Trabajo Fin de grado
Grado en Ingeniería de Tecnologías de Telecomunicación.

Sistema de medida remota de señales eléctricas en tiempo real.

Autor: Ángel Ruiz Martínez
Tutor: Sergio Vázquez Pérez

Dpto. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Trabajo Fin de Grado
Grado en
Ingeniería de Tecnologías de Telecomunicación.

Sistema de medida remota de señales eléctricas en tiempo real.

Autor:
Ángel Ruiz Martínez

Tutor:
Sergio Vázquez Pérez
Profesor titular

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2018

Trabajo Fin de Grado: Sistema de medida remota de señales eléctricas en tiempo real.

Autor: Ángel Ruiz Martínez

Tutor: Sergio Vázquez Pérez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

Agradecimientos

Me gustaría agradecer a Sergio Vázquez Pérez por su implicación en el proyecto y todas las indicaciones y conocimientos que me ha proporcionado para la realización del presente proyecto. A su vez, por extensión, a todo el Departamento de Electrónica de la Escuela Técnica Superior de Ingenieros y a la Universidad de Sevilla por los medios prestados para la realización del mismo. También agradecer a Abraham Márquez Alcaide por el desarrollo de la interfaz externa, una de las partes utilizadas en la elaboración del mismo.

Resumen

El objetivo del trabajo es la realización de una monitorización y control de manera remota de un convertidor analógico-digital. Para ello nos serviremos de la placa ZedBoard y de los diversos recursos de los que dispone. Usaremos uno de los dos microprocesadores para la gestión del convertidor y en el otro microprocesador ejecutaremos un sistema operativo Linux muy básico que nos ayudará con las comunicaciones del sistema.

Para la realización del proyecto deberemos fabricar en primer lugar una placa de desarrollo para conectar esta con la ZedBoard y poder obtener los datos del mismo para su posterior envío. Además, tendremos que desarrollar todas las aplicaciones necesarias para todo el proceso.

Abstract

The objective of the project is the realization of a remote monitoring and control of an analog-digital converter. For this we will use the ZedBoard and the various resources available. We will use one of the two microprocessors for the management of the converter and in the other microprocessor we will execute a basic Linux OS that will help us with the communications of the system.

For the realization of the project we will have to first manufacture a development board to connect it with the ZedBoard and to obtain the data of it for later shipment. In addition, we will have to develop all the necessary applications for the whole process.

Índice

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

En el presente trabajo utilizaremos los recursos de la placa ZedBoard para conseguir controlar de manera remota, vía internet, un convertidor electrónico de potencia.

Durante el desarrollo del mismo tendremos que trabajar en diferentes campos tanto hardware como software. La estructura del mismo será la siguiente:

- Conocimiento y desarrollo del convertidor analógico-digital a utilizar. A su vez tendremos que fabricar una placa de adaptación para su interconexión y uso.
- Conocimiento y desarrollo de la Zedboard. Describiremos los recursos que usaremos y como haremos el desarrollo hardware de la misma.
- Interconexionado de elementos para su funcionamiento global.
- Desarrollo software. En este punto tendremos que crear todas las aplicaciones o sistemas necesarios para el correcto funcionamiento del sistema.
- Ejemplo de ejecución. Incluiremos además una guía de como ejecutar el sistema de manera clara y concisa.

El funcionamiento del sistema consistirá en la obtención de datos mediante el convertidor ADC para su posterior procesado dentro de uno de los núcleos ARM de la Zedboard. Al mismo tiempo, el segundo de los núcleos será utilizado para con la ayuda de un LINUX resolver toda la parte de las comunicaciones y poder enviar dichos datos vía internet.

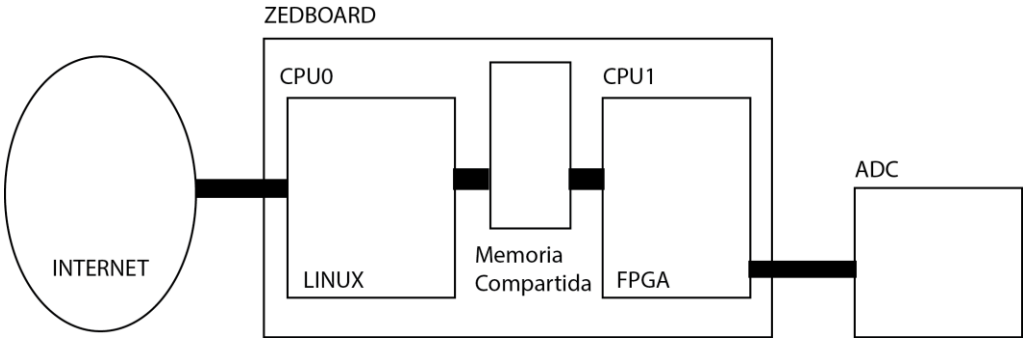


Ilustración 1-1: Esquema completo del sistema

2 ADS8365

Para la realización de nuestro proyecto necesitamos manejar una serie de convertidores analógico-digitales que nos proporcionaran los datos que queremos procesar. En el mercado existen una infinidad de opciones y nosotros hemos optado por el ADS8365. A continuación, veremos las principales características del mismo y porqué estas son interesantes para nosotros, así como, todo lo necesario, procedimientos y componentes para la realización de la placa de desarrollo para el uso del mismo.

Ilustración 2-1: ADS8365

2.1 Características

El ADS8365 [1] es un integrado compuesto de seis convertidores analógico digitales cuya frecuencia de funcionamiento es de 250kSPS (muestras por segundo). Estos seis convertidores nos hacen tener accesibles seis canales diferenciales que además pueden ser muestreados de manera simultánea. Esta capacidad para muestrear al mismo tiempo los seis canales es una de las características que nos ha hecho decantarnos por este integrado. La información de salida el dispositivo nos la muestra como una palabra de 16 bits.

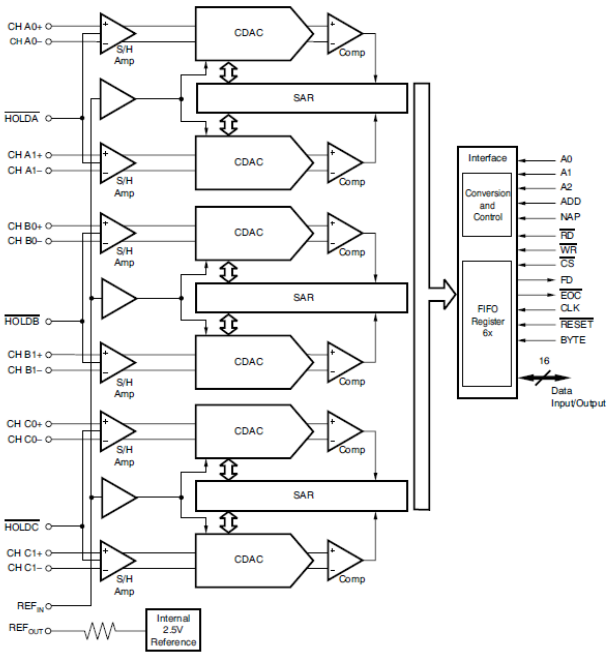


Ilustración 2-2: Diagrama de bloques ADS8365

2.2 Placa de Desarrollo

Para poder acceder a los datos del convertidor es necesario que hagamos una placa de adaptación para facilitar la conexión con la Zedboard. Debemos añadir una serie de elementos extras para el correcto funcionamiento del mismo.

2.2.1 Elementos

En el Anexo ***Esquemático placa desarrollo ADS8365*** podemos encontrar el esquemático completo correspondiente al diseño de la placa.

Como ya se ha comentado antes el integrado ADS8365 tiene accesibles seis canales pero nosotros no necesitamos tener acceso a todos para comprobar que funciona el sistema así que en nuestra placa de desarrollo solo tendremos accesibles cuatro de ellos con los que haremos las pruebas. Además, en el integrado estos canales son diferenciales y no nos interesa eso por lo que hemos incluido el integrado INA159 [2] añadiéndolo a cada entrada para así ya no tener una entrada diferencial.

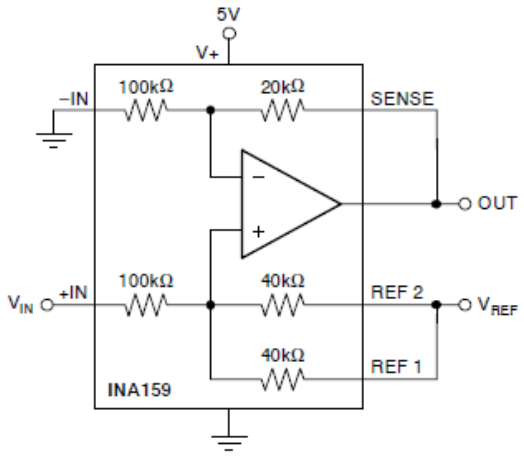


Ilustración 2-3: INA159

A la entrada marcada como +IN conectaremos nuestra señal de entrada y la marcada como OUT irá a entrada

positiva de uno de los canales. La parte negativa de esa entrada diferencial al canal irá conectada a la VREF de este integrado. Necesitamos un valor de tensión de referencia VREF de 2.5V y para conseguir este valor de manera estable usaremos el integrado OPA343 [3].

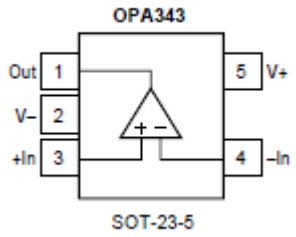


Ilustración 2-4: OPA343

Por último, también necesitaremos otra tensión de referencia para el ADS8365 en este caso de valor 2.048V. Para conseguir esta nos serviremos de la referencia de tensión LM4040 [4].

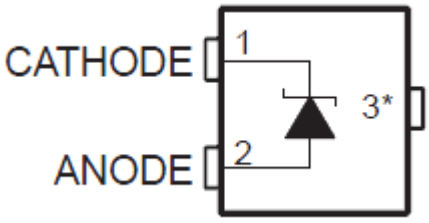


Ilustración 2-5: LM4040

Solo falta añadir algunos comentarios sobre el diseño completo. Para el correcto funcionamiento del sistema necesitaremos tres alimentaciones diferentes en la placa de desarrollo. La alimentación analógica marcada como AVDD será de 5v, la alimentación digital, marcada como BVDD será de 3V y, por último, necesitaremos dos tensiones de +15V y -15V para dotar a los canales del rango de funcionamiento correcto. Además, para optimizar la conexión con la Zedboard necesitaremos tener accesibles las dieciséis entradas correspondientes a la configuración y control de integrado y las dieciséis salidas correspondientes a los datos.

En el Anexo ***Rutado placa desarrollo ADS8365*** podemos encontrar como quedaría la placa lista para ser fabricada con todos los elementos rutados. Para la realización de dicho rutado se ha usado el programa EAGLE [5].

2.2.2 Fabricación

Como estamos ante una placa de desarrollo realizaremos la fabricación manual de la misma para usarla como prototipo de pruebas. Seguiremos un proceso de fabricación estándar para este tipo de placas haciendo uso de los elementos disponibles en el laboratorio.

El primer paso es realizar las impresiones de ambas caras de la placa en acetato. Cortaremos nuestra placa con el tamaño adecuado y colocaremos cada una de las caras en su posición. Hay que poner especial cuidado a la hora de alinear bien las caras para una correcta exposición.

Una vez tengamos las caras bien alineadas introduciremos la placa en la máquina insoladora durante un tiempo aproximado de 2 minutos y 45 segundos.

A continuación, cogeremos nuestra placa y la meteremos en un recipiente con el líquido revelador. El tiempo necesario no es algo fijo así que estaremos atentos para ver cuando la impresión de las pistas se hace visible. Entonces sacaremos la placa del líquido revelador y la limpiaremos bien con agua. Un detalle es importante es recordar reciclar este líquido revelador porque se puede usar varias veces.

Necesitamos ahora eliminar el cobre sobrante. Para ello prepararemos una solución que estará compuesta de, en

este orden, una parte de agua fuerte, una parte de agua oxigenada y dos partes de agua. Sumergiremos la placa en este líquido, con precaución porque puede ser perjudicial tocarlo y respirar los vapores, y cuando se haya eliminado todo el cobre sobrante la sacaremos y la secaremos con la ayuda de un papel.

El último paso de este primer proceso es eliminar la resina existente. Para ellos nos ayudaremos de un poco de acetona. Otra opción es volver a insolar la placa y pasarla por el revelador. Con esto nuestra placa ya debe estar lista para pasar al siguiente paso.

Pasaremos a realizar los taladros necesarios para los diferentes elementos que lo necesiten. En nuestra placa tendremos elementos que requerirán taladros y otros de montaje superficial. Una vez realizados todos los taladros resulta interesante hacer una primera comprobación de que hasta ahora todo el proceso va bien. Para ello haremos una prueba de continuidad sobre las diferentes pistas. Si todo es correcto podemos proseguir.

El último paso es realizar la soldadura de los diferentes elementos. Este proceso de soldadura tiene un orden que debemos seguir. En primer lugar, deberemos soldar las diferentes vías que tenga nuestra placa. A continuación, soldaremos los diferentes componentes SMD (de montaje superficial) y, posteriormente, el resto de los integrados. Por último, solo queda soldar los conectores y los pines.

Antes de dar por finalizada nuestra placa volveremos a realizar una comprobación de continuidad de la misma. Si todo es correcto nuestra placa está lista.

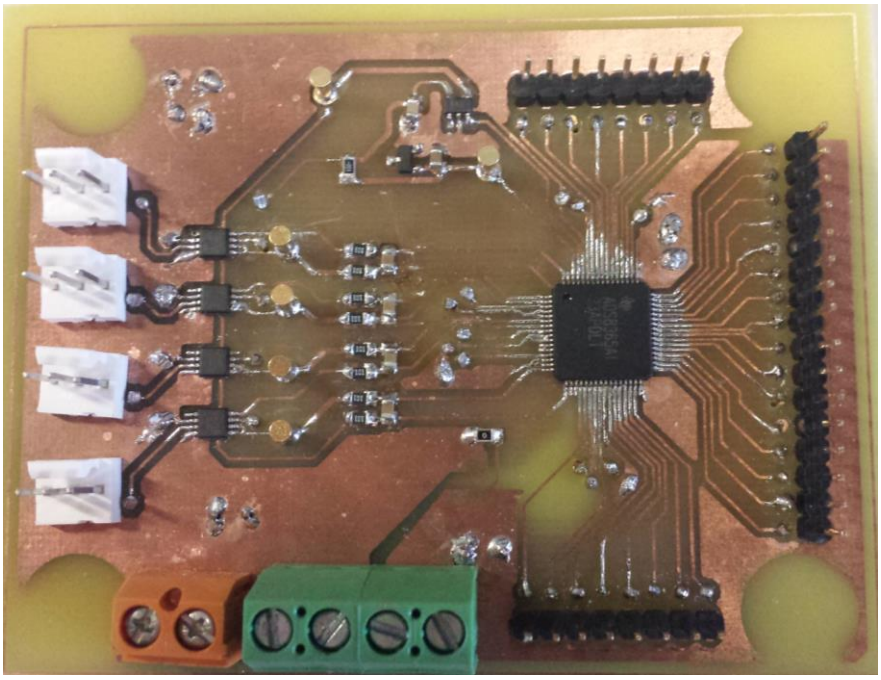


Ilustración 2-6: Placa ADS8365 finalizada

2.2.3 Configuración

El integrado ADS8365 tiene accesibles 16 líneas de configuración. De ellas no todas nos serán útiles para nuestro desarrollo así que explicaremos las que sí debemos usar.

- CS y RD: Estas líneas son activas a nivel bajo y ambas deberán activarse para poder realizar una lectura en alguno de los canales.
- EOC: Activa a nivel bajo. Nos indica que tenemos un nuevo dato listo para poder leer.
- FD: Activa a nivel bajo. En el caso de que estemos usando nuestro integrado en modo FiFo (convertir

todos los canales de manera secuencial) indica que tenemos un dato en el primer canal.

- BYTE: Permite configurar el integrado en modo 8-bits.
- RESET: Activo a nivel bajo. Reinicia el sistema.
- HOLDA/HOLDB/HOLDC: Activos a nivel bajo. Permiten seleccionar que canales queremos convertir. Hay que comentar que el dispositivo reparte los canales en tres pares: A0, A1, B0, B1, C0 y C1. Con estas señales podemos controlar qué par queremos convertir.
- A0/A1/A2: Permite seleccionar qué canales queremos leer. Su selección se lleva a cabo de la siguiente manera:

A2	A1	A0	CHANNEL TO BE READ
0	0	0	CH A0
0	0	1	CH A1
0	1	0	CH B0
0	1	1	CH B1
1	0	0	CH C0
1	0	1	CH C1
1	1	0	Cycle mode reads registers CH A0 to CH C1 on successive transitions of the read line
1	1	1	FIFO mode

Ilustración 2-7: Selección de canales a leer

- CLK: Señal de reloj.

En el Anexo ***Máscaras de configuración*** podemos encontrar todas las máscaras necesarias para las diferentes acciones que nos pueden interesar.

3 ZEDBOARD

El elemento principal de nuestro sistema es un SoC de Xilinx con el chip Zynq-7000 que recibe el nombre de Zedboard [6], en concreto, el modelo Zynq XC7Z020-1CLG484.

3.1 Características generals

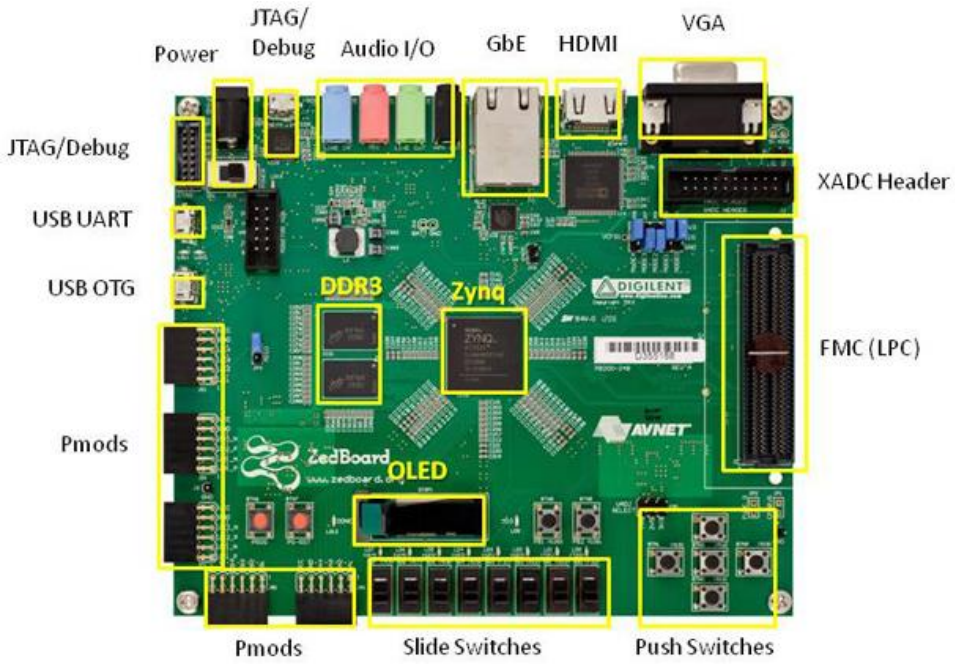


Ilustración 3-1: Zedboard elementos

Entre los dispositivos de los que se compone dicho chip se encuentran dos núcleos ARM Cortex-A9 a una frecuencia de 667Mhz cada uno. La parte lógica, también referenciada como FPGA, es equivalente al modelo Artix-7 de Xilinx con 85.000 celdas lógicas, 53.200 LUTs y 106.400 Flips-Flops. Además, posee una memoria RAM DDR3 de 512 MB.

