Diseño y Analisis Comparativo de un Microprocesador RISC Básico y Controladores Perifericos Especiales

Leyva G[1], Jaramillo R[2], De la Parra I[2]., Esparza J[2], Carreras C[1].
[1] ETSIT, Universidad Politécnica de Madrid, España
gleyvah@yahoo.com
[2] Universidad Autónoma de Aguascalientes, México
r_jara@hotmail.com

RESUMEN

El diseño de sistemas digitales está cambiando rápidamente hacia partes estándar de aplicación específica (ASSP) cuya configuración esta basada en Lógica Programable (CPLD's y FPGA's). Las alternativas para el diseño digital han aumentado enormemente con la consolidación de las FPGA's, con lo cual los diseñadores deben de explorar diversas alternativas de implementación. Los SoC reconfigurables, también llamados Plataformas FPGA tienen un costo inicial mínimo y permiten el diseño de microprocesadores y periféricos en un solo chip. Esta solución queda muy bien situada en diseños de baja producción. En este trabajo se muestra el diseño de un microprocesador RISC al cual se han añadido controladores especiales de hardware: un controlador de teclado matricial, un controlador de pantalla de cristal líquido, una interfaz paralela programable y un puerto de comunicaciones en serie. También se muestra un sistema de microprocesador comercial al cual se han añadido solamente los controladores. Los resultados obtenidos muestran que la solución en un solo chip es menos rápida, debido al RISC, mientras que la solución de microprocesador externo y controladores periféricos en un chip es más rápida, requiere de una FPGA más barata y el esfuerzo de diseño es menor.

PALABRAS CLAVE:
FPGA, Periféricos, Pipeline, RISC, SoC.

I. SoC

Los circuitos de Lógica programable han incrementado su presencia en el Diseño Digital. Estos ya no están destinados a sustituir compuertas lógicas, decodificadores de direcciones o funciones especiales. El diseño de sistemas digitales está cambiando rápidamente hacia partes estándar de aplicación específica (ASSP) cuya configuración esta basada en Lógica Programable (CPLD's y FPGA's).

Diseñar un sistema para FPGA's no es una tarea fácil especialmente si el diseño incluye microprocesadores. Algunos diseñadores implementan sistemas en circuitos integrados (System-on chip, Soc) manteniendo fijo el diseño del microprocesador y usando frecuentemente algunos periféricos estándar. Otra manera de implementar un SoC es incorporando un arreglo de periféricos dentro de un Circuito Integrado de Aplicación Específica (ASIC), sin embargo esta solución conlleva costos iniciales de diseño más altos. Otra alternativa que se podría plantear es el uso de un microprocesador estándar con todos los periféricos dentro de un chip de Lógica Programable, reduciendo con esto el número de chips de periféricos.

Los SoC reconfigurables, también llamados Plataformas FPGA[9], tienen un costo inicial mínimo y además, gracias a sus inmensos recursos en Silicio (algunos hasta 10 millones de compuertas) permiten la implementación de microprocesadores y periféricos en un solo chip: Esta solución queda muy bien situada en diseños de baja producción o en prototipos de Sistemas Digitales. Por supuesto el diseño de SoC's reconfigurables presenta ventajas y desventajas.

Entre las principales desventajas podemos mencionar:

- Se requiere mucha experiencia para diseñar en FPGA's de gran capacidad.
- El diseño de microprocesadores y periféricos en FPGA's es menos eficiente que los ASIC estándar.
- El costo de las FPGA's es alto. Sin embargo, los SoC reconfigurables pueden ser utilizados
- en productos con una producción reducida (1000 o menos piezas) o en diseños donde el costo del chip es una fracción muy pequeña del costo del producto.

Por su parte, el diseño basado en sistemas reconfigurables ofrece las siguientes ventajas:

- Reducción de costos de diseño, porque los cambios al Chip pueden hacerse inmediatamente durante la fase de desarrollo.
- Reducción en horas de uso de simuladores, porque el Hardware real está disponible inmediatamente.
- Las actualizaciones y correcciones de fallas pueden ser realizadas en campo, debido a que todo el hardware puede ser modificado en cualquier momento, además del código de aplicación.

