INTERFACE PCI PARA PC COMPATIBLE

Ing. Luis O. Ventre - Ing. Leonardo Gamarra
Laboratory of Computer Architecture - Universidad Nacional De Córdoba
Córdoba, C.P. 5000 – Argentina
lventre@gmail.com - leogamarra@gmail.com

Supported By:
Ing. Orlando Micolini – Ing. Adriana Damiani
Laboratory of Computer Architecture, Universidad Nacional de Córdoba
Córdoba- CP 5000 –Argentina
omicolini@compuar.com - adriana.damiani@gmail.com

ABSTRACT:

We call BUS to any group of lines whose main purpose its interconnect different devices. One of the most critical problem in digital aplicattions its to establish a fast access communication between peripheral devices. With the intention of diminishing the limitations in the data transfer rate, in year 1992, Intel develops the Standard bus of interconnection of peripheral components (PCI).

The PCI Bus, is the most used standard for the development of aplication where work togheter external components to the CPU; this paper discuss the development of my end career project, which raises the implementation of a CORE, intrument that allows to fulfill the highest requirements of the communication protocol PCI, this instrument is implemented on a reprogrammable logic FPGA Board.

Also, explain the implement of drivers for testing the development under an operating system, and several visual aplications were created to validate the communication and bidirectional data transfer.

At the end of this paper, we measured and studied the results to specify the reasons that allows us to affirm that the development is successful.

Keywords: Computer Architecture, PCI Bus, Core, FPGA.

1-INTRODUCCIÓN:

Los buses son los encargados de realizar el intercambio de información entre dispositivos periféricos y la CPU en la arquitectura de computadoras, su rol es fundamental en todo tipo de aplicaciones. Hasta hace un tiempo los principales buses como ISA, EISA o Micro Channel presentaban, ante aplicaciones gráficas, o de requerimientos elevados, problemas bien determinados como:

- Baja frecuencia de operación.
- Longitud de palabra limitada.
- Importante latencia de acceso a los dispositivos.
- Poca escalabilidad.

Con el avance de la tecnología de los periféricos multimedia y de almacenamiento masivo de información, creció significativamente la necesidad de intercambiar grandes cantidades de datos, con la consiguiente necesidad de aumentar la velocidad de transferencia de los mismos. Estos problemas provocaron una inevitable evolución en los buses con ideales óptimos de:

- Flexibilidad – Amplio espectro de conexión a distintos dispositivos.
- Alcanzar un nivel de estándar a nivel de mercado mundial.
- Expansibilidad – para la admisión de numerosos dispositivos.
- Performance - alta velocidad de interconexión .

Con esta perspectiva surge en el año 1992, desarrollado por Intel la norma que define al Bus PCI (Peripheral Component Interconnect), PCI (PCI-SIG) [1] define, "*el objetivo es desarrollar un estándar industrial de la arquitectura de bus local de altas prestaciones que facilite el desarrollo de nuevos periféricos"*

Bajo esta arquitectura de Bus es factible realizar incontadas aplicaciones con diversas funcionalidades, pero todas y cada una de ellas requiere el desarrollo de un CORE PCI, el cual es el núcleo que implementa el protocolo de intercomunicación tal cual define la norma. Debido a los requisitos de la misma, la implementación de dicho CORE necesita de manera indispensable de lógica avanzada, como por Ej.: FPGA o chips PLX[2].
En el presente trabajo se desarrolla un CORE PCI, en una FPGA y se implementan aplicaciones de transferencia de datos.

2-REQUERIMIENTOS

- Desarrollar el protocolo de comunicación para lograr la interconexión bidireccional entre un dispositivo esclavo y un maestro a través del bus PCI.
- Implementarlo en un dispositivo esclavo.
- Desarrollar un driver para poder tener acceso al dispositivo y a los registros de configuración del CORE PCI, con el fin de realizar el testing.
- Realizar una o mas aplicaciones para demostrar la bidireccionalidad de la comunicación entre los dispositivos.

