Configuración de un display LCD para el mostrado de mensajes de texto mediante FPGA

Gustavo Zoireff
Laboratorio III – Instituto Balseiro
Viernes 6 de diciembre de 2013

Se programó en VHDL un código que, una vez cargado a la FPGA, permitió mostrar mensajes de texto en un display LCD. Los mensajes fueron codificados en ASCII, almacenados y posteriormente leídos desde una memoria de tipo FIFO. Se realizaron las simulaciones correspondientes para verificar el funcionamiento correcto de los módulos implementados. Además, es posible reutilizar tales módulos para futuras aplicaciones realizando simples modificaciones.

I. INTRODUCCIÓN

A. PmodCLS™ [1]

PmodCLS (figura 1) es el nombre comercial asignado al módulo del display LCD desarrollado por Digilent®. Este puede ser utilizado para mostrar información importante durante el desarrollo de un programa o como interfaz al usuario una vez finalizado un proyecto.

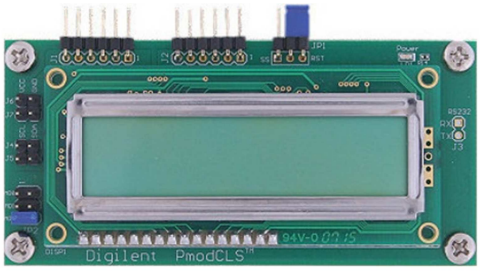

Figura 1. Módulo del display LCD PmodCLS.

El módulo puede usarse en proyectos usando la placa de desarrollo de FPGA de Digilent.
En la figura 2 puede apreciarse el diagrama de bloques del módulo, el cual está embebido en el microcontrolador ATmega48.
En este trabajo se propone un diseño que permite configurar el PmodCLS con una placa de desarrollo de FPGA.

B. Descripción del funcionamiento

La comunicación con la placa se establece mediante las conexiones serie UART, SPI o TWI. Los caracteres son mostrados en el display enviando información a través de los canales de comunicación. Los caracteres aparecen en la pantalla en la ubicación del cursor de la placa. La ubicación del cursor, la comunicación y otras instrucciones son configuradas mediante secuencias de escape.
En este proyecto se utilizó la comunicación SPI dado que ya fue previamente implementado por el fabricante.

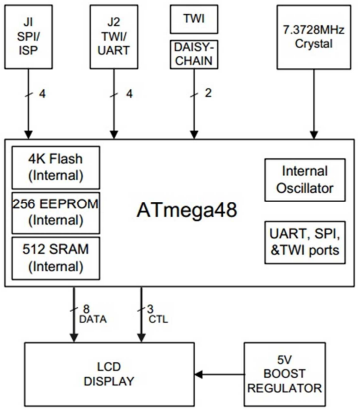

Figura 2. Diagrama de bloques del módulo PmodCLS.

C. Set de Instrucciones [2]

El PmodCLS es capaz de ejecutar numerosas instrucciones diferentes. Las instrucciones son enviadas mediante secuencias de escape. Una secuencia de escape comienza con el caracter ESC (en código ASCII 0x1B), seguido del corchete izquierdo '[', y luego por 0 o más parámetros separados por ';', finalizando con el caracter comando.
Debido a que el objetivo del proyecto era escribir un texto, la secuencia de escape utilizada fue

ESC [j ESC [0 ; 0 H TEXTOSUPERIOR ESC [1 ; 0 H TEXTOINFERIOR

En donde la secuencia "*X ; Y H*" indican que en la fila X (0 fila superior, 1 fila inferior) y columna Y del display se escribirá una letra codificada en ASCII en formato hexadecimal.
Es de importancia remarcar que el número de columna Y depende de si se está en la fila 0 o 1, pues en la fila 0 pueden escribirse 16 letras y en la 1 se pueden escribir 15. TEXTOSUPERIOR y TEXTOINFERIOR son los correspondientes textos a ingresar como un vector el cual contiene en sus elementos la letra a mostrar.

