Puente IEEE1284 en modo EPP a bus Wishbone.

Trapanotto, A.; Brengi, D.; Tropea, S.

Instituto Nacional de Tecnología Industrial, Buenos Aires, Argentina,
andres_t@inti.gov.ar, brengi@inti.gov.ar, salvador@inti.gov.ar
http://utic.inti.gov.ar

Resumen. En el diseño de circuitos digitales complejos las distintas funciones se dividen jerárquicamente en módulos independientes llamados *cores* con el objetivo de facilitar su posterior reutilización. Los módulos se describen a través de su comportamiento y su arquitectura de interconexión utilizando lenguajes de descripción de hardware como VHDL. Se presenta en este trabajo el desarrollo de un *core* que hace de módulo de interconexión entre el puerto paralelo de una computadora en modo EPP *Enhanced Parallel Port* y cualquier otro módulo compatible con la interfase *Wishbone*.

1 Introducción

Se presenta en este trabajo el desarrollo de un *core* que hace de puente (módulo de interconexión) entre el puerto paralelo de una computadora en modo EPP *Enhanced Parallel Port* [1] definido en el estándar IEEE1284 [2] y cualquier otro módulo que utilice interfase *Wishbone* [3]. Este módulo y su banco de pruebas se describen utilizando lenguaje VHDL [4]. En el diagrama en bloques podemos observar un esquema general de aplicación.

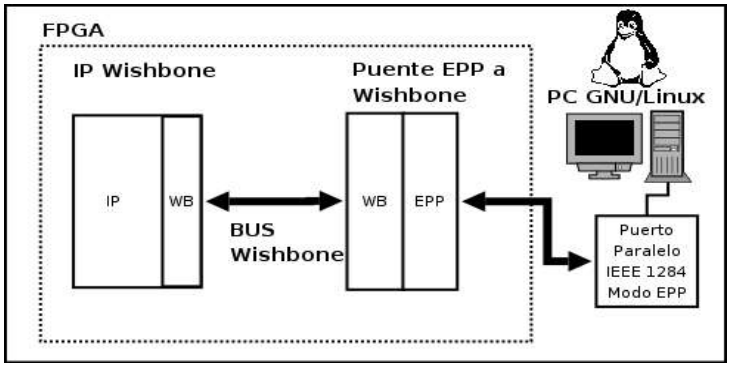

Fig. 1. Diagrama general de aplicación del puente EPP a *Wishbone.*

2 El Puerto Paralelo

La gran mayoría de las computadoras actuales poseen un puerto paralelo. Si bien fue originalmente pensado para el control de impresoras, en la actualidad muchos otros dispositivos

lo utilizan. Explicaremos brevemente las capacidades de este puerto, enfocando particularmente en el modo EPP, por ser éste el utilizado para nuestro trabajo.

2.2 El Estándar IEEE1284

La necesidad de incrementar en forma ordenada las capacidades del puerto paralelo llevaron al desarrollo del estándar IEEE1284. En este estándar se definen 5 modos de transferencia de datos:
- Solo salida: Modo tradicional, compatible con Centronics.
- Solo entrada: Modo *Nibble.* Se utilizan la lineas de estado para ingreso de datos.
- Solo entrada: Modo Byte. Ingresan 8 bits usando el puerto de datos bidireccional.
- Bidireccional: EPP *Enhanced Parallel Port.*
- Bidireccional: ECP *Extended Capability Port.*

En las computadoras actuales, los últimos dos modos poseen hardware adicional que asisten en la transferencia de datos, simplificando las rutinas de software y acelerando las transferencias. La norma también brinda un método para la negociación del modo de transferencia y define la interfase eléctrica y física.

2.3 Las Señales en el Modo EPP

Los nombres y funciones tradicionales de las señales del puerto paralelo se redefinen en el modo EPP según la Tabla 1.

Estándar	EPP		
Nombre	**Nombre**	**Descripción**	**Dirección**
/STROBE	/WRITE	Indica si la operación es de lectura o de escritura.	Salida
/AUTOFEED	/DATASTB	Indica datos válidos durante una operación de escritura. Durante la lectura indica que el *host* está listo para la recepción.	Salida
/SELECTIN	/ADDRSTB	Indica si AD1 a AD8 se trata de una dirección o de un dato.	Salida
/INIT	/RESET	Termina el modo EPP y retorna a SPP	Salida
/ACK	/INTR	Permite que el periférico interrumpa al *host*.	Entrada
BUSY	/WAIT	El periférico indica que se ha completado la transferencia.	Entrada
D1 a D8	AD1 a AD8	Líneas de datos o direcciones.	Bidireccional

Estándar	EPP		
PE, SELECT y /ERROR	Sin definir	Solo utilizados en la negociación.	Entrada

Tabla 1. Nombre y significado de las señales en el modo EPP.

2.4 Negociación y Transferencia de Datos

No es obligatorio que todos los periféricos conectados al puerto paralelo soporten los 5 posibles modos de transferencia. Para determinar las capacidades de cada periférico se implementa un método de identificación. La negociación es una secuencia de eventos en el puerto paralelo que permite identificar las capacidades de un periférico IEEE1284 sin afectar el comportamiento de los dispositivos antiguos. Luego de la etapa de negociación, y si esta resulta exitosa, se ingresa al modo EPP donde la PC puede realizar cuatro operaciones básicas: Escritura de datos, lectura de datos, escritura de dirección y lectura de dirección.

2.5 Utilización del Puerto Paralelo desde el Sistema Operativo Gnu/Linux

El kernel de Linux actual posee soporte para las capacidades extendidas del puerto paralelo a través del módulo *parport* [5] [6]. Este módulo brinda al programador la posibilidad de realizar las operaciones de lectura y escritura sobre el puerto paralelo utilizando el estándar IEEE1284. Para esto se utiliza el dispositivo /dev/parport0 y las operaciones clásicas de lectura y escritura de archivos en lenguaje C: *read* y *write*. Para la negociación inicial y para pasar del modo datos al modo de direcciones se utiliza el mecanismo de comunicación *ioctl*. El acceso al puerto paralelo también puede hacerse a través de la librería libieee1284 [7] que brinda una mayor abstracción que el módulo *parport*. Para este trabajo se utilizó únicamente el módulo *parport*.

3 El Sistema de Interconexión *Wishbone*

Para aprovechar al máximo los desarrollos de *cores* y sacar mayor beneficio de la modularidad que estos brindan, es necesario definir y establecer un mecanismo de interconexión común. Se buscó una especificación abierta y libre para cubrir esta necesidad. Tomando como referencia el proyecto *OpenCores* [8] se selecciona la arquitectura de interconexión *Wishbone* para *SoC*. La especificación *Wishbone* nos brinda un método flexible pero estandarizado de interconexión de *cores*, Además de sus características y ventajas técnicas, esta especificación es de dominio público y se permite su libre utilización sin pago de licencias o *royalties*, razón por la cual es una de las más utilizadas en los diseños *cores* de libre uso y distribución.

4 El Módulo Desarrollado

4.1 Descripción General

En la Fig.2 se presenta un diagrama en bloques del módulo. El bloque de negociación se encarga del diálogo inicial y realiza la conexión del puente una vez establecido el modo EPP con el *host.* Cuando el *host* decide finalizar el modo EPP utilizando la línea de /RESET, el bloque de negociación desconecta el puente.
La compatibilidad entre las señales EPP y el bus *Wishbone* se realiza de la siguiente manera: Las operaciones de lectura y escritura de direcciones se realizan sobre un registro interno al módulo. Las operaciones de lectura y escritura de datos se realizan directamente sobre el dispositivo conectado al bus *Wishbone* en la posición seleccionada por el registro de direcciones.

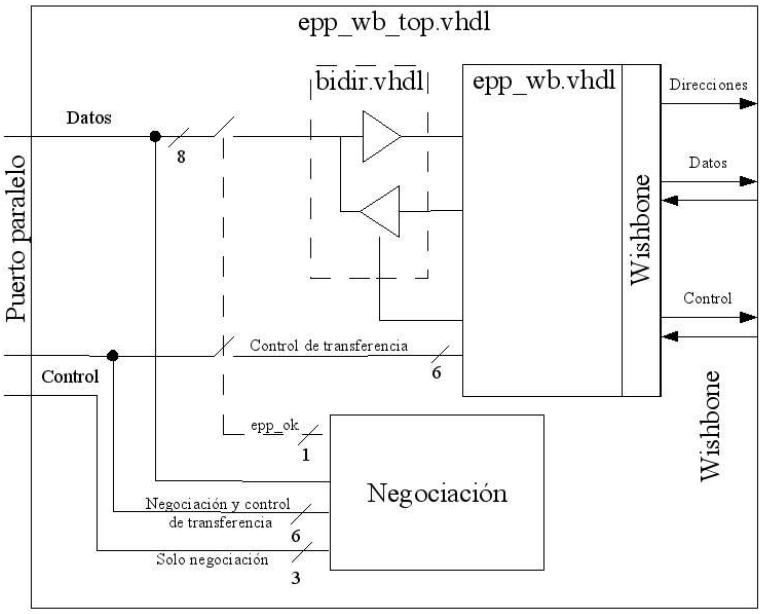

Fig. 2. Diagrama en bloques de la implementación del puente EPP a *Wishbone.*

4.2 El Código VHDL

La descripción está dividida en tres archivos: epp_wb_top.vhdl es el de mayor jerarquía conteniendo las interconexiones principales y el proceso de negociación, epp_wb.vhdl implementa la interconexión, y bidir.vhdl es simplemente un bus bidireccional de tres estados. Para la codificación en VHDL se respetó la sintaxis recomendada por el proyecto FPGALibre [9].

4.3 Herramientas de Software Utilizadas

Se utilizan todas las herramientas del ciclo de desarrollo del proyecto FPGALibre: *GNU Make, ghdl, bakalint, GTK-Wave, ISE WebPack 6.0* y *xilinx-jtag*.

4.4 Verificación por Simulación

Cómo banco de pruebas se implementa un diseño formado por una memoria RAM genérica de 256 bytes compatible con *Wishbone* y conectada a un puerto paralelo a través del puente EPP a *Wishbone*. Dado que la memoria ocupa todo el espacio de memoria disponible, no hace falta implementar lógica adicional de direccionamiento. Por otra parte, como esta memoria es el único dispositivo esclavo conectado al bus *Wishbone*, tampoco es necesaria lógica extra de arbitraje en el bus de datos. De esta forma la conexión es directa entre la memoria y el puente bajo prueba.
Se crea una descripción escrita en lenguaje VHDL que instancia el banco de pruebas antes descripto y recrea las señales en funcionamiento normal de un puerto paralelo. Se analizan las situaciones de: Negociación a modo EPP, escritura y lectura del registro de direcciones y por último escritura y lectura de datos.
Las simulaciones fueron satisfactorias. Se presentan como ejemplo los resultados de algunas de las pruebas visualizadas con *GTK-Wave*.

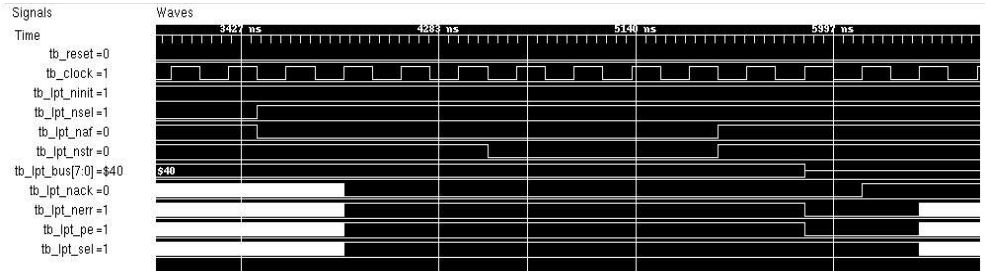

Fig. 3. Resultado de la simulación. Negociación.

Fig. 4. Resultado de la simulación. *Escritura de dirección.*

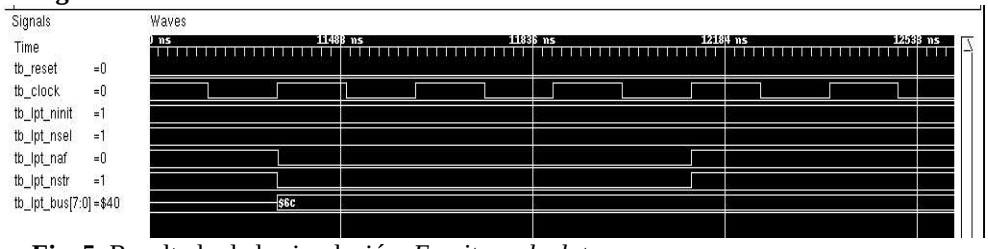

Fig. 5. Resultado de la simulación. *Escritura de dato.*

4.5 Documentación

La especificación *Wishbone* exige una documentación mínima sobre el módulo. En la Tabla 2 se presenta un resumen de la misma:

Hoja de datos WISHBONE para el puente EPP a *Wishbone*	
Descripción general:	Puente EPP a *Wishbone* de 8 bit, compatible con IEEE1284.
Operaciones soportadas:	MASTER, READ/WRITE
Puerto de datos, dimensión: Puerto de datos, granularidad: Puerto de datos, tamaño máximo de operador: Orden de transferencia de datos: Secuencia de transferencia de datos:	8-bit 8-bit 8-bit *Big endian* y/o *little endian* Sin definir
Restricciones del reloj:	Fclk > 200 Khz

Tabla 2. Resumen de los datos exigidos por la especificación *Wishbone*.

5 Verificación en Hardware

Además de la verificación realizada por simulación, se realiza una prueba funcional de la descripción. Para esta prueba se utilizan los siguientes elementos:

- Computadora PC con puerto paralelo en modo EPP.
- Sistema operativo Debian [10] GNU/Linux con kernel 2.6.8.
- Programas de comunicación y prueba en el lado PC realizados en lenguaje C, utilizando el módulo parport para el acceso al puerto paralelo en modo EPP.
- Plaqueta de desarrollo DS-BD-2LSC "Memec Spartan II LC" con una FPGA Spartan II XC2S100 de Xilinx.
- Cable de extensión del puerto paralelo (directo).
- Plaqueta auxiliar para el conexionado entre el cable y la plaqueta de desarrollo. Esta plaqueta posee conectores, testpoints y terminador para cumplir con la norma IEEE1284.
- Osciloscopio digital Agilent 54622D con 16 entradas para canales digitales.
- Descripción VHDL del puente IEEE1284 a Wishbone conectado a una memoria RAM Wishbone.

El programa en la PC realiza la negociación del modo EPP e inicia operaciones de escritura en la memoria conectada al puente. Luego se realiza una lectura de la memoria y se compara el resultado con los datos enviados anteriormente.
Las señales obtenidas en el osciloscopio y el correcto funcionamiento de las rutinas de verificación por software en la PC confirman el comportamiento de los módulos utilizados.
El diseño completo utiliza 32 macroceldas y 1 BlockRAM.
Utilizando un oscilador de 25Mhz para la FPGA se logran transferencias de hasta 330 KBytes/seg.

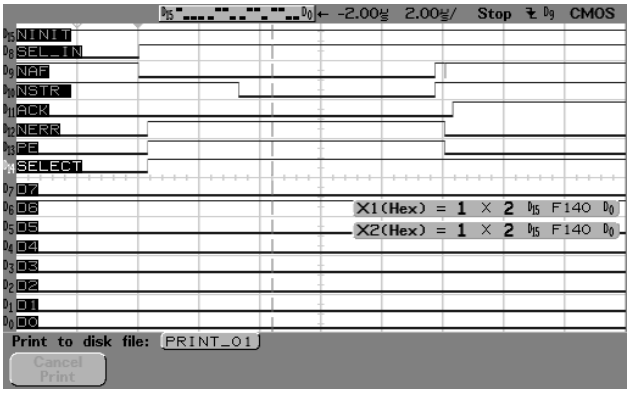

Fig. 6. Señales observadas con osciloscopio. Negociación.

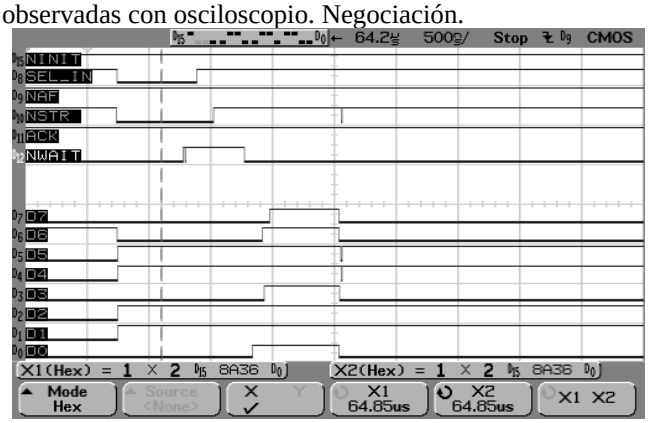

Fig. 7. Señales observadas con osciloscopio. Escritura de dirección.

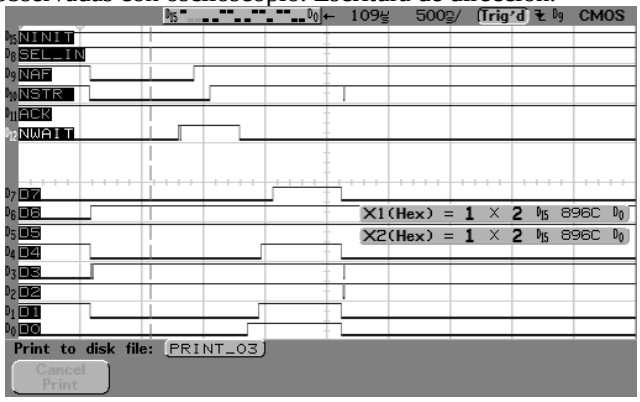

Fig. 8. Señales observadas con osciloscopio. Escritura de dato.

6 Licencia

Este módulo junto a su banco de pruebas y toda la documentación asociada se ha publicado bajo licencia GPL (GNU General Public License) [11] y se encuentra disponible libremente en los sitios de *OpenCores* y FPGALibre.

7 Conclusiones

Con este diseño se logra aprovechar el hardware adicional que casi todas las computadoras modernas incorporan a su puerto paralelo para trabajar en modo EPP. Esta extensión esta especificada en el estándar IEEE 1284 y permite transferencias de entrada y salida de información a alta velocidad y el direccionamiento de varios dispositivos utilizando el puerto paralelo.

El módulo desarrollado permite a una FPGA comunicarse en modo EPP con una PC y se conecta dentro de la FPGA con cualquier módulo compatible con *Wishbone*.

Como verificación se ha conectado este puente a una memoria *Wishbone*. La simulación y la implementación en hardware funcionaron correctamente.

Se espera en el futuro aplicar el puente desarrollado para trabajos que involucren periféricos dedicados conectados a una PC. Esto permitirá resolver problemas de alta complejidad utilizando el puerto paralelo y un solo chip FPGA con todos los módulos de descripción de hardware funcionando en su interior. También podrá utilizarse para realizar validaciones en hardware de *cores* compatibles con el bus *Wishbone*, accediendo al mismo fácilmente desde la PC.

Referencias

1. Interfacing the Enhanced Parallel Port (http://www.beyondlogic.org/epp/epp.htm, verificado 02/dic/05).
2. IEEE Std 1284.1-1997 IEEE Standard for Information Technology—Transport Independent Printer/System Interface (TIP/SI). (http://standards.ieee.org/, verificado 02/dic/05).
3. “The Wishbone System-on-chip (Soc) Interconnect Architecture for Portable IP Cores” (http://www.opencores.org/projects.cgi/web/wishbone/wishbone, verificado 02/dic/05).
4. “Very high speed integrated circuits Hadrware Description Language” (http://www.vhdl.org/, verificado 22/dic/05).
5. Tim Waugh. The Linux 2.4 Parallel Port Subsystem. (http://people.redhat.com/twaugh/parport/html/parportguide.html, verificado 22/dic/05).
6. Jose Renau Ardevol. Sistemas alternativos de interconexión: El IEEE1284 en Linux. (http://www.cse.ucsc.edu/~renau/docs/msurl.pdf, verificado 02/dic/05).
7. Tim Waugh. Libieee1284. (http://cyberelk.net/tim/libieee1284/, http://sourceforge.net/projects/libieee1284/, verificados 02/dic/05).
8. Opencores.org (http://www.opencores.org/, verificado 02/dic/05).
9. FPGALibre. (http://fpgalibre.sourceforge.net, verificado 02/dic/05).
10. Debian GNU/Linux (http://www.debian.org/, verificado 02/dic/05).
11. GPL. GNU General Public License. (http://www.gnu.org/copyleft/gpl.html, verficado 02/dic/05).