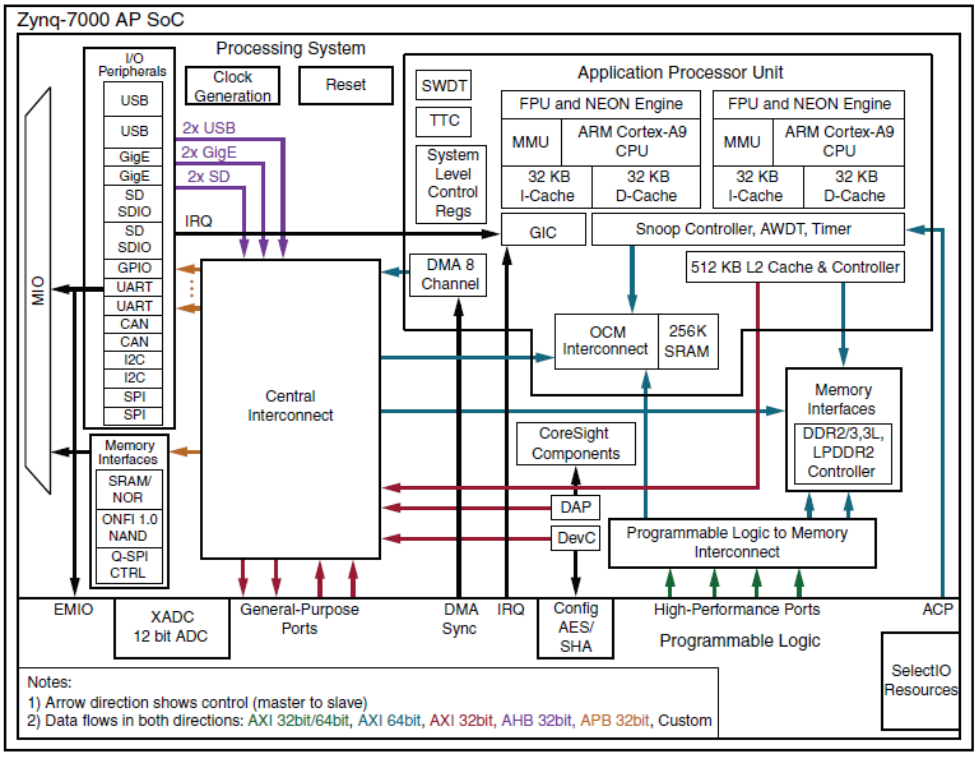


Ilustración 3-2: Zedboard bloques funcionales

3.2 Recursos a usar

No todos los recursos disponibles los utilizaremos en el desarrollo del proyecto. Los que nosotros usaremos principalmente son:

- Núcleos ARM Cortex-A9: Usaremos los dos. En el primero ejecutaremos un sistema operativo Linux personalizado para nuestra aplicación, lo crearemos nosotros mismos. En el segundo instalaremos las aplicaciones necesarias para la gestión de los datos del convertidor.

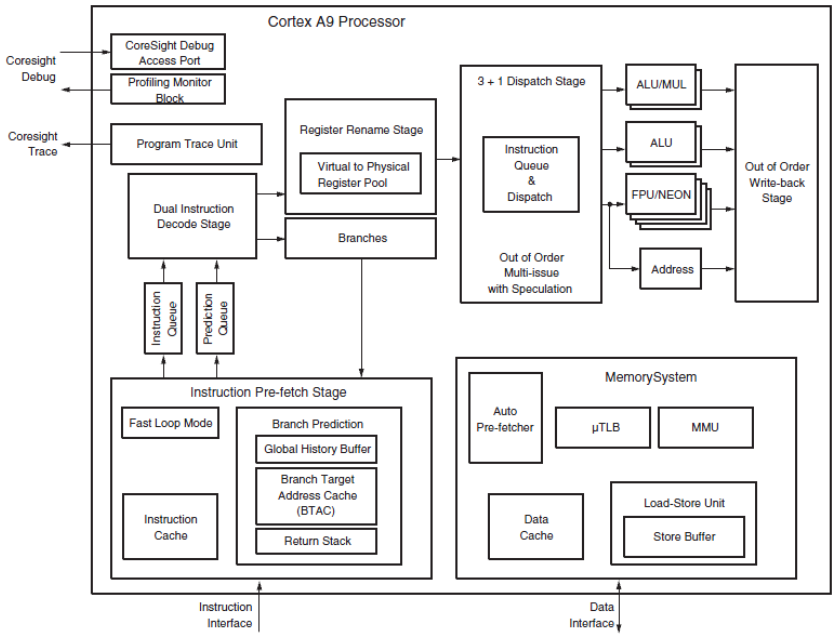


Ilustración 3-3: Arquitectura Cortex-A9

- Dispositivo para tarjetas SD: En una tarjeta SD incluiremos todos los archivos necesarios para la ejecución tanto del sistema operativo como de las diferentes aplicaciones desarrolladas.

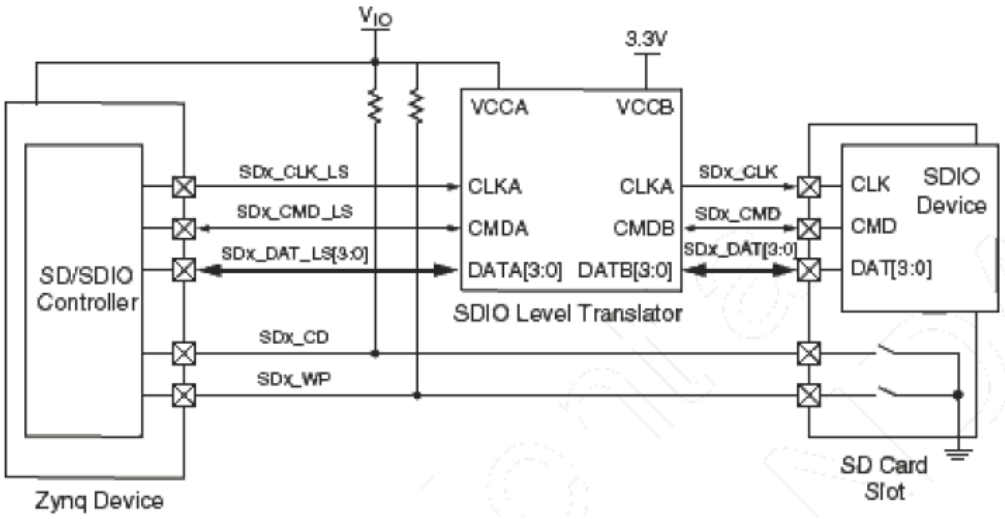


Ilustración 3-4: Interfaz SD

- USB UART: Usaremos esta interfaz para la comunicación entre el PC y la Zedboard. Con esto podremos realizar la ejecución del sistema operativo Linux y trabajar con él.
- XADC Header: No usaremos todo el conector como tal, pero sí uno de sus pines con el fin de obtener una alimentación necesaria para el funcionamiento de la placa de desarrollo del ADS8365.

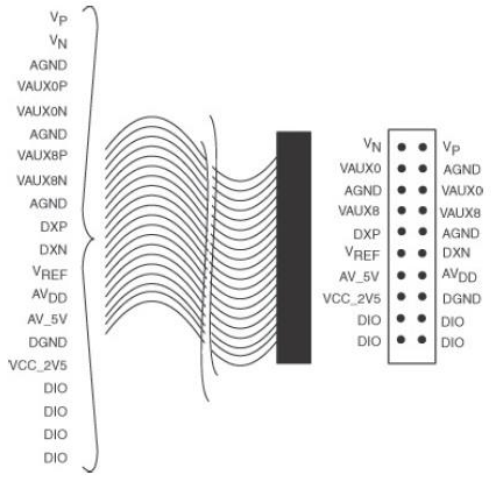


Ilustración 3-5: XADC Header

- DDR3 Memory: Usaremos los dispositivos de memoria para almacenar datos y poder procesarlos para su envío posterior.
- PMODs: Usaremos estos puertos tanto para la obtención de los 16 bits de datos como para la configuración y el control del convertidor.

Pmod	Signal Name	Zynq pin	Pmod	Signal Name	Zynq pin
JA1	JA1	Y11	JB1	JB1	W12
	JA2	AA11		JB2	W11
	JA3	Y10		JB3	V10
	JA4	AA9		JB4	W8
	JA7	AB11		JB7	V12
	JA8	AB10		JB8	W10
	JA9	AB9		JB9	V9
	JA10	AA8		JB10	V8

Pmod	Signal Name	Zynq pin	Pmod	Signal Name	Zynq pin
JC1 Differential	JC1_N	AB6	JD1 Differential	JD1_N	W7
	JC1_P	AB7		JD1_P	V7
	JC2_N	AA4		JD2_N	V4
	JC2_P	Y4		JD2_P	V5
	JC3_N	T6		JD3_N	W5
	JC3_P	R6		JD3_P	W6
	JC4_N	U4		JD4_N	U5
	JC4_P	T4		JD4_P	U6

Pmod	Signal Name	Zynq pin	MIO
JE1 MIO Pmod	JE1	A6	MIO13
	JE2	G7	MIO10
	JE3	B4	MIO11
	JE4	C5	MIO12
	JE7	G6	MIO0
	JE8	C4	MIO9
	JE9	B6	MIO14
	JE10	E6	MIO15

Ilustración 3-6: PMODs

- Jumpers: Son necesarios para poder configurar la Zedboard para que funcione de manera correcta para cumplir las especificaciones necesarias para nuestra aplicación.

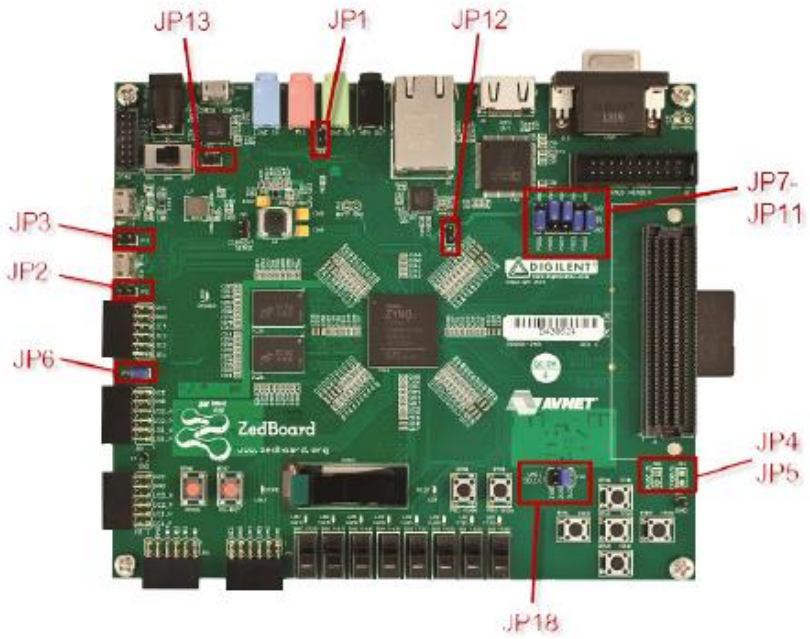


Ilustración 3-7: Jumpers de configuración

- Otros pines: Usaremos algunos pines aislados para la obtención de alguna alimentación y tierras para el sistema global.

3.3 Desarrollo hardware

Nos encontramos ante una placa de desarrollo que dispone de una gran cantidad de lógica programable que debemos adecuar a nuestro funcionamiento. Esta configuración consiste en la modificación interna de la placa para obtener el acceso a los recursos que necesitamos para la realización de nuestro proyecto.

Este desarrollo lo haremos con la ayuda del programa VIVADO [7] en su versión v2014.4 de Xilinx. Con este programa podremos realizar las diferentes conexiones internas para tener acceso a los recursos. Con la ayuda de este programa podremos realizar diferentes tareas.

Por un lado, tendremos una simulación del sistema en el que podremos realizar una serie de pruebas para comprobar cómo se debería comportar el sistema real una vez que lo hagamos funcionar.

Por otro lado, una vez el sistema completo este definido y configurado podremos exportarlo y obtener el archivo necesario (bitstream) para poder ejecutar el sistema completo.

Además, podremos enlazar nuestra configuración hardware con otro software, el SDK de Xilinx, para poder realizar las diferentes aplicaciones que queramos incluir en nuestro sistema.

3.3.1 Creación de un proyecto

Para poder iniciar un proyecto debemos hacer unos pasos iniciales para poder ligar nuestro diseño a la placa determinada que estemos usando. Comenzaremos por crear un proyecto nuevo dándole a ***Create New Project***.

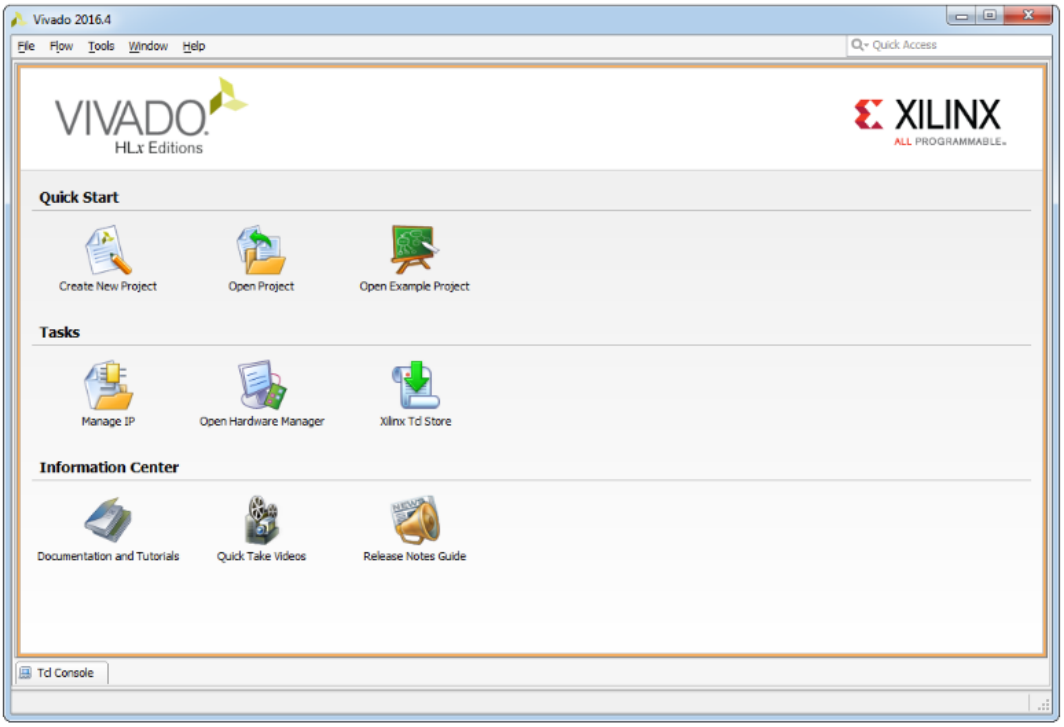


Ilustración 3-8: VIVADO Proyecto nuevo 1

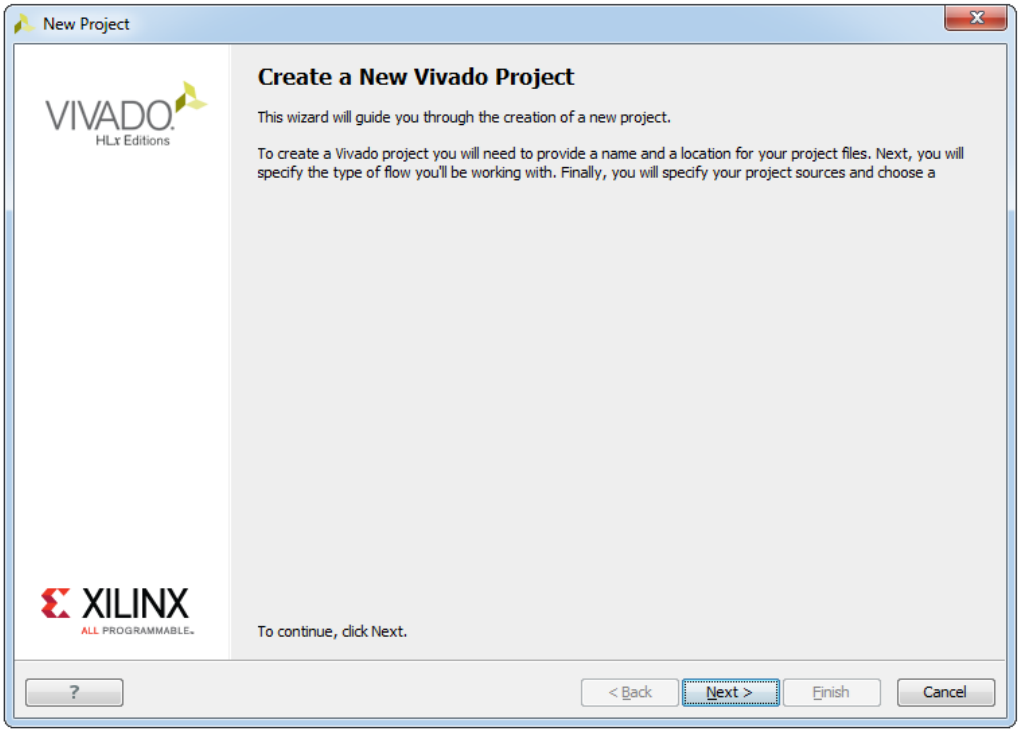


Ilustración 3-9: VIVADO Proyecto nuevo 2

A continuación, debemos elegir el nombre del proyecto, así como, la localización que queramos que tenga dentro de nuestro ordenador.

