Módulo de interfaz para cuatro lámparas de siete - segmentos en FPGA

Danelia Matos Molina1, Alexander Suárez Léon2, Roger Rivero Labrada3, Rubén D. López Noa4, Berta Pallerols Mir5

1 Centro Provincial de Electromedicina Santiago de Cuba

2 Universidad de Oriente, aasl@fie.uo.edu.cu

3 Universidad de Oriente, roger@fie.uo.edu.cu

4 Universidad de Oriente, lnoa@fie.uo.edu.cu

5 Universidad de Oriente, bertapm@fie.uo.edu.cu

RESUMEN

El presente artículo expone el diseño, síntesis e implementación sobre FPGA de un circuito digital para el manejo mediante visualización multiplexada de cuatro lámparas de siete segmentos. Se describe el principio de funcionamiento y se detalla el diseño de los bloques funcionales principales que componen el circuito. Se especifican los recursos asignados en la implementación para un FPGA específico y se analiza la respuesta en frecuencia del diseño. Finalmente se muestran y discuten los resultados de una simulación post implementación y se valora la utilidad y posibles aplicaciones del dispositivo.

Palabras claves: FPGA, VHDL, lámparas 7 segmentos.

ABSTRACT

In this paper is showed the design, synthesis and implementation of a multiplexed visualization controller circuit for four seven segments display lamps on FPGA. The functional principle is described and the designs of the functional blocks that composite the device are detailed. The assigned resources are stipulate for specific FPGA and the frequency response of the design is analyzed. In the end the results of a post place and route simulation model of the circuit are showed and discussed. The usefulness of the design and potential applications are argued too.

KeyWords: FPGA, VHDL, 7 - segment displays.

Interface module for four seven - segments display lamps on FPGA.

INTRODUCCIÓN

La tarjeta de desarrollo Spartan-3 Starter Kit cuenta con cuatro lámparas LED de 7 segmentos, controladas por salidas configurables del FPGA como se muestra en la Figura 1. Cada dígito comparte ocho conexiones de control comunes para iluminar los segmentos LED independientemente. Cada caracter individual incorpora además una entrada de control de ánodo propia. [1].

El número de patilla del FPGA asociado a cada señal de control aparece entre paréntesis en la figura. Para iluminar un caracter se debe colocar un nivel bajo en las patillas correspondientes a los segmentos que deben encenderse al tiempo que la señal de control de ánodo correspondiente a la lámpara específica debe ser igualmente puesta en nivel bajo.

La figura muestra el ejemplo para colocar el número “2” en la lámpara más a la izquierda, obsérvese como los segmentos a, b, g, e y d que deben encenderse tienen un nivel bajo aplicado mientras el resto de los segmentos (f, c y dp) están apagados y tienen un nivel alto.

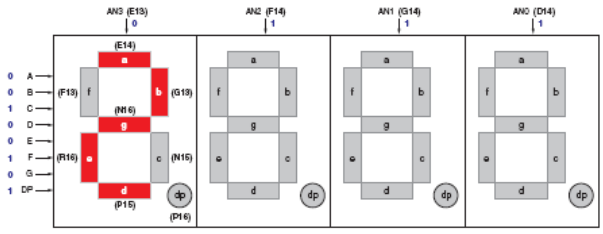

Figura 1

Conexión de las lámparas y el FPGA.

La patilla de control de ánodo de la lámpara que está más a la izquierda se encuentra en nivel bajo (encendido) mientras el resto está en nivel alto (apagado).

Este sistema exige entonces que las señales de control de encendido de los LED sean multiplexadas en tiempo; lo cual no es más que encender cada lámpara en pequeños intervalos de tiempo consecutivos activando cada vez su entrada de control de ánodo respectiva y colocando la información a visualizar en las patillas de segmentos, la Figura 2 muestra este proceso que debe repetirse a una frecuencia determinada.

La frecuencia debe elegirse de manera que debido a la persistencia de la imagen en el cerebro humano todas las lámparas parezcan encendidas al mismo tiempo, un buen rango, habitual en los displays de computadoras y televisores es de 60 – 100 Hz.

Este tipo de técnica reduce de manera significativa el número de patillas a utilizar para controlar las lámparas pero implica la realización de un circuito para controlar la visualización continua de los datos.