J. Díaz de León, G. González, J. Figueroa (Eds.): Avances en Ciencias de la Computación, pp. 295-299, 2003.

A. Diseño basado en Componentes

La especificación de sistemas configurables es hecha en lenguajes de Hardware tales como VHDL Verilog. Frecuentemente la especificación se realiza a nivel de transferencia de registros (RTL) lo que posibilita que los diseños sean transferibles a ASIC's una vez que el diseño ha sido probado o que el volumen de producción se incrementa. Una de las razones principales para usar lógica programable es la rapidez con la que pueden realizarse los diseños, pues con las FPGA's actuales pueden realizarse en unas pocas semanas. Pero no siempre es posible realizar esto, los diseños con FPGA's de millones de puertas [11], pueden tomar algunos meses de desarrollo. Para reducir este tiempo los fabricantes de FPGA's han desarrollado bloques de propiedad intelectual (IP) que son módulos que pueden ser insertados en el diseño. Tales IP van desde módulos simples tales como contadores o decodificadores, hasta módulos complejos, tales como interfaces PCI, módulos RAM y bloques de microprocesadores. En el siguiente apartado se describe un SoC que está compuesto por un microprocesador RISC de ocho instrucciones y algunos periféricos que podría formar parte de un sistema de control programable orientado al ámbito industrial.

II. IMPLEMENTACIÓN DEL SoC

A. Arquitectura RISC

La arquitectura RISC tiene como objetivos ciclos de reloj mucho más pequeños, la minimización del número de estos ciclos para la ejecución de instrucciones y el soporte necesario para el "pipelining". Para llevar a cabo lo anterior es necesario el empleo de compiladores más sofisticados, lo cual, conjuntado con su reducido número de instrucciones derivan en requerimientos mayores de memoria de programa normalmente en el rango de 20% y 30% más. Finalmente, la arquitectura RISC requiere menor área de silicio que su contraparte el CISC.

Procesamiento paralelo y "pipeline" son las técnicas generales dominantes en el mejoramiento del desempeño cuando se aborda el diseño de procesadores [1]. Estas técnicas han sido ampliamente usadas en el diseño de procesadores de propósito general. Existe un gran número de características[2] de los procesadores de propósito general que puede ser usado en procesadores de uso específico.

B. Diseño del microprocesador RISC

El modelo base para este diseño puede verse en [3]. A este modelo se le ha añadido nueva funcionalidad, como acceso a memoria de programa, reformas al conjunto de instrucciones y acceso a memoria de datos externa.

1) Definición del conjunto de instrucciones

Para este pequeño microprocesador se han seleccionado ocho instrucciones básicas. La tabla 1 muestra el conjunto de instrucciones.

MOVE <source1>,<destination>
ADD <source1>,<source2>,<destination>
SUB <source1>,<source2>,<destination>
CJE <source1>,<source2>,<code>
LOAD <value>,<destination>
INPUT <value>,<destination>
OUTPUT <value>,<source1>
NOP No instrucción.

Tabla 1. Descripción del conjunto de instrucciones del microprocesador

2) Definición arquitectural

Con el conjunto de instrucciones definido, el segundo punto que trataremos es la interfaz del microprocesador. Para mantener el diseño de este dispositivo tan simple y económico en área como sea posible, se consideraron las siguientes características:

- Memoria de programa, de un ancho de palabra de 32 bits y una longitud máxima de 256 instrucciones.
- 16 registros internos de 16 bits para uso corriente durante la ejecución del programa.
- Acceso a memoria RAM 256x16 (dentro de la FPGA)
- Acceso a controladores periféricos construidos dentro de la FPGA.

La fig. 1 muestra el diagrama a bloques del microprocesador y los dispositivos periféricos.

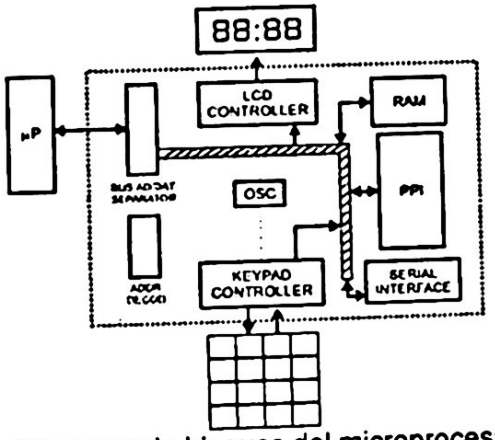

Fig. 1. Diagrama de bloques del microprocesador y controladores periféricos.