[1] PCI (PCI-Special Interest Group) www.pcisig.com - [PciSpec, 98]
[2] PLX Technology www.plxtech.com

3-OBJETIVOS

Realizar el desarrollo de un CORE que implemente la interfaz PCI en una FPGA permitiendo minimizar los problemas descriptos en la introducción. Para cumplir este objetivo es necesario:

- Seleccionar el Kit Fpga para la implementación.
- Elección del lenguaje de programación para el desarrollo.
- Desarrollar el CORE con las herramientas selectas.
- Realizar uno o mas Drivers para el testing bajo sistemas operativos.
- Realizar aplicaciones y hard necesarios para validar el funcionamiento del desarrollo.

4-DESARROLLO

4.1-Nuestro proyecto

El lenguaje selecto para el desarrollo fue VHDL (Very High Speed Integrated Circuit Hardware Description Language) por sus siguientes ventajas:

- Standard principal de mercado.
- Primordial lenguaje de desarrollo de aplicaciones PCI.
- Alta disponibilidad de recursos bibliográficos.

El desarrollo implementado puede observarse en la figura 1:

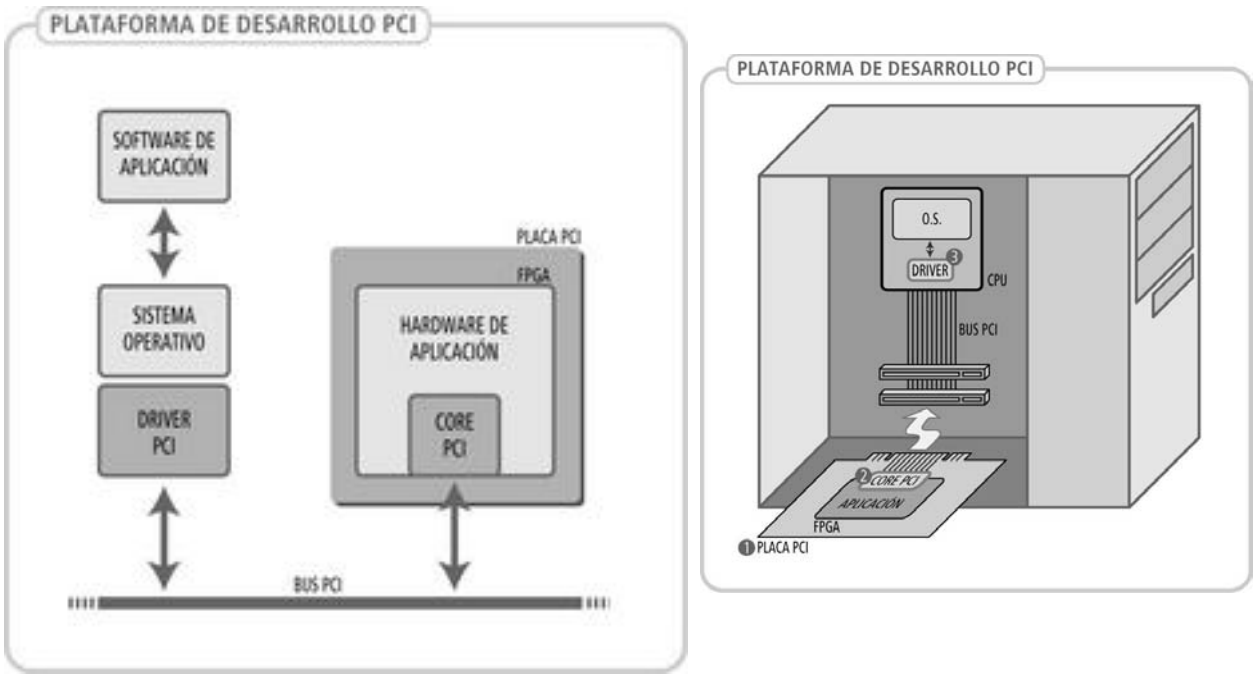


Figura 1: Esquema gráfico de los objetivos a llevar a cabo.

Una vez selecto el lenguaje se describen la estructura y metodología de trabajo de nuestro proyecto.

