

Curso de Especialización en Sistemas Embebidos

Microarquitecturas y Softcores

12va Cohorte

2021

***Trabajo Práctico Final***

Alumno:

Cristian Trinidad

Historial de cambios

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Versión** | **Cambio** |
| 17/04/2021 | 1.0 | Versión Inicial |
|  |  |  |
|  |  |  |

# 

Índice

[1 Introducción 4](#__RefHeading___Toc3274_4072171326)

[1.1 Propósito 4](#__RefHeading___Toc3276_4072171326)

[1.2 Alcance 4](#__RefHeading___Toc3278_4072171326)

[2 Breve descripción del driver de display implementado en CLP. 5](#__RefHeading___Toc3280_4072171326)

[2.1 Interfaz de E/S del display 5](#__RefHeading___Toc3282_4072171326)

[2.2 Interfaz de E/S del módulo lcd\_controller 6](#__RefHeading___Toc3284_4072171326)

[3 Implementacion del wrapper 8](#__RefHeading___Toc3286_4072171326)

[3.1 Interfaz de E/S del wrapper 8](#__RefHeading___Toc3288_4072171326)

[3.2 Registros de lectura/escritura 9](#__RefHeading___Toc3290_4072171326)

[4 Simulaciones 11](#__RefHeading___Toc3292_4072171326)

[4.1 Testbench 11](#__RefHeading___Toc3294_4072171326)

[4.2 Operacion de escritura en el bus 12](#__RefHeading___Toc3296_4072171326)

[4.3 Operacion de lectura en el bus 13](#__RefHeading___Toc3298_4072171326)

[4.4 Operacion normal 14](#__RefHeading___Toc3300_4072171326)

[5 Implementación en Quartus 15](#__RefHeading___Toc3302_4072171326)

[5.1 Esquemático 15](#__RefHeading___Toc3304_4072171326)

[5.2 Pines de entrada/salida 16](#__RefHeading___Toc3306_4072171326)

[5.3 Recursos de la FPGA 18](#__RefHeading___Toc3308_4072171326)

[5.4 Firmware 18](#__RefHeading___Toc3310_4072171326)

# Introducción

# Propósito

En este trabajo se propone tomar el driver LCD para display alfanumérico con controlador Hitachi HD44780 implementado en CLP y comunicarlos con el softcore Nios II a travez del bus Avalon. Para tal fin, se va a utilizar el kit de desarrollo DE1-SoC rev C que contiene una FPGA de Altera Cyclone V modelo 5CSEMA5F31C6N.

# Alcance

Se implementará:

1. El wrapper necesario para establecer la comunicación con el microcontraldor.
2. La generacion de un IP a partir del wrapper y los archivos vhd diseñados en CLP.
3. Validacion del wrapper a traves de simulaciones utilizando ghdl.
4. La implementacion de rutinas basicas en C para el manejo del IP a travez del uC.

