MEMORIA

GRUPO 4

AUTORES:
Haizea Bermejo Pascual
Marcos Chouciño Grijalbo
Tserenlkham Undrakh

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

Fecha de entrega del proyecto: 22 de diciembre
Fecha de entrega de la memoria: 12 de enero
Fecha de exposición: 19 de enero

Resumen

ÍNDICE

Introducción

En este proyecto se diseñará y se desarrollará un sistema digital capaz de controlar una pantalla LT24 para hacer dibujos simples y para enviar una imagen contenida en un fichero. Para ello se dividirá el sistema en tres módulos : LT24Setup inicializará la pantalla LCD, LCD_CTRL hará las operaciones básicas sobre la pantalla y LCD_DRAWING hará las operaciones de borrado de pantalla y dibujar una línea.

Objetivo del proyecto

Crear un sistema digital con estas funcionalidades mínimas: Comando de borrado de pantalla, comando para dibujar una línea, opción de cambio de color y comando para recibir una imagen mediante conexión serie (UART) con una velocidad fija y dibujarla en la pantalla.

Metodología del proyecto

UC+UP

El sistema digital se dividirá en tres módulos y los módulos se diseñarán mediante una unidad de control(UC) y una unidad de proceso(UP). La unidad de control controla el funcionamiento de todo el sistema y la unidad de proceso se encarga de ejecutar las señales de control.

Diseño VHDL

Tras diseñar los módulos del sistema digital se definirá cada uno editándolo mediante el lenguaje de descripción de hardware VHDL y para hacer la edición VHDL se usará la herramienta ModelSim.

Simulación

Después de completar la edición de los módulos se harán simulaciones para verificar que su comportamiento es el deseado mediante ModelSim dando valores a las entradas y analizando el cronograma resultante.

Implementación Hardware

Se hará la síntesis del sistema completo mediante Quartus y después se analizará si las desviaciones del comportamiento ideal, debidas a los retardos de los componentes no incumplen las restricciones temporales. Luego, se procederá a programar el FPGA.

Test del prototipo

Finalmente, se realizarán pruebas para comprobar el correcto funcionamiento del prototipo.

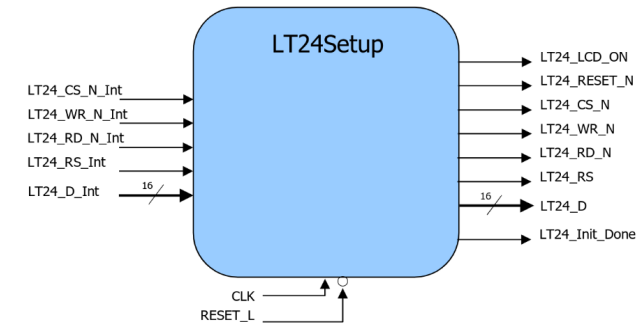
Herramientas de diseño, simulación y validación

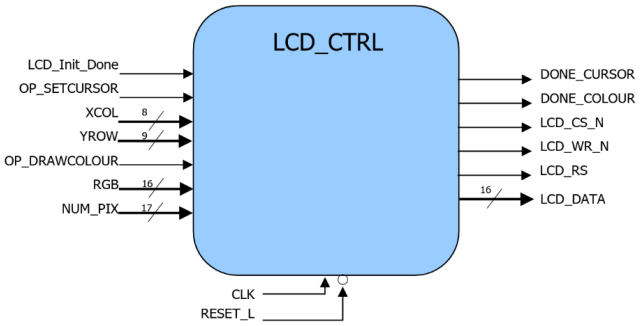
Edición VHDL y simulación: ModelSim
Síntesis y verificación: Quartus
Hardware de implementación: placa DE1-SoC (FPGA) y pantalla LT24

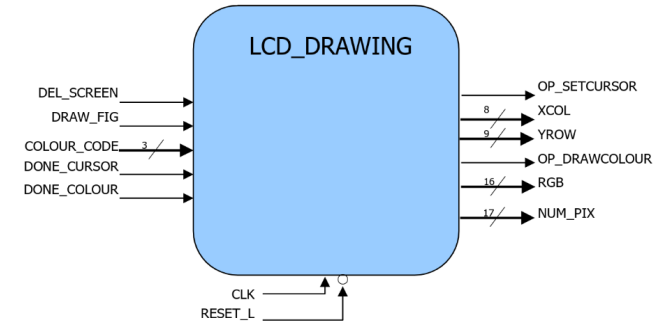
Diseño del sistema

Descripción general del sistema, división en módulos

EL sistema se divide en tres módulos: LT24Setup, LCD_CTRL y LCD_DRAWING. El módulo LT24Setup es el responsable de la inicialización de la pantalla LCD, el módulo LCD_CTRL hace las operaciones básicas sobre la pantalla y el módulo LCD_DRAWING borra la pantalla o dibuja una línea sobre la pantalla.







