se adquirieron también conceptos mas que suficientes sobre el lenguaje VHDL para poder realizar cambios en la aplicación inicial en caso de que fueran necesarios.

Para llevar a cabo dicha formación, en concreto la que respecta a la utilización del SoPC Builder y del Nios II para Eclipse, se han leído manuales y realizado tutoriales de diferentes tipos, los cuales se pueden encontrar en la página web oficial de altera "http://www.altera.com/". Altera nos ofrece infinidad de manuales y tutoriales con ejemplos prácticos para cada uno de los casos.

Esta parte del PFC ha consumido gran parte del tiempo disponible puesto que es la etapa en la cual esta basada el objetivo principal del proyecto. Por esta razón se ha hecho gran hincapié en intentar abarcar el mayor conocimiento posible sobre estas herramientas, viendo la gran diversidad y magnitud de aplicaciones que se pueden realizar combinando las citadas herramientas.

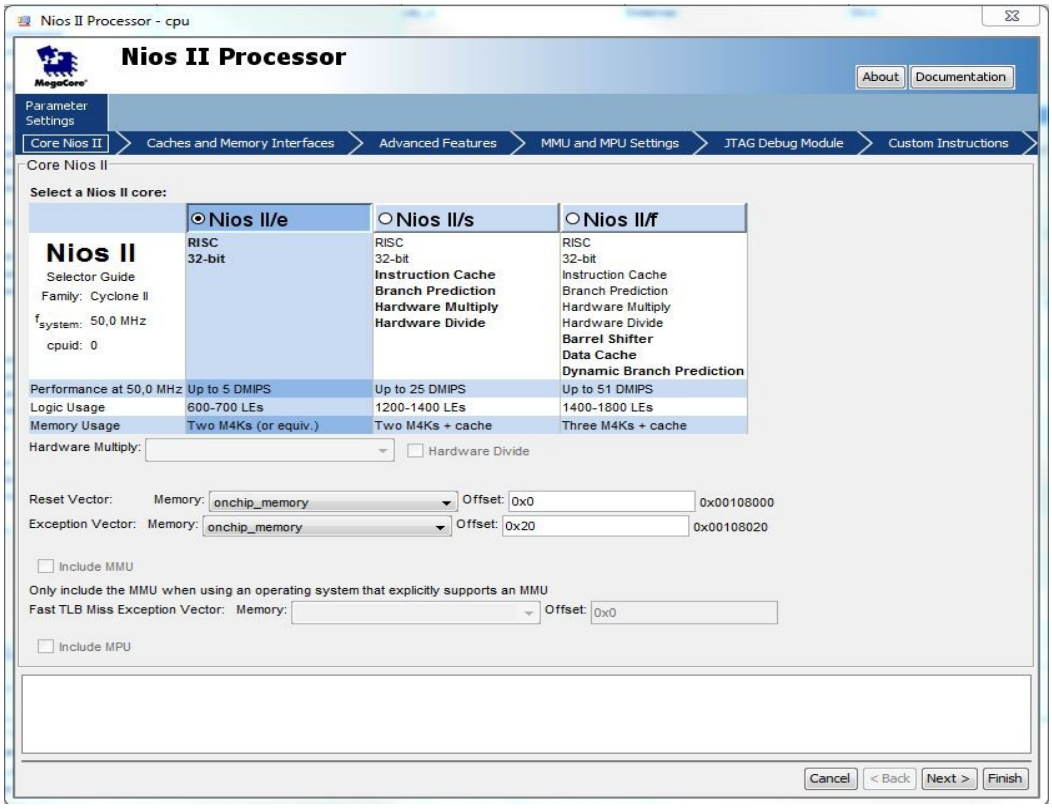
A lo largo del desarrollo de la aplicación ha hecho falta recurrir infinidad de veces al material utilizado a lo largo de esta formación e incluso a buscar nuevo material mas completo para poder tratar de solventar los problemas encontrados a lo largo de la realización de la aplicación

5.2.- Desarrollo de la aplicación

En este apartado se explicará como se han realizado la aplicación práctica del proyecto.

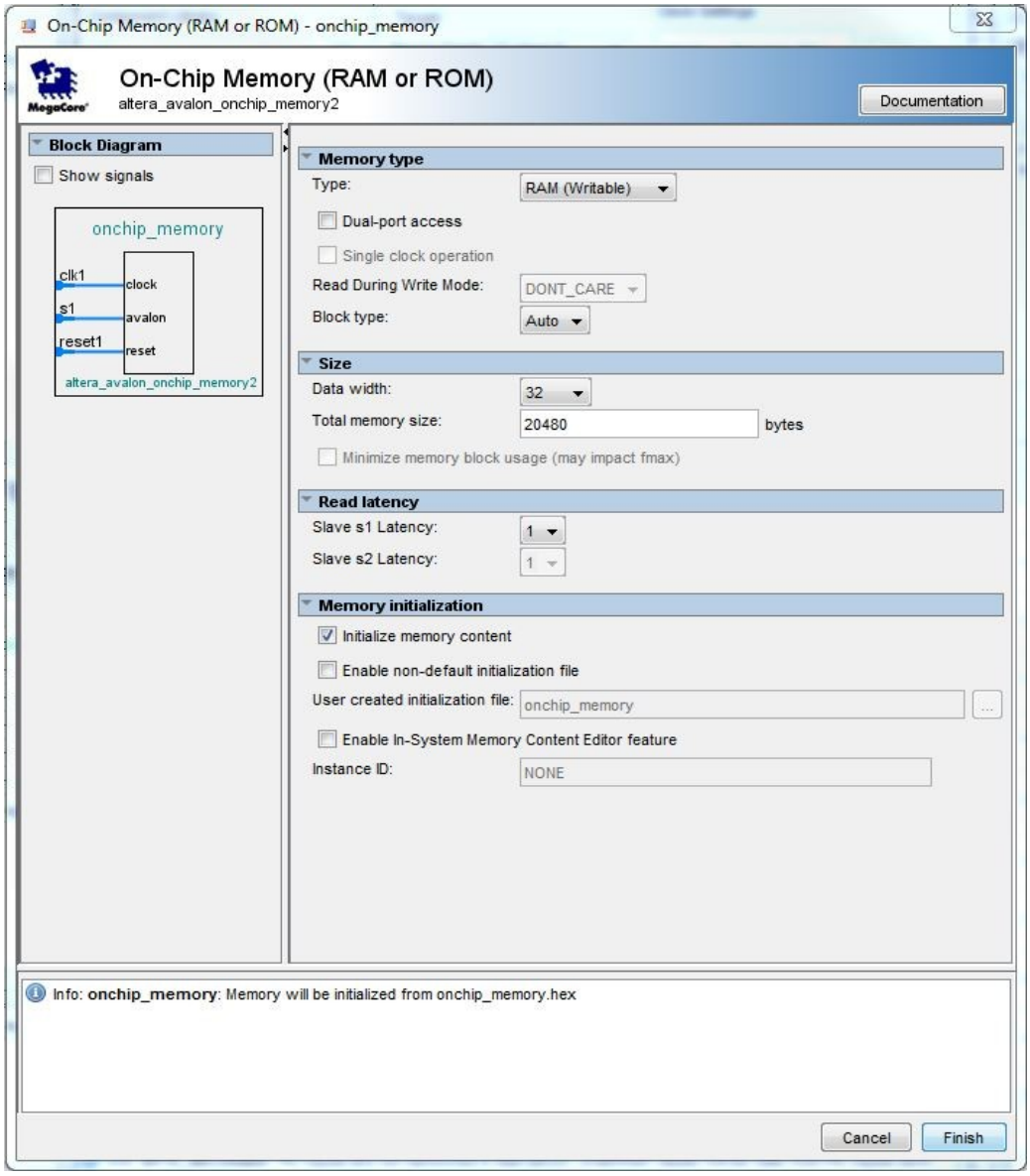
5.2.1.- Módulo Grab_Repr en SoPC Builder

Como bien nos indican en los tutoriales, primero se tuvo que agregar el componente principal, lo que será la "cpu" de nuestro microprocesador simulado, el procesador Nios II (*Component Library/processors/Nios II processor*). Es el cerebro del sistema, todo el software será interpretado por el y enviará señales correspondientes a sus periféricos para completar las tareas asignadas. Disponemos de tres tipos de procesadores: Nios II/e, Nios II/s y Nios II/f. Como se puede ver en la imagen, cada uno de ellos tiene diferentes carácteristicas y elegimos el Nios II/e por ser el mas simple y adecuado para nuestro caso en particular. Lo renombraremos como *cpu* para una mayor comodidad al utilizarlo.