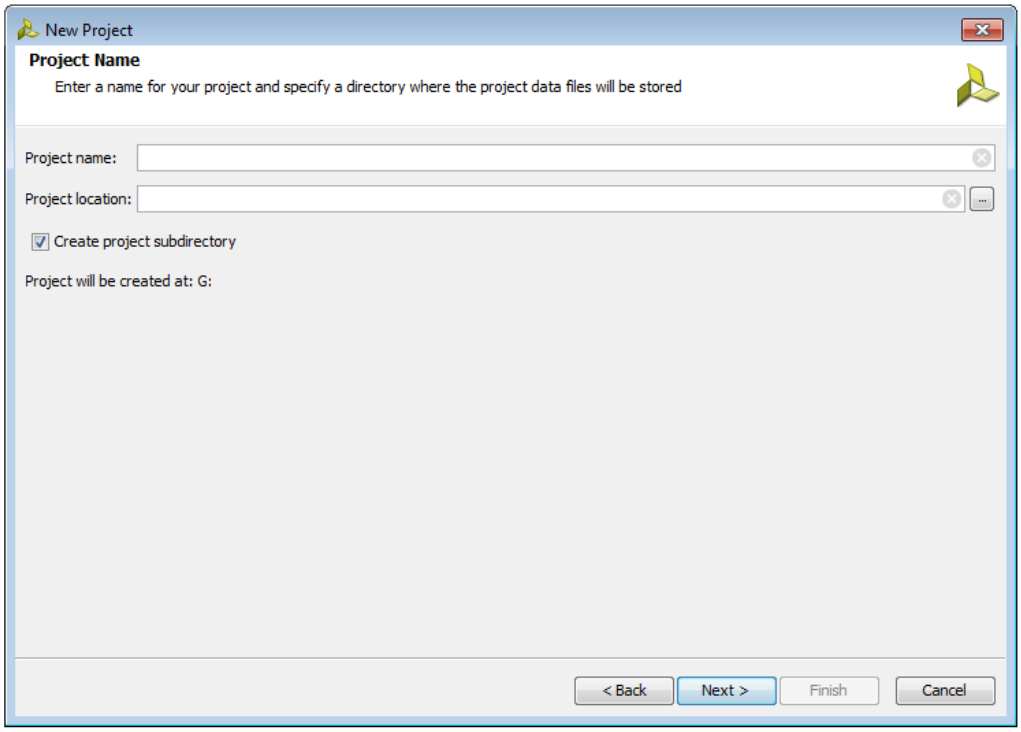


Ilustración 3-10: VIVADO Proyecto nuevo 3

El siguiente paso es indicar el tipo de proyecto que vamos a realizar. En nuestro caso elegiremos ***RTL Project***

para que así podamos elegir los diferentes componentes y poder personalizarlo ajustándonos a nuestras necesidades.

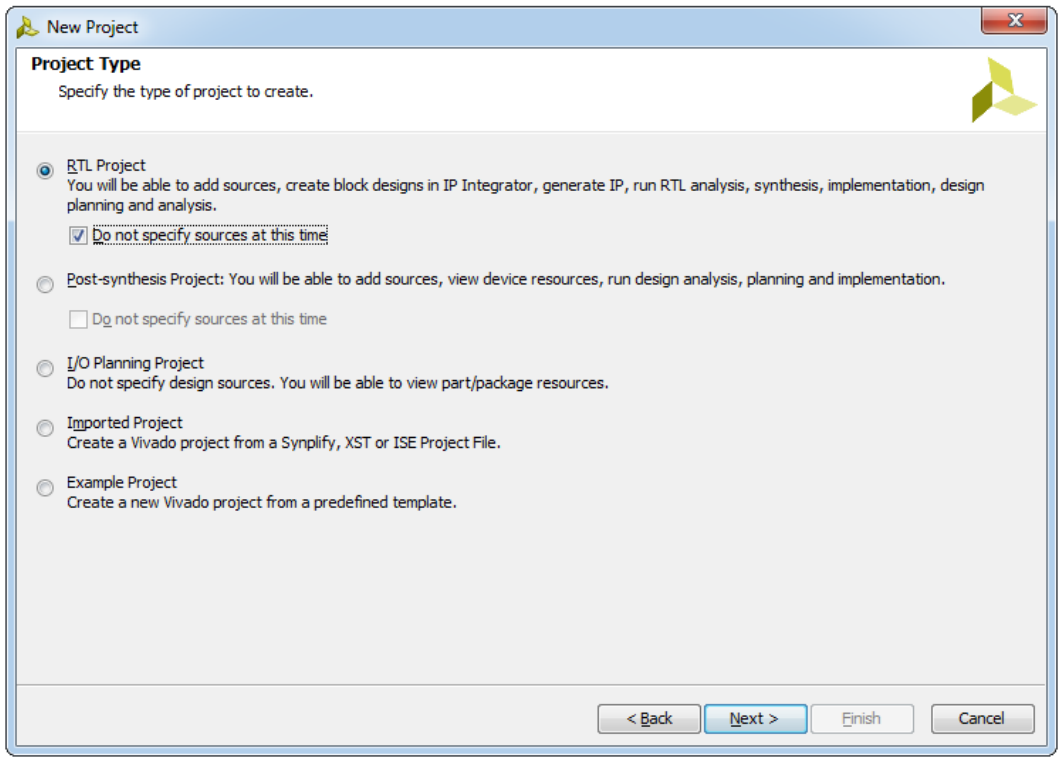


Ilustración 3-11: VIVADO Proyecto nuevo 4

Por último, solo nos queda seleccionar la placa con la que estamos trabajando para que la configuración final sea la adecuada. En nuestro caso, seleccionaremos la ZedBoard.

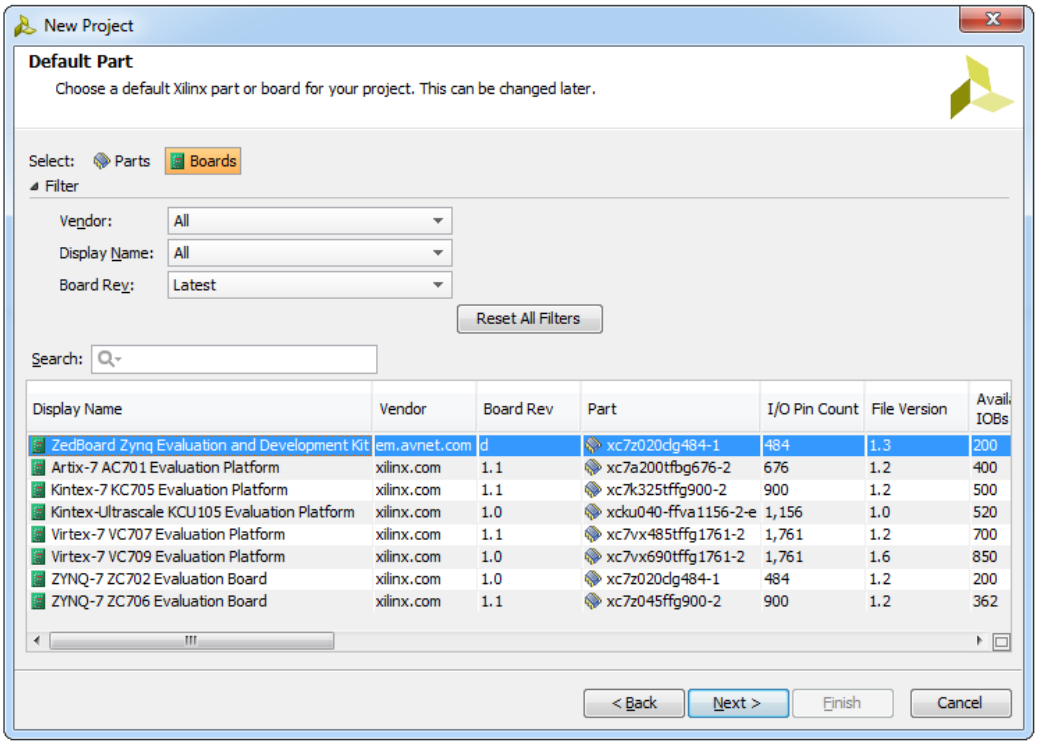


Ilustración 3-12: VIVADO Proyecto nuevo 5

Con esto queda finalizada la parte de la creación del proyecto y nos aparecerá la pantalla principal del programa donde podremos empezar a trabajar.

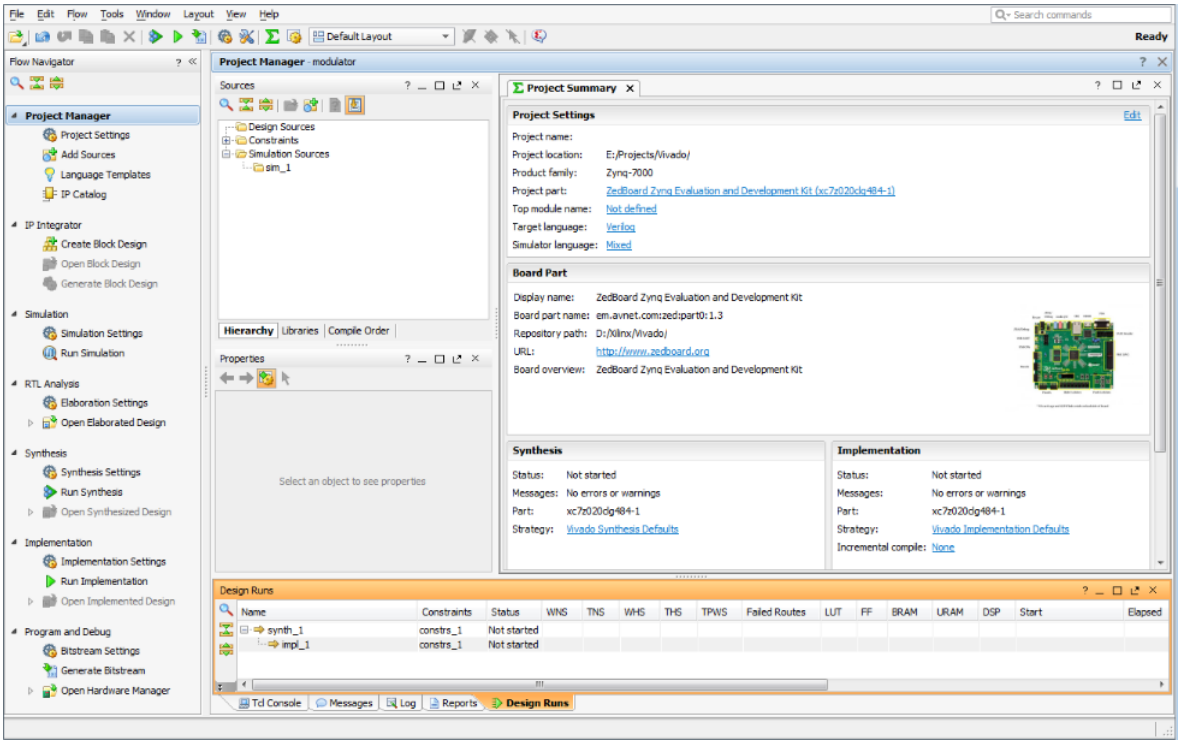


Ilustración 3-13: VIVADO Pantalla principal

En esta pantalla encontraremos diferentes funcionalidades. En la parte izquierda tenemos un menú que nos servirá para la realización de los diferentes pasos del proceso de desarrollo. Además, podemos ver mucha información acerca del mismo, desde un resumen hasta los diferentes elementos que lo componen.

3.3.2 Selección de recursos

Para poder ir seleccionando los recursos necesarios debemos ir al menú de la izquierda y pulsar sobre Create Block Design. Esto nos abrirá un nuevo panel donde podremos ir añadiendo y realizando las conexiones necesarias para nuestro sistema.

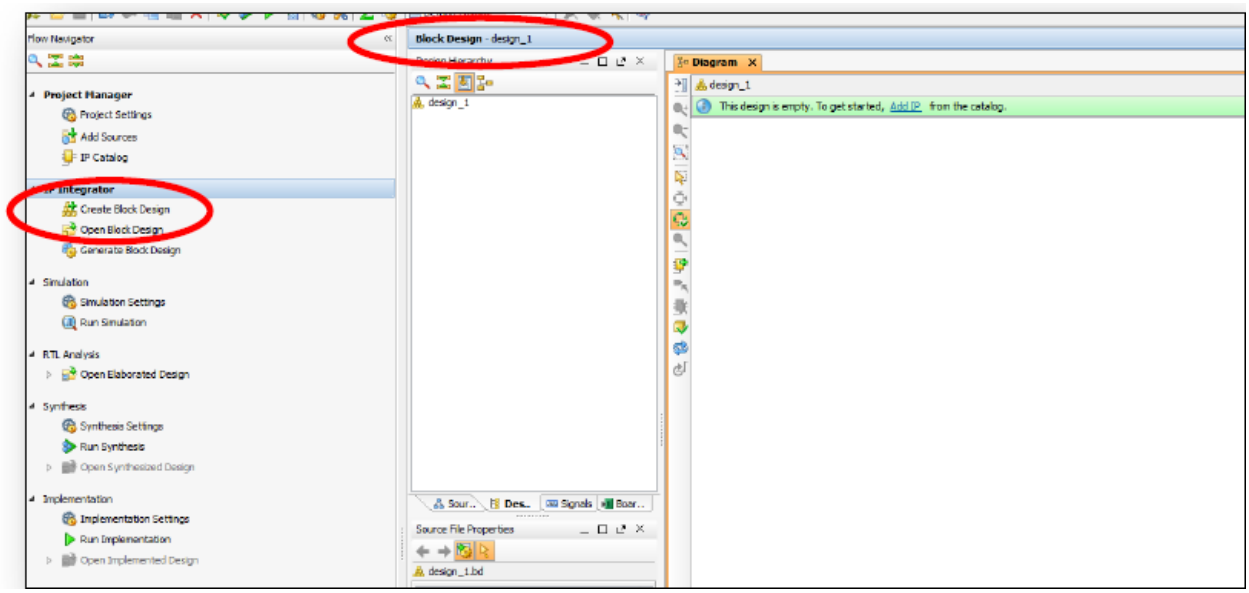


Ilustración 3-14: VIVADO Create Block Design

Para ir añadiendo los diferentes recursos debemos pulsar con el botón derecho sobre el panel de diseño y seleccionar la opción ***Add IP***. También podemos pulsar directamente sobre el icono que aparece en la parte izquierda de este panel.

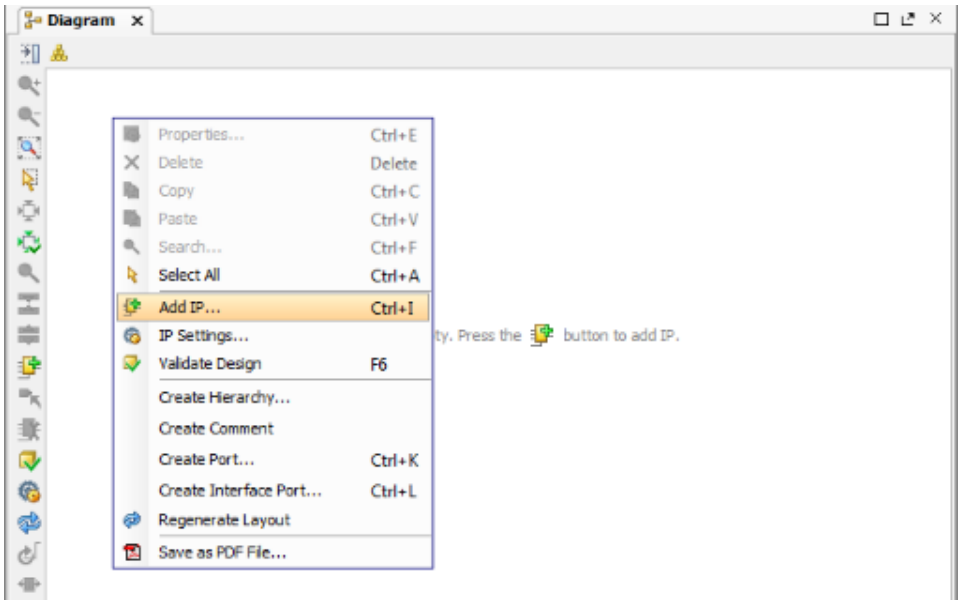


Ilustración 3-15: VIVADO Add IP

El primer elemento que debemos buscar para añadir es ***ZYNQ7 Processing System***. Se trata del núcleo principal

del sistema y al que conectaremos el resto de recursos.

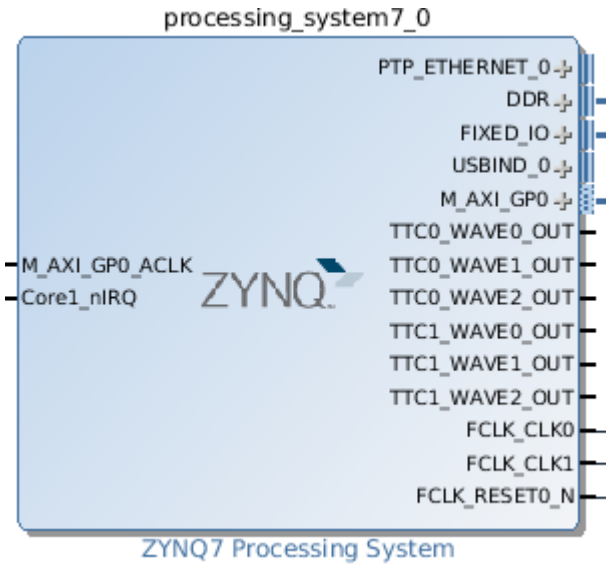


Ilustración 3-16: VIVADO Processing System

A continuación, debemos añadir el componente ***Processor System Reset*** que nos servirá para la gestión del RESET del sistema en cada uno de sus puntos.

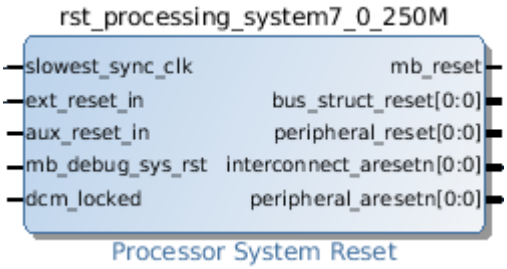


Ilustración 3-17: VIVADO Processor System Reset

Para tener accesibles las líneas de datos y configuración en los PMODs deberemos añadir un elemento intermedio que nos ayudará a la gestión de los mismos. Buscaremos ***AXI Interconnect*** y ***AXI GPIO***. De este último debemos añadir dos elementos iguales y así diferenciaremos las líneas de datos y las líneas de configuración.