Los pasos que se siguieron para la implementación del trabajo fueron:

1- Aprendizaje del lenguaje VHDL.
2- Análisis de factibilidad del proyecto, investigando en la documentación de proyectos similares.
3- Estudio de la arquitectura del bus PCI y estudio de la norma.
4- Elección del kit apropiado en precio/prestación sobre el que basamos nuestro trabajo.
5- Diseño y codificación del CORE.
6- Simulaciones.
7- Debug y testing.
8- Mediciones y validación.

4.2-Selección de Herramienta:

Raggedstone1
Spartan-3 FPGA Development Board

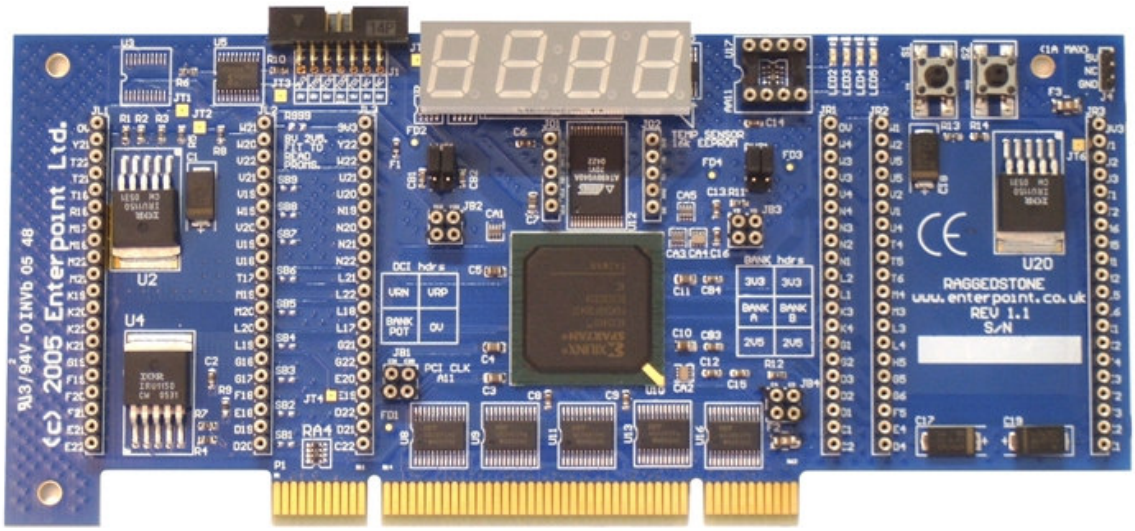


Figura 2: Raggedstone1 Spartan-3 FPGA Development Board.

Luego de un profundo análisis de mercado, se seleccionó el kit que muestra la figura 2, el mismo estaba integrado con una fpga XILINX XC3s400, de considerable tamaño, cabe mencionar que como se verá más adelante nuestro CORE completamente ocupa aproximadamente un 40 % de las capacidades del dispositivo, además funciona en ambos buses como puede observarse, se puede instalar en buses PCI de 3,3 v así como también en buses de 5v.

Dentro de sus principales características se destacan:

- FLASH memory, 4 Mbit.
- 1 x 16KBit serial EEprom fitted.
- 1 x LM75 Sensor de temperatura para múltiples aplicaciones incorporado.
- 4 x 7 SEGMENT digit LED – Con la posibilidad de removerlos para tener extras I/O.
- 4 LEDS de múltiple aplicación.
- Socket para utilización de modulo de Clock externo, o I/O adicionales.
- 2 Pushes switch para múltiple aplicación.
- Bancos de entrada salida configurables para 3,3v/2,5v
- Aproximadamente 120 I/O disponibles vía pin out + Display removable I/O.

4.3-Implementación:

La codificación y estructura del core están fundados en la norma PCI[3].