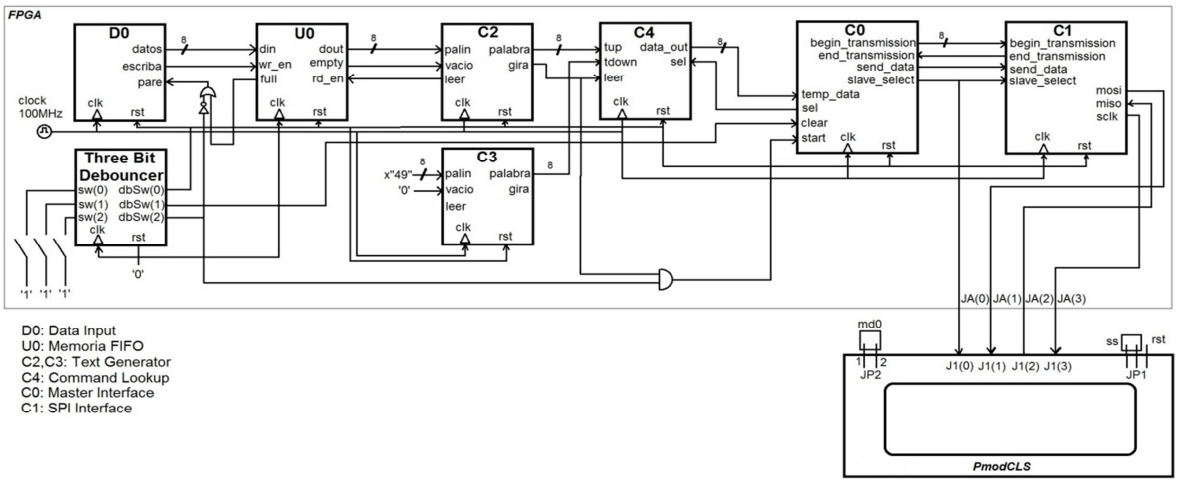

Figura 3. Diagrama de bloques general del proyecto.

II. DISEÑO e IMPLEMENTACIÓN

Utilizando los conceptos explicados anteriormente, se programó un software en VHDL [3] que una vez cargado a la placa mostraba un texto predefinido. En la figura 3 puede apreciarse el diagrama de bloques general del conjunto.

Se realizaron distintos módulos que cumplían una función específica, simplificando así el código principal, en el cuál se realizó la interconexión entre ellos.

El botón de RESET era *sw(0),* el de borrado de pantalla era *sw(1)* y el de encendido era *sw(2)*. La secuencia para el funcionamiento correcto es primero cerrar y abrir la *sw(0)*, luego repetir lo mismo con *sw(1)* y por último dejar *sw(2)* cerrada.

En estas condiciones, se borra la pantalla y comienza el flujo de información por el sistema. Los datos se generan en D0 y son guardados en la memoria FIFO U0. Se utilizaron C2 y C3 como generadores del texto superior e inferior, respectivamente (ambos módulos son idénticos). C2 leía los datos de U0 y C3 leía un valor constante de código ASCII x"49" (letra I). A su vez, C2 actualiza la palabra según un clock interno, y está vinculado a los demás módulos mediante la señal *gira*. Esta señal indica al módulo C4 en que momento leer el dato *palabra*.

En C4 se escribe la secuencia de escape y se envían los datos a C0.

C0 y C1 controlan la comunicación SPI y constantemente envían información hacia el display.

Notar que todos los módulos están sincronizados por el clock de la placa que son 100MHz. Además, existe

una serie de variables de control que también garantizan que el flujo de información esté sincronizado.

III. DESCRIPCIÓN DE BLOQUES

A. Data Input D0

Este bloque se encarga de generar una palabra y trasmitirla a la memoria FIFO U0. Sería el equivalente a una "caja negra" que entrega datos en código ASCII.

