Diseño y Construcción de Sistemas Digitales
Facultad de Informática
UPV/EHU

Diseño y Construcción de Sistemas Digitales

Proyecto LCD LT24

Iker Fernández
Jon Moriñigo

26 de octubre de 2021

Resumen

Esta es la documentación de la primera parte del proyecto realizado en la asignatura **Diseño y Construcción de Sistemas Digitales** durante la mitad del priemr cuatrimestre del curso 2021-2022.

Esta fase del proyecto consiste en controlar la pantalla LCD LT24 para poder hacer dibujos simples, enviando los comandos usando el teclado. Para ello hemos tenido que modelar la placa DE1-SoC de Altera y pantalla LCD LT24 y se han usado los programas ModelSim y Quartus para ello.

Las dos funcionalidades que se han implementado son borrar pantalla y dibujar diagonal, las cuales serán presentadas en una presentación delante del profesor. Este documento describe el la metodología utilizada para lograr el objetivo y muestra todo el recorrido realizado a lo largo del proyecto así como una explicación de cada uno de los diseños.

Para poder trabajar en pareja se ha hecho uso de herramientas orientadas a proyectos en grupo como GitHub (donde se encuentra todo el proyecto actualizado y sus diferentes versiones), Googledrive y OverLeaf (para generar la documentación).

Índice general

1. Introducción

Este trabajo ha sido desarrollado en la asignatura **Diseño y Construcción de Sistemas Digitales**, ofertada el 1º curso de Ingeniería Informática en la Facultad de Informática de la Universidad del País Vasco (UPV/EHU). Siendo este realizado en el curso 2021 - 2022 por Jon Moriñigo e Iker Fernández.

En la primera sesión del curso se nos presentó una pregunta motriz y un escenario sobre el cual íbamos a trabajar *(Figura 2.1)*. El proyecto consiste en controlar la pantalla LCD LT24 para poder hacer dibujos simples, enviando los comandos usando el teclado. Estos comandos son simples y de funcionalidad mínima: borrar pantalla, cambiar color, dibujar hacia arriba...

Nuestras herramientas de trabajo han sido los programas ModelSim para la simulación y Quartus para el diseño. Una vez simulados virtualmente nuestros algoritmos se han utilizado la placa DE1-SoC de Altera y pantalla LCD LT24.

Finalmente comentar que esta es la primera fase del proyecto, la cual tan solo se nos ha pedido implementar dos funcionalidades: borrar pantalla y dibujar diagonal. El objetivo es que a medida que avance la asignatura tengamos la capacidad para implementar nuevas funcionalidades, las cuales se presentaran en las siguiente iteración.

(a) Pagina 1 (b) Pagina 2

Figura 1.1: Pregunta Motriz

2. Objetivo del Proyecto

El objetivo del proyecto es claro, aprender a diseñar y construir sistemas digitales de complejidad media partiendo de conocimientos previos y a adquiriendo los conocimientos necesarios. Es decir, se quiere poner en practica la teoría de la asignatura a través de un proyecto global.

Como es evidente es prácticamente imposible hacer el proyecto entero bien a la primera. Es por ello que primero en clase se han realizado y corregido varios ejercicios de menor complejidad. También es necesario dividir el proyecto en 2 iteraciones para poder ver y solucionar los problemas a medida que el proyecto avance.

El trabajo se realiza en parejas, lo cual no significa que el trabajo se tenga que dividir equitativamente. El objetivo de la asignatura es obtener el máximo conocimiento posible, por eso es importante que ambos integrantes hagan el trabajo de manera simultanea. De esta manera los dos aprenderán a buscar una solución por su propia cuenta y, en caso de tener un problema o duda, ayudarse entre si.

3. Metodología de diseño

3.1 UC+UP

Lo primero que se hizo en el proyecto fue la parte de la unidad de control y la unidad de proceso tanto del LCD_CONTROL primero como del LCD_DRAWING. La metodología que se ha usado es la misma que en clase para los ejercicios.

Empezamos haciendo la unidad de control del LCD_CONTROL en clase, cogiendo como referencia los ejercicios hechos previos en clase, sin embargo las primeras versiones fueron bastante pobres y poco realistas debido a la inexperiencia en el diseño de sistemas digitales. La unidad de proceso se hace a la par que la unidad de control añadiendo los componentes que sean necesarios en esta. Una vez terminada la primera versión se utilizo como modelo para hacer el VHDL. La parte del LCD_DRAWING se hizo después de comprobar que la simulación del LCD_CONTROL funcionaba bien.
Como es normal tanto la unidad de control como la de proceso se han ido modificando a lo largo del proyecto puesto que la primera versión era muy pobre. A la hora de hacer el VHDL es donde mas modificaciones se han hecho puesto que es donde se han realizado las simulaciones y donde se determina si hace bien la funcionalidad. Finalmente a la hora del la implementación del Hardware no se tuvieron que hacer apenas modificaciones puesto que la simulación iba correcta.

3.2 Diseño VHDL

Para el diseño del VDHL usamos como plantilla el VHDL de un ejercicio modelo que hicimos en clase entre todos. Las instrucciones que se nos dieron y el modelo que usamos nos sirvieron como base para empezar a hacer las primeras versiones del proyecto. El principal problema fue la falta de conocimiento e información sobre el programa ModelSim, sin embargo a medida que avanzábamos en el proyecto la fluidez y la eficiencia mejoraban considerablemente.