A continuación se muestra en la Figura 3 la FSM (Finite State Machine) propuesta por la norma:

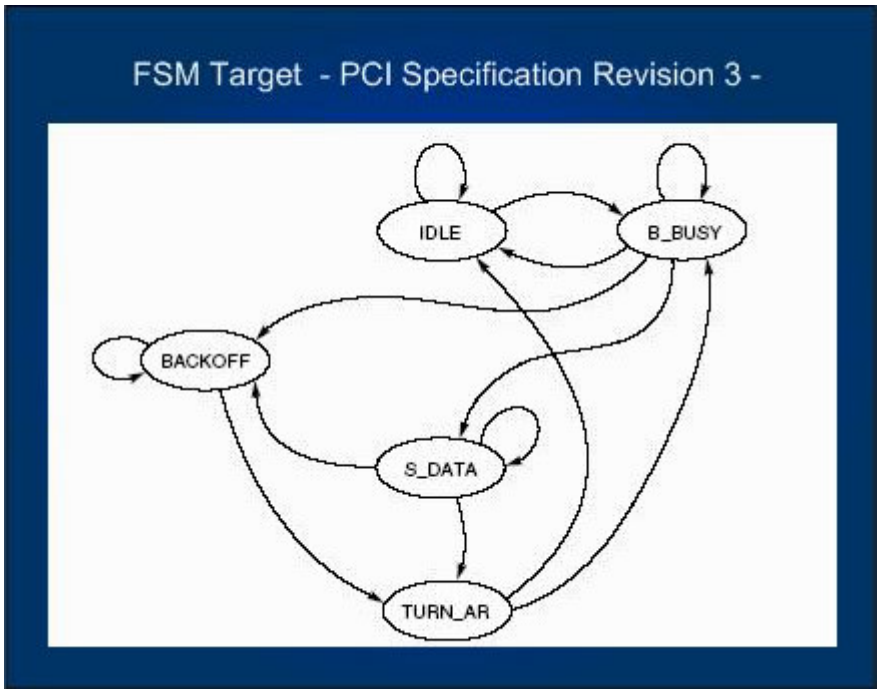


Figura 3: FSM sugerida por PCI Specification Revision 3.

Basándonos en esta estructura, nuestro CORE cuenta con una máquina de estados de desarrollo propio. Fue necesario sumar a la FSM un conjunto de Bloques Lógicos para cumplir las funcionalidades exigidas por la norma.

[3] PCI SPECIFICATION REVISIÓN 3 - PCI-SIG - www.pcisig.com/specifications

Las principales funcionalidades se observan en el siguiente diagrama (figura 4):

