Proyecto 2: Controlador de Vídeo
Trabajo Realizado por:
Arnau Mora Gras & Carlos Villena Jiménez
amorgra@teleco.upv.es & cviljim@teleco.upv.es

Índice

Introducción

Para este trabajo, se ha trabajado mediante la plataforma GitHub, que nos permite trabajar en grupo, y compartir una única base de código, lo que facilita la coordinación y evita que haya múltiples códigos en circulación, y después haya problemas para cuadrar el trabajo.

Se puede consultar el repositorio en el siguiente enlace:

Github: https://github.com/SDP-2023/Proyecto2

Estructura del proyecto

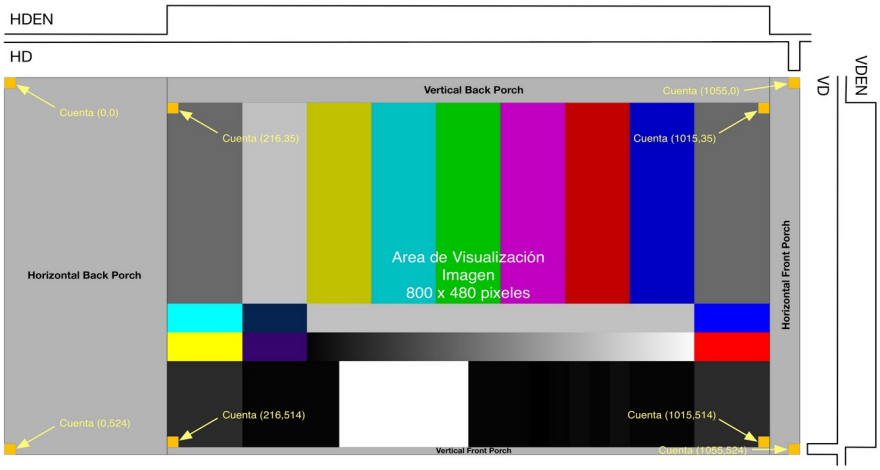
Para mantener el orden del proyecto, lo hemos mantenido en todo momento dividido en diferentes subdirectorios. Los principales son:

- dos: Contiene algunos scripts .do para correr en Questa/ModelSim.
- hex/mif: Contienen los archivos para cargar en las memorias ROM.

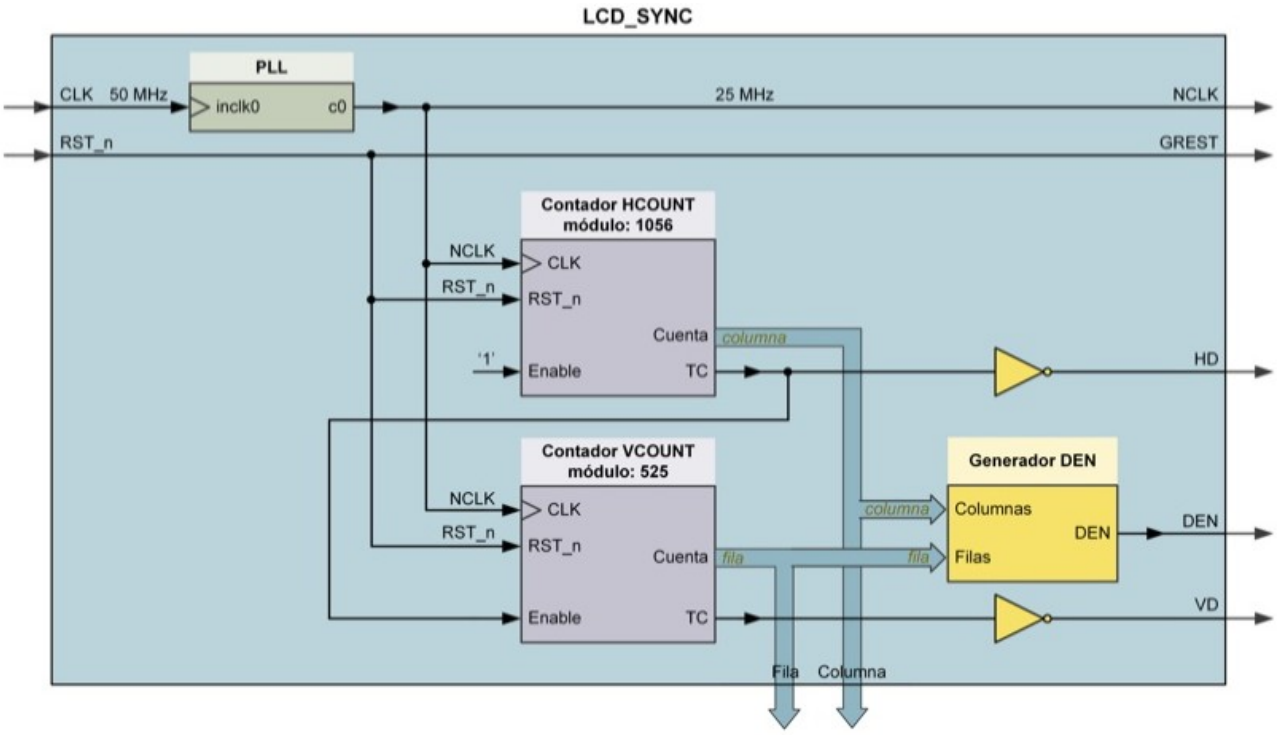
- img/img-pdf: Contiene imágenes del proyecto, la mayoría usadas para esta misma memoria.
- modulos: Contiene los módulos de Verilog necesarios para el proyecto.

Apartado 1: Señales de Sincronismo

En este apartado se pretende crear todos los módulos que generarán las señales que nos permiten controlar la pantalla. Las señales tienen unos márgenes llamados front-porch y back-porch en los cuales no se muestra nada aún, sólo sirven para configurar algunos parámetros en caso de ser necesario. En nuestro caso, para la pantalla de 800x480 pixeles, necesitamos un back-porch de 216 pixeles para X y 35 para Y; y un front-porch de 40 pixeles para X y 10 para Y. Esta imagen lo ilustra a la perfección:

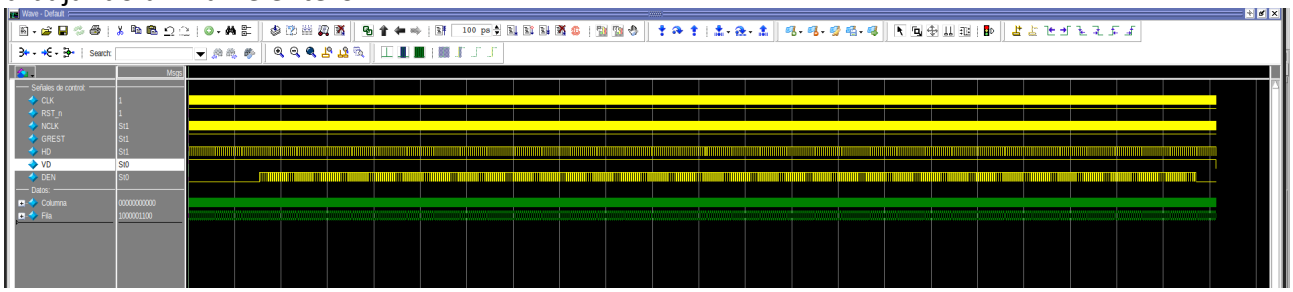


Para saber cuándo dibujar y cuando no, tenemos las señales HDEN (para X) y VDEN (para Y), que estarán activas cuando se deba dibujar el contenido, y desactivadas cuando no. Además, las señales HD y VD nos indicarán cuando se haya llegado al final de la fila y columna respectivamente.
Con esto en mente, se procede a desarrollar el módulo LCD_SYNC, que a partir de una señal de reloj genera todas las señales necesarias, además de una cuenta de fila y columna. Para hacer esto, se estructura el módulo de la siguiente forma:



El módulo llamado PLL se genera mediante Quartus, y reduce la frecuencia del reloj a la deseada para el correcto funcionamiento de la pantalla. Después tenemos dos contadores: HCOUNT y VCOUNT; que cuentan los píxeles dibujados horizontal y verticalmente, siguiendo la cuenta con las salidas Columna y Fila. También tenemos aquí el generador DEN, que activará la salida DEN cuando se esté dibujando en pantalla.
Para los contadores se ha aprovechado el código de las anteriores prácticas, realizando las conexiones mostradas anteriormente. El generador DEN se ha desarrollado dentro del módulo GeneradorDen, que sencillamente activa la salida cuando Columna tenga un valor entre 216 y 1015, y Fila entre 35 y 514.