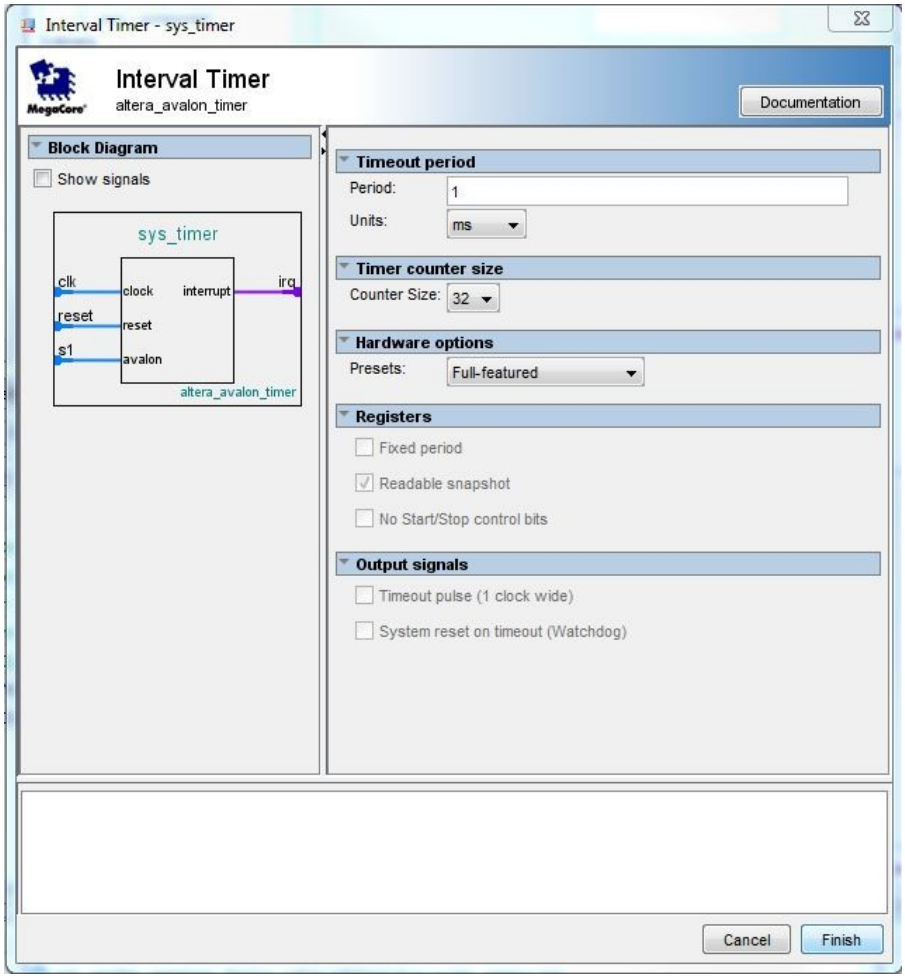


A continuación agregaremos uno de los periféricos mas importantes del sistema, el System ID (*Component Library/peripherals/System ID Peripheral*). Este permite a las herramientas de desarrollo de software para Nios II saber si la aplicación desarrollada pertenece al hardware objetivo. Este periferico asignará dos números (System ID y Time stamp) cuya combinación es única para cada proyecto de el cual sera reconocido por el desarrollador de software (Nios II Tools for Eclipse en nuestro caso).

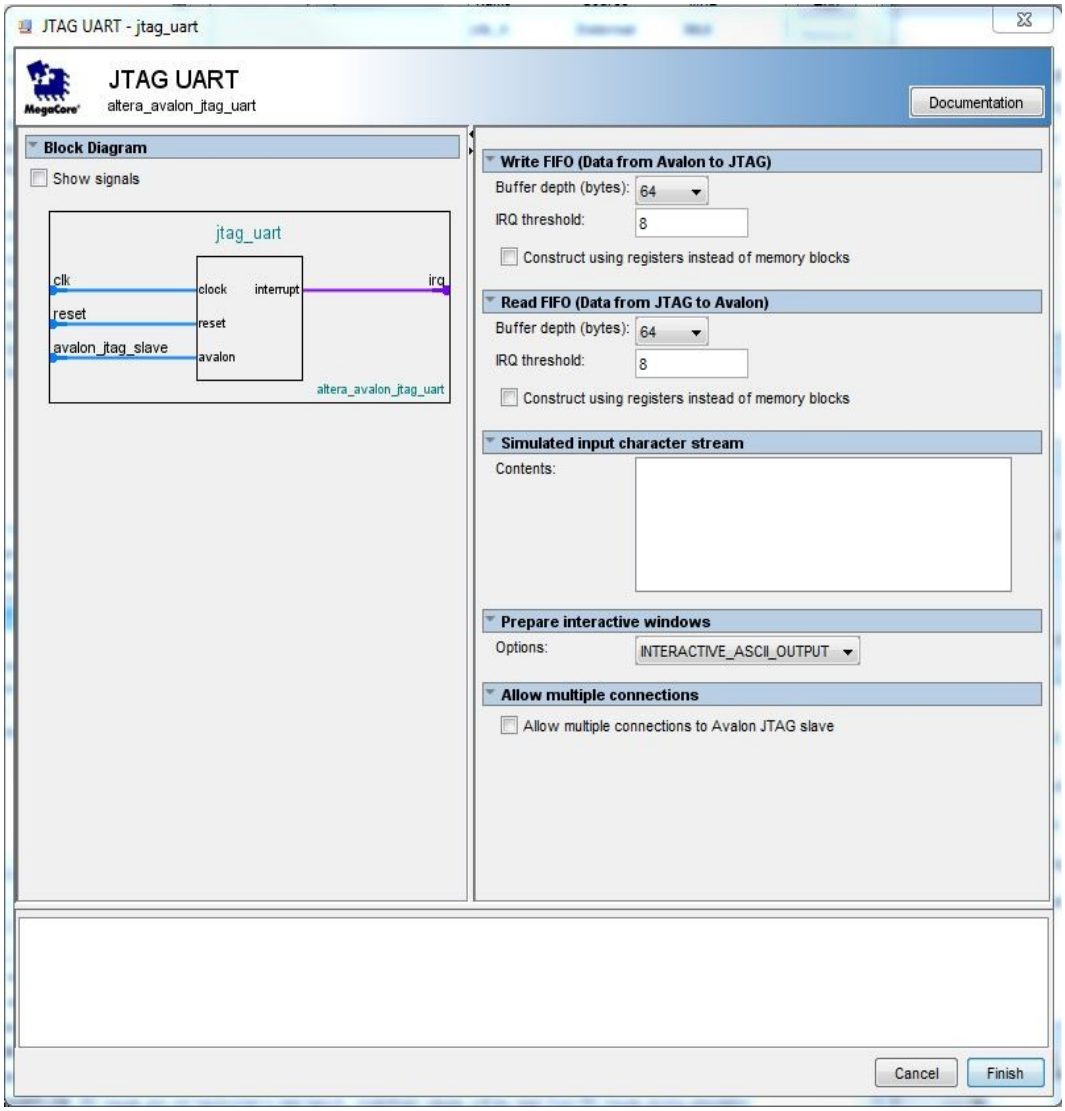
Otro de los componentes que necesitaremos para el desarrollo de la aplicación será una memoria implementada dentro del dispositivo FPGA como espacio para almacenar el software del sistema así como datos, hablamos de la On-Chip Memory (*Component Library/Memories/Memories and Memory Controllers/On-Chip/On-Chip Memory RAM or ROM*). Entre sus datos configurables elegiremos una memoría de tipo RAM (Writable) con un Data Width de 32 y 20Kb de tamaño. Lo renombraremos como *onchip_memory.*