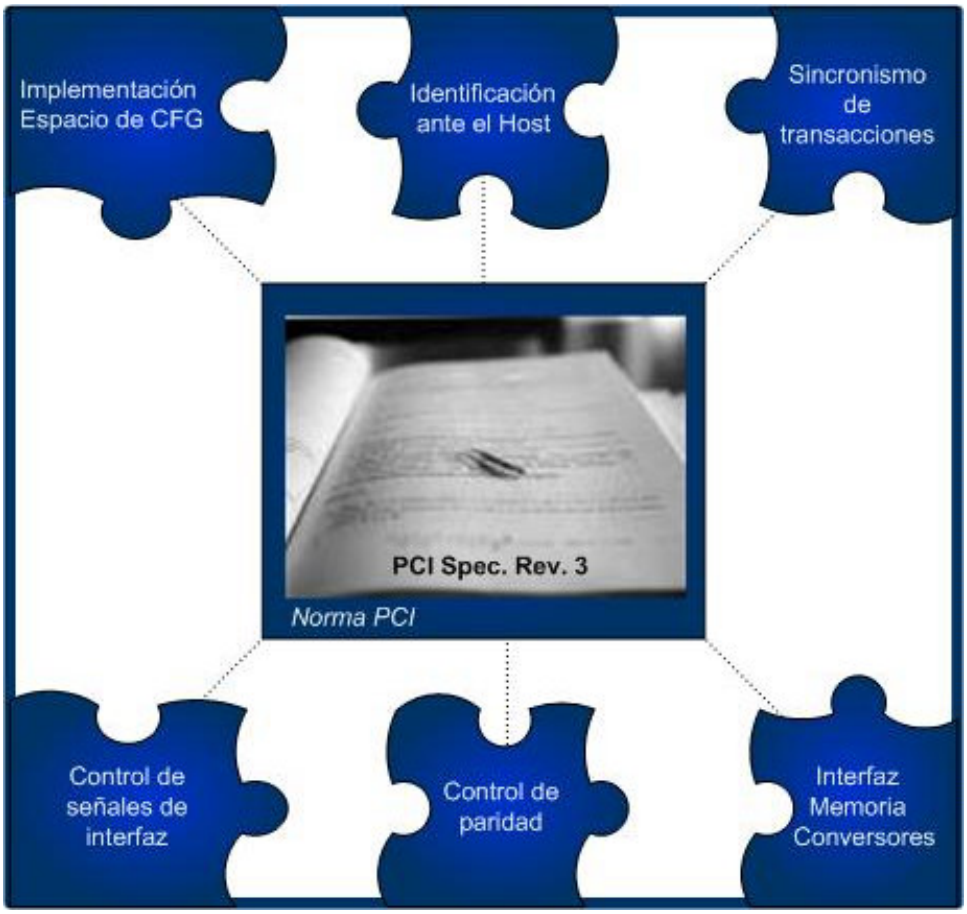


Figura 4: Diagrama funcional de Core PCI.

4.4-Arquitectura:

Para cumplir con nuestro diagrama funcional, implementamos nueve Bloques Lógicos con una estructura FLAT, en la cual en cada Bloque se desarrolla una tarea determinada en forma paralela y todos los Bloques son instanciados por un Bloque esquemático.
En la Figura 5 se puede apreciar la arquitectura implementada:

BLOQUE	**FUNCIÓN**
PCI	Esquemático – Instancia componentes
Parcontrol	Control de errores de paridad
Regdir	Registro de la dirección del dispositivo objeto de la transacción
C	Comparar direcciones en fase de direccionamiento
RdDatos	Manejo bidireccional de transferencia de datos
Fsm	Maquina de estados – Finite State Machine
Def	Definiciones
ConfSpace	Registro de espacio de configuración del dispositivo
Cout	Generar las combinaciones de señales de salida