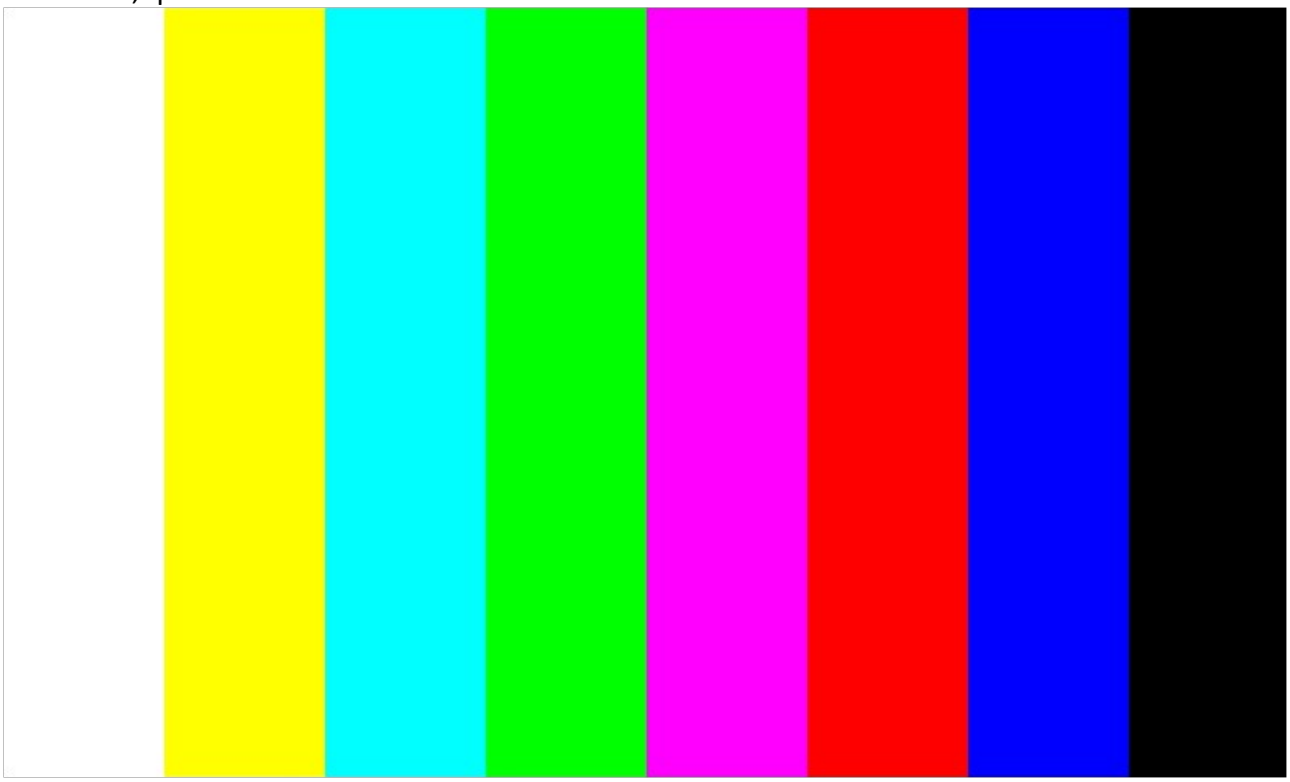
Este diseño se ha verificado, para comprobar que funciona correctamente. El testbench se encuentra en tb_lcd_sync.sv. Aquí dejamos unas capturas del módulo funcionando correctamente, dibujando un frame entero.



Apartado 2: Barras de Colores

El objetivo de este apartado es comprobar el correcto funcionamiento de los módulos anteriores. Para ello nos hemos ayudado del simulador de pantalla disponible en PoliformaT, que nos ha permitido saber que el módulo funciona de forma adecuada.
En este caso, hemos creado el módulo BARRAS_LCD, que toma como entradas la señal de reloj, que se conecta a LCD_SYNC creado anteriormente. La salida de Fila de éste último módulo es

comparada con distintos valores generados proceduralmente (para no tener que calcularlos manualmente), que se computan a partir de unos parámetros del módulo (configurados por defecto a los valores que nos interesan). Lo que se hace es, usando parámetros locales, se divide el ancho de la pantalla en 8 barras, y tomando en consideración los back-porch correspondientes, se generan las barras de los colores deseados.
Para comprobar que funciona correctamente, el testbench exporta un fichero vga.txt, que contiene todos los datos con marca de tiempo de los valores de las salidas, que después es importado al simulador, que renderiza la imagen. A continuación se muestra el resultado del simulador, que es correcto:



Para obtener la mezcla RGB de colores, se ha usado una rueda RGB, que nos permite saber qué mezclas hacer para cada uno de ellos:

La Simulación en placa FMPG-A de módulo de las barras quedaría como se aprecia en la siguiente imagen:

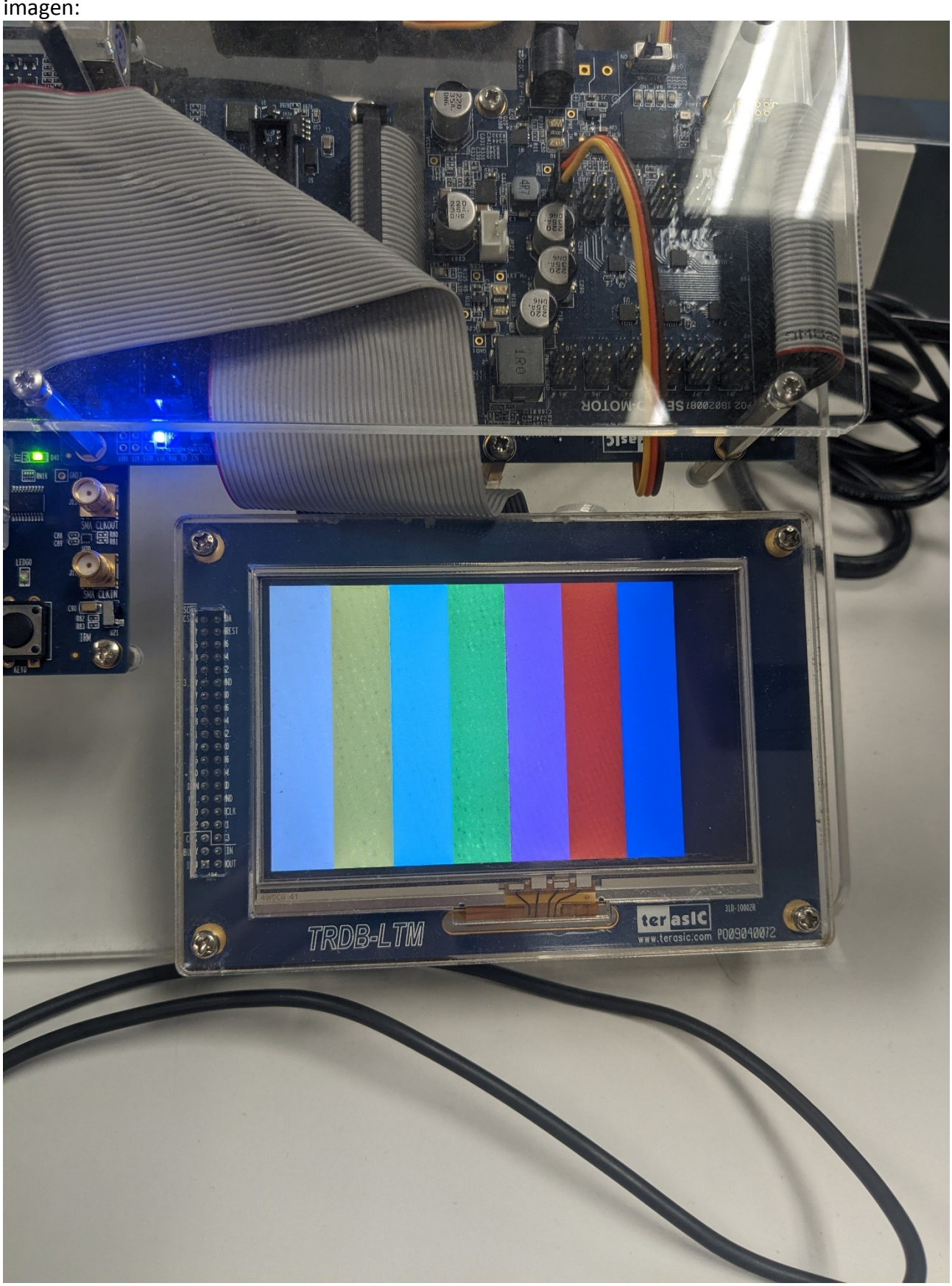


Fig.5. Imagen de la observación en placa del módulo BARRAS_LCD.v

Apartado 3: Imagen en pantalla

El objetivo de esta subtarea es la visualización en pantalla de una imagen o fotografía previamente almacenada en una memoria ROM interna.