Descripción de cada módulo

LCD_CTRL

Entradas

- LCD_Init_Done: si su valor es 0, indica que la LCD está todavía en proceso de inicialización (el módulo LT24Setup controla esta señal) y no se puede realizar ninguna tarea. Cuando su valor es 1, el módulo LCD_CTRL podrá realizar las

operaciones básicas.

- OP_SETCURSOR, XCOL e YROW: las tres entradas están relacionadas, la primera indica que se quiere realizar la operación de posicionamiento del cursor, y las dos siguientes indican la posición concreta. El valor de XCOL estará en el rango 0-239 (8 bits) y el de YROW entre 0-319 (9 bits).
- OP_DRAWCOLOUR, RGB y NUM_PIX: las tres entradas están relacionadas, la primera indica que se quiere realizar la operación de colorear pixels, y las dos siguientes indican el color (entrada RGB) y el número de pixels a colorear (entrada NUM_PIX).

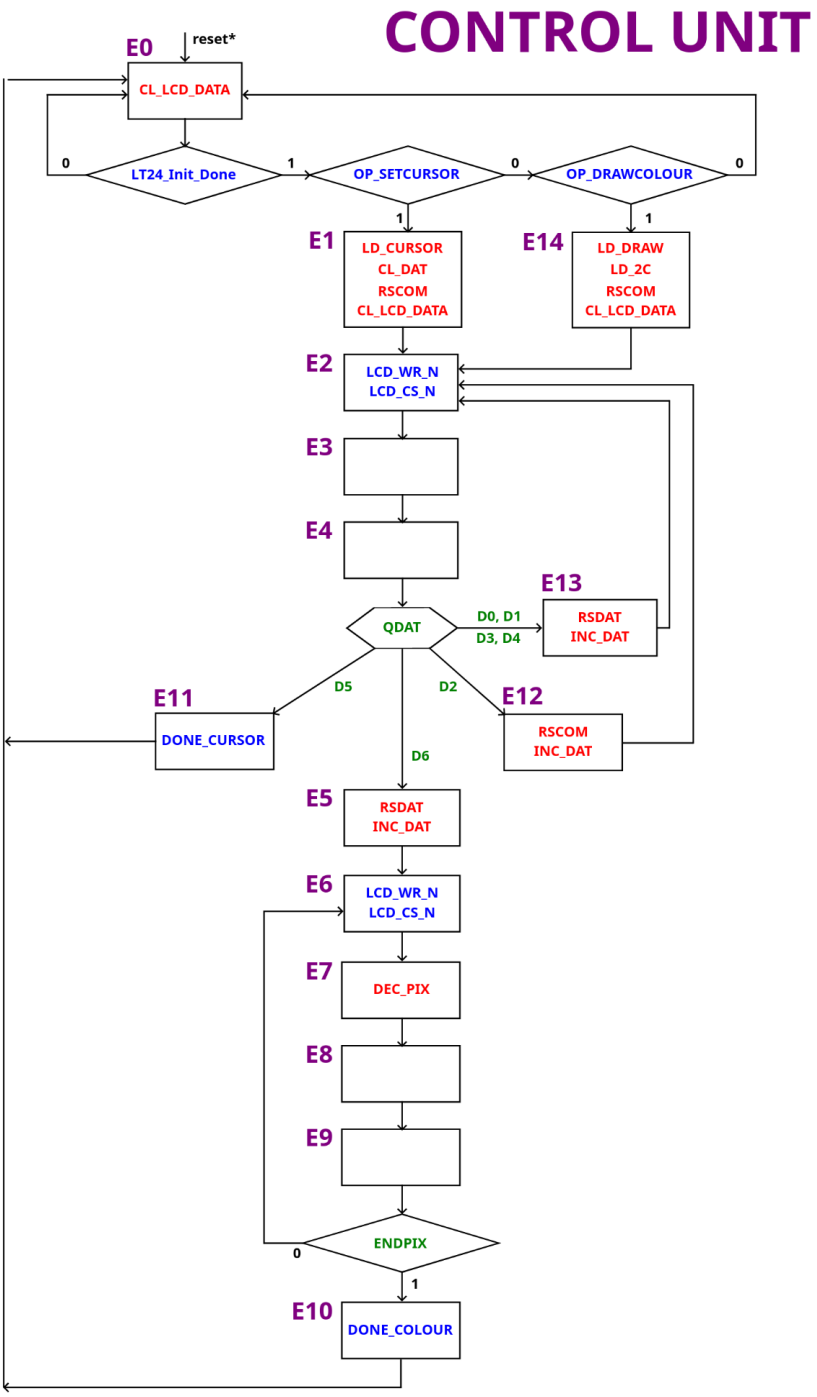
Salidas

- DONE_CURSOR: indica que ha finalizado la operación de posicionamiento del cursor.
- DONE_COLOUR: indica que ha finalizado la operación de dibujar pixels.
- LCD_CS_N, LCD_WR_N, LCD_RS y LCD_DATA: tanto para realizar la operación de posicionamiento como para la de dibujar, hay que enviar una secuencia concreta de información por esas salidas. Los detalles de esa secuencia están en la guía rápida de la pantalla LT24, y la información recogida en los apartados siguientes se ha obtenido de ahí, puesto que esa información es imprescindible para realizar y entender el diseño del módulo LCD_CTRL.

Funcionalidad

La finalidad de este módulo es la implementación de las operaciones básicas sobre la pantalla:

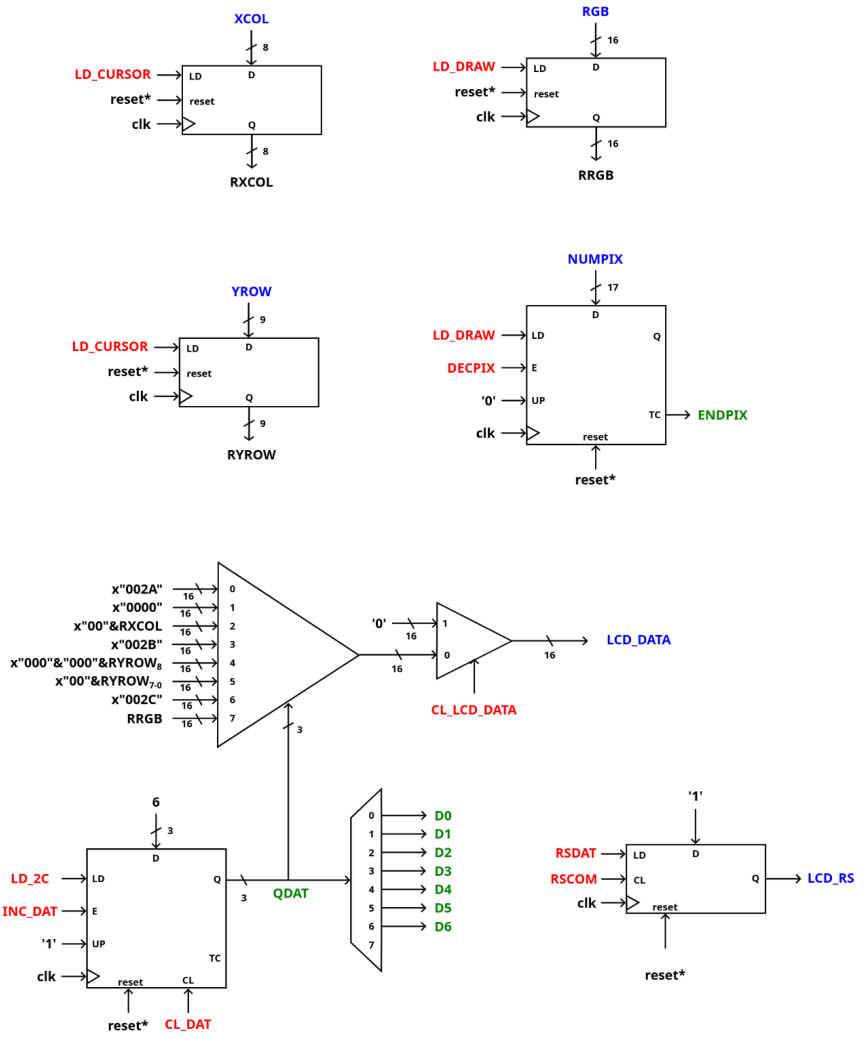
- Colocar el cursor en la posición X, Y
- dibujar uno (o más) píxeles desde la posición actual.



- En el estado E0 se asegura que la salida LCD_DATA está a 0 (CL_LCD_DATA activado) y se comprueba si la pantalla LCD ya ha sido inicializada (LT24_Init_Done). Si todavía no está inicializada no se analiza nada más. Si por el contrario ya está inicializada, se analiza si alguna de las operaciones básicas está siendo solicitada: señales OP_SETCURSOR y OP_DRAWCOLOUR. Si la petición de posicionamiento del curso está activada se continúa en el estado E1, y si la señal de dibujar pixels está activada se continúa en el estado E14.

- Operación de posicionamiento del cursor. Para posicionar el cursor hay que controlar el valor de las salidas LCD_WR_N, LCD_CS_N, LCD_RS y LCD_DATA. Las dos primeras las va a generar directamente la unidad de control, pero las otras vendrán del registro de 1 bit y de los multiplexores (ver esquema). La descripción de lo que hay que ir enviando por la salida LCD_DATA para posicionar el cursor está bien detallada en el apartado 1.1 y se corresponde con las 6 primeras entradas del multiplexor. Por lo tanto, después de realizar las inicializaciones necesarias en el estado E1, utilizaremos los estados E2, E3, E4, E11, E12 y E13 para realizar esos envíos mediante un bucle. Los tres primeros estados son iguales para todos los envíos, pero dependiendo del dato enviado, es decir, del analizando el valor de QDAT, el siguiente estado podrá ser E13, E12 o E11. El estado E11 es el último estado del bucle. A él se va tras haber realizado toda la secuencia de envíos , y por ello activamos en él la señal DONE_CURSOR. La diferencia entre E13 y E12 está en el control de la señal LCD_RS, ya que no tiene el mismo valor en todos los envíos (al enviar 2A o 2B su valor es L, pero al enviar los datos de esos comandos, debe valer H).
- Operación de dibujar pixels. Igual que para el cursor, para dibujar pixels hay que controlar el valor de las salidas LCD_WR_N, LCD_CS_N, LCD_RS y LCD_DATA. Sin embargo, en este caso la secuencia a enviar es más simple (véase el apartado 1.2). Una vez realizadas las las inicializaciones necesarias en el estado E14, se enviará el comando 2C mediante los estados E2, E3, E4 y E5, y después, dentro de un bucle, se utilizarán los estados E6, E7, E8 y E9 para dibujar el número de pixels solicitado. Para controlar el bucle se utiliza el contador descendente, que se va decrementando en cada iteración (DEC_PIX). Al llegar a 0 (ENDPIX) el bucle finaliza y se va al estado E10 en el que se activa la señal DONE_COLOUR.

PROCESS UNIT



Componentes:

- 3 registros para guardar la información de entrada: XCOL, YROW y RGB.
- Un contador descendente para guardar la entrada NUMPIX. Lo decrementaremos con cada envío del color RGB para saber cuándo se ha finalizado el proceso de dibujar NUMPIX pixels.
- Dos multiplexores y un contador para conseguir el valor adecuado en cada momento en la salida LCD_DATA . El multiplexor de dos entradas lo utilizamos para poner en la salida o bien el valor 0 o bien el valor que viene de otro multiplexor de 8 entradas. En ese segundo multiplexor tenemos todas las opciones de envío, ya que en las entradas 0-5 se dispone de forma ordenada de la información que hay que enviar para posicionar el cursor (los dos comandos y los parámetros de cada uno); en las

entradas 6-7, la información ordenada que hay que enviar para dibujar pixels (comando y parámetro). El orden es importante, ya que nos permite hacer la selección en el multiplexor a través de un contador: iremos incrementando el contador para ir cambiando la información en el momento adecuado.

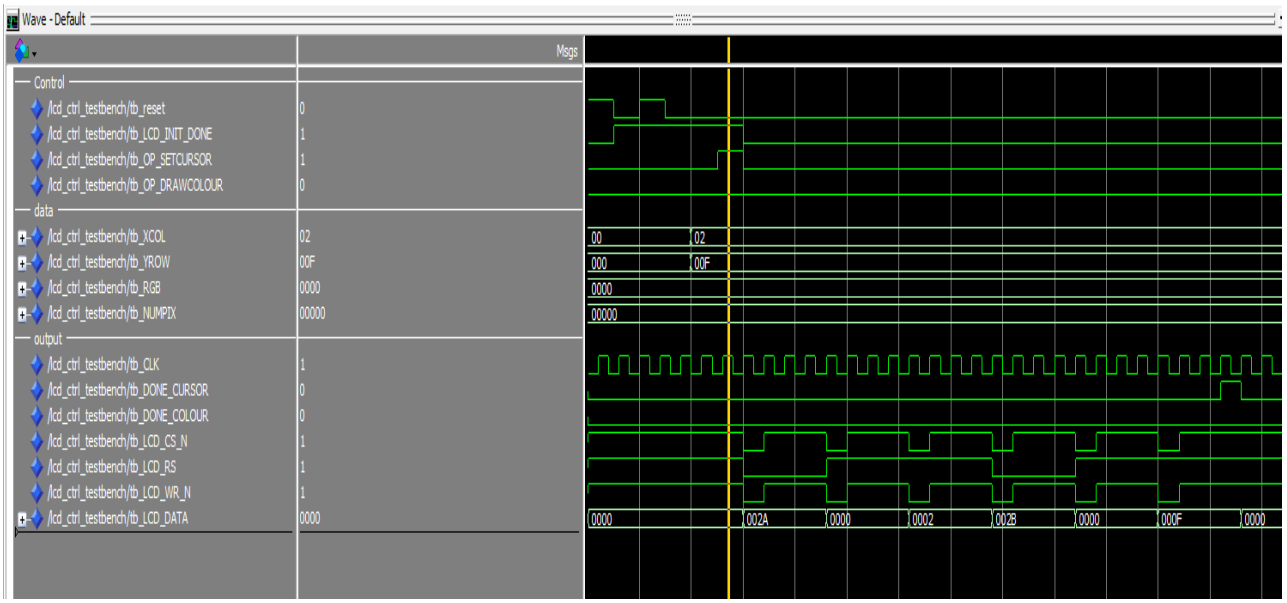
- Un descodificador para analizar el valor del contador, ya que según sea el valor se tomarán varias decisiones en el algoritmo de control: continuar o no con los envíos, activar o no la señal RS etc.
- Un registro de 1 bit para determinar el valor de RS.

descripción

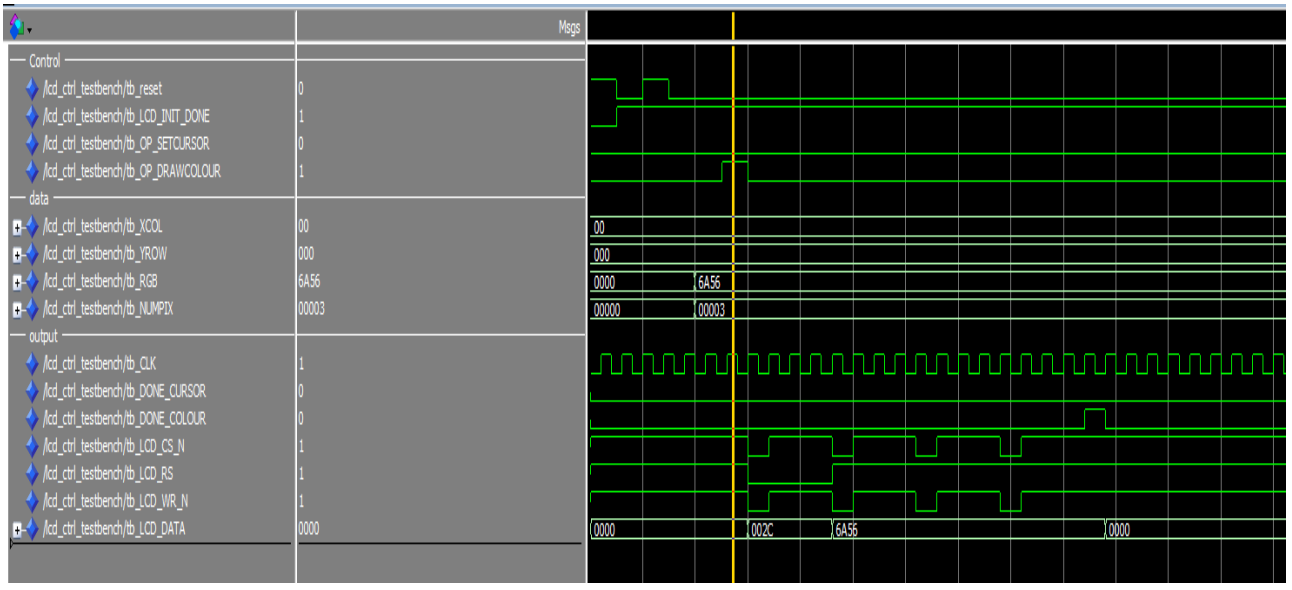
VHDL

Simulación

SETCURSOR:
En la siguiente imagen se puede ver el funcionamiento de LCD_CTRL cuando se prueba la opción "OP_SETCURSOR".
Se empieza reseteando la máquina (reset 1) y activando a la vez LCD_INIT_DONE y OP_DRAWCOLOUR, pasandole también datos por las señales XCOY e YROW. Como esperábamos, pasa la señal 2A seguido por los datos relevantes, que en este caso es la señal 0002, posteriormente se manda la señal de control 2B y la otra señal relevante 000F. Una vez terminado la operación se activa la señal DONE_CURSOR y se vuelve al estado inicial.



DRAWCOLOUR:
En la siguiente imagen se puede ver el funcionamiento de LCD_CTRL cuando se prueba la opción "OP_DRAWCOLOUR". Se reseteara la maquina y después se activará OP_DRAWCOLOUR Y LCD_INIT_DONE para entrar en ese modo. En la señal data se mandará la señal 2C para indicar el funcionamiento y se enviará la señal cargada por RGB por data un total de 3 veces(indicado por la señal WR_N). Se manda ese número de veces porque es el indicado por la señal de entrada NUMPIX. Para finalizar se activará la señal DONECOLOR para indicar el final de la operación.



LCD_DRAWING

Entradas

- DEL_SCREEN: Indica que se quiere realizar la operación de borrar la pantalla.
- DRAW_FIG: Indica que se quiere realizar la operación de dibujar.

- COLOUR_CODE: Indica el color con el que se quiere colorear los pixels.
- DONE_CURSOR: Indica que ha finalizado la colocación del cursor.
- DONE_COLOUR: Indica que ha finalizado la operación de dibujar pixels

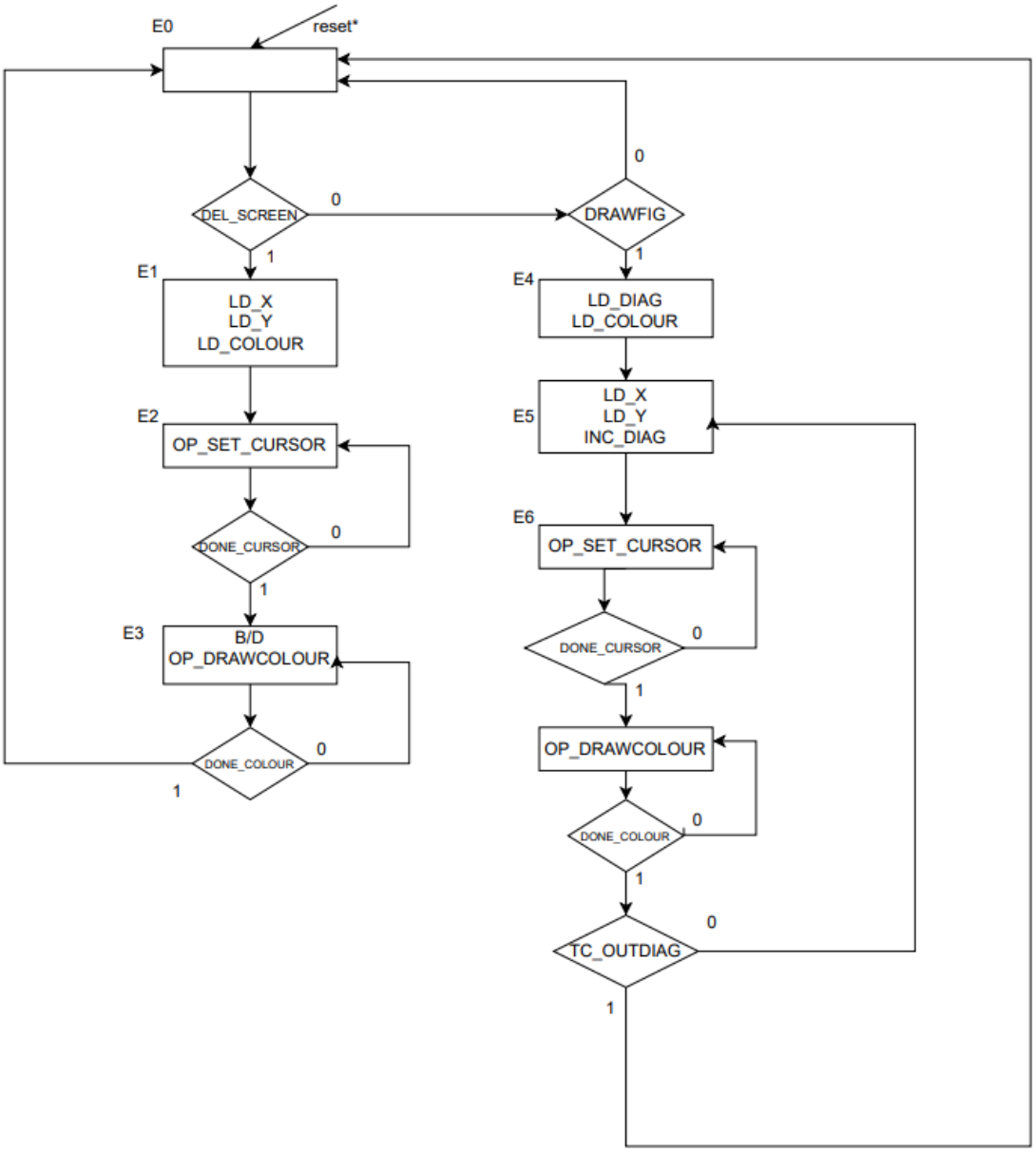
Salidas

- OP_SETCURSOR, XCOL e YROW: las tres salidas están relacionadas, la primera indica que se quiere realizar la operación de posicionamiento del cursor, y las dos siguientes indican la posición concreta. El valor de XCOL estará en el rango 0-239 (8 bits) y el de YROW entre 0-319 (9 bits)
- OP_DRAWCOLOUR, RGB y NUM_PIX: las tres salidas están relacionadas, la primera indica que se quiere realizar la operación de colorear pixels, y las dos siguientes indican el color (entrada RGB) y el número de pixels a colorear (entrada NUM_PIX).

Funcionalidad

La funcionalidad de este módulo es borrar la pantalla o dibujar una linea sobre la pantalla.

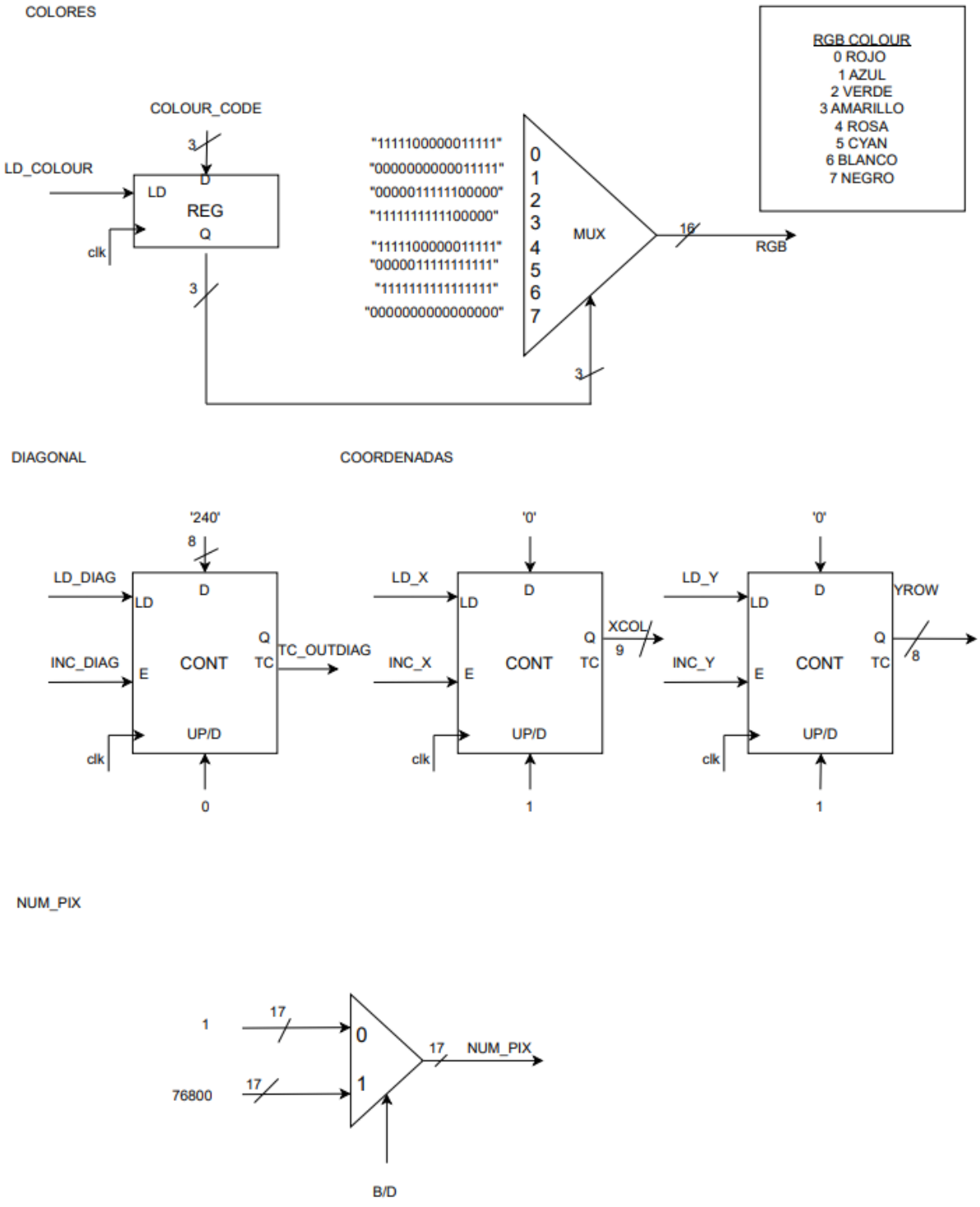
UC+UP



- En el estado E0 se analiza si alguna de las operaciones está siendo solicitada: señales DEL_SCREEN y DRAWFIG. Si la petición de borrado de pantalla está activada se continúa en el estado E1, y si la señal de dibujar está activada se continúa en el estado E4.
- Operación de borrado de pantalla: Para borrar la pantalla hay que posicionar el cursor en la posición 0,0 y luego pintar todos los pixeles de la pantalla de negro. Para posicionar el cursor hay que controlar las salidas OP_SETCURSOR, XCOL e YROW. La primera la genera la unidad de control y las otras vienen de dos contadores. Utilizamos el estado E1 para generar las señales XCOL e YROW y en el estado E2 se envía la señal OP_SETCURSOR.

Para dibujar los píxeles a negro hay que controlar las salidas OP_DRAWCOLOUR, RGB y NUM_PIX. La primera la genera la unidad de control y las otras vienen de los dos multiplexores. Se utiliza el estado E1 para generar la señal RGB y el estado E3 para las señales OP_DRAWCOLOUR y NUM_PIX.

- Operación dibujar en la pantalla: Para dibujar una línea en la pantalla hay que posicionar el cursor controlando las señales OP_SETCURSOR, XCOL e YROW y luego dibujar un pixel con las señales OP_DRAWCOLOUR, RGB y NUM_PIX. Esto se hace, dentro de un bucle, mediante los estados E5, E6 y E7 y se controla mediante un contador descendente. Al llegar a 0 se finaliza y se va al estado E0.



Componentes:

- Un registro y un multiplexor con las opciones de colores para guardar la entrada COLOUR_CODE y poner en la salida RGB el valor correspondiente.
- Dos contadores para conseguir y controlar las salidas XCOL e YROW.
- Un multiplexor para conseguir el número de píxeles adecuado en la salida NUM_PIX en cada momento, 1 o 76800.
- Un contador descendente para saber cuando el valor de XCOL =240.

descripción VHDL

Simulación