3) Definición del "*pipeline*"

El diseño del microprocesador está basado en "*pipeline*". Para mayor información acerca de las ventajas y desventajas de este tipo de diseños puede consultarse [4]. El microprocesador está definido en cuatro etapas de "pipelining".

La etapa de predecodificación es la encargada de realizar la interfaz con las instrucciones de

programa. Las señales de entrada para esta etapa son recibidas del módulo de instrucciones. La salida de esta etapa provee toda la información necesaria a la etapa de decodificación de instrucciones.

En un microprocesador diseñado en "pipeline" las instrucciones deben pasar a través de cada etapa en cada ciclo de reloj. El diseño es más complicado de lo que parece porque debe de estar preparado con características que incluyen dejar pasar ciertos datos. En general, se necesita "bypassing" para dejar pasar datos a través de un módulo porque son necesarios en el siguiente. Este mecanismo es encontrado con frecuencia en diseños basados en "pipeline". La fig. 2 muestra cómo las instrucciones han sido ejecutadas dentro de la estructura del "pipeline".

Time	Fetch	Predecode	Decode	Execute
1	LOAD #FA, reg0			
2	LOAD #1, reg1	LOAD #FA, reg0		
3	ADD reg0, reg1, reg13	LOAD #1, reg1	LOAD #FA, reg0	
4	READ reg13	ADD reg0, reg1, reg13	LOAD #1, reg1	LOAD #FA, reg0
5		READ reg13	ADD reg0, reg1, reg13	LOAD #1, reg1
6			READ reg13	ADD reg0, reg1, reg13
7				READ reg13

Fig. 2. Ejecución de instrucciones en un "pipeline".

4) Definición arquitectural del microprocesador
Internamente el microprocesador está dividido en cinco bloques funcionales. Las diferentes funciones de este diseño han sido tomadas en cuenta para realizar la partición de bloques. Los bloques obtenidos son: módulo de instrucciones ("instruction memory"), módulo predecodificador ("predecode"), módulo codificador ("decode"), banco de registros ("register file") y módulo de ejecución ("execute"). Cada uno de los módulos descritos tiene diferente funcionalidad. La fig. 3 muestra la interconexión interna del microprocesador.

En la fig. 3 tres señales son de vital importancia y en ellas centraremos nuestra atención. En primer término está la señal de reloj, la cual es una entrada en cada uno de los módulos internos del microprocesador. En segundo término esta la señal "*Flush*". Esta señal es puesta a nivel 1 lógico cuando alguna instrucción produce un salto en la ejecución del programa. Cuando esto se lleva a cabo, todas las instrucciones que están en los módulos funcionales deben ser desechadas. Esto podrá permitir que las nuevas instrucciones (a donde el salto ha ocurrido) del módulo de instrucciones sean pasadas a los demás módulos. En tercera instancia mencionaremos la señal "Jump", la cual es una salida del módulo de ejecución y que le indica al módulo de instrucciones que un salto se ha producido y que éste deberá enviar nuevas instrucciones a partir de la dirección de salto del programa.