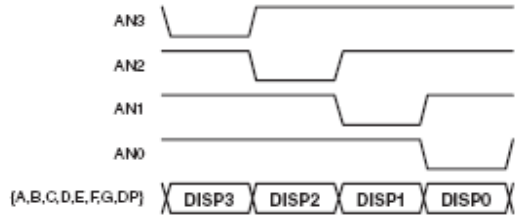

Figura 2

Carta de tiempo de visualización.

En el presente documento se detalla el diseño de un circuito digital que permita operar las cuatro lámparas según el diagrama de la Figura 2.

El objetivo de este documento es mostrar el diseño e implementación en FPGA de una sencilla unidad de comunicación serie asincrónica para su uso con procesadores empotrados.

METODOLOGÍA

El principio de funcionamiento del diseño está dado en tres (3) aspectos fundamentales:

1. El diseño debe contener un área de almacenamiento que permita guardar en algún formato el dato que se va a visualizar en cada lámpara.
2. Debe funcionar acorde a la carta de tiempo mostrada en la Figura 1.
3. Debe contener un circuito conversor de código, del formato utilizado a siete segmentos.

Analizando cada uno de estos puntos para el primer caso (punto 1), apunta a la implementación de una pequeña RAM o un banco de registros cuyas tallas estarán en dependencia del formato elegido para el dato a visualizar.

Para el caso tres (3) se trata de un circuito combinacional que convierta a código siete segmentos a partir del formato utilizado. Por ejemplo el circuito SSI estándar 7447 realiza esta función de BCD a siete segmentos.

En el caso del punto 2, trata el tema del comportamiento funcional y temporal del circuito propuesto, tema que tiene mucho que ver con la frecuencia de refrescamiento de las lámparas y la cantidad de estas.

Para refrescar cada lámpara es necesaria una frecuencia de actualización que como ya se vio puede considerarse aceptable entre 60 y 100 Hz (60 Hz es el estándar VESA de refrescamiento vertical en monitores con resolución 640x480). Si se tiene en cuenta que se trata de cuatro (4) lámparas el rango de repetición de las cuatro señales de control es entonces entre 240 y 400 Hz.

Estas frecuencias pueden considerarse bajas si se toma en consideración que el oscilador a cristal externo provee una frecuencia de 50 MHz la cual puede ser dividida internamente con el empleo de un administrador de reloj digital (DCM). [2].

Los DCM permiten dividir la frecuencia de entrada en múltiples factores configurables por el usuario; para el caso en particular se ha elegido un factor de 10 o lo que es lo mismo una frecuencia del reloj para el diseño de 5 MHz.

Como se aprecia el divisor del DCM no es suficiente para obtener frecuencias como las que necesita el diseño por tanto otro aspecto a tener en cuenta es el uso de divisores de frecuencia internos realizados normalmente a base de contadores.

Elementos generales

El módulo de interfaz es un circuito, ver Figura 3, con un total de 24 patillas, 12 entradas y 12 salidas. Las entradas corresponden a reloj (CLK), reinicio asincrónico (RST), selección de chip (CS), habilitación de escritura (WE), selección de la lámpara (A[1:0]) y dato a visualizar en la lámpara (D[5:0]).

El formato en que se representan estos datos se explica más adelante. Por su parte las salidas corresponden al código zero – hot de habilitación de la lámpara (AN[3:0]), los siete segmentos habituales (OS[6:0]) y el punto que es un segmento adicional, DP.

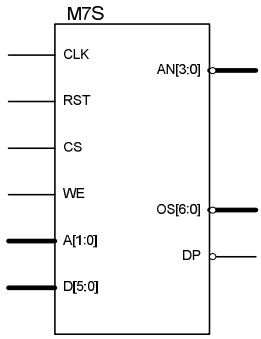

Figura 3

Diagrama de entradas/salidas del módulo.

Internamente el circuito está constituido por cuatro (4) bloques funcionales independientes, ver Figura 4.

- Máquina de estado finito. (FSM).
- Banco de registros. (RB).
- Memoria ROM de 16x7. (ROM).
- Circuito de interfaz de salida. (OIC).