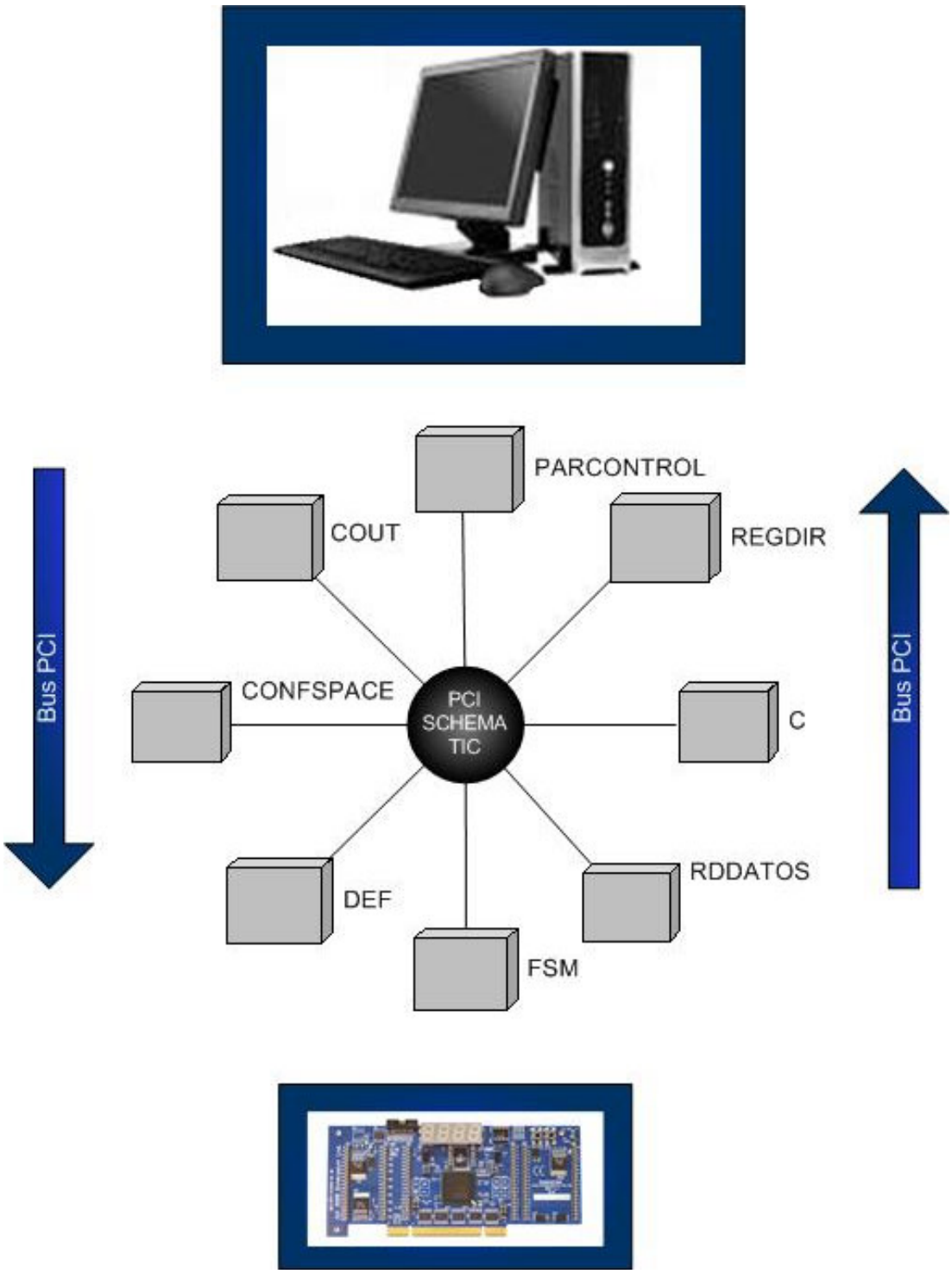


Figura 5: Estructura de bloques de Nuestro CORE.

4.5-Driver:

Para testear el funcionamiento bajo entorno windows se utilizo WinDriver[4] para desarrollar el Driver de nuestra tarjeta. Esta herramienta nos permitía leer los registros de configuración, y también nos da la posibilidad de leer y escribir los Base Address Registers (Espacio de configuración de nuestro dispositivo).

En la figura 6 se puede observar la información del espacio de configuración de nuestro dispositivo.

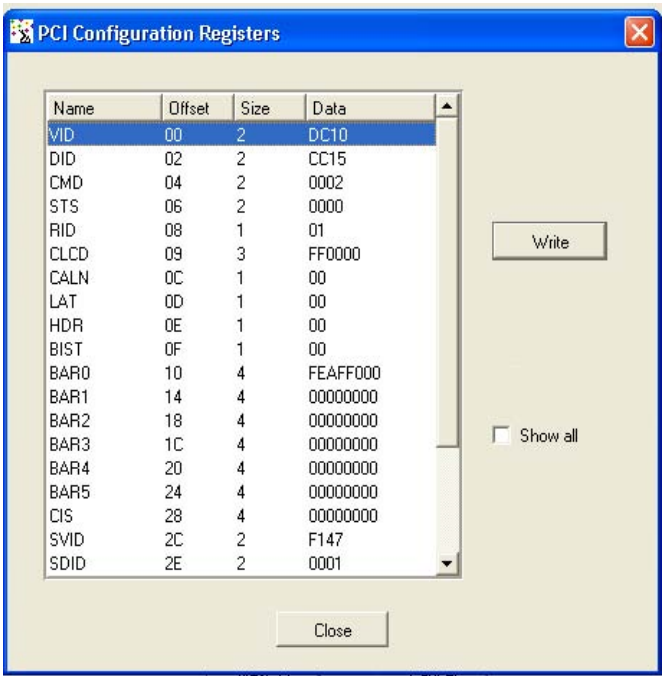


Figura 6: Lista de Registros de Configuración accedidos por WinDriver.

4.6-Aplicaciones:

Para validar el funcionamiento de nuestro CORE bajo el sistema operativo fue necesario implementar un nuevo driver y una serie de aplicaciones gráficas entre ellas Generador de Funciones – Osciloscopio Analizador de señal digitalizada. APIs desarrolladas (Figuras 7, 8 y 9).