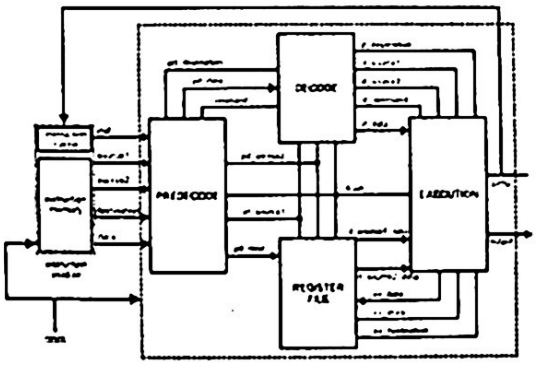

Fig. 3. Bloques internos básicos del microprocesador

III. DISEÑO DE CONTROLADORES PERIFERICOS

A. Interfaz Periférica Programable (PPI)
Este controlador se diseño en base a la estructura del PPI comercial 8255 [6], el cual se compone de tres puertos (A,B,C) divididos en dos grupos debido a la fragmentación del puerto C, lo que nos brinda una mayor versatilidad en cuanto a configuración. La fig. 4 muestra la arquitectura interna de la interfaz periférica programable. El grupo A se compone por el puerto A y la parte alta del puerto C (PC7 – PC4), mientras que el grupo B se integra por el puerto B y la parte baja del puerto C (PC3 – PC0). La palabra de control mantiene el formato original del 8255 [6].

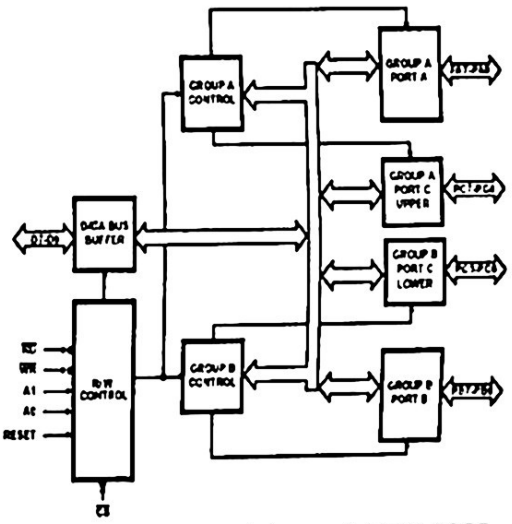

Fig. 4. Arquitectura interna del PPI 8255.

Operaciones básicas

A1	A0	RD	WR	CS	
0	0	0	1	0	Port A → Data Bus
0	1	0	1	0	Port B → Data Bus
1	0	0	1	0	Port C → Data Bus
1	1	0	1	0	Cntrl Word → Data Bus
0	0	1	0	0	Data Bus → Port A

0	1	1	0	0	Data Bus → Port B
1	0	1	0	0	Data Bus → Port C
1	1	1	0	0	Data Bus → Cntrl Word
X	X	X	X	1	Data_Bus → Tri-state
X	X	1	1	0	Data_Bus → Tri-state

B. Controlador para LCD

El siguiente controlador se diseñó como interfaz entre un microprocesador y una pantalla de LCD modelo FE0502 del fabricante AND[8]. Cada segmento se compone de dos conexiones: FP y BP, donde este último es común a todos los segmentos de la pantalla. Para que el segmento se encienda se requiere que entre FP y BP exista un voltaje de CA de preferencia de 5 Vrms y 32 Hz, sin componente de CD. Ya que los dispositivos lógicos sólo entregan señales '1' ó '0', es necesario hacer un arreglo en el que se pueda entregar corriente alterna; el más comúnmente utilizado es con compuerta XOR [8].

En este arreglo, la señal lógica 'Control' requerirá un '1' cuando se quiera encender el segmento, y un '0' cuando se quiera mantener apagado. Entonces, se requerirá un arreglo por cada segmento individual de la pantalla.

Para desarrollar la interfaz propiamente, se tendrán dos partes principales como se muestra en la Fig. 5. La primera es el arreglo de compuertas XOR y la segunda será un conjunto de registros en los cuáles el microprocesador pueda escribir usando las señales del bus de datos, el bus de direcciones y WR. Estos registros guardarán los valores que se conectarán con las señales 'Control' que encienden o apagan los segmentos.

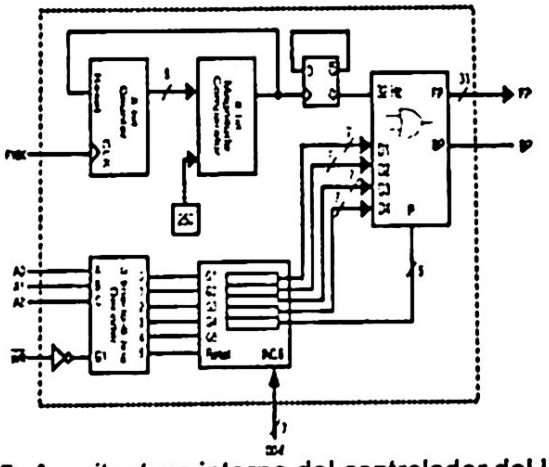

Fig. 5. Arquitectura interna del controlador del LCD.