El módulo fue implementado con una máquina de 3 estados: IDLE, SEL y CUENTA y las variables i (índice), *esc* (bandera que indica que escriba en la memoria), *dat* (datos enviados), *P* (una palabra arbitraria, con *P(i)* una letra), y pare (una bandera de control externa que detiene el proceso). El esquema puede verse en la figura 4.

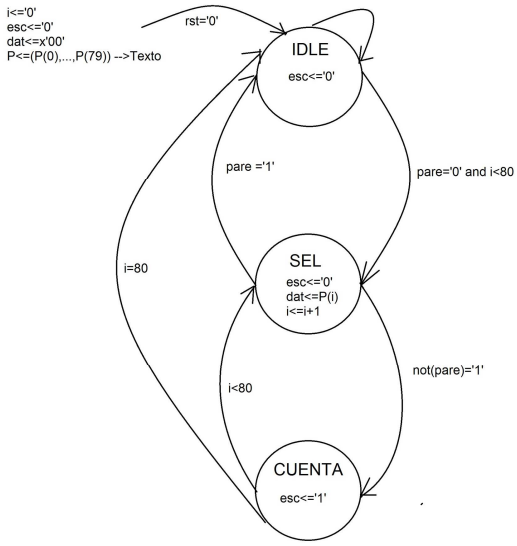

Figura 4. Diagrama de estados del módulo D0.

Cuando *reset = '1'*, se establecen los valores de *i*, *esc*,*dat* y *P* por defecto como se indica en la figura 4.

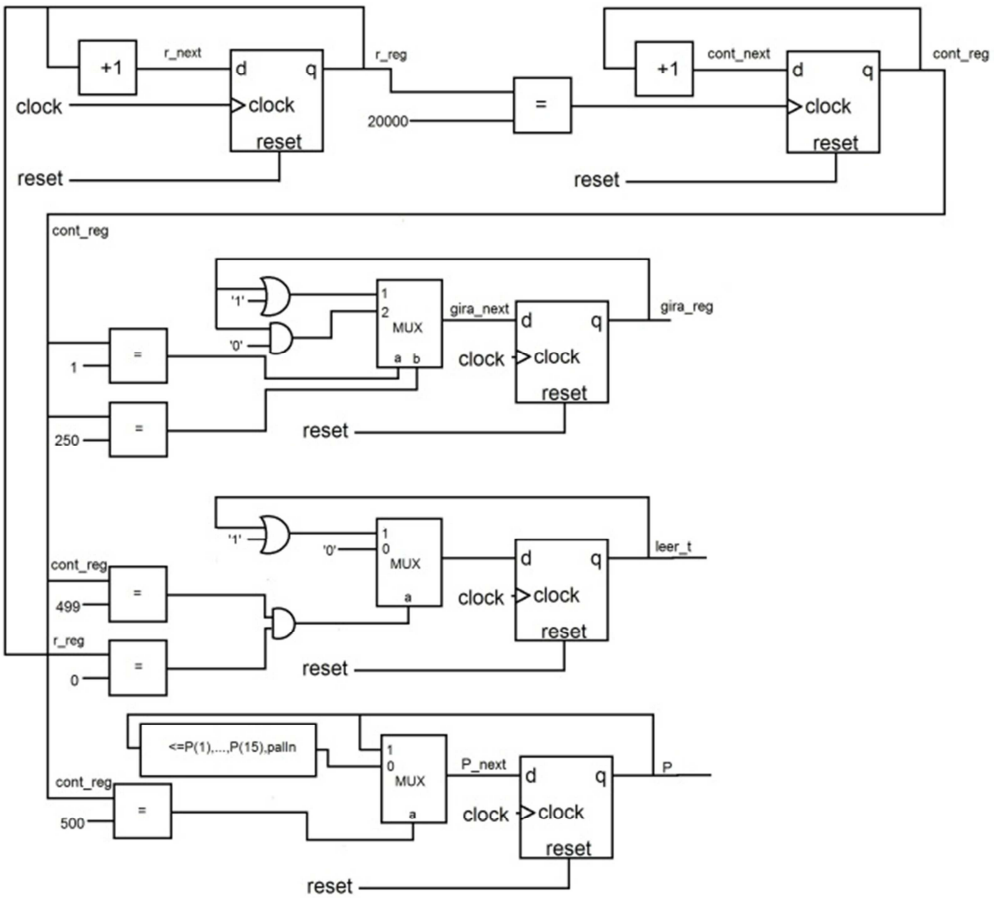

Figura 5. Diagrama de estados de Text Generator.