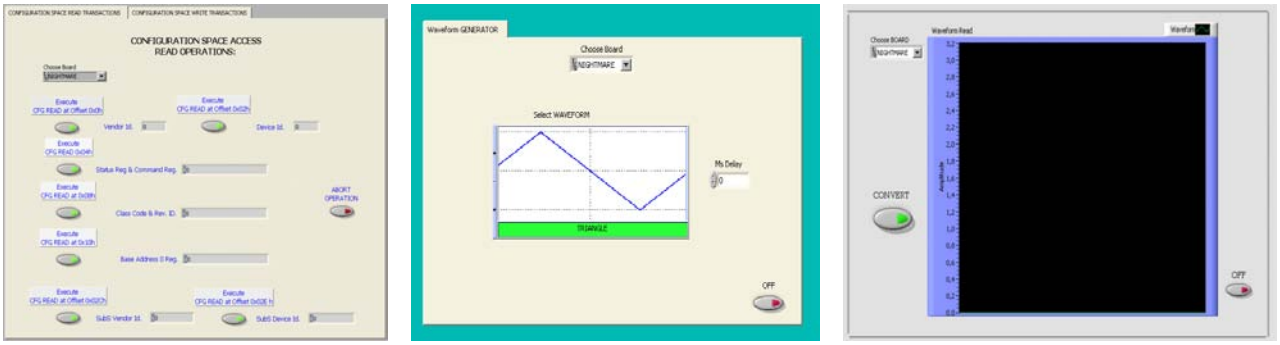


Figura 7, 8 y 9: Aplicaciones Gráficas.

[4] Driver Development Tools – Windriver - www.jungo.com

5-MEDICIONES

Para medir el tiempo de respuesta y el sincronismo de un dispositivo estándar PCI (Placa de video) se utilizo un ANALIZADOR LÓGICO, modelo LA4000/4280 80 channels, este equipo en particular constaba de 80 canales, adaptador a puerto USB y además un rate máximo de sampleo de 200Ms/s. Una vez observado el comportamiento se trabajó sobre nuestra tarjeta para igualar la respuesta temporal analizada.

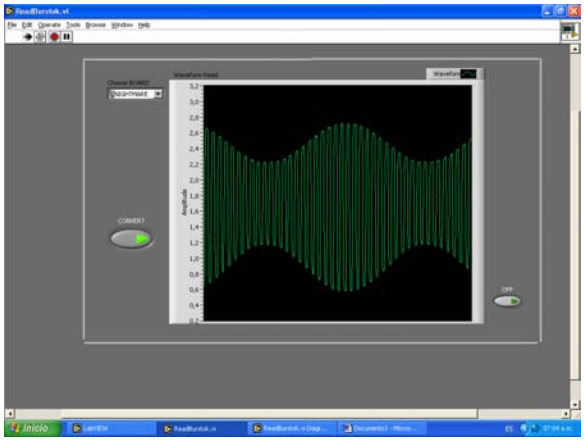
El resultado obtenido muestra que nuestro dispositivo se configura automáticamente en tiempo de arranque a una frecuencia de bus de 33 MHZ, cumpliendo con el grado de estándar, bajo el nivel mas rápido de respuesta temporal al maestro: Devsel Timing FAST[5].

Otra medición realizada en tiempo de compilación fue el espacio de pastilla ocupado por el CORE implementado, así como también el espacio total utilizado por el CORE más el código necesario para el desarrollo de las aplicaciones de almacenamiento de datos (buffer).

Los resultados obtenidos fueron:

El CORE sin aplicaciones empleaba aproximadamente el 20% de los recursos, mientras que el espacio total utilizado llegaba al 45% aproximadamente sumando a nuestro CORE las aplicaciones mencionadas en el párrafo anterior.

Para validar el funcionamiento del proyecto se implementaron dos placas conversoras una AD con el micro ADC Max1449 y otra DA con el micro DAC0800. Equipados conjuntamente con un par de generadores de funciones y un osciloscopio se corroboró el funcionamiento, digitalizando señales moduladas en amplitud de una frecuencia aproximada de 100 Khz, estando limitados no solo por las placas sino por el tiempo de muestreo del software y la mutiplexación del bus PCI, imágenes de este proceso se observan en las figuras 10 y 11.