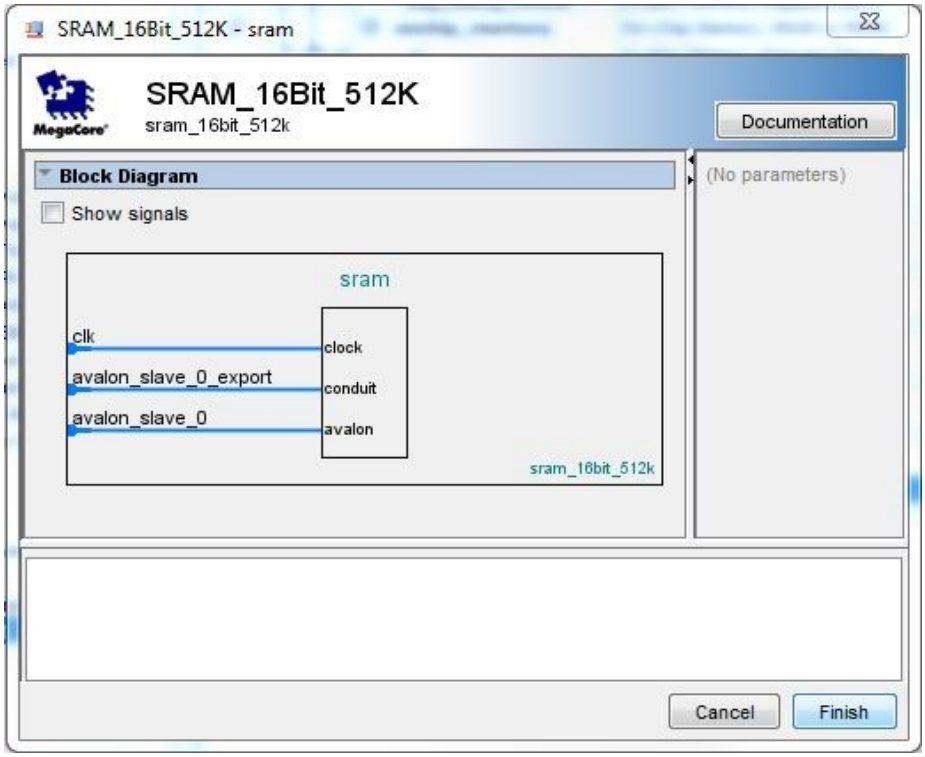
Para el encendido de los Leds necesitaremos que nuestra aplicación tenga la opción de medir intervalos de tiempo por lo que agregaremos un timer adicional para poder realizar dicha operación (*Component Library/Peripherals/Microcontroller Peripherals/interval Timer*). Lo configuraremos con un periodo de 1ms, un counter size de 32 y marcando Full-Featured entr las opciones disponibles en Presets. Será renombrado como *sys_timer.*



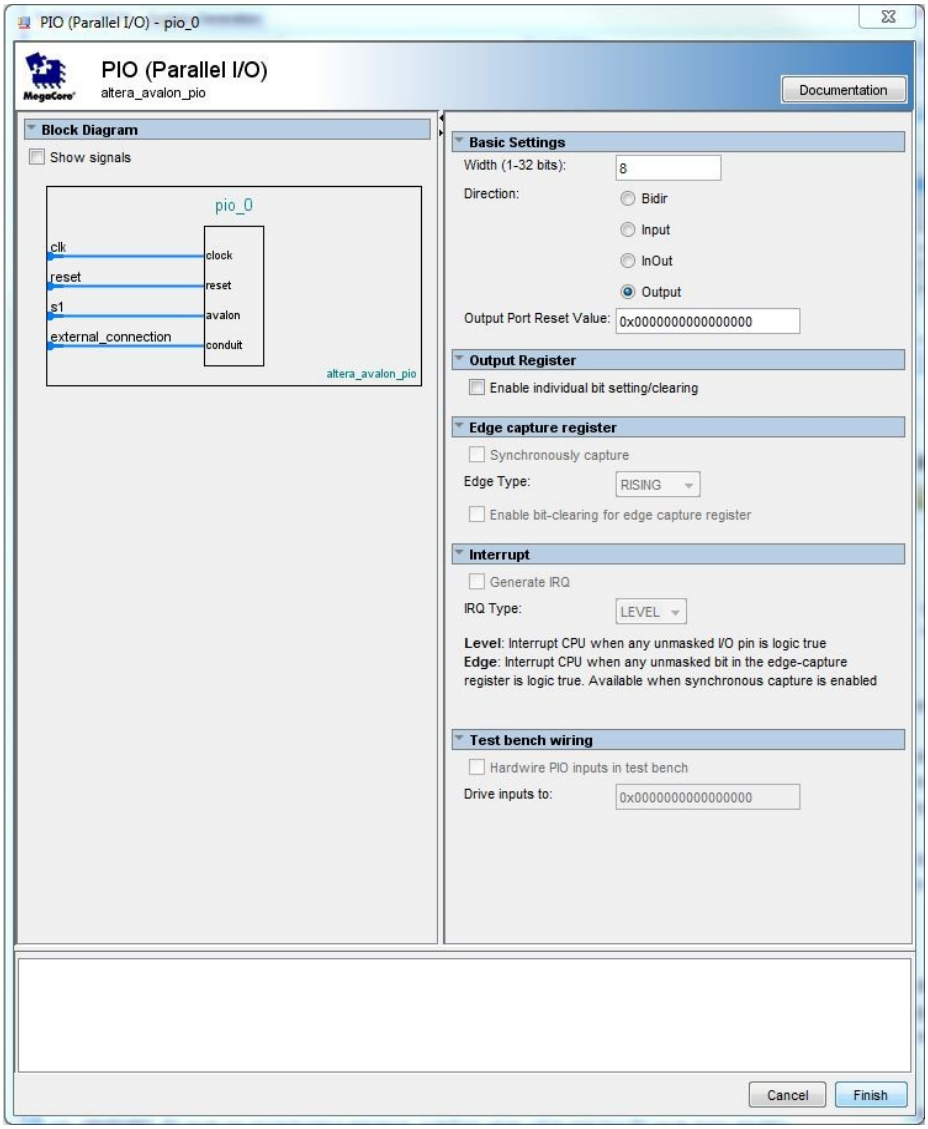
Se añade a continuación el JTAG UART que es una forma cómoda de comunicar datos con el procesador Nios II a través del cable de descarga USB-Blaster (*Component Library/Interface Protocols/Serial/JTAG UART*). Le daremos a finish dejando sin tocar ninguno de los parámetros configurables que aparecen por defecto.



Para poder utilizar la memoria SRAM externa implementada en VHDL en la cual guardaremos los Samples recogidos de las grabaciones para su posterior lectura en la reproducción, añadiremos un nuevo componente a la librería de SoPC. Se trata de un controlador de una SRAM el cual nos viene incluido en el CD de la placa DE2. Este controlador nos facilitará el uso de dicho componente de forma significativa. Para añadir un nuevo componente pulsaremos dentro de Component Library donde pone New component... seguido abriremos la pestaña File y pulsaremos sobre Open. Se nos abrirá una ventana de búsqueda en la cual seleccionaremos el archivo correspondiente *"class.ptf"* del CD para que nos generé de forma automática el nuevo componente. Se nos creara una nueva seccion dentro de Component Library llamada Terasic Technologies Inc dentro de la cual encontraremos el componente en cuestión llamado SRAM_16bit_512K. Clickaremos en finish ya que no dispone de parámetros configurables y lo renombraremos como *sram*.