Para generar el código utilizábamos como referencia la unidad de control y la unidad de proceso, las cuales las teníamos siempre a mano puesto que eran necesarias para saber qué teníamos que programar. Para comprobar si el vhdl generado funcionaba se hacía a través de la simulación del ModelSim de la cual hablaremos en el siguiente apartado.

En esta parte el LCD_CONTROL era el que más nos constó puesto que era el diseño más complejo. Una vez que se realizó este se empezó con el LCD_DRAWING, el cual era más

sencillo.

3.3 Simulación

En la simulación en ModelSim se utilizaron ficheros vhdl para generar un test bench. Estos sirven para simular y ver las señales del LCD_CONTROL y LCD_DRAWING.

En el caso del LCD_CONTROL teníamos un manual donde se nos explicaba que forma tenían que tener las señales para que funcionase bien el programa, por tanto era más fácil para saber si iba a funcionar bien. El problema principal era coordinar bien los tiempos con los cambios de estados.

Para el LCD_DRAWING no teníamos un cronograma ejemplo para saber sí funcionaba correctamente, sin embargo no fue problema puesto que era más fácil de implementar. Para conectar los dos programas y simularlos a la vez, se genero un test bench conjunto donde se veía si las señales hacían su correcto funcionamiento.

3.4 Implementación Hardware

Una vez finalizada la simulación y comprobado su correcto funcionamiento ya era el momento de probarlo en la placa. Para ello utilizamos el programa Quartus el cual nos servía para diseñar y programar los componentes de la placa Intel.

Al tener ya el diseño del VHDL y la simulación hecha era muy sencillo crear un TOP para que funcionase en la placa, puesto que solo teníamos que asignar botones y leds a las entradas y salidas de los módulos del LT24.

Para familiarizarnos con el programa y hacer pruebas sobre la placa, hicimos el TOP del LCD_CTRL a partir de un proyecto modelo que se nos dio en clase. Tras la verificar que funcionaba bien, hicimos la ejecución final con el TOP que nos dieron de la fase 1.

3.5 Test del prototipo

LA ultima parte es testear que el diseño de vhdl que hemos hecho anteriormente funciona en la placa. Quartus proporciona herramientas que automáticamente generan un código que se puede ejecutar en la placa.

Comprobar el correcto funcionamiento era bastante sencillo puesto que se asignaban dos botones los cuales tenían las dos funcionalidades que se nos pedían para la fase 1. En teoría si la simulación funcionaba bien esta parte también debería funcionar bien. Sin embargo a la hora de ejecutar pruebas pudimos localizar bugs por escenarios que no habíamos considerado. Para solucionarlos modificamos el vhdl desde ModelSim y volvimos a hacer la simulación y las pruebas nescesarias.

3.6 Herramientas de diseño, simulación, validación

Como he ido comentando a lo largo de la documentación hemos usado los programas ModelSim y Quartus para el diseño, simulación y validación. Ambos son programas que ofrece Intel para Diseño y Simulación de sus placas FPGA´s.

Para comprender el funcionamientos de ambos programas hemos seguido los respectivos tutoriales que se nos han dado en la asignatura. Sin embargo a medida que avanzábamos en el proyecto mejor nos envolvíamos con estas herramientas. También hemos hecho pruebas con ejercicios propuestos en clase más sencillos los cuales nos han servido para ver como hay que hacer la correcta simulación.

4. Diseño del sistema

4.1 Descripción General del Sistema

4.2 Módulos

El sistema está construido por tres módulos los cuales están conectados entre si por sus salidas y entradas para poder comunicarse entre ellos. Los módulos son LT24Setup, LCD_CTRL y LCD_DRAWING y cada uno realiza una función.

4.2.1 LT24Setup

El módulo LT24Setup es responsable de la inicialización de la pantalla LCD. Se inicia automáticamente cuando se activa la señal de Reset y activa la salida LT24_Init_Done cuando se completa el proceso de inicialización. De ahí en adelante, está listo para procesar adecuadamente la información proveniente del resto de señales de entrada, es decir, para transmitir la información generada por otro módulo al componente ILI.

Las entradas del LT24Setup son las siguientes:

- **LT24_CS_N_Int:** El LCD_CTRL indica por esta señal que se envía un dato o comando.
- **LT24_WR_N_Int:** El LCD_CTRL indica por esta señal que se envía un dato o comando.
- **LT24_RD_N_Int:** No se usa.
- **LT24_RS_Int:** El LCD_CTRL indica por esta señal que se envía un comando.
- **LT24_D_Int:** El LCD_CTRL indica por esta señal el comando a ejecutar, la posición a moverse o el color a poner en la pantalla.

Las salidas del LT24Setup son las siguientes

-
- **LT24_LCD_ON**Enciende la pantalla LCD.
- **LT24_RESET_N**Resetea la pantalla LCD.

- **LT24_CS_N**Por esta señal indica a la pantalla LCD que se envía un dato o comando.
- **LT24_WR_N**Por esta señal indica a la pantalla LCD que se envía un dato o comando.
- **LT24_RD_N**No se usa.
- **LT24_RS**Por esta señal indica a la pantalla LCD que se envía un comando.
- **LT24_D**Por esta señal indica a la pantalla LCD el comando a ejecutar, la posición a moverse o el color a poner en la pantalla.
- **LT24_Init_Done:** Señal que indica al LCD_CTRL que la pantalla esta lista.