Para la primera parte también es necesario implementar el oscilador de 32 Hz, lo que se logra generando un divisor de frecuencia con un contador para lograr el submúltiplo 32 Hz. A su vez, la segunda parte requerirá de un decodificador de dirección habilitado por la señal WR, y de un conjunto de cinco celdas del tipo "latch" con siete bits cada una.

C. Controlador de teclado matricial.

Este controlador periférico se diseñó en base al circuito comercial M74C922, para codificar completamente un arreglo de 16 teclas de 1 polo 1 tiro [7]. El rastreo de la tecla presionada es activado por un oscilador externo cuya frecuencia máxima deberá ser menor a 10 KHz. El decodificador también cuenta con resistencias internas de "*Pull-up*". La eliminación de rebotes al presionar la tecla es realizada digitalmente al interior del componente.). La fig. 6 muestra el diagrama de bloques interno del controlador de teclado.

1) Teoría de operación.

El controlador de teclado es conectado a una matriz de 4 filas por cuatro columnas, en total de 16 teclas de un polo 1 tiro. Cuando ninguna tecla es presionada, las filas de entrada (Yn) están puestas permanentemente en un valor de 1 lógico por medio de "*pull-up*" internos y las columnas de salida (Xn) activan un cero lógico secuencialmente. La velocidad de rastreo del teclado es controlada por la entrada de reloj (CLK), la cual activa a un contador de dos bits y a un decodificador 2 a 4. Cuando una tecla es presionada y su columna correspondiente exhibe un cero lógico, una de las filas de entrada (Yn) será forzada a cero lógico, además, un circuito interno deshabilitará al contador con lo cual el cero en esa misma columna será mantenido. El código de la tecla presionada que será enviado a las salidas del componente (DATA) será formado por el valor congelado del contador y de un codificador 4 a 2 de las filas de entrada y también será puesta a uno lógico la señal que indica que un dato esta disponible (DA).

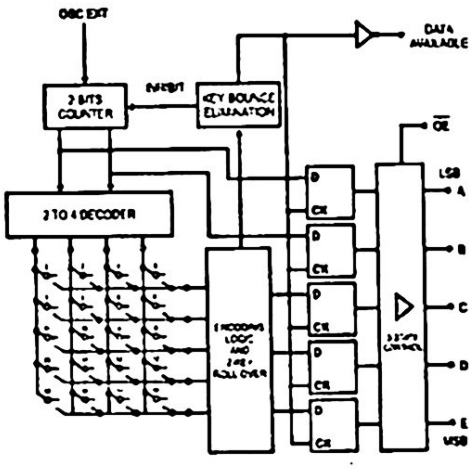

Fig. 6. Arquitectura del decodificador de teclado matricial.

D. UART

A continuación se enuncia el funcionamiento de la UART así como la descripción sus etapas. Su diseño esta basado en la arquitectura de la UART del AVR1200 [10] y su diagrama de bloques se muestra en la fig. 7.

1) Generación de Baudios

La función de la primera etapa es convertir la frecuencia de reloj que le es suministrada a la frecuencia de transmisión y recepción de datos. Para encontrar la frecuencia de comunicación la frecuencia del reloj se divide entre 16 veces un valor que es colocado en el registro de baudios de configuración.

2) Transmisión y recepción de datos

Une vez que ya se tiene la etapa de generación de baudios, para enviar un dato por la línea de transmisión lo único que se tiene que realizar es escribir un dato en el registro destinado para ello. Inmediatamente después de que se almacene un dato en el registro, comenzará la transmisión del dato, el formato de transmisión es: bit de inicio, byte a transmitir, bit de paridad y finalmente bit de paro. No se debe de enviar un dato antes de que se haya terminado de transmitir el primero por lo cual el controlador tiene una bandera que nos indica el fin de transmisión. Para la recepción de datos, se hace un muestreo a la mitad de cada bit durante la transmisión. Este módulo de la UART también cuenta con una bandera que indica la recepción de un nuevo dato, esta bandera se pone automáticamente en cero cuando se lee el registro de recepción de información.