A continuación se profundiza en la estructura interna y funcionalidad de estos elementos.

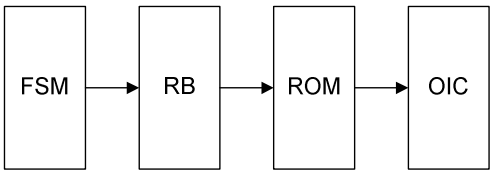

Figura 4

Diagrama en bloques de la estructura interna del módulo.

Máquina de estado finito (FSM)

Para poder implementar el aspecto 2, del principio de funcionamiento es necesario que exista un elemento (bloque) dentro del circuito que se encargue de secuenciar los datos (que se encuentran almacenados según el punto 1) a visualizar en las lámparas. Para ello se ha diseñado una máquina de estado finito (FSM) cuyo diagrama de entrada/salida se muestra en la Figura 5. [3], [4].

Desde el punto de vista de las entradas y salidas del circuito general, la FSM recibe las señales de entrada de reloj, (CLK) y reinicio asincrónico, (RST). Y tiene como salida la señal de habilitación de lámpara, (AN[3:0]); cuenta además con una señal interna que va al banco de registros para seleccionar el dato que se visualizará, (AD[1:0]).

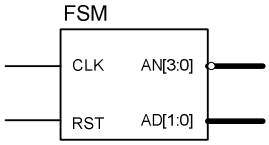

Figura 5

Diagrama de entradas/salidas de la FSM.

De manera general el modo de funcionamiento de la FSM consiste en:

- Seleccionar un dato a visualizar del banco de registros y activar el ánodo de la lámpara en la cual se visualizará el dato seleccionado.
- Entrar en un estado de espera mientras un contador no sea igual a 0.
- Obtener el próximo valor a visualizar.

Esta secuencia se repite para cada una de las lámparas hasta que se active por alguna razón la señal de reinicio asincrónico.

La Figura 6 muestra el diagrama de estados de la FSM, que ha sido diseñada con la herramienta StateCAD® de Xilinx®, contiene cuatro (4) estados, el de reinicio (**s_rst**) y tres (3) adicionales que ejecutan las funciones detalladas más arriba.

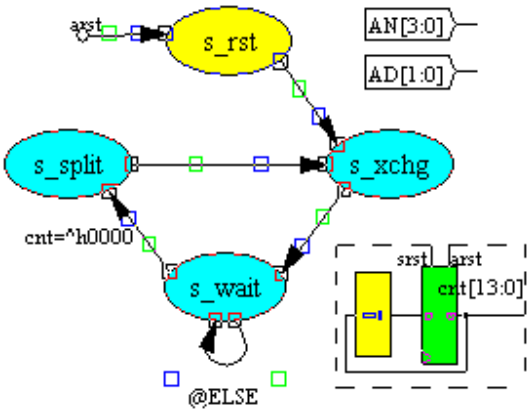

Figura 6

Diagrama de estados de la FSM.

La salida AN[3:0] se implementa con un registro de desplazamiento de 4 bits a la derecha, en el estado **s_xchg**, mientras que la salida AD[1:0], se implementa en este mismo estado con un contador binario descendente de dos (2) bits. Igualmente en este estado la señal **srst**, es puesta a nivel bajo, lo que equivale a disparar el contador descendente **cnt[13:0]**.

El estado de espera (**s_wait**) aguarda a que el contador **cnt[13:0]** alcance el valor cero, para obtener el número de bits de este contador se procede de la manera siguiente:

Asumiendo que se tiene una frecuencia del reloj interna de 5 MHz, para dividir correctamente la frecuencia y obtener la tasa de refrescamiento esperada, el conteo de pulsos de 5 MHz debe cumplir la relación (1).

$$\frac{5\,\text{MHz}}{400\,\text{Hz}} \le c \le \frac{5\,\text{MHz}}{240\,\text{Hz}} \qquad (1)$$

Para calcular el número de bits de semejante contador se tiene entonces la relación (2).

$$\log_2\left|\frac{5\,\text{MHz}}{400\,\text{Hz}}\right| \le n \le \log_2\left|\frac{5\,\text{MHz}}{240\,\text{Hz}}\right| \qquad (2)$$

Por tanto el rango

$$13.6 \le n \le 14.34 \quad \Rightarrow \quad n = 14 \qquad (3)$$

Lo que conduce a la conclusión de que el contador debe tener 14 bits como justamente implementa **cnt[13:0]**.