Luego, con *reset='0'*, se pasa al estado IDLE. En este estado se asigna *esc='0'* y se pasa al estado SEL, si se cumplen las condiciones indicadas en la figura 4. En SEL continúa *esc = '0'*, se asigna *dat* con *P(i)* y se incrementa i en 1 antes de pasar al estado CUENTA.
En CUENTA se envían los datos pues *esc = '1'*, y retorna a SEL si *i < 80*.
Cuando *i = 80* retorna a IDLE y se queda allí hasta el próximo *reset='1'*.

B. Memoria FIFO U0

Este módulo fue creado con *Core Generator* del programa Xilinx Designe Suite 12.4. Consta de una entrada y salida de datos que son manipulados según *wr_en* (controla la escritura de datos) y *rd_en* (controla la lectura de datos), y además están las banderas *full* y *empty* que indican si la memoria está llena o vacía.
Se generó una memoria de 8x512 bits. Como cada letra ocupa 1 byte, U0 es capaz de guardar textos de hasta 512 caracteres.

C. Text Generator C2-C3

En este bloque se leen datos provenientes de U0 según la señal de control *leer*. Se implementó utilizando máquinas de estado tipo Mealy, compuestas por latchs, comparadores y multiplexores según puede observarse en la figura 5.
Se genera un contador de pulsos de reloj llamado *r_reg*, el cual se introduce a un comparador y se genera otro clock a una frecuencia mucho menor a la de la placa (~5KHz).
Con esta nueva frecuencia, se inicializa otro contador *cont_reg*. Este contador sirve para poder mostrar los datos a una frecuencia tal que el usuario pueda ver el mensaje en la pantalla sin inconvenientes. Luego, se generan las señales de control *gira_reg*, *leer_t* y el mensaje *P*.
El mensaje *P* toma la palabra que llega de U0 y la coloca en *P(16)*. Como ésta se actualiza cada vez que *cont_reg = 500*, se tiene como resultado que el mensaje se va mostrando letra a letra y girando hacia la izquierda en la pantalla del display.

D. Command Lookup C4

Este bloque fue originalmente dado por el fabricante y modificado para el propósito de este trabajo. El diagrama de estados puede verse en al figura 6.
Cuando *reset = '1'*, el estado actual es IDLE. Cuando la señal externa *leer = '1'* (esta señal esta controlada por *gira_reg*) pasa al estado ESPERA.
En ESPERA se asigna a *command* las palabras *(cup, tup, cdown, tdown)* concatenadas, en donde *cup* y *cdown* contienen las secuencias de escape correspondientes a los textos superior en inferior *tup* y *tdown*, provenientes de C2 y C3.
Cuando *leer = '0'* se pasa al estado cuenta, en donde se detiene un determinado tiempo. Esto se hace para evitar errores de sincronización.
Finalmente, se retorna al estado IDLE y se repite el proceso.
Se envían los datos *command(sel)*, donde *sel* es la posición de la letra en cuestión y está comandada por C0 (Master Interface).