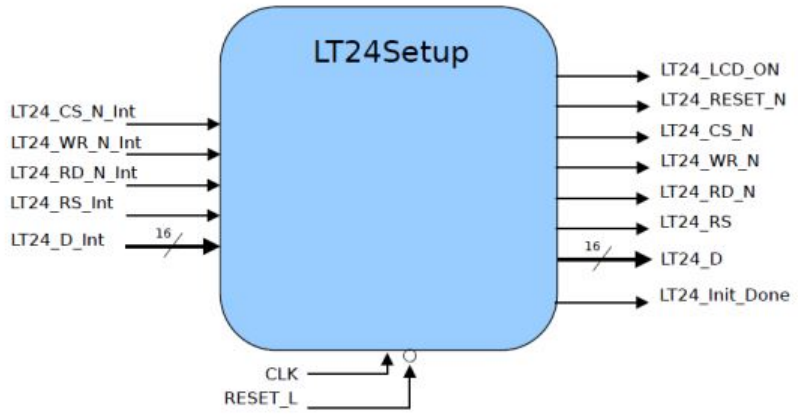

Figura 4.1: Módulo LT24Setup

4.2.2 LCD_CTRL

Este módulo es el encargado de mandar los comandos de posición y de dibujar a la pantalla. Es el que confirma al LCD_DRAWING cuando hace alguna de estas dos acciones.
Las entradas del LCD_CTRL son las siguientes:

- **LT24_Init_Done:** El LT24Setup avisa por esta señal cuando la placa está preparada.
- **OP_SETCURSOR:** El LCD_DRAWING avisa por esta señal que tiene posicionar el cursor.
- **XCOL:** El LCD_DRAWING indica por esta señal la posición del eje X.
- **YROW:** El LCD_DRAWING indica por esta señal la posición del eje X.

- **OP_DRAWCOLOUR:** El LCD_DRAWING avisa por esta señal que tiene dibujar en la pantalla.
- **RGB:** El LCD_DRAWING avisa por esta señal que color dibujar.
- **NUM_PIX:** El LCD_DRAWING avisa por esta señal la cantidad de pixeles a dibujar.

Las salidas del LCD_CTRL son las siguientes

- **DONE_CURSOR:** Señal que indica al LCD_DRAWING si se ha posicionado el cursor.
- **DONE_COLOUR:** Señal que indica al LCD_DRAWING si se ha dibujado la pantalla.
- **LCD_CS_N:** Señal que indica al LT24Setup que se envía un dato o comando.
- **LCD_WR_N:** Señal que indica al LT24Setup que se envía un dato o comando.
- **LCD_RS:** Señal que indica al LT24Setup que se envía un comando.
- **LCD_DATA:** Señal que indica al LT24Setup el comando a ejecutar, la posición a moverse o el color a poner en la pantalla.

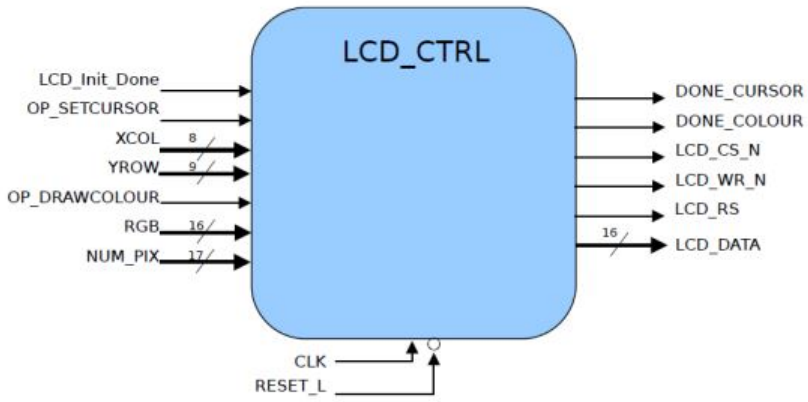

Figura 4.2: Módulo LCD_CTRL

4.2.3 LCD_DRAWING

El segundo módulo es el que se encargará de dibujar en pantalla lo que nosotros indiquemos. En este caso se podrá borrar la pantalla o dibujar una diagonal.

Las entradas del LCD_DRAWING son las siguientes:

- **DEL_SCREEN:** Señal que se activa para borrar la pantalla.
- **DRAW_DIAG:** Señal que se activa para dibujar una diagonal en la pantalla.
- **COLOUR_CODE:** Señal que indica que color dibujar.
- **DONE_CURSOR:** El LCD_CTRL indica si se ha posicionado el cursor.
- **DONE_COLOUR:** El LCD_CTRL indica si se ha dibujado la pantalla.

Las salidas del LCD_DRAWING son las siguientes

- **OP_SETCURSOR:** Señal que avisa al LCD_CTRL que tiene posicionar el cursor.
- **XCOL:** Señal que indica al LCD_CTRL la posición del eje X.
- **YROW:** Señal que indica al LCD_CTRL la posición del eje X.
- **OP_DRAWCOLOUR:** Señal que avisa al LCD_CTRL que tiene dibujar en la pantalla.
- **RGB:** Señal que indica al LCD_CTRL que color dibujar.
- **NUM_PIX:** Señal que indica al LCD_CTRL la cantidad de pixeles a dibujar.