El resto de señales de las cuales se compone el módulo GRAB_REPR hecho en VHDL se simularán añadiendo Puertos de Entrada/Salida simples (*Component Library /Peripherals/Microcontroller Peripherals/PIO (Parallel I/O)*). Varias de las señales necesitan solo de un pin para ser representadas pero a pesar de poder agrupar varias de ellas en un mismo puerto de 8 bits, se representaran por separado para facilitar la posterior comprobación de interrupciones individualmente en caso de que fuera necesario. Como se puede apreciar en la siguiente imagen, disponemos de diferentes campos de configuración como el numero de bits, el tipo de puerto (entrada, salida, bidireccional...), el valor que recibirá el puerto al ser reseteado...



Este tipo de periférico permite la interacción del sistema con el mundo exterior. Las señales representadas mediante PIO's son las siguientes:

Puertos de entrada:

- BYTE_RECOGIDO (1 bit)
- GRABANDO (1 bit)
- INREADY (1 bit)

- NOTA_PULSADA (1 bit)
- OUTREADY (1 bit)
- SAMPLEIN (16 bits)
- TECLA (8 bits)

Puertos de salida:

- LEDS (8 bits)
- SAMPLEOUT (16 bits)

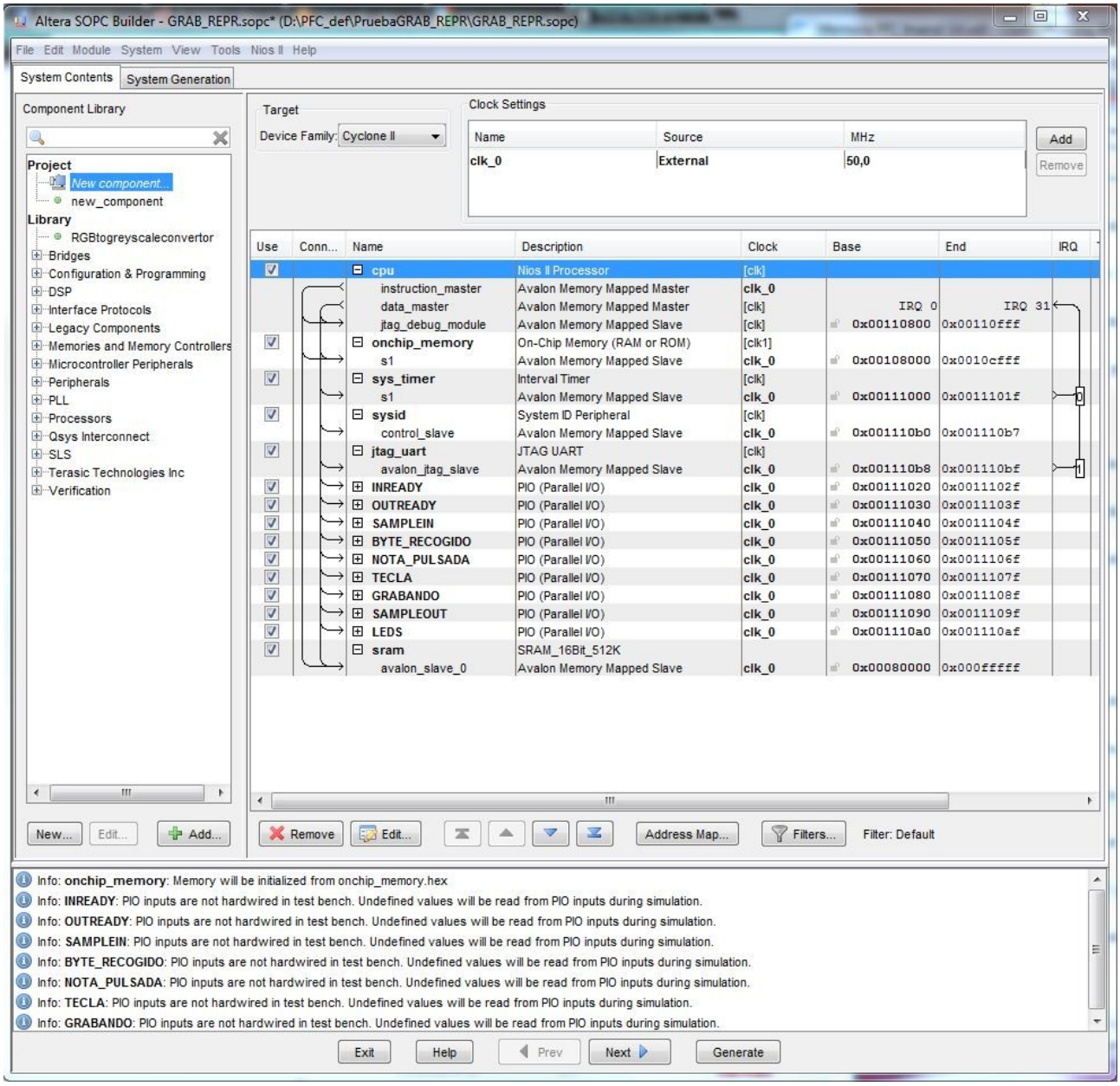
Una vez finalizada la agregación de todos los componentes necesarios para la realización del sistema, tendremos que realizar una limpieza general de dicho sistema para eliminar los errores y warnings mostrados por el SoPC Builder. A continuación mostraremos los pasos a seguir para eliminar dichos errores y warnings.

Nios II al igual que otros microcontroladores controla sus periféricos por medio de operaciones de lectura/escritura de registros. Es por esta razón que cada periférico debe tener asignada una dirección de inicio que no se sobreponga a la de otro periférico. Para que el sistema asigne de forma automática tanto un patrón de direcciones adecuadas como el orden de prioridad de interrupciones, tan solo tendremos que seguir dos simples pasos, hacer click en la pestaña de System y acto seguido pulsar sobre las opciones de *Auto-Assign Base Addresses* y sobre *Auto-Assign IRQs* respectivamente. En el caso de la asignación de prioridad de interrupciones será mejor asignarlo de manera manual ya que SoPC Builder no conoce las características del hardware. Esto hará que desaparezcan algunos de los mensajes de error.

Para quitar el resto de mensajes de error deberemos volver a seleccionar el componente anteriormente añadido de Nios II Proccesor al cual hemos llamado *cpu y* donde pone Reset Vector y Exception Vector tendremos que seleccionar nuestra memoria particular que anteriormente hemos nombrado como *onchip_memory.*