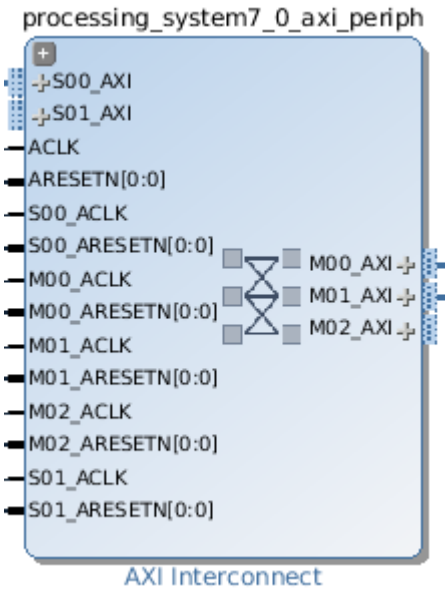


Ilustración 3-18: VIVADO AXI Interconnect

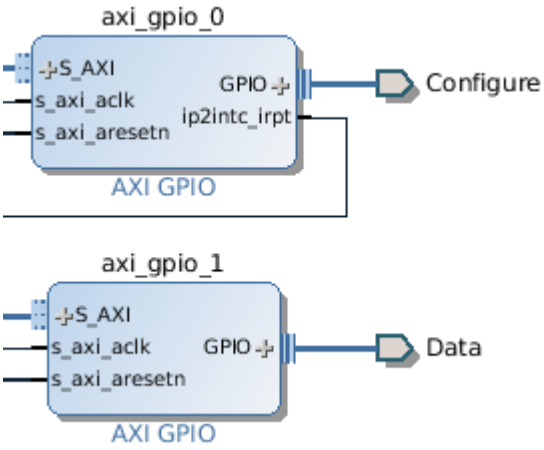


Ilustración 3-19: VIVADO AXI GPIO

El siguiente paso a realizar es la interconexión de los elementos. Para realizar este proceso consultaremos el Anexo VIVADO ***Diagrama de Bloques*** y observaremos las conexiones que debemos hacer, así como, las diferentes terminaciones extras que debemos tener como la salida para la memoria DDR o el reloj que usaremos para el ADC.

Una vez tenemos el diagrama de bloques completo el siguiente paso es realizar las pertinentes configuraciones de los mismos. Empezaremos con el núcleo principal del sistema haciendo doble click sobre el bloque llamado ZYNQ7 Processing System. Esto nos abrirá un cuadro de diálogo donde podremos hacer las diferentes configuraciones. En este caso nos encargaremos de configurar las interrupciones y la señal de reloj que usaremos para el control del ADC.

Para la gestión de interrupciones pulsaremos sobre el menú de la izquierda en la sección ***Interrupts*** y marcaremos las siguientes casillas.

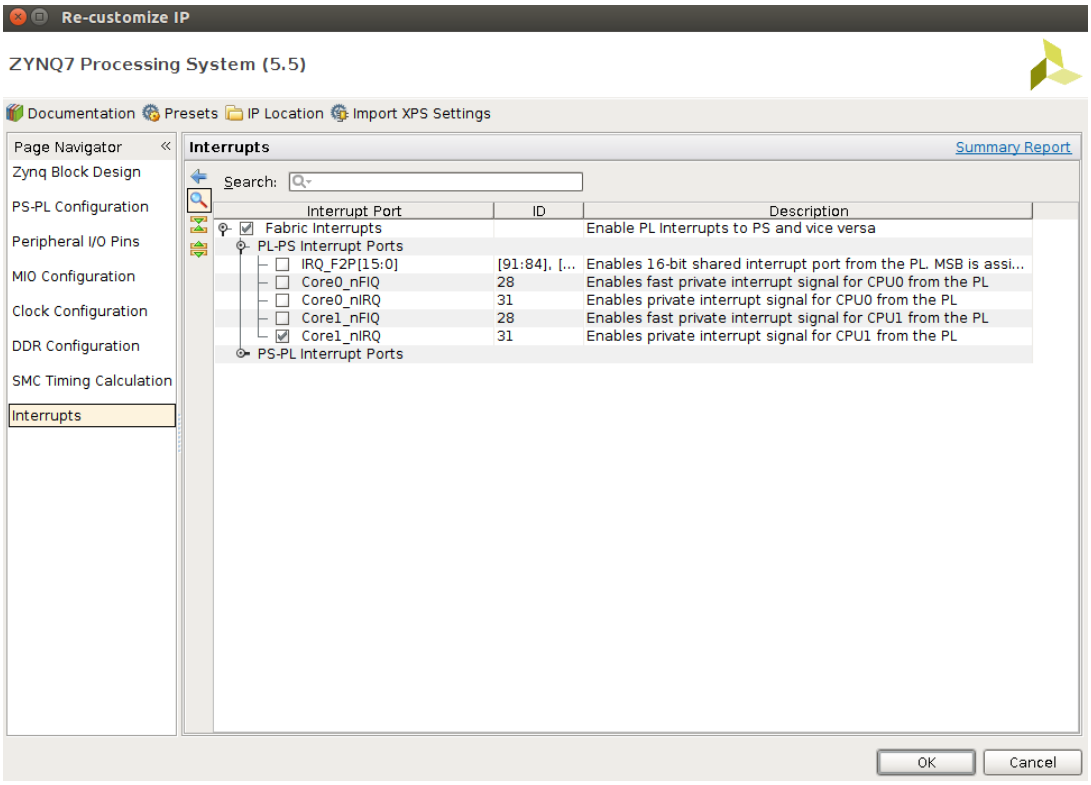


Ilustración 3-20: VIVADO Interrupciones

La placa de desarrollo ZedBoard consta de dos temporizadores y nosotros haremos uso de uno de ellos para proporcionar al ADC una señal de reloj. Tendremos que configurarlo para obtener una señal de 5 Mhz. Para ello nos vamos a la parte de ***Clock Configuration*** y lo configuraremos.

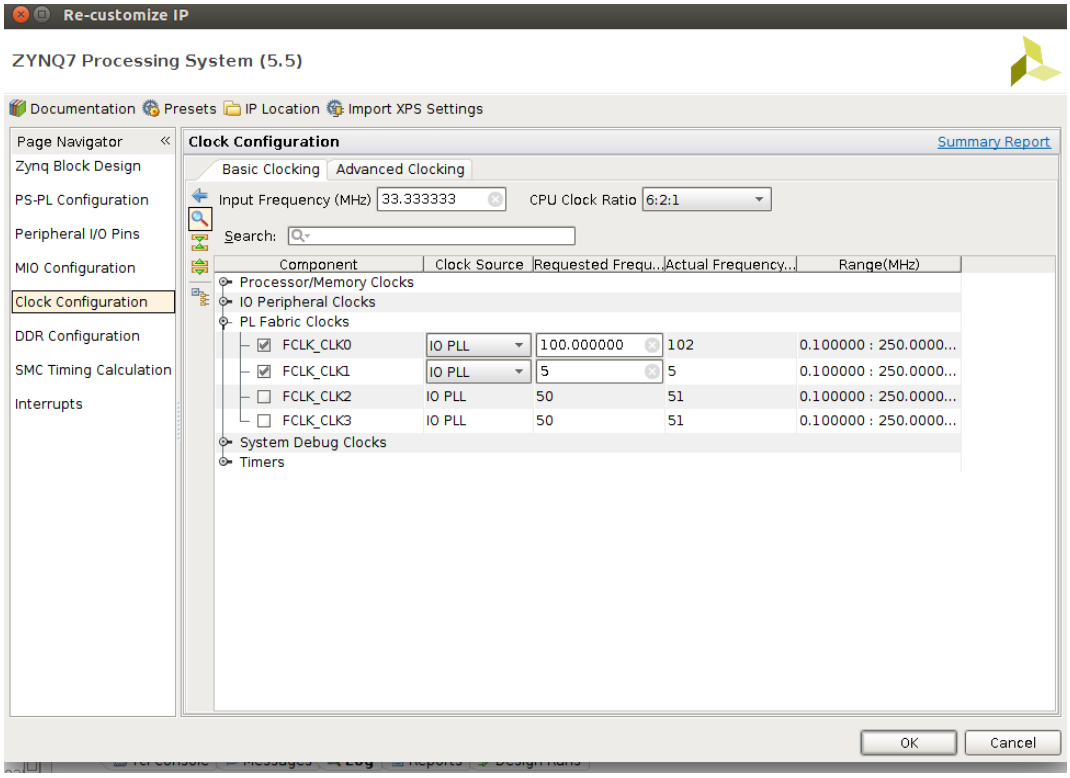


Ilustración 3-21: VIVADO Señal CLK

Por último, debemos entrar en cada uno de los dos AXI GPIO para asignar la relación entre los pines del ADC y los pines de la FPGA de manera correcta.

Pines del ADC (CONFIGURE)	Pines de la FPGA (CONFIGURE)	Pines del ADC (DATA)	Pines de la FPGA (DATA)
HOLDC	AA8(JA10)	D0	V4(JD2N)
HOLDB	AB9(JA9)	D1	V5(JD2P)
HOLDA	AB10(JA10)	D2	W7(JD1N)
A0	AB11(JA7)	D3	V7(JD1P)
A1	AA9(JA4)	D4	U5(JD4N)
A2	Y10(JA3)	D5	U6(JD4P)
ADD	AA11(JA2)	D6	W5(JD3N)
RESET	Y11(JA1)	D7	W6(JD3P)
NAP	V8(JB10)	D8	AA4(JC2N)
BYTE	V9(JB9)	D9	Y4(JC2P)
FD	W10(JB8)	D10	AB6(JC1N)
EOC	V12(JB7)	D11	AB7(JC1P)
CLK	W8(JB4)	D12	U4(JC4N)
RD	V10(JB3)	D13	T4(JC4P)
WR	W11(JB2)	D14	T6(JC3N)
CS	W12(JB1)	D15	R6(JC3P)

Ilustración 3-22: Asignación de Pines

Con esto tendremos el sistema completo, configurado y listo.

3.3.3 Finalizar Proyecto

Con el diseño completo solo quedan los últimos pasos para finalizar el proyecto. Para comprobar de que todo va correctamente en primer lugar tendremos que validar el diseño. Para ello, en la barra de herramientas superior tendremos que ir a ***Tools/Validate Design*** o bien pulsar ***F6***. El siguiente paso es guardar el diseño. Para ello iremos a ***File/Save Block Design***.

Ahora debemos crear un archivo .vhd que tendrá toda la información en lenguaje VHDL sobre la configuración lógica de la FPGA (véase Anexo ***main_wrapper.vhd***). Para ello necesitamos ir a la pantalla ***Sources***, buscar la carpeta ***Design Sources*** y buscar un archivo de extensión .bd. Pulsaremos con el botón derecho sobre dicho archivo y elegiremos la opción ***Create HDL Wrapper*** para, posteriormente, seleccionar ***Let Vivado manage wrapper and auto-update***.

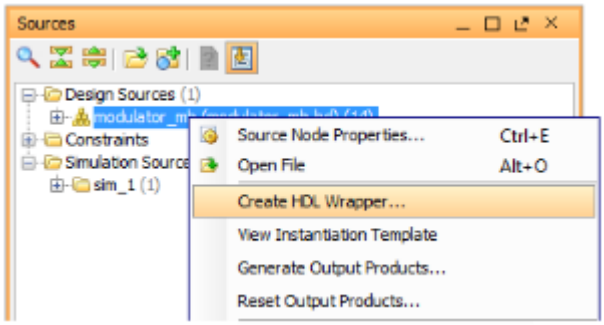


Ilustración 3-23: VIVADO Create HDL Wrapper

Para completar ya solo queda hacer la síntesis del circuito, la implementación y generar el archivo bitstream. Para ello podemos ir seleccionando cada uno de los pasos o ir directamente a Generate Bitstream que esto hará que se pase por los tres procesos obligatoriamente. Si no existe ningún error podremos continuar con el desarrollo.

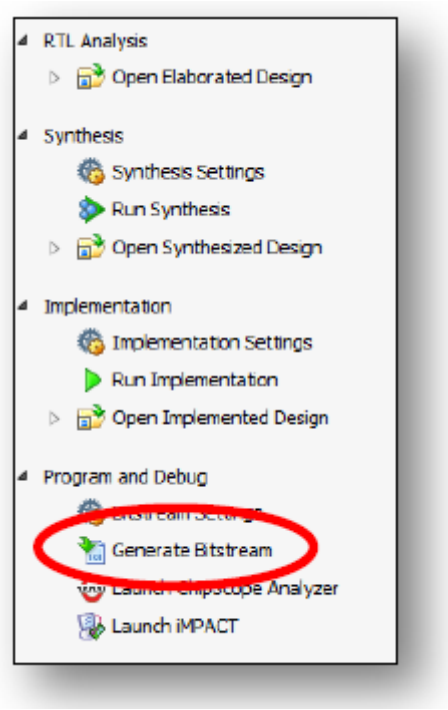


Ilustración 3-24: VIVADO Generar Bitstream

4 CONEXIONADO

Para concluir con la parte física del proyecto solo nos queda la conexión de los diferentes elementos del sistema completo. Podemos suponer la ZedBoard como el núcleo del sistema y enfocaremos este proceso de conexionado desde el punto de vista de la misma.

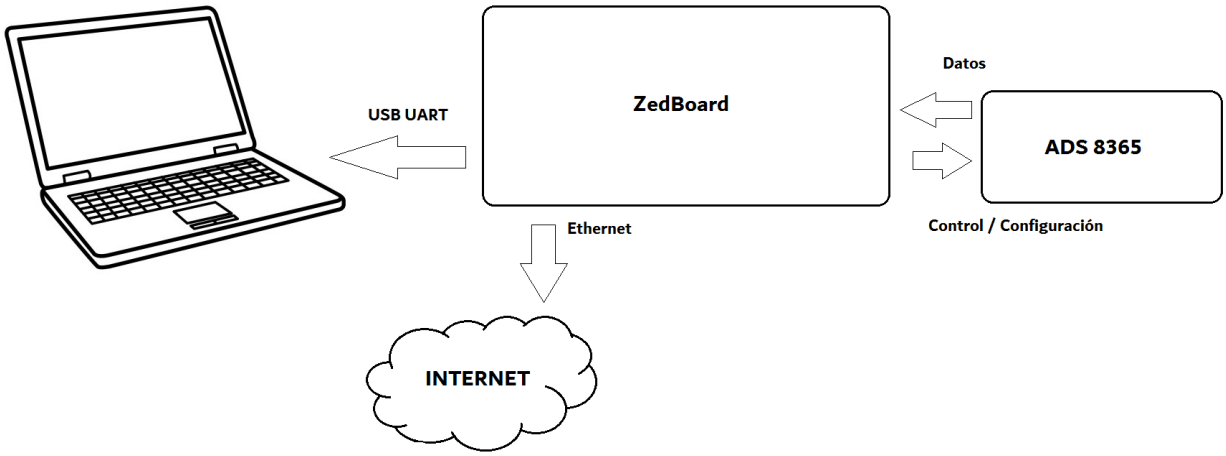


Ilustración 4-1: Interconexión sistema completo

- ZedBoard-PC: Esta conexión se hace a través del puerto UART de la Zedboard y el puerto USB del ordenador. Esto se realiza para mediante un terminal serie en el ordenador poder ejecutar el sistema Linux que se encontrará ejecutándose en uno de los núcleos de la ZedBoard.

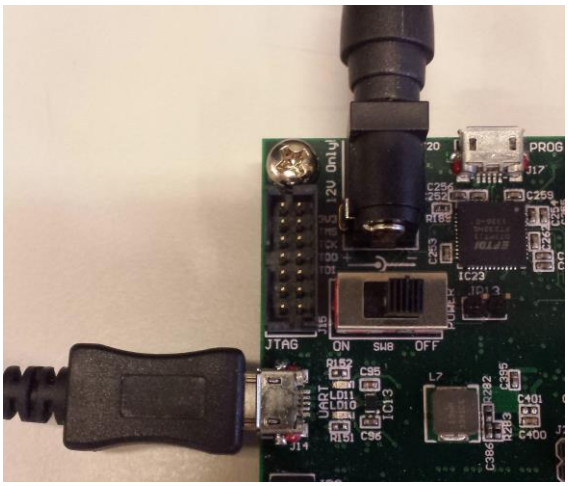


Ilustración 4-2: UART

- ZedBoard-Internet: Para la conexión de la ZedBoard a internet usaremos el puerto Ethernet del que dispone la ZedBoard.

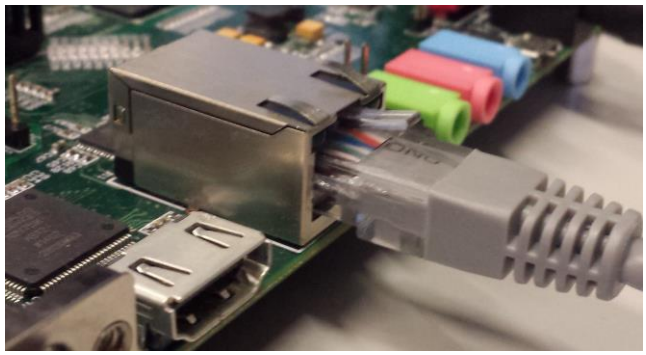


Ilustración 4-3: Ethernet

- ZedBoard-ADS8365: Entre ambos dispositivos tendremos que conectar muchas señales. La placa de desarrollo del ADS8365 necesita varias alimentaciones diferentes, una para la parte analógica y otra para la parte digital, y para resolver esto lo que haremos será sacar dichas alimentaciones de la placa de la Zedboard. Explorando los diferentes pines de los que dispone dicha placa se pueden encontrar los 3V necesarios para la parte digital en uno de los PMODs marcado con VCC.

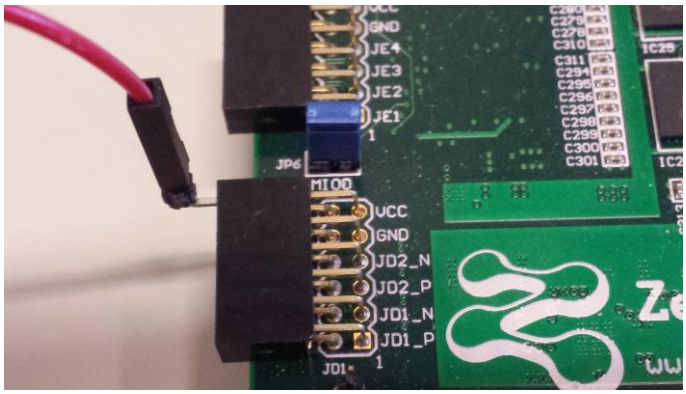


Ilustración 4-4: VCC 3V

Para sacar los 5V necesarios para la parte analógica recurriremos al conector XADC Header que en uno de sus pines tiene esa tensión accesible.

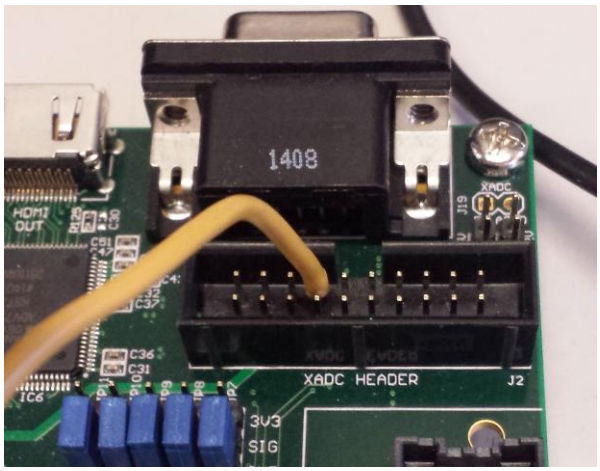


Ilustración 4-5: VCC 5V

Necesitaremos además un pin de tierra (GND) para unificar las mismas entre las dos placas y para ello recurriremos a alguno de los pines de los que dispone la ZedBoard para este fin.