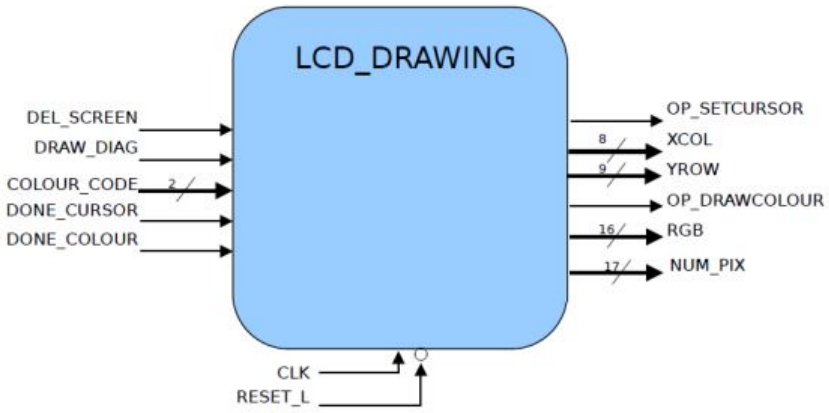

Figura 4.3: Módulo LCD_DRAWING

4.3 UC+UP

4.3.1 LCD_CRTL

El primero modulo que teníamos que diseñar es el del LCD_CTRL, el cual tiene dos funcionalidades principalmente, posicionar el cursor y dibujar. Para ello según cual de las dos señales se active irá a una rama u a otra.

En la parte del OP_SETCURSOR se posicionará el cursor, indicando la posición X e Y. Para posicionar el cursor se necesitan 6 iteraciones de 4 ciclos de reloj cada uno. Primero se manda el comando para indicar la posición X, y a continuación se envía en dos iteraciones el valor indicado. Para la posición Y se vuelve a hacer lo mismo.oo
La parte del OP_DRAWCOLOR es similar a la anterior. En esta se manda el comando para dibujar y a continuación se dibujan la cantidad de pixeles que se indican.

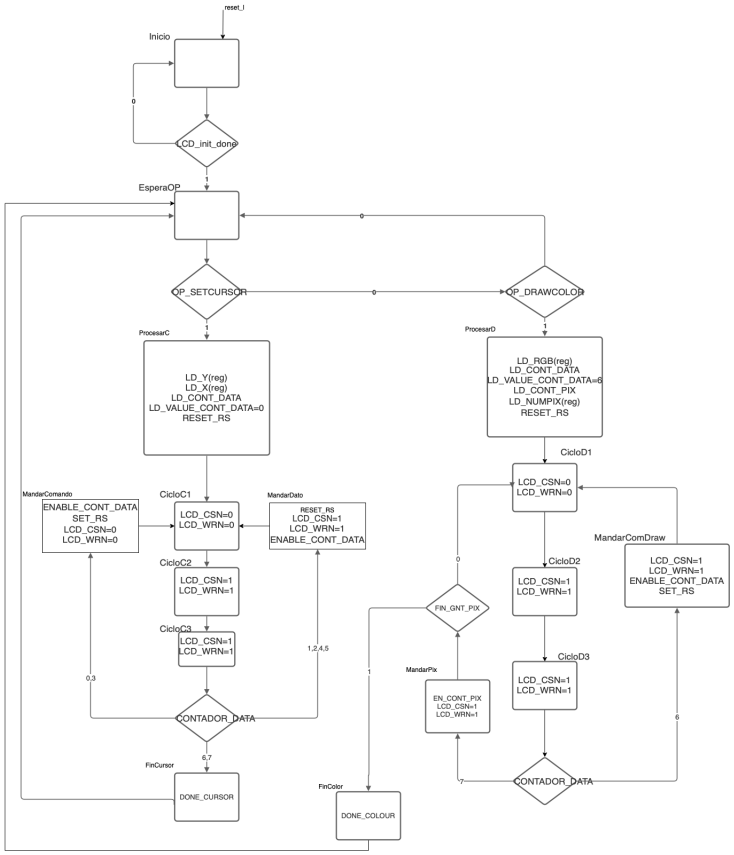

Figura 4.4: Unidad de Control de LCD_CTRL

En la Unidad de Proceso se han utilizado los siguientes componentes:

- **Registro Estado:** Registro para guardar el estado en el que se encuentra la maquina.
- **Registro X:** Registro para guardar el valor de la posición X.
- **Registro Y:** Registro para guardar el valor de la posición X.
- **Registro ContadorCiclos:** Contador de ciclos para controlar si hay que mandar dato o comando.
- **Registro RGB:** Registro para guardar el color del que se quiere dibujar el pixel.
- **Registro NUMPIX:** Registro para guardar el numero de pixeles que se quiere dibujar.
- **Registro RS:** Registro para guardar el valor de RS.
- **Contador Pixeles:** Contador que lleva la cuenta de los pixeles dibujados.
- **Multiplexor Datos:** Multiplexor que según el ciclo de la operación que indique el registro de contador de ciclos sacará dato o comando.

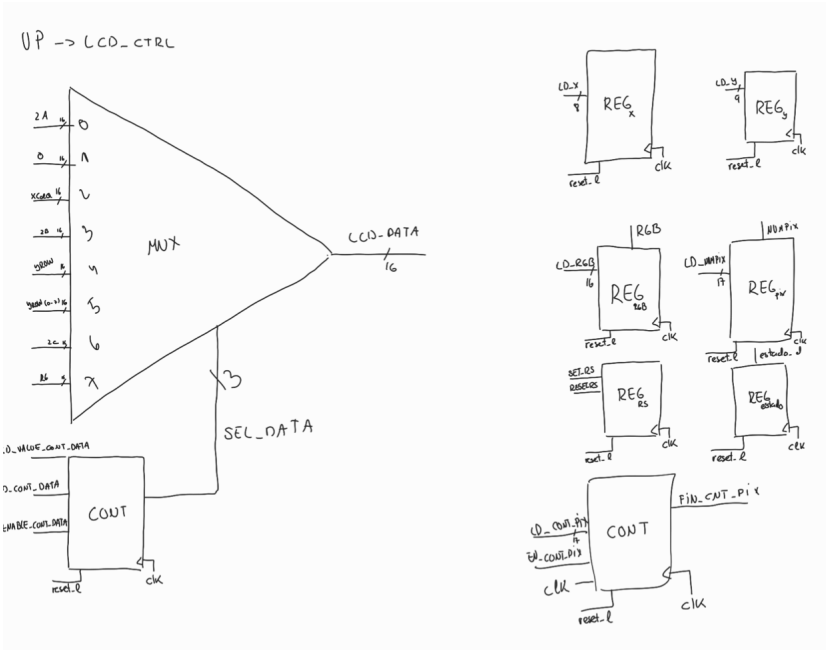

Figura 4.5: Unidad de Proceso de LCD_CTRL

4.3.2 LCD_DRAWING

El segundo módulo es el que se encargará de dibujar en pantalla lo que nosotros indiquemos. En este caso se podrá borrar la pantalla o dibujar una diagonal.

Para borrar la pantalla primero se pone el cursor al principio de la pantalla y se espera a que el LCD_CTRL confirme que se ha posicionado. Seguidamente se pintan todos los pixeles de negro y se espera a que el LCD_CTRL de la señal para finalizar la tarea.

Para dibujar una diagonal al igual que para borrarla se posiciona el cursor al principio de la pantalla y se espera que el LCD_CTRL se ubique. A continuación pintará la pantalla del color que se haya indicado, para ello avanzará un pixel en X e Y, y pintara cinco pixeles. Cuando llegue al final del eje X empezará otra vez desde el pincipio del mismo eje y cuando llegue al final del eje Y se acabara la tarea.

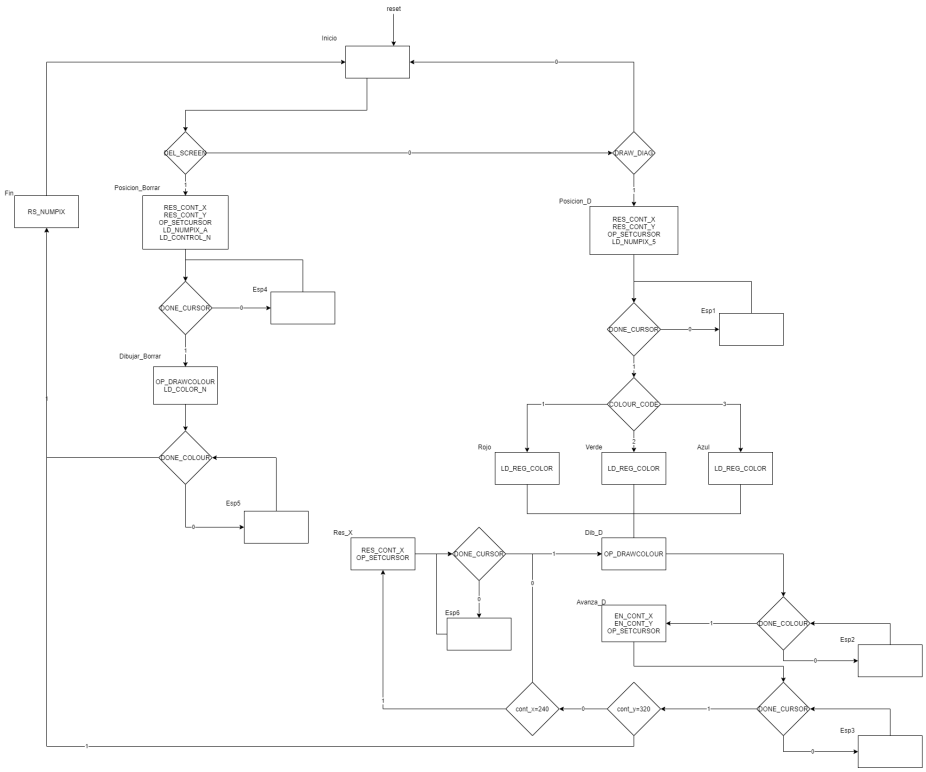

Figura 4.6: Unidad de Control de LCD_DRAWING

En la Unidad de Proceso se han utilizado los siguientes componentes:

- **Registro Estado:** Registro para guardar el estado en el que se encuentra la maquina.

- **Registro Color:** Registro para guardar el valor que se quiere pintar.
- **Registro NumPix:** Registro para indicar que valor tiene que cargar Numpix.
- **Contador X:** Contador que recorre el eje X.
- **Contador Y:** Contador que recorre el eje Y.