Para ello se propone el siguiente diseño a realizar:

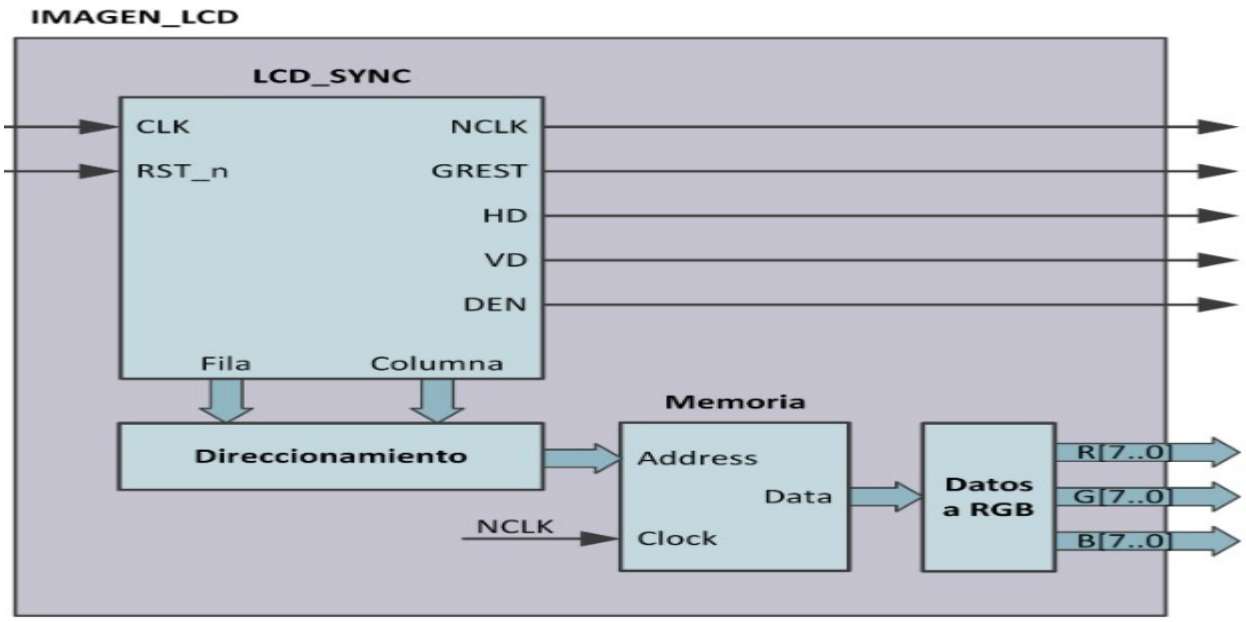


Fig.6. Estructura de módulos en bloques para la implementación de “IMAGENES_LCD.v”

Teniendo esto en claro, observamos según la imagen que debemos de instanciar el módulo previamente creado y mencionado en los apartados superiores (LCD_SYNC), después de este, los vectores de “Fila” y ”Columna” irán a un nuevo módulo que determina la dirección exacta de acceso a las posiciones de memoria según esos vectores que proporciona el bloque “LCD_SYNC”.

Justo después se busca en la memoria interna creada mediante el servicio que ofrece Quartus en IP-Catalog para guardar una imagen determinada (.hex) para así sacar los datos de color que tiene dicha imagen y poder plasmarla como un valor “RGB”.

La memoria en este caso es de cantidad de palabras “122768” para que así no quede recortada la imagen en ninguna posición.

En cuanto al tipo de direccionamiento seleccionado es el de tipo (X-Y) cuyo RTL queda así:

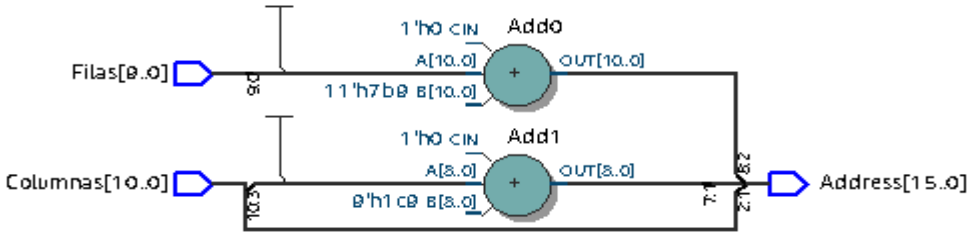


Fig.7. Diseño RTL del Módulo de Direccionamiento para Imagenes_lcd.v

En cuanto al cambio de parámetro de dato a dato RGB se realiza mediante un diseño muy sencillo de asignación de los tres bits más importantes de cada sección de color.

El diseño RTL quedaría tal que así:

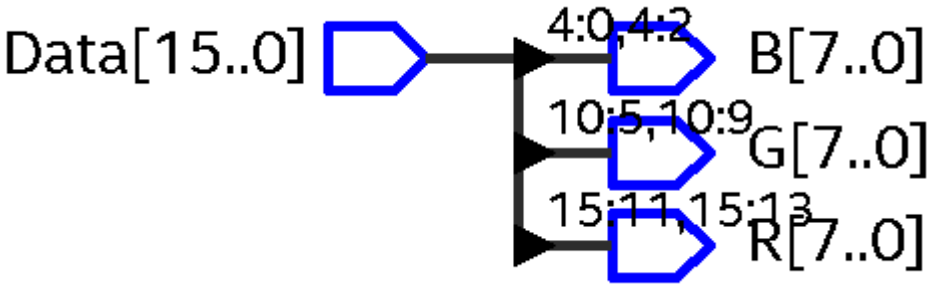


Fig.8. Diseño RTL del módulo Datos_a_RGB.v

En cuanto al conjunto del módulo para la visualización de la imagen sobre la placa FPG-A ó el simulador de pantallas ofrecido en recursos de PoliformaT se realiza exactamente con la instanciación de los cuatro módulos característicos de la Figura “5”.

En el documento (.qpf) se encontrará el diseño sobre verilog de todos y cada uno de los diseños, o incluso en el enlace al repositorio de GitHub.