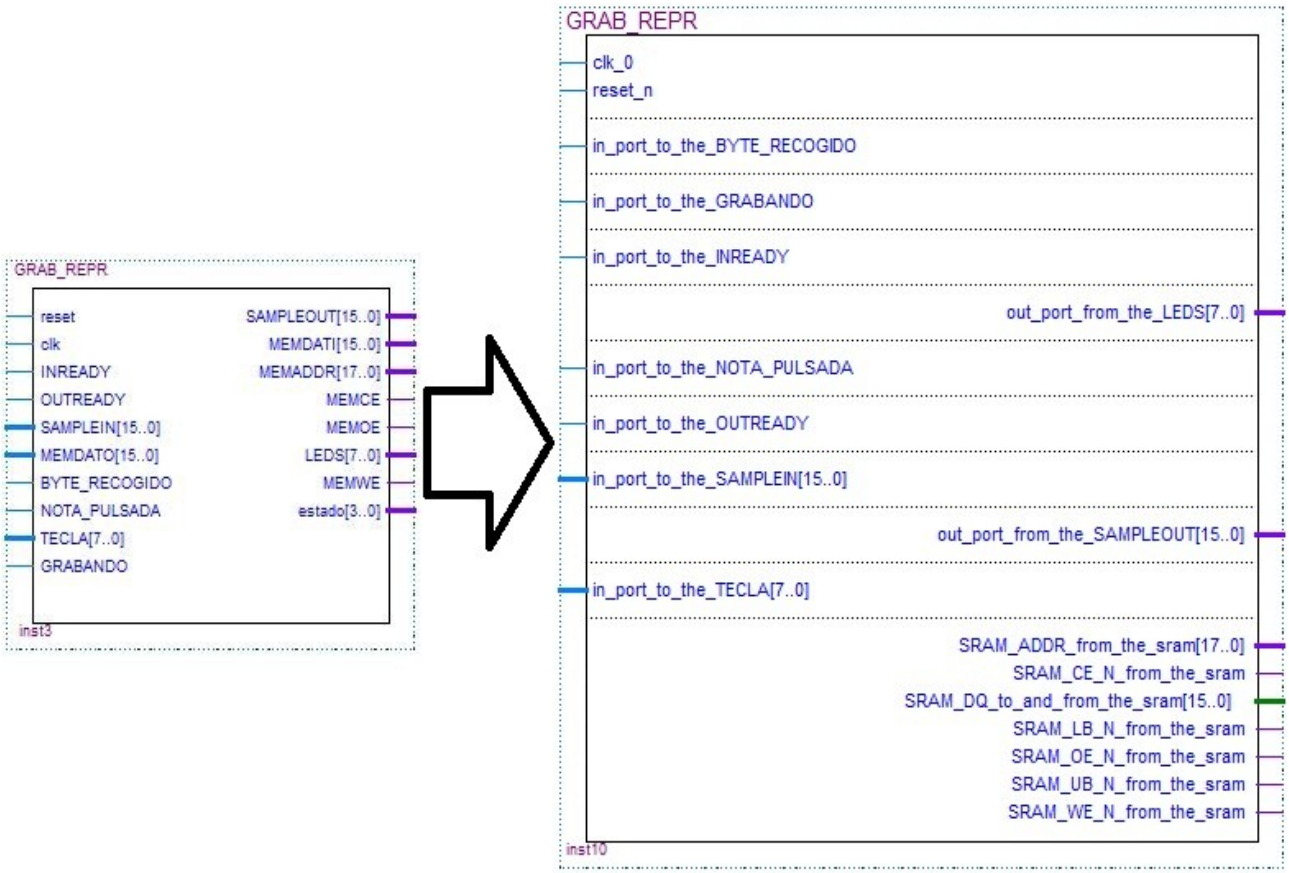
Tras estos sencillos pasos podremos generar con éxito el módulo GRAB_REPR creado con el SoPC Builder el cual contendrá todos los componentes añadidos en los anteriores pasos como bien se puede observar en la siguiente imagen. Este proceso creará una descripción en lenguaje HDL de todos los componentes que se agregaron al sistema SoPC Builder.

El resultado será un concentrado de las características del sistema que podrá ser utilizado por Quartus II y por Nios II Tools for Eclipse para continuar con el desarrollo de la aplicación.



Una vez generado el módulo toca volver al proyecto de Quartus II para unir las dos partes de las que disponemos, los módulos creados en VHDL y el módulo obtenido mediante el SoPC Builder.

Para realizar dicho paso tan solo tenemos que hacer doble click sobre una sección cualquiera, preferiblemente vacía para seleccionar en la carpeta *Project* el módulo GRAB_REPR. Una vez añadido deberemos unir las señales del GRAB_REPR con las señales correspondientes del resto del proyecto. Una vez realizado esto compilaremos el proyecto para ver que no da ningún error. A continuación se muestra la comparación entre el módulo GRAB_REPR del anterior proyecto el cual se ha eliminado con el módulo GRAB_REPR del proyecto actual el cual a sido añadido.



Módulo Viejo ***Módulo Nuevo***

5.2.2.- Programacíon mediante Eclipse

Como bien se ha mencionado en la sección 3.6 se puede programar el sistema creado tanto a bajo nivel (lenguaje maquina) como a alto nivel (lenguaje C). En este caso, para la realización de este PFC se ha utilizado la segunda opción ya que me resulta mas familiar y fácil de utilizar. El código generado se puede ver en el CD adjunto a esta documentación. Antes de empezar a programarlo hay que tener en cuenta un pequeño detalle, Eclipse es una plataforma de desarrollo de software capaz de adaptarse a diferentes lenguajes de programación, por lo que es importante asegurarse de que se encuentra configurado para el desarrollo de aplicaciones para Nios II. En caso de no ser así se modificará accediendo a *Window/Open perspective/Other.../Nios II.* A parte de esto, también es muy importante importar los ficheros adecuados a la hora de crear un nuevo proyecto desde el Eclipse. Si seleccionamos la opción de "Create a new BSP project based on the application project Template" El propio Eclipse nos pedirá que elijamos el fichero correspondiente a la parte creada desde Quartus II nada mas intentar crear un proyecto nuevo. De esta manera al programar el Nios II podremos tener variables ya creadas de los componentes añadidos desde el SoPC Builder con sus direcciones correspondientes, sus banderas de interrupción... sin tener que preocuparnos de todo esto.

Mas adelante mostraremos unos simples ejemplos para probar la plataforma obtenida y mostrar la sencillez del código resultante.

5.2.3.- Problemas surgidos

A lo largo de este desarrollo han surgido muchos problemas de los cuales no todos se han podido solucionar. La mayoría de ellos son debidos a falta de conceptos o conocimientos básicos y que gracias a la información de la página oficial de altera, el foro de dicha página y la ayuda de parte del profesorado de la facultad han podido ser solucionados. Muchos de estos también han sido ajenos a errores sintácticos que es lo que suele darse habitualmente en el desarrollo de software.

A continuación se explican los diferentes problemas rugidos y la solución encontrada para cada uno de los casos:

- ***Compatibilidad de versiones:*** Este es un error que ya se ha mencionado anteriormente en el capítulo 2.5 de esta memoria. Como bien se a señalado con anterioridad este problema a ocasionado múltiples fallos los cuales tienen una solución sencilla pero costosa en lo que a tiempo se refiere. Esta solución consiste en volver a recompilar primero el sistema del SoPC Builder, luego actualizarlo en el proyecto del Quartus II y volver a compilar este proyecto y volver a cargarlo en la placa y por último volver a recompilar también la aplicación software creada desde el Eclipse.

- ***Señal Reset del módulo GRAB_REPR:*** Al sustituir el módulo GRAB_REPR antiguo por el creado nuevo, se conectaron todas las señales del proyecto anterior al nuevo módulo sin darse cuenta del detalle de que al conectarle la señal correspondiente al pin del reset_n del módulo, la generada desde el Quartus llamda

RESET, no era la adecuada para hacer que el proyecto funcionara. Este fallo hizo que el desarrollo del proyecto se ralentizara muchísimo puesto que al compilar desde el Quartus no da problema alguno pero al compilar desde el Eclipse da error y no deja compilarlo. Al no dar el error en el Quartus se estuvo buscando el fallo durante mucho tiempo en el código del Eclipse, en las conexiones físicas... hasta que se descubrió cual era el problema de verdad. La solución para arreglar esto es bien sencilla, simplemente basta con cambiar dicha señal del RESET por la de VCC y no dará mas problemas.

- ***Componente agregado de SRAM:*** El hecho de haber incluido el nuevo componente SRAM para facilitar el manejo y ahorrar tiempo ha acabado resultando un tanto incómodo puesto que cada vez que se va a modificar algo, se recompila, se cambia de equipo... se pierde la información correspondiente a la SRAM y se tiene que eliminar en el sistema de SoPC Builder y volver a recompilar y actualizar todo paso por paso con la pérdida de tiempo que ello supone.