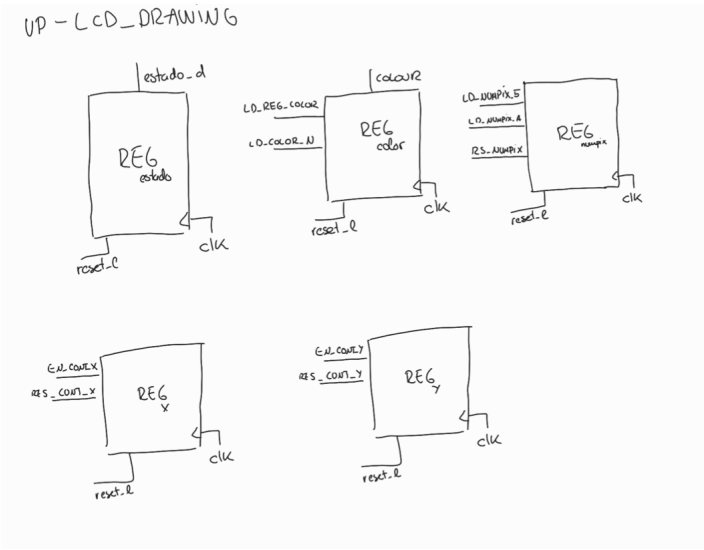

Figura 4.7: Unidad de Proceso de LCD_DRAWING

4.4 Descripción VHDL

Los módulos están programados en archivos VHDL los cuales tiene una estructura fija:

- Primero se define la entidad o el módulo. Aqui se definen todas las entradas y salidas y qué tipo de señal son.
- Luego se define la arquitectura de la entidad creada
- Lo primero que se define en la arquitectura son los estados, registros y señales que se van a utilizar para el correcto funcionamiento
- Luego definimos con un proceso los cambios de estado y las condiciones que se tienen que cumplir para que esto pase.
- Seguidamente damos valores a las entradas y salidas del módulo y a las señales que hemos creado anteriormente.

- Finalente definimos a base de procesos los contadores, registros y multiplexores que hemos definido en la unidad de proceso.

Esta es la estructura que se ha usado con todos los vdhl, el código de cada uno está almacenado una misma la carpeta.

4.5 Simulación

Para la simulación hay un archivo de test bench para cada uno de los vhdl los cuales hemos usado para hacer simulaciones en ModelSim.

5. Test y Simulación

A continuación se van a mostrar unas capturas que hicimos de las pruebas del modelo VDHL. Estas se han hecho desde ModelSim como ya hemos mencionado y nos han servido para fijarnos en los ciclos y transiciones de estado.

A la hora de ejecutar el programa en la placa si veíamos que algo no funcionaba de la manera correcta era bastante complicado intentar localizar el fallo. Es por eso que nos hemos intentado aseguran un buen comportamiento de este desde el principio, y en caso de que no funcionase bien en la placa, se volvía a rediseñar el modelo y a hacer la pruebas desde ModelSim y no directamente a la placa.

5.1 Simulación del LCD_CTRL

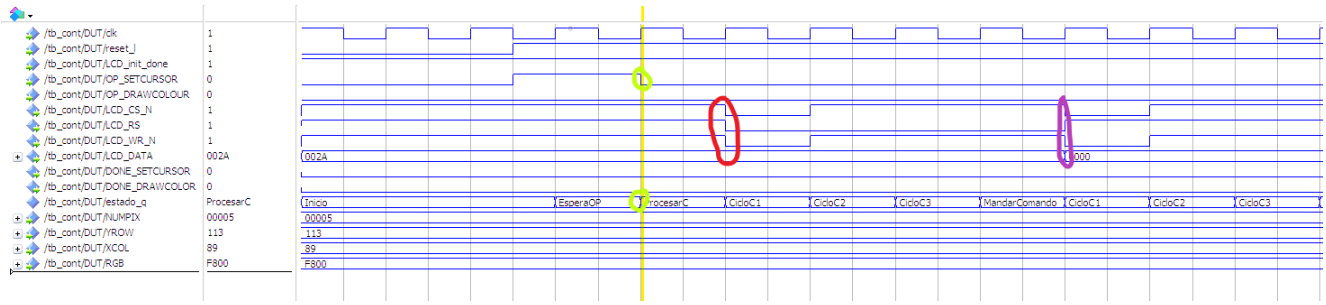

Figura 5.1: Primera simulación LCD_CTRL

Las zonas verdes muestran el momento en el que se activa la señal OP_SETCURSOR y se observa como en el mismo ciclo se cambia de estado a procesar la señal. Por otro lado (Zona roja), podemos observar en la activación de envío del comando en las señales RS, WR_N y CS_N. Finalmente en la zona morada podemos apreciar como al enviarse un dato se activas las señales WR_N y CS_N pero no la RS.

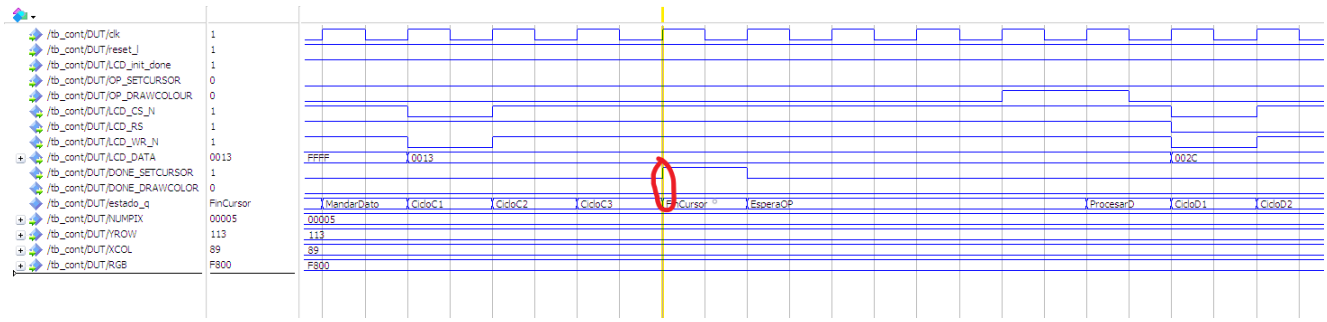

Figura 5.2: Segunda simulación LCD_CTRL

La zona roja nos muestra como al finalizar la ejecución del OP_SETCURSOR se activa la señal DONE_SETCURSOR en el estado "FinCursor".

