ELO212: Laboratorio de Sistemas Digitales
Semestre I-2019
Guía de Actividades Sesión 9

1. Objetivos

Los objetivos de esta sesión son:

- Integrar y consolidar lo aprendido en sesiones anteriores aplicándolo a un problema de diseño semi-abierto.
- Experimentar con la configuración y uso de IP Cores para bloques de memoria y controladores de reloj.
- Reforzar los conceptos y experiencia ganada con reutilización de código propio y de terceros.
- Entender y aprender a lidiar con restricciones físicas de los dispositivos (recursos limitados, resoluciones de relojes, etc.), y tomar decisiones de diseño de manera acorde.
- Aprender a justificar y reportar decisiones de diseño ante requerimientos abiertos y/o que no sean factibles.

2. Introducción

Hasta ahora, en cada sesión de laboratorio se les ha entregado una guía de trabajo con indicaciones muy precisas sobre lo *que* debían hacer y el *como* debían hacerlo. Este enfoque no tenía por objetivo imponer arbitrariamente estilos del tipo "se hace así porque el profesor lo dice", sino que apuntaban a acotar el espacio de diseño y reducir la probabilidad de ocurrencia de problemas que podrían encontrar al no seguir convenciones, para así poner el foco en los conceptos fundamentales y enfatizar las buenas prácticas utilizadas en el diseño digital moderno con HDLs. Con este fin, tanto la estructura y el flujo de las experiencias, así como los archivos base y ejemplos provistos, fueron planeados y discutidos por el staff para ir cubriendo distintos conceptos de forma incremental, integrando y aplicando nuevos conceptos y técnicas a medida que fueran adquiriendo más experiencia y viendo nuevos contenidos en los cursos teóricos [1].

[1] Al menos esto fue la intención, basándonos fuertemente en nuestra propia experiencia. Sin embargo, una asignatura de este estilo siempre presenta desafíos, y creemos que siempre se puede mejorar. Cualquier comentario o feedback al respecto será muy útil, tanto para nosotros como para sus compañeros que tomarán el curso en las próximas iteraciones. El feedback no tiene ninguna bonificación o penalización, incluso si son quejas. Pueden plantear sus comentarios por medio de Piazza o conversando directamente con sus profesores, si lo prefieren incluso después que termine el curso y ya tengan su nota.

En esta última sesión de laboratorio se utilizará un método de trabajo distinto al de las sesiones anteriores, en el sentido de que solo recibirán requerimientos funcionales sobre un sistema a implementar, pero Uds. deberán decidir la estrategia de diseño y componentes a utilizar. Esto es representativo de situaciones que debe resolver un ingeniero en el mundo real, donde recibirán requirimientos funcionales de personas que no saben nada – y no tendrían por que saber– sobre la tecnología y el funcionamiento interno de un sistema (en este caso un sistema digital), y será trabajo de Uds. tomar las decisiones correspondientes para evaluar la factibilidad y los costos asociados a la implementación. En esta ocasión solo se entregarán especificaciones generales y archivos base mínimos, y Uds. deberán investigar, experimentar, tomar y justificar cualquier decisión para la implementación.

Si bien los problemas que se proponen son acotados y simples como para considerarse "de juguete", son representativos en el sentido que deberán identificar conceptos y componentes que se puedan reutilizar de las experiencias anteriores, extrapolar lo que han aprendido para utilizar técnicas avanzadas no vistas o discutidas en cátedras o sesiones, aprender a buscar información sobre una aplicación que desconocen, abstraerse de la descripción genérica y traspasarla al contexto de implementación digital [2], además de integrar especificaciones de software y hardware.

Haciendo la analogía usada en las primeras cátedras sobre aprender a andar en bicicleta, en las últimas experiencias les hemos ido sacando las rueditas de apoyo, y ahora ya deben conducir en una pendiente por si solos. Si bien es una pendiente suave y controlada como para un curso introductorio, es representativa y provee una buena base para seguir practicando. Para quienes quieran seguir explorando esta línea de trabajo, hay cursos avanzados en los que deberán mostrar sus capacidades para hacer piruetas [3].

3. Actividades asociadas a la sesión

3.1. Filtros digitales para procesamiento de imágenes

El diagrama de la Figura 1 muestra una representación de alto nivel de un sistema de procesamiento de imágenes que aplica filtros en cascada a una imagen recibida desde un sistema externo. El sistema almacena la imagen en una memoria interna con *configuración de doble puerta*, con puertos independientes para la lectura y escritura de datos (cada una con un reloj independiente). El puerto de escritura está conectado a una interfaz UART que recibe y almacena los pixeles de una imagen proveniente desde un sistema externo. El puerto de lectura es controlado mediante una interfaz VGA, la cual lee en forma continua el contenido del buffer para ser procesado por los filtros y mostrados en el display. La interfaz VGA muestra en todo momento el contenido del buffery los interpreta como pixeles, sin saber si el buffer tiene almacenada una imagen válida o no.

Se pide implementar el sistema de la Figura 1, de forma que permita almacenar, procesar y mostrar una imagen almacenada en el buffer en un display con resolución de 1024x768 pixeles a 75 fps. Asuma que el dispositivo externo envía los pixeles de la imagen codificados en 24 bits, utilizando 8 bits por cada canal de color.

La funcionalidad de cada uno de los filtros mostrados se describe en forma general a continuación (será parte de su trabajo averiguar detalles más técnicos y tomar decisiones sobre su implementación en una FPGA):

[2] De acuerdo a los diagramas de niveles de abstracción, deberán bajar un problema desde la capa de aplicación a la capa de compuertas lógicas

[3] `https://www.youtube.com/watch?v=QOQ1CHBjxIU`

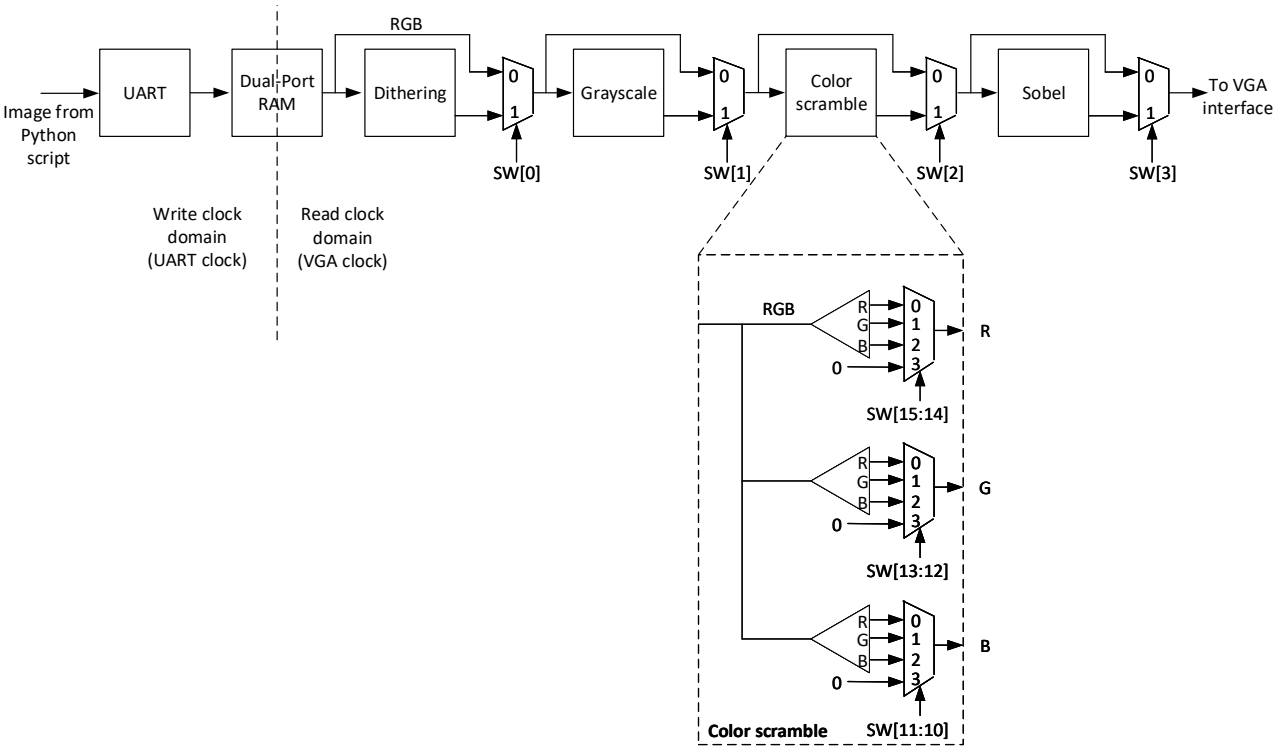

Figura 1: Diagrama de alto nivel de dispositivos distribuidos conectados por interfaz UART

- *Color scramble*: intercambia los valores de los canales de color de los pixeles. Por ejemplo, el canal R que va a la interfaz VGA puede transmitir el canal R original, el canal G, el canal B, o estar apagado, de acuerdo a la señal de selección del multiplexor conectado a la interfaz VGA. Este comportamiento es similar para cada uno de los canales, donde cada uno puede transmitir cualquier canal del pixel de entrada o estar apagado de acuerdo a la configuración de los switches.

- *Grayscale*: convierte una imagen a color en RGB a una imagen en escala de grises. Para esto se debe hacer una manipulación aritmética con los valores de los canales de color de cada pixel.

- *Dithering*: filtro que permite mejorar la presentación de una imagen en un sistema con paleta de colores reducida. La técnica consiste en aplicar una especie de ruido a una imagen para dar la impresión de contar con una mayor número de tonalidades de las que efectivamente están disponibles en el sistema. Por ejemplo, en un sistema que solo soporta blanco y negro (un bit de resolución de color), es posible aparentar distintos niveles de grises alternando pixeles blancos y negros con distinta separación entre ellos. Para una imagen a color el concepto es similar. Por ejemplo, al mostrar una imagen de 24 bits de color en un display que solo cuenta con 12 bits, se pueden utilizar los bits mas significativos para definir el color mas cercano en la paleta reducida – lo cual se conoce como *thresholding* – y usar los bits menos significativos para "redondear" el color (intuitivamente sería como aproximar un número con decimales a su parte entera).

- *Sobel*: filtro para detección de bordes en una imagen. Este filtro calcula el valor de cada pixel por medio de una suma ponderada de sus pixeles vecinos en el eje horizontal y vertical. Esto quiere decir que para calcular el valor de un pixel, se necesita saber el valor de los pixeles

vecinos de la línea anterior y también de la línea siguiente. Para esta parte deberá investigar acerca del operador Sobel, y definir estructuras de hardware o estrategias que le permitan realizar la operacion en forma eficiente, requiriéndose decisiones de diseño que deberá reportar y justificar.

Cada uno de los filtros anteriores se aplican en cascada, y pueden ser activados o desactivados mediante los switches.

En el repositorio del curso se encuentra disponible la carpeta `image-filters-example`, el cual contiene un script en Python que permite leer una imagen de entrada, la convierte a un formato y resolución adecuados, y transmite los pixeles a traves del puerto serial. En la FPGA deberá implementar la lógica que permita recibir los pixeles enviados desde su computador y almacenar los datos en el buffer. Esta carpeta además contiene una imagen de ejemplo y un archivo .bit que implementa parte de lo planteado anteriormente. Puede utilizar el .bit para programar su tarjeta, utilizar el script Python para cargar la imagen, y mover los switches de la tarjeta para ver los efectos de los distintos filtros (Sobel no esta incluido). Se recomienda revisar el codigo del script en Python para entender como se realiza la comunicacion serial, y asi poder modificarlo de ser necesario.

Para su diseño considere lo siguiente:

- Revise el manual para el uso de bloques de memoria basados en BRAM en el IP Catalog de Vivado. En particular, aprenda a configurar y utilizar el modo Dual-Port.
- Dimensione el tamaño máximo (ancho y largo) de la imagen de entrada que puede almacenar en la FPGA utilizando bloques BRAM. Basándose en las especificaciones del enunciado del problema, dimensione su memoria apuntando a almacenar el mayor número posible de pixeles que quepan en la FPGA. Una vez determinada la dimensión de la imagen, debe modificar el script en Python para que transforme la imagen de entrada a esta resolución, y adaptar la lógica de recepción y almacenamiento en la FPGA de forma acorde.
- Considere las operaciones aritméticas requeridas para traspasar una imagen color a escala de grises. ¿Cuales son los costos asociados a implementar multiplicaciones y divisiones en HDL? ¿Pueden aproximarse con versiones de bajo costo? Tome las decisiones de diseño adecuadas y aplíquelas a su implementación.
- Como concepto, el dithering es bastante utilizado y bien conocido; sin embargo, hay muchas formas de implementarlo, unas más complicadas y eficientes que otras. Hay bastante información y ejemplos en la red, incluyendo muchos métodos basados en la difusión de error [4]. Sin embargo, por simplicidad, se recomienda aplicar algún método basado en *ordered dithering*, los que operan a nivel local en cada pixel y por lo tanto son más sencillos de implementar en hardware. Como tip, una forma sencilla de realizar el dithering es utilizar los 4 bits mas significativos de cada canal como valor de thresholding, y utilizar 2 bits adicionales del color para "redondear" el valor de thresholding al valor más cercano utilizando la matriz correspondiente.

Se enfatiza que es extremadamente importante que planee muy bien su diseño antes de ponerse a escribir una línea de código HDL. Para esto, lo primero es estar seguro que entienda muy bien el problema. Se recomienda que haga pruebas en papel y lápiz para determinar los cálculos de los distintos algoritmos, además de implementar una versión en software (utilizando su lenguaje favorito) para entender el procesamiento y tener un punto de comparación. Notar que los algoritmos de procesamiento

[4] `http://www.tannerhelland.com/4660/dithering-eleven-algorithms-source-code/`

de imágenes planteados son muy básicos y por lo tanto muchos lenguajes ofrecen librerías que permiten realizar el procesamiento en forma muy eficiente con solo una líena de código. Sin embargo, el uso de estas librerías no le entrega información para implementarlo por su cuenta.

En cuanto a la implementación en hardware, recuerde el lema ***dividir y conquistar***. Identifique las componentes principales, implementales y verifíquelas en forma individual antes de integrarlas en el sistema completo. Por ejemplo, el primer paso para el procesamiento será asegurarse que puede recibir y mostrar la imagen en forma directa, sin ningún filtro en el medio. Solo una vez que tenga eso entendido, probado y verificado, proceda a agregar los filtros uno por uno. No intente agregar todos los filtros juntos sin antes verificar que cada uno funciona por separado. Los filtros de dithering y Sobel son los más complejos, y se recomienda dejarlo para el final.

Se espera que se encuentren con mucho problemas – lo cual es intencional – y deban tomar varias decisiones. Además, revisen cuidadosamente los reportes y warnings de los procesos de síntesis e implementación lógica. Si se reportan problemas de timing, su sistema puede no funcionar o funcionar aleatoriamente [5]. Además, dada la diversidad de tareas a realizar, es muy importante que trabajen en forma coordinada entre los integrantes de su grupo.

4. Evaluación

Las actividades para la sesión serán evaluadas en base a la siguiente tabla:

- Carga imagen por medio de la UART para que sea mostrada en pantalla sin modificaciones (15 puntos).
- Aplica filtro escala de grises (15 puntos).
- Aplica filtro color scrambler (15 puntos).
- Aplica filtro dithering (25 puntos).
- Aplica filtro Sobel (30 puntos).

Como siempre, la puntuación de cada actividad es todo o nada, y no hay puntajes intermedios por cosas que "casi funcionan". En esta ocasión la interrogación asociada a cada actividad será mas estricta y su diseño deberá funcionar en forma perfecta. No se revisarán implementaciones que contengan *artifacts* que no puedan ser debidamente justificados, como imagenes desalineadas, pixeles mal coloreados, agujeros de color, entre muchos otros que pueden surgir por un mal diseño u omisión a detalles en el flujo de procesamiento.

Las actividades solo recibirán puntaje si son completadas antes de finalizar la sesión correspondiente. Si no alcanza a terminar en la sesión correspondiente, aún debe completar la actividad aunque sin recibir puntaje como requisito de aprobación de la asignatura.

[5] Si encuentran problemas de este tipo, háganlo saber por Piazza o consulten a sus profesores y ayudantes.