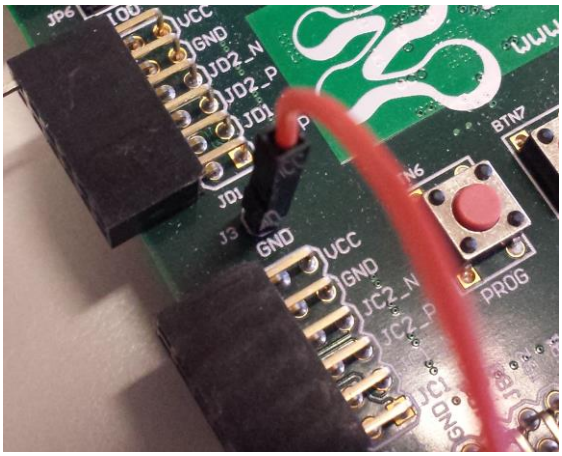


Ilustración 4-6: GND

Por último, debemos conectar tanto las líneas correspondientes a los datos como las líneas de control y configuración del ADS8368. Para esta conexión haremos uso de los PMODs que configuramos en el proceso de desarrollo hardware poniendo mucha atención a la hora de conectarlos y siguiendo el siguiente esquema.

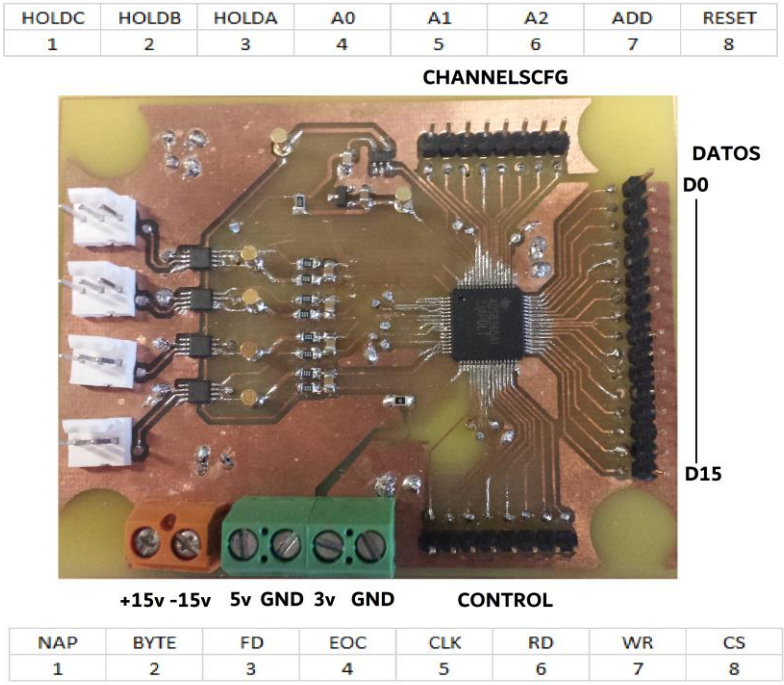


Ilustración 4-7: Conexiones ADS8365

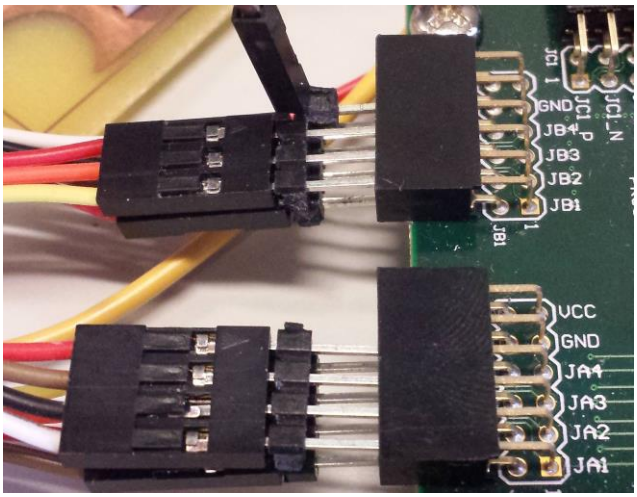


Ilustración 4-8: Conexiones Control y Configuración

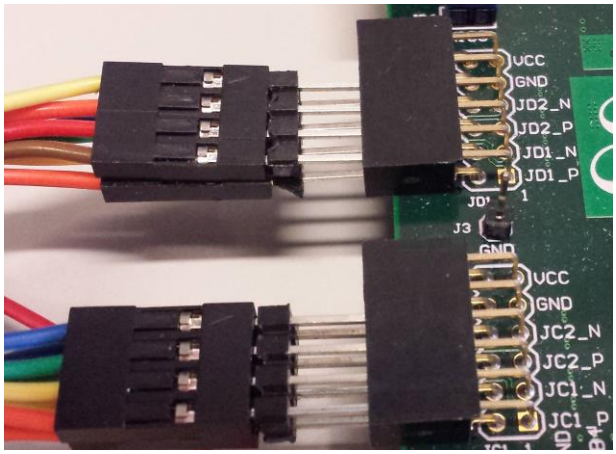


Ilustración 4-9: Conexiones Datos

Por último, solo debemos conectar la ZedBoard a la red de alimentación mediante el transformador que se incluye con la placa. Todo el sistema está conectado de manera correcta y listo para empezar a funcionar.

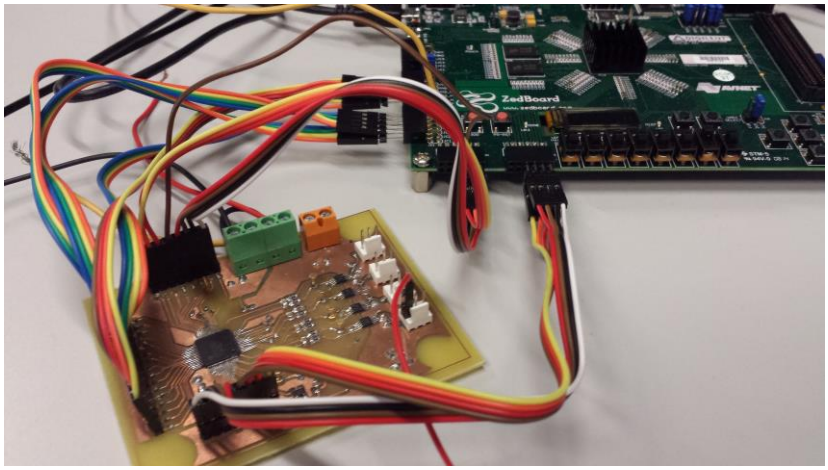


Ilustración 4-10: Conexión ZedBoard – ADS8365

5 DESARROLLO SOFTWARE

El uso de diferentes plataformas para el desarrollo del proyecto hace obligatorio el trabajo a nivel de software de cada una de estas. Desde el punto de vista de la programación el sistema podemos centralizarlo de nuevo en la ZedBoard. Esta placa posee dos núcleos ARM Cortex-A9 que usaremos para funcionalidades completamente diferentes.

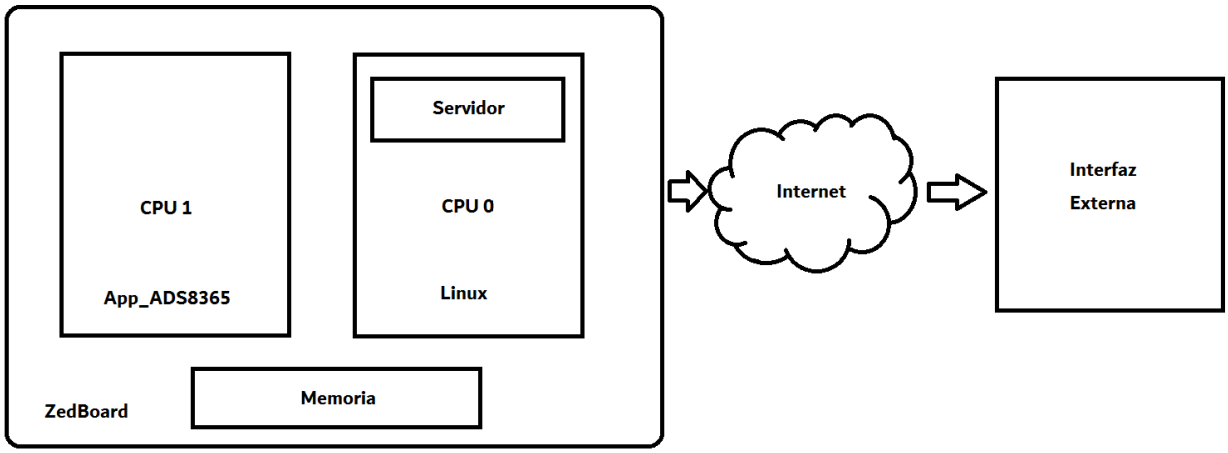


Ilustración 5-1: Desarrollo Software Sistema

En un primer núcleo, al que llamaremos CPU 0, ejecutamos un sistema operativo Linux personalizado y, sobre él, la aplicación que nos ayudará a gestionar el servidor. Este servidor se usará para la comunicación de los datos, vía internet, con una interfaz externa que también deberá ser programada.

En el segundo núcleo, CPU 1, programaremos una aplicación que se encargará de la gestión de la placa del ADS8365.

Aunque los dos núcleos hacen un trabajo paralelo y diferente, al final, trabajarán con los mismos datos. Toda esta información compartida estará contenida en la memoria de la ZedBoard para que ambos puedan acceder a ella.

5.1 LINUX

El objetivo de usar un sistema operativo Linux en nuestro desarrollo es el de tener una plataforma con muchas posibilidades que nos ayude a resolver problemas como la comunicación vía internet o la ejecución de otras aplicaciones que queramos desarrollar. Con objeto de optimizar el diseño no nos centraremos en la instalación de una distribución existente en el mercado de este sistema operativo, sino que realizaremos nuestra propia versión de un sistema operativo Linux muy ligero pero que cumpla todas las necesidades que tengamos.

5.1.1 Compilación

Para el correcto funcionamiento de nuestro sistema necesitamos un sistema operativo Linux que no ocupe muchos recursos y que además se ejecute al inicio desde una tarjeta SD que introduciremos en la ZedBoard. Además, como este sistema será el que se inicie al arrancar la placa debemos incluir también la aplicación que se ejecutará de manera paralela en el otro núcleo para que funcione de manera transparente al usuario. Para poder ejecutar de esta manera nos serviremos de los archivos fuente del kernel de Linux para poder modificarlos y ajustarlos a nuestras necesidades. Es importante tener en cuenta el control de versiones ya que alguno de los procedimientos puede ser ligeramente diferente si se cambia alguna de ellas. En nuestro desarrollo usaremos el ***kernel de Linux*** correspondiente a la versión ***3.14.4***. Por otro lado, para poder realizar la modificación y la

posterior compilación tendremos que instalar un software llamado ***Petalinux 2014.4*** [8] y todo el trabajo lo realizaremos bajo un sistema operativo Linux en este caso con la distribución ***Ubuntu 14.04 LTS*** [9] con un núcleo ARM de 64 bits.

Para comenzar el desarrollo necesitaremos descargar tres archivos fuente [10] con los que trabajaremos y los cuales serán necesarios para la ejecución. Estos son:

- ***linux-xlnx-xilinx-v14.4.zip***
- ***u-boot-xlnx-xilinx-v14.4.zip***
- ***uramdisk.image.gz***

Los dos primeros son archivos genéricos pero el último es un archivo de imagen que depende de la plataforma sobre la que vayamos a ejecutar nuestro sistema, en este caso, ZedBoard.

Al final del proceso de compilación y configuración debemos tener una serie de archivos que depositaremos en la tarjeta de memoria SD y que harán posible la ejecución del sistema y de las distintas aplicaciones. Estos archivos necesarios son:

- ***uImage***
- ***devicetree.dtb***
- ***uramdisk.image.gz***
- ***BOOT.bin***
- ***s_control.elf*** (Esta será la aplicación que queremos ejecutar sobre Linux)

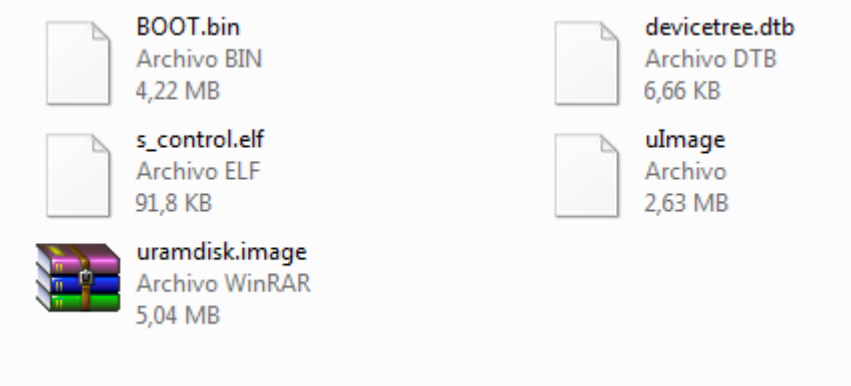


Ilustración 5-2: Archivos tarjeta SD

Una vez tengamos los archivos fuente descargados y el software necesario instalado empezaremos por generar los diferentes archivos.

NOTA: Es recomendable antes de iniciar el desarrollo ejecutar las variables de entorno. Para ello, según dónde tengamos instalados nuestro software, ejecutaremos:

- *source /opt/xilinx/Vivado/.../settings64.sh*
- *source /opt/xilinx/ISE/.../settings64.sh*

5.1.1.1 uImage

Trabajaremos desde una ventada de terminal Linux. En primer lugar, deberemos descomprimir el archivo fuente ***linux-xlnx-xilinx-v14.4.zip***, dirigirnos hacia el directorio del archivo y escribir una serie de comandos. Todo el proceso debe daros como resultado la aparición de un archivo con el nombre ***uImage***.

Procedimiento:

I. *unzip linux-xlnx-xilinx-v14.4.zip*

II. *cd linux-xlnx-xilinx-v14.4*

III. *export CROSS_COMPILE=arm-xilinx-eabi-*

IV. *make ARCH=arm Xilinx_zynq_defconfig*

V. *make ARCH=arm menuconfig*

VI. *(EXIT)*

VII. *make ARCH=arm UIMAGE_LOADHDDR=0x8000 uImage*

Una vez ejecutados todos los pasos podremos encontrar el archivo ***uImage*** en el directorio ***linux-xlnx-xilinx-v14.4/arch/arm/boot***.

5.1.1.2 devicetree.dtb

Para poder obtener este archivo volvemos a recurrir a un terminal de Linux. Debemos colocarnos en el directorio donde hemos descomprimido el archivo ***linux-xlnx-xilinx-v14.4.zip*** y realizar lo siguiente:

I. cd arch/arm/boot/dts

II. cp zynq-zed.dts zynq-zed-Suh.dts

Con esta orden haremos una copia para mantener el archivo original y poder hacer las modificaciones necesarias en la copia. Estás modificaciones consisten en indicar que solo usaremos una de las CPUs existentes además de delimitar la memoria que usará está.

```
42      chosen {
43          bootargs = "console=ttyPS0,115200 root=/dev/ram rw ip=:::::eth0:dhcp earlyprintk";
44          linux,stdout-path = "/axi@0/serial@e0001000";
45      } ;
```

Ilustración 5-3: Devicetree Modificación CPU

En la línea 43 debemos añadir antes de ***root***: *maxcpus=1*

```
79      ps7_ddr_0: memory@0 {
80          device_type = "memory";
81          reg = < 0x0 0x20000000 >;
82      } ;
```

Ilustración 5-4: Devicetree Modificación Memoria

En la línea 81 debemos cambiar el valor *0x20000000* por *0x18000000*.

Para poder obtener finalmente el archivo devicetree.dtb volvemos al terminal Linux en el punto en el que lo dejamos y ejecutamos la siguiente línea:

I. *../../../../scripts/dtc/dtc -I dts -O dtb -o devicetree.dtb zynq-zed-Suh.dts*

NOTA: Algunas de las rutas pueden cambiar debido a la instalación en el equipo pero las terminaciones en las rutas deben ser las mismas.

5.1.1.3 uramdisk.image.gz

Este archive no es necesario modificarlo, se copiará directamente a la tarjeta SD y la única precaución como ya comentamos es realizar su descarga teniendo en cuenta la plataforma que estamos usando, ZedBoard.

5.1.1.4 BOOT.bin

Para este archivo necesitaremos generar varios archivos adicionales. Necesitaremos:

- ***main_wrapper.bit***
- ***u-boot.elf***
- ***fsbl.elf***
- ***app_cpu1.elf***

Una vez que tengamos todos estos archivos para poder generar el ***BOOT.bin*** necesitaremos generar un archivo intermedio bootimage.bif cuyo contenido puede depender de las rutas propias de nuestra instalación.

```
1  the_ROM_image:
2  {
3      [bootloader]/home/master4/controller/controller.sdk/fsbl/Debug/fsbl.elf
4      /home/master4/controller/controller.sdk/main_wrapper_hw_platform_2/main_wrapper.bit
5      /home/master4/zed/u-boot-xlnx-xilinx-v14.4/u-boot.elf
6      /home/master4/controller/controller.sdk/app_cpu1/Debug/app_cpu1.elf
7  }
```

Ilustración 5-5: bootimage.bif

Una vez lo tengamos todo preparado solo tendremos que volver al terminal Linux y ejecutar el siguiente código dentro de la carpeta donde hayamos generado el ***bootimage.bif***.

bootgen -image bootimage.bif -o i -boot.bin -w on

5.1.1.4.1 main_wrapper.bit

Este archivo contiene información relacionada con la parte lógica de la FPGA y para obtenerlo solo tenemos que exportar el proyecto que realizamos en Vivado y acordarnos de incluir el archivo bitstream. Cuando haya finalizado la exportación lo encontraremos en la carpeta del proyecto, en nuestro caso, con el nombre de ***main_wrapper.bit***.

5.1.1.4.2 u-boot.elf

Para la generación de este archivo volveremos al terminal de Linux. Nos debemos colocar sobre el directorio donde descargaremos el archivo fuente llamado ***u-boot-xlnx-xilinx-v14.4***.zip y ejecutar los siguientes comandos:

I. *unzip u-boot-xlnx-xilinx-v14.4.zip*
II. *cd u-boot-xlnx-xilinx-v14.4*
III. *export CROSS_COMPILE=arm-xilinx-eabi-*
IV. *make zynq-zed-config*
V. *make*
VI. *cp u-boot u-boot.elf*

Con esto ya tendremos el archivo que buscábamos.

5.1.1.4.3 fsbl.elf

Para generar los archivos restantes trabajaremos con el SDK de Xilinx, para ello, una vez exportado el proyecto en Vivado lanzaremos el SDK directamente desde la opción que existe en el menú como ***Launch SDK***…

En el SDK podremos desarrollar las diferentes aplicaciones que usaremos tanto ejecutándose de manera transparente como es el caso de la ***app_cpu1*** como las que queramos ejecutar nosotros manualmente.