En cuanto al Diseño RTL, quedaría tal que así:

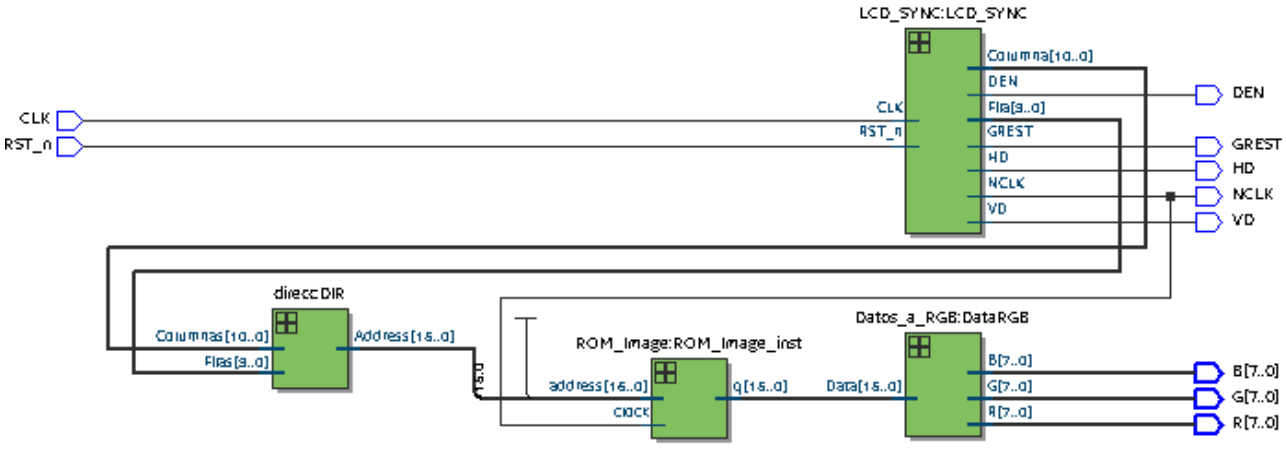


Fig.9. Diseño RTL del módulo IMÁGENES_LCD.v

En cuanto a la verificación sobre TestBench de este mismo módulo el resultado de la Simulación quedaría tal que así:

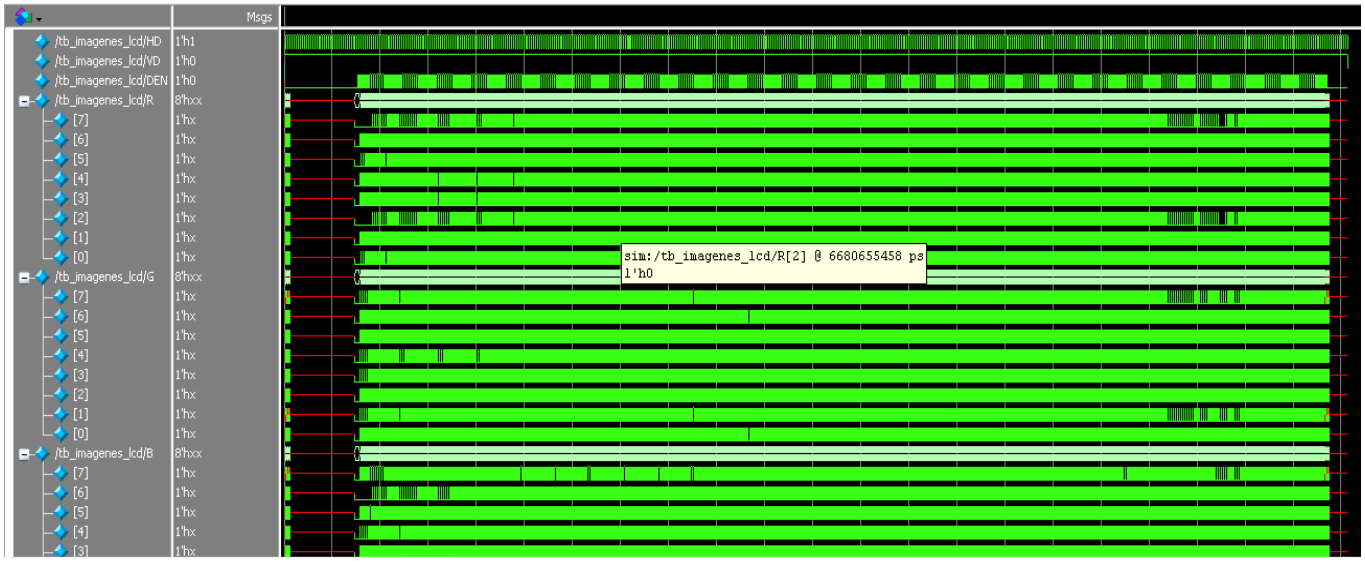


Fig.10. Simulación de la Verificación del módulo IMÁGENES_LCD.v (tb_imagenes_lcd.v)

En cuanto a la imagen obtenida en el simulador de pantalla del archivo “vga_imagenes.txt”

Quedaría tal que así:

Fig.11. Representación en simulador de pantalla del módulo TestBench de IMÁGENES_LCD.v

En cuanto a la simulación en la propia placa FPG-A del laboratorio, se observa así:

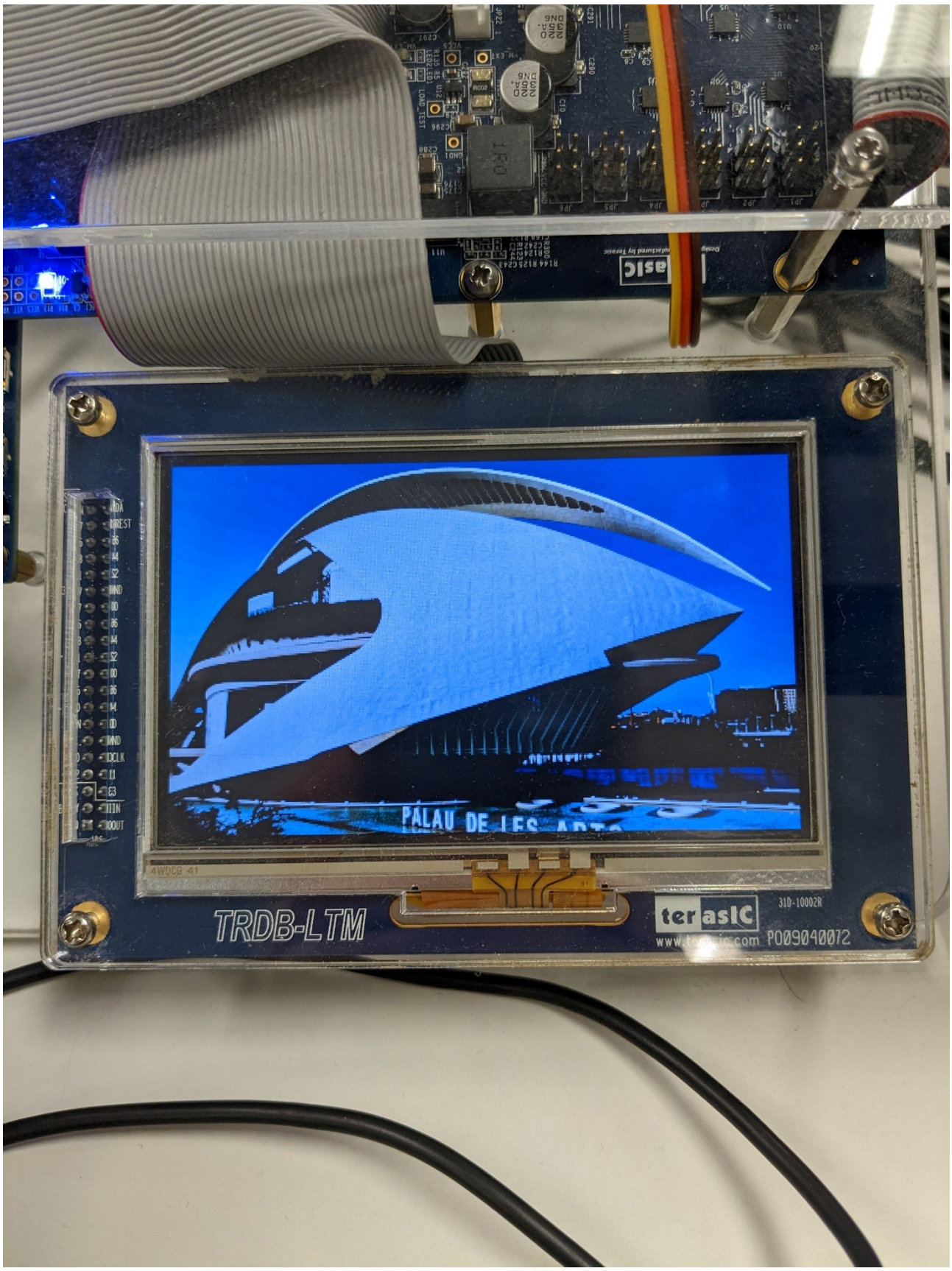


Fig.12. Simulación del módulo IMÁGENES_LCD.v en placa FPG-A

Apartado 4: Caracteres en Pantalla

El objetivo de este apartado consiste en visualizar en la pantalla caracteres, concretamente llenar toda la pantalla con caracteres iguales o diferentes.