Finalmente el estado **s_split**, deja todo listo para la próxima secuencia de refrescamiento, para ello detiene el

contador (**srst = 1**) y determina el próximo valor de la salida AN[3:0] a través de la ecuación (4).

```
din = AD0 or AD1                    (4)
```

Sin olvidar que la ecuación del registro de desplazamiento en el estado **s_xchg** es:

```
AN = din:AN3…AN1                    (5)
```

Banco de registros (RB)

El banco de registros (RB), ver Figura 7, lleva a cabo las exigencias del punto 1, es decir implementa el área necesaria para el almacenamiento de los datos a visualizar. Las patillas de entrada corresponden al reloj, (CLK), habilitación de circuito, (CS), habilitación de escritura, (WE), dirección de lectura, (RA[1:0]), dirección de escritura (WA[1:0]) y datos de entrada (DI[5:0]). Las señal de salida es DO[5:0]. [5].

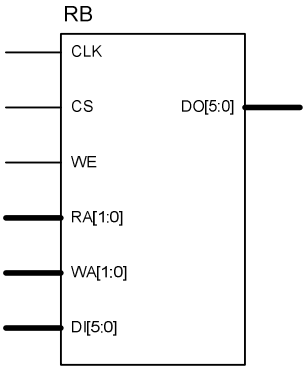

Figura 7

Diagrama de entradas y salidas del banco de registros.

Un banco de registros (RB) es intuitivamente un área de almacenamiento direccionable elemento a elemento tanto para lectura como para escritura, con la particularidad de mantener los canales de lectura y escritura separados de manera que se pueda escribir y leer de manera independiente en todo momento. Una manera eficaz de imaginar un banco de registros es considerar que se trata de una RAM que mantiene dos buses de direcciones separados uno para la escritura de datos y otro para la lectura de estos, además de tener un bus de datos de entrada y un bus de datos de salida independientes, donde el bus de datos de entrada son los datos que se desean escribir en la dirección señalada por el bus de direcciones de escritura y el bus de datos de salida que contiene el valor de la posición de memoria señalada por el bus de direcciones de lectura. Esta manera de ver un banco de registros es útil pues da pie a su implementación en lenguajes de descripción de hardware (HDL), como se muestra en el siguiente código.

```
process (CLK, CS)
begin
  if CS = '1' then
    if (CLK'EVENT and CLK = '1') then
      if (WE = '0') then
        RAM(conv_integer(WA)) <= DI;
      end if;
    end if;
  end if;
end process;

DO <= RAM(conv_integer(RA));
```

La cantidad de registros en el banco es igual a 2^N donde N es el número de bits de los buses de direcciones (escritura y lectura).

El tamaño de los buses de datos responden al formato utilizado para almacenar la información a visualizar en este caso es de 6 bits correspondiendo al formato que se muestra en la Figura 8.

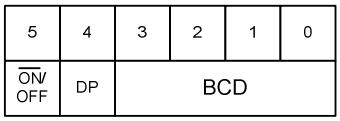

Figura 8

Formato de los datos a visualizar.

El formato es muy simple, el bit 5 determina si la lámpara estará encendida o apagada, (0 encendida, 1 apagada); el bit 4 es el bit del segmento punto en este caso es al revés (0, apagado, 1 encendido). Finalmente los bits del 3 al 0 corresponden al código BCD del dígito a visualizar.

Memoria ROM y circuito de interfaz de salida

Finalmente la memoria ROM implementa el último aspecto tratado en el principio de funcionamiento, es decir, el aspecto de la conversión de código. Funcionalmente la ROM no es para nada distinta al 7447 estándar. Para su implementación se han utilizado las primitivas ROM16x1 de Xilinx® en número de 7, una para cada segmento. La Figura 9 muestra el diagrama de entrada/salida.

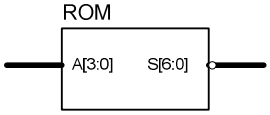

Figura 9

Memoria ROM 16x7.