Para obtener la primera, ***fsbl.elf***, simplemente tenemos que crear una aplicación nueva del tipo ***ZYNQ FSBL***

indicando que el procesador será el 0.

5.1.1.4.4 app_cpu1.elf

Esta será la aplicación encargada de controlar el convertidor. Su funcionamiento se explica en el apartado correspondiente. Una vez tengamos el código de la misma deberemos crear una aplicación nueva en el SDK incluyendo que el procesador usado será el 1. Además, tendremos que hacer dos modificaciones en el BSP creado.

Ilustración 5-6: app_cpu1 modificación BSP 1

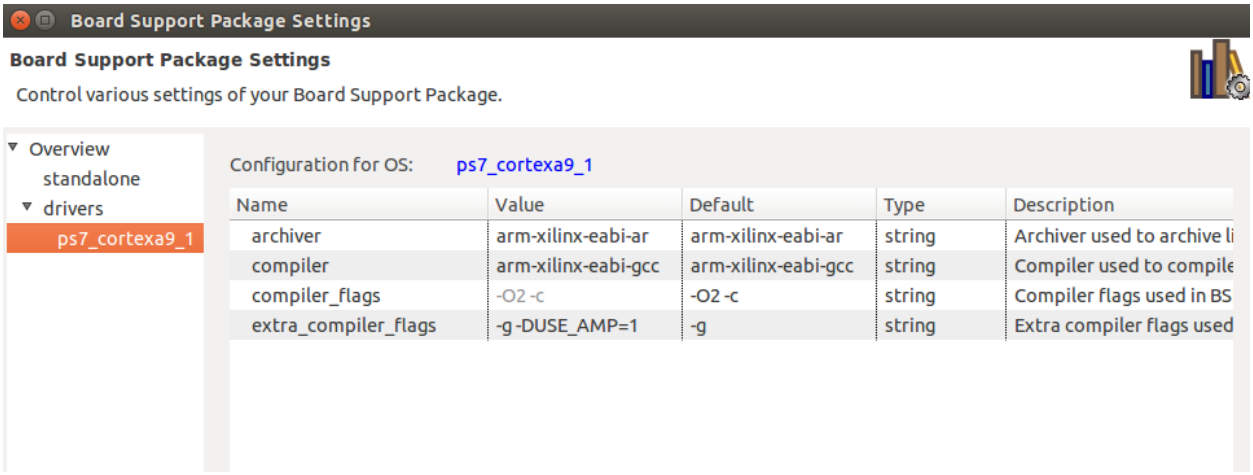


Ilustración 5-7: app_cpu1 modificación BSP 2

Por último, al crear la aplicación se creará un archivo ***lscript.ld*** que debemos asegurarnos de que tenga los valores correctos para la memoria.

ps7_ddr_0_S_AXI_BASEADDR 0x18000000 0x80000000

5.1.2 Funcionamiento

Para poder ejecutar el sistema simplemente debemos iniciar la ZedBoard con la SD. Con esto, mediante un terminal GTK podremos ejecutar el Linux que hemos compilado. Una vez iniciado, podremos usarlo como un terminal de Linux normal y tendremos accesibles multitudes de funciones, aunque no todas las que podemos encontrar en otro tipo de distribuciones. Si queremos ejecutar alguna aplicación solo debemos proceder como se hace normalmente en un sistema operativo Linux.

Nuestro sistema operativo sí tiene DHCP instalado por lo que para conectarnos a internet solo tendremos que conectarnos mediante un cable Ethernet a un router que permita este tipo de acceso. Si no, siempre podremos hacer una configuración manual de la misma.

5.2 Aplicación ADS8365 (app_cpu1)

El código complete del programa se puede encontrar en el Anexo **APP_CPU1**. El SDK de Xilinx genera unos archivos adicionales de manera automática. Aquí se presenta el contenido de ***app_cpu1.c, app_cpu1.h*** y ***mask.h*** que son los que hemos programado manualmente.

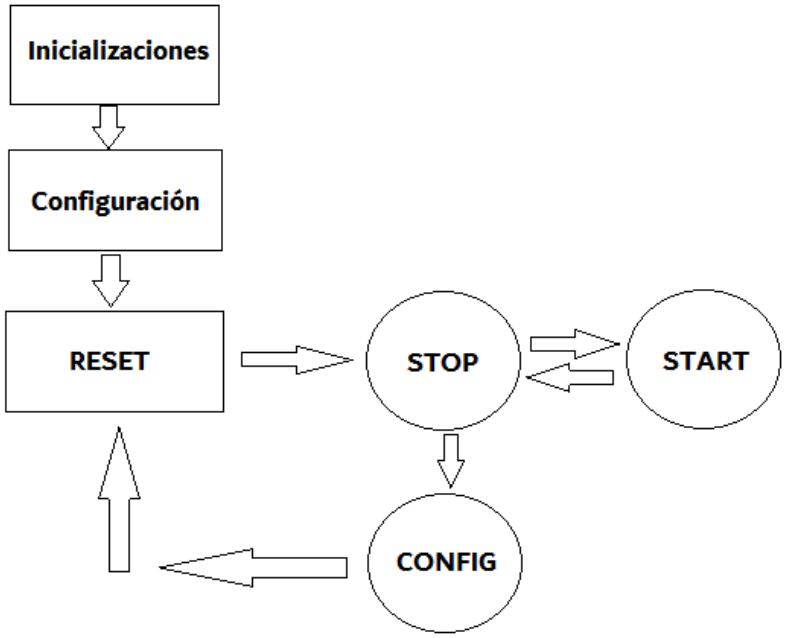


Ilustración 5-8: Flujo de programa APP_CPU1

El flujo del programa comienza inicializando los PMODs, tanto de datos como de configuración, las interrupciones, tanto la externa correspondiente al pin ***EOC*** y la del temporizador y el propio temporizador. También declara los PMODs que sean entradas a la placa del ADS8365 como ***INPUT***. Una vez finalizada la etapa de inicialización y configuración hace un reset del sistema. A partir de aquí el programa lo gobierna una máquina de tres estados.

- ***STOP***: Detiene la conversión. Este estado permite ir tanto al estado START como al estado CONFIG.
- ***START***: Inicia la conversión. Este estado solo permite volver al estado de STOP.
- ***CONFIG***: Permite la configuración de la frecuencia de muestreo del sistema. Este estado siempre vuelve al estado STOP pero antes hace un reset del sistema.

La conversión se realiza atendiendo a las interrupciones del timer que se configura con el parámetro de la frecuencia de muestreo que le pasemos. Cada vez que existe una interrupción, se hace una conversión por parte del ADC.

Para este procedimiento se codifica qué canales se quieren leer, que recordemos que es elegido de manera externa, mediante un número y se hace una triple comparación, una por cada par de canales A, B y C, para saber si debemos leerlo o no.

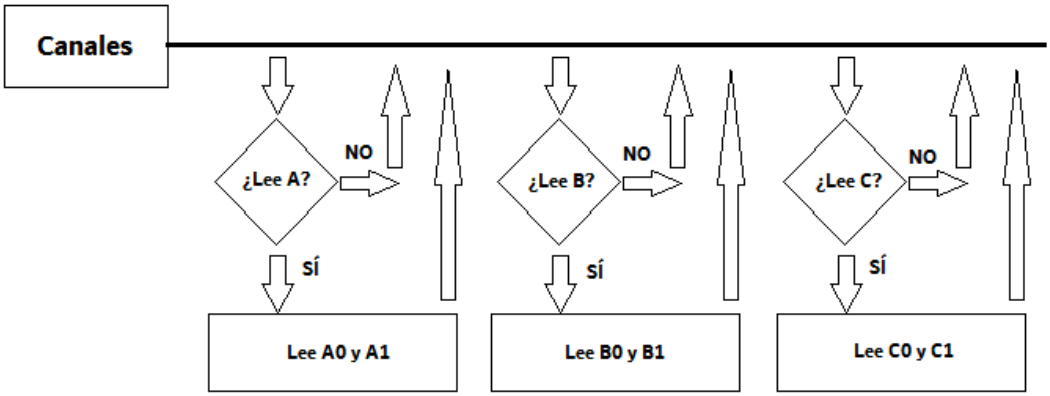


Ilustración 5-9: App_cpu1 Canales

En el archivo ***mask.h*** podemos encontrar todas las máscaras que hemos utilizado para hacer referencia a los pines y las diferentes zonas de memoria a las que accederemos ya sea para leer o almacenar los diferentes datos que necesitemos.

5.3 Aplicación Servidor (s_control)

Esta aplicación será la que ejecutemos en el sistema operativo Linux. Por esto, cuando la vayamos a programar en el SDK a la hora de crear nueva aplicación tenemos que indicar que será utilizada bajo un sistema operativo Linux. Además, tenemos que indicar que será ejecutado por la CPU 0.

Ilustración 5-10: Flujo de programa s_control

La aplicación comienza con una fase de configuración en la que escribe los registros de configuración del

convertidor, la frecuencia de muestreo a utilizar y los canales a convertir. Posteriormente, envía la información a la aplicación ***app_cpu1*** a través de la memoria compartida para que esta pase al estado ***CONFIG***. A continuación, realiza un proceso similar para que el convertidor pase al estado ***START***.

Llegados a este punto el servidor crea una conexión TCP y espera a que el cliente, a través de la interfaz externa, se conecte al sistema. Una vez realizada la conexión empieza a capturar los datos. El sistema ya se queda en ese estado capturando continuamente. Desde la interfaz externa se puede elegir en cada momento los canales que se quieren ver y se puede cambiar la frecuencia de muestreo. Para esto, la aplicación escribe los valores correspondientes en la memoria compartida para que puedan ser procesados.

Las funciones principales del sistema son:

- ***wait_sock***: Es la encargada de esperar a que el cliente, en nuestro caso la interfaz, se conecte.
- ***rd_mem***: Se usa para poder leer datos de la memoria compartida.
- ***wr_mem***: Se usa para poder escribir datos en la memoria compartida.
- ***rd_data***: Lee la tabla de datos y la divide.
- ***captura***: Empaqueta los datos y los envía.

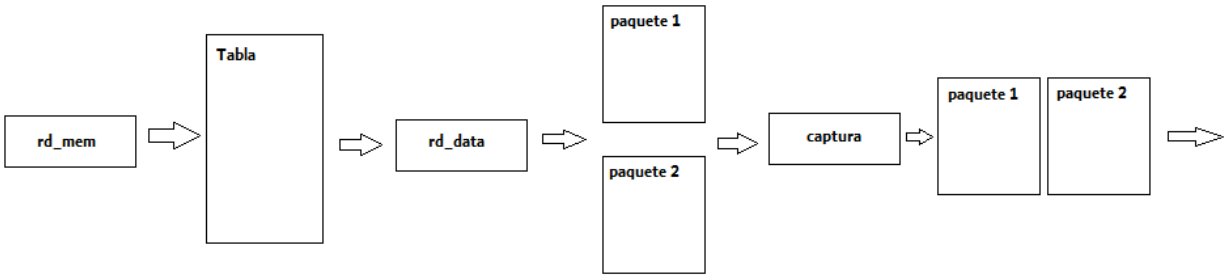


Ilustración 5-11: s_control Gestión Datos

Mediante la función ***rd_mem*** leeremos los datos correspondientes al canal que queramos que se encuentran en la memoria compartida de la ZedBoard. Como esos datos son demasiado grandes como para enviarlos en un solo paquete debemos dividirlos. Esto lo hacemos con la función ***rd_data*** que hace dos paquetes del mismo tamaño. Por último, la función ***captura*** coge esos paquetes que ya si tienen un tamaño adecuado y los envía.

5.4 Interfaz Externa

Esta interfaz ya se encontraba en proceso de desarrollo y nosotros lo que hemos hecho es adaptarnos a su funcionamiento y realizar pequeñas modificaciones para ajustarla a nuestras necesidades.

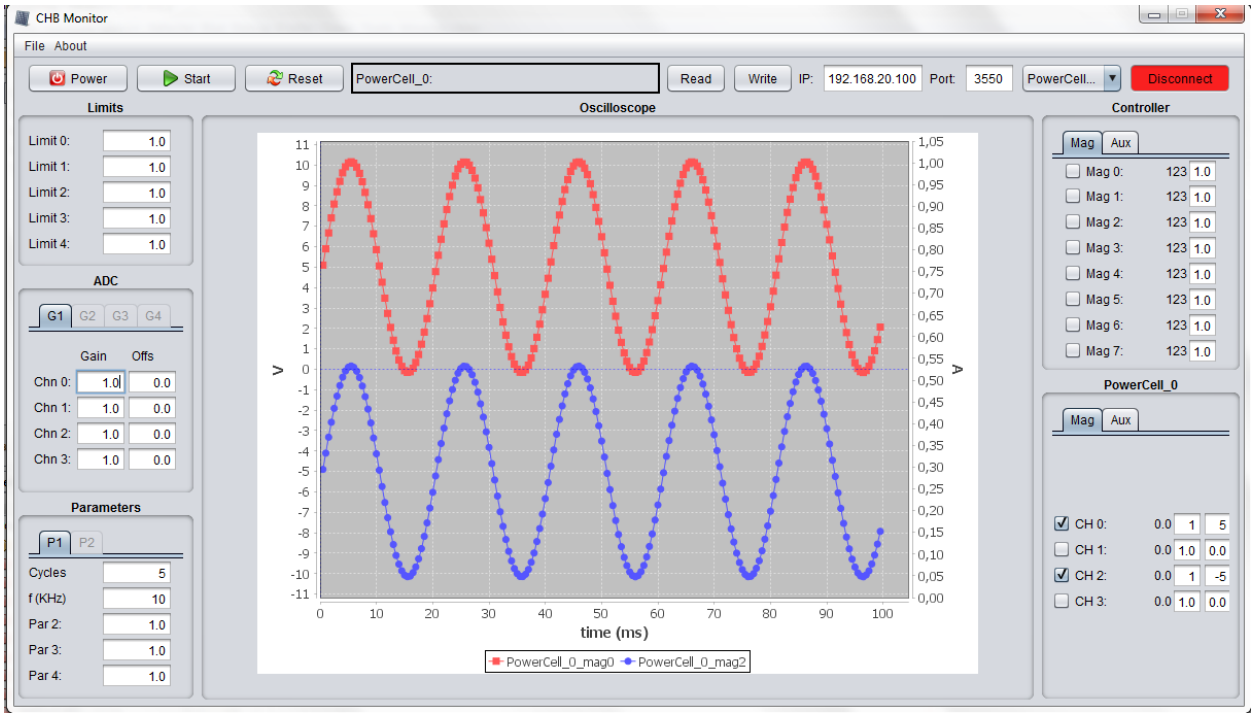


Ilustración 5-12: Interfaz externa

Como se trata de una interfaz en desarrollo no todos los elementos tienen aún funcionalidad, pero sí algunos de los que nos interesan. En la parte superior podemos observar una barra indicadora del sistema al que nos vamos a conectar, así como varios botones. El botón ***Write*** se usará para indicar que queremos escribir en la memoria compartida alguno de los valores. Los campos ***IP*** y ***Port*** serán usados para introducir la información de red correspondiente a la ZedBoard para poder establecer la conexión entre ella y el servidor.

En la parte central tenemos la representación de los datos que vamos capturando del convertidor. A la izquierda tenemos dos campos donde podremos configurar tanto la frecuencia de muestreo como el número de ciclos que queremos pintar en la zona central.

Por último, en el lado derecho podemos seleccionar qué canales queremos ver, desde ninguno hasta cuatro simultáneamente, así como si queremos añadir un desfase en la misma para poder facilitar la identificación cuando observamos varias señales al mismo tiempo.

6 EJEMPLO DE FUNCIONAMIENTO

Con todo el sistema listo solo queda seguir los pasos precisos para su ejecución y comprobar que todo funciona de manera correcta. Siguiendo las instrucciones del apartado de conexionado solo basta añadir lo necesario para realizar las pruebas.

En este caso utilizaremos un generador de señal que será la que usaremos como señal de entrada en el convertidor. Por simplicidad, cuando queramos ver más de un canal usaremos la misma señal en todos.

Ilustración 6-1: Generador de señal

Necesitaremos también una fuente de alimentación que nos proporcione +15V y -15V para obtener el rango de entrada deseado.

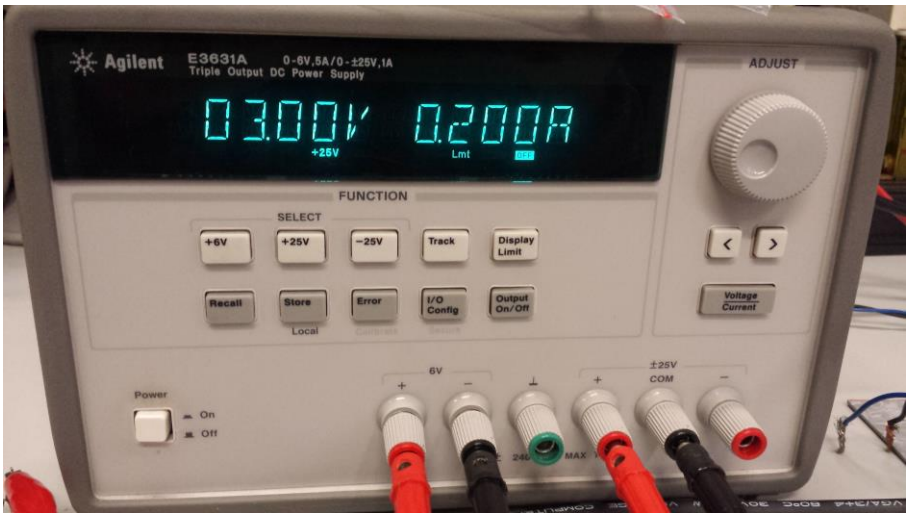


Ilustración 6-2: Fuente de alimentación

6.1 Inicio del servidor

Los últimos detalles antes de encender la ZedBoard es asegurarnos que los jumpers están en la configuración que necesitamos. En primer lugar, el JP18 debe estar configurado para tener una tensión de 2V5.

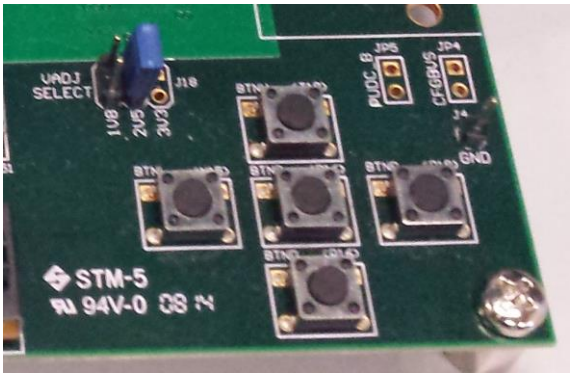


Ilustración 6-3: JP18 2V5

En segundo lugar, debemos configurar los JP7-JP11 de manera que la ZedBoard se inicie desde la tarjeta SD y, por tanto, el Linux y las aplicaciones que hemos cargado.