Dado que la intención es visualizar en pantalla caracteres "0" habrá que tener en cuenta, que la pantalla está actualizándose continuamente, por ello es necesario el uso de una memoria interna para mantener los caracteres y que puedan ser mostrados en pantalla.
Esta memoria (CHAR_ROM.v) está también creada a partir de la aplicación de IP-Catalog de Quartus en donde cambia un poco sus especificaciones, concretamente con capacidad de 8192 palabras y 1 bit de tamaño como se muestra en la siguiente figura:

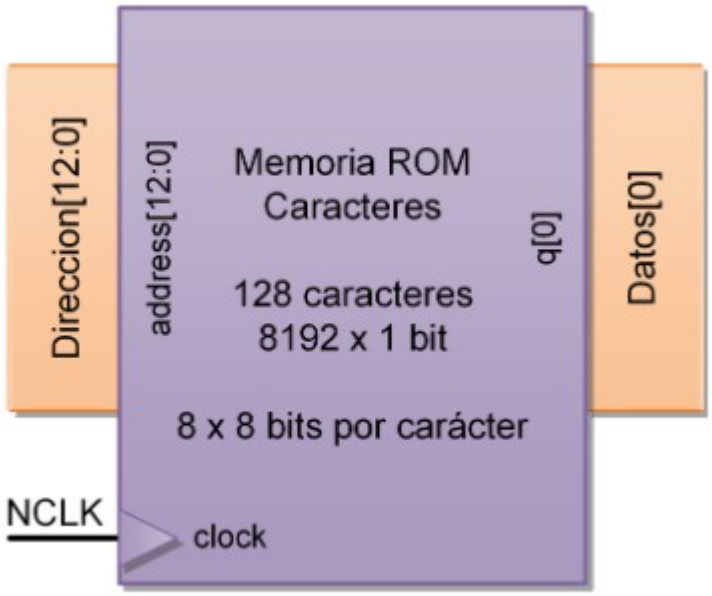


Fig.14. Presentación básica del bloque de la memoria del módulo CARACTERES_LCD.v

A partir de aquí se presenta la propuesta de diseño para la realización de este apartado:

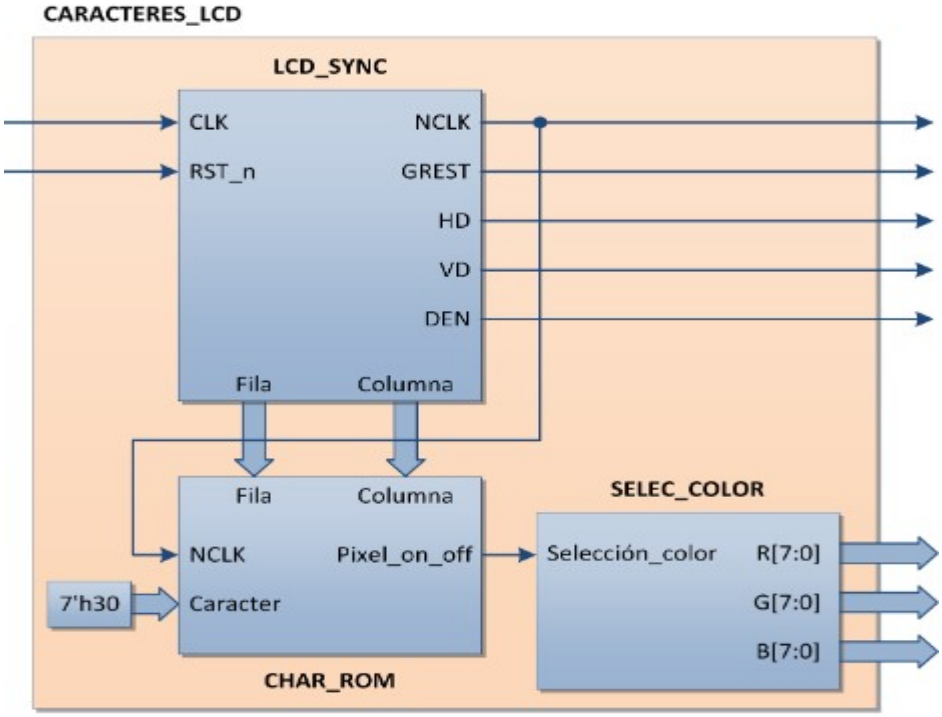


Fig.14. Diagrama de Bloques del Módulo CARACTERES_LCD.v

Donde podemos observar que vuelve a aparecer el módulo de sincronismos de señales de la pantalla (LCD_SYNC.v) de donde se cogerán nuevamente las señales de Fila y Columna, los cuales esta vez será divididos entre pasar por el módulo de direccionamiento reformado (ACORTADO EN TAMAÑO) y otras componentes de fila y columna pasaran como unos registros por el módulo de la nueva memoria ROM.

El bloque presentado por el guion de la práctica a seguir es tal que así:

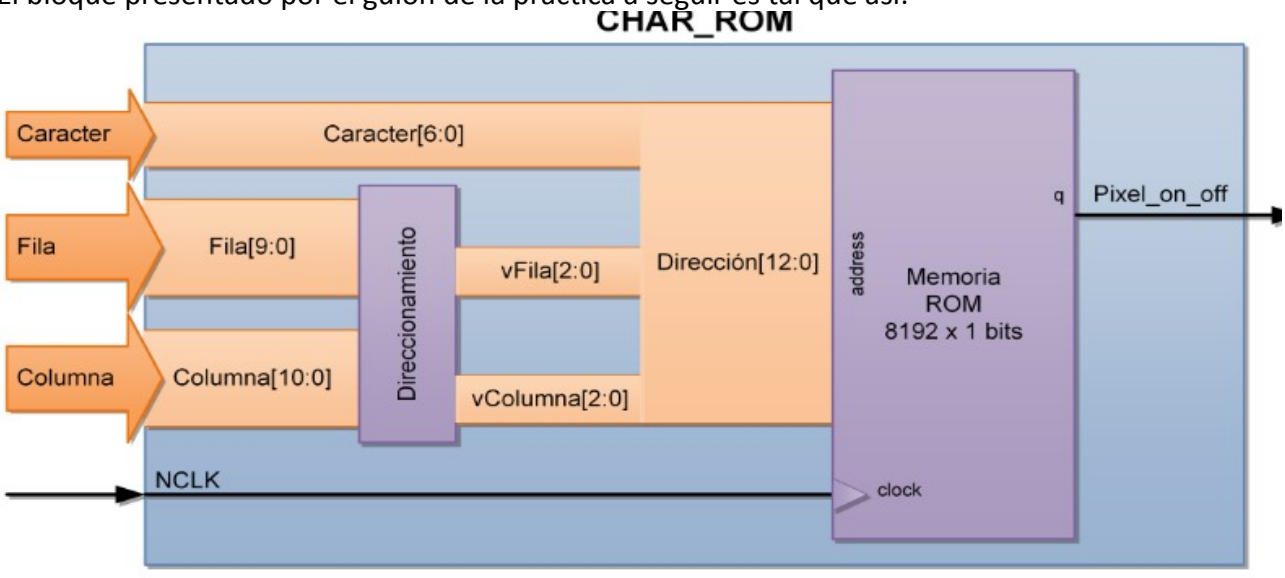


Fig.15. Diagramas de Bloques del módulo CHAR_ROM.v

En cuanto al módulo modificado de direccionamiento para este apartado, lo que se ha efectuado es un cambio de redimensionamiento en donde se acorta el tamaño de direcciones a 6 bits eligiéndose los tres más capables de las filas y los otros tres más capables de las columnas (Address = {m[2:0],n[2:0]}).

En cuanto a su diseño RTL quedaría tal que así :

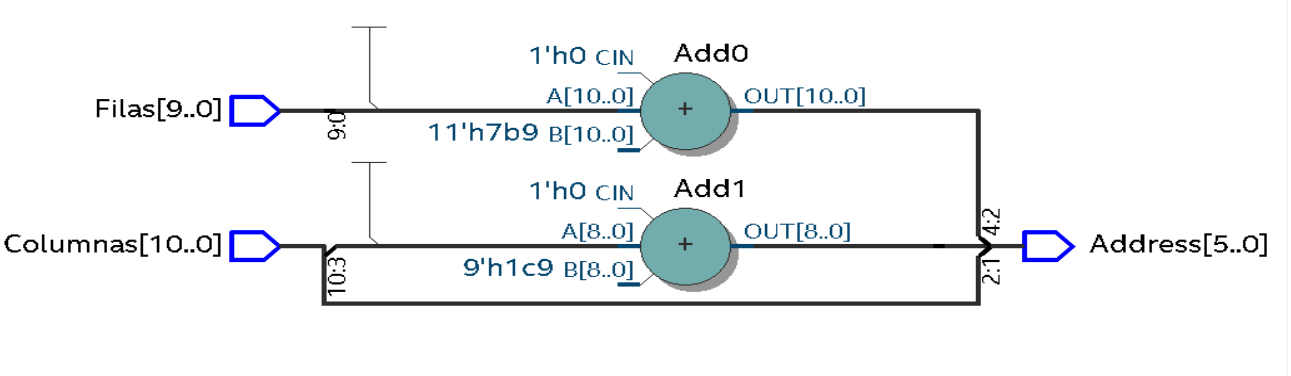


Fig.16. Diserño RTL del módulo direcc_texto.v usado en CARACTERES_LCD.v

En cuanto al diseño del módulo de CARACTERES_LCD.v se ha realizado como en el apartado anterior, es decir, mediante la instancia de los módulo previamente mencionados y realizados. Estos módulos son el de LCD_SYNC.v ; CHAR_ROM.v donde previamente se ha instanciado el módulo de direcc_texto.v y ROM_char.v ; SELEC_COLOR.v .

Su diseño RTL_viewer quedaría tal que así :

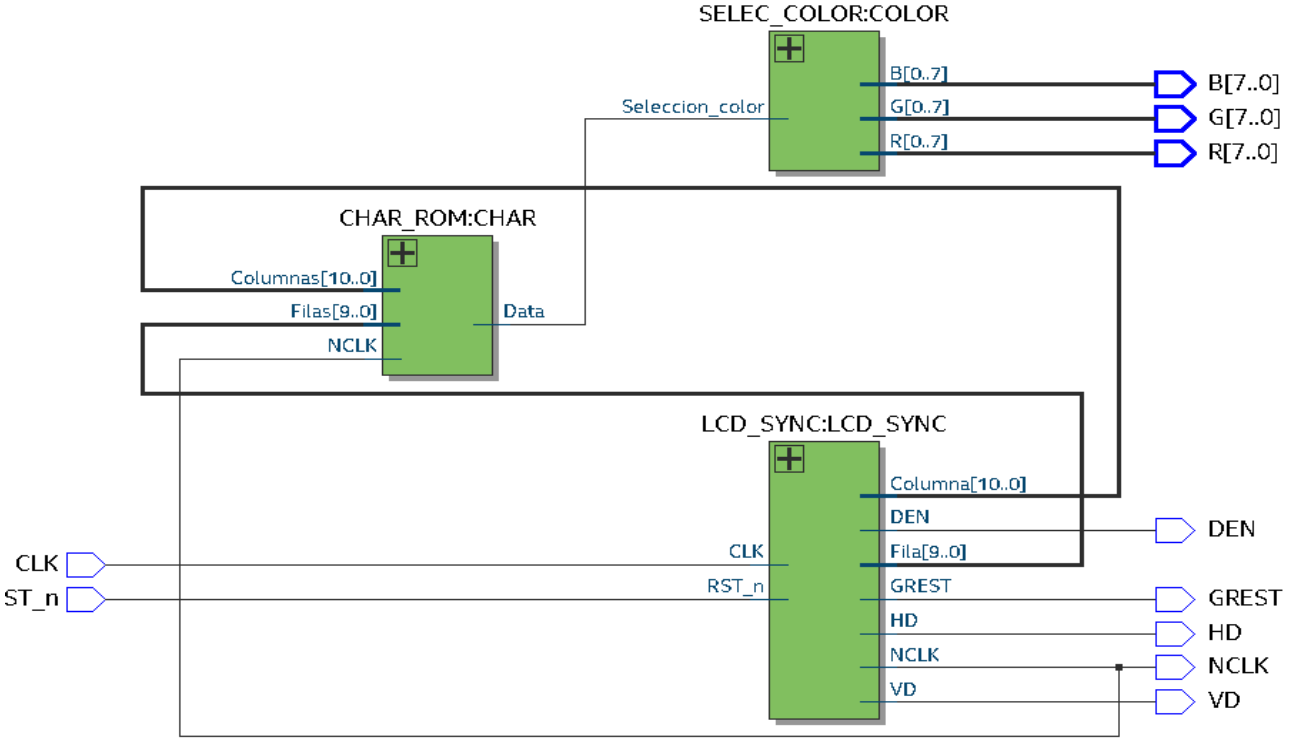


Fig.17. Diseño RTL del módulo de CARACTERES_LCD.v

Para terminar con este apartado se ha procedido con la implementación del testbench del mismo módulo (tc_caracteres_lcd.v) cuya simulación ha salido e forma satisfactoria.

Tal como se observa en la siguiente figura:

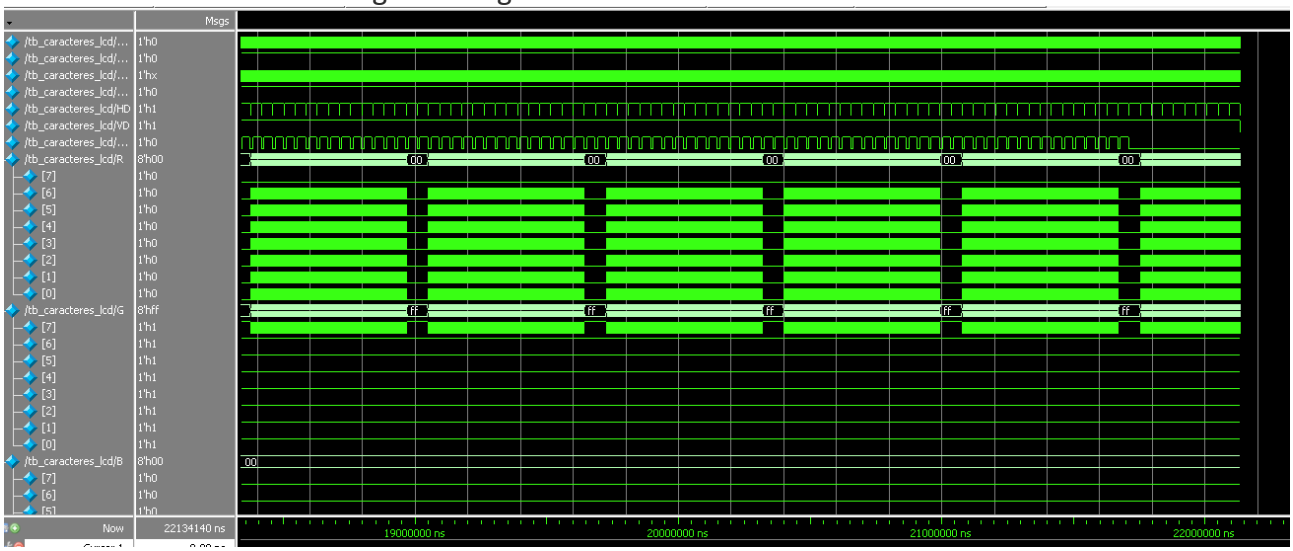


Fig.18. Simulación del testbench del módulo de CARACTERES_LCD.v

En cuanto a la visualización del archivo de texto generado vga_caracteres_lcd.txt se aprecia en el simulador de pantalla de PoliformaT de la siguiente forma:

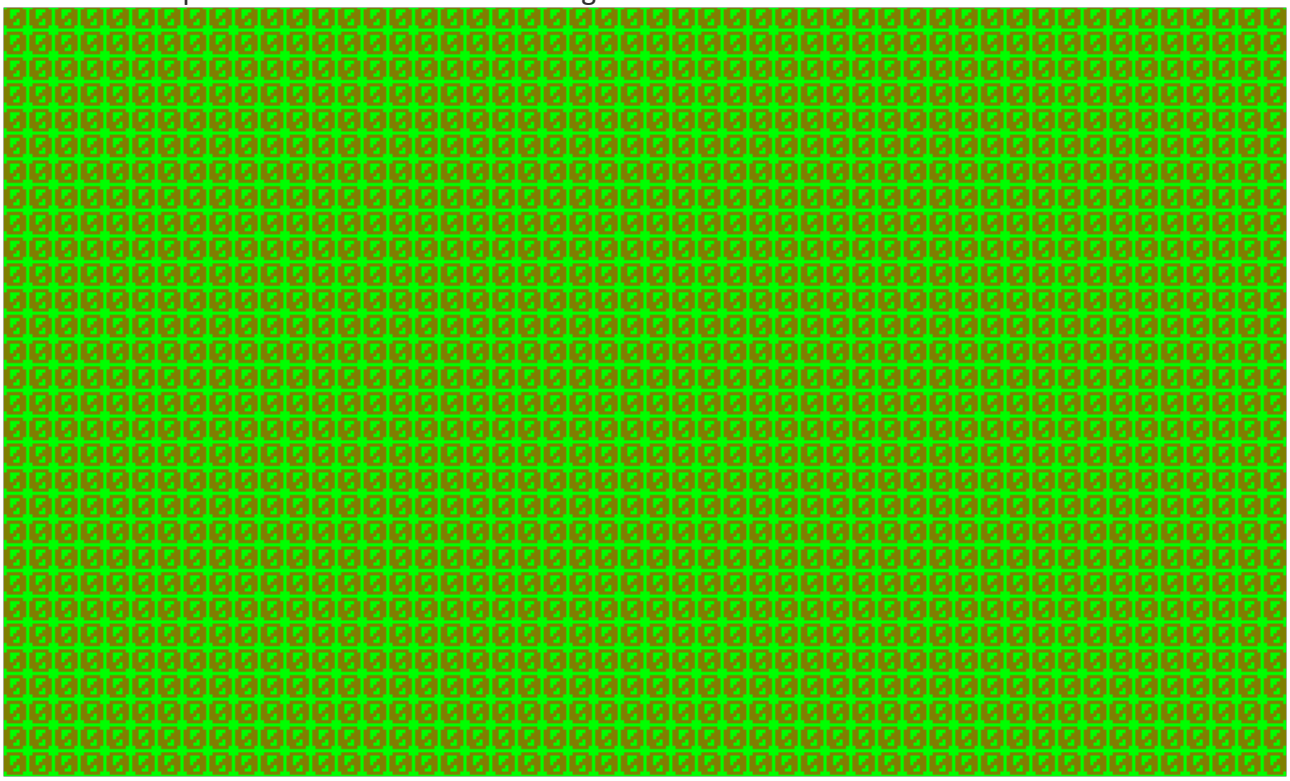


Fig.19. Visualización en pantalla de simulación del módulo CARACTERES_LCD.v

En cuanto a la visualización sobre la placa FPG-A se podría observar de la siguiente forma:

Apartado 5: Texto en Pantalla

En este apartado se quiere visualizar los nombres de los componentes del grupo de prácticas en la pantalla, en una línea bastante centrada.

La mejor opción es implementar una memoria de pantalla para el texto. Se trataría de utilizar la misma filosofía que en el apartado de visualización de la imagen, pero aplicada a caracteres en lugar de píxeles. A partir del apartado anterior, se añadiría una memoria que proporcione el valor del carácter que se quiere visualizar en cada instante.

En la siguiente imagen se observa el diagrama de bloques proporcionado por el guion:

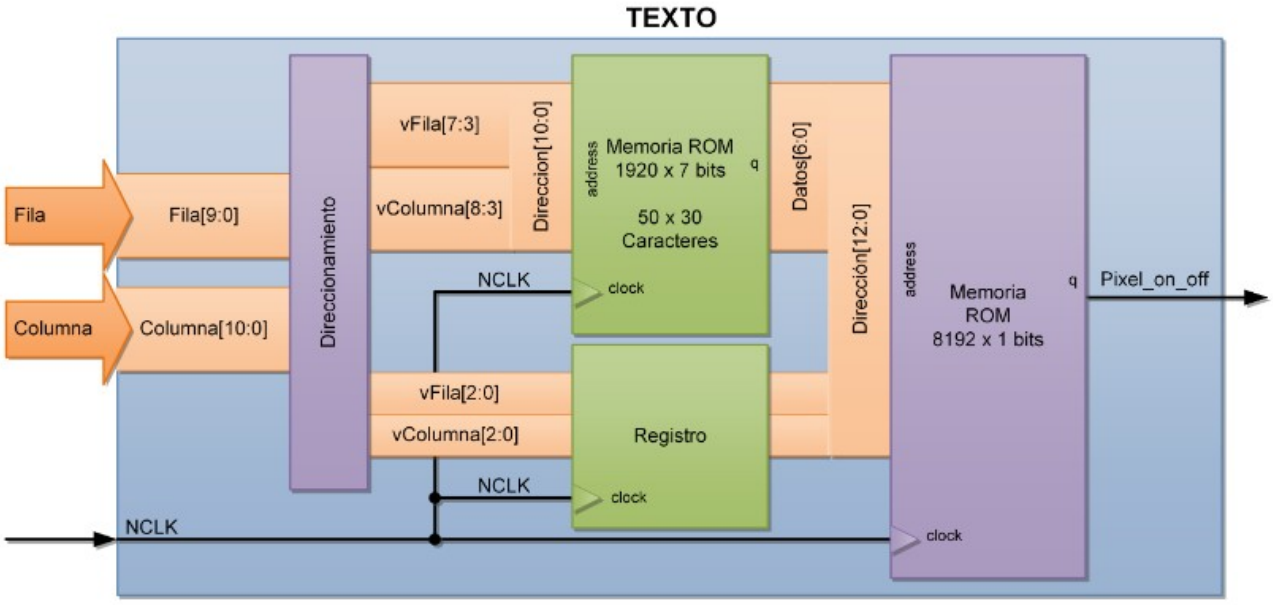


Fig.20. Diagrama de bloques para desarrollo del módulo TEXTO_LCD.v

En cuanto a la memoria interna desarrollada en esta subtarea se ha realizado nuevamente mediante la aplicación IP-Catalog de Quartus, pero con un tamaño de 2048 de tamaño de palabras en y 7 bits de palabra vez de 1920 y 7 ya que daba fallos a la hora de la simulación del testBench en cuanto a sobre paso de límite de la memoria.

En cuanto al direccionamiento usado, sería el mismo módulo usado en Apartado 3 (Fig.6.).

Para el desarrollo del módulo “TEXTO_LCD.v”, primeramente se produjo el diseño del módulo “TEXTO.v” en el cual se instancia las memorias ROM internas “ROM_CHAR.v” y “ROM_text.v” las cuales contienen lo que es la asignación de caracteres y y asignación de texto sobre caracteres respectivamente.

Como la primera memoria mencionada ya ha sido descrita sobre le apartado anterior (Apartado_4), comentaremos un poco sobre la del texto con los nombres “Arnau Mora Gras” y “Carlos Villena Jimenez”, los cuales pertenecen a los creadores de este mismo documento.

Lo que se ha cargado en la memoria rom es un archivo (.mif)(MapInfo Interchange Format) en donde se almacenan en unos espacios de memoria en concreto las letras de nuestro nombre

completo en formato Hexadecimal.

Este archivo instanciado en memoria lo hemos denominado de la siguiente forma:

“text_rom_arnaumoragras_carlosvillenajimenez.mif”, el cual ha sido modificado a partir del archivo (.mif) proporcionado por el guion.

El diseño RTL del módulo TEXTO.v quedaría tal que así:

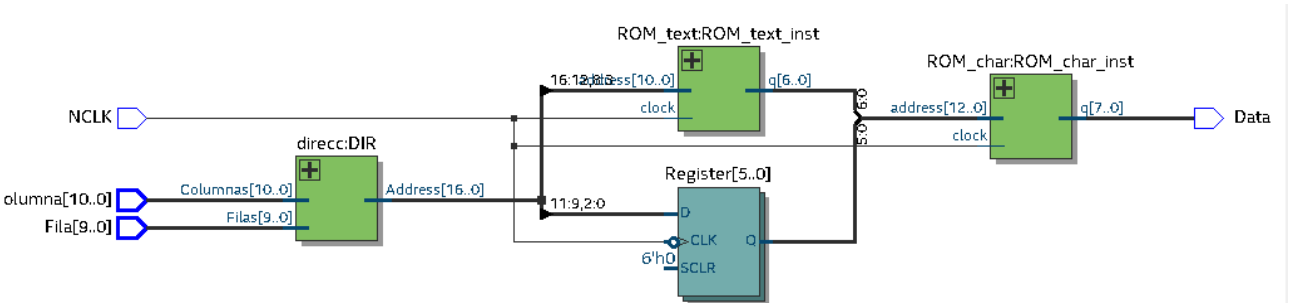


Fig.21. Diseño RTL del módulo TEXTO.v

Una vez realizado este módulo se continúa con el módulo final, cuyo RTL queda así:

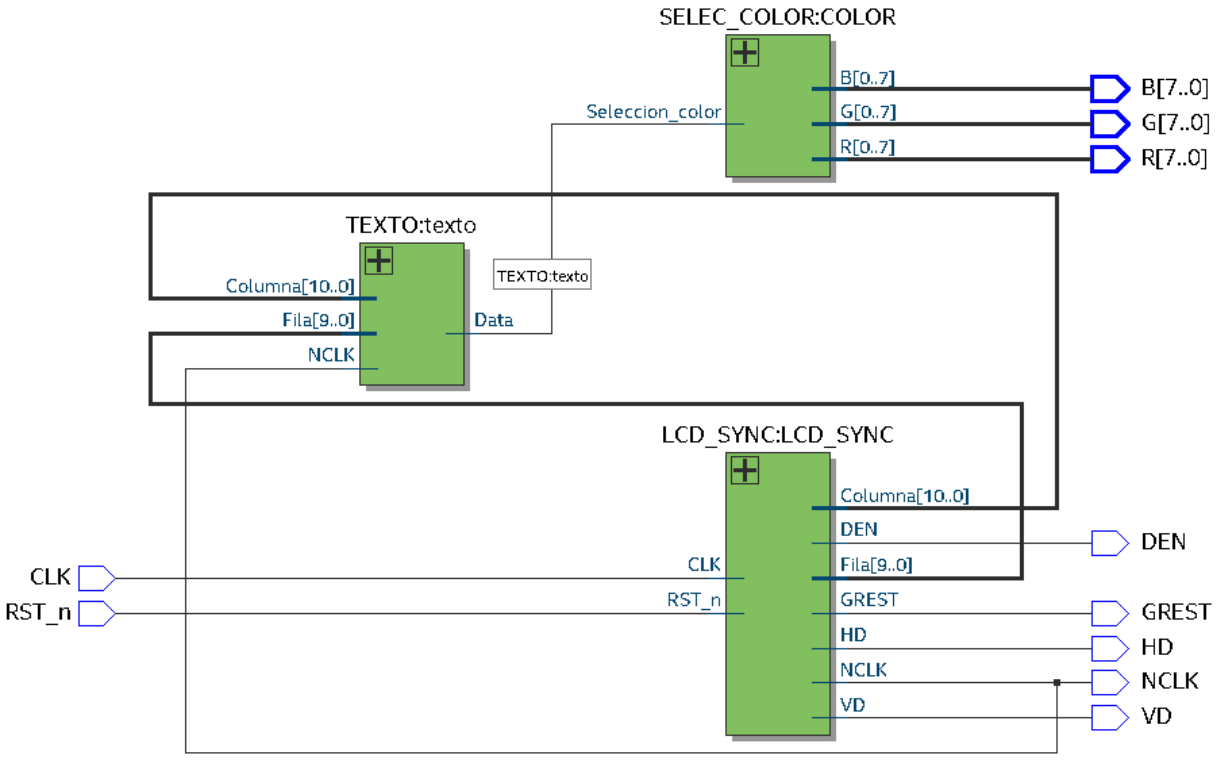


Fig.22. Diseño del módulo completo TEXTO_LCD.v

Una vez acabado todo esto y como se ha realizado en subtareas anteriores se propone con la implementación del módulo de testBench de este módulo (tb_texto_lcd.v) el cual se ha realizado casi que con los mismo parámetro e igual forma que todos los demás en este proyecto.

La simulación de dicha verificación se aprecia en la siguiente figura:

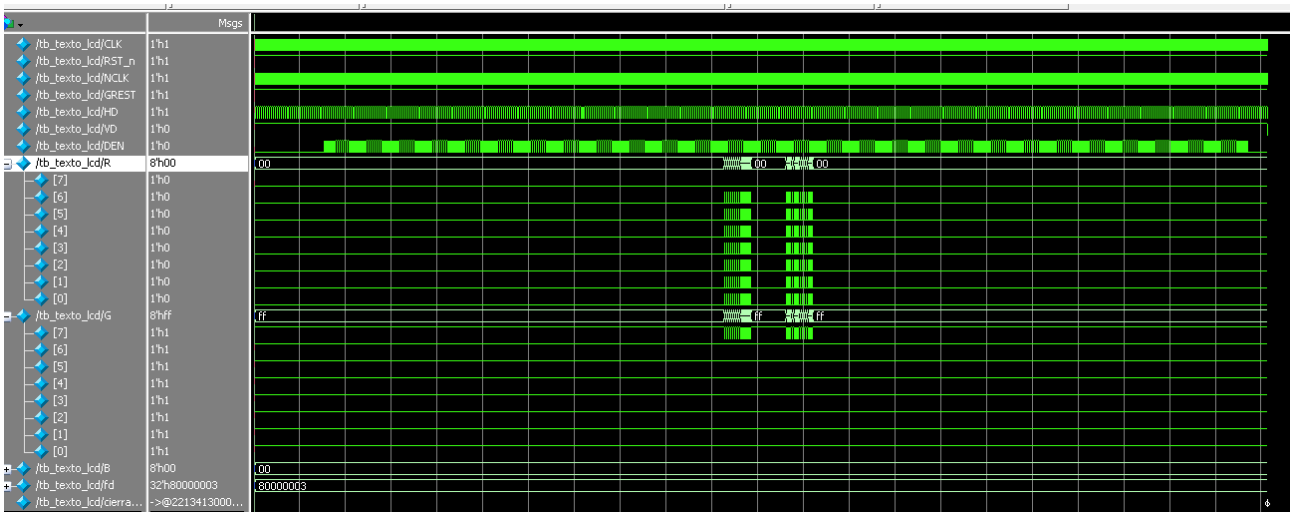


Fig.23. Simulación del TestBench del módulo TEXTO_LCD.v (tb_texto_lcd.v)

En cuanto a la simulación y visualización de la imagen con texto sobre la pantalla de simulación proporcionada por Poliformat quedaría tal que así:

Fig.24. Visualización de los nombres sobre la pantalla de simulación

En donde podemos observar un leve desplazamiento del nombre de “Carlos Villena Jimenez” sobre el nombre de “Arnau Mora Gras”, lo que querría decir que no se encuentra de forma centrada uno con respecto al otro.

En cuanto a la simulación de la imagen del módulo “TEXTO.LCD.v” sobre la placa FPG-A se puede observar en la siguiente imagen de abajo:

Problemas observados

En todo momento hemos intentado mantener la estructura del proyecto bien ordenada, con los archivos dentro de carpetas, y con nombres coherentes. No obstante, esto nos ha causado algunos problemas, que vale la pena puntualizar.

Quartus no se lleva muy bien con las simulaciones, y hemos detectado problemas al mantener archivos .mif o .hex en subdirectorios, ya que, al simular, se copian directamente a la raíz de la carpeta de simulación, en lugar de al subdirectorio en cuestión, lo cual causa que ModelSim no los encuentre. Para solucionar esto, hemos optado por copiarlos manualmente al subdirectorio una vez realizamos la simulación; aunque una solución perfectamente válida sería copiarlos al directorio base del proyecto, y actualizar la ROM correspondiente.

Conclusión

En este proyecto hemos aprendido cosas muy importantes sobre el funcionamiento de las señales digitales. Conceptos como los porch o señales de sincronización son muy importantes en los sistemas actuales de transmisión de imágenes. Además, se ha trabajado más en profundidad el uso de memorias ROM tanto para caracteres, como imágenes o textos, lo que nos brinda unos conocimientos sobre las aplicaciones de estas que sin duda serán útiles en el futuro.