El circuito de interfaz de salida (OIC) es un arreglo de puertas OR que permite la implementación de la función de apagado de las lámparas, para ello se apoya del bit 5 del formato utilizado y de la propiedad del OR:

```
x or 1 = 1                    (6)
```

La Figura 10 muestra el diagrama de entrada/salida y la estructura interna del circuito de interfaz de salida. Tener en cuenta que BIT4 = dp.

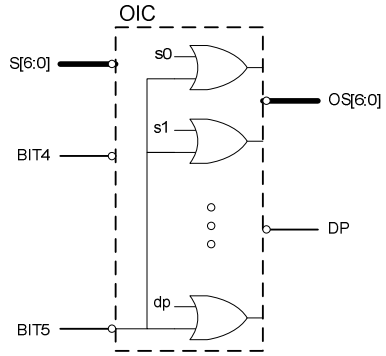

Figura 10

Circuito de interfaz de salida.

RESULTADOS

Las tablas de costo de implementación en el FPGA se muestran, Tabla I y Tabla II, en cada caso se muestran los costes de implementación de los bloques funcionales y finalmente para la estructura completa:

Tabla I: Tabla de costos de implementación XC3S100FT256

Elemento	S	%	SFF	%	4ILUT	%
FSM	27	0.35	31	0.20	47	0.31
RB	21	0.27	24	0.16	16	0.10
ROM	4	0.05	-	-	7	0.05
OIC	5	0.07	-	-	8	0.05
M7S	53	0.69	54	0.35	82	0.53

S: Slices

SFF: Slices Flip/Flops

4ILUT: 4 - Input LUTs

Tabla II: Tabla de costos de bloques de entrada/salida y señales de reloj XC3S100FT256

Elemento	IOBs	%	GCLK	%
FSM	5	2.9	1	12.5
RB	10	5.8	1	12.5
ROM	-	-	-	-
OIC	8	4.6	-	-
M7S	23	13.3	1	12.5

IOBs: Bloques de entrada/salida

GCLK: Buffers globales de reloj

Como se observa el consumo de recursos es inferior al 1% de los disponibles en el FPGA, lo que indica que se trata de un diseño que comparado al tamaño del dispositivo FPGA utilizado es pequeño.

Otro análisis a tener en cuenta es la respuesta en frecuencia del sistema. Desde el punto de vista de la caracterización temporal se trata de un sistema **clock to pad** los cuales pueden representarse como se muestra en la Figura 11.

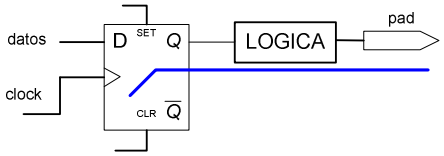

Figura 11

Clock to pad (C2P).

La línea azul representa el camino o retraso desde el flanco del reloj (clock) hasta la patilla de salida (pad). Independientemente a ello en la Tabla III se muestran los peores casos para los distintos tipos de retrasos y la Tabla IV muestra los caminos que introducen estos retrasos, obsérvese detenidamente como es precisamente **clock to pad** (C2P) el retraso más importante del sistema.

Tabla III: Retrasos en ns de la lógica del sistema.

C2S	C2P	P2S
5.014ns	9.590ns	4.400ns

Tabla IV: Peores caminos en cada caso.

C2S	C2P	P2S
cnt[0 -13]	A[0] - CLK	CLK – OS[6]

C2S: Clock to setup

C2P: Clock to pad

P2S: Pad to setup

La simulación **Post Place and Route** se muestra en dos etapas, la primera, ver Figura X compete a la

introducción de los valores a visualizar en el dispositivo y la segunda los ciclos de visualización, ver Figura 12.

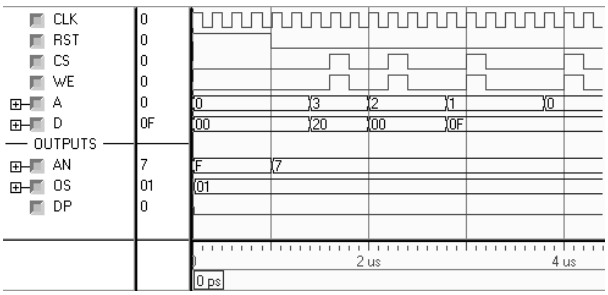

Figura 12

Introducción de valores en el módulo.

