Dispositivo genérico portátil de entrada/salida

Nelson Acosta, Daniel Simonelli* y Claudio Aciti
INCA - Departamento de Computación y Sistemas - Facultad de Ciencias Exactas
Universidad Nacional del Centro de la Provincia de Buenos Aires
(7000) TANDIL - Argentina
Email: nacosta@exa.unicen.edu.ar
Web: http://www.exa.unicen.edu.ar/inca/

Keywords: Dispositivos, sistemas de entrada/salida, embedded systems, sistemas dedicados, FPGA

RESUMEN

Normalmente los dispositivos portátiles dedicados a la captura de datos son muy especializados y costosos. Por otro lado, siempre es posible crear dispositivos a medida de la aplicación. Esta opción, si bien es muy frecuente en ambientes académicos, sólo se justifica cuando el lugar del desarrollo presenta exigencias económicas o tecnológicas de gran peso dado que para cada aplicación se requiere diseñar y construir hardware y software específicos. En el presente artículo, se presenta un dispositivo basado en FPGA (*Field Programmable Gate Array*) que permite reutilizar gran parte del hardware (por medio de una configuración adecuada) y el software de control en un entorno de tiempo real como RTL (*Real Time Linux*).

I. Introducción

Una situación bastante común en el ámbito de la ingeniería es la evaluación de dispositivos nuevos. Dicha labor involucra controlar diversas funciones y planificar como leer y escribir en sus registros internos; generadores de datos y analizadores lógicos, entre otros.

Otra solución consiste en utilizar una placa de entrada/salida dedicada la cual suele ser costosa y requerir software especial. La conexión de cualquier actuador o sensor a una computadora debe efectuarse a través del bus de entrada/salida de la computadora y esta condición no se cumple en forma natural. Normalmente se diseñan dispositivos *ad hoc* para satisfacer esta necesidad lo cual conlleva un considerable esfuerzo económico.

Hay placas de entrada/salida que cubren en gran medida esta brecha. Dichas tarjetas se conectan al bus de la computadora (ISA, EISA, PCMCIA, SCSI, VME, etc.) sin embargo son específicas para cada aplicación y, por ende, costosas.

Una buena alternativa surge de considerar un puerto de entrada/salida que exista virtualmente en todas las PC's y *notebook*, por ejemplo el puerto paralelo. En [Kav00] se detalla como es

* - Becario de perfeccionamiento de la Comisión de Investigaciones Científicas de la Provincia de Buenos Aires

posible medir temperaturas a través de este puerto utilizando una simple placa basada en el circuito integrado AD7821. Sin embargo, a pesar de tratarse de un dispositivo sencillo y de bajo costo, sigue siendo específico para esta aplicación; otra variable a ser analizada involucra el diseño de circuitería *ad hoc*.

El objetivo de este trabajo es presentar una alternativa a dicha situación a través del empleo de lógica configurable (FPGA). A fin de cubrir un amplio espectro de dispositivos de entrada/salida, se han considerado los siguientes:

- convertidores analógico/digitales
- convertidores digitales/analógicos
- codificadores angulares (*encoders*)
- motores paso a paso
- servomotores
- control de motores DC
- memorias externas con datos recolectados 'en línea'
- relés
- sensores ópticos

II. Diseño para entrada/salida genérica

Para dotar al diseño de mayor generalidad se han fijado los siguientes objetivos:

a) Reutilización total o parcial del software de la aplicación;
b) Reutilización total de los drivers;
c) Reutilización total de la electrónica digital;
d) Asistencia total para el (re)diseño de la configuración de la electrónica digital;
e) Reutilización parcial de la electrónica analógica;
f) Ejecución sobre un sistema operativo que brinde servicios de tiempo real;
g) Utilización de una computadora portátil con un mínimo de 10 Megabytes de RAM como plataforma (restricción impuesta por la característica anterior); y
h) Conexión de la placa a la computadora por medio del puerto paralelo.

Para poder responder a los requerimientos expuestos, se diseñó una placa basada en una FPGA. Conceptualmente, se procura aprovechar las ventajas conjuntas que surgen de la configurabilidad de la lógica programable y la versatilidad del puerto paralelo utilizado; tanto para la configuración del hardware como para la aplicación del usuario.

En estas condiciones, es posible conectar a una PC diversos dispositivos a través de la placa propuesta, relevando al diseñador de la tediosa tarea de materializar una tarjeta electrónica para su aplicación. Sólo se necesita efectuar un diseño lógico para configurar a la FPGA y escribir el código del driver.

La principal motivación para el enfoque escogido estriba en el hecho de que los ingenieros están optando por usar FPGA's en lugar de ASIC's porque son flexibles, de rápido uso, y mucho más accesibles que tiempo atrás. Hay que tener en cuenta el hecho de que en el mercado actual el ciclo de vida de un producto es a menudo más corto que su ciclo de desarrollo. Las presiones de la salida al mercado son aún más fuertes que las imperantes hace unos pocos años. Para lidiar con esta presión los diseñadores necesitan más flexibilidad de diseño.

La flexibilidad de las FPGA's basadas en RAM estática (SRAM) proviene de su inherente naturaleza reconfigurable. Esto permite agregar funcionalidad adicional a un producto final sin tener que modificar físicamente la placa. Por el contrario, para cambiar la funcionalidad de un ASIC hay que hacer un nuevo ASIC, lo que resulta en cambios en la placa.

El contar con esta flexibilidad hace que el factor de riesgo disminuya, lo que muchos diseñadores encuentran muy deseable. Con las FPGA's los errores pueden corregirse y las revisiones al diseño pueden hacerse simplemente

reconfigurando la FPGA. Comparativamente, si el diseño se implementara a través de un ASIC cualquier error requeriría crear un nuevo conjunto de máscaras, lo cual consume tiempo y dinero.

Hay un punto para cualquier proceso de diseño en el cual la cantidad, el tiempo y las consideraciones de tipo económico vuelcan la balanza hacia diseñar con ASIC's o con FPGA's. A veces, la decisión es muy fácil. Si se tiene en cuenta que el ciclo de diseño de un ASIC involucra 18 meses, si se desea reducirlo a unas pocas semanas hay que utilizar FPGA's. Otras veces no es tan fácil la toma de decisión.

Hay fuertes argumentos para demostrar que si bien es cierto que el proceso de diseño basado en FPGA's es mejor respecto al de diseño con ASIC's, nunca los reemplazarán. Si se necesita, por ejemplo, un (micro)controlador, un convertidor analógico a digital ó digital a analógico en un sistema, dicha funcionalidad no se puede integrar dentro de una FPGA. Esto se puede lograr con un ASIC y núcleos de IP's (*propiedad intelectual*) empotrados, lo que dio origen a lo que hoy en día se conoce como "*System on a chip*" (SOC's) [Fau97], [Mor97], [Hor97] y [Kra97].

II.a. Electrónica digital

Todos los dispositivos necesitan una electrónica básica para acceder al bus de la computadora además de la electrónica dedicada al control de cada uno de los dispositivos en sí (codificadores angulares, convertidores, servomotores, motores DC o PAP). Estos dos bloques electrónicos pueden diseñarse y materializarse por separado o en un único chip (ASIC). En el caso de aplicaciones con requerimiento dinámico es recomendable que dicha plataforma se configure externamente. Las FPGA proporcionan una gran cantidad de puertas digitales configurables externamente, por lo cual es posible integrar en un único chip toda la electrónica digital de la aplicación. Así sólo será necesario diseñar la parte digital de los sensores o actuadores (tratados normalmente como módulos de biblioteca).

Por disponibilidad en el laboratorio del INCA y estandarización se utiliza una FPGA de Xilinx modelo XC4010E con encapsulado PLCC de 84 pines. Este modelo dispone de una capacidad de 10000 puertas lógicas equivalentes, con 1120 flip-flops, 61 puertos de entrada/salida libres para el usuario, líneas rápidas para la transmisión de señales de reloj, pines de entrada y salida programables (buffer, triestado, inversión, registro, etc).

La arquitectura básica de una FPGA se define por medio de tres grandes bloques: lógica, entrada/salida y conexiones. La mayor cantidad de área está cubierta por un arreglo de dos dimensiones que implementa la lógica del diseño. Ese plano es rodeado por los bloques de entrada y salida que deben adecuarse a las necesidades de la aplicación. Por último, el plano anterior está cubierto por otro encargado de la interconexión de todos los componentes.

La lógica digital se implementa en una FPGA utilizando: a) componentes básicos tales como compuertas y unidades de memoria, b) pequeñas tablas de consulta (LUT o *Look-Up Table*) que permiten implementar cualquier función lógica de varias entradas. Las FPGA de Xilinx utilizan LUTs.

II.b. Linux de Tiempo Real

RTLinux (*RealTime Linux*) es una extensión a Linux que maneja tareas de tiempo crítico [Bar96], [Bar97] y [Yod99a]. RTLinux en lugar de modificar al núcleo para hacerlo predecible, usa uno más pequeño y de tiempo real (sobre el procesador i386). Así, el núcleo de tiempo real comparte uno o más procesadores con el núcleo estándar. Linux es ejecutado como una tarea en segundo plano, mientras se hayan completado las tareas de tiempo real y disponga recursos para liberar, se le permite ejecutar cualesquiera

tareas que se le asignen. De esta forma el sistema se puede utilizar para aplicaciones tales como adquisición de datos, control, y robótica; mientras que aún se comporta como una estación de trabajo Linux estándar.

En RTLinux todas las interrupciones son manejadas inicialmente por el núcleo de tiempo real; cuando no hay más tareas de tiempo real para ejecutarse son enviadas a la tarea Linux. RTLinux divide las interrupciones en dos grupos: aquellas que se hallan bajo el control de Linux y las que son controladas por RTLinux. Las aplicaciones de tiempo real consisten de tareas que se incorporan como módulos del núcleo factibles de ser cargados, y procesos Linux/Unix que se encargan de asignación de datos, display, accesos de red y muchas otras funciones que no se hallan restringidas por el comportamiento del peor caso. El organizador de tareas por defecto que viene con RTLinux es predictivo, con prioridad fija y considera a la tarea Linux como la tarea de prioridad más baja [Den99], [RTL99], [UNLP] y [Aco00].

Adicionalmente al desarrollo de la aplicación en sí, para sistemas empotrados se debe afrontar el desafío de realizar la instalación del sistema operativo en máquinas no demasiado actualizadas, con muy poca memoria RAM y con un disco rígido de muy bajas prestaciones [Epp99] y [Cur00].

II.c. Puerto paralelo

Por generalidad, disponibilidad y bajo costo se decide utilizar el puerto paralelo de la computadora para controlar la placa de entrada y salida genérica.

Así, el puerto paralelo es la interfaz más utilizada para proyectos a medida. Los puertos paralelos actuales cumplen con el estándar de IEEE 1284, cuya primera versión data de 1984 [IEE84]. Aquí se definen los cinco modos de operación del puerto: a) Compatibilidad, b) *Nibble*, c) Byte, d) EPP (*Enhanced Parallel Port*), y e) ECP (*Extended Capabilities*). Los modos Compatibilidad, Nibble y Byte son los que mantienen la retrocompatibilidad con el SPP. Los modos EPP y ECP requieren hardware adicional pero pueden trabajar a mayores velocidades. En este caso el driver se diseña para que trabaje en modo Byte o SPP.

El puerto paralelo tiene señales activas en nivel bajo. Algunas de estas señales se invierten por hardware. Por ejemplo, si se aplica +5v al pin de la señal ¬*Busy* al leer el bit de estado correspondiente habrá un cero.

Las salidas del puerto paralelo trabajan normalmente a niveles lógicos TTL. La corriente que pueden drenar o alimentar es de aproximadamente 12 mA, pero normalmente es muy común que no llegue a tales valores, oscilando entre los 4 y 6 mA.

Los puertos paralelos normalmente se encuentran en las direcciones del bus de entrada/salida 0x03bc, 0x0378 y 0x0278 para LPT 1, 2 y 3 respectivamente. Cada puerto consiste de tres registros: el puerto de datos, el registro de estado y el registro de control. El puerto paralelo normalmente utiliza una ficha de conexión DB-25 hembra.

DB25	Centronics	SPP	Registro	Bits Registro
1	1	strobe	control	0
2-9	2-9	datos	dato	0-7
10	10	ack	estado	6
11	11	busy	estado	7
12	12	paper Out	estado	5
13	13	select	estado	4
14	14	line feed	control	1
15	32	error	estado	3
16	31	initialize	control	2
17	36	select-in	control	1
18-25	19-30	ground	- - -	- - -

El puerto paralelo normalmente genera una interrupción por IRQ5 o IRQ7. Las

interrupciones de este puerto son controladas por el bit 4 del registro de control (*IRQ enable via ACK line*). Una vez habilitadas, las interrupciones pueden ocurrir normalmente en la transición de subida de la señal ACK (de 0 a 1), aunque en algunos casos hay puertos que trabajan con las transiciones de bajada (de 1 a 0). Si se produjo una interrupción en la línea ACK, se indica en el bit 2 del registro de estado.

Computadora		Placa FPGA		Detalles varios
DB25	SPP	FPGA	Pin	
1	strobe	-program	55	
2-9	datos	datos	(*)	
10	ack	i/o	35	con pull-up
11	busy	ready/bsy	70	
12	paperOut	done	53	
13	select	i/o	84	
14	linefeed	-init	41	con pull-up
15	error	i/o	49	
16	initialize	cclk	73	con pull-up
17	select-in	i/o	3	
18-25	ground	ground		

(*) Pines: 23, 20, 19, 18, 17, 16, 15, 14

Dispositivos		Pines FPGA
Codificador 1	canal A	4
	canal B	5
Codificador 2	canal A	6
	canal B	7
Convertidor A/D	d7 a d0	77 a 83
	a0, a1, a2	8, 10, 13
	mode	24
	int	25
	cs	26
Controlador PAP	oe	65
	strobe	62
	in4 a in1	58 a 61

El bit 5 del registro de control permite que el puerto paralelo trabaje con las 8 líneas de datos para entrada y salida. Cuando este bit está activo, los pines 2 a 9 entran en estado de alta impedancia. Entonces, se puede leer datos a través de las líneas 2-9. Cualquier dato que se escriba en el puerto de datos se almacenará pero no estará disponible en los pines 2-9 hasta que el bit 5 del registro de control sea 0.

También se utiliza un *display* de siete segmentos conectado al bus de datos y un conjunto de leds como indicadores de los estados de la FPGA durante la configuración.

II.d. Configuración de la FPGA

Hay cuatro pasos principales con relación a la carga de la configuración en una FPGA: borrado de la memoria de configuración, inicialización, configuración e inicio (*start-up*).

Cuando la alimentación se conecta a la FPGA, un circuito interno fuerza la inicialización de la lógica de configuración. Cuando los niveles de tensión son adecuados, se realiza una verificación interna de la memoria de configuración, mediante un ciclo de escritura y lectura, y se inicia una demora de 16 MS. Este retraso se aplica sólo al conectarse la alimentación, nunca para una reconfiguración. Durante este tiempo o mientras la señal PROGRAM sea 1, la FPGA borra la memoria de configuración. Luego que la señal INIT se mantenga 250 nanosegundos en nivel alto se inicia la segunda etapa.

Durante las etapas de inicialización y configuración, las señales LDC e INIT se mantienen en nivel bajo. Luego que la señal INIT se reconoce en nivel alto, comienza la detección del modo de configuración escogido (a través de las líneas M0, M1 y M2).

La configuración a la FPGA se envía por medio de bloques, cada uno de los cuales está formado por un código de descripción, información y un código de error. Si hay un error en un bloque, toda la configuración se cancela, y se baja la señal INIT.

La familia XC4000e dispone de 3 eventos para determinar que la configuración está completa: sube la señal DONE, se desactivan los *set/reset* internos y se activan los pines de entrada/salida del usuario. Estos eventos pueden ocurrir en

cualquier secuencia. Por defecto, los pines de entrada/salida del usuario se liberan un ciclo de reloj después de que el pin DONE se encuentra en nivel alto. Si la señal DONE no sube, los pines de entrada /salida del usuario permanecen en alta impedancia a través de un *pull-up* formado por una resistencia cuyo valor se encuentra entre 50 y 100 Kohms.

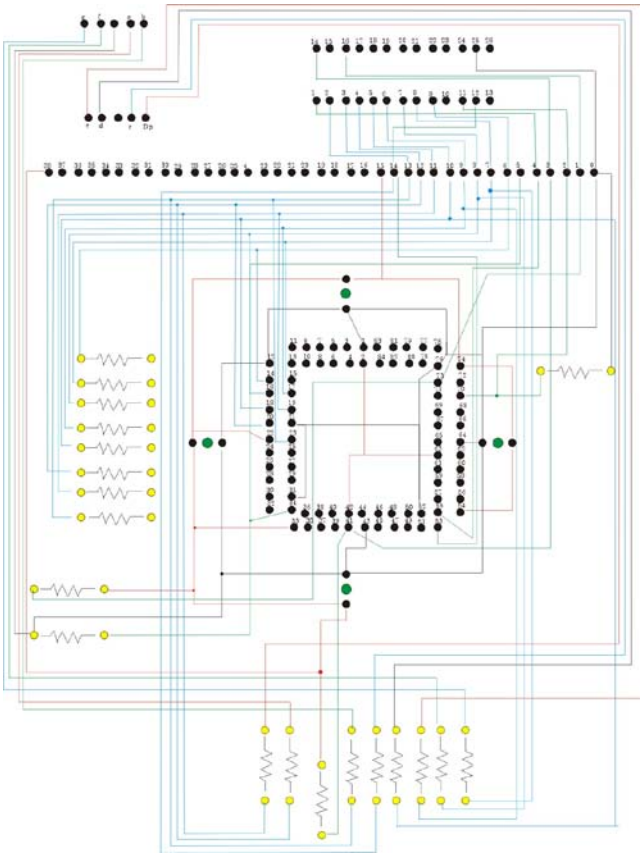

Fig. 1 - Diagrama global de la placa

Para la configuración de la FPGA, se entra en el modo "*Periférico Síncrono*" (también conocido como modo "*Paralelo Esclavo*"). Es la PC la encargada de generar la señal de reloj (CCLK) para la configuración. Cada byte de configuración debe estar disponible 60 nanosegundos antes que se produzca el flanco de subida de CLK. Así, cada byte de dato se lee sobre cada octavo flanco de subida de la señal CLK. La señal CLK también causa que la señal RDY/BUSY suba y se mantenga por espacio del periodo de un CLK. La señal RDY/BUSY se fuerza a valor alto por medio de un *pull-up* por alta impedancia, previo a que INIT esté en nivel alto. La señal RDY/BUSY está disponible como una señal de *handshake*. La FPGA es la encargada de serializar los datos, por medio de los 8 flancos siguientes sobre CLK.

II.e) Placa de prototipo

La placa se materializa conteniendo la FPGA y los dispositivos básicos (convertidor analógico digital, codificador angular, display de siete segmentos, motor PAP). En la Fig. 1 se muestra el diagrama de conexiones sólo de la FPGA con sus conexiones al DB25 y todas las señales necesarias para la configuración.

Fig. 2 - Diagrama global del sistema

En el hardware configurable soportado por medio de la FPGA, se implementa el control de los dispositivos. Con este propósito, se definió una arquitectura con un bus interno de 8 líneas de datos y 9 de control (Fig. 2). De esta forma, los elementos controlados (como máximo 4) utilizan un detector de direcciones a modo de selector de chip. A continuación se describe el empleo de un convertidor A/D, dos codificadores angulares que generan interrupción a la PC y dos controladores de

motores PAP.

III. Diseño de aplicaciones

Si bien la FPGA puede ser configurada para soportar muchos dispositivos distintos, en esta etapa del proyecto sólo se han construido los siguientes módulos de interfaz.

Convertidor A/D. Se puede utilizar un convertidor tipo MAX117CPI de *Maxim*, o el ADC12038CIN o el ADC0858CIN de *National Semiconductor*. En cualquier caso, la FPGA selecciona el canal y recoge los valores para almacenarlos en registros internos. El ADC12038 también puede generar la señal de reloj para la conversión aunque su oscilador interno no es muy confiable. Con el ADC0858 la FPGA también realiza el multiplexado de la señal de interrupción a la PC. Por último, como característica relevante se destaca el hecho de que la PC siempre tiene acceso a la información de cualquier canal.

Codificador angular. Se utilizan codificadores angulares como el HEDS-5701G00 de *Hewlett Packard* con la capacidad de generar 360 pulsos por revolución en dos canales. Dichos canales (A y B) se hallan desplazados 90 grados (lo que permite calcular el sentido de giro, la posición y la velocidad). Así, se diseña un módulo encargado de mantener actualizado el valor del ángulo desplazado. Se puede materializar un codificador absoluto o relativo de 8 a 24 bits. Estos últimos pueden generar una señal de interrupción a la PC en caso de desbordamiento.

Control de motores PAP. Se utiliza como driver de potencia el L6219 de *SGS-Thompson* o el UCN5814 de *Sprague*. La FPGA genera la secuencia de pulsos a la frecuencia definida por la velocidad y el sentido de giro. Si el driver dispone de un registro interno, se genera la señal de reloj. Caso contrario, se utiliza uno definido externamente. Si la precisión de la velocidad no es relevante se puede utilizar también el oscilador interno.

IV. Análisis de los resultados

Este diseño surge como respuesta a los requerimientos establecidos por un proyecto que se desarrolla en las instalaciones del INCA. Entre las principales necesidades se puede citar la obtención de datos de dos codificadores angulares, de 8 señales analógicas y de varias señales de entrada y salida digital. Además, la aplicación debe ser portátil, ya que debe interactuar sobre un automóvil en viaje.

Todos los valores adquiridos corresponden a una instantánea del estado del vehículo, el margen de error es limitado por el tiempo de respuesta máximo del sistema.

El diseño y la materialización de una aplicación de tiempo real que utiliza esta placa involucra tanto o más tiempo que el necesario para el desarrollo del dispositivo, dependiendo de cuán exigente sea la interfaz al usuario (de existir). Por otra parte, si el sistema es dedicado, el desarrollo se ve afectado por la instalación del sistema operativo, problema fundamentalmente originado por falta de espacio o el uso de dispositivos sin driver.

Los porcentajes de utilización de las FPGA's, considerando los modelos XC4005 y XC4010 son del 75% y del 53%, respectivamente. El tiempo de reconfiguración es menor a un minuto. Adicionalmente, para estos dispositivos la electrónica analógica adicional requerida es mínima.

La placa diseñada se conectó a un *notebook Texas Instruments* dotada con un procesador Pentium ejecutando a 100 MHz con 8 Mbytes de RAM. El software se compiló utilizando la versión de RTL2.2 [Yod99b], aunque la plataforma donde se ejecuta corre con la versión Mini-RTL 2.0 Pre 5 [McG00]. Finalmente, cabe citar que se ha desarrollado una distribución muy reducida, orientada a sistemas empotrados, que utiliza muy poco espacio tanto de RAM como de disco rígido [Cur00].

V. Conclusiones y futuros trabajos

No se puede concluir que con esta técnica de diseño de aplicaciones se resuelvan todos los grandes problemas del ciclo de diseño; sin embargo, el reuso tanto del hardware como del software permite una reducción considerable del tiempo de desarrollo de la aplicación. Por otro lado, se ha dado un primer paso hacia el diseño de un dispositivo genérico que sirva de interfaz con cualquier otro dispositivo (*un meta-dispositivo*). Aún no se ha logrado el diseño de un driver que permita la conexión de gran variedad de dispositivos sin modificación alguna.

Además esta optimización será más significativa al aumentar la cantidad de módulos de las librerías (de software de aplicación, de los drivers, y de las configuraciones para la FPGA), y está en fase de desarrollo un sistema de asistencia para el diseño de aplicaciones empotradas de tiempo real basadas en este tipo de placas.

Se está diseñando una versión de la placa utilizando el chip FIPSOC. Por medio de este enfoque se procura reducir considerablemente la cantidad de electrónica adicional; ya que se utilizará el área analógica configurable del FIPSOC para realizar el tratamiento de las señales analógicas de entrada, así con este esquema sólo habrá que materializar la electrónica de potencia fuera de la FPGA.

Agradecimiento

Este trabajo no podría haberse realizado sin la colaboración y asistencia técnica del Ing. Hugo Javier Curti.

Referencias

[Aco00] Nelson Acosta, “Real Time Linux: the device maker handbook”. Reporte interno INCA, Marzo 2000.

[Bar96] Michael Barakanov and Victor Yodaiken, "Real Time Linux", http://luz.cs.nmt.edu/~rtlinux, March 3, 1996.

[Bar97] Michael Barakanov, "A Linux based Real-Time Operating System", Master of Science in Computer Science, New Mexico Institute of Mining and Technology, Socorro, New Mexico, June 1997.

[Cur00] Hugo Curti & Nelson Acosta, "RTLinux in a portable computer with minimal configuration", Reporte Interno INCA, UNCPBA, Tandil, Argentina. Mayo 2000.

[Den99] L. Deng & Z. Gao, "RTLinux installation and testing", http://www.embedded.com, October 1999.

[Epp99] Jerry Epplin, "Linux as an Embedded Operating System", http://www.embedded.com, October 1999.

[Fau97] Julio Faura, Miguel A. Aguirre, Juan M. Moreno, Phuoc van Duong, Josep M. Insenser, "FIPSOC: A Field Programmable System On a Chip", DCIS'97

[Hor97] Julio Faura, Chris Horton, Phuoc van Duong, Jordi Madrenas, Miguel Angel Aguirre, and Josep Maria Insenser, A Novel Mixed Signal Programmable Device with On-Chip Microprocessor", Custom Integrated Circuits Conference 1997 (CICC'97)

[IEE84] "IEEE 1284 Protocol Analyzer: User's Guide", Genoa Technology, Dic. 1995

[Kav00] Ken Kavanagh, "Use a PC Printer Port as a Control Device When Evaluating New Designs", Electronic Design, April 17, 2000

[Kra97] Julio Faura, Chris Horton, Bernd Krah, Joan Cabestany, Miguel Angel Aguirre, and Josep Maria Insenser, "A New Field Programmable System-on-a-chip for Mixed Signal

Integration", European Design & Test Conference 1997
[McG00] Nicholas Mc Guire, “Mini-RTL Technical reference”, Universitaet Wien, Inst. F. Materialphysik. http://www.thinkingnerds.com/projects/minirtl/.
[Mor97] J.M.Moreno, J.Cabestany, J. Faura, C. Horton, P.van Duong, M.A. Aguirre, J.M.Insenser, "FIPSOC. A Novel Mixed Programmable Device for System Prototyping, MIXDES'97 (re-edited in a post-proceeding book by Kluwer Academic Publishers)
[RTL99] "RT-Linux Manual Project", RT-Linux Documentation Group, http://www.rtlinux.org, June 1999.
[UNLP] “Testing Real-Time Linux”, http://www.fisica.unlp.edu.ar.
[Yod99a] Victor Yodaiken, "The RTL Manifesto", New Mexico, USA, http://www.rtlinux.org
[Yod99b] Victor Yodaiken and Michael Barakanov, "RTLinux version two", http://www.rtlinux.org