- ***Reconocimiento del System ID:*** A pesar de no tener fallos de compilación en ninguna de las partes, durante bastantes días fue imposible avanzar en la realización del PFC puesto que a la hora de compilar desde el Eclipse, este daba fallo a la hora de reconocer el ID del sistema. Tras revisar la configuración se detectó que el Eclipse no obtenía ningún System ID ni Time Stamp del proyecto a ejecutar importado desde el Quartus II. Tras arreglar tanto el fallo de las versiones (utilizando la misma para realizar todos los pasos previos) como el fallo de la señal reset, y modificando el

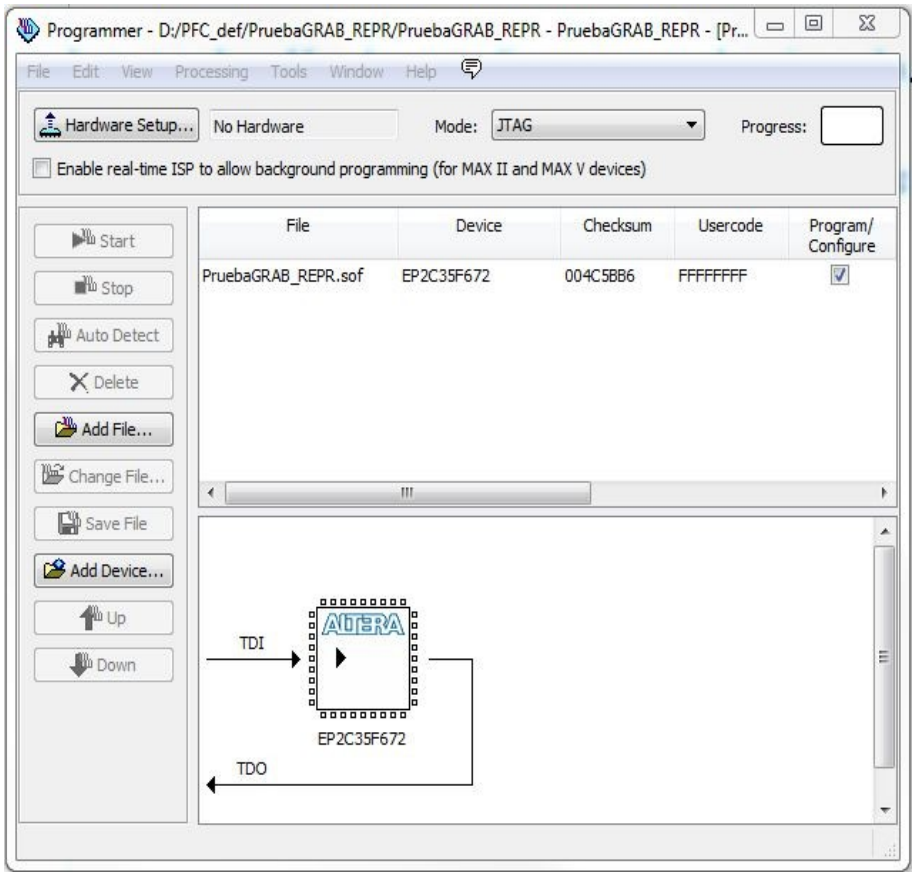
nombre del componente que asigna estos números al sistema de SoPC Builder por el de "*sysid"* se consiguió que reconociera bien la placa pudiendo así compilar la aplicación correctamente. Lo de modificar el nombre puede parecer insignificante puesto que el propio SoPC Builder le asigna otro nombre (System_ID_0) automáticamente pero en varias entradas del foro de la página oficial de altera y en varios de los tutoriales ajenos a altera apuntan que el hecho de ponerle otro nombre que no sea *sysid* o que exista mas de un periférico de este tipo, hará que falle provocando el no reconocimiento de la placa.

5.2.4.- Compilación y programación de las aplicaciónes

Una vez solucionados estos errores, se ha conseguido una correcta compilación tanto de la parte correspondiente al Quartus II (SoPC Builder incluido) como a la parte correspondiente a la programación del Nios II mediante Eclipse. Ahora que se ha conseguido todo esto solo queda descargar el sistema en la placa DE2 y ejecutarlo.

Descargar el programa a la placa resulta sencillo, únicamente se tiene que abrir en el Quartus II la pestaña llamada Tools y en el desplegable que aparece clickar sobre Programer. Se nos abrirá una ventana similar a la mostrada en la imagen inferior en el cual tras seleccionar el fichero .sof adecuado y a su vez, una vez detectada la conexión con el *USB-Blaster*, nos permitirá darle a Start para comenzar la descarga a la placa. En caso de no detectarnos el *USB-Blaster* habrá que darle al boton de *Hardware Setup* y seleccionarlo. Una vez se haya completado la descarga satisfactoriamente se nos mostrara en la barra de arriba a la derecha un 100% en color verde. Hecho

esto la placa está dispuesta para ser usada.



El siguiente paso será cargar el software compilado mediante Eclipse en la placa. Entraremos en Eclipse y haremos click derecho sobre el proyecto y le daremos a "*ejecutar como/Nio II Hardware*". Esperaremos a que se cargue del todo y ya podremos utilizar la aplicación cargada.

A continuación se muestran cuatro sencillos ejemplos utilizando los LEDs e interruptores de la placa DE2, el teclado MIDI y un micrófono y altavoces externos para comprobar el funcionamiento de parte de la plataforma y mostrar así cuatro simples ejemplos de posibles sistemas a desarrollar sobre dicha plataforma.

5.2.4.1.- Encender los LED's:

Funcionalidad: este sencillo programa va encendiendo los LED's del primero al cuarto intermitentemente para poder comprobar que la comunicacíon con los LED's y su funcionamiento es correcto.

Código:

```
#include <stdio.h>
#include "system.h"
#include "altera_avalon_pio_regs.h"
int main()
{
	int count = 0;
	int delay;
	while(1)
	{
		count = 0;
		while(count <= 0xf)
		{
			usleep(500000);
			#ifdef LEDS_BASE
				IOWR_ALTERA_AVALON_PIO_DATA(LEDS_BASE, count);
			#endif
			count++;
		}
	}
	return 0;
}
```

5.2.4.2.- Prueba Switch:

Funcionalidad: este programa detecta el estado en el que se encuentra el Switch (PLAY o REC) y te indica mediante la consola del Eclipse el estado en el que se encuentra. Mientras se este ejecutando habrá que cambiar el estado del Switch para comprobar que este funciona correctamente. De ser así nos indicara el nuevo estado en el que se encuentra mediante la consola.

Código:

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

int main()
{
	int count1=0, count2=0;
	printf("la TECLA funciona!\n");
	while(1)
	{
		if (IORD_ALTERA_AVALON_PIO_DATA(GRABANDO_BASE)==64 && count1==0){
			printf("PLAY//Reproduciendo...");
			count1=1;
		}
		else if (IORD_ALTERA_AVALON_PIO_DATA(GRABANDO_BASE)==65 && count2==0){
			printf("REC//Grabando...");
			count2=1;
		}
	}
	return 0;
}
```

5.2.4.3.- Prueba Teclas MIDI:

Funcionalidad: este programa detecta la tecla pulsada en el teclado MIDI e indica en la consola el nombre de la tecla pulsada. De esta manera podremos comprobar que tanto la comunicación con el teclado como el reconocimiento de las teclas es el correcto. Por ahorrar tiempo se ha realizado el ejemplo tan solo con las notas centrales del teclado musical.

Código:

```
#include <stdio.h>
#include "system.h"
#include "altera_avalon_pio_regs.h"
int main()
{
	int count=0, pulsada=1;
	while(1)
	{
		if(IORD_ALTERA_AVALON_PIO_DATA(NOTA_PULSADA_BASE)==65 && pulsada==1){
			usleep(100);
			count= IORD_ALTERA_AVALON_PIO_DATA(TECLA_BASE);
			pulsada=0;
			switch (count){
			case 8313:
				printf("has pulsado SOL \n");
			break;
			case 8345:
				printf("has pulsado LA \n");
			break;
			case 8377:
				printf("has pulsado SI \n");
			break;
			case 8393:
				printf("has pulsado DO \n");
			break;
			case 8425:
				printf("has pulsado RE \n");
			break;
			default:
```

```
                                        printf("tecla desconocida \n");
                                }
                        }
                        if(IORD_ALTERA_AVALON_PIO_DATA(NOTA_PULSADA_BASE)==64 && pulsada==0){
                                pulsada=1;
                        }
                }
                return 0;
}
```

5.2.4.4.- PruebaOUT:

Funcionalidad: este programa reproduce directamente por la salida (altavoces externos) lo que se escucha por la entrada (micrófono). De esta forma nos aseguramos un correcto funcionamiento de los diferentes codecs de audio y módulos de entrada y salida del audio.

Código:

```
#include <stdio.h>

#include "system.h"
#include "altera_avalon_pio_regs.h"
int main()
{


while(1)
{
IOWR_ALTERA_AVALON_PIO_DATA(SAMPLEOUT_BASE,IORD_ALTERA_AVALON_PIO_DATA(SAMPLEIN_BASE));
}
return 0;
}
```

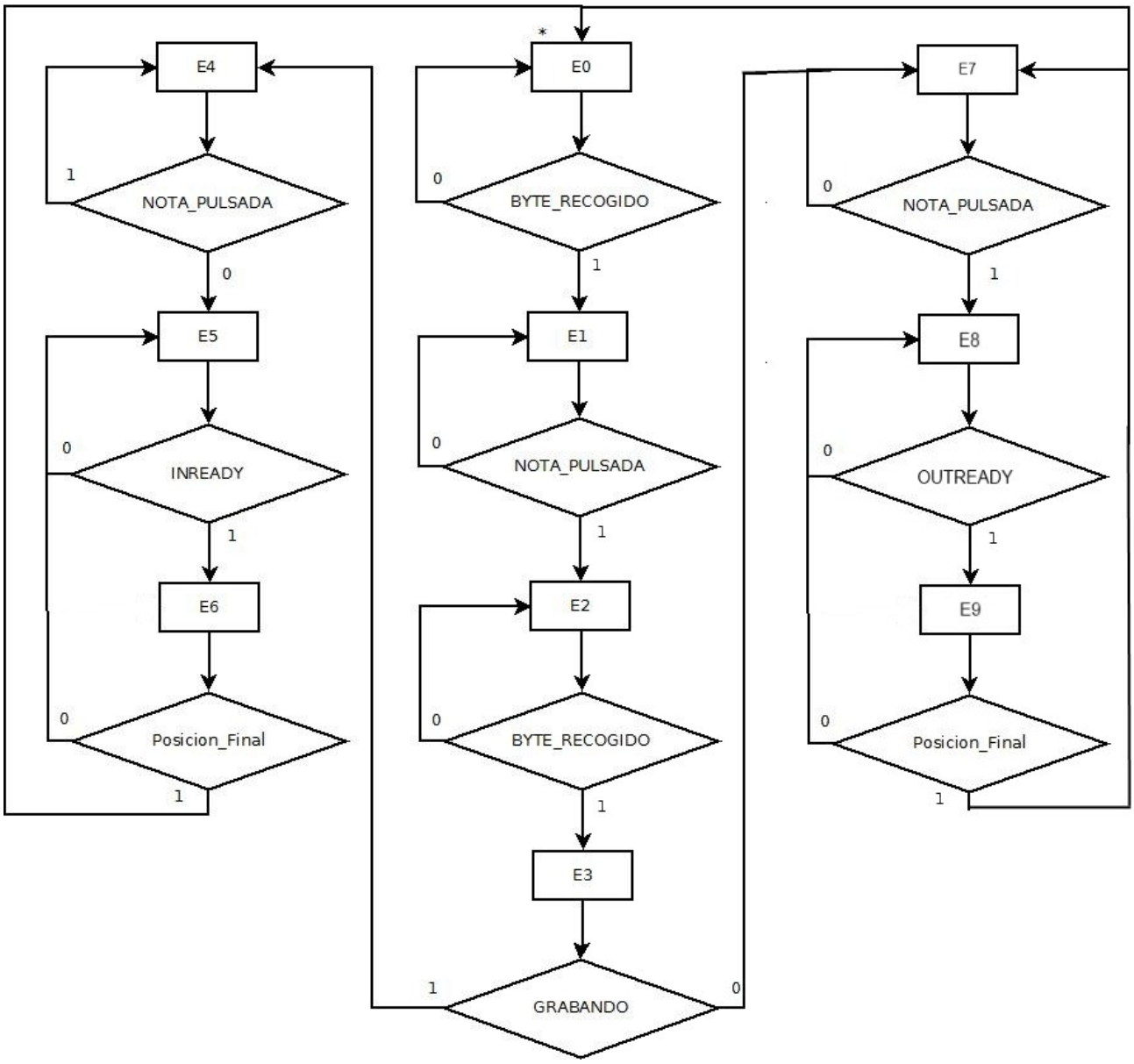
5.2.5.- Aplicación GRAB_REPR

En este capítulo explicaremos la realización de la aplicación GRAB_REPR la cual es el objetivo secundario de este PFC. La finalidad de esta aplicación es conseguir que nuestro sistema grabe y reproduzca sonidos captados por un micrófono, guardando y asignando diferentes sonidos a cada una de las teclas del teclado MIDI. Se intentará conseguir una funcionalidad similar a la del PFC en el cual está basado este proyecto.

A pesar de que el paso de ejecución paralela a ejecución secuencial no se trate de una simple traducción, y de que de hacerse así se asegura un mal funcionamiento de la aplicación, se ha basado el desarrollo del programa en el mismo diagrama ASM del PFC anterior. Ya que la funcionalidad final que se quiere conseguir es la misma se a tomado como base dicho algoritmo pero no se a seguido al pie de la letra puesto que como acabamos de mencionar, la diferencia entre ambos tipos de ejecuciones conllevaría a un resultado erróneo.

Para realizar la ya citada “traducción” ha sido importante mantener cierto nivel de abstracción en todo momento para no caer en el error de copiar el código literalmente y fijándose tan solo en los casos de uso de la aplicación. Es decir, utilizando tan solo el hecho de saber que se debe hacer con cada una de las señales que se activan o con los datos que llegan al módulo Grab_Repr.

El algoritmo en el cual está basado el desarrollo es el siguiente:



Comenzaremos en el E0 en el cual inicializaremos los LEDs para que estén apagados al comienzo de la ejecución del programa. Una vez inicializados deberemos esperar a la interrupción de la señal BYTE_RECOGIDO (primera interrupción será la de note_on). Una vez interrumpido pasaremos al E1 en el cual tendremos que esperar a que se pulse una nota del teclado. Sabremos que se a pulsado cuando se active la interrupción correspondiente a la señal NOTA_PULSADA. Ahora

que ya se a pulsado la nota pasaremos al E2 donde habrá que volver a esperar la nueva interrupción de BYTE_RECOGIDO. Una vez interrumpido pasaremos al E3. En este estado tendremos en la variable TECLA el valor de la tecla que ha sido pulsada por lo que la guardaremos para su posterior uso.

Llegados a este punto (seguimos en el E3), toca comprobar el estado del Switch para saber si estamos en modo Reproducción o en modo de Grabación. Para saber en que estado se encuentra el Switch tan solo tendremos que comprobar el valor de la señal GRABANDO la cual nos indicara si estamos en modo PLAY (1) o en modo REC (0). Según el valor obtenido pasaremos al E4 o E7 respectivamente.

En el caso de haber pasado al E4 tendremos que esperar a que se suelte la tecla (NOTA_PULSADA=0). Una vez se haya soltado pasaremos al E5. Aquí tendremos que realizar el mismo bucle 10485 veces (posiciones de memoria asignadas a cada una de las teclas). Según el código de la tecla que se haya pulsado le corresponderán diferentes posiciones de memoria para así poder guardar diferentes sonidos para diferentes teclas. Ahora habrá que esperar a que la señal INREDY se active para indicarnos que tenemos un Sample preparado en SAMPLEIN para ser guardado en memoria. Una vez activada dicha señal se pasara al E6 y se guardara el sample obtenido en la posición de memoria que le corresponda en cada caso. Llegado a este punto se volverá al E5 hasta que acabemos con todas las posiciones de memoria disponibles. Durante dicha "grabación" iremos encendiendo los LEDs progresivamente para que el usuario tenga una referencia visual del tiempo estimado para la grabación. Una vez finalizadas las posiciones disponibles volveremos al E0 con la muestra grabada almacenada en memoria.

En el caso de que en el E3 hayamos detectado que nos encontramos en modo de reproducción saltaremos al E7. En este modo el programa comprobará que la tecla sigue pulsada (NOTA_PULSADA=1). Mientras no se haya soltado pasaremos al E8. Aquí al igual que en el modo de Grabación, tendremos que realizar el mismo bucle 10485 veces (posiciones de memoria asignadas a cada una de las teclas). Según el código de la tecla que se haya pulsado le corresponderán diferentes posiciones de memoria para así poder reproducir diferentes sonidos para diferentes teclas. Ahora habrá que esperar a que la señal OUTREDY se active para indicarnos que se ha finalizado de reproducir el Sample anterior y tenemos preparado siguiente en SAMPLEOUT para ser reproducido. Una vez activada dicha señal se pasara al E9 y se colocara el sample obtenido de la posición de memoria que le corresponda en el SAMPLEOUT para que pueda ser reproducido. Llegado a este punto se volverá al E8 hasta que acabemos con todas las posiciones de memoria disponibles. Durante dicha "reproducción" iremos encendiendo los LEDs progresivamente para que el usuario tenga una referencia visual del tiempo estimado para la grabación. Una vez finalizadas las posiciones disponibles volveremos al E7 y realizaremos estos últimos pasos hasta que se suelte la tecla, en cuyo caso volveremos al estado inicial del programa, el E0.

6.- CONCLUSIONES Y LINEAS ABIERTAS

6.1.- Desviaciones

No se pudo cumplir con a planificación inicial de las horas estimadas para la realización del proyecto puesto que la parte del desarrollo llevo mas tiempo del previsto. Esto se debe a varias razones en concreto:

- *Compatibilidad de versiones:* La versión utilizada tanto del Quartus II y SoPC Builder, como la de Nios II para Eclipse, son diferentes en las diferentes maquinas utilizadas en la universidad y en entorno de trabajo habitual (domicilio particular). Al no disponer de un ordenador portátil no se ha podido solventar ese problema y a lo largo del desarrollo a causado múltiples problemas los cuales se explicaran en el apartado "*5.2.- Desarrollo de la aplicación*".

- *Pasar de ejecución paralela a secuencial:* En un principio se subestimo la dificultad que supondría pasar de una aplicación de ejecución paralela a ejecución secuencial.

- *Desconocimiento de las herramientas:* El hecho de que no se tuviera conocimiento previo alguno sobre las herramientas utilizadas ha desencadenado que a pesar de haber dado por finalizada la fase de formación, se haya tenido que recurrir nuevamente a la búsqueda y lectura de manuales y tutoriales a lo largo de la implementación de la aplicación.

Como se ha comentado, la planificación inicial sufrió demoras temporales tanto en los apartados de formación (al tener que volver a recurrir a la búsqueda y lectura de nuevo material), como en el apartado de desarrollo de la aplicación. A continuación mostramos dichas diferencias representadas mediante una tabla:

TAREA	T. Planificado	T. Real
Formación en el uso de la tecnología SoPC Builder	25 horas	25 horas
Formación en el uso de la tecnología Nios II	25 horas	35 horas
Desarrollo de la aplicación	40 horas	70 horas
Documentación del Proyecto	40 horas	40 horas
TOTAL	**130 horas**	**170 horas**

6.2.- Resultados obtenidos

Los objetivos establecidos al comienzo del PFC han sido 3, los dos principales que eran aprender a desarrollar aplicaciones H/S-c y desarrollar la ya mencionada plataforma hardware, y el secundario de realizar la modificación en la aplicación anterior.

El primero de los objetivos principales se da por cumplido puesto que a lo largo de este proyecto se han adquirido conocimientos suficientes para desarrollar aplicaciones H/S-c. Durante la realización de las pequeñas aplicaciones y tutoriales de prueba se a comprobado la facilidad que dan las herramientas utilizadas para que sin saber gran cosa se puedan realizar diseños mas o menos complicados. Una de las cosas buenas que saco del proyecto es la soltura cogida trabajando con las

herramientas utilizadas. Al comienzo el simple hecho de crear un nuevo proyecto y hacer que compilara resultaba casi imposible puesto que aparecían fallos de todo tipo los cuales no se sabían solventar por falta de conocimiento, en cambio ahora, no resulta complicado hacer pequeñas aplicaciones de forma rápida y sencilla.

El segundo de los objetivos principales también lo daremos por completado puesto que como se ha podido observar en capítulos anteriores, la plataforma hardware esta lista para ser utilizada y funciona perfectamente. Costó conseguir realizar dicha plataforma pero una vez obtenida se ha comprobado que una vez conseguida la base, se pueden desarrollar diferentes tipos de aplicaciones sin complicación alguna (salvo pequeños errores que siempre pueden surgir).

La parte mala de este PFC a sido el hecho de no haber cumplido con el objetivo secundario propuesto, el de conseguir que una vez modificada la aplicación anterior sustituyendo e incluyendo el nuevo módulo diseñado, esta disponga d la misma funcionalidad que antes. Esto se debe en gran parte a que el cálculo del tiempo al planificar el proyecto fue erróneo y a que al no disponer de ningún conocimiento previo, han surgido muchos imprevistos que aparentemente eran fáciles de solucionar y al final han tardado demasiado tiempo en solventarse.

En conclusión, no me satisface del todo el resultado del PFC puesto que no se han logrado todos los objetivos propuestos en su comienzo, pero se saca una lectura positiva del PFC que es que se ha aprendido a usar nuevas herramientas y tecnologías las cuales puede que en un futuro ayuden y faciliten la obtención de un puesto de trabajo. Como se dijo al comienzo del proyecto, una de las

razones que había motivado el proyecto era aprender cosas nuevas para poder abrir mas puertas en un futuro por lo que a pesar de tener esa espina clavada por no conseguir que la aplicación funcione, se finaliza la asignatura con un relativo buen sabor de boca.

En cuanto a posibles líneas de desarrollo abiertas, se podría plantear el hecho de poder finalizar la práctica cuando se disponga de mas tiempo ya que las razones que han hecho que este PFC resulte inviable han sido la falta de tiempo y de conocimiento. Ambas razones tienen solución por lo que no se descarta que en un futuro se acabe el trabajo e incluso que se añada alguna mejora o nueva funcionalidad.

7.-BIBLIOGRAFIA

Apuntes de la asignatura de Laboratorio de Diseño Digital.

Entradas Wikipedia:

- http://es.wikipedia.org/wiki/Quartus_II
- http://en.wikipedia.org/wiki/Nios_II
- http://en.wikipedia.org/wiki/Sopc_builder
- http://es.wikipedia.org/wiki/VHDL

Página oficial de Altera:

- http://www.altera.com/

Foro oficial de Altera:

- http://www.alteraforum.com/