## 

# Breve descripción del driver de display implementado en CLP.

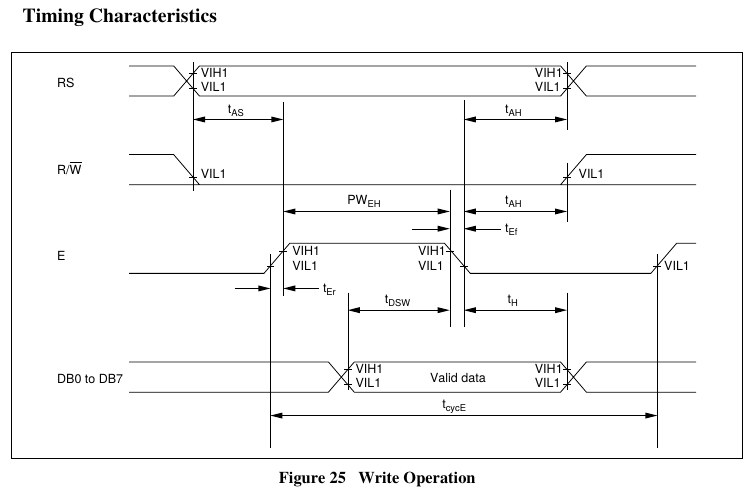
# Interfaz de E/S del display

|  |  |  |
| --- | --- | --- |
| **Pin** | **Símbolo** | **Función** |
| 1 | VSS | Power Ground |
| 2 | VDD | Power supply for logic circuit(+5V) |
| 3 | V0 | For LCD drive voltage (variable) |
| 4 | RS(C/D) | H: Display Data,  L: Display Instruction |
| 5 | R/W | H: Data Read (LCM to MPU) ;  L: Data Write (MPU to LCM) |
| 6 | EN | Enable signal.  Write mode (R/W = L) data of DB<0:7> is latched at the falling edge of E.  Read mode (R/W = H) DB<0:7> appears the reading data while E is at high level |
| 7-14 | DB0-DB7 | Data bus |
| 15 | A | Power for LED Backlight (+V) |
| 16 | K | Power for LED Backlight (Ground) |

El display cuenta con una interfaz paralela de 8-bits (DB0-7) la cual puede ser utilizada en formato 4-bits haciendo 2 escrituras o lecturas.

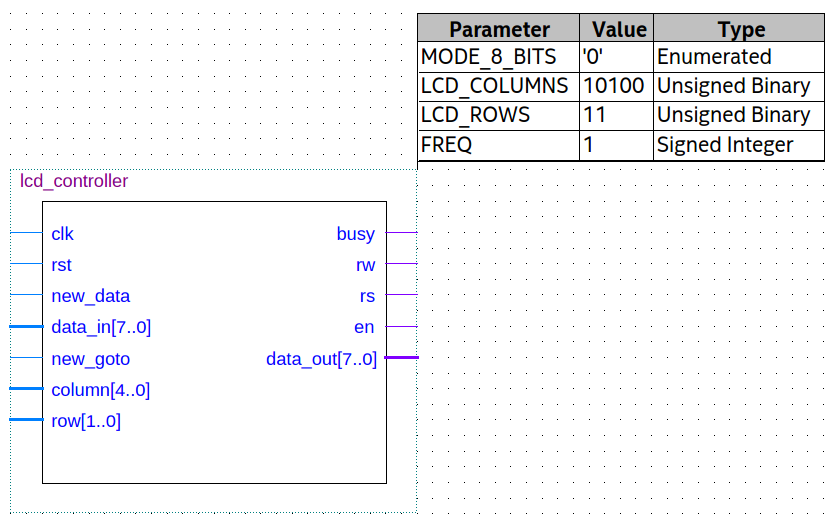
Para el manejo del display se deben comandar las I/O RS, R/W, EN y (DB0-7).

Ejemplo del manejo de las señales del display:



# Interfaz de E/S del módulo lcd\_controller

Para comandar las señales anteriores se implemento el siguiente contraldor en CLP:



entity lcd\_controller is

generic (

MODE\_8\_BITS : std\_logic := '1'; -- 8-bits or 4-bits

LCD\_COLUMNS : std\_logic\_vector(4 downto 0) := "10100"; -- 20

LCD\_ROWS : std\_logic\_vector(1 downto 0) := "11"; -- 4

FREQ : integer := 1 -- system clock frequency in MHz

);

port (

clk : in std\_logic; --system clock

rst : in std\_logic; --reset

new\_data : in std\_logic; --new data\_in valid

data\_in : in std\_logic\_vector(7 downto 0); --data

new\_goto : in std\_logic; --new column and row valid

column : in std\_logic\_vector(4 downto 0); -- characters in a row

row : in std\_logic\_vector(1 downto 0); -- row number

busy : out std\_logic; --lcd controller busy

rw, rs, en : out std\_logic; --read/write, setup/data, and enable for lcd

data\_out : out std\_logic\_vector(7 downto 0)); --data output to LCD

end;

**Genéricos/parámetros:**

MODE\_8\_BITS: se utiliza para configurar el modo de operación en 4 u 8-bits.

LCD\_COLUMNS: número de columnas/caracteres del display a utilizar.

LCD\_ROWS: número de filas del display a utilizar.

FREQ. Frecuencia de operación del bloque, se utiliza para calcular los tiempos de espera.