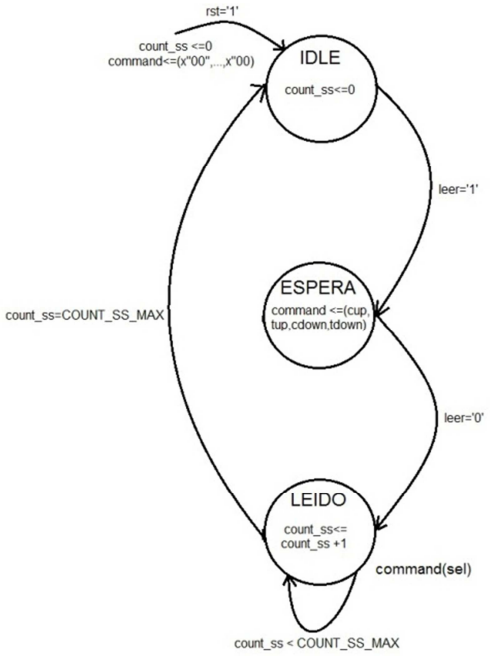

Figura 6. Diagrama de estados de Command Lookup

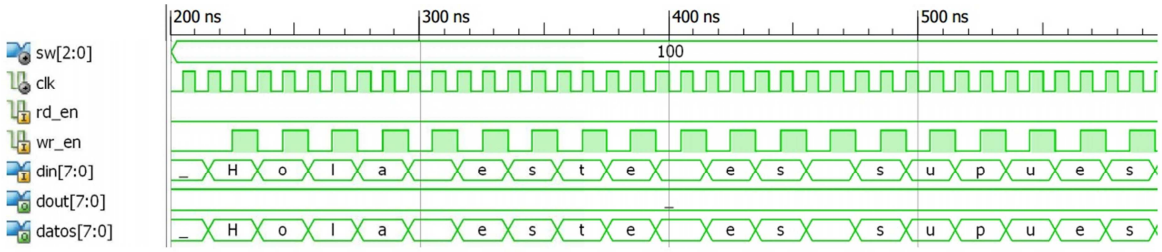

Figura 7. Resultados de la simulación de los módulos D0 y U0.

E. Master Interface C0 y SPI Interface C1

Ambos bloques constituyen el sistema de comunicación SPI con el cual se comunica con PmodCLS.
Los módulos fueron creados por el fabricante. En ellos se realizan la operación de borrado (*clear)* e inicio de la comunicación (*start)*, que está íntimamente ligado a la variable de control *gira_reg*.

IV. RESULTADOS Y DISCUSIÓN

Se realizó la simulación de los módulos D0 y U0 en conjunto para verificar que la lectura- escritura de datos sea posible. Los resultados pueden apreciarse en la figura 7.
Se observa que claramente que *datos* se guardan en *din* de la memoria FIFO U0 cuando wr_en está en 0, que es la secuencia generada en D0.
No fue posible simular la salida pues era necesario demasiado tiempo de simulación, pero pudo verificarse correctamente el funcionamiento del sistema completo una vez cargado el software a la placa.
Se pudo mostrar un mensaje generado en D0, el cual se observaba progresivamente en la pantalla de PmodCLS.
La ventaja de haber propuesto el esquema de la figura 3 es que es posible remplazar el módulo D0 Data Input por cualquier sistema que entregue datos codificados en ASCII de tal forma que *cualquier* mensaje, hasta 512 palabras, pueda mostrarse.
Además, es posible mostrar mensajes independientes en la fila superior e inferior del display debido a que los módulos C2 y C3 son independientes entre ellos.

V. CONCLUSIÓN

Mediante la utilización de una FPGA se logró configurar el display LCD PmodCLS para mostrar un mensaje arbitrario por su pantalla. Los resultados de la simulación fueron favorables y se observó el

comportamiento esperado para los módulos D0 Data Input y U0 Memoria FIFO. La implementación del sistema en VHDL permitió utilizar distintos módulos y ensamblar el conjunto para mostrar así un mensaje predeterminado, tanto en la fila superior como en la inferior de la pantalla.

Por último, podrían reutilizarse tales módulos y realizar modificaciones para poder enviar mensajes de cualquier procedencia, por ejemplo, desde una computadora hacia la placa.

---

[1] PmodCLS™ Serial LCD Display Module Reference Manual – Digilent® – December 14, 2007.
[2] PmodCLS Interface Reference Project – Digilent® – June 25, 2012.
[3] FPGA prototyping by VHDL examples – Pong P. Chu – 2008.