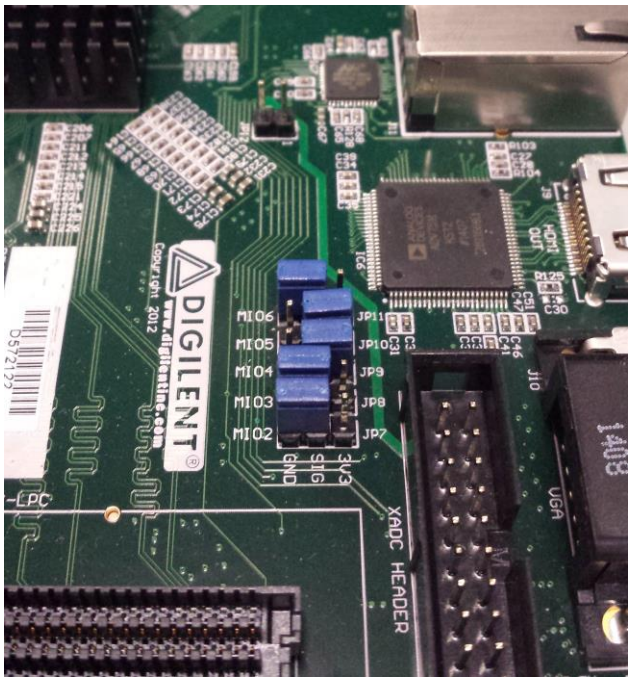


Ilustración 6-4: JP7-JP11

Por último, tenemos que asegurarnos de haber introducido la tarjeta SD en la ranura correspondiente y con todos los archivos necesarios.

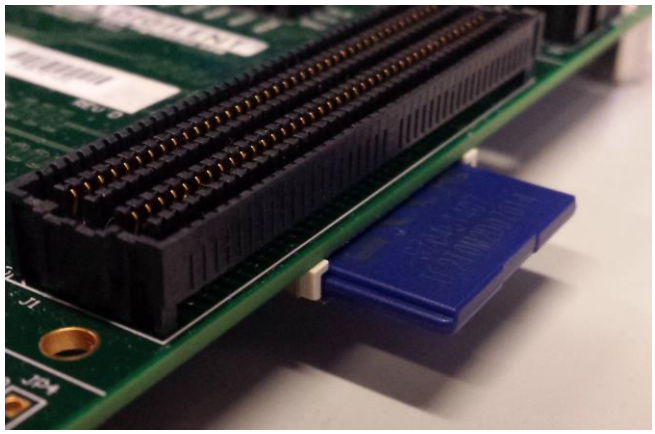


Ilustración 6-5: Tarjeta SD

Conectado y comprobado todo, enchufaremos a la corriente el transformador de la ZedBoard y pondremos en ON el switch ON/OFF de la ZedBoard. Necesitamos ejecutar un terminal para recibir los datos del puerto UART de la ZedBoard y para eso abriremos la aplicación GtkTerm en nuestro Linux en el ordenador.

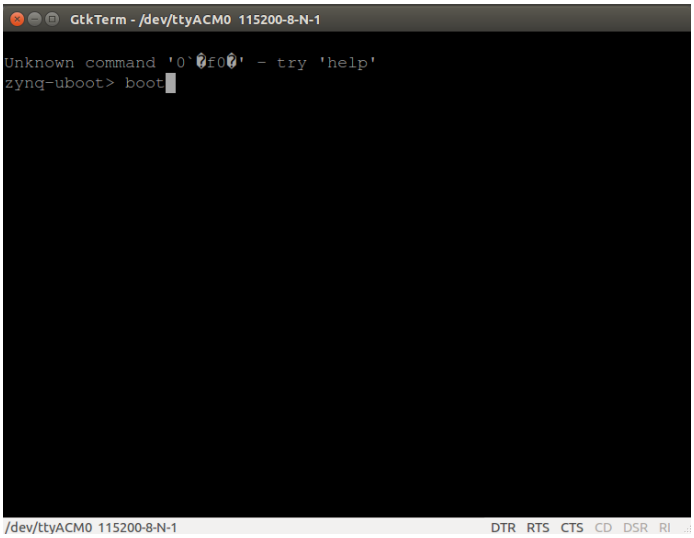


Ilustración 6-6: GtkTerm boot

Nos aparecerá una pantalla en negro en la que deberemos teclear el comando boot y pulsar enter. El sistema empezará a arrancar y a configurarse. Una vez este proceso haya terminado ya podremos utilizarlo y manejarnos a través de la consola como si fuese un terminal de Linux normal. Veremos como aparece la palabra zynq en el prompt para indicarnos que estamos conectados con la ZedBoard.

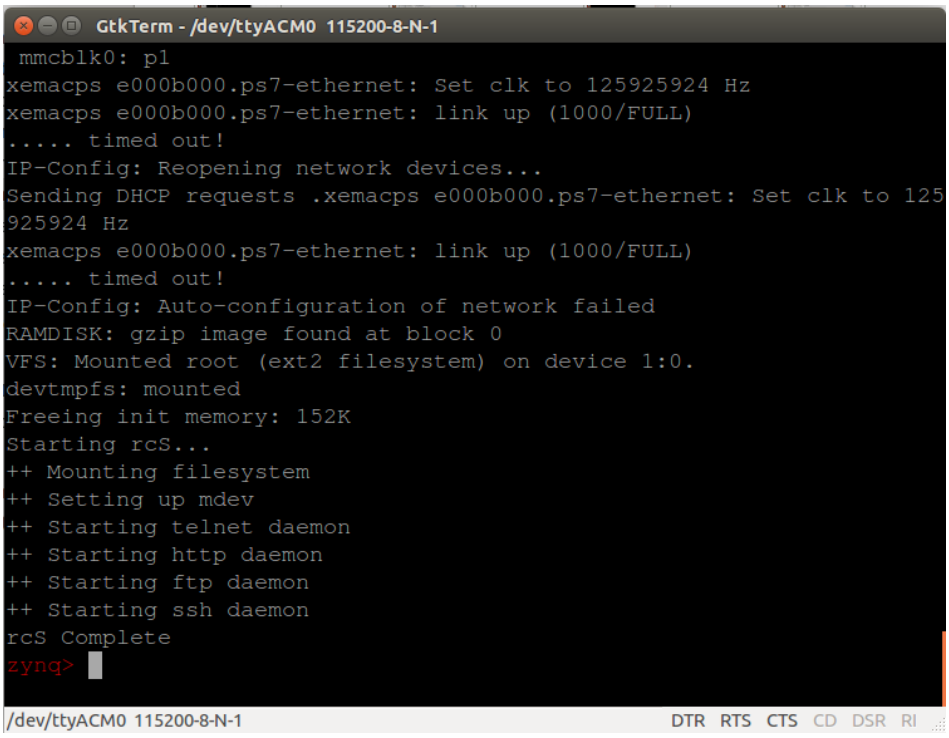


Ilustración 6-7: Sistema iniciado

Una primera comprobación que es recomendable hacer para estar seguros de que todo va correctamente es asegurarnos de que tenemos conexión a internet. Para ello lo más fácil es realizar un ping a una dirección conocida, de nuestra misma red, o de fuera. En nuestro caso la hemos realizado a GOOGLE. Es posible que si no está bien configurado el DNS al poner directamente la dirección Google.com la ZedBoard no sea capaz de resolverla. Para solucionar esto, nos dirigiremos a la carpeta /etc de nuestro sistema y crearemos un archivo con el nombre resolv.conf con la información de los servidores DNS. Si no, siempre se puede hacer ping a la dirección IP directamente y no debe haber ningún problema.

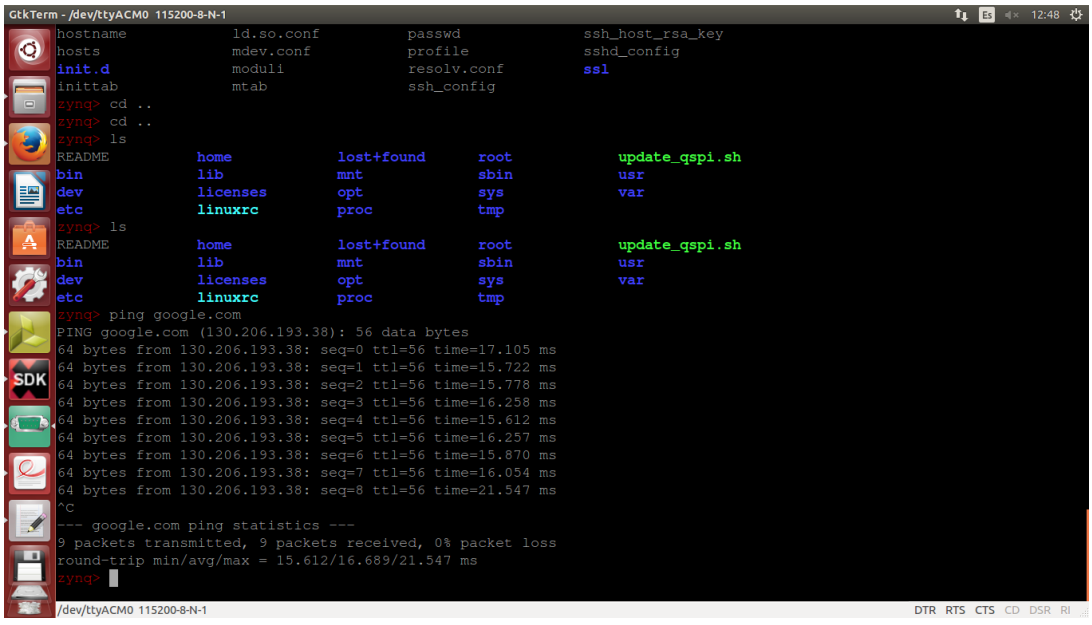


Ilustración 6-8: Ping GOOGLE

Realizada esta comprobación solo queda iniciar el servidor. Para ello deberemos dirigirnos a la carpeta donde

se encuentran los archivos de la tarjeta SD y ejecutar la aplicación:

I. *cd mnt*

II. *./s_control.elf*

Con esto el servidor estará iniciado y mostrará un mensaje indicando que espera una conexión externa.

6.2 Inicio de la interfaz

Para poder ejecutar la interfaz JAVA se debe tener instalado el programa NetBeans IDE 8.0.2 [11]. Una vez instalado se debe ejecutar y abrir el proyecto de la interfaz. Al ejecutar por primera vez el programa pedirá algunas librerías extra que se deben instalar. Estas librerías se encuentran con el resto de archivos del proyecto. Para instalarlas se debe extraer previamente cada una e ir añadiéndolas al programa. El proceso completo es el siguiente:

- Al abrir el proyecto aparecerá una advertencia de que existen algunos problemas y se debe pulsar el botón ***Resolve Problems…***
- Aparecerá una lista con las librerías que faltan. Se debe hacer el proceso para cada una de ellas. Se selecciona una y se pulsa ***Resolve…***
- En la siguiente ventana se debe pulsar ***New Library…*** en Library Name se debe poner el nombre que nos aparecía en la lista anterior correspondiente a la librería que se quiera añadir y se pulsa ***OK***.
- Por último, se debe pulsar el botón ***Add JAR/Folder…*** y buscar el archivo que se ha descomprimido anteriormente y pulsa aceptar. Con esto debe aparecer como solucionado el problema.
- Se debe repetir el proceso con todas librerías que se piden.

Ahora sí ya se puede ejecutar el programa pulsando ***Run Project*** o pulsando la tecla ***F6***. Una vez iniciada la interfaz lo primero que tenemos que hacer es realizar la conexión con el servidor. Para ello debemos rellenar el campo IP y el puerto con los datos de nuestra ZedBoard. La IP dependerá de la que nos asignen en nuestra red pero el puerto siempre será el 3550. Con esta información solo queda seleccionar ***PowerCell_0*** y darle al botón ***Connect***.

Ilustración 6-9: Barra de Conexión

Si todo ha ido correctamente, nuestra interfaz estará conectada con el servidor y el convertidor empezará a funcionar y mandar información. Podemos empezar a usarla.

En primer lugar, podemos marcar qué canales queremos ver y si deseamos algún tipo de coeficiente de magnitud y fase para cada uno de los canales.

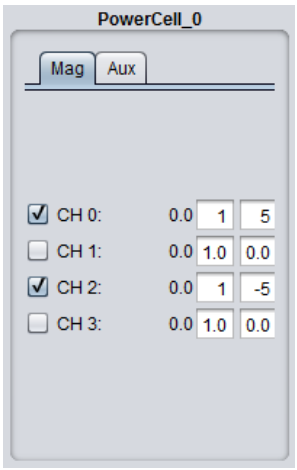


Ilustración 6-10: Coeficientes de Magnitud y Fase

Por otro lado, también tenemos accesibles dos campos donde podemos configurar tanto la frecuencia de muestreo como el número de ciclos que queremos mostrar en la pantalla. Recordar que para que en este caso los cambios tengan efecto debemos pulsar el botón ***Write***.

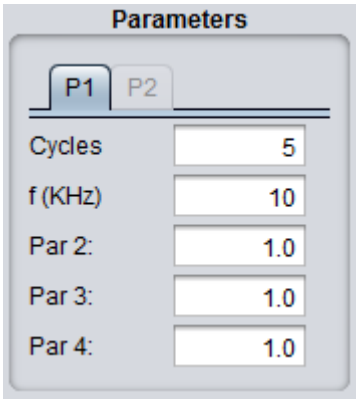


Ilustración 6-11: Cycles y Frecuencia

Veremos ahora lo que muestra nuestra interfaz ante varias señales de entrada diferentes.

6.3 Ejemplo 1: Onda Cuadrada

Señal de entrada: Onda Cuadrada.

Canales a mostrar: 1.

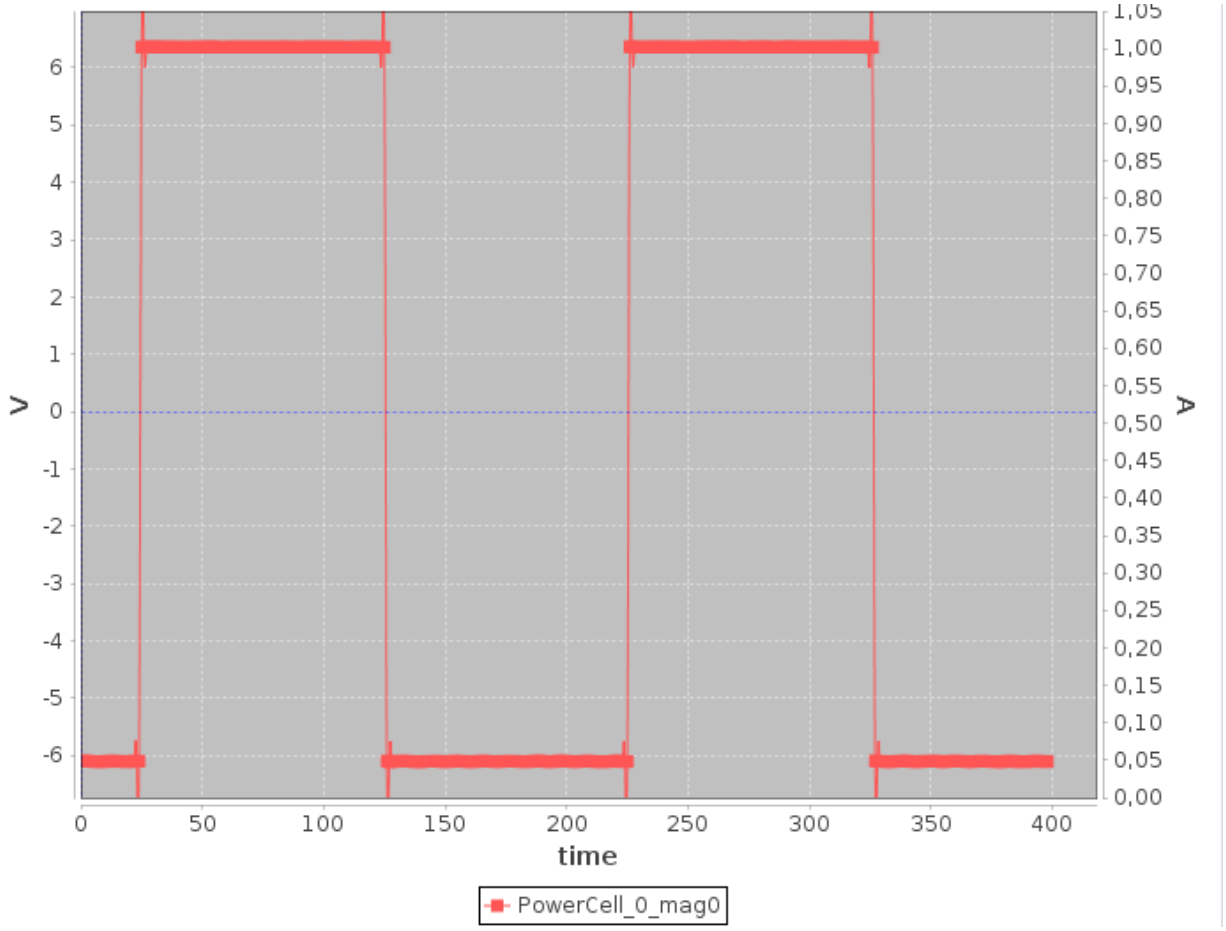


Ilustración 6-12: Onda cuadrada

6.4 Ejemplo 2: Seno – 2 canales

Señal de entrada: Seno.

Canales a mostrar: 2.

Condición: OFFSET.