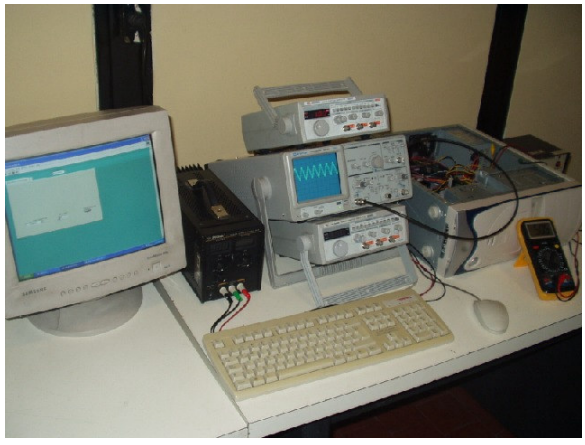
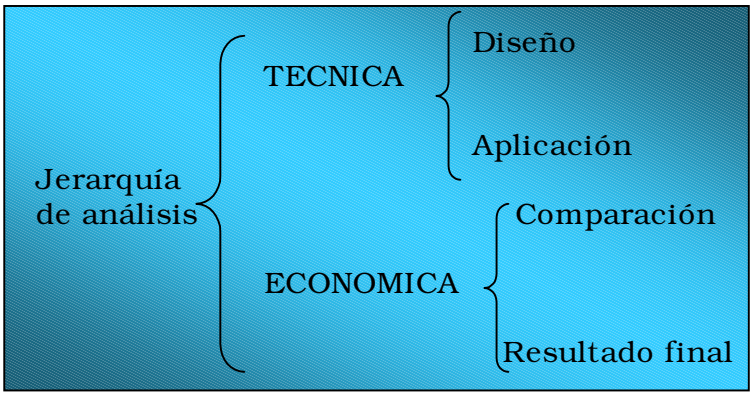


Figura 10 y 11: Transferencia bidireccional de datos – Análisis de señales y Generación de patrones de onda.

[5] Devsel Timing - PCI SPECIFICATION REV 3.0.

6-CONCLUSIONES:

Las conclusiones del trabajo pueden analizarse jerárquicamente de acuerdo a la siguiente estructura:



En el análisis técnico del diseño del proyecto, se destaca la selección de la FPGA para la implementación:

El criterio utilizado para la selección fue adecuado, seleccionándose una FPGA de nivel medio Spartan-3 XC3S400[6] (8.064 logic cells) y con un speed grade –4.

Se concluye en una selección correcta debido a que se superó las exigencias previstas, observando que el espacio en la pastilla en ningún caso supero el 50 %, lo que permite añadir futuras aplicaciones tipo add-on al core realizado, destacando que la versión inferior de FPGA limita seriamente el diseño, y con respecto a la velocidad de respuesta se mostró que con un speed grade -4 la pastilla responde correctamente al funcionamiento del bus de 33 Mhz superando los objetivos temporales determinados.

Profundizando el análisis técnico, debe destacarse que nuestro CORE cumplió los objetivos predeterminados de:

- Sincronismo con bus PCI
- Transferencia bidireccional de datos
- Compatibilidad con dispositivos PCI

Con las herramientas mencionadas puede desarrollarse el CORE PCI y el DRIVER correspondiente, y debe destacarse que con estos dos indispensables elementos: CORE - DRIVER pueden implementarse numerosas aplicaciones en el campo de la tecnología y las señales digitales.

Analizando económicamente el proyecto, se observa que el valor de lo obtenido en función al precio pagado por el mismo, costo-beneficio fue positivo, porque debe considerarse que el proyecto se llevo a cabo con la tarjeta PCI compatible más barata disponible en el mercado (u$d 90.-) y los resultados obtenidos superaron ampliamente todos los objetivos planteados en el proyecto, cumpliendo todos y cada uno de los requerimientos.

[6] Xilinx Spartan 3 Family
ww.xilinx.com/products/silicon_solutions/fpgas/spartan_series/spartan3_fpgas/index.htm