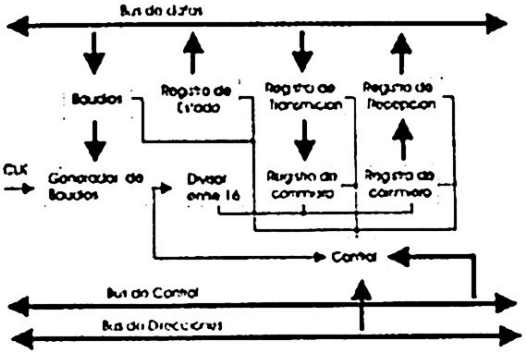

Fig. 7. Diagrama de bloques de la UART.

IV. RESULTADOS

Enseguida se presentan los resultados de síntesis hardware realizadas en Xilinx Foundation 4.1 para una FPGA VIRTEX100E-PQ240. La solución con microprocesador externo y controladores juntos en un solo chip es viable, debido al bajo costo de implementación. En la tabla 2 se presentan los resultados. La frecuencia máxima de operación para el diseño es de 65 Mhz y el área de chip ocupada es de 13%.

	SLICES	F-Fs	LUTs	IOBs
TECLADO	15	27	18	8
PPI	51	40	101	24
LCD	48	43	48	29
UART	55	70	49	2
TOTAL	157	172	199	85

TABLA 2. Controladores en un chip.

Si optamos por la solución Soc obtenemos los resultados que se muestran en la tabla 3. La frecuencia máxima de operación es de 40 Mhz y el área de chip ocupada es de 67%

	SLICES	F-Fs	LUTs	IOBs
CONTRO-LADORES	157	172	199	85
RISC	572	413	827	5
SoC	729	585	1026	90

TABLA 3. Implementación en un chip

Para establecer una comparativa más real se realizó la síntesis en la familia Spartan-II de Xilinx encontrándose que la solución µP+FPGA requiere de una FPGA XC2S30-TQ144 con un costo de 15 USD más 3.5 USD del µP, mientras que la solución SoC requiere de una FPGA XC2S100 con un costo de 24 USD.

V. CONCLUSIONES

Con base a los resultados obtenidos podemos deducir que la alternativa SoC implica el diseño de microprocesadores y periféricos lo que se refleja directamente en una mayor necesidad de área dentro de la FPGA y menor velocidad de procesamiento. Además de un mayor tiempo de desarrollo y depuración de errores del diseño.

La solución µP+FPGA demanda FPGAs de menor capacidad y menor precio por lo que desde el punto de vista económico es más viable para volúmenes bajos de producción en comparación con la opción SoC la cual requiere de FPGAs de mucha mayor densidad de puertas y generalmente mayor costo.

REFERENCIAS

[1] R.Y Wang; A. Krishnamurthy; R.P. Martin and others, Modeling Communication pipeline Latency. SIGMETRICS'98. pp.22-32, 1998.

[2] Jan M. Rabaey, Miodrag Potkonjak and others, Challenges and Opportunities en Broadband and Wireless Communication Designs, ICCAD'2000, pp.76-82, 2000.

[3] Foonk L., Lee; "VHDL coding and logic Synthesis with Synopsys", Edit. Academic Press, 2000.

[4] Hennessy, John and Patterson, David, "Computer Architecture: A Quantitative Approach", Edit. Morgan Kaufmann Publication, 2000.

[5] Peatman, John, Design with microcontrollers, McGraw Hill, , pp.320-341, 1988

[6] http://www.intel.com/design/periphrl/datashts/

[7] http://fairchildsemi.com/pf/MM/MM74C922.html

[8] http://www.purdyelectronics.com/ANDDisplays/paneldis.cfm

[9] Gajski, Daniel, Specification and Design of Embedded Systems, McGraw Hill, , pp.320-341, 1988

[10] página de Internet de AVR1200: http://www.atmel.com/dyn/products/product_card.asp?part_id=1992

[11] http://direct.xilinx.com/bvdocs/publications/ds022-2.pdf