Se ha dividido de esta forma ya que la primera etapa se ejecuta relativamente rápido y una sola vez. Mientras la segunda se ejecuta autónoma y continuamente a frecuencias relativamente bajas.

Los datos deben ser introducidos por otro circuito que puede ser un microprocesador o microcontrolador (ver más abajo el acápite DISCUSIÓN).

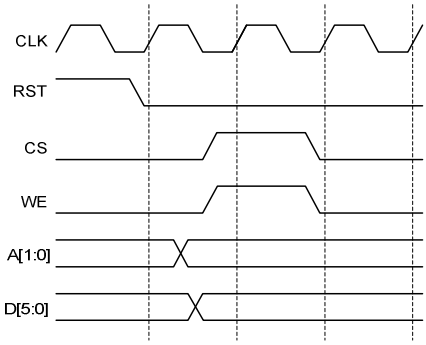

Figura 13

Diagrama de tiempo para escritura en el módulo.

Sin embargo, también puede ser cualquier dispositivo capaz de secuenciar la operación de escritura en el banco de registros que es en definitiva cumplir con el diagrama de tiempo de la Figura 13.

La Figura 14 muestra la salida de los datos en formato de 7 segmentos obsérvese como se repite el ciclo en menos de 15 ms. La salida de los datos corresponde a la visualización de la palabra O.F.F. en las lámparas 2 – 0, mientras que la lámpara número 3 debe permanecer apagada.

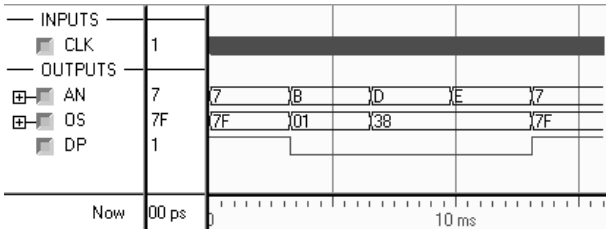

Figura 14

Funcionamiento del módulo a partir de los valores introducidos.

DISCUSIÓN

En el acápite anterior se mostraron los resultados desde la síntesis a la implementación, así como la simulación post mapeo y conexionado (**Post Place and Route**). A continuación se discuten las posibles ventajas y desventajas que pudiera presentar la utilización del módulo.

La desventaja fundamental del uso del módulo a entender de los autores es el hecho de que se trata de hardware sintetizado por tanto ocupará espacio físico dentro del FPGA, aunque mirando la tabla I se puede concluir que esta utilización de los recursos es bastante pequeña (0.69%) para el FPGA XC3S1000.

La otra desventaja es que pudiera ser deseable mayor control sobre el encendido de los LEDS particulares de cada lámpara, aspecto que no se tiene en cuenta en el diseño presentado, sin embargo, esta dificultad puede ser fácilmente erradicada agregando unos pocos elementos adicionales.

Otro argumento en contra es que si se va utilizar con procesadores o microcontroladores empotrados el trabajo de sincronización y manejo de los datos y de las señales de control puede hacerse por software eliminando la necesidad del módulo. La Figura 15 muestra esta situación tomando como ejemplo un sistema que incorpora el procesador **Silmaril 1.1**, desarrollado por los autores de este trabajo, obsérvese que siempre se necesitará una interfaz mínima (IM) para evitar que el tráfico en los buses afecte de manera involuntaria las lámparas. [6].

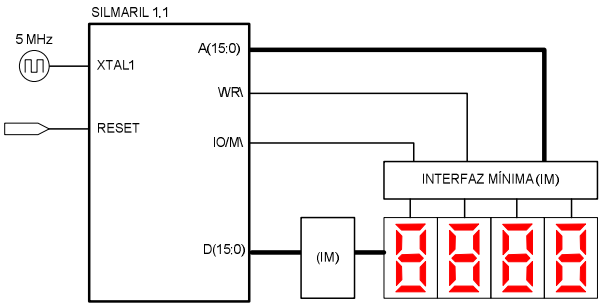

Figura 15

Sistema con microprocesador Silmaril 1.1 e interfaz mínima.