El mismo comportamiento visto en las dos anteriores pruebas realiza la ejecución de la instrucción OP_DRAWCOLOR.

5.2 Simulación del LCD_DRAWING

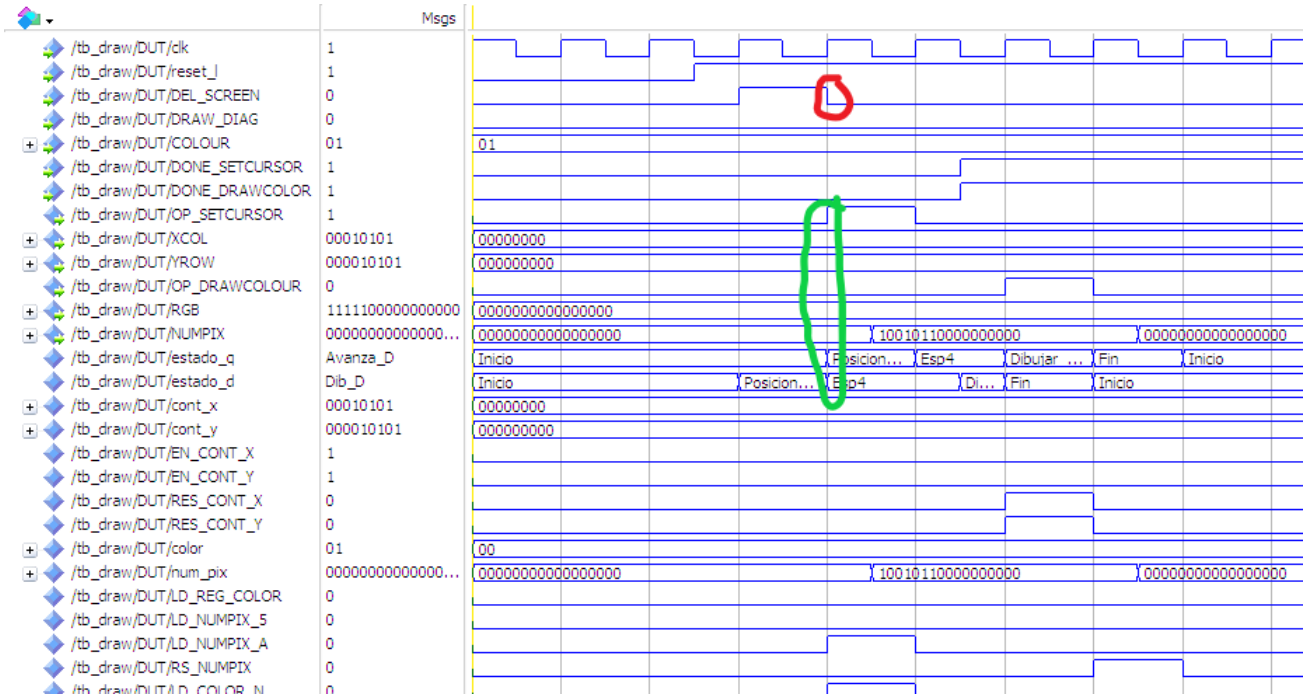

Figura 5.3: Primera simulación LCD_DRAWING

En la zona roja podemos observar como al activarse la señal DEL_SCREEN cambiamos de estado a Procesar la posición del cursor elevando para ello la señal OP_SET_CURSOR (Zona verde). También podemos observar como el numero de píxeles se establece al máximo para realizar un barrido completo de la pantalla.

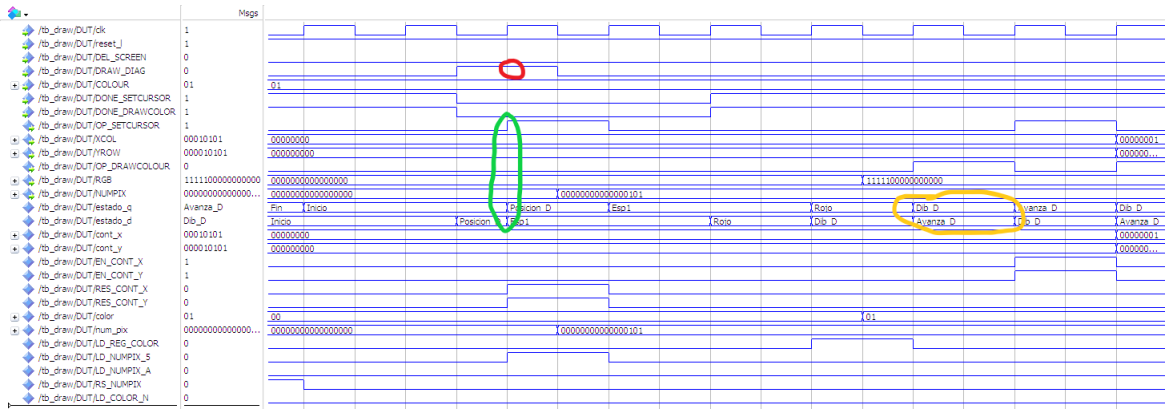

Figura 5.4: Segunda simulación LCD_DRAWING

En la zona roja podemos observar la activación de la señal DEL_SCREEN se produce en la zona verde un cambio de estado para procesar el dibujo de los píxeles correspondientes. A continuación en la zona amarilla se puede apreciar como el algoritmo funciona. Al avanzar pinta tres píxeles en el color seleccionado, en este caso rojo, y a continuación baja una fila. Así hasta terminar su ejecución y llegar al final de la pantalla LCD y volver al estado de Espera.