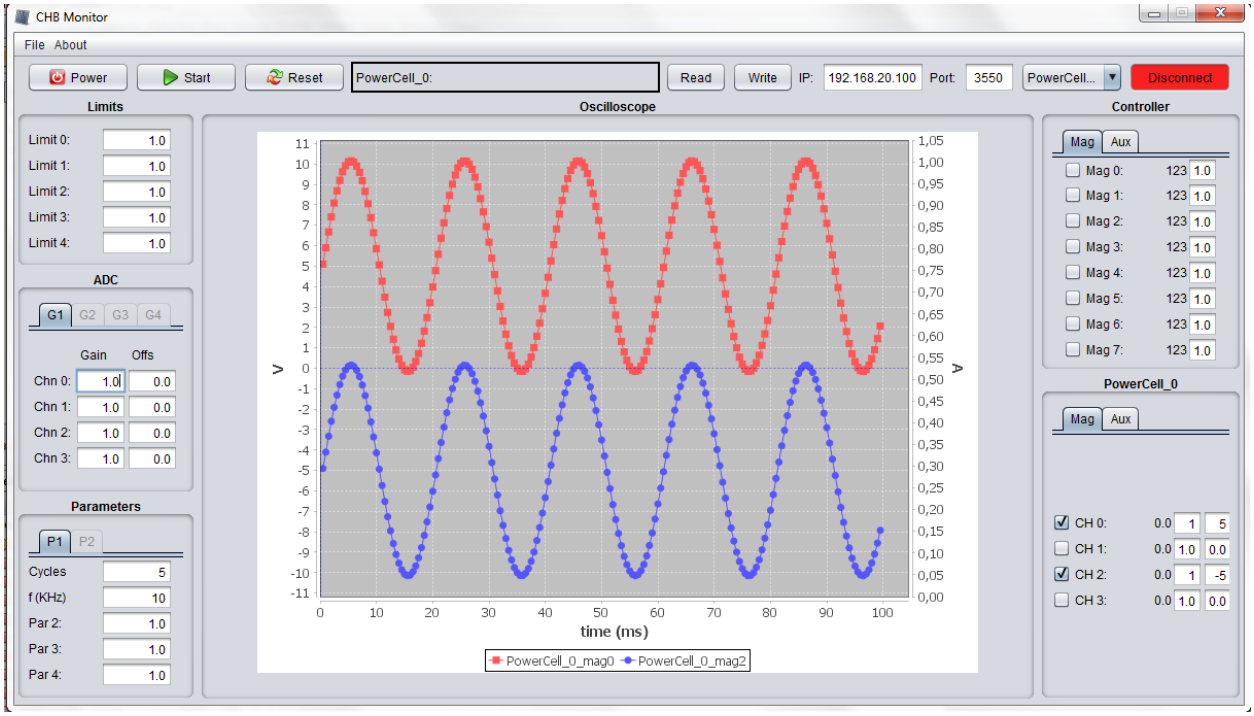


Ilustración 6-13: Seno – 2 Canales con offset

6.5 Ejemplo 3: Seno – 4 canales

Señal de entrada: Seno.

Canales a mostrar: cuatro.

Condición: OFFSET.

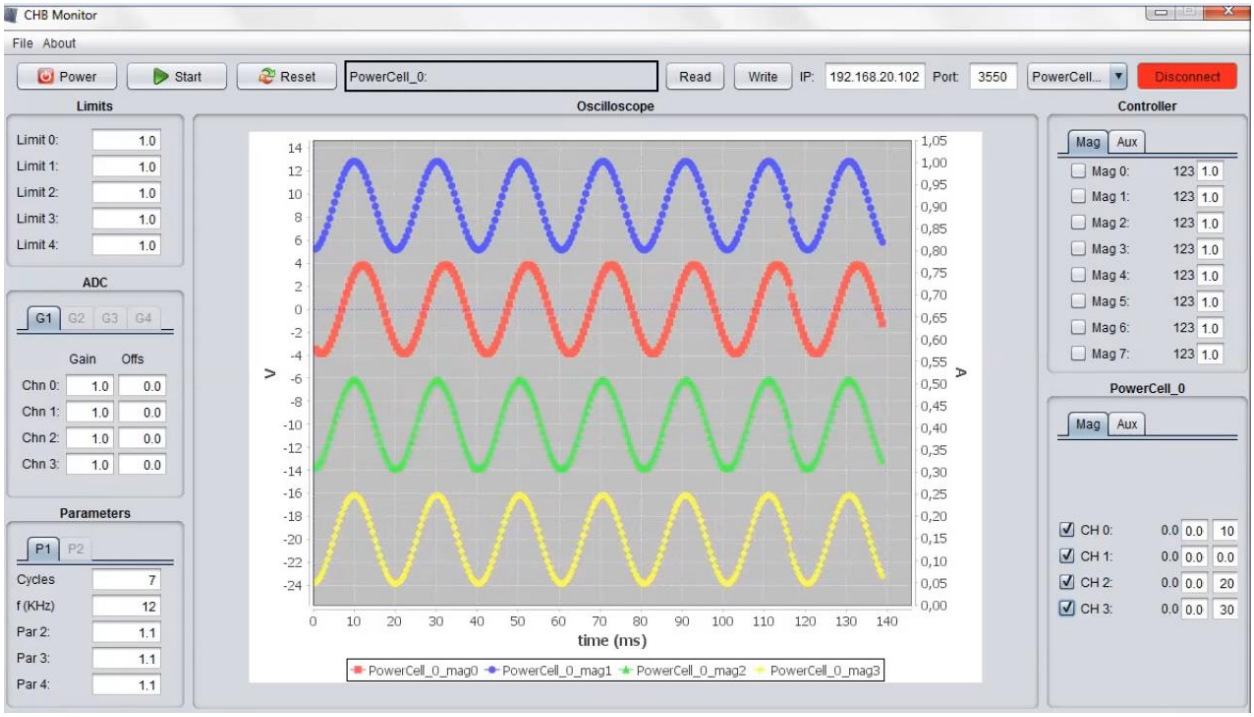
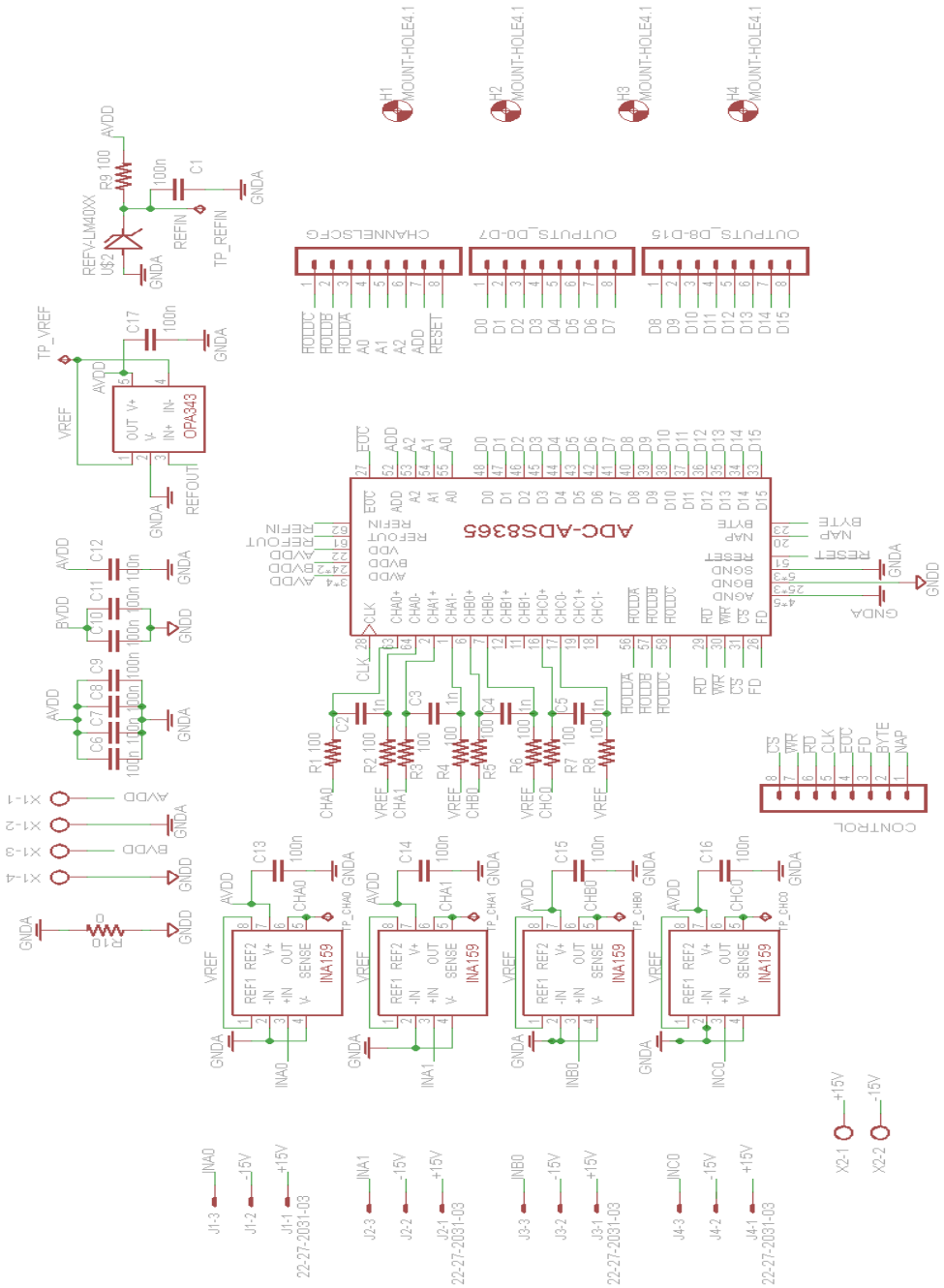
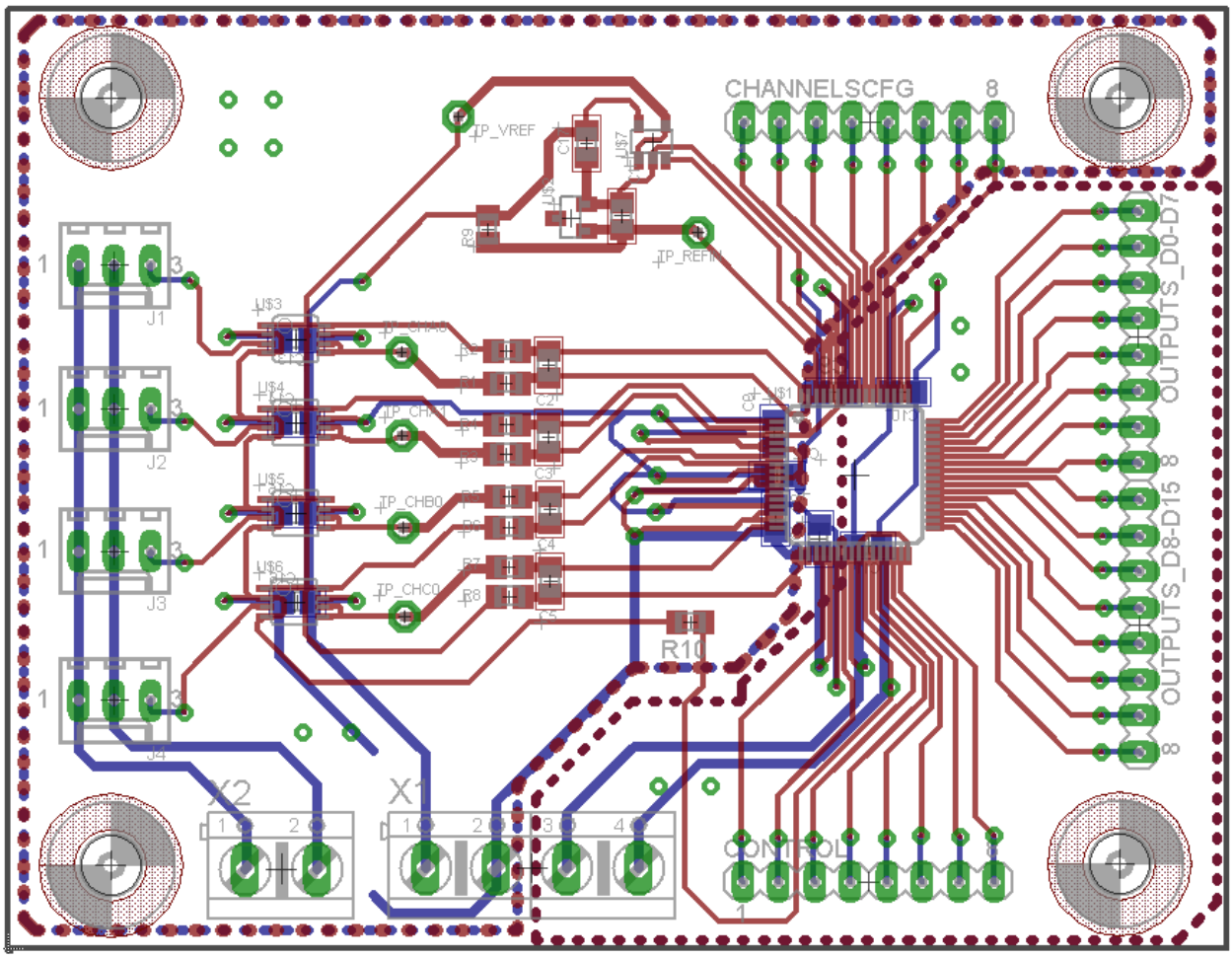


Ilustración 6-14: Seno – 4 Canales con offset.

Esquemático placa desarrollo ADS8365



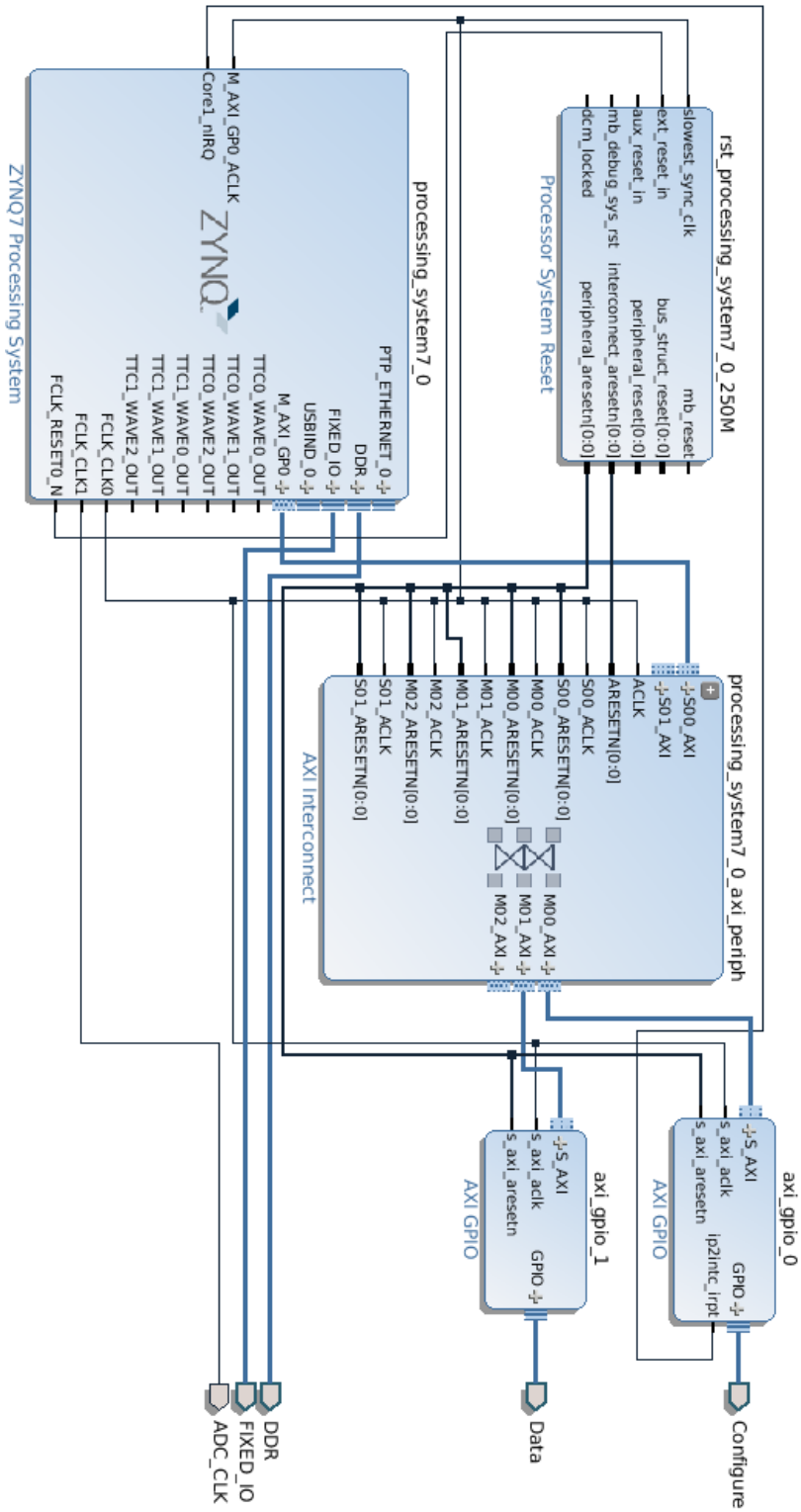
Rutado placa desarrollo ADS8365



Máscaras de configuración

	NAP	BYTE	FD	EOC	CLK	RD	WR	CS	HOLDC	HOLDB	HOLDA	A0	A1	A2	ADD	RESET
Reset	0	0	-	-	-	1	1	0	1	1	1	0	0	0	0	0
Reposo	0	0	-	-	-	1	1	0	1	1	1	0	0	0	0	1
Convertir canal A	0	0	-	-	-	1	1	0	1	1	0	0	0	0	0	1
Leer canal A0	0	0	-	-	-	0	1	0	1	1	0	0	0	0	0	1
Leer canal A1	0	0	-	-	-	0	1	0	1	1	0	1	0	0	0	1
Convertir canal B	0	0	-	-	-	1	1	0	1	0	1	0	1	0	0	1
Leer canal B0	0	0	-	-	-	0	1	0	1	0	1	0	1	0	0	1
Leer canal B1	0	0	-	-	-	0	1	0	1	0	1	1	1	0	0	1
Convertir canal C	0	0	-	-	-	1	1	0	0	1	1	0	0	1	0	1
Leer canal C0	0	0	-	-	-	0	1	0	0	1	1	0	0	1	0	1
Leer canal C1	0	0	-	-	-	0	1	0	0	1	1	1	0	1	0	1
FIFO	0	0	-	-	-	0	1	0	0	0	0	1	1	1	0	1

VIVADO Diagrama de bloques



REFERENCIAS

[1] Datasheet ADS8365

http://www.ti.com/lit/ds/symlink/ads8365.pdf

[2] Datasheet INA159

http://www.ti.com/lit/ds/symlink/ina159.pdf

[3] Datasheet OPA343

http://www.ti.com/lit/ds/symlink/opa343.pdf

[4] Datasheet LM4040

http://www.ti.com/lit/ds/symlink/lm4040-n.pdf

[5] EAGLE

https://www.autodesk.com/products/eagle/free-download

[6] Documentación Zedboard

http://zedboard.org/support/documentation/1521

[7] VIVADO

https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive.html

[8] Petalinux

http://www.wiki.xilinx.com/PetaLinux

[9] Ubuntu

https://www.ubuntu.com/download/desktop

[10] Archivos fuente Linux

http://www.wiki.xilinx.com/Linux#Xilinx%20Linux-Compiling%20Linux%20from%20Source-Build%20The%20Kernel

http://www.wiki.xilinx.com/Fetch%20Sources

[11] NetBeans

https://netbeans.org/downloads/

GLOSARIO

ADC: Analog to Digital Converter

CPU: Central Processing Unit

DHCP: Dynamic Host Configuration Protocol

FPGA: Field Programmable Gate Array

LUT: LookUp Tables

PMOD: Peripheral MODules

RTL: Register Transfer Level

SD: Secure Digital

SDK: Software Development Kit

SMD: Sourface-Mount Device

SoC: System on Chip

SPS: Samples per second

TCP: Transmission Control Protocol

UART: Universal Asynchronous Receiver-Transmitter

USB: Universal Serial Bus