Sin embargo, es este último argumento precisamente el que puede convertirse en el principal para utilizar el módulo por dos razones fundamentales:

- Si se va utilizar un procesador se liberaría al programador de la tarea adicional de tener que programar la visualización (incluso de conocer el código 7 segmentos).
- El hecho de tener un diagrama de tiempo sencillo y elementos de almacenamiento hace posible su empleo con circuitos simples.

En la Figura 16 se muestra la misma situación de la Figura 15 incluyendo el módulo de visualización.

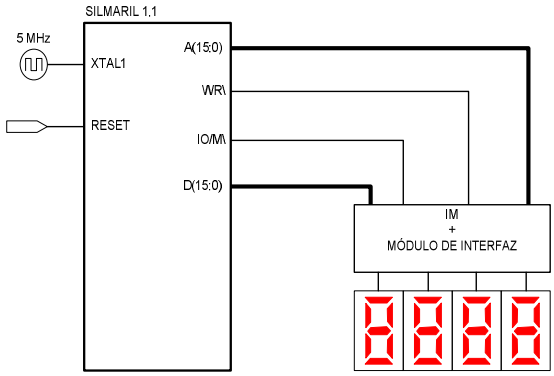

Figura 16

Sistema con microprocesador Silmaril 1.1 y módulo de interfaz.

CONCLUSIONES

El poco costo en recursos del módulo diseñado, los resultados satisfactorios de la simulación unido a la facilidad de implementación son a modo de ver de los autores las principales virtudes del diseño propuesto por lo que se puede concluir que se trata de una opción a valorar para sistemas que requieran de visualización sencilla tal como es el caso de la visualización multiplexada en lámparas de 7 segmentos.

REFERENCIAS

1. **XILINX, INC:** "Xilinx UG130 Spartan-3 Starter Kit Board User Guide". 2004.
2. **XILINX, INC:** "Xilinx Libraries Guide". 2004.
3. **WAKERLY, J. F:** "Digital Design. Principles and Practices". 4ed. New Jersey: Pearson Perentice Hall, pp. 553 – 587. 2006.
4. **XILINX, INC:** "StateCAD® Release 6.2i Help". 2004.
5. **ASHENDEN, P. J:** "The VHDL Cookbook". 1990.
6. **SUÁREZ LEÓN, A. A:** "Diseño del microprocesador Silmaril 1.1 para sistemas empotrados en FPGA" en *Conferencia Internacional FIE 2008*. 2008

AUTORES

Danelia Matos Molina, Ing. en Automática, Profesor Instructor, Centro Provincial de Electromedicina, Santiago de Cuba, Dpto. Electro-óptica. Miembro del Grupo Científico de Electrónica Programable *Ni*.

Alexander A. Suárez León, Ing. en Automática, Instructor recién graduado, Universidad de Oriente. FIE, Dpto. Ingeniería Biomédica. aasl@fie.uo.edu.cu Miembro del Grupo Científico de Electrónica Programable *Ni*.

Roger E. Rivero Labrada, Ing. en Materiales y Componentes Electrónicos, Profesor Asistente, Master en Ciencias, Universidad de Oriente. FIE, Dpto. Ingeniería Biomédica. roger@fie.uo.edu.cu. Subdirector Grupo Científico de Electrónica Programable *Ni*. Jefe de Carrera Ingeniería Biomédica.

Rubén D. López Noa, Ing. en Automática, Profesor Auxiliar, Master en Ciencias. Universidad de Oriente FIE, Dpto. Ingeniería Biomédica. lnoa@fie.uo.edu.cu Director del Grupo Científico de Electrónica Programable *Ni*. Jefe de Departamento Ingeniería Biomédica.

Berta Pallerols Mir, Ing. en Telecomunicaciones, Profesora Auxiliar, Universidad de Oriente. FIE, Dpto. Ingeniería en Telecomunicaciones bertapm@fie.uo.edu.cu. Miembro del Grupo Científico de Electrónica Programable *Ni*. Jefa de Departamento Ingeniería en Telecomunicaciones.