6. Manual de usuario

A continuación una breve descripción de como inicializar la placa y hacer uso de sus funcionalidades.

6.1 Encender e inicializar la placa

Para usar la placa primero hay que conectarla a la corriente con la fuente de alimentación y conectarla mediante el cable USB a un ordenador que tenga instalado Quartus con el proyecto final ya compilado. En este momento la placa estará iluminando una secuencia en los leds y la pantalla estará encendida en blanco.

Para empezar la ejecución, dentro de Quartus hay que ir a Tools-Programer y se nos abrirá una ventana nueva. En esta ventana, si es la primera vez que ejecutamos el programa, habrá que pulsar Hardware Setup, seleccionar DE-SoC, y pulsar Close. Tras esto, ya puede pulsarse el botón Auto Detect y seleccionar 5CSEMA5 para buscar dispositivos en la placa.
Aparecerán en la parte inferior dos dispositivos. Seleccionamos el que corresponde a la FPGA (5CSEMA5), pulsamos Change File..., y elegimos el fichero .sof correspondiente al diseño en la subcarpeta output_files. Finalmente activamos la casilla Program del fichero que queremos programar y pulsamos Start.

6.2 Borrar Pantalla

La primera de las funciones es borrar pantalla, que como indica su nombre sirve para poner todos los pixeles de la pantalla en negro. Se puede usar en cualquier momento y sin ninguna condición.

Para borrar la pantalla está asignado el botón número dos *(KEY2)*. Tras pulsarlo, si ha ido correctamente se encenderán los leds 7 y 8 *(LEDR7 y LEDR8)* y la pantalla se pondrá completamente en negro. Se recomienda nada más empezar la ejecución usar esta función, puesto que de inicio la pantalla está en blanco y a la hora de dibujar la diagonal se ve mejor con el fondo en negro.

6.3 Dibujar Diagonal

La segunda función es dibujar diagonal, que en este caso se dibuja una diagonal de cinco pixeles de anchura. En cuanto llega al fin del eje X, sigue a la misma altura desde el inicio de este eje. La diagonal se puede dibujar en rojo, verde o azul.

Para dibujar la diagonal está asignado el botón número tres *(KEY3)*, y habrá que indicar el color de la diagonal con los switches cero y uno *(SW0 y SW1)*: para el rojo debe estar activado el switch cero y desactivado el uno *(SW0=1 y SW1=0)*, para el verde debe estar activado el switch uno y desactivado el cero *(SW0=0 y SW1=1)* y para el azul ambos switches deberán esta activados *(SW0=1 y SW1=1)*. Tras pulsar el botón tres, si ha ido correctamente se encenderán los leds 7 y 8 *(LEDR7 y LEDR8)* y la pantalla mostrará la diagonal en su debido color.

Finalmente si ambos switches están desactivados *(SW0=1 y SW1=0)* no se mostrará ninguna diagonal y la pantalla se quedará en el mismo estado en el que estaba.

7. Conclusiones y propuesta de posibles mejoras

Este proyecto consiste en poner en practica todos los conocimientos que hemos obtenido hasta el momento en la asignatura Diseño y Construcción de Sistemas Digitales sobre el modelaje del hardware. El proyecto está dividido en dos iteraciones, lo cual significa que esta primera fase es solo la de introducción, por tanto nuestra visión y conclusiones sobre el proyecto y la asignatura pueden variar en un futuro.

Para esta iteración se nos ha pedido completar las dos primeras funcionalidades que se nos habían planteado a principios de curso, las cuales hemos conseguido desarrollar y ejecutar de forma correcta. El objetivo con esto era aplicar la teoría y los ejercicios hechos en clase a una placa real de una manera sencilla y guiada. Como conclusiones podemos decir que hemos aprendido como se hace una buena construcción de un sistema digital: diseño, simulación, modelaje y verificación.

El proyecto todavía esta en su primera versión, lo que significa que todavía hay margen de mejora y desarrollo. Para la segunda iteración se nos propone crear dos nuevas funcionalidades propias. Visto como funciona la plca y los programas de simulación y modelaje, ahora tenemos más claro que cosas se podrían hacer en la placa dentro de nuestras posibilidades y restricciones. Es por eso que en futuras versiones podríamos hacer una funcionalidad interactiva, como visualizar un objeto que se mueve en tiempo real, o incluso dibujar formas con distintos colores.

En conclusión este nos ha parecido una buena introducción al mundo de sistemas digitales y que nos sirve de forma sencilla para motivarnos a seguir trabajando en esta asignatura.

8. Seguimiento del Proyecto

A continuación mostramos un seguimiento de las horas dedicadas a este proyecto fuera de las horas lectivas.

Semana	L	M	X	J	V	S	D
27/9-03/10							
04/10-10/10			4h	2h			
11/10-17/10			2h		2h		
18/10-24/10		1.5h		1h			
25/10-31/10				2h		1h	
01/11-07/11	2,5h	3h	1h	2h	4h	3h	3h

9. Bibliografía

- **eGela (UPV/EHU) - Diseño y Construcción de Sistemas Digitales**
  Guía rapida para Quartus y ModelSim
  LT24 Manual/Guia Rapida

- **StackOverflow**
  Dudas generales de VHDL