**Entradas:**

*clk*: señal de reloj del bloque.

*rst*: señal de reset.

*new\_data*: nuevo carácter valido en el puerto data\_in.

*data\_in*: carácter ASCII a escribir.

*new\_goto*: nuevo pedido de posicionamiento del cursor a través de column y row.

*column*: columna donde se desea posicionar el cursor.

*row*: fila donde se desea posicionar el cursor.

**Salidas:**

*busy*: controlador ocupado.

*rw, rs, en*: señales de control al display LCD.

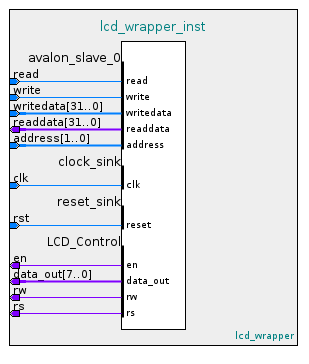
*data\_out*: salida paralelo al display LCD.

El controlador implementado realiza 3 funciones:

1. Inicialización del display.
2. Escritura de un carácter.
3. Función de posicionamiento del cursor en el display.

# Implementacion del wrapper

# Interfaz de E/S del wrapper



entity lcd\_controller\_wrapper is

generic (

MODE\_8\_BITS : std\_logic := '1'; -- 8-bits or 4-bits

LCD\_COLUMNS : std\_logic\_vector(4 downto 0) := "10100"; -- 20

LCD\_ROWS : std\_logic\_vector(1 downto 0) := "11"; -- 4

FREQ : integer := 1 -- system clock frequency in MHz

);

port (

clk, rst : in std\_logic;

-- Avalon bus

read, write : in std\_logic;

address : in std\_logic\_vector(1 downto 0);

writedata : in std\_logic\_vector(31 downto 0);

readdata : out std\_logic\_vector(31 downto 0);

-- LCD outputs

rw, rs, en : out std\_logic; --read/write, setup/data, and enable for lcd

data\_out : out std\_logic\_vector(7 downto 0) --data output to LCD

);

end lcd\_controller\_wrapper;

Este wrapper instancia el modulo de CLP y toma sus mismos parametros:

* MODE\_8\_BITS: se utiliza para configurar el modo de operación en 4 u 8-bits.
* LCD\_COLUMNS: número de columnas/caracteres del display a utilizar.
* LCD\_ROWS: número de filas del display a utilizar.
* FREQ. Frecuencia de operación del bloque, se utiliza para calcular los tiempos de espera.

Incorpara las señales necesarias para el uso de bus Avalon:

* **Entradas:**
  + *read, write*: bits que indican el pedido de una lectura o escritura desde el bus.
  + *address*: direccion del pedido. Se utilizo una direaccion de 4bits.
  + *writedata*: datos de 32 bits a escribir en el bloque wraper cuando se recibe un pedido de escritura a traves de la entrada *write*.
* **Salidas:**
  + *readdata:* datos de 32 bits a leer proporcioandos por el wrapper cuando recibe un pedido de lectura a traves de la entrada *read*.

Y las salidas siguientes se incluyeron en una interfaz llamada LCD\_Control y salen como I/Os del wrapper para ser conectadas en los pines de la FPGA:

* *rw, rs, en*: señales de control al display LCD.
* *data\_out*: salida paralelo al display LCD.

Las siguientes señales del contraldor LCD de CLP son ahora manejadas a traves del bus Avalon:

* **Entradas:**
  + *new\_data*: nuevo carácter valido en el puerto data\_in.
  + *data\_in*: carácter ASCII a escribir.
  + *new\_goto*: nuevo pedido de posicionamiento del cursor a través de column y row.
  + *column*: columna donde se desea posicionar el cursor.
  + *row*: fila donde se desea posicionar el cursor.
* **Salidas:**
  + *busy*: controlador ocupado.

# Registros de lectura/escritura

Se implementaron 3 registros de 32bits para acceder al IP desde el bus Avalon:

1. CHAR: registro para escribir un carácter (write-only) – Direccion 0x0

* Bits 0-7 *data\_in*: carácter a imprimir en el display
* Bits 8-31: no utlizados

1. POSITION: Registro para posiscionar el cursor (write-only) – Direccion 0x1

* Bits 0-1 *row*: fila donde se desea posicionar el cursor ()
* Bits 2-6 *column*: columna donde se desea posicionar el cursor
* Bits 7-31: no utlizados

1. STATUS: Registro de estado (read-only) – Direccion 0x2

* Bit 0 *busy*: controlador ocupado.
* Bits 2-31: no utlizados

Al escribir el registro CHAR con el carácter a mandar en el display, el wrapper automaticamente copia el carácter en los bits 0-7 de la señal ***writedata*** del bus Avalon a la entrada ***data\_in*** del bloque ***lcd\_controller***. Adicionalmente el wrapper debe poner en 1 la señal ***new\_data*** para decirle al bloque ***lcd\_controller*** que debe enviar el carácter en su entrada ***data\_in*** al display.

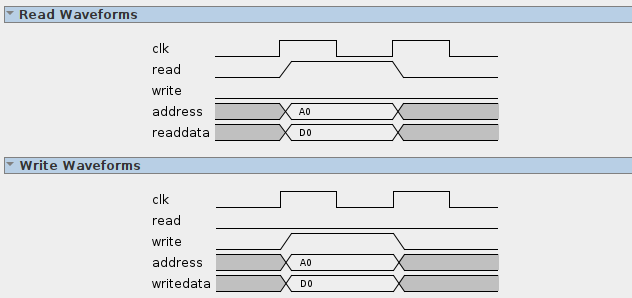
Al escribir el registro POSITION con el los valores de *row* y *column,*  el wrapper realiza una accion similar a la anterior, copia los valores de la señal ***writedata*** del bus Avalon a las entradas ***row*** y ***column*** del bloque ***lcd\_controller*** y pone en 1 la señal ***new\_goto*** para decirle al bloque ***lcd\_controller*** que debe enviar el comando de poscionar cursor con los datos en las entradas ***row*** y ***column*** al display.

Por ultimo, el registro STATUS es utilizado por el software, para a travez del bus Avalon consultan el estado del controlador. En este caso el estado corresponde al bit bussy que indica si el controlador esta ocupado enviando un carácter. No se puede enviar otro carácter hasta que busy sea 0.

# Simulaciones

# Testbench

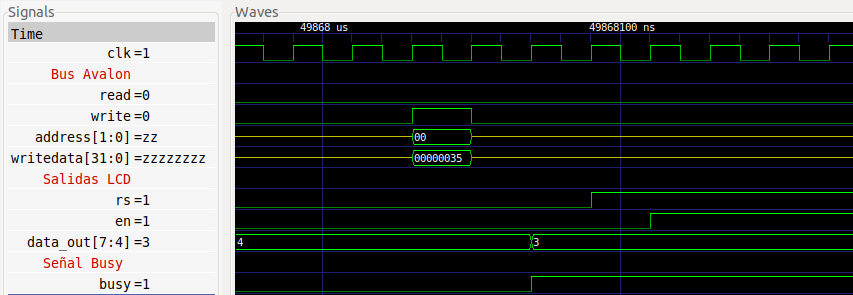
A partir de la documentacion de Quartus del bus Avalon se obtuvieron las siguientes formas de ondas utilizadas para la implementacion y luego para el testbench:



Se modulo el bus y se verifico que el nuevo sistema implementado responda como el bloque implementado en CLP pero ahora manejado por el bus Avalon.

Otro detalle incluido en las simulaciones es la frecuencia de 50Mhz que ahora ingresa al modulo de CLP, ya que este fue probado con 1Mhz en la materia anterior. En el testbench modifique la frecuencia y verique las formas de onda aun respeten los tiempos requeridos por el display.

# Operacion de escritura en el bus

En la figura se puede observar las señales del bus Avalon (read, write, address y writedata) address y writedata estan en Z antes y despues del pedido de escritura o lectura.

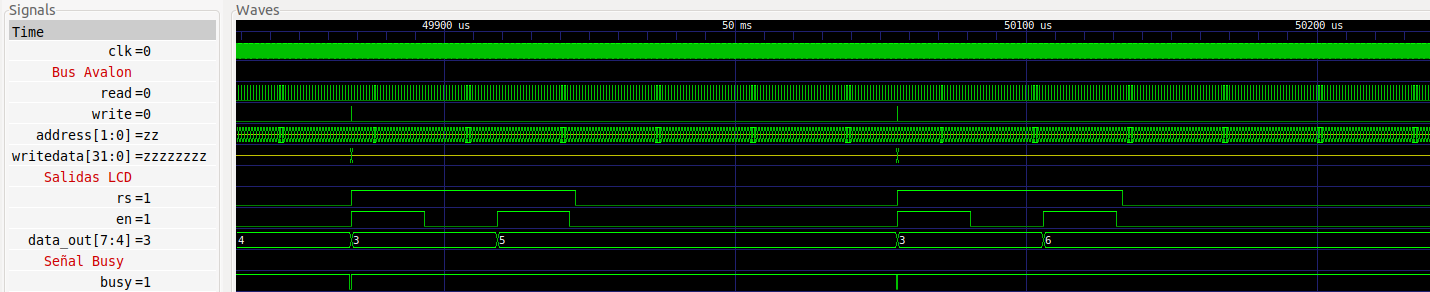
Cuando aparece el pedido de escritura de un carácter se apuntan la señal address a 0x00, y el data a escribir es 35, se puede ver que luego de un clk se levanta la señal de busy del lcd\_controller y las señales de rs, en y data\_out comienzan a tener actividad. En particualr data\_out saco la parte alta del dato escrito 0x3 ya que se esta trabajando en 4-bits.

# Operacion de lectura en el bus

# 

La operación de lectura es similar, pero en este caso se levanta el bit read del bus Avalon y las direcciones apuntan al registro de STATUS (0x10), en ese momento el warpper escribe en la señal readdata el valor del bit busy (‘1’) en ese momento.

# Operacion normal



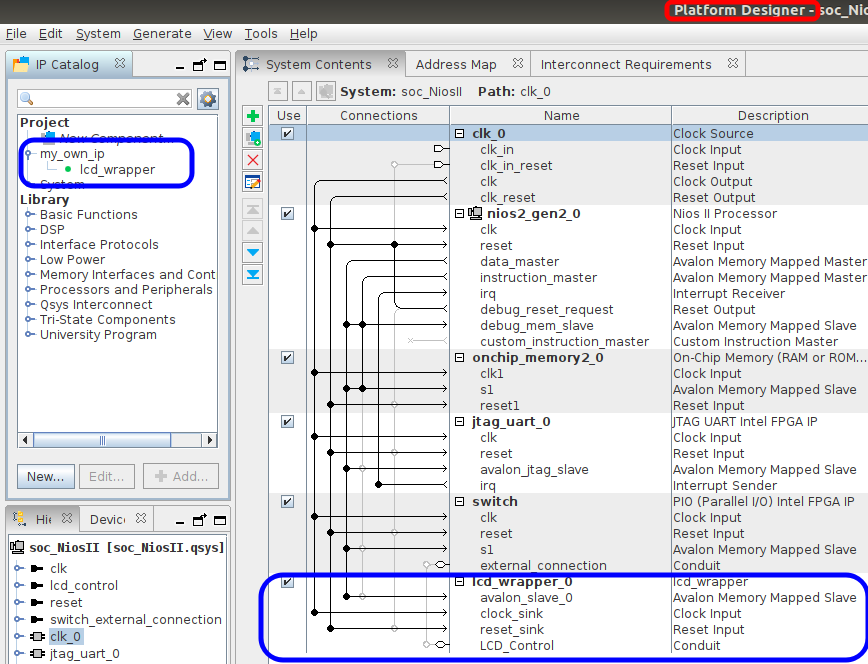
En esta simulacion se muestra un zoom out del caso de la escritura del carácter 0x35 para mostrar como se manda en dos tandas. Se observa el compartamieto de todas las señales del salidas al LCD, el cual es el mismo que en el trabajo de CLP.

Por otro lado, se observa una gran actividad en la linea read y address del bus Avalon, esto se debe a que el testbench esa preunando todo el tiempo por el bit busy para saber cuando puede enviar otro carácter.

# Implementación en Quartus

# Esquemático

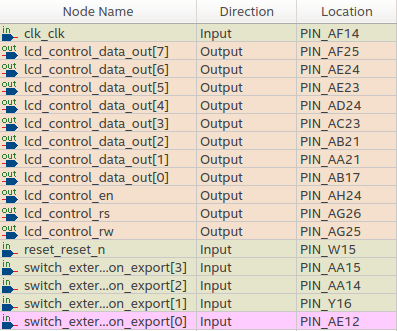
En la siguiente figura se observa el esquemático planteado en Quartus utilizando Platform Designer:

**Bloque instanciados:**

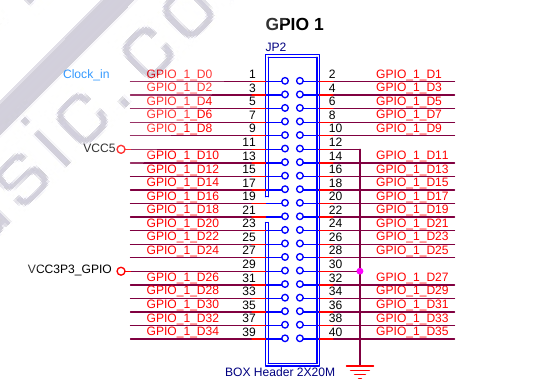
* Clock Source (clk\_0): utilizando una frecuencia de 50Mhz.
* Nios II Processor (nios2\_gen2\_0): procesador softcore.
* On-Chip RAM Memory
* JTAG UART: utilizado en software para poder acceder al JTAG e imprimir valores por la consola.
* PIO – Parallel I/O: para poder leer los switches de la placa de desarrollo.
* lcd\_wrapper: bloque diseñado para la materia.

# Pines de entrada/salida

Como se comentó, se utiliza la placa DE1-SoC rev C. La imagen de Pin Planner muestra los I/O utilizados:



La siguiente figura muestra el conector JP2 para los GPIO\_1:



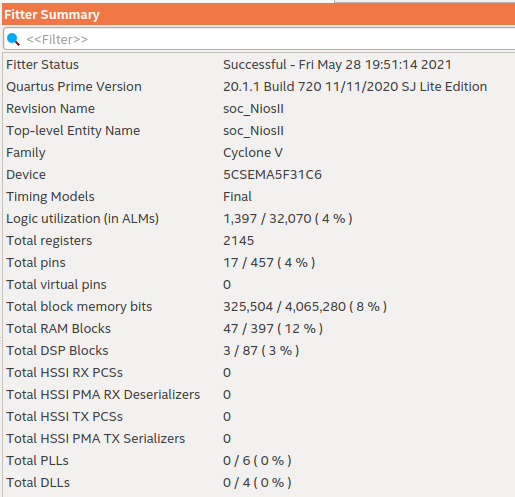
Se cuenta con un display de 20x4 líneas del tipo 2004A que está conectado para ser usado con datos de 4-bits.



Conexionado entre display y conector JP2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Display** | | | **Kit de desarrollo** | |
| **Pin** | **Función** | **Pin de JP2** | | **Función** |
| 1 | VSS | 30 | | GND |
| 2 | VDD | 29 | | VCC3P3\_GPIO |
| 3 | V0 | 30 | | GND |
| 4 | RS(C/D) | 13 | | GPIO\_1\_D10 |
| 5 | R/W | 30 | | GND |
| 6 | E | 14 | | GPIO\_1\_D11 |
| 11 | DB4 | 5 | | GPIO\_1\_D4 |
| 12 | DB5 | 6 | | GPIO\_1\_D5 |
| 13 | DB6 | 7 | | GPIO\_1\_D6 |
| 14 | DB7 | 8 | | GPIO\_1\_D7 |
| 15 | A | 30 | | GND |
| 16 | K | 29 | | VCC3P3\_GPIO |

# Recursos de la FPGA



# Firmware

Para el firmware se utlizo la herramienta “Nios II Software build tools from Eclipse”.

Para la implementacion se deben